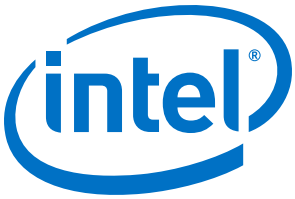


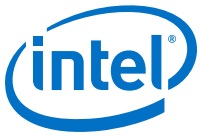
Intel Acceleration Stack Quick Start Guide for Intel® Programmable Acceleration Card with Intel® Arria® 10 GX FPGA

Updated for Intel® Acceleration Stack for Intel® Xeon® CPU with FPGAs: **1.2.1**

Subscribe
Send Feedback

UG-20166 | 2020.03.06
Latest document on the web: **PDF** | **HTML**

Contents



1. About This Document

This document serves as a high level quick start guide to help you with how to install key software packages, update the flash image, run diagnostics, and manage security. It also steps you through an example AFU in a virtualized and non-virtualized environment.

1.1. Intended Audience

The intended audience for this document is system engineers, platform architects, and hardware and software developers.

1.2. Acronym List

Acronym	Expansion	Description
AFU	Accelerator Functional Unit	Hardware Accelerator implemented in FPGA logic which offloads a computational operation for an application from the CPU to improve performance.
AF	Acceleration Function	Compiled Hardware Accelerator image implemented in FPGA logic that accelerates an application.
ASE	AFU Simulation Environment	Co-simulation environment that allows you to use the same host application and AF in a simulation environment. ASE is part of the Intel Acceleration Stack for FPGAs.
BIP	Bitstream Authentication IP	Performs integrity and authentication checks on a bitstream using ECDSA-256 and SHA2-256 and returns a pass or fail state to the TCM.
CCI-P	Core Cache Interface	CCI-P is the standard interface that enables communication with the host.
FIM	FPGA Interface Manager	The FPGA hardware containing the FPGA Interface Unit (FIU) and external interfaces for memory, networking, etc. The FPGA Interface Manager (FIM) may also be referred to as BBS (Blue-Bits, Blue BitStream) in the Acceleration Stack installation directory tree and in source code comments. The Accelerator Function (AF) interfaces with the FIM at run time.
FIU	FPGA Interface Unit	FIU is a platform interface layer that acts as a bridge between platform interfaces like PCIe* and AFU-side interfaces such as CCI-P.
FME	FPGA Management Engine	Provides the following functions: • Thermal monitoring • Performance monitoring • Partial reconfiguration • Global errors
HSSI	High-speed Serial Interface	Reference to the multi-gigabit serial transceiver I/O in the FIM and the corresponding interface to the AFU.
IOMMU	Input–Output Memory Management Unit	An IOMMU is a memory management unit that connects a Direct Memory Access (DMA) I/O bus to main memory. The IOMMU maps device-visible virtual addresses to physical addresses.

continued...

Acronym	Expansion	Description
OPAE	Open Programmable Acceleration Engine	The OPAE is a software framework for managing and accessing AFs.
PR	Partial Reconfiguration	The ability to dynamically reconfigure a portion of an FPGA while the remaining FPGA design continues to function. The FPGA includes PR region. You can reprogram these regions at run time to implement different AFUs as system requirements dictate.
TCM	Trusted Configuration Manager	Receives all updates to the AFU/PR region, FIM, and BMC, then authenticates them using the BIP. Authenticated bitstreams are loaded to their appropriate destination.

1.3. Acceleration Glossary

Table 1. Acceleration Stack for Intel® Xeon® CPU with FPGAs Glossary

Term	Abbreviation	Description
Intel® Acceleration Stack for Intel Xeon® CPU with FPGAs	Acceleration Stack	A collection of software, firmware, and tools that provides performance-optimized connectivity between an Intel FPGA and an Intel Xeon processor.
Intel Programmable Acceleration Card with Intel Arria® 10 GX FPGA	Intel PAC with Intel Arria 10 GX FPGA	PCIe FPGA accelerator card. Contains an FPGA Interface Manager (FIM) that pairs with an Intel Xeon processor over the PCIe bus.

2. Introduction

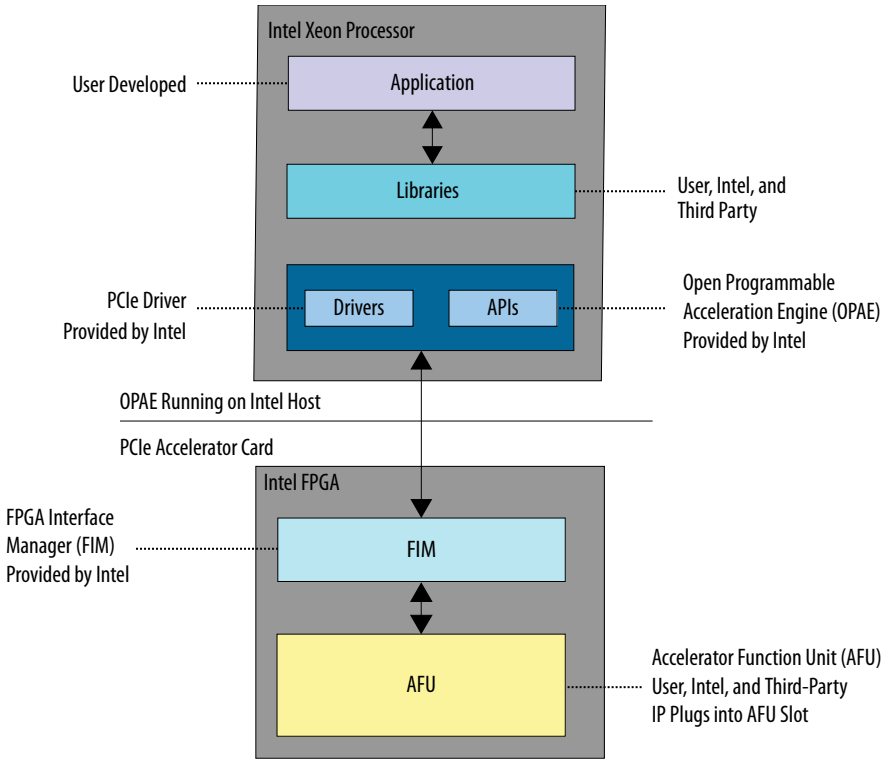
This guide provides a brief introduction to the Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA, abbreviated as Intel PAC with Intel Arria 10 GX FPGA in this document. This guide provides the instructions to:

- Install the OPAE software
- Upgrade the Intel PAC with Intel Arria 10 GX FPGA FIM and BMC firmware
- Activate the security features on the Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA platform
- Load and run a loopback test and the `hello_fpga` basic design example in both non-virtualized and virtualized environments

The Acceleration Stack is a collection of software, firmware, and tools that allows both software and RTL developers to take advantage of the power of Intel FPGAs. By offloading computationally intensive tasks to the FPGA, the acceleration platform frees the Intel Xeon processor for other critical processing tasks.

The Intel PAC with Intel Arria 10 GX FPGA, an accelerator card, connects to the Intel Xeon processor through the PCIe interface on the motherboard.

Figure 1. Overview of the Intel PAC with Intel Arria 10 GX FPGA Platform Hardware and Software



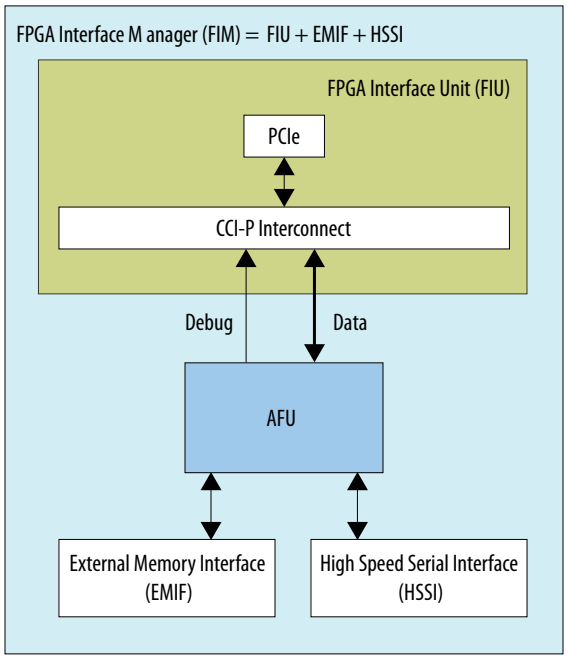
To take advantage of the flexibility of the FPGA, you can reconfigure a predefined, partial reconfiguration (PR) region of the Intel Arria 10 GX FPGA at run time. You can design multiple AFUs to swap in and out of this PR region. The Open Programmable Acceleration Engine (OPAE) software running on the Intel Xeon processor handles all the user-facing details of the reconfiguration process.

Security and reconfiguration are some of the many utilities that the OPAE provides. The OPAE also provides libraries, drivers, and sample programs useful for AFU development.

To facilitate dynamically loading AFUs, the Acceleration Stack includes the following two components:

- The FIM provides a framework to load AFUs on the Intel PAC with Intel Arria 10 GX FPGA. The FIM also includes the PR regions for the AFUs and the IP necessary to authenticate them. The FIM contains the FPGA logic to support the accelerators, including the PCIe IP core, the CCI-P fabric, the on-board DDR memory interfaces, and the FPGA Management Engine (FME). At power up, an on-board FPGA configuration flash containing the FIM bitstream image configures the FIM. The PR regions are empty until the OPAE software programs the AFU images. The FIM framework is fixed. The current release of the FIM for the Intel PAC with Intel Arria 10 GX FPGA supports a single PR region.
- The Acceleration Stack supports creation of AFU images with either RTL or OpenCL* design flows. An AFU image includes the AFU PR region bitstream and metadata that provides OPAE information on AFU characteristics and operational parameters. The current release supports dynamically swapping a single AFU image in a single PR region per installed Intel FPGA PAC.

Figure 2. Intel Arria 10 with a Single AFU PR Region



The AFU connects to the Intel Xeon processor through the CCI-P interface and then the PCIe link. The Intel PAC with Intel Arria 10 GX FPGA platform uses a simplified version of the CCI-P interface. For more information about the CCI-P interface, refer to the *Intel Acceleration Stack for Intel Xeon CPU with FPGAs Core Cache Interface (CCI-P) Reference Manual*.

The AFU has access to two banks of private DDR4-SDRAM memory, totaling 8 GB. Each DDR4 memory bank interface has a standard Avalon® Memory-Mapped (Avalon-MM) interface. For more information about this interface, refer to the *Avalon-MM Interface Specifications*.

The Intel PAC with Intel Arria 10 GX FPGA supports a single QSFP+ network port.

For more information about security, refer to *Security User Guide: Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA*.

Related Information

- Documentation Available for the Intel Acceleration Stack for Intel Xeon CPU with FPGAs 1.2.1 Release on page 51
- 10 Gbps Ethernet Accelerator Functional Unit (AFU) Design Example User Guide
- 40 Gbps Ethernet Accelerator Functional Unit (AFU) Design Example User Guide
- Intel Ethernet QSFP+ Cables Product Brief
- Avalon-MM Interfaces
  For more information about the Avalon-MM protocol, including extensive timing diagrams.
- Acceleration Stack for Intel Xeon CPU with FPGAs Core Cache Interface (CCI-P) Reference Manual
  CCI-P is a host interface bus for an AFU.
- Intel Programmable Acceleration Card (PAC) with Intel Arria 10 GX FPGA Datasheet
- Resources for Intel PAC with Intel Arria 10 GX FPGA
  For a comprehensive list of documentation available for the Intel Acceleration Task.
- Security User Guide: Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA

3. Getting Started

3.1. Intel Acceleration Stack Hardware Features

The Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA supports the following features:

- Two banks of 4 gigabyte (GB) private memory for a total memory of 8 GB
- One, PCI Express* Gen3 x8 link over a x16 card mechanical card edge
- 4 x 10 Gbps Ethernet (10GbE) or 1 x 40 Gbps Ethernet (40GbE)
- Remote In-System Debug
- Temperature monitoring through PLDM over MCTP or from host software over PCIe
- Secure updating of the PR region, FIM, and BMC through the TCM and BIP in the Intel PAC with Intel Arria 10 GX FPGA

3.2. System Requirements

- Validated servers:
  - For the most current list of validated servers, refer to the Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA Qualified Servers and Ordering Info web page.
- Validated Linux* releases:
  - Red Hat* Enterprise Linux* (RHEL) 7.6, kernel version 3.10.0-957
  - Ubuntu* 18.04, kernel 4.15
- PACSign tool requires Python 3.0
- Open Programmable Acceleration Engine (OPAE) tools require Python 2.7

You can use the same server for all development, including the following activities:

- Developing software
- Running sample programs and diagnostics
- Creating and simulating AFUs
- Generating the loadable AFU images

Note:

- The system you use to compile the hardware design must have at least 48 GB of free memory.
- Known good software and hardware combination configurations can be found at the Intel FPGA Acceleration Hub Software Download Intel web page.
- Software dependencies are downloaded through the internet by the installer.

Table 2. Useful Linux Commands

The following Linux commands provide information about your system.

Command	Description
`sudo dmidecode -t bios`	BIOS information, including revision
`cat /proc/cpuinfo`	CPU information
`cat /etc/redhat-release`	RHEL version information
`cat /proc/version`	Linux kernel version
`cat /etc/lsb-release` or `cat /etc/os-release`	Ubuntu version information

3.3. Installing the Intel PAC with Intel Arria 10 GX FPGA Card In the Host Machine

Follow these instructions to install the Intel PAC with Intel Arria 10 GX FPGA card.

Precautions for Hardware Installation

You must open the server chassis to install the Intel PAC with Intel Arria 10 GX FPGA card. Follow all safety precautions and the electrostatic discharge (ESD) guidelines provided to avoid damaging the server or the Intel PAC with Intel Arria 10 GX FPGA.

Warning: To avoid electric shock, power down your server and unplug it from the power outlet before opening the server chassis.

ESD Guidelines

Electronics components on the Intel PAC with Intel Arria 10 GX FPGA and server are sensitive to ESD. To avoid damaging them, please follow these ESD prevention guidelines:

- Wear a grounded ESD strap during the Intel PAC with Intel Arria 10 GX FPGA installation.
- Leave the Intel PAC with Intel Arria 10 GX FPGA in its ESD-safe packaging until you are ready to install the card.
- During installation, handle the Intel PAC with Intel Arria 10 GX FPGA only by the edge of the board.
- Never touch any exposed circuitry, edge connectors, or printed circuits on the Intel PAC with Intel Arria 10 GX FPGA or server.
- Do not put the Intel PAC with Intel Arria 10 GX FPGA on any metal surface during installation.
- If you must put the Intel PAC with Intel Arria 10 GX FPGA down, put the card in the ESD-safe packaging.

1. Open your server chassis.
2. Identify an available PCIe Gen3 x16 mechanical slot with enough clearance to house the Intel PAC with Intel Arria 10 GX FPGA.
3. Remove any I/O panel covers for the slot you are using.

4. Install the Intel PAC with Intel Arria 10 GX FPGA in the PCIe slot by inserting the PCIe x16 edge connector and ensuring the card retention hook is properly engaged.
5. If supported by your server chassis, use the two screws provided to secure the I/O panel bracket to the server chassis.
6. Reinstall the chassis cover.
7. Enable the following options in the BIOS if you plan on using virtualization/ containers:
   - Intel VT-x (Intel Virtualization Technology for IA-32 and Intel 64 Processors)
   - Intel VT-d (Intel Virtualization Technology for Directed I/O)

3.4. Installing the Intel Acceleration Stack

You have the option of downloading the Acceleration Stack for Runtime or the Acceleration Stack for Development. If you are a software developer who develops and integrates your host application with accelerator functions, download the Acceleration Stack for Runtime. If you are an accelerator function developer who creates, debugs and simulates accelerator functions, download the Acceleration Stack for Development. Security functionality is equal across both.

The following table describes each Acceleration Stack package.

Table 3. Intel Acceleration Stack Download Options

Components	Intel Acceleration Stack for Runtime	Intel Acceleration Stack for Development
Purpose	Software development of runtime host application	Develop the hardware Accelerator function using RTL or OpenCL BSP with Intel Quartus® Prime Pro Edition and Acceleration Stack.
Intel Acceleration Stack Version	Intel Acceleration Stack Version 1.2.1	Intel Acceleration Stack Version 1.2.1
OPAE Software Development Kit (SDK) version for 1.1.2	OPAE SDK version 1.1.2-2	OPAE SDK version 1.1.2-2
Intel Quartus Prime Pro Edition Software	Not included or required	Intel Quartus Prime Pro Edition software version 19.2.0 including SR-IOV license
OpenCL Software	Intel FPGA Runtime Environment for OpenCL 19.4.0.67	Intel FPGA SDK for OpenCL 19.4.0.67
Download Size	328 MB	~9.25 GB
Intel Acceleration Stack download	You can access the download (a10_gx_pac_ias_1_2_1_pv_rte_installer.tar.gz) by clicking here: Acceleration Stack for Runtime (RTE).	You can access the download (a10_gx_pac_ias_1_2_1_pv_dev_installer.tar.gz) by clicking here: Acceleration Stack for Development (DEV).
Board Management Controller (BMC) version	Firmware version 26895	Firmware version 26895

For more information about where to find the Acceleration Stack package, contact your Intel support representative.

3.4.1. Installing the Intel Acceleration Stack Runtime Package on the Host Machine

1. Extract the runtime archive file:

```
tar xvf *rte_installer.tar.gz
```

2. Change to the installation directory.

```
cd *rte_installer
```

3. This step only applies to RHEL 7.6 (skip this step if you are using Ubuntu 18.04). Install extra packages for Enterprise Linux (EPEL):

```
sudo yum install https://dl.fedoraproject.org/pub/epel/\
epel-release-latest-7.noarch.rpm
```

```
sudo subscription-manager repos --enable "rhel-*-optional-rpms"\
--enable "rhel-*-extras-rpms"
```

4. Run setup.sh.

```
./setup.sh
```

5. If you receive a prompt with the message: *Intel Acceleration Stack Runtime Package is only supported on RHEL 7.6.* kernel 3.10.* or Ubuntu 18.04.* kernel 4.15.**, then you are currently using an unsupported operating system/kernel combination.

   Next, a prompt appears with the following question: *Do you want to continue to install the software?* Answering **Yes** allows the setup script to attempt to install OPAE onto this unsupported operating and kernel version.

   For complete operating system support information, refer to section System Requirements *on page 10*.

6. Next, a prompt appears with the following question: *Do you wish to install the OPAE?*

Option	Description
Answer Yes	If you require the use of PACSign to sign bitstreams prior to loading on a Intel FPGA PAC, admin and network access is required.
Answer No	If you do not have admin and network access. After the installation, follow the manual steps listed in the Installing the OPAE Software Package section.

7. Next, a prompt appears with the following question: *Do you wish to install OPAE PACsign package?*

Option	Description
Answer Yes	If you have admin and network access.
Answer No	If you do not have admin and network access. After the installation, follow the manual steps listed in the Installing the OPAE PACSign section.

8. Accept the license.

9. When you receive an installation directory prompt for the Intel PAC with Intel Arria 10 GX FPGA release package, you can specify an install directory. Otherwise, the installer uses the default directory at `/home/<username>/intelrtestack`.
10. When you receive an installation directory prompt for the Intel FPGA RTE OpenCL package, you can specify an install directory. Otherwise, the installer uses the default directory at `/opt/opencl_rte`.

    Next, a prompt appears stating the following: ***Warning**: Elevated permissions are required in order to write to directory `/opt`. Do you want to proceed?* Answer **Yes**.

    Note: If there is not enough available storage under `/opt` for the OpenCL RTE, the installer fails and halts execution.
11. Source the initialization script after installation completes to set the required environment variables.

```
source /home/<username>/intelrtestack/init_env.sh
```

    Note: To avoid having to setup the environment variables after every reboot, you can save this command to your shell initialization script.

Related Information

Installing the OPAE Software Package on page 17

3.4.2. Installing the Intel Acceleration Stack Development Package on the Host Machine

1. Extract the runtime archive file:

```
tar xvf *dev_installer.tar.gz
```

2. Change to the installation directory.

```
cd *dev_installer
```

3. This step only applies to RHEL 7.6 (skip this step if you are using Ubuntu 18.04). Install extra packages for Enterprise Linux(EPEL):

```
sudo yum install https://dl.fedoraproject.org/pub/epel/epel-release\
-latest-7.noarch.rpm
```

```
sudo subscription-manager repos --enable "rhel-*-optional-rpms"\
--enable "rhel-*-extras-rpms"
```

4. Run setup.sh.

```
./setup.sh
```

5. If you receive a prompt with the message: *Intel Acceleration Stack Runtime Package is only supported on RHEL 7.6.* kernel 3.10.* or Ubuntu 18.04.* kernel 4.15.**, then you are currently using an unsupported operating system/kernel combination.

   Next, a prompt appears with the following question: *Do you want to continue to install the software?* Answering **Yes** allows the setup script to attempt to install OPAE onto this unsupported operating and kernel version.

   For complete operating system support information, refer to System Requirements *on page 10*.

6. Next, a prompt appears with the following question: *Do you wish to install the OPAE?*

Option	Description
Select Yes	If you have admin and network access.
Select No	If you do not have admin and network access. After the installation, follow the manual steps listed in the Installing the OPAE Software Package section.

7. Next, a prompt appears with the following question: *Do you wish to install OPAE PACsign package?*

Option	Description
Answer Yes	If you require the use of PACSign to sign bitstreams prior to loading on an Intel FPGA PAC. Admin and network access are required.
Answer No	If you do not have admin and network access. After the installation, follow the manual steps listed in the Installing the OPAE PACSign section.

8. Accept the license.
9. When you receive an installation directory prompt for the Intel PAC with Intel Arria 10 GX FPGA release package, you can specify an install directory. Otherwise, the installer uses the default directory at `/home/<username>/inteldevstack`.
10. When you receive an installation directory prompt for the Intel Quartus Prime Pro Edition release package, you can specify an install directory. Otherwise, the installer uses the default directory at `/opt/intelFPGA_pro/quartus_19.2.0b57`.

    Next, a prompt appears with the following: ***Warning**: Elevated permissions are required to write to the `/opt` directory. Do you want to proceed?* Answer **Yes**.

    Note: If there is not enough available storage under `/opt` for the OpenCL RTE, the installer fails and halts execution.
11. Source the initialization script after installation completes to set the required environment variables.

```
source /home/<username>/inteldevstack/init_env.sh
```

    Note: To avoid having to setup the environment variables after every reboot, you can save this command to your shell initialization script.

3.4.3. Understanding the Extracted Intel PAC with Intel Arria 10 GX FPGA Release Package

The `init_env.sh` script defines the `OPAE_PLATFORM_ROOT` environment variable. `OPAE_PLATFORM_ROOT` points to the extracted `a10_gx_pac_ias*` release directory. Depending on your previous choice, `a10_gx_pac_ias*` is available in one of the following directories:

- If you installed the Acceleration Stack for Runtime: `/home/<username>/intelrtestack/*`
- If you installed the Acceleration Stack for Development: `/home/<username>/inteldevstack/*`
- If you chose a custom installation directory, `a10_gx_pac_ias*`: `/<custom_install_directory>/*`

Note: If installation fails, please rerun the installer and select **No** when prompted with: *Do you wish to install the OPAE?* After installation completes, follow the manual steps to install OPAE as detailed in the Installing the OPAE Software Package on page 17 and Troubleshooting OPAE Installation on RHEL on page 49 sections.

Figure 3. Intel PAC with Intel Arria 10 GX FPGA 1.2.1 Directory Structure

This figure shows extracted directory structure and some of the most important files:

Related Information

- Troubleshooting OPAE Installation on RHEL on page 49
- Installing the OPAE Software Package on page 17

4. Installing the OPAE Software Package

The OPAE is a software framework for managing and accessing programmable accelerators (FPGAs).

Note: Python 3 is required to install the OPAE PACSign tool.

Sections *RHEL 7.6 : Installing the OPAE Framework from Prebuilt Binaries (RPM)*, *Ubuntu: Installing the OPAE Framework from Prebuilt Binaries (deb)*, and *(Optional) Building and Installing the OPAE Software from Source Code* can be **skipped** if you have already installed OPAE by answering **Yes** when prompted by the script `setup.sh` (Acceleration Stack installer package).

After completing the OPAE framework installation, the following software and libraries are available:

- The Intel FPGA Driver
- The OPAE source at: `$OPAE_PLATFORM_ROOT/sw/opae*`
- The OPAE software development kit (SDK)

4.1. RHEL 7.6: Installing the OPAE Framework from Prebuilt Binaries (RPM)

Before you can install and build the OPAE software, you must install the required packages by running the following commands:

```
sudo yum install https://dl.fedoraproject.org/pub/epel/\
epel-release-latest-7.noarch.rpm
```

```
sudo yum install gcc gcc-c++ cmake make autoconf automake\
libxml2 libxml2-devel json-c-devel boost ncurses ncurses-devel\
ncurses-libs boost-devel libuuid libuuid-devel python2-jsonschema\
doxygen rsync hwloc-devel libpng12 python2-pip tbb-devel
```

```
sudo pip install intelhex
```

Note: These commands only install the missing packages.

Complete the following steps to install the OPAE framework:

1. Remove any previous version of the OPAE framework

   ```
   sudo yum remove opae*
   ```

2. Change to the OPAE installation software directory:

   ```
   cd $OPAE_PLATFORM_ROOT/sw
   ```

3. Install the latest OPAE framework and driver:

```
sudo yum install opae*.rpm
```

4. Update dynamic linker run-time bindings:

```
sudo ldconfig
```

5. Check the Linux kernel installation:

```
lsmod | grep fpga
```

Sample output:

```
intel_fpga_pac_hssi    24389  0
intel_fpga_fme         87460  0
intel_fpga_afu         36165  0
ifpga_sec_mgr          13757  1 intel_fpga_fme
fpga_mgr_mod           14812  1 intel_fpga_fme
intel_fpga_pci         26722  2 intel_fpga_afu,intel_fpga_fme
```

After completing the OPAE installation, the binaries and libraries are available in the following directories:

Directory	OPAE Driver Package	Content
/usr/bin	opae-tools* opae-tools-extra*	OPAE tool binaries. For a full listing of the tools and their purpose, refer to the *OPAE FPGA TOOLS* section on the **Open Programmable Acceleration Engine** web page.
/usr/include	opae-devel*	The header files required for linking host applications.
/usr/lib64	opae-libs* opae-ase*	The OPAE shared object libraries.

6. Verify rpm installation:

```
rpm -qa | grep opae
```

Sample output:

```
opae.admin-1.0.2-3.noarch
opae-tools-extra-1.1.2-2.x86_64
opae.pac_sign-1.0.3-1.x86_64
opae-devel-1.1.2-2.x86_64
opae-one-time-update-a10-gx-pac-1.2.1-11.noarch
opae-intel-fpga-driver-2.0.3-2.x86_64
opae-super-rsu-a10-gx-pac-1.2.1-12.noarch
opae-ase-1.1.2-2.x86_64
opae-tools-1.1.2-2.x86_64
opae-libs-1.1.2-2.x86_64
```

For more information, refer to the *Troubleshooting OPAE Installation on RHEL* section.

Related Information

Troubleshooting OPAE Installation on RHEL on page 49

4.2. Ubuntu: Installing the OPAE Framework from Prebuilt Binaries (deb)

Before you can install and build the OPAE software, you must install the required packages by running the following two commands:

```
sudo apt-get -f cmake install dkms libjson-c3 uuid-dev libjson-c-dev\
libhwloc-dev python-pip libjson-c-dev libhwloc-dev linux-headers-$(uname -r)
libtbb-dev
```

```
sudo pip install intelhex
```

Complete the following steps to install the OPAE framework:

1. Remove any previous version of the OPAE framework.

   ```
   sudo apt-get remove opae*
   sudo apt-get remove python-opae.admin
   sudo apt-get remove python3-opae.pac-sign
   ```

2. Change to the OPAE installation software directory.

   ```
   cd $OPAE_PLATFORM_ROOT/sw
   ```

3. Install the latest OPAE driver and framework.

   ```
   sudo dpkg -i *.deb
   ```

   Note: If dpkg fails due to missing dependencies, please run the following command to resolve:

   ```
   sudo apt-get install -f
   ```

4. Check the Linux kernel installation.

   ```
   lsmod | grep fpga
   ```

   Sample Output:

   ```
   intel_fpga_pac_iopll    16384  0
   intel_fpga_pac_hssi    24576  0
   intel_fpga_fme         90112  0
   intel_fpga_afu         36864  0
   intel_fpga_pci         28672  2 intel_fpga_fme,intel_fpga_afu
   fpga_mgr_mod           16384  1 intel_fpga_fme
   ifpga_sec_mgr          16384  2 intel_max10,intel_fpga_fme
   ```

5. After completing the OPAE installation, the binaries and libraries are available in the following directories:

Directory	OPAE Driver Package	Content
/usr/bin	opae-tools* opae-tools-extra*	OPAE tool binaries. For a full listing of the tools and their purpose, refer to the *OPAE FPGA TOOLS* section on the **Open Programmable Acceleration Engine** web page.
/usr/include	opae-devel*	The header files required for linking host applications.
/usr/lib	opae-libs* opae-ase*	The OPAE shared object libraries.

6. Verify deb packages installation:

   ```
   dpkg -l | grep opae
   ```

Sample Output:

```
opae-a10-gx-pac-fpgaotsu-base             1.2.1             all
Intel PAC one-time-update
opae-a10-gx-pac-super-rsu-base            1.2.1             all
Intel PAC super-rsu
opae-ase                                  1.1.2             amd64
OPAE AFU Simulation Environment
opae-devel                                1.1.2             amd64
OPAE headers, sample source, and documentation
opae-intel-fpga-driver                    2.0.3             all
DKMS-enabled Intel FPGA driver source code.
opae-libs                                 1.1.2             amd64
OPAE runtime
opae-samplesrasescriptsc                  1.1.2             amd64
Open Programmable Acceleration Engine
opae-tools                                1.1.2             amd64
OPAE base tool binaries
opae-tools-extra                          1.1.2             amd64
OPAE extra tool binaries
python-opae.admin                         1.0.2             all
OPAE Administration
python3-opae.pac-sign                     1.0.3-1           amd64
opae.pac_sign provides Python classes for interfacing withOP
```

4.3. (Optional) Building and Installing the OPAE Software from Source Code

1. Complete the following steps to install Intel FPGA Driver:
   a. Remove any previous version:

      RHEL:

      ```
      sudo yum remove opae*
      ```

      Ubuntu:

      ```
      sudo apt-get remove opae*
      sudo apt-get remove python-opae.admin
      sudo apt-get remove python3-opae.pac-sign
      ```

   b. Install the Extra Packages for Enterprise Linux (EPEL):

      RHEL:

      ```
      sudo yum install https://dl.fedoraproject.org/pub/epel/\
      epel-release-latest-7.noarch.rpm
      ```

   c. Change to the OPAE installation software directory:

      ```
      cd $OPAE_PLATFORM_ROOT/sw
      ```

   d. Install the driver:

      RHEL:

      ```
      sudo yum install opae-intel-fpga*.rpm
      ```

      Ubuntu:

      ```
      sudo dpkg -i opae-intel-fpga-driver_*.deb
      ```

2. Build and install the OPAE SDK from source:

a. Change to the OPAE software directory and extract the .tar file:

```
cd $OPAE_PLATFORM_ROOT/sw
tar xf opae*.tar.gz
```

b. Complete the following steps to build the OPAE software:

```
cd opae*
mkdir build && cd build
cmake .. -DBUILD_ASE=OFF -DCMAKE_INSTALL_PREFIX=<path to install
directory> -DCMAKE_BUILD_TYPE=Release
```

For example:

```
cmake .. -DBUILD_ASE=ON -DCMAKE_INSTALL_PREFIX=/home/john/\
opaeinstall -DCMAKE_BUILD_TYPE=Release
```

Note: You may get an error because the `cmake` command cannot find the git repository. You can safely ignore this error message. You do not need the git repository to successfully build the OPAE software.

c. Run the following command to build the executables and libraries:

```
make install
```

Note: By default, if you choose the RPM installation flow, the binaries, libraries and include files are under `/usr/`. If you build and install the OPAE from the source flow, the binaries, libraries, and include files are under `<path to install directory>`.

d. Set the appropriate environment variable to ensure tools, libraries, and include files are in your search path. To avoid rerunning this command whenever you restart or open a new terminal, add these directory environment variables to your shell configuration file, `/etc/bashrc`.

```
export PATH=<path to OPAE install directory>/bin:$PATH
```

```
export C_INCLUDE_PATH=<path to OPAE install directory>/include:\
$C_INCLUDE_PATH
```

To check for static libraries use the following paths:

- RHEL:

```
export LIBRARY_PATH=<path to OPAE install directory>\
/lib64:$LIBRARY_PATH
```

- Ubuntu:

```
export LIBRARY_PATH=<path to OPAE install directory>\
/lib:$LIBRARY_PATH
```

To check for shared libraries use the following paths:

- RHEL:

```
export LD_LIBRARY_PATH=<path to OPAE install directory>/\
lib64:$LD_LIBRARY_PATH
```

- Ubuntu:

```
export LD_LIBRARY_PATH=<path to OPAE install directory>/\
lib:$LD_LIBRARY_PATH
```

4.4. Installing the OPAE PACSign

If you have performed the steps from the Ubuntu: Installing the OPAE framework from prebuilt binaries (deb) section or the (Optional) Building and Installing the OPAE Software from Source Code, then this step is redundant.

Note: Python 3 is required to install the OPAE PACSign tool.

Install the OPAE PACSign package:

```
$ cd $OPAE_PLATFORM_ROOT/sw
```

On RHEL:

```
$ sudo yum install opae.pac_sign-1.0.3-1.x86_64.rpm
```

On Ubuntu:

```
$ sudo dpkg -i python3-opae.pac-sign_1.0.3-1_amd64.deb
```

5. Identifying the Flash Image and BMC Firmware

Each Acceleration Stack Release requires a different version of the FIM. Run the `fpgainfo` tool to **identify the FIM (PR Interface ID) and BMC firmware** currently loaded.

```
sudo fpgainfo fme
```

Sample Output (after updating to 1.2.1 FIM):

```
Board Management Controller, microcontroller FW version 26895
Last Power Down Cause: POK_CORE
Last Reset Cause: None
//****** FME ******//
Object Id                        : 0xED00000
PCIe s:b:d:f                     : 0000:D8:00:0
Device Id                        : 0x09C4
Socket Id                        : 0x00
Ports Num                        : 01
Bitstream Id                     : 0x124000200000338
Bitstream Version                : 1.2.4
Pr Interface Id                  : 38d782e3-b612-5343-b934-2433e348ac4c
Boot Page                        : user
```

Table 4. Correspondence Between Acceleration Stack, FIM, and OPAE Versions

Note: Intel recommends porting AFUs and workloads to the Intel Acceleration Stack v1.2.1. You must recompile and validate when upgrading to the new release. The Intel Acceleration Stack v1.2.1 cannot be downgraded to previous versions.

Acceleration Stack Version	FIM Version (PR Interface ID)	OPAE Version	BMC Firmware Version
1.2.1	38d782e3-b612-5343-b934-2433e348ac4c	1.1.2-2	26895 (bootloader version 26895)
1.2	69528db6-eb31-577a-8c36-68f9faa081f6	1.1.2-1	26889 (bootloader version 26879)
1.1	9926ab6d-6c92-5a68-aabc-a7d84c545738	1.0.2	26822
1.0	ce489693-98f0-5f33-946d-560708be108a	0.13.1	26815

If your FIM and BMC firmware version correspond to the most recent version for 1.2.1, then proceed to the next section: *Running FPGA Diagnostics*. If your FIM version is out of date, go to *Appendix A: Updating the FIM and BMC Firmware* for further instructions.

Table 5. Acceleration Stack Security Package Versions for 1.2.1

RHEL	Ubuntu
One-Time-Secure-Update RPM: 1.2.1-13	One-Time-Secure-Update: 1.2.1
Super-RSU RPM Version: 1.2.1-13	Super-RSU Base: 1.2.1
`opae.admin` Version: 1.0.2-3	`python-opae.admin`: 1.0.2
`opae-intel-fpga-driver-2.0.3-3`	`opae-intel-fpga-driver`
`opae.pac_sign`: 1.0.3-1	`python3-opae.pac-sign`: 1.0.3-1

Related Information

Updating the FIM and BMC Firmware on page 38

6. Running FPGA Diagnostics

This section presents instructions on how to run the FPGA diagnostics by using the `fpgabist` utility. The current AFUs accepted are `nlb_mode_3` and `dma_afu`, running `fpgadiag` and `fpga_dma_test` tests, respectively.

Note: If a flash is programmed with a root entry hash, you must ensure that the AFUs are signed with an appropriate root key and code signing key before running the FPGA diagnostics. For more information on signing, refer to *Security User Guide: Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA*.

1. Configure the number of system hugepages the FPGA `fpgadiag` utility requires:

```
sudo sh -c "echo 20 > /sys/kernel/mm/hugepages/hugepages-\
2048kB/nr_hugepages"
```

2. Configure and run diagnostics with the NLB_3 AFU image.

```
sudo fpgabist $OPAE_PLATFORM_ROOT/hw/samples/nlb_mode_3/bin/\
nlb_mode_3_unsigned.gbs
```

Sample partial output:

```
Cachelines Read_Count Write_Count Cache_Rd_Hit Cache_Wr_Hit Cache_Rd_Miss
Cache_Wr_Miss   Eviction 'Clocks(@400 MHz)'   Rd_Bandwidth   Wr_Bandwidth
1024  480797340   488815296            0           0
0             0          0       1000021563      6.234 GB/s    6.256 GB/s

VH0_Rd_Count VH0_Wr_Count VH1_Rd_Count VH1_Wr_Count VL0_Rd_Count
VL0_Wr_Count480797340        488815297                        0
0            0            0

Built-in Self-Test Completed.
```

3. Configure and run diagnostics with the DMA AFU image.

```
sudo fpgabist $OPAE_PLATFORM_ROOT/hw/samples/dma_afu/bin/
dma_afu_unsigned.gbs
```

Sample partial output:

```
Running test in HW mode
Buffer Verification Success!
Buffer Verification Success!
Running DDR sweep test
Allocated test buffer
Fill test buffer
DDR Sweep Host to FPGA
Measured bandwidth = 6616.881910 Megabytes/sec
Clear buffer
DDR Sweep FPGA to Host
Measured bandwidth = 6932.201347 Megabytes/sec
Verifying buffer.
Buffer Verification Success!
Finished Executing DMA Tests
```

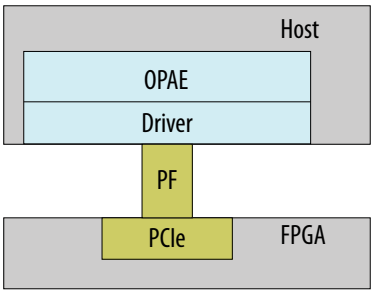
Related Information

- OPAE FPGA Tools - fpgabist
- Security User Guide: Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA

7. Running the OPAE in a Non-Virtualized Environment

This section shows OPAE examples running directly on the Bare Metal operating system without a virtual machine nor SR-IOV. The host links to the FPGA with a single PCIe physical function (PF).

Figure 4. OPAE Driver in Non-Virtualized Mode



7.1. Loading an AFU Image into the FPGA

You can utilize the `fpgasupdate` utility to load an AFU image. In Acceleration Stack 1.2.1 and later versions, the Intel FPGA PAC must be programmed with AFU images that have been prepended with mandatory headers. These headers are applied by the PACSign tool. For more information on the PACSign tool, please refer to the *Security User Guide: Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA*.

The samples included with Acceleration Stack have been processed by PACSign and the AFU binary files are located at:

```
$OPAE_PLATFORM_ROOT/hw/samples/<AFU Name>/bin/*_unsigned.gbs
```

If the Intel FPGA PAC is programmed with a root entry hash following the steps in the *Security User Guide: Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA*, then the provided AFU bitstreams (for example: `hello_afu_unsigned.gbs`) must be signed using PACSign with the appropriate root and code signing keys before you can successfully program the signed AFU bitstream.

```
sudo fpgasupdate <AFU image>
```

The `fpgasupdate` tool can program a signed AFU bitstream provided that there is a root entry hash programmed into the flash.

Note: Programming the signed AFU bitstream can take up to 2 minutes; and programming the unsigned AFU bitstream takes seconds.

The `fpgasupdate` tool also accepts PCIe Bus:Device:Function (BDF) as an additional optional argument if multiple cards are connected to the server. Use the help text (`-h`) to see how additional arguments must be passed. For example: `sudo fpgasupdate -h`.

To identify the BDF run the following command:

```
lspci | grep 09c4
```

Sample output:

```
37:00.0 Processing accelerators: Intel Corporation Device 09c4
```

In the Sample Output, the PCIe Bus is 0x37, the Device is 0x00, and the Function is 0x0.

Related Information

- Intel FPGA Software Licensing Support
- `fpgasupdate` web page
- Security User Guide: Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA

7.2. OPAE Sample Application Programs

7.2.1. Running the Hello FPGA Example

The `hello_fpga` sample host application uses the OPAE library to test the hardware in native loopback mode (NLB). Load the FPGA with the `nlb_mode_0` AFU image to run this example.

Run the following commands to test the `hello_fpga` sample host application:

1. Run the following command to load the AFU image:

   ```
   sudo fpgasupdate $OPAE_PLATFORM_ROOT/hw/samples/nlb_mode_0/bin/\
   nlb_mode_0_unsigned.gbs
   ```

2. Configure the system hugepages to allocate 20, 2 MB hugepages that this utility requires. This command requires root privileges:

   ```
   sudo sh -c "echo 20 > /sys/kernel/mm/hugepages/\
   hugepages-2048kB/nr_hugepages"
   ```

3. Compile the source code for `hello_fpga` located at `$OPAE_PLATFORM_ROOT/sw/opae*/samples/hello_fpga.c`:

   ```
   cd $OPAE_PLATFORM_ROOT/sw
   ```

4. Extract the `tar` file:

   ```
   tar xf opae*.tar.gz
   ```

   Note: This step is only necessary if you installed the OPAE software from binaries. For more information, refer to section *Installing the OPAE Software from Prebuilt Binaries*.

5. Move to the OPAE directory:

```
cd opae*
```

6. Compile the example:

   RHEL:

```
gcc -o hello_fpga -std=gnu99 -rdynamic \
-ljson-c -luuid -lpthread -lopae-c -lm -Wl,-rpath \
-lopae-c $OPAE_PLATFORM_ROOT/sw/opae*/samples/hello_fpga.c
```

   Ubuntu:

```
gcc -o hello_fpga -std=gnu99 -rdynamic \
-ljson-c -luuid -lpthread -lopae-c -lm -Wl,--no-as-needed \
-lopae-c -luuid $OPAE_PLATFORM_ROOT/sw/opae*/samples/hello_fpga.c
```

7. To run the example, type the following command:

Option	Description
For the OPAE RPM installation:	`sudo ./hello_fpga`

For an OPAE installation from source:	`sudo LD_LIBRARY_PATH=\`
	`$LD_LIBRARY_PATH:/<path to opae install>/\`
	`lib64 ./hello_fpga`

   Sample output:

```
Running Test
Done Running Test
```

   For more information about the `hello_fpga` example, refer to the following files:

   - Source code located at `$OPAE_PLATFORM_ROOT/sw/opae*/samples/hello_fpga.c`
   - *Native Loopback Accelerator Functional Unit (AFU) User Guide* for AFU register descriptions.

Related Information

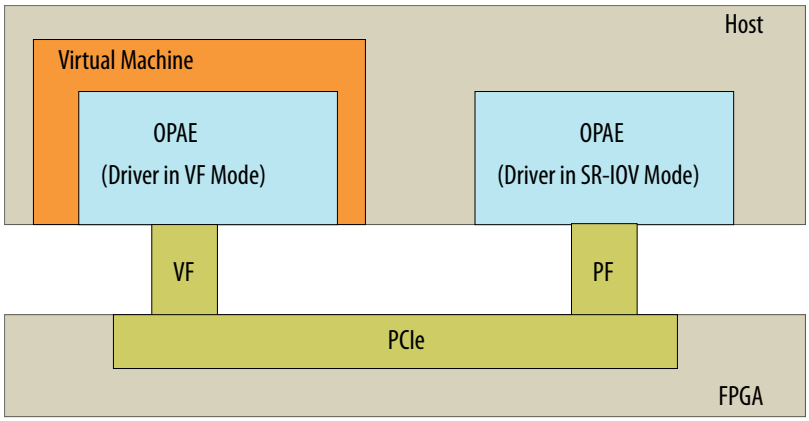
RHEL 7.6: Installing the OPAE Framework from Prebuilt Binaries (RPM) on page 17

8. Running the OPAE in a Virtualized Environment

In SR-IOV mode, a host processor uses a physical function (PF) to access management functions. A virtual machine (VM) uses a virtual function (VF) to access the AFU.

Note: Partial reconfiguration (PR) is not available in this mode.

Figure 5. OPAE Driver in SR-IOV Mode



You must complete all the steps in the *Getting Started* and *Installing the OPAE Software* chapters before you can set up a virtualized environment. An application running in a virtual machine that connects to a VF through OPAE cannot initiate partial reconfiguration. The permission table in the FME enforces this restriction. The permission table only allows partial reconfiguration through a PF. Consequently, you must load the AFU image on the host before continuing with the steps to create a virtualized environment.

Run the following command on the host to load the AFU image.

```
sudo fpgasupdate \
$OPAE_PLATFORM_ROOT/hw/samples/nlb_mode_0/bin/nlb_mode_0_unsigned.gbs
```

Related Information

- Getting Started on page 10
- Installing the OPAE Software Package on page 17

8.1. Updating Settings Required for VFs

To use SR-IOV and pass a VF to a virtual machine, you must enable the Intel IOMMU driver on the host. Complete the following steps to enable the Intel IOMMU driver:

1. Add `intel_iommu=on` to the kernel command line by updating the GRUB configuration.
2. GRUB reads its configuration from either the `/boot/grub2/grub.cfg` file on traditional BIOS-based machines or from the `/boot/efi/EFI/redhat/grub.cfg` file on UEFI machines. Depending on your system, execute one of the instructions below:

   => BIOS based machine:

   ```
   grub2-mkconfig -o /boot/grub2/grub.cfg
   ```

   => UEFI based machine

   ```
   grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
   ```
3. Restart to apply the new GRUB configuration file.
4. To verify the GRUB update, run the following command:

   ```
   cat /proc/cmdline
   ```

   The sample output below shows *`intel_iommu=on`* on the kernel command line.

   Sample output:

   ```
   BOOT_IMAGE=/vmlinuz-3.10.0-957.21.1.el7.x86_64
   root=/dev/mapper/cl_<server-name>-root ro intel_iommu=on
   crashkernel=auto rd.lvm.lv=cl_<server-name>/root
   rd.lvm.lv=cl_<server-name>/swap rhgb quiet
   ```

8.2. Configuring the VF Port on the Host

By default, the PF controls the AFU port. The following procedure transfers AFU control to the VF. After the transfer to VF control, applications running on the VM can access the AFU.

In a multicard system, if you want to configure the VF on only a single PCIe device, run the below command to find a device mapping for the specific PCIe device:

```
ls -l /sys/class/fpga/intel-fpga-dev.*
```

Sample output:

```
/sys/class/fpga/intel-fpga-dev.0 -> ../../devices/
pci0000:36/0000:36:00.0/0000:37:00.0/fpga/intel-fpga-dev.0

/sys/class/fpga/intel-fpga-dev.1 -> ../../devices/pci0000:ae/
0000:ae:00.0/0000:af:00.0/fpga/intel-fpga-dev.1
```

To target PCIe B:D.F (AF:00.0) and B:D.F (37:00.0) in the following commands, use instance id **1** and **0** instead of ***** respectively.

1. Run the following three commands, individually, to export the required paths:

```
export port_path=$(find /sys/class/fpga/intel-fpga-dev.* \
-maxdepth 1 -follow -iname intel-fpga-port.*)
```

```
export link_path=$(readlink -m /$port_path/../)
```

```
export pci_path=$link_path/../../
```

2. Release the port controlled by the PF using the `fpgaport` tool:

```
sudo fpgaport release /dev/intel-fpga-fme.* 0
```

3. Enable SR-IOV and VFs. Each VF has 1 AFU Port:

```
sudo sh -c "echo 1 > $pci_path/sriov_numvfs"
```

4. Find the additional device number for the VF device:

```
lspci -nn | grep :09c[45]
```

Sample output:

```
04:00.0 Processing accelerators [1200]: Intel Corporation Device [8086:09c4]
04:00.1 Processing accelerators [1200]: Intel Corporation Device [8086:09c5]
```

`lspci` shows an additional device number, 09c5. This is the VF device you assign to a VM. The original bus and device numbers for the PF remains 09c4.

Note that the Domain:Bus:Device.Function (BDF) notation for the VF device in this example is: 000:04:00.1. Replace this BDF with the appropriate BDF for your system.

5. Load the vfio-pci driver:

```
sudo modprobe vfio-pci
```

6. Unbind the VF device from its driver:

```
sudo sh -c "echo 0000:04:00.1 > \
/sys/bus/pci/devices/0000:04:00.1/driver/unbind"
```

7. Find the vendor and device ID for the VF device:

```
lspci -n -s 04:00.1
```

Sample output:

```
04:00.1 1200: 8086:09c5
```

8. Bind the VF to the `vfio-pci` driver:

```
sudo sh -c "echo 8086 09c5 > \
/sys/bus/pci/drivers/vfio-pci/new_id"
```

8.3. Running the Hello FPGA Example on Virtual Machine

This section assumes that you have set up the Virtual Machine (VM) and connected to the virtual function (VF) device with ID 09c5. On the virtual machine, install the Intel FPGA Driver and OPAE Software. For instructions, refer to section *Installing the OPAE Software Package*.

Complete the following steps to test the operation of the NLB mode 0 AFU in a virtualized environment:

1. Configure the system hugepages to allocate 20, 2 MB hugepages that this utility requires. This command requires root privileges:

```
sudo sh -c "echo 20 > /sys/kernel/mm/hugepages/hugepages-\
2048kB/nr_hugepages"
```

2. Complete the following commands to extract the `.tar` file:

```
tar xf $OPAE_PLATFORM_ROOT/sw/opae*.tar.gz
cd opae*
```

3. To compile, type the following command:

```
gcc -o hello_fpga -std=gnu99 -rdynamic \
-ljson-c -luuid -lpthread -lopae-c -lm -Wl,-rpath -lopae-c \
$OPAE_PLATFORM_ROOT/sw/opae*/samples/hello_fpga.c
```

4. Run the example:

```
sudo ./hello_fpga
```

   Sample output:

```
Running Test
Done Running Test
```

   For more information about the `hello_fpga` sample host application, refer to the following files:

   - Source code located at `$OPAE_PLATFORM_ROOT/sw/opae*/samples/hello_fpga.c`
   - *Native Loopback Accelerator Functional Unit (AFU) User Guide* for AFU register descriptions.

Related Information

- Installing the OPAE Software Package on page 17
- Running the Hello FPGA Example on page 28
- Native Loopback Accelerator Functional Unit (AFU) User Guide
- Documentation Available for the Intel Acceleration Stack for Intel Xeon CPU with FPGAs 1.2.1 Release on page 51

8.3.1. Disconnecting the VF from the VM and Reconnecting to the PF

1. Uninstall the driver on the VM:

```
sudo yum remove opae-intel-fpga-driver.x86_64
```

2. Detach the VF from the VM.

   On the host machine, unbind the VF PCI device from the vfio-pci driver:

```
sudo sh -c "echo -n 0000:04:00.1 > /sys/bus/pci/drivers/vfio-pci/unbind"
```

3. Bind the VF to the intel-fpga driver:

```
sudo sh -c "echo -n 0000:04:00.1 > \
  /sys/bus/pci/drivers/intel-fpga-pci/bind"
```

4. To ensure you have the correct `$pci_path` for disconnection, type:

```
export pci_path=/sys/class/fpga/intel-fpga-dev.*/device
```

   To target PCIe `B:D.F (AF:00.0)` and `B:D.F (37:00.0)` in the following commands, use instance id **1** and **0** instead of ***** respectively.

5. Set to 0 VFs and disable SR-IOV:

```
sudo sh -c "echo 0 > $pci_path/sriov_numvfs"
```

6. Assign the port to the PF using the `fpgaport` tool:

```
sudo fpgaport assign /dev/intel-fpga-fme.* 0 --numvfs 0
```

9. Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA Archives

Intel Acceleration Stack Version	User Guide (PDF)
1.2	Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA
1.1	Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA

10. Document Revision History for Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA

Document Version	Intel Quartus Prime Version	Changes
2020.03.06	1.2.1 (supported with Intel Quartus Prime Pro Edition 19.2)	• Updated tool versions • Updated download size for "Acceleration Stack for Development" • Updated the Intel PAC with Intel Arria 10 GX FPGA 1.2.1 Directory Structure figure in section *Understanding the Extracted Intel PAC with Intel Arria 10 GX FPGA Release Package* • Removed CentOS as a system requirement • Removed section *Installing Required OS Packages and Components While Installing CentOS 7.4* • Updated steps and output for sections: — *RHEL 7.6 : Installing the OPAE Framework from Prebuilt Binaries (RPM)* — *Ubuntu: Installing the OPAE Framework from Prebuilt Binaries (deb)* — *Identifying the Flash Image and BMC Firmware* — *Running FPGA Diagnostics* — *Running the OPAE in a Non-Virtualized Environment* • Updated the output for section *Why do I see a "No Suitable slots found" message when running fpgaconf on my AFU image?* • Updated *Installing the Intel PAC with Intel Arria 10 GX FPGA Card In the Host Machine* — Added "Precautions for Hardware Installation" — Updated steps • Added new section *Installing the PACSign* • Added 1.2 version of the Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA to section *Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA Archives*. • Added security and authentication features: — Trusted Configuration Manager (TCM) and Bitstream Authentication IP (BIP) for secure updates — OPAE security tools: • FPGA one-time secure update (`fpgaotsu`) • FPGA secure update (`fpgasupdate`) • Super-RSU (`super-rsu`) • PACSign
2019.08.05	1.2 (supported with Intel Quartus Prime Pro Edition 17.1.1)	Corrected a document title in appendix *Documentation Available for the Intel Acceleration Stack for Intel Xeon CPU with FPGAs 1.2 Release*: From *HSSI User Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA* to *Networking Interface for Open Programmable Acceleration Engine: Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA*.

continued...

Document Version	Intel Quartus Prime Version	Changes
2019.05.30	1.2 (supported with Intel Quartus Prime Pro Edition 17.1.1)	• Updated steps under section: — *Configuring the VF Port on the Host* — *Updating FPGA Flash and BMC Firmware* — *Troubleshooting OPAE Installation on CentOS*
2019.03.08	1.2 (supported with Intel Quartus Prime Pro Edition 17.1.1)	Added a note regarding use of `pacd` at the top of section *Handling Graceful Thermal Shutdown*.
2019.02.23	1.2 (supported with Intel Quartus Prime Pro Edition 17.1.1)	Added a note regarding upgrading to the Intel Acceleration Stack version 1.2 in the *Correspondence Between Acceleration Stack, FIM, and OPAE Versions* table in section *Identify the Flash Image and BMC Firmware*.
2019.02.06	1.2 (supported with Intel Quartus Prime Pro Edition 17.1.1)	• Added a note about the `init_env.sh` script in the *Installing the Intel Acceleration Stack Runtime Package on the Host Machine* and *Intel Acceleration Stack Development Package on the Host Machine*. • Moved section *Updating the Flash Image and BMC Firmware* to an appendix. • Added a important note about BMC firmware version permissions after upgrading to version 26889. • Added section *Troubleshooting Frequently Asked Questions (FAQ)*.
2018.12.04	1.2 (supported with Intel Quartus Prime Pro Edition 17.1.1)	• Added the following sections: — *Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA Archives* that contains the archived versions of this document. — *Ubuntu: Installing the OPAE Framework from Prebuilt Binaries (deb)* — Handling Graceful Thermal Shutdown • Removed section *Updating the Board Management Controller (BMC) Configuration and Firmware*.
2018.08.27	1.1 supported with Intel Quartus Prime Pro Edition 17.1.1)	Added the following sections: • *FPGA Device Access Permission* • *Memlock Unit* • *Hugepage Settings*
2018.08.14	1.1 supported with Intel Quartus Prime Pro Edition 17.1.1)	Made the following changes: • Changed `a10_gx_pac_ias_1_1_pv_eth.pv` to `a10_gx_pac_ias_1_1_pv_eth.patch` in *Installing the Intel Acceleration Stack Runtime package on the Host Machine* and *Installing the Intel Acceleration Stack Development Package on the Host Machine* • Corrected the OPAE Software Development Kit (SDK) version `1.1` `.tar` file names and Download sizes in *Installing the Intel Acceleration Stack*
2018.08.06	1.1 supported with Intel Quartus Prime Pro Edition 17.1.1)	Initial release.

A. Updating the FIM and BMC Firmware

A.1. Selecting the Correct Update Method

The following table provides instructions to update the FIM based on the FIM currently running on the Intel PAC with Intel Arria 10 GX FPGA.

To determine the FIM version on the Intel PAC with Intel Arria 10 GX FPGA, refer to section Identifying the Flash Image and BMC Firmware on page 23.

Table 6. Selecting the Correct Update Method

Current FIM Version loaded on the PAC	FIM and BMC Update Instructions
1.2 or newer	1. Ensure the Intel Acceleration Stack version 1.2.1 release is installed. 2. Follow the instructions in section Updating FPGA Flash and BMC Firmware *on page 39*.
1.1	1. Navigate to the Intel Acceleration Stack Archived Versions web page and ensure that you select Intel Acceleration Stack Version 1.2 from the left column. 2. Select and download the Intel Acceleration Stack download. 3. Scroll down to the Intel Acceleration Stack Version 1.2 Resources section and click on the **Get Resources** button. 4. Unzip the resources folder and retrieve the *Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA* document. 5. Refer to the following sections in the *Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA* document to provide you with instructions on how to use the `setup_fim_and_bmc.sh` script to update the FIM and BMC to the 1.2 release: • *Installing the Intel Acceleration Stack* • *Identifying the Flash Image and BMC Firmware* • *Updating the FIM and BMC Firmware*
Older than 1.1	1. Navigate to the Intel Acceleration Stack Archived Versions web page and ensure that you select Intel Acceleration Stack Version 1.1 from the left column. 2. Select and download the Intel Acceleration Stack download. 3. Scroll down to the Intel Acceleration Stack Version 1.1 Resources section and click on the **Get Resources** button. 4. Unzip the resources folder and retrieve the *Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA* document. 5. Refer to the following sections in the *Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA* document to provide you with instructions on how to update the FIM and BMC to the 1.1 release: • *Installing the Intel Acceleration Stack* • *Identifying and Updating the FIM* • *Updating the Board Management Controller (BMC) Configuration and Firmware* *Note:* You can identify the BMC firmware from the *Download Intel Acceleration Stack Version 1.1* table on the Intel Acceleration Stack Archived Versions web page.

Related Information

- Open Programmable Acceleration Engine - Documentation web page on GIT
- Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA Archives on page 35
  Provides a list of user guides for previous versions of the Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA IP core.

A.1.1. Updating FPGA Flash and BMC Firmware

Before you begin:

- The FIM must be at version 1.2 and the BMC version must be 26889.
- OPAE Software package (Version 1.1.2-2) must be installed. This can be checked by running:

```
rpm -qa | grep opae
```

- Do not interrupt the commands.
- These commands must be run on a Host Machine and not on a Virtual Machine.
- Stop any service or daemon accessing the FPGA before updating the FPGA flash. For example, stop the `pacd` service by using command:

```
systemctl stop pacd.service
```

- All files required for the FIM and BMC update are located at `/usr/share/opae/a10-gx-pac/*`.

Important: You must not downgrade your BMC firmware after upgrading to version 26895. BMC version 26895 only supports the Acceleration Stack 1.2.1 FIM (38d782e3-b612-5343-b934-2433e348ac4c) and no prior Acceleration Stack release is supported. Please check compatibility of intended AFU workloads with Acceleration Stack 1.2.1 before upgrading the board. For questions, please contact your Intel sales representative or workload solutions partner.

The following steps update the Intel PAC with Intel Arria 10 GX FPGA card:

1. Run the one-time secure update command to update the FPGA flash:

   [RHEL]:

   ```
   sudo fpgaotsu /usr/share/opae/a10*/one*/otsu-09C4.json
   ```

   [Ubuntu]:

   ```
   sudo fpgaotsu /usr/share/opae/a10*/fpgaotsu/base/otsu-09C4.json
   ```

   Note: This command can take up to 40 minutes to complete. Stop any service accessing the FPGA such as `pacd` before performing the update.

Figure 6. Sample Output:

```
[2020-01-15 23:00:12,976] [INFO    ] [MainThread] Intel PAC with Intel Arria 10 GX FPGA 0000:05:00.0 is not secure.
[2020-01-15 23:00:12,981] [WARNING ] [MainThread] Update starting. Please do not interrupt.
[2020-01-15 23:00:12,981] [INFO    ] [0000:05:00.0] Updating Intel PAC with Intel Arria 10 GX FPGA : 0000:05:00.0
[2020-01-15 23:00:12,982] [INFO    ] [0000:05:00.0] Erasing flash@0x7fd0000 for 196608 bytes
[2020-01-15 23:00:12,997] [INFO    ] [0000:05:00.0] Writing flash@0x7ff0000 for 104 bytes (RushCreek_Release_key_blk2)
(100%) [########################################] [104/104 bytes][Time:0:00:00.000318]
[2020-01-15 23:00:12,998] [INFO    ] [0000:05:00.0] Reading flash@0x7ff0000 for 104 bytes for verification
(100%) [########################################] [104/104 bytes][Time:0:00:00.004193]
[2020-01-15 23:00:13,003] [INFO    ] [0000:05:00.0] Verified flash@0x7ff0000 for 104 bytes (RushCreek_Release_key_blk2)
[2020-01-15 23:00:13,003] [INFO    ] [0000:05:00.0] Erasing flash@0x1800000 for 41943040 bytes
[2020-01-15 23:00:15,977] [INFO    ] [0000:05:00.0] Writing flash@0x1800000 for 41943040 bytes (dcp_1_2_1_rot_reversed.rpd)
(100%) [########################################] [41943040/41943040 bytes][Time:0:00:48.696761]
[2020-01-15 23:01:04,696] [INFO    ] [0000:05:00.0] Reading flash@0x1800000 for 41943040 bytes for verification
(100%) [########################################] [41943040/41943040 bytes][Time:0:00:41.287431]
[2020-01-15 23:01:46,014] [INFO    ] [0000:05:00.0] Verified flash@0x1800000 for 41943040 bytes (dcp_1_2_1_rot_reversed.rpd)
[2020-01-15 23:01:46,015] [INFO    ] [0000:05:00.0] Erasing flash@0x10000 for 25100288 bytes
[2020-01-15 23:02:24,593] [INFO    ] [0000:05:00.0] Writing flash@0x10000 for 25100288 bytes (dcp_1_2_1_rot_reversed.rpd)
(100%) [########################################] [25100288/25100288 bytes][Time:0:00:29.741170]
[2020-01-15 23:02:54,348] [INFO    ] [0000:05:00.0] Reading flash@0x10000 for 25100288 bytes for verification
(100%) [########################################] [25100288/25100288 bytes][Time:0:00:24.656107]
[2020-01-15 23:03:19,024] [INFO    ] [0000:05:00.0] Verified flash@0x10000 for 25100288 bytes (dcp_1_2_1_rot_reversed.rpd)
[2020-01-15 23:03:19,024] [INFO    ] [0000:05:00.0] Erasing flash@0x0 for 65536 bytes
[2020-01-15 23:03:19,139] [INFO    ] [0000:05:00.0] Writing flash@0x0 for 65536 bytes (dcp_1_2_1_rot_reversed.rpd)
(100%) [########################################] [65536/65536 bytes][Time:0:00:00.086483]
[2020-01-15 23:03:19,227] [INFO    ] [0000:05:00.0] Reading flash@0x0 for 65536 bytes for verification
(100%) [########################################] [65536/65536 bytes][Time:0:00:00.064667]
[2020-01-15 23:03:19,292] [INFO    ] [0000:05:00.0] Verified flash@0x0 for 65536 bytes (dcp_1_2_1_rot_reversed.rpd)
[2020-01-15 23:03:19,292] [INFO    ] [0000:05:00.0] Erasing flash@0x4000000 for 66912256 bytes
[2020-01-15 23:04:02,432] [INFO    ] [0000:05:00.0] Writing flash@0x3000000 for 71106560 bytes (dcp_1_2_1_rot_reversed.rpd)
(100%) [########################################] [71106560/71106560 bytes][Time:0:01:12.890863]
[2020-01-15 23:05:15,360] [INFO    ] [0000:05:00.0] Reading flash@0x3000000 for 71106560 bytes for verification
(100%) [########################################] [71106560/71106560 bytes][Time:0:01:09.815452]
[2020-01-15 23:06:25,229] [INFO    ] [0000:05:00.0] Verified flash@0x3000000 for 71106560 bytes (dcp_1_2_1_rot_reversed.rpd)
[2020-01-15 23:06:25,278] [INFO    ] [MainThread] Total time: 0:06:12.296996
[2020-01-15 23:06:25,278] [INFO    ] [MainThread] One-Time Secure Update OK
```

2. Power cycle the server.
3. Run the super-rsu command to update the BMC bootloader and firmware.

   [RHEL]:

   ```
   sudo super-rsu /usr/share/opae/a10*/super*/*.json
   ```

   [Ubuntu]:

   ```
   sudo super-rsu /usr/share/opae/a10-gx-pac/super-rsu/base/rsu-09C4.json
   ```

Figure 7. Sample Output:

```
[2020-01-15 23:09:10,991] [WARNING ] [MainThread  ] - Update starting. Please do not interrupt.
[2020-01-15 23:09:11,213] [WARNING ] Update starting. Please do not interrupt.
[2020-01-15 23:09:11,213] [INFO    ] updating from file /usr/share/opae/a10-gx-pac/super-rsu/a10sa4_bootloader-26895-fw_Release.bin with size 38016
[2020-01-15 23:09:11,315] [INFO    ] writing to staging area
(100%) [########################################] [38016/38016 bytes][Time:0:00:01.013865]
[2020-01-15 23:09:12,339] [INFO    ] applying update to 0000:05:00.0
(100%) [########################################][Time:0:00:08.011400]
[2020-01-15 23:09:20,351] [INFO    ] update of 0000:05:00.0 complete
[2020-01-15 23:09:20,352] [INFO    ] Secure update OK
[2020-01-15 23:09:20,352] [INFO    ] Total time: 0:00:09.139215
[2020-01-15 23:09:20,655] [WARNING ] Update starting. Please do not interrupt.
[2020-01-15 23:09:20,656] [INFO    ] updating from file /usr/share/opae/a10-gx-pac/super-rsu/a10sa4-26895-fw_Release.bin with size 244864
[2020-01-15 23:09:20,757] [INFO    ] writing to staging area
(100%) [########################################] [244864/244864 bytes][Time:0:00:01.514407]
[2020-01-15 23:09:22,282] [INFO    ] applying update to 0000:05:00.0
(100%) [########################################][Time:0:00:44.059691]
[2020-01-15 23:10:06,342] [INFO    ] update of 0000:05:00.0 complete
[2020-01-15 23:10:06,343] [INFO    ] Secure update OK
[2020-01-15 23:10:06,343] [INFO    ] Total time: 0:00:45.687732
[2020-01-15 23:10:07,064] [INFO    ] [MainThread  ] - 1 board updated. A power-cycle is required.
[2020-01-15 23:10:07,065] [INFO    ] [MainThread  ] - super_rsu.pyc completed in: 0:00:56.072473
[2020-01-15 23:10:07,065] [INFO    ] [MainThread  ] - super-rsu exiting with code '0'
1 board updated. A power-cycle is required.
```

4. Power cycle the server.
5. Ensure BMC version (26895) and PR Interface ID or FIM version (38d782e3-b612-5343-b934-2433e348ac4c) is displayed while running:

   ```
   sudo fpgainfo fme
   ```

For more information about the `fpgasupdate` tool, refer to the Open Programmable Acceleration Engine - Documentation web page on GIT and the *Security User Guide: Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA* (available at a later date).

Related Information

Open Programmable Acceleration Engine (OPAE) Documentation

B. Handling Graceful Thermal Shutdown

Note:

- Qualified OEM server systems provide adequate cooling for standard workloads and the use of `pacd` may be optional.
- For details about using `pacd`, including considerations that may lead to an unexpected system reboot, refer to the pacd documentation on the OPAE web page.

The Intel FPGA PAC Daemon (`pacd`) is a program that can be used to help protect the server from crashing due to hardware reaching an upper or lower non-recoverable sensor threshold. `pacd` is capable of monitoring any of the 23 sensors reported by Board Management Controller. `pacd` can be run standalone, as a daemon, or as a *systemd* service. When the *OPAE tools extra package* is installed, `pacd` gets placed in the OPAE binaries directory (default: */usr/bin*) along with configuration and service files – `pacd.conf` and `pacd.service`, respectively.

On startup, `pacd` sets its thresholds to a constant -5 degrees or 5 percent lower power under the BMC threshold values. This is to pass on the Graceful Thermal Shutdown responsibility to `pacd`.

`pacd` periodically reads the sensor values and if the values exceed the threshold, it sends a `SIGHUP` signal to all running processes and makes the board inaccessible from the host. The daemon waits for a configurable time specified by `-c` in `pacd.conf`, as described below, to cool down the board. After this configurable wait time elapses, the `pacd` service programs the specified AFU. Ensure that the AFU host application monitors for a `SIGHUP` signal and exits.

Note: Partial reconfiguration cannot be initiated from a Virtual Machine. Hence, `pacd` cannot run on a Virtual Machine.

`pacd` can be set up as a *systemd* service as follows (using a shell with elevated privileges (sudo)):

1. Edit the `pacd.conf` file to update the "`DefaultGBSOptions`" entry with a list of AFUs appropriate for your FIM. Use the full absolute path to each AFU file and precede each file name with '`-n`'.

   ```
   sudo vim /usr/bin/pacd.conf
   ```

   Edit entry:

   ```
   DefaultGBSOptions=-n /home/<username>/intelrtestack/\
   a10_gx_pac_ias_1_2_1/hw/samples/nlb_mode_3/bin/nlb_mode_3_unsigned.gbs
   ```

Note: Optional settings include:

- PCIe address (For example: -S, -B, -D, -F), `pacd` monitors all Intel FPGA PACs matching the PCIe address components specified. For example, if you specify -B 5 only, all Intel FPGA PACs on PCIe bus 5 becomes monitored.
- Sensor Threshold—The thresholds are global, so specifying -T 11:95.0:93.0 monitors sensor 11 on all selected Intel FPGA PACs. When the value exceeds 95.0, it causes the default bitstream specified with -n in pacd.conf to be programmed (PR). The sensor is considered triggereduntil its value drops below 93.0.
- You can specify a cool down period by:
  a. Changing the `CooldownInterval=<time period>`
  b. Setting `-c <time period>` for `ThresholdOptions` in `pacd.conf` or

  If both are set, then the `-c <time period>` for `ThresholdOptions` takes precedence.
- The Sensor Number can be found by running this command:

```
sudo fpgainfo bmc
```

Examine the remaining option variables and adjust as appropriate for your system.

2. Copy `pacd.conf` to the default *systemd* service configuration directory (typically */etc/sysconfig*).

   RHEL:

```
sudo cp /usr/bin/pacd.conf /etc/sysconfig/
```

   Ubuntu:

```
sudo cp /usr/bin/pacd.conf /etc/default
```

3. Edit the `pacd.service` file to update "`EnvironmentFile`" entry to reflect where the `pacd.conf` file was copied. Prepend the path name with a single dash '`-`', and specify the path as absolute.

```
sudo vim /usr/bin/pacd.service
```

   Edit entry:
   RHEL:

```
EnvironmentFile=-/etc/sysconfig/pacd.conf
```

   Ubuntu:

```
EnvironmentFile=-/etc/default/pacd.conf
```

4. Copy `pacd.service` to `/etc/systemd/system/pacd.service`. This makes `pacd` visible to *systemd*.

   RHEL:

```
sudo cp /usr/bin/pacd.service /etc/systemd/system/
```

   Ubuntu:

```
sudo cp /usr/bin/pacd.service /lib/systemd/system/
```

5. Start `pacd` as a *systemd* service.

   Note: Please use `sudo` if command cannot be run in regular user mode.

   ```
   systemctl daemon-reload
   ```

   ```
   systemctl start pacd.service
   ```

6. Optional: To enable `pacd` to re-start on boot, execute

   ```
   systemctl enable pacd.service
   ```

   To check whether `pacd` has started and to check state or actions, please examine the log file (specified in `pacd.conf` on the "`LogFile`" line).

   For a full list of `systemctl` commands, run the following command:

   ```
   systemctl -h
   ```

7. To verify that the service is running, run the following command:

   ```
   systemctl status pacd.service
   ```

8. To stop the service:

   ```
   systemctl stop pacd.service
   ```

Related Information

Open Programmable Acceleration Engine - Documentation web page on GIT

C. FPGA Device Access Permission

To be able to run AFU samples and other OPAE tools as non-root user, you can run the below command to change file access permissions.

Use file access permissions on the Intel FPGA device file directories, `/dev/intel-fpga-fme.*` and `/dev/intel-fpga-port.*` to control access to FPGA accelerators and devices. Use the same file access permissions to control access to the files reachable through `/sys/class/fpga/`.

The * denotes the respective socket, for example 0 or 1.

Typically, you must change these permissions after every restart. To make the changes permanent, add these permissions to `/etc/bashrc` as well.

Here are the commands to run:

```
sudo chmod 666 /dev/intel-fpga-fme.*
sudo chmod 666 /dev/intel-fpga-port.*
sudo chmod 666 /sys/class/fpga/intel-fpga-dev.*/intel-fpga-port.*\
/userclk_freqcmd
sudo chmod 666 /sys/class/fpga/intel-fpga-dev.*/intel-fpga-port.*\
/userclk_freqcntrcmd
sudo chmod 666 /sys/class/fpga/intel-fpga-dev.*/intel-fpga-port.*/errors/clear
```

D. Memlock Limit

Depending on the requirements of your application, you may also want to increase the maximum amount of memory that a user process can lock. The exact method may vary with your Linux distribution.

Use `ulimit -l` to check the current memlock setting:

```
ulimit -l
```

To permanently remove the locked memory limit for a regular user, add the following lines to `/etc/security/limits.conf`:

```
user1    hard   memlock           unlimited
user1    soft   memlock           unlimited
```

The previous commands remove the limit on locked memory for `user1`. To remove memory locks for all users, replace `user1` with `*`:

```
*    hard   memlock           unlimited
*    soft   memlock           unlimited
```

Note: Settings in the `/etc/security/limits.conf` file do not apply to services. To increase the locked memory limit for a service, modify the application's `systemd` service file to add the following line:

```
[Service]
LimitMEMLOCK=infinity
```

E. Hugepage Settings

Use the hugepage settings to reserve `2 MB-hugepages` or `1 GB-hugepages`. For example, the `hello_fpga` sample requires several `2 MB-hugepages`.

The following command below reserves 20, `2 MB-hugepages`:

```
sudo sh -c 'echo 20 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages'
```

The following command below reserves 4, `1 GB-hugepages`:

```
sudo sh -c 'echo 4 > /sys/kernel/mm/hugepages/hugepages-1048576kB/nr_hugepages'
```

Note: To make these changes permanent, include them as part of `/etc/bashrc`.

F. Troubleshooting Frequently Asked Questions (FAQ)

F.1. Why do I see a "No Suitable slots found" message when running `fpgaconf` on my AFU image?

If you see a **No suitable slots found** message, ensure that your FIM version is compatible with your AFU image by completing the following steps:

1. Refer to Identifying the Flash Image and BMC Firmware on page 23 to determine the required *<FIM version>*.
2. To verify that the AFU is compatible with the FIM version, run the following command:

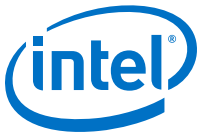
```
packager gbs-info --gbs=<gbs-file>
```

For example, for `nlb_mode_0.gbs` run the following command:

```
packager gbs-info --gbs=$OPAE_PLATFORM_ROOT/hw/samples/nlb_mode_0/bin/
\nlb_mode_0_unsigned.gbs
```

Sample output:

```
{
    "afu-image": {
        "interface-uuid": "14574a97-8b70-5111-a272-4c8d4197642c
",
        "afu-top-interface": {
            "class": "ccip_std_afu_avalon_mm",
            "module-ports": [
                {
                    "params": {
                        "clock": "pClk"
                    },
                    "optional": true,
                    "class": "local-memory"
                }
            ]
        },
        "magic-no": 488605312,
        "power": 0,
        "accelerator-clusters": [
            {
                "total-contexts": 1,
                "name": "nlb_mode_0",
                "accelerator-type-uuid": "d8424dc4-a4a3-c413-
f89e-433683f9040b"
            }
        ]
    },
    "version": 1,
    "signature": {
        "root_hash": "0xf8 0xff 0x7e 0x0a 0x52 0xa3 0x78 0x48 0x3c 0x85
0x30 0x1d 0xf4 0x9c 0x7d 0x55 0xff 0xd2 0x6f 0x79 0x41 0x21 0xbd 0xb8 0xb1
0x02 0xd7 0xe1 0xc3 0x13 0x2b 0xb9",
        "CSK": null,
        "root_pub_key-Y": "0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
```



```
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00",
        "root_pub_key-X": "0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00",
        "CSK_pub_key-Y": "0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00",
        "CSK_pub_key-X": "0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00",
        "root_key": null
    }
}
```

The interface-uuid should match the FIM version (PR interface ID) you found in Identifying the Flash Image and BMC Firmware on page 23.

F.2. How do I flash the FIM or program the AFU in a multicard system?

The OPAE commands `fpgasupdate` and `fpgabist` use the PCIe bus, device and function numbers as arguments to target the specific Intel FPGA PAC.

Use the `--help` option for details on how to use these commands. For example:

```
fpgasupdate --help
```

F.3. Which environment variables are required?

To ensure all environment variables are set, you must source the initialization script that is provided as part of the installer.

or

```
source /<installation directory>/inteldevstack/init_env.sh
```

F.4. What actions do I take if I see the error message "Error enumerating resources: no driver available"?

1. Validate that your card is detected by PCIe.

   ```
   lspci | grep 09c4
   ```

   If it is not detected, remove the card and then plug it back in again.
2. Reinstall OPAE by following steps listed in section *Installing the OPAE Software*.

Related Information

Installing the OPAE Software Package on page 17

F.5. Troubleshooting OPAE Installation on RHEL

Find the Intel FPGA drivers loaded.

```
sudo lsmod | grep fpga
```

If no output is returned, follow the below troubleshooting steps.

1. Find the kernel version used by the system

   ```
   uname -a
   ```

   Sample Output:

   ```
   Linux  3.10.0-957.el7.x86_64 #1 SMP Mon Sept 21 23:36:36 UTC 2018 x86_64
   x86_64 x86_64 GNU/Linux
   ```

2. Update the kernel source

   ```
   sudo yum install "kernel-devel-uname-r == $(uname -r)"
   ```

3. Remove the old kernel header and install the relevant kernel headers.

   ```
   sudo yum remove kernel-headers.x86_64
   sudo yum  install kernel-headers-`uname -r`
   ```

4. Remove the Intel FPGA driver.

   ```
   sudo yum remove opae-*.x86_64
   ```

5. Reinstall the driver by either running the install script (`setup.sh`) or the command below:

   ```
   cd $OPAE_PLATFORM_ROOT/sw
   sudo yum install opae-*.rpm
   ```

   In some cases, if you don't have the right kernel, you may need to update the kernel using:

   ```
   sudo yum update
   ```

   Reboot the system and select the new kernel in the grub menu. Then follow the above steps 2 to 5.

G. Documentation Available for the Intel Acceleration Stack for Intel Xeon CPU with FPGAs 1.2.1 Release

The following documents are on the Intel FPGA web page. To access a document, click the link.

Table 7. Documentation Available for the Intel Acceleration Stack for Intel Xeon CPU with FPGAs 1.2.1 Release

Document	Link to Access Document
10 Gbps Ethernet AFU Design Example User Guide	10 Gbps Ethernet Accelerator Functional Unit (AFU) Design Example User Guide
40 Gbps Ethernet AFU Design Example User Guide	40 Gbps Ethernet Accelerator Functional Unit (AFU) Design Example User Guide
Open Programmable Acceleration Engine (OPAE) C API Programming Guide	GitHub Link
Open Programmable Acceleration Engine (OPAE) Linux Device Driver Architecture Guide	GitHub Link
Open Programmable Acceleration Engine (OPAE) Tools Guide	GitHub Link
Intel Accelerator Functional Unit (AFU) Simulation Environment (ASE) User Guide	GitHub Link
Intel Accelerator Functional Unit (AFU) Simulation Environment (ASE) Quick Start User Guide	Intel Accelerator Functional Unit (AFU) Simulation Environment (ASE) Quick Start User Guide
Acceleration Stack for Intel Xeon CPU with FPGAs Core Cache Interface (CCI-P) Reference Manual	Acceleration Stack for Intel Xeon CPU with FPGAs Core Cache Interface (CCI-P) Reference Manual
Accelerator Functional Unit (AFU) Developer User Guide	Accelerator Functional Unit (AFU) Developer User Guide
Streaming DMA Accelerator Functional Unit (AFU) User Guide	Streaming DMA Accelerator Functional Unit AFU User Guide
Native Loopback Accelerator Functional Unit (AFU) User Guide	Native Loopback Accelerator Functional Unit (AFU) User Guide
Networking Interface for Open Programmable Acceleration Engine (OPAE): Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA, previously known as *HSSI User Guide for Intel FPGA Programmable Acceleration Card (Intel FPGA PAC) Intel Arria 10 GX FPGA*	Networking Interface for Open Programmable Acceleration Engine (OPAE): Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA
OpenCL on Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA Quick Start User Guide	OpenCL on Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA Quick Start User Guide
Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA with Intel Arria 10 GX FPGA Datasheet	Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA with Intel Arria 10 GX FPGA Datasheet
Intel Acceleration Stack for Intel Xeon CPU with FPGAs v1.2.1 Release Notes	Intel Acceleration Stack for Intel Xeon CPU with FPGAs v1.2.1 Release Notes
Security User Guide: Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA	Security User Guide: Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA