



# **Aerial CUDA-Accelerated RAN**

*Release 24-3*

**NVIDIA Corporation**

**Feb 07, 2025**



# Contents

<b>1 Aerial cuBB</b>	<b>5</b>
1.1 Getting Started	5
1.1.1 Aerial cuBB Content Map	6
1.2 Product Brief	6
1.2.1 cuPHY Features Overview	6
1.2.1.1 Supported Features	6
1.2.2 Aerial CUDA-Accelerated RAN Features for 5G gNB	38
1.2.2.1 Highlights	38
1.2.2.2 Network, Services, and KPIs	42
1.2.3 Supported Systems	49
1.2.3.1 Highlights	49
1.2.3.2 Aerial CUDA-Accelerated RAN Overall Platform Qualification	49
1.2.4 Operations, Administration, and Management (OAM) Guide	49
1.2.4.1 OAM Operation	50
1.2.4.2 Fault Management	52
1.2.4.3 OAM Configuration	58
1.3 cuBB Release Notes	118
1.3.1 cuBB Software Manifest	118
1.3.1.1 Aerial CUDA-Accelerated RAN Software Manifest	118
1.3.1.2 Kubernetes Software Manifest	120
1.3.2 Supported Features and Configurations	120
1.3.2.1 PUSCH	120
1.3.2.2 PUCCH	121
1.3.2.3 PRACH	121
1.3.2.4 PDSCH	121
1.3.2.5 PDCCH	121
1.3.2.6 SS Block	121
1.3.2.7 CSI-RS	122
1.3.2.8 SRS	122
1.3.2.9 MIMO Features	122
1.3.2.10 LDPC Decoder	122
1.3.2.11 SHM Logger	122
1.3.3 Multicell Capacity	122
1.3.4 Supported Test Vector Configurations	125
1.3.4.1 PUSCH	125
1.3.4.2 PUCCH	127
1.3.4.3 PRACH	129
1.3.4.4 PDSCH	130
1.3.4.5 PDCCH	132
1.3.4.6 SS Block	134
1.3.4.7 CSI-RS	134
1.3.4.8 SRS	135
1.3.4.9 mSlot_mCell	137

1.3.4.10	LDPC Performance	137
1.3.5	SCF FAPI Support	142
1.3.5.1	Overview	142
1.3.5.2	SCF FAPI Messages Supported	142
1.3.5.3	Vendor Specific Message	146
1.3.5.4	Dynamic Beamforming for 64T64R	150
1.3.5.5	Static Beamforming for 64T64R	153
1.3.5.6	Additional Aerial Specific Error Codes Reported in ERROR.indication from L1 to L2	153
1.3.6	Limitations	154
1.3.6.1	Known Limitations	154
1.3.6.2	Known Issues	157
1.3.7	Acknowledgements	158
1.3.7.1	Abseil	158
1.3.7.2	Backward-cpp	161
1.3.7.3	BoringSSL	162
1.3.7.4	Benchmark	166
1.3.7.5	Bloaty	170
1.3.7.6	c-ares	174
1.3.7.7	CivetWeb	174
1.3.7.8	Data plane API	175
1.3.7.9	DPDK	178
1.3.7.10	Eigen	180
1.3.7.11	Fluent Helm Charts	185
1.3.7.12	Fmtlog	188
1.3.7.13	GDRCopy	189
1.3.7.14	Google APIs	189
1.3.7.15	GoogleTest	193
1.3.7.16	gRPC	193
1.3.7.17	libuv	198
1.3.7.18	LibYAML	199
1.3.7.19	Libyang	199
1.3.7.20	Mimalloc	200
1.3.7.21	Prometheus Client Library for Modern C++	200
1.3.7.22	Protocol Buffers	201
1.3.7.23	protoc-gen-validate (PGV)	202
1.3.7.24	RE2	205
1.3.7.25	UDPA API	206
1.3.7.26	zlib	210
1.3.7.27	CLI11	210
1.3.7.28	gsl-lite	211
1.3.7.29	cmake-modules	211
1.4	cuBB Installation Guide	212
1.4.1	Installing Tools on Grace Hopper MGX System	212
1.4.1.1	Supermicro Grace Hopper MGX Configuration	212
1.4.1.2	Cable Connection	213
1.4.1.3	System Firmware Upgrade	215
1.4.1.4	Install Ubuntu 22.04 Server	218
1.4.1.5	Configure the Network Interfaces	222
1.4.1.6	Disable Auto Upgrade	223
1.4.1.7	Install NVIDIA Optimized Ubuntu Kernel	224
1.4.1.8	Configure Linux Kernel Command-line	224
1.4.1.9	Apply the Changes and Reboot to Load the Kernel	225
1.4.1.10	Install Dependency Packages	225



1.4.1.11	Install DOCA OFED and Mellanox Firmware Tools on the Host . . . . .	225
1.4.1.12	Install CUDA Driver . . . . .	227
1.4.1.13	Install GDRCopy Driver . . . . .	228
1.4.1.14	Install Docker CE . . . . .	229
1.4.1.15	Install the Nvidia Container Toolkit . . . . .	229
1.4.1.16	Update BF3 BFB Image and NIC Firmware . . . . .	230
1.4.1.17	Install ptp4l and phc2sys . . . . .	233
1.4.1.18	Setup the Boot Configuration Service . . . . .	237
1.4.1.19	Running Aerial on Grace Hopper . . . . .	239
1.4.2	Installing Tools on Dell R750 . . . . .	239
1.4.2.1	Dell PowerEdge R750 Server Configuration . . . . .	240
1.4.2.2	BF3 NIC Installation . . . . .	240
1.4.2.3	Configure BIOS Settings . . . . .	240
1.4.2.4	Install Ubuntu 22.04 Server . . . . .	242
1.4.2.5	Disable Auto Upgrade . . . . .	242
1.4.2.6	Install the Low-Latency Kernel . . . . .	243
1.4.2.7	Configure Linux Kernel Command-line . . . . .	243
1.4.2.8	Apply the Changes and Reboot to Load the Kernel . . . . .	244
1.4.2.9	Disabling Nouveau . . . . .	244
1.4.2.10	Install Dependency Packages . . . . .	245
1.4.2.11	Install RSHIM and Mellanox Firmware Tools on the Host . . . . .	245
1.4.2.12	Install Docker CE . . . . .	247
1.4.2.13	Update BF3 BFB Image and NIC Firmware . . . . .	247
1.4.2.14	Set Persistent NIC Interface Name . . . . .	250
1.4.2.15	Install ptp4l and phc2sys . . . . .	251
1.4.2.16	Setup the Boot Configuration Service . . . . .	255
1.4.3	Installing and Upgrading Aerial cuBB . . . . .	257
1.4.3.1	Removing the Old Aerial cuBB Container . . . . .	257
1.4.3.2	Installing the New Aerial cuBB Container . . . . .	257
1.4.4	Aerial System Scripts . . . . .	257
1.4.4.1	System Configuration Validation Script . . . . .	257
1.4.5	Troubleshooting . . . . .	263
1.4.5.1	Hugepages Issues . . . . .	264
1.4.5.2	Remove Old CUDA Toolkit and Driver . . . . .	264
1.4.5.3	How to Fix Apt Update Error Due to Incorrect System Time . . . . .	264
1.4.5.4	How to Resize the Default LVM Volume . . . . .	264
1.4.5.5	How to Identify the NIC Interface Name and MAC Address . . . . .	266
1.5	cuBB Quickstart Guide . . . . .	266
1.5.1	cuBB Quickstart Overview . . . . .	266
1.5.2	Generating TV and Launch Pattern Files . . . . .	268
1.5.2.1	Using Aerial Python mcore Module . . . . .	268
1.5.2.2	Using Matlab . . . . .	270
1.5.3	Running Aerial cuPHY . . . . .	270
1.5.3.1	Building Aerial cuPHY . . . . .	270
1.5.3.2	Building and running on separate servers . . . . .	272
1.5.3.3	Running the cuPHY Examples . . . . .	272
1.5.3.4	Running cuPHY Performance Testing Scripts . . . . .	277
1.5.4	Running cuBB End-to-End . . . . .	279
1.5.4.1	Building the cuBB End-to-End . . . . .	281
1.5.4.2	nvlog Configuration . . . . .	283
1.5.4.3	Updating Configuration Files for End-to-End . . . . .	284
1.5.4.4	Running Environment Initialization for End-to-End . . . . .	287
1.5.4.5	Running Examples for End-to-End (SCF FAPI) . . . . .	288
1.5.4.6	Run in Test Mode (TM) . . . . .	315

1.5.4.7	Mixed O-RAN IOT Profiles (CAT-A-NoBF + CAT-A-DBF)	315
1.5.4.8	Mixed BFP9/BFP14	316
1.5.4.9	Mixed IQ data format for F08 Test Case	317
1.5.4.10	UL Measurements	318
1.5.4.11	Cell Life-Cycle Test	321
1.5.4.12	Terminate cuphycontroller Using a gRPC Message	322
1.5.4.13	Update M-plane Parameters Using gRPC Message	322
1.5.4.14	Dynamic PRACH Configuration and Init Sequence Test	334
1.5.4.15	Duplicate Configuration and Init Sequence Test	341
1.5.4.16	How to Get Aerial Metrics	343
1.5.4.17	Run an Additional Logging Stream Container	345
1.5.4.18	Run Multiple L2 Instances with Single L1 Instance	345
1.5.4.19	OAM Commands in Multiple L2 Instances	348
1.5.5	cuBB on NVIDIA Cloud Native Stack	348
1.5.5.1	Installation of NVIDIA Cloud Native Stack	348
1.5.5.2	Building Aerial Binary Container	351
1.5.5.3	Deploying Binary Container using Helm Chart	351
1.5.5.4	Theory of Operation	353
1.6	cuBB Integration Guide	356
1.6.1	cuMAC-CP	356
1.6.1.1	cuMAC-CP API Procedures	358
1.6.1.2	cuMAC-CP API Messages	362
1.6.1.3	L2 integration notes	375
1.6.1.4	Tests	375
1.6.2	NVIPC	378
1.6.2.1	NVIPC Overview	378
1.6.2.2	NVIPC Integration	384
1.7	cuBB Developer Guide	390
1.7.1	cuBB Software Architecture Overview	390
1.7.2	Aerial cuPHY Components	391
1.7.2.1	L2 Adapter	391
1.7.2.2	cuPHY Driver	392
1.7.2.3	FH Driver Library	392
1.7.2.4	cuPHY Controller	392
1.7.2.5	cuPHY	395
1.7.2.6	Running cuPHY Examples	434
1.7.3	Test MAC and RU Emulator Architecture Overview	436
1.7.4	5G MATLAB Models for Testing and Validation	437
1.7.4.1	Waveform compliance test	439
1.7.4.2	Test Vector Generation	439
1.7.4.3	PHY Performance Simulation	440
1.7.4.4	nrSim Configuration	440
1.7.4.5	nrSim Usage	443
1.7.4.6	Matlab Environment Preparation	443
1.7.4.7	External Configuration Mode (runSim)	444
1.7.4.8	Internal Configuration Mode (runRegression)	444
1.7.5	AI/ML Components for PUSCH Channel Estimation	446
1.7.5.1	How to Enable and Run the TrTEngine Model	446
1.7.5.2	TrTEngine YAML File Example	446
1.7.5.3	Example Python Script	447
1.7.6	References	449
1.8	Glossary	450

## 2 Aerial cuMAC

453

2.1	Getting Started with cuMAC	453
2.1.1	Data Flow	453
2.1.2	Quick Setup	454
2.1.2.1	Prerequisites	454
2.1.2.2	Getting and building cuMAC	455
2.2	cuMAC API Reference	456
2.2.1	cuMAC API Data Structures	456
2.2.1.1	CumacCellGrpPrms	456
2.2.1.2	cumacCellGrpUeStatus	460
2.2.1.3	cumacSchdSol	462
2.2.2	cuMAC Scheduler Module API	463
2.2.2.1	Multi-cell proportional-fairness UE down-selection	463
2.2.2.2	Multi-cell proportional-fairness PRB scheduler	464
2.2.2.3	Multi-cell layer selection	465
2.2.2.4	Multi-cell MCS selection + outer-loop link adaptation (OLLA)	465
2.3	Examples	466
2.3.1	cuMAC test vectors generated as HDF5 files	466
2.3.2	Single-TTI tests	467
2.3.3	Continuous-time tests	468
<b>3</b>	<b>Aerial Data Lake</b>	<b>473</b>
3.1	Target Audience	473
3.2	Key Features	473
3.3	Design	474
3.4	Installation	476
3.5	Usage	476
3.6	Multi-Cell	477
3.7	Using Data Lake in Notebooks	477
3.8	Database Administration	477
3.8.1	Database Import	478
3.8.2	Database Queries	479
3.8.3	Fresh Data	481
3.8.4	Dropping Data	481
3.8.5	Notes and Known Limitations	481
<b>4</b>	<b>pyAerial</b>	<b>483</b>
4.1	Overview	483
4.1.1	Key Features	484
4.1.2	Target Audience	484
4.1.3	Value Proposition	484
4.1.4	Release Notes	486
4.2	Getting Started with pyAerial	487
4.2.1	Pre-requisites	487
4.2.2	Installing pyAerial	487
4.2.3	Testing the installation	488
4.2.4	Running the example Jupyter notebooks	488
4.3	Examples of Using pyAerial	489
4.3.1	Running a PUSCH link simulation	489
4.3.1.1	Using pyAerial to run a PUSCH link simulation	489
4.3.2	LDPC encoding-decoding chain	498
4.3.2.1	Using pyAerial for LDPC encoding-decoding chain	498
4.3.3	Dataset generation by simulation	504
4.3.3.1	Using pyAerial for data generation by simulation	504
4.3.4	Dataset generation for LLRNet	511

4.3.4.1	LLRNet: Dataset generation . . . . .	512
4.3.5	LLRNet model training . . . . .	517
4.3.5.1	LLRNet: Model training and testing . . . . .	517
4.3.6	Neural receiver validation . . . . .	529
4.3.6.1	Using pyAerial to evaluate a PUSCH neural receiver . . . . .	530
4.3.7	Machine learning based channel estimation for 5G NR PUSCH . . . . .	539
4.3.7.1	Channel Estimation for Uplink Shared Channel (PUSCH) in pyAerial . . . . .	540
4.3.8	Channel estimation on transmissions captured using Aerial Data Lake . . . . .	559
4.3.8.1	Using pyAerial for channel estimation on Aerial Data Lake data . . . . .	560
4.3.9	Decoding PUSCH transmissions captured using Aerial Data Lake . . . . .	566
4.3.9.1	Using pyAerial for PUSCH decoding on Aerial Data Lake data . . . . .	566
4.4	API Reference . . . . .	584
4.4.1	Physical layer for 5G . . . . .	584
4.4.1.1	Receiver algorithms . . . . .	584
4.4.1.2	PDSCH . . . . .	598
4.4.1.3	PUSCH . . . . .	602
4.4.1.4	LDPC 5G . . . . .	605
4.4.1.5	Fading channel . . . . .	616
4.4.1.6	Parameter classes . . . . .	619
4.4.2	Utilities . . . . .	620
4.4.2.1	FAPI and Matlab interface utilities . . . . .	620
4.4.2.2	Data storing utilities . . . . .	621
4.4.2.3	CUDA utilities . . . . .	626
	<b>Python Module Index</b>	<b>627</b>
	<b>Index</b>	<b>629</b>

Aerial CUDA-Accelerated RAN brings together the Aerial software for 5G and AI frameworks and the NVIDIA accelerated computing platform, enabling TCO reduction and unlocking infrastructure monetization for telcos.

Aerial CUDA-Accelerated RAN has the following key features:

- ▶ Software-defined, scalable, modular, highly programmable and cloud-native, without any fixed function accelerators. Enables the ecosystem to flexibly adopt necessary modules for their commercial products.
- ▶ Full-stack acceleration of DU L1, DU L2+, CU, UPF and other network functions, enabling workload consolidation for maximum performance and spectral efficiency, leading to best-in-class system TCO.
- ▶ General purpose infrastructure, with multi-tenancy that can power both traditional workloads and cutting-edge AI applications for best-in-class RoA.

### **What's New in 24-3**

The following new features are available in release 24-3 for Aerial CUDA-Accelerated RAN:

- ▶ **Aerial cuPHY:** CUDA accelerated inline PHY
  - ▶ Multi-cell support for mMIMO (up to 3 cells)
  - ▶ Scheduling DL in special slots
  - ▶ Increase SRS slots in 4T4R and mMIMO
  - ▶ SRS CS multiplexing for different UEs
  - ▶ UL PUSCH channel estimation at PRG level
  - ▶ RKHS channel estimation
- ▶ **Aerial E2E:** System level / End-to-End validation
  - ▶ Fronthaul Port Failover Validation (Active-Standby) of C/U/S-Planes
  - ▶ Concluded Ch.8 Conformance testing with PRACH
  - ▶ MIG validation of AI + RAN
- ▶ **Aerial Redundancy/Resiliency:** CUDA accelerated RAN Redundancy/Resiliency features
  - ▶ RU Health Monitor - actively detect FH connectivity issues with ORU and take corrective action
  - ▶ Introduce L1 recovery period - If L1 is running late, drop FAPI messages for some time to allow L1 to recover
  - ▶ nvIPC pcap acquisition improvements - Introduced capability to add filters (cell-id , msg-id level) to nvIPC pcap acquisition
  - ▶ Backtrace output on console - Aerial prints backtrace on console in case of crash
- ▶ **Aerial cuMAC:** CUDA accelerated MAC scheduler
  - ▶ DRL MCS selection module
    - ▶ Pre-trained neural networks available under aerial\_sdk/cuMAC/testVectors
    - ▶ Inference based on TensorRT
  - ▶ 64TR MU-MIMO scheduler
    - ▶ UE sorting algorithm based on SRS SNR estimates

- ▶ UE grouping algorithm based on SRS channel coefficient estimates
- ▶ Aperiodic SRS resource manager
  - ▶ Combined with MU-MIMO UE sorting algorithm
- ▶ 4T4R system simulation with GPU-based TDL channel model
- ▶ Improved algorithms & CUDA implementation for type-0 and type-1 4T4R schedulers
- ▶ **pyAerial**: Python interface to Aerial cuPHY
  - ▶ CSI-RS transmission pipeline
  - ▶ RSRP and pre- and post-equalizer SINR estimation
  - ▶ Carrier frequency offset and timing advance estimation
  - ▶ CRC checking
  - ▶ OFDM fading channel simulation
  - ▶ Support of multiple UE groups for PUSCH receiver pipeline and its components
  - ▶ An improved API to PUSCH receiver pipeline and its components

### What's New in 24-2.1

The following new features are available in release 24-2.1 for Aerial CUDA-Accelerated RAN:

- ▶ **Aerial cuPHY**: CUDA accelerated inline PHY
  - ▶ 64T64R Massive MIMO:
    - ▶ 100 MHz DL max combined 16 layers + UL max combined 8 layers + SRS
    - ▶ 64T64R SRS + Dynamic + Static Beamforming Weights
    - ▶ Support multiple dynamic UE groups
    - ▶ Support flexible PRG size and PRB number
    - ▶ Support SRS buffer indexing from L2
    - ▶ Support non  $2^n$  layers
    - ▶ Use different section IDs when splitting the C-Plane section
    - ▶ FH messaging for CSIRS + PDSCH and other channel combinations
  - ▶ Support GH200+BF3 as RU emulator platform

### What's New in 24-2

The following new features are available in release 24-2 for Aerial CUDA-Accelerated RAN:

- ▶ **Aerial cuPHY**: CUDA accelerated inline PHY
  - ▶ MGX Grace Hopper multicell capacity w/ telco-grade traffic model
    - ▶ 20 peak loaded 4T4R @ 100MHz
    - ▶ Capacity also validated with more challenging traffic model
      - ▶ PUSCH and PDCCH symbols in the S-slot
  - ▶ L1-L2 interface enhancements
    - ▶ Separate FAPI request timelines for PDSCH and PDCCH
- ▶ **Aerial cuMAC**: CUDA accelerated MAC scheduler

- ▶ cuMAC-Sch
  - ▶ 4T4R CUDA implementation complete
- ▶ cuMAC-CP
  - ▶ 4T4R implementation (Functional – early access)
- ▶ **Aerial cuBB/E2E:** System level / End-to-End validation
  - ▶ Over-The-Air (OTA) validation:
    - ▶ CBRS O-RU
    - ▶ 8 UE OTA w/ 6 UE/TTI for > 8 hours
  - ▶ RedHat-OCP:
    - ▶ Multicell capacity validated on MGX (GH200+BF3)
  - ▶ O-RAN Fronthaul:
    - ▶ 16-bit fixed point IQ sample validated E2E (Keysight eLSU)
    - ▶ Simultaneous dual-port FH capability (8 peak cells; 4 per port)
  - ▶ L2 integration:
    - ▶ Multi-L2 container instances per L1 validated E2E
- ▶ **pyAerial:** Python interface to Aerial cuPHY
  - ▶ TensorRT inference engine
    - ▶ Jupyter notebook example using pyAerial to validate a neural PUSCH receiver
  - ▶ LDPC API improvements
    - ▶ Added soft outputs to LDPC decoder
  - ▶ LS channel estimation
  - ▶ Limited support for Grace Hopper
    - ▶ Run pyAerial together with Aerial Data Lakes





---

# Chapter 1. Aerial cuBB

The NVIDIA cuBB SDK provides GPU accelerated 5G signal processing pipeline including cuPHY for Layer 1 PHY, cuMAC for L2 scheduler, delivering unprecedented throughput and efficiency by keeping all the processing within the high-performance GPU memory.

Aerial cuBB is a software-defined, scalable, modular, highly programmable and cloud-native, without any fixed function accelerators. Enables the ecosystem to flexibly adopt necessary modules for their commercial products.

Aerial cuBB has the following key components:

- ▶ **cuPHY**: L1 library of the Aerial CUDA-Accelerated RAN. It is designed as an inline accelerator to run on NVIDIA GPUs and it does not require any additional hardware accelerator.
- ▶ **cuMAC**: L2 MAC Scheduler library of the Aerial CUDA-Accelerated RAN for accelerating 5G/6G MAC layer scheduler functions with NVIDIA GPUs.

## 1.1. Getting Started

Aerial cuBB brings together the Aerial software for 5G and AI frameworks and the NVIDIA accelerated computing platform, enabling TCO reduction and unlocking infrastructure monetization for telcos.

## 1.1.1. Aerial cuBB Content Map

The following table describes the different sections of the Aerial cuBB developer documentation.

Section	Description
<i>Product Brief</i>	Provides an in-depth exploration of the standards and configurations supported by Aerial cuBB.
<i>cuBB Release Notes</i>	Outlines what Aerial cuBB features and settings have changed in the most recent release of Aerial CUDA-Accelerated RAN.
<i>cuBB Installation Guide</i>	Describes how to install Aerial cuBB.
<i>cuBB Quickstart</i>	Describes how to run cuBB software examples.
<i>cuBB Integration Guide</i>	Provides reference information related to the CUDA RAN MAC Scheduler Control Plane (cuMAC-CP) and NVIPC messaging standard.
<i>cuBB Developer Guide</i>	Provides reference information for the cuPHY library software stack.

## 1.2. Product Brief

This section includes an in-depth exploration of the standards and configurations supported by Aerial cuBB.

### 1.2.1. cuPHY Features Overview

This section provides an overview of supported features in cuPHY.

#### 1.2.1.1 Supported Features

##### 1.2.1.1.1 Aerial CUDA-Accelerated RAN Layer 1

Aerial CUDA-Accelerated RAN adheres to 3GPP Release 15 standard specifications to deliver the necessary Layer 1 capabilities for a gNB.

### 1.2.1.1.1.1 3GPP Release 15

Aerial cuPHY adheres to 3GPP Release 15 standard specifications to deliver the following capabilities for gNB Layer 1.

Overall PHY capabilities include:

- ▶ Error detection on the transport channel and indication to higher layers
- ▶ FEC encoding/decoding of the transport channel
- ▶ Hybrid ARQ soft-combining
- ▶ Rate matching of the coded transport channel to physical channels
- ▶ Mapping of the coded transport channel onto physical channels
- ▶ Power weighting of physical channels
- ▶ Modulation and demodulation of physical channels including:
  - ▶ Frequency and time synchronization
  - ▶ Radio characteristics measurements and indication to higher layers
  - ▶ Multiple Input Multiple Output (MIMO) antenna processing
  - ▶ Transmit Diversity (TX diversity)
  - ▶ Digital and Analog Beamforming
  - ▶ RF processing

### 1.2.1.1.2 PHY FH Interface

#### 1.2.1.1.2.1 Aerial CUDA-Accelerated RAN PHY Overall Capabilities

Features	Configuration	Supported
Standard support	3GPP 5G NR Rel 15	P
Duplexing Mode	TDD	Y
Narrow Bandwidth (MHz)	30MHz, 40 MHz, 50MHz, 80 MHz	P
Channel Bandwidth (MHz)	100 MHz	Y
Subcarrier Spacing (kHz)	30kHz	Y
Maximum Number of Subcarriers (Max number of RBs x Num of Subcarriers per RB) = 273 x 12	3276	Y
Downlink Waveform	CP-OFDM	Y
Uplink Waveform	CP-OFDM	Y

continues on next page

Table 1 – continued from previous page

<b>Features</b>	<b>Configuration</b>	<b>Supported</b>
	DFT-s-OFDM (for data and control)	Y
	Configurable to DFT-s-OFDM (for data & Control)	Y
Number of Downlink SU-MIMO layers	Up to 4	Y
Number of Uplink SU-MIMO layers	1, 2	Y
Number of Tx physical antennas	1	N
	2	Y
	4	Y
	8	N
	64	Y
Number of Rx physical antennas	1	N
	2	Y
	4	Y
	8	N
	64	Y
Slot format	DDDSUUDDDD S = 6:4:4 (DL: G: UL)	Y
Carrier Aggregation	Configurable component carriers	Y
Configurable BW Parts	Up to 4	Y
BBU-RRU split option	7.1	Y
	7.2	Y
	8	N
Maximum Downlink throughput per user (Mbps) 4T4R configuration	1870	Y
Maximum Uplink throughput per user (Mbps) 4T4R configuration	467	Y

### 1.2.1.1.3 TS 38.211 Numerologies, Physical Resources, Modulation, Sequence, Signal Generation

#### 1.2.1.1.3.1 Aerial CUDA-Accelerated RAN PHY Numerologies

Feature	Configuration	Supported
Numerologies:Normal CP	$\mu=0$ :SCS=15kHz, 14symbol/slot, 10slot/frame, 1slot/subframe, Normal CP	N
	$\mu=1$ :SCS=30kHz, 14symbol/slot, 20slot/frame, 2slot/subframe, Normal CP	Y
	$\mu=2$ :SCS=60kHz, 14symbol/slot, 40slot/frame, 4slot/subframe, Normal CP	N
	$\mu=3$ :SCS=120kHz, 14symbol/slot, 80slot/frame, 8slot/subframe, Normal CP	N
	$\mu=4$ :SCS=240kHz, 14symbol/slot, 160slot/frame, 16slot/subframe, Normal CP	N
Numerologies:Extended CP	$\mu=2$ :SCS=60kHz, 12symbol/slot, 40slot/frame, 4slot/subframe, Extended CP	N

#### 1.2.1.1.3.2 Aerial CUDA-Accelerated RAN Overall PHY Physical Resources

Feature	Supported
Antenna Ports	Y
Resource Grid	Y
Resource Elements	Y
Resource Block	Y
Resource Block - Common Resource Block(CRB)	Y
Resource Block - Physical Resource Block(PRB)	Y
Resource Block - Virtual Resource Block (VRB)	Y
Bandwidth Part (BWP) Dynamically adapt the carrier bandwidth and numerology in which a UE operates A bandwidth part is a subset of contiguous common resource blocks for a given numerology $\mu_i$ in bandwidth part $i$ on a given carrier. A UE can be configured with up to four bandwidth parts in UL and DL	Y

1.2.1.1.3.3 Aerial CUDA-Accelerated RAN PHY Physical Resources – BWP

Feature	Supported
Bandwidth Part (BWP) Dynamically adapt the carrier bandwidth and numerology in which a UE operates A bandwidth part is a subset of contiguous common resource blocks for a given numerology $\Pi_i$ in bandwidth part $i$ on a given carrier A UE can be <b>configured</b> with up to <b>four</b> bandwidth parts in <b>both</b> UL and DL	Y
Default Aerial CUDA-Accelerated RAN startup configuration to not use BWP, can be enabled to support BWP on a per carrier basis (while cell OOS)	N
Default Aerial CUDA-Accelerated RAN startup configuration to not use BWP, can be enabled to support BWP on a per carrier basis at startup	N

1.2.1.1.3.4 Aerial CUDA-Accelerated RAN Overall Carrier Aggregation

Feature	Description	Supported (emulated)
Carrier Aggregation	Transmissions in multiple cells can be aggregated to support inter-band and intra-band configurations	Y
100MHz	Up to 2 cells aggregation(1CC,2CC)	Y
	Up to 4 cells aggregation(1CC,2CC,3CC, 4CC)	Y
Narrowband Carrier Aggregation (ZMhz)	Configurable upto 4 component carriers	Y

1.2.1.1.3.5 Aerial CUDA-Accelerated RAN PHY Modulation Mapper

Modulation Scheme	Supported
Pi/2 BPSK	Y
BPSK	Y
QPSK	Y
16QAM	Y
64QAM	Y
256QAM	Y

### 1.2.1.1.3.6 Aerial CUDA-Accelerated RAN PHY Sequence Generation

Feature	Description	Supported
Sequence Generation	Pseudo-random sequence generation Generic pseudo-random sequences are defined by a length-31 Gold sequence	Y
	Low-PAPR sequence generation type 1	Y
	Low-PAPR sequence generation type 2	Y

### 1.2.1.1.3.7 OFDM Baseband Signal Generation (UL DFT-S-OFDM)

Feature	Configuration	Supported
Signal generation for all channels except PRACH & RIM-RS		RU support expected
PRACH		RU support expected
RIM-RS		RU support expected
Uplink waveform Support concurrent UE configuration to use CP-OFDM or DFT-S-OFDM on same cell.	DFT-S-OFDM for UL. Some specific parameters: <ul style="list-style-type: none"> <li>▶ Support for PUSCH and for PUCCH format 3</li> <li>▶ Support 0.5 pi-BPSK for Modulation</li> <li>▶ Support DMRS group hopping</li> <li>▶ Support DMRS sequence hopping</li> </ul>	Y

1.2.1.1.4 TS 38.211 Channels

1.2.1.1.4.1 Aerial CUDA-Accelerated RAN Physical Overall Channels and Reference Signals

Category		L1 requirement	Supported
Downlink Channels (TX )		PDSCH processing	Y
		PDCCH processing	Y
		PBCH processing	Y
Downlink signals (TX )		DMRS for PDSCH	Y
		DMRS for PDCCH	Y
		DMRS for PBCH	Y
		PSS, SSS	Y
		CSI-RS, TRS	Y
		PT-RS	N
Downlink Physical Resources		Antenna ports starting with 1000 for PDSCH	Y
		Antenna ports starting with 2000 for PDCCH	Y
		Antenna ports starting with 3000 for channel-state information reference signals	Y
		Antenna ports starting with 4000 for SS/PBCH block transmission	Y
Uplink Channels (RX )		PUSCH processing	Y
		PUCCH processing	Y
		PRACH processing	Y
Uplink signals (RX )		DMRS for PUSCH	Y
		DMRS for PUCCH	Y
		SRS	Y
		PT-RS	N
Uplink physical Resources		Antenna ports starting with 0 for PUSCH and associated demodulation reference signals	Y
		Antenna ports starting with 1000 for SRS	Y
		Antenna ports starting with 2000 for PUCCH	Y
		Antenna port 4000 for PRACH	Y



## 1.2.1.1.4.2 Aerial CUDA-Accelerated RAN Overall Channel - PUSCH (Physical Uplink Shared Channel)

Features	Configuration	Supported
Number of codewords	1	Y
Scrambling		Y
Modulation schemes	Pi/2-BPSK	Y
	QPSK	Y
	16 QAM	Y
	64 QAM	Y
	256 QAM	Y
PUSCH transform precoding mode	Disable	Y
	Enable	Y
Precoding	Implemented in UE for UL	Y
HARQ process	Number of HARQ process = 1	Y
HARQ process	Maximum number of HARQ process is 16	Y
Mapping to virtual resource blocks		Y
VRB to PRB mapping Type	Non-interleaved	Y
	Interleaved	N
Transmission Mode	SU-MIMO up to 4 layers	Y
	MU-MIMO up to 8 layers	Y
PUSCH DMRS CDM group without data	PUSCH DMRS CDM group without data 1	Y
	PUSCH DMRS CDM group without data 2	Y
PUSCH users per TTI	16	Y
Uplink algorithm	UL HARQ control	Y
	UL Channel Estimation LS	Y
	MRC, MMSE for equalizer	Y
	IRC, MMSE for equalizer	Y
	Frequency Offset Correction	Y
Rate Matching	I_LBRM = 1 (Limited Buffer Rate Matching)	Y
	I_LBRM = 0 (Limited Buffer Rate Matching)	Y

Aerial CUDA-Accelerated RAN Overall Channel - PUCCH (Physical Uplink Control Channel)

<b>Format</b>	<b>Configuration</b>	<b>Supported</b>
Format	0	Y
	1	Y
	2	Y
	3	Y
	4	N
UCI sched coding, AFC, DFT (Format 1)		N
Modulation schemes	Pi/2-BPSK, BPSK, QPSK	Y
Scheduling Request SR	Support needed	Y
Group hopping	neither	Y
	disable	Y
	enable	Y
Sequence cyclic shift	Zadoff-Chu sequence	Y
Intra-slot Frequency hopping/second hop PRB	Support	Y
Inter-slot Frequency hopping/second hop PRB	Support	Y
PUCCH over multiple slots	Number of slots - 2,4,8	N
Frequency Offset Correction	PUCCH format 1, 3	N
Multi-UE support	24 UEs / TTI	Y
PUCCH UCI HARQ-ACK Polar	codeblock CB size < 359, liftsize = 8	Y

### 1.2.1.1.4.3 1-Capabilities-TSx211-6-3-3] Aerial CUDA-Accelerated RAN Overall Channel - PRACH(PHY Random Access Channel)

Feature	Configuration	Supported
Format	A1	N
	A2	N
	A3	N
	B1	N
	B2	N
	B3	N
	B4	Y
	C0	N
	C2	N
	0	N
	1	N
	2	N
	3	N
	Subcarrier Spacing (kHz)	1.25
5		N
15		N
30		Y
Sequence cyclic shift	Zadoff-Chu sequence	Y
Preamble length	839	N
	139	Y
Number of PRACH occasions per TTI	4 FDM	Y
Contention based Random Access	Configurable non-contention based Random Access	N

## 1.2.1.1.4.4 Aerial CUDA-Accelerated RAN Overall PHY - UL Reference Signals

## PUSCH

Signal	Configuration	Supported
PUSCH DMRS sequence generation when transform precoding is disabled		Y
PUSCH DMRS sequence generation when transform precoding is enabled	Neither group, nor sequence hopping is enabled	Y
	Group hopping is enabled and sequence hopping is disabled	Y
	Sequence hopping is enabled and group hopping is disabled	Y
Demodulation reference signal for PUSCH Mapping to physical resources	DM-RS configuration type 1	Y
	DM-RS configuration type 2	N
	UL-DMRS-max-len=1	Y
	UL-DMRS-max-len=2	Y
	UL-DMRS-add-pos=0	Y
	UL-DMRS-add-pos=1	Y
	UL-DMRS-add-pos=2	Y
	UL-DMRS-add-pos=3	Y
Phase-tracking reference signals for PUSCH Sequence generation	transform precoding is not enabled	N
	transform precoding is enabled	N
Phase-tracking reference signals for PUSCH Mapping to physical resources	transform precoding is disabled	N
	transform precoding is enabled	N

## PUCCH

Signal	Configuration	Supported
Demodulation reference signal for PUCCH format 1	no intra-slot frequency hopping	Y
	intra-slot frequency hopping enabled	Y
Demodulation reference signal for PUCCH format 2		Y
Demodulation reference signal for PUCCH format 3 (Format 4 not supported)	No additional DM-RS, No hopping	Y
	No Additional DM-RS, hopping	Y
	Additional DM-RS, No hopping	Y
	Additional DM-RS, hopping	Y

## SRS

Signal	Configuration	Supported
Sounding reference signal resource	Antenna ports=1, 1OFDM symbols	Y
	Antenna ports=1, 2OFDM symbols	Y
	Antenna ports=1, 4OFDM symbols	Y
	Antenna ports=2, 1OFDM symbols	Y
	Antenna ports=2, 2OFDM symbols	Y
	Antenna ports=2, 4OFDM symbols	Y
	Antenna ports=4, 1OFDM symbols	Y
	Antenna ports=4, 2OFDM symbols	Y
Sounding reference signal Sequence generation	KTC=2	Y
	KTC=4	Y
	KTC=8	Y
Sounding reference signal Mapping to physical resources	CSRS=0~63	Y
Sounding reference signal slot configuration	Indicated by higher layer parameter SRS-Config	Y

## PTRS

Signal	Configuration	Supported
PTRS		N

**1.2.1.1.4.5 Aerial CUDA-Accelerated RAN Overall Channel - PDSCH(PHY DL Shared Channel)**

Feature	Configuration	Supported
Scrambling		Y
Modulation schemes	QPSK	Y
	16 QAM	Y
	64 QAM	Y
	256 QAM	Y
Transmission Mode	4T4R SU-MIMO up to 4 layers	Y
	64T64R MU-MIMO up to 16 layers	Y
Number of codewords	1	Y
	2	N
Number of antenna ports	1000 - 1011	Y
Number of physical antennas	4	Y
	64	Y
Beam Forming weights computation	BF m2	N
Precoding	non-codebook	Y
	pre-coding weight	Y
	Type I Single-Panel Codebook	N
	Type I Multi-Panel Codebook	N
	Type II Codebook	N
	Type II Port Selection Codebook	N
PDSCH mapping type	Type A	Y
	Type B	Y
Resource allocation type	Type 0 (4T4R only)	Y
	Type 1	Y
VRB to PRB mapping Type	Non-interleaved	Y
	Interleaved	N
PDSCH DMRS CDM groups without data	1	Y
	2	Y

continues on next page

Table 3 – continued from previous page

<b>Feature</b>	<b>Configuration</b>	<b>Supported</b>
	3	N/A
Number PDSCH users per TTI	16	Y
Power Control	PDSCH	Y
	DMRS - PDSCH	Y

## 1.2.1.1.4.6 Aerial CUDA-Accelerated RAN Overall Channel - PDCCH (Physical DL Control Channel)

Feature	Configuration	Supported
Scrambling	Up to 2 codewords	N
CORESET	Normal	Y
	RMSI CORESET	Y
SSB - RMSI CORESET multiplexing pattern	Pattern 1	Y
Aggregation Level	1	Y
	2	Y
	4	Y
	8	Y
	16	Y
Modulation schemes	QPSK	Y
Layer mapping	Supported	Y
Antenna port mapping	Supported	Y
Mapping to virtual resource blocks	Supported	Y
Mapping from virtual to physical resource blocks	Non-interleaved VRB-to-PRB mapping	Y
Polar code	Block length up to 128 bits	Y
DMRS (Demodulation Reference Signal)	m-sequence	Y
CCE To REG Mapping Type	Non-interleaved	Y
	Interleaved	Y
Number OFDM symbol of CORESET	1	Y
	2	Y
	3	Y
Power Control	PDCCH	Y
	DMRS-PDCCH	Y
DCI format	0_0	NA
	0_1	NA
	1_0	NA
	1_1	NA
	2_x	NA
Precoding	Precoding Matrix Idx based precoding in the DU	Y



## 1.2.1.1.4.7 Aerial CUDA-Accelerated RAN Overall Channel - PBCH (Physical Broadcast Channel)

	Configuration	cuBB Tested
Precoding		Y
Scrambling	SS/PBCH block index Lmax=4	N
	SS/PBCH block index Lmax=8	N
	SS/PBCH block index Lmax=64	N
Modulation schemes	QPSK	Y
Mapping to Physical Resources		Y
DMRS Support	Support	Y
DMRS config type	Type 1	Y
	Type 2	N
DMRS type A Pos	Pos2	Y
	Pos3	Y
DMRS max length	1	Y
	2	Y
DMRS Additional Position	Pos0	Y
	Pos1	Y
	Pos2	Y
	Pos3	Y

## 1.2.1.1.4.8 Aerial CUDA-Accelerated RAN Overall - PHY DL Reference Signals

## PDSCH

Feature	Configuration	Supported
Demodulation reference signals for PDSCH Sequence generation		Y
Demodulation reference signals for PDSCH Mapping to physical resources	DM-RS configuration type 1	Y
	DM-RS configuration type 2	N
	DL-DMRS-max-len=1	Y
	DL-DMRS-max-len=2	Y
	DL-DMRS-add-pos=0	Y
	DL-DMRS-add-pos=1	Y
	DL-DMRS-add-pos=2	Y
	DL-DMRS-add-pos=3	Y
Phase-tracking reference signals (PTRS) for PDSCH Mapping to physical resources	LPT-RS=1	N
	LPT-RS=2	N
	LPT-RS=4	N

### PDCCH

Feature	Configuration	Supported
Demodulation reference signals for PDCCH Sequence generation		Y
Demodulation reference signals for PDCCH Mapping to physical resources		Y

### PBCH

Feature	Configuration	Supported
Demodulation reference signals for PBCH Sequence generation		Y
Demodulation reference signals for PBCH Mapping to physical resources		Y
CSI reference signals		
CSI reference signals	Zero-power	Y
	non-zero-power	Y
CSI reference signals Sequence generation	nID equals the higher-layer parameter ScramblingID	Y

continues on next page

Table 4 – continued from previous page

Feature	Configuration	Supported
CSI reference signals Mapping to physical resources	Row 1: 1 port, Density = 3, CDMtype = No CDM	Y
	Row 2: 1 port, Density = 1, 0.5, CDM-type = No CDM	Y
	Row 3: 2 port, Density = 1, 0.5, CDM-type = FD-CDM2	Y
	Row 4: 4 port, Density = 1, CDMtype = FD-CDM2	Y
	Row 5: 4 port, Density = 1, CDMtype = FD-CDM2	Y
	Row 6: 8 port, Density = 1, CDMtype = FD-CDM2	Y
	Row 7: 8 port, Density = 1, CDMtype = FD-CDM2	Y
	Row 8: 8 port, Density = 1, CDMtype = CDM4 (FD2, TD2)	Y
	Row 9: 12 port, Density = 1, CDMtype = FD-CDM2	N
	Row 10: 12 port, Density = 1, CDM-type = CDM4 (FD2, TD2)	N
	Row 11: 16 port, Density = 1, 0.5, CDMtype = FD-CDM2	N
	Row 12: 16 port, Density = 1, 0.5, CDMtype = CDM4 (FD2, TD2)	N
	Row 13: 24 port, Density = 1, 0.5, CDMtype = FD-CDM2	N
	Row 14: 24 port, Density = 1, 0.5, CDMtype = CDM4(FD2, TD2)	N
	Row 15: 24 port, Density = 1, 0.5, CDMtype = CDM8(FD2, TD4)	N
	Row 16: 32 port, Density = 1, 0.5, CDMtype = FD-CDM2	N
	Row 17: 32 port, Density = 1, 0.5, CDMtype = CDM4(FD2, TD2)	N
	Row 18: 32 port, Density = 1, 0.5, CDMtype = CDM8(FD2, TD4)	N

RIM

Feature	Configuration	Supported
RIM reference signal General	The first RIM-RS type can be used to convey information	N
	The second RIM-RS type depends on configuration only	N
RIM reference signal Sequence generation		N
RIM reference signal Mapping to physical resources		N
RIM reference signal RIM-RS configuration	Enough Indication is disabled	N
	Enough Indication is enabled	N
Positioning Reference		
Positioning reference signal Sequence generation		N
Positioning reference signal Mapping to physical resources	LPRS = 2, Kcomb = 2	N
	LPRS = 4, Kcomb = 2	N
	LPRS = 6, Kcomb = 2	N
	LPRS = 12, Kcomb = 2	N
	LPRS = 4, Kcomb = 4	N
	LPRS = 12, Kcomb = 4	N
	LPRS = 6, Kcomb = 6	N
	LPRS = 12, Kcomb = 6	N
	LPRS = 12, Kcomb = 12	N

### Synchronization Signals

Feature	Configuration	Supported
SSB numerology	30 kHz	Y
SSB precoding	supported	Y
SSB burst set configuration	2 SS blocks w/ single SSB burst set	Y
Synchronization signal generation	PSS generation and mapping to physical resources	Y
	SSS generation and mapping to physical resources	Y
SS/PBCH block	Mapping of PSS within an SS/PBCH block	Y
	Mapping of SSS within an SS/PBCH block	Y
	Mapping of PBCH and DM-RS within an SS/PBCH block	Y
	Time-frequency structure and time location of an SS/PBCH block	Y

### 1.2.1.1.5 TS 38.212 Multiplexing and Channel Coding

#### 1.2.1.1.5.1 Aerial CUDA-Accelerated RAN Overall Multiplexing and Channel Coding

Feature	Configuration	Supported
General Procedures	CRC calculation All CRC len supported (6, 11, 16, 24)	Y
	Code block segmentation and code block CRC attachment <ul style="list-style-type: none"> <li>▶ Polar coding</li> <li>▶ Low density parity check coding</li> </ul>	Y
Transport to physical channel mapping - UL	UL-SCH -> PUSCH	Y
	RACH -> PRACH	Y
	UCI -> PUCCH,PUSCH	Y
Transport to physical channel mapping - DL	DL-SCH -> PDSCH	Y
	BCH -> PBCH	Y
	PCH -> PDSCH	Y
	DCI -> PDCCH	Y
Channel coding schemes	Polar coding	Y
	Low density parity check coding (LDPC)	Y

continues on next page

Table 5 – continued from previous page

Feature	Configuration	Supported
	Channel coding of small block lengths	Y
Rate matching	Rate matching for Polar code	Y
	Rate matching for LDPC code	Y
	Rate matching for channel coding of small block lengths	Y
Code block concatenation	sequentially concatenating the rate matching outputs for the different code blocks <ul style="list-style-type: none"> <li>▶ LDPC</li> <li>▶ Polar Coding</li> </ul>	Y
uplink transport channels and control information	Random access channel	Y
	Uplink shared channel <ul style="list-style-type: none"> <li>▶ LDPC graph selection</li> <li>▶ Rate Matching</li> <li>▶ Code block concatenation</li> <li>▶ Data &amp; Control Multiplexing</li> </ul>	Y
	Uplink control information <ul style="list-style-type: none"> <li>▶ Uplink control information on PUCCH</li> <li>▶ Uplink control information on PUSCH</li> </ul>	Y
downlink transport channels and control information	Broadcast channel	Y
	Downlink shared channel and paging channel	Y
	Downlink control information <ul style="list-style-type: none"> <li>▶ DCI formats</li> <li>▶ CRC attachment</li> <li>▶ Channel coding</li> </ul>	Y
UCI multiplexing on PUCCH	support muxing mode as per 38.212 - 6.3.1.1	Y

## 1.2.1.1.6 TS 38.213 Physical Layer Procedures for Control

## 1.2.1.1.6.1 Aerial CUDA-Accelerated RAN Overall - PHY Control Procedures

## UE procedures (Not applicable to base station)

Category	L1 requirement	Supported
Synchronization procedures	Cell search	NA
	Transmission timing adjustments	NA
	Timing for secondary cell activation / deactivation	NA
Radio link monitoring	SSB based	NA
	CSI-RS based	NA
Link recovery procedures	radio link failure	NA
	beam failure recovery	NA
Uplink power control	Physical uplink shared channel	NA
	Physical uplink control channel	NA
	Sounding reference signal	NA
	Physical random access channel	NA
	Power ramping counter suspension	NA
	Dual connectivity	NA
	Power headroom report	NA

## PHY RACH

Category	L1 requirement	Supported
	Type-1 random access procedure	Y
	Type-2 random access procedure	N

## UE procedures (Not applicable to base station)

Category	L1 requirement	Supported
HARQ-ACK codebook determination	CBG-based HARQ-ACK codebook determination	NA
	Type-1 HARQ-ACK codebook determination in physical up-link control channel	NA
	Type-1 HARQ-ACK codebook determination in physical up-link shared channel	NA
	Type-2 HARQ-ACK codebook determination in physical up-link control channel	NA
	Type-2 HARQ-ACK codebook determination in physical up-link shared channel	NA
	Type-3 HARQ-ACK codebook determination	NA

### UCI reporting on PUSCH

Category	L1 Requirement	Supported
Short block codes for UCI	Input: 1 - 11 bits output 32 bits	Y
Multiplexing of coded UCI bits to PUSCH	CSI part 1, support maximum 48 bit	Y
	CSI part 1 and CSI part 2, support maximum 48 bit	Y
	Decoding UCI on PUSCH with PUSCH data (UCI-ON-PUSCH scaling) 0.5/0.65/0.8/1	N
	Decoding UCI on PUSCH without PUSCH data (UCI-ON-PUSCH scaling) 0.5/0.65/0.8/1	N
	HARQ information length maximum 128	Y
	Semi-static offset	N
	Dynamic offset	N

### UCI Reporting on PUCCH

Category	L1 Requirement	Supported
UCI reporting on PUCCH	PUCCH Resource Sets before RRC connection establishment	N
	PUCCH Resource Sets for RRC connected UE	N
	UE procedure for reporting multiple UCI types	N
	PUCCH repetition procedure	N



**UE Procedures (Not applicable to base station)**

Category	L1 Requirement		Supported
UE procedure for determining physical downlink control channel assignment	Type0-PDCCH search space	common	NA
	Type0A-PDCCH search space	common	NA
	Type1-PDCCH search space	common	NA
	Type2-PDCCH search space	common	NA
	Type3-PDCCH search space	common	NA
	UE-specific search space		NA

**UE Procedure for Receiving Control Information**

Category	L1 Requirement	Supported
PDCCH validation for DL SPS and UL grant Type 2		NA
PDCCH validation for DL SPS and UL grant Type 2		NA
PDCCH monitoring indication and dormancy/non-dormancy behaviour for SCells		NA
Search space set group switching		NA
HARQ-ACK information for PUSCH transmissions		NA

**UE-Group Common Signaling**

Category	L1 Requirement	Supported
UE-group common signalling	Slot configuration	N
	UE procedure for determining slot format	N
	Interrupted transmission indication	N
	Cancellation indication	N
	Group TPC commands for PUCCH/PUSCH	N
	SRS switching	N

**Bandwidth Part Operation**

Category	L1 Requirement	Supported
BWP	Configurable upto 4	Y
	Set of resource blocks and slot symbols of CORE-SET for Type0-PDCCH search space set when {SS/PBCH block, PDCCH} SCS is {15, 15} kHz for frequency bands with minimum channel bandwidth 5 MHz or 10 MHz	N
	Set of resource blocks and slot symbols of CORE-SET for Type0-PDCCH search space set when {SS/PBCH block, PDCCH} SCS is {15, 15} kHz for frequency bands operated with shared spectrum channel access	N
	Set of resource blocks and slot symbols of CORE-SET for Type0-PDCCH search space set when {SS/PBCH block, PDCCH} SCS is {15, 30} kHz for frequency bands with minimum channel bandwidth 5 MHz or 10 MHz	N
	Set of resource blocks and slot symbols of CORE-SET for Type0-PDCCH search space set when {SS/PBCH block, PDCCH} SCS is {30, 15} kHz for frequency bands with minimum channel bandwidth 5 MHz or 10 MHz	N
	Set of resource blocks and slot symbols of CORE-SET for Type0-PDCCH search space set when {SS/PBCH block, PDCCH} SCS is {30, 30} kHz for frequency bands with minimum channel bandwidth 5 MHz or 10 MHz	N
	Set of resource blocks and slot symbols of CORE-SET for Type0-PDCCH search space set when {SS/PBCH block, PDCCH} SCS is {30, 30} kHz for frequency bands operated with shared spectrum channel access	N
	Set of resource blocks and slot symbols of CORE-SET for Type0-PDCCH search space set when {SS/PBCH block, PDCCH} SCS is {30, 15} kHz for frequency bands with minimum channel bandwidth 40MHz	N
	Set of resource blocks and slot symbols of CORE-SET for Type0-PDCCH search space set when {SS/PBCH block, PDCCH} SCS is {30, 30} kHz for frequency bands with minimum channel bandwidth 40MHz	N
	Set of resource blocks and slot symbols of CORE-SET for Type0-PDCCH search space set when {SS/PBCH block, PDCCH} SCS is {120, 60} kHz	N

continues on next page

Table 7 – continued from previous page

Category	L1 Requirement	Supported
	Set of resource blocks and slot symbols of CORE-SET for Type0-PDCCH search space set when {SS/PBCH block, PDCCH} SCS is {120, 120} kHz	N
	Set of resource blocks and slot symbols of CORE-SET for Type0-PDCCH search space set when {SS/PBCH block, PDCCH} SCS is {240, 60} kHz	N
	Set of resource blocks and slot symbols of CORE-SET for Type0-PDCCH search space set when {SS/PBCH block, PDCCH} SCS is {240, 120} kHz	N
	Parameters for PDCCH monitoring occasions for Type0-PDCCH CSS set - SS/PBCH block and CORESET Multiplexing pattern 1 and FR1	N
	Parameters for PDCCH monitoring occasions for Type0-PDCCH CSS set - SS/PBCH block and CORESET Multiplexing pattern 1 and FR2	N
	PDCCH monitoring occasions for Type0-PDCCH CSS set - SS/PBCH block and CORESET Multiplexing pattern 2 and {SS/PBCH block, PDCCH} SCS {120, 60} kHz	N
	PDCCH monitoring occasions for Type0-PDCCH CSS set - SS/PBCH block and CORESET Multiplexing pattern 2 and {SS/PBCH block, PDCCH} SCS {240, 120} kHz	N
	PDCCH monitoring occasions for Type0-PDCCH CSS set - SS/PBCH block and CORESET Multiplexing pattern 3 and {SS/PBCH block, PDCCH} SCS {120, 120} kHz	N
Integrated access-backhaul operation		N
Dual active protocol stack based handover		N

## 1.2.1.1.7 TS 38.214 Physical Layer Procedures for Data

## 1.2.1.1.7.1 Aerial CUDA-Accelerated RAN Overall PHY Data Procedures

Category	L1 Requirement	Supported
<b>UL PUSCH Procedures</b>		
Transmission Scheme	Codebook-based	Y
	Non-codebook-based	Y
Resource allocation	Type 0 (4T4R only)	N
	Type 1	Y
Modulation order, redundancy version and transport block size determination		Y
Code block group based PUSCH transmission		N
MCS Table	Table64QAM	Y
	Table256QAM	Y
	Table64QAMLowSE	Y
PUSCH mapping type	Type A	Y
	Type B	Y
CBG retransmission bitmap	Enable	N
	Disable	Y

### 1.2.1.1.8 FH Interfaces

#### 1.2.1.1.8.1 Aerial CUDA-Accelerated RAN Overall 4T4R L1 - L2 Layer Interface Based on SCF FAPI

Feature	Configuration (10.02)	Supported (Emulated)
<b>SCF control interface must support the following messages</b>		
Config.request	4T4R	Y
Config.response	4T4R	Y
Start.request	4T4R	Y
Stop.request	4T4R	Y
Stop.indication	4T4R	Y
Error.indication	4T4R	Y
Param.request (cap query)	4T4R	Y
Param.response	4T4R	Y
<b>SCF data interface includes the following messages</b>		
DL_TTI.request	4T4R	Y
UL_TTI.request	4T4R	Y
UL_DCI.request	4T4R	Y
SLOT errors	4T4R	Y
TX_Data.request	4T4R	Y
Rx_Data.indication	4T4R	Y
CRC.indication	4T4R	Y
UCI.indication	4T4R	Y
SRS.indication	4T4R	Y
RACH.indication	4T4R	Y

#### 1.2.1.1.8.2 Aerial CUDA-Accelerated RAN Overall PHY FH Interface

Feature	Description	Supported
IOT Profiles	<p><b>Simultaneous</b> support of TDD profile(s) and TDD pattern on single GPU</p> <ul style="list-style-type: none"> <li>▶ NR TDD IOT Profile 1: NR-TDD-FR1-CAT-A-NoBF</li> <li>▶ NR TDD IOT Profile 2: NR-TDD-FR1-CAT-A-DBF</li> </ul>	Y
O-RAN CUS plane features with fronthaul 7.2-x split: [10][11]	<p><b>Simultaneous</b> O-RU category support on same GPU/DU</p> <ul style="list-style-type: none"> <li>▶ CAT-A (precoding supported for PDCSH)</li> <li>▶ CAT-B</li> </ul>	Y
Beamforming	<ul style="list-style-type: none"> <li>▶ Predefined beamID based beamforming</li> </ul>	Y
IQ compression & bit-width	<p><b>Simultaneous support for</b></p> <ul style="list-style-type: none"> <li>▶ Static-bit-width Fixed point IQ (14 bit)</li> <li>▶ BFP IQ Compression (9 bit)</li> </ul>	Y
O-DU timing	<ul style="list-style-type: none"> <li>▶ Defined transport delay method</li> </ul>	Y
Synchronization	<ul style="list-style-type: none"> <li>▶ G8275.1 (full timing support)</li> <li>▶ LLS-C3 with PTP + SyncE</li> </ul>	Y
Transport features	<ul style="list-style-type: none"> <li>▶ eCPRI</li> <li>▶ Application layer fragmentation</li> <li>▶ QoS over fronthaul</li> </ul>	Y
Section types	<ul style="list-style-type: none"> <li>▶ Section Type 1 (DL/UL channels)</li> <li>▶ Section Type 3 (PRACH)</li> <li>▶ Multiple sections within a single C-plane message</li> </ul>	Y

continues on next page

Table 8 – continued from previous page

Feature	Description	Supported
Digital power scaling	<ul style="list-style-type: none"> <li>▶ UL gain correction</li> <li>▶ DL reference level adjustment</li> </ul>	Y
Rx window monitoring,	Counters like <ul style="list-style-type: none"> <li>▶ Data received too early</li> <li>▶ Data received too late</li> <li>▶ Data received on-time</li> </ul>	Y
Scale	Support for upto 8 peak - 16 avg 100Mhz carriers	Y

### 1.2.1.1.9 Measurements

#### 1.2.1.1.9.1 Aerial CUDA-Accelerated RAN Overall PHY Measurements - 4T4R

##### PUSCH measurements

Measurements	Supported Config	Supported cuBB Tested
RSS	4T4R	Y
RSRP	4T4R	Y
Pn+I pre-eq (Noise+Interference power)	4T4R	Y
Pn+I post-eq (Noise+Interference power)	4T4R	Y
SINR pre-eq	4T4R	Y
SINR post-eq	4T4R	Y
Timing Advance	4T4R	Y

##### PUCCH measurements

Measurements	Supported Config	Supported cuBB Tested
PUCCH Format 0	4T4R	Y
PFO RSS	4T4R	Y
PFO RSRP	4T4R	Y

continues on next page

Table 10 – continued from previous page

Measurements	Supported Config	Supported cuBB Tested
PF0 Pn+i	4T4R	Y
PF0 timing advance	4T4R	Y
PUCCH Format 1	4T4R	Y
PF1 RSS	4T4R	Y
PF1 RSRP	4T4R	Y
PF1 Pn+i	4T4R	Y
PF1 timing advance	4T4R	Y
PUCCH Format 2	4T4R	Y
PF2 RSS	4T4R	Y
PF2 RSRP	4T4R	Y
PF2 Pn+i	4T4R	Y
PF2 timing advance	4T4R	Y
PUCCH Format 3	4T4R	Y
PF3 RSS	4T4R	Y
PF3 RSRP	4T4R	Y
PF3 Pn+i	4T4R	Y
PF3 timing advance	4T4R	Y
PUCCH Format 4	4T4R	N
PF4 RSS	4T4R	N
PF4 RSRP	4T4R	N
PF4 Pn+i	4T4R	N
PF4 timing advance	4T4R	N

**PRACH measurements**

Measurements	Supported Config	Supported cuBB Tested
Pn+i (Noise+Interference power)	4T4R	Y
Preamble signal strength	4T4R	Y

**SRS measurements**



Measurements	Supported Config	Supported cuBB Tested
SNR	4T4R	Y
Received signal strength	4T4R	Y
Timing advance	4T4R	Y

### All channels measurements

Measurements	Supported Config	Supported cuBB Tested
Both pre-equalization and post-equalization across all channels should be configurable and supported	4T4R	N

### 1.2.1.1.10 TS 38.104 (base station radio Tx and Rx) Base Station (BS) Radio Transmission and Reception

#### 1.2.1.1.10.1 Aerial CUDA-Accelerated RAN Overall PHY Performance Conformance

Feature	Configuration	Supported
<b>PUSCH</b>		
PUSCH with transform precoding disabled	4T4R	Y
PUSCH with transform precoding enabled	4T4R	Y
UCI multiplexed on PUSCH	4T4R	Y
<b>PUCCH</b>		
DTX to ACK probability	4T4R	N
Performance requirements for PUCCH format 0	4T4R	N
Performance requirements for PUCCH format 1	4T4R	N
Performance requirements for PUCCH format 2	4T4R	N
Performance requirements for PUCCH format 3	4T4R	N
Performance requirements for PUCCH format 4	4T4R	N
Performance requirements for multi-slot PUCCH	4T4R	N
<b>PRACH</b>		
Performance requirements for PRACH	PRACH False alarm probability	N
	PRACH detection requirements	N

## 1.2.2. Aerial CUDA-Accelerated RAN Features for 5G gNB

The 5G gNB capabilities, procedures, and interfaces have dependencies on Aerial CUDA-Accelerated RAN PHY Layer. The purpose of this section is to ensure that the Aerial CUDA-Accelerated RAN provides support for gNB capabilities, procedures, and interfaces.

### 1.2.2.1 Highlights

- ▶ PUCCH Format 1 I+N and SINR, DTX for UCI on PUSCH
- ▶ Predefined BeamId support
- ▶ Foxconn O-RU support
- ▶ Cell life cycle management
- ▶ 4T4R TDD 7 beam support
- ▶ 8-port CSI-RS
- ▶ Dynamic OAM supporting out-of-service updates:
  1. Dest MAC and VLAN ID
  2. exponent\_dl
  3. dl\_iq\_data\_fmt
  4. ul\_iq\_data\_fmt
  5. exponent\_ul
  6. max\_amp\_ul
  7. section\_3\_time\_offset
  8. pusch\_prb\_stride
  9. prach\_prb\_stride
  10. fh\_len\_range
  11. lower\_guard\_bw
  12. gps\_alpha (Shared across cells)
  13. gps\_beta (Shared across cells)
  14. prachRootSequenceIndex
  15. prachZeroCorrConf
  16. numPrachFdOccasions
  17. restrictedSetConfig
  18. prachConfigIndex
  19. K1
- ▶ Fronthaul Extension to 50km
- ▶ Simultaneous fronthaul ports for higher fronthaul bandwidth
- ▶ Multiple BandWidth Part (BWP) support
- ▶ 4T4R TDD bandwidth: 10MHz, 30MHz, 40MHz, 50MHz and 80MHz

- ▶ Carrier aggregation:
  1. 100MHz + 80MHz
  2. 100MHz + 40MHz
  3. 80MHz + 40MHz
  4. 100MHz + 80MHz + 40MHz
- ▶ L1 startup time within 30 seconds
- ▶ Support for multiple L2 on a single converged card
- ▶ Cell-Id starts from 0 for all pods

### 1.2.2.1.1 Capabilities

#### 1.2.2.1.1.1 Homogeneous Cell Lifecycle Mgmt - Cell State Mgmt (IS/OOS)

Feature	Supported
Support cell activation and de-activation. This is commonly referred to as taking a carrier OOS (Out of Service) and bringing it to IS (In Service) states	Y

#### 1.2.2.1.1.2 Fronthaul Port Failover Validation (Active-Standby) of C/U/S-Planes

Feature	Supported
Supports switching to secondary FH port within same BF3 card on primary FH port failure, transitioning C/U/S-Plane traffic from primary port to secondary FH port without stopping L1	Y

1.2.2.1.2 Procedures

1.2.2.1.2.1 Aerial CUDA-Accelerated RAN Overall Beam and Carrier Mobility

Feature	Configuration	Supported
Inter-gNB Handover	UE moves from 1 gNB to another gNB <ul style="list-style-type: none"> <li>▶ UL RRC transfer</li> <li>▶ UE Context Modification Request/Response</li> <li>▶ UE Context Release</li> <li>▶ Serving and Target gNB cells can support different frequencies</li> </ul>	N
Intra-DU Handover	Cell-level mobility: UE establishes new connection to new carrier (inter-cell) supported by UE context modification procedure <ul style="list-style-type: none"> <li>▶ UE Context Modification Request/Response</li> <li>▶ UE Context Release</li> <li>▶ Serving and Target Cells can support different frequencies</li> </ul>	Y
Beam Mobility	UE establishes data path to new beam within carrier coverage (intra-cell)	N
Mobility at low speeds	Aerial CUDA-Accelerated RAN shall support pedestrian mobility by modeling the 3GPP channels and 38.104 requirements	N
Mobility at vehicular speeds	Aerial CUDA-Accelerated RAN shall support mobility at high vehicular speeds - up to 70mph (Doppler Shift = 400Hz)	N

### 1.2.2.1.2.2 UL Power Control

Feature	Description	Supported
Single UE Power Control	BS initiated power control for single UEs	Y
UE Group Power Control	BS initiated power control for UE groups	Y

### 1.2.2.1.2.3 Carrier Aggregation

Feature	Description	Supported
Carrier Aggregation	Transmissions in multiple cells can be aggregated to support <b>inter-band and intra-band configurations</b>	Y
100MHz	Up to 2 cells aggregation (1CC,2CC), intra-band contiguous	Y
	Up to 2 cells aggregation (1CC,2CC), intra-band non-contiguous	Y
	Up to 4 cells aggregation(1CC,2CC,3CC,4CC), inter-band non contiguous	Y
Narrowband Carrier Aggregation (ZMhz)	Configurable up to 4 component carriers	Y

### 1.2.2.1.3 Interfaces

#### 1.2.2.1.3.1 gNB Interfaces

Interface	Supported
NG Interface (TS 38.410)	Y
Xn interface (TS 38.420)	N
F1 interface (TS 38.470)	Y
E1 interface (TS 38.460)	N
Front Haul interface - ORAN 7.2 Split (CUS version 3)	Y
E2 interface	N
O1 interface	N

### 1.2.2.2 Network, Services, and KPIs

This section includes E2E integration configuration and KPIs for appropriate NEs across 5G RAN, CN, and 5G infrastructure.

#### 1.2.2.2.1 Highlights

- ▶ 4 Peak Cells validated in eCPRI/ RF cabled setup. 8 Average cells (50% traffic) validated in eCPRI setup
- ▶ 4 DL Layers and 2 UL Layers validated in 4T4R configuration
- ▶ 6 UE/TTI Validated
- ▶ Simultaneous Front Haul capability supported.
- ▶ Multi L2 also validated with each L2 supporting different cells.
- ▶ 1 Cell OTA verified

#### 1.2.2.2.2 E2E Summary

- ▶ **4 Peak Cells in E2E configuration (CN + RAN + UE-EM) via eCPRI connection to test equipment**  
(Achieving aggregate DL throughput of 5.8Gbps and aggregate UL throughput of 840Mbps)
- ▶ **4 Peak Cell in E2E configuration (CN + RAN + UE-EM) via RF cable connection to O-RU**  
(Achieving aggregate DL throughput of 5.72Gbps and aggregate UL throughput of 800Mbps)
- ▶ **1 Cell OTA in E2E configuration (CN + RAN + CUE) via OTA connection to UE device**  
(Achieving DL throughput of 955Mbps and UL throughput of 105Mbps)
- ▶ **1 Cell OTA testing in Bands n78 and CBRS**

## 1.2.2.2.3 4T4R Overall Configuration and KPIs

Feature	Configuration
Release 15 SA	TDD 7.2 CatA
Subcarrier spacing (SCS)	30kHz
sub-6 frequency spectrum	n78 Germany (3700 - 3800 MHz)
sub-6 frequency spectrum	n48 US CBRS (3550 - 3700 MHz)
Channel bandwidth	100 Mhz
MIMO Layers support	DL : 4 layer UL : 2 layer
100MHz cells	Up to 4 peak cells Up to 8 average cells (50%)
Peak throughput per cell	DL : 1.46 Gbps per cell UL : 210 Mbps per cell
Number of RRC Connected UEs per cell (Cabled Environment)	100
Number of RRC Connected UEs per cell (OTA Environment)	8
Number of UEs/TTI	DL : 16 UE/TTI UL : 16 UE/TTI
Frame structure and slot format	DDDSUDDDD S = 6:4:4 (DL: G: UL)
	DSUUU
	DDDSU
User plane latency (RRC connected mode)	10ms one way for DL and UL
Synchronization and Timing support	IEEE 1588v2 PTP / SyncE ORAN LLS-C3
MTU size	1500 / 8192 bytes
Modulation	256 QAM DL 256 QAM UL
Soak Testing	8 hours

1.2.2.2.4 Aerial CUDA-Accelerated RAN Overall ORU Ecosystem

ORU	Configuration	Freq Band
Foxconn RPQN-7801E	4T4R	3.7GHz - 3.8GHz (indoors)
Fujitsu TA08029-B059	4T4R	3.6GHz - 3.7GHz
Foxconn RPON-7800	4T4R	3.7GHz - 3.8GHz (outdoors)
Foxconn RPQN-4800E	4T4R	CBRS 3.55GHz - 3.7GHz, (indoors)

1.2.2.2.5 Aerial CUDA-Accelerated RAN Overall UE Ecosystem

UE	Configuration
Camera FourFaith Camera F-SC241-216-5G	SU-MIMO 4DL, 1UL
Camera FourFaith Camera F-SC241-216-5G (EU)	SU-MIMO 4DL, 2UL
CUE OnePlus Nord 5G AC2003 EU/UK Model	SU-MIMO 4DL, 1UL
CUE Oppo Reno 5G	SU-MIMO 4DL, 1UL
CUE Samsung S22	SU-MIMO 4DL, 1UL
CUE Samsung S23	SU-MIMO 4DL, 1UL
CUE Xiaomi 13 pro	SU-MIMO 4DL, 2UL
CUE Google Pixel 8	SU-MIMO 4DL, 1UL
Programmable UE NI X410 with OAI stack	SU-MIMO 4DL, 1UL



### 1.2.2.2.6 5G Infrastructure Integration

#### 1.2.2.2.6.1 5G RAN Integration

Function	Features	Supported
gNB	<p>Baseband functions for</p> <ul style="list-style-type: none"> <li>▶ signal processing using multiple antennas</li> <li>▶ signal processing for detecting and correcting errors in the wireless transmission</li> <li>▶ signal processing to ensure that the wireless transmission is secure</li> <li>▶ managing the wireless resources efficiently between different devices in the network</li> </ul>	Y
O-RU	<p>Radio functions to convert digital information into signals that can be transmitted wirelessly, ensuring that the transmitted signals are in the right frequency bands and have the correct power levels.</p> <p>Includes antennas which radiate the electrical signals into radio waves</p>	Y
UE	<p>End user devices such as smartphones, routers, tablets, HMDs, CPEs</p>	Y

## 1.2.2.2.6.2 5G Mobile Core (NGC) integration

Function	Features	Supported
AMF Core Access and Mobility Management Function	Connection and reachability management, mobility management, access authentication and authorization, location services	Y
SMF Session Management Function	UE session, including IP address allocation, selection of associated UP function, control aspects of QoS, and control aspects of UP routing.	Y
PCF Policy Control Function	Manage policy rules that other CP functions then enforce.	Y
UDM Unified Data Management	Manage user identity, including generation of authentication credentials.	Y
AUSF Authentication Server Function	Essentially an authentication server	Y
UDR Unified Data Repository	Repository of subscriber information that can be used by other microservices. For example UDM	Y
NCHF New Charging Function	Cover all the network's needs of charging and interaction with billing systems	Y
CP - SDSF Structured Data Storage	"Helper" service used to store structured data.	Y
CP - UDSF Unstructured Data Storage	"helper" service used to store unstructured data.	Y
CP - NEF Network Exposure Function	Expose select capabilities to third-party services, including translation between internal and external representations for data. Could be implemented by an "API Server" in a microservices-based system.	N
CP - NRF NF Repository Function	A means to discover available services.	N
CP - NSSF Network Slicing Selector Function	A means to select a Network Slice to serve a given UE. Network slices are essentially a way to partition network resources in order to differentiate service given to different users.	N
UP - UPF User Plane Function	Forwards traffic between RAN and the Internet. In addition to packet forwarding, it is responsible for policy enforcement, lawful intercept, traffic usage reporting, and QoS policing	Y

### 1.2.2.2.7 5G NSE Overall Network Deployment Topologies

Topology	Configuration	Supported
On Prem Isolated Island	Co-located gNB + CN + MEC applications	Y
Colocated 5G infra with low latency MEC applications + centralized 5GC	MEC applications + gNB + UPF with centralized 5G CN (CUPS support - with SBA and to minimize latency in user plane)	N
Campus Distributed MEC applications (latency tolerant)	Campus Distributed MEC applications + colocated (gNB + UPF + CN) - (Non latency sensitive applications can be distributed and leverage an existing enterprise network data stream)	Y
CUPS Architecture Support		N

1.2.2.2.7.1 Aerial E2E Reference BOM and Component Manifest

5G Infra Component	HW and SW Revision Manifest	Supported
gNB	SMC Grace Hopper MGX Serve with BF3 NIC	Y
	Dell PowerEdge R750 Server with A100X	Y
	Altran L2+	Y
CN	Dell PowerEdge R750 Server	Y
	Altran CN	Y
FH Switch	Dell PowerSwitch S5248F-ON	Y
	Adva switch FSP 150 XG400	Y
	Spectrum switch SN3750X	Y
	Ciena switch 5164	Y
GM	Cisco switch N9K-C93180YC-FX3S	Y
	QULSAR Qg 2 Multi-Sync Gateway	Y
Cables	Dell C2G 1m LC-LC 50/125 Duplex Multimode OM4 Fiber Cable - Aqua - 3ft – Optical patch cable	Y
	NVIDIA MCP1650-V01AE30 DAC Cable Ethernet 200GbE QSFP28 1.5m	Y
	NVIDIA MCP1600-C001E30N DAC Cable Ethernet 100GbE QSFP28 1m	Y
	Beyondtech 5m (16ft) LC UPC to LC UPC Duplex OM3 Multimode PVC (OFNR) 2.0mm Fiber Optic Patch Cable	Y
PDUs	CableCreation 3ft Cat5/Cat6 Ethernet Cables	Y
	Tripp Lite 1.4kW Single-Phase Monitored PDU with LX Platform Interface, 120V Outlets (8 5-15R), 5-15P, 12ft Cord, 1U Rack-Mount, TAA	Y
Transceivers	Finisar SFP-to-RJ45 Transceiver	Y
	Intel Ethernet SFP+SR Optics	Y
	Dell SFP28-25G-SR Transceiver	Y
Ethernet Switch	Netgear ProSafe Plus JGS524E Rackmount	Y

## 1.2.3. Supported Systems

Aerial cuPHY is a software-defined workload hosted on NVIDIA-certified EGX servers and a stack that uses the CUDA OS platform and GPU/NIC/CPU firmware and toolkits. This section highlights the Aerial cuPHY workload configuration interdependencies as part of the NVIDIA platform stack.

### 1.2.3.1 Highlights

- ▶ Grace Hopper MGX system supports 20 4T4R Peak cells / 20 4T4R average BFP9 cells
- ▶ Supports Massive MIMO: 64T64R (16DL | 8UL) @ 100MHz w/ SRS-based Beamforming

### 1.2.3.2 Aerial CUDA-Accelerated RAN Overall Platform Qualification

System	Configuration	Applications
Grace Hopper MGX	<ul style="list-style-type: none"> <li>▶ 72-core NVIDIA Grace CPU</li> <li>▶ NVIDIA H100 Tensor Core GPU</li> <li>▶ 480GB of LPDDR5X memory with ECC</li> <li>▶ Supports 96GB of HBM3</li> <li>▶ BF3 NIC x2</li> </ul>	gNB, RU emulator
x86 platform	Dell R750 <ul style="list-style-type: none"> <li>▶ Server Skew 10-AYCG</li> <li>▶ Intel Xeon Gold 6336Y 2.4G, 24C/48T</li> <li>▶ PCIe Gen4</li> <li>▶ Memory 512GB DDR4</li> <li>▶ Storage 2TB</li> <li>▶ BF3 NIC</li> </ul>	RU emulator only

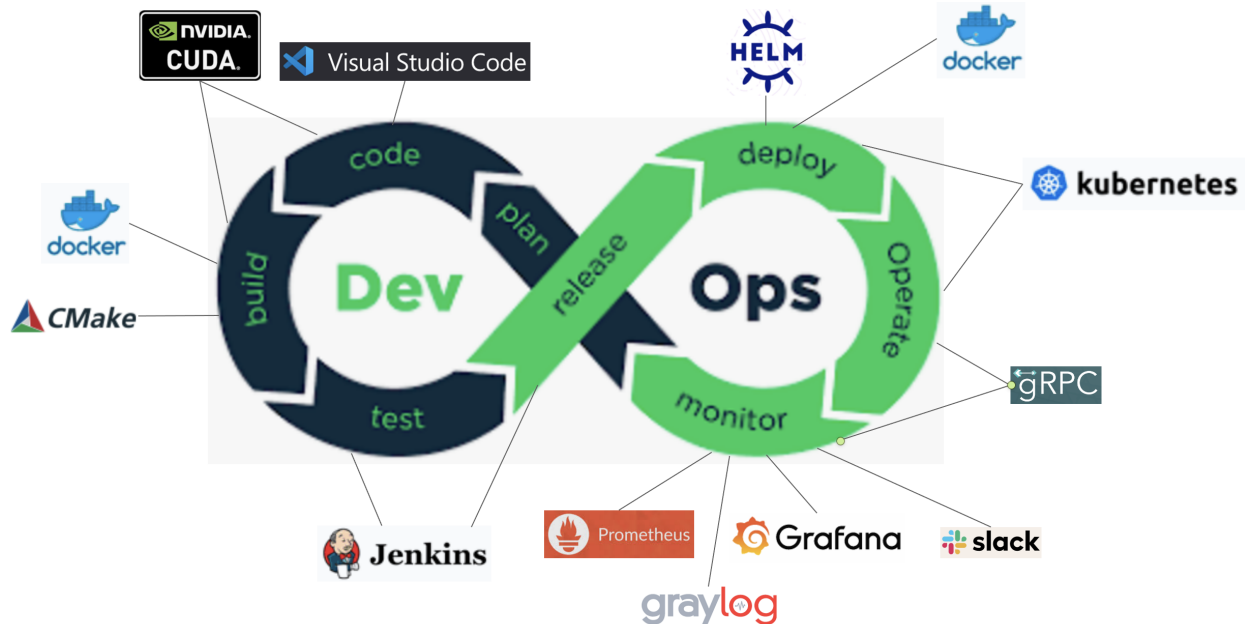
## 1.2.4. Operations, Administration, and Management (OAM) Guide

The Operations, Administration, and Management (OAM) guide covers Aerial OAM capabilities that include startup configuration using YAML configuration files, run-time configuration and status using remote procedure calls, high performance logging, and metrics reporting using the Prometheus framework.

### 1.2.4.1 OAM Operation

#### 1.2.4.1.1 Cloud Native DevOps

Aerial CUDA-Accelerated RAN is based on cloud-native principles and supports a DevOps work-flow using industry standard tools such as Kubernetes, gRPC, and Prometheus.



#### 1.2.4.1.2 Aerial Applications

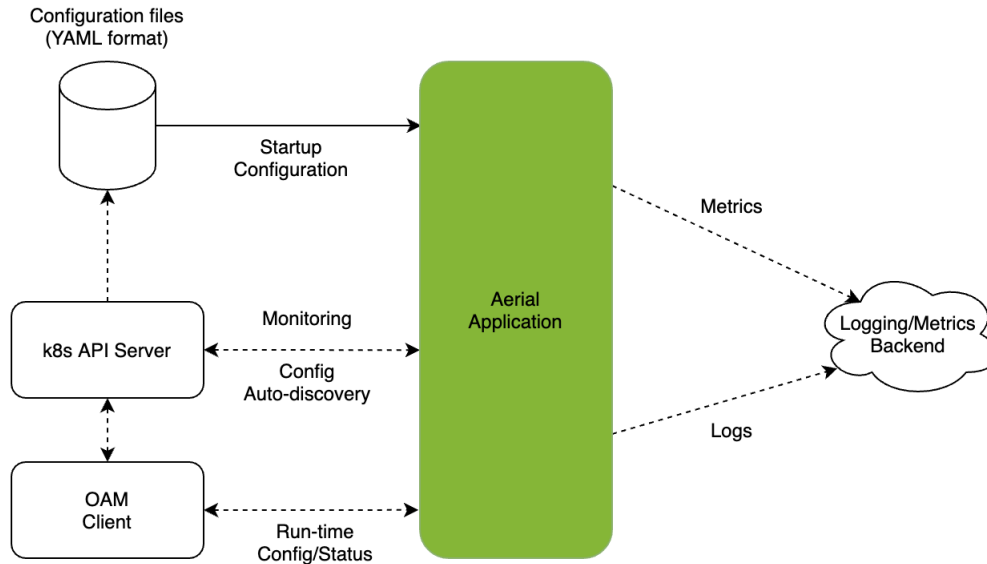
The Aerial framework includes three primary applications for end to end L1 implementation and testing.

- ▶ **cuphycontroller** is the full L1 stack application. This application implements the adaptation layer from L2 to the cuPHY API, orchestrates the cuPHY API scheduling, and sends/receives ORAN compliant Fronthaul traffic over the NIC. Several independently configurable adaptation layers from L2 to the cuPHY API are available.
- ▶ **test\_mac** application, for integration testing, implements a mock L2 that is capable of interfacing with cuphycontroller over the L2/L1 API.
- ▶ **ru-emulator** application, for integration testing, implements a mock O-RU + UE that is capable of interfacing with cuphycontroller over the ORAN compliant Fronthaul interface.

Every Aerial application supports the following:

- ▶ Configuration at startup through the use of YAML-format configuration files.
- ▶ Support for optionally-configured cloud-based logging and metrics backends.
- ▶ Support for optionally-deployed OAM clients for run-time configuration and status queries.
- ▶ When deployed as a Kubernetes pod:
  - ▶ Support for application monitoring and configuration auto-discovery through the Kubernetes API.

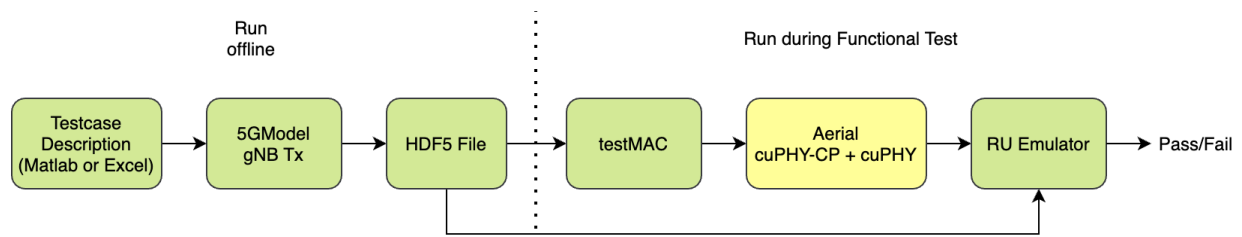
- ▶ Configuration YAML files can optionally be mounted as a Kubernetes ConfigMap, separating the container image from the configuration.
- ▶ Configuration YAML files can optionally be templated using the Kubernetes kustomization.yaml format,



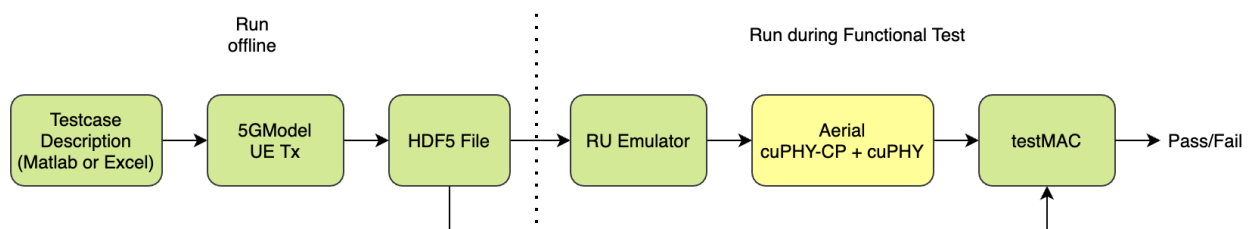
### 1.2.4.1.3 Deployment Scenarios

#### 1.2.4.1.3.1 Functional Testing

For real-time functional correctness testing, test cases are generated offline in HDF5 binary file format, then played back in real-time through the testMAC and RU Emulator applications. The Aerial cuPHY-CP + cuPHY components under test, run in real-time to exercise GPU and Fronthaul Network interfaces. Test case sequencing is enabled through configurable launch pattern files read by testMAC and RU Emulator. The diagram below shows an example of downlink functional testing:

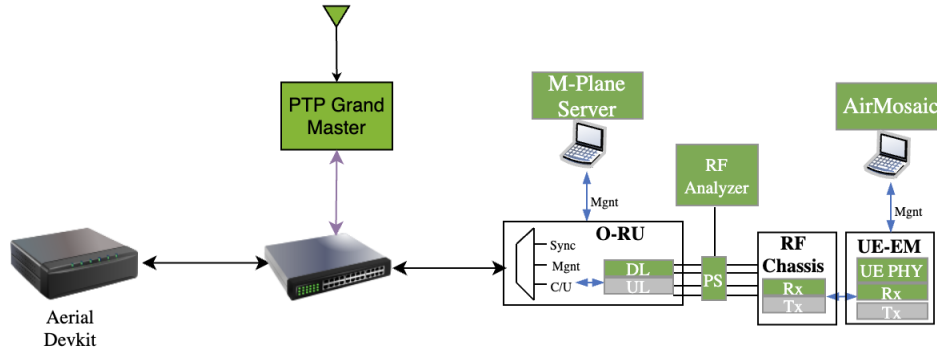


The diagram below shows an example of uplink functional testing:

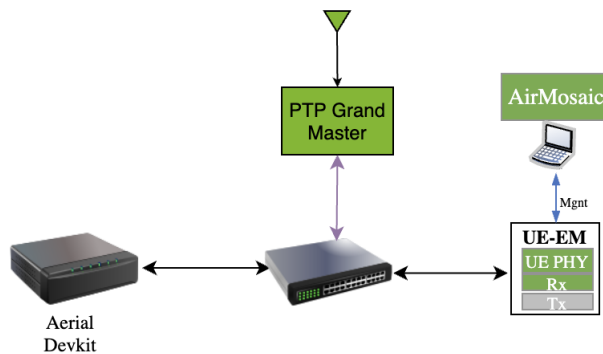


### 1.2.4.1.3.2 End to End Testing

A variety of end to end testing scenarios are possible. Shown below is one example using an Aerial gNB system implementing the CU+DU, an ORAN compliant RU connected to the DU via the ORAN fronthaul interface, and UE test equipment from Keysight.



Another example is the all-digital eCPRI topology is shown below with an Aerial gNB system implementing the CU+DU with the Keysight test equipment implementing the O-RU and UE functions.



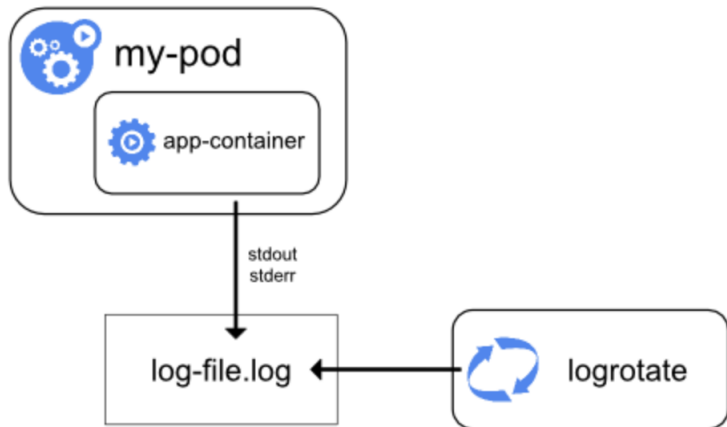
### 1.2.4.2 Fault Management

#### 1.2.4.2.1 Logging

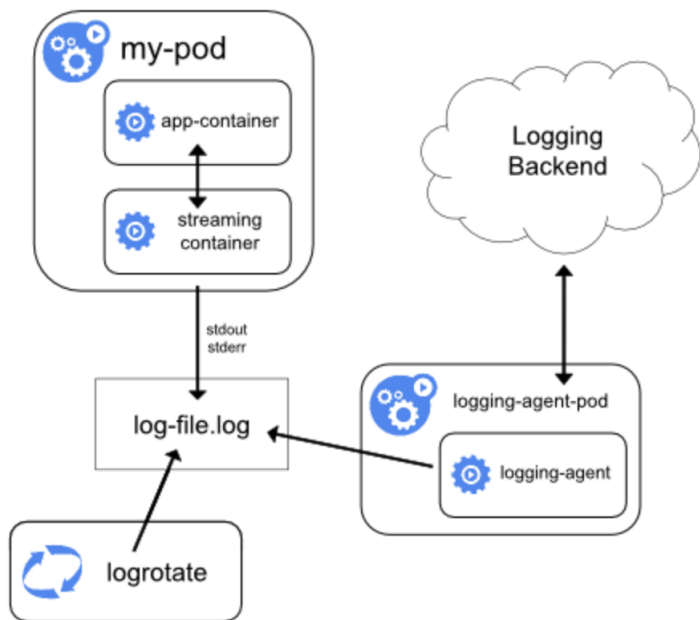
Aerial follows the best practices of Kubernetes (<https://kubernetes.io/docs/concepts/cluster-administration/logging/>) for implementing logging.

The cuphycontroller application outputs log messages, where the log level is less than or equal to the `nvlog.console_log_level` cuphycontroller YAML configuration parameter, directly to stdout using the **logging at the node level** pattern:

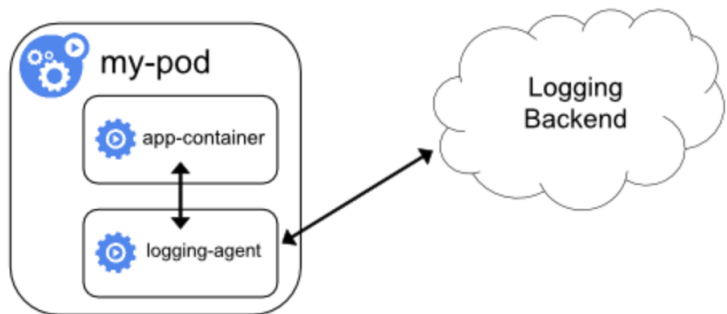




For high performance logs, Aerial uses a shared memory logger to offload the I/O bottleneck from the real-time threads. Log messages, where level is less than or equal to the `nvlog.shm_log_level` cu-pycontroller YAML configuration parameter, are output to the shared memory logger. The shared memory logger outputs can be retrieved using either the **streaming sidecar** pattern with logs written directly to the local disk:



Or the **sidecar with logging agent** pattern to stream directly to an external logging backend:



### 1.2.4.2.2 nvlog message format

Each nvlog message is a string of the form “[Software Component Name] Msg” prefixed with the following space-separated optional fields:

- ▶ Date
- ▶ Timestamp
- ▶ Primary or Secondary nvlog process
- ▶ Log level
- ▶ Log event code id
- ▶ Log event code string
- ▶ CPU core number the calling thread is running on
- ▶ 64-bit sequence number
- ▶ Thread ID
- ▶ Thread Name

These fields are enabled in the nvlog\_config.yaml.

An example nvlog message is:

```
20:58:09.036299 C [NVLOG.CPP] nvlog_create: name=phy shm_level=1
console_level=1 max_file_size=0x10000000 shm_cache_size=0x200000
log_buf_size=1024 prefix_opts=0x09
```

The message above had the following prefaces enabled:

- ▶ Timestamp
- ▶ Log level

Here are three more example nvlog messages, where all prefixed fields are enabled, taken at the start of the cuphycontroller process execution:

```
2021-09-15 21:29:22.926521 P C 0 SUCCESS 1 0 140699056300032
cuphycontroller [NVLOG.CPP] nvlog_create: name=phy shm_level=1
console_level=1 max_file_size=0x10000000 shm_cache_size=0x200000
log_buf_size=1024 prefix_opts=0xFF

2021-09-15 21:29:22.926560 P C 0 SUCCESS 1 1 140699056300032
cuphycontroller [CTL.SCF] Config file:
/cuBB_21-3/cuPHY-CP/cuphycontroller/config/cuphycontroller_V08.yaml

2021-09-15 21:29:23.130882 P C 0 SUCCESS 22 2 140699056300032
cuphycontroller [CTL.YAML] Standalone mode: No
```

Here is an example of an nvlog message at Fault level with Event Code AERIAL\_MEMORY\_EVENT:

```
20:58:09.036299 F MEMORY_EVENT Unable to allocate memory for FH buffers
```

The message above had the following prefaces enabled:

- ▶ Timestamp
- ▶ Log level
- ▶ Log event code string

The fields are further described herein:

Date is YYYY-MM-DD format, for example, 1970-01-01

Timestamp is HH:MM:SS.us, for example, 20:58:09.036299

Primary process is P, secondary process is S.

Log level is:

- ▶ F - Fatal
- ▶ E - Error
- ▶ C - Console
- ▶ W - Warning
- ▶ I - Info
- ▶ D - Debug
- ▶ V - Verbose

Log event code string or log event code id is a string (or a numerical id) that indicates the category of event that has occurred.

#### 1.2.4.2.3 nvlog Components

Aerial implements the following default logging component tags:

**nvlog component:**

- ▶ 10: "NVLOG"
- ▶ 11: "NVLOG.TEST"
- ▶ 12: "NVLOG.ITAG"

**nvipc component:**

- ▶ 30: "NVIIPC"

**cuPHY-CP Controller component:**

- ▶ 100: "CTL"
- ▶ 101: "CTL.SCF"
- ▶ 102: "CTL.ALTRAN"
- ▶ 103: "CTL.DRV"
- ▶ 104: "CTL.YAML"

**cuPHY-CP driver component:**

- ▶ 200: "DRV"
- ▶ 201: "DRV.SA"
- ▶ 202: "DRV.TIME"
- ▶ 203: "DRV.CTX"
- ▶ 204: "DRV.API"
- ▶ 205: "DRV.FH"

- ▶ 206: "DRV.GEN\_CUDA"
- ▶ 207: "DRV.GPUDEV"
- ▶ 208: "DRV.PHYCH"
- ▶ 209: "DRV.TASK"
- ▶ 210: "DRV.WORKER"
- ▶ 211: "DRV.DLBUF"
- ▶ 212: "DRV.CSIRS"
- ▶ 213: "DRV.PBCH"
- ▶ 214: "DRV.PDCCH\_DL"
- ▶ 215: "DRV.PDSCH"
- ▶ 216: "DRV.MAP\_DL"
- ▶ 217: "DRV.FUNC\_DL"
- ▶ 218: "DRV.HARQ\_POOL"
- ▶ 219: "DRV.ORDER\_CUDA"
- ▶ 220: "DRV.ORDER\_ENTITY"
- ▶ 221: "DRV.PRACH"
- ▶ 222: "DRV.PUCCH"
- ▶ 223: "DRV.PUSCH"
- ▶ 224: "DRV.MAP\_UL"
- ▶ 225: "DRV.FUNC\_UL"
- ▶ 226: "DRV.ULBUF"
- ▶ 227: "DRV.MPS"
- ▶ 228: "DRV.METRICS"
- ▶ 229: "DRV.MEMFOOT"
- ▶ 230: "DRV.CELL"

**cuPHY-CP cuphyl2adapter component:**

- ▶ 300: "L2A"
- ▶ 301: "L2A.MAC"
- ▶ 302: "L2A.MACFACT"
- ▶ 303: "L2A.PROXY"
- ▶ 304: "L2A.EPOLL"
- ▶ 305: "L2A.TRANSPORT"
- ▶ 306: "L2A.MODULE"
- ▶ 307: "L2A.TICK"
- ▶ 308: "L2A.UEMD"

**cuPHY-CP scfl2adapter component:**

- ▶ 330: "SCF"
- ▶ 331: "SCF.MAC"
- ▶ 332: "SCF.DISPATCH"
- ▶ 333: "SCF.PHY"
- ▶ 334: "SCF.SLOTCMD"
- ▶ 335: "SCF.L2SA"
- ▶ 336: "SCF.DUMMYMAC"

**cuPHY-CP testMAC component:**

- ▶ 400: "MAC"
- ▶ 401: "MAC.LP"
- ▶ 402: "MAC.FAPI"
- ▶ 403: "MAC.UTILS"
- ▶ 404: "MAC.SCF"
- ▶ 405: "MAC.ALTRAN"
- ▶ 406: "MAC.CFG"
- ▶ 407: "MAC.PROC"

**cuPHY-CP ru-emulator component:**

- ▶ 500: "RU"
- ▶ 501: "RU.EMULATOR"
- ▶ 502: "RU.PARSER"

**cuPHY-CP aerial-fh-driver component:**

- ▶ 600: "FH"
- ▶ 601: "FH.FLOW"
- ▶ 602: "FH.FH"
- ▶ 603: "FH.GPU\_MP"
- ▶ 604: "FH.LIB"
- ▶ 605: "FH.MEMREG"
- ▶ 606: "FH.METRICS"
- ▶ 607: "FH.NIC"
- ▶ 608: "FH.PDUMP"
- ▶ 609: "FH.PEER"
- ▶ 610: "FH.QUEUE"
- ▶ 611: "FH.RING"
- ▶ 612: "FH.TIME"

**cuPHY-CP compression\_decompression component:**

- ▶ 700: "COMP"

**cuPHY-CP cuphyoam component:**

- ▶ 800: "OAM"

**cuPHY component:**

- ▶ 900: "CUPHY"

**Note**

These strings can be changed using the `nvlog_config.yaml`.

**1.2.4.2.4 Event codes**

The following is the list of event codes (see `aerial_event_code.h`). The event strings match the event code names, minus the `AERIAL_`.

```
| AERIAL_SUCCESS                = 0,
| AERIAL_INVALID_PARAM_EVENT   = 1,
| AERIAL_INTERNAL_EVENT        = 2,
| AERIAL_CUDA_API_EVENT        = 3,
| AERIAL_DPDK_API_EVENT        = 4,
| AERIAL_THREAD_API_EVENT       = 5,
| AERIAL_CLOCK_API_EVENT       = 6,
| AERIAL_NVIPC_API_EVENT       = 7,
| AERIAL_ORAN_FH_EVENT         = 8,
| AERIAL_CUPHYDRV_API_EVENT     = 9,
| AERIAL_INPUT_OUTPUT_EVENT    = 10,
| AERIAL_MEMORY_EVENT          = 11,
| AERIAL_YAML_PARSER_EVENT     = 12,
| AERIAL_NVLOG_EVENT           = 13,
| AERIAL_CONFIG_EVENT          = 14,
| AERIAL_FAPI_EVENT            = 15,
| AERIAL_NO_SUPPORT_EVENT      = 16,
| AERIAL_SYSTEM_API_EVENT      = 17,
| AERIAL_L2ADAPTER_EVENT       = 18,
| AERIAL_RU_EMULATOR_EVENT    = 19,
```

**1.2.4.3 OAM Configuration****1.2.4.3.1 Startup Configuration (cuphycontroller)**

The application binary name for the combined cuPHY-CP + cuPHY is `cuphycontroller`. When `cuphycontroller` starts, it reads static configuration from configuration YAML files. This section describes the fields in the YAML files.

#### **1.2.4.3.1.1 l2adapter\_filename**

This field contains the filename of the YAML-format config file for I2 adapter configuration.

#### **1.2.4.3.1.2 aerial\_metrics\_backend\_address**

Aerial Prometheus metrics backend address.

#### **1.2.4.3.1.3 low\_priority\_core**

CPU core shared by all low-priority threads, isolated CPU core is preferred. Can be non-isolated CPU core but make sure no other heavy load task on it.

#### **1.2.4.3.1.4 nic\_tput\_alert\_threshold\_mbps**

This parameter is used to monitor NIC throughput. The units are in Mbps, that is, 85000 = 85 Gbps. This value is almost the max throughput that can be achieved with accurate send scheduling for a 100 Gbps link. A gRPC client(reference: \$cuBB\_SDK/cuPHY-CP/cuphyoam/examples/test\_grpc\_push\_notification\_client.cpp) needs to be implemented to receive the alert.

#### **1.2.4.3.1.5 cuphydriver\_config**

This container holds configuration for cuphydriver.

#### **1.2.4.3.1.6 standalone**

- 0 - run cuphydriver integrated with other cuPHY-CP components
- 1 - run cuphydriver in standalone mode (no I2adapter, etc)

#### **1.2.4.3.1.7 validation**

Enables additional validation checks at run-time.

- 0 - Disabled
- 1 - Enabled

#### **1.2.4.3.1.8 num\_slots**

Number of lots to run in cuphydriver standalone test.

#### **1.2.4.3.1.9 log\_level**

cuPHYDriver log level: DBG, INFO, ERROR.

#### **1.2.4.3.1.10 profiler\_sec**

Number of seconds to run the CUDA profiling tool.

#### **1.2.4.3.1.11 dpdk\_thread**

Sets the CPU core used by the primary DPDK thread. It does not have to be an isolated core. And the DPDK thread itself is defaulted to 'SCHED\_FIFO+priority 95'.

#### **1.2.4.3.1.12 dpdk\_verbose\_logs**

Enable maximum log level in DPDK.

0 - Disable

1 - Enable

#### **1.2.4.3.1.13 accu\_tx\_sched\_res\_ns**

Sets the accuracy of the accurate transmit scheduling, in units of nanoseconds.

#### **1.2.4.3.1.14 accu\_tx\_sched\_disable**

Disable accurate TX scheduling.

0 - packets are sent according to the TX timestamp

1 - packets are sent whenever it is convenient

#### **1.2.4.3.1.15 fh\_stats\_dump\_cpu\_core**

Sets the CPU core used by the FH stats logging thread. It does not have to be an isolated core. And currently the default FH stats polling interval is 500ms.



#### 1.2.4.3.1.16 `pdump_client_thread`

CPU core to use for pdump client. Set to -1 to disable fronthaul RX traffic PCAP capture.

See:

1. [https://doc.dpdk.org/guides/howto/packet\\_capture\\_framework.html](https://doc.dpdk.org/guides/howto/packet_capture_framework.html)
2. aerial-fh README.md

#### 1.2.4.3.1.17 `mps_sm_pusch`

Number of SMs for PUSCH channel.

#### 1.2.4.3.1.18 `mps_sm_pucch`

Number of SMs for PUCCH channel.

#### 1.2.4.3.1.19 `mps_sm_pusch`

Number of SMs for PUSCH channel.

#### 1.2.4.3.1.20 `mps_sm_prach`

Number of SMs for PRACH channel.

#### 1.2.4.3.1.21 `mps_sm_ul_order`

Number of SMs for UL order kernel.

#### 1.2.4.3.1.22 `mps_sm_pdsch`

Number of SMs for PDSCH channel.

#### 1.2.4.3.1.23 `mps_sm_pdcch`

Number of SMs for PDCCH channel.

**1.2.4.3.1.24 mps\_sm\_pbch**

Number of SMs for PBCH channel.

**1.2.4.3.1.25 mps\_sm\_srs**

Number of SMs for SRS channel.

**1.2.4.3.1.26 mps\_sm\_gpu\_comms**

Number of SMs for GPU comms.

**1.2.4.3.1.27 nics**

Container for NIC configuration parameters.

**1.2.4.3.1.28 nic**

PCIe bus address of the NIC port.

**1.2.4.3.1.29 mtu**

Maximum transmission size, in bytes, supported by the Fronthaul U-plane and C-plane.

**1.2.4.3.1.30 cpu\_mbufs**

Number of preallocated DPDK memory buffers (mbufs) used for Ethernet packets.

**1.2.4.3.1.31 uplane\_tx\_handles**

The number of pre-allocated transmit handles that link the U-plane prepare() and transmit() functions.

**1.2.4.3.1.32 txq\_count**

NIC transmit queue count.

Must be large enough to handle all cells attached to this NIC port.

Each cell uses one TXQ for C-plane and *txq\_count\_uplane* TXQs for U-plane.

**1.2.4.3.1.33 rxq\_count**

Receive queue count.

This value must be large enough to handle all cell attached to this NIC port.

Each cell uses one RXQ to receive all uplink traffic.

**1.2.4.3.1.34 txq\_size**

Number of packets that can fit in each transmit queue.

**1.2.4.3.1.35 rxq\_size**

Number of packets that can be buffered in each receive queue.

**1.2.4.3.1.36 gpu**

CUDA device to receive uplink packets from this NIC port.

**1.2.4.3.1.37 gpus**

List of GPU device IDs. To use gpudirect, the GPU must be on the same PCIe root complex as the NIC. To maximize performance, the GPU should be on the same PCIe switch as the NIC. Only the first entry in the list is used.

**1.2.4.3.1.38 workers\_ul**

List of pinned CPU cores used for uplink worker threads.

**1.2.4.3.1.39 workers\_dl**

List of pinned CPU cores used for downlink worker threads.

**1.2.4.3.1.40 debug\_worker**

For performance debug purpose, this is set to a free core to work with the enable\_\*\_tracing logs.

**1.2.4.3.1.41 workers\_sched\_priority**

cuPHYDriver worker threads scheduling priority.

**1.2.4.3.1.42 dpdk\_file\_prefix**

Shared data file prefix to use for the underlying DPDK process.

**1.2.4.3.1.43 wfreq**

Filename containing the coefficients for channel estimation filters, in HDF5 (.h5) format.

**1.2.4.3.1.44 cell\_group**

Enable cuPHY cell groups.

0 - disable 1 - enable

**1.2.4.3.1.45 cell\_group\_num**

Number of cells to be configured in L1 for the test.

**1.2.4.3.1.46 enable\_h2d\_copy\_thread**

Enable/disable offloading of h2d copy in L2A to a separate copy thread.

**1.2.4.3.1.47 h2d\_copy\_thread\_cpu\_affinity**

CPU core on which the h2d copy thread in L2A should run. Applicable only if enable\_h2d\_copy\_thread is 1.

**1.2.4.3.1.48 h2d\_copy\_thread\_sched\_priority**

h2d copy thread priority in L2A. Applicable only if enable\_h2d\_copy\_thread is 1.

**1.2.4.3.1.49 fix\_beta\_dl**

Fix the beta\_dl for local test with RU Emulator so that the output values are a bytematch to the TV.

**1.2.4.3.1.50 prometheus\_thread**

Pinned CPU core for updating NIC metrics once per second.

**1.2.4.3.1.51 start\_section\_id\_srs**

ORAN CUS start section ID for the SRS channel.

**1.2.4.3.1.52 start\_section\_id\_prach**

ORAN CUS start section ID for the PRACH channel.

**1.2.4.3.1.53 enable\_ul\_cuphy\_graphs**

Enable UL processing with CUDA graphs.

**1.2.4.3.1.54 enable\_dl\_cuphy\_graphs**

Enable DL processing with CUDA graphs.

**1.2.4.3.1.55 section\_3\_time\_offset**

Time offset, in units of nanoseconds, for the PRACH channel.

**1.2.4.3.1.56 ul\_order\_timeout\_cpu\_ns**

Timeout, in units of nanoseconds, for the uplink order kernel to receive any U-plane packets for this slot.

**1.2.4.3.1.57 ul\_order\_timeout\_gpu\_ns**

Timeout, in units of nanoseconds, for the order kernel to complete execution on the GPU.

**1.2.4.3.1.58 pusch\_sinr**

Enable pusch sinr calculation (0 by default).

#### **1.2.4.3.1.59 pusch\_rssi**

Enable PUSCH RSSI calculation (0 by default).

#### **1.2.4.3.1.60 pusch\_tdi**

Enable PUSCH TDI processing (0 by default).

#### **1.2.4.3.1.61 pusch\_cfo**

Enable PUSCH CFO calculations (0 by default).

#### **1.2.4.3.1.62 pusch\_dftsofdm**

DFT-s-OFDM enable/disable flag: 0 - disable, 1 - enable.

#### **1.2.4.3.1.63 pusch\_to**

It is only used for timing offset reporting to L2. If the timing offset estimate is not used by L2, it can be disabled.

#### **1.2.4.3.1.64 pusch\_select\_eqcoeffalgo**

Algorithm selector for PUSCH noise interference estimation and channel equalization. The following values are supported: 0: Regularized zero-forcing (RZF) 1: Diagonal MMSE regularization 2: Minimum Mean Square Error - Interference Rejection Combining (MMSE-IRC) 3: MMSE-IRC with RBLW covariance shrinkage 4: MMSE-IRC with OAS covariance shrinkage.

#### **1.2.4.3.1.65 pusch\_select\_chestalgo**

Channel estimation algorithm selection: 0 - legacy MMSE, 1 - multi-stage MMSE with delay estimation.

#### **1.2.4.3.1.66 pusch\_tbsizecheck**

Tb size verification enable/disable flag: 0 - disable, 1 - enable.

#### 1.2.4.3.1.67 `pusch_subSlotProcEn`

Sub-slot processing enable/disable flag: 0 - disable, 1 - enable. The early HARQ feature will be enabled accordingly when this flag is enabled. To get HARQ values in UCI.indication for UCI on PUSCH, before complete PUSCH slot processing, L2 should include PHY configuration TLV 0x102B (indicationInstancesPerSlot) with UCI.indication set to 2, according to Table 3–36 in SCF FAPI 222.10.04. If UCI.indication set to 2 in CONFIG.request for any cell the early HARQ feature will get activated for all cells.

#### 1.2.4.3.1.68 `pusch_deviceGraphLaunchEn`

Static flag to allow device graph launch in PUSCH.

#### 1.2.4.3.1.69 `pusch_waitTimeOutPreEarlyHarqUs`

Timeout threshold in microseconds for receiving OFDM symbols for PUSCH early-HARQ processing.

#### 1.2.4.3.1.70 `pusch_waitTimeOutPostEarlyHarqUs`

Timeout threshold in microseconds for receiving OFDM symbols for PUSCH non-early-HARQ processing (essentially all the PUSCH symbols).

#### 1.2.4.3.1.71 `puxch_polarDcdrListSz`

List size used in List Decoding of Polar codes.

#### 1.2.4.3.1.72 `enable_cpu_task_tracing`

The flag is used to trace and instrument DL/UL CPU tasks running on existing cuphydriver cores.

#### 1.2.4.3.1.73 `enable_prepare_tracing`

It's for tracing the U-plane packet preparation kernel durations and end times and need the debug worker to be enabled.

#### 1.2.4.3.1.74 `enable_dl_cqe_tracing`

Enables tracing of DL CQEs (debug feature to check for DL U-plane packets' timing at the NIC).

#### **1.2.4.3.1.75 ul\_rx\_pkt\_tracing\_level**

This YAML param can be set to 3 different values: 0 (default, recommended) : Only keeps count of the early/ontime/late packet counters per slot as seen by the DU (Reorder kernel) for the Uplink U-plane packets. 1 : Also Captures and logs earliest/latest packet timestamp per symbol per slot as seen by the DU. 2 : Also Captures and logs timestamp of each packet received per symbol per slot as seen by the DU.

#### **1.2.4.3.1.76 split\_ul\_cuda\_streams**

Keep default of 0. This allows back to back UL slots to overlap their processing. Keep disabled to maintain performance of first UL slot in every group of 2.

#### **1.2.4.3.1.77 aggr\_obj\_non\_avail\_th**

Keep the default value at 5. This param sets the threshold for successive non-availability of L1 objects (can be interpreted as L1 handler necessary to schedule PHY compute tasks to the GPU). Unavailability could imply the execution timeline falling behind the expected L1 timeline budget.

#### **1.2.4.3.1.78 dl\_wait\_th\_ns**

This parameter is used for error handling in the event of GPU failure. You must keep the defaults.

#### **1.2.4.3.1.79 sendCPlane\_timing\_error\_th\_ns**

Keep the default value at 50000 (50 us). The threshold is used as a check for the proximity of the current time during C-plane task's execution to the actual scheduled C-plane packet's transmission time. Meeting the threshold check would result in C-plane packet transmission being dropped for the slot.

#### **1.2.4.3.1.80 pusch\_forcedNumCsi2Bits**

Debug feaure if > 0, overrides the number of PUSCH CSI-P2 bits for all CSI-P2 UCIs with the non-zero value provided. Recommend setting it to 0.

#### **1.2.4.3.1.81 mMIMO\_enable**

Keep at default of 0. This flag is reserved for future capability.



#### 1.2.4.3.1.82 enable\_srs

Enable/disable SRS

#### 1.2.4.3.1.83 enable\_csip2\_v3

Enable/disable the the support of CSI part2 defined by FAPI 10.03 Table 3-77

#### 1.2.4.3.1.84 pusch\_aggr\_per\_ctx

Number of PUSCH objects per context (3 by default).

#### 1.2.4.3.1.85 prach\_aggr\_per\_ctx

Number of PRACH objects per context (2 by default).

#### 1.2.4.3.1.86 pucch\_aggr\_per\_ctx

Number of PUCCH objects per context (4 by default).

#### 1.2.4.3.1.87 srs\_aggr\_per\_ctx

Number of SRS objects per context (2 by default).

#### 1.2.4.3.1.88 ul\_input\_buffer\_per\_cell

Number of UL buffers allocated per cell (10 by default).

#### 1.2.4.3.1.89 ul\_input\_buffer\_per\_cell\_srs

Number of UL buffers allocated per cell for SRS (4 by default).

#### 1.2.4.3.1.90 ue\_mode

Flag for spectral efficiency feature. Must be enabled on the RU side YAML to emulate UE operation.

**1.2.4.3.1.91 cplane\_disable**

Disable C-plane for all cells.

0 - Enable C-plane 1 - Disable C-plane

**1.2.4.3.1.92 cells**

List of containers of cell parameters.

**1.2.4.3.1.93 name**

Name of the cell

**1.2.4.3.1.94 cell\_id**

ID of the cell.

**1.2.4.3.1.95 src\_mac\_addr**

Source MAC address for U-plane and C-plane packets. Set to 00:00:00:00:00:00 to use the MAC address of the NIC port in use.

**1.2.4.3.1.96 dst\_mac\_addr**

Destination MAC address for U-plane and C-plane packets.

**1.2.4.3.1.97 nic**

gNB NIC port to which the cell is attached.

Must match the 'nic' key value in one of the elements of in the 'nics' list.

**1.2.4.3.1.98 vlan**

VLAN ID used for C-plane and U-plane packets.

#### 1.2.4.3.1.99 **pcp**

QoS priority codepoint used for C-plane and U-plane Ethernet packets.

#### 1.2.4.3.1.100 **txq\_count\_uplane**

Number of transmit queues used for U-plane.

#### 1.2.4.3.1.101 **eAxC\_id\_ssb\_pbch**

List of eAxC IDs to use for SSB/PBCH.

#### 1.2.4.3.1.102 **eAxC\_id\_pdcch**

List of eAxC IDs to use for PDCCH.

#### 1.2.4.3.1.103 **eAxC\_id\_pdsch**

List of eAxC IDs to use for PDSCH.

#### 1.2.4.3.1.104 **eAxC\_id\_csirs**

List of eAxC IDs to use for CSI RS.

#### 1.2.4.3.1.105 **eAxC\_id\_pusch**

List of eAxC IDs to use for PUSCH.

#### 1.2.4.3.1.106 **eAxC\_id\_pucch**

List of eAxC IDs to use for PUCCH.

#### 1.2.4.3.1.107 **eAxC\_id\_srs**

List of eAxC IDs to use for SRS.

**1.2.4.3.1.108 eAxC\_id\_prach**

List of eAxC IDs to use for PRACH.

**1.2.4.3.1.109 dl\_iq\_data\_fmt:comp\_meth**

DL U-plane compression method: 0: Fixed point 1: BFP

**1.2.4.3.1.110 dl\_iq\_data\_fmt:bit\_width**

Number of bits used for each RE on DL U-plane channels. Fixed point supported value: 16 BFP supported value: 9, 14, 16

**1.2.4.3.1.111 ul\_iq\_data\_fmt:comp\_meth**

UL U-plane compression method: 0: Fixed point 1: BFP

**1.2.4.3.1.112 ul\_iq\_data\_fmt:bit\_width**

Number of bits used per RE on uplink U-plane channels. Fixed point supported value: 16 BFP supported value: 9, 14, 16

**1.2.4.3.1.113 fs\_offset\_dl**

Downlink U-plane scaling per ORAN CUS 6.1.3.

**1.2.4.3.1.114 exponent\_dl**

Downlink U-plane scaling per ORAN CUS 6.1.3.

**1.2.4.3.1.115 ref\_dl**

Downlink U-plane scaling per ORAN CUS 6.1.3.

**1.2.4.3.1.116 fs\_offset\_ul**

Uplink U-plane scaling per ORAN CUS 6.1.3.

**1.2.4.3.1.117 exponent\_ul**

Uplink U-plane scaling per ORAN CUS 6.1.3.

**1.2.4.3.1.118 max\_amp\_ul**

Maximum full scale amplitude used in uplink U-plane scaling per ORAN CUS 6.1.3.

**1.2.4.3.1.119 mu**

3GPP subcarrier bandwidth index 'mu'.

0 - 15 kHz 1 - 30 kHz 2 - 60 kHz 3 - 120 kHz 4 - 240 kHz

**1.2.4.3.1.120 T1a\_max\_up\_ns**

Scheduled timing advance before time-zero for downlink U-plane egress from DU, per ORAN CUS.

**1.2.4.3.1.121 T1a\_max\_cp\_ul\_ns**

Scheduled timing advance before time-zero for uplink C-plane egress from DU, per ORAN CUS.

**1.2.4.3.1.122 Ta4\_min\_ns**

Start of DU reception window after time-zero, per ORAN CUS.

**1.2.4.3.1.123 Ta4\_max\_ns**

End of DU reception window after time-zero, per ORAN CUS.

**1.2.4.3.1.124 Tcp\_adv\_dl\_ns**

Downlink C-plane timing advance ahead of U-plane, in units of nanoseconds, per ORAN CUS.

**1.2.4.3.1.125 ul\_u\_plane\_tx\_offset\_ns**

Flag for spectral efficiency feature. Must be set on the RU side YAML to offset UL transmission start from T0.

#### 1.2.4.3.1.126 `pusch_prb_stride`

Memory stride, in units of PRBs, for the PUSCH channel. Affects GPU memory layout.

#### 1.2.4.3.1.127 `prach_prb_stride`

Memory stride, in units of PRBs, for the PRACH channel. Affects GPU memory layout.

#### 1.2.4.3.1.128 `srs_prb_stride`

Memory stride, in units of PRBs, for the SRS. Affects GPU memory layout.

#### 1.2.4.3.1.129 `pusch_ldpc_max_num_itr_algo_type`

0 - Fixed LDPC iteration count

1 - MCS based LDPC iteration count

Recommend setting `pusch_ldpc_max_num_itr_algo_type:1`

#### 1.2.4.3.1.130 `pusch_fixed_max_num_ldpc_itrs`

Unused currently, reserved to replace `pusch_ldpc_n_iterations`.

#### 1.2.4.3.1.131 `pusch_ldpc_n_iterations`

Iteration count is set to `pusch_ldpc_n_iterations`, when the fixed LDPC iteration count option is selected (`pusch_ldpc_max_num_itr_algo_type:0`). Because the default value of `pusch_ldpc_max_num_itr_algo_type` is 1 (iteration count optimized based on MCS), `pusch_ldpc_n_iterations` is unused.

#### 1.2.4.3.1.132 `pusch_ldpc_algo_index`

Algorithm index for LDPC decoder: 0 - automatic choice.

#### 1.2.4.3.1.133 `pusch_ldpc_flags`

`pusch_ldpc_flags` are flags that configure the LDPC decoder. `pusch_ldpc_flags:2` selects an LDPC decoder that optimizes for throughput i.e processes more than one codeword (for example, 2) instead of latency.

#### 1.2.4.3.1.134 `pusch_ldpc_use_half`

##### Indication of input data type of LDPC decoder:

0 - single precision, 1 - half precision

#### 1.2.4.3.1.135 `pusch_nMaxPrb`

This is for memory allocation of max PRB range of peak cells compared to average cells.

#### 1.2.4.3.1.136 `ul_gain_calibration`

UL Configured Gain used to convert dBFS to dBm. Default value, if unspecified: 48.68

#### 1.2.4.3.1.137 `lower_guard_bw`

Lower Guard Bandwidth expressed in kHz. Used for deriving freqOffset for each RACH Occasion. Default is 845.

#### 1.2.4.3.1.138 `tv_pusch`

HDF5 file containing static configuration (for example, filter coefficients) for the PUSCH channel.

#### 1.2.4.3.1.139 `tv_prach`

HDF5 file containing static configuration (for example, filter coefficients) for the PRACH channel.

#### 1.2.4.3.1.140 `pusch_ldpc_n_iterations`

PUSCH LDPC channel coding iteration count.

#### 1.2.4.3.1.141 `pusch_ldpc_early_termination`

PUSCH LDPC channel coding early termination.

0 - Disable 1 - Enable

### 1.2.4.3.2 Startup Configuration (l2\_adapter\_config)

#### 1.2.4.3.2.1 msg\_type

Defines the L2/L1 interface API. Supported options are:

- ▶ scf\_fapi\_gnb - Use the small cell forum API.

#### 1.2.4.3.2.2 phy\_class

Same as msg\_type.

#### 1.2.4.3.2.3 tick\_generator\_mode

The SLOT indication interval generator mode:

0 - poll + sleep. During each tick the threads sleep some time to release the CPU core to avoid hanging the system, then they poll the system time. 1 - sleep. Sleep to absolute timestamp, no polling. 2 - timer\_fd. Start a timer and call epoll\_wait() on the timer\_fd.

#### 1.2.4.3.2.4 allowed\_fapi\_latency

Allowed maximum latency of SLOT FAPI messages, which send from L2 to L1, otherwise the message is ignored and dropped.

Unit: slot. Default is 0, it means L2 message should be received in current slot.

#### 1.2.4.3.2.5 allowed\_tick\_error

Allowed tick interval error.

Unit: us

Tick interval error is printed in statistic style. If observed tick error > allowed, the log is printed as Error level.

#### 1.2.4.3.2.6 timer\_thread\_config

Configuration for the timer thread.



#### 1.2.4.3.2.7 name

Name of thread.

#### 1.2.4.3.2.8 cpu\_affinity

Id of pinned CPU core used for timer thread.

#### 1.2.4.3.2.9 sched\_priority

Scheduling priority of timer thread.

#### 1.2.4.3.2.10 message\_thread\_config

Configuration container for the L2/L1 message processing thread.

#### 1.2.4.3.2.11 name

Name of thread.

#### 1.2.4.3.2.12 cpu\_affinity

Id of pinned CPU core used for timer thread.

#### 1.2.4.3.2.13 sched\_priority

Scheduling priority of message thread.

#### 1.2.4.3.2.14 ptp

ptp configs for GPS\_ALPHA, GPS\_BETA.

#### 1.2.4.3.2.15 gps\_alpha

GPS Alpha value for ORAN WG4 CUS section 9.7.2. Default value = 0, if undefined.

**1.2.4.3.2.16 gps\_beta**

GPS Beta value for ORAN WG4 CUS section 9.7.2. Default value = 0, if undefined.

**1.2.4.3.2.17 mu\_highest**

Highest supported mu, used for scheduling TTI tick rate.

**1.2.4.3.2.18 slot\_advance**

Timing advance ahead of time-zero, in units of slots, for L1 to notify L2 of a slot request.

**1.2.4.3.2.19 enableTickDynamicSfnSlot**

Enable dynamic slot/sfn.

**1.2.4.3.2.20 staticPucchSlotNum**

Debugging param for testing against RU Emulator to send set static PUCCH slot number.

**1.2.4.3.2.21 staticPuschSlotNum**

Debugging param for testing against RU Emulator to send set static PUSCH slot number.

**1.2.4.3.2.22 staticPdschSlotNum**

Debugging param for testing against RU Emulator to send set static PDSCH slot number.

**1.2.4.3.2.23 staticPdcchSlotNum**

Debugging param for testing against RU Emulator to send set static PDCCH slot number.

**1.2.4.3.2.24 staticCsiRsSlotNum**

Debugging param for testing against RU Emulator to send set static CSI-RS slot number.

#### 1.2.4.3.2.25 staticSsbSlotNum

Override the incoming slot number with the YAML configured SlotNumber for SS/PBCH.

Example

```
staticSsbSlotNum:10
```

#### 1.2.4.3.2.26 staticSsbPcid

Debugging param for testing against RU Emulator to send set static SSB phycellId.

#### 1.2.4.3.2.27 staticSsbSFN

Debugging param for testing against RU Emulator to send set static SSB SFN.

#### 1.2.4.3.2.28 pucch\_dtx\_thresholds

Array of scale factors for DTX Thresholds of each PUCCH format.

Default value, if not present, is 1.0, which means the thresholds are not scaled.

For PUCCH format 0 and 1, -100.0 is replaced with 1.0.

Example:

```
pucch_dtx_thresholds: [-100.0, -100.0, 1.0, 1.0, -100.0]
```

#### 1.2.4.3.2.29 pusch\_dtx\_thresholds

Scale factor for DTX Thresholds of UCI on PUSCH.

Default value, if not present, is 1.0, which means the threshold is not scaled.

Example:

```
pusch_dtx_thresholds: 1.0
```

#### 1.2.4.3.2.30 enable\_precoding

Enable/Disable Precoding PDUs to be parsed in L2Adapter.

Default value is 0 enable\_precoding: 0/1

#### **1.2.4.3.2.31 prepone\_h2d\_copy**

Enable/Disable preponing of H2D copy in L2Adapter.

Default value is 1 prepone\_h2d\_copy: 0/1

#### **1.2.4.3.2.32 enable\_beam\_forming**

Enables/Disables BeamIds to parsed in L2Adapter.

Default value : 0 enable\_beam\_forming: 1

#### **1.2.4.3.2.33 dl\_tb\_loc**

Transport block location in inside nvipc buffer.

Default value is 1 dl\_tb\_loc: 0 # TB is located in inline with nvipc's msg buffer. dl\_tb\_loc: 1 # TB is located in nvipc's CPU data buffer. dl\_tb\_loc: 2 # TB is located in nvipc's GPU buffer.

#### **1.2.4.3.2.34 instances**

Container for cell instances.

#### **1.2.4.3.2.35 name**

Name of the instance.

#### **1.2.4.3.2.36 nvipc\_config\_file**

Config dedicated YAML file for nvipc. Example: nvipc\_multi\_instances.yaml

#### **1.2.4.3.2.37 transport**

Configuration container for L2/L1 message transport parameters.

#### **1.2.4.3.2.38 type**

Transport type. One of shm, dpdk, or udp.

#### 1.2.4.3.2.39 **udp\_config**

Configuration container for the udp transport type.

#### 1.2.4.3.2.40 **local\_port**

UDP port used by L1.

#### 1.2.4.3.2.41 **remote\_port**

UDP port used by L2.

#### 1.2.4.3.2.42 **shm\_config**

Configuration container for the shared memory transport type.

#### 1.2.4.3.2.43 **primary**

Indicates process is primary for shared memory access.

#### 1.2.4.3.2.44 **prefix**

Prefix used in creating shared memory filename.

#### 1.2.4.3.2.45 **cuda\_device\_id**

Set this parameter to a valid GPU device ID to enable CPU data memory pool allocation in host pinned memory. Set to -1 to disable this feature.

#### 1.2.4.3.2.46 **ring\_len**

Length, in bytes, of the ring used for shared memory transport.

#### 1.2.4.3.2.47 **mempool\_size**

Configuration container for the memory pools used in shared memory transport.

**1.2.4.3.2.48 cpu\_msg**

Configuration container for the shared memory transport for CPU messages (that is, L2/L1 FAPI messages).

**1.2.4.3.2.49 buf\_size**

Buffer size in bytes.

**1.2.4.3.2.50 pool\_len**

Pool length in buffers.

**1.2.4.3.2.51 cpu\_data**

Configuration container for the shared memory transport for CPU data elements (that is, downlink and uplink transport blocks).

**1.2.4.3.2.52 buf\_size**

Buffer size in bytes.

**1.2.4.3.2.53 pool\_len**

Pool length in buffers.

**1.2.4.3.2.54 cuda\_data**

Configuration container for the shared memory transport for GPU data elements.

**1.2.4.3.2.55 buf\_size**

Buffer size in bytes.

**1.2.4.3.2.56 pool\_len**

Pool length in buffers.

#### 1.2.4.3.2.57 **dpgk\_config**

Configurations for the DPDK over NIC transport type.

#### 1.2.4.3.2.58 **primary**

Indicates process is primary for shared memory access.

#### 1.2.4.3.2.59 **prefix**

The name used in creating shared memory files and searching DPDK memory pools.

#### 1.2.4.3.2.60 **local\_nic\_pci**

The NIC address or name used in IPC.

#### 1.2.4.3.2.61 **peer\_nic\_mac**

The peer NIC MAC address, only need to be set in secondary process (L2/MAC).

#### 1.2.4.3.2.62 **cuda\_device\_id**

Set this parameter to a valid GPU device ID to enable CPU data memory pool allocation in host pinned memory. Set to -1 to disable this feature.

#### 1.2.4.3.2.63 **need\_eal\_init**

Whether nvipc needs to call rte\_eal\_init() to initiate the DPDK context. 1 - initiate by nvipc; 0 - initiate by other module in the same process.

#### 1.2.4.3.2.64 **lcore\_id**

The logic core number for nvipc\_nic\_poll thread.

#### 1.2.4.3.2.65 **mempool\_size**

Configuration container for the memory pools used in shared memory. transport.

**1.2.4.3.2.66 cpu\_msg**

Configuration container for the shared memory transport for CPU messages (that is, L2/L1 FAPI messages).

**1.2.4.3.2.67 buf\_size**

Buffer size in bytes.

**1.2.4.3.2.68 pool\_len**

Pool length in buffers.

**1.2.4.3.2.69 cpu\_data**

Configuration container for the shared memory transport for CPU data elements (that is, downlink and uplink transport blocks).

**1.2.4.3.2.70 buf\_size**

Buffer size in bytes.

**1.2.4.3.2.71 pool\_len**

Pool length in buffers.

**1.2.4.3.2.72 cuda\_data**

Configuration container for the shared memory transport for GPU data elements.

**1.2.4.3.2.73 buf\_size**

Buffer size in bytes.

**1.2.4.3.2.74 pool\_len**

Pool length in buffers.



#### 1.2.4.3.2.75 app\_config

Configurations for all transport types, mostly used for debug.

#### 1.2.4.3.2.76 grpc\_forward

Whether to enable forwarding nvipc messages and how many messages to be forwarded automatically from initialization. Here count = 0 means forwarding every message forever.

0: disabled; 1: enabled but doesn't start forwarding at initial; -1: enabled and start forwarding at initial with count = 0; Other positive number: enabled and start forwarding at initial with count = grpc\_forward.

#### 1.2.4.3.2.77 debug\_timing

For debug only.

Whether to record timestamp of allocating, sending, receiving, releasing of all nvipc messages.

#### 1.2.4.3.2.78 pcap\_enable

For debug only.

Whether to capture nvipc messages to pcap file.

#### 1.2.4.3.2.79 pcap\_cpu\_core

CPU core of background pcap log save thread.

#### 1.2.4.3.2.80 pcap\_cache\_size\_bits

Size of /dev/shm/\${prefix}\_pcap. If set to 29, size is  $2^{29} = 512\text{MB}$ .

#### 1.2.4.3.2.81 pcap\_file\_size\_bits

Max size of /dev/shm/\${prefix}\_pcap. If set to 31, size is  $2^{31} = 2\text{GB}$ .

#### 1.2.4.3.2.82 pcap\_max\_data\_size

Max DL/UL FAPI data size to capture reduce pcap size.

### 1.2.4.3.3 Startup Configuration (ru-emulator)

The application binary name for the combined O-RU + UE emulator is ru-emulator. When ru-emulator starts, it reads static configuration from a configuration YAML file. This section describes the fields in the YAML file.

#### 1.2.4.3.3.1 core\_list

List of CPU cores that RU Emulator could use.

#### 1.2.4.3.3.2 nic\_interface

PCIe address of NIC to use that is, b5:00:1.

#### 1.2.4.3.3.3 peerethaddr

MAC address of cuPHYController port.

#### 1.2.4.3.3.4 nvlog\_name

The nvlog instance name for ru-emulator. Detailed nvlog configurations are in nvlog\_config.yaml.

#### 1.2.4.3.3.5 cell\_configs

Cell configs agreed upon with DU.

#### 1.2.4.3.3.6 name

Cell string name (largely unused).

#### 1.2.4.3.3.7 eth

Cell MAC address.

#### 1.2.4.3.3.8 dl\_iq\_data\_fmt:comp\_meth

DL U-plane compression method: 0: Fixed point 1: BFP

#### 1.2.4.3.3.9 dl\_iq\_data\_fmt:bit\_width

Number of bits used for each RE on DL U-plane channels. Fixed point supported value: 16 BFP supported value: 9, 14, 16

#### 1.2.4.3.3.10 ul\_iq\_data\_fmt:comp\_meth

UL U-plane compression method: 0: Fixed point 1: BFP

#### 1.2.4.3.3.11 ul\_iq\_data\_fmt:bit\_width

Number of bits used for each RE on UL U-plane channels. Fixed point supported value: 16 BFP supported value: 9, 14, 16

#### 1.2.4.3.3.12 flow\_list

eAxC list

#### 1.2.4.3.3.13 eAxC\_prach\_list

eAxC prach list

#### 1.2.4.3.3.14 vlan

vlan to use for RX and TX

#### 1.2.4.3.3.15 nic

Index of the nic to use in the nics list.

#### 1.2.4.3.3.16 tti

Slot indication interval.

#### 1.2.4.3.3.17 validate\_dl\_timing

Validate DL timing (need to be PTP synchronized).

**1.2.4.3.3.18 timing\_histogram**

generate histogram

**1.2.4.3.3.19 timing\_histogram\_bin\_size**

histogram bin size

**1.2.4.3.3.20 oran\_timing\_info**

**1.2.4.3.3.21 dl\_c\_plane\_timing\_delay**

t1a\_max\_up from ORAN

**1.2.4.3.3.22 dl\_c\_plane\_window\_size**

DL C Plane RX ontime window size.

**1.2.4.3.3.23 ul\_c\_plane\_timing\_delay**

T1a\_max\_cp\_ul from ORAN.

**1.2.4.3.3.24 ul\_c\_plane\_window\_size**

UL C Plane RX ontime window size.

**1.2.4.3.3.25 dl\_u\_plane\_timing\_delay**

T2a\_max\_up from ORAN.

**1.2.4.3.3.26 dl\_u\_plane\_window\_size**

DL U Plane RX ontime window size.

**1.2.4.3.3.27 ul\_u\_plane\_tx\_offset**

Ta4\_min\_up from ORAN.

### 1.2.4.3.4 Run-time Configuration/Status

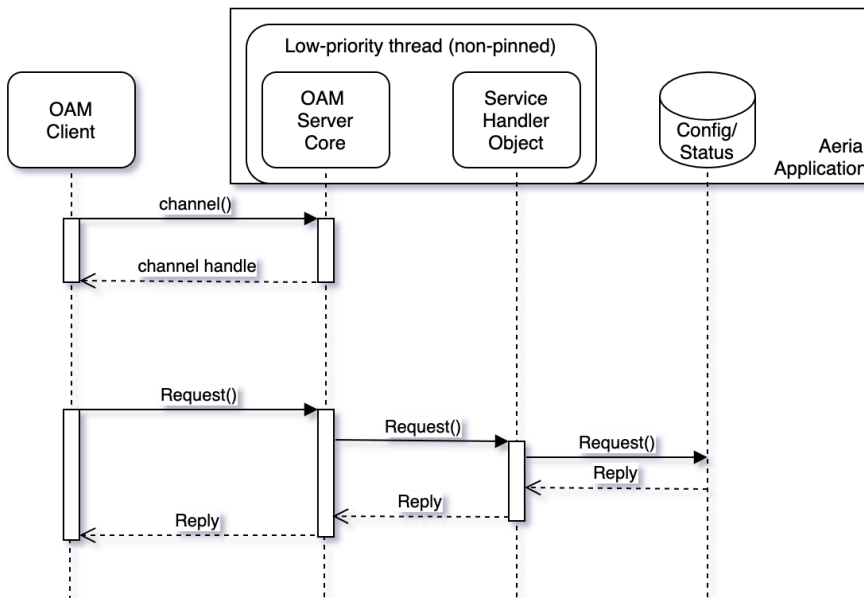
During run-time, Aerial components can be re-configured or queried for status through gRPC remote procedure calls (RPCs). The RPCs are defined in “protocol buffers” syntax, allowing support for clients written in any of the languages supported by gRPC and protocol buffers.

More information about gRPC may be found at: <https://grpc.io/docs/what-is-grpc/core-concepts/>

More information about protocol buffers may be found at: <https://developers.google.com/protocol-buffers>

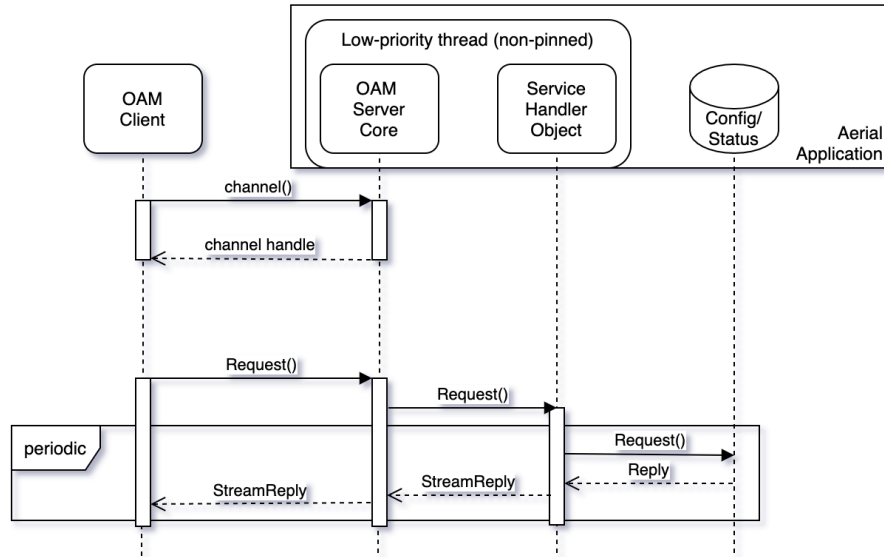
#### 1.2.4.3.4.1 Simple Request/Reply Flow

Aerial applications support a request/reply flow using the gRPC framework with protobufs messages. At run-time, certain configuration items may be updated and certain status information may be queried. An external OAM client interfaces with the Aerial application acting as the gRPC server.



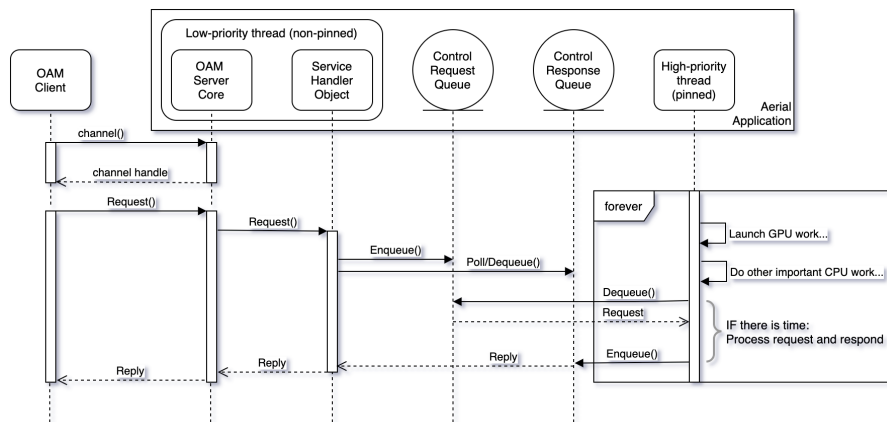
### 1.2.4.3.4.2 Streaming Request/Replies

Aerial applications support the gRPC streaming feature for sending periodic status between client and server.



### 1.2.4.3.4.3 Asynchronous Interthread Communication

Certain request/reply scenarios require interaction with the high-priority CPU-pinned threads orchestrating GPU work. These interactions occur through Aerial-internal asynchronous queues, and requests are processed on a best effort basis that prioritizes the orchestration of GPU kernel launches and other L1 tasks.



#### 1.2.4.3.4.4 Aerial Common Service Definition

```
/\*
\* Copyright (c) 2021, NVIDIA CORPORATION. All rights reserved.
\*
\* NVIDIA CORPORATION and its licensors retain all intellectual property
\* and proprietary rights in and to this software, related documentation
\* and any modifications thereto. Any use, reproduction, disclosure or
\* distribution of this software and related documentation without an
express
\* license agreement from NVIDIA CORPORATION is strictly prohibited.
\*/

syntax = "proto3";

package aerial;

service Common {
  rpc GetSFN (GenericRequest) returns (SFNReply) {}
  rpc GetCpuUtilization (GenericRequest) returns (CpuUtilizationReply) {}
  rpc SetPuschH5DumpNextCrc (GenericRequest) returns (DummyReply) {}
  rpc GetFAPISStream (FAPISStreamRequest) returns (stream FAPISStreamReply)
  {}
}

message GenericRequest {
  string name = 1;
}

message SFNReply {
  int32 sfn = 1;
  int32 slot = 2;
}

message DummyReply {
}

message CpuUtilizationPerCore {
```

(continues on next page)

(continued from previous page)

```
int32 core_id = 1;
int32 utilization_x1000 = 2;
}
message CpuUtilizationReply {
  repeated CpuUtilizationPerCore core = 1;
}
message FAPIStreamRequest {
  int32 client_id = 1;
  int32 total_msgs_requested = 2;
}
message FAPIStreamReply {
  int32 client_id = 1;
  bytes msg_buf = 2;
  bytes data_buf = 3;
}
```

#### 1.2.4.3.4.5 rpc GetCpuUtilization

The GetCpuUtilization RPC returns a variable-length array of CPU utilization per-high-priority-core.

CPU utilization is available through the Prometheus node exporter, however the design approach used by Aerial high-priority threads results in a false 100% CPU core utilization per thread. This RPC allows retrieval of the actual CPU utilization of high-priority threads. High-priority threads are pinned to specific CPU cores.



#### 1.2.4.3.4.6 rpc GetFAPISStream

This RPC requests snooping of one or more (up to infinite number) of SCF FAPI messages. The snooped messages are delivered from the Aerial gRPC server to a third party client. See `cuPHY-CP/cuphyoam/examples/aerial_get_l2msgs.py` for an example client.

#### 1.2.4.3.4.7 rpc TerminateCuphycontroller

This RPC message terminates `cuPHYController` with immediate effect.

#### 1.2.4.3.4.8 rpc CellParamUpdateRequest

This RPC message updates cell configuration without stopping the cell. Message specification:

```
message CellParamUpdateRequest {
  int32 cell_id = 1;
  string dst_mac_addr = 2;
  int32 vlan_tci = 3;
}
```

*dst\_mac\_addr* must be in 'XX:XX:XX:XX:XX:XX' format.

*vlan\_tci* must include the 16-bit TCI value of 802.1Q tag.

#### 1.2.4.3.4.9 List of Parameters Supported by Dynamic OAM via gRPC and CONFIG.request (M-plane)

The Configuration unit is accross all cells/per cell config. The Cell outage is either in-service or out-of-service.

##### Note

With OAM commands, you can use `localhost` for the `$SERVER_IP` when running on DU server. Otherwise, use the DU server numeric IP address. `$CELL_ID` is the mplane id, which starts from 1. The default values of the parameters can be found in the corresponding `cuphycontroller` YAML config file: `$cuBB_SDK/cuPHY-CP/cuphycontroller/config/cuphycontroller_XXX.yaml`

#### 1.2.4.3.4.10 ru\_type

- ▶ **Parameter Name:** ru\_type
- ▶ **Configuration Unit:** per-cell config
- ▶ **Cell Outage:** out-of-service
- ▶ **OAM Command:**

```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/  
→examples/aerial_cell_multi_attrs_update.py --server_ip $SERVER_IP --cell_id  
→$CELL_ID --ru_type $RU_TYPE
```

- ▶ **Notes:** \$RU\_TYPE : 1 for FXN\_RU, 2 for FJT\_RU, 3 for OTHER\_RU(including ru\_emulator)

#### 1.2.4.3.4.11 nic

- ▶ **Parameter Name:** nic
- ▶ **Configuration Unit:** per-cell config
- ▶ **Cell Outage:** out-of-service
- ▶ **OAM Command:**

```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/  
→examples/aerial_cell_multi_attrs_update.py --server_ip $SERVER_IP --cell_id  
→$CELL_ID --nic $NIC
```

- ▶ **Notes:** The nic PCIe address. It has to be one of the nic ports configured in cuphycontroller YAML file.

#### 1.2.4.3.4.12 dst\_mac\_addr

- ▶ **Parameter Name:** dst\_mac\_addr
- ▶ **Configuration Unit:** per-cell config
- ▶ **Cell Outage:** out-of-service
- ▶ **OAM Command:**

```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/  
→examples/aerial_cell_multi_attrs_update.py --server_ip $SERVER_IP --cell_id  
→$CELL_ID --dst_mac_addr $DST_MAC_ADDR --vlan_id $VLAN_ID --pcp $PCP
```

- ▶ **Notes:** The dst\_mac\_addr, vlan\_id, and pcp parameters must be updated together.

#### 1.2.4.3.4.13 vlan\_id

- ▶ **Parameter Name:** vlan\_id
- ▶ **Configuration Unit:** per-cell config
- ▶ **Cell Outage:** out-of-service
- ▶ **OAM Command:**

```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/
↳examples/aerial_cell_multi_attrs_update.py --server_ip $SERVER_IP --cell_id
↳$CELL_ID --dst_mac_addr $DST_MAC_ADDR --vlan_id $VLAN_ID --pcp $PCP
```

- ▶ **Notes:** The dst\_mac\_addr, vlan\_id, and pcp parameters must be updated together.

#### 1.2.4.3.4.14 pcp

- ▶ **Parameter Name:** pcp
- ▶ **Configuration Unit:** per-cell config
- ▶ **Cell Outage:** out-of-service
- ▶ **OAM Command:**

```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/
↳examples/aerial_cell_multi_attrs_update.py --server_ip $SERVER_IP --cell_id
↳$CELL_ID --dst_mac_addr $DST_MAC_ADDR --vlan_id $VLAN_ID --pcp $PCP
```

- ▶ **Notes:** The dst\_mac\_addr, vlan\_id, and pcp parameters must be updated together.

#### 1.2.4.3.4.15 dl\_iq\_data\_fmt

- ▶ **Parameter Name:** dl\_iq\_data\_fmt
- ▶ **Configuration Unit:** per-cell config
- ▶ **Cell Outage:** out-of-service
- ▶ **OAM Command:**

```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/
↳examples/aerial_cell_multi_attrs_update.py --server_ip $SERVER_IP --cell_id
↳$CELL_ID --dl_comp_meth $COMP_METH --dl_bit_width $BIT_WIDTH
```

#### 1.2.4.3.4.16 ul\_iq\_data\_fmt

- ▶ **Parameter Name:** ul\_iq\_data\_fmt
- ▶ **Configuration Unit:** per-cell config
- ▶ **Cell Outage:** out-of-service
- ▶ **OAM Command:**

```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/  
↳examples/aerial_cell_multi_attrs_update.py --server_ip $SERVER_IP --cell_id  
↳$CELL_ID --ul_comp_meth $COMP_METH --ul_bit_width $BIT_WIDTH
```

#### 1.2.4.3.4.17 exponent\_dl

- ▶ **Parameter Name:** exponent\_dl
- ▶ **Configuration Unit:** per-cell config
- ▶ **Cell Outage:** out-of-service
- ▶ **OAM Command:**

```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/  
↳examples/aerial_cell_multi_attrs_update.py --server_ip $SERVER_IP --cell_id  
↳$CELL_ID --exponent_dl $EXPONENT_DL
```

#### 1.2.4.3.4.18 exponent\_ul

- ▶ **Parameter Name:** exponent\_ul
- ▶ **Configuration Unit:** per-cell config
- ▶ **Cell Outage:** out-of-service
- ▶ **OAM Command:**

```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/  
↳examples/aerial_cell_multi_attrs_update.py --server_ip $SERVER_IP --cell_id  
↳$CELL_ID --exponent_ul $EXPONENT_UL
```

#### 1.2.4.3.4.19 prusch\_prb\_stride

- ▶ **Parameter Name:** prusch\_prb\_stride
- ▶ **Configuration Unit:** per-cell config
- ▶ **Cell Outage:** out-of-service
- ▶ **OAM Command:**

```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/  
↳examples/aerial_cell_multi_attrs_update.py --server_ip $SERVER_IP --cell_id  
↳$CELL_ID --pusch_prb_stride $PUSCH_PRB_STRIDE
```

#### 1.2.4.3.4.20 prach\_prb\_stride

- ▶ **Parameter Name:** prach\_prb\_stride
- ▶ **Configuration Unit:** per-cell config
- ▶ **Cell Outage:** out-of-service
- ▶ **OAM Command:**

```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/
↳examples/aerial_cell_multi_attrs_update.py --server_ip $SERVER_IP --cell_id
↳$CELL_ID --prach_prb_stride $PRACH_PRB_STRIDE
```

#### 1.2.4.3.4.21 max\_amp\_ul

- ▶ **Parameter Name:** max\_amp\_ul
- ▶ **Configuration Unit:** per-cell config
- ▶ **Cell Outage:** out-of-service
- ▶ **OAM Command:**

```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/
↳examples/aerial_cell_multi_attrs_update.py --server_ip $SERVER_IP --cell_id
↳$CELL_ID --max_amp_ul $MAX_AMP_UL
```

#### 1.2.4.3.4.22 section\_3\_time\_offset

- ▶ **Parameter Name:** section\_3\_time\_offset
- ▶ **Configuration Unit:** per-cell config
- ▶ **Cell Outage:** out-of-service
- ▶ **OAM Command:**

```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/
↳examples/aerial_cell_multi_attrs_update.py --server_ip $SERVER_IP --cell_id
↳$CELL_ID --section_3_time_offset $SECTION_3_TIME_OFFSET
```

#### 1.2.4.3.4.23 fh\_distance\_range

- ▶ **Parameter Name:** fh\_distance\_range
- ▶ **Configuration Unit:** per-cell config
- ▶ **Cell Outage:** out-of-service
- ▶ **OAM Command:**

```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/
↳examples/aerial_cell_multi_attrs_update.py --server_ip $SERVER_IP --cell_id
↳$CELL_ID --fh_distance_range $FH_DISTANCE_RANGE
```

- **Notes:** `$FH_DISTANCE_RANGE`: 0 for 0~30km, 1 for 20~50km. Suppose the following are the default configs in the cuhycontroller YAML config file that correspond to `FH_DISTANCE_RANGE` option 0 (0~30km).

```
t1a_max_up_ns : d1
t1a_max_cp_ul_ns : d2
ta4_min_ns : d3
ta4_max_ns : d4
```

Updating the `FH_DISTANCE_RANGE` option to 1 (20~50km) adjusts the following values:

```
t1a_max_up_ns : d1+$FH_EXTENSION_DELAY_ADJUSTMENT
t1a_max_cp_ul_ns : d2+$FH_EXTENSION_DELAY_ADJUSTMENT
ta4_min_ns : d3+$FH_EXTENSION_DELAY_ADJUSTMENT
ta4_max_ns : d4+$FH_EXTENSION_DELAY_ADJUSTMENT
```

`$FH_EXTENSION_DELAY_ADJUSTMENT` is 100us for now and can be tuned in source file:

```

${cuBB_SDK}/cuPHY-CP/cuphydriver/include/constant.hpp#L207
static constexpr uint32_t FH_EXTENSION_DELAY_ADJUSTMENT = 100000;//100us
```

#### 1.2.4.3.4.24 ul\_gain\_calibration

- **Parameter Name:** `ul_gain_calibration`
- **Configuration Unit:** per-cell config
- **Cell Outage:** out-of-service
- **OAM Command:**

```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/
↪examples/aerial_cell_multi_attrs_update.py --server_ip $SERVER_IP --cell_id
↪$CELL_ID --ul_gain_calibration $UL_GAIN_CALIBRATION
```

#### 1.2.4.3.4.25 lower\_guard\_bw

- **Parameter Name:** `lower_guard_bw`
- **Configuration Unit:** per-cell config
- **Cell Outage:** out-of-service
- **OAM Command:**

```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/
↪examples/aerial_cell_multi_attrs_update.py --server_ip $SERVER_IP --cell_id
↪$CELL_ID --lower_guard_bw $LOWER_GUARD_BW
```

#### 1.2.4.3.4.26 ref\_dl

- ▶ **Parameter Name:** ref\_dl
- ▶ **Configuration Unit:** per-cell config
- ▶ **Cell Outage:** out-of-service
- ▶ **OAM Command:**

```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/
↳examples/aerial_cell_multi_attrs_update.py --server_ip $SERVER_IP --cell_id
↳$CELL_ID --ref_dl $REF_DL
```

#### 1.2.4.3.4.27 attenuation\_db

- ▶ **Parameter Name:** attenuation\_db
- ▶ **Configuration Unit:** per-cell config
- ▶ **Cell Outage:** out-of-service
- ▶ **OAM Command:**

```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/
↳examples/aerial_cell_param_attn_update.py $CELL_ID $ATTENUATION_DB
```

#### 1.2.4.3.4.28 gps\_alpha

- ▶ **Parameter Name:** gps\_alpha
- ▶ **Configuration Unit:** accross all cells
- ▶ **Cell Outage:** out-of-service
- ▶ **OAM Command:**

```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/
↳examples/aerial_cell_multi_attrs_update.py --server_ip $SERVER_IP --gps_alpha
↳$GPS_ALPHA
```

#### 1.2.4.3.4.29 gps\_beta

- ▶ **Parameter Name:** gps\_beta
- ▶ **Configuration Unit:** accross all cells
- ▶ **Cell Outage:** out-of-service
- ▶ **OAM Command:**

```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/
↳examples/aerial_cell_multi_attrs_update.py --server_ip $SERVER_IP --gps_beta
↳$GPS_BETA
```

- ▶ **Notes:** All cells have to be in idle state before configuring this param

#### 1.2.4.3.4.30 prachRootSequenceIndex

- ▶ **Parameter Name:** prachRootSequenceIndex
- ▶ **Configuration Unit:** per-cell config
- ▶ **Cell Outage:** out-of-service
- ▶ **OAM Command:** Via FAPI CONFIG.request. Refer to the *Dynamic PRACH Configuration and Init Sequence Test* section for more details.

#### 1.2.4.3.4.31 prachZeroCorrConf

- ▶ **Parameter Name:** prachZeroCorrConf
- ▶ **Configuration Unit:** per-cell config
- ▶ **Cell Outage:** out-of-service
- ▶ **OAM Command:** Via FAPI CONFIG.request. Refer to the *Dynamic PRACH Configuration and Init Sequence Test* section for more details.

#### 1.2.4.3.4.32 numPrachFdOccurrences

- ▶ **Parameter Name:** numPrachFdOccurrences
- ▶ **Configuration Unit:** per-cell config
- ▶ **Cell Outage:** out-of-service
- ▶ **OAM Command:** Via FAPI CONFIG.request. Refer to the *Dynamic PRACH Configuration and Init Sequence Test* section for more details.

#### 1.2.4.3.4.33 restrictedSetConfig

- ▶ **Parameter Name:** restrictedSetConfig
- ▶ **Configuration Unit:** per-cell config
- ▶ **Cell Outage:** out-of-service
- ▶ **OAM Command:** Via FAPI CONFIG.request. Refer to the *Dynamic PRACH Configuration and Init Sequence Test* section for more details.

#### 1.2.4.3.4.34 prachConfigIndex

- ▶ **Parameter Name:** prachConfigIndex
- ▶ **Configuration Unit:** per-cell config
- ▶ **Cell Outage:** out-of-service
- ▶ **OAM Command:** Via FAPI CONFIG.request. Refer to the *Dynamic PRACH Configuration and Init Sequence Test* section for more details.



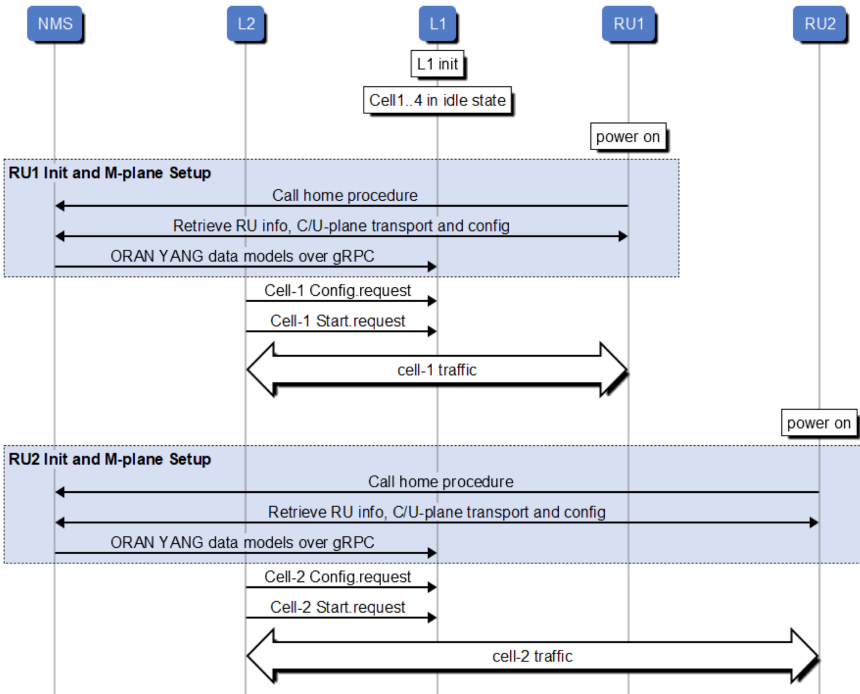
1.2.4.3.4.35 K1

- ▶ **Parameter Name:** K1
- ▶ **Configuration Unit:** per-cell config
- ▶ **Cell Outage:** out-of-service
- ▶ **OAM Command:** Via FAPI CONFIG.request. Refer to the *Dynamic PRACH Configuration and Init Sequence Test* section for more details.

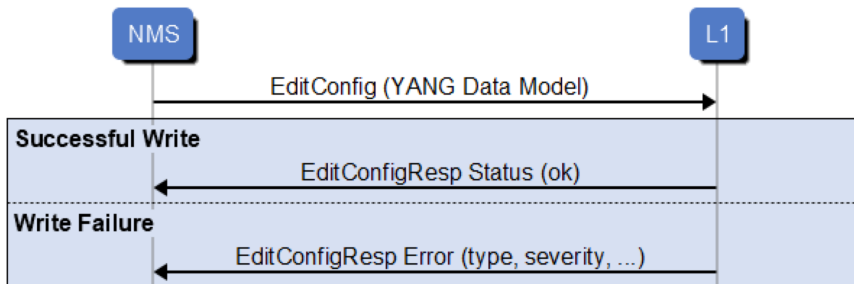
1.2.4.3.5 M-Plane Hybrid Mode ORAN YANG Model Provisioning

Aerial supports M-plane hybrid mode, which allows NMS/SMO, using ORAN YANG data models to pass RU capabilities, C/U-plane transport config, and U-plane config to L1.

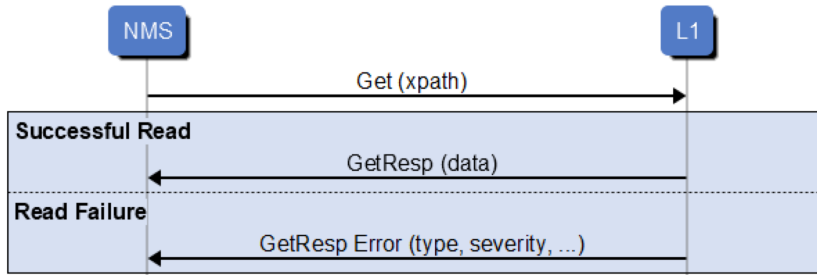
Here is the high level sequence diagram:



Data Model Procedures-Yang data tree write procedure



Data Model Procedures-Yang data tree read procedure



#### 1.2.4.3.5.1 Data Model Transfer APIs(gRPC ProtoBuf contract)

```

syntax = "proto3";
package p9_messages.v1;

service P9Messages {
  rpc HandleMsg (Msg) returns (Msg) {}
}

message Msg
{
  Header header = 1;
  Body body = 2;
}

message Header
{
  string msg_id = 1;           // Message identifier to
                              // 1) Identify requests and notifications
                              // 2) Correlate requests and response
  optional string oru_name = 2; // The name (identifier) of the O-RU, if present.
  int32 vf_id = 3;           // The identifier for the FAPI VF ID
  int32 phy_id = 4;          // The identifier for the FAPI PHY ID
  optional int32 trp_id = 5;  // The identifier PHY's TRP, if any
}

message Body
{
  oneof msg_body
  {
    Request request = 1;
    Response response = 2;
  }
}

message Request
{
  oneof req_type
  {
    Get get = 1;
    EditConfig edit_config = 2;
  }
}

message Response

```

(continues on next page)

(continued from previous page)

```

{
  oneof resp_type
  {
    GetResp get_resp = 1;
    EditConfigResp edit_config_resp = 2;
  }
}

message Get { repeated bytes filter = 1; }

message GetResp
{
  Status status_resp = 1;
  bytes data = 2;
}

message EditConfig
{
  bytes delta_config = 1; // List of Node changes with the associated operation to
  ↪apply to the node
}

message EditConfigResp { Status status_resp = 1; }

message Error
{
  string error_type = 1; // Type of error as defined in RFC 6241 section 4.3
  string error_tag = 2; // Error tag defined in RFC 6241, Appendix B
  string error_severity = 3; // Error severity defined in RFC 6241, Appendix B
  string error_app_tag = 4; // Error app tag defined in RFC 6241, Appendix B
  string error_path = 5; // Error path defined in RFC 6241, Appendix B
  string error_message = 6; // Error message defined in RFC 6241, Appendix B
}

message Status
{
  enum StatusCode
  {
    OK = 0;
    ERROR_GENERAL = 1;
  }
  StatusCode status_code = 1;
  repeated Error error = 2; // Optional: Error information
}

```

#### 1.2.4.3.5.2 List of Parameters Supported by YANG Model

The Configuration unit is accross all cells/per cell config. The Cell outage is either in-service or out-of-service.

#### 1.2.4.3.5.3 o-du-mac-address

- ▶ **Parameter Name:** o-du-mac-address
- ▶ **Configuration Unit:** Per-cell config
- ▶ **Cell Outage:** Out-of-service
- ▶ **Description:** DU side mac address. It is translated to the corresponding 'nic' internally.
- ▶ **YANG Model:**
  - ▶ o-ran-uplane-conf.yang
  - ▶ o-ran-processing-element.yang
  - ▶ ietf-interfaces.yang
- ▶ **xpath:** /processing-elements/ru-elements/transport-flow/eth-flow/  
o-du-mac-address

#### 1.2.4.3.5.4 ru-mac-address

- ▶ **Parameter Name:** ru-mac-address
- ▶ **Configuration Unit:** Per-cell config
- ▶ **Cell Outage:** Out-of-service
- ▶ **Description:** MAC address of the corresponding RU
- ▶ **YANG Model:**
  - ▶ o-ran-uplane-conf.yang
  - ▶ o-ran-processing-element.yang
  - ▶ ietf-interfaces.yang
- ▶ **xpath:** /processing-elements/ru-elements/transport-flow/eth-flow/  
ru-mac-address

#### 1.2.4.3.5.5 vlan-id

- ▶ **Parameter Name:** vlan-id
- ▶ **Configuration Unit:** Per-cell config
- ▶ **Cell Outage:** Out-of-service
- ▶ **Description:** VLAN ID
- ▶ **YANG Model:**
  - ▶ ietf-interfaces.yang

- ▶ o-ran-interfaces.yang
- ▶ o-ran-processing-element.yang
- ▶ **xpath:** /processing-elements/ru-elements/transport-flow/eth-flow/vlan-id

#### 1.2.4.3.5.6 pcp

- ▶ **Parameter Name:** pcp
- ▶ **Configuration Unit:** Per-cell config
- ▶ **Cell Outage:** Out-of-service
- ▶ **Description:** VLAN priority level
- ▶ **YANG Model:**
  - ▶ ietf-interfaces.yang
  - ▶ o-ran-interfaces.yang
  - ▶ o-ran-processing-element.yang
- ▶ **xpath:** /interfaces/interface/class-of-service/u-plane-marking

#### 1.2.4.3.5.7 ul\_iq\_data\_fmt: bit\_width

- ▶ **Parameter Name:** ul\_iq\_data\_fmt: bit\_width
- ▶ **Configuration Unit:** Per-cell config
- ▶ **Cell Outage:** Out-of-service
- ▶ **Description:** Indicate the bit length after compression.
  - ▶ **BFP values:** 9 and 14; 16 for no compression
  - ▶ **Fixed point values:** Currently only supports 16
- ▶ **YANG Model:** o-ran-uplane-conf.yang
- ▶ **xpath:** /user-plane-configuration/low-level-tx-endpoints/compression/iq-bitwidth

#### 1.2.4.3.5.8 ul\_iq\_data\_fmt: comp\_meth

- ▶ **Parameter Name:** ul\_iq\_data\_fmt: comp\_meth
- ▶ **Configuration Unit:** Per-cell config
- ▶ **Cell Outage:** Out-of-service
- ▶ **Description:** Indicates the UL compression method.
  - ▶ BFP values: BLOCK\_FLOATING\_POINT
  - ▶ Fixed point values: NO\_COMPRESSION
- ▶ **YANG Model:** o-ran-uplane-conf.yang

- ▶ **xpath:** /user-plane-configuration/low-level-tx-endpoints/compression/compression-method

#### 1.2.4.3.5.9 dl\_iq\_data\_fmt: bit\_width

- ▶ **Parameter Name:** dl\_iq\_data\_fmt: bit\_width
- ▶ **Configuration Unit:** Per-cell config
- ▶ **Cell Outage:** Out-of-service
- ▶ **Description:** Indicates the bit length after compression.
  - ▶ BFP values: 9 and 14; 16 for no compression
  - ▶ Fixed point values: Currently only supports 16
- ▶ **YANG Model:** o-ran-uplane-confyang
- ▶ **xpath:** /user-plane-configuration/low-level-rx-endpoints/compression/iq-bitwidth

#### 1.2.4.3.5.10 dl\_iq\_data\_fmt: comp\_meth

- ▶ **Parameter Name:** dl\_iq\_data\_fmt: comp\_meth
- ▶ **Configuration Unit:** Per-cell config
- ▶ **Cell Outage:** Out-of-service
- ▶ **Description:** Indicates the DL compression method.
  - ▶ BFP values: BLOCK\_FLOATING\_POINT
  - ▶ Fixed point values: NO\_COMPRESSION
- ▶ **YANG Model:** o-ran-uplane-confyang
- ▶ **xpath:** /user-plane-configuration/low-level-rx-endpoints/compression/compression-method

#### 1.2.4.3.5.11 exponent\_dl

- ▶ **Parameter Name:** exponent\_dl
- ▶ **Configuration Unit:** Per-cell config
- ▶ **Cell Outage:** Out-of-service
- ▶ **Description:**
- ▶ **YANG Model:**
  - ▶ o-ran-uplane-confyang
  - ▶ o-ran-compression-factors.yang
- ▶ **xpath:** /user-plane-configuration/low-level-rx-endpoints/compression/exponent

#### 1.2.4.3.5.12 exponent\_ul

- ▶ **Parameter Name:** exponent\_ul
- ▶ **Configuration Unit:** Per-cell config
- ▶ **Cell Outage:** out-of-service
- ▶ **Description:**
- ▶ **YANG Model:**
  - ▶ o-ran-uplane-conf.yang
  - ▶ o-ran-compression-factors.yang
- ▶ **xpath:** /user-plane-configuration/low-level-tx-endpoints/compression/exponent

#### 1.2.4.3.5.13 Reference Examples

The following is a client-side reference implementation:

```
$cuBB_SDK/cuPHY-CP/cuphyoam/examples/p9_msg_client_grpc_test.cpp
```

Below are a few examples for update and retrieval of related params.

#### 1.2.4.3.5.14 Update ru-mac-address, vlan-id, and pcp

```
#step 1: Edit $cuBB_SDK/cuPHY-CP/cuphyoam/examples/mac_vlan_pcp.xml and update ru_mac,
↳vlan_id and pcp accordingly
#step 2: Run below cmd to do the provisioning
$cuBB_SDK/build/cuPHY-CP/cuphyoam/p9_msg_client_grpc_test --phy_id $mplane_id --cmd
↳edit_config --xml_file $cuBB_SDK/cuPHY-CP/cuphyoam/examples/mac_vlan_pcp.xml
#step 3: Run below cmds to retrieve the config
$cuBB_SDK/build/cuPHY-CP/cuphyoam/p9_msg_client_grpc_test --phy_id $mplane_id --cmd
↳get --xpath /o-ran-processing-element:processing-elements
$cuBB_SDK/build/cuPHY-CP/cuphyoam/p9_msg_client_grpc_test --phy_id $mplane_id --cmd
↳get --xpath /ietf-interfaces:interfaces
```

#### 1.2.4.3.5.15 Update o-du-mac-address(du nic port)

```
#step 1: Edit $cuBB_SDK/cuPHY-CP/cuphyoam/examples/nic_du_mac.xml and update du_mac,
↳which is translated to the corresponding nic port internally
#step 2: Run below cmd to do the provisioning
$cuBB_SDK/build/cuPHY-CP/cuphyoam/p9_msg_client_grpc_test --phy_id $mplane_id --cmd
↳edit_config --xml_file $cuBB_SDK/cuPHY-CP/cuphyoam/examples/nic_du_mac.xml
#step 3: Run below cmd to retrieve the config
$cuBB_SDK/build/cuPHY-CP/cuphyoam/p9_msg_client_grpc_test --phy_id $mplane_id --cmd
↳get --xpath /o-ran-processing-element:processing-elements
```

#### 1.2.4.3.5.16 Update DL/UL IQ data format

```
#step 1: Edit $cuBB_SDK/cuPHY-CP/cuphyoam/examples/iq_data_fmt.xml and update DL/UL IQ
↳data format accordingly
(compression-method: BLOCK_FLOATING_POINT for BFP or NO_COMPRESSION for fixed point)
(iq-bitwidth: 9, 14, 16 for BFP or 16 for fixed point)
#step 2: Run below cmd to do the provisioning
$cuBB_SDK/build/cuPHY-CP/cuphyoam/p9_msg_client_grpc_test --phy_id $mplane_id --cmd
↳edit_config --xml_file $cuBB_SDK/cuPHY-CP/cuphyoam/examples/iq_data_fmt.xml
#step 3: Run below cmd to retrieve the config
$cuBB_SDK/build/cuPHY-CP/cuphyoam/p9_msg_client_grpc_test --phy_id $mplane_id --cmd
↳get --xpath /o-ran-uplane-conf:user-plane-configuration
```

#### 1.2.4.3.5.17 Update dl and ul Exponent

```
#step 1: Edit $cuBB_SDK/cuPHY-CP/cuphyoam/examples/dl_ul_exponent.xml and dl and ul
↳exponent accordingly
#step 2: Run below cmd to do the provisioning
$cuBB_SDK/build/cuPHY-CP/cuphyoam/p9_msg_client_grpc_test --phy_id $mplane_id --cmd
↳edit_config --xml_file $cuBB_SDK/cuPHY-CP/cuphyoam/examples/dl_ul_exponent.xml
#step 3: Run below cmd to retrieve the config
$cuBB_SDK/build/cuPHY-CP/cuphyoam/p9_msg_client_grpc_test --phy_id $mplane_id --cmd
↳get --xpath /o-ran-uplane-conf:user-plane-configuration
```

### 1.2.4.3.6 Logging

#### 1.2.4.3.6.1 Log Levels

Nvlog supports the following log levels: Fatal, Error, Console, Warning, Info, Debug, and Verbose.

A Fatal log message results in process termination. For other log levels, the process continues execution. A typical deployment sends Fatal, Error, and Console levels to stdout. Console level is for printing something that is neither a warning nor an error, but you want to print to stdout.

#### 1.2.4.3.6.2 nvlog

This YAML container contains parameters related to nvlog configuration, see `nvlog_config.yaml`.

#### 1.2.4.3.6.3 name

Used to create the shared memory log file. Shared memory handle is `/dev/shm/${name}.log` and temp logfile is named `/tmp/${name}.log`.



#### 1.2.4.3.6.4 primary

In all processes logging to the same file, set the first starting process to be primary, set others to be secondary.

#### 1.2.4.3.6.5 shm\_log\_level

Sets the log level threshold for the high performance shared memory logger. Log messages with a level at or below this threshold are sent to the shared memory logger.

Log levels: 0 - NONE, 1 - FATAL, 2 - ERROR, 3 - CONSOLE, 4 - WARNING, 5 - INFO, 6 - DEBUG, 7 - VERBOSE

Setting the log level to LOG\_NONE means no logs are sent to the shared memory logger.

#### 1.2.4.3.6.6 console\_log\_level

Sets the log level threshold for printing to the console. Log messages with a level at or below this threshold are printed to stdout.

#### 1.2.4.3.6.7 max\_file\_size\_bits

Define the rotating log file `/var/log/aerial/${name}.log` size. Size =  $2^{\text{bits}}$ .

#### 1.2.4.3.6.8 shm\_cache\_size\_bits

Define the SHM cache file `/dev/shm/${name}.log` size. Size =  $2^{\text{bits}}$ .

#### 1.2.4.3.6.9 log\_buf\_size

Max log string length of one time call of the nvlog API.

#### 1.2.4.3.6.10 max\_threads

The maximum number of threads that are using nvlog all together.

#### 1.2.4.3.6.11 save\_to\_file

Whether to copy and save the SHM cache log to a rotating log file under `/var/log/aerial/` folder.

### 1.2.4.3.6.12 `cpu_core_id`

CPU core ID for the background log saving thread. -1 means the core is not pinned.

### 1.2.4.3.6.13 `prefix_opts`

bit5 - thread\_id bit4 - sequence number bit3 - log level bit2 - module type bit1 - date bit0 - time stamp  
Refer to `nvlog.h` for more details.

### 1.2.4.3.7 Metrics

The OAM Metrics API is used internally by cuPHY-CP components to report metrics (counters, gauges, and histograms). The metrics are exposed via a Prometheus Aerial exporter.

#### 1.2.4.3.7.1 Host Metrics

Host metrics are provided via the Prometheus node exporter. The node exporter provides many thousands of metrics about the host hardware and OS, such as but not limited to:

- ▶ CPU statistics
- ▶ Disk statistics
- ▶ Filesystem statistics
- ▶ Memory statistics
- ▶ Network statistics

See [https://github.com/prometheus/node\\_exporter](https://github.com/prometheus/node_exporter) and <https://prometheus.io/docs/guides/node-exporter/> for detailed documentation on the node exporter.

#### 1.2.4.3.7.2 GPU Metrics

GPU hardware metrics are provided through the GPU Operator via the Prometheus DCGM-Exporter. The DCGM-Exporter provides many thousands of metrics about the GPU and PCIe bus connection, such as but not limited to:

- ▶ GPU hardware clock rates
- ▶ GPU hardware temperatures
- ▶ GPU hardware power consumption
- ▶ GPU memory utilization
- ▶ GPU hardware errors including ECC
- ▶ PCIe throughput

See <https://github.com/NVIDIA/gpu-operator> for details on the GPU operator.

See <https://github.com/NVIDIA/gpu-monitoring-tools> for detailed documentation on the DCGM-Exporter.

An example Grafana dashboard is available at <https://grafana.com/grafana/dashboards/12239>.

### 1.2.4.3.7.3 Aerial Metric Naming Conventions

In addition to metrics available through the node exporter and DCGM-Exporter, Aerial exposes several application metrics.

Metric names are per <https://prometheus.io/docs/practices/naming/> and follows the format `aerial_<component>_<sub-component>_<metricdescription>_<units>`.

Metric types are per [https://prometheus.io/docs/concepts/metric\\_types/](https://prometheus.io/docs/concepts/metric_types/).

The component and sub-component definitions are in the table below. For each metric, the description, metric type, and metric tags are provided. Tags are a way of providing granularity to metrics without creating new metrics.

Component	Sub-Component	Description
cuphycp		cuPHY Control Plane application
	fapi	L2/L1 interface metrics
	cplane	Fronthaul C-plane metrics
	uplane	Fronthaul U-plane metrics
	net	Generic network interface metrics
cuphy		cuPHY L1 library
	pbch	Physical Broadcast Channel metrics
	pdsch	Physical Downlink Shared Channel metrics
	pdccch	Physical Downlink Common Channel metrics
	pusch	Physical Uplink Shared Channel metrics
	pucch	Physical Uplink Common Channel metrics
	prach	Physical Random Access Channel metrics

### 1.2.4.3.7.4 Metrics Exporter Port

Aerial metrics are exported on port 8081. Configurable in cuphycontroller YAML file via 'aerial\_metrics\_backend\_address'.

### 1.2.4.3.7.5 L2/L1 Interface Metrics

#### 1.2.4.3.7.6 aerial\_cuphycp\_slots\_total

Counts the total number of processed slots.

Metric type: counter

Metric tags:

- ▶ type: "UL" or "DL"
- ▶ cell: "cell number"

#### **1.2.4.3.7.7 aerial\_cuphycp\_fapi\_rx\_packets**

Counts the total number of messages L1 receives from L2.

Metric type: counter

Metric tags:

- ▶ msg\_type: "type of PDU"
- ▶ cell: "cell number"

#### **1.2.4.3.7.8 aerial\_cuphycp\_fapi\_tx\_packets**

Counts the total number of messages L1 transmits to L2.

Metric type: counter

Metric tags:

- ▶ msg\_type: "type of PDU"
- ▶ cell: "cell number"

#### **1.2.4.3.7.9 Fronthaul Interface Metrics**

##### **1.2.4.3.7.10 aerial\_cuphycp\_cplane\_tx\_packets\_total**

Counts the total number of C-plane packets transmitted by L1 over ORAN Fronthaul interface.

Metric type: counter

Metric tags:

- ▶ cell: "cell number"

##### **1.2.4.3.7.11 aerial\_cuphycp\_cplane\_tx\_bytes\_total**

Counts the total number of C-plane bytes transmitted by L1 over ORAN Fronthaul interface.

Metric type: counter

Metric tags:

- ▶ cell: "cell number"

##### **1.2.4.3.7.12 aerial\_cuphycp\_uplane\_rx\_packets\_total**

Counts the total number of U-plane packets received by L1 over ORAN Fronthaul interface.

Metric type: counter

Metric tags:

- ▶ cell: "cell number"

#### 1.2.4.3.7.13 aerial\_cuphycp\_uplane\_rx\_bytes\_total

Counts the total number of U-plane bytes received by L1 over ORAN Fronthaul interface.

Metric type: counter

Metric tags:

- ▶ cell: "cell number"

#### 1.2.4.3.7.14 aerial\_cuphycp\_uplane\_tx\_packets\_total

Counts the total number of U-plane packets transmitted by L1 over ORAN Fronthaul interface.

Metric type: counter

Metric tags:

- ▶ cell: "cell number"

#### 1.2.4.3.7.15 aerial\_cuphycp\_uplane\_tx\_bytes\_total

Counts the total number of U-plane bytes transmitted by L1 over ORAN Fronthaul interface.

Metric type: counter

Metric tags:

- ▶ cell: "cell number"

#### 1.2.4.3.7.16 aerial\_cuphycp\_uplane\_lost\_prbs\_total

Counts the total number of PRBs expected but not received by L1 over ORAN Fronthaul interface.

Metric type: counter

Metric tags:

- ▶ cell: "cell number"
- ▶ channel: One of "prach" or "pusch"

#### 1.2.4.3.7.17 NIC Metrics

#### 1.2.4.3.7.18 aerial\_cuphycp\_net\_rx\_failed\_packets\_total

Counts the total number of erroneous packets received.

Metric type: counter

Metric tags:

- ▶ nic: "nic port BDF address"

#### **1.2.4.3.7.19 aerial\_cuphycp\_net\_rx\_nombuf\_packets\_total**

Counts the total number of receive packets dropped due to the lack of free mbufs.

Metric type: Counter

Metric tags:

- ▶ nic: "nic port BDF address"

#### **1.2.4.3.7.20 aerial\_cuphycp\_net\_rx\_dropped\_packets\_total**

Counts the total number of receive packets dropped by the NIC hardware.

Metric type: Counter

Metric tags:

- ▶ nic: "nic port BDF address"

#### **1.2.4.3.7.21 aerial\_cuphycp\_net\_tx\_failed\_packets\_total**

Counts the total number of instances a packet failed to transmit.

Metric type: Counter

Metric tags:

- ▶ nic: "nic port BDF address"

#### **1.2.4.3.7.22 aerial\_cuphycp\_net\_tx\_accu\_sched\_missed\_interrupt\_errors\_total**

Counts the total number of instances accurate send scheduling missed an interrupt.

Metric type: Counter

Metric tags:

- ▶ nic: "nic port BDF address"

#### **1.2.4.3.7.23 aerial\_cuphycp\_net\_tx\_accu\_sched\_rearm\_queue\_errors\_total**

Counts the total number of accurate send scheduling rearm queue errors.

Metric type: Counter

Metric tags:

- ▶ nic: "nic port BDF address"

#### 1.2.4.3.7.24 `aerial_cuphycp_net_tx_accu_sched_clock_queue_errors_total`

Counts the total number accurate send scheduling clock queue errors.

Metric type: Counter

Metric tags:

- ▶ nic: "nic port BDF address"

#### 1.2.4.3.7.25 `aerial_cuphycp_net_tx_accu_sched_timestamp_past_errors_total`

Counts the total number of accurate send scheduling timestamp in the past errors.

Metric type: Counter

Metric tags:

- ▶ nic: "nic port BDF address"

#### 1.2.4.3.7.26 `aerial_cuphycp_net_tx_accu_sched_timestamp_future_errors_total`

Counts the total number of accurate send scheduling timestamp in the future errors.

Metric type: Counter

Metric tags:

- ▶ nic: "nic port BDF address"

#### 1.2.4.3.7.27 `aerial_cuphycp_net_tx_accu_sched_clock_queue_jitter_ns`

Current measurement of accurate send scheduling clock queue jitter, in units of nanoseconds.

Metric type: Gauge

Metric tags:

- ▶ nic: "nic port BDF address"

Details:

This gauge shows the TX scheduling timestamp jitter, that is, how far each individual Clock Queue (CQ) completion is from UTC time.

If you set CQ completion frequency to 2MHz (tx\_pp=500), you might see the following completions:

cqe 0 at 0 ns

cqe 1 at 505 ns

cqe 2 at 996 ns

cqe 3 at 1514 ns

...

tx\_pp\_jitter is the time difference between two consecutive CQ completions.

#### **1.2.4.3.7.28 aerial\_cuphycp\_net\_tx\_accu\_sched\_clock\_queue\_wander\_ns**

Current measurement of the divergence of Clock Queue (CQ) completions from UTC time over a longer time period (~8s).

Metric type: Gauge

Metric tags:

- ▶ nic: "nic port BDF address"

#### **1.2.4.3.7.29 Application Performance Metrics**

##### **1.2.4.3.7.30 aerial\_cuphycp\_slot\_processing\_duration\_us**

Counts the total number of slots with GPU processing duration in each 250us-wide histogram bin.

Metric type: Histogram

Metric tags:

- ▶ cell: "cell number"
- ▶ channel: one of "pbch", "pdcch", "pdsch", "prach", or "pusch"
- ▶ le: histogram less-than-or-equal-to 250us-wide histogram bins, for 250, 500, ..., 2000, +inf bins.

##### **1.2.4.3.7.31 aerial\_cuphycp\_slot\_pusch\_processing\_duration\_us**

Counts the total number of PUSCH slots with GPU processing duration in each 250us-wide histogram bin.

Metric type: Histogram

Metric tags:

- ▶ cell: "cell number"
- ▶ le: histogram less-than-or-equal-to 250us-wide histogram bins, range 0 to 2000us.

##### **1.2.4.3.7.32 aerial\_cuphycp\_pusch\_rx\_tb\_bytes\_total**

Counts the total number of transport block bytes received in the PUSCH channel.

Metric type: Counter

Metric tags:

- ▶ cell: "cell number"



#### 1.2.4.3.7.33 aerial\_cuphycp\_pusch\_rx\_tb\_total

Counts the total number of transport blocks received in the PUSCH channel.

Metric type: Counter

Metric tags:

- ▶ cell: "cell number"

#### 1.2.4.3.7.34 aerial\_cuphycp\_pusch\_rx\_tb\_crc\_error\_total

Counts the total number of transport blocks received with CRC errors in the PUSCH channel.

Metric type: Counter

Metric tags:

- ▶ cell: "cell number"

#### 1.2.4.3.7.35 aerial\_cuphycp\_pusch\_nrofuesperslot

Counts the total number of UEs processed in each slot per histogram bin PUSCH channel.

Metric type: Histogram

Metric tags:

- ▶ cell: "cell number"
- ▶ le: Histogram bin less-than-or-equal-to for 2, 4, ..., 24, +inf bins.

#### 1.2.4.3.7.36 PRACH Metrics

##### 1.2.4.3.7.37 aerial\_cuphy\_prach\_rx\_preambles\_total

Counts the total number of detected preambles in PRACH channel.

Metric type: Counter

Metric tags:

- ▶ cell: "cell number"

#### 1.2.4.3.7.38 PDSCH Metrics

##### 1.2.4.3.7.39 aerial\_cuphycp\_slot\_pdsch\_processing\_duration\_us

Counts the total number of PDSCH slots with GPU processing duration in each 250us-wide histogram bin.

Metric type: Histogram

Metric tags:

- ▶ cell: "cell number"

- ▶ le: histogram less-than-or-equal-to 250us-wide histogram bins, range 0 to 2000us.

#### **1.2.4.3.7.40 aerial\_cuphy\_pdsch\_tx\_tb\_bytes\_total**

Counts the total number of transport block bytes transmitted in the PDSCH channel.

Metric type: Counter

Metric tags:

- ▶ cell: "cell number"

#### **1.2.4.3.7.41 aerial\_cuphy\_pdsch\_tx\_tb\_total**

Counts the total number of transport blocks transmitted in the PDSCH channel.

Metric type: Counter

Metric tags:

- ▶ cell: "cell number"

#### **1.2.4.3.7.42 aerial\_cuphycp\_pdsch\_nrofuesperslot**

Counts the total number of UEs processed in each slot per histogram bin PDSCH channel.

Metric type: Histogram

Metric tags:

- ▶ cell: "cell number"
- ▶ le: Histogram bin less-than-or-equal-to for 2, 4, ..., 24, +inf bins.

## **1.3. cuBB Release Notes**

### **1.3.1. cuBB Software Manifest**

**Release Version: 24-3**

#### **1.3.1.1 Aerial CUDA-Accelerated RAN Software Manifest**

Description	Revision
Host OS	<ul style="list-style-type: none"> <li>▶ Grace Hopper platform: Ubuntu 22.04 with 6.5.0-1019-nvidia-64k kernel</li> <li>▶ x86 platform: Ubuntu 22.04 with 5.15.0-1042-nvidia-lowlatency kernel</li> </ul>
GH200	<ul style="list-style-type: none"> <li>▶ CUDA Toolkit: 12.6.2</li> <li>▶ GPU Driver (OpenRM): 560.35.03</li> </ul>
BF3 NIC	<ul style="list-style-type: none"> <li>▶ BFB: bf-bundle-2.7.0-33_24.04_ubuntu-22.04_prod.bfb</li> <li>▶ NIC FW: 32.41.1000</li> </ul>
DOCA OFED	24.04-0.6.6 (only required on Grace Hopper platform)
GDRCopy	2.4.1
DOCA	2.7 (included in cuBB container)
DPDK	22.11 (Included in Mellanox DOCA)
NV Container Toolkit	1.16.2
SCF	222.10.02 (partial upgrade to 222.10.04)
Server	<ul style="list-style-type: none"> <li>▶ Supermicro Grace Hopper MGX ARS-111GL-NHR (Config 2)</li> <li>▶ Dell PowerEdge R750 with dual Intel(R) Xeon(R) Gold 6336Y CPU @ 2.40GHz</li> <li>▶ Gigabyte(E251-U70) with Intel(R) Xeon(R) Gold 6240R CPU @ 2.40GHz (EOL)</li> </ul>
GPU	GH200

**Note**

- ▶ Aerial support of AX800, A100X, A100, CX6-DX has reached End of Life (EOL).
- ▶ Aerial has been using DMA-buf, inbox driver and OpenRM driver since 23-4 release. So MOFED and nvidia-peermem are not needed anymore. On the x86 platform, the 5.15 kernel with DMA-buf and inbox driver are used. On the Grace Hopper platform, the 6.2 kernel with DMA-buf and DOCA OFED are used.

### 1.3.1.2 Kubernetes Software Manifest

Description	Revision
Host OS	Grace Hopper platform: Ubuntu 22.04 with 6.5.0-1019-nvidia-64k kernel
Container OS	Ubuntu 22.04
Containerd	1.5.8
Kubernetes	1.23
Helm	3.8
BF3 NIC FW	32.41.1000
GPU Operator	24.6.2
CUDA Toolkit	12.6.2
NVIDIA GPU Driver	560.35.03

## 1.3.2. Supported Features and Configurations

This release of the Aerial cuBB supports the following configurations and features. These features are verified with test vectors in a simulated environment using TestMAC and RU emulator.

### 1.3.2.1 PUSCH

- ▶ SU-MIMO layers: up to 4
- ▶ MU-MIMO layers: up to 8
- ▶ Modulation and coding rates: MCS 0 – MCS 27
- ▶ Optimized LDPC decoder
- ▶ UCI on PUSCH (HARQ up to 11 bits + CSI part 1 + CSI part 2 up to 11 bits)
- ▶ Time-interpolated channel estimation and equalization
- ▶ SINR reporting to L2
- ▶ MMSE-IRC receiver
- ▶ Early HARQ in UCI.indication

### 1.3.2.2 PUCCH

- ▶ Format 0 + DTX detection
- ▶ Format 1 + DTX detection
- ▶ Format 2 (unsegmented payload) + DTX detection
- ▶ Format 3 (unsegmented payload) + DTX detection
- ▶ SINR / confidence level reporting to L2

### 1.3.2.3 PRACH

- ▶ Format 0
- ▶ Format B4 (multiple per slot in FDM)
- ▶ Interference level reporting

### 1.3.2.4 PDSCH

- ▶ SU-MIMO layers: up to 4
- ▶ MU-MIMO layers: up to 16
- ▶ Modulation and coding rates: MCS 0 – MCS 27
- ▶ Supports Cat-A O-RAN split and Cat-B O-RAN split. For Cat-A O-RAN split, PDSCH is implemented up to modulation and precoding (identity matrix precoder) For Cat-B O-RAN split, PDSCH is implemented up to the rate matching block.
- ▶ Precoding (4 layers)

### 1.3.2.5 PDCCH

- ▶ Interleaved and non-interleaved mode
- ▶ Aggregation level (AL) 1, 2, 4, 8, 16
- ▶ 1, 2, 3 symbol CORESET
- ▶ Precoding (1 layer)

### 1.3.2.6 SS Block

- ▶ PSS, SSS generation
- ▶ DMRS and PBCH generation and time-frequency mapping
- ▶ Precoding (1 layer)

### 1.3.2.7 CSI-RS

- ▶ NZP-CSI-RS
- ▶ ZP-CSI-RS
- ▶ Precoding (1 layer)

### 1.3.2.8 SRS

- ▶ Support SRS reporting for upto 64T64R BB Antenna ports.
- ▶ Support SRS reporting according to 5G FAPI 222.10.04 for beamManagement, codebook and non-codebook SRS usage.
- ▶ Support SRS reporting according to 5G FAPI 222.10.02 for SINR reporting.

### 1.3.2.9 MIMO Features

- ▶ Support 64 Transmit and Receive antenna ports
- ▶ Support SRS-based channel estimation, buffering and FAPI-compliant reporting to L2
- ▶ Support PUSCH and PDSCH Dynamic beamforming weight (BFW) calculation from SRS channel estimates (regularized zero-forcing)
- ▶ Support up to 8 layers multi-user MIMO PUSCH
- ▶ Support up to 16 layers multi-user MIMO PDSCH
- ▶ Support of upto 3 cells with multi-user MIMO configuration

### 1.3.2.10 LDPC Decoder

- ▶ Standalone LDPC decoder

### 1.3.2.11 SHM Logger

- ▶ Support for C++ `std::format` style logging like `std::format("{} {}!", "Hello", "world", "something");`
- ▶ Support for C (`printf`) style formatted strings.

## 1.3.3. Multicell Capacity

CPU core usage for multicell benchmark:

On Grace Hopper:

- ▶ 1 isolated physical Grace core for core-locked PTP applications (phc2sys+ptp4l)
- ▶ 10 additional isolated Grace cores for the other core-locked cuphycontroller threads

On x86 based targets:

Without hyperthreading using “1+6+fractional” x86 cores. The shorthand “1+6+fractional” x86 cores is defined as follows:

- ▶ 1 isolated physical x86 core for core-locked PTP applications (phc2sys+ptp4l) and the core-locked cuphycontroller L2A H2D prepone thread
- ▶ 6 additional isolated physical x86 cores for the other core-locked cuphycontroller threads
- ▶ A fraction of a shared floating x86 core for non-core-locked cuphycontroller threads

Additionally, the tested L2 timeline is as follows:

- ▶ FAPI SLOT.indication for Slot N is sent from L1 to L2 at the wall-clock time for Slot N-3 (i.e. 3 slot advance).
- ▶ For the DDDSUUDDDD TDD pattern with 0-based slot numbering, L2 has up-to-500us, for slot%10 in {2,3,4,5,6}, from SLOT.indication to deliver all FAPI PDUs for Slot N.
- ▶ For the DDDSUUDDDD TDD pattern with 0-based slot numbering, L2 has up-to-250us, for slot%10 in {0,1,7,8,9}, from SLOT.indication to deliver all FAPI PDUs for Slot N.

As of 24-2:

Supports 500us L2 processing budget and 7 beam peak and average patterns as defined below using 100MHz:

On Grace Hopper:

- ▶ BFP9: 20 4T4R Peak cells / 20 4T4R average cells

while respecting the following configuration for 7 beam traffic patterns:

Table 15: TDD 4T4R - 80 Slot Traffic Models

7-beam config	Configuration (4 UL streams RU->DU)	
	Peak	Average
Compression	BFP9 and BFP14	BFP9 and BFP14
Max PxsCH PRB	270	132
DL Throughput/cell	1469.14 Mbps	523.10 Mbps
UL Throughput/cell	212.64 Mbps	79.91 Mbps
Peak DL Fronthaul Bandwidth / cell	11.06 Gbps BFP14	5.46 Gbps BFP14
	7.14 Gbps BFP9	3.58 Gbps BFP9
Peak UL Fronthaul Bandwidth / cell	11.88 Gbps BFP14	6.34 Gbps BFP14
	8.03 Gbps BFP9	4.57 Gbps BFP9
SSB slots	Frame 0 & 2: 0,1,2,3	Frame 0 & 2: 0,1,2,3
#SSB per slot	Frame 0 & 2: 2,2,2,1	Frame 0 & 2: 2,2,2,1
TRS slots	Frame 0-3: 6,7,8,9,10,11	Frame 0-3: 6,7,8,9,10,11
	Frame 0 & 2: 16,17	Frame 0 & 2: 16,17
TRS Symbols	Even cells: 6,10	Even cells: 6,10
	Odd cells: 5,9	Odd cells: 5,9
CSI-RS slots	Frame 0: 8,10,16	Frame 0: 8,10,16
	Frame 1: 6,8,10	Frame 1: 6,8,10
	Frame 2: 6	Frame 2: 6
CSI-RS Symbols	Even cells: 12	Even cells: 12
	Odd cells: 13	Odd cells: 13
PDSCH #DCI	12 (6 DL + 6 UL per slot)	12 (6 DL + 6 UL per slot)
UE/TTI/Cell	6 per DL slot, 6 per UL slot	6 per DL slot, 6 per UL slot
UCI on PUSCH HARQ+CSIP1+CSIP2 (bits)	4+37+5	4+37+5
PUCCH format	1	1
PUCCH payload (bits)	18	18
PRACH format	B4	B4
PRACH slots	Frame 0-3: 5, 15	Frame 0-3: 5, 15
PRACH occasions	Slot 5: 4, Slot 15: 3	Slot 5: 4, Slot 15: 3

**Note**

Stated performance achievement and CPU core count usage is for L1 workload only (additional non-L1 workloads in E2E setting may have an impact on the achieved performance and/or CPU core count usage)



**Note**

Performance achievement is measured by running L1 in steady-state traffic mode (e.g. impact of workloads such as cell reconfiguration on other cells is not captured)

As of 24-3 additionally:

On Grace Hopper:

- ▶ BFP14: 3 64T64R Peak cells / 64T64R average cells are supported functionally.

Notes:

- ▶ Currently performance evaluation is on going for the multi-cell mu-MIMO and benchmarking for the same will be published in the next release.

## 1.3.4. Supported Test Vector Configurations

This release of Aerial cuBB currently supports the following test-vector configurations.

### 1.3.4.1 PUSCH

TC start	TC End	Description	TV Generated	cuPHY Pass	cuBB Pass
7201	7201	base	1	1	1
7202	7203	mcsTable	2	2	2
7204	7204	mcs	1	1	1
7205	7207	num of layers	3	3	3
7208	7208	rb0, Nrb	1	1	1
7209	7210	sym0	2	2	2
7211	7211	dmsr0	1	1	1
7212	7213	Nsym	2	2	2
7214	7214	SCID	1	1	1
7215	7215	BWP0, nBWP	1	1	1
7216	7216	RNTI	1	1	1
7217	7219	addPos	3	3	3
7220	7220	dataScId	1	1	1
7221	7222	maxLen	2	2	2
7223	7223	dmrsScId	1	1	1
7224	7224	nCdm	1	1	1

continues on next page

Table 16 – continued from previous page

TC start	TC End	Description	TV Gener- ated	cuPHY Pass	cuBB Pass
7225	7225	port0	1	1	1
7227	7227	nAnt=2	1	1	1
7228	7228	nAnt=16	1	1	0
7229	7229	slotIdx	1	1	1
7230	7232	rvIdx	3	3	3
7233	7235	FDM	3	3	3
7236	7241	CDM	6	6	6
7242	7244	rvIdx>0/BGN=1	3	3	3
7245	7245	ulGridSize=106	1	1	0
7246	7247	dmrs_par per Ueg	2	2	2
7248	7250	additional FDM	3	3	3
7251	7257	precoding	7	7	7
7258	7260	mapping type B	3	3	3
7261	7272	Flexible DMRS ports	12	12	12
7273	7273	MCS > 28	1	1	1
7274	7279	additional nCDM=1	6	6	6
7280	7283	Flexible SLIV	4	4	4
7301	7320	multi-params	20	20	18
7321	7323	LBRM	3	3	3
7324	7326	HARQ-rx	3	3	0
7327	7330	8/16 UEs	4	4	4
7331	7338	multiple layers	8	8	8
7340	7340	Multi-layers with nAnt=16	1	1	0
7401	7403	CFO	3	3	3
7404	7406	TO	3	3	3
7407	7407	RSSI	1	1	1
7408	7408	CFO w/ SDM	1	1	1
7409	7409	TO w/ SDM	1	1	1
7410	7411	CEE-TDI	2	2	2
7412	7413	rx power	2	2	2
7414	7414	TDI maxLen = 2	1	1	1
7415	7417	small/big/zero rx	3	3	3

continues on next page

Table 16 – continued from previous page

TC start	TC End	Description	TV Gener- ated	cuPHY Pass	cuBB Pass
7418	7419	additional TDI	2	2	2
7420	7426	IRC=0	7	7	7
7427	7432	SINR meas	6	6	6
7501	7516	UCI on PUSCH (w/o data)	16	16	16
7517	7530	UCI on PUSCH (w/ data)	14	14	14
7531	7531	UciOnPusch DTX	1	1	1
7532	7532	UciOnPusch CRC fail	1	1	1
7533	7534	UciOnPusch addPos	2	2	2
7551	7570	UciOnPusch (multi-params)	20	20	20
7571	7575	UCI w/ and w/o data	5	5	5
7601	7613	FR1 BW mu = 1	13	13	13
7614	7621	FR1 BW mu = 0	8	8	0
7901	7901	demo_msg3	1	1	1
7902	7902	demo_traffic_ul	1	1	1
7903	7904	UciOnPusch conformance	0	0	0
7016	7153	sweep Zc/mcs (skip 7016,7017,7024,7025,7032,7033)	130	130	130

### 1.3.4.2 PUCCH

TC Start	TC End	Description	TV Gener- ated	cuPHY Pass	cuBB Pass
6001	6003	bases for format 0	3	3	3
6004	6010	vary single parameter for format 0	7	7	7
6011	6040	vary multiple parameters for format 0	30	30	30
6041	6046	vary slotIdx (single-UCI) for format 0	6	6	6
6047	6056	multi-UCI tests for format 0	10	10	10
6057	6061	vary slotIdx (multi-UCI) for format 0	5	5	5
6101	6103	bases for format 1	3	3	3

continues on next page

Table 17 – continued from previous page

TC Start	TC End	Description	TV Gener- ated	cuPHY Pass	cuBB Pass
6104	6116	vary single parameter for format 1	13	13	13
6117	6146	vary multiple parameters for format 1	30	30	30
6147	6155	vary slotIdx (single-UCI) for format 1	9	9	9
6156	6173	multi-UCI tests for format 1	18	18	18
6175	6192	TA estimation for format 1	18	18	18
6193	6194	192 UCI groups for format 1	2	2	2
6201	6203	bases for format 2	3	3	3
6204	6219	test Nf for format 2	16	16	16
6220	6235	test Nt and freq hopping for format 2	16	16	16
6236	6236	11 info bits and 2 PRBS for format 2	1	1	1
6239	6245	different payload sizes for format 2	7	7	7
6301	6310	bases for format 3	10	10	10
6311	6313	multi-UCI tests for format 3	3	3	3
6314	6324	tests with freqHop enabled for format 3	11	11	11
6325	6335	tests with freqHop disabled for format 3	11	11	11
6336	6346	tests with add'l DMRS position, freqHop enabled for format 3	11	11	11
6347	6357	tests with add'l DMRS position, freqHop disabled for format 3	11	11	11
6358	6364	different payload sizes for format 3	7	7	7
6365	6373	24-UCI tests for format 3	9	9	9
6501	6513	sweep different bandwidth for format 0, mu = 1	13	13	13
6514	6526	sweep different bandwidth for format 1, mu = 1	13	13	13
6527	6539	sweep different bandwidth for format 2, mu = 1	13	13	13

continues on next page

Table 17 – continued from previous page

TC Start	TC End	Description	TV Gener- ated	cuPHY Pass	cuBB Pass
6540	6552	sweep different bandwidth for format 3, mu = 1	13	13	13
6553	6560	sweep different bandwidth for format 0, mu = 0	8	8	0
6561	6568	sweep different bandwidth for format 1, mu = 0	8	8	0
6569	6576	sweep different bandwidth for format 2, mu = 0	8	8	0
6577	6584	sweep different bandwidth for format 3, mu = 0	8	8	0
6585	6586	rx power for format 0	2	2	2
6587	6588	rx power for format 1	2	2	2
6589	6590	rx power for format 2	2	2	2
6591	6592	rx power for format 3	2	2	2
6593	6595	very small/very big/forcRxZero rx power for format 0	3	3	3
6596	6598	very small/very big/forcRxZero rx power for format 1	3	3	3
6599	6601	very small/very big/forcRxZero rx power for format 2	3	3	3
6602	6605	very small/very big/forcRxZero rx power for format 3	4	4	4
6801	6802	perf TV F08	2	2	2
6803	6804	perf TV F14	2	2	2

### 1.3.4.3 PRACH

TC Start	TC End	Description	TV Gener- ated	cuPHY Pass	cuBB Pass
5001	5001	base	1	1	1
5002	5002	format 0	1	1	0
5003	5003	rootIdx	1	1	1
5004	5004	zoneldx	1	1	1
5005	5005	prmbIdx	1	1	1

continues on next page

Table 18 – continued from previous page

TC Start	TC End	Description	TV Generated	cuPHY Pass	cuBB Pass
5006	5006	Nant	1	1	0
5007	5007	N_nc	1	1	1
5008	5008	delay	1	1	1
5009	5009	SNR	1	1	1
5010	5010	CFO	1	1	1
5011	5011	2-UE	1	1	1
5012	5012	4-UE	1	1	1
5013	5013	4FDM/16UE	1	1	1
5014	5018	rx power	5	5	5
5101	5101	FDD,mu=0,B4,nAnt=2	1	1	0
5102	5102	FDD,mu=1,B4,nAnt=4	1	1	1
5103	5103	TDD,mu=0,B4,nAnt=8	1	1	0
5104	5104	TDD,mu=1,B4,nAnt=16	1	1	0
5105	5105	FDD,mu=0,F0,nAnt=16	1	1	0
5106	5106	FDD,mu=1,F0,nAnt=8	1	1	0
5107	5107	TDD,mu=0,F0,nAnt=4	1	1	0
5108	5108	TDD,mu=1,F0,nAnt=2	1	1	0
5201	5213	FR1 BW mu = 1	13	13	13
5214	5221	FR1 BW mu = 0	8	8	0
5801	5802	perf TV F08	2	2	2
5803	5804	perf TV F14	2	2	0
5901	5901	demo_msg1	1	1	1
5911	5914	comformance TC	4	4	1

#### 1.3.4.4 PDSCH

TC Start	TC End	Description	TV Generated	cuPHY Pass	cuBB Pass
3201	3201	base	1	1	1
3202	3203	mcsTable	2	2	2
3204	3204	mcs	1	1	1

continues on next page

Table 19 – continued from previous page

TC Start	TC End	Description	TV Gener- ated	cuPHY Pass	cuBB Pass
3205	3207	num of layers	3	3	3
3208	3208	rb0, Nrb	1	1	1
3209	3210	sym0	2	2	2
3211	3211	dmrs0	1	1	1
3212	3213	Nsym	2	2	2
3214	3214	SCID	1	1	1
3215	3215	BWP0, nBWP	1	1	1
3216	3216	RNTI	1	1	1
3217	3219	addPos	3	3	3
3220	3220	dataScId	1	1	1
3221	3222	maxLen	2	2	2
3223	3223	dmrsScId	1	1	1
3224	3224	nCdm	1	1	1
3225	3225	port0	1	1	1
3226	3228	nAnt	3	3	3
3229	3229	slotIdx	1	1	1
3230	3232	rvIdx	3	3	3
3233	3235	FDM	3	3	3
3236	3241	SDM/SCID	6	6	6
3242	3244	rvIdx>0/BGN=1	3	3	3
3245	3245	dIGridSize=106	1	1	0
3246	3247	dmrs_par per Ueg	2	2	2
3248	3254	precoding	7	7	7
3255	3257	mapping type B	3	3	3
3258	3260	mixed precoding	3	3	3
3261	3261	refPoint	1	1	1
3262	3262	TxPower	1	1	1
3263	3263	modComp	1	0	0
3264	3264	precoding (mixed nPorts)	1	1	1
3265	3265	TxPower with 2 UEs	1	1	1
3266	3267	different rv	2	2	2
3268	3269	multi-layer	2	2	2

continues on next page

Table 19 – continued from previous page

TC Start	TC End	Description	TV Generated	cuPHY Pass	cuBB Pass
3271	3276	nCDM = 1	6	6	6
3321	3322	LBRM	2	2	2
3323	3333	RE map from CSI-RS	11	11	11
3334	3336	8/16 UEs (SU-MIMO)	3	3	3
3337	3337	16 UEs (MU-MIMO)	1	1	1
3401	3413	FR1 BW mu = 1	13	13	13
3414	3421	FR1 BW mu = 0	8	8	0
3901	3901	demo_coreset0	1	1	1
3902	3902	demo_msg2	1	1	1
3903	3903	demo_msg4	1	1	1
3904	3904	demo_traffic_dl	1	1	1
3001	3015	multi-params	15	15	15
3016	3154	sweep Zc/mcs (3016,3017,3024,3025,3032,3033 are skipped)	131	131	131

1.3.4.5 PDCCH

TC Start	TC End	Description	TV Generated	cuPHY Pass	cuBB Pass
2001	2001	base	1	1	1
2002	2002	slotIdx	1	1	1
2003	2003	nBWP	1	1	1
2004	2004	BPW0	1	1	1
2005	2005	sym0	1	1	1
2006	2007	Nsym	2	2	2
2008	2009	crstIdx	2	2	2
2010	2010	intl	1	1	1
2011	2012	nBndl	2	2	2
2013	2014	nIntl	2	2	2
2015	2015	nShift	1	1	1
2016	2016	isCSS	1	1	1

continues on next page



Table 20 – continued from previous page

TC Start	TC End	Description	TV Gener- ated	cuPHY Pass	cuBB Pass
2017	2017	rnti	1	1	1
2018	2018	scrblD	1	1	1
2019	2019	scrBRnti	1	1	1
2020	2022	aggrL	3	3	3
2023	2023	dbQam	1	1	1
2024	2024	dbDmrs	1	1	1
2025	2025	Npayload	1	1	1
2026	2027	crstMap	2	2	2
2028	2028	nDCI	1	1	1
2029	2029	Npayload	1	1	1
2030	2030	aggrL	1	1	1
2031	2031	precoding	1	1	1
2032	2032	modComp	1	0	0
2033	2033	multi-PDCCH	1	1	1
2101	2112	multi-params	12	12	12
2201	2213	FR1 BW mu = 1	13	13	13
2214	2221	FR1 BW mu = 0	8	8	0
2801	2802	perf TV F14	2	2	2
2803	2804	perf TV F08	2	2	2
2805	2806	perf TV F09	2	2	2
2901	2901	demo_msg2	1	1	1
2902	2902	demo_msg4	1	1	1
2903	2903	demo_coreset0	1	1	1
2904	2904	demo_traffic_dl	1	1	1
2905	2905	demo_msg5	1	1	1

## 1.3.4.6 SS Block

TC Start	TC End	Description	TV Generated	cuPHY Pass	cuBB Pass
1001	1001	base	1	1	1
1002	1002	mu = 0	1	1	0
1003	1003	N_CELL_ID	1	1	1
1004	1004	n_hf = 1	1	1	1
1005	1005	L_max = 4	1	1	1
1006	1006	k_SSB	1	1	1
1007	1007	offsetPointA	1	1	1
1008	1008	SFN	1	1	1
1009	1009	blockIdx	1	1	1
1010	1010	precoding	1	1	1
1011	1011	betaPss	1	1	1
1101	1101	mu=0, 1SSB	1	1	0
1102	1102	mu=1, 1SSB	1	1	1
1103	1103	mu=1, 2SSB	1	1	0
1104	1104	mu=1, 2SSB	1	1	1
1202	1213	FR1 BW, mu = 1	12	12	12
1214	1221	FR1 BW, mu = 0	8	8	0
1801	1801	Perf TV	1	1	1
1901	1901	demo_ssb	1	1	1
1902	1902	for CP pipeline	1	1	1

## 1.3.4.7 CSI-RS

TC Start	TC End	Description	TV Generated	cuPHY Pass	cuBB Pass
4001	4004	nPorts = 1	4	4	4
4005	4007	nPorts = 2	3	3	3
4008	4009	nPorts = 4	2	2	2
4010	4012	nPorts = 8	3	3	3
4013	4038	nPorts > 8, row > 8	26	26	0

continues on next page

Table 22 – continued from previous page

TC Start	TC End	Description	TV Gener- ated	cuPHY Pass	cuBB Pass
4039	4039	RB0	1	1	1
4040	4040	nRB	1	1	1
4041	4041	sym0	1	1	1
4042	4042	sym1	1	1	0
4043	4043	nID	1	1	1
4044	4044	power control	1	1	1
4045	4050	freqDomainAllocation	6	6	5
4051	4051	idxSlot	1	1	1
4052	4054	batching	3	3	3
4055	4055	small gird size	1	1	0
4056	4056	TRS	1	1	1
4057	4057	precoding	1	1	1
4058	4058	modComp	1	0	0
4059	4060	16/32 CSIRS PDUs	2	2	2
4101	4103	multiple parameters	3	3	3
4201	4213	FR1 BW mu = 1	13	13	13
4214	4221	FR1 BW mu = 0	8	8	0
4801	4801	perf TV F08	1	1	1
4802	4802	perf TV F09	1	1	1
4803	4803	perf TV F14	1	1	0

#### 1.3.4.8 SRS

TC Start	TC End	Description	TV Gener- ated	cuPHY Pass	cuBB Pass
8001	8001	base	1	1	1
8002	8002	rnti	1	1	1
8003	8003	Nap=2	1	1	1
8004	8004	Nap=4	1	1	1
8005	8005	nSym=2	1	1	1
8006	8006	nSym=4	1	1	1

continues on next page

Table 23 – continued from previous page

TC Start	TC End	Description	TV Gener- ated	cuPHY Pass	cuBB Pass
8007	8007	Nrep=2	1	1	1
8008	8008	Nrep=4	1	1	1
8009	8009	sym0	1	1	1
8010	8010	cfgIdx	1	1	1
8011	8011	seqId	1	1	1
8012	8012	bwIdx=1	1	1	1
8013	8013	bwIdx=2	1	1	1
8014	8014	bwIdx=3	1	1	1
8015	8015	cmbSize	1	1	1
8016	8016	cmbOffset	1	1	1
8017	8017	cyclic shift	1	1	1
8018	8018	freqPosition	1	1	1
8019	8019	freqShift	1	1	1
8020	8020	freqHopping=1	1	1	1
8021	8021	freqHopping=2	1	1	1
8022	8022	freqHopping=3	1	1	1
8023	8023	grpSeqHopping=1	1	1	1
8024	8024	grpSeqHopping=2	1	1	1
8025	8025	rsrcType,Tsrs,Toffset	1	1	0
8026	8026	idxSlot	1	1	1
8027	8033	multi-SRS	1	1	1
8034	8034	rsrcType,Tsrs,Toffset	1	1	0
8035	8035	16 users wideband	1	1	1
8051	8057	multiple parameters	7	7	7
8101	8164	sweep cfgIdx	64	64	64
8201	8213	FR1 BW mu=1	13	13	13
8214	8221	FR1 BW mu=0	8	8	0
8222	8226	rx power	5	5	5
8227	8227	additional BW	1	1	1
8301	8302	SRS integration	2	2	2
8401	8415	32 nAnt	15	15	15
8420	8421	32 nAnt	2	2	2

continues on next page

Table 23 – continued from previous page

TC Start	TC End	Description	TV Generated	cuPHY Pass	cuBB Pass
8501	8524	64 nAnt	24	24	24
8801	8801	F09 perf TV	1	1	1
8802	8802	20M perf TV	1	1	1

#### 1.3.4.9 mSlot\_mCell

TC Start	TC End	Description	TV Generated	cuPHY Pass	cuBB Pass
90001	90007	single channel	7	7	7
90011	90012	dlnmix/ulmix	2	2	2
90013	90015	s-slot	3	3	3
90016	90018	multi-cell base case	3	3	3
90019	90019	prcd+noPrcd	1	1	1
90020	90020	BFP14+BFP9	1	1	1
90021	90022	HARQ	2	2	2
90023	90023	empty slot	1	1	1
90032	90037	multi-slot combo TC	6	6	6
90041	90046	SRS + UL + DL	6	6	6
90051	90056	mixed cells	6	6	6
90057	90058	adaptive re-tx	2	2	2
90060	90060	SRS even/odd frames	1	1	1
90501	90505	bug TCs	5	5	5
90601	90603	multi-channel TCs	3	3	3

#### 1.3.4.10 LDPC Performance

The `ldpc\_perf\_collect.py` Python script from the cuPHY repository can be used to perform error rate tests for the cuPHY LDPC decoder. There are test input files defined for  $Z = [64, 128, 256, 384]$ ,  $BG = [1, 2]$ . The tests check whether the block error rate (BLER, also sometimes referred to as Frame Error Rate or FER) is less than 0.1.

From the build directory, the following commands run the tests:

```
../util/ldpc/ldpc\_perf\_collect.py --mode test -i
../util/ldpc/test/ldpc\_decode\_BG1\_Z64\_BLER0.1.txt -f -w 800 -P
```

(continues on next page)

(continued from previous page)

```

../util/ldpc/ldpc\_perf\_collect.py --mode test -i
../util/ldpc/test/ldpc\_decode\_BG1\_Z128\_BLER0.1.txt -f -w 800 -P

../util/ldpc/ldpc\_perf\_collect.py --mode test -i
../util/ldpc/test/ldpc\_decode\_BG1\_Z256\_BLER0.1.txt -f -w 800 -P

../util/ldpc/ldpc\_perf\_collect.py --mode test -i
../util/ldpc/test/ldpc\_decode\_BG1\_Z384\_BLER0.1.txt -f -w 800 -P

../util/ldpc/ldpc\_perf\_collect.py --mode test -i
../util/ldpc/test/ldpc\_decode\_BG2\_Z64\_BLER0.1.txt -f -w 800 -P

../util/ldpc/ldpc\_perf\_collect.py --mode test -i
../util/ldpc/test/ldpc\_decode\_BG2\_Z128\_BLER0.1.txt -f -w 800 -P

../util/ldpc/ldpc\_perf\_collect.py --mode test -i
../util/ldpc/test/ldpc\_decode\_BG2\_Z256\_BLER0.1.txt -f -w 800 -P

../util/ldpc/ldpc\_perf\_collect.py --mode test -i
../util/ldpc/test/ldpc\_decode\_BG2\_Z384\_BLER0.1.txt -f -w 800 -P
    
```

Each test input file contains multiple tests for different code rates, as specified by the number of parity nodes. The format of the input files has the following form:

#	BG	Z	num\_parity	num\_iter	SNR	max\_BER	max\_BLER
1	384	4	10	6.87	1	0.1	
1	384	5	10	6.15	1	0.1	
1	384	6	10	5.64	1	0.1	
1	384	7	10	5.17	1	0.1	
1	384	8	10	4.79	1	0.1	
...							

After running each of the test cases, the `ldpc\_perf\_collect.py` script displays an output table:

```

+-----+
| # BG      Z num\_parity num\_iter      SNR      max\_BER      BER      max\_BLER
| BLER STATUS |
|
| 1 384      4      10      6.870 1.000000e+00 4.833980e-04 1.000000e-01 8.
| 750000e-02 PASS |
|
| 1 384      5      10      6.150 1.000000e+00 1.481120e-04 1.000000e-01 7.
| 250000e-02 PASS |
|
| 1 384      6      10      5.640 1.000000e+00 5.652230e-05 1.000000e-01 8.
| 000000e-02 PASS |
|
|
    
```

(continues on next page)

(continued from previous page)

	1	384	7	10	5.170	1.000000e+00	7.886480e-05	1.000000e-01	8.
↪	750000e-02	PASS							
↪									
	1	384	8	10	4.790	1.000000e+00	1.673470e-04	1.000000e-01	8.
↪	375000e-02	PASS							
↪									
	1	384	9	10	4.480	1.000000e+00	1.185190e-04	1.000000e-01	7.
↪	625000e-02	PASS							
↪									
	1	384	10	10	4.200	1.000000e+00	8.552320e-05	1.000000e-01	8.
↪	875000e-02	PASS							
↪									
	1	384	11	10	3.920	1.000000e+00	5.385890e-05	1.000000e-01	8.
↪	375000e-02	PASS							
↪									
	1	384	12	10	3.660	1.000000e+00	1.234020e-04	1.000000e-01	9.
↪	125000e-02	PASS							
↪									
	1	384	13	10	3.450	1.000000e+00	7.013490e-05	1.000000e-01	8.
↪	000000e-02	PASS							
↪									
	1	384	14	10	3.220	1.000000e+00	7.620150e-05	1.000000e-01	8.
↪	125000e-02	PASS							
↪									
	1	384	15	10	3.020	1.000000e+00	5.800190e-05	1.000000e-01	7.
↪	250000e-02	PASS							
↪									
	1	384	16	10	2.830	1.000000e+00	8.774270e-05	1.000000e-01	8.
↪	375000e-02	PASS							
↪									
	1	384	17	10	2.640	1.000000e+00	4.838420e-05	1.000000e-01	7.
↪	750000e-02	PASS							
↪									
	1	384	18	10	2.500	1.000000e+00	3.950640e-05	1.000000e-01	7.
↪	875000e-02	PASS							
↪									
	1	384	19	10	2.310	1.000000e+00	3.551140e-05	1.000000e-01	8.
↪	375000e-02	PASS							
↪									
	1	384	20	10	2.150	1.000000e+00	2.500590e-05	1.000000e-01	8.
↪	500000e-02	PASS							
↪									

(continues on next page)

(continued from previous page)

1	384	21	10	1.980	1.000000e+00	3.181230e-05	1.000000e-01	7.
↪	625000e-02	PASS						
↪								
1	384	22	10	1.810	1.000000e+00	3.299600e-05	1.000000e-01	8.
↪	000000e-02	PASS						
↪								
1	384	23	10	1.670	1.000000e+00	2.618960e-05	1.000000e-01	9.
↪	125000e-02	PASS						
↪								
1	384	24	10	1.530	1.000000e+00	3.136840e-05	1.000000e-01	7.
↪	875000e-02	PASS						
↪								
1	384	25	10	1.400	1.000000e+00	2.663350e-05	1.000000e-01	8.
↪	375000e-02	PASS						
↪								
1	384	26	10	1.270	1.000000e+00	3.255210e-05	1.000000e-01	8.
↪	625000e-02	PASS						
↪								
1	384	27	10	1.140	1.000000e+00	2.692950e-05	1.000000e-01	7.
↪	500000e-02	PASS						
↪								
1	384	28	10	0.999	1.000000e+00	5.149150e-05	1.000000e-01	9.
↪	250000e-02	PASS						
↪								
1	384	29	10	0.889	1.000000e+00	3.225620e-05	1.000000e-01	8.
↪	750000e-02	PASS						
↪								
1	384	30	10	0.772	1.000000e+00	3.536340e-05	1.000000e-01	9.
↪	375000e-02	PASS						
↪								
1	384	31	10	0.650	1.000000e+00	4.113400e-05	1.000000e-01	9.
↪	125000e-02	PASS						
↪								
1	384	32	10	0.547	1.000000e+00	3.965440e-05	1.000000e-01	8.
↪	750000e-02	PASS						
↪								
1	384	33	10	0.428	1.000000e+00	5.489460e-05	1.000000e-01	9.
↪	625000e-02	PASS						
↪								
1	384	34	10	0.333	1.000000e+00	5.030780e-05	1.000000e-01	8.
↪	875000e-02	PASS						
↪								

(continues on next page)



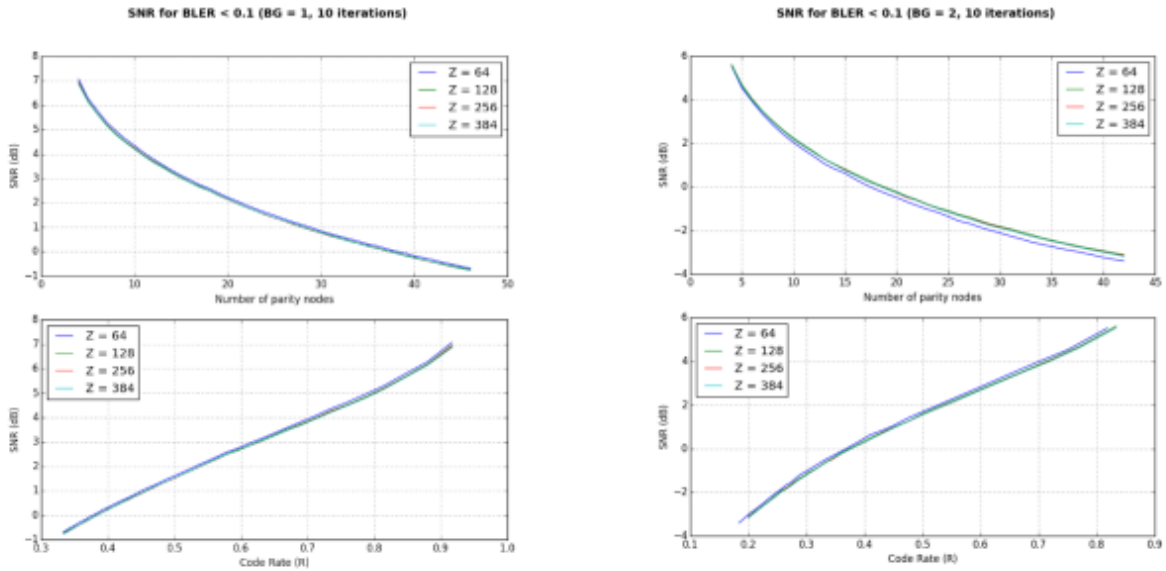
(continued from previous page)

```

| 1 384 35 10 0.220 1.000000e+00 3.906250e-05 1.000000e-01 8.
↪875000e-02 PASS |
|
↪
| 1 384 36 10 0.127 1.000000e+00 2.929690e-05 1.000000e-01 8.
↪250000e-02 PASS |
|
↪
| 1 384 37 10 0.034 1.000000e+00 3.225620e-05 1.000000e-01 9.
↪000000e-02 PASS |
|
↪
| 1 384 38 10 -0.066 1.000000e+00 2.737330e-05 1.000000e-01 8.
↪375000e-02 PASS |
|
↪
| 1 384 39 10 -0.170 1.000000e+00 2.722540e-05 1.000000e-01 8.
↪500000e-02 PASS |
|
↪
| 1 384 40 10 -0.253 1.000000e+00 3.521540e-05 1.000000e-01 7.
↪500000e-02 PASS |
|
↪
| 1 384 41 10 -0.344 1.000000e+00 5.563450e-05 1.000000e-01 9.
↪375000e-02 PASS |
|
↪
| 1 384 42 10 -0.424 1.000000e+00 2.559780e-05 1.000000e-01 8.
↪750000e-02 PASS |
|
↪
| 1 384 43 10 -0.515 1.000000e+00 4.690460e-05 1.000000e-01 9.
↪500000e-02 PASS |
|
↪
| 1 384 44 10 -0.605 1.000000e+00 5.755800e-05 1.000000e-01 9.
↪125000e-02 PASS |
|
↪
| 1 384 45 10 -0.693 1.000000e+00 3.980230e-05 1.000000e-01 8.
↪000000e-02 PASS |
|
↪
| 1 384 46 10 -0.766 1.000000e+00 5.208330e-05 1.000000e-01 9.
↪875000e-02 PASS |
|
↪
| 43 TESTS PASSED, 0 TESTS FAILED
↪
+-----+
↪-----+

```

Plots of current SNR values used for BLER testing are shown below:



### 1.3.5. SCF FAPI Support

#### 1.3.5.1 Overview

Aerial cuBB supports the 5G FAPI 222.10.02 defined by the Small Cell Forum. This release supports most of the control interface (P5) and data path interface (P7) SCF messages.

#### 1.3.5.2 SCF FAPI Messages Supported

The table below summarizes the status of the SCF FAPI messages supported.

SCF Messages	PDU Types	SCF L2 Adapter	SCF Test-MAC	E2E with SCF TestMAC and RU Emulator
DL_TTI.request	PDCCH <sup>[9]</sup>	Y	Y	Y
	PDSCH <sup>[7]</sup>	Y	Y	Y
	CSI-RS <sup>[9]</sup>	Y	Y	Y <sup>[3]</sup>
	SSB <sup>[9]</sup>	Y	Y	Y
UL_TTI.request	PRACH	Y	Y	Y
	PUSCH <sup>[7]</sup>	Y	Y	Y
	PUCCH <sup>[9]</sup>	Y	Y	Y
	SRS <sup>[5][6][10]</sup>	Y	Y	Y
UL_DCI.request	PDCCH:sup` [9] `	Y	Y	Y

continues on next page

Table 25 – continued from previous page

SCF Messages	PDU Types	SCF L2 Adapter	SCF Test-MAC	E2E with SCF TestMAC and RU Emulator
SLOT errors		N	N	N
TX_Data.request <sup>[1]</sup>	PDSCH	Y	Y	Y
Rx_Data.indication <sup>[</sup>	PUSCH (also contains RNTI, HARQ Id, UL_CQI, Timing adv, RSSI)	Y	Y	Y
CRC.indication	CRC	Y	Y	Y
UCI.indiaction	PUSCH <sup>[8]</sup>	Y	Y	Y
	PUCCH format 0,1	Y	Y	Y
	PUCCH format 2,3,4	Y	Y	PF2 and PF3 only
	SR for format 0,1	Y	Y	Y
	SR for format 2,3,4	Y	Y	Y <sup>[4]</sup>
	HARQ for format 0,1	Y	Y	Y
	HARQ for format 2,3,4	Y	Y	PF2 and PF3 only
	CSI part 1	Y	Y	Y
	CSI part 2	Y	Y	PUSCH only
	RSSI and UL SINR metrics	Y	Y	PUSCH, UCI on PUSCH and PF0,1,2,3
SRS.indication <sup>[5][6][</sup>	SRS	Y	Y	Y
RACH.indication	PRACH	Y	Y	Y
Config.request <sup>[2]</sup>		Y	Y	
Config.response		Y	Y	
Start.request		Y	Y	
Stop.request		Y	Y	
Stop.indication		Y	Y	
Error.indication		Y	Y	
Param.request		N	N	
Param.response		N	N	

**Note[1]:** The SCF implementation is based on SCF\_222.10.02, but with the following exceptions:

- ▶ PDU Length of TX\_DATA.request and RX\_DATA.indication are changed to 32-bits. This is defined in SCF\_222.10.03.
- ▶ The implementation supports multiple UE per TTI when the TLV tag is 2 in each PDU. However, the offset value in the TLV is ignored and L1 assumes all TBs in that slot placed in a flat buffer one after the other.

- The `RX_DATA.indication` FAPI message contains the MAC PDU (TB data) in the `data_buf` of the NVIPC message.

Field	Type	Description
TX_DATA.request PDU Length	uint16_t	The total length (in bytes) of the PDU description and PDU data, without the padding bytes. Value: 0 ~ 65535 Change type to uint32_t, value range is: 0 ~ 2^32 -1 <b>[NVIDIA change]</b> : Use it as the PDU data (TB data) size without the PDU description.
RX_DATA.Indication PDU Length	uint16_t	The length of PDU in bytes. A length of 0 indicates a CRC or decoding error. Value: 0 ~ 65535  Change type to uint32_t, value range is: 0 ~ 2^32 -1
RX_DATA.Indication PDU	Variable	The contents of PDU. This will be a MAC PDU. [NVIDIA workaround]: Removed this field, do not parse it in wire-shark dissector. For SCF_222.10.04, although the tag value is set to 1, the MAC PDU is still delivered in a separate NVIPC buffer.
UL_TTI.request SRFlag	uint8_t	Indicates SR. Only valid for format 0 and 1. <b>[NVIDIA workaround]</b> : Enhance to use it as BitLenSr for format 2, 3, 4.

**Note[2]**: Precoding Matrix (Table 3-33) with vendor tag 0xA011 is supported. Digital beam table (Table 3-32) is not supported.

**Note[3]**: For NZP CSI-RS, only 4 antennas and single CSI-RS PDU.

**Note[4]**: The current implementation supports multi-bit SR over PUCCH format 2, 3, and 1. Because SCF FAPI 10.02 doesn't provide any field explicitly suggesting the bit length of the SR in the PUCCH\_PDU of `UL_TTI.request`, use the `SRFlag` field to provide the SR bit length. For example, if the desired SR bit length is 3, set `SRFlag = 3`.

**Note[5]:** SRS.indication and SRS PDU in UL\_TTI.request are supported according to SCF FAPI 222.10.02. SRS can be enabled when flag enable\_srs is set in the cuphycontroller\_xxx.yaml file i.e. enable\_srs: 1.

**Note[6]:** SRS.indication and SRS PDU in UL\_TTI.request are also supported according to SCF FAPI 222.10.04, which needs to be enabled with the “-DSCF\_FAPI\_10\_04=ON” build option and flag enable\_srs is set in the cuphycontroller\_xxx.yaml file i.e. enable\_srs: 1, as described in *Running cuBB End-to-End*.

- ▶ The format of the SRS.indication message is given in SCF FAPI 222.10.04 Table 3-129; the report TLV is defined in Table 3-130.
- ▶ The supported report type is Normalized Channel I/Q Matrix defined in Table 3.132 for codebook or nonCodebook SRS usage.
- ▶ The SRS Report TLV tag is 1 (customized value), the length is the actual report size in bytes without padding, the value field has the offset (in bytes) into the data\_buf portion of NVIPC message for each SRS PDU. The report data is placed in the data\_buf portion of the NVIPC message for all SRS PDUs.
- ▶ In case of wideband SRS, it is possible that the data\_buf portion of NVIPC message carrying SRS.indication does not have enough space to accommodate SRS channel vectors for all the SRS PDUs. In this case, Aerial supports splitting of SRS.indication into multiple message. This feature can be enabled using CONFIG TLV 0x102B / indicationInstancesPerSlot as defined in 5G FAPI 222.10.04 specification table 3-36 for PHY configuration. If this TLV is not enabled by L2 and SRS.indication cannot accommodate all the SRS channel vectors, the SRS.indication will carry partial SRS information. On processing such a SRS PDU, an error indication with error code 0x35 is sent to L2 indicating partial SRS indication.
- ▶ Table 3.131 FAPIv3 Beamforming report, with PRG-level resolution for beamManagement SRS usage is also supported. The SRS Report TLV tag is 2 (customized value), is defined for encoding the SINR reports in the msg\_buf at an offset of 32 bit from the value field, the length is the actual report size in bytes without padding. Also, currently PRG size of 2 is only supported.
- ▶ A combination Usage beamManagement + codebook & beamManagement + non-codebook is also supported.
- ▶ A user defined parameter srsChestBufferIndex is added in FAPI 10.04 version of the PDU's. More details in “**Note 10**”.

**Note[7]:** If flag mMIMO\_enable is set in the cuphycontroller\_xxx.yaml file i.e. mMIMO\_enable: 1 to enable Dynamic Beamforming, indicates that the L2 shall encode the TX Precoding and Beamforming PDU & RX Beamforming PDU to include fields for numPRGs, prgSize and digBFInterface but L2 shall not encode the beamIdx because when Dynamic Beamforming is used, L2 does not have information available for beamIdx but L2 needs to provide the remaining information in the PDU to L1. Dynamic Beamforming: digBFInterfaces = 0 in Tx Precoding and Beamforming PDU & Rx Beamforming PDU and further parameters should not be encoded for i.e. PMidx, beamIdx for Tx Precoding and Beamforming PDU and beamIdx for Rx Beamforming PDU. This is due to the limitation in the size of the NVIPC msg\_buf containing DL\_TTI PDU's.

**Note[8]:** To get HARQ values in UCI.indication for UCI on PUSCH, before complete PUSCH slot processing, L2 should include PHY configurationTLV 0x102B (indicationInstancesPerSlot) with UCI.indication set to 2, according to Table 3-36 in SCF FAPI 222.10.04. If UCI.indication set to 2 in config.request for any cell the early HARQ feature will get activated for all cells.

**Note[9]:** For application of static beam weights that are received in FAPI 10.02 Table 3-61 Digital beam Table (DBT) PDU in Cell\_Config request for the beamIdx's sent in Tx Precoding and Beamforming PDU & Rx Beamforming PDU, we need compile with -DENABLE\_STATIC\_BFW=ON and

enable\_beam\_forming should be set in l2\_adapter\_config\_xxx.yaml. For Static Beamforming: dig-BFInterfaces != 0 in Tx Precoding and Beamforming PDU & Rx Beamforming PDU. Allowed range of BeamId for Static Beamforming weights application is 1 to 1024 rest of the BeamId's are used with Dynamic Beamforming weights.

**Note[10]:** Below are the changes implemented for SRS buffer indexing which is expected to be handled from L2.

- ▶ L1 would pre-allocate a fixed size of 256 buffers to store the SRS Channel Estimates.
- ▶ L2 can configure TLV 0xA019 - NUM\_SRS\_CHEST\_BUFFERS (uint32\_t) for lesser number of buffers than 256, if not sent 256 buffers per cells will be allocated by default.
- ▶ L2 should manage these buffer indexes among all the active UE's.
- ▶ L2 should specify the buffer index in UL\_TTI SRS PDU in the field srsChestBufferIndex.
- ▶ L1 will respond with the same buffer Index (srsChestBufferIndex) in the corresponding SRS.IND.
- ▶ L2 also needs to specify the SRS CH\_EST buffer index in the srsChestBufferIndex field of DLBFW\_CVI.request/ ULBFW\_CVI.request which it wants L1 to use as an input for Dynamic Beamforming weights calculation.
- ▶ If L2 configures same buffer Index (srsChestBufferIndex) for any SRS PDU for which corresponding buffer SRS.INDICATION is not received earlier, the error indication will be reported in that case.
- ▶ If L2 sends SRS buffer Index (srsChestBufferIndex) in DLBFW\_CVI.request/ ULBFW\_CVI.request for which SRS.INDICATION is not yet sent, error will be reported in cuPHY and same will be dropped.

### 1.3.5.3 Vendor Specific Message

A new vendor specific message SLOT . response was added after the 22-4 release. Before the 22-4 release, L2 has to set an event using the nvIPC notify function to inform L1 about "EOM" after sending the last FAPI message. This works well for single cell and when all FAPI messages are on time. L1 also uses the nvIPC notify function to set an event after sending each message.

The new SLOT . response FAPI message is used by L2 as the last FAPI message for each cell in each slot. It has the following advantages:

- ▶ It works as "EOM" for each cell in each slot.
- ▶ Each cell sends a SLOT . response as the last FAPI message of each slot.
- ▶ L2 should send SLOT . response even in empty slots (i.e. slots that have no scheduling).
- ▶ A "Dummy" or empty DL/UL TTI are optional/not-required.
- ▶ The notify event from L2 is optional/not-required.

The SLOT . response message format is shown below:

```

/*****
* Slot . response
*****/

typedef struct
{
    scf_fapi_body_header_t msg_hdr;
    uint16_t sfm;

```

(continues on next page)

(continued from previous page)

```
uint16_t slot;  
} __attribute__((__packed__)) scf_fapi_slot_rsp_t;
```

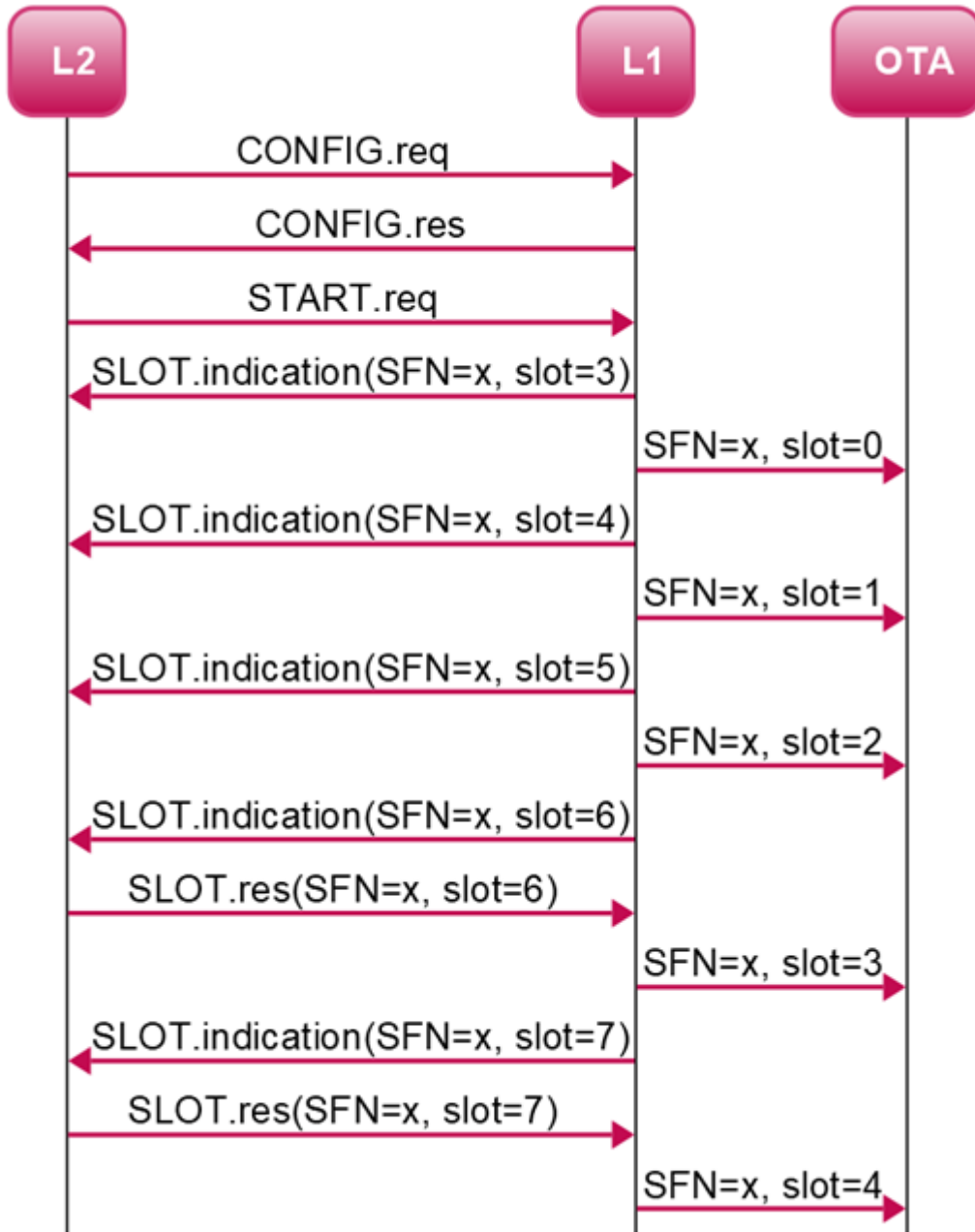
Message-id 0x8F is used **for** this message

```
{ ...  
SCF_FAPI_RX_PRACH_INTEFERNCE_INDICATION = 0x8E,  
  SCF_FAPI_SLOT_RESPONSE                = 0x8F,  
  SCF_FAPI_RESV_2_END                    = 0xFF,  
} scf_fapi_message_id_e;
```

L1 continues to send a notify event after all FAPI messages to L2 to minimize impact  
→ on L2.

### 1.3.5.3.1 Message Sequence

An example message sequence is shown below:



**Note**

On receiving the first SLOT.indication, L2 is unable to send SLOT.response for 2-3 slots because it has a slot advance of 3.



### 1.3.5.3.2 Impact of Late Messages

- ▶ All messages are late for a cell (DL\_TTI+TX\_DATA+UL\_DCI or UL\_TTI)
  - ▶ All messages are dropped for the said cell. No impact on other cells.
- ▶ DL\_TTI arrived on time but TX\_DATA.request is late for a cell
  - ▶ This is considered as a partial slot. Due to cell grouping, PDSCH & DL-PDCCH is dropped for all cells.
- ▶ UL\_TTI is late for a cell
  - ▶ UL-SCH is not processed for the said cell. No impact on other cells.
- ▶ UL\_DCI is late for a cell
  - ▶ UL-PDCCH is not processed for the said cell. No impact on other cells.
- ▶ SLOT.response is late for a cell
  - ▶ All FAPI messages received in time will be processed for the cell.

### 1.3.5.3.3 How to Enable or Disable SLOT.response

This feature is enabled by default in L1 after the 23-1 release. When integrating with L2, L2 is required to send this vendor-specific message in the manner described above.

Option ENABLE\_L2\_SLT\_RSP should be configured with the same value in L1, L2 and libnvipc.so standalone build for L2. Refer to cuBB Quickstart Guide for details.

If L2 doesn't support the SLOT.response message, disable this feature by setting the "-ENABLE\_L2\_SLT\_RSP=OFF" flag in the cmake command:

```
cmake <existing flags> -DENABLE_L2_SLT_RSP=OFF
```

Once the feature is enabled, the following is true:

- ▶ L2 has to send a vendor-specific SLOT.response message as the last FAPI message for each cell.
  - ▶ L2 to send this message even in empty slot (where nothing is scheduled).
- ▶ allowed\_fapi\_latency is deprecated and presumed to be 0.
  - ▶ L2 to complete sending all FAPI messages within the 500 us time-budget marked by SLOT.indication from L1.
  - ▶ Late FAPI messages will be dropped.
- ▶ A "Dummy" DL/UL TTI messages in empty slots is optional.
- ▶ A notify event after sending all FAPI messages is optional.
  - ▶ ipc\_sync\_mode in the L2 Adapter config file is deprecated.
- ▶ L1 will continue to send a Notify event after all FAPI messages to minimize impact on L2.

### 1.3.5.4 Dynamic Beamforming for 64T64R

- ▶ To enable this feature in Aerial software, flag `mMIMO_enable` should be set/introduced in the `cuphycontroller_xxx.yaml` file i.e. `mMIMO_enable: 1`.
- ▶ Two additional TLVs are required in `CONFIG.req`:
  - ▶ **TLV 0xA016 denoting NUM\_TX\_PORT (uint8\_t)**
    - ▶ This field specifies the number of DL BB ports for PHY. 5G FAPI 222.10.04 described the field `numTxAnt` and `numRxAnt` in Table 3-37 as - 'numTxAnt cannot exceed the number of DL BB ports for the PHY'. Hence the fields in table 3-37 represent the logical antenna ports.
    - ▶ 5G FAPI 223 describes baseband ports as a mapping between layers to RU TX/RX ports. PHY needs to know the BB ports from L2 (see Fig 3-3 in SCF-223.2.0.4).
    - ▶ This field will be used by PHY to read the number of DL BB ports.
    - ▶ If the TLV is not received from L2 and flag `mMIMO_enable` is set in the `cuphycontroller_xxx.yaml` file i.e. `mMIMO_enable: 1`, the default value for number of DL BB ports is set to 8.
  - ▶ **TLV 0xA017 denoting NUM\_RX\_PORT (uint8\_t)**
    - ▶ This field specifies the number of UL BB ports for PHY. 5G FAPI 222.10.04 described the field `numTxAnt` and `numRxAnt` in Table 3-37 as - 'numRxAnt cannot exceed the number of UL BB ports for the PHY'. Hence the fields in table 3-37 represent the logical antenna ports.
    - ▶ 5G FAPI 223 describes baseband ports as a mapping between layers to RU TX/RX ports. PHY needs to know the BB ports from L2 (see Fig 3-3 in SCF-223.2.0.4).
    - ▶ This field will be used by PHY to read the number of UL BB ports
    - ▶ If the TLV is not received from L2 and flag `mMIMO_enable` is set in the `cuphycontroller_xxx.yaml` file i.e. `mMIMO_enable: 1`, the default value for number of UL BB ports is set to 4.
- ▶ DL & UL TTI have an additional field added for TRP scheme. See Note-6 in SCF FAPI Messages supported section
- ▶ Dynamic Beamforming is supported for PDSCH and PUSCH only
- ▶ A UE that is scheduled for SRS on S-slot should not be scheduled for dynamic beamforming of PDSCH and PUSCH in subsequent D & U slots until SRS indication for the UE is received. This prevents a race condition between L1 and L2 where the SRS channel vectors have been updated in the GPU hosted memory, but the latest SRS channel vectors are yet to be sent to L2. In this case, L2 might make a scheduling decision based on stale SRS channel vectors and the BFW calculation might happen with refreshed SRS channel vectors.

Two new FAPI messages have been defined from L2 to L1 to implement beamforming weight calculation in L1 as follows:

- ▶ `SCF_FAPI_DL_BFW_CVI_REQUEST = 0x90`
- ▶ `SCF_FAPI_UL_BFW_CVI_REQUEST = 0x91`

Structure of the FAPI message from L2 to L1 for beamforming weight calculation are as below. The same message structure is used for DL(PDSCH) and UL(PUSCH). When used for DL(PDSCH), it is referred to as `DLBFW_CVI.request` and when used for UL(PUSCH), it is referred to as `ULBFW_CVI.request`.

Table 3-1001 DLBFW\_CVI.request message body

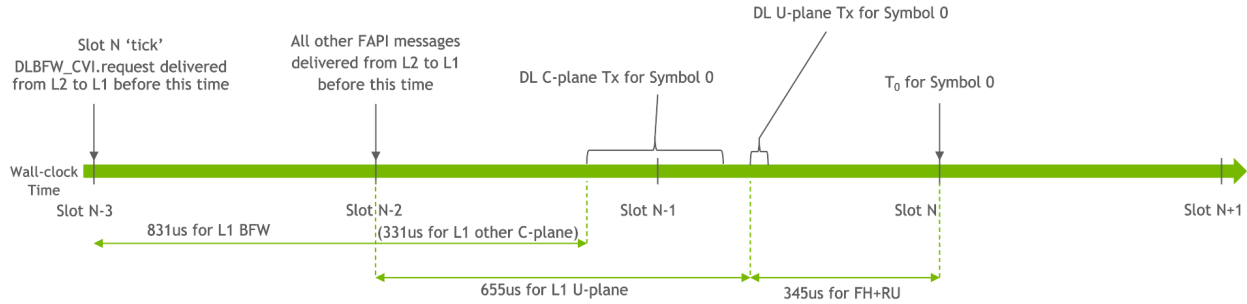
Field	Type	Description
SFN	uint16_t	SFN Value: 0 -> 1023
Slot	uint16_t	Slot Value: 0 ->159
nPDUs	uint8_t	Number of PDUs that are included in this message. All PDUs in the message are numbered in order. Each PDU is corresponding to a UE Group. Value: 0 -> 255
For Number of PDUs {		
PDUSize	uint16_t	Size of the PDU control information (in bytes). This length value includes the 4 bytes required for the PDU type and PDU size parameters. Value 0 -> 65535
DLBFW CVI Configuration	structure	See Table 3-1002 DLBFW CVI PDU
}		

Table 3-1002 DLBFW CVI PDU

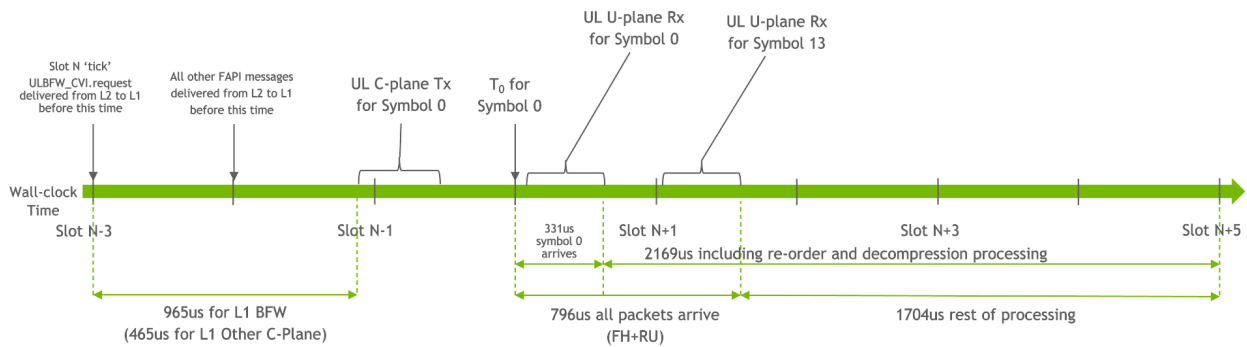
Field	Type	Description
rbStart	uint16_t	For resource allocation type 1. [TS38.214, sec 5.1.2.2.2] The starting resource block within the BWP for this PDSCH. Value: 0->274
rbSize	uint16_t	For resource allocation type 1. [TS38.214, sec 5.1.2.2.2] The number of resource block within for this PDSCH. Value: 1->275
numPRGs	uint16_t	Number of PRGs spanning this allocation. Value : 1->275
prgSize	uint16_t	Size in RBs of a precoding resource block group (PRG) – to which same precoding and digital beamforming gets applied. Value: 1->275
nUe	uint8_t	Number of UE in this group Value 1 -> 16
For Number of UEs {		
RNTI	uint16_t	The RNTI used for identifying the UE when receiving the PDU Value: 1 -> 65535.
srsChestBufferIndex	uint32_t	Buffer index of SRS Channel estimates stored in device memory. Value: 0 ->255.
pduIndex	uint16_t	PDU index associated to pduIndex in PDSCH PDU of DL_TTI.request Value: 0 -> 65535
gnbAntIdxStart	uint8_t	Corresponding to "gI" indicated by "Array representing channel matrix H" in SRS.indication. L1 can regard antenna indices from "gnbAntIdxStart" to "gnbAntIdxEnd" are used. Value: 0
gnbAntIdxEnd	uint8_t	Corresponding to "gI" indicated by "Array representing channel matrix H" in SRS.indication. L1 can regard antenna indices from "gnbAntIdxStart" to "gnbAntIdxEnd" are used. Value: 63
numOfUeAnt	uint8_t	Value: 1->4
For number of UE Ants {		
ueAntIdx	uint8_t	Corresponding to "uI" indicated by "Array representing channel matrix H" in SRS.indication. Value: 0,1,2,3
}		
}		

Timeline for receiving DLBFW\_CVI.request and ULBFW\_CVI.request is as shown below:

Downlink timeline for slot N



### Uplink timeline for slot N - PUSCH/PUCCH/PRACH



#### 1.3.5.5 Static Beamforming for 64T64R

- ▶ One additional TLV is required in CONFIG.req:
  - ▶ TLV 0xA010 DIGITAL\_BEAM\_TABLE\_PDU (uint8\_t)

This TLV is used in Cell\_Config request when DBT PDU needs to be encoded, which contains the static beamforming weights that will be used for certain channels.
- ▶ The “fixed RTW” and their corresponding weights come from FAPI 10.02 Table 3-61 Digital beam Table (DBT) PDU.
- ▶ A new bigger NVIPC buffer pool (cpu\_large: {buf\_size: 4096000, pool\_len: 64}) is defined to store the entire DBT PDU in a single buffer. The same should be used for encoding and sending FAPI 10.02 Table 3-61 Digital beam Table (DBT) PDU.
- ▶ DBT PDU can only be processed or stored when enable\_beam\_forming is set in l2\_adapter\_config\_xxx.yaml.

#### 1.3.5.6 Additional Aerial Specific Error Codes Reported in ERROR.indication from L1 to L2

Additional Aerial specific error codes have been added, starting from value 0x33, and L2 may receive these error codes in ERROR.indication message from L1 to L2. The following is an example:

```
SCF_ERROR_CODE_FAPI_END = 0x32,

//Vendor specific error codes ---- begin

SCF_ERROR_CODE_L1_PROC_OBJ_UNAVAILABLE_ERR = 0x33,
```

(continues on next page)

(continued from previous page)

```

SCF_ERROR_CODE_MSG_LATE_SLOT_ERR          = 0x34, //Indicates that L1's timer thread
↳did not wake up on the slot boundary and slot indication for the indicated SFN,slot
↳is late and will not be sent from L1 to L2

SCF_ERROR_CODE_PARTIAL_SRS_IND_ERR        = 0x35, //Indicates partial SRS indication

SCF_ERROR_CODE_L1_DL_CPLANE_TX_ERROR     = 0x36, //Indicates a DL C-plane trasmission
↳error (Timing/Functional)

SCF_ERROR_CODE_L1_UL_CPLANE_TX_ERROR     = 0x37, //Indicates a UL C-plane trasmission
↳error (Timing/Functional)

SCF_ERROR_CODE_L1_DL_GPU_ERROR           = 0x38, //Indicates a DL GPU pipeline processing
↳error

SCF_ERROR_CODE_L1_DL_CPU_TASK_ERROR      = 0x39, //Indicates a DL CPU Task incompletion
↳error

SCF_ERROR_CODE_L1_UL_CPU_TASK_ERROR      = 0x3A, //Indicates a UL CPU Task incompletion
↳error

SCF_ERROR_CODE_L1_P1_EXIT_ERROR          = 0x3B, //Indicates Part 1 of the error
↳indication during L1 app exit process

SCF_ERROR_CODE_L1_P2_EXIT_ERROR          = 0x3C, //Indicates Part 2 of the error
↳indication during L1 app exit process post cudaDeviceSynchronize if CUDA coredump
↳env variables are set

SCF_ERROR_CODE_L1_DL_CH_ERROR            = 0x3D, //Indicates DL channel run (CPU/GPU)
↳error

SCF_ERROR_CODE_L1_UL_CH_ERROR            = 0x3E //Indicates UL channel run (CPU/GPU)
↳error

```

## 1.3.6. Limitations

### 1.3.6.1 Known Limitations

- ▶ The cuPHY library and binaries are intended for the Linux environment on the qualified platforms only.
- ▶ The supported configurations are limited to those listed above. Other configurations are not supported and may not perform well.
- ▶ Only homogeneous configurations supported for multiple cells.
- ▶ The configurable YAML parameters `enable_h2d_copy_thread`, `h2d_copy_thread_cpu_affinity`, and `h2d_copy_thread_sched_priority` are optional in the `cuphycontroller` YAML file. If these parameters are not present, the code uses the default values and throws the exception "YAML invalid key:" on the `cuphycontroller` console. This exception message has no impact on the functionality and can be disregarded.
- ▶ GPU Initiated Comms for DL (`gpu_init_comms_dl` flag in the `cuphycontroller` config yam) is required to be enabled by default from 22-2.4 release onwards. The flag enables the feature within Aerial L1 to engage GPU kernels to prepare and send U-Plane packets on the DL as opposed to

CPU Initiated Comms (gpu\_init\_comms\_dl=0) which exercises CPU code/consumes CPU cycles to prepare/send U-plane packets on the DL.

- ▶ No simultaneous DL and UL scheduling in S-slot. However, DL-only s-slot is supported in E2E test with O-RU.
- ▶ When the FAPI messages for a given cell are sent via nvipc, L1 expects an explicit notify (once per cell) via nvipc. In the case of multiple cells, multiple explicit notify APIs be called from L2. When a cell doesn't have any messages for a given slot, L1 expects dummy DL\_TTI and/or UL\_TTI.request, that is (nPDU = 0), to be sent "per cell". If the Slot Response feature is enabled by compiling Aerial with -DENABLE\_L2\_SLT\_RSP=ON, this step is optional.
- ▶ For multi cells operation, L2 can signal the L2Adapter in 2 ways:
  - ▶ Single event per slot: which contains SCF FAPI messages for all cells. The single event is raised by calling nvipc notify(1) once per slot after the messages for all the cells are sent.
  - ▶ Single event per cell: which is signaled by L2 after all FAPI messages for a given cell are sent. It is expected that multiple nvipc notify(1) are called for multiple cells. The number of times that notify is being called must be the same as the number of active cells. A cell is marked active after START.req is received from L2. In this case, L1 expects dummy DL\_TTI and UL\_TTI described above. This is the default behavior.

To select the operation mode, set the ipc\_sync\_mode in yaml:

```
# Option 1: Sync per slot
ipc_sync_mode: 0
# Option 2: Sync per active cell
ipc_sync_mode: 1
```

If Slot Response feature is enabled by compiling Aerial with -DENABLE\_L2\_SLT\_RSP=ON, this setting is a no-op as L1 does not expect any event from L2.

- ▶ Cell life cycle management:
  - ▶ All cells have to be configured before any cell start.
  - ▶ No In-service configuration update.
  - ▶ CONFIG.request received in CONFIGURED (Out-of-Service) state can be used to change PCI and the supported PRACH parameters specified in dynamic PRACH section in cuBB quickstart guide only. PHY ignores any other TLVs received in CONFIG.request. If CONFIG.response indicates success, then only PCI and supported PRACH parameters are changed. All other parameters remain as in the initial CONFIG.request received for the cell.
  - ▶ PHY reconfiguration of a cell in CONFIGURED (Out-of-Service) state can take upto 40ms to complete (details below). Another CONFIG.request for any cell during this time (around 20ms) that occurs before receiving a CONFIG.response returns a CONFIG.response with the error code "MSG\_INVALID\_STATE". The ERROR.indication will NOT be sent for this error. L2 needs to wait to receive a CONFIG.response before sending a CONFIG.request for another cell in CONFIGURED state.
    - ▶ If Aerial is configured for 4 cells and 3 cells are In-service with data running, reconfiguration of 1 cell (Out-of-Service) can take around 40ms to complete
    - ▶ If Aerial is configured for 4 cells and 3 cells are In-service with no data running, reconfiguration of 1 cell (O-RU) can take around 20 ms to complete
  - ▶ If CONFIG.response is received with error code "MSG\_INVALID\_CONFIG", then reconfiguration was unsuccessful and the cell is still with the configuration received in initial CONFIG.request.

- ▶ No UE attach allowed in all cells during the reconfiguration time.
- ▶ Dynamic M-plane parameters:
  - ▶ When OAM sends gRPC message to change MAC address in M-plane, it must be a valid O-RU MAC address.
- ▶ The `nvlog_observer` and `nvlog_collect` are deprecated in 23-1.
- ▶ F13 test cases are deprecated in 23-2.
- ▶ Early HARQ in UCI.indication:
  - ▶ This feature is supported only for the first UL slot (x4 slots) and when all the early-HARQ bits are resident in symbols 0-3.
  - ▶ UCI.Indication with early HARQ will not have any measurement values.
  - ▶ If only HARQ is scheduled on PUSCH then with this feature enabled, no UCI.indication will be sent to L2 after full slot processing of PUSCH. Consequently no measurements for that slot will be reported to L2.
  - ▶ If CSI reports are also scheduled on PUSCH along with HARQ, then UCI.Indication with early HARQ will not have any measurement values. But the UCI.indication sent after full slot processing of PUSCH will have the measurements.
  - ▶ A constraint to enable early-HARQ is that these HARQ bits should be fully resident in OFDM symbols 0-3. So HARQ bits resident in OFDM symbols 0-3 will be in the 1st UCI.indication (that is, early-HARQ indication) and all other HARQ bits in the subsequent UCI.indication (that is, after full slot PUSCH processing completes).
- ▶ Multiple cell operation without issuing dummy config.req:
  - ▶ L2 should wait for at least 40msec between two CONFIG.request even at the initial stage, so that CONFIG.response is received by L2.
  - ▶ L2 can retry the failed CONFIG.request for a given cell after 1 sec.
- ▶ Multi-L2 with single cuphycontroller per GPU:
  - ▶ The total cell number of all L2 instances cannot exceed the `cell_group_num` configured in `cuphycontroller.yaml`.
  - ▶ nvIPC only supports static cell allocation defined in the `nvipc_multi_instances.yaml` for multiple L2 instances. The number of cells and the cell mapping in each L2 instance cannot change after L1 is configured..
  - ▶ Support dynamic cell start/stop in each L2 instance. Do not support dynamic L2 restart. L2 instance needs to hold the `nvipc` instance after connecting to L1.
- ▶ 64T64R TDD single cell:
  - ▶ PDSCH Resource Allocation Type 0 (RAT0) is not supported.
  - ▶ SRS reports related to antennaSwitching (FAPI 222.10.04, Table 3-133 - Channel SVD Representation) is not supported.



### 1.3.6.2 Known Issues

- ▶ cuBB test case 7600 reports CRC error when debug synch check is enabled. Issue only appears when certain versions of synch debug tool are enabled.
- ▶ The support for CPU Initiated Comms (gpu\_init\_comms\_dl=0) mode is no longer available after the 22-2.4 release and it is recommended that this mode not be enabled for testing purposes.
- ▶ Support up to 8 DMRS ports, if the allocations are contiguous in PUSCH.
- ▶ SCHED\_FIFO + 100% CPU poll thread causes the system to hang on the 5.4.0-65-lowlatency kernel. The solution is one of the following:
  - ▶ Configure the kernel option CONFIG\_RCU\_NOCB\_CPU=y, recompile, and install the kernel.
  - ▶ Upgrade the host system to 5.15.0-71-lowlatency or later.
- ▶ CUDA application on Grace Hopper:
  - ▶ CUDA applications on the Grace Hopper platform require ATS support. Currently, ATS is not enabled on the arm64 platform when IOMMU passthrough is enabled.
- ▶ NIC string conversion issue on Grace Hopper:
  - ▶ While working on dynamic CPU core assignments in K8s pod, we need to parse and dump the cuphycontroller config yaml file. On the Grace Hopper, the `nic: 0000:01:00.0` will be converted to `nic: 60.0`. This is because the PCIe address might be interpreted as a 60 based integer according to '<https://yaml.org/type/int.html>'. The fix is to explicitly tell yaml parser to interpret the PCIe address as a string by putting single quotation marks around or `!!str` before the pcie address, e.g., `nic: '0000:01:00.0'` or `nic: !!str 0000:01:00.0`.

```
From
  sed -i "s/nic:./nic: 0000:01:00.0/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/
↪config/cuphycontroller_P5G_FXN.yaml
to
  sed -i "s/nic:./nic: '0000:01:00.0'/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/
↪config/cuphycontroller_P5G_FXN.yaml
  (or sed -i "s/nic:./nic: !!str 0000:01:00.0/" ${cuBB_SDK}/cuPHY-CP/
↪cuphycontroller/config/cuphycontroller_P5G_FXN.yaml)
```

- ▶ There is a known issue (DL C-plane send error) when running 8C Peak (59/59b/59c) test case with multiple L2.
- ▶ The following test cases are not passing. They could be functionality issues or test framework issues:

Channel	Test Cases	Feature
PDSCH	3881	64TR
PUSCH	7600	Multiple CSIP2

## 1.3.7. Acknowledgements

### 1.3.7.1 Abseil

Apache License  
Version 2.0, January 2004  
<https://www.apache.org/licenses/>

#### TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

##### 1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of

(continues on next page)

(continued from previous page)

the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. **Grant of Copyright License.** Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.
3. **Grant of Patent License.** Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.
4. **Redistribution.** You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:
  - (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and
  - (b) You must cause any modified files to carry prominent notices stating that You changed the files; and
  - (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
  - (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained

(continues on next page)

(continued from previous page)

within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. **Submission of Contributions.** Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.
6. **Trademarks.** This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.
7. **Disclaimer of Warranty.** Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.
8. **Limitation of Liability.** In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

(continues on next page)

(continued from previous page)

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[ ]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");  
you may not use this file except in compliance with the License.  
You may obtain a copy of the License at

<https://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

### 1.3.7.2 Backward-cpp

Copyright 2013 Google Inc. All Rights Reserved.

The MIT License (MIT)

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

(continues on next page)

(continued from previous page)

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

### 1.3.7.3 BoringSSL

BoringSSL is a fork of OpenSSL. As such, large parts of it fall under OpenSSL licensing. Files that are completely new have a Google copyright and an ISC license. This license is reproduced at the bottom of this file.

Contributors to BoringSSL are required to follow the CLA rules for Chromium: <https://cla.developers.google.com/clas>

Files in `third_party/` have their own licenses, as described therein. The MIT license, for `third_party/ fiat`, which, unlike other `third_party` directories, is compiled into non-test libraries, is included below.

The OpenSSL toolkit stays under a dual license, i.e. both the conditions of the OpenSSL License and the original SSLeay license apply to the toolkit. See below for the actual license texts. Actually both licenses are BSD-style Open Source licenses. In case of any license issues related to OpenSSL please contact `openssl-core@openssl.org`.

The following are Google-internal bug numbers where explicit permission from some authors is recorded for use of their work. (This is purely for our own record keeping.)

27287199  
27287880  
27287883

OpenSSL License  
-----

```
/* =====  
* Copyright (c) 1998-2011 The OpenSSL Project. All rights reserved.  
*  
* Redistribution and use in source and binary forms, with or without  
* modification, are permitted provided that the following conditions  
* are met:  
*  
* 1. Redistributions of source code must retain the above copyright  
* notice, this list of conditions and the following disclaimer.  
*  
* 2. Redistributions in binary form must reproduce the above copyright  
* notice, this list of conditions and the following disclaimer in  
* the documentation and/or other materials provided with the  
* distribution.  
*  
* 3. All advertising materials mentioning features or use of this  
* software must display the following acknowledgment:
```

(continues on next page)

(continued from previous page)

```

*   "This product includes software developed by the OpenSSL Project
*   for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
*
* 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
*   endorse or promote products derived from this software without
*   prior written permission. For written permission, please contact
*   openssl-core@openssl.org.
*
* 5. Products derived from this software may not be called "OpenSSL"
*   nor may "OpenSSL" appear in their names without prior written
*   permission of the OpenSSL Project.
*
* 6. Redistributions of any form whatsoever must retain the following
*   acknowledgment:
*   "This product includes software developed by the OpenSSL Project
*   for use in the OpenSSL Toolkit (http://www.openssl.org/)"
*
* THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
* EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
* PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR
* ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
* NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
* LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
* STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
* =====
*
* This product includes cryptographic software written by Eric Young
* (eay@cryptsoft.com). This product includes software written by Tim
* Hudson (tjh@cryptsoft.com).
*
*/

```

Original SSLeay License

-----

```

/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
* All rights reserved.
*
* This package is an SSL implementation written
* by Eric Young (eay@cryptsoft.com).
* The implementation was written so as to conform with Netscapes SSL.
*
* This library is free for commercial and non-commercial use as long as
* the following conditions are aheared to. The following conditions
* apply to all code found in this distribution, be it the RC4, RSA,
* lhash, DES, etc., code; not just the SSL code. The SSL documentation
* included with this distribution is covered by the same copyright terms
* except that the holder is Tim Hudson (tjh@cryptsoft.com).
*
* Copyright remains Eric Young's, and as such any Copyright notices in
* the code are not to be removed.

```

(continues on next page)

(continued from previous page)

```

* If this package is used in a product, Eric Young should be given attribution
* as the author of the parts of the library used.
* This can be in the form of a textual message at program startup or
* in documentation (online or textual) provided with the package.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
* 1. Redistributions of source code must retain the copyright
* notice, this list of conditions and the following disclaimer.
* 2. Redistributions in binary form must reproduce the above copyright
* notice, this list of conditions and the following disclaimer in the
* documentation and/or other materials provided with the distribution.
* 3. All advertising materials mentioning features or use of this software
* must display the following acknowledgement:
* "This product includes cryptographic software written by
* Eric Young (eay@cryptsoft.com)"
* The word 'cryptographic' can be left out if the routines from the library
* being used are not cryptographic related :-).
* 4. If you include any Windows specific code (or a derivative thereof) from
* the apps directory (application code) you must include an acknowledgement:
* "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
*
* THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
* ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
* FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
* DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
* OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
* LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
* OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
* SUCH DAMAGE.
*
* The licence and distribution terms for any publically available version or
* derivative of this code cannot be changed. i.e. this code cannot simply be
* copied and put under another distribution licence
* [including the GNU Public Licence.]
*/

```

ISC license used for completely new code in BoringSSL:

```

/* Copyright (c) 2015, Google Inc.
*
* Permission to use, copy, modify, and/or distribute this software for any
* purpose with or without fee is hereby granted, provided that the above
* copyright notice and this permission notice appear in all copies.
*
* THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
* WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
* MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY
* SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
* WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION
* OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN

```

(continues on next page)



(continued from previous page)

```
* CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE. */
```

The code in `third_party/ fiat`  carries the MIT license:

Copyright (c) 2015-2016 the fiat-crypto authors (see <https://github.com/mit-plv/fiat-crypto/blob/master/AUTHORS>).

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Licenses for support code

-----

Parts of the TLS test suite are under the Go license. This code is not included in BoringSSL (i.e. `libcrypto` and `libssl`) when compiled, however, so distributing code linked against BoringSSL does not trigger this license:

Copyright (c) 2009 The Go Authors. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- \* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- \* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

- \* Neither the name of Google Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT

(continues on next page)

(continued from previous page)

LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

BoringSSL uses the Chromium test infrastructure to run a continuous build, trybots etc. The scripts which manage this, and the script for generating build metadata, are under the Chromium license. Distributing code linked against BoringSSL does not trigger this license.

Copyright 2015 The Chromium Authors. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- \* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- \* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- \* Neither the name of Google Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

#### 1.3.7.4 Benchmark

Apache License  
Version 2.0, January 2004  
<http://www.apache.org/licenses/>

#### TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

##### 1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by

(continues on next page)

(continued from previous page)

the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable

(continues on next page)

(continued from previous page)

copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.
4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:
  - (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and
  - (b) You must cause any modified files to carry prominent notices stating that You changed the files; and
  - (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
  - (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions

(continues on next page)

(continued from previous page)

for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. **Submission of Contributions.** Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.
6. **Trademarks.** This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.
7. **Disclaimer of Warranty.** Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.
8. **Limitation of Liability.** In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.
9. **Accepting Warranty or Additional Liability.** While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following

(continues on next page)

(continued from previous page)

boilerplate notice, with the fields enclosed by brackets "[ ]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");  
you may not use this file except in compliance with the License.  
You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

### 1.3.7.5 Bloaty

Apache License  
Version 2.0, January 2004  
<http://www.apache.org/licenses/>

#### TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

##### 1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical

(continues on next page)

(continued from previous page)

transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.
3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate

(continues on next page)

(continued from previous page)

as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:
  - (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and
  - (b) You must cause any modified files to carry prominent notices stating that You changed the files; and
  - (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
  - (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.
6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

(continues on next page)



(continued from previous page)

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.
8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.
9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[ ]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");  
you may not use this file except in compliance with the License.  
You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS,

(continues on next page)

(continued from previous page)

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
See the License for the specific language governing permissions and  
limitations under the License.

### 1.3.7.6 c-ares

# c-ares license

Copyright (c) 2007 - 2018, Daniel Stenberg with many contributors, see AUTHORS  
file.

Copyright 1998 by the Massachusetts Institute of Technology.

Permission to use, copy, modify, and distribute this software and its  
documentation for any purpose and without fee is hereby granted, provided that  
the above copyright notice appear in all copies and that both that copyright  
notice and this permission notice appear in supporting documentation, and that  
the name of M.I.T. not be used in advertising or publicity pertaining to  
distribution of the software without specific, written prior permission.  
M.I.T. makes no representations about the suitability of this software for any  
purpose. It is provided "as is" without express or implied warranty.

### 1.3.7.7 CivetWeb

Copyright (c) 2013-2021 The CivetWeb developers (CREDITS.md)

Copyright (c) 2004-2013 Sergey Lyubka

Copyright (c) 2013 No Face Press, LLC (Thomas Davis)

Copyright (c) 2013 F-Secure Corporation

Permission is hereby granted, free of charge, to any person obtaining a copy of this  
↪ software and associated documentation files (the "Software"), to deal in the  
↪ Software without restriction, including without limitation the rights to use, copy,  
↪ modify, merge, publish, distribute, sublicense, and/or sell copies of the Software,  
↪ and to permit persons to whom the Software is furnished to do so, subject to the  
↪ following conditions:

The above copyright notice and this permission notice shall be included in all copies  
↪ or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,  
↪ INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A  
↪ PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT  
↪ HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION  
↪ OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE  
↪ SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

### 1.3.7.8 Data plane API

Apache License  
Version 2.0, January 2004  
<http://www.apache.org/licenses/>

#### TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

##### 1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to

(continues on next page)

(continued from previous page)

communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.
3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.
4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:
  - (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and
  - (b) You must cause any modified files to carry prominent notices stating that You changed the files; and
  - (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
  - (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed

(continues on next page)

(continued from previous page)

as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. **Submission of Contributions.** Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.
6. **Trademarks.** This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.
7. **Disclaimer of Warranty.** Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.
8. **Limitation of Liability.** In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.
9. **Accepting Warranty or Additional Liability.** While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity,

(continues on next page)

(continued from previous page)

or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[ ]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner].

Licensed under the Apache License, Version 2.0 (the "License");  
you may not use this file except in compliance with the License.  
You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

### 1.3.7.9 DPDK

The DPDK uses the Open Source BSD-3-Clause license for the core libraries and drivers. The kernel components are naturally GPL-2.0 licensed.

Including big blocks of License headers in all files blows up the source code with mostly redundant information. An additional problem is that even the same licenses are referred to by a number of slightly varying text blocks (full, abbreviated, different indentation, line wrapping and/or white space, with obsolete address information, ...) which makes validation and automatic processing a nightmare.

To make this easier, DPDK uses a single line reference to Unique License Identifiers in source files as defined by the Linux Foundation's SPDX project (<https://spdx.org/>).

Adding license information in this fashion, rather than adding full license text, can be more efficient for developers; decreases errors; and improves automated detection of licenses. The current set of valid, predefined SPDX

(continues on next page)

(continued from previous page)

identifiers is set forth on the SPDX License List at <https://spdx.org/licenses/>.

DPDK uses first line of the file to be SPDX tag. In case of `*#!*` scripts, SPDX tag can be placed in 2nd line of the file.

For example, to label a file as subject to the BSD-3-Clause license, the following text would be used:

```
SPDX-License-Identifier: BSD-3-Clause
```

To label a file as GPL-2.0 (e.g., for code that runs in the kernel), the following text would be used:

```
SPDX-License-Identifier: GPL-2.0
```

To label a file as dual-licensed with BSD-3-Clause and GPL-2.0 (e.g., for code that is shared between the kernel and userspace), the following text would be used:

```
SPDX-License-Identifier: (BSD-3-Clause OR GPL-2.0)
```

To label a file as dual-licensed with BSD-3-Clause and LGPL-2.1 (e.g., for code that is shared between the kernel and userspace), the following text would be used:

```
SPDX-License-Identifier: (BSD-3-Clause OR LGPL-2.1)
```

Any new file contributions in DPDK shall adhere to the above scheme.

It is also being recommended to replace the existing license text in the code with SPDX-License-Identifiers.

Any exception to the DPDK IP policies shall be approved by DPDK Tech Board and DPDK Governing Board. Steps for any exception approval:

1. Mention the appropriate license identifier form SPDX. If the license is not listed in SPDX Licenses. It is the submitters responsibility to get it first listed.
2. Get the required approval from the DPDK Technical Board. Technical Board may advise the author to check alternate means first. If no other alternative are found and the merit of the contributions are important for DPDK's mission, it may decide on such exception with two-thirds vote of the members.
3. Technical Board then approach Governing Board for such limited approval for the given contribution only.

Any approvals shall be documented in "Licenses/exceptions.txt" with record dates.

DPDK project supported licenses are:

1. BSD 3-clause "New" or "Revised" License  
 SPDX-License-Identifier: BSD-3-Clause  
 URL: <http://spdx.org/licenses/BSD-3-Clause#licenseText>  
 DPDK License text: licenses/bsd-3-clause.txt
2. GNU General Public License v2.0 only  
 SPDX-License-Identifier: GPL-2.0  
 URL: <http://spdx.org/licenses/GPL-2.0.html#licenseText>  
 DPDK License text: licenses/gpl-2.0.txt

(continues on next page)

(continued from previous page)

3. GNU Lesser General Public License v2.1  
SPDX-License-Identifier: LGPL-2.1  
URL: <http://spdx.org/licenses/LGPL-2.1.html#licenseText>  
DPDK License text: licenses/lgpl-2.1.txt

### 1.3.7.10 Eigen

#### Mozilla Public License Version 2.0

##### 1. Definitions

###### 1.1. "Contributor"

means each individual or legal entity that creates, contributes to the creation of,  
↪ or owns Covered Software.

###### 1.2. "Contributor Version"

means the combination of the Contributions of others (if any) used by a  
↪ Contributor and that particular Contributor's Contribution.

###### 1.3. "Contribution"

means Covered Software of a particular Contributor.

###### 1.4. "Covered Software"

means Source Code Form to which the initial Contributor has attached the notice in  
↪ Exhibit A, the Executable Form of such Source Code Form, and Modifications of  
↪ such Source Code Form, in each case including portions thereof.

###### 1.5. "Incompatible With Secondary Licenses"

means

a. that the initial Contributor has attached the notice described in Exhibit B  
↪ to the Covered Software; or

b. that the Covered Software was made available under the terms of version 1.1  
↪ or earlier of the License, but not also under the terms of a Secondary License.

###### 1.6. "Executable Form"

means any form of the work other than Source Code Form.

###### 1.7. "Larger Work"

means a work that combines Covered Software with other material, in a separate  
↪ file or files, that is not Covered Software.

###### 1.8. "License"

means this document.

###### 1.9. "Licensable"

means having the right to grant, to the maximum extent possible, whether at the  
↪ time of the initial grant or subsequently, any and all of the rights conveyed by  
↪ this License.

###### 1.10. "Modifications"

means any of the following:

a. any file in Source Code Form that results from an addition to, deletion from,

(continues on next page)



(continued from previous page)

↪ or modification of the contents of Covered Software; or

b. any new file in Source Code Form that contains any Covered Software.

#### 1.11. "Patent Claims" of a Contributor

means any patent claim(s), including without limitation, method, process, and  
↪ apparatus claims, in any patent Licensable by such Contributor that would be  
↪ infringed, but for the grant of the License, by the making, using, selling,  
↪ offering for sale, having made, import, or transfer of either its Contributions or  
↪ its Contributor Version.

#### 1.12. "Secondary License"

means either the GNU General Public License, Version 2.0, the GNU Lesser General  
↪ Public License, Version 2.1, the GNU Affero General Public License, Version 3.0, or  
↪ any later versions of those licenses.

#### 1.13. "Source Code Form"

means the form of the work preferred for making modifications.

#### 1.14. "You" (or "Your")

means an individual or a legal entity exercising rights under this License. For  
↪ legal entities, "You" includes any entity that controls, is controlled by, or is  
↪ under common control with You. For purposes of this definition, "control" means (a)  
↪ the power, direct or indirect, to cause the direction or management of such entity,  
↪ whether by contract or otherwise, or (b) ownership of more than fifty percent (50%)  
↪ of the outstanding shares or beneficial ownership of such entity.

## 2. License Grants and Conditions

### 2.1. Grants

Each Contributor hereby grants You a world-wide, royalty-free, non-exclusive  
↪ license:

a. under intellectual property rights (other than patent or trademark)  
↪ Licensable by such Contributor to use, reproduce, make available, modify, display,  
↪ perform, distribute, and otherwise exploit its Contributions, either on an  
↪ unmodified basis, with Modifications, or as part of a Larger Work; and

b. under Patent Claims of such Contributor to make, use, sell, offer for sale,  
↪ have made, import, and otherwise transfer either its Contributions or its  
↪ Contributor Version.

### 2.2. Effective Date

The licenses granted in Section 2.1 with respect to any Contribution become  
↪ effective for each Contribution on the date the Contributor first distributes such  
↪ Contribution.

### 2.3. Limitations on Grant Scope

The licenses granted in this Section 2 are the only rights granted under this  
↪ License. No additional rights or licenses will be implied from the distribution or  
↪ licensing of Covered Software under this License. Notwithstanding Section 2.1(b)  
↪ above, no patent license is granted by a Contributor:

a. for any code that a Contributor has removed from Covered Software; or

b. for infringements caused by: (i) Your and any other third party's  
↪ modifications of Covered Software, or (ii) the combination of its Contributions

(continues on next page)

(continued from previous page)

↪with other software (except as part of its Contributor Version); or

c. under Patent Claims infringed by Covered Software in the absence of its  
↪Contributions.

This License does not grant any rights in the trademarks, service marks, or logos of  
↪any Contributor (except as may be necessary to comply with the notice requirements  
↪in Section 3.4).

#### 2.4. Subsequent Licenses

No Contributor makes additional grants as a result of Your choice to distribute the  
↪Covered Software under a subsequent version of this License (see Section 10.2) or  
↪under the terms of a Secondary License (if permitted under the terms of Section 3.  
↪3).

#### 2.5. Representation

Each Contributor represents that the Contributor believes its Contributions are its  
↪original creation(s) or it has sufficient rights to grant the rights to its  
↪Contributions conveyed by this License.

#### 2.6. Fair Use

This License is not intended to limit any rights You have under applicable copyright  
↪doctrines of fair use, fair dealing, or other equivalents.

#### 2.7. Conditions

Sections 3.1, 3.2, 3.3, and 3.4 are conditions of the licenses granted in Section 2.1.

### 3. Responsibilities

#### 3.1. Distribution of Source Form

All distribution of Covered Software in Source Code Form, including any Modifications  
↪that You create or to which You contribute, must be under the terms of this License.  
↪ You must inform recipients that the Source Code Form of the Covered Software is  
↪governed by the terms of this License, and how they can obtain a copy of this  
↪License. You may not attempt to alter or restrict the recipients' rights in the  
↪Source Code Form.

#### 3.2. Distribution of Executable Form

If You distribute Covered Software in Executable Form then:

a. such Covered Software must also be made available in Source Code Form, as  
↪described in Section 3.1, and You must inform recipients of the Executable Form how  
↪they can obtain a copy of such Source Code Form by reasonable means in a timely  
↪manner, at a charge no more than the cost of distribution to the recipient; and

b. You may distribute such Executable Form under the terms of this License, or  
↪sublicense it under different terms, provided that the license for the Executable  
↪Form does not attempt to limit or alter the recipients' rights in the Source Code  
↪Form under this License.

#### 3.3. Distribution of a Larger Work

You may create and distribute a Larger Work under terms of Your choice, provided that  
↪You also comply with the requirements of this License for the Covered Software. If  
↪the Larger Work is a combination of Covered Software with a work governed by one or  
↪more Secondary Licenses, and the Covered Software is not Incompatible With  
↪Secondary Licenses, this License permits You to additionally distribute such  
↪Covered Software under the terms of such Secondary License(s), so that the

(continues on next page)

(continued from previous page)

→recipient of the Larger Work may, at their option, further distribute the Covered  
→Software under the terms of either this License or such Secondary License(s).

### 3.4. Notices

You may not remove or alter the substance of any license notices (including copyright  
→notices, patent notices, disclaimers of warranty, or limitations of liability)  
→contained within the Source Code Form of the Covered Software, except that You may  
→alter any license notices to the extent required to remedy known factual  
→inaccuracies.

### 3.5. Application of Additional Terms

You may choose to offer, and to charge a fee for, warranty, support, indemnity or  
→liability obligations to one or more recipients of Covered Software. However, You  
→may do so only on Your own behalf, and not on behalf of any Contributor. You must  
→make it absolutely clear that any such warranty, support, indemnity, or liability  
→obligation is offered by You alone, and You hereby agree to indemnify every  
→Contributor for any liability incurred by such Contributor as a result of warranty,  
→support, indemnity or liability terms You offer. You may include additional  
→disclaimers of warranty and limitations of liability specific to any jurisdiction.

## 4. Inability to Comply Due to Statute or Regulation

If it is impossible for You to comply with any of the terms of this License with  
→respect to some or all of the Covered Software due to statute, judicial order, or  
→regulation then You must: (a) comply with the terms of this License to the maximum  
→extent possible; and (b) describe the limitations and the code they affect. Such  
→description must be placed in a text file included with all distributions of the  
→Covered Software under this License. Except to the extent prohibited by statute or  
→regulation, such description must be sufficiently detailed for a recipient of  
→ordinary skill to be able to understand it.

## 5. Termination

5.1. The rights granted under this License will terminate automatically if You fail  
→to comply with any of its terms. However, if You become compliant, then the rights  
→granted under this License from a particular Contributor are reinstated (a)  
→provisionally, unless and until such Contributor explicitly and finally terminates  
→Your grants, and (b) on an ongoing basis, if such Contributor fails to notify You  
→of the non-compliance by some reasonable means prior to 60 days after You have come  
→back into compliance. Moreover, Your grants from a particular Contributor are  
→reinstated on an ongoing basis if such Contributor notifies You of the non-  
→compliance by some reasonable means, this is the first time You have received  
→notice of non-compliance with this License from such Contributor, and You become  
→compliant prior to 30 days after Your receipt of the notice.

5.2. If You initiate litigation against any entity by asserting a patent infringement  
→claim (excluding declaratory judgment actions, counter-claims, and cross-claims)  
→alleging that a Contributor Version directly or indirectly infringes any patent,  
→then the rights granted to You by any and all Contributors for the Covered Software  
→under Section 2.1 of this License shall terminate.

5.3. In the event of termination under Sections 5.1 or 5.2 above, all end user  
→license agreements (excluding distributors and resellers) which have been validly  
→granted by You or Your distributors under this License prior to termination shall  
→survive termination.

## 6. Disclaimer of Warranty

Covered Software is provided under this License on an "as is" basis, without warranty

(continues on next page)

(continued from previous page)

↳ of any kind, either expressed, implied, or statutory, including, without limitation,  
↳ warranties that the Covered Software is free of defects, merchantable, fit for a  
↳ particular purpose or non-infringing. The entire risk as to the quality and  
↳ performance of the Covered Software is with You. Should any Covered Software prove  
↳ defective in any respect, You (not any Contributor) assume the cost of any  
↳ necessary servicing, repair, or correction. This disclaimer of warranty constitutes  
↳ an essential part of this License. No use of any Covered Software is authorized  
↳ under this License except under this disclaimer.

#### 7. Limitation of Liability

Under no circumstances and under no legal theory, whether tort (including negligence),  
↳ contract, or otherwise, shall any Contributor, or anyone who distributes Covered  
↳ Software as permitted above, be liable to You for any direct, indirect, special,  
↳ incidental, or consequential damages of any character including, without limitation,  
↳ damages for lost profits, loss of goodwill, work stoppage, computer failure or  
↳ malfunction, or any and all other commercial damages or losses, even if such party  
↳ shall have been informed of the possibility of such damages. This limitation of  
↳ liability shall not apply to liability for death or personal injury resulting from  
↳ such party's negligence to the extent applicable law prohibits such limitation.  
↳ Some jurisdictions do not allow the exclusion or limitation of incidental or  
↳ consequential damages, so this exclusion and limitation may not apply to You.

#### 8. Litigation

Any litigation relating to this License may be brought only in the courts of a  
↳ jurisdiction where the defendant maintains its principal place of business and such  
↳ litigation shall be governed by laws of that jurisdiction, without reference to its  
↳ conflict-of-law provisions. Nothing in this Section shall prevent a party's ability  
↳ to bring cross-claims or counter-claims.

#### 9. Miscellaneous

This License represents the complete agreement concerning the subject matter hereof.  
↳ If any provision of this License is held to be unenforceable, such provision shall  
↳ be reformed only to the extent necessary to make it enforceable. Any law or  
↳ regulation which provides that the language of a contract shall be construed  
↳ against the drafter shall not be used to construe this License against a  
↳ Contributor.

#### 10. Versions of the License

##### 10.1. New Versions

Mozilla Foundation is the license steward. Except as provided in Section 10.3, no one  
↳ other than the license steward has the right to modify or publish new versions of  
↳ this License. Each version will be given a distinguishing version number.

##### 10.2. Effect of New Versions

You may distribute the Covered Software under the terms of the version of the License  
↳ under which You originally received the Covered Software, or under the terms of any  
↳ subsequent version published by the license steward.

##### 10.3. Modified Versions

If you create software not governed by this License, and you want to create a new  
↳ license for such software, you may create and use a modified version of this  
↳ License if you rename the license and remove any references to the name of the  
↳ license steward (except to note that such modified license differs from this  
↳ License).

##### 10.4. Distributing Source Code Form that is Incompatible With Secondary Licenses

(continues on next page)

(continued from previous page)

If You choose to distribute Source Code Form that is Incompatible With Secondary  
↳ Licenses under the terms of this version of the License, the notice described in  
↳ Exhibit B of this License must be attached.

#### Exhibit A - Source Code Form License Notice

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.  
↳ 0. If a copy of the MPL was not distributed with this file, You can obtain one at  
↳ <https://mozilla.org/MPL/2.0/>.

If it is not possible or desirable to put the notice in a particular file, then You  
↳ may include the notice in a location (such as a LICENSE file in a relevant  
↳ directory) where a recipient would be likely to look for such a notice.

You may add additional accurate notices of copyright ownership.

#### Exhibit B - "Incompatible With Secondary Licenses" Notice

This Source Code Form is "Incompatible With Secondary Licenses", as defined by the  
↳ Mozilla Public License, v. 2.0.

### 1.3.7.11 Fluent Helm Charts

Apache License  
Version 2.0, January 2004  
<http://www.apache.org/licenses/>

#### TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

##### 1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,  
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by  
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all  
other entities that control, are controlled by, or are under common  
control with that entity. For the purposes of this definition,  
"control" means (i) the power, direct or indirect, to cause the  
direction or management of such entity, whether by contract or  
otherwise, or (ii) ownership of fifty percent (50%) or more of the  
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity  
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,  
including but not limited to software source code, documentation  
source, and configuration files.

"Object" form shall mean any form resulting from mechanical  
transformation or translation of a Source form, including but  
not limited to compiled object code, generated documentation,  
and conversions to other media types.

(continues on next page)

(continued from previous page)

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.
3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.
4. Redistribution. You may reproduce and distribute copies of the

(continues on next page)

(continued from previous page)

Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

- (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and
- (b) You must cause any modified files to carry prominent notices stating that You changed the files; and
- (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
- (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. **Submission of Contributions.** Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.
6. **Trademarks.** This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.
7. **Disclaimer of Warranty.** Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS,

(continues on next page)

(continued from previous page)

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. **Limitation of Liability.** In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.
9. **Accepting Warranty or Additional Liability.** While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

### 1.3.7.12 Fmtlog

MIT License

Copyright (c) 2021 Meng Rao <raomeng1@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.



### 1.3.7.13 GDRCopy

Copyright (c) 2014-2021, NVIDIA CORPORATION. All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

### 1.3.7.14 Google APIs

Apache License  
Version 2.0, January 2004  
<http://www.apache.org/licenses/>

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

#### 1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but

(continues on next page)

(continued from previous page)

not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.
3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

(continues on next page)

(continued from previous page)

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:
  - (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and
  - (b) You must cause any modified files to carry prominent notices stating that You changed the files; and
  - (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
  - (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.
6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.
7. Disclaimer of Warranty. Unless required by applicable law or

(continues on next page)

(continued from previous page)

agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. **Limitation of Liability.** In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.
9. **Accepting Warranty or Additional Liability.** While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[ ]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");  
you may not use this file except in compliance with the License.  
You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

(continues on next page)

(continued from previous page)

See the License for the specific language governing permissions and limitations under the License.

### 1.3.7.15 GoogleTest

Copyright 2008, Google Inc.  
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- \* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- \* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

- \* Neither the name of Google Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

### 1.3.7.16 gRPC

Apache License  
Version 2.0, January 2004  
<http://www.apache.org/licenses/>

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

#### 1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common

(continues on next page)

(continued from previous page)

control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

(continues on next page)

(continued from previous page)

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.
4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:
  - (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and
  - (b) You must cause any modified files to carry prominent notices stating that You changed the files; and
  - (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
  - (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

(continues on next page)

(continued from previous page)

5. **Submission of Contributions.** Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.
6. **Trademarks.** This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.
7. **Disclaimer of Warranty.** Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.
8. **Limitation of Liability.** In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.
9. **Accepting Warranty or Additional Liability.** While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[ ]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a

(continues on next page)



(continued from previous page)

file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

-----  
BSD 3-Clause License

Copyright 2016, Google Inc.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

-----  
Mozilla Public License, v. 2.0

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at <https://mozilla.org/MPL/2.0/>.

### 1.3.7.17 libuv

Copyright (c) 2015-present libuv project contributors.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

====

This license applies to parts of libuv originating from the <https://github.com/joyent/libuv> repository:

====

Copyright Joyent, Inc. and other Node contributors. All rights reserved. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

====

This license applies to all parts of libuv that are not externally maintained libraries.

The externally maintained libraries used by libuv are:

- tree.h (from FreeBSD), copyright Niels Provos. Two clause BSD license.
- inet\_pton and inet\_ntop implementations, contained in src/inet.c, are

(continues on next page)

(continued from previous page)

copyright the Internet Systems Consortium, Inc., and licensed under the ISC license.

- stdint-msvc2008.h (from msinttypes), copyright Alexander Chemeris. Three clause BSD license.
- pthread-fixes.c, copyright Google Inc. and Sony Mobile Communications AB. Three clause BSD license.
- android-ifaddrs.h, android-ifaddrs.c, copyright Berkeley Software Design Inc, Kenneth MacKay and Emergya (Cloud4all, FP7/2007-2013, grant agreement n° 289016). Three clause BSD license.

### 1.3.7.18 LibYAML

Copyright (c) 2017-2020 Ingy döt Net  
Copyright (c) 2006-2016 Kirill Simonov

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

### 1.3.7.19 Libyang

Copyright (c) 2015-2021, CESNET  
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- \* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- \* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- \* Neither the name of CESNET nor the names of

(continues on next page)

(continued from previous page)

contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

### 1.3.7.20 Mimalloc

MIT License

Copyright (c) 2018-2021 Microsoft Corporation, Daan Leijen

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

### 1.3.7.21 Prometheus Client Library for Modern C++

MIT License

Copyright (c) 2016-2019 Jupp Mueller  
Copyright (c) 2017-2019 Gregor Jasny

And many contributors, see  
<https://github.com/jupp0r/prometheus-cpp/graphs/contributors>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell

(continues on next page)

(continued from previous page)

copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

### 1.3.7.22 Protocol Buffers

Copyright 2008 Google Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- \* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- \* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- \* Neither the name of Google Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Code generated by the Protocol Buffer compiler is owned by the owner of the input file used when generating it. This code is not standalone and requires a support library to be linked with it. This support library is itself covered by the above license.

### 1.3.7.23 protoc-gen-validate (PGV)

Apache License  
Version 2.0, January 2004  
<http://www.apache.org/licenses/>

#### TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

##### 1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to

(continues on next page)

(continued from previous page)

communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.
3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.
4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:
  - (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and
  - (b) You must cause any modified files to carry prominent notices stating that You changed the files; and
  - (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
  - (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed

(continues on next page)

(continued from previous page)

as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. **Submission of Contributions.** Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.
6. **Trademarks.** This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.
7. **Disclaimer of Warranty.** Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.
8. **Limitation of Liability.** In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.
9. **Accepting Warranty or Additional Liability.** While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity,

(continues on next page)



(continued from previous page)

or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[ ]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");  
you may not use this file except in compliance with the License.  
You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

### 1.3.7.24 RE2

```
// Copyright (c) 2009 The RE2 Authors. All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
// met:
//
// * Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
// * Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following disclaimer
// in the documentation and/or other materials provided with the
// distribution.
// * Neither the name of Google Inc. nor the names of its
// contributors may be used to endorse or promote products derived from
// this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
```

(continues on next page)

(continued from previous page)

```
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

### 1.3.7.25 UDPA API

Apache License  
Version 2.0, January 2004  
<http://www.apache.org/licenses/>

#### TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

##### 1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the

(continues on next page)

(continued from previous page)

editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.
3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.
4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:
  - (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and
  - (b) You must cause any modified files to carry prominent notices

(continues on next page)

(continued from previous page)

stating that You changed the files; and

- (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
- (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

- 5. **Submission of Contributions.** Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.
- 6. **Trademarks.** This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.
- 7. **Disclaimer of Warranty.** Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.
- 8. **Limitation of Liability.** In no event and under no legal theory,

(continues on next page)

(continued from previous page)

whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. **Accepting Warranty or Additional Liability.** While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[ ]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");  
you may not use this file except in compliance with the License.  
You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

### 1.3.7.26 zlib

(C) 1995-2017 Jean-loup Gailly and Mark Adler

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Jean-loup Gailly  
jloup@gzip.org

Mark Adler  
madler@alumni.caltech.edu

### 1.3.7.27 CLI11

CLI11 2.2 Copyright (c) 2017-2024 University of Cincinnati, developed by Henry Schreiner under NSF AWARD 1414736. All rights reserved.

Redistribution and use in source and binary forms of CLI11, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

### 1.3.7.28 gsl-lite

The MIT License (MIT)

Copyright (c) 2015-2019 Martin Moene  
Copyright (c) 2019-2021 Moritz Beutel  
Copyright (c) 2015-2018 Microsoft Corporation. All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

### 1.3.7.29 cmake-modules

Boost Software License - Version 1.0 - August 17th, 2003

Permission is hereby granted, free of charge, to any person or organization obtaining a copy of the software and accompanying documentation covered by this license (the "Software") to use, reproduce, display, distribute, execute, and transmit the Software, and to prepare derivative works of the Software, and to permit third-parties to whom the Software is furnished to do so, all subject to the following:

The copyright notices in the Software and this entire statement, including the above license grant, this restriction and the following disclaimer, must be included in all copies of the Software, in whole or in part, and all derivative works of the Software, unless such copies or derivative works are solely in the form of machine-executable object code generated by a source language processor.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

This section describes the supported configurations, test-vector configurations, and limitations for this release of Aerial cuPHY.

## 1.4. cuBB Installation Guide

This section describes how to install the Aerial cuBB.

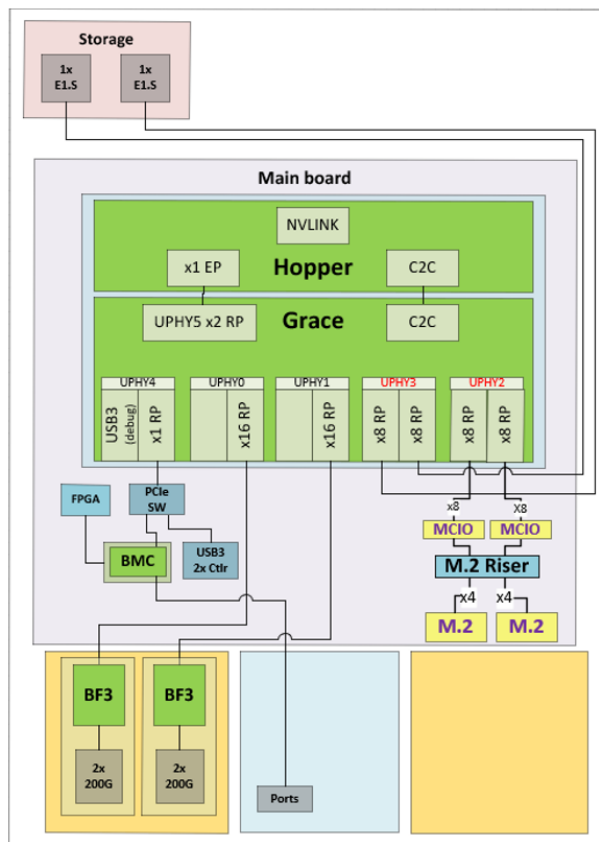
### 1.4.1. Installing Tools on Grace Hopper MGX System

This chapter describes how to install the required kernel, driver, and tools on the host. This is a one-time installation and can be skipped if the system has been configured already.

- ▶ In the following sequence of steps, the target host is [Supermicro Grace Hopper MGX System](#).
- ▶ Depending on the release, tools that are installed in this section may need to be upgraded in the [Installing and Upgrading Aerial cuBB](#) section.
- ▶ After everything is installed and updated, refer to the [cuBB Quick Start Guide](#) for how to use Aerial cuBB.

#### 1.4.1.1 Supermicro Grace Hopper MGX Configuration

Supermicro Server SKU: *ARS-111GL-NHR (Config 2)*



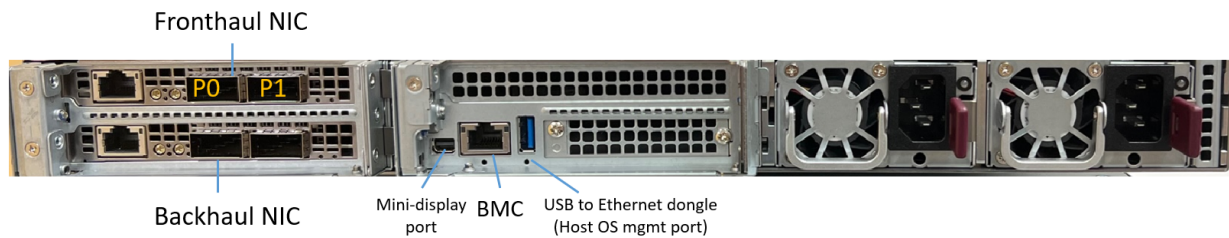
CPU Module	NVIDIA Grace Hopper Superchip, CG1
BIOS	SMC
BMC	SMC
Chassis length (mm)	900
Chassis width	19"
Chassis form factor	1U
Cooling	Air
M.2 1	960GB
M.2 2	TBD
Cabling	Hot
Left short slot 0	1x E1.5 4TB
Left short slot 1	
Center short slot 0	
Center short slot 1	
Right short slot 0	1x E1.5 4TB
Right short slot 1	
Left long slot 0	BF3 B3220 900-9D3B6-00CV-AA0
Left long slot 1	
Center slot 0	BF3 B3220 900-9D3B6-00CV-AA0
Center slot 1	IO board
Right long slot 0	2x 1600W CRPS
Right long slot 1	
Power	PSU
Storage bay	

Top View:





Back View:



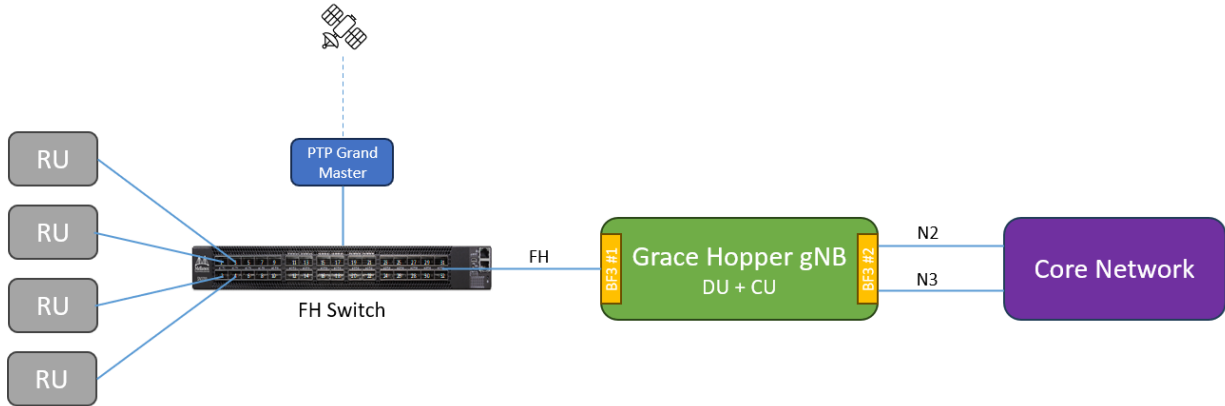
## 1.4.1.2 Cable Connection

### 1.4.1.2.1 Host OS Internet Connection

The BF3 NICs are reserved for fronthaul and backhaul connections, a USB to Ethernet dongle to the back USB port is recommended for the host OS internet connection.

### 1.4.1.2.2 E2E Test Connection

To run end-to-end test with O-RU, the BF3 fronthaul port#0 or port#1 must be connected to the fronthaul switch. Make sure the PTP is configured to use the port connected to the fronthaul switch. The following diagram shows a typical E2E connection in O-RAN LLS-C3 topology.



### 1.4.1.2.3 cuBB Test Connection

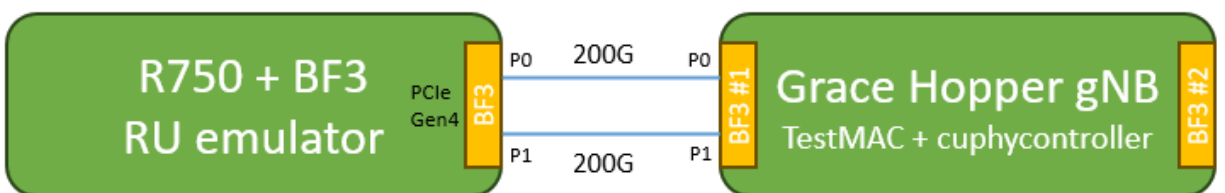
To run cuBB end-to-end test with TestMAC and RU emulator, a R750 RU emulator is recommended to pair with the Grace Hopper MGX system. The BF3 NIC (P/N: 900-9D3B6-00CV-AA0) should be installed on the slot 7 of the R750 server as the picture shown below.



To provision the R750 RU emulator, follow the instructions at [Installing Tools on Dell R750](#). Because the R750 RU emulator has no GPU, the [Installing CUDA Driver](#) can be skipped. Note that the PCI addresses of the BF3 ports are ca:00.0 and ca:00.1 on the R750 RU emulator.

```
$ lshw -c network -businfo
Bus info      Device      Class      Description
-----
pci@0000:04:00.0  eno8303    network    NetXtreme BCM5720 Gigabit Etherne
pci@0000:04:00.1  eno8403    network    NetXtreme BCM5720 Gigabit Etherne
pci@0000:ca:00.0  aerial00   network    MT43244 BlueField-3 integrated Co
pci@0000:ca:00.1  aerial01   network    MT43244 BlueField-3 integrated Co
```

The Mellanox 200GbE direct attach copper cable is required to connect the Grace Hopper MGX and R750 RU emulator to run more than 10 cells. The 100GbE direct attach copper cable should be able to support 10C 59c BFP9 but it is not going to work for 20C 60c BFP9.



To run RU emulator on R750 + BF3, update the RU emulator yaml as below:

```
# For RU Emulator on R750 system
sed -i "s/ul_core_list.*/ul_core_list: [5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,
↵37,39,41,43]/" $SRU_YAML
sed -i "s/dl_core_list.*/dl_core_list: [4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,
↵36,38,40,42]/" $SRU_YAML
sed -i "s/aerial_fh_split_rx_tx_mempool.*/aerial_fh_split_rx_tx_mempool: 1/" $SRU_YAML
sed -i "s/low_priority_core.*/low_priority_core: 45/" $SRU_YAML
```

### 1.4.1.3 System Firmware Upgrade

During the first boot, login to BMC to check the firmware inventory. Go to **Dashboard -> Maintenance -> Firmware Management -> Inventory** to see the current firmware versions.

The screenshot displays the BMC Firmware Management Inventory page. The left sidebar shows the navigation menu with 'Firmware Management' selected. The main content area has tabs for 'Update' and 'Inventory', with 'Inventory' active. A filter bar at the top allows filtering by component type: BMC, BIOS, ERoT, CPLD, NIC, Power Supply, GPU, and FPGA. Below the filter is a 'Platform Resiliency Actions' section with a table of components and their versions.

Name ↑	Version
BIOS	BIOS Date: 10/26/2023 Ver 1.0
BIOS Backup	BIOS Date: 10/26/2023 Ver 1.0
BIOS Golden	BIOS Date: 10/10/2023 Ver 0.1
BMC	01.02.01
BMC Backup	01.00.12
BMC Golden	01.00.11
BMC Staging	Not Present
CPLD BackplaneA BPN-E1S5-GP101M	CPLD_ID: 2929 Rev: 01
CPLD Motherboard Miscellaneous	CPLD_ID: 03E0 Rev: 0B
ERoT CPU 0	01.03.0114.0000_n01
ERoT CPU Golden	01.03.0103.0000_n01
ERoT GPU 0	01.03.0114.0000_n01
FPGA motherboard	0.8A
FPGA motherboard Golden	0.78
MGX GPU	96.00.84.00.02

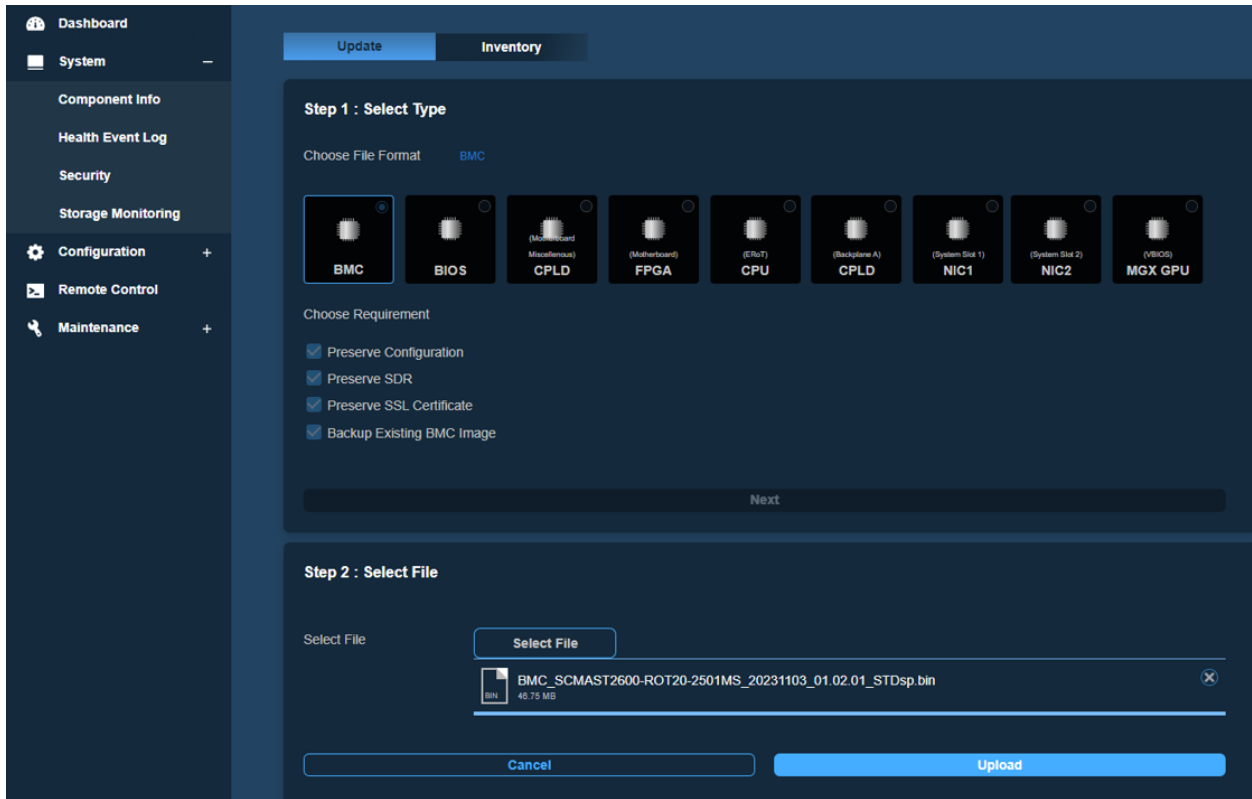
Here is the list of the minimum required versions. Upgrade the firmware to the following or newer versions, if your system has older firmware.

Component	Firmware Version	Firmware filename
BMC	1.02.01 (20231103)	BMC_SCMAST2600-ROT20-2501MS_20231103_01.02.01_STDsp.bin
BIOS	1.0 (20231026)	BIOS_G1SMH-G-1D31_20231026_1.0_STDsp.bin
FPGA	0.8A	FPGA_MBD-G1SMH-G-10XX1D31_20231018_00.8A.XX_STDsp.bin
VBIOS	96.00.84.00.02	g530_0206_888__9600840002-prod.fwpkg
EROT	1.03.0114.0000-n01	cec1736-ecfw-01.03.0114.0000-n01-rel-prod.fwpkg
CPLD Mother-board Misc	VOB	CPLD_XO3-GP03E0-10XX03E0_20231020_0B.XX.XX_STDsp.jed

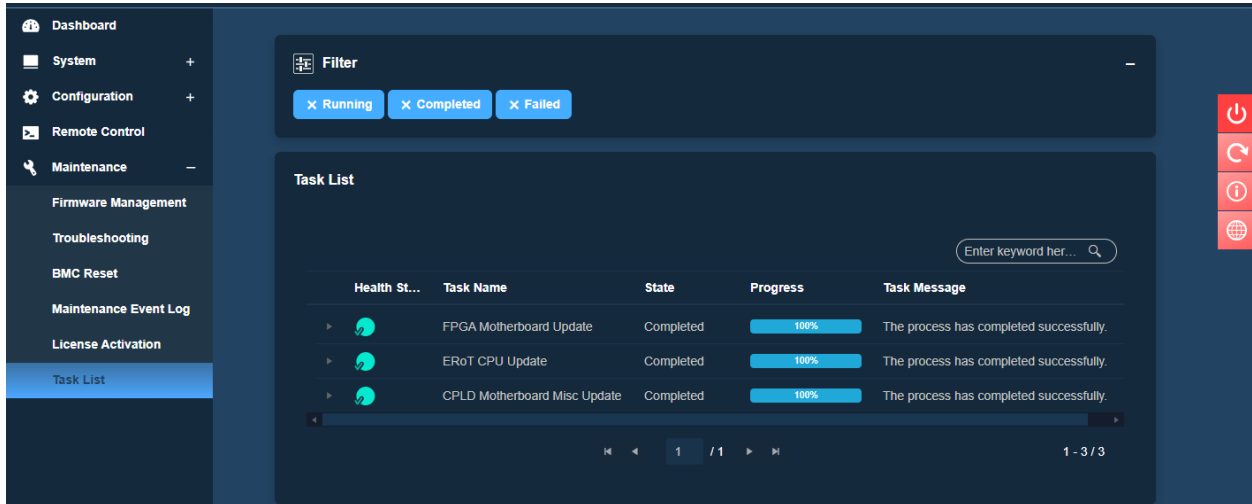
The recommended firmware update sequence is:

1. Power off host
2. Update BMC
3. Update CPLD motherboard misc
4. Update CPU ERoT
5. Update FPGA
6. A/C power cycle
7. Update BIOS
8. Update VBIOS
9. Reboot or Power cycle

To update the firmware for a specific component, go to **Dashboard -> Maintenance -> Firmware Management -> Update** then select the component icon -> **Next -> Select File -> Upload -> Update**. For example, select BMC and its firmware file as follows:



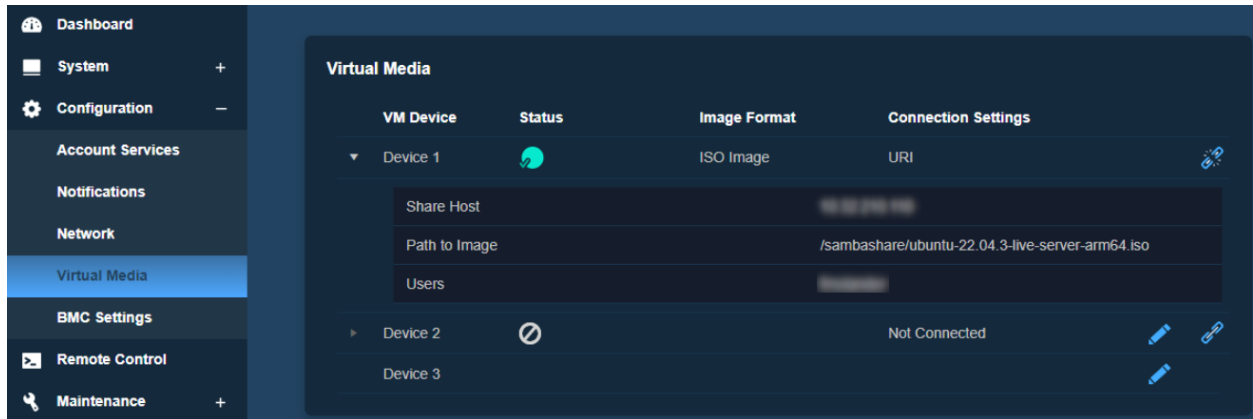
For non-BMC firmware update, it is queued in the task list to update in next boot.



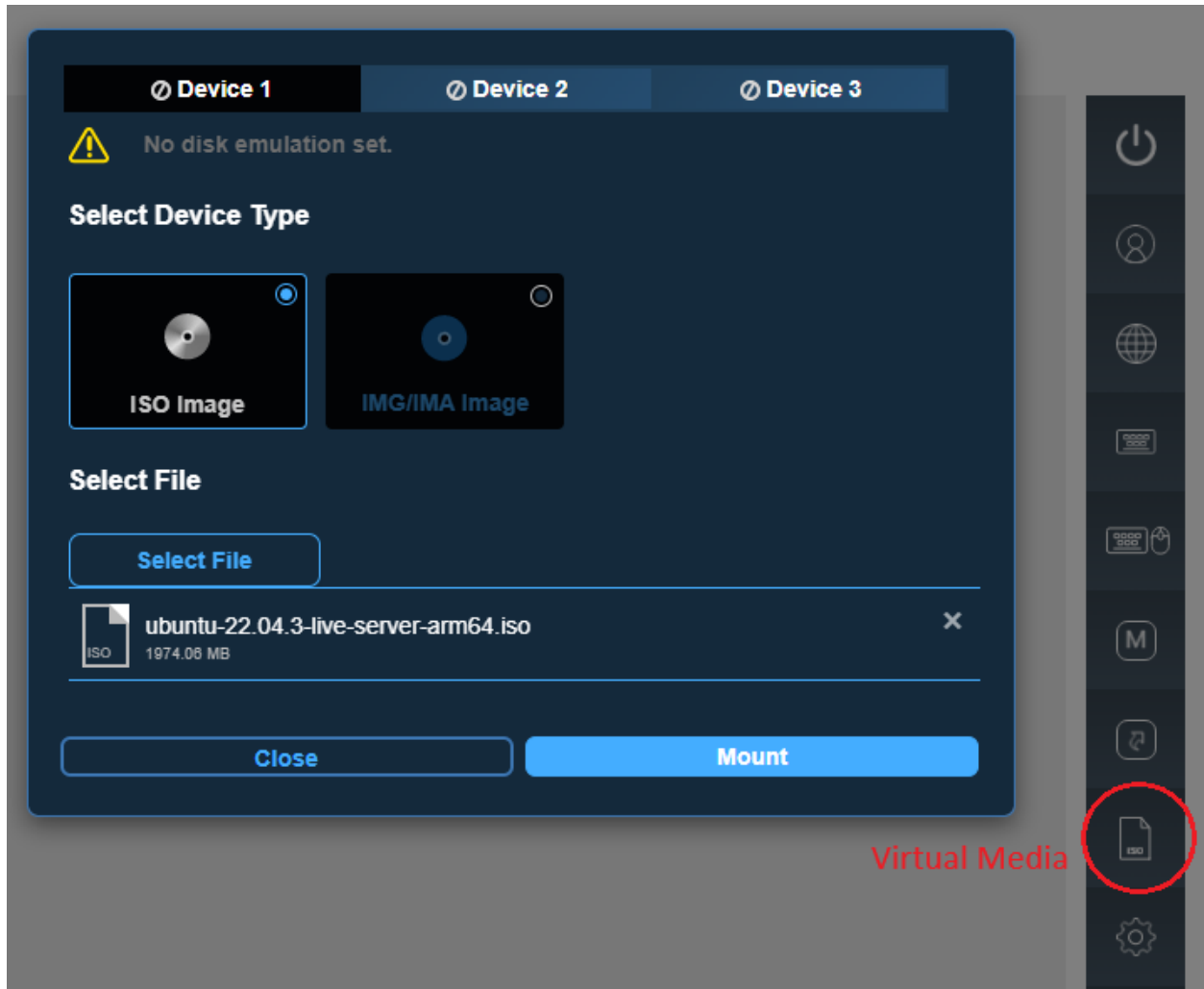
### 1.4.1.4 Install Ubuntu 22.04 Server

Download the Ubuntu server 22.04 ISO image for ARM-based system from <https://ubuntu.com/download/server/arm>. Before installing the system OS, prepare a bootable USB drive contains the OS image or configure the virtual media in the BMC for remote installation. Also verify that a USB to Ethernet dongle is connected to the back USB port for host internet access.

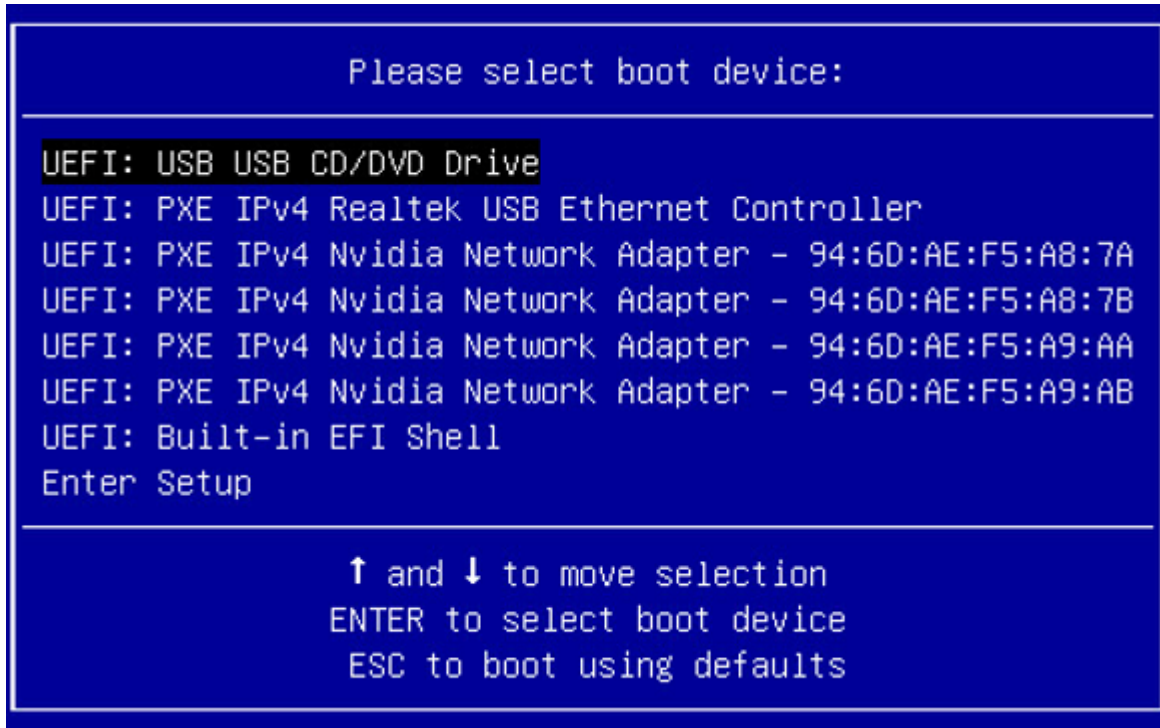
There are two ways to configure the virtual media. One is to share the OS ISO image by Windows network sharing or Samba sharing on Linux. Then go to BMC **Dashboard** -> **Configuration** -> **Virtual Media** to enter the virtual media connection info including the share host ip, image path, username and password. After the connection info is saved, click the **Link icon** to connect.



Another way to configure virtual media, is to select the Virtual Media icon from the remote console then mount the OS ISO image to the virtual CD/DVD drive.



Reboot the system after the virtual media is configured and connected. Press **F11** to enter the BIOS boot menu and select **UEFI: USB CD/DVD Drive** to boot with the virtual media.



Launch the SOL console from the BMC Remote Control menu. The SOL console is required to complete the Ubuntu OS installation.

### Note

The Ubuntu 22.04.3 installation media does not include a required patch for the resolution of an issue with the *ast* driver. The *ast* driver is used to interface with the BMC. The absence of this patch causes distorted output from the on-board display port and remote console. Because of this, the OS installation must be done on the *SOL console*. The fix is included in the NVIDIA optimized Ubuntu kernel. After installing the NVIDIA optimized Ubuntu kernel, the output of the on-board display and the remote console from BMC will be normal again.



After seeing the GRUB menu from the SOL console, select **Ubuntu Server with the HWE Kernel** to install the Ubuntu server OS.





Follow the Ubuntu installation process with the notable selection below:

- ▶ **Continue in rich mode**
- ▶ **Continue without updating**
- ▶ **Ubuntu Server**
- ▶ **Install OpenSSH server**

When the installation is done, the console shows **Install complete** and **Reboot now**. Reboot the system and check the following:

- ▶ Check if the system time is correct to avoid apt update error.

Run the following commands to set the date and time via NTP once (this will not enable the NTP service):

```
sudo apt-get install ntpdate
sudo ntpdate -s pool.ntp.org
```

- ▶ Check if the GPU and NIC are detected by the OS.

Use the following commands to determine whether the GPU and NIC are detected by the OS:

```
$ lspci | grep -i nvidia
# GH200 GPU
0009:01:00.0 3D controller: NVIDIA Corporation Device 2342 (rev a1)

$ lspci | grep -i mellanox
# The first BF3 NIC (Fronthaul NIC)
0000:01:00.0 Ethernet controller: Mellanox Technologies MT43244 BlueField-3
↳integrated ConnectX-7 network controller (rev 01)
0000:01:00.1 Ethernet controller: Mellanox Technologies MT43244 BlueField-3
```

(continues on next page)

(continued from previous page)

```

↪integrated ConnectX-7 network controller (rev 01)
0000:01:00.2 DMA controller: Mellanox Technologies MT43244 BlueField-3 SoC Management
↪Interface (rev 01)
# The second BF3 NIC (Backhaul NIC)
0002:01:00.0 Ethernet controller: Mellanox Technologies MT43244 BlueField-3
↪integrated ConnectX-7 network controller (rev 01)
0002:01:00.1 Ethernet controller: Mellanox Technologies MT43244 BlueField-3
↪integrated ConnectX-7 network controller (rev 01)
0002:01:00.2 DMA controller: Mellanox Technologies MT43244 BlueField-3 SoC Management
↪Interface (rev 01)

```

Use the following command to change the hostname:

```
$ sudo hostnamectl set-hostname NEW_HOSTNAME
```

To display the GRUB menu during boot, create `/etc/default/grub.d/menu.cfg` with the following content:

```

$ cat <<"EOF" | sudo tee /etc/default/grub.d/menu.cfg
GRUB_TIMEOUT_STYLE=menu
GRUB_TIMEOUT=5
GRUB_TERMINAL="console serial"
GRUB_CMDLINE_LINUX_DEFAULT=""
GRUB_SERIAL_COMMAND="$GRUB_SERIAL_COMMAND serial --unit=0 --speed=115200 --word=8 --
↪parity=no --stop=1"
EOF

```

#### 1.4.1.5 Configure the Network Interfaces

The following installation steps need an Internet connection. Ensure that you have the proper netplan config for your local network.

The network interface names could change after reboot. To ensure persistent network interface names after reboot, create a persistent net link files under `/etc/systemd/network`, one for each interface.

To find the MAC address of the BlueField-3 NIC, run `lshw` to check for network devices and look for the ConnectX-7 entries.

```

$ sudo apt-get install jq -y
$ sudo lshw -json -C network | jq '.[ ] | "\(.product), MAC: \(.serial)"' | grep
↪"ConnectX-7"
"MT43244 BlueField-3 integrated ConnectX-7 network controller, MAC: 94:6d:ae:ww:ww:ww"
"MT43244 BlueField-3 integrated ConnectX-7 network controller, MAC: 94:6d:ae:xx:xx:xx"
"MT43244 BlueField-3 integrated ConnectX-7 network controller, MAC: 94:6d:ae:yy:yy:yy"
"MT43244 BlueField-3 integrated ConnectX-7 network controller, MAC: 94:6d:ae:zz:zz:zz"

```

Create files at `/etc/systemd/network/` with the desired name for the interface and the MAC address found in the previous step.

#### Note

The rest of the document will assume the `aerial00` and `aerial01` interfaces are the ones connected to the RU emulator for the cuBB testing or the fronthaul switch for the E2E tests and that `aerial00` is the interface used for PTP.

```
$ sudo nano /etc/systemd/network/20-aerial00.link

[Match]
MACAddress=94:6d:ae:ww:ww:ww

[Link]
Name=aerial00

$ sudo nano /etc/systemd/network/20-aerial01.link

[Match]
MACAddress=94:6d:ae:xx:xx:xx

[Link]
Name=aerial01

$ sudo nano /etc/systemd/network/20-aerial02.link

[Match]
MACAddress=94:6d:ae:yy:yy:yy

[Link]
Name=aerial02

$ sudo nano /etc/systemd/network/20-aerial03.link

[Match]
MACAddress=94:6d:ae:zz:zz:zz

[Link]
Name=aerial03
```

To apply the change:

```
$ sudo netplan apply
```

#### 1.4.1.6 Disable Auto Upgrade

Edit the `/etc/apt/apt.conf.d/20auto-upgrades` system file, and change the “1” to “0” for both lines. This prevents the installed version of the low latency kernel from being accidentally changed with a subsequent software upgrade.

```
$ sudo nano /etc/apt/apt.conf.d/20auto-upgrades
APT::Periodic::Update-Package-Lists "0";
APT::Periodic::Unattended-Upgrade "0";
```

Disable the `fwupd-refresh` timer to prevent `fwupdmg` from automatically checking for any updates.

```
$ sudo systemctl mask fwupd-refresh.timer
```

### 1.4.1.7 Install NVIDIA Optimized Ubuntu Kernel

Run the following commands to install the NVIDIA optimized Ubuntu kernel.

```
$ sudo apt update
# NOTE: This will install the specific kernel version, not the latest NVIDIA optimized
↪kernel.
$ sudo apt install -y linux-image-6.5.0-1019-nvidia-64k
```

Then, update the GRUB to change the default boot kernel. The version to use here depends on the latest version that was installed with the previous command:

```
# Update grub to change the default boot kernel
$ sudo sed -i 's/^GRUB_DEFAULT=.*GRUB_DEFAULT="Advanced options for Ubuntu>Ubuntu,
↪with Linux 6.5.0-1019-nvidia-64k"/' /etc/default/grub
```

### 1.4.1.8 Configure Linux Kernel Command-line

Ensure the **iommu.passthrough=y** kernel parameter is NOT passed to the kernel. This parameter prevents the GPU driver from loading so it must be removed if it is present.

Verify that the parameter is present by running:

```
$ grep iommu.passthrough=y /proc/cmdline
```

If the parameter is present, find the file that contains this parameter and remove it. For example:

```
$ grep -rns iommu.passthrough /etc/default/grub*
# Remove iommu.passthrough=y from the found file
$ sudo sed -i 's/ iommu.passthrough=y//' /etc/default/<found file>
```

To set kernel command-line parameters, edit the GRUB\_CMDLINE\_LINUX parameter in the grub file /etc/default/grub.d/cmdline.cfg and append or update the parameters described below. The following kernel parameters are optimized for GH200. To automatically append the grub file with these parameters, enter this command:

```
$ cat <<"EOF" | sudo tee /etc/default/grub.d/cmdline.cfg
GRUB_CMDLINE_LINUX="$GRUB_CMDLINE_LINUX pci=realloc=off pci=pcie_bus_safe default_
↪hugepagesz=512M hugepagesz=512M hugepages=48 tsc=reliable processor.max_cstate=0
↪audit=0 idle=poll rcu_nocb_poll nosoftlockup irqaffinity=0 isolcpus=managed_irq,
↪domain,4-64 nohz_full=4-64 rcu_nocbs=4-64 earlycon module_blacklist=nouveau acpi_
↪power_meter.force_cap_on=y numa_balancing=disable init_on_alloc=0 preempt=none"
EOF
```

#### Note

The hugepage size is 512MB which is optimized for the 64k page size kernel on ARM.

### 1.4.1.9 Apply the Changes and Reboot to Load the Kernel

```
$ sudo update-grub
$ sudo reboot
```

After rebooting, enter this command to verify that the kernel command-line parameters are configured properly:

```
$ uname -r
6.5.0-1019-nvidia-64k

$ cat /proc/cmdline
BOOT_IMAGE=/vmlinuz-6.5.0-1019-nvidia-64k root=/dev/mapper/ubuntu--vg-ubuntu--lv ro
↪pci=realloc=off pci=pcie_bus_safe default_hugepagesz=512M hugepagesz=512M
↪hugepages=48 tsc=reliable processor.max_cstate=0 audit=0 idle=poll rcu_nocb_poll
↪nosoftlockup irqaffinity=0 isolcpus=managed_irq,domain,4-64 nohz_full=4-64 rcu_
↪nocbs=4-64 earlycon module_blacklist=nouveau acpi_power_meter.force_cap_on=y numa_
↪balancing=disable init_on_alloc=0 preempt=none
```

Enter this command to check if hugepages are enabled:

```
$ grep -i huge /proc/meminfo
AnonHugePages:          0 kB
ShmemHugePages:        0 kB
FileHugePages:         0 kB
HugePages_Total:       48
HugePages_Free:        48
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:         524288 kB
Hugetlb:               25165824 kB
```

### 1.4.1.10 Install Dependency Packages

Enter these commands to install the prerequisite packages:

```
$ sudo apt-get update
$ sudo apt-get install -y build-essential linux-headers-$(uname -r) dkms unzip
↪linuxptp pv apt-utils net-tools
```

### 1.4.1.11 Install DOCA OFED and Mellanox Firmware Tools on the Host

Check if there is an existing MOFED installed on the host system.

```
$ ofed_info -s
OFED-internal-23.10-1.1.9:
```

Uninstall MOFED if it is present.

```
$ sudo /usr/sbin/ofed_uninstall.sh
```

Enter the following commands to install DOCA OFED.

```
# Install DOCA OFED
$ wget https://www.mellanox.com/downloads/DOCA/DOCA_v2.7.0/host/doca-host_2.7.0-
↳204000-24.04-ubuntu2204_arm64.deb
$ sudo dpkg -i doca-host_2.7.0-204000-24.04-ubuntu2204_arm64.deb
$ sudo apt update
$ sudo apt install -y doca-ofed

# To check what version of OFED you have installed
$ ofed_info -s
OFED-internal-24.04-0.6.6:
```

Enter the following commands to install Mellanox firmware tools.

```
# Install Mellanox Firmware Tools
$ export MFT_VERSION=4.28.0-92
$ wget https://www.mellanox.com/downloads/MFT/mft-$MFT_VERSION-arm64-deb.tgz
$ tar xvf mft-$MFT_VERSION-arm64-deb.tgz
$ sudo mft-$MFT_VERSION-arm64-deb/install.sh

$ sudo mst version
mst, mft 4.28.0-92, built on Apr 25 2024, 15:22:48. Git SHA Hash: N/A

$ sudo mst start

# check NIC PCIe bus addresses and network interface names
$ sudo mst status -v
MST modules:
-----
    MST PCI module is not loaded
    MST PCI configuration module loaded
PCI devices:
-----
DEVICE_TYPE          MST                               PCI                               RDMA
↳NET                 NUMA
BlueField3(rev:1)    /dev/mst/mt41692_pciconf1.1    0002:01:00.1    mlx5_3
↳net-aerial03        0
BlueField3(rev:1)    /dev/mst/mt41692_pciconf1      0002:01:00.0    mlx5_2
↳net-aerial02        0
BlueField3(rev:1)    /dev/mst/mt41692_pciconf0.1    0000:01:00.1    mlx5_1
↳net-aerial01        0
BlueField3(rev:1)    /dev/mst/mt41692_pciconf0      0000:01:00.0    mlx5_0
↳net-aerial00        0
```

Enter these commands to check the link status of port 0:

```
# Here is an example if the port 0 of fronthaul NIC is connected to another server or
↳switch via a 200GbE DAC cable.
$ sudo mlxlink -d 0000:01:00.0
```

Operational Info

```
-----
State                : Active
Physical state       : LinkUp
Speed                : 200G
Width                : 4x
FEC                  : Standard_RS-FEC - (544,514)
Loopback Mode        : No Loopback
```

(continues on next page)

(continued from previous page)

```

Auto Negotiation           : ON

Supported Info
-----
Enabled Link Speed (Ext.)  : 0x00003ff2 (200G_2X,200G_4X,100G_1X,100G_2X,100G_
↳4X,50G_1X,50G_2X,40G,25G,10G,1G)
Supported Cable Speed (Ext.) : 0x000017f2 (200G_4X,100G_2X,100G_4X,50G_1X,50G_
↳2X,40G,25G,10G,1G)

Troubleshooting Info
-----
Status Opcode              : 0
Group Opcode               : N/A
Recommendation             : No issue was observed

Tool Information
-----
Firmware Version          : 32.39.2048
amBER Version             : 2.22
MFT Version               : mft 4.26.1-3

```

#### 1.4.1.12 Install CUDA Driver

If the system has an older driver installed, unload the current driver modules and uninstall the old driver, using the following:

```

# Unload the current driver modules
$ for m in $(lsmod | awk "/^[^[:space:]]*(nvidia|nv_|gdrdrv)/ {print \$1}"); do echo
↳Unload $m...; sudo rmmod $m; done

# Remove the driver if it was installed by runfile installer before.
$ sudo /usr/bin/nvidia-uninstall

```

Create the driver module config with the following recommended settings:

```

$ cat <<EOF | sudo tee /etc/modprobe.d/nvidia.conf
options nvidia NVreg_RegistryDwords="RMNvLinkDisableLinks=0x3FFFF;"
EOF

```

Run the following commands to install the **NVIDIA open-source GPU kernel driver** (OpenRM).

```

# Install NVIDIA GPU driver 560.35.03 to run Aerial L1 in non-MIG mode.
$ wget https://us.download.nvidia.com/XFree86/aarch64/560.35.03/NVIDIA-Linux-aarch64-
↳560.35.03.run
$ sudo sh NVIDIA-Linux-aarch64-560.35.03.run --silent -m kernel-open

```

*# Verify that the driver is loaded successfully*

```

$ nvidia-smi
+-----+
| NVIDIA-SMI 560.35.03           Driver Version: 560.35.03           CUDA Version: 12.6
|
+-----+-----+-----+-----+
| GPU   Name                     Persistence-M | Bus-Id              Disp.A | Volatile Uncorr.

```

(continues on next page)





#### 1.4.1.14 Install Docker CE

The full official instructions for installing Docker CE can be found here: <https://docs.docker.com/engine/install/ubuntu/#install-docker-engine>. The following instructions are one supported way of installing Docker CE:

##### Warning

To work correctly, the CUDA driver must be installed before Docker CE or nvidia-container-toolkit installation. It is recommended that you install the CUDA driver before installing Docker CE or the nvidia-container-toolkit.

```
$ sudo apt-get update
$ sudo apt-get install -y ca-certificates curl gnupg
$ sudo install -m 0755 -d /etc/apt/keyrings
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /
→etc/apt/keyrings/docker.gpg
$ sudo chmod a+r /etc/apt/keyrings/docker.gpg
$ echo \
    "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg]
→https://download.docker.com/linux/ubuntu \
    "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
    sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
$ sudo apt-get update
$ sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin
→docker-compose-plugin
$ sudo docker run --rm hello-world
```

#### 1.4.1.15 Install the Nvidia Container Toolkit

Locate and follow the nvidia-container-toolkit [install instructions](#).

Or use the following instructions as an alternate way to install the nvidia-container-toolkit. Version **1.16.2** is supported.

##### Warning

To work correctly, the CUDA driver must be installed before Docker CE or nvidia-container-toolkit installation. It is recommended that you install the CUDA driver before installing Docker CE or the nvidia-container-toolkit.

```
$ curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey | sudo gpg --dearmor
→-o /usr/share/keyrings/nvidia-container-toolkit-keyring.gpg \
    && curl -s -L https://nvidia.github.io/libnvidia-container/stable/deb/nvidia-
→container-toolkit.list | \
    sed 's#deb https://#deb [signed-by=/usr/share/keyrings/nvidia-container-toolkit-
→keyring.gpg] https://#g' | \
    sudo tee /etc/apt/sources.list.d/nvidia-container-toolkit.list \
    && \
    sudo apt-get update
$ sudo apt-get install -y nvidia-container-toolkit
```

(continues on next page)

(continued from previous page)

```
$ sudo nvidia-ctk runtime configure --runtime=docker
$ sudo systemctl restart docker
$ sudo docker run --rm --runtime=nvidia --gpus all ubuntu nvidia-smi
```

**Note**

If you have nvidia-container-toolkit installed on the existing system, check the version by running the `nvidia-ctk --version` command. If it is older than 1.16.2, run the following commands to upgrade to the current version:

```
$ nvidia-ctk --version
NVIDIA Container Toolkit CLI version 1.14.4
commit: d167812ce3a55ec04ae2582eff1654ec812f42e1

$ sudo apt update
$ sudo apt-get install -y nvidia-container-toolkit

$ nvidia-ctk --version
NVIDIA Container Toolkit CLI version 1.16.2
commit: a5a5833c14a15fd9c86bcece85d5ec6621b65652
```

**1.4.1.16 Update BF3 BFB Image and NIC Firmware****Note**

- ▶ The following instructions are for BF3 NIC (**OPN: 900-9D3B6-00CV-A; PSID: MT\_000000884**) specifically.
- ▶ There is no need to switch to DPU mode if using the BFB image below.
- ▶ This BFB image will update the NIC firmware automatically.

```
# Enable MST
```

```
$ sudo mst start
$ sudo mst status
```

```
MST modules:
```

```
-----
MST PCI module is not loaded
MST PCI configuration module loaded
```

```
MST devices:
```

```
-----
/dev/mst/mt41692_pciconf0      - PCI configuration cycles access.
                               domain:bus:dev.fn=0000:01:00.0 addr.reg=88 data.
↔reg=92 cr_bar.gw_offset=-1
                               Chip revision is: 01
/dev/mst/mt41692_pciconf1      - PCI configuration cycles access.
                               domain:bus:dev.fn=0002:01:00.0 addr.reg=88 data.
↔reg=92 cr_bar.gw_offset=-1
                               Chip revision is: 01
```

(continues on next page)

(continued from previous page)

```

# Download the BF3 BFB image
$ wget https://content.mellanox.com/BlueField/BFBs/Ubuntu22.04/bf-bundle-2.7.0-33_24.
->04_ubuntu-22.04_prod.bfb
# Update the BFB image of the 1st BF3
$ sudo bfb-install -r rshim0 -b bf-bundle-2.7.0-33_24.04_ubuntu-22.04_prod.bfb
# Update the BFB image of the 2nd BF3
$ sudo bfb-install -r rshim1 -b bf-bundle-2.7.0-33_24.04_ubuntu-22.04_prod.bfb

Pushing bfb
1.41GiB 0:01:24 [17.1MiB/s] [
->                                     <=>]
Collecting BlueField booting status. Press Ctrl+C to stop...
INFO[PSC]: PSC BL1 START
INFO[BL2]: start
INFO[BL2]: boot mode (rshim)
INFO[BL2]: VDDQ adjustment complete
INFO[BL2]: VDDQ: 1120 mV
INFO[BL2]: DDR POST passed
INFO[BL2]: UEFI loaded
INFO[BL31]: start
INFO[BL31]: lifecycle GA Secured
INFO[BL31]: VDD: 851 mV
ERR[BL31]: MB timeout
INFO[BL31]: runtime
INFO[UEFI]: eMMC init
INFO[UEFI]: eMMC probed
INFO[UEFI]: UPVS valid
INFO[UEFI]: PMI: updates started
INFO[UEFI]: PMI: total updates: 1
INFO[UEFI]: PMI: updates completed, status 0
INFO[UEFI]: PCIe enum start
INFO[UEFI]: PCIe enum end
INFO[UEFI]: UEFI Secure Boot (enabled)
INFO[UEFI]: Redfish enabled
INFO[BL31]: Partial NIC
INFO[BL31]: power capping disabled
INFO[UEFI]: exit Boot Service
INFO[MISC]: Ubuntu installation started
INFO[MISC]: Installing OS image
INFO[MISC]: Ubuntu installation completed
WARN[MISC]: Skipping BMC components upgrade.
INFO[MISC]: Updating NIC firmware...
INFO[MISC]: NIC firmware update done
INFO[MISC]: Installation finished

# Wait 10 minutes to ensure the card initializes properly after the BFB installation
$ sleep 600

# NOTE: Requires a full power cycle from host with cold boot

# Verify NIC FW version after reboot
$ sudo mst start
$ sudo flint -d /dev/mst/mt41692_pciconf0 q
Image type:          FS4

```

(continues on next page)

(continued from previous page)

```

FW Version:          32.41.1000
FW Release Date:    28.4.2024
Product Version:    32.41.1000
Rom Info:           type=UEFI Virtio net version=21.4.13 cpu=AMD64,AARCH64
                   type=UEFI Virtio blk version=22.4.13 cpu=AMD64,AARCH64
                   type=UEFI version=14.34.12 cpu=AMD64,AARCH64
                   type=PXE version=3.7.400 cpu=AMD64
Description:        UID                               GuidNumber
Base GUID:          946dae0300f5aa8e                       38
Base MAC:           946daef5aa8e                       38
Image VSD:          N/A
Device VSD:         N/A
PSID:               MT_00000000884
Security Attributes: secure-fw

```

Run the following commands to configure the BF3 NIC:

```

# Setting BF3 port to Ethernet mode (not Infiniband)
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set LINK_TYPE_P1=2
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set LINK_TYPE_P2=2

$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set INTERNAL_CPU_MODEL=1
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set INTERNAL_CPU_PAGE_
↳SUPPLIER=EXT_HOST_PF
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set INTERNAL_CPU_ESWITCH_
↳MANAGER=EXT_HOST_PF
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set INTERNAL_CPU_IB_VPORT0=EXT_
↳HOST_PF
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set INTERNAL_CPU_OFFLOAD_
↳ENGINE=DISABLED

$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set CQE_COMPRESSION=1
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set PROG_PARSE_GRAPH=1
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set ACCURATE_TX_SCHEDULER=1
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set FLEX_PARSER_PROFILE_ENABLE=4
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set REAL_TIME_CLOCK_ENABLE=1

$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set EXP_ROM_VIRTIO_NET_PXE_
↳ENABLE=0
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set EXP_ROM_VIRTIO_NET_UEFI_ARM_
↳ENABLE=0
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set EXP_ROM_VIRTIO_NET_UEFI_x86_
↳ENABLE=0
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set EXP_ROM_VIRTIO_BLK_UEFI_ARM_
↳ENABLE=0
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set EXP_ROM_VIRTIO_BLK_UEFI_x86_
↳ENABLE=0

# NOTE: Requires a full power cycle from host with cold boot

# Verify that the NIC FW changes have been applied
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 q | grep "CQE_COMPRESSION\|PROG_PARSE_
↳GRAPH\|ACCURATE_TX_SCHEDULER\|FLEX_PARSER_PROFILE_ENABLE\|REAL_TIME_CLOCK_ENABLE\
↳\|INTERNAL_CPU_MODEL\|LINK_TYPE_P1\|LINK_TYPE_P2\|INTERNAL_CPU_PAGE_SUPPLIER\
↳\|INTERNAL_CPU_ESWITCH_MANAGER\|INTERNAL_CPU_IB_VPORT0\|INTERNAL_CPU_OFFLOAD_ENGINE"
INTERNAL_CPU_MODEL          EMBEDDED_CPU(1)
INTERNAL_CPU_PAGE_SUPPLIER  EXT_HOST_PF(1)

```

(continues on next page)

(continued from previous page)

INTERNAL_CPU_ESWITCH_MANAGER	EXT_HOST_PF(1)
INTERNAL_CPU_IB_VPORT0	EXT_HOST_PF(1)
INTERNAL_CPU_OFFLOAD_ENGINE	DISABLED(1)
FLEX_PARSER_PROFILE_ENABLE	4
PROG_PARSE_GRAPH	True(1)
ACCURATE_TX_SCHEDULER	True(1)
CQE_COMPRESSION	AGGRESSIVE(1)
REAL_TIME_CLOCK_ENABLE	True(1)
LINK_TYPE_P1	ETH(2)
LINK_TYPE_P2	ETH(2)

#### 1.4.1.17 Install ptp4l and phc2sys

Enter these commands to configure PTP4L, assuming that aerial100 NIC interface and CPU core 41 are used for PTP:

```
$ cat <<EOF | sudo tee /etc/ptp.conf
[global]
dataset_comparison          G.8275.x
G.8275.defaultDS.localPriority 128
maxStepsRemoved            255
logAnnounceInterval        -3
logSyncInterval            -4
logMinDelayReqInterval     -4
G.8275.portDS.localPriority 128
network_transport          L2
domainNumber               24
tx_timestamp_timeout       30
slaveOnly 1

clock_servo pi
step_threshold 1.0
egressLatency 28
pi_proportional_const 4.65
pi_integral_const 0.1

[aerial00]
announceReceiptTimeout 3
delay_mechanism E2E
network_transport L2
EOF

$ cat <<EOF | sudo tee /lib/systemd/system/ptp4l.service
[Unit]
Description=Precision Time Protocol (PTP) service
Documentation=man:ptp4l
After=network.target

[Service]
Restart=always
RestartSec=5s
Type=simple
ExecStartPre=ifconfig aerial00 up
ExecStartPre=ethtool --set-priv-flags aerial00 tx_port_ts on
ExecStartPre=ethtool -A aerial00 rx off tx off
```

(continues on next page)

(continued from previous page)

```
ExecStartPre=ifconfig aerial01 up
ExecStartPre=ethtool --set-priv-flags aerial01 tx_port_ts on
ExecStartPre=ethtool -A aerial01 rx off tx off
ExecStart=taskset -c 41 /usr/sbin/ptp4l -f /etc/ptp.conf
```

```
[Install]
WantedBy=multi-user.target
EOF
```

```
$ sudo systemctl daemon-reload
$ sudo systemctl restart ptp4l.service
$ sudo systemctl enable ptp4l.service
```

One server becomes the master clock, as shown below:

```
$ sudo systemctl status ptp4l.service

ptp4l.service - Precision Time Protocol (PTP) service
  Loaded: loaded (/lib/systemd/system/ptp4l.service; enabled; vendor preset:
  →enabled)
  Active: active (running) since Fri 2024-08-30 01:25:57 UTC; 2min 16s ago
  Docs: man:ptp4l
  Main PID: 3404 (ptp4l)
  Tasks: 1 (limit: 598789)
  Memory: 2.6M
  CPU: 126ms
  CGroup: /system.slice/ptp4l.service
          3404 /usr/sbin/ptp4l -f /etc/ptp.conf

Aug 30 01:25:57 r750-01 ptp4l[3404]: [14.291] port 0: INITIALIZING to LISTENING on
  →INIT_COMPLETE
Aug 30 01:25:57 r750-01 ptp4l[3404]: [14.291] port 1: link down
Aug 30 01:25:57 r750-01 ptp4l[3404]: [14.291] port 1: LISTENING to FAULTY on FAULT_
  →DETECTED (FT_UNSPECIFIED)
Aug 30 01:25:57 r750-01 ptp4l[3404]: [14.323] selected local clock a088c2.ffffe.47be40
  →as best master
Aug 30 01:25:57 r750-01 ptp4l[3404]: [14.323] port 1: assuming the grand master role
Aug 30 01:26:56 r750-01 ptp4l[3404]: [73.338] port 1: link up
Aug 30 01:26:56 r750-01 ptp4l[3404]: [73.368] port 1: FAULTY to LISTENING on INIT_
  →COMPLETE
Aug 30 01:26:57 r750-01 ptp4l[3404]: [73.860] port 1: LISTENING to MASTER on ANNOUNCE_
  →RECEIPT_TIMEOUT_EXPIRES
Aug 30 01:26:57 r750-01 ptp4l[3404]: [73.860] selected local clock a088c2.ffffe.47be40
  →as best master
Aug 30 01:26:57 r750-01 ptp4l[3404]: [73.860] port 1: assuming the grand master role
```

The other becomes the secondary, follower clock, as shown below:

```
$ sudo systemctl status ptp4l.service

ptp4l.service - Precision Time Protocol (PTP) service
  Loaded: loaded (/lib/systemd/system/ptp4l.service; enabled; vendor preset:
  →enabled)
  Active: active (running) since Fri 2024-08-30 01:29:33 UTC; 47s ago
  Docs: man:ptp4l
  Process: 1509 ExecStartPre=ifconfig aerial00 up (code=exited, status=0/SUCCESS)
  Process: 3069 ExecStartPre=ethtool --set-priv-flags aerial00 tx_port_ts on
```

(continues on next page)

(continued from previous page)

```

↪(code=exited, status=0/SUCCESS)
  Process: 3755 ExecStartPre=ethtool -A aerial00 rx off tx off (code=exited,
↪status=0/SUCCESS)
  Process: 3822 ExecStartPre=ifconfig aerial01 up (code=exited, status=0/SUCCESS)
  Process: 3827 ExecStartPre=ethtool --set-priv-flags aerial01 tx_port_ts on
↪(code=exited, status=0/SUCCESS)
  Process: 3862 ExecStartPre=ethtool -A aerial01 rx off tx off (code=exited,
↪status=0/SUCCESS)
Main PID: 3870 (ptp4l)
Tasks: 1 (limit: 73247)
Memory: 9.2M
CPU: 183ms
CGroup: /system.slice/ptp4l.service
        3870 /usr/sbin/ptp4l -f /etc/ptp.conf

Aug 30 01:30:12 aerial-mgx-cg1-01 ptp4l[3870]: [107.479] rms    3 max    6 freq
↪+9551 +/- 12 delay -94 +/- 0
Aug 30 01:30:13 aerial-mgx-cg1-01 ptp4l[3870]: [108.479] rms    3 max    6 freq
↪+9556 +/- 10 delay -94 +/- 0
Aug 30 01:30:14 aerial-mgx-cg1-01 ptp4l[3870]: [109.479] rms    3 max    4 freq
↪+9552 +/- 13 delay -94 +/- 0
Aug 30 01:30:15 aerial-mgx-cg1-01 ptp4l[3870]: [110.479] rms    3 max    6 freq
↪+9556 +/- 12 delay -94 +/- 1
Aug 30 01:30:16 aerial-mgx-cg1-01 ptp4l[3870]: [111.479] rms    3 max    7 freq
↪+9558 +/- 14 delay -94 +/- 0
Aug 30 01:30:17 aerial-mgx-cg1-01 ptp4l[3870]: [112.479] rms    4 max    7 freq
↪+9567 +/- 12 delay -94 +/- 0
Aug 30 01:30:18 aerial-mgx-cg1-01 ptp4l[3870]: [113.479] rms    3 max    5 freq
↪+9569 +/- 7 delay -94 +/- 0
Aug 30 01:30:19 aerial-mgx-cg1-01 ptp4l[3870]: [114.479] rms    3 max    6 freq
↪+9574 +/- 8 delay -94 +/- 1
Aug 30 01:30:20 aerial-mgx-cg1-01 ptp4l[3870]: [115.479] rms    3 max    5 freq
↪+9577 +/- 9 delay -94 +/- 0
Aug 30 01:30:21 aerial-mgx-cg1-01 ptp4l[3870]: [116.479] rms    4 max    7 freq
↪+9583 +/- 12 delay -94 +/- 0

```

Enter the commands to turn off NTP:

```

$ sudo timedatectl set-ntp false
$ timedatectl
    Local time: Fri 2024-08-30 01:30:36 UTC
    Universal time: Fri 2024-08-30 01:30:36 UTC
    RTC time: Fri 2024-08-30 01:30:36
    Time zone: Etc/UTC (UTC, +0000)
System clock synchronized: no
    NTP service: inactive
    RTC in local TZ: no

```

Run PHC2SYS as service:

PHC2SYS is used to synchronize the system clock to the PTP hardware clock (PHC) on the NIC.

Specify the network interface used for PTP and system clock as the slave clock.

```

# If more than one instance is already running, kill the existing
# PHC2SYS sessions.

```

(continues on next page)

(continued from previous page)

```
# Command used can be found in /lib/systemd/system/phc2sys.service
# Update the ExecStart line to the following
$ cat <<EOF | sudo tee /lib/systemd/system/phc2sys.service
[Unit]
Description=Synchronize system clock or PTP hardware clock (PHC)
Documentation=man:phc2sys
Requires=ptp41.service
After=ptp41.service

[Service]
Restart=always
RestartSec=5s
Type=simple
# Gives ptp41 a chance to stabilize
ExecStartPre=sleep 2
ExecStart=/bin/sh -c "taskset -c 41 /usr/sbin/phc2sys -s /dev/ptp\$(ethtool -
↪T aerial00 | grep PTP | awk '{print \$4}') -c CLOCK_REALTIME -n 24 -O 0 -R
↪256 -u 256"

[Install]
WantedBy=multi-user.target
EOF
```

After the PHC2SYS config file is changed, run the following:

```
$ sudo systemctl daemon-reload
$ sudo systemctl restart phc2sys.service

# Set to start automatically on reboot
$ sudo systemctl enable phc2sys.service

# check that the service is active and has converged to a low rms value (<30) and that
↪the correct NIC has been selected (aerial00):
$ sudo systemctl status phc2sys.service

phc2sys.service - Synchronize system clock or PTP hardware clock (PHC)
Loaded: loaded (/lib/systemd/system/phc2sys.service; enabled; vendor preset:
↪enabled)
Active: active (running) since Fri 2024-08-30 01:31:35 UTC; 18min ago
Docs: man:phc2sys
Process: 3871 ExecStartPre=sleep 2 (code=exited, status=0/SUCCESS)
Main PID: 4006 (sh)
Tasks: 2 (limit: 73247)
Memory: 6.0M
CPU: 3.628s
CGroup: /system.slice/phc2sys.service
         4006 /bin/sh -c "taskset -c 41 /usr/sbin/phc2sys -s /dev/ptp\$(ethtool -
↪T aerial00 | grep PTP | awk '{print \$4}') -c CLOCK_REALTIME -n 24 -O 0 -R 256 -u
↪256"
         4012 /usr/sbin/phc2sys -s /dev/ptp2 -c CLOCK_REALTIME -n 24 -O 0 -R 256
↪-u 256

Aug 30 01:48:09 aerial-mgx-c1-01 phc2sys[4012]: [1184.489] CLOCK_REALTIME rms 8
↪max 22 freq +5522 +/- 47 delay 480 +/- 0
Aug 30 01:48:10 aerial-mgx-c1-01 phc2sys[4012]: [1185.505] CLOCK_REALTIME rms 7
↪max 19 freq +5542 +/- 30 delay 480 +/- 2
Aug 30 01:48:11 aerial-mgx-c1-01 phc2sys[4012]: [1186.521] CLOCK_REALTIME rms 7
```

(continues on next page)



(continued from previous page)

```

↪max 19 freq +5530 +/- 36 delay 480 +/- 0
Aug 30 01:48:12 aerial-mgx-c1-01 phc2sys[4012]: [1187.537] CLOCK_REALTIME rms 7
↪max 19 freq +5534 +/- 43 delay 480 +/- 2
Aug 30 01:48:13 aerial-mgx-c1-01 phc2sys[4012]: [1188.553] CLOCK_REALTIME rms 9
↪max 22 freq +5557 +/- 64 delay 480 +/- 0
Aug 30 01:48:14 aerial-mgx-c1-01 phc2sys[4012]: [1189.569] CLOCK_REALTIME rms 9
↪max 23 freq +5516 +/- 52 delay 480 +/- 0
Aug 30 01:48:15 aerial-mgx-c1-01 phc2sys[4012]: [1190.586] CLOCK_REALTIME rms 7
↪max 19 freq +5538 +/- 32 delay 480 +/- 0
Aug 30 01:48:16 aerial-mgx-c1-01 phc2sys[4012]: [1191.602] CLOCK_REALTIME rms 7
↪max 19 freq +5534 +/- 27 delay 480 +/- 0
Aug 30 01:48:17 aerial-mgx-c1-01 phc2sys[4012]: [1192.618] CLOCK_REALTIME rms 8
↪max 18 freq +5538 +/- 42 delay 480 +/- 0
Aug 30 01:48:18 aerial-mgx-c1-01 phc2sys[4012]: [1193.634] CLOCK_REALTIME rms 8
↪max 20 freq +5547 +/- 47 delay 480 +/- 0

```

Verify that the system clock is synchronized:

```

$ timedatectl
    Local time: Fri 2024-08-30 01:48:25 UTC
    Universal time: Fri 2024-08-30 01:48:25 UTC
    RTC time: Fri 2024-08-30 01:48:25
    Time zone: Etc/UTC (UTC, +0000)
System clock synchronized: yes
NTP service: inactive
RTC in local TZ: no

```

#### 1.4.1.18 Setup the Boot Configuration Service

Create the directory `/usr/local/bin` and create the `/usr/local/bin/nvidia.sh` file to run the commands with every reboot.

##### Note

The command for “`nvidia-smi lgc`” expects just one GPU device (`-i 0`). This needs to be modified if the system uses more than one GPU. The mode must be set to 1 for the GH200 so that it can utilize the max clock rate, otherwise it is limited to 1830MHz with the default `mode=0`.

```

$ cat <<"EOF" | sudo tee /usr/local/bin/nvidia.sh
#!/bin/bash

mst start

nvidia-smi -i 0 -lgc $(nvidia-smi -i 0 --query-supported-clocks=graphics --format=csv,
↪noheader,nounits | sort -h | tail -n 1) --mode=1
nvidia-smi -mig 0

echo -1 > /proc/sys/kernel/sched_rt_runtime_us
EOF

```

Create a system service file to be loaded after network interfaces are up.

```
$ cat <<EOF | sudo tee /lib/systemd/system/nvidia.service
[Unit]
After=network.target

[Service]
ExecStart=/usr/local/bin/nvidia.sh

[Install]
WantedBy=default.target
EOF
```

Create a system service file for nvidia-persistenced to be run at startup.

#### Note

This file was created following the sample from `/usr/share/doc/NVIDIA_GLX-1.0/samples/nvidia-persistenced-init.tar.bz2`

```
$ cat <<EOF | sudo tee /lib/systemd/system/nvidia-persistenced.service
[Unit]
Description=NVIDIA Persistence Daemon
Wants=syslog.target

[Service]
Type=forking
ExecStart=/usr/bin/nvidia-persistenced
ExecStopPost=/bin/rm -rf /var/run/nvidia-persistenced

[Install]
WantedBy=multi-user.target
EOF
```

Then set the file permissions, reload the systemd daemon, enable the service, restart the service when installing the first time, and check status

```
$ sudo chmod 744 /usr/local/bin/nvidia.sh
$ sudo chmod 664 /lib/systemd/system/nvidia.service
$ sudo chmod 664 /lib/systemd/system/nvidia-persistenced.service
$ sudo systemctl daemon-reload
$ sudo systemctl enable nvidia-persistenced.service
$ sudo systemctl enable nvidia.service
$ sudo systemctl restart nvidia.service
$ sudo systemctl restart nvidia-persistenced.service
$ sudo systemctl status nvidia.service
$ sudo systemctl status nvidia-persistenced.service
```

The output of the last command should look like this:

```
$ sudo systemctl status nvidia.service
nvidia.service
  Loaded: loaded (/lib/systemd/system/nvidia.service; enabled; vendor preset:
  →enabled)
  Active: inactive (dead) since Fri 2024-06-07 20:11:55 UTC; 2s ago
  Process: 3300619 ExecStart=/usr/local/bin/nvidia.sh (code=exited, status=0/
  →SUCCESS)
```

(continues on next page)

(continued from previous page)

```

Main PID: 3300619 (code=exited, status=0/SUCCESS)
CPU: 1.091s

Jun 07 20:11:54 server nvidia.sh[3300620]: Loading MST PCI module - Success
Jun 07 20:11:54 server nvidia.sh[3300620]: [warn] mst_pciconf is already loaded,
↳ skipping
Jun 07 20:11:54 server nvidia.sh[3300620]: Create devices
Jun 07 20:11:55 server nvidia.sh[3300620]: Unloading MST PCI module (unused) - Success
Jun 07 20:11:55 server nvidia.sh[3302599]: GPU clocks set to "(gpuClkMin 1980,
↳ gpuClkMax 1980)" for GPU 00000009:01:00.0
Jun 07 20:11:55 server nvidia.sh[3302599]: All done.
Jun 07 20:11:55 server nvidia.sh[3302600]: Disabled MIG Mode for GPU 00000009:01:00.0
Jun 07 20:11:55 server nvidia.sh[3302600]: All done.
Jun 07 20:11:55 server systemd[1]: nvidia.service: Deactivated successfully.
Jun 07 20:11:55 server systemd[1]: nvidia.service: Consumed 1.091s CPU time.

$ sudo systemctl status nvidia-persistenced.service
nvidia-persistenced.service - NVIDIA Persistence Daemon
Loaded: loaded (/lib/systemd/system/nvidia-persistenced.service; enabled; vendor
↳ preset: enabled)
Active: active (running) since Wed 2024-06-05 21:42:17 UTC; 1 day 22h ago
Main PID: 1858 (nvidia-persiste)
Tasks: 1 (limit: 146899)
Memory: 36.5M
CPU: 2.353s
CGroup: /system.slice/nvidia-persistenced.service
        1858 /usr/bin/nvidia-persistenced

Jun 05 21:42:15 server systemd[1]: Starting NVIDIA Persistence Daemon...
Jun 05 21:42:15 server nvidia-persistenced[1858]: Started (1858)
Jun 05 21:42:17 server systemd[1]: Started NVIDIA Persistence Daemon.

```

#### 1.4.1.19 Running Aerial on Grace Hopper

The default MGX CG1 configs within the Aerial source are:

- ▶ cuPHY-CP/cuphycontroller/config/cuphycontroller\_F08\_CG1.yaml
- ▶ cuPHY-CP/cuphycontroller/config/l2\_adapter\_config\_F08\_CG1.yaml

Pass **F08\_CG1** to the cuphycontroller\_scf executable to select them.

## 1.4.2. Installing Tools on Dell R750

This chapter describes how to install the required kernel, driver, and tools on the host. This is a one-time installation and can be skipped if the system has been configured already.

- ▶ In the following sequence of steps, the target host is [Dell PowerEdge R750](#).
- ▶ Depending on the release, tools that are installed in this section may need to be upgraded in the [Installing and Upgrading Aerial cuBB](#) section.
- ▶ After everything is installed and updated, refer to the *cuBB Quick Start Guide* on how to use Aerial cuBB.

### 1.4.2.1 Dell PowerEdge R750 Server Configuration

1. Dual Intel Xeon Gold 6336Y CPU @ 2.4G, 24C/48T (185W)
2. 512GB RDIMM, 3200MT/s
3. 1.92TB, Enterprise NVMe
4. Riser Config 2, Full Length, 4x16, 2x8 slots (PCIe gen 4)
5. Dual, Hot-Plug Power Supply Redundant (1+1), 1400W or 2400W
6. GPU Enablement

### 1.4.2.2 BF3 NIC Installation

R750 supports PCIe 4.0 x16 at slot 2,3,6,7 and x8 at slot 4,5. Follow the table below to install BF3 NIC and ensure the PCIe/GPU power cable is connected properly. These are the GPU installation instructions from [Dell R750 Installation Manual](#).

**NOTE:** Only use `SIG_PWR_3` connector on the motherboard for PCIe/GPU power.

NIC	Slot	PCIe/GPU Power	NUMA
BF3	7 (Riser 4)	SIG_PWR_3	1

### 1.4.2.3 Configure BIOS Settings

During the first boot, change the BIOS settings in the following order. The same settings can be changed via BMC: **Configuration** > **BIOS Settings**.

**Integrated Devices:** Enable Memory Mapped I/O above 4GB and change Memory Mapped I/O Base to `12TB`.

### Integrated Devices

	Current Value
User Accessible USB Ports	All Ports On
iDRAC Direct USB Port	On
Embedded NIC1 and NIC2	Enabled
I/OAT DMA Engine	Disabled
Embedded Video Controller	Enabled
I/O Snoop HoldOff Response	2K Cycles
Current State of Embedded Video Controller	Enabled
SR-IOV Global Enable	Disabled
OS Watchdog Timer	Disabled
Empty Slot Unhide	Disabled
Memory Mapped I/O above 4GB	Enabled
Memory Mapped I/O Base	12TB

**System Profile Settings:** Change System Profile to *Performance* and Workload Profile to *Low Latency Optimized Profile*.

### System Profile Settings

	Current Value
System Profile	Performance
CPU Power Management	Maximum Performance
Memory Frequency	Maximum Performance
Turbo Boost	Enabled
C1E	Disabled
C States	Disabled
Memory Patrol Scrub	Standard
Memory Refresh Rate	1x
Uncore Frequency	Maximum
Energy Efficient Policy	Performance
Monitor/Mwait	Enabled
Workload Profile	Low Latency Optimized Profile
CPU Interconnect Bus Link Power Management	Disabled
PCI ASPM L1 Link Power Management	Disabled

**Processor Settings:** Aerial CUDA-Accelerated RAN supports both *HyperThreaded mode (experimental)* or *non-HyperThreaded mode (default)* but make sure the kernel command line and the CPU core affinity in the cuPHYController YAML match the BIOS settings.

To enable HyperThreading, enable the Logical Processor. To disable HyperThreading, disable the Logical Processor.

### Processor Settings

	Current Value
Logical Processor	Disabled
CPU Interconnect Speed	Maximum data rate
Virtualization Technology	Enabled
Directory Mode	Enabled
Adjacent Cache Line Prefetch	Enabled
Hardware Prefetcher	Enabled
DCU Streamer Prefetcher	Enabled
DCU IP Prefetcher	Enabled
Sub NUMA Cluster	Disabled
MADT Core Enumeration	Round Robin

Save the BIOS settings, then reboot the system.

#### 1.4.2.4 Install Ubuntu 22.04 Server

After installing Ubuntu 22.04 Server, verify the following:

- ▶ System time is correct to avoid apt update error. If not, see [How to fix system time](#).
- ▶ LVM volume uses the whole disk space. If not, see [How to resize LVM volume](#).
- ▶ GPU and NIC are detected by the OS:

Use the following commands to determine whether the NIC is detected by the OS:

```
$ lspci | grep -i mellanox
ca:00.0 Ethernet controller: Mellanox Technologies MT43244 BlueField-3 integrated
↪ConnectX-7 network controller (rev 01)
ca:00.1 Ethernet controller: Mellanox Technologies MT43244 BlueField-3 integrated
↪ConnectX-7 network controller (rev 01)
ca:00.2 DMA controller: Mellanox Technologies MT43244 BlueField-3 SoC Management
↪Interface (rev 01)
```

#### 1.4.2.5 Disable Auto Upgrade

Edit the `/etc/apt/apt.conf.d/20auto-upgrades` system file, and change the “1” to “0” for both lines. This prevents the installed version of the low latency kernel from being accidentally changed with a subsequent software upgrade.

```
$ sudo nano /etc/apt/apt.conf.d/20auto-upgrades
APT::Periodic::Update-Package-Lists "0";
APT::Periodic::Unattended-Upgrade "0";
```

Disable the `fwupd-refresh` timer to prevent `fwupdmg` from automatically checking for any updates.

```
$ sudo systemctl mask fwupd-refresh.timer
```

### 1.4.2.6 Install the Low-Latency Kernel

If the low latency kernel is not installed, you must remove the old kernels and keep only the latest generic kernel. Enter the following command to list the installed kernels:

```
$ dpkg --get-architecture | grep -i 'linux-image' | awk '/ii/{ print $2}'

# To remove old kernel
$ sudo apt-get purge linux-image-<old kernel version>
$ sudo apt-get autoremove
```

Install the low-latency kernel with the specific version listed in the release manifest.

```
$ sudo apt-get update
$ sudo apt-get install -y linux-image-5.15.0-1042-nvidia-lowlatency
```

Update the GRUB to change the default boot kernel:

```
# Update grub to change the default boot kernel
$ sudo sed -i 's/^GRUB_DEFAULT=.*GRUB_DEFAULT="Advanced options for Ubuntu>Ubuntu,
↪with Linux 5.15.0-1042-nvidia-lowlatency"/' /etc/default/grub
```

### 1.4.2.7 Configure Linux Kernel Command-line

To set kernel command-line parameters, edit the GRUB\_CMDLINE\_LINUX\_DEFAULT parameter in the GRUB file /etc/default/grub and append/update the parameters described below. The following kernel parameters are optimized for Xeon Gold 6336Y CPU and 512GB memory.

To automatically append the GRUB file with these changes, enter this command:

```
# When HyperThread is disabled (default)
$ sudo sed -i 's/^GRUB_CMDLINE_LINUX_DEFAULT="[^\"]*/& pci=realloc=off default_
↪hugepagesz=1G hugepagesz=1G hugepages=16 tsc=reliable clocksource=tsc intel_idle.
↪max_cstate=0 mce=ignore_ce processor.max_cstate=0 intel_pstate=disable audit=0
↪idle=poll rcu_nocb_poll nosoftlockup iommu=off irqaffinity=0-3 isolcpus=managed_irq,
↪domain,4-47 nohz_full=4-47 rcu_nocbs=4-47 noht numa_balancing=disable/' /etc/
↪default/grub

# When HyperThread is enabled (experimental)
$ sudo sed -i 's/^GRUB_CMDLINE_LINUX_DEFAULT="[^\"]*/& pci=realloc=off default_
↪hugepagesz=1G hugepagesz=1G hugepages=16 tsc=reliable clocksource=tsc intel_idle.
↪max_cstate=0 mce=ignore_ce processor.max_cstate=0 intel_pstate=disable audit=0
↪idle=poll rcu_nocb_poll nosoftlockup iommu=off irqaffinity=0-3 isolcpus=managed_irq,
↪domain,4-95 nohz_full=4-95 rcu_nocbs=4-95 numa_balancing=disable/' /etc/default/grub
```

The CPU-cores-related parameters must be adjusted depending on the number of CPU cores on the system. In the example above, the “4-47” value represents CPU core numbers 4 to 47; you may need to adjust this parameter depending on the HW configuration. By default, only one DPDK thread is used. The isolated CPUs are used by the entire cuBB software stack. Use the `nproc --all` command to see how many cores are available. Do not use core numbers that are beyond the number of available cores.

#### Warning

These instructions are specific to Ubuntu 22.04 with a 5.15 low-latency kernel provided by Canonical. Make sure the kernel commands provided here are suitable for your OS and kernel versions and revise these settings to match your system if necessary.

### 1.4.2.8 Apply the Changes and Reboot to Load the Kernel

```
$ sudo update-grub
$ sudo reboot
```

After rebooting, enter the following command to verify that the system has booted into the low-latency kernel:

```
$ uname -r
5.15.0-1042-nvidia-lowlatency
```

Enter this command to verify that the kernel command-line parameters are configured properly:

```
$ cat /proc/cmdline
BOOT_IMAGE=/vmlinuz-5.15.0-1042-nvidia-lowlatency root=/dev/mapper/ubuntu--vg-ubuntu--
↪lv ro pci=realloc=off default_hugepagesz=1G hugepagesz=1G hugepages=16 tsc=reliable
↪clocksource=tsc intel_idle.max_cstate=0 mce=ignore_ce processor.max_cstate=0 intel_
↪pstate=disable audit=0 idle=poll rcu_nocb_poll nosoftlockup iommu=off irqaffinity=0-
↪3 isolcpus=managed_irq,domain,4-47 nohz_full=4-47 rcu_nocbs=4-47 noht numa_
↪balancing=disable
```

Enter this command to verify if hugepages are enabled:

```
$ grep -i huge /proc/meminfo
AnonHugePages:          0 kB
ShmemHugePages:         0 kB
FileHugePages:          0 kB
HugePages_Total:       16
HugePages_Free:         16
HugePages_Rsvd:         0
HugePages_Surp:         0
Hugepagesize:          1048576 kB
Hugetlb:                16777216 kB
```

### 1.4.2.9 Disabling Nouveau

Enter this command to disable nouveau:

```
$ cat <<EOF | sudo tee /etc/modprobe.d/blacklist-nouveau.conf
blacklist nouveau
options nouveau modeset=0
EOF
```

Regenerate the kernel initramfs and reboot the system:

```
$ sudo update-initramfs -u
$ sudo reboot
```



### 1.4.2.10 Install Dependency Packages

Enter these commands to install prerequisite packages:

```
$ sudo apt-get update
$ sudo apt-get install -y build-essential linux-headers-$(uname -r) dkms unzip
↪ linuxptp pv
```

### 1.4.2.11 Install RSHIM and Mellanox Firmware Tools on the Host

#### Note

Aerial has been using Mellanox inbox driver instead of MOFED since the 23-4 release. MOFED must be removed if it is installed on the system.

Check if there is an existing MOFED installed on the host system.

```
$ ofed_info -s
MLNX_OFED_LINUX-23.07-0.5.0.0:
```

Uninstall MOFED if it is present.

```
$ sudo /usr/sbin/ofed_uninstall.sh
```

Enter the following commands to install rshim driver.

```
# Install rshim
$ wget https://developer.nvidia.com/downloads/networking/secure/doca-sdk/DOCA_2.7/
↪ doca-host_2.7.0-209000-24.04-ubuntu2204_amd64.deb
$ sudo dpkg -i doca-host_2.7.0-209000-24.04-ubuntu2204_amd64.deb
$ sudo apt-get update
$ sudo apt install rshim
```

Enter the following commands to install Mellanox firmware tools.

```
# Install Mellanox Firmware Tools
$ export MFT_VERSION=4.28.0-92
$ wget https://www.mellanox.com/downloads/MFT/mft-$MFT_VERSION-x86_64-deb.tgz
$ tar xvf mft-$MFT_VERSION-x86_64-deb.tgz
$ sudo mft-$MFT_VERSION-x86_64-deb/install.sh

# Verify the install Mellanox firmware tool version
$ sudo mst version
mst, mft 4.28.0-92, built on Apr 25 2024, 15:22:58. Git SHA Hash: N/A

$ sudo mst start

# check NIC PCIe bus addresses and network interface names
$ sudo mst status -v

# Here is the result of GPU#1 on slot 7
MST modules:
-----
MST PCI module is not loaded
```

(continues on next page)

(continued from previous page)

```

MST PCI configuration module loaded
PCI devices:
-----
DEVICE_TYPE          MST          PCI          RDMA          NET
-----
↪                   NUMA
BlueField3(rev:1)    /dev/mst/mt41692_pciconf0.1  ca:00.1  mlx5_1  net-
↪aerial01           1
BlueField3(rev:1)    /dev/mst/mt41692_pciconf0    ca:00.0  mlx5_0  net-
↪aerial00           1

```

Enter these commands to check the link status of port 0:

```

# Here is an example if port 0 is connected to another server via a 200GbE DAC cable.

$ sudo mlxlink -d /dev/mst/mt41692_pciconf0

Operational Info
-----
State                : Active
Physical state       : LinkUp
Speed                : 200G
Width                : 4x
FEC                  : Standard_RS-FEC - (544,514)
Loopback Mode        : No Loopback
Auto Negotiation     : ON

Supported Info
-----
Enabled Link Speed (Ext.) : 0x00003ff2 (200G_2X,200G_4X,100G_1X,100G_2X,100G_
↪4X,50G_1X,50G_2X,40G,25G,10G,1G)
Supported Cable Speed (Ext.) : 0x000017f2 (200G_4X,100G_2X,100G_4X,50G_1X,50G_
↪2X,40G,25G,10G,1G)

Troubleshooting Info
-----
Status Opcode         : 0
Group Opcode          : N/A
Recommendation        : No issue was observed

Tool Information
-----
Firmware Version      : 32.41.1000
amBER Version         : 3.2
MFT Version           : mft 4.28.0-92

```

### 1.4.2.12 Install Docker CE

The full official instructions for installing Docker CE can be found on the Docker website: <https://docs.docker.com/engine/install/ubuntu/#install-docker-engine>. The following instructions are one supported way of installing Docker CE:

#### Warning

To work correctly, the CUDA driver must be installed before Docker CE or nvidia-container-toolkit installation. It is recommended that you install the CUDA driver before installing Docker CE or the nvidia-container-toolkit.

```
$ sudo apt-get update
$ sudo apt-get install -y ca-certificates curl gnupg
$ sudo install -m 0755 -d /etc/apt/keyrings
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /
→etc/apt/keyrings/docker.gpg
$ sudo chmod a+r /etc/apt/keyrings/docker.gpg
$ echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg]
→https://download.docker.com/linux/ubuntu \
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
$ sudo apt-get update
$ sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin
→docker-compose-plugin
$ sudo docker run --rm hello-world
```

### 1.4.2.13 Update BF3 BFB Image and NIC Firmware

#### Note

- ▶ The following instructions are for BF3 NIC (**OPN: 900-9D3B6-00CV-A; PSID: MT\_000000884**) specifically.
- ▶ There is no need to switch to DPU mode if using the BFB image below.
- ▶ This BFB image will update the NIC firmware automatically.

```
# Enable MST
$ sudo mst start
$ sudo mst status

MST modules:
-----
MST PCI module is not loaded
MST PCI configuration module loaded

MST devices:
-----
/dev/mst/mt41692_pciconf0          - PCI configuration cycles access.
domain:bus:dev.fn=0000:ca:00.0 addr.reg=88 data.
```

(continues on next page)

(continued from previous page)

```

↪ reg=92 cr_bar.gw_offset=-1
                                Chip revision is: 01

# Download the BF3 BFB image
$ wget https://content.mellanox.com/BlueField/BFBs/Ubuntu22.04/bf-bundle-2.7.0-33_24.
↪ 04_ubuntu-22.04_prod.bfb
# Here is the command to flash BFB image. NOTE: If there are multiple BF3 NICs, repeat
↪ the same command with rshim<0..N-1>. N is the number of BF3 NICs.
$ sudo bfb-install -r rshim0 -b bf-bundle-2.7.0-33_24.04_ubuntu-22.04_prod.bfb

Pushing bfb
1.41GiB 0:01:24 [17.1MiB/s] [
↪                                     <=>]
Collecting BlueField booting status. Press Ctrl+C to stop...
INFO[PSC]: PSC BL1 START
INFO[BL2]: start
INFO[BL2]: boot mode (rshim)
INFO[BL2]: VDDQ adjustment complete
INFO[BL2]: VDDQ: 1120 mV
INFO[BL2]: DDR POST passed
INFO[BL2]: UEFI loaded
INFO[BL31]: start
INFO[BL31]: lifecycle GA Secured
INFO[BL31]: VDD: 851 mV
ERR[BL31]: MB timeout
INFO[BL31]: runtime
INFO[UEFI]: eMMC init
INFO[UEFI]: eMMC probed
INFO[UEFI]: UPVS valid
INFO[UEFI]: PMI: updates started
INFO[UEFI]: PMI: total updates: 1
INFO[UEFI]: PMI: updates completed, status 0
INFO[UEFI]: PCIe enum start
INFO[UEFI]: PCIe enum end
INFO[UEFI]: UEFI Secure Boot (enabled)
INFO[UEFI]: Redfish enabled
INFO[BL31]: Partial NIC
INFO[BL31]: power capping disabled
INFO[UEFI]: exit Boot Service
INFO[MISC]: Ubuntu installation started
INFO[MISC]: Installing OS image
INFO[MISC]: Ubuntu installation completed
WARN[MISC]: Skipping BMC components upgrade.
INFO[MISC]: Updating NIC firmware...
INFO[MISC]: NIC firmware update done
INFO[MISC]: Installation finished

# Wait 10 minutes to ensure the card initializes properly after the BFB installation
$ sleep 600

# NOTE: Requires a full power cycle from host with cold boot

# Verify NIC FW version after reboot
$ sudo mst start
$ sudo flint -d /dev/mst/mt41692_pciconf0 q

```

(continues on next page)

(continued from previous page)

```

Image type:          FS4
FW Version:          32.41.1000
FW Release Date:    28.4.2024
Product Version:    32.41.1000
Rom Info:           type=UEFI Virtio net version=21.4.13 cpu=AMD64,AARCH64
                   type=UEFI Virtio blk version=22.4.13 cpu=AMD64,AARCH64
                   type=UEFI version=14.34.12 cpu=AMD64,AARCH64
                   type=PXE version=3.7.400 cpu=AMD64
Description:        UID                               GuidNumber
Base GUID:          946dae0300f5aa8e                    38
Base MAC:           946daef5aa8e                      38
Image VSD:          N/A
Device VSD:         N/A
PSID:               MT_0000000884
Security Attributes: secure-fw

```

Run the following commands to configure the BF3 NIC:

```

# Setting BF3 port to Ethernet mode (not Infiniband)
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set LINK_TYPE_P1=2
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set LINK_TYPE_P2=2

$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set INTERNAL_CPU_MODEL=1
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set INTERNAL_CPU_PAGE_
↪SUPPLIER=EXT_HOST_PF
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set INTERNAL_CPU_ESWITCH_
↪MANAGER=EXT_HOST_PF
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set INTERNAL_CPU_IB_VPORT0=EXT_
↪HOST_PF
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set INTERNAL_CPU_OFFLOAD_
↪ENGINE=DISABLED

$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set CQE_COMPRESSION=1
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set PROG_PARSE_GRAPH=1
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set ACCURATE_TX_SCHEDULER=1
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set FLEX_PARSER_PROFILE_ENABLE=4
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set REAL_TIME_CLOCK_ENABLE=1

$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set EXP_ROM_VIRTIO_NET_PXE_
↪ENABLE=0
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set EXP_ROM_VIRTIO_NET_UEFI_ARM_
↪ENABLE=0
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set EXP_ROM_VIRTIO_NET_UEFI_x86_
↪ENABLE=0
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set EXP_ROM_VIRTIO_BLK_UEFI_ARM_
↪ENABLE=0
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 --yes set EXP_ROM_VIRTIO_BLK_UEFI_x86_
↪ENABLE=0

# NOTE: Requires a full power cycle from host with cold boot

# Verify that the NIC FW changes have been applied
$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 q | grep "CQE_COMPRESSION\|PROG_PARSE_
↪GRAPH\|ACCURATE_TX_SCHEDULER\|FLEX_PARSER_PROFILE_ENABLE\|REAL_TIME_CLOCK_ENABLE\
↪INTERNAL_CPU_MODEL\|LINK_TYPE_P1\|LINK_TYPE_P2\|INTERNAL_CPU_PAGE_SUPPLIER\
↪INTERNAL_CPU_ESWITCH_MANAGER\|INTERNAL_CPU_IB_VPORT0\|INTERNAL_CPU_OFFLOAD_ENGINE"
INTERNAL_CPU_MODEL                               EMBEDDED_CPU(1)

```

(continues on next page)

(continued from previous page)

INTERNAL_CPU_PAGE_SUPPLIER	EXT_HOST_PF(1)
INTERNAL_CPU_ESWITCH_MANAGER	EXT_HOST_PF(1)
INTERNAL_CPU_IB_VPORT0	EXT_HOST_PF(1)
INTERNAL_CPU_OFFLOAD_ENGINE	DISABLED(1)
FLEX_PARSER_PROFILE_ENABLE	4
PROG_PARSE_GRAPH	True(1)
ACCURATE_TX_SCHEDULER	True(1)
CQE_COMPRESSION	AGGRESSIVE(1)
REAL_TIME_CLOCK_ENABLE	True(1)
LINK_TYPE_P1	ETH(2)
LINK_TYPE_P2	ETH(2)

#### 1.4.2.14 Set Persistent NIC Interface Name

Configure the network link files so that the NIC interfaces always come up with the same name. Run `lshw -c network -businfo` to find the current interface name on the target bus address then run `ip link` to find the corresponding MAC address by the interface name. After identifying the MAC address, create files at `/etc/systemd/network/NN-persistent-net.link` with the following information:

```
[Match]
MACAddress={{item.mac}}

[Link]
Name={{item.name}}
```

The following network link files set the converged accelerator port#0 to aerial00 and port#1 to aerial01:

```
$ sudo nano /etc/systemd/network/11-persistent-net.link

# Update the MAC address to match the converged accelerator port 0 MAC address
[Match]
MACAddress=48:b0:2d:xx:xx:xx

[Link]
Name=aerial00

$ sudo nano /etc/systemd/network/12-persistent-net.link

# Update the MAC address to match the converged accelerator port 1 MAC address
[Match]
MACAddress=48:b0:2d:yy:yy:yy

[Link]
Name=aerial01
```

Reboot the system after creating these files.

### 1.4.2.15 Install ptp4l and phc2sys

Enter these commands to configure PTP4L assuming the `aerial100` NIC interface and CPU core **41** are used for PTP:

```
$ cat <<EOF | sudo tee /etc/ptp.conf
[global]
dataset_comparison          G.8275.x
G.8275.defaultDS.localPriority 128
maxStepsRemoved            255
logAnnounceInterval        -3
logSyncInterval            -4
logMinDelayReqInterval     -4
G.8275.portDS.localPriority 128
network_transport          L2
domainNumber               24
tx_timestamp_timeout       30
# When used as an RU and PTP master, set slaveOnly to 0
slaveOnly 0

clock_servo pi
step_threshold 1.0
egressLatency 28
pi_proportional_const 4.65
pi_integral_const 0.1

[aerial00]
announceReceiptTimeout 3
delay_mechanism E2E
network_transport L2
EOF

cat <<EOF | sudo tee /lib/systemd/system/ptp4l.service
[Unit]
Description=Precision Time Protocol (PTP) service
Documentation=man:ptp4l
After=network.target

[Service]
Restart=always
RestartSec=5s
Type=simple
ExecStartPre=ifconfig aerial00 up
ExecStartPre=ethtool --set-priv-flags aerial00 tx_port_ts on
ExecStartPre=ethtool -A aerial00 rx off tx off
ExecStartPre=ifconfig aerial01 up
ExecStartPre=ethtool --set-priv-flags aerial01 tx_port_ts on
ExecStartPre=ethtool -A aerial01 rx off tx off
ExecStart=taskset -c 41 /usr/sbin/ptp4l -f /etc/ptp.conf

[Install]
WantedBy=multi-user.target
EOF

$ sudo systemctl daemon-reload
$ sudo systemctl restart ptp4l.service
$ sudo systemctl enable ptp4l.service
```

One server becomes the master clock, as shown below:

```
$ sudo systemctl status ptp41.service

• ptp41.service - Precision Time Protocol (PTP) service
  Loaded: loaded (/lib/systemd/system/ptp41.service; enabled; vendor preset:
  ↪enabled)
  Active: active (running) since Tue 2023-08-08 19:37:56 UTC; 2 weeks 3 days ago
  Docs: man:ptp41
  Main PID: 1120 (ptp41)
  Tasks: 1 (limit: 94533)
  Memory: 460.0K
  CPU: 9min 8.089s
  CGroup: /system.slice/ptp41.service
          1120 /usr/sbin/ptp41 -f /etc/ptp.conf

Aug 09 18:12:35 aerial-devkit taskset[1120]: ptp41[81287.043]: selected local clock
  ↪b8cef6.ffff.d333be as best master
Aug 09 18:12:35 aerial-devkit taskset[1120]: ptp41[81287.043]: port 1: assuming the
  ↪grand master role
Aug 11 20:44:51 aerial-devkit taskset[1120]: ptp41[263223.379]: timed out while
  ↪polling for tx timestamp
Aug 11 20:44:51 aerial-devkit taskset[1120]: ptp41[263223.379]: increasing tx_
  ↪timestamp_timeout may correct this issue, but it is likely caused by a driver bug
Aug 11 20:44:51 aerial-devkit taskset[1120]: ptp41[263223.379]: port 1: send sync
  ↪failed
Aug 11 20:44:51 aerial-devkit taskset[1120]: ptp41[263223.379]: port 1: MASTER to
  ↪FAULTY on FAULT_DETECTED (FT_UNSPECIFIED)
Aug 11 20:45:07 aerial-devkit taskset[1120]: ptp41[263239.522]: port 1: FAULTY to
  ↪LISTENING on INIT_COMPLETE
Aug 11 20:45:08 aerial-devkit taskset[1120]: ptp41[263239.963]: port 1: LISTENING to
  ↪MASTER on ANNOUNCE_RECEIPT_TIMEOUT_EXPIRES
Aug 11 20:45:08 aerial-devkit taskset[1120]: ptp41[263239.963]: selected local clock
  ↪b8cef6.ffff.d333be as best master
Aug 11 20:45:08 aerial-devkit taskset[1120]: ptp41[263239.963]: port 1: assuming the
  ↪grand master role
```

The other becomes the secondary, follower clock, as shown below:

```
$ sudo systemctl status ptp41.service

• ptp41.service - Precision Time Protocol (PTP) service
  Loaded: loaded (/lib/systemd/system/ptp41.service; enabled; vendor preset:
  ↪enabled)
  Active: active (running) since Tue 2023-08-22 16:25:41 UTC; 3 days ago
  Docs: man:ptp41
  Main PID: 3251 (ptp41)
  Tasks: 1 (limit: 598810)
  Memory: 472.0K
  CPU: 2min 48.984s
  CGroup: /system.slice/ptp41.service
          3251 /usr/sbin/ptp41 -f /etc/ptp.conf

Aug 25 19:58:34 aerial-r750 taskset[3251]: ptp41[272004.187]: rms      8 max   15 freq -
  ↪14495 +/-  9 delay   11 +/-  0
Aug 25 19:58:35 aerial-r750 taskset[3251]: ptp41[272005.187]: rms      6 max   12 freq -
  ↪14480 +/-  7 delay   11 +/-  1
Aug 25 19:58:36 aerial-r750 taskset[3251]: ptp41[272006.187]: rms      8 max   12 freq -
  (continues on next page)
```



(continued from previous page)

```

↪14465 +/- 5 delay 10 +/- 0
Aug 25 19:58:37 aerial-r750 taskset[3251]: ptp4l[272007.187]: rms 11 max 18 freq -
↪14495 +/- 10 delay 11 +/- 1
Aug 25 19:58:38 aerial-r750 taskset[3251]: ptp4l[272008.187]: rms 12 max 21 freq -
↪14515 +/- 7 delay 12 +/- 1
Aug 25 19:58:39 aerial-r750 taskset[3251]: ptp4l[272009.187]: rms 7 max 12 freq -
↪14488 +/- 7 delay 12 +/- 1
Aug 25 19:58:40 aerial-r750 taskset[3251]: ptp4l[272010.187]: rms 7 max 12 freq -
↪14479 +/- 7 delay 11 +/- 1
Aug 25 19:58:41 aerial-r750 taskset[3251]: ptp4l[272011.187]: rms 10 max 20 freq -
↪14503 +/- 11 delay 11 +/- 1
Aug 25 19:58:42 aerial-r750 taskset[3251]: ptp4l[272012.188]: rms 10 max 20 freq -
↪14520 +/- 7 delay 13 +/- 1
Aug 25 19:58:43 aerial-r750 taskset[3251]: ptp4l[272013.188]: rms 2 max 7 freq -
↪14510 +/- 4 delay 12 +/- 1

```

Enter the commands to turn off NTP:

```

$ sudo timedatectl set-ntp false
$ timedatectl
Local time: Thu 2022-02-03 22:30:58 UTC
        Universal time: Thu 2022-02-03 22:30:58 UTC
                RTC time: Thu 2022-02-03 22:30:58
                Time zone: Etc/UTC (UTC, +0000)
System clock synchronized: no
        NTP service: inactive
        RTC in local TZ: no

```

Run PHC2SYS as service:

PHC2SYS is used to synchronize the system clock to the PTP hardware clock (PHC) on the NIC.

Specify the network interface used for PTP and system clock as the slave clock.

```

# If more than one instance is already running, kill the existing
# PHC2SYS sessions.

# Command used can be found in /lib/systemd/system/phc2sys.service
# Update the ExecStart line to the following
$ cat <<EOF | sudo tee /lib/systemd/system/phc2sys.service
[Unit]
Description=Synchronize system clock or PTP hardware clock (PHC)
Documentation=man:phc2sys
After=ntpdate.service
Requires=ptp4l.service
After=ptp4l.service

[Service]
Restart=always
RestartSec=5s
Type=simple
# Gives ptp4l a chance to stabilize
ExecStartPre=sleep 2
ExecStart=/bin/sh -c "taskset -c 41 /usr/sbin/phc2sys -s /dev/ptp$(ethtool -
↪T aerial00| grep PTP | awk '{print $4}') -c CLOCK_REALTIME -n 24 -0 0 -R
↪256 -u 256"

```

(continues on next page)

(continued from previous page)

```
[Install]
WantedBy=multi-user.target
EOF
```

After the PHC2SYS config file is changed, run the following:

```
$ sudo systemctl daemon-reload
$ sudo systemctl restart phc2sys.service

# Set to start automatically on reboot
$ sudo systemctl enable phc2sys.service

# check that the service is active and has converged to a low rms value (<30) and that
↪the correct NIC has been selected (aerial00):
$ sudo systemctl status phc2sys.service
phc2sys.service - Synchronize system clock or PTP hardware clock (PHC)
   Loaded: loaded (/lib/systemd/system/phc2sys.service; enabled; vendor preset:
   ↪enabled)
   Active: active (running) since Fri 2023-02-17 17:02:35 UTC; 7s ago
     Docs: man:phc2sys
    Main PID: 2225556 (phc2sys)
       Tasks: 1 (limit: 598864)
      Memory: 372.0K
    CGroup: /system.slice/phc2sys.service
           2225556 /usr/sbin/phc2sys -a -r -n 24 -R 256 -u 256

Feb 17 17:02:35 aerial-devkit phc2sys[2225556]: [1992363.445] reconfiguring after
↪port state change
Feb 17 17:02:35 aerial-devkit phc2sys[2225556]: [1992363.445] selecting CLOCK_
↪REALTIME for synchronization
Feb 17 17:02:35 aerial-devkit phc2sys[2225556]: [1992363.445] selecting aerial00 as
↪the master clock
Feb 17 17:02:36 aerial-devkit phc2sys[2225556]: [1992364.457] CLOCK_REALTIME rms 15
↪max 37 freq -19885 +/- 116 delay 1944 +/- 6
Feb 17 17:02:37 aerial-devkit phc2sys[2225556]: [1992365.473] CLOCK_REALTIME rms 16
↪max 42 freq -19951 +/- 103 delay 1944 +/- 7
Feb 17 17:02:38 aerial-devkit phc2sys[2225556]: [1992366.490] CLOCK_REALTIME rms 13
↪max 31 freq -19909 +/- 81 delay 1944 +/- 6
Feb 17 17:02:39 aerial-devkit phc2sys[2225556]: [1992367.506] CLOCK_REALTIME rms 9
↪max 27 freq -19918 +/- 40 delay 1945 +/- 6
Feb 17 17:02:40 aerial-devkit phc2sys[2225556]: [1992368.522] CLOCK_REALTIME rms 8
↪max 24 freq -19925 +/- 11 delay 1945 +/- 9
Feb 17 17:02:41 aerial-devkit phc2sys[2225556]: [1992369.538] CLOCK_REALTIME rms 9
↪max 23 freq -19915 +/- 36 delay 1943 +/- 8
```

Verify that the system clock is synchronized:

```
$ timedatectl
Local time: Thu 2022-02-03 22:30:58 UTC
   Universal time: Thu 2022-02-03 22:30:58 UTC
         RTC time: Thu 2022-02-03 22:30:58
       Time zone: Etc/UTC (UTC, +0000)
System clock synchronized: yes
   NTP service: inactive
      RTC in local TZ: no
```

### 1.4.2.16 Setup the Boot Configuration Service

Create the directory `/usr/local/bin` and create the `/usr/local/bin/nvidia.sh` file to run the commands with every reboot. The command for “`nvidia-smi lgc`” expects just one GPU device (`-i 0`). This needs to be modified, if the system uses more than one GPU.

```
$ cat <<"EOF" | sudo tee /usr/local/bin/nvidia.sh
#!/bin/bash

mst start

nvidia-smi -i 0 -lgc $(nvidia-smi -i 0 --query-supported-clocks=graphics --format=csv,
↪noheader,nounits | sort -h | tail -n 1)
nvidia-smi -mig 0

echo -1 > /proc/sys/kernel/sched_rt_runtime_us
EOF
```

Create a system service file to be loaded after network interfaces are up.

```
$ cat <<EOF | sudo tee /lib/systemd/system/nvidia.service
[Unit]
After=network.target

[Service]
ExecStart=/usr/local/bin/nvidia.sh

[Install]
WantedBy=default.target
EOF
```

Create a system service file for `nvidia-persistenced` to be run at startup.

#### Note

This file was created following the sample from `/usr/share/doc/NVIDIA_GLX-1.0/samples/nvidia-persistenced-init.tar.bz2`

```
cat <<EOF | sudo tee /lib/systemd/system/nvidia-persistenced.service
[Unit]
Description=NVIDIA Persistence Daemon
Wants=syslog.target

[Service]
Type=forking
ExecStart=/usr/bin/nvidia-persistenced
ExecStopPost=/bin/rm -rf /var/run/nvidia-persistenced

[Install]
WantedBy=multi-user.target
EOF
```

Then set the file permissions, reload the `systemd` daemon, enable the service, restart the service when installing the first time, and check status

```

sudo chmod 744 /usr/local/bin/nvidia.sh
sudo chmod 664 /lib/systemd/system/nvidia.service
sudo chmod 664 /lib/systemd/system/nvidia-persistenced.service
sudo systemctl daemon-reload
sudo systemctl enable nvidia-persistenced.service
sudo systemctl enable nvidia.service
sudo systemctl restart nvidia.service
sudo systemctl restart nvidia-persistenced.service
sudo systemctl status nvidia.service
sudo systemctl status nvidia-persistenced.service

```

The output of the last command should look like this:

```

aerial@server:~$ sudo systemctl status nvidia.service
nvidia.service
  Loaded: loaded (/lib/systemd/system/nvidia.service; enabled; vendor preset:
↳ enabled)
  Active: inactive (dead) since Fri 2024-06-07 20:26:06 UTC; 2s ago
  Process: 251860 ExecStart=/usr/local/bin/nvidia.sh (code=exited, status=0/SUCCESS)
  Main PID: 251860 (code=exited, status=0/SUCCESS)
  CPU: 788ms

Jun 07 20:26:05 server nvidia.sh[251862]: Starting MST (Mellanox Software Tools)
↳ driver set
Jun 07 20:26:05 server nvidia.sh[251862]: Loading MST PCI module - Success
Jun 07 20:26:05 server nvidia.sh[251862]: [warn] mst_pciconf is already loaded,
↳ skipping
Jun 07 20:26:05 server nvidia.sh[251862]: Create devices
Jun 07 20:26:06 server nvidia.sh[251862]: Unloading MST PCI module (unused) - Success
Jun 07 20:26:06 server nvidia.sh[252732]: GPU clocks set to "(gpuClkMin 1410,
↳ gpuClkMax 1410)" for GPU 00000000:CF:00.0
Jun 07 20:26:06 server nvidia.sh[252732]: All done.
Jun 07 20:26:06 server nvidia.sh[252733]: Disabled MIG Mode for GPU 00000000:CF:00.0
Jun 07 20:26:06 server nvidia.sh[252733]: All done.
Jun 07 20:26:06 server systemd[1]: nvidia.service: Deactivated successfully.

aerial@server:~$ sudo systemctl status nvidia-persistenced.service
nvidia-persistenced.service - NVIDIA Persistence Daemon
  Loaded: loaded (/lib/systemd/system/nvidia-persistenced.service; enabled; vendor
↳ preset: enabled)
  Active: active (running) since Fri 2024-06-07 20:25:57 UTC; 3s ago
  Process: 251836 ExecStart=/usr/bin/nvidia-persistenced (code=exited, status=0/
↳ SUCCESS)
  Main PID: 251837 (nvidia-persiste)
  Tasks: 1 (limit: 598792)
  Memory: 672.0K
  CPU: 9ms
  CGroup: /system.slice/nvidia-persistenced.service
          251837 /usr/bin/nvidia-persistenced

Jun 07 20:25:57 server systemd[1]: Starting NVIDIA Persistence Daemon...
Jun 07 20:25:57 server nvidia-persistenced[251837]: Started (251837)
Jun 07 20:25:57 server systemd[1]: Started NVIDIA Persistence Daemon.

```

## 1.4.3. Installing and Upgrading Aerial cuBB

You must update the dependent software components to the specific version listed in the [Release Manifest](#).

If you are upgrading a Grace Hopper MGX system, follow [Installing Tools on Grace Hopper](#) to upgrade the dependent SW first.

If you are upgrading a Dell R750 system with A100X converged accelerator, follow [Installing Tools on Dell R750](#) to upgrade the dependent SW first.

### 1.4.3.1 Removing the Old Aerial cuBB Container

This step is optional. To remove the old cuBB container, enter the following commands:

```
$ sudo docker stop <cuBB container name>
$ sudo docker rm <cuBB container name>
```

### 1.4.3.2 Installing the New Aerial cuBB Container

The cuBB container is available on the [NVIDIA GPU Cloud \(NGC\)](#). Follow the instructions on that page to pull the container and to run the container.

#### Note

If you receive the cuBB container image via nvonline, run “docker load < cuBB container image file” to load the image. Then use the same docker run command detailed on the NGC page to launch it.

## 1.4.4. Aerial System Scripts

### 1.4.4.1 System Configuration Validation Script

Included in the release package is a script that checks and displays key system configuration settings that are important for running the Aerial cuBB SDK.

```
$ pip3 install psutil
$ cd $cuBB_SDK/cuPHY/util/cuBB_system_checks
$ sudo -E python3 ./cuBB_system_checks.py
```

The output of `cuBB_system_checks.py` may differ slightly between bare-metal and container versions of the environment. The script helps to retrieve the software-component versions and hardware configuration. Refer to the [Release Manifest](#) in the *cuBB Release Notes* to ensure the correct software-component versions are installed. Below is an example output on a bare-metal platform:

```
# To get the system or ptp info, the command has to run on the host.
$ sudo -E python3 ./cuBB_system_checks.py --sys
```

```
-----General-----
Hostname                : smc-gh-01
```

(continues on next page)

(continued from previous page)

```

IP address           : 192.168.1.100
Linux distro        : "Ubuntu 22.04.3 LTS"
Linux kernel version : 6.5.0-1019-nvidia
-----System-----
Manufacturer        : Supermicro
Product Name        : ARS-111GL-NHR
Base Board Manufacturer : Supermicro
Base Board Product Name : G1SMH-G
Chassis Manufacturer  : Supermicro
Chassis Type         : Other
Chassis Height        : 1 U
Processor            : Grace A02
Max Speed             : Unknown
Current Speed         : 3402 MHz

```

```
$ sudo -E python3 ./cuBB_system_checks.py
```

```

-----General-----
Hostname            : smc-gh-01
IP address          : 192.168.1.100
Linux distro        : "Ubuntu 22.04.3 LTS"
Linux kernel version : 6.5.0-1019-nvidia
-----Kernel Command Line-----
Audit subsystem     : audit=0
Clock source        : N/A
HugePage count      : hugepages=32
HugePage size       : hugepagesz=512M
CPU idle time management : idle=poll
Max Intel C-state   : N/A
Intel IOMMU         : N/A
IOMMU               : N/A
Isolated CPUs       : N/A
Corrected errors    : N/A
Adaptive-tick CPUs  : nohz_full=4-47
Soft-lockup detector disable : nosoftlockup
Max processor C-state : processor.max_cstate=0
RCU callback polling : rcu_nocb_poll
No-RCU-callback CPUs : rcu_nocbs=4-47
TSC stability checks : tsc=reliable
-----CPU-----
CPU cores           : 72
Thread(s) per CPU core : 1
CPU MHz             : N/A
CPU sockets         : 1
-----Environment variables-----
CUDA_DEVICE_MAX_CONNECTIONS : N/A
cuBB_SDK            : N/A
-----Memory-----
HugePage count      : 32
Free HugePages      : 31
HugePage size       : 524288 kB
Shared memory size  : 240G
-----Nvidia GPUs-----
GPU driver version   : 555.42.02
CUDA version         : 12.5

```

(continues on next page)

(continued from previous page)

```

GPU0
  GPU product name       : NVIDIA GH200 480GB
  GPU persistence mode   : Enabled
  Current GPU temperature : 36 C
  GPU clock frequency    : 1980 MHz
  Max GPU clock frequency : 1980 MHz
  GPU PCIe bus id       : 00000009:01:00.0
-----GPUdirect topology-----
      GPU0    NIC0    NIC1    NIC2    NIC3    CPU Affinity    NUMA Affinity    GPU
↪NUMA ID
GPU0    X      SYS     SYS     SYS     SYS     0-71    0                1
NIC0    SYS     X      PIX     SYS     SYS
NIC1    SYS     PIX     X      SYS     SYS
NIC2    SYS     SYS     SYS     X      PIX
NIC3    SYS     SYS     SYS     PIX     X

```

## Legend:

```

X      = Self
SYS    = Connection traversing PCIe as well as the SMP interconnect between NUMA
↪nodes (e.g., QPI/UPI)
NODE   = Connection traversing PCIe as well as the interconnect between PCIe Host
↪Bridges within a NUMA node
PHB    = Connection traversing PCIe as well as a PCIe Host Bridge (typically the CPU)
PXB    = Connection traversing multiple PCIe bridges (without traversing the PCIe
↪Host Bridge)
PIX    = Connection traversing at most a single PCIe bridge
NV#    = Connection traversing a bonded set of # NVLinks

```

## NIC Legend:

```

NIC0: mlx5_0
NIC1: mlx5_1
NIC2: mlx5_2
NIC3: mlx5_3

```

## -----Mellanox NICs-----

```

NIC0
  NIC product name       : BlueField3
  NIC part number        : 900-9D3B6-00CV-A_Ax
  NIC PCIe bus id       : /dev/mst/mt41692_pciconf1
  NIC FW version         : 32.41.1000
  FLEX_PARSER_PROFILE_ENABLE : 4
  PROG_PARSE_GRAPH       : True(1)
  ACCURATE_TX_SCHEDULER  : True(1)
  CQE_COMPRESSION        : AGGRESSIVE(1)
  REAL_TIME_CLOCK_ENABLE : True(1)
NIC1
  NIC product name       : BlueField3
  NIC part number        : 900-9D3B6-00CV-A_Ax
  NIC PCIe bus id       : /dev/mst/mt41692_pciconf0
  NIC FW version         : 32.41.1000
  FLEX_PARSER_PROFILE_ENABLE : 4
  PROG_PARSE_GRAPH       : True(1)
  ACCURATE_TX_SCHEDULER  : True(1)

```

(continues on next page)

(continued from previous page)

```

CQE_COMPRESSION           : AGGRESSIVE(1)
REAL_TIME_CLOCK_ENABLE    : True(1)
-----Mellanox NIC Interfaces-----
Interface0
  Name                     : aerial00
  Network adapter          : mlx5_0
  PCIe bus id              : 0000:01:00.0
  Ethernet address         : 94:6d:ae:c7:62:00
  Operstate                 : up
  MTU                      : 1514
  RX flow control          : off
  TX flow control          : off
  PTP hardware clock       : 0
  QoS Priority trust state : pcp
  PCIe MRRS                : 4096 bytes
Interface1
  Name                     : aerial01
  Network adapter          : mlx5_1
  PCIe bus id              : 0000:01:00.1
  Ethernet address         : 94:6d:ae:c7:62:01
  Operstate                 : up
  MTU                      : 1500
  RX flow control          : off
  TX flow control          : off
  PTP hardware clock       : 1
  QoS Priority trust state : pcp
  PCIe MRRS                : 512 bytes
Interface2
  Name                     : aerial02
  Network adapter          : mlx5_2
  PCIe bus id              : 0002:01:00.0
  Ethernet address         : 94:6d:ae:c7:6b:80
  Operstate                 : down
  MTU                      : 1500
  RX flow control          : on
  TX flow control          : on
  PTP hardware clock       : 2
  QoS Priority trust state : pcp
  PCIe MRRS                : 512 bytes
Interface3
  Name                     : aerial03
  Network adapter          : mlx5_3
  PCIe bus id              : 0002:01:00.1
  Ethernet address         : 94:6d:ae:c7:6b:81
  Operstate                 : down
  MTU                      : 1500
  RX flow control          : on
  TX flow control          : on
  PTP hardware clock       : 3
  QoS Priority trust state : pcp
  PCIe MRRS                : 512 bytes
-----Linux PTP-----
ptp4l.service - Precision Time Protocol (PTP) service
  Loaded: loaded (/lib/systemd/system/ptp4l.service; enabled; vendor preset:
  ↪enabled)
  Active: active (running) since Wed 2024-06-05 21:42:18 UTC; 6h ago

```

(continues on next page)



(continued from previous page)

```

Docs: man:ptp4l
Process: 4267 ExecStartPre=ethtool --set-priv-flags aerial01 tx_port_ts on
↳(code=exited, status=0/SUCCESS)
Process: 4386 ExecStartPre=ethtool -A aerial01 rx off tx off (code=exited,
↳status=0/SUCCESS)
Main PID: 4508 (ptp4l)
Tasks: 1 (limit: 146899)
Memory: 8.2M
CPU: 17.936s
CGroup: /system.slice/ptp4l.service
        4508 /usr/sbin/ptp4l -f /etc/ptp.conf

Jun 06 03:45:21 smc-gh-01 ptp4l[4508]: [21807.308] rms      2 max      5 freq -1855 +/-
↳11 delay -96 +/-  0
Jun 06 03:45:22 smc-gh-01 ptp4l[4508]: [21808.308] rms      3 max      6 freq -1848 +/-
↳10 delay -96 +/-  0
Jun 06 03:45:23 smc-gh-01 ptp4l[4508]: [21809.308] rms      2 max      4 freq -1851 +/-
↳ 9 delay -96 +/-  1
Jun 06 03:45:24 smc-gh-01 ptp4l[4508]: [21810.308] rms      2 max      4 freq -1851 +/-
↳ 8 delay -97 +/-  1
Jun 06 03:45:25 smc-gh-01 ptp4l[4508]: [21811.308] rms      3 max      6 freq -1864 +/-
↳13 delay -96 +/-  0
Jun 06 03:45:26 smc-gh-01 ptp4l[4508]: [21812.308] rms      2 max      5 freq -1860 +/-
↳10 delay -96 +/-  0
Jun 06 03:45:27 smc-gh-01 ptp4l[4508]: [21813.308] rms      2 max      5 freq -1852 +/-
↳10 delay -97 +/-  0
Jun 06 03:45:28 smc-gh-01 ptp4l[4508]: [21814.308] rms      3 max      5 freq -1858 +/-
↳12 delay -96 +/-  1
Jun 06 03:45:29 smc-gh-01 ptp4l[4508]: [21815.308] rms      3 max      5 freq -1849 +/-
↳10 delay -97 +/-  0
Jun 06 03:45:30 smc-gh-01 ptp4l[4508]: [21816.308] rms      3 max      5 freq -1850 +/-
↳13 delay -97 +/-  0

phc2sys.service - Synchronize system clock or PTP hardware clock (PHC)
Loaded: loaded (/lib/systemd/system/phc2sys.service; enabled; vendor preset:
↳enabled)
Active: active (running) since Wed 2024-06-05 21:42:20 UTC; 6h ago
Docs: man:phc2sys
Process: 4529 ExecStartPre=sleep 2 (code=exited, status=0/SUCCESS)
Main PID: 4873 (sh)
Tasks: 2 (limit: 146899)
Memory: 2.1M
CPU: 1min 14.399s
CGroup: /system.slice/phc2sys.service
        4873 /bin/sh -c "taskset -c 47 /usr/sbin/phc2sys -s /dev/ptp\$(ethtool -
↳T aerial00 | grep PTP | awk '{print \$4}') -c CLOCK_REALTIME -n 24 -O 0 -R 256 -u
↳256"
        4878 /usr/sbin/phc2sys -s /dev/ptp0 -c CLOCK_REALTIME -n 24 -O 0 -R 256
↳-u 256

Jun 06 03:45:20 smc-gh-01 phc2sys[4878]: [21806.453] CLOCK_REALTIME rms      8 max      20
↳freq +8730 +/- 44 delay 512 +/-  0
Jun 06 03:45:21 smc-gh-01 phc2sys[4878]: [21807.469] CLOCK_REALTIME rms      8 max      20
↳freq +8758 +/- 36 delay 512 +/-  0
Jun 06 03:45:22 smc-gh-01 phc2sys[4878]: [21808.486] CLOCK_REALTIME rms      7 max      19
↳freq +8740 +/- 44 delay 512 +/-  3

```

(continues on next page)

(continued from previous page)

```

Jun 06 03:45:23 smc-gh-01 phc2sys[4878]: [21809.502] CLOCK_REALTIME rms 7 max 18
↪freq +8749 +/- 35 delay 512 +/- 0
Jun 06 03:45:24 smc-gh-01 phc2sys[4878]: [21810.519] CLOCK_REALTIME rms 7 max 16
↪freq +8744 +/- 35 delay 512 +/- 0
Jun 06 03:45:25 smc-gh-01 phc2sys[4878]: [21811.535] CLOCK_REALTIME rms 8 max 21
↪freq +8722 +/- 55 delay 512 +/- 0
Jun 06 03:45:26 smc-gh-01 phc2sys[4878]: [21812.552] CLOCK_REALTIME rms 9 max 23
↪freq +8750 +/- 61 delay 512 +/- 2
Jun 06 03:45:28 smc-gh-01 phc2sys[4878]: [21813.570] CLOCK_REALTIME rms 8 max 20
↪freq +8749 +/- 49 delay 512 +/- 2
Jun 06 03:45:29 smc-gh-01 phc2sys[4878]: [21814.589] CLOCK_REALTIME rms 6 max 18
↪freq +8735 +/- 29 delay 512 +/- 2
Jun 06 03:45:30 smc-gh-01 phc2sys[4878]: [21815.608] CLOCK_REALTIME rms 7 max 18
↪freq +8762 +/- 40 delay 512 +/- 3

```

```
-----Software Packages-----
```

```

cmake                : N/A
docker               /usr/bin  : 26.1.3
gcc                  /usr/bin  : 11.4.0
git-lfs              /usr/bin  : 3.0.2
MOFED                : N/A
meson                 : N/A
ninja                 : N/A
ptp41                 /usr/sbin  : 3.1.1-3

```

```
-----Loaded Kernel Modules-----
```

```

GDRCopy              : gdrdrv
GPUDirect RDMA       : N/A
Nvidia                : nvidia

```

```
-----Non-persistent settings-----
```

```

VM swappiness         : vm.swappiness = 0
VM zone reclaim mode  : vm.zone_reclaim_mode = 0

```

```
-----Docker images-----
```

#### 1.4.4.1.1 Checking the NIC Status

To query back the Mellanox NIC firmware settings initialized with the script above, use these commands:

```

$ sudo mlxconfig -d /dev/mst/mt41692_pciconf0 q | grep "CQE_COMPRESSION\|PROG_PARSE_
↪GRAPH\
  \ | ACCURATE_TX_SCHEDULER\ | FLEX_PARSER_PROFILE_ENABLE\ | REAL_TIME_CLOCK_ENABLE\
↪ | INTERNAL_CPU_MODEL\
  \ | LINK_TYPE_P1\ | LINK_TYPE_P2\ | INTERNAL_CPU_PAGE_SUPPLIER\ | INTERNAL_CPU_ESWITCH_
↪MANAGER\
  \ | INTERNAL_CPU_IB_VPORT0\ | INTERNAL_CPU_OFFLOAD_ENGINE"

INTERNAL_CPU_MODEL          EMBEDDED_CPU(1)
INTERNAL_CPU_PAGE_SUPPLIER  EXT_HOST_PF(1)
INTERNAL_CPU_ESWITCH_MANAGER EXT_HOST_PF(1)
INTERNAL_CPU_IB_VPORT0     EXT_HOST_PF(1)
INTERNAL_CPU_OFFLOAD_ENGINE DISABLED(1)
FLEX_PARSER_PROFILE_ENABLE  4
PROG_PARSE_GRAPH           True(1)
ACCURATE_TX_SCHEDULER      True(1)

```

(continues on next page)

(continued from previous page)

CQE_COMPRESSION	AGGRESSIVE(1)
REAL_TIME_CLOCK_ENABLE	True(1)
LINK_TYPE_P1	ETH(2)
LINK_TYPE_P2	ETH(2)

To check the current status of a NIC port, use this command:

```
$ sudo mlxlink -d /dev/mst/mt41692_pciconf0
```

```
Operational Info
-----
State                               : Active
Physical state                       : LinkUp
Speed                                 : 200G
Width                                 : 4x
FEC                                   : Standard_RS-FEC - (544,514)
Loopback Mode                         : No Loopback
Auto Negotiation                     : ON

Supported Info
-----
Enabled Link Speed (Ext.)            : 0x00003ff2 (200G_2X,200G_4X,100G_1X,100G_2X,100G_
↳4X,50G_1X,50G_2X,40G,25G,10G,1G)
Supported Cable Speed (Ext.)        : 0x000017f2 (200G_4X,100G_2X,100G_4X,50G_1X,50G_
↳2X,40G,25G,10G,1G)

Troubleshooting Info
-----
Status Opcode                        : 0
Group Opcode                         : N/A
Recommendation                       : No issue was observed

Tool Information
-----
Firmware Version                    : 32.41.1000
amBER Version                        : 3.2
MFT Version                          : mft 4.28.0-92
```

Alternatively, you can use the [System Configuration Validation Script](#) to obtain a full list of configuration settings.

## 1.4.5. Troubleshooting

This page documents solutions to common issues that you might encounter.

### 1.4.5.1 Hugepages Issues

Normally the hugepages settings are updated through the `/etc/default/grub` configuration file. However, depending on the version of operating system, the settings changes may become overwritten by another configuration file: `/etc/grub`.

### 1.4.5.2 Remove Old CUDA Toolkit and Driver

If the system has an old version installed, run the following to remove the CUDA Toolkit and driver :

```
sudo apt-get --purge remove "*cublas*" "*cufft*" "*curand*" "*cusolver*" "*cusparse*"
↳ "*npp*" "*nvjpeg*" "cuda*" "nsight*" "*nvidia*"
sudo apt-get autoremove
```

### 1.4.5.3 How to Fix Apt Update Error Due to Incorrect System Time

You may see the apt update error if the system time is incorrect.

```
E: Release file for https://download.docker.com/linux/ubuntu/dists/focal/InRelease is
↳ not valid yet (invalid for another 2d 10h 51min 11s).
Updates for this repository will not be applied.
```

Run the following commands to set the date and time via NTP once (this will not enable the NTP service):

```
sudo apt-get install ntpdate
sudo ntpdate -s pool.ntp.org
```

### 1.4.5.4 How to Resize the Default LVM Volume

When installing Ubuntu 22.04 server, it partitions the whole disk but only creates a 200GB logical volume. This is what you will see on a newly installed devkit:

```
# Devkit has 1TB SSD but default lv uses only 200GB
lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
loop0 7:0 0 55.5M 1 loop /snap/core18/2246
loop1 7:1 0 55.5M 1 loop /snap/core18/2253
loop2 7:2 0 67.3M 1 loop /snap/lxd/21545
loop3 7:3 0 67.2M 1 loop /snap/lxd/21835
loop4 7:4 0 61.9M 1 loop /snap/core20/1242
loop5 7:5 0 61.9M 1 loop /snap/core20/1169
loop6 7:6 0 32.5M 1 loop /snap/snapd/13640
loop7 7:7 0 42.2M 1 loop /snap/snapd/14066
sda 8:0 0 894.3G 0 disk
  sda1 8:1 0 512M 0 part /boot/efi
  sda2 8:2 0 1G 0 part /boot
  sda3 8:3 0 892.8G 0 part
  ubuntu--vg-ubuntu--lv 253:0 0 200G 0 lvm /
```

The following commands resize the logic volume to use the entire disk, then resize the file system to use the entire logic volume.

```

# Test mode first
sudo lvresize -t -v -l +100%FREE /dev/mapper/ubuntu--vg-ubuntu--lv

# Remove -t if test mode succeeds
sudo lvresize -v -l +100%FREE /dev/mapper/ubuntu--vg-ubuntu--lv
lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
loop0 7:0 0 55.5M 1 loop /snap/core18/2246
loop1 7:1 0 55.5M 1 loop /snap/core18/2253
loop2 7:2 0 67.3M 1 loop /snap/lxd/21545
loop3 7:3 0 67.2M 1 loop /snap/lxd/21835
loop4 7:4 0 61.9M 1 loop /snap/core20/1242
loop5 7:5 0 61.9M 1 loop /snap/core20/1169
loop6 7:6 0 32.5M 1 loop /snap/snapd/13640
loop7 7:7 0 42.2M 1 loop /snap/snapd/14066
sda 8:0 0 894.3G 0 disk
  sda1 8:1 0 512M 0 part /boot/efi
  sda2 8:2 0 1G 0 part /boot
  sda3 8:3 0 892.8G 0 part
ubuntu--vg-ubuntu--lv 253:0 0 892.8G 0 lvm /

# Resize file system
sudo resize2fs -p /dev/mapper/ubuntu--vg-ubuntu--lv
df -h -T

Filesystem                                Type      Size  Used Avail Use% Mounted on
udev                                       devtmpfs  39G   0    39G   0% /dev
tmpfs                                       tmpfs     9.4G  2.0M  9.4G   1% /run
/dev/mapper/ubuntu--vg-ubuntu--lv        ext4      878G  77G  764G  10% /
tmpfs                                       tmpfs     47G   0    47G   0% /dev/shm
tmpfs                                       tmpfs     5.0M   0    5.0M   0% /run/lock
tmpfs                                       tmpfs     47G   0    47G   0% /sys/fs/cgroup
/dev/sda2                                  ext4      976M  460M  450M  51% /boot
/dev/loop0                                squashfs  56M   56M   0 100% /snap/core18/2246
/dev/sda1                                  vfat     511M  5.3M  506M   2% /boot/efi
/dev/loop1                                squashfs  56M   56M   0 100% /snap/core18/2253
/dev/loop5                                squashfs  62M   62M   0 100% /snap/core20/1169
/dev/loop2                                squashfs  68M   68M   0 100% /snap/lxd/21545
/dev/loop4                                squashfs  62M   62M   0 100% /snap/core20/1242
/dev/loop6                                squashfs  33M   33M   0 100% /snap/snapd/13640
/dev/loop3                                squashfs  68M   68M   0 100% /snap/lxd/21835
/dev/loop7                                squashfs  43M   43M   0 100% /snap/snapd/14066
overlay                                    overlay   878G  77G  764G  10% /var/lib/docker/
↪overlay2/851cbfd83b022a24f61fb0f87a007c56da8065a7528f6b661bf45d3d65ccc787/merged
tmpfs                                       tmpfs     9.4G  4.0K  9.4G   1% /run/user/1000

```

### 1.4.5.5 How to Identify the NIC Interface Name and MAC Address

Use the `sudo lshw -c network |grep -i 'product\|bus info\|name\|serial'` command to find the bus address and MAC address of each NIC on the system. Here is an example:

```
$ sudo lshw -c network |grep -i 'product\|bus info\|name\|serial'
  product: I210 Gigabit Network Connection
  bus info: pci@0000:05:00.0
  logical name: eno1
  serial: 18:c0:4d:79:49:b6
  product: I210 Gigabit Network Connection
  bus info: pci@0000:06:00.0
  logical name: enp6s0
  serial: 18:c0:4d:79:49:b7
  product: MT2892 Family [ConnectX-6 Dx]
  bus info: pci@0000:b5:00.0
  logical name: ens6f0
  serial: b8:ce:f6:33:fd:ee
  product: MT2892 Family [ConnectX-6 Dx]
  bus info: pci@0000:b5:00.1
  logical name: ens6f1
  serial: b8:ce:f6:33:fd:ef
```

## 1.5. cuBB Quickstart Guide

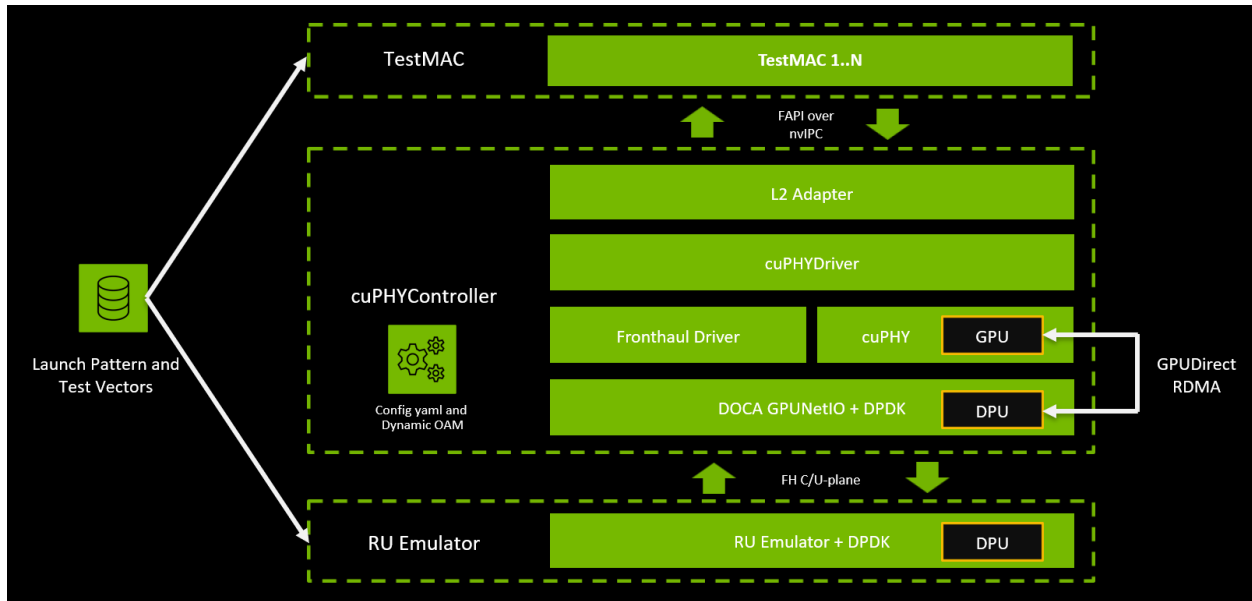
This section explains how to run the Aerial cuBB software examples.

### 1.5.1. cuBB Quickstart Overview

The diagrams below show the Aerial cuBB software and hardware components.

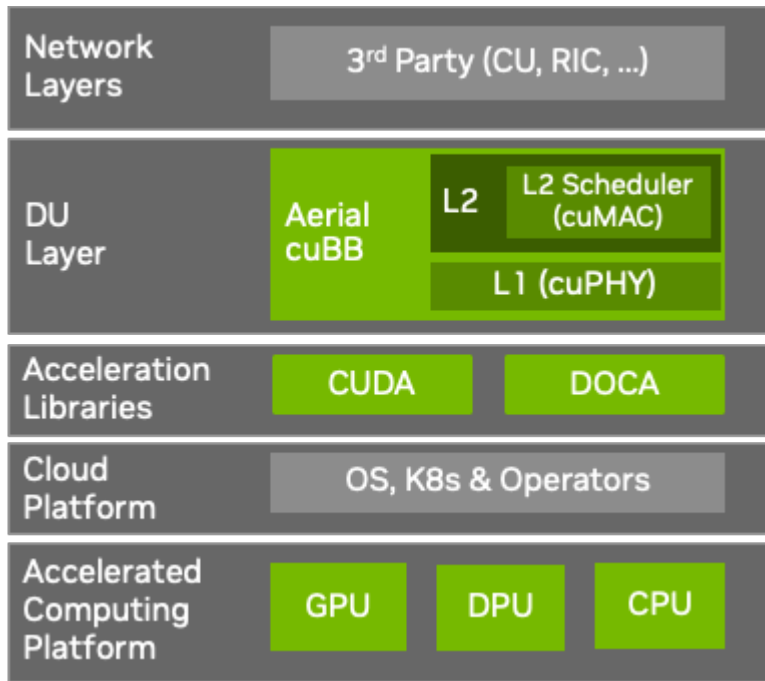
- ▶ **cuPHY** is the GPU-Accelerated 5G PHY layer software library and examples. It provides GPU-offloaded 5G signal processing.
- ▶ **DPDK** is the software library that provides network data transfer acceleration. The public version of DPDK now contains features like eCPRI flow steering and accurate TX scheduling, which Aerial uses.
- ▶ **cuPHY-CP** is the cuPHY Control-Plane software that provides the control plane interface between the layer 1 cuPHY and the upper layer stack.

Shown below is the block diagram of the cuPHY-CP. It supports multi-cell. Included with cuPHY-CP are the built-in test MAC and RU emulator modules.



The Aerial cuBB makes use of the DPDK for the network interface. It provides efficient high-speed network data connectivity to GPU processing of network data.

The diagram below shows the overall Aerial cuBB software and hardware stack layers:



## 1.5.2. Generating TV and Launch Pattern Files

Since the cuBB 22-2.2 release, the test vectors are not included in the release package. You must generate the TV files before running cuPHY examples or cuBB end-to-end test.

### Note

TV generation is **NOT** supported on ARM because Matlab Compiler SDK doesn't support it yet.

### 1.5.2.1 Using Aerial Python mcore Module

No Matlab license is required to generate TV files using the Aerial Python mcore module. The cuBB container already has aerial\_mcore installed.

To generate the test vectors required for end-to-end testing, follow these steps:

1. Run the following inside the Aerial container. It completes in less than a minute.

```
cd ${cuBB_SDK}/5GModel/aerial_mcore/examples
source ../scripts/setup.sh
../scripts/gen_e2e_ota_tvs.sh
ls -lh GPU_test_input/
cp GPU_test_input/* ${cuBB_SDK}/testVectors/
```

The following is example output from the above commands:

```
aerial@c_aerial_aerial:/opt/nvidia/cuBB/5GModel/aerial_mcore$ source ../scripts/
↪setup.sh
[Aerial Python]aerial@c_aerial_aerial:/opt/nvidia/cuBB/5GModel/aerial_mcore$ ./
↪scripts/gen_e2e_ota_tvs.sh
Finished genCuPhyChEstCoeffs
Elapsed time: 1.166473150253296 seconds
[Aerial Python]aerial@c_aerial_aerial:/opt/nvidia/cuBB/5GModel/aerial_mcore$ ls -
↪lh ../GPU_test_input/
-rw-rw-r-- 1 aerial aerial 90K Oct 17 2023 ../cuPhyChEstCoeffs.h5
```

### Note

The cuPhyChEstCoeffs.h5 file can be found in the /opt/nvidia/cuBB/testVectors directory of both the x86 and ARM containers.

2. Copy the output to the testVectors folder.

To generate all of the TV files, including files that are not necessary for E2E testing, follow these steps:

1. Run the following commands inside the Aerial container.

```
cd ${cuBB_SDK}/5GModel/aerial_mcore/examples
source ../scripts/setup.sh
export REGRESSION_MODE=1
time python3 ./example_5GModel_regression.py allChannels
echo $?
ls -a1F GPU_test_input/
du -h GPU_test_input/
```



**Note**

The TV generation may take a few hours on the devkit with the current isocpus parameter setting in the kernel command line. The host must have at least 64GB of memory and 430GB of available disk space. Hyperthreading must be enabled.

- Review the output from the above commands; an example is shown below. The “real” time takes less than one hour on a 24-core x86 host. The `echo $?` command shows the exit code of the process, which should be 0, while a non-zero exit code indicates a failure.

Channel	Compliance_Test	Error	Test_Vector	Error	Performance_Test	Fail
SSB	37	0	42	0	0	0
PDCCH	71	0	80	0	0	0
PDSCH	274	0	286	0	0	0
CSIRS	86	0	87	0	0	0
DL MIX	0	0	1049	0	0	0
PRACH	60	0	60	0	48	0
PUCCH	469	0	469	0	96	0
PUSCH	388	0	398	0	41	0
SRS	125	0	125	0	0	0
UL MIX	0	0	576	0	0	0
BFW	58	0	58	0	0	0
Total	1568	0	3230	0	185	0

Total time for runRegression is 2147 seconds  
Parallel pool using the 'local' profile is shutting down.

```
real    36m51.931s
user    585m1.704s
sys     10m28.322s
```

To generate the launch pattern for each test case using `cubb_scripts`, follow these steps:

- Run the following commands:

```
cd $cuBB_SDK
cd cubb_scripts
python3 auto_lp.py -i ../5GModel/aerial_mcore/examples/GPU_test_input -t launch_
↪pattern_nrSim.yaml
```

- Copy the launch pattern and TV files to the `testVectors` repo:

```
cd $cuBB_SDK
cp ../5GModel/aerial_mcore/examples/GPU_test_input/*h5 ./testVectors/
cp ../5GModel/aerial_mcore/examples/GPU_test_input/launch_pattern* ./testVectors/
↪multi-cell/`
```

### 1.5.2.2 Using Matlab

To generate TV files using Matlab:

1. Run the following command in Matlab:

```
cd('nr_matlab'); startup; [nTC, errCnt] = runRegression({'TestVector'}, {
↳ 'allChannels'}, 'compact', [0, 1] );
```

All the cuPHY TVs are generated and stored under nr\_matlab/GPU\_test\_input.

2. Generate the launch pattern for each test case using cubb\_scripts:

```
cd $cuBB_SDK
cd cubb_scripts
python3 auto_lp.py -i ../5GModel/nr_matlab/GPU_test_input -t launch_
↳ pattern_nrSim.yaml
```

3. Copy the launch pattern and TV files to testVectors repo.

```
cd $cuBB_SDK
cp ../5GModel/nr_matlab/GPU_test_input/TVnr_* ./testVectors/.
cp ../5GModel/nr_matlab/GPU_test_input/launch_pattern* ./testVectors/multi-cell/.
```

## 1.5.3. Running Aerial cuPHY

Aerial cuPHY provides the cuPHY library and several examples that link with the library. Here we include instructions on using MATLAB to generate TVs. Please refer to [Generating TV and Launch Pattern Files](#) for using Aerial Python mcore Module to generate TVs.

### 1.5.3.1 Building Aerial cuPHY

#### 1.5.3.1.1 Prerequisites

The following instructions assume the system configuration and Aerial cuBB installation are done. If not, see the [cuBB Install Guide](#) to complete the installation or upgrade process.

After powering on the system, use the following commands to verify that the GPU and NIC are in the correct state:

```
# Verify GPU is detected and CUDA driver version matches the release manifest.
$ nvidia-smi
```

Verify that the NIC is in the correct state on the host (this is only required to run cuBB end-to-end):

```
# Verify NIC is detected: Example CX6-DX
```

```
$ sudo lshw -c network -businfo
```

Bus info	Device	Class	Description
pci@0000:05:00.0	eno1	network	I210 Gigabit Network Connection

(continues on next page)

(continued from previous page)

```

pci@0000:06:00.0  enp6s0      network      I210 Gigabit Network Connection
pci@0000:b5:00.0  ens6f0      network      MT2892 Family [ConnectX-6 Dx]
pci@0000:b5:00.1  ens6f1      network      MT2892 Family [ConnectX-6 Dx]

# Verify the link state is right. Assuming NIC port 0 is connected.

$ sudo mlxlink -d b5:00.0

Operational Info
-----
State                : Active
Physical state       : LinkUp
Speed                : 100G
Width                : 4x
FEC                  : Standard RS-FEC - RS(528,514)
Loopback Mode        : No Loopback
Auto Negotiation     : ON

Supported Info
-----
Enabled Link Speed (Ext.) : 0x000007f2 (100G_2X,100G_4X,50G_1X,50G_2X,40G,25G,
↪10G,1G)
Supported Cable Speed (Ext.) : 0x000002f2 (100G_4X,50G_2X,40G,25G,10G,1G)

Troubleshooting Info
-----
Status Opcode         : 0
Group Opcode          : N/A
Recommendation        : No issue was observed.

```

### 1.5.3.1.2 Set Up the Host Environment

Set up the environment by following the cuBB Installation Guide for the server type you are using.

### 1.5.3.1.3 Launch the cuBB Container

Use the following command to launch the cuBB container:

```
$ sudo docker exec -it cuBB /bin/bash
```

### 1.5.3.1.4 Build Aerial cuPHY in the Container

Build cuPHY in the cuBB container using the following commands:

```

$ cd /opt/nvidia/cuBB/cuPHY
$ cmake -Bbuild -GNinja -DCMAKE_TOOLCHAIN_FILE=cmake/toolchains/native -DCMAKE_
↪INSTALL_PREFIX=./install
$ cmake --build build

```

cuPHY is, by default, built in Release mode. The option BUILD\_DOCS=ON is also enabled by default to allow the make to generate the Doxygen documentation for the cuPHY library API. To disable this op-

tion, pass `-DBUILD_DOCS=OFF` to the CMake command line. The output directory is `cuPHY/install/docs`.

To put the built cuPHY headers and libraries into an installation directory so that other applications using the cuPHY library can compile and link with cuPHY, use the commands from the current build directory:

```
$ cmake --install build
```

This creates the `include` and `lib` directories under the `cuPHY/install` directory.

### 1.5.3.2 Building and running on separate servers

When building the source code on one server, and running the binaries on another server, it might be important to use the correct toolchain for the target.

The source code directory `cuPHY/cmake/toolchains` contains toolchains for the following targets:

x86-64: `devkit`, `r750`, `x86-64`

arm: `grace-cross`, `bf3`

A new toolchain file might need to be created if using a different target.

The toolchain file defines what compiler to use, and the value of `AERIAL_ARCH_TUNE_FLAGS`

One way to make sure that the flag is correct, is to do the following:

Run the `aerial_sdk` container on the target, inside the container run the following command:

```
$ gcc -march=native -Q --help=target
```

Run the `aerial_sdk` container on the build server, inside the container run the following command:

```
$ gcc -march=<march for target> -Q --help=target
```

Make sure the outputs from both commands are the same. Create a toolchain file and use it when building `aerial_sdk`:

```
$ cmake -Bbuild -GNinja -DCMAKE_TOOLCHAIN_FILE=cmake/toolchains/my-target
```

### 1.5.3.3 Running the cuPHY Examples

This section describes how to run the Aerial cuPHY standalone example programs. They read test vector data files as input. Refer to the [Supported Test Vector Configurations](#) section of the cuPHY Release Notes to determine which test vectors to use for different configurations. Do not use old test vectors from previous cuBB releases with the example programs of this release.

### 1.5.3.3.1 Generating Test Vectors using Matlab 5GModel

Run this Matlab command:

```
cd('nr_matlab'); startup; [nTC, errCnt] = runRegression({'TestVector'}, {'allChannels
→'}, 'compact', [0, 1] );
```

All the cuPHY test vectors are generated and stored under nr\_matlab/GPU\_test\_input.

### 1.5.3.3.2 Instructions for Testing cuPHY Channels Manually

#### 1.5.3.3.2.1 PUSCH

##### Test Vectors

Match test vector name with *PUSCH\_gNB\_CUPHY\_\*.h5*

##### How to Run

- ▶ Streams mode: cuPHY/build/examples/pusch\_rx\_multi\_pipe/cuphy\_ex\_pusch\_rx\_multi\_pipe -i ~/<tv\_name>.h5
- ▶ Graphs mode: cuPHY/build/examples/pusch\_rx\_multi\_pipe/cuphy\_ex\_pusch\_rx\_multi\_pipe -i ~/<tv\_name>.h5 -m 1

##### Expected Outcome

Test 1 (CRC test KPI): All test cases must have zero CRC errors (only CRC errors, not correct ones, are reported when the channel is run).

#### 1.5.3.3.2.2 PUCCH

##### Test Vectors

Match test vector name with *PUCCH\_F\*\_gNB\_CUPHY\_\*.h5*

##### How to Run

```
PUCCH format 0/1/2/3: cuPHY/build/examples/pucch_rx_pipeline/cuphy_ex_pucch_rx_pipeline -i <tv_name>
```

##### Expected Outcome

- ▶ cuphy\_ex\_pucch\_Fx\_receiver checks if the test vector includes PFX UCI first.
- ▶ If the test-vector UCI format is not expected, it displays “No PFX UCI received”.

```
./build/examples/pucch_F0_receiver/cuphy_ex_pucch_F0_receiver -i
../GPU_test_input/TVnr_6109_PUCCH_gNB_CUPHY_s0p126.h5
***** COMPUTE-SANITIZER

No PF0 UCI received.
***** LEAK SUMMARY: 0 bytes leaked in 0 allocations
***** ERROR SUMMARY: 0 errors
```

- ▶ If the test-vector UCI format is expected, it compares UCI output.xzsd.

```

./build/examples/pucch_F0_receiver/cuphy_ex_pucch_F0_receiver -i
../GPU_test_input/TVnr_6026_PUCCH_gNB_CUPHY_s0p1.h5
===== COMPUTE-SANITIZER

comparing cuPHY F0 UCI output to reference output: 0 mismatches out of 1 UCIs
===== LEAK SUMMARY: 0 bytes leaked in 0 allocations
===== ERROR SUMMARY: 0 errors

```

### 1.5.3.3.2.3 PRACH

#### Test Vectors

Match test vector name with *PRACH\_gNB\_CUPHY\_\*.h5*

#### How to Run

```

cuPHY/build/examples/prach_receiver_multi_cell/prach_receiver_multi_cell -i
<tv_name> -r <num_iteration> -k

```

#### Expected Outcome

- ▶ *prach\_receiver\_multi\_cell* compares against the reference measurements in the test vector.
- ▶ Measured values are displayed and if they are within tolerance the message is displayed:

```

=====> Test PASS

```

### 1.5.3.3.2.4 PDSCH

#### Test Vectors

Match test vector name with *PDSCH\_gNB\_CUPHY\_\*.h5*

#### How to Run

- ▶ PDSCH in non-AAS mode, streams: *cuPHY/build/examples/pdsch\_tx/cuphy\_ex\_pdsch\_tx ~/<tv\_name>.h5 2 0 0*
- ▶ PDSCH in non-AAS mode, graphs: *cuPHY/build/examples/pdsch\_tx/cuphy\_ex\_pdsch\_tx ~/<tv\_name>.h5 2 0 1*

#### Expected Outcome

Test 1 (correctness against reference model): Channel reports correct match with reference model

### 1.5.3.3.2.5 PDCCH

#### Test Vectors

Match test vector name with *PDCCH\_gNB\_CUPHY\_\*.h5*

#### How to Run

- ▶ Streams mode: *cuPHY/build/examples/pcdch/embed\_pdcch\_tf\_signal -i ~/<tv\_name>.h5 -m 0*

- ▶ Graphs mode: `cuPHY/build/examples/pdcch/embed_pdcch_tf_signal -i ~/<tv_name>.h5 -m 1`

**Expected Outcome**

Test 1 (correctness against reference model): Test PASS

**1.5.3.3.2.6 SSB****Test Vectors**

Match test vector name with `SSB_gNB_CUPHY_*.h5`

**How to Run**

- ▶ Streams mode: `cuPHY/build/examples/ss/testSS -i ~/<tv_name>.h5 -m 0`
- ▶ Graphs mode: `cuPHY/build/examples/ss/testSS -i ~/<tv_name>.h5 -m 1`

**Expected Outcome**

Test 1 (correctness against reference model): Test PASS

**1.5.3.3.2.7 CSI-RS****Test Vectors**

Match test vector name with `CSIRS_gNB_CUPHY_*.h5`

**How to Run**

- ▶ Streams mode: `cuPHY/build/examples/csi_rs/nzp_csi_rs_test -i <tv_name> -m 0`
- ▶ Graphs mode: `cuPHY/build/examples/csi_rs/nzp_csi_rs_test -i <tv_name> -m 1`

**Expected Outcome**

Test 1 (correctness against reference model): Test PASS

**1.5.3.3.2.8 SRS****Test Vectors**

Match test vector name with `SRS_gNB_CUPHY_*.h5`

**How to Run**

- ▶ Streams mode: `cuPHY/build/examples/srs_rx_pipeline/cuphy_ex_srs_rx_pipeline -i <tv_name> -r <num_iteration> -m 0`
- ▶ Graphs mode: `cuPHY/build/examples/srs_rx_pipeline/cuphy_ex_srs_rx_pipeline -i <tv_name> -r <num_iteration> -m 1`

**Expected Outcome**

Test 1 (correctness against reference model): SRS reference check: PASSED!; Timing results are provided

### 1.5.3.3.2.9 BFC

#### Test Vectors

Match test vector name with *BFW\_gNB\_CUPHY\_\*.h5*

#### How to Run

- ▶ Streams mode: `cuPHY/build/examples/bfc/cuphy_ex_bfc -i <tv_name> -r <num_iteration> -m 0`
- ▶ Graphs mode: `cuPHY/build/examples/bfc/cuphy_ex_bfc -i <tv_name> -r <num_iteration> -m 1`
- ▶ Add `-c` to enable reference check (default disabled)

#### Expected Outcome

Test 1 (measure latency without reference check): Timing results are provided

Test 2 (correctness against reference model using `-c`): Test PASS; Timing results are provided

### 1.5.3.3.3 Instructions for LDPC Performance Test

The `ldpc_perf_collect.py` Python script from the cuPHY repository can be used to perform error rate tests for the cuPHY LDPC decoder. There are test input files defined for  $Z = [64, 128, 256, 384]$ ,  $BG = [1, 2]$ . The current tests check whether the block error rate (BLER, also sometimes referred to as Frame Error Rate or FER) is less than 0.1.

From the build directory, the following commands run the tests:

```

./util/ldpc/ldpc_perf_collect.py --mode test -i ./util/ldpc/test/ldpc_decode_BG1_
↪Z64_BLER0.1.txt -f -w 800 -P
./util/ldpc/ldpc_perf_collect.py --mode test -i ./util/ldpc/test/ldpc_decode_BG1_
↪Z128_BLER0.1.txt -f -w 800 -P
./util/ldpc/ldpc_perf_collect.py --mode test -i ./util/ldpc/test/ldpc_decode_BG1_
↪Z256_BLER0.1.txt -f -w 800 -P
./util/ldpc/ldpc_perf_collect.py --mode test -i ./util/ldpc/test/ldpc_decode_BG1_
↪Z384_BLER0.1.txt -f -w 800 -P
./util/ldpc/ldpc_perf_collect.py --mode test -i ./util/ldpc/test/ldpc_decode_BG2_
↪Z64_BLER0.1.txt -f -w 800 -P
./util/ldpc/ldpc_perf_collect.py --mode test -i ./util/ldpc/test/ldpc_decode_BG2_
↪Z128_BLER0.1.txt -f -w 800 -P
./util/ldpc/ldpc_perf_collect.py --mode test -i ./util/ldpc/test/ldpc_decode_BG2_
↪Z256_BLER0.1.txt -f -w 800 -P
./util/ldpc/ldpc_perf_collect.py --mode test -i ./util/ldpc/test/ldpc_decode_BG2_
↪Z384_BLER0.1.txt -f -w 800 -P

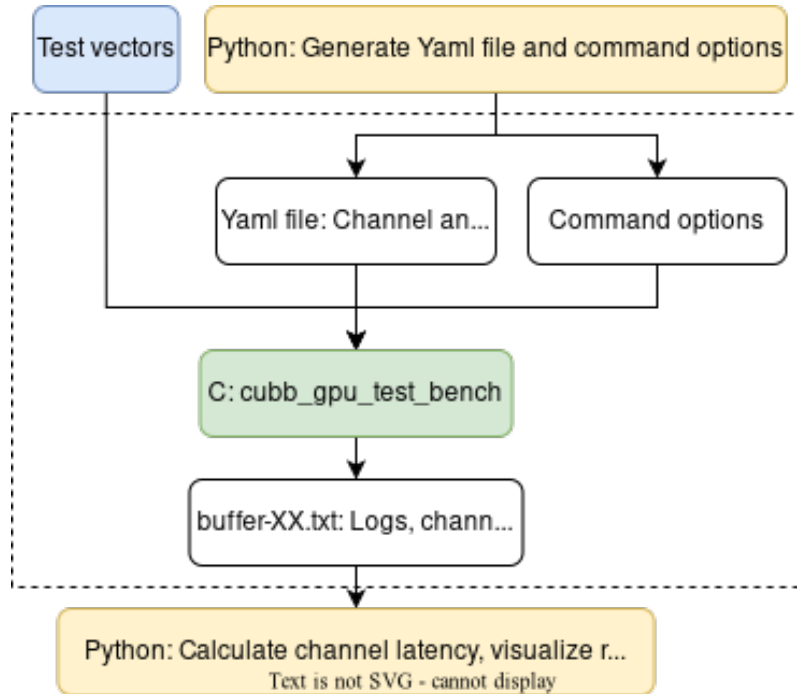
```

Each test input file contains multiple tests for different code rates, as specified by the number of parity nodes.



### 1.5.3.4 Running cuPHY Performance Testing Scripts

*aerial\_sdk/testBench*s provides a multi-cell multi-channel test bench to test cuPHY standalone performance. It relies on [NVIDIA Multi-Process Service \(MPS\)](#) to share the GPU among multiple channels. Specifically, there are two folders and their relationship can be summarized as follows:



- ▶ *cubb\_gpu\_test\_bench*: a C test bench that runs the multi-cell multi-channel cuPHY standalone GPU workload (that is, without I/O to and from NIC or layer 2). The input of *cubb\_gpu\_test\_bench* are test vectors, a Yaml file, and some command options to run the GPU workload. The output is a *buffer-XX.txt* file that has the logs, channel start/end times, debug info, etc. Here XX is the number of cells used in testing.
- ▶ *perf*: a set of Python scripts to automate performance testing using *cubb\_gpu\_test\_bench*. The Python scripts can help generate the Yaml file and command options, config GPU and MPS before running *cubb\_gpu\_test\_bench*; collect the test results by reading the output *buffer-XX.txt* from *cubb\_gpu\_test\_bench*.

#### 1.5.3.4.1 Generating Test Vectors using Matlab 5GModel

Run this Matlab command:

```

cd <5GModel root>/nr_matlab
startup
genCfgTV_perf_ss('performance-avg.xlsm');
genCfgTV_perf_ss_bwc('performance-avg.xlsm');
genCfgTV_perf_pucch();
genCfgTV_perf_pdch();
genCfgTV_perf_prach();
genCfgTV_perf_csirs();
genCfgTV_perf_ssb();
genCfgTV_perf_srs();
  
```

All the cuPHY Performance test vectors are generated and stored under `nr_matlab/GPU_test_input`.

#### 1.5.3.4.2 Measuring cuPHY Performance using `cubb_gpu_test_bench`

Requirements:

- ▶ The performance measurements can be run using a Linux environment making one of more GPU available. Such environment is here assumed to have:
  - ▶ bash or zsh as default shell
  - ▶ Python 3.8+ and the following packages: numpy, pyCUDA, pyYAML
  - ▶ CUDA toolkit 11.4 or above properly configured so that `nvidia-cuda-mps-control` and `nvidia-smi` are in PATH
  - ▶ The executable `cubb_gpu_test_bench` is located in the `<testBenches>/build` folder.

There are three steps when measuring cell capacity using `cubb_gpu_test_bench`. The `perf` folder provides some pre defined test cases. Below is an example of 4T4R (F08) using TDD pattern DDDSU-UDDDD.

1. Generate the JSON file that defines the use case (e.g., 8~16 peak or average cells)

```
python3 generate_avg_TDD.py --peak 8 9 10 11 12 13 14 15 16 --avg 0 --exact --
↪case F08
```

2. Measure the latency of all channels based on predefined patterns

```
python3 measure.py --cuphy <testBenches>/build --vectors <test_vectors> --config
↪testcases_avg_F08.json --uc uc_avg_F08_TDD.json --delay 100000 --gpu <GPU_ID> --
↪freq <GPU_freq> --start <cell_start> --cap <cell_cap> --iterations 1 --slots
↪<nSlots> --power <budget> --target <sms_prach> <sms_pdcch> <sms_pucch> <sms_
↪pdsch> <sms_pusch> <sms_ssb> --2cb_per_sm --save_buffer --priority --prach --
↪prach_isolate --pdcch --pdcch_isolate --pucch --pucch_isolate --tdd_pattern
↪dddsuudddd --pusch_cascaded --ssb --csirs --groups_dl --pack_pdsch --groups_
↪pusch --ldpc_parallel <--graph>
```

- ▶ `<GPU_ID>`: The ID of the GPU on which the measurements are to be run (e.g. 0 for single GPU systems)
- ▶ `<GPU_freq>`: The GPU clock frequency in MHz
- ▶ `<cell_start>`: The minimum number of cells to be tested
- ▶ `<cell_cap>`: The maximum number of cells to be tested. The Python scripts will run `cubb_gpu_test_bench` for a range of [`<cell_start>`, `<cell_cap>`] cells and collect the latency results.
- ▶ `<budget>`: The power budget in Watts
- ▶ `<sms_channelName>`: The number of streaming multiprocessors used per MPS sub-context for each channel during the run, where `channelName` can be “PRACH”, “PDCCH”, “PUCCH”, “PDSCH”, “PUSCH”, or “SSB”
- ▶ `<--graph>` Runs the measurement in graph mode. If this parameter is not included, stream mode will be used.

**Note**

Use `--test` to see which YAML file and command options the Python scripts generated without running the tests on GPU.

3. Visualize the latency of each channel (this step requires Python library `matplotlib`). We generate a `compare-<date>.png` file showing the CDF of the latency for all tested channels:

- ▶ If run in stream mode:

```
python3 compare.py --filename <sms_prach>_<sms_pdcch>_<sms_pucch>_<sms_pdsch>_
↳<sms_pusch>_<sms_ssb>_sweep_streams_avg_F08.json --cells <nCell>+0
```

- ▶ If run in graph mode:

```
python3 compare.py --filename <sms_prach>_<sms_pdcch>_<sms_pucch>_<sms_pdsch>_
↳<sms_pusch>_<sms_ssb>_sweep_graphs_avg_F08.json --cells <nCell>+0
```

Where `<nCell>` is the number of cells we would like to visualize the latency results

It is possible to compare latency results of different number of cells in one figure. For instance, we can compare the latency of 8 cells and 9 cells:

```
python3 compare.py --filename <sms_prach>_<sms_pdcch>_<sms_pucch>_<sms_pdsch>_
↳<sms_pusch>_<sms_ssb>_sweep_graphs_avg_F08.json <sms_prach>_<sms_pdcch>_<sms_
↳pucch>_<sms_pdsch>_<sms_pusch>_<sms_ssb>_sweep_graphs_avg_F08.json --cells 8+0
↳9+0
```

In all cases, Aerial CUDA-Accelerated RAN offers the possibility of measuring the latency of all workloads including:

- ▶ Dynamic and heterogeneous traffic (meaning that each cell is stimulated with different test vectors and every slot sees a different allocation of the test vectors to the considered cells)
- ▶ Specific traffic models

### 1.5.4. Running cuBB End-to-End

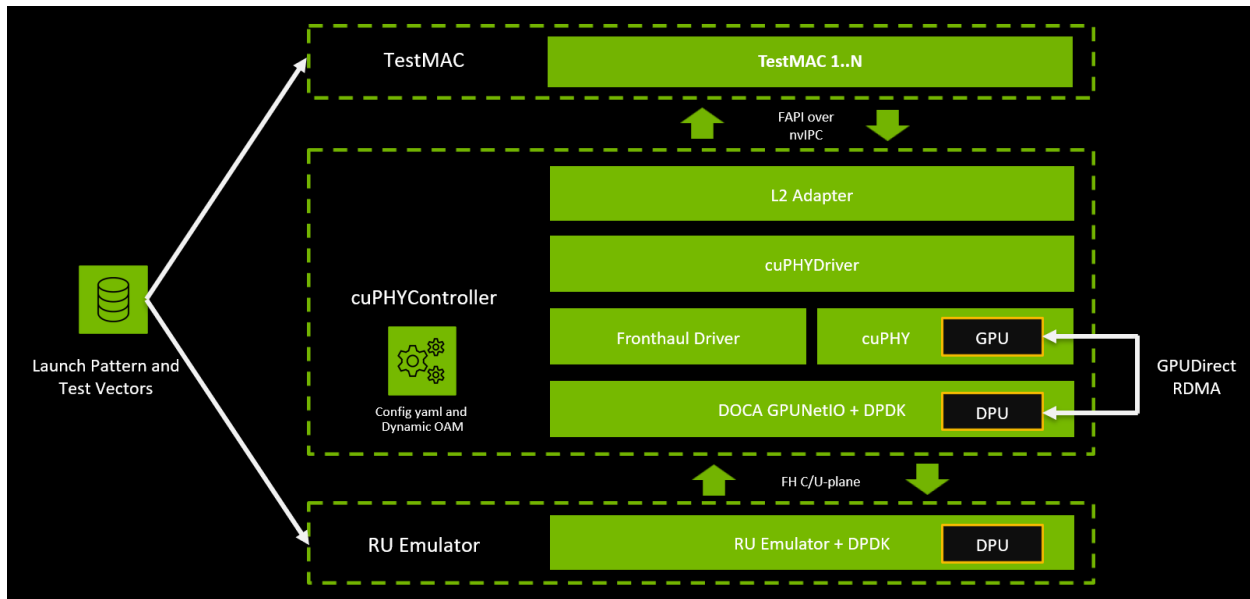
Beyond the cuPHY layer 1 PHY software and its standalone examples, this section describes how to build and run the cuBB software components shown in the block diagram below.

- ▶ The cuPHYController block operates between L2 and the RU fronthaul interface. It interfaces through cuPHY and DOCA GPUNetIO + DPDK to operate the GPU and the NIC.
  - ▶ L2 Adapter: This module communicates with L2 or TestMAC through FAPI messages over nvIPC. It receives downlink and uplink scheduling commands from L2 and converts it to internal cuPHYDriver API calls.
  - ▶ cuPHYDriver: This module distributes the UL and DL tasks among the available worker threads. It interacts with GPU for the following tasks:
    - ▶ To prepare and trigger a new UL/DL cuPHY processing through the cuPHY API.
    - ▶ To launch UL packets ordering with CUDA kernel.

It interacts with DPU/NIC through the Fronthaul Driver to send and receive ORAN fronthaul packets (C/U-plane).

- ▶ The RU Emulator emulates the network traffic of single or multiple RU. It validates the following:
  - ▶ All packet timing for DL direction packets (i.e. DL-C, UL-C, DL-U) based on configurable ORAN packet windows.
  - ▶ It checks for all packets that the eCPRI packet structure is aligned to ORAN specs.
  - ▶ It validates the IQ samples in the DL U-plane payload and expected section sizes for different compression methods.
  - ▶ It validates the BFW IQ samples in DL/UL C-plane, and RE mask in DL-C for CSI-RS/PDSCH.
  - ▶ It validates UL-C section information for PUCCH/PUSCH/PRACH/SRS and responds with corresponding UL U-plane.
- ▶ The TestMAC simulates the L2 and provides the FAPI interface over nvIPC. It validates the following:
  - ▶ It calculates the expected throughput data from the launch pattern and TVs and print to console. Then a python script can be used to validate the throughput of both TestMAC and RU. The throughput data include: Prmb/HARQ/SR/CSI/SRS number, channel numbers, DL/UL data rate. Unit is number per seconds.
  - ▶ It validates the UL FAPI message data structure and TB buffers by comparing with the preloaded data from TVs.
  - ▶ It validates the UL FAPI timing (The number of slots that the UL FAPI messages expect to receive).

The cuPHYController is exercised with an environment between the RU Emulator and the TestMAC.



The L1/L2 interface is based on the 5G FAPI 222.10.02 with partial 222.10.04 defined by the Small Cell Forum (SCF). For the supported message and PDU types and exceptions, refer to *cuBB Release Notes*.

### 1.5.4.1 Building the cuBB End-to-End

The following procedure describes the steps for building the end-to-end components in Aerial cuBB.

1. Inside the cuBB container, go to the SDK folder:

```
$ cd /opt/nvidia/cuBB
```

2. Create the build directory with build options:

- DSCF\_FAPI\_10\_04=ON to enable the supported FAPI 10.04 fields (for example, SRS).
- DENABLE\_CONFORMANCE\_TM\_PDSCH\_PDCCH=ON to run Test Mode (TM) tests.
- DENABLE\_20C=ON to run more than 16 cells on Grace Hopper MGX system
- DENABLE\_STATIC\_BFW=ON to run 64T64R with dynamic + static beamforming test

#### Note

The compile time flag *DYNAMIC\_SFN\_SLOT* has been replaced by the *l2\_adapter* yaml startup time option *enableTickDynamicSfnSlot*. The default is 1 (Dynamic SFN slot enabled) if this field is not present in the *l2\_adapter* yaml. It is no longer necessary to run cmake with the *-DDYNAMIC\_SFN\_SLOT=ON/OFF* flag. The same binaries can be used in RU emulator configuration and eLSU/O-RU configuration. The *DYNAMIC\_SFN\_SLOT* option has been removed entirely from CMakeLists.txt since Aerial 23-4 release.

For example, to run F08 performance benchmarking, use the following CMake command:

```
$ cmake -Bbuild -GNinja -DCMAKE_TOOLCHAIN_FILE=cuPHY/cmake/toolchains/native -
↳DSCF_FAPI_10_04=ON -DENABLE_CONFORMANCE_TM_PDSCH_PDCCH=ON
```

To run 20C test on Grace Hopper MGX system, use the following CMake command:

```
$ cmake -Bbuild -GNinja -DCMAKE_TOOLCHAIN_FILE=cuPHY/cmake/toolchains/native -
↳DSCF_FAPI_10_04=ON -DENABLE_CONFORMANCE_TM_PDSCH_PDCCH=ON -DENABLE_20C=ON
```

To build with default option, use the following CMake command:

```
$ cmake -Bbuild -GNinja -DCMAKE_TOOLCHAIN_FILE=cuPHY/cmake/toolchains/native
```

The following are supported build variants:

#### FAPI 10.02

RU Type \ Build Options	Default (no build flag)
RU emulator: No build flag; instead, <i>enableTickDynamicSfnSlot</i> : 0 is set in the <i>l2_adapter</i> YAML file	<pre>cmake -Bbuild -GNinja -DCMAKE_TOOLCHAIN_ ↳FILE=cuPHY/cmake/toolchains/native</pre>
Keysight eLSU: Default (no build flag)	<pre>cmake -Bbuild -GNinja -DCMAKE_TOOLCHAIN_ ↳FILE=cuPHY/cmake/toolchains/native</pre>

**Enable FAPI 10.04**

RU Type \ Build Options	DSCF_FAPI_10_04=ON
RU emulator: No build flag; instead, enableTickDynamicSfnSlot: 0 is set in the l2_adapter YAML file	<pre>cmake -Bbuild -GNinja -DCMAKE_TOOLCHAIN_ ↪FILE=cuPHY/cmake/toolchains/native -DSCF_ ↪FAPI_10_04=ON</pre>
Keysight eLSU: Default (no build flag)	N/A

**Enable TestMode**

RU Type \ Build Options	DENABLE_CONFORMANCE_TM_PDSCH_PDCCH=ON
RU emulator: No build flag; instead, enableTickDynamicSfnSlot: 0 is set in the l2_adapter YAML file	<pre>cmake -Bbuild -GNinja -DCMAKE_TOOLCHAIN_ ↪FILE=cuPHY/cmake/toolchains/native -DENABLE_ ↪CONFORMANCE_TM_PDSCH_PDCCH=ON</pre>
Keysight eLSU: Default (no build flag)	N/A

**Enable 20C on Grace Hopper MGX**

RU Type \ Build Options	DENABLE_20C=ON
RU emulator: No build flag; instead, enableTickDynamicSfnSlot: 0 is set in the l2_adapter YAML file	<pre>cmake -Bbuild -GNinja -DCMAKE_TOOLCHAIN_ ↪FILE=cuPHY/cmake/toolchains/native -DSCF_FAPI_10_04=ON `` -DENABLE_CONFORMANCE_TM_PDSCH_PDCCH=ON -DENABLE_20C=ON</pre>
Keysight eLSU: Default (no build flag)	N/A

**Note**

When building for E2E test, “-DENABLE\_L2\_SLT\_RSP=ON” is enabled by default in the cmake build options. It requires the L2 to support the vendor-specific message “SLOT.response”. If the L2 doesn’t support it, “-DENABLE\_L2\_SLT\_RSP=OFF” must be included in the cmake build option to turn off this feature in L1.

ENABLE\_L2\_SLT\_RSP=ON is recommended.

Option ENABLE\_L2\_SLT\_RSP must be configured with the same value in L1, L2, and lib-nvipc.so standalone build for L2: (1) L1: cuBB\_SDK. (2) libnvipc.so standalone build for L2. Refer to \${cuBB\_SDK}/cuPHY-CP/gt\_common\_libs/README.md. (3) L2: gNB DU code which includes nv\_ipc.h. To confirm whether it was enabled, run “grep ENABLE\_L2\_SLT\_RSP

build/CMakeCache.txt” for (1) and (2), print sizeof(nv\_ipc\_t) in L2 code for (3).

### 3. Build the Aerial cuBB components as follows.

To build all Aerial cuBB components, use these commands:

```
$ cd ${cuBB_SDK}
$ cmake --build build
```

To build only the cuPHY, use these commands:

```
$ cd ${cuBB_SDK}/cuPHY
$ cmake -Bbuild -GNinja -DCMAKE_TOOLCHAIN_FILE=cmake/toolchains/native
$ cmake --build build
```

To build only the Test MAC, use these commands:

```
$ cd ${cuBB_SDK}
$ cmake --build build -t test_mac
```

To build only the cuPHY controller, use these commands:

```
$ cd ${cuBB_SDK}
$ cmake --build build -t cuphycontroller_scf
```

To build only the cuPHY driver, use these commands:

```
$ cd ${cuBB_SDK}
$ cmake --build build -t cuphydriver
```

To build only the RU emulator, use these commands:

```
$ cd ${cuBB_SDK}
$ cmake --build build -t ru_emulator
```

To compile the Aerial code in the container on a devkit or Dell R750 machine that has isolcpus restricting cores, you can override isolcpus using the following command: The example command uses cores 10-20.

```
$ sudo chrt -r 1 sudo -u aerial taskset -c 10-20 cmake --build build
```

#### 1.5.4.2 nvlog Configuration

Aerial-SDK use nvlog as logger. It is based on the opensource FMT logger. Configuration file is located at ./cuPHY/nvlog/config/nvlog\_config.yaml.

Log files are stored at /tmp directory by default and the path can be overridden by environment variable AERIAL\_LOG\_PATH.

Maximum log file size can be configured by max\_file\_size\_bytes to avoid exhausting the system disk storage.

To configure global log level, set “shm\_log\_level: <level>”. To configure log level for a specific tag, add a “shm\_level: <level>” line under the tag name line. As an example, below configuration sets global log level to 3 - CONSOLE level and sets “FH.LATE\_PACKETS” tag to 5 - INFO level.

```
# log files stored at /tmp directory (default)
# log file path can be customized using environment variable $AERIAL_LOG_PATH
# Log levels: 0 - NONE, 1 - FATAL, 2 - ERROR, 3 - CONSOLE, 4 - WARNING, 5 - INFO, 6 -
↪DEBUG, 7 - VERBOSE

nvlog:
  shm_log_level: 3 # Global log level
  max_file_size_bytes: 5000000000 # Size in bytes The rotating log files in /tmp
↪(default)
  nvlog_tags:
    - 0: "" # Reserve number 0 for no tag print
      shm_level: 5 # Example: overlay shm_log_level for a tag

    - 621: "FH.LATE_PACKETS"
      shm_level: 5
```

### 1.5.4.3 Updating Configuration Files for End-to-End

This section describes the config parameters that you can modify to run end-to-end.

#### 1.5.4.3.1 Server #1 (to Run TestMAC and cuPHYController)

There are several common configurations. Check and edit the following parameters in the .yaml file:

1. Configure the NIC address in the following configuration files depending on the setup you are using, these are the default files provided:
  - ▶ cuphycontroller\_F08\_CG1.yaml
  - ▶ cuphycontroller\_F08\_R750.yaml
  - ▶ cuphycontroller\_nrSim\_SCF.yaml
2. Edit the NIC PCIe address to match the NIC hardware PCIe address. For example, the FH NIC on R750 gNB uses PCIe address 0000:cc:00.0:

```
$ sed -i "s/ nic:./ nic: 0000:cc:00.0/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/
↪config/cuphycontroller_F08_R750.yaml
```

3. Check the GPU ID for the GPU that is sharing the PCIe switch with the NIC. The gpus parameter shown below has a default value of 0 for a GPU ID of 0. If GPU 0 is not the GPU you want to use, replace 0 in the sed command line and run it:

```
$ sed -i "/gpus:/{n;s/.*/ - 0/}" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/
↪cuphycontroller_F08_*.yaml
```

If the system has only one GPU card, you can keep the default setting of 0.

To identify which GPU is sharing the PCIe switch with the NIC, use the following command:

```
$ nvidia-smi topo -m
```

In the output, look for the GPU connected to the NIC with connection type of PIX (where they intersect in the table). In the example below, GPU 0 in the column is the one with the PIX intersecting with Mellanox mlx5\_0 and mlx5\_1. Use GPU ID value of 0 for the .yaml gpus parameter.



	GPU0	mlx5_0	mlx5_1	CPU Affinity
GPU0	X	PIX	PIX	0-23
mlx5_0	PIX	X	PIX	
mlx5_1	PIX	PIX	X	

The meaning of PIX is:

**X** = Self  
**SYS** = Connection traversing PCIe and the SMP interconnect between NUMA nodes (e.g., QPI/UPI)  
**NODE** = Connection traversing PCIe and the interconnect between PCIe Host Bridges within a NUMA node  
**PHB** = Connection traversing PCIe and a PCIe Host Bridge (typically the CPU)  
**PXB** = Connection traversing multiple PCIe bridges (without traversing the PCIe Host Bridge)  
**PIX** = Connection traversing at most a single PCIe bridge  
**NV#** = Connection traversing a bonded set of # NVLinks

#### Note

Aerial-SDK expects the set of eAxCid ports to be the same between DL and UL channels (excluding PRACH). Make sure that the same set of port indices in the YAML configuration file are configured for DL and UL channels. For example, if the set of port indices [0,8,1,2] are configured for PDSCH, the same setting should be used for PDCCH, SSB/PBCH, and CSI-RS. Similarly, if the set of port indices [0,8] are configured for PUSCH, the same set of indices should be used for PUCCH. The number of eAxCid ports between DL and UL channels does not need to be the same.

To enable early HARQ, set `pusch_subSlotProcEn` to 1 in `cuphycontroller` config:

```
sed -i "s/ pusch_subSlotProcEn: */ pusch_subSlotProcEn: 1/" ${cuBB_SDK}/cuPHY-CP/
↳cuphycontroller/config/cuphycontroller_F08_*.yaml
```

To activate early HARQ, set `uciIndPerSlot` to 2 in `test_mac_config.yaml`:

```
sed -i "s/ uciIndPerSlot : */ uciIndPerSlot : 2/" ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/
↳test_mac_config.yaml
```

#### Note

This split UCI.indication with early-HARQ feature is enabled only in FAPI 10.04. To enable this feature, build with compilation flag `-DSCF_FAPI_10_04=ON`. This feature is enabled at `cuPHY`, if `pusch_subSlotProcEn` is set to 1 in `cuphycontroller` config. But `cuPHY` does not report early HARQ for UCI on PUSCH until L2 sends `config.request` with TLV `0x102B` `indicationInstancesPerSlot.UCI.indication = 2`. To instruct `testMac` to send this TLV in `config.request` set `uciIndPerSlot` to 2 in `test_mac_config.yaml`.

```
sed -i "s/ pusch_subSlotProcEn: */ pusch_subSlotProcEn: 1/" ${cuBB_SDK}/cuPHY-CP/
↳cuphycontroller/config/cuphycontroller_F08_*.yaml
```

```
sed -i "s/ uciIndPerSlot : */ uciIndPerSlot : 2/" ${cuBB_SDK}/cuPHY-CP/testMAC/
↳testMAC/test_mac_config.yaml
```

```
sed -i "s/ mCh_segment_proc_enable: */ mCh_segment_proc_enable: 1/" ${cuBB_SDK}/
↳cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_*.yaml
```

```
sed -i "s/ channel_segment_timelines: */ channel_segment_timelines: 1/" ${cuBB_SDK}/
↳cuPHY-CP/testMAC/testMAC/test_mac_config.yaml
```

**Note**

To enable enhanced L1-L2 interace, early-HARQ feature must be enabled as above and compiled with FAPI 10.04. To enable this feature, build with compilation flag `-DSCF_FAPI_10_04=ON`. To instruct testMac to send TLV CONFIG\_TLV\_VENDOR\_CHAN\_SEGMENT (0xA018), set `channel_segment_timelines` to 1 in `test_mac_config.yaml`. The expectation is that there is an `ErrorIndication` sent when the timelines don't meet the processing from `cuPHYDriver`.

**1.5.4.3.2 Server #2 (to Run RU Emulator)**

The RU emulator reads a configuration file located at: `cuBB_SDK/cuPHY-CP/ru-emulator/config/config.yaml`.

Before running the `ru-emulator`, modify the `config.yaml` to match your server system hardware settings.

There are two parameters to modify in the `config.yaml` file:

```
# PCI Address of NIC interface used
nic_interface: b5:00.0
# MAC address of cuPHYController port in use on server#1
peerethaddr: 1c:34:da:ff:ff:fe
```

Update the `nic_interface` and `peerethaddr` according to the systems used. Look up the addresses of these NIC interfaces.

- ▶ `nic_interface` is the NIC port PCIe bus address on the system running RU emulator. Replace `0000:b5:00.0` with the PCIe address of NIC for use.
- ▶ `peerethaddr` is the NIC port MAC address on the system running `cuPHYController`. Replace the MAC address with the MAC address of the NIC used in Server#1.

Replace `0000:b5:00.0` with the PCIe address of NIC port for use:

```
$ sed -i "s/nic_interface.*/nic_interface: 0000:b5:00.0/" ${cuBB_SDK}/cuPHY-CP/ru-
↪emulator/config/config.yaml
```

Replace the MAC address with the MAC address of the NIC port used in Server#1:

```
$ sed -i "s/peerethaddr.*/peerethaddr: 1c:34:da:ff:ff:fe/" ${cuBB_SDK}/cuPHY-CP/ru-
↪emulator/config/config.yaml
```

Run the following command on the host to identify the correct PCIe address and the MAC address.

```
$ sudo lshw -c network -businfo
```

Bus info	Device	Class	Description
pci@0000:05:00.0	eno1	network	I210 Gigabit Network Connection
pci@0000:06:00.0	enp6s0	network	I210 Gigabit Network Connection
pci@0000:b5:00.0	ens6f0	network	MT2892 Family [ConnectX-6 Dx]
pci@0000:b5:00.1	ens6f1	network	MT2892 Family [ConnectX-6 Dx]
	vethdf87878	network	Ethernet interface

To find the MAC address of the NIC port, run the following command:

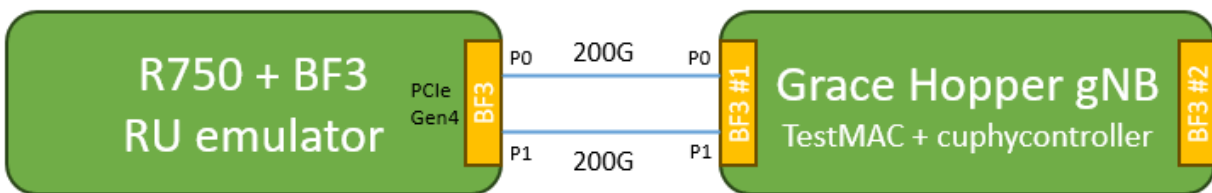
```
$ ifconfig -a
...
68: ens6f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1514 qdisc mq state UP group
↳ default qlen 1000
    link/ether 1c:34:da:ff:ff:fe brd ff:ff:ff:ff:ff:ff
    inet6 fe80::bace:f6ff:fe33:fe16/64 scope link
        valid_lft forever preferred_lft forever
69: ens6f1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
↳ default qlen 1000
    link/ether 1c:34:da:ff:ff:ff brd ff:ff:ff:ff:ff:ff
    inet6 fe80::bace:f6ff:fe33:fe17/64 scope link
        valid_lft forever preferred_lft forever
```

The MAC addresses of the NIC port are under the link/ether label.

#### 1.5.4.4 Running Environment Initialization for End-to-End

This section describes how to run the various cuBB software components together. Here, the cuBB uses the GPU and the NIC for cuPHY L1 compute and for network data traffic acceleration.

A network connection is used between the two servers to physically connect the RU emulator and the cuBB gNB software stack.



To verify that PTP4L and PHC2SYS services are running, run the following commands on the host:

```
$ sudo systemctl status ptp4l.service
...
# check that the service is active and has low rms value (<30):
$ sudo systemctl status phc2sys.service
```

Verify the System Clock is synchronized and that NTP is off:

```
$ timedatectl
    Local time: Thu 2022-02-03 22:30:58 UTC
    Universal time: Thu 2022-02-03 22:30:58 UTC
    RTC time: Thu 2022-02-03 22:30:58
    Time zone: Etc/UTC (UTC, +0000)
System clock synchronized: yes
    NTP service: inactive
    RTC in local TZ: no
```

### 1.5.4.5 Running Examples for End-to-End (SCF FAPI)

This section describes how to run the cuBB end-to-end using the SCF FAPI.

There are three use case examples:

- ▶ Use case 1: testMAC + SCF L2 Adapter Standalone
- ▶ Use case 2: testMAC + cuPHYController\_SCF + RU Emulator
- ▶ Use case 3: testMAC + cuPHYController\_SCF + RU Emulator P5G PRACH

#### 1.5.4.5.1 Running testMAC + SCF L2 Adapter Standalone

1. Build all the modules as described in *Building cuBB for End-to-End*.
2. Run l2adapter in standalone mode:

```
sudo $cuBB_SDK/build/cuPHY-CP/scf12adapter/scf_app/cuphycontroller\  
/l2_adapter_cuphycontroller_scf
```

3. Run testMAC after l2adapter starts.

You can run different cases:

```
sudo $cuBB_SDK/build/cuPHY-CP/testMAC/testMAC/test_mac <Fxx> <xC> [-- channels  
-><CHANNELS>] --no-validation
```

Examples:

```
sudo $cuBB_SDK/build/cuPHY-CP/testMAC/testMAC/test_mac F08 1C --no-validation  
sudo $cuBB_SDK/build/cuPHY-CP/testMAC/testMAC/test_mac F08 2C --no-validation  
sudo $cuBB_SDK/build/cuPHY-CP/testMAC/testMAC/test_mac F08 3C --no-validation  
sudo $cuBB_SDK/build/cuPHY-CP/testMAC/testMAC/test_mac F08 4C --no-validation
```

4. Test result and test log: In the testMAC terminal output below, you can see the TTI tick counter and throughput:

```
08:32:15.793986 Cell 0 | DL 1586.28 Mbps 1600 Slots | UL 249.10 Mbps 400 Slots  
->| Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 150 | INV 0  
08:32:15.793996 Cell 1 | DL 1586.28 Mbps 1600 Slots | UL 249.10 Mbps 400 Slots  
->| Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 150 | INV 0  
08:32:15.794000 Cell 2 | DL 1586.28 Mbps 1600 Slots | UL 249.10 Mbps 400 Slots  
->| Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 150 | INV 0  
08:32:15.794003 Cell 3 | DL 1586.28 Mbps 1600 Slots | UL 249.10 Mbps 400 Slots  
->| Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 150 | INV 0
```

#### 1.5.4.5.2 Running testMAC + cuPHYController\_SCF + RU Emulator

##### Note

Before running the cuBB test case, restart MPS in each run. Run the following commands to export environment variables and restart MPS in the cuphycontroller terminal (do not run this for test\_mac and ru-emulator).

```
# Export variables
export CUDA_DEVICE_MAX_CONNECTIONS=8
export CUDA_MPS_PIPE_DIRECTORY=/var
export CUDA_MPS_LOG_DIRECTORY=/var

# Stop existing MPS
sudo -E echo quit | sudo -E nvidia-cuda-mps-control

# Start MPS
sudo -E nvidia-cuda-mps-control -d
sudo -E echo start_server -uid 0 | sudo -E nvidia-cuda-mps-control
```

The nvlog level can be changed in `cuBB_SDK/cuPHY/nvlog/config/nvlog_config.yaml` if needed. For example, to change to console only log level:

```
name: phy
- shm_log_level:5 # SHM log level
+ shm_log_level: 3 # SHM log level
```

Execute the following command to disable GPU (if there is one) for `ru_emulator`.

```
export CUDA_VISIBLE_DEVICES=""
```

Export might not work in some system environments. In this case, add the value before command as shown in the following example:

```
sudo -E CUDA_VISIBLE_DEVICES="" ./ru_emulator xxx
```

Without `CUDA_VISIBLE_DEVICES=""`, the following log is seen when `ru_emulator` is started with a GPU on the host. It does not affect the functionality.

```
15:15:56.251444 [FH.FLOW] [/opt/nvidia/cuBB/cuPHY-CP/aerial-fh-driver/lib/flow.cpp:
↪201] cuda failed with invalid argument
```

Configure the workers in `{cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_CG1.yaml` to for the CPUs to use in the L1 `cuPHYDriver`:

```
workers_ul:
- 5
- 6
workers_dl:
- 11
- 12
- 13
```

#### 1.5.4.5.2.1 Running the F08 Test Cases

Configure the `cell_group` in `{cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_CG1.yaml`: Set `cell_group` to 1 and set `cell_group_num` to the number of cells to run.

For example, to run 1C:

```
cell_group: 1
cell_group_num: 1
```

To run 2C:

```
cell_group: 1
cell_group_num: 2
```

To run 3C:

```
cell_group: 1
cell_group_num: 3
```

To run 4C:

```
cell_group: 1
cell_group_num: 4
```

F08 traffic patterns:

For Patterns 59C and 60C, you must enable the OTA conformance features in `cuphycontroller_F08_CG1.yaml`:

```
pusch_tdi: 1
pusch_cfo: 1
pusch_to: 1
pusch_dftsofdm: 0
pusch_select_eqcoeffalgo: 1
puxch_polarDcdrListSz: 8
```

For Patterns 60C you must set the `pusch_nMaxPrb` for each cell in `cuphycontroller_F08_CG1.yaml`:

```
pusch_nMaxPrb: 136
```

For Pattern 61, you must set the `pusch_nMaxPrb` for each cell in `cuphycontroller_F08_CG1.yaml`, this allows us to test 20C on Grace Hopper system:

```
pusch_nMaxPrb: 36
```

23-4 onwards supports early HARQ processing. For the 59C and 60C patterns, enable early HARQ processing with the following configurations:

```
# For early HARQ
sed -i 's/uciIndPerSlot : */uciIndPerSlot : 2/' ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/
↪test_mac_config.yaml
sed -i "s/pusch_subSlotProcEn: */pusch_subSlotProcEn: 1/" ${cuBB_SDK}/cuPHY-CP/
↪cuphycontroller/config/cuphycontroller_F08_CG1.yaml

# For early non HARQ
sed -i 's/uciIndPerSlot : */uciIndPerSlot : 0/' ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/
↪test_mac_config.yaml
sed -i "s/pusch_subSlotProcEn: */pusch_subSlotProcEn: 0/" ${cuBB_SDK}/cuPHY-CP/
↪cuphycontroller/config/cuphycontroller_F08_CG1.yaml
```

```
# For Enhanced L1 - L2 Interface
sed -i 's/uciIndPerSlot : */uciIndPerSlot : 2/' ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/
↪test_mac_config.yaml
sed -i "s/pusch_subSlotProcEn: */pusch_subSlotProcEn: 1/" ${cuBB_SDK}/cuPHY-CP/
↪cuphycontroller/config/cuphycontroller_F08_CG1.yaml
```

(continues on next page)

(continued from previous page)

```
sed -i "s/ mCh_segment_proc_enable:.* / mCh_segment_proc_enable: 1/" ${cuBB_SDK}/cuPHY-
↳CP/cuphycontroller/config/cuphycontroller_F08_CG1.yaml
sed -i "s/ channel_segment_timelines:.* / channel_segment_timelines: 1/"${cuBB_SDK}/
↳cuPHY-CP/testMAC/testMAC/test_mac_config.yaml
```

```
# Run F08 1C only as Enhanced L1 - L2 Interface is intended for 1 Cell.
```

For 24-3 and onward, patterns 59C peak and 60C average are the latest patterns used for performance testing.

For performance testing, use the following settings for testMAC to adjust the schedule time of the FAPI command, this requires a builder thread:

```
# testMAC configs for scheduling FAPI messages with appropriate L2 delay, also
↳configure testMAC to stop after 600k slots:
sed -i 's/schedule_total_time:.* /schedule_total_time: 455000/' ${cuBB_SDK}/cuPHY-CP/
↳testMAC/testMAC/test_mac_config.yaml
sed -i 's/builder_thread_enable:.* /builder_thread_enable: 1/' ${cuBB_SDK}/cuPHY-CP/
↳testMAC/testMAC/test_mac_config.yaml
sed -i 's/fapi_delay_bit_mask:.* /fapi_delay_bit_mask: 0xF/' ${cuBB_SDK}/cuPHY-CP/
↳testMAC/testMAC/test_mac_config.yaml

# optionally configure the test duration with the number of test_slots. Keep test_slots:
↳0 to run indefinitely.
sed -i 's/test_slots: 0/test_slots: 600000/' ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_
↳mac_config.yaml

# testMAC core configs, use free cores on the same NUMA, for example, the following
↳settings can be applied to an R750 using NUMA 1:
sed -i -z 's/ cpu_affinity:\s*[0-9]\+ / cpu_affinity: 35/2' ${cuBB_SDK}/cuPHY-CP/
↳testMAC/testMAC/test_mac_config.yaml
sed -i -z 's/ cpu_affinity:\s*[0-9]\+ / cpu_affinity: 33/1' ${cuBB_SDK}/cuPHY-CP/
↳testMAC/testMAC/test_mac_config.yaml
```

You must enable the PUSCH conformance flags and RU Emulator validation to account for beamforming:

```
# cuphycontroller configs for PUSCH conformance flags:
sed -i "s/pusch_tdi:.* /pusch_tdi: 1/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/
↳cuphycontroller_F08_CG1.yaml
sed -i "s/pusch_cfo:.* /pusch_cfo: 1/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/
↳cuphycontroller_F08_CG1.yaml
sed -i "s/pusch_to:.* /pusch_to: 1/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/
↳cuphycontroller_F08_CG1.yaml
sed -i "s/puxch_polarDcdrListSz:.* /puxch_polarDcdrListSz: 8/" ${cuBB_SDK}/cuPHY-CP/
↳cuphycontroller/config/cuphycontroller_F08_CG1.yaml

# RU emulator beamforming validation config
sed -i "s/enable_beam_forming:.* /enable_beam_forming: 1/" ${cuBB_SDK}/cuPHY-CP/ru-
↳emulator/config/config.yaml

# RU emulator 20C performance configurations on running a R750 system:
sed -i "s/ul_core_list.* /ul_core_list: [5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,
↳37,39,41,43]/" ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml
sed -i "s/dl_core_list.* /dl_core_list: [4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,
↳36,38,40,42]/" ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml
sed -i "s/aerial_fh_split_rx_tx_mempool.* /aerial_fh_split_rx_tx_mempool: 1/" ${cuBB_
```

(continues on next page)

(continued from previous page)

```
↪SDK}/cuPHY-CP/ru-emulator/config/config.yaml
sed -i "s/low_priority_core.*/low_priority_core: 45/" ${cuBB_SDK}/cuPHY-CP/ru-emulator/
↪config/config.yaml
```

To test 4T 4R TDD 7 beams series 59 and 60 with 80 slot patterns have been generated:

► **Series 59c: 20C peak cells, 7 beams, Full BW CSI-RS, OTA, 4 UL streams, 18 PUCCH UCIs + 6 PUSCH UCIs freq-multiplexed**

- PDSCH: 6 UEG / slot, MCS 27, 45 PRBs / UEG, (42 PRBs / UEG when having SSB)
- PUSCH: 6 UEG / slot, MCS 27, 42 PRBs / UEG, (34 PRBs / UEG when having 4 PRACH, 36 PRBs / UEG when having 3 PRACH)
- UCI@PUSCH: 4 HARQ, 37 CSI-1, 5 CSI-2
- PDCCH: 12 DCI / slot (6 DL + 6 UL)
- PUCCH: 18 UE frequency multiplexed (PF1)

► **Frame 0**

- Slot 0, 1, 2: ssb (2 blocks),
- Slot 3, ssb (1 block)
- Slot 6,8,10,16, TRS + CSIRS
- Slot 7,9,11,17, TRS
- Slot 5,15, PRACH

► **Frame 1**

- Slot 6,8,10, TRS + CSIRS
- Slot 7,9,11, TRS
- Slot 5,15, PRACH

► **Frame 2**

- Slot 0, 1, 2: ssb \*2,
- Slot 3, ssb
- Slot 6,7,8,9,10,11, 16,17 TRS
- Slot 5,15, PRACH

► **Frame 3**

- Slot 6,7,8,9,10,11 TRS
- Slot 5,15, PRACH
- TRS/CSI-RS in symbol 6+10 / 12 for even case number
- TRS/CSI-RS in symbol 5+9 / 13 for odd case number

► **Series 59d: 20C peak cells, 7 beams, Full BW CSI-RS, OTA, 4 UL streams, 24 PUCCH UCIs freq-multiplexed:**



- ▶ PDSCH: 6 UEG / slot, MCS 27, 45 PRBs / UEG, (42 PRBs / UEG when having SSB)
- ▶ PUSCH: 6 UEG / slot, MCS 27, 41 PRBs / UEG, (33 PRBs / UEG when having 4 PRACH, 35 PRBs / UEG when having 3 PRACH)
- ▶ UCI@PUSCH: 0 HARQ, 37 CSI-1, 5 CSI-2
- ▶ PDCCH: 12 DCI / slot (6 DL + 6 UL)
- ▶ PUCCH: 24 UE frequency multiplexed (PF1)
- ▶ **Frame 0**
  - ▶ Slot 0, 1, 2: ssb (2 blocks),
  - ▶ Slot 3, ssb (1 block)
  - ▶ Slot 6,8,10,16, TRS + CSIRS
  - ▶ Slot 7,9,11,17, TRS
  - ▶ Slot 5,15, PRACH
- ▶ **Frame 1**
  - ▶ Slot 6,8,10, TRS + CSIRS
  - ▶ Slot 7,9,11, TRS
  - ▶ Slot 5,15, PRACH
- ▶ **Frame 2**
  - ▶ Slot 0, 1, 2: ssb \*2,
  - ▶ Slot 3, ssb
  - ▶ Slot 6,7,8,9,10,11, 16,17 TRS
  - ▶ Slot 5,15, PRACH
- ▶ **Frame 3**
  - ▶ Slot 6,7,8,9,10,11 TRS
  - ▶ Slot 5,15, PRACH
  - ▶ TRS/CSI-RS in symbol 6+10 / 12 for even case number
  - ▶ TRS/CSI-RS in symbol 5+9 / 13 for odd case number
- ▶ **Series 59e: 30C peak cells, 7 beams, Full BW CSI-RS, 1 dmrs, 4 UL streams, 18 PUCCH UCIs + 6 PUSCH UCIs freq-multiplexed**
  - ▶ Same settings as 59c expect that only 1 dmrs.
  - ▶ PDSCH: 6 UEG / slot, MCS 27, 45 PRBs / UEG, (42 PRBs / UEG when having SSB)
  - ▶ PUSCH: 6 UEG / slot, MCS 27, 42 PRBs / UEG, (34 PRBs / UEG when having 4 PRACH, 36 PRBs / UEG when having 3 PRACH)
  - ▶ UCI@PUSCH: 4 HARQ, 37 CSI-1, 5 CSI-2
  - ▶ PDCCH: 12 DCI / slot (6 DL + 6 UL)

- ▶ PUCCH: 18 UE frequency multiplexed (PF1)
- ▶ TRS/CSI-RS in symbol 6+10 / 12 for even case number
- ▶ TRS/CSI-RS in symbol 5+9 / 13 for odd case number
- ▶ **Series 60c: 7 beams, 100 MHz (273 PRBs), 20C, ave cell, OTA, disjoint PDSCH and CSIRS, 4 UL streams, 18 PUCCH UCIs freq-multiplexed**
  - ▶ PDSCH: 6 UEG / slot, MCS 27, 22 PRBs / UEG, (18 PRBs / UEG when having ssb)
  - ▶ PUSCH: 6 UEG / slot, MCS 27, 19 PRBs / UEG, (11 PRBs / UEG when having 4 PRACH, 13 PRBs / UEG when having 3 PRACH)
  - ▶ UCI@PUSCH: 4 HARQ, 37 CSI-1, 5 CSI-2 (early HARQ enabled)
  - ▶ PDCCH: 12 DCI / slot (6 DL + 6 UL)
  - ▶ PUCCH: 18 UE frequency multiplexed (PF1)
  - ▶ **Frame 0**
    - ▶ Slot 0, 1, 2: ssb (2 blocks),
    - ▶ Slot 3, ssb (1 block)
    - ▶ Slot 6,8,10,16, TRS + CSIRS
    - ▶ Slot 7,9,11,17, TRS
    - ▶ Slot 5,15, PRACH
  - ▶ **Frame 1**
    - ▶ Slot 6,8,10, TRS + CSIRS
    - ▶ Slot 7,9,11, TRS
    - ▶ Slot 5,15, PRACH
  - ▶ **Frame 2**
    - ▶ Slot 0, 1, 2: ssb \*2,
    - ▶ Slot 3, ssb
    - ▶ Slot 6,7,8,9,10,11, 16,17 TRS
    - ▶ Slot 5,15, PRACH
  - ▶ **Frame 3**
    - ▶ Slot 6,7,8,9,10,11 TRS
    - ▶ Slot 5,15, PRACH
    - ▶ TRS/CSI-RS in symbol 6+10 / 12 for even case number
    - ▶ TRS/CSI-RS in symbol 5+9 / 13 for odd case number
- ▶ **Series 60d: 7 beams, 100 MHz (273 PRBs), 20C, ave cell, OTA, disjoint PDSCH and CSIRS, 4 UL streams, 24 PUCCH UCIs freq-multiplexed:**
  - ▶ PDSCH: 6 UEG / slot, MCS 27, 22 PRBs / UEG, (18 PRBs / UEG when having ssb)

- ▶ PUSCH: 6 UEG / slot, MCS 27, 18 PRBs / UEG, (10 PRBs / UEG when having 4 PRACH, 12 PRBs / UEG when having 3 PRACH)
- ▶ UCI@PUSCH: 0 HARQ, 37 CSI-1, 5 CSI-2 (early HARQ enabled)
- ▶ PDCCH: 12 DCI / slot (6 DL + 6 UL)
- ▶ PUCCH: 24 UE frequency multiplexed (PF1)
- ▶ **Frame 0**
  - ▶ Slot 0, 1, 2: ssb (2 blocks),
  - ▶ Slot 3, ssb (1 block)
  - ▶ Slot 6,8,10,16, TRS + CSIRS
  - ▶ Slot 7,9,11,17, TRS
  - ▶ Slot 5,15, PRACH
- ▶ **Frame 1**
  - ▶ Slot 6,8,10, TRS + CSIRS
  - ▶ Slot 7,9,11, TRS
  - ▶ Slot 5,15, PRACH
- ▶ **Frame 2**
  - ▶ Slot 0, 1, 2: ssb \*2,
  - ▶ Slot 3, ssb
  - ▶ Slot 6,7,8,9,10,11, 16,17 TRS
  - ▶ Slot 5,15, PRACH
- ▶ **Frame 3**
  - ▶ Slot 6,7,8,9,10,11 TRS
  - ▶ Slot 5,15, PRACH
  - ▶ TRS/CSI-RS in symbol 6+10 / 12 for even case number
  - ▶ TRS/CSI-RS in symbol 5+9 / 13 for odd case number
- ▶ **Series 62c: 30C peak cells, 7 beams, Full BW CSI-RS, OTA, 4 UL streams, 18 PUCCH UCIs + 6 PUSCH UCIs freq-multiplexed, PUSCH in S slot**
  - ▶ 59c + 4 symbols of pusch in S slot
  - ▶ PDSCH: 6 UEG / slot, MCS 27, 45 PRBs / UEG, (42 PRBs / UEG when having SSB)
  - ▶ PUSCH: 6 UEG / slot, MCS 27, 42 PRBs / UEG, (34 PRBs / UEG when having 4 PRACH, 36 PRBs / UEG when having 3 PRACH)
  - ▶ UCI@PUSCH: 4 HARQ, 37 CSI-1, 5 CSI-2
  - ▶ PDCCH: 12 DCI / slot (6 DL + 6 UL)
  - ▶ PUCCH: 18 UE frequency multiplexed (PF1)

► **Frame 0**

- Slot 0, 1, 2: ssb (2 blocks),
- Slot 3, ssb (1 block)
- Slot 6,8,10,16, TRS + CSIRS
- Slot 7,9,11,17, TRS
- Slot 5,15, PRACH

► **Frame 1**

- Slot 6,8,10, TRS + CSIRS
- Slot 7,9,11, TRS
- Slot 5,15, PRACH

► **Frame 2**

- Slot 0, 1, 2: ssb \*2,
- Slot 3, ssb
- Slot 6,7,8,9,10,11, 16,17 TRS
- Slot 5,15, PRACH

► **Frame 3**

- Slot 6,7,8,9,10,11 TRS
- Slot 5,15, PRACH
- TRS/CSI-RS in symbol 6+10 / 12 for even case number
- TRS/CSI-RS in symbol 5+9 / 13 for odd case number

► **Series 63c: 7 beams, 100 MHz (273 PRBs), 20C, ave cell, OTA, disjoint PDSCH and CSIRS, 4 UL streams, 18 PUCCH UCIs freq-multiplexed, PUSCH in S slot**

- 59c + 4 symbols of pusch in S slot
- PDSCH: 6 UEG / slot, MCS 27, 22 PRBs / UEG, (18 PRBs / UEG when having ssb)
- PUSCH: 6 UEG / slot, MCS 27, 19 PRBs / UEG, (11 PRBs / UEG when having 4 PRACH, 13 PRBs / UEG when having 3 PRACH)
- UCI@PUSCH: 4 HARQ, 37 CSI-1, 5 CSI-2 (early HARQ enabled)
- PDCCH: 12 DCI / slot (6 DL + 6 UL)
- PUCCH: 18 UE frequency multiplexed (PF1)

► **Frame 0**

- Slot 0, 1, 2: ssb (2 blocks),
- Slot 3, ssb (1 block)
- Slot 6,8,10,16, TRS + CSIRS

- ▶ Slot 7,9,11,17, TRS
- ▶ Slot 5,15, PRACH

▶ **Frame 1**

- ▶ Slot 6,8,10, TRS + CSIRS
- ▶ Slot 7,9,11, TRS
- ▶ Slot 5,15, PRACH

▶ **Frame 2**

- ▶ Slot 0, 1, 2: ssb \*2,
- ▶ Slot 3, ssb
- ▶ Slot 6,7,8,9,10,11, 16,17 TRS
- ▶ Slot 5,15, PRACH

▶ **Frame 3**

- ▶ Slot 6,7,8,9,10,11 TRS
- ▶ Slot 5,15, PRACH
- ▶ TRS/CSI-RS in symbol 6+10 / 12 for even case number
- ▶ TRS/CSI-RS in symbol 5+9 / 13 for odd case number

```
sudo -E ./cuphycontroller_scf F08_CG1
sudo ./test_mac F08 4C 59C
sudo ./ru_emulator F08 4C 59C
```

```
21:40:26.213585 WRN 2231 0 [RU] Cell 0 DL 1469.14 Mbps 1400 Slots | UL 213.84 Mbps
↪400 Slots | PBCH 200 | PDCCH_UL 1600 | PDCCH_DL 1600 | CSI_RS 700 | PRACH
↪200 Slots | PUCCH 400 Slots | DL_C_ON 100.00% DL_U_ON 99.91% UL_C_ON 100.00%
↪|Seconds 459
21:40:26.213591 WRN 2231 0 [RU] Cell 1 DL 1469.14 Mbps 1400 Slots | UL 213.84 Mbps
↪400 Slots | PBCH 200 | PDCCH_UL 1600 | PDCCH_DL 1600 | CSI_RS 700 | PRACH
↪200 Slots | PUCCH 400 Slots | DL_C_ON 100.00% DL_U_ON 99.94% UL_C_ON 100.00%
↪|Seconds 459
21:40:26.213595 WRN 2231 0 [RU] Cell 2 DL 1469.14 Mbps 1400 Slots | UL 213.84 Mbps
↪400 Slots | PBCH 200 | PDCCH_UL 1600 | PDCCH_DL 1600 | CSI_RS 700 | PRACH
↪200 Slots | PUCCH 400 Slots | DL_C_ON 100.00% DL_U_ON 99.92% UL_C_ON 100.00%
↪|Seconds 459
21:40:26.213599 WRN 2231 0 [RU] Cell 3 DL 1469.14 Mbps 1400 Slots | UL 213.84 Mbps
↪400 Slots | PBCH 200 | PDCCH_UL 1600 | PDCCH_DL 1600 | CSI_RS 700 | PRACH
↪200 Slots | PUCCH 400 Slots | DL_C_ON 100.00% DL_U_ON 99.95% UL_C_ON 100.00%
↪|Seconds 459
```

On R750 A100X DU system F08 4C with pattern 60 (average pattern):

```
sudo -E ./cuphycontroller_scf F08_CG1
sudo ./test_mac F08 4C 60C
sudo ./ru_emulator F08 4C 60C
```

```

22:01:12.039024 WRN 2375 0 [RU] Cell 0 DL 523.10 Mbps 1400 Slots | UL 94.65 Mbps
↪400 Slots | PBCH 200 | PDCCH_UL 1600 | PDCCH_DL 1600 | CSI_RS 700 | PRACH
↪200 Slots | PUCCH 400 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00%
↪|Seconds 471
22:01:12.039030 WRN 2375 0 [RU] Cell 1 DL 523.10 Mbps 1400 Slots | UL 94.65 Mbps
↪400 Slots | PBCH 200 | PDCCH_UL 1600 | PDCCH_DL 1600 | CSI_RS 700 | PRACH
↪200 Slots | PUCCH 400 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00%
↪|Seconds 471
22:01:12.039034 WRN 2375 0 [RU] Cell 2 DL 523.10 Mbps 1400 Slots | UL 94.65 Mbps
↪400 Slots | PBCH 200 | PDCCH_UL 1600 | PDCCH_DL 1600 | CSI_RS 700 | PRACH
↪200 Slots | PUCCH 400 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00%
↪|Seconds 471
22:01:12.039037 WRN 2375 0 [RU] Cell 3 DL 523.10 Mbps 1400 Slots | UL 94.65 Mbps
↪400 Slots | PBCH 200 | PDCCH_UL 1600 | PDCCH_DL 1600 | CSI_RS 700 | PRACH
↪200 Slots | PUCCH 400 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00%
↪|Seconds 471

```

### 1.5.4.5.2.2 Simultaneous FH Port Test Configs with RU Emulator

The following TC can be tested with both FH ports:

- ▶ BFP9 2C 59c

To set up the two port test, you must set up the configurations appropriately.

You can choose between the following verified 2 port test topologies:

#### ▶ 1 GH and 1 RU server

- ▶ GH P0 <-> RU P0
- ▶ GH P1 <-> RU P1

#### ▶ 1 GH and 2 RU server

- ▶ GH P0 <-> RU 1 P0
- ▶ GH P1 <-> RU 2 P0

Note: For the scenario with 1 GH and 2 RU server, we need the three setups to be synchronized, i.e. with a FH switch as the PTP master in between the three systems.

cuPHYController configuration:

```

nics:
- nic: 0000:01:00.0
  mtu: 1514
  cpu_mbufs: 196608
  uplane_tx_handles: 64
  txq_count: 48
  rxq_count: 16
  txq_size: 8192
  rxq_size: 16384
  gpu: 0
- nic: 0000:01:00.1
  mtu: 1514
  cpu_mbufs: 196608

```

(continues on next page)

(continued from previous page)

```

uplane_tx_handles: 64
txq_count: 48
rxq_count: 16
txq_size: 8192
rxq_size: 16384
gpu: 0

```

In the cuPHYController cell configurations, you could set port that the cell would run traffic on:

```

cells:
- name: 0-RU 0
[...]
  nic: 0000:01:00.0
- name: 0-RU 1
[...]
  nic: 0000:01:00.1

```

For the first topology with a single RU emulator system, you could specify the NIC interfaces and the peer ethernet addresses with the address of the DU ports, for example:

```

nics:
- nic_interface: 0000:cc:00.0
- nic_interface: 0000:cc:00.1
peers:
- peerethaddr: 48:b0:2d:a6:28:02 # MAC address of DU port 0
- peerethaddr: 48:b0:2d:a6:28:03 # MAC address of DU port 1

```

Similarly for RU emulator config, appropriately assign the NIC and peer addresses, based on the index in the lists defined above:

```

cell_configs:
-
  name: "Cell1"
  peer: 0
  nic: 0
-
  name: "Cell2"
  peer: 1
  nic: 1

```

### 1.5.4.5.3 Running RU on a GH server

For running the RU on a GH server, please update the core bindings for the RU for the GH CPU numbering.

To support the 20C peak cell performance test cases, NUMA is not an issue, as an example, we can use the below core assignments:

Note that 41 is skipped due to the PTP4L/PHC2SYS core binding.

```

ul_core_list: [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
↪ 23]
dl_core_list: [24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 42,
↪ 43, 44]
low_priority_core: 45
aerial_fh_dpdk_thread: 46

```

Note: Please be sure to include the “0000” in the PCIe nic\_interface:

```
nics:  
- nic_interface: 0000:01:00.0
```

Please build the RU emulator binary on the arm server, the execution command is the same as the above examples.

### 1.5.4.5.4 Running the nrSim Test Cases

#### 1.5.4.5.4.1 PBCH

```
# Restart MPS  
sudo -E ./cuphycontroller_scf nrSim_SCF  
sudo ./test_mac nrSim 1901 --channels PBCH  
sudo ./ru_emulator nrSim 1901 --channels PBCH  
# Expect RU Emulator to report 100 PBCH per second
```

#### 1.5.4.5.4.2 PDCCH\_DL

```
# Restart MPS  
sudo -E ./cuphycontroller_scf nrSim_SCF  
sudo ./test_mac nrSim 2901 --channels PDCCH_DL  
sudo ./ru_emulator nrSim 2901 --channels PDCCH_DL  
# Expect RU Emulator to report 100 PDCCH_DL per second
```

#### 1.5.4.5.4.3 PDSCH

```
# Restart MPS  
sudo -E ./cuphycontroller_scf nrSim_SCF  
sudo ./test_mac nrSim 3901 --channels PDSCH  
sudo ./ru_emulator nrSim 3901 --channels PDSCH  
# Expect RU Emulator to report 100 PDSCH per second
```

#### 1.5.4.5.4.4 PUSCH

```
# Restart MPS  
sudo -E ./cuphycontroller_scf nrSim_SCF  
sudo ./test_mac nrSim 7901 --channels PUSCH  
sudo ./ru_emulator nrSim 7901 --channels PUSCH  
# Expect testMAC to report 100 PUSCH per second  
  
# PUSCH Mapping Type B  
# Restart MPS  
sudo -E ./cuphycontroller_scf nrSim_SCF  
sudo ./test_mac nrSim 7258 --channels PUSCH  
sudo ./ru_emulator nrSim 7258 --channels PUSCH  
# Expect testMAC to report 100 PUSCH per second
```

(continues on next page)



(continued from previous page)

```

# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 7259 --channels PUSCH
sudo ./ru_emulator nrSim 7259 --channels PUSCH
# Expect testMAC to report 100 PUSCH per second

#CSI P2
sed -i "s/enable_csip2_v3.*/enable_csip2_v3: 1/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/
↪config/cuphycontroller_nrSim_SCF.yaml
sed -i "s/enable_csip2_v3.*/enable_csip2_v3: 1/" ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/
↪test_mac_config.yaml

# Restart MPS
sed -i "s/ uciIndPerSlot :.*/ uciIndPerSlot : 2/" ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/
↪test_mac_config.yaml
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 7599 --channels PUSCH
sudo ./ru_emulator nrSim 7599 --channels PUSCH
# Expect testMAC to report 100 PUSCH and 100 CSIP2 per second

# Restart MPS
sed -i "s/ uciIndPerSlot :.*/ uciIndPerSlot : 2/" ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/
↪test_mac_config.yaml
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 7600 --channels PUSCH
sudo ./ru_emulator nrSim 7600 --channels PUSCH
# Expect testMAC to report 100 PUSCH and 100 CSIP2 per second

```

#### 1.5.4.5.4.5 PRACH

```

# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 5901 --channels PRACH
sudo ./ru_emulator nrSim 5901 --channels PRACH
# Expect testMAC to report 100 Preambles per second

# PRACH 16 PID/Slot and PRACH B4 4FDM
# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 5013 --channels PRACH
sudo ./ru_emulator nrSim 5013 --channels PRACH
Expect testMAC to receive 1600 Preambles per second

- Change tv_prach field as below in cuphycontroller_nrSim_SCF.yaml
  tv_prach: TVnr_5013_PRACH_gNB_CUPHY_s1p0.h5

Expect 4 RO occasions in each slot in phy.log in sequence mentioned in below.
RO 0 - PrmbIndex (2,5,8,11)
RO 1 - PrmbIndex (14,17,20,23)
RO 2 - PrmbIndex (32,35,26,29)
RO 3 - PrmbIndex (38,41,44,47)

# grep -i "RO\|prmbIndex" phy.log

```

(continues on next page)

(continued from previous page)

```

15:57:41.161874 I [DRV.PRACH] RO 0 SFN 599.01 Preambles num detected 4
15:57:41.161878 I [DRV.PRACH] SFN 599.01 #0 prmbIndex 2 prmbDelay 0.000000
↪prmbPower -2.878487
15:57:41.161880 I [DRV.PRACH] SFN 599.01 #1 prmbIndex 5 prmbDelay 0.000000
↪prmbPower -2.801307
15:57:41.161883 I [DRV.PRACH] SFN 599.01 #2 prmbIndex 8 prmbDelay 0.000000
↪prmbPower -3.207683
15:57:41.161886 I [DRV.PRACH] SFN 599.01 #3 prmbIndex 11 prmbDelay 0.000000
↪prmbPower -3.423241
15:57:41.161901 I [DRV.PRACH] RO 1 SFN 599.01 Preambles num detected 4
15:57:41.161904 I [DRV.PRACH] SFN 599.01 #0 prmbIndex 14 prmbDelay 0.000000
↪prmbPower -4.193221
15:57:41.161906 I [DRV.PRACH] SFN 599.01 #1 prmbIndex 17 prmbDelay 0.000000
↪prmbPower -4.011869
15:57:41.161909 I [DRV.PRACH] SFN 599.01 #2 prmbIndex 20 prmbDelay 0.000000
↪prmbPower -3.471422
15:57:41.161912 I [DRV.PRACH] SFN 599.01 #3 prmbIndex 23 prmbDelay 0.000000
↪prmbPower -3.552692
15:57:41.161924 I [DRV.PRACH] RO 2 SFN 599.01 Preambles num detected 4
15:57:41.161927 I [DRV.PRACH] SFN 599.01 #0 prmbIndex 32 prmbDelay 0.000000
↪prmbPower -4.954414
15:57:41.161930 I [DRV.PRACH] SFN 599.01 #1 prmbIndex 35 prmbDelay 0.000000
↪prmbPower -3.706564
15:57:41.161933 I [DRV.PRACH] SFN 599.01 #2 prmbIndex 26 prmbDelay 0.000000
↪prmbPower -4.333083
15:57:41.161935 I [DRV.PRACH] SFN 599.01 #3 prmbIndex 29 prmbDelay 0.000000
↪prmbPower -3.994442
15:57:41.161945 I [DRV.PRACH] RO 3 SFN 599.01 Preambles num detected 4
15:57:41.161947 I [DRV.PRACH] SFN 599.01 #0 prmbIndex 38 prmbDelay 0.000000
↪prmbPower -3.341729
15:57:41.161950 I [DRV.PRACH] SFN 599.01 #1 prmbIndex 41 prmbDelay 0.000000
↪prmbPower -4.641103
15:57:41.161952 I [DRV.PRACH] SFN 599.01 #2 prmbIndex 44 prmbDelay 0.000000
↪prmbPower -4.189767
15:57:41.161955 I [DRV.PRACH] SFN 599.01 #3 prmbIndex 47 prmbDelay 0.000000
↪prmbPower -4.946166

```

#### 1.5.4.5.4.6 NZP CSI\_RS

```

# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 4001 --channels CSI_RS
sudo ./ru_emulator nrSim 4001 --channels CSI_RS
# Expect RU Emulator to report 100 CSI_RS per second

```

#### 1.5.4.5.4.7 PDSCH + ZP CSI\_RS

To run TC 3323, 3338, and 3339, add `--channels CSI_RS+PDSCH` in the `test_mac` and `ru_emulator` commands.

```
# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 3323 --channels CSI_RS+PDSCH
sudo ./ru_emulator nrSim 3323 --channels CSI_RS+PDSCH
# Expect RU Emulator to count 100 CSI_RS and 100 PDSCH per second
```

#### 1.5.4.5.4.8 Precoding

```
# Below steps are applicable to precoding test for PDSCH, PDCCH, PBCH, and CSI_RS
# In l2_adapter_config_nrSim_SCF.yaml, set enable_precoding to 1
sed -i -z "s/enable_precoding: 0/enable_precoding: 1/" $cuBB_SDK/cuPHY-CP/
↳cuphycontroller/config/l2_adapter_config_nrSim_SCF.yaml
# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 3248 --channels PDSCH
# Reset enable_precoding to 0
sed -i -z "s/enable_precoding: 1/enable_precoding: 0/" $cuBB_SDK/cuPHY-CP/
↳cuphycontroller/config/l2_adapter_config_nrSim_SCF.yaml

# In ru-emulator/config/config.yaml, set dl_approx_validation to 1
sed -i -z "s/dl_approx_validation: 0/dl_approx_validation: 1/" $cuBB_SDK/cuPHY-CP/ru-
↳emulator/config/config.yaml

sudo ./ru_emulator nrSim 3248 --channels PDSCH
# Expect testMAC and RU Emulator both see 1.36 Mbps 100 Slots per second
# Reset dl_approx_validation to 0
sed -i -z "s/dl_approx_validation: 1/dl_approx_validation: 0/" $cuBB_SDK/cuPHY-CP/ru-
↳emulator/config/config.yaml
```

Note for 24-3 we need to enable `oam_cell_ctrl_cmd` on RU Emulator side for precoding-enabled nrSim test cases as well.

```
sed -i "s/oam_cell_ctrl_cmd:.*oam_cell_ctrl_cmd: 1/" $cuBB_SDK/cuPHY-CP/ru-emulator/
↳config/config.yaml
```

#### 1.5.4.5.4.9 PUCCH HARQ

```
# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 6001 --channels PUCCH
sudo ./ru_emulator nrSim 6001 --channels PUCCH
# Expect testMAC to report 100 HARQ indications and ru-emulator to report 100 PUCCH per
↳second
```

#### 1.5.4.5.4.10 PUCCH Format 2

```
# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 6201 --channels PUCCH
sudo ./ru_emulator nrSim 6201 --channels PUCCH
# Expect testMAC to report 100 HARQ indications and ru-emulator to report 100 PUCCH per
↪second
```

#### 1.5.4.5.4.11 PUCCH HARQ/SR

```
# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 6049 --channels PUCCH
sudo ./ru_emulator nrSim 6049 --channels PUCCH
# Expect testMAC to report 300 HARQ + 300 SR and ru-emulator to report 100 PUCCH per
↪second
```

#### 1.5.4.5.4.12 PUCCH Format 3

```
# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 6301 --channels PUCCH
sudo ./ru_emulator nrSim 6301 --channels PUCCH
# Expect testMAC to report 100 HARQ indications and ru-emulator to report 100 PUCCH per
↪second
```

#### 1.5.4.5.4.13 UCI on PUSCH

```
# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 7501
sudo ./ru_emulator nrSim 7501
# Expect testMAC to report 100 HARQ/s and UL slots/s

# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 7502
sudo ./ru_emulator nrSim 7502
# Expect testMAC to report 100 HARQ/s and UL slots/s

# Restart MPS
#UCI on PUSCH CSI part 2
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 7517
sudo ./ru_emulator nrSim 7517 --channel PUSCH
```

```
For 7517-7519, 7524-26, 7528-29
# Expect testMAC to report 100 CSI part2/s and 100 UL slots/s
# Expect cuphycontroller to report 0 CRC for 100 slots/s and 1.61 Mbps UL throughput
```

(continues on next page)

(continued from previous page)

```

For 7520-7523, 7527, 7530
# Expect testMAC to report 100 CSI part2/s
# Expect cuphycontroller to report 0 CRC for 100 slots/s

```

#### 1.5.4.5.4.14 SRS

To enable FAPI 10.04 fields for the SRS test, add `-DSCF_FAPI_10_04=ON` in the cmake options and do a clean build. The test cases for SRS validation are 8301 and 8302.

In `cuphycontroller_nrSim_SCF.yaml` - `enable_srs: 1`

```

# Restart MPS
# Running 8301
sudo ./ru_emulator nrSim 8301 --channels SRS or ./ru_emulator nrSim 8301 (default
↳support all channels)
sudo ./test_mac nrSim 8301 --channels SRS or ./test_mac nrSim 8301 (default support
↳all channels)
sudo -E ./cuphycontroller_scf nrSim_SCF
# Expect the testMac to report the number of received SRS is between 97 and 103 and INV
↳values per second to be 0.
# If the INV Values are greater than 0, there is either a SRS report mismatch or SRS
↳report parameter mismatch.

# Restart MPS
# Running 8302
sudo ./ru_emulator nrSim 8302 --channels SRS or ./ru_emulator nrSim 8302 (default
↳support all channels)
sudo ./test_mac nrSim 8302 --channels SRS or ./test_mac nrSim 8302 (default support
↳all channels)
sudo -E ./cuphycontroller_scf nrSim_SCF
# Expect the testMac to report the number of received SRS is between 97 and 103 and INV
↳values per second to be 0.
# If the INV Values are greater than 0, there is either a SRS report mismatch or SRS
↳report parameter mismatch.

```

#### 1.5.4.5.4.15 S-slot

```

# Restart MPS
sudo ./ru_emulator nrSim 90013 --channels 0x1ff
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 90013 --channels 0x1ff
# Expect RU Emulator to report 50 DL and PDCCH_DL per second, testMAC to report 50 HARQ
↳per second

# Restart MPS
sudo ./ru_emulator nrSim 90015 --channels 0x1ff
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 90015 --channels 0x1ff
# Expect RU Emulator to report 50 DL and PDCCH_DL per second, testMAC to report 50 HARQ
↳per second

```

#### 1.5.4.5.4.16 Multiple SSB

```
# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./test_mac nrSim 1104 --channels PBCH
sudo ./ru_emulator nrSim 1104 --channels PBCH
# Expect RU Emulator to report 100 PBCH per second
```

#### 1.5.4.5.4.17 PUSCH TDI

```
# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF_tdi
sudo ./test_mac nrSim 7411 --channels PUSCH
sudo ./ru_emulator nrSim 7411 --channels PUSCH
# Expect testMAC and RU Emulator both see 1.79 Mbps 100 Slots per second
```

#### 1.5.4.5.4.18 PUSCH SINR and Noise

```
# For TCs 7265,7266,7268,7269,7271,7272
# Change cuphycontroller_nrSim_SCF.yaml file to have 8 eAxCIds for PUSCH
eAxC_id_pusch:      [8,0,1,2,3,4,5,6]
sed -i s/"eAxC_id_pusch:      \\[8,0,1,2\]/eAxC_id_pusch:      \\[8,0,1,2,3,4,5,6\]
↪]/1" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml

# For TCs 7264,7267,7270 no change to cuphycontroller_nrSim_SCF.yaml
# Restart MPS
sudo ./test_mac nrSim 7265 --channels PUSCH
sudo ./ru_emulator nrSim 7265 --channels PUSCH
# Revert if changed earlier
sed -i s/"eAxC_id_pusch:      \\[8,0,1,2,3,4,5,6\]/eAxC_id_pusch:      \\[8,0,1,2\]
↪]/1" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml
```

#### 1.5.4.5.4.19 mSlot\_mCell Test Cases

TCs 90001-90006,90011-90015, 90061-90063, 90700-90705 - TDD Pattern DSUUU TCs: 90061,90062, 90063 - Multi-Slot SRS TCs: 90700, 90701, 90702, 90703, 90705

```
# nrSim config generation
cd ${cuBB_SDK}/cubb_scripts/autoconfig
python3 auto_controllerConfig.py -i ../../testVectors/ -t ../../cuPHY-CP/
↪cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml -o ../../cuPHY-CP/
↪cuphycontroller/config
python3 auto_RuEmulatorConfig.py -i ../../cuPHY-CP/cuphycontroller/config -t ../../
↪cuPHY-CP/ru-emulator/config/config.yaml -o ../../cuPHY-CP/ru-emulator/config

# backup default nrSim config
cp ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml ${cuBB_
↪SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml.orig
cp ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml ${cuBB_SDK}/cuPHY-CP/ru-
↪emulator/config/config.yaml.orig
```

(continues on next page)

(continued from previous page)

```

cp ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config.yaml ${cuBB_SDK}/cuPHY-CP/
↪testMAC/testMAC/test_mac_config.yaml.orig

# Use nrSim_SCF_900xx config
cp ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF_900xx.yaml $
↪{cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml
cp ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/ru_emulator_config_900xx.yaml ${cuBB_SDK}/
↪cuPHY-CP/ru-emulator/config/config.yaml
python3 auto_TestMacConfig.py -t ../../cuPHY-CP/testMAC/testMAC/test_mac_config.yaml.
↪orig -c 900xx -p CG1 -o ../../cuPHY-CP/testMAC/testMAC/test_mac_config.yaml

#For TCs 90061-90063, set pusch_aggr_per_ctx to 9, prach_aggr_per_ctx to 4 and ul_input_
↪buffer_per_cell to 15 in cuphycontroller_nrSim_SCF.yaml
sed -i "s/ pusch_aggr_per_ctx:.* / pusch_aggr_per_ctx: 9/" ${cuBB_SDK}/cuPHY-CP/
↪cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml
sed -i "s/ prach_aggr_per_ctx:.* / prach_aggr_per_ctx: 4/" ${cuBB_SDK}/cuPHY-CP/
↪cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml
sed -i "s/ ul_input_buffer_per_cell:.* / ul_input_buffer_per_cell: 15/" ${cuBB_SDK}/
↪cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml

# For TCs, 90703 and 90704, set prach_aggr_per_ctx to 4 and ul_input_buffer_per_cell to
↪12 in cuphycontroller_nrSim_SCF.yaml
sed -i "s/ prach_aggr_per_ctx:.* / prach_aggr_per_ctx: 4/" ${cuBB_SDK}/cuPHY-CP/
↪cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml
sed -i "s/ ul_input_buffer_per_cell:.* / ul_input_buffer_per_cell: 12/" ${cuBB_SDK}/
↪cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml

# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./ru_emulator nrSim 900xx --channels 0x1ff
sudo ./test_mac nrSim 900xx --channels 0x1ff

# Restore nrSim config file
cp ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml.orig $
↪{cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml
cp ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml.orig ${cuBB_SDK}/cuPHY-CP/ru-
↪emulator/config/config.yaml
cp ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config.yaml.orig ${cuBB_SDK}/cuPHY-CP/
↪testMAC/testMAC/test_mac_config.yaml

```

#### 1.5.4.5.4.20 64T64R SRS + Dynamic Beamforming Weights + Static Beamforming Weights Test Cases

Here are the steps to build and run the 64T6R SRS and dynamic beamforming weights related tests.

Build options:

```

cmake -Bbuild -GNinja -DCMAKE_TOOLCHAIN_FILE=cmake/toolchains/native -DSCF_FAPI_10_
↪04=ON -DENABLE_STATIC_BFW=ON
cmake --build build

```

Verify all of the following launch patterns for Dynamic DL-BFW+PDSCH, Dynamic UL-BFW+PUSCH, All channels Static+Dynamic Beamforming Weight:

Basic full allocation:

1. 100 MHz DL 16 Layers (1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1) - 90090
2. 100 MHz UL 8 Layers 1+1+1+1+1+1+1+1 layer - 90091
3. 100 MHz DL 8 PDU SRS - 8514
4. 100 MHz DL 1 layer - 90092
5. 100 MHz UL 1 layer - 90093
6. 100 MHz DL 2 layers - 90094
7. 100 MHz UL 2 layers - 90095
8. 100 MHz DL 1+1 layers - 90096
9. 100 MHz UL 1+1 layers - 90097
10. 100 MHz DL 2+2 layers - 90098
11. 100 MHz UL 2+2 layers - 90099
12. 100 MHz DL 1+1+1+1 layers - 90100
13. 100 MHz UL 1+1+1+1 layers - 90101
14. 100 MHz DL 2+2+2+2 layers - 90102
15. 100 MHz UL 1+1+1+1+1+1+1+1 layers - 90110
16. 100 MHz DL 2+2+2+2+2+2+2+2 layers - 90111
17. 100 MHz UL 2+2+2+2 layers - 90112
18. 100 MHz DL 4+4+4+4 layers - 90113
19. 100 MHz DL 16 Layers (1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1) + UL 8 Layers (1+1+1+1+1+1+1+1) + SRS - 90103
20. 100 MHz DL 1 layer + UL 1 layer + SRS - 90104
21. 100 MHz DL 2 layers + UL 2 layers + SRS - 90105
22. 100 MHz DL 1+1 layers + UL 1+1 layer + SRS - 90106
23. 100 MHz DL 2+2 layers + UL 2+2 layers + SRS - 90107
24. 100 MHz DL 1+1+1+1 layers + UL 1+1+1+1 layers + SRS - 90108
25. 100 MHz DL 2+2+2+2 layers + UL 1+1+1+1 layers + SRS - 90109
26. 100 MHz UL 1+1+1+1+1+1+1+1 layers + DL 2+2+2+2+2+2+2+2 layers + SRS - 90114
27. 100 MHz UL 2+2+2+2 layer + DL 4+4+4+4 layers + SRS - 90115

Multiple UE groups with one BWP:

1. 100 MHz UL 2 UE grps, same layers, prb sizes, start prb - 90116
2. 100 MHz DL 2 UE grps, same layers, prb sizes, start prb - 90117
3. 100 MHz UL 2 UE grps. Different prb sizes. - 90118
4. 100 MHz DL 2 UE grps. Different prb sizes. - 90119
5. 100 MHz UL 2 UE grps. Different start prbs. - 90120
6. 100 MHz DL 2 UE grps. Different start prbs. - 90121
7. 100 MHz UL 2 UE grps. Different layers. - 90122



8. 100 MHz DL 2 UE grps. Different layers. - 90123
9. 100 MHz UL 2 UE grps. All prms different. - 90124
10. 100 MHz DL 2 UE grps. All prms different. - 90125
11. 100 MHz UL 4 UE grps. all different layers. - 90126
12. 100 MHz DL 4 UE grps. all different layers. - 90127
13. 100 MHz UL 8 UE grps - 90128
14. 100 MHz DL 8 UE grps - 90129
15. 100 MHz UL + DL 2 UE grps, same layers, prb sizes, start prb + SRS - 90130
16. 100 MHz UL + DL 2 UE grps. Different prb sizes + SRS - 90131
17. 100 MHz UL + DL 2 UE grps. Different start prbs. + SRS - 90132
18. 100 MHz UL + DL 2 UE grps. Different layers. + SRS - 90133
19. 100 MHz UL + DL 2 UE grps. All prms different. + SRS - 90134
20. 100 MHz UL + DL 4 UE grps. all different layers. + SRS - 90135
21. 100 MHz UL + DL 8 UE grps + SRS - 90136

Flexible PRG size/PRB number:

1. 100 MHz DL 8 layer partial PRB allocation - 90137
2. 100 MHz UL 4 layer partial PRB allocation - 90138
3. 100 MHz DL partial PRB allocation - 90139
4. 100 MHz UL partial PRB allocation - 90140
5. 100 MHz DL 8 layer + UL 4 layer partial PRB allocation + SRS - 90143
6. 100 MHz DL + UL partial PRB allocation + SRS - 90144

Multiple PRG sizes:

1. 100MHz 64 PRBs, prgSize=4 UL - 90141
2. 100MHz 64 PRBs, prgSize=4 DL - 90142
3. prgSize\_SRS != prgSize\_BFW DL - 90146
4. prgSize\_SRS != prgSize\_BFW UL - 90147
5. 100MHz 64 PRBs, prgSize=4 UL + DL + SRS - 90145
6. prgSize\_SRS != prgSize\_BFW DL + UL + SRS - 90148

Dynamic + Static Beamforming:

1. All Channels UEGO (static, 1 UE) - 90606
2. All Channels UEGO (static, 1 UE) + UEG1 (static, 1 UE) - 90607
3. All Channels UEGO (dynamic, 4 UEs) + UEG1 (dynamic, 1 UE) + UEG2 (static, 1 UE) - 90608
4. All Channels UEGO (dynamic, 1 UE) + UEG1 (static, 1 UE) - 90609
5. All Channels UEGO (dynamic, 2 UEs) + UEG1 (static, 1 UE) - 90610
6. All Channels DL(dynamic, 1x8 UEs) + UL (dynamic 2x1 UEs) - 90611
7. All Channels Static+Dynamic Config - 90612

8. PDSCH + CSIRS (nPrb =< 255, sym 0) - 90613
9. PDSCH + CSIRS (nPrb > 255, sym 0) - 90614
10. PDSCH + CSIRS (nPrb > 255, sym 6) - 90615
11. PDSCH + CSIRS (prgSize = 273, nPrg = 1) - 90616
12. All Channels - SRS on UL slot 3 & 4 - 90620
13. All Channels - SRS on UL slot 3 & 5 - 90621
14. All Channels - SRS on UL slot 3, 4 & 5 - 90622
15. 2 multi User MIMO Cells - All Channels UEG0 (dynamic, 4 UEs) + UEG1 (dynamic, 1 UE) + UEG2 (static, 1 UE) - 90630
16. 3 multi User MIMO Cells - All Channels UEG0 (dynamic, 4 UEs) + UEG1 (dynamic, 1 UE) + UEG2 (static, 1 UE) - 90631
17. 2 multi User MIMO Cells - All Channels Cell 1 (No BFW) + Cell 2 (BFW) - 90632
18. 2 multi User MIMO Cells - All Channels Cell 1 (No BFW) + Cell 2 (No BFW) + Cell 3 (BFW) - 90633
19. 3 multi User MIMO Cells - 100 MHz DL 16 Layers (1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1) + UL 8 Layers (1+1+1+1+1+1+1+1) + SRS - 90634

For 64T64R SRS 85xx, the TV's need to be executed. You can generate the config using the autoconfig scripts for the above launch patterns, with the exception that only the following parameters need to be explicitly modified in the generated config file:

```
In cuphycontroller_nrSim_SCF.yaml - enable_srs: 1, mMIMO_enable: 1, mtu: 8192, cpu_
↳mbufs: 196608
In cuphycontroller_nrSim_SCF_CG1.yaml - enable_srs: 1, mMIMO_enable: 1, mtu: 8192,
↳cpu_mbufs: 196608
In ru-emulator: config.yaml - aerial_fh_mtu: 8192, enable_mmimo: 1
```

```
# nrSim config generation
cd ${cuBB_SDK}/cubb_scripts/autoconfig
python3 auto_controllerConfig.py -i ../../testVectors/ -t ../../cuPHY-CP/
↳cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml -o ../../cuPHY-CP/
↳cuphycontroller/config
python3 auto_RuEmulatorConfig.py -i ../../cuPHY-CP/cuphycontroller/config -t ../../
↳cuPHY-CP/ru-emulator/config/config.yaml -o ../../cuPHY-CP/ru-emulator/config

# backup default nrSim config
cp ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml ${cuBB_
↳SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml.orig
cp ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml ${cuBB_SDK}/cuPHY-CP/ru-
↳emulator/config/config.yaml.orig
cp ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config.yaml ${cuBB_SDK}/cuPHY-CP/
↳testMAC/testMAC/test_mac_config.yaml.orig

# Use nrSim_SCF_900xx config
cp ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF_900xx.yaml $
↳{cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml
cp ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/ru_emulator_config_900xx.yaml ${cuBB_SDK}/
↳cuPHY-CP/ru-emulator/config/config.yaml
python3 auto_TestMacConfig.py -t ../../cuPHY-CP/testMAC/testMAC/test_mac_config.yaml.
↳orig -c 900xx -p CG1 -o ../../cuPHY-CP/testMAC/testMAC/test_mac_config.yaml

# Restart MPS
```

(continues on next page)

(continued from previous page)

```

sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./ru_emulator nrSim 90xxx --channels 0x7ff
sudo ./test_mac nrSim 90xxx --channels 0x7ff

# Restore nrSim config file
cp ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml.orig $
↪ ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml
cp ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml.orig ${cuBB_SDK}/cuPHY-CP/ru-
↪ emulator/config/config.yaml
cp ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config.yaml.orig ${cuBB_SDK}/cuPHY-CP/
↪ testMAC/testMAC/test_mac_config.yaml

```

### 1.5.4.5.5 FAPI Message Reference Check

The cuBB software supports the FAPI message reference check. The values and payloads of RX\_DATA.ind, CRC.ind, UCI.ind, and RACH.ind are compared with the related INDx PDU of the TV. If validation fails, a “mismatch” WARN level log is printed to testmac.log by testMAC.

#### Note

Some validation failures are not fixed yet. The current known validation failures are not reported with “INV > 0” by default.

The following configurations are implemented to configure test\_mac reporting. The default configuration for FAPI validation is as follows:

```

# FAPI indication validating
# validate_enable: 0 - disabled; 1 - report error level; 2 - report error and warning
↪ levels
validate_enable: 1
# validate_log_opt: 0 - no print; 1 - print per MSG; 2 - print per PDU; 3 - force print
↪ all
validate_log_opt: 1

```

The following is an example validation failure log with default configuration:

```

09:35:02.205513 W [MAC.FAPI] SFN 0.5 Cell 6 CRC.ind mismatch: 0 err 6 warn [crc->num_
↪ cb=192 tv.NumCb=4] [meas->u1_cqi=255 tv.UL_CQI=206] [meas->rssi=65535 tv.RSSI=880]

```

One FAPI message can contain multiple PDUs, and one PDU can contain multiple validation failures.

- ▶ Set “validate\_enable: 1” to report only some validation failures with “INV > 0” in test\_mac console. Known validation failures are not reported with “INV > 0” (but can still be seen in the “mismatch” WARN log).
- ▶ Set “validate\_enable: 2” to report all validation failures with “INV > 0” in test\_mac console.
- ▶ Set “validate\_log\_opt: 1” to print one line “mismatch” log with at most three mismatched values per FAPI message, and print the total mismatched PDU count (e.g. “0 err, 6 warn”) per FAPI message (avoids performance dropping).
- ▶ Set “validate\_log\_opt: 2” to print all validation failures in the “mismatch” WARN log, one line per PDU.

Example log with “validate\_log\_opt: 2”:

```
07:32:09.407972 W [MAC.FAPI] SFN 0.14 Cell 0 CRC.ind PDU0 mismatch: [crc->num_cb=0 tv.
↪NumCb=5] [meas->ul_cqi=255 tv.UL_CQI=206] [meas->rssi=65535 tv.RSSI=1280]
07:32:09.407976 W [MAC.FAPI] SFN 0.14 Cell 0 CRC.ind PDU1 mismatch: [crc->num_cb=0 tv.
↪NumCb=5] [meas->ul_cqi=255 tv.UL_CQI=206] [meas->rssi=65535 tv.RSSI=1280]
07:32:09.407979 W [MAC.FAPI] SFN 0.14 Cell 0 CRC.ind PDU2 mismatch: [crc->num_cb=0 tv.
↪NumCb=5] [meas->ul_cqi=255 tv.UL_CQI=206] [meas->rssi=65535 tv.RSSI=1280]
```

The current recommended test instructions:

- ▶ Use the default configuration to test, then `grep "mismatch"` in `phy.log` to check whether there is a validation failure.
- ▶ Configure `"validate_log_opt: 2"` to print all validation failures, if required.

#### 1.5.4.5.6 Running testMAC + cuPHYController\_SCF + RU Emulator P5G PRACH

This use case runs the Private 5G SIB1 and PRACH demo msg1-4 between the RU Emulator and the testMAC.

You need additional modifications to the default `cuPHYController_P5G_GH.yaml` to test against RU emulator. Ensure it matches the configs here. You must also set the PCIe NIC address that is currently in use:

##### 1.5.4.5.6.1 Server#1

```
sed -i "s/dl_iq_data_fmt.*/dl_iq_data_fmt: {comp_meth: 1, bit_width: 16}/" ${cuBB_SDK}/
↪cuPHY-CP/cuphycontroller/config/cuphycontroller_P5G_GH.yaml
sed -i "s/ul_iq_data_fmt.*/ul_iq_data_fmt: {comp_meth: 1, bit_width: 16}/" ${cuBB_SDK}/
↪cuPHY-CP/cuphycontroller/config/cuphycontroller_P5G_GH.yaml
sed -i "s/pcp.*/pcp: 7/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_
↪P5G_GH.yaml
sed -i "0,/dst_mac_addr.*/{s/dst_mac_addr: 20:04:9B:9E:27:A3}/" ${cuBB_
↪SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_P5G_GH.yaml
sed -i "s/enableTickDynamicSfnSlot.*/enableTickDynamicSfnSlot: 0/" ${cuBB_SDK}/cuPHY-
↪CP/cuphycontroller/config/l2_adapter_config_P5G_GH.yaml
```

##### 1.5.4.5.6.2 Server#2

Replace `0000:b5:00.0` with the PCIe address of the NIC to use. Also, replace the MAC address with the MAC address of the NIC used in *Server#1* (the server running `cuPHYController` and `testMAC`):

```
sed -i "s/nic_interface.*/nic_interface: 0000:b5:00.0/" ${cuBB_SDK}/cuPHY-CP/ru-
↪emulator/config/config.yaml
```

Change the `dl_iq_data_fmt/ul_iq_data_fmt` to BFP 16. Ensure you change it back to BFP 14 for other tests.

```
sed -i "s/dl_iq_data_fmt.*/dl_iq_data_fmt: {comp_meth: 1, bit_width: 16}/" ${cuBB_SDK}
↪/cuPHY-CP/ru-emulator/config/config.yaml
sed -i "s/ul_iq_data_fmt.*/ul_iq_data_fmt: {comp_meth: 1, bit_width: 16}/" ${cuBB_SDK}
↪/cuPHY-CP/ru-emulator/config/config.yaml
sed -i "s/eAxC_DL: \[8,0,1,2\]/eAxC_DL: \[0,8,1,9\]/1" ${cuBB_SDK}/cuPHY-CP/ru-
```

(continues on next page)

(continued from previous page)

```

↪emulator/config/config.yaml
sed -i "s/eAxC_UL: \[8,0,1,2\]/eAxC_UL: \[0,8,1,9\]/1" ${cuBB_SDK}/cuPHY-CP/ru-
↪emulator/config/config.yaml
sed -i "s/eAxC_prach_list: \[15,7,0,1\]/eAxC_prach_list: \[7,15,6,14\]/1" ${cuBB_SDK}/
↪cuPHY-CP/ru-emulator/config/config.yaml

```

Run the emulator:

```
sudo ./ru_emulator P5G PRACH --channels 0x1FF
```

Run the cuPHY controller and the testMAC:

```
sudo -E ./cuphycontroller_scf P5G_GH
sudo ./test_mac P5G PRACH --channels 0x1FF
```

Expected RU emulator console:

```

00:44:12.169849 Cell 0 DL 0.17 Mbps 100 Slots | UL 0.03 Mbps 50 Slots | PBCH
↪ 50 | PDCCH_UL 0 | PDCCH_DL 150 | PRACH 50 Slots | Seconds 25
00:44:13.169848 Cell 0 DL 0.17 Mbps 100 Slots | UL 0.03 Mbps 50 Slots | PBCH
↪ 50 | PDCCH_UL 0 | PDCCH_DL 150 | PRACH 50 Slots | Seconds 26
00:44:14.169849 Cell 0 DL 0.17 Mbps 100 Slots | UL 0.03 Mbps 50 Slots | PBCH
↪ 50 | PDCCH_UL 0 | PDCCH_DL 150 | PRACH 50 Slots | Seconds 27

```

Expected testMAC console:

```

00:44:11.565232 Cell 0 | DL 0.26 Mbps 150 Slots | UL 0.03 Mbps 50 Slots |
↪ Prmb 50 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
00:44:12.565230 Cell 0 | DL 0.26 Mbps 150 Slots | UL 0.03 Mbps 50 Slots |
↪ Prmb 50 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
00:44:13.565230 Cell 0 | DL 0.26 Mbps 150 Slots | UL 0.03 Mbps 50 Slots |
↪ Prmb 50 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0

```

Expected cuPHYController logs to be flooded with preamble detection:

```

00:44:11.565224 C [SCF.PHY] Cell 0 | DL 0.26 Mbps 150 Slots | UL 0.03 Mbps
↪ 50 Slots CRC 0 ( 0 ) | Tick 2000
00:44:12.565224 C [SCF.PHY] Cell 0 | DL 0.26 Mbps 150 Slots | UL 0.03 Mbps
↪ 50 Slots CRC 0 ( 0 ) | Tick 4000
00:44:13.565224 C [SCF.PHY] Cell 0 | DL 0.26 Mbps 150 Slots | UL 0.03 Mbps
↪ 50 Slots CRC 0 ( 0 ) | Tick 6000

```

#### 1.5.4.5.7 Running End-to-End with Full Stack

This section provides a guide on reference cuPHYController YAML to be used when using Aerial CUDA-Accelerated RAN with Full Stack application.

When running full stack Aerial CUDA-Accelerated RAN on Grace Hopper, use the following file as a starting point to be modified according to your lab configuration.

1. When using Keysight RU-SIM as a Radio Unit, use `cuphycontroller_P5G_GH.yaml` as a reference.
2. When using Foxconn O-RU as a Radio Unit, use `cuphycontroller_P5G_FXN_GH.yaml` as a reference.

**Note**

You need to modify the above mentioned reference files based on to your End-to-End setup.

**1.5.4.5.8 Capture Logs**

Collect the text logs after testing.

1. By default, the logs get stored in the /tmp location. You can use the AERIAL\_LOG\_PATH environment variable to set the logfile path.
2. When the log size exceeds 50GB, a new file gets created (e.g. phy.log, phy.log.1, phy.log.2 ... phy.log.7).
  - a. The test MAC logs are named as testmac.log, testmac.log.1, etc.
  - b. The RU logs are named as ru.log, ru.log.1, etc.
3. These file segments are reused in a cyclic manner by overwriting the oldest files.

For SHM IPC, if you see the IPC buffer pool full during testing, run the following command to dump IPC status after test:

```
# For SHM IPC, dump nvipc message queues after test
sudo ./build/cuPHY-CP/gt_common_libs/nvIPC/tests/dump/ipc_dump

# If not using default nvipc configurations, need input the nvipc "prefix" and yaml
↪ config file like below.
# For Multi-L2 case, the "prefix" names are different for each L2 instance, see related
↪ nvipc config yaml files.
sudo ./build/cuPHY-CP/gt_common_libs/nvIPC/tests/dump/ipc_dump nvipc ./cuPHY-CP/
↪ cuphycontroller/config/l2_adapter_config_F08_CG1.yaml
```

To capture PCAP log, please run “export NVIPC\_DEBUG\_EN=1” in command line of cuphycontroller or config below pcap\_enable=1. Max size limitation of PCAP logs can be configured in the yaml file.

```
# Transport settings for nvIPC
transport:
  type: shm
  app_config:
    pcap_enable: 1
    pcap_cpu_core: -1 # CPU core of background pcap log save thread
    pcap_cache_size_bits: 29 # 2^29 = 512MB, size of /dev/shm/${prefix}_pcap
    pcap_file_size_bits: 31 # 2^31 = 2GB, max size of /dev/shm/${prefix}_pcap
    pcap_max_data_size: 8000 # Max DL/UL FAPI data size to capture reduce pcap size.
```

After cuphycontroller exits, run below command with nvipc “prefix” to collect PCAP logs. The “prefix” comes from l2adapter config yaml. It is “nvipc” by default. In Multi-L2 case, different L2 instance has different “prefix” names.

```
# Usage: sudo ./pcap_collect <prefix> [destination path]
sudo $cuBB_SDK/build/cuPHY-CP/gt_common_libs/nvIPC/tests/pcap/pcap_collect nvipc

# The nvipc.pcap can be seen at current directory (by default) or the inputed
↪ "destination path".
```

### 1.5.4.6 Run in Test Mode (TM)

To run any test where at least one cell is in Test Mode (TM), you need to ensure cmake was run with `-DENABLE_CONFORMANCE_TM_PDSCH_PDCCH=ON`.

This option is needed for nrSIM TCs 2036 (*PDCCH*) and 3296 (*PDSCH*), as well as for DLMIX TCs 120-128 with both PDSCH and PDCCH present. This requirement extends to any multi-cell launch pattern with at least one of these TM test vectors present.

For test cases where no TM cell is present, the cmake option value is not relevant for the functional correctness of cuBB tests.

### 1.5.4.7 Mixed O-RAN IOT Profiles (CAT-A-NoBF + CAT-A-DBF)

To run mixed one cell with CAT-A-NoBF and another cell with CAT-A-DBF, use the nrSIM TC 90019 and run the following:

```
# nrSim config generation
cd ${cuBB_SDK}/cubb_scripts/autoconfig
python3 auto_controllerConfig.py -i ../../testVectors/ -t ../../cuPHY-CP/
↳cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml -o ../../cuPHY-CP/
↳cuphycontroller/config
python3 auto_RuEmulatorConfig.py -i ../../cuPHY-CP/cuphycontroller/config -t ../../
↳cuPHY-CP/ru-emulator/config/config.yaml -o ../../cuPHY-CP/ru-emulator/config

# backup default nrSim config
cp ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml ${cuBB_
↳SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml.orig
cp ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml ${cuBB_SDK}/cuPHY-CP/ru-
↳emulator/config/config.yaml.orig
cp ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config.yaml ${cuBB_SDK}/cuPHY-CP/
↳testMAC/testMAC/test_mac_config.yaml.orig

# Use nrSim_SCF_90019 config
cp ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF_90019.yaml $
↳{cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml
cp ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/ru_emulator_config_90019.yaml ${cuBB_SDK}/
↳cuPHY-CP/ru-emulator/config/config.yaml
python3 auto_TestMacConfig.py -t ../../cuPHY-CP/testMAC/testMAC/test_mac_config.yaml.
↳orig -c 90019 -p CG1 -o ../../cuPHY-CP/testMAC/testMAC/test_mac_config.yaml

# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./ru_emulator nrSim 90019 --channels 0x1ff
sudo ./test_mac nrSim 90019 --channels 0x1ff

# Restore nrSim config file
cp ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml.orig $
↳{cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml
cp ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml.orig ${cuBB_SDK}/cuPHY-CP/ru-
↳emulator/config/config.yaml
cp ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config.yaml.orig ${cuBB_SDK}/cuPHY-CP/
↳testMAC/testMAC/test_mac_config.yaml
```

Expected result:



```

# Expected Tput and passing criteria
Expected thrtput: Cell 0: [DL=1.36/100]
Expected thrtput: Cell 1: [DL=2.72/100]
Pass criterion low: Cell 0: [DL=1.31/97]
Pass criterion high: Cell 0: [DL=1.40/103]
Pass criterion low: Cell 1: [DL=2.63/97]
Pass criterion high: Cell 1: [DL=2.80/103]

# Example ru-emulator output
16:24:15.218189 Cell 0 DL 1.36 Mbps 100 Slots | UL 0.00 Mbps 0 Slots | DL_C_
↪ON 100.00% DL_U_ON 100.00% UL_C_ON 0.00% |Seconds 45
16:24:15.218201 Cell 1 DL 2.72 Mbps 100 Slots | UL 0.00 Mbps 0 Slots | DL_C_
↪ON 100.00% DL_U_ON 100.00% UL_C_ON 0.00% |Seconds 45
16:24:16.218191 Cell 0 DL 1.36 Mbps 100 Slots | UL 0.00 Mbps 0 Slots | DL_C_
↪ON 100.00% DL_U_ON 100.00% UL_C_ON 0.00% |Seconds 46
16:24:16.218204 Cell 1 DL 2.72 Mbps 100 Slots | UL 0.00 Mbps 0 Slots | DL_C_
↪ON 100.00% DL_U_ON 100.00% UL_C_ON 0.00% |Seconds 46

```

### 1.5.4.8 Mixed BFP9/BFP14

```

# nrSim config generation
cd ${cuBB_SDK}/cubb_scripts/autoconfig
python3 auto_controllerConfig.py -i ../../testVectors/ -t ../../cuPHY-CP/
↪cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml -o ../../cuPHY-CP/
↪cuphycontroller/config
python3 auto_RuEmulatorConfig.py -i ../../cuPHY-CP/cuphycontroller/config -t ../../
↪cuPHY-CP/ru-emulator/config/config.yaml -o ../../cuPHY-CP/ru-emulator/config

# backup default nrSim config
cp ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml ${cuBB_
↪SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml.orig
cp ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml ${cuBB_SDK}/cuPHY-CP/ru-
↪emulator/config/config.yaml.orig
cp ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config.yaml ${cuBB_SDK}/cuPHY-CP/
↪testMAC/testMAC/test_mac_config.yaml.orig

# Use nrSim_SCF_90020 config
cp ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF_90020.yaml $
↪${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml
cp ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/ru_emulator_config_90020.yaml ${cuBB_SDK}/
↪cuPHY-CP/ru-emulator/config/config.yaml
python3 auto_TestMacConfig.py -t ../../cuPHY-CP/testMAC/testMAC/test_mac_config.yaml.
↪orig -c 90020 -p CG1 -o ../../cuPHY-CP/testMAC/testMAC/test_mac_config.yaml

# Restart MPS
sudo -E ./cuphycontroller_scf nrSim_SCF
sudo ./ru_emulator nrSim 90020 --channels 0x1ff
sudo ./test_mac nrSim 90020 --channels 0x1ff

# Restore nrSim config file
cp ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml.orig $
↪${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_nrSim_SCF.yaml
cp ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml.orig ${cuBB_SDK}/cuPHY-CP/ru-
↪emulator/config/config.yaml
cp ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config.yaml.orig ${cuBB_SDK}/cuPHY-CP/
↪testMAC/testMAC/test_mac_config.yaml

```



Expected result:

```
# Expected throughput and passing criteria
ExpectedSlots: Cell=0 PUSCH=100 PDSCH=0 PDCCH_UL=0 PDCCH_DL=0 PBCH=0 PUCCH=0 PRACH=0
↳CSI_RS=0 SRS=0
ExpectedData: Cell=0 DL=0.000000 UL=41.797600 Prmb=0 HARQ=0 SR=0 CSI1=0 CSI2=0 ERR=0
↳INV=0
ExpectedSlots: Cell=1 PUSCH=100 PDSCH=0 PDCCH_UL=0 PDCCH_DL=0 PBCH=0 PUCCH=0 PRACH=0
↳CSI_RS=0 SRS=0
ExpectedData: Cell=1 DL=0.000000 UL=41.797600 Prmb=0 HARQ=0 SR=0 CSI1=0 CSI2=0 ERR=0
↳INV=0

# Example testMAC output
07:09:34.600006 Cell 0 | DL 0.00 Mbps 0 Slots | UL 41.80 Mbps 100 Slots |
↳Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
07:09:34.600015 Cell 1 | DL 0.00 Mbps 0 Slots | UL 41.80 Mbps 100 Slots |
↳Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
07:09:35.600006 Cell 0 | DL 0.00 Mbps 0 Slots | UL 41.80 Mbps 100 Slots |
↳Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
07:09:35.600014 Cell 1 | DL 0.00 Mbps 0 Slots | UL 41.80 Mbps 100 Slots |
↳Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
07:09:36.600006 Cell 0 | DL 0.00 Mbps 0 Slots | UL 41.80 Mbps 100 Slots |
↳Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
07:09:36.600013 Cell 1 | DL 0.00 Mbps 0 Slots | UL 41.80 Mbps 100 Slots |
↳Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
07:09:37.600008 Cell 0 | DL 0.00 Mbps 0 Slots | UL 41.80 Mbps 100 Slots |
↳Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
07:09:37.600017 Cell 1 | DL 0.00 Mbps 0 Slots | UL 41.80 Mbps 100 Slots |
↳Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
07:09:38.600006 Cell 0 | DL 0.00 Mbps 0 Slots | UL 41.80 Mbps 100 Slots |
↳Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
07:09:38.600014 Cell 1 | DL 0.00 Mbps 0 Slots | UL 41.80 Mbps 100 Slots |
↳Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
07:09:39.600008 Cell 0 | DL 0.00 Mbps 0 Slots | UL 41.80 Mbps 100 Slots |
↳Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
```

#### 1.5.4.9 Mixed IQ data format for F08 Test Case

Here is an example to run the mixed compression using F08 test case for 1 16-bit Fixed point + 1 BFP9 + N BFP 14 cells, where N = 1,2,3. Set the value for dl\_iq\_data\_fmt and ul\_iq\_data\_fmt to 16-bit fixed point for the 1st cell and BFP 9 for the 2nd cell in both the cuPHYController\_F08\_\*.yaml file and RU emulator config.yaml file. Set the value for dl\_iq\_data\_fmt and ul\_iq\_data\_fmt to BFP 14 for all other cells.

```
# First cell
dl_iq_data_fmt: {comp_meth: 0, bit_width: 16}
ul_iq_data_fmt: {comp_meth: 0, bit_width: 16}

# Second cell
dl_iq_data_fmt: {comp_meth: 1, bit_width: 9}
ul_iq_data_fmt: {comp_meth: 1, bit_width: 9}

# All other cells
dl_iq_data_fmt: {comp_meth: 1, bit_width: 14}
ul_iq_data_fmt: {comp_meth: 1, bit_width: 14}
```

The throughput levels must be the same as non-mixed case with only BFP 14.

```

16:17:05.609792 C [SCF.PHY] Cell 0 | DL 1586.28 Mbps 1600 Slots | UL 249.10 Mbps
↪400 Slots CRC 0 ( 0) | Tick 16000
16:17:05.609808 C [SCF.PHY] Cell 1 | DL 1586.28 Mbps 1600 Slots | UL 249.10 Mbps
↪400 Slots CRC 0 ( 0) | Tick 16000
16:17:05.609814 C [SCF.PHY] Cell 2 | DL 1586.28 Mbps 1600 Slots | UL 249.10 Mbps
↪400 Slots CRC 0 ( 0) | Tick 16000
16:17:05.609822 C [SCF.PHY] Cell 3 | DL 1586.28 Mbps 1600 Slots | UL 249.10 Mbps
↪400 Slots CRC 0 ( 0) | Tick 16000
    
```

### 1.5.4.10 UL Measurements

To enable UL measurements in PHY, set the the following to 1 in `cuphycontroller_nrSim_SCF.yaml` and all measurements are in dBm unit.

```

pusch_sinr: 1 # 0 - Disabled; 1 - PostEq value; 2 - PreEq value
pusch_rssi: 1
pusch_tdi: 1
pusch_cfo: 1
    
```

#### Note

From release 22-2.3 onwards, SINR reporting can be configured to report pre- or post-equalizer values from the `cuphycontroller_nrSim_SCF.yaml` file, as shown above.

To enable FAPI 10.04 fields, add `-DSCF_FAPI_10_04=ON` in the `cmake` options and do a clean build.

To enable RSSI and RSRP measurements, L2 has to send Measurement Config TLV in `config.request` with a value of 1 for dBm as described in table 3-27 of FAPI 10.02 and table 3-48 of FAPI 10.04. To enable the same in `testMac`:

- ▶ RSSI is enabled by default.
- ▶ For RSRP, set the following to 1 in the `$cuBB_SDK/testMAC/testMAC/test_mac_config.yaml` file:

```
rsrpMeasurement: 1
```

L2 vendors have requested additional interference level reporting for PUSCH, UCI on PUSCH, and PUCCH (PF2,3 only supported). For this purpose, vendor specific messages have been defined to indicate the Aerial instance that reports these measurements. To enable this reporting, L2 has to send 2 additional TLVs in `config.request` as mentioned in `CONFIG.request`.

Tag	Field	Type	Description
0xA012	PNMeasurement	uint8_t	Post equalisation noise variance measurement Value: 0: Do not report 1: dBm
0xA014	PF_234_Interference	uint8_t	Interference power per UE. Value: 0: Do not report 1: dBm

After it is enabled, for every `CRC.indication`, Aerial sends an additional `RX_PE_Noise_Variance.indication`. For every `UCI.indication` carrying PF2,3, Aerial sends a `PF_234_Interference.indication`.

To enable interference reporting in testMac, set the following to 1 in the \$cuBB\_SDK/testMAC/testMAC/test\_mac\_config.yaml file:

```
pf_234_interference: 1
pnMeasurement: 1
```

Enable DEBUG level log for tag SCF.PHY as follows in the cuPHY/nvlog/config/nvlog\_config.yaml file:

```
- 333: "SCF.PHY"
  shm_level: 6      # Example: overlay shm_log_level for a tag
```

The following example shows results in the phy.log log:

```
05:22:56.350648 D [SCF.PHY] >>> SCF_FAPI_UCI_INDICATION: PUCCH interference Raw=1520
↳dbm 733 numMeasurements 1
05:22:56.350664 I [MAC.SCF] SFN 375.0 <<< SCF_FAPI_RX_PF_234_INTEFERNCE_INDICATION:
↳num=0 meas=733
```

For DTX detection for UCI on PUSCH, look for “detection status”. Sample below.

```
03:30:11.983670 D [SCF.PHY] >>> SCF_FAPI_UCI_INDICATION: HARQ detection status 4
03:30:11.983671 D [SCF.PHY] >>> SCF_FAPI_UCI_INDICATION:UCI on PUSCH HARQ bitlen 2
03:30:11.983671 D [SCF.PHY] >>> SCF_FAPI_UCI_INDICATION: PUCCH F234 CSI Part1
↳detection status 4 CSI P1 bit len 10
```

#### 1.5.4.10.1 Verification of PUSCH Measurement Reporting for BFP-9/14/16

Change the value of BFP in the matlab file 5GModel/nr\_matlab/config. Generate cuPHY and FAPI TV and run the test.

```
% BFP setting for cuPHY UL TV generation and UL performance simulation
SimCtrl.BFPforCuphy = 16; % 16, 14 or 9 for FP16, BFP14 or BFP9
```

Set log level to 6 and take h5dump of IND1 in FAPI TV.

```
h5dump -d IND1 TVnr_7427_gNB_FAPI_s0.h5_BFP9

HDF5 "TVnr_7427_gNB_FAPI_s0.h5_BFP9" {
DATASET "IND1" {
DATATYPE H5T_COMPOUND

{ H5T_STD_U32LE "idxPdu"; H5T_STD_U32LE "type"; H5T_STD_U32LE "TbCrcStatus"; H5T_STD_
↳U32LE "NumCb"; H5T_STD_U32LE "UL_CQI"; H5T_STD_U32LE "TimingAdvance"; H5T_STD_U32LE
↳"RSSI"; H5T_STD_U32LE "RSRP"; H5T_STD_I16LE "sinrdB"; H5T_STD_I16LE "postEqSinrdB";
↳H5T_STD_I16LE "noiseVardB"; H5T_STD_I16LE "postEqNoiseVardB"; }
Compare RSSI, RSRP, sinrdB and noiseVar values against the FAPI values in logs -

>>> SCF_FAPI_CRC_INDICATION 10.04 ul-sinr=20000 ta=63 ta-ns=16803 rssi=853 rsrp=912

[SCF.PHY] >>> RX_PE_NOISE_VARIANCE_INDICATION: PHY sfn=0x153 slot=0x0 num_meas=1
↳meas[0]=633
```

For comparing the raw values of UL measurements against the TV, take the h5dump of the following values from cuPHY TV. Then compare reference\_sinrdB, reference\_rssi, reference\_rsrpdB, and reference\_noiseVardB values against the raw values in logs.

```

23:18:48.950917 D [SCF.PHY] Raw RSSI=6.020887 db ul_configured_gain=48.680000
23:18:48.950919 D [SCF.PHY] Raw SINR=40.000000
23:18:48.950920 D [SCF.PHY] Raw RSRP =-0.045194 db ul_configured_gain =48.680000

23:18:48.950938 D [SCF.PHY] Raw PE Noise variance=-40.000000 ul_configured_gain=48.
↪680000

```

#### 1.5.4.10.2 Verification of PUCCH Measurement Reporting for BFP-9/14/16

Change the value of BFP in the 5GModel/nr\_matlab/config matlab file. Generate cuPHY and FAPI TV and run the test.

```

% BFP setting for cuPHY UL TV generation and UL performance simulation
SimCtrl.BFPforCuphy = 16; % 16, 14 or 9 for FP16, BFP14 or BFP9

```

Set log level to 6.

Match the value in logs against these fields in cuPHY TV.

Format 0 - F0UcisOutRef. Compare RSSI and RSRP values against corresponding value in logs.

```

SCF_FAPI_UCI_INDICATION 10.04: PUCCH : Raw SINR=0.000000 RSSI=16.824944 RSRP=10.804343

```

Format 1 - F1UcisOutRef. Compare “SinrDB”, “RSSI”, and “RSRP” values against the corresponding value in logs.

```

SCF_FAPI_UCI_INDICATION 10.04: PUCCH : Raw SINR=25.059706 RSSI=-3.927727 RSRP=-9.
↪948327

```

Format 2/3 - pucchF234\_refSnrBuffer, pucchF234\_refRsrpBuffer, pucchF234\_refRssiBuffer, and pucchF234\_refInterfBuffer. Compare them against relevant values in logs.

```

[SCF.PHY] Raw SINR=28.154160 RSRP=-0.132790 ul_configured_gain=48.680000

```

```

[SCF.PHY] Raw RSSI=5.887811 ul_configured_gain=48.680000

```

```

>>> SCF_FAPI_UCI_INDICATION: PUCCH interference Raw=-28.286949 dbm 750
↪numMeasurements 1

```

#### 1.5.4.10.3 Verification of PRACH Interference Level Report for BFP-9/14/16

Enable config in test\_mac\_config.yaml.

```

prach_interference: 1

```

Run the nrSim 5013 test.

```

nrSim 5013 --channels PRACH

```

Get “nOcc=x Raw PRACH interference” (x=0~3) from phy.log and get “PDUx\_noise” (x=1~4) from TV:

```

# phy.log
grep -o "PHY nOcc=[0-9] Raw PRACH interference=.*" phy.log
PHY nOcc=0 Raw PRACH interference=-16.046867 ul_configured_gain=48.680000 FAPI

```

(continues on next page)

(continued from previous page)

```

↪value=872
PHY nOcc=1 Raw PRACH interference=-16.921370 ul_configured_gain=48.680000 FAPI
↪value=863
PHY nOcc=2 Raw PRACH interference=-17.524746 ul_configured_gain=48.680000 FAPI
↪value=857
PHY nOcc=3 Raw PRACH interference=-18.472067 ul_configured_gain=48.680000 FAPI
↪value=848

# TV
h5ls -ld TVnr_5013_gNB_FAPI_s1.h5/PDU1_noise TVnr_5013_gNB_FAPI_s1.h5/PDU2_noise TVnr_
↪5013_gNB_FAPI_s1.h5/PDU3_noise TVnr_5013_gNB_FAPI_s1.h5/PDU4_noise
PDU1_noise          Dataset {1, 1}
  Data:
    (0,0) -16.0463
PDU2_noise          Dataset {1, 1}
  Data:
    (0,0) -16.0842
PDU3_noise          Dataset {1, 1}
  Data:
    (0,0) -16.0449
PDU4_noise          Dataset {1, 1}
  Data:
    (0,0) -16.294

```

Expected result:

```

abs(Raw PRACH interference - PDUx_noise) < 3
# Example, in above log the 4th occasion: abs(-18.472067 + 16.294) = 2.178067 < 3

```

#### 1.5.4.11 Cell Life-Cycle Test

##### To restart all cells while multiple cells are running

In the test\_mac\_config.yaml file, set the following:

```

# testMAC/test_mac_config.yaml

# Total slot number in test
test_slots: 8000 # When 1 slot = 0.5 ms, 8000 slots = 4 seconds.
# Restart interval after test_slots finished. Unit is second
restart_interval: 5

```

This instructs the testMAC to schedule 8000 slots then send cell stop request to all cells. After waiting 5 seconds, TestMAC sends a config request and cell start request to all cells.

Use the following commands to verify with the F08 4C pattern A case. The expected result is full throughput runs for approximately 4 seconds, test\_mac throughput stops, and ru-emulator throughput reduces to 0 for about 5 seconds, then the procedure repeats.

```

sudo ./cuPHY-CP/ru-emulator/ru_emulator/ru_emulator F08 4C 60
sudo -E ./cuPHY-CP/cuphycontroller/examples/cuphycontroller_scf F08
sudo ./cuPHY-CP/testMAC/testMAC/test_mac F08 4C 60

```

### 1.5.4.12 Terminate cuphycontroller Using a gRPC Message

Run the F08 E2E test case as usual:

```
sudo -E ${cuBB_SDK}/build/cuPHY-CP/cuphycontroller/examples/cuphycontroller_scf F08
sudo ${cuBB_SDK}/build/cuPHY-CP/ru-emulator/ru_emulator/ru_emulator F08 1C 60
sudo ${cuBB_SDK}/build/cuPHY-CP/testMAC/testMAC/test_mac F08 1C 60
```

Terminate cuphycontroller while the E2E test is running:

```
cd ${cuBB_SDK}/build/cuPHY-CP/cuphyoam
python3 ../../../../cuPHY-CP/cuphyoam/examples/aerial_terminate_cuphycontroller.py
```

Verify that the cuphycontroller stops running, and that `aerial_terminate_cuphycontroller.py` prints the following output:

```
12:23:32 Terminating cuphycontroller...
12:23:36 cuphycontroller terminated successfully!
```

### 1.5.4.13 Update M-plane Parameters Using gRPC Message

Dynamically changing M-plane parameters via gRPC message are often used with cell life during the initial cell setup with RUs, as well as to replace the RU while cells are running. See [List of parameters supported by dynamic OAM via gRPC and CONFIG.request \(M-plane\)](#).

The following sequence diagram shows an example of both scenarios:

- **Initial cell and M-plane setup:** After launching cuphycontroller, L1 is initialized and all cells are in idle state to be configured. The max number of cells is defined by the `cell_group_num` parameter in the cuphycontroller YAML config. In the example sequence diagram, the OAM sends a gRPC message to update M-plane parameters so that L1 gets the details to connect to the right RU for each cell. Then L2 sends `CONFIG.request` to configure the cell.

#### Note

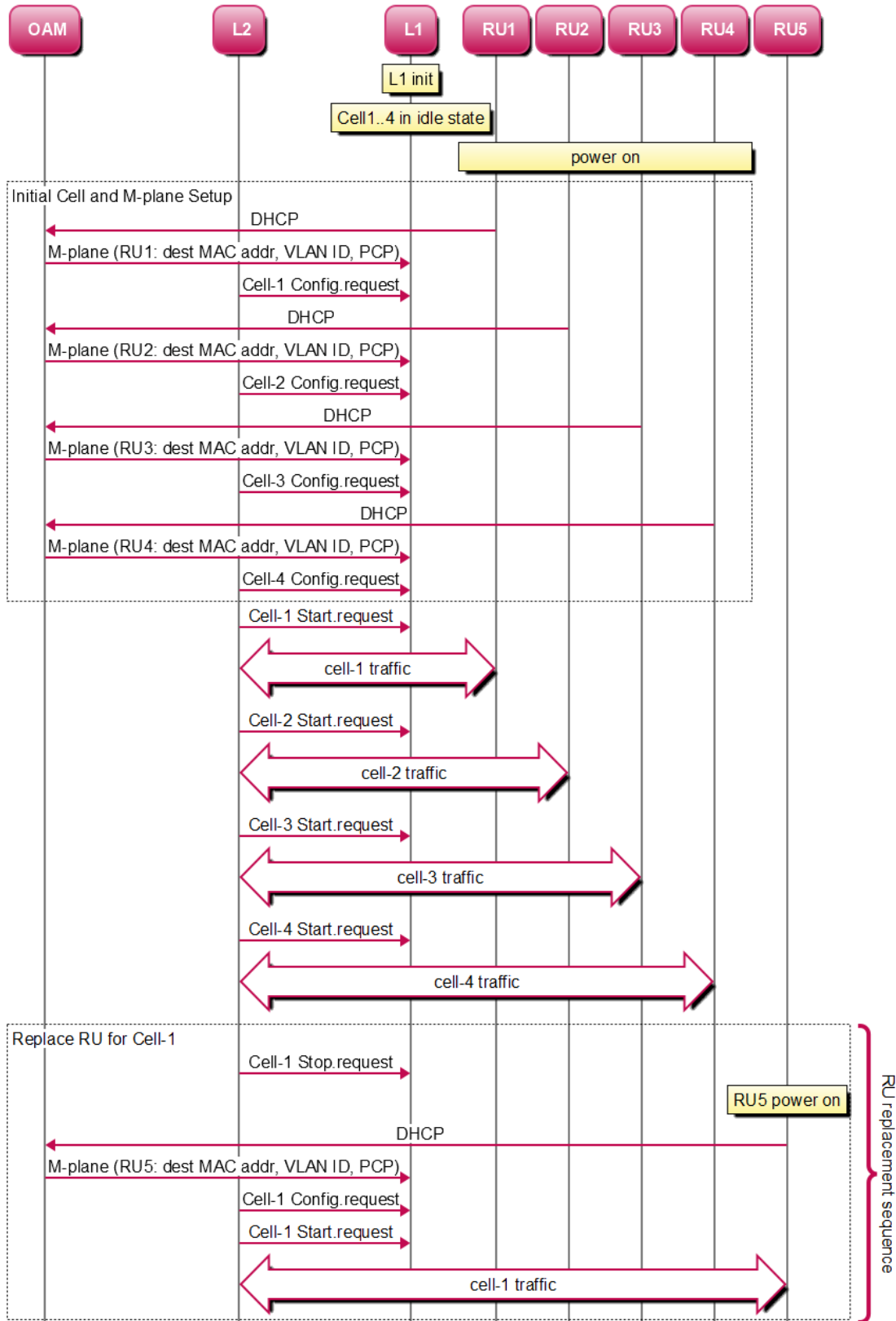
In the current implementation, all cells must be configured before any cell `Start.request`.

- **RU replacement while other cells are running:** The example sequence diagram shows the sequence to move the cell-1 traffics from RU1 to RU5. Firstly, the L2 must stop scheduling traffics on cell-1 and send `cell Stop.request` to cell-1. After that, OAM sends the new M-plane parameters via gRPC message for L1 to connect to RU5. Then L2 sends `Config.request` and `Start.request` to bring cell-1 to a running state again.

#### Note

In the current implementation, the cell `Config.request` after the first cell `Start.request` has no effect.

### Cell Start and RU replacement Sequence



### 1.5.4.13.1 X2 Launch Pattern Files Generation

In subsequent sections, X2 launch pattern files are needed for a related test. The `$cuBB_SDK/cuPHY-CP/cuphyoam/examples/launch_pattern_x2_update.py` script is used to generate them.

Here is the usage:

```
usage: launch_pattern_x2_update.py [-h] -f LAUNCH_PATTERN_FILE -o OUTPUT_DIR
launch_pattern_x2_update.py: error: the following arguments are required: -f/--launch_
↪pattern_file, -o/--output_dir
```

For example, to generate the X2 launch pattern file for TC “F08 2C 59”, run the following in container. This generates the corresponding ‘`$cuBB_SDK/testVectors/multi-cell/launch_pattern_F08_2C_59_X2.yaml`’ file.

```
python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/launch_pattern_x2_update.py -f $cuBB_SDK/
↪testVectors/multi-cell/launch_pattern_F08_2C_59.yaml -o $cuBB_SDK/testVectors/multi-
↪cell/'
```

### 1.5.4.13.2 Initial OAM Update

Here is an example of a 4 cell test. Run `cuphycontroller` with the wrong initial configurations, then use the gRPC message to update them to the right values.

#### 1.5.4.13.2.1 DST MAC Address OAM Initial Update Test - Single Cell

Update configs:

```
# Update 'cell_group_num' to 1 in cuphycontroller yaml config
sed -i "s/cell_group_num.*/cell_group_num: 1/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/
↪config/cuphycontroller_F08_*.yaml

# Use below settings for "Cell1" in ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml,
↪ note that only the eth mac address is changed cell_configs:
-
  name: "Cell1"
  eth: "20:04:9B:9E:27:B3"
  eAxC_UL: [8,0,1,2]
  eAxC_DL: [8,0,1,2]
  eAxC_prach_list: [15,7,0,1]
  dl_iq_data_fmt: {comp_meth: 1, bit_width: 14}
  ul_iq_data_fmt: {comp_meth: 1, bit_width: 14}
  peer: 0
  nic: 0
  vlan: 2
  pcp: 7
```

If you don’t perform the OAM update to change the cell 1 destination MAC address, the following test fails. The expected test result is no throughput on the ru-emulator side.

```
sudo -E ./cuphycontroller_scf F08_CG1
sudo ./ru_emulator F08 1C 59
sudo ./test_mac F08 1C 59
```



If you perform the OAM update to change the cell 1 destination MAC address, the following test passes. The expected test result is throughput on the ru-emulator side.

```
sudo -E ./cuphycontroller_scf F08_CG1
# Below OAM update command should be executed on the same server as cuphycontroller
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/
↪aerial_cell_param_net_update.py 1 20:04:9B:9E:27:B3 E002
sudo ./ru_emulator F08 1C 59
sudo ./test_mac F08 1C 59
```

#### 1.5.4.13.2.2 VLAN ID OAM Initial Update Test - Single Cell

Update configs:

```
# Update 'cell_group_num' to 1 in cuphycontroller yaml config
sed -i "s/cell_group_num.*/cell_group_num: 1/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/
↪config/cuphycontroller_F08_*.yaml
# Use below settings for "Cell1" in ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml,
↪ note that only the VLAN ID is changedcell_configs:
-
  name: "Cell1"
  eth: "20:04:9B:9E:27:A3"
  eAxC_UL: [8,0,1,2]
  eAxC_DL: [8,0,1,2]
  eAxC_prach_list: [15,7,0,1]
  dl_iq_data_fmt: {comp_meth: 1, bit_width: 14}
  ul_iq_data_fmt: {comp_meth: 1, bit_width: 14}
  peer: 0
  nic: 0
  vlan: 3
  pcp: 7
```

If you don't perform the OAM update to change the cell 1 VLAN ID, the following test fails. The expected test result is no throughput on the ru-emulator side.

```
sudo -E ./cuphycontroller_scf F08_CG1
sudo ./ru_emulator F08 1C 59
sudo ./test_mac F08 1C 59
```

If you perform the OAM update to change the cell 1 VLAN ID, the following test passes. The expected test result is throughput on the ru-emulator side.

```
sudo -E ./cuphycontroller_scf F08_CG1
# Below OAM update command should be executed on the same server as cuphycontroller
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/
↪aerial_cell_param_net_update.py 1 20:04:9B:9E:27:A3 E003
sudo ./ru_emulator F08 1C 59
sudo ./test_mac F08 1C 59
```

### 1.5.4.13.2.3 VLAN PCP OAM Initial Update Test - Single Cell

Update configs:

```
# Update 'cell_group_num' to 1 in cuphycontroller yaml config
sed -i "s/cell_group_num.*/cell_group_num: 1/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/
↪config/cuphycontroller_F08_*.yaml
# Use below settings for "Cell1" in ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml,
↪ note that only the PCP is changed cell_configs:
-
  name: "Cell1"
  eth: "20:04:9B:9E:27:A3"
  eAxC_UL: [8,0,1,2]
  eAxC_DL: [8,0,1,2]
  eAxC_prach_list: [15,7,0,1]
  dl_iq_data_fmt: {comp_meth: 1, bit_width: 14}
  ul_iq_data_fmt: {comp_meth: 1, bit_width: 14}
  peer: 0
  nic: 0
  vlan: 2
  pcp: 4
```

If you don't perform the OAM update to change the cell 1 PCP, the following test fails. The expected test result is no throughput on the ru-emulator side.

```
sudo -E ./cuphycontroller_scf F08_CG1
sudo ./ru_emulator F08 1C 59
sudo ./test_mac F08 1C 59
```

If you perform the OAM update to change the cell 1 PCP, the following test passes. The expected test result is throughput on the ru-emulator side.

```
sudo -E ./cuphycontroller_scf F08_CG1
# Below OAM update command should be executed on the same server as cuphycontroller
cd ${cuBB_SDK}/build/cuPHY-CP/cuphyoam && python3 ${cuBB_SDK}/cuPHY-CP/cuphyoam/examples/
↪aerial_cell_param_net_update.py 1 20:04:9B:9E:27:A3 8002
sudo ./ru_emulator F08 1C 59
sudo ./test_mac F08 1C 59
```

### 1.5.4.13.2.4 DST MAC + VLAN ID + PCP OAM Initial Update Test - Multi-Cells

```
# Update 'cell_group_num' to 4 in cuphycontroller yaml config
sed -i "s/cell_group_num.*/cell_group_num: 4/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/
↪config/cuphycontroller_F08_*.yaml

# Change eth, vlan, pcp of Cell 1~4 to any wrong values (here only show the values
↪which require change) in ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_
↪F08_*.yaml
-
  cell_id: 1
  dst_mac_addr: 20:20:20:20:20:A1
  vlan: 3
  pcp: 2
-
  cell_id: 2
```

(continues on next page)

(continued from previous page)

```

dst_mac_addr: 20:20:20:20:20:A2
vlan: 4
pcp: 3
-
cell_id: 3
dst_mac_addr: 20:20:20:20:20:A3
vlan: 5
pcp: 4
-
cell_id: 4
dst_mac_addr: 20:20:20:20:20:A4
vlan: 6
pcp: 5

```

If you don't perform the OAM update, the E2E test fails. The expected test result is no throughput on the ru-emulator side.

```

sudo -E ./cuphycontroller_scf F08_CG1
sudo ./ru_emulator F08 4C 59
sudo ./test_mac F08 4C 59

```

If you perform the OAM update for MAC + VLAN + PCP of the 4 cells to correct values, the E2E test passes. The expected test result is normal throughputs for all cells.

```

sudo -E ./cuphycontroller_scf F08_CG1
# OAM update MAC + VLAN + PCP of the 4 cells after cuphycontroller_scf started
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/
↪aerial_cell_param_net_update.py 1 20:04:9B:9E:27:A3 E002
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/
↪aerial_cell_param_net_update.py 2 26:04:9D:9E:29:B3 E002
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/
↪aerial_cell_param_net_update.py 3 20:34:9A:9E:29:B3 E002
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/
↪aerial_cell_param_net_update.py 4 22:34:9C:9E:29:A3 E002

sudo ./ru_emulator F08 4C 59
sudo ./test_mac F08 4C 59

```

### 1.5.4.13.3 Dynamic OAM Update

#### 1.5.4.13.3.1 DST MAC Address OAM On-the-Fly Update Test - Single Cell

Update the 'restart\_interval' and 'test\_cell\_update' sections with the following values in the testMac config file (\$cuBB\_SDK/cuPHY-CP/testMAC/testMAC/test\_mac\_config.yaml).

**testMAC configs:** Add cell\_id 0 to the "test\_cells" list to enable the test. Notice 'vlan' and 'pcp' is the same, but 'dst\_mac' is different here. Change 'test\_sequence' if required to test more cases.

```

restart_interval: 3

# For cell net parameters update test
# Configs of slot_point=0 only runs at init, other configs will run repeatably.
test_cell_update:
test_cells: [0]

```

(continues on next page)

(continued from previous page)

```
test_sequence:
- slot_point: 20000
  configs:
  - {cell_id: 0, dst_mac: 20:04:9B:9E:27:A3, vlan: 2, pcp: 7}
- slot_point: 40000
  configs:
  - {cell_id: 0, dst_mac: 26:04:9D:9E:29:B3, vlan: 2, pcp: 7}
```

testMAC automatically calls the following script to change the net parameters during the testMAC initialization and before cell restarting (Note: m-plane cell\_id = testMAC cell\_id + 1).

```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/
↪aerial_cell_param_net_update.py <m-plane cell_id> <dst_mac> <pcp_vlan>
```

In the ru-emulator config file (\$cuBB\_SDK/cuPHY-CP/ru-emulator/config/config.yaml), change “Cell2” parameters to be the same as “Cell1”, except for “eth” (the only difference is the eth MAC address).

```
-
  name: "Cell1"
  eth: "20:04:9B:9E:27:A3"
  eAxC_UL: [8,0,1,2]
  eAxC_DL: [8,0,1,2]
  eAxC_prach_list: [15,7,0,1]
  dl_iq_data_fmt: {comp_meth: 1, bit_width: 14}
  ul_iq_data_fmt: {comp_meth: 1, bit_width: 14}
  peer: 0
  nic: 0
  vlan: 2
  pcp: 7
-
  name: "Cell2"
  eth: "26:04:9D:9E:29:B3"
  eAxC_UL: [8,0,1,2]
  eAxC_DL: [8,0,1,2]
  eAxC_prach_list: [15,7,0,1]
  dl_iq_data_fmt: {comp_meth: 1, bit_width: 14}
  ul_iq_data_fmt: {comp_meth: 1, bit_width: 14}
  peer: 0
  nic: 0
  vlan: 2
  pcp: 7
```

Run normal F08 Pattern 0 1C E2E test commands, except change the ru-emulator parameter “1C” to “1C\_X2”. The following only shows the test case parameters; refer to the F08 cases for full instructions:

```
sudo ./ru_emulator F08 1C_59_X2
sudo -E ./cuphycontroller_scf F08_CG1
sudo ./test_mac F08 1C 59
```

Test result:

- ▶ ru-emulator throughput first starts on cell 1.
- ▶ ru-emulator throughput switches between cell 0 to cell 1, and repeats.
- ▶ The switching time points are decided by the above “slot\_point” in testMAC configurations. Currently 20000 slots = 10 seconds.

### 1.5.4.13.3.2 VLAN ID OAM On-the-Fly Update Test - Single Cell

Update 'restart\_interval' and 'test\_cell\_update' section with the following in the testMac config file \$cuBB\_SDK/cuPHY-CP/testMAC/testMAC/test\_mac\_config.yaml.

**testMAC configs:** Add cell\_id 0 to "test\_cells" list to enable the test. Notice 'dst\_mac' and 'pcp' are same, 'vlan' is different here. Change test\_sequence if required to test more cases.

```
restart_interval: 3

# For cell net parameters update test
test_cell_update:
test_cells: [0]
test_sequence:
- slot_point: 20000
  configs:
  - {cell_id: 0, dst_mac: 20:04:9B:9E:27:A3, vlan: 3, pcp: 7}
- slot_point: 40000
  configs:
  - {cell_id: 0, dst_mac: 20:04:9B:9E:27:A3, vlan: 2, pcp: 7}
```

testMAC automatically calls the following script to change the net parameters, and stop then restart the cell. (Note: m-plane cell\_id = testMAC cell\_id + 1)

```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/
↪aerial_cell_param_net_update.py <m-plane cell_id> <dst_mac> <pcp_vlan>
```

Update the 'cell\_configs' section with the following for "Cell1" and "Cell2" in the ru-emulator config file \$cuBB\_SDK/cuPHY-CP/ru-emulator/config/config.yaml. Note: The only difference is the vlan id.

```
-
  name: "Cell1"
  eth: "20:04:9B:9E:27:A3"
  eAxC_UL: [8,0,1,2]
  eAxC_DL: [8,0,1,2]
  eAxC_prach_list: [15,7,0,1]
  dl_iq_data_fmt: {comp_meth: 1, bit_width: 14}
  ul_iq_data_fmt: {comp_meth: 1, bit_width: 14}
  peer: 0
  nic: 0
  vlan: 2
  pcp: 7
-
  name: "Cell2"
  eth: "20:04:9B:9E:27:A3"
  eAxC_UL: [8,0,1,2]
  eAxC_DL: [8,0,1,2]
  eAxC_prach_list: [15,7,0,1]
  dl_iq_data_fmt: {comp_meth: 1, bit_width: 14}
  ul_iq_data_fmt: {comp_meth: 1, bit_width: 14}
  peer: 0
  nic: 0
  vlan: 3
  pcp: 7
```

Run normal F08 Pattern 0 1C E2E test commands except change ru-emulator parameter "1C" to "1C\_X2". This example only shows the test case parameters, refer to F08 cases for full instructions:

```
sudo ./ru_emulator F08 1C_59_X2
sudo -E ./cuphycontroller_scf F08_CG1
sudo ./test_mac F08 1C 59
```

Expected test result: ru-emulator to have throughput changed between cell 0 to cell 1, and repeat. The change time points are decided by the above “slot\_point” in testMAC configurations. Currently 20000 slots = 10 seconds.

### 1.5.4.13.3.3 VLAN PCP OAM On-the-Fly Update Test - Single Cell

Update ‘restart\_interval’ and ‘test\_cell\_update’ sections with the following in the testMac config file \$cuBB\_SDK/cuPHY-CP/testMAC/testMAC/test\_mac\_config.yaml.

**testMAC configs:** add cell\_id 0 to “test\_cells” list to enable the test. Notice ‘dst\_mac’ and ‘vlan’ are same, ‘pcp’ is different here. Change test\_sequence if requires to test more cases.

```
restart_interval: 3

# For cell net parameters update test
test_cell_update:
test_cells: [0]
test_sequence:
- slot_point: 20000
  configs:
  - {cell_id: 0, dst_mac: 20:04:9B:9E:27:A3, vlan: 2, pcp: 4}
- slot_point: 40000
  configs:
  - {cell_id: 0, dst_mac: 20:04:9B:9E:27:A3, vlan: 2, pcp: 7}
```

testMAC automatically calls the following script to change the net parameters, and stop then restart the cell. (Note: m-plane cell\_id = testMAC cell\_id + 1)

```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/
↪aerial_cell_param_net_update.py <m-plane cell_id> <dst_mac> <pcp_vlan>
```

Update the ‘cell\_configs’ section with the following for “Cell1” and “Cell2” in the ru-emulator config file \$cuBB\_SDK/cuPHY-CP/ru-emulator/config/config.yaml. Note: The only difference is the PCP value.

```
-
  name: "Cell1"
  eth: "20:04:9B:9E:27:A3"
  eAxC_UL: [8,0,1,2]
  eAxC_DL: [8,0,1,2]
  eAxC_prach_list: [15,7,0,1]
  dl_iq_data_fmt: {comp_meth: 1, bit_width: 14}
  ul_iq_data_fmt: {comp_meth: 1, bit_width: 14}
  peer: 0
  nic: 0
  vlan: 2
  pcp: 7
-
  name: "Cell2"
  eth: "20:04:9B:9E:27:A3"
  eAxC_UL: [8,0,1,2]
```

(continues on next page)

(continued from previous page)

```
eAxC_DL: [8,0,1,2]
eAxC_prach_list: [15,7,0,1]
dl_iq_data_fmt: {comp_meth: 1, bit_width: 14}
ul_iq_data_fmt: {comp_meth: 1, bit_width: 14}
peer: 0
nic: 0
vlan: 2
pcp: 4
```

Run normal F08 Pattern 0 1C E2E test commands except change ru-emulator parameter “1C” to “1C\_X2” The following example only shows the test case parameters, refer to F08 cases for full instructions:

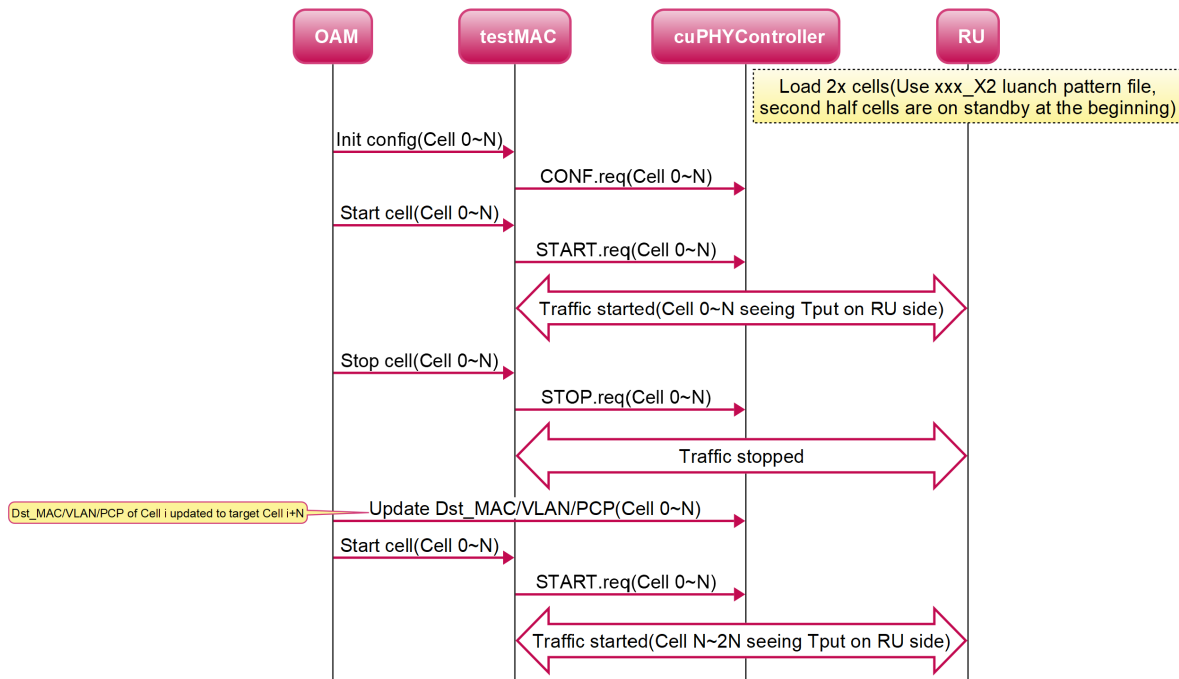
```
sudo ./ru_emulator F08 1C_59_X2
sudo -E ./cuphycontroller_scf F08_CG1
sudo ./test_mac F08 1C 59
```

Expected test result: ru-emulator has throughput changed between cell 0 to cell 1, and repeat. The change time points are decided by above “slot\_point” in testMAC configurations. Currently 20000 slots = 10 seconds.

### 1.5.4.13.3.4 DST MAC OAM On-the-Fly Update Test (with OAM Cell Ctrl Command) - Multi-Cells

The following sequence diagram shows the capability of updating Dst\_MAC/VLAN/PCP on the fly with multi-cell running.

#### Dynamic multi-cell Dst\_MAC/Vlan/PCP OAM update(with Cell ctrl cmd)



Configuration update:

```

# Save original configuration before the test
cp ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config.yaml ${cuBB_SDK}/cuPHY-CP/testMAC/
↪testMAC/test_mac_config.yaml.orig
cp ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml ${cuBB_SDK}/cuPHY-CP/ru-
↪emulator/config/config.yaml.orig
cp ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_CG1.yaml ${cuBB_SDK}
↪/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_CG1.yaml.orig

# Update config
sed -i "s/oam_cell_ctrl_cmd:./oam_cell_ctrl_cmd: 1/" ${cuBB_SDK}/cuPHY-CP/testMAC/
↪testMAC/test_mac_config.yaml

sed -i "s/eAxC_UL:./eAxC_UL: \\[0,1,2,3\\]/" ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/
↪config.yaml
sed -i "s/eAxC_DL:./eAxC_DL: \\[0,1,2,3\\]/" ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/
↪config.yaml
sed -i "s/eAxC_prach_list:./eAxC_prach_list: \\[5,6,7,10\\]/" ${cuBB_SDK}/cuPHY-CP/ru-
↪emulator/config/config.yaml

sed -i "s/cell_group_num:./cell_group_num: 4/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/
↪config/cuphycontroller_F08_CG1.yaml
sed -i "s/\\[.*/g" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_
↪CG1.yaml
sed -i "s/eAxC_id:./&\\[0, 1, 2, 3\\]/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/
↪cuphycontroller_F08_CG1.yaml
sed -i "s/eAxC_id_prach:./eAxC_id_prach: \\[5, 6, 7, 10\\]/" ${cuBB_SDK}/cuPHY-CP/
↪cuphycontroller/config/cuphycontroller_F08_CG1.yaml

# Restore the configuration after the test
cp ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config.yaml.orig ${cuBB_SDK}/cuPHY-CP/
↪testMAC/testMAC/test_mac_config.yaml
cp ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/config.yaml.orig ${cuBB_SDK}/cuPHY-CP/ru-
↪emulator/config/config.yaml
cp ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_CG1.yaml.orig $
↪${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_CG1.yaml

```

RU-Emulator will use launch pattern file “xC\_59\_X2” for test:

- ▶ launch\_pattern\_F08\_4C\_59\_X2.yaml
- ▶ launch\_pattern\_F08\_8C\_59\_X2.yaml

#### Note

There is a known issue with running launch\_pattern\_F08\_8C\_59\_X2.yaml.

Run normal F08 4C 59 E2E test commands except change ru-emulator parameter “4C” to “4C\_59\_X2”. The following example only shows the test case parameters, refer to F08 cases for full instructions:

```

sudo ./ru_emulator F08 4C_59_X2
sudo -E ./cuphycontroller_scf F08_CG1
sudo ./test_mac F08 4C 59

```

Init CONF.req is sent to all cells (executed on DU server):



```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam/
for i in {0..3}; do python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell_ctrl_cmd.
↪py --server_ip localhost --cell_id $i --cmd 3 && sleep 1; done;
```

START.req sent to all cells (executed on DU server):

```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam/
for i in {0..3}; do python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell_ctrl_cmd.
↪py --server_ip localhost --cell_id $i --cmd 1 && sleep 1; done;
```

At this point, validate that the RU emulator sees cell 0~3 have tput:

```
14:01:53.484104 Cell 0 DL 871.56 Mbps 1600 Slots | UL 111.55 Mbps 400 Slots | PBCH 300 | PDCCH_UL 1600 | PDCCH_DL 1600 | PRACH 100 Slots | PUCCH 400 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00%
|Seconds 1096
14:01:53.484139 Cell 1 DL 871.56 Mbps 1600 Slots | UL 111.55 Mbps 400 Slots | PBCH 300 | PDCCH_UL 1600 | PDCCH_DL 1600 | PRACH 100 Slots | PUCCH 400 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00%
|Seconds 1096
14:01:53.484161 Cell 2 DL 871.56 Mbps 1600 Slots | UL 111.55 Mbps 400 Slots | PBCH 300 | PDCCH_UL 1600 | PDCCH_DL 1600 | PRACH 100 Slots | PUCCH 400 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00%
|Seconds 1096
14:01:53.484184 Cell 3 DL 871.56 Mbps 1600 Slots | UL 111.55 Mbps 400 Slots | PBCH 300 | PDCCH_UL 1599 | PDCCH_DL 1600 | PRACH 100 Slots | PUCCH 400 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00%
|Seconds 1096
14:01:53.484205 Cell 4 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH 0 | PDCCH_UL 0 | PDCCH_DL 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00%
|Seconds 1096
14:01:53.484224 Cell 5 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH 0 | PDCCH_UL 0 | PDCCH_DL 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00%
|Seconds 1096
14:01:53.484243 Cell 6 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH 0 | PDCCH_UL 0 | PDCCH_DL 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00%
|Seconds 1096
14:01:53.484262 Cell 7 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH 0 | PDCCH_UL 0 | PDCCH_DL 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00%
|Seconds 1096
14:01:54.484105 Cell 0 DL 871.56 Mbps 1600 Slots | UL 111.55 Mbps 400 Slots | PBCH 300 | PDCCH_UL 1600 | PDCCH_DL 1600 | PRACH 100 Slots | PUCCH 400 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00%
|Seconds 1097
14:01:54.484138 Cell 1 DL 871.56 Mbps 1600 Slots | UL 111.55 Mbps 400 Slots | PBCH 300 | PDCCH_UL 1600 | PDCCH_DL 1600 | PRACH 100 Slots | PUCCH 400 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00%
|Seconds 1097
14:01:54.484163 Cell 2 DL 871.56 Mbps 1600 Slots | UL 111.55 Mbps 400 Slots | PBCH 300 | PDCCH_UL 1600 | PDCCH_DL 1600 | PRACH 100 Slots | PUCCH 400 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00%
|Seconds 1097
14:01:54.484186 Cell 3 DL 871.56 Mbps 1600 Slots | UL 111.55 Mbps 400 Slots | PBCH 300 | PDCCH_UL 1600 | PDCCH_DL 1599 | PRACH 100 Slots | PUCCH 400 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00%
|Seconds 1097
14:01:54.484208 Cell 4 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH 0 | PDCCH_UL 0 | PDCCH_DL 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00%
|Seconds 1097
14:01:54.484227 Cell 5 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH 0 | PDCCH_UL 0 | PDCCH_DL 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00%
|Seconds 1097
14:01:54.484246 Cell 6 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH 0 | PDCCH_UL 0 | PDCCH_DL 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00%
|Seconds 1097
14:01:54.484265 Cell 7 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH 0 | PDCCH_UL 0 | PDCCH_DL 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00%
|Seconds 1097
14:01:55.484111 Cell 0 DL 871.56 Mbps 1600 Slots | UL 111.55 Mbps 400 Slots | PBCH 300 | PDCCH_UL 1600 | PDCCH_DL 1600 | PRACH 100 Slots | PUCCH 400 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00%
|Seconds 1098
14:01:55.484152 Cell 1 DL 871.56 Mbps 1600 Slots | UL 111.55 Mbps 400 Slots | PBCH 300 | PDCCH_UL 1600 | PDCCH_DL 1600 | PRACH 100 Slots | PUCCH 400 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00%
|Seconds 1098
14:01:55.484177 Cell 2 DL 871.56 Mbps 1600 Slots | UL 111.55 Mbps 400 Slots | PBCH 300 | PDCCH_UL 1600 | PDCCH_DL 1600 | PRACH 100 Slots | PUCCH 400 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00%
|Seconds 1098
14:01:55.484201 Cell 3 DL 871.56 Mbps 1600 Slots | UL 111.55 Mbps 400 Slots | PBCH 300 | PDCCH_UL 1601 | PDCCH_DL 1601 | PRACH 100 Slots | PUCCH 400 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00%
|Seconds 1098
14:01:55.484225 Cell 4 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH 0 | PDCCH_UL 0 | PDCCH_DL 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00%
|Seconds 1098
14:01:55.484246 Cell 5 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH 0 | PDCCH_UL 0 | PDCCH_DL 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00%
|Seconds 1098
14:01:55.484267 Cell 6 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH 0 | PDCCH_UL 0 | PDCCH_DL 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00%
|Seconds 1098
14:01:55.484289 Cell 7 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH 0 | PDCCH_UL 0 | PDCCH_DL 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_C_ON 100.00% DL_U_ON 99.99% UL_C_ON 100.00%
|Seconds 1098
```

STOP.req sent to all cells (executed on DU server):

```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam/
for i in {0..3}; do python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell_ctrl_cmd.
↪py --server_ip localhost --cell_id $i --cmd 0 && sleep 1; done;
```

**OAM update** Cell i destination MAC updated to target cell i+4 on RU-Emulator side. That is: 004, 105, 206, 3->7) (executed on DU server):

```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam/
python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell_param_net_update.py 1 20:04:
↪9B:9E:27:05 E002 && sleep 1
python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell_param_net_update.py 2 20:04:
↪9B:9E:27:06 E002 && sleep 1
python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell_param_net_update.py 3 20:04:
↪9B:9E:27:07 E002 && sleep 1
python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell_param_net_update.py 4 20:04:
↪9B:9E:27:08 E002 && sleep 1
```

START.req sent to all cells (executed on DU server):

```
cd $cuBB_SDK/build/cuPHY-CP/cuphyoam/
for i in {0..3}; do python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell_ctrl_cmd.
↵py --server_ip localhost --cell_id $i --cmd 1 && sleep 1; done;
```

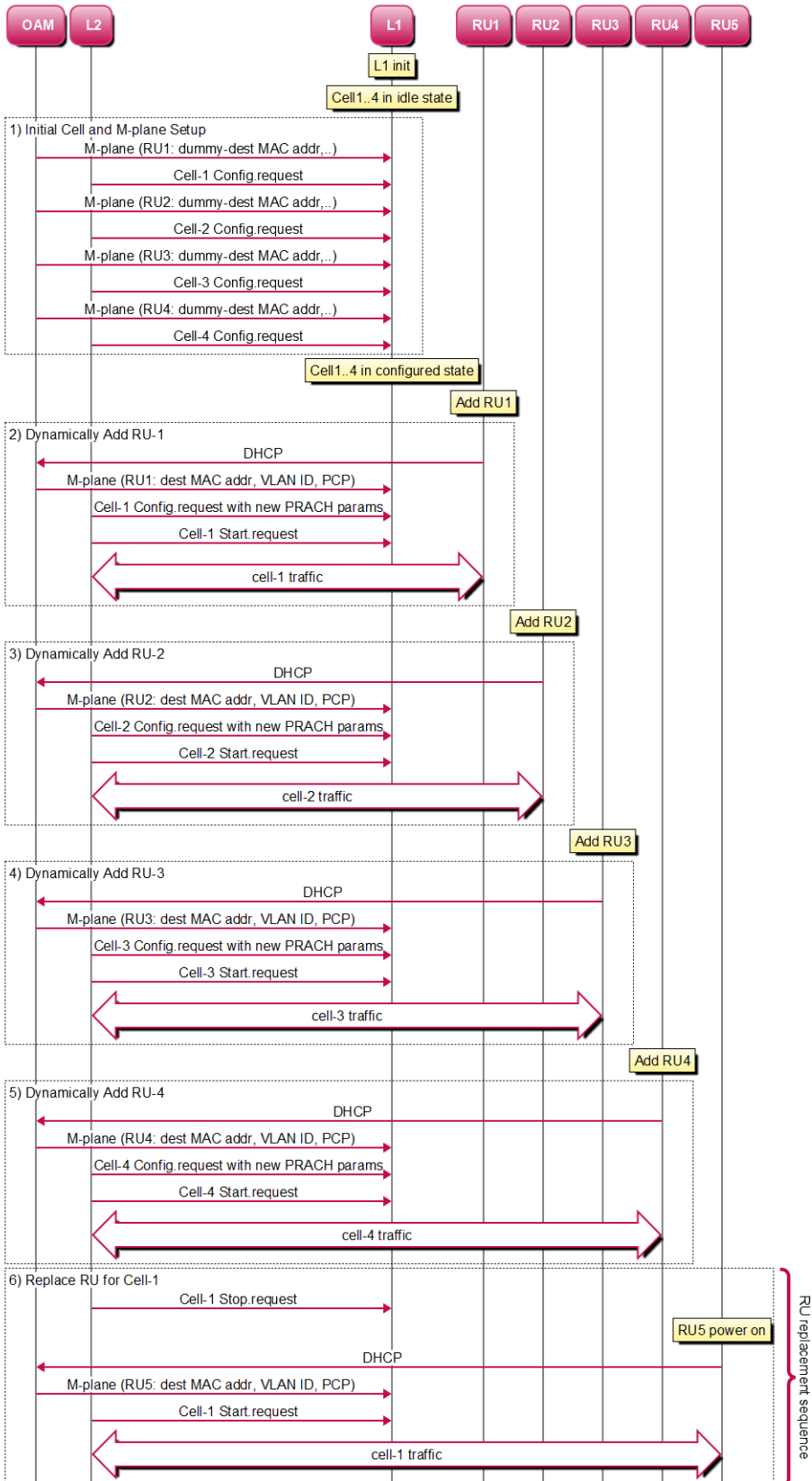
At this point, validate that the RU-Emulator sees cell 4~7 have tput:

13:58:37.484188 Cell 0 DL	0.00 Mbps	0 Slots	UL	0.00 Mbps	0 Slots	PBCH	0	PDCCH_UL	0	PDCCH_DL	0	PRACH	0 Slots	PUCCH	0 Slots	DL_C_ON 100.00%	DL_U_ON 99.99%	UL_C_ON 100.00%
13:58:37.484147 Cell 1 DL	0.00 Mbps	0 Slots	UL	0.00 Mbps	0 Slots	PBCH	0	PDCCH_UL	0	PDCCH_DL	0	PRACH	0 Slots	PUCCH	0 Slots	DL_C_ON 100.00%	DL_U_ON 99.99%	UL_C_ON 100.00%
13:58:37.484168 Cell 2 DL	0.00 Mbps	0 Slots	UL	0.00 Mbps	0 Slots	PBCH	0	PDCCH_UL	0	PDCCH_DL	0	PRACH	0 Slots	PUCCH	0 Slots	DL_C_ON 100.00%	DL_U_ON 99.99%	UL_C_ON 100.00%
13:58:37.484188 Cell 3 DL	0.00 Mbps	0 Slots	UL	0.00 Mbps	0 Slots	PBCH	0	PDCCH_UL	0	PDCCH_DL	0	PRACH	0 Slots	PUCCH	0 Slots	DL_C_ON 100.00%	DL_U_ON 99.99%	UL_C_ON 100.00%
13:58:37.484210 Cell 4 DL	871.56 Mbps	1600 Slots	UL	111.55 Mbps	400 Slots	PBCH	300	PDCCH_UL	1600	PDCCH_DL	1600	PRACH	100 Slots	PUCCH	400 Slots	DL_C_ON 100.00%	DL_U_ON 99.99%	UL_C_ON 100.00%
13:58:37.484213 Cell 5 DL	871.56 Mbps	1600 Slots	UL	111.55 Mbps	400 Slots	PBCH	300	PDCCH_UL	1600	PDCCH_DL	1600	PRACH	100 Slots	PUCCH	400 Slots	DL_C_ON 100.00%	DL_U_ON 99.99%	UL_C_ON 100.00%
13:58:37.484252 Cell 6 DL	871.56 Mbps	1600 Slots	UL	111.55 Mbps	400 Slots	PBCH	300	PDCCH_UL	1600	PDCCH_DL	1600	PRACH	100 Slots	PUCCH	400 Slots	DL_C_ON 100.00%	DL_U_ON 99.99%	UL_C_ON 100.00%
13:58:37.484272 Cell 7 DL	871.56 Mbps	1600 Slots	UL	111.55 Mbps	400 Slots	PBCH	300	PDCCH_UL	1600	PDCCH_DL	1600	PRACH	100 Slots	PUCCH	400 Slots	DL_C_ON 100.00%	DL_U_ON 99.99%	UL_C_ON 100.00%
3:58:38.484108 Cell 0 DL	0.00 Mbps	0 Slots	UL	0.00 Mbps	0 Slots	PBCH	0	PDCCH_UL	0	PDCCH_DL	0	PRACH	0 Slots	PUCCH	0 Slots	DL_C_ON 100.00%	DL_U_ON 99.99%	UL_C_ON 100.00%
3:58:38.484144 Cell 1 DL	0.00 Mbps	0 Slots	UL	0.00 Mbps	0 Slots	PBCH	0	PDCCH_UL	0	PDCCH_DL	0	PRACH	0 Slots	PUCCH	0 Slots	DL_C_ON 100.00%	DL_U_ON 99.99%	UL_C_ON 100.00%
3:58:38.484165 Cell 2 DL	0.00 Mbps	0 Slots	UL	0.00 Mbps	0 Slots	PBCH	0	PDCCH_UL	0	PDCCH_DL	0	PRACH	0 Slots	PUCCH	0 Slots	DL_C_ON 100.00%	DL_U_ON 99.99%	UL_C_ON 100.00%
3:58:38.484185 Cell 3 DL	0.00 Mbps	0 Slots	UL	0.00 Mbps	0 Slots	PBCH	0	PDCCH_UL	0	PDCCH_DL	0	PRACH	0 Slots	PUCCH	0 Slots	DL_C_ON 100.00%	DL_U_ON 99.99%	UL_C_ON 100.00%
3:58:38.484205 Cell 4 DL	871.56 Mbps	1600 Slots	UL	111.55 Mbps	400 Slots	PBCH	300	PDCCH_UL	1600	PDCCH_DL	1600	PRACH	100 Slots	PUCCH	400 Slots	DL_C_ON 100.00%	DL_U_ON 99.99%	UL_C_ON 100.00%
3:58:38.484228 Cell 5 DL	871.56 Mbps	1600 Slots	UL	111.55 Mbps	400 Slots	PBCH	300	PDCCH_UL	1600	PDCCH_DL	1600	PRACH	100 Slots	PUCCH	400 Slots	DL_C_ON 100.00%	DL_U_ON 99.99%	UL_C_ON 100.00%
3:58:38.484248 Cell 6 DL	871.56 Mbps	1600 Slots	UL	111.55 Mbps	400 Slots	PBCH	300	PDCCH_UL	1600	PDCCH_DL	1600	PRACH	100 Slots	PUCCH	400 Slots	DL_C_ON 100.00%	DL_U_ON 99.99%	UL_C_ON 100.00%
3:58:38.484271 Cell 7 DL	871.56 Mbps	1600 Slots	UL	111.55 Mbps	400 Slots	PBCH	300	PDCCH_UL	1600	PDCCH_DL	1600	PRACH	100 Slots	PUCCH	400 Slots	DL_C_ON 100.00%	DL_U_ON 99.99%	UL_C_ON 100.00%
13:58:39.484105 Cell 0 DL	0.00 Mbps	0 Slots	UL	0.00 Mbps	0 Slots	PBCH	0	PDCCH_UL	0	PDCCH_DL	0	PRACH	0 Slots	PUCCH	0 Slots	DL_C_ON 100.00%	DL_U_ON 99.99%	UL_C_ON 100.00%
13:58:39.484138 Cell 1 DL	0.00 Mbps	0 Slots	UL	0.00 Mbps	0 Slots	PBCH	0	PDCCH_UL	0	PDCCH_DL	0	PRACH	0 Slots	PUCCH	0 Slots	DL_C_ON 100.00%	DL_U_ON 99.99%	UL_C_ON 100.00%
13:58:39.484157 Cell 2 DL	0.00 Mbps	0 Slots	UL	0.00 Mbps	0 Slots	PBCH	0	PDCCH_UL	0	PDCCH_DL	0	PRACH	0 Slots	PUCCH	0 Slots	DL_C_ON 100.00%	DL_U_ON 99.99%	UL_C_ON 100.00%
13:58:39.484176 Cell 3 DL	0.00 Mbps	0 Slots	UL	0.00 Mbps	0 Slots	PBCH	0	PDCCH_UL	0	PDCCH_DL	0	PRACH	0 Slots	PUCCH	0 Slots	DL_C_ON 100.00%	DL_U_ON 99.99%	UL_C_ON 100.00%
13:58:39.484199 Cell 4 DL	871.56 Mbps	1600 Slots	UL	111.55 Mbps	400 Slots	PBCH	300	PDCCH_UL	1600	PDCCH_DL	1600	PRACH	100 Slots	PUCCH	400 Slots	DL_C_ON 100.00%	DL_U_ON 99.99%	UL_C_ON 100.00%
13:58:39.484220 Cell 5 DL	871.56 Mbps	1600 Slots	UL	111.55 Mbps	400 Slots	PBCH	300	PDCCH_UL	1600	PDCCH_DL	1600	PRACH	100 Slots	PUCCH	400 Slots	DL_C_ON 100.00%	DL_U_ON 99.99%	UL_C_ON 100.00%
13:58:39.484241 Cell 6 DL	871.56 Mbps	1600 Slots	UL	111.55 Mbps	400 Slots	PBCH	300	PDCCH_UL	1600	PDCCH_DL	1600	PRACH	100 Slots	PUCCH	400 Slots	DL_C_ON 100.00%	DL_U_ON 99.99%	UL_C_ON 100.00%
13:58:39.484260 Cell 7 DL	871.56 Mbps	1600 Slots	UL	111.55 Mbps	400 Slots	PBCH	300	PDCCH_UL	1600	PDCCH_DL	1600	PRACH	100 Slots	PUCCH	400 Slots	DL_C_ON 100.00%	DL_U_ON 99.99%	UL_C_ON 100.00%

### 1.5.4.14 Dynamic PRACH Configuration and Init Sequence Test

This sequence shows the changing of the PCI and 4 PRACH parameters after the initial config of a cell. There is also a possibility of changing the RU's VLAN ID and MAC address connected to the cell.

### L1 Init and Adding RU Dynamically



To support the sequence above, testMac has been enhanced to send CONFIG.req and START.req using OAM commands. Aerial has been enhanced to support dynamic PRACH parameter configuration and change of PCI in release 22-2.3. Changing the VLAN-id and DST MAC address was supported in previous releases and is used to support the Init sequence as shown above. The six PRACH parameters that can be changed are as follows:

- ▶ prachRootSequenceIndex
- ▶ restrictedSetConfig
- ▶ prachConfigIndex
- ▶ prachZeroCorrConf
- ▶ numPrachFdOccurrences
- ▶ K1
- ▶ prachConfigIndex
- ▶ restrictedSetConfig

To test this feature, testMac and ru-emulator are started with a higher number of cells from the cuphycontroller, and then OAM commands are used to change the configuration of a given cell.

Enable testMac to take OAM commands for CONFIG and START of a cell - change the test\_mac\_config.yaml file as follows:

```
# Send cell config/start/stop request via OAM command
oam_cell_ctrl_cmd: 1
```

To test the sequence with n cells, change cell\_group\_num to n in cuphycontroller\_F08\_\*.yaml and other corresponding files.

```
cell_group: 1
cell_group_num: n
fix_beta_dl: 0
```

For example, for 8C -

```
cell_group: 1
cell_group_num: 8
fix_beta_dl: 0
```

Update flow lists on both cuphycontroller and ru-emulator config:

```
sed -i "s/eAxC_UL:.*eAxC_UL: \\[0,1,2,3\\]/" ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/
↪config.yaml
sed -i "s/eAxC_DL:.*eAxC_DL: \\[0,1,2,3\\]/" ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/
↪config.yaml
sed -i "s/eAxC_prach_list:.*eAxC_prach_list: \\[5,6,7,10\\]/" ${cuBB_SDK}/cuPHY-CP/ru-
↪emulator/config/config.yaml

sed -i "s/\\[.*//g" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_
↪CG1.yaml
sed -i "s/eAxC_id.*&\\[0, 1, 2, 3\\]/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/
↪cuphycontroller_F08_CG1.yaml
sed -i "s/eAxC_id_prach.*eAxC_id_prach: \\[5, 6, 7, 10\\]/" ${cuBB_SDK}/cuPHY-CP/
↪cuphycontroller/config/cuphycontroller_F08_CG1.yaml
```

Run cuphycontroller, testMac for > nC and ru-emulator for > nC. For example, for 8C:

```
sudo ./cuPHY-CP/ru-emulator/ru_emulator/ru_emulator F08 9C 14
sudo -E ./cuPHY-CP/cuphycontroller/examples/cuphycontroller_scf F08
sudo ./cuPHY-CP/testMAC/testMAC/test_mac F08 9C 14
```

After testMac has created the gRPC Server and after you see the following logs on the testMac console, you can issue the OAM commands from the OAM window.

```
gRPC Server listening on 0.0.0.0:50052
20:33:56.124414 C [NVIPC:SHM] shm_ipc_open: forward_enable=0 fw_max_msg_buf_count=0
↳fw_max_data_buf_count=0
20:33:56.124434 C [MAC.PROC] set_launch_pattern_and_configs: fapi_type=1 tb_loc=1
20:33:56.124439 C [MAC.PROC] test_mac: create SCF FAPI interface
```

Execute the OAM commands for testMac from an OAM window:

- ▶ CONFIG.req command for all n cells. cmd=3 is for CONFIG.req
- ▶ Start cell-0 (cmd=1)

For example, for 8C:

```
export cuBB_SDK=$(pwd)
cd build/cuPHY-CP/cuphyoam/

for i in {0..7}; do python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell_ctrl_cmd.
↳py --server_ip localhost --cell_id $i --cmd 3 && sleep 1; done; //Send CONFIG.req
↳for cell 0~7
python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell_ctrl_cmd.py --server_ip
↳localhost --cell_id 0 --cmd 1 // Send START.req for cell-0
```

At this time you can see traffic running only for cell-0 on testMac, cuphycontroller and ru-emulator console:

```
# testMac console
20:34:22.124683 C [MAC.SCF] cell_init: cell_id=0 fapi_type=SCF global_tick=-1 first_
↳init=1
20:34:26.124793 C [MAC.SCF] cell_init: cell_id=1 fapi_type=SCF global_tick=-1 first_
↳init=1
20:34:28.124858 C [MAC.SCF] cell_start: cell_id=0 fapi_type=SCF global_tick=-1
04:55:13.040024 Cell 0 | DL 829.36 Mbps 1600 Slots | UL 122.92 Mbps 400 Slots |
↳Prmb 100 | HARQ 12000 | SR 0 | CSI1 2400 | CSI2 2400 | ERR 0 | INV 0
04:55:13.040037 Cell 1 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
↳Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
04:55:13.040045 Cell 2 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
↳Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
04:55:13.040051 Cell 3 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
↳Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
04:55:13.040058 Cell 4 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
↳Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
04:55:13.040065 Cell 5 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
↳Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
04:55:13.040069 Cell 6 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
↳Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
04:55:13.040074 Cell 7 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
↳Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
04:55:13.040081 Cell 8 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
↳Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
04:55:14.040025 Cell 0 | DL 829.36 Mbps 1600 Slots | UL 122.92 Mbps 400 Slots |
```

(continues on next page)

(continued from previous page)

```

↪ Prmb 100 | HARQ 12000 | SR 0 | CSI1 2400 | CSI2 2400 | ERR 0 | INV 0
04:55:14.040037 Cell 1 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
↪ Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
04:55:14.040045 Cell 2 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
↪ Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
04:55:14.040049 Cell 3 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
↪ Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
04:55:14.040054 Cell 4 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
↪ Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
04:55:14.040061 Cell 5 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
↪ Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
04:55:14.040067 Cell 6 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
↪ Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
04:55:14.040071 Cell 7 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
↪ Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
04:55:14.040077 Cell 8 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
↪ Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0

# cuphycontroller console
04:55:13.040004 C [SCF.PHY] Cell 0 | DL 829.36 Mbps 1600 Slots | UL 122.92 Mbps
↪ 400 Slots CRC 0 ( 0 ) | Tick 142000
04:55:13.040018 C [SCF.PHY] Cell 1 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps
↪ 0 Slots CRC 0 ( 0 ) | Tick 142000
04:55:13.040023 C [SCF.PHY] Cell 2 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps
↪ 0 Slots CRC 0 ( 0 ) | Tick 142000
04:55:13.040027 C [SCF.PHY] Cell 3 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps
↪ 0 Slots CRC 0 ( 0 ) | Tick 142000
04:55:13.040033 C [SCF.PHY] Cell 4 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps
↪ 0 Slots CRC 0 ( 0 ) | Tick 142000
04:55:13.040037 C [SCF.PHY] Cell 5 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps
↪ 0 Slots CRC 0 ( 0 ) | Tick 142000
04:55:13.040044 C [SCF.PHY] Cell 6 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps
↪ 0 Slots CRC 0 ( 0 ) | Tick 142000
04:55:13.040051 C [SCF.PHY] Cell 7 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps
↪ 0 Slots CRC 0 ( 0 ) | Tick 142000
04:55:14.040005 C [SCF.PHY] Cell 0 | DL 829.36 Mbps 1600 Slots | UL 122.92 Mbps
↪ 400 Slots CRC 0 ( 0 ) | Tick 144000
04:55:14.040019 C [SCF.PHY] Cell 1 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps
↪ 0 Slots CRC 0 ( 0 ) | Tick 144000
04:55:14.040023 C [SCF.PHY] Cell 2 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps
↪ 0 Slots CRC 0 ( 0 ) | Tick 144000
04:55:14.040028 C [SCF.PHY] Cell 3 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps
↪ 0 Slots CRC 0 ( 0 ) | Tick 144000
04:55:14.040033 C [SCF.PHY] Cell 4 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps
↪ 0 Slots CRC 0 ( 0 ) | Tick 144000
04:55:14.040040 C [SCF.PHY] Cell 5 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps
↪ 0 Slots CRC 0 ( 0 ) | Tick 144000
04:55:14.040046 C [SCF.PHY] Cell 6 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps
↪ 0 Slots CRC 0 ( 0 ) | Tick 144000
04:55:14.040050 C [SCF.PHY] Cell 7 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps
↪ 0 Slots CRC 0 ( 0 ) | Tick 144000

```

Now give OAM commands to switch the change PCI and PRACH parameters for cell-1 to cell 'n+1'.

For example, the following command triggers testMac to send another CONFIG.req for cell-1 with parameters for cell-9. The DST MAC address in the parameters for aerial\_cell\_param\_new\_update.py script must be the DST MAC address of n+1 cell in the cuphycontroller YAML file. For example, for 8C



testcase, the DST MAC address for cell-9 in the cuphycontroller YAML file is:

dst\_mac\_addr: 20:04:9B:9E:27:09

```
python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell_ctrl_cmd.py --server_ip
↳localhost --cell_id 1 --cmd 2 --target_cell_id 8 //Send CONFIG.req for cell-1 with
↳PRACH parameters read from TV for cell-8 and PCI of cell-8
python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell_param_net_update.py 2 xx:xx:
↳xx:xx:xx:xx E002 // Set VLAN-id and DST MAC address of cell-1 to point
↳to VLAN-id & DST MAC address of cell-9 in cuphycontroller yaml file
python3 $cuBB_SDK/cuPHY-CP/cuphyoam/examples/aerial_cell_ctrl_cmd.py --server_ip
↳localhost --cell_id 1 --cmd 1 // Send START.req for cell-1
```

Now testMAC and cuphycontroller see traffic for cell-0 and cell-1 while RU-Emulator sees traffic for cell-0 and cell-8.

```
# testMac console
20:35:00.125020 C [MAC.SCF] cell_start: cell_id=1 fapi_type=SCF global_tick=61130
20:35:00.560041 Cell 0 | DL 1731.61 Mbps 1600 Slots | UL 240.94 Mbps 400 Slots |
↳Prmb 100 | HARQ 12000 | SR 0 | CSI1 2400 | CSI2 2400 | ERR 0 | INV 0
20:35:00.560053 Cell 1 | DL 752.17 Mbps 695 Slots | UL 6.63 Mbps 174 Slots |
↳Prmb 43 | HARQ 5220 | SR 0 | CSI1 1044 | CSI2 1044 | ERR 0 | INV 174
20:35:00.560058 Cell 2 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
↳Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
20:35:00.560063 Cell 3 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
↳Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
20:35:01.560039 Cell 0 | DL 1731.61 Mbps 1600 Slots | UL 240.94 Mbps 400 Slots |
↳Prmb 100 | HARQ 12000 | SR 0 | CSI1 2400 | CSI2 2400 | ERR 0 | INV 0
20:35:01.560050 Cell 1 | DL 1731.61 Mbps 1600 Slots | UL 15.06 Mbps 400 Slots |
↳Prmb 100 | HARQ 12000 | SR 0 | CSI1 2400 | CSI2 2400 | ERR 0 | INV 400
20:35:01.560055 Cell 2 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
↳Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
20:35:01.560060 Cell 3 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
↳Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
20:35:02.560041 Cell 0 | DL 1731.61 Mbps 1600 Slots | UL 240.94 Mbps 400 Slots |
↳Prmb 100 | HARQ 12000 | SR 0 | CSI1 2400 | CSI2 2400 | ERR 0 | INV 0
20:35:02.560053 Cell 1 | DL 1731.61 Mbps 1600 Slots | UL 15.06 Mbps 400 Slots |
↳Prmb 100 | HARQ 12000 | SR 0 | CSI1 2400 | CSI2 2400 | ERR 0 | INV 400
20:35:02.560058 Cell 2 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
↳Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
20:35:02.560063 Cell 3 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
↳Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
20:35:03.560040 Cell 0 | DL 1731.61 Mbps 1600 Slots | UL 240.94 Mbps 400 Slots |
↳Prmb 100 | HARQ 12000 | SR 0 | CSI1 2400 | CSI2 2400 | ERR 0 | INV 0
20:35:03.560051 Cell 1 | DL 1731.61 Mbps 1600 Slots | UL 15.06 Mbps 400 Slots |
↳Prmb 100 | HARQ 12000 | SR 0 | CSI1 2400 | CSI2 2400 | ERR 0 | INV 400
20:35:03.560056 Cell 2 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
↳Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
20:35:03.560061 Cell 3 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
↳Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
20:35:04.560043 Cell 0 | DL 1731.61 Mbps 1600 Slots | UL 240.94 Mbps 400 Slots |
↳Prmb 100 | HARQ 12000 | SR 0 | CSI1 2400 | CSI2 2400 | ERR 0 | INV 0
20:35:04.560054 Cell 1 | DL 1731.61 Mbps 1600 Slots | UL 15.06 Mbps 400 Slots |
↳Prmb 100 | HARQ 12000 | SR 0 | CSI1 2400 | CSI2 2400 | ERR 0 | INV 400
20:35:04.560059 Cell 2 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
↳Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
20:35:04.560064 Cell 3 | DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots |
↳Prmb 0 | HARQ 0 | SR 0 | CSI1 0 | CSI2 0 | ERR 0 | INV 0
```

(continues on next page)

(continued from previous page)

```

# cuPhyController console
20:35:00.560005 C [SCF.PHY] Cell 0 | DL 1731.61 Mbps 1600 Slots | UL 240.94 Mbps
↪400 Slots CRC 0 ( 0) | Tick 62000
20:35:00.560014 C [SCF.PHY] Cell 1 | DL 752.17 Mbps 695 Slots | UL 104.81 Mbps
↪174 Slots CRC 0 ( 0) | Tick 62000
20:35:01.560004 C [SCF.PHY] Cell 0 | DL 1731.61 Mbps 1600 Slots | UL 240.94 Mbps
↪400 Slots CRC 0 ( 0) | Tick 64000
20:35:01.560012 C [SCF.PHY] Cell 1 | DL 1731.61 Mbps 1600 Slots | UL 240.94 Mbps
↪400 Slots CRC 0 ( 0) | Tick 64000
20:35:02.560005 C [SCF.PHY] Cell 0 | DL 1731.61 Mbps 1600 Slots | UL 240.94 Mbps
↪400 Slots CRC 0 ( 0) | Tick 66000
20:35:02.560013 C [SCF.PHY] Cell 1 | DL 1731.61 Mbps 1600 Slots | UL 240.94 Mbps
↪400 Slots CRC 0 ( 0) | Tick 66000
20:35:03.560005 C [SCF.PHY] Cell 0 | DL 1731.61 Mbps 1600 Slots | UL 240.94 Mbps
↪400 Slots CRC 0 ( 0) | Tick 68000
20:35:03.560012 C [SCF.PHY] Cell 1 | DL 1731.61 Mbps 1600 Slots | UL 240.94 Mbps
↪400 Slots CRC 0 ( 0) | Tick 68000
20:35:04.560006 C [SCF.PHY] Cell 0 | DL 1731.61 Mbps 1600 Slots | UL 240.94 Mbps
↪400 Slots CRC 0 ( 0) | Tick 70000
20:35:04.560013 C [SCF.PHY] Cell 1 | DL 1731.61 Mbps 1600 Slots | UL 240.94 Mbps
↪400 Slots CRC 0 ( 0) | Tick 70000
20:35:05.457529 C [SCF.PHY] Cell 0 | DL 1553.04 Mbps 1435 Slots | UL 215.64 Mbps
↪358 Slots CRC 0 ( 0)
20:35:05.457541 C [SCF.PHY] Cell 1 | DL 1553.04 Mbps 1435 Slots | UL 215.64 Mbps
↪358 Slots CRC 0 ( 0)
20:35:05.457676 C [SCF.PHY] Cell 0 | DL 1553.04 Mbps 1435 Slots | UL 215.64 Mbps
↪358 Slots CRC 0 ( 0)
20:35:05.457681 C [SCF.PHY] Cell 1 | DL 1553.04 Mbps 1435 Slots | UL 215.64 Mbps
↪358 Slots CRC 0 ( 0)

```

```

# ru-emulator console
12:15:45.760099 Cell 8 DL 829.36 Mbps 1600 Slots | UL 122.92 Mbps 400 Slots | PBCH
↪ 100 | PDCCH_DL 1600 | CSI_RS 1600 | PRACH 100 Slots | PUCCH 400 Slots | DL_C_
↪ON 100.00% DL_U_ON 100.00% UL_C_ON 100.00% |Seconds 513
12:15:46.760025 Cell 0 DL 829.36 Mbps 1600 Slots | UL 122.92 Mbps 400 Slots | PBCH
↪ 100 | PDCCH_DL 1600 | CSI_RS 1600 | PRACH 100 Slots | PUCCH 400 Slots | DL_C_
↪ON 100.00% DL_U_ON 100.00% UL_C_ON 100.00% |Seconds 514
12:15:46.760041 Cell 1 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH
↪ 0 | PDCCH_DL 0 | CSI_RS 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_C_
↪ON 0.00% DL_U_ON 0.00% UL_C_ON 0.00% |Seconds 514
12:15:46.760049 Cell 2 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH
↪ 0 | PDCCH_DL 0 | CSI_RS 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_C_
↪ON 0.00% DL_U_ON 0.00% UL_C_ON 0.00% |Seconds 514
12:15:46.760054 Cell 3 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH
↪ 0 | PDCCH_DL 0 | CSI_RS 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_C_
↪ON 0.00% DL_U_ON 0.00% UL_C_ON 0.00% |Seconds 514
12:15:46.760060 Cell 4 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH
↪ 0 | PDCCH_DL 0 | CSI_RS 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_C_
↪ON 0.00% DL_U_ON 0.00% UL_C_ON 0.00% |Seconds 514
12:15:46.760073 Cell 5 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH
↪ 0 | PDCCH_DL 0 | CSI_RS 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_C_
↪ON 0.00% DL_U_ON 0.00% UL_C_ON 0.00% |Seconds 514
12:15:46.760078 Cell 6 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH
↪ 0 | PDCCH_DL 0 | CSI_RS 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_C_
↪ON 0.00% DL_U_ON 0.00% UL_C_ON 0.00% |Seconds 514

```

(continues on next page)



(continued from previous page)

```

12:15:46.760083 Cell 7 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH
↪ 0 | PDCCH_DL 0 | CSI_RS 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_C_
↪ON 0.00% DL_U_ON 0.00% UL_C_ON 0.00% |Seconds 514
12:15:46.760090 Cell 8 DL 829.36 Mbps 1600 Slots | UL 122.92 Mbps 400 Slots | PBCH
↪ 100 | PDCCH_DL 1600 | CSI_RS 1600 | PRACH 100 Slots | PUCCH 400 Slots | DL_C_
↪ON 100.00% DL_U_ON 100.00% UL_C_ON 100.00% |Seconds 514
12:15:47.760024 Cell 0 DL 829.36 Mbps 1600 Slots | UL 122.92 Mbps 400 Slots | PBCH
↪ 100 | PDCCH_DL 1600 | CSI_RS 1600 | PRACH 100 Slots | PUCCH 400 Slots | DL_C_
↪ON 100.00% DL_U_ON 100.00% UL_C_ON 100.00% |Seconds 515
12:15:47.760041 Cell 1 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH
↪ 0 | PDCCH_DL 0 | CSI_RS 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_C_
↪ON 0.00% DL_U_ON 0.00% UL_C_ON 0.00% |Seconds 515
12:15:47.760047 Cell 2 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH
↪ 0 | PDCCH_DL 0 | CSI_RS 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_C_
↪ON 0.00% DL_U_ON 0.00% UL_C_ON 0.00% |Seconds 515
12:15:47.760053 Cell 3 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH
↪ 0 | PDCCH_DL 0 | CSI_RS 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_C_
↪ON 0.00% DL_U_ON 0.00% UL_C_ON 0.00% |Seconds 515
12:15:47.760060 Cell 4 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH
↪ 0 | PDCCH_DL 0 | CSI_RS 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_C_
↪ON 0.00% DL_U_ON 0.00% UL_C_ON 0.00% |Seconds 515
12:15:47.760066 Cell 5 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH
↪ 0 | PDCCH_DL 0 | CSI_RS 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_C_
↪ON 0.00% DL_U_ON 0.00% UL_C_ON 0.00% |Seconds 515
12:15:47.760076 Cell 6 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH
↪ 0 | PDCCH_DL 0 | CSI_RS 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_C_
↪ON 0.00% DL_U_ON 0.00% UL_C_ON 0.00% |Seconds 515
12:15:47.760082 Cell 7 DL 0.00 Mbps 0 Slots | UL 0.00 Mbps 0 Slots | PBCH
↪ 0 | PDCCH_DL 0 | CSI_RS 0 | PRACH 0 Slots | PUCCH 0 Slots | DL_C_
↪ON 0.00% DL_U_ON 0.00% UL_C_ON 0.00% |Seconds 515
12:15:47.760089 Cell 8 DL 829.36 Mbps 1600 Slots | UL 122.92 Mbps 400 Slots | PBCH
↪ 100 | PDCCH_DL 1600 | CSI_RS 1600 | PRACH 100 Slots | PUCCH 400 Slots | DL_C_
↪ON 100.00% DL_U_ON 100.00% UL_C_ON 100.00% |Seconds 515
12:15:48.760023 Cell 0 DL 829.36 Mbps 1600 Slots | UL 122.92 Mbps 400 Slots | PBCH
↪ 100 | PDCCH_DL 1600 | CSI_RS 1600 | PRACH 100 Slots | PUCCH 400 Slots | DL_C_
↪ON 100.00% DL_U_ON 100.00% UL_C_ON 100.00% |Seconds 516

```

#### 1.5.4.15 Duplicate Configuration and Init Sequence Test

Duplicate Cell Config.request is a feature that enables dynamically configuring and starts a cell on an individual basis. The Config.request for all the cells need not be sent before a Start.Req is issued. To enable this feature, the following configuration in L2Adapter and testMac must be provisioned.

```

sed -i "s/duplicate_config_all_cells.*/duplicate_config_all_cells: 1/" ${cuBB_SDK}/
↪cuPHY-CP/cuphycontroller/config/l2_adapter_config_F08_CG1.yaml

sed -i "s/cell_config_wait.*/cell_config_wait: 1000/" ${cuBB_SDK}/cuPHY-CP/testMAC/
↪testMAC/test_mac_config.yaml

sed -i "s/oam_cell_ctrl_cmd.*/oam_cell_ctrl_cmd: 1/" ${cuBB_SDK}/cuPHY-CP/testMAC/
↪testMAC/test_mac_config.yaml

```

To test the sequence with n cells, change cell\_group\_num to n in cuphycontroller\_F08\_\*.yaml and other corresponding files.

```
cell_group: 1
cell_group_num: n
fix_beta_dl: 0
```

For example, for 8C:

```
cell_group: 1
cell_group_num: 8
fix_beta_dl: 0
```

Update flow lists on both cuphycontroller and ru-emulator config:

```
sed -i "s/eAxC_UL:.*eAxC_UL: \\[0,1,2,3\\]/" ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/
↪config.yaml
sed -i "s/eAxC_DL:.*eAxC_DL: \\[0,1,2,3\\]/" ${cuBB_SDK}/cuPHY-CP/ru-emulator/config/
↪config.yaml
sed -i "s/eAxC_prach_list:.*eAxC_prach_list: \\[5,6,7,10\\]/" ${cuBB_SDK}/cuPHY-CP/ru-
↪emulator/config/config.yaml

sed -i "s/\\[.*//g" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_F08_
↪CG1.yaml
sed -i "s/eAxC_id.*&\\[0, 1, 2, 3\\]/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/
↪cuphycontroller_F08_CG1.yaml
sed -i "s/eAxC_id_prach.*eAxC_id_prach: \\[5, 6, 7, 10\\]/" ${cuBB_SDK}/cuPHY-CP/
↪cuphycontroller/config/cuphycontroller_F08_CG1.yaml
```

Run cuphycontroller, testMac for > nC and ru-emulator for > nC. For example, for 8C:

```
sudo ./cuPHY-CP/ru-emulator/ru_emulator/ru_emulator F08 8C 14
sudo -E ./cuPHY-CP/cuphycontroller/examples/cuphycontroller_scf F08_CG1
sudo ./cuPHY-CP/testMAC/testMAC/test_mac F08 8C 14
```

After testMac has created the gRPC Server and after you see the following logs on the testMac console, you can issue the OAM commands from the OAM window:

```
gRPC Server listening on 0.0.0.0:50052
20:33:56.124414 C [NVIPC:SHM] shm_ipc_open: forward_enable=0 fw_max_msg_buf_count=0
↪fw_max_data_buf_count=0
20:33:56.124434 C [MAC.PROC] set_launch_pattern_and_configs: fapi_type=1 tb_loc=1
20:33:56.124439 C [MAC.PROC] test_mac: create SCF FAPI interface
```

Execute the OAM commands for testMac from a OAM window:

- ▶ CONFIG.req command for 1 cell at a time. cmd=3 is for CONFIG.req
- ▶ Start cell-0 (cmd=1)
- ▶ Repeat the above sequence for all cells

```
export cuBB_SDK=$(pwd)
cd build/cuPHY-CP/cuphyoam/
#Note that the config&start of cells can be in any order
for i in {0..7}; do python3 ${cuBB_SDK}/cuPHY-CP/cuphyoam/examples/aerial_cell_ctrl_cmd.
↪py --server_ip localhost --cell_id $i --cmd 3 && sleep 3 && python3 ${cuBB_SDK}/cuPHY-
↪CP/cuphyoam/examples/aerial_cell_ctrl_cmd.py --server_ip localhost --cell_id $i --
↪cmd 1; done; //Send CONFIG.req and Start.req for cell 0~7
```

```

# testMac console
20:34:22.124683 C [MAC.SCF] cell_init: cell_id=0 fapi_type=SCF global_tick=-1
↪first_init=1
20:34:26.124793 C [MAC.SCF] cell_init: cell_id=1 fapi_type=SCF global_tick=-1
↪first_init=1
20:34:28.124858 C [MAC.SCF] cell_start: cell_id=0 fapi_type=SCF global_tick=-1

04:55:13.040024 Cell 0 | DL      829.36 Mbps 1600 Slots | UL  122.92 Mbps  400 Slots |
↪Prmb  100 | HARQ 12000 | SR    0 | CSI1 2400 | CSI2 2400 | ERR  0 | INV 0
04:55:13.040037 Cell 1 | DL      829.36 Mbps 1600 Slots | UL  122.92 Mbps  400 Slots |
↪Prmb   0 | HARQ   0 | SR    0 | CSI1   0 | CSI2   0 | ERR  0 | INV 0
04:55:13.040045 Cell 2 | DL      829.36 Mbps 1600 Slots | UL  122.92 Mbps  400 Slots |
↪Prmb   0 | HARQ   0 | SR    0 | CSI1   0 | CSI2   0 | ERR  0 | INV 0
04:55:13.040051 Cell 3 | DL      829.36 Mbps 1600 Slots | UL  122.92 Mbps  400 Slots |
↪Prmb   0 | HARQ   0 | SR    0 | CSI1   0 | CSI2   0 | ERR  0 | INV 0
04:55:13.040058 Cell 4 | DL      829.36 Mbps 1600 Slots | UL  122.92 Mbps  400 Slots |
↪Prmb   0 | HARQ   0 | SR    0 | CSI1   0 | CSI2   0 | ERR  0 | INV 0
04:55:13.040065 Cell 5 | DL      829.36 Mbps 1600 Slots | UL  122.92 Mbps  400 Slots |
↪Prmb   0 | HARQ   0 | SR    0 | CSI1   0 | CSI2   0 | ERR  0 | INV 0
04:55:13.040069 Cell 6 | DL      829.36 Mbps 1600 Slots | UL  122.92 Mbps  400 Slots |
↪Prmb   0 | HARQ   0 | SR    0 | CSI1   0 | CSI2   0 | ERR  0 | INV 0
04:55:13.040074 Cell 7 | DL      829.36 Mbps 1600 Slots | UL  122.92 Mbps  400 Slots |
↪Prmb   0 | HARQ   0 | SR    0 | CSI1   0 | CSI2   0 | ERR  0 | INV 0

# cuphycontroller console
04:55:13.040004 C [SCF.PHY] Cell 0 | DL      829.36 Mbps 1600 Slots | UL  122.92 Mbps
↪ 400 Slots CRC  0 (  0) | Tick 142000
04:55:13.040018 C [SCF.PHY] Cell 1 | DL      829.36 Mbps 1600 Slots | UL  122.92 Mbps
↪ 400 Slots CRC  0 (  0) | Tick 142000
04:55:13.040023 C [SCF.PHY] Cell 2 | DL      829.36 Mbps 1600 Slots | UL  122.92 Mbps
↪ 400 Slots CRC  0 (  0) | Tick 142000
04:55:13.040027 C [SCF.PHY] Cell 3 | DL      829.36 Mbps 1600 Slots | UL  122.92 Mbps
↪ 400 Slots CRC  0 (  0) | Tick 142000
04:55:13.040033 C [SCF.PHY] Cell 4 | DL      829.36 Mbps 1600 Slots | UL  122.92 Mbps
↪ 400 Slots CRC  0 (  0) | Tick 142000
04:55:13.040037 C [SCF.PHY] Cell 5 | DL      829.36 Mbps 1600 Slots | UL  122.92 Mbps
↪ 400 Slots CRC  0 (  0) | Tick 142000
04:55:13.040044 C [SCF.PHY] Cell 6 | DL      829.36 Mbps 1600 Slots | UL  122.92 Mbps
↪ 400 Slots CRC  0 (  0) | Tick 142000
04:55:13.040051 C [SCF.PHY] Cell 7 | DL      829.36 Mbps 1600 Slots | UL  122.92 Mbps
↪ 400 Slots CRC  0 (  0) | Tick 142000

```

### 1.5.4.16 How to Get Aerial Metrics

Run the following on gNB Server#1. Make sure `-DAERIAL_METRICS=1` added in cmake config:

```
curl localhost:8081/metrics
```

Set the Prometheus thread to a proper CPU core number. For testing on GH setup, change the F08 config file so that DPDK-related metrics are updated, then launch cuphycontroller:

```
sed -i "s/prometheus_thread.*/prometheus_thread: 23/" ${cuBB_SDK}/cuPHY-CP/
↪cuphycontroller/config/cuphycontroller_F08_CG1.yaml
sudo -E ${cuBB_SDK}/build/cuPHY-CP/cuphycontroller/examples/cuphycontroller_scf F08_CG1
```

Do NOT start test\_mac yet. Query the metrics. All metrics should be 0 except for:

- ▶ aerial\_cuphytcp\_net\_tx\_accu\_sched\_clock\_queue\_jitter\_ns
- ▶ aerial\_cuphytcp\_net\_tx\_accu\_sched\_clock\_queue\_wander\_ns

Launch RU emulator:

```
sudo ${cuBB_SDK}/build/cuPHY-CP/ru-emulator/ru_emulator/ru_emulator F08 8C_59
```

Run testMAC with 20000 slots:

```
sed -i "s/test_slots.*/test_slots: 20000/" ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_
↪mac_config.yaml
sudo -E ${cuBB_SDK}/build/cuPHY-CP/testMAC/testMAC/test_mac F08 8C_59
```

Let the test finish. Wait until you see that the test\_mac output shows 20000 slots finished:

```
13:16:37.244835 C [MAC.FAPI] Finished running 20000 slots test
```

Don't kill the cuphycontroller yet. Query the metrics again, see the example log as follows:

```
...
# HELP aerial_cuphytcp_slots_total Aerial cuPHY-CP total number of processed Downlink
↪slots
# TYPE aerial_cuphytcp_slots_total counter
aerial_cuphytcp_slots_total{cell="8",type="UL"} 4000
aerial_cuphytcp_slots_total{cell="8",type="DL"} 16000
aerial_cuphytcp_slots_total{cell="7",type="UL"} 4000
aerial_cuphytcp_slots_total{cell="6",type="UL"} 4000
aerial_cuphytcp_slots_total{cell="3",type="DL"} 16000
aerial_cuphytcp_slots_total{cell="7",type="DL"} 16000
aerial_cuphytcp_slots_total{cell="2",type="UL"} 4000
aerial_cuphytcp_slots_total{cell="3",type="UL"} 4000
aerial_cuphytcp_slots_total{cell="1",type="DL"} 16000
aerial_cuphytcp_slots_total{cell="1",type="UL"} 4000
aerial_cuphytcp_slots_total{cell="2",type="DL"} 16000
aerial_cuphytcp_slots_total{cell="6",type="DL"} 16000
aerial_cuphytcp_slots_total{cell="4",type="UL"} 4000
aerial_cuphytcp_slots_total{cell="5",type="DL"} 16000
aerial_cuphytcp_slots_total{cell="4",type="DL"} 16000
aerial_cuphytcp_slots_total{cell="5",type="UL"} 4000
...
# HELP aerial_cuphytcp_on_time_uplane_rx_packets_total Aerial cuPHY-CP Uplink U-plane
↪packets which arrived within their receive windows
# TYPE aerial_cuphytcp_on_time_uplane_rx_packets_total counter
aerial_cuphytcp_on_time_uplane_rx_packets_total{cell="7"} 1680000
aerial_cuphytcp_on_time_uplane_rx_packets_total{cell="6"} 1680000
aerial_cuphytcp_on_time_uplane_rx_packets_total{cell="2"} 1680000
aerial_cuphytcp_on_time_uplane_rx_packets_total{cell="1"} 1680000
aerial_cuphytcp_on_time_uplane_rx_packets_total{cell="8"} 1680000
aerial_cuphytcp_on_time_uplane_rx_packets_total{cell="3"} 1680000
aerial_cuphytcp_on_time_uplane_rx_packets_total{cell="4"} 1680000
aerial_cuphytcp_on_time_uplane_rx_packets_total{cell="5"} 1680000
...
# HELP aerial_cuphytcp_cplane_tx_bytes_total Aerial cuPHY-CP C-plane TX bytes
# TYPE aerial_cuphytcp_cplane_tx_bytes_total counter
aerial_cuphytcp_cplane_tx_bytes_total{cell="7"} 101048000
aerial_cuphytcp_cplane_tx_bytes_total{cell="6"} 101048000
```

(continues on next page)

(continued from previous page)

```

aerial_cuphycp_cplane_tx_bytes_total{cell="2"} 101048000
aerial_cuphycp_cplane_tx_bytes_total{cell="1"} 101048000
aerial_cuphycp_cplane_tx_bytes_total{cell="8"} 101048000
aerial_cuphycp_cplane_tx_bytes_total{cell="3"} 101048000
aerial_cuphycp_cplane_tx_bytes_total{cell="4"} 101048000
aerial_cuphycp_cplane_tx_bytes_total{cell="5"} 101048000

```

#### 1.5.4.17 Run an Additional Logging Stream Container

##### Note

The `nvlog_observer` and `nvlog_collect` are deprecated in 23-1.

- 1) By default the logs are stored in '/tmp' location. You can set the environment variable `AERIAL_LOG_PATH` to define a customized logfile path.
- 2) The moment the log size crosses 20GB, a new file gets created. Like `phy.log`, `phy.log.1`, `phy.log.2` ... `phy.log.7`.

#### 1.5.4.18 Run Multiple L2 Instances with Single L1 Instance

Rel-23-3 support static cell allocation for different L2 instances.

There's a known limitation that all cells need to be configured (by FAPI CONFIG.req) before any cell starts scheduling. With the duplicate Cell Config.request feature introduced in 23-4, the dynamic L2 instances can be supported without the above limitation but the cell config on each L2 instance must be the same.

Example: Run two L2 instances with 4 cells for each and one L1 instance with 8 cells.

- 1) Assign a different "prefix" in `nvipc` config for each L2 instance. The "prefix" is a string whose length should be less than 32.

```

# nvipc config yaml for each L2 instance

# For L2 instance 0: test_mac_config.yaml
prefix: nvipc

# For L2 instance 1: test_mac_config_1.yaml
prefix: nvipc1

```

The first testMAC instance uses the default `test_mac_config.yaml`. After it is configured properly, make a copy of `test_mac_config.yaml` and configure it for the second testMAC instance. To run multiple testMAC instances on the same machine, CPU cores, logger name, and OAM server port must be changed. The following are the example commands to configure the 2nd testMAC instance:

```

cp ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config.yaml ${cuBB_SDK}/cuPHY-CP/
↪testMAC/testMAC/test_mac_config_1.yaml
sed -i 's/prefix:./prefix: nvipc1/g' ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_
↪mac_config_1.yaml
sed -i 's/log_name:./log_name: testmac1.log/g' ${cuBB_SDK}/cuPHY-CP/testMAC/

```

(continues on next page)

(continued from previous page)

```

↪testMAC/test_mac_config_1.yaml
sed -i 's/oam_server_addr:./oam_server_addr: 0.0.0.0:50053/g' ${cuBB_SDK}/cuPHY-
↪CP/testMAC/testMAC/test_mac_config_1.yaml
sed -i '/sched_thread_config/{ N; N; N; s/cpu_affinity:[^\n]*/cpu_affinity: 14/g}
↪' ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config_1.yaml
sed -i '/recv_thread_config/{ N; N; N; s/cpu_affinity:[^\n]*/cpu_affinity: 15/g}'
↪${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config_1.yaml
sed -i '/builder_thread_config/{ N; N; N; s/cpu_affinity:[^\n]*/cpu_affinity: 16/
↪g}' ${cuBB_SDK}/cuPHY-CP/testMAC/testMAC/test_mac_config_1.yaml

```

- 2) Switch nvipc config to nvipc\_multi\_instances.yaml in L1.

```

# l2_adapter_config_XXX.yaml
nvipc_config_file: nvipc_multi_instances.yaml

```

- 3) Config “prefix” in L1 and assign L1 cells for each L2 instance.

Assume 8 cells are configured in cuphycontroller\_XXX.yaml, the indexes for them are 0 ~ 7.

```

L1 cells: 0 ~ 7
The 1st L2 instance cells: 0 ~ 3
The 2nd L2 instance cells: 4 ~ 7

```

Then configure:

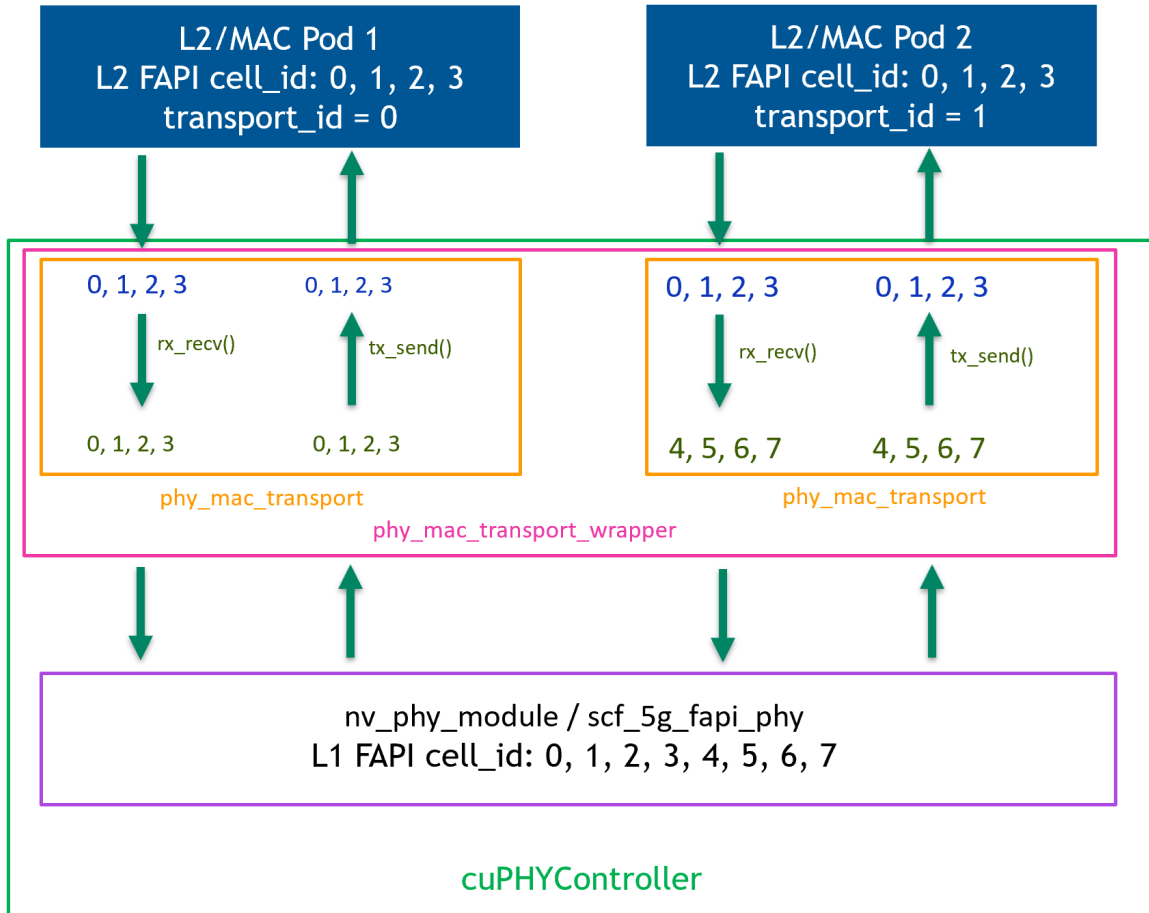
```

# nvipc_multi_instances.yaml
transport:
- transport_id: 0
  phy_cells: [0, 1, 2, 3]
  type: shm
  shm_config:
  prefix: nvipc
  ...

- transport_id: 1
  phy_cells: [4, 5, 6, 7]
  type: shm
  shm_config:
  prefix: nvipc1
  ...

```

The cell\_id map between L1 and L2 is maintained **in** cuphycontroller:



## 4) Run the test.

Use F08 8C\_60 for example:

```
sudo ./ru_emulator F08 8C_60

sudo -E ./cuphycontroller_scf F08_CG1

# Run the 1st test_mac instance with default config file: test_mac_config.yaml
sudo ./test_mac F08 8C_60 --cells 0xF0

# Run the 2nd test_mac instance with another config file: test_mac_config_1.yaml
sudo ./test_mac F08 8C_60 --cells 0xF0 --config test_mac_config_1.yaml
```

## 5) Review the 8 cells of throughput in the L1 and 4 cells of throughput in each L2 instance.

### 1.5.4.19 OAM Commands in Multiple L2 Instances

OAM commands do not change in the case of multiple L2 instances. Note the following:

- ▶ The “schedule\_total\_time” tolerance in Multi-L2 cases is a bit lower than Single-L2 cases. Please set schedule\_total\_time of Multi-L2 case to at least 20us lower than the same case for Single-L2. Recommended to set to 400000 for functionality test.

The minor difference in FAPI timing tolerance is expected because there are multiple NVIPC instances working in difference processes and additional SLOT.ind messages are added.

- ▶ There are two types of cell IDs used in L1:
  - ▶ **FAPI cell\_id**: cell instance index in each app. It starts from 0 and is unique in each L1/L2 app instance (but can be duplicated in different L2 app instances). It is also used as cell\_id / handle\_id in FAPI message.
  - ▶ **mplane\_id**: an arbitrary integer value that is configurable in cuphycontroller\_XXX.yaml and is used in cuPHYDriver.

The “cell\_id” in OAM commands is the mplane\_id not the FAPI cell\_id.

```
# cuphycontroller_XXX.yaml

cells:
- name: 0-RU 0 # FAPI cell_id is the cell instance index. For the first cell,
  ↪FAPI cell_id = 0
  cell_id: 1 # Here "cell_id" is actually "mplane_id" in source code. Current
  ↪default config is: mplane_id = FAPI cell_id + 1
```

In the following OAM command example, pass mplane\_id = 1 to select the first cell.

```
# Usage: aerial_cell_param_net_update.py cell_id dst_mac_addr vlan_tci

cd $cuBB_SDK/build/cuPHY-CP/cuphyoam && python3 $cuBB_SDK/cuPHY-CP/cuphyoam/
  ↪examples/aerial_cell_param_net_update.py 1 20:04:9B:9E:27:B3 E002
```

## 1.5.5. cuBB on NVIDIA Cloud Native Stack

NVIDIA Cloud Native Stack (formerly known as Cloud Native Core) is a collection of software that runs cloud native workloads on NVIDIA GPUs. This section describes how to install and run the Aerial cuBB software examples on NVIDIA Cloud Native Stack and related components to run Aerial cuBB.

### 1.5.5.1 Installation of NVIDIA Cloud Native Stack

Prerequisite: The server must already have the OS, NVIDIA Driver, and other configuration as described in *Installing Tools on Grace Hopper* or *Installing Tools on Dell R750*.

The steps to install NVIDIA Cloud Native Stack follows the [NVIDIA Cloud Native Stack v13.0 installation guide on GitHub](#), starting with section “Installing Container Runtime”, with the following additional notes:

- ▶ Select containerd when given the choice between containerd or CRI-O in the install guide.
- ▶ For running an ru-emulator on a server without a GPU, it is necessary to remove/comment out the “BinaryName” field from /etc/containerd/config.toml on that server.



If this step is not done, an ru-emulator failed to start error message can occur

```
State:      Terminated
Reason:     StartError
Message:    failed to create containerd task: failed to create shim task: OCI
↳runtime create failed: runc create failed: unable to start container process:
↳error during container init: error running hook #0: error running hook: exit
↳status 1, stdout: , stderr: Auto-detected mode as 'legacy'
nvidia-container-cli: initialization error: nvidia error: driver not loaded: unknown
Exit Code:  128
Started:    Thu, 01 Jan 1970 00:00:00 +0000
Finished:   Wed, 17 Jan 2024 05:25:27 +0000
```

► Enable k8s CPU Manager, Topology Manager, and Memory Manager.

1. Update each worker node's `/var/lib/kubelet/config.yaml`. The file to use depends on the server type.

```
# For Aerial Devkit servers
$ cat <<EOF | sudo tee -a /var/lib/kubelet/config.yaml
# Additional Configuration

# Feature Gates
featureGates:
  MemoryManager: true

# CPU Manager Configuration
cpuManagerPolicy: "static"
cpuManagerPolicyOptions:
  full-pcpus-only: "true"
reservedSystemCPUs: 0-2,22-23

# Topology Manager Configuration
topologyManagerPolicy: "restricted"
topologyManagerScope: "container"

# Memory Manager Configuration
memoryManagerPolicy: "Static"
reservedMemory:
  - numaNode: 0
    limits:
      memory: 100Mi
EOF

# for Dell R750 servers
$ cat <<EOF | sudo tee -a /var/lib/kubelet/config.yaml
# Additional Configuration

# Feature Gates
featureGates:
  MemoryManager: true

# CPU Manager Configuration
cpuManagerPolicy: "static"
cpuManagerPolicyOptions:
  full-pcpus-only: "true"
reservedSystemCPUs: 0-3
```

(continues on next page)

(continued from previous page)

```
# Topology Manager Configuration
topologyManagerPolicy: "restricted"
topologyManagerScope: "pod"

# Memory Manager Configuration
memoryManagerPolicy: "Static"
reservedMemory:
  - numaNode: 0
    limits:
      memory: 50Mi
  - numaNode: 1
    limits:
      memory: 50Mi
EOF
```

2. Drain each worker node.

```
# Run from k8s master or other server where you have the kube config
kubectl drain $nodeName --force --ignore-daemonsets
```

3. Restart each worker node's kubelet.

```
# Run on worker node
sudo systemctl stop kubelet
sudo rm -f /var/lib/kubelet/cpu_manager_state
sudo rm -f /var/lib/kubelet/memory_manager_state
sudo systemctl start kubelet
sudo systemctl status kubelet
```

4. Confirm kubelet status, verify that it is healthy.

```
$ systemctl status kubelet
kubelet.service - kubelet: The Kubernetes Node Agent
   Loaded: loaded (/lib/systemd/system/kubelet.service; enabled; vendor preset:
   → enabled)
   Drop-In: /etc/systemd/system/kubelet.service.d
           10-kubeadm.conf
   Active: active (running) since Thu 2023-10-12 19:36:05 UTC; 5s ago
```

5. Uncordon the node.

```
# Run from k8s master or other server where you have the kube config
kubectl uncordon $nodeName
```

6. Setup a registry secret called "regcred" to be able to pull from \$YourPrivateRegistry container registry. Follow the procedure described in the [Kubernetes documentation](#). If you are using \$YourPrivateRegistry=nvcr.io, please remember to generate an API Key from the [NGC API-Key Setup Portal](#) if you don't already have one.

### 1.5.5.2 Building Aerial Binary Container

This section describes how to build an Aerial binary container for the cuphycontroller\_scf, test\_mac, and ru\_emulator applications, along with some example scenario test vectors.

1. Extract the build script.

```
mkdir -p cuPHY-CP/
docker pull nvcr.io/ea_aerial_sdk/aerial:24-2-cubb
docker run --rm -u `id -u` -v ./staging nvcr.io/ea_aerial_sdk/aerial:24-2-cubb
↪ cp -a /opt/nvidia/cuBB/cuPHY-CP/container /staging/cuPHY-CP/
```

Install dependencies

```
sudo apt update
sudo apt install python3-pip -y
pip3 install hpccm
```

2. Build the binary container and push to your private container repository.

```
AERIAL_BUILD_IMAGE=nvcr.io/ea_aerial_sdk/aerial:24-2-cubb AERIAL_RELEASE_REPO=
↪ $YourPrivateRepo/ AERIAL_RELEASE_VERSION_TAG=$YourTag ./cuPHY-CP/container/
↪ build_binary.sh
docker push $YourPrivateRepo/aerial_binary:$YourTag-amd64
```

### 1.5.5.3 Deploying Binary Container using Helm Chart

Configure the NGC cli tool - follow the steps in <https://ngc.nvidia.com/setup/installers/cli>

Login to NGC

```
$ ngc config set
```

2. Fetch the Helm Chart from NGC.

```
ngc registry chart pull ea_aerial_sdk/aerial-l1
ngc registry chart pull ea_aerial_sdk/aerial-ru-emulator
```

3. Create value overload files specific to your environment. You must change the following values:

- ▶ YourRUEmulatorNodeName
- ▶ YourPrivateRepo
- ▶ YourTag
- ▶ <MAC Address of DU's FH Port>
- ▶ YourDUNodeName

```
$ cat <<EOF | tee override-ru-emulator-binary.yaml
# Deployment customization
extraSpec:
  nodeName: "YourRUEmulatorNodeName"

image:
  repository: YourPrivateRepo/
  name: aerial_binary
```

(continues on next page)

(continued from previous page)

```

pullPolicy: Always
# Overrides the image tag whose default is the chart appVersion.
tag: "YourTag"

peerethaddr: "<MAC Address of DU's FH Port>"

# Spacing is critical below
extraSetup: |
  \# ru-emulator extra setup\`

      sed -i "s/enable_beam_forming:./enable_beam_forming: 1/" ../cuPHY-CP/ru-
↪emulator/config/config_dyn.yaml

      \# Configure NIC PCIe Address\`
      sed -i "s/nic_interface:./nic_interface: 0000:3b:00.0/" ../cuPHY-CP/ru-
↪emulator/config/config_dyn.yaml

EOF

$ cat <<EOF | tee override-l1-binary.yaml
# Deployment customization
extraSpec:
  nodeName: "YourDUNodeName"

image:
  repository: YourPrivateRepo/
  name: aerial_binary
  pullPolicy: Always
  # Overrides the image tag whose default is the chart appVersion.
  tag: "YourTag"

enableTestMACContainer: 1

# Spacing is critical below
extraSetup: |
  \# Aerial L1 extra setup\`

      \# Launch pattern related configuration\`
      sed -i "s/cell_group_num: ./cell_group_num: 16/" ../cuPHY-CP/
↪cuphycontroller/config/cuphycontroller_dyncore.yaml;
      sed -i "s/pusch_nMaxPrb: ./pusch_nMaxPrb: 136/" ../cuPHY-CP/
↪cuphycontroller/config/cuphycontroller_dyncore.yaml;

      \# 3GPP conformance\`
      sed -i "s/pusch_tdi:./pusch_tdi: 1/" ../cuPHY-CP/cuphycontroller/config/
↪cuphycontroller_dyncore.yaml;
      sed -i "s/pusch_cfo:./pusch_cfo: 1/" ../cuPHY-CP/cuphycontroller/config/
↪cuphycontroller_dyncore.yaml;
      sed -i "s/pusch_to:./pusch_to: 1/" ../cuPHY-CP/cuphycontroller/config/
↪cuphycontroller_dyncore.yaml;
      sed -i "s/puxch_polarDcdrListSz:./puxch_polarDcdrListSz: 8/" ../cuPHY-
↪CP/cuphycontroller/config/cuphycontroller_dyncore.yaml;

      \# Configure NIC PCIe Address\`
      sed -i "s/ nic:./ nic: 0000:cc:00.1/" ../cuPHY-CP/cuphycontroller/
↪config/cuphycontroller_dyncore.yaml;

```

(continues on next page)

(continued from previous page)

```
# Spacing is critical below
extraTestMACSetup: |
  \# testMAC extra setup\`

  \#sed -i "s/test_slots: 0/test_slots: 100000/" ../cuPHY-CP/testMAC/
↪testMAC/test_mac_config_dyncore.yaml;\`
  sed -i "s/schedule_total_time: 0/schedule_total_time: 470000/" ../cuPHY-
↪CP/testMAC/testMAC/test_mac_config_dyncore.yaml;
  sed -i "s/fapi_delay_bit_mask: 0/fapi_delay_bit_mask: 0xF/" ../cuPHY-CP/
↪testMAC/testMAC/test_mac_config_dyncore.yaml;
  sed -i "s/builder_thread_enable: 0/builder_thread_enable: 1/" ../cuPHY-
↪CP/testMAC/testMAC/test_mac_config_dyncore.yaml;
EOF
```

#### 4. Deploy the Helm Chart.

```
helm install aerial-ru-emulator-test aerial-ru-emulator-0.20234.0.tgz -f override-
↪ru-emulator-binary.yaml
helm install aerial-l1-test aerial-l1-0.20234.0.tgz -f override-l1-binary.yaml
```

#### 5. View the logs for each container.

```
# Run in separate windows
kubectl logs aerial-l1-test -f
kubectl logs aerial-l1-test -c aerial-testmac-ctr -f
kubectl logs aerial-ru-emulator-test -f
```

#### 6. Remove the Helm Chart and destroy the pods when finished.

```
helm uninstall aerial-l1-test
helm uninstall aerial-ru-emulator-test
```

### 1.5.5.4 Theory of Operation

At pod deployment time, k8s dynamically assigns dedicated CPU cores to the Aerial L1 cuphycontroller\_scf container and the testMAC container (if it is deployed). When the container starts up, the \$cuBB\_SDK/cubb\_scripts/autoconfig/auto\_assign\_cores.py script runs to map the k8s-assigned cores to the various Aerial functions. The following template configuration YAML files are used by the auto\_assign\_cores.py script:

- ▶ \$cuBB\_SDK/cuPHY-CP/cuphycontroller/config/cuphycontroller\_\${configL1}.yaml -> \$cuBB\_SDK/cuPHY-CP/cuphycontroller/config/cuphycontroller\_dyncore.yaml
- ▶ \$cuBB\_SDK/cuPHY-CP/cuphycontroller/config/\${l2adapter\_filename} -> \$cuBB\_SDK/cuPHY-CP/cuphycontroller/config/l2\_adapter\_dyncore.yaml
- ▶ \$cuBB\_SDK/cuPHY-CP/testMAC/testMAC/\${configMAC} -> \$cuBB\_SDK/cuPHY-CP/testMAC/testMAC/test\_mac\_config\_dyncore.yaml

The variables used above come from:

- ▶ \$cuBB\_SDK: Environment variable defined in container
- ▶ \$configL1: Helm chart aerial-l1/values.yaml (or override-l1-binary.yaml if overridden) variable 'configL1'

- ▶ \$l2adapter\_filename: YAML configuration parameter 'l2adapter\_filename' defined in the template cuphycontroller configuration yaml.
- ▶ \$configMAC: Helm chart aerial-l1/values.yaml (or override-l1-binary.yaml if overridden) variable 'configMAC'

An example run of the auto\_assign\_cores.py script for the aerial-l1-ctr container is:

```

Detected HT Enabled
Detected Multiple NUMA Nodes: [0, 1]. Will use node 1 for scheduling.
OS core affinity: [5, 7, 9, 11, 13, 15, 17, 53, 55, 57, 59, 61, 63, 65]
OS core affinity for numa node 1: [5, 7, 9, 11, 13, 15, 17, 53, 55, 57, 59, 61, 63,
↪65]
OS isolated cores: [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,
↪22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42,
↪43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63,
↪64, 65, 66, 67,
68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88,
↪89, 90, 91, 92, 93, 94, 95]
Tentative primary cores: [5, 7, 9, 11, 13, 15, 17]
Cuphycontroller core assignment strategy for HT enabled:
  * 1 low priority primary core (shared with dpdk EAL), HT sibling for h2d_copy thread
  * {args.workers_ul_count} UL worker primary cores, HT siblings idle
  * {args.workers_dl_count} DL worker primary cores, HT siblings idle
  * 1 L2A timer thread primary core, HT sibling for L2A msg processing thread
Need 7 physical cores (plus 0 reserved), potential affinity for 7 isolated physical
↪cores
+-----+-----+-----+-----+
↪-----+
| Primary |           Primary Core | Sibling |
↪Sibling Core |           |         |
| Core Number |           Uses | Core Number |
↪Uses |           |         |
+-----+-----+-----+-----+
↪-----+
|    5    | low priority threads (inc. DPDK EAL) |    53   | H2D copy
↪      |           |         |
+-----+-----+-----+-----+
↪-----+
|    7    | UL Worker |           |    55   | [idle]
↪      |           |         |
+-----+-----+-----+-----+
↪-----+
|    9    | UL Worker |           |    57   | [idle]
↪      |           |         |
+-----+-----+-----+-----+
↪-----+
|   11    | DL Worker |           |    59   | [idle]
↪      |           |         |
+-----+-----+-----+-----+
↪-----+
|   13    | DL Worker |           |    61   | [idle]
↪      |           |         |
+-----+-----+-----+-----+
↪-----+
|   15    | DL Worker |           |    63   | [idle]
↪      |           |         |
+-----+-----+-----+-----+
↪-----+

```

(continues on next page)

(continued from previous page)

```

|    17    | L2A timer                |    65    | L2A msg
↪processing |                          |          |
+-----+-----+-----+-----+
↪-----+
Parsing cuphycontroller configuration template: /opt/nvidia/cuBB/cuPHY-CP/
↪cuphycontroller/config/cuphycontroller_F08_R750.yaml
Writing cuphycontroller configuration: /opt/nvidia/cuBB/cuPHY-CP/cuphycontroller/
↪config/cuphycontroller_dyncore.yaml
Parsing l2adapter configuration template: /opt/nvidia/cuBB/cuPHY-CP/cuphycontroller/
↪config/l2_adapter_config_F08_R750.yaml
Writing l2adapter configuration: /opt/nvidia/cuBB/cuPHY-CP/cuphycontroller/config/l2_
↪adapter_config_dyncore.yaml

```

An example run of the auto\_assign\_cores.py script for the aerial-l1-ctr container is:

```

Detected HT Enabled
Detected Multiple NUMA Nodes: [0, 1]. Will use node 1 for scheduling.
OS core affinity: [19, 21, 23, 67, 69, 71]
OS core affinity for numa node 1: [19, 21, 23, 67, 69, 71]
OS isolated cores: [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,
↪22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42,
↪43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63,
↪64, 65, 66, 67,
68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88,
↪89, 90, 91, 92, 93, 94, 95]
Tentative primary cores: [19, 21, 23]
testMAC core assignment strategy:
  * 1 low priority primary core, HT sibling idle
  * 1 mac_recv thread primary core, HT sibling idle
  * 1 builder thread primary core, HT sibling idle
Need 3 physical cores (plus 0 reserved), potential affinity for 3 isolated physical
↪cores
+-----+-----+-----+-----+
↪-----+
| Primary | Primary Core | Sibling |
↪Sibling Core | Uses | Core Number |
| Core Number | Uses | Core Number |
↪Uses |
+-----+-----+-----+-----+
↪-----+
|    19    | [testmac] low priority threads |    67    | [idle]
↪
+-----+-----+-----+-----+
↪-----+
|    21    | [testmac] recv |    69    | [idle]
↪
+-----+-----+-----+-----+
↪-----+
|    23    | [testmac] builder |    71    | [idle]
↪
+-----+-----+-----+-----+
↪-----+
Parsing testmac configuration template: /opt/nvidia/cuBB/cuPHY-CP/testMAC/testMAC/
↪test_mac_config.yaml
Writing testmac configuration: /opt/nvidia/cuBB/cuPHY-CP/testMAC/testMAC/test_mac_
↪config_dyncore.yaml

```

## 1.6. cuBB Integration Guide

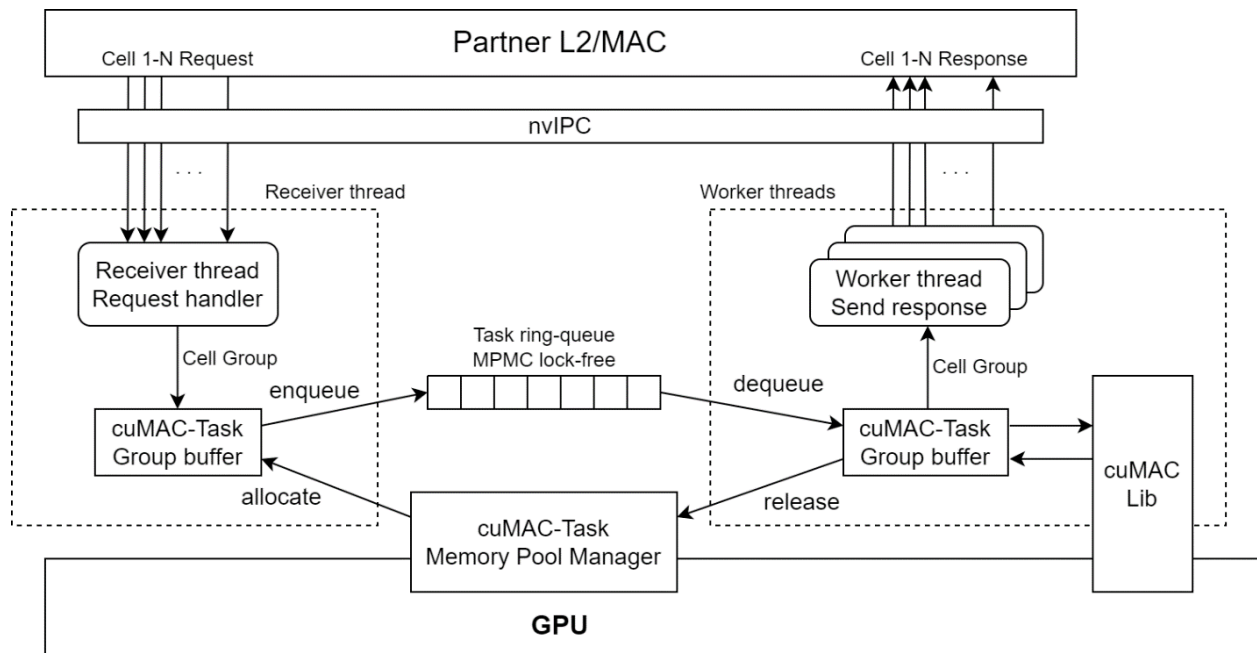
The cuBB Integration Guide contains the following sections:

<b>cuMAC-CP</b>	A specification for the CUDA RAN MAC Scheduler Control Plane (cuMAC-CP), which provides an interface between 5G/6G L2 (MAC Scheduler Functions) and Aerial cuMAC library.
<b>NVIPC</b>	An API specification for NVIPC, an NVIDIA messaging standard for communication between two processes on the same system.

### 1.6.1. cuMAC-CP

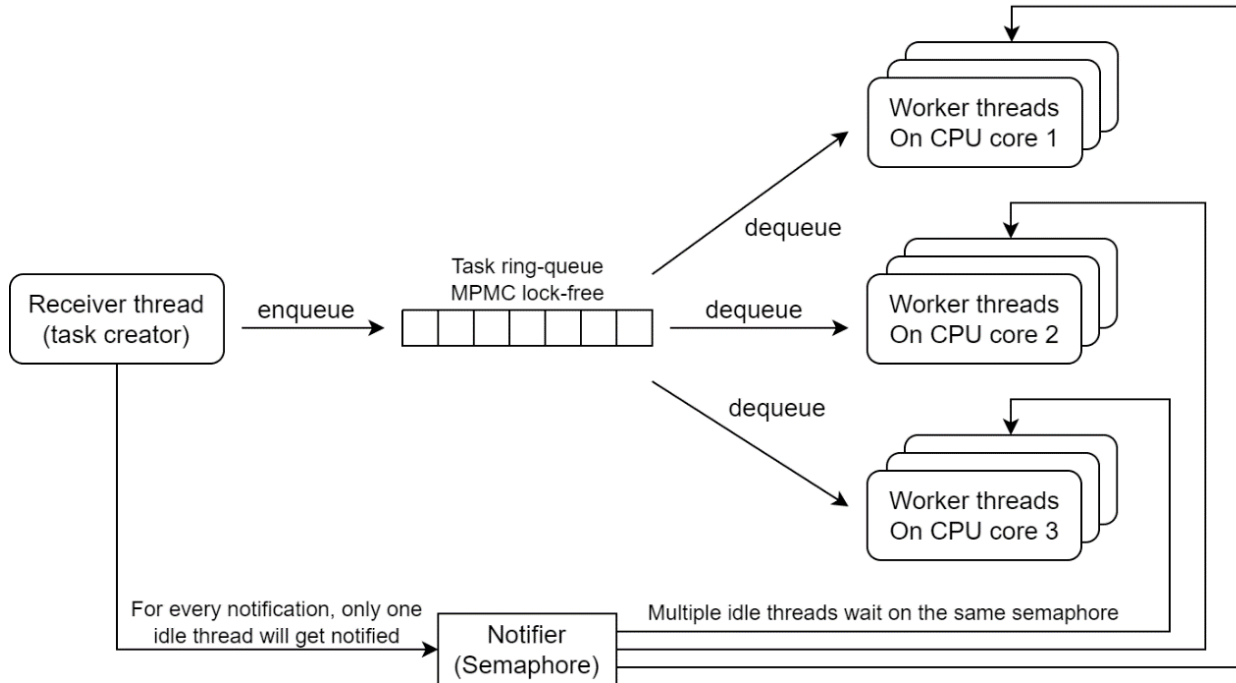
CUDA RAN MAC Scheduler Control Plane (cuMAC-CP) is a process which provides an interface between 5G/6G L2 (MAC Scheduler Functions) and Aerial cuMAC library, with scheduler functions accelerated on GPU. It accepts L2/MAC scheduling request per cell, translates to cuMAC tasks and call cuMAC lib APIs to process on GPU. After processing is finished, it returns the scheduling results to L2/MAC by response message per cell.

Below is the cuMAC-CP architecture diagram.



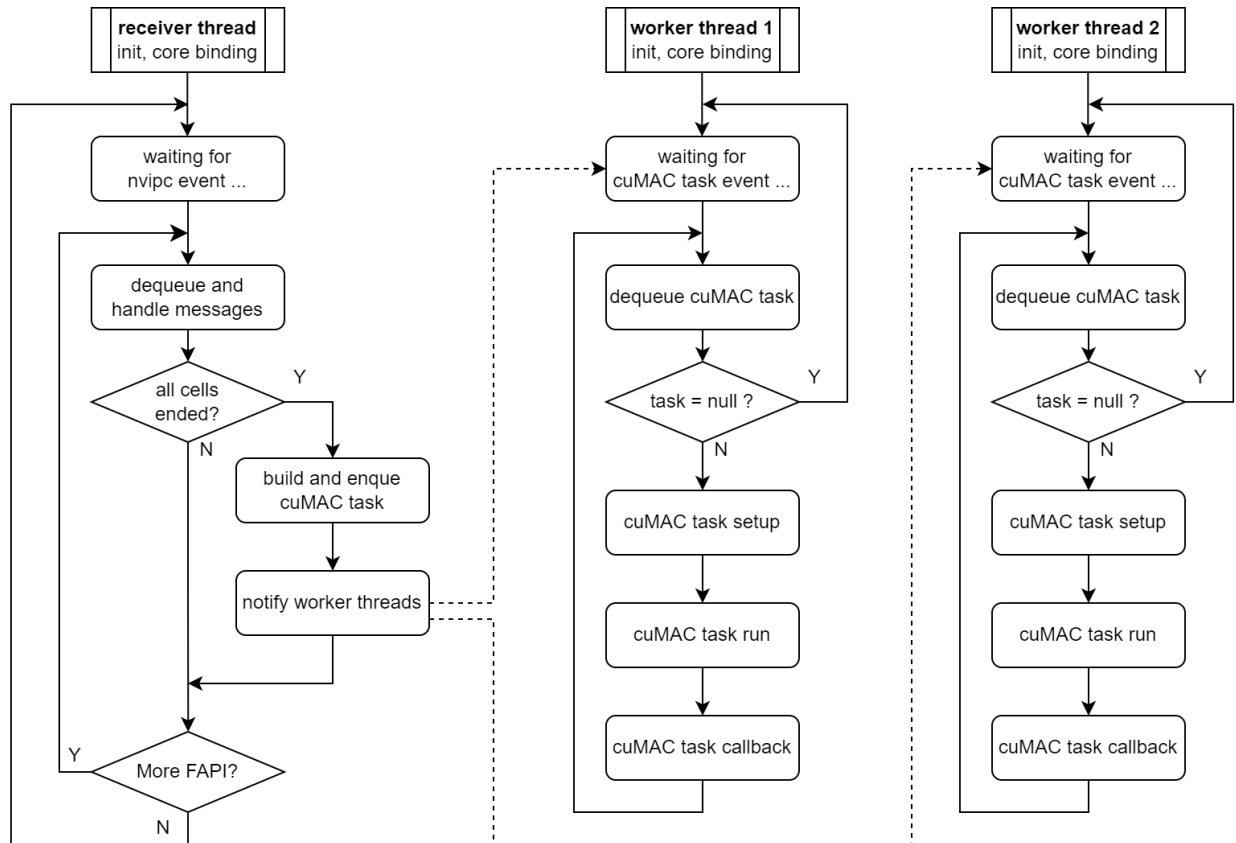
cuMAC-CP has 1 receiver thread and multiple worker threads which need to be bound to dedicated CPU cores. The thread model is as below. CPU core numbers are configurable by yaml file (24-2 release supports only 1 worker thread per core).





The receiver thread allocates a `cumac_task` object and necessary data buffers for each slot. Once cuMAC-CP received schedule request messages from L2/MAC for all cells, the receiver thread assembles them into cell group, populates the `cumac_task` object and pushes it into the lock-free task queue, then increase the semaphore to notify the worker threads.

All worker threads wait on the same semaphore after initialization. Every time the semaphore is increased by the receiver thread, one worker thread will get the semaphore, dequeue cuMAC task and call cuMAC lib APIs to process it. After processing is finished, the worker thread creates per cell response messages and sends them to L2. Below is the program flow chart.



### 1.6.1.1 cuMAC-CP API Procedures

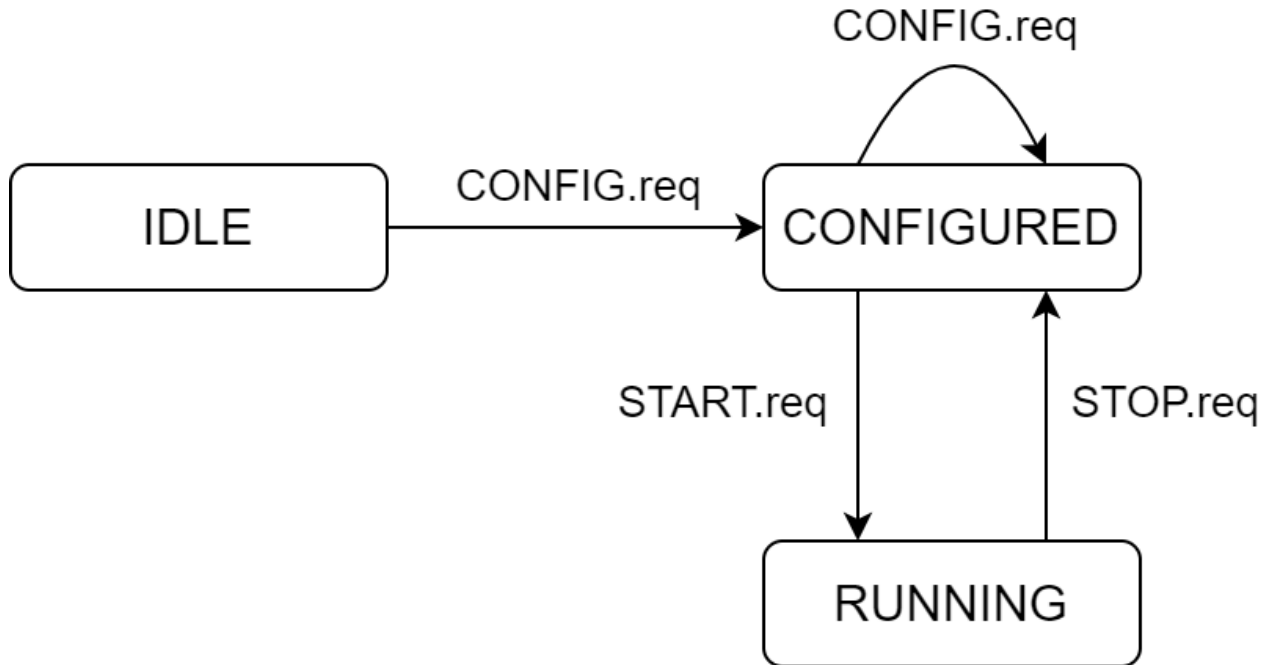
This section gives an overview of the procedures which use the cuMAC-CP API. These procedures are split into two groups: configuration procedures and slot procedures. Configuration procedures handle the management of the cuMAC-CP resource initialization or re-initialization and are expected to occur infrequently. Slot procedures determine the structure of each slot and operate with a periodicity based on the sub-carrier spacing numerology, namely 125us, 250us, 500us or 1ms periodicity.

#### 1.6.1.1.1 Configuration Procedures

The following are configuration procedures supported by the cuMAC-CP API:

- Initialization

cuMAC-CP have implemented three states for each cell: IDLE, CONFIGURED and RUNNING. The status transition of a cell can be executed by configuration procedures as shown in the figure below.



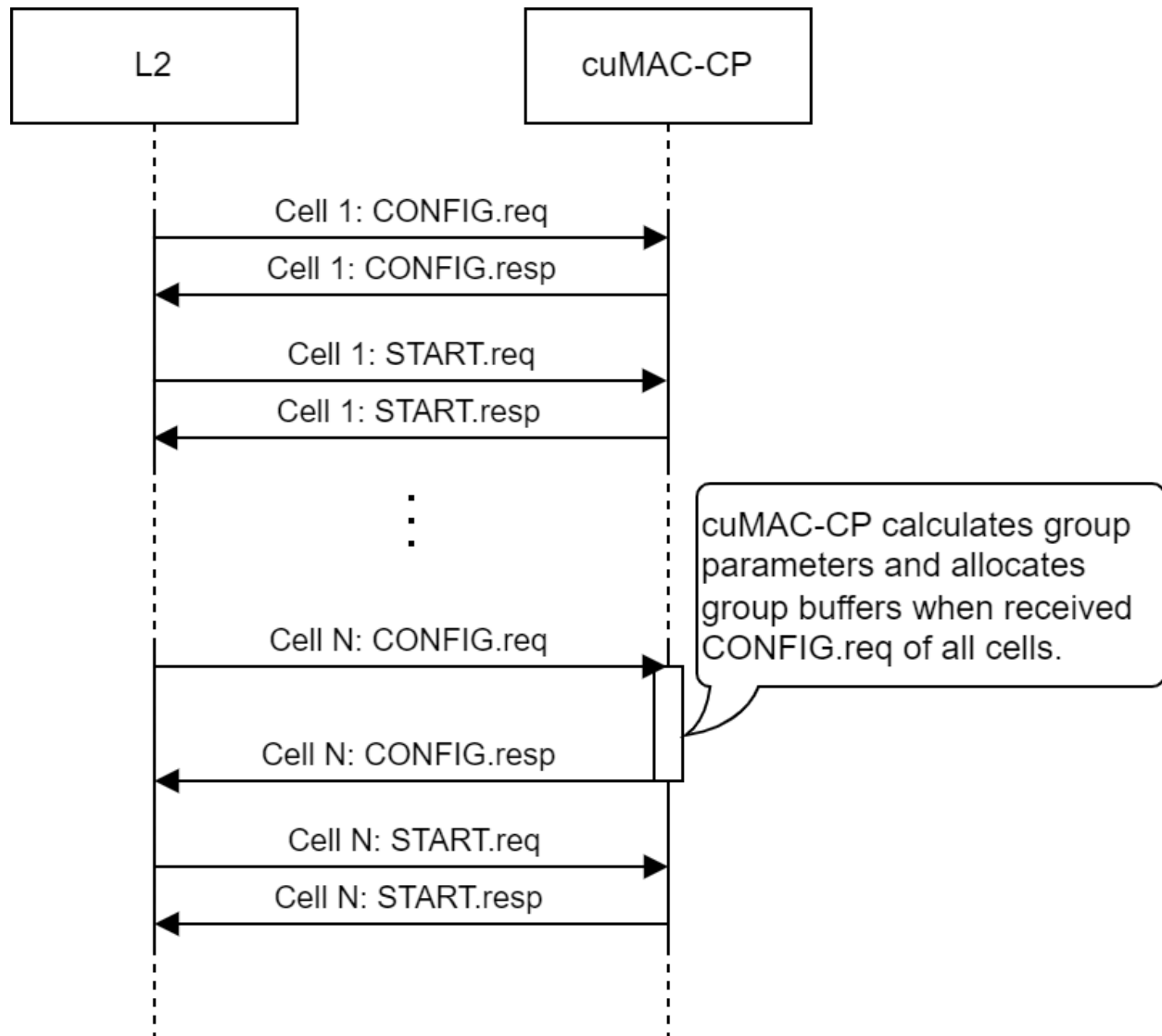
According to the above figure, the supported configuration messages for each state are listed in the table below.

Idle State	Configured State	Running State
CONFIG.request	CONFIG.request	STOP.request
	START.request	

#### 1.6.1.1.2 Initialization procedure

The initialization procedure includes 2 steps, as illustrated in the following sequence chart:

- ▶ Each cell should send a CONFIG.request message and expect to receive a CONFIG.response message.
- ▶ Each cell should send a START.request message and expect to receive a START.response message.

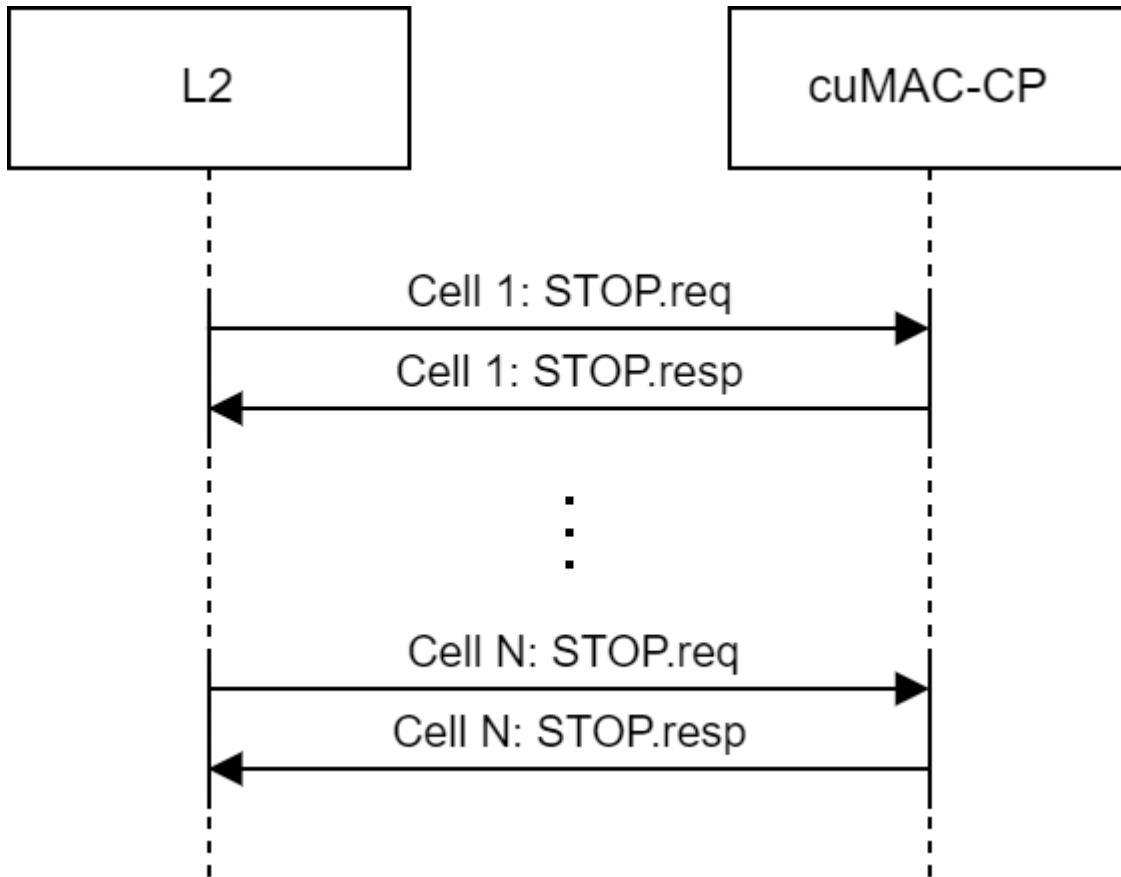


Note the following:

- ▶ For each cell, START . request should be sent after receiving CONFIG.response.
- ▶ Different cells are independent from each other so message order between different cells is not mandatory.
- ▶ After receiving CONFIG . request for all cells, cuMAC-CP will calculate required memory size and allocate GPU memory for the whole cell group. The parameters used in SLOT messages should be the same as in the CONFIG . request.
- ▶ All cells are to be initialized before starting slot procedures. SLOT messages which are sent before all cells initialized will be ignored.

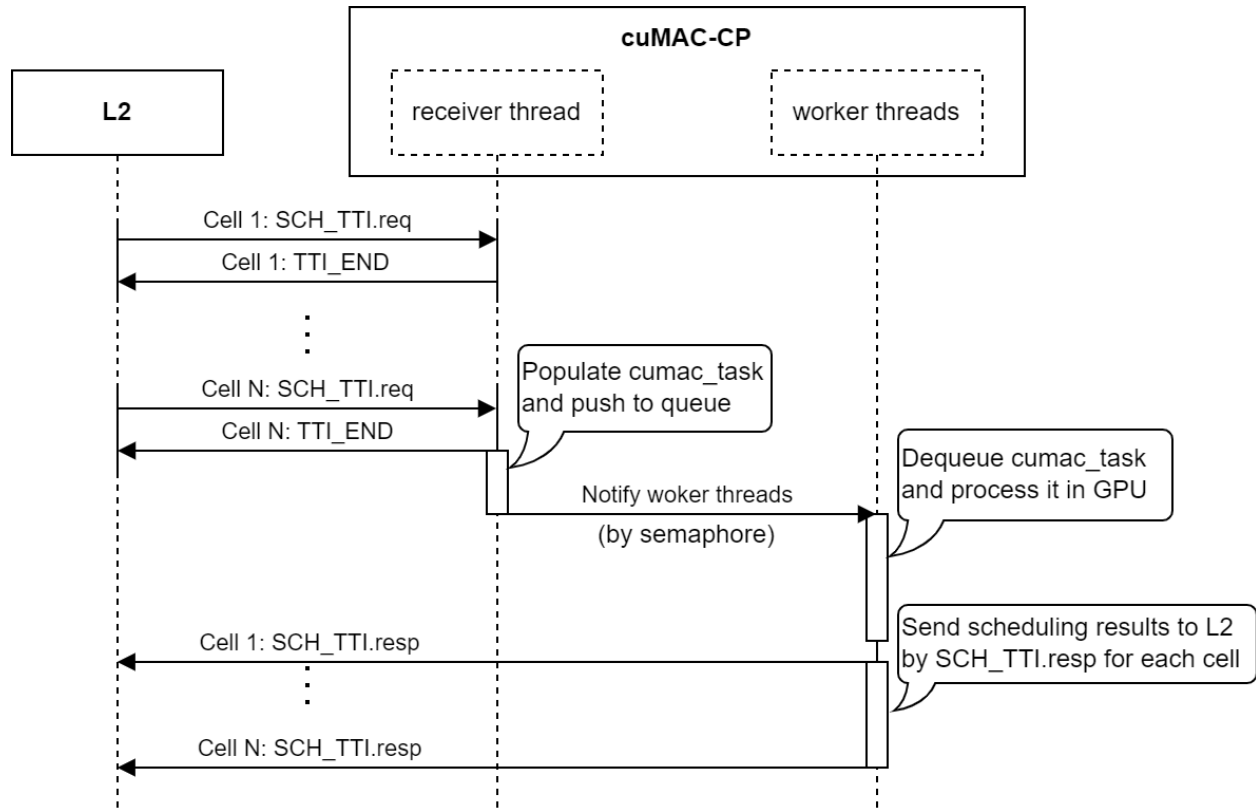
### 1.6.1.1.3 Termination procedure

For each cell, L2 sends STOP.request and expects to receive STOP.response. The target cell will be disabled in cuMAC-CP.



### 1.6.1.1.4 SLOT procedures

After all slots are initialized, L2 can start sending SLOT messages. Currently there are 2 messages to send for each cell: SCH\_TTI.request and SCH\_TTI.response. cuMAC-CP will wait for all enabled cells message finishing then populate a cumac\_task structure and process it. If successfully processed, cuMAC-CP will send a SCH\_TTI.response to L2 for each cell.



### 1.6.1.2 cuMAC-CP API Messages

This section provides a description of the cuMAC-CP API message formats. It defines the PHY API message header, the message bodies and the error codes associated with the cuMAC-CP API.

The cuMAC-CP API messages include 2 kinds: configuration procedure messages and slot procedure messages. Both kinds of messages start with the generic header as below table

Field	Type	Description
message_count	uint8_t	Number of messages included in PHY API message
handle_id	uint8_t	An opaque handle (purpose not defined, however, usages may include a PHY ID or Carrier ID)
type_id	uint16_t	Message type ID
body_len	uint32_t	Length of message body (bytes)

All the message IDs are listed as below.

Message	Value	Message Body Definition
CONFIG.request	0x02	
CONFIG.response	0x03	
START.request	0x04	
STOP.request	0x05	
STOP.response	0x06	
ERROR.indication	0x07	TODO
START.response	0x08	
RESERVED	0x09-0x7F	
SCH_TTI.request	0x82	
TTI_END	0x83	
SCH_TTI.response	0x8F	
TTI_ERR.indication	0x90	TODO
RESERVED	0x91-0xFF	

### 1.6.1.2.1 Configuration Procedure Messages

Configuration procedure messages are followed by configuration message bodies.

#### 1.6.1.2.1.1 CONFIG.request

The CONFIG.request message body is defined in cuMAC per cell message definition.

Field	Type	Description
nMaxCell	uint8_t	A constant integer for the maximum number of cells in the cell group. Value: 0 -> 255
nMaxActUePer-Cell	uint16_t	A constant integer for the maximum number of active UEs per cell. Value: 0 -> 65535
nMaxSchUePer-Cell	uint8_t	A constant integer for the maximum number of UEs that can be scheduled per TTI per cell. Value: 0 -> 255
nMaxPrg	uint16_t	A constant integer for the maximum number of PRGs for allocation in each cell Value: 0 -> 65535

continues on next page

Table 33 – continued from previous page

Field	Type	Description
nPrbPerPrg	uint8_t	A constant integer for the number of PRBs per PRG (PRB group) Value: 0 -> 255
nMaxBsAnt	uint8_t	A constant integer for the maximum number of BS antenna ports. Value: 0 -> 255
nMaxUeAnt	uint8_t	A constant integer for the maximum number of UE antenna ports. Value: 0 -> 255
scSpacing	uint32_t	Subcarrier spacing of the carrier. Value: 15000, 30000, 60000, 120000 (Hz)
allocType	uint8_t	Indicator for type-0 or type-1 PRG allocation. Value: 0: type-0 allocation 1: type-1 allocation
precoderType	uint8_t	Indicator for the precoder type. Value: 0: No precoding 1: SVD precoding
receiverType	uint8_t	Indicator for the receiver type. Value: 0: MMSE receiver 1: MMSE-IRC receiver
colMajChanAccess	uint8_t	Indicator for whether the estimated narrow-band SRS channel matrices are stored in column-major order or in row-major order. Value: 0: row-major 1: column-major
betaCoeff	float	Coefficient for adjusting the cell-edge UEs' performance in multi-cell scheduling. Value: non-negative real number. The default value is 1.0, representing pure proportional-fairness scheduling.
sinValThr	float	Singular value threshold for layer selection. Value: in (0, 1). Default value is 0.1
corrThr	float	Channel vector correlation value threshold for layer selection. Value: in (0, 1). Default value is 0.5
prioWeightStep	uint16_t	Step size for UE priority weight increment per TTI if UE does not get scheduled. For priority-based UE selection. Value: 0 -> 65535. Default is 100



#### 1.6.1.2.1.2 CONFIG.response

The message body is as below.

Field	Type	Description
error_code	uint8_t	0: no error. Other: TBD

#### 1.6.1.2.1.3 START.request

The message body is as below.

Field	Type	Description
start_param	uint8_t	Reserved, not used yet.

#### 1.6.1.2.1.4 START.response

The message body is as below.

Field	Type	Description
error_code	uint8_t	0: no error. Other: TBD

#### 1.6.1.2.2 Slot procedure messages

Slot procedures have an additional SFN/SLOT header as below table, then followed by slot message bodies.

Name	Type	Description
sfn	uint16_t	SFN number
slot	uint16_t	SLOT number

## 1.6.1.2.2.1 SCH\_TTI.request

Field	Type	Description
taskBitMask	uint32_t	Indicate which cuMAC tasks to be scheduled. Each bit represents 1 task type: 0x01: multiCellUeSelection 0x02: multiCellScheduler 0x04: multiCellLayerSel 0x08: mcsSelectionLUT Value: 0x1, 0x3, 0x7, 0xF
cellID	uint16_t	Cell ID. Value: 0 -> 65535
ULDLSch	uint8_t	Indication for UL/DL scheduling. Value: 0: UL scheduling 1: DL scheduling
nActiveUe	uint16_t	Total number of active UEs in the cell. Value: 0 -> 65535
nSrsUe	uint16_t	The number of UEs in the cell that have refreshed SRS channel estimates. Value: 0 -> 65535
nPrbGrp	uint16_t	The number of PRGs that can be allocated for the current TTI, excluding the PRGs reserved for HARQ re-transmissions. Value: 0 -> nMaxPrg
nBsAnt	uint8_t	Number of BS antenna ports. Value: 0 -> nMaxBsAnt
nUeAnt	uint8_t	Number of UE antenna ports. Value: 0 -> nMaxUeAnt
sigmaSqrd	float	Noise variance. Value: noise variance value in watts
offsets	struct	Buffer offset for each data: 0xFFFFFFFF: buffer not used (Invalid). Other: buffer offset based on nvipc data_buf

The data buffers are populated in NVIPC DATA buffer. The “offsets” structure defines the buffer offsets of all the buffers.

Note: not all buffers are used. The offsets of not used buffers should be set to 0xFFFFFFFF. For each task type, the required buffers are listed in the following tables.

Offset name	Offset Type	Buffer description
CRNTI	uint32_t	C-RNTIs of all active UEs in the cell

continues on next page

Table 35 – continued from previous page

Offset name	Offset Type	Buffer description
srsCRNTI	uint32_t	C-RNTIs of the UEs that have refreshed SRS channel estimates in the cell.
prgMsk	uint32_t	Bit map for the availability of each PRG for allocation
postEqSinr	uint32_t	Array of the per-PRG per-layer post-equalizer SINRs of all active UEs in the cell
wbSinr	uint32_t	Array of wideband per-layer post-equalizer SINRs of all active UEs in the cell
estH_fr	uint32_t	For FP32. Array of the subband (per-PRG) SRS channel estimate coefficients for all active UEs in the cell
estH_fr_half	uint32_t	For FP16. Array of the subband (per-PRG) SRS channel estimate coefficients for all active UEs in the cell
prdMat	uint32_t	Array of the precoder/beamforming weights for all active UEs in the cell
detMat	uint32_t	Array of the detector/beamforming weights for all active UEs in the cell
sinVal	uint32_t	Array of the per-UE, per-PRG, per-layer singular values obtained from the SVD of the channel matrix
avgRatesActUe	uint32_t	Array of the long-term average data rates of all active UEs in the cell
prioWeightActUe	uint32_t	For priority-based UE selection. Priority weights of all active UEs in the cell
tbErrLastActUe	uint32_t	TB decoding error indicators of all active UEs in the cell
newDataActUe	uint32_t	Indicators of initial transmission/retransmission for all active UEs in the cell
allocSolLastTxActUe	uint32_t	The PRG allocation solution for the last transmissions of all active UEs in the cell
mcsSelSolLastTxActUe	uint32_t	MCS selection solution for the last transmissions of all active UEs in the cell
layerSelSolLastTxActUe	uint32_t	Layer selection solution for the last transmissions of all active UEs in the cell

The data buffer details are described below.

Field	Type	Description
CRNTI	uint16_t[nActiveUe]	C-RNTIs of all active UEs in the cell. Value of each element: Denote uldx = 0, 1, ..., nActiveUe-1 as the active UE index in the cell. CRNTI[uldx] is the uldx-th active UE's C-RNTI.
srsCRNTI	uint16_t[nSrsUe]	C-RNTIs of the UEs that have refreshed SRS channel estimates in the cell. Value of each element: Denote uldx = 0, 1, ..., nSrsUe-1 as the index of the UE with refreshed SRS in the cell. srsCRNTI[uldx] is the uldx-th UE's C-RNTI.
prgMsk	uint8_t[nPrbGrp]	Bit map indicating the availability of each PRG for allocation. Value of each element: Denote prgIdx = 0, 1, ..., nPrbGrp-1 as the PRG index prgMsk[prgIdx] is the availability indicator for the prgIdx-th PRG. 0 - unavailable, 1 - available
postEqSinr	float[nActiveUe* nPrb-Grp*nUeAnt]	Array of the per-PRG per-layer post-equalizer SINRs of all active UEs in the cell. Value of each element: Denote uldx = 0, 1, ..., nActiveUe-1 as the active UE index in the cell. Denote prgIdx = 0, 1, ..., nPrbGrp-1 as the PRG index. Denote layerIdx = 0, 1, ..., nUeAnt-1 as the layer index. post EqSinr[uldx*nPrbGrp*nUeAnt + prgIdx*nUeAnt + layerIdx] is the uldx-th active UE's post-equalizer SINR on the prgIdx-th PRG and the layerIdx-th layer.
wbSinr	float[nActiveUe*nUeAnt]	Array of wideband per-layer post-equalizer SINRs of all active UEs in the cell. Value of each element: Denote uldx = 0, 1, ..., nActiveUe-1 as the active UE index in the cell. Denote layerIdx = 0, 1, ..., nUeAnt-1 as the layer index. wbSinr[uldx*nUeAnt + layerIdx] is the uldx-th active UE's wideband post-equalizer SINR on the layerIdx-th layer.

continues on next page

Table 36 – continued from previous page

Field	Type	Description
For FP32: estH_fr For FP16: estH_fr_half	For FP32: cuComplex[nSrsUe*nPrbGrp* nBsAnt*nUeAnt] For FP16: __nv_bfloat162[nSrsUe* nPrb- Grp*nBsAnt*nUeAnt]	Array of the refreshed subband (per-PRG) SRS channel estimates in the cell. Value of each element: Denote prgIdx = 0, 1, ..., nPrbGrp-1 as the PRG index. Denote uldx = 0, 1, ..., nSrsUe-1 as the index of the UE with refreshed SRS in the cell. Denote bsAntIdx = 0, 1, ..., nBsAnt-1 as the BS antenna port index Denote ueAntIdx = 0, 1, ..., nUeAnt as the UE antenna port index estH_fr[p prgIdx*nSrsUe*nBsAnt*nUeAnt + uldx*nBsAnt*nUeAnt + bsAntIdx*nUeAnt + ueAntIdx] is the complex channel coefficient for the uldx-th UE on the prgIdx-th PRG, the bsAntIdx-th BS antenna port and the ueAntIdx-th UE antenna port. (The above applies to the FP 16 version as well)
prdMat	For DL: cuComplex[nSrsUe*nPrbGrp* nBsAnt*nBsAnt] For UL: cuComplex[nSrsUe*nPrbGrp* nUeAnt*nUeAnt]	Array of the precoder/beamforming weights for the UEs that have refreshed SRS channel estimates in the cell. Value of each element: (for DL) Denote uldx = 0, 1, ..., nSrsUe-1 as the index of the UE with refreshed SRS in the cell. Denote prgIdx = 0, 1, ..., nPrbGrp-1 as the PRG index. Denote inPortIdx = 0, 1, ..., nBsAnt-1 as the precoder input port index Denote outPortIdx = 0, 1, ..., nBsAnt-1 as the precoder output port index prdMat[ uldx*nPrbGrp*nBsAnt*nBsAnt + prgIdx* nBsAnt*nBsAnt + inPortIdx*nBsAnt + outPortIdx] is the precoder/beamforming weight of the uldx-th UE on the prgIdx-th PRG and between the inPortIdx-th input port and the outPortIdx-th output port. (for UL, replace nBsAnt by nUeAnt in above description)

continues on next page

Table 36 – continued from previous page

Field	Type	Description
detMat	For DL: cuComplex[nSrsUe* nPrbGrp*nUeAnt*nUeAnt] For UL: cuComplex[nSrsUe* nPrbGrp*nBsAnt*nBsAnt]	Array of the detector/beamforming weights for the UEs that have refreshed SRS channel estimates in the cell. Value of each element: (for DL) Denote uldx = 0, 1, ..., nSrsUe-1 as the index of the UE with refreshed SRS in the cell. Denote prgIdx = 0, 1, ..., nPrbGrp-1 as the PRG index. Denote inPortIdx = 0, 1, ..., nUeAnt-1 as the detector input port index. Denote outPortIdx = 0, 1, ..., nUeAnt-1 as the detector output port index. detMat[ uldx*nPrbGrp*nUeAnt*nUeAnt + prgIdx*nUeAnt*nUeAnt + inPortIdx*nUeAnt + outPortIdx] is the detector weight of the uldx-th UE on the prgIdx-th PRG and between the inPortIdx-th input port and the outPortIdx-th output port. (for UL, replace nUeAnt by nBsAnt in above description)
sinVal	float[nSrsUe*nPrbGrp*nUeAnt]	Array of the per-UE, per-PRG, per-layer singular values obtained from the SVD of the refreshed SRS channel matrices. Value of each element: Denote uldx = 0, 1, ..., nSrsUe-1 as the index of the UE with refreshed SRS in the cell. Denote prgIdx = 0, 1, ..., nPrbGrp-1 as the PRG index. Denote layerIdx = 0, 1, ..., nUeAnt-1 as the layer index. sinVal[uldx*nPrbGrp*nUeAnt + prgIdx*nUeAnt + layerIdx] is the uldx-th UE's layerIdx-th largest singular value on the prgIdx-th PRG. For each UE and on each PRG, the singular values are stored in descending order.
avgRate-sActUe	float[nActiveUe]	Array of the long-term average data rates of all active UEs in the cell. Value of each element: Denote uldx = 0, 1, ..., nActiveUe-1 as the active UE index in the cell. avgRatesActUe[uldx] is the long-term average throughput of the uldx-th active UE in the cell.
prioWeightActUe	uint16_t[nActiveUe]	For priority-based UE selection. Priority weights of all active UEs in the cell. Value of each element: Denote uldx = 0, 1, ..., nActiveUe-1 as the active UE index in the cell. prioWeightActUe[uldx] is the uldx-th active UE's priority weight. 0xFFFF indicates an invalid element.

continues on next page

Table 36 – continued from previous page

Field	Type	Description
tbErrLastActUe	int8_t[nActiveUe]	<p>TB decoding error indicators of all active UEs in the cell. Value of each element:</p> <p>Denote <math>uldx = 0, 1, \dots, nActiveUe-1</math> as the active UE index in the cell.</p> <p>tbErrLastActUe[uldx] is the uldx-th active UE's TB decoding error indicator:</p> <ul style="list-style-type: none"> <li>-1 - the last transmission is not a new transmission (is a re-transmission)</li> <li>0 - decoded correctly</li> <li>1 - decoding error</li> </ul> <p>** Note that if the last transmission of a UE is not a new transmission, tbErrLastActUe of that UE should be set to -1.</p>
new-DataActUe	int8_t[nActiveUe]	<p>Indicators of initial transmission/retransmission for all active UEs in the cell. Value of each element:</p> <p>Denote <math>uldx = 0, 1, \dots, nActiveUe-1</math> as the active UE index in the cell.</p> <p>newDataActUe[uldx] is the indicator of initial transmission/retransmission for the uldx-th active UE in the cell.</p> <ul style="list-style-type: none"> <li>0 - retransmission</li> <li>1 - new data/initial transmission</li> <li>-1 indicates an invalid element</li> </ul>
allocSolLastTxActUe	<p>For type-0 PRG allocation: int16_t[nPrbGrp]</p> <p>For type-1 PRG allocation: int16_t[2*nActiveUe]</p>	<p>The PRG allocation solution for the last transmissions of all active UEs in the cell. Value of each element:</p> <p>For type-0 PRG allocation:</p> <p>Denote <math>prgIdx = 0, 1, \dots, nPrbGrp-1</math> as the PRG index. allocSolLastTxActUe[prgIdx] indicates the active UE index (0, 1, ..., nActiveUe-1) that the prgIdx-th PRG is allocated to.</p> <ul style="list-style-type: none"> <li>-1 indicates that a given PRG is not allocated to any UE.</li> </ul> <p>For type-1 PRG allocation:</p> <p>Denote <math>uldx = 0, 1, \dots, nActiveUe-1</math> as the active UE index in the cell.</p> <p>allocSolLastTxActUe[2*uldx] is the starting PRG index of the uldx-th active UE.</p> <p>allocSolLastTxActUe[2*uldx + 1] is the ending PRG index of the uldx-th active UE.</p> <ul style="list-style-type: none"> <li>-1 indicates that a given UE is not being allocated to any PRG.</li> </ul>

continues on next page

Table 36 – continued from previous page

Field	Type	Description
mcsSel-SolLastTx-ActUe	int16_t[nActiveUe]	MCS selection solution for the last transmissions of all active UEs in the cell. Value of each element: Denote uldx = 0, 1, ..., nActiveUe-1 as the active UE index in the cell. mcsSelSolLastTxActUe[uldx] indicates the MCS level for the uldx-th active UE in the cell. Range of each element: 0, 1, ..., 27 (Currently only support Table 5.1.3.1-2: MCS index table 2, 3GPP TS 38.214). -1 indicates an element is invalid.
layerSel-SolLastTx-ActUe	uint8_t[nActiveUe]	Layer selection solution for the last transmissions of all active UEs in the cell. Value of each element: Denote uldx = 0, 1, ..., nActiveUe-1 as the active UE index in the cell. layerSelSolLastTxActUe[uldx] indicates the number of layers selected for the uldx-th active UE in the cell. Range of each element: 0, 1, ..., nUeAnt-1 The selected layers have singular values descending from the largest one.

The following are required data buffers for each cuMAC task type, which is defined by taskBitMask.

Task Type	cuMAC name	module	Required data for L2 to pass to cuMAC-CP
0x01	multiCellUeSelection		prgMsk, wbSinr, avgRatesActUe.
0x02	multiCellScheduler		All TaskType=0x01 buffers and postEqSinr, sinVal, estH_fr, detMat, prdMat.
0x04	multiCellLayerSel		All TaskType=0x02 buffers.
0x08	mcsSelectionLUT		All TaskType=0x04 buffers and tbErrLastActUe.

### 1.6.1.2.2.2 TTI\_END

Name	Type	Description
end_param	uint8_t	Reserved, not used yet.



### 1.6.1.2.2.3 SCH\_TTI.response

This message is used to return scheduling results to L2. The results are populated in NVIPC DATA buffer,

Name	Type	Description
offsets	struct	Buffer offset for each data: 0xFFFFFFFF: buffer not used (Invalid). Other: buffer offset based on nvipc data_buf

The data buffers are populated in NVIPC DATA buffer. The “offsets” structure defines the buffer offsets of all the buffers.

Note: not all buffers are used. The offset of not used buffers should be set to 0xFFFFFFFF.

Offset Name	Offset Type	Buffer description
setSchdUePerCellTTI	uint32_t	Set of IDs of the selected UEs for the cell
allocSol	uint32_t	PRB group allocation solution for all active UEs in the cell
layerSelSol	uint32_t	Layer selection solution for all active UEs in the cell
mcsSelSol	uint32_t	MCS selection solution for all active UEs in the cell

The data buffers details are as below.

Field	Type	Description
setSchdUePer-CellTTI	uint16_t [nMaxSchUePerCell]	Set of IDs of the selected UEs for the cell. Value of each element: Denote $i = 0, 1, \dots, nMaxSchUePerCell-1$ as the $i$ -th selected UE for the cell. setSchdUePerCellTTI[i] is within $\{0, 1, \dots, nActiveUe-1\}$ and represents the active UE index of the $i$ -th selected UE.

continues on next page

Table 38 – continued from previous page

Field	Type	Description
allocSol	For type-0 PRG allocation: int16_t[nPrbGrp] For type-1 PRG allocation: int16_t[2*nActiveUe]	PRB group allocation solution for all active UEs in the cell. Value of each element: For type-0 PRG allocation: Denote prgIdx = 0, 1, ..., nPrbGrp-1 as the PRG index. allocSol[prgIdx] indicates the active UE index (0, 1, ..., nActiveUe-1) that the prgIdx-th PRG is allocated to. -1 indicates that a given PRG is not allocated to any UE. For type-1 PRG allocation: Denote uldx = 0, 1, ..., nActiveUe-1 as the active UE index in the cell. allocSol[2*uldx] is the starting PRG index of the uldx-th active UE. allocSol[2*uldx + 1] is the ending PRG index of the uldx-th active UE. -1 indicates that a given UE is not being allocated to any PRG.
mcsSelSol	int16_t[nActiveUe]	MCS selection solution for all active UEs in the cell. Value of each element: Denote uldx = 0, 1, ..., nActiveUe-1 as the active UE index in the cell. mcsSelSol[uldx] indicates the MCS level for the uldx-th active UE in the cell. Range of each element: 0, 1, ..., 27 (Currently only support Table 5.1.3.1-2: MCS index table 2, 3GPP TS 38.214). -1 indicates an element is invalid.
layerSelSol	uint8_t[nActiveUe]	Layer selection solution for all active UEs in the cell. Value of each element: Denote uldx = 0, 1, ..., nActiveUe-1 as the active UE index in the cell. layerSelSol[uldx] indicates the number of layers selected for the uldx-th active UE in the cell. Range of each element: 0, 1, ..., nUeAnt-1 The selected layers have singular values descending from the largest one.

#### 1.6.1.2.2.4 TTI\_ERR.indication

Name	Type	Description
msg_id	in t32_t	SFN number
error_code	in t32_t	0: no error. Other: TBD
reason_code	in t32_t	0: no error. Other: TBD

### 1.6.1.3 L2 integration notes

#### 1.6.1.3.1 NVIPC integration

Refer to <aerial\_sdk>/cuPHY-CP/gt\_common\_libs/README.md for L2 to copy NVIPC source and build libnvipc.so for L2.

Recommend L2 to create nvipc\_config.yaml and copy the “transport” configure from <aerial\_sdk>/cuMAC-CP/config/cumac\_cp.yaml. NVIPC configuration should be same between L2 and cuMAC-CP.

Refer to Aerial NVIPC section for NVIPC technical details.

#### 1.6.1.3.2 cuMAC message definitions

Public header file for cuMAC message definitions is <aerial\_sdk>/cuMAC-CP/lib/cumac\_msg.h. Copy it to L2 and include it in L2 source.

**Note for 24-2 release:** Supports 4 cuMAC modules in functionality. But only the first module can be finished in 1 slot window yet.

- ▶ multiCellUeSelection
- ▶ multiCellScheduler
- ▶ multiCellLayerSel
- ▶ mcsSelectionLUT

### 1.6.1.4 Tests

#### 1.6.1.4.1 Basic cuMAC-CP Standalone Test: TestMAC + cuMAC-CP

##### 1.6.1.4.1.1 Configurations

In cumac\_cp.yaml, configure cell\_num and free CPU cores (low\_priority\_core, cumac\_cp\_rcv thread core, woker\_cores) for cuMAC-CP. Example:

```
# CPU core shared by all low-priority threads
low_priority_core: 19
```

(continues on next page)

(continued from previous page)

```
recv_thread_config:
  name: cumac_cp_recv
  cpu_affinity: 25
  sched_priority: 95
# cuMAC task worker cores
worker_cores: [31, 32, 33, 34, 35, 36, 37, 38]
# Total cell number
cell_num: 8
```

In `test_cumac_config.yaml`, configure `cumac_cell_num`, `cumac_cp_standalone=1` and free CPU cores for testMAC (`low_priority_core`, `cumac_recv/cumac_sched/cumac_builder` threads cores).

```
recv_thread_config:
  name: cumac_recv
  cpu_affinity: 43
  sched_priority: 95
sched_thread_config:
  name: cumac_sched
  cpu_affinity: 39
  sched_priority: 96
builder_thread_config:
  name: cumac_builder
  cpu_affinity: 39
  sched_priority: 95
# Run test_mac + cumac_cp only without depending on L1
cumac_cp_standalone: 1
```

In `test_mac_config.yaml`, configure `test_cumac_config_file: test_cumac_config.yaml` to enable cuMAC-CP test.

```
# Set to yaml file like test_cumac_config.yaml to enable cuMAC-CP test
test_cumac_config_file: test_cumac_config.yaml
```

### 1.6.1.4.1.2 Run Steps

1. Generate per cell TVs for cuMAC-CP.

```
mkdir $cuBB_SDK/testVectors/cumac

cd $cuBB_SDK/testVectors/cumac

sudo
$cuBB_SDK/build/cuMAC/examples/multiCellSchedulerUeSelection/
↪multiCellSchedulerUeSelection -t 3
```

2. Run cumac\_cp first and then run test\_mac.

```
# 1. Run cumac_cp

sudo $cuBB_SDK/build/cuMAC-CP/cumac_cp

# 2. Run test_mac

sudo $cuBB_SDK/build/cuPHY-CP/testMAC/testMAC/test_mac F08 8C_60c
```

### 1.6.1.4.1.3 Test Results

Will see throughput in cumac\_cp and test\_mac console outputs.

cumac\_cp console output example:

```
16:11:36.999875 WRN 27518 0 25 [CUMCP.HANDLER] Cell 0 | CUMAC 2000 | ERR 0 | Slots
↪4000
16:11:36.999876 WRN 27518 0 25 [CUMCP.HANDLER] Cell 1 | CUMAC 2000 | ERR 0 | Slots
↪4000
16:11:36.999877 WRN 27518 0 25 [CUMCP.HANDLER] Cell 2 | CUMAC 2000 | ERR 0 | Slots
↪4000
16:11:36.999877 WRN 27518 0 25 [CUMCP.HANDLER] Cell 3 | CUMAC 2000 | ERR 0 | Slots
↪4000
16:11:36.999877 WRN 27518 0 25 [CUMCP.HANDLER] Cell 4 | CUMAC 2000 | ERR 0 | Slots
↪4000
16:11:36.999877 WRN 27518 0 25 [CUMCP.HANDLER] Cell 5 | CUMAC 2000 | ERR 0 | Slots
↪4000
16:11:36.999877 WRN 27518 0 25 [CUMCP.HANDLER] Cell 6 | CUMAC 2000 | ERR 0 | Slots
↪4000
16:11:36.999877 WRN 27518 0 25 [CUMCP.HANDLER] Cell 7 | CUMAC 2000 | ERR 0 | Slots
↪4000
```

test\_mac console output example:

```
16:11:37.000364 WRN 27533 0 39 [CUMAC.HANDLER] Cell 0 | CUMAC 2000 | ERR 0 | INV 0 |
↪Slots 4000
16:11:37.000364 WRN 27533 0 39 [CUMAC.HANDLER] Cell 1 | CUMAC 2000 | ERR 0 | INV 0 |
↪Slots 4000
16:11:37.000364 WRN 27533 0 39 [CUMAC.HANDLER] Cell 2 | CUMAC 2000 | ERR 0 | INV 0 |
↪Slots 4000
16:11:37.000364 WRN 27533 0 39 [CUMAC.HANDLER] Cell 3 | CUMAC 2000 | ERR 0 | INV 0 |
↪Slots 4000
16:11:37.000365 WRN 27533 0 39 [CUMAC.HANDLER] Cell 4 | CUMAC 2000 | ERR 0 | INV 0 |
```

(continues on next page)

(continued from previous page)

```

↪Slots 4000
16:11:37.000365 WRN 27533 0 39 [CUMAC.HANDLER] Cell 5 | CUMAC 2000 | ERR 0 | INV 0 |
↪Slots 4000
16:11:37.000365 WRN 27533 0 39 [CUMAC.HANDLER] Cell 6 | CUMAC 2000 | ERR 0 | INV 0 |
↪Slots 4000
16:11:37.000365 WRN 27533 0 39 [CUMAC.HANDLER] Cell 7 | CUMAC 2000 | ERR 0 | INV 0 |
↪Slots 4000

```

### 1.6.1.4.2 Other Configurations for Test and Debug

#### 1.6.1.4.2.1 Select cuMAC Modules in TestMAC

In `test_cumac_config.yaml`, configure `task_bitmask` to select cuMAC modules to be enabled. Example:

- ▶ 0x1 – only the first module `multiCellUeSelection`.
- ▶ 0xF – all the 4 modules

```

# CUMAC task bitmask: b0 - multiCellUeSelection; b1 - multiCellScheduler; b2 -
↪multiCellLayerSel; b3 - mcsSelectionLUT
task_bitmask: 0xF

```

When enabled INFO level log, will be able to see task bitmask and task processing timing log in `/tmp/cumac_cp.log` (`grep "finish" cumac_cp.yaml`)

#### 1.6.1.4.2.2 Enable Buffer Dumping in cuMAC-CP

Configure `debug_option` can print data buffers but will impact timing and performance, so only use it during debug.

## 1.6.2. NVIPC

NVIPC is a messaging system for communication between two processes on the same system. It solves two problems:

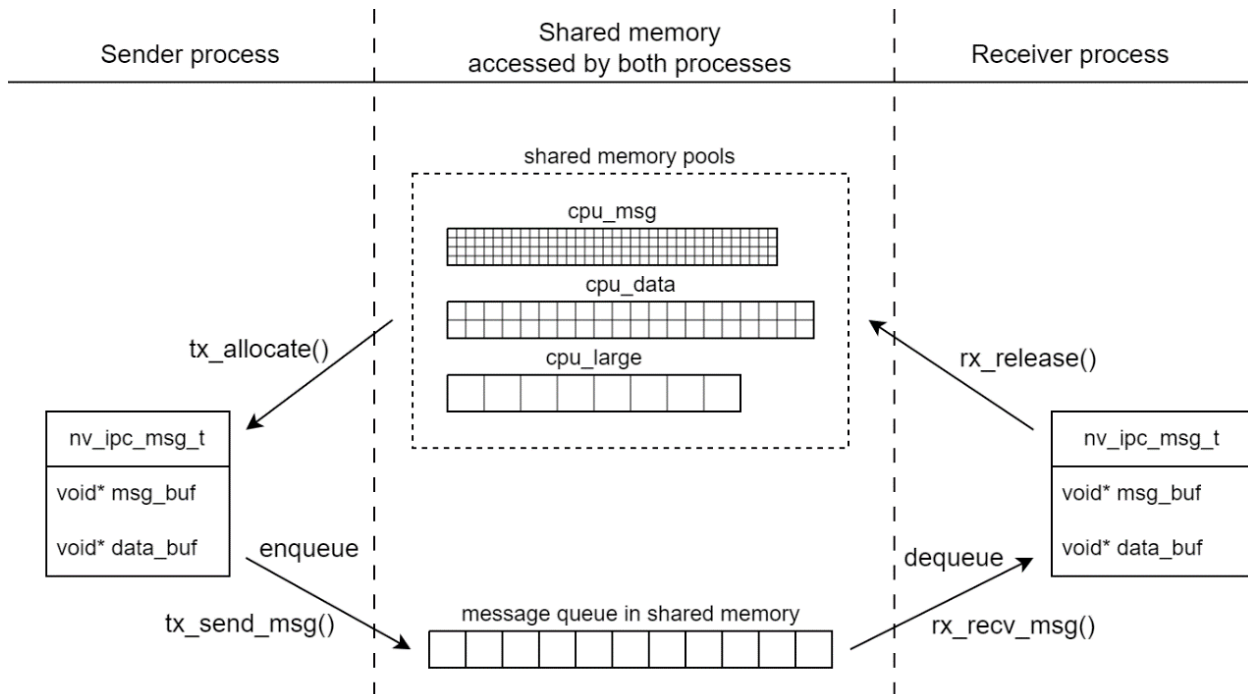
- ▶ How to transfer messages between two processes
- ▶ How to notify a receiver when sending is finished

### 1.6.2.1 NVIPC Overview

This section provides an overview of NVIPC messaging functionality.

### 1.6.2.1.1 NVIPC Message Transfer

To achieve low-latency and high performance, NVIPC uses lock-free memory pools and lock-free queues to deliver the messages. The message transfer module architecture is as below.



### 1.6.2.1.2 NVIPC API Definitions

An NVIPC message is divided into two parts:

- **MSG:** Handled in control logic which runs in CPU thread.
- **DATA:** Handled with high performance computing which runs in CPU thread or GPU context.

A struct `nv_ipc_msg_t` is defined to represent a generic NVIPC message, as shown below.

```
typedef struct
{
    int32_t msg_id;    // IPC message ID
    int32_t cell_id;  // Cell ID
    int32_t msg_len;  // MSG part length
    int32_t data_len; // DATA part length
    int32_t data_pool; // DATA memory pool ID
    void* msg_buf;    // MSG buffer pointer
    void* data_buf;   // MSG buffer pointer
}
```

(continues on next page)

(continued from previous page)

```
} nv_ipc_msg_t;
```

The MSG part and DATA part are stored in different buffers. MSG part presence is mandatory, DATA part presence is optional. `data_buf` is null when there is no DATA part present.

NVIPC creates multiple memory pools at initial to manage the MSG part and DATA part buffers. An enum type `nv_ipc_mempool_id_t` is defined as the memory pool indicator. MSG buffer is allocated from CPU shared memory pool. DATA buffer can be allocated from CPU shared memory pool or CUDA shared memory pool.

```
typedef enum
{
    NV_IPC_MEMPOOL_CPU_MSG    = 0, // CPU SHM pool for MSG part
    NV_IPC_MEMPOOL_CPU_DATA  = 1, // CPU SHM pool for DATA part
    NV_IPC_MEMPOOL_CPU_LARGE = 2, // CPU SHM pool for large DATA part
    NV_IPC_MEMPOOL_CUDA_DATA = 3, // CUDA SHM pool for DATA part
    NV_IPC_MEMPOOL_GPU_DATA  = 4, // CUDA SHM pool which supports GDR copy
    NV_IPC_MEMPOOL_NUM       = 5
} nv_ipc_mempool_id_t;
```

And a serials of APIs are defined in struct

```
struct nv_ipc_t
{
    // De-initiate and destroy the nv_ipc_t instance
    int (*ipc_destroy)(nv_ipc_t* ipc);

    // Memory allocate/release for TX side
    int (*tx_allocate)(nv_ipc_t* ipc, nv_ipc_msg_t* msg, uint32_t options);
    int (*tx_release)(nv_ipc_t* ipc, nv_ipc_msg_t* msg);

    // Memory allocate/release for RX side
    int (*rx_allocate)(nv_ipc_t* ipc, nv_ipc_msg_t* msg, uint32_t options);
    int (*rx_release)(nv_ipc_t* ipc, nv_ipc_msg_t* msg);

    // Send a ipc_msg_t message. Return -1 if failed
    int (*tx_send_msg)(nv_ipc_t* ipc, nv_ipc_msg_t* msg);

    // Call tx_tti_sem_post() at the end of a TTI
```

(continues on next page)



(continued from previous page)

```

int (\*tx_tti_sem_post)(nv_ipc_t\* ipc);
// Call rx_tti_sem_wait() and then receive all messages in a TTI
int (\*rx_tti_sem_wait)(nv_ipc_t\* ipc);
// Get a ipc_msg_t message. Return -1 if no available.
int (\*rx_recv_msg)(nv_ipc_t\* ipc, nv_ipc_msg_t\* msg);
// Get SHM event FD or UDP socket FD for epoll
int (\*get_fd)(nv_ipc_t\* ipc);
// Write tx_fd to notify the event, Only need for SHM
int (\*notify)(nv_ipc_t\* ipc, int value);
// Read rx_fd to clear the event. Only need for SHM
int (\*get_value)(nv_ipc_t\* ipc);
// More not used
};

```

### 1.6.2.1.3 Lock-Free Data Structures

A lock-free queue named “array queue” is implemented in NVIPC. The array queue has the following features:

- ▶ FIFO (first in first out).
- ▶ Lock-free: supports multiple producers and multiple consumers without lock.
- ▶ Finite size: max length is defined at initial: N.
- ▶ Valid values are integers: 0, 1, ..., N-1, can be used as the node index/pointer.
- ▶ Doesn't support duplicate values.

Based on the lock-free array queue, generic memory pools and ring queues are implemented, and they are also lock-free:

- ▶ Memory pool: array queue + memory buffer array
- ▶ FIFO ring queue: array queue + element node array

### 1.6.2.1.4 NVIPC Memory Pools

Several shared memory pools are implemented in NVIPC. They are accessible by both the primary process and the secondary process. Each memory pool is an array of fixed size buffers. Buffer size and pool length (buffer count) are configurable by yaml file. If the buffer size or pool length is configured to 0, that memory pool will not be created. Below is the default NVIPC memory pools configuration which is used in cuPHY-CP.

Memory Pool ID	SHM file name at /dev/shm/	Comment
NV_IPC_MEMPOOL_CPU_MSG	<prefix>_cpu_msg	CPU memory for transfer MSG part.
NV_IPC_MEMPOOL_CPU_DATA	<prefix>_cpu_data	CPU memory for transfer DATA part.
NV_IPC_MEMPOOL_CPU_LARGE	<prefix>_cpu_large	CPU memory for transfer large DATA part.
NV_IPC_MEMPOOL_CUDA_DATA	<prefix>_cuda_data	GPU memory. Not used.
NV_IPC_MEMPOOL_GPU_DATA	<prefix>_gpu_data	GPU memory with GDR copy. Not used.

After NVIPC primary app initialized, the SHM files will present in /dev/shm/ folder.

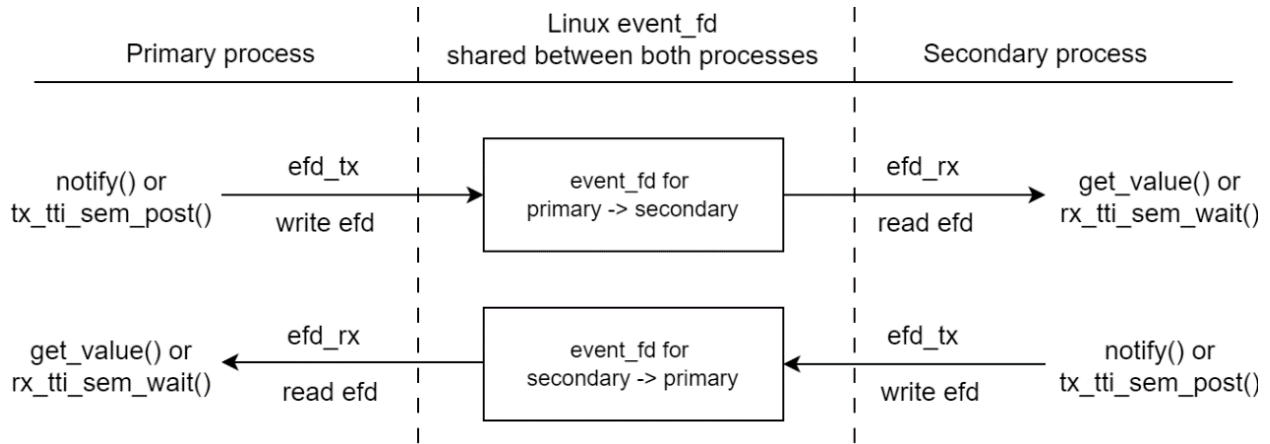
#### 1.6.2.1.4.1 Bi-Directional Message Queues

Two ring queues will be created to deliver the message buffer indices. The TX ring in sender app and RX ring in receiver app are the same ring in shared memory of the system. Both DL and UL ring queues are stored in the same shared memory file.

SHM file name at /dev/shm/	Internal code name	IPC direction	PHY /PRIMARY	MAC /SECONDARY
<prefix>_shm	<prefix>_ring_m2s	Uplink	TX	RX
<prefix>_shm	<prefix>_ring_s2m	Downlink	RX	TX

#### 1.6.2.1.5 NVIPC message notification

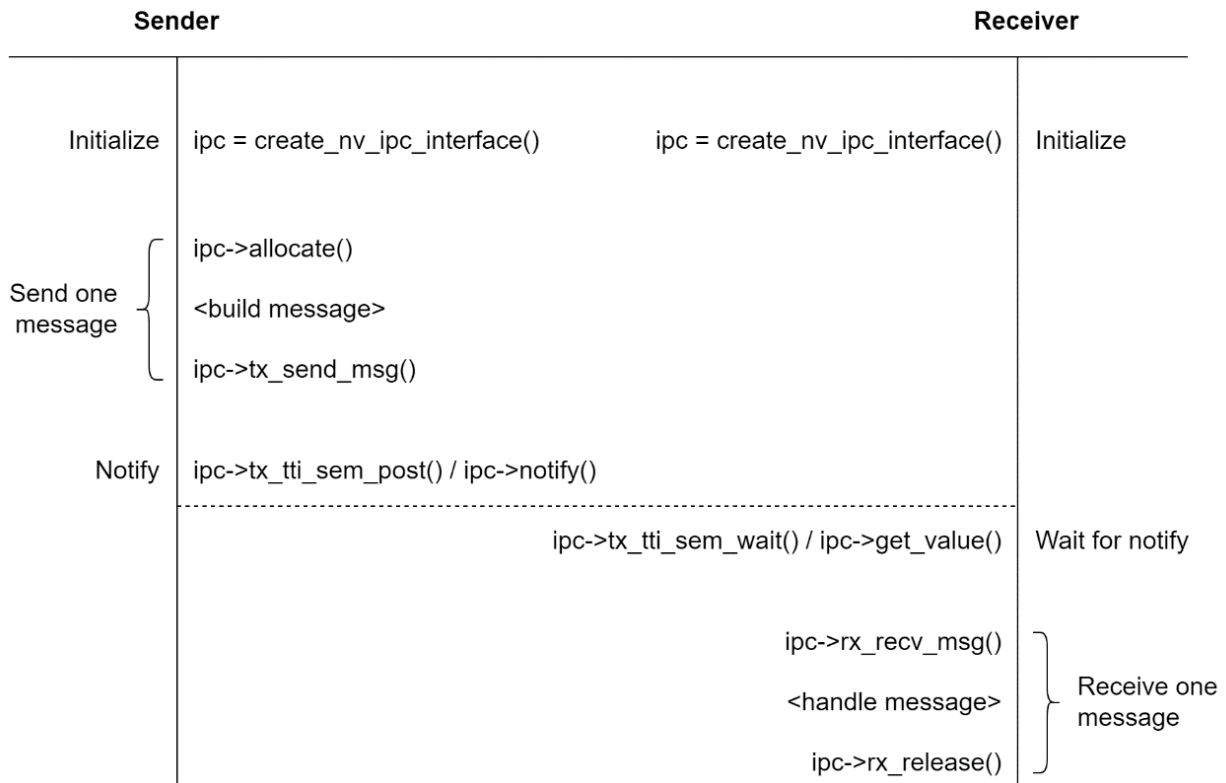
NVIPC uses Linux event\_fd to make notifications. It supports multiple I/O with select /poll/epoll mechanism. The message notification module architecture is as below.



Each process creates an `event_fd` file descriptor `efd_rx` for incoming message notification and share it with peer process. The local `efd_rx` for receiving is shared as `efd_tx` for sending in peer side. Receiver process can call `get_fd()` to get the I/O descriptor and use poll/epoll to get the notification. Besides, the `event_fd` is initiated with `EFD_SEMAPHORE` flag so it can work like a semaphore. NVIPC provides both event/select style and semaphore style notification APIs.

### 1.6.2.1.6 NVIPC message flow

A typical message transfer flow is shown below.



Since the memory pools and ring queues support lock-less concurrence, the use of notification APIs is not mandatory. If user don't want to use notification, the receiver should poll the incoming message

queue by keep dequeuing from the lock-free queue. `rx_recv_msg()` function returns -1 when the queue is empty.

### 1.6.2.2 NVIPC Integration

#### 1.6.2.2.1 Configuration

NVIPC supports YAML file to store the configurations. Recommend L2 partner to use YAML file. Below is the example configuration used in Aerial L1. Both primary and secondary processes should have the same `ring_len` and `mempool_size` configurations.

```
transport:
  type: shm
  shm_config:
    primary: 1
    prefix: nvipc # Note: prefix string length should < 32
    cuda_device_id: 0
    ring_len: 8192
    mempool_size:
      cpu_msg:
        buf_size: 8192
        pool_len: 4096
      cpu_data:
        buf_size: 576000
        pool_len: 1024
      cpu_large:
        buf_size: 4096000
        pool_len: 64
      cuda_data:
        buf_size: 307200
        pool_len: 0
      gpu_data:
        buf_size: 576000
        pool_len: 0
```

(continues on next page)

(continued from previous page)

```

app_config:
  grpc_forward: 0
  debug_timing: 0
  pcap_enable: 0
  pcap_cpu_core: 17 # CPU core of background pcap log save thread
  pcap_cache_size_bits: 29 # 2^29 = 512MB, size of /dev/shm/${prefix}_pcap
  pcap_file_size_bits: 31 # 2^31 = 2GB, max size of /var/log/aerial/${prefix}_pcap.
  ↳ Requires pcap_file_size_bits > pcap_cache_size_bits.
  pcap_max_data_size: 8000 # Max DL/UL FAPI data size to capture reduce pcap size.

```

### 1.6.2.2.2 Initiation

Here is the reference code for initialization. The NVIPC primary process is responsible to create and initiate SHM pools, ring queues. The NVIPC secondary process looks up the created pools and queues. In Aerial L1 is the primary process, L2 should be configured as the secondary process.

```

// Create configuration
nv_ipc_config_t config;

// Select module_type for primary or secondary process
nv_ipc_module_t module_type = NV_IPC_MODULE_PRIMARY/SECONDARY;

// Recommended initialization: load nvipc configurations from yaml file
load_nv_ipc_yaml_config(&config, yaml_path, module_type);

// Optional: set default configs and overwrite what need change
config.ipc_transport = *NV_IPC_TRANSPORT_SHM*;

**if**\ (set_nv_ipc_default_config(&config, module_type) < 0) {
NVLOGE(TAG, "%s: set configuration failed\n", \_func\_);
**return** -1;
}

// Override the default configurations
config.transport_config.shm.cuda_device_id = test_cuda_device_id;

// Create IPC interface: nv_ipc_t ipc

```

(continues on next page)

(continued from previous page)

```

nv_ipc_t\* ipc;

**if**\ ((ipc = create_nv_ipc_interface(&config)) == *NULL*) {
NVLOGE(TAG, "%s: create IPC interface failed\\n", \_\_func\_\_);
**return** -1;
}

```

After successfully created IPC interface, some shared memory files can be seen in /dev/shm/ folder. For example, if <prefix>="nvipc":

```

ls -al /dev/shm/nvipc\*

nvipc_shm
nvipc_cpu_msg
nvipc_cpu_data
nvipc_cpu_large

```

### 1.6.2.2.3 De-Initialization

Below example code is for de-initialization.

```

**if**\ (ipc->ipc_destroy(ipc) < 0) {
NVLOGE(TAG, "%s close IPC interface failed\\n", \_\_func\_\_);
}

```

### 1.6.2.2.4 Sending

The procedure for sending is as follows:

```
allocate buffers -> fill content -> send.
```

When fill content, for CUDA memory, the data\_buf is a CUDA memory pointer which can't be accessed directly in CPU memory space. The NVIPC APIs provide basic memcpy functions to copy between CPU memory and CUDA memory. For more CUDA operation, user can directly access the GPU memory buffer with CUDA APIs.

```

nv_ipc_msg_t send_msg,

send_msg.msg_id = fapi_msg_id; // Optional: FAPI message ID

send_msg.msg_len = fapi_msg_len; // Max length is the MSG buffer size,
configurable

send_msg.data_len = fapi_data_len; // Max length is the MSG buffer size,

```

(continues on next page)

(continued from previous page)

```

configurable

send_msg.data_pool = *NV_IPC_MEMPOOL_CPU_DATA*; // Options: CPU_MSG,
CPU_DATA, CUDA_DATA

// Allocate buffer for TX message

**if**\ (ipc->tx_allocate(ipc, &send_msg, 0) != 0)

{
NVLOGE(TAG, "%s error: allocate buffer failed\\n", \_\_func\_\_);
**return** -1;
}

// Fill the MSG content

int8_t fapi_msg[SHM_MSG_BUF_SIZE];

memcpy(send_msg.msg_buf, fapi_msg, fapi_len);

// Fill the DATA content if data exist.

int8_t fapi_data[SHM_MSG_DATA_SIZE];

**if** (send_msg.data_pool == *NV_IPC_MEMPOOL_CPU_DATA*) { // CPU_DATA
case

memcpy(send_msg.data_buf, fapi_data, send_msg.data_len);

} **else** **if** (send_msg.data_pool == *NV_IPC_MEMPOOL_CUDA_DATA*) {
// CUDA_DATA case

**if**\ (ipc->cuda_memcpy_to_device(ipc, send_msg.data_buf, fapi_data,
send_msg.data_len) < 0){

NVLOGE(TAG, "%s CUDA copy failed\\n", \_\_func\_\_);

}

} **else** { // NO_DATA case

// NO data, do nothing

}

// Send the message

**if**\ (ipc->tx_send_msg(ipc, &send_msg) < 0){

NVLOGE(TAG, "%s error: send message failed\\n", \_\_func\_\_);

// May need future retry or release the send_msg buffers

```

(continues on next page)

(continued from previous page)

```
// If fail, check configuration: ring queue length > memory pool length
}
```

### 1.6.2.2.5 Receive

The procedure for sending is as follows:

```
receive -> handle message -> release buffers
```

```
nv_ipc_msg_t recv_ms\ *g*
**if**\ (ipc->rx_recv_msg(ipc, &recv_msg) < 0)
{
LOGV(TAG, "%s: no more message available\\n", \_func\_);
**return** -1;
}
// Example: Handle MSG part
int8_t fapi_msg[SHM_MSG_BUF_SIZE];
memcpy(fapi_msg, recv_msg.msg_buf, recv_msg.msg_len);
// Example: Handle DATA part
int8_t fapi_data[SHM_MSG_BUF_SIZE];
**if** (recv_msg.data_pool == *NV_IPC_MEMPOOL_CPU_DATA*) { // CPU_DATA
case
memcpy(fapi_data, recv_msg.data_buf, &recv_msg.data_len);
} **else** **if** (recv_msg.data_pool == *NV_IPC_MEMPOOL_CUDA_DATA*) {
// CUDA_DATA case
**if**\ (ipc->cuda_memcpy_to_host(ipc, fapi_data, recv_msg.data_buf,
recv_msg.data_len) < 0){
LOGE(TAG, "%s CUDA copy failed\\n", \_func\_);
}
} **else** { // NO_DATA case
// NO data, do nothing
}
**if**\ (ipc->rx_release(ipc, &recv_msg) < 0){
```

(continues on next page)



(continued from previous page)

```
LOGW(TAG, "%s: release error\\n", \_\_func\_\_);
}
```

### 1.6.2.2.6 Notification

Two types of notification APIs are provided: semaphore style and event\_fd style. Each NVIPC process can choose any type no matter what the peer process chooses, but keep using one type in one process.

In low level of the SHM IPC library event\_fd is implemented. The semaphore API interface is a wrapper of the event\_fd implementation.

The APIs are ready to use after IPC interface successfully created by create\_nv\_ipc\_interface().

For semaphore types, it's easy to use:

- Receiver:

```
ipc->tx_tti_sem_wait(ipc);
```

- Sender:

```
ipc->tx_tti_sem_post(ipc);
```

For event\_fd style, user should get the fd and use epoll functions to listen to I/O events.

- Receiver:

```
**struct** epoll_event ev, events[MAX_EVENTS];
**int** epoll_fd = epoll_create1(0);
**if**\ (epoll_fd == -1)
{
NVLOGE(TAG, "%s epoll_create failed\\n", \_\_func\_\_);
}
**int** ipc_rx_event_fd = ipc->get_fd(ipc); // IPC notification API:
get_fd()
ev.\ *events* = *EPOLLIN*;
ev.\ *data*.\ *fd* = ipc_rx_event_fd;
**if**\ (epoll_ctl(epoll_fd, *EPOLL_CTL_ADD*, ev.\ *data*.\ *fd*, &ev)
== -1)
{
NVLOGE(TAG, "%s epoll_ctl failed\\n", \_\_func\_\_);
}
}
```

(continues on next page)

(continued from previous page)

```

**while**\ (1)
{
**int** nfds = epoll_wait(epoll_fd, events, MAX_EVENTS, -1);
**if**\ (nfds == -1)
{
NVLOGE(TAG, "epoll_wait notified: *nfds*\ =%d\\n", nfds);
}
**for**\ (**int** n = 0; n < nfds; ++n)
{
**if**\ (events[n].\ *data*.\ *fd* == ipc_rx_event_fd)
{
ipc->get_value(ipc); // IPC notification API: get_value()
// Receive incoming message here
}
}
}
close(epoll_fd);

```

► Sender:

```
ipc->notify(ipc, 1); // IPC notification API: notify()
```

## 1.7. cuBB Developer Guide

### 1.7.1. cuBB Software Architecture Overview

The cuPHY library software stack is shown in the figure below. It consists of L2 adapter, cuPHY driver, cuPHY CUDA kernels that process PHY channels and cuPHY controller.

The interface between the L2 and L1 goes through nvipc interface, which is provided as a separate library. L2 and L1 communicate using FAPI protocol [6]. L2 adapter takes in slot commands from the L2 and translates them into L1 tasks, which are then consumed by cuPHY driver. Similarly, L1 task results are sent from cuPHY driver to L2 adapter, which are then communicated to L2.

The user transport block (TB) data in both DL and UL directions go through the same nvipc interface. The data exchange directly happens between cuPHY and L2 with the control of cuPHY driver.

cuPHY driver controls execution of cuPHY L1 kernels and manages the movement of data in and out of these kernels. The interface between the cuPHY L1 kernels and the NIC is also managed by the cuPHY driver by using the FH driver, that is provided as a library.

cuPHY controller is the main application that initializes the cell configurations, FH buffers and configures all threads that are used by L1 control tasks.

The functionality of each of these components is explained in more detail in the [Components](#) section.

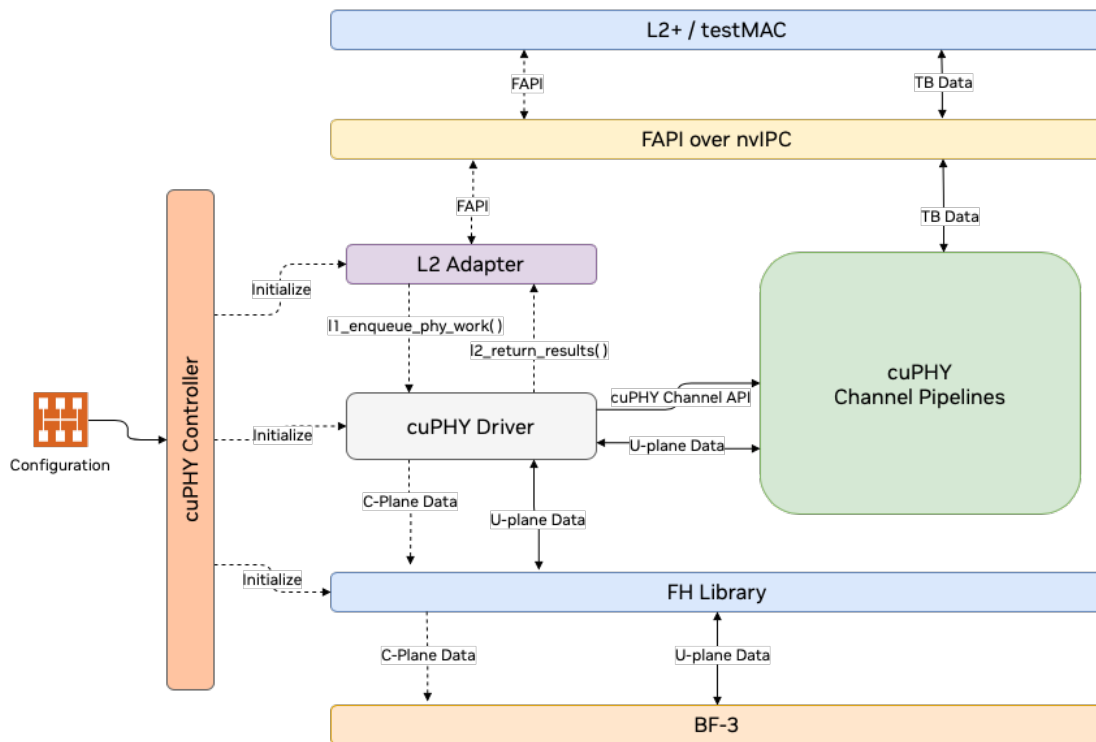


Fig. 1: cuPHY Software Stack

## 1.7.2. Aerial cuPHY Components

### 1.7.2.1 L2 Adapter

The L2 Adapter is the interface between the L1 and the L2, which translates SCF FAPI commands to slot commands. The slot commands are received by cuPHY driver to initiate cuPHY tasks. It makes use of nvipc library to transport messages and data between L1 and L2. It is also responsible for sending slot indications to drive the timing of the L1-L2 interface. L2 Adapter keeps track of the slot timing and it can drop messages received from L2 if they are received late.

### 1.7.2.2 cuPHY Driver

The cuPHY driver is responsible for orchestrating the work on the GPU and the FH by using cuPHY and FH libraries. It processes L2 slot commands generated by L2 adapter to launch tasks and communicates cuPHY outputs (e.g. CRC indication, UCI indication, measurement reports, etc.) back to L2. It uses L2 adapter FAPI message handler library to communicate with L2.

cuPHY driver configures and initiates DL and UL cuPHY tasks, which in turn launch CUDA kernels on the GPU. These processes are managed at the slot level. The cuPHY driver also controls CUDA kernels responsible for transmission and reception of user plane (U-plane) packets to and from the NIC interface. The CUDA kernels launched by the driver take care of re-ordering and decompression of UL packets and compression of DL packets. The DL packets are transmitted by GPU initiated communications after the compression.

cuPHY driver interacts with the FH interface using ORAN compliant FH library to coordinate transmission of FH control plane (C-plane) packets. The transmission of C-plane packets is done via DPDK library calls (CPU initiated communication). The U-plane packets are communicated through transmit and receive queues created by the cuphycontroller.

### 1.7.2.3 FH Driver Library

The FH library ensures timely transmission and reception of FH packets between the O-DU and O-RU. It uses accurate send scheduling functions of the NIC to comply with the timing requirements of the O-RAN FH specification.

The FH driver maintains the context and connection per eAxCid. It is responsible of encoding and decoding of FH commands for U-plane and C-plane messages.

The FAPI commands received from the L2 trigger processing of DL or UL slots. C-plane messages are for both DL and UL generated on the CPU and communicated to the O-RU through the NIC interface with DPDK. The payload of DL U-plane packets are prepared on the GPU and sent to the NIC interface from the memory pool on the GPU with the DOCA GPU NetIO library. The flow of DL C-plane and U-plane packets is illustrated in the below figure.

As shown in the above figure, UL U-plane packets received from the O-RU are directly copied to GPU memory from the NIC interface with the DOCA GPU NetIO library. The UL data is decompressed and processed by GPU kernels. After the UL kernels are completed, the decoded UL data transport blocks are sent to the L2.

### 1.7.2.4 cuPHY Controller

The cuPHY controller is the main application that initializes the system with the desired configuration. During the start-up process, cuPHY controller creates a new context (memory resources, tasks) for each new connection with a O-RU, identified by MAC address, VLAN ID and set of eAxCids. It starts cuphydriver DL/UL worker threads and assigns them to CPU cores as configured in the yaml file. It also prepares GPU resources and initiates FH driver and NIC class objects.

cuPHY controller prepares L1 according to the desired gNB configuration. It can also bring a carrier in and out of service with the cell lifecycle management functionality.

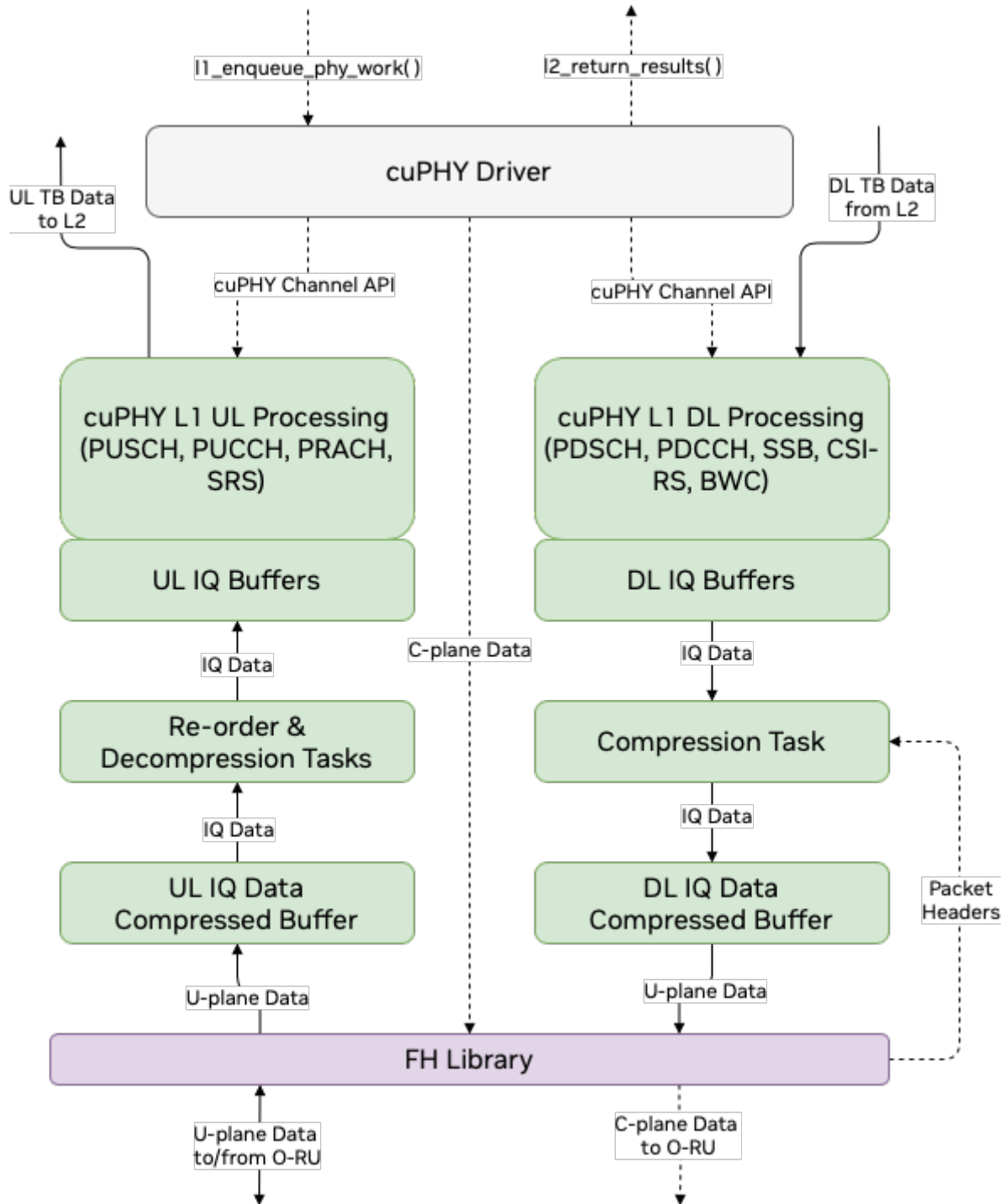


Fig. 2: User and Control Plane Data Flow through cuPHY driver and cuPHY tasks

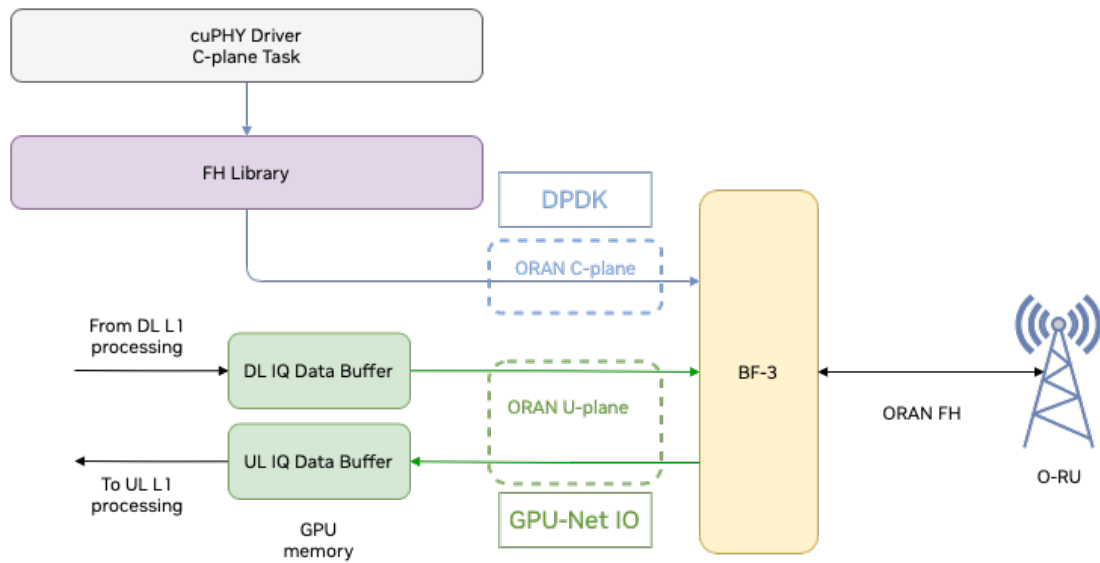


Fig. 3: Flow of packets on the FH

### 1.7.2.5 cuPHY

cuPHY is a CUDA implementation of 5G NR PHY layer signal processing functions. The cuPHY library supports all 5G NR PHY channels in compliance with 3GPP Release 15 specification. As shown in the below figure, cuPHY library corresponds to upper PHY stack according to O-RAN 7.2x split option [8].

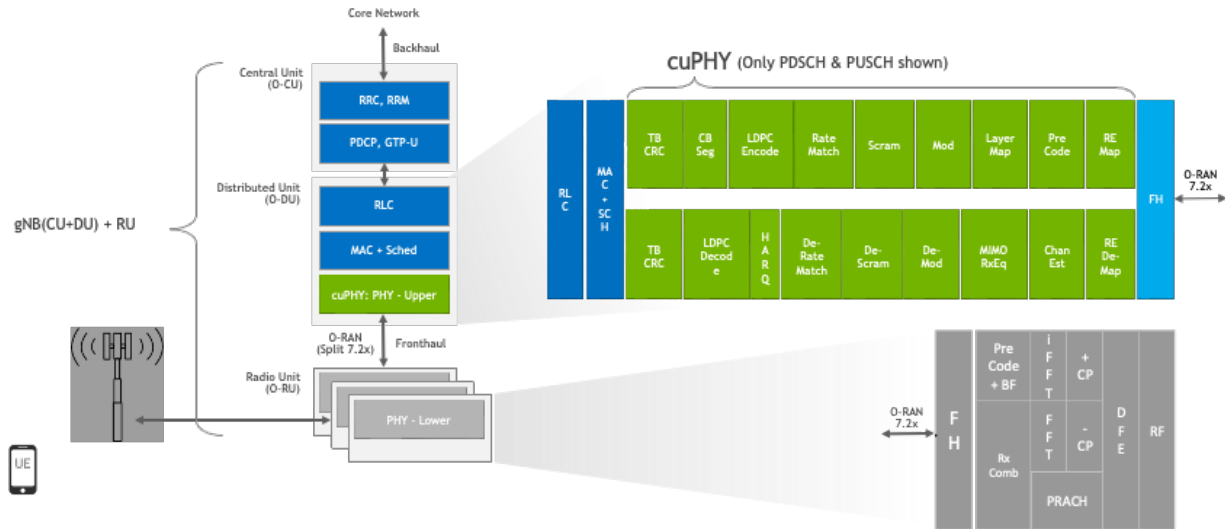


Fig. 4: cuPHY library within 5G NR software stack

cuPHY is optimized to take advantage of the massive parallel processing capability of the GPU architecture by running the workloads in parallel when possible. cuPHY driver orchestrates signal processing tasks running on the GPU. These tasks are organized according to the PHY layer channel type, e.g. PDSCH, PUSCH, SSB, etc. A task related to a given channel is termed as pipeline. For example, PDSCH channel is processed in PDSCH pipeline and the PUSCH channel is processed in PUSCH pipeline. Each pipeline includes a series of functions related to the specific pipeline and consists of multiple CUDA kernels. Each pipeline is capable of running signal processing workloads for multiple cells. The pipelines are dynamically managed for each slot by cuPHY driver with channel aggregate objects. The group of cuPHY channel pipelines that is executed in a given time slot depends on what is scheduled by the L2 in that time slot.

The cuPHY library exposes a set of APIs per PHY channel to create, destroy, setup, configure and run each pipeline as shown in the following figure. L2 adapter translates SCF FAPI messages and other system configurations and cuPHY driver invokes associated cuPHY APIs for each slot. The API's shown as grey such as (Re)-Config, StateUpdate are not currently supported.

The following are descriptions of the APIs in the above figure:

- ▶ **Create:** performs pipeline construction time operations, such as PHY and CUDA object instantiation, memory allocations, etc.
- ▶ **Destroy:** executes teardown procedures of a pipeline and frees allocated resources.
- ▶ **Setup:** sets up PHY descriptors with slot information and batching needed to execute the pipeline.
- ▶ **Run:** launches a pipeline.

The following sections provide more details on the implementation of each cuPHY channel pipeline.

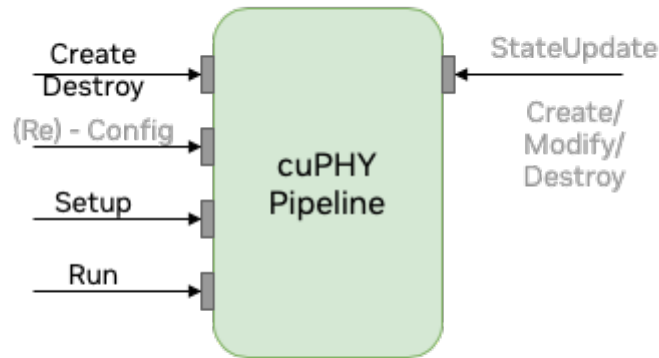


Fig. 5: cuPHY API interface

### 1.7.2.5.1 PDSCH Pipeline

The PDSCH pipeline receives configuration parameters for each cell and the UE and the corresponding DL transport blocks (TBs). After completing the encoding of the PDSCH channel, the pipeline outputs IQ samples mapped to the resource elements (REs) allocated to the PDSCH. The PDSCH pipeline consists of multiple CUDA kernels, which are launched with CUDA graph functionality to reduce the kernel launch overhead. The diagram of the CUDA graph used by PDSCH pipeline is shown in the following figure. The green boxes represent CUDA kernels and the orange boxes represent input and output buffers.

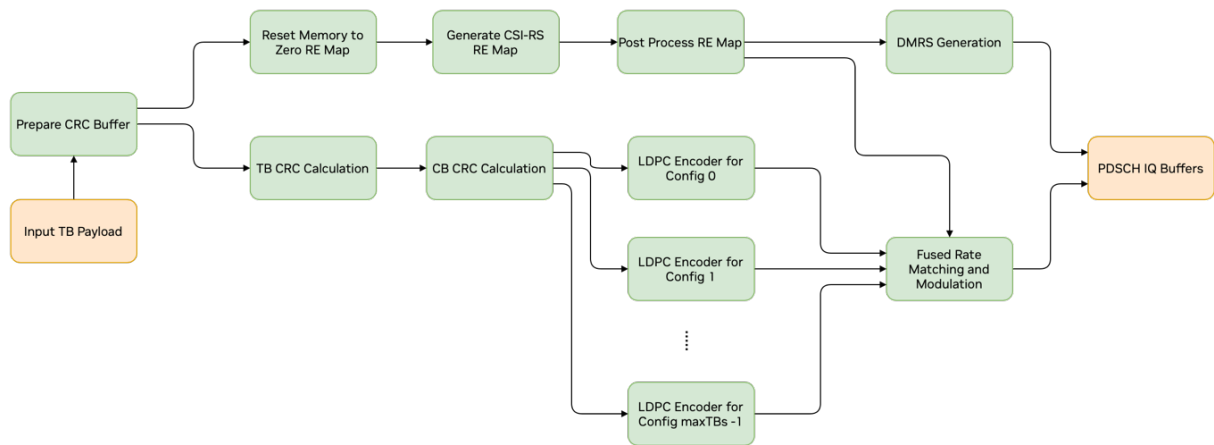


Fig. 6: Graph Diagram of the PDSCH Pipeline

The PDSCH pipeline contains the following components:

- ▶ CRC calculation of the TBs and code-blocks (CBs)
- ▶ LDPC encoding



- ▶ Fused Rate Matching and Modulation Mapper
- ▶ DMRS generation

The CRC calculation component performs the code block segmentation and the CRC calculation. The CRC is calculated first for each TB and then for each CB. The fused rate matching and modulation component performs rate-matching, scrambling, layer-mapping, pre-coding and modulation. This component is also aware of which resource elements it should skip if CSI-RS is configured.

The PDSCH pipeline involves the following kernels:

- ▶ `prepare_crc_buffers`
- ▶ `crcDownlinkPdschTransportBlockKernel`
- ▶ `crcDownlinkPdschCodeBlocksKernel`
- ▶ `ldpc_encode_in_bit_kernel`
- ▶ `fused_dl_rm_and_modulation`
- ▶ `fused_dmrs`

Kernels exercised only if CSI-RS parameters are present are as follows:

- ▶ `zero_memset_kernel`
- ▶ `genCsirsReMap`
- ▶ `postProcessCsirsReMap`

The cuPHY PDSCH transmit pipeline populates parts of a 3D tensor buffer of I/Q samples in GPU memory, where each sample is a complex number using `fp16`, i.e. each sample is a `__half2` using `x` for the real part and `y` for the imaginary part. The output 3D tensor buffer is allocated by the cuPHY driver when the application is first launched and it is reset for every slot (i.e., between successive PDSCH launches) by the cuPHY driver. Here, re-setting the buffer means, it is initialized to all zero values.

The output tensor contains 14 symbols on time domain (x-axis), 273 PRBs (Physical Resource Blocks) on frequency domain (y-axis), and up to 16 layers on spatial domain (z-axis). For the y-axis, each PRB contains 12 REs, and each RE is a `__half2` data. Contiguous PRBs for the same OFDM symbol and spatial layer are allocated next to each other on memory. The resources are mapped in memory in the following order: frequency domain, time domain and then the spatial domain (or layer domain). This is the maximum size of the output buffer needed for a cell per slot.

The PDSCH only fills in parts of that buffer, i.e., its allocated PRBs, based on various configuration parameters it receives that vary over time. Parts of the slot can be filled by other downlink control channels. From a PDSCH standpoint, only the two `fused_*` kernels listed above, `fused_dl_rm_and_modulation` and `fused_dmrs` write to the output buffer. The fused rate-matching and modulation kernel writes data part of the I/Q samples, while the DMRS kernel only writes the DMRS symbols, i.e., only 1 or 2 contiguous symbols in the x-dimension. Note that, unlike other components, DMRS is not dependent on any of the previous pipeline stages.

The PDSCH pipeline expects pre-populated structs `cuphyPdschStatPrms_t` (cuPHY PDSCH static parameters) and `cuphyPdschDynPrms_t` (cuPHY PDSCH dynamic parameters) that include the input data and the necessary configuration parameters.

The TB data input can exist either in CPU or GPU memory depending on the `cuphyPdsch-DatIn_t.pBufferType`. If this is `GPU_BUFFER`, then the host to device (H2D) memory copies for that data can happen before PDSCH setup is executed for each cell. This is called prepone H2D copy and it can be configured by setting the `prepone_h2d_copy` flag in the `l2_adapter_config_*.yaml` file. If prepone H2D copy is not enabled, the copy operations happen as part of PDSCH setup. It is highly recommended that the prepone H2D copy should be enabled to achieve high capacity in a multiple cell scenario.

The way LDPC kernels are initiated can change when multiple TBs are configured on PDSCH. If the LDPC configuration parameters are identical across TBs, PDSCH launches a single LDPC kernel for all TBs (as it is the case for the other PDSCH components). If the LDPC configuration parameters vary across the TBs, then multiple LDPC kernels are launched, one for each unique configuration parameters set. Each LDPC kernel is launched on a separate CUDA stream.

The PDSCH CUDA graph contains only kernel nodes and has the layout shown in the PDSCH graph diagram shown above. As it is not possible to dynamically change the graph geometry at runtime, PDSCH\_MAX\_HET\_LDPC\_CONFIGS\_SUPPORTED potential LDPC kernel nodes are created. Depending on the LDPC configuration parameters and the number of TBs, only a subset of these kernels perform LDPC encoding. The remaining nodes are disabled at runtime if needed per PDSCH. The DMRS kernel node is not dependent on any of the other PDSCH kernels. Therefore, it can be placed anywhere in the graph. The three kernels preceding the DMRS in the graph are only exercised if CSI-RS parameters are present (or CSI-RS is configured). These kernels compute information needed by the fused rate matching and modulation kernel about the REs that need to be skipped.

### 1.7.2.5.2 PDCCH Pipeline

The cuPHY PDCCH channel processing involves the following kernels:

- ▶ encodeRateMatchMultipleDCIsKernel
- ▶ genScramblingSeqKernel
- ▶ genPdcchTfSignalKernel

When running in graphs mode, the CUDA graph launched on every slot contains only kernel nodes and its current layout is as depicted in the below figure.

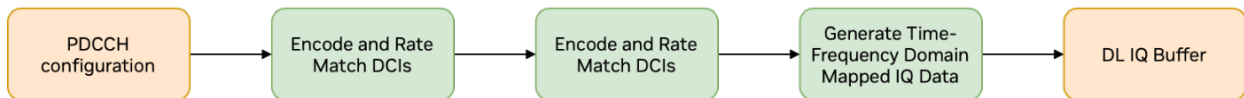


Fig. 7: cuPHY PDCCH graph layout

PDCCH kernel takes static and dynamic parameters as in PDSCH.

Notes on PDCCH configuration and dataset conventions:

- ▶ The PdcchParams dataset contains the coreset parameters for a given cell. Dataset DciParams\_coreset\_0\_dci\_0 contains the DCI parameters for the first DCI of coreset 0. There is a separate dataset for every DCI in a cell with the naming convention: DciParams\_coreset\_<i>\_dci\_<j>, where i has values from 0 up to (number of coresets - 1), while j starts from 0 for every coreset i and goes up to (PdcchParams[i].numDIDci - 1) for that coreset.
- ▶ Dataset DciPayload\_coreset\_0\_dci\_0 contains the DCI payload, in bytes, for the first DCI of coreset 0. It follows the naming convention mentioned above DciParams\_coreset\_0\_dci\_0.
- ▶ Dataset(s) DciPmW\_coreset\_i\_dci\_j hold the precoding matrix for a given DCI, coreset pair, if it has precoding enabled.
- ▶ X\_tf\_fp16 is the 3D output tensor for that cell and is used for reference checks in the various PDCCH examples.
- ▶ X\_tf\_cSamples\_bfp\* datasets that contain compressed data are not used in cuPHY, since compression happens in cuphydriver after all cuPHY processing for all downlink channels scheduled in a slot has completed.

### 1.7.2.5.3 SSB Pipeline

The cuPHY SS Block channel processing involves the following kernels:

- ▶ encodeRateMatchMultipleSSBsKernel
- ▶ ssbModTfSigKernel

When running in graphs mode, the CUDA graph launched on every slot contains only these two kernel nodes connected in sequence.

Notes on SSB configuration and dataset conventions:

- ▶ The SSTxParams dataset contains all the nSsb, SSB parameters for a given cell.
- ▶ SSB bursts cannot be multiplexed in frequency domain, they can only be multiplexed in time domain.
- ▶ nSsb datasets contains the number of SSBs in a cell, this is also the size of the SSTxParams dataset.
- ▶ x\_mib contains the Master Information Block (MIB) for each SSB in the cell as an uint32\_t element; only the least significant 24-bits of each element are valid.
- ▶ Dataset(s) Ssb\_PM\_W\* contain the precoding matrices if precoding is enabled for a given SSB.
- ▶ X\_tf\_fp16 is the 3D output tensor for that cell and is used for reference checks in the various SSB examples. Every I/Q sample there is stored as \_\_half2c.  
X\_tf is similar to X\_tf\_fp16 but every I/Q sample there is stored as float2 instead of \_\_half2; not currently used in cuPHY.
- ▶ X\_tf\_cSamples\_bfp\* datasets hold the output compressed and are not used in cuPHY as compression is applied as part of the cuphydriver.

### 1.7.2.5.4 CSI-RS Pipeline

The cuPHY CSI-RS channel processing involves the following kernels:

- ▶ genScramblingKernel
- ▶ genCsirsTfSignalKernel

When running in graphs mode, the CUDA graph launched on every slot contains only these two kernel nodes connected in sequence.

Notes on CSI-RS configuration and dataset conventions:

- ▶ CsirsParamsList contains configuration parameters which are used for non-zero power signal generation (e.g., NZP, TRS).
- ▶ Please note that CsirsParamsList dataset can have multiple elements. All elements in the dataset can be processed with single setup/run call.
- ▶ X\_tf\_fp16 is the 3D reference output tensor for that cell and is used for reference checks in the various CSI-RS examples. Every I/Q sample there is stored as \_\_half2c.
- ▶ X\_tf is similar to X\_tf\_fp16 but every I/Q sample there is stored as float2 instead of \_\_half2; not currently used in cuPHY.
- ▶ X\_tf\_cSamples\_bfp\* datasets hold the output compressed and are not used in cuPHY as compression is applied as part of cuphydriver.

- ▶ `X_tf_remap` is reference output for RE Map, this is not used currently as current implementation only generates NZP signal.
- ▶ Dataset(s) `Csirs_PM_W*` contain precoding matrices and are used if precoding is enabled.

### 1.7.2.5.5 PUSCH Pipeline

The PUSCH pipeline includes the following components (which are illustrated in the *PUSCH Pipeline Front End* and *PUSCH and CSI Part 1 Decoding* figures):

- ▶ Least squares (LS) channel estimation
- ▶ Minimum Mean Square Error (MMSE) channel estimation
- ▶ Noise and interference covariance estimation
- ▶ Shrinkage and whitening
- ▶ Channel Equalization
- ▶ Carrier frequency offset (CFO) estimation and CFO averaging
- ▶ Timing offset (TO) estimation and averaging.
- ▶ Received signal strength indicator (RSSI) estimation and averaging
- ▶ Noise variance estimation
- ▶ Received signal received power (RSRP) estimation and averaging
- ▶ SNR estimation
- ▶ De-rate matching
- ▶ LDPC backend

If CSI part 2 is configured, the following components are also used (these components are illustrated in the *PUSCH and CSI Part 1 Decoding* and *PUSCH and CSI Part 2 Decoding* figures):

- ▶ Simplex decoder or RM decoder or Polar decoder (for CSI decoding of CSI part 1 depending on the UCI payload size)
- ▶ CSI part 2 de-scrambling and de-rate matching
- ▶ Simplex decoder or RM decoder or Polar decoder (for CSI decoding of CSI part 2 depending on the UCI payload size)

The PUSCH pipeline receives IQ samples, which are provided by order and decompression kernels. The received IQ data is stored in the address `cuphyPuschDataIn_t PhyPuschAggr::DataIn.pTDataRx` as the `cuphyTensorPrm_t` type. The IQ samples are represented by half precision (16-bits) real and imaginary values. The size of the input buffer is multiplication of number of maximum PRBs (273), number of subcarriers per PRB (12), number of OFDM symbols per slot (14) and number of maximum antenna ports per cell (16). This buffer is created for each cell.

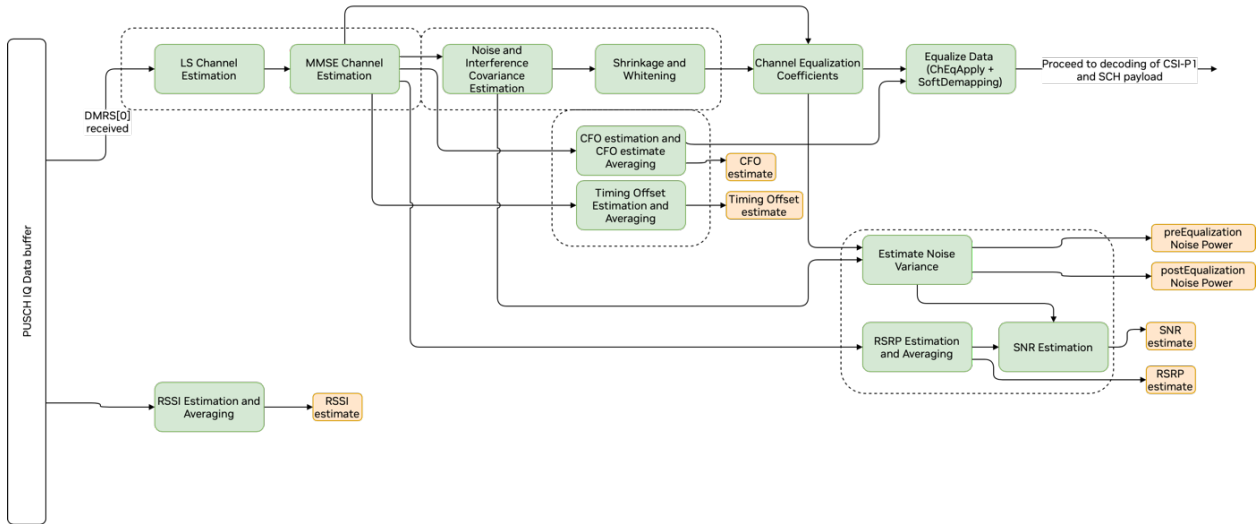


Fig. 8: Graph Diagram of the PUSCH Pipeline Front End

1.7.2.5.5.1 Channel Estimation

First Stage (LS CE)	
Input Buffer	PhyPuschAggr : :DataIn .pTDataRx
Data type	CUPHY_C_16_F: tensor vector of IQ samples
Dimensions	[(ORAN_MAX_PRB*CUPHY_N_TONES_PER_PRB), OFDM_SYMBOLS_PER_SLOT, MAX_AP_PER_SLOT]: [(273*12),14,16]
Description	IQ samples of the input data received from the FH for an UL slot. The I/Q data are represented in half precision float.
Output Buffer	PuschRx : :m_tRefDmrsLSEstVec [ i ] <b>Note:</b> The index i refers to a PRB range.
Data type	CUPHY_C_32_F: float complex IQ samples
Dimensions	[(CUPHY_N_TONES_PER_PRB*(number_of PRBs)/2), NUM_LAYERS, NUM_ANTENNAS, NH]: [(12*(number of PRBs)/2), (number of layers), (number of RX antennas), (number of DMRS symbols)]
Description	IQ samples of the initial channel estimates on DMRS symbols. The I/Q data are represented in half precision float.
Output Buffer	PuschRx::m_tRefDmrsAccumVec[i] <b>Note:</b> the index i refers to a PRB range.
Data Type	CUPHY_C_32_F: float complex IQ samples
Dimensions	[ 1 , 2 ]: Two dimensions for one active and one non-active buffer
Description	Holds summation of $\text{conj}(H_{1s}[k]) * H_{1s}[k+1]$ in a given PRB range, which is then used to calculate mean delay in the next stage. The index k refers to the subcarrier index in a given PRB range. $\text{conj}()$ represents the conjugation function.

Channel estimation (CE) consists of two stages: least-squares (LS) CE and minimum-mean-square (MMSE) CE.

The first LS CE stage invokes a kernel `windowedChEstPreNoDftS0fdmKernel1()`. DMRS symbols are used to obtain initial channel estimate on DMRS REs and to calculate mean delay of the chan-

nel impulse response (CIR). The mean delay and the initial estimates are then used to obtain channel estimates in data REs on the second stage with MMSE filtering operation.

The second stage invokes a dispatch kernel `chEstFilterNoDftS0fdmDispatchKernel()` to support different configurations. The dispatch kernel first calculates mean channel delay by using the stored value `m_tRefDmrsAccumVec` from the first stage. It then chooses an appropriate kernel depending on number of PRBs in the given PUSCH allocation and number of consecutive DMRS symbols (`drvdUeGrpPrms.dmrsMaxLen`). The MMSE filtering operation is done by a kernel `windowedChEstFilterNoDftS0fdmKernel()`.

The component-level unit test of `cuphy_ex_ch_est` based on the testbench of cuPHY PUSCH pipeline can be used to verify the functional correctness of the existing or new PUSCH DMRS channel estimation implemented in CUDA against the 5GModel-generated references. There are several major steps to exploiting `cuphy_ex_ch_est`:

1. Generate `staticApiDataset` to include static parameters for PUSCH pipeline, `dynApiDataset` to include dynamic parameters for PUSCH pipeline, and `evalDataset` to include 5GModel-generated references for the evaluation purpose from cuPHY PUSCH TVs.
2. Create the object `puschRx` of C++ class `PuschRx`, which encapsulates the main functionalities, structs, and internal parameters corresponding to cuPHY PUSCH pipeline from `staticApiDataset` and initialize its internal static parameters.
3. Call `expandFrontEndParameters()` of `puschRx` to initialize the array of struct `cuphyPuschRxUeGrpPrms_t` in CPU by using `dynApiDataset`; allocate GPU device-memory buffers for each UE group to hold input I/Q samples (i.e., `tInfoDataRx`) and channel estimation results (e.g., `tInfoHEst`, `tInfoDmrsLSEst`).
4. Call `cuphyPuschRxChEstGetDescrInfo()` to calculate the sizes of `puschRxChEstStatDescr_t` and `puschRxChEstDynDescr_t`; create the corresponding CPU/GPU buffers to hold static and dynamic parameters (descriptors) (i.e., `puschRxChEstStatDescr_t` and `puschRxChEstDynDescr_t`) used directly as inputs to channel estimation kernels.
5. Call `cuphyCreatePuschRxChEst()` to create a channel estimation object of C++ class `puschRxChEst` and the corresponding handler `puschRxChEstHndl`, initialize `puschRxChEstStatDescr_t`, and return a status code indicating whether the operation was successful or not; copy the contents of `puschRxChEstStatDescr_t` from CPU buffers to GPU buffers.
6. Call `cuphySetupPuschRxChEst()` to populate the `puschRxChEstDynDescr_t` from `cuphyPuschRxUeGrpPrms_t` and other parameters, select/configurate the kernels to be used, and create kernel launch configurations `cuphyPuschRxChEstLaunchCfgs_t` to include kernel node parameters and kernel input arguments; copy the contents of `cuphyPuschRxUeGrpPrms_t` and `puschRxChEstDynDescr_t` from CPU buffers to GPU buffers.
7. Launch channel estimation kernels based on `cuphyPuschRxChEstLaunchCfgs_t` to read input I/Q samples, perform channel estimation, and generate channel estimation results.
8. Destroy the channel estimation object and release the corresponding resources by calling `cuphyDestroyPuschRxChEst()`;
9. Evaluate the channel estimation results by comparing GPU outputs with 5GModel-generated references and report the accuracy of the results.

Second Stage (MMSE CE)	
Input Buffer	PuschRx:: m_tRefDmrsLSEstVec[i]
Input Buffer	PuschRx:: m_tRefDmrsAccumVec[i]
Description	Refer to the <b>First Stage (LS CE)</b> table
Input CE Filters	statDescr.tPrmFreqInterpCoefsSmall statDescr.tPrmFreqInterpCoefs statDescr.tPrmFreqInterpCoefs4
Description	Interpolation filter coefficients depending on the number of PRBs
Data type	CUPHY_C_32_F: float complex IQ samples
Dimensions	[(N_TOTAL_DMRS_INTERP_GRID_TONES_PER_CLUSTER + N_INTER_DMRS_GRID_FREQ_SHIFT), N_TOTAL_DMRS_GRID_TONES_PER_CLUSTER, 3], 3 filters: 1 for middle, 1 lower edge and 1 upper edge tPrmFreqInterpCoefs: [49, 48, 3] tPrmFreqInterpCoefs4: [25, 25, 3] tPrmFreqInterpCoefsSmall: [37, 18, 3]
Description	These CE filters are used to do frequency=domain interpolation and remove FOCC effect. The filter coefficients are different depending on PRB count and PRB location (i.e. edge PRBs have different filter coefficients from central PRBs). These coefficients can be calculated by 5GModel or obtained directly from any cuPHY PUSCH test vectors or cuPhyChEstCoeffs.h5 in aerial_sdk/testVectors.
Input CE Sequences	statDescr.tPrmShiftSeq statDescr.tPrmShiftSeq4 statDescr.tPrmUnShiftSeq statDescr.tPrmUnShiftSeq4
Data type	CUPHY_C_16_F: float complex IQ samples
Dimensions	[(N_DATA_PRB*N_DMRS_GRID_TONES_PER_PRB), 1] tPrmShiftSeq: [48, 1] tPrmShiftSeq4: [24, 1] [(N_DATA_PRB*N_DMRS_INTERP_TONES_PER_GRID*N_DMRS_GRIDS_PER_PRB + N_INTER_DMRS_GRID_FREQ_SHIFT), 1] tPrmUnShiftSeq: [97, 1]tPrmUnShiftSeq4: [49, 1]
Description	These CE sequences are used to shift (and unshift) the estimated channel impulse responses for the filtering purpose. These sequences can be calculated by 5GModel or obtained directly from any cuPHY PUSCH test vectors or cuPhyChEstCoeffs.h5 in aerial_sdk/testVectors. These sequences are only used in the single-stage CE but not two-stage CE which calculates CE sequences online

continues on next page



Table 41 – continued from previous page

Second Stage (MMSE CE)	
Output Buffer	PuschRx::m_tRefHEstVec[i] <b>Note:</b> the index i refers to a PRB range (or UE group)
Data type	CUPHY_C_32_F: float complex IQ samples
Dimensions	[NUM_ANTENNAS, NUM_LAYERS, NF, NH]: [(number of RX antennas), (number of layers), (12*(number of PRBs)), (number of DMRS symbols)]
Description	Estimates of the received channel on the DMRS symbols.

#### 1.7.2.5.5.2 Noise and Interference Covariance Estimation

Input Buffer	Receives outputs of channel estimation kernel as input.
Output Buffer	PuschRx:: m_tRefNoiseVarPreEq
Data type	CUPHY_R_32_F: float real values
Dimensions	[1, NUM_UE_GROUPS]
Description	Estimates of the noise variance pre-equalization per UE group (or PRB range).
Output Buffer	PuschRx:: m_tRefLwlnvVec[i] <i>Note:</i> the index i refers to a PRB range (or UE group)
Data Type	CUPHY_C_32_F: float complex IQ samples
Dimensions	[NUM_ANTENNAS, NUM_ANTENNAS, numPRB]: [(number of RX antennas), (number of RX antennas),(number of PRBs)]
Description	Inverse Cholesky factor of noise-interference tensor information.

#### 1.7.2.5.5.3 Carrier Frequency and Timing Offset Estimation

The carrier frequency offset (CFO) is caused by local oscillators at the UE / RU drifting from the nominal carrier frequency. In the case of UE, the offset will be independent for each UE (but the same for all RF streams). At the RU, the offset is expected to be equal for all RF streams.

CFO can have the following effects on the received signal:

- ▶ Inter-carrier interference (ICI), whereby sub-carriers are not orthogonal
- ▶ A linear phase rotation observed along different symbols (i.e. in the time domain)

CFO estimation is typically based on repetitions over the time domain that allow estimation of the phase rotation. Phase rotation requires a complex multiplication at the equalizer stage, while mitigation of ICI requires a time domain operation or a matrix multiplication. ICI mitigation is not implemented in Aerial.

CFO estimator in Aerial uses channel estimates of the DMRS symbols to calculate a correction factor for the CFO. The algorithm currently supports multiple CFO corrections from multiple UEs multiplexed in FDM mode. It has the following limitations:

- ▶ It is not possible to estimate and compensate for different CFOs originating from multiple UEs multiplexed in CDM mode (e.g. MU-MIMO).
- ▶ CFO compensation is only applied to PUSCH. It requires at least 2 DMRS symbols. If more than two DMRS symbols are available, only 2 are used.
- ▶ Maximum CFO correction is limited to  $\frac{1}{2L}\Delta f$ , where  $L$  is the maximum separation between the DMRS symbols and  $\Delta f$  is the subcarrier spacing.
- ▶ Only phase correction is applied. ICI resulting from CFO is not compensated.

In the following, we formulate the adopted solution for CFO compensation. We assume a single UE for simplicity. The received OFDM signal can be represented as

$$y_n = \left(\frac{1}{N}\right) \left[ \sum_{k=-K}^K X_k H_k e^{j\frac{2\pi n(k+\epsilon)}{N}} \right] + \omega_n, \quad n = 0, 1, \dots, N-1$$

Where  $n$  is the time sample index and  $k$  is the subcarrier index.  $X_k$  is the transmitted QAM symbol and  $H_k$  is the channel coefficient on the subcarrier  $k$ .  $\epsilon$  is the CFO.

After the FFT, we obtain the following:

$$Y_k = (X_k H_k) \left\{ \frac{\sin(\pi\epsilon)}{N \sin\left(\frac{\pi\epsilon}{N}\right)} \right\} e^{j\frac{\pi\epsilon(N-1)}{N}} + I_k + W_k$$

The term  $I_k$  denotes ICI and is given by

$$Y_k = \sum_{l=-K, l \neq k}^K (X_l H_l) \left\{ \frac{\sin(\pi\epsilon)}{N \sin\left(\frac{\pi(l-k+\epsilon)}{N}\right)} \right\} e^{j\frac{\pi\epsilon(N-1)}{N}} e^{-j\frac{\pi\epsilon(l-k)}{N}}$$

ICI degrades the EVM of the received signal, can be expressed as follows (for a normalized signal/channel):

$$EVM = E[|I_k^2|] = \sum_{l=-K, l \neq k}^K E[|H_l|^2] \frac{\sin^2(\pi\epsilon)}{\left(N \sin\left(\frac{\pi(l-k+\epsilon)}{N}\right)\right)^2}$$

Moreover, CFO causes a linear phase variation in the received symbols as follows:

$$Y_{2k} = Y_{1k} e^{j2\pi\epsilon}$$

Where  $Y_{1k}$  and  $Y_{2k}$  are the received signal on subcarrier  $k$  on symbols 1 and 2, respectively. Note that the symbol indices do not correspond to their actual placement in the slot (i.e. they may not be consecutive in the slot).

A maximum likelihood estimator for CFO can be obtained as [12]:

$$\hat{\epsilon} = \left(\frac{1}{2}\right) \tan^{-1} \left\{ \frac{\sum_{k \in k_i} \text{Im}[Y_{2k} Y_{1k}^*]}{\sum_{k \in k_i} \text{Re}[Y_{2k} Y_{1k}^*]} \right\}$$

Where  $k_i$  is the set of REs allocated in a PUSCH transmission.

The maximum correctable offset is  $0.5/L$ , where  $L$  is the time domain separation between the symbols. Aerial algorithm uses DMRS symbols for CFO estimation, which requires at least two DMRS symbols to be configured in a slot.

The preamble detection algorithm of PRACH is capable of handling the maximum CFO without any additional CFO correction. Detection of PUCCH is less sensitive to CFO due to lower modulation order (QPSK) and in some cases shorter duration. PUCCH receiver algorithm does not include CFO correction. If required, CFO correction can be implemented for PUCCH reception in the future.

Timing offset (TO) is caused by a timing misalignment between the UE and the gNB. It results in excess delay of the channel impulse response (CIR). A large enough TO may also result in signal distortion if it causes the CIR to exceed the cyclic prefix.

Assuming that the duration of the CIR + TO is smaller than the cyclic prefix, a TO will manifest itself as a linear phase along the frequency domain, denoted as

$$Y_k = (X_k H_k) e^{-j\pi\tau_0 k} + W_k, n = 0, 1, \dots, N - 1$$

Denote the DMRS channel estimates as for the  $p$ -th antenna,  $l$ -th layer,  $k_1$ -th PRB and  $k_2$ -th RE within PRB  $k$ ,  $k_2 \in \{0, 1, \dots, 10\}$  by  $\hat{H}_{p,l,k_1,k_2,n_d}$  with  $n_d$  as the symbol index out of  $D$  DMRS symbols in a slot. We can obtain the normalized timing offset as

$$\hat{T} = -\frac{1}{2\pi} \text{phase}(R)$$

where

$$R = \sum_{k,l,k_1,k_2,n_d} H_{p,l,k_1,k_2,n_d} H_{p,l,k_1,k_2+1,n_d}^*$$

The absolute timing offset in seconds can be obtained as

$$\hat{t} = \frac{1}{15000 \times 2^\mu} \hat{T}$$

where  $\mu = \{0, 1, 2, 3, 4\}$  is the numerology parameter corresponding to  $\{15, 30, 60, 120, 240\}$  kHz sub carrier frequency spacing.

Input Buffers	PuschRx::m_tRefHEstVec[i] This buffer is received from Channel Estimation kernel. <i>Note:</i> the index $i$ refers to a PRB range (or UE group).
Output Buffer	PuschRx:: m_tRefCfoEstVec[i] <i>Note:</i> the index $i$ refers to a PRB range (or UE group)
Data Type	CUPHY_R_32_F: float real values
Dimensions	[MAX_ND_SUPPORTED, (number of UEs)]: [14, (number of UEs)]
Description	CFO estimate vector.
Output Buffer	PuschRx:: m_tRefCfoHz

continues on next page

Table 42 – continued from previous page

Data Type	CUPHY_R_32_F: float real values.
Dimensions	[1, (number of UEs)]
Descriptions	CFO estimate values in Hz.
Output Buffer	PuschRx:: m_tRefTaEst
Data Type	CUPHY_R_32_F: float real values.
Dimensions	[1, (number of UEs)]
Descriptions	Timing offset estimates.
Output Buffer	PuschRx:: m_tRefCfoPhaseRot
Data Type	CUPHY_C_32_F: float complex values.
Dimensions	[CUPHY_PUSCH_RX_MAX_N_TIME_CH_EST, CUPHY_PUSCH_RX_MAX_N_LAYERS_PER_UE_GROUP, MAX_N_USER_GROUPS_SUPPORTED] [(max number of channel estimates in time, =4), (max layers per UE group, =8), (max UE groups, =128)]
Descriptions	Carrier offset phase rotation values
Output Buffer	PuschRx:: m_tRefTaPhaseRot
Data Type	CUPHY_C_32_F: float complex values.
Dimensions	[1, CUPHY_PUSCH_RX_MAX_N_LAYERS_PER_UE_GROUP] : [1, (max layers per UE group, =8)]
Descriptions	Carrier offset phase rotation values

#### 1.7.2.5.5.4 Soft De-mapper

After equalization, the LLR of each bit is calculated according to the following table for the QAM symbol:  $Z_r + Z_j$  where  $Z_r$  and  $Z_j$  are the real and imaginary components of the symbol. The LLR of each bit will be scaled by postEqMSE of each symbol as the output of the soft-demapper.

#### 1.7.2.5.5.5 4QAM

$A$

$$\frac{1}{\sqrt{2}}$$

**LLR of Real Bits**

$$\lambda_{c_0} = Z_r$$

**LLR of Imaginary Bits**

$$\lambda_{c_0} = Z_i$$

#### 1.7.2.5.5.6 16QAM

$A$

$$\frac{1}{\sqrt{10}}$$

**LLR of Real Bits**

$$\lambda_{c_0} = Z_r$$

$$\lambda_{c_1} = -|Z_r| + 2A$$

**LLR of Imaginary Bits**

$$\lambda_{c_0} = Z_i$$

$$\lambda_{c_1} = -|Z_i| + 2A$$

#### 1.7.2.5.5.7 64QAM

$A$

$$\frac{1}{\sqrt{42}}$$

**LLR of Real Bits**

$$\lambda_{c_0} = Z_r$$

$$\lambda_{c_1} = -|Z_r| + 4A$$

$$\lambda_{c_2} = -||Z_r| - 4A| + 2A$$

**LLR of Imaginary Bits**

$$\lambda_{c_0} = Z_i$$

$$\lambda_{c_1} = -|Z_i| + 4A$$

$$\lambda_{c_2} = -||Z_i| - 4A| + 2A$$

**1.7.2.5.5.8 256QAM** $A$ 

$$\frac{1}{\sqrt{170}}$$

**LLR of Real Bits**

$$\lambda_{c_0} = Z_r$$

$$\lambda_{c_1} = -|Z_r| + 8A$$

$$\lambda_{c_2} = -||Z_r| - 8A| + 4A$$

$$\lambda_{c_3} = -|||Z_r| - 8A| - 4A| + 2A$$

**LLR of Imaginary Bits**

$$\lambda_{c_0} = Z_i$$

$$\lambda_{c_1} = -|Z_i| + 8A$$

$$\lambda_{c_2} = -||Z_i| - 8A| + 4A$$

$$\lambda_{c_3} = -|||Z_i| - 8A| - 4A| + 2A$$

<b>Channel Equalization Coefficients Computation Kernel</b>	
Input Buffers	PuschRx::m_tRefHEstVec[i], PuschRx::m_tRefLwInvVec[i], PuschRx::m_tRefCfoEstVec[i] These buffers are received from Noise and Interference Covariance Estimation, Channel Estimation and CFO Estimation kernels. <b>Note:</b> The index <i>i</i> refers to a PRB range (or UE group).
Output Buffer	PuschRx:: m_tRefReeDiagInvVec[i] <b>Note:</b> The index <i>i</i> refers to a PRB range (or UE group)
Data Type	CUPHY_R_32_F: float real values
Dimensions	[CUPHY_N_TONES_PER_PRB, NUM_LAYERS, NUM_PRBS, nTimeChEq]: [12*(number of PRBs), (number of layers), (number of PRBs), (number of time domain estimates)]
Description	Channel equalizer residual error vector.
Output Buffer	PuschRx:: m_tRefCoefVec[i] <b>Note:</b> The index <i>i</i> refers to a PRB range (or UE group)
Data Type	CUPHY_C_32_F: float complex IQ samples
Dimensions	[NUM_ANTENNAS, CUPHY_N_TONES_PER_PRB, NUM_LAYERS, NUM_PRBS, NH]: [(number of RX antennas), 12*(number of PRBs), (number of layers), (number of PRBs), (number of DMRS positions)]
Descriptions	Channel equalizer coefficients.

Channel Equalization MMSE Soft De-mapping Kernel	
Input Buffers	PuschRx:: m_tRefCoefVec[ i], PuschRx : :m_tRefCfoEstVec[ i], PuschRx : m_tRefReeDiagInvVec[ i] PuschRx:: m_drvdUeGrpPrmsCpu[ i].tInfoDataRx These buffers are received from Noise and Interference Covari- ance Estimation, Channel Estimation and CFO Estimation ker- nels. <i>Note:</i> the index i refers to a PRB range (or UE group).
Output Buffer	PuschRx:: m_tRefDataEqVec[ i] <i>Note:</i> the index i refers to a PRB range (or UE group)
Data Type	CUPHY_C_16_F: tensor vector of half float IQ samples.
Dimensions	[NUM_LAYERS, NF, NUM_DATA_SYMS ]: [(number of layers), 12*(number of PRBs), (number of data OFDM symbols)]
Description	Equalized QAM data symbols.
Output Buffer	PuschRx:: m_tRefLLRVec[ i] <b>Note:</b> the index i refers to a PRB range (or UE group)
Data Type	CUPHY_R_16_F : tensor vector of half float real samples.
Dimensions	[CUPHY_QAM_256, NUM_LAYERS, NF, NUM_DATA_SYMBOLS ]: [(number of bits for 256QAM = 8), (number of layers), (number of layers), 12*(number of PRBs), (number of data OFDM symbols)]
Descriptions	Output LLRs or softbits. Used if UCI on PUSCH is enabled.
Output Buffer	PuschRx:: m_tRefLLRCdm1Vec[ i] <b>Note:</b> the refers to a PRB range (or UE group)index i
Data Type	CUPHY_R_16_F: tensor vector of half float real samples.
Dimensions	[CUPHY_QAM_256, NUM_LAYERS, NF, NUM_DATA_SYMBOLS ]: [(number of bits for 256QAM = 8), (number of layers), (number of layers), 12*(number of PRBs), (number of data OFDM symbols)]
Descriptions	Output LLRs or softbits. Used if there is no UCI on PUSCH.



### 1.7.2.5.5.9 De-rate matching and Descrambling

Input Buffer	PuschRx::m_tRefLLRVec[i] or PuschRx::m_tRefLLRCdm1Vec[i], PuschRx::m_pTbPrmsGpu
Output Buffer	PuschRx::m_pHarqBuffers
Data type	uint8_t
Dimensions	Function of TB size and number of TBs.
Description	Rate-matching/descrambling output. It is on a host pinned GPU memory. It is mapped to PhyPuschAggr::DataInOut.pHarqBuffersInOut

### 1.7.2.5.5.10 RSSI Estimation

The RSSI is calculated from the received signal by first calculating the received signal power on each RE and each receive antenna. The total power is then calculated by summation of received power across the frequency resources and receive antennas. The RSSI is then obtained by averaging over DMRS symbols as defined in the SCF FAPI specification.

The RSSI is calculated as

$$R_{RSSI} = \frac{1}{D} \sum_{p,k,n_d} Y_{p,k,n_d} Y_{p,k,n_d}^*$$

where  $Y_{p,k,n_d}$  is the received signal of the  $p$ -th receive antenna, the  $k$ -th subcarrier and the  $n_d$ -th OFDM symbol of the  $d$ -th DMRS symbol.

Input Buffer	PuschRx::m_drvdUeGrpPrmsCpu[i].tInfoDataRx
Output Buffer	PuschRx::m_tRefRssiFull
Data type	CUPHY_R_32_F : tensor vector of float real samples.
Dimensions	[MAX_ND_SUPPORTED, MAX_N_ANTENNAS_SUPPORTED, nUEgroups]: [(max number of time domain estimates, = 14), (max number of antennas, =64), (number of UE groups)]
Description	Measured RSSI (per symbol, per antenna, per UE group).
Output Buffer	PuschRx::m_tRefRssi
Data type	CUPHY_R_32_F : tensor vector of float real samples.
Dimensions	[1, nUEgroups]:[1, (number of UE groups)]
Description	Measured RSSI per UE group.

### 1.7.2.5.5.11 RSRP and SINR Estimation

The RSRP is calculated as

$$R_{RSRP} = \frac{1}{PKD} \sum_{p,l,k,n_d} H_{p,l,k,n_d} H_{p,l,k,n_d}^*$$

Where  $H_{p,l,k,n_d}$  is the estimated channel frequency response of the

$p$ -th receive antenna,  $l$ -th layer,  $k$ -th subcarrier and  $n_d$ -th OFDM symbol of the  $D$  DMRS symbols. In the equation,  $P$  is the total number of receive antennas,  $K$  is the total number of subcarriers and  $D$  is the total number of DMRS symbols in a slot.

In order to obtain an SINR estimation, we first obtain the noise signal as

$$\tilde{r}_{p,k_{DMRS},n_d} = Y_{p,k_{DMRS},n_d} - \sum_l H_{p,l,k_{DMRS},n_d} X_{DMRS,l}$$

Where  $Y_{p,k_{DMRS},n_d}$  is the received signal of the  $p$ -th receive antenna, the  $k_{DMRS}$ -th DMRS subcarrier and the  $n_d$ -th DMRS symbol.  $H_{p,l,k_{DMRS},n_d}$  is the estimated channel response of the  $p$ -th receive antenna,  $l$ -th layer,  $k_{DMRS}$ -th DMRS subcarrier and the  $n_d$ -th OFDM symbol of the  $d$ -th DMRS symbol.  $X_{DMRS,l}$  is the DMRS symbol of the  $l$ -th layer.

The noise variance can then be estimated as

$$\sigma_{noise}^2 = \frac{1}{PK_{DMRS}D} \sum_{p,k,n_d} \tilde{r}_{p,k_{DMRS},n_d} \tilde{r}_{p,k_{DMRS},n_d}^*$$

Where  $P$  is the total number of receive antennas and  $K_{DMRS}$  is the total number of subcarriers in a DMRS symbol. In order to compensate for the reduction in the noise power estimation caused by the channel estimation filter, a correction factor (not shown here) is added to the noise variance. The SINR can then be obtained by  $SINR = \frac{1}{\sigma_{noise}^2}$

Input Buffer	PuschRx::m_tRefHEstVec[i], m_tRefNoiseVarPreEq	PuschRx::	m_tRefReeDiagInvVec[i],	PuschRx::
Output Buffer	PuschRx:: m_tRefRsrp			
Data type	CUPHY_R_32_F : tensor vector of float real samples.			
Dimensions	[1, nUEgroups]:[1, (number of UE groups)]			
Description	RSRP values across UEs.			
Output Buffer	PuschRx:: m_tRefNoiseVarPostEq			
Data type	CUPHY_R_32_F : tensor vector of float real samples.			
Dimensions	[1, nUEgroups]:[1, (number of UE groups)]			
Description	Post-equalization noise variances across UEs			
Output Buffer	PuschRx:: m_tRefSinrPreEq			
Data type	CUPHY_R_32_F : tensor vector of float real samples.			
Dimensions	[1, nUEgroups]:[1, (number of UE groups)]			
Description	Pre-equalization SINR values across UEs.			
Output Buffer	PuschRx:: m_tRefSinrPostEq			
Data type	CUPHY_R_32_F : tensor vector of float real samples.			
Dimensions	[1, nUEgroups]:[1, (number of UE groups)]			
Description	Post-equalization SINR values across UEs.			

#### 1.7.2.5.5.12 UCI on PUSCH Decoder

If UCI is configured on PUSCH channel, output of the soft-demapper first goes through de-segmentation to separate HARQ, CSI part 1 and CSI part 2 and SCH softbits (or LLRs). This initial step is done by the kernel `uciOnPuschSegLLRs0Kernel()`.

If CSI-part2 is present, CSI-part2 control kernel is launched as shown in the figure below as a dashed box. This kernel determines the number of CSI-part2 bits and rate-matched bits and selects the correct decoder kernels and initiates their setup functions.

De-segmentation of CSI-part2 payload is done by `uciOnPuschSegLLRs2Kernel()` kernel, which separates CSI-part2 UCI and SCH softbits.

UCI on PUSCH De- segmentation of First Phase	
Input Buffer	PuschRx:: m_tPrmLLRVec[i]
Output Buffer	PuschRx::m_pTbPrmsGpu->pUePrmsGpu[i].d_harqLLRs;
Data type	__half*
Dimensions	Single dimensional array, the size depending on the payload.
Description	HARQ soft bits.
Output Buffer	PuschRx::m_pTbPrmsGpu->pUePrmsGpu[ueIdx].d_csi1LLRs;
Data type	__half*
Dimensions	Single dimensional array, the size depending on the payload.
Description	CSI part 1 soft bits.
Output Buffer	PuschRx::m_pTbPrmsGpu->pUePrmsGpu[i].d_schAndCsi2LLRs
Data type	__half*
Dimensions	Single dimensional array, the size depending on the payload.
Description	Shared channel (SCH) and CSI part 2 soft bits.

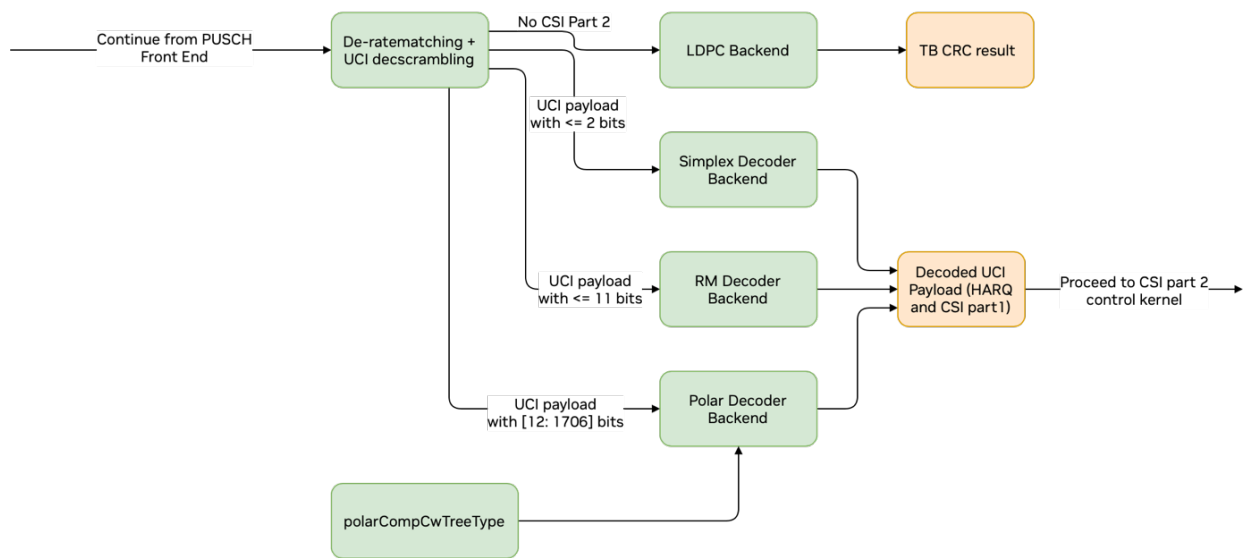


Fig. 9: Graph Diagram of the PUSCH and CSI Part 1 Decoding

UCI on PUSCH De- segmentation of Second Phase	
Input Buffer	PuschRx:: m_tPrmLLRVec[i]
Output Buffer	PuschRx::m_pTbPrmsGpu->pUePrmsGpu[i].d_schAndCsi2LLRs;
Data type	__half*
Dimensions	Single dimensional array, the size depending on the payload.
Description	Pointer to SCH softbits
Output Buffer	PuschRx::m_pTbPrmsGpu->pUePrmsGpu[i].d_schAndCsi2LLRs + PuschRx::m_pTbPrmsGpu->pUePrmsGpu[i].G;
Data type	__half*
Dimensions	Single dimensional array, the size depending on the payload.
Description	Pointer to CSI part2 softbits

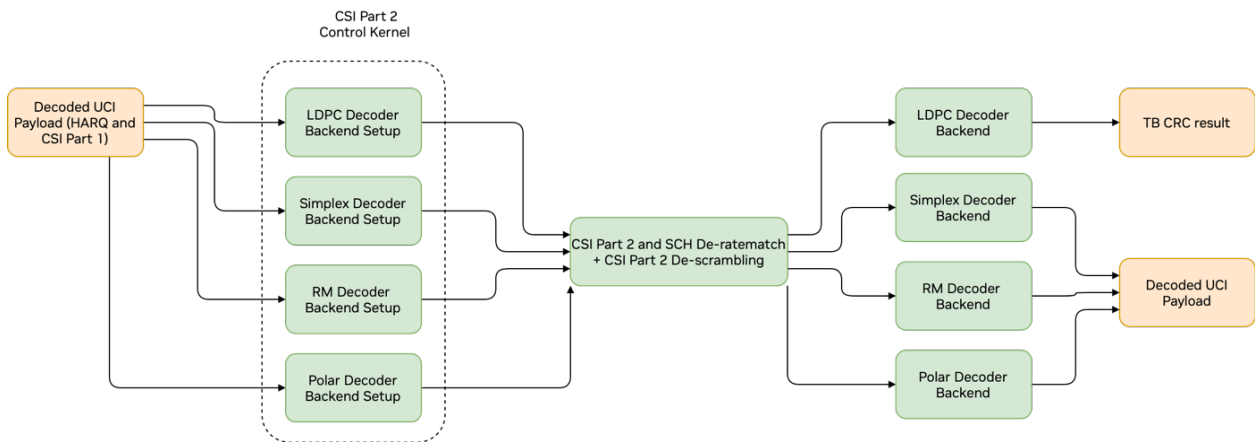


Fig. 10: Graph Diagram of the PUSCH and CSI Part 2 Decoding

### 1.7.2.5.5.13 Simplex Decoder

The simplex decoder implements maximum likelihood (ML) decoder. It receives input LLRs and outputs estimated codewords. It also reports HARQ DTX status.

Input Buffer	PuschRx:: m_pSpxCwPrmsCpu[spxCwIdx].d_LLRs
Data type	__half*
Dimensions	Single dimensional array, the size depending on the payload.
Description	Pointer to input LLRs
Output Buffer	PuschRx:: m_pSpxCwPrmsCpu[spxCwIdx].d_cbEst
Data type	uint32_t*
Dimensions	Single dimensional array, the size depending on the payload.
Description	Decoded UCI payload.
Output Buffer	PuschRx:: m_pSpxCwPrmsCpu[spxCwIdx].d_DTXStatus
Data type	Uin8_t*
Dimensions	Parameter.
Description	Pointer to HARQ detection status.

### 1.7.2.5.5.14 Reed Muller (RM) Decoder

The RM decoder implements maximum likelihood (ML) decoder. It receives input LLRs and outputs estimated codewords. It also reports HARQ DTX status.

Input Buffer	PuschRx:: m_pSpxCwPrmsCpu[rmCwIdx].d_LLRs
Data type	__half*
Dimensions	Single dimensional array, the size depending on the payload.
Description	Pointer to input LLRs
Output Buffer	PuschRx:: m_pSpxCwPrmsCpu[rmCwIdx].d_cbEst
Data type	uint32_t*
Dimensions	Single dimensional array, the size depending on the payload.
Description	Decoded UCI payload.
Output Buffer	PuschRx:: m_pSpxCwPrmsCpu[rmCwIdx].d_DTXStatus
Data type	Uint8_t*
Dimensions	Parameter.
Description	Pointer to HARQ detection status.

#### 1.7.2.5.5.15 Polar Decoder

Polar decoder uses CRC aided list decoder with tree pruning. There are many variants of the decoding algorithm that is used in decoding of Polar codes. Please see [2, 3] for some of the related work. The exact implementation in cuPHY is optimized for the GPU architecture.

The tree-pruning algorithms combine leaf nodes together, which is a better data structure for execute decoding in parallel. Hence it is more suitable for GPU architecture. There are different methods of forming leaf nodes in the tree pruning algorithm. In our implementation we use rate-0 and rate-1 leaf codewords. In rate-0 leaf nodes, multiple bits are always frozen and are zero, whereas there are no frozen bits in rate-1 leaf nodes. In rate-1 codewords, LLRs can be decoded in parallel.

Tree pruning is done by `compCwTreeTypesKernel()` before the input LLRs are received by the Polar Decoder kernel.

If the list size is equal to 1, `polarDecoderKernel()`, if the list size is greater than 1, `listPolarDecoderKernel()` is run.

Input Buffer	PuschRx:: m_cwTreeLLRsAddrVec
Data type	__half*
Dimensions	Single dimensional array, the size depending on the payload.
Description	Pointer to codeword tree of LLR addresses.
Output Buffer	PuschRx:: m_cbEstAddrVec
Data type	uint32_t*
Dimensions	Single dimensional array, the size depending on the payload.
Description	Pointer to estimated CB addresses.

### 1.7.2.5.5.16 LDPC Decoder

LDPC decoder is implemented with normalized layered min-sum algorithm [1] and it uses short float (FP16) data type as log-likelihood ratio (LLR) metrics.

Input Buffer	PuschRx:: m_LDPCDecodeDescSet.llr_input[m_LDPCDecodeDescSet .num_tbs] The first address is also mapped to PuschRx::m_pHarqBuffers[uelIdx]
Data type	cuphyTransportBlockLLRDesc_t
Dimensions	Single dimensional array, the size depending on the number of valid TB descriptors. The max size is 32.
Description	Input LLR buffers.
Output Buffer	PuschRx:: m_LDPCDecodeDescSet.tb_output[m_LDPCDecodeDescSet .num_tbs] The first address is also mapped to PuschRx::d_LDPCOut + offset Offset is a function of UE index and number of codewords per UE.
Data type	cuphyTransportBlockDataDesc_t
Dimensions	Single dimensional array, the size depending on the number of valid TB descriptors.
Description	Pointer to estimated TB addresses.



### 1.7.2.5.5.17 CRC Decoder

Code Block CRC Decoder Kernel	
Input Buffer	PuschRx::d_pLDPCOut, PuschRx:: m_pTbPrmsGpu
Descriptions	LDPC decoder output and TB parameters needed to decode the CRC.
Output Buffer	PuschRx:: m_outputPrms.pCbCrcsDevice;
Data type	uint32_t
Dimensions	[1, total number of CBs (across UEs)]
Description	CRC output.
Output Buffer	PuschRx:: m_outputPrms.pTbPayloadsDevice
Data type	Uin8_t
Dimensions	[1, total number of TB payload bytes]
Description	TB payload.
Transport Block CRC Decoder Kernel	
Input Buffer	PuschRx:: m_outputPrms.pTbPayloadsDevice, PuschRx:: m_pTbPrmsGpu
Output Buffer	PuschRx:: m_outputPrms.pTbCrcsDevice
Data Type	uint32_t
Dimensions	[1, total number of TBs (across UEs)]
Description	TB CRC output.

### 1.7.2.5.6 PUCCH Pipeline

The PUCCH pipeline can be divided into logical stages. The first, front-end processing, is unique for each PUCCH format and involves descrambling and demodulation to recover transmitted symbols. For formats 0 and 1, this is the only stage performed as there is no decoding necessary to recover data. For formats 2 and 3, this is followed by decoding. Here, the kernels used are the same as those in PUSCH for the same decoding type. Finally, the decoded data is segmented into HARQ, SR and CSI payloads.

The kernels responsible for front-end processing are as follows:

- ▶ pucchFORxKernel
- ▶ pucchF1RxKernel
- ▶ pucchF2RxKernel

► pucchF3RxKernel

With each corresponding to formats 0 through 3 respectively. For formats 0 and 1, hard decisions are made as part of demodulation to recover 1 or 2 payload bits, depending on specific configuration. For formats 2 and 3, LLRs are recovered from demodulation and used for decoding. Each front-end processing kernel also calculates RSSI, and RSRP and uses DMRS to perform SINR, interference, and timing advance estimation.

For formats 2 and 3, payloads less than 12 bits in length are handled by the Reed Muller decoder kernel. Payloads of 12 bits and larger are handled by a de-rate matching and de-interleaving kernel (polSegDeRmDeItlKernel) and then processed by the polar decoder kernel.

Finally, formats 2 and 3 decoded payloads are segmented by a segmentation kernel (pucchF234UciSegKernel) to recover the corresponding HARQ, SR, and CSI payloads.

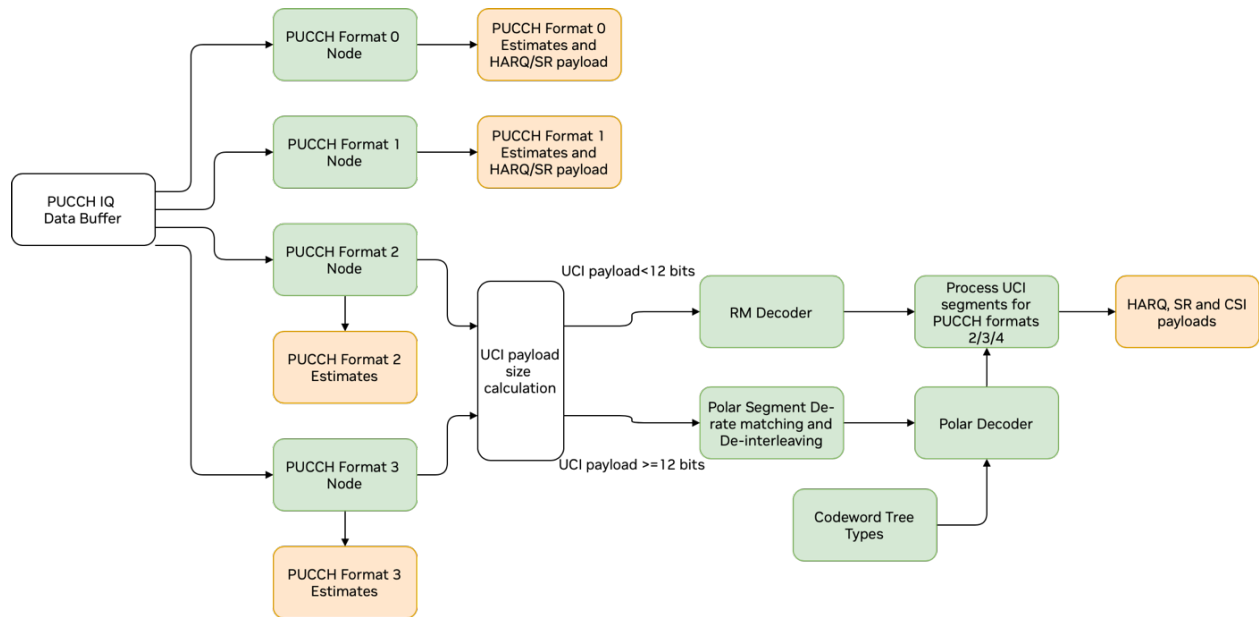


Fig. 11: Graph Diagram of the PUCCH Pipeline

Input Buffer	PucchRx::m_tPrmDataRxBufCpu[i].tInfoDataRx
Data type	CUPHY_C_16_F : tensor vector of IQ samples
Dimensions	[(ORAN_MAX_PRB*CUPHY_N_TONES_PER_PRB), OFDM_SYMBOLS_PER_SLOT, MAX_AP_PER_SLOT]
Output Buffer	PucchRx::m_outputPrms.pFOUciOutGpu
Data type	cuphyPucchF0F1UciOut_t*
Dimensions	Single dimensional array of length equal to the number of format 0 UCIs
Description	HARQ values and estimator measurements, including SINR, Interference, RSSI, RSRP (in dB) and timing advance (in uSec) per UCI

continues on next page

Table 43 – continued from previous page

Output Buffer	PucchRx::m_outputPrms.pFOUciOutGpu
Data type	cuphyPucchF0F1UciOut_t*
Dimensions	Single dimensional array of length equal to the number of format 1 UCIs
Description	HARQ values and estimator measurements, including SINR, Interference, RSSI, RSRP (in dB) and timing advance (in uSec) per UCI
Output Buffer	PucchRx:: m_tSinr
Data type	CUPHY_R_32_F : tensor vector of float values.
Dimensions	[(number of format 2 & 3 UCIs)]
Description	Measured SINR per UCI (in dB)
Output Buffer	PucchRx:: m_tRssi
Data type	CUPHY_R_32_F : tensor vector of float values.
Dimensions	[(number of format 2 & 3 UCIs)]
Description	Measured RSSI per UCI (in dB)
Output Buffer	PucchRx:: m_tRsrp
Data type	CUPHY_R_32_F : tensor vector of float values.
Dimensions	[(number of format 2 & 3 UCIs)]
Description	Measured RSRP per UCI (in dB)
Output Buffer	PucchRx:: m_tInterf
Data type	CUPHY_R_32_F : tensor vector of float values.
Dimensions	[(number of format 2 & 3 UCIs)]
Description	Measured Interference per UCI (in dB)
Output Buffer	PucchRx:: m_tNoiseVar
Data type	CUPHY_R_32_F : tensor vector of float values.
Dimensions	[(number of format 2 & 3 UCIs)]
Description	Measured Noise Variance per UCI (in dB)
Output Buffer	PucchRx:: m_tTaEst
Data type	CUPHY_R_32_F : tensor vector of float values.
Dimensions	[(number of format 2 & 3 UCIs)]

continues on next page

Table 43 – continued from previous page

Description	Measured Timing Advance per UCI (in uSec)
Output Buffer	PucchRx::m_tUciPayload
Data type	CUPHY_R_8U : tensor vector of unsigned bytes
Dimensions	[(total number payload bytes for format 2 & 3 UCIs rounded up to 4-byte words for each payload)]
Description	Format 2 & 3 UCI payloads rounded to 4-byte words. If 1 UCI has HARQ & CSI-P1 of 1 bit each, they will each get a 4-byte word for a total of 8 bytes.
Output Buffer	PucchRx:: m_tHarqDetectionStatus
Data type	CUPHY_R_8U : tensor vector of unsigned bytes
Dimensions	[(number of format 2 & 3 UCIs)]
Description	HARQ detection status
Output Buffer	PucchRx:: m_tCsiP1DetectionStatus
Data type	CUPHY_R_8U : tensor vector of unsigned bytes
Dimensions	[(number of format 2 & 3 UCIs)]
Description	CSI Part 1 detection status
Output Buffer	PucchRx:: m_tCsiP2DetectionStatus
Data type	CUPHY_R_8U : tensor vector of unsigned bytes
Dimensions	[(number of format 2 & 3 UCIs)]
Description	CSI Part 2 detection status

### 1.7.2.5.7 PRACH Pipeline

The PRACH pipeline uses IQ samples segmented for each occasion and performs detection and estimation for configured PRACH signals. This process operates across a number of kernels as follows:

1. The `prach_compute_correlation` kernel takes input IQ data and performs averaging among repetitions followed by a time-domain correlation (done in frequency domain) against a reference version of the expected PRACH signal. This kernel simultaneously operates on each PRACH occasion.
2. An inverse FFT kernel transforms the frequency domain correlation results to time domain. A separate kernel operates on each occasion.
3. The `prach_compute_pdp` kernel performs non-coherent combining of correlation results for each preamble zone. It then calculates power and the peak index and value for each preamble zone.
4. The `prach_search_pdp` kernel computes preamble and noise power estimates and reports the preamble index with peak power. It also does threshold-based detection declaration.

There is also a separate set of kernels as part of the PRACH pipeline for performing RSSI calculations.

1. The memsetRssi kernel clears a device buffer used in computing RSSI.
2. The prach\_compute\_rssi kernel computes RSSI for each PRACH occasion both for each antenna and average power over all antennas
3. The memcpyRssi kernel stores the RSSI results in host-accessible memory

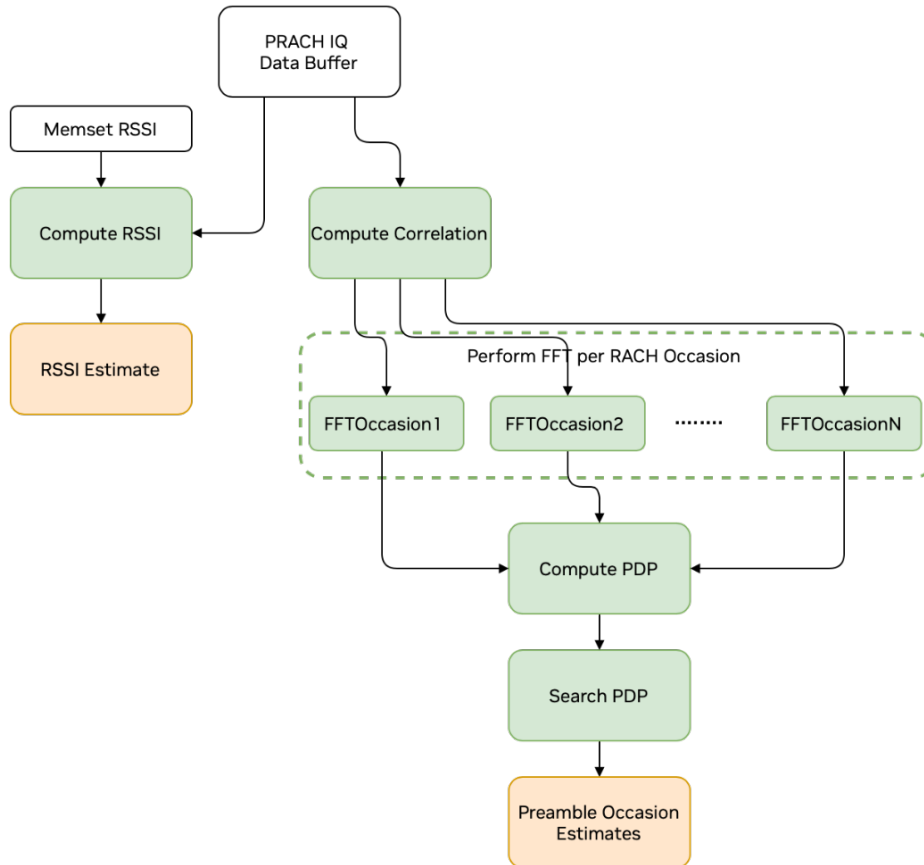


Fig. 12: Graph Diagram of the PRACH Pipeline

Input Buffer	PrachRx:: h_dynParam[i].dataRx
Data type	CUPHY_C_16_F : tensor for each occasion buffer
Dimensions	[(Preamble length+5)*Number of repetitions , N_ant]
Output Buffer	PrachRx:: numDetectedPrmb
Data type	CUPHY_R_32U : tensor vector of uint32
Dimensions	[1, PRACH_MAX_OCCASIONS_AGGGR]
Description	Number of detected preambles for each occasion

continues on next page

Table 44 – continued from previous page

Output Buffer	PrachRx:: prmbIndexEstimates
Data type	CUPHY_R_32U : tensor vector of uint32
Dimensions	[PRACH_MAX_NUM_PREAMBLES, PRACH_MAX_OCCASIONS_AGGR]
Description	Detected preamble index for each preamble and occasion
Output Buffer	PrachRx:: prmbDelayEstimates
Data type	CUPHY_R_32_F : tensor vector of float values
Dimensions	[PRACH_MAX_NUM_PREAMBLES, PRACH_MAX_OCCASIONS_AGGR]
Description	Delay estimate for each preamble and occasion
Output Buffer	PrachRx:: prmbPowerEstimates
Data type	CUPHY_R_32_F : tensor vector of float values
Dimensions	[PRACH_MAX_NUM_PREAMBLES, PRACH_MAX_OCCASIONS_AGGR]
Description	Power estimate for each preamble and occasion
Output Buffer	PrachRx:: antRssi
Data type	CUPHY_R_32_F : tensor vector of float values
Dimensions	[N_ant, PRACH_MAX_OCCASIONS_AGGR]
Description	RSSI for each antenna and occasion
Output Buffer	PrachRx:: rssi
Data type	CUPHY_R_32_F : tensor vector of float values
Dimensions	[1, PRACH_MAX_OCCASIONS_AGGR]
Description	RSSI for each occasion
Output Buffer	PrachRx:: interference
Data type	CUPHY_R_32_F : tensor vector of float values
Dimensions	[1, PRACH_MAX_OCCASIONS_AGGR]
Description	Interference for each occasion
Output Buffer	PrachRx:: prmbPowerEstimates
Data type	CUPHY_R_32_F : tensor vector of float values
Dimensions	[PRACH_MAX_NUM_PREAMBLES, PRACH_MAX_OCCASIONS_AGGR]
Description	Power estimate for each preamble and occasion

continues on next page

Table 44 – continued from previous page

Output Buffer	PrachRx:: antRssi
Data type	CUPHY_R_32_F : tensor vector of float values
Dimensions	[N_ant, PRACH_MAX_OCCASIONS_AGGR]
Description	RSSI for each antenna and occasion
Output Buffer	PrachRx:: rssi
Data type	CUPHY_R_32_F : tensor vector of float values
Dimensions	[1, PRACH_MAX_OCCASIONS_AGGR]
Description	RSSI for each occasion
Output Buffer	PrachRx:: interference
Data type	CUPHY_R_32_F : tensor vector of float values
Dimensions	[1, PRACH_MAX_OCCASIONS_AGGR]
Description	Interference for each occasion

### 1.7.2.5.8 SRS Pipeline Overview

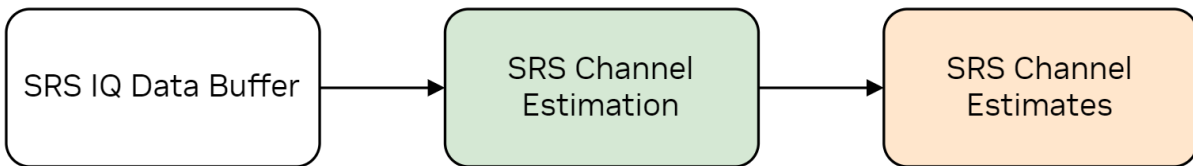
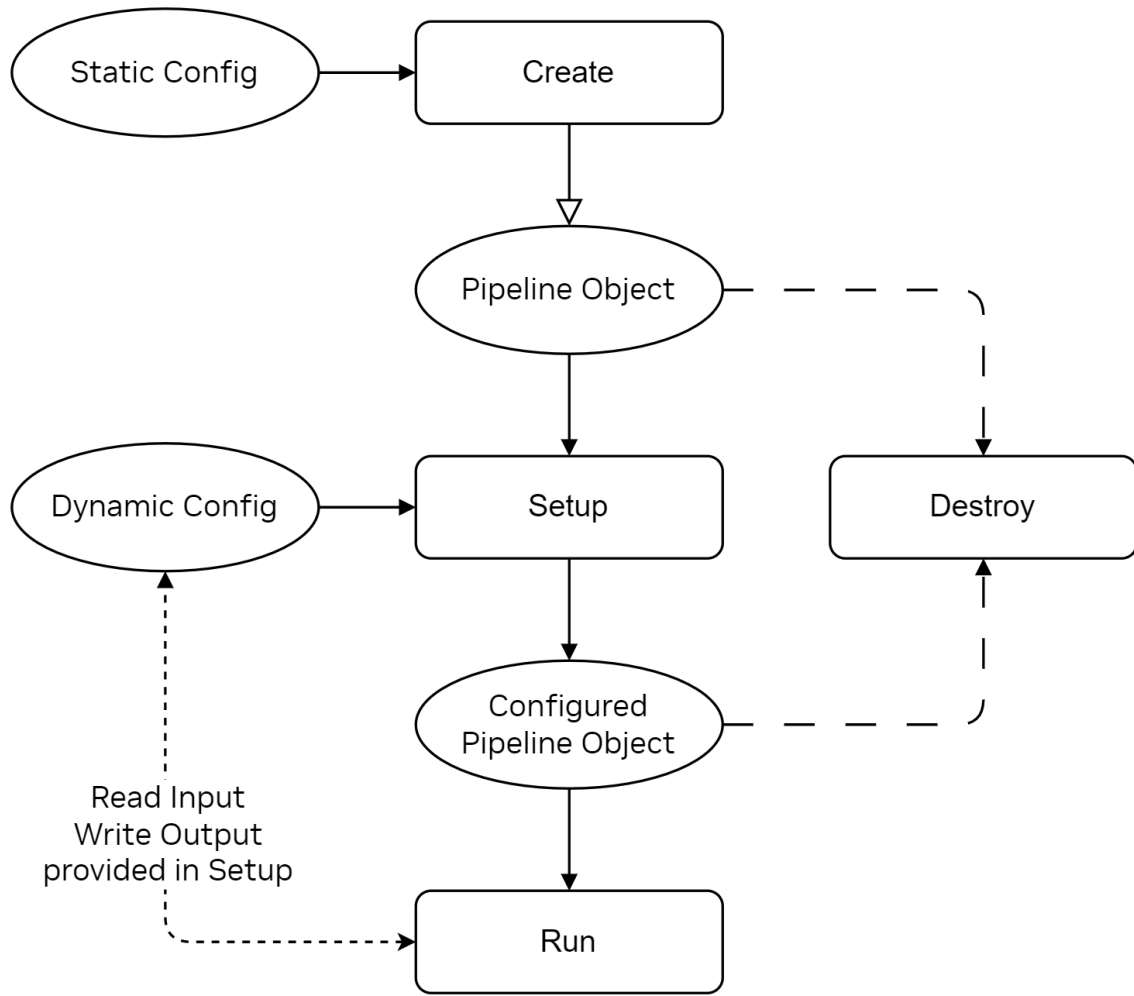
The SRS Pipeline implements the signal reference symbol (SRS) channel estimation for cellular uplink transmissions. The module takes received IQ samples from the gNB O-RU antennas as input and outputs the estimated channel coefficients for each subcarrier and antenna port. The module supports different bandwidths, and transmission modes as specified by the 3GPP standards.

#### 1.7.2.5.8.1 SRS Pipeline Lifecycle

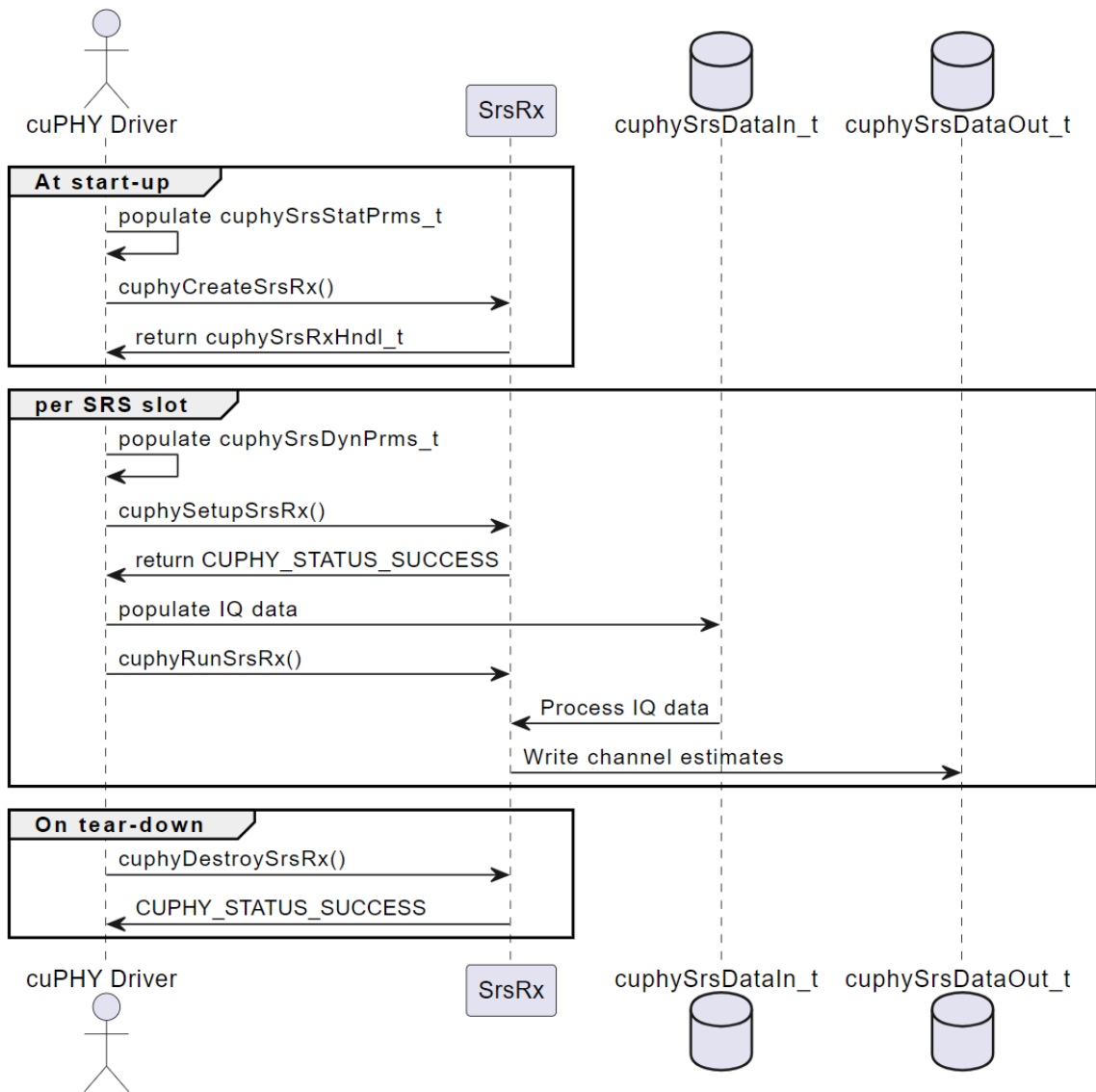
The SRS Pipeline module consists of a C++ class, `srsChEst` that encapsulates the main functionality and a C API that provides an interface for external applications. The C API consists of four functions: `cuphyCreateSrsRx()`, `cuphySetupSrsRx()`, `cuphyRunSrsRx()`, and `cuphyDestroySrsRx()`. Each of these functions corresponds to a phase in the pipeline lifecycle responsible for creating, configuring, running, and destroying the SRS Pipeline instance respectively.

#### 1.7.2.5.8.2 SRS Pipeline Execution

The SRS pipeline supports graph execution, however the graph simply consists of a single node for the channel estimation kernel.





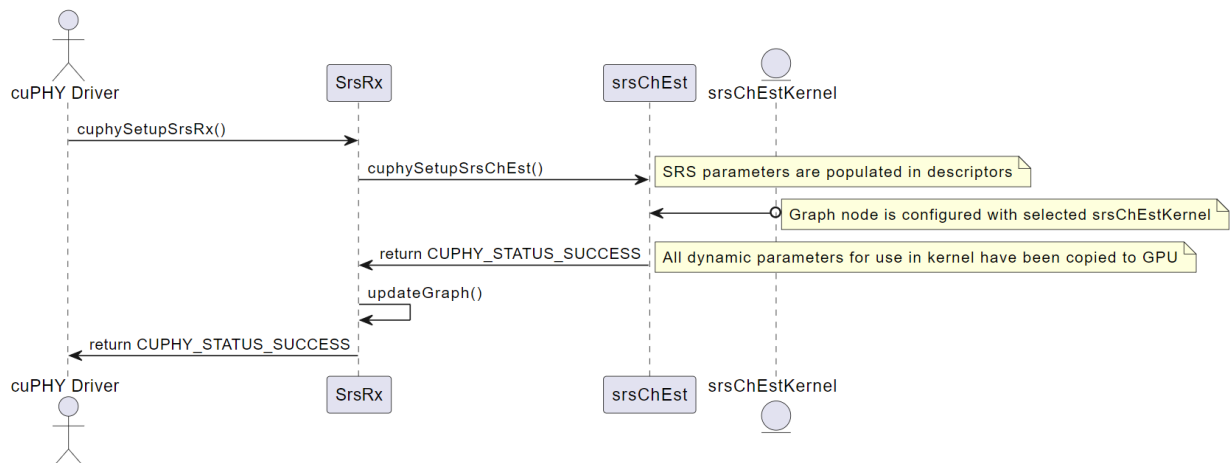


### 1.7.2.5.8.3 cuphyCreateSrsRx()

This function creates an instance of the SRS Pipeline and initializes its internal parameters and memory. The function takes a pointer to a structure of type **cuphySrsStatPrms\_t** as input, which contains configuration parameters for the SRS Pipeline that are expected to be constant for the pipeline's existence and that determine upper bounds for memory sizing, such as the number of antennas, and subcarriers spacing. The function returns a handle to the pipeline object, **cuphySrsRxHndl\_t**, which represents the SRS Pipeline instance and holds its state information. The function also performs some sanity checks on the input parameters.

### 1.7.2.5.8.4 cuphySetupSrsRx()

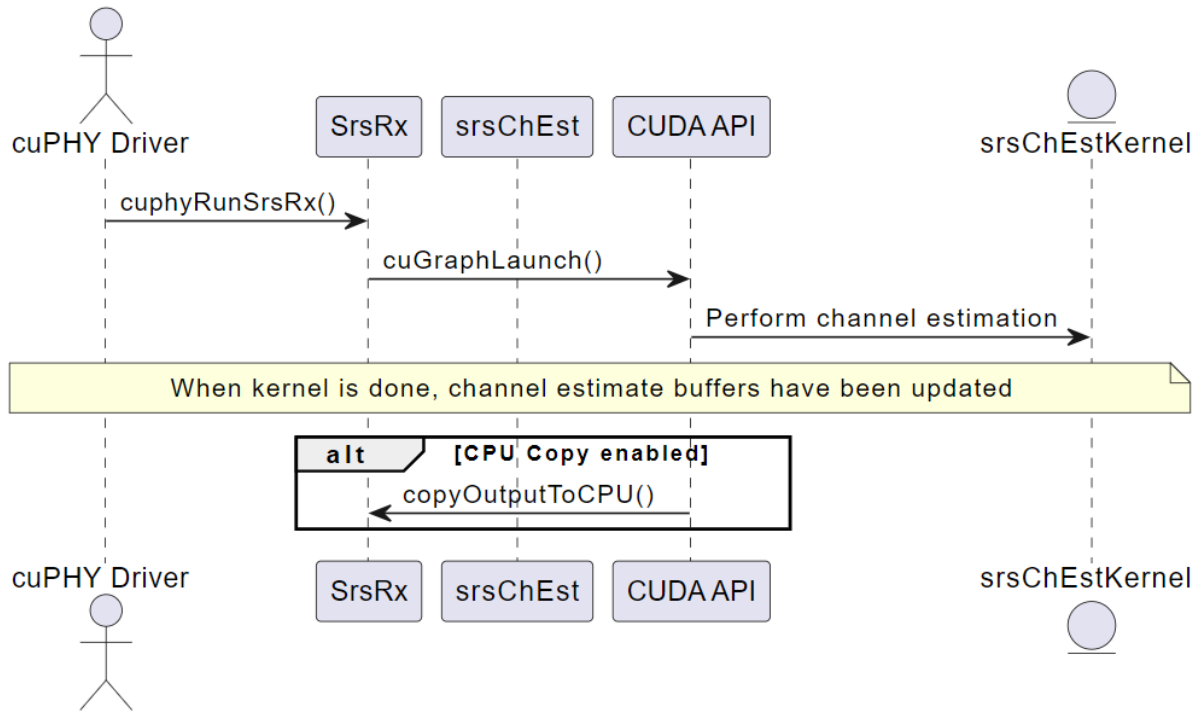
This function configures the SRS Pipeline instance with the specific parameters for each transmission. The function takes a pointer to the SRS Pipeline instance and a pointer to a structure of type **cuphySrsDynPrms\_t** as input. The structure contains the dynamic parameters for the SRS Pipeline, such as the PRBs in use, SRS configuration index, and SRS hopping bandwidth. The function populates descriptors for use by the SRS channel estimation kernel from the input data structure and sets up pointers for input and output data to prepare for processing. The function returns a status code indicating whether the operation was successful or not.



### 1.7.2.5.8.5 cuphyRunSrsRx()

This function runs the SRS Pipeline processing on a given set of IQ samples. The function takes a pointer to the SRS Pipeline instance. Input data locations were configured during the setup process described in **cuphySrsDataIn\_t** as part of the **cuphySrsDynPrms\_t** structure. This function will launch a kernel configured to read from those locations to perform channel estimation, including: - Extract the SRS symbols from the frequency domain samples based on the SRS configuration index and hopping bandwidth - Applies phase rotation and scaling to the SRS symbols to compensate for the channel effect - Estimates the channel coefficients for each subcarrier and antenna port using the SRS symbols and the known SRS sequences - Averages the channel estimates over multiple SRS symbols to reduce the noise

The kernel will output the channel estimates according to the **cuphySrsDataOut\_t** structure provided as part of **cuphySrsDynPrms\_t** in the setup process. The run function also returns a status code indicating whether the processing was successful or not.



### 1.7.2.5.8.6 cuphyDestroySrsRx()

This function destroys the SRS Pipeline instance and frees its resources. The function takes a pointer to the SRS Pipeline instance as input and releases its memory. The function returns a status code indicating whether the operation was successful or not.

### 1.7.2.5.8.7 Input and Output Data

Input Buffer	SrsRx:: m_hPrmDataRx[i].pTDataRx
Data type	array of per-cell tensors of IQ samples of element type CUPHY_C_16_F
Dimensions	[(ORAN_MAX_PRB*CUPHY_N_TONES_PER_PRB), OFDM_SYMBOLS_PER_SLOT, MAX_AP_PER_SLOT]
Output Buffer	SrsRx::m_outputPrms.h_chEstBuffInfo
Data type	cuphySrsChEstBuffInfo_t*
Dimensions	Array of per-user structures containing a tensor of [nPrbGrpEsts, nGnbAnts, nUeAnts] each element being CUPHY_C_16_F a scalar indicating start PRB group & PRB group size
Description	Buffer describing the channel estimate results from SRS

continues on next page

Table 45 – continued from previous page

Output Buffer	SrsRx::m_outputPrms.h_srsReports
Data type	cuphySrsReport_t*
Dimensions	Array of per-user structures
Description	Structure contains per-user estimates including timing, signal, and noise estimates
Output Buffer	SrsRx::m_outputPrms.h_rbSnrBuffer
Data type	Floating point array SINRs
Dimensions	[m_nPrbs* m_nSrsUes]
Description	Array containing per-RB SNR estimates
Output Buffer	SrsRx::m_outputPrms.h_rbSnrBuffOffsets
Data type	Array of 32-bit unsigned integers
Dimensions	[m_nSrsUes]
Description	Single dimensional array containing per-user offset into h_rbSnrBuffer
Output Buffer	SrsRx::m_outputPrms.h_srsChEstToL2
Data type	cuphySrsChEstToL2_t*
Dimensions	Array of pointers to per-user buffers. Each buffer is of dimension [nPrbGrpEsts, nGnbAnts, nUeAnts] with each element being represented as float2 a scalar indicating start PRB group & PRB group size
Description	This and h_chEstBuffInfo above describe the same channel estimates but this one is in CPU memory using complex FP32 and the other is in GPU memory using FP16.

#### 1.7.2.5.8.8 Memory Management

The SRS Pipeline uses different kinds of memory for its operation, and the caller is responsible for allocating and freeing some of them. The following table summarizes the types of memory used by the pipeline, their ownership, lifetime, and location.

Memory Type	Ownership	Lifetime	Location	Description
Pipeline Working	Pipeline	Allocated during <code>cuphyCreateSrsRx()</code> and freed during <code>cuphyDestroySrsRx()</code>	CPU & GPU	Memory used by pipeline for its internal processing, such as intermediate buffers, coefficients, etc.
<code>cuphySrsStatPrms_t</code>	Caller	Only valid during <code>cuphyCreateSrsRx()</code>	CPU	Memory used to store the static parameters of the pipeline, such as number of antennas, channels, etc.
<code>cuphySrsDynPrms_t</code>	Caller	Only valid during <code>cuphySetupSrsRx()</code>	CPU	Memory used to store the dynamic parameters of the pipeline, such as SRS bandwidth configuration, input data pointers, output buffer pointers, etc.
<code>cuphySrsDataIn_t</code>	Caller	Valid during <code>cuphyRunSrsRx()</code>	GPU	Memory used to store the input data for pipeline, such as IQ samples from the antennas
<code>cuphySrsDataOut_t</code>	Caller	Valid after <code>cuphyRunSrsRx()</code>	GPU	Memory used to store the output data from the pipeline, such as channel estimates.

The caller should ensure that the memory allocated for the input and output data is sufficient for the pipeline's operation, and that the pointers are correctly set in the dynamic configuration parameters. The pipeline may not check the validity or size of the memory. It is assumed to be consistent with the static and dynamic parameters. The caller should also ensure that the memory is not modified by other processes while the pipeline is using it.

#### 1.7.2.5.8.9 SRS channel estimation algorithms

The current pipeline implementation performs MMSE channel estimation based on the received SRS signals. The channel estimation algorithm consists of the following steps:

- ▶ Load received SRS subcarriers, remove ZC cover-code and average repetitions
- ▶ Remove cyclic shifts and apply wide filter to estimate channel
- ▶ Estimate delay phase ramp
- ▶ Remove delay phase ramp from received signal by multiplying with a shift sequence
- ▶ Remove cyclic shifts and apply narrow filter to estimate channel
- ▶ Average estimates. Estimate energy and noise

- ▶ Calculate correlation w.r.t. cyclic shift in use and not in-use: sum over, PRB, antenna, cyclic shift

The pipeline saves the channel estimate, the signal energy, the noise variance, and the correlation values to the output buffers to be made available for use elsewhere.

### 1.7.2.5.9 Performance Optimization

The cuPHY library is designed to accelerate PHY layer functionality of commercial grade 5G gNB DU. Software optimizations ensure reduced latency and scalable performance with the increased number of cells. We can categorize them as:

- ▶ Use of CUDA Graphs: The cuPHY library makes use of CUDA graph feature to reduce kernel launch latency. The CUDA kernels implementing signal processing components within each cuPHY physical layer channel pipeline are represented as nodes in a CUDA graph and the inter-component dependencies as edges between nodes. Since graph creation is expensive, a base graph with the worst case topology is created during initialization of channel pipelines where there are several specializations of component kernels. When the channel is scheduled for a given slot only the necessary subset of graph nodes are updated and enabled.
- ▶ Use of MPS (Multi-Process Service): The cuPHY driver creates multiple MPS contexts, each with an upper limit to the maximum number of SMs (Streaming Multiprocessors) that can be used by kernels launched there. MPS contexts for control channels (e.g. PUCCH, PDCCH) usually have significantly lower SM limits compared to MPS contexts for shared channels due to the expected computation load. Each MPS context also has one or more CUDA streams associated with it, with potentially different CUDA stream priorities.
- ▶ Kernel fusion: the cuPHY implementation may fuse functionality from different processing steps into a single CUDA kernel for improved performance. For example, the rate matching, scrambling and modulation processing steps of the downlink shared channel are all performed in a single kernel. The motivation for these customizations is to reduce memory access latency and therefore improve performance. For example, assume that there are two kernels that are run in sequence. The first kernel makes a computation, writes the output to the global memory and the second kernel needs to read this output from the global memory to continue the computation. In this case, fusing these two kernels can reduce the number of accesses to the global memory, which has higher latency.
- ▶ Optimization of L1-L2 data flow: Data flow between the L2 and L1, and between the L1 and the FH are important for optimization of the latency. Data TB payloads for PDSCH channel need to be copied from L2 to L1 whenever a PDSCH channel is scheduled by the L2. The size of TBs increases with higher data throughput and the number of TBs also can also increase with the number of cells and the number of UEs scheduled on a given time slot. cuPHY library pipelines the TB H2D (host to device) copy to run in parallel with PDSCH channel setup processing. Such pipelining hides the TB H2D copy latency reducing overall PDSCH completion time.

### 1.7.2.6 Running cuPHY Examples

The cuPHY library includes example programs that can be used to test cuPHY channel pipelines and components. How to run cuPHY channel pipelines are explained in Aerial Release Guide Document in the section “Running the cuPHY Examples”. Please refer to the release guide on how to run the cuPHY channel pipelines. In running these examples, note that recent cuPHY implementation uses graphs mode to improve performance.

cuPHY library also includes examples for its components. Some examples are provided below.

#### **Uplink channel estimation**

```
cuPHY/build/examples/ch_est/cuphy_ex_ch_est -i ~/<tv_name>.h5
```

Sample test run:

```
cuPHY/build/examples/ch_est/cuphy_ex_ch_est -i
TVnr_7550_PUSCH_gNB_CUPHY_s0p0.h5

UE group 0: ChEst SNR: 138.507 dB
ChEst test vector TVnr_7550_PUSCH_gNB_CUPHY_s0p0.h5 PASSED
22:53:17.726075 datasets.cpp:974 WRN[90935 ] [CUPHY.PUSCH_RX] LDPC throughput mode
↳disabled
22:53:17.943272 cuphy.hpp:84 WRN[90935 ] [CUPHY.MEMFOOT]cuphyMemoryFootprint - GPU
↳allocation:
684.864 MiB for cuPHY PUSCH channel object (0x7ffc16f09f90).
22:53:17.943273 pusch_rx.cpp:1188 WRN[90935 ] [CUPHY.PUSCH_RX] PuschRx:
Running with eqCoeffAlgo 3
```

### Simplex decoder

```
cuPHY/build/examples/simplex_decoder/cuphy_ex_simplex_decoder -i ~/<tv_name>.h5
```

Sample test run:

```
cuPHY/build/examples/simplex_decoder/cuphy_ex_simplex_decoder -i
TVnr_61123_SIMPLEX_gNB_CUPHY_s0p0.h5
AERIAL_LOG_PATH unset
Using default log path
Log file set to /tmp/simplex_decoder.log
22:57:29.115870 WRN 92956 0 [NVLOG.CPP] Using
/opt/nvidia/cuBB/cuPHY/nvlog/config/nvlog_config.yaml for nvlog configuration
22:57:33.455795 WRN 92956 0 [CUPHY.PUSCH_RX] Simplex code: found 0 mismatches out of
↳1 codeblocks

Exiting bg_fmtlog_collector - log queue ever was full: 0
```

### PUSCH de-rate match

```
cuPHY/build/examples/pusch_rateMatch/cuphy_ex_rateMatch -i ~/<tv_name>.h5
```

Sample test run:

```
cuPHY/build/examples/pusch_rateMatch/cuphy_ex_pusch_rateMatch -i
TVnr_7143_PUSCH_gNB_CUPHY_s0p0.h5

AERIAL_LOG_PATH unset
Using default log path
Log file set to /tmp/pusch_rateMatch.log
22:58:20.673934 WRN 93384 0 [NVLOG.CPP] Using cuPHY/nvlog/config/nvlog_config.yaml
for nvlog configuration
22:58:20.896254 WRN 93384 0 [CUPHY.PUSCH_RX] LDPC throughput mode disabled
nUes 1, nUeGrps 1
nMaxCbsPerTb 3 num_CBs 3
uciOnPuschFlag OFF
nMaxTbs 1 nMaxCbsPerTb 3 maxBytesRateMatch 156672
22:58:21.037299 WRN 93384 0 [CUPHY.MEMFOOT] cuphyMemoryFootprint - GPU
allocation: 684.864 MiB for cuPHY PUSCH channel object (0x7ffe23b0f690).
22:58:21.037302 WRN 93384 0 [CUPHY.PUSCH_RX] PuschRx: Running with eqCoeffAlgo 3
22:58:21.037810 WRN 93384 0 [CUPHY.PUSCH_RX] detected 0 mismatches out
```

(continues on next page)

(continued from previous page)

```
of 65280 rateMatchedLLRs
Exiting bg_fmtlog_collector - log queue ever was full: 0
```

### 1.7.3. Test MAC and RU Emulator Architecture Overview

TestMAC and RU emulator are the tools that are used by developers to test the system in a controlled environment. TestMAC functions as the L2/L1 interface, which schedules packets according to a predefined launch pattern. RU emulator is a basic implementation of ORAN FH interface. Its functions include verifying the timing of FH packets, checking the integrity of DL IQ samples and scheduling the transmission of UL IQ samples.

Functional blocks of TestMAC are displayed in the following figure. TestMAC is responsible for scheduling DL packets and validating received UL messages. TestMAC uses a predefined launch pattern for scheduling. The launch pattern defines the TDD pattern across multiple frames and the test vectors (TVs) used on each slot. The test vectors contain the L1 configuration for each PHY channel in a given slot. TestMAC obtains the slot timing from L1 via L2 adapter. The timing is indicated by the slot indication message. TestMAC prepares the FAPI message according to the L1 configuration contained in the TV. If a given slot is an UL, TestMAC parses the corresponding TV and compares the received data with the expected values included in the TV.

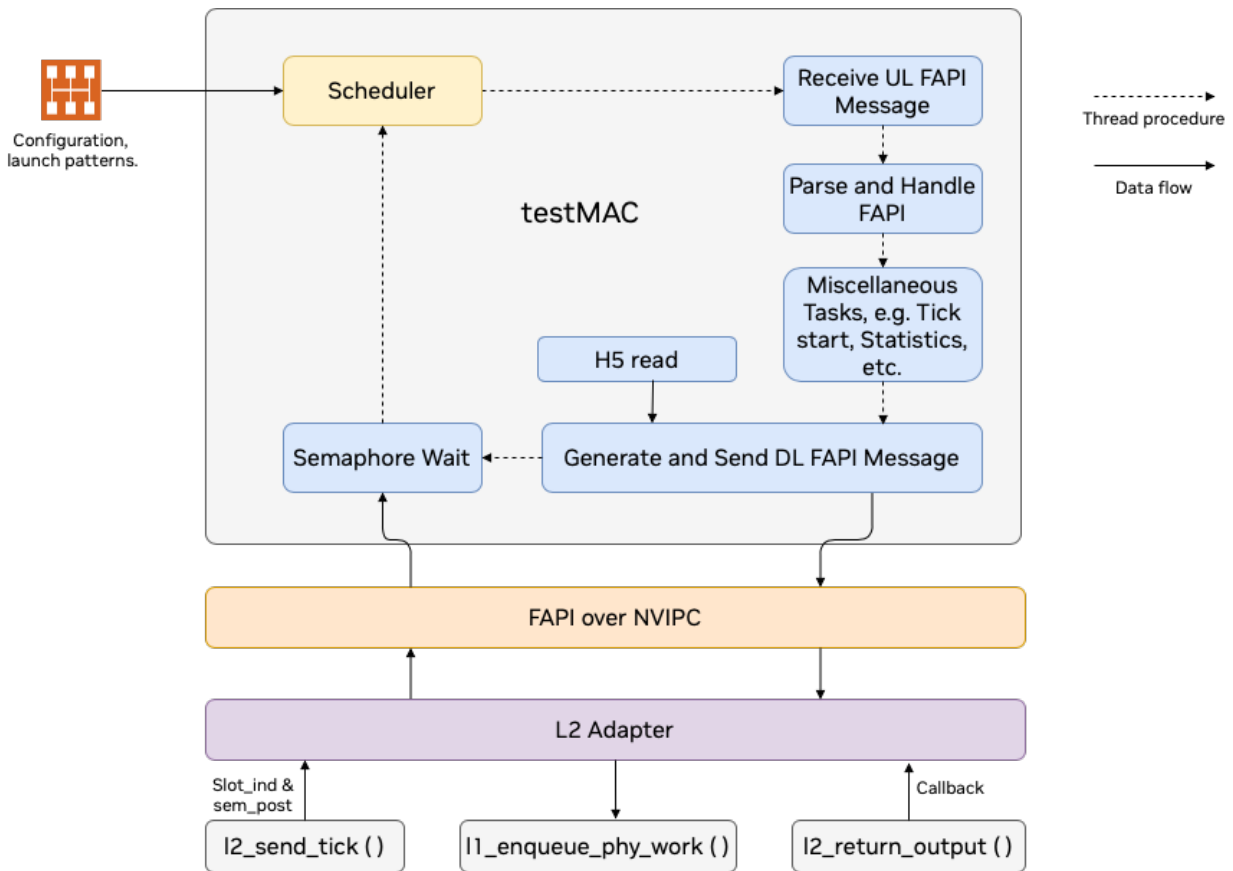


Fig. 13: Test MAC functionality

The RU emulator has the following functions:



- ▶ Validation of timing of the transmitted packets by the DU (DL u-plane, DL c-plane, UL c-plane)
- ▶ Validation of the transmitted IQ samples or DL u-plane payload data
- ▶ Transmission of UL u-plane packets as a response to UL c-plane messages

The logic used by RU emulator to process received packets is displayed in the following figure. If the received packet is a U-plane, RU emulator will continue parsing the packet header to retrieve eAxC id, frame number, subframe number, slot id, startSym index, number of symbols, start PRB index and number of PRBs. It then compares the payload with the corresponding data included in the TV. If the received packet is a C-plane message for an UL packet, they are again parsed to extract the information for the UL data allocation same as for DL packets. RU emulator then transmits the UL u-plane data symbol by symbol and it uses accurate send scheduling function.

RU emulator needs the cuphycontroller configuration to obtain PCI address of the NIC interface, MAC address of the peer system, cell configurations, VLAN ID and eAxCid values for each cell. It also uses launch pattern file to understand the TDD pattern and the L1 configuration for each slot.

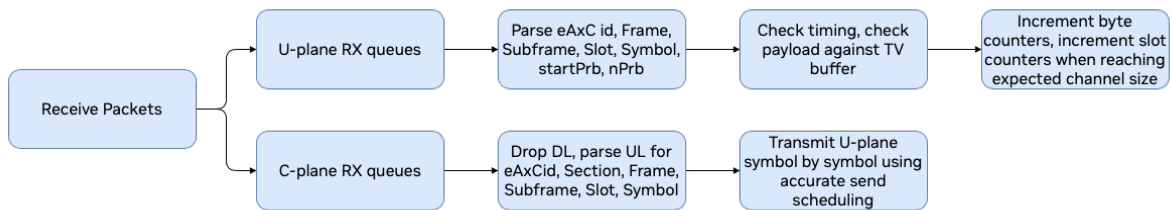


Fig. 14: RU Emulator received packet processing

## 1.7.4. 5G MATLAB Models for Testing and Validation

Aerial CUDA-Accelerated RAN includes a simulation model called `nr_sim` that is written in Matlab matching with the CUDA implementation in `cuPHY` library. It can be found under `$cuBB_SDK/5GModel/nr_matlab`. It serves as a reference model for Aerial design and verification, which covers from L1/L2 FAPI interface to O-DU/O-RU FH interface.

A high level function block diagram of the `nr_sim` is shown in the following figure. The core of `nrSim` is the simulation engine `nrSimulator.m`, which includes Matlab models for gNB transmitter and receiver, MIMO fading channel and UE transmitter. `nrSimulator.m` can be called by `runSim.m` with external configuration mode or by `runRegression.m` with internal configuration mode.

The simulator provides three major features: waveform compliance test, test vector generation and PHY performance simulation.

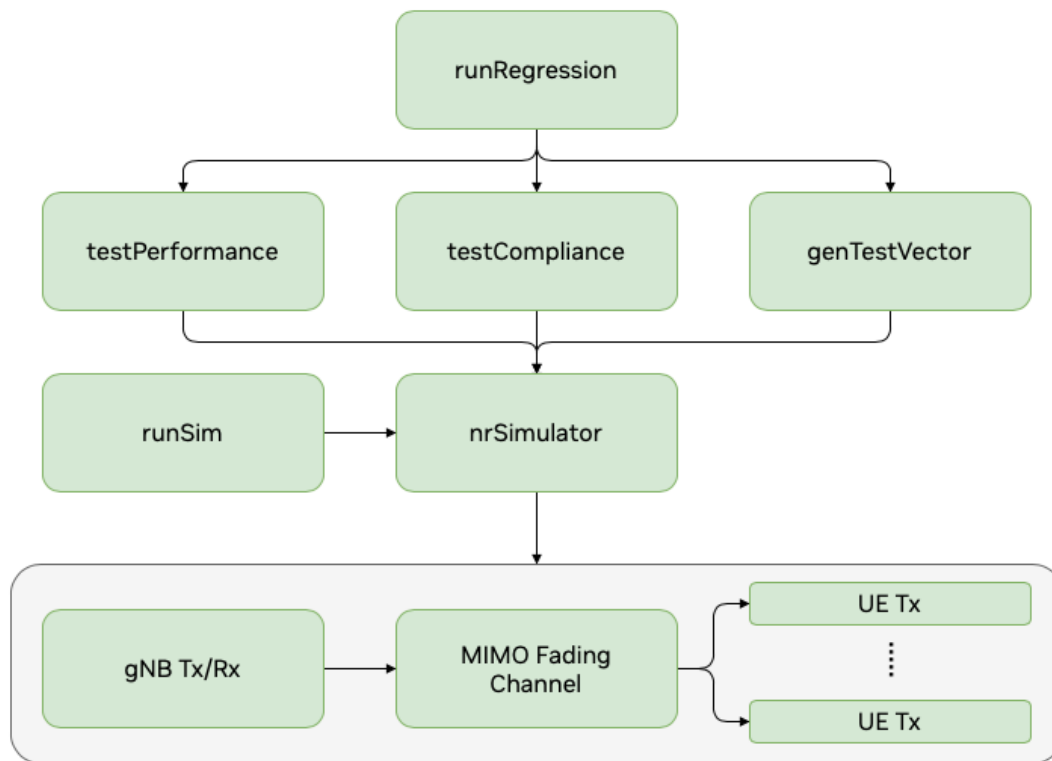


Fig. 15: nr\_sim functionality

### 1.7.4.1 Waveform compliance test

The purpose of waveform compliance test is to make sure our understanding of 3GPP standards regarding signal waveform generation is correct. It is achieved by checking nrSim generated signal against Matlab 5G Toolbox generated signal.

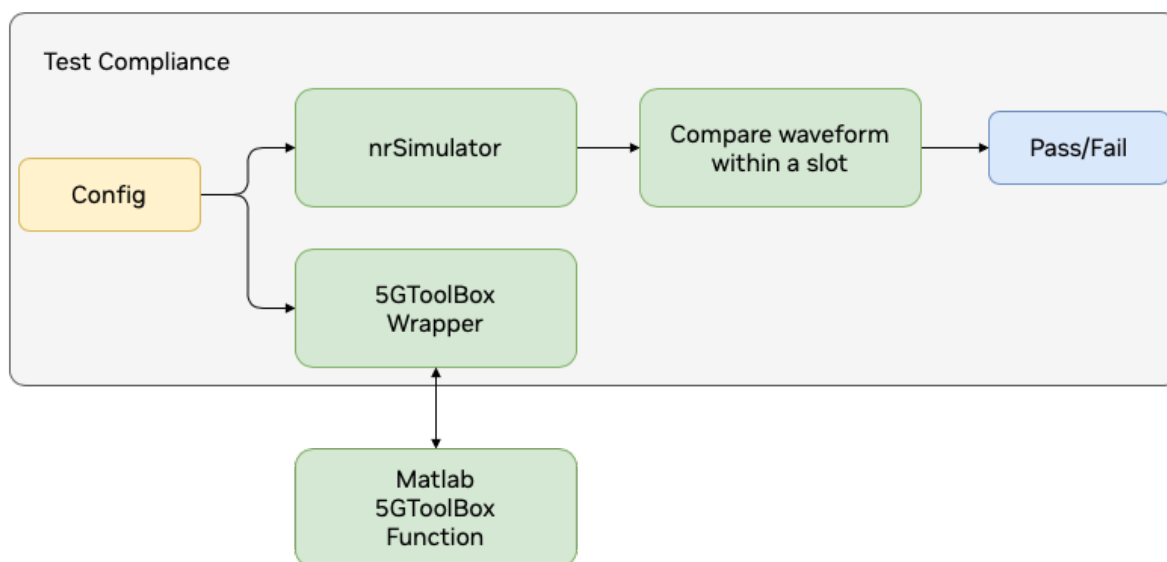


Fig. 16: Waveform compliance test

### 1.7.4.2 Test Vector Generation

nrSim can generate test vectors for L2/L1 FAPI PDU, cuPHY channel pipeline API parameters, cuPHY channel pipeline output and the compressed samples in a slot.

Two types of test vectors will be generated for each test case configuration.

- ▶ FAPI test vector including FAPI PDU for all the channels in this slot and FH compressed samples for this slot. There is only one FAPI TV per slot.
- ▶ cuPHY test vector including cuPHY parameters and input/output for a cuPHY channel pipeline call. There can be multiple cuPHY TVs per slot.

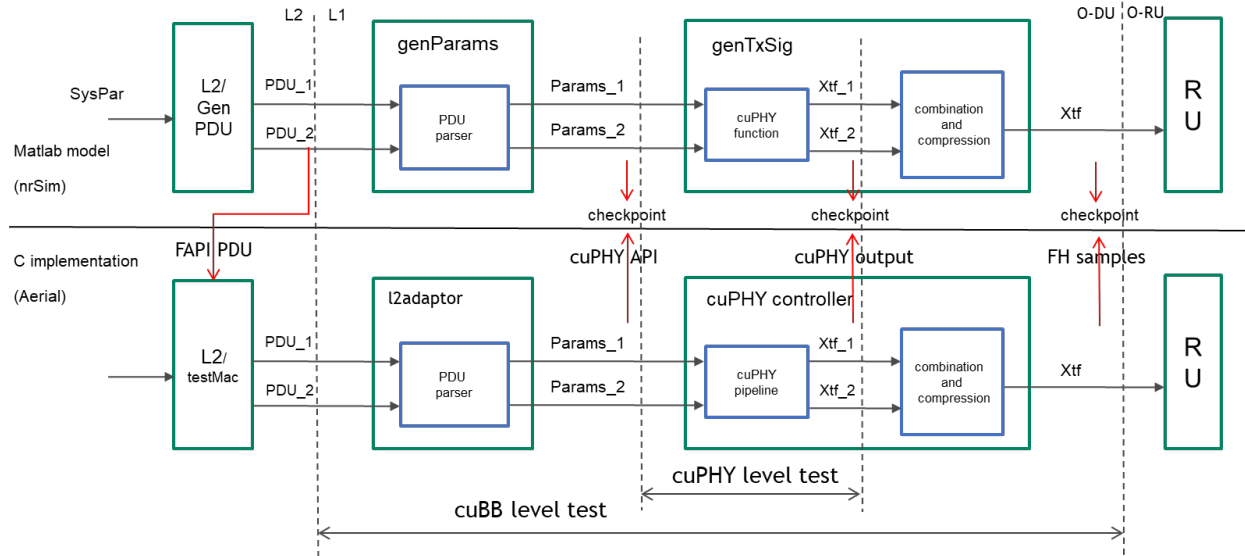


Fig. 17: Test vector generation

### 1.7.4.3 PHY Performance Simulation

The purpose of this test is to make sure that Aerial PHY performance can meet 3GPP requirement by checking nrSim performance simulation results with the same channel condition and test configuration specified by the 3GPP standard.

### 1.7.4.4 nrSim Configuration

The input to the simulation engine nrSimulator.m is a single data structure SysPar, which includes all the 3GPP related configurations and simulation control related configurations. The outputs of nrSimulator include SysPar, UE (array of structures for all UEs) and gNB (structure for gNB).

[SysPar, UE, gNB] = nrSimulator(SysPar)

Matlab functions listed in the table below generate the default configuration for the parameters in SysPar.

Data Structure	Field	Description	Matlab function for default configuration
SysPar	testAlloc	Specify DL/UL direction and the number of each type of channels allocated for the slot	initSysPar
	carrier	Specify carrier level configuration	cfgCarrier
	ssb	Specify SSB configuration	cfgSsb
	pdccch	Specify PDCCH channel configuration	cfgPdcch
	pdsch	Specify PDSCH channel configuration	cfgPdsch
	csirs	Specify CSIRS channel configuration	cfgCsirs

continues on next page

Table 47 – continued from previous page

Data Structure	Field	Description	Matlab function for default configuration
	prach	Specify PRACH channel configuration	cfgPrach
	pucch	Specify PUCCH channel configuration	cfgPucch
	pusch	Specify PUSCH channel configuration	cfgPusch
	srs	Specify SRS channel configuration	cfgSrs
	Chan	Specify MIMO fading channel configuration	cfgChan
	SimCtrl	Specify Simulation control parameters	cfgSimCtrl

Configuration options for the testAlloc is summarized in the table below. DL and UL fields indicate if the test is for a DL or an UL slot. The remaining fields hold the number of PHY channel allocations for the test. A given test can include multiple combinations of PHY channels, i.e. 1 SSB allocation, 4 PDCCH allocations, 4 PDSCH allocations, etc.

Data structure	Field	Description
testAlloc	DL	Enable DL test
	UL	Enable UL test
	Ssb	Enable SSB allocation
	Pdcch	Number of PDCCH channels in a slot
	Pdsch	Number of PDSCH channels in a slot
	Csirs	Number of CSIRS channels in a slot
	Prach	Number of PRACH channels in a slot
	Pucch	Number of PUCCH channels in a slot
	Pusch	Number of PUSCH channels in a slot
	Srs	Number of SRS channels in a slot

SysPar definition for 3GPP carrier and slot configuration with each channel is mostly based on SCF-FAPI specification.

The Chan configuration refers to MIMO fading channel model.

Data structure	Field	Description
Chan	Type	AWGN, TDLx-xx-xxx (3GPP MIMO fading channel)
	SNR	Channel SNR in dB
	Delay	Channel propagation delay in second
	CFO	Carrier frequency offset in Hz
	Use5Gtoolbox	Reserved
	gain	Reserved

The SimCtrl structure includes global configuration settings that are used in the simulation.

Datastructure	Field	Sub-field	Description	
SimCtrl	N_UE		Number of UEs	
	N_frame		Number of frames per run	
	N_slot_run		Number of slots in a frame to run. (0: run all slots in a frame)	
	timeDomainSim		Enable time domain simulation (required for applying fading channel model, delay and CFO)	
	plotFigure	tfGrid		Plot time/freq domain signal
		constellation		Plot constellation before and after equalizer
	genTV	Enable		Enable TV generation at gNB side
		enableUE		Enable TV generation at UE side
		tvDirName		Name for TV directory
		cuPHY		Enable cuPHY TV in h5 format
		FAP		Enable FAP TV in h5 format
		FAPyaml		Enable FAP TV in yaml format
		slotIdx		Indices of slots on which TV will be generated
		forceSlotIdxFlag		Force slot index = slotIdx(1) for every slot
		bypassComp		Bypass FH sample compression
		idx		Reserved
	TVname		Prefix for the name of TVs	

continues on next page

Table 48 – continued from previous page

Datastructure	Field	Sub-field	Description
		fp16AlgoSel	0: Use half function (Matlab fixed point toolbox required) 1: Use vpf16 function (Matlab fixed point toolbox not required)
		CFOflag	Enable CFO correction
		enableRssiMeas	Enable RSSI measurement
	capSamp		Reserved
	result		Reserved

#### 1.7.4.5 nrSim Usage

For different test and simulation purpose, nrSim provides two modes to change the configurations and run the Matlab model.

- ▶ External configuration mode (runSim): This mode is to use an external configuration file in yaml format to update the parameters. nrSim will read this yaml configuration file and set the SysPar parameters accordingly. It is recommended that non matlab developer uses this mode to generate test vectors which requires no change to the Matlab code.
- ▶ Internal configuration mode (runRegression): This mode is to change the SysPar parameters directly in the Matlab code between initSysPar and nrSimulator. Matlab developer can pre-define a set of configuration used by compliance test, test vector generation and PHY performance simulation. Multiple runs can be performed in this mode with different configurations.

#### 1.7.4.6 Matlab Environment Preparation

Matlab version:

- ▶ R2020a or later

Matlab licenses:

- ▶ MATLAB
- ▶ Communications Toolbox
- ▶ DSP System Toolbox
- ▶ Signal Processing Toolbox
- ▶ Fixed-Point Designer (optional)
  - ▶ Call half function to accelerate testing/simulation
  - ▶ Can be disabled by setting SimCtrl.fp16AlgoSel = 1
- ▶ Parallel Computing Toolbox (optional)
  - ▶ Accelerate testing/simulation automatically

- ▶ 5G Toolbox (optional)
  - ▶ Not required for TV generation
  - ▶ Required for waveform compliance test and performance simulation

Preparation:

- ▶ After download the source code, launch Matlab on the directory of nr\_matlab and run startup to add all sub-directories into Matlab search path.

#### 1.7.4.7 External Configuration Mode (runSim)

- 1) Find the yaml configuration template file `cfg_template.yaml` under `nr_matlab`. If it is missing, run `genCfgTemplate` to generate it.
- 2) Use a text editor to change the parameters in the yaml file. Basically `cfg_template.yaml` is a yaml (text) version of `SysPar` data structure. Please refer to section 3 for the description of `SysPar` parameters. After change is done, save it to another file name, for example, `cfg_test.yaml`.
- 3) Run `runSim(cfg_filename, tv_filename)`, for example, `runSim(cfg_test.yaml, my_test)`. `nrSim` will read `cfg_test.yaml` file, update `SysPar` accordingly, run `nrSimulator` and generate test vector files with name starting with `my_test`. The generated TV files are stored under the folder named by `SysPar.SimCtrl.tvDirName`, for example, `GPU_test_input`.
- 4) Another option is to use `runSim(cfg_filename, test, tv_filename)`,

Notes:

- ▶ This mode only supports test vector generation with `SimCtrl.genTV.enable` set to 1. It does not support waveform compliance test and PHY performance test.
- ▶ If `SimCtrl.plotFigure.tfGrid` is set to 1, the time/freq signal in a frame or the specified number of slots in a frame (controlled by `N_slot_run`) can be plotted to provide visualized channel allocations.
- ▶ Non Matlab developer can write script in any language to modify the yaml template file and automatically generate a number of different yaml configuration files for different testing purpose.

#### 1.7.4.8 Internal Configuration Mode (runRegression)

Instead of updating configuration through the external YAML configuration file case by case, the internal configuration mode changes `SysPar` parameters directly inside the Matlab code, which allows Matlab developers to define and execute a batch of test cases more efficiently. The main function for this mode is `runRegression`, which supports a flexible combination of `testSet`, `channelSet` and `caseSet` as the input arguments.

`runRegression(testSet, channelSet, caseSet)`

Argument	Values	Value Selection
<code>testSet</code>	<code>Compliance, TestVector, Performance, allTests</code>	Multiple
<code>channelSet</code>	<code>ssb, pdcch, pdsch, csirs, dlmix, allDL, prach, pucch, pusch, srs, ulmix, allUL, allChannels</code>	Multiple
<code>caseSet</code>	<code>full, compact, selected</code>	Single



Here are some example commands.

- ▶ Full regression test for all channels  
`runRegression({'allTests'}, {'allChannels'}, 'full')`
- ▶ Waveform compliance test and test vector generation for pdcch and pdsch channels with compact set  
`runRegression({'Compliance', 'TestVector'}, {'pdcch', 'pdsch'}, 'compact')`
- ▶ PHY performance simulation for PRACH channel

`runRegression({'Performance'}, {'prach'}, 'full')`

The test cases for each channel are defined in the Matlab file `testCompGenTV_xxxx.m`, where `xxxx` is the channel name. Matlab developer can modify the Matlab file to create and assign test cases for full set, compact set and selected set.

- ▶ full set includes all the test cases which can be generated by `nrSim` and pass waveform compliance test against 5G Toolbox.
- ▶ compact set includes a subset of full set test cases which are supported by `cuPHY` implementation. TVs from Compact set can be used for nightly CICD regression test.
- ▶ selected set includes a subset of compact set test cases which are essential for `cuPHY` verification. TVs from Selected set can be used for merge request (MR) CICD regression test.

Notes:

- ▶ `testCompGenTV_dlmix` and `testCompGenTV_ulmix` supports multi-channel multi-slot TV generation without waveform compliance check.
- ▶ `testPerformance_prach`, `testPerformance_pusch` and `testPerformance_pucch` support PHY performance test for PRACH (format 0/B4), PUSCH (non-UCI) and PUCCH (format 0/1).

Below is an example of full regression test summary with Matlab command

`runRegression({'allTests'}, {'allChannels'}, 'full')`

Channel	Compliance_Test	Error	Test_Vector	Error	Performance_Test	Fail
SSB	15	0	15	0	0	0
PDCCH	47	0	47	0	0	0
PDSCH	222	0	222	0	0	0
CSIRS	55	0	55	0	0	0
DLMIX	0	0	16	0	0	0
PRACH	20	0	20	0	48	0
PUCCH	110	0	110	0	60	0
PUSCH	199	0	199	0	32	0
SRS	2	0	2	0	0	0
ULMIX	0	0	6	0	0	0
<b>Total</b>	<b>670</b>	<b>0</b>	<b>692</b>	<b>0</b>	<b>140</b>	<b>0</b>

Elapsed time is 1221.657852 seconds.

Fig. 18: An example output of a full regression test summary

## 1.7.5. AI/ML Components for PUSCH Channel Estimation

PUSCH Channel Estimation can load the TrTEngine model and run it as part of the PUSCH-RX pipeline. This channel estimation mode can work for 1UE and 1 Cell.

### 1.7.5.1 How to Enable and Run the TrTEngine Model

Follow these steps to enable the TrTEngine model as part of the PUSCH-RX pipeline:

1. Enable TrTEng in the `cuphycontroller` top-level YAML. For example, you use the following setting in `cuphycontroller_nrSim_SCF_CG1.yaml`:

```
puschrx_chest_factory_settings_filename: /opt/nvidia/cuBB/cuPHY/examples/ch_
↪est/chest_trt.yaml
```

2. Change the `testMAC` validation value to 1 in `test_mac.yaml`:

```
validate_enable: 1
```

### 1.7.5.2 TrTEngine YAML File Example

The following is an example of the TrTEngine YAML file. The names of the input/output tensors might differ depending on the AI model. Depending on the chosen location, the path and engine filename might also be different.

The dimensions of the input and outputs are shown in the YAML file below.

```
---
chest_factory: trtengine

file: /opt/nvidia/cuBB/cuPHY/examples/ch_est/channel_estimator_fp16-True_tf32-True.
↪engine

max_batch_size: 1

inputs:
- name: z
  dataType: 0 # CUPHY_R_32F
  dims: [1638, 4, 4, 2, 2] # subcarriers, layers, antennas, symbols, real&imag

outputs:
- name: zout
  dataType: 0 # CUPHY_R_32F
  dims: [4, 4, 3276, 2, 2] # antennas, layers, subcarriers, symbols, real&imag
...
```

### 1.7.5.3 Example Python Script

The `torch_to_trt_chest_example.py` example script takes a PyAerial-based model and compiles it into a TrTEngine that can run in cuPHY.

#### 1.7.5.3.1 Prerequisites

Before using this script, ensure the model to be compiled accepts inputs from the PyAerial Least Squares Channel Estimator (`algo=3`) and outputs an estimate with the same shape as the PyAerial MMSE Estimator (`algo=1`).

- ▶ **INPUT:** [batch, subcarriers, layers, rx antennas, symbols, 2]
- ▶ **OUTPUT:** [batch, rx antennas, layers, subcarriers, symbols, 2]

#### Note

The output should have 2x more subcarriers than the input.

#### 1.7.5.3.2 How to Execute the Script

1. Copy the model definition in PyTorch or TensorFlow to this script. This example uses a PyTorch model; refer to the **Notes** section below for more information on using TensorFlow.
2. Load the trained model weights.
3. Select the desired input type:
  - ▶ “<use\_tvs> = True”: Test Vectors for cuPHY validation
  - ▶ “<use\_tvs> = False”: A channel generated with Sionna and estimated with PyAerial
  - ▶ Optionally, any other input and output can be saved and used with the model as long as the dimensions match the input and output.
4. Provide the model as an argument to the `check_results()` function.

#### Note

This function should not be modified because it emulates the steps in cuPHY.

5. Compare the MSE of the model provided with the MSE of LS. This should give an indication of whether the model is working properly. The results for the example model (depending on the input type) are the following:
  - a. Using TVs, the MSEs obtained for LS and the model are as follows:
    - ▶ LS = -7.6 dB; ML: -14.1 dB (tv = 7201)
    - ▶ LS = -7.6 dB; ML: -13.4 dB (tv = 7217)
  - b. Using Sionna channels, the MSEs obtained are as follows:
    - ▶ LS = -20.0 dB; ML = -24.8 dB

6. Later in the script, the model should be exported to ONNX and evaluated again. Results should match the previous numbers.
7. The model is compiled to TrT using polygraphy. The results should again match the ones previously obtained.

### 1.7.5.3.3 Notes

- ▶ This script can be executed end-to-end without modification, and the provided values should appear in the respective steps.
- ▶ Any pre- or post-processing applied to the input data should be included inside the model. This limits the operations allowed in the model.
  - ▶ Refer to the [PyTorch ONNX documentation](#) for supported operations when exporting to ONNX with PyTorch.
  - ▶ Refer to the [TensorFlow ONNX documentation](#) for supported operations when exporting to ONNX with TensorFlow.
    - ▶ Note further that “The current implementation only works for graphs that do not contain any control flow or embedding related ops,” as described in the TensorFlow GitHub documentation.
  - ▶ Refer to the [ONNX-TensorRT documentation](#) for supported operations when compiling to TRT.
  - ▶ If the model cannot be adjusted to work without a given forbidden operation, then a possible workaround is to use a plugin (e.g. a [TensorRT DFT plugin](#)).
- ▶ The batch dimension is used to stack the estimations of multiple users.
- ▶ If you’re using a TensorFlow model instead of PyTorch, you will need to determine how to export the model to ONNX. One option is `tf2onnx`, which is available in PyPI.
- ▶ MD5 sums:
  - ▶ `channel_estimator.onnx`: 64af9b805c9c7e7d831a93dbb4a646ad (repeatable)
  - ▶ `channel_estimator_fp16-True_tf32-True.engine`: 2170dd84c2e64470b3f221ca6a310ef3 (not repeatable)
- ▶ Dependencies information:
  - ▶ `mamba install numpy pytorch torchvision torchaudio pytorch-cuda=12.4 -c pytorch -c nvidia`
  - ▶ `pip install ipykernel polygraphy onnx nvidia-pyindex nvidia-tensorrt`
  - ▶ Ensure that the container version of TRT matches the Python version: `pip install tensorrt==10.3 onnx2torch`

## 1.7.6. References

- [1] 3GPP, “NR; Physical channels and modulation,” 3GPP TR 38.211, v15.4.0.
- [2] 3GPP, “NR; Multiplexing and channel coding,” 3GPP TR 38.212, v15.4.0.
- [3] 3GPP, “NR; Physical layer procedures for control,” 3GPP TR 38.213, v15.4.0.
- [4] 3GPP, “NR; Physical layer procedures for data,” 3GPP TR 38.214, v15.4.0.
- [5] 3GPP, “NR; Physical layer measurements,” 3GPP TR 38.215, v15.4.0.
- [6] Small cell forum, “SCF 222 5G FAPI PHY API,” v10.02, March 2020.
- [7] NVIDIA GPU Direct RDMA, <https://developer.nvidia.com/gpudirect>.
- [8] O-RAN Working Group 4 (Open Fronthaul Interfaces WG), Control, User and Synchronization Plane Specification, O-RAN.WG4.CUS.0-v07.02.
- [9] Jinghu Chen and M. P. C. Fossorier, “Near optimum universal belief propagation based decoding of low-density parity check codes,” in *IEEE Transactions on Communications*, vol. 50, no. 3, pp. 406-414, March 2002.
- [10] K. Chen, B. Li, H. Shen, J. Jin and D. Tse, “Reduce the Complexity of List Decoding of Polar Codes by Tree-Pruning,” in *IEEE Communications Letters*, vol. 20, no. 2, pp. 204-207, Feb. 2016.
- [11] G. Sarkis, P. Giard, A. Vardy, C. Thibault and W. J. Gross, “Fast List Decoders for Polar Codes,” in *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 2, pp. 318-328, Feb. 2016.

Aerial CUDA-Accelerated RAN is a set of software defined libraries that are optimized to run 5G gNB workloads on GPU. These libraries include cuPHY, cuMAC and pyAerial. In this section, we focus on layer-1 (L1), or physical (PHY) layer of 5G gNB software stack as defined by 3GPP [1-5].

cuPHY is the 5G L1 library of the Aerial CUDA-Accelerated RAN. It is designed as an inline accelerator to run on NVIDIA GPUs and it does not require any additional hardware accelerator. It is implemented according to the O-RAN 7.2 split option [8]. cuPHY library takes advantage of massively parallel GPU architecture to accelerate computationally heavy signal processing tasks. It also makes use of fast GPU I/O interface between the NVIDIA Bluefield-3 (BF3) NIC and GPU (GPU Direct RDMA [7]) to improve the latency.

BF3 NIC provides the fronthaul (FH) connectivity in addition to the IEEE 1588 compliant timing synchronization. The BF3 NIC also has a built-in SyncE and eCPRI windowing functionality, which meets G.8273.2 timing requirements.

In the following, we first give an overview of cuPHY library software stack. cuPHY library consists of L1 controller components running on the CPU and PHY layer functions running on the GPU. After providing the overview, we will go into details of each component and explain how L1 controller components interact with each other and L2. Finally, we will go over the PHY layer signal processing functions, which are accelerated as CUDA kernel implementations.

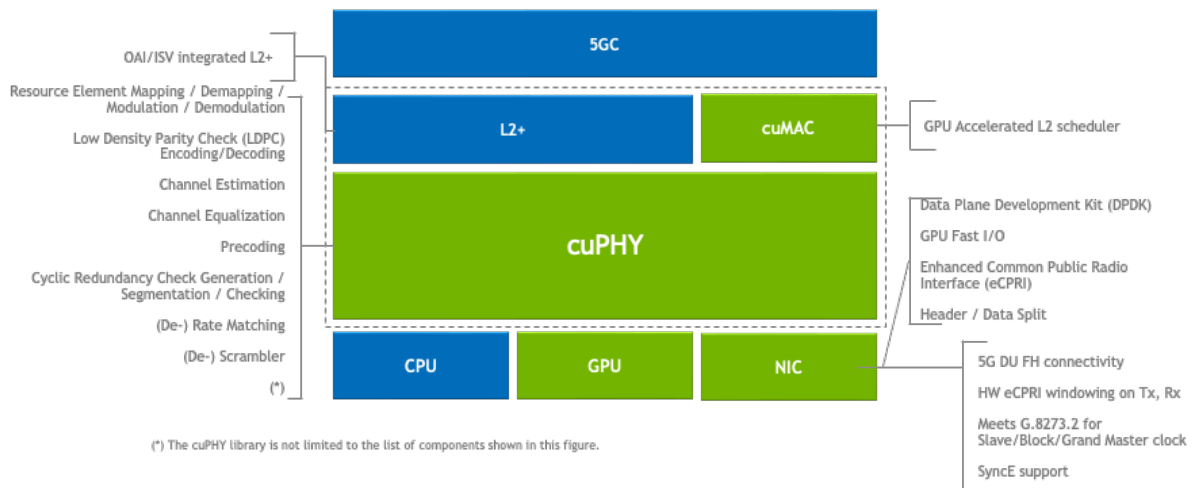


Fig. 19: Aerial CUDA-Accelerated Software Stack within 5G gNB DU

## 1.8. Glossary

Term or Abbreviation	Definition
3GPP	Third Generation Partnership Project
5G NR	Fifth generation new radio
Aerial	Software suite that accelerates 5G RAN functions with the GPU
CB	Code Block
CORESET	Control Resource Set
CSI	Channel State Information
CSI-RS	Channel State Information Reference Signal
cuBB	CUDA GPU software libraries/tools that accelerate 5G RAN compute-intensive processing
cuMAC	CUDA-based platform for accelerating 5G/6G MAC layer scheduler functions with NVIDIA GPUs
cuPHY	CUDA 5G PHY layer software library of the cuBB
cuPHY-CP	cuPHY control-plane software
CDM/FDM/TDM	Code-division multiplexing, Frequency Division Multiplexing, Time-Division Multiplexing
CMake	CMake is a software tool for configuring the makefiles for building the CUDA examples
CUDA	Compute Unified Device Architecture
CX6-DX	Mellanox ConnectX6-DX NIC

continues on next page

Table 50 – continued from previous page

Term or Abbreviation	Definition
DCI	Downlink Control Information
DL	Downlink
DMRS	Demodulation Reference Signal
DOCA	A software framework that helps developers create applications and services on top of the NVIDIA BlueField networking platform
DPDK	Data Plane Development Kit
DU or O-DU	O-RAN Distributed Unit (a logical node hosting RLC/MAC/High-PHY layers based on a lower layer functional split.)
eAxC	Extended Antenna Carrier: a data flow for a single antenna (or spatial stream) for a single carrier in a single sector
eCPRI	Enhanced Common Public Radio Interface
FAPI	Functional Application Programming Interface
FH	Fronthaul
GDR	GPUDirect RDMA
H2D	Host-to-device memory
HDF5	A data file format used for storing test vectors. The HDF5 software library provides functions for reading and writing the test vector files
LDPC	Low-Density Parity Check Code
MIB	Master Information Block
MU-MIMO	Multi-User Multiple Input - Multiple Output
NIC	Network interface card
O-RAN	Open Radio Access Network
PBCH	Physical Broadcast Channel
PDCCH	Physical Downlink Control Channel
PDSCH	Physical Downlink Shared Channel
PUCCH	Physical Uplink Control Channel
PRACH	Physical Random Access Channel
PRB	Physical Resource Block
PUSCH	Physical Uplink Shared Channel
RAN	Radio Access Network
RE	Resource Element
REG	Resource Element Group
RB	Resource Block

continues on next page

Table 50 – continued from previous page

Term or Abbreviation	Definition
RM	Reed-Muller
RU or O-RU	O-RAN Radio Unit: a logical node hosting Low-PHY layer and RF processing based on a lower layer functional split
SCF	Small Cell Forum
SIB/SIB1	System Information Block
SSB	Synchronization Signal Block
SU-MIMO	Single-User Multiple Input - Multiple Output
SyncE	Synchronous Ethernet: An ITU-T standard to provide a synchronization signal to network resources
TB	Transport Block
TTI	Transmission Time Interval
TV	Test Vector
UCI	Uplink Control Information
UE-EM	UE Emulator Test Equipment
UL	Uplink



---

# Chapter 2. Aerial cuMAC

## 2.1. Getting Started with cuMAC

All cuMAC data structures and scheduler module classes are included in the name space `cumac`. The header files `api.h` and `cumac.h` should be included in the application program of cuMAC.

### 2.1.1. Data Flow

A diagram of cuMAC data flow for both CPU MAC scheduler host and GPU execution is given in following figure:

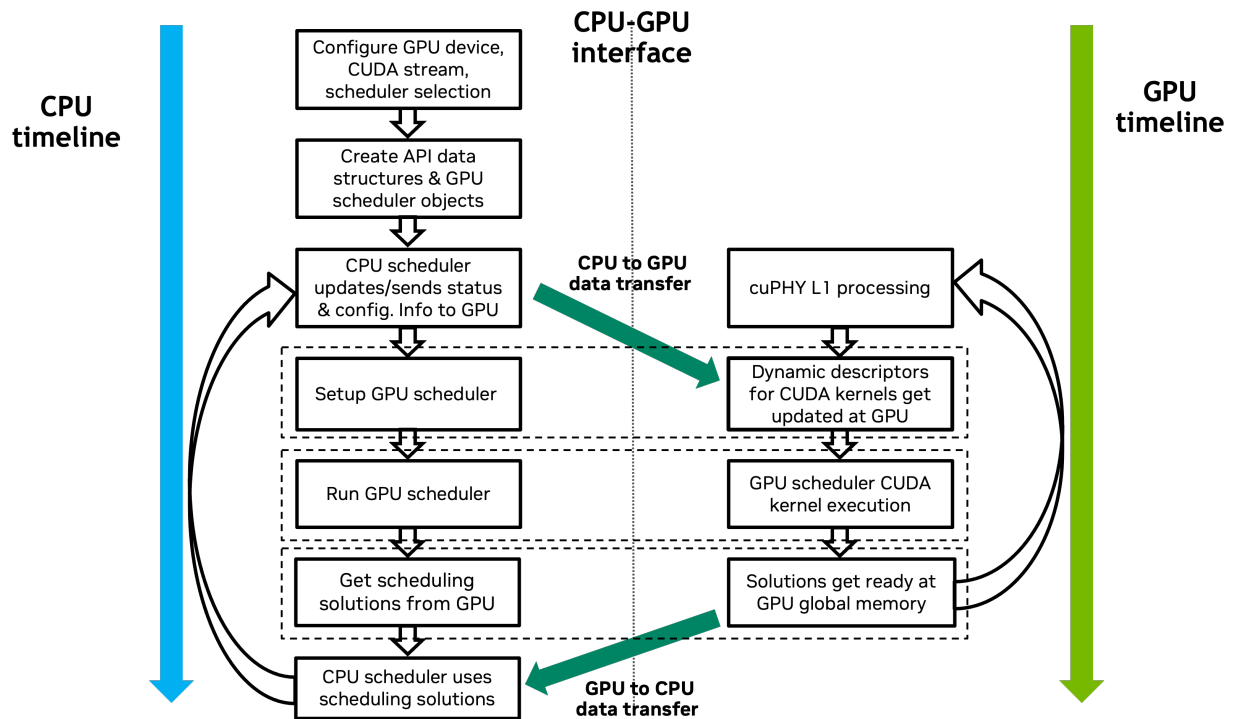


Fig. 1: cuMAC multi-cell scheduler execution data flow

Each cuMAC scheduler module (UE selection, PRB allocation, layer selection, MCS selection, etc.) is implemented as a C++ class, consisting of constructors with different combinations of input arguments, a destructor, a `setup ()` function to set up the CUDA kernels in each TTI and a `run ()` function to execute the scheduling algorithms in each TTI.

All parameters and data buffers required by the cuMAC scheduler modules are wrapped into three cuMAC API data structures, including `cumacCellGrpUeStatus`, `cumacCellGrpPrms`, and `cumacSchdSol`. Each of these data structures contains a number of constant parameters, and a number of data buffers whose memories are allocated on GPU.

In the initialization phase, the objects of all cuMAC scheduler modules are created using their corresponding constructors. Meanwhile, the above-mentioned three API data structures are also created, with their constant parameters being properly set up and data buffers getting memory allocations on GPU.

In the per-TTI execution, the CPU MAC scheduler host first prepares all the required data in GPU memory for the three API data structures. Then the `setup ()` function of each cuMAC scheduler module is called 1) to pass the required constant parameters and addresses of the data buffer GPU memories from the API data structures to the scheduler module objects, and 2) to complete the internal configuration of the CUDA kernels. Next, the `run ()` function of each scheduler module is called to execute the scheduling algorithms and obtain the scheduling solutions. Finally, the CPU MAC host transfers the computed scheduling solutions from GPU to CPU and applies them in the system.

## 2.1.2. Quick Setup

### 2.1.2.1 Prerequisites

1. CMake (version 3.18 or newer)

If you have a version of CMake installed, the version number can be determined as follows:

```
cmake --version
```

You can download the latest version of CMake from the [official CMake website](#).

2. CUDA (version 12 or newer)

CMake intrinsic CUDA support will automatically detect a CUDA installation using a CUDA compiler (nvcc), which is located via the `PATH` environment variable. To check for nvcc in your `PATH`:

```
which nvcc
```

To use a non-standard CUDA installation path (or to use a specific version of CUDA):

```
export PATH=/usr/local/cuda-12.0/bin:$PATH`
```

For more information on CUDA support in CMake, see <https://devblogs.nvidia.com/building-cuda-applications-cmake/>. (The statement above is equivalent to “`-gencode arch=compute_80,code=sm_80 -gencode arch=compute_90,code=sm_90`”.)

3. cuMAC requires a minimum GPU architecture of Ampere or newer.
4. HDF5 (Hierarchical Data Format 5)

The cuMAC CMake system currently checks for a specific version (1.10) of HDF5. To install a specific version of HDF5 from a source code archive:

- a. Remove the original hdf5 library (if necessary)

```
dpkg -l | grep hdf5
```

```
sudo apt-get remove <name of these libraries>
```

- b. To build from source:

```
wget https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.5/src/
↳ hdf5-1.10.5.tar.gz`
tar -xzf hdf5-1.10.5.tar.gz
cd hdf5-1.10.5
./configure --prefix=/usr/local --enable-cxx --enable-build-mode=production
sudo make install
```

### 2.1.2.2 Getting and building cuMAC

1. To download cuMAC, you can use the following link:

```
git clone --recurse-submodules https://gitlab-master.nvidia.com/gputelecom/cumac
```

2. To build cuMAC, use the following commands:

```
cd cumac
mkdir build && cd build
cmake ..
make
```

#### Additional CMake options:

- Creating a release build (using the default list of target architectures):

```
cmake -DCMAKE_BUILD_TYPE=Release ..
```

- Creating a debug build (using the default list of target architectures):

```
cmake .. -DCMAKE_BUILD_TYPE=Debug
```

- Specifying a single GPU architecture (e.g., to reduce compile time):

```
cmake .. -DCMAKE_CUDA_ARCHITECTURES="80"
```

- Specifying multiple GPU architectures:

```
cmake .. -DCMAKE_CUDA_ARCHITECTURES="80;90"
```

The statement above is equivalent to `-gencode arch=compute_80,code=sm_80 -gencode arch=compute_90,code=sm_90` .

## 2.2. cuMAC API Reference

### 2.2.1. cuMAC API Data Structures

#### 2.2.1.1 CumacCellGrpPrms

API data structure containing cell group information of the coordinated cells.

Field	Type	Description
nUe	uint16_t	Total number of selected UEs in a TTI of all coordinated cells. Value: 0 -> 65535
nActiveUe	uint16_t	Total number of active UEs of all coordinated cells. Value: 0 -> 65535
numUeSchd-PerCellTTI	uint8_t	Number of UEs selected/scheduled per TTI per cell. Value: 0 -> 255
nCell	uint16_t	Total number of coordinated cells. Value: 0 -> 65535
nPrbGrp	uint16_t	Total number of PRGs per cell. Value: 0 -> 65535
nBsAnt	uint8_t	Number of BS antenna ports. Value: 0 -> 255
nUeAnt	uint8_t	Number of UE antenna ports. Value: 0 -> 255
W	float	Frequency bandwidth (Hz) of a PRG. Value: 12 * subcarrier spacing * number of PRBs per PRG
sigmaSqrd	float	Noise variance. Value: noise variance value in watts
precodingScheme	uint8_t	Precoder type. Value: 0: No precoding 1: SVD precoder
receiverScheme	uint8_t	Receiver/equalizer type. Value: Currently only support 1: MMSE-IRC receiver
allocType	uint8_t	PRG allocation type. Value: 0: non-consecutive type-0 allocation 1: consecutive type-1 allocation
betaCoeff	float	Coefficient for adjusting the cell-edge UEs' performance in multi-cell scheduling Value: non-negative real number. The default value is 1.0, representing the classic proportional-fairness scheduling.

continues on next page

Table 1 – continued from previous page

Field	Type	Description
sinValThr	float	Singular value threshold for layer selection. Value: in (0, 1). Default value is 0.1
prioWeight-Step	uint16_t	For priority-weight based scheduling algorithm. Step size for UE priority weight increment per TTI if UE does not get scheduled. Value: default 100
cellId	uint16_t[nCell]	IDs of coordinated cells. One dimensional array. Value of each element: Denote cldx = 0, 1, ..., nCell-1 as the coordinated cell index. cellId[cldx] is the ID of the cldx-th coordinated cell
cellAssoc	uint8_t[nCell*nUe]	Cell-UE association indication for all the selected UEs of the coordinated cells. One dimensional array. Value of each element: Denote cldx = 0, 1, ..., nCell-1 as the coordinated cell index. Denote uldx = 0, 1, ..., nUe-1 as the selected UE index in the coordinated cells. cellAssoc[cldx*nUe + uldx] == 1 means the uldx-th selected UE is associated with cldx-th coordinated cell, 0 otherwise.
cellAssocActUe	uint8_t[nCell*nActiveUe]	Cell-UE association indication for all active UEs of the coordinated cells. One dimensional array. Value of each element: Denote cldx = 0, 1, ..., nCell-1 as the coordinated cell index. Denote uldx = 0, 1, ..., nActiveUe-1 as the global active UE index in the coordinated cells. cellAssocActUe[cldx*nUe + uldx] == 1 means the uldx-th active UE is associated with cldx-th coordinated cell, 0 otherwise.
prgMsk	uint8_t[nCell][nPrbGrp]	Per-cell bit map for the availability of each PRG for allocation Two-dimensional array. Value of each element: Denote cldx = 0, 1, ... nCell-1 as the coordinated cell index Denote prgIdx = 0, 1, ..., nPrbGrp-1 as the PRG index prgMsk[cldx][prgIdx] is the availability indicator for the prgIdx-th PRG in the cldx-th coordinated cell 0 - unavailable, 1 - available

continues on next page

Table 1 – continued from previous page

Field	Type	Description
postEqSinr	float[nActiveUe* nPrbGrp*nUeAnt]	<p>Array of the per-PRG per-layer post-equalizer SINRs of all active UEs in the coordinated cells.</p> <p>One dimensional array.</p> <p>Value of each element:</p> <p>Denote uldx = 0, 1, ..., nActiveUe-1 as the global active UE index in the coordinated cells.</p> <p>Denote prgIdx = 0, 1, ..., nPrbGrp-1 as the PRG index.</p> <p>Denote layerIdx = 0, 1, ..., nUeAnt-1 as the layer index.</p> <p>postEqSinr[uldx*nPrbGrp* nUeAnt + prgIdx*nUeAnt + layerIdx] is the uldx-th active UE's post-equalizer SINR on the prgIdx-th PRG and the layerIdx-th layer.</p>
wbSinr	float[nActiveUe* nUeAnt]	<p>Array of wideband per-layer post-equalizer SINRs of all active UEs in the coordinated cells.</p> <p>One dimensional array.</p> <p>Value of each element:</p> <p>Denote uldx = 0, 1, ..., nActiveUe-1 as the global active UE index in the coordinated cells.</p> <p>Denote layerIdx = 0, 1, ..., nUeAnt-1 as the layer index.</p> <p>wbSinr[uldx*nUeAnt + layerIdx] is the uldx-th active UE's wideband post-equalizer SINR on the layerIdx-th layer.</p>
FP32: estH_fr or FP16: estH_fr_half	<p>FP32: cuComplex[nCell][nUe*nPrbGrp* nBsAnt*nUeAnt]</p> <p>Or</p> <p>FP16: __nv_bfloat162 [nCell][nUe* nPrbGrp*nBsAnt* nUeAnt]</p>	<p>Per-cell array of the narrow-band SRS channel estimate coefficients for the selected UEs in the coordinated cells.</p> <p>Two-dimensional array: the 1st dimension is for cells, and the 2nd dimension is for UEs, PRGs, and antenna ports.</p> <p>Value of each element:</p> <p>Denote cldx = 0, 1, ..., nCell-1 as the coordinated cell index.</p> <p>Denote uldx = 0, 1, ..., nUe-1 as the selected UE index in the coordinated cells.</p> <p>Denote prgIdx = 0, 1, ..., nPrbGrp-1 as the PRG index.</p> <p>Denote bsAntIdx = 0, 1, ..., nBsAnt-1 as the BS antenna port index</p> <p>Denote ueAntIdx = 0, 1, ..., nUeAnt as the UE antenna port index</p> <p>estH_fr[cldx][uldx* nPrbGrp*nBsAnt*nUeAnt + prgIdx* nBsAnt*nUeAnt + bsAntIdx*nUeAnt + ueAntIdx] is the complex channel coefficient between the cldx-th cell and the uldx-th selected UE in the cell group on the prgIdx-th PRG, the bsAntIdx-th BS antenna port and the ueAntIdx-th UE antenna port.</p> <p>(The above applies to the FP16 version as well)</p>

continues on next page

Table 1 – continued from previous page

Field	Type	Description
prdMat	cuComplex[nUe* nPrbGrp* nBsAnt* nBsAnt]	<p>Array of the precoder/beamforming weights for the selected UEs in the coordinated cells.</p> <p>One dimensional array.</p> <p>Value of each element:</p> <p>Denote uldx = 0, 1, ..., nUe-1 as the selected UE index in the coordinated cells.</p> <p>Denote prgIdx = 0, 1, ..., nPrbGrp-1 as the PRG index.</p> <p>Denote inPortIdx = 0, 1, ..., nBsAnt-1 as the precoder input port index</p> <p>Denote outPortIdx = 0, 1, ..., nBsAnt-1 as the precoder output port index</p> <p>prdMat[uldx*nPrbGrp* nBsAnt*nBsAnt + prgIdx* nBsAnt*nBsAnt + inPortIdx *nBsAnt + outPortIdx] is the precoder/beamforming weight of the uldx-th selected UE in the coordinated cells on the prgIdx-th PRG and between the inPortIdx-th input port and the outPortIdx-th output port.</p>
detMat	cuComplex[nUe* nPrbGrp* nUeAnt* nUeAnt]	<p>Array of the detector/beamforming weights for the selected UEs in the coordinated cells.</p> <p>One dimensional array.</p> <p>Value of each element:</p> <p>Denote uldx = 0, 1, ..., nUe-1 as the selected UE index in the coordinated cells.</p> <p>Denote prgIdx = 0, 1, ..., nPrbGrp-1 as the PRG index.</p> <p>Denote inPortIdx = 0, 1, ..., nUeAnt-1 as the detector input port index.</p> <p>Denote outPortIdx = 0, 1, ..., nUeAnt-1 as the detector output port index.</p> <p>detMat[uldx*nPrbGrp*nUeAnt*nUeAnt + prgIdx*nUeAnt*nUeAnt + inPortIdx*nUeAnt + outPortIdx] is the detector weight of the uldx-th selected UE on the prgIdx-th PRG and between the inPortIdx-th input port and the outPortIdx-th output port.</p>
sinVal	float[nUe* nPrbGrp*nUeAnt]	<p>Array of the per-UE, per-PRG, per-layer singular values obtained from SVD.</p> <p>One dimensional array.</p> <p>Value of each element:</p> <p>Denote uldx = 0, 1, ..., nUe-1 as the 0-based UE index for the selected UEs in the coordinated cells.</p> <p>Denote prgIdx = 0, 1, ..., nPrbGrp-1 as the PRG index.</p> <p>Denote layerIdx = 0, 1, ..., nUeAnt-1 as the layer index.</p> <p>sinVal[uldx*nPrbGrp*nUeAnt + prgIdx*nUeAnt + layerIdx] is the UE uldx's layerIdx-th largest singular value on PRG prgIdx</p> <p>For each UE and on each PRG, the singular values are stored in descending order.</p>

## 2.2.1.2 cumacCellGrpUeStatus

API data structure containing the per-UE information of the coordinated cell group.

Field	Type	Description
avgRates	float[nUe]	Array of the long-term average data rates of the selected UEs in the coordinated cells. One dimensional array. Value of each element: Denote uldx = 0, 1, ..., nUe-1 as the selected UE index in the coordinated cells. avgRates[uldx] is the long-term average throughput of the uldx-th selected UE in the coordinated cells.
avgRate- sActUe	float[nActiveUe]	Array of the long-term average data rates of all active UEs in the coordinated cells One dimensional array. Value of each element: Denote uldx = 0, 1, ..., nActiveUe-1 as the global active UE index in the coordinated cells. avgRatesActUe[uldx] is the long-term average throughput of the uldx-th active UE in the coordinated cells.
prioWeigh- tActUe	uint16_t [nActiveUe]	For priority-based UE selection. Priority weights of all active UEs in the coordinated cells One dimensional array. Value of each element: Denote uldx = 0, 1, ..., nActiveUe-1 as the global UE index for all active UEs in the coordinated cells. prioWeightActUe[uldx] is the uldx-th active UE's priority weight. 0xFFFF indicates an invalid element.
tbErrLast	int8_t[nUe]	Array of the selected UEs' transport block (TB) decoding error indicators of the last transmissions One dimensional array. Value of each element: Denote uldx = 0, 1, ..., nUe-1 as the selected UE index in the coordinated cells. tbErrLast[uldx] is the uldx-th selected UE's TB decoding error indicator of the last transmission. -1 - the last transmission is not a new transmission (is a re-transmission) 0 - decoded correctly 1 - decoding error ** Note that if the last transmission of a UE is not a new transmission, tbErrLast of that UE should be set to -1.

continues on next page



Table 2 – continued from previous page

Field	Type	Description
tbErrLastActUe	int8_t[nActiveUe]	<p>TB decoding error indicators of all active UEs in the coordinated cells.</p> <p>One dimensional array.</p> <p>Value of each element:</p> <p>Denote <math>uldx = 0, 1, \dots, nActiveUe-1</math> as the global UE index for all active UEs in the coordinated cells.</p> <p>tbErrLastActUe[uldx] is the uldx-th active UE's TB decoding error indicator:</p> <p>-1 - the last transmission is not a new transmission (is a re-transmission)</p> <p>0 - decoded correctly</p> <p>1 - decoding error</p> <p>** Note that if the last transmission of a UE is not a new transmission, tbErrLastActUe of that UE should be set to -1.</p>
newDataActUe	int8_t[nActiveUe]	<p>Indicators of initial transmission/retransmission for all active UEs.</p> <p>One dimensional array.</p> <p>Value of each element:</p> <p>Denote <math>uldx = 0, 1, \dots, nActiveUe-1</math> as the global UE index for all active UEs in the coordinated cells.</p> <p>newDataActUe[uldx] is the indicator of initial transmission/retransmission for the uldx-th active UE in the coordinated cells</p> <p>0 – retransmission</p> <p>1 - new data/initial transmission</p> <p>-1 indicates an invalid element</p>
allocSolLastTx	<p>For type-0 PRG allocation:</p> <p>int16_t[nCell*nPrbGrp]</p> <p>For type-1 PRG allocation:</p> <p>int16_t[2*nUe]</p>	<p>The PRG allocation solution of the last transmission of the selected/scheduled UEs in the coordinated cells</p> <p>Format referring to the description for allocSol in the cumacSchdSol structure</p>
mcsSelSolLastTx	int16_t[nUe]	<p>MCS selection solution of the last transmission of the selected/scheduled UEs in the coordinated cells.</p> <p>Format referring to the description for mcsSelSol in the cumacSchdSol structure</p>
layerSelSolLastTx	uint8_t[nUe]	<p>Layer selection solution of the last transmission of the selected/scheduled UEs in the coordinated cells.</p> <p>Format referring to the description for layerSelSol in the cumacSchdSol structure</p>

2.2.1.3 cumacSchdSol

API data structure containing the scheduling solutions.

Field	Type	Description
setSchdUePerCellTTI	uint16_t[nCell* numUeSchdPerCellTTI]	Set of global IDs of the selected UEs per cell per TTI. One dimensional array. Value of each element: Denote cldx = 0, 1, ..., nCell-1 as the coordinated cell index. Denote i = 0, 1, ..., numUeSchdPerCellTTI-1 as the i-th selected UE in a given cell. setSchdUePerCellTTI[cldx*numUeSchdPerCellTTI + i] is within {0, 1, ..., nActiveUe-1} and represents the global active UE index of the i-th selected UE in the cldx-th coordinated cell.
allocSol	For type-0 PRG allocation: int16_t[nCell*nPrbGrp] For type-1 PRG allocation: int16_t[2*nUe]	PRB group allocation solution for the selected UEs per TTI in the coordinated cells One dimensional array. Value of each element: For type-0 PRG allocation: Denote prgIdx = 0, 1, ..., nPrbGrp-1 as the PRG index. Denote cldx = 0, 1, ..., nCell-1 as the coordinated cell index. allocSol[prgIdx*nCell + cldx] indicates the selected UE index (0, 1, ..., nUe-1) that the prgIdx-th PRG is allocated to in the cldx-th coordinated cell. -1 indicates that a given PRG in a cell is not allocated to any UE. For type-1 PRG allocation: Denote uldx = 0, 1, ..., nUe-1 as the selected UE index in the coordinated cells. allocSol[2*uldx] is the starting PRG index of the uldx-th selected UE. allocSol[2*uldx + 1] is the ending PRG index of the uldx-th selected UE plus one. -1 indicates that a given UE is not being allocated to any PRG.
pfMetricArr	float[array_size] array_size = nCell * the minimum power of 2 that is no less than nPrbGrp* numUeSchdPerCellTTI	Array to store the computed PF metrics per UE and per PRG. Only used for type-1 PRG allocation. One dimensional array. GPU memory allocated for CUDA kernel execution. Not used externally. Memory should be allocated when initializing the cuMAC API. Value of each element: floating-type value of a computed PF metric.

continues on next page

Table 3 – continued from previous page

Field	Type	Description
pflIdArr	uint16_t [array_size] array_size = nCell * the minimum power of 2 that is no less than nPrbGrp*n umUeSchd- PerCellTTI	Array to indicate the PRG and UE indices of the sorted PF metrics. Only used for type-1 PRG allocation. One dimensional array. GPU memory allocated for CUDA kernel execution. Not used externally. Memory should be allocated when initializing the cuMAC API. Value of each element: 0 -> 65535
mcsSelSol	int16_t[nUe]	MCS selection solution for the selected UEs per TTI in the coordinated cells One dimensional array. Value of each element: Denote uldx = 0, 1, ..., nUe-1 as the selected UE index in the coordinated cells. mcsSelSol[uldx] indicates the MCS level for the uldx-th selected UE in the coordinated cells. Range of each element: 0, 1, ..., 27 (Currently only support Table 5.1.3.1-2: MCS index table 2, 3GPP TS 38.214). -1 indicates an element is invalid.
layerSelSol	uint8_t[nUe]	Layer selection solution for the selected UEs per TTI in the coordinated cells. One dimensional array. Value of each element: Denote uldx = 0, 1, ..., nUe-1 as the selected UE index in the coordinated cells. layerSelSol[uldx] indicates the number of layers selected for the uldx-th selected UE in the coordinated cells. Range of each element: 0, 1, ..., nUeAnt-1 The selected layers have singular values descending from the largest one.

## 2.2.2. cuMAC Scheduler Module API

### 2.2.2.1 Multi-cell proportional-fairness UE down-selection

Wrapper class and public member functions:

```
class cumac::multiCellUeSelection
public:
// constructor
multiCellUeSelection();
```

(continues on next page)

(continued from previous page)

```

// destructor
~multiCellUeSelection();

// setup() function for per-TTI algorithm execution
void setup(cumac::cumacCellGrpUeStatus* cellGrpUeStatus,
           cumac::cumacSchdSol* schdSol,
           cumac::cumacCellGrpPrms* cellGrpPrms,
           uint8_t in_enableHarq,
           cudaStream_t strm);
// requires external synchronization
// set in_enableHarq to 1 if HARQ is enabled; 0 otherwise

// run() function for per-TTI algorithm execution
void run(cudaStream_t strm);
// requires external synchronization

// parameter/data buffer logging function for debugging purpose
void debugLog();
// for debugging only, printing out dynamic descriptor parameters

```

### 2.2.2.2 Multi-cell proportional-fairness PRB scheduler

Wrapper class and public member functions:

```

class cumac::multiCellScheduler

public:
// constructor
multiCellScheduler();

// destructor
~multiCellScheduler();

// setup() function for per-TTI algorithm execution
void setup(cumac::cumacCellGrpUeStatus* cellGrpUeStatus,
           cumac::cumacSchdSol* schdSol,
           cumac::cumacCellGrpPrms* cellGrpPrms,
           uint8_t in_DL,
           uint8_t in_columnMajor,
           uint8_t in_halfPrecision,
           uint8_t in_lightWeight,
           cudaStream_t strm);
// set in_DL to 1 if setup for DL scheduling; 0 otherwise
// in_columnMajor: 0 - row-major channel access, 1 - column-major channel access
// in_halfPrecision: 0 - call FP32 floating type kernel, 1 - call FP16 (bfloat162)
↳ half-precision kernel
// in_lightWeight: 0 - call heavy-weight kernel, 1 - call light-weight kernel
// in_enableHarq: 0 - HARQ disabled, 1 - HARQ enabled
// requires external synchronization

// run() function for per-TTI algorithm execution
void run(cudaStream_t strm);
// requires external synchronization

// parameter/data buffer logging function for debugging purpose

```

(continues on next page)

(continued from previous page)

```
void debugLog();
// for debugging only, printing out dynamic descriptor parameters
```

### 2.2.2.3 Multi-cell layer selection

Wrapper class and public member functions:

```
class cumac::multiCellLayerSel
public:
// constructor
multiCellLayerSel();

// desctructor
~multiCellLayerSel();

// setup() function for per-TTI algorithm execution
void setup(cumacCellGrpUeStatus* cellGrpUeStatus,
           cumacSchdSol* schdSol,
           cumacCellGrpPrms* cellGrpPrms,
           uint8_t in_enableHarq,
           cudaStream_t strm);
// in_enableHarq: 0 - HARQ disabled, 1 - HARQ enabled
// requires external synchronization

// run() function for per-TTI algorithm execution
void run(cudaStream_t strm);
// requires external synchronization

// parameter/data buffer logging function for debugging purpose
void debugLog();
// for debugging only, printing out dynamic descriptor parameters
```

### 2.2.2.4 Multi-cell MCS selection + outer-loop link adaptation (OLLA)

Wrapper class and public member functions:

```
class cumac::mcsSelectionLUT
public:
// constructor
mcsSelectionLUT(uint16_t nActiveUe, cudaStream_t strm);
// requires external synchronization
// uint16_t nActiveUe is the (maximum) total number of active UEs in all
// coordinated cells

// destructor
~mcsSelectionLUT();

// setup() function for per-TTI algorithm execution
void setup(cumacCellGrpUeStatus* cellGrpUeStatus,
           cumacSchdSol* schdSol,
           cumacCellGrpPrms* cellGrpPrms,
```

(continues on next page)

(continued from previous page)

```

        uint8_t in_DL,
        uint8_t in_baseline,
        cudaStream_t strm);
// in_DL: 0 - UL, 1 - DL
// in_baseline: 0 - not using baseline algorithm, 1 - using baseline
algorithm
// requires external synchronization

// run() function for per-TTI algorithm execution
void run(cudaStream_t strm);

// parameter/data buffer logging function for debugging purpose
void debugLog();
// for debugging only, printing out dynamic descriptor parameters

```

Outer-loop link adaptation (OLLA) data structure:

```

// structure containing outer-loop link adaptation algorithm parameters
struct ollaParam {
    float delta; // offset to SINR estimation
    float delta_ini; // initial value for delta parameter
    float delta_up; // step size for increasing delta parameter
    float delta_down; // step size for decreasing delta parameter
};

```

## 2.3. Examples

### 2.3.1. cuMAC test vectors generated as HDF5 files

Test vectors are located in the testVectors directory. Each test vector contains parameters and data arrays defined in the cuMAC API structures (aerial\_sdk/cuMAC/src/api.h): cumacCellGrpUeStatus, cumacCellGrpPrms, and cumacSchdSol.

Parameter configurations can be specified the aerial\_sdk/cuMAC/examples/parameters.h file.

Use the multiCellSchedulerUeSelection testbench (aerial\_sdk/cuMAC/examples/multiCellSchedulerUeSelection) to create TVs:

► DL TV:

```

./aerial_sdk/cuMAC/build/examples/multiCellSchedulerUeSelection/
↪multiCellSchedulerUeSelection -t 1

```

► UL TV:

```

./aerial_sdk/cuMAC/build/examples/multiCellSchedulerUeSelection/
↪multiCellSchedulerUeSelection -d 0 -t 1

```

An H5 TV is created after the last simulated TTI. The assumption is that the simulation duration is long enough so that the scheduler algorithm's performance converges.

## 2.3.2. Single-TTI tests

Given the same input parameters of a single TTI, GPU and CPU implementations of the same scheduler algorithms should give the same output solution.

Two types of tests:

- ▶ Per scheduler module tests: DL/UL UE selection, DL/UL PRG allocation, DL/UL layer selection, and DL/UL MCS selection
- ▶ Complete DL/UL scheduler pipeline tests

TV loading-based single-TTI testbench (`aerial_sdk/cuMAC/examples/tvLoadingTest`).

After building `cumac`, use the following command to check input arguments of the testbench: `./aerial_sdk/cuMAC/build/examples/tvLoadingTest/tvLoadingTest -h`

- ▶ Per scheduler module tests:

- ▶ DL UE selection:

```
./aerial_sdk/cuMAC/build/examples/tvLoadingTest/tvLoadingTest -i [path to TV]
↪ -g 2 -d 1 -m 01000
```

- ▶ DL PRG allocation:

```
./aerial_sdk/cuMAC/build/examples/tvLoadingTest/tvLoadingTest -i [path to TV]
↪ -g 2 -d 1 -m 00100
```

- ▶ DL layer selection:

```
./aerial_sdk/cuMAC/build/examples/tvLoadingTest/tvLoadingTest -i [path to TV]
↪ -g 2 -d 1 -m 00010
```

- ▶ DL MCS selection:

```
./aerial_sdk/cuMAC/build/examples/tvLoadingTest/tvLoadingTest -i [path to TV]
↪ -g 2 -d 1 -m 00001
```

- ▶ UL scheduler modules can be tested by setting input argument: `-d 0`

- ▶ Complete DL/UL scheduler pipeline tests

- ▶ DL/UL scheduler modules executed sequentially: UE selection > PRG allocation > layer selection > MCS selection

- ▶ DL scheduler pipeline:

```
./aerial_sdk/cuMAC/build/examples/tvLoadingTest/tvLoadingTest -i [path to TV]
↪ -g 2 -d 1 -m 01111
```

- ▶ UL scheduler pipeline:

```
./aerial_sdk/cuMAC/build/examples/tvLoadingTest/tvLoadingTest -i [path to TV]
↪ -g 2 -d 0 -m 01111
```

Passing criteria:

Solutions computed by CPU and GPU should match exactly: testbench returns 1 (PASS) or 0 (FAIL)

### 2.3.3. Continuous-time tests

With the same initial state, GPU and CPU implementations of the same scheduler algorithms should achieve similar performance curves when running for a period of time.

- ▶ Complete DL/UL scheduler pipeline tests
  - ▶ Continuous-time testbench (aerial\_sdk/cuMAC/examples/multiCellSchedulerUeSelection)
  - ▶ After building cumac, use the following command to check input arguments of the testbench:

```
./aerial_sdk/cuMAC/build/examples/multiCellSchedulerUeSelection/  
↩multiCellSchedulerUeSelection -h
```

- ▶ No need to use pre-generated H5 TVs. All parameters are computed using cuMAC internal simulator.
- ▶ Simulator configuration can be specified using the aerial\_sdk/cuMAC/examples/parameters.h file.
- ▶ DL/UL scheduler modules executed sequentially: UE selection > PRG allocation > layer selection > MCS selection
- ▶ DL scheduler pipeline test:

```
./aerial_sdk/cuMAC/build/examples/multiCellSchedulerUeSelection/  
↩multiCellSchedulerUeSelection
```

- ▶ UL scheduler pipeline test:

```
./aerial_sdk/cuMAC/build/examples/multiCellSchedulerUeSelection/  
↩multiCellSchedulerUeSelection -d 0
```

Passing criteria:

Performance curves achieved by GPU and CPU scheduler implementations should match: testbench returns 1 (PASS) or 0 (FAIL)

Two types of performance curves:

- ▶ Sum throughput of all cells
- ▶ CDF of per-UE throughput

Aerial cuMAC is a CUDA-based platform for accelerating 5G/6G MAC layer scheduler functions with NVIDIA GPUs. cuMAC supported scheduler functions include UE selection/grouping, PRB allocation, layer selection, MCS selection/link adaptation and dynamic beamforming, all designed for the joint scheduling of multiple coordinated cells. cuMAC offers a C/C++ based API for the offloading of scheduler functions from the L2 stack in the DUs to GPUs. In the future, cuMAC will evolve into a platform that combines AI/ML based scheduler enhancements with GPU acceleration.

cuMAC is the main component of the Aerial L2 scheduler acceleration solution. The figure above illustrates the overall data flow of the scheduler acceleration. The full solution consists of the following components: 1) Aerial Scheduler Acceleration API, which is a per-cell message passing-based interface between the 3<sup>rd</sup> party L2 stack on DU/CU and cuMAC-CP, 2) cuMAC-CP, 3) cell group-based cuMAC API, and 4)cuMAC multi-cell scheduler (cuMAC-sch) modules.

The 3<sup>rd</sup> party L2 stack sits on the CPU and contains a single-cell L2 scheduler for each individual cell under its control. To offload L2 scheduling to GPU for acceleration/performance purposes, in each



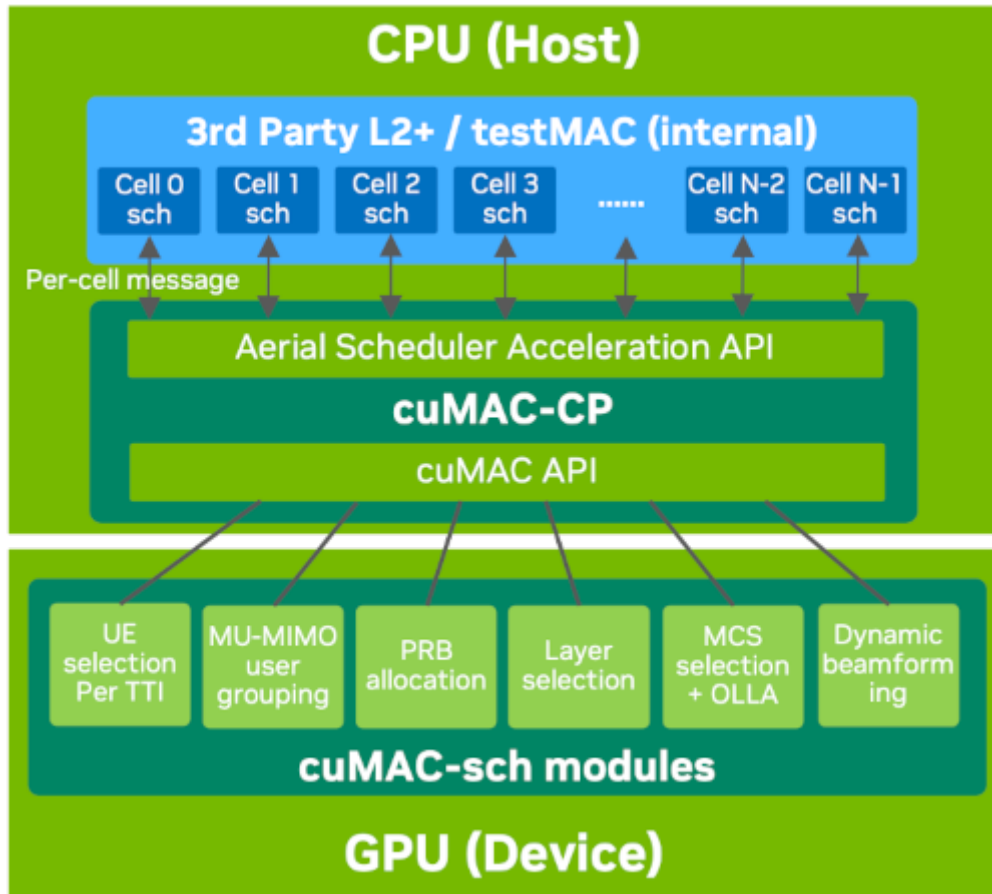


Fig. 2: Aerial L2 scheduler acceleration data flow chart

time slot (TTI), the L2 stack host sends per-cell request messages to cuMAC-CP through the Aerial Scheduler Acceleration API, which consists of required scheduling input & config. information from each single-cell scheduler. Upon receiving the per-cell request messages, cuMAC-CP integrates all scheduler input information from those (coordinated) cells into the cuMAC API cell group data structures and populates the GPU data buffers contained in these structures. Next, the cuMAC multi-cell scheduler (cuMAC-sch) modules are called by cuMAC-CP through cuMAC API to compute scheduling solutions for the given time slot (TTI). After the cuMAC-sch modules complete the computation and the scheduling solutions become available in the GPU memory, cuMAC-CP converts them into per-cell response messages and sends them back to the L2 stack host on CPU through the Aerial Scheduler Acceleration API. Finally, the L2 stack host uses the obtained solutions to schedule the cells under its control.

When there are multiple coordinated cell groups, a separate set of Aerial Scheduler Acceleration API, cuMAC-CP, cuMAC API and cuMAC instances should be constructed and maintained for each cell group.

### Implementation Details

- **Multi-cell scheduling** - All cuMAC scheduling algorithms are implemented as CUDA kernels that are executed by GPU and jointly compute the scheduling solutions (PRB allocation, MCS selection, layer selection, etc.) for a group of cells at the same time. The algorithms can be constrained to single cell scheduling by configuring a single cell in the cell group. A comparison between the single-cell scheduler and multi-cell scheduler approaches is given in the below figure.

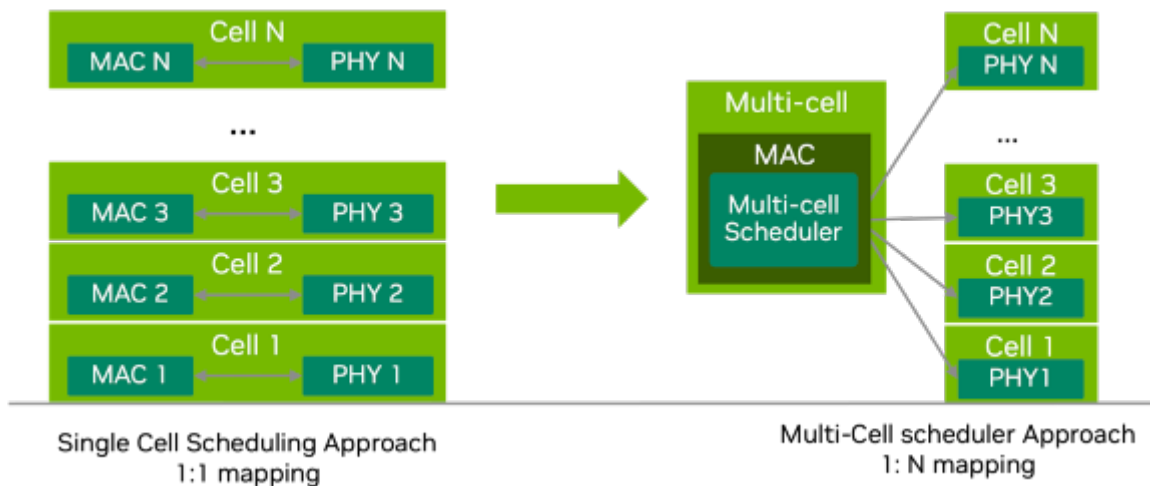


Fig. 3: Single-cell scheduler approach vs. multi-cell scheduler approach

### ► Scheduling algorithm CUDA implementation

- **PF UE down-selection algorithm** - cuMAC offers a PF-based UE selection algorithm to down-select a subset of UEs for new transmissions or HARQ re-transmissions in each TTI from the pool of all active UEs in each cell of a cell group. The association of UEs and cells in the cell group is an input to the UE selection module. When selecting UEs for each cell in each TTI, the UE selection algorithm first assigns a priority weight to each active UE in a cell and then sorts all active UEs in descending order of the priority weight. The subset of UEs that have the highest priority weights in each cell are selected for scheduling in a TTI. The number of selected UEs per cell is an input parameter to this module. HARQ re-transmissions are always assigned with the highest priority weight. For the new-transmission UEs, their priority weights are the PF metrics, calculated as the ratio of each UE's long-term average throughput and its instantaneous achievable data rate. The UE selection algorithm is imple-

mented as CUDA kernels that run on GPU and jointly select UEs for all cells in a cell group at the same time.

- ▶ **PF PRB allocation algorithms** - cuMAC offers algorithms to perform channel-aware and frequency-selective PRB allocation among a group of cells and their connected active UEs on a per-TTI basis. The input arguments to the PRB allocation algorithms include the narrow-band SRS channel estimates (MIMO channel matrices) per cell-UE link, the association solutions between cells and UEs, and other UE status and cell group parameters. The output is the PRB allocation solution for the cell group, whose data format depends on the type of allocation: 1) for type-0 allocation, a per UE binary bitmap indicating whether each PRB is allocated to the UE, and 2) for type-1 allocation, with 2 elements per UE indicating the starting and ending PRB indices for the UE's allocation. Two versions of the PRB allocation algorithms are provided, one for single cell scheduling and the other for multi-cell joint scheduling. A major difference between the two versions is that the multi-cell algorithm considers the impact of inter-cell interference in the evaluation of per-PRB SINRs, which can be derived from the narrow-band SRS channel estimates. The single-cell version does not explicitly consider inter-cell interference and only utilizes information restricted to each individual cell. The multi-cell algorithm can lead to a globally optimized resource allocation in a cell group by leveraging all available information from the coordinated multiple cells. A prototyping CUDA kernel implementation of PRB allocation algorithms is provided in the figure below.
- ▶ **Layer selection algorithm** - cuMAC offers layer selection algorithms that choose the best set of layers for transmission for a UE based on the singular value distribution across the UE's multiple layers. A predetermined singular value threshold is used to find the number of layers (with descending singular values) that can be supported on each subband (PRB group). Then the minimum number of layers across all allocated subbands to the UE is chosen as the optimal layer selection solution. Input arguments to the layer selection algorithms include the PRB allocation solution per UE, the singular values of each UE's channel on its allocated subbands, the association solutions between cells and UEs, and other UE status and cell group parameters. The output is the per-UE layer selection solution. The layer selection algorithm is implemented as CUDA kernels that run on GPU and jointly select layers for all UEs in a cell group at the same time.
- ▶ **MCS selection algorithm** - cuMAC offers MCS selection algorithms that choose the best feasible MCS (highest level that can meet a given BLER target) per UE based on a given PRB allocation solution. An outer-loop link adaptation algorithm is integrated internally to the MCS selection algorithm, which offsets the SINR estimates based on previous transport block decoding results per UE link. Input arguments to the MCS selection algorithms include the PRB allocation solution per UE, the narrow-band SRS channel estimates (MIMO channel matrices) per cell-UE link, the association solutions between cells and UEs, the decoding results of the last transport block for each UE, and other UE status and cell group parameters. The output is the per-UE MCS selection solution. The MCS selection algorithm is implemented as CUDA kernels that run on GPU and jointly select MCS for all UEs in a cell group at the same time.
- ▶ **Support for HARQ** - all the above cuMAC scheduler algorithms can support HARQ re-transmissions with non-adaptative mode, i.e., reusing the same scheduling solution of the initial transmission for re-transmissions.
- ▶ **CPU reference code** - CPU C++ implementation of the above algorithms is also provided for verification and performance evaluation purposes.
- ▶ **Different CSI types** - cuMAC offers scheduler algorithm CUDA kernels to work with different CSI types, including SRS channel coefficient estimates and CSI-RS based channel quality information.
- ▶ **Support for FP32 and FP16** - cuMAC offers scheduler algorithm CUDA kernels implemented in

FP32 and FP16. Using FP16 kernels can help reduce scheduler latency with a minor performance loss.

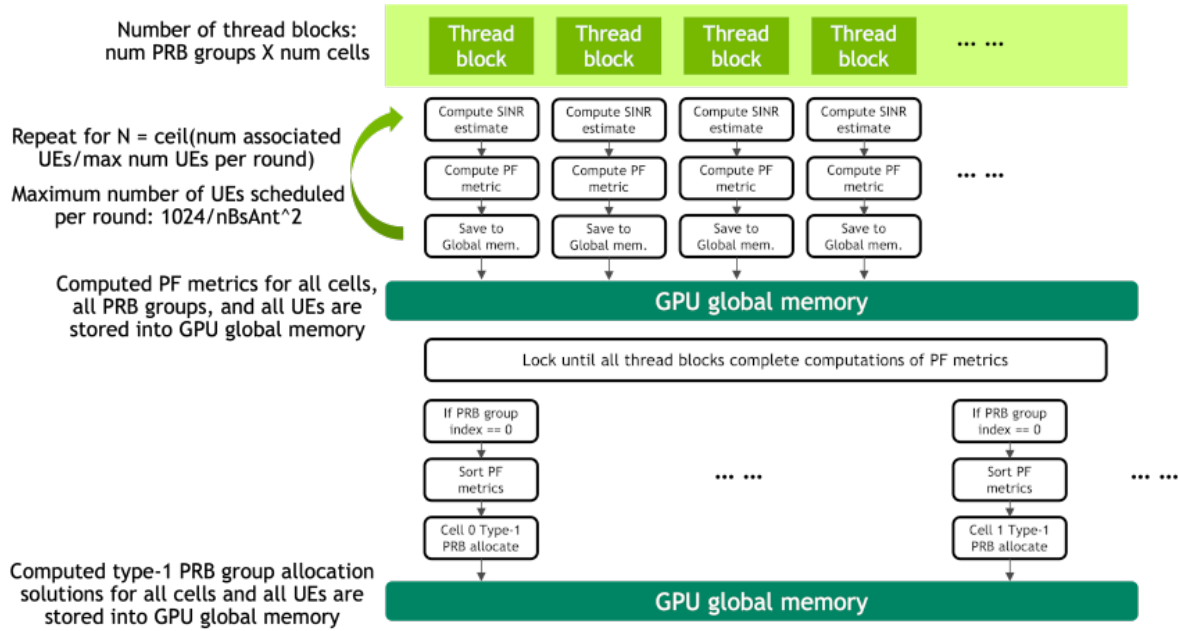


Fig. 4: A prototyping CUDA kernel implementation of PRB allocation algorithms

---

# Chapter 3. Aerial Data Lake

6G will be artificial intelligence (AI) native. AI and machine learning (ML) will extend through all aspects of next generation networks from the radio, baseband processing, the network core including system management, orchestration and dynamic optimization processes. GPU hardware, together with programming frameworks will be essential to realize this vision of a software defined native-AI communication infrastructure.

The application of AI/ML in the physical layer has particularly been a hot research topic.

There is no AI without data. While the synthetic data generation capabilities of Aerial Omniverse Digital Twin (AODT) and Sionna/SionnaRT are essential aspects of a research project, availability of over-the-air (OTA) waveform data from real-time systems is equally important. This is the role of Aerial Data Lake. It is a data capture platform supporting the capture of OTA radio frequency (RF) data from virtual radio access network (vRAN) networks built on the Aerial CUDA-Accelerated RAN. Aerial Data Lake consists of a data capture application (app) running on the base station (BS) distributed unit (DU), a database of samples collected by the app, and an application programming interface (API) for accessing the database.

## 3.1. Target Audience

Industry and university researchers and developers looking to bring ML to the physical layer with the end goal of benchmarking on OTA testbeds like NVIDIA ARC-OTA or other GPU-based BSs.

## 3.2. Key Features

Aerial Data Lake has the following features:

### **Real-time capture of RF data from OTA testbed**

- ▶ Aerial Data Lake is designed to operate with gnBs built on the Aerial CUDA-Accelerated RAN and that employ the Small Cell Forum FAPI interface between L2 and L1. One example system being the NVIDIA ARC-OTA network testbed. I/Q samples from O-RUs connected to the GPU platform via a O-RAN 7.2x split fronthaul interface are delivered to the host CPU and exported to the Aerial Data Lake database.

### **Aerial Data Lake APIs to access the RF database**

- ▶ The data passed to the layer-2 via `RX_Data.Indication` and `UL_TTI.Request` are exported to the database. The fields in these data structures form the basis of the database access APIs.

**Scalable and time coherent over arbitrary number of BSs**

- ▶ The data collection app runs on the same CPU that supports the DU. It runs on a single core, and the database runs on free cores. Because each BS is responsible for collecting its own uplink data, the collection process scales as more BSs are added to the network testbed. Database entries are time-stamped so data collected over multiple BSs can be used in a training flow in a time-coherent manner.

**Use in conjunction with pyAerial to generate training data for neural network physical layer designs**

- ▶ Aerial Data Lake can be used in conjunction with the NVIDIA pyAerial CUDA-Accelerated Python L1 library. Using the Data Lake database APIs, pyAerial can access RF samples in a Data Lake database and transform those samples into training data for all the signal processing functions in an uplink or downlink pipeline.

### 3.3. Design

Aerial Data Lake sits beside the Aerial L1 and copies out data that would be useful for machine learning into an external database.

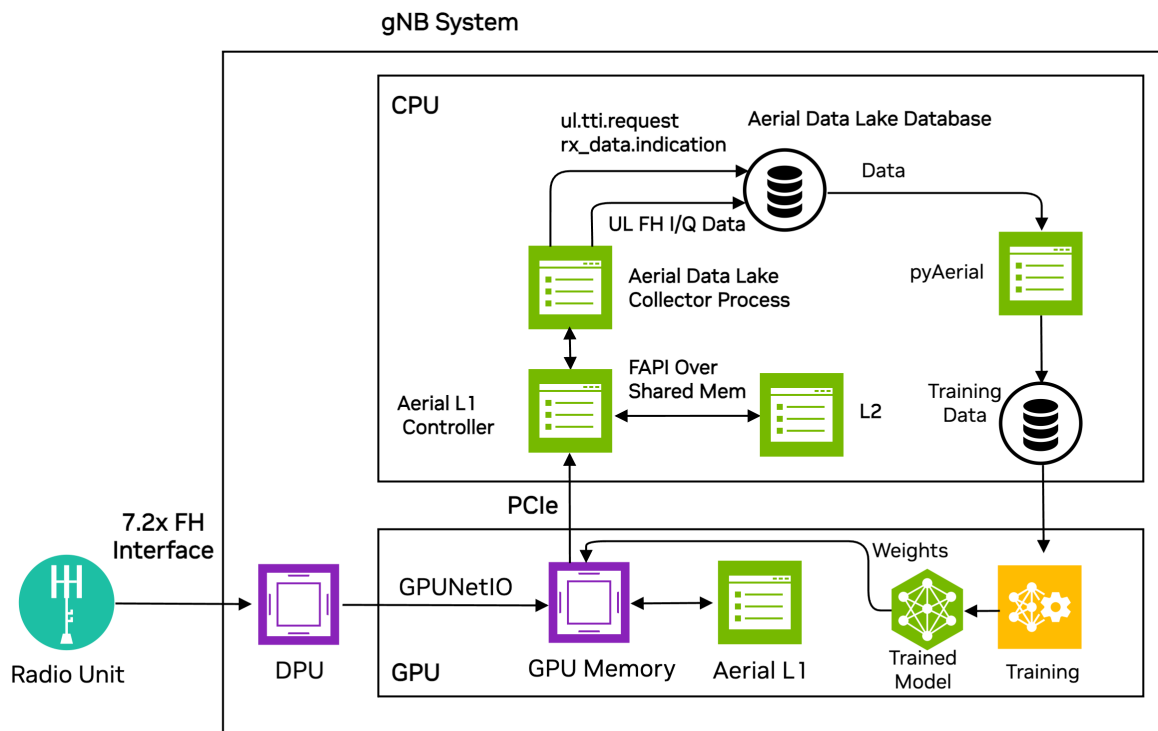


Fig. 1: Figure 1: The Aerial Data Lake data capture platform as part of the gNB.

Uplink I/Q data from one or more O-RAN radio units (O-RUs) is delivered to GPU memory where it is both processed by the Aerial L1 PUSCH baseband pipeline and delivered to host CPU memory. The Aerial Data Lake collector process writes the I/Q samples to the Aerial Data Lake database in the *fh* table. The *fh* table has columns for SFN, Slot, IQ samples as *fhData*, and the start time of that SFN.slot as *TsTaiNs*.

The collector app saves data that the L2 sent to L1 to describe UL OTA transmissions in UL\_TTI.Request messages as well as data returned to the L2 via RX\_Data.Indication and CRC.Indication. This data is then written to the *fapi* database table. These messages and the fields within them are described in [SCF 5G FAPI PHY Spec version 10.02](#), sections 3.4.3, 3.4.7, and 3.4.8.

Each gNB in a network testbed collects data from all O-RUs associated with it. That is, data collection over the span of a network is performed in a distributed manner, each gNB is building its own local database. Training can be performed locally at each gNB, and site-specific optimizations can be realized with this approach. Since the data in a database is time-stamped, the local databases can be consolidated at a centralized compute resource and training performed using the time aligned aggregated data. In cases where the aerial pusch pipeline was unable to decode due to channel conditions, retransmissions can be used as ground truth as long as one of the retransmissions succeeds, allowing the user to test algorithms with better performance than the originals.

The Aerial Data Lake database storage requirements depend on the number of O-RUs, the antenna configuration of the O-RU, the carrier bandwidth, the TDD pattern and the number of samples to be collected. Collecting IQ samples of 1 million transmissions from a single RU 4T4R O-RU employing a single 100MHz carrier will consume approximately 660 GB of storage.

Aerial Data Lake database comprises the fronthaul RF data. However, for many training applications access to data at other nodes in the receive pipeline is required. A pyAerial pipeline, together with the Data Lake database APIs, can access samples from an Aerial Data Lake database and transform that data into training data for any function in the pipeline.

Figure 2 illustrates data ingress from a Data Lake database into a pyAerial pipeline and using standard Python file I/O to generate training data for a soft de-mapper.

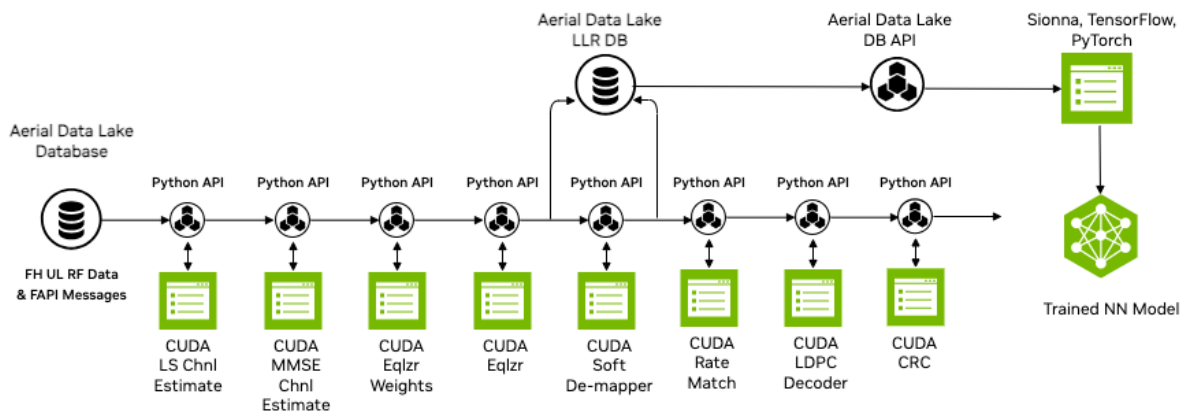


Fig. 2: Figure 2: pyAerial is used in conjunction with the NVIDIA data collection platform, namely, Aerial Data Lake to build training data sets for any node in the layer-1 downlink or uplink signal processing pipeline. The example shows a Data Lake database of over-the-air samples transformed into training data for a neural network soft de-mapper.

## 3.4. Installation

Aerial Data Lake is compiled by default as part of cuphycontoller. If you would like to record fresh data every time cuphycontroller is started, see the section on Fresh Data.

Start by installing [Clickhouse](#) database on the server collecting the data. The command below will download and run an instance of the clickhouse server in a docker container.

```
docker run -d \
--network=host \
-v $(realpath ./ch_data):/var/lib/clickhouse/ \
-v $(realpath ./ch_logs):/var/log/clickhouse-server/ \
--cap-add=SYS_NICE --cap-add=NET_ADMIN --cap-add=IPC_LOCK \
--name my-clickhouse-server --ulimit nofile=262144:262144 clickhouse/clickhouse-server
```

By default clickhouse will not drop large tables, and will return an error if attempted. The clickhouse-cpp library does not return exceptions so to avoid what looks like a cuphycontroller crash we recommend allowing it to drop large tables using the following command:

```
sudo touch './ch_data/flags/force_drop_table' && sudo chmod 666 './ch_data/flags/
↪force_drop_table'
```

## 3.5. Usage

In the cuphycontoller adapter yaml configuration file, enable data collection by specifying a core then start cuphycontroller as usual. The core should be on the same NUMA node as the rest of cuphycontroller, i.e. should follow the same pattern as the rest of the cores. An example of this can be found commented out in cuphycontroller\_P5G\_FXN\_R750.yaml.

```
cuphydriver_config:
# Fields added for data collection
  datalake_core: 19 # Core on which data collection runs. E.g isolated odd on R750,
↪any isolated core on gigabyte
  datalake_address: localhost
  datalake_samples: 1000000 # Number of samples to collect for each UE/RNTI. Defaults
↪to 1M
```

When enabled the *DataLake* object is created and *DataLake::dbInit()* initializes the two tables in the database. After cuphycontroller runs the PUSCH pipeline, cupycontroller calls *DataLake::notify()* with the addresses of the data to be saved, which *DataLake* then saves. When *DataLake::waitForLakeData* wakes up it calls *DataLake::dbInsert()* which appends data to respective *Clickhouse* columns, then sleeps waiting for more data. Once 50 PUSCH transmissions have been stored or a total of *data-lake\_samples* have been received the columns are appended to a *Clickhouse::Block* and inserted into the respective table.



## 3.6. Multi-Cell

Datalakes can be configured to capture data from multiple cells controlled by the same L1. The Jupyter notebook *datalake\_pusch\_multicell.ipynb* shows an example of using data captured from multiple cells. To capture the data for this example, cell 41 was controlled by testmac and cell 51 was controlled by a real L2. In order to do this the cuphycontroller L2 interface needs to be configured to work with two cells and L2s, and testmac needs to be configured to use `/dev/shm/nvipc1` rather than `/dev/shm/nvipc`. L2 should use the slot pattern `DDDSU`. Core allocations will need to be adjusted to suite the server being used.

## 3.7. Using Data Lake in Notebooks

Follow pyAerial instructions to build and launch that container. It must be run on a server with a GPU.

Three example notebooks are included:

- *datalake\_channel\_estimation.ipynb* performs channel estimation and plots the result.
- *datalake\_pusch\_decoding.ipynb* goes further and runs the full PUSCH decoding pipeline, both a fused version and a version built up of constituent parts.
- *datalake\_pusch\_multicell.ipynb* shows an exmple of trying to decode the same transmissions from multiple UEs across two cells.

See the [pyAerial examples section](#) for details.

## 3.8. Database Administration

### Note

These instructions assume that the cuBB container has been installed and started as in [Installing and Upgrading Aerial cuBB](#)

and that the clickhouse server has been installed as in the section on [Installation](#)

In the following examples this denotes a bash prompt:

```
$
```

and this denotes a clickhouse client prompt

```
aerial-gnb :)
```

### 3.8.1. Database Import

There are example *fapi* and *fh* tables included in Aerial CUDA-Accelerated RAN container. These tables can be imported into the clickhouse database by copying them from the container to the clickhouse `user_files` folder, then using the client to import them:

```
$ docker cp cuBB:/opt/nvidia/cuBB/pyaerial/notebooks/data/fh.parquet .
$ docker cp cuBB:/opt/nvidia/cuBB/pyaerial/notebooks/data/fapi.parquet .
$ sudo cp *.parquet ./ch_data/user_files/
```

A clickhouse client is needed to interact with the server. To download it and run it do the following:

```
curl https://clickhouse.com/ | sh
./clickhouse client

aerial@aerial-gnb:~$ ./clickhouse client
ClickHouse client version 24.3.1.1159 (official build).
Connecting to localhost:9000 as user default.
Connected to ClickHouse server version 24.3.1.

aerial-gnb :)
```

This is the clickhouse client prompt. Use the client to import the sample data into the clickhouse server using these commands:

```
aerial-gnb :) create table fh ENGINE = MergeTree primary key TsTainS settings allow_
↪ nullable_key=1 as select * from file('fh.parquet',Parquet)
aerial-gnb :) create table fapi ENGINE = MergeTree primary key TsTainS settings allow_
↪ nullable_key=1 as select * from file('fapi.parquet',Parquet)
```

Now check that they have been imported:

```
aerial-gnb :) select table, formatReadableSize(sum(bytes)) as size from system.parts
↪ group by table
```

The output will look similar to this:

```
SELECT
  `table`,
  formatReadableSize(sum(bytes)) AS size
FROM system.parts
GROUP BY `table`

Query id: 95451ea7-6ea9-4eec-b297-15de78036ada



| table | size     |
|-------|----------|
| fh    | 5.55 MiB |
| fapi  | 3.88 KiB |


```

You now have three slots of PUSCH transmissions from 5-6 real UEs received by two cells loaded in the database and can run the example notebooks.

## 3.8.2. Database Queries

To show some information about the entries (rows) you can run the following at the clickhouse client prompt:

Show counts of transmissions for all RNTIs

```
aerial-gnb :) select rnti, count(*) from fapi group by rnti
```

Output:

```
SELECT
  rnti,
  count(*)
FROM fapi
GROUP BY rnti
```

Query id: 603141a2-bc02-4950-8e9e-1d3f366263c6

rnti	count()
1624	3
20000	3
20216	3
47905	2
53137	2
57375	3
62290	3

Show select information from all rows of the fapi table

```
aerial-rf-gnb :) from fapi select TsTaiNs,SFN,Slot,nUEs,rbStart,rbSize,tbCrcStatus,
↪CQI order by TsTaiNs,rbStart
```

Output:

```
SELECT
  TsTaiNs,
  SFN,
  Slot,
  nUEs,
  rbStart,
  rbSize,
  tbCrcStatus,
  CQI
FROM fapi
ORDER BY
  TsTaiNs ASC,
  rbStart ASC
```

Query id: f42d9192-1de1-4cc6-b3eb-932b22ecab3e

	TsTaiNs	SFN	Slot	nUEs	rbStart	rbSize	tbCrcStatus	CQI
↪	2024-07-19 10:42:46.272000000	391	4	7	0	8	1	
↪	-7.352562							
↪	2024-07-19 10:42:46.272000000	391	4	7	0	5	0	
↪	31.75534							

(continues on next page)

(continued from previous page)

2024-07-19 10:42:46.272000000	391	4	7	5	5	0
↪ 30.275444						
2024-07-19 10:42:46.272000000	391	4	7	10	5	0
↪ 31.334328						
2024-07-19 10:42:46.272000000	391	4	7	15	5	0
↪ 30.117304						
2024-07-19 10:42:46.272000000	391	4	7	20	5	0
↪ 29.439499						
2024-07-19 10:42:46.272000000	391	4	7	25	248	0
↪ 25.331459						
2024-07-19 10:42:47.292000000	493	4	6	0	8	1
↪ -7.845479						
2024-07-19 10:42:47.292000000	493	4	6	0	5	0
↪ 29.412682						
2024-07-19 10:42:47.292000000	493	4	6	5	5	0
↪ 30.186537						
2024-07-19 10:42:47.292000000	493	4	6	10	5	0
↪ 30.366463						
2024-07-19 10:42:47.292000000	493	4	6	15	5	0
↪ 29.590645						
2024-07-19 10:42:47.292000000	493	4	6	20	253	0
↪ 28.494812						
2024-07-19 10:42:48.212000000	585	4	6	0	8	1
↪ -8.030928						
2024-07-19 10:42:48.212000000	585	4	6	0	5	0
↪ 31.359173						
2024-07-19 10:42:48.212000000	585	4	6	5	5	0
↪ 30.353489						
2024-07-19 10:42:48.212000000	585	4	6	10	5	0
↪ 29.3033						
2024-07-19 10:42:48.212000000	585	4	6	15	5	0
↪ 28.298597						
2024-07-19 10:42:48.212000000	585	4	6	20	253	0
↪ 26.621593						

19 rows in set. Elapsed: 0.002 sec.

Show start times of fh table

```
aerial-rf-gnb :) from fh select TsTaiNs,TsSwNs,SFN,Slot,CellId,nUEs
```

Output:

```
SELECT
  TsTaiNs,
  TsSwNs,
  SFN,
  Slot,
  CellId,
  nUEs
FROM fh
```

Query id: 6926d88e-6e9c-4818-b127-aef96913cfc0

TsTaiNs	TsSwNs	SFN	Slot	CellId	nUEs
2024-07-19 10:42:46.272000000	2024-07-19 10:42:46.273113183	391	4	41	

(continues on next page)

(continued from previous page)

```

↪      7
2024-07-19 10:42:46.272000000  2024-07-19 10:42:46.273113183  391    4    51
↪      7
2024-07-19 10:42:47.292000000  2024-07-19 10:42:47.293139202  493    4    41
↪      6
2024-07-19 10:42:47.292000000  2024-07-19 10:42:47.293139202  493    4    51
↪      6
2024-07-19 10:42:48.212000000  2024-07-19 10:42:48.213139622  585    4    41
↪      6
2024-07-19 10:42:48.212000000  2024-07-19 10:42:48.213139622  585    4    51
↪      6

```

6 rows in set. Elapsed: 0.002 sec.

### 3.8.3. Fresh Data

The database of IQ samples grows quite quickly. If you want to get fresh data on every run, you can automatically remove the tables by uncommenting the following lines in `cuPHY-CP/data_lakes/data_lakes.cpp`:

```
//dbClient->Execute("DROP TABLE IF EXISTS fapi");
//dbClient->Execute("DROP TABLE IF EXISTS fh");
```

### 3.8.4. Dropping Data

You can manually drop all of the data from the database with these commands:

```
aerial-gnb :) drop table fh
aerial-gnb :) drop table fapi
```

### 3.8.5. Notes and Known Limitations

Currently datalake converts complex half floating point values to floats in c++ which takes ~2ms per cell. During that time, when samples are being inserted into the database, PUSCH notifications can be missed and a note will be printed in the phy log:

```
[CTL.DATA_LAKE] Notify not called for 39.4 dbInsert busy
```



---

# Chapter 4. pyAerial

PyAerial provides a Python API towards the 5G signal processing functionality included in the Aerial cuPHY library.

## 4.1. Overview

As 6G research gains momentum, and with many new technologies in its purview, one thing is clear, AI/ML will feature prominently in the next generation RAN. It will play a pivotal role in realizing all parts of the network infrastructure from the radio units, baseband processing, the network core including system management, orchestration and dynamic optimization processes. GPU hardware, together with programming frameworks will be essential to realize this vision of a software defined native-AI communication infrastructure.

The application of AI/ML in the physical layer has in particular been a hot research topic. There is a lot of emphasis on neural network architectures and optimization strategies mostly performed in the context of simulation. The next step for the research community and commercial system developers is to bring AI/ML applied in layer-1 to reality in over-the-air real-time testbeds and operator-network scale systems.

This is where pyAerial enters the picture. pyAerial is a Python library of physical layer components that can be used as part of the workflow in taking a design from simulation to real-time operation. It helps with end-to-end verification of a neural network integration into a PHY pipeline and helps bridge the gap from the world of training and simulation in TensorFlow/PyTorch to real-time operation in an over-the-air testbed.

The pyAerial library provides a Python-callable bit-accurate GPU-accelerated library for all of the signal processing CUDA kernels in the NVIDIA cuBB layer-1 PDSCH and PUSCH pipelines. In other words, the pyAerial Python classes behave in a numerically identical manner to the kernels employed in cuBB because a pyAerial class employs the exact same CUDA code as the corresponding cuBB kernel: it is the CUDA kernel but with a Python API.

Using pyAerial library components complete layer-1 pipelines can be composed in Python. User code or inference engines, from NVIDIA TensorRT, or custom CUDA code, can be included in the datapath as shown in the lower part of Figure 1. This rapid prototyping design and verification flow is used for dataplane functional performance evaluation. It is a step in the workflow for verifying a physical layer design prior to deployment in a real-time over-the-air GPU base station.

pyAerial can also be used in conjunction with the NVIDIA data collection platform *Aerial Data Lake*. An Aerial Data Lake database consists of RF samples from a 7.2x fronthaul interface together with L2 meta-information to enable database search and query operations. A pyAerial pipeline can access samples from Aerial Data Lake database using the Data Lake Python APIs, and transform that data into training data for any function in the pipeline. Figure 2 illustrates data ingress from a Data Lake

database into a pyAerial pipeline and using standard Python file I/O to generate training data for a soft de-mapper.

### 4.1.1. Key Features

pyAerial has the following key features:

#### **Feature 1: Productive Python for rapid prototyping of layer-1 pipelines**

pyAerial library components are CUDA kernels with Python bindings. The productive environment of Python permits the rapid assembly of signal processing pipelines in Python. All of the analytic and visualization aspects of Python can be used for performance characterization, signal visualization and debugging.

#### **Feature 2: Simulate machine learning in the physical layer before over-the-air operation**

With the goal of going from model training and simulation in TensorFlow or PyTorch to real-time over-the-air operation, pyAerial provides a convenient way to verify, evaluate and benchmark your physical layer prior to deployment in an OTA testbed.

#### **Feature 3: Fast simulation with CUDA optimized kernels**

pyAerial library components are CUDA under the hood. Simulation is fast on a GPU. When you are simulating the coding chain, including for example an LDPC decoder, optimized CUDA code is implementing these computationally heavy functions.

#### **Feature 4: Generate data sets for any node in layer-1 uplink or downlink pipeline**

pyAerial is designed to be used in conjunction with the NVIDIA data collection platform *Aerial Data Lake*. pyAerial can access RF samples in a Data Lake database and transform those samples into training data for all of the signal processing functions in and uplink or downlink pipeline.

#### **Feature 5: Bit accurate simulation**

Because pyAerial is Python running on CUDA, the performance you observe in BLER and other characterization metrics is what is identical to the performance of the real-time over-the-air system.

### 4.1.2. Target Audience

Industry and university researchers and developers looking to bring machine learning to the physical layer with the end goal of benchmarking on over-the-air testbeds like NVIDIA ARC-OTA or other GPU-based base stations.

### 4.1.3. Value Proposition

Fast bit-accurate GPU accelerated simulation of neural-network downlink and uplink signal processing pipelines. Rapid prototyping and functional verification of a real-time layer-1 in preparation for real-time deployment. Convenient Python environment aids debugging and provides easy access to all nodes in the pipeline for visualization and analysis. Easy to use Python environment for producing BLER and other statistics of interest for a real-time bit-accurate GPU layer-1 implementation. Transform RF sample captures for over-the-air captures into data for training layer-1 functions or compositions of multiple functions.



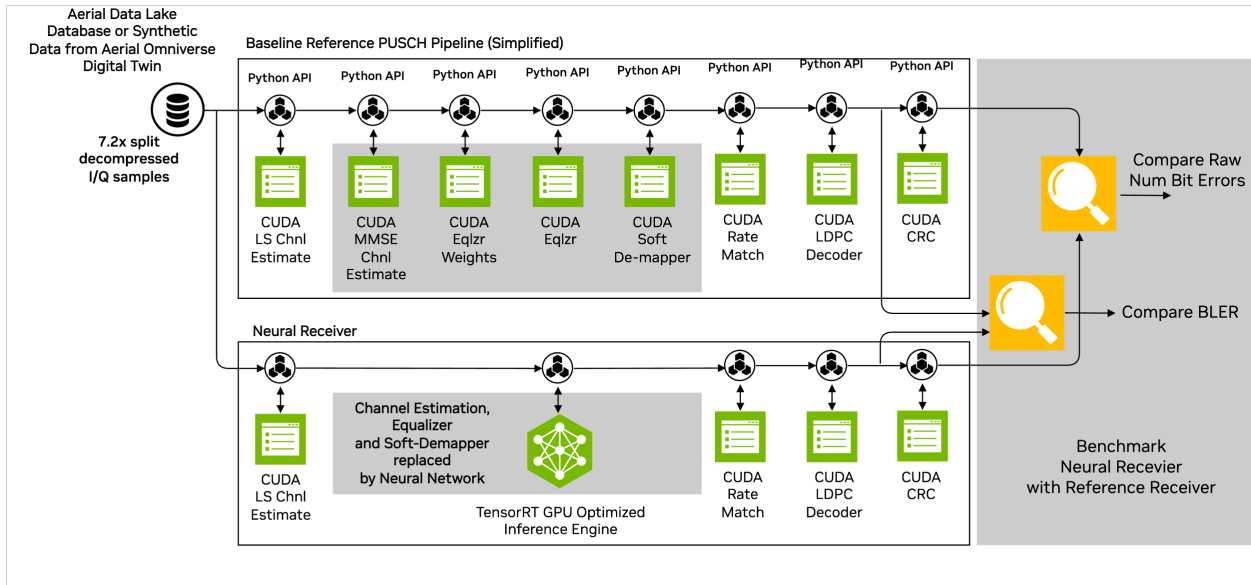


Fig. 1: Figure 1: Using pyAerial to verify a neural pipeline context of a full uplink pipeline. This is one of the verification steps to moving to real-time operation over-the-air on a GPU base station.

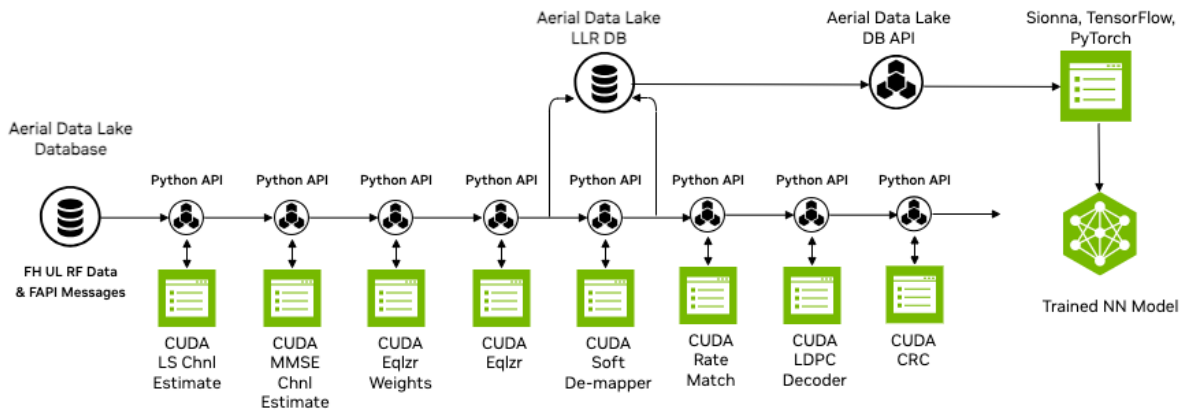


Fig. 2: Figure 2: pyAerial is used in conjunction with the NVIDIA data collection platform *Aerial Data Lake* to build training data sets for any node in the layer-1 downlink or uplink signal processing pipeline. The example shows a Data Lake database of over-the-air samples transformed into training data for a neural network soft de-mapper, using pyAerial. Data gets extracted at the input and output of the de-mapper, and stored in the database.

## 4.1.4. Release Notes

### Release version: 24-3

- ▶ New in this release:
  - ▶ CSI-RS transmission pipeline
  - ▶ RSRP and pre- and post-equalizer SINR estimation
  - ▶ Carrier frequency offset and timing advance estimation
  - ▶ CRC checking
  - ▶ OFDM fading channel simulation
  - ▶ Support of multiple UE groups for PUSCH receiver pipeline and its components
  - ▶ An improved API to PUSCH receiver pipeline and its components
- ▶ Supported configurations:
  - ▶ AX800, A100X and A100 GPUs with the x86 platform.
    - ▶ CUDA Toolkit: 12.6.2
    - ▶ GPU Driver: 560.35.03
  - ▶ Limited support on the Grace Hopper platform: The pyAerial Python package is supported, but the container does not include TensorFlow or Sionna. Thus, for example only the Aerial Data Lake example notebooks can be run on the Grace Hopper platform.
- ▶ Complete list of supported features:
  - ▶ Python API to the cuPHY library. This includes currently the following features:
    - ▶ PUSCH receiver pipeline
    - ▶ PDSCH transmission pipeline
    - ▶ CSI-RS transmission pipeline
    - ▶ Channel estimation (note: The RKHS algorithm supported by cuPHY is currently not exposed through the pyAerial API)
    - ▶ Noise and interference estimation
    - ▶ Channel equalization and soft demapping
    - ▶ RSRP and pre- and post-equalizer SINR estimation
    - ▶ Carrier frequency offset and timing advance estimation
    - ▶ LDPC encoding
    - ▶ LDPC decoding
    - ▶ LDPC rate matching
    - ▶ CRC checking
    - ▶ SRS channel estimation
    - ▶ TensorRT inference engine
  - ▶ OFDM fading channel simulation

## 4.2. Getting Started with pyAerial

### 4.2.1. Pre-requisites

Running pyAerial requires its own container, which also contains machine learning tools commonly used together with pyAerial:

- ▶ **NVIDIA Sionna** (version 0.19.0)
- ▶ **NVIDIA TensorRT** (version 10.6.0)
- ▶ **TensorFlow** (version 2.15.1)

To create and launch the pyAerial container, the following are needed:

- ▶ **NVIDIA Aerial CUDA-Accelerated RAN container**, see instructions [here](#).
- ▶ **Docker** installation, see instructions [here](#).
- ▶ **HPC Container Maker (HPCCM)** installation

The source code needs to be copied from the NVIDIA Aerial CUDA-Accelerated RAN container to a directory outside the container. The source code can be copied into the cuBB directory as follows (for example, see the note below):

```
docker run --rm -d --name cuBB <container image file>
docker cp cuBB:/opt/nvidia/cuBB cuBB
docker stop cuBB
cd cuBB
```

#### Note

The first command above can be omitted if the container is already running. Similarly, the stop command can be omitted if one wishes to keep the cuBB container running. The above example is showing one way of copying the source code from within the container into a directory outside the container.

The HPC Container Maker can be installed as follows:

```
pip install hpccm
```

### 4.2.2. Installing pyAerial

Once the above pre-requisites are fulfilled, the pyAerial container is built using the following script:

```
export cuBB_SDK=`pwd`
AERIAL_BASE_IMAGE=<container image file> $cuBB_SDK/pyaerial/container/build.sh
```

The container can then be launched using the following script:

```
$cuBB_SDK/pyaerial/container/run.sh
```

Once the container is running, pyAerial can be built and installed as follows (these commands are issued inside the pyAerial container):

```
cd $cuBB_SDK
cmake -Bbuild -GNinja -DCMAKE_TOOLCHAIN_FILE=cuPHY/cmake/toolchains/native -DNVIPC_
↪FMTLOG_ENABLE=OFF
cmake --build build -t _pycuphy pycuphycpp
./pyaerial/scripts/install_dev_pkg.sh
```

### Note

Note that pyAerial, similarly to Aerial cuPHY, is by default built for GPUs with compute capabilities 8.0 or 9.0, and these are also what pyAerial has been tested against. There is no guarantee that pyAerial will work correctly with other GPUs. However, pyAerial can be built for other compute capabilities with an additional cmake option, for example for CC 8.9:

```
cmake -Bbuild -GNinja -DCMAKE_TOOLCHAIN_FILE=cuPHY/cmake/toolchains/native -DNVIPC_
↪FMTLOG_ENABLE=OFF -DCMAKE_CUDA_ARCHITECTURES="89"
```

## 4.2.3. Testing the installation

To test that the installation works, the example Jupyter notebooks can be run as described below. Alternatively, the unit tests can be run as follows:

```
$cuBB_SDK/pyaerial/scripts/run_unit_tests.sh
```

Note : Unit tests are based on Aerial CUDA-Accelerated RAN test vectors. Those need to be mounted within the pyAerial container, and environment variable TEST\_VECTOR\_DIR set to point to the test vector directory. Refer to the Aerial CUDA-Accelerated RAN documentation on how to generate the test vectors.

One simple way to test the installation is to run (within the pyAerial container):

```
python3 -c "import aerial"
```

which should pass without errors.

## 4.2.4. Running the example Jupyter notebooks

NVIDIA pyAerial contains a number of example notebooks in Jupyter notebook format. The Jupyter notebooks can be run interactively within the pyAerial container using JupyterLab. This is done by starting a JupyterLab server as follows:

```
cd $cuBB_SDK/pyaerial/notebooks
jupyter lab --ip=0.0.0.0
```

and then pointing the browser to the given address. Note that the Aerial Data Lake notebooks require the example database to be created first. Refer to Aerial Data Lake documentation on how to start the clickhouse server and create the example database.

Pre-executed versions of the notebooks are found here: [Examples of Using pyAerial](#).

## 4.3. Examples of Using pyAerial

We provide a number of examples of using NVIDIA pyAerial for GPU-accelerated 5G NR signal processing, and for machine learning experiments. The examples are in Jupyter notebook format. The notebooks here are pre-executed, but they can be also interactively run following the instructions in [Getting Started with pyAerial](#).

### Note

Note that when running the notebooks, exceptions are not always displayed in Jupyter notebooks the way that it would be if a python script had been run, so in some cases it can be easier to convert the notebook to a script and run that. This can be done as follows:

```
jupyter nbconvert --to script <notebook_name>.ipynb
```

To interact with the data and code in place, specific lines can be debugged by adding `breakpoint()` inline.

### 4.3.1. Running a PUSCH link simulation

The first example shows how to use pyAerial for modeling 5G NR compliant PUSCH transmission and reception. In this example, the whole PUSCH pipeline is modeled within pyAerial, using the cuPHY library as a backend for GPU acceleration.

The notebook shows two ways of running the PUSCH receiver pipeline: In the first, the user only needs to make a single call using the Python API, and the whole PUSCH receiver is run. In the other, the PUSCH receiver pipeline is split into its different receiver components, each called separately using the Python API. This approach enables replacing any of the PUSCH receiver components for example by an AI/ML model, and benchmarking that against the conventional receiver.

NVIDIA Sionna is used in the example for radio channel modeling.

#### 4.3.1.1 Using pyAerial to run a PUSCH link simulation

This example shows how to use the pyAerial cuPHY Python bindings to run a PUSCH link simulation. PUSCH transmitter is emulated by PDSCH transmission with properly chosen parameters, that way making it a 5G NR compliant PUSCH transmission. Building a PUSCH receiver using pyAerial is demonstrated in two ways, first by using a fully fused, complete, PUSCH receiver called from Python using just a single function call. The same is then achieved by building the complete PUSCH receiver using individual separate Python function calls to individual PUSCH receiver components.

The NVIDIA [Sionna](#) library is utilized for simulating the radio channel based on 3GPP channel models.

```
[1]: # Check platform.
import platform
if platform.machine() != 'x86_64':
    raise SystemExit("Unsupported platform!")
```

#### 4.3.1.1.1 Imports

```
[2]: %matplotlib widget
import datetime
from collections import defaultdict
import os
os.environ["CUDA_VISIBLE_DEVICES"] = "0"
os.environ['TF_CPP_MIN_LOG_LEVEL'] = "3" # Silence TensorFlow.

import numpy as np
import matplotlib.pyplot as plt
import sionna
import tensorflow as tf

from aerial.phy5g.pdsch import PdschTx
from aerial.phy5g.pusch import PuschRx
from aerial.phy5g.algorithms import ChannelEstimator
from aerial.phy5g.algorithms import ChannelEqualizer
from aerial.phy5g.algorithms import NoiseIntfEstimator
from aerial.phy5g.algorithms import Demapper
from aerial.phy5g.ldpc import get_mcs
from aerial.phy5g.ldpc import get_tb_size
from aerial.phy5g.ldpc import random_tb
from aerial.phy5g.ldpc import LdpcDeRateMatch
from aerial.phy5g.ldpc import LdpcDecoder
from aerial.phy5g.ldpc import CrcChecker
from aerial.phy5g.types import PuschLdpcKernelLaunch
from aerial.phy5g.params import PuschConfig
from aerial.phy5g.params import PuschUeConfig
from aerial.util.cuda import get_cuda_stream
from simulation_monitor import SimulationMonitor

# Configure the notebook to use only a single GPU and allocate only as much memory as
# needed.
# For more details, see https://www.tensorflow.org/guide/gpu.
gpus = tf.config.list_physical_devices('GPU')
tf.config.experimental.set_memory_growth(gpus[0], True)
```

#### 4.3.1.1.2 Parameters

Set simulation parameters, numerology, PUSCH parameters and channel parameters here.

```
[3]: # Simulation parameters.
esno_db_range = np.arange(-5.4, -4.4, 0.2)
num_slots = 10000
min_num_tb_errors = 250

# Numerology and frame structure. See TS 38.211.
num_ofdm_symbols = 14
fft_size = 4096
cyclic_prefix_length = 288
subcarrier_spacing = 30e3
num_guard_subcarriers = (410, 410)
num_slots_per_frame = 20
```

(continues on next page)

(continued from previous page)

```

# System/gNB configuration
num_tx_ant = 1          # UE antennas
num_rx_ant = 2          # gNB antennas
cell_id = 41           # Physical cell ID
enable_pusch_tdi = 0    # Enable time interpolation for equalizer coefficients
eq_coeff_algo = 1       # Equalizer algorithm

# PUSCH configuration
rnti = 1234             # UE RNTI
scid = 0                # DMRS scrambling ID
data_scid = 0           # Data scrambling ID
layers = 1              # Number of layers
mcs_index = 2           # MCS index as per TS 38.214 table. Note: Es/No range may
↳need to be changed too to get meaningful results.
mcs_table = 0           # MCS table index
dmrs_ports = 1          # Used DMRS port.
start_prb = 0           # Start PRB index.
num_prbs = 273          # Number of allocated PRBs.
start_sym = 2           # Start symbol index.
num_symbols = 12        # Number of symbols.
dmrs_scrm_id = 41       # DMRS scrambling ID
dmrs_syms = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] # Indicates which symbols are
↳used for DMRS.
dmrs_max_len = 1
dmrs_add_ln_pos = 0
num_dmrs_cdm_grps_no_data = 2
mod_order, code_rate = get_mcs(mcs_index, mcs_table+1) # Different indexing for MCS
↳table.
tb_size = get_tb_size(mod_order, code_rate, dmrs_syms, num_prbs, start_sym, num_
↳symbols, layers) # TB size in bits

# Channel parameters
carrier_frequency = 3.5e9 # Carrier frequency in Hz.
delay_spread = 100e-9     # Nominal delay spread in [s]. Please see the CDL
↳documentation
# about how to choose this value.
link_direction = "uplink"
channel_model = "Rayleigh" # Channel model: Suitable values:
# "Rayleigh" - Rayleigh block fading channel model (sionna.
↳channel.RayleighBlockFading)
# "CDL-x", where x is one of ["A", "B", "C", "D", "E"] - for
↳3GPP CDL channel models
# as per TR 38.901.
speed = 0.8333           # UE speed [m/s]. The direction of travel will chosen
↳randomly within the x-y plane.

```

#### 4.3.1.1.3 Create the PUSCH pipelines

As mentioned, PUSCH transmission is emulated here by the PDSCH transmission chain. Note that the static cell parameters and static PUSCH parameters are given upon creating the PUSCH transmission/reception objects. Dynamically (per slot) changing parameters are however set when actually running the transmission/reception, see further below.

```
[4]: # PUSCH configuration object. Note that default values are used for some parameters
# not given here.
pusch_ue_config = PuschUeConfig(
    scid=scid,
    layers=layers,
    dmrs_ports=dmrs_ports,
    rnti=rnti,
    data_scid=data_scid,
    mcs_table=mcs_table,
    mcs_index=mcs_index,
    code_rate=int(code_rate * 10),
    mod_order=mod_order,
    tb_size=tb_size // 8
)
# Note that this is a list. One UE group only in this case.
pusch_configs = [PuschConfig(
    ue_configs=[pusch_ue_config],
    num_dmrs_cdm_grps_no_data=num_dmrs_cdm_grps_no_data,
    dmrs_scrm_id=dmrs_scrm_id,
    start_prb=start_prb,
    num_prbs=num_prbs,
    dmrs_syms=dmrs_syms,
    dmrs_max_len=dmrs_max_len,
    dmrs_add_ln_pos=dmrs_add_ln_pos,
    start_sym=start_sym,
    num_symbols=num_symbols
)]

pusch_tx = PdschTx(
    cell_id=cell_id,
    num_rx_ant=num_tx_ant,
    num_tx_ant=num_tx_ant,
)

# This is the fully fused PUSCH receiver chain.
pusch_rx = PuschRx(
    cell_id=cell_id,
    num_rx_ant=num_rx_ant,
    num_tx_ant=num_rx_ant,
    enable_pusch_tdi=enable_pusch_tdi,
    eq_coeff_algo=eq_coeff_algo,
    # To make this equal separate PUSCH Rx components configuration:
    ldpc_kernel_launch=PuschLdpcKernelLaunch.PUSCH_RX_LDPC_STREAM_SEQUENTIAL
)

# The PUSCH receiver chain built from separately called pyAerial Python components is
↪ defined here.
class PuschRxSeparate:
    """PUSCH receiver class.
```

(continues on next page)



(continued from previous page)

*This class encapsulates the whole PUSCH receiver chain built using pyAerial components.*

```

def __init__(self,
             num_rx_ant,
             enable_pusch_tdi,
             eq_coeff_algo):
    """Initialize the PUSCH receiver."""
    self.cuda_stream = get_cuda_stream()

    # Build the components of the receiver.
    self.channel_estimator = ChannelEstimator(
        num_rx_ant=num_rx_ant,
        cuda_stream=self.cuda_stream
    )
    self.channel_equalizer = ChannelEqualizer(
        num_rx_ant=num_rx_ant,
        enable_pusch_tdi=enable_pusch_tdi,
        eq_coeff_algo=eq_coeff_algo,
        cuda_stream=self.cuda_stream
    )
    self.noise_intf_estimator = NoiseIntfEstimator(
        num_rx_ant=num_rx_ant,
        eq_coeff_algo=eq_coeff_algo,
        cuda_stream=self.cuda_stream
    )
    self.derate_match = LdpcDeRateMatch(
        enable_scrambling=True,
        cuda_stream=self.cuda_stream
    )
    self.decoder = LdpcDecoder(cuda_stream=self.cuda_stream)
    self.crc_checker = CrcChecker(cuda_stream=self.cuda_stream)

def run(self,
       rx_slot,
       slot,
       pusch_configs):
    """Run the receiver."""
    # Channel estimation.
    ch_est = self.channel_estimator.estimate(
        rx_slot=rx_slot,
        slot=slot,
        pusch_configs=pusch_configs
    )

    # Noise and interference estimation.
    lw_iny, noise_var_pre_eq = self.noise_intf_estimator.estimate(
        rx_slot=rx_slot,
        channel_est=ch_est,
        slot=slot,
        pusch_configs=pusch_configs
    )

    # Channel equalization and soft demapping. The first return value are the LLRs,
    # second are the equalized symbols. We only want the LLRs now.

```

(continues on next page)

(continued from previous page)

```

llrs = self.channel_equalizer.equalize(
    rx_slot=rx_slot,
    channel_est=ch_est,
    lw_inv=lw_inv,
    noise_var_pre_eq=noise_var_pre_eq,
    pusch_configs=pusch_configs
)[0]

# (De)rate matching.
coded_blocks = self.derate_match.derate_match(
    input_llrs=llrs,
    pusch_configs=pusch_configs
)

# LDPC decoding.
code_blocks = self.decoder.decode(
    input_llrs=coded_blocks,
    pusch_configs=pusch_configs
)

# Code block desegmentation, CRC removal, CRC checking.
decoded_tbs, tb_crcs = self.crc_checker.check_crc(
    input_bits=code_blocks,
    pusch_configs=pusch_configs
)

return tb_crcs, decoded_tbs

pusch_rx_separate = PuschRxSeparate(
    num_rx_ant=num_rx_ant,
    enable_pusch_tdi=enable_pusch_tdi,
    eq_coeff_algo=eq_coeff_algo
)

```

#### 4.3.1.1.4 Channel generation using Sionna

Simulating the transmission through the radio channel takes advantage of the channel model implementations available in NVIDIA Sionna. In Sionna, the transmission can be simulated directly in frequency domain by defining a resource grid. In our case, reference signal patterns and data carrying resource elements are defined elsewhere within the Aerial code, hence we define resource grid as a simple dummy grid containing only data symbols.

See also: [Sionna documentation](#)

```

[5]: # Define the resource grid.
resource_grid = sionna.ofdm.ResourceGrid(
    num_ofdm_symbols=num_ofdm_symbols,
    fft_size=fft_size,
    subcarrier_spacing=subcarrier_spacing,
    num_tx=1,
    num_streams_per_tx=1,
    cyclic_prefix_length=cyclic_prefix_length,
    num_guard_carriers=num_guard_subcarriers,
    dc_null=False,
    pilot_pattern=None,

```

(continues on next page)

(continued from previous page)

```

    pilot_ofdm_symbol_indices=None
)
resource_grid_mapper = sionna.ofdm.ResourceGridMapper(resource_grid)
remove_guard_subcarriers = sionna.ofdm.RemoveNulledSubcarriers(resource_grid)

# Define the antenna arrays.
ue_array = sionna.channel.tr38901.Antenna(
    polarization="single",
    polarization_type="V",
    antenna_pattern="38.901",
    carrier_frequency=carrier_frequency
)
gnb_array = sionna.channel.tr38901.AntennaArray(
    num_rows=1,
    num_cols=int(num_rx_ant/2),
    polarization="dual",
    polarization_type="cross",
    antenna_pattern="38.901",
    carrier_frequency=carrier_frequency
)

if channel_model == "Rayleigh":
    ch_model = sionna.channel.RayleighBlockFading(
        num_rx=1,
        num_rx_ant=num_rx_ant,
        num_tx=1,
        num_tx_ant=num_tx_ant
    )

elif "CDL" in channel_model:
    cdl_model = channel_model[-1]

    # Configure a channel impulse reponse (CIR) generator for the CDL model.
    ch_model = sionna.channel.tr38901.CDL(
        cdl_model,
        delay_spread,
        carrier_frequency,
        ue_array,
        gnb_array,
        link_direction,
        min_speed=speed
    )
else:
    raise ValueError(f"Invalid channel model {channel_model}!")

channel = sionna.channel.OFDMChannel(
    ch_model,
    resource_grid,
    add_awgn=True,
    normalize_channel=True,
    return_channel=False
)

def apply_channel(tx_tensor, No):
    """Transmit the Tx tensor through the radio channel."""
    # Add batch and num_tx dimensions that Sionna expects and reshape.

```

(continues on next page)

(continued from previous page)

```

tx_tensor = tf.transpose(tx_tensor, (2, 1, 0))
tx_tensor = tf.reshape(tx_tensor, (1, -1))[None, None]
tx_tensor = resource_grid_mapper(tx_tensor)
rx_tensor = channel((tx_tensor, No))
rx_tensor = remove_guard_subcarriers(rx_tensor)
rx_tensor = rx_tensor[0, 0]
rx_tensor = tf.transpose(rx_tensor, (2, 1, 0))
return rx_tensor

```

#### 4.3.1.1.5 Run the actual simulation

Here we loop across the Es/No range, and simulate a number of slots for each Es/No value. A single transport block is simulated within a slot. The simulation starts over from the next Es/No value when a minimum number of transport block errors is reached.

```

[6]: cases = ["Fused", "Separate"]
monitor = SimulationMonitor(cases, esno_db_range)

# Loop the Es/No range.
bler = []
for esno_db in esno_db_range:
    monitor.step(esno_db)
    num_tb_errors = defaultdict(int)

    # Run multiple slots and compute BLER.
    for slot_idx in range(num_slots):
        slot_number = slot_idx % num_slots_per_frame

        # Get random transport block.
        tb_input = random_tb(mod_order, code_rate, dmrs_syms, num_prbs, start_sym,
↪ num_symbols, layers)

        # Transmit PUSCH. This is where we set the dynamically changing parameters.
        # Input parameters are given as lists as the interface supports multiple UEs.
        tx_tensor = pusch_tx.run(
            tb_inputs=[tb_input],           # Input transport block in bytes.
            num_ues=1,                     # We simulate only one UE here.
            slot=slot_number,              # Slot number.
            num_dmrs_cdm_grps_no_data=num_dmrs_cdm_grps_no_data,
            dmrs_scrm_id=dmrs_scrm_id,     # DMRS scrambling ID.
            start_prb=start_prb,          # Start PRB index.
            num_prbs=num_prbs,            # Number of allocated PRBs.
            dmrs_syms=dmrs_syms,          # List of binary numbers indicating which
↪ symbols are DMRS.
            start_sym=start_sym,           # Start symbol index.
            num_symbols=num_symbols,       # Number of symbols.
            scids=[scid],                  # DMRS scrambling ID.
            layers=[layers],               # Number of layers (transmission rank).
            dmrs_ports=[dmrs_ports],       # DMRS port(s) to be used.
            rntis=[rnti],                  # UE RNTI.
            data_scids=[data_scid],        # Data scrambling ID.
            code_rates=[code_rate],        # Code rate x 1024.
            mod_orders=[mod_order]         # Modulation order.
        )[0]

```

(continues on next page)

(continued from previous page)

```

# Channel transmission using TF and Sionna.
No = pow(10., -esno_db / 10.)
rx_tensor = apply_channel(tx_tensor, No)
rx_tensor = np.array(rx_tensor)

# Run the fused PUSCH receiver.
# Note that this is where we set the dynamically changing parameters.
tb_crcs, tbs = pusch_rx.run(
    rx_slot=rx_tensor,
    slot=slot_number,
    pusch_configs=pusch_configs
)
num_tb_errors["Fused"] += int(np.array_equal(tbs[0], tb_input) == False)

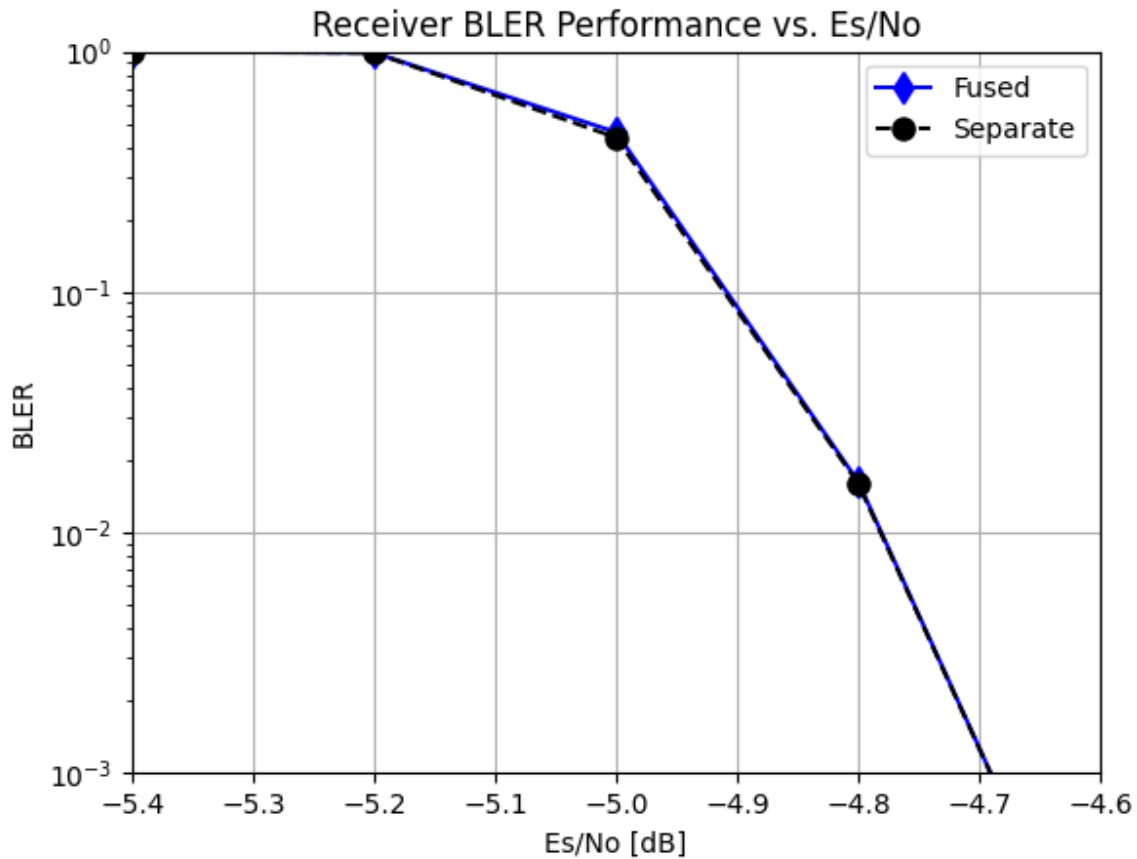
# Run the receiver built from separately called components.
tb_crcs, tbs = pusch_rx_separate.run(
    rx_slot=rx_tensor,
    slot=slot_number,
    pusch_configs=pusch_configs
)
num_tb_errors["Separate"] += int(np.array_equal(tbs[0], tb_input) == False)

monitor.update(num_tbs=slot_idx + 1, num_tb_errors=num_tb_errors)
if (np.array(list(num_tb_errors.values())) >= min_num_tb_errors).all():
    break # Next Es/No value.

monitor.finish_step(num_tbs=slot_idx + 1, num_tb_errors=num_tb_errors)
monitor.finish()

```

Es/No (dB)	TBs	Fused		Separate		ms/TB
		TB Errors	BLER	TB Errors	BLER	
-5.40	250	250	1.0000	250	1.0000	120.2
-5.20	253	250	0.9881	250	0.9881	117.6
-5.00	570	263	0.4614	250	0.4386	117.8
-4.80	10000	163	0.0163	160	0.0160	118.0
-4.60	10000	1	0.0001	1	0.0001	119.2



### 4.3.2. LDPC encoding-decoding chain

The second example gives an example of using cuPHY's GPU accelerated 5G NR LDPC encoding and decoding chain (including also rate matching) modules through the pyAerial Python API. The encoding/decoding modules are expected to be useful for example in AI/ML model validation when implementing some parts of the receiver using machine learning.

#### 4.3.2.1 Using pyAerial for LDPC encoding-decoding chain

This example shows how to use the pyAerial Python bindings to run 5G NR LDPC encoding, rate matching and decoding. Information bits, i.e. a transport block, get segmented into code blocks, LDPC encoded and rate matched onto the available time-frequency resources (resource elements), all following TS 38.212 precisely. The bits are then transmitted over an AWGN channel using QPSK modulation. At the receiver side, log likelihood ratios are extracted from the received symbols, (de)rate matching is performed and LDPC decoder is run to get the transmitted information bits. Finally, the code blocks are concatenated back into a received transport block.

pyAerial utilizes the cuPHY library underneath for all components. In this example however, CRCs are just random blocks of bits in this example as we can compare the transmitted and received bits directly to compute block error rates.

The NVIDIA [Sionna](#) library is utilized for simulating the radio channel.

```
[1]: # Check platform.
import platform
if platform.machine() != 'x86_64':
    raise SystemExit("Unsupported platform!")
```

#### 4.3.2.1.1 Imports

```
[2]: %matplotlib widget
from cuda import cudart
from collections import defaultdict
import os
os.environ["CUDA_VISIBLE_DEVICES"] = "0"
os.environ['TF_CPP_MIN_LOG_LEVEL'] = "3" # Silence TensorFlow.

import numpy as np
import sionna
import tensorflow as tf

from aerial.phy5g.ldpc import LdpcEncoder
from aerial.phy5g.ldpc import LdpcDecoder
from aerial.phy5g.ldpc import LdpcRateMatch
from aerial.phy5g.ldpc import LdpcDeRateMatch
from aerial.phy5g.ldpc import CrcChecker
from aerial.phy5g.ldpc import get_mcs
from aerial.phy5g.ldpc import random_tb
from aerial.phy5g.ldpc import code_block_segment
from aerial.phy5g.ldpc import get_crc_len
from simulation_monitor import SimulationMonitor

# Configure the notebook to use only a single GPU and allocate only as much memory as
# ↪needed.
# For more details, see https://www.tensorflow.org/guide/gpu.
gpus = tf.config.list_physical_devices('GPU')
tf.config.experimental.set_memory_growth(gpus[0], True)

from tensorflow.python.ops.numpy_ops import np_config
np_config.enable_numpy_behavior()
```

#### 4.3.2.1.2 Parameters

Set simulation parameters, some numerology parameters, enable/disable scrambling etc.

```
[3]: # Simulation parameters.
esno_db_range = np.arange(8.1, 8.8, 0.1)
num_slots = 10000
min_num_tb_errors = 250

# Numerology and frame structure. See TS 38.211.
```

(continues on next page)

(continued from previous page)

```

num_prb = 100          # Number of allocated PRBs. This is used to compute the
↳transport block     # as well as the rate matching length.
start_sym = 0         # P×SCH start symbol
num_symbols = 14      # Number of symbols in a slot.
num_slots_per_frame = 20 # Number of slots in a single frame.
num_layers = 1
dmrs_sym = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]

# Rate matching procedure includes scrambling if this flag is set.
enable_scrambling = True

# The scrambling initialization value is computed as per TS 38.211
# using the RNTI and data scrambling ID.
rnti = 20000          # UE RNTI
data_scid = 41        # Data scrambling ID
cinit = (rnti << 15) + data_scid

rv = 0                # Redundancy version
mcs = 9               # MCS index as per TS 38.214 table. Note: Es/No range may
↳need to be changed too to get meaningful results.

mod_order, code_rate = get_mcs(mcs)
code_rate /= 1024.

```

#### 4.3.2.1.3 Create the LDPC coding chain objects

The LDPC coding chain objects are created here. This includes the following:

- ▶ LdpcEncoder which takes the information bits, i.e. the transport block, segmented into code blocks as its input, and outputs encoded code blocks.
- ▶ LdpcRateMatch which takes encoded code blocks as its input and outputs a rate matched (and optionally scrambled) stream of bits.
- ▶ LdpcDerateMatch which takes the received stream of log-likelihood ratios (LLRs) as its input and outputs derate matched code blocks of LLRs which can be fed to the LDPC decoding. This block performs also descrambling if scrambling is enabled in the pipeline.
- ▶ LdpcDecoder which takes the output of LDPC derate matching and decodes the LLRs into code blocks that can then be further concatenated into a received transport block.
- ▶ CrcChecker which takes the output of the LDPC decoding block, removed code block CRCs, concatenates code blocks into a full transport block, and finally checks and removed the transport block CRC.

All components are based on TS 38.212 and thus can be used for transmitting/receiving 5G NR compliant bit streams.

Also the Sionna channel components and modulation mapper are created here.

```

[4]: # Create also the CUDA stream that running the objects requires.
      cudart.cudaSetDevice(0)
      cuda_stream = cudart.cudaStreamCreate()[1]
      cudart.cudaStreamSynchronize(cuda_stream)

```

(continues on next page)



(continued from previous page)

```

# Create the Aerial Python LDPC objects.
ldpc_encoder = LdpcEncoder(cuda_stream=cuda_stream)
ldpc_decoder = LdpcDecoder(cuda_stream=cuda_stream)
ldpc_rate_match = LdpcRateMatch(enable_scrambling=enable_scrambling, cuda_stream=cuda_
↳stream)
ldpc_derate_match = LdpcDeRateMatch(enable_scrambling=enable_scrambling, cuda_
↳stream=cuda_stream)
crc_checker = CrcChecker(cuda_stream=cuda_stream)

# Create the Sionna modulation mapper/demapper and the AWGN channel.
mapper = sionna.mapping.Mapper("qam", mod_order)
demapper = sionna.mapping.Demapper("app", "qam", mod_order)
channel = sionna.channel.AWGN()

```

#### 4.3.2.1.4 Main simulation loop

```

[5]: case = "LDPC decoding perf."
monitor = SimulationMonitor([case], esno_db_range)

# Loop the Es/No range.
for esno_db in esno_db_range:

    monitor.step(esno_db)
    num_tb_errors = defaultdict(int)

    # Run multiple slots and compute BLER.
    for slot_idx in range(num_slots):
        slot_number = slot_idx % num_slots_per_frame

        # Generate a random transport block (in bits).
        transport_block = random_tb(
            mod_order=mod_order,
            code_rate=code_rate * 1024,
            dmrs_syms=dmrs_sym,
            num_prbs=num_prb,
            start_sym=start_sym,
            num_symbols=num_symbols,
            num_layers=num_layers,
            return_bits=True
        )
        tb_size = transport_block.shape[0]

        # Attach a CRC. This is emulated to get the TB size with CRC right, however the
↳CRC is in this case just random
        # bits as we are comparing the transmitted and received bits directly to get
↳the BLER (instead of doing an actual
        # CRC check).
        crc_length = get_crc_len(tb_size)
        crc = np.random.randint(0, 1, size=crc_length, dtype=np.uint8)
        transport_block = np.concatenate((transport_block, crc))

        # Code block segmentation happens here. Note: This is just Python at the moment.
        code_blocks = code_block_segment(tb_size, transport_block, code_rate)

```

(continues on next page)

(continued from previous page)

```

# Run the LDPC encoding. The LDPC encoder takes a K x C array as its input,
↪where K is the number of bits per code
# block and C is the number of code blocks. Its output is N x C where N is the
↪number of coded bits per code block.
# If there is more than one code block, a code block CRC (random in this case
↪as we do not need an actual CRC) is
# attached to
coded_bits = ldpc_encoder.encode(
    input_data=code_blocks,
    tb_size=tb_size,
    code_rate=code_rate,
    redundancy_version=rv
)

# Run rate matching. This needs rate matching length as its input, meaning the
↪number of bits that can be
# transmitted within the allocated resource elements. The input data is fed as
↪32-bit floats.
num_data_sym = (np.array(dmrs_sym[start_sym:start_sym + num_symbols]) == 0).
↪sum()
rate_match_len = num_data_sym * num_prb * 12 * num_layers * mod_order
rate_matched_bits = ldpc_rate_match.rate_match(
    input_data=coded_bits,
    tb_size=tb_size,
    code_rate=code_rate,
    rate_match_len=rate_match_len,
    mod_order=mod_order,
    num_layers=num_layers,
    redundancy_version=rv,
    cinit=cinit
)

# Map the bits to symbols and transmit through an AWGN channel. All this in
↪Sionna.
rate_matched_bits = rate_matched_bits[:, 0]

no = sionna.utils.ebndb2no(esno_db, num_bits_per_symbol=1, coderate=1)
tx_symbols = mapper(rate_matched_bits[None])
rx_symbols = channel([tx_symbols, no])
llr = -1. * demapper([rx_symbols, no])[0, :].numpy()[:, None]

# Run receiver side (de)rate matching. The input is the received array of bits
↪directly, and the output
# is a NumPy array of size N x C of log likelihood ratios, represented as 32-
↪bit floats. Descrambling
# is also performed here in case scrambling is enabled.
derate_matched_bits = ldpc_derate_match.derate_match(
    input_llrs=[llr],
    tb_sizes=[tb_size],
    code_rates=[code_rate],
    rate_match_lengths=[rate_match_len],
    mod_orders=[mod_order],
    num_layers=[num_layers],
    redundancy_versions=[rv],
    ndis=[1],

```

(continues on next page)

(continued from previous page)

```

        cinits=[cinit]
    )

    # Run LDPC decoding. The decoder takes the derate matching output as its input
    ↪and returns
    decoded_bits = ldpc_decoder.decode(
        input_llrs=derate_matched_bits,
        tb_sizes=[tb_size],
        code_rates=[code_rate],
        redundancy_versions=[rv],
        rate_match_lengths=[rate_match_len]
    )

    # Combine code blocks into a transport block. CRC ignored as it was just random
    ↪bits in this example.
    decoded_tb, _ = crc_checker.check_crc(
        input_bits=decoded_bits,
        tb_sizes=[tb_size],
        code_rates=[code_rate]
    )
    decoded_tb = np.unpackbits(decoded_tb) # To bits.

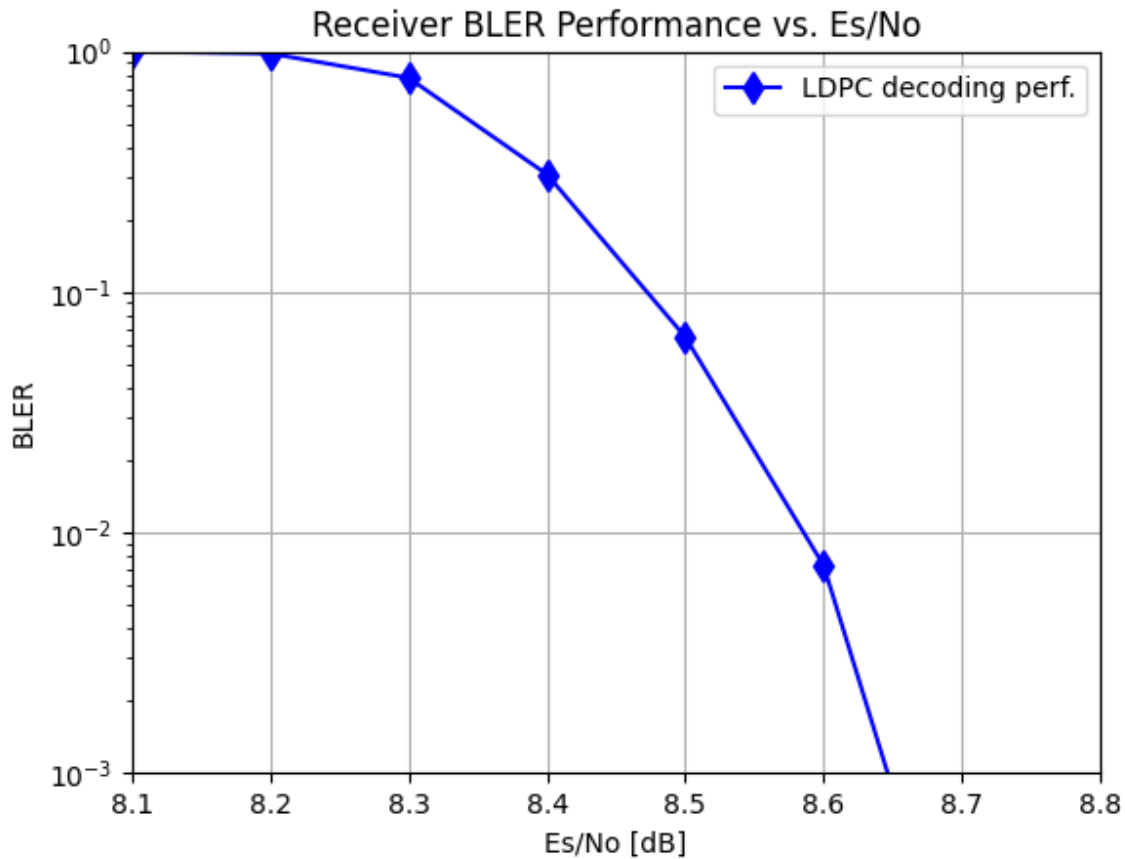
    tb_error = not np.array_equal(decoded_tb, transport_block[:-crc_length])
    num_tb_errors[case] += tb_error
    monitor.update(num_tbs=slot_idx + 1, num_tb_errors=num_tb_errors)
    if (np.array(list(num_tb_errors.values()))) >= min_num_tb_errors).all():
        break # Next Es/No value.

    monitor.finish_step(num_tbs=slot_idx + 1, num_tb_errors=num_tb_errors)
monitor.finish()

```

LDPC decoding perf.

Es/No (dB)	TBs	TB Errors	BLER	ms/TB
8.10	250	250	1.0000	19.9
8.20	255	250	0.9804	15.7
8.30	322	250	0.7764	15.7
8.40	811	250	0.3083	15.7
8.50	3846	250	0.0650	15.6
8.60	10000	73	0.0073	15.4
8.70	10000	1	0.0001	15.4
8.80	10000	0	0.0000	15.3



### 4.3.3. Dataset generation by simulation

#### 4.3.3.1 Using pyAerial for data generation by simulation

This notebook generates a fully 5G NR compliant PUSCH/PDSCH dataset using NVIDIA cuPHY through its Python bindings in pyAerial for PUSCH/PDSCH slot generation and NVIDIA Sionna for radio channel modeling. PUSCH/PDSCH slots get generated and transmitted through different radio channels. Usually, in order to make models as generalizable as possible, it is desirable to train the models with as wide variety of different channel models as possible. This notebook enables generation of a dataset containing samples generated with a number of different channel models, including e.g. those used by 3GPP, as well as with different MCS classes and other transmission parameters.

```
[1]: # Check platform.  
import platform  
if platform.machine() != 'x86_64':  
    raise SystemExit("Unsupported platform!")
```

#### 4.3.3.1.1 Imports

```
[2]: import warnings
warnings.filterwarnings('ignore')

import itertools
import os
os.environ["CUDA_VISIBLE_DEVICES"] = "0"
os.environ['TF_CPP_MIN_LOG_LEVEL'] = "3" # Silence TensorFlow.

import numpy as np
import pandas as pd
import sionna
import tensorflow as tf
from tqdm.notebook import tqdm

from aerial.phy5g.pdsch import PdschTx
from aerial.phy5g.ldpc.util import get_mcs, random_tb
from aerial.util.fapi import dmrs_bit_array_to_fapi
from aerial.util.data import PuschRecord
from aerial.util.data import save_pickle

# This is for Sionna and pyAerial to coexist on the same GPU:
# Configure the notebook to use only a single GPU and allocate only as much memory as
# needed.
# For more details, see https://www.tensorflow.org/guide/gpu.
gpus = tf.config.list_physical_devices('GPU')
tf.config.experimental.set_memory_growth(gpus[0], True)
```

#### 4.3.3.1.2 Dataset generation parameters

The parameters used to generate the dataset are modified here. Note that some parameters are given as lists, meaning that multiple values may be given for those parameters. Typically one would like the training dataset to be as diverse as possible in order to make the models generalize well to various channel conditions and to different transmission parameters.

```
[3]: # This is the target directory. It gets created if it does not exist.
dataset_dir = 'data/example_simulated_dataset/QPSK'
os.makedirs(dataset_dir, exist_ok=True)

# Number of samples is divided roughly evenly between the options below.
num_samples = 12000

# A list of channel models: Suitable values:
# "Rayleigh" - Rayleigh block fading channel model (sionna.channel.RayleighBlockFading)
# "CDL-x", where x is one of ["A", "B", "C", "D", "E"] - for 3GPP CDL channel models
# as per TR 38.901.
channel_models = ["CDL-D"]

# Speeds to include in the dataset
# This is UE speed in m/s. The direction of travel will be chosen randomly within the x-
# y plane.
speeds = [0.8333]

# Delay spreads to include in the dataset.
```

(continues on next page)

(continued from previous page)

```

# This is the nominal delay spread in [s]. Please see the CDL documentation
# about how to choose this value.
delay_spreads = [100e-9]

# A list of MCS indices (as per TS 38.214) to include in the dataset.
# MCS table value refers to TS 38.214 as follows:
# 1: TS38.214, table 5.1.3.1-1.
# 2: TS38.214, table 5.1.3.1-2.
# 3: TS38.214, table 5.1.3.1-3.
mcss = [1] # 1, 10, 19 used for QPSK, 16QAM and 64QAM, respectively.
mcs_table = 2

# Es/No values to include in the dataset.
# esnos = [9.0, 9.25, 9.5, 9.75, 10.0, 10.25, 10.5, 10.75, 11.0] # MCS 19
# esnos = [-0.5, -0.25, 0.0, 0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0, 2.25, 2.5, 2.
↪75, 3.0] # MCS 10
enos = [-7.75, -7.5, -7.25, -7.0, -6.75, -6.5] # MCS 1

# These are fixed for the dataset.
num_tx_ant = 1
num_rx_ant = 4
cell_id = 41
carrier_frequency = 3.5e9 # Carrier frequency in Hz.
link_direction = "uplink"
layers = 1
rnti = 20001
scid = 0
data_scid = 41
dmrs_port = 1
dmrs_position = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]
start_sym = 0
num_symbols = 14
start_prb = 0
num_prbs = 273

# Numerology and frame structure. See TS 38.211.
mu = 1
num_ofdm_symbols = 14
fft_size = 4096
cyclic_prefix_length = 288
subcarrier_spacing = 30e3
num_guard_subcarriers = (410, 410)
num_slots_per_frame = 20

```

#### 4.3.3.1.3 Channel generation

Radio channel generation is done using [NVIDIA Sionna](#).

```

[4]: class Channel(sionna.channel.OFDMChannel):
    def __init__(self,
                 link_direction,
                 channel_model,
                 num_tx_ant,
                 num_rx_ant,

```

(continues on next page)

(continued from previous page)

```

        carrier_frequency,
        delay_spread,
        speed,
        resource_grid):
self.resource_grid = resource_grid
self.resource_grid_mapper = sionna.ofdm.ResourceGridMapper(resource_grid)
self.remove_guard_subcarriers = sionna.ofdm.RemoveNulledSubcarriers(resource_
→grid)

# Define the antenna arrays.
ue_array = sionna.channel.tr38901.Antenna(
    polarization="single",
    polarization_type="V",
    antenna_pattern="38.901",
    carrier_frequency=carrier_frequency
)
gnb_array = sionna.channel.tr38901.AntennaArray(
    num_rows=1,
    num_cols=int(num_rx_ant/2),
    polarization="dual",
    polarization_type="cross",
    antenna_pattern="38.901",
    carrier_frequency=carrier_frequency
)

if channel_model == "Rayleigh":
    ch_model = sionna.channel.RayleighBlockFading(
        num_rx=1,
        num_rx_ant=num_rx_ant,
        num_tx=1,
        num_tx_ant=num_tx_ant
    )

elif "CDL" in channel_model:
    cdl_model = channel_model[-1]

    # Configure a channel impulse reponse (CIR) generator for the CDL model.
    ch_model = sionna.channel.tr38901.CDL(
        cdl_model,
        delay_spread,
        carrier_frequency,
        ue_array,
        gnb_array,
        link_direction,
        min_speed=speed
    )
else:
    raise ValueError(f"Invalid channel model {channel_model}!")

super().__init__(
    ch_model,
    resource_grid,
    add_awgn=True,
    normalize_channel=True,
    return_channel=False
)

```

(continues on next page)

(continued from previous page)

```

def __call__(self, tx_tensor, No):
    # Add batch and num_tx dimensions that Sionna expects and reshape.
    tx_tensor = tf.transpose(tx_tensor, (2, 1, 0))
    tx_tensor = tf.reshape(tx_tensor, (1, -1))[None, None]
    tx_tensor = self.resource_grid_mapper(tx_tensor)
    rx_tensor = super().__call__((tx_tensor, No))
    rx_tensor = self.remove_guard_subcarriers(rx_tensor)
    rx_tensor = rx_tensor[0, 0]
    rx_tensor = tf.transpose(rx_tensor, (2, 1, 0))
    return rx_tensor

# Define the resource grid.
resource_grid = sionna.ofdm.ResourceGrid(
    num_ofdm_symbols=num_ofdm_symbols,
    fft_size=fft_size,
    subcarrier_spacing=subcarrier_spacing,
    num_tx=1,
    num_streams_per_tx=1,
    cyclic_prefix_length=cyclic_prefix_length,
    num_guard_carriers=num_guard_subcarriers,
    dc_null=False,
    pilot_pattern=None,
    pilot_ofdm_symbol_indices=None
)

```

#### 4.3.3.1.4 PDSCH transmitter

This creates the PDSCH transmitter. However due to the symmetry of 5G NR PDSCH and PUSCH, this may be used to generate also PUSCH frames with certain parameterization. In this notebook this is used as a PUSCH transmitter to generate uplink slots.

```

[5]: pxsch_tx = PdschTx(
    cell_id=cell_id,
    num_rx_ant=num_tx_ant,
    num_tx_ant=num_tx_ant,
)

```

#### 4.3.3.1.5 Dataset generation

The actual dataset generation is done here. The different channel, SNR and MCS parameters are swept through, with a number of samples per parameterization chosen such that the total number of samples will be close to the desired number.

The PxSCH transmitter created above is used to generate a Tx frame. This Tx frame is then fed through the Sionna-generated radio channel. The resulting data is recorded in a Parquet file containing PUSCH records following roughly the [Small Cell Forum FAPI specification](#) format.

```

[6]: num_cases = len(channel_models) * len(esnos) * len(speeds) * len(delay_spreads) *
    ↪ len(mcscs)
    num_samples_per_param = num_samples // num_cases

```

(continues on next page)



(continued from previous page)

```

# loop different channel models, speeds, delay spreads, MCS levels etc.
pusch_records = []
for (channel_model, esno, speed, delay_spread, mcs) in \
    (pbar := tqdm(itertools.product(channel_models, esnos, speeds, delay_spreads,
    ↪mcss), total=num_cases)):

    status_str = f"Generating... ({channel_model} | {esno} dB | {speed} m/s | {delay_
    ↪spread} s | MCS {mcs})"
    pbar.set_description(status_str)

    # Create the channel model.
    channel = Channel(
        link_direction=link_direction,
        channel_model=channel_model,
        num_tx_ant=num_tx_ant,
        num_rx_ant=num_rx_ant,
        carrier_frequency=carrier_frequency,
        delay_spread=delay_spread,
        speed=speed,
        resource_grid=resource_grid
    )

    for sample in range(num_samples_per_param):
        # Generate the dataframe.
        slot_number = sample % num_slots_per_frame

        # Get modulation order and coderate.
        mod_order, coderate = get_mcs(mcs, mcs_table)
        tb_input = random_tb(mod_order, coderate, dmrs_position, num_prbs, start_sym,
        ↪num_symbols, layers)

        # Transmit PXSCH. This is where we set the dynamically changing parameters.
        # Input parameters are given as lists as the interface supports multiple UEs.
        tx_tensor = pxsch_tx.run(
            tb_inputs=[tb_input],           # Input transport block in bytes.
            num_ues=1,                     # We simulate only one UE here.
            slot=slot_number,              # Slot number.
            dmrs_syms=dmrs_position,       # List of binary numbers indicating which
        ↪symbols are DMRS.
            start_sym=start_sym,          # Start symbol index.
            num_symbols=num_symbols,       # Number of symbols.
            scids=[scid],                  # DMRS scrambling ID.
            layers=[layers],               # Number of layers (transmission rank).
            dmrs_ports=[dmrs_port],        # DMRS port(s) to be used.
            rntis=[rnti],                  # UE RNTI.
            data_scids=[data_scid],        # Data scrambling ID.
            code_rates=[coderate],         # Code rate
            mod_orders=[mod_order]         # Modulation order
        )[0]

        # Channel transmission and noise.
        No = pow(10., -esno / 10.)
        rx_tensor = channel(tx_tensor, No)
        rx_tensor = np.array(rx_tensor)

        # Save the sample.

```

(continues on next page)

(continued from previous page)

```

rx_iq_data_filename = "rx_iq-{}_esno{}_speed{}_ds{}_mcs{}_-{}.pkl".
↪format(channel_model, esno, speed, delay_spread, mcs, sample)
rx_iq_data_fullpath = os.path.join(dataset_dir, rx_iq_data_filename)
save_pickle(data=rx_tensor, filename=rx_iq_data_fullpath)

# Save noise power and SNR data as user data.
user_data_filename = "user_data-{}_esno{}_speed{}_ds{}_mcs{}_-{}.pkl".
↪format(channel_model, esno, speed, delay_spread, mcs, sample)
user_data_fullpath = os.path.join(dataset_dir, user_data_filename)
user_data = dict(
    snr=esno,
    noise_var=No
)
save_pickle(data=user_data, filename=user_data_fullpath)

pusch_record = PuschRecord(
    # SCF FAPI 10.02 UL_TTI.request message parameters:
    pduIdx=0,
    SFN=(sample // num_slots_per_frame) % 1023,
    Slot=slot_number,
    nPDUs=1,
    RachPresent=0,
    nULSCH=1,
    nULCCH=0,
    nGroup=1,
    PDUSize=0,
    pduBitmap=1,
    RNTI=rnti,
    Handle=0,
    BWPSize=273,
    BWPStart=0,
    SubcarrierSpacing=mu,
    CyclicPrefix=0,
    targetCodeRate=coderate * 10,
    qamModOrder=mod_order,
    mcsIndex=mcs,
    mcsTable=mcs_table - 1, # Different indexing
    TransformPrecoding=1, # Disabled.
    dataScramblingId=data_scid,
    nrOfLayers=1,
    ulDmrsSymbPos=dMrs_bit_array_to_fapi(dMrs_position),
    dMrsConfigType=0,
    ulDmrsScramblingId=cell_id,
    puschIdentity=cell_id,
    SCID=scid,
    numDmrsCdmGrpsNoData=2,
    dMrsPorts=1, # Note that FAPI uses a different format compared to cuPHY.
    resourceAlloc=1,
    rbBitmap=np.array(36 * [0]),
    rbStart=0,
    rbSize=273,
    VRBtoPRBMapping=0,
    FrequencyHopping=0,
    txDirectCurrentLocation=0,
    uplinkFrequencyShift7p5khz=0,
    StartSymbolIndex=start_sym,

```

(continues on next page)

(continued from previous page)

```

    NrOfSymbols=num_symbols,
    puschData=None,
    puschUci=None,
    puschPtrs=None,
    dftsOfdm=None,
    Beamforming=None,

    # SCF FAPI 10.02 RxData.indication message parameters:
    HarqID=0,
    PDULen=len(tb_input),
    UL_CQI=255, # Set to invalid 0xFF.
    TimingAdvance=0,
    RSSI=65535, # Set to invalid 0xFFFF.
    macPdu=tb_input,

    TbCrcStatus=0,
    NumCb=0,
    CbCrcStatus=None,

    rx_iq_data_filename=rx_iq_data_filename,
    user_data_filename=user_data_filename,

    errInd = ""
)
pusch_records.append(pusch_record)

print("Saving...")
df_filename = os.path.join(dataset_dir, "12_metadata.parquet")
df = pd.DataFrame.from_records(pusch_records, columns=PuschRecord._fields)
df.to_parquet(df_filename, engine="pyarrow")
print("All done!")

0%|          | 0/6 [00:00<?, ?it/s]

Saving...
All done!

```

This notebook generates a fully 5G NR compliant PUSCH/PDSCH dataset using pyAerial. The cuPHY library is used through its Python bindings in pyAerial for PUSCH/PDSCH slot generation, and NVIDIA Sionna is used for radio channel modeling. The PUSCH/PDSCH slots get generated and transmitted through different radio channels.

The example stores the dataset for use in the consequent LLRNet examples. Equally well the data could be generated on the fly during simulation.

### 4.3.4. Dataset generation for LLRNet

In this example, pyAerial is used to generate a log-likelihood ratio dataset based on the PUSCH/PDSCH dataset generated in the previous example. Using pyAerial, the complete PUSCH receiver chain is formed, and LLR data is collected after the channel equalizer. The log-likelihood ratio data is used to train an LLRNet model in the next example. LLRNet, published in

Shental, J. Hoydis, “Machine LLRning: Learning to Softly Demodulate”, <https://arxiv.org/abs/1907.01512>

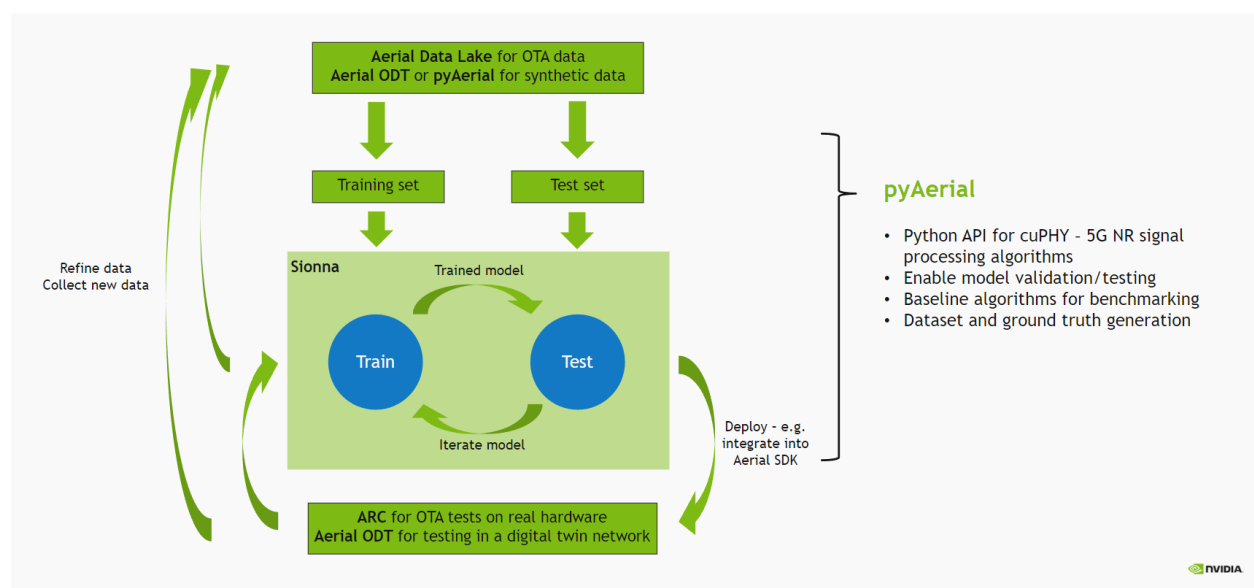
is a simple neural network model that takes equalizer outputs, i.e. the complex-valued equalized symbols, as its input and outputs the corresponding log-likelihood ratios (LLRs) for each bit, basically replacing the conventional soft demapper in the receiver chain.

### Note

This notebook requires that the former example in *Dataset generation by simulation* has been run first.

#### 4.3.4.1 LLRNet: Dataset generation

The wireless ML design flow using Aerial is depicted in the figure below.



In this notebook, we take data generated in the *Using pyAerial for data generation by simulation* example and generate a dataset for training LLRNet using pyAerial. **Note that the data is assumed to have been generated prior to running this notebook.**

LLRNet, published in

- O. Shental, J. Hoydis, “Machine LLRning: Learning to Softly Demodulate”, <https://arxiv.org/abs/1907.01512>

is a simple neural network model that takes equalizer outputs, i.e. the complex-valued equalized symbols, as its input and outputs the corresponding log-likelihood ratios (LLRs) for each bit. This model is used to demonstrate the whole ML design flow using Aerial, from capturing the data to deploying the model into 5G NR PUSCH receiver, replacing the conventional soft demapper in cuPHY. In this notebook a dataset is generated. We use pyAerial to call cuPHY functionality to get equalized symbols out for pre-captured/-simulated Rx data, as well as the corresponding log-likelihood ratios from a conventional soft demapper.

```
[1]: # Check platform.
import platform
if platform.machine() != 'x86_64':
    raise SystemExit("Unsupported platform!")
```

#### 4.3.4.1.1 Imports

```
[2]: import os
os.environ["CUDA_VISIBLE_DEVICES"] = "0"

from cuda import cudart
import numpy as np
import pandas as pd
from tqdm.notebook import tqdm
from IPython.display import Markdown
from IPython.display import display

from aerial.phy5g.algorithms import ChannelEstimator
from aerial.phy5g.algorithms import NoiseIntfEstimator
from aerial.phy5g.algorithms import ChannelEqualizer
from aerial.phy5g.algorithms import Demapper
from aerial.phy5g.ldpc import LdpcDeRateMatch
from aerial.phy5g.ldpc import LdpcDecoder
from aerial.phy5g.ldpc import CrcChecker
from aerial.phy5g.params import PuschConfig
from aerial.phy5g.params import PuschUeConfig
from aerial.util.data import PuschRecord
from aerial.util.data import load_pickle
from aerial.util.data import save_pickle
from aerial.util.fapi import dmrs_fapi_to_bit_array

import warnings
warnings.filterwarnings("error")
```

#### 4.3.4.1.2 Load the source data

The source data can be either real data collected from an over the air setup, or synthetic data generated by simulation.

**Note:** This notebook uses data generated using this notebook: [Using pyAerial for data generation by simulation](#), which needs to be run before this notebook.

```
[3]: # This is the source data directory which is assumed to contain the source data.
DATA_DIR = "data/"
source_dataset_dir = DATA_DIR + "example_simulated_dataset/QPSK/"
# This is the target dataset directory. It gets created if it does not exist.
target_dataset_dir = DATA_DIR + "example_llrnet_dataset/QPSK/"
os.makedirs(target_dataset_dir, exist_ok=True)

# Load the main data file.
try:
    df = pd.read_parquet(source_dataset_dir + "l2_metadata.parquet", engine="pyarrow")
except FileNotFoundError:
    display(Markdown("**Data not found - has example_simulated_dataset.ipynb been run? ↪**"))

print(f"Loaded {df.shape[0]} PUSCH records.")
```

Loaded 12000 PUSCH records.

#### 4.3.4.1.3 Dataset generation

Here, pyAerial is used to run channel estimation, noise/interference estimation and channel equalization to get the equalized symbols, corresponding to the LLRNet input, as well as the log-likelihood ratios, corresponding to the LLRNet target output.

```
[4]: cuda_stream = cudart.cudaStreamCreate()[1]

# Take modulation order from the first record. The assumption is that all
# entries have the same modulation order here.
mod_order = df.loc[0].qamModOrder
# These hard-coded too.
num_rx_ant = 2
enable_pusch_tdi = 1
eq_coeff_algo = 1

# Create the PUSCH Rx components for extracting the equalized symbols and log-
↳likelihood ratios.
channel_estimator = ChannelEstimator(
    num_rx_ant=num_rx_ant,
    cuda_stream=cuda_stream
)
noise_intf_estimator = NoiseIntfEstimator(
    num_rx_ant=num_rx_ant,
    eq_coeff_algo=eq_coeff_algo,
    cuda_stream=cuda_stream
)
channel_equalizer = ChannelEqualizer(
    num_rx_ant=num_rx_ant,
    eq_coeff_algo=eq_coeff_algo,
    enable_pusch_tdi=enable_pusch_tdi,
    cuda_stream=cuda_stream)
derate_match = LdpcDeRateMatch(enable_scrambling=True, cuda_stream=cuda_stream)
demapper = Demapper(mod_order=mod_order)
decoder = LdpcDecoder(cuda_stream=cuda_stream)
crc_checker = CrcChecker(cuda_stream=cuda_stream)

# Loop through the PUSCH records and create new ones.
pusch_records = []
tb_errors = []
snrs = []
for pusch_record in (pbar := tqdm(df.itertuples(index=False), total=df.shape[0])):

    pbar.set_description("Running cuPHY to get equalized symbols and log-likelihood
↳ratios...")

    tbs = len(pusch_record.macPdu)
    ref_tb = pusch_record.macPdu
    slot = pusch_record.Slot

    # Just making sure the hard-coded value is correct.
    assert mod_order == pusch_record.qamModOrder

    # Wrap the parameters in a PuschConfig structure.
    pusch_ue_config = PuschUeConfig(
        scid=pusch_record.SCID,
        layers=pusch_record.nrOfLayers,
```

(continues on next page)

(continued from previous page)

```

dmrs_ports=pusch_record.dmrPorts,
rnti=pusch_record.RNTI,
data_scid=pusch_record.dataScramblingId,
mcs_table=pusch_record.mcsTable,
mcs_index=pusch_record.mcsIndex,
code_rate=pusch_record.targetCodeRate,
mod_order=pusch_record.qamModOrder,
tb_size=len(pusch_record.macPdu)
)
# Note that this is a list. One UE group only in this case.
pusch_configs = [PuschConfig(
    ue_configs=[pusch_ue_config],
    num_dmrs_cdm_grps_no_data=pusch_record.numDmrsCdmGrpsNoData,
    dmrs_scrm_id=pusch_record.ulDmrsScramblingId,
    start_prb=pusch_record.rbStart,
    num_prbs=pusch_record.rbSize,
    dmrs_syms=dmrs_fapi_to_bit_array(pusch_record.ulDmrsSymbPos),
    dmrs_max_len=1,
    dmrs_add_ln_pos=1,
    start_sym=pusch_record.StartSymbolIndex,
    num_symbols=pusch_record.NrOfSymbols
)]

# Load received IQ samples.
rx_iq_data_filename = source_dataset_dir + pusch_record.rx_iq_data_filename
rx_slot = load_pickle(rx_iq_data_filename)
num_rx_ant = rx_slot.shape[2]

# Load user data.
user_data_filename = source_dataset_dir + pusch_record.user_data_filename
user_data = load_pickle(user_data_filename)

# Run the channel estimation (cuPHY).
ch_est = channel_estimator.estimate(
    rx_slot=rx_slot,
    slot=slot,
    pusch_configs=pusch_configs
)

# Run noise/interference estimation (cuPHY), needed for equalization.
lw_inv, noise_var_pre_eq = noise_intf_estimator.estimate(
    rx_slot=rx_slot,
    channel_est=ch_est,
    slot=slot,
    pusch_configs=pusch_configs
)

# Run equalization and mapping to log-likelihood ratios.
llrs, equalized_sym = channel_equalizer.equalize(
    rx_slot=rx_slot,
    channel_est=ch_est,
    lw_inv=lw_inv,
    noise_var_pre_eq=noise_var_pre_eq,
    pusch_configs=pusch_configs
)
ree_diag_inv = channel_equalizer.ree_diag_inv[0]

```

(continues on next page)

(continued from previous page)

```

ree_diag_inv = np.transpose(ree_diag_inv[...], 0], (1, 2, 0)).reshape(ree_diag_inv.
↪shape[1], -1)

# Just pick one (first) symbol from each PUSCH record for the LLRNet dataset.
# This is simply to reduce the size of the dataset - training LLRNet does not
# require a lot of data.
user_data["llrs"] = llrs[0][:mod_order, 0, :, 0]
user_data["eq_syms"] = equalized_sym[0][0, :, 0]
map_llrs = demapper.demap(equalized_sym[0][0, :, 0], ree_diag_inv[0, ...])
user_data["map_llrs"] = map_llrs

# Save pickle files for the target dataset.
rx_iq_data_fullpath = target_dataset_dir + pusch_record.rx_iq_data_filename
user_data_fullpath = target_dataset_dir + pusch_record.user_data_filename
save_pickle(data=rx_slot, filename=rx_iq_data_fullpath)
save_pickle(data=user_data, filename=user_data_fullpath)

pusch_records.append(pusch_record)

#####
↪###
# Run through the rest of the receiver pipeline to verify that this was legit LLR
↪data.

# De-rate matching and descrambling.
coded_blocks = derate_match.derate_match(
    input_llrs=llrs,
    pusch_configs=pusch_configs
)

# LDPC decoding of the derate matched blocks.
code_blocks = decoder.decode(
    input_llrs=coded_blocks,
    pusch_configs=pusch_configs
)

# Combine the code blocks into a transport block.
tb, _ = crc_checker.check_crc(
    input_bits=code_blocks,
    pusch_configs=pusch_configs
)

tb_errors.append(not np.array_equal(tb[:tbs], ref_tb[:tbs]))
snrs.append(user_data["snr"])

0%|          | 0/12000 [00:00<?, ?it/s]

```

```

[5]: print("Saving...")
df_filename = os.path.join(target_dataset_dir, "l2_metadata.parquet")
df = pd.DataFrame.from_records(pusch_records, columns=PuschRecord._fields)
df.to_parquet(df_filename, engine="pyarrow")
print("All done!")

```

```

Saving...
All done!

```



### 4.3.5. LLRNet model training

In this example, the LLR data from the previous example is used to train and validate an LLRNet model for computing log-likelihood ratios. The trained LLRNet is plugged in the PUSCH receiver chain, replacing the conventional soft demapper, and its performance is validated. The model also gets exported into ONNX format consumed by the NVIDIA TensorRT inference engine.

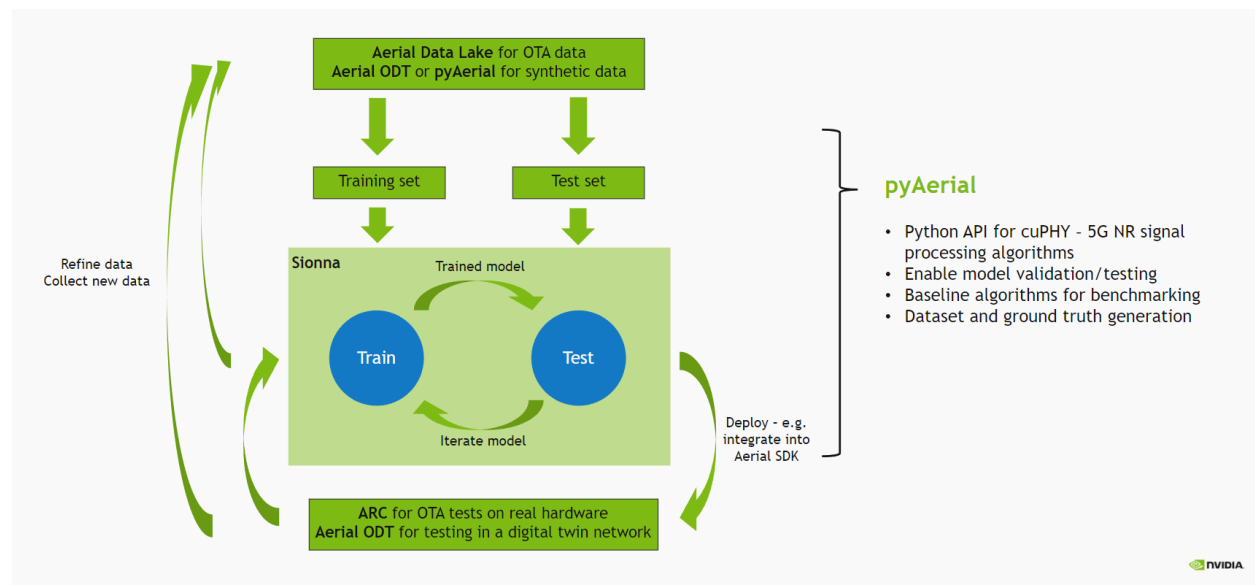
The example shows essentially how to use pyAerial for validating a component of the PUSCH receiver chain, and how to export a model in a format that is ready to be integrated in a real system.

#### Note

This notebook requires that the former example in [Dataset generation for LLRNet](#) has been run first - that generates the data for this notebook.

#### 4.3.5.1 LLRNet: Model training and testing

The wireless ML design flow using Aerial is depicted in the figure below.



In this notebook, we use the generated LLRNet data for training and validating LLRNet as part of the PUSCH receiver chain, implemented using pyAerial, with Aerial SDK cuPHY library working as the back-end. The LLRNet is plugged in the PUSCH receiver chain in place of the conventional soft demapper. So this notebook works as an example of using pyAerial for model validation.

Finally, the model is exported into a format consumed by the TensorRT inference engine that is used for integrating the model into Aerial SDK for testing the model with real hardware in an over the air environment.

**Note 1:** This notebook requires that the Aerial test vectors have been generated. The test vector directory is set below in `AERIAL_TEST_VECTOR_DIR` variable. **Note 2:** This notebook also requires that the notebook example on LLRNet dataset generation has been run first.

```
[1]: # Check platform.
import platform
if platform.machine() != 'x86_64':
    raise SystemExit("Unsupported platform!")
```

#### 4.3.5.1.1 Imports

```
[2]: %matplotlib widget
import os
os.environ["CUDA_VISIBLE_DEVICES"] = "0"
os.environ['TF_CPP_MIN_LOG_LEVEL'] = "3" # Silence TensorFlow.
os.environ["CUDA_MODULE_LOADING"] = "LAZY"

import cuda
import h5py as h5
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import tf2onnx
import onnx
from IPython.display import Markdown
from IPython.display import display

# PyAerial components
from aerial.phy5g.algorithms import ChannelEstimator
from aerial.phy5g.algorithms import ChannelEqualizer
from aerial.phy5g.algorithms import NoiseIntfEstimator
from aerial.phy5g.algorithms import Demapper
from aerial.phy5g.algorithms import TrtEngine
from aerial.phy5g.algorithms import TrtTensorPrms
from aerial.phy5g.ldpc import LdpcDeRateMatch
from aerial.phy5g.ldpc import LdpcDecoder
from aerial.phy5g.ldpc import CrcChecker
from aerial.phy5g.params import PuschConfig
from aerial.phy5g.params import PuschUeConfig
from aerial.util.cuda import get_cuda_stream
from aerial.util.data import load_pickle
from aerial.util.fapi import dmrs_fapi_to_bit_array

# Configure the notebook to use only a single GPU and allocate only as much memory as
# ↪ needed.
# For more details, see https://www.tensorflow.org/guide/gpu.
gpus = tf.config.list_physical_devices('GPU')
tf.config.experimental.set_memory_growth(gpus[0], True)
```

```
[3]: tb_errors = dict(aerial=dict(), llrnet=dict(), logmap=dict())
tb_count = dict(aerial=dict(), llrnet=dict(), logmap=dict())
```

```
[4]: # Dataset root directory.
DATA_DIR = "data/"
```

(continues on next page)

(continued from previous page)

```

# Aerial test vector directory.
AERIAL_TEST_VECTOR_DIR = "/mnt/cicd_tvsvs/develop/GPU_test_input/"

# LLRNet dataset directory.
dataset_dir = DATA_DIR + "example_llrnet_dataset/QPSK/"

# LLRNet model target path
llrnet_onnx_file = f"../models/llrnet.onnx"
llrnet_trt_file = f"../models/llrnet.trt"

# Training vs. testing SNR. Assume these exist in the dataset.
train_snr = [-7.75, -7.5, -7.25, -7.0, -6.75, -6.5]
test_snr = [-7.75, -7.5, -7.25, -7.0, -6.75, -6.5]

# Training, validation and test split in percentages if the same SNR is used for
# training and testing.
train_split = 45
val_split = 5
test_split = 50

# Training hyperparameters.
batch_size = 32
epochs = 5
step = tf.Variable(0, trainable=False)
boundaries = [350000, 450000]
values = [5e-4, 1e-4, 1e-5]
# values = [0.05, 0.01, 0.001]
learning_rate_fn = tf.keras.optimizers.schedules.PiecewiseConstantDecay(boundaries,
    ↪ values)
optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate_fn, weight_decay=1e-
    ↪ 4)
# optimizer = tf.keras.optimizers.experimental.SGD(learning_rate=0.05, weight_decay=1e-
    ↪ 4, momentum=0.9)

# Modulation order. LLRNet needs to be trained separately for each modulation order.
mod_order = 2

```

#### 4.3.5.1.2 Define the LLRNet model

The LLRNet model follows the original paper

- O. Shental, J. Hoydis, "Machine LLRning: Learning to Softly Demodulate", <https://arxiv.org/abs/1907.01512>

and is a very simple MLP model with a single hidden layer. It takes the equalized symbols in its input with the real and imaginary parts separated, and outputs soft bits (log-likelihood ratios) that can be further fed into LDPC (de)rate matching and decoding.

```

[5]: model = keras.Sequential(
    [
        layers.Dense(16, input_dim=2, activation="relu"),
        layers.Dense(8, activation="linear")
    ]
)
def loss(llr, predictions):

```

(continues on next page)

(continued from previous page)

```

mae = tf.abs(predictions[:, :mod_order] - llr)
mse = tf.reduce_mean(tf.square(mae))
return mse

```

#### 4.3.5.1.3 Training, validation and testing datasets

Here, the dataset gets loaded and split into training, validation and testing datasets, as well as put in the right format for the model.

```

[6]: # Load the main data file
try:
    df = pd.read_parquet(dataset_dir + "l2_metadata.parquet", engine="pyarrow")
except FileNotFoundError:
    display(Markdown("**Data not found - has llrnet_dataset_generation.ipynb been run?
    ↪**"))

# Query the entries for the selected modulation order.
df = df[df["qamModOrder"] == mod_order]

# Collect the dataset by SNR.
llrs = dict()
eq_syms = dict()
indices = dict()
for pusch_record in df.itertuples():
    user_data_filename = dataset_dir + pusch_record.user_data_filename
    user_data = load_pickle(user_data_filename)

    if user_data["snr"] not in llrs.keys():
        llrs[user_data["snr"]] = []
        eq_syms[user_data["snr"]] = []
        indices[user_data["snr"]] = []

    llrs[user_data["snr"]].append(user_data["map_llrs"])
    eq_syms[user_data["snr"]].append(user_data["eq_syms"])

    indices[user_data["snr"]].append(pusch_record.Index)

llr_train, llr_val = [], []
sym_train, sym_val = [], []
test_indices = []
for key in llrs.keys():
    llrs[key] = np.stack(llrs[key])
    eq_syms[key] = np.stack(eq_syms[key])

# Randomize the order.
permutation = np.arange(llrs[key].shape[0])
np.random.shuffle(permutation)
llrs[key] = llrs[key][permutation, ...]
eq_syms[key] = eq_syms[key][permutation, ...]
indices[key] = list(np.array(indices[key])[permutation])

# Separate real and imaginary parts of the symbols.
eq_syms[key] = np.stack((np.real(eq_syms[key]), np.imag(eq_syms[key])))

```

(continues on next page)

(continued from previous page)

```

num_slots = llrs[key].shape[0]
if key in train_snr and key in test_snr:
    num_train_slots = int(np.round(train_split / 100 * num_slots))
    num_val_slots = int(np.round(val_split / 100 * num_slots))
    num_test_slots = int(np.round(test_split / 100 * num_slots))
elif key in train_snr:
    num_train_slots = int(np.round(train_split / (train_split + val_split) * num_
↪slots))
    num_val_slots = int(np.round(val_split / (train_split + val_split) * num_
↪slots))
    num_test_slots = 0
elif key in test_snr:
    num_train_slots = 0
    num_val_slots = 0
    num_test_slots = num_slots
else:
    num_train_slots = 0
    num_val_slots = 0
    num_test_slots = 0

# Collect training/validation/testing sets.
llr_train.append(llrs[key][:num_train_slots, ...])
llr_val.append(llrs[key][num_train_slots:num_train_slots+num_val_slots, ...])
sym_train.append(eq_syms[key][:, :num_train_slots, ...])
sym_val.append(eq_syms[key][:, num_train_slots:num_train_slots+num_val_slots, ...
↪])
# Just indices for the test set.
test_indices += indices[key][num_train_slots:num_train_slots+num_
↪val_slots+num_test_slots]

llr_train = np.transpose(np.concatenate(llr_train, axis=0), (1, 0, 2))
llr_val = np.transpose(np.concatenate(llr_val, axis=0), (1, 0, 2))
sym_train = np.concatenate(sym_train, axis=1)
sym_val = np.concatenate(sym_val, axis=1)

# Fetch the total number of slots in each set.
num_train_slots = llr_train.shape[1]
num_val_slots = llr_val.shape[1]
num_test_slots = len(test_indices)

normalizer = 1.0 #np.sqrt(np.var(llr_train))
llr_train = llr_train / normalizer
llr_val = llr_val / normalizer

# Reshape into samples x mod_order array.
llr_train = llr_train.reshape(mod_order, -1).T
llr_val = llr_val.reshape(mod_order, -1).T
# Reshape into samples x 2 array.
sym_train = sym_train.reshape(2, -1).T
sym_val = sym_val.reshape(2, -1).T

print(f"Total number of slots in the training set: {num_train_slots}")
print(f"Total number of slots in the validation set: {num_val_slots}")
print(f"Total number of slots in the test set: {num_test_slots}")

```

Total number of slots in the training set: 5400  
 Total number of slots in the validation set: 600  
 Total number of slots in the test set: 6000

#### 4.3.5.1.4 Model training and validation

Model training is done using Keras here.

```
[7]: print("Training...")
model.compile(loss=loss, optimizer=optimizer, metrics=[loss])
model.fit(
    x=sym_train,
    y=llr_train,
    batch_size=batch_size,
    epochs=epochs,
    verbose=1,
    validation_data=(sym_val, llr_val),
    shuffle=True
)
```

Training...  
 Epoch 1/5

WARNING: All log messages before absl::InitializeLog() is called are written to STDERR  
 I0000 00:00:1728635635.827841 20098 device\_compiler.h:186] Compiled cluster using  
 ↪XLA! This line is logged at most once for the lifetime of the process.

```
552825/552825 [=====] - 790s 1ms/step - loss: 32.6663 - val_
↪loss: 32.3584
Epoch 2/5
552825/552825 [=====] - 794s 1ms/step - loss: 32.5126 - val_
↪loss: 32.3580
Epoch 3/5
552825/552825 [=====] - 791s 1ms/step - loss: 32.5123 - val_
↪loss: 32.3583
Epoch 4/5
552825/552825 [=====] - 792s 1ms/step - loss: 32.5125 - val_
↪loss: 32.3583
Epoch 5/5
552825/552825 [=====] - 788s 1ms/step - loss: 32.5124 - val_
↪loss: 32.3582
```

```
[7]: <keras.src.callbacks.History at 0x7f03d83de4d0>
```

#### 4.3.5.1.5 Export to TensorRT

Finally, the model gets exported to ONNX format. The ONNX format needs to be converted to TRT engine format to be consumed by the TensorRT inference engine, this is done here using the command line tool trtexec.

```
[8]: input_signature = [tf.TensorSpec([None, 2], tf.float32, name="input")]
onnx_model, _ = tf2onnx.convert.from_keras(model, input_signature)
onnx.save(onnx_model, llrnet_onnx_file)
print("ONNX model created. Converting to TRT engine file...")
command = f"trtexec " + \
```

(continues on next page)

(continued from previous page)

```

f"--onnx={llrnet_onnx_file} " + \
f"--saveEngine={llrnet_trt_file} " + \
f"--skipInference " + \
f"--minShapes=input:1x2 " + \
f"--optShapes=input:42588x2 " + \
f"--maxShapes=input:85176x2 " + \
f"--inputIOFormats=fp32:chw " + \
f"--outputIOFormats=fp32:chw" + \
f"> /dev/null"
return_val = os.system(command)
if return_val == 0:
    print("TRT engine model created.")
else:
    raise SystemExit("Failed to create the TRT engine file!")

```

ONNX model created. Converting to TRT engine file...  
TRT engine model created.

#### 4.3.5.1.6 Define a PUSCH receiver chain using pyAerial

This class encapsulates the whole PUSCH receiver chain. The components include channel estimation, noise and interference estimation, channel equalization and soft demapping, LDPC (de)rate matching and LDPC decoding. The receiver outputs the received transport block in bytes.

The soft demapping part can be replaced by LLRNet.

```

[9]: class Receiver:
    """PUSCH receiver class.

    This class encapsulates the whole PUSCH receiver chain built using
    pyAerial components.
    """

    def __init__(self,
                 llrnet_model_file,
                 num_rx_ant,
                 enable_pusch_tdi,
                 eq_coeff_algo):
        """Initialize the PUSCH receiver."""
        self.cuda_stream = get_cuda_stream()

        # Build the components of the receiver.
        self.channel_estimator = ChannelEstimator(
            num_rx_ant=num_rx_ant,
            cuda_stream=self.cuda_stream
        )
        self.channel_equalizer = ChannelEqualizer(
            num_rx_ant=num_rx_ant,
            enable_pusch_tdi=enable_pusch_tdi,
            eq_coeff_algo=eq_coeff_algo,
            cuda_stream=self.cuda_stream
        )
        self.noise_intf_estimator = NoiseIntfEstimator(
            num_rx_ant=num_rx_ant,
            eq_coeff_algo=eq_coeff_algo,

```

(continues on next page)

(continued from previous page)

```

        cuda_stream=self.cuda_stream
    )
    self.demapper = Demapper(mod_order=mod_order)
    self.trt_engine = TrtEngine(
        llrnet_model_file,
        max_batch_size=85176,
        input_tensors=[TrtTensorPrms('input', (2,)), np.float32]],
        output_tensors=[TrtTensorPrms('dense_1', (8,)), np.float32])
    )

    self.derate_match = LdpcDeRateMatch(
        enable_scrambling=True,
        cuda_stream=self.cuda_stream
    )
    self.decoder = LdpcDecoder(cuda_stream=self.cuda_stream)
    self.crc_checker = CrcChecker(cuda_stream=self.cuda_stream)
    self.llr_method = "llrnet"

def set_llr_method(self, method):
    """Set the used LLR computation method.

    Args:
        method (str): Either "aerial" meaning the conventional log-likelihood
            ratio computation, or "llrnet" for using LLRNet instead.
    """
    if method not in ["aerial", "logmap", "llrnet"]:
        raise ValueError("Invalid LLR computation method!")
    self.llr_method = method

def run(
    self,
    rx_slot,
    slot,
    pusch_configs):
    """Run the receiver."""
    # Channel estimation.
    ch_est = self.channel_estimator.estimate(
        rx_slot=rx_slot,
        slot=slot,
        pusch_configs=pusch_configs
    )

    # Noise and interference estimation.
    lw_inv, noise_var_pre_eq = self.noise_intf_estimator.estimate(
        rx_slot=rx_slot,
        channel_est=ch_est,
        slot=slot,
        pusch_configs=pusch_configs
    )

    # Channel equalization and soft demapping. Note that the cuPHY kernel actually
    ↪ computes both
    # the equalized symbols and the LLRs.
    llr, eq_sym = self.channel_equalizer.equalize(
        rx_slot=rx_slot,

```

(continues on next page)



(continued from previous page)

```

        channel_est=ch_est,
        lw_inv=lw_inv,
        noise_var_pre_eq=noise_var_pre_eq,
        pusch_configs=pusch_configs
    )

    # Use the LLRNet model here to get the log-likelihood ratios.
    dmrs_syms = pusch_configs[0].dmrs_syms
    start_sym = pusch_configs[0].start_sym
    num_symbols = pusch_configs[0].num_symbols
    num_prbs = pusch_configs[0].num_prbs
    mod_order = pusch_configs[0].ue_configs[0].mod_order
    layers = pusch_configs[0].ue_configs[0].layers
    num_data_sym = (np.array(dmrs_syms[start_sym:start_sym + num_symbols]) == 0).
↪sum()
    if self.llr_method == "llrnet":
        # Put the input in the right format.
        eq_sym_input = np.stack((np.real(eq_sym[0]), np.imag(eq_sym[0]))).
↪reshape(2, -1).T
        # Run the model.
        llr_output = self.trt_engine.run({"input": eq_sym_input})["dense_1"]

        # Reshape the output in the right format for the LDPC decoding process.
        llr_output = np.array(llr_output)[..., :mod_order].T.reshape(mod_order,
↪layers, num_prbs * 12, num_data_sym)
        llr_output *= normalizer
    elif self.llr_method == "aerial":
        llr_output = llr[0]
    elif self.llr_method == "logmap":
        inv_noise_var_lin = self.channel_equalizer.ree_diag_inv[0]
        inv_noise_var_lin = np.transpose(inv_noise_var_lin[..., 0], (1, 2, 0)).
↪reshape(inv_noise_var_lin.shape[1], -1)
        llr_output = self.demapper.demap(eq_sym[0], inv_noise_var_lin[..., None])

    # De-rate matching and descrambling.
    coded_blocks = self.derate_match.derate_match(
        input_llrs=llr_output,
        pusch_configs=pusch_configs
    )

    # LDPC decoding of the derate matched blocks.
    code_blocks = self.decoder.decode(
        input_llrs=coded_blocks,
        pusch_configs=pusch_configs
    )

    # Combine the code blocks into a transport block.
    tb, _ = self.crc_checker.check_crc(
        input_bits=code_blocks,
        pusch_configs=pusch_configs
    )

    return tb[0]

```

## 4.3.5.1.7 Model testing on Aerial test vectors

```
[10]: if mod_order == 2:
    test_vector_filename = "TVnr_7201_PUSCH_gNB_CUPHY_s0p0.h5"
elif mod_order == 4:
    test_vector_filename = "TVnr_7916_PUSCH_gNB_CUPHY_s0p0.h5"
elif mod_order == 6:
    test_vector_filename = "TVnr_7203_PUSCH_gNB_CUPHY_s0p0.h5"
filename = AERIAL_TEST_VECTOR_DIR + test_vector_filename
input_file = h5.File(filename, "r")

num_rx_ant = input_file["gnb_pars"]["nRx"][0]
enable_pusch_tdi = input_file["gnb_pars"]["TdiMode"][0]
eq_coeff_algo = input_file["gnb_pars"]["eqCoeffAlgoIdx"][0]

receiver = Receiver(
    llrnet_trt_file,
    num_rx_ant=num_rx_ant,
    enable_pusch_tdi=enable_pusch_tdi,
    eq_coeff_algo=eq_coeff_algo
)

# Extract the test vector data and parameters.
rx_slot = np.array(input_file["DataRx"])[ "re" ] + 1j * np.array(input_file["DataRx"])[
    ↪ "im" ]
rx_slot = rx_slot.transpose(2, 1, 0)

slot = np.array(input_file["gnb_pars"]["slotNumber"])[0]

# Wrap the parameters in a PuschConfig structure.
pusch_ue_config = PuschUeConfig(
    scid=input_file["tb_pars"]["nSCID"][0],
    layers=input_file["tb_pars"]["numLayers"][0],
    dmrs_ports=input_file["tb_pars"]["dmrsPortBmsk"][0],
    rnti=input_file["tb_pars"]["nRnti"][0],
    data_scid=input_file["tb_pars"]["dataScramId"][0],
    mcs_table=input_file["tb_pars"]["mcsTableIndex"][0],
    mcs_index=input_file["tb_pars"]["mcsIndex"][0],
    code_rate=input_file["tb_pars"]["targetCodeRate"][0],
    mod_order=input_file["tb_pars"]["qamModOrder"][0],
    tb_size=input_file["tb_pars"]["nTbByte"][0],
    rv=input_file["tb_pars"]["rv"][0],
    ndi=input_file["tb_pars"]["ndi"][0]
)

# Note that this is a list. One UE group only in this case.
pusch_configs = [PuschConfig(
    ue_configs=[pusch_ue_config],
    num_dmrs_cdm_grps_no_data=input_file["tb_pars"]["numDmrsCdmGrpsNoData"][0],
    dmrs_scrm_id=input_file["tb_pars"]["dmrsScramId"][0],
    start_prb=input_file["ueGrp_pars"]["startPrb"][0],
    num_prbs=input_file["ueGrp_pars"]["nPrb"][0],
    dmrs_syms=dmrs_fapi_to_bit_array(input_file["ueGrp_pars"]["dmrsSymLocBmsk"][0]),
    dmrs_max_len=input_file["tb_pars"]["dmrsMaxLength"][0],
    dmrs_add_ln_pos=input_file["tb_pars"]["dmrsAddlPosition"][0],
    start_sym=input_file["ueGrp_pars"]["StartSymbolIndex"][0],
    num_symbols=input_file["ueGrp_pars"]["NrOfSymbols"][0]
)]
```

(continues on next page)

(continued from previous page)

```

# Run the receiver with the test vector parameters.
receiver.set_llr_method("llrnet")
tb = receiver.run(
    rx_slot=rx_slot,
    slot=slot,
    pusch_configs=pusch_configs
)

# Check that the received TB matches with the transmitted one.
tb_size = pusch_configs[0].ue_configs[0].tb_size
if np.array_equal(np.array(input_file["tb_data"][:tb_size, 0]), tb[:tb_size]):
    print("CRC check passed!")
else:
    print("CRC check failed!")

```

CRC check passed!

#### 4.3.5.1.8 Model testing on synthetic/simulated data

```

[11]: for pusch_record in df.take(test_indices).itertuples(index=False):

    user_data_filename = dataset_dir + pusch_record.user_data_filename
    user_data = load_pickle(user_data_filename)
    snr = user_data["snr"]

    rx_iq_data_filename = dataset_dir + pusch_record.rx_iq_data_filename
    rx_slot = load_pickle(rx_iq_data_filename)

    ref_tb = pusch_record.macPdu
    tb_size = len(pusch_record.macPdu)
    slot = pusch_record.Slot

    # Wrap the parameters in a PuschConfig structure.
    pusch_ue_config = PuschUeConfig(
        scid=pusch_record.SCID,
        layers=pusch_record.nrOfLayers,
        dmrs_ports=pusch_record.dmrsPorts,
        rnti=pusch_record.RNTI,
        data_scid=pusch_record.dataScramblingId,
        mcs_table=pusch_record.mcsTable,
        mcs_index=pusch_record.mcsIndex,
        code_rate=pusch_record.targetCodeRate,
        mod_order=pusch_record.qamModOrder,
        tb_size=tb_size
    )
    # Note that this is a list. One UE group only in this case.
    pusch_configs = [PuschConfig(
        ue_configs=[pusch_ue_config],
        num_dmrs_cdm_grps_no_data=pusch_record.numDmrsCdmGrpsNoData,
        dmrs_scrm_id=pusch_record.ulDmrsScramblingId,
        start_prb=pusch_record.rbStart,
        num_prbs=pusch_record.rbSize,
        dmrs_syms=dmrs_fapi_to_bit_array(pusch_record.ulDmrsSymbPos),

```

(continues on next page)

(continued from previous page)

```

    dmrs_max_len=1,
    dmrs_add_ln_pos=1,
    start_sym=pusch_record.StartSymbolIndex,
    num_symbols=pusch_record.NrOfSymbols
  ])

  for llr_method in ["aerial", "llrnet", "logmap"]:

    if snr not in tb_errors[llr_method].keys():
      tb_errors[llr_method][snr] = 0
      tb_count[llr_method][snr] = 0

    receiver.set_llr_method(llr_method)
    tb = receiver.run(
      rx_slot=rx_slot,
      slot=slot,
      pusch_configs=pusch_configs
    )

    tb_count[llr_method][snr] += 1
    tb_errors[llr_method][snr] += (not np.array_equal(tb[:tb_size], ref_tb[:tb_
↪size]))

```

```

[12]: esno_dbs = tb_count["aerial"].keys()
      bler = dict(aerial=[], llrnet=[], logmap=[])
      for esno_db in esno_dbs:
        bler["aerial"].append(tb_errors["aerial"][esno_db] / tb_count["aerial"][esno_db])
        bler["llrnet"].append(tb_errors["llrnet"][esno_db] / tb_count["llrnet"][esno_db])
        bler["logmap"].append(tb_errors["logmap"][esno_db] / tb_count["logmap"][esno_db])

```

```

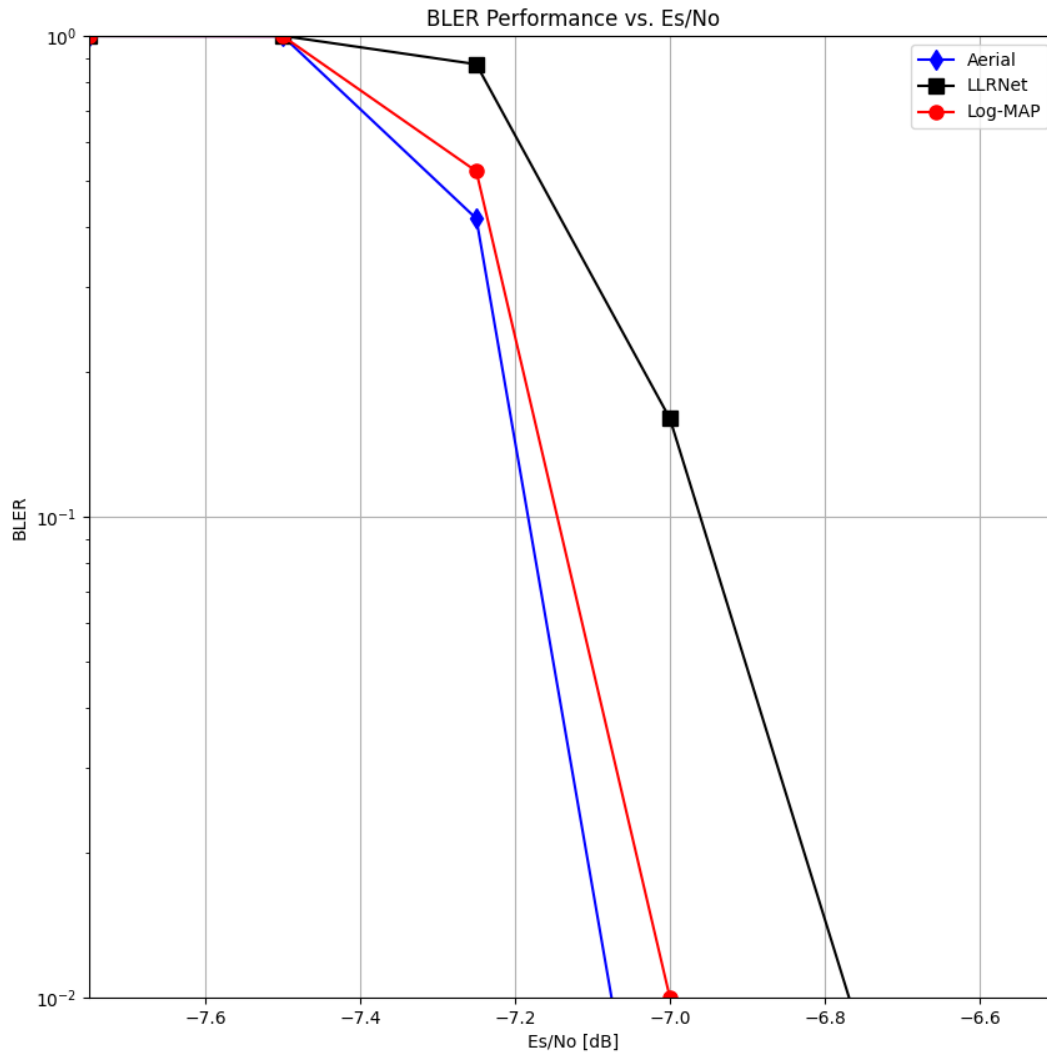
[13]: esno_dbs = np.array(list(esno_dbs))
      fig = plt.figure(figsize=(10, 10))
      plt.yscale('log')
      plt.ylim(0.01, 1)
      plt.xlim(np.min(esno_dbs), np.max(esno_dbs))
      plt.title("BLER Performance vs. Es/No")
      plt.ylabel("BLER")
      plt.xlabel("Es/No [dB]")
      plt.grid()
      plt.plot(esno_dbs, bler["aerial"], marker="d", linestyle="-", color="blue",
↪markersize=8)
      plt.plot(esno_dbs, bler["llrnet"], marker="s", linestyle="-", color="black",
↪markersize=8)
      plt.plot(esno_dbs, bler["logmap"], marker="o", linestyle="-", color="red",
↪markersize=8)
      plt.legend(["Aerial", "LLRNet", "Log-MAP"])

```

```

[13]: <matplotlib.legend.Legend at 0x7f02ac7bc760>

```



### 4.3.6. Neural receiver validation

In this example, a trained neural network -based PUSCH receiver is validated using pyAerial. The model is based on the following paper:

S. Cammerer, F. Ait Aoudia, J. Hoydis, A. Oeldemann, A. Roessler, T. Mayer and A. Keller, "A Neural Receiver for 5G NR Multi-user MIMO", IEEE Globecom Workshops (GC Wkshps), Dec. 2023, <https://arxiv.org/abs/2312.02601>

The neural receiver is compared against the conventional PUSCH receiver using pyAerial. For running inference, we use pyAerial's bindings to cuPHY's TensorRT wrappers.

#### 4.3.6.1 Using pyAerial to evaluate a PUSCH neural receiver

This example shows how to use the pyAerial cuPHY Python bindings to evaluate a trained neural network-based PUSCH receiver. In this example, the neural network is used to replace channel estimation, noise and interference estimation and channel equalization, and thus outputs log-likelihood ratios directly. The model is a variant of what has been proposed in

S. Cammerer, F. Ait Aoudia, J. Hoydis, A. Oeldemann, A. Roessler, T. Mayer and A. Keller, “A Neural Receiver for 5G NR Multi-user MIMO”, IEEE Globecom Workshops (GC Wkshps), Dec. 2023.

The rest of the PUSCH receiver pipeline following the neural receiver, meaning LDPC decoding chain, is modeled using pyAerial. Also, the neural receiver takes LS channel estimates as inputs in addition to the received PUSCH slot. These are also obtained using pyAerial. The neural receiver-based PUSCH receiver is compared against the conventional PUSCH receiver, which is built using pyAerial’s (fully fused) PUSCH pipeline.

PUSCH transmitter is emulated by PDSCH transmission with properly chosen parameters, that way making it a 5G NR compliant PUSCH transmission. The NVIDIA [Sionna](#) library is utilized for simulating the radio channel based on 3GPP channel models.

```
[1]: # Check platform.
import platform
if platform.machine() != 'x86_64':
    raise SystemExit("Unsupported platform!")
```

##### 4.3.6.1.1 Imports

```
[2]: %matplotlib widget
from collections import defaultdict
import os
os.environ["CUDA_VISIBLE_DEVICES"] = "0"
os.environ["TF_CPP_MIN_LOG_LEVEL"] = "3" # Silence TensorFlow.
os.environ["CUDA_MODULE_LOADING"] = "LAZY"

import numpy as np
import sionna
import tensorflow as tf

from aerial.phy5g.pdsch import PdschTx
from aerial.phy5g.pusch import PuschRx
from aerial.phy5g.algorithms import ChannelEstimator
from aerial.phy5g.algorithms import TrtEngine
from aerial.phy5g.algorithms import TrtTensorPrms
from aerial.phy5g.ldpc import get_mcs
from aerial.phy5g.ldpc import random_tb
from aerial.phy5g.ldpc import get_tb_size
from aerial.phy5g.ldpc import LdpcDeRateMatch
from aerial.phy5g.ldpc import LdpcDecoder
from aerial.phy5g.ldpc import CrcChecker
from aerial.phy5g.types import PuschLdpcKernelLaunch
from aerial.phy5g.params import PuschConfig
from aerial.phy5g.params import PuschUeConfig
from aerial.util.cuda import get_cuda_stream
from simulation_monitor import SimulationMonitor
```

(continues on next page)

(continued from previous page)

```
# Configure the notebook to use only a single GPU and allocate only as much memory as
↪needed.
# For more details, see https://www.tensorflow.org/guide/gpu.
gpus = tf.config.list_physical_devices('GPU')
tf.config.experimental.set_memory_growth(gpus[0], True)
```

#### 4.3.6.1.2 Parameters

Set simulation parameters, numerology, PUSCH parameters and channel parameters here.

```
[3]: # Simulation parameters.
esno_db_range = np.arange(-4.0, -2.8, 0.2)
num_slots = 10000
min_num_tb_errors = 250

# Numerology and frame structure. See TS 38.211.
num_ofdm_symbols = 14
fft_size = 4096
cyclic_prefix_length = 288
subcarrier_spacing = 30e3
num_guard_subcarriers = (410, 410)
num_slots_per_frame = 20

# System/gNB configuration
num_tx_ant = 1           # UE antennas
num_rx_ant = 4           # gNB antennas
cell_id = 41             # Physical cell ID
enable_pusch_tdi = 1     # Enable time interpolation for equalizer coefficients
eq_coeff_algo = 1        # Equalizer algorithm

# PUSCH parameters
rnti = 1234              # UE RNTI
scid = 0                 # DMRS scrambling ID
data_scid = 0            # Data scrambling ID
layers = 1               # Number of layers
mcs_index = 7            # MCS index as per TS 38.214 table.
mcs_table = 0            # MCS table index
dmrs_ports = 1           # Used DMRS port.
start_prb = 0            # Start PRB index.
num_prbs = 273           # Number of allocated PRBs.
start_sym = 0            # Start symbol index.
num_symbols = 12         # Number of symbols.
dmrs_scrm_id = 41        # DMRS scrambling ID
dmrs_syms = [1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0] # Indicates which symbols are
↪used for DMRS.
dmrs_max_len = 1
dmrs_add_ln_pos = 2
num_dmrs_cdm_grps_no_data = 2
mod_order, code_rate = get_mcs(mcs_index, mcs_table+1) # Different indexing for MCS
↪table.
tb_size = get_tb_size(mod_order, code_rate, dmrs_syms, num_prbs, start_sym, num_
↪symbols, layers) # TB size in bits

# Channel parameters
carrier_frequency = 3.5e9 # Carrier frequency in Hz.
```

(continues on next page)

(continued from previous page)

```

delay_spread = 100e-9      # Nominal delay spread in [s]. Please see the CDL
↳documentation          # about how to choose this value.
link_direction = "uplink"
channel_model = "Rayleigh" # Channel model: Suitable values:
                          # "Rayleigh" - Rayleigh block fading channel model (sionna.
↳channel.RayleighBlockFading)
                          # "CDL-x", where x is one of ["A", "B", "C", "D", "E"] - for
↳3GPP CDL channel models
                          #
                          # as per TR 38.901.
speed = 0.8333           # UE speed [m/s]. The direction of travel will chosen
↳randomly within the x-y plane.

```

#### 4.3.6.1.3 Create the model file for the TRT engine

The TRT engine is built based on TensorRT plan files which are not portable across different platforms. Hence the plan file is created here from a supplied ONNX file.

```

[4]: MODEL_DIR = "../models"
nrx_onnx_file = f"{MODEL_DIR}/neural_rx.onnx"
nrx_trt_file = f"{MODEL_DIR}/neural_rx.trt"
command = f"trtexec " + \
    f"--onnx={nrx_onnx_file} " + \
    f"--saveEngine={nrx_trt_file} " + \
    f"--skipInference " + \
    f"--inputIOFormats=fp32:chw, fp32:chw, fp32:chw, fp32:chw, fp32:chw, int32:chw, int32:
↳chw " + \
    f"--outputIOFormats=fp32:chw, fp32:chw " + \
    f"--shapes=rx_slot_real:1x3276x12x4, rx_slot_imag:1x3276x12x4, h_hat_real:
↳1x4914x1x4, h_hat_imag:1x4914x1x4 " + \
    f"> /dev/null"
return_val = os.system(command)
if return_val == 0:
    print("TRT engine model created.")
else:
    raise SystemExit("Failed to create the TRT engine file!")

```

TRT engine model created.

#### 4.3.6.1.4 Create the PUSCH pipelines

As mentioned, PUSCH transmission is emulated here by the PDSCH transmission chain. Note that the static cell parameters and static PUSCH parameters are given upon creating the PUSCH transmission/reception objects. Dynamically (per slot) changing parameters are however set when actually running the transmission/reception, see further below.

```

[5]: pusch_tx = PdschTx(
    cell_id=cell_id,
    num_rx_ant=num_tx_ant,
    num_tx_ant=num_tx_ant,
)

# This is the fully fused PUSCH receiver chain.

```

(continues on next page)



(continued from previous page)

```

pusch_rx = PuschRx(
    cell_id=cell_id,
    num_rx_ant=num_rx_ant,
    num_tx_ant=num_rx_ant,
    enable_pusch_tdi=enable_pusch_tdi,
    eq_coeff_algo=eq_coeff_algo,
    # To make this equal separate PUSCH Rx components configuration:
    ldpc_kernel_launch=PuschLdpcKernelLaunch.PUSCH_RX_LDPC_STREAM_SEQUENTIAL
)

# PUSCH configuration object. Note that default values are used for some parameters
# not given here.
pusch_ue_config = PuschUeConfig(
    scid=scid,
    layers=layers,
    dmrs_ports=dmrs_ports,
    rnti=rnti,
    data_scid=data_scid,
    mcs_table=mcs_table,
    mcs_index=mcs_index,
    code_rate=int(code_rate * 10),
    mod_order=mod_order,
    tb_size=tb_size // 8
)

# Note that this is a list. One UE group only in this case.
pusch_configs = [PuschConfig(
    ue_configs=[pusch_ue_config],
    num_dmrs_cdm_grps_no_data=num_dmrs_cdm_grps_no_data,
    dmrs_scrm_id=dmrs_scrm_id,
    start_prb=start_prb,
    num_prbs=num_prbs,
    dmrs_syms=dmrs_syms,
    dmrs_max_len=dmrs_max_len,
    dmrs_add_ln_pos=dmrs_add_ln_pos,
    start_sym=start_sym,
    num_symbols=num_symbols
)]

class NeuralRx:
    """PUSCH neural receiver class.

    This class encapsulates the PUSCH neural receiver chain built using
    pyAerial components.
    """

    def __init__(self,
                 num_rx_ant,
                 enable_pusch_tdi,
                 eq_coeff_algo):
        """Initialize the neural receiver."""
        self.cuda_stream = get_cuda_stream()

        # Build the components of the receiver. The channel estimator outputs just the
        ↪LS # channel estimates.

```

(continues on next page)

(continued from previous page)

```

self.channel_estimator = ChannelEstimator(
    num_rx_ant=num_rx_ant,
    ch_est_algo=3, # This is LS channel estimation.
    cuda_stream=self.cuda_stream
)

# Create the pyAerial TRT engine object. This wraps TensorRT and links it
→together
# with the rest of cuPHY. Here pyAerial's Python bindings to the engine are used
# to run inference with the neural receiver model.
# The inputs of the neural receiver are:
# - LS channel estimates
# - The Rx slot
# - Active DMRS ports (active layers out of the layers that the neural receiver
→supports)
# - DMRS OFDM symbol locations (indices)
# - DMRS subcarrier positions within a PRB (indices)
# Note that the shapes are given without batch size.
self.trt_engine = TrtEngine(
    trt_model_file="./models/neural_rx.trt",
    max_batch_size=1,
    input_tensors=[TrtTensorPrms('rx_slot_real', (3276, 12, 4), np.float32),
                  TrtTensorPrms('rx_slot_imag', (3276, 12, 4), np.float32),
                  TrtTensorPrms('h_hat_real', (4914, 1, 4), np.float32),
                  TrtTensorPrms('h_hat_imag', (4914, 1, 4), np.float32),
                  TrtTensorPrms('active_dmrs_ports', (1,), np.float32),
                  TrtTensorPrms('dmrs_ofdm_pos', (3,), np.int32),
                  TrtTensorPrms('dmrs_subcarrier_pos', (6,), np.int32)],
    output_tensors=[TrtTensorPrms('output_1', (8, 1, 3276, 12), np.float32),
                   TrtTensorPrms('output_2', (1, 3276, 12, 8), np.float32)]
)

# LDPC (de)rate matching and decoding implemented using pyAerial.
self.derate_match = LdpcDeRateMatch(
    enable_scrambling=True,
    cuda_stream=self.cuda_stream
)
self.decoder = LdpcDecoder(cuda_stream=self.cuda_stream)
self.crc_checker = CrcChecker(cuda_stream=self.cuda_stream)

def run(
    self,
    rx_slot,
    slot,
    pusch_configs=pusch_configs
):
    """Run the receiver."""
    # Channel estimation.
    ch_est = self.channel_estimator.estimate(
        rx_slot=rx_slot,
        slot=slot,
        pusch_configs=pusch_configs
    )

    # This is the neural receiver part.
    # It outputs the LLRs for all symbols.

```

(continues on next page)

(continued from previous page)

```

dmrs_ofdm_pos = np.where(np.array(pusch_configs[0].dmrs_syms))[0].astype(np.
↳int32)
dmrs_ofdm_pos = dmrs_ofdm_pos[None, ...]
dmrs_subcarrier_pos = np.array([[0, 2, 4, 6, 8, 10]], dtype=np.int32)
active_dmrs_ports = np.ones((1, 1), dtype=np.float32)
rx_slot_in = rx_slot[None, :, pusch_configs[0].start_sym:pusch_configs[0].
↳start_sym+pusch_configs[0].num_symbols, :]
ch_est_in = np.transpose(ch_est[0], (0, 3, 1, 2)).reshape(ch_est[0].shape[0]
↳* ch_est[0].shape[3], ch_est[0].shape[1], ch_est[0].shape[2])
ch_est_in = ch_est_in[None, ...]
input_tensors = {
    "rx_slot_real": np.real(rx_slot_in).astype(np.float32),
    "rx_slot_imag": np.imag(rx_slot_in).astype(np.float32),
    "h_hat_real": np.real(ch_est_in).astype(np.float32),
    "h_hat_imag": np.imag(ch_est_in).astype(np.float32),
    "active_dmrs_ports": active_dmrs_ports.astype(np.float32),
    "dmrs_ofdm_pos": dmrs_ofdm_pos.astype(np.int32),
    "dmrs_subcarrier_pos": dmrs_subcarrier_pos.astype(np.int32)
}
outputs = self.trt_engine.run(input_tensors)

# The neural receiver outputs some values also for DMRS symbols, remove those
# from the output.
data_syms = np.array(pusch_configs[0].dmrs_syms[pusch_configs[0].start_sym:
↳pusch_configs[0].start_sym + pusch_configs[0].num_symbols]) == 0
llrs = np.take(outputs["output_1"][0, ...], np.where(data_syms)[0], axis=3)

coded_blocks = self.derate_match.derate_match(
    input_llrs=[llrs],
    pusch_configs=pusch_configs
)

code_blocks = self.decoder.decode(
    input_llrs=coded_blocks,
    pusch_configs=pusch_configs
)

decoded_tbs, _ = self.crc_checker.check_crc(
    input_bits=code_blocks,
    pusch_configs=pusch_configs
)

return decoded_tbs

neural_rx = NeuralRx(
    num_rx_ant=num_rx_ant,
    enable_pusch_tdi=enable_pusch_tdi,
    eq_coeff_algo=eq_coeff_algo
)

```

#### 4.3.6.1.5 Channel generation using Sionna

Simulating the transmission through the radio channel takes advantage of the channel model implementations available in NVIDIA Sionna. In Sionna, the transmission can be simulated directly in frequency domain by defining a resource grid. In our case, reference signal patterns and data carrying resource elements are defined elsewhere within the Aerial code, hence we define resource grid as a simple dummy grid containing only data symbols.

See also: [Sionna documentation](#)

```
[6]: # Define the resource grid.
resource_grid = sionna.ofdm.ResourceGrid(
    num_ofdm_symbols=num_ofdm_symbols,
    fft_size=fft_size,
    subcarrier_spacing=subcarrier_spacing,
    num_tx=1,
    num_streams_per_tx=1,
    cyclic_prefix_length=cyclic_prefix_length,
    num_guard_carriers=num_guard_subcarriers,
    dc_null=False,
    pilot_pattern=None,
    pilot_ofdm_symbol_indices=None
)
resource_grid_mapper = sionna.ofdm.ResourceGridMapper(resource_grid)
remove_guard_subcarriers = sionna.ofdm.RemoveNulledSubcarriers(resource_grid)

# Define the antenna arrays.
ue_array = sionna.channel.tr38901.Antenna(
    polarization="single",
    polarization_type="V",
    antenna_pattern="38.901",
    carrier_frequency=carrier_frequency
)
gnb_array = sionna.channel.tr38901.AntennaArray(
    num_rows=1,
    num_cols=int(num_rx_ant/2),
    polarization="dual",
    polarization_type="cross",
    antenna_pattern="38.901",
    carrier_frequency=carrier_frequency
)

if channel_model == "Rayleigh":
    ch_model = sionna.channel.RayleighBlockFading(
        num_rx=1,
        num_rx_ant=num_rx_ant,
        num_tx=1,
        num_tx_ant=num_tx_ant
    )

elif "CDL" in channel_model:
    cdl_model = channel_model[-1]

    # Configure a channel impulse response (CIR) generator for the CDL model.
    ch_model = sionna.channel.tr38901.CDL(
        cdl_model,
        delay_spread,
```

(continues on next page)

(continued from previous page)

```

        carrier_frequency,
        ue_array,
        gnb_array,
        link_direction,
        min_speed=speed
    )
else:
    raise ValueError(f"Invalid channel model {channel_model}!")

channel = sionna.channel.OFDMChannel(
    ch_model,
    resource_grid,
    add_awgn=True,
    normalize_channel=True,
    return_channel=False
)

def apply_channel(tx_tensor, No):
    """Transmit the Tx tensor through the radio channel."""
    # Add batch and num_tx dimensions that Sionna expects and reshape.
    tx_tensor = tf.transpose(tx_tensor, (2, 1, 0))
    tx_tensor = tf.reshape(tx_tensor, (1, -1))[None, None]
    tx_tensor = resource_grid_mapper(tx_tensor)
    rx_tensor = channel((tx_tensor, No))
    rx_tensor = remove_guard_subcarriers(rx_tensor)
    rx_tensor = rx_tensor[0, 0]
    rx_tensor = tf.transpose(rx_tensor, (2, 1, 0))
    return rx_tensor

```

#### 4.3.6.1.6 Run the actual simulation

Here we loop across the Es/No range, and simulate a number of slots for each Es/No value. A single transport block is simulated within a slot. The simulation starts over from the next Es/No value when a minimum number of transport block errors is reached.

```

[7]: cases = ["PUSCH Rx", "Neural Rx"]
monitor = SimulationMonitor(cases, esno_db_range)

# Loop the Es/No range.
bler = []
for esno_db in esno_db_range:
    monitor.step(esno_db)
    num_tb_errors = defaultdict(int)

    # Run multiple slots and compute BLER.
    for slot_idx in range(num_slots):
        slot_number = slot_idx % num_slots_per_frame

        # Get modulation order and coderate.
        tb_input = random_tb(mod_order, code_rate, dmrs_syms, num_prbs, start_sym,
↪ num_symbols, layers)

        # Transmit PUSCH. This is where we set the dynamically changing parameters.
        # Input parameters are given as lists as the interface supports multiple UEs.

```

(continues on next page)

(continued from previous page)

```

tx_tensor = pusch_tx.run(
    tb_inputs=[tb_input],          # Input transport block in bytes.
    num_ues=1,                    # We simulate only one UE here.
    slot=slot_number,            # Slot number.
    num_dmrs_cdm_grps_no_data=num_dmrs_cdm_grps_no_data,
    dmrs_scrm_id=dmrs_scrm_id,    # DMRS scrambling ID.
    start_prb=start_prb,         # Start PRB index.
    num_prbs=num_prbs,           # Number of allocated PRBs.
    dmrs_syms=dmrs_syms,         # List of binary numbers indicating which
    ↪symbols are DMRS.
    start_sym=start_sym,         # Start symbol index.
    num_symbols=num_symbols,      # Number of symbols.
    scids=[scid],                # DMRS scrambling ID.
    layers=[layers],             # Number of layers (transmission rank).
    dmrs_ports=[dmrs_ports],     # DMRS port(s) to be used.
    rntis=[rnti],                # UE RNTI.
    data_scids=[data_scid],       # Data scrambling ID.
    code_rates=[code_rate],       # Code rate x 1024.
    mod_orders=[mod_order]       # Modulation order.
)[0]

# Channel transmission using TF and Sionna.
No = pow(10., -esno_db / 10.)
rx_tensor = apply_channel(tx_tensor, No)
rx_tensor = np.array(rx_tensor)

# Run the fused PUSCH receiver.
# Note that this is where we set the dynamically changing parameters.
tb_crcs, tbs = pusch_rx.run(
    rx_slot=rx_tensor,
    slot=slot_number,
    pusch_configs=pusch_configs
)
num_tb_errors["PUSCH Rx"] += int(np.array_equal(tbs[0], tb_input) == False)

# Run the neural receiver.
tbs = neural_rx.run(
    rx_slot=rx_tensor,
    slot=slot_number,
    pusch_configs=pusch_configs
)
num_tb_errors["Neural Rx"] += int(np.array_equal(tbs[0], tb_input) == False)

monitor.update(num_tbs=slot_idx + 1, num_tb_errors=num_tb_errors)
if np.array(list(num_tb_errors.values())) >= min_num_tb_errors.all():
    break # Next Es/No value.

monitor.finish_step(num_tbs=slot_idx + 1, num_tb_errors=num_tb_errors)
monitor.finish()

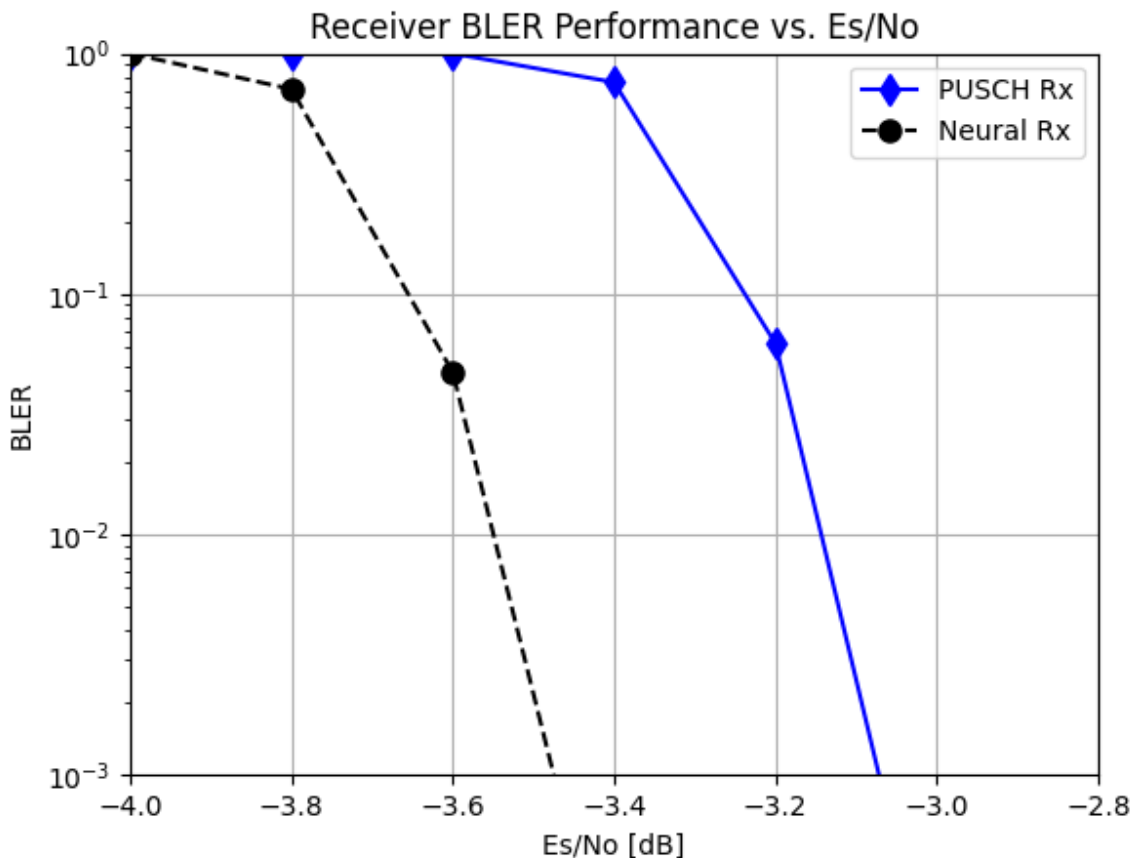
```

Es/No (dB)	PUSCH Rx			Neural Rx		ms/TB
	TBs	TB Errors	BLER	TB Errors	BLER	
-4.00	250	250	1.0000	250	1.0000	199.5
-3.80	352	352	1.0000	250	0.7102	198.1
-3.60	5292	5289	0.9994	250	0.0472	198.4

(continues on next page)

(continued from previous page)

-3.40	10000	7587	0.7587	1	0.0001	200.0
-3.20	10000	622	0.0622	0	0.0000	199.6
-3.00	10000	1	0.0001	0	0.0000	199.1
-2.80	10000	0	0.0000	0	0.0000	198.8



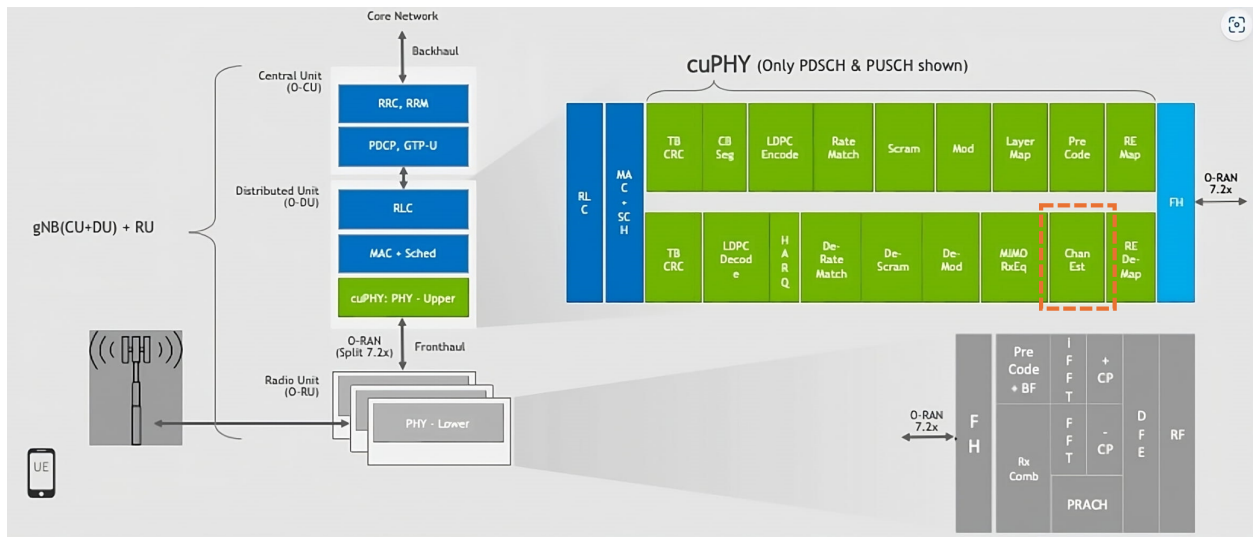
### 4.3.7. Machine learning based channel estimation for 5G NR PUSCH

This notebook provides another example on how to validate machine learning models using pyAerial. In particular, the model implements 5G NR PUSCH channel estimation using DMRS pilots. The model is trained using PyTorch, and the channel estimation error is validated against cuPHY algorithms using pyAerial.

### 4.3.7.1 Channel Estimation for Uplink Shared Channel (PUSCH) in pyAerial

This notebook provides researchers an example of how to prototype machine learning in pyAerial. PyAerial is the Python bindings for the Aerial SDK, NVIDIA's L1 and L2 accelerated stack that is also integrated in the Aerial Omniverse Digital Twin (AODT). This enables researchers to develop standard-compliant approaches focusing on enhancing their link-level performance. Subsequently, the approach can be evaluated realistically in AODT, showing how the link-level performance translates to system-level KPIs.

In particular, this notebook focuses on improving the channel estimation based on the DMRS pilots in a PUSCH transmission. First, we isolate the channel estimator block from the pyAerial PUSCH pipeline. The channel estimation is on one of the first receiver blocks, as seen in the figure below:



To isolate the channel estimation block, we refer to the modular PUSCH pipeline in [Example PUSCH Simulation.ipynb](#). There, we see how to interface channel estimation downstream with resource element (RE) demapper and with the wireless channel, and upstream with other components like the MIMO Equalizer. Similar approaches can be done for other blocks in the receiver or transmitter pipelines.

**This notebook uses PyTorch to train a convolutional neural network that improves and interpolates least squares (LS) channel estimates. The training uses a custom LS estimator which interfaces with a resource grid, resource mapper, and channel generator from Sionna. The trained models are then integrated and validated in pyAerial PUSCH pipeline with standard-compliant signal transmission and reception blocks running on top of Sionna channels.**

\*

See below how we train and test the channel estimator models.



```
[1]: import os
# GPU setup
GPU_IDX = 2
os.environ["CUDA_VISIBLE_DEVICES"] = str(GPU_IDX) # Select only one GPU
os.environ['TF_CPP_MIN_LOG_LEVEL'] = "3" # Silence TensorFlow.

import torch
import numpy as np
from tqdm import tqdm
```

(continues on next page)



(continued from previous page)

```

import tensorflow as tf
import matplotlib.pyplot as plt

# Our imports
from channel_gen_funcs import SionnaChannelGenerator
from channel_gen_funcs import PyAerialChannelEstimateGenerator
from channel_est_models import ChannelEstimator
from channel_est_models import ComplexMSELoss
import utils as ut

dev = torch.device(f'cuda')
torch.set_default_device(dev)

# General parameters
num_prbs = 48 # Number of PRBs in the UE allocated bandwidth
interp = 1 # Interpolation factor. If 2, it estimates also for REs
↳without DMRS
models_folder = f'saved_models_prbs={num_prbs}_interp={interp}' # Folder to save
↳trained models

# Training parameters
train_snrs = np.array([-10]) # Train models for these SNRs. Suggested: np.arange(-10, 40.
↳1, 10)
training_ch_model = 'UMa' # Channel model ['Rayleigh', 'CDL-x', 'TDL-x', 'UMa', 'UMi
↳'],
# where x is in ["A", "B", "C", "D", "E"] as per TR 38.901
n_iter = 5000 # Number of training iterations. Best results: 30k
batch_size = 32 # Batch size = number of channels to train simultaneously
model_reuse = True # Whether to load the weights of the last trained model
# (this trick improves convergence speed considerably)

# Testing parameters
test_snrs = np.arange(-10, 40.1, 5) # Test models for these SNRs. Suggested: np.
↳arange(-10, 40.1, 5)
testing_ch_model = 'UMi' # Channel for testing
n_iter_test = 1000 # Number of batches of channels to evaluate
↳(batch=32)

[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]

```

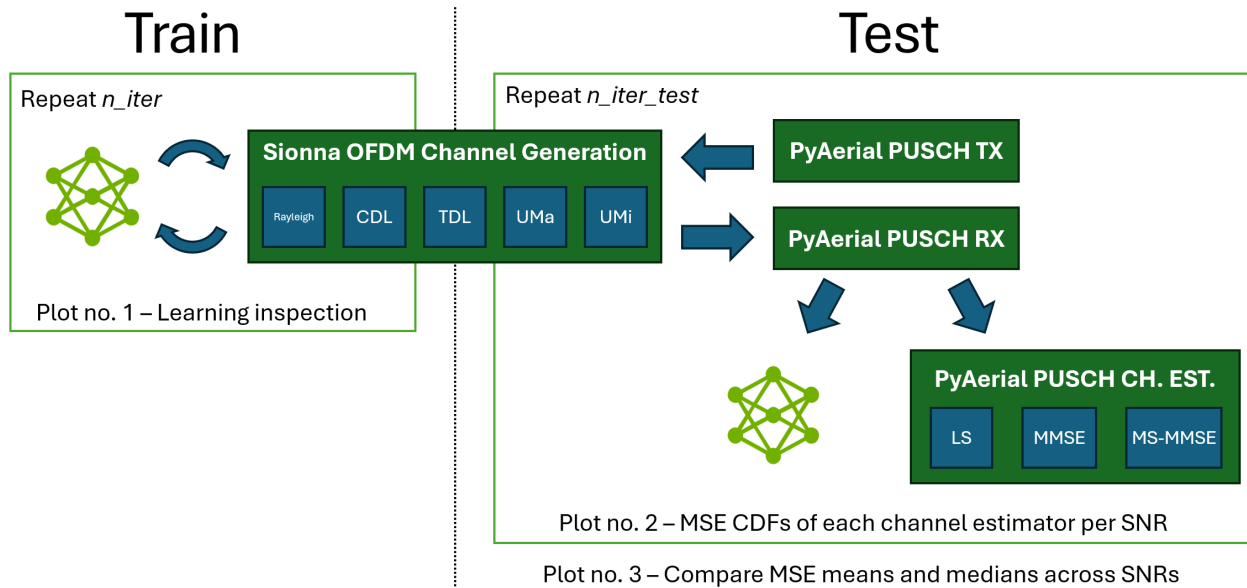
#### 4.3.7.1.1 Training channel estimation model

The example machine learning model uses the least squares (LS) estimates and outputs a more accurate channel estimate. In its base configuration, the DMRS has 1/2 density in frequency (i.e. one RE for every two subcarriers). Therefore, our ChannelEstimator can either output as many estimates as DMRS pilots (half of the subcarriers using an interpolation factor `interp=1`) or twice as many estimates (with the parameter `interp=2`) thus covering all subcarriers.

*Important note on training:* Our approach consists of training one model per SNR. SNR-specific models can learn more accurately how to estimate the channels for SNRs close to the original SNR that was used for model training. This approach also solves the problem where low SNR channels incur in higher loss and lead to the model focusing on them and not working for the high SNR cases.

For training, the model interfaces directly with Sionna channel models. For testing, the model is integrated in pyAerial's PUSCH pipeline and evaluated alongside other classic channel estimators, like the

minimum mean squared error (MMSE) and the multi-stage MMSE (MS-MMSE). A diagram of training and testing is below:



```
[2]: models_dir = ut.get_model_training_dir(models_folder, training_ch_model,
                                           num_prbs, n_iter, batch_size)
os.makedirs(models_dir, exist_ok=True)

# Channel generator for training
train_ch_gen = SionnaChannelGenerator(num_prbs, training_ch_model, batch_size)

last_model = ''
for snr_idx, snr in enumerate(train_snr):
    print(f'Training model for SNRs: {snr} dB')
    save_model_path = ut.get_snr_model_path(models_dir, snr)

    model = ChannelEstimator(num_prbs*12//2, freq_interp_factor=interp).to(dev)
    reload_last_model = model_reuse and last_model
    if reload_last_model:
        model.load_state_dict(torch.load(last_model))

    criterion = ComplexMSELoss()
    optimizer = torch.optim.AdamW(model.parameters(), lr=1e-3, weight_decay=1e-4)

    model.train()
    train_loss, mse_loss = [], []
    count = []
    for i in (pbar := tqdm(range(n_iter // (5 if reload_last_model else 1)))):
        h, h_ls = train_ch_gen.gen_channel_jit(snr)

        # Transition tensors to PyTorch for learning
        h, h_ls = torch.tensor(h.numpy()).to(dev), torch.tensor(h_ls.numpy()).to(dev)

        # Uncomment to inspect estimates vs ground-truth channel
        # ut.compare_ch_estimates([h_ls.cpu().numpy()[0, :], h.cpu().numpy()[0, :]],
        #                          ['LS', 'GT'], title=f'SNR = {snr} dB')

    h_hat = model(h_ls[... , ::2])
```

(continues on next page)

(continued from previous page)

```

loss = criterion(h_hat, h[... , ::(3-interp)])
optimizer.zero_grad(); loss.backward(); optimizer.step()

train_loss += [ut.db(loss.item())]
pbar.set_description(f"Iteration {i+1}/{n_iter}")
pbar.set_postfix_str(f"Training loss: {train_loss[-1]:.1f} dB")

last_model = save_model_path
torch.save(model.state_dict(), save_model_path)
ut.plot_losses([train_loss], ['train loss'], title=f'SNR = {snr} dB')

```

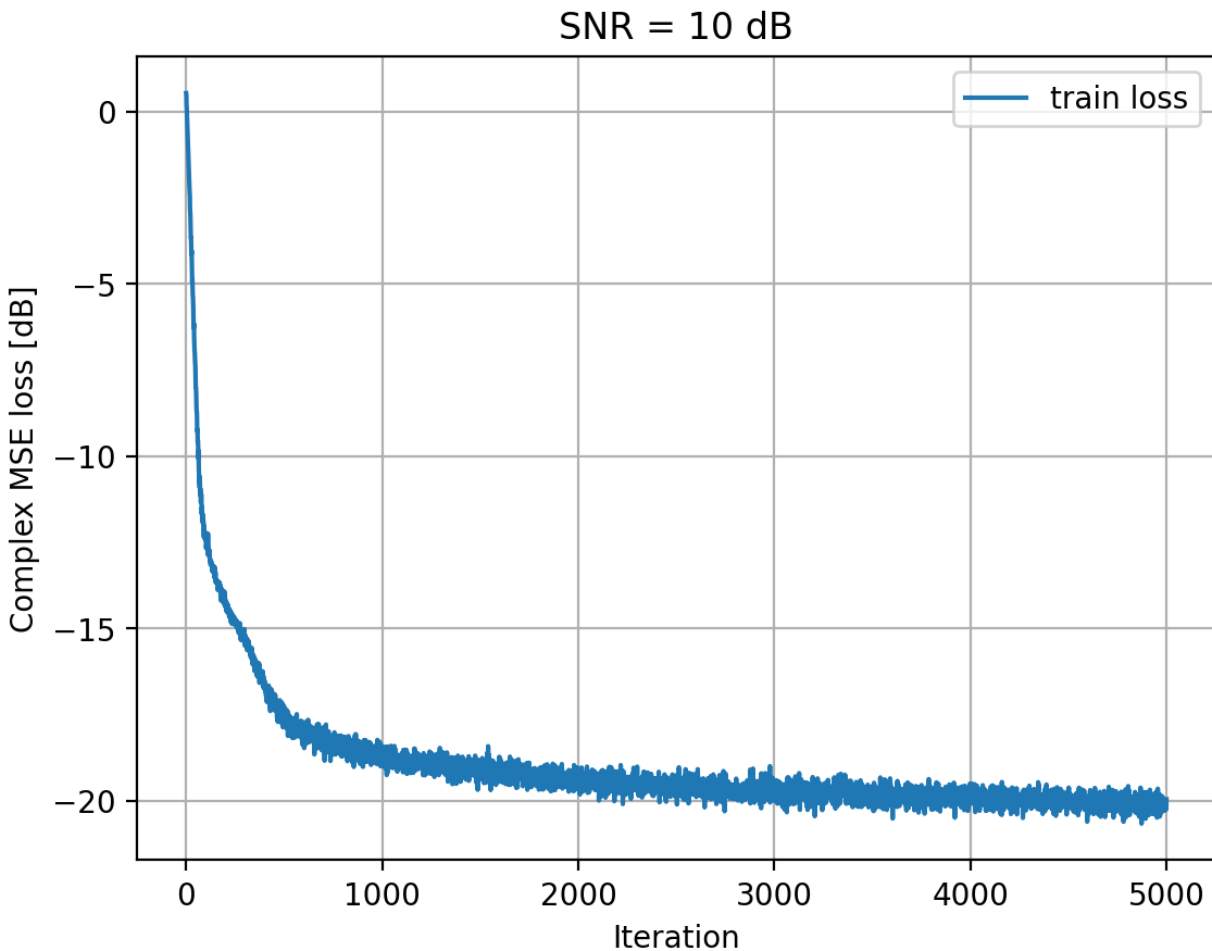
XLA can lead to reduced numerical precision. Use with care.

Training model for SNRs: 10 dB

```

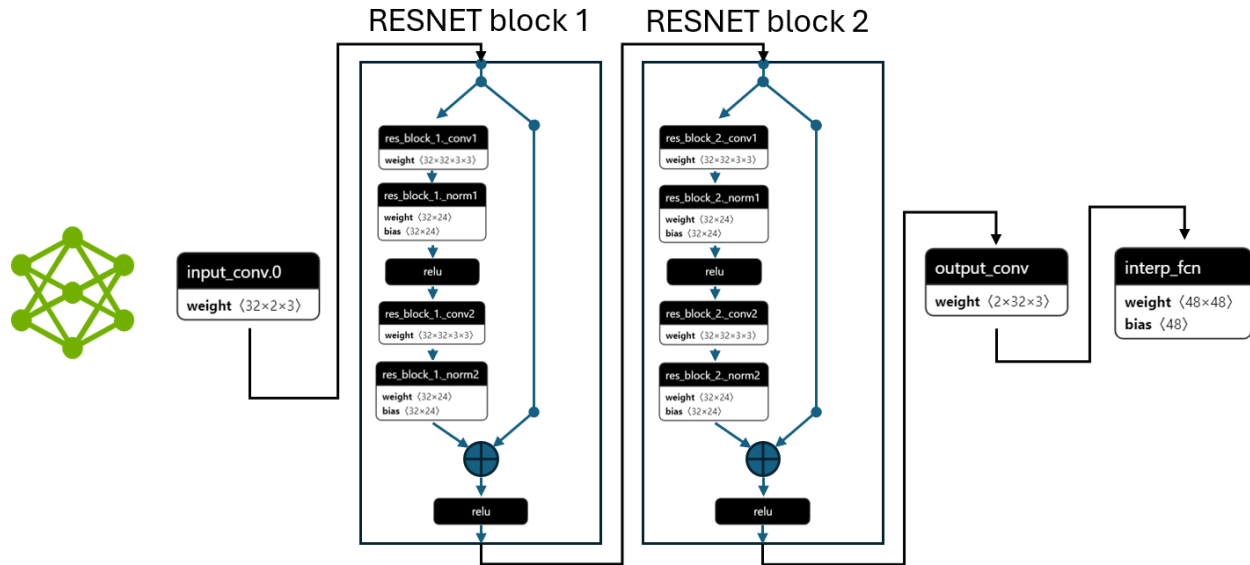
0%|
↪ | 0/5000 [00:00<?, ?it/s]WARNING: All
↪ log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1729133880.135412 9045 device_compiler.h:186] Compiled cluster using
↪ XLA! This line is logged at most once for the lifetime of the process.
Iteration 5000/5000: 100
↪ %| | 5000/5000
↪ [00:49<00:00, 101.48it/s, Training loss: -20.0 dB]

```



## 4.3.7.1.2 Testing channel estimation model

The model trained above is a deep convolutional network, made of RESNET layers, possibly with a fully connected layer at the end in case interpolation is enabled (`interp = 2`). The diagram for this network is presented below for an example user allocated 48 PRBs:



Now we evaluate this network using the LS channel estimates extracted from a PUSCH receiver, as opposed to manually extracted from the channel. The output of the network is compared with MS-MMSE.

```
[3]: train_dir = ut.get_model_training_dir(models_folder, training_ch_model,
                                         num_prbs, n_iter, batch_size)

snr_losses_ls = []           # Least Squares channel estimation losses
snr_losses_ml = []          # Machine learning channel estimation losses
snr_losses_ml2 = []         # Machine learning channel estimation losses (median)

snr_losses_ls_pyaerial = [] # LS from PyAerial
snr_losses_mmse_pyaerial = [] # MMSE from PyAerial
snr_losses_mmse_pyaerial2 = [] # MMSE from PyAerial (median)

# Channel generator for testing
test_ch_gen = SionnaChannelGenerator(num_prbs, testing_ch_model, batch_size=32)

# Create pyAerial channel estimate generator, using the Sionna Channel Generator
pyaerial_ch_est_gen = PyAerialChannelEstimateGenerator(test_ch_gen)

for snr_idx, snr in enumerate(test_snrs):
    print(f'Testing SNR {snr} dB')

    # Select model trained on the SNR closest to the test SNR
    snr_model_idx = np.argmin(abs(train_snrs - snr))
    snr_model = train_snrs[snr_model_idx]
    print(f'Testing model trained on SNR {snr_model}')

    # Load ML model
    model = ChannelEstimator(num_prbs*12//2, freq_interp_factor=interp).to(dev)
```

(continues on next page)

(continued from previous page)

```

model.load_state_dict(torch.load(ut.get_snr_model_path(train_dir, snr_model)))

criterion = ComplexMSELoss()

model.eval()
ls_loss, ml_loss, ls_loss_pyaerial, mmse_loss_pyaerial = [], [], [], []
with torch.no_grad():
    for i in tqdm(range(n_iter_test), desc='Testing LS & MS-MMSE in PyAerial'):
        # Internally generate channels, add noise, receive the DM-RS symbols and
        ↪ estimate the channel
        ls, mmse, gt = pyaerial_ch_est_gen(snr)

        # Evaluate pyAerial classic estimators
        for b in range(len(ls)):
            ls_loss_pyaerial += [ut.complex_mse_loss(ls[b], gt[b][::2])]
            mmse_loss_pyaerial += [ut.complex_mse_loss(mmse[b], gt[b])]

        # Evaluate ML approach
        h, h_ls = torch.tensor(gt).to(dev), torch.tensor(ls).to(dev)
        h_hat = model(h_ls) # h_ls has half the subcarriers of h already
        ls_loss += [criterion(h_ls, h[... , :2]).item()]
        ml_loss += [criterion(h_hat, h[... , :(3-interp)]).item()]

    # Compute means and medians of LS, LS+ML and MS-MMSE
    snr_losses_ls += [ut.db(np.mean(ls_loss))]
    snr_losses_ml += [ut.db(np.mean(ml_loss))]
    snr_losses_ml2 += [ut.db(np.median(ml_loss))]

    snr_losses_ls_pyaerial += [ut.db(np.mean(ls_loss_pyaerial))]
    snr_losses_mmse_pyaerial += [ut.db(np.mean(mmse_loss_pyaerial))]
    snr_losses_mmse_pyaerial2 += [ut.db(np.median(mmse_loss_pyaerial))]

    print(f'Avg. ML test loss for {snr} dB SNR is {snr_losses_ml[-1]:.1f} dB')

    # Plot CDFs of MSE losses
    ut.plot_annotated_cdfs([ml_loss, mmse_loss_pyaerial], ['LS+ML', 'MS-MMSE'],
                          title=f'MSE CDFs for SNR = {snr} dB')

```

XLA can lead to reduced numerical precision. Use with care.

Testing SNR -10.0 dB

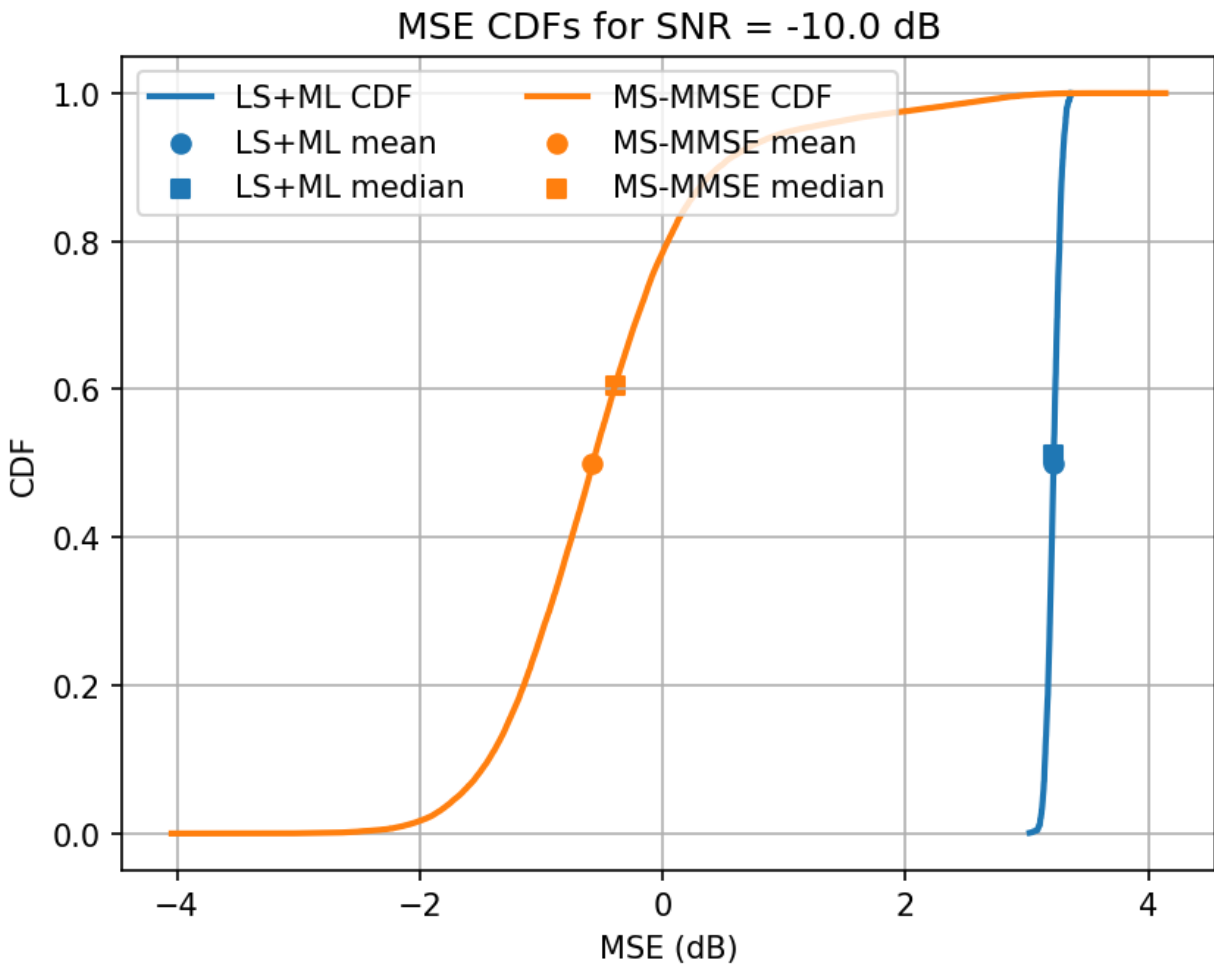
Testing model trained on SNR 10

Testing LS & MS-MMSE in PyAerial: 100

↪ % |

↪ 1000/1000 [01:11<00:00, 14.01it/s]

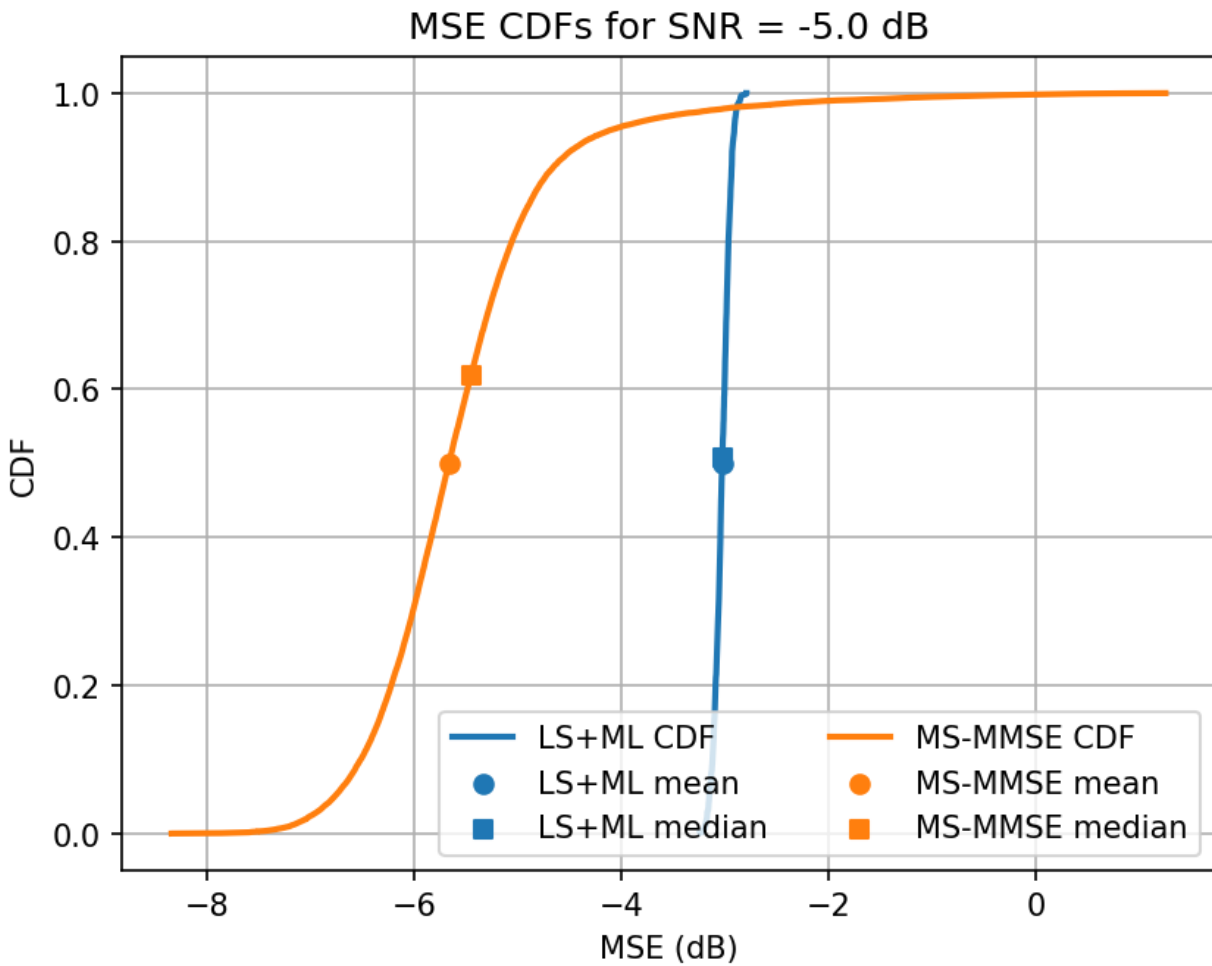
Avg. ML test loss for -10.0 dB SNR is 3.2 dB



Testing SNR -5.0 dB  
 Testing model trained on SNR 10

```
Testing LS & MS-MMSE in PyAerial: 100
↪ % |
↪ 1000/1000 [00:46<00:00, 21.36it/s]
```

Avg. ML test loss for -5.0 dB SNR is -3.0 dB



Testing SNR 0.0 dB

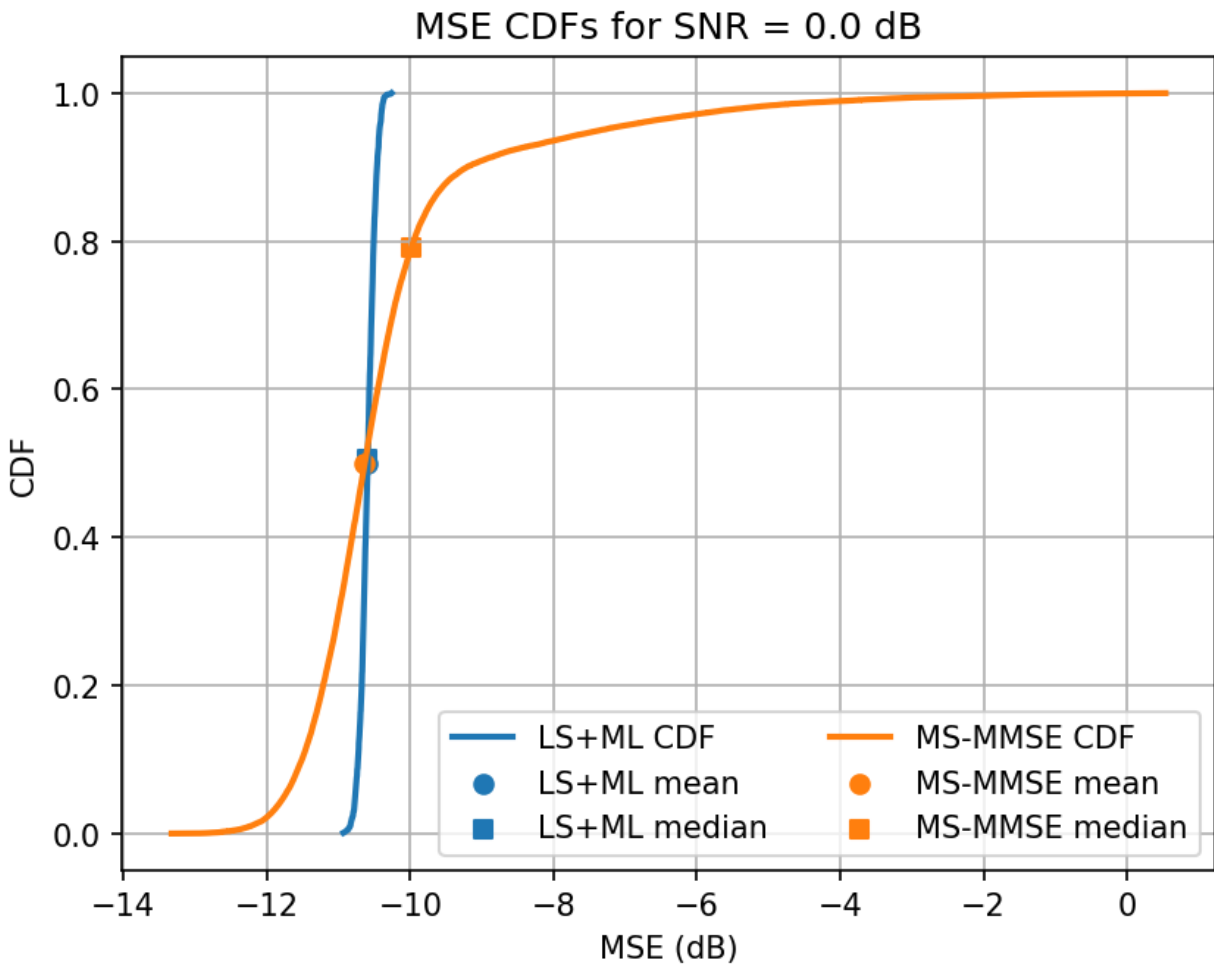
Testing model trained on SNR 10

Testing LS & MS-MMSE in PyAerial: 100

↪ % |

↪ 1000/1000 [00:47<00:00, 21.25it/s]

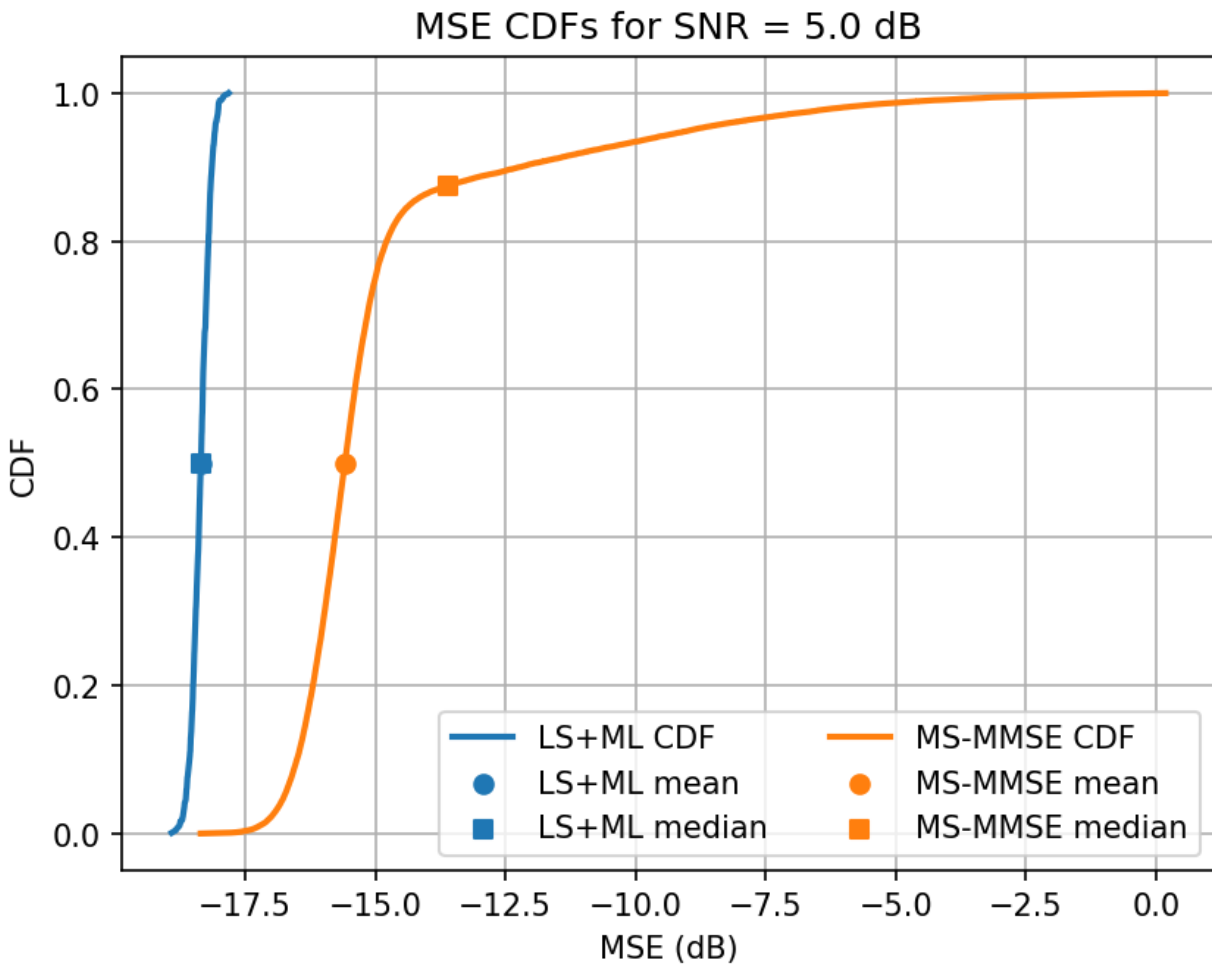
Avg. ML test loss for 0.0 dB SNR is -10.6 dB



```

Testing SNR 5.0 dB
Testing model trained on SNR 10
Testing LS & MS-MMSE in PyAerial: 100
↪ % |
↪ 1000/1000 [00:46<00:00, 21.39it/s]
Avg. ML test loss for 5.0 dB SNR is -18.4 dB
    
```





Testing SNR 10.0 dB

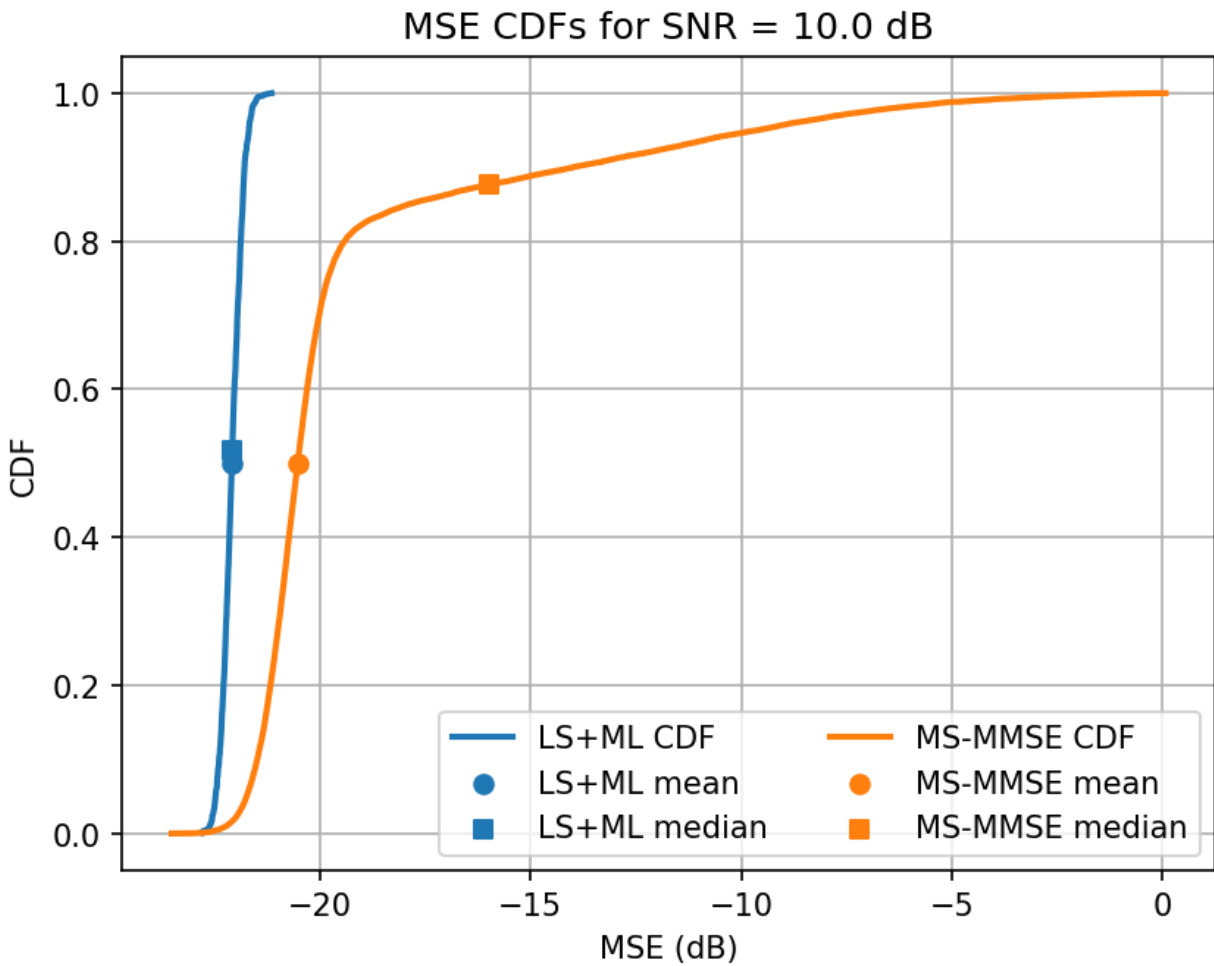
Testing model trained on SNR 10

Testing LS & MS-MMSE in PyAerial: 100

↪ % |

↪ 1000/1000 [00:46<00:00, 21.33it/s]

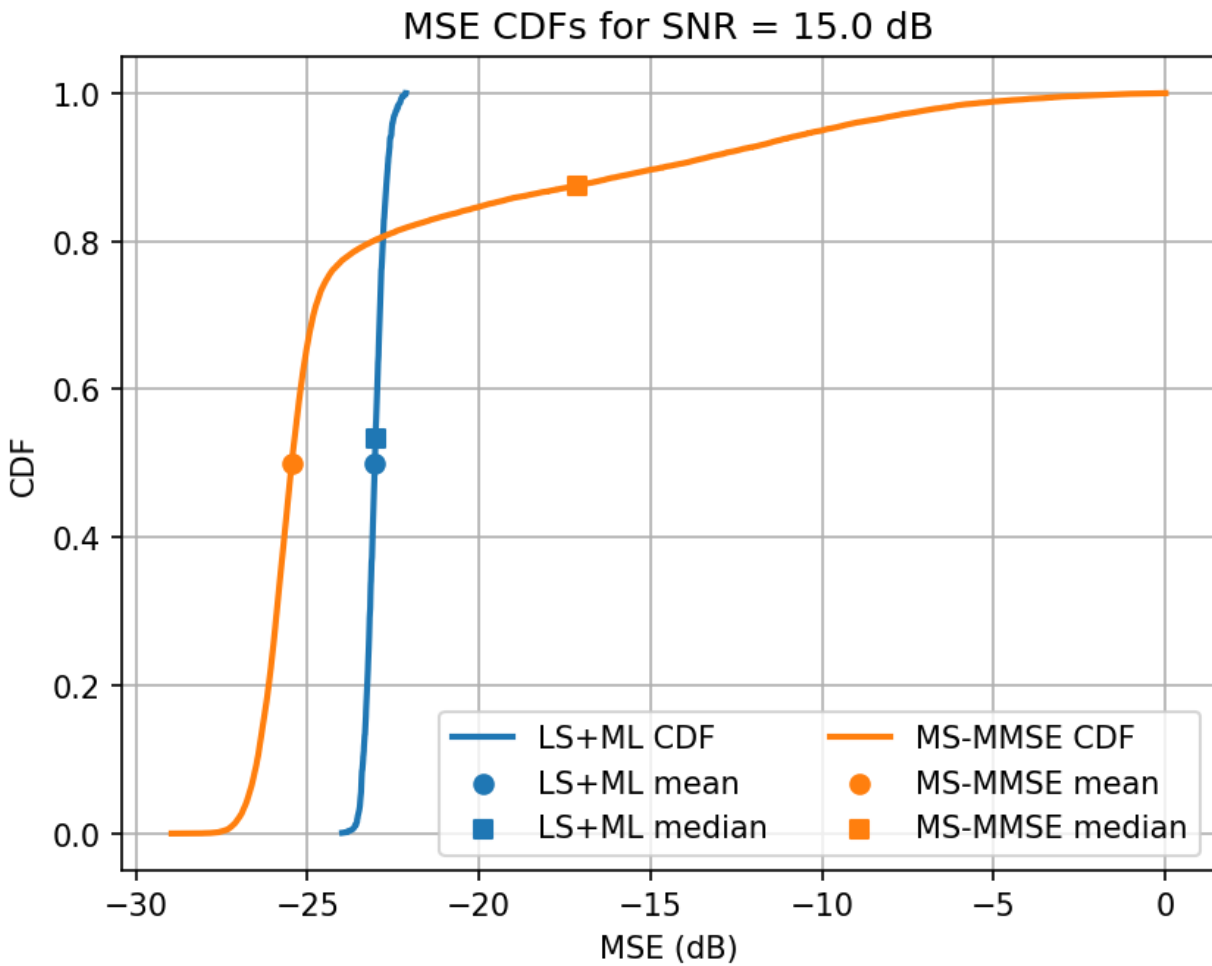
Avg. ML test loss for 10.0 dB SNR is -22.1 dB



Testing SNR 15.0 dB  
 Testing model trained on SNR 10

```
Testing LS & MS-MMSE in PyAerial: 100
↪ % |
↪ 1000/1000 [00:46<00:00, 21.39it/s]
```

Avg. ML test loss for 15.0 dB SNR is -23.0 dB



Testing SNR 20.0 dB

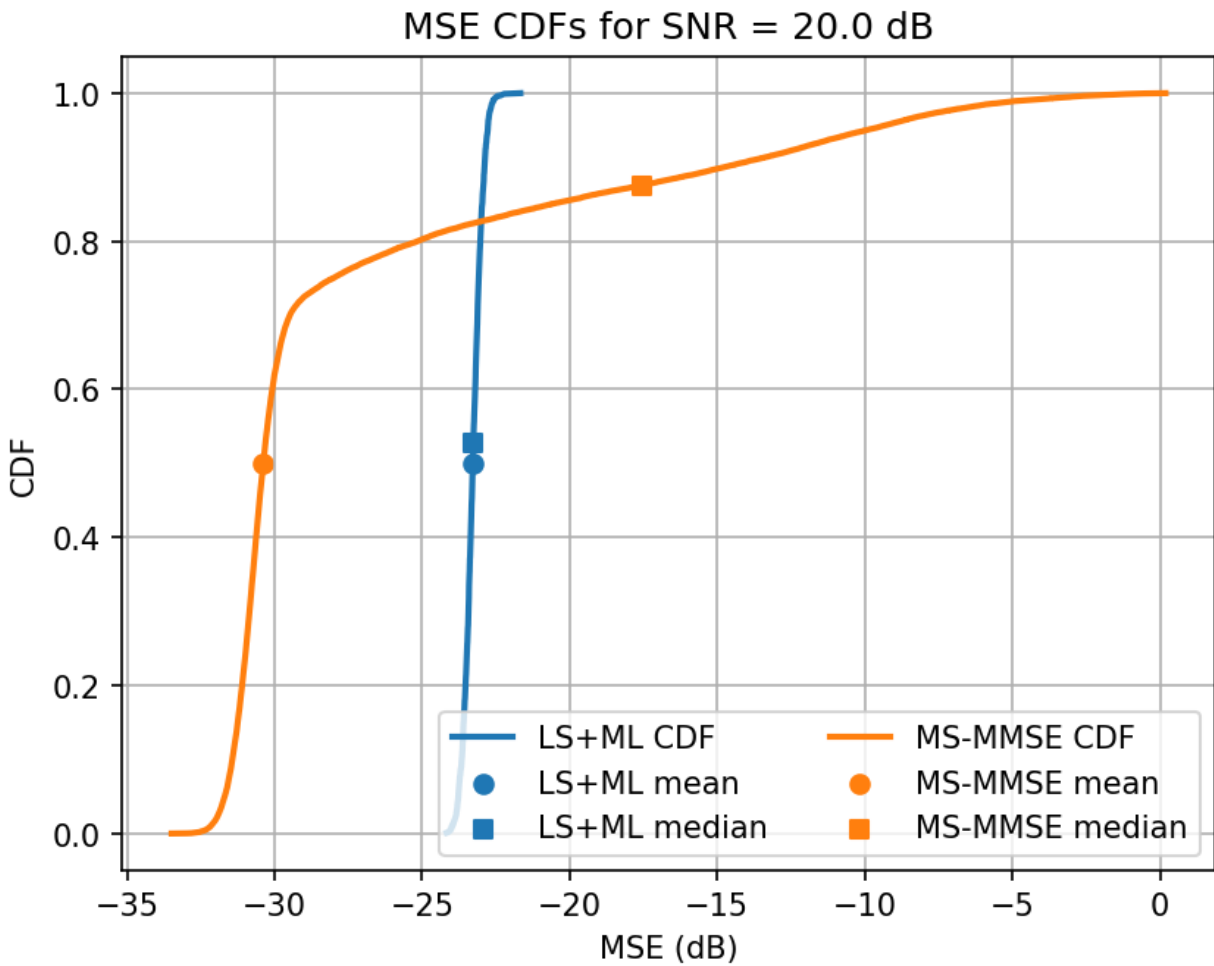
Testing model trained on SNR 10

Testing LS & MS-MMSE in PyAerial: 100

↪ % |

↪ 1000/1000 [00:46<00:00, 21.36it/s]

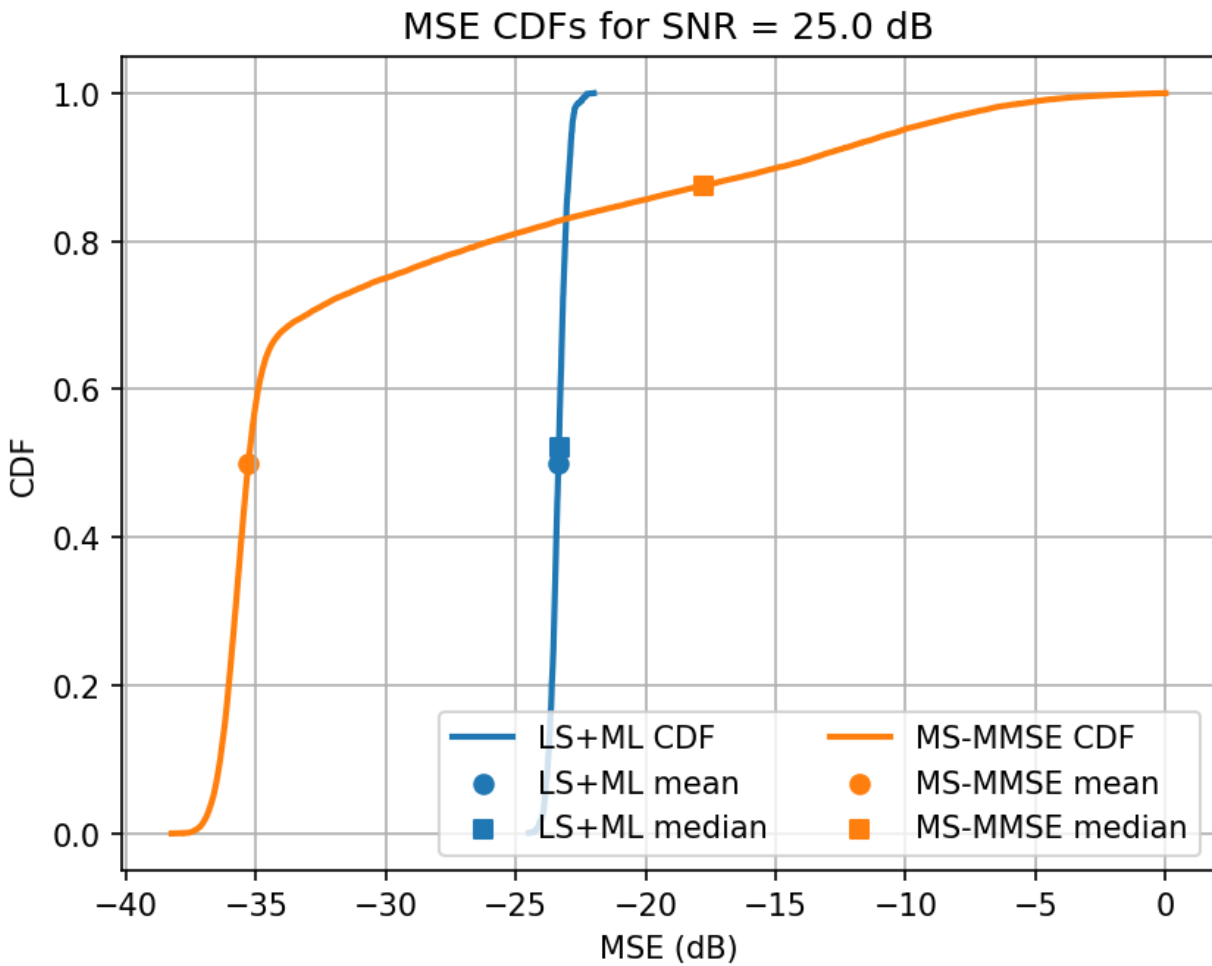
Avg. ML test loss for 20.0 dB SNR is -23.3 dB



Testing SNR 25.0 dB  
 Testing model trained on SNR 10

```
Testing LS & MS-MMSE in PyAerial: 100
↪ % |
↪ 1000/1000 [00:46<00:00, 21.38it/s]
```

Avg. ML test loss for 25.0 dB SNR is -23.3 dB



Testing SNR 30.0 dB

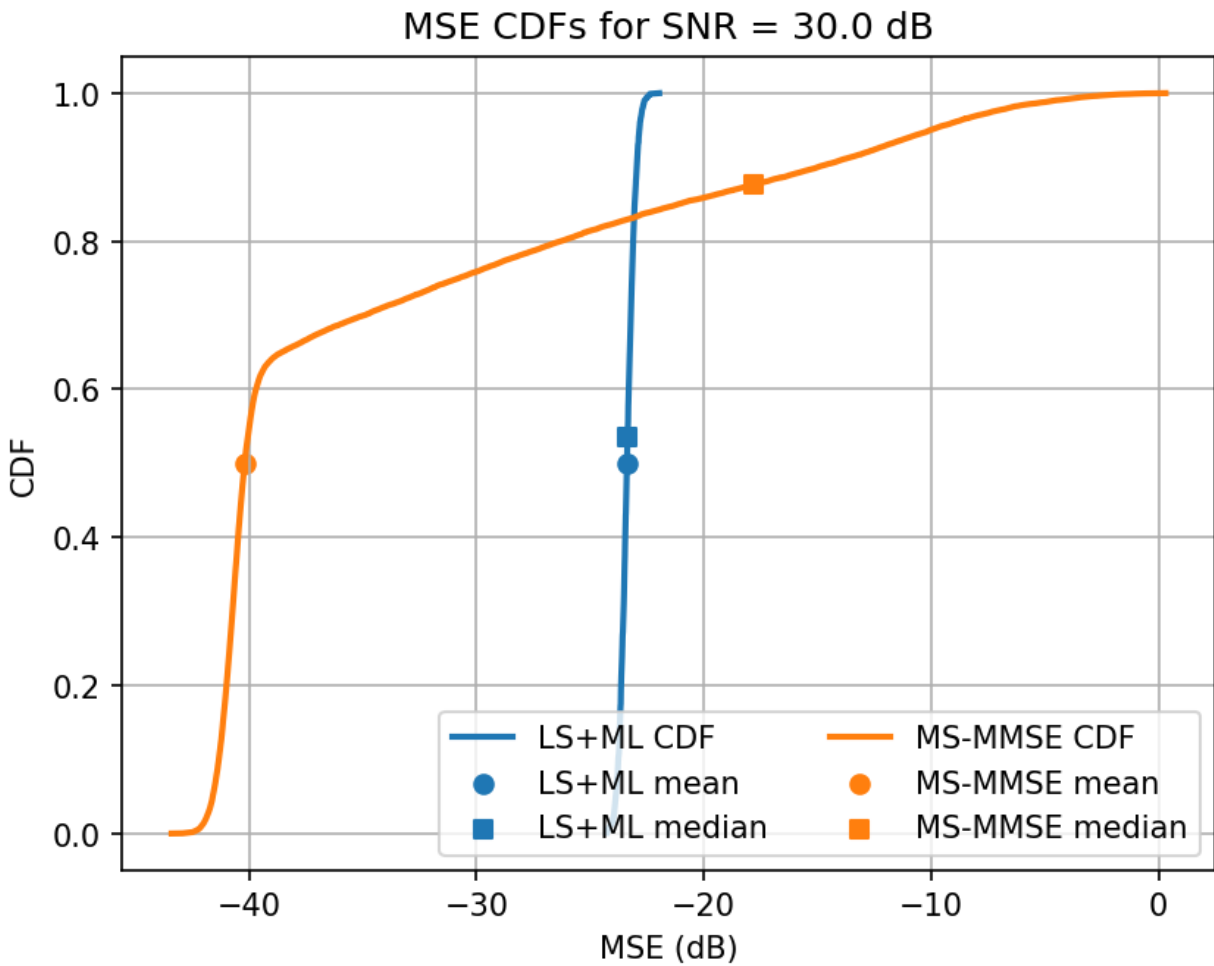
Testing model trained on SNR 10

Testing LS & MS-MMSE in PyAerial: 100

↪ % |

↪ 1000/1000 [00:46<00:00, 21.36it/s]

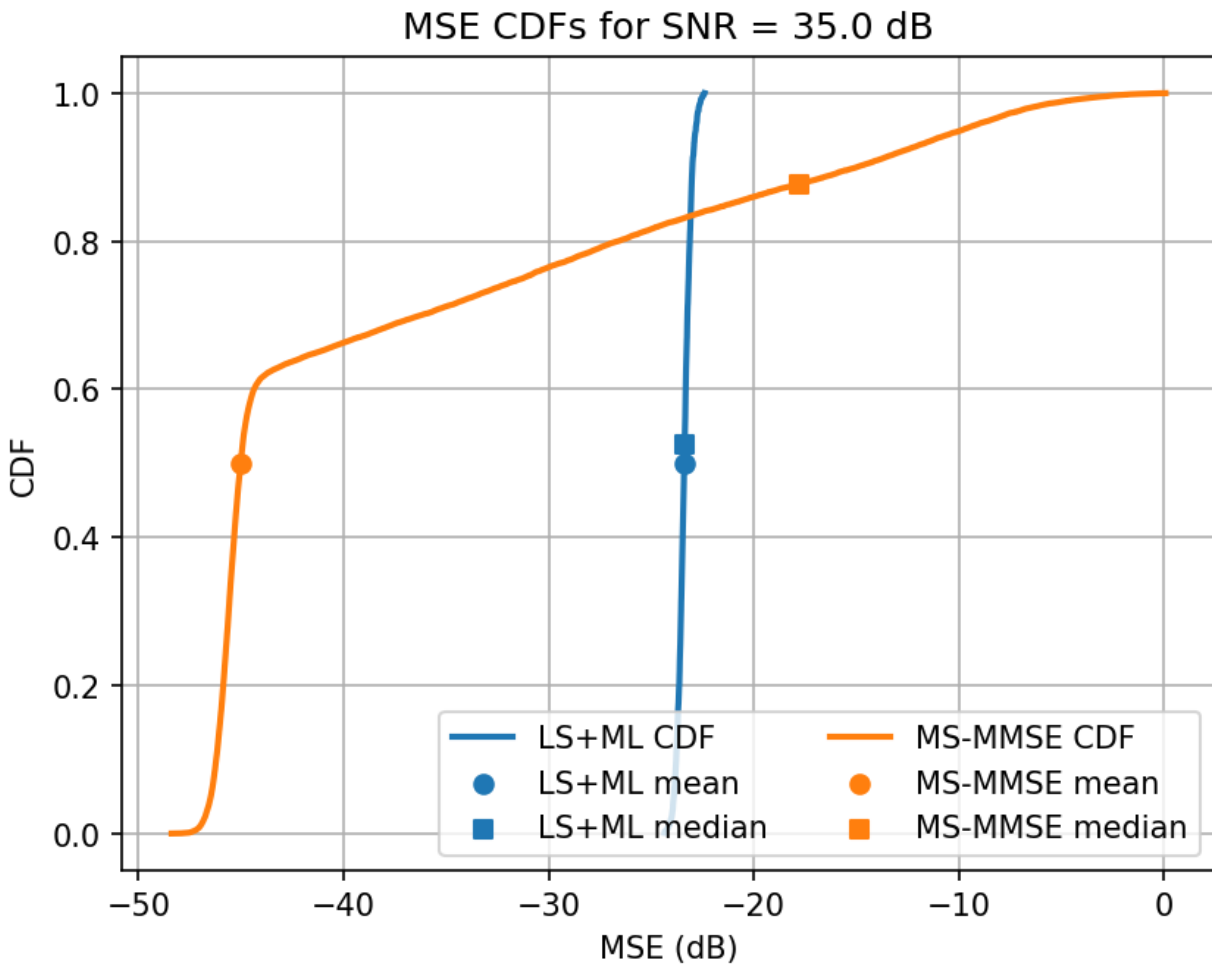
Avg. ML test loss for 30.0 dB SNR is -23.4 dB



Testing SNR 35.0 dB  
 Testing model trained on SNR 10

```
Testing LS & MS-MMSE in PyAerial: 100
↪ % |
↪ 1000/1000 [00:46<00:00, 21.39it/s]
```

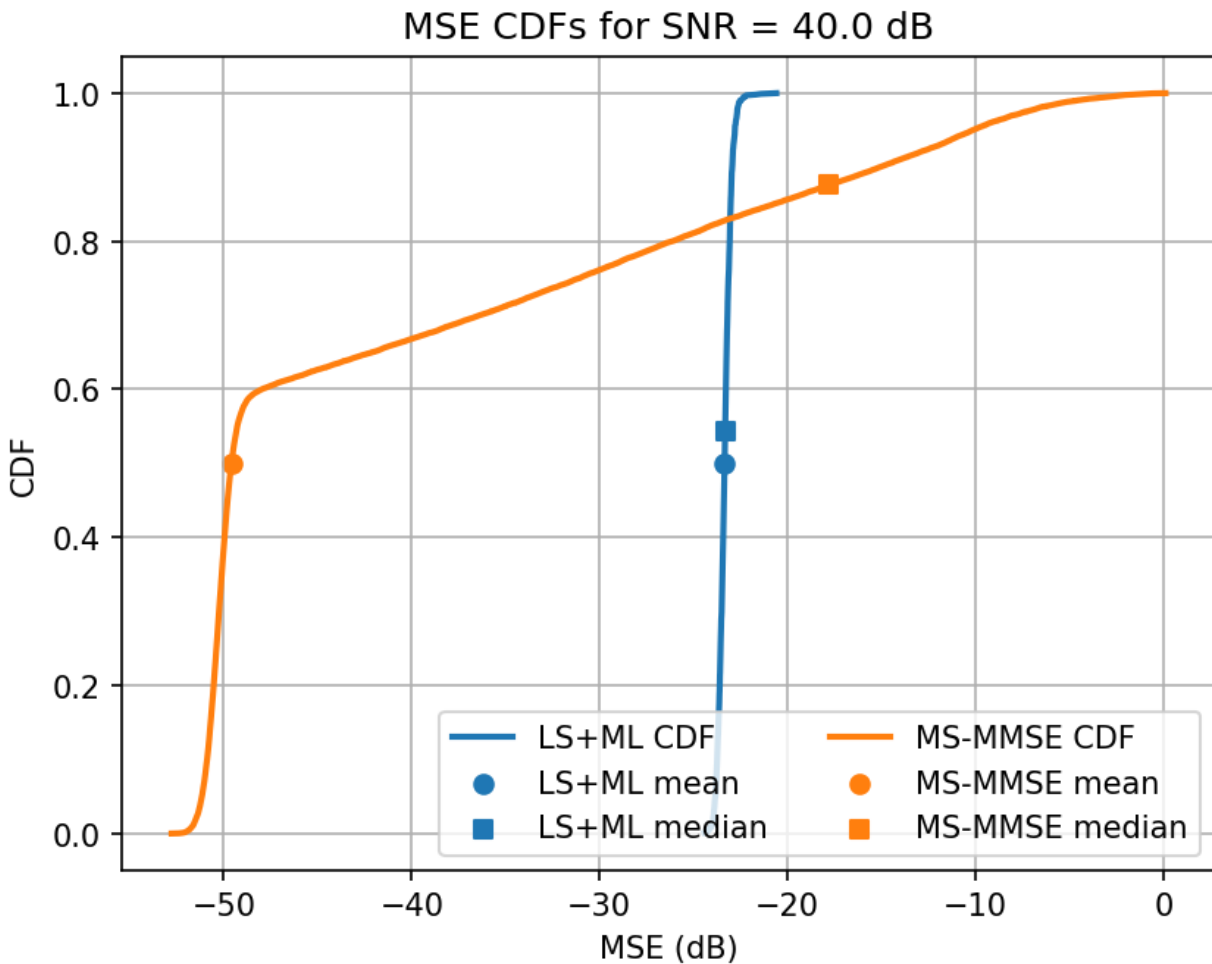
Avg. ML test loss for 35.0 dB SNR is -23.4 dB



Testing SNR 40.0 dB  
 Testing model trained on SNR 10

```
Testing LS & MS-MMSE in PyAerial: 100
↪ % |
↪ 1000/1000 [00:46<00:00, 21.36it/s]
```

Avg. ML test loss for 40.0 dB SNR is -23.3 dB



**Observation:** When we fine tune our training, we see the ML model outperforming the MS-MMSE approach for most SNRs. The performance decays slightly when interpolation is necessary. Additionally, the ML seems more reliable for a wider class of channels. The variance of estimates is lower for ML, on average, it's performance saturates for high SNRs, even if the median continues to decay.

#### 4.3.7.1.3 Plot comparison across SNRs

**Requirement:** `test_snrs` must have more than one element. Read below why we would want to do this.

**Model switching depending on SNR:** One of the challenges in channel estimation is having it work across SNRs. Lower SNRs have higher channel estimation mean squared error (MSE), which influences more heavily the loss of these samples in machine learning models, thus leading the model learn only low-SNR channels. One way to avoid this problem is to do a model-switching approach. In model-switching, each model is trained for a single SNR and use the model that has the closest SNR to the SNR of the user.

Note that this approach requires a *sufficiently good* estimate of the SNR so the correct model is chosen. Usually, acquiring such an estimate is not difficult - for example, using the MMSE channel estimate should have more resolution than needed. As such, here we assume the SNR of the user is known and the closest model is selected.

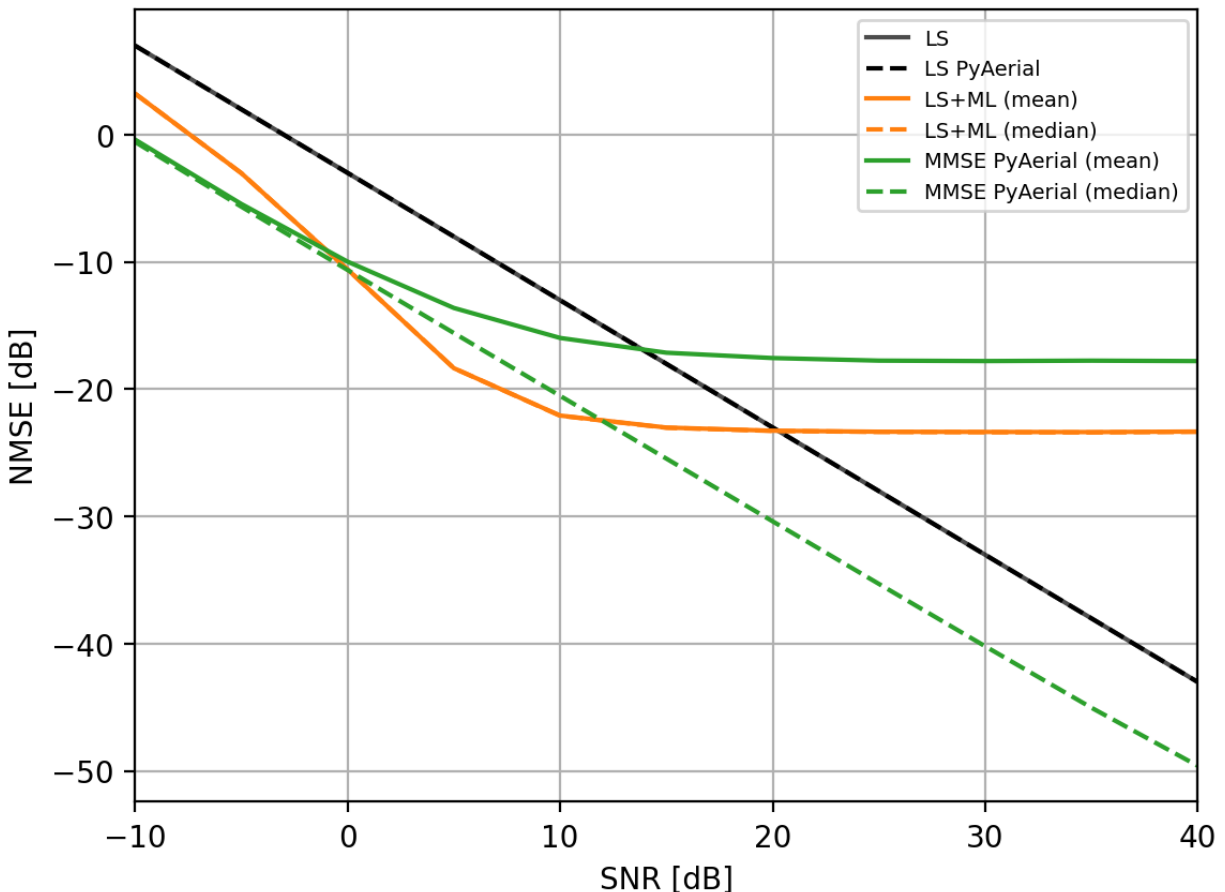


If we set `train_snrns = [-10, 0, 10, 20, 30, 40]` and `test_snrns = [-10, -5, 0, 5, 10, 15, ..., 40]`, then we will see that the model trained for an SNR of -10 dB is also used to estimate channels at -5 dB, and the model trained for 0 dB is also used at 5 dB, etc. This leads to a higher MSE in SNRs divisible by 5 but not 10.

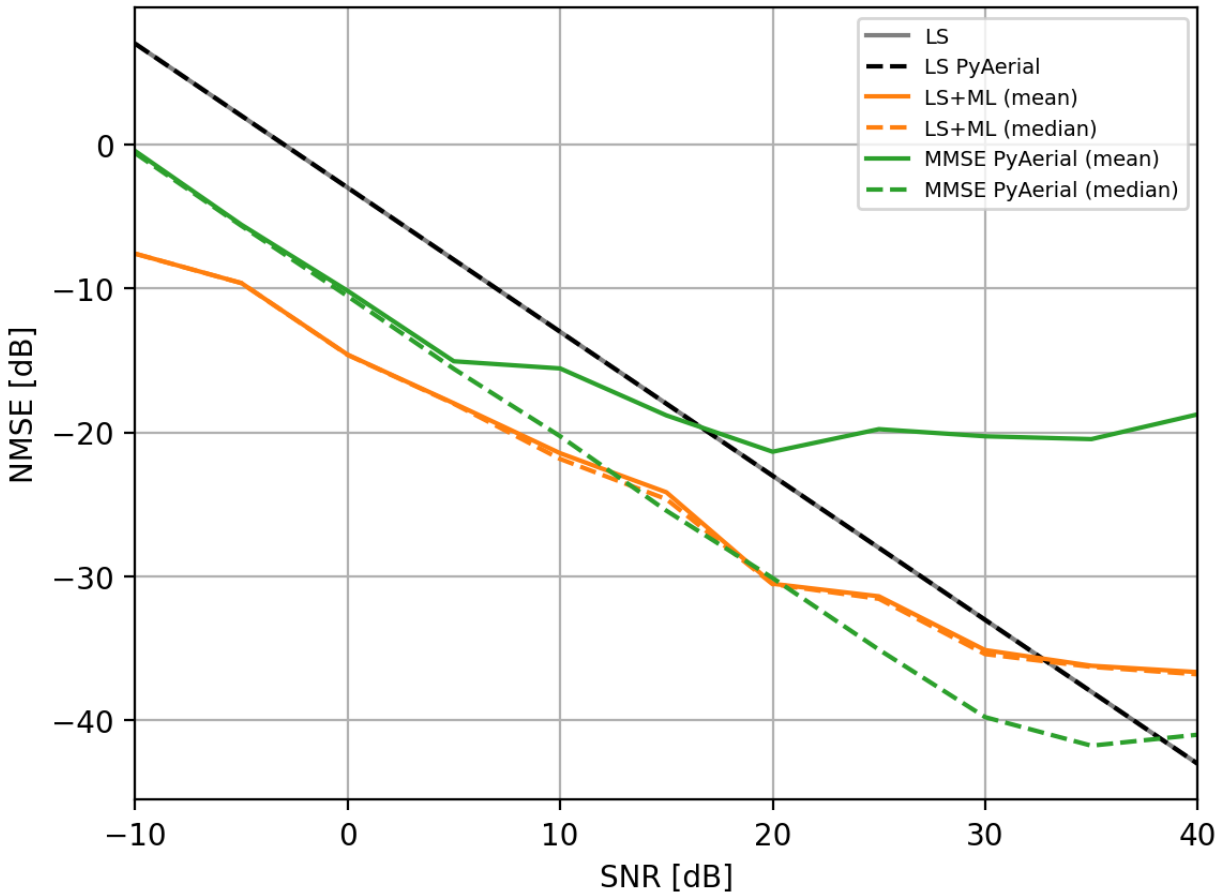
```
[4]: plt.figure(dpi=200)

plt.plot(test_snrns, snr_losses_ls,          '-', label='LS',
         ↪color='k', alpha=.7)
plt.plot(test_snrns, snr_losses_ls_pyaerial,  '--', label='LS PyAerial',
         ↪color='k')
plt.plot(test_snrns, snr_losses_ml,          '-', label='LS+ML (mean)',
         ↪color='tab:orange')
plt.plot(test_snrns, snr_losses_ml2,        '--', label='LS+ML (median)',
         ↪color='tab:orange')
plt.plot(test_snrns, snr_losses_mmse_pyaerial, '-', label='MMSE PyAerial (mean)',
         ↪color='tab:green')
plt.plot(test_snrns, snr_losses_mmse_pyaerial2, '--', label='MMSE PyAerial (median)',
         ↪color='tab:green')

plt.xlabel('SNR [dB]')
plt.ylabel('NMSE [dB]')
plt.xlim((min(test_snrns), max(test_snrns)))
plt.legend(fontsize=7)
plt.grid()
plt.show()
```



Below is an example of this plot for the case `interp = 2` and 48 PRBs.



Some noteworthy gains compared to MS-MMSE:

- ▶ 5-8 dB gain for SNRs  $\in [-10, 0]$  dB
- ▶ 3-5 dB gain for SNRs  $\in [0, 10]$  dB
- ▶ 0-3 dB gain for SNRs  $\in [10, 20]$  dB
- ▶ after 10 dB, the ML model is comparatively more reliable, with a more predictable CDF and better average performance

Note that this approach is expected to work better for higher PRB allocations. Likely due to being able to extract more information from the channel. The model also has better performance when `interp = 1` since interpolation is not necessary, which makes the network harder to train by the addition of the fully connected at the end.

#### 4.3.7.1.3.1 Considerations for Real Deployments

For such approach to work in real deployments, it requires two additional steps we choose to omit here for simplicity:

- ▶ SNR estimation: required to estimate the optimized model to perform channel estimation. Here, we consider the SNR is known.
- ▶ PRB parallelization: during inference, the PRB parallelizer would split the LS estimates (e.g. 78 PRBs) into chunks that could be processed in parallel by the trained models of different sizes, and then put back together. As an example, if we trained models for {1, 4, 16} PRBs, the 78 PRB estimate would result in 4 batches for the 16 PRB model, 3 batches for the 4 PRB and 2 batches for the 1 PRB ( $4 * 16 + 3 * 4 + 2 * 1 = 64 + 12 + 2 = 78$ )

#### 4.3.7.1.3.2 Assessing System-level Performance in the Aerial Omniverse Digital Twin

This notebook can be used to generate models compatible with the machine learning example of PUSCH channel estimation in the AODT. As long as the `models_folder` variable is kept constant across runs, a single folder will be populated with the correct structure for multiple SNRs and PRBs. As mentioned in the AODT user guide, this folder will then need to be moved to a directory accessible by the AODT backend, and the `config_est.ini` file populated with the absolute path to the folder.

##### Benefits of using PyAerial as a bridge to AODT

- ▶ AODT uses a high-performance EM solver for computing raytracing propagation simulations. Raytracing is necessary for studying ML approaches in site-specific settings, offering insight and explainability to edge-cases previously unavailable in stochastic simulations.
- ▶ AODT RAN simulations use the same software running on the same hardware deployed in the real world. This unprecedented combination creates an accurate system representation, giving researchers the possibility to design new features (AI/ML powered or not) and assess their network-wide end-to-end impact.
- ▶ PyAerial currently provides a Python interface only to cuPHY, the PHY layer of Aerial. As such, comparisons beyond the PHY are not possible in PyAerial, and the last link-level quantity that can be computed is block error rates. The AODT, on the other hand, integrates both Aerial's cuPHY and cuMAC, allowing researchers to measure how channel estimation impacts higher layers.

For more information about how to run this ML channel estimation in AODT, see the [AODT user guide](#).

## 4.3.8. Channel estimation on transmissions captured using Aerial Data Lake

This example shows how to query PUSCH data from an Aerial Data Lake database and perform channel estimation on that PUSCH data using pyAerial.

### Note

This notebook requires that the clickhouse server used by Aerial Data Lake is running, and that the example data has been imported into a database. Refer to the Aerial Data Lake installation docs on how to do this.

#### 4.3.8.1 Using pyAerial for channel estimation on Aerial Data Lake data

This example shows how to use the pyAerial bindings to run cuPHY GPU accelerated channel estimation for 5G NR PUSCH. 5G NR PUSCH data is read from an example over the air captured PUSCH dataset collected and stored using Aerial Data Lake, and the channel is estimated using pyAerial and cuPHY based on the corresponding PUSCH parameters.

**Note:** This example requires that the clickhouse server is running and that the example data has been stored in the database. Refer to the Aerial Data Lake documentation on how to do this.

```
[1]: # Check platform.
import platform
if platform.machine() not in ['x86_64', 'aarch64']:
    raise SystemExit("Unsupported platform!")
```

##### 4.3.8.1.1 Imports

```
[2]: import math
import os
os.environ["CUDA_VISIBLE_DEVICES"] = "0"

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Connecting to clickhouse on remote server
import clickhouse_connect

# Import the channel estimator and some utilities for converting
# the DMRS fields in the right format from the SCF FAPI format that the dataset follows.
from aerial.phy5g.algorithms import ChannelEstimator
from aerial.util.fapi import dmrs_fapi_to_bit_array
```

##### 4.3.8.1.2 Data

We use an example dataset which has been captured from a real over the air PUSCH transmission. The “fapi” table in the database contains the metadata for each PUSCH transmission and the “fh” table contains all of the samples for that slot.

```
[3]: # Create the pyAerial (cuPHY) channel estimator.
num_ues = 1
num_rx_ant = 4
channel_estimator = ChannelEstimator(num_rx_ant=num_rx_ant)

# Connect to the local database
client = clickhouse_connect.get_client(host='localhost')

# Pick some pusched records from the database
pusched_records = client.query_df('select * from fapi order by TsTais limit 10')
```

#### 4.3.8.1.3 Run channel estimation

From the PUSCH record we extract the PUSCH DMRS parameters and use the TAI time entry to select the IQ samples for that slot. Channel estimation is then run using the extracted parameters, and the absolute values of the estimated channels are plotted in the same figure.

```
[4]: for index, pusch_record in pusch_records.iterrows():
    query = f"""select TsTaiNs, fhData from fh where
            TsTaiNs == {pusch_record.TsTaiNs.timestamp()}
            """

    fh = client.query_df(query)

    # Make sure that the fronthaul database is complete for the SFN.Slot we've chosen
    if fh.index.size < 1:
        pusch_records = pusch_records.drop(index)
        continue;

    fh_samp = np.array(fh['fhData'][0], dtype=np.float32)
    rx_slot = np.swapaxes(fh_samp.view(np.complex64).reshape(4, 14, 273*12), 2, 0)

    # Extract all the needed parameters from the PUSCH record.
    slot = int(pusch_record.Slot)
    rntis = [pusch_record.rnti]
    layers = [pusch_record.nrOfLayers]
    start_prb = pusch_record.rbStart
    num_prbs = pusch_record.rbSize
    start_sym = pusch_record.StartSymbolIndex
    num_symbols = pusch_record.NrOfSymbols
    scids = [int(pusch_record.SCID)]
    data_scids = [pusch_record.dataScramblingId]
    dmrs_scrm_id = pusch_record.ulDmrsScramblingId
    num_dmrs_cdm_grps_no_data = pusch_record.numDmrsCdmGrpsNoData
    dmrs_syms = dmrs_fapi_to_bit_array(int(pusch_record.ulDmrsSymbPos))
    dmrs_ports = [pusch_record.dmrsPorts]
    dmrs_max_len = 1
    dmrs_add_ln_pos = 2
    num_subcarriers = num_prbs * 12
    mcs_tables = [pusch_record.mcsTable]
    mcs_indices = [pusch_record.mcsIndex]
    coderates = [pusch_record.targetCodeRate/10.]
    tb_sizes = [pusch_record.TBSize]
    mod_orders = [pusch_record.qamModOrder]
    tb_input = np.array(pusch_record.pduData)

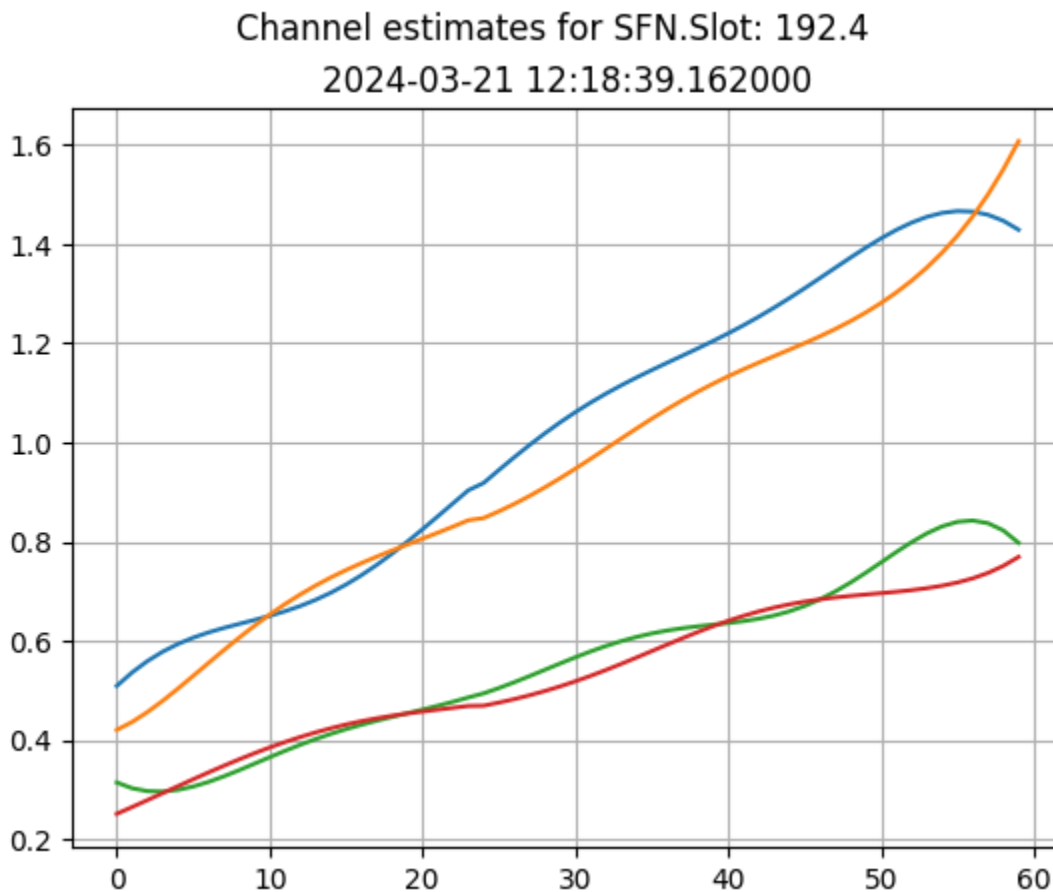
    # Run PyAerial (cuPHY) channel estimation.
    ch_est = channel_estimator.estimate(
        rx_slot=rx_slot,
        num_ues=num_ues,
        layers=layers,
        scids=scids,
        slot=slot,
        dmrs_ports=dmrs_ports,
        dmrs_syms=dmrs_syms,
        dmrs_scrm_id=dmrs_scrm_id,
        dmrs_max_len=dmrs_max_len,
```

(continues on next page)

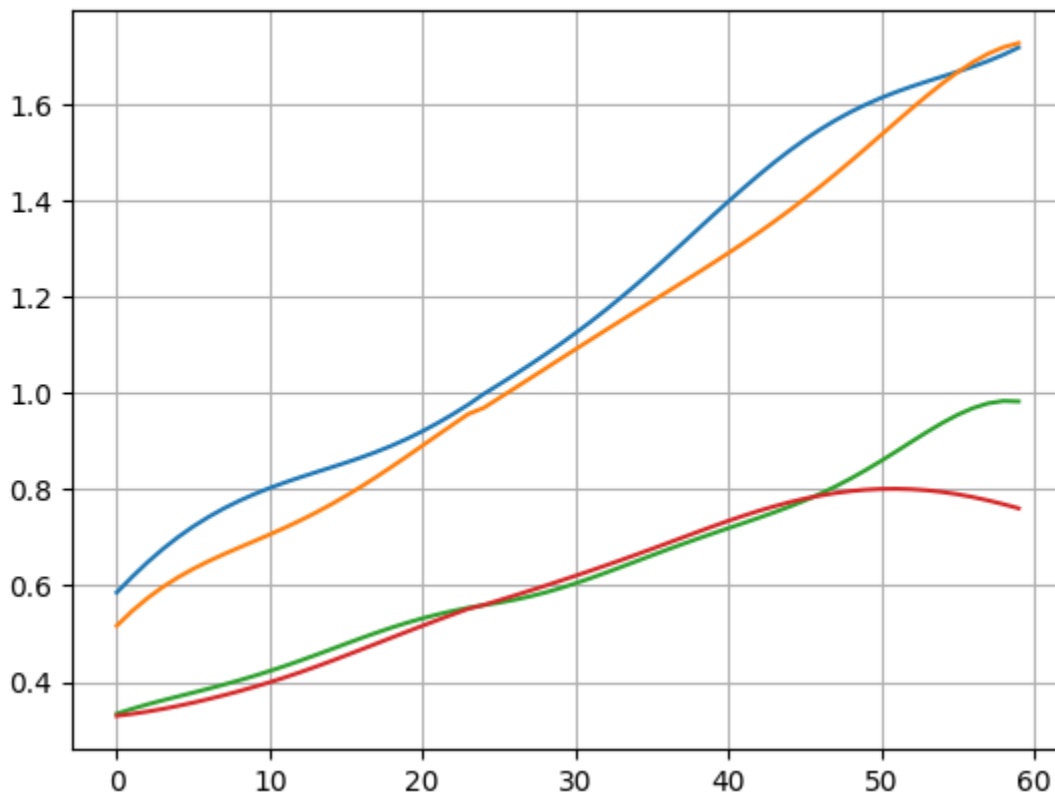
(continued from previous page)

```
dmrs_add_ln_pos=dmrs_add_ln_pos,
num_dmrs_cdm_grps_no_data=num_dmrs_cdm_grps_no_data,
start_prb=start_prb,
num_prbs=num_prbs,
prg_size=1,
num_ul_streams=1,
start_sym=start_sym,
num_symbols=num_symbols
)

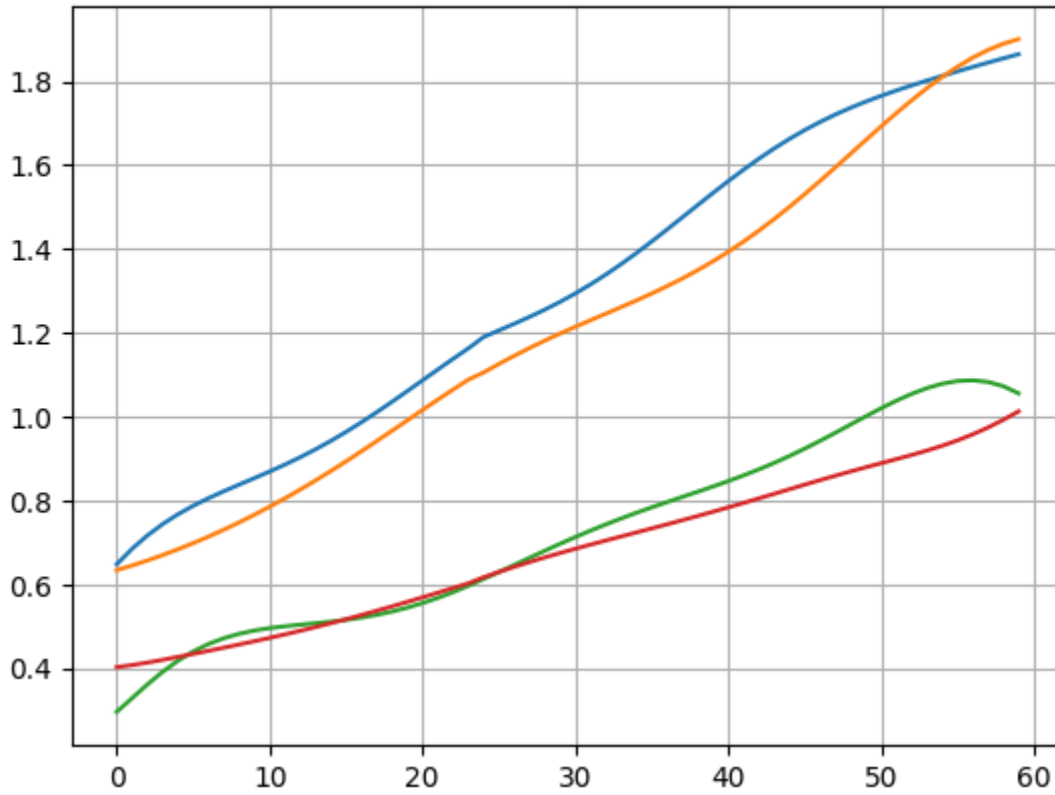
fig, axs = plt.subplots(1)
fig.suptitle("Channel estimates for SFN.Slot: "+str(pusch_record.SFN)+"."
↪ "+str(pusch_record.Slot))
axs.set_title(pusch_record.TsTaiNs)
for ant in range(4):
    axs.plot(np.abs(ch_est[0][ant, 0, :, 0]))
axs.grid(True)
plt.show()
```



Channel estimates for SFN.Slot: 194.14  
2024-03-21 12:18:39.187000

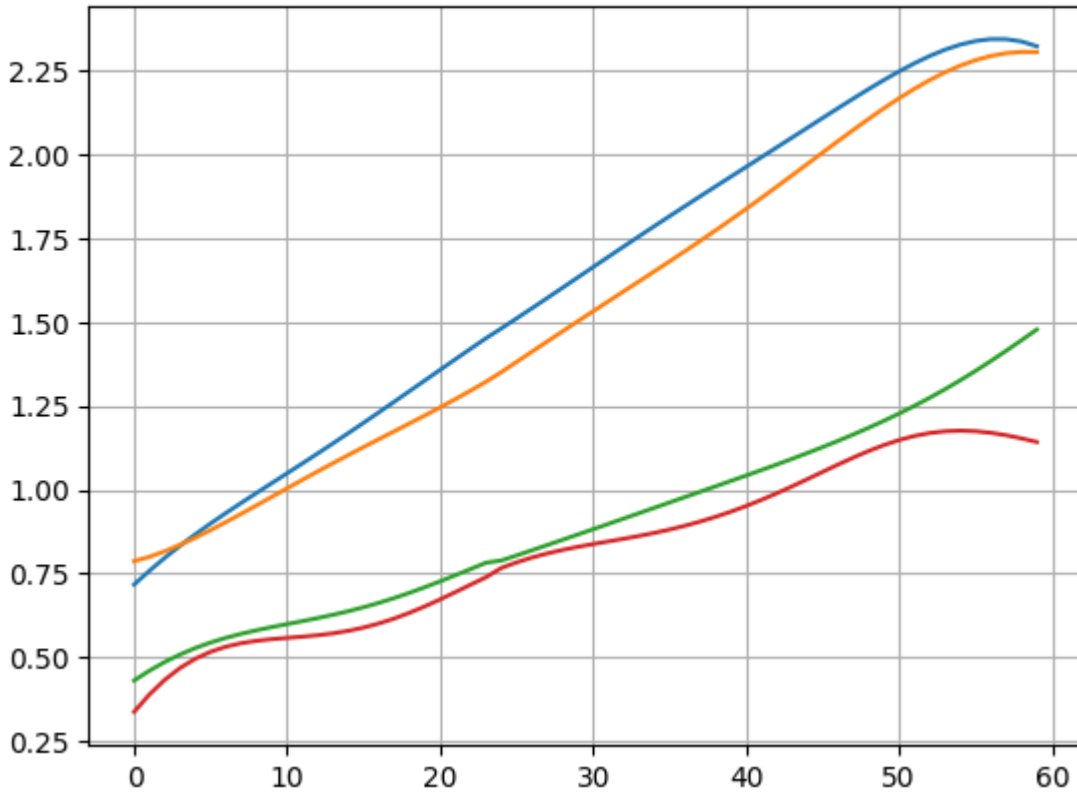


Channel estimates for SFN.Slot: 195.4  
2024-03-21 12:18:39.192000

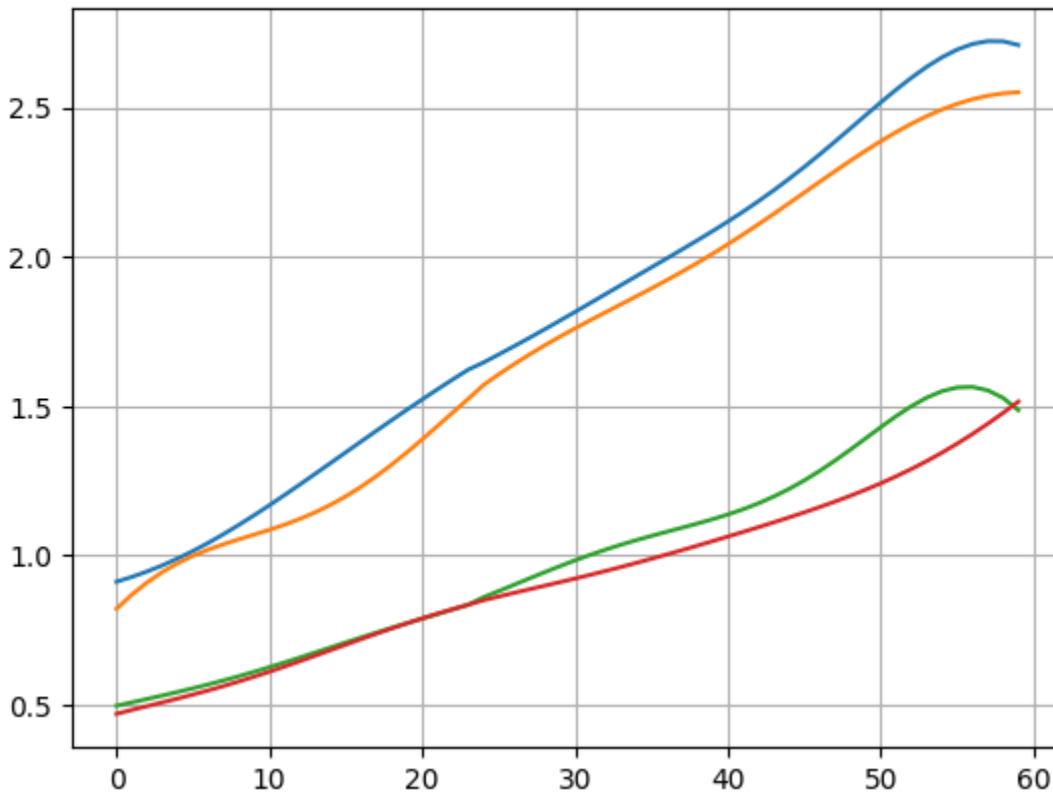




Channel estimates for SFN.Slot: 201.4  
2024-03-21 12:18:39.252000



Channel estimates for SFN.Slot: 209.4  
2024-03-21 12:18:39.332000



For more information, refer to the [Aerial Data Lake section](#).

## 4.3.9. Decoding PUSCH transmissions captured using Aerial Data Lake

### 4.3.9.1 Using pyAerial for PUSCH decoding on Aerial Data Lake data

This example shows how to use the pyAerial bindings to run cuPHY GPU accelerated PUSCH decoding for 5G NR PUSCH. The 5G NR PUSCH data is read from an example over the air captured PUSCH dataset collected and stored using Aerial Data Lake. Building a PUSCH receiver using pyAerial is demonstrated in two ways, first by using a fully fused, complete, PUSCH receiver called from Python using just a single function call. The same is then achieved by building the complete PUSCH receiver using individual separate Python function calls to individual PUSCH receiver components.

**Note:** This example requires that the clickhouse server is running and that the example data has been stored in the database. Refer to the Aerial Data Lake documentation on how to do this.

```
[1]: # Check platform.  
import platform
```

(continues on next page)

(continued from previous page)

```
if platform.machine() not in ['x86_64', 'aarch64']:
    raise SystemExit("Unsupported platform!")
```

#### 4.3.9.1.1 Imports

```
[2]: import math
import os
os.environ["CUDA_VISIBLE_DEVICES"] = "0"

import numpy as np
import pandas as pd
from IPython.display import Markdown
from IPython.display import display

# Connecting to clickhouse on remote server
import clickhouse_connect

# Plotting with Bokeh.
import matplotlib.pyplot as plt

# pyAerial imports
from aerial.phy5g.algorithms import ChannelEstimator
from aerial.phy5g.algorithms import ChannelEqualizer
from aerial.phy5g.algorithms import NoiseIntfEstimator
from aerial.phy5g.algorithms import Demapper
from aerial.phy5g.ldpc import LdpcDeRateMatch
from aerial.phy5g.ldpc import LdpcDecoder
from aerial.phy5g.ldpc import CrcChecker
from aerial.phy5g.pusch import PuschRx
from aerial.util.cuda import get_cuda_stream
from aerial.util.fapi import dmrs_fapi_to_bit_array

# Hide log10(10) warning
_ = np.seterr(divide='ignore', invalid='ignore')
```

#### 4.3.9.1.2 Create the PUSCH pipelines

This is a PUSCH receiver pipeline made up of separately called pyAerial PUSCH receiver components.

```
[3]: # Whether to plot intermediate results within the PUSCH pipeline, such as channel
      ↪ estimates and equalized symbols.
plot_figures = True

num_ues = 1
num_tx_ant = 2           # UE antennas
num_rx_ant = 4          # gNB antennas
cell_id = 41           # Physical cell ID
enable_pusch_tdi = 0    # Enable time interpolation for equalizer coefficients
eq_coeff_algo = 1       # Equalizer algorithm

# The PUSCH receiver chain built from separately called pyAerial Python components is
↪ defined here.
```

(continues on next page)

(continued from previous page)

```

class PuschRxSeparate:
    """PUSCH receiver class.

    This class encapsulates the whole PUSCH receiver chain built using
    pyAerial components.
    """

    def __init__(self,
                 num_rx_ant,
                 enable_pusch_tdi,
                 eq_coeff_algo,
                 plot_figures):
        """Initialize the PUSCH receiver."""
        self.cuda_stream = get_cuda_stream()

        # Build the components of the receiver.
        self.channel_estimator = ChannelEstimator(
            num_rx_ant=num_rx_ant,
            cuda_stream=self.cuda_stream)
        self.channel_equalizer = ChannelEqualizer(
            num_rx_ant=num_rx_ant,
            enable_pusch_tdi=enable_pusch_tdi,
            eq_coeff_algo=eq_coeff_algo,
            cuda_stream=self.cuda_stream)
        self.noise_intf_estimator = NoiseIntfEstimator(
            num_rx_ant=num_rx_ant,
            eq_coeff_algo=eq_coeff_algo,
            cuda_stream=self.cuda_stream)
        self.derate_match = LdpcDeRateMatch(
            enable_scrambling=True,
            cuda_stream=self.cuda_stream)
        self.decoder = LdpcDecoder(cuda_stream=self.cuda_stream)
        self.crc_checker = CrcChecker(cuda_stream=self.cuda_stream)

        # Whether to plot the intermediate results.
        self.plot_figures = plot_figures

    def run(
        self,
        rx_slot,
        num_ues,
        slot,
        num_dmrs_cdm_grps_no_data,
        dmrs_scrm_id,
        start_prb,
        num_prbs,
        dmrs_syms,
        dmrs_max_len,
        dmrs_add_ln_pos,
        start_sym,
        num_symbols,
        scids,
        layers,
        dmrs_ports,
        rntis,
        data_scids,

```

(continues on next page)

(continued from previous page)

```

code_rates,
mod_orders,
tb_sizes
):
    """Run the receiver."""
    # Channel estimation.
    ch_est = self.channel_estimator.estimate(
        rx_slot=rx_slot,
        num_ues=num_ues,
        slot=slot,
        num_dmrs_cdm_grps_no_data=num_dmrs_cdm_grps_no_data,
        dmrs_scrm_id=dmrs_scrm_id,
        start_prb=start_prb,
        num_prbs=num_prbs,
        prg_size=1,
        num_ul_streams=1,
        dmrs_syms=dmrs_syms,
        dmrs_max_len=dmrs_max_len,
        dmrs_add_ln_pos=dmrs_add_ln_pos,
        start_sym=start_sym,
        num_symbols=num_symbols,
        scids=scids,
        layers=layers,
        dmrs_ports=dmrs_ports
    )

    # Noise and interference estimation.
    lw_inv, noise_var_pre_eq = self.noise_intf_estimator.estimate(
        rx_slot=rx_slot,
        channel_est=ch_est,
        num_ues=num_ues,
        slot=slot,
        num_dmrs_cdm_grps_no_data=num_dmrs_cdm_grps_no_data,
        dmrs_scrm_id=dmrs_scrm_id,
        start_prb=start_prb,
        num_prbs=num_prbs,
        dmrs_syms=dmrs_syms,
        dmrs_max_len=dmrs_max_len,
        dmrs_add_ln_pos=dmrs_add_ln_pos,
        start_sym=start_sym,
        num_symbols=num_symbols,
        scids=scids,
        layers=layers,
        dmrs_ports=dmrs_ports
    )

    # Channel equalization and soft demapping. The first return value are the LLRs,
    # second are the equalized symbols. We only want the LLRs now.
    llrs, sym = self.channel_equalizer.equalize(
        rx_slot=rx_slot,
        channel_est=ch_est,
        lw_inv=lw_inv,
        noise_var_pre_eq=noise_var_pre_eq,
        num_ues=num_ues,
        num_dmrs_cdm_grps_no_data=num_dmrs_cdm_grps_no_data,
        start_prb=start_prb,

```

(continues on next page)

(continued from previous page)

```

        num_prbs=num_prbs,
        dmrs_syms=dmrs_syms,
        dmrs_max_len=dmrs_max_len,
        dmrs_add_ln_pos=dmrs_add_ln_pos,
        start_sym=start_sym,
        num_symbols=num_symbols,
        layers=layers,
        mod_orders=mod_orders
    )

    if self.plot_figures:
        fig, axs = plt.subplots(1,4)
        for ant in range(4):
            axs[ant].imshow(10*np.log10(np.abs(rx_slot[:, :, ant]**2)), aspect=
↪ 'auto')
            axs[ant].set_ylim([pusch_record.rbStart * 12, pusch_record.rbSize *
↪ 12])
            axs[ant].set_title('Ant ' + str(ant))
            axs[ant].set(xlabel='Symbol', ylabel='Resource Element')
            axs[ant].label_outer()
            fig.suptitle('Power in RU Antennas')

        fig, axs = plt.subplots(1,2)
        axs[0].scatter(rx_slot.reshape(-1).real, rx_slot.reshape(-1).imag)
        axs[0].set_title("Pre-Equalized samples")
        axs[0].set_aspect('equal')

        axs[1].scatter(np.array(sym).reshape(-1).real, np.array(sym).reshape(-1).
↪ imag)
        axs[1].set_title("Post-Equalized samples")
        axs[1].set_aspect('equal')

        fig, axs = plt.subplots(1)
        axs.set_title("Channel estimates from the PUSCH pipeline")
        for ant in range(4):
            axs.plot(np.abs(ch_est[0][ant, 0, :, 0]))
        axs.legend(["Rx antenna 0, estimate",
                  "Rx antenna 1, estimate",
                  "Rx antenna 2, estimate",
                  "Rx antenna 3, estimate"])
        axs.grid(True)
        plt.show()

        num_data_sym = (np.array(dmrs_syms[start_sym:start_sym + num_symbols]) == 0).
↪ sum()
        cinits = [(rntis[ue] << 15) + data_scids[ue] for ue in range(num_ues)]
        rate_match_lengths = [num_data_sym * mod_orders[ue] * num_prbs * 12 *
↪ layers[ue]
                            for ue in range(num_ues)]
        tb_sizes = [s * 8 for s in tb_sizes]
        code_rates = [c / 1024. for c in code_rates]
        rvs = [0,] * num_ues
        ndis = [1,] * num_ues

```

(continues on next page)

(continued from previous page)

```

        coded_blocks = self.derate_match.derate_match(
            input_llrs=llrs,
            tb_sizes=tb_sizes,
            code_rates=code_rates,
            rate_match_lengths=rate_match_lengths,
            mod_orders=mod_orders,
            num_layers=layers,
            redundancy_versions=rvs,
            ndis=ndis,
            cinits=cinits
        )

        code_blocks = self.decoder.decode(
            input_llrs=coded_blocks,
            tb_sizes=tb_sizes,
            code_rates=code_rates,
            redundancy_versions=rvs,
            rate_match_lengths=rate_match_lengths
        )

        decoded_tbs, tb_crcs = self.crc_checker.check_crc(
            input_bits=code_blocks,
            tb_sizes=tb_sizes,
            code_rates=code_rates
        )

        return decoded_tbs

pusch_rx_separate = PuschRxSeparate(
    num_rx_ant=num_rx_ant,
    enable_pusch_tdi=enable_pusch_tdi,
    eq_coeff_algo=eq_coeff_algo,
    plot_figures=plot_figures
)

# This is the fully fused PUSCH receiver chain.
pusch_rx = PuschRx(
    cell_id=cell_id,
    num_rx_ant=num_rx_ant,
    num_tx_ant=num_rx_ant,
    enable_pusch_tdi=enable_pusch_tdi,
    eq_coeff_algo=eq_coeff_algo
)

```

#### 4.3.9.1.3 Querying the database

Below shows how to connect to the clickhouse database and querying the data from it.

```

[4]: # Connect to the local database
client = clickhouse_connect.get_client(host='localhost')

# Pick a packet from the database
pusch_records = client.query_df('select * from fapi where mcsIndex != 0 order by
↳TsTaiNs limit 10')

```

## 4.3.9.1.4 Extract the PUSCH parameters and run the pipelines

```
[5]: for index, pusch_record in pusch_records.iterrows():
    query = f"""select TsTaiNs, fhData from fh where
            TsTaiNs == {pusch_record.TsTaiNs.timestamp()}
            """
    fh = client.query_df(query)

    display(Markdown("### Example {} - SFN.Slot {}.{} from time {}".format(index + 1, pusch_record.SFN, pusch_record.Slot, pusch_
    record.TsTaiNs
    )))

    # Make sure that the fronthaul database is complete for the SFN.Slot we've chosen
    if fh.index.size < 1:
        pusch_records = pusch_records.drop(index)
        continue;

    fh_samp = np.array(fh['fhData'][0], dtype=np.float32)
    rx_slot = np.swapaxes(fh_samp.view(np.complex64).reshape(4, 14, 273 * 12), 2, 0)

    # Extract all the needed parameters from the PUSCH record.
    slot = int(pusch_record.Slot)
    rntis = [pusch_record.rnti]
    layers = [pusch_record.nrOfLayers]
    start_prb = pusch_record.rbStart
    num_prbs = pusch_record.rbSize
    start_sym = pusch_record.StartSymbolIndex
    num_symbols = pusch_record.NrOfSymbols
    scids = [int(pusch_record.SCID)]
    data_scids = [pusch_record.dataScramblingId]
    dmrs_scrm_id = pusch_record.ulDmrsScramblingId
    num_dmrs_cdm_grps_no_data = pusch_record.numDmrsCdmGrpsNoData
    dmrs_syms = dmrs_fapi_to_bit_array(int(pusch_record.ulDmrsSymbPos))
    dmrs_ports = [pusch_record.dmrsPorts]
    dmrs_max_len = 1
    dmrs_add_ln_pos = 2
    mcs_tables = [pusch_record.mcsTable]
    mcs_indices = [pusch_record.mcsIndex]
    coderates = [pusch_record.targetCodeRate / 10.]
    tb_sizes = [pusch_record.TBSize]
    mod_orders = [pusch_record.qamModOrder]
    tb_input = np.array(pusch_record.pduData)

    # Run the receiver built from separately called components.
    tbs = pusch_rx_separate.run(
        rx_slot=rx_slot,
        num_ues=num_ues,
        slot=slot,
        num_dmrs_cdm_grps_no_data=num_dmrs_cdm_grps_no_data,
        dmrs_scrm_id=dmrs_scrm_id,
        start_prb=start_prb,
        num_prbs=num_prbs,
        dmrs_syms=dmrs_syms,
        dmrs_max_len=dmrs_max_len,
        dmrs_add_ln_pos=dmrs_add_ln_pos,
        start_sym=start_sym,
```

(continues on next page)



(continued from previous page)

```

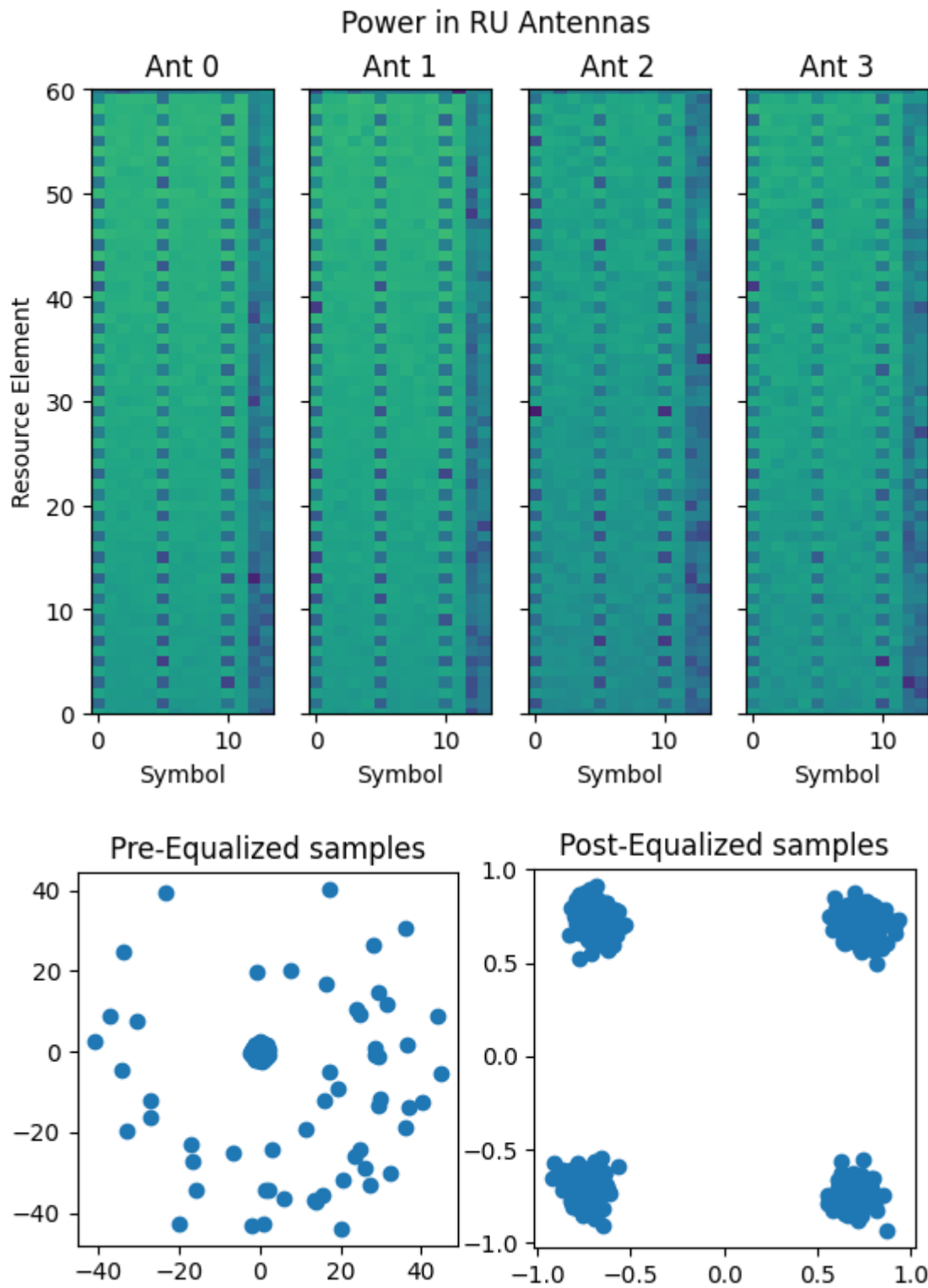
num_symbols=num_symbols,
scids=scids,
layers=layers,
dmrs_ports=dmrs_ports,
rntis=rntis,
data_scids=data_scids,
code_rates=coderates,
mod_orders=mod_orders,
tb_sizes=tb_sizes
)
if np.array_equal(tbs[0][:tb_input.size], tb_input):
    display(Markdown("**Separated kernels PUSCH decoding success** for SFN.Slot {}.{} from
→ {} from time {}".format(pusch_record.SFN, pusch_record.Slot, pusch_record.TsTaiNs)))
else:
    display(Markdown("**Separated kernels PUSCH decoding failure**"))
    print("Output bytes:")
    print(tbs[0][:tb_input.size])
    print("Expected output:")
    print(tb_input)

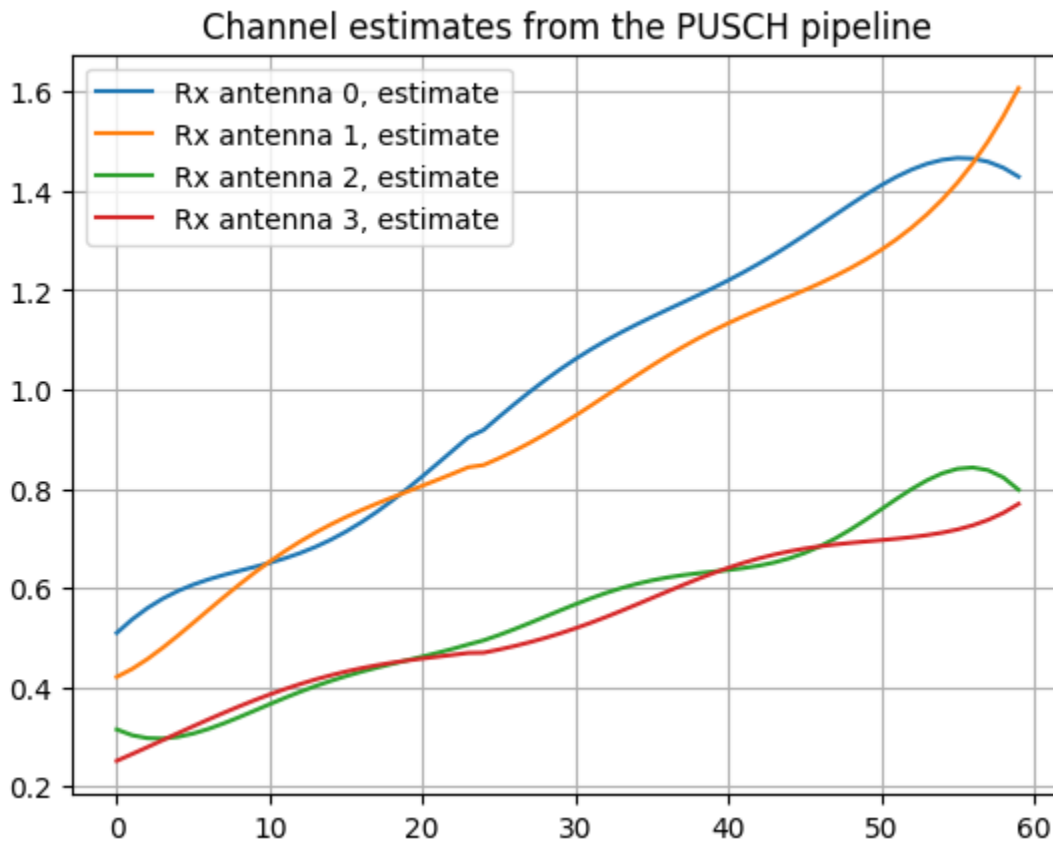
# Run the fused PUSCH receiver.
# Note that this is where we set the dynamically changing parameters.
tb_crcs, tbs = pusch_rx.run(
    rx_slot=rx_slot,
    num_ues=num_ues,
    slot=slot,
    num_dmrs_cdm_grps_no_data=num_dmrs_cdm_grps_no_data,
    dmrs_scrm_id=dmrs_scrm_id,
    start_prb=start_prb,
    num_prbs=num_prbs,
    dmrs_syms=dmrs_syms,
    dmrs_max_len=dmrs_max_len,
    dmrs_add_ln_pos=dmrs_add_ln_pos,
    start_sym=start_sym,
    num_symbols=num_symbols,
    scids=scids,
    layers=layers,
    dmrs_ports=dmrs_ports,
    rntis=rntis,
    data_scids=data_scids,
    code_rates=coderates,
    mod_orders=mod_orders,
    tb_sizes=tb_sizes
)

if np.array_equal(tbs[0][:tb_input.size], tb_input):
    display(Markdown("**Fused PUSCH decoding success** for SFN.Slot {}.{} from
→ time {}".format(pusch_record.SFN, pusch_record.Slot, pusch_record.TsTaiNs)))
else:
    display(Markdown("**Fused PUSCH decoding failure**"))
    print("Output bytes:")
    print(tbs[0][:tb_input.size])
    print("Expected output:")
    print(tb_input)

```

#### 4.3.9.1.4.1 Example 1 - SFN.Slot 192.4 from time 2024-03-21 12:18:39.162000

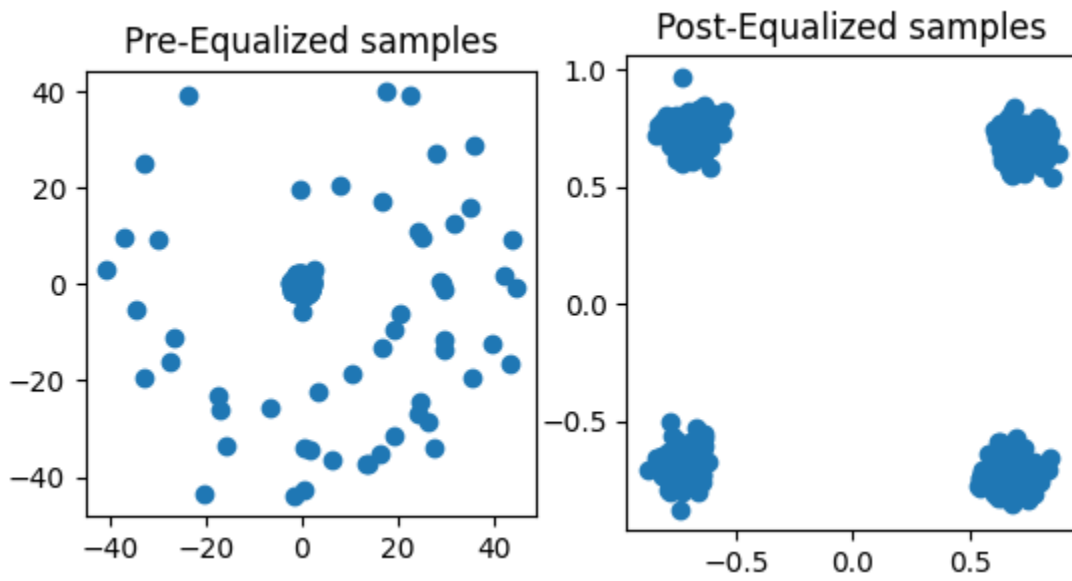
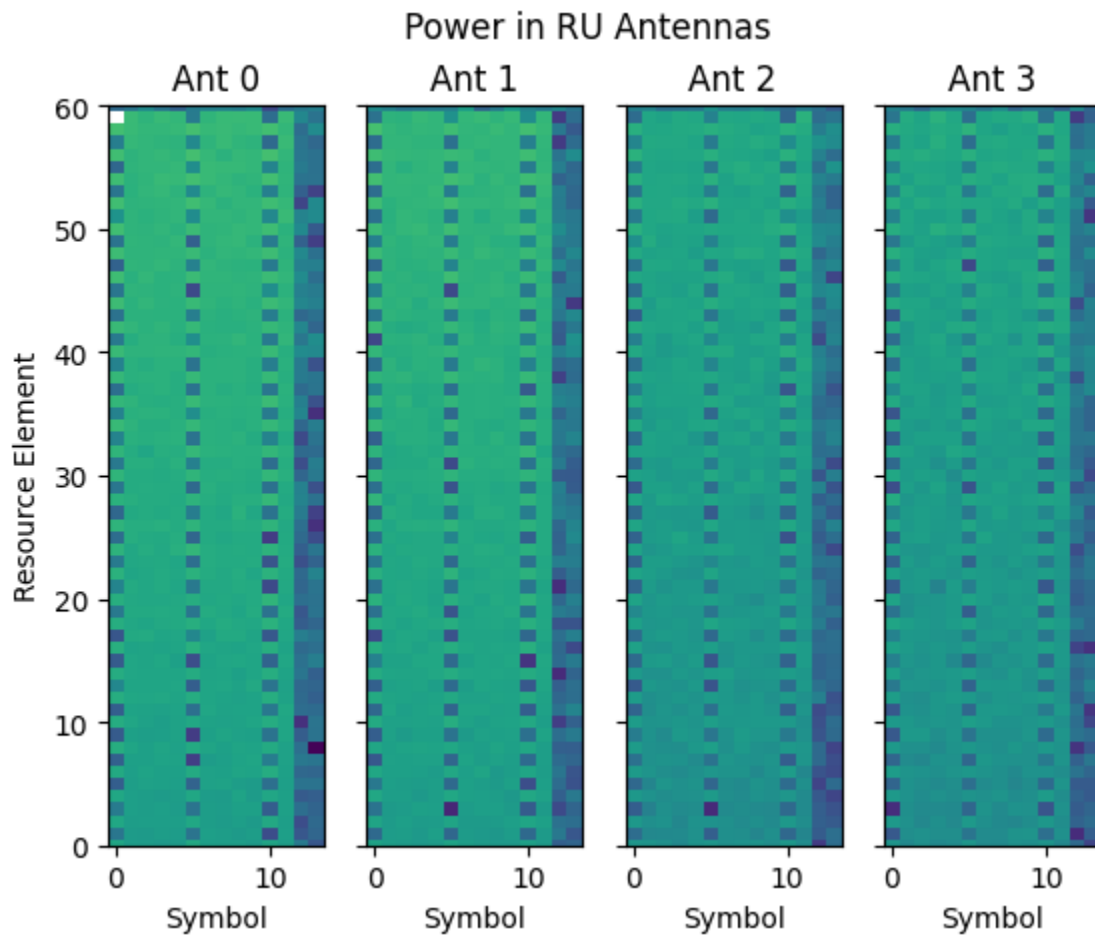


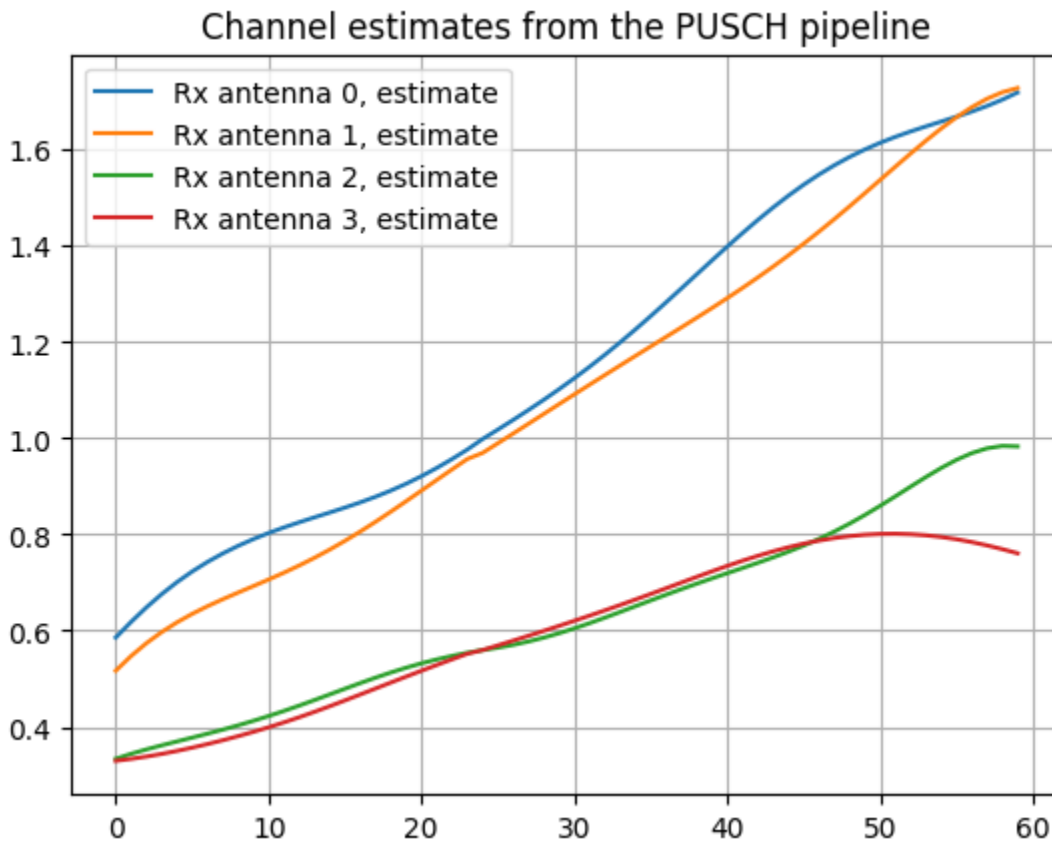


**Separated kernels PUSCH decoding success** for SFN.Slot 192.4 from time 2024-03-21 12:18:39.162000

**Fused PUSCH decoding success** for SFN.Slot 192.4 from time 2024-03-21 12:18:39.162000

**4.3.9.1.4.2 Example 2 - SFN.Slot 194.14 from time 2024-03-21 12:18:39.187000**

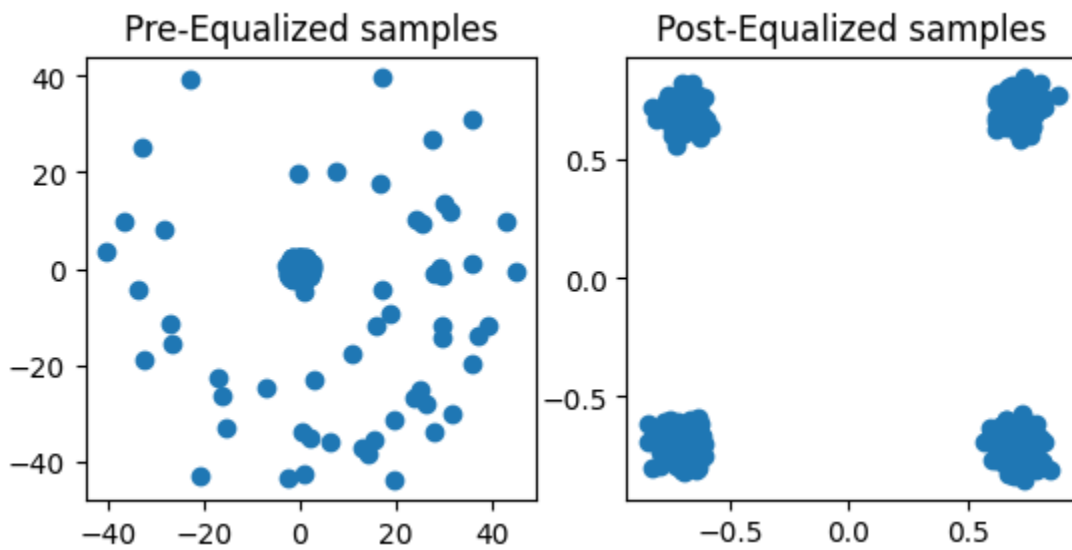
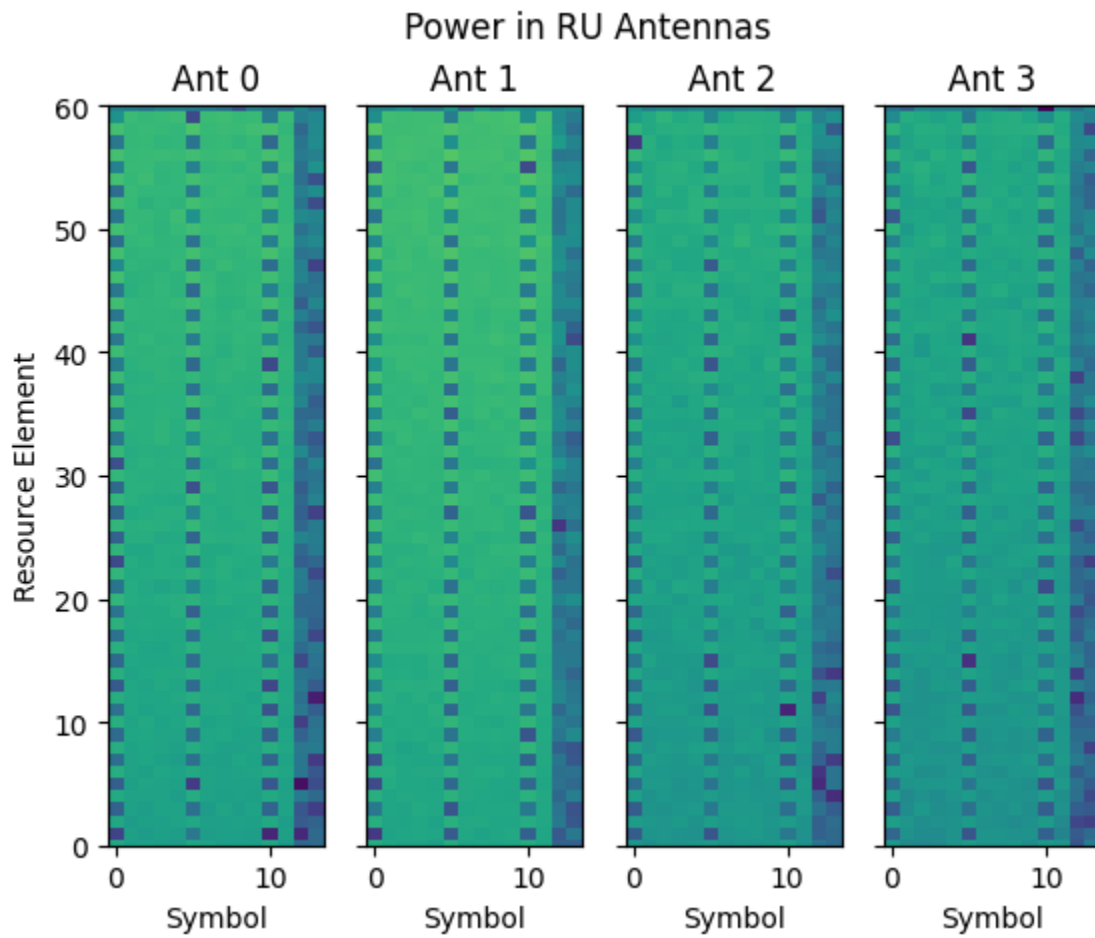


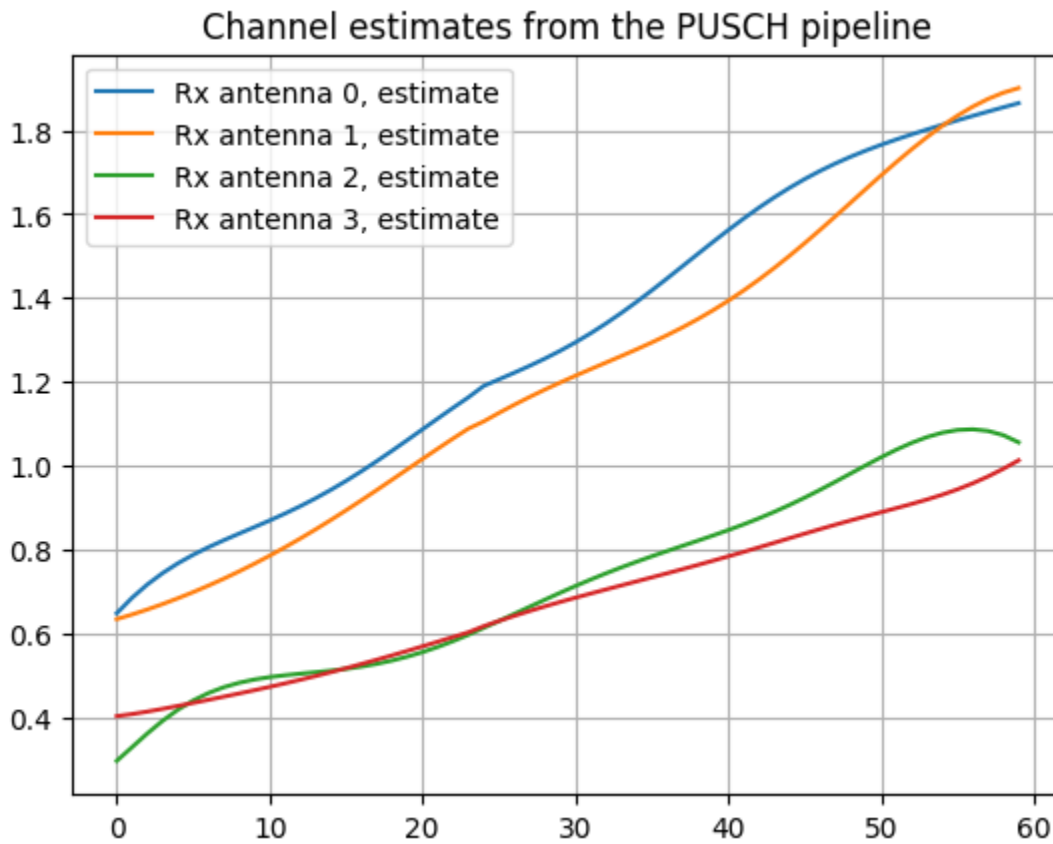


**Separated kernels PUSCH decoding success** for SFN.Slot 194.14 from time 2024-03-21 12:18:39.187000

**Fused PUSCH decoding success** for SFN.Slot 194.14 from time 2024-03-21 12:18:39.187000

**4.3.9.1.4.3 Example 3 - SFN.Slot 195.4 from time 2024-03-21 12:18:39.192000**

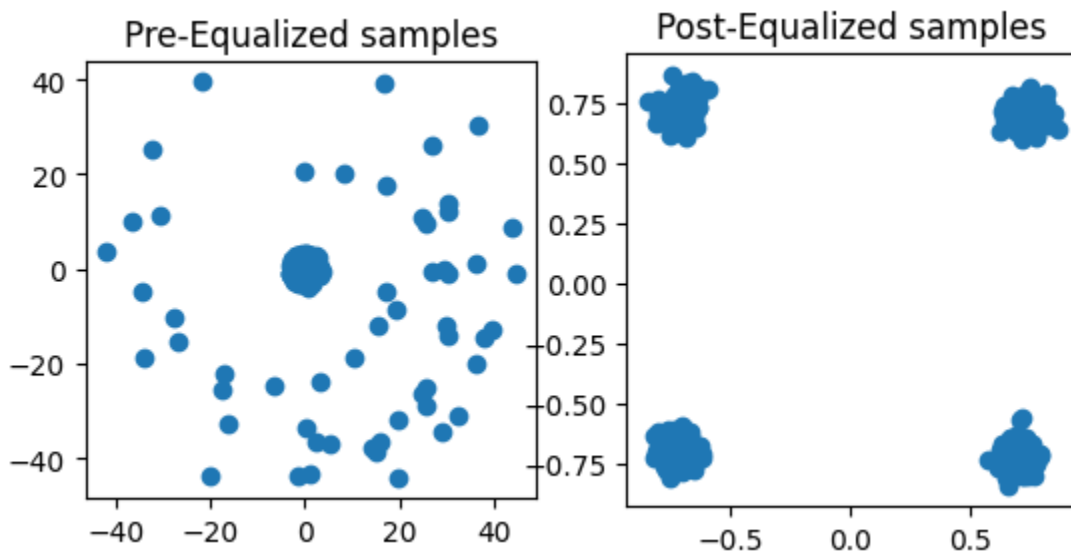
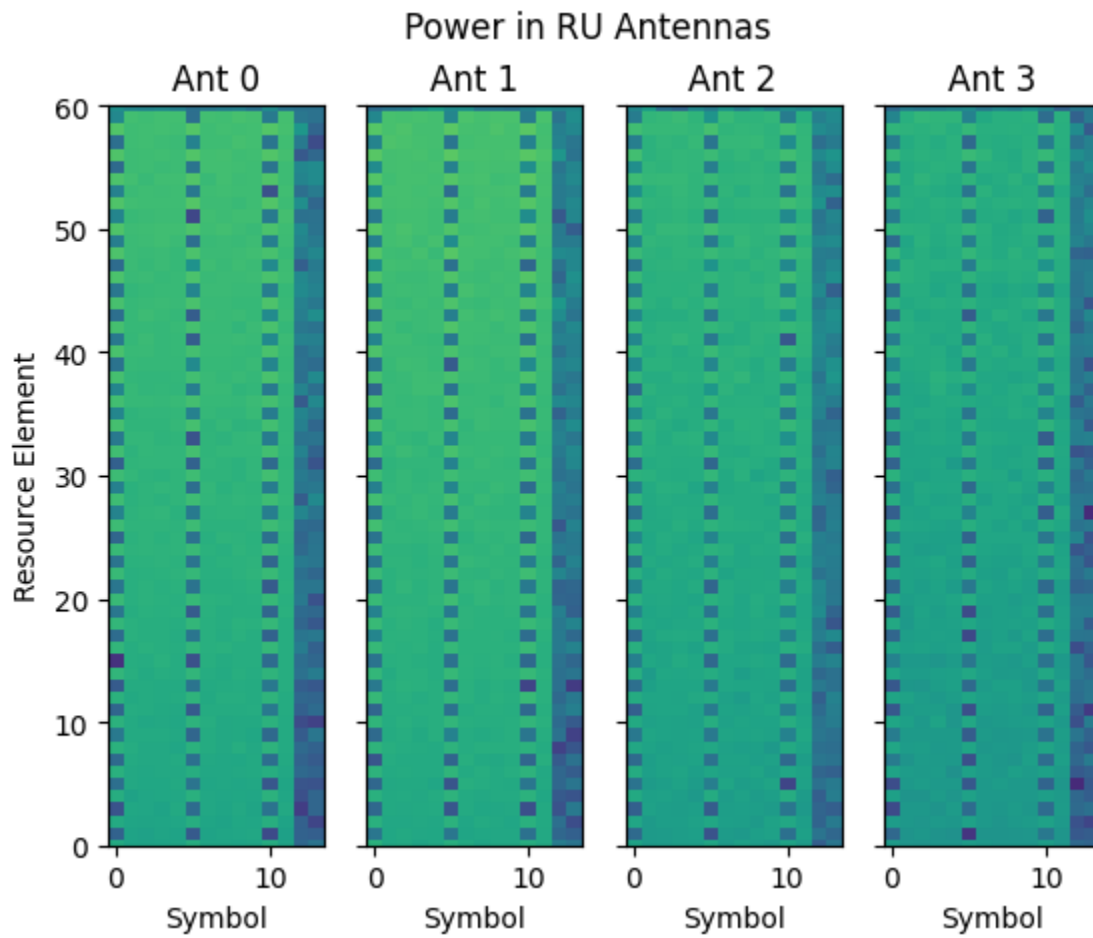




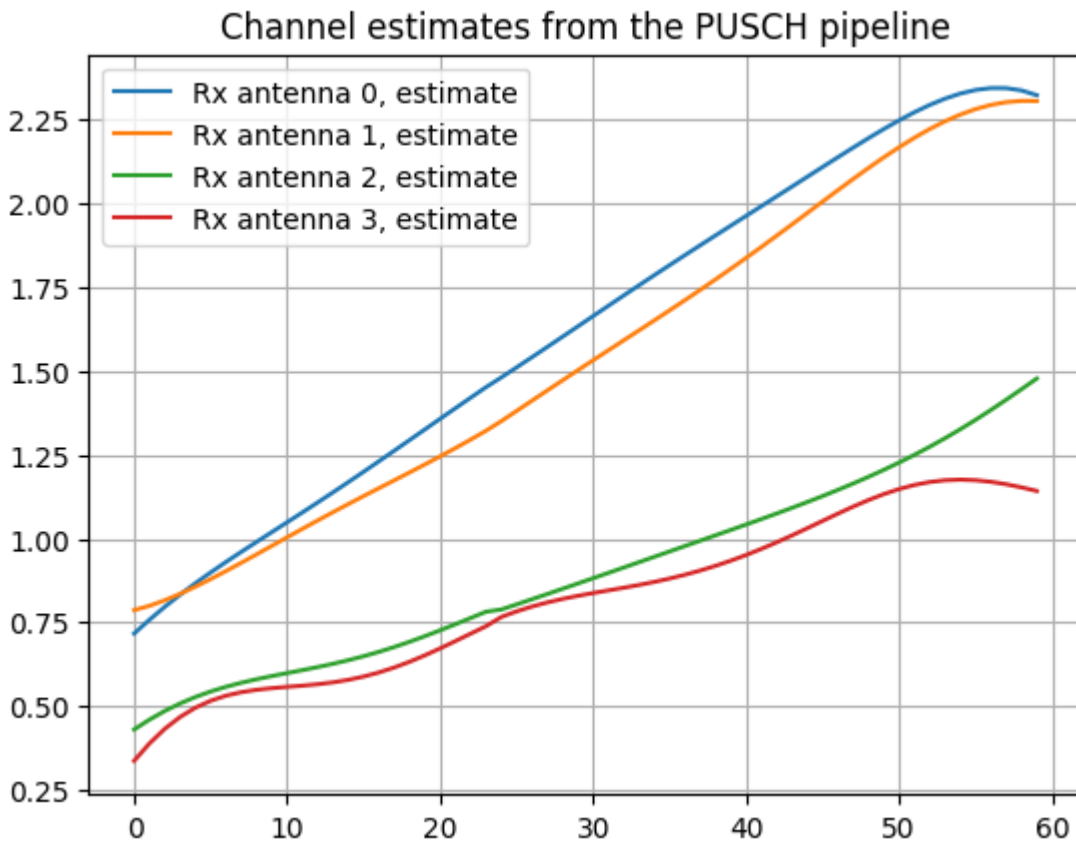
**Separated kernels PUSCH decoding success** for SFN.Slot 195.4 from time 2024-03-21 12:18:39.192000

**Fused PUSCH decoding success** for SFN.Slot 195.4 from time 2024-03-21 12:18:39.192000

**4.3.9.1.4.4 Example 4 - SFN.Slot 201.4 from time 2024-03-21 12:18:39.252000**



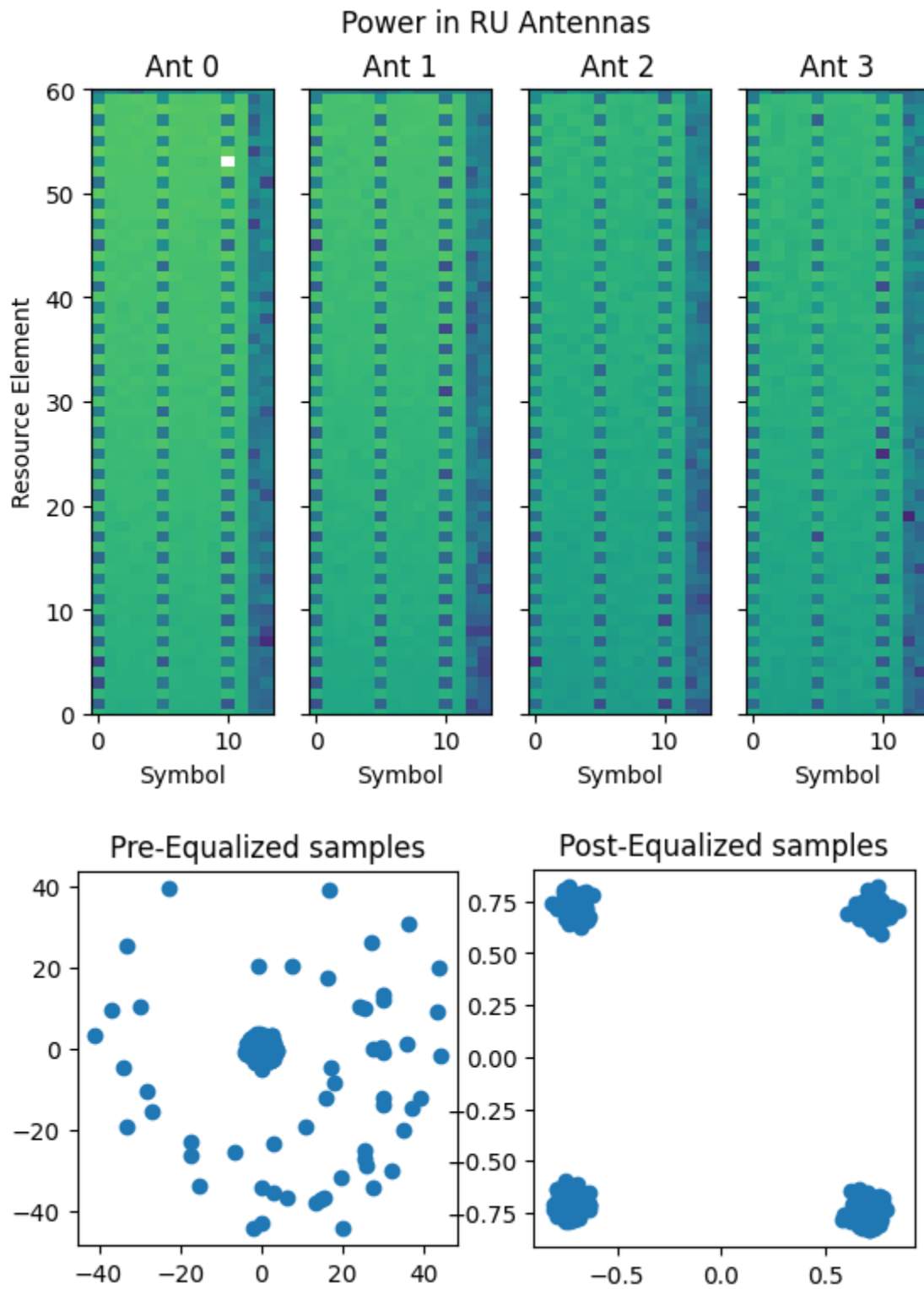


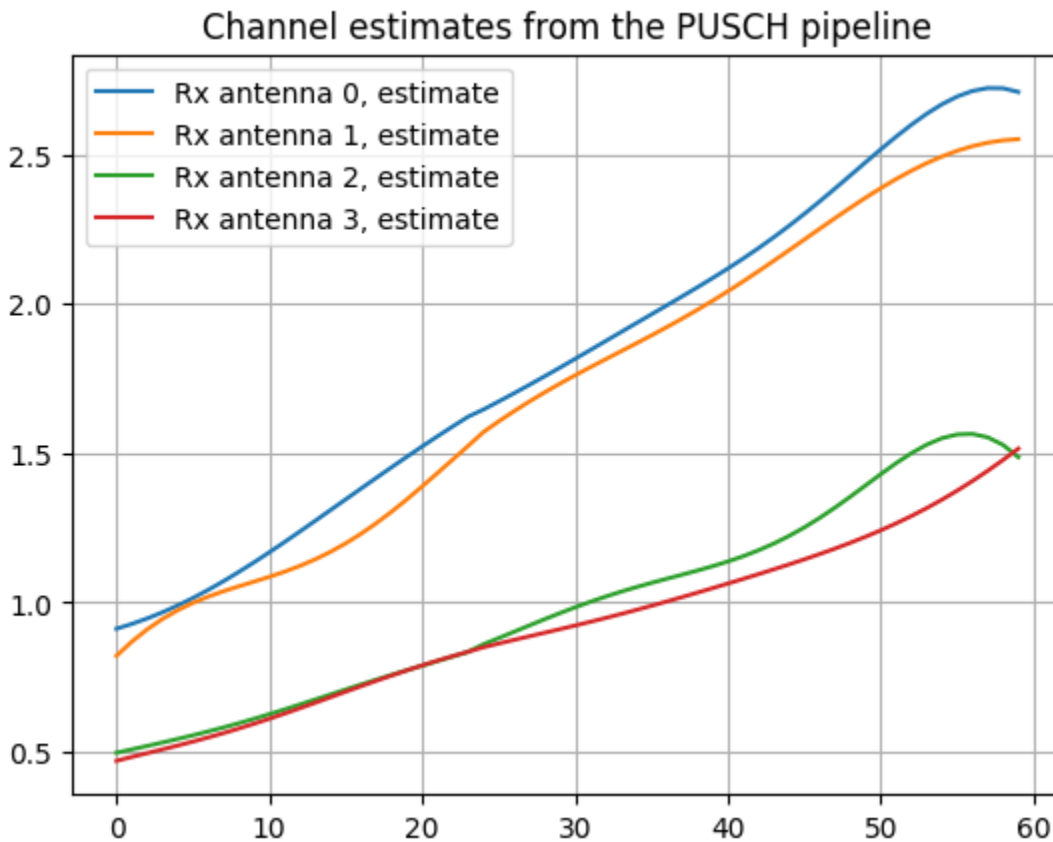


**Separated kernels PUSCH decoding success** for SFN.Slot 201.4 from time 2024-03-21 12:18:39.252000

**Fused PUSCH decoding success** for SFN.Slot 201.4 from time 2024-03-21 12:18:39.252000

**4.3.9.1.4.5 Example 5 - SFN.Slot 209.4 from time 2024-03-21 12:18:39.332000**





**Separated kernels PUSCH decoding success** for SFN.Slot 209.4 from time 2024-03-21 12:18:39.332000

**Fused PUSCH decoding success** for SFN.Slot 209.4 from time 2024-03-21 12:18:39.332000

Similarly to the previous example, this example illustrates the use of pyAerial in context of Aerial Data Lake. In this example, the PUSCH data queried from the database is run through a full PUSCH receiver pipeline implemented using the pyAerial API. The example also illustrates how the pyAerial PUSCH components enable fetching intermediate results from the receiver pipeline.

#### Note

Similarly to the previous notebook, this notebook requires that the clickhouse server used by Aerial Data Lake is running, and that the example data has been imported into a database. Refer to the Aerial Data Lake installation docs on how to do this.

For more information, refer to the [Aerial Data Lake section](#).

## 4.4. API Reference

### 4.4.1. Physical layer for 5G

This module contains classes implementing the 5G NR physical layer using GPU acceleration through the cuPHY library. The module contains full PDSCH transmitter and PUSCH receiver pipelines in *PdschTx* and *PuschRx*, respectively. The other parts of this module contain individual components of the transmitter-receiver chain, such as for example the LDPC encoder and decoder in *LdpcEncoder* and *LdpcDecoder*, and the channel estimator in *ChannelEstimator*.

#### 4.4.1.1 Receiver algorithms

This module contains a number of receiver algorithms implemented in cuPHY, thus using GPU acceleration.

**class** `aerial.phy5g.algorithms.channel_estimator.ChannelEstimator`

Channel estimator class.

This class implements traditional MMSE-based channel estimation on the DMRS symbols of the received slot signal. It outputs the channel estimates for all resource elements in the DMRS symbols. Similarly to many other classes in pyAerial, this class handles groups of UEs sharing the same time-frequency resources with one call, i.e. it supports MU-MIMO.

```
__init__(
    num_rx_ant,
    ch_est_algo=1,
    enable_per_prg_chest=0,
    enable_ul_rx_bf=0,
    cuda_stream=None,
    chest_filter_h5=None,
    w_freq_array=None,
    w_freq4_array=None,
    w_freq_small_array=None,
    shift_seq_array=None,
    unshift_seq_array=None,
    shift_seq4_array=None,
    unshift_seq4_array=None,
)
```

Initialize ChannelEstimator.

The channel estimation filters can be given as an H5 file or directly as Numpy arrays. If neither is given, the channel estimator is using default filters.

#### Parameters

- ▶ **num\_rx\_ant** (*int*) – Number of receive antennas.
- ▶ **ch\_est\_algo** (*int*) – Channel estimation algorithm.
- ▶ **MMSE** (- 0 -)
- ▶ **estimation** (- 1 - *Multi-stage MMSE with delay*)
- ▶ **yet** (- 2 - *RKHS not supported by pyAerial*)

- ▶ **only** (- 3 - LS channel estimation)
- ▶ **enable\_per\_prg\_chest** (*int*) – Enable/disable PUSCH per-PRG channel estimation.
- ▶ **0** (-) – Disable (default).
- ▶ **1** (-) – Enable.
- ▶ **enable\_ul\_rx\_bf** (*int*) – Enable/disable beamforming for PUSCH.
- ▶ **0** – Disable (default).
- ▶ **1** – Enable.
- ▶ **cuda\_stream** (*int*) – The CUDA stream. If not given, one will be created.
- ▶ **chest\_filter\_h5** (*str*) – Filename of an HDF5 file containing channel estimation filters.
- ▶ **w\_freq\_array** (*np.ndarray*)
- ▶ **w\_freq4\_array** (*np.ndarray*)
- ▶ **w\_freq\_small\_array** (*np.ndarray*)
- ▶ **shift\_seq\_array** (*np.ndarray*)
- ▶ **unshift\_seq\_array** (*np.ndarray*)
- ▶ **shift\_seq4\_array** (*np.ndarray*)
- ▶ **unshift\_seq4\_array** (*np.ndarray*)

**Return type**

None

```

estimate(
    *,
    rx_slot,
    slot,
    pusch_configs=None,
    num_ues=None,
    num_dmrs_cdm_grps_no_data=None,
    dmrs_scrm_id=None,
    start_prb=None,
    num_prbs=None,
    prg_size=None,
    num_ul_streams=None,
    dmrs_syms=None,
    dmrs_max_len=None,
    dmrs_add_ln_pos=None,
    start_sym=None,
    num_symbols=None,
    scids=None,
    layers=None,
    dmrs_ports=None,
)

```

Run channel estimation for multiple UE groups.

This runs the cuPHY channel estimation for all UE groups included in *pusch\_configs*. If this argument is not given, all the other arguments need to be given and cuPHY channel estimation is run only for a single UE group sharing the same time-frequency resources, i.e. having

the same PRB allocation, and the same start symbol and number of allocated symbols. This single UE group is the parameterized by the all other arguments.

The method can be called using either Numpy or CuPy arrays. In case the input arrays are located on the GPU (CuPy), the output will be on the GPU (CuPy). So the return type shall be the same as used for `rx_slot` when calling the method.

### Parameters

- ▶ **rx\_slot** (*Array*) – Input received data as a frequency x time x Rx antenna Numpy or CuPy array with type `complex64` entries.
- ▶ **slot** (*int*) – Slot number.
- ▶ **pusch\_configs** (*List[PuschConfig]*) – List of PUSCH configuration objects, one per UE group. If this argument is given, the rest are ignored. If not given, all other arguments need to be given (only one UE group supported in that case).
- ▶ **num\_ues** (*int*) – Number of UEs in the single UE group.
- ▶ **num\_dmrs\_cdm\_grps\_no\_data** (*int*) – Number of DMRS CDM groups without data [3GPP TS 38.212, sec 7.3.1.1]. Value: 1->3.
- ▶ **dmrs\_scrm\_id** (*int*) – DMRS scrambling ID.
- ▶ **start\_prb** (*int*) – Start PRB index of the UE allocation.
- ▶ **num\_prbs** (*int*) – Number of allocated PRBs for the UE group.
- ▶ **prg\_size** (*int*) – The Size of PRG in PRB for PUSCH per-PRG channel estimation.
- ▶ **num\_ul\_streams** (*int*) – The number of active streams for this PUSCH.
- ▶ **dmrs\_syms** (*List[int]*) – For the UE group, a list of binary numbers each indicating whether the corresponding symbol is a DMRS symbol. The length of the list equals the number of symbols in the slot. 0 means no DMRS in the symbol and 1 means the symbol is a DMRS symbol.
- ▶ **dmrs\_max\_len** (*int*) – The `maxLength` parameter, value 1 or 2, meaning that DMRS are single-symbol DMRS or single- or double-symbol DMRS.
- ▶ **dmrs\_add\_ln\_pos** (*int*) – Number of additional DMRS positions.
- ▶ **start\_sym** (*int*) – Start symbol index for the UE group allocation.
- ▶ **num\_symbols** (*int*) – Number of symbols in the UE group allocation.
- ▶ **scids** (*List[int]*) – DMRS sequence initialization SCID [TS38.211, sec 7.4.1.1.2] for each UE in the UE group. Value is 0 or 1.
- ▶ **layers** (*List[int]*) – Number of layers for each UE in the UE group. The length of the list equals the number of UEs.
- ▶ **dmrs\_ports** (*List[int]*) – DMRS ports for each UE in the UE group. The format of each entry is in the SCF FAPI format as follows: A bitmap (mask) starting from the LSB where each bit indicates whether the corresponding DMRS port index is used.

### Returns

The channel estimates as a Rx ant x layer x frequency x time Numpy or CuPy array, per UE group.

**Return type**

List[Array]

**class** aerial.phy5g.algorithms.noise\_intf\_estimator.NoiseIntfEstimator

Noise and interference estimator class.

This class implements an algorithm for noise and interference estimation. It calls the corresponding cuPHY algorithms and provides the estimates as needed for cuPHY equalization and soft demapping.

It needs channel estimates as its input, along with the received data symbols.

```
__init__(
    num_rx_ant,
    eq_coeff_algo,
    cuda_stream=None,
)
```

Initialize NoiseIntfEstimator.

**Parameters**

- ▶ **num\_rx\_ant** (*int*) – Number of receive antennas.
- ▶ **eq\_coeff\_algo** (*int*) – Algorithm used to compute equalizer coefficients.
  - ▶ 0: Zero-forcing equalizer.
  - ▶ 1: MMSE with noise variance only.
  - ▶ 2: MMSE-IRC.
- ▶ **cuda\_stream** (*int*) – The CUDA stream. If not given, one will be created.

**Return type**

None

```
estimate(
    *,
    rx_slot,
    channel_est,
    slot,
    pusch_configs=None,
    num_ues=None,
    num_dmrs_cdm_grps_no_data=None,
    dmrs_scrm_id=None,
    start_prb=None,
    num_prbs=None,
    dmrs_syms=None,
    dmrs_max_len=None,
    dmrs_add_ln_pos=None,
    start_sym=None,
    num_symbols=None,
    scids=None,
    layers=None,
    dmrs_ports=None,
)
```

Estimate noise and interference.

This runs the cuPHY noise and interference estimation for all UE groups included in *pusch\_configs*. If this argument is not given, all the other arguments need to be given and cuPHY noise and interference estimation is run only for a single UE group sharing the same time-frequency resources, i.e. having the same PRB allocation, and the same start symbol and number of allocated symbols.

The method can be called using either Numpy or CuPy arrays. In case the input arrays are located on the GPU (CuPy), the output will be on the GPU (CuPy). So the return type shall be the same as used for *rx\_slot* when calling the method.

### Parameters

- ▶ **rx\_slot** (*Array*) – Input received data as a frequency x time x Rx antenna Numpy or CuPy array with type *complex64* entries.
- ▶ **channel\_est** (*List[Array]*) – The channel estimates as a Rx ant x layer x frequency x time Numpy or CuPy array, per UE group.
- ▶ **slot** (*int*) – Slot number.
- ▶ **pusch\_configs** (*List[PuschConfig]*) – List of PUSCH configuration objects, one per UE group. If this argument is given, the rest are ignored. If not given, all other arguments need to be given (only one UE group supported in that case).
- ▶ **num\_ues** (*int*) – Number of UEs in the UE group.
- ▶ **num\_dmrs\_cdm\_grps\_no\_data** (*int*) – Number of DMRS CDM groups without data [3GPP TS 38.212, sec 7.3.1.1]. Value: 1->3.
- ▶ **dmrs\_scrm\_id** (*int*) – DMRS scrambling ID.
- ▶ **start\_prb** (*int*) – Start PRB index of the UE allocation.
- ▶ **num\_prbs** (*int*) – Number of allocated PRBs for the UE group.
- ▶ **dmrs\_syms** (*List[int]*) – For the UE group, a list of binary numbers each indicating whether the corresponding symbol is a DMRS symbol. The length of the list equals the number of symbols in the slot. 0 means no DMRS in the symbol and 1 means the symbol is a DMRS symbol.
- ▶ **dmrs\_max\_len** (*int*) – The *maxLength* parameter, value 1 or 2, meaning that DMRS are single-symbol DMRS or single- or double-symbol DMRS.
- ▶ **dmrs\_add\_ln\_pos** (*int*) – Number of additional DMRS positions.
- ▶ **start\_sym** (*int*) – Start symbol index for the UE group allocation.
- ▶ **num\_symbols** (*int*) – Number of symbols in the UE group allocation.
- ▶ **scids** (*List[int]*) – DMRS sequence initialization SCID [TS38.211, sec 7.4.1.1.2] for each UE. Value is 0 or 1.
- ▶ **layers** (*List[int]*) – Number of layers for each UE. The length of the list equals the number of UEs.
- ▶ **dmrs\_ports** (*List[int]*) – DMRS ports for each UE. The format of each entry is in the SCF FAPI format as follows: A bitmap (mask) starting from the LSB where each bit indicates whether the corresponding DMRS port index is used.

### Returns



A tuple containing:

- ▶ *List[Array]*: Inverse of the Cholesky decomposition of the noise/interference covariance matrix per PRB, per UE group. The size of each entry in this list is number of Rx antennas x number of Rx antennas x number of PRBs.
- ▶ *Array*: Pre-equalization wideband noise variance estimate per UE, i.e. one value per UE averaged over the whole frequency allocation. This value is in dB.

**Return type**

List[Array], Array

**class** `aerial.phy5g.algorithms.channel_equalizer.ChannelEqualizer`

Channel equalizer class.

This class implements MMSE-based channel equalization along with soft demapping to get log-likelihood ratios for channel decoding.

It needs channel estimates and noise and interference estimates as its input, along with the received data symbols.

```
__init__(
    num_rx_ant,
    eq_coeff_algo,
    enable_pusch_tdi,
    cuda_stream=None,
)
```

Initialize ChannelEqualizer.

**Parameters**

- ▶ **num\_rx\_ant** (*int*) – Number of receive antennas.
- ▶ **eq\_coeff\_algo** (*int*) – Algorithm used to compute equalizer coefficients.
  - ▶ 0: Zero-forcing equalizer.
  - ▶ 1: MMSE with noise variance only.
  - ▶ 2: MMSE-IRC.
- ▶ **enable\_pusch\_tdi** (*int*) – Whether to use time-domain interpolation.
- ▶ **cuda\_stream** (*int*) – The CUDA stream. If not given, one will be created.

**Return type**

None

```
equalize(
    *,
    rx_slot,
    channel_est,
    lw_inv,
    noise_var_pre_eq,
    pusch_configs=None,
    num_ues=None,
    num_dmrs_cdm_grps_no_data=None,
    start_prb=None,
    num_prbs=None,
    dmrs_syms=None,
    dmrs_max_len=None,
```

```

dmrs_add_in_pos=None,
start_sym=None,
num_symbols=None,
layers=None,
mod_orders=None,
)

```

Run equalization and soft demapping.

This runs the cuPHY equalization for all UE groups included in *pusch\_configs*. If this argument is not given, all the other arguments need to be given and cuPHY equalization is run only for a single UE group sharing the same time-frequency resources, i.e. having the same PRB allocation, and the same start symbol and number of allocated symbols.

The method can be called using either Numpy or CuPy arrays. In case the input arrays are located on the GPU (CuPy), the output will be on the GPU (CuPy). So the return type shall be the same as used for *rx\_slot* when calling the method.

### Parameters

- ▶ **rx\_slot** (*Array*) – Input received data as a frequency x time x Rx antenna Numpy or CuPy array with type *np.complex64* entries.
- ▶ **channel\_est** (*List[Array]*) – The channel estimates as a Rx ant x layer x frequency x time Numpy or Cupy array, per UE group.
- ▶ **lw\_inv** (*List[Array]*) – Inverse of the Cholesky decomposition of the noise/interference covariance matrix per PRB, per UE group. The size of each entry in this list is number of Rx antennas x number of Rx antennas x number of PRBs.
- ▶ **noise\_var\_pre\_eq** (*Array*) – Average pre-equalizer noise variance in dB. One value per UE group.
- ▶ **pusch\_configs** (*List[PuschConfig]*) – List of PUSCH configuration objects, one per UE group. If this argument is given, the rest are ignored. If not given, all other arguments need to be given (only one UE group supported in that case).
- ▶ **num\_ues** (*int*) – Number of UEs in the UE group.
- ▶ **num\_dmrs\_cdm\_grps\_no\_data** (*int*) – Number of DMRS CDM groups without data [3GPP TS 38.212, sec 7.3.1.1]. Value: 1->3.
- ▶ **start\_prb** (*int*) – Start PRB index of the UE allocation.
- ▶ **num\_prbs** (*int*) – Number of allocated PRBs for the UE group.
- ▶ **dmrs\_syms** (*List[int]*) – For the UE group, a list of binary numbers each indicating whether the corresponding symbol is a DMRS symbol. The length of the list equals the number of symbols in the slot. 0 means no DMRS in the symbol and 1 means the symbol is a DMRS symbol.
- ▶ **dmrs\_max\_len** (*int*) – The *maxLength* parameter, value 1 or 2, meaning that DMRS are single-symbol DMRS or single- or double-symbol DMRS.
- ▶ **dmrs\_add\_in\_pos** (*int*) – Number of additional DMRS positions.
- ▶ **start\_sym** (*int*) – Start symbol index for the UE group allocation.
- ▶ **num\_symbols** (*int*) – Number of symbols in the UE group allocation.
- ▶ **layers** (*List[int]*) – Number of layers for each UE.

- ▶ **mod\_orders** (*List[int]*) – QAM modulation order for each UE.

### Returns

A tuple containing:

- ▶ *List[Array]*: Log-likelihood ratios for the received bits to be fed into decoding (rate matching). One Numpy array per UE group and the size of each Numpy array is 8 x number of layers x number of subcarriers x number of data symbols. The size of the first dimension is fixed to eight as modulations up to 256QAM are supported and cuPHY returns the same size independently of modulation. Only the first entries corresponding to the actual number of bits are used.
- ▶ *List[Array]*: Equalized symbols, one Numpy array per UE group. The size of each Numpy array is equal to number of layers x number of subcarriers x number of data symbols.

### Return type

List[Array], List[Array]

## class aerial.phy5g.algorithms.rsrp\_estimator.RsrpEstimator

RSRP, post- and pre-equalizer SINR estimator class.

This class implements RSRP estimation as well as post- and pre-equalizer SINR estimation for PUSCH receiver pipeline.

**\_\_init\_\_** (*num\_rx\_ant, enable\_pusch\_tdi, cuda\_stream=None*)

Initialize RsrpEstimator.

### Parameters

- ▶ **num\_rx\_ant** (*int*) – Number of receive antennas.
- ▶ **enable\_pusch\_tdi** (*int*) – Whether time-interpolation is used in computing equalizer coefficients. This impacts post-equalizer SINR.
- ▶ **cuda\_stream** (*int*) – The CUDA stream. If not given, one will be created.

### Return type

None

```
estimate(
    channel_est,
    ree_diag_inv,
    noise_var_pre_eq,
    pusch_configs=None,
    num_ues=None,
    num_prbs=None,
    dmrs_add_in_pos=None,
    layers=None,
)
```

Run RSRP and post- and pre-equalizer SINR estimation.

The method can be called using either Numpy or CuPy arrays. In case the input arrays are located on the GPU (CuPy), the output will be on the GPU (CuPy). So the return type shall be the same as used for *rx\_slot* when calling the method.

### Parameters

- ▶ **channel\_est** (*List[Array]*) – The channel estimates as a Rx ant x layer x frequency x time Numpy or CuPy array, per UE group.
- ▶ **ree\_diag\_inv** (*List[Array]*) – Inverse of post-equalizer residual error covariance diagonal, per UE group.
- ▶ **noise\_var\_pre\_eq** (*Array*) – Average pre-equalizer noise variance in dB. One value per UE group.
- ▶ **pusch\_configs** (*List[PuschConfig]*) – List of PUSCH configuration objects, one per UE group. If this argument is given, the rest are ignored. If not given, all other arguments need to be given (only one UE group supported in that case).
- ▶ **num\_ues** (*int*) – Number of UEs in the UE group.
- ▶ **num\_prbs** (*int*) – Number of allocated PRBs for the UE group.
- ▶ **dmrs\_add\_ln\_pos** (*int*) – Number of additional DMRS positions. This is used to derive the total number of DMRS symbols.
- ▶ **layers** (*List[int]*) – Number of layers for each UE.

### Returns

A tuple containing:

- ▶ *Array*: RSRP values per UE.
- ▶ *Array*: Pre-equalization SINR values per UE.
- ▶ *Array*: Post-equalization SINR values per UE.

### Return type

*Array, Array, Array*

## **class** aerial.phy5g.algorithms.cfo\_ta\_estimator.CfoTaEstimator

CFO and TA estimator class.

This class implements an algorithm for carrier frequency offset and timing advance estimation. It calls the corresponding cuPHY algorithms and provides the estimates as needed for other cuPHY algorithms.

It needs channel estimates as its input.

```
__init__(
    num_rx_ant,
    mu=1,
    enable_cfo_correction=True,
    enable_to_estimation=True,
    cuda_stream=None,
)
```

Initialize CfoTaEstimator.

### Parameters

- ▶ **num\_rx\_ant** (*int*) – Number of receive antennas.
- ▶ **mu** (*int*) – Numerology. Values in [0, 3]. Default: 1.
- ▶ **enable\_cfo\_correction** (*int*) – Enable/disable CFO correction:

- ▶ 0: Disable.
- ▶ 1: Enable (default).
- ▶ **enable\_to\_estimation** (*int*) – Enable/disable time offset estimation:
  - ▶ 0: Disable.
  - ▶ 1: Enable (default).
- ▶ **cuda\_stream** (*int*) – The CUDA stream. If not given, one will be created.

**Return type**

None

```

estimate(
    channel_est,
    pusch_configs=None,
    num_ues=None,
    num_prbs=None,
    dmrs_syms=None,
    dmrs_max_len=None,
    dmrs_add_in_pos=None,
    layers=None,
)

```

Estimate carrier frequency offset and timing advance.

**Parameters**

- ▶ **channel\_est** (*List[Array]*) – The channel estimates as a Rx ant x layer x frequency x time Numpy or CuPy array, per UE group.
- ▶ **pusch\_configs** (*List[PuschConfig]*) – List of PUSCH configuration objects, one per UE group. If this argument is given, the rest are ignored. If not given, all other arguments need to be given (only one UE group supported in that case).
- ▶ **num\_ues** (*int*) – Number of UEs in the UE group.
- ▶ **num\_prbs** (*int*) – Number of allocated PRBs for the UE group.
- ▶ **dmrs\_syms** (*List[int]*) – For the UE group, a list of binary numbers each indicating whether the corresponding symbol is a DMRS symbol. The length of the list equals the number of symbols in the slot. 0 means no DMRS in the symbol and 1 means the symbol is a DMRS symbol.
- ▶ **dmrs\_max\_len** (*int*) – The *maxLength* parameter, value 1 or 2, meaning that DMRS are single-symbol DMRS or single- or double-symbol DMRS.
- ▶ **dmrs\_add\_in\_pos** (*int*) – Number of additional DMRS positions.
- ▶ **layers** (*List[int]*) – Number of layers for each UE. The length of the list equals the number of UEs.

**Returns**

A tuple containing:

- ▶ *Array*: Carrier frequency offset per UE, in Hz.
- ▶ *Array*: Timing offset per UE, in microseconds.

**Return type**

Array, Array

**class** `aerial.phy5g.algorithms.demapper.Demapper`

This class provides demapping of symbols to log-likelihood ratios.

The algorithm used is the exact log-MAP mapping, which is computationally intensive. Note also that this is currently implemented purely in Python so it may be slow.

`__init__(mod_order)`

Initialize demapper.

**Parameters****mod\_order** (*int*) – Modulation order. Supported values: 2, 4, 6, 8.**Return type**

None

`demap(syms, noise_var_inv)`

Run demapping.

**Parameters**

- ▶ **syms** (*np.ndarray*) – An array of modulation symbols.
- ▶ **noise\_var\_inv** (*np.ndarray*) – Inverse of noise variance per subcarrier. The size of this array must broadcast with *syms*.

**Returns**

Log-likelihood ratios. The first dimension is modulation order, otherwise the dimensions are the same as those of *syms*.

**Return type**`np.ndarray`**class** `aerial.phy5g.algorithms.trt_engine.TrtTensorPrms`

Class to hold the TRT input and output tensor parameters.

**property** `cuphy_data_type: aerial.pycuphy.DataType`

Convert data type to cuPHY data type format.

`__init__(name, dims, data_type=numpy.float32)`**Parameters**

- ▶ **name** (*str*)
- ▶ **dims** (*List[int]*)
- ▶ **data\_type** (*type*)

**Return type**

None

**class** `aerial.phy5g.algorithms.trt_engine.TrtEngine`

TensorRT engine class.

This class implements a simple wrapper around NVIDIA's TensorRT and its cuPHY API. It takes a TRT engine file as its input, along with the names and dimensions of the input and output tensors. The TRT engine file can be generated offline from an *.onnx* file using the *trtexec* tool.

```

__init__(
    trt_model_file,
    max_batch_size,
    input_tensors,
    output_tensors,
    cuda_stream=None,
)

```

Initialize TrtEngine.

#### Parameters

- ▶ **trt\_model\_file** (*str*) – This is TRT engine (model) file.
- ▶ **max\_batch\_size** (*int*) – Maximum batch size.
- ▶ **input\_tensors** (*List[TrtTensorPrms]*) – A mapping from tensor names to input tensor dimensions. The names are strings that must match with those found in the TRT model file, and the shapes are iterables of integers. The batch dimension is skipped.
- ▶ **output\_tensors** (*List[TrtTensorPrms]*) – A mapping from tensor names to output tensor dimensions. The names are strings that must match with those found in the TRT model file, and the shapes are iterables of integers. The batch dimension is skipped.
- ▶ **cuda\_stream** (*int*) – The CUDA stream. If not given, one will be created.

#### Return type

None

```
run(input_tensors)
```

Run the TensorRT model.

This runs the model using NVIDIA TensorRT engine.

#### Parameters

**input\_tensors** (*dict*) – A mapping from input tensor names to the actual input tensors. The tensor names must match with those given in the initialization, and with those found in the TRT model. Actual batch size is read from the tensor size. The tensors can be either Numpy or CuPy arrays.

#### Returns

A mapping from output tensor names to the actual output tensors.

#### Return type

dict

```
class aerial.phy5g.algorithms.srs_channel_estimator.SrsCellPrms
```

SRS cell parameters.

A list of SRS cell parameters is given to the SRS channel estimator as input, one entry per cell.

#### Parameters

- ▶ **slot\_num** (*np.uint16*) – Slot number.
- ▶ **frame\_num** (*np.uint16*) – Frame number.
- ▶ **srs\_start\_sym** (*np.uint8*) – SRS start symbol.
- ▶ **num\_srs\_sym** (*np.uint8*) – Number of SRS symbols.

- ▶ **num\_rx\_ant\_srs** (*np.uint16*) – Number of SRS Rx antennas.
- ▶ **mu** (*np.uint8*) – Subcarrier spacing parameter, see TS 38.211.

**class** `aerial.phy5g.algorithms.srs_channel_estimator.UeSrsPrms`

UE SRS parameters.

A list of UE SRS parameters is given to the SRS channel estimator as input, one entry per UE.

#### Parameters

- ▶ **cell\_idx** (*np.uint16*) – Index of cell user belongs to.
- ▶ **num\_ant\_ports** (*np.uint8*) – Number of SRS antenna ports. 1,2, or 4.
- ▶ **num\_syms** (*np.uint8*) – Number of SRS symbols. 1,2, or 4.
- ▶ **num\_repetitions** (*np.uint8*) – Number of repetitions. 1,2, or 4.
- ▶ **comb\_size** (*np.uint8*) – SRS comb size. 2 or 4.
- ▶ **start\_sym** (*np.uint8*) – Starting SRS symbol. 0 - 13.
- ▶ **sequence\_id** (*np.uint16*) – SRS sequence ID. 0 - 1023.
- ▶ **config\_idx** (*np.uint8*) – SRS bandwidth configuration idindex. 0 - 63.
- ▶ **bandwidth\_idx** (*np.uint8*) – SRS bandwidth index. 0 - 3.
- ▶ **comb\_offset** (*np.uint8*) – SRS comb offset. 0 - 3.
- ▶ **cyclic\_shift** (*np.uint8*) – Cyclic shift. 0 - 11.
- ▶ **frequency\_position** (*np.uint8*) – Frequency domain position. 0 - 67.
- ▶ **frequency\_shift** (*np.uint16*) – Frequency domain shift. 0 - 268.
- ▶ **frequency\_hopping** (*np.uint8*) – Frequency hopping options. 0 - 3.
- ▶ **resource\_type** (*np.uint8*) – Type of SRS allocation. 0: Aperiodic. 1: Semi-persistent. 2: Periodic.
- ▶ **periodicity** (*np.uint16*) – SRS periodicity in slots. 0, 2, 3, 5, 8, 10, 16, 20, 32, 40, 64, 80, 160, 320, 640, 1280, 2560.
- ▶ **offset** (*np.uint16*) – Slot offset value. 0 - 2569.
- ▶ **group\_or\_sequence\_hopping** (*np.uint8*) – Hopping configuration. 0: No hopping. 1: Group hopping. 2: Sequence hopping.
- ▶ **ch\_est\_buff\_idx** (*np.uint16*) – Index of which buffer to store SRS estimates into.
- ▶ **srs\_ant\_port\_to\_ue\_ant\_map** (*np.ndarray*) – Mapping between SRS antenna ports and UE antennas in channel estimation buffer: Store estimates for SRS antenna port *i* in `srs_ant_port_to_ue_ant_map[i]`.
- ▶ **prg\_size** (*np.uint8*) – Number of PRBs per PRG.

**class** `aerial.phy5g.algorithms.srs_channel_estimator.SrsReport`

SRS output report.

This report is returned by the SRS channel estimator.

#### Parameters



- ▶ **to\_est\_micro\_sec** (*np.float32*) – Time offset estimate in microseconds.
- ▶ **wideband\_snr** (*np.float3*) – Wideband SNR.
- ▶ **wideband\_noise\_energy** (*np.float32*) – Wideband noise energy.
- ▶ **wideband\_signal\_energy** (*np.float32*) – Wideband signal energy.
- ▶ **wideband\_sc\_corr** (*np.complex64*) – Wideband subcarrier correlation.
- ▶ **wideband\_cs\_corr\_ratio\_db** (*np.float32*)
- ▶ **wideband\_cs\_corr\_use** (*np.float32*)
- ▶ **wideband\_cs\_corr\_not\_use** (*np.float32*)

**class** aerial.phy5g.algorithms.srs\_channel\_estimator.**SrsChannelEstimator**

SrsChannelEstimator class.

This class implements SRS channel sounding for 5G NR.

**\_\_init\_\_** (*chest\_params=None, cuda\_stream=None*)

Initialize SrsChannelEstimator.

#### Parameters

- ▶ **chest\_params** (*dict*) – Dictionary of channel estimation filters and parameters. Set to None to use defaults.
- ▶ **cuda\_stream** (*int*) – The CUDA stream. If not given, one will be created.

#### Return type

None

**estimate**(  
*rx\_data,*  
*num\_srs\_ues,*  
*num\_srs\_cells,*  
*num\_prb\_grps,*  
*start\_prb\_grp,*  
*srs\_cell\_prms,*  
*srs\_ue\_prms,*  
 )

Run SRS channel estimation.

#### Parameters

- ▶ **rx\_data** (*Array*) – Input RX data, size num\_subcarriers x num\_srs\_sym x num\_rx\_ant.
- ▶ **num\_srs\_ues** (*int*) – Number of UEs.
- ▶ **num\_srs\_cells** (*int*) – Number of SRS cells.
- ▶ **num\_prb\_grps** (*int*) – Number of PRB groups.
- ▶ **start\_prb\_grp** (*int*) – Start PRB group.
- ▶ **srs\_cell\_prms** (*List[SrsCellPrms]*) – List of SRS cell parameters, one per cell.
- ▶ **srs\_ue\_prms** (*List[UeSrsPrms]*) – List of UE SRS parameters, one per UE.

**Returns**

A tuple containing:

- ▶ *List[Array]*: A list of channel estimates, one per UE. The channel estimate is a `num_prb_grps x num_rx_ant x num_tx_ant` numpy array.
- ▶ *Array*: SNRs per RB per UE.
- ▶ *List[SrsReport]*: A list of SRS wideband statistics reports, one per UE. Note: This gets copied to host memory always.

**Return type**

List[Array], Array, List[SrsReport]

**4.4.1.2 PDSCH**

This module contains classes related to the Physical Downlink Shared Channel, PDSCH, as well as to Channel State Information Reference Signals, CSI-RS.

**class** `aerial.phy5g.pdsch.pdsch_tx.PdschTx`

PDSCH transmitter.

This class implements the whole PDSCH transmission pipeline from the transmitted transport block to the transmitted frequency-domain symbols.

```
__init__(
    cell_id,
    num_rx_ant,
    num_tx_ant,
    num_ul_bwp=273,
    num_dl_bwp=273,
    mu=1,
    cuda_stream=None,
)
```

Initialize PdschTx.

**Parameters**

- ▶ **cell\_id** (*int*) – Physical cell ID.
- ▶ **num\_rx\_ant** (*int*) – Number of receive antennas.
- ▶ **num\_tx\_ant** (*int*) – Number of transmit antennas.
- ▶ **num\_ul\_bwp** (*int*) – Number of PRBs in a uplink bandwidth part. Default: 273.
- ▶ **num\_dl\_bwp** (*int*) – Number of PRBs in a downlink bandwidth part. Default: 273.
- ▶ **mu** (*int*) – Numerology. Values in [0, 3]. Default: 1.
- ▶ **cuda\_stream** (*int*) – The CUDA stream. If not given, one will be created.

**Return type**

None

```
run(
    tb_inputs,
```

```

num_ues,
slot,
num_dmrs_cdm_grps_no_data=2,
dmrs_scrm_id=41,
resource_alloc=1,
prb_bitmap=None,
start_prb=0,
num_prbs=273,
dmrs_syms=None,
start_sym=2,
num_symbols=12,
scids=None,
layers=None,
dmrs_ports=None,
bwp_starts=None,
ref_points=None,
rntis=None,
data_scids=None,
precoding_matrices=None,
mcs_tables=None,
mcs_indices=None,
code_rates=None,
mod_orders=None,
rvs=None,
num_prb_lbrms=None,
max_layers=None,
max_qms=None,
csirs_rrc_dyn_prms=None,
)

```

Run PDSCH transmission.

Set dynamic PDSCH parameters and call cuPHY to run the PDSCH transmission.

#### Parameters

- ▶ **tb\_inputs** (*List[np.ndarray]*) – Transport blocks in bytes for each UE.
- ▶ **num\_ues** (*int*) – Number of UEs.
- ▶ **slot** (*int*) – Slot number.
- ▶ **num\_dmrs\_cdm\_grps\_no\_data** (*int*) – Number of DMRS CDM groups without data [3GPP TS 38.212, sec 7.3.1.1]. Value: 1->3.
- ▶ **dmrs\_scrm\_id** (*int*) – Downlink DMRS scrambling ID.
- ▶ **resource\_alloc** (*int*) – Resource allocation type.
- ▶ **prb\_bitmap** (*List[int]*) – Array of bytes indicating bitmask for allocated RBs.
- ▶ **start\_prb** (*int*) – Start PRB index for the UE group.
- ▶ **num\_prbs** (*int*) – Number of allocated PRBs for the UE group.
- ▶ **dmrs\_syms** (*List[int]*) – For the UE group, a list of binary numbers each indicating whether the corresponding symbol is a DMRS symbol.
- ▶ **start\_sym** (*int*) – Start OFDM symbol index of the UE group allocation.

- ▶ **num\_symbols** (*int*) – Number of symbols in the allocation, starting from *start\_sym*.
- ▶ **scids** (*List[int]*) – DMRS sequence initialization for each UE [TS38.211, sec 7.4.1.1.2].
- ▶ **layers** (*List[int]*) – Number of layers for each UE.
- ▶ **dmrs\_ports** (*List[int]*) – DMRS ports for each UE. The format of each entry is in the SCF FAPI format as follows: A bitmap (mask) starting from the LSB where each bit indicates whether the corresponding DMRS port index is used.
- ▶ **bwp\_starts** (*List[int]*) – Bandwidth part start (PRB number starting from 0). Used only if reference point is 1.
- ▶ **ref\_points** (*List[int]*) – DMRS reference point per UE. Value 0 or 1.
- ▶ **rntis** (*List[int]*)
- ▶ **data\_scids** (*List[int]*) – Data scrambling IDs for each UE, more precisely *dataScramblingIdentityPdsch* [TS38.211, sec 7.3.1.1].
- ▶ **precoding\_matrices** (*List[np.ndarray]*) – Precoding matrices, one per UE. The shape of each precoding matrix is number of layers x number of Tx antennas. If set to None, precoding is disabled.
- ▶ **mcs\_tables** (*List[int]*) – MCS table per UE.
- ▶ **mcs\_indices** (*List[int]*) – MCS index per UE.
- ▶ **code\_rates** (*List[int]*) – Code rate for each UE in 3GPP format, i.e. code rate x 1024.
- ▶ **mod\_orders** (*List[int]*) – Modulation order for each UE.
- ▶ **rvs** (*List[int]*) – Redundancy version per UE (default: 0 for each UE).
- ▶ **num\_prb\_lbrms** (*List[int]*) – Number of PRBs used for LBRM TB size computation. Possible values: {32, 66, 107, 135, 162, 217, 273}.
- ▶ **max\_layers** (*List[int]*) – Number of layers used for LBRM TB size computation (at most 4).
- ▶ **max\_qms** (*List[int]*) – Modulation order used for LBRM TB size computation. Value: 6 or 8.
- ▶ **csirs\_rrc\_dyn\_prms** (*List[CsiRsRrcDynPrms]*) – List of CSI-RS RRC dynamic parameters, see *CsiRsRrcDynPrms*. Note that no CSI-RS symbols get written, this is only to make sure that PDSCH does not get mapped to the CSI-RS resource elements.

### Returns

A tuple containing:

- ▶ *np.ndarray*: Transmitted OFDM symbols in a frequency x time x antenna tensor.
- ▶ *np.ndarray*: Coded bits in a num\_codewords x num\_bits\_per\_codeword tensor.

### Return type

*np.ndarray*, *np.ndarray*

```

classmethod cuphy_to_tx(
    tx_slot,
    num_ues,
    dmrs_ports,
    scids,
    precoding_matrices=None,
)

```

Map cuPHY outputs to Tx antenna ports.

#### Parameters

- ▶ **tx\_slot** (*numpy.ndarray*) – Transmit buffer from cuPHY.
- ▶ **num\_ues** (*int*) – Number of UEs.
- ▶ **dmrs\_ports** (*List[int]*) – DMRS ports for each UE. The format of each entry is in the SCF FAPI format as follows: A bitmap (mask) starting from the LSB where each bit indicates whether the corresponding DMRS port index is used.
- ▶ **scids** (*List[int]*) – DMRS sequence initialization for each UE [TS38.211, sec 7.4.1.1.2].
- ▶ **precoding\_matrices** (*List[np.ndarray]*) – Precoding matrices, one per UE. The shape of each precoding matrix is number of layers x number of Tx antennas. If set to None, precoding is disabled.

#### Returns

Transmitted OFDM symbols in a frequency x time x antenna tensor.

#### Return type

*np.ndarray*

```

class aerial.phy5g.pdsch.csirs_tx.CsiRsTx

```

CSI-RS transmitter.

This class implements CSI-RS transmission within a slot.

```

__init__(num_prb_dl_bwp, cuda_stream=None)

```

Initialize CsiRsTx.

#### Parameters

- ▶ **num\_prb\_dl\_bwp** (*List[int]*) – Number of PRBs in DL BWP.
- ▶ **cuda\_stream** (*int*) – The CUDA stream. If not given, one will be created.

#### Return type

None

```

run(csirs_cell_dyn_prms, tx_buffers, precoding_matrices=None)

```

Run CSI-RS transmission.

#### Parameters

- ▶ **csirs\_cell\_dyn\_prms** (*List[CsiRsCellDynPrms]*) – A list of CSI-RS cell dynamic parameters, one entry per cell. See *CsiRsCellDynPrms*.
- ▶ **tx\_buffers** (*List[np.ndarray]*) – A list of transmit slot buffers, one per cell. These represent the slot buffers prior to inserting the CSI-RS.

- ▶ **precoding\_matrices** (*List[CsiRsPmwOneLayer]*) – A list of precoding matrices. This list gets indexed by the *pmw\_prm\_idx* field in *CsiRsRrcDynPrms* (part of *CsiRsCellDynPrms*).

**Returns**

Transmit buffers for the slot for each cell after inserting CSI-RS.

**Return type**

List[np.ndarray]

**4.4.1.3 PUSCH**

This module contains classes related to the Physical Uplink Shared Channel, PUSCH.

**class** `aerial.phy5g.pusch.pusch_rx.PuschRx`

PUSCH receiver pipeline.

This class implements the whole PUSCH reception pipeline from the received OFDM post-FFT symbols to the received transport block (along with CRC check).

```
__init__(
    cell_id,
    num_rx_ant,
    num_tx_ant,
    num_ul_bwp=273,
    num_dl_bwp=273,
    mu=1,
    enable_cfo_correction=0,
    enable_to_estimation=0,
    enable_pusch_tdi=0,
    eq_coeff_algo=1,
    enable_per_prg_chest=0,
    enable_ul_rx_bf=0,
    ldpc_kernel_launch=aerial.pycuphy.PuschLdpcKernelLaunch.PUSCH_RX_ENABLE_DRIVER_LDPC_LAUNCH,
)
```

Initialize PuschRx.

**Parameters**

- ▶ **cell\_id** (*int*) – Physical cell ID.
- ▶ **num\_rx\_ant** (*int*) – Number of receive antennas.
- ▶ **num\_tx\_ant** (*int*) – Number of transmit antennas.
- ▶ **num\_ul\_bwp** (*int*) – Number of PRBs in a uplink bandwidth part. Default: 273.
- ▶ **num\_dl\_bwp** (*int*) – Number of PRBs in a downlink bandwidth part. Default: 273.
- ▶ **mu** (*int*) – Numerology. Values in [0, 3]. Default: 1.
- ▶ **enable\_cfo\_correction** (*int*) – Enable/disable CFO correction:
  - ▶ 0: Disable (default).
  - ▶ 1: Enable.
- ▶ **enable\_to\_estimation** (*int*) – Enable/disable time offset estimation:

- ▶ 0: Disable (default).
- ▶ 1: Enable.
- ▶ **enable\_pusch\_tdi** (*int*) – Time domain interpolation on PUSCH.
  - ▶ 0: Disable (default).
  - ▶ 1: Enable.
- ▶ **eq\_coeff\_algo** (*int*) – Algorithm for equalizer coefficient computation.
  - ▶ 0 - ZF.
  - ▶ 1 - MMSE (default).
  - ▶ 2 - MMSE-IRC.
- ▶ **enable\_per\_prg\_chest** (*int*) – Enable/disable PUSCH per-PRG channel estimation.
  - ▶ 0: Disable (default).
  - ▶ 1: Enable.
- ▶ **enable\_ul\_rx\_bf** (*int*) – Enable/disable beamforming for PUSCH.
  - ▶ 0: Disable (default).
  - ▶ 1: Enable.
- ▶ **ldpc\_kernel\_launch** (*PuschLdpcKernelLaunch*) – LDPC kernel launch method.

**Return type**

None

```
run(
  rx_slot,
  slot=0,
  pusch_configs=None,
  num_ues=1,
  num_dmrs_cdm_grps_no_data=2,
  dmrs_scrm_id=41,
  start_prb=0,
  num_prbs=273,
  prg_size=1,
  num_ul_streams=1,
  dmrs_syms=None,
  dmrs_max_len=2,
  dmrs_add_ln_pos=1,
  start_sym=2,
  num_symbols=12,
  scids=None,
  layers=None,
  dmrs_ports=None,
  rntis=None,
  data_scids=None,
  mcs_tables=None,
  mcs_indices=None,
  code_rates=None,
  mod_orders=None,
```

```

    tb_sizes=None,
    rvs=None,
    ndis=None,
)

```

Run PUSCH Rx.

This runs the cuPHY PUSCH receiver pipeline based on the given parameters. Multiple UE groups are supported if the *PuschConfig* based API is used. Otherwise, the pipeline gets run only for a single UE group sharing the same time-frequency resources, i.e. having the same PRB allocation, and the same start symbol and number of allocated symbols. In this case default values get filled for the parameters that are not given.

### Parameters

- ▶ **rx\_slot** (*Array*) – A tensor representing the receive slot buffer of the cell.
- ▶ **slot** (*int*) – Slot number.
- ▶ **pusch\_configs** (*List[PuschConfig]*) – List of PUSCH configuration objects, one per UE group. If this argument is given, the rest are ignored. If not given, the other arguments will be used (default values are used for the parameters that are not given). Only one UE group is supported in that case.
- ▶ **num\_ues** (*int*) – Number of UEs in the UE group.
- ▶ **num\_dmrs\_cdm\_grps\_no\_data** (*int*) – Number of DMRS CDM groups without data [3GPP TS 38.212, sec 7.3.1.1]. Value: 1->3.
- ▶ **dmrs\_scrm\_id** (*int*) – DMRS scrambling ID.
- ▶ **start\_prb** (*int*) – Start PRB index of the UE group allocation.
- ▶ **num\_prbs** (*int*) – Number of allocated PRBs for the UE group.
- ▶ **prg\_size** (*int*) – The Size of PRG in PRB for PUSCH per-PRG channel estimation.
- ▶ **nUplinkStreams** (*int*) – The number of active streams for this PUSCH.
- ▶ **dmrs\_syms** (*List[int]*) – For the UE group, a list of binary numbers each indicating whether the corresponding symbol is a DMRS symbol.
- ▶ **dmrs\_max\_len** (*int*) – The *maxLength* parameter, value 1 or 2, meaning that DMRS are single-symbol DMRS or single- or double-symbol DMRS.
- ▶ **dmrs\_add\_ln\_pos** (*int*) – Number of additional DMRS positions.
- ▶ **start\_sym** (*int*) – Start OFDM symbol index for the UE group allocation.
- ▶ **num\_symbols** (*int*) – Number of symbols in the UE group allocation.
- ▶ **scids** (*List[int]*) – DMRS sequence initialization for each UE [TS38.211, sec 7.4.1.1.2].
- ▶ **layers** (*List[int]*) – Number of layers for each UE.
- ▶ **dmrs\_ports** (*List[int]*) – DMRS ports for each UE. The format of each entry is in the SCF FAPI format as follows: A bitmap (mask) starting from the LSB where each bit indicates whether the corresponding DMRS port index is used.
- ▶ **rntis** (*List[int]*)
- ▶ **data\_scids** (*List[int]*) – Data scrambling IDs for each UE, more precisely *dataScramblingIdentityPdsch* [TS38.211, sec 7.3.1.1].



- ▶ **mcs\_tables** (*List[int]*) – MCS table to use for each UE (see TS 38.214).
- ▶ **mcs\_indices** (*List[int]*) – MCS indices for each UE.
- ▶ **code\_rates** (*List[float]*) – Code rate for each UE. This is the number of information bits per 1024 coded bits.
- ▶ **mod\_orders** (*List[int]*) – Modulation order for each UE.
- ▶ **tb\_sizes** (*List[int]*) – TB size in bytes for each UE.
- ▶ **rvs** (*List[int]*) – Redundancy versions for each UE.
- ▶ **ndis** (*List[int]*) – New data indicator per UE.
- ▶ **num\_ul\_streams** (*int*)

**Returns**

A tuple containing:

- ▶ *Array*: Transport block CRCs.
- ▶ *List[Array]*: Transport blocks, one per UE, without CRC.

**Return type**

*Array*, *List[Array]*

**4.4.1.4 LDPC 5G**

This module contains the API for using the GPU-accelerated LDPC coding chain from the cuPHY library. This includes encoding and decoding, rate matching and CRC checking. Additionally, this module contains a number of utility functions for example for determining the LDPC base graph, transport block size, etc.

**class** `aerial.phy5g.ldpc.decoder.LdpcDecoder`

LDPC decoder.

This class supports decoding of LDPC code blocks encoded following TS 38.212. It uses cuPHY accelerated LDPC decoding routines under the hood.

```
__init__(
    num_iterations=10,
    throughput_mode=False,
    cuda_stream=None,
)
```

Initialize LdpcDecoder.

**Parameters**

- ▶ **num\_iterations** (*int*) – Number of LDPC decoder iterations. Default: 10.
- ▶ **throughput\_mode** (*bool*) – Enable throughput mode. Default: False.
- ▶ **cuda\_stream** (*int*) – The CUDA stream. If not given, one will be created.

**Return type**

None

```

decode(
    *,
    input_llrs,
    pusch_configs=None,
    tb_sizes=None,
    code_rates=None,
    redundancy_versions=None,
    rate_match_lengths=None,
    num_iterations=None,
)

```

Decode function for LDPC decoder.

The decoder outputs decoded code blocks which can be further concatenated into the received transport block using [CrcChecker](#).

The method can be called using either Numpy or CuPy arrays. In case the input arrays are located on the GPU (CuPy), the output will be on the GPU (CuPy). So the return type shall be the same as used for *input\_llrs* when calling the method.

#### Parameters

- ▶ **input\_llrs** (*List[Array]*) – Input LLRs per UE, each array is a N x C array of 32-bit floats, N being the number of LLRs per code block and C being the number of code blocks.
- ▶ **pusch\_configs** (*List[PuschConfig]*) – List of PUSCH configuration objects, one per UE group. If this argument is given, the rest are ignored. If not given, all other arguments need to be given.
- ▶ **tb\_sizes** (*List[int]*) – Transport block size in bits, without CRC, per UE.
- ▶ **code\_rates** (*List[float]*) – Target code rates per UE.
- ▶ **redundancy\_versions** (*List[int]*) – Redundancy version, i.e. 0, 1, 2, or 3, per UE.
- ▶ **rate\_match\_lengths** (*int*) – Number of rate matching output bits of each UE. This is equal to N.
- ▶ **num\_iterations** (*int*) – Number of LDPC iterations. If not given, use the default from the constructor.

#### Returns

The decoded bits.

#### Return type

List[Array]

```
set_num_iterations(num_iterations)
```

Set a particular value for the number of iterations to be run.

#### Parameters

**num\_iterations** (*int*) – Value of the number of iterations.

#### Return type

None

```
set_throughput_mode(throughput_mode)
```

Enable throughput mode.

**Parameters**

**throughput\_mode** (*bool*) – Enable (True) throughput mode.

**Return type**

None

**get\_soft\_bits()**

Get the soft bit output from the decoder.

**Returns**

The soft bits in an array.

**Return type**

List[Array]

**class** `aerial.phy5g.ldpc.encoder.LdpcEncoder`

LDPC encoder.

This class provides encoding of transmitted transport block bits using LDPC coding following TS 38.212. The encoding process is GPU accelerated using cuPHY routines. As the input, the transport blocks are assumed to be attached with the CRC and segmented to code blocks (as per TS 38.212).

```
__init__(
    num_profiling_iterations=0,
    puncturing=True,
    max_num_code_blocks=152,
    cuda_stream=None,
)
```

Initialize LdpcEncoder.

Initialization does all the necessary memory allocations for cuPHY.

**Parameters**

- ▶ **num\_profiling\_iterations** (*int*) – Number of profiling iterations. Set to 0 to disable profiling. Default: 0.
- ▶ **puncturing** (*bool*) – Whether to puncture the systematic bits (2Zc). Default: True.
- ▶ **max\_num\_code\_blocks** (*int*) – Maximum number of code blocks. Memory is allocated based on this. Default: 152.
- ▶ **cuda\_stream** (*int*) – The CUDA stream. If not given, one will be created.

**Return type**

None

**encode**(*input\_data, tb\_size, code\_rate, redundancy\_version*)

Encode function for LDPC encoder.

The input to this function is code blocks, meaning that the code block segmentation is expected to be done before calling this function. Code block segmentation can be done using `code_block_segment()`.

**Parameters**

- ▶ **input\_data** (*np.ndarray*) – The input code blocks as a  $K \times C$  array where  $K$  is the number of input bits per code block (including CRCs) and  $C$  is the number of code blocks. The dtype of the input array must be *np.float32*.
- ▶ **tb\_size** (*int*) – Transport block size in bits, without CRC.
- ▶ **code\_rate** (*float*) – Target code rate.
- ▶ **redundancy\_version** (*int*) – Redundancy version, 0, 1, 2, or 3.

**Returns**

Encoded bits as a  $N \times C$  array where  $N$  is the number of encoded bits per code block.

**Return type**

*np.ndarray*

**set\_profiling\_iterations**(*num\_profiling\_iterations*)

Set a particular value for the number of profiling iterations to be run.

**Parameters**

**num\_profiling\_iterations** (*int*) – Value of the number of profiling iterations.

**Return type**

None

**set\_puncturing**(*puncturing*)

Set puncturing flag.

**Parameters**

**puncturing** (*bool*) – Whether to puncture the systematic bits ( $2 \times Z_c$ ). Default: True.

**Return type**

None

**class** `aerial.phy5g.ldpc.rate_match.LdpcRateMatch`

LDPC rate matching.

```
__init__(
    enable_scrambling=True,
    num_profiling_iterations=0,
    max_num_code_blocks=152,
    cuda_stream=None,
)
```

Initialize LdpcRateMatch.

Initialization does all the necessary memory allocations for cuPHY.

**Parameters**

- ▶ **enable\_scrambling** (*bool*) – Whether to enable scrambling after code block concatenation.
- ▶ **num\_profiling\_iterations** (*int*) – Number of profiling iterations. Set to 0 to disable profiling. Default: 0 (no profiling).
- ▶ **max\_num\_code\_blocks** (*int*) – Maximum number of code blocks. Memory will be allocated based on this number.
- ▶ **cuda\_stream** (*int*) – The CUDA stream. If not given, one will be created.

**Return type**

None

```

rate_match(
    input_data,
    tb_size,
    code_rate,
    rate_match_len,
    mod_order,
    num_layers,
    redundancy_version,
    cinit,
)

```

LDPC rate matching function.

This function does rate matching of LDPC code blocks following TS 38.212. If scrambling is enabled, it also scrambles the rate matched bits. In this case the *c\_init* value needs to be set to an appropriate scrambling sequence initialization value.

**Parameters**

- ▶ **input\_data** (*np.ndarray*) – Input bits as a N x C numpy array with dtype *np.float32*, where N is the number of bits per code block and C is the number of code blocks.
- ▶ **tb\_size** (*int*) – Transport block size in bits without CRC.
- ▶ **code\_rate** (*float*) – Code rate.
- ▶ **rate\_match\_len** (*int*) – Number of rate matching output bits.
- ▶ **mod\_order** (*int*) – Modulation order.
- ▶ **num\_layers** (*int*) – Number of layers.
- ▶ **redundancy\_version** (*int*) – Redundancy version, i.e. 0, 1, 2, or 3.
- ▶ **cinit** (*int*) – The *c\_init* value used for initializing scrambling.

**Returns**

Rate matched bits.

**Return type***np.ndarray*

```

set_profiling_iterations(num_profiling_iterations)

```

Set a particular value for the number of profiling iterations to be run.

**Parameters**

**num\_profiling\_iterations** (*int*) – Value of the number of profiling iterations.

**Return type**

None

```

class aerial.phy5g.ldpc.derate_match.LdpcDeRateMatch

```

LDPC derate matching.

```

__init__(enable_scrambling=True, cuda_stream=None)

```

Initialize LdpcDeRateMatch.

Initialization does all the necessary memory allocations for cuPHY.

**Parameters**

- ▶ **enable\_scrambling** (*bool*) – Whether to descramble the bits before derate matching. Default: True.
- ▶ **cuda\_stream** (*int*) – The CUDA stream. If not given, one will be created.

**Return type**

None

```

derate_match(
    *,
    input_llrs,
    pusch_configs=None,
    tb_sizes=None,
    code_rates=None,
    rate_match_lengths=None,
    mod_orders=None,
    num_layers=None,
    redundancy_versions=None,
    ndis=None,
    cinits=None,
    ue_grp_idx=None,
)

```

LDPC derate matching function.

**Parameters**

- ▶ **input\_llrs** (*List[Array]*) – Input LLRs as a  $N \times 1$  array with dtype *np.float32*, where  $N$  is the number of LLRs coming from the equalizer. Ordering of this input data is *bitsPerQam x numLayers x numSubcarriers x numDataSymbols*. One entry per UE group.
- ▶ **pusch\_configs** (*List[PuschConfig]*) – List of PUSCH configuration objects, one per UE group. If this argument is given, the rest are ignored. If not given, all other arguments need to be given.
- ▶ **tb\_sizes** (*List[int]*) – Transport block sizes in bits without CRC, per UE.
- ▶ **code\_rates** (*List[float]*) – Code rates per UE.
- ▶ **rate\_match\_lengths** (*List[int]*) – Number of rate matching output bits, the same as  $N$ , per UE.
- ▶ **mod\_orders** (*List[int]*) – Modulation order per UE.
- ▶ **num\_layers** (*List[int]*) – Number of layers per UE.
- ▶ **redundancy\_versions** (*List[int]*) – Redundancy version, i.e. 0, 1, 2, or 3, per UE.
- ▶ **ndis** (*List[int]*) – New data indicator per UE.
- ▶ **cinits** (*List[int]*) – The *c\_init* value used for initializing scrambling for each UE.
- ▶ **ue\_grp\_idx** (*List[int]*) – The UE group index for each UE. Default is one-to-one mapping.

**Returns**

Derate matched LLRs for each UE.

**Return type**

List[Array]

**class** aerial.phy5g.ldpc.crc\_check.CrcChecker

CRC checking.

This class supports decoding the code block CRCs, desegmenting code blocks together, assembling the transport block and also finally decoding the transport block CRCs. It uses cuPHY accelerated CRC routines under the hood.

**\_\_init\_\_**(*cuda\_stream=None*)

Initialize CrcChecker.

**Parameters**

**cuda\_stream** (*int*) – The CUDA stream. If not given, one will be created.

**Return type**

None

**check\_crc**(

```

    *,
    input_bits,
    pusch_configs=None,
    tb_sizes=None,
    code_rates=None,
)

```

CRC checking.

This method takes LDPC decoder output as its input, checks the code block CRCs, desegments code blocks, combines them into a transport block and checks the transport block CRC. It returns the transport block payloads without CRC, as well as the transport block CRC check results. The code block CRC results are stored as well and may be queried separately.

**Parameters**

- ▶ **input\_bits** (*List[Array]*) – LDPC decoder outputs per UE, each array is a  $K \times C$  array of 32-bit floats,  $K$  being the number of bits per code block and  $C$  being the number of code blocks.
- ▶ **pusch\_configs** (*List[PuschConfig]*) – List of PUSCH configuration objects, one per UE group. If this argument is given, the rest are ignored. If not given, all other arguments need to be given.
- ▶ **tb\_sizes** (*List[int]*) – Transport block size in bits, without CRC, per UE.
- ▶ **code\_rates** (*List[float]*) – Target code rates per UE.

**Returns**

A tuple containing:

- ▶ *List[Array]*: Transport block payloads in bytes, without CRC, for each UE.
- ▶ *List[Array]*: Transport block CRC check results for each UE.

**Return type**

List[Array], List[Array]

`aerial.phy5g.ldpc.util.get_mcs(mcs, table_idx=2)`

Get modulation order and code rate based on MCS index.

#### Parameters

- ▶ **mcs** (*int*) – MCS index pointing to the table indicated by *table\_idx*.
- ▶ **table\_idx** (*int*) – Index of the MCS table in TS 38.214 section 5.1.3.1. Values: - 1: TS38.214, table 5.1.3.1-1. - 2: TS38.214, table 5.1.3.1-2. - 3: TS38.214, table 5.1.3.1-3.

#### Returns

A tuple containing:

- ▶ *int*: Modulation order.
- ▶ *float*: Code rate \* 1024.

#### Return type

*int*, *float*

```
aerial.phy5g.ldpc.util.get_tb_size(  
    mod_order,  
    code_rate,  
    dmrs_syms,  
    num_prbs,  
    start_sym,  
    num_symbols,  
    num_layers,  
)
```

Get transport block size based on given parameters.

Determine transport block size as per TS 38.214 section 5.1.3.2.

#### Parameters

- ▶ **mod\_order** (*int*) – Modulation order.
- ▶ **code\_rate** (*float*) – Code rate \* 1024 as in section 5.1.3.1 of TS 38.214.
- ▶ **dmrs\_syms** (*List[int]*) – List of binary numbers indicating which symbols contain DMRS.
- ▶ **num\_prbs** (*int*) – Number of PRBs.
- ▶ **start\_sym** (*int*) – Starting symbol.
- ▶ **num\_symbols** (*int*) – Number of symbols.
- ▶ **num\_layers** (*int*) – Number of layers.

#### Returns

Transport block size in bits.

#### Return type

*int*

```
aerial.phy5g.ldpc.util.get_base_graph(tb_size, code_rate)
```

Get LDPC base graph.



**Parameters**

- ▶ **tb\_size** (*int*) – Transport block size in bits, without CRC.
- ▶ **code\_rate** (*float*) – Code rate.

**Returns**

Base graph, 1 or 2.

**Return type**

int

`aerial.phy5g.ldpc.util.max_code_block_size(base_graph)`

Get maximum LDPC code block size based on base graph.

**Parameters**

**base\_graph** (*int*) – Base graph, 1 or 2.

**Returns**

Maximum code block size.

**Return type**

int

`aerial.phy5g.ldpc.util.find_lifting_size(base_graph, tb_size)`

Find lifting size for base graph.

**Parameters**

- ▶ **base\_graph** (*int*) – Base graph, 1 or 2.
- ▶ **tb\_size** (*int*) – Transport block size in bits without CRC.

**Returns**

Lifting size.

**Return type**

int

`aerial.phy5g.ldpc.util.get_num_info_nodes(base_graph, tb_size)`

Get number of information nodes.

Note: This is the value  $K_b$  in TS 38.212.

**Parameters**

- ▶ **base\_graph** (*int*) – Base graph, 1 or 2.
- ▶ **tb\_size** (*int*) – Transport block size without any CRCs.

**Returns**

The number of information nodes ( $K_b$ ).

**Return type**

int

`aerial.phy5g.ldpc.util.get_code_block_num_info_bits(base_graph, tb_size)`

Get number of information bits in a code block.

This is the number  $K'$  in TS 38.212, i.e. the number of information bits without the filler bits.

**Parameters**

- ▶ **base\_graph** (*int*) – Base graph, 1 or 2.
- ▶ **tb\_size** (*int*) – Transport block size in bits, without CRC.

**Returns**

Number of information bits in a code block.

**Return type**

*int*

```
aerial.phy5g.ldpc.util.get_code_block_size(tb_size, code_rate)
```

Get code block size.

This is the number K in TS 38.212, i.e. the number of information bits including filler bits.

**Parameters**

- ▶ **tb\_size** (*int*) – Transport block size in bits, without CRC.
- ▶ **code\_rate** (*float*) – Code rate.

**Returns**

Code block size.

**Return type**

*int*

```
aerial.phy5g.ldpc.util.get_num_code_blocks(tb_size, code_rate)
```

Return the number of code blocks for a transport block.

**Parameters**

- ▶ **tb\_size** (*int*) – Transport block size in bits, without CRC.
- ▶ **code\_rate** (*float*) – Code rate.

**Returns**

The number of code blocks (C).

**Return type**

*int*

```
aerial.phy5g.ldpc.util.code_block_segment(tb_size, transport_block, code_rate)
```

Do code block segmentation.

This function does code block segmentation as per TS 38.212 section 5.2.2. Randomly generated 24-bit string is attached to each code block to emulate code block CRC if there is more than one code block.

**Parameters**

- ▶ **tb\_size** (*int*) – Transport block size in bits, without CRC.
- ▶ **transport\_block** (*np.ndarray*) – Transport block in bits, CRC included.
- ▶ **code\_rate** (*float*) – Code rate.

**Returns**

The code blocks.

**Return type**

np.ndarray

```
aerial.phy5g.ldpc.util.code_block_desegment(
    code_blocks,
    tb_size,
    code_rate,
    return_bits=True,
)
```

Concatenate code blocks coming from LDPC decoding into a transport block.

This function desegments code blocks into a transport block as per TS 38.212, and removes the CRCs, i.e. does the opposite of `code_block_segment()`.

**Parameters**

- ▶ **code\_blocks** (*np.ndarray*) – The code blocks coming out of the LDPC decoder as a N x C array.
- ▶ **tb\_size** (*int*) – Transport block size in bits, without CRC.
- ▶ **code\_rate** (*float*) – Code rate.
- ▶ **return\_bits** (*bool*) – If True (default), give the return value in bits. Otherwise convert to bytes.

**Returns**

The transport block with CRC, in bits or bytes depending on the value of `return_bits`.

**Return type**

np.ndarray

```
aerial.phy5g.ldpc.util.add_crc_len(tb_size)
```

Append CRC length to transport block size.

**Parameters**

**tb\_size** (*int*) – Transport block size in bits without CRC.

**Returns**

Transport block size in bits with CRC.

**Return type**

int

```
aerial.phy5g.ldpc.util.random_tb(
    mod_order,
    code_rate,
    dmrs_syms,
    num_prbs,
    start_sym,
    num_symbols,
    num_layers,
    return_bits=False,
)
```

Generate a random transport block.

Generates random transport block according to given parameters. The transport block size is first determined as per TS 38.214 section 5.1.3.2.

**Parameters**

- ▶ **mod\_order** (*int*) – Modulation order.
- ▶ **code\_rate** (*float*) – Code rate \* 1024 as in section 5.1.3.1 of TS 38.214.
- ▶ **dmrs\_syms** (*List[int]*) – List of binary numbers indicating which symbols contain DMRS.
- ▶ **num\_prbs** (*int*) – Number of PRBs.
- ▶ **start\_sym** (*int*) – Starting symbol.
- ▶ **num\_symbols** (*int*) – Number of symbols.
- ▶ **num\_layers** (*int*) – Number of layers.
- ▶ **return\_bits** (*bool*) – Whether to return the transport block in bits (True) or bytes (False).

**Returns**

Random transport block payload.

**Return type**

np.ndarray

`aerial.phy5g.ldpc.util.get_crc_len(tb_size)`

Return CRC length based on transport block size.

**Parameters**

**tb\_size** (*int*) – Transport block size in bits without CRC.

**Returns**

CRC length (either 16 or 24 bits).

**Return type**

int

**4.4.1.5 Fading channel**

This module contains the `FadingChan` class which can be used to simulate OFDM transmission through a tapped delay line (TDL) fading channel. The implementation is using GPU acceleration.

**class** `aerial.phy5g.chan_models.fading_chan.FadingChan`

Fading channel class.

This class implements the fading channel that processes the frequency Tx samples and outputs frequency Rx samples. It includes OFDM modulation, tapped delay line (TDL) channel, OFDM demodulation, and adds noise based on input SNR.

```
__init__(
    cuphy_carrier_prms,
    tdl_cfg,
    freq_in,
    proc_sig_freq=False,
    disable_noise=False,
    rand_seed=0,
)
```

Initialize the `FadingChan` class.

- ▶ `cuphy_carrier_prms`: carrier parameters for the channel
- ▶ `tdl_config`: configuration of TDL channel
- ▶ `freq_in`: input frequency tx
- ▶ `proc_sig_freq`: processing signal in freq domain will use the CFR from TDL class to process data on frequency domain. This mode may be inaccurate if CFO presents.
- ▶ `disable_noise`: disable additive Gaussian noise
- ▶ `rand_seed`: random seed for TDL channel generation

#### Parameters

- ▶ `cuphy_carrier_prms` (*aerial.pycuphy.CuphyCarrierPrms*)
- ▶ `tdl_cfg` (*aerial.pycuphy.TdlConfig*)
- ▶ `freq_in` (*numpy.ndarray*)
- ▶ `proc_sig_freq` (*bool*)
- ▶ `disable_noise` (*bool*)
- ▶ `rand_seed` (*int*)

#### Return type

None

`add_noise_with_snr` (*snr\_db, enable\_swap\_tx\_rx=False*)

Add Gaussian noise to a complex signal with a specified SNR.

#### Parameters

- ▶ `snr_db` (*float*) – Desired Signal-to-Noise Ratio in decibels.
- ▶ `enable_swap_tx_rx` (*bool*) – Swap tx and rx to simulate UL channel using DL class.

#### Returns

The frequency-domain signal with noise added.

#### Return type

`np.ndarray`

`add_noise_with_snr_numpy` (*snr\_db, enable\_swap\_tx\_rx=False*)

Add Gaussian noise to a complex signal with a specified SNR.

#### Parameters

- ▶ `snr_db` (*float*) – Desired Signal-to-Noise Ratio in decibels.
- ▶ `enable_swap_tx_rx` (*bool*) – Swap tx and rx to simulate UL channel using DL class.
- ▶ `pycuphy.GauNoiseAdder` (*This function is GPU-accelerated by `add_noise_with_snr()` and*)

#### Returns

The frequency-domain signal with noise added.

**Return type**

np.ndarray

**dump\_channel** (*freq\_in=None*)

Dump TDL channel to numpy arrays.

**Returns**

A tuple containing two numpy arrays (cfr\_sc, cfr\_prbg).

**Return type**

tuple

**Parameters****freq\_in** (*numpy.ndarray | None*)**get\_genie\_channel** (*freq\_in=None, cfr\_sc=None*)

need to do ofdm demodulation of the rx-tx ant pair sample to get genie channel

**Parameters**▶ **freq\_in** (*numpy.ndarray | None*)▶ **cfr\_sc** (*numpy.ndarray | None*)**Return type**

None

```
run(  
    tti_idx,  
    snr_db,  
    enable_swap_tx_rx=False,  
    continuous_fading=True,  
    tx_column_major_ind=False,  
    freq_in=None,  
)
```

Run the fading channel.

**Parameters**▶ **tti\_idx** (*int*) – TTI index.▶ **snr\_db** (*float*) – Signal-to-Noise Ratio in dB.▶ **enable\_swap\_tx\_rx** (*bool*) – Swap tx and rx to simulate UL channel using DL class.▶ **continuous\_fading** (*bool*) – Whether enable continuous fading.▶ **freq\_in** (*np.ndarray*) – Frequency domain input samples.▶ **tx\_column\_major\_ind** (*bool*)**Returns**

Frequency domain samples after channel processing.

**Return type**

np.ndarray

#### 4.4.1.6 Parameter classes

This module contains classes to hold various parameters. In particular, the PUSCH receiver pipeline and its components accept *PuschConfig* and *PuschUeConfig* as arguments.

##### **class** aerial.phy5g.params.**PuschUeConfig**

A class holding all dynamic PUSCH parameters for a single slot, single UE.

##### **Parameters**

- ▶ **scid** (*int*) – DMRS sequence initialization [TS38.211, sec 7.4.1.1.2].
- ▶ **layers** (*int*) – Number of layers.
- ▶ **dmrs\_ports** (*int*) – Allocated DMRS ports.
- ▶ **rnti** (*int*) – The 16-bit RNTI value of the UE.
- ▶ **data\_scid** (*List[int]*) – Data scrambling ID, more precisely *dataScramblingIdentityPdsch* [TS38.211, sec 7.3.1.1].
- ▶ **mcs\_table** (*int*) – MCS table to use (see TS 38.214).
- ▶ **mcs\_index** (*int*) – MCS index to use.
- ▶ **code\_rate** (*int*) – Code rate, expressed as the number of information bits per 1024 coded bits expressed in 0.1 bit units.
- ▶ **mod\_order** (*int*) – Modulation order.
- ▶ **tb\_size** (*int*) – TB size in bytes.
- ▶ **rv** (*List[int]*) – Redundancy version.
- ▶ **ndi** (*List[int]*) – New data indicator.

##### **class** aerial.phy5g.params.**PuschConfig**

A class holding all dynamic PUSCH parameters for a single slot, single UE group.

##### **Parameters**

- ▶ **num\_dmrs\_cdm\_grps\_no\_data** (*int*) – Number of DMRS CDM groups without data [3GPP TS 38.212, sec 7.3.1.1]. Value: 1->3.
- ▶ **dmrs\_scrm\_id** (*int*) – DMRS scrambling ID.
- ▶ **start\_prb** (*int*) – Start PRB index of the UE group allocation.
- ▶ **num\_prbs** (*int*) – Number of allocated PRBs for the UE group.
- ▶ **prg\_size** (*int*) – The Size of PRG in PRB for PUSCH per-PRG channel estimation.
- ▶ **num\_ul\_streams** (*int*) – The number of active streams for this PUSCH.
- ▶ **dmrs\_syms** (*List[int]*) – For the UE group, a list of binary numbers each indicating whether the corresponding symbol is a DMRS symbol.
- ▶ **dmrs\_max\_len** (*int*) – The *maxLength* parameter, value 1 or 2, meaning that DMRS are single-symbol DMRS or single- or double-symbol DMRS. Note that this needs to be consistent with *dmrs\_syms*.
- ▶ **dmrs\_add\_1n\_pos** (*int*) – Number of additional DMRS positions. Note that this needs to be consistent with *dmrs\_syms*.

- ▶ **start\_sym** (*int*) – Start OFDM symbol index for the UE group allocation.
- ▶ **num\_symbols** (*int*) – Number of symbols in the UE group allocation.

## 4.4.2. Utilities

### 4.4.2.1 FAPI and Matlab interface utilities

The FAPI module contains various utilities for handling the interface between the PUSCH database schema (SCF FAPI) and cuPHY.

`aerial.util.fapi.dmrs_fapi_to_bit_array(dmrs_symb_pos)`

Convert the DMRS symbol position decimal value to a bit array.

**Parameters**

**dmrs\_symb\_pos** (*np.uint16*) – DMRS symbol position decimal value as defined in SCF FAPI.

**Returns**

A bit array to be used for cuPHY interface, indicating the positions of DMRS symbols. The first bit corresponds to OFDM symbol 0.

**Return type**

list

`aerial.util.fapi.dmrs_bit_array_to_fapi(x)`

Convert a bit array to DMRS symbol position decimal value.

**Parameters**

**x** (*list*) – A bit array to be used for cuPHY interface, indicating the positions of DMRS symbols. The first bit corresponds to OFDM symbol 0.

**Returns**

DMRS symbol position decimal value as defined in SCF FAPI.

**Return type**

np.uint16

`aerial.util.fapi.dmrs_fapi_to_sym(dmrs_symb_pos)`

Convert the DMRS symbol position decimal value to a list of DMRS symbol indices.

**Parameters**

**dmrs\_symb\_pos** (*np.uint16*) – DMRS symbol position decimal value as defined in SCF FAPI.

**Returns**

A list of DMRS symbol indices.

**Return type**

list

`aerial.util.fapi.mac_pdu_to_bit_array(mac_pdu)`

Convert MAC PDU bytes to a bit array.

**Parameters**

**mac\_pdu** (*list*) – A list of bytes, the content of the MAC PDU.

**Returns**

The same MAC PDU as a bit array, i.e. the bytes are converted to a list of bits.



**Return type**

list

```
aerial.util.fapi.bit_array_to_mac_pdu(bits)
```

Convert a bit array to MAC PDU bytes.

**Parameters**

**bits** (*list*) – A MAC PDU as a bit array.

**Returns**

A list of bytes corresponding to the above MAC PDU.

**Return type**

list

**4.4.2.2 Data storing utilities****class** aerial.util.data.PuschRecord

Implements column schema of a PUSCH dataframe row.

The *PuschRecord* includes fields collected from the data collection agent, and SCF FAPI message content for the PUSCH channels from UL\_TTI.request, RxData.indication, and CRC.indication.

**Parameters**

- ▶ **SFN** – System Frame Number. Value: 0 - 1023.
- ▶ **Slot** – Slot number. Value: 0 - 159.
- ▶ **nPDUs** – Number of PDUs that were included in the UL\_TTI.request message.
- ▶ **RachPresent** – Indicates if a RACH PDU was included in the UL\_TTI.request message.
  - ▶ 0: No RACH in this slot.
  - ▶ 1: RACH in this slot.
- ▶ **nULSCH** – Number of ULSCH PDUs that were included in the UL\_TTI.request message. Value: 0 - 255.
- ▶ **nULCCH** – Number of ULCCH PDUs that were included in the UL\_TTI.request message. Value: 0 - 255.
- ▶ **nGroup** – Number of UE Groups that were included in the UL\_TTI.request message. Value: 0 - 8.
- ▶ **PDUSize** – Size of the PDU control information (in bytes). This length value includes the 4 bytes required for the PDU type and PDU size parameters. Value: 0 - 65535.
- ▶ **nUE** – Number of UEs in this group. For SU-MIMO, one group includes one UE only. For MU-MIMO, one group includes up to 12 UEs. Value: 1 - 6, None if nGroup = 0.
- ▶ **pduIdx** – This value is an index for number of PDU identified by nPDU in the UL\_TTI.request message. Value: 0 - 255, None if nGroup = 0.
- ▶ **pduBitmap** – Bitmap indicating presence of optional PDUs.
  - ▶ Bit 0: puschData (Indicates data is expected on the PUSCH).

- ▶ Bit 1: puschUci (Indicates UCI is expected on the PUSCH).
- ▶ Bit 2: puschPtrs (Indicates PTRS included (FR2)).
- ▶ Bit 3: dftsOfdm (Indicates DFT S-OFDM transmission).
- ▶ All other bits reserved.
- ▶ **RNTI** – The RNTI used for identifying the UE when receiving the PDU. Value: 1 - 65535.
- ▶ **Handle** – An opaque handling returned in the RxData.indication and/or UCI.indication message.
- ▶ **BWPSize** – Bandwidth part size [TS38.213 sec 12]. Number of contiguous PRBs allocated to the BWP. Value: 1 - 275.
- ▶ **BWPStart** – Bandwidth part start RB index from reference CRB [TS38.213 sec 12]. Value: 0 - 274.
- ▶ **SubcarrierSpacing** – SubcarrierSpacing [TS38.211 sec 4.2]. Value: 0 - 4.
- ▶ **CyclicPrefix** – Cyclic prefix type [TS38.211 sec 4.2].
  - ▶ 0: Normal
  - ▶ 1: Extended
- ▶ **targetCodeRate** – Target coding rate [TS38.214 sec 6.1.4.1]. This is the number of information bits per 1024 coded bits expressed in 0.1 bit units.
- ▶ **qamModOrder** – QAM modulation [TS38.214 sec 6.1.4.1]. Values:
  - ▶ 2,4,6,8 if transform precoding is disabled.
  - ▶ 1,2,4,6,8 if transform precoding is enabled.
- ▶ **mcsIndex** – MCS index [TS38.214, sec 6.1.4.1], should match value sent in DCI. Value: 0 - 31.
- ▶ **mcsTable** – MCS-Table-PUSCH [TS38.214, sec 6.1.4.1]. Value:
  - ▶ 0: notqam256 [TS38.214, table 5.1.3.1-1].
  - ▶ 1: qam256 [TS38.214, table 5.1.3.1-2].
  - ▶ 2: qam64LowSE [TS38.214, table 5.1.3.1-3].
  - ▶ 3: notqam256-withTransformPrecoding [TS38.214, table 6.1.4.1-1].
  - ▶ 4: qam64LowSE-withTransformPrecoding [TS38.214, table 6.1.4.1-2].
- ▶ **TransformPrecoding** – Indicates if transform precoding is enabled or disabled [TS38.214, sec 6.1.4.1] [TS38.211 6.3.1.4].
  - ▶ 0: Enabled
  - ▶ 1: Disabled
- ▶ **dataScramblingId** – dataScramblingIdentityPdsch [TS38.211, sec 6.3.1.1]. It equals the higher-layer parameter Data-scrambling-Identity if configured and the RNTI equals the C-RNTI, otherwise L2 needs to set it to physical cell ID. Value: 0 - 65535.
- ▶ **nrOfLayers** – Number of layers [TS38.211, sec 6.3.1.3]. Value: 1 - 4.

- ▶ **ulDmrsSymbPos** – DMRS symbol positions [TS38.211, sec 6.4.1.1.3 and Tables 6.4.1.1.3-3 and 6.4.1.1.3-4]. Bitmap occupying the 14 LSBs with bit 0 corresponding to the first symbol and for each bit, value 0 indicates no DMRS and value 1 indicates DMRS.
- ▶ **dmrsConfigType** – UL DMRS config type [TS38.211, sec 6.4.1.1.3].
  - ▶ 0: type 1
  - ▶ 1: type 2
- ▶ **ulDmrsScramblingId** – UL-DMRS-Scrambling-ID [TS38.211, sec 6.4.1.1.1 ]. If provided and the PUSCH is not a msg3 PUSCH, otherwise, L2 should set this to physical cell ID. Value: 0 - 65535.
- ▶ **puschIdentity** – PUSCH-ID [TS38.211, sec 6.4.1.1.2 ]. If provided and the PUSCH is not a msg3 PUSCH, otherwise, L2 should set this to physical cell ID. Value: 0 - 1007.
- ▶ **SCID** – DMRS sequence initialization [TS38.211, sec 6.4.1.1.1]. Should match what is sent in DCI 0\_1, otherwise set to 0. Value : 0 - 1.
- ▶ **numDmrsCdmGrpsNoData** – Number of DM-RS CDM groups without data [TS38.212 sec 7.3.1.1]. Value: 1 - 3.
- ▶ **dmrsPorts** – DMRS ports. [TS38.212 7.3.1.1.2] provides description between DCI 0-1 content and DMRS ports. Bitmap occupying the 11 LSBs with bit 0 corresponding to antenna port 1000 and bit 11 corresponding to antenna port 1011 and for each bit:
  - ▶ 0: DMRS port not used.
  - ▶ 1: DMRS port used.
- ▶ **resourceAlloc** – Resource Allocation Type [TS38.214, sec 6.1.2.2].
  - ▶ 0: Type 0.
  - ▶ 1: Type 1.
- ▶ **rbBitmap** – For resource allocation type 0. [TS38.214, sec 6.1.2.2.1] [TS 38.212, 7.3.1.1.2] bitmap of RBs, 273 rounded up to multiple of 32. This bitmap is in units of VRBs. LSB of byte 0 of the bitmap represents the first RB of the BWP. Each element is of type *numpy.uint8*.
- ▶ **rbStart** – For resource allocation type 1. [TS38.214, sec 6.1.2.2.2]. The starting resource block within the BWP for this PUSCH. Value: 0 - 274.
- ▶ **rbSize** – For resource allocation type 1. [TS38.214, sec 6.1.2.2.2]. The number of resource block within for this PUSCH. Value: 1 - 275.
- ▶ **VRBtoPRBMapping** – VRB to PRB mapping [TS38.211, sec 6.3.1.7].
  - ▶ 0: Non-interleaved.
  - ▶ 1: Interleaved.
- ▶ **FrequencyHopping** – For resource allocation type 1, indicates if frequency hopping is enabled. [TS38.212, sec 7.3.1.1] [TS38.214, sec 6.3].
  - ▶ 0: Disabled.
  - ▶ 1: Enabled.

- ▶ **txDirectCurrentLocation** – The uplink Tx Direct Current location for the carrier. Only values in the value range of this field between 0 and 3299, which indicate the subcarrier index within the carrier corresponding to the numerology of the corresponding uplink BWP and value 3300, which indicates “Outside the carrier” and value 3301, which indicates “Undetermined position within the carrier” are used. [TS38.331, UplinkTxDirectCurrentBWP IE]. Value: 0 - 4095.
- ▶ **uplinkFrequencyShift7p5khz** – Indicates whether there is 7.5 kHz shift or not. [TS38.331, UplinkTxDirectCurrentBWP IE].
  - ▶ 0: False.
  - ▶ 1: True.
- ▶ **StartSymbolIndex** – Start symbol index of PUSCH mapping from the start of the slot, S. [TS38.214, Table 6.1.2.1-1]. Value: 0 - 13.
- ▶ **NrOfSymbols** – PUSCH duration in symbols, L. [TS38.214, Table 6.1.2.1-1]. Value: 1 - 14.
- ▶ **puschData** – See SCF FAPI 10.02, Table 3-47. dict{'cbPresentAndPosition': array([], dtype=int32), 'harqProcessID': np.uint8, 'newDataIndicator': np.uint8, 'numCb': np.uint8, 'rvIndex': np.uint8, 'TBSIZE': np.uint32}
- ▶ **puschUci** – See SCF FAPI 10.02, Table 3-48.
- ▶ **puschPtrs** – See SCF FAPI 10.02, Table 3-49.
- ▶ **dftsOfdm** – See SCF FAPI 10.02, Table 3-50.
- ▶ **Beamforming** – See SCF FAPI 10.02, Table 3-53.
- ▶ **HarqID** – HARQ process ID. Value: 0 - 15.
- ▶ **PDULen** – Length of PDU in bytes. A length of 0 indicates a CRC or decoding error.
- ▶ **UL\_CQI** – SNR.
- ▶ **TimingAdvance** – Timing advance.
- ▶ **RSSI** – RSSI. See SCF FAPI 10.02 Table 3-16 for RSSI definition.
- ▶ **macPdu** – Contents of MAC PDU. Each element is of type *numpy.uint8*.
- ▶ **TbCrcStatus** – Indicates CRC result on TB data. Each element is of type *numpy.uint8*.
  - ▶ 0: Pass.
  - ▶ 1: Fail.
- ▶ **NumCb** – If CBG is not used this parameter can be set to zero. Otherwise the number of CBs in the TB. Value: 0 - 65535.
- ▶ **CbCrcStatus** – Byte-packed array where each bit indicates CRC result on CB data. Each element is of type *numpy.uint8*.
  - ▶ 0: Pass.
  - ▶ 1: Fail.
  - ▶ None if NumCb = 0.

- ▶ **rx\_iq\_data\_filename** – Filename of the received OFDM IQ data file. This file contains the complex OFDM slot data as a frequency x time x antenna numpy array.
- ▶ **user\_data\_filename** – Filename of the user data file. This file may contain for example ground truth data.
- ▶ **errInd** – Freeform error indication message.

## Notes

The PDULen field is 32 bits whereas SCF FAPI 10.02 incorrectly uses 16 bits. Using 32 bits allows MAC PDUs larger than 65535 bytes.

### **static from\_series(series)**

Create a PuschRecord from a Pandas Series entry (e.g. a DataFrame row).

#### **Parameters**

**series** (*pandas.Series*) – The input dataframe row.

#### **Returns**

The PUSCH record built from the given Pandas Series.

#### **Return type**

*PuschRecord*

### **static columns()**

Return the field names of PuschRecord.

#### **Return type**

*Tuple*

### **aerial.util.data.save\_pickle(data, filename, s3=None)**

Save the data in a pickle file either locally or on S3.

#### **Parameters**

- ▶ **data** (*np.ndarray or dict*) – The data to be saved.
- ▶ **filename** (*str*) – Full path of the file to be used.
- ▶ **s3** (*s3fs.S3FileSystem*) – The S3 filesystem to be used. Set to None for local filesystem.

#### **Return type**

None

### **aerial.util.data.load\_pickle(filename, s3=None)**

Load data from a pickle file, either a local file or on S3.

#### **Parameters**

- ▶ **filename** (*str*) – Full path of the file to be used.
- ▶ **s3** (*s3fs.S3FileSystem*) – The S3 filesystem to be used. Set to None for local filesystem.

#### **Returns**

The loaded data.

**Return type**

np.ndarray or dict

### 4.4.2.3 CUDA utilities

`aerial.util.cuda.get_cuda_stream()`

Return a CUDA stream.

**Returns**

A new CUDA stream.

**Return type**

`cudaStream_t`

`aerial.util.cuda.check_cuda_errors(result)`

Check CUDA errors.

**Parameters**

**result** (`cudaError_t`) – CUDA error value.

**Return type**

*Any*

## Copyright

©2024-2025, NVIDIA Corporation

---

# Python Module Index

## a

- aerial.phy5g.algorithms.cfo\_ta\_estimator, 592
- aerial.phy5g.algorithms.channel\_equalizer, 589
- aerial.phy5g.algorithms.channel\_estimator, 584
- aerial.phy5g.algorithms.demapper, 594
- aerial.phy5g.algorithms.noise\_intf\_estimator, 587
- aerial.phy5g.algorithms.rsrp\_estimator, 591
- aerial.phy5g.algorithms.srs\_channel\_estimator, 595
- aerial.phy5g.algorithms.trt\_engine, 594
- aerial.phy5g.chan\_models.fading\_chan, 616
- aerial.phy5g.ldpc.crc\_check, 611
- aerial.phy5g.ldpc.decoder, 605
- aerial.phy5g.ldpc.derate\_match, 609
- aerial.phy5g.ldpc.encoder, 607
- aerial.phy5g.ldpc.rate\_match, 608
- aerial.phy5g.ldpc.util, 611
- aerial.phy5g.params, 619
- aerial.phy5g.pdsch.csirs\_tx, 601
- aerial.phy5g.pdsch.pdsch\_tx, 598
- aerial.phy5g.pusch.pusch\_rx, 602
- aerial.util.cuda, 626
- aerial.util.data, 621
- aerial.util.fapi, 620





---

# Index

## Non-alphabetical

`__init__()` (`aerial.phy5g.algorithms.cfo_ta_estimator.CfoTaEstimator` method), 592

`__init__()` (`aerial.phy5g.algorithms.channel_equalizer.ChannelEqualizer` method), 589

`__init__()` (`aerial.phy5g.algorithms.channel_estimator.ChannelEstimator` method), 584

`__init__()` (`aerial.phy5g.algorithms.demapper.Demapper` method), 594

`__init__()` (`aerial.phy5g.algorithms.noise_intf_estimator.NoiseIntfEstimator` method), 587

`__init__()` (`aerial.phy5g.algorithms.rsrp_estimator.RsrpEstimator` method), 591

`__init__()` (`aerial.phy5g.algorithms.srs_channel_estimator.SrsChannelEstimator` method), 597

`__init__()` (`aerial.phy5g.algorithms.trt_engine.TrtEngine` method), 594

`__init__()` (`aerial.phy5g.algorithms.trt_engine.TrtTensorPpms` method), 594

`__init__()` (`aerial.phy5g.chan_models.fading_chan.FadingChan` method), 616

`__init__()` (`aerial.phy5g.ldpc.crc_check.CrcChecker` method), 611

`__init__()` (`aerial.phy5g.ldpc.decoder.LdpcDecoder` method), 605

`__init__()` (`aerial.phy5g.ldpc.derate_match.LdpcDerateMatch` method), 609

`__init__()` (`aerial.phy5g.ldpc.encoder.LdpcEncoder` method), 607

`__init__()` (`aerial.phy5g.ldpc.rate_match.LdpcRateMatch` method), 608

`__init__()` (`aerial.phy5g.pdsch.csirs_tx.CsiRsTx` method), 601

`__init__()` (`aerial.phy5g.pdsch.pdsch_tx.PdschTx` method), 598

`__init__()` (`aerial.phy5g.pusch.pusch_rx.PuschRx` method), 602

`add_crc_len()` (in module `aerial.phy5g.ldpc.util`), 615

`add_noise_with_snr()` (`aerial.phy5g.chan_models.fading_chan.FadingChan` method), 617

`add_noise_with_snr_numpy()` (`aerial.phy5g.chan_models.fading_chan.FadingChan` method), 617

`aerial.phy5g.algorithms.cfo_ta_estimator` module, 592

`aerial.phy5g.algorithms.channel_equalizer` module, 589

`aerial.phy5g.algorithms.channel_estimator` module, 584

`aerial.phy5g.algorithms.demapper` module, 594

`aerial.phy5g.algorithms.noise_intf_estimator` module, 587

`aerial.phy5g.algorithms.rsrp_estimator` module, 591

`aerial.phy5g.algorithms.srs_channel_estimator` module, 597

`aerial.phy5g.algorithms.trt_engine` module, 594

`aerial.phy5g.chan_models.fading_chan` module, 616

`aerial.phy5g.ldpc.crc_check` module, 611

`aerial.phy5g.ldpc.decoder` module, 605

`aerial.phy5g.ldpc.derate_match` module, 609

`aerial.phy5g.ldpc.encoder` module, 607

`aerial.phy5g.ldpc.rate_match` module, 608

`aerial.phy5g.ldpc.util` module, 611

`aerial.phy5g.params` module, 619

`aerial.phy5g.pdsch.csirs_tx` module, 601

`aerial.phy5g.pdsch.pdsch_tx` module, 598

`aerial.phy5g.pusch.pusch_rx` module, 602

`aerial.util.cuda` module, 617

## A

module, 626  
 aerial.util.data  
   module, 621  
 aerial.util.fapi  
   module, 620

**B**

bit\_array\_to\_mac\_pdu() (in module aerial.util.fapi), 621

**C**

CfoTaEstimator (class in aerial.phy5g.algorithms.cfo\_ta\_estimator), 592

ChannelEqualizer (class in aerial.phy5g.algorithms.channel\_equalizer), 589

ChannelEstimator (class in aerial.phy5g.algorithms.channel\_estimator), 584

check\_crc() (aerial.phy5g.ldpc.crc\_check.CrcChecker method), 611

check\_cuda\_errors() (in module aerial.util.cuda), 626

code\_block\_desegment() (in module aerial.phy5g.ldpc.util), 615

code\_block\_segment() (in module aerial.phy5g.ldpc.util), 614

columns() (aerial.util.data.PuschRecord static method), 625

CrcChecker (class in aerial.phy5g.ldpc.crc\_check), 611

CsiRsTx (class in aerial.phy5g.pdsch.csirs\_tx), 601

cuphy\_data\_type (aerial.phy5g.algorithms.trt\_engine.TrtTensorFlow property), 594

cuphy\_to\_tx() (aerial.phy5g.pdsch.pdsch\_tx.PdschTx class method), 600

**D**

decode() (aerial.phy5g.ldpc.decoder.LdpcDecoder method), 605

demap() (aerial.phy5g.algorithms.demapper.Demapper method), 594

Demapper (class in aerial.phy5g.algorithms.demapper), 594

derate\_match() (aerial.phy5g.ldpc.derate\_match.LdpcDeRateMatch method), 610

dmrs\_bit\_array\_to\_fapi() (in module aerial.util.fapi), 620

dmrs\_fapi\_to\_bit\_array() (in module aerial.util.fapi), 620

dmrs\_fapi\_to\_sym() (in module aerial.util.fapi), 620

dump\_channel() (aerial.phy5g.chan\_models.fading\_chan.FadingChan method), 618

**E**

encode() (aerial.phy5g.ldpc.encoder.LdpcEncoder method), 607

equalize() (aerial.phy5g.algorithms.channel\_equalizer.ChannelEqualizer method), 589

estimate() (aerial.phy5g.algorithms.cfo\_ta\_estimator.CfoTaEstimator method), 593

estimate() (aerial.phy5g.algorithms.channel\_estimator.ChannelEstimator method), 585

estimate() (aerial.phy5g.algorithms.noise\_intf\_estimator.NoiseIntfEstimator method), 587

estimate() (aerial.phy5g.algorithms.rsrp\_estimator.RsrpEstimator method), 591

estimate() (aerial.phy5g.algorithms.srs\_channel\_estimator.SrsChannelEstimator method), 597

FadingChan (class in aerial.phy5g.chan\_models.fading\_chan), 616

find\_lifting\_size() (in module aerial.phy5g.ldpc.util), 613

from\_series() (aerial.util.data.PuschRecord static method), 625

**G**

get\_base\_graph() (in module aerial.phy5g.ldpc.util), 612

get\_code\_block\_num\_info\_bits() (in module aerial.phy5g.ldpc.util), 613

get\_code\_block\_size() (in module aerial.phy5g.ldpc.util), 614

get\_crc\_len() (in module aerial.phy5g.ldpc.util), 616

get\_cuda\_stream() (in module aerial.util.cuda), 626

get\_genie\_channel() (aerial.phy5g.chan\_models.fading\_chan.FadingChan method), 618

get\_mcs() (in module aerial.phy5g.ldpc.util), 611

get\_num\_code\_blocks() (in module aerial.phy5g.ldpc.util), 614

get\_num\_info\_nodes() (in module aerial.phy5g.ldpc.util), 613

get\_soft\_bits() (aerial.phy5g.ldpc.decoder.LdpcDecoder method), 607

get\_tb\_size() (in module aerial.phy5g.ldpc.util), 612

**L**

LdpcDecoder (class in aerial.phy5g.ldpc.decoder), 605

LdpcDeRateMatch (class in aerial.phy5g.ldpc.derate\_match), 609

LdpcEncoder (class in aerial.phy5g.ldpc.encoder), 607

LdpcRateMatch (class in aerial.phy5g.ldpc.rate\_match), 608

load\_pickle() (in module aerial.util.data), 625

## M

mac\_pdu\_to\_bit\_array() (in module aerial.util.fapi), 620

max\_code\_block\_size() (in module aerial.phy5g.ldpc.util), 613

module

- aerial.phy5g.algorithms.cfo\_ta\_estimator, 592
- aerial.phy5g.algorithms.channel\_equalizer, 589
- aerial.phy5g.algorithms.channel\_estimator, 584
- aerial.phy5g.algorithms.demapper, 594
- aerial.phy5g.algorithms.noise\_intf\_estimator, 587
- aerial.phy5g.algorithms.rsrp\_estimator, 591
- aerial.phy5g.algorithms.srs\_channel\_estimator, 595
- aerial.phy5g.algorithms.trt\_engine, 594
- aerial.phy5g.chan\_models.fading\_chan, 616
- aerial.phy5g.ldpc.crc\_check, 611
- aerial.phy5g.ldpc.decoder, 605
- aerial.phy5g.ldpc.derate\_match, 609
- aerial.phy5g.ldpc.encoder, 607
- aerial.phy5g.ldpc.rate\_match, 608
- aerial.phy5g.ldpc.util, 611
- aerial.phy5g.params, 619
- aerial.phy5g.pdsch.csirs\_tx, 601
- aerial.phy5g.pdsch.pdsch\_tx, 598
- aerial.phy5g.pusch.pusch\_rx, 602
- aerial.util.cuda, 626
- aerial.util.data, 621
- aerial.util.fapi, 620

## N

NoiseIntfEstimator (class in aerial.phy5g.algorithms.noise\_intf\_estimator), 587

## P

PdschTx (class in aerial.phy5g.pdsch.pdsch\_tx), 598

PuschConfig (class in aerial.phy5g.params), 619

PuschRecord (class in aerial.util.data), 621

PuschRx (class in aerial.phy5g.pusch.pusch\_rx), 602

PuschUeConfig (class in aerial.phy5g.params), 619

## R

random\_tb() (in module aerial.phy5g.ldpc.util), 615

rate\_match() (aerial.phy5g.ldpc.rate\_match.LdpcRateMatch method), 609

RsrpEstimator (class in aerial.phy5g.algorithms.rsrp\_estimator), 591

run() (aerial.phy5g.algorithms.trt\_engine.TrtEngine method), 595

run() (aerial.phy5g.chan\_models.fading\_chan.FadingChan method), 618

run() (aerial.phy5g.pdsch.csirs\_tx.CsiRsTx method), 601

run() (aerial.phy5g.pdsch.pdsch\_tx.PdschTx method), 598

run() (aerial.phy5g.pusch.pusch\_rx.PuschRx method), 603

## S

save\_pickle() (in module aerial.util.data), 625

set\_num\_iterations() (aerial.phy5g.ldpc.decoder.LdpcDecoder method), 606

set\_profiling\_iterations() (aerial.phy5g.ldpc.encoder.LdpcEncoder method), 608

set\_profiling\_iterations() (aerial.phy5g.ldpc.rate\_match.LdpcRateMatch method), 609

set\_puncturing() (aerial.phy5g.ldpc.encoder.LdpcEncoder method), 608

set\_throughput\_mode() (aerial.phy5g.ldpc.decoder.LdpcDecoder method), 606

SrsCellPrms (class in aerial.phy5g.algorithms.srs\_channel\_estimator), 595

SrsChannelEstimator (class in aerial.phy5g.algorithms.srs\_channel\_estimator), 597

SrsReport (class in aerial.phy5g.algorithms.srs\_channel\_estimator), 596

## T

TrtEngine (class in aerial.phy5g.algorithms.trt\_engine), 594

*aerial.phy5g.algorithms.trt\_engine*), 594  
TrtTensorPrms (class in  
*aerial.phy5g.algorithms.trt\_engine*), 594

## U

UeSrsPrms (class in  
*aerial.phy5g.algorithms.srs\_channel\_estimator*),  
596