Device Configuration User Guide: Agilex™ 5 FPGAs and SoCs

Updated for Quartus® Prime Design Suite: **24.1**

Online Version
Send Feedback

813773
2024.04.01

Contents

1. Device Configuration User Guide: Agilex™ 5 FPGAs and SoCs

1.1. Agilex™ 5 Configuration Overview

All Agilex™ 5 FPGAs and SoCs are devices that 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 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 Quartus Prime software also compresses FPGA bitstreams, reducing memory utilization such as the on-board quad SPI flash device that is storing the FPGA bitstreams.

Intel describes configuration schemes from the point of view of the FPGA. Agilex 5 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 Agilex 5 configuration schemes support design security and partial reconfiguration. All Agilex 5 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.

Agilex 5 devices support the following configuration schemes:

- Avalon® streaming (Avalon-ST)
- JTAG
- Configuration via Protocol (CvP)
- Active Serial (AS) normal and fast modes

Table 1. Agilex 5 Devices Configuration Scheme, Data Width, and MSEL

Configuration Scheme		Data Width (bits)	MSEL[2:0]
Passive	Avalon-ST	16	101
		8	110
	JTAG	1	111
	Configuration via Protocol (CvP)	x4, x8 lanes(1)	001(2)
Active	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 Agilex 5 devices. Avalon-ST configuration supports x8 and x16 modes. The x16 bit mode uses general-purpose I/Os (GPIOs) for configuration. The x8 bit mode uses dedicated SDM I/O pins.

Note: The AVST_data[15:0], 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. Refer to the Enabling Dual-Purpose Pins on page 41 section for more information.

Avalon-ST supports backpressure using the AVST_READY and AVST_VALID pins. Because the time to process the incoming bitstream varies, backpressure support is necessary to transfer data to the Agilex 5 device. For more information about the Avalon-ST refer to the *Avalon Interface Specifications*.

JTAG

You can configure the Agilex 5 device using the dedicated JTAG pins. The JTAG port provides seamless access to many useful tools and functions. In addition to configuring the Agilex 5, you use the JTAG port for debugging with Signal Tap or the System Console tools.

(1) For more information, refer to the *Agilex 5 Configuration via Protocol (CvP) Implementation User Guide*.

(2) Before you can configure the core image using CvP, you must configure either the periphery image or the full image configuration via the AS scheme. The CvP core image is loaded by the host via PCIe link, which is referred to as passive.

The JTAG port has the highest priority and overrides the MSEL pin settings. Consequently, you can configure the Agilex 5 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 Agilex 5 device over the PCIe link. You can specify up to a x8 PCIe link. Agilex 5 devices support two CvP modes, CvP initialization and CvP update.

Note: For fast configuration data sources, the data rate of the device's internal configuration data path might limit the configuration data rate; and, not the PCIe link width.

CvP initialization process includes the following two steps:

1. During the board power up, the CvP uses quad SPI memory in AS x4 mode to configure the FPGA with the periphery image to enable CvP interface that includes the PCIe IP. The 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 initialization configuration. After the Agilex 5 enters user mode, you can use the CvP update mode to reconfigure the FPGA fabric. This mode has the following advantages:

- Allows to change core algorithms logic blocks.
- 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 Agilex 5 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 that this mode does not delay for 10 ms before beginning configuration. Use this mode to meet the link 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 Agilex 5 device accesses flash memory immediately after exiting reset. The power supply must be able to provide an equally fast ramp up for the Agilex 5 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.

Related Information

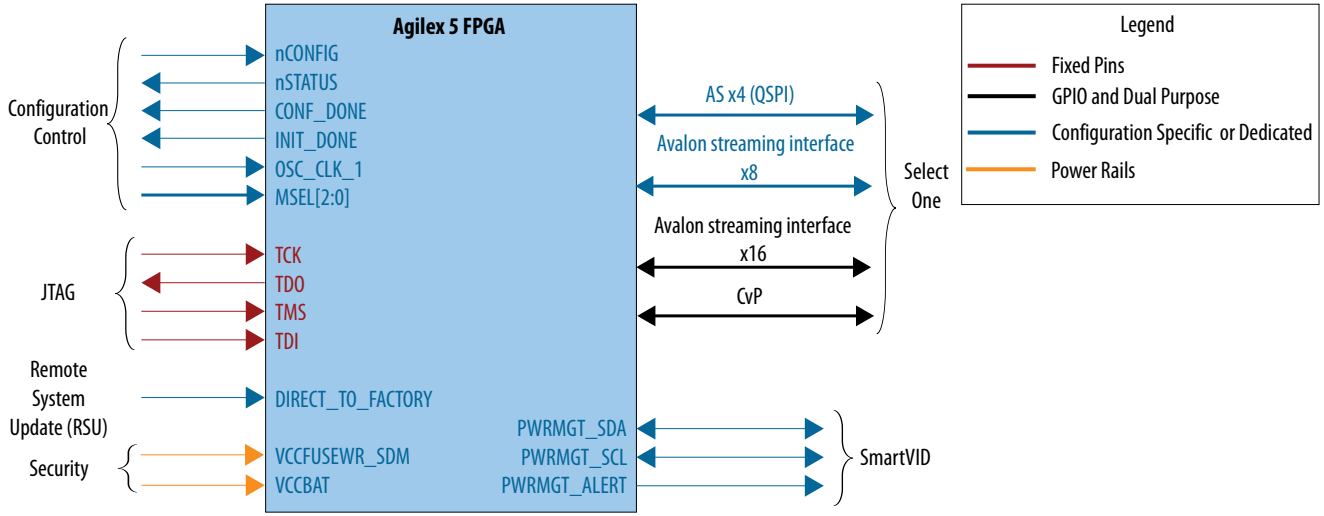
- Configuration via Protocol (CvP) Implementation User Guide: Agilex 5 FPGAs and SoCs
- Agilex 5 FPGAs and SoCs Device Data Sheet
- Agilex 5 FPGAs and SoCs Device Overview
- Avalon Interface Specifications

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 Quartus Prime software.

Figure 1. Agilex 5 Configuration Interfaces



This user guide discusses most of the interfaces shown in the figure. Refer to the separate *Agilex 5 Configuration via Protocol (CvP) Implementation User Guide* and *Agilex 5 Power Management User Guide* for more information about those features.

Related Information

- SDM Pin Mapping on page 29
- Configuration via Protocol (CvP) Implementation User Guide: Agilex 5 FPGAs and SoCs
- Power Management User Guide: Agilex 5 FPGAs and SoCs

1.1.2. Intel Download Cables Supporting Configuration in Agilex 5 Devices

Intel provides the following cables to download your design to the Agilex 5 device on the PCB. Download cables support prototyping activity by providing detailed debug messages via 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. Agilex 5-Supported Download Cable Capabilities

Download Cable	Protocol Support Agilex 5 Device	Cable Connection to PCB
Intel® FPGA Download Cable II (formerly the USB-Blaster II)	JTAG, AS	10-pin female plug
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.

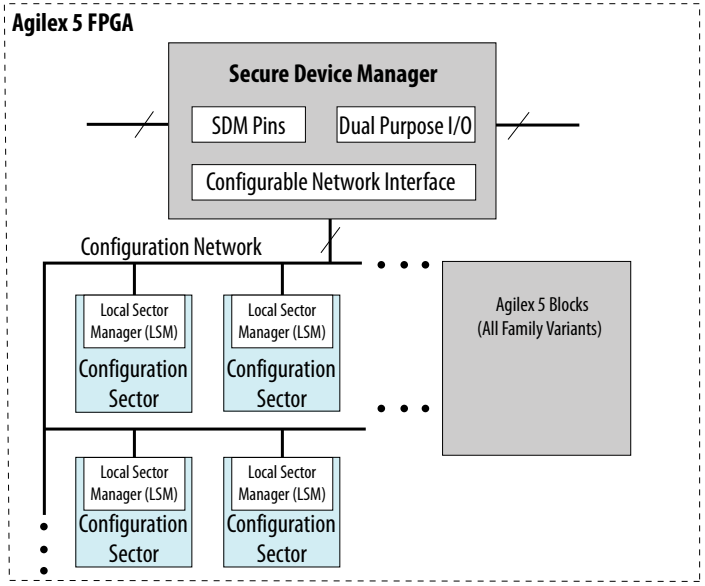
1.2. Agilex 5 Configuration Architecture

The Secure Device Manager (SDM) is a triple-redundant processor-based module that manages configuration and the security features of Agilex 5 devices. The SDM is available on all Agilex 5 devices.

The block diagram below provides an overview of the Agilex 5 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 integrity checks in user mode
- Other sub-systems: Refer to the *Agilex 5 FPGAs and SoCs Device Overview*.

Figure 2. Agilex 5 Configuration Architecture Block Diagram



Related Information

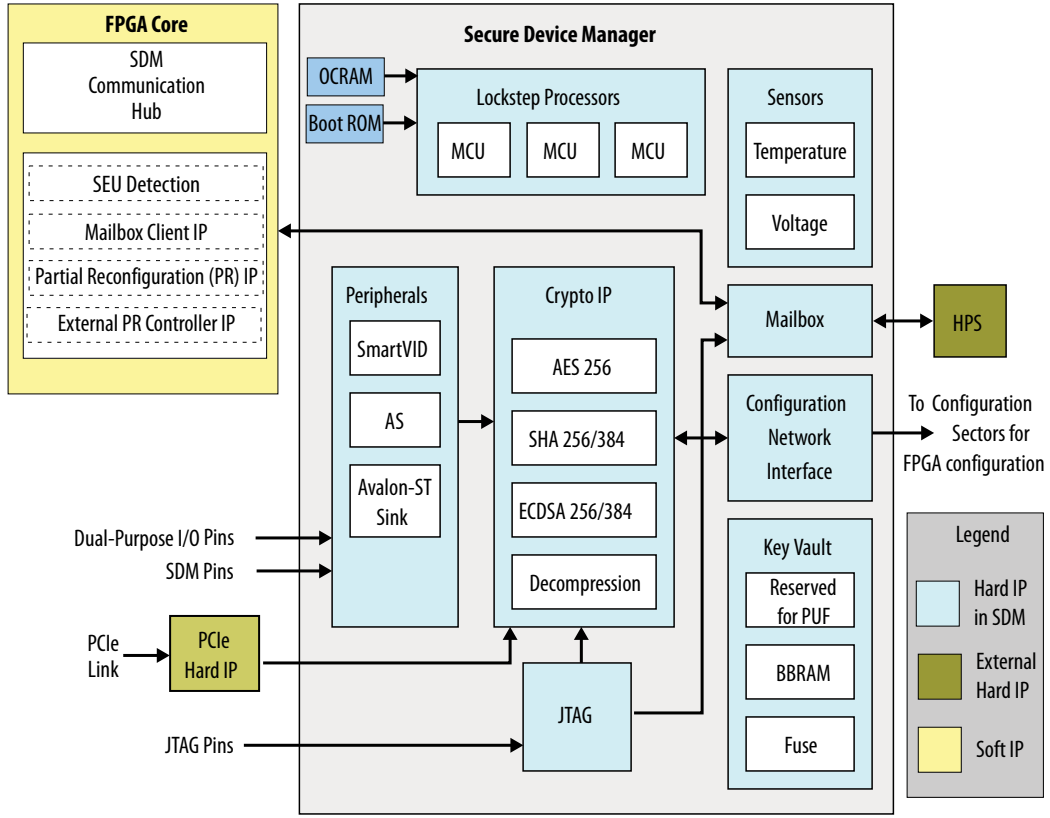
Agilex 5 FPGAs and SoCs Device Overview

1.2.1. Secure Device Manager

The SDM comprises peripherals, cryptographic IP and sensors, boot ROM, triple-redundant lockstep processors, and other blocks shown in the *SDM Block Diagram* figure. The SDM performs and manages the following security functions:

- Configuration bitstream authentication: During the configuration state, the SDM authenticates the Intel-generated configuration firmware and configuration bitstream, ensuring that configuration bitstream is from a trusted source. All Agilex 5 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.

Figure 3. SDM Block Diagram



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

- SDM uses temperature sensor for SmartVID feature to communicate to the 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 enables active configuration schemes via dedicated SDM pins.

- The Avalon-ST x8 configuration scheme uses SDM I/O pins. The Avalon-ST x16 configuration scheme uses 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 Quartus Prime software compresses the configuration data. All Agilex 5 devices compress the configuration bitstream. This feature is always enabled. When specifying an encrypted configuration bitstream, the Quartus Prime Pro Edition software compresses the configuration bitstream before encryption.
- A specific PCIe block included in the Agilex 5 device supports CvP.

Related Information

- SDM Pin Mapping on page 29
- Power Management User Guide: Agilex 5 FPGAs and SoCs

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 Quartus Prime Pro Edition Release to the bitstream generated from 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`

For more information about the output file types, refer to the *Quartus Prime Pro Edition User Guide: Programmer*.

Newer versions of the 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 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 Quartus Prime Pro Edition software. You can simply regenerate your configuration bitstream with the new version of the **Programming File Generator**.

Related Information

Quartus Prime Pro Edition User Guide: Programmer
For information about the programming file generator output file types.

1.2.1.2. Specifying Boot Order for Agilex 5 SoC Devices

For Agilex 5 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. Note that FPGA First option does not allow FPGA reconfiguration by using HPS. This user 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.

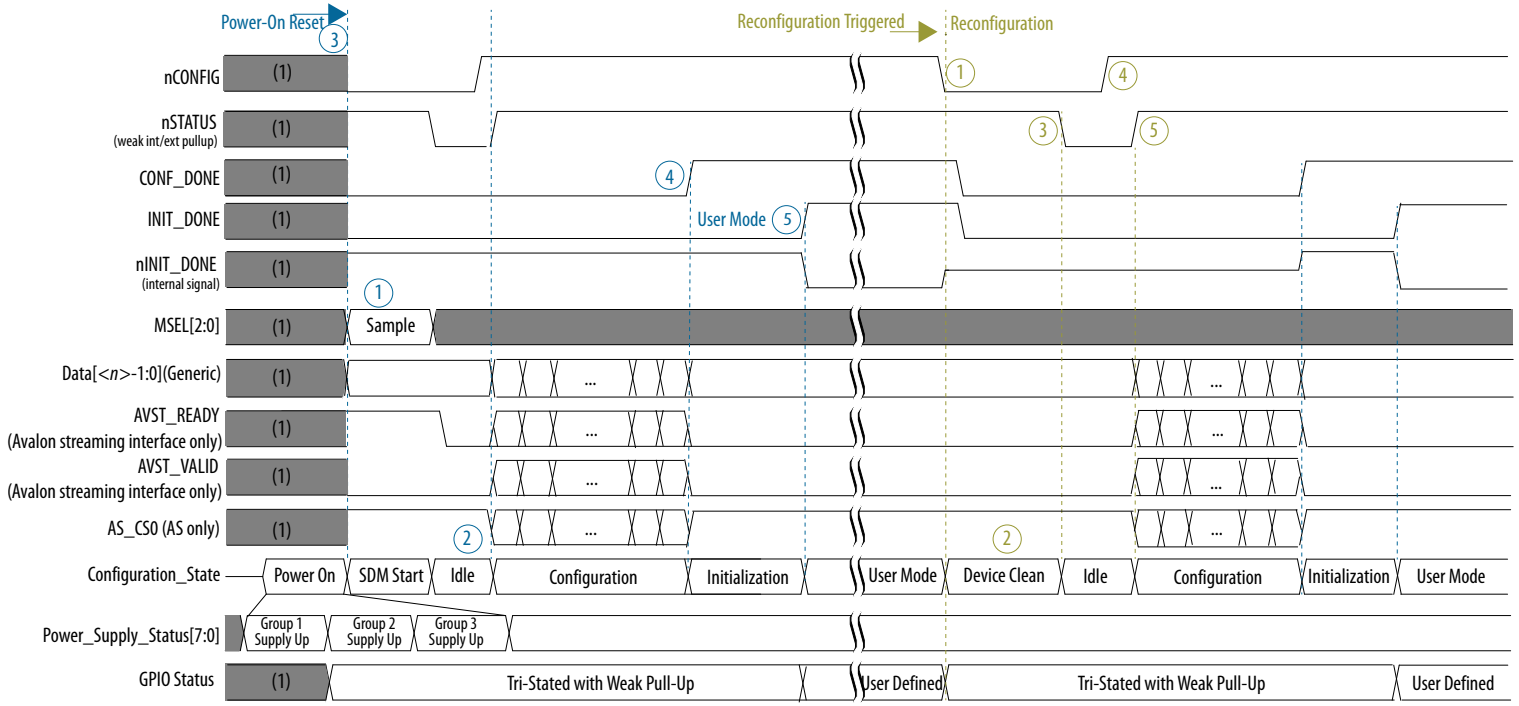
Related Information

Hard Processor System Booting User Guide: Agilex 5 SoCs

2. Agilex 5 Configuration Details

2.1. Agilex 5 Configuration Timing Diagram

Figure 4. Power-On, Configuration, and Reconfiguration Timing Diagram



Note:
(1) All I/O pins in SDM and HPS bank except the VSIGP_0, VSIGN_0, VSIGP_1, VSIGN_1, and RREF_SDM pins are in undetermined state during device power up and power down.
Input signals of an I/O pin at any point during the power up and power down should not exceed the I/O buffer power supply rail of the bank. When using pin in the GPIO bank with 1.5V VCCIO_PIO, the pin voltage must not exceed the VCCIO_PIO rail or 1.2V, whichever is lower.

Note: CONF_DONE and INIT_DONE are deasserted during device clean state after full device reconfiguration is triggered.

The SDM drives Agilex 5 device configuration.

Power-On Status

The power-on reset (POR) holds the Agilex 5 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 power supplies ramp time do not meet the t_{RAMP} time, the Agilex 5 device I/O pins state is unknown.

For more information about POR refer to the *Agilex 5 Power Management User Guide.* For more information about t_{RAMP} refer to the *Agilex 5 Device Datasheet*.

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.

Note: To receive a valid nSTATUS response from the device, your host must only monitor this signal after the device power group 3 is powered up to the recommended operating condition and after the maximum POR delay specification is met. For more information, refer to the POR delay specification in *Agilex 5 Device Data Sheet*.

When an error occurs, nSTATUS pulses low 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 configuration scheme, 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*. Throughout the configuration, 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 Agilex 5 device asserts INIT_DONE to indicate the Agilex 5 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 reset release as described in the Including the Reset Release Intel FPGA IP in Your Design on page 140. 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 in the timing diagram illustrates the Agilex 5 device reconfiguration. If you change the MSEL setting after power-on, you must power-cycle the Agilex 5. Power cycling forces the SDM to sample the MSEL pins before reconfiguring the device.

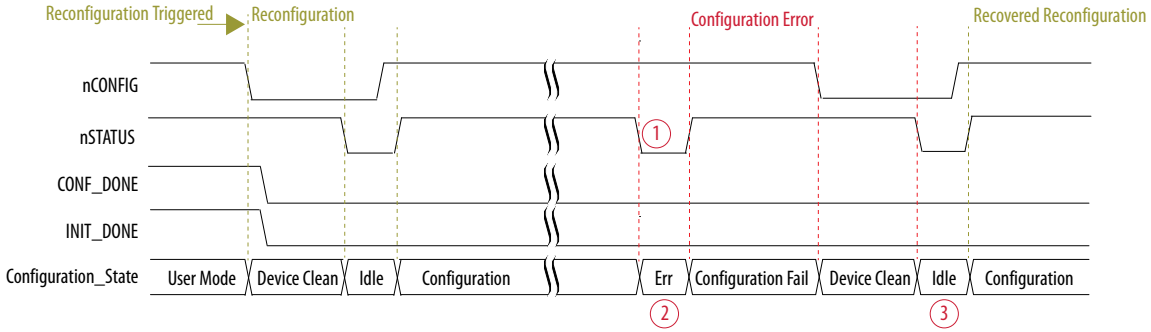
The numbers in the *Reconfiguration* part of the timing diagram mark the following events:

1. The external host drives nCONFIG signal low. **nCONFIG signal must be held low until the device drives the nSTATUS 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.

Note: If you do not monitor the nSTATUS signal and you are using device security features, pulse the nCONFIG signal low for at least 1000 ms to initiate a reconfiguration request.

Recoverable Configuration Error

Figure 5. Recoverable Error during Reconfiguration Timing Diagram



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

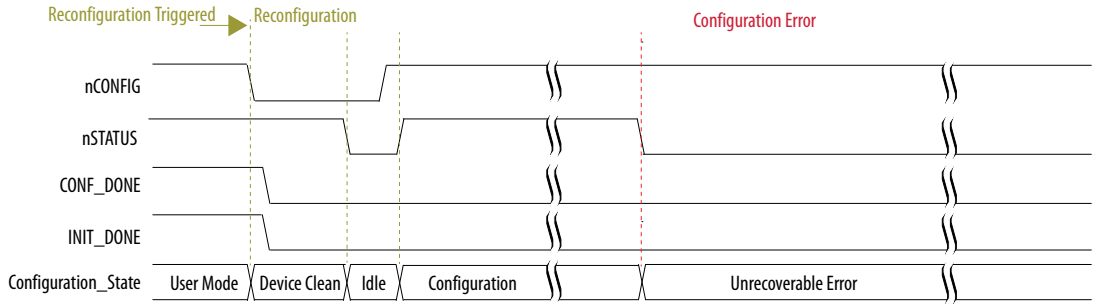
1. The SDM drives nSTATUS signal low for a period of time specified in the *Agilex 5* Device Datasheet to indicate a recoverable configuration error. The Agilex 5 device may not assert CONF_DONE indicating that device did not receive the complete configuration bitstream. The device does not assert INIT_DONE indicating that the configuration did not complete successfully. nCONFIG should continue to be driven high until after nSTATUS has returned back to high state.

   If an error occurs during JTAG configuration, the SDM does not change the state of the nSTATUS signal. You can monitor the error messages that the Quartus Prime Pro Edition Programmer generates for error reporting.
2. The SDM enters the error state.
3. The SDM enters the idle state if the nCONFIG signal drives to low. 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.

Unrecoverable Configuration Error

Figure 6. Unrecoverable Error during Reconfiguration Timing Diagram



In rare instances, a configuration error or a security event may be unrecoverable. In these cases, the SDM drives nSTATUS low and it stays low. You must perform a power cycle to restart the reconfiguration process. To ensure error recovery under all reconfiguration circumstances, Intel recommends that you design your system to continuously monitor nSTATUS and enable device power cycling if needed.

Note that in the case of a double bit ECC error in the SDM RAM, nSTATUS is also asserted low and stays low.

Bootrom recovery is a feature introduced to eliminate the need to power cycle the FPGA in the rare instance when the FPGA enters an unrecoverable error state. This feature is available in AGF 019/023, AGI 019/023, and onwards.

If you are using Avalon streaming interface x16 configuration mode, bootrom recovery requires you to follow the dual-purpose pins restrictions in Enabling Dual-Purpose Pins.

If you are using QSPI flash, you must connect the serial flash or quad SPI flash reset pin to the AS_nRST pin. The SDM must fully control the QSPI reset. Do not connect the quad SPI reset pin to any external host.

Related Information

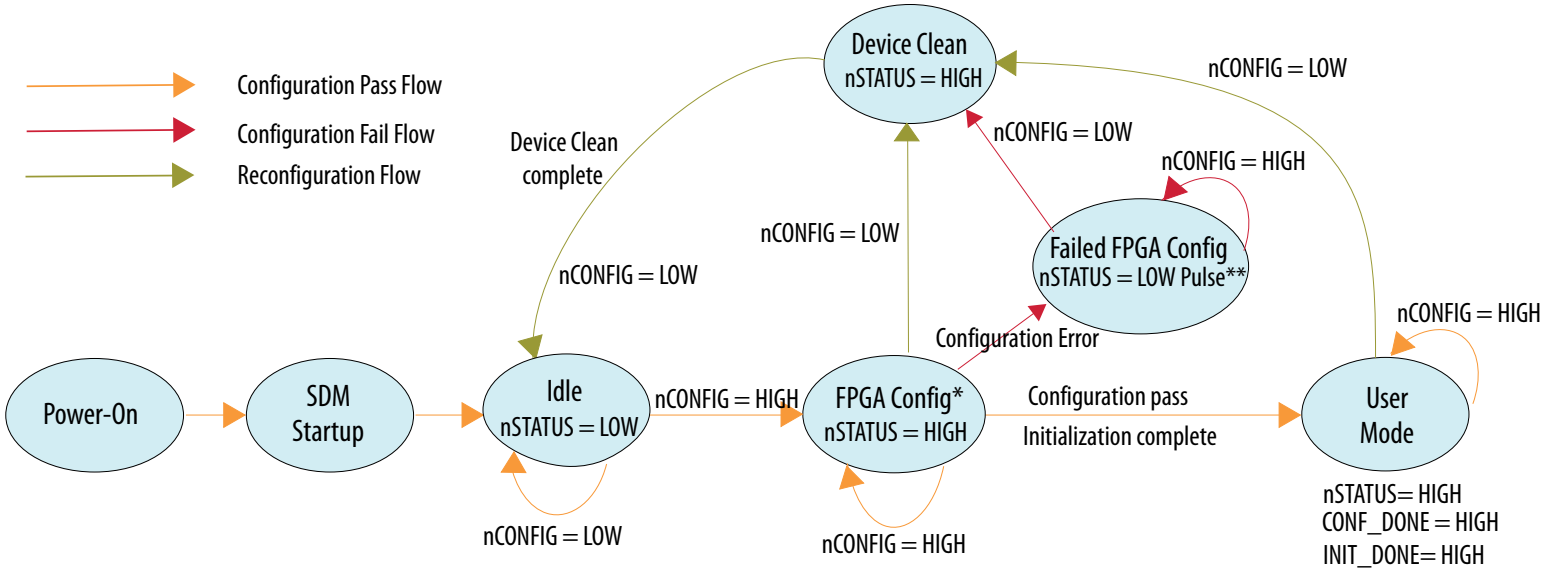
- Standard (non-RSU) Image Layout in Flash on page 172
- RSU Image Layout in Flash – SDM Perspective on page 172
- Agilex 5 FPGAs and SoCs Device Data Sheet
- Agilex 5 FPGAs and SoCs Device Overview

- Power Management User Guide: Agilex 5 FPGAs and SoCs
- Quad SPI Flash Layout on page 172
  For information about storing firmware, configuration, and application data in flash devices.
- Enabling Dual-Purpose Pins on page 41

2.2. Configuration Flow Diagram

This topic describes the configuration flow for Agilex 5 devices.

Figure 7. Agilex 5 FPGA Configuration Flow



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

** For nSTATUS low pulse duration, refer to the t_{STO} parameter in the Intel Device Datasheet.

Note: You can perform JTAG configuration anytime from any state if the device is powered up and the power is intact. The Agilex 5 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.

Power-On

- The Agilex 5 power supplies follow the guidelines in the *Power-Up Sequence Requirements for Agilex 5 Devices* section of the *Agilex 5 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 power-on stage, 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.
- All I/O pins in SDM and HPS bank except the VSIGP_0, VSIGN_0, VSIGP_1, VSIGN_1, and RREF_SDM pins are in undetermined state during device power up and power down.
- Input signals of an I/O pin at any point during the power up and power down should not exceed the I/O buffer power supply rail of the bank. When using pin in the GPIO bank with 1.5V VCCIO_PIO, the pin voltage must not exceed the VCCIO_PIO rail or 1.2V, whichever is lower.

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.

FPGA Configuration

- 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.
- In the reconfiguration flow:
  - If the power-on reset (POR) was triggered after reconfiguration, the boot ROM loads the firmware after exiting POR.
  - For reconfiguration triggered by a pulse of nCONFIG low, the SDM checks the updated firmware and compares with existing firmware.
    - If both firmware are the same, the SDM continues with configuration flow.
    - If firmwares are different, the SDM transitions to the firmware that comes together with the bitstream.
- The nCONFIG pin remains high during configuration and in user mode. The host monitors the nSTATUS pin continuously for configuration errors.
- The power management activity is ongoing during the device configuration. For more information, refer to the *Agilex 5 Power Management User Guide*.
- 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.

Failed FPGA Configuration

- A low pulse on the nSTATUS pin indicates a configuration error.
- An internal device wipe occurs followed by errors requiring 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. After CONF_DONE asserts and before INIT_DONE asserts, parts of the device start to enter user mode. The assertion of INIT_DONE indicates that the entire device entered user mode. 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 Agilex 5 device drives CONF_DONE and INIT_DONE low.
- The SDM drives the nSTATUS pin low when device cleaning completes.

Related Information

Power Management User Guide: Agilex 5 FPGAs and SoCs

2.3. Device Response to Configuration and Reset Events

The following table summarizes the device response to various external configuration and reset events.

Note: HPS_COLD_nRESET is a SDM input pin that manages the HPS reset.

Table 3. Device Response Due To Configuration and Reset Events

Events marked by a tick (√) require reset initiated by provided reset type.

Action	Reset Type		
	Power Cycle	nCONFIG	HPS_COLD_nRESET
Wipe the FPGA	√	√	—
Sample MSEL pins	√	—	—
Read fuses	√	—	—
Run the SDM boot ROM code	√	—	—
Reset the SDM	√	—	—
Reset the HPS	√	√	√

Note: When using QSPI, you can use Remote System Update (RSU) to load a specific image with the same device responses as nCONFIG.

Related Information

Power Management User Guide: Agilex 5 FPGAs and SoCs

2.4. Additional Clock Requirements for HPS and GTS Transceivers

The Agilex 5 device has specific reference clocks for HPS, HPS EMIF IP, and GTS transceivers. These clock requirements must be met before the FPGA configuration begins.

FPGA Configuration

The Agilex 5 device requires additional clocks for HPS, HPS EMIF IP and GTS transceiver. You must provide a free-running, stable reference clock to these blocks before configuration begins. The clock frequencies must match the frequency settings specified in the Quartus Prime software during configuration. These reference clocks are in addition to the configuration clock requirements for an internal or external oscillator described in OSC_CLK_1 Requirements on page 53.

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

Table 4. Additional reference clocks

Block	Clock name
HPS reference clock	HPS_OSC_CLK[3] [4]
HPS EMIF	pll_ref_clk

Note: The GTS transceiver power supplies VCCEHT_GTS and VCCERT_GTS are required for successful configuration.

Note: The free running and stable reference clock to GTS transceiver has a dependency on the IP settings. Refer to the *Guidelines for GTS System PLL Clocks Intel FPGA IP Usage* section in the *Agilex 5 FPGA GTS Transceiver Architecture and PMA and FEC Direct PHY IP User Guide* for more details.

Quartus Prime Pro Edition software allows you to configure the HPS prior to FPGA configuration. To enable this option, select **HPS First** in the **Assignments ➤ Device ➤ Device and Pin Options ➤ Configuration ➤ HPS/FPGA Configuration order** dialog box.

HPS First Configuration

Agilex 5 devices have the option of booting the HPS before configuring the FPGA core logic. This method is known as the HPS first configuration. When you choose this option in the Quartus Prime Pro Edition software, the following clocks must be operational prior to the FPGA I/O, HPS I/O, and HPS boot, also called a phase 1 configuration.

Table 5. Verify Clocks are Operational before Phase 1 Configuration

Block	Clock name
HPS reference clock	HPS_OSC_CLK
HPS EMIF	pll_ref_clk

The remaining clocks specified in the FPGA Configuration on page 27 must be fully operational prior the FPGA core logic configuration, also called phase 2 configuration.

(3) Only available when HPS is enabled.

(4) If you use the FPGA to HPS free clock as the HPS PLL reference clock, the HPS_OSC_CLK clock may not be required.

Related Information

Agilex 5 FPGA GTS Transceiver Architecture and PMA and FEC Direct PHY IP User Guide User Guide

2.5. Agilex 5 Configuration Pins

The Agilex 5 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 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.

Related Information

Pin Connection Guidelines: Agilex 5 FPGAs and SoCs

2.5.1. 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 **Assignments ➤ Device ➤ Device and Pin Options** dialog box in the Quartus Prime software.

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 6. 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_nCSO0
SDM_IO6	—	AVSTx8_DATA4	AS_DATA3
SDM_IO7	MSEL1	—	AS_nCSO2
SDM_IO8	—	AVSTx8_READY	AS_nCSO3
SDM_IO9	MSEL2	—	AS_nCSO1
SDM_IO10	—	AVSTx8_DATA7	—
SDM_IO11	—	AVSTx8_VALID	—
SDM_IO12	—	—	—
SDM_IO13	—	AVSTx8_DATA5	—
SDM_IO14	—	AVSTx8_CLK	—
SDM_IO15	—	AVSTx8_DATA6	AS_nRST
SDM_IO16	—	—	—

Related Information

Pin Connection Guidelines: Agilex 5 FPGAs and SoCs

2.5.2. MSEL Settings

After power-on MSEL[2:0] pins specify the configuration scheme for Agilex 5 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 8. MSEL Pull-Up and Pull-Down Circuit Diagram

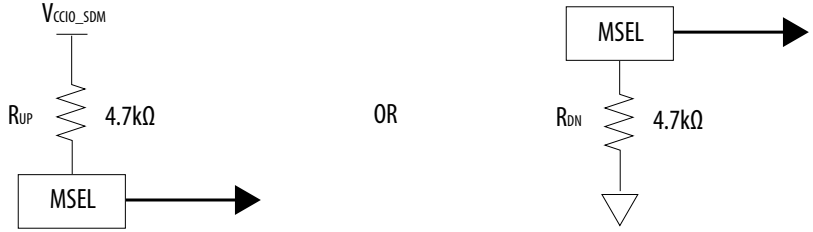


Table 7. MSEL Settings for Each Configuration Scheme of Agilex 5 Devices

Configuration Scheme	MSEL[2:0]
Avalon-ST (x16)	101
Avalon-ST (x8)	110
AS (Fast mode – for CvP)(5) device begins to access it.	001
AS (Normal mode)(6).	011
JTAG only(7)	111

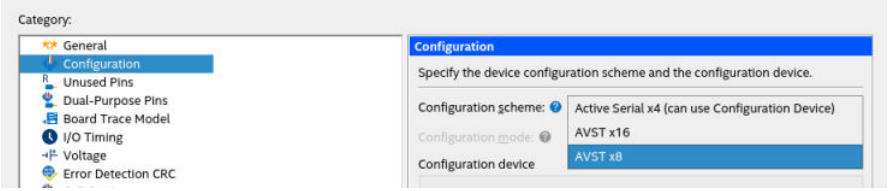
You must also specify the configuration scheme on the **Configuration** page of the **Device and Pin Options** dialog box in the Quartus Prime Software.

(5) If you use AS Fast mode, you must ramp all power supplies to the recommended operating condition within 10 ms. This ramp up requirement ensures that the AS x4 device is within its operating voltage range when the Agilex 5

(6) If you use AS Normal mode, you must fully ramp the V_{CCIO_SDM} supply to the recommended operating condition within 10 ms

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

Figure 9. Specify Configuration Scheme to Specify MSEL Value



2.5.3. Device Configuration Pins for Optional Configuration Signals

All configuration schemes use the same dedicated pins for the standard control signals shown in the *Agilex 5 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.

Table 8. 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	
PWRMGT_SCL	SDM_IO0	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
PWRMGT_ALERT	SDM_IO0 SDM_IO9 SDM_IO12	SDM_IO0 SDM_IO9 SDM_IO12	SDM_IO0 SDM_IO12

continued...

Signal Names	Configuration Scheme		
	Avalon-ST		AS x4
	x8	x16	
CONF_DONE	SDM_IO0 SDM_IO5 SDM_IO12 SDM_IO16	SDM_IO0 SDM_IO1 SDM_IO2 SDM_IO3 SDM_IO4 SDM_IO6 SDM_IO7 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_IO16
INIT_DONE	SDM_IO0 SDM_IO5 SDM_IO12 SDM_IO16	SDM_IO0 SDM_IO1 SDM_IO2 SDM_IO3 SDM_IO4 SDM_IO6 SDM_IO7 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_IO16
CVP_CONFDONE	SDM_IO0 SDM_IO5 SDM_IO7	Not supported	SDM_IO0 SDM_IO10 SDM_IO11

continued...

Signal Names	Configuration Scheme		
	Avalon-ST		AS x4
	x8	x16	
	SDM_IO9 SDM_IO12 SDM_IO16		SDM_IO12 SDM_IO13 SDM_IO14 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_IO13 SDM_IO14 SDM_IO15 SDM_IO16	SDM_IO0 SDM_IO10 SDM_IO11 SDM_IO12 SDM_IO13 SDM_IO14 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_IO0 SDM_IO10 SDM_IO11 SDM_IO12 SDM_IO13 SDM_IO14 SDM_IO16

continued...

Signal Names	Configuration Scheme		
	Avalon-ST		AS x4
	x8	x16	
		SDM_IO12 SDM_IO13 SDM_IO14 SDM_IO15 SDM_IO16	
Direct to Factory Image	Not applicable	Not applicable	SDM_IO0 SDM_IO10 SDM_IO11 SDM_IO12 SDM_IO13 SDM_IO14 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_IO10 SDM_IO11 SDM_IO12 SDM_IO13 SDM_IO14 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.

Related Information

- Configuration via Protocol (CvP) Implementation User Guide: Agilex 5 FPGAs and SoCs
- Pin Connection Guidelines: Agilex 5 FPGAs and SoCs
- Power Management User Guide: Agilex 5 FPGAs and SoCs

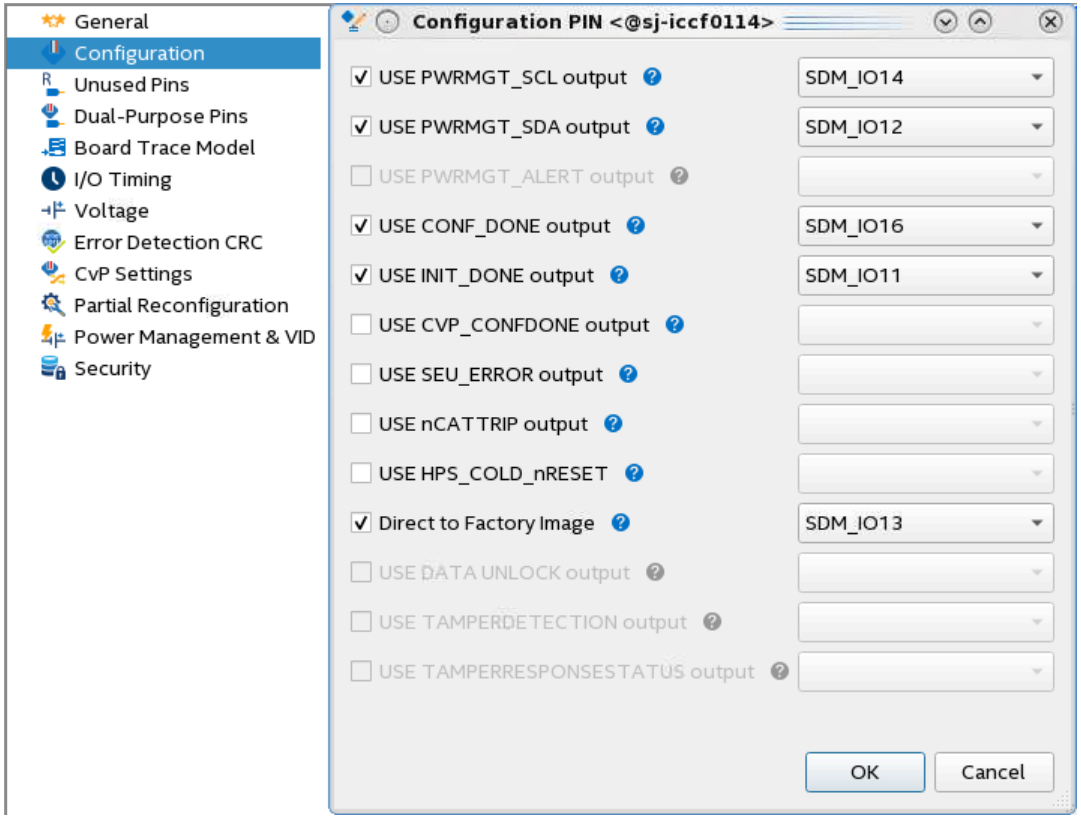
2.5.3.1. Specifying Optional Configuration Pins

You enable and assign the SDM I/O pins using the 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.

Figure 10. Specifying Optional Configuration Pins



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

2.5.3.1.1. 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 Agilex 5 device is ready it drives nSTATUS to follow nCONFIG.

The host should drive nCONFIG low to initiate device cleaning. Then the host should drive nCONFIG high to 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.

2.5.3.1.2. 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 Agilex 5 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 Agilex 5 device drives nSTATUS high.

In previous device families the deassertion of nSTATUS indicates the device is ready for configuration. For Agilex 5 devices, when using Avalon-ST configuration scheme, after the Agilex 5 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. Agilex 5 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. Agilex 5 always drives nSTATUS. Consequently, you cannot wire OR an Agilex 5 nSTATUS signal with the nSTATUS signal from earlier device families.

nSTATUS must be pulled high externally and the SDM must sample nSTATUS high when V_{CCIO_SDM} ramps up to the recommended operating voltage.

2.5.3.1.3. CONF_DONE and INIT_DONE

For Agilex 5 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. The CONF_DONE signal indicates that the configuration bitstream is received successfully. The INIT_DONE pin indicates that the device operates within the design.

In the current implementation, you cannot wire an Agilex 5 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 Quartus Prime software or in the 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 follow the Including the Reset Release Intel FPGA IP in Your Design on page 140 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 Quartus Prime Pro Edition **Device and Pin Options** menu defines.

2.5.3.1.4. SDM_IO Pins

Agilex 5 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 Agilex 5 circuitry pulls SDM_IO0, SDM_IO8 and SDM_IO16 weakly low through a 20 kΩ resistor. Internal Agilex 5 circuitry pulls all other SDM_IO pins weakly high during power-on.

Figure 11. Configuration Pin Selection in the Quartus Prime Pro Edition Software

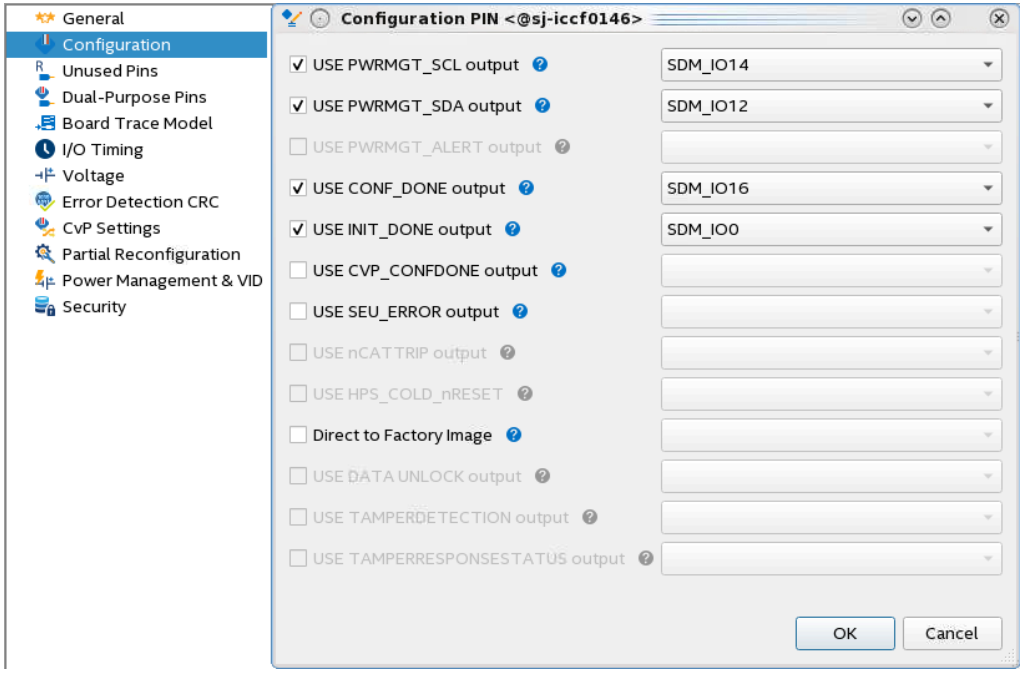
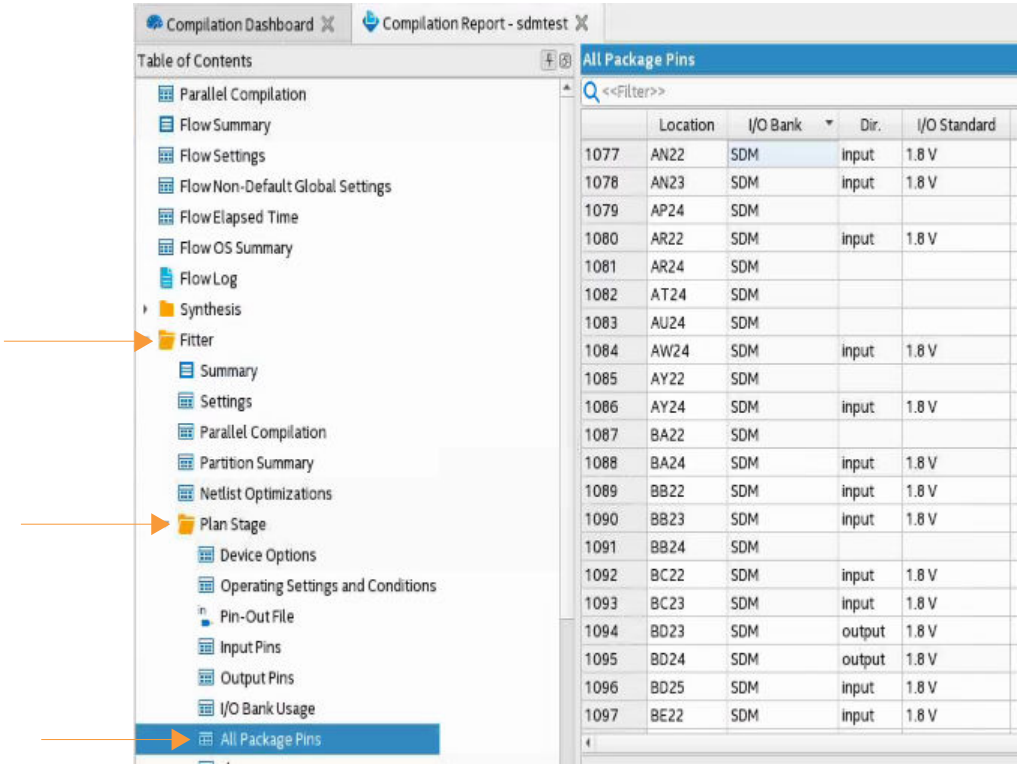


Figure 12. Fitter Report and SDM_IO Pin Reporting



2.5.3.2. Enabling Dual-Purpose Pins

AVST_CLK, AVST_DATA[15:0], 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.

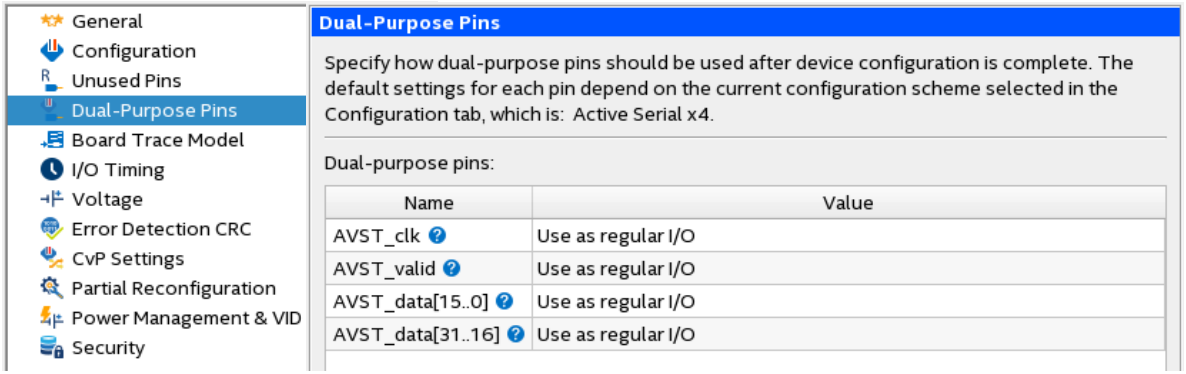
If you use these pins as GPIOs, make the following assignments:

- 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.

Figure 13. Enabling Dual-Purpose Pins



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).
- AVSTx16 configuration scheme cannot be used in designs that include the HPS. HPS-EMIF signals and AVSTx16 signals are both located in the same bank, therefore, they cannot be used simultaneously. The AVSTx8 mode uses dedicated SDM I/O pins, therefore can be used in designs that include the HPS.
- In designs that do not include the HPS, you can use dual purpose AVST pins after entering user mode, with the restrictions listed in the following table.

Table 9. Dual-Purpose Pin Restrictions for Avalon Streaming x16 Configuration Scheme

Dual-Purpose Pin	Avalon Streaming x16	
	Not Used in User Mode	Used in User Mode
AVST_CLK	Setting: As input tri-stated	Setting: Set as regular I/O Pin Connection: Set as Input and assign ALL pins in pin assignment
AVST_VALID		
AVST_DATA[15:0]		

Note:
- All pins in the same group name must be assigned to the physical pin in pin assignment. For instance, if only 2 out of 16 pins from AVST_DATA[15:0] are used, then all 16 pins must be assigned to physical pins including the unused pins in the user design.
- All pins with pin assignments must be in known state, whether weak pull-up or weak pull-down.

2.5.3.3. I/O Standards and Features for Configuration Pins

The SDM pins have different I/O standards and features in different configuration schemes. You can assign the unused SDM pins for other functions in the Quartus Prime software.

Table 10. Agilex 5 AS ×4 Configuration Scheme—Dedicated Configuration Pins

Pin Function	SDM I/O	Direction	I/O Standard	Schmitt Trigger/TTL Input	Weak Pull-Up/ Pull-Down	Drive Strength	Open Drain	Slew Rate
AS_DATA1	SDM_IO1	Bidirectional	1.8 V LVCMOS	Schmitt Trigger	Disable	8 mA	Disable	Fast
AS_CLK	SDM_IO2	Output	1.8 V LVCMOS	—	—	8 mA	Disable	Fast
AS_DATA2	SDM_IO3	Bidirectional	1.8 V LVCMOS	Schmitt Trigger	Disable	8 mA	Disable	Fast
AS_DATA0	SDM_IO4	Bidirectional	1.8 V LVCMOS	Schmitt Trigger	Disable	8 mA	Disable	Fast
AS_nCSO0	SDM_IO5	Output	1.8 V LVCMOS	—	—	8 mA	Disable	Fast
AS_DATA3	SDM_IO6	Bidirectional	1.8 V LVCMOS	Schmitt Trigger	Disable	8 mA	Disable	Fast
AS_nCSO2	SDM_IO7	Output	1.8 V LVCMOS	—	—	8 mA	Disable	Fast

continued...

Pin Function	SDM I/O	Direction	I/O Standard	Schmitt Trigger/TTL Input	Weak Pull-Up/ Pull-Down	Drive Strength	Open Drain	Slew Rate
AS_nCSO3	SDM_IO8	Output	1.8 V LVCMOS	—	—	8 mA	Disable	Fast
AS_nCSO1	SDM_IO9	Output	1.8 V LVCMOS	—	—	8 mA	Disable	Fast
AS_nRST	SDM_IO15	Output	1.8 V LVCMOS	—	—	8 mA	Disable	Fast

Table 11. Agilex 5 AS ×4 Configuration Scheme—Unused Configuration Pins

For the unused configuration pins, the drive strength, open drain, and slew rate settings are not applicable.

SDM I/O	Direction	I/O Standard	Schmitt Trigger/TTL Input	Weak Pull-Up/Pull-Down
SDM_IO0	Input	1.8 V LVCMOS	Schmitt Trigger	Weak pull-down with 20 kΩ resistor
SDM_IO10	Input	1.8 V LVCMOS	Schmitt Trigger	Weak pull-up with 20 kΩ resistor
SDM_IO11	Input	1.8 V LVCMOS	Schmitt Trigger	Weak pull-up with 20 kΩ resistor
SDM_IO12	Input	1.8 V LVCMOS	Schmitt Trigger	Weak pull-up with 20 kΩ resistor
SDM_IO13	Input	1.8 V LVCMOS	Schmitt Trigger	Weak pull-up with 20 kΩ resistor
SDM_IO14	Input	1.8 V LVCMOS	Schmitt Trigger	Weak pull-up with 20 kΩ resistor
SDM_IO16	Input	1.8 V LVCMOS	Schmitt Trigger	Weak pull-down with 20 kΩ resistor

Table 12. Agilex 5 Avalon Streaming Interface ×8 Configuration Scheme—Dedicated Configuration Pins

Pin Function	SDM I/O	Direction	I/O Standard	Schmitt Trigger/TTL Input	Weak Pull-Up/ Pull-Down	Drive Strength	Open Drain	Slew Rate
AVSTx8_DATA2	SDM_IO1	Input	1.8 V LVCMOS	Schmitt Trigger	Disable	—	—	—
AVSTx8_DATA0	SDM_IO2	Input	1.8 V LVCMOS	Schmitt Trigger	Disable	—	—	—
AVSTx8_DATA3	SDM_IO3	Input	1.8 V LVCMOS	Schmitt Trigger	Disable	—	—	—
AVSTx8_DATA1	SDM_IO4	Input	1.8 V LVCMOS	Schmitt Trigger	Disable	—	—	—
AVSTx8_DATA4	SDM_IO6	Input	1.8 V LVCMOS	Schmitt Trigger	Disable	—	—	—
AVSTx8_READY	SDM_IO8	Output	1.8 V LVCMOS	—	—	8 mA	Disable	Fast

continued...

Pin Function	SDM I/O	Direction	I/O Standard	Schmitt Trigger/TTL Input	Weak Pull-Up/ Pull-Down	Drive Strength	Open Drain	Slew Rate
AVSTx8_DATA7	SDM_IO10	Input	1.8 V LVCMOS	Schmitt Trigger	Disable	—	—	—
AVSTx8_VALID	SDM_IO11	Input	1.8 V LVCMOS	Schmitt Trigger	Weak pull-down with 20 kΩ resistor	—	—	—
AVSTx8_DATA5	SDM_IO13	Input	1.8 V LVCMOS	Schmitt Trigger	Disable	—	—	—
AVSTx8_CLK	SDM_IO14	Input	1.8 V LVCMOS	Schmitt Trigger	Disable	—	—	—
AVSTx8_DATA6	SDM_IO15	Input	1.8 V LVCMOS	Schmitt Trigger	Disable	—	—	—

Table 13. Agilex 5 Avalon Streaming Interface ×8 Configuration Scheme—Unused Configuration Pins

For the unused configuration pins, the drive strength, open drain, and slew rate settings are not applicable.

SDM I/O	Direction	I/O Standard	Schmitt Trigger/TTL Input	Weak Pull-Up/Pull-Down
SDM_IO0	Input	1.8 V LVCMOS	Schmitt Trigger	Weak pull-down with 20 kΩ resistor
SDM_IO5	Input	1.8 V LVCMOS	Schmitt Trigger	Weak pull-up with 20 kΩ resistor
SDM_IO7	Input	1.8 V LVCMOS	Schmitt Trigger	Weak pull-up with 20 kΩ resistor
SDM_IO9	Input	1.8 V LVCMOS	Schmitt Trigger	Weak pull-up with 20 kΩ resistor
SDM_IO12	Input	1.8 V LVCMOS	Schmitt Trigger	Weak pull-up with 20 kΩ resistor
SDM_IO16	Input	1.8 V LVCMOS	Schmitt Trigger	Weak pull-down with 20 kΩ resistor

Table 14. Agilex 5 Avalon Streaming Interface ×16 Configuration Scheme—Dedicated Configuration Pins

For all pin functions in this table:

- The I/O location is the SDM shared GPIO bank.
- The weak pull-up or pull-down, and open drain options are not applicable.

Pin Function	Direction	I/O Standard	Drive Strength	Slew Rate
AVST_CLK	Input	1.2 V LVCMOS	—	—
AVST_READY	Output	1.2 V LVCMOS	Series 34 Ω OCT without calibration	Slow
AVST_VALID	Input	1.2 V LVCMOS	—	—
AVST_DATA	Input	1.2 V LVCMOS	—	—

Table 15. Agilex 5 Optional Configuration Pins

The SDM I/O for each pin function in this table is as assigned in the Quartus Prime configuration pins option.

Pin Function	Direction	I/O Standard	Schmitt Trigger/ TTL Input	Weak Pull-Up/ Pull-Down	Drive Strength	Open Drain	Slew Rate
PWRMGT_SCL	Bidirectional	1.8V LVCMOS	Schmitt Trigger	Weak pull-up with 20 kΩ resistor	2 mA	Enable	Slow
PWRMGT_SDA	Bidirectional	1.8V LVCMOS	Schmitt Trigger	Weak pull-up with 20 kΩ resistor	2 mA	Enable	Slow
PWRMGT_ALERT	Output	1.8V LVCMOS	—	—	2 mA	Enable	Slow
CONF_DONE	Output	1.8V LVCMOS	—	—	8 mA	Disable	Fast
INIT_DONE	Output	1.8V LVCMOS	—	—	8 mA	Disable	Fast
CvP_CONFDONE	Output	1.8V LVCMOS	—	—	8 mA	Disable	Fast
SEU_ERROR	Output	1.8V LVCMOS	—	—	8 mA	Disable	Fast
HPS_COLD_nRESET	Bidirectional	1.8V LVCMOS	Schmitt Trigger	Weak pull-up with 20 kΩ resistor	2 mA	Enable	Fast
Direct to factory image	Input	1.8V LVCMOS	Schmitt Trigger	Weak pull-down with 20 kΩ resistor	—	—	—

continued...

Pin Function	Direction	I/O Standard	Schmitt Trigger/ TTL Input	Weak Pull-Up/ Pull-Down	Drive Strength	Open Drain	Slew Rate
nCATTRIP	Output	1.8V LVCMOS	—	—	2 mA	Disable	Slow
TAMPERDETECTION	Output	1.8V LVCMOS	—	—	8 mA	Disable	Fast
TAMPERRESPONSESTATUS	Output	1.8V LVCMOS	—	—	8 mA	Disable	Fast

Related Information

Device Security User Guide: Agilex 5 FPGAs and SoCs
Device Security User Guide: Agilex 5 FPGAs and SoCs (Intel RDC item #815428)

2.5.3.3.1. IBIS Model

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

Related Information

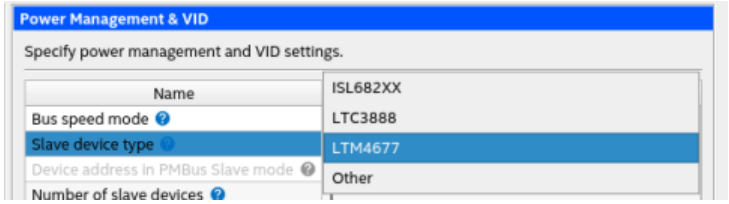
IBIS Models for Intel Devices

2.5.3.4. 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 LTC3888 or LTM4677 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. Select a regulator you use on your board. If you are using a different PMBus regulator, change the default setting from **LTM4677** to **Other**.

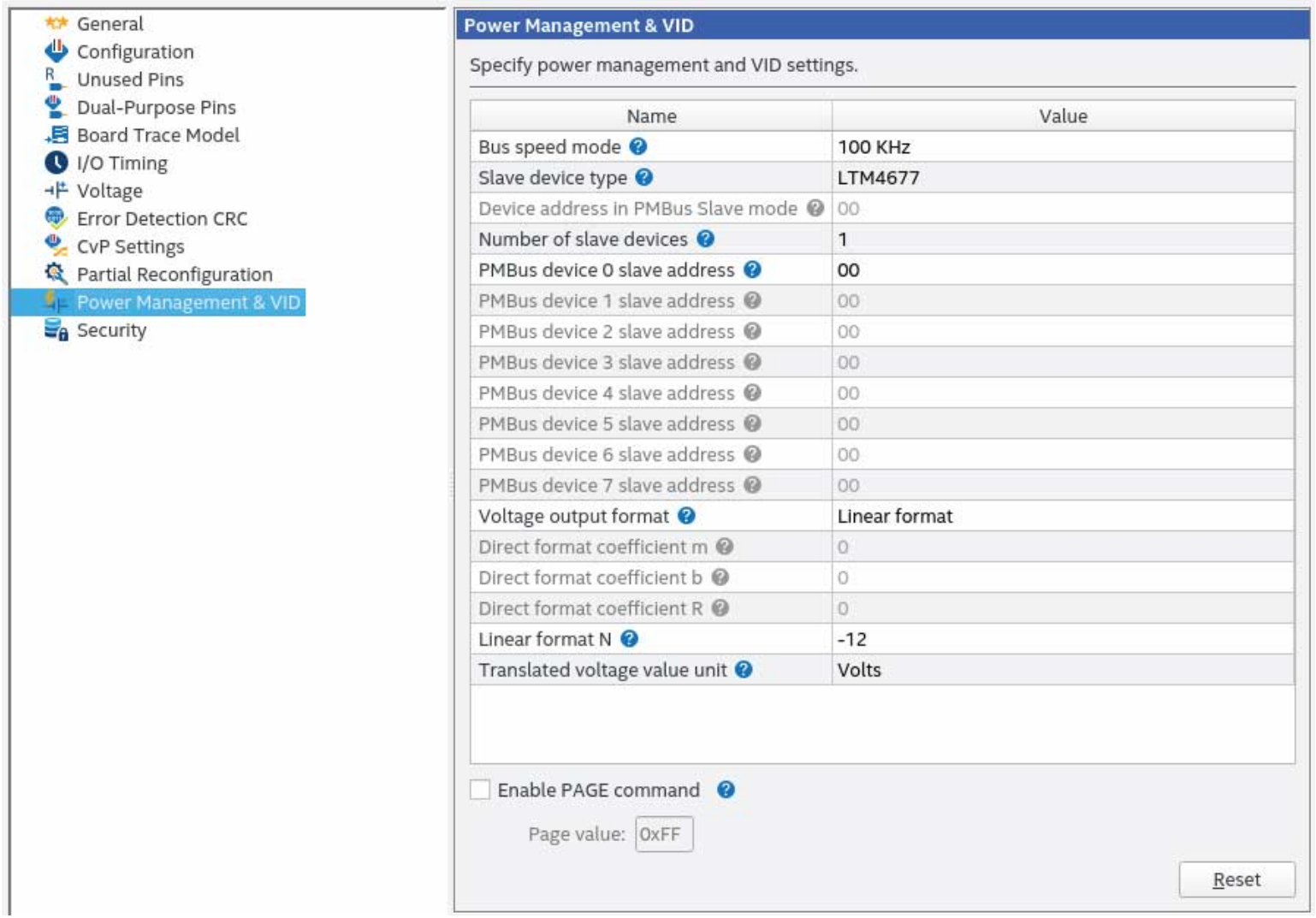
Figure 14. Specifying the Slave Device Type for Power Management and VID



Quartus Prime Pro Edition software allows you to control the payload value of the Page command. Some PMBus devices contain more than one page per bank of registers, the Page command allows you to select the target Page per bank registers. The Page payload available range is between 0x00 and 0xFF.

Note: Prior the Quartus Prime Pro Edition software release, the Page command selected all banks by default.

Figure 15. Specifying the Page Command Setting

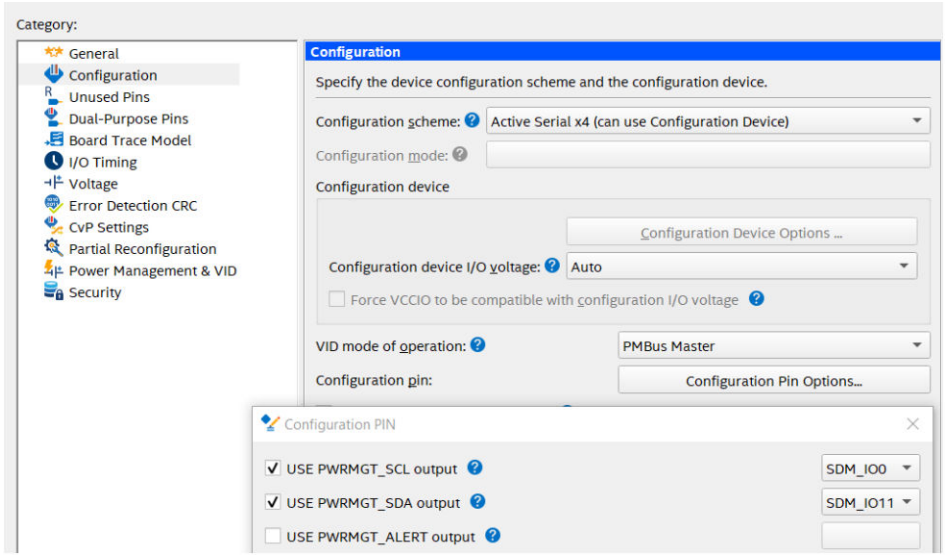


Agilex 5 SmartVID devices require SDM pins listed in the table below configured to control the VID regulator. To configure the pins, go to **Device ➤ Device and Pin Options ➤ Configuration**. Set the **VID mode of operation** accordingly to either **PMBus Master** or **PMBus Slave**, and set **Configuration pin** to **Configuration Pin Options...**.

Table 16. PMBus Modes Pins for VID Regulator

Pin	VID Mode of Operation	
	PMBus Master	PMBus Slave
PWMGT_SDA	Required	Required
PWMGT_SCL	Required	Required
PWRMGT_ALERT	Not available	Optional

Figure 16. Specifying the PMBUS Mode Pins for VID Regulator



Refer to the *SmartVID Standard Power Devices* section in the *Agilex 5 Power Management User Guide* for more information about the Power Management and VID implementation guide.

Related Information

Power Management User Guide: Agilex 5 FPGAs and SoCs

2.5.3.5. Specifying Pins for Partial Reconfiguration (PR)

The partial reconfiguration signals use GPIO pins.

The following signals control partial reconfiguration in Agilex 5 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

2.6. Configuration Clocks

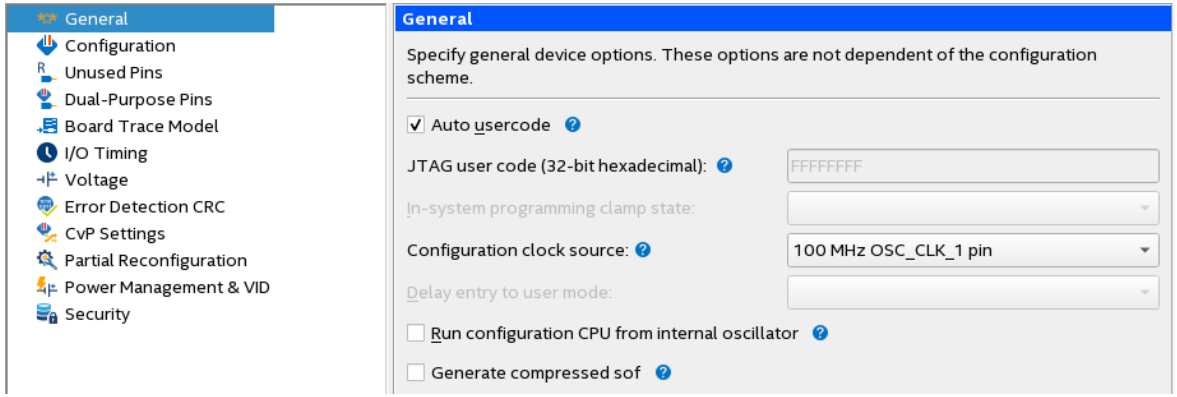
2.6.1. Setting Configuration Clock Source

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.

Figure 17. Specifying the Configuration Clock Source



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

2.6.2. 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 Quartus Prime software, the device loads the majority of the configuration bitstream at 250 MHz[8]. Agilex 5 devices include an internal oscillator in addition to OSC_CLK_1 which runs the configuration process at a frequency between 160 MHz - 230 MHz. Agilex 5 devices always use this internal oscillator to load the first section of the bitstream, up to a maximum of 512 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. If you use transceivers, EMIF, MIPI, and PHY Lite interfaces, you must provide an external clock to this pin and enable OSC_CLK_1 as the configuration clock source in the Quartus Prime software.

When you specify OSC_CLK_1 for configuration, the OSC_CLK_1 clock must be a stable and free-running clock.

When you specify AS configuration scheme and nCONFIG is pull high, the SDM starts the configuration once the device exits the POR state. Ensure the OSC_CLK_1 clock is available before SDM starts to load the bitstream from the quad SPI flash or you need to supply a stable free-running clock before/at the same time V_{CCIO_SDM} ramps up to the typical voltage level.

(8) For Agilex 5 devices with speed grade -6s and -6X devices, the clock speed for configuration network runs at 200 MHz when using OSC_CLK_1 and only supports AS_CLK at frequency of 25 MHz, 50 MHz, and 100 MHz.

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 or the OSC_CLK_1 is not stable and free running due to an interruption or a frequency change.
- 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 Agilex 5 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 Quartus Prime software.

Configuration Clock Requirements for Reconfiguration Without Power Cycling the Device

When you specify OSC_CLK_1 for configuration and reconfigure without powering down the Agilex 5 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

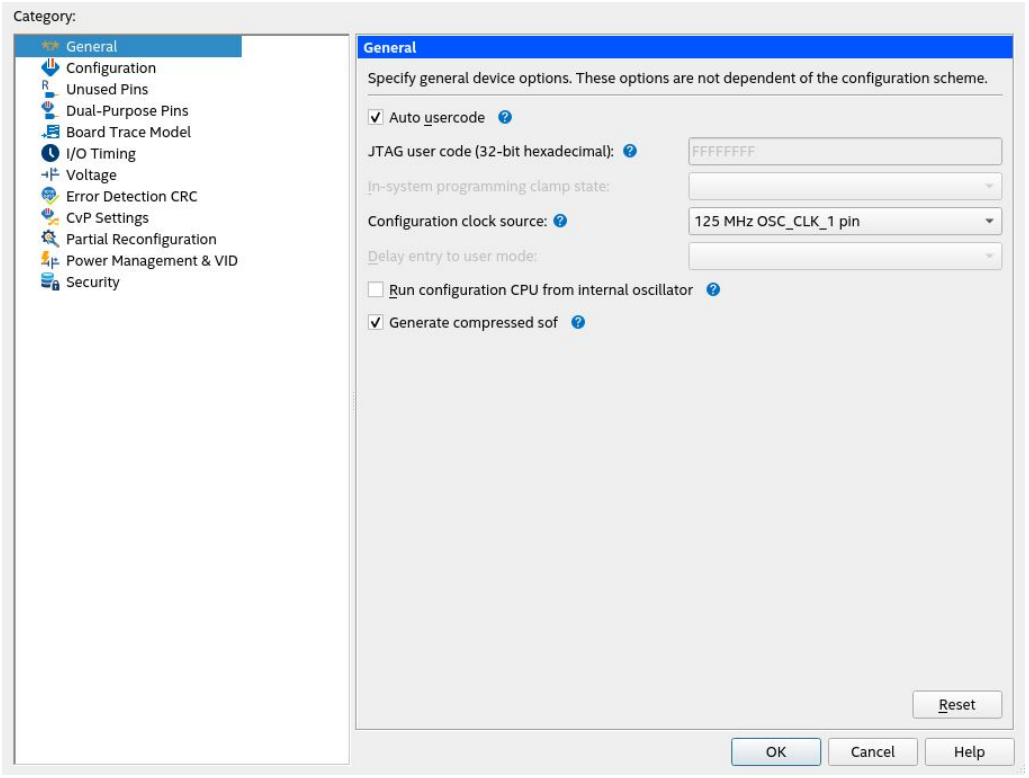
After a power-down, when you specify OSC_CLK_1 for configuration, the Agilex 5 device uses the internal oscillator to load the first section of the bitstream and OSC_CLK_1 for the remainder.

2.7. Generating Compressed .sof File

Quartus Prime Pro Edition software allows you to generate a compressed `.sof` file. The compressed `.sof` file size is smaller compared to the non-compressed `.sof` file.

To enable this option, go to **Assignments ➤ Device ➤ Device and Pin Options ➤ General** and check **Generate compressed sof** checkbox. By default, the option is disabled.

Figure 18. Compressed .sof Selection in the Quartus Prime Pro Edition Software



3. Agilex 5 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 Agilex 5 devices. This scheme uses an external host, such as a microprocessor, MAX® II, MAX V, or 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 Parallel Flash Loader II Intel FPGA IP with a MAX II, MAX V, or MAX 10 device as the host to read configuration data from the flash memory device and configure the Agilex 5 device. The Avalon-ST configuration scheme is called passive because the external host, not the Agilex 5 device, controls configuration.

Table 17. Avalon-ST Configuration Data Width, Clock Rates, and Data Rates

Mbps is an abbreviation for Megabits per second.

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

Table 18. Required Configuration Signals for the Avalon-ST Configuration Scheme

You can use an 8- or 16-bit Avalon-ST configuration data bus. You specify SDM I/O pin functions using the **Device ➤ Device and Pin Options ➤ Configuration** dialog box in the Quartus Prime software. For Avalon-ST x16 configuration, you can reassign the GPIO, dual-purpose configuration pins for other functions in user mode using the **Device ➤ 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}
MSEL[2:0]	SDM I/O	Input	V_{CCIO_SDM}

continued...

Signal Name	Pin Type	Direction	Powered by
CONF_DONE(9)	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[15:0]	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 *Agilex 5 Data Sheet* for configuration timing estimates.

The x16 mode uses 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 of some signals, to accommodate both power requirements.

Attention: Access to the I/O pins located in bank 3A with pin index[91...95] is not allowed for the AVSTx16 configuration scheme. You must leave these pins unconnected. For more information, refer to the device pin mapping files to identify the exact pin location.

Note: Although the INIT_DONE configuration signal is not required for configuration, Intel recommends that you use this signal. 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 Parallel Flash Loader II Intel FPGA IP to drive configuration, refer to the *Avalon Streaming Interfaces* in the *Avalon Interface Specifications* for protocol details.

Related Information

- Agilex 5 FPGAs and SoCs Device Data Sheet

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

- Pin Connection Guidelines: Agilex 5 FPGAs and SoCs
- Avalon Interface Specifications

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 Parallel Flash Loader II Intel FPGA 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 Quartus Prime Programmer to the PCB.

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

Figure 19. Components and Design Flow for .pof Programming

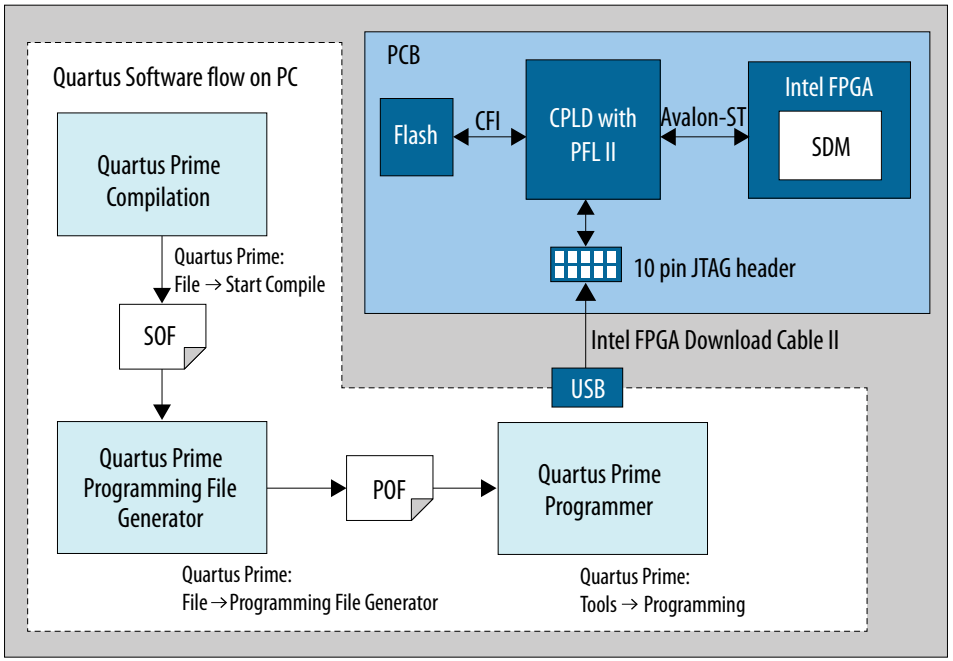


Table 19. Output File Types

Programming File Type	Extension	Description
Programmer Object File	`.pof`	The `.pof` is a proprietary Intel FPGA file type. Use the Parallel Flash Loader II Intel FPGA IP 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.

Related Information

Avalon Interface Specifications

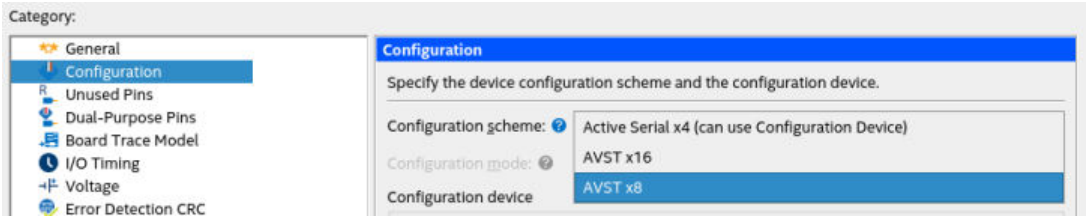
3.1.2. Enabling Avalon-ST Device Configuration

You enable the Avalon-ST device configuration scheme in the 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.

Figure 20. Selecting the Configuration Scheme Bus Width



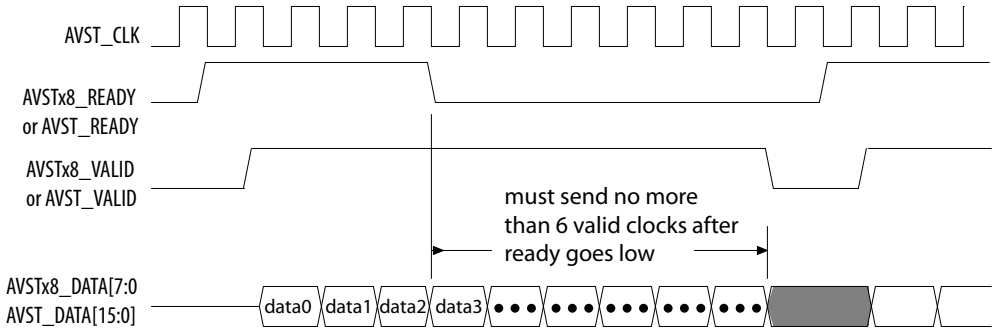
4. Click **OK** to confirm and close the **Device and Pin Options** dialog box.

3.1.3. The AVST_READY Signal

Before beginning configuration, do 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.

If you use the Parallel Flash Loader II Intel FPGA IP as the configuration host, the AVST_READY synchronizer logic is included.

Figure 21. Monitoring the AVST_READY Signal and Responding to Backpressure



The configuration files for Agilex 5 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 Agilex 5 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: To receive a valid nSTATUS response from the device, your host must only monitor this signal after the device power group 3 is powered up to the recommended operating condition and after the maximum POR delay specification is met. For more information, refer to the POR delay specification in the *Agilex 5 Device Data Sheet*.

Note: For Avalon-ST x16, 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 start sending data when AVST_READY asserts.

The AVST_READY signal sent by the Agilex 5 device to the host is not synchronized with the AVSTx8_CLK or AVST_CLK. To configure the Agilex 5 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 a 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 Parallel Flash Loader II Intel FPGA IP or your Avalon-ST controller logic.

  — `fpga_avst_ready` is the AVST_READY signal from the Agilex 5 device.

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

Note: You must properly constrain the AVST_CLK and AVST_DATA signals at the host. Perform timing analysis on both signals between the host and the Agilex 5 device to ensure the Avalon-ST configuration timing specifications are met. Refer to the *Avalon-ST Configuration Timing* section of the *Agilex 5 Device Data Sheet* for information about the timing specifications.

Note: The AVST_CLK signal must run continuously during configuration. The AVST_READY signal does not 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 Parallel Flash Loader II Intel FPGA IP as the configuration host, you can use the Quartus Prime software to store the binary configuration data to the flash memory through the Parallel Flash Loader II Intel FPGA IP.

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 configuration scheme only uses dual-purpose I/O pins that you can use as general-purpose I/O pins after configuration.

Related Information

- Agilex 5 FPGAs and SoCs Device Data Sheet
- Avalon Interface Specifications
- Avalon Streaming Configuration Timing
  For Avalon Streaming Timing Parameters for Configuration.

3.1.4. RBF Configuration File Format

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

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

Table 20. 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 22. Connections for Avalon-ST x8 Single-Device Configuration

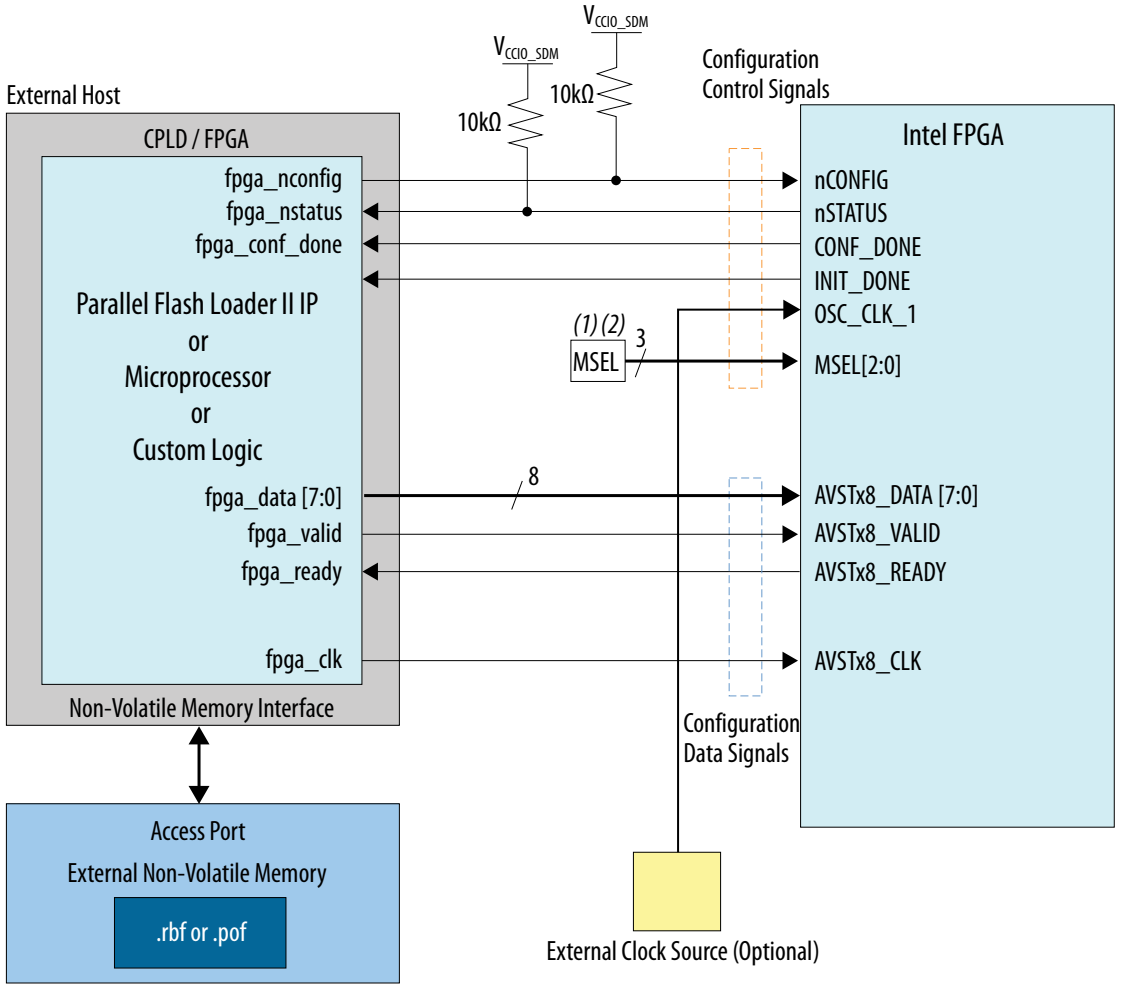
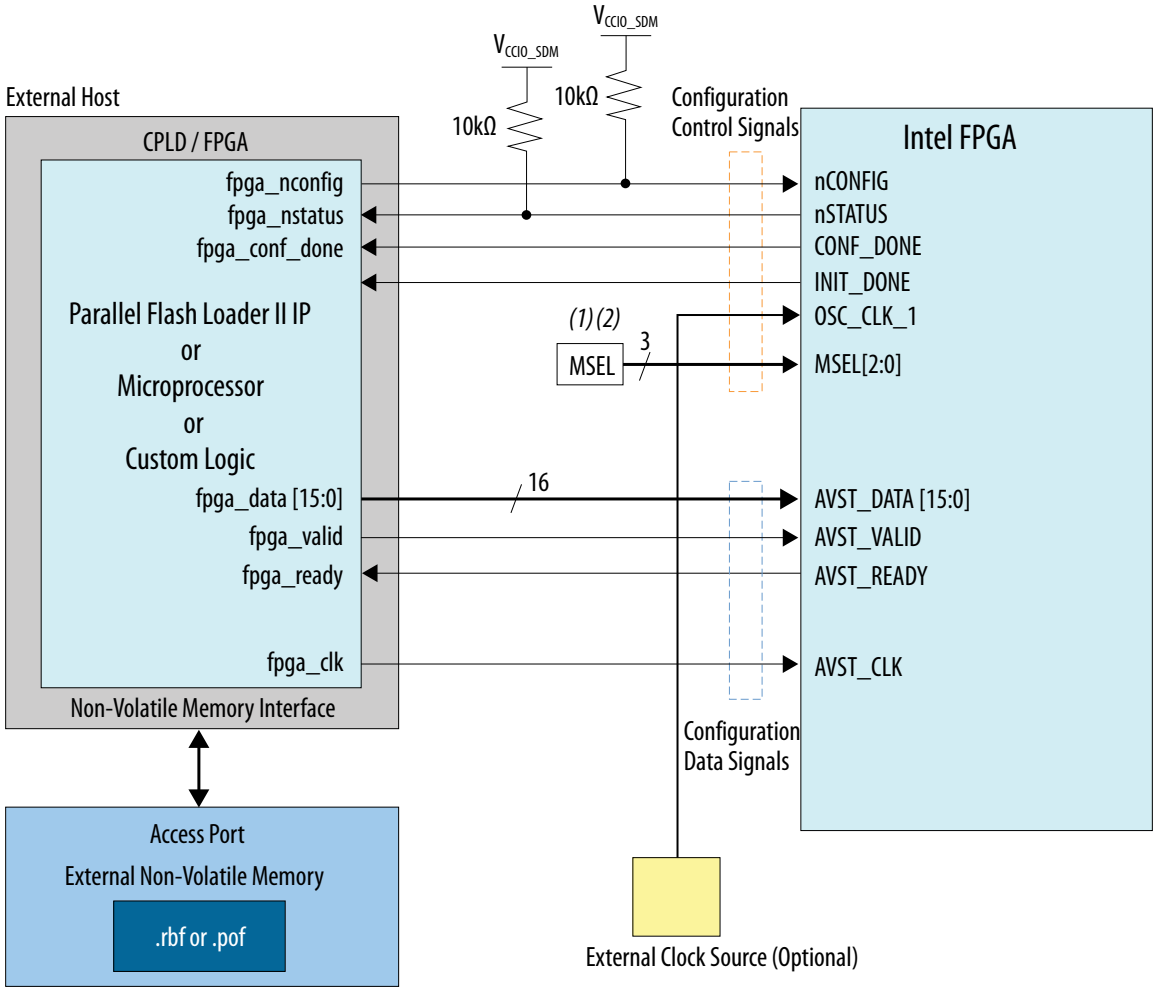


Figure 23. Connections for Avalon-ST x16 Single-Device Configuration



Notes for Figure:

1. Refer to *MSEL Settings* for the correct resistor pull-up and pull-down values for all configuration schemes.
2. The 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 31
- Pin Connection Guidelines: Agilex 5 FPGAs and SoCs

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 mode, power the I/O bank containing the x16 pins (I/O Bank 3A) at 1.2 V.
- Ensure you select the appropriate Avalon-ST configuration scheme in your Quartus Prime Pro Edition project.
- Ensure the MSEL pins reflect this mode on the PCB.

- Verify that the host device does not drive configuration pins before the Agilex 5 device powers up.
- Ensure nCONFIG remains high during the configuration process.
- Verify that CONF_DONE and INIT_DONE pins correlate to the Quartus Prime SDM I/O pins assignment and the board-level connections.
- Ensure that during the power up, no external component drives the nSTATUS signal low.

Related Information

Configuration Debugging Checklist on page 210

3.1.7. IP for Use with the Avalon-ST Configuration Scheme: Parallel Flash Loader II Intel FPGA IP (PFL II)

3.1.7.1. Functional Description

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

- Program configuration data into a flash memory device using JTAG interface.
- Configure the Agilex 5 device with the Avalon-ST configuration scheme from the flash memory device.

Note: Use the Parallel Flash Loader II Intel FPGA IP with the Avalon-ST configuration scheme in Agilex 5 devices, not the earlier Parallel Flash Loader Intel FPGA IP.

Note: The current implementation does not support programming two QSPI devices with two separate PFL images in a single programming cycle. To program multiple QSPI devices, you must program each QSPI flash device with a single PFL image separately.

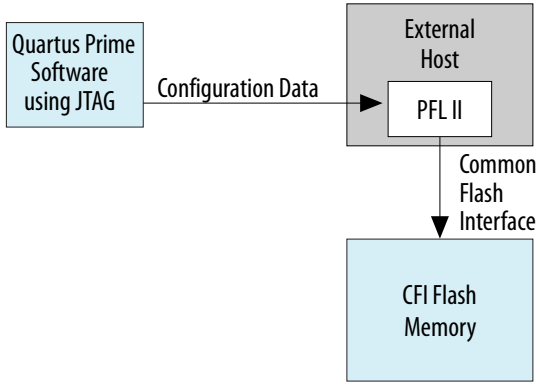
Note: The Parallel Flash Loader II Intel FPGA IP does not support HPS cold reset.

3.1.7.1.1. Generating and Programming a .pof into CFI Flash

The 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 Parallel Flash Loader II Intel FPGA IP using the Quartus Prime **File ➤ Programming File Generator**.
2. Using the Quartus Prime Programmer to write the Agilex 5 device `.pof` to the flash device.

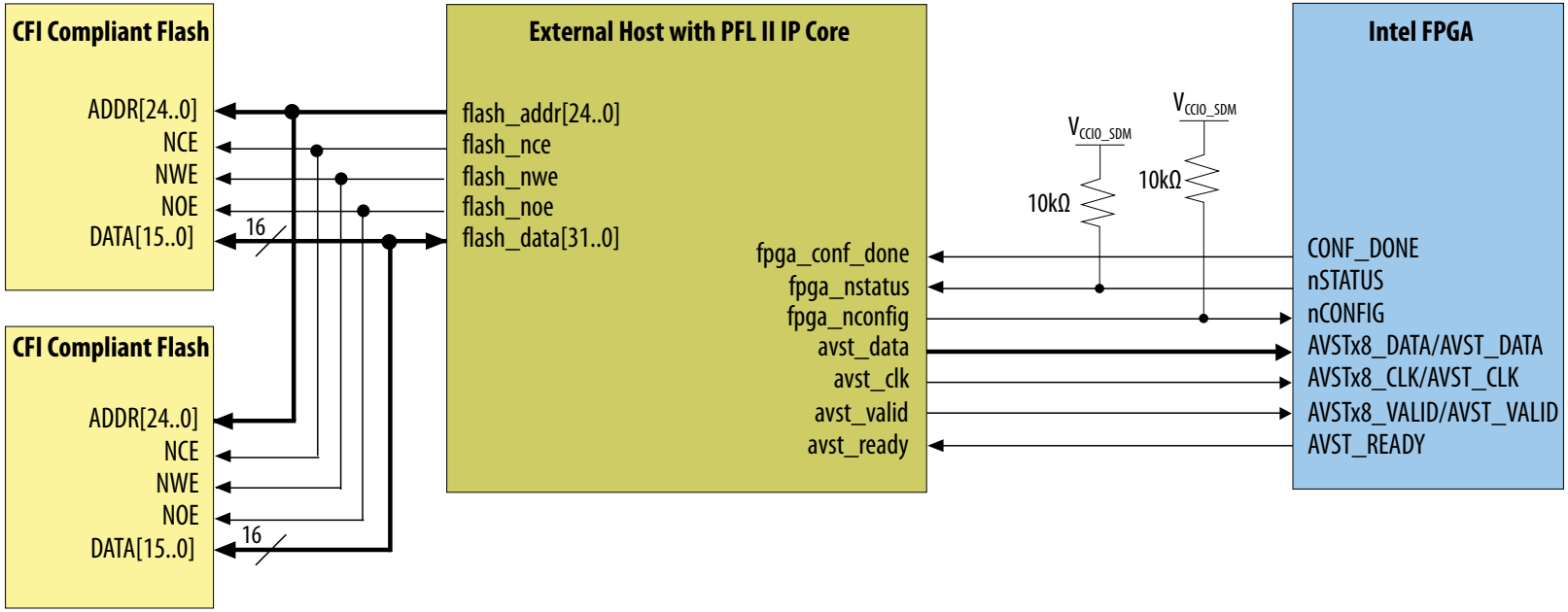
Figure 24. Programming the CFI Flash Memory with the JTAG Interface



The Parallel Flash Loader II Intel FPGA IP supports dual flash memory devices in burst read mode to achieve faster configuration times. You can connect two MT28EW CFI flash memory devices to the host in parallel using the same data bus, clock, and control signals. Intel does not support connecting two of non-MT28W CFI flash memory devices to the Parallel Flash Loader II Intel FPGA IP in parallel. During FPGA configuration, the AVST_CLK frequency is four times faster than the `flash_clk` frequency.

Figure 25. Parallel Flash Loader II Intel FPGA IP with Dual MT28EW CFI Flash Memory Devices

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



3.1.7.1.2. Implementing Multiple Pages in the Flash .pof

The Parallel Flash Loader II Intel FPGA IP 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 Quartus Prime software to automatically determine the start address of the page. The 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`.

3.1.7.1.3. Storing Option Bits

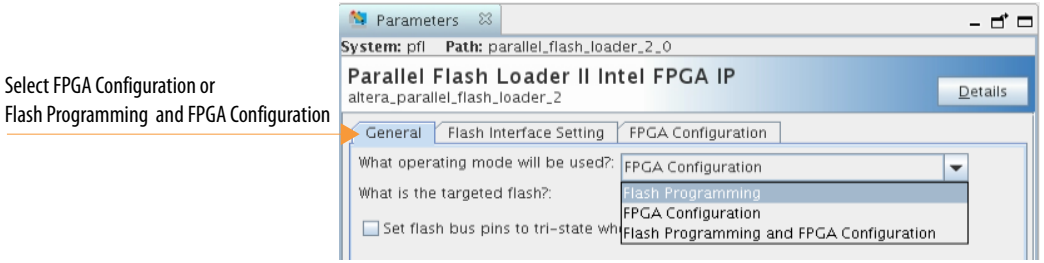
In addition to design data, the flash memory stores the option bits. You must specify the address for the options bits in two places: the Parallel Flash Loader II Intel FPGA IP and in the option bits address of the flash memory device.

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 Parallel Flash Loader II Intel FPGA IP writes this bit after successfully programming the page.

You set the option bits in the Parallel Flash Loader II Intel FPGA IP using the parameter editor. By default, the Parallel Flash Loader II Intel FPGA 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 26. General Tab of the Parallel Flash Loader II Intel FPGA 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.

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 Parallel Flash Loader II Intel FPGA IP parameters.

The 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 Agilex 5 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 21. 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

continued...

Sector Offset	Value
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[10]	.pof version
0x81-0xFF	Reserved

Caution: To prevent the Parallel Flash Loader II Intel FPGA IP 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.

Related Information

Parallel Flash Loader II Intel FPGA IP Parameters on page 78

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

3.1.7.1.4. 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 Quartus Prime software.

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

Table 22. 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 23. 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
0x0C – 0x0F	0x00352E30

For Page 0 if you append the start address bits[31:11] with 13'b0000000000000, the result is 32'b00000000000000001000000000000000 = 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.

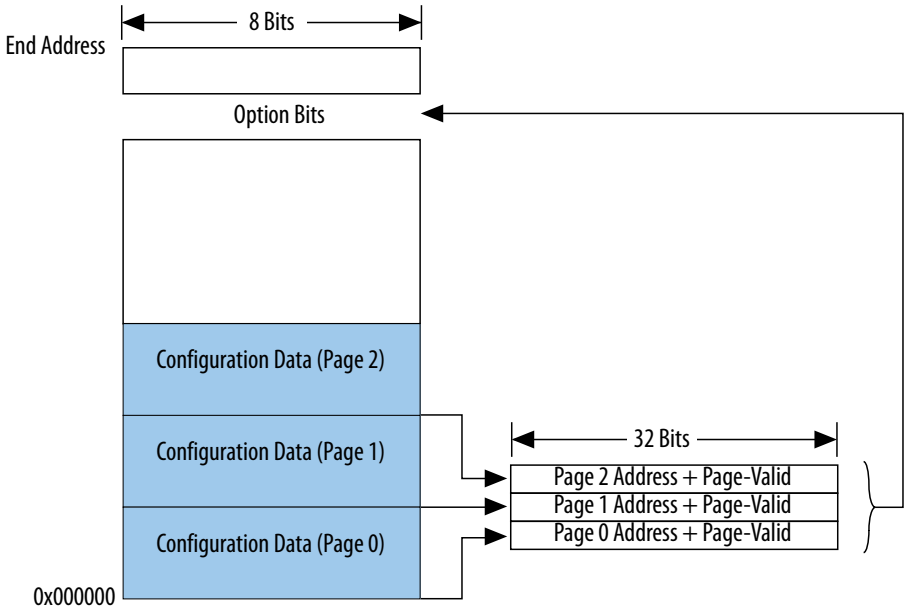
If you append end address 0x00352E30 with 2'b11, the result is 32'b00000000110101001011100011000011 = 0xD4B8C3.

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

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

The following figure shows a 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 27. 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 28. 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 Parallel Flash Loader II Intel FPGA IP sets the Page-Valid bits after successfully programming the pages.

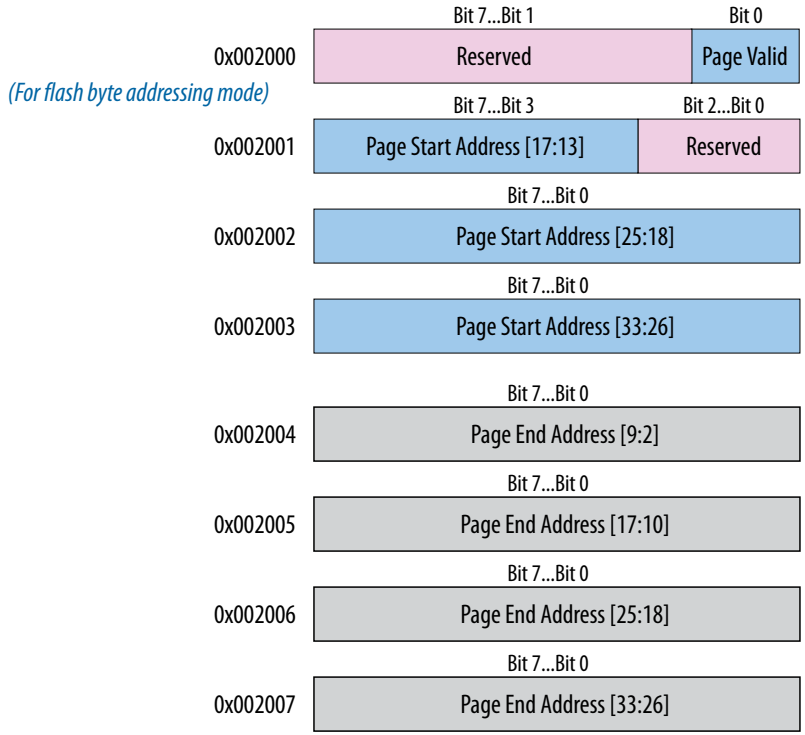


Table 24. Byte Address Range for CFI Flash Memory Devices with Different Densities

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

continued...

CFI Device (Megabit)	Address Range
64	0x0000000–0x07FFFFF
128	0x0000000–0x0FFFFFF
256	0x0000000–0x1FFFFFF
512	0x0000000–0x3FFFFFF
1024	0x0000000–0x7FFFFFF

3.1.7.2. Designing with the Parallel Flash Loader II Intel FPGA IP for Avalon-ST Single Device Configuration

This section describes the procedures on how to use the Parallel Flash Loader II Intel FPGA IP.

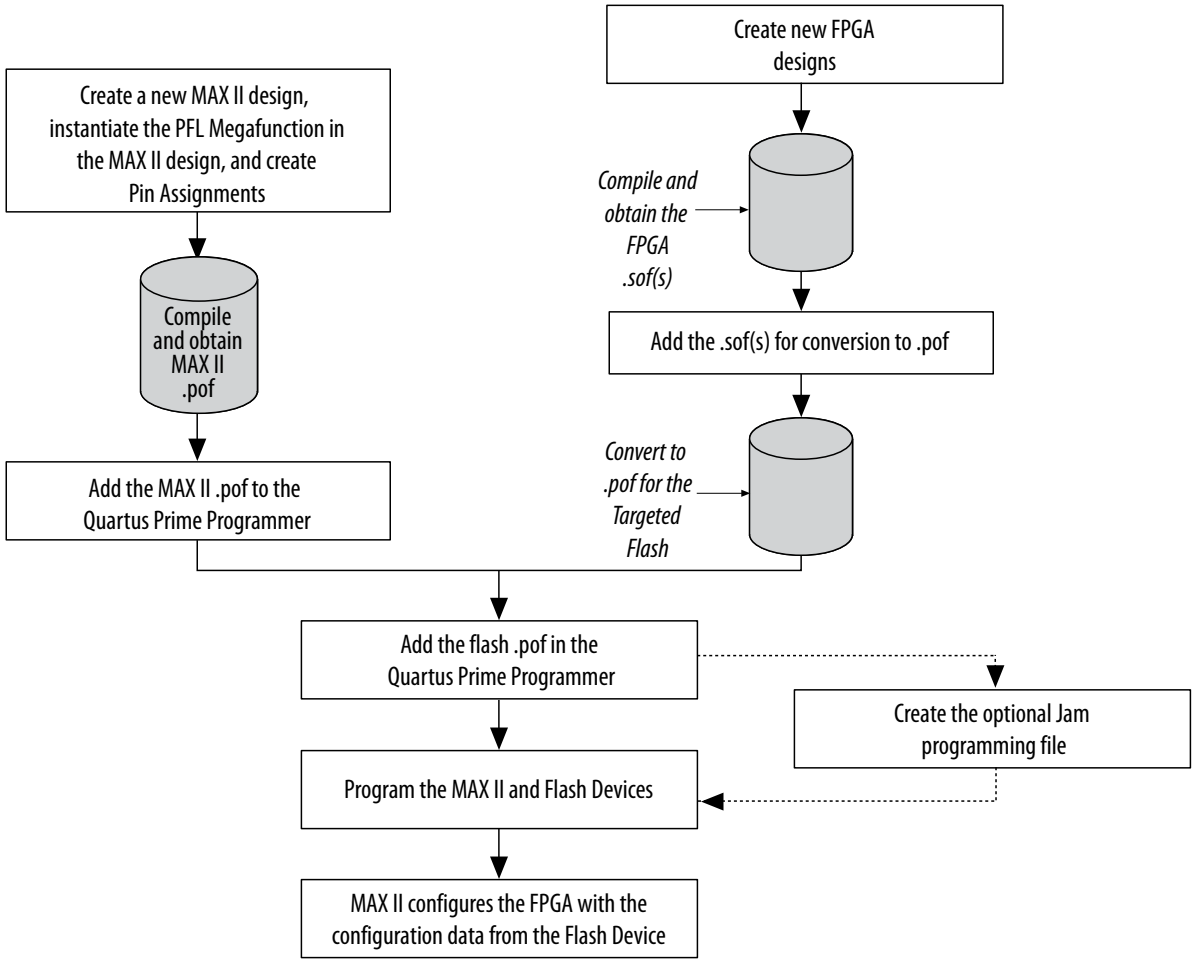
To target a MAX II, MAX V, or MAX 10 device requires the use of Quartus Prime Standard Edition software whereas targeting a Agilex 5 requires Quartus Prime Pro Edition software.

The process of creating the Avalon-ST single device configuration design targeting a MAX II, MAX V, or MAX 10 device involves three steps.

1. Generate the AVST design for the MAX device with the default option address.
2. Create the Agilex 5 `.pof` file in setting the appropriate option bits.
3. Regenerate the Parallel Flash Loader II Intel FPGA IP with the option bits used to generate the Agilex 5 `.pof` file and recompile the MAX 10 design.

Figure 29. Process for Using the Parallel Flash Loader II Intel FPGA IP

Figure shows the process for using the Parallel Flash Loader II Intel FPGA IP, using MAX II as an example.



3.1.7.2.1. Parallel Flash Loader II Intel FPGA IP Parameters

Table 25. Parallel Flash Loader II Intel FPGA IP 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 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 Parallel Flash Loader II Intel FPGA IP.
Set flash bus pins to tri-state when not in use	• **On** • **Off**	Allows the Parallel Flash Loader II Intel FPGA IP to tri-state all pins interfacing with the flash memory device when the Parallel Flash Loader II Intel FPGA IP does not require access to the flash memory.

Table 26. Parallel Flash Loader 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 Parallel Flash Loader II Intel FPGA IP.
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 Parallel Flash Loader II Intel FPGA IP, 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**(11)	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. 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 Parallel Flash Loader II Intel FPGA IP 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 th

(11) This mode is not supported in Agilex 5 FPGAs.

Table 27. Parallel Flash Loader II Flash Programming Parameters

Options	Value	Description
Flash programming IP optimization target	• **Area** • **Speed**	Specifies the flash programming IP optimization. If you optimize the Parallel Flash Loader II Intel FPGA IP for **Speed**, the flash programming time is shorter, but the IP uses more LEs. If you optimize the Parallel Flash Loader II Intel FPGA IP for **Area**, the IP 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 Parallel Flash Loader II Intel FPGA IP 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 28. Parallel Flash Loader 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 Parallel Flash Loader II Intel FPGA IP 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 Parallel Flash Loader II Intel FPGA IP 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.
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**(12)	Specifies the width of the Avalon-ST interface.

continued...

(12) This mode is not supported in Agilex 5 FPGAs.

Options	Value	Description
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 Parallel Flash Loader II Intel FPGA IP reconfigures the FPGA with data from the page that failed. • If you select **Retry from fixed address**, the Parallel Flash Loader II Intel FPGA IP 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 Parallel Flash Loader II Intel FPGA IP 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`.
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 (MT28EW)** • **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 (MT28EW)**—applicable for MT28EW 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**.

3.1.7.2.2. Parallel Flash Loader II Intel FPGA IP Signals

Table 29. Parallel Flash Loader II Intel FPGA IP Signals

Pin	Type	Weak Pull-Up	Function
pfl_nreset	Input	—	Asynchronous reset for the Parallel Flash Loader II Intel FPGA IP. Pull high to enable FPGA configuration. To prevent FPGA configuration, pull low when you do not use the Parallel Flash Loader II Intel FPGA IP. This pin does not affect the Parallel Flash Loader II Intel FPGA 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 Parallel Flash Loader II Intel FPGA IP 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 Parallel Flash Loader II Intel FPGA 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 Parallel Flash Loader II Intel FPGA IP for flash programming.
fpga_pgm[]	Input	—	Determines the page for the configuration. This pin is not available if you are only using the Parallel Flash Loader II Intel FPGA 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 Parallel Flash Loader II Intel FPGA 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 Parallel Flash Loader II Intel FPGA IP stops reading the data from the flash memory device. This pin is not available if you are only using the Parallel Flash Loader II Intel FPGA 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 Parallel Flash Loader II Intel FPGA 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 Parallel Flash Loader II Intel FPGA IP for flash programming.

continued...

Pin	Type	Weak Pull-Up	Function
pfl_flash_access_request	Output	—	For system-level synchronization. When necessary, this pin connects to a processor or an arbiter. The Parallel Flash Loader II Intel FPGA IP drives this pin high when the JTAG interface accesses the flash or the Parallel Flash Loader II Intel FPGA 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 Parallel Flash Loader II Intel FPGA IP.
flash_data[]	Input or Output (bidirectional pin)	—	Bidirectional data bus to transmit or receive 8-, 16-, or 32-bit[13] data. Intel recommends that you turn **On** the **Set flash bus pins to tri-state when not in use** option in the Parallel Flash Loader II Intel FPGA IP. [14]
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.

continued...

(13) 32-bit data is not supported in Agilex 5 devices.

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

Pin	Type	Weak Pull-Up	Function
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 Parallel Flash Loader II Intel FPGA IP. (14)
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

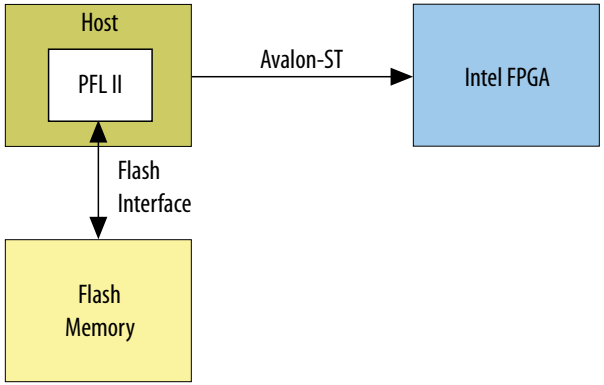
Avalon Interface Specifications

3.1.7.3. Generating the Parallel Flash Loader II Intel FPGA IP

3.1.7.3.1. Controlling Avalon-ST Configuration with Parallel Flash Loader II Intel FPGA IP

The Parallel Flash Loader II Intel FPGA IP in the host determines when to start the configuration process, read the data from the flash memory device, and configure the Agilex 5 device using the Avalon-ST configuration scheme.

Figure 30. FPGA Configuration with Flash Memory Data



You can use the Parallel Flash Loader II Intel FPGA IP to either program the flash memory devices, configure your FPGA, or both. To perform both functions, create separate Parallel Flash Loader II Intel FPGA IP 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 Parallel Flash Loader II Intel FPGA IP 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

3.1.7.3.2. Mapping Parallel Flash Loader II Intel FPGA IP and Flash Address

The address connections between the Parallel Flash Loader II Intel FPGA IP and the flash memory device vary depending on the flash memory device vendor and data bus width.

Figure 31. Flash Memory in 8-Bit Mode

The address connection between the Parallel Flash Loader II Intel FPGA IP and the flash memory device are the same.

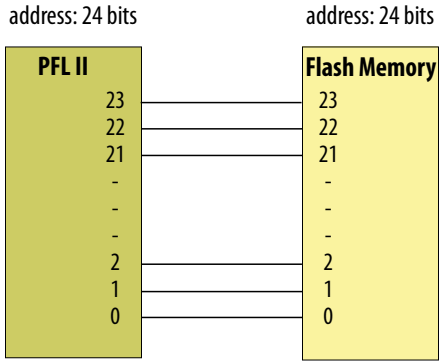


Figure 32. Flash Memories in 16-Bit Mode

The flash memory addresses in 16-bit flash memory shift one bit down in comparison with the flash addresses in Parallel Flash Loader II Intel FPGA IP. The flash address in the flash memory starts from bit 1 instead of bit 0.

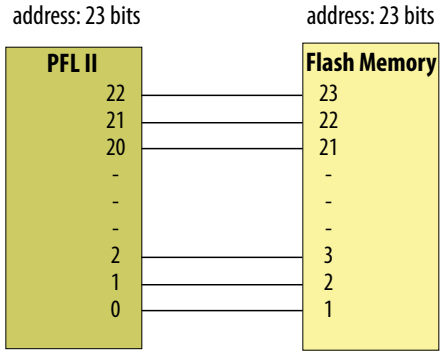


Figure 33. Cypress and Micron M28, M29 Flash Memory in 8-Bit Mode

The flash memory addresses in Cypress 8-bit flash shifts one bit up. Address bit 0 of the Parallel Flash Loader II Intel FPGA IP connects to data pin D15 of the flash memory.

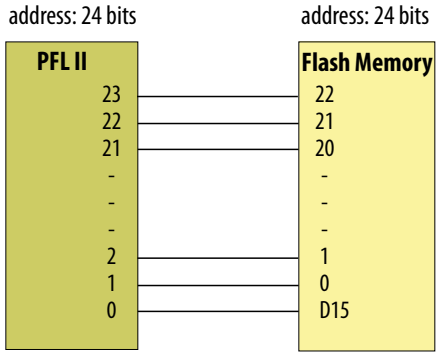
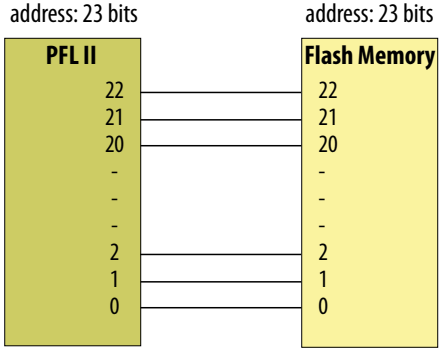


Figure 34. Cypress and Micron M28, M29 Flash Memory in 16-Bit Mode

The address bit numbers in the Parallel Flash Loader II Intel FPGA IP and the flash memory device are the same.



3.1.7.3.3. Creating a Single Parallel Flash Loader II Intel FPGA IP for Programming and Configuration

Follow these steps to create a single Parallel Flash Loader II Intel FPGA IP instantiation 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 and FPGA Configuration**.
3. In the same tab, for **What is the targeted flash?**, select **CFI Parallel Flash**.
4. Specify the parameters on the **Flash Interface Settings** tab:
   a. Select **1** for a single flash device.
   b. Select **CFI 1 Gbit** as the largest used flash device.
   c. Select **16 bits** for the flash interface data width.
5. Specify the parameters on the **FPGA Configuration** to match your design.
6. Compile and generate a `.pof` for the flash memory device. Ensure that you tri-state all unused I/O pins.

3.1.7.3.4. Creating Separate Parallel Flash Loader II Intel FPGA IP Functions

Follow these steps to create separate Parallel Flash Loader II Intel FPGA 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 Parallel Flash Loader II Intel FPGA IP 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 Parallel Flash Loader II Intel FPGA 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 CPLD 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 Parallel Flash Loader II block to tri-state the correct FPGA configuration pins.

3.1.7.4. Constraining the Parallel Flash Loader II Intel FPGA IP

You can specify many design constraints in Quartus Prime project settings by editing the `.sdc` files.

3.1.7.4.1. Parallel Flash Loader II Intel FPGA IP Recommended Design Constraints to FPGA Avalon-ST Pins

Example 1. Create a `pfl_clk` clock and a generated `AVST_CLK` clock

Example below creates a `pfl_clk` clock running at 50 MHz, supplied by the `clk_50m_sysmax` input clock.

```
set pfl_clk_period 20.000
create_clock -name {clk_50m_sysmax} -period $pfl_clk_period [get_ports {clk_50m_sysmax}]
create_generated_clock -name AVST_CLK -source [get_ports {clk_50m_sysmax}] [get_ports {avst_clk}]
```

Example 2. Set output delay for Parallel Flash Loader II Intel FPGA IP output pins

Example below sets the output delay for the AvST_DATA and AvST_VALID pins.

```
set avst_data_tracemax 0.250
set avst_data_tracemin 0.000
set avst_clk_tracemax 0.250
set avst_clk_tracemin 0.000
set fpga_Tsu 2.100
set fpga_Th 0.100
set fpga_out_max_dly [expr $avst_data_tracemax + $fpga_Tsu - $avst_clk_tracemin]
set fpga_out_min_dly [expr $avst_data_tracemin - $fpga_Th - $avst_clk_tracemax]

set_output_delay -add_delay -max -clock [get_clocks {AVST_CLK}] $fpga_out_max_dly [get_ports {avst_d[*] avst_valid}]
set_output_delay -add_delay -min -clock [get_clocks {AVST_CLK}] $fpga_out_min_dly [get_ports {avst_d[*] avst_valid}]
```

Example 3. Setting a false path

You can set the AVST_READY input pin to a false path since this pin is not synchronous to the AVST_CLK clock. The host must synchronize the AVST_READY signal to the AVST_CLK signal using a 2-stage register synchronizer.

```
set_false_path -from [get_ports {avst_ready}] -to *
```

3.1.7.4.2. Parallel Flash Loader II Intel FPGA IP Recommended Design Constraints for Using QSPI Flash

Example 4. Create a FLASH_CLK clock

Example below creates assigns QSPI flash clock pin (`flash_dc1_ic0`) to the flash clock.

```
create_generated_clock -name FLASH_CLK -source [get_ports {clk_50m_sysmax}] [get_ports {flash_dc1_io0}]
```

Example 5. Set output delay for Parallel Flash Loader II Intel FPGA IP output pins

Example below sets the output delay for the QSPI flash data and chip select pins.

```
#flash_dc1_io1/3/4/5 = QSPI flash data pins,
#flash_dc1_io2 = QSPI flash chip select pins
set flash_data_tracemax 0.250
set flash_data_tracemin 0.000
set flash_clk_tracemax 0.250
set flash_clk_tracemin 0.000
set flash_Tsu 2.700
set flash_Th 2.000
set flash_out_max_dly [expr $flash_data_tracemax + $flash_Tsu - $flash_clk_tracemin]
```

```
set flash_out_min_dly [expr $flash_data_tracemin - $flash_Th - $flash_clk_tracemax]

set_output_delay -add_delay -max \
-clock [get_clocks {FLASH_CLK}] $flash_out_max_dly [get_ports {flash_dc1_io1 flash_dc1_io3 flash_dc1_io4
flash_dc1_io5 flash_dc1_io2}]

set_output_delay -add_delay -min \
-clock [get_clocks {FLASH_CLK}] $flash_out_min_dly [get_ports { flash_dc1_io1 flash_dc1_io3 flash_dc1_io4
flash_dc1_io5 flash_dc1_io2}]
```

Example 6. Set input delay for input pins

Example below sets the input delay for the QSPI flash data.

```
set flash_tco_max 7.000
set flash_tco_min 1.000
set in_max_dly [expr $flash_data_tracemax + $flash_tco_max + $flash_clk_tracemax]
set in_min_dly [expr $flash_data_tracemin + $flash_tco_min + $flash_clk_tracemin]

set_input_delay -clock { FLASH_CLK } -max $in_max_dly [get_ports {flash_dc1_io1 flash_dc1_io3 flash_dc1_io4
flash_dc1_io5}]
set_input_delay -clock { FLASH_CLK } -min $in_min_dly [get_ports {flash_dc1_io1 flash_dc1_io3 flash_dc1_io4
flash_dc1_io5}]
```

3.1.7.4.3. Parallel Flash Loader II Intel FPGA IP Recommended Design Constraints for using CFI Flash

Example 7. Create a FLASH_CLK clock

Example below assigns CFI flash clock pin (`flash_clk`) to the flash clock. You constrain the flash_clk pin only when using burst mode.

```
create_generated_clock -name FLASH_CLK -source [get_ports {clk_50m_max5}] [get_ports {flash_clk}]
```

Example 8. Set output delay for output pins

Example below sets the output delay for CFI flash output pins.

```
set flash_data_tracemax 0.250
set flash_data_tracemin 0.000
set flash_clk_tracemax 0.250
set flash_clk_tracemin 0.000
set flash_Tsu 3.500
set flash_Th 2.000
set flash_out_max_dly [expr $flash_data_tracemax + $flash_Tsu - $flash_clk_tracemin]
set flash_out_min_dly [expr $flash_data_tracemin - $flash_Th - $flash_clk_tracemax]
```

```
#Note: For normal mode, the clock is referred to input pfl_clk clock(clk_50m_max5) of PFL II IP.
#If burst mode is used, the clock is referred to flash clock of PFL II IP.

set_output_delay -add_delay -max -clock [get_clocks {clk_50m_max5}] \
$flash_out_max_dly [get_ports {flash_nce[0] flash_nce[1] flash_noe flash_nwe flash_addr[*] flash_data[*]}]

set_output_delay -add_delay -min -clock [get_clocks {clk_50m_max5}] \
$flash_out_min_dly [get_ports { flash_nce[0] flash_nce[1] flash_noe flash_nwe flash_addr[*] flash_data[*]}]

#Only need to constraint flash_advn pin when using burst mode.
set_output_delay -add_delay -max -clock [get_clocks { FLASH_CLK }] $flash_out_max_dly [get_ports {flash_nadv}]
set_output_delay -add_delay -min -clock [get_clocks { FLASH_CLK }] $flash_out_min_dly [get_ports {flash_nadv}]
```

Example 9. Set input delay for input pins

Example below sets the input delay for CFI flash data.

```
# For Normal Mode
set flash_noe_tracemax 0.250
set flash_noe_tracemin 0.000
set flash_tco_max 7.000
set flash_tco_min 0.000
set normal_in_max_dly [expr $flash_data_tracemax + $flash_tco_max + $ flash_noe_tracemax]
set normal_in_min_dly [expr $flash_data_tracemin + $flash_tco_min + $ flash_noe_tracemin]

set_input_delay -clock { clk_50m_max5 } -max $normal_in_max_dly [get_ports {flash_data[*]}]
set_input_delay -clock { clk_50m_max5 } -min $normal_in_min_dly [get_ports {flash_data[*]}]
```

```
# For Burst mode
set flash_tco_max 5.500
set flash_tco_min 2.000
set burst_in_max_dly [expr $flash_data_tracemax + $flash_tco_max + $flash_clk_tracemax]
set burst_in_min_dly [expr $flash_data_tracemin + $flash_tco_min + $flash_clk_tracemin]

set_input_delay -clock { FLASH_CLK } -max $burst_in_max_dly [get_ports {flash_data[*]}]
set_input_delay -clock { FLASH_CLK } -min $burst_in_min_dly [get_ports {flash_data[*]}]
```

3.1.7.4.4. Parallel Flash Loader II Intel FPGA IP Recommended Constraints for Other Input Pins

Example 10. Set a false path for Parallel Flash Loader II Intel FPGA IP input pins

You can set the `pfl_nreset` input reset pin to a false path since this pin is asynchronous.

```
set_false_path -from [get_ports {pfl_nreset}] -to *
```

Example 11. Set input delay to Parallel Flash Loader II Intel FPGA IP input pins

Example below sets the input delay for the `pfl_flash_access_granted` pin.

- You don't have to constraint the path when you use the device arbiter logic to control the pin,
- You don't have to constraint the path when not using the device arbiter logic or the external processor to control the pin and you loopback the `pfl_flash_access_request` signal to the `pfl_flash_access_granted` pin.
- You can constraint the path when a processor or an external device controls the `pfl_flash_access_granted` pin.

```
set_input_delay -clock {clk_50m_sysmax} -max [<pfl_flash_access_granted_tco_max> +
<pfl_flash_access_granted_tracemax>] [get_ports {pfl_flash_access_granted}]
set_input_delay -clock {clk_50m_sysmax} -min [<pfl_flash_access_granted_tco_min> +
<pfl_flash_access_granted_tracemin>] [get_ports {pfl_flash_access_granted}]
```

Example 12. Set a false path for `fpga_pgm[]` input pin

You can set a false path to quasi-static signals such as reset and configuration signals (`fpga_conf_done`, `fpga_nstatus`) that are stable for a long periods of time.

```
set_false_path -from [get_ports {fpga_pgm[]}] -to *
```

Example 13. Set input delay to `pfl_nreconfigure` input pin

If you use the external component to drive this pin, you must set the input delay path to drive the `pfl_nreconfigure` pin.

```
set_input_delay -clock {clk_50m_sysmax} -max [<pfl_nreconfigure_tco_max> + <pfl_nreconfigure_tracemax>] \
[get_ports {pfl_nreconfigure}]

set_input_delay -clock {clk_50m_sysmax} -min [<pfl_nreconfigure_tco_min> + <pfl_nreconfigure_tracemin>] \
[get_ports {pfl_nreconfigure}]
```

Example 14. Set input delay to `pfl_reset_watchdog` pin

If you use the external component to drive this pin, you must set the input delay path to drive the `pfl_reset_watchdog` pin.

```
set_input_delay -clock {clk_50m_sysmax} -max [$pfl_reset_watchdog_tco_max + $pfl_reset_watchdog_tracemax] \
[get_ports {pfl_nreconfigure}]

set_input_delay -clock {clk_50m_sysmax} -min [$pfl_reset_watchdog_tco_min + $pfl_reset_watchdog_tracemin] \
[get_ports {pfl_nreconfigure}]
```

3.1.7.4.5. Parallel Flash Loader II Intel FPGA IP Recommended Constraints for Other Output Pins

Example 15. Set output delay to Parallel Flash Loader II Intel FPGA IP output pins

Example below sets the output delay for the `pfl_flash_access_request` output pin.

- You don't have to constraint the path when this signal feeds arbiter logic or device tristate logic,
- You don't have to constraint the path when this signal feeds the `pfl_flash_access_granted` input pin while not using the device arbiter logic or the external processor.
- You can constraint the path when this signal feeds the processor or an external device controls.

```
set_output_delay -add_delay -clock [get_clocks { clk_50m_sysmax }] \
-max $flash_access_request_tracemax [get_ports {pfl_flash_access_request}]

set_output_delay -add_delay -clock [get_clocks { clk_50m_sysmax }] \
-min $flash_access_request_tracemin [get_ports {pfl_flash_access_request}]
```

Example 16. Set output delay to `flash_nreset` output pin

The `flash_nreset` output pin is available in Burst mode only.

```
set_output_delay -add_delay -max -clock [get_clocks { FLASH_CLK }] $flash_out_max_dly [get_ports {flash_nreset}]
set_output_delay -add_delay -min -clock [get_clocks { FLASH_CLK }] $flash_out_min_dly [get_ports {flash_nreset}]
```

Example 17. Set a false path for `fpga_nconfig` output pin

You can set the `fpga_nconfig` output pin to a false path since the `nCONFIG` is an asynchronous input pin..

```
set_false_path -from [get_ports {fpga_nconfig}] -to *
```

Example 18. Set output delay to `pfl_watchdog_error` output pin

- You don't have to constraint the path when the signal feeds the internal logic.
- You can constraint the path when signal feeds an external host.

```
set_output_delay -add_delay -clock [get_clocks { clk_50m_sysmax }] \
-max $pfl_watchdog_error_tracemax [get_ports {pfl_watchdog_error}]

set_output_delay -add_delay -clock [get_clocks { clk_50m_sysmax }] \
-min $pfl_watchdog_error_tracemin [get_ports {pfl_watchdog_error}]
```

3.1.7.5. Using the Parallel Flash Loader II Intel FPGA IP

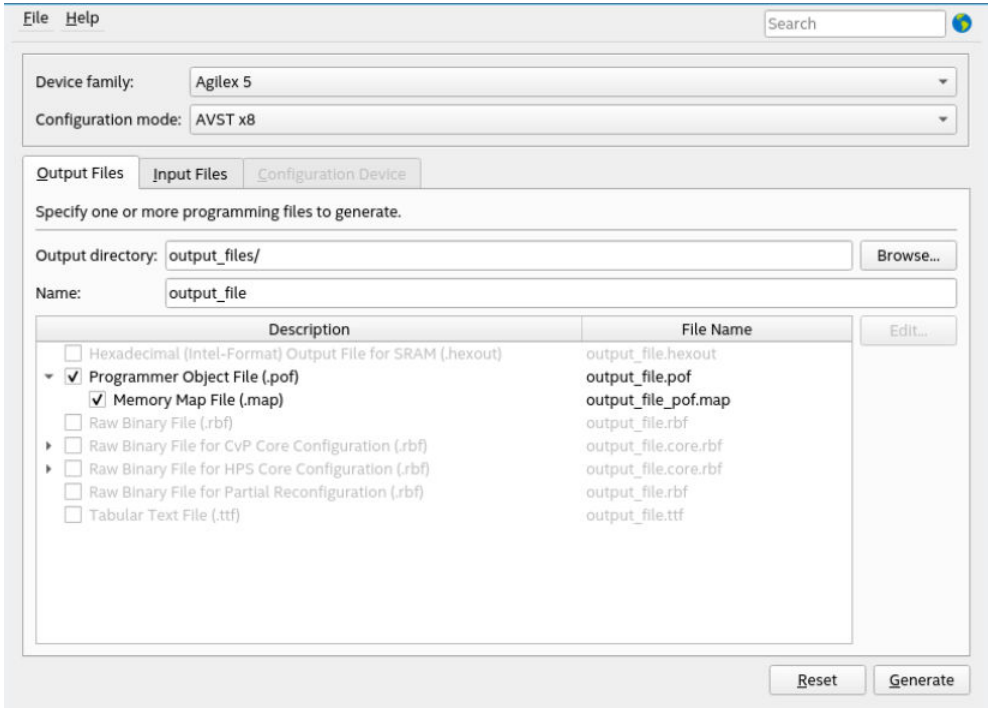
3.1.7.5.1. Converting .sof to .pof File

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.

Complete these steps to convert `.sof` file to a `.pof`:

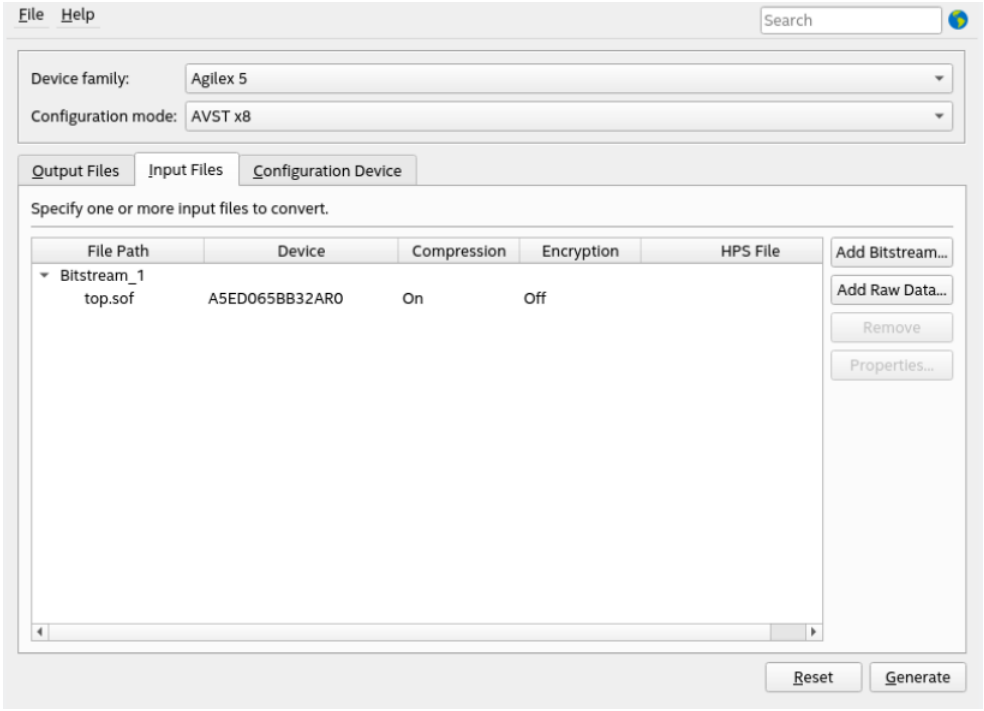
1. Click **File ➤ Programming File Generator**.
2. For **Device family** select **Agilex 5**.
3. For **Configuration mode** select Avalon-ST configuration scheme that you plan to use.
4. For **Output directory**, click **Browse** to select your output file directory.
5. For **Name** specify a name for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

Figure 35. Programming File Generator Output Files Tab



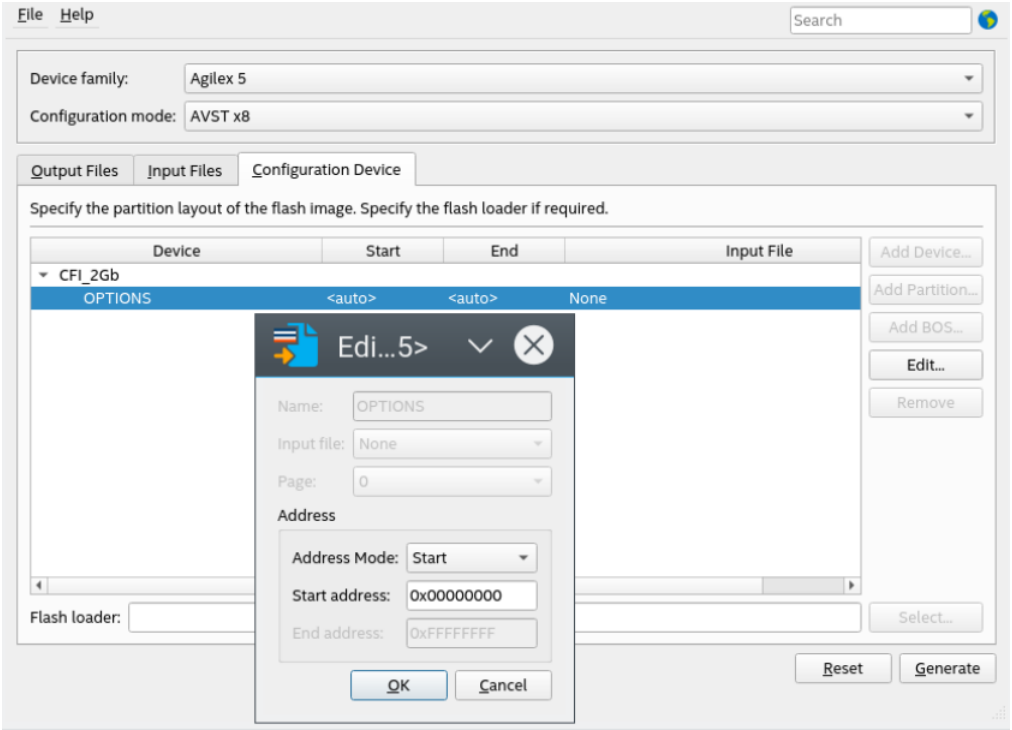
7. To specify a `.sof` that contains the configuration bitstream, on the **Input Files** tab, click **Add Bitstream**.

Figure 36. Input Files Tab



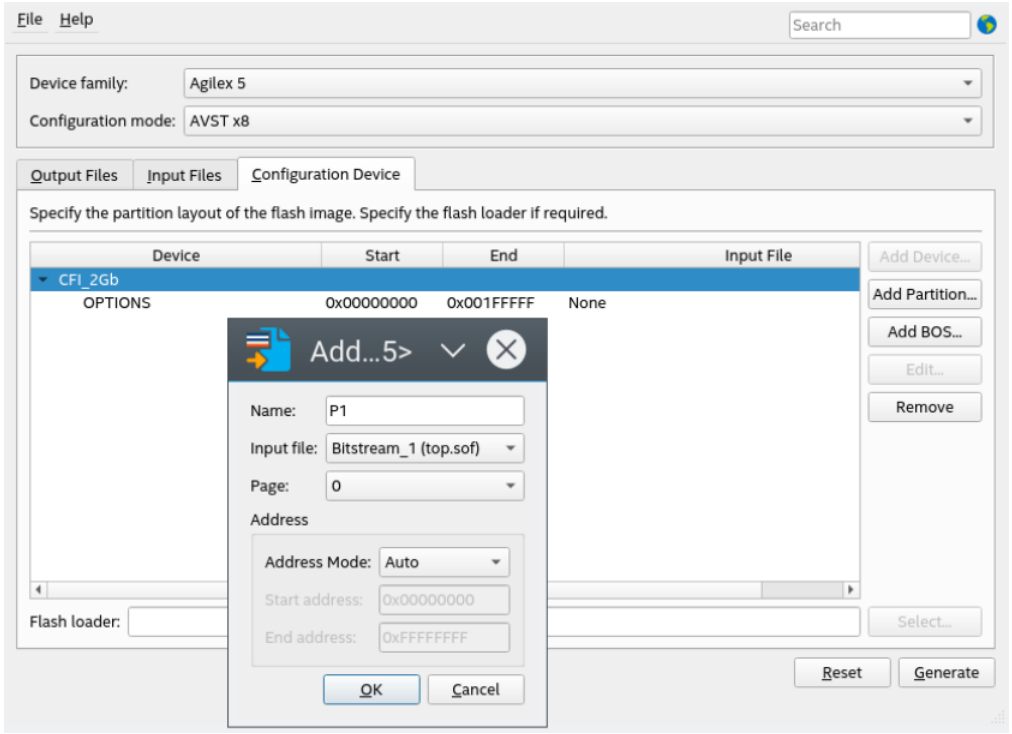
8. To include raw data, click **Add Raw Data** and specify a Hexadecimal (Intel-Format) Output File (`.hex`) or binary (`.bin`) file. This step is optional.
9. On the **Configuration Device** tab, click **Add device**. The **Add Device** dialog box appears. Select your flash device from the drop-down list of available parallel flash devices.
10. 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 Parallel Flash Loader II Intel FPGA 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 37. Edit Partition: OPTIONS for Flash Device



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

Figure 38. 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.

3.1.7.5.2. Programming CPLDs and Flash Memory Devices Sequentially

This procedure provides a single set of instructions for the 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.

3.1.7.5.3. 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 Parallel Flash Loader II Intel FPGA 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.

3.1.7.5.4. Defining New CFI Flash Memory Device

The Parallel Flash Loader II Intel FPGA IP (PFL II) 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 30. 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 39. Using the Programmer Edit Menu to Define a New Flash Device

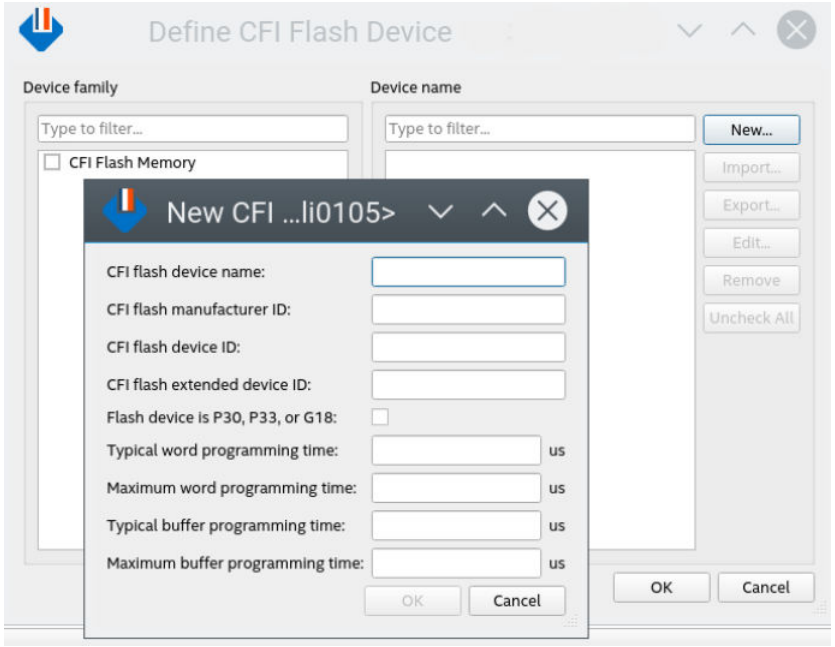


Table 31. Parameter Settings for New CFI Flash Device

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

continued...

Parameter	Description
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 Quartus Prime Pro Edition Programmer.

3.1.7.6. Supported Flash Memory Devices

The Quartus Prime software generates the PFL II IP core logic for the flash programming bridge and FPGA configuration.

Table 32. CFI Flash Memory Devices Supported by PFL II Intel FPGA IP Core

If your CFI device is not listed in this table but is compatible with an Quartus Prime flash device, Intel recommends selecting **Define CFI Flash Device** in the Quartus Prime software.

Manufacturer	Product Family	Data Width	Density (Mbit)	Device Name
Micron	MT28EW	8 or 16	256	MT28EW256ABA1
			512	MT28EW512ABA1
			1024	MT28EW01GABA1
Infineon	GL-S	16	256	S29GL256S
			512	S29GL512S
			1024	S29GL01GS

Table 33. Quad SPI Flash Memory Devices Supported by PFL II Intel FPGA IP Core

Manufacturer	Product Family	Density (Mbit)	Device Name
Micron	MT25QU	256	MT25QU256ABA
		512	MT25QU512ABB
		1024	MT25QU01GBBB
		2048	MT25QU02GCBB
Macronix	MX66U	2048	MX66U2G45G

3.2. AS Configuration

In AS configuration schemes, the SDM block in the Agilex 5 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 Agilex 5 device. For more information about the power sequence, refer to the *Agilex 5 Power Management User Guide.*

Important: You must connect the serial flash or quad SPI flash reset pin to the AS_nRST pin. The SDM must fully control the QSPI reset. Do not connect the quad SPI reset pin to any external host.

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

Table 34. Agilex 5 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	166 MHz	664 Mbps	Fast mode - 001 Normal mode - 011

Table 35. Required Configuration Signals for the AS Configuration Scheme

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	Input	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}
AS_nRST(15)	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.

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_nCS00 asserts low when the SDM starts to communicate with the QSPI flash. The remaining chip select pins, AS_nCS01 - AS_nCS03 deassert high.

Related Information

- Agilex 5 FPGAs and SoCs Device Data Sheet
- Pin Connection Guidelines: Agilex 5 FPGAs and SoCs
- Power Management User Guide: Agilex 5 FPGAs and SoCs

(15) Even if not currently utilized, connect the pin to ensure future compatibility.

3.2.1. AS Configuration Scheme Hardware Components and File Types

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

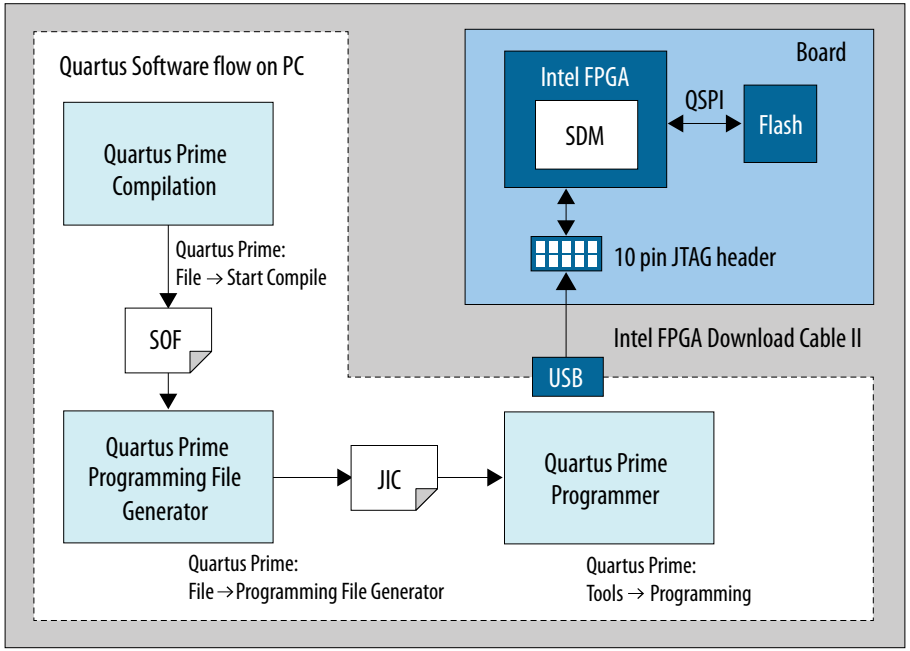
- Quad SPI flash memory
- The Intel FPGA Download Cable II to connect the Quartus Prime Programmer to the PCB.

For PCIe designs including the Configuration via Protocol (CvP), Intel recommends that you use Micron QSPI flash memory. When using the Micron QSPI flash memory, the boot ROM uses the AS x4 mode to load the initial configuration firmware faster to meet the PCIe wake-up time for host enumeration.

When using other flash memory types for PCIe designs, the boot ROM reads the firmware using AS x1 mode. Intel recommends asserting the PERST# signal low for a minimum of 200 ms counting from the device exiting the power-on reset (POR) state. This ensures the PCIe endpoint enters the link training state prior to the PERST# signal deasserts.

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

Figure 40. 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 Quartus Prime or Third-Party Programmer.

Table 36. 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.

continued...

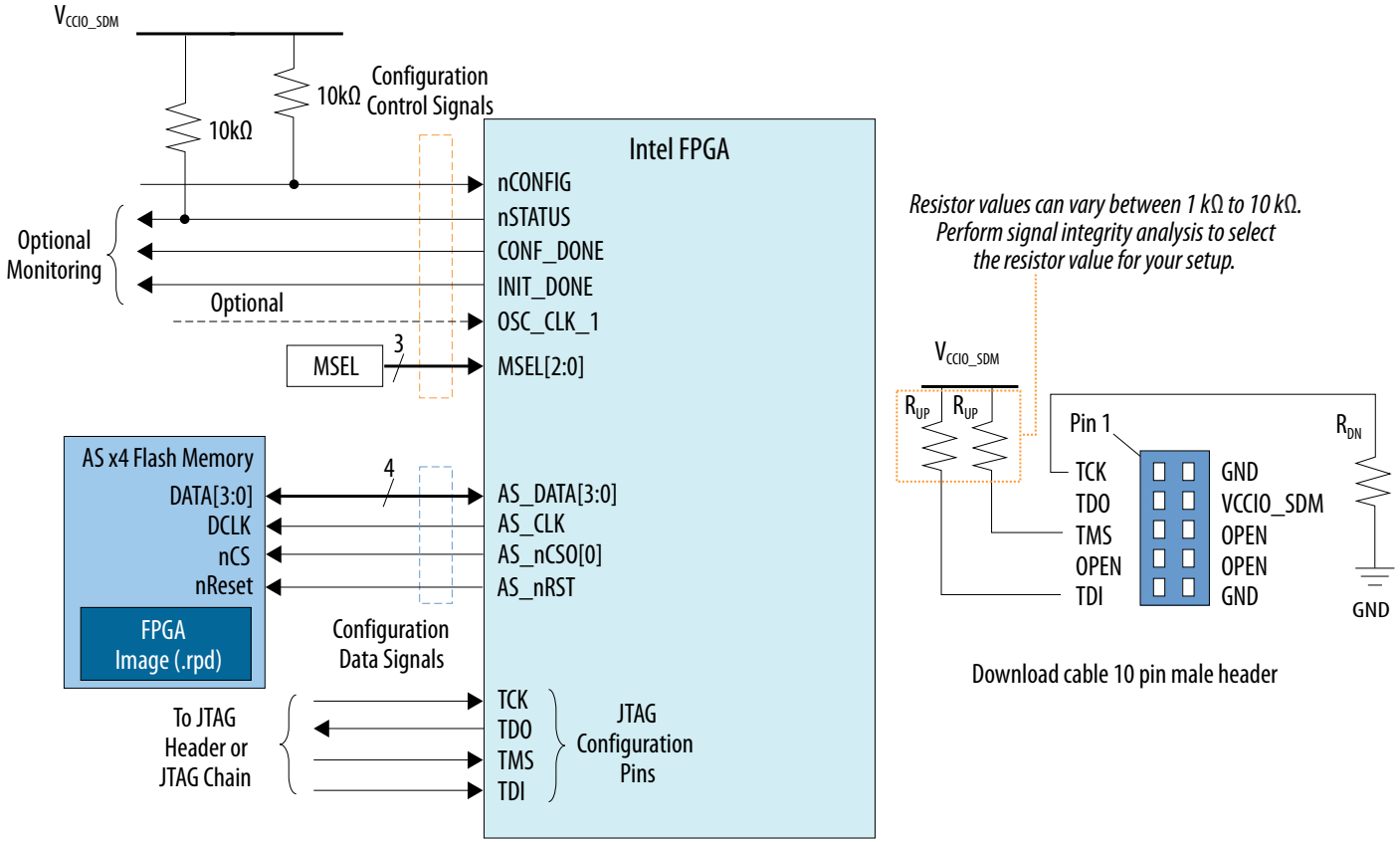
Programming File Type	Extension	Description
		You initially use the JTAG cable to load a SDM Helper SOF into the Agilex 5 device. The SDM can then load the flash device with the Agilex 5 design.
Raw Programming Data File	`.rpd`	Stores data for configuration with a third-party programming hardware. You generate Raw Programming Data Files from a `.pof` or `.sof`. The `.rpd` file is a subset of a `.pof` or `.jic` that includes only device-specific binary programming data for Active Serial configuration scheme with quad SPI configuration devices and remote system update.

Related Information

Programming Serial Flash Devices using the AS Interface on page 115

3.2.2. AS Single-Device Configuration

Figure 41. Connections for AS x4 Single-Device Configuration



Related Information

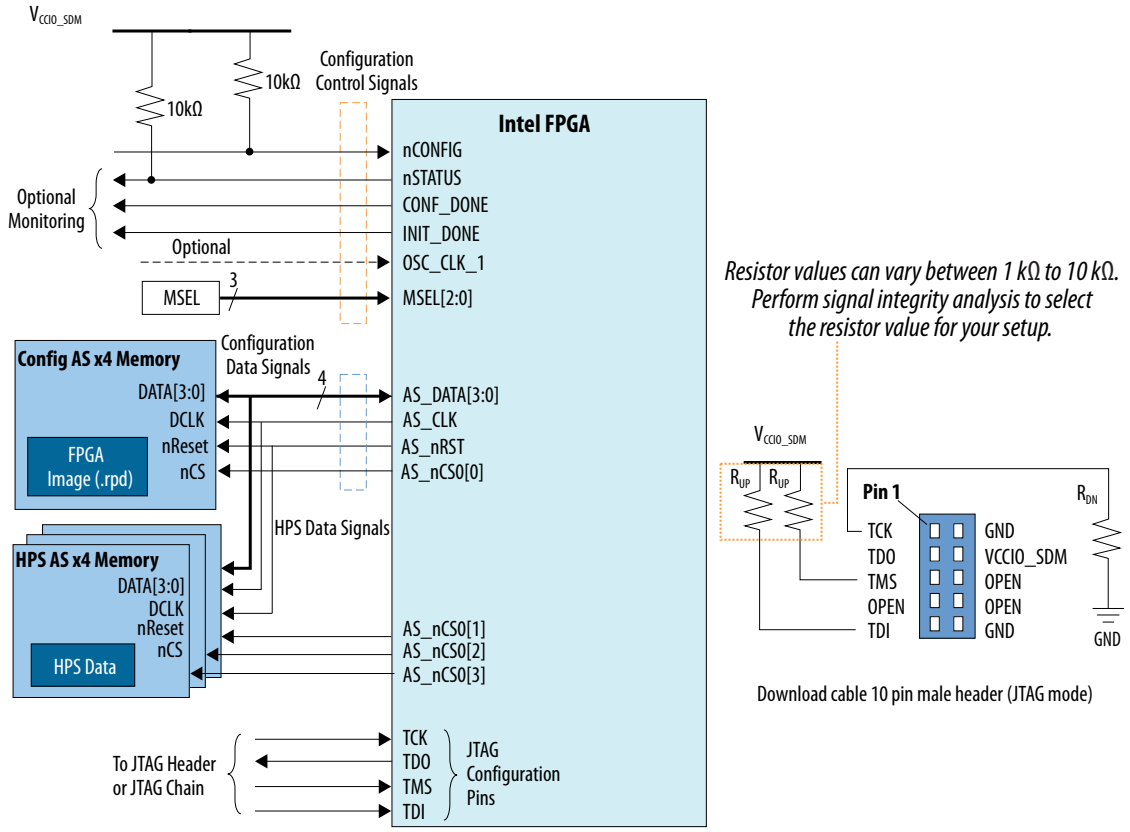
- MSEL Settings on page 31
- Pin Connection Guidelines: Agilex 5 FPGAs and SoCs

3.2.3. AS Using Multiple Serial Flash Devices

Agilex 5 devices support one AS x4 flash memory device for AS configuration. Once the device enters user mode, you can then use up to four AS x4 flash memories with Mailbox Client Intel FPGA IP or HPS as data storage. The MSEL pins operate as MSEL only during POR state. After SDM samples the MSEL pins during the boot ROM state for AS x4 mode, the SDM repurposes the MSEL pins as chip select pins. You must 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.

Refer to Command Sequence to Perform Quad SPI Operations for additional information about accessing multiple serial flash devices via Mailbox Client Intel FPGA IP.

Figure 42. 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 37. Maximum AS_CLK Frequency as a Function of Board Capacitance Loading and Clock Source

This table is preliminary.

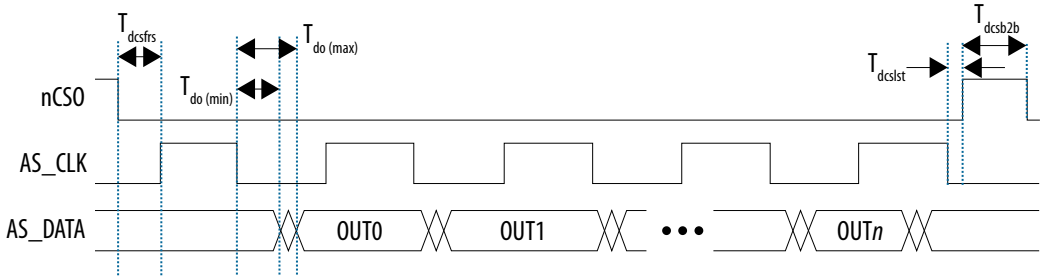
Capacitance Loading (pF)	Maximum Supported AS_CLK (MHz)	
	OSC_CLK_1[16] (MHz)	Internal Oscillator (MHz)
10	166/125	115
30	100	77
37	71.5	77
80	50	58
140	25	25

Related Information

MSEL Settings on page 31

3.2.4. AS Configuration Timing Parameters

Figure 43. AS Configuration Serial Output Timing Diagram



(16) For Agilex 5 devices with speed grade -6S and -6X devices, the clock speed for configuration network runs at 200 MHz when using OSC_CLK_1 and only supports AS_CLK at frequency of 25 MHz, 50 MHz, and 100 MHz.

Figure 44. AS Configuration Serial Input Timing Diagram

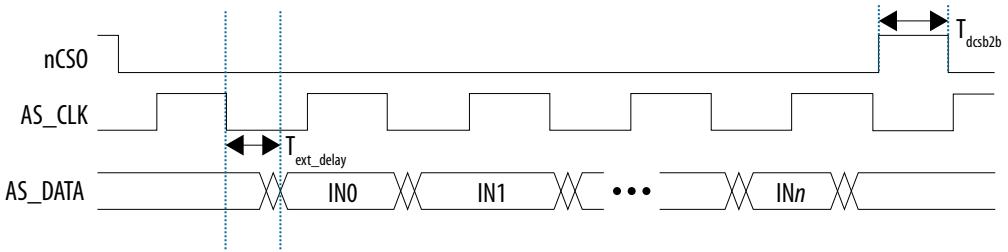


Table 38. T_{ext_delay} as a Function of AS_CLK Frequency

Symbol	Configuration Clock Source	Frequency	Min (ns)	Max (ns)
T_{ext_delay}	Internal Oscillator	115 MHz	0	20
		77 MHz	0	20
		58 MHz	0	20
		25 MHz	0	24
	OSC_CLK_1[17]	166 MHz	0	13.5
		125 Mhz	0	18
		100 MHz	0	24
		71.5 MHz	0	35
		50 MHz	0	24
		25 MHz	0	24

Note: For more information about the timing parameters, refer to the *Agilex 5 Device Datasheet*.

Related Information

Agilex 5 FPGAs and SoCs Device Data Sheet

(17) For Agilex 5 devices with speed grade -6S and -6X devices, the clock speed for configuration network runs at 200 MHz when using OSC_CLK_1 and only supports AS_CLK at frequency of 25 MHz, 50 MHz, and 100 MHz.

3.2.5. Skew Tolerance Guidelines

3.2.5.1. Maximum Allowable External AS_DATA Pin Skew Delay Guidelines

You must minimize the skew on the AS_DATA and AS_CLK 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

Use the following equations to determine the skew between AS_CLK and AS_DATA:

1. Skew(AS_CLK – AS_DATA) > –AS_CLK/2 + $T_{do(max)}$ + T_{su}
2. Skew(AS_CLK – AS_DATA) < AS_CLK/2 + $T_{do(min)}$ – T_{ho}

Hence, the allowable range for skew between AS_CLK and AS_DATA is as follows:

–AS_CLK/2 + $T_{do(max)}$ + T_{su} < Skew(AS_CLK – AS_DATA) < AS_CLK/2 + $T_{do(min)}$ – T_{ho}

- T_{su} = Data setup time required by the quad SPI flash. Refer to your quad SPI flash datasheet.
- T_{ho} = Data hold time required by the quad SPI flash. Refer to your quad SPI flash datasheet.
- T_{do} = AS_DATA[3:0] output delay. Refer to the AS configuration timing specifications in the *Agilex 5 Device Data Sheet*.
- AS_CLK = AS_CLK clock period.

Example 19. Example to Determine the Skew for 1 GB Quad SPI Flash Devices

T_{su} = 1.75 ns

T_{ho} = 2.0 ns

$T_{do(max)}$ = 0.6 ns

$T_{do(min)}$ = –0.6 ns

AS_CLK = 10 ns (100 MHz)

1. Skew(AS_CLK – AS_DATA) > –AS_CLK/2 + $T_{do(max)}$ + T_{su}

   Skew(AS_CLK – AS_DATA) > –10/2 + 0.6 + 1.75

Skew(AS_CLK – AS_DATA) > –2.65 ns

2. Skew(AS_CLK – AS_DATA) < AS_CLK/2 + $T_{do(min)}$ – T_{ho}

   Skew(AS_CLK – AS_DATA) < 10/2 – 0.6 – 2.0

   Skew(AS_CLK – AS_DATA) < 2.4 ns

The allowable range for skew between AS_CLK and AS_DATA is –2.65 ns < Skew(AS_CLK – AS_DATA) < 2.4 ns

Related Information

Agilex 5 FPGAs and SoCs Device Data Sheet

3.2.5.2. Maximum Allowable External nCSO Pin Skew Delay Guidelines

Use the following equations to determine the skew between nCSO and AS_CLK:

1. Skew (T_{bd_clk} – T_{bd_ncso}) > T_{su_ncso} – T_{dcsfrs}
2. Skew (T_{bd_clk} – T_{bd_ncso}) < AS_CLK/2 + T_{dcslst} – T_{ho_ncso}

Hence, the allowable range for skew between nCSO and AS_CLK is as follows:

T_{su_ncso} - T_{dcsfrs} < Skew (T_{bd_clk} – T_{bd_ncso}) < AS_CLK/2 + T_{dcslst} - T_{ho_ncso}

- T_{su_ncso} = Chip select setup time required by the quad SPI flash. Refer to your quad SPI flash datasheet.
- T_{ho_ncso} = Chip select hold time required by the quad SPI flash. Refer to your quad SPI flash datasheet.
- T_{dcsfrs} = AS_nCSO[3:0] asserted to first AS_CLK edge. Refer to the AS configuration timing specifications in the *Agilex 5 Device Data Sheet*.
- T_{dcslst} = Last AS_CLK edge to AS_nCSO[3:0] deasserted. Refer to the AS configuration timing specifications in the *Agilex 5 Device Data Sheet*.
- AS_CLK = AS_CLK clock period.

Example 20. Example to Determine the Skew for 1 GB Quad SPI Flash Devices

T_{su_ncso} = 2.7 ns

T_{ho_ncso} = 3.375 ns

T_{dcsfrs} = 8.5 ns

T_{dcslst} = 6.8 ns

AS_CLK = 6.024 ns (166 MHz)

1. Skew (T_{bd_clk} – T_{bd_ncso}) > T_{su_ncso} – T_{dcsfrs}
   Skew (T_{bd_clk} – T_{bd_ncso}) > 2.7 – 8.5
   Skew (T_{bd_clk} – T_{bd_ncso}) > –5.8 ns
2. Skew (T_{bd_clk} – T_{bd_ncso}) < AS_CLK/2 + T_{dcslst} – T_{ho_ncso}
   Skew (T_{bd_clk} – T_{bd_ncso}) < 6.024/2 + 6.8 – 3.375
   Skew (T_{bd_clk} – T_{bd_ncso}) < 6.437 ns

The allowable range for skew between nCS0 and AS_CLK is -5.8 ns < Skew (T_{bd_clk} – T_{bd_ncso}) < 6.437 ns

Related Information

Agilex 5 FPGAs and SoCs Device Data Sheet

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 interfaces for in-system programming:

- Active Serial
- JTAG

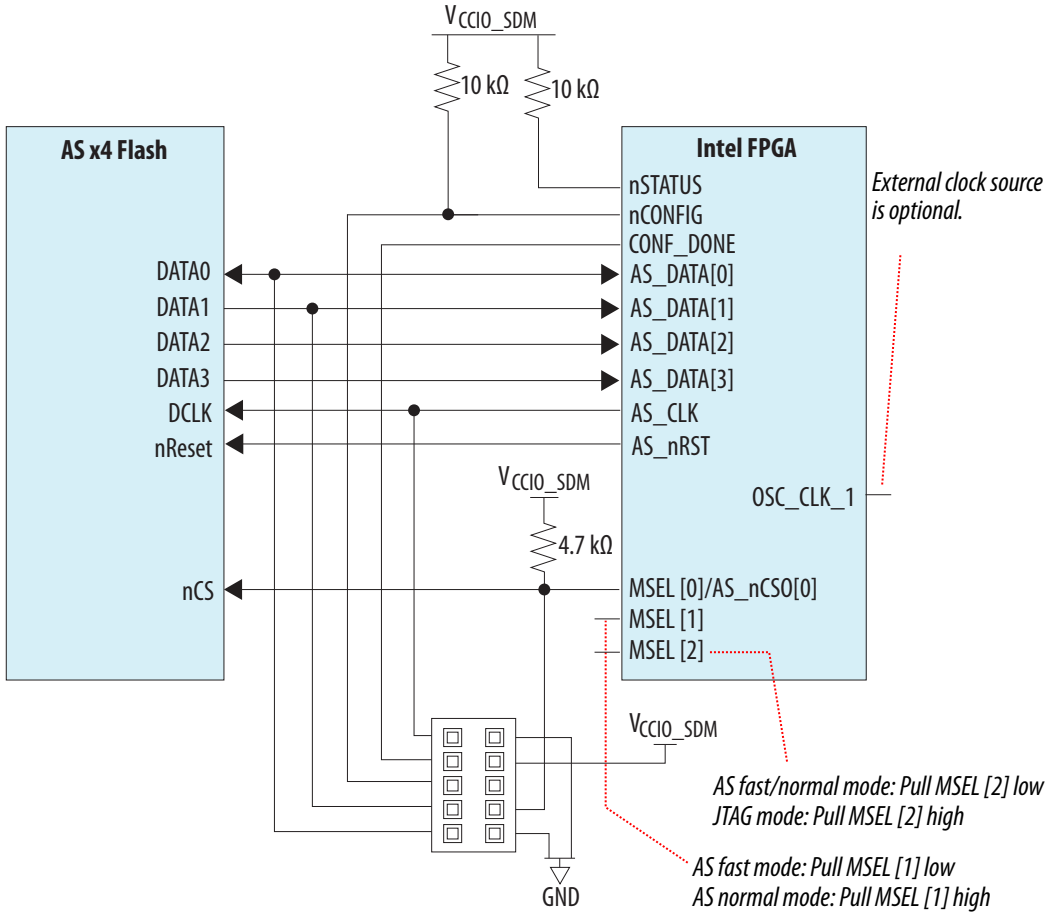
3.2.6.1. Programming Serial Flash Devices using the AS Interface

When you select AS programming, the 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_nRST, AS_DATA0-AS_DATA3, and AS_nCS00-AS_nCS03. The 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.

Attention: When you power up the Agilex 5 device with an empty serial flash device and use the AS interface to program the `.rpd` file into this serial flash device, you must power cycle the Agilex 5 device to configure the device from the flash successfully.

Figure 45. AS Programming Using Quartus Prime or Third-Party Programmer



3.2.6.2. Programming Serial Flash Devices using the JTAG Interface

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

Figure 46. Programming Your Serial Configuration Device Using JTAG

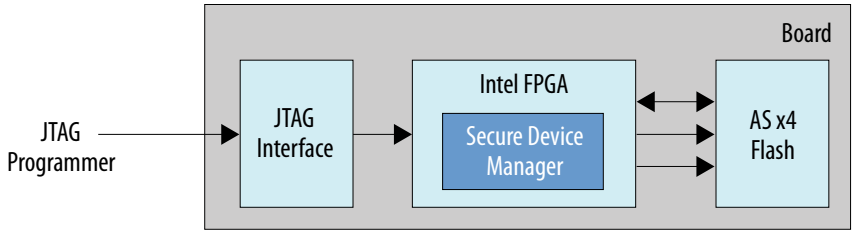
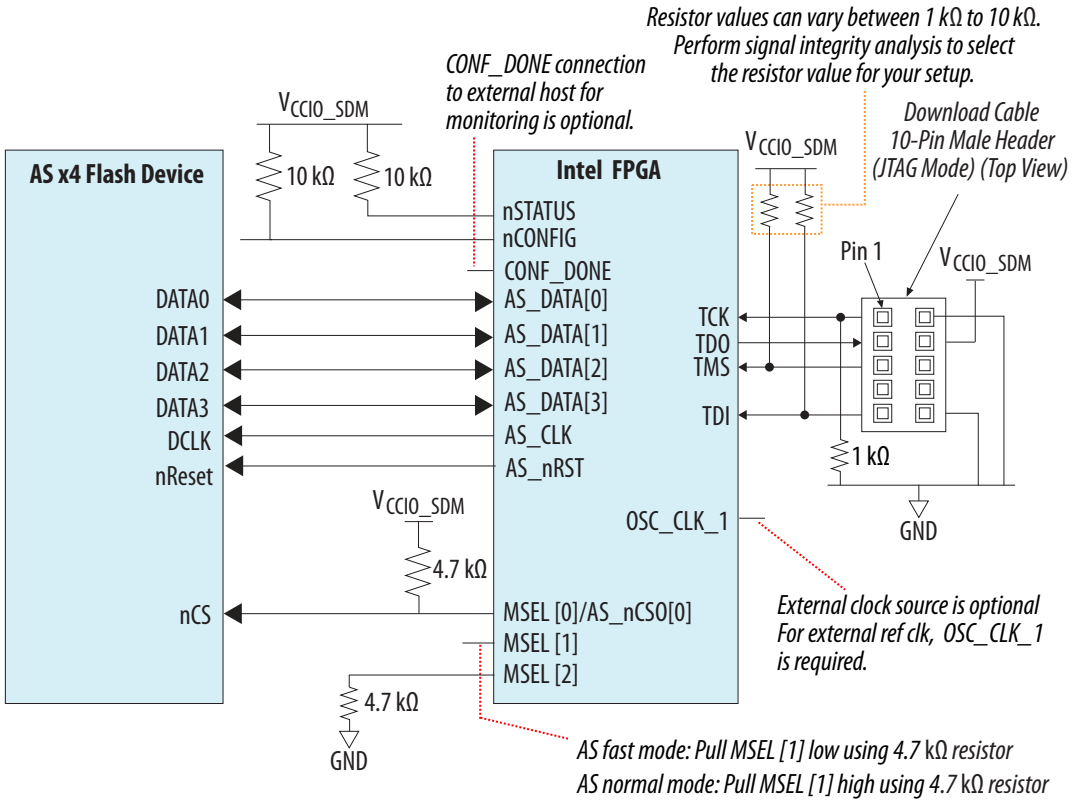


Figure 47. Connections for Programming the Serial Flash Devices using the JTAG Interface



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

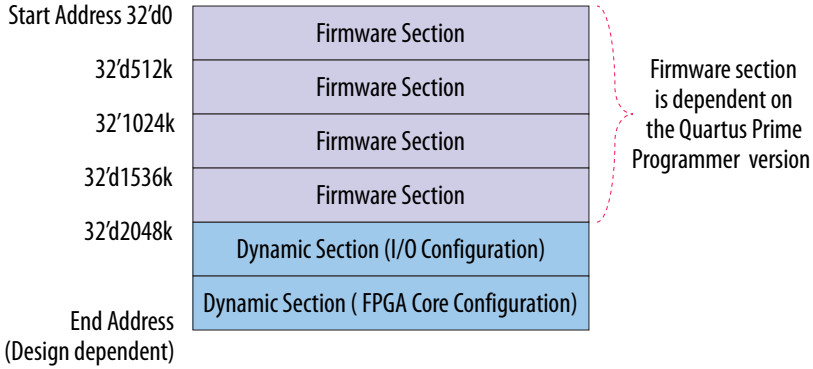
3.2.7. Serial Flash Memory Layout

Serial flash devices store the configuration data in sections. The HPS first stage boot loader (FSBL) location depends on the selected serial flash memory layout.

Non-HPS Case:

The following diagram illustrates sections of a non-HPS Agilex 5 configuration data mapping in a serial flash device. The non-HPS bitstreams do not include first stage boot loader (FSBL). Refer to *Agilex 5 SoC FPGA Bitstream Sections* of the *HPS Technical Reference Manual* for more information about flash memory layout for HPS devices.

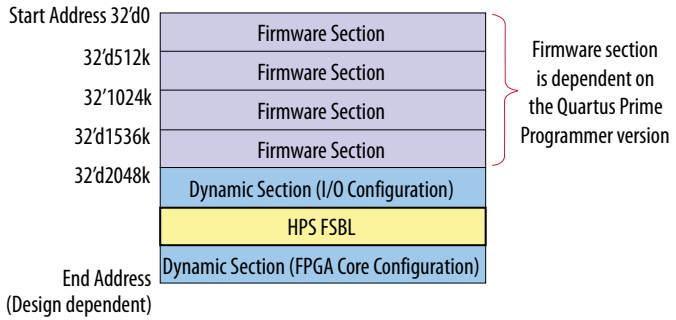
Figure 48. Serial Flash Memory Layout Diagram: Non-HPS Case



HPS Case with FPGA First Option:

The following diagram illustrates sections of an HPS Agilex 5 configuration data mapping in a serial flash device and the HPS first stage boot loader (FSBL) location when you select the FPGA First option.

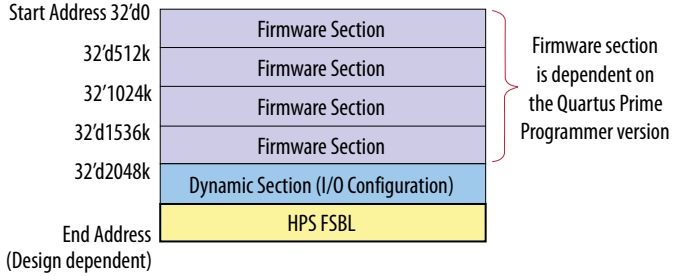
Figure 49. Serial Flash Memory Layout Diagram: HPS Case with FPGA First Option



HPS Case with HPS First Option and Dual Flash:

The following diagram illustrates sections of an HPS Agilex 5 configuration data mapping in a serial flash device when you select the HPS First option.

Figure 50. Serial Flash Memory Layout Diagram: HPS Case with HPS First Option



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 Agilex 5 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.

Agilex 5 devices support various third-party flash devices operating at 1.8 V. For more details, refer to *Intel Supported Third Party Configuration Devices*.

Related Information

Device Configuration-Support Center: Intel Supported Third Party Configuration Devices
Shows the criteria of third party configuration devices supported by Quartus Prime Programming File Generator and Quartus Prime Programmer.

3.2.8. AS_CLK

The Agilex 5 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 166 MHz.

Set the maximum required frequency for the AS_CLK pin in the Quartus Prime software as described in Active Serial Configuration Software Settings on page 122. The AS_CLK pin runs at or below your selected frequency.

Table 39. Supported Configuration Clock Source and AS_CLK Frequencies in Agilex 5 Devices

The table displays valid AS_CLK settings for the respective configuration clock source.

Configuration Clock Source	AS_CLK Frequency (MHz)
Internal oscillator	25 58 77 115
OSC_CLK_1[18] (25/100/125 MHz)	25
	continued...

(18) For Agilex 5 devices with speed grade -6S and -6X devices, the clock speed for configuration network runs at 200 MHz when using OSC_CLK_1 and only supports AS_CLK at frequency of 25 MHz, 50 MHz, and 100 MHz.

Configuration Clock Source	AS_CLK Frequency (MHz)
	50 71.5 100 125 166

Note: The configuration fails if the firmware receives bitstream with an invalid AS_CLK setting.

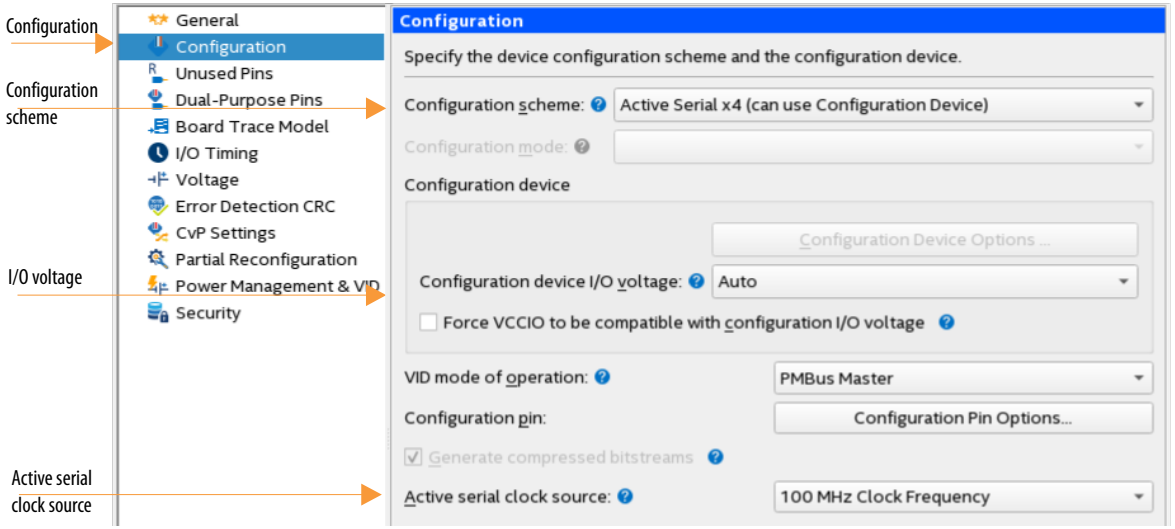
3.2.9. Active Serial Configuration Software Settings

You must set the parameters in the **Device and Pin Options** of the 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.

Figure 51. Selecting the Configuration Scheme



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. Quartus Prime Programming Steps

3.2.10.1. Generating Programming Files using the Programming File Generator

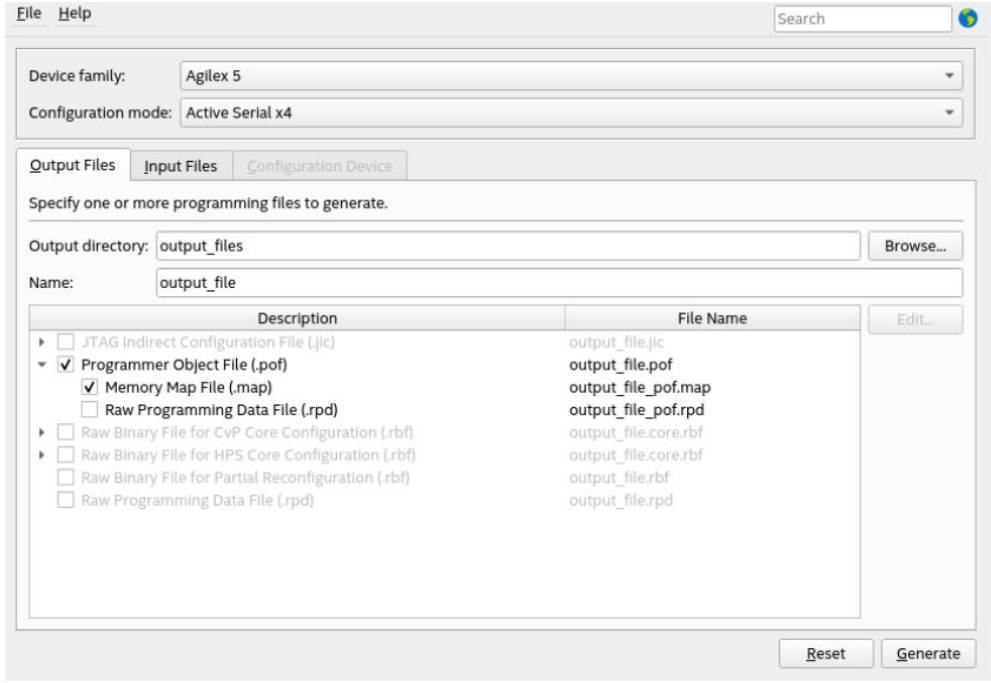
By default, the Quartus Prime Compiler's Assembler module generates the `.sof` file required for device programming at the end of full compilation. You can use the **Programming File Generator** to generate programming files for alternative device programming methods, such as the `.jic` for flash programming, or `.rpd` for third-party programmer configuration.

Note: If you are generating an `.rpd` for remote system update (RSU), you must follow the instructions in Generating an Application Image on page 187 in the *Remote System Update* chapter. This procedure generates flash programming files for Agilex 5 devices.

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

1. Click **File ➤ Programming File Generator** .
2. For **Device Family** select Agilex 5
3. In the **Configuration mode**, select **Active Serial x4**.
4. Specify the **Output directory** and **Name** for the file you generate.
5. On the **Output Files** tab, enable the checkbox to select the files you want to generate. The AS scheme supports the **Programmer Object File (.pof)**, **JTAG Indirect Configuration File (.jic)**, and **Raw Programming Data File (.rpd)** file types.

Figure 52. Programming File Generator Output Files

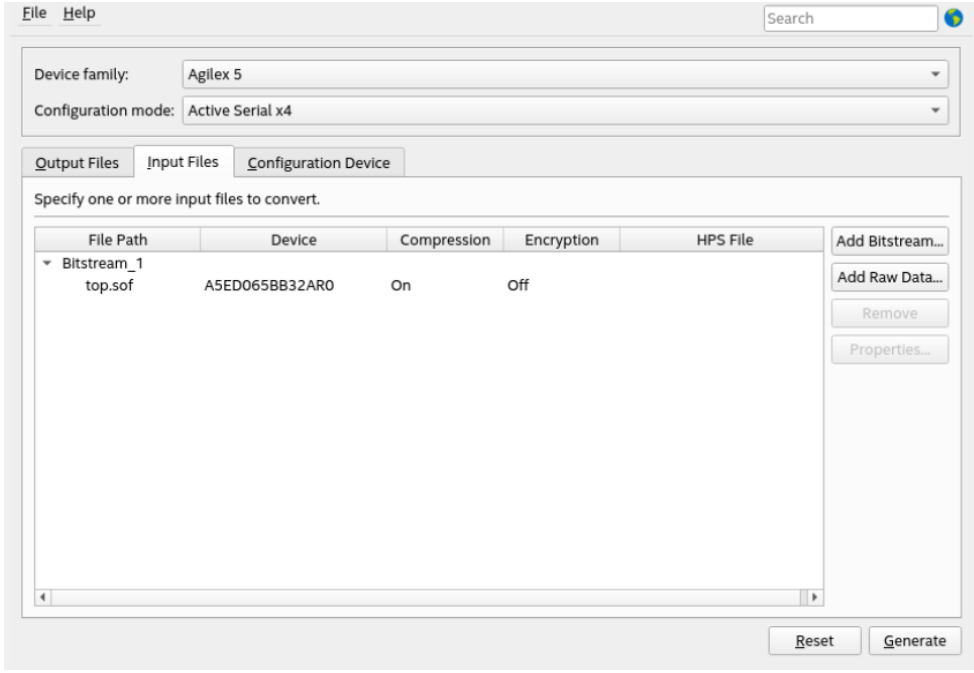


6. For the **JTAG Indirect Configuration File (.jic)** and **Programmer Object File (.pof)** you can turn on the **Memory Map File (.map)**. This option describes flash memory address locations.

The **Input Files** tab is now available.

7. On the **Input Files** tab, click **Add Bitstream** and browse to your configuration bitstream.

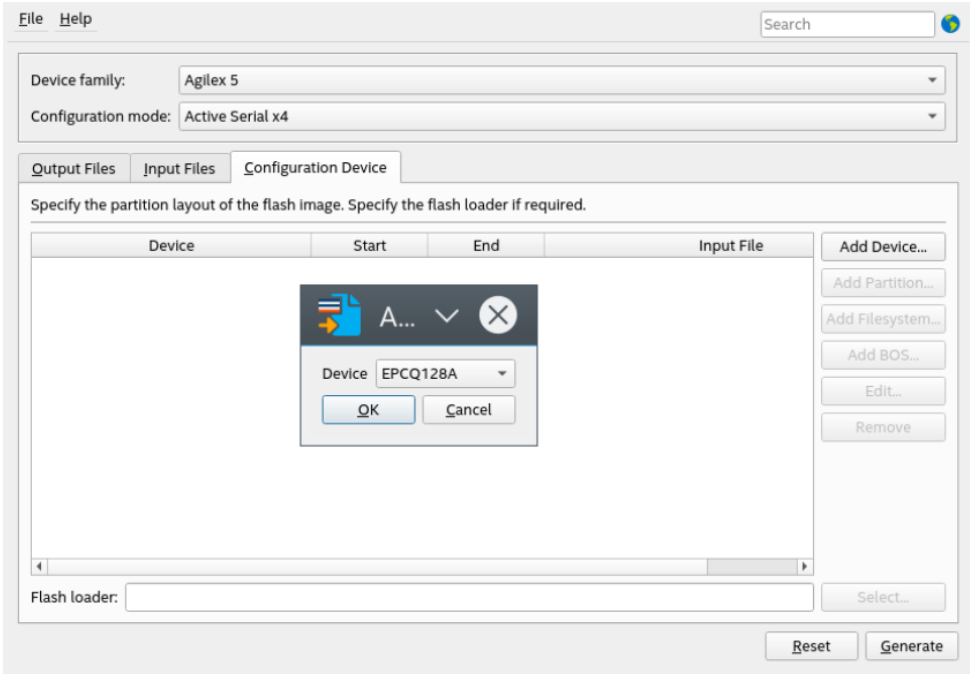
Figure 53. Programming File Generator Input Files



8. On the **Configuration Device** tab, click **Add Device**. You can select your flash device 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 Quartus Prime Pro Edition User Guide: Programmer.

   Note: When using several different Serial NOR Flash devices from different vendors, a single `.jic` flash image can be reused. Create the `.jic` flash image file by selecting any supported Serial NOR Flash, preferably one with the smallest size that you are considering in your design. If the other Serial NOR Flash devices you are considering in your design are also in the supported devices list and they are of equal or larger size than what you initially selected, the image can be reused.

Figure 54. Programming File Generator: Configuration Device Tab



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 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(s).
   a. 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 use this file to regenerate the programming file by running the following command:

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

Related Information

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. Agilex 5 devices using AS mode, try to load a firmware section from addresses 0, 512k, 1024k and 1536k 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.

When the device powers on with MSEL set to JTAG, the SDM tristates the AS pins, AS_CLK, AS_nRST, AS_DATA0-AS_DATA3, and AS_nCS00-AS_nCS03. If MSEL is either AS fast or normal, the SDM drives the AS pins until you power cycle the Agilex 5 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, you must ramp all power supplies to the recommended operating condition within 10 ms. This ramp-up requirement ensures that the AS x4 device is within its operating voltage range when the Agilex 5 device begins accessing the AS x4 device.

When using AS fast mode, all power supplies to the Agilex 5 device must be fully ramped-up to the recommended operating conditions before the SDM releases from reset. To meet the PCIe power-up-to-active time requirement for CvP, all the power supplies to the Agilex 5 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 design meets the power-supply ramp requirements for fast AS mode. If using fast mode, all power supplies must ramp to the recommended operating condition within 10 ms.
- Ensure that the flash is powered up and ready to be accessed when the Agilex 5 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 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 QSPI 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 AS Programming Using Quartus Prime or Third-Party Programmer 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 the QSPI flash device through the AS or JTAG header.
  - If you are programming the flash device using the JTAG header as shown in Programming Your Serial Configuration Device Using JTAG, force the nCONFIG signal to low. When nCONFIG is low, the image cannot load from the QSPI flash device. Then, update the image in quad serial flash through the JTAG header.
- If you are using AS x4 flash memories with AS Fast mode, you must ramp up all power supplies to the recommended operating condition within 10 ms. This ramp-up requirement ensures that the QSPI flash device is within its operating voltage range when the Agilex 5 device begins to access it.
- Check endianness of the `.rpd` if using a third-party programmer to program the QSPI flash device. You should generate the `.rpd` as big endian.
- If you are using the OSC_CLK_1 clock source for configuration, ensure OSC_CLK_1 is free running and stable before SDM starts to load a bitstream from the Quad SPI device. The SDM starts configuration after the device exits the POR state if nCONFIG is held high.
- Try a lower AS clock frequency setting.
- If you drive nCONFIG signal using the external host, ensure it remains high during the AS x4 configuration scheme.
- If you pulse nCONFIG low for reconfiguration, ensure that the nSTATUS acknowledges nCONFIG. If nSTATUS does not follow the nCONFIG signal, the FPGA may not exit power on reset state. You may need to power cycle the PCB.
- Ensure no external component drives the nSTATUS signal low during the power up.

3.3. JTAG Configuration

JTAG-chain device programming is ideal during development. JTAG-chain device configuration uses the JTAG pins to configure the Agilex 5 FPGA directly with the `.sof/.rbf` file. Configuration using the JTAG device chain allows faster development because it does not require you to program 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 Quartus Prime software generates a `.sof/.rbf` file containing the FPGA design information. You can use the `.sof/.rbf` file with a JTAG programmer to configure the Agilex 5 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 Jam STAPL Format File (`.jam`) or Jam Byte Code File (`.jbc`) for JTAG configuration. After the JTAG configuration, the host executes `CONFIG_STATUS` SDM command to ensure the configuration is successful.

Agilex 5 devices automatically compress the configuration bitstream. You cannot disable compression in Agilex 5 devices.

Table 40. Agilex 5 Configuration Data Width, Clock Rates, and Data Rates

Mbps is an abbreviation for Megabits per second.

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

Note: The JTAG port has the highest priority and overrides the `MSEL` pin settings. Consequently, you can configure the Agilex 5 device over JTAG even if the `MSEL` pin specify a different configuration scheme unless you disabled JTAG for security reasons.

Table 41. Power Rails for the Agilex 5 Device Configuration Pins

Configuration Function	Pin Type	Direction	Powered by
`TCK`	Fixed	Input	V_{CCIO_SDM}
`TDI`[19]	Fixed	Input	V_{CCIO_SDM}
`TMS`[19]	Fixed	Input	V_{CCIO_SDM}
`TDO`[19]	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	Input	V_{CCIO_SDM}

Related Information

- Programming Support for Jam STAPL Language
- Agilex 5 FPGAs and SoCs Device Data Sheet

(19) The JTAG pins can access the HPS JTAG chain in Agilex 5 SoC devices.

- Pin-Out Files for Intel FPGAs

3.3.1. 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 55. JTAG Configuration Scheme

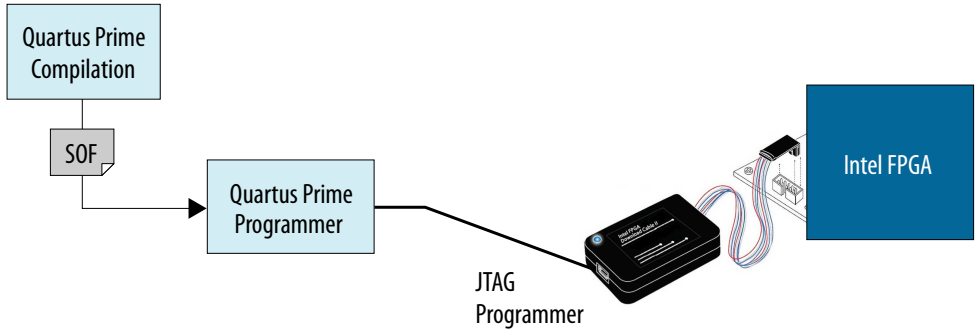
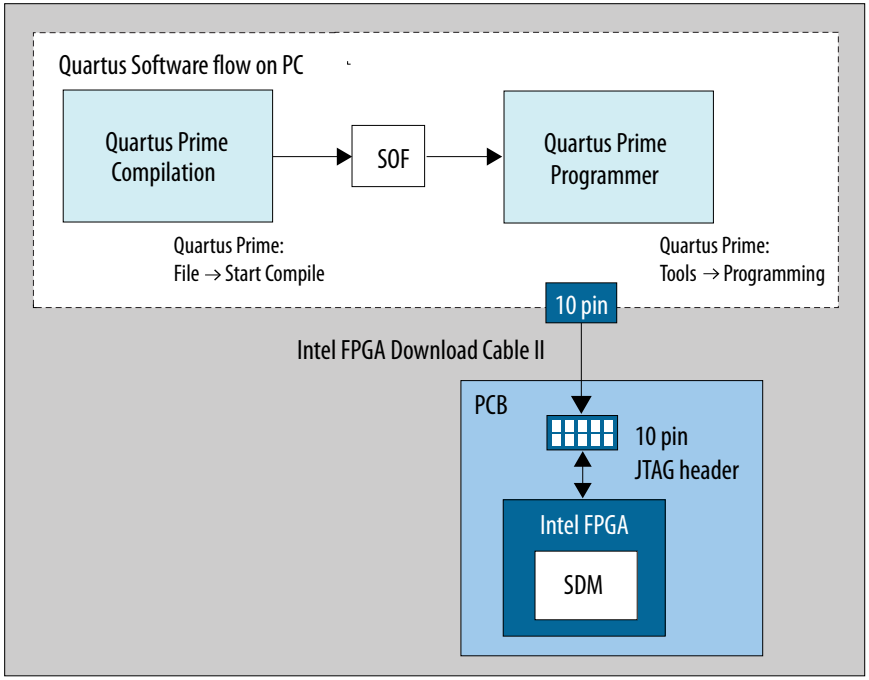


Figure 56. Components and Design Flow for JTAG Programming



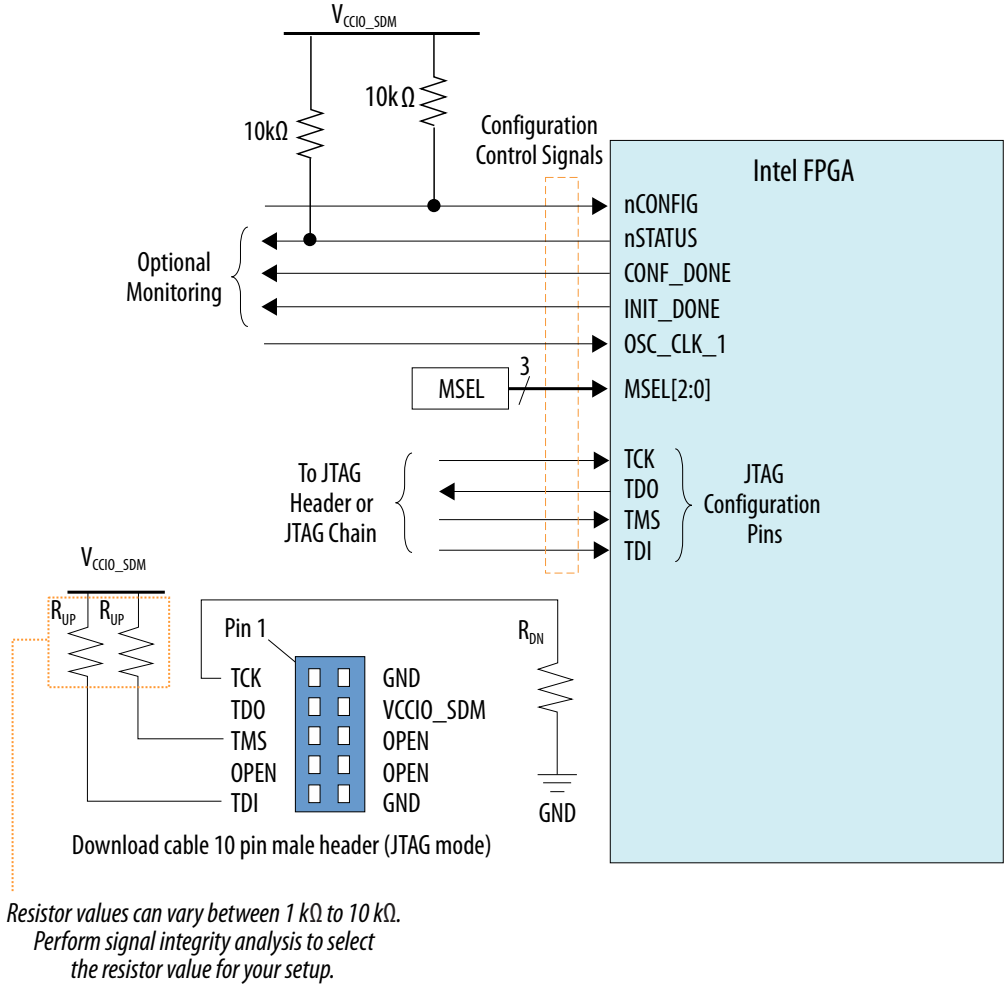
3.3.2. JTAG Device Configuration

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 Agilex 5 device through JTAG using a download cable or a microprocessor.

3.3.2.1. JTAG Single-Device Configuration using Download Cable Connections

Figure 57. Connection Setup for JTAG Single-Device Configuration using Download Cable



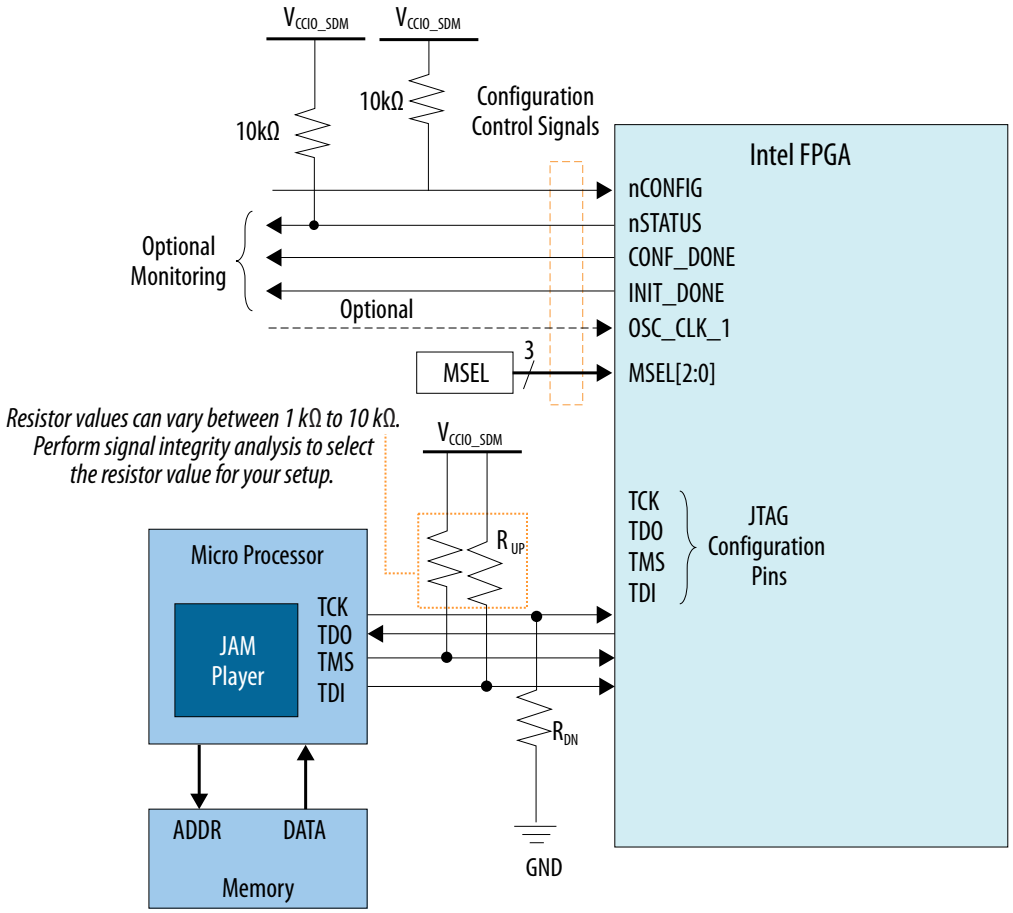
Related Information

Intel FPGA Download Cable II User Guide

3.3.2.2. JTAG Single-Device Configuration using a Microprocessor

Refer to the *Agilex 5 Device Family Pin Connection Guidelines* for additional information about individual pin usage and requirements.

Figure 58. Connection Setup for JTAG Single-Device Configuration using a Microprocessor



Related Information

Pin Connection Guidelines: Agilex 5 FPGAs and SoCs

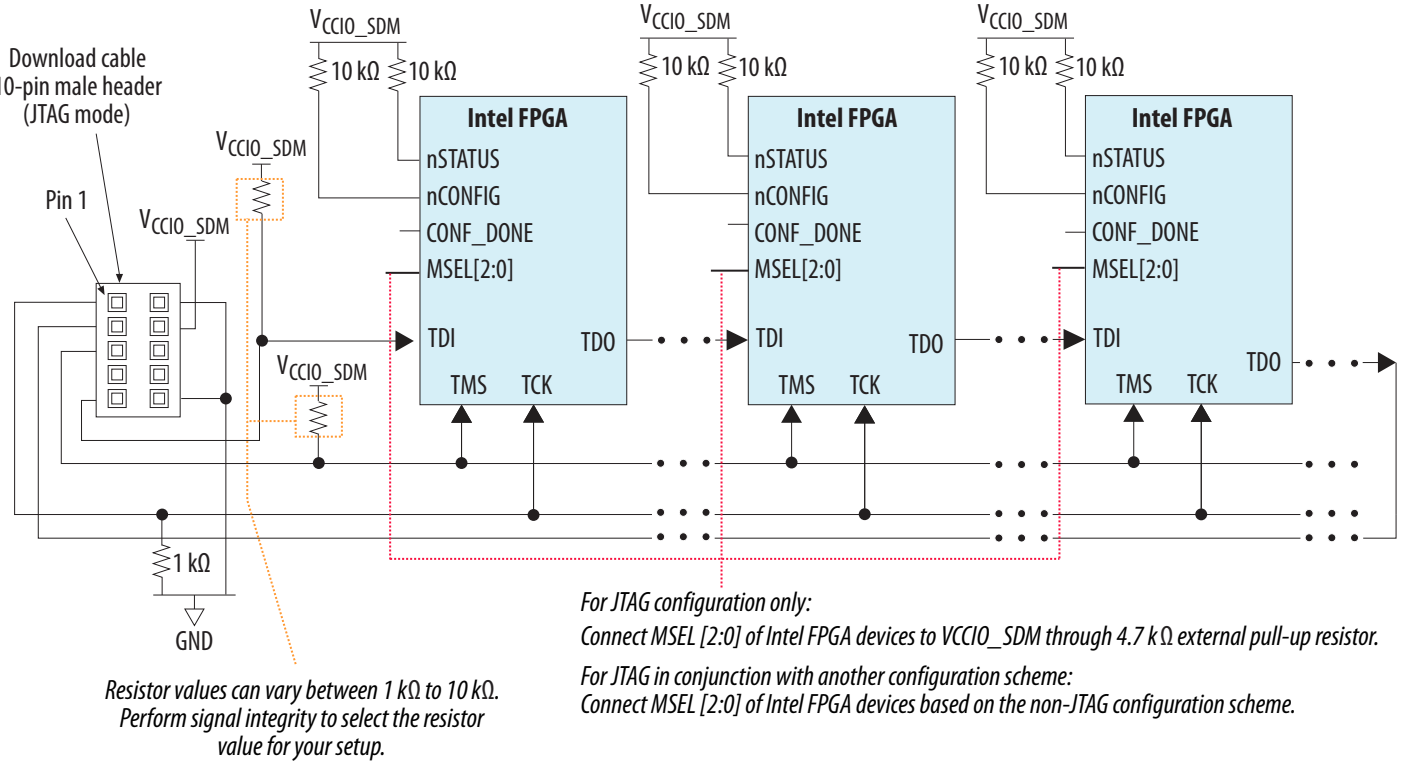
3.3.3. JTAG Multi-Device Configuration

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.

3.3.3.1. JTAG Multi-Device Configuration using Download Cable

Figure 59. Connection Setup for JTAG Multi Device Configuration using Download Cable



3.3.4. 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, nSTATUS 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 Quartus Prime Pro Edition Programmer generates for error reporting.

Note: For Agilex 5 devices that support HPS, 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 does not change state during JTAG configuration.
- Another way to determine whether the device has exited the POR state is to use the Quartus Prime Programmer to detect the device. If the programmer can detect the Agilex 5 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 Agilex 5 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 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.

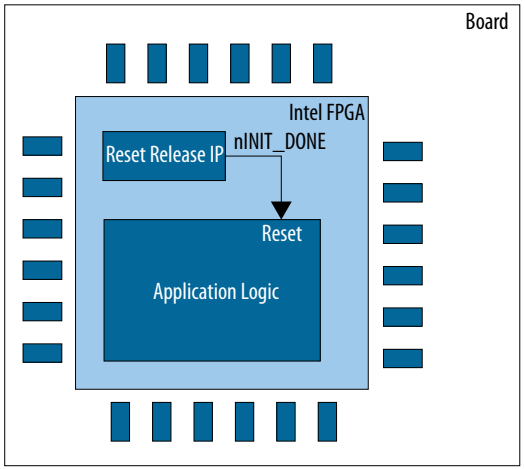
- When the MSEL setting on the PCB is not JTAG, if you use the JTAG interface for reconfiguration after an initial reconfiguration using AS or the Avalon-ST interface, the `.sof` must be in the file format you specified in the Quartus Prime project. For example, if you initially configure the MSEL pins for AS configuration and configure using the AS scheme, a subsequent JTAG reconfiguration using a `.sof` generated for Avalon-ST fails.
- Ensure that during the power up, no external component drives the nSTATUS signal low.
- If the JTAG configuration is failing when MSEL is set to AS configuration mode, erase the QSPI flash device by loading the `.jic` file in the Quartus Prime Programmer without configuring the helper image, start erasing the QSPI flash device, and then power cycle the board before retrying the JTAG configuration.

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

Intel requires that you either use the Reset Release Intel FPGA IP to hold your design in reset until configuration is complete.

The Reset Release Intel FPGA IP is available in the 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 Intel FPGA 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 60. Reset Release Intel FPGA IP nINIT_DONE Internal Connection



View the video guide below for a quick walk-through to understand the importance of using Reset Release Intel FPGA IP and how to include it in your design.

Note: The Reset Release Intel FPGA IP for Agilex 5 uses the component name `intel_user_rst_clkgate`.

Related Information

An Essential Reset for Stratix® 10 and Agilex Devices

4.1. Understanding the Reset Release IP Requirement

Agilex 5 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: If you use multiple Reset Release Intel FPGA IP instances in your design, the nINIT_DONE signals are driven directly from the same source in SDM.

4.2. 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 61. Locate Reset ReleaseIntel FPGA IP in IP Catalog

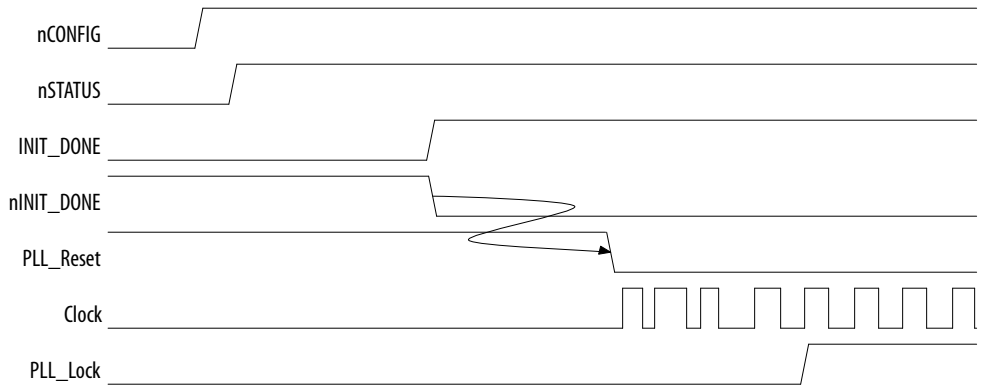
2. Double click the **Reset Release Intel FPGA IP** to add the Reset Release IP to your design.
3. In the **New IP Variant** dialog box, browse to your IP directory and specify a file name for the Reset Release IP. Then click **Create**. The Reset Release IP is now included in your project.

4.3. 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 Agilex 5 device, you should gate the PLL reset input with `nINIT_DONE` as shown the figure.

The `nINIT_DONE` signal, whether driven directly from the Reset Release IP or from an external input through FPGA core logic cells, is unstable during FPGA configuration. Any logic that uses `nINIT_DONE` to control the reset should be designed to tolerate re-assertion during the FPGA configuration.

Figure 62. 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)`.

4.4. 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 reconfigurations. 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 *Quartus Prime Pro Edition User Guide: Partial Reconfiguration*.

Related Information

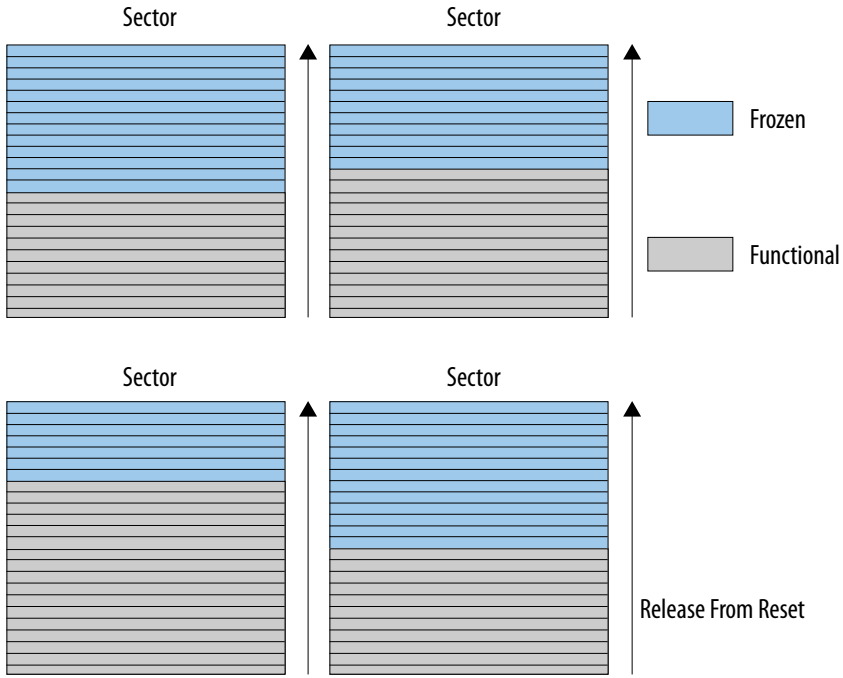
Creating a Partial Reconfiguration Design

4.5. 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 63. 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 Agilex 5 device in reset until entire fabric enters user mode.

4.5.1. 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 enable and initialize user registers in the LABs, DSP, and embedded memory blocks.
4. 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.
5. The FPGA is now in user mode and ready for operation.

4.5.2. Embedded Memory Block Initial Conditions

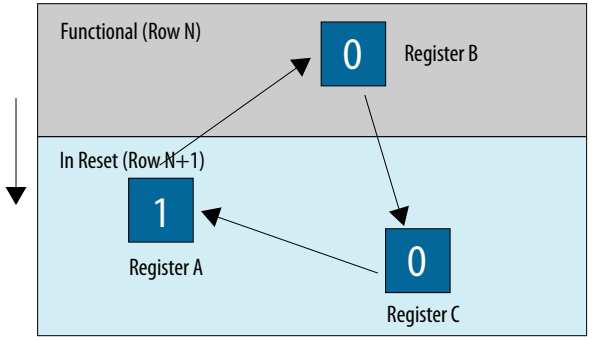
Initialized content of embedded memory blocks is stable during configuration. However, designs that contain logic to modify embedded memory can result in spurious writes. Spurious writes can occur if you fail to gate the write enable with an appropriate reset.

4.5.3. 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.

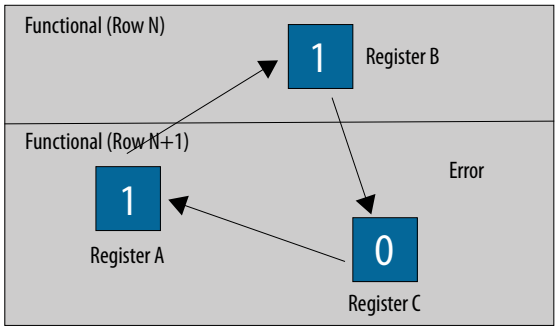
The following example shows how an inadequate reset strategy might result in an illegal state in a one-hot state machine. In this example, the design does not reset any of the state machine registers. The state machine design depends on registers entering an initial state. Without an adequate reset, 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 64. 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 65. 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)

Agilex 5 devices support the RSU feature to allow you to update the FPGA image and reconfigure the device remotely. 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 reconfigure the FPGA 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.

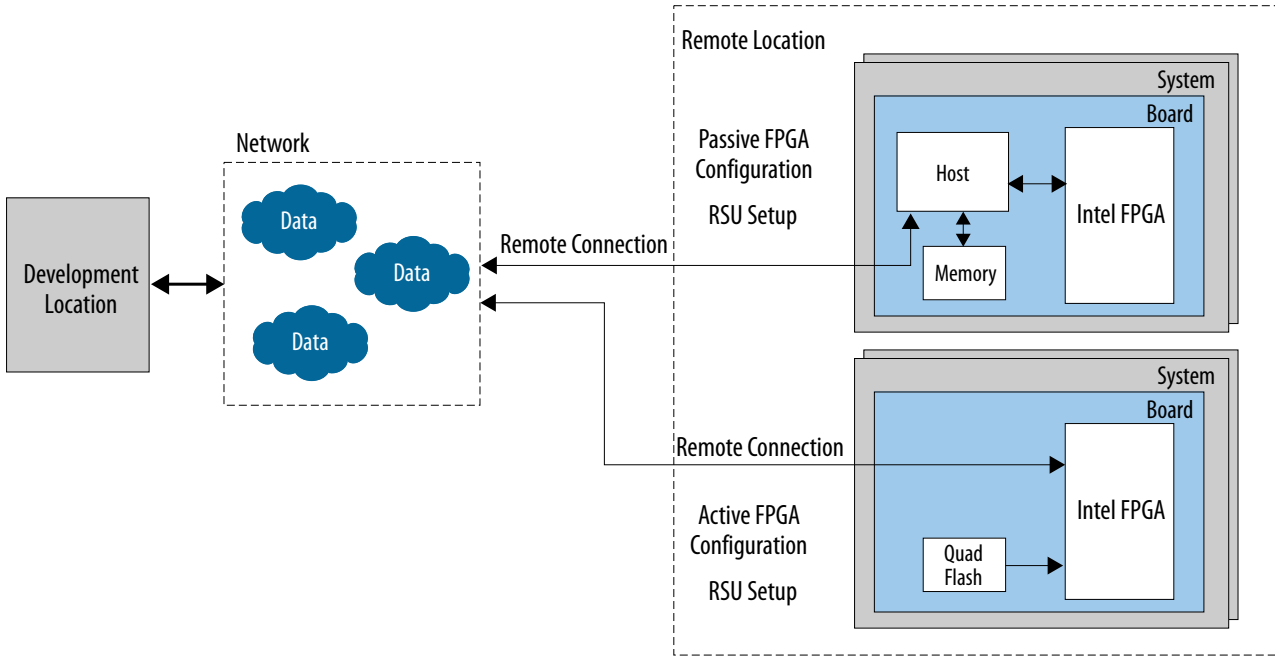
You can initiate reconfiguration by sending an SDM command to the Mailbox Client Intel FPGA IP in your logic design. The RSU performs configuration error detection during and after the reconfiguration process. If errors in the application image occur, the SDM firmware loads the next application image, or revert to factory image if next application image is not available and provides error status information.

This chapter explains the remote system update implementation for active configuration schemes. The FPGA drives the RSU. For the Agilex 5 SoC devices, HPS can drive the RSU process.

For passive configuration schemes, an external host implements remote system update rather than the Agilex 5 device. The external host manages the configuration image and initiate FPGA reconfiguration with the new image as needed.

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

Figure 66. Typical Remote System Update Process



5.1. Remote System Update Functional Description

5.1.1. RSU Glossary

Table 42. 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 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 address to this block, 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 backup application image that the RSU loads when all attempts to load an application image fail. The factory image should provide enough functionality for the device to recover when all application images are corrupt. Once the factory image loads, you can program new application images to replace the failing images.

continued...

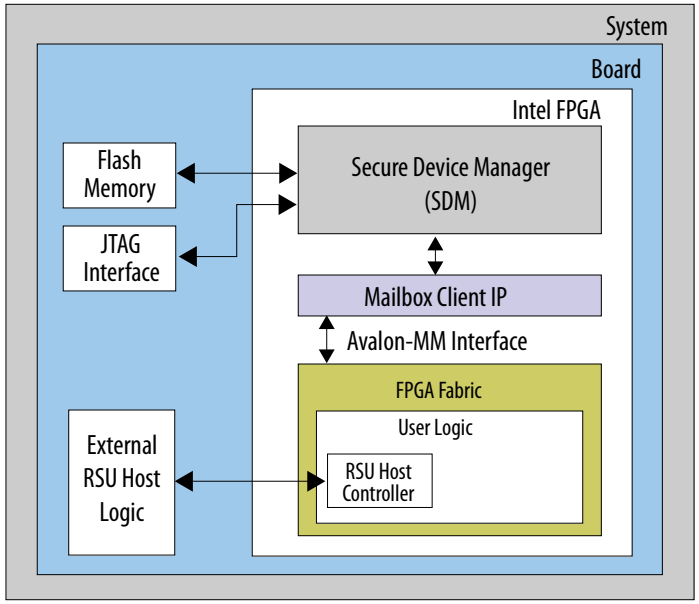
Term	Meaning
	The SDM loads the factory image in the following circumstances: • You assign the `Direct to Factory Image` function to an SDM I/O pin and assert the pin after a power-on reset or `nCONFIG` deassertion. • All SDM attempts to load the application images fail. • You make a request to the SDM to load the factory image. The configuration system treats the factory image in the same way as it does an application image.
Initial RSU 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-related items in flash: • The factory image • The decision firmware • The decision firmware data Intel recommends that your factory update image include the minimum amount of logic necessary to debug your design if your application image or images fail to load.

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® V 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 a Mailbox Client Intel FPGA IP as shown in the figure below. The Mailbox Client Intel FPGA IP sends and receives remote system update operation commands and responses, such as `QSPI_READ` and `QSPI_WRITE`.

Figure 67. Agilex 5 Remote System Update Components



5.1.3. Remote System Update Configuration Images

Agilex 5 devices using remote system update require the following configuration images:

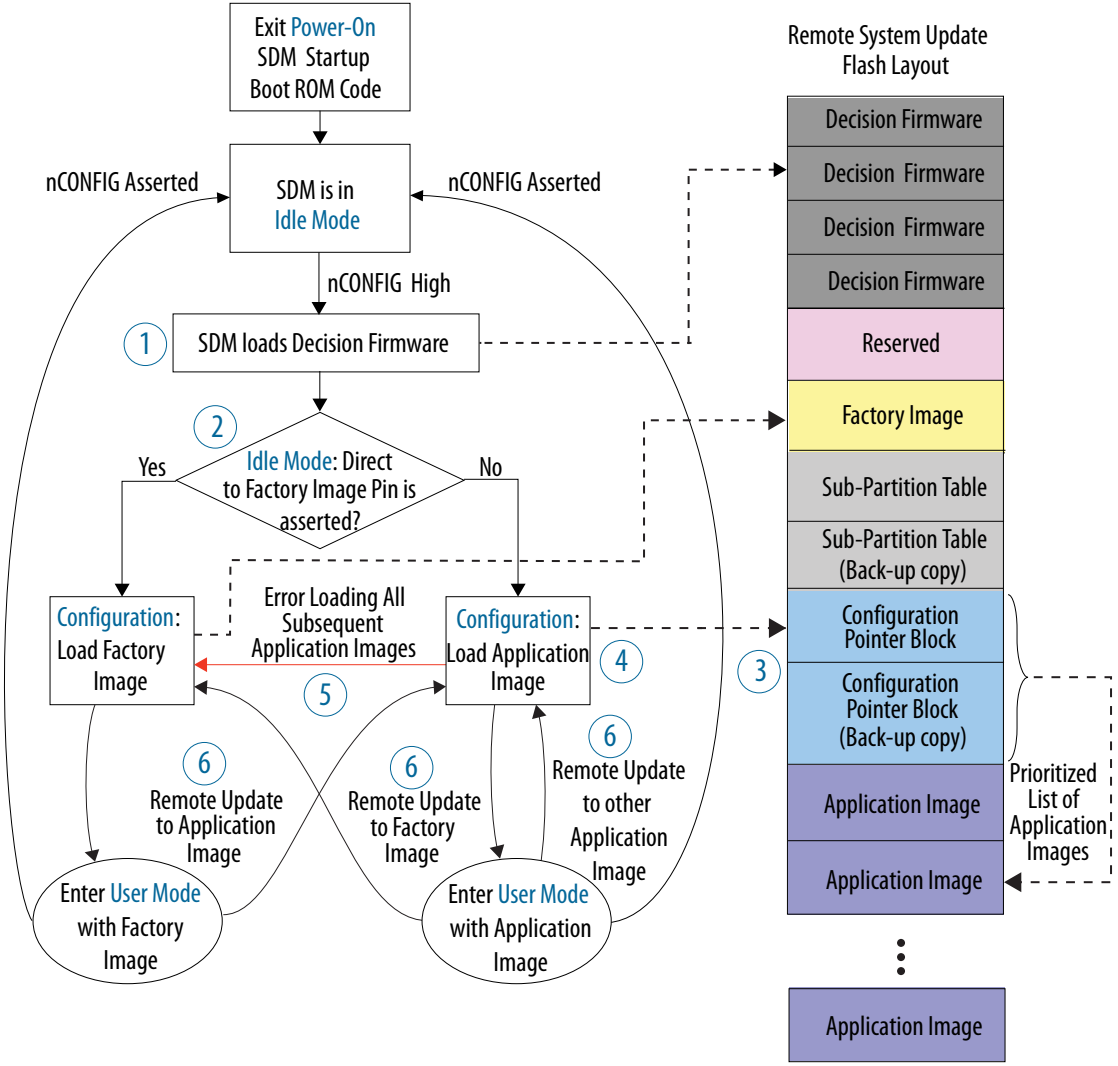
- Factory image—This image includes logic to implement the following functions:
  - Your design-specific logic to obtain new application images
  - Your design-specific logic to request reconfiguration using a specific application image
  - Image storage in flash memory
- Application image—contains logic 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, Agilex 5 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 Intel FPGA IP with the device in user mode.

5.1.4. Remote System Update Configuration Sequence

Figure 68. Remote System Update Configuration Sequence

In the following figure the blue text are states shown in the Configuration Flow Diagram on page 23.



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. 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 158 for step-by-step details for programming the initial RSU flash image into the flash.)

   If you trigger the reconfiguration with a pulse of nCONFIG low, the SDM loads the decision firmware.
2. The optional Direct to Factory Image pin controls whether the SDM firmware loads the factory or application image. You can assign the Direct to Factory Image 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 images load successfully, the SDM loads the factory image.

   Note: 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.

   Note: For every unsuccessful configuration, the nSTATUS asserts a low pulse to indicate configuration failure and the SDM proceeds to load the next image automatically, do not assert nCONFIG low to attempt to load the next application image.
6. During the remote update to other application image or factory image, the SDM loads the decision firmware.

Note: You must keep nCONFIG high until the device enters the user mode.

- Keep the nCONFIG signal high after the device powers up and throughout the entire device configuration to load an application or factory image.
- Keep the nCONFIG signal high during the remote update to other application image or factory image by using the RSU_IMAGE_UPDATE command.

You drive nCONFIG low only when the device is in user mode to trigger the reconfiguration.

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 first failing image.

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:

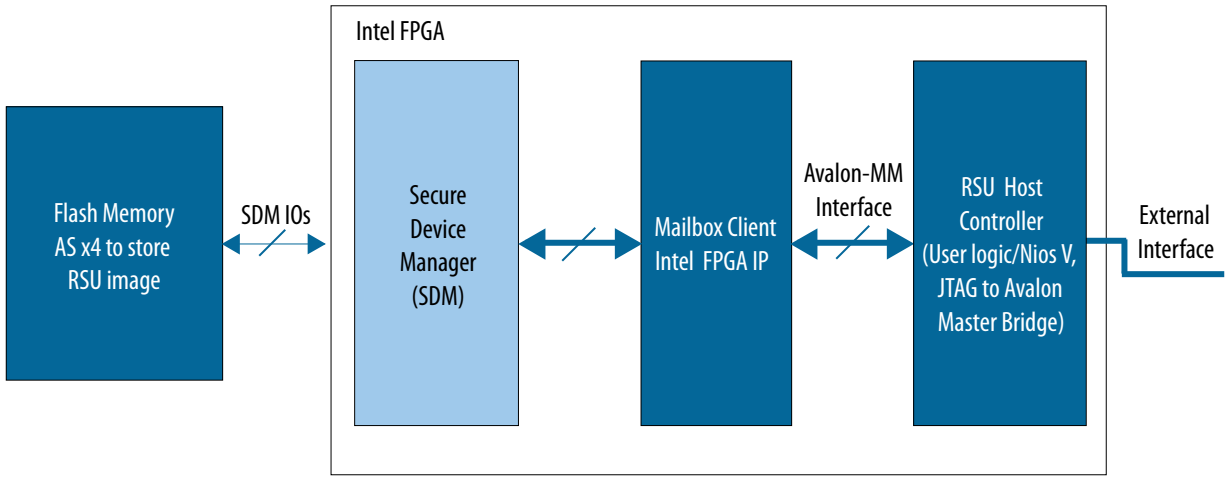
- If there are vulnerabilities in the firmware
- If there are errors in the firmware or in the factory image

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 using one of the following methods:
   - Trigger reconfiguration using the RSU_IMAGE_UPDATE command. This method does not require you to update CPB0 and CPB1.
   - Trigger reconfiguration by pulsing the nCONFIG signal or power cycling the device. This method requires you to write the factory update image start address in both CPB0 and CPB1. After factory image updates, the SDM automatically removes the address from CPB0 and CPB1.
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 update image removes itself from the flash and its address pointer from the CPB, then the SDM loads the updated decision firmware followed by 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 69. Agilex 5 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 V 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 the Mailbox Client Intel FPGA IP. The Mailbox Client Intel FPGA IP functions as the messenger between the remote system update host and the SDM. It passes the commands to and responses from the SDM.
2. The initial RSU 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 duplicate copy of factory 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 182 for the step by step process to generate the initial RSU image and single application 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 initial RSU 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

Related Information

Mailbox Client Intel FPGA IP User Guide

5.3. Commands and Responses

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

The first word of the command and response packets is a header that provides basic information about the command or response.

Block Diagram

The following figure illustrates the role of the Mailbox Client Intel FPGA IP in a Agilex 5 design. The Mailbox Client Intel FPGA IP enables communication with the SDM to access quad SPI flash memory and system status.

Figure 70. Role of the Mailbox Client Intel FPGA IP

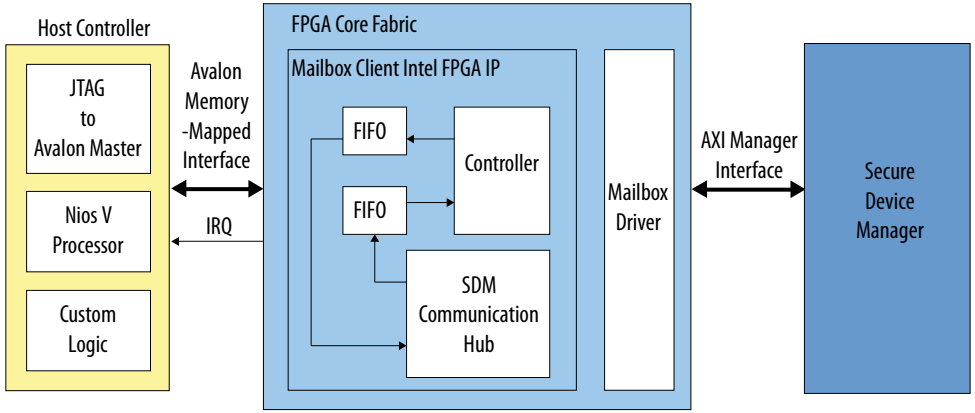
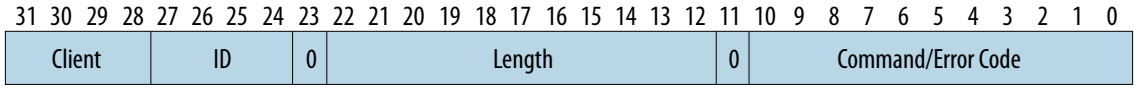


Figure 71. Command and Response Header Format in Agilex 5



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

The following table describes the fields of the header command.

Table 43. Command and Response Header Description

Header	Bit	Description
CLIENT	[31:28]	The client ID. The response header returns the client ID specified in the command header.
ID	[27:24]	The command ID. The response header returns the ID specified in the command header. Refer to *Operation Commands* for command descriptions.
0	[23]	Reserved.
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.

continued...

Header	Bit	Description
		If there is a mismatch between the command length specified in the command header and the number of words sent. The IP raises bit 3 of the Interrupt Status Register (COMMAND_INVALID) and the Mailbox Client Intel FPGA IP must be reset.
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. If the command succeeds, the `Error Code` is 0. If the command fails, refer to the error codes defined in the *Error Code Responses*.

5.3.1. Operation Commands

Resetting Quad SPI Flash

Important: For Agilex 5 devices, you must connect the serial flash or quad SPI flash reset pin to the AS_nRST pin. The SDM must fully control the QSPI reset. Do not connect the quad SPI reset pin to any external host.

RSU SDM Command Use Case

Important: All RSU-related SDM commands (`RSU_IMAGE_UPDATE`, `RSU_GET_SPT`, `RSU_STATUS`, and `RSU_NOTIFY`) are only valid when the SDM loads the RSU image from QSPI flash using AS x4 configuration mode.

Table 44. Command List and Description

Command	Code (Hex)	Command Length [20]	Response Length [20]	Description
`RSU_IMAGE_UPDATE`	5C	2	0	Triggers reconfiguration from the data source that 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. When sending the argument to the IP, you first send bits [31:0] followed by bits [63:32]. If you do not provide this argument its value is assumed to be 0. • Bit [31:0]: The start address of an application image. • Bit [63:32]: Reserved (write as 0).

continued...

(20) This number does not include the command or response header.

<table>
<tr><th>Command</th><th>Code (Hex)</th><th>Command Length [20]</th><th>Response Length [20]</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td></td><td>Once the device processes this command, it returns the response header to response FIFO before it proceeds to reconfigure the device. Ensure the host PC or host controller stops servicing other interrupts and focuses on reading the response header data to indicate the command completed successfully. Otherwise, the host PC or host controller may not be able to receive the response once the reconfiguration process started.
Once the device proceeds with reconfiguration, the link between the external host and FPGA is lost. If you use PCIe in your design, you need to re-enumerate the PCIe link.
Important: When resetting quad SPI, you must follow instructions specified in Resetting Quad SPI Flash on page 161.</td></tr>
<tr><td>RSU_GET_SPT</td><td>5A</td><td>0</td><td>4</td><td>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:
<table>
<tr><th>Word</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>SPT0[63:32]</td><td rowspan="2">SPT0 address in quad SPI flash.</td></tr>
<tr><td>1</td><td>SPT0[31:0]</td></tr>
<tr><td>2</td><td>SPT1[63:32]</td><td rowspan="2">SPT1 address in quad SPI flash.</td></tr>
<tr><td>3</td><td>SPT1[31:0]</td></tr>
</table></td></tr>
<tr><td>CONFIG_STATUS</td><td>4</td><td>0</td><td>6</td><td>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:
<table>
<tr><th>Word</th><th>Summary</th><th>Description</th></tr>
<tr><td>0</td><td>State</td><td>Describes the most recent configuration related error. Returns 0 when there are no configuration errors.
The error field has 2 fields:
• Upper 16 bits: Major error code.
• Lower 16 bits: Minor error code.
Refer to Appendix: CONFIG_STATUS and RSU_STATUS Error Code Descriptions in the Mailbox Client Intel FPGA IP User Guide for more information.</td></tr>
<tr><td>1</td><td>Quartus Version</td><td>Available in Quartus Prime software versions between 19.4 and 21.2, the field displays:</td></tr>
</table></td></tr>
</table>

continued...

(20) This number does not include the command or response header.

Command	Code (Hex)	Command Length [20]	Response Length [20]	Description		
						• Bit [31:28]: Index of the firmware or decision firmware copy that was used the most recently. Possible values are 0, 1, 2, and 3. • Bit [27:24]: Reserved • Bit [23:16]: Value is '0' Available in Quartus Prime software version 21.3 or later, the Quartus version displays: • Bit [31:28]: Index of the firmware or decision firmware copy that was used the most recently. Possible values are 0, 1, 2, and 3. • Bit [27:24]: Reserved • Bit [23:16]: Major Quartus release number • Bit [15:8]: Minor Quartus release number • Bit [7:0]: Quartus update number For example, in Quartus Prime software version 21.3.1, the following values represent the major and minor Quartus release numbers, and the Quartus update number: • Bit [23:16] = 8'd21 = 8'h15 • Bit [15:8] = 8'd3 = 8'h3 • Bit [7:0] = 8'd1 = 8'h1
				2	Pin status	• Bit [31]: Current nSTATUS output value (active low) • Bit [30]: Detected nCONFIG input value (active low) • Bit [29:8]: Reserved • Bit [7:6]: Configuration clock source — 01 = Internal oscillator — 10 = OSC_CLK_1 • Bit [5:3]: 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

continued...

(20) This number does not include the command or response header.

<table>
<tr><th>Command</th><th>Code (Hex)</th><th>Command Length [20]</th><th>Response Length [20]</th><th colspan="3">Description</th></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td>• Bit [3]: SEU_ERROR
• Bit [2]: CVP_DONE
• Bit [1]: INIT_DONE
• Bit [0]: CONF_DONE</td></tr>
<tr><td></td><td></td><td></td><td></td><td>4</td><td>Error location</td><td>Contains the error location. Returns 0 if there are no errors.</td></tr>
<tr><td></td><td></td><td></td><td></td><td>5</td><td>Error details</td><td>Contains the error details. Returns 0 if there are no errors.</td></tr>
<tr><td>RSU_STATUS</td><td>5B</td><td>0</td><td>9</td><td colspan="3">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:</td></tr>
<tr><td></td><td></td><td></td><td></td><td>Word</td><td>Summary</td><td>Description</td></tr>
<tr><td></td><td></td><td></td><td></td><td>0-1</td><td>Current image</td><td>Flash offset of the currently running application image.</td></tr>
<tr><td></td><td></td><td></td><td></td><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 AS x4, does not clear this field. Information about failing image only updates when the Mailbox Client Intel FPGA IP receives a new RSU_IMAGE_UPDATE command and successfully configures from the update image.</td></tr>
<tr><td></td><td></td><td></td><td></td><td>4</td><td>State</td><td>Failure code of the failing image.
The error field has two parts:
• Bit [31:16]: Major error code
• Bit [15:0]: Minor error code
Returns 0 for no failures. Refer to Appendix: CONFIG_STATUS and RSU_STATUS Error Code Descriptions in the Mailbox Client Intel FPGA IP User Guide for more information.</td></tr>
</table>

continued...

[20] This number does not include the command or response header.

Command	Code (Hex)	Command Length [20]	Response Length [20]	Description		
				5	Version	RSU interface version and error source. For more information, refer to RSU Status and Error Codes section in the *Hard Processor System Remote System Update User Guide*.
				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 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 Quartus Prime Pro Edition software.
`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 Quartus Prime Pro Edition software.		
`QSPI_OPEN`	32	0	0	Requests exclusive access to the quad SPI. You issue this request before any other QSPI requests. 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. Upon receiving the OK response, issue the `QSPI_SET_CS` command to select the flash devices. 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.		

continued...

(20) This number does not include the command or response header.

Command	Code (Hex)	Command Length [20]	Response Length [20]	Description
				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 AS x4 configuration. For the Avalon streaming interface (Avalon ST) configuration scheme, you must connect QSPI flash memories to GPIO pins. Access to the quad SPI flash memory devices via any mailbox client IP is not available by default in designs that include the HPS, unless you disable the QSPI in HPS software configuration. *Important:* When resetting quad SPI, you must follow instructions specified in Resetting Quad SPI Flash on page 161.
QSPI_CLOSE	33	0	0	Closes the exclusive access to the quad SPI interface. *Important:* When resetting quad SPI, you must follow instructions specified in Resetting Quad SPI Flash on page 161.
QSPI_SET_CS	34	1	0	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. Below values correspond to the respective nCSO[3:0] pins. — Value 4'h0000 selects the flash that corresponds to nCSO[0]. — Value 4'h0001 selects the flash that corresponds to nCSO[1]. — Value 4'h0002 selects the flash that corresponds to nCSO[2]. — Value 4'h0003 selects the flash that corresponds to nCSO[3]. • Bits[27:0]: Reserved (write as 0). *Note:* Agilex 5 devices support one AS x4 flash memory device for AS configuration from quad SPI device connected to nCSO[0]. Once the device enters user mode, you can then use up to four AS x4 flash memories with Mailbox Client Intel FPGA IP or HPS as data storage. The Mailbox Client Intel FPGA IP or HPS can use nCSO[3:0] to access quad SPI devices. During AS x4 configuration scheme in Agilex 5 devices, this command is required after every QSPI_OPEN command. During JTAG configuration scheme, this command is required after every QSPI_OPEN command. *Important:* When resetting quad SPI, you must follow instructions specified in Resetting Quad SPI Flash on page 161.
QSPI_READ	3A	2	N	Reads the attached quad SPI device. The maximum transfer size is 4 kilobytes (KB) or 1024 words. Takes two arguments: *continued...*

(20) This number does not include the command or response header.

Command	Code (Hex)	Command Length [20]	Response Length [20]	Description
				• 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. *Important:* When resetting quad SPI, you must follow instructions specified in Resetting Quad SPI Flash on page 161.
QSPI_WRITE	39	2+N	0	Writes data to the quad SPI device. The maximum transfer size is 4 kilobytes (KB) or 1024 words. Takes three arguments: • The flash address offset (one word). The write address must be word aligned. • 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. To prepare memory for writes, use the QSPI_ERASE command before issuing this command. *Note:* You cannot run the QSPI_WRITE command while device configuration is in progress. *Important:* When resetting quad SPI, you must follow instructions specified in Resetting Quad SPI Flash on page 161.
QSPI_ERASE	38	2	0	Erases a 4/32/64 KB sector of the quad SPI device. Takes two arguments: • The flash address offset to start the erase (one word). Depending on the number of words to erase, the start address must be: — 4 KB aligned if number words to erase is 0x400 — 32 KB aligned if number words to erase is 0x2000 — 64 KB aligned if number words to erase is 0x4000 Returns an error for non-4/32/64 KB aligned addresses. • The number of words to erase is specified in multiples of: — 0x400 to erase 4 KB (100 words) of data. This option is the minimum erase size. — 0x2000 to erase 32 KB (500 words) of data — 0x4000 to erase 64 KB (1000 words) of data A successful erase returns the OK response code. *Important:* When resetting quad SPI, you must follow instructions specified in Resetting Quad SPI Flash on page 161.

continued...

(20) This number does not include the command or response header.

Command	Code (Hex)	Command Length [20]	Response Length [20]	Description
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. The read data return is in multiple of 4 bytes. If the bytes to read is not an exact multiple of 4 bytes, it is padded with multiple of 4 bytes until the next word boundary and the padded bit value is zero. *Important:* When resetting quad SPI, you must follow instructions specified in Resetting Quad SPI Flash on page 161.
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. The command pads the data with zero. *Important:* When resetting quad SPI, you must follow instructions specified in Resetting Quad SPI Flash on page 161.
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. *Important:* When resetting quad SPI, you must follow instructions specified in Resetting Quad SPI Flash on page 161.

For CONFIG_STATUS and RSU_STATUS major and minor error code descriptions, refer to *Appendix: CONFIG_STATUS and RSU_STATUS Error Code Descriptions* in the *Mailbox Client Intel FPGA IP User Guide*.

[20] This number does not include the command or response header.

Related Information

Mailbox Client Intel FPGA IP User Guide: CONFIG_STATUS and RSU_STATUS Error Code Descriptions
For more information about the CONFIG_STATUS and RSU_STATUS error codes.

5.3.2. Error Code Responses

Table 45. 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 currently loaded boot ROM cannot decode or recognize the command code.
3	UNKNOWN_COMMAND	Indicates that the currently loaded firmware cannot decode the command code.
4	INVALID_COMMAND_PARAMETERS	Indicates that the command is incorrectly formatted. For example, the length field setting in header is not valid.
6	COMMAND_INVALID_ON_SOURCE	Indicates that the command is from a source for which it is not enabled.
8	CLIENT_ID_NO_MATCH	Indicates that the Client ID cannot complete the request to close the exclusive access to quad SPI. The Client ID does not match the existing client with the current exclusive access to quad SPI.
9	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 • An invalid chip select value, displaying value of more than 3 • An invalid address in RSU case
A	AUTHENTICATION_FAIL	Indicates the configuration bitstream signature authentication failure.
B	TIMEOUT	This error indicates time out due to the following conditions: • Command • Waiting for QSPI_READ operation to complete • Waiting for the requested temperature reading from one of the temperature sensors. May indicate a potential hardware error in the temperature sensor.
C	HW_NOT_READY	Indicates one of the following conditions: • The hardware is not ready. Can indicate either an initialization or configuration problem. The hardware may refer to quad SPI. • RSU image is not used to configure the FPGA.

continued...

<table>
<tr><th>Value (Hex)</th><th>Error Code Response</th><th colspan="4">Description</th></tr>
<tr><td>D</td><td>HW_ERROR</td><td colspan="4">Indicates that the command completed unsuccessfully due to unrecoverable hardware error.</td></tr>
<tr><td rowspan="4">80 - 8F</td><td rowspan="4">COMMAND_SPECIFIC_ERROR</td><td colspan="4">Indicates a command specific error due to an SDM command you used.</td></tr>
<tr><td>SDM Command</td><td>Error Name</td><td>Error code</td><td>Description</td></tr>
<tr><td>GET_CHIPID</td><td>EFUSE_SYSTEM_FAILURE</td><td>0x82</td><td>Indicates that the eFuse cache pointer is invalid.</td></tr>
<tr><td>QSPI_OPEN/
QSPI_CLOSE/
QSPI_SET_CS/
QSPI_READ_DEVICE_REG/
QSPI_WRITE_DEVICE_REG/
QSPI_SEND_DEVICE_OP/
QSPI_READ</td><td>QSPI_HW_ERROR</td><td>0x80</td><td>Indicates QSPI flash memory error. This error indicates one of the following conditions:
A QSPI flash chip select setting problem
A QSPI flash initialization problem
A QSPI flash resetting problem
A QSPI flash settings update problem</td></tr>
<tr><td>QSPI_ALREADY_OPEN</td><td>0x81</td><td>Indicates that the client's exclusive access to QSPI flash via QSPI_OPEN command is already open.</td></tr>
<tr><td>100</td><td>NOT_CONFIGURED</td><td colspan="4">Indicates that the device is not configured.</td></tr>
<tr><td>1FF</td><td>ALT_SDM_MBOX_RESP_DEVICE_ BUSY</td><td colspan="4">Indicates that the device is busy due to following use cases:
RSU: Firmware is unable to transition to different version due to an internal error.
HPS: HPS is busy when in HPS reconfiguration process or HPS cold reset.</td></tr>
<tr><td>2FF</td><td>ALT_SDM_MBOX_RESP_NO_VALID_RESP_AVAILABLE</td><td colspan="4">Indicates that there is no valid response available.</td></tr>
<tr><td>3FF</td><td>ALT_SDM_MBOX_RESP_ERROR</td><td colspan="4">General Error.</td></tr>
</table>

5.3.3. Error Code Recovery

The table below describes possible steps to recover from an error code. Error recovery depends on specific use case.

Table 46. Error Code Recovery for Known Error Codes

Value	Error Code Response	Error Code Recovery
4	INVALID_COMMAND_PARAMETERS	Resend the command header or header with arguments with corrected parameters.

continued...

Value	Error Code Response	Error Code Recovery
		For example, ensure that the length field setting in header is sent with the correct value.
6	COMMAND_INVALID_ON_SOURCE	Resend the command from valid source such as JTAG, HPS, or core fabric.
8	CLIENT_ID_NO_MATCH	Wait for the client who opened the access to quad SPI to complete its access and then closes the exclusive access to quad SPI.
9	INVALID_ADDRESS	Possible error recovery steps: For QSPI operation: • Send command with a valid chip select. • Send command with a valid QSPI flash address. For RSU: Send command with a valid start address of the factory image or application.
B	TIMEOUT	Possible recovery steps: For QSPI operation: Check signal integrity of QSPI interfaces and attempt command again. For HPS restart operation: Retry to send the command again.
C	HW_NOT_READY	Possible recovery steps: For QSPI operation: • Reconfigure the device via source. • Ensure that IP used to build your design allows access to the QSPI flash. • Check for QSPI_SET_CS commands after every QSPI_OPEN request. For RSU: Configure the device with RSU image.
80	QSPI_HW_ERROR	Check the QSPI interface signal integrity and ensure the QSPI device is not damaged.
81	QSPI_ALREADY_OPEN	Client already opened QSPI. Continue with the next operation.
82	EFUSE_SYSTEM_FAILURE	Attempt reconfiguration or power cycle. If error persists after reconfiguration or power cycle, the device may be damaged and unrecoverable.
100	NOT_CONFIGURED	Send a bitstream that configures the HPS.
1FF	ALT_SDM_MBOX_RESP_DEVICE_ BUSY	Possible error recovery steps: For QSPI operation: Wait for ongoing configuration or other client to complete operation. For RSU: Reconfigure device to recover from internal error. For HPS restart operation: Wait for reconfiguration via HPS or HPS Cold Reset to complete.

5.4. Quad SPI Flash Layout

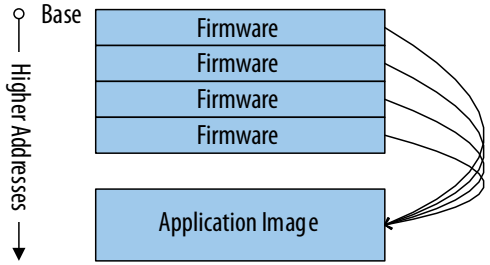
5.4.1. High Level Flash Layout

5.4.1.1. Standard (non-RSU) Image Layout in Flash

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 application image in flash.

Figure 72. Image Layout in Flash: Non-RSU

In this figure, each of the four firmware copies points to the Application Image.



5.4.1.2. RSU Image Layout in Flash – 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 firmware data
- One factory image
- Two configuration pointer blocks (CPBs)

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 `max_retry` parameter value.

The pointer blocks contain a list of application images to try until one of them is successful. If none is successful, the SDM loads the factory image. To ensure reliability, the pointer block includes a main (CPB0) and backup copy (CPB1). The decision firmware first uses the CPB0 pointer block. If SDM detects content corruption, then it uses CPB1 backup copy. You may execute the RSU_STATUS command to identify the CPB0 corruption; in the case of CPB0 corruption, the minor error code of RSU_STATUS displays `0xD010`. To recover the CPB0 corruption, you may erase the CPB0 and copy the content from CPB1.

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.

5.4.1.2.1. Firmware Version Information

The Quartus Prime firmware version is stored as a 32bit value at offset `0x420` in the following:

- Decision firmware (each of the four copies),
- Application images,
- Factory update images,
- Decision firmware update images,
- Combined application images.

The format for the version field is described below:

Table 47. Firmware Version Fields

Bit Field	Bits	Description
version_major	31:24	Major release number for Quartus Prime. Example: set to 20 for release 20.1.
version_minor	23:16	Minor release number for Quartus Prime. Example: set to 1 for release 20.1.
version_update	15:8	Update release number for Quartus Prime. Example: set to 2 for release 20.1.2
reserved	7:0	Set to zero

Note: Quartus Prime patches do not change the firmware version field. All patches have the same firmware version field as the base release on top of which the patches are applied to.

Both U-Boot and LibRSU enable the decision firmware versions to be queried. For examples on querying the decision firmware version for both U-Boot and Linux, refer to the *Remote System Update Example* section. The decision firmware version must first be queried in U-Boot for the value to be available in Linux.

5.4.1.3. RSU Image Layout – Your Perspective

The sub-partition table (SPT) is used for managing the allocation of the quad SPI flash.

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

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

Table 48. Typical Sub-Partitions of the 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

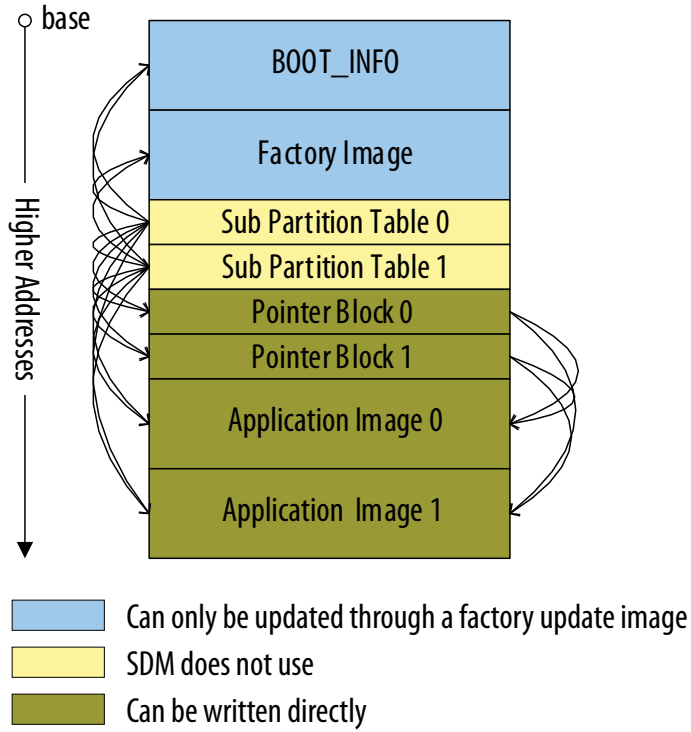
continued...

Sub-partition Name	Contents
CPB1	Pointer block 1
P1	Application image 1
P2	Application image 2

Figure 73. RSU Image Layout - Your Perspective

In this figure:

- SPT0 and SPT1 point to everything:
  - BOOT_INFO
  - Factory Image
  - Pointer Block 0 and Pointer Block 1
  - All Application Images
- Pointer Block 0 and Pointer Block 1 point to all Application Images



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 Image Sub-Partitions Layout

The RSU Image Sub-Partitions Layout table shows the layout of RSU image stored in QSPI flash.

If you anticipate changes to the factory image, you may consider reserving additional memory space. By default, the Quartus Prime Programming File Generator reserves additional 256 kB of memory space for a factory image. To increase the partition size, update the **End Address** value in the **Edit Partition** dialog box window as described in the *Generating the Initial RSU Image*.

Note: If the new factory image has a size bigger than the allocated partition size, it overwrites the SPT and CPB blocks next to it, causing the decision firmware unable to read the CPB correctly and fall back to the factory image during subsequent reconfiguration.

Table 49. RSU Image 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 M	512 K	Decision firmware	
1.5 M	512 K	Decision firmware	
2 M	8 K + 24 K pad	Decision firmware data	
2 M+32 K	32 K	Reserved for SDM	
2 M+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

continued...

Flash Offset	Size (in bytes)	Contents	Sub-Partition Name
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 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 Quartus Prime Programming File Generator software supports up to 126 partitions. Each sub-partition descriptor is 32 bytes.

Note: The firmware never updates the SPT.

Table 50. Sub-partition Table Layout

Offset	Size (in bytes)	Description
0x000	4	Magic number 0x57713427
0x004	4	Version number: • 0 - before Quartus Prime Pro Edition software version 20.4 • 1 - starting with Quartus Prime Pro Edition software version 20.4
0x008	4	Number of entries
0x00C	4	Checksum: • 0 - before Quartus Prime Pro Edition software version 20.4 • CRC32 checksum - starting with Quartus Prime Pro Edition software version 20.4
0x010	16	Reserved
0x020	32	Sub-partition Descriptor 1
0x040	32	Sub-partition Descriptor 2
0xFE0	32	Sub-partition Descriptor 126

Starting with Quartus Prime Pro Edition software version 20.4, the SPT header contains a CRC32 checksum that is computed over the whole SPT. The value of the CRC32 checksum field itself is assumed as zero when the checksum is computed.

The checksum is provided as a convenience so that SPT corruptions can better be detected by HPS software. By default the feature is turned off.

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

Table 51. 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:

- System flag, if set to 1: Reserved for RSU system. For partition offset value, refer to RSU Image Sub-Partitions Layout.
- Read-only flag, if set to 1: The system protects partition against direct writes.

Note: You can use the read-only flag as an additional information during your API development. Read-only flag=1 indicates the content can only be changed by API from Intel.

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

Table 52. Flags Specifying Contents and Access

Partition	System	Read Only
BOOT_INFO	1	1
FACTORY_IMAGE	1	1
SPT0	1	0
SPT1	1	0

continued...

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

Note: In order to successfully update SPTs, the HPS software (U-Boot or Linux) must be configured to have a QSPI erase granularity of 32 KB or less. When configured with a coarser erase granularity (like 64 KB for example), the operation fails. All supported flash devices offer erase granularities of 4 KB, 32 KB, and 64 KB, and the default for the current HPS software is 4 KB.

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 53. 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	Total number of image pointer slots allocated = 508
0x18	8	Reserved
0x20 (IPTAB)[21]	8	First (lowest priority) image pointer slot (IPTAB)

continued...

(21) The offset may vary in future firmware updates.

Offset	Size (in bytes)	Description
	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. 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 new slot. Consequently, you can make 508 updates before the pointer block must be erased. The erase procedure is called *pointer block compression*. This procedure is power failure safe, as there are two copies of the pointer block. The copies are in different flash erase sectors. While one copy is being updated the other copy is still valid.

Note: In order to successfully update SPTs and CPBs, the HPS software (U-Boot or Linux) must be configured to support a minimum QSPI erase granularity smaller or equal to CPB and SPT sizes. All supported flash devices offer erase granularities of 4 KB, 32 KB, and 64 KB. HPS software is typically configured with either 4 KB or 64 KB erase granularity. When the HPS software is configured with 64 KB erase granularity, the CPB and SPT sizes must be configured in Programming File Generator to be 64 KB instead of the default 32 KB.

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 by 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 all 0's to this address.

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

1. Read all the valid entries from the configuration
2. Erase the pointer block
3. Add all previously valid entries
4. Add the address of 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 V 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.

Note: In order to successfully perform CPB compression, the HPS software (U-Boot or Linux) must be configured to have a QSPI erase granularity of 32 KB or less. When configured with a coarser erase granularity (like 64 KB for example), the operation fails. All supported flash devices offer erase granularities of 4 KB, 32 KB, and 64 KB, and the default for the current HPS software is 4 KB.

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

Use the Quartus Prime Programming File Generator tool to generate the Agilex 5 remote system update flash programming files.

Note: The absolute address option is removed from the Programming File Generator tool in the Quartus Prime software version 22.1 onwards. The Quartus Prime software applies the relative address flow by default.

5.5.1. Generating the Initial RSU Image

The following sections provide steps to generate the initial RSU Image.

For generic applications, you can generate the initial image using the `.sof` file.

5.5.1.1. Generating the Initial RSU Image Using SOF Files

Follow these steps to generate the initial RSU image:

1. On the **File** menu, click **Programming File Generator**.
2. Select Agilex 5 from the **Device family** drop-down list.
3. Select the configuration scheme from the **Configuration mode** drop-down list. The current 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:
   - 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 image `.sof` file and click **Open**. Repeat this step for the application image `.sof`.
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 Files** drop-down list and click **OK**.

   Note: You must assign Page 0 to Factory Image. Intel recommends that you let the 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 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. If you generate `.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 performs a verify operation using the Programmer, 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`.
15. 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 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.

5.5.1.2. Generating the Initial RSU Image Using RBF Files

Follow these steps to generate the initial RSU image using `.rbf` file:

1. Run the following command to generate a `.rbf` file from a factory or application image file (`.sof`).

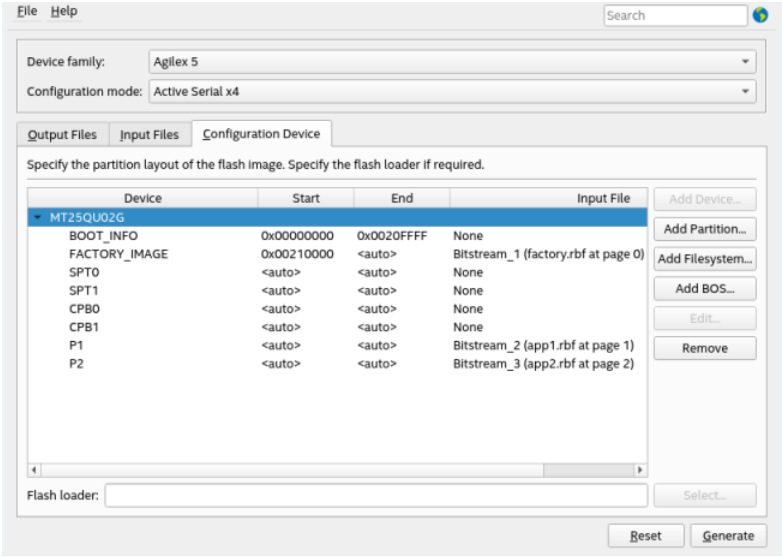
   ```
   quartus_pfg -c factory.sof factory.rbf
   quartus_pfg -c app1.sof app1.rbf
   quartus_pfg -c app2.sof app2.rbf
   ```
2. Run the following command to generate a boot `.rbf` file from a factory image file (`.sof`)

   ```
   quartus_pfg -c factory.sof boot.rbf -o rsu_boot=ON
   ```
3. On the **File** menu, click **Programming File Generator**.

4. Select Agilex 5 from the **Device family** drop-down list.
5. Select the configuration scheme from the **Configuration mode** drop-down list. The current Quartus Prime only supports remote system update feature in **Active Serial x4**.
6. On the **Output Files** tab, assign the output directory and file name.
7. Select the output file type.
   - JTAG Indirect Configuration File (`.jic`)/Programmer Object File (`.pof`)
   - Memory Map File (`.map`)
   - Raw Programming File (`.rpd`)
8. On the **Input Files** tab, click **Add Bitstream**, select the factory image `.rbf` file and click **Open**. Repeat this step for the application image `.rbf`.
9. 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.
   a. Assign `boot.rbf` to the `BOOT_INFO` partition.
   b. Assign `factory.rbf` to the `FACTORY_IMAGE` partition.
   c. Assign `app1.rbf` and `app2.rbf` to the `P1` and `P2` respectively.

      Note: `P1` and `P2` are user-defined partition names.
10. If you generate the `.jic` file, click **Select** at the Flash Loader. Select the device family and device name. Click **OK**.
11. Click **Generate** to generate the remote system update programming files. After generating the programming file, you can proceed to program the flash memory.

Figure 74. Generating Remote System Update Programming Files



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:

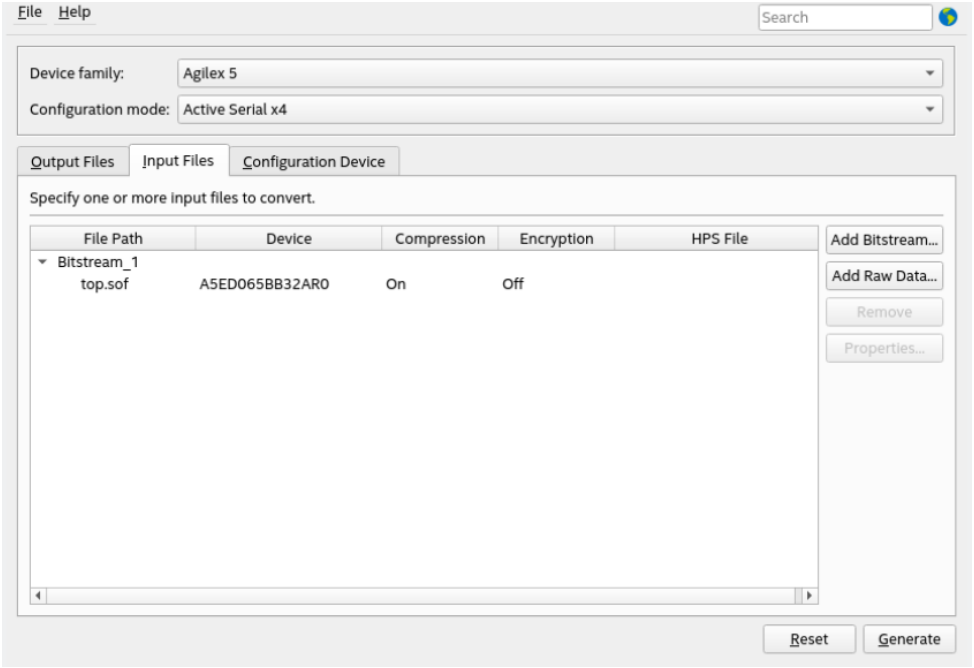
```
quartus_pfg -c fpga.sof application.rpd -o mode=ASX4 -o bitswap=ON
```

Alternatively, you can use the 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 Agilex 5 from the **Device family** drop-down list.
3. Select the configuration mode from the **Configuration mode** drop-down list. The current 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.
   - Raw Programming File (`.rpd`)
6. 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.

   Note: The rsu1.tcl script that Intel provides performs the bit swap operation. Consequently, if you are using this script, set **Bit swap** to **Off**.
7. 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 75. Specify the .sof File



8. 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 bitswap=ON -o rsu_upgrade=ON
```

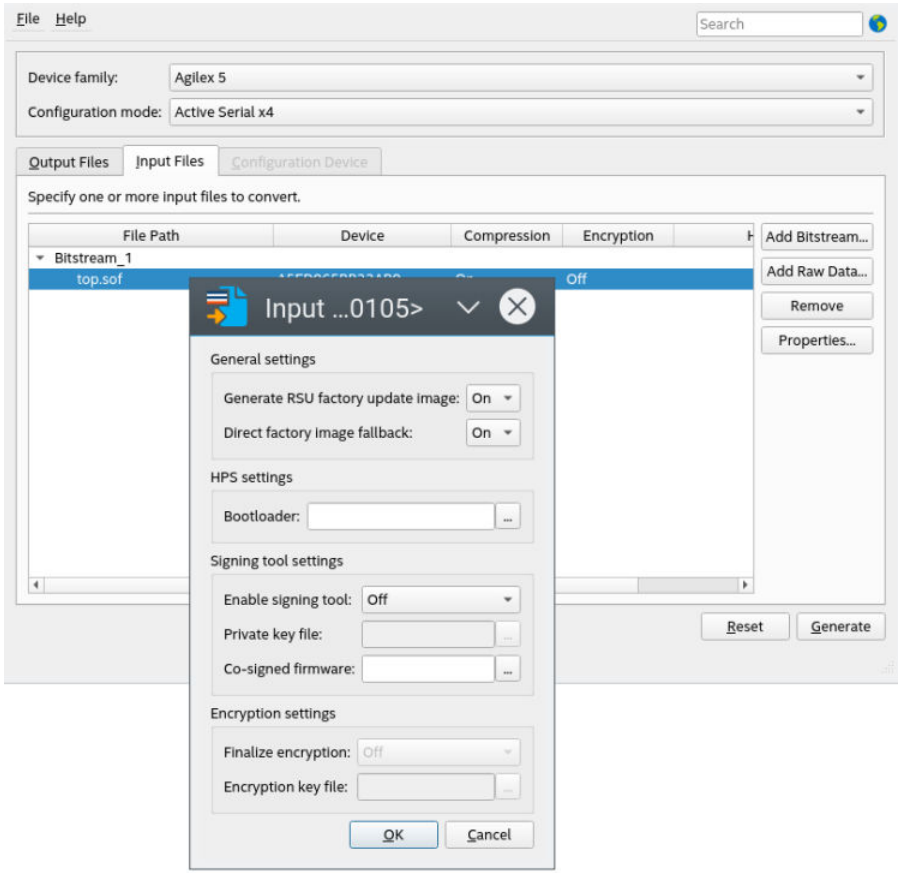
Alternatively, you can use the Quartus Prime Pro Edition **Programming File Generator** to generate a factory update image (`.rpd`). You can use this image to update the decision firmware, decision firmware data, and the factory image.

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.

1. On the **File** menu, click **Programming File Generator**.
2. Select Agilex 5 from the **Device family** drop-down list.
3. Select the configuration mode from the **Configuration mode** drop-down list. The current 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 file name.
5. Select the output file type:
   - Raw Programming File (`.rpd`)
6. 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.

   Note: The rsu1.tcl script that Intel provides performs the bit swap operation. Consequently, if you are using this script, set **Bit swap** to **Off**.
7. 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 76. Specify the .sof File



8. Select the `.sof` and then click **Properties.** Turn **On Generate RSU factory update image**. Specify the **Bootloader** file.

   Note: You only have to specify the **Bootloader** file for Agilex 5 devices that support HPS.

9. Click **Generate** to generate the RSU programming files. You can now update the decision firmware, decision firmware data, and the factory image in flash memory. 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 Command List and Descriptions 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*`.

   Note:
   - This step is mandatory for JTAG configuration schemes.
   - For AS x4 configuration schemes:
     - If your system has multiple quad SPI devices, this step is mandatory to access data stored in quad SPI devices connected to `nCSO[3:0]`.
     - If your system has only one quad SPI device, this step is optional to access data stored in the quad SPI device connected to `nCSO[0]`.
3. Perform the desired operations. The following operations are available:
   - `QSPI_READ`
   - `QSPI_WRITE`
   - `QSPI_ERASE`
   - `QSPI_READ_DEVICE_REG`
   - `QSPI_WRITE_DEVICE_REG`
   - `QSPI_SEND_DEVICE_OP`
   - `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.

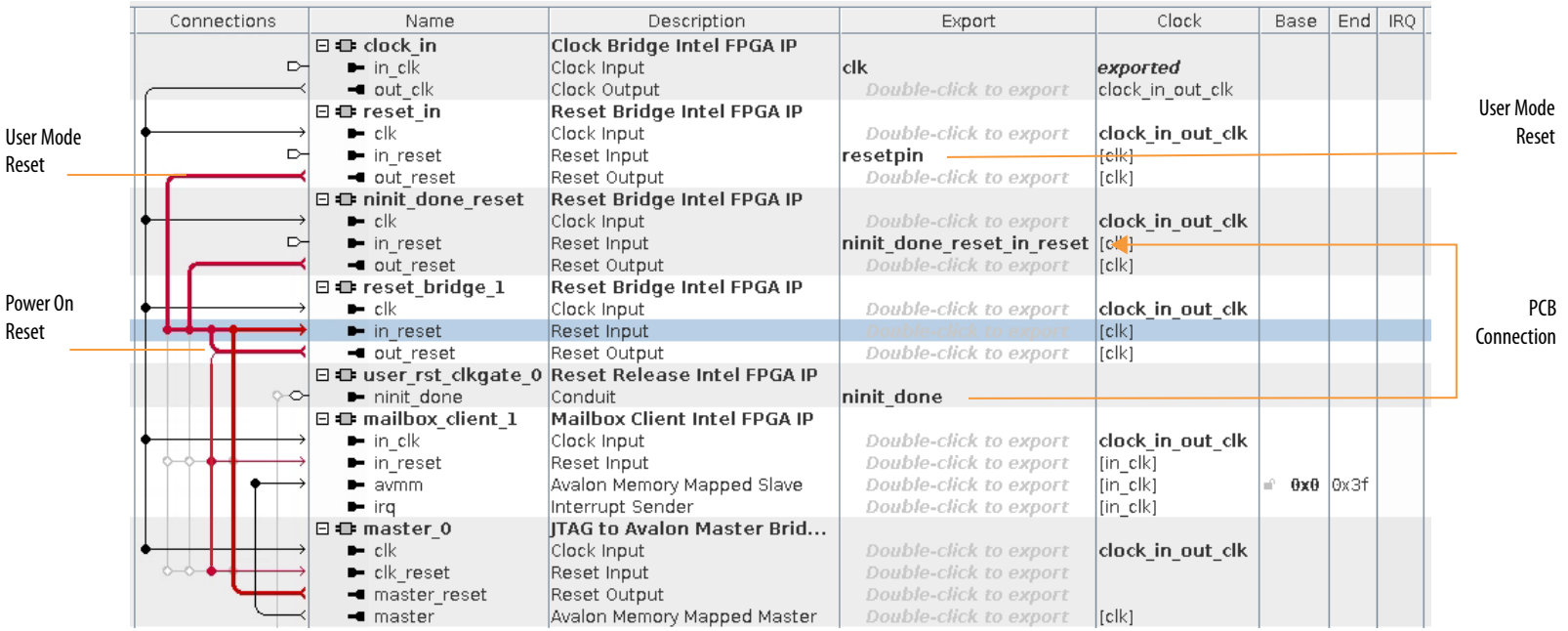
5.6.1. Prerequisites

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 Agilex 5 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 Intel FPGA 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 77. 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

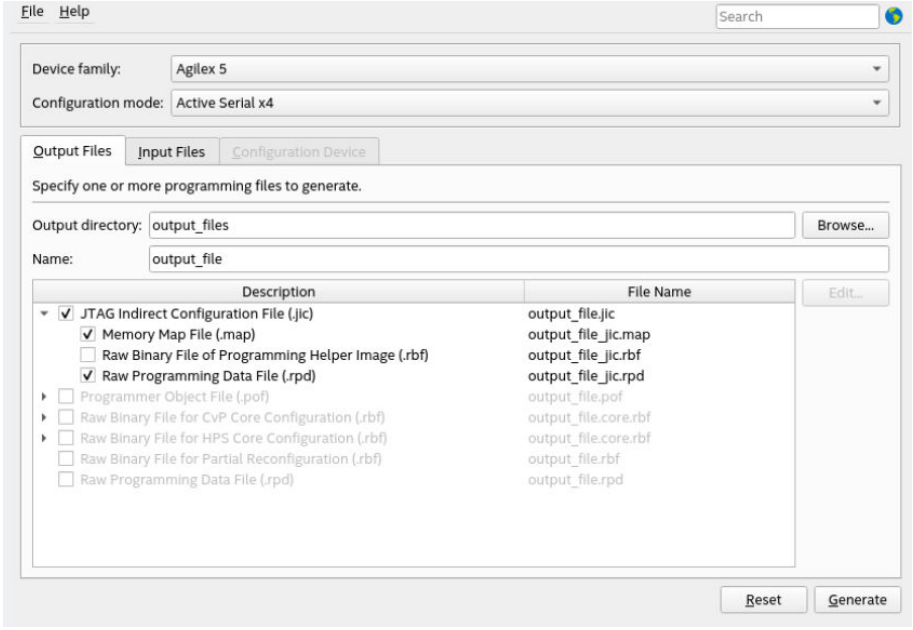
1. On the **File** menu, click **Programming File Generator**.
2. Select Agilex 5 from the **Device family** drop-down list.
3. Select the configuration mode from the **Configuration mode** drop-down list. The current Quartus Prime Software only supports remote system update feature in **Active Serial 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:

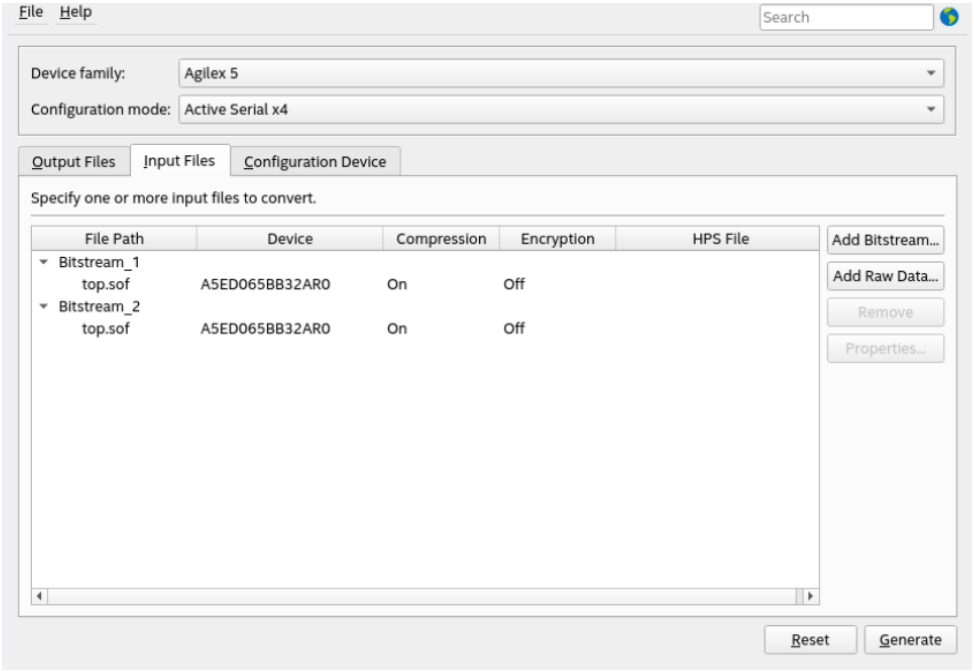
   - JTAG Indirect Configuration File (`.jic`)
   - Memory Map File (`.map`)
   - Raw Programming File (`.rpd`). Generating the `.rpd` file is optional.

Figure 78. Output Files Tab: Creating Initial Flash Image



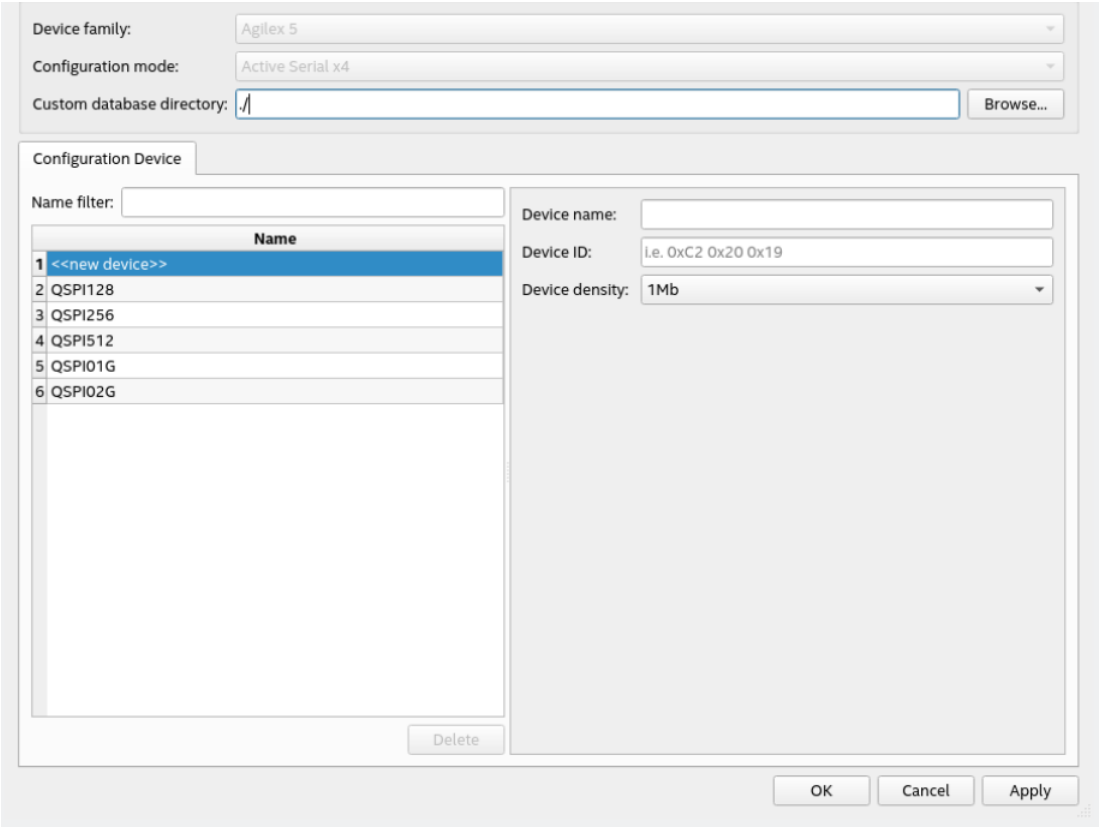
6. On the **Input Files** tab, click **Add Bitstream**, select the factory image `.sof` file and click **Open**. Repeat this step for the application image `.sof`.
   a. Bitstream_1 is the bitstream for factory image.
   b. Bitstream_2 is the bitstream for application image.

Figure 79. Input Files Tab: Specifying the .sof



7. On the **Configuration Device** tab, click **Add Device**, select the density of the flash memory and click **OK**. The Programming File Generator tool automatically populates the flash partitions.

Figure 80. Configuration Device Tab



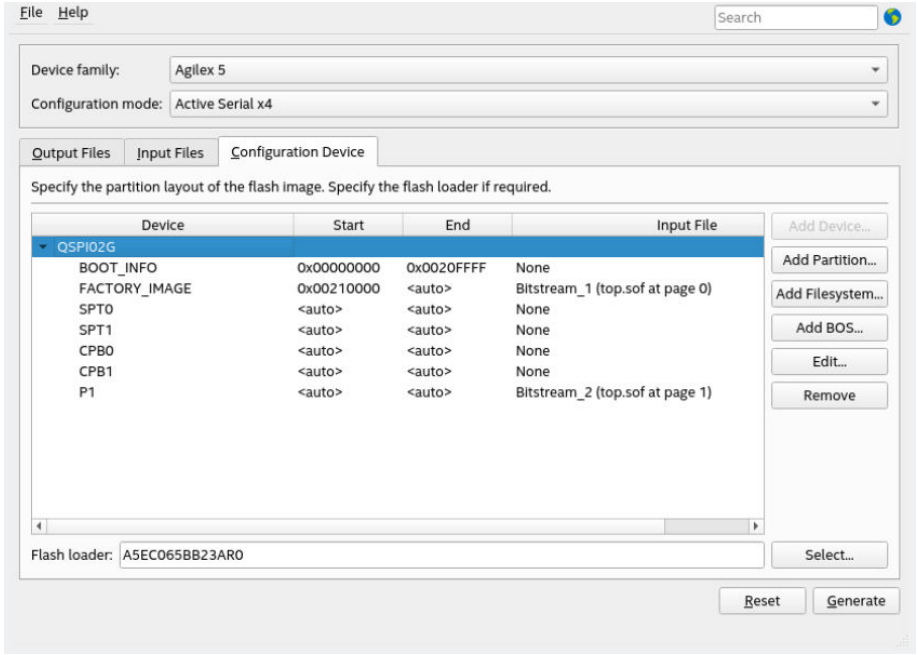
8. Select the **FACTORY_IMAGE** partition and click **Edit**.
9. On the **Edit Partition** dialog box, select **Bitstream_1** as the factory image `.sof` in the **Input file** drop-down list. Keep the default settings for the **Page** and **Address Mode**. Click **OK**.
10. Select the **QSPI02G** flash 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. Select **QSPI02G** flash memory, click **Edit**, and select the **Min Erase Size**.

    Note: The default minimum erase size is 4 KB. It is not necessary to change this value if your flash memory supports a minimum erase size of 4 KB.

13. For **Flash loader** click **Select**. Select Agilex 5 from **Device family** list. Select **A5EC065BB23AR0** 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 81. 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
P1                          0x004B0000      0x006EAFFF

Configuration device: A5EC065BB23AR0
Configuration mode: Active Serial x4

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

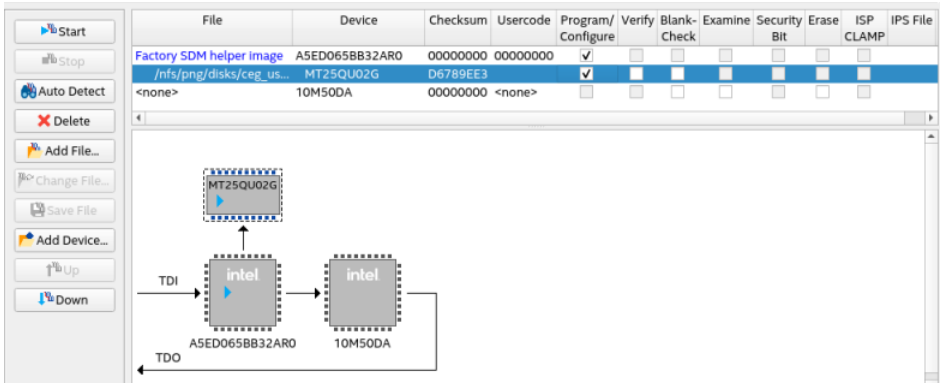
You can program the initial remote system update image from the command line. In the following command, substitute your `.jic` for `output_file.jic` if necessary.

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

Alternatively, you can use the 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. Select the checkbox for 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 Agilex 5 device with the application image using the AS x4 configuration scheme.

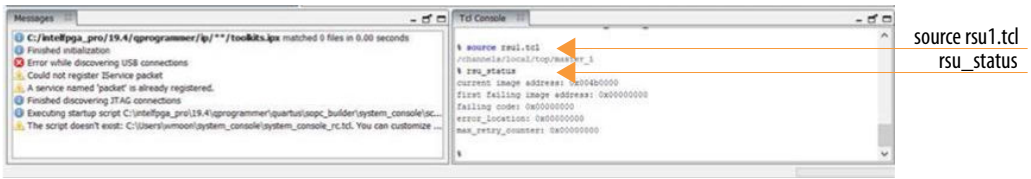
Figure 82. Programming the Flash Memory with the Initial RSU Image



Note: This example does not assign the `Direct to Factory Image` pin. Consequently, the device is configured 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 image the device is configured with as shown in the following example:
   a. In the 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 83. 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 0x00210000`

         This command reconfigures the device with factory image. Address `0x00210000` 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 0x004B0000`

         This command reconfigures the device with the application image. Address `0x004B0000` 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 84. 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 85. 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: When using HPS to manage RSU, you must update both copies of the Configuration Pointer Block (CPB0 and CPB1) and the sub-partition table (SPT). In a non-HPS case, while updates to both copies of the pointer blocks are mandatory, the updates to the sub-partition table are not required. For more details about the SPT and CPB, refer to Sub-partition Table Layout for the sub-partition table layout and Pointer Block Layout for the pointer block layout.

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.

Table 54. Configuration Firmware Pointer Block Contents

CPB Start Address + 0x20	Content	Value
CPB0 + 0x20 = 0x004A0020	Current application image pointer entry (highest priority)	0x004B0000
CPB0 + 0x28 = 0x004A0028	Next image pointer entry	0xFFFFFFFF
CPB1 + 0x20 = 0x004A8020	Current application image pointer entry (highest priority)	0x004B0000
CPB1 + 0x28 = 0x002EC028	Next image pointer entry	0xFFFFFFFF

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 86. 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 87. Writing an Address Pointer to the New Image Pointer Entry

```
% qspi_write_one_word 0x004a0028 0x03FF0000
% qspi_write_one_word 0x004a8028 0x03FF0000
```

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.

Figure 88. Verifying the Update to the New Image Pointer

```
% qspi_read 0x004a0028 1
0x03FF0000
% qspi_read 0x004a8028 1
0x03FF0000
```

Host software can now reconfigure the Agilex 5 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 55. 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)	0x03FF0000
CPB1 + 0x20 = 0x004A8020	Old application image pointer entry (lower priority)	0x004B0000
CPB1 + 0x28 = 0x004A8028	Current/New application image pointer entry (highest priority)	0x03FF0000

Figure 89. Read Current CPB Values

```
% qspi_read 0x004A0020 1
0x004B0000
% qspi_read 0x004A0028 1
0x03FF0000
% qspi_read 0x004A8020 1
0x004B0000
% qspi_read 0x004A8028 1
0x03FF0000
```

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 90. Remove Application Image

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

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 91. 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. Agilex 5 Configuration Features

6.1. Device Security

The Agilex 5 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

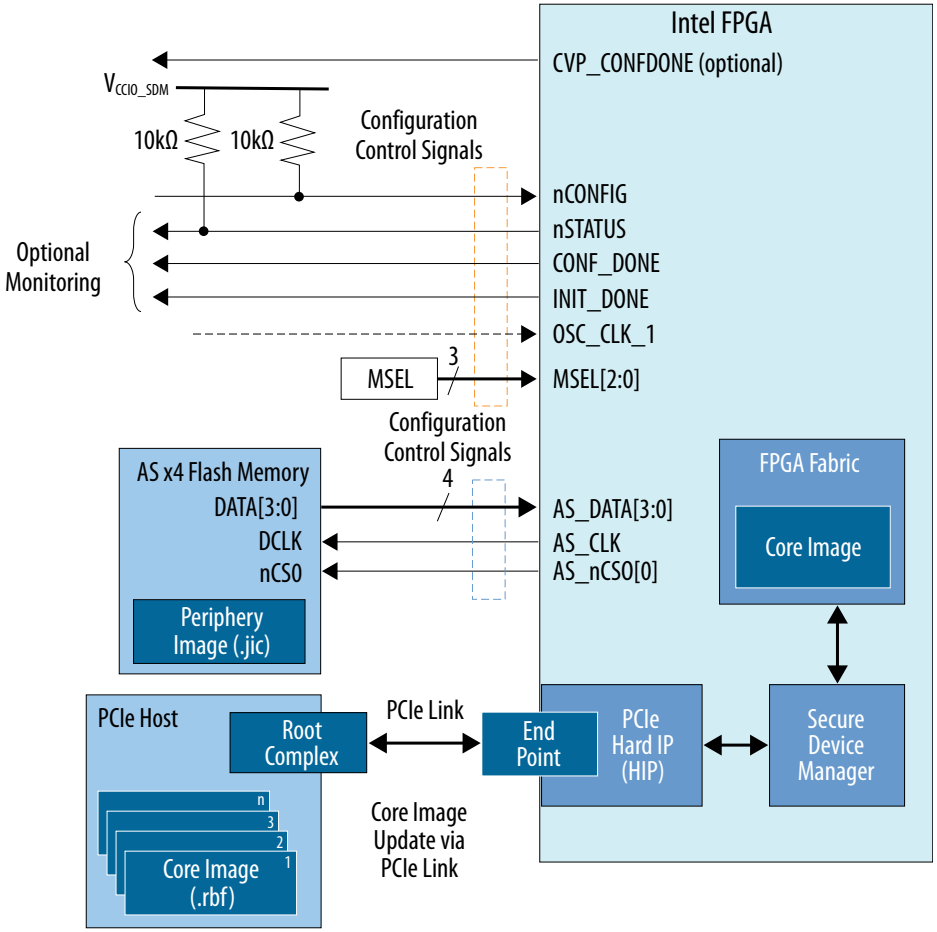
Related Information

Device Security User Guide: Agilex 5 FPGAs and SoCs
Device Security User Guide: Agilex 5 FPGAs and SoCs (Intel RDC item #815428)

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 92. Agilex 5 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 received 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

Quartus Prime Pro Edition User Guide: Partial Reconfiguration

7. Agilex 5 Debugging Guide

This section describes best practices in debugging configuration issues on SDM devices. View the video guide below for a quick walk-through.

7.1. Configuration Debugging Checklist

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

Table 56. General Configuration Debugging Checklist

	Checklist Item	Complete?
1	Verify that all configuration resistors are correctly connected (MSEL, nCONFIG, nSTATUS, CONF_DONE, INIT_DONE, PWRMGT_SDA, PWRMGT_SCL).	☐
2	Verify that you are following the correct power-up and power-down sequences.	☐
3	Verify that the SDM I/Os assignments are correct by checking the Quartus Prime Compilation QSF and Fitter reports.	☐
4	For SmartVID devices (-V or -E), ensure that all PMBus pins are connected to Agilex 5 device.	☐
5	Verify that SmartVID settings follow the recommendations in the *Agilex 5 Power Management User Guide*	☐
6	Verify that the Agilex 5 -V or -E device has its own voltage regulator module for V_{CC}, V_{CCP}, V_{CCL_HPS}, and $V_{CCPLLDIG\_HPS}$	☐
7	After configuration are the nCONFIG, nSTATUS, CONF_DONE, and INIT_DONE pins high? For more information about configuration status, refer to the *Understanding Configuration Status Using `quartus_pgm` Command* section.	☐
8	Is the SDM operating Boot ROM code or configuration firmware?	☐
9	Are the MSEL pins correctly connected on board?	☐
10	For designs that use transceivers, HBM2, PCIe, or EMIF, are the reference clocks stable and free running before configuration begins?	☐
11	Verify that selected clocks match the frequency setting specified in the Quartus Prime software during configuration.	☐
12	Does your design include the Reset Release IP?	☐
13	To avoid configuration failures, disconnect the PMBus regulator's JTAG download cable before configuring Agilex 5 -V devices.	☐
14	Verify that the Agilex 5 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 Agilex 5 devices that support HPS, ensure that the HPS and EMIF IOPLL reference clocks 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 Quartus Prime Software?	☐

Related Information

Power Management User Guide: Agilex 5 FPGAs and SoCs

7.2. Agilex 5 Configuration Architecture Overview

Agilex 5 devices employ configuration architecture that is quite similar to the 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 Agilex 5 and 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. Understanding Configuration Status Using quartus_pgm command

You can read the device configuration status from the command line directly. In the following command, specify the cable index and the JTAG chain device index to obtain the configuration status.

```
quartus_pgm -c <cable index> -m jtag -device=<device order in jtag chain> -status
```

The following example output shows the device status with cable index set to 1 and the second device selected as the targeted device in the JTAG chain after a successful configuration.

```
quartus_pgm -c 1 -m jtag -device=2 -status
```

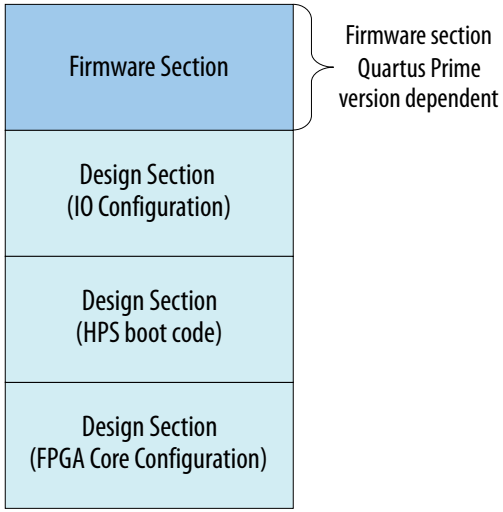
```
Info (21597): Response of CONFIG_STATUS
   Device is running in user mode
   00006000  RESPONSE_CODE=OK, LENGTH=6
   00000000  STATE=IDLE
   00000000  Version
   C000000F  MSEL=JTAG, nSTATUS=1, nCONFIG=1
   00000003  CONF_DONE=1, INIT_DONE=1, CVP_DONE=0, SEU_ERROR=0
   00000000  Error location
   00000000  Error details
```

7.4. 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 93. Example of an Agilex 5 Configuration Bitstream Structure



The firmware section is not part of the `.sof` file. The Quartus Prime Pro Edition Programmer adds the firmware to the `.sof`. The programmer adds the firmware when configuring an Agilex 5 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.

7.5. 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 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 and double adjacent bit error detection and correction. The multi-bit error and non-adjacent bit error are detected, but cannot be corrected. Additionally, some Intel FPGAs contain dedicated circuitry to help detect and correct errors.

For more information about SEUs, refer to *Agilex 5 SEU Mitigation User Guide*.

Related Information

SEU Mitigation User Guide: Agilex 5 FPGAs and SoCs

7.6. Reading the Unique 64-Bit CHIP ID

The Chip ID Intel FPGA IP in each Agilex 5 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 Agilex 5 device.

Related Information

Mailbox Avalon ST Client Intel FPGA IP User Guide

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 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.

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 Quartus Prime.
- Try configuring the Agilex 5 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 65
- Debugging Guidelines for the AS Configuration Scheme on page 128
- Debugging Guidelines for the JTAG Configuration Scheme on page 137

7.8. Configuration Debugger Tool

You can use the Configuration Debugger Tool to debug configuration issues. Refer to the *AN 955: Programmer's Configuration Debugger Tool* for more information about the tool.

Related Information

AN 955: Programmer’s Configuration Debugger Tool

8. Document Revision History for the Device Configuration User Guide: Agilex 5 FPGAs and SoCs

Document Version	Quartus Prime Version	Changes
2024.04.01	24.1	Initial release.