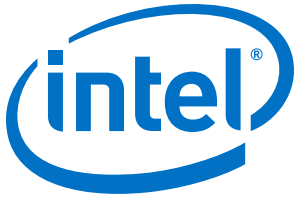


Intel® Agilex™ Configuration User Guide

Updated for Intel® Quartus® Prime Design Suite: **19.4**

UG-20205 | 2020.03.13
Latest document on the web: PDF | HTML

Contents

1. Intel® Agilex™ Configuration User Guide

1.1. Intel® Agilex™ Configuration Overview

All Intel® Agilex™ devices include a Secure Device Manager (SDM) to manage FPGA configuration and security. The SDM provides a failsafe, strongly authenticated, programmable security mode for device configuration. Previous FPGA families include a fixed state machine to manage device configuration.

The Intel Quartus® Prime software also provides flexible and robust security features to protect sensitive data, intellectual property, and the device itself under both remote and physical attacks. Configuration bitstream authentication ensures that the firmware and configuration bitstream are from a trusted source. Encryption prevents theft of intellectual property. The Intel Quartus Prime software also compresses FPGA bitstreams, reducing memory utilization.

Intel describes configuration schemes from the point-of-view of the FPGA. Intel Agilex devices support active and passive configuration schemes. In active configuration schemes the FPGA acts as the master and the external memory acts as a slave device. In passive configuration schemes an external host acts as the master and controls configuration. The FPGA acts as the slave device. All Intel Agilex configuration schemes support design security, and partial reconfiguration. All Intel Agilex active configuration schemes support remote system update (RSU) with quad SPI flash memory. To implement RSU in passive configuration schemes, an external controller must store and drive the configuration bitstream.

Intel Agilex devices support the following configuration schemes:

- Avalon® Streaming (Avalon-ST)
- JTAG
- Configuration via Protocol (CvP)
- Active Serial (AS) normal and fast modes
- Secure Digital and Multi Media Card (SD/MMC)

Table 1. Intel Agilex Configuration Data Width, Clock Rates, and Data Rates

Configuration Scheme		Data Width (bits)	MSEL[2:0]
Passive	Avalon-ST	32	000
		16	101
		8	110
	JTAG	1	111
	Configuration via Protocol (CvP)	x8, x16 lanes(1)	001(2)
Active	SD/MMC	4/8	100
	AS - fast mode	4	001
	AS - normal mode	4	011

Avalon-ST

The Avalon-ST configuration scheme is a passive configuration scheme. Avalon-ST is the fastest configuration scheme for Intel Agilex devices. Avalon-ST configuration supports x8, x16, and x32 modes. The x16 and x32 bit modes use general-purpose I/Os (GPIOs) for configuration. The x8 bit mode uses dedicated SDM I/O pins.

Note: The `AVST_data[15:0]`, `AVST_data[31:0]`, `AVST_clk`, and `AVST_valid` use dual-purpose GPIOs which operate at 1.2 V. You can use these pins as regular I/Os after the device enters user mode.

Avalon-ST supports backpressure using the AVST_READY and AVST_VALID pins. Because the time to decompress the incoming bitstream varies, backpressure support is necessary to transfer data to the Intel Agilex device. For more information about the Avalon-ST refer to the *Avalon Interface Specifications*.

(1) Intel Agilex P-tile PCIe* port 0 supports Gen3 and Gen4 x8 and x16. For more information, refer to the *Intel FPGA P-Tile Avalon Streaming (Avalon -ST) IP for PCI Express* User Guide* and the *Intel Agilex Configuration via Protocol (CvP) Implementation User Guide*.

(2) Before you can use CvP you must configure either the periphery image or full image configuration via the AS scheme. Then you can configure the core image using CvP.

JTAG

You can configure the Intel Agilex device using the dedicated JTAG pins. The JTAG port provides seamless access to many useful tools and functions. In addition to configuring the Intel Agilex, you use the JTAG port for debugging with Signal Tap or the System Console tools.

The JTAG port has the highest priority and overrides the MSEL pin settings. Consequently, you can configure the Intel Agilex device over JTAG even if the MSEL pins specify a different configuration scheme unless you disabled JTAG for security reasons.

CvP

CvP uses an external PCIe host device as a Root Port to configure the Intel Agilex device over the PCIe link. You can specify up to a x16 PCIe link. Typically, the bitstream compression ratio and the SDM input buffer data rate, not the PCIe link width, limit the configuration data rate. Intel Agilex devices support two CvP modes, CvP init and CvP update.

CvP initialization process includes the following two steps:

1. CvP configures the FPGA periphery image which includes I/O and hard IP blocks, including the PCIe IP. CvP uses quad SPI memory in AS x4 mode to configure the FPGA fabric. Because the PCIe IP is in the periphery image, PCIe link training establishes the PCIe link of the CvP PCIe IP before the core fabric configures.
2. The host device uses the CvP PCIe link to configure your design in the core fabric.

CvP update mode updates the FPGA core image using the PCIe link already established from a previous full chip configuration or CvP init configuration. After the Intel Agilex enters user mode, you can use the CvP update mode to reconfigure the FPGA fabric. This mode has the following advantages:

- Allows reprogramming of the core to run different algorithms.
- Provides a mechanism for standard updates as a part of a release process.
- Customizes core processing for different components that are part of a complex system.

  For both CvP Init and CvP Update modes, the maximum data rate depends on the PCIe generation and number of lanes.

For Intel Agilex SoC devices, CvP is only supported in FPGA configuration first mode.

AS Normal Mode

Active Serial x4 or AS x4 or Quad SPI is an active configuration scheme that supports flash memories capable of three- and four-byte addressing. Upon power up, the SDM boots from a boot ROM which uses three-byte addressing to load the configuration firmware from the Quad SPI flash. After the configuration firmware loads, the Quad SPI flash operates using four-byte addressing for the rest of the configuration process.

AS Fast Mode

The only difference between AS normal mode and fast mode is speed. Use AS fast mode when configuration timing is a concern. This mode does not delay for 10 ms before beginning configuration. Use this mode to meet the 100 ms of power up requirement for PCIe or for other systems with strict timing requirements.

In AS fast mode, the power-on sequence must ensure that the quad SPI flash memory is out of reset before the SDM because the Intel Agilex device accesses flash memory immediately after exiting reset. The power supply must be able to provide an equally fast ramp up for the Intel Agilex device and the external AS x4 flash devices. Failing to meet this requirement causes the SDM to report that the memory is missing. Consequently, configuration fails.

SD/MMC

SD/MMC is an active configuration scheme. The Intel Agilex SDM can initiate configuration from SD, Secure Digital High Capacity (SDHC*), Secure Digital Extended Capacity (SDXC*), MMC cards, and eMMC devices. The advantages of this mode are cost, capacity, availability, portability, and compatibility. Because the SDM I/O configuration pins in Intel Agilex devices operate at 1.8 volt an intermediate voltage level shifter may be required to interface with the higher voltage I/Os in SD/MMC devices.

Note: The SD/MMC configuration scheme is not supported in the current release.

Related Information

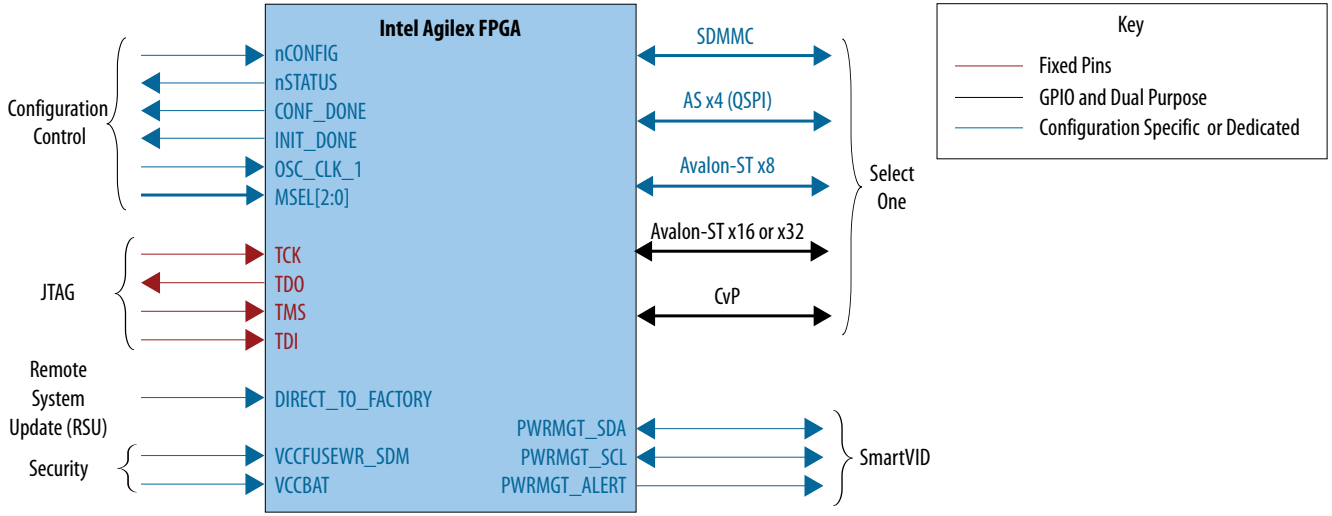
- Avalon Interface Specifications
- Intel Agilex Configuration via Protocol (CvP) Implementation User Guide
- Intel Agilex Device Data Sheet
- Intel FPGA P-Tile Avalon Streaming (Avalon ST) IP for User Guide

1.1.1. Configuration and Related Signals

The following figure shows the configuration interfaces and configuration-related device functions. Pins shown in dark blue use dedicated SDM I/Os. Pins shown in black use general purpose I/Os (GPIOs). Pins shown in red are dedicated JTAG I/Os.

You specify SDM I/O pin functions using the **Device ➤ Configuration ➤ Device and Pin Options** dialog box in the Intel Quartus Prime software.

Figure 1. Intel Agilex Configuration Interfaces



This user guide discusses most of the interfaces shown in the figure. Refer to the separate *Intel Agilex Configuration via Protocol (CvP) Implementation User Guide* and *Intel Agilex Power Management User Guide* for more information about those features.

Related Information

- SDM Pin Mapping on page 25
- Intel Agilex Power Management User Guide
- Intel Agilex Configuration via Protocol (CvP) Implementation User Guide

1.1.2. Intel Download Cables Supporting Configuration in Intel Agilex Devices

Intel provides the following cables to download your design to the Intel Agilex device on the PCB. Download cables support prototyping activity by providing detailed debug messages via Intel Quartus Prime Programmer. You must use Intel download cables for advanced debugging using the Signal Tap logic analyzer or the System Console tools.

Table 2. Intel Agilex-Supported Download Cable Capabilities

Download Cable	Protocol Support Intel Agilex Device	Cable Connection to PCB
Intel FPGA Download Cable II (formerly the USB-Blaster II)	JTAG, AS	10-pin female plug 3M Part number: 2510-6002UB
Intel FPGA Ethernet Cable (formerly the Ethernet Blaster II)	JTAG, AS	10-pin female plug

The Intel FPGAs and Programmable Devices / Download Cables provides more information about the download cables and includes links to the user guides for all cables listed in the table above.

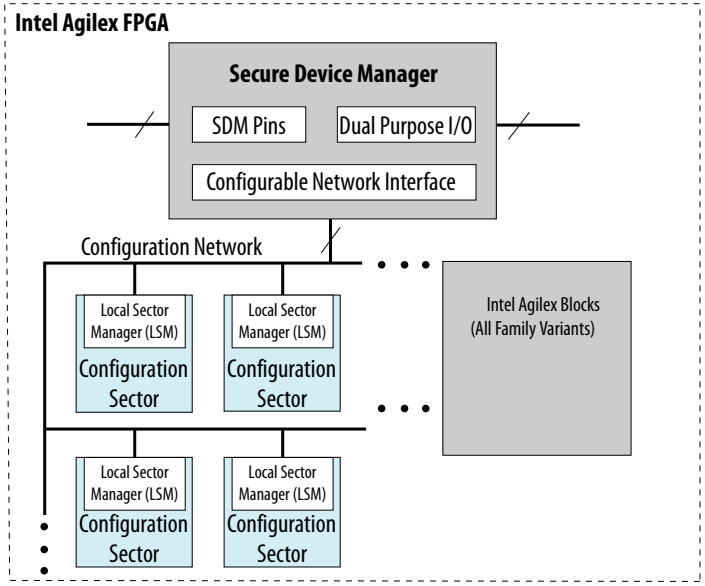
1.2. Intel Agilex Configuration Architecture

The Secure Device Manager (SDM) is a triple-redundant processor-based module that manages configuration and the security features of Intel Agilex devices. The SDM is available on all Intel Agilex FPGAs and SoC devices.

The block diagram below provides an overview of the Intel Agilex configuration architecture which includes the following blocks:

- SDM: More information about the SDM is contained in later sections.
- Configuration network: The SDM uses this dedicated, parallel configuration network to distribute the configuration bitstream to Local Sector Managers (LSMs). You cannot access this network.
- LSMs: The LSM is a microprocessor. Each configuration sector includes an LSM. The LSM parses configuration bitstream and configures the logic elements for its sector. After configuration, the LSM performs the following functions:
  - Monitors for single event upsets at the sector level
  - Processes responses to single event upsets (SEUs)
  - Performs hashing or integrity checks in real time
- Specific blocks for Intel Agilex variants:
  - E-Tile: Chip-to-chip high performance pulse width modulation (PAM4)
  - P-Tile: Supports PCIe Gen4

Figure 2. Intel Agilex Configuration Architecture Block Diagram



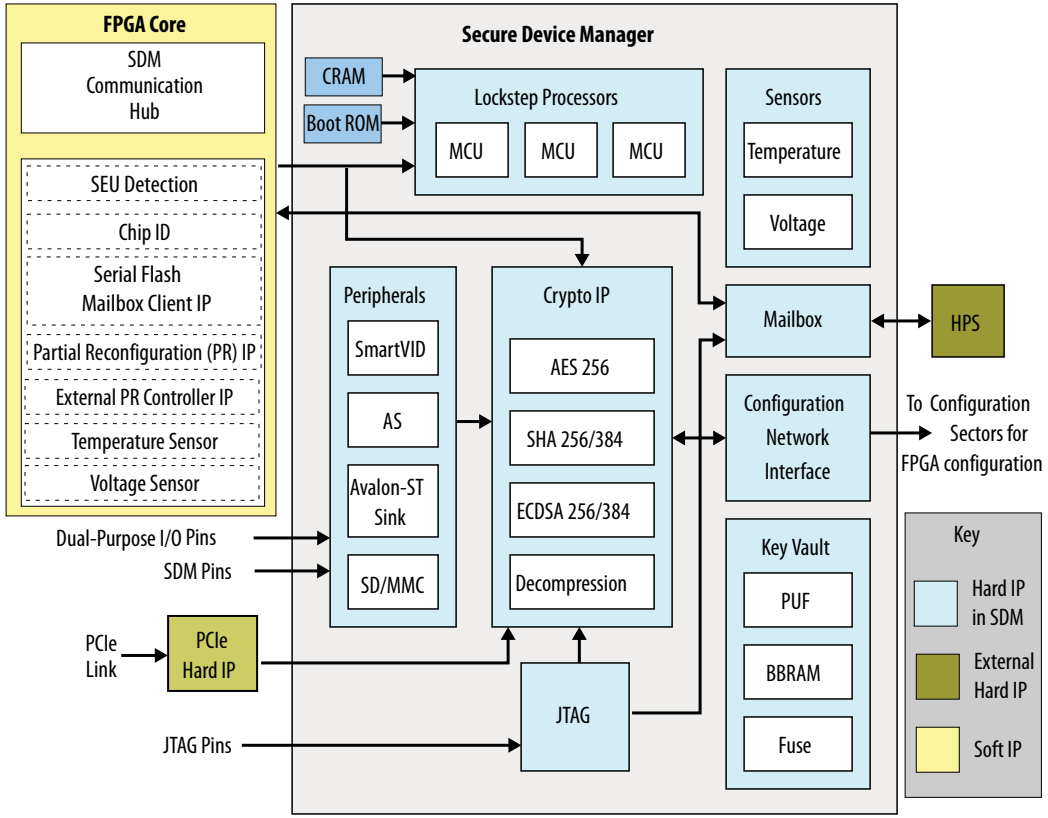
1.2.1. Secure Device Manager

The SDM comprises peripherals, cryptographic IP and sensors, boot ROM, triple-redundant lockstep processors, and other blocks shown the block diagram below. The SDM performs and manages the following security functions:

- Configuration bitstream authentication: After power-on during startup, the SDM triple-redundant lockstep processors run code from the boot ROM. The boot ROM code authenticates the Intel-generated configuration firmware and configuration bitstream, ensuring that configuration bitstream is from a trusted source. All Intel Agilex support authentication.
- Encryption: Encryption protects the configuration bitstream or confidential data from unauthorized third-party access.
- Side channel attack protection: Side channel attack protection guards AES Key and confidential data under non-intrusive attacks.
- Integrity checking: Integrity checking verifies that an accidental event has not corrupted the configuration bitstream. This function is active, even if you do not enable authentication.

These security features are available in Intel Agilex devices that support advanced security.

Figure 3. SDM Block Diagram



Here is an overview of the additional functions the SDM controls:

- The Power Management block consists of a voltage and temperature sensor which enables the SmartVID feature via an external PMBus voltage regulator when you select -V and -E devices.
- The AES/SHA and other Crypto Accelerator blocks implement secure configuration and boot.
- The AS and SD/MMC configuration flash controllers enable active configuration schemes via dedicated SDM pins.

- The x8 Avalon-ST configuration scheme uses SDM I/O pins. The x16 and x32 Avalon-ST configuration schemes use dedicated SDM I/O pins and dual-purpose I/O pins. Refer to the *SDM Pin Mapping* for more information.
- To reduce configuration file size and support smaller memory sizes, and enable faster configuration, the Intel Quartus Prime software compresses the configuration data. All Intel Agilex devices compress the configuration bitstream. You cannot disable this feature. If specify an encrypted configuration bitstream, the Intel Quartus Prime Pro Edition software compresses the configuration bitstream before encryption.
- A specific PCIe block included in the Intel Agilex device supports CvP.

Related Information

- SDM Pin Mapping on page 25
- Intel Agilex Device Feature Status Description
  For information about security features that are currently supported and security features that are planned to be supported in the future.

1.2.1.1. Updating the SDM Firmware

When you generate a configuration bitstream using the **File ➤ Programming File Generator** menu item, the bitstream assembler adds all firmware (including the SDM firmware) that matches the Intel Quartus Prime Pro Edition Release to the `.sof`.

Depending on the configuration scheme you specify the resulting file can be in any of the following formats:

- Raw Binary File, `.rbf`
- Programmer Object File, `.pof`
- JTAG Indirect Configuration, `.jic`
- Raw Programming Data, `.rpd`
- Jam*Standard Test and Programming Language (STAPL) STAPL, `.jam`
- Jam Byte Code, `.jbc`

Newer versions of the Intel Quartus Prime software typically include new or updated SDM features implemented in firmware. When regenerating your configuration bitstream, Intel recommends using the latest version of the Intel Quartus Prime Pro Edition Software which includes the latest firmware. You do not need to recompile your `.sof` to use the firmware from a newer version of the Intel Quartus Prime Pro Edition Software. You can simply regenerate your configuration bitstream with the new version of the **Programming File Generator**.

1.2.1.2. Specifying Boot Order for Intel Agilex SoC Devices

For Intel Agilex SoC devices you can specify the configuration order, choosing either the FPGA First or the Hard Processor System (HPS) First options. When you select the FPGA First option, the SDM fully configures the FPGA, then configures the HPS SDRAM pins, loads the HPS first stage boot loader (FSBL) and takes the HPS out of reset. In this mode the fabric begins functioning just before the HPS exits reset. This use guide defines a state when the FPGA is functional. Configuration and initialization are complete.

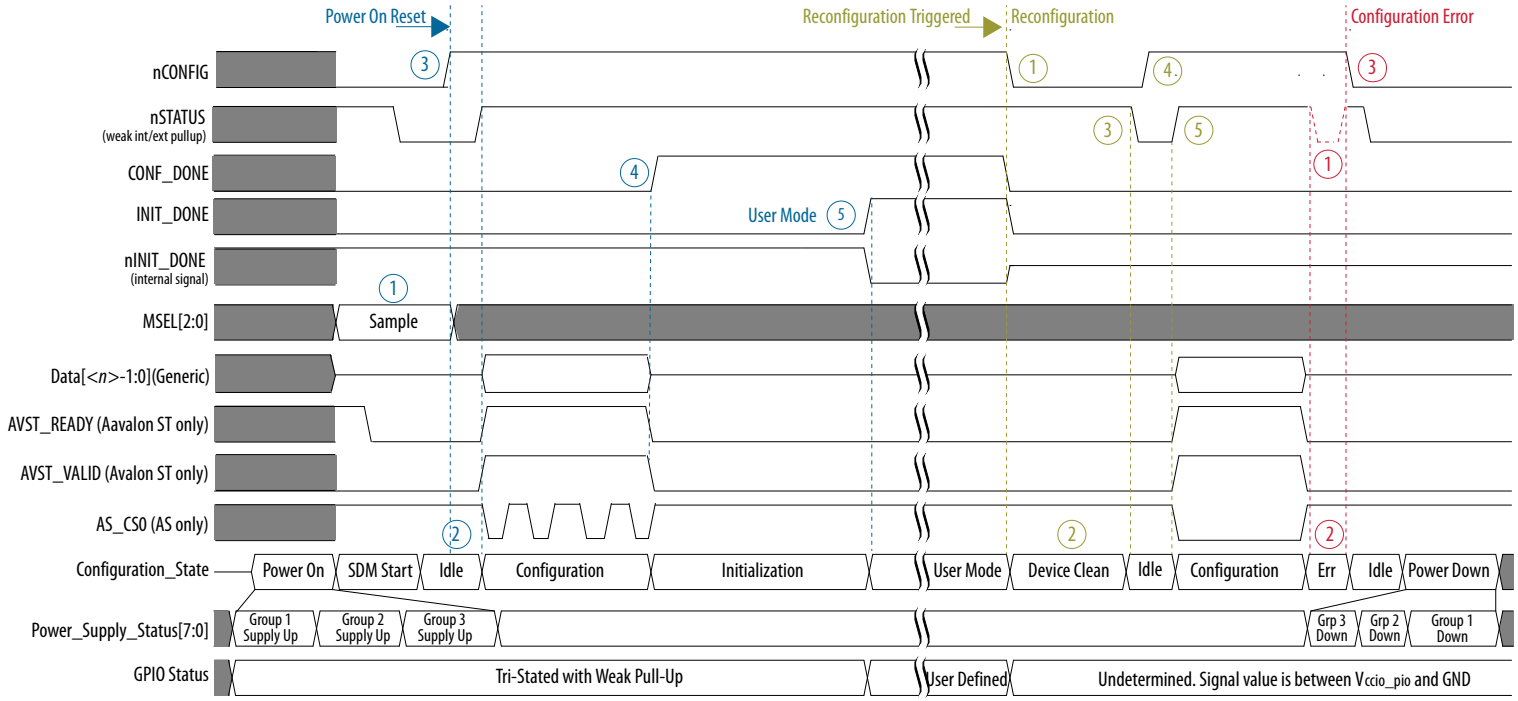
When you select the HPS First option, the SDM first configures the HPS SDRAM pins, loads the HPS FSBL and takes the HPS out of reset. Then the HPS configures the FPGA I/O and FPGA fabric at a later time. The HPS First option has the following advantages:

- Minimizes the amount of SDM flash memory required.
- Minimizes the amount of time it takes for the HPS software to be up and running.
- Supports FPGA reconfiguration while the HPS is running.

2. Intel Agilex Configuration Details

2.1. Intel Agilex Configuration Timing Diagram

Figure 4. Configuration, Reconfiguration, and Error Timing Diagram



The SDM drives Intel Agilex device configuration.

Initial Configuration Timing

The first section of the figure shows the expected timing for initial configuration after a normal power-on reset . Initially, the application logic drives the nCONFIG signal low (POR). Under normal conditions nSTATUS follows nCONFIG because nSTATUS reflects the current configuration state. nCONFIG must only change when it has the same value as nSTATUS.

When an error occurs, nSTATUS pulses low for approximately 1 ms and asserts high when the device is ready to accept reconfiguration.

The numbers in the *Initial Configuration* part of the timing diagram mark the following events:

1. The SDM boots up and samples the MSEL signals to determine the specified FPGA configuration scheme. The SDM does not sample the MSEL pins again until the next power cycle.
2. With the nCONFIG signal low, the SDM enters Idle mode after booting.

   Note: For Avalon-ST x16 and x|32 configuration schemes the host must drive nCONFIG low until it samples nSTATUS low. If the host fails to drive nCONFIG low until it samples nSTATUS low there is a chance that configuration may fail.
3. When the external host drives nCONFIG signal high, the SDM initiates configuration. The SDM drives the nSTATUS signal high, signaling the beginning of FPGA configuration. The SDM receives the configuration bitstream on the interface that the MSEL bus specified in *Step 1* The diagram shows AVST_READY and AVST_VALID continuously high. It is possible for AVST_READY to deassert which would require AVST_VALID to deassert within six cycles.
4. The SDM drives the CONF_DONE signal high, indicating the SDM received the bitstream successfully.
5. When the Intel Agilex device asserts INIT_DONE to indicate the FPGA has entered user mode. GPIO pins exit the high impedance state. The time between the assertion of CONF_DONE and INIT_DONE is variable. For FPGA First configuration, INIT_DONE asserts after initialization of the FPGA fabric, including registers and state machines. For HPS first configuration, the HPS application controls the time between CONF_DONE and INIT_DONE. INIT_DONE does not assert until after the software running on the HPS such as U-Boot or the operating system (OS) initiates the configuration, the FPGA configures and enters user mode..

   The entire device does not enter user mode simultaneously. Intel requires you to include the Including the Reset Release Intel FPGA IP in Your Design on page 118 in your design. Use the nINIT_DONE output of the Reset Release Intel FPGA IP to hold your application logic in the reset state until the entire FPGA fabric is in user mode. Failure to include this IP in your design may result in intermittent application logic failures.

Reconfiguration Timing

The second event the timing diagram illustrates the Intel Agilex device reconfiguration. If you change the MSEL setting after power-on, you must power-cycle the Intel Agilex. Power cycling forces the SDM to sample the MSEL pins before reconfiguring the device.

The numbers in the *Reconfiguration* part of the timing diagram mark the following events:

1. The external host drives nCONFIG signal low.
2. The SDM initiates device cleaning.
3. The SDM drives the nSTATUS signal low when device cleaning is complete.
4. The external host drives the nCONFIG signal high to initiate reconfiguration.
5. The SDM drives the nSTATUS signal high signaling the device is ready for reconfiguration and starts to reconfigure.

Configuration Error

The numbers in the *Configuration Error* part of the timing diagram mark the following events:

1. The SDM drives nSTATUS signal low for `1 ms -0.5 ms/+9.5 ms` to indicate a configuration error. The Intel Agilex device does not assert CONF_DONE indicating that configuration did not complete successfully.
2. The SDM enters the error state. During the error state, nCONFIG should be in the high state. The application must drive nCONFIG from high to low and then from low to high to restart configuration.
3. The SDM enters the idle state. The external host deasserts nCONFIG. The device is ready for reconfiguration by driving a low to high transition on nCONFIG. You can also power cycle the device by following the device power down sequence.

   Note: The nCONFIG signal can only change levels when it has the same value as nSTATUS. This restriction means that when nSTATUS = 1, nCONFIG can transition from 1 to 0. When nSTATUS = 0, nCONFIG can transition from 0 to 1. Apart from error reporting, nSTATUS only changes to follow nCONFIG.

Power Supply Status

The power-on reset (POR) holds the Intel Agilex device in the reset state until the power supply outputs are within the recommended operating range. t_{RAMP} defines the maximum power supply ramp time. If POR does not meet the t_{RAMP} time, the Intel Agilex device I/O pins and programming registers remain tri-stated.

For more information about POR refer to the *Intel Agilex Power Management User Guide.* For more information about t_{RAMP} refer to the *Intel Agilex datasheet*.

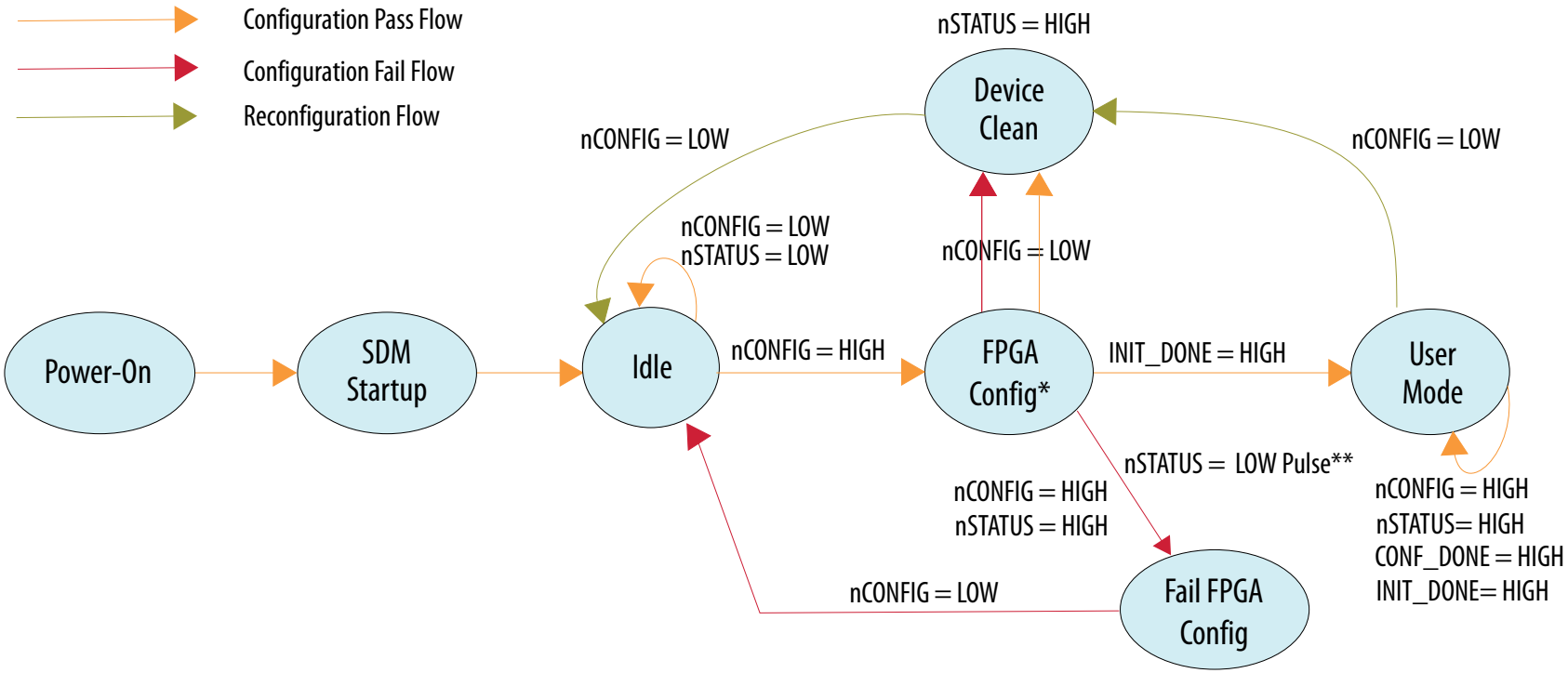
Related Information

- Standard (non-RSU) Flash Layout on page 151
- RSU Flash Layout – SDM Perspective on page 151
- Quad SPI Flash Layout on page 151
  For information about storing firmware, configuration, and application data in flash devices.
- Intel Agilex Device Datasheet
  For the following timing diagrams that define set-up, hold, and propagation delay timing parameters: *AS Configuration Serial Output Timing Diagram*, *AS Configuration Serial Input Timing Diagram*, and *Avalon ST Configuration Timing Diagram*.
- Intel Agilex Power Management User Guide

2.2. Configuration Flow Diagram

This topic describes the configuration flow for Intel Agilex devices.

Figure 5. Intel Agilex FPGA Configuration Flow



*FPGA first mode, fabric configuration begins immediately.
HPS first mode, HPS configures the fabric.

**minimum = 0.5 ms, maximum = 10.0 ms

Power Up

- The Intel Agilex power supplies power following the guidelines in the *Power-Up Sequence Requirements for Intel Agilex Devices* section of the *Intel Agilex Power Management User Guide*.
- A device-wide power-on reset (POR) asserts after the power supplies reach the correct operating voltages. The external power supply ramp must not be slower than the minimum ramping rate until the supplies reach the operating voltage.
- During configuration, internal circuitry pulls the SDM_IO0, SDM_IO8, and SDM_IO16 low internally. Internal circuitry pulls the remaining SDM_IO pins to a weak high.
- After POR, internal circuitry also pulls all GPIO pins to a weak high until the device enters user mode.

SDM Startup

- The SDM samples the MSEL pins during power-on.
- If MSEL is set to JTAG, the SDM remains in the Startup state.
- The SDM runs firmware stored in the on-chip boot ROM and enters the Idle state until the host drives nCONFIG high. The host should not drive nCONFIG high before all clocks are stable.

Idle

- The SDM remains in IDLE state until the external host initiates configuration by driving the nCONFIG pin from low to high. Alternatively, the SDM enters the idle state after it exits the error state.

Configuration Start

- After the SDM receives a configuration initiation request (nCONFIG = HIGH), the SDM signals the beginning of configuration by driving the nSTATUS pin high.
- Upon receiving configuration data, the SDM performs authentication, decryption and decompression.
- The nCONFIG pin remains high during configuration and in user mode. The host monitors the nSTATUS pin continuously for configuration errors.

Configuration Pass

- The SDM drives the CONF_DONE pin high after successfully receiving full bitstream.
- The CONF_DONE pin signals an external host that bitstream transfer is successful.

Configuration Error

- A low pulse on the nSTATUS pin indicates a configuration error.
- Errors require reconfiguration.
- After a low pulse indicating an error, configuration stops. The nSTATUS pin remains high.
- Following an error, the SDM drives nSTATUS low after the external host drives nCONFIG low.
- The device enters Idle state after the nSTATUS pin recovers to initial pre-configuration low state.

User Mode

- The SDM drives the INIT_DONE pin high after initializing internal registers and releases GPIO pins from the high impedance state. The device enters user mode. The entire device does not enter user mode simultaneously. Intel requires you to include the *Reset Release* in your design. Use the nINIT_DONE output of the Reset Release Intel FPGA IP to hold your application logic in the reset state until the entire FPGA fabric is in user mode. Failure to include this IP in your design may result in intermittent application logic failures.
- The nCONFIG pin should remain high in user mode.
- You may re-configure the device by driving nCONFIG pin from low to high.

Device Clean

- In the Device Clean state the design stops functioning.
- Device cleaning zeros out all configuration data.
- The Intel Agilex device drives CONF_DONE and INIT_DONE low.
- The SDM drives the nSTATUS pin low when device cleaning completes.

JTAG Configuration

Note: You can perform JTAG configuration anytime from any state except the power-on and SDM startup state. The Intel Agilex device cancels the previous configuration and accepts the reconfiguration data from the JTAG interface. The nCONFIG signal must be held in a stable state during JTAG configuration. A falling edge on the nCONFIG signal cancels the JTAG configuration.

Note: The SDM only samples the MSEL pins at power-on. The SDM drives nCONFIG high to initiate bitstream configuration using the configuration scheme you specified at power-on.

2.3. Additional Clock Requirements for HPS, PCIe, and HBM2

The Intel Agilex device has additional clock requirements for PCIe, HPS EMIF IP, eSRAM, and the High Bandwidth Memory (HBM2) IP.

To avoid configuration failures, the Intel Agilex device requires additional clocks for transceivers, PCIe, HPS EMIF IP, and all E-tile variants. You must provide a free-running, stable reference clock to these blocks before configuration begins. This reference clock is in addition to the configuration clock requirements for an internal or external oscillator described in OSC_CLK_1 Requirements on page 38.

These blocks and their specific clock names are as listed below.

- HBM2: `pll_ref_clk` and `ext_core_clk`
- HPS EMIF: `pll_ref_clk`
- E-tile transceivers: REFCLK_GXE
- P-tile transceivers: REFCLK_GXP

Note: The transceiver power supplies must be at nominal levels for successful configuration. You can use the V_{CC} and V_{CCP} power supplies for limited transceiver channel testing. Designs that include many transceivers require an auxiliary power supply to operate reliably.

2.4. Intel Agilex Configuration Pins

The Intel Agilex device uses SDM_IO pins for device configuration. Control of SDM I/O pins passes from internal FPGA circuitry, to the Boot ROM, and finally to the value your application logic specifies.

1. After power-on, SDM I/O pins 0, 8, and 16 have weak pull-downs. All other SDM I/O pins have weak pull-ups. (These initial voltage levels ensure correct operation during initialization. For example, for Avalon-ST configuration SDM_IO8 is the Avalon-ST ready signal which should not be asserted until the device reaches the FPGA Configuration state.)
2. The Boot ROM samples MSEL to determine the configuration scheme you specified and drives pins required for that configuration scheme. SDM I/O pins not required for the your configuration scheme remain weakly pulled up.
3. In approximately 10 ms the SDM I/O pins take on the state that your design specifies.
4. After device cleaning, the SDM reads pin information from firmware and restores the pin states that your design specifies. If you reconfigure the device, the SDM uses the updated pin information when initializing the device.

2.4.1. SDM Pin Mapping

You can use SDM I/O pins for configuration and other functions such as power management and SEU detection. You specify SDM I/O pin functions using the **Device ➤ Configuration ➤ Device and Pin Options** dialog box in the Intel Quartus Prime software. All SDM input signals include Schmitt triggers. All SDM outputs are open collector.

Fixed SDM I/O Pin Assignments for Avalon-ST x8 and AS x4

The Avalon-ST x8 and AS x4 configuration schemes use the dedicated SDM I/O pin assignments listed in in the table below. Use the assignments in this table for MSEL and AVSTx8_DATA0 to AVSTx8_DATA8 and AS x4.

Table 3. SDM Pin Mapping for Avalon-ST x8 and AS x4

SDM Pins	MSEL Function	Configuration Source Function	
		Avalon-ST x8	AS x4
SDM_IO0	—	—	—
SDM_IO1	—	AVSTx8_DATA2	AS_DATA1
SDM_IO2	—	AVSTx8_DATA0	AS_CLK
SDM_IO3	—	AVSTx8_DATA3	AS_DATA2
SDM_IO4	—	AVSTx8_DATA1	AS_DATA0
SDM_IO5	MSEL0	—	AS_nCSO
SDM_IO6	—	AVSTx8_DATA4	AS_DATA3
SDM_IO7	MSEL1	—	AS_nCS2
SDM_IO8	—	AVSTx8_READY	AS_nCS3
SDM_IO9	MSEL2	—	AS_nCS1
SDM_IO10	—	AVSTx8_DATA7	—
SDM_IO11	—	AVSTx8_VALID	—
SDM_IO12	—	—	—
SDM_IO13	—	AVSTx8_DATA5	—

continued...

SDM Pins	MSEL Function	Configuration Source Function	
		Avalon-ST x8	AS x4
SDM_IO14	—	AVSTx8_CLK	—
SDM_IO15	—	AVSTx8_DATA6	—
SDM_IO16	—	—	—

2.4.2. MSEL Settings

After power-on MSEL[2:0] pins specify the configuration scheme for Intel Agilex devices. Use 4.7-kΩ resistors to pull the MSEL[2:0] pins up to V_{CCIO_SDM} or down to ground as required by the MSEL[2:0] setting for your configuration scheme.

Figure 6. MSEL Pull-Up and Pull-Down Circuit Diagram

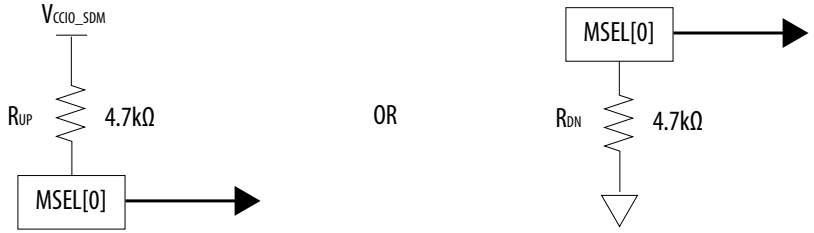


Table 4. MSEL Settings for Each Configuration Scheme of Intel Agilex Devices

Configuration Scheme	MSEL[2:0]
Avalon-ST (x32)	000
Avalon-ST (x16)	101
Avalon-ST (x8)	110
AS (Fast mode – for CvP)(3)	001

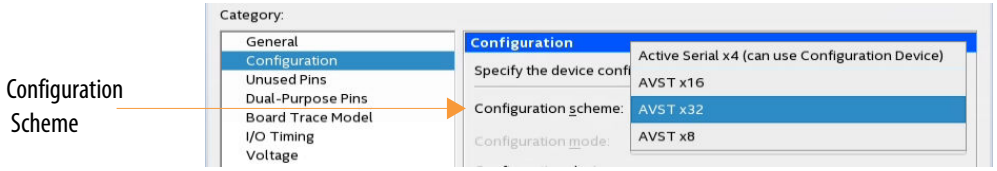
continued...

(3) If you use AS Fast mode and are not concerned about 100 ms PCIe linkup, you must still ramp the V_{CCIO_SDM} supply within 18 ms. This ramp-up requirement ensures that the AS x4 device is within its operating voltage range when the Intel Agilex device begins to access it.

Configuration Scheme	MSEL[2:0]
AS (Normal mode)	011
SD/MMC x4/x8	100
JTAG only(4)	111

You must also specify the configuration scheme on the **Configuration** page of the **Device and Pin Options** dialog box in the Intel Quartus Prime Software.

Figure 7. Specify Configuration Scheme to Specify MSEL Value



2.4.3. Device Configuration Pins for Optional Configuration Signals

All configuration schemes use the same dedicated pins for the standard control signals shown in the *Intel Agilex Configuration Timing Diagram*. Many other optional configuration signals do not have dedicated pin assignments.

Device Configuration Pins without Fixed Assignments

Note: Although the CONF_DONE and INIT_DONE configuration signals are not required, Intel recommends that you use these signals as an indicator to ensure that configuration is successful. The SDM drives the CONF_DONE signal high after successfully receiving full bitstream. The SDM drives the INIT_DONE signal high to indicate the device is fully in user mode. These signals are important when debugging configuration.

(4) JTAG configuration works with any valid MSEL settings, unless disabled for security.

Table 5. Available SDM I/O Pin Assignments for Configuration Signals that Do Not Use Dedicated SDM I/O Pins

Signal Names	Configuration Scheme			
	Avalon-ST			AS x4
	x8	x16	x32	
PWRMGT_SCL	SDM_IO0	SDM_IO0 SDM_IO14	SDM_IO0 SDM_IO14	SDM_IO0 SDM_IO14
PWRMGT_SDA	SDM_IO12 SDM_IO16	SDM_IO11 SDM_IO12 SDM_IO16	SDM_IO11 SDM_IO12 SDM_IO16	SDM_IO11 SDM_IO12 SDM_IO16
PWRMGT_ALERT	SDM_IO0 SDM_IO9 SDM_IO12	SDM_IO0 SDM_IO9 SDM_IO12	SDM_IO0 SDM_IO12	SDM_IO0 SDM_IO12
CONF_DONE	SDM_IO0 SDM_IO5 SDM_IO12 SDM_IO16	SDM_IO0 SDM_IO1 SDM_IO2 SDM_IO3 SDM_IO4 SDM_IO5 SDM_IO6 SDM_IO7 SDM_IO10 SDM_IO11 SDM_IO12 SDM_IO13 SDM_IO14 SDM_IO15 SDM_IO16	SDM_IO0 SDM_IO1 SDM_IO2 SDM_IO3 SDM_IO4 SDM_IO5 SDM_IO6 SDM_IO7 SDM_IO9 SDM_IO10 SDM_IO11 SDM_IO12 SDM_IO13 SDM_IO14 SDM_IO15 SDM_IO16	SDM_IO0 SDM_IO10 SDM_IO11 SDM_IO12 SDM_IO13 SDM_IO14 SDM_IO15 SDM_IO16
INIT_DONE	SDM_IO0 SDM_IO5 SDM_IO12 SDM_IO16	SDM_IO0 SDM_IO1 SDM_IO2 SDM_IO3 SDM_IO4	SDM_IO0 SDM_IO1 SDM_IO2 SDM_IO3 SDM_IO4	SDM_IO0 SDM_IO10 SDM_IO11 SDM_IO12 SDM_IO13

continued...

Signal Names	Configuration Scheme			
	Avalon-ST			AS x4
	x8	x16	x32	
		SDM_IO5 SDM_IO6 SDM_IO7 SDM_IO9 SDM_IO10 SDM_IO11 SDM_IO12 SDM_IO13 SDM_IO14 SDM_IO15 SDM_IO16	SDM_IO5 SDM_IO6 SDM_IO7 SDM_IO9 SDM_IO10 SDM_IO11 SDM_IO12 SDM_IO13 SDM_IO14 SDM_IO15 SDM_IO16	SDM_IO14 SDM_IO15 SDM_IO16
CVP_CONFDONE	Not supported	Not supported	Not supported	SDM_IO0 SDM_IO10 SDM_IO11 SDM_IO12 SDM_IO13 SDM_IO14 SDM_IO15 SDM_IO16
SEU_ERROR	SDM_IO0 SDM_IO5 SDM_IO7 SDM_IO9 SDM_IO12 SDM_IO16	SDM_IO0 SDM_IO1 SDM_IO2 SDM_IO3 SDM_IO4 SDM_IO5 SDM_IO6 SDM_IO7 SDM_IO9 SDM_IO10 SDM_IO11 SDM_IO12	SDM_IO0 SDM_IO1 SDM_IO2 SDM_IO3 SDM_IO4 SDM_IO5 SDM_IO6 SDM_IO7 SDM_IO9 SDM_IO10 SDM_IO11 SDM_IO12	SDM_IO0 SDM_IO10 SDM_IO11 SDM_IO12 SDM_IO13 SDM_IO14 SDM_IO15 SDM_IO16

continued...

Signal Names	Configuration Scheme			
	Avalon-ST			AS x4
	x8	x16	x32	
		SDM_IO13 SDM_IO14 SDM_IO15 SDM_IO16	SDM_IO13 SDM_IO14 SDM_IO15 SDM_IO16	
HPS_COLD_nRESET	SDM_IO0 SDM_IO5 SDM_IO7 SDM_IO9 SDM_IO12 SDM_IO16	SDM_IO0 SDM_IO1 SDM_IO2 SDM_IO3 SDM_IO4 SDM_IO5 SDM_IO6 SDM_IO7 SDM_IO9 SDM_IO10 SDM_IO11 SDM_IO12 SDM_IO13 SDM_IO14 SDM_IO15 SDM_IO16	SDM_IO0 SDM_IO1 SDM_IO2 SDM_IO3 SDM_IO4 SDM_IO5 SDM_IO6 SDM_IO7 SDM_IO9 SDM_IO10 SDM_IO11 SDM_IO12 SDM_IO13 SDM_IO14 SDM_IO15 SDM_IO16	SDM_IO0 SDM_IO10 SDM_IO11 SDM_IO12 SDM_IO13 SDM_IO14 SDM_IO15 SDM_IO16
Direct to Factory Image	Not applicable	Not applicable	Not applicable	SDM_IO0 SDM_IO10 SDM_IO11 SDM_IO12 SDM_IO13 SDM_IO14 SDM_IO15 SDM_IO16
DATA UNLOCK	SDM_IO0 SDM_IO5	SDM_IO0 SDM_IO1	SDM_IO0 SDM_IO1	SDM_IO0 SDM_IO10

continued...

Signal Names	Configuration Scheme			
	Avalon-ST			AS x4
	x8	x16	x32	
	SDM_IO7 SDM_IO9 SDM_IO12 SDM_IO16	SDM_IO2 SDM_IO3 SDM_IO4 SDM_IO5 SDM_IO6 SDM_IO7 SDM_IO9 SDM_IO10 SDM_IO11 SDM_IO12 SDM_IO13 SDM_IO14 SDM_IO15 SDM_IO16	SDM_IO2 SDM_IO3 SDM_IO4 SDM_IO5 SDM_IO6 SDM_IO7 SDM_IO9 SDM_IO10 SDM_IO11 SDM_IO12 SDM_IO13 SDM_IO14 SDM_IO15 SDM_IO16	SDM_IO11 SDM_IO12 SDM_IO13 SDM_IO14 SDM_IO15 SDM_IO16
nCATTRIP	SDM_IO0 SDM_IO5 SDM_IO7 SDM_IO9 SDM_IO12 SDM_IO16	SDM_IO0 SDM_IO1 SDM_IO2 SDM_IO3 SDM_IO4 SDM_IO5 SDM_IO6 SDM_IO7 SDM_IO9 SDM_IO10 SDM_IO11 SDM_IO12 SDM_IO13 SDM_IO14 SDM_IO15 SDM_IO16	SDM_IO0 SDM_IO1 SDM_IO2 SDM_IO3 SDM_IO4 SDM_IO5 SDM_IO6 SDM_IO7 SDM_IO9 SDM_IO10 SDM_IO11 SDM_IO12 SDM_IO13 SDM_IO14 SDM_IO15 SDM_IO16	SDM_IO0 SDM_IO10 SDM_IO11 SDM_IO12 SDM_IO13 SDM_IO14 SDM_IO15 SDM_IO16

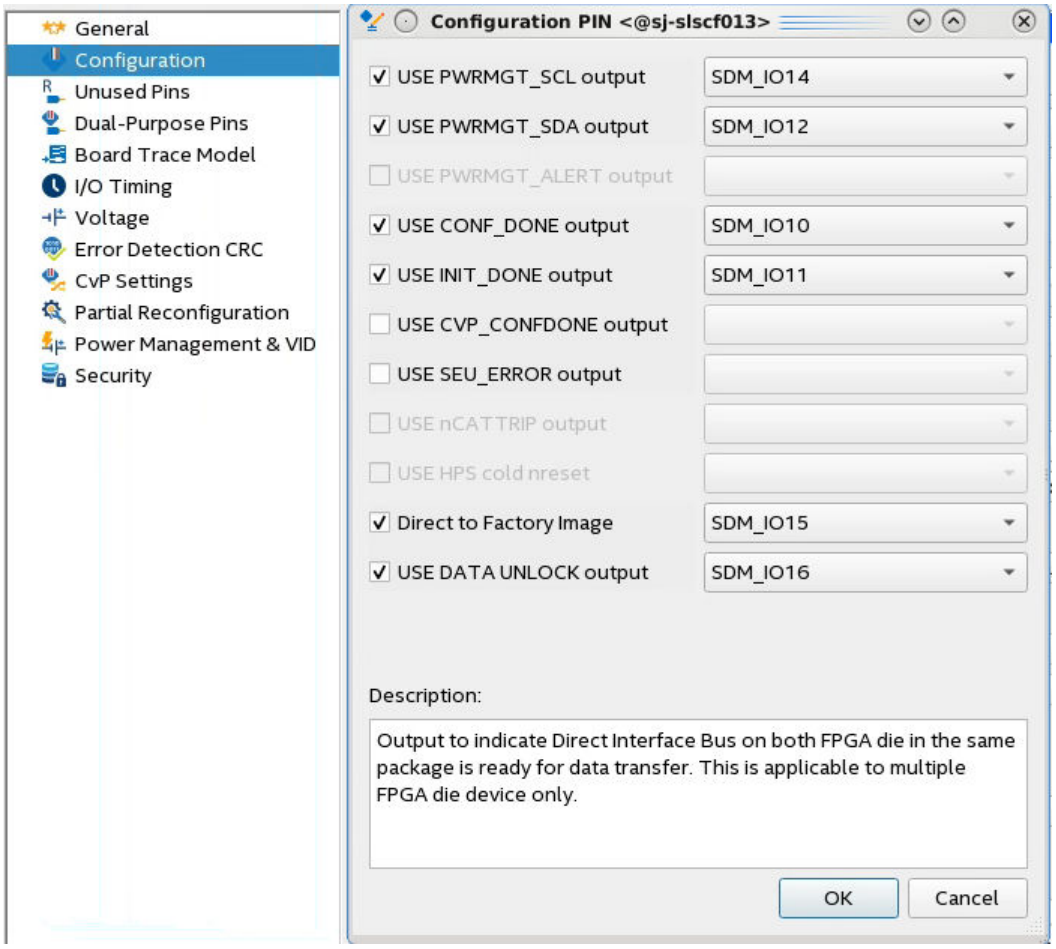
Note: Intel recommends that you assign the CONF_DONE and INIT_DONE pins to SDM I/O pins 0 or 16. These pins have weak internal pull-downs resistors. If you cannot use these pins, Intel recommends that you include external 4.7-kΩ pull-down resistors to avoid false signaling.

2.4.3.1. Specifying Optional Configuration Pins

You enable and assign the SDM I/O pins using the Intel Quartus Prime software.

Complete the following steps to assign these additional configuration pins:

1. On the **Assignments** menu, click **Device**.
2. In the **Device and Pin Options** dialog box, select the **Configuration** category and click **Configuration Pins Options**.
3. In the **Configuration Pin** window, enable and assign the configuration pin that you want to include in your design.



4. Click **OK** to confirm and close the **Configuration Pin** dialog box.

2.4.3.2. Enabling Dual-Purpose Pins

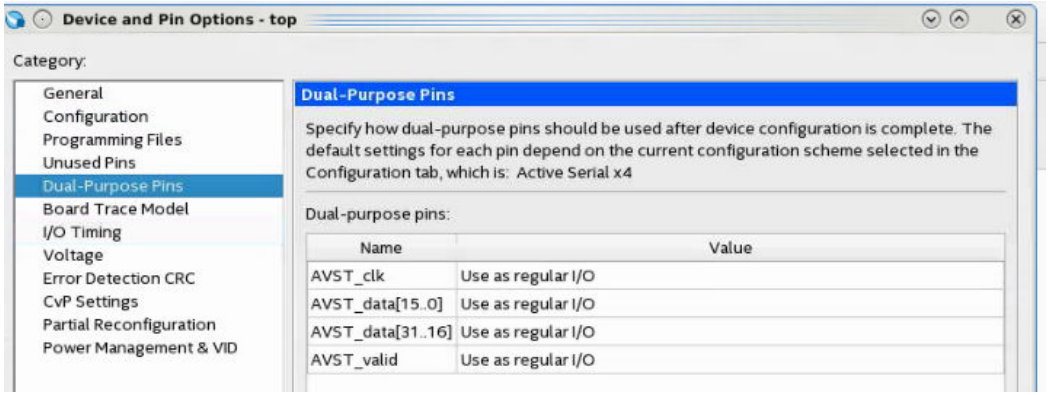
AVST_CLK, AVST_DATA[15:0], AVST_DATA[31:16], AVST_READY, and AVST_VALID are dual-purpose pins. Once the device enters user mode these pins can function either as GPIOs or as tri-state inputs.

If you use these pins as GPIOs, make the following assignments:

- Set V_{CCIO} of the I/O bank at 1.2 V
- Assign the 1.2 V I/O standard to these pins

Complete the following steps to assign these settings to the dual-purpose pins:

1. On the **Assignments** menu, click **Device**.
2. In the **Device and Pin Options** dialog box, select the **Dual-Purpose Pins** category.
3. In the **Dual-purpose pins** table, set the pin functionality in the **Value** column.



4. Click **OK** to confirm and close the **Device and Pin Options**

   Attention: When you use the Avalon ST configuration scheme the dual-purpose Avalon ST pins have the following restrictions:

   - You cannot use the Avalon-ST interface for partial reconfiguration (PR).
   - You cannot use the Avalon-ST pins in user mode in designs that include the HPS. This restriction means that you cannot use the Avalon-ST as dual-purpose I/Os in designs that include the HPS.

2.4.3.3. Configuration Pins I/O Standard, Drive Strength, and IBIS Model

Table 6. Intel Agilex Configuration Pins I/O Standard, Drive Strength, and IBIS Model

Configuration Pin Function	Location	Direction	I/O Standard	Drive Strength (mA)	Weak Pull-Up/Pull-Down
TDO	SDM I/O bank	Output	1.8 V LVCMOS	8	Weak pull-up
TMS	SDM I/O bank	Input	Schmitt Trigger Input	—	Weak pull-up
TCK	SDM I/O bank	Input	Schmitt Trigger Input	—	Weak pull-down
TDI	SDM I/O bank	Input	Schmitt Trigger Input	—	Weak pull-up
nSTATUS	SDM I/O bank	Output	1.8V LVCMOS	8	Weak pull-up
OSC_CLK_1	SDM I/O bank	Input	Schmitt Trigger Input	—	Weak pull-down
nCONFIG	SDM I/O bank	Input	Schmitt Trigger Input	—	Weak pull-up
SDM_IO[0],SDM_IO[8], SDM_IO[16]	SDM I/O bank	I/O	Schmitt Trigger Input or 1.8 V LVCMOS	8	Weak pull-down
SDM_IO[7:1],SDM_IO[15:9]	SDM I/O bank	I/O	Schmitt Trigger Input or 1.8 V LVCMOS	—	Weak pull-up
AVST_CLK	SDM Shared GPIO bank	Input	1.2 V LVCMOS	—	—
AVST_READY	SDM Shared GPIO bank	Output	1.2 V LVCMOS	Series 34 ohm on-chip termination (OCT) without calibration(5)	—
AVST_VALID	SDM Shared GPIO bank	Input	1.2 V LVCMOS	—	—
AVST_DATA	SDM Shared GPIO bank	Input	1.2 V LVCMOS	—	—

You can download the IBIS models from the *IBIS Models for Intel Devices* web page. The Intel Quartus Prime software does not support IBIS model generation for configuration pins in the current release.

(5) Slow slew rate signal

Unused SDM Pins

You can specify other functions on unused SDM pins in the Intel Quartus Prime software.

Related Information

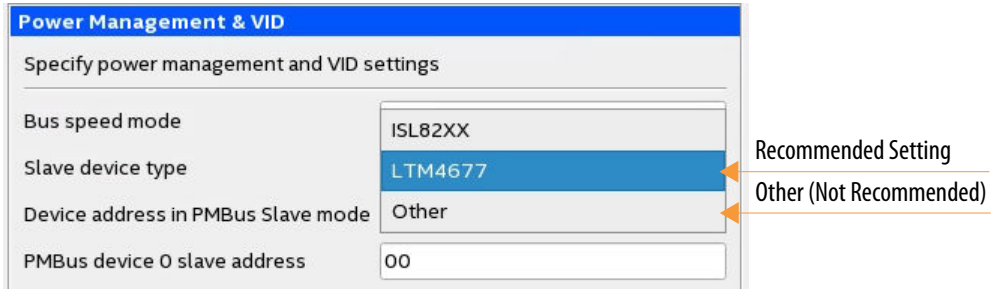
IBIS Models for Intel Devices

2.4.3.4. SDM I/O Pins for Power Management and SmartVID

SDM pins are also available for the SmartVID power management feature for -V and -E devices.

Intel recommends that you use the Analog Devices LTM4677 Dual 18A or Single 36A µModule Regulator with Digital Power System Management to regulate the PMBus. The LTM4677 device is the default setting for the **Device ➤ Device and Pin Options ➤ Power Management & VID ➤ Slave device type** parameter. If you are using a different PMBus regular change the default setting from **LTM4677** to **Other**.

Figure 8. Specifying the Slave Device Type for Power Management and VID



Refer to the *Intel Agilex Power Management User Guide* for more information about the pin assignments and PMBus setting.

Related Information

Intel Agilex Power Management User Guide

2.4.3.5. Specifying Pins for Partial Reconfiguration (PR)

The partial reconfiguration signals use GPIO pins.

The following signals control partial reconfiguration in Intel Agilex devices:

- PR_REQUEST
- PR_READY
- PR_ERROR
- PR_DONE

Connect these partial reconfiguration signals to the Partial Reconfiguration External Configuration Controller Intel FPGA IP.

Related Information

Creating a Partial Reconfiguration Design

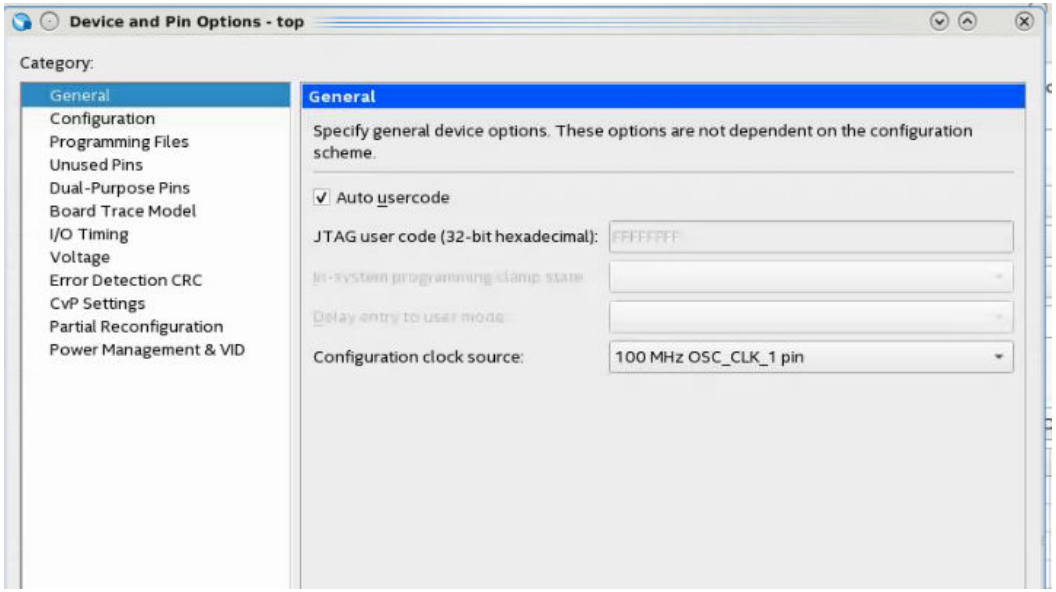
2.5. Configuration Clocks

2.5.1. Setting Configuration Clock Source

You must specify the configuration clock source by selecting either the internal oscillator or OSC_CLK_1 with the supported frequency. By default, the SDM uses the internal oscillator for device configuration. Specify an OSC_CLK_1 clock source for the fastest configuration time.

Complete the following steps to select the configuration clock source:

1. To specify OSC_CLK_1 as the clock source, on the **Assignments** menu, click **Device**.
2. In the **Device and Pin Options** dialog box, select the **General** category.
3. Specify the configuration clock source from the **Configuration clock source** drop down menu.



4. Click **OK** to confirm and close the **Device and Pin Options**.

Related Information

OSC_CLK_1 Clock Input on page 38

2.5.2. OSC_CLK_1 Clock Input

OSC_CLK_1 Requirements

When you drive the OSC_CLK_1 input clock with an external clock source and enable OSC_CLK_1 in the Intel Quartus Prime software, the device loads the majority of the configuration bitstream at 250 MHz. Intel Agilex devices include an internal oscillator in addition to OSC_CLK_1 which runs the configuration process at a frequency between 170-230 MHz. Intel Agilex devices always use this internal oscillator to load the first section of the bitstream, approximately 200 kilobyte (KB). The SDM can use either clock source for the remainder of device configuration. If you use the internal oscillator, you can leave the OSC_CLK_1 unconnected.

Note: Device configuration may fail under the following conditions when you select the OSC_CLK_1 as the clock source for configuration:

- You fail to drive the OSC_CLK_1 pin.
- You drive the OSC_CLK_1 pin at an incorrect frequency. Select one of the following input reference clock frequencies to drive the OSC_CLK_1 pin:
  - 25 MHz
  - 100 MHz
  - 125 MHz

The Intel Agilex device multiplies the OSC_CLK_1 source clock frequency to generate a 250 MHz clock for configuration. Using an OSC_CLK_1 source enables the fastest possible configuration. Refer to *Setting Configuration Clock Source* for instructions setting this frequency using the Intel Quartus Prime Software.

You can also specify this frequency by editing your `.qsf` file. Here are the possible assignments:

```
# EXTERNAL OSCILLATOR CLOCK  VIA OSC_CLK_1 PIN
set_global_assignment -name DEVICE_INITIALIZATION_CLOCK OSC_CLK_1_25MHZ
set_global_assignment -name DEVICE_INITIALIZATION_CLOCK OSC_CLK_1_100MHZ
set_global_assignment -name DEVICE_INITIALIZATION_CLOCK OSC_CLK_1_125MHZ
```

Configuration Clock Requirements for Reconfiguration Without Power Cycling the Device

When you specify OSC_CLK_1 for configuration and reconfigure without powering down the Intel Agilex device, the device can only reconfigure with OSC_CLK_1. In this scenario, OSC_CLK_1 must be a free-running clock.

Configuration Clock Requirements for Configuration After Powering Cycling the Device

After a power-down, when you specify OSC_CLK_1 for configuration, the Intel Agilex device uses the internal oscillator to load the first section of the bitstream and OSC_CLK_1 for the remainder.

Related Information

Setting Configuration Clock Source on page 37

3. Intel Agilex Configuration Schemes

3.1. Avalon-ST Configuration

The Avalon-ST configuration scheme replaces the FPP mode available in earlier device families. Avalon-ST is the fastest configuration scheme for Intel Agilex devices. This scheme uses an external host, such as a microprocessor, MAX® II, MAX V, or Intel MAX 10 device to drive configuration. The external host controls the transfer of configuration data from external storage such as flash memory to the FPGA. The logic that controls the configuration process resides in the external host. You can use the PFL II IP with a MAX II, MAX V, or Intel MAX 10 device as the host to read configuration data from the flash memory device and configure the Intel Agilex device. The Avalon-ST configuration scheme is called passive because the external host, not the Intel Agilex device, controls configuration.

Table 7. Avalon-ST Configuration Data Width, Clock Rates, and Data Rates

Mb is an abbreviation for Megabits per second.

Protocol	Data Width (bits)	Max Clock Rate	Max Data Rate	MSEL[2:0]
Avalon-ST	32	125 MHz	4000 Mb	000
	16	125 MHz	2000 Mb	101
	8	125 MHz	1000 Mb	110

Table 8. Required Configuration Signals for the Avalon-ST Configuration Scheme

You can use an 8-, 16-, or 32-bit Avalon-ST configuration data bus. You specify SDM I/O pin functions using the **Device ➤ Configuration ➤ Device and Pin Options** dialog box in the Intel Quartus Prime software. For the Avalon-ST x16 and x32 configuration, you can reassign the GPIO, dual-purpose configuration pins for other functions in user mode using the **Device ➤ Configuration ➤ Device and Pin Options ➤ Dual-Purpose Pins** dialog box.

Signal Name	Pin Type	Direction	Powered by
nSTATUS	SDM I/O	Output	V_{CCIO_SDM}
nCONFIG	SDM I/O	Input	V_{CCIO_SDM}

continued...

Signal Name	Pin Type	Direction	Powered by
MSEL[2:0]	SDM I/O, Dual-Purpose	Input	V_{CCIO_SDM}
CONF_DONE[6]	SDM I/O	Output	V_{CCIO_SDM}
AVSTx8_READY	SDM I/O	Output	V_{CCIO_SDM}
AVST_READY	GPIO, Dual-Purpose	Output	V_{CCIO}
AVSTx8_DATA[7:0]	SDM I/O	Input	V_{CCIO_SDM}
AVSTx8_VALID	SDM I/O	Input	V_{CCIO_SDM}
AVSTx8_CLK	SDM I/O	Input	V_{CCIO_SDM}
AVST_DATA[31:0]	GPIO, Dual-Purpose	Input	V_{CCIO}
AVST_VALID	GPIO, Dual-Purpose	Input	V_{CCIO}
AVST_CLK	GPIO, Dual-Purpose	Input	V_{CCIO}

Refer to the *Intel Agilex Data Sheet* for configuration timing estimates.

The x16 and x32 modes use GPIO pins that only support the 1.2 V I/O standard. The SDM I/O pins require a 1.8 V power supply. Consequently, you may need a voltage-level translation between the FPGA and external host because some signals, to accommodate both power requirements.

Note: Although the INIT_DONE configuration signal is not required for configuration, Intel recommends that you use this signals. The SDM drives the INIT_DONE signal high to indicate the device is fully in user mode. This signal is important when debugging configuration.

Note: If you create custom logic instead of using the PFL II IP to drive configuration, refer to the *Avalon Streaming Interfaces* in the *Avalon Interface Specifications* for protocol details.

Related Information

- Avalon Interface Specifications
- Intel Agilex Device Data Sheet

(6) CONF_DONE is required if you are using the Intel FPGA Parallel Flash Loader II IP as the configuration host.

3.1.1. Avalon-ST Configuration Scheme Hardware Components and File Types

You can use the following components to implement the Avalon-ST configuration scheme:

- A CPLD with PFL II IP and common flash interface (CFI) flash or Quad SPI flash memory
- A custom host, typically a microprocessor, with any external memory
- The Intel FPGA Download Cable II to connect the Intel Quartus Prime Programmer to the PCB.

The following block diagram illustrates the components and design flow using the Avalon-ST configuration scheme.

Figure 9. Components and Design Flow for .pof Programming

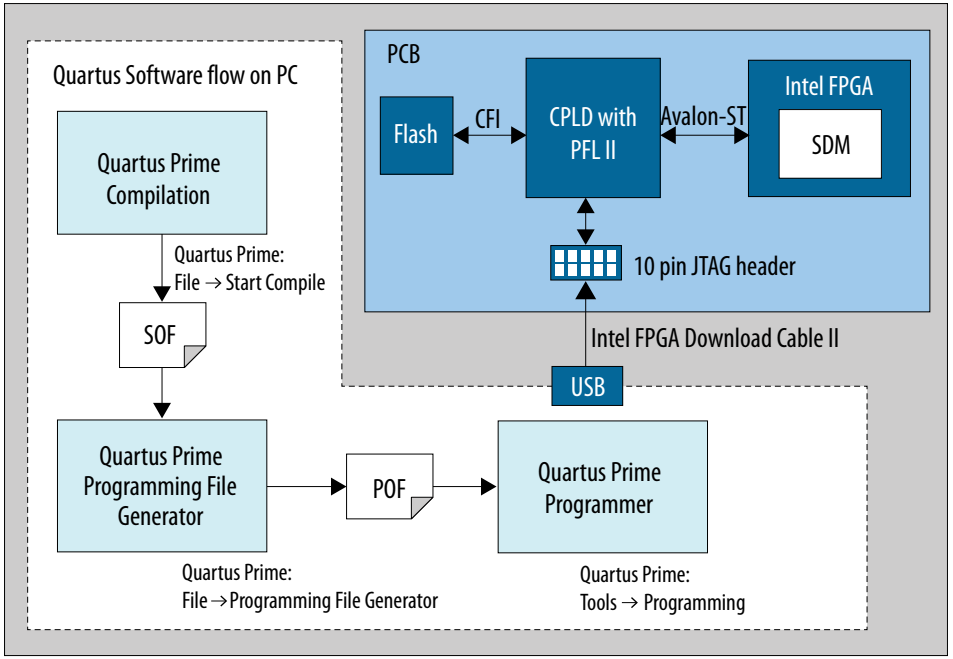


Table 9. Output File Types

Programming File Type	Extension	Description
Programmer Object File	`.pof`	The `.pof` is a proprietary Intel FPGA file type. Use the PFL II IP core via a JTAG header to write the `.pof` to an external CFI flash or serial flash device.
Raw Binary File	`.rbf`	You can also use the `.rbf` with the Avalon-ST configuration scheme and an external host such as a CPU or microcontroller. You can program the configuration bitstreams or data in the `.rbf` file directly into flash via a third-party programmer. Then, you can use an external host to configure the device with the Avalon-ST configuration scheme.

If you choose a third-party microprocessor for Avalon-ST configuration, refer to the *Avalon Streaming Interfaces* in the *Avalon Interface Specifications* for protocol details.

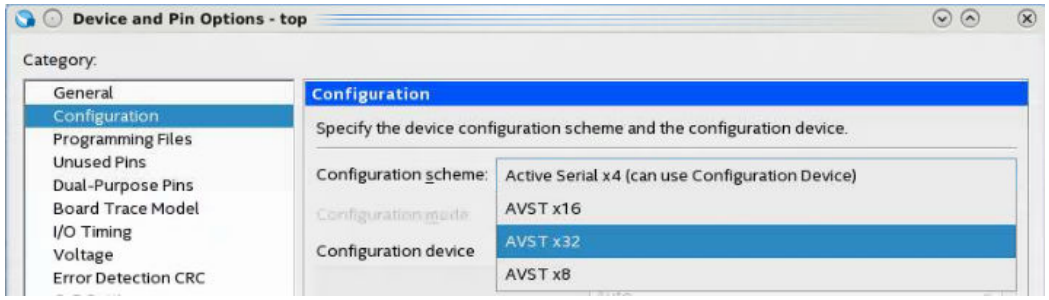
Note: Intel Agilex devices using Avalon ST x32 configuration and DDR x72 external memory interfaces are limited to a maximum of three memory interfaces. The Avalon ST x8 and x16 can support up to four DDR x72 external memory interfaces.

3.1.2. Enabling Avalon-ST Device Configuration

You enable the Avalon-ST device configuration scheme in the Intel Quartus Prime software.

Complete the following steps to specify an Avalon-ST interface for device configuration.

1. On the **Assignments** menu, click **Device**.
2. In the **Device and Pin Options** dialog box, select the **Configuration** category.
3. In the **Configuration** window, in the **Configuration scheme** dropdown list, select the appropriate Avalon-ST bus width.

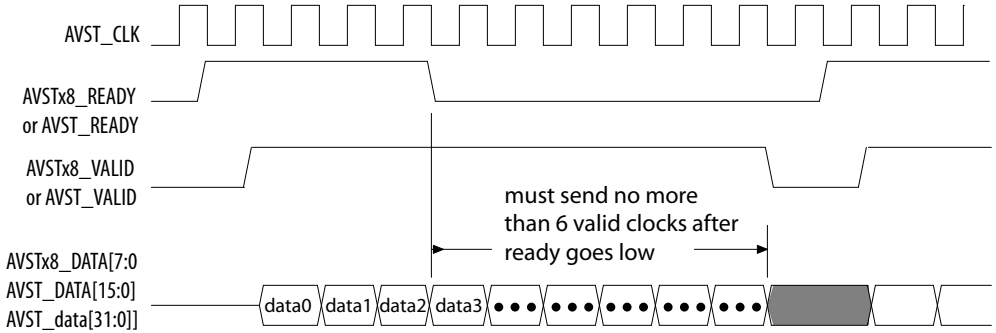


4. Click **OK** to confirm and close the **Device and Pin Options** dialog box.

3.1.3. The AVST_READY Signal

Before beginning configuration, trigger device cleaning by toggling the nCONFIG pin from high to low to high. This nCONFIG transition also returns the device to the configuration state.

Figure 10. Monitoring the AVST_READY Signal and Responding to Backpressure



The configuration files for Intel Agilex devices can be highly compressed. During configuration, the decompression of the bit stream inside the device requires the host to pause before sending more data. The Intel Agilex device asserts the AVST_READY signal when the device is ready to accept data. The AVST_READY signal is only valid when the nSTATUS pin is high. In addition, the host must handle backpressure by monitoring the AVST_READY signal and may assert AVST_VALID signal any time after the assertion of AVST_READY signal. The host must monitor the AVST_READY signal throughout the configuration.

Note: For Avalon-ST x16 and x32, after Power-On-Reset you must not send data to the device until it indicates it is ready using nSTATUS. You must drive nCONFIG low and wait for nSTATUS to go low. Next, you should drive nCONFIG high and wait for nSTATUS to go high. The device can starting sending data when AVST_READY asserts.

The AVST_READY signal sent by the Intel Agilex device to the host is not synchronized with the AVSTx8_CLK or AVST_CLK. To configure the Intel Agilex device successfully, the host must adhere to the following constraints:

- The host must drive no more than six data words after the deassertion of the AVST_READY signal including the delay incurred by the 2-stage register synchronizer.
- The host must synchronize the AVST_READY signal to the AVST_CLK signal using a 2-stage register synchronizer. Here is Register transfer level (RTL) example code for 2-stage register synchronizer:

```
always @(posedge avst_clk or negedge reset_n)
    begin
        if (~reset_n)
        begin
            fpga_avst_ready_reg1 <= 0;
            fpga_avst_ready_reg2 <= 0;
        else
            fpga_avst_ready_reg1 <= fpga_avst_ready;
            fpga_avst_ready_reg2 <= fpga_avst_ready_reg1;
        end
    end
```

  Where:

  — The AVST_CLK signal comes from either PFL II IP or your Avalon-ST controller logic.

  — `fpga_avst_ready` is the AVST_READY signal from the Intel Agilex device.

  — `fpga_avst_ready_reg2` signal is the AVST_READY signal that is synchronous to AVST_CLK.

You must properly constrain the AVST_CLK and AVST_DATA signals at the host. Perform timing analysis on both signals between the host and Intel Agilex device to ensure the Avalon-ST configuration timing specifications are met. Refer to the *Avalon-ST Configuration Timing* section of the *Intel Agilex Device Data Sheet* for information about the timing specifications.

Note: The AVST_CLK signal must run continuously during configuration. The AVST_READY signal cannot assert unless the clock is running.

Optionally, you can monitor the CONF_DONE signal to indicate the flash has sent all the data to FPGA or to indicate the configuration process is complete.

If you use the PFL II IP core as the configuration host, you can use the Intel Quartus Prime software to store the binary configuration data to the flash memory through the PFL II IP core.

If you use the Avalon-ST Adapter IP core as part of the configuration host, set the **Source Ready Latency** value between 1-6.

Avalon-ST x8 configuration scheme uses the SDM pins only. Avalon-ST x16 and x32 configuration scheme only use dual-purpose I/O pins that you can use as general-purpose I/O pins after configuration.

Related Information

- Avalon Interface Specifications
- Avalon-ST Configuration Timing
  For Avalon-ST Timing Parameters for Configuration in Intel Agilex Devices.

3.1.4. RBF Configuration File Format

If you do not use the Parallel Flash Loader II Intel FPGA IP core to program the flash, you must generate the `.rbf` file.

The data in `.rbf` file are in little-endian format

Table 10. Writing 32-bit Data

For a x32 data bus, the first byte in the file is the least significant byte of the configuration double word, and the fourth byte is the most significant byte.

Double Word = 01EE1B02			
LSB: BYTE0 = 02	BYTE1 = 1B	BYTE2 = EE	MSB: BYTE3 = 01
D[7:0]	D[15:8]	D[23:16]	D[31:24]
0000 0010	0001 1011	1110 1110	0000 0001

Table 11. Writing 16-bit Data

For a x16 data bus, the first byte in the file is the least significant byte of the configuration word, and the second byte is the most significant byte of the configuration word.

WORD0 = 1B02		WORD1 = 01EE	
LSB: BYTE0 = 02	MSB: BYTE1 = 1B	LSB: BYTE2 = EE	MSB: BYTE3 = 01
D[7:0]	D[15:8]	D[7:0]	D[15:8]
0000 0010	0001 1011	1110 1110	0000 0001

3.1.5. Avalon-ST Single-Device Configuration

Figure 11. Connections for Avalon-ST x8 Single-Device Configuration

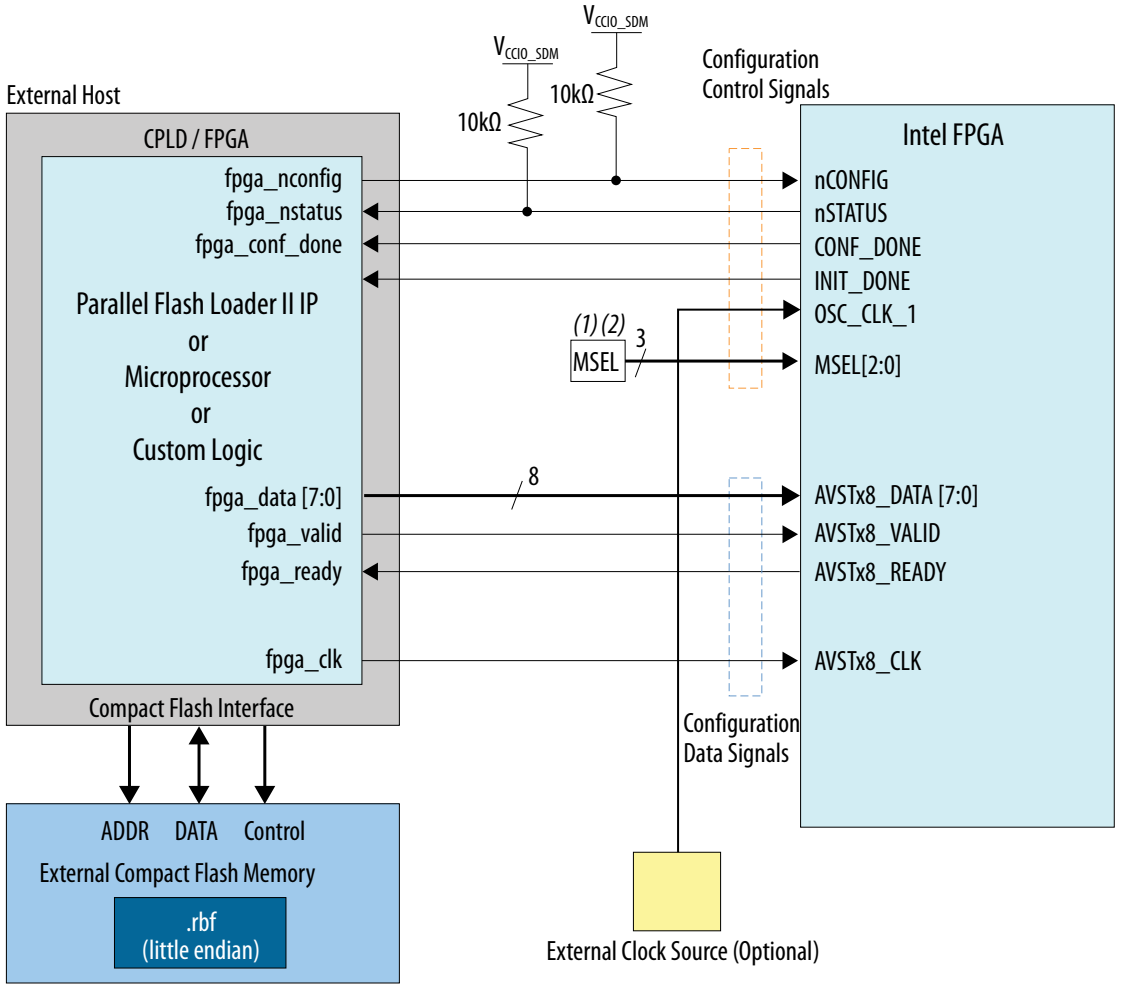


Figure 12. Connections for Avalon-ST x16 Single-Device Configuration

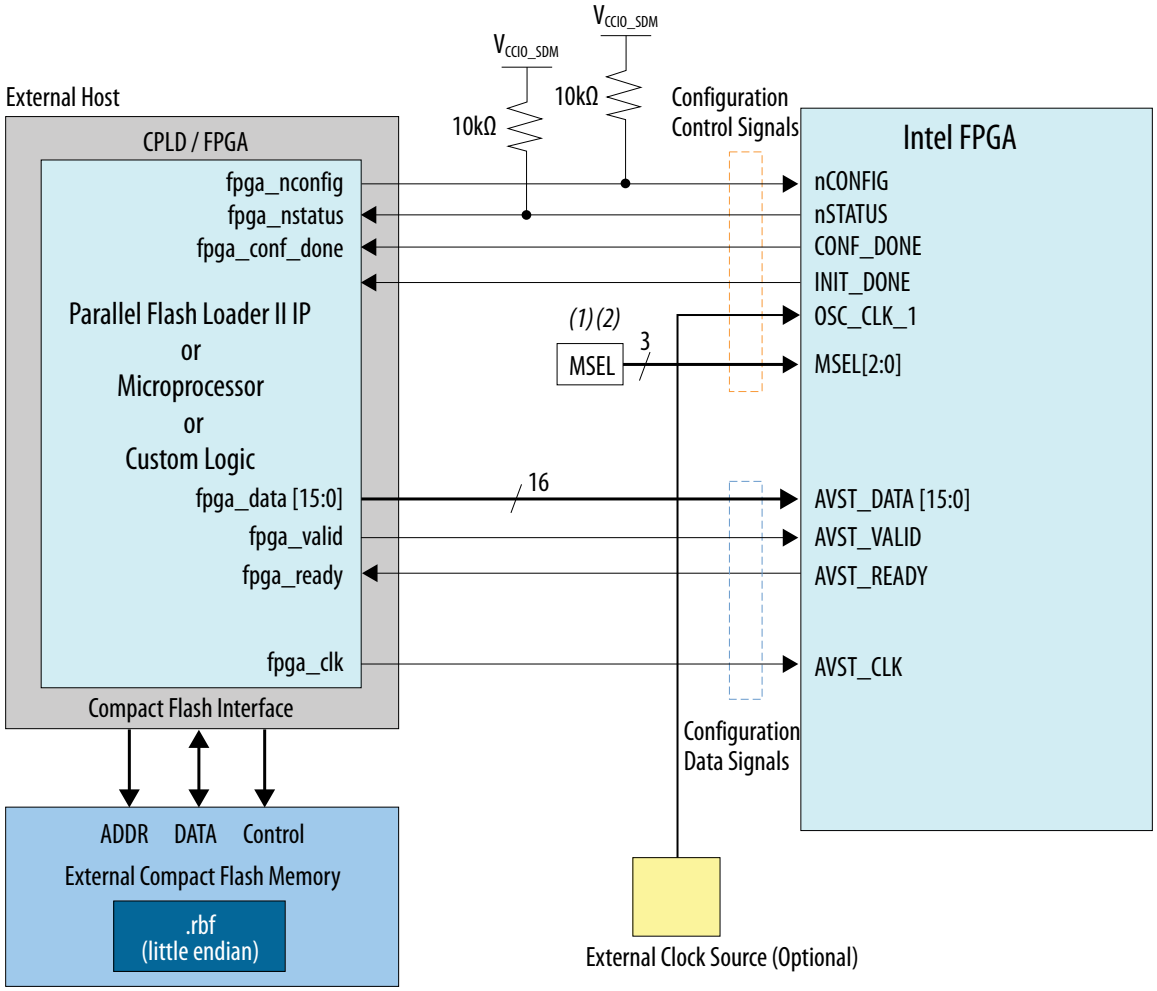
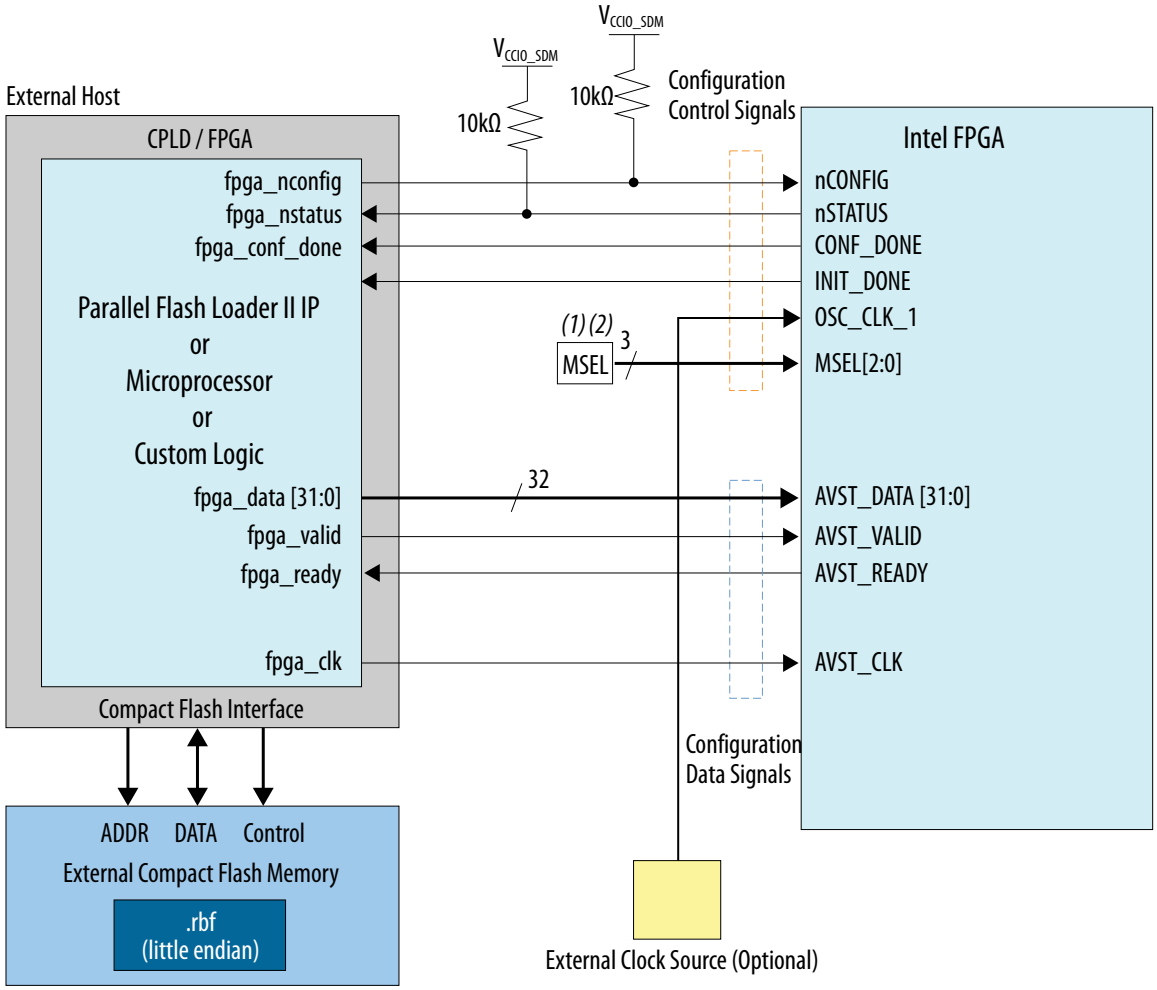


Figure 13. Connections for Avalon-ST x32 Single-Device Configuration



Notes for Figure:

1. Refer to *MSEL Settings* for the correct resistor pull-up and pull-down values for all configuration schemes.
2. The MSEL pins are dual-purpose. After power-on, you can reassign these pins to other functions. For more information, refer to *Enabling Dual Purpose Pins*
3. The synchronizers shown in all three figures can be internal if the host is an FPGA or CPLD. If the host is a microprocessor, you must use discrete synchronizers.

Related Information

MSEL Settings on page 26

3.1.6. Debugging Guidelines for the Avalon-ST Configuration Scheme

The Avalon-ST configuration scheme replaces the previously available in fast passive parallel (FPP) modes. This configuration scheme retains similar functionality and performance. Here are the important differences:

- The Avalon-ST configuration scheme requires you to monitor the flow control signal, AVST_READY. The AVST_READY signal indicates if the device can receive configuration data.
- The AVST_CLK and AVSTx8_CLK clock signals cannot pause when configuration data is not being transferred. Data is not transferred when AVST_READY and AVST_VALID are low. The AVST_CLK and AVSTx8_CLK clock signals must run continuously until CONF_DONE asserts.

Debugging Suggestions

Review the general *Configuration Debugging Checklist* in the *Debugging Guide* chapter before considering these debugging tips that pertain to the Avalon-ST configuration scheme.

- Only assert AVST_VALID after the SDM asserts AVST_READY.
- Only assert AVST_VALID when the AVST_DATA is valid.
- Ensure that the AVST_CLK clock signal is continuous and free running until configuration completes. The AVST_CLK can stop after CONF_DONE asserts. The initialization state does not require the AVST_CLK signal.
- If using x8 mode, ensure that you use the dedicated SDM_IO pins for this interface (clock, data, valid and ready).
- If using x16 or x32 mode, power the I/O bank containing the x16 or x32 pins (I/O Bank 3A) at 1.8 V.

- Ensure you select the appropriate Avalon-ST configuration scheme in your Intel Quartus Prime Pro Edition project.
- Ensure the MSEL pins reflect this mode on the PCB.
- Verify that host device does not drive configuration pins before the Intel Agilex device powers up.

Related Information

Configuration Debugging Checklist on page 185

3.1.7. QSF Assignments for Avalon-ST x8

You can specify many Intel Quartus Prime project settings using Intel Quartus Prime Software GUI or by editing the Intel Quartus Prime Settings File (`.qsf`). The following assignments in the `.qsf` show typical settings for a Intel Agilex device using Avalon-ST x8 configuration.

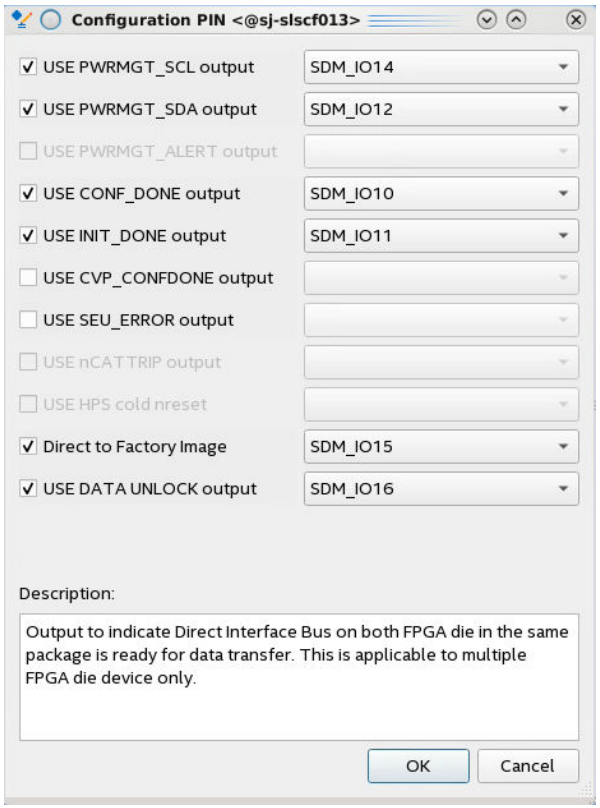
These settings are for a Intel Agilex SmartVID device operating in PMBus slave mode which requires most of the SDM_IO pins. Refer to the *Intel Agilex Power Management User Guide* for the PMBus constraints in master mode.

```
# Fitter Assignments
# ==================
set_global_assignment -name CONFIGURATION_VCCIO_LEVEL 1.8V

# SDM IO Assignments
# ==================
set_global_assignment -name USE_PWRMGT_SCL SDM_IO0
set_global_assignment -name USE_PWRMGT_SDA SDM_IO12
set_global_assignment -name USE_PWRMGT_ALERT SDM_IO9
set_global_assignment -name USE_CONF_DONE SDM_IO16
set_global_assignment -name USE_SEU_ERROR SDM_IO5

# Configuration settings
# ======================
# The following setting also supports Intel Agilex devices
set_global_assignment -name STRATIXV_CONFIGURATION_SCHEME "AVST X8"
set_global_assignment -name USE_CONFIGURATION_DEVICE OFF
set_global_assignment -name ERROR_CHECK_FREQUENCY_DIVISOR 256
set_global_assignment -name GENERATE_PR_RBF_FILE ON
set_global_assignment -name ENABLE_ED_CRC_CHECK ON
set_global_assignment -name MINIMUM_SEU_INTERVAL 479

# SmartVID feature PMBus settings [Slave mode settings only]
```

```
# ========================================================
set_global_assignment -name VID_OPERATION_MODE "PMBUS SLAVE"
set_global_assignment -name PWRMGT_DEVICE_ADDRESS_IN_PMBUS_SLAVE_MODE 3F
```

You can also set the SDM_IO configuration pins using the **Assignments ➤ Device ➤ Device and Pin Options ➤ Configuration ➤ Configuration Pin Options**.

Figure 14. Set SDM_IO Configuration Pins Using the Intel Quartus Prime Software



Related Information

- PMBus Master Mode
  In the *Intel Stratix® 10 Power Management User Guide*
- Intel Quartus Prime Pro Settings File Reference Manual

3.1.8. QSF Assignments for Avalon-ST x16

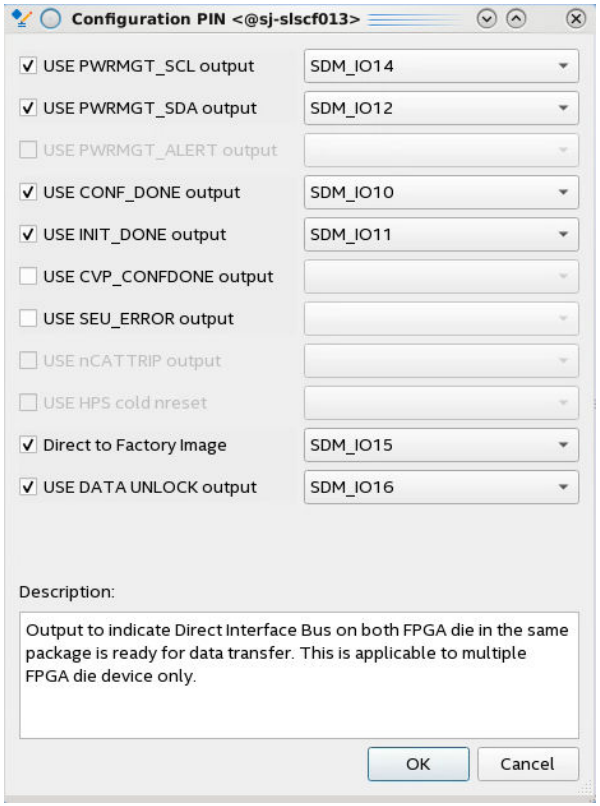
You can specify many Intel Quartus Prime project settings using Intel Quartus Prime Software GUI or by editing the Intel Quartus Prime Settings File (`.qsf`). The following assignments in the `.qsf` show typical settings for a Intel Agilex device using Avalon-ST x16 configuration.

These settings are for a Intel Agilex SmartVID device operating in PMBus slave mode which requires most of the SDM_IO pins. Refer to the *Intel Agilex Power Management User Guide* for the PMBus constraints in master mode.

```
# Fitter Assignments
# ==================
set_global_assignment -name CONFIGURATION_VCCIO_LEVEL 1.8V

# SDM IO Assignments
# ==================
set_global_assignment -name USE_PWRMGT_SCL SDM_IO0
set_global_assignment -name USE_PWRMGT_SDA SDM_IO12
set_global_assignment -name USE_PWRMGT_ALERT SDM_IO9
set_global_assignment -name USE_CONF_DONE SDM_IO16
set_global_assignment -name USE_SEU_ERROR SDM_IO5

# Configuration settings
# ======================
#
# The following setting also supports Intel Agilex devices
set_global_assignment -name STRATIXV_CONFIGURATION_SCHEME "AVST X16"
set_global_assignment -name USE_CONFIGURATION_DEVICE OFF
set_global_assignment -name ERROR_CHECK_FREQUENCY_DIVISOR 256
set_global_assignment -name GENERATE_PR_RBF_FILE ON
set_global_assignment -name ENABLE_ED_CRC_CHECK ON
set_global_assignment -name MINIMUM_SEU_INTERVAL 479

# SmartVID feature PMBus settings [Slave mode settings only]
# ==========================================================
set_global_assignment -name VID_OPERATION_MODE "PMBUS SLAVE"
set_global_assignment -name PWRMGT_DEVICE_ADDRESS_IN_PMBUS_SLAVE_MODE 3F
```

You can also set the SDM_IO configuration pins using the **Assignments ➤ Device ➤ Device and Pin Options ➤ Configuration ➤ Configuration Pin Options**.

Figure 15. Set SDM_IO Configuration Pins Using the Intel Quartus Prime Software



Related Information

- PMBus Master Mode
  In the *Intel Stratix® 10 Power Management User Guide*
- Intel Quartus Prime Pro Settings File Reference Manual

3.1.9. QSF Assignments for Avalon-ST x32

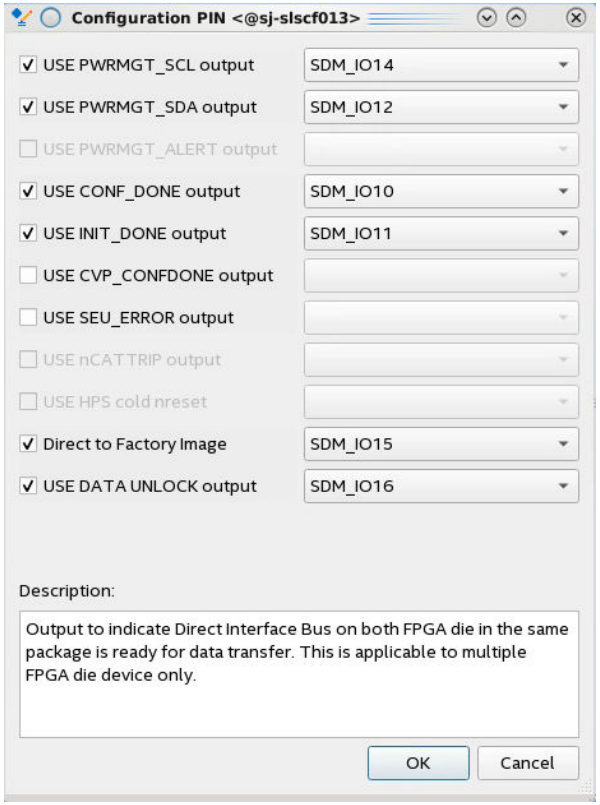
You can specify many Intel Quartus Prime project settings using Intel Quartus Prime Software GUI or by editing the Intel Quartus Prime Settings File (`.qsf`). The following assignments in the `.qsf` show typical settings for a Intel Agilex device using Avalon-ST x32 configuration.

These settings are for a Intel Agilex SmartVID device operating in PMBus slave mode which requires most of the SDM_IO pins. Refer to the *Intel Agilex Power Management User Guide* for the PMBus constraints in master mode.

```
# Fitter Assignments
# ==================
set_global_assignment -name CONFIGURATION_VCCIO_LEVEL 1.8V

# SDM IO Assignments
# ==================
set_global_assignment -name USE_PWRMGT_SCL SDM_IO14
set_global_assignment -name USE_PWRMGT_SDA SDM_IO16
set_global_assignment -name USE_PWRMGT_ALERT SDM_IO12
set_global_assignment -name USE_CONF_DONE SDM_IO5
set_global_assignment -name USE_INIT_DONE SDM_IO0
set_global_assignment -name USE_SEU_ERROR SDM_IO1

# Configuration settings
# ======================
#
# The following setting also supports Intel Agilex devices
set_global_assignment -name STRATIXV_CONFIGURATION_SCHEME "AVST X32"
set_global_assignment -name USE_CONFIGURATION_DEVICE OFF
set_global_assignment -name ERROR_CHECK_FREQUENCY_DIVISOR 256
set_global_assignment -name GENERATE_PR_RBF_FILE ON
set_global_assignment -name ENABLE_ED_CRC_CHECK ON
set_global_assignment -name MINIMUM_SEU_INTERVAL 479

# SmartVID feature PMBus settings [Slave mode settings only]
# ==========================================================
set_global_assignment -name VID_OPERATION_MODE "PMBUS SLAVE"
set_global_assignment -name PWRMGT_DEVICE_ADDRESS_IN_PMBUS_SLAVE_MODE 3F
```

You can also set the SDM_IO configuration pins using the **Assignments ➤ Device ➤ Device and Pin Options ➤ Configuration ➤ Configuration Pin Options**.

Figure 16. Set SDM_IO Configuration Pins Using the Intel Quartus Prime Software



Related Information

- PMBus Master Mode
  In the *Intel Stratix® 10 Power Management User Guide*
- Intel Quartus Prime Pro Settings File Reference Manual

3.1.10. IP for Use with the Avalon-ST Configuration Scheme: Intel FPGA Parallel Flash Loader II IP Core

3.1.10.1. Functional Description

You can use the Parallel Flash Loader II Intel FPGA IP (PFL II) with an external host, such as the MAX II, MAX V, or Intel MAX 10 devices to complete the following tasks:

- Program configuration data into a flash memory device using JTAG interface.
- Configure the Intel Agilex device with the Avalon-ST configuration scheme from the flash memory device.

  Note: Intel Agilex device configuration is not available in the current release.

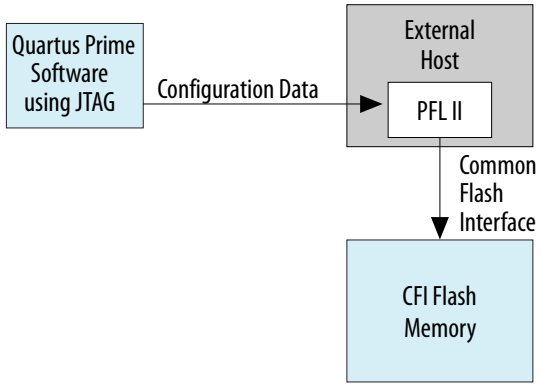
Note: Use the Parallel Flash Loader II IP Intel FPGA IP and not the earlier Parallel Flash Loader IP with the Avalon-ST configuration scheme in Intel Agilex devices.

3.1.10.1.1. Generating and Programming a .pof into CFI Flash

The Intel Quartus Prime software generates the `.sof` when you compile your design. You use the `.sof` to generate the `.pof`. This process includes the following steps:

1. Generating a `.pof` for the PFL II IP using the Intel Quartus Prime **File ➤ Programming File Generator**.
2. Using the Intel Quartus Prime Programmer to write the Intel Agilex device `.pof` to the flash device.

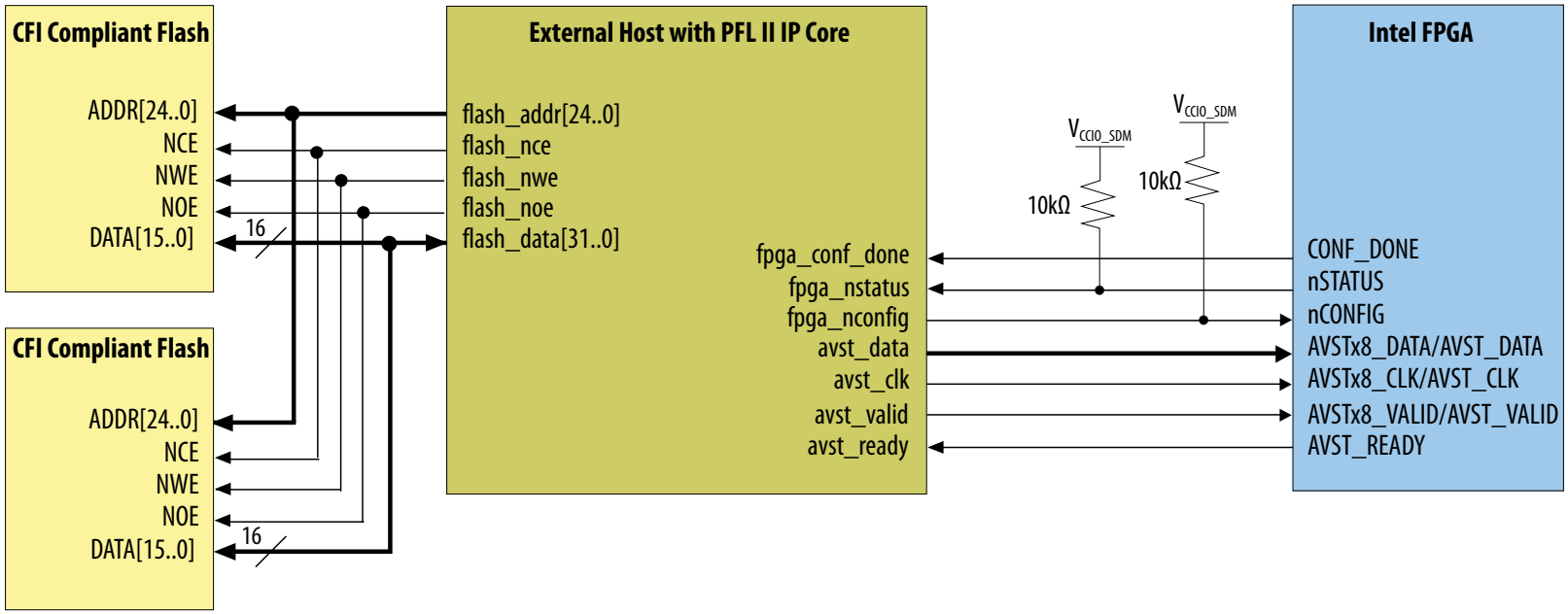
Figure 17. Programming the CFI Flash Memory with the JTAG Interface



The PFL II IP core supports dual flash memory devices in burst read mode to achieve faster configuration times. You can connect two MT29EW CFI flash memory devices to the host in parallel using the same data bus, clock, and control signals. During FPGA configuration, the AVST_CLK frequency is four times faster than the `flash_clk` frequency.

Figure 18. PFL II IP core with Dual CFI Flash Memory Devices

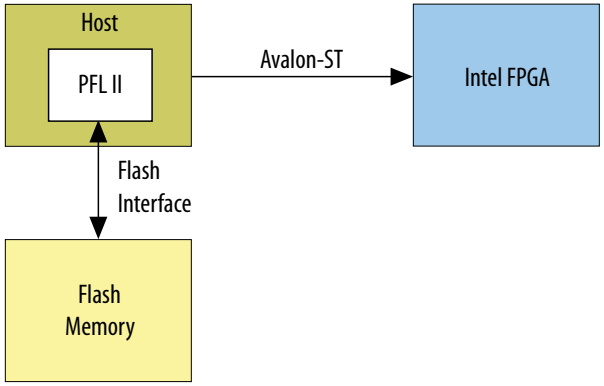
The flash memory devices must have the same memory density from the same device family and manufacturer.



3.1.10.1.2. Controlling Avalon-ST Configuration with PFL II IP Core

The PFL II IP core in the host determines when to start the configuration process, read the data from the flash memory device, and configure the Intel Agilex device using the Avalon-ST configuration scheme.

Figure 19. FPGA Configuration with Flash Memory Data



You can use the PFL II IP core to either program the flash memory devices, configure your FPGA, or both. To perform both functions, create separate PFL II functions if any of the following conditions apply to your design:

- You modify the flash data infrequently.
- You have JTAG or In-System Programming (ISP) access to the configuration host.
- You want to program the flash memory device with non-Intel FPGA data, for example initialization storage for an ASSP. You can use the PFL II IP core to program the flash memory device for the following purposes:
  - To write the initialization data
  - To store your design source code to implement the read and initialization control with the host logic

3.1.10.1.3. Mapping PFL II IP Core and Flash Address

The address connections between the PFL II IP core and the flash memory device vary depending on the flash memory device vendor and data bus width.

Figure 20. Flash Memory in 8-Bit Mode

The address connection between the PFL II IP core and the flash memory device are the same.

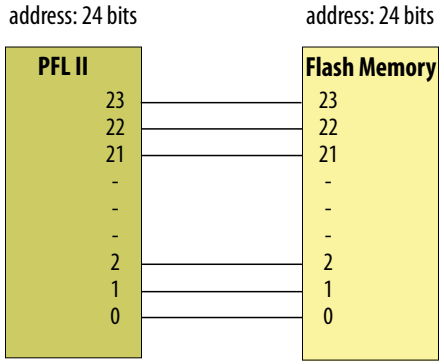


Figure 21. Flash Memories in 16-Bit Mode

The flash memory addresses in 16-bit flash memory shift one bit down in comparison with the flash addresses in PFL II IP core. The flash address in the flash memory starts from bit 1 instead of bit 0.

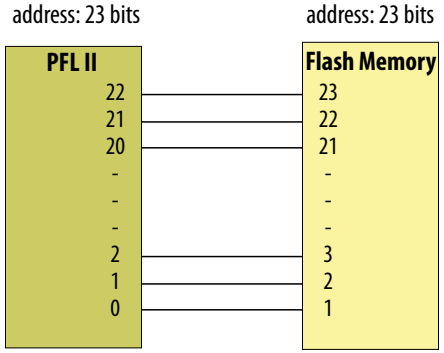


Figure 22. Cypress and Micron M28, M29 Flash Memory in 8-Bit Mode

The flash memory addresses in Cypress 8-bit flash shifts one bit up. Address bit 0 of the PFL II IP core connects to data pin D15 of the flash memory.

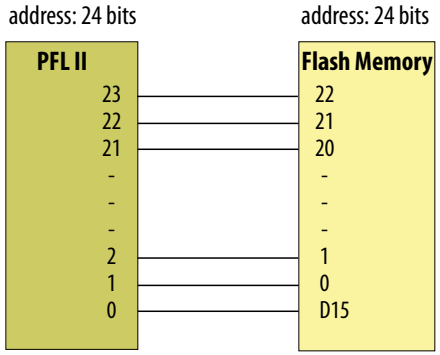
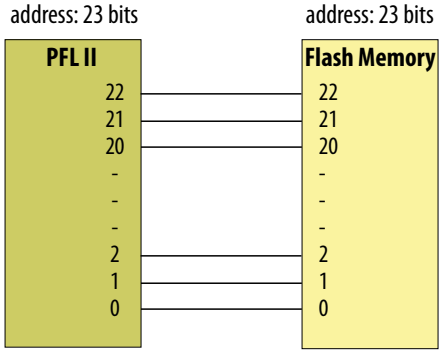


Figure 23. Cypress and Micron M28, M29 Flash Memory in 16-Bit Mode

The address bit numbers in the PFL II IP core and the flash memory device are the same.



3.1.10.1.4. Implementing Multiple Pages in the Flash .pof

The PFL II IP core stores configuration data in a maximum of eight pages in a flash memory block.

The total number of pages and the size of each page depends on the flash density. Here are some guidelines for storing your designs to pages:

- Always store designs for different FPGA chains on different pages.
- You may choose store different designs for a FPGA chain on a single page or on multiple pages.
- When you choose to store the designs for a FPGA chain on a single page, the design order must match the JTAG chain device order.

Use the generated `.sof` to create a flash memory device `.pof`. The following address modes are available for the `.sof` to `.pof` conversion:

- Block mode—allows you to specify the start and end addresses for the page.
- Start mode—allows you to specify only the start address. The start address for each page must be on an 8 KB boundary. If the first valid start address is `0×000000`, the next valid start address is an increment of `0×2000`.
- Auto mode—allows the Intel Quartus Prime software to automatically determine the start address of the page. The Intel Quartus Prime software aligns the pages on a 128 KB boundary. If the first valid start address is `0x000000`, the next valid start address is an multiple of `0x20000`.

3.1.10.1.5. Storing Option Bits

In addition to design data, the flash memory stores the option bits. You must specify the address for the options bits in two places: the PFL II IP and in the option bits address of the flash memory device.

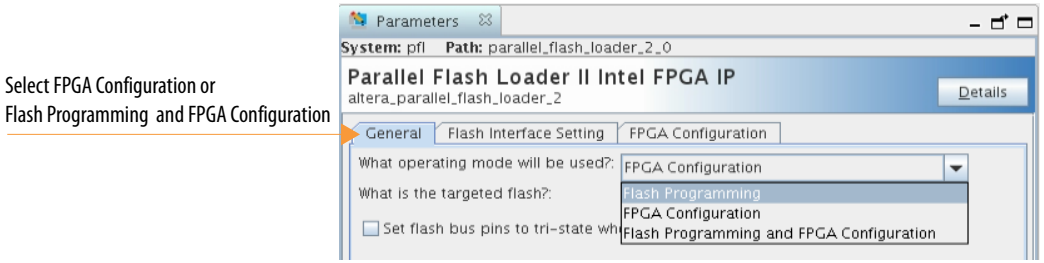
The option bits contain the following information:

- The start address for each page.
- The `.pof` version for flash programming. This value is the same for all pages.
- The `Page-Valid` bits for each page. The `Page-Valid` bit is bit 0 of the start address. The PLF II IP core writes this bit after successfully programming the page.

You use the **Programming File Generator** dialog box to specify the **Start address** of the option bits. Specify your flash device using **Add Device** on the **Configuration Tab** of the **Programming File Generator** dialog box. Then click **OPTIONS** and **EDIT** to specify the **Start address** for the option bits. This **Start address** must match the address you specify for **What is the byte address of the option bits, in hex?** when specifying the PFL II IP parameters.

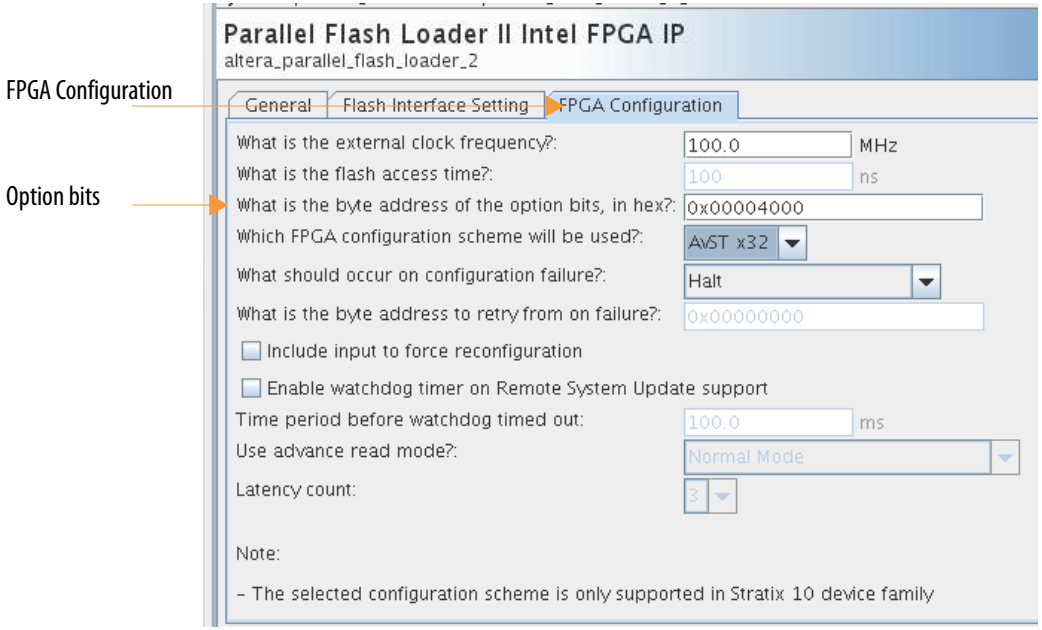
You set the option bits in the PFL II IP Intel FPGA IP using the parameter editor. By default the PFL II IP displays **Flash Programming** for the **What operating mode will be used?** parameter. In this default state, the **FPGA Configuration** tab is not visible. Select either **FPGA Configuration** or **Flash Programming and FPGA Configuration** for the **What operating mode will be used** parameter on the **General** tab. The following figure shows the **FPGA Configuration** option.

Figure 24. General Tab of the PFL II IP



Specify the options bits hex address for the **What is the base address of the option bits, in hex?** parameter on the **FPGA Configuration** tab.

Figure 25. FPGA Configuration Tab of the PFL II IP



You use the **Programming File Generator** dialog box to specify the **Start address** of the option bits. Specify your flash device using **Add Device** on the **Configuration Tab** of the **Programming File Generator** dialog box. Then click **OPTIONS** and **EDIT** to specify the **Start address** for the option bits. This **Start address** must match the address you specify for **What is the byte address of the option bits, in hex?** when specifying the PFL II IP parameters.

The Intel Quartus Prime **Programming File Generator** generates the information for the `.pof` version when you convert the `.sofs` to `.pofs`. The value for the `.pof` version for Intel Agilex is `0x05`. The following table shows an example of page layout for a `.pof` using all eight pages. This example stores the `.pof` version at 0x80.

Table 12. Option Bits Sector Format

Sector Offset	Value
0x00–0x03	Page 0 start address
0x04–0x07	Page 0 end address
0x08–0x0B	Page 1 start address
0x0C–0x0F	Page 1 end address
0x10–0x13	Page 2 start address
0x14–0x17	Page 2 end address
0x18–0x1B	Page 3 start address
0x1C–0x1F	Page 3 end address
0x20–0x23	Page 4 start address
0x24–0x27	Page 4 end address
0x28–0x2B	Page 5 start address
0x2C–0x2F	Page 5 end address
0x30–0x33	Page 6 start address
0x34–0x37	Page 6 end address
0x38–0x3B	Page 7 start address
0x3C–0x3F	Page 7 end address
0x40–0x7F	Reserved
0x80[7]	.pof version
0x81-0xFF	Reserved

Caution: To prevent the PFL II IP core from malfunctioning, do not overwrite any information in the option bits sector. Always store the option bits in unused addresses in the flash memory device.

(7) The .pof version occupies only one byte in the option bits sector.

Related Information

PFL II Parameters on page 79

3.1.10.1.6. Verifying the Option Bit Start and End Addresses

You can decode the start and end address that you specified for each of the SOF page when converting a `.sof` to `.pof` file from the 32-bit value of the sector offset address. If you encounter a configuration error you can verify that the generated bitstream addresses match the addresses you specified in the Intel Quartus Prime Software.

The following table shows the bit fields of the start address.

Table 13. Start Address Bit Content

Bit	Width	Description
31:11	21	Addressable start address
10:1	10	Reserved bits
0	1	Page valid bit • 0=Valid • 1=Error

Table 14. End Address Bit Content

Bit	Width	Description
31:0	32	Addressable end address

To restore the addresses:

- Start address—append 13'b0000000000000 to the addressable start address
- End address—append 2'b11 to the addressable end address

For a `.pof` that has two page addresses with the values shown in the following table.

Sector Offset	Value
0x00 – 0x03	0x00004000
0x04 – 0x07	0x00196E30
0x08 – 0x0B	0x001C0000

Sector Offset	Value
0x0C – 0x0F	0x00352E30

For Page 0 if you append the start address bits[31:11] with 13'b0000000000000, the result is 32'b00000000000000010000000000000000 = 0x10000.

If you append the end address 0x00196E0 with 2'b11, the result is 26'b00011001011011100011000011 = 0x65B8C3.

For Page 1 if you append the start address with 13'b0000000000000, the result is 32'b00000000000111000000000000000000 = 0x700000.

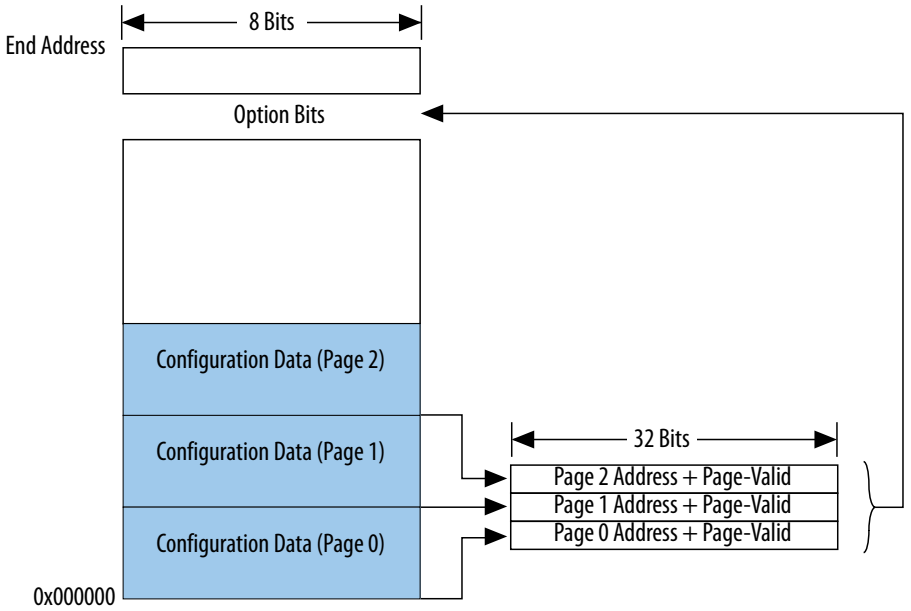
If you append end address 0x00352E30 with 2'b11, the result is 32'b00000000110101001011100011000011 = 0xD4B8C3.

The start and end address must be correlated with the start and end address for each page printed in the .map file.

3.1.10.1.7. Implementing Page Mode and Option Bits in the CFI Flash Memory Device

The following figure shows an sample layout of a .pof with three pages. The end addresses depend on the density of the flash memory device. For different density devices refer to the *Byte Address Range for CFI Flash Memory Devices with Different Densities* table below. The option bits follow the configuration data in memory.

Figure 26. Implementing Page Mode and Option Bits in the CFI Flash Memory Device



The following figure shows the layout of the option bits for a single page. Because the start address must be on an 8 KB boundary, bits 0-12 of the page start address are set to zero and are not stored in the option bits.

Figure 27. Page Start Address, End Address, and Page-Valid Bit Stored as Option Bits

The Page-Valid bits indicate whether each page is successfully programmed. The PFL II IP core sets the Page-Valid bits after successfully programming the pages.

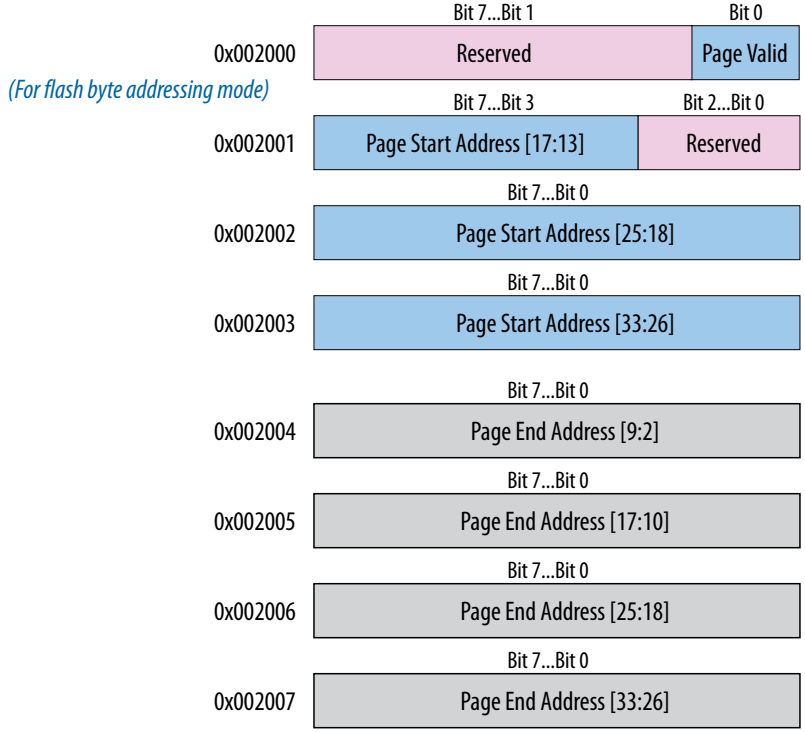


Table 15. Byte Address Range for CFI Flash Memory Devices with Different Densities

CFI Device (Megabit)	Address Range
8	0x0000000–0x00FFFFF
16	0x0000000–0x01FFFFF
32	0x0000000–0x03FFFFF
64	0x0000000–0x07FFFFF

continued...

CFI Device (Megabit)	Address Range
128	0x0000000-0x0FFFFFF
256	0x0000000-0x1FFFFFF
512	0x0000000-0x3FFFFFF
1024	0x0000000-0x7FFFFFF

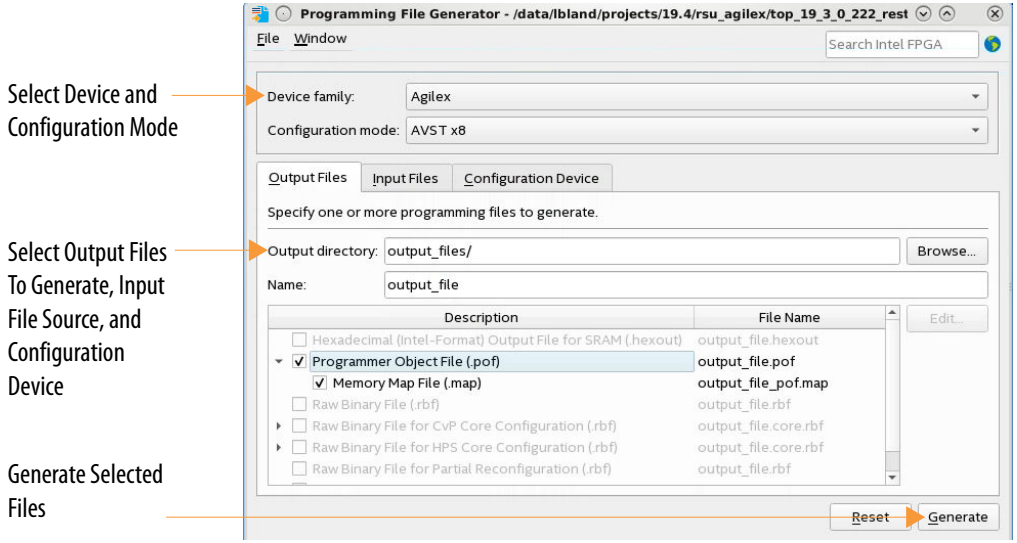
3.1.10.2. Using the PFL II IP Core

3.1.10.2.1. Converting .sof to .pof File

You can use the **Programming File Generator** to convert the `.sof` file to a `.pof`. The **Programming File Generator** options change dynamically according to your device and configuration mode selection.

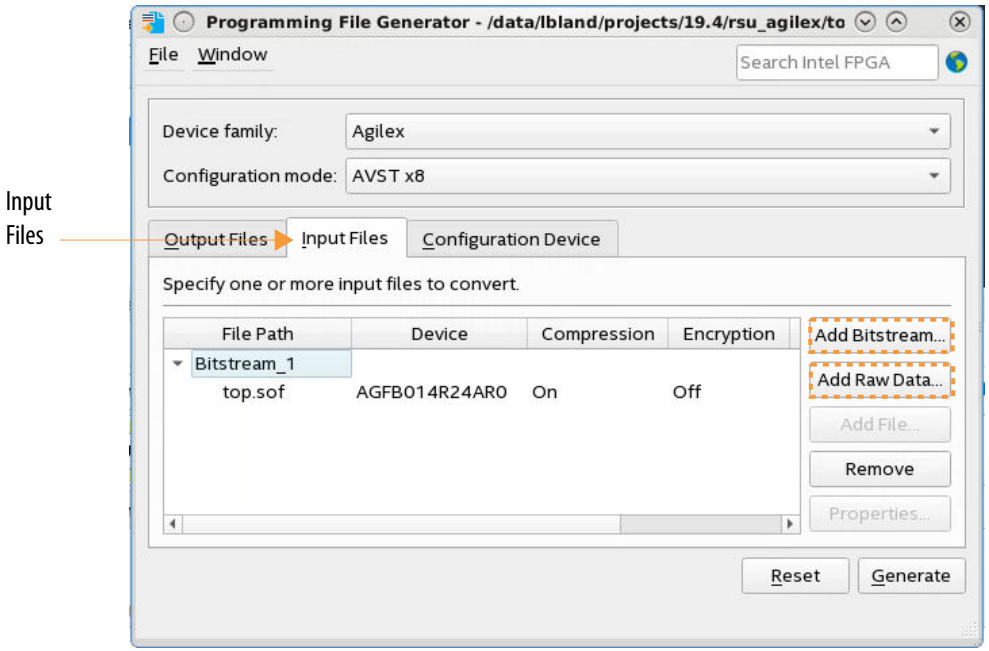
1. Click **File ➤ Programming File Generator**.
2. For **Device family** select **Intel Agilex**.
3. For **Configuration mode** select Avalon-ST configuration scheme that you plan to use.
4. For **Output directory**, click **Browse** to select your output file directory.
5. For **Name** specify a name for your output file.
6. On the **Output Files** tab, enable the checkbox for generation of the file or files you want to generate.
7. Specify the **Output directory** and **Name** for the file or files you generate.

Figure 28. Programming File Generator Output Files Tab



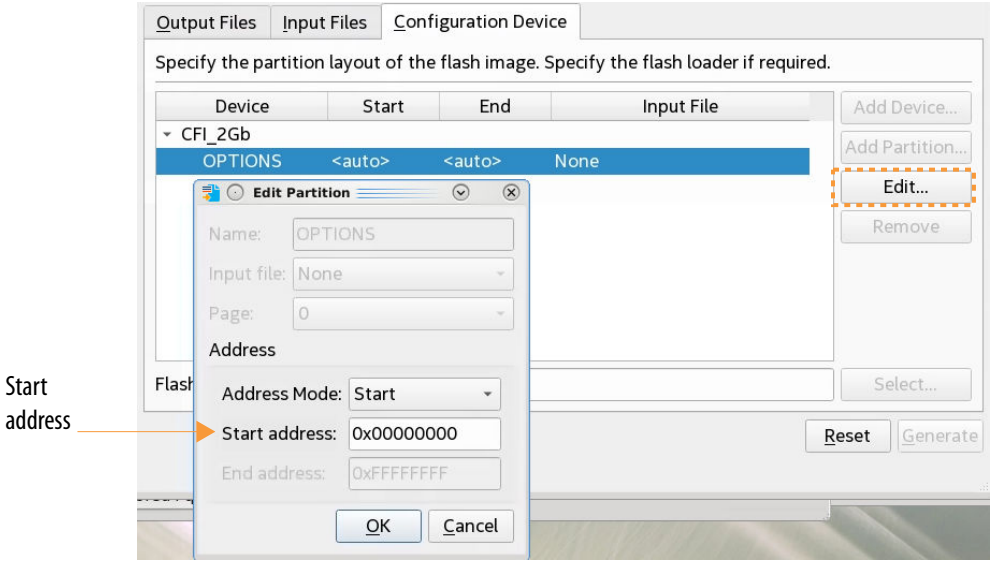
8. To specify a `.sof` that contains the configuration bitstream, on the **Input Files** tab, click **Add Bitstream**.

Figure 29. Input Files Tab



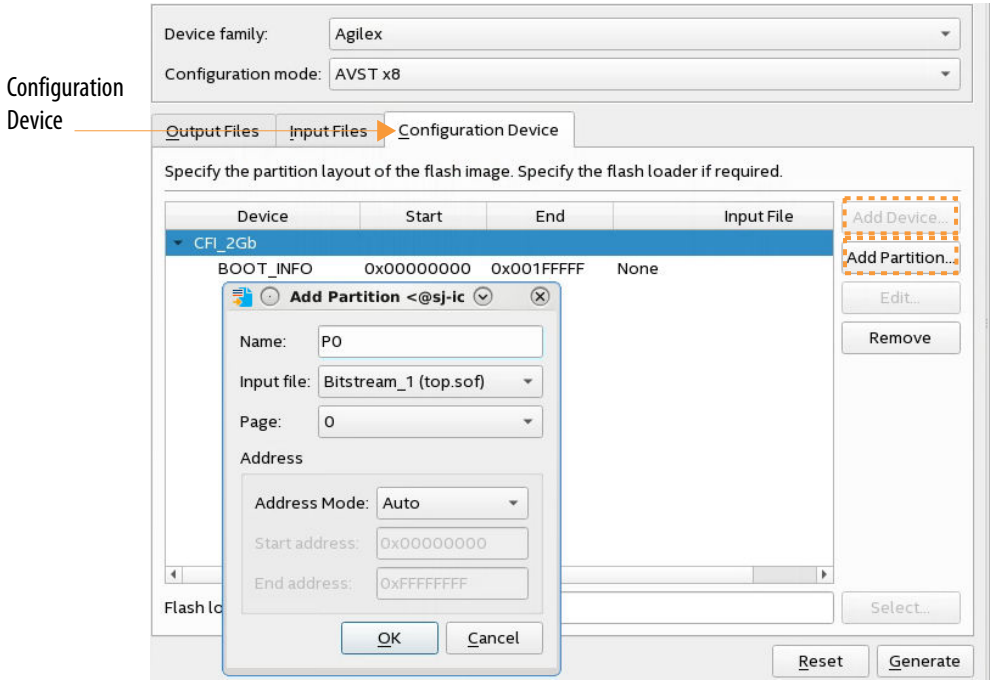
9. To include raw data, click **Add Raw Data** and specify a Hexadecimal (Intel-Format) Output File (`.hex`) or binary (`.bin`) file. This step is optional.
10. On the **Configuration Device** tab, click **Add device**. The **Add Device** dialog box appears. Select your flash device from the drop-down list of available parallel flash devices.
11. Click **OPTIONS** and then **Edit**. In the **Edit Partition** dialog box specify the **Start address** of the **Options** in flash memory. This address must match the address you specify for **What is the byte address of the option bits, in hex?** when specifying the PFL II IP parameters. Ensure that the option bits sector does not overlap with the configuration data pages and that the start address is on an 8 KB boundary.

Figure 30. Edit Partition: OPTIONS for Flash Device



12. With the flash device selected, click **Add Partition** to specify a partition in flash memory.

Figure 31. Add Flash Device and Partition



a. For **Name** select a Partition name.

b. For **Input File** specify the `.sof`.

c. From the **Page** dropdown list, select the page to write this `.sof`.

d. For **Address mode** select the addressing mode to use.

   The following modes are available:

   - **Auto**— For the tool to automatically allocates a block in the flash device to store the data.
   - **Block**—To specify the start and end address of the flash partition.
   - **Start**—To specify the start address of the partition. The tool assigns the end address of the partition based on the input data size.

e. For **Block** and **Start** options, specify the address information.

3.1.10.2.2. Creating Separate PFL II Functions

Follow these steps to create separate PFL II IP instantiations for programming and configuration control:

1. In the IP Catalog locate the Parallel Flash Loader II Intel FPGA IP.
2. On the **General** tab for **What operating mode will be used**, select **Flash Programming Only**.
3. Intel recommends that you turn on the **Set flash bus pins to tri-state when not in use**.
4. Specify the parameters on the **Flash Interface Settings** and **Flash Programming** tabs to match your design.
5. Compile and generate a `.pof` for the flash memory device. Ensure that you tri-state all unused I/O pins.
6. To create a second PFL II instantiation for FPGA configuration, on the **General** tab, for **What operating mode will be used**, select **FPGA Configuration**.
7. Use this **Flash Programming Only** instance of the PFL II IP to write data to the flash device.
8. Whenever you must program the flash memory device, program the CPLD with the flash memory device `.pof` and update the flash memory device contents.
9. Reprogram the host with the production design `.pof` that includes the configuration controller.

Note: By default, all unused pins are set to ground. When programming the configuration flash memory device through the host JTAG pins, you must tri-state the FPGA configuration pins common to the host and the configuration flash memory device. You can use the `pfl_flash_access_request` and `pfl_flash_access_granted` signals of the PFL II block to tri-state the correct FPGA configuration pins.

3.1.10.2.3. Programming CPLDs and Flash Memory Devices Sequentially

This procedure provides a single set of instructions for the Intel Quartus Prime Programmer to configure the CPLD and write the flash memory device.

1. Open the **Programmer** and click **Add File** to add the `.pof` for the CPLD.
2. Right-click the **CPLD .pof** and click **Attach Flash Device**.
3. In the **Flash Device** menu, select the appropriate density for the flash memory device.
4. Right-click the flash memory device density and click **Change File**.
5. Select the `.pof` generated for the flash memory device. The Programmer appends the `.pof` for the flash memory device to the `.pof` for the CPLD.
6. Repeat this process if your chain has additional devices.
7. Check all the boxes in the **Program/Configure** column for the new `.pof` and click **Start** to program the CPLD and flash memory device.

3.1.10.2.4. Programming CPLDs and Flash Memory Devices Separately

Follow these instructions to program the CPLD and the flash memory devices separately:

1. Open the **Programmer** and click **Add File**.
2. In the **Select Programming File**, add the targeted `.pof`, and click **OK**.
3. Check the boxes under the **Program/Configure** column of the `.pof`.
4. Click **Start** to program the CPLD.
5. After the programming progress bar reaches 100%, click **Auto Detect**.

   For example, if you are using dual Micron or Macronix flash devices, the programmer window shows a dual chain in your setup. Alternatively, you can add the flash memory device to the programmer manually. Right-click the CPLD `.pof` and click **Attach Flash Device**. In the **Select Flash Device** dialog box, select the device of your choice.
6. Right-click the flash memory device density and click **Change File**.

Note: For designs with more than one flash device, you must select the density that is equivalent to the sum of the densities of all devices. For example, if the design includes two 512-Mb CFI flash memory devices, select CFI 1 Gbit.

7. Select the `.pof` generated for the flash memory device. The Programmer attaches the `.pof` for the flash memory device to the `.pof` of the CPLD.
8. Check the boxes under the **Program/Configure** column for the added `.pof` and click **Start** to program the flash memory devices.

   Note: If your design includes the PFL II IP the Programmer allows you to program, verify, erase, blank-check, or examine the configuration data page, the user data page, and the option bits sector separately. The programmer erases the flash memory device if you select the `.pof` of the flash memory device before programming. To prevent the Programmer from erasing other sectors in the flash memory device, select only the pages, .hex data, and option bits.

3.1.10.2.5. Defining New CFI Flash Memory Device

The PFL II IP core supports Intel- and AMD-compatible flash memory devices. In addition to the supported flash memory devices, you can define the new Intel- or AMD-compatible CFI flash memory device in the PFL II-supported flash database using the **Define New CFI Flash Device** function.

To add a new CFI flash memory device to the database or update a CFI flash memory in the database, follow these steps:

1. In the Programmer window, on the Edit menu, select **Define New CFI Flash Device**. The following table lists the three functions available in the Define CFI Flash Device window.

Table 16. Functions of the Define CFI Flash Device Feature

Function	Description
New	Add a new Intel- or AMD-compatible CFI flash memory device into the PFL II-supported flash database.
Edit	Edit the parameters of the newly added Intel- or AMD-compatible CFI flash memory device in the PFL II-supported flash database.
Remove	Remove the newly added Intel- or AMD-compatible CFI flash memory device from the PFL II-supported flash database.

2. To add a new CFI flash memory device or edit the parameters of the newly added CFI flash memory device, select **New** or **Edit**. The **New CFI Flash Device** dialog box appears.
3. In the **New CFI Flash Device** dialog box, specify or update the parameters of the new flash memory device. You can obtain the values for these parameters from the data sheet of the flash memory device manufacturer.

Figure 32. Using the Programmer Edit Menu to Define a New Flash Device

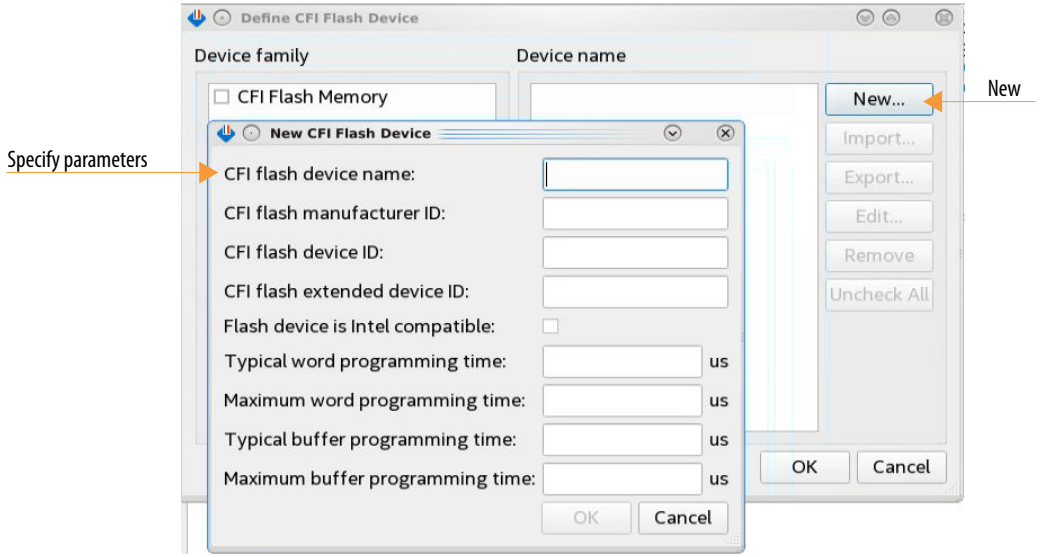


Table 17. Parameter Settings for New CFI Flash Device

Parameter	Description
CFI flash device name	Define the CFI flash name
CFI flash device ID	Specify the CFI flash identifier code
CFI flash manufacturer ID	Specify the CFI flash manufacturer identification number
CFI flash extended device ID	Specify the CFI flash extended device identifier, only applicable for AMD-compatible CFI flash memory device
Flash device is Intel compatible	Turn on the option if the CFI flash is Intel compatible
Typical word programming time	Typical word programming time value in µs unit
Maximum word programming time	Maximum word programming time value in µs unit
Typical buffer programming time	Typical buffer programming time value in µs unit
Maximum buffer programming time	Maximum buffer programming time value in µs unit

Note: You must specify either the word programming time parameters, buffer programming time parameters, or both. Do not leave both programming time parameters with the default value of zero.

4. Click **OK** to save the parameter settings.
5. After you add, update, or remove the new CFI flash memory device, click **OK**.

The Windows registry stores user flash information. Consequently, you must have system administrator privileges to store the parameters in the **Define New CFI Flash Device** window in the Intel Quartus Prime Pro Edition Programmer.

3.1.10.3. PFL II Parameters

Table 18. PFL II General Parameters

Options	Value	Description
What operating mode will be used?	• **Flash Programming** • **FPGA Configuration** • **Flash Programming and FPGA Configuration**	Specifies the operating mode of flash programming and FPGA configuration control in one IP core or separate these functions into individual blocks and functionality.
What is the targeted flash?	• **CFI Parallel Flash** • **Quad SPI Flash**	Specifies the flash memory device connected to the PFL II IP core.
Set flash bus pins to tri-state when not in use	• **On** • **Off**	Allows the PFL II IP core to tri-state all pins interfacing with the flash memory device when the PFL II IP core does not require access to the flash memory.

Table 19. PFL II Flash Interface Setting Parameters

Options	Value	Description
How many flash devices will be used?	• **1–16**	Specifies the number of flash memory devices connected to the PFL II IP core.
What's the largest flash device that will be used?	• **8 Mbit–4 Gbit**	Specifies the density of the flash memory device to be programmed or used for FPGA configuration. If you have more than one flash memory device connected to the PFL II IP core, specify the largest flash memory device density. For dual CFI flash, select the density that is equivalent to the sum of the density of two flash memories. For example, if you use two 512-Mb CFI flashes, you must select **CFI 1 Gbit**.
What is the flash interface data width	• **8** • **16** • **32**	Specifies the flash data width in bits. The flash data width depends on the flash memory device you use. For multiple flash memory device support, the data width must be the same for all connected flash memory devices.

continued...

Options	Value	Description
		Select the flash data width that is equivalent to the sum of the data width of two flash memories. For example, if you are targeting dual solution, you must select **32 bits** because each CFI flash data width is 16 bits.
Allow user to control FLASH_NRESET pin	• **On** • **Off**	Creates a FLASH_NRESET pin in the PFL II IP core to connect to the reset pin of the flash memory device. A low signal resets the flash memory device. In burst mode, this pin is available by default. When using a Cypress GL flash memory, connect this pin to the RESET pin of the flash memory.

Table 20. PFL II Flash Programming Parameters

Options	Value	Description
Flash programming IP optimization target	• **Area** • **Speed**	Specifies the flash programming IP optimization. If you optimize the PFL II IP core for **Speed**, the flash programming time is shorter, but the IP core uses more LEs. If you optimize the PFL II IP core for **Area**, the IP core uses fewer LEs, but the flash programming time is longer.
Flash programming IP FIFO size	• **16** • **32**	Specifies the FIFO size if you select **Speed** for flash programming IP optimization. The PFL II IP core uses additional LEs to implement FIFO as temporary storage for programming data during flash programming. With a larger FIFO size, programming time is shorter.
Add Block-CRC verification acceleration support	• **On** • **Off**	Adds a block to accelerate verification.

Table 21. PFL II FPGA Configuration Parameters

Options	Value	Description
What is the external clock frequency?	Provide the frequency of your external clock.	Specifies the user-supplied clock frequency for the IP core to configure the FPGA. The clock frequency must not exceed two times the maximum clock (AVST_CLK) frequency the FPGA can use for configuration. The PFL II IP core can divide the frequency of the input clock maximum by two.
What is the flash access time?	Provide the access time from the flash data sheet.	Specifies the flash access time. This information is available from the flash datasheet. Intel recommends specifying a flash access time that is equal to or greater than the required time. For CFI parallel flash, the unit is in ns. For NAND flash, the unit is in µs. NAND flash uses pages instead of bytes and requires greater access time. This option is disabled for quad SPI flash.

continued...

Options	Value	Description
What is the byte address of the option bits, in hex?	Provide the byte address of the option bits.	Specifies the option bits start address in flash memory. The start address must reside on an 8 KB boundary. This address must be the same as the bit sector address you specified when converting the `.sof` to a `.pof`. For more information refer to *Storing Option Bits*.
Which FPGA configuration scheme will be used?	• **Avalon-ST x8** • **Avalon-ST x16** • **Avalon-ST x32**	Specifies the width of the Avalon-ST interface.
What should occur on configuration failure?	• **Halt** • **Retry same page** • **Retry from fixed address**	Configuration behavior after configuration failure. • If you select **Halt**, the FPGA configuration stops completely after failure. • If you select **Retry same page**, after failure, the PFL II IP core reconfigures the FPGA with data from the page that failed. • If you select **Retry from fixed address**, the PFL II IP core reconfigures the FPGA a fixed address.
What is the byte address to retry from failure	—	If you select **Retry from fixed address** for configuration failure option, this option specifies the flash address the PFL II IP core to reads from.
Include input to force reconfiguration	• **On** • **Off**	Includes the optional `pfl_nreconfigure` reconfiguration input pin to enable reconfiguration of the FPGA.
Enable watchdog timer on Remote System Update support	• **On** • **Off**	Enables a watchdog timer for remote system update support. Turning on this option enables the `pfl_reset_watchdog` input pin and `pfl_watchdog_error` output pin. This option also specifies the period before the watchdog timer times out. The watchdog timer runs at the `pfl_clk frequency`.

continued...

Options	Value	Description
Time period before the watchdog timer times out	—	Specifies the time out period for the watchdog timer. The default time out period is 100 ms.
Use advance read mode?	• **Normal mode** • **Intel Burst mode** • **16 byte page mode (GL only)** • **32 byte page mode (MT23EW)** • **Micron Burst Mode (M58BW)**	This option improves the overall flash access time for the read process during the FPGA configuration. • **Normal mode**—applicable for all flash memory • **Intel Burst mode**—Applicable for devices that support bursting. Reduces sequential read access time • **16 byte page mode (GL only)**—applicable for Cypress GL flash memory only • **32 byte page mode (MT23EW)**—applicable tor MT23EW only • **Micron Burst Mode (M58BW)**—applicable for Micron M58BW flash memory only For more information about the read-access modes of the flash memory device, refer to the respective flash memory data sheet.
Latency count	• **3** • **4** • **5**	Specifies the latency count for **Intel Burst mode**.

3.1.10.4. PFL II Signals

Table 22. PFL II Signals

Pin	Type	Weak Pull-Up	Function
`pfl_nreset`	Input	—	Asynchronous reset for the PFL II IP core. Pull high to enable FPGA configuration. To prevent FPGA configuration, pull low when you do not use the PFL II IP core. This pin does not affect the PFL II IP flash programming functionality.
`pfl_flash_access_granted`	Input	—	For system-level synchronization. A processor or any arbiter that controls access to the flash drives this input pin. To use the PFL II IP core function as the flash master pull this pin high. Driving the `pfl_flash_access_granted` pin low prevents the JTAG interface from accessing the flash and FPGA configuration.
`pfl_clk`	Input	—	User input clock for the device. This is the frequency you specify for the **What is the external clock frequency?** parameter on the Configuration tab of the PFL II IP. This frequency must not be higher than the maximum DCLK frequency you specify for FPGA during configuration. This pin is not available if you are only using the PFL II IP for flash programming.

continued...

Pin	Type	Weak Pull-Up	Function
fpga_pgm[]	Input	—	Determines the page for the configuration. This pin is not available if you are only using the PFL II IP for flash programming.
fpga_conf_done	Input	10 kΩ Pull-Up Resistor	Connects to the CONF_DONE pin of the FPGA. The FPGA releases the pin high if the configuration is successful. During FPGA configuration, this pin remains low. This pin is not available if you are only using the PFL II IP for flash programming.
fpga_nstatus	Input	10 kΩ Pull-Up Resistor	Connects to the nSTATUS pin of the FPGA. This pin is high before the FPGA configuration begins and must stay high during FPGA configuration. If a configuration error occurs, the FPGA pulls this pin low and the PFL II IP core stops reading the data from the flash memory device. This pin is not available if you are only using the PFL II IP for flash programming.
pfl_nreconfigure	Input	—	When low initiates FPGA reconfiguration. To implement manual control of reconfiguration connect this pin to a switch. You can use this input to write your own logic in a CPLD to trigger reconfiguration via the PFL II IP. You can use pfl_nreconfigure to drive the fpga_nconfig output signal initiating reconfiguration. The pfl_clk pin registers this signal. This pin is not available if you are only using the PFL II IP for flash programming.
pfl_flash_access_request	Output	—	For system-level synchronization. When necessary, this pin connects to a processor or an arbiter. The PFL II IP core drives this pin high when the JTAG interface accesses the flash or the PFL II IP configures the FPGA. This output pin works in conjunction with the flash_noe and flash_nwe pins.
flash_addr[]	Output	—	The flash memory address. The width of the address bus depends on the density of the flash memory device and the width of the flash_data bus. Intel recommends that you turn **On** the **Set flash bus pins to tri-state when not in use** option in the PFL II .
flash_data[]	Input or Output (bidirectional pin)	—	Bidirectional data bus to transmit or receive 8-, 16-, or 32-bit data. Intel recommends that you turn **On** the **Set flash bus pins to tri-state when not in use** option in the PFL II. (8)
flash_nce[]	Output	—	Connects to the nCE pin of the flash memory device. A low signal enables the flash memory device. Use this bus for multiple flash memory device support. The flash_nce pin connects to each nCE pin of all the connected flash memory devices. The width of this port depends on the number of flash memory devices in the chain.

continued...

(8) Intel recommends that you do not insert logic between the PFL II pins and the host I/O pins, especially on the flash_data and fpga_nconfig pins.

Pin	Type	Weak Pull-Up	Function
flash_nwe	Output	—	Connects to the nWE pin of the flash memory device. When low enables write operations to the flash memory device.
flash_noe	Output	—	Connects to the nOE pin of the flash memory device. When low enables the outputs of the flash memory device during a read operation.
flash_clk	Output	—	For burst mode. Connects to the CLK input pin of the flash memory device. The active edges of CLK increment the flash memory device internal address counter. The flash_clk frequency is half of the pfl_clk frequency in burst mode for a single CFI flash. In dual CFI flash solution, the flash_clk frequency runs at a quarter of the pfl_clk frequency. Use this pin for burst mode only. Do not connect these pins from the flash memory device to the host if you are not using burst mode.
flash_nadv	Output	—	For burst mode. Connects to the address valid input pin of the flash memory device. Use this signal to latch the start address. Use this pin for burst mode only. Do not connect these pins from the flash memory device to the host if you are not using burst mode.
flash_nreset	Output	—	Connects to the reset pin of the flash memory device. A low signal resets the flash memory device.
fpga_nconfig	Open Drain Output	10-kW Pull-Up Resistor	Connects to the nCONFIG pin of the FPGA. A low pulse resets the FPGA and initiates configuration. These pins are not available for the flash programming option in the PFL II IP core. (8)
pfl_reset_watchdog	Input	—	A switch signal to reset the watchdog timer before the watchdog timer times out. To reset the watchdog timer hold the signal high or low for at least two pfl_clk clock cycles.
pfl_watchdog_error	Output	—	When high indicates an error condition to the watchdog timer.

Related Information

Avalon Interface Specifications

3.2. AS Configuration

In AS configuration schemes, the SDM block in the Intel Agilex device controls the configuration process and interfaces. The serial flash configuration device stores the configuration data. During AS Configuration, the SDM first powers on with the boot ROM. Then, the SDM loads the initial configuration firmware from AS x4 flash. After the configuration firmware loads, this firmware controls the remainder of the configuration process, including I/O configuration and FPGA core configuration. Designs including an HPS, can use the HPS to access serial flash memory after the initial configuration.

Note: The serial flash configuration device must be fully powered up at the same time or before ramping up V_{CCIO_SDM} of the Intel Agilex device.

The AS configuration scheme supports AS x4 (4-bit data width) mode only.

Table 23. Intel Agilex Configuration Data Width, Clock Rates, and Data Rates

Mode		Data Width (bits)	Max Clock Rate	Max Data Rate	MSEL[2:0]
Active	Active Serial (AS)	4	133 MHz	532 Mb	Fast mode - 001 Normal mode - 011

Table 24. Required Configuration Signals for the AS Configuration Scheme

You specify SDM I/O pin functions using the **Device ➤ Configuration ➤ Device and Pin Options** dialog box in the Intel Quartus Prime software. You can reassign the GPIO, dual-purpose configuration pins for other functions in user mode.

Configuration Function	Pin Type	Direction	Powered by
nSTATUS	SDM I/O	Output	V_{CCIO_SDM}
nCONFIG	SDM I/O	Input	V_{CCIO_SDM}
MSEL[2:0]	SDM I/O, Dual-Purpose	Input	V_{CCIO_SDM}
CONF_DONE	SDM I/O	Output	V_{CCIO_SDM}
AS_nCSO[3:0]	SDM I/O	Output	V_{CCIO_SDM}
AS_DATA[3:0]	SDM I/O	Bidirectional	V_{CCIO_SDM}
AS_CLK	SDM I/O	Output	V_{CCIO_SDM}

Note: Although the CONF_DONE and INIT_DONE configuration signals are not required, Intel recommends that you use these signals. The SDM drives the CONF_DONE signal high after successfully receiving full bitstream. The SDM drives the INIT_DONE signal high to indicate the device is fully in user mode.These signals are important when debugging configuration. You can reassign the GPIO, dual-purpose configuration pins for other functions in user mode.

MSEL Pin Function for the AS x4 Configuration Scheme

The SDM samples the MSEL pins immediately after power-on in the SDM Start state. After the SDM samples the MSEL pins, the MSEL pins become active-low chips selects. For AS x4 designs using one flash device, AS_nCS0 asserts low. The remaining chip select pins, AS_nCS1 - AS_nCS3 deassert high.

Related Information

AS Configuration Timing in Intel Agilex Devices
For timing parameter minimum, typical, and maximum values.

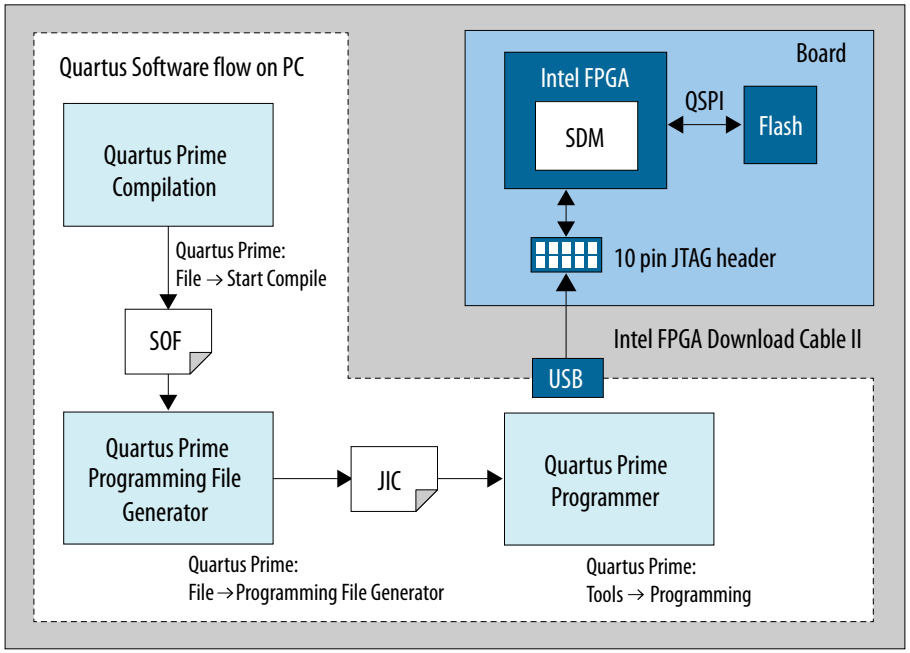
3.2.1. AS Configuration Scheme Hardware Components and File Types

You use the following components to implement the AS configuration scheme:

- Quad SPI flash memory
- The Intel FPGA Download Cable II to connect the Intel Quartus Prime Programmer to the PCB.

The following block diagram illustrates the components and design flow using the AS configuration scheme.

Figure 33. Components and Design Flow for .jic Programming



In addition to AS programming using a `.jic`, the Programmer supports direct programming of the quad SPI flash using a `.pof` as shown in *AS Programming Using Intel Quartus Prime or Third-Party Programmer.*

Table 25. Output File Types

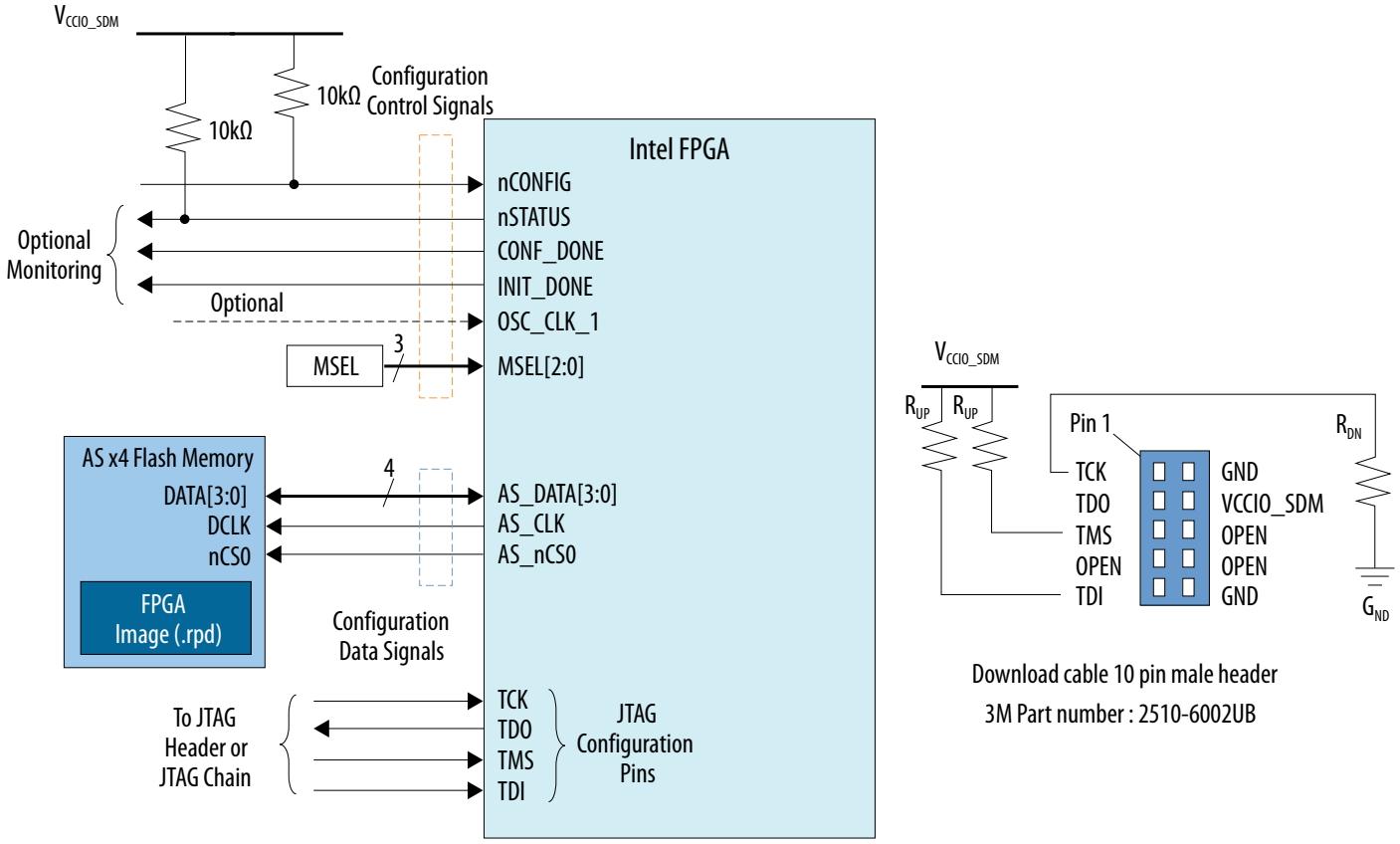
Programming File Type	Extension	Description
JTAG Indirect Configuration File	`.jic`	The `.jic` enables serial flash programming via Intel FPGA JTAG pins. This file type is available only for ASx4 configuration. A newly populated board using the ASx4 configuration scheme requires initial SDM firmware programming. The helper SOF image provides the required SDM firmware. You initially use the JTAG cable to load a SDM Helper SOF into the Intel Agilex device. The SDM can then load the flash device with the Intel Agilex design.

Related Information

Programming Serial Flash Devices using the AS Interface on page 92

3.2.2. AS Single-Device Configuration

Figure 34. Connections for AS x4 Single-Device Configuration



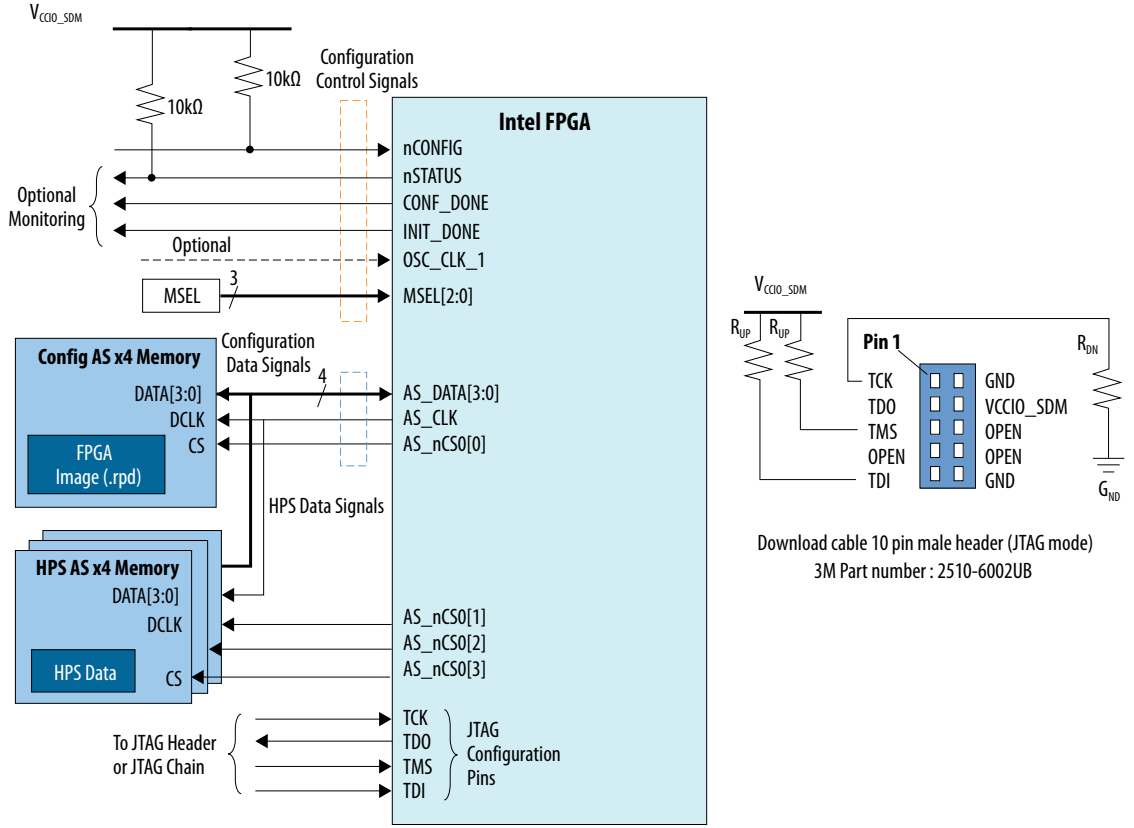
Related Information

MSEL Settings on page 26

3.2.3. AS Using Multiple Serial Flash Devices

Intel Agilex devices support one AS x4 flash memory device for AS configuration and up to three AS x4 flash memories for use with HPS data storage. The MSEL pins are dual-purpose and operate as MSEL only during POR state. After the FPGA device enters user mode, you can repurpose the MSEL pins as chip select pins. You must to ensure appropriate chip select pin connections to the configuration AS x4 flash memory and the HPS AS x4 flash memory. Each flash device has a dedicated AS_nCS0 pin but shares other pins.

Figure 35. Connections for AS Configuration with Multiple Serial Flash Devices



The following table shows the maximum supported AS_CLK frequency for a range of capacitance loading values when using multiple flash devices. The maximum AS_CLK frequency also depends on whether you use the OSC_CLK_1 or internal oscillator as the clock source.

Table 26. Maximum AS_CLK Frequency as a Function of Board Capacitance Loading and Clock Source

Capacitance Loading (pF)	Maximum Supported AS_CLK (MHz)	
	OSC_CLK_1 (MHz)	Internal Oscillator (MHz)
10	133/125	115
19	108	115
30	100	77
37	71.5	77
80	50	58
140	25	25

Related Information

- MSEL Settings on page 26
- Intel Agilex Device Data Sheet

3.2.4. AS Configuration Timing Parameters

Figure 36. AS Configuration Serial Output Timing Diagram

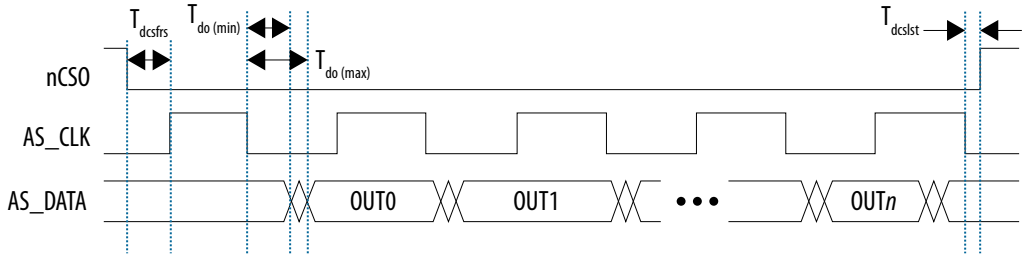
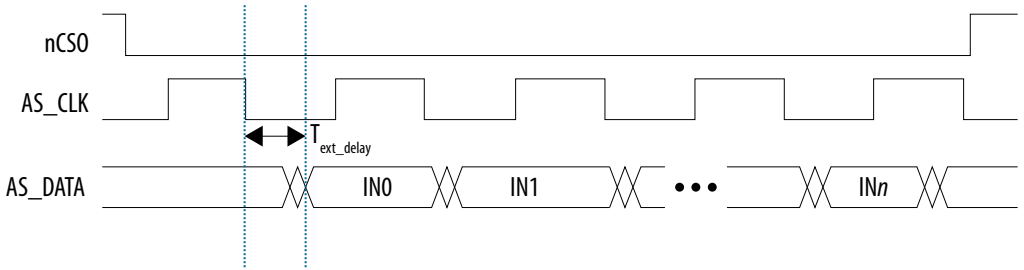


Figure 37. AS Configuration Serial Input Timing Diagram



Note: For more information about the timing parameters, refer to the *Intel Agilex Device Datasheet*.

3.2.5. Maximum Allowable External AS_DATA Pin Skew Delay Guidelines

You must minimize the skew on the AS data pins.

Skew delay includes the following elements:

- The delay due to the differences in board traces lengths on the PCB
- The capacitance loading of the flash device

The table below lists the maximum allowable skew delay depending on the AS_CLK frequency. Intel recommends that you to perform IBIS simulations to ensure that the skew delay does not exceed the maximum delay specified in this table.

Table 27. Maximum Skew for AS Data Pins in Nanoseconds (ns)

Symbol	Description	Frequency	Min	Typical	Max
T_{ext_skew}	Skew delay for AS_DATA for the AS_CLK frequency specified	133 MHz	—	—	3.60
		125 MHz	—	—	4.00
		115 MHz	—	—	4.20
		108 MHz	—	—	4.60
		100 MHz	—	—	5.0
		<100 MHz	—	—	5.0

3.2.6. Programming Serial Flash Devices

You can program serial flash devices in-system using the Intel FPGA Download Cable II or Intel FPGA Ethernet Cable.

You have the following two in-system programming options:

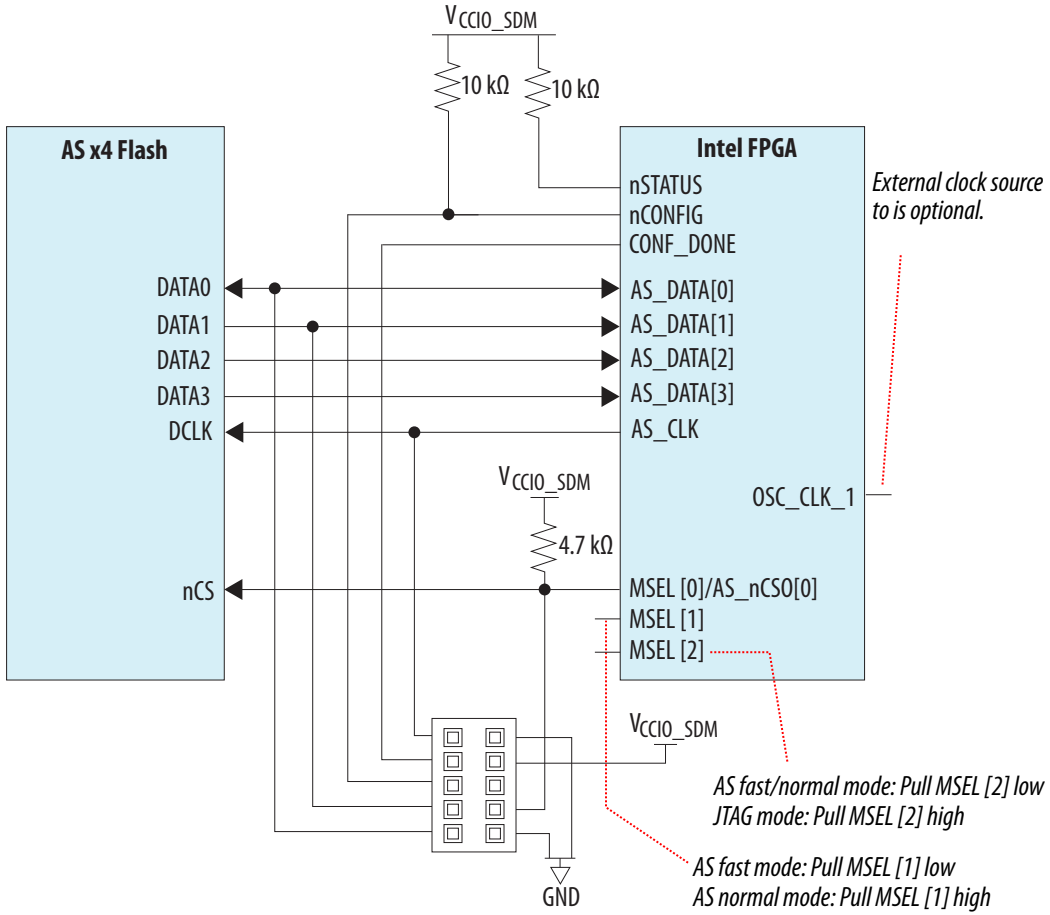
- Active Serial
- JTAG

3.2.6.1. Programming Serial Flash Devices using the AS Interface

When you select AS programming the Intel Quartus Prime software or any supported third-party software programs the configuration data directly into the serial flash device.

You must set MSEL to JTAG. When MSEL is set to JTAG, the SDM tristates the following AS pins: AS_CLK, AS_DATA0-AS_DATA3, and AS_nCS0-AS_nCS3. The Intel Quartus Prime Programmer programs the flash memory devices via the AS header. If you are using the Generic Serial Flash Interface Intel FPGA IP to write the flash memory the flash device must be connected to GPIO to access the flash device.

Figure 38. AS Programming Using Intel Quartus Prime or Third-Party Programmer



3.2.6.2. Programming Serial Flash Devices using the JTAG Interface

The Intel Quartus Prime Programmer interfaces to the SDM device through JTAG interface and programs the serial flash device. The SDM emulates AS programming.

Figure 39. Programming Your Serial Configuration Device Using JTAG and SDM Emulation of AS

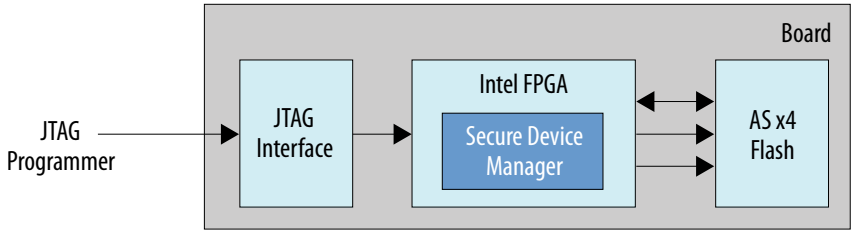
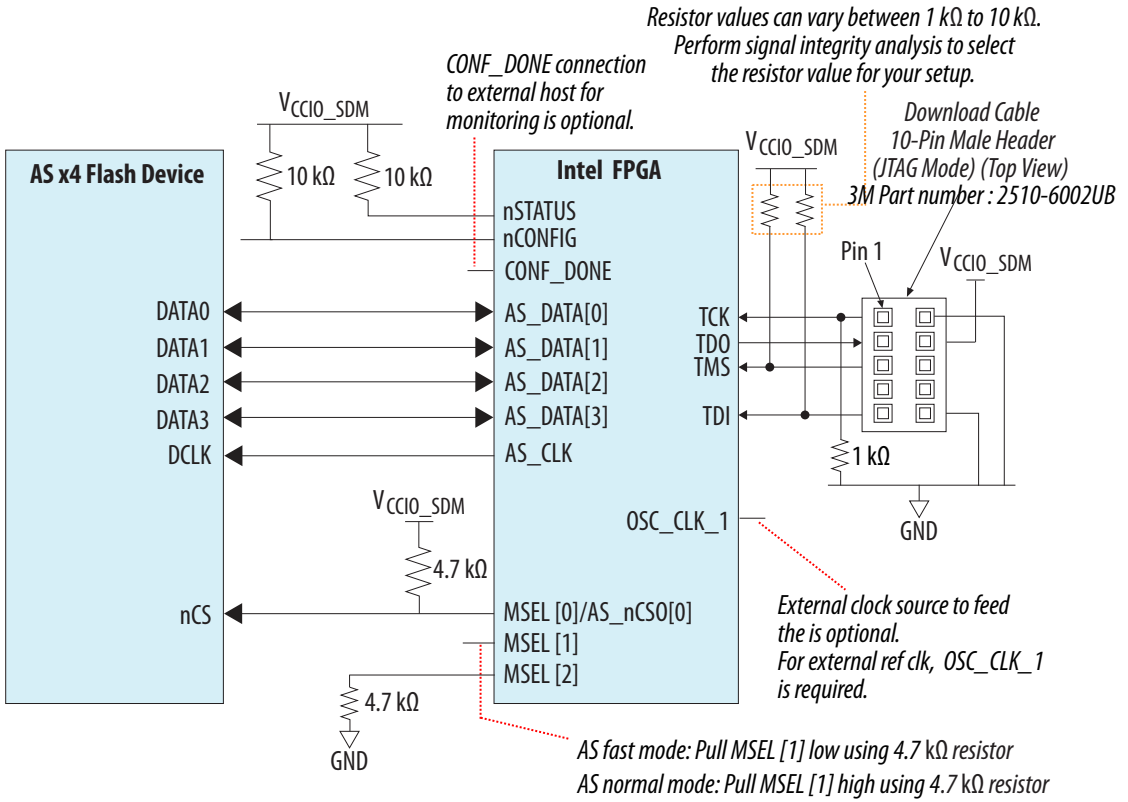


Figure 40. Connections for Programming the Serial Flash Devices using the JTAG Interface



Intel recommends using the JTAG interface to prepare the Quad SPI flash device for later use in AS mode.

This configuration scheme includes the following steps:

1. In the Intel Quartus Prime Programmer, select the **JTAG** programming mode and initiate programming by clicking **Start**.
2. The Programmer drives `.jic` configuration data to the board using the JTAG header connection.
3. The programmer first configures the SDM with configuration firmware. Then, the SDM drives configuration data from the programmer to the AS x4 flash device using SDM_IOs.
4. To use the Intel Agilex device in AS mode after successful programming of the flash device, set the MSEL pins to either AS fast or AS normal mode and power cycle the device.

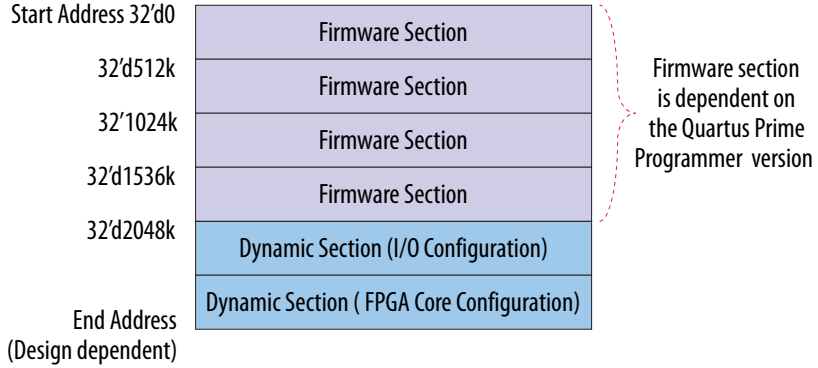
The Intel Quartus Prime Programmer interfaces to the SDM device through JTAG interface and programs the serial flash device.

3.2.7. Serial Flash Memory Layout

Serial flash devices store the configuration data in sections.

The following diagram illustrates sections of a non-HPS Intel Agilex configuration data mapping in a serial flash device. Refer to *Intel Agilex SoC FPGA Bitstream Sections* of the *HPS Technical Reference Manual* for more information about flash memory layout for HPS devices.

Figure 41. Serial Flash Memory Layout Diagram



If you use a third-party programmer to program an `.rpd`, ensure that the configuration data is stored starting from address 0 of the serial flash device. If you use `.jic` or `.pof` files, the Intel Agilex Programmer automatically programs the configuration data starting from address 0 of the serial flash device.

3.2.7.1. Understanding Quad SPI Flash Byte-Addressing

At power-on the SDM operates from boot ROM. The SDM loads configuration firmware from Quad SPI flash using 3-byte addressing. Once loaded, if the flash size is 256 Mb or larger, the SDM configures the Quad SPI flash to operate in 4-byte addressing mode and continues to load the rest of the bitstream until configuration completes.

Intel Agilex devices support the following third-party flash devices operating at 1.8 V:

- Macronix MX66U 512 Mb, 1 and 2 gigabits (Gb)
- Macronix MX25U 128 Mb, 256 Mb, and 512 Mb
- Micron MT25QU 128 Mb, 256 Mb, 512 Mb, 1 Gb, and 2 Gb

Micron and Macronix both offer Quad SPI memories a density range of 128Mb—2Gb.

3.2.8. AS_CLK

The Intel Agilex device drives AS_CLK to the serial flash device. An internal oscillator or the external clock that drives the OSC_CLK_1 pin generates AS_CLK. Using an external clock source allows the AS_CLK to run at a higher frequency. If you provide a 25 MHz, 100 MHz, or 125 MHz clock to the OSC_CLK_1 pin, the AS_CLK can run up to 133 MHz.

Set the maximum required frequency for the AS_CLK pin in the Intel Quartus Prime software as described in Active Serial Configuration Software Settings on page 98. The AS_CLK pin runs at or below your selected frequency.

Table 28. Supported configuration clock source and AS_CLK Frequencies in Intel Agilex Devices

Configuration Clock Source	AS_CLK Frequency (MHz)
Internal oscillator	25 58 77 115
OSC_CLK_1	25 50

continued...

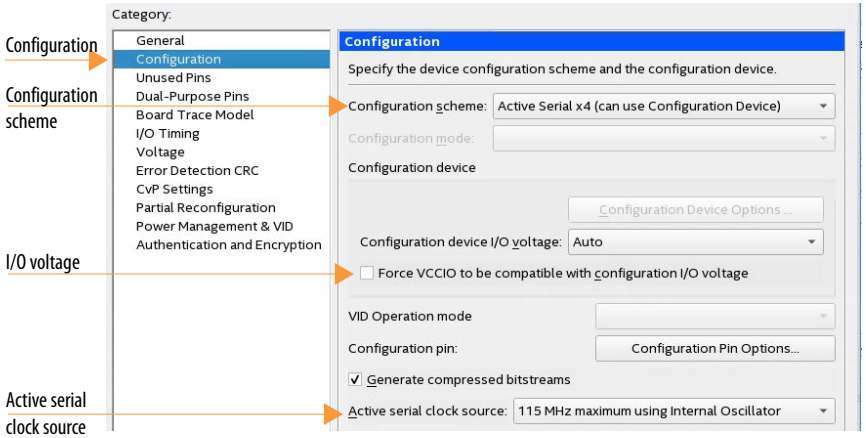
Configuration Clock Source	AS_CLK Frequency (MHz)
	71.5 100 108 125 133

3.2.9. Active Serial Configuration Software Settings

You must set the parameters in the **Device and Pin Options** of the Intel Quartus Prime software when using the AS configuration scheme.

To set the parameters for AS configuration scheme, complete the following steps:

1. On the **Assignments** menu, click **Device**.
2. In the **Device and Pin Options** select the **Configuration** category.
   a. Select **Active Serial x4** from the **Configuration scheme** drop down menu.



   b. Select **Auto** or **1.8 V** in the **Configuration device I/O voltage** drop-down list.
   c. Select the AS clock frequency from the **Active serial clock source** drop-down list.
3. Click **OK** to confirm and close the **Device and Pin Options**.

3.2.10. Intel Quartus Prime Programming Steps

3.2.10.1. Generating Programming Files using the Programming File Generator

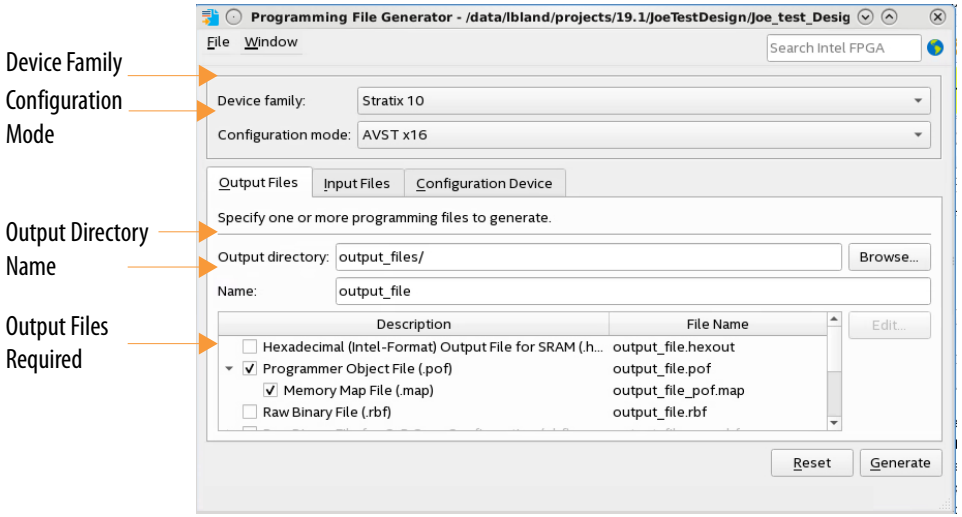
By default, the Intel Quartus Prime Compiler's Assembler module generates the primary files required for device programming at the end of full compilation. Primary programming files include the `.sof`, `.pof`, and `.rpd`. You can use the **Programming File Generator** to generate programming files for alternative device programming methods, such as the `.jic` for flash programming,`.rbf` for partial reconfiguration, or `.rpd` for third-party programmer configuration. The **Programming File Generator** supports Intel Agilex devices. The legacy **Convert Programming Files** dialog box does not support some advanced programming features for Intel Agilex devices.

Note: If you are generating an `.rpd` for remote system update (RSU), you must follow the instructions in Generating an Application Image on page 163 in the *Remote System Update* chapter. This procedure generates flash programming files for Intel Agilex devices.

Complete the following steps generate the programming file or files you require:

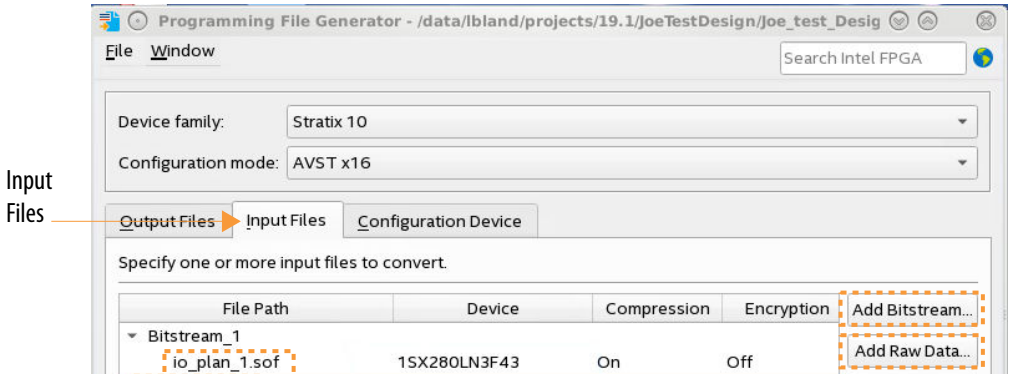
1. Click **File Programming File Generator**.
2. For **Device Family** select Intel Agilex
3. In the **Configuration mode**, select **Active Serial x4**.
4. Specify the **Output directory** and **Name** for the file you generate.
5. Under **Output directory**, select the appropriate file type for your design. The AS scheme supports the **Programmer Object File (.pof)**, **JTAG Indirect Configuration File (.jic)**, and **Raw Programming Data File (.rpd)** file types.

Figure 42. Programming File Generator Output Files



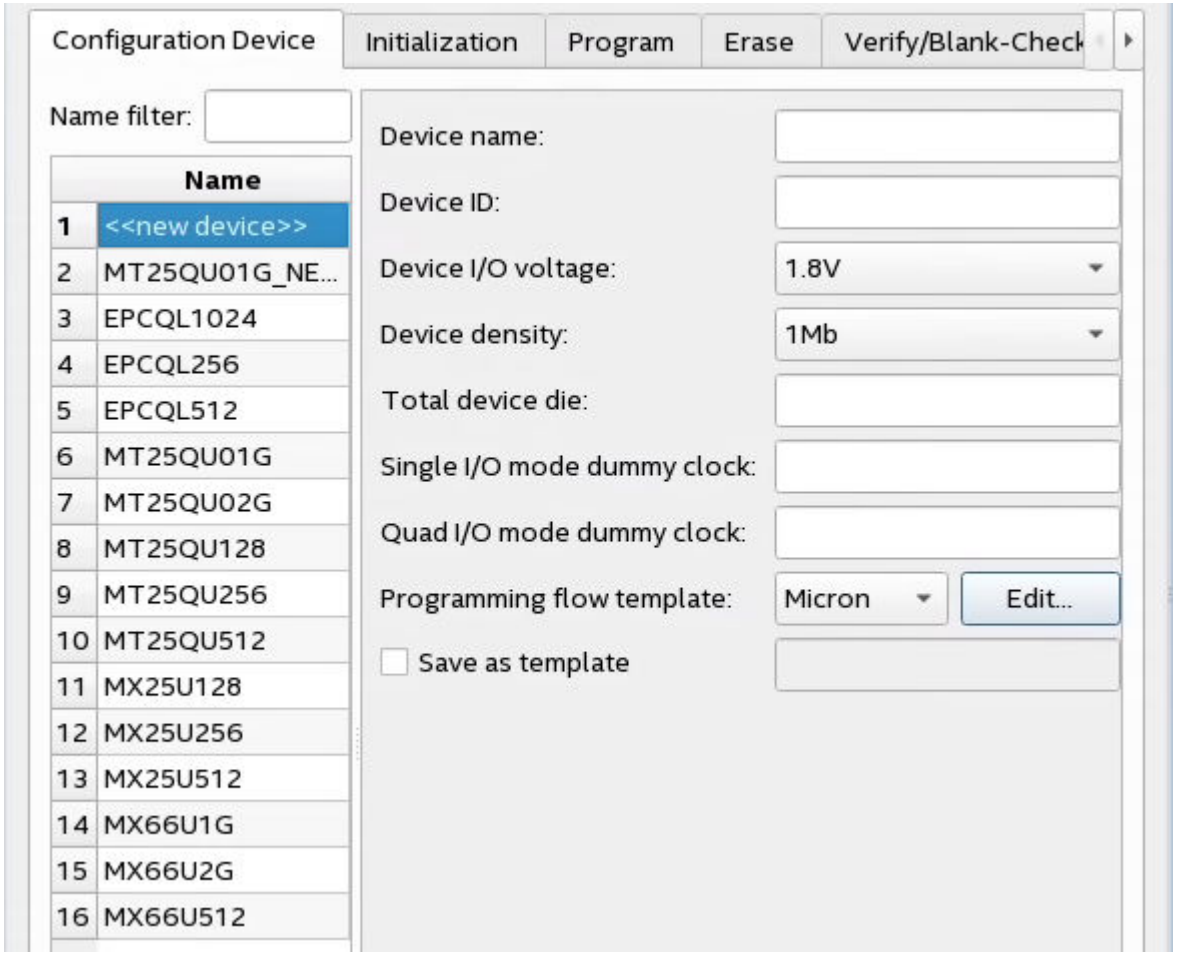
6. For the **JTAG Indirect Configuration File (.jic)** and **Programmer Object File (.pof)** you can turn on the **Memory Map File (.map)**. This option describes flash memory address locations.
   The **Input Files** tab is now available.
7. On the **Input Files** tab, click **Add Bitstream** and browse to your configuration bitstream.

Figure 43. Programming File Generator Input Files



8. On the **Configuration Device** tab, click **Add Device**. You can select your flash device from the from the **Configuration Device** list, or define a custom device using the available menu options. For more information about defining a custom configuration device, refer to the *Configuration Device Tab Settings (Programming File Generator)* in the Intel Quartus Prime Pro Edition User Guide: Programmer

Figure 44. Programming File Generator Input Files



Note: You do not need to specify the flash device for `.rpd` files because the `.rpd` format is independent of the flash device. In contrast, the `.pof` and `.jic` files include both programming data and additional data specific to the configuration device. The Intel Quartus Prime Programmer uses this additional data to establish communication with the configuration device and then write the programming data.

9. Click **Generate** to generate the programming file or files.

Related Information

Intel Quartus Prime Pro Edition User Guide: Programmer
For comprehensive information about programming file generation and conversion.

3.2.10.2. Programming .pof files into Serial Flash Device

To program the `.pof` into the serial flash device through the AS header, perform the following steps:

1. In the **Programmer** window, click **Hardware Setup** and select the desired download cable.
2. In the **Mode** list, select **Active Serial Programming**.
3. Click **Auto Detect** button on the left pane.
4. Select the device to be programmed and click **Add File**.
5. Select the `.pof` to be programmed to the selected device.
6. Click **Start** to start programming.

3.2.10.3. Programming .jic files into Serial Flash Device

To program the `.jic` into the serial flash device through the JTAG interface, perform the following steps:

1. In the **Programmer** window, click **Hardware Setup** and select the desired download cable.
2. In the **Mode** list, select **JTAG**.
3. Select the device to be programmed and click **Add File**.
4. Select the `.jic` to be programmed to the selected device.
5. Click **Start** to start programming.

3.2.11. Debugging Guidelines for the AS Configuration Scheme

The AS configuration scheme operation is like earlier device families. However, there is one significant difference. Intel Agilex devices using AS mode, try to load a firmware section from addresses 0, 512, 1024 and 1536 in the serial flash device connected to the CS0 pin.

If the configuration bitstream does not include a valid image, the SDM asserts an error by driving nSTATUS low. You can recover from the error by reconfiguring the FPGA over JTAG, or by driving nCONFIG low.

SDM tristates AS pins, AS_CLK, AS_DATA0-AS_DATA3, and AS_nCS0-AS_nCS3, only when the device powers on if you set MSEL to JTAG. If MSEL is either AS fast or normal, the SDM drives the AS pins until you power cycle the Intel Agilex device. Unlike earlier device families, the AS pins are not tristated when the device enters user mode.

The AS configuration scheme has power-on requirements. If you use AS Fast mode and are not concerned about 100 ms PCIe link training requirement, you must still ramp the V_{CCIO_SDM} supply within 18 ms. This ramp-up requirement ensures that the AS x4 device is within its operating voltage range when the Intel Agilex device begins assessing the AS x4 device.

When using AS fast mode, all power supplies to the Intel Agilex device must be fully ramped-up to the recommended operating conditions before the SDM releases from reset. To meet the PCIe 100 ms power-up-to-active time requirement for CvP, the V_{CCIO_SDM} power to the Intel Agilex device must be at the recommended operating range within 10 ms.

Debugging Suggestions

Here are some debugging tips for the AS configuration scheme:

- Ensure that the boot address for your configuration image is correctly defined when generating the programming file for the flash. The boot address defaults to 0 for AS configuration.
- Ensure that the design meets the power-supply ramp requirements for fast AS mode. If using fast mode, V_{CCIO_SDM} must ramp up within 18 ms.
- Ensure that the flash is powered up and ready to be accessed when the Intel Agilex device exits power-on reset.
- If you are using an external clock source for configuration, ensure the OSC_CLK_1 pin is fed correctly, and the frequency matches the frequency you set for the OSC_CLK_1 in your Intel Quartus Prime Pro Edition project.
- Ensure the MSEL pins reflect the correct AS configuration scheme.

- If the AS configuration is failing due to a corrupt image inside the serial flash device and reprogramming does not resolve the problem, you have two possible solutions depending on the components you are using for configuration:
  - If you are using a third-party programmer to configure the flash directly from an AS or JTAG header as shown in Figure 38 on page 93 change the MSEL setting to JTAG. Setting MSEL to JTAG prevents the corrupt image from loading automatically at power-on. Then, update the image in quad serial flash through the AS or JTAG header.
  - If you are programming the flash device using the JTAG header as shown in Figure 39 on page 94, force the nCONFIG signal to low. When nCONFIG is low, the image cannot load from the quad SPI flash device. Then, update the image in quad serial flash through the JTAG header.
- If you are using AS x4 flash memories, ensure that you use AS Fast mode, if you are not concerned about 100 ms PCIe linkup, you must still ramp the V_{CCIO_SDM} supply within 18 ms. This ramp-up requirement ensures that the AS x4 device is within its operating voltage range when the Intel Agilex device begins to access it.
- Check endianness of the `.rpd` if using a third-party programmer to program Quad SPI device. You should generate the `.rpd` as big endian.

3.2.12. QSF Assignments for AS

You can specify many Intel Quartus Prime project settings using Intel Quartus Prime Software GUI or by editing the Intel Quartus Prime Settings File (`.qsf`). The following assignments in the `.qsf` show typical settings for a Intel Agilex device using AS configuration.

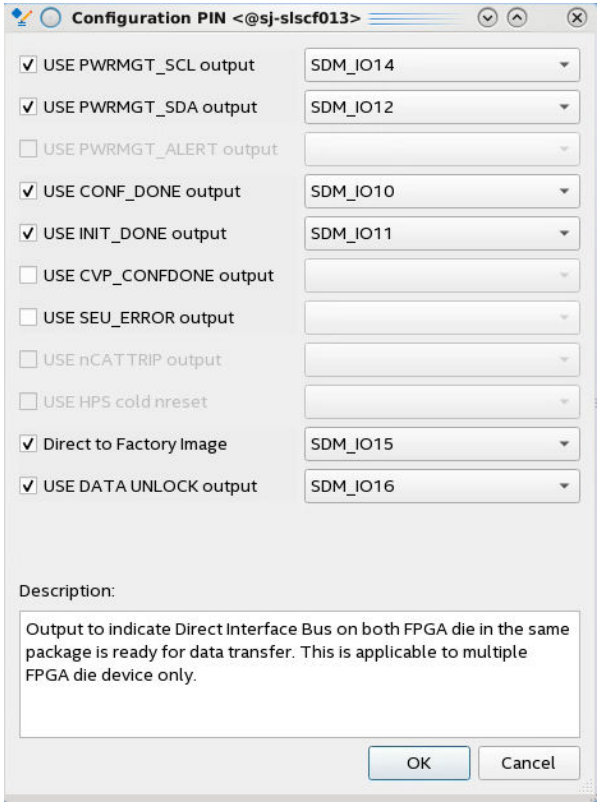
These settings are for a Intel Agilex SmartVID device operating in PMBus slave mode which requires most of the SDM_IO pins. Refer to the *Intel Agilex Power Management User Guide* for the PMBus constraints in master mode.

```
# Fitter Assignments
# ==================
set_global_assignment -name DEVICE 1SG280LU3F50E3VG
set_global_assignment -name CONFIGURATION_VCCIO_LEVEL 1.8V

# SDM IO Assignments
# ==================
set_global_assignment -name USE_PWRMGT_SCL SDM_IO14
set_global_assignment -name USE_PWRMGT_SDA SDM_IO11
set_global_assignment -name USE_PWRMGT_ALERT SDM_IO12
set_global_assignment -name USE_CONF_DONE SDM_IO16
set_global_assignment -name USE_INIT_DONE SDM_IO0
set_global_assignment -name USE_CVP_CONFDONE SDM_IO13
set_global_assignment -name USE_SEU_ERROR SDM_IO10
set_global_assignment -name SDM_DIRECT_TO_FACTORY_IMAGE SDM_IO15
```

```
# Configuration settings
# ======================

# The following setting also supports Intel Agilex devices
set_global_assignment -name STRATIXV_CONFIGURATION_SCHEME "ACTIVE SERIAL X4"
set_global_assignment -name USE_CONFIGURATION_DEVICE ON
set_global_assignment -name ERROR_CHECK_FREQUENCY_DIVISOR 256
set_global_assignment -name GENERATE_PR_RBF_FILE ON
set_global_assignment -name ENABLE_ED_CRC_CHECK ON
set_global_assignment -name MINIMUM_SEU_INTERVAL 479

# SmartVID feature PMBus settings [Slave mode settings only]
# ==========================================================
set_global_assignment -name VID_OPERATION_MODE "PMBUS SLAVE"
set_global_assignment -name PWRMGT_DEVICE_ADDRESS_IN_PMBUS_SLAVE_MODE 3F
```

You can also set the SDM_IO configuration pins using the **Assignments ➤ Device ➤ Device and Pin Options ➤ Configuration ➤ Configuration Pin Options**.

Figure 45. Set SDM_IO Configuration Pins Using the Intel Quartus Prime Software



Related Information

- PMBus Master Mode
  In the *Power Management User Guide*
- Intel Quartus Prime Pro Settings File Reference Manual

3.3. SD/MMC Configuration

Note: Contact your Intel sales representative for information about SD/MMC support.

In the configuration scheme using SD memory cards or MMC, the memory cards store configuration data. The SDM uses the on-chip SD or MMC controller to interface to the memory cards. The SDM block reads the configuration data from the memory cards for the configuration process. The configuration from SD and MMC supports x4 SD memory cards and x8 MMC.

Table 29. Intel Agilex Configuration Data Width, Clock Rates, and Data Rates

Mode		Data Width (bits)	Max Clock Rate	Max Data Rate	MSEL[2:0]
Active	SD/MMC	4 or 8	50 MHz	400 Mb	3'b100

Table 30. Required Configuration Signals for the SD/MMC Configuration Scheme

You specify SDM I/O pin functions using the **Device ➤ Configuration ➤ Device and Pin Options** dialog box in the Intel Quartus Prime software.

Configuration Function		Direction	Powered by
nSTATUS	SDM I/O	Output	V_{CCIO_SDM}
nCONFIG	SDM I/O	Input	V_{CCIO_SDM}
MSEL[2:0]	SDM I/O, Dual-Purpose	Input	V_{CCIO_SDM}
SDMMC_CFG_CMD	GPIO	Output	V_{CCIO_SDM}
SDMMC_CFG_DATA[7:0]	GPIO	Bidirectional	V_{CCIO_SDM}
SDMMC_CFG_CCLK	GPIO	Output	V_{CCIO_SDM}

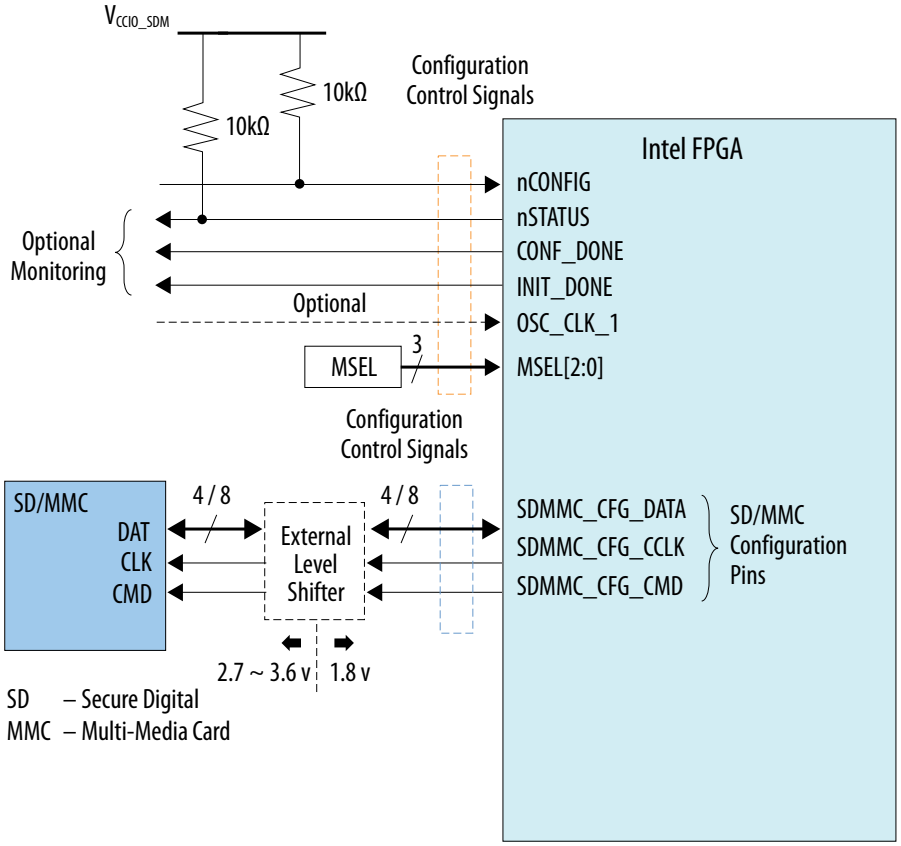
Note: Although the CONF_DONE and INIT_DONE configuration signals are not required, Intel recommends that you use these signals. The SDM drives the CONF_DONE signal high after successfully receiving full bitstream. The SDM drives the INIT_DONE signal high to indicate the device is fully in user mode. You can reassign the GPIO, dual-purpose configuration pins for other functions in user mode.

Related Information

MSEL Settings on page 26

3.3.1. SD/MMC Single-Device Configuration

Figure 46. Connections for SD/MMC Single-Device Configuration



Note: The External Level Shifter is not mandatory for embedded multimedia cards (eMMC).

3.4. JTAG Configuration

JTAG-chain device programming is ideal during development. JTAG-chain device configuration uses the JTAG pins to configure the Intel Agilex FPGA directly with the `.sof` file. Configuration using the JTAG device chain allows faster development because it does not require you to program an external flash memory. You can also use JTAG to reprogram if the image stored in quad SPI memory. You can also use the JTAG configuration scheme to reprogram the quad SPI memory if the quad SPI content is corrupted or invalid.

The Intel Quartus Prime software generates a `.sof` containing the FPGA design information. You can use the `.sof` with a JTAG programmer to configure the Intel Agilex device. The Intel FPGA Download Cable II and the Intel FPGA Ethernet Cable both can support the V_{CCIO_SDM} supply at 1.8 V. Alternatively, you can use the JamSTAPL Format File (`.jam`) or Jam Byte Code File (`.jbc`) for JTAG configuration.

Intel Agilex devices automatically compress the configuration bitstream. You cannot disable compression in Intel Agilex devices.

Table 31. Intel Agilex Configuration Data Width, Clock Rates, and Data Rates

Mode		Data Width (bits)	Max Clock Rate	Max Data Rate	MSEL[2:0]
Passive	JTAG	1	30 MHz	30 Mb	3'b111

Note: The JTAG port has the highest priority and overrides the MSEL pin settings. Consequently, you can configure the Intel Agilex device over JTAG even if the MSEL pin specify a different configuration scheme unless you disabled JTAG for security reasons.

Table 32. Power Rails for the Intel Agilex Device Configuration Pins

You can view the pin assignments for fixed pins in the Pin-Out File for your device. You specify SDM I/O pin functions using the **Device ➤ Configuration ➤ Device and Pin Options** dialog box in the Intel Quartus Prime software.

Configuration Function	Pin Type	Direction	Powered by
TCK	Fixed	Input	V_{CCIO_SDM}
TDI[9]	Fixed	Input	V_{CCIO_SDM}
TMS[9]	Fixed	Input	V_{CCIO_SDM}

continued...

(9) The JTAG pins can access the HPS JTAG chain in Intel Agilex SoC devices.

Configuration Function	Pin Type	Direction	Powered by
TDO[9]	Fixed	Output	V_{CCIO_SDM}
nSTATUS	SDM I/O	Output	V_{CCIO_SDM}
nCONFIG	SDM I/O	Input	V_{CCIO_SDM}
MSEL[2:0]	SDM I/O, Dual-Purpose	Input	V_{CCIO_SDM}

Note: Although the CONF_DONE and INIT_DONE configuration signals are not required, Intel recommends that you use these signals. The SDM drives the CONF_DONE signal high after successfully receiving full bitstream. The SDM drives the INIT_DONE signal high to indicate the device is fully in user mode.

Note: Pin-Out files are not yet available for Intel Agilex devices.Intel Agilex

Related Information

Programming Support for Jam STAPL Language

3.4.1. JTAG Configuration Scheme Hardware Components and File Types

The following figure illustrates JTAG programming. This is the simplest device configuration scheme. You do not have to use the **File ➤ Programming File Generator** to convert the .sof file to a .pof.

Figure 47. JTAG Configuration Scheme

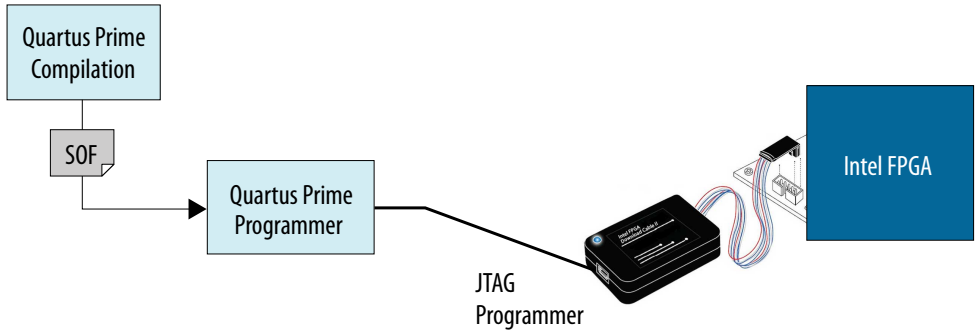
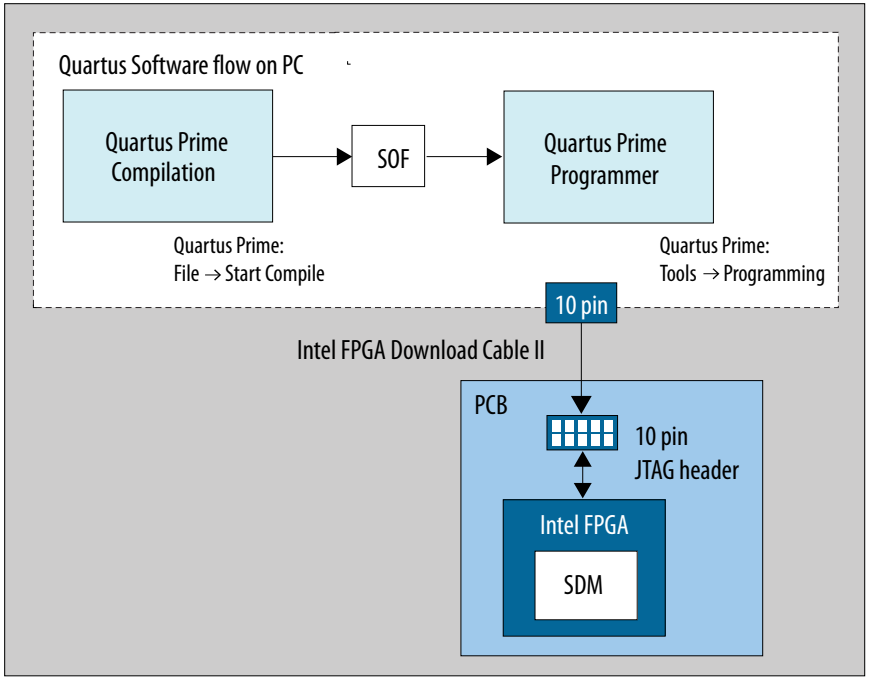


Figure 48. Components and Design Flow for JTAG Programming



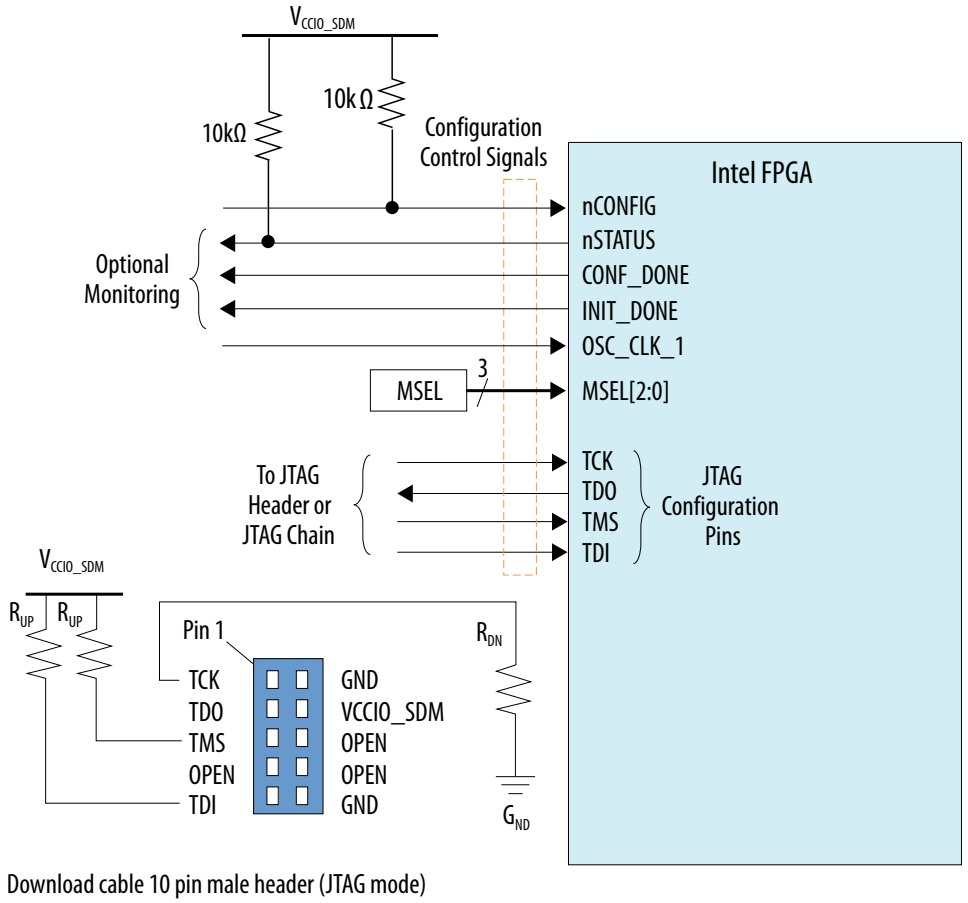
3.4.2. JTAG Device Configuration

To configure a single device in a JTAG chain, the programming software sets the other devices to bypass mode. A device in bypass mode transfers the programming data from the TDI pin to the TDO pin through a single bypass register. The configuration data is available on the TDO pin one clock cycle later.

You can configure the Intel Agilex device through JTAG using a download cable or a microprocessor.

3.4.2.1. JTAG Single-Device Configuration using Download Cable Connections

Figure 49. Connection Setup for JTAG Single-Device Configuration using Download Cable



Download cable 10 pin male header (JTAG mode)

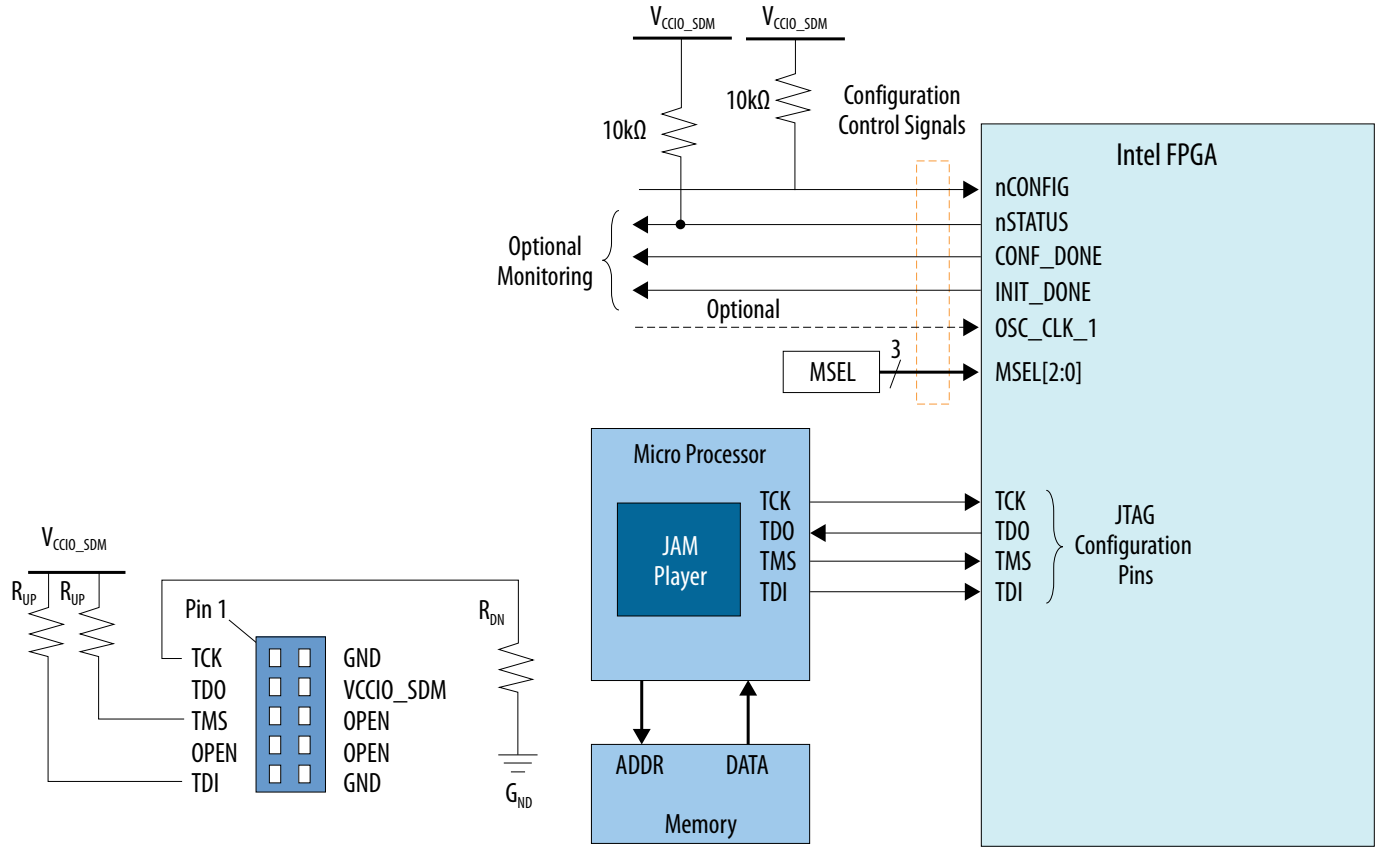
Related Information

Intel FPGA Download Cable II User Guide

3.4.2.2. JTAG Single-Device Configuration using a Microprocessor

Refer to the *Intel Agilex Device Family Pin Connection Guidelines* for additional information about individual pin usage and requirements.

Figure 50. Connection Setup for JTAG Single-Device Configuration using a Microprocessor



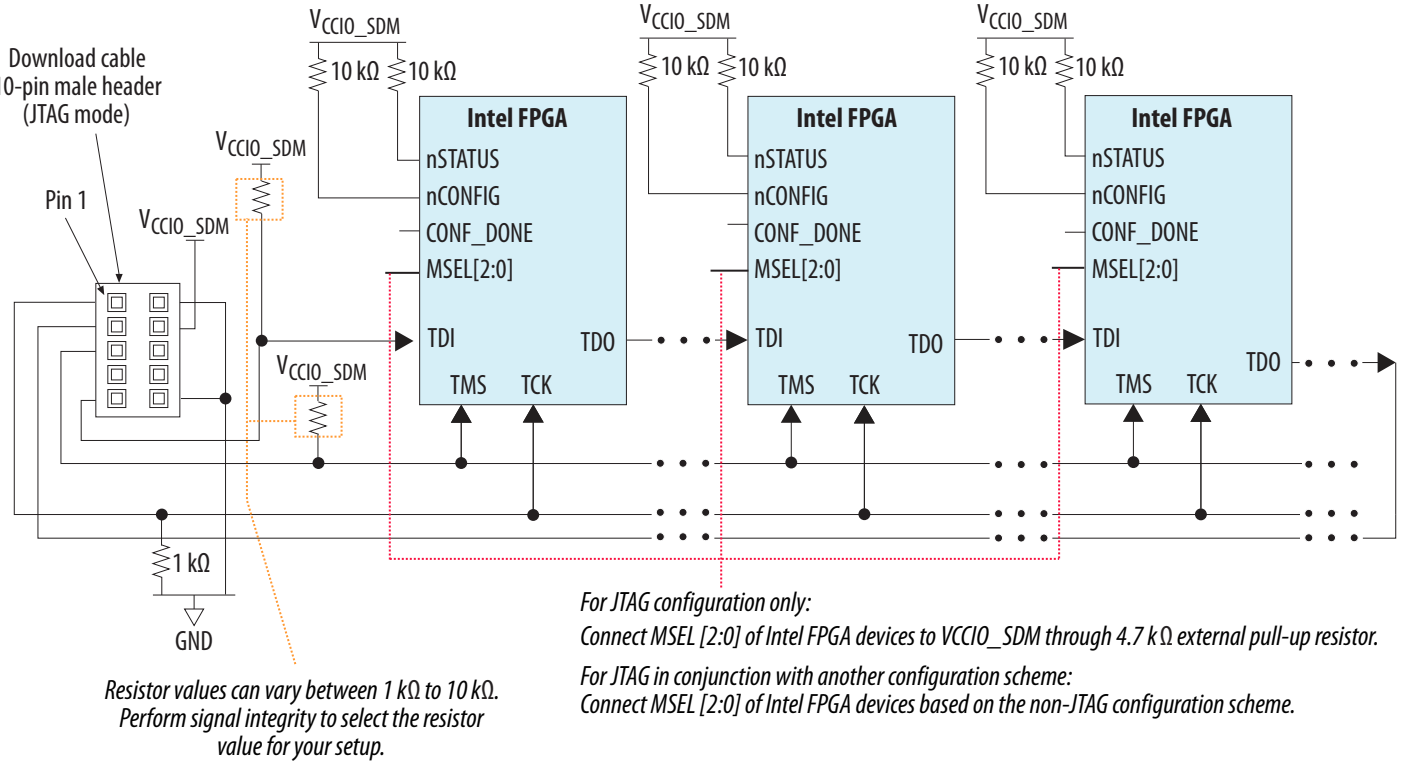
3.4.3. JTAG Multi-Device Configuration

You can configure multiple devices in a JTAG chain. Observe the following pin connections and guidelines for this configuration setup:

- One JTAG-compatible header connects to several devices in a JTAG chain. The drive capability of the download cable is the only limit on the number of devices in the JTAG chain.
- If you have four or more devices in a JTAG chain, buffer the TCK, TDI, and TMS pins with an on-board buffer. You can also connect other Intel FPGA devices with JTAG support to the chain.

3.4.3.1. JTAG Multi-Device Configuration using Download Cable

Figure 51. Connection Setup for JTAG Multi Device Configuration using Download Cable



3.4.4. Debugging Guidelines for the JTAG Configuration Scheme

The JTAG configuration scheme overrides all other configuration schemes. The SDM is always ready to accept configuration over JTAG unless a security feature disables the JTAG interface. JTAG is particularly useful in recovering a device that may be in an unrecoverable state reached when trying to configure using a corrupted image.

An nCONFIG falling edge terminates any JTAG access and the device reverts to the MSEL-specified boot source. nCONFIG must be stable during JTAG configuration. nSTATUS follows nCONFIG during JTAG configuration. Consequently, nCONFIG also must be stable.

Unlike other configuration schemes, nSTATUS does not assert if an error occurs during JTAG configuration. You must monitor the error messages that the Intel Quartus Prime Pro Edition Programmer generates for error reporting.

Note: For Intel Agilex SX devices when you choose to configure the FPGA fabric first, the JTAG chain has no mechanism to redeliver the HPS boot information following a cold reset. Consequently, you must reconfig the device with the .sof file or avoid cold resets to continue operation.

Debugging Suggestions

Here are some debugging tips for JTAG:

- Verify that the JTAG pin connections are correct.
- If JTAG configuration is failing, check that the FPGA has successfully powered up and exited POR. One strategy is to check the hand shaking behavior between nCONFIG and nSTATUS by driving nCONFIG low and ensuring that nSTATUS also goes low.
- Verify that the nCONFIG pin remains high during JTAG configuration.
- Another way to determine whether the device has exited the POR state is to use the Intel Quartus Prime Programmer to detect the device. If the programmer can detect the Intel Agilex device, it has exited the POR state.
- If you are using an Intel FPGA Download Cable II, reduce the cable clock speed to 6 MHz.
- If you have multiple devices in the JTAG chain, try to disconnect other devices from the JTAG chain to isolate the Intel Agilex device.
- If you specify the OSC_CLK_1 as the clock source for configuration, ensure that OSC_CLK_1 is running at the frequency you specify in the Intel Quartus Prime software.
- For designs including the High Bandwidth Memory (HBM2) IP or any IP using transceivers, you must provide a free running and stable reference clock to the device before device configuration begins. All transceiver power supplies must be at the required voltage before configuration begins.
- When the MSEL setting on the PCB is not JTAG, if you use the JTAG interface for reconfiguration after an initial reconfiguration using AS or the Avalon-ST interface, the .sof must be in the file format you specified in the Intel Quartus Prime project. For example, if you initially configure the MSEL pins for AS configuration and configure using the AS scheme, a subsequent JTAG reconfiguration using a .sof generated for Avalon-ST fails.

4. Including the Reset Release Intel FPGA IP in Your Design

Intel requires that you either use the Reset Release Intel FPGA IP or the INIT_DONE signal routed back in through a pin to hold your design in reset until configuration is complete.

The Reset Release Intel FPGA IP is available in the Intel Quartus Prime Software. This IP consists of a single output signal, nINIT_DONE. The nINIT_DONE signal is the core version of the INIT_DONE pin and has the same function in both FPGA First and HPS First configuration modes. Intel recommends that you hold your design in reset while the nINIT_DONE signal is high or while the INIT_DONE pin is low. When you instantiate the Reset Release IP in your design, the SDM drives the nINIT_DONE signal. Consequently, the IP does not consume any FPGA fabric resources, but does require routing resources.

Figure 52. Reset Release Intel FPGA IP nINIT_DONE Internal Connection

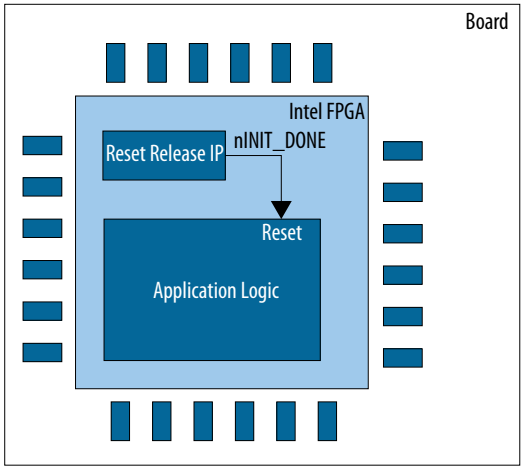
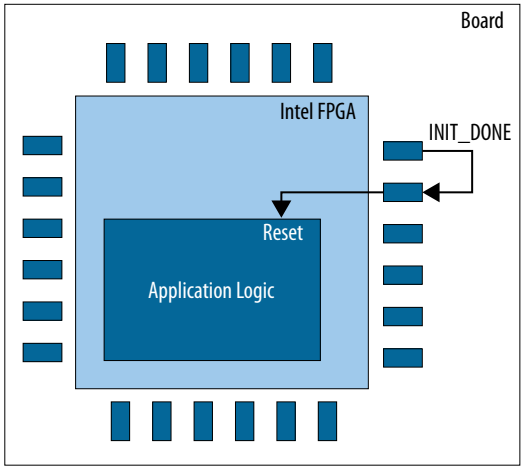


Figure 53. Reset Release Intel FPGA IP INIT_DONE External Connection

If you do not include the Reset Release Intel FPGA IP in your design, you must feed the INIT_DONE signal back into your design as an input to your reset logic as shown in this figure.



4.1. Understanding the Reset Release IP Requirement

Intel Agilex devices use a parallel, sector-based architecture that distributes the core fabric logic across multiple sectors. Device configuration proceeds in parallel with each Local Sector Manager (LSM) configuring its own sector. Consequently, FPGA registers and core logic do not exit reset at exactly the same time, as has always been the case in previous families.

The continual increases in clock frequency, device size, and design complexity now necessitate a reset strategy that considers the possible effects of slight differences in the release from reset. The Reset Release Intel FPGA IP holds a control circuit in reset until the device has fully entered user mode. The Reset Release FPGA IP generates an inverted version of the internal INIT_DONE signal, nINIT_DONE for use in your design.

After nINIT_DONE asserts (low), all logic is in user mode and operates normally. You can use the nINIT_DONE signal in one of the following ways:

- To gate an external or internal reset.
- To gate the reset input to the transceiver and I/O PLLs.
- To gate the write enable of design blocks such as embedded memory blocks, state machine, and shift registers.
- To synchronously drive register reset input ports in your design.

Attention: When you instantiate Reset Release Intel FPGA IP in your design, the Intel Quartus Prime Fitter selects one Local Sector Manager (LSM) to output the nINIT_DONE signal. An Intel Quartus Prime Pro Edition legality check prevents you from instantiating more than on instance of the Reset Release Intel FPGA IP. Multiple instances results in some skew between the nINIT_DONE signals.

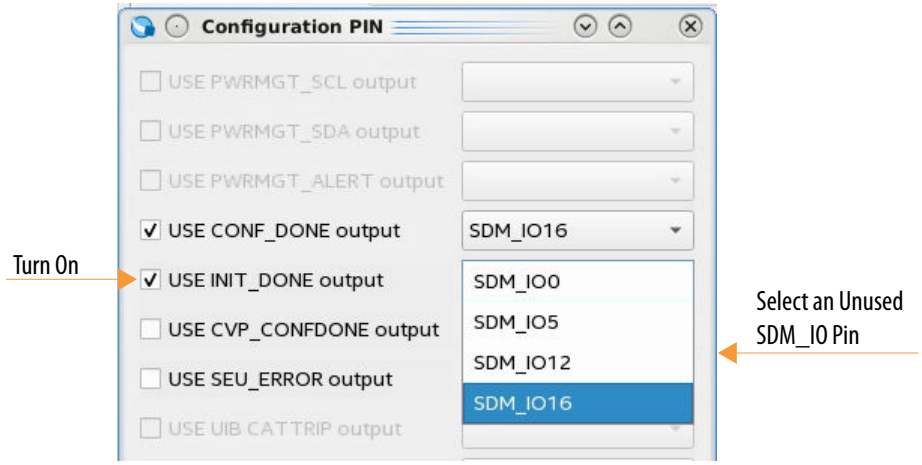
4.2. Assigning INIT_DONE To an SDM_IO Pin

If you choose to route INIT_DONE to an external pin, you must assign INIT_DONE to an SDM_IO pin.

Complete the following steps to make this assignment.

1. On the Intel Quartus Prime Assignments menu, select **Device ➤ Device and Pin Options ➤ Configuration Pin**, turn on the **Use INIT_DONE** output.
2. In the drop-down list, select any SDM_IO pin that is available.

Figure 54. Assigning INIT_DONE to SDM_IO Pin



Note: The Reset Release IP generates the `nINIT_DONE` internal signal whether or not you choose to assign `INIT_DONE` to an SDM_IO pin.

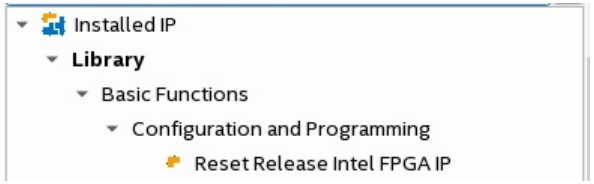
4.3. Instantiating the Reset Release IP In Your Design

The Reset Release IP is available in the IP Catalog in the **Basic Functions ➤ Configuration and Programming** category. This IP has no parameters.

Complete the following steps to instantiate the Reset Release IP in your design.

1. In the IP Catalog, type `reset release` in the search window to find the Reset Release Intel FPGA IP.

Figure 55. Locate Reset ReleaseIntel FPGA IP in IP Catalog

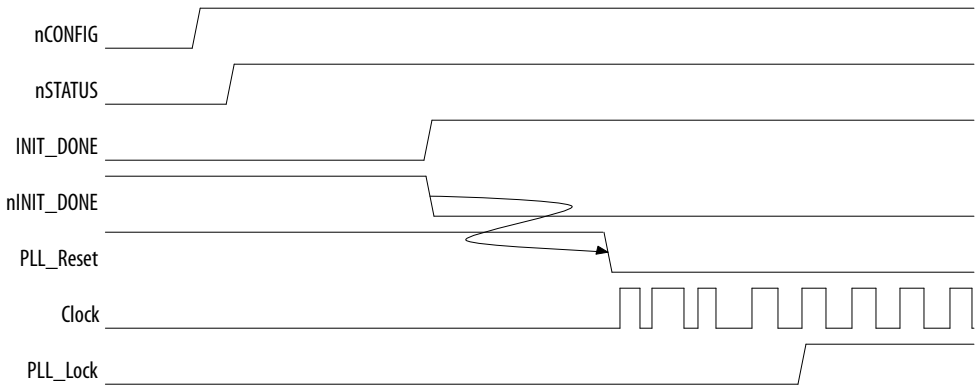


2. Double click the **Reset Release Intel FPGA IP** to add the Reset Release IP to your design.
3. In the **New IP Variant** dialog box, browse to your IP directory and specify a file name for the Reset Release IP. Then click **Create**. The Reset Release IP is now included in your project.

4.4. Gating the PLL Reset Signal

In older FPGA device families, designs frequently used the PLL lock signal to hold the custom FPGA logic in reset until the PLL locked. In newer Intel device families the lock time of PLLs can be less than the initialization time. In some cases the PLL may lock before the device completes initialization. Consequently, if you use the locked output of the PLL to control resets in the Intel Agilex device, you should gate the PLL reset input with `nINIT_DONE` as shown the figure.

Figure 56. Using nINIT_DONE to Gate the PLL_Reset Signal



Another alternative if you are using `PLL_Lock` in your reset sequence is to gate the `PLL_Lock` output with the `nINIT_DONE` signal, `(PLL_Lock && !nINIT_DONE)`.

4.5. Guidance When Using Partial Reconfiguration (PR)

The PR Region Controller IP provides reset logic that ensures that the static region of the device and the PR personas do not interact during PR.

The Reset Release IP is only necessary to manage reset for full FPGA core configuration and subsequent full FPGA core configurations. The Reset Release IP is not necessary to prevent interaction between the static and PR personas during the PR process. For more information about PR refer to the *Intel Quartus Prime Pro Edition User Guide: Partial Reconfiguration*.

Related Information

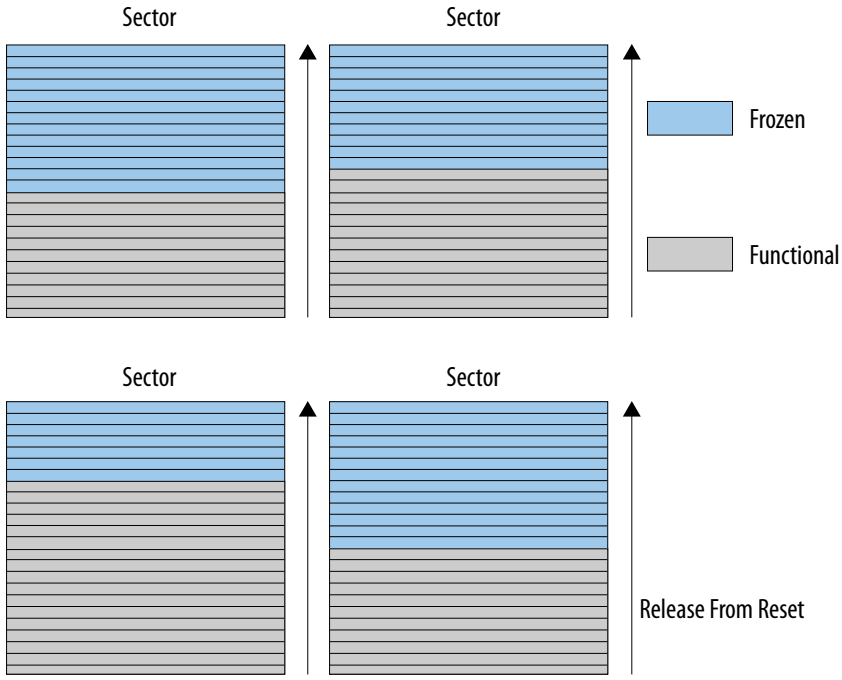
Creating a Partial Reconfiguration Design

4.6. Detailed Description of Device Configuration

Each Local Sector Manager (LSM) configures its own sector. A sector comprises multiple logic array block (LAB) rows. A logical function can span multiple rows and multiple sectors.

During configuration global configuration control signals hold the core fabric in a frozen state to prevent electrical contention. The LSMs work in parallel to asynchronously unfreeze the sectors. Within a sector the LSM unfreezes LAB rows and registers in the LABs sequentially. The LSMs work to unfreeze the fabric in parallel across all sectors without synchronization. Consequently, logic in different sectors or in the same sector but in different rows could begin to operate while other logic is still frozen. The INIT_DONE signal asserts when all the LSMs have entered user mode.

Figure 57. Releasing LAB Rows and Registers in the LABs Sequentially and Asynchronously Across Sectors



The following topics provide more detail about device configuration and initialization, and possible consequences if you do not use the Reset Release IP to hold the Intel Agilex device in reset until entire fabric enters user mode.

4.6.1. Device Initialization

The following steps summarize device initialization:

1. An external host drives a configuration request to the Secure Device Manager (SDM) by driving nCONFIG high. The SDM exits the IDLE state and signals the beginning of configuration by driving nSTATUS high and driving configuration data.
2. The SDM asserts CONF_DONE indicating that the Intel FPGA has successfully received all the configuration data.
3. The SDM uses the configuration logic to start non-gated clocks in the fabric. Intel Hyperflex™ registers begin shifting data. Consequently, the initial conditions of Intel Hyperflex registers can be random. Use the **Disable Register Power-up Initialization** setting in the Intel Quartus Prime **Configuration** dialog box to disable Intel Hyperflex register initialization during power-on as explained below.
4. The SDM uses the configuration logic to enable and initialize user registers in the LABs, DSP, and embedded memory blocks.
5. The SDM drives INIT_DONE to indicate that the device has fully entered user mode. The Reset Release IP asserts nINIT_DONE. Intel recommends that you use nINIT_DONE to gate your reset logic.
6. The FPGA is now in user mode and ready for operation.

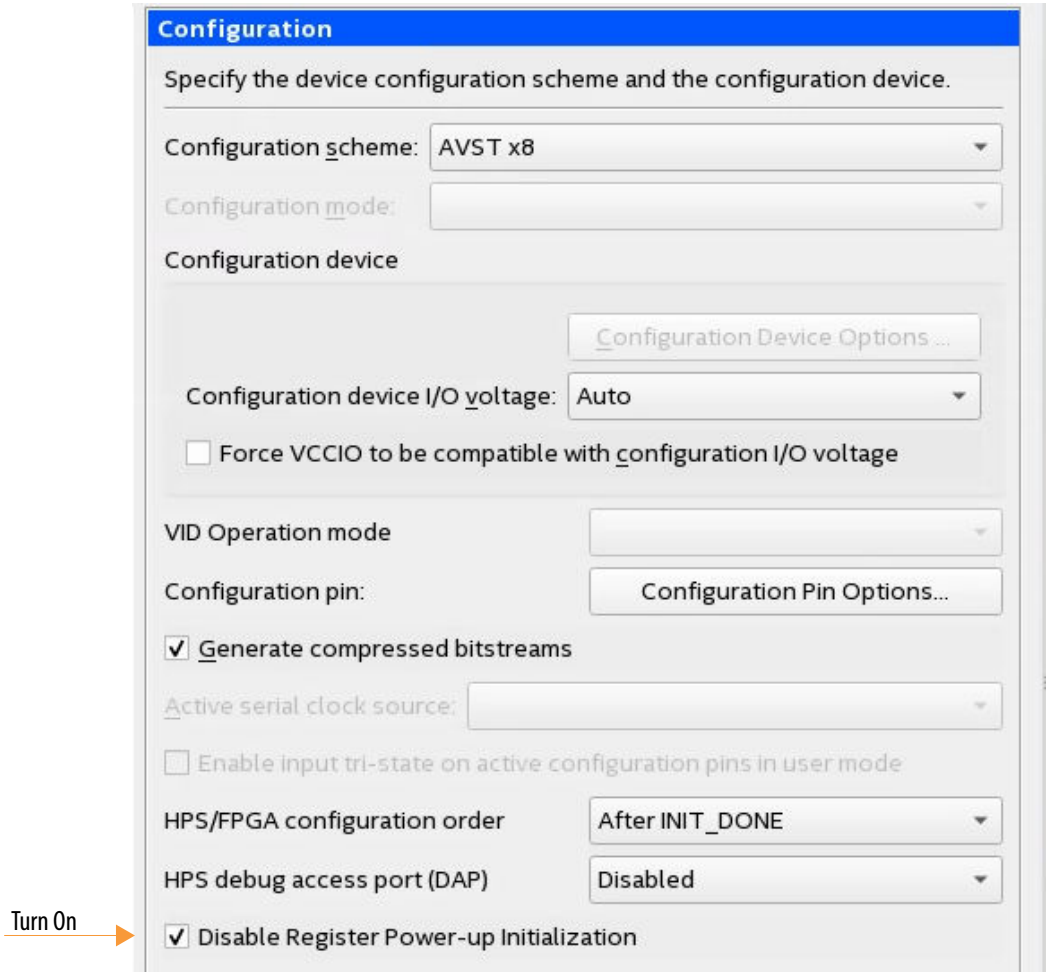
4.6.2. Preventing Register Initialization During Power-On

If not held in reset, both ALM and Intel Hyperflex registers may lose their initial state if they initialize before their respective source.

You can prevent registers from initializing during power-on by enabling an option in the Intel Quartus Prime software. Complete the following steps to turn on this option:

1. On the Assignments menu select **Device ➤ Device and Pin Options ➤ Configuration**.
2. In the **Configuration** dialog box, turn on **Disable Register Power-up Initialization.**

Figure 58. Disabling Register Initialization During Power-On



Note: Coming out of reset, you cannot rely on the value of registers with initial conditions unless you gate your system reset using one of the following options:

- Designs that include the Reset Release IP must route `nINIT_DONE` to the system reset.
- Designs that do not include the Reset Release IP must route `INIT_DONE` to an external pin and feed `INIT_DONE` back into the FPGA as an input to system reset.

4.6.3. Embedded Memory Block Initial Conditions

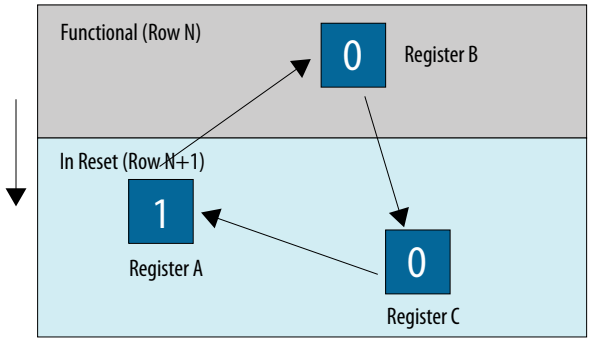
Initialized content of embedded memory blocks is stable during configuration. However, designs that contain logic to modify embedded memory can result in spurious writes. Spurious writes can occur if you fail to gate the write enable with an appropriate reset.

4.6.4. Protecting State Machine Logic

To guarantee correct operation of state machines, your reset logic must hold the FPGA fabric in reset until the entire fabric enters user mode.

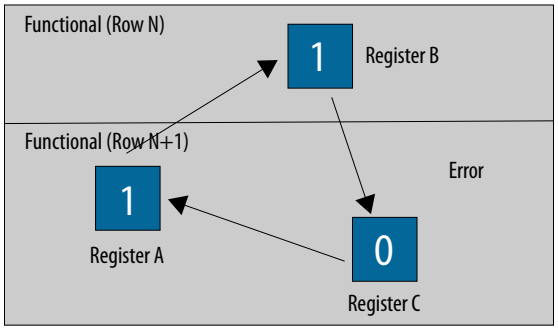
The following example shows how an inadequate reset strategy might result in an illegal state in a one-hot state machine. In this example, the design does not reset any of the state machine registers. The state machine design depends on registers entering an initial state. Without an adequate reset, this state machine begins operating when part of the device is active. Nearby logic included in the state machine remains frozen, before `INIT_DONE` asserts.

Figure 59. Partially Initialized Design - INIT_DONE = 0



Register B in the active section is operational and takes on the value of Register A in the next clock cycle. Register A is still in the freeze register state and does not respond to the clock edge. Register A remains in the current state.

Figure 60. Advance One Clock Cycle, Device Completely In User Mode - INIT_DONE = 1



The entire fabric is now in user mode. The state machine enters an illegal or unknown state with two ones in a one-hot state machine. To prevent this illegal state, use the Reset Release IP to hold the circuit in reset until INIT_DONE asserts indicating that the entire fabric has entered user mode.

5. Remote System Update (RSU)

RSU implements device reconfiguration using dedicated RSU circuitry available in all Intel Agilex devices. RSU has the following advantages:

- Provides a mechanism to deliver feature enhancements and bug fixes without recalling your products
- Reduces time-to-market
- Extends product life

Using RSU and the Mailbox Client Intel FPGA IP you can write configuration bitstreams to the AS x4 flash device. Then you can use the Mailbox Client Intel FPGA IP to instruct the SDM to restart from the updated image. You can store multiple application images and a single factory image in the configuration device. Your design manages remote updates of the application images in the configuration device.

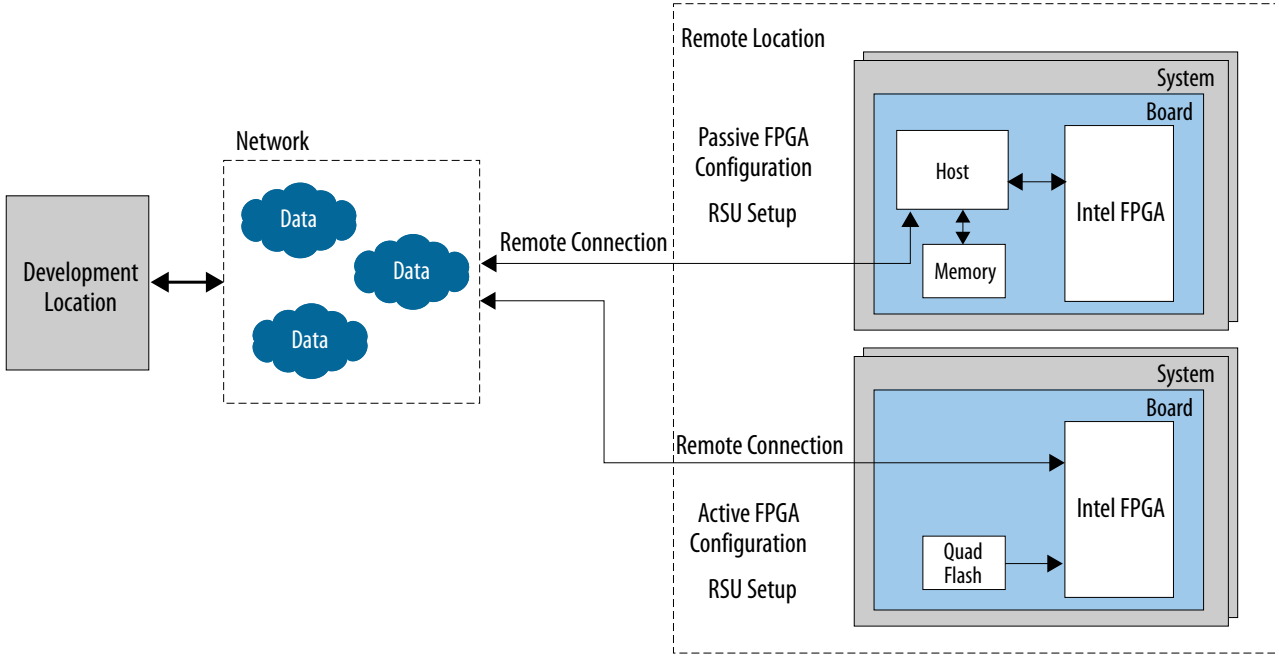
A command to the Mailbox Client Intel FPGA IP initiates reconfiguration. The RSU performs configuration error detection during and after the reconfiguration process. If errors in the application image or images prevent reconfiguration, the configuration circuitry reverts to the factory image and provides error status information.

This chapter explains the remote system update implementation for active configuration schemes. The FPGA drives the RSU. For the Intel Agilex SoC devices, HPS can drive the RSU process.

For passive configuration schemes, an external host implements remote system update rather than the Intel Agilex device. To learn more about remote system update for passive configuration schemes, refer to *Remote Update Intel FPGA IP User Guide* for remote system update implementations in earlier device families.

The following figure shows functional diagrams for typical remote system update processes.

Figure 61. Typical Remote System Update Process



Note: An Intel Agilex version of the *Intel Stratix® 10 SoC Remote System Update (RSU) User Guide* is not yet available. The RSU SoC implementation in Intel Agilex is very similar to the implementation in Intel Stratix 10.

Related Information

Remote Update Intel FPGA IP User Guide

5.1. Remote System Update Functional Description

5.1.1. RSU Glossary

Table 33. RSU Glossary

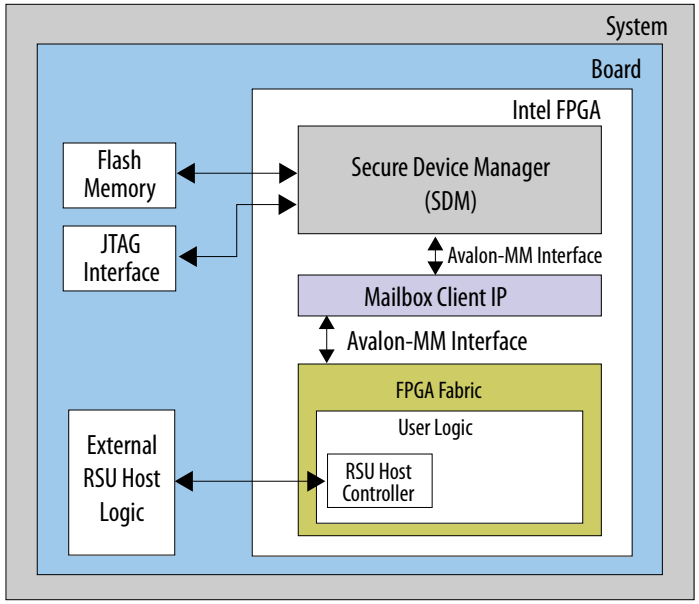
Term	Meaning
Firmware	Firmware that runs on SDM. Implements many functions including the functions listed here: • FPGA configuration • Voltage regulator configuration • Temperature measurement • HPS software load • HPS Reset • RSU • Read, erase, and program flash memory • Device security, including authentication and encryption
Decision firmware	Firmware to identify and load the highest priority image. Previous versions of this user guide refer to *decision firmware* as *static firmware*. Starting in version 19.1 of the Intel Quartus Prime software, you can use RSU to update this firmware.
Decision firmware data	Decision firmware data structure containing the following information: • The Direct to Factory Image pin assignment. • PLL settings for the external clock source. This optional clock source drives OSC_CLK_1. For more information, refer to *OSC_CLK_1 Clock Input*. • Quad SPI pins.
Configuration pointer block (CPB)	A list of application image addresses in order of priority. When you add an image, that image becomes the highest priority.
Sub-partition table (SPT)	Data structure to facilitate the management of the flash storage.
Application image	Configuration bitstream that implements your design. This image includes the SDM firmware.
Factory image	The fallback configuration bitstream that the RSU loads when all attempts to load an application image fail.
Initial RSU flash image	Contains the factory image, the application images, the decision firmware, and the associated RSU data structures.
Factory update image	An image that updates the following RSU-releated items in flash: • The factory image • The decision firmware • The decision firmware data

5.1.2. Remote System Update Using AS Configuration

Remote system update using AS configuration includes the following components:

- Your remote system update host design. The host can be custom logic, the HPS, or a Nios® II processor in the FPGA.
- One factory image.
- Flash memory for image storage.
- At least one application image.
- Designs that do not use the HPS as the remote system update host require an Mailbox Client Intel FPGA IP as shown in the figure below. The Mailbox Client sends and receives remote system update operation commands and responses, such as `QSPI_READ` and `QSPI_WRITE`.

Figure 62. Intel Agilex Remote System Update Components



5.1.3. Remote System Update Configuration Images

Intel Agilex devices using remote system update require the following configuration images:

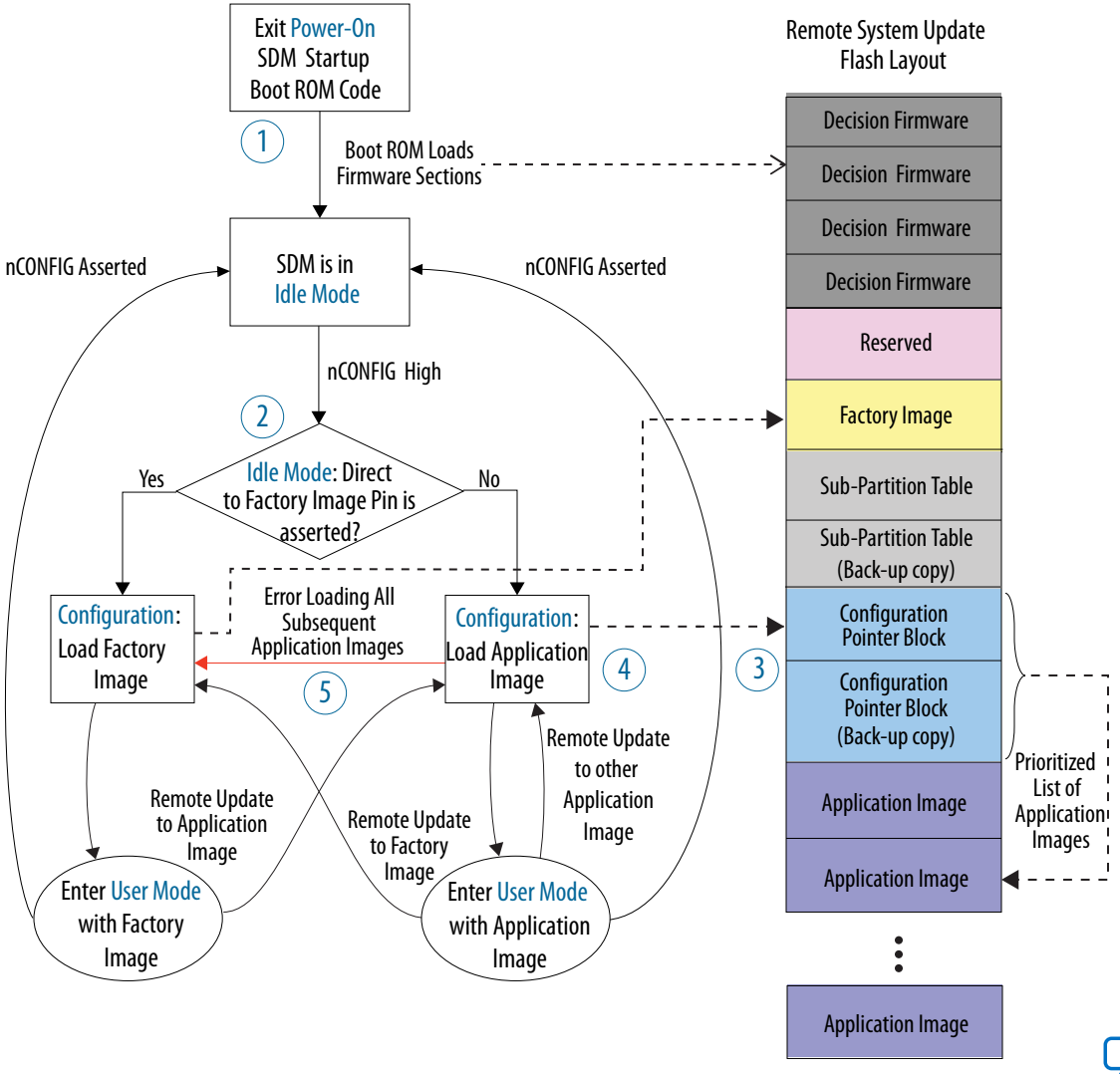
- A Factory image—This image includes logic to implement the following functions:
  - Your design-specific logic to obtain new application images
  - Your design-specific logic to request reconfiguration using a specific application image
  - Image storage in flash memory
- Application image—contains logic to implement the custom application. The application image must also contain logic to obtain new application images and store the images in the flash memory.

Depending on the storage space of your flash memory, Intel Agilex remote system update supports one factory image and up to 507 application images. The Quartus Programming File Generator only supports up to seven remote system update images. However, you can add more images using the Mailbox Client IP with the device in user mode.

5.1.4. Remote System Update Configuration Sequence

Figure 63. Remote System Update Configuration Sequence

In the following figure the blue text are states shown in the Configuration Flow Diagram on page 21.



Reconfiguration includes the following steps:

1. After the device exits power-on-reset (POR), the boot ROM loads flash memory from the first valid decision firmware from one of the copies at addresses 0, 512 K, 1024 K, or 1536 K to initialize the SDM. The same configuration firmware is present in each of these locations. This firmware is part of the initial RSU flash image. (Refer to Step 2 of Guidelines for Performing Remote System Update Functions for Non-HPS on page 139 for step-by-step details for programming the initial RSU flash image into the flash.)
2. The optional Direct to Factory pin controls whether the SDM firmware loads the factory or application image. You can assign the Direct to Factory input to any unused SDM pin. The SDM loads the application image if you do not assign this pin.
3. The configuration pointer block in the flash device maintains a list of pointers to the application images.
4. When loading an application image, the SDM traverses the pointer block in reverse order. The SDM loads the highest priority image. When image loading completes, the device enters user mode.
5. If loading the newest (highest priority) image is unsuccessful, the SDM tries the next application image from the list. If none of the application loads successfully, the SDM loads the factory image.
6. If loading the factory image fails, you can recover by reprogramming the quad SPI flash with the initial RSU flash image using the JTAG interface.

5.1.5. RSU Recovery from Corrupted Images

When an RSU fails, the Mailbox Client Intel FPGA IP RSU_STATUS command provides information about the current configuration status, including the currently running image and most recent failing image. The `rsu1.tcl` script implements the RSU_STATUS commands. You can download the `rsu1.tcl` script from the following web page. Under Device Configuration Support Center, click **Advanced Configuration Features**, click the triangle next to **Remote System Upgrade** to expand this section, then click Example of Tcl Script.

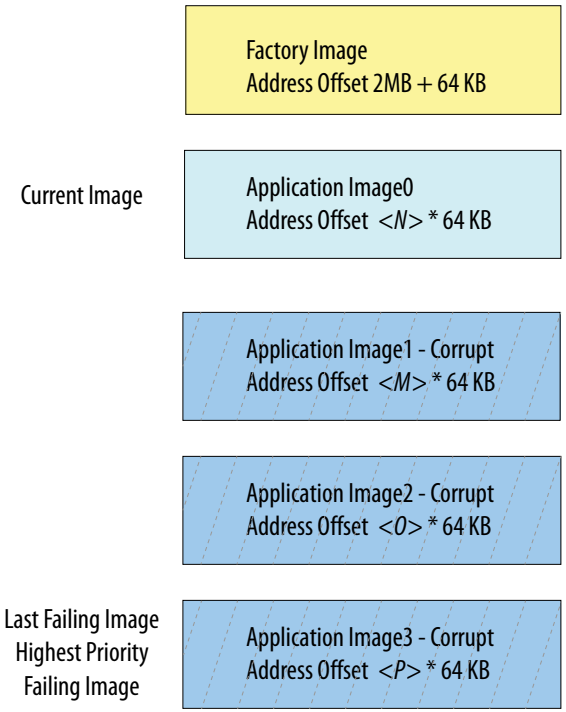
The following example illustrates recovery from a corrupted image:

Multiple Corrupted Images

If the flash memory includes multiple corrupted images, the RSU_STATUS only reports status for the highest priority failing image. The following example illustrates this procedure.

- The flash memory includes the following four images, in order of priority:
  1. Application Image3 (highest priority)
  2. Application Image2
  3. Application image1
  4. Application image0 (lowest priority)
- Application Image3, Application Image2, and Application Image1 are corrupted.
- RSU_STATUS includes the following information:
  - Current_Image: Application Image0
  - `Highest priority failing image`, `State`, `Version`, `Error location`, `Error details`: records information for Application Image3 which is the highest priority failing image.

Figure 64. Multiple Corrupt Images



Related Information

Operation Commands on page 142

5.1.6. Updates with the Factory Update Image

In rare instances you may need to update flash memory with a new factory image and the associated decision firmware and decision firmware data.

An update may be required for the following reasons:

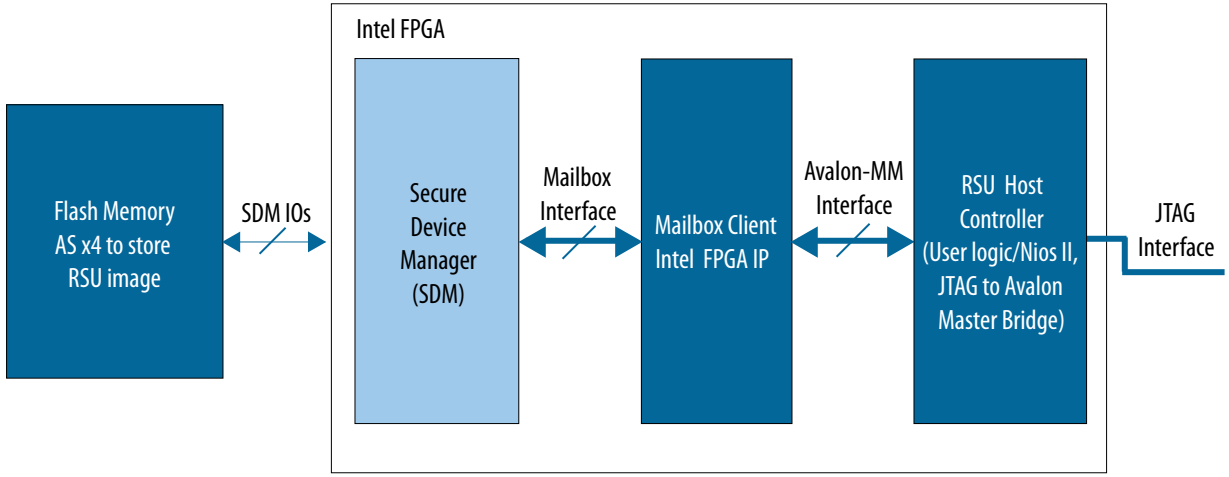
- If there are vulnerabilities in the firmware
- If there are errors in the firmware or in the factory image

Intel provides a safe solution for you to update the factory image and the associated decision firmware and decision firmware data remotely. The update process stores multiple copies of critical data so that if power is lost or the update is disrupted, the device is still able to restart and continue the update. The update continues automatically when power is restored. Here are the steps to perform the update:

1. Generate the factory update image using the Programming File Generator. The image contains the new factory Image, decision firmware, and decision firmware data.
2. Program the factory update image, (`*.rpd`) to an empty partition slot starting from a new sector boundary in the flash device.
3. Trigger reconfiguration to load the update image from the starting address specified in step 2.
4. The updated image performs the following operations:
   a. Erases and replaces the previous decision firmware and decision firmware data in the flash device.
   b. Reprograms the new factory image in the flash device.
   c. After the update completes, the updated image removes itself from the CPB and loads the application image or the factory image if an application image is not available.
5. If the update process used an application slot, you must restore the application image by writing the application image `.rpd` to the application slot and the CPB.

5.2. Guidelines for Performing Remote System Update Functions for Non-HPS

Figure 65. Intel Agilex Modules and Interfaces to Implement RSU Using Images Stored in Flash Memory



Here are guidelines to follow when implementing remote system update:

1. The factory or application image must at least contain a remote system update host controller and the Mailbox Client Intel FPGA IP.
   - You can use either custom logic, the Nios II processor, or the JTAG to Avalon Master Bridge IP as a remote system update host controller.
   - The remote system update host controller controls the remote system update function by sending commands to and receiving responses from the SDM via Mailbox Client Intel FPGA IP. The Mailbox Client functions as the messenger between the remote system update host and SDM. It passes the commands to and responses from the SDM.
2. The pre-generated standard remote system update image file should include a factory image and at least one application image. The remote system update image must be programmed into the flash memory. You can use a dummy image to begin developing RSU functionality before the actual application image is complete. In user mode you can program additional application images.

- Refer to Generating Remote System Update Image Files Using the Programming File Generator on page 161 for the step by step process to generate the standard and single remote system update image files using the programming file generator.

3. The remote system update requires you to use the AS x4 configuration scheme to configure the FPGA with the pre-generated remote system update image.
4. Once the device enters user mode with either the factory image or an application image, the remote system update host can perform the following remote system update operations:
   a. Reconfiguring the device with an application or factory image:
      i. From factory image to an application image or vice versa
      ii. From an application image to another application image
   b. Erasing the application image
   c. Adding an application image
   d. Updating an application or factory image

5.3. Commands and Responses

The remote system update host communicates with the SDM using command and response packets via the Mailbox Client Intel FPGA IP.

Block Diagram

Intel FPGA IPThe following figure illustrates the role of the Mailbox Client Intel FPGA IP in a Intel Agilex design. The Mailbox Client IP enables communication with the SDM to access quad SPI flash memory and system status.

Mailbox Client Role

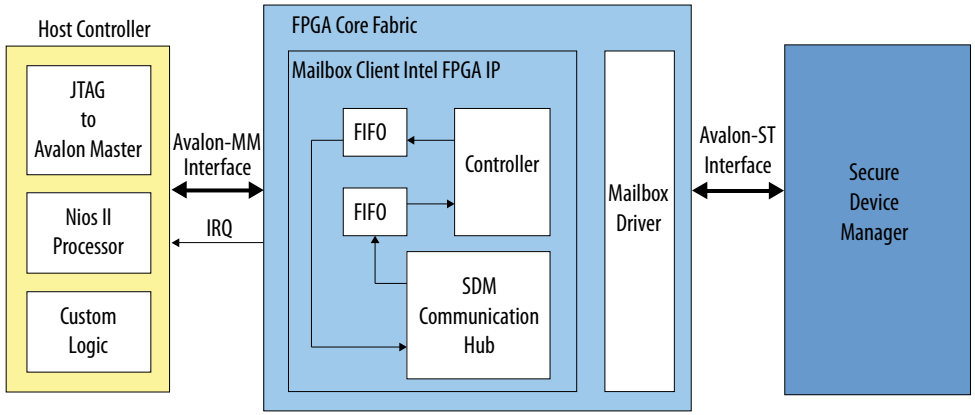
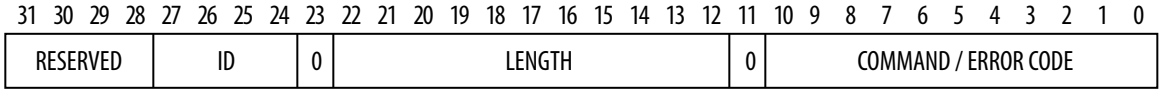


Figure 66. Command and Response Header Format



Note: The LENGTH field in the command header must match the command length of corresponding command.The following table describes the fields of the header command. Your client must read all the response words, even if your client does not interpret all the response words.

Table 34. Mailbox Client Intel FPGA IP Command and Response Header Description

Header	Bit	Description
Reserved	[31:28]	Reserved.
ID	[27:24]	The command ID. The response header returns the ID specified in the command header. Set different IDs in each command to match responses with commands.
0	[23]	Reserved.

continued...

Header	Bit	Description
Length	[22:12]	Number of words of arguments following the header. The IP responds with an error if a wrong number of words of arguments is entered for a given command.
Reserved	[11]	Reserved. Must be set to 0.
Command Code/Error Code	[10:0]	Command Code specifies the command. The Error Code indicates whether the command succeeded or failed. In the command header, these bits represent command code. In the response header, these bits represent error code.

5.3.1. Operation Commands

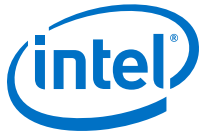
Table 35. Command List and Description

Command	Code (Hex)	Command Length (10)	Response Length (10)	Description
RSU_IMAGE_UPDATE	5C	2	0	Triggers reconfiguration from the data source which can be either the factory or an application image. This command takes an optional 64-bit argument that specifies the reconfiguration data address in the flash. If you do not provide this argument its value is assumed to be 0. • Bit [63:32]: Reserved (write as 0). • Bit [31:0]: The start address of an application image. Returns a non-zero response if the device is already processing a configuration command.
RSU_GET_SPT	5A	0	4	RSU_GET_SPT retrieves the quad SPI flash location for the two sub-partition tables that the RSU uses: SPT0 and SPT1. The 4-word response contains the following information:

Offset	Name	Description
0	SPT0[63:32]	SPT0 address in quad SPI flash.
1	SPT0[31:0]	
2	SPT1[63:32]	SPT0 address in quad SPI flash.
3	SPT1[31:0]	

continued...

(10) This number does not include the command and response header.



Command	Code (Hex)	Command Length [10]	Response Length [10]	Description		
CONFIG_STATUS	4	0	6	Reports the status of the last reconfiguration. You can use this command to check the configuration status during and after configuration. The response contains the following information:		
				Word	**Summary**	**Description**
				0	State	Describes the most recent configuration related error. Returns 0 when there are no configuration errors. The error field hCONFIG_STATUSas 2 fields: • Upper 16 bits: Major error code. • Lower 16 bits: Minor error code. Refer to the Table 36 on page 148 and Table 37 on page 149 for more information.
				1	Version	The version of the RSU data structure.
				2	Pin status	• Bit [31]: Current nSTATUS output value (active low) • Bit [30]: Detected nCONFIG input value (active low) • Bit [29:3]: Reserved • Bit [2:0]: The MSEL value at power up
				3	Soft function status	Contains the value of each of the soft functions, even if you have not assigned the function to an SDM pin. • Bit [31:6]: Reserved • Bit [5]: HPS_WARMRESET • Bit [4]: HPS_COLDRESET • Bit [3]: SEU_ERROR • Bit [2]: CVP_DONE • Bit [1]: INIT_DONE • Bit [0]: CONF_DONE
				4	Error location	Contains the error location. Returns 0 if there are no errors.
				5	Error details	Contains the error details. Returns 0 if there are no errors.

continued...

(10) This number does not include the command and response header.

<table>
<tr><th>Command</th><th>Code (Hex)</th><th>Command Length (10)</th><th>Response Length (10)</th><th>Description</th></tr>
<tr><td>RSU_STATUS</td><td>5B</td><td>0</td><td>9</td><td>Reports the current remote system upgrade status. You can use this command to check the configuration status during configuration and after it has completed. This command returns the following responses:
<table>
<tr><th>Word</th><th>Summary</th><th>Description</th></tr>
<tr><td>0-1</td><td>Current image</td><td>Flash offset of the currently running application image.</td></tr>
<tr><td>2-3</td><td>Failing image</td><td>Flash offset of the highest priority failing application image. If multiple images are available in flash memory, stores the value of the first image that failed. A value of all 0s indicates no failing images. If there are no failing images, the remainder of the remaining words of the status information do not store valid information.
Note: A rising edge on nCONFIG to reconfigure from ASx4, does not clear this field. Information about failing image only updates when the Mailbox Client receives a new RSU_IMAGE_UPDATE command and successfully configures from the update image.</td></tr>
<tr><td>4</td><td>State</td><td>Failure code of the failing image.
The error field has two parts:
• Upper 16 bits: Major error code.
• Lower 16 bits: Minor error code.
Returns 0 for no failures. Refer to the Table 36 on page 148 and Table 37 on page 149 for more information.</td></tr>
<tr><td>5</td><td>Error Reporting and Version</td><td>Starting with version 19.4 of the Intel Quartus Prime Pro Edition software, contains the following fields:</td></tr>
</table>
continued...</td></tr>
</table>

(10) This number does not include the command and response header.

Command	Code (Hex)	Command Length [10]	Response Length [10]	Description		
						• Bits[31:28]: Specifies the currently used decision firmware index. • Bits[27:16]: Source of the error. Available in version 19.3 of the Intel Quartus Prime Pro Edition software or later. The following encodings are valid: — 0x0000: No error — 0x0ACF: Application firmware reported an error — 0x0DCF: Decision firmware reported an error • Bits[15:8]: Application firmware RSU interface version. RSU interface version. The following encodings are valid: — 0x0000: Pre 19.3 release — 0x0001: 19.3 release • Bits[7:0]: Decision firmware RSU interface version. The following encodings are valid: — 0x0000: Pre 19.3 release — 0x0001: 19.3 release — 0x0002: 19.4 release
				6	Error location	Stores the error location of the failing image. Returns 0 for no errors.
				7	Error details	Stores the error details for the failing image. Returns 0 if there are no errors.
				8	Current image retry counter	Count of the number of retries that have been attempted for the current image. The counter is 0 initially. The counter is set to 1 after the first retry, then 2 after a second retry. Specify the maximum number of retries in your Intel Quartus Prime Settings File (`.qsf`). The command is: `set_global_assignment -name RSU_MAX_RETRY_COUNT 3`. Valid values for the `MAX_RETRY` counter are 1-3. The actual number of available retries is `MAX_RETRY -1` This field was added in version 19.3 of the Intel Quartus Prime Pro Edition Software.

continued...

(10) This number does not include the command and response header.

Command	Code (Hex)	Command Length [10]	Response Length [10]	Description
QSPI_OPEN	32	0	1	Requests exclusive access to the quad SPI. The SDM accepts the request if the quad SPI is not in use and the SDM is not configuring the device. Returns OK if the SDM grants access. Returns the ALT_SDM_MBOX_RESP_DEVICE_BUSY when the quad SPI flash is busy. *Note:* The SDM grants exclusive access to the client using this mailbox. Other clients cannot access the quad SPI until the active client relinquishes access using the QSPI_CLOSE command.
QSPI_CLOSE	33	0	1	Closes the exclusive access to the quad SPI interface.
QSPI_SET_CS	34	1	1	Specifies one of the attached quad SPI devices via the chip select lines. Takes a one-word argument as described below: • Bits[31:28]: Flash device to select. The value 4'b0000 selects the flash that corresponds to nCSO[0]. nCSO[0] is the only signal that the FPGA can use to access the quad SPI flash device. The HPS can use nCSO[3:1] to access HPS data. • Bits[27:0]: Reserved (write as 0). The HPS can use nCSO[3:1] to access 3 additional quad SPI devices. This command is optional for the AS x4 configuration scheme. Is required for all other configuration schemes. Access to the QSPI flash memory devices using SDM_IO pins is only available for the AS x4 configuration scheme, JTAG configuration, and a design compiled for ASx4 configuration. For the Avalon ST configuration scheme, you must connect QSPI flash memories to GPIO pins.
QSPI_READ	3A	2	N	Reads the attached quad SPI device. The maximum read size is 4 kilobytes (KB). Takes two arguments: • The quad SPI flash address (one word). The address must be word aligned. The device returns the 0x1 error code for non-aligned addresses. • Number of words to read (one word). When successful returns OK followed by the read data from the quad SPI device. A failure response returns an error code. For a partially successful read, QSPI_READ may erroneously return the OK status. *Note:* You cannot run the QSPI_READ command while device configuration is in progress.
QSPI_WRITE	39	2+N	0	Writes data to the quad SPI device. Takes three arguments: • The flash address offset (one word). The write address must be word aligned. The device returns error code 0x3FF for non-aligned addresses. • The number of words to write (one word). • The data to be written (one or more words). A successful write returns the OK response code.

continued...

(10) This number does not include the command and response header.

Command	Code (Hex)	Command Length [10]	Response Length [10]	Description
				To prepare memory for writes, Intel recommends using the QSPI_ERASE command before issuing this command. *Note:* You cannot run the QSPI_WRITE command while device configuration is in progress.
QSPI_ERASE	38	2	0	Erases a sector of the quad SPI device. Takes two arguments: • The flash address offset to start the erase (one word). The address must be the start address of a sector within the flash memory; consequently, the address must be 64 KB aligned. Returns an error for non-64 KB aligned addresses. • The number of words to erase specified in multiples of 0x4000 words. A successful erase returns the OK response code.
QSPI_READ_DEVICE_REG	35	2	N	Reads registers from the quad SPI device. The maximum read is 8 bytes. Takes two arguments. • The opcode for the read command. • The number of bytes to read. A successful read returns the OK response code followed by the data read from the device. Pads data that is not a multiple of 4 bytes to the next word boundary.

continued...

(10) This number does not include the command and response header.

Command	Code (Hex)	Command Length (10)	Response Length (10)	Description
QSPI_WRITE_DEVICE_REG	36	2+N	0	Writes to registers of the quad SPI. The maximum write is 8 bytes. Takes three arguments: • The opcode for the write command. • The number of bytes to write. • The data to write. To perform a sector erase or sub-sector erase, you must specify the serial flash address in most significant byte (MSB) to least significant byte (LSB) order as the following example illustrates. To erase a sector of a Micron 2 gigabit (Gb) flash at address 0x04FF0000 using the QSPI_WRITE_DEVICE_REG command, write the flash address in MSB to LSB order as shown here: Header: 0x00003036 Opcode: 0x000000DC Number of bytes to write: 0x00000004 Flash address: 0x0000FF04 A successful write returns the OK response code. This command pads data that is not a multiple of 4 bytes to the next word boundary.
QSPI_SEND_DEVICE_OP	37	1	0	Sends a command opcode to the quad SPI. Takes one argument: • The opcode to send the quad SPI device. A successful command returns the OK response code.
RSU_NOTIFY	5D	1	0	Clears all error information in the RSU_STATUS response and resets the retry counter. The one-word argument has the following fields: • 0x00050000: Clear current reset retry counter. Resetting the current retry counter sets the counter back to zero, as if the current image was successfully loaded for the first time. • 0x00060000: Clear error status information. • All other values are reserved. This command is not available before version 19.3 of the Intel Quartus Prime Pro Edition Software.

Table 36. CONFIG_STATUS and RSU_STATUS Major Error Code Descriptions

Major Error Code	Error Type	Description
0xF001	BITSTREAM_ERROR	Potential unsigned bitstream used. Ensure the bitstream is signed with the correct key.
0xF002	HARDWARE_ACCESS_FAILURE	Failure to communicate to PMBus-compliant voltage regulator. Check your power management and smart voltage identification (SmartVID) parameter settings and PMBus interface connections.

continued...

(10) This number does not include the command and response header.

Major Error Code	Error Type	Description
0xF003	BITSTREAM_CORRUPTION	Bitstream is corrupt. Ensure the bit stream in configuration device or flash is not corrupt.
0xF004	INTERNAL_ERROR	This major code may indicate the following error events: • An error in the SDM Crypto IP task. • An RSU operation error. Refer to Table 37 on page 149 for more information.
0xF005	DEVICE_ERROR	Indicates an SDM internal device error. The following errors are possible: • A device cleaning failure • An HPS configuration failure Contact your local Field Applications Engineer (FAE). Alternatively, submit a Service Request on the My Intel support page to get the support on capturing the error log for further debug.
0xF006	HPS_WATCHDOG_TIMEOUT	HPS watchdog timeout failure. Ensure that your design resets the watchdog timer correctly.
0xF007	INTERNAL_UNKNOWN_ERROR	Indicates an internal device error due to an unknown task. You can contact your local Field Applications Engineer (FAE). Alternatively, submit a Service Request on the My Intel support page to get the support on capturing the error log for further debug.

Table 37. CONFIG_STATUS and RSU_STATUS Minor Error Code Descriptions

Minor Error Code	Error Type	Description
0xD001	RSU_CMF_AUTH_ERR	Authentication failure for the firmware.
0xD002	RSU_USER_AUTH_ERR	Authentication failure for the design.
0xD003	RSU_CMF_DESC_SHA_MISMATCH	The SHA does not match for the firmware descriptor.
0xD004	RSU_POINTERS_NOT_FOUND_ERR	Unable to read data from boot ROM on first boot after the device exits power-on reset (POR).
0xD005	RSU_QSPI_REQ_CHANGE	Unable to configure the quad SPI flash during RSU initialization.
0xD006	RSU_FACTORY_IMAGE_FAILED	Failed to load any image, including the factory image.
0xD007	RSU_CMF_TYPE_ERR	The firmware version does not match the version that was previously loaded.

5.3.2. Error Code Responses

Table 38. Error Codes

Value (Hex)	Error Code Response	Description
0	OK	Indicates that the command completed successfully. A command may erroneously return the OK status if a command, such as QSPI_READ is partially successful.
1	INVALID_COMMAND	Indicates that the command is incorrectly formatted.
2	UNKNOWN_BR	Indicates that the command code is not understood.
3	UNKNOWN	Indicates that the currently loaded firmware cannot decode the command code.
4	INVALID_COMMAND_PARAMETERS	The length or indirect setting in header is not valid. Or the command data is invalid.
5	COMMAND_INVALID_ON_SOURCE	Command is from a source for which it is not enabled.
6	CLIENT_ID_NO_MATCH	Indicates that the Client ID requesting quad SPI or SD MMC access does not have exclusive access.
7	INVALID_ADDRESS	The address is invalid. This error indicates one of the following conditions: • An unaligned address • An address range problem • A read permission problem
8	TIMEOUT	The command timed out.
9	HW_NOT_READY	The hardware is not ready. Can indicate either an initialization or configuration problem.
100	NOT_CONFIGURED	Indicates that the device is not configured.
1FF	ALT_SDM_MBOX_RESP_DEVICE_BUSY	Indicates that the device is busy.
2FF	ALT_SDM_MBOX_RESP_NO_VALID_RESP_AVAILABLE	Indicates that there is no valid response available.
3FF	ALT_SDM_MBOX_RESP_ERROR	General Error.

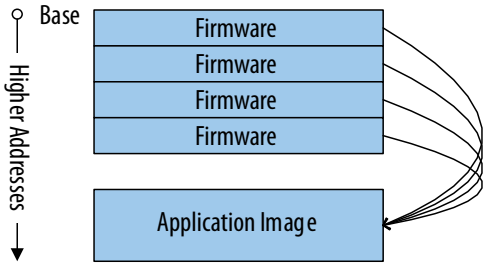
5.4. Quad SPI Flash Layout

5.4.1. High Level Flash Layout

5.4.1.1. Standard (non-RSU) Flash Layout

In the standard (non-RSU) case, the flash contains four firmware images and the application image. To guard against possible corruption, there are four redundant copies of the firmware. The firmware contains a pointer to the location of the highest priority application image in flash. Typically the application image is immediately after the four firmware copies, but the Intel Quartus Prime Pro Edition tools do not require this location.

Figure 67. Flash Layout - Non-RSU

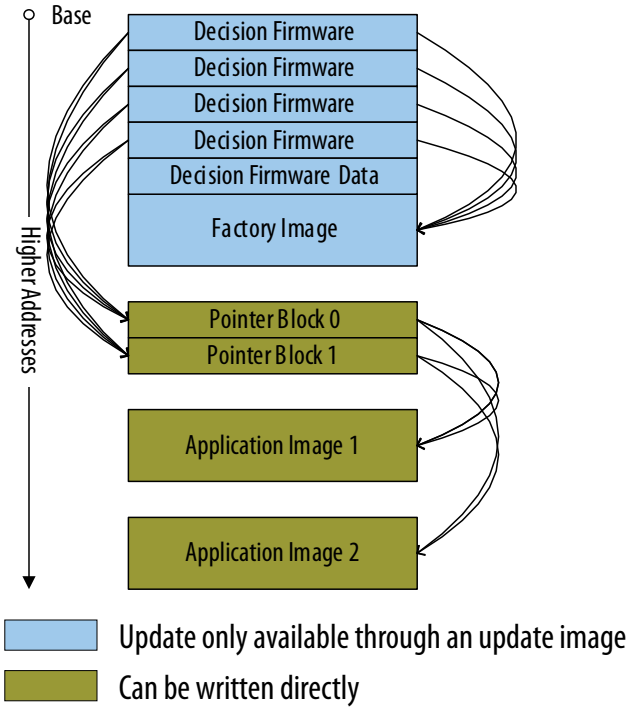


5.4.1.2. RSU Flash Layout – SDM Perspective

In the RSU case, decision firmware replaces the standard firmware. The decision firmware copies have pointers to the following structures in flash:

- Decision data
- One factory image
- Two Pointer Blocks (CPBs)

Figure 68. RSU Flash Layout - SDM Perspective



The decision firmware data stores basic settings, including the following:

- The clock and pins that connect to quad SPI flash memory
- The **Direct to Factory Image** pin that forces the SDM to load the factory image

  Note: You can set this pin on the following menu in the factory image project: **Assignments ➤ Device ➤ Device and Pin Options ➤ Configuration ➤ Configuration Pin Options**

The pointer blocks contain a list of application images to try until one of them is successful. If none are successful, the SDM loads the factory image. To ensure reliability, the pointer block includes a main and a backup copy in case an update operation fails.

Both the factory image and the application images start with firmware. First, the decision firmware loads the firmware. Then, that firmware loads the rest of the image. These implementation details are not shown in the figure above. For more information, refer to the *Application Image Layout* section.

5.4.1.3. RSU Flash Layout – Your Perspective

The sub-partition table (SPT) manages the quad SPI flash.

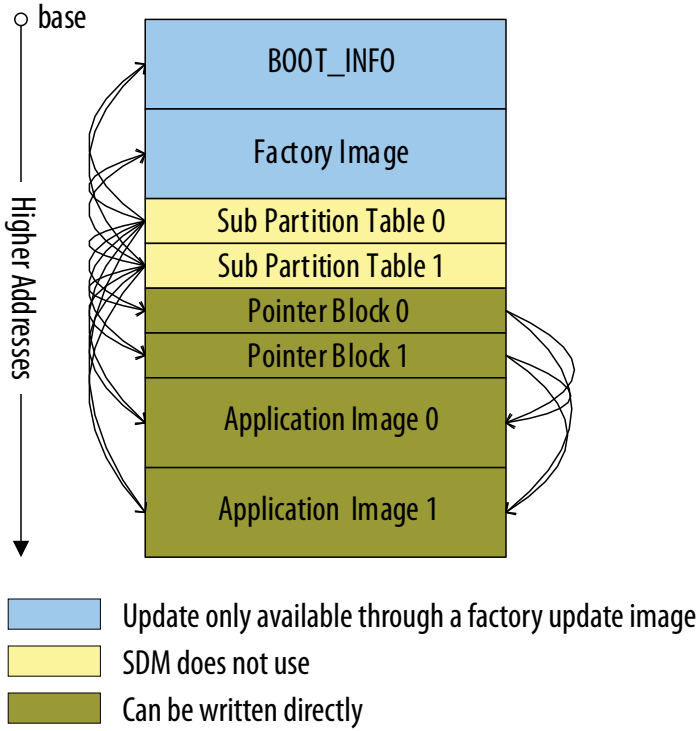
The Intel Quartus Prime Programming File Generator creates the SPT when creating the initial manufacturing image. To ensure reliable operation, the Programming File Generator creates two copies of the sub-partition table and the configuration pointer block, SPT0 and SPT1 and CPB0 and CPB1

The initial RSU image stored in flash typically contains the following partitions:

Table 39. Typical Sub-Partitions of the Initial RSU Image

Sub-partition Name	Contents
BOOT_INFO	Decision firmware and decision firmware data
FACTORY_IMAGE	Factory Image
SPT0	Sub-partition table 0
SPT1	Sub-partition table 1
CPB0	Pointer block 0
CPB1	Pointer block 1
P1 (you enter)	Production image 1
P2 (you enter)	Production image 2

Figure 69. RSU Flash Layout - Your Perspective



To summarize, your view of flash memory is different from SDM view in two ways:

- You do not need to know the addresses of the decision firmware, decision firmware data, and factory image.
- You have access to the sub-partition tables. The sub-partition tables provide access to the data structures required for remote system update.

5.4.2. Detailed Quad SPI Flash Layout

5.4.2.1. RSU Sub-Partitions Layout

The *Flash Sub-Partitions Layout* table shows the layout of RSU flash images.

Table 40. Flash Sub-Partitions Layout

Flash Offset	Size (in bytes)	Contents	Sub-Partition Name
0 K	512 K	Decision firmware	BOOT_INFO
512 K	512 K	Decision firmware	
1 MB	512 K	Decision firmware	
1.5 MB	512 K	Decision firmware	
2 MB	8 K + 24 pad	Decision firmware data	
2M+32 K	32 K	Reserved for SDM	
2M+64 K	varies	Factory image	FACTORY_IMAGE
Next	4 K + 28 K pad	Sub-partition table (copy 0)	SPT0
Next	4 K + 28 K pad	Sub-partition table (copy 1)	SPT1
Next	4 K + 28 K pad	Pointer block (copy 0)	CPB0
Next	4 K + 28 K pad	Pointer block (copy 1)	CPB1
Next	varies	Application image 1	You assign
Next	varies	Application image 2	You assign

The Intel Quartus Prime Programming File Generator allows you to create many user partitions. These partitions can contain application images and other items such as the Second Stage Boot Loader (SSBL), Linux* kernel, or Linux root file system.

When you create the initial flash image, you can create up to seven partitions for application images. There are no limitations on creating empty partitions.

5.4.2.2. Sub-Partition Table Layout

The following table shows the structure of the sub-partition table. The Intel Quartus Prime software supports up to 126 partitions. Each sub-partition descriptor is 32 bytes.

Table 41. Sub-partition Table Layout

Offset	Size (in bytes)	Description
0x000	4	Magic number 0x57713427
0x004	4	Version number (0 for this document)
0x008	4	Number of entries
0x00C	20	Reserved
0x020	32	Sub-partition Descriptor 1
0x040	32	Sub-partition Descriptor 2
0xFE0	32	Sub-partition Descriptor 126

Each 32-byte sub-partition descriptor contains the following information:

Table 42. Sub-partition Descriptor Layout

Offset	Size	Description
0x00	16	Sub-partition name, including a null string terminator
0x10	8	Sub-partition start offset
0x18	4	Sub-partition length
0x1C	4	Sub-partition flags

Two flags are currently defined:

- bit 0: system/reserved
- bit 1: read only

The Intel Quartus Programming File Generator sets these flags as follows at image creation time, then they are not changed afterward:

Partition	System	Read Only
BOOT_INFO	1	1
FACTORY_IMAGE	1	1

continued...

Partition	System	Read Only
SPT0	1	0
SPT1	1	0
CPB0	1	0
CPB1	1	0
P1	0	0
P2	0	0

5.4.2.3. Configuration Pointer Block Layout

The configuration pointer block contains a list of application image addresses. The SDM tries the images in sequence until one of them is successful or all fail. The structure contains the following information:

Table 43. Pointer Block Layout

Offset	Size (in bytes)	Description
0x00	4	Magic number 0x57789609
0x04	4	Size of pointer block header (0x18 for this document)
0x08	4	Size of pointer block (4096 for this document)
0x0C	4	Reserved
0x10	4	Offset to image pointers (IPTAB)
0x14	4	Number of image pointer slots (NSLOTS)
0x18	—	Reserved
IPTAB	8	First (lowest priority) image pointer slot
	8	Second (2nd lowest priority) image pointer slot
	8	…
	8	Last (highest priority) image pointer

The configuration pointer block can contain up to 508 application image pointers. The actual number is listed as NSL0TS. A typical configuration pointer block update procedure consists of adding a new pointer and potentially clearing an older pointer. Typically, the pointer block update uses one additional entry. Consequently, you can make 508 remote system updates before the pointer block must be erased. The erase procedure is called *pointer block compression*. This procedure is safe. There are two copies of pointer block. The copies are in different flash erase sectors. While one copy is being updated the other copy is still valid.

5.4.2.4. Modifying the List of Application Images

The SDM uses the configuration pointer block to determine priority of application images.

The pointer block operates taking into account the following characteristics of quad SPI flash memory:

- On a sector erase, all the sector flash bits become 1’s.
- A program operation can only turn 1’s into 0’s.

The pointer block contains an array of values which have the following meaning:

- All 1’s – the entry is unused. The client can write a pointer to this entry. This is the state after a quad SPI erase operation occurs on the pointer block.
- All 0’s – the entry has been previously used and then canceled.
- A combination of 1's and 0's – a valid pointer to an application image.

When the configuration pointer block is erased, all entries are marked as unused. To add an application image to the list, the client finds the first unused location and writes the application image address to this location. To remove an application image from the list, the client finds the application image address in the pointer block list and writes this address to 0s.

If the configuration pointer block runs out of space for new application images, the client compresses the pointer block by completing the following actions:

- Read all the valid entries from the configuration
- Erasing the pointer block
- Adding all previously valid entries
- Adding the new image

When using HPS to manage RSU, both the U-Boot and LIBRSU clients implement the block compression. For designs that drive RSU from FPGA logic, you can implement pointer block compression many different ways, including Nios II code, a scripting language, or a state machine.

Pointer block compression does not occur frequently because the pointer block has up to 508 available entries.

There are two configuration pointer blocks: a primary (CPB0) and a backup (CPB1). Two blocks enable the list of application images to be protected if a power failure occurs just after erasing one of them. When a CPB is erased and re-created, the header is written last. The CPB header is checked prior to use to prevent accidental use if a power failure occurred. For more information, refer to the *Configuration Pointer Block Layout* topic. When compressing, the client compresses (erases and rewrites) the primary CPB completely. Once the primary CPB is valid, it is safe to modify the secondary CPB. When rewriting, the magic number at the start of a CPB is the last word written in the CPB. (After this number is written only image pointer slot values can be changed.)

When the client writes the application image to flash, it ensures that the pointers within the main image pointer of its first signature block are updated to point to the correct locations in flash. When using HPS to manage RSU, both the U-Boot and LIBRSU clients implement the required pointer updates. For more information, refer to the *Application Image Layout* topic.

5.4.2.5. Application Image Layout

The application image comprises SDM firmware and the configuration data. The configuration data includes up to four sections. The SDM firmware contains pointers to those sections. The table below shows the location of the number of sections and the section pointers in a application image.

By default the first 16 bytes of the application image starting at address offset 0x1FC0 are 0. However you can use these 16 bytes to store a Version ID to identify your application image. Providing this Version ID allows you to verify the images stored in flash memory at a later time.

Table 44. Application Image Section Pointers

Offset	Size (in bytes)	Description
0x1F00	4	Number of sections
	…	
0x1F08	8	Address of 1st section
0x1F10	8	Address of 2nd section
0x1F18	8	Address of 3rd section
0x1F20	8	Address of 4th section
	…	
0x1FFC	4	CRC32 of 0x1000 to 0x1FFB

The section pointers must match the actual location of the FPGA image in flash. Two options are available to meet this requirement:

- You can generate the application image to match the actual location in quad SPI flash memory, Because different systems may have a different set of updates. This option may not be practical.
- You can generate the application image as if it is located at address zero, then update the pointers to match the actual location.

Note: When using HPS to manage RSU, both U-Boot and LIBRSU clients implement the above procedure to relocate application images targeting address zero in the actual destination slot address.

Here is the procedure to update the pointers from an application image created for `INITIAL_ADDRESS` to `NEW_ADDRESS`:

1. Create the application image, targeting the `INITIAL_ADDRESS`.
2. Read the 32-bit value from offset 0xF100 of the application image to determine the number of sections.
3. For `<s>= 1 to number_of_sections`:
   a. `section_pointer` = read the 64-bit section pointer from 0xF100 + (s * 8)
   b. Subtract `INITIAL_ADDRESS` from `section_pointer`
   c. Add `NEW_ADDRESS` to `section_pointer`
   d. Store updated `section_pointer`
4. Recompute the CRC32 for addresses 0x1000 to 0x1FFB. Store the new value at offset 0x1FFC. The CRC32 value must be computed on a copy of the data using the following procedure:
   a. Swap the bits of each byte so that the bits occur in reverse order and compute the CRC.
   b. Swap the bytes of the computed CRC32 value to appear in reverse order.
   c. Swap the bits in each byte of the CRC32 value.
   d. Write the CRC32 value to flash.

Note: The factory update image has a different format. Refer to the *Generating a Factory Update Image* topic for the correct procedure to generate the factory update image.

5.5. Generating Remote System Update Image Files Using the Programming File Generator

Use the Intel Quartus Prime Programming File Generator tool to generate the Intel Agilex remote system update flash programming files.

5.5.1. Generating the Initial RSU Image

Follow these steps to generate the initial RSU image:

1. On the **File** menu, click **Programming File Generator**.
2. Select Intel Agilex from the **Device family** drop-down list.
3. Select the configuration scheme from the **Configuration scheme** drop-down list. The current Intel Quartus Prime only supports remote system update feature in **Active Serial x4**.
4. On the **Output Files** tab, assign the output directory and file name.
5. Select the output file type.

   Select the following file types for AS x4 configuration mode:

   - JTAG Indirect Configuration File (`.jic`)/Programmer Object File (`.pof`)
   - Memory Map File (`.map`)
   - Raw Programming File (`.rpd`)
6. On the **Input Files** tab, click **Add Bitstream**, select the factory and application image `.sof` files and click Open.
7. On the **Configuration Device** tab, click **Add Device**, select your flash memory and click **OK**. The Programming File Generator tool automatically populates the flash partitions.
8. Select the FACTORY_IMAGE partition and click **Edit**.
9. In the **Edit Partition** dialog box, select your factory image `.sof` file in the Input file drop-down list and click **OK**.

   Note: You must assign Page 0 to Factory Image. Intel recommends that you let the Intel Quartus Prime software assign the Start address of the FACTORY_IMAGE automatically by retaining the default value for **Address Mode** which is **Auto**. From the **Address Mode** drop down list, select **Block** to set an **End address** value for the FACTORY_IMAGE. The **Programming File Generator** reserves and assigns the start and end flash addresses to store BOOT_INFO, SPT0, SPT1, CPB0, and CPB1.
10. Select the flash memory and click **Add Partition**.

11. In the **Add Partition** dialog box, select for application image `.sof` file from the **Input file** drop-down list, assign the page number.
12. Repeat this step for additional application images and click **OK**. You can add up to seven partitions for seven application images. The **page 1** application image is the highest priority, and the **page 7** image is the lowest priority.
13. For `.jic` files,

    Click **Select** at the Flash loader, select your device family and device name, and click **OK**.
14. Click **Generate** to generate the remote system update programming files. After generating the programming file, you can proceed to program the flash memory.

    Note: The generated `.jic` file contains only the initial flash data. If a remote host updates the initial flash image and then the application performs a verify operation, the verify operation fails. Verification fails because the verify operation compares the updated image to the initial flash data. If you want to verify the updated flash image, you read back the updated image from flash and compare it to the expected `.rpd` file.

    You can use the programmer to examine the flash content and compare it to the new flash image `.rpd`.

    Note: If you plan to update the factory image, Intel recommends reserving an additional 64 KB space for possible expansion of the factory image. Complete the following steps to reserve extra space for updates to the factory image:

    a. Identify the new end address by adding 64 KB to the existing `END ADDRESS` of the `FACTORY_IMAGE`. The end address is available in the `.map` file. For example, if the current end address is `0x00423FF`, the new end address is `0x00523FF` .

    b. Repeat the steps to regenerate the new `.jic` file. On the **Configuration Device** tab, select the `FACTORY_IMAGE` partition and click **Edit**. In the **Edit Partition** dialog box, under the **Address Mode** drop down list, select **Block** to set the new **End address** value for the `FACTORY_IMAGE`.

    c. You can optionally **Click File ➤ Save As ..** to save the configuration parameters as a file with the `.pfg` extension. The `.pfg` file contains your settings for the Programming File Generator. After you save the `.pfg`, you can the use this file to regenerate the programming file by running the following command:

```
quartus_pfg -c <configuration_file>.pfg
```

The `.pfg` file is actually an XML file which you can edit to replace absolute file paths with relative file paths. You cannot edit the`.pfg` file for other reasons. You can open and edit the `.pfg` in the Programming File Generator.

5.5.2. Generating an Application Image

You can generate the RSU image from the command line directly, by running the `quartus_pfg` with the following arguments:

```
quartus_pfg -c fpga.sof application.rpd -o mode=ASX4 -o start_address=<address> -o bitswap=ON
```

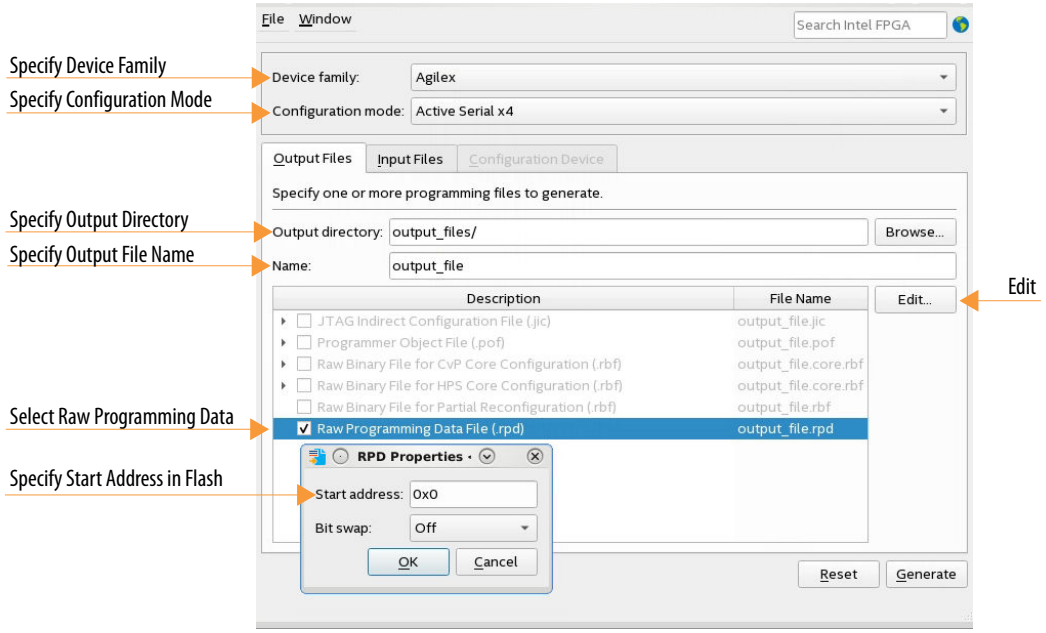
Alternatively, you can use the Intel Quartus Prime Pro Edition **Programming File Generator** to generate the `.rpd` image by completing the following procedure:

1. On the **File** menu, click **Programming File Generator**.
2. Select Intel Agilex from the **Device family** drop-down list.
3. Select the configuration mode from the **Configuration mode** drop-down list. The current Intel Quartus Prime only supports remote system update feature in **Active Serial x4**.
4. On the **Output Files** tab, assign the output directory and file name.
5. Select the output file type.

   Select the following file types for AS x4 configuration mode:

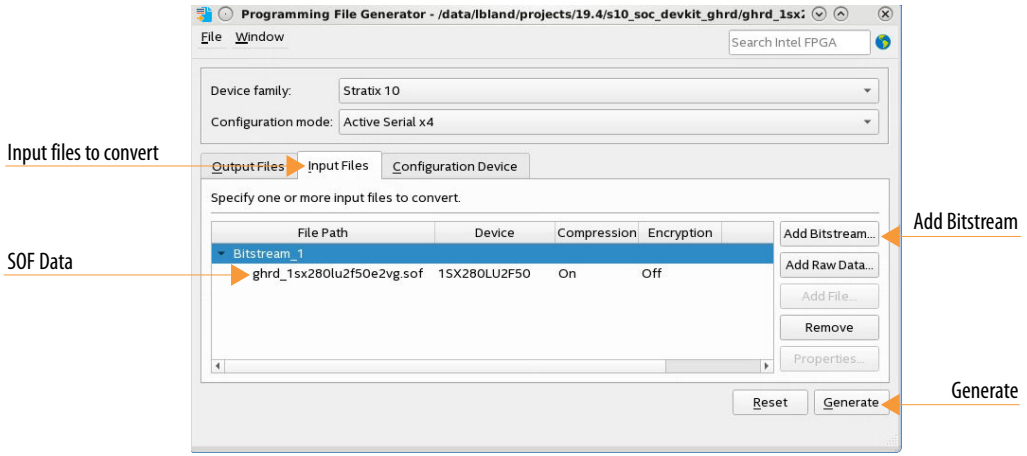
   - Raw Programming File (`.rpd`)
6. Click the **Edit...** button and assign the **Start address** for the image in flash memory. This **Start address** must match the starting address of the target partition in flash memory.

Figure 70. Specifying Parameters for an Application .rpd Stored in Flash Memory



7. By default, the `.rpd` file type is little-endian, if you are using a third-party programmer that does not support the little-endian format. Set the **Bit swap** to **On** to generate the `.rpd` file in big endian format.
8. On the **Input Files** tab, click **Add Bitstream**. Change the **Files of type** to SRAM Object File (`*.sof`). Then, select application image `.sof` file and click **Open**.

Figure 71. Specify the .sof File



9. Click **Generate** to generate the remote system update programming files. You can now program the flash memory. You can save the configuration in a `.pfg` file for later use.

5.5.3. Generating a Factory Update Image

You can generate the factory update image from the command line directly, by running the `quartus_pfg` with the following arguments:

```
quartus_pfg -c fpga.sof factory_update.rpd -o mode=ASX4 -o start_address=<address> -o bitswap=ON -o rsu_upgrade=ON
```

Alternatively, you can use the Intel Quartus Prime Pro Edition **Programming File Generator** to generate a factory update image (`.rpd`). You can use this image to update the decision firmware, decision firmware data, and the factory image.

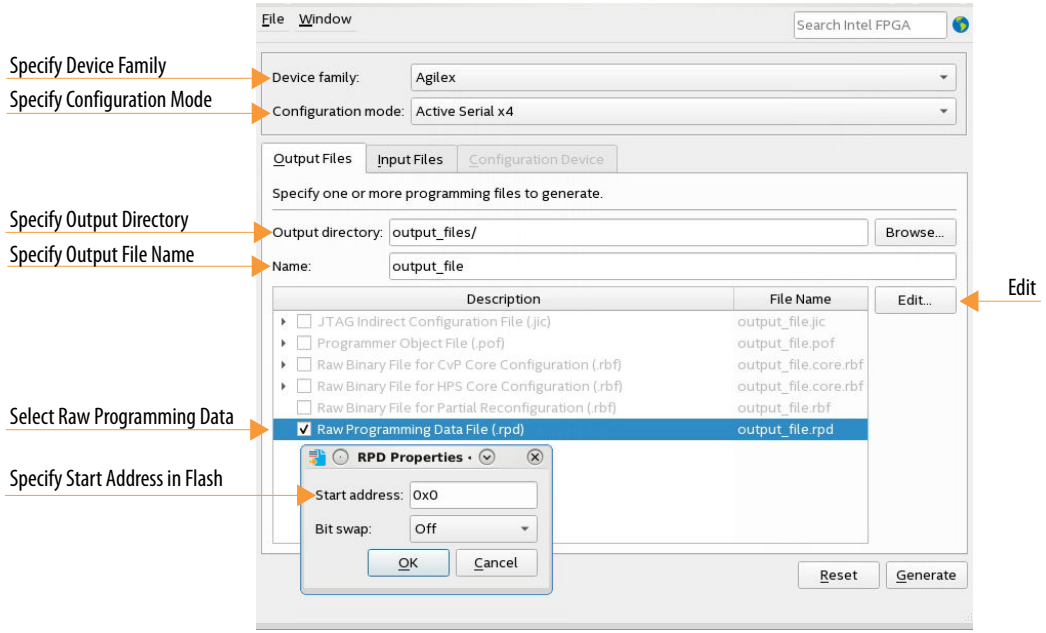
1. *Note:* The `.rpd` to program flash memory includes firmware pointer information for image addresses. You must use the **Programming File Generator** to generate the `.rpd` for flash devices.

   On the **File** menu, click **Programming File Generator**.
2. Select Intel Agilex from the **Device family** drop-down list.
3. Select the configuration mode from the **Configuration mode** drop-down list. The current Intel Quartus Prime only supports the RSU feature in the **Active Serial x4** configuration mode.
4. On the **Output Files** tab, assign the **Output directory** and **Name**.

5. Select the `.rpd` output file type.
6. Click the **Edit...** button and assign the **Start address** for the update image in flash memory. This **Start address** should be the sector boundary of unused space in flash memory.

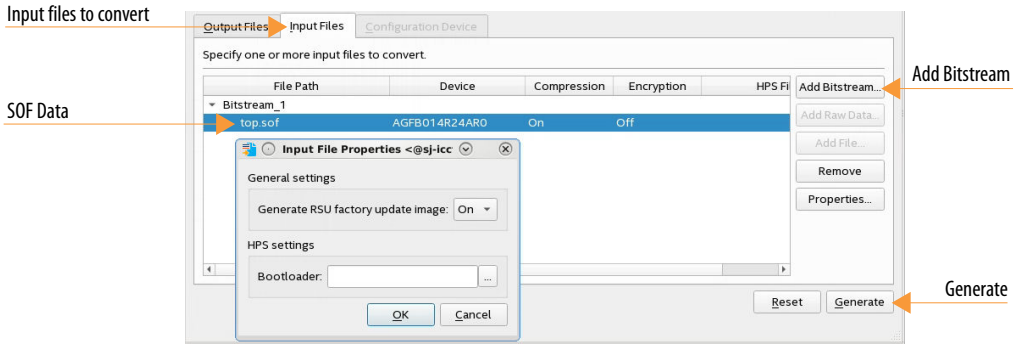
   Note: If unused space is not available, you can use an application image space other than application image 1. In this case after the update operation completes you must restore the application image by writing the associated application image (`.rpd`) to the application slot.

Figure 72. Specifying Parameters for Single .rpd Stored in Flash Memory



7. By default, the `.rpd` file type is little-endian. If you are using a third-party programmer that does not support the little-endian format, set **Bit swap** to **On** to generate the `.rpd` file in big endian format.
8. On the **Input Files** tab, click **Add Bitstream**. If necessary, change the **Files of type** to SRAM Object File (`*.sof`). Then, select factory image `.sof` file and click **Open**.

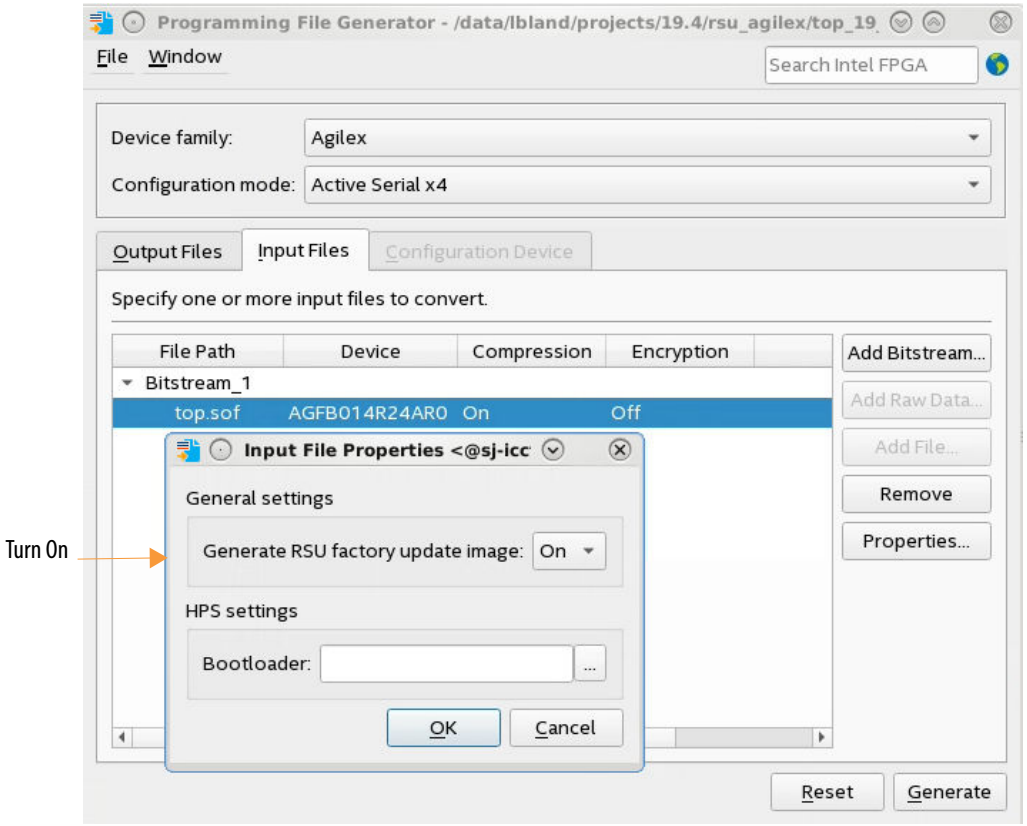
Figure 73. Specify the .sof File



9. Select the `.sof` and then click **Properties.** Turn **On Generate RSU factory update image**. The **Bootloader** parameter

   Note: You only have to specify the **Bootloader** parameter for Intel Agilex SX devices.

Figure 74. Turn On Remote System Firmware Upgrade



10. Click **Generate** to generate the RSU programming files. You can now update the Intel Agilex firmware. You can save the configuration in a `.pfg` file for later use.

5.5.4. Command Sequence To Perform Quad SPI Operations

Here is the recommended command sequence to access quad SPI flash memory or perform an RSU update operation.

Refer to Table 35 on page 142 for more information about these commands.

1. Request exclusive access to the AS x4 interface: `QSPI_OPEN`.
2. Specify a quad SPI flash chip: `QSPI_SET_CS*`. This command is optional for the AS x4 configuration scheme and mandatory for other configuration schemes.
3. Perform the desired operation or operations. The following operations are available: `QSPI_READ`, `QSPI_WRITE`, `QSPI_ERASE`, `QSPI_READ_DEVICE_REG`, `QSPI_WRITE_DEVICE_REG`, `QSPI_SEND_DEVICE_OP`, and `RSU_IMAGE_UPDATE`.
4. Close exclusive access to the AS x4 interface: `QSPI_CLOSE`.

5.6. Remote System Update from FPGA Core Example

This section presents a complete remote system update example, including the following steps:

1. Creating the initial remote system update image (`.jic`) containing the bitstreams for the factory image and one application image.
2. Programming the flash memory with the initial remote system update image that subsequently configures the device.
3. Reconfiguring the device with an application or factory image.
4. Creating a single remote system update (`.rpd`) containing the bitstreams to add an application image in user mode.
5. Adding an application image.
6. Removing an application image.

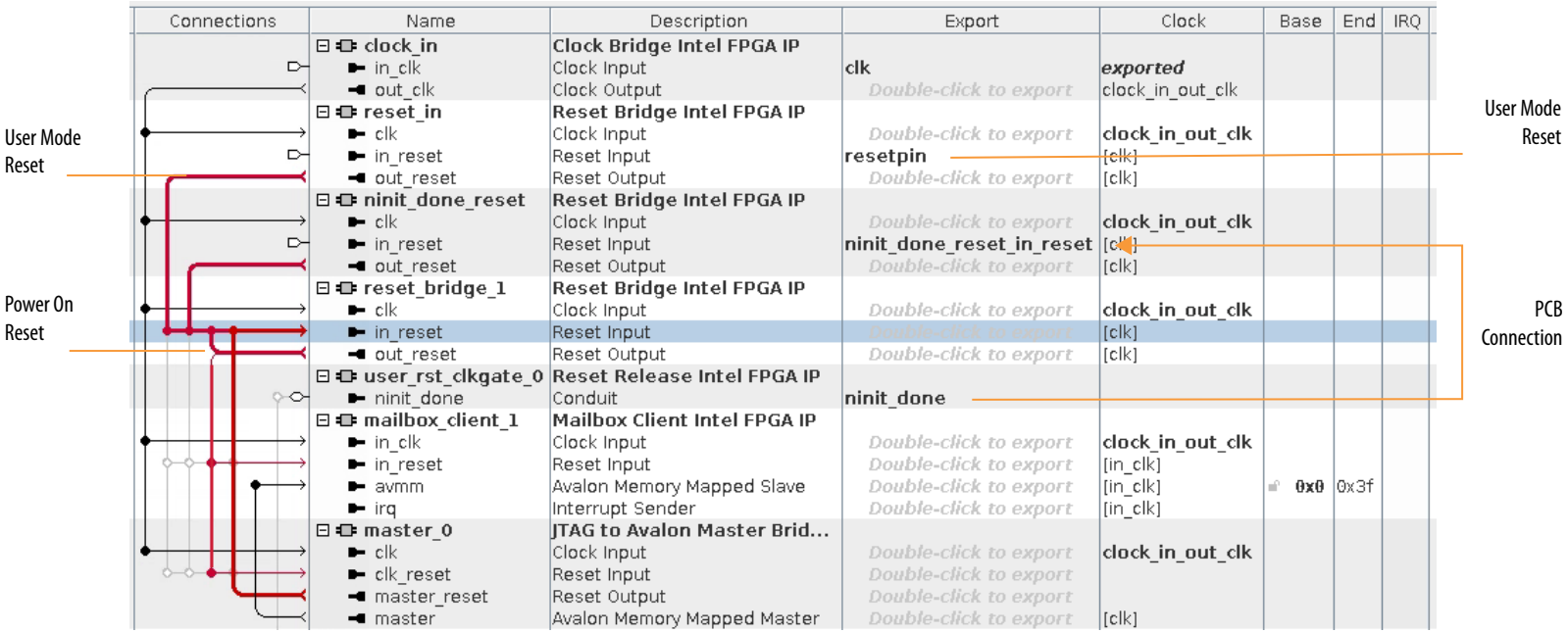
5.6.1. Prerequisites

To run this remote system update example, your system must meet the following hardware and software requirements:

- You should create and download this example to the Intel Agilex SoC Development Kit.
- Your design should include the Mailbox Client Intel FPGA IP that connects to a JTAG to Avalon Master Bridge as shown the Platform Designer system. The JTAG to Avalon Master Bridge acts as the remote system update host controller for your factory and application images.
- In addition, your design must include the Reset Release Intel FPGA IP. This component holds the design in reset until the entire FPGA fabric has entered user mode.
- The `ninit_done_reset` and `reset_bridge_1` components create a two-stage reset synchronizer to release the Mailbox Client Intel FPGA IP and JTAG to Avalon Master Bridge Intel FPGA IP from reset when the device configuration is complete and the device is in user mode.

- The `ninit_done` output signal from Reset Release IP gates this reset by connecting to the `ninit_done_reset in_reset` pin.
- The `reset_in` Reset Bridge Intel FPGA IP provides a user mode reset. In this design, the exported `resetpin` connects to application logic.

Figure 75. Required Communication and Host Components for the Remote System Update Design Example

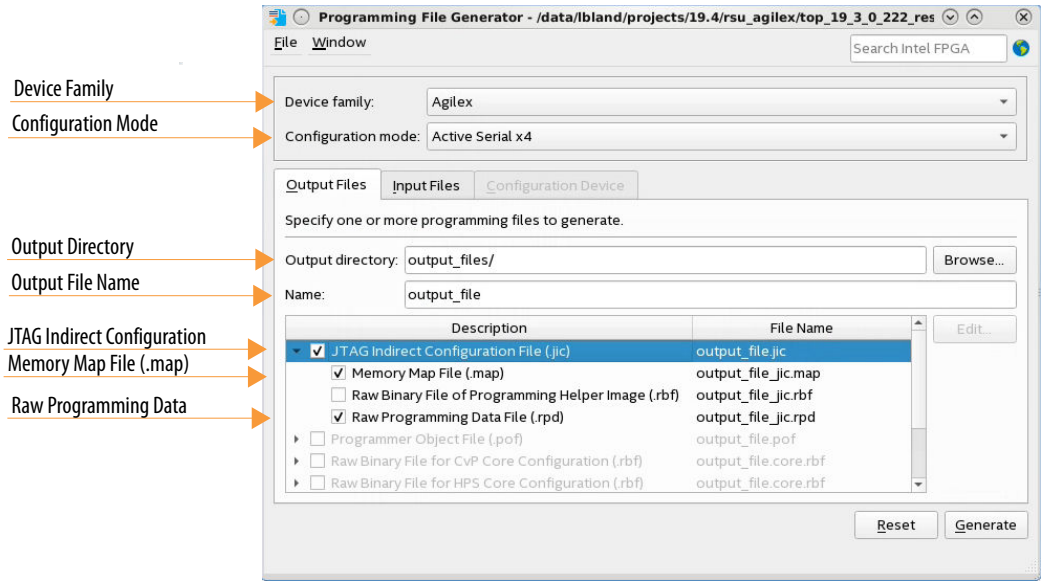


5.6.2. Creating Initial Flash Image Containing Bitstreams for Factory Image and One Application Image

1. On the **File** menu, click **Programming File Generator**.
2. Select Intel Agilex from the **Device family** drop-down list.
3. Select the configuration mode from the **Configuration mode** drop-down list. The current Intel Quartus Prime Software only supports remote system update feature in **Active Serial x4**.

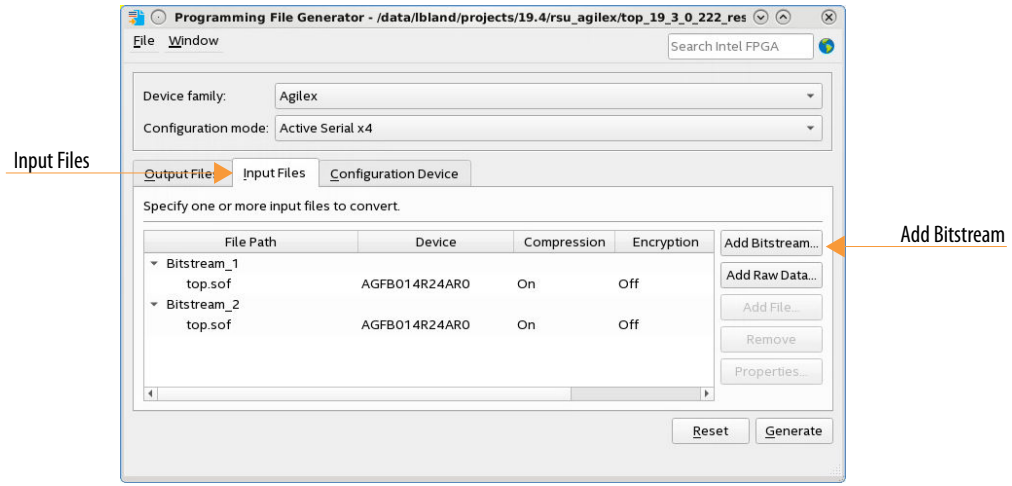
4. On the **Output Files** tab, assign the output directory and file name.
5. Select the output file type.
   Select the following file types for the Active Serial (AS) x4 configuration mode:
   - JTAG Indirect Configuration File (`.jic`)
   - Memory Map File (`.map`)
   - Raw Programming File (`.rpd`). Generating the `.rpd` file is optional.

Figure 76. Output Files Tab: Creating Initial Flash Image



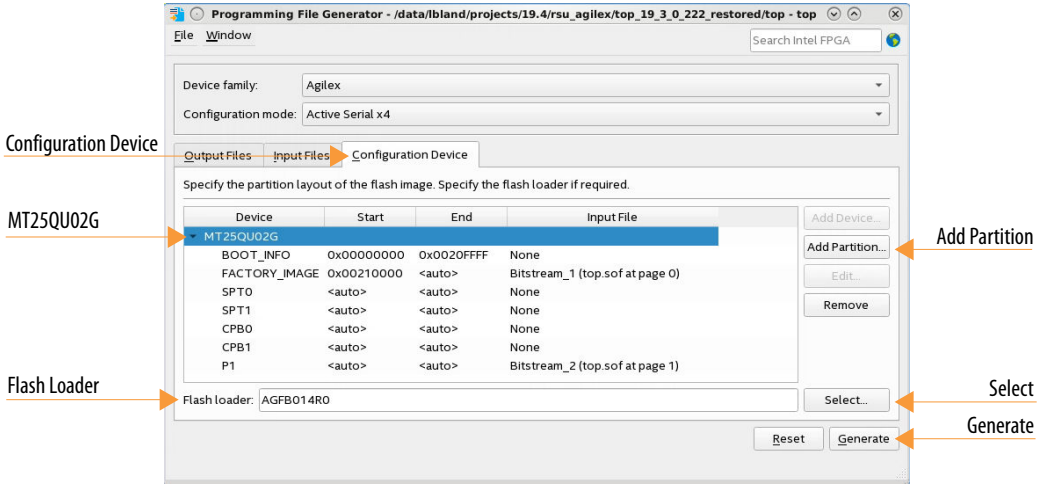
6. On the **Input Files** tab, click **Add Bitstream**, select the factory and application image `.sof` files and click **Open**.
   a. Bitstream_1 is the bitstream for factory image.
   b. Bitstream_2 is the bitstream for application image.

Figure 77. Input Files Tab: Specifying the .sof



7. On the **Configuration Device** tab, click **Add Device**, select **MT25QU02G** flash memory and click **OK**. The Programming File Generator tool automatically populates the flash partitions.
8. Select the **FACTORY_IMAGE** partition and click **Edit**.
9. On the **Edit Partition** dialog box, select **Bitstream_1** as the factory image `.sof` in the **Input file** drop-down list. Keep the default settings for the **Page** and **Address Mode**. Click **OK**.
10. Select the **MT25QU02G** flash memory and click **Add Partition**.
11. In the **Add Partition** dialog box, select **Bitstream_2** for the application image `.sof` in the **Input file** drop-down list. Assign **Page: 1**.Keep the default settings for **Address Mode**. Click **OK**.
12.
13. For **Flash loader** click **Select**. Select Intel Agilex from **Device family** list. Select **AGFA014R24A3E3VR0** for the **Device name**. Click **OK**.
14. Click **Generate** to generate the remote system update programming files. The Programming File Generator generates the following files:
    a. `Initial_RSU_Image.jic`
    b. `Initial_RSU_Image_jic.map`

Figure 78. Configuration Tab: Add Device, Partition, Flash Loader and Generate



The following example output shows the generated `.map` file. The `.map` lists the start addresses of the factory image, CPB0, CPB1, and one application image. The remote system update requires these addresses.

```
BLOCK                START ADDRESS    END ADDRESS
BOOT_INFO                 0x00000000      0x0020FFFF
FACTORY_IMAGE             0x00210000      0x0048FFFF
SPT0                      0x00490000      0x00497FFF
SPT1                      0x00498000      0x0049FFFF
CPB0                      0x004A0000      0x004A7FFF
CPB1                      0x004A8000      0x004AFFFF
Application Image         0x004B0000      0x006EAFFF

Configuration device: AGFB014R0
Configuration mode: Active Serial x4
Quad-Serial configuration device dummy clock cycle: 15

Notes:

- Data checksum for this conversion is 0xC0D25FD7
- All the addresses in this file are byte addresses
```

After generating the programming file, you can program the flash memory.

5.6.3. Programming Flash Memory with the Initial Remote System Update Image

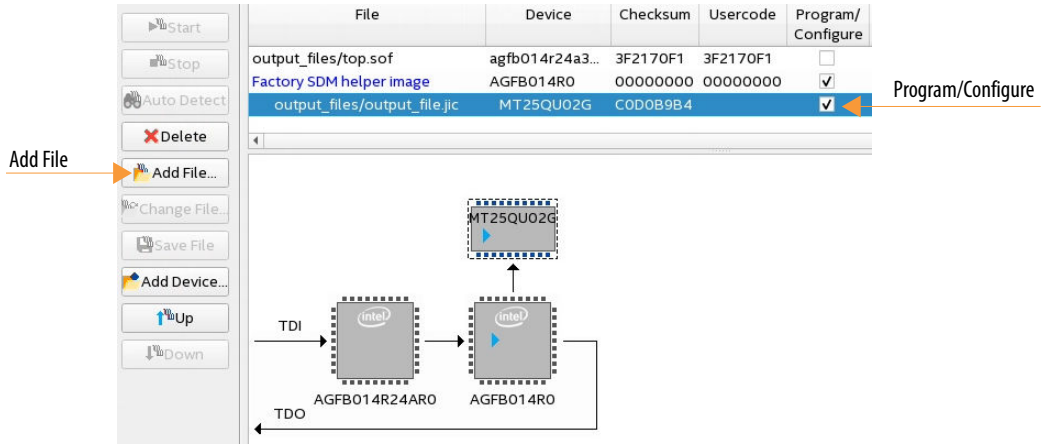
You can program the initial remote system update image from the command line. In the following command, substitute your `.jic` for `output_file.jic` if necessary.

```
quartus_pgm -c 1 -m jtag -o "pvi;./output_file.jic
```

Alternatively, you can use the Intel Quartus Prime Programmer to program the initial RSU update image by completing the following procedure:

1. open the **Programmer** and click **Add File**. Select the generated `.jic` file (`output_file.jic`) and click **Open**.
2. Turn on the **Program/Configure** for the attached `.jic` file.
3. To begin programming the flash memory with the initial remote system update image, click **Start**.
4. Configuration is complete when the progress bar reaches 100%. Power cycle the board to automatically configure the Intel Agilex device with the application image using the AS x4 configuration scheme.

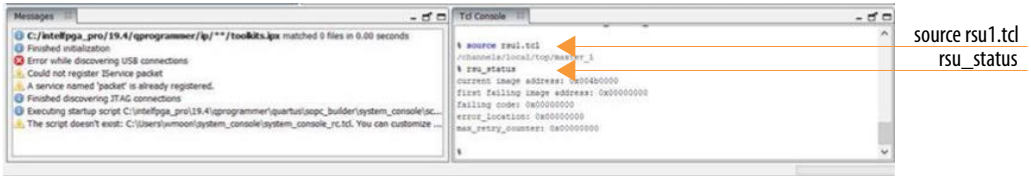
Figure 79. Programming the Flash Memory with the Initial RSU Image



Note: This example does not assign the **Direct to Factory Image** pin. Consequently, the Programmer configures the device with the application image. The application image is the default image if the design does not use the **Direct to Factory Image** pin.

5. Use the RSU_STATUS command to determine which bitstream image the Programmer is using as shown in the following example:
   a. In the Intel Quartus Prime software, select **Tools ➤ System Debugging Tools ➤ System Console** to launch the system console.
   b. In the Tcl Console pane, type `source rsu1.tcl` to open the example of Tcl script to perform the remote system update commands. Refer to the *Related Information* for a link to `rsu1.tcl`.
   c. Type the `rsu_status` command to report the current remote system update status. You can retrieve the current running image address from the remote system update status report. The current image address must match the start address for the application image printed in the `.map` file.

Figure 80. Running Tcl Commands Available in rsu1.tcl



Related Information

rsu1.tcl

5.6.4. Reconfiguring the Device with an Application or Factory Image

The following steps describe the process to reconfigure the device with a different application image or the factory image using operation commands after the device is in user mode.

1. The remote system update host sends the RSU_IMAGE_UPDATE command to perform the remote system update to the new application image or factory image.
   a. For example, in the Tcl console of the System Console, type the following command to initiate a remote system update to the factory image.
      i. `rsu_image_update 0x00110000`

This command reconfigures the device with factory image. Address `0x00110000` is the start address of the factory image as shown in the `.map` file. The JTAG host automatically disconnects from the System Console once the device reconfiguration is successful. You must restart the System Console to re-establish the connection with the device to perform next command.

ii. `rsu_image_update 0x002F4000`

This command reconfigures the device with the application image. Address `0x002F4000` is the start address of the application image as shown in the `.map` file.

Optional: Retrieve the remote system update status by using the `rsu_status` command to ensure you have successfully reconfigured the device.

2. In the Tcl console of the System Console, type `rsu_status` to verify the current image. The following figure shows the device is being reconfigured with the factory image.

Figure 81. Verify Current Image Using rsu_status Command

```
$ source rsu1.tcl
/channels/local/top/master_1
$ rsu_status
current image address 0x004b0000
first failing image address 0x00000000
failing code 0x00000000
error location 0x00000000
0x00000000
```

5.6.5. Adding an Application Image

Complete the following steps to add an application image to flash memory:

1. Set up exclusive access to the AS x4 interface and flash memory by running the `QSPI_OPEN` and `QSPI_SET_CS` commands in the Tcl Console window. You now have exclusive access to the AS x4 interface and flash until you relinquish access by running the `QSPI_CLOSE` command. Write the new application image to the flash memory using the `QSPI_WRITE` command.
2. Alternatively, the `rsu1.tcl` script includes the `program_flash` function that programs a new application image into flash memory. The following command accomplishes this task:

```
program_flash new_application_image.rpd 0x03FF0000 1024
```

The `program_flash` function takes three arguments:

a. The `.rpd` file to write to flash memory.

b. The start address.

c. Number of words to write for each `QSPI_WRITE` command. The `QSPI_WRITE` supports up to 1024 words per write instruction.

Figure 82. Program New Application Image

```
$ source rsu1.tcl
/channels/local/top/master_1
$ program_flash new_application_image.rpd 0x03ff0000 1024
total number of words is 584704
total number of page is 571
total number of sector is 36
reading rpd is completed
start writing flash
writing flash is completed
```

3. Write the new application image start address to a new image pointer entry in the configuration firmware pointer block (CPB) using the `QSPI_WRITE` command. Ensure that the new image pointer entry value is `0xFFFFFFFF` before initiating the write.

   Note: You must update both copies (CBP0 and CBP1) when editing the configuration firmware pointer block and sub-partition table. Refer to Table 1 for more details about the configuration firmware pointer block.

Based on the example described above, the address offset `0x20` in the CPB0 and CPB1 must point to the start address of the application image. The next new image pointer entry value must be `0xFFFFFFFF` before you write the start address of the new application image to the next image pointer entry.

You can use the `QSPI_read` function verify that the new image pointer entry value is `0xFFFFFFFF` . The `QSPI_read` function takes in two arguments:

1. Start address
2. Number of words to read

Figure 83. Verifying that the New Image Pointer entry Value is 0xFFFFFFFF

```
$ qspi_read 0x004a0020 1
0x004b0000
$ qspi_read 0x004a0028 1
0xffffffff
% qspi_read 0x004a8020 1
0x004b0000
% qspi_read 0x004a8028 1
0xffffffff
```

You can now proceed to write the new application image address to next image entry by using the `QSPI_write_one_word` function. The `QSPI_write_one_word` function takes in two arguments:

1. Address
2. The value of the word

Figure 84. Writing an Address Pointer to the New Image Pointer Entry

```
% qspi_write_one_word 0x004a0028 0x02000000
% qspi_write_one_word 0x004a8028 0x02000000
```

You can now do a `QSPI_read` function to the next image pointer entry to ensure that it is written with the start address of the desired new application image.

Verifying the Update to the New Image Pointer

```
% qspi_read 0x004a0028 1
0x02000000
% qspi_read 0x004a8028 1
0x02000000
```

Host software can now reconfigure the Intel Agilex FPGA with the new application image by asserting the `nCONFIG` pin. Alternatively, you can power cycle the PCB. After reconfiguration, check the current image address. The expected address is `0x03ff0000`. After adding a new image, your application image list includes the newly added application image and the old application image, which is now a secondary image. The newly added application image has the highest priority.

Note: When the remote system update host loads an application image, the decision firmware traverses the image pointer entries in reverse order. The new image has the highest priority when you restart the device.

5.6.6. Removing an Application Image

1. Set up exclusive access to the AS x4 interface and flash memory by running the QSPI_OPEN and QSPI_SET_CS commands in the Tcl Console window.. You now have exclusive access to the AS x4 interface and flash until you relinquish access by running the QSPI_CLOSE command. Write the new application image to the flash memory using the QSPI_WRITE command.
2. Write 0x00000000 to the application image start address stored in the image pointer entry of the configuration firmware pointer block (CPB0 and CPB1) using the QSPI_WRITE command.

   Note: You must update both copies (copy0 and copy1) when editing the configuration firmware pointer block and sub-partition table.
3. Erase the application image content in the flash memory using the QSPI_ERASE command.
4. To remove a new application image, add another new application image in the next or subsequent image pointer entry or allow the device to fall back to the previous or secondary application image in your application image list. The following table shows correct entries for image pointer entries for CPB0 and CPB1 for offsets 0x20 and 0x28 :

Table 45. Configuration Firmware Pointer Block Contents

CPB Start Address + 0x20	Content	Value
CPB0 + 0x20 = 0x004A0020	Old application image pointer entry (lower priority)	0x004B000
CPB0 + 0x28 = 0x004A0028	Current/new application image pointer entry (highest priority)	0x02000000
CPB1 + 0x20 = 0x004A8020	Old application image pointer entry (lower priority)	0x004B0000
CPB1 + 0x28 = 0x004A8028	Current/New application image pointer entry (highest priority)	0x02000000

Figure 85. Read Current CPB Values

```
% qspi_read 0x004A0020 1
0x004B0000
% qspi_read 0x004A0028 1
0x02000000
% qspi_read 0x004A8020 1
0x004B0000
% qspi_read 0x004A8028 1
0x02000000
```

You can now remove the current or new application image address image pointer entry by writing the value to `0x00000000` using the `QSPI_write_one_word` function as shown in the following example. The `QSPI_write_one_word` function takes address and data arguments. Be sure to erase the application content that you just removed from flash memory.

Figure 86. Remove Application Image

```
% qspi_write_one_word 0x004A0028 0x00000000
% qspi_write_one_word 0x002ec028 0x004A8028
```

You can use a `QSPI_read` to the image pointer entry at offset `0x28` for CBP0 and CPB1 to verify completion of the `QSPI_write_one_word` commands .

Figure 87. Verify the Writes

```
% qspi_read 0x004A0028 1
% qspi_read 0x004A8028 1
```

You can now configure the device with the old application image. The old application image has the highest priority if you power cycle the device or the host asserts the `nCONFIG` pin. You can run the `rsu_status` report to check the status of the current image address.

6. Intel Agilex Configuration Features

6.1. Device Security

Note: Contact your Intel sales representative for more information about the device security support in Intel Agilex devices.

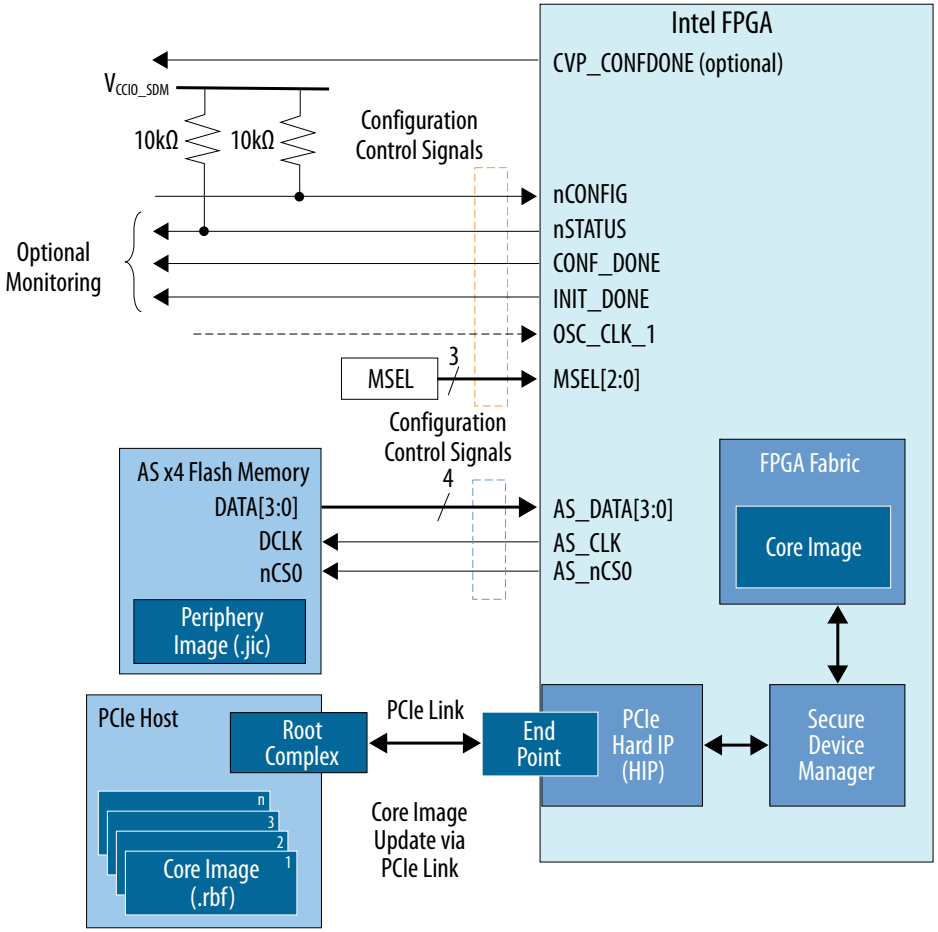
The Intel Agilex device provides the following flexible and robust security features to protect sensitive data and intellectual property:

- User image authentication and encryption
- Public-Key based authentication
- Advanced Encryption Standard (AES)-256 Encryption
- JTAG Disable
- JTAG Debug Disable/Enable
- Side channel protection
- Physical anti-tampering protection

6.2. Configuration via Protocol

The CvP configuration scheme creates separate images for the periphery and core logic. You can store the periphery image in a local configuration device and the core image in host memory, reducing system costs and increasing the security for the proprietary core image. CvP configures the FPGA fabric through the PCI Express (PCIe) link and is available for Endpoint variants only.

Figure 88. Intel Agilex CvP Configuration Block Diagram



The CvP configuration scheme supports the following modes:

- CvP Initialization Mode:

  In this mode an external configuration device stores the periphery image and it loads into the FPGA through the Active Serial x4 (Fast mode) configuration scheme. The host memory stores the core image and it loads into the FPGA through the PCIe link.

  After the periphery image configuration completes, the CONF_DONE signal goes high and the FPGA starts PCIe link training. When PCIe link training completes, the PCIe link transitions to the Link Training and Status State Machine (LTSSM) L0 state and then through PCIe enumeration. The PCIe host then configures the core through the PCIe link. The PCIe reference clock must be running for the link for link training.

  After the core image configuration is complete, the CvP_CONFDONE pin (if enabled) goes high, indicating the FPGA has receiver the full configuration bitstream over the PCIe link. INIT_DONE indicates that configuration is complete.

- CvP Update Mode:

  CvP update mode is a reconfiguration scheme that uses the PCIe link to deliver an updated bitstream to a target device after the device enters user mode. The periphery images which includes the PCIe link remains active, allowing CvP update to use this link to reconfigure the core fabric. In this mode, the FPGA device initializes by loading the full configuration image from the external local configuration device to the FPGA or after CvP initialization.

  You can perform CvP update on a device that you originally configure using CvP initialization or any other configuration scheme.

6.3. Partial Reconfiguration

Partial reconfiguration (PR) allows you to reconfigure a portion of the FPGA dynamically, while the remaining FPGA design continues to function. You can define multiple personas for a region in your design, without impacting operation in areas outside this region. This methodology is effective in systems with multiple functions that time-share the same FPGA device resources. PR enables the implementation of more complex FPGA systems.

Related Information

Intel Quartus Prime Pro Edition User Guide: Partial Reconfiguration

7. Intel Agilex Debugging Guide

7.1. Configuration Debugging Checklist

Work through this checklist to identify issues that may result in operational failures.

Table 46. General Configuration Debugging Checklist

	Checklist Item	Complete?
1	Verify that the V_{cc} ,V_{ccp} ,V_{ccio_sdm} V_{ccpt} ,V_{cceram}, V_{ccadc} supplies are in the proper range by using SDM Debug Toolkit.	☐
2	Verify that all configuration resistors are correctly connected (MSEL, nCONFIG, nSTATUS, CONF_DONE, INIT_DONE).	☐
3	Verify that you are following the correct power-up and power-down sequences.	☐
4	Verify that the SDM I/Os assignments are correct by checking the Intel Quartus Prime Compilation QSF and Fitter reports.	☐
5	For SmartVID devices (-V or -E), ensure that all PMBus pins are connected to Intel Agilex device.	☐
6	Verify that SmartVID settings follow the recommendations in the *Intel Agilex Power Management User Guide*	☐
7	Verify that the Intel Agilex -V or -E device has its own voltage regulator module for V_{CC}, V_{CCP}, V_{CCL_HPS}, and $V_{CCPLLDIG\_HPS}$	☐
8	After configuration are the nCONFIG, nSTATUS CONF_DONE, and INIT_DONE pins high? Use the SDM Debug Toolkit to determine these levels.	☐
9	Is the SDM operating Boot ROM code or configuration firmware? Use the SDM Debug Toolkit to answer this question.	☐
10	Are the MSEL pins correctly connected on board? Use the SDM Debug Toolkit to answer this question.	☐
11	For designs that use transceivers, HBM2, PCIe, or EMIF, are the reference clocks stable and free running before configuration begins?	☐
12	Does your design include the Reset Release IP?	☐

continued...

	Checklist Item	Complete?
13	To avoid configuration failures, disconnect the PMBus regulator's JTAG download cable before configuring Intel Agilex -V devices.	☐
14	If the SDM Debug Toolkit is not operational, verify that the Intel Agilex device has exited POR by checking nCONFIG, nSTATUS, CONF_DONE and INIT_DONE pins using an oscilloscope.	☐
15	Is the configuration clock source chosen appropriately? You can use an internal oscillator or the OSC_CLK_1 pin.	☐
16	For designs driving the OSC_CLK_1 pin is the frequency 25, 100, or 125 MHz?	☐
17	For Intel Agilex SX parts ensure that the HPS and EMIF IOPLL are stable and free running before configuration begins. The actual frequency should match the setting specified in Platform Designer.	☐
18	Are proper slave addresses set for the PMBus voltage regulator modules using the Intel Quartus Prime Software?	☐
19	For designs that use 3 V I/0, verify that the transceiver tiles are powered up before configuration begins.	☐

Related Information

Intel Agilex Power Management User Guide

7.2. Intel Agilex Configuration Architecture Overview

Intel Agilex devices employ configuration architecture that is quite similar to the Intel Stratix 10 architecture. The Secure Device Manager (SDM), a dedicated hard processor, controls and monitors all aspects of device configuration from device power-on reset. This configuration architecture is different from earlier Intel FPGA device families where state machines control configuration.

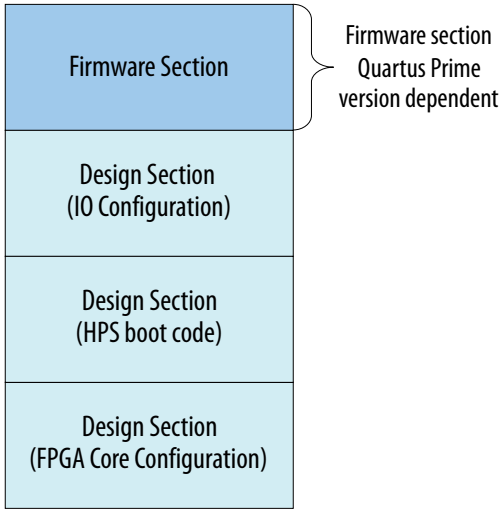
There are important differences between Intel Agilex and Intel Stratix 10 devices and previous device families with respect to available configuration modes, configuration pin behavior, and connection guidelines. In addition, the bitstream format is different. Knowing about these differences and how these pins behave can help you understand and debug configuration issues.

7.3. Configuration File Format Differences

Detailed information about the configuration file format is proprietary. This topic explains the general structure and differences from previous device families.

The configuration file format differs significantly from previous device families. The configuration bitstream begins with a SDM firmware section. The SDM loads the boot ROM firmware during power-on reset. Design sections for I/O configuration, HPS boot code (if applicable), and fabric configuration follow the firmware section. Configuration begins after the SDM boot ROM performs device consistency checks.

Figure 89. Example of an Intel Agilex Configuration Bitstream Structure



The firmware section is not part of the `.sof` file. The Intel Quartus Prime Pro Edition Programmer adds the firmware to the `.sof`. The programmer adds the firmware when configuring an Intel Agilex device or when it converts the `.sof` to another format. The version of firmware that the Programmer adds depends on the version of the Programmer you are using.

7.4. Understanding SEUs

SEUs are rare, unintended changes in the state of an FPGA's internal memory elements caused by cosmic radiation effects. The change in state is a soft error and the FPGA incurs no permanent damage. Because of the unintended memory state, the FPGA may operate erroneously until background scrubbing fixes the upset.

The Intel Quartus Prime software offers several features to detect and correct the effects of SEU, or soft errors, and to characterize the effects of SEU on your designs. LSM firmware provides SEU single bit error correction and multi-bit error detection per LSM. Additionally, some Intel FPGAs contain dedicated circuitry to help detect and correct errors.

For more information about SEUs, refer to *Intel Agilex SEU Mitigation User Guide*.

7.5. Reading the Unique 64-Bit CHIP ID

he Chip ID Intel FPGA IP in each Intel Agilex device stores a unique 64-bit chip ID. Refer to the *Mailbox Avalon ST Client IP User Guide* learn how to read the Chip ID from the Intel Agilex device.

7.6. E-Tile Transceivers May Fail To Configure

Making the PRESERVE_UNUSED_XCVR_CHANNEL assignment to completely unused E-tile transceivers may cause configuration failures in Intel Agilex devices.

The Intel Quartus Prime Programmer detects an internal error and fails to configure your device under the following conditions:

- You have made the PRESERVE_UNUSED_XCVR_CHANNEL assignment to an entire unused E-tile.
- Your design does not provide a reference clock to this unused E-tile.

The reference clock is necessary to generate a pseudo-random data signal to prevent the transceiver from degrading over time. You must instantiate at least one dummy channel in the E-tile using the Native PHY IP GUI. Provide this channel at least one reference clock. All preserved channels in a single E-tile can use the same reference clock.

When your design uses some channels in an E-tile, you can use the per-pin PRESERVE_UNUSED_XCVR_CHANNEL QSF assignment to preserve only the channels in the E-tile that you intend to use. If you never intend to use a channel, you should not add the per-pin PRESERVE_UNUSED_XCVR_CHANNEL QSF assignment.

Here are some examples of PRESERVE_UNUSED_XCVR_CHANNEL QSF assignments.

```
#Global QSF assignment
set_global_assignment -name PRESERVE_UNUSED_XCVR_CHANNEL ON

#Per-pin QSF assignment
set_instance_assignment -name PRESERVE_UNUSED_XCVR_CHANNEL ON -to AA75
```

Related Information

Unused Transceiver Channels
For more detailed information about preserving unused transceiver channels in E-tile devices.

7.7. Understanding and Troubleshooting Configuration Pin Behavior

Configuration typically fails for one of the following reasons:

- The host times outs
- A configuration data error occurs
- An external event interrupts configuration
- An internal error occurs

Here are some very common causes of configuration failures:

- Check OSC_CLK_1 frequency. It must match the frequency you specified in the Intel Quartus Prime Software and the clock source on your board.
- Ensure a free running reference clock is present for designs using transceivers, PCIe, or HBM2. These reference clocks must be available until the device enters user mode.
- For designs using the HPS and the external memory interface (EMIF), ensure that the EMIF clock is present.
- For designs using SmartVID (-V and -E devices), ensure that this feature is set-up and operating correctly. Ensure that the voltage regulator supports SmartVID.

Here are some debugging suggestions that apply to any configuration mode:

- To rule out issues with OSC_CLK_1 select the **Internal Oscillator** option in the Intel Quartus Prime.
- Try configuring the Intel Agilex device with a simple design that does not contain any IP. If configuration via a non-JTAG scheme fails with a simple design, try JTAG configuration with the MSEL pins set specifically to JTAG.

The following topics describe the expected behavior of configuration pins. In addition, these topics provide some suggestions to assist in debugging configuration failures. Refer to the separate sections on each configuration scheme for debugging suggestions that pertain to a specific configuration scheme.

Related Information

- Debugging Guidelines for the Avalon-ST Configuration Scheme on page 50
- Debugging Guidelines for the AS Configuration Scheme on page 104
- Debugging Guidelines for the JTAG Configuration Scheme on page 116

7.7.1. nCONFIG

The nCONFIG pin is a dedicated, input pin of the SDM. nCONFIG has two functions:

- Hold-off initial configuration
- Initiate FPGA reconfiguration

The nCONFIG pin transition from low to high signals a configuration or reconfiguration request. The nSTATUS pin indicates device readiness to initiate FPGA configuration.

The configuration source can only change the state of the nCONFIG pin when it has the same value as nSTATUS. When the Intel Agilex device is ready it drives nSTATUS to follow nCONFIG.

The host should drive nCONFIG low to initiate device cleaning. Then the host should deassert nCONFIG initiate configuration. If the host drives nCONFIG low during a configuration cycle, that configuration cycle stops. The SDM expects a new configuration cycle to begin.

Debugging Suggestions

The host drives nCONFIG. Be sure that it is not floating or stuck low. nCONFIG should remain high during configuration.

7.7.2. nSTATUS

nSTATUS has the following two functions:

- To behave as an acknowledge for nCONFIG.
- To behave as an error status signal. It is important to monitor nSTATUS to identify configuration failures.

Note: nSTATUS does not go low for PR failures or failures using the JTAG configuration scheme.

Generally, the Intel Agilex device changes the value of nSTATUS to follow the value of nCONFIG, except after an error. For example, after POR, nSTATUS asserts after nCONFIG asserts. When the host drives nCONFIG high, the Intel Agilex device drives nSTATUS high.

In previous device families the deassertion of nSTATUS indicates the device is ready for configuration. For Intel Agilex devices, when using Avalon-ST configuration scheme, after the Intel Agilex device drives nSTATUS high, you must also monitor the AVST_READY signal to determine when the device is ready to accept configuration data.

nSTATUS asserts if an error occurs during configuration. The pulse ranges from 0.5 ms to 10 ms.

nSTATUS assertion is asynchronous to data error detection. Intel Agilex devices do not support the **auto-restart configuration after error** option.

Previous device families implement the nSTATUS as an open drain with a weak internal pull-up. Intel Agilex always drives nSTATUS. Consequently, you cannot wire OR an Intel Agilex nSTATUS signal with the nSTATUS signal from earlier device families.

Debugging Suggestions

Ensure nSTATUS acknowledges nCONFIG. If nSTATUS is not following nCONFIG, the FPGA may not have exited POR. You may need to power cycle the PCB.

7.7.3. CONF_DONE and INIT_DONE

For Intel Agilex devices, both CONF_DONE and INIT_DONE share multiplexed SDM_IO pins. Previous device families implement the CONF_DONE and INIT_DONE pins as open drains with a weak internal pull-up. Consequently, you cannot wire OR an Intel Agilex CONF_DONE or INIT_DONE signal with the nSTATUS signal from previous device families. Otherwise, CONF_DONE and INIT_DONE behave as these signals behaved in earlier device families. If you assign CONF_DONE and INIT_DONE to SDM_IO16 and SDM_IO0, weak internal pull-downs pull these pins low at power-on reset. Ensure you specify these pins in the Intel Quartus Prime Software or in the Intel Quartus Prime settings file, (.qsf). CONF_DONE and INIT_DONE are low prior to and during configuration. CONF_DONE asserts when the device finishes receiving configuration data. INIT_DONE asserts when the device enters user mode.

Note: The entire device does not enter user mode simultaneously.Intel recommends that you include the Including the Reset Release Intel FPGA IP in Your Design on page 118 to hold your application logic in the reset state until the entire FPGA fabric is in user mode.

CONF_DONE and INIT_DONE are optional signals. You can use these pins for other functions that the Intel Quartus Prime Pro Edition **Device and Pin Options** menu defines.

Debugging Suggestions

Place the CONF_DONE and INIT_DONE pins on the SDM_IO pins that correlate with the board-level connection. Refer to *SDM Pin Mapping* and *Setting Additional Configuration Pins* for more information.

7.7.4. SDM_IO Pins

Intel Agilex devices include 17 SDM_IO pins that you can configure to implement specific functions such as CONF_DONE and INIT_DONE. The configuration bitstream controls the pin locations for the SDM_IO pins.

Internal Intel Agilex circuitry pulls SDM_IO0, SDM_IO8 and SDM_IO16 weakly low through a 25 kΩ resistor. Internal Intel Agilex circuitry pulls all other SDM_IO pins weakly high during power-on.

Debugging Suggestions

Check the Intel Quartus Prime Pro Edition settings and Fitter report to ensure that the SDM_IO configuration matches your PCB design. The following screen shots show where to configure these signals and how to confirm the SDM_IO pin settings in the Fitter report.

Figure 90. Configuration Pin Selection in the Intel Quartus Prime Pro Edition Software

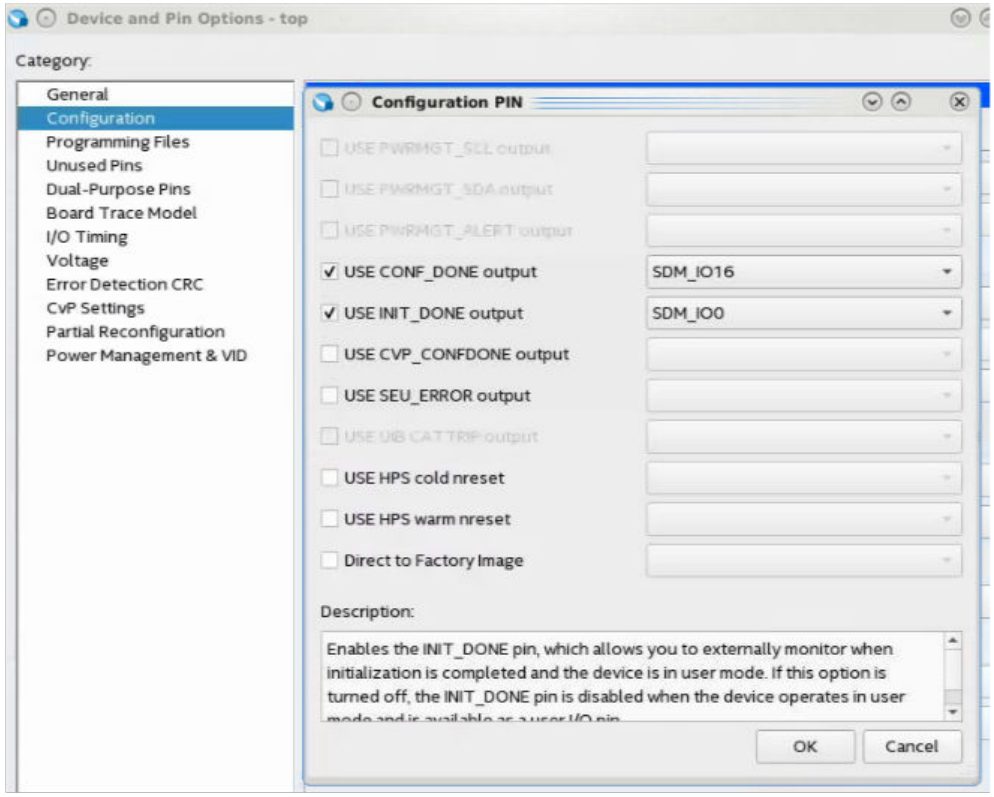
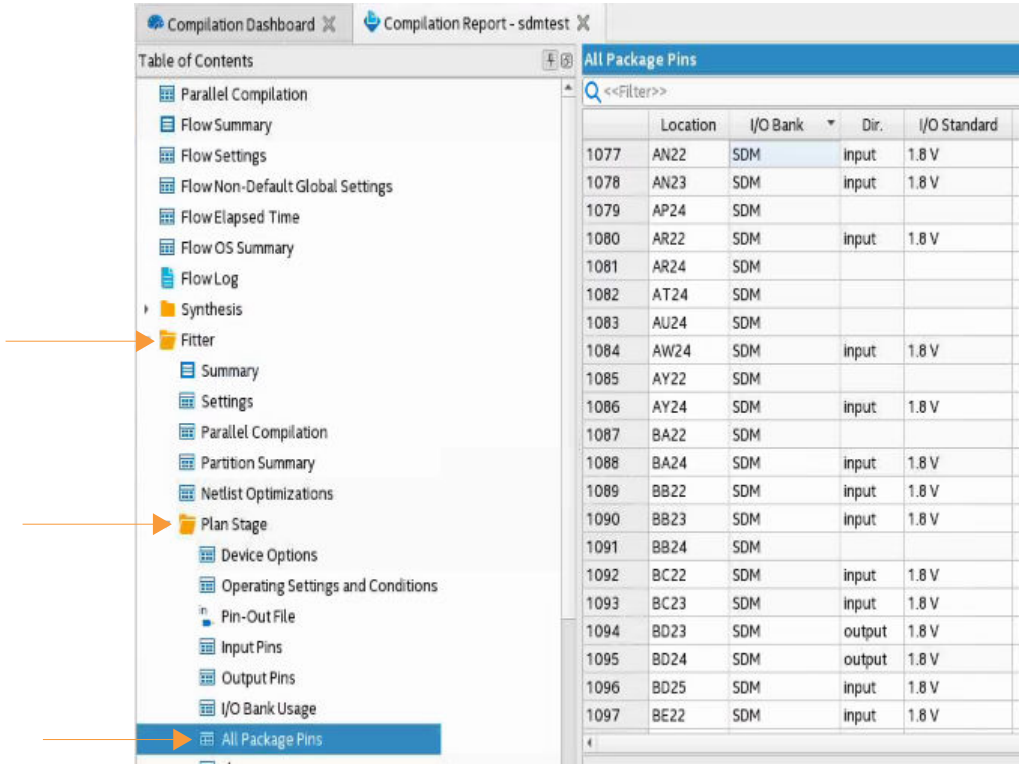


Figure 91. Fitter Report and SDM_IO Pin Reporting



Starting with the Intel Quartus Prime Pro Edition Software, version 18.1, an SDM Debug Toolkit is available through the System Console, **Tools ➤ System Debugging Tools ➤ System Console ➤ Intel Agilex SDM Debug Toolkit**.

8. Intel Agilex Configuration User Guide Archives

If the table does not list a software version, the user guide for the previous software version applies.

Intel Quartus Prime Version	User Guide
19.3	Intel Agilex Configuration User Guide
19.2	Intel Agilex Configuration User Guide
19.1	Intel Agilex Configuration User Guide

9. Document Revision History for the Intel Agilex Configuration User Guide

Document Version	Intel Quartus Prime Version	Changes
2020.03.13	19.4	Made the following changes: • Updated maximum supported AS_CLK frequency in the *Maximum AS_CLK Frequency as a Function of Board Capacitance Loading and Clock Source* table. The AS_CLK maximum frequency for 37 pF capacitance loading using OS_CLK_1 as a clock source is 71.5 MHz, not 80 MHz. • Removed 80 MHz support from the *Supported configuration clock source and AS_CLK Frequencies in Intel Agilex Devices* table.
2020.01.08	19.4	Made the following changes: • Corrected Data Width (bits) field for CvP in *Table 1. Intel Agilex Configuration Data Width, Clock Rates, and Data Rates*. Intel Agilex support x8 and x16 CvP for the Gen3 and Gen4 data rates.
2019.12.16	19.4	Made the following changes: • Added a new chapter covering the Reset Release Intel FPGA IP and why it must be included in your design. • Added the following components to the *Required Communication and Host Components for the Remote System Update Design Example* figure: — Reset Release Intel FPGA IP — Reset Bridge Intel FPGA IP the to the RSU design example shown the • Updated *Remote System Update (RSU)* chapter to provide accurate addresses for Intel Agilex devices. • Updated *Remote System Update (RSU)* chapter to provide figures with Intel Agilex devices. • Added the following text to the *OSC_CLK_1 Clock Input* topic: *When you specify OSC_CLK_1 for configuration and reconfigure without powering down the Intel Agilex device, the device can only reconfigure with OSC_CLK_1. In this scenario, OSC_CLK_1 must be a free-running clock.* • Added the following text to the definition of the `Failing image` field of the RSU_STATUS command: *Note:* A rising edge on `nCONFIG` to reconfigure from ASx4, does not clear this field. Information about failing image only updates when the Mailbox Client receives a new `RSU_IMAGE_UPDATE` command and successfully configures from the update image.

continued...

Document Version	Intel Quartus Prime Version	Changes
		• Added the following restriction to the definition of QSPI_SET_CS: *Access to the QSPI flash memory devices using SDM_IO pins is only available for the AS x4 configuration scheme, JTAG configuration, and a design compiled for ASx4 configuration. For the Avalon ST configuration scheme, you must connect QSPI flash memories to GPIO pins.* • Removed support for a 16-byte Version ID as the first 16 bytes of the application image. This feature is not supported in Intel Agilex devices. • Updated the final suggestion in *Debugging Guidelines for the JTAG Configuration Scheme* topic, to the following: *When the MSEL setting on the PCB is not JTAG, if you use the JTAG interface for reconfiguration after an initial reconfiguration using AS or the Avalon-ST interface, the `.sof` must be in the file format you specified in the Intel Quartus Prime project. For example, if you initially configure the MSEL pins for AS configuration and configure using the AS scheme, a subsequent JTAG reconfiguration using a `.sof` generated for Avalon-ST fails.*
2019.10.09	19.3	Made the following changes: • Corrected definition of RSU_STATUS command. This command has 9, not 10 words. • Added *E-Tile Transceivers May Fail To Configure* to the *Debugging chapter.* • Revised the *Modifying the List of Application Images* topic.
2019.09.30	19.3	Made the following changes to the device and software: • Added the optional nCATTRIP (catastrophic trip) SDM I/O signal. This signal asserts when the core temperature is greater than 125° C. • Added the an eighth word to the to the RSU_STATUS response: Word 8: Current image retry counter. • Added new field to the 5th word of the RSU_STATUS response. This field specifies the source of a reported error. • Added RSU_NOTIFY to the available operation commands. • Changed the number of images that the Programming File Generator supports from 3 to 7. • Removed write restrictions for lower addresses in flash memory. (The device firmware must still reside at address 0x0.) Made the following changes to the user guide: • Added many topics showing how to implement in the Intel Quartus Prime Pro Edition Software. • Changed the err status pulse range from 1 ms ±50% to 0.5 ms to 10 ms. • Removed the SDM Firmware state from the Intel Intel Agilex FPGA Configuration Flow diagram. This state is part of the FPGA Configuration state. • Updated recommendations on how to debug a corrupt configuration bitstream for the AS x4 configuration scheme in the *Debugging Guidelines for the AS Configuration Scheme* topic. • Corrected the signal name in *The AVST_READY Signal* topic: *The device can starting sending data when AVST_READY asserts.* • Added note that the Avalon ST x32 configuration scheme is limited to 3, DDR x72 DDR external memory interfaces. The Avalon ST x8 and x16 configuration schemes can support up to 4, x72 DDR external memory interfaces. • Corrected the Pin Type in the *Required Configuration Signals for the Avalon-ST Configuration Scheme* table. AVSTx8_READY is an SDM I/O pin. AVST_READY is a GPIO or Dual-Purpose pin. • Corrected minor errors and typos.

continued...

Document Version	Intel Quartus Prime Version	Changes
2019.07.01	19.2	Made the following changes: • Corrected Step 3 in the *Initial Configuration Timing* description. The step should say, with `nConfig` low, the SDM enters Idle mode after booting. • Added note that designs using Avalon-ST x16 and x32 configuration scheme may need to include a voltage translator between the FPGA and external host because some signals, to accommodate the GPIO pins that only support the 1.2 V I/O standard and the SDM I/O pins require a 1.8 V power supply. • Created separate topic covering partial configuration. • Revised and reorganized all topics covering configuration pin assignments: — Clarified the behavior of the `MSEL` pins in AS x4 mode. — Added information about the SDM_IO pin states during power-on and after device cleaning to the *Intel Agilex Configuration Pins* topic. — Created separate topics covering partial configuration and SmartVID signals. • Made the following changes to the RSU chapter: — Added the following topics: • *RSU Glossary* • *Standard (non-RSU) Flash Layout* • *RSU Flash Layout – SDM Perspective* • *RSU Flash Layout – Your Perspective* • *Detailed Quad SPI Flash Layout* • *Sub-partitions Layout* • *Sub-Partition Table Layout* • *CMF Pointer Block Layout* • *Modifying the List of Application Images* • *Application Image Layout* • *Command Sequence To Perform Quad SPI Operations* — The static firmware has been replaced by decision CMF. — The update image now includes the factory image, the decision CMF and the decision CMF data. — The `QSPI_ERASE` command is now 4 KB aligned. The number of words to erase must be a multiple of 1024. — Added definitions of major and minor error codes for `RSU_STATUS` and `CONFIG_STATUS`. • Added footnote explaining that before you can use CvP you must configure either the periphery image or the full image via the AS configuration scheme. Then, you can configure the core image using CvP. • Added recommendation to use the Analog Devices LTM4677 device to regulate the PMBus for SmartVID devices. You set this parameter here: **Device ➤ Device and Pin Options ➤ Power Management & VID ➤ Slave device type**. • Corrected maximum speed and data rate in the *Intel Agilex Configuration Data Width, Clock Rates, and Data Rates* table. The Max Clock Rate is 33 MHz. The Max Data Rate is 33 Mbps. • Added the eSRAM clocks to the list of free-running clocks that must be stable before configuration begins. • The Reset Release Intel Agilex FPGA IP is now available for Intel Agilex devices.

continued...

Document Version	Intel Quartus Prime Version	Changes
		• Removed vector for `Power_Supply_Status` in the *Configuration, Reconfiguration, and Error Timing Diagram* figure. • Corrected the *Intel Agilex* FPGA Configuration Flow diagram. The transition between `FPGA Config*` and `User Mode` should say `INIT_DONE = HIGH`. • Corrected the following statement in the *Debugging Guidelines for the JTAG Configuration Scheme* topic: *An nSTATUS falling edge terminates any JTAG access and the device reverts to the MSEL-specified boot source. nSTATUS must be stable during JTAG configuration.*. In both sentence, nSTATUS should be *nCONFIG.* • Removed pin assignments for `CVP_CONFDONE` for the Avalon-ST in the *Available SDM I/O Pin Assignments for Configuration Signals that Do Not Use Dedicated SDM I/O Pins* table. CvP does not the support Avalon-ST x8 configuration scheme in Intel Agilex devices.
2019.04.03	19.1	Removed references to documents that are not yet available.
2019.04.02	19.1	Initial Release