Intel Stratix 10 Configuration User Guide

Updated for Intel® Quartus® Prime Design Suite: 19.1
3.1.9. QSF Assignments for Avalon-ST x32 ................................................................. 54
3.1.10. IP for Use with the Avalon-ST Configuration Scheme: Intel FPGA Parallel Flash Loader II IP Core ................................................................. 56
3.2. AS Configuration .............................................................................................. 81
  3.2.1. AS Configuration Scheme Hardware Components and File Types .............. 82
  3.2.2. AS Single-Device Configuration .................................................................. 84
  3.2.3. AS Using Multiple Serial Flash Devices ...................................................... 85
  3.2.4. AS Configuration Timing Parameters ......................................................... 87
  3.2.5. Maximum Allowable External AS_DATA Pin Skew Delay Guidelines .......... 88
  3.2.6. Programming Serial Flash Devices ............................................................ 88
  3.2.7. Serial Flash Memory Layout ...................................................................... 92
  3.2.8. AS_CLK ................................................................................................. 93
  3.2.9. Active Serial Configuration Software Settings ......................................... 94
  3.2.10. Intel Quartus Prime Programming Steps ................................................ 95
  3.2.11. Debugging Guidelines for the AS Configuration Scheme ....................... 98
  3.2.12. QSF Assignments for AS ....................................................................... 99
3.3. Configuration from SD/MMC ............................................................................ 102
  3.3.1. SD/MMC Single-Device Configuration ...................................................... 102
3.4. JTAG Configuration ......................................................................................... 104
  3.4.1. JTAG Configuration Scheme Hardware Components and File Types ......... 104
  3.4.2. JTAG Device Configuration .................................................................... 106
  3.4.3. JTAG Multi-Device Configuration ........................................................... 109
  3.4.4. Debugging Guidelines for the JTAG Configuration Scheme .................... 110

4. Remote System Update ...................................................................................... 113
  4.1. Remote System Update Functional Description ............................................ 115
    4.1.1. Remote System Update Using AS Configuration ................................... 115
    4.1.2. Remote System Update Configuration Images ....................................... 116
    4.1.3. Remote System Update Configuration Sequence .................................. 117
    4.1.4. RSU Recovery from Corrupted Images ................................................ 118
    4.1.5. Update of Static Firmware and Factory Image ...................................... 120
  4.2. Guidelines for Performing Remote System Update Functions for Non-HPS .... 122
  4.3. Commands and Responses ....................................................................... 123
    4.3.1. Operation Commands ........................................................................ 125
    4.3.2. Error Code Responses ........................................................................ 129
  4.4. Remote System Update Flash Device Layout .............................................. 131
    4.4.1. Configuration Firmware Pointer Block (CPB) ....................................... 132
4.5. Generating Remote System Update Image Files using Programming File Generator ............................................................
  4.5.1. Generating a Standard RSU Image .................................................................................................................. 133
  4.5.2. Generating a Single RSU Image ...................................................................................................................... 134
  4.5.3. Generating an Update Image for Static Firmware and Factory Image .............................................................. 136
4.6. Remote System Update from FPGA Core Example .......................................................................................................... 140
4.7. Prerequisites ............................................................................................................................................................ 140
4.8. Creating Initial Flash Image Containing Bitstreams for Factory Image and One Application Image ........................................ 142
4.9. Programming Flash Memory with the Initial Remote System Update Image ....................................................................... 148
4.10. Reconfiguring the Device with an Application or Factory Image ...................................................................................... 152
4.11. Adding an Application Image .................................................................................................................................... 153
4.12. Removing an Application Image ................................................................................................................................ 158

5. Intel Stratix 10 Configuration Features ......................................................................................................................................................... 161
  5.1. Device Security ......................................................................................................................................................... 161
  5.2. Configuration via Protocol ........................................................................................................................................... 161
  5.3. Partial Reconfiguration ............................................................................................................................................... 163

6. Intel Stratix 10 Debugging Guide ....................................................................................................................................... 164
  6.1. Configuration Debugging Checklist .............................................................................................................................. 164
  6.2. Intel Stratix 10 Configuration Architecture Overview ...................................................................................................... 165
  6.3. SDM Debug Toolkit Overview ...................................................................................................................................... 165
     6.3.1. Using the SDM Debug Toolkit .......................................................................................................................... 170
  6.4. Configuration Pin Differences from Previous Device Families ............................................................................................ 172
  6.5. Configuration File Format Differences .................................................................................................................................. 174
  6.6. Understanding SEUs .................................................................................................................................................. 175
  6.7. Reading the Unique 64-Bit CHIP ID .............................................................................................................................. 175
  6.8. Understanding and Troubleshooting Configuration Pin Behavior ............................................................................................ 175
     6.8.1. nCONFIG ..................................................................................................................................................... 176
     6.8.2. nSTATUS ...................................................................................................................................................... 177
     6.8.3. CONF_DONE and INIT_DONE .......................................................................................................................... 178
     6.8.4. SDM_IO Pins ................................................................................................................................................ 178

7. Intel Stratix 10 Configuration User Guide Archives ............................................................................................................ 181

1. Intel® Stratix® 10 Configuration User Guide

1.1. Intel® Stratix® 10 Configuration Overview

All Intel® Stratix® 10 devices include a Secure Device Manager (SDM) to manage FPGA configuration and security. The SDM provides a failsafe, strongly authenticated, programmable security mode for device configuration. Previous FPGA families include a fixed state machine to manage device configuration.

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

Intel describes configuration schemes from the point-of-view of the FPGA. Intel Stratix 10 devices support active and passive configuration schemes. In active configuration schemes the FPGA acts as the master and the external memory acts as a slave device. In passive configuration schemes an external host acts as the master and controls configuration. The FPGA acts as the slave device. All Intel Stratix 10 configuration schemes support design security, remote system update, and partial reconfiguration. To implement remote system update in passive configuration schemes, an external controller must store and drive the configuration bitstream.

Intel Stratix 10 devices support the following configuration schemes:

- Avalon® Streaming (Avalon-ST)
- JTAG
- Configuration via Protocol (CvP)
- Active Serial (AS) normal and fast modes
- Secure Digital and Multi Media Card (SD/MMC)
Table 1. Intel Stratix 10 Configuration Data Width, Clock Rates, and Data Rates

<table>
<thead>
<tr>
<th>Configuration Scheme</th>
<th>Data Width (bits)</th>
<th>MSEL[2:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passive</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avalon-ST</td>
<td>32</td>
<td>000</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>101</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>110</td>
</tr>
<tr>
<td>JTAG</td>
<td>1</td>
<td>111</td>
</tr>
<tr>
<td>Configuration via Protocol (CvP)</td>
<td>x1, x2, x4, x8, x16 lanes</td>
<td>001</td>
</tr>
<tr>
<td>Active</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD/MMC</td>
<td>4/8</td>
<td>100</td>
</tr>
<tr>
<td>AS - fast mode</td>
<td>4</td>
<td>001</td>
</tr>
<tr>
<td>AS - normal mode</td>
<td>4</td>
<td>011</td>
</tr>
</tbody>
</table>

Avalon-ST

The Avalon-ST configuration scheme is a passive configuration scheme. Avalon-ST is the fastest configuration scheme for Intel Stratix 10 devices. Avalon-ST configuration supports x8, x16, and x32 modes. The x16 and x32 bit modes use general-purpose I/Os (GPIOs) for configuration. The x8 bit mode uses dedicated SDM I/O pins.

Avalon-ST supports backpressure using the AVST_READY and AVST_VALID pins. Because the time to decompress the incoming bitstream varies, backpressure support is necessary to transfer data to the Intel Stratix 10 device. For more information about the Avalon-ST refer to the Avalon Interface Specifications.

JTAG

You can configure the Intel Stratix 10 device using the dedicated JTAG pins. The JTAG port provides seamless access to many useful tools and functions. In addition to configuring the Intel Stratix 10, the JTAG port is used for debugging with Signal Tap or the System Console tools.

The JTAG port has the highest priority and overrides the MSEL pin settings. Consequently, you can configure the Intel Stratix 10 device over JTAG even if the MSEL pin specify a different configuration scheme unless you disabled JTAG for security reasons.

CvP

CvP uses an external PCIe® host device as a Root Port to configure the Intel Stratix 10 device over the PCIe link. You can specify up to a x16 PCIe link. Intel Stratix 10 devices support two CvP modes, CvP init and CvP update.
CvP initialization process includes the following two steps:

1. CvP configures the FPGA periphery image which includes I/O information and hard IP blocks, including the PCIe IP. Because the PCIe IP is in the periphery image, PCIe link training establishes the PCIe link of the CvP PCIe IP before the core fabric configures.

2. The host device uses the CvP PCIe link to configure your design in the core fabric.

CvP update mode updates the FPGA core image using the PCIe link already established from a previous full chip configuration or CvP init configuration. After the Intel Stratix 10 enters user mode, you can use the CvP update mode to reconfigure the FPGA fabric. This mode has the following advantages:

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

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

For Intel Stratix 10 SoC devices, CvP is only supported in FPGA configuration first mode.

For more information refer to the *Intel Stratix 10 Configuration via Protocol (CvP) Implementation User Guide*.

**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. This mode supports Intel's serial flash configuration memory solution for the following third-party flash devices:

- Micron MT25QU128, MT25QU256, MT25QU512, MT25QU01G, MT25QU02G

Refer to the *Supported Flash Devices for Intel Stratix 10 Devices* for complete list of supported flash devices.

**AS Fast Mode**

The only difference between AS normal mode and fast mode is speed. Use AS fast mode when configuration timing is a concern. Use this mode to meet the 100 ms of power up requirement for PCIe or for other systems with strict timing requirements.
In AS fast mode, the SDM first powers the external AS x4 flash. The power supply must be able to provide an equally fast ramp up for the Intel Stratix 10 device and the external AS x4 flash devices. Failing to meet this requirement causes the SDM to assume the memory is missing. Consequently, configuration fails.

Refer to the *Intel Stratix 10 Device Family Pin Connection Guidelines* and *AN692: Power Sequencing Considerations for Intel Cyclone® 10 GX, Intel Arria® 10, and Intel Stratix 10 Devices* for additional details.

**SD/MMC**

SD/MMC is an active configuration scheme. The Intel Stratix 10 SDM can initiate configuration from SD, Secure Digital High Capacity (SDHC*), Secure Digital Extended Capacity (SDXC*), MMC cards, and eMMC devices. The SD/MMC mode is almost identical to AS x4. The advantages of this mode are cost, capacity, availability, portability, and compatibility. Because Intel Stratix 10 devices operate at 1.8 volt an intermediate voltage level shifter may be required to interface with the higher voltage I/Os in SD/MMC devices.

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

**Related Information**

- Avalon Interface Specifications
- Device Configuration - Support Center
- Intel Stratix 10 Configuration via Protocol (CvP) Implementation User Guide
- Intel Stratix 10 Device Datasheet (Core and HPS)
- Supported Flash Devices for Intel Stratix 10 Devices

### 1.1.1. Configuration and Related Signals

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

You specify SDM I/O pin functions using the **Device ➤ Configuration ➤ Device and Pin Options** dialog box in the Intel Quartus Prime software.
Figure 1. Intel Stratix 10 Configuration Interfaces

This user guide discusses most of the interfaces shown in the figure. Refer to the separate Intel Stratix 10 Configuration via Protocol (CvP) Implementation User Guide and Intel Stratix 10 Power Management User Guide for more information about those features.

Related Information
- SDM Pin Mapping on page 24
- Intel Stratix 10 Configuration via Protocol (CvP) Implementation User Guide
- Intel Stratix 10 Power Management User Guide
1.1.2. Intel Download Cables Supporting Configuration in Intel Stratix 10 Devices

Intel provides the following cables to download your design to the Intel Stratix 10 device on the PCB. Download cables support prototyping activity by providing detailed debug messages via Intel Quartus Prime Programmer. You must use Intel download cables for advanced debugging using the Signal Tap logic analyzer or the System Console tools.

Table 2. Intel Stratix 10-Supported Download Cable Capabilities

<table>
<thead>
<tr>
<th>Download Cable</th>
<th>Protocol Support Intel Stratix 10 Device</th>
<th>Cable Connection to PCB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel FPGA Download Cable II (formerly the USB-Blaster II)</td>
<td>JTAG, AS</td>
<td>10-pin female plug</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3M Part number: 2510-6002UB</td>
</tr>
<tr>
<td>Intel FPGA Ethernet Cable (formerly the Ethernet Blaster II)</td>
<td>JTAG, AS</td>
<td>10-pin female plug</td>
</tr>
</tbody>
</table>

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

1.2. Intel Stratix 10 Configuration Architecture

The Secure Device Manager (SDM) is a triple-redundant processor-based module that manages configuration and the security features of Intel Stratix 10 devices. The SDM is available on all Intel Stratix 10 FPGAs and SoC devices.
The block diagram below provides an overview of the Intel Stratix 10 configuration architecture which includes the following blocks:

- **SDM**: More information about the SDM is contained in later sections.
- **Configuration network**: The SDM uses this dedicated, parallel configuration network to distribute the configuration bitstream to Local Sector Managers (LSMs). You cannot access this network.
- **LSMs**: The LSM is a microprocessor. Each configuration sector includes an LSM. The LSM parses configuration bitstream and configures the logic elements for its sector. After configuration, the LSM performs the following functions:
  - Monitors for single event upsets at the sector level
  - Processes responses to single event upsets (SEUs)
  - Performs hashing or integrity checks in real time
- **Specific blocks for Intel Stratix 10 variants**:
  - SX devices include the hard processor system (HPS) in addition to FPGA logic.
  - MX devices include a High Bandwidth Memory (HBM2) in addition to FPGA logic.
  - GX devices include FPGA logic and L- and H-Tile transceivers.
  - TX devices include FPGA logic and E- and H-Tile transceivers.
1.2.1. Secure Device Manager

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

- **Configuration bitstream authentication**: After power-on during startup, the SDM triple-redundant lockstep processors run code from the boot ROM. The boot ROM code authenticates the Intel-generated configuration firmware and configuration bitstream, ensuring that configuration bitstream is from a trusted source.
- **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.
Note: These security features are available in Intel Stratix 10 devices that support advanced security. The ordering codes for Intel Stratix 10 devices that include advanced security features includes the AS (Advanced Security) suffix. Contact your Intel Programmable Solutions representative to get additional information about Intel Stratix 10 device security features.

Figure 3. SDM Block Diagram
Here is an overview of the additional functions the SDM controls:

- The Power Management block consists of a voltage and temperature sensor which enables the SmartVID feature via an external PMBus voltage regulator when you select -V devices.
- The AES/SHA and other Crypto Accelerator blocks implement secure configuration and boot.
- The Key Vault provides volatile and non-volatile cryptographic key storage. To mitigate potential side-channel attacks, crypto functions that use keys require the keys a special hardware storage mechanism.
- The AS and SD/MMC configuration flash controllers enable active configuration schemes via dedicated SDM pins.
- The x8 Avalon-ST configuration scheme uses SDM I/O pins. The x16 and x32 Avalon-ST configuration schemes use dedicated SDM I/O pins and dual-purpose I/O pins. Refer to the SDM Pin Mapping for more information.
- To reduce configuration file size and support smaller memory sizes, and enable faster configuration, the Intel Quartus Prime software compresses the configuration data. Intel Stratix 10 devices all compress the configuration bitstream. You cannot disable this feature. The decompression block in the SDM decompresses both encrypted and non-encrypted configuration files.
- A specific PCIe block included in the Intel Stratix 10 device supports CvP.

Related Information
SDM Pin Mapping on page 24

1.2.1.1. Updating the SDM Firmware

When you generate a configuration bitstream using the File ➤ Programming File Generator menu item, the Intel Quartus Prime combines the SDM firmware with your .sof.

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

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

Newer versions of the Intel Quartus Prime software typically include new or updated SDM features.
Intel recommends using the latest version of the Intel Quartus Prime software when regenerating your configuration bitstream. Even if your .sof has not changed, the **Programming File Generator** adds the latest SDM firmware which may address known problems or add new features.

### 1.2.1.2. Specifying Boot Order for Intel Stratix 10 SoC Devices

For Intel Stratix 10 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. 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

For more information about specifying configuration order refer to the *FPGA Configuration First Mode* and *HPS Boot First Mode* chapters in the *Intel Stratix 10 SoC FPGA Boot User Guide*.

**Related Information**

- FPGA Configuration First Mode
- HPS Boot First Mode
2. Intel Stratix 10 Configuration Details

2.1. Intel Stratix 10 Configuration Timing Diagram

Figure 4. Configuration, Reconfiguration, and Error Timing Diagram

This internal signal is available after CONF_DONE goes high in designs that include the Reset Release IP.
The SDM drives Intel Stratix 10 device configuration.

**Initial Configuration Timing**

The first section of the figure shows the expected timing for initial configuration after a normal power-on reset. The initial state of the nCONFIG and nSTATUS signals is low.

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 high, the SDM enters Idle mode after booting.

3. When the external host drives nCONFIG signal high, the SDM initiates configuration. The SDM drives the nSTATUS signal high, signaling the beginning of FPGA configuration. The SDM receives the configuration bitstream on the interface that the MSEL bus specified in *Step 1*. The diagram shows AVST_READY and AVST_VALID continuously high. It is possible for AVST_READY to deassert which would require AVST_VALID to deassert within six cycles.

4. The SDM drives the CONF_DONE signal high, indicating the SDM received the bitstream successfully.

5. When the Intel Stratix 10 device asserts INIT_DONE the FPGA enters 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 uboot or the operating system (OS) initiates the configuration.

The entire device does not enter user mode simultaneously. Intel recommends that you include the Intel Stratix 10 Reset Release IP on page 21 to hold your application logic in the reset state until the entire FPGA fabric is in user mode. Failure to include this IP in your design may result in intermittent application logic failures.

**Reconfiguration Timing**

The second event the timing diagram illustrates the Intel Stratix 10 device reconfiguration. If you change the MSEL setting after power-on, you must power-cycle the Intel Stratix 10. Power cycling forces the SDM to sample the MSEL pins before reconfiguring the device.
The numbers in the Reconfiguration part of the timing diagram mark the following events:
1. The external host drives nCONFIG signal low.
2. The SDM initiates device cleaning.
3. The SDM drives the nSTATUS signal low when device cleaning is complete.
4. The external host drives the nCONFIG signal high to initiate reconfiguration.
5. The SDM drives the nSTATUS signal high signaling the device is ready for reconfiguration.

Configuration Error

The numbers in the Configuration Error part of the timing diagram mark the following events:
1. The SDM drives nSTATUS signal low for 1 ms ±50% to indicate a configuration error. The Intel Stratix 10 devices does not assert CONF_DONE indicating that configuration did not complete successfully.
2. The SDM enters the error state.
3. The SDM enters the idle state. The external host deasserts nCONFIG. The device is ready for reconfiguration by driving a low to high transition on nCONFIG. You can also power cycle the device by following the device power down sequence.

Power Supply Status

The power-on reset (POR) holds the Intel Stratix 10 device in the reset state until the power supply outputs are within the recommended operating range. $t_{RAMP}$ defines the maximum power supply ramp time. If POR does not meet the $t_{RAMP}$ time, the Intel Stratix 10 device I/O pins and programming registers remain tri-stated.

For more information about POR refer to the Intel Stratix 10 Power Management User Guide. For more information about $t_{RAMP}$ refer to the Intel Stratix 10 datasheet.

Related Information

- Intel Intel Stratix 10 Power Management User Guide
- Intel Intel Stratix 10 Device Datasheet (Core and HPS)
- Should clocks and resets in user logic be gated until the configuration process is completed in Intel Stratix 10?
2.2. Configuration Flow Diagram

This topic describes the configuration flow for Intel Stratix 10 devices.

**Figure 5. Intel Stratix 10 FPGA Configuration Flow**

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

**1 MS ±50% when SDM operates from bootROM code.

**Power Up**

- The Intel Stratix 10 power supplies power following the guidelines in the *Power-Up Sequence Requirements for Intel Stratix 10 Devices* section of the *Intel Stratix 10 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.
- The SDM_IO0, SDM_IO8, and SDM_IO16 pins remain low internally. Internal circuitry pulls the remaining SDM_IO pins to a weak high.
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.
- The SDM also enters the Idle state after it exits the error state.

Configuration Start

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

Configuration Pass

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

Configuration Error

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

- The SDM drives the INIT_DONE pin high after initializing internal registers and releases GPIO pins from the high impedance state. The device enters user mode. The entire device does not enter user mode at the same instant.
- The nCONFIG pin should remain high in user mode.
- You may re-configure the device by driving nCONFIG pin from low to high.

Device Clean

- In the Device Clean state the design stops functioning.
- Device cleaning zeros out all configuration data.
- The Intel Stratix 10 device drives CONF_DONE and INIT_DONE low.
- The SDM drives the nSTATUS pin low when device cleaning completes.

JTAG Configuration

**Note:** You can perform JTAG configuration anytime from any state except the power-on and SDM startup state. The Intel Stratix 10 device cancels the previous configuration and accepts the reconfiguration data from the JTAG interface. The nCONFIG signal must be held in a stable or low state during JTAG configuration. A falling edge on the nCONFIG signal cancels the JTAG configuration.

**Note:** The SDM only samples the MSEL pins at power-on and initiates bitstream configuration using the configuration scheme specified at power-on.

**Related Information**

2.3. Intel Stratix 10 Reset Release IP

Intel strongly recommends that you use Intel Stratix 10 Reset Release IP in your design to provide a known initialized state for your logic to begin operation.

The Reset Release IP is available in the Intel Quartus Prime Software, version 19.1 and later. 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. Because gating clocks may interfere with logic timing, Intel recommends that you use the nINIT_DONE signal to hold your design in reset.
If your design includes any initialized Intel Hyperflex™ registers, these registers must be gated using the \texttt{nINIT\_DONE} signal. Gating Intel Hyperflex registers with the \texttt{nINIT\_DONE} signal prevents these registers from losing their initial value. When you instantiate the Reset Release IP in your design, the SDM drives the \texttt{nINIT\_DONE} signal. Consequently, the IP does not consume any FPGA fabric resources, but does require routing resources. You can find the Reset Release IP in the Intel Quartus Prime and Platform Designer IP Catalogs under \textbf{Basic Functions/Configuration and Programming}.

\textbf{Figure 6.  Intel Stratix 10 Reset Release IP nINIT\_DONE Internal Connection}
Figure 7. **Intel Stratix 10 Reset Release IP nINIT_DONE External Connection**

**Related Information**
Should clocks and resets in user logic be gated until the configuration process is completed in Intel Stratix 10?

### 2.4. Additional Clock Requirements for Transceivers, HPS, PCIe, EMIF, and HBM2

The Intel Stratix 10 device has additional clock requirements for transceivers, the HPS, PCIe, External Memory Interface (EMIF), and the High Bandwidth Memory (HBM2) IP.

For successful configuration, the Intel Stratix 10 device requires additional clocks for transceivers, the HPS, PCIe, EMIF, and the HBM2 IP. You must provide a free-running, stable reference clock to these blocks before configuration begins. This reference clock is in addition to the configuration clock requirements for an internal or external oscillator described in **OSC_CLK_1 Requirements** on page 37. These blocks and their specific clock names are as listed below.
• HBM2: pll_ref_clk and ext_core_clk
• HPS: pll_ref_clk and ext_core_clk
• L- and H-Tile transceivers: REFCLK_GXB
• E-Tile transceivers: REFCLK_GXE

Note: The transceiver power supplies must be a nominal levels for successful configuration. You can use the V_CC and V_CCP power supplies for limited transceiver channel testing. Designs that include many transceivers require an auxiliary power supply to operate reliably.

2.5. Intel Stratix 10 Configuration Pins

The Intel Stratix 10 device uses SDM pins for device configuration.

2.5.1. SDM Pin Mapping

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

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

The Avalon-ST x8 and AS x4 configuration schemes use the dedicated SDM I/O pin assignments listed in Table 3 on page 24. Use the assignment in this table for MSEL and AVSTx8_DATA0 to AVSTx8_DATA8 and AS x4. Refer to Table 4 on page 26 for SDM I/O pin assignments for other configuration functions.

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

<table>
<thead>
<tr>
<th>SDM Pins</th>
<th>MSEL Function</th>
<th>Configuration Source Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDM_I00</td>
<td>—</td>
<td>Avalon-ST x8</td>
</tr>
<tr>
<td>SDM_I00</td>
<td>—</td>
<td>AS x4</td>
</tr>
<tr>
<td>SDM_I01</td>
<td>—</td>
<td>AVSTx8_DATA2</td>
</tr>
<tr>
<td>SDM_I01</td>
<td>—</td>
<td>AS_DATA1</td>
</tr>
<tr>
<td>SDM_I02</td>
<td>—</td>
<td>AVSTx8_DATA0</td>
</tr>
<tr>
<td>SDM_I02</td>
<td>—</td>
<td>AS_CLK</td>
</tr>
</tbody>
</table>

continued...
### SDM I/O Pin Assignments for Configuration Functions that Do Not Uses Dedicated SDM I/O Pins

Avalon-ST x16, and x32 configuration schemes use both SDM I/O and general-purpose I/Os (GPIOs) for device configuration. The GPIO pin assignments are fixed. The SDM I/O pin assignments are flexible. You assign SDM I/O pin functions using the `Device ➤ Configuration ➤ Device and Pin Options` dialog box in the Intel Quartus Prime software. The following table lists the possible pin assignments for configuration functions that you can assign to SDM I/O pins.

<table>
<thead>
<tr>
<th>SDM Pins</th>
<th>MSEL Function</th>
<th>Configuration Source Function</th>
<th>Avalon-ST x8</th>
<th>AS x4</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDM_IO3</td>
<td>—</td>
<td>AVSTx8_DATA3</td>
<td>AS_DATA2</td>
<td></td>
</tr>
<tr>
<td>SDM_IO4</td>
<td>—</td>
<td>AVSTx8_DATA1</td>
<td>AS_DATA0</td>
<td></td>
</tr>
<tr>
<td>SDM_IO5</td>
<td>MSEL0</td>
<td>—</td>
<td>AS_nCS0</td>
<td></td>
</tr>
<tr>
<td>SDM_IO6</td>
<td>—</td>
<td>AVSTx8_DATA4</td>
<td>AS_DATA3</td>
<td></td>
</tr>
<tr>
<td>SDM_IO7</td>
<td>MSEL1</td>
<td>—</td>
<td>AS_nCS2</td>
<td></td>
</tr>
<tr>
<td>SDM_IO8</td>
<td>—</td>
<td>AVST_READY</td>
<td>AS_nCS3</td>
<td></td>
</tr>
<tr>
<td>SDM_IO9</td>
<td>MSEL2</td>
<td>—</td>
<td>AS_nCS1</td>
<td></td>
</tr>
<tr>
<td>SDM_IO10</td>
<td>—</td>
<td>AVSTx8_DATA7</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>SDM_IO11</td>
<td>—</td>
<td>AVSTx8_VALID</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>SDM_IO12</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>SDM_IO13</td>
<td>—</td>
<td>AVSTx8_DATA5</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>SDM_IO14</td>
<td>—</td>
<td>AVSTx8_CLK</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>SDM_IO15</td>
<td>—</td>
<td>AVSTx8_DATA6</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>SDM_IO16</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>Signal Names</td>
<td>Configuration Scheme</td>
<td>Avalon-ST</td>
<td>AS x4</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------</td>
<td>-----------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>x8</td>
<td>x16</td>
<td>x32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWRMGT_SCL</td>
<td></td>
<td>SDM_IO0</td>
<td>SDM_IO0</td>
<td>SDM_IO0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO14</td>
<td>SDM_IO14</td>
<td>SDM_IO14</td>
</tr>
<tr>
<td>PWRMGT_SDA</td>
<td></td>
<td>SDM_IO12</td>
<td>SDM_IO11</td>
<td>SDM_IO11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO16</td>
<td>SDM_IO12</td>
<td>SDM_IO12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWRMGT_ALERT</td>
<td></td>
<td>SDM_IO0</td>
<td>SDM_IO0</td>
<td>SDM_IO0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO9</td>
<td>SDM_IO12</td>
<td>SDM_IO12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONF_DONE</td>
<td></td>
<td>SDM_IO0</td>
<td>SDM_IO0</td>
<td>SDM_IO0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO5</td>
<td>SDM_IO1</td>
<td>SDM_IO10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO12</td>
<td>SDM_IO2</td>
<td>SDM_IO11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO16</td>
<td>SDM_IO3</td>
<td>SDM_IO12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO4</td>
<td>SDM_IO4</td>
<td>SDM_IO13</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO5</td>
<td>SDM_IO5</td>
<td>SDM_IO14</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO6</td>
<td>SDM_IO6</td>
<td>SDM_IO15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO7</td>
<td>SDM_IO7</td>
<td>SDM_IO16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO10</td>
<td>SDM_IO9</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO11</td>
<td>SDM_IO10</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO12</td>
<td>SDM_IO11</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO13</td>
<td>SDM_IO12</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO14</td>
<td>SDM_IO13</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO15</td>
<td>SDM_IO14</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO16</td>
<td>SDM_IO15</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INIT_DONE</td>
<td></td>
<td>SDM_IO0</td>
<td>SDM_IO0</td>
<td>SDM_IO0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO5</td>
<td>SDM_IO1</td>
<td>SDM_IO10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO12</td>
<td>SDM_IO2</td>
<td>SDM_IO11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO16</td>
<td>SDM_IO3</td>
<td>SDM_IO12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO4</td>
<td>SDM_IO4</td>
<td>SDM_IO13</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO5</td>
<td>SDM_IO5</td>
<td>SDM_IO14</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO6</td>
<td>SDM_IO6</td>
<td>SDM_IO15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO7</td>
<td>SDM_IO7</td>
<td>SDM_IO16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO10</td>
<td>SDM_IO9</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO11</td>
<td>SDM_IO10</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO12</td>
<td>SDM_IO11</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO13</td>
<td>SDM_IO12</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO14</td>
<td>SDM_IO13</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO15</td>
<td>SDM_IO14</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDM_IO16</td>
<td>SDM_IO15</td>
<td></td>
</tr>
</tbody>
</table>

*continued...*
| Signal Names | SDM_IO0 | SDM_IO5 | SDM_IO10 | SDM_IO15 | Not supported | SDM_IO0 | SDM_IO5 | SDM_IO10 | SDM_IO15 | Not supported | SDM_IO0 | SDM_IO5 | SDM_IO10 | SDM_IO15 | Not supported | SDM_IO0 | SDM_IO5 | SDM_IO10 | SDM_IO15 | Not supported | SDM_IO0 | SDM_IO5 | SDM_IO10 | SDM_IO15 |
|-------------|--------|--------|----------|----------|--------------|--------|--------|----------|----------|--------------|--------|--------|----------|----------|--------------|--------|--------|----------|----------|--------------|--------|--------|----------|----------|--------------|--------|--------|----------|----------|
| x8          | SDM_IO6 | SDM_IO7 | SDM_IO9  | SDM_IO10 | SDM_IO11     | SDM_IO12| SDM_IO13| SDM_IO14 | SDM_IO15 | SDM_IO16     | SDM_IO6 | SDM_IO7 | SDM_IO9  | SDM_IO10 | SDM_IO11     | SDM_IO12| SDM_IO13| SDM_IO14 | SDM_IO15 | SDM_IO16     | SDM_IO6 | SDM_IO7 | SDM_IO9  | SDM_IO10 |
| x16         | SDM_IO5 | SDM_IO6 | SDM_IO7  | SDM_IO9  | SDM_IO10     | SDM_IO11| SDM_IO12| SDM_IO13| SDM_IO14 | SDM_IO15     | SDM_IO5 | SDM_IO6 | SDM_IO7  | SDM_IO9  | SDM_IO10     | SDM_IO11| SDM_IO12| SDM_IO13| SDM_IO14 | SDM_IO15     | SDM_IO5 | SDM_IO6 | SDM_IO7  | SDM_IO9  |
| x32         |        |        |          |          |              |        |        |          |          |              |        |        |          |          |              |        |        |          |          |              |        |        |          |          |
| AS x4       |        |        |          |          |              |        |        |          |          |              |        |        |          |          |              |        |        |          |          |              |        |        |          |          |
### 2. Intel Stratix 10 Configuration Details

<table>
<thead>
<tr>
<th>Signal Names</th>
<th>Configuration Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Avalon-ST</td>
</tr>
<tr>
<td></td>
<td>x8</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>HPS_COLD_nRESET</td>
<td>SDM_IO0</td>
</tr>
<tr>
<td></td>
<td>SDM_IO5</td>
</tr>
<tr>
<td></td>
<td>SDM_IO7</td>
</tr>
<tr>
<td></td>
<td>SDM_IO9</td>
</tr>
<tr>
<td></td>
<td>SDM_IO12</td>
</tr>
<tr>
<td></td>
<td>SDM_IO16</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Direct to Factory Image | SDM_IO0 | SDM_IO0 | SDM_IO0 |
|                        | SDM_IO5 | SDM_IO1 | SDM_IO10 |
|                        | SDM_IO7 | SDM_IO2 | SDM_IO11 |
|                        | SDM_IO9 | SDM_IO3 | SDM_IO12 |
|                        | SDM_IO12| SDM_IO4 | SDM_IO13 |
|                        | SDM_IO16| SDM_IO5 | SDM_IO14 |
|                        | SDM_IO16| SDM_IO6 | SDM_IO15 |
|                        | SDM_IO16| SDM_IO7 | SDM_IO16 |
|                        | SDM_IO9 | SDM_IO9 |       |
|                        | SDM_IO10| SDM_IO10|       |
|                        | SDM_IO11| SDM_IO11|       |
## Signal Names

<table>
<thead>
<tr>
<th>Configuration Scheme</th>
<th>Avalon-ST</th>
<th>AS x4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>x8</strong></td>
<td><strong>x16</strong></td>
<td><strong>x32</strong></td>
</tr>
<tr>
<td>SDM_I012</td>
<td>SDM_I012</td>
<td></td>
</tr>
<tr>
<td>SDM_I013</td>
<td>SDM_I013</td>
<td></td>
</tr>
<tr>
<td>SDM_I014</td>
<td>SDM_I014</td>
<td></td>
</tr>
<tr>
<td>SDM_I015</td>
<td>SDM_I015</td>
<td></td>
</tr>
<tr>
<td>SDM_I016</td>
<td>SDM_I016</td>
<td></td>
</tr>
<tr>
<td><strong>DATA_UNLOCK</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDM_I00</td>
<td>SDM_I00</td>
<td>SDM_I00</td>
</tr>
<tr>
<td>SDM_I05</td>
<td>SDM_I01</td>
<td>SDM_I010</td>
</tr>
<tr>
<td>SDM_I07</td>
<td>SDM_I02</td>
<td>SDM_I011</td>
</tr>
<tr>
<td>SDM_I09</td>
<td>SDM_I03</td>
<td>SDM_I012</td>
</tr>
<tr>
<td>SDM_I012</td>
<td>SDM_I04</td>
<td>SDM_I013</td>
</tr>
<tr>
<td>SDM_I016</td>
<td>SDM_I05</td>
<td>SDM_I014</td>
</tr>
<tr>
<td></td>
<td>SDM_I06</td>
<td>SDM_I015</td>
</tr>
<tr>
<td></td>
<td>SDM_I07</td>
<td>SDM_I016</td>
</tr>
<tr>
<td></td>
<td>SDM_I09</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SDM_I010</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SDM_I011</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SDM_I012</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SDM_I013</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SDM_I014</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SDM_I015</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SDM_I016</td>
<td></td>
</tr>
</tbody>
</table>

### Related Information
- Intel Stratix 10 Device Pinouts
- Intel Stratix 10 Device Family Pin Connection Guidelines
2.5.2. MSEL Settings

The MSEL[2:0] pins set the configuration scheme for Intel Stratix 10 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 configuration scheme. You must also specify the configuration scheme on the Configuration page of the Device and Pin Options dialog box in the Intel Quartus Prime Software.

Figure 8. MSEL Pull-Up and Pull-Down Circuit Diagram

![MSEL Pull-Up and Pull-Down Circuit Diagram](image)

Table 5. MSEL Settings for Each Configuration Scheme of Intel Stratix 10 Devices

<table>
<thead>
<tr>
<th>Configuration Scheme</th>
<th>MSEL[2:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avalon-ST (x32)</td>
<td>000</td>
</tr>
<tr>
<td>Avalon-ST (x16)</td>
<td>011</td>
</tr>
<tr>
<td>Avalon-ST (x8)</td>
<td>100</td>
</tr>
<tr>
<td>AS (Fast mode – for CvP)(^1)</td>
<td>001</td>
</tr>
<tr>
<td>AS (Normal mode)</td>
<td>011</td>
</tr>
<tr>
<td>SD/MMC x4/x8</td>
<td>100</td>
</tr>
<tr>
<td>JTAG only(^2)</td>
<td>111</td>
</tr>
</tbody>
</table>

\(^1\) If you use AS Fast mode and are not concerned about 100 ms PCIe linkup, you must still ramp the \(V_{CCIO_{SDM}}\) supply within 18 ms. This ramp-up requirement ensures that the AS x4 device is within its operating voltage range when the Intel Stratix 10 device begins to access it.

\(^2\) JTAG configuration works with any MSEL settings, unless disabled for security.
2.5.3. Device Configuration Pins

Device Configuration Pins without Fixed Assignments

All configuration schemes use the same dedicated pins for the standard control signals shown in Intel Stratix 10 Configuration Timing Diagram on page 16.

There are no dedicated pins for the following signals:

- PR_REQUEST
- PR_ERROR
- PR_DONE
- CvP_CONFDONE
- SEU_ERROR
- DIRECT_TO_FACTORY
- CONF_DONE
- INIT_DONE
- HPS_COLD_nRESET
- PWRMGT_SCL
- PWRMGT_SDA
- PWRMGT_ALERT
- DATA_UNLOCK

You can use the unused SDM I/O pins for CvP_CONFDONE, DIRECT_TO_FACTORY, and SEU_ERROR pins. You can only use GPIO for PR_REQUEST, PR_ERROR, and PR_DONE pins by specifying them in the Intel Quartus Prime software and connecting them to the Partial Reconfiguration External Configuration Controller Intel Stratix 10 FPGA IP.
Table 6. Intel Stratix 10 Device Configuration Pins

<table>
<thead>
<tr>
<th>Configuration Function</th>
<th>Configuration Scheme</th>
<th>Direction</th>
<th>Powered by</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCK(3)</td>
<td>JTAG</td>
<td>Input</td>
<td>V_CCIO_SDM</td>
</tr>
<tr>
<td>TDI(3)</td>
<td>JTAG</td>
<td>Input</td>
<td>V_CCIO_SDM</td>
</tr>
<tr>
<td>TMS(3)</td>
<td>JTAG</td>
<td>Input</td>
<td>V_CCIO_SDM</td>
</tr>
<tr>
<td>TDO(3)</td>
<td>JTAG</td>
<td>Output</td>
<td>V_CCIO_SDM</td>
</tr>
<tr>
<td>nSTATUS</td>
<td>All schemes</td>
<td>Output</td>
<td>V_CCIO_SDM</td>
</tr>
<tr>
<td>nCONFIG</td>
<td>All schemes</td>
<td>Input</td>
<td>V_CCIO_SDM</td>
</tr>
<tr>
<td>MSEL<a href="4">2:0</a></td>
<td>All schemes</td>
<td>Input</td>
<td>V_CCIO_SDM</td>
</tr>
<tr>
<td>CONF_DONE(5)</td>
<td>All schemes</td>
<td>Output</td>
<td>V_CCIO_SDM</td>
</tr>
<tr>
<td>INIT_DONE(6)</td>
<td>All schemes</td>
<td>Output</td>
<td>V_CCIO_SDM</td>
</tr>
<tr>
<td>OSC_CLK_1</td>
<td>All schemes</td>
<td>Input</td>
<td>V_CCIO_SDM</td>
</tr>
<tr>
<td>AS_nCSO[3:0]</td>
<td>AS</td>
<td>Output</td>
<td>V_CCIO_SDM</td>
</tr>
<tr>
<td>AS_DATA[3:0]</td>
<td>AS</td>
<td>Bidirectional</td>
<td>V_CCIO_SDM</td>
</tr>
<tr>
<td>AS_CLK</td>
<td>AS</td>
<td>Output</td>
<td>V_CCIO_SDM</td>
</tr>
<tr>
<td>AVST_READY</td>
<td>Avalon-ST x8/x16/32</td>
<td>Output</td>
<td>V_CCIO_SDM</td>
</tr>
<tr>
<td>AVSTx8_DATA[7:0]</td>
<td>Avalon-ST x8</td>
<td>Input</td>
<td>V_CCIO_SDM</td>
</tr>
<tr>
<td>AVSTx8_VALID</td>
<td>Avalon-ST x8</td>
<td>Input</td>
<td>V_CCIO_SDM</td>
</tr>
</tbody>
</table>

(3) The JTAG pins can access the HPS JTAG chain in Intel Stratix 10 SoC devices.

(4) The MSEL[2:0] pins are dual purpose. You can assign any unused MSEL[2:0] pin to other functions such as power management or non-dedicated configuration pins.

(5) You enable the CONF_DONE pin function in the Intel Quartus Prime Software. The Avalon-ST configuration scheme using the Parallel Flash Loader (PFL) II requires this pin.

(6) You enable the INIT_DONE pin function in the Intel Quartus Prime Software. This pin is optional for all configuration schemes.
Note: To avoid false signaling indicating successful configuration, Intel recommends that you include an external weak pull-down resistor for CONF_DONE and INIT_DONE pins.

SDM pins are also available for the SmartVID power management feature for -V devices. You must also set the correct Power Management Bus (PMBus) settings when using the SmartVID feature. Refer to the Intel Stratix 10 Power Management User Guide for more information about the pin assignments and PMBus setting.

Related Information
Intel Stratix 10 Power Management User Guide

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

Table 7. Intel Stratix 10 Configuration Pins I/O Standard, Drive Strength, and IBIS Model

<table>
<thead>
<tr>
<th>Configuration Pin Function</th>
<th>Direction</th>
<th>I/O Standard</th>
<th>Drive Strength (mA)</th>
<th>IBIS Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDO</td>
<td>Output</td>
<td>1.8V LVCMOS</td>
<td>8</td>
<td>18_io_d8s1_sdm_lvl</td>
</tr>
<tr>
<td>TMS</td>
<td>Input</td>
<td>Schmitt Trigger Input</td>
<td>—</td>
<td>18_in_sdm_lv</td>
</tr>
<tr>
<td>TCK</td>
<td>Input</td>
<td>Schmitt Trigger Input</td>
<td>—</td>
<td>18_in_sdm_lv</td>
</tr>
</tbody>
</table>

(7) These are dual purpose configuration pins. You can use these pins as GPIOs in user mode.
### Configuration Pin Function

<table>
<thead>
<tr>
<th>Configuration Pin Function</th>
<th>Direction</th>
<th>I/O Standard</th>
<th>Drive Strength (mA)</th>
<th>IBIS Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDI</td>
<td>Input</td>
<td>Schmitt Trigger Input</td>
<td>—</td>
<td>18_in_sdm_lv</td>
</tr>
<tr>
<td>nSTATUS</td>
<td>Output</td>
<td>1.8V LVCMOS</td>
<td>8</td>
<td>18_in_sdm_lv</td>
</tr>
<tr>
<td>OSC_CLK_1</td>
<td>Input</td>
<td>Schmitt Trigger Input</td>
<td>—</td>
<td>18_in_sdm_lv</td>
</tr>
<tr>
<td>nCONFIG</td>
<td>Input</td>
<td>Schmitt Trigger Input</td>
<td>—</td>
<td>18_in_sdm_lv</td>
</tr>
<tr>
<td>SDM_IO[16:0]</td>
<td>I/O</td>
<td>Schmitt Trigger Input or 1.8V LVCMOS</td>
<td>8</td>
<td>Input: 18_in_sdm_lv \nOutput: 18_in_sdm_lv</td>
</tr>
<tr>
<td>AVST_DATA[31:0], AVST_CLK, AVST_VALID</td>
<td>I/O</td>
<td>Schmitt Trigger Input or 1.8V LVCMOS</td>
<td>8</td>
<td>Input: 18_in_sdm_lv \nOutput: 18_in_sdm_lv</td>
</tr>
</tbody>
</table>

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

### Unused SDM Pins

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

### Related Information

IBIS Models for Intel Devices

### 2.5.4. Setting Additional Configuration Pins

You enable and assign the SDM pins listed in the [Device Configuration Pins without Fixed Assignments](#) on page 31 topic using the Intel Quartus Prime software.

Complete the following steps to assign these additional configuration pins:

1. On the Assignments menu, click Device.
2. In the Device and Pin Options dialog box, select the Configuration category and click Configuration Pins Options.
3. In the Configuration Pin window, enable and assign the configuration pin that you want to enable.
4. Click **OK** to confirm and close the **Configuration Pin** dialog box.

### 2.5.5. Enabling Dual-Purpose Pins

AVST_CLK, AVST_DATA[15:0], AVST_DATA[31:16], 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 1.8 V
- Assign the 1.8 V I/O standard to these pins
Complete the following steps to assign these settings to the dual-purpose pins:

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

### 2.6. 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. Specify an **OSC_CLK_1** clock source for the fastest configuration time. On the **Assignments** menu, click **Device**.
2. In the **Device and Pin Options** dialog box, select the **General** category.
3. Specify the configuration clock source from the **Configuration clock source** drop down menu.
4. Click **OK** to confirm and close the **Device and Pin Options**.

**Related Information**

**OSC_CLK_1 Clock Input** on page 37

### 2.7. Configuration Clocks

#### 2.7.1. OSC_CLK_1 Clock Input

**OSC_CLK_1 Requirements**

When you drive OSC_CLK_1 input clock with an external clock source and enable OSC_CLK_1 in the Intel Quartus Prime software, the device loads the majority of the configuration bitstream at 250 MHz. Intel Stratix 10 devices include an internal oscillator in addition to OSC_CLK_1 which runs the configuration process at a frequency between 170-230 MHz. Intel Stratix...
10 devices always use this internal oscillator to load the first section of the bitstream (approximately 200 kilobyte (KB)). The SDM can use either clock source for the remainder of device configuration. If you use the internal oscillator, you can leave the OSC_CLK_1 unconnected.

**Note:**

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

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

The Intel Stratix 10 device multiplies the OSC_CLK_1 source clock frequency to generate a 250 MHz clock for configuration. Using an OSC_CLK_1 source enables the fastest possible configuration. Refer to Setting Configuration Clock Source for instructions setting this frequency using the Intel Quartus Prime Software.

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

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

**Related Information**

- Intel Stratix 10 L- and H-Tile Transceiver PHY User Guide
- Intel Stratix 10 E-Tile Transceiver PHY User Guide
- Intel Stratix 10 External Memory Interfaces IP User Guide
- Setting Configuration Clock Source on page 36
3. Intel Stratix 10 Configuration Schemes

3.1. Avalon-ST Configuration

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

Table 8. Avalon-ST Configuration Data Width, Clock Rates, and Data Rates

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Data Width (bits)</th>
<th>Max Clock Rate</th>
<th>Max Data Rate</th>
<th>MSEL[2:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avalon-ST</td>
<td>32</td>
<td>125 MHz</td>
<td>4000 Mbps</td>
<td>000</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>125 MHz</td>
<td>2000 Mbps</td>
<td>101</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>125 MHz</td>
<td>1000 Mbps</td>
<td>110</td>
</tr>
</tbody>
</table>

Refer to the Intel Stratix 10 Data Sheet for configuration timing estimates.

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

Related Information

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

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

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

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

**Figure 9. Components and Design Flow for .pof Programming**
### Table 9. Output File Types

<table>
<thead>
<tr>
<th>Programming File Type</th>
<th>Extension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programmer Object File</td>
<td>.pof</td>
<td>The .pof is a proprietary Intel FPGA file type. Use the PFL II IP core via a JTAG header to write the .pof to an external CFI flash or serial flash device.</td>
</tr>
<tr>
<td>Raw Binary File</td>
<td>.rbf</td>
<td>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.</td>
</tr>
</tbody>
</table>

**Note:** 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.
3.1.2. Enabling Avalon-ST Device Configuration

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

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

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

3.1.3. The AVST_READY Signal

Before beginning configuration, trigger device cleaning by toggling the \texttt{nCONFIG} pin from high to low to high. This \texttt{nCONFIG} transition also returns the device to the configuration state.
Figure 10. Monitoring the AVST_READY Signal and Responding to Backpressure

The configuration files for Intel Stratix 10 devices can be highly compressed. During configuration, the decompression of the bit stream inside the device requires the host to pause before sending more data. The Intel Stratix 10 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.

The AVST_READY signal sent by the Intel Stratix 10 device to the host is not synchronized with the AVSTx8_CLK or AVST_CLK. To configure the Intel Stratix 10 device successfully, the host must adhere to the following constraints:

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

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

Where:

- The AVST_CLK signal comes from either PFL II IP or your Avalon-ST controller logic.
- fpga_avst_ready is the AVST_READY signal from the Intel Stratix 10 device.
- fpga_avst_ready_reg2 signal is the AVST_READY signal that is synchronous to AVST_CLK.

You must properly constrain the AVST_CLK and AVST_DATA signals at the host. Perform timing analysis on both signals between the host and Intel Stratix 10 device to ensure the Avalon-ST configuration timing specifications are met. Refer to the Avalon-ST Configuration Timing section of the Intel Stratix 10 Device Data Sheet for information about the timing specifications.

**Note:**

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

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

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

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

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

**Related Information**

- Avalon-ST Configuration Timing in Intel Stratix 10 Device Datasheet
- Avalon Interface Specifications
3.1.4. RBF Configuration File Format

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

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

Table 10. Writing 32-bit Data
For a x32 data bus, the first byte in the file is the least significant byte of the configuration double word, and the fourth byte is the most significant byte.

<table>
<thead>
<tr>
<th>Double Word = 01EE1B02</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSB: BYTE0 = 02</td>
</tr>
<tr>
<td>0000 0010</td>
</tr>
</tbody>
</table>

Table 11. Writing 16-bit Data
For a x16 data bus, the first byte in the file is the least significant byte of the configuration word, and the second byte is the most significant byte of the configuration word.

<table>
<thead>
<tr>
<th>WORD0 = 1802</th>
<th>WORD1 = 01EE</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSB: BYTE0 = 02</td>
<td>MSB: BYTE1 = 1B</td>
</tr>
<tr>
<td>LSB: BYTE2 = EE</td>
<td>MSB: BYTE3 = 01</td>
</tr>
<tr>
<td>D[7:0]</td>
<td>D[15:8]</td>
</tr>
<tr>
<td>0000 0010</td>
<td>0001 1011</td>
</tr>
<tr>
<td>D[7:0]</td>
<td>D[15:8]</td>
</tr>
<tr>
<td>1110 1110</td>
<td>0000 0001</td>
</tr>
</tbody>
</table>

3.1.5. Avalon-ST Single-Device Configuration

Refer to the Intel Stratix 10 Device Family Pin Connection Guidelines for additional information about individual pin usage and requirements.
Figure 11. Connections for Avalon-ST x8 Single-Device Configuration

Intel FPGA

nCONFIG
nSTATUS
CONF_DONE
INIT_DONE
OSC_CLK_1
MSEL[2:0]

Configuration Data Signals

Parallel Flash Loader II IP
or
Microprocessor
or
Custom Logic

fpga_data [7:0]
fpga_valid
fpga_ready
fpga_clk

Configuration Control Signals

Intel FPGA

fpga_nconfig
fpga_nstatus
fpga_conf_done

Parallel Flash Loader II IP
or
Microprocessor
or
Custom Logic

fpga_data [7:0]
fpga_valid
fpga_ready
fpga_clk

Compact Flash Interface

ADDR DATA Control
External Compact Flash Memory

.rbf
(little endian)

External Host

CPLD / FPGA

10kΩ

MSEL

Synchronizers

External Clock Source

Parallel Flash Loader II IP
or
Microprocessor
or
Custom Logic

fpga_data [7:0]
fpga_valid
fpga_ready
fpga_clk

Compact Flash Interface

ADDR DATA Control
External Compact Flash Memory

.rbf
(little endian)
Figure 12. Connections for Avalon-ST x16 Single-Device Configuration
Figure 13. Connections for Avalon-ST x32 Single-Device Configuration

- **Parallel Flash Loader II IP or Microprocessor or Custom Logic**
  - `fpga_data[31:0]`
  - `fpga_valid`
  - `fpga_ready`
  - `fpga_clk`

- **External Compact Flash Memory**
  - ADDR
  - DATA
  - Control
  - rbf
    (little endian)

- **External Clock Source**
  - `osc_clk`

- **Configuration Data Signals**
  - `AVST_DATA[31:0]`
  - `AVST_VALID`
  - `AVST_READY`
  - `AVST_CLK`

- **Configuration Control Signals**
  - `nCONFIG`
  - `nSTATUS`
  - `CONF_DONE`
  - `INIT_DONE`
  - `OSC_CLK_1`
  - `MSEL[2:0]`
  - `fpga_conf_done`
  - `fpga_nstatus`
  - `fpga_nconfig`
  - `fpga_data[31:0]`

- **Synchronizers**

- **V_{CCIO_SDM}**

- **10kΩ**

- **Intel FPGA**

- **3. Intel Stratix 10 Configuration Schemes**

Intel Stratix 10 Configuration User Guide
Notes for Figure:

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

Related Information
- MSEL Settings on page 30
- Enabling Dual-Purpose Pins on page 35
- Intel Stratix 10 Device Family Pin Connection Guidelines

3.1.6. Debugging Guidelines for the Avalon-ST Configuration Scheme

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

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

Debugging Suggestions

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

- Only assert AVST_VALID after the SDM asserts AVST_READY.
- Only assert AVST_VALID when the AVST_DATA is valid.
- Ensure that the AVST_CLK clock signal is continuous and free running until configuration completes. The AVST_CLK can stop after CONF_DONE asserts. The initialization state does not require the AVST_CLK signal.
- If using x8 mode, ensure that you use the dedicated SDM_IO pins for this interface (clock, data, valid and ready).
- If using x16 or x32 mode, power the I/O bank containing the x16 or x32 pins (I/O Bank 3A) at 1.8 V.
• Ensure you select the appropriate Avalon-ST configuration scheme in your Intel Quartus Prime Pro Edition project.
• Ensure the MSEL pins reflect this mode on the PCB.
• Verify that host device does not drive configuration pins before the Intel Stratix 10 device powers up.

Related Information
Intel Stratix 10 Debugging Guide on page 164

3.1.7. QSF Assignments for Avalon-ST x8

You can specify many Intel Quartus Prime project settings using Intel Quartus Prime Software GUI or by editing the Intel Quartus Prime Settings File (.qsf). The following assignments in the .qsf show typical settings for a Intel Stratix 10 device using Avalon-ST x8 configuration.

These settings are for a Intel Stratix 10 SmartVID device operating in PMBus slave mode which requires most of the SDM_IO pins. Refer to the Intel Stratix 10 Power Management User Guide for the PMBus constraints in master mode.

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

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

# Configuration settings
# ======================
set_global_assignment -name STRATIXV_CONFIGURATION_SCHEME "AVST X8"
set_global_assignment -name USE_CONFIGURATION_DEVICE OFF
set_global_assignment -name ERROR_CHECK_FREQUENCY_DIVISOR 256
set_global_assignment -name ENABLE_ED_CRC_CHECK ON
set_global_assignment -name MINIMUM_SEU_INTERVAL 479

# SmartVID feature PMBus settings [Slave mode settings only]
# ==============================================================
set_global_assignment -name VID_OPERATION_MODE "PMBUS SLAVE"
set_global_assignment -name PWRMGT_DEVICE_ADDRESS_IN_PMBUS_SLAVE_MODE 3F
```
You can also set the SDM_IO configuration pins using the Assignments ➤ Device ➤ Device and Pin Options ➤ Configuration ➤ Configuration Pin Options.

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

Related Information

- PMBus Master Mode
  In the Intel Stratix 10 Power Management User Guide
3.1.8. QSF Assignments for Avalon-ST x16

You can specify many Intel Quartus Prime project settings using Intel Quartus Prime Software GUI or by editing the Intel Quartus Prime Settings File (.qsf). The following assignments in the .qsf show typical settings for an Intel Stratix 10 device using Avalon-ST x16 configuration.

These settings are for an Intel Stratix 10 SmartVID device operating in PMBus slave mode which requires most of the SDM_IO pins. Refer to the *Intel Stratix 10 Power Management User Guide* for the PMBus constraints in master mode.

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

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

# Configuration settings
# ----------------------
set_global_assignment -name STRATIXV_CONFIGURATION_SCHEME "AVST X16"
set_global_assignment -name USE_CONFIGURATION_DEVICE OFF
set_global_assignment -name ERROR_CHECK_FREQUENCY_DIVISOR 256
set_global_assignment -name GENERATE_PR_RBF_FILE ON
set_global_assignment -name ENABLE_ED_CRC_CHECK ON
set_global_assignment -name MINIMUM_SEU_INTERVAL 479

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

You can also set the SDM_IO configuration pins using the Assignments ➤ Device ➤ Device and Pin Options ➤ Configuration ➤ Configuration Pin Options.
Figure 15. Set SDM_IO Configuration Pins Using the Intel Quartus Prime Software

Related Information

- PMBus Master Mode
  In the Intel Stratix 10 Power Management User Guide
3.1.9. QSF Assignments for Avalon-ST x32

You can specify many Intel Quartus Prime project settings using Intel Quartus Prime Software GUI or by editing the Intel Quartus Prime Settings File (.qsf). The following assignments in the .qsf show typical settings for a Intel Stratix 10 device using Avalon-ST x32 configuration.

These settings are for a Intel Stratix 10 SmartVID device operating in PMBus slave mode which requires most of the SDM_IO pins. Refer to the Intel Stratix 10 Power Management User Guide for the PMBus constraints in master mode.

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

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

# Configuration settings
# ------------------------
set_global_assignment -name STRATIXV_CONFIGURATION_SCHEME "AVST X32"
set_global_assignment -name USE_CONFIGURATION_DEVICE OFF
set_global_assignment -name ERROR_CHECK_FREQUENCY_DIVISOR 256
set_global_assignment -name GENERATE_PR_RBF_FILE ON
set_global_assignment -name ENABLE_ED_CRC_CHECK ON
set_global_assignment -name MINIMUM_SEU_INTERVAL 479

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

You can also set the SDM_IO configuration pins using the Assignments ➤ Device ➤ Device and Pin Options ➤ Configuration ➤ Configuration Pin Options.
```
Figure 16. Set SDM_IO Configuration Pins Using the Intel Quartus Prime Software

Related Information
- PMBus Master Mode
  In the Intel Stratix 10 Power Management User Guide
3.1.10. IP for Use with the Avalon-ST Configuration Scheme: Intel FPGA Parallel Flash Loader II IP Core

3.1.10.1. Functional Description

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

- Program configuration data into a flash memory device using JTAG interface.
- Configure the Intel Stratix 10 device with the Avalon-ST configuration scheme from the flash memory device.

*Note:* Use the Parallel Flash Loader II IP Intel FPGA IP and not the earlier Parallel Flash Loader IP with the Avalon-ST configuration scheme in Intel Stratix 10 devices.

3.1.10.1.1. Generating and Programming a .pof into CFI Flash

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

2. Using the Intel Quartus Prime Programmer to write the Intel Stratix 10 device .pof to the flash device.

**Figure 17.** Programming the CFI Flash Memory with the JTAG Interface
The PFL II IP core supports dual flash memory devices in burst read mode to achieve faster configuration times. You can connect two MT29EW CFI flash memory devices to the host in parallel using the same data bus, clock, and control signals. During FPGA configuration, the AVST_CLK frequency is four times faster than the flash_clk frequency.

**Figure 18. PFL II IP core with Dual CFI Flash Memory Devices**
The flash memory devices must have the same memory density from the same device family and manufacturer.

**Related Information**
Intel Stratix 10 GX FPGA Development Kit

**3.1.10.1.2. Controlling Avalon-ST Configuration with PFL II IP Core**
The PFL II IP core in the host determines when to start the configuration process, read the data from the flash memory device, and configure the Intel Stratix 10 device using the Avalon-ST configuration scheme.
You can use the PFL II IP core to either program the flash memory devices, configure your FPGA, or both. To perform both functions, create separate PFL II functions if any of the following conditions apply to your design:

- You modify the flash data infrequently.
- You have JTAG or In-System Programming (ISP) access to the configuration host.
- You want to program the flash memory device with non-Intel FPGA data, for example initialization storage for an ASSP.

You can use the PFL II IP core to program the flash memory device for the following purposes:
- To write the initialization data
- To store your design source code to implement the read and initialization control with the host logic

### 3.1.10.1.3. Mapping PFL II IP Core and Flash Address

The address connections between the PFL II IP core and the flash memory device vary depending on the flash memory device vendor and data bus width.
Figure 20. Flash Memory in 8-Bit Mode
The address connection between the PFL II IP core and the flash memory device are the same.

address: 24 bits

<table>
<thead>
<tr>
<th>PFL II</th>
<th>Flash Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>23</td>
</tr>
<tr>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 21. Flash Memories in 16-Bit Mode
The flash memory addresses in 16-bit flash memory shift one bit down in comparison with the flash addresses in PFL II IP core. The flash address in the flash memory starts from bit 1 instead of bit 0.

address: 23 bits

<table>
<thead>
<tr>
<th>PFL II</th>
<th>Flash Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>23</td>
</tr>
<tr>
<td>21</td>
<td>22</td>
</tr>
<tr>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Figure 22. Cypress and Micron M28, M29 Flash Memory in 8-Bit Mode
The flash memory addresses in Cypress 8-bit flash shifts one bit up. Address bit 0 of the PFL II IP core connects to data pin D15 of the flash memory.

Figure 23. Cypress and Micron M28, M29 Flash Memory in 16-Bit Mode
The address bit numbers in the PFL II IP core and the flash memory device are the same.

3.1.10.1.4. Implementing Multiple Pages in the Flash .pof
The PFL II IP core stores configuration data in a maximum of eight pages in a flash memory block.
The total number of pages and the size of each page depends on the flash density. Here are some guidelines for storing your designs to pages:

- Always store designs for different FPGA chains on different pages.
- You may choose to 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 0x000000, the next valid start address is an increment of 0x2000.
- Auto mode—allows the Intel Quartus Prime software to automatically determine the start address of the page. The Intel Quartus Prime software aligns the pages on a 128 KB boundary. If the first valid start address is 0x000000, the next valid start address is a multiple of 0x20000.

### 3.1.10.1.5. Storing Option Bits

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

The option bits contain the following information:

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

The option bit start address displays at the top of the Input files to convert section of the Convert Programming File dialog box.
You set the option bits in the PFL II IP Intel FPGA IP using the parameter editor. By default the PFL II IP displays **Flash Programming** for the **What operating mode will be used?** parameter. In this default state, the **FPGA Configuration** tab is not visible. Select either **FPGA Configuration** or **Flash Programming and FPGA Configuration** for the **What operating mode will be used** parameter on the **General** tab. The following figure shows the **FPGA Configuration** option.

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.

![Start Address for Options Bit In Flash Memory](image)

![General Tab of the PFL II IP](image)
The Intel Quartus Prime Convert Programming File tool generates the information for the .pof version when you convert the .sofs to .pofs. The value for the .pof version for Intel Stratix 10 is 0x05. The following table shows an example of page layout for a .pof using all eight pages. This example stores the .pof version at 0x80.

Table 12. Option Bits Sector Format

<table>
<thead>
<tr>
<th>Sector Offset</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00-0x03</td>
<td>Page 0 start address</td>
</tr>
<tr>
<td>0x04-0x07</td>
<td>Page 0 end address</td>
</tr>
<tr>
<td>0x08-0x0B</td>
<td>Page 1 start address</td>
</tr>
<tr>
<td>0x0C-0x0F</td>
<td>Page 1 end address</td>
</tr>
</tbody>
</table>

continued...
### Sector Offset | Value
--- | ---
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(8) | .pof version
0x81–0xFF | Reserved

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

**Related Information**

PFL II Parameters on page 76

---

(8) The .pof version occupies only one byte in the option bits sector.
3.1.10.1.6. Verifying the Option Bit Start and End Addresses

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

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

**Table 13. Start Address Bit Content**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:11</td>
<td>21</td>
<td>Addressable start address</td>
</tr>
<tr>
<td>10:1</td>
<td>10</td>
<td>Reserved bits</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Page valid bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0=Valid</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1=Error</td>
</tr>
</tbody>
</table>

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.

**Table 14. End Address Bit Content**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>32</td>
<td>Addressable end address</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sector Offset</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00 – 0x03</td>
<td>0x00004000</td>
</tr>
<tr>
<td>0x04 – 0x07</td>
<td>0x00196E30</td>
</tr>
<tr>
<td>0x08 – 0x0B</td>
<td>0x001C0000</td>
</tr>
<tr>
<td>0x0C – 0x0F</td>
<td>0x00352E30</td>
</tr>
</tbody>
</table>
For Page 0 if you append the start address bits[31:11] with 13'b0000000000000, the result is 32'b00000000000000010000000000000000 = 0x10000.

If you append the end address 0x00196E0 with 2'b11, the result is 26'b00011001011011001100011 = 0x65B8C3.

For Page 1 if you append the start address with 13'b0000000000000, the result is 32'b0000000000011100000000000000000000 = 0x700000.

If you append end address 0x00352E30 with 2'b11, the result is 32'b000000011010100101100011000011 = 0xD4B8C3.

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

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

The following figure shows an sample layout of a .pof with three pages. The end addresses depend on the density of the flash memory device. For different density devices refer to the Byte Address Range for CFI Flash Memory Devices with Different Densities table below. The option bits follow the configuration data in memory.
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 PFL II IP core sets the Page-Valid bits after successfully programming the pages.

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

<table>
<thead>
<tr>
<th>CFI Device (Megabit)</th>
<th>Address Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0x00000000–0x00FFFFFF</td>
</tr>
<tr>
<td>16</td>
<td>0x00000000–0x01FFFFFF</td>
</tr>
<tr>
<td>32</td>
<td>0x00000000–0x03FFFFFF</td>
</tr>
<tr>
<td>64</td>
<td>0x00000000–0x07FFFFFF</td>
</tr>
</tbody>
</table>

*continued...*
### 3.1.10.2. Using PFL II IP Core

#### 3.1.10.2.1. Converting .sof to .pof File

Follow these steps to convert the .sof file to a .pof:

1. On the File menu click Convert Programming Files.
2. For Programming file type specify .pof and name the file.
3. For Configuration device select the CFI flash memory device with the correct density. For example, CFI_1Gb is a CFI device with 1-Gigabit (Gb) capacity.
4. For Mode, select the configuration scheme that you plan to use.
5. To add the configuration data, under Input files to convert, select SOF Data.
6. Click Add File and browse to the .sof or .sos the design requires.

You can write more than one .sof to the same page if your design includes a chain of FPGAs. The order of the .sof files must be the same as the device order of the JTAG chain. To store the data for additional .sof files on a different page, click Add SOF page.

7. Select SOF Data and click Properties to set the page number and set the Selected pages comment. Under Address mode for selected pages, select from the following options:
   - Select Auto to let the Intel Quartus Prime software automatically set the start address for that page.
   - Select Block to specify the start and end addresses.
   - Select Start to specify the start address only.

Click OK.
8. You can also store Hexadecimal (Intel-Format) Output File (.hex) data in the flash memory device. You can use hexadecimal files to store data for a number of reasons, including to initialize memory and to create application counters.
   a. On the **Input files to convert** sub-window of the **Convert Programming Files** dialog box, select **Add Hex Data**.
   b. In the **Add Hex Data** dialog box, select either absolute or relative addressing mode.
      - For absolute addressing mode, the Programmer writes the .hex data to the flash memory device at the same address location listed in the .hex.
      - For relative addressing mode, you specify a start address. The Programmer writes the .hex data to the flash memory device using the value you specify for **Set start address**. The software maintains differences between the addresses. If you do not provide a value for **Set start address**, the software selects an address.

   *Note:* You can add non-configuration data to the .pof by selecting the .hex that contains your data when creating the flash memory device .pof.
9. Under **Hex file** browse to and select the .hex file.

10. Click **Options/Boot info** to specify the start address to store the option bits.

   This start address must be identical to the address you specify for **What is the byte address of the option bits, in hex?** when specifying the PFL II IP parameters. Ensure that the option bits sector does not overlap with the configuration data pages and that the start address is on an 8 KB boundary.
11. To compress the .sof file that is stored in the .pof, turn on the Compression mode in the Options dialog box and click OK.

12. Click Generate to create the .pof.

3.1.10.2.2. Creating Separate PFL II Functions

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

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

3.1.10.2.3. Programming CPLDs and Flash Memory Devices Sequentially

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

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

3.1.10.2.4. Programming CPLDs and Flash Memory Devices Separately

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

1. Open the Programmer and click Add File.
2. In the Select Programming File, add the targeted .pof, and click OK.
3. Check the boxes under the Program/Configure column of the .pof.
4. Click Start to program the CPLD.
5. After the programming progress bar reaches 100%, click Auto Detect.
   For example, if you are using dual Micron or Macronix flash devices, the programmer window shows a dual chain in your setup. Alternatively, you can add the flash memory device to the programmer manually. Right-click the CPLD .pof and click Attach Flash Device. In the Select Flash Device dialog box, select the device of your choice.
6. Right-click the flash memory device density and click Change File.
Note: For designs with more than one flash device, you must select the density that is equivalent to the sum of the densities of all devices. For example, if the design includes two 512-Mb CFI flash memory devices, select CFI 1 Gbit.

7. Select the .pof generated for the flash memory device. The Programmer attaches the .pof for the flash memory device to the .pof of the CPLD.

8. Check the boxes under the Program/Configure column for the added .pof and click Start to program the flash memory devices.

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

3.1.10.2.5. Defining New CFI Flash Memory Device

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

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

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>New</td>
<td>Add a new Intel- or AMD-compatible CFI flash memory device into the PFL II-supported flash database.</td>
</tr>
<tr>
<td>Edit</td>
<td>Edit the parameters of the newly added Intel- or AMD-compatible CFI flash memory device in the PFL II-supported flash database.</td>
</tr>
<tr>
<td>Remove</td>
<td>Remove the newly added Intel- or AMD-compatible CFI flash memory device from the PFL II-supported flash database.</td>
</tr>
</tbody>
</table>

2. To add a new CFI flash memory device or edit the parameters of the newly added CFI flash memory device, select New or Edit. The New CFI Flash Device dialog box appears.

3. In the New CFI Flash Device dialog box, specify or update the parameters of the new flash memory device. You can obtain the values for these parameters from the data sheet of the flash memory device manufacturer.
Figure 32. Using the Programmer Edit Menu to Define a New Flash Device

Table 17. Parameter Settings for New CFI Flash Device

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFI flash device name</td>
<td>Define the CFI flash name</td>
</tr>
<tr>
<td>CFI flash device ID</td>
<td>Specify the CFI flash identifier code</td>
</tr>
<tr>
<td>CFI flash manufacturer ID</td>
<td>Specify the CFI flash manufacturer identification number</td>
</tr>
<tr>
<td>CFI flash extended device ID</td>
<td>Specify the CFI flash extended device identifier, only applicable for AMD-compatible CFI flash memory device</td>
</tr>
<tr>
<td>Flash device is Intel compatible</td>
<td>Turn on the option if the CFI flash is Intel compatible</td>
</tr>
<tr>
<td>Typical word programming time</td>
<td>Typical word programming time value in µs unit</td>
</tr>
<tr>
<td>Maximum word programming time</td>
<td>Maximum word programming time value in µs unit</td>
</tr>
<tr>
<td>Typical buffer programming time</td>
<td>Typical buffer programming time value in µs unit</td>
</tr>
<tr>
<td>Maximum buffer programming time</td>
<td>Maximum buffer programming time value in µs unit</td>
</tr>
</tbody>
</table>
Note: You must specify either the word programming time parameters, buffer programming time parameters, or both. Do not leave both programming time parameters with the default value of zero.

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

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

3.1.10.3. PFL II Parameters

Table 18. PFL II General Parameters

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

Table 19. PFL II Flash Interface Setting Parameters

<table>
<thead>
<tr>
<th>Options</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>How many flash devices will be used?</td>
<td>• 1–16</td>
<td>Specifies the number of flash memory devices connected to the PFL II IP core.</td>
</tr>
<tr>
<td>What's the largest flash device that will be used?</td>
<td>• 8 Mbit–4 Gbit</td>
<td>Specifies the density of the flash memory device to be programmed or used for FPGA configuration. If you have more than one flash memory device connected to the PFL II IP core, specify the largest flash memory device density. For dual CFI flash, select the density that is equivalent to the sum of the density of two flash memories. For example, if you use two 512-Mb CFI flashes, you must select CFI 1 Gbit.</td>
</tr>
<tr>
<td>What is the flash interface data width</td>
<td>• 8</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>• 16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 32</td>
<td></td>
</tr>
</tbody>
</table>
Options | Value | Description
--- | --- | ---
Select the flash data width that is equivalent to the sum of the data width of two flash memories. For example, if you are targeting dual solution, you must select **32 bits** because each CFI flash data width is 16 bits.

Allow user to control FLASH_NRESET pin
- **On**
- **Off**

Creates a `FLASH_NRESET` pin in the PFL II IP core to connect to the reset pin of the flash memory device. A low signal resets the flash memory device. In burst mode, this pin is available by default.

When using a Cypress GL flash memory, connect this pin to the `RESET` pin of the flash memory.

### Table 20. PFL II Flash Programming Parameters

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

### Table 21. PFL II FPGA Configuration Parameters

<table>
<thead>
<tr>
<th>Options</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>What is the external clock frequency?</strong></td>
<td>Provide the frequency of your external clock.</td>
<td>Specifies the user-supplied clock frequency for the IP core to configure the FPGA. The clock frequency must not exceed two times the maximum clock (<code>AVST_CLK</code>) frequency the FPGA can use for configuration. The PFL II IP core can divide the frequency of the input clock maximum by two.</td>
</tr>
<tr>
<td><strong>What is the flash access time?</strong></td>
<td>Provide the access time from the flash data sheet.</td>
<td>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.</td>
</tr>
</tbody>
</table>

*continued...*
<table>
<thead>
<tr>
<th>Options</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is the byte address of the option bits, in hex?</td>
<td>Provide the byte address of the option bits.</td>
<td>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.</td>
</tr>
<tr>
<td>Which FPGA configuration scheme will be used?</td>
<td>• Avalon-ST x8</td>
<td>Specifies the width of the Avalon-ST interface.</td>
</tr>
<tr>
<td></td>
<td>• Avalon-ST x16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Avalon-ST x32</td>
<td></td>
</tr>
<tr>
<td>What should occur on configuration failure?</td>
<td>• Halt</td>
<td>Configuration behavior after configuration failure.</td>
</tr>
<tr>
<td></td>
<td>• Retry same page</td>
<td>• If you select Halt, the FPGA configuration stops completely after failure.</td>
</tr>
<tr>
<td></td>
<td>• Retry from fixed address</td>
<td>• If you select Retry same page, after failure, the PFL II IP core reconfigures the FPGA with data from the page that failed.</td>
</tr>
<tr>
<td></td>
<td>—</td>
<td>• If you select Retry from fixed address, the PFL II IP core reconfigures the FPGA at a fixed address.</td>
</tr>
<tr>
<td>What is the byte address to retry from failure</td>
<td>—</td>
<td>If you select Retry from fixed address for configuration failure option, this option specifies the flash address the PFL II IP core to reads from.</td>
</tr>
<tr>
<td>Include input to force reconfiguration</td>
<td>• On</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>• Off</td>
<td>Includes the optional pfl_nreconfigure reconfiguration input pin to enable reconfiguration of the FPGA.</td>
</tr>
<tr>
<td>Enable watchdog timer on Remote System Update support</td>
<td>• On</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Off</td>
<td></td>
</tr>
</tbody>
</table>
3. Intel Stratix 10 Configuration Schemes

Table 22. PFL II Signals

<table>
<thead>
<tr>
<th>Pin</th>
<th>Type</th>
<th>Weak Pull-Up</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>pfl_nreset</td>
<td>Input</td>
<td>—</td>
<td>Asynchronous reset for the PFL II IP core. Pull high to enable FPGA configuration. To prevent FPGA configuration, pull low when you do not use the PFL II IP core. This pin does not affect the PFL II IP flash programming functionality.</td>
</tr>
<tr>
<td>pfl_flash_access_granted</td>
<td>Input</td>
<td>—</td>
<td>For system-level synchronization. A processor or any arbiter that controls access to the flash drives this input pin. To use the PFL II IP core function as the flash master pull this pin high. Driving the pfl_flash_access_granted pin low prevents the JTAG interface from accessing the flash and FPGA configuration.</td>
</tr>
<tr>
<td>pfl_clk</td>
<td>Input</td>
<td>—</td>
<td>User input clock for the device. This is the frequency you specify for the <strong>What is the external clock frequency?</strong> parameter on the Configuration tab of the PFL II IP. This frequency must not be higher than the maximum DCLK frequency you specify for FPGA during configuration. This pin is not available if you are only using the PFL II IP for flash programming.</td>
</tr>
<tr>
<td>Pin</td>
<td>Type</td>
<td>Weak Pull-Up</td>
<td>Function</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------</td>
<td>----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>fpga_pgm[]</td>
<td>Input</td>
<td>—</td>
<td>Determines the page for the configuration. This pin is not available if you are only using the PFL II IP for flash programming.</td>
</tr>
<tr>
<td>fpga_conf_done</td>
<td>Input</td>
<td>10 kΩ Pull-Up Resistor</td>
<td>Connects to the CONF_DONE pin of the FPGA. The FPGA releases the pin high if the configuration is successful. During FPGA configuration, this pin remains low. This pin is not available if you are only using the PFL II IP for flash programming.</td>
</tr>
<tr>
<td>fpga_nstatus</td>
<td>Input</td>
<td>10 kΩ Pull-Up Resistor</td>
<td>Connects to the nSTATUS pin of the FPGA. This pin is high before the FPGA configuration begins and must stay high during FPGA configuration. If a configuration error occurs, the FPGA pulls this pin low and the PFL II IP core stops reading the data from the flash memory device. This pin is not available if you are only using the PFL II IP for flash programming.</td>
</tr>
<tr>
<td>pfl_nreconfigure</td>
<td>Input</td>
<td>—</td>
<td>When low initiates FPGA reconfiguration. To implement manual control of reconfiguration connect this pin to a switch. You can use this input to write your own logic in a CPLD to trigger reconfiguration via the PFL II IP. When FPGA reconfiguration begins, the fpga_nconfig pin is pulled low to reset the FPGA device. The pfl_clk pin registers this signal. This pin is not available if you are only using the PFL II IP for flash programming.</td>
</tr>
<tr>
<td>pfl_flash_access_request</td>
<td>Output</td>
<td>—</td>
<td>For system-level synchronization. When necessary, this pin connects to a processor or an arbiter. The PFL II IP core drives this pin high when the JTAG interface accesses the flash or the PFL II IP configures the FPGA. This output pin works in conjunction with the flash_noe and flash_nwe pins.</td>
</tr>
<tr>
<td>flash_addr[]</td>
<td>Output</td>
<td>—</td>
<td>The flash memory address. The width of the address bus depends on the density of the flash memory device and the width of the flash_data bus. Intel recommends that you turn On the Set flash bus pins to tri-state when not in use option in the PFL II.</td>
</tr>
<tr>
<td>flash_data[]</td>
<td>Input or Output (bidirectional pin)</td>
<td>—</td>
<td>Bidirectional data bus to transmit or receive 8-, 16-, or 32-bit data. Intel recommends that you turn On the Set flash bus pins to tri-state when not in use option in the PFL II. (9)</td>
</tr>
<tr>
<td>flash_nce[]</td>
<td>Output</td>
<td>—</td>
<td>Connects to the nCE pin of the flash memory device. A low signal enables the flash memory device. Use this bus for multiple flash memory device support. The flash_nce pin connects to each nCE pin of all the connected flash memory devices. The width of this port depends on the number of flash memory devices in the chain.</td>
</tr>
</tbody>
</table>

(9) Intel recommends that you do not insert logic between the PFL II pins and the host I/O pins, especially on the flash_data and fpga_nconfig pins.
<table>
<thead>
<tr>
<th>Pin</th>
<th>Type</th>
<th>Weak Pull-Up</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>flash_nwe</td>
<td>Output</td>
<td>—</td>
<td>Connects to the nWE pin of the flash memory device. When low enables write operations to the flash memory device.</td>
</tr>
<tr>
<td>flash_noe</td>
<td>Output</td>
<td>—</td>
<td>Connects to the nOE pin of the flash memory device. When low enables the outputs of the flash memory device during a read operation.</td>
</tr>
<tr>
<td>flash_clk</td>
<td>Output</td>
<td>—</td>
<td>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.</td>
</tr>
<tr>
<td>flash_nadv</td>
<td>Output</td>
<td>—</td>
<td>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.</td>
</tr>
<tr>
<td>flash_nreset</td>
<td>Output</td>
<td>—</td>
<td>Connects to the reset pin of the flash memory device. A low signal resets the flash memory device.</td>
</tr>
<tr>
<td>fpga_nconfig</td>
<td>Open Drain Output</td>
<td>10-kW Pull-Up Resistor</td>
<td>Connects to the nCONFIG pin of the FPGA. A low pulse resets the FPGA and initiates configuration. These pins are not available for the flash programming option in the PFL II IP core. (9)</td>
</tr>
<tr>
<td>pfl_reset_watchdog</td>
<td>Input</td>
<td>—</td>
<td>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.</td>
</tr>
<tr>
<td>pfl_watchdog_error</td>
<td>Output</td>
<td>—</td>
<td>When high indicates an error condition to the watchdog timer.</td>
</tr>
</tbody>
</table>

**Related Information**

Avalon Interface Specifications

**3.2. AS Configuration**

In AS configuration schemes, the SDM block in the Intel Stratix 10 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\textsubscript{CCIO\_SDM} of the Intel Stratix 10 device.

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

**Table 23. Intel Stratix 10 Configuration Data Width, Clock Rates, and Data Rates**

<table>
<thead>
<tr>
<th>Mode</th>
<th>Data Width (bits)</th>
<th>Max Clock Rate</th>
<th>Max Data Rate</th>
<th>MSEL[2:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>Active Serial (AS)</td>
<td>4</td>
<td>133 MHz</td>
<td>532 Mbps</td>
</tr>
</tbody>
</table>

**Related Information**

- Can I use 3rd party QSPI flash devices for Active Serial configuration of Intel Stratix 10 devices?
- **AS Configuration Timing in Intel Stratix 10 Devices**
  - For timing parameter minimum, typical, and maximum values.

**3.2.1. AS Configuration Scheme Hardware Components and File Types**

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

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

The following block diagram illustrates the components and design flow using the AS configuration scheme.
In addition to AS programming using a .jic, the Programmer supports direct programming of the Quad SPI flash using a .pof as shown in AS Programming Using Intel Quartus Prime or Third-Party Programmer.

### Table 24. Output File Types

<table>
<thead>
<tr>
<th>Programming File Type</th>
<th>Extension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JTAG Indirect Configuration File</td>
<td>.jic</td>
<td>The .jic enables serial flash programming via Intel FPGA JTAG pins. This file type is available only for AS configuration. Before programming the flash, the Programmer configures the FPGA with the Serial Flash Helper Design.</td>
</tr>
</tbody>
</table>

**Related Information**
- Can I use 3rd party QSPI flash devices for Active Serial configuration of Intel Stratix 10 devices?
- Programming Serial Flash Devices using the AS Interface on page 88
3.2.2. AS Single-Device Configuration

Refer to the Intel Stratix 10 Device Family Pin Connection Guidelines for additional information about individual pin usage and requirements.

Figure 34. Connections for AS x4 Single-Device Configuration
3. Intel Stratix 10 Configuration Schemes

3.2.3. AS Using Multiple Serial Flash Devices

Intel Stratix 10 devices support one AS x4 flash memory device for AS configuration and up to three AS x4 flash memories for use with HPS data storage. The MSEL pins are dual-purpose and operate as MSEL only during POR state. After the FPGA device enters user mode, you can repurpose the MSEL pins as chip select pins. You must ensure appropriate chip select pin connections to the configuration AS x4 flash memory and HPS AS x4 flash memory. Each flash device has a dedicated AS_nCSO pin but shares other pins.

Refer to the Intel Stratix 10 Device Family Pin Connection Guidelines for additional information about individual pin usage and requirements.
Figure 35. Connections for AS Configuration with Multiple Serial Flash Devices

- **Pin 1**
  - TCK
  - TDO
  - TMS
  - TDI
  - JTAG Configuration Pins
  - VCCIO_SDM
  - GND
  - OPEN
  - OPEN
  - OPEN
  - OPEN

- **Optional Monitoring**
  - MSEL

- **Configuration Data Signals**
  - DATA[3:0]
  - DCLK
  - CS
  - FPGA Image (.rpd)

- **HPS Data Signals**
  - AS_DATA[3:0]
  - AS_CLK
  - AS_nCS0[0]
  - AS_nCS0[1]
  - AS_nCS0[2]
  - AS_nCS0[3]

- **Configuration Control Signals**
  - nCONFIG
  - nSTATUS
  - CONF_DONE
  - INIT_DONE
  - OSC_CLK_1
  - MSEL[2:0]

- **HPS AS x4 Memory**
  - DATA[3:0]
  - DCLK
  - CS

- **Config AS x4 Memory**
  - DATA[3:0]
  - DCLK
  - CS

- **Intel FPGA**
  - 10kΩ

Download cable 10 pin male header (JTAG mode)
3M Part number : 2510-6002UB
Note: When using multiple flash devices, the clock frequency must be reduced. Refer to the *Intel Stratix 10 Device Datasheet* for more information.

**Related Information**
- MSEL Settings on page 30
- Intel Stratix 10 Device Datasheet (Core and HPS)
- Intel Stratix 10 Device Family Pin Connection Guidelines

### 3.2.4. AS Configuration Timing Parameters

**Figure 36.** AS Configuration Serial Output Timing Diagram

![AS Configuration Serial Output Timing Diagram](image1)

**Figure 37.** AS Configuration Serial Input Timing Diagram

![AS Configuration Serial Input Timing Diagram](image2)

Note: For more information about the timing parameters, refer to the *Intel Stratix 10 Device Datasheet*. 
3.2.5. Maximum Allowable External AS_DATA Pin Skew Delay Guidelines

You must minimize the skew on the AS data pins.

Skew delay includes the following elements:
- The delay due to the differences in board traces lengths on the PCB
- The capacitance loading of the flash device

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Frequency</th>
<th>Min</th>
<th>Typical</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{ext_skew}$</td>
<td>Skew delay for AS_DATA for the AS_CLK frequency specified</td>
<td>133 MHz</td>
<td>—</td>
<td>—</td>
<td>3.60</td>
</tr>
<tr>
<td></td>
<td></td>
<td>125 MHz</td>
<td>—</td>
<td>—</td>
<td>4.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>115 MHz</td>
<td>—</td>
<td>—</td>
<td>4.20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>108 MHz</td>
<td>—</td>
<td>—</td>
<td>4.60</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100 MHz</td>
<td>—</td>
<td>—</td>
<td>5.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;100 MHz</td>
<td>—</td>
<td>—</td>
<td>5.0</td>
</tr>
</tbody>
</table>

3.2.6. Programming Serial Flash Devices

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

You have the following two in-system programming options:
- Active Serial
- JTAG

3.2.6.1. Programming Serial Flash Devices using the AS Interface

When you select AS programming the Intel Quartus Prime software or any supported third-party software programs the configuration data directly into the serial flash device.
You must set MSEL to JTAG. When MSEL is set to JTAG, the SDM tristates the following AS pins: AS_CLK, AS_DATA0-AS_DATA3, and AS_nCS0-AS_nCS3. The Intel Quartus Prime Programmer programs the flash memory devices via the AS header.

**Figure 38. AS Programming Using Intel Quartus Prime or Third-Party Programmer**
3.2.6.2. Programming Serial Flash Devices using the JTAG Interface

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

Figure 39. Programming Your Serial Configuration Device Using JTAG and SDM Emulation of AS
Figure 40. Connections for Programming the Serial Flash Devices using the JTAG Interface

Intel recommends using the JTAG interface to prepare the Quad SPI flash device for later use in AS mode.
This configuration scheme includes the following steps:

1. In the Intel Quartus Prime Programmer, select the JTAG programming mode and initiate programming by clicking Start.
2. The Programmer drives .jic configuration data to the board using the JTAG header connection.
3. The programmer first configures the SDM with configuration firmware. Then, the SDM drives configuration data from the programmer to the AS x4 flash device using SDM_IOs.
4. To use the Intel Stratix 10 device in AS mode after successful programming of the flash device, set the MSEL pins to either AS fast or AS normal mode and power cycle the device.

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

### 3.2.7. Serial Flash Memory Layout

Serial flash devices store the configuration data in sections.

The following diagram illustrates sections of a non-HPS Intel Stratix 10 configuration data mapping in a serial flash device. Refer to *Intel Stratix 10 SoC FPGA Bitstream Sections* of the HPS Technical Reference Manual for more information about flash memory layout for HPS devices.

**Figure 41. Serial Flash Memory Layout Diagram**

```
<table>
<thead>
<tr>
<th>Start Address 32'd0</th>
<th>32'd256k</th>
<th>32'd512k</th>
<th>32'd768k</th>
<th>32'd1024k</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firmware Section</td>
<td>Firmware Section</td>
<td>Firmware Section</td>
<td>Firmware Section</td>
<td>Dynamic Section (I/O Configuration)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>End Address</td>
<td>(Design dependent)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Firmware section is static and Quartus Prime version dependent.
If you use a third-party programmer to program an `.rpd`, ensure that the configuration data is stored starting from address 0 of the serial flash device. If you use `.jic` or `.pof` files, the Intel Stratix 10 Programmer automatically programs the configuration data starting from address 0 of the serial flash device.

Intel currently support the following listed Supported Flash Devices for Intel Stratix 10 10

**Related Information**

Intel Stratix 10 SoC FPGA Bitstream Sections

**3.2.7.1. Understanding Quad SPI Flash Byte-Addressing**

At power-on the SDM operates from bootROM. The SDM loads configuration firmware from Quad SPI flash using 3-byte addressing. Once loaded, the SDM configures the Quad SPI flash to operate in 4-byte addressing mode and continues to load the rest of the bitstream until configuration completes.

Intel Stratix 10 devices support the following third-party flash devices operating at 1.8 V:

- Macronix MX66U 512 MB, 1 and 2 gigabytes (GB)
- Macronix MX25U 128 megabyte (MB), 256 MB, and 512 MB
- Micron MT25QU 128 MB, 256 MB, 512 MB, 1 GB, and 2 GB

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

**3.2.8. AS_CLK**

The Intel Stratix 10 device drives AS_CLK to the serial flash device. An internal oscillator or the external clock that drives the OSC_CLK_1 pin generates AS_CLK. Using an external clock source allows the AS_CLK to run at a higher frequency. If you provide a 25 MHz, 100 MHz, or 125 MHz clock to the OSC_CLK_1 pin, the AS_CLK can run up to 133 MHz.

Set the maximum required frequency for the AS_CLK pin in the Intel Quartus Prime software as described in Active Serial Configuration Software Settings on page 94. The AS_CLK pin runs at or below your selected frequency.
Table 26. Supported configuration clock source and AS_CLK Frequencies in Intel Stratix 10 Devices

<table>
<thead>
<tr>
<th>Configuration Clock Source</th>
<th>AS_CLK Frequency (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal oscillator</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>58</td>
</tr>
<tr>
<td></td>
<td>77</td>
</tr>
<tr>
<td></td>
<td>115</td>
</tr>
<tr>
<td>OSC_CLK_1</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>71.5</td>
</tr>
<tr>
<td></td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>108</td>
</tr>
<tr>
<td></td>
<td>125</td>
</tr>
<tr>
<td></td>
<td>133</td>
</tr>
</tbody>
</table>

You can set the frequency of the AS clock using a QSF assignment. Here are the possible assignments:

```
# INTERNAL OSCILLATOR CLOCK
set_global_assignment -name ACTIVE_SERIAL_CLOCK AS_FREQ_25MHZ_IOSC
set_global_assignment -name ACTIVE_SERIAL_CLOCK AS_FREQ_58MHZ_IOSC
set_global_assignment -name ACTIVE_SERIAL_CLOCK AS_FREQ_77MHZ_IOSC
set_global_assignment -name ACTIVE_SERIAL_CLOCK AS_FREQ_115MHZ_IOSC
```

3.2.9. Active Serial Configuration Software Settings

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

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

1. On the Assignments menu, click Device.
2. In the Device and Pin Options select the Configuration category.
   a. Select Active Serial x4 from the Configuration scheme drop down menu.
b. Select **Auto** or **1.8 V** in the **Configuration device I/O voltage** drop-down list.

c. Select the AS clock frequency from the **Active serial clock source** drop-down list.

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

**Related Information**

Can I use 3rd party QSPI flash devices for Active Serial configuration of Intel Stratix 10 devices?

### 3.2.10. Intel Quartus Prime Programming Steps

#### 3.2.10.1. Generating Programming Files using Convert Programming Files

The Intel Quartus Prime **Convert Programming File** dialog box converts the .sof input file to a .pof, .jic, or .rpd file.

**Note:** If you are generating an .rpd for remote system update (RSU), you must follow the instructions in Generating a Single RSU Image on page 134 in the Remote System Update chapter. This procedure generates flash programming files for Intel Stratix 10 devices.
To convert the programming files, complete the following steps:

1. On the File menu, click **Convert Programming Files**.
2. Under **Output programming file**, select appropriate file type for your design. The AS scheme supports the **Programmer Object File (.pof)**, **JTAG Indirect Configuration File (.jic)**, and **Raw Programming Data File (.rpd)** file types.
3. In the **Mode** list, select **Active Serial x4**.
4. 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, click **Option/Boot Info** button. In the **Options** dialog box, set the RPD File Endianness to **Big Endian**.

**Figure 42. Specifying RPD Bit-Level Endianness**

5. In the **File name** field, specify the file name for the programming file you want to create.
6. Under **Advanced** to generate a Memory Map File (.map), turn on **Create Memory Map File (Generate output_file.map)** This option is not available for .rpd files.
7. To generate a Raw Programming Data (.rpd), turn on **Create config data RPD (Generate output_file_auto.rpd)**.
8. For .jic output, select **Flash Loader** and click **Add device**. Select your device family and device name and click **OK**.
9. You can add the .sof on the **Input files to convert** list.
3. Intel Stratix 10 Configuration Schemes
UG-S10CONFIG | 2019.04.10

Figure 43. AS Convert Programming File Options for .jic Generation

10. For .rpd generation, you can add the .pof file in the Input files to convert list as the source file to generate the .rpd file.

11. Click Generate to generate related programming file.

3.2.10.2. Programming .pof files into Serial Flash Device

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

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

3.2.10.3. Programming .jic files into Serial Flash Device

To program the .jic into the serial flash device through the JTAG interface, perform the following steps:
1. In the Programmer window, click Hardware Setup and select the desired download cable.
2. In the Mode list, select JTAG.
3. Select the device to be programmed and click Add File.
4. Select the .jic to be programmed to the selected device.
5. Click Start to start programming.

3.2.11. Debugging Guidelines for the AS Configuration Scheme

The AS configuration scheme operation is like earlier device families. However, there is one significant difference. Intel Stratix 10 devices using AS mode, try to load a firmware section from addresses 0, 256k, 512k and 768k in the serial flash device connected to the CS0 pin. The firmware section is static for a particular Intel Quartus Prime Pro Edition release.

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

SDM tristates AS pins, AS_CLK, AS_DATA0-AS_DATA3, and AS_nCS0-AS_nCS3, only when the device powers on if you set MSEL to JTAG. If MSEL is either AS fast or normal, the SDM drives the AS pins until you power cycle the Intel Stratix 10 device. Unlike earlier device families, the AS pins are not tristated when the device enters user mode.

The AS configuration scheme has power-on requirements. If you use AS Fast mode and are not concerned about 100 ms PCIe link training requirement, you must still ramp the VCCIO_SDM supply within 18 ms. This ramp-up requirement ensures that the AS x4 device is within its operating voltage range when the Intel Stratix 10 device begins assessing the AS x4 device.

When using AS fast mode, all power supplies to the Intel Stratix 10 device must be fully ramped-up to the recommended operating conditions within 10 ms. To meet the PCIe 100 ms power-up-to-active time requirement for CvP, the VCCIO_SDM power to the Intel Stratix 10 device must be at the recommended operating range within 10 ms.
Debugging Suggestions

Here are some debugging tips for the AS configuration scheme:

- Ensure that the boot address for your configuration image is correctly defined when generating the programming file for the flash. The boot address defaults to 0 for AS configuration.
- Ensure that the design meets the power-supply ramp requirements for fast AS mode. If using fast mode, $V_{CCIO_{SDM}}$ must ramp up within 18 ms.
- Ensure that the flash is powered up and ready to be accessed when the Intel Stratix 10 device exits power-on reset.
- If you are using an external clock source for configuration, ensure the OSC_CLK_1 pin is fed correctly, and the frequency matches the frequency you set for the OSC_CLK_1 in your Intel Quartus Prime Pro Edition project.
- Ensure the MSEL pins reflect the correct AS configuration scheme.
- If the AS configuration is failing due to a corrupt image inside the serial flash device, change the MSEL pins to JTAG only mode, verify that configuration is successful over JTAG. Then, erase and reprogram the serial flash device.
- If you are using AS x4 flash memories, ensure that you use AS Fast mode, if you are not concerned about 100 ms PCIe linkup, you must still ramp the $V_{CCIO_{SDM}}$ supply within 18 ms. This ramp-up requirement ensures that the AS x4 device is within its operating voltage range when the Intel Stratix 10 device begins to access it.
- Check endianness of the .rpd if using a third-party programmer to program Quad SPI device. You should generate the .rpd as big endian.

Related Information

Intel Stratix 10 Debugging Guide on page 164

3.2.12. QSF Assignments for AS

You can specify many Intel Quartus Prime project settings using Intel Quartus Prime Software GUI or by editing the Intel Quartus Prime Settings File (.qsf). The following assignments in the .qsf show typical settings for a Intel Stratix 10 device using AS configuration.

These settings are for a Intel Stratix 10 SmartVID device operating in PMBus slave mode which requires most of the SDM_IO pins. Refer to the Intel Stratix 10 Power Management User Guide for the PMBus constraints in master mode.

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

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

# Configuration settings
# ---------------------
set_global_assignment -name STRATIXV_CONFIGURATION_SCHEME "ACTIVE SERIAL X4"
set_global_assignment -name USE_CONFIGURATION_DEVICE ON
set_global_assignment -name ERROR_CHECK_FREQUENCY_DIVISOR 256
set_global_assignment -name GENERATE_PR_RBF_FILE ON
set_global_assignment -name ENABLE_ED_CRC_CHECK ON
set_global_assignment -name MINIMUM_SEU_INTERVAL 479

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

You can also set the SDM_IO configuration pins using the Assignments ➤ Device ➤ Device and Pin Options ➤ Configuration ➤ Configuration Pin Options.
Figure 44. Set SDM_IO Configuration Pins Using the Intel Quartus Prime Software

Related Information

- PMBus Master Mode
  In the Power Management User Guide
3.3. Configuration from SD/MMC

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

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

Table 27. Intel Stratix 10 Configuration Data Width, Clock Rates, and Data Rates

<table>
<thead>
<tr>
<th>Mode</th>
<th>Data Width (bits)</th>
<th>Max Clock Rate</th>
<th>Max Data Rate</th>
<th>MSEL[2:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active SD/MMC</td>
<td>4 or 8</td>
<td>50 MHz</td>
<td>400 Mbps</td>
<td>3'b100</td>
</tr>
</tbody>
</table>

Related Information

- MSEL Settings on page 30
- SD MCC Configuration Timing in Intel Stratix 10 Devices

3.3.1. SD/MMC Single-Device Configuration

Refer to the Intel Stratix 10 Device Family Pin Connection Guidelines for additional information about individual pin usage and requirements.
Figure 45. Connections for SD/MMC Single-Device Configuration

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

Related Information
Intel Stratix 10 Device Family Pin Connection Guidelines
3.4. JTAG Configuration

JTAG-chain device programming is ideal during development. JTAG-chain device configuration uses the JTAG pins to configure the Intel Stratix 10 FPGA directly with the .sof file. Configuration using the JTAG device chain is faster because it does not require you to program an external flash memory. You can also use JTAG to reprogram if image stored in memory is corrupt, preventing the device from configuring using the AS scheme.

The Intel Quartus Prime software generates a .sof containing the FPGA design information. You can use the .sof with a JTAG programmer to configure the Intel Stratix 10 device. The Intel FPGA Download Cable II and the Intel FPGA Ethernet Cable both can support the VCCIO_SDM supply at 1.8 V. Alternatively, you can use the JamSTAPL Format File (.jam) or Jam Byte Code File (.jbc) for JTAG configuration.

Intel Stratix 10 devices automatically compress the configuration bitstream. You cannot disable compression in Intel Stratix 10 devices.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Data Width (bits)</th>
<th>Max Clock Rate</th>
<th>Max Data Rate</th>
<th>MSEL[2:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passive</td>
<td>JTAG</td>
<td>1</td>
<td>30 MHz</td>
<td>3'b111</td>
</tr>
</tbody>
</table>

Table 28. Intel Stratix 10 Configuration Data Width, Clock Rates, and Data Rates

Related Information

- Programming Support for Jam STAPL Language
- JTAG Configuration Timing in Intel Stratix 10 Devices

3.4.1. JTAG Configuration Scheme Hardware Components and File Types

The following figure illustrates JTAG programming. This is the simplest device configuration scheme. You do not have to use the File ➤ Convert Programming Files... to convert the .sof file to a .pof.
3. Intel Stratix 10 Configuration Schemes

**Figure 46. JTAG Configuration Scheme**

- Quartus Prime Compilation
- SOF
- Quartus Prime Programmer
- Intel FPGA
- JTAG Programmer

**Figure 47. Components and Design Flow for JTAG Programming**

- Quartus Software flow on PC
  - Quartus Prime Compilation
  - SOF
  - Quartus Prime Programmer
  - Quartus Prime: File → Start Compile
  - Quartus Prime: Tools → Programming
- Intel FPGA Download Cable II
- PCB
  - 10 pin JTAG header
  - Intel FPGA
  - SDM
3.4.2. JTAG Device Configuration

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

You can configure the Intel Stratix 10 device through JTAG using a download cable or a microprocessor.

3.4.2.1. JTAG Single-Device Configuration using Download Cable Connections

Refer to the Intel Stratix 10 Device Family Pin Connection Guidelines for additional information about individual pin usage and requirements.
Figure 48. Connection Setup for JTAG Single-Device Configuration using Download Cable

**Related Information**
- Intel FPGA Download Cable II User Guide
- Intel Stratix 10 Device Family Pin Connection Guidelines
3.4.2.2. JTAG Single-Device Configuration using a Microprocessor

Refer to the *Intel Stratix 10 Device Family Pin Connection Guidelines* for additional information about individual pin usage and requirements.

**Figure 49.** Connection Setup for JTAG Single-Device Configuration using a Microprocessor

**Related Information**

*Intel Stratix 10 Device Family Pin Connection Guidelines*
3.4.3. JTAG Multi-Device Configuration

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

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

Figure 50. Connection Setup for JTAG Multi Device Configuration using Download Cable

Resistor values can vary between 1 kΩ to 10 kΩ. Perform signal integrity to select the resistor value for your setup.

For JTAG configuration only:
Connect MSEL [2:0] of Intel FPGA devices to VCCIO_SDM through 4.7 kΩ external pull-up resistor.

For JTAG in conjunction with another configuration scheme:
Connect MSEL [2:0] of Intel FPGA devices based on the non-JTAG configuration scheme.

3.4.4. Debugging Guidelines for the JTAG Configuration Scheme

The JTAG configuration scheme overrides all other configuration schemes. The SDM is always ready to accept configuration over JTAG unless a security feature disables the JTAG interface. JTAG is particularly useful in recovering a device that may be in an unrecoverable state reached when trying to configure using a corrupted image.
An nSTATUS falling edge terminates any JTAG access and the device reverts to the MSEL-specified boot source. nSTATUS must be stable during JTAG configuration. nSTATUS follows nCONFIG during JTAG configuration. Consequently, nCONFIG also must be stable.

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

**Note:** For Intel Stratix 10 SX devices when you choose to configure the FPGA fabric first, the JTAG chain has no mechanism to redeliver the HPS boot information following a cold reset. Consequently, you must reconfig the device with the .sof file or avoid cold resets to continue operation.

**Debugging Suggestions**

Here are some debugging tips for JTAG:

- Verify that the JTAG pin connections are correct.
- If JTAG configuration is failing, check that the FPGA has successfully powered up and exited POR. One strategy is to check the hand shaking behavior between nCONFIG and nSTATUS by driving nCONFIG low and ensuring that nSTATUS also goes low.
- Verify that the nCONFIG pin remains high during JTAG configuration.
- Another way to determine whether the device has exited the POR state is to use the Intel Quartus Prime Programmer to detect the device. If the programmer can detect the Intel Stratix 10 device, it has exited the POR state.
- If you are using an Intel FPGA Download Cable II, reduce the cable clock speed to 6 MHz.
- If you have multiple devices in the JTAG chain, try to disconnect other devices from the JTAG chain to isolate the Intel Stratix 10 device.
- If you specify the OSC_CLK_1 as the clock source for configuration, ensure that OSC_CLK_1 is running at the frequency you specify in the Intel Quartus Prime software.
- For designs including the High Bandwidth Memory (HBM2) IP or any IP using transceivers, you must provide a free running and stable reference clock to the device before device configuration begins. All transceiver power supplies must be at the required voltage before configuration begins.
- When you use the JTAG interface for reconfiguration after an initial reconfiguration using AS or the Avalon-ST interface, the .sof must be in the file format you specified in the Intel Quartus Prime project. For example, if 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.
Related Information

Intel Stratix 10 Debugging Guide on page 164
4. Remote System Update

Remote system update implements device reconfiguration using dedicated remote system update circuitry available in all Intel Stratix 10 devices. Remote system update 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 remote system update, you use the Intel Stratix 10 Serial Flash Mailbox Client Intel FPGA IP core to write configuration bitstreams to the AS x4 flash device. Then you can use the Mailbox Client Intel Stratix 10 FPGA IP core to instruct the SDM to reboot from the updated image. You can store multiple application images and a single factory image in the configuration device. Your design manages remote updates of the application images in the configuration device.

A command to the Mailbox Client Intel Stratix 10 FPGA IP core initiates reconfiguration. The remote update system performs configuration error detection during and after the reconfiguration process. If errors in the application images prevent reconfiguration, the configuration circuitry reverts to the default factory image and provides error status information.

The following figure shows functional diagrams for typical remote system update processes. For passive configuration schemes, the host implements remote system update rather than the Intel Stratix 10 device. To learn more about remote system update for passive configuration schemes, refer to Remote Update Intel FPGA IP User Guide for remote system update implementations in earlier device families. This document explains the remote system update implementation for active configuration schemes.
Figure 51. Typical Remote System Update Process

Related Information
Remote Update Intel FPGA IP User Guide
4. Remote System Update Functional Description

4.1. Remote System Update Using AS Configuration

Remote system update using AS configuration includes the following components:

- Your external remote system update host design. The host can be custom logic, the HPS, or a Nios® II processor in the FPGA.
- One factory image.
- Flash memory for image storage.
- At least one application image.
- Designs that do not use the HPS as the remote system update host require an Intel Stratix 10 Mailbox Client FPGA IP core as shown in the figure below. The Mailbox Client sends and receives remote system update operation commands and responses. To perform RSU operations through a Intel Stratix 10 device that does not include the HPS, you must use the Intel Stratix 10 Mailbox Client FPGA IP to issue commands. Optionally, you may use Intel Stratix 10 Serial Flash Mailbox Client IP to perform flash transactions, such as rewriting the application image to the serial flash.

Figure 52. Intel Stratix 10 Remote System Update Components
4.1.2. Remote System Update Configuration Images

Intel Stratix 10 devices using remote system update require the following configuration images:

- A Factory image—contains logic with enough functionality to implement the following functions:
  - Your design-specific logic to obtain new application images
  - Your design-specific logic to request reconfiguration using a specific application image
  - Image storage in flash memory

- Application image—contains logic to implement the custom application. The application image must also contain logic to obtain new application images and store the images in the flash memory.

Depending on the storage space of your flash memory, Intel Stratix 10 remote system update supports one factory application image and up to 507 application images. The Quartus Programming File Generator only supports up to three remote system update images. However, you can add more images using the Mailbox Client IP or Serial Flash Mailbox Client IP with the device in user mode.
4.1.3. Remote System Update Configuration Sequence

Figure 53. Remote System Update Configuration Sequence

In the following figure the blue text are states shown in the Configuration Flow Diagram on page 19.
Reconfiguration includes the following steps:

1. After the device exits power-on-reset (POR), the boot ROM loads flash memory from one of the four static firmware slots at addresses 0, 256k, 512k, or 768k to initialize the SDM. The same configuration firmware is present in each of these locations. (Refer to Step 2 of Guidelines for Performing Remote System Update Functions for Non-HPS on page 122 for step-by-step details for programming the firmware into the flash.)

2. The optional direct-to-factory pin controls whether the SDM firmware loads the factory or application image. You can assign the direct-to-factory input to any unused SDM pin. The SDM loads the application image if you do not assign this pin.

3. The configuration firmware pointer block in the flash device maintains a list of pointers to the application images.

4. When loading an application image, the SDM traverses the pointer block in reverse order. The SDM loads the highest priority image. When image loading completes, the device enters user mode.

5. If loading the newest (highest priority) image is unsuccessful, the SDM tries the next application image from the list. If none of the application loads successfully, the SDM loads the factory image.

6. If loading the factory image fails, you can recover by reprogramming the Quad SPI flash with the RSU image using the JTAG interface.

### 4.1.4. RSU Recovery from Corrupted Images

When an RSU fails, the Mailbox Client Intel Stratix 10 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 Example of Tcl Script under Intel Stratix 10 Devices.
The following two 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
  2. Application Image2
  3. Application Image1
  4. Application Image0
- Application Image3, Application Image2, and Application Image1 are corrupted.
- RSU_STATUS includes the following information:
  - Current Image: Application Image0
  - Last failing image, State, Version, Error location, Error details record information for Application Image3 which is the highest priority failing image.
4. Remote System Update

4.1.5. Update of Static Firmware and Factory Image

In rare instances you may need to update the static firmware or factory Image,
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 static firmware and factory image remotely. Here are the steps to perform the update:

1. Generate the update Image `update.rpd` using Programming File Generator. This bitstream consists of both new static firmware and a factory image.

2. Program the `update.rpd` to a new empty slot starting from a new sector boundary of the flash device. Trigger reconfiguration to load the new Image.

3. The updated image performs the following operations:
   a. Erases and replaces the previous static firmware in the flash device.
   b. Reprograms the new factory Image in the flash device. If a power failure occurs during the update process the reprogramming continues after power is restored.
   c. After the update completes, the updated image removes itself from the flash and loads the application image or the factory image if an application image is not available.
4.2. Guidelines for Performing Remote System Update Functions for Non-HPS

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 a Mailbox Client Intel Stratix 10 FPGA IP.
   - You can use either custom logic, the Nios II processor, or the JTAG to Avalon Master Bridge IP as a remote system update host controller.
   - The remote system update host controller controls the remote system update function by sending commands to and receiving responses from the SDM via Mailbox Client Intel Stratix 10 FPGA IP. The Mailbox Client functions as the messenger between the remote system update host and SDM. It passes the commands to and responses from the SDM.

2. The pre-generated standard remote system update image file should include a factory image and at least one application image. The remote system update image must be programmed into the flash memory. You can use a dummy image to begin developing RSU functionality before the actual application image is complete. In user mode you can program additional application images.
Refer to Generating Remote System Update Image Files using Programming File Generator on page 133 for the step by step process to generate the standard and single remote system update image files using the programming file generator.

3. The remote system update requires you to use the AS x4 configuration scheme to configure the FPGA with the pre-generated remote system update image.

4. Once the device enters user mode with either the factory image or an application image, the remote system update host can perform the following remote system update operations:
   a. Reconfiguring the device with an application or factory image
      i. From factory image to an application image or vice versa
      ii. From an application image to another application image
   b. Erasing the application image
   c. Adding an application image
   d. Updating an application image

**Related Information**
- Intel Stratix 10 SoC Development Kit User Guide
- Mailbox Client Intel Stratix 10 FPGA IP Core User Guide

### 4.3. Commands and Responses

The remote system update host communicates with the SDM using command and response packets via the Mailbox Client Intel Stratix 10 FPGA IP.

**Block Diagram**

The following figure illustrates the role of the Mailbox Client Intel Stratix 10 in a Intel Stratix 10 design. The Mailbox Client IP enables communication with the SDM to access Quad SPI flash memory and system status including temperature and voltage readings.
Mailbox Client Role

Host Controller
- JTAG to Avalon Master
- Nios II Processor
- Custom Logic
- Avalon-MM Interface
- IRQ

FPGA Core Fabric
- Mailbox Client Stratix 10 Intel FPGA IP
  - FIFO
  - Controller
  - SDM Communication Hub
  - Mailbox Driver

Secure Device Manager
- Avalon-ST Interface

Figure 56. Command and Response Header Format

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>RESERVED</td>
</tr>
<tr>
<td>30</td>
<td>ID</td>
</tr>
<tr>
<td>28</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>LENGTH</td>
</tr>
<tr>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>COMMAND / ERROR CODE</td>
</tr>
</tbody>
</table>

Note: The LENGTH field in the command header must match the command length of corresponding command. The following table describes the fields of the header command.

Table 29. Mailbox Client Intel Stratix 10 FPGA IP Command and Response Header Description

<table>
<thead>
<tr>
<th>Header</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>[27:24]</td>
<td>The command ID. The response header returns the ID specified in the command header. Set different IDs in each command to match responses with commands.</td>
</tr>
<tr>
<td>0</td>
<td>[23]</td>
<td>Reserved.</td>
</tr>
</tbody>
</table>

continued...
### 4.3.1. Operation Commands

Table 30. **Mailbox Client Intel Stratix 10 FPGA IP Command List and Descriptions (RSU Functions for Non-HPS Variants)**

<table>
<thead>
<tr>
<th>Command</th>
<th>Code (Hex)</th>
<th>Command Length (10)</th>
<th>Response Length (10)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSU_IMAGE_UPDATE</td>
<td>5C</td>
<td>2</td>
<td>0</td>
<td>Triggers reconfiguration from data source or configuration data stored in AS x4 flash memory. This command takes a 64-bit argument to specify the reconfiguration address in flash memory. • Bit [63:32]: Reserved (write as 0). • Bit [31:0]: The start address of an application image. Returns non-zero response if the device is already processing a reconfiguration.</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 fields:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Word</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(10) This number does not include the command and response header.
<table>
<thead>
<tr>
<th>Command</th>
<th>Code (Hex)</th>
<th>Command Length (10)</th>
<th>Response Length (10)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0xF002</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>HARDWARE_ACCESS_FAILURE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0xF003</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BITSTREAM_CORRUPTION</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0xF004</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>INTERNAL_ERROR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0xF005</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DEVICE_ERROR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0xF006</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>HPS_WATCHDOG_TIMEOUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0xF007</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>INTERNAL_UNKNOWN_ERROR</td>
</tr>
</tbody>
</table>

1. Version  0 for this version.

2. Pin status  Specifies the values of the following configuration control signals.
   • Bit [31]: Current nSTATUS output value (active low).
   • Bit [30]: Detected nCONFIG input value (active low).
   • Bit [29:3]: Reserved.
   • Bit [2:0]: The MSEL value at power up.

3. Soft function status  Specifies the value of each of the soft functions, whether you assigned the function assigned to an SDM pin.
   • Bit [31:4]: Reserved.
   • Bit [3]: SEU_ERROR.
   • Bit [2]: CVP_DONE.
   • Bit [1]: INIT_DONE.
   • Bit [0]: CONF_DONE.

4. Error location  Contains the error location. Returns 0 for no error.

5. Error details  Contains the error details. Returns 0 for no error.

RSU_STATUS  5B  0  8  Reports the current remote system update status. This command returns the following responses:

<table>
<thead>
<tr>
<th>Word</th>
<th>Summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-1</td>
<td>Current image</td>
<td>Flash offset of the currently running application image.</td>
</tr>
</tbody>
</table>

---

(10) This number does not include the command and response header.
### 4. Remote System Update

**GET_DESIGNHASH**

<table>
<thead>
<tr>
<th>Command</th>
<th>Code (Hex)</th>
<th>Command Length (10)</th>
<th>Response Length (10)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET_DESIGNHASH</td>
<td>14</td>
<td>1</td>
<td>3</td>
<td>The GET_DESIGNHASH command is little-endian. This command returns a number that is the 3 least significant words of the hash value. Number format is 0x(Byte N-1)(Byte N-2)...(Byte2)(Byte1)(Byte0). In contrast, the Assembler reports a complete hash value as hex stream, with byte ordering from low to high, (Byte0)(Byte1)(Byte2)... (Byte N-2)(Byte N-1). For example, for the design hash value dbe4554730f6bb487e52eb31cc68b194 from the Assembler report (*_.asm.rpt), the Mailbox Client GET_DESIGNHASH command returns 0x31eb527e48bbf6304755e4db.</td>
</tr>
</tbody>
</table>

| QSPI_OPEN   | 32         | 0                   | 0                    | Clients use this command to request exclusive access AS x4 interface. The SDM returns the appropriate response: |

| Command Code (Hex) Command Length (10) Response Length (10) Description |
|-------------|---------------------|----------------------|-------------|
| 2-3         | First failing image | Flash offset of the first failing application image. A value of all 1s indicates no failing images. When there are no failing images, the following words do not contain meaningful data. |
| 4           | State               | Failure code of the last failing image. The error field has two parts: • Upper 16 bits: Major error code. • Lower 16 bits: Minor errors that do not contain meaningful data. The following major error codes are defined: |

<table>
<thead>
<tr>
<th>Major Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xF001</td>
<td>BITSTREAM_ERROR</td>
</tr>
<tr>
<td>0xF002</td>
<td>HARDWARE_ACCESS_FAILURE</td>
</tr>
<tr>
<td>0xF003</td>
<td>BITSTREAM_CORRUPTION</td>
</tr>
<tr>
<td>0xF004</td>
<td>INTERNAL_ERROR</td>
</tr>
<tr>
<td>0xF005</td>
<td>DEVICE_ERROR</td>
</tr>
<tr>
<td>0xF006</td>
<td>HPS_WATCHDOG_TIMEOUT</td>
</tr>
<tr>
<td>0xF007</td>
<td>INTERNAL_UNKNOWN_ERROR</td>
</tr>
</tbody>
</table>

| 5 | Version | Contains the value of each of the soft functions, whether that function is on an SDM pin. |
| 6 | Error location | Contains the error location of the last failing image. Returns 0 for no error. |
| 7 | Error details | Contains the error details of the last failing image. Returns 0 for no error. |

This number does not include the command and response header.
<table>
<thead>
<tr>
<th>Command</th>
<th>Code (Hex)</th>
<th>Command Length &lt;sup&gt;(10)&lt;/sup&gt;</th>
<th>Response Length &lt;sup&gt;(10)&lt;/sup&gt;</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QSPI_CLOSE</td>
<td>33</td>
<td>0</td>
<td>0</td>
<td>Closes exclusive access to the AS x4.</td>
</tr>
<tr>
<td>QSPI_SET_CS</td>
<td>34</td>
<td>1</td>
<td>0</td>
<td>Selects the flash memory using chip select lines.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bit [31:28]: Chip selects for flash memories 0-4 using one-hot encoding.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bit [27:0]: Reserved (write as 0).</td>
</tr>
<tr>
<td>QSPI_READ</td>
<td>3A</td>
<td>2</td>
<td>N</td>
<td>Reads the attached QSPI flash device. Defines the following 2 parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The flash address offset to start a read. The read must be word-aligned. Reads one word.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Number of words to read.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When successful returns the OK response code followed by the data read from the QSPI flash device.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When unsuccessful, returns 1 of the following responses:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Returns an error code.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>In the case of multiple reads, if some of the reads fail returns read data for the successful reads and OK for the failure.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: The maximum transfer size is 4 KB. The QSPI_READ command cannot run during configuration.</td>
</tr>
<tr>
<td>QSPI_WRITE</td>
<td>39</td>
<td>2+N</td>
<td>0</td>
<td>Writes data to the flash memory. Defines the following 3 parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The word start address in flash memory. The write must be word aligned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The size in words.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A successful write returns an OK response code.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The client may need to issue QSPI_ERASE command before issuing this command to prepare the memory for writing.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: The maximum transfer size is limited to 4 KB. The QSPI_WRITE command cannot run during configuration.</td>
</tr>
<tr>
<td>QSPI_ERASE</td>
<td>38</td>
<td>2</td>
<td>0</td>
<td>Erases a sector of the flash memory. Defines the following 2 parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The word start address in flash memory to begin the erasure. The address must be the start address of a sector in the flash memory, resulting in 64 KB alignment. Returns an error if the address is not 64 KB aligned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The number of words to erase. The erasure size must be a multiple of 4000 (hexadecimal) words.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A successful erase returns an OK response code.</td>
</tr>
</tbody>
</table>

<sup>(10)</sup> This number does not include the command and response header.

continued...
4.3.2. Error Code Responses

Table 31. Mailbox Client Intel Stratix 10 FPGA IP Error Code Responses and Description

<table>
<thead>
<tr>
<th>Value (Hex)</th>
<th>Error Code Response</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>OK</td>
<td>Indicates that the command completed successfully. Depending on the command delivered to the Mailbox Client, the response error code may not be sufficient to ensure that the operation completed successfully.</td>
</tr>
<tr>
<td>1</td>
<td>INVALID_COMMAND</td>
<td>Indicates that the command is in an incorrect format.</td>
</tr>
<tr>
<td>2</td>
<td>UNKNOWN_BR</td>
<td>Indicates that the command code is not understood.</td>
</tr>
<tr>
<td>3</td>
<td>UNKNOWN</td>
<td>Indicates that the command code is not understood by the currently loaded firmware.</td>
</tr>
<tr>
<td>100</td>
<td>NOT_CONFIGURED</td>
<td>Indicates that the device is not configured.</td>
</tr>
</tbody>
</table>

(10) This number does not include the command and response header.
<table>
<thead>
<tr>
<th>Value (Hex)</th>
<th>Error Code Response</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1FF</td>
<td>ALT_SDM_MBOX_RESP_DEVICE_BUSY</td>
<td>Indicates that the device is busy.</td>
</tr>
<tr>
<td>2FF</td>
<td>ALT_SDM_MBOX_RESP_NO_VALID_RESP_AVAILABLE</td>
<td>Indicates that there is no valid response available.</td>
</tr>
<tr>
<td>3FF</td>
<td>ALT_SDM_MBOX_RESP_ERROR</td>
<td>General Error</td>
</tr>
</tbody>
</table>
4.4. Remote System Update Flash Device Layout

The Intel Quartus Prime Programming Files Generator populates the flash memory when you generate the remote system update programming files.

The following table shows the format of the programming files. Application images follow the configuration firmware pointer. By default the first 16 bytes of the application image .rpd starting at address offset 0x1FC0 are 0. However you can use these 16 bytes to store a Version ID to identify your application image. Providing this Version ID allows you to verify the images stored in flash memory at a later time.

Table 32. Remote System Update Flash Memory Layout

The start of flash address 0, or the 0xA2 partition within a partitioned flash address 0, must be set up as shown in the following table.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Size (Byte)</th>
<th>Usage</th>
<th>Sub-Partition Name</th>
<th>Sub-Partition Flag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Reserved Address (Bit 0)</td>
<td>Read-Only (Bit 1)</td>
</tr>
<tr>
<td>0k</td>
<td>256k</td>
<td>Static Firmware Section</td>
<td>BOOT_INFO (remote system update boot image)</td>
<td>YES</td>
</tr>
<tr>
<td>256k</td>
<td>256k</td>
<td>Static Firmware Section</td>
<td>FACTORY_IMAGE</td>
<td>YES</td>
</tr>
<tr>
<td>512k</td>
<td>256k</td>
<td>Static Firmware Section</td>
<td></td>
<td></td>
</tr>
<tr>
<td>768k</td>
<td>256k</td>
<td>Static Firmware Section</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1024</td>
<td>64k</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1M+256k</td>
<td>Varies</td>
<td>Factory Image (11)</td>
<td>FACTORY_IMAGE</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Next</td>
<td>32k</td>
<td>Sub-partition table</td>
<td>SPT0</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Next + 32k</td>
<td>32k</td>
<td>Sub-partition table (Back-up copy)</td>
<td>SPT1</td>
<td>YES</td>
</tr>
<tr>
<td>Next + 32k</td>
<td>32k</td>
<td>Configuration firmware pointer block</td>
<td>CPB0</td>
<td>YES</td>
</tr>
<tr>
<td>Next + 32k</td>
<td>32k</td>
<td>Configuration firmware pointer block (Back-up copy)</td>
<td>CPB1</td>
<td>YES</td>
</tr>
</tbody>
</table>

(11) The Intel Quartus Prime software reserves an extra 256 KB of memory following this image for potential factory image updates.
4.4.1. Configuration Firmware Pointer Block (CPB)

The configuration firmware accesses the configuration firmware pointer block when performing remote system update. The Intel Quartus Prime Programming Files Generator sets up the initial configuration firmware pointer block. Each copy of the configuration firmware pointer block (CPB0/CPB1) must be exactly 4 KB.

The configuration firmware does not load an image if a pointer contains a value of all zeros or all ones.

*Note:* Application images must align to partition boundaries in the flash device. If an application image is less than a full partition, the rest of the partition cannot be used.

**Table 33. Configuration Firmware Pointer Block Format**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Size (Bytes)</th>
<th>Sub-Partition Name</th>
<th>Example Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>32</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>
| 0x20   | 8            | First (lowest priority) image pointer slot | • Bit [31:0]: Applications Image N Start address  
• Bit [63:32]: Reserved |
| 0x28   | 8            | Second (2nd lowest priority) image pointer slot | • Bit [31:0]: Applications Image 2 Start address  
• Bit [63:32]: Reserved |
| And so on | 8            | —                  | —               |
| 0xFF0 | 8            | Last (highest priority) image pointer | • Bit [31:0]: Applications Image 1 Start address  
• Bit [63:32]: Reserved |
| 0xFF8 | —            | Reserved           |                 |

(12) User-assigned sub-partition name.
4.5. Generating Remote System Update Image Files using Programming File Generator

Use the Intel Quartus Prime Programming File Generator tool to generate the Intel Stratix 10 remote system update flash programming files.

4.5.1. Generating a Standard RSU Image

Follow these steps to generate a standard RSU image:

1. On the File menu, click Programming File Generator.
2. Select Intel Stratix 10 from the Device family drop-down list.
3. Select the configuration scheme from the Configuration scheme drop-down list. The current Intel Quartus Prime only supports remote system update feature in Active Serial x4.
4. On the Output Files tab, assign the output directory and file name.
5. Select the output file type.
   - Select the following file types for AS x4 configuration mode:
     - JTAG Indirect Configuration File (.jic)/Programmer Object File (.pof)
     - Memory Map File (.map)
     - Raw Programming File (.rpd)
6. On the Input Files tab, click Add Bitstream, select the factory and application image .sof files and click Open.
7. On the Configuration Device tab, click Add Device, select your flash memory and click OK. The Programming File Generator tool automatically populates the flash partitions.
8. Select the FACTORY_IMAGE partition and click Edit.
9. In the Edit Partition dialog box, select your factory image .sof file in the Input file drop-down list and click OK.
   
   Note: You must assign Page 0 to Factory Image. Intel recommends that you let the Intel Quartus Prime software assign the Start address of the FACTORY_IMAGE automatically by retaining the default value for Address Mode which is Auto. From the Address Mode drop down list, select Block to set an End address value for the FACTORY_IMAGE. The Programming File Generator reserves and assigns the start and end flash addresses to store BOOT_INFO, SPT0, SPT1, CPOB, and CPBI.
10. Select the flash memory and click Add Partition.
11. In the **Add Partition** dialog box, select for application image .sof file from the **Input file** drop-down list, assign the page number.

12. Repeat this step for additional application images and click **OK**. You can add up to three partitions for three application images. The page 1 application image is the highest priority, and the page 3 image is the lowest priority.

13. For .jic files, click **Select** at the Flash loader, select your device family and device name, and click **OK**.

14. Click **Generate** to generate the remote system update programming files. After generating the programming file, you can proceed to program the flash memory.

   *Note:* The generated .jic file contains only the initial flash data. If a remote host updates the initial flash image and then the application performs a verify operation, the verify operation fails.

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

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

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

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

### 4.5.2. Generating a Single RSU Image

Follow these steps to generate single RSU image (.rpd) for adding or updating an application image in user mode.

1. On the **File** menu, click **Programming File Generator**.

2. Select Intel Stratix 10 from the **Device family** drop-down list.

3. Select the configuration mode from the **Configuration mode** drop-down list. The current Intel Quartus Prime only supports remote system update feature in **Active Serial x4**.

4. On the **Output Files** tab, assign the output directory and file name.

5. Select the output file type.
   
   Select the following file types for AS x4 configuration mode:
• Raw Programming File (.rpdl)

6. Click the Edit... button and assign the Start address for the image in flash memory. This Start address must match the starting address of the target partition in flash memory.

**Figure 57. Specifying Parameters for Single .rpdl Stored in Flash Memory**

7. By default, the .rpdl 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 .rpdl file in big endian format.

8. On the Input Files tab, click Add Bitstream. Change the Files of type to SRAM Object File (*.sof). Then, select application image .sof file and click Open.
9. Click **Generate** to generate the remote system update programming files. You can now program the flash memory.

### 4.5.3. Generating an Update Image for Static Firmware and Factory Image

Follow these steps to generate an update Image (.rpd) for a static firmware and factory image update.

**Note:** Starting with the Stratix 10 device family the .rpd to program flash memory include firmware pointer information for image addresses as shown in Table 32 on page 131. 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 Intel Stratix 10 from the **Device family** drop-down list.
3. Select the configuration mode from the **Configuration mode** drop-down list. The current Intel Quartus Prime only supports remote system update feature in **Active Serial x4**.
4. On the **Output Files** tab, assign the output directory and file name.
5. Select the output file type.
   - Select the following file types for AS x4 configuration mode:
     - Raw Programming File (.rpd)
6. Click the **Edit...** button and assign the **Start address** for the image in flash memory. This **Start address** must match the starting address of the target partition in flash memory.
Figure 59. Specifying Parameters for Single .rpδ Stored in Flash Memory

7. By default, the .rpδ 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 .rpδ file in big endian format.

8. On the Input Files tab, click Add Bitstream. Change the Files of type to SRAM Object File (*.sof). Then, select application image .sof file and click Open.
9. Select the `.sof` and then click **Properties**. Turn **On Remove System Firmware Update**.
10. Click **Generate** to generate the RSU programming files. You can now update the Intel Stratix 10 firmware.
4.6. 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.

4.7. Prerequisites

   To run this remote system update example, your system must meet the following hardware and software requirements:
• You should be running the Intel Quartus Prime Pro Edition software version 18.0 Update 1 or later.
• You should create and download this example to the Intel Stratix 10 SoC Development Kit.
• Your design should include the Mailbox Client Intel Stratix 10 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.

**Figure 62. Required Communication and Host Components for the Remote System Update Design Example**
4.8. Creating Initial Flash Image Containing Bitstreams for Factory Image and One Application Image

1. On the File menu, click Programming File Generator.
2. Select Intel Stratix 10 from the Device family drop-down list.
3. Select the configuration mode from the Configuration mode drop-down list. The current Intel Quartus Prime Software only supports remote system update feature in Active Serial x4.
4. On the Output Files tab, assign the output directory and file name.
5. Select the output file type.
   Select the following file types for the Active Serial (AS) x4 configuration mode:
   • JTAG Indirect Configuration File (.jic)
   • Memory Map File (.map)
   • Raw Programming File (.rpd). It is optional to generate the .rpd file.
6. On the Input Files tab, click Add Bitstream, select the factory and application image .sof files and click Open.
a. Bitstream_1 is the bitstream for factory image.
b. Bitstream_2 is the bitstream for application image.
7. On the **Configuration Device** tab, click **Add Device**, select **MT25QU02G** flash memory and click **OK**. The Programming File Generator tool automatically populates the flash partitions.

8. Select the **FACTORY_IMAGE** partition and click **Edit**.

9. On the **Edit Partition** dialog box, select **Bitstream_1** as the factory image .sof in the **Input file** drop-down list. Keep the default settings for **Page 0** and **Address Mode**. Click **OK**.

10. Select the **MT25QU02G** flash memory and click **Add Partition**.

11. In the **Add Partition** dialog box, select **Bitstream_2** for the application image .sof in the **Input file** drop-down list. Assign **Page: 1**. Keep the default settings for **Address Mode**. Click **OK**.

12. For **Flash loader** click **Select**. Select Intel Stratix 10 from **Device family** list. Select **1SX280LU3S2** for the **Device name**. Click **OK**.

13. 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
### Programming File Generator

#### Device family: Stratix 10
- **Configuration mode:** Active Serial x4

#### Output Files

**Device** | **Start** | **End** | **Input File**
--- | --- | --- | ---
MT25QU02G | 0x00000000 | 0x0010FFFF | None
**BOOT_INFO** | 0x00010000 | <auto> | Bitstream_1 (sdmtest.sof at page 0)
**FACTORY_IMAGE** | <auto> | <auto> | None
SPT0 | <auto> | <auto> | None
SPT1 | <auto> | <auto> | None
CPB0 | <auto> | <auto> | None
CPB1 | <auto> | <auto> | None
**Application Image** | <auto> | <auto> | Bitstream_2 (sdmtest.sof at page 1)

**Flash loader:** 1SX280LU3S2

---

**Remote System Update**

UG-S10CONFIG | 2019.04.10
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.

<table>
<thead>
<tr>
<th>BLOCK</th>
<th>START ADDRESS</th>
<th>END ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOT_INFO</td>
<td>0x00000000</td>
<td>0x0010FFFF</td>
</tr>
<tr>
<td>FACTORY_IMAGE</td>
<td>0x00110000</td>
<td>0x002D3FFF</td>
</tr>
<tr>
<td>SPT0</td>
<td>0x002D4000</td>
<td>0x002DBFFF</td>
</tr>
<tr>
<td>SPT1</td>
<td>0x002DC000</td>
<td>0x002E3FFF</td>
</tr>
<tr>
<td>CPB0</td>
<td>0x002E4000</td>
<td>0x002EBFFF</td>
</tr>
<tr>
<td>CPB1</td>
<td>0x002EC000</td>
<td>0x002F3FFF</td>
</tr>
<tr>
<td>Application Image</td>
<td>0x002F4000</td>
<td>0x004B7FFF</td>
</tr>
</tbody>
</table>

Configuration device: 1SX280LU3S2
Configuration mode: Active Serial x4
Quad-Serial configuration device dummy clock cycle: 15

Notes:
- Data checksum for this conversion is 0xBFFB90A5
- All the addresses in this file are byte addresses

After generating the programming file, you can program the flash memory.
4.9. Programming Flash Memory with the Initial Remote System Update Image

1. Open Programmer, click Add File. Select the generated .jic file (Initial_RSU_Image.jic) and click Open.
2. Turn on the Program/Configure for the attached .jic file.
3. To begin programming the flash memory with the initial remote system update image, click Start.
4. Configuration is complete when the progress bar reaches 100%. Power cycle the board to automatically configure the Intel Stratix 10 device with the application image using the AS x4 configuration scheme.
4. Remote System Update

Intel Stratix 10 Configuration User Guide
5. Use the **RSU_STATUS** command to determine which bitstream image the Programmer is using as shown in the following example:

   a. In the Intel Quartus Prime software, select **Tools ➤ System Debugging Tools ➤ System Console** to launch the system console.

   b. In the Tcl Console pane, type `source rsu1.tcl` to open the example of Tcl script to perform the remote system update commands. Refer to the **Related Information** for a link to `rsu1.tcl`.

   c. Type the `rsu_status` command to report the current remote system update status. You can retrieve the current running image address from the remote system update status report. The current image address must match with the start address for the application image printed in the `.map` file.
4. Remote System Update

Intel Stratix 10 Configuration User Guide

Send Feedback
4.10. Reconfiguring the Device with an Application or Factory Image

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

1. The remote system update host sends the RSU_IMAGE_UPDATE command to perform the remote system update to the new application image or factory image.
   a. For example, in the Tcl console of the System Console, type the following command to initiate a remote system update to the factory image.
      i. rsu_image_update 0x00110000
         *Note:* This command reconfigures the device with factory image. Address 0x00110000 is the start address of the factory image as shown in the .map file. The JTAG host automatically disconnects from the System Console once the device reconfiguration is successful. You must restart the System Console to re-establish the connection with the device to perform next command.
      ii. rsu_image_update 0x002F4000
         *Note:* This command reconfigures the device with the application image. Address 0x002F4000 is the start address of the application image as shown in the .map file.
   Optional: Retrieve the remote system update status by using the rsu_status command to ensure you have successfully reconfigured the device.

2. In the Tcl console of the System Console, type rsu_status to verify the current image. The following figure shows the device is being reconfigured with the factory image.
4.11. 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:

   ```bash
   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.

```
% source rsul.tcl
/channels/local/top/master_1
% program_flash new_application_image.rpd 0x03ff0000 1024
total number of words is 458752
total number of page is 448
total number of sector is 28
reading rpd is completed
start erasing flash
erasing flash is completed
start writing flash
writing flash is completed
```

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

Note: You must update both copies (CBP0 and CBP1) when editing the configuration firmware pointer block and sub-partition table. Refer to Table 33 on page 132 for more details about the configuration firmware pointer block.

Based on the example described above, the address offset `0x20` in the CPB0 and CPB1 must point to the start address of the application image. The next new image pointer slot value must be `0xFFFFFFFF` before you write the start address of the new application image to the next image pointer slot.
### Table 34. Configuration Firmware Point Block Contents

<table>
<thead>
<tr>
<th>CPB Start Address + 0x20</th>
<th>Content</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPB0 + 0x20 = 0x002E4020</td>
<td>Current application image pointer slot (highest priority)</td>
<td>0x002F4000</td>
</tr>
<tr>
<td>CPB0 + 0x28 = 0x002E4028</td>
<td>Next image pointer slot</td>
<td>0xFFFFFFFF</td>
</tr>
<tr>
<td>CPB1 + 0x20 = 0x002EC020</td>
<td>Current application image pointer slot (highest priority)</td>
<td>0x002F4000</td>
</tr>
<tr>
<td>CPB1 + 0x28 = 0x002EC028</td>
<td>Next image pointer slot</td>
<td>0xFFFFFFFF</td>
</tr>
</tbody>
</table>

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

1. Start address
2. Number of words to read
You can now proceed to write the new application image address to next image slot 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
You can now do a `QSPI_read` function to the next image pointer slot to ensure that it is written with the start address of the desired new application image.

Verifying the Update to the New Image Pointer Slot at 0x00234028

Host software can now reconfigure the Intel Stratix 10 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 static firmware traverses the image pointer slots in reverse order. The new image has the highest priority when you restart the device.
4.12. 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 slot 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 slot 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 slots for CPB0 and CPB1 for offsets 0x20 and 0x28:

<table>
<thead>
<tr>
<th>CPB Start Address + 0x20</th>
<th>Content</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPB0 + 0x20 = 0x002E4020</td>
<td>Old application image pointer slot (lower priority)</td>
<td>0x002F4000</td>
</tr>
<tr>
<td>CPB0 + 0x28 = 0x002E4028</td>
<td>Current/new application image pointer slot (highest priority)</td>
<td>0x03FF0000</td>
</tr>
<tr>
<td>CPB1 + 0x20 = 0x002EC020</td>
<td>Old application image pointer slot (lower priority)</td>
<td>0x002F4000</td>
</tr>
<tr>
<td>CPB1 + 0x28 = 0x002EC028</td>
<td>Current/New application image pointer slot (highest priority)</td>
<td>0x03FF0000</td>
</tr>
</tbody>
</table>
You can now remove the current or new application image address image pointer slot 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.

```
% qspi_read 0x002e4020 1
0x002f4000

% qspi_read 0x002e4028 1
0x03ff0000

% qspi_read 0x002ec020 1
0x002f4000

% qspi_read 0x002ec028 1
ISR is empty
0x03ff0000
```

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

```
% qspi_write_one_word 0x002e4028 0x00000000

% qspi_write_one_word 0x002ec028 0x00000000
```
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, 0x002f4000.
5. Intel Stratix 10 Configuration Features

5.1. Device Security

Note: Contact your Intel sales representative for more information about the device security support in Intel Stratix 10 devices.

The Intel Stratix 10 device provides the following flexible and robust security features to protect sensitive data and intellectual property:

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

5.2. 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 65. Intel Stratix 10 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 is fully configured.

- **CvP Update Mode**
  CvP update mode is a reconfiguration scheme that allows an FPGA device to deliver an updated bitstream to a target device after the device enters user mode. 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.

**Related Information**
Intel Stratix 10 Configuration via Protocol (CvP) Implementation User Guide

### 5.3. 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**
6. Intel Stratix 10 Debugging Guide

6.1. Configuration Debugging Checklist

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

Table 35. General Configuration Debugging Checklist

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

continued...
To avoid configuration failures, disconnect the PMBus regulator’s JTAG download cable before configuring Intel Stratix 10-V devices.

If the SDM Debug Toolkit is not operational, verify that the Intel Stratix 10 device has exited POR by checking nCONFIG, nSTATUS, CONF_DONE and INIT_DONE pins using an oscilloscope.

Is the configuration clock source chosen appropriately? You can use an internal oscillator or the OSC_CLK_1 pin.

For designs driving the OSC_CLK_1 pin is the frequency 25, 100, or 125 MHz?

For Intel Stratix 10 SX parts ensure that the HPS and EMIF IOPLL are stable and free running before configuration begins. The actual frequency should match the setting specified in Platform Designer.

Are proper slave addresses set for the PMBus voltage regulator modules using the Intel Quartus Prime Software?

For designs that use 3 V I/O, verify that the transceiver tiles are powered up before configuration begins.

**Related Information**

- Debugging Guidelines for the Avalon-ST Configuration Scheme on page 49
- Debugging Guidelines for the AS Configuration Scheme on page 98
- Debugging Guidelines for the JTAG Configuration Scheme on page 110
- Intel Stratix 10 Power Management User Guide

### 6.2. Intel Stratix 10 Configuration Architecture Overview

Intel Stratix 10 devices employ a new configuration 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 differs from previous Intel FPGA device families where state machines control configuration.

There are important differences between Intel Stratix 10 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.

### 6.3. SDM Debug Toolkit Overview

The SDM Debug Toolkit provides access to current status of the Intel Stratix 10 device. To use these commands you must have a valid design loaded that includes the module that you intend to access. The SDM Debug Toolkit includes the four tabs:

**Configuration Status**

The **Read Configuration Status** option provides the current value of MSEL, Configuration Pin Values, and Chip ID. The following information would be useful for debugging with the help of Intel: State, Error Location and Error Detail.
Voltage Sensor

- The **Read** option in the **External Channel** window reads the voltage on available channels. The **Read** option in the **Internal Power Supplies** window provides the values of internal power supplies.

Figure 66. Read External Channel and Read Internal Power Supplies
Temperature Sensor

- **Read** option reads the temperature in Celsius of the Intel Stratix 10 device, the HSSI channels, and the UIB_TOP and UIB_BOTTOM. The universal interface bus (UIB) blocks are general-purpose SiP interfaces for HBM2.

- **HPS Reset Control**: Provides the following two options to reset the HPS: **Release HPS from Reset** option and **HPS Cold Reset**.

Figure 67. HPS Reset Options
**HPS Reset Control**

Provides the following two options to reset the HPS: **Release HPS from Reset** option and **HPS Cold Reset**.

**Figure 68. HPS Reset Options**
6.3.1. Using the SDM Debug Toolkit

To use the Intel Stratix 10 SDM Debug Toolkit, bring up the System Console in the same version of the Intel Quartus Prime software that you used to configure the Intel Stratix 10 device.

Complete the following steps to become familiar with the Intel Stratix 10 SDM Debug Toolkit:

1. In a Nios II command shell, type the following command:

   ```
   % system-console
   ```

2. Under Intel Stratix 10 SDM Debug Toolkit click **Launch**.

3. Verify that the Intel FPGA Download Cable II connects to the PC and the 10-pin JTAG header connects to the Intel Stratix 10 device.

4. In the Intel Stratix 10 SDM Debug Toolkit, click **Refresh Connections**. The **Refresh Connections** is above the tabs in the SDM Debug Toolkit GUI.

5. On the **Configuration Status** tab, click **Read Configuration Status**. The following figure shows the **Configuration Status** after a successful read.
Figure 69. Read Configuration Status Command
6.4. Configuration Pin Differences from Previous Device Families

Intel Stratix 10 configuration pin behavior is different from earlier device families. Knowing about these differences and how these pins behave can help you understand and debug configuration issues.

<table>
<thead>
<tr>
<th>Configuration Pin Names (Pre-Intel Stratix 10)</th>
<th>Intel Stratix 10 Pin Names</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRST</td>
<td>Not Available</td>
<td>Use the TMS reset sequence. Hold TMS high for 5 TCK cycles.</td>
</tr>
<tr>
<td>CLKUSR</td>
<td>OSC_CLK_1</td>
<td>An external source you can supply to increase the configuration throughput to 250 MHz. Using an external clock source Transceivers, the HPS, PCIe, and the High Bandwidth Memory (HBM2) require you this external clock.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 25</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 125</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Refer to Setting Configuration Clock Source for instructions on setting the clock source and frequency in the Intel Quartus Prime Pro Edition software.</td>
</tr>
<tr>
<td>CRC_ERROR</td>
<td>Any unused SDM_IO (SEU_ERROR)</td>
<td>No dedicated location. Now called SEU_ERROR. Ignore until after CONF_DONE asserts.</td>
</tr>
<tr>
<td>CONF_DONE</td>
<td>SDM_IO5, SDM_IO16 (CONF_DONE)</td>
<td>No single dedicated pin location. No longer Open Drain. External pull-up Is not mandatory.</td>
</tr>
<tr>
<td>DCLK (PS - FPP)</td>
<td>AVST_CLK, AVSTx8_CLK</td>
<td>x8 mode has a dedicated clock input on SDM_IO14 (AVSTx8_CLK). For other Avalon-ST modes, use AVST_CLK. AVST_CLK and AVSTx8_CLK must be continuous and cannot pause during configuration.</td>
</tr>
<tr>
<td>DCLK (AS)</td>
<td>SDM_IO2 (AS_CLK)</td>
<td>When using the internal oscillator in AS mode, the AS_CLK runs in the range of 57 - 133 based on AS_CLK selection. If you provide a 25 MHz, 100 MHz or 125 MHz clock to the OSC_CLK_1 pin, the AS_CLK can run up to 133 MHz.</td>
</tr>
<tr>
<td>DEV_OE</td>
<td>Not Available</td>
<td></td>
</tr>
<tr>
<td>DEV_CLRn</td>
<td>Not Available</td>
<td></td>
</tr>
<tr>
<td>INIT_DONE</td>
<td>SDM_IO0, SDM_IO16 (INIT_DONE)</td>
<td>No longer Open Drain.</td>
</tr>
<tr>
<td>MSEL[0]</td>
<td>SDM_IO5 (MSEL[0])</td>
<td>After the SDM samples MSEL this pin functions as per the configuration mode selected. Do not connect directly to power. Use 4.7 KΩ pull-up or pull-downs, as appropriate.</td>
</tr>
<tr>
<td>MSEL[1]</td>
<td>SDM_IO7 (MSEL[1])</td>
<td>After the SDM samples MSEL this pin functions as per the configuration mode selected. Do not connect directly to power. Use 4.7 KΩ pull-up or pull-downs, as appropriate.</td>
</tr>
</tbody>
</table>

continued...
### Configuration Pin Names (Pre-Intel Stratix 10) - Intel Stratix 10 Pin Names - Notes

<table>
<thead>
<tr>
<th>Configuration Pin Names (Pre-Intel Stratix 10)</th>
<th>Intel Stratix 10 Pin Names</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSEL[2]</td>
<td>SDM_IO9 (MSEL[2])</td>
<td>After the SDM samples MSEL, this pin functions as per the configuration mode selected. Do not connect directly to power. Use 4.7 KΩ pull-up or pull-downs, as appropriate.</td>
</tr>
<tr>
<td>NSTATUS</td>
<td>nSTATUS</td>
<td>No longer Open Drain. Intel recommends a 10 KΩ pull-up to VCCIO_SDM.</td>
</tr>
<tr>
<td>NCEO</td>
<td>Not Available</td>
<td>Multi-device configuration is not supported.</td>
</tr>
<tr>
<td>NCEO</td>
<td>Not Available</td>
<td>Multi-device configuration is not supported.</td>
</tr>
<tr>
<td>DATA[31:0] (PP32/PP16)</td>
<td>AVST_DATA[31:0]</td>
<td>Avalon-ST x8 uses SDM pins for data pins.</td>
</tr>
<tr>
<td>DATA[7:0] (PP8)</td>
<td>SDM _IO pins (AVSTx8_DATA&lt;n&gt;)</td>
<td>Intel Stratix 10 supports up to 4 cascaded AS devices</td>
</tr>
<tr>
<td>nCSO[2:0]</td>
<td>SDMIO_8 (AS_nCSO3)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SDMIO_7 (AS_nCSO2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SDMIO_9 (AS_nCSO1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SDMIO_10 (AS_nCSO0)</td>
<td></td>
</tr>
<tr>
<td>nIO_PULLUP</td>
<td>Not Available</td>
<td>Use a JTAG instruction to invoke.</td>
</tr>
<tr>
<td>AS_DATA0_ASDO</td>
<td>SDM_IO4 (AS_DATA0)</td>
<td></td>
</tr>
<tr>
<td>AS_DATA[3:1]</td>
<td>SDM_IO6 (AS_DATA3)</td>
<td>Unlike earlier device families, the AS interface does not automatically tristate at power-on. When you set MSEL to JTAG, the SDM drives the AS_CLK, AS_DATA0-AS_DATA3, and AS_nCSO-AS_nCS3, MSEL pins until POR.</td>
</tr>
<tr>
<td></td>
<td>SDM_IO3 (AS_DATA2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SDM_IO1 (AS_DATA1)</td>
<td></td>
</tr>
<tr>
<td>PR_REQUEST</td>
<td>GPIO*</td>
<td>No dedicated location.</td>
</tr>
<tr>
<td>PR_READY</td>
<td>GPIO*</td>
<td>No dedicated location.</td>
</tr>
<tr>
<td>PR_ERROR</td>
<td>GPIO*</td>
<td>No dedicated location.</td>
</tr>
<tr>
<td>PR_DONE</td>
<td>GPIO*</td>
<td>No dedicated location.</td>
</tr>
<tr>
<td>CVP_CONF_DONE</td>
<td>Any unused SDM_IO CVP_CONF_DONE</td>
<td></td>
</tr>
</tbody>
</table>

### Related Information

- [Setting Configuration Clock Source](#) on page 36
6.5. 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 70. Example of an Intel Stratix 10 Configuration Bitstream Structure

The firmware section is not part of the .sof file. The Intel Quartus Prime Pro Edition Programmer adds the firmware to the .sof. The programmer adds the firmware when configuring an Intel Stratix 10 device or when it converts the .sof to another format.
6.6. Understanding SEUs

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

The Intel Quartus Prime software offers several features to detect and correct the effects of SEU, or soft errors, as well as to characterize the effects of SEU on your designs. Additionally, some Intel FPGAs contain dedicated circuitry to help detect and correct errors.

For more information about SEUs, refer to Intel Stratix 10 SEU Mitigation User Guide.

Related Information
Intel Stratix 10 SEU Mitigation User Guide

6.7. Reading the Unique 64-Bit CHIP ID

The Chip ID Intel IP core in each Intel Stratix 10 device stores a unique 64-bit chip ID. After the Chip ID Intel IP core receives a valid clock input, the chip ID is available on the chip_id[63:0] output port. You can read the chip ID using the JTAG interface. The chip ID may be useful for debugging. For more information about the chip ID refer to the Chip ID Intel FPGA IP Cores User Guide.

Related Information
Chip ID Intel FPGA IP Cores User Guide

6.8. 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 \texttt{OSC\_CLK\_1} frequency. It must match the frequency you specified in the Intel Quartus Prime Software and the clock source on your board.
- Ensure a free running reference clock is present for designs using transceivers, PCIe, or HBM2. These reference clocks must be available until the device enters user mode.
- For designs using the HPS and the external memory interface (EMIF), ensure that the EMIF clock is present.
- For designs using SmartVID (-V devices), ensure that this feature is set-up and operating correctly. Ensure that the voltage regulator supports SmartVID.

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

- To rule out issues with \texttt{OSC\_CLK\_1} select the \textbf{Internal Oscillator} option in the Intel Quartus Prime.
- Try configuring the Intel Stratix 10 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 \texttt{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.

\textbf{Related Information}

- \textbf{Debugging Guidelines for the Avalon-ST Configuration Scheme} on page 49
- \textbf{Debugging Guidelines for the AS Configuration Scheme} on page 98
- \textbf{Debugging Guidelines for the JTAG Configuration Scheme} on page 110

\textbf{6.8.1. nCONFIG}

The \texttt{nCONFIG} pin is a dedicated, input pin in the SDM. \texttt{nCONFIG} has two functions:

- Hold-off initial configuration
- Initiate FPGA reconfiguration

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

The configuration source can only change the state of the \texttt{nCONFIG} pin when it has the same value as \texttt{nSTATUS}. When the Intel Stratix 10 device is ready it drives \texttt{nSTATUS} to follow \texttt{nCONFIG}. 
The host should assert nCONFIG to clear the device. Then the host should deassert nCONFIG initiate configuration. If nCONFIG asserts during a configuration cycle, that configuration cycle stops. The SDM expects a new configuration cycle to begin.

**Debugging Suggestions**

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

### 6.8.2. 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.

Generally, the Intel Stratix 10 device changes the value of nSTATUS to follow the value of nCONFIG, except after an error. For example, after POR, nSTATUS asserts after nCONFIG asserts. When the host drives nCONFIG high, the Intel Stratix 10 device drives nSTATUS high.

In previous device families the deassertion of nSTATUS indicates the device is ready for configuration. For Intel Stratix 10 devices, when using Avalon-ST configuration scheme, after the Intel Stratix 10 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. If an error occurs during configuration, the length of the nSTATUS low pulse varies depending upon the type of failure. The pulse ranges from .5 ms to 1.5 ms.

nSTATUS assertion is asynchronous to data error detection. Intel Stratix 10 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. Consequently, you cannot wire OR an Intel Stratix 10 nSTATUS signal with the nSTATUS signal from earlier device families.

**Debugging Suggestions**

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

For Intel Stratix 10 devices, both CONF_DONE and INIT_DONE share multiplexed SDM_IO pins. Previous device families implement the CONF_DONE and INIT_DONE pins as open drains with a weak internal pull-up. Consequently, you cannot wire OR an Intel Stratix 10 CONF_DONE or INIT_DONE signal with the nSTATUS signal from previous device families. Otherwise, CONF_DONE and INIT_DONE behave as these signals behaved in earlier device families. If you assign CONF_DONE and INIT_DONE to SDM_IO16 and SDM_IO0, weak internal pull-downs pull these pins low at power-on reset. Ensure you specify these pins in the Intel Quartus Prime Software or in the Intel Quartus Prime settings file, (.qsf). CONF_DONE and INIT_DONE are low prior to and during configuration. CONF_DONE asserts when the device finishes receiving configuration data. INIT_DONE asserts when the device enters user mode.

**Note:**

The entire device does not enter user mode simultaneously. Intel recommends that you include the Intel Stratix 10 Reset Release IP on page 21 to hold your application logic in the reset state until the entire FPGA fabric is in user mode.

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

**Debugging Suggestions**

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

**Related Information**

- SDM Pin Mapping on page 24
- Setting Additional Configuration Pins on page 34

6.8.4. SDM_IO Pins

Intel Stratix 10 devices include 17 SDM_IO pins that you can configure to implement specific functions such as CONF_DONE and INIT_DONE. The chosen function must follow the GX, MX, TX, and SX Device Family Pin Connections Guidelines. The configuration bitstream controls the pin locations for the SDM_IO pins.

Internal Intel Stratix 10 circuitry pulls SDM_IO0, SDM_IO8 and SDM_IO16 weakly low through a 25 kΩ resistor. Internal Intel Stratix 10 circuitry pulls all SDM_IO pins weakly high during power-on.
Debugging Suggestions

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

*Figure 71. Configuration Pin Selection in the Intel Quartus Prime Pro Edition Software*

Related Information
SDM Debug Toolkit Overview on page 165
# 7. Intel Stratix 10 Configuration User Guide Archives

If an IP core version is not listed, the user guide for the previous IP core version applies.

<table>
<thead>
<tr>
<th>IP Core Version</th>
<th>User Guide</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.1</td>
<td>Intel Stratix 10 Configuration User Guide</td>
</tr>
<tr>
<td>18.0</td>
<td>Intel Stratix 10 Configuration User Guide</td>
</tr>
<tr>
<td>17.1</td>
<td>Intel Stratix 10 Configuration User Guide</td>
</tr>
</tbody>
</table>
# 8. Document Revision History for the Intel Stratix 10 Configuration User Guide

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2019.04.10</td>
<td>19.1</td>
<td>Updated the transceiver reference clocks.</td>
</tr>
</tbody>
</table>
| 2019.04.01       | 19.1                        | Made the following additions and enhancements:  
|                  |                             | • Added *Intel Stratix 10 Reset Release IP*, use the `nINIT_DONE` output of this IP to hold your application logic in reset until the entire FPGA fabric enters user mode.  
|                  |                             | • Added *Configuration Scheme Components and File Types* topics illustrating the software flow and programming file outputs for the AS, Avalon-ST, and JTAG programming schemes.  
|                  |                             | • Added the following topics to the *Stratix 10 Configuration Debugging Guide* chapter:  
|                  |                             | — *Debugging Checklist*  
|                  |                             | — *SDM Debug Toolkit Overview*  
|                  |                             | — *Using the SDM Debug Toolkit*  
|                  |                             | — *Reading the Unique 64-Bit CHIP ID*  
|                  |                             | — *Understanding SEUs*  
|                  |                             | • Added *Maximum Allowable External AS_DATA Pin Skew Delay Guidelines* topic.  
|                  |                             | • Added *Generating an Update Image for Static Firmware and Factory Image* topic.  
|                  |                             | • Added topics covering SDM_IO pin assignments and QSF settings for the Avalon-ST x8, x16, x32, and AS configuration schemes.  
|                  |                             | • Added note in the *Avalon-ST Configuration Timing* topic. This note covers special requirements for driving configuration data Avalon-ST x16 and x32 configurations.  
|                  |                             | • Added the following signals to the *Intel Stratix 10 Configuration Timing Diagram*: `nINIT_DONE`, `Data<n>-1:0]`, `AVST_READY`, `AVST_VALID`, and `AS_CS0`.  
|                  |                             | • Added a 10K Ω pull-up resistor to `nCONFIG` and corrected file type for flash image in the following figures:  
|                  |                             | — *Connections for AS x4 Single-Device Configuration*  
|                  |                             | — *Connections for AS Configuration with Multiple Serial Flash Devices*  
|                  |                             | — *Connections for Programming the Serial Flash Devices using the JTAG Interface*  
|                  |                             | • Added the IBIS model name to the *Intel Stratix 10 Configuration Pins I/O Standard and Drive Strength* table. Renamed this table *Configuration Pins I/O Standard, Drive Strength, and IBIS Model*.  

*Other names and brands may be claimed as the property of others.*
### Changes

- Added **Updating the SDM Firmware** topic in the *Intel Stratix 10 Configuration Overview* chapter.
- Added the following guidance in *Debugging Guidelines for the JTAG Configuration Scheme* topic: When you use the JTAG interface for reconfiguration after an initial reconfiguration using AS or the Avalon-ST interface, the `.sof` must be in the file format you specified in the *Intel Quartus Prime* project.
- Added the following signals to the list of device configuration pins that do not have fixed assignments:
  - CONF_DONE
  - INITDONE
  - HPS_COLD_nRESET
  - TEMP_ALERT
- Improved definitions of programming file output types.
- Edited *Using the PFL II IP Core* for clarity and style. Added many screenshots illustrating the step to complete a task.
- Updated the supported flash memory devices and supported SD* card types in the *Intel Stratix 10 Configuration Overview* topic.
- Updated the **SDM Pin Mapping** table to include the following:
  - Avalon-ST x16, and x32 configuration scheme
  - Pins for SmartVID
- Added definition of the `GETDESIGN_HASH` command to the *Mailbox Client Intel Stratix 10 FPGA IP Command List and Description* table.
- Renamed the topic title **Commands and Error Codes to Commands and Responses**.
- Updated the descriptions for **Length** and **Command Code/Error Code** in the *Mailbox Client Intel Stratix 10 FPGA IP Command and Response Header Description* table.
- Added PLL reference clock requirement to the *Additional Clock Requirements for Transceivers, HPS, PCIe, High Bandwidth Memory (HBM2) and SmartVID* topic.
- Updated the *Generating a Single RSU Image* topic to clarify that the `.rpd` for Intel Stratix 10 devices now includes firmware pointer information for image addresses and is not compatible with earlier generation methods.
- Removed a note in *Remote System Upgrade Configuration Images* topic, saying that the application image is optional and can be added later. The initial RSU setup requires both a factory image and an application image.
- Added the following note to the *Configuration Firmware Pointer Block (CPB)* topic:
  **Note:** Application images must align to partition boundaries in the flash device. If an application image is less than a full partition, the rest of the sector cannot be used.
- Added a new topic **RSU Recovery from Corrupted Images** that explains how the SDM recovers from attempts to load corrupted images.
- Added 71.5 MHz as a supported frequency for the `OSC_CLK_1` for AS configuration.
- Removed **Supported Flash Devices** appendix. This appendix has been replace by the following web page *Supported Flash Devices for Intel Stratix 10 Devices* which provides more information about flash devices for different purposes.
- Removed references to P30 and P33 flash memory devices. These CFI flash devices are no longer available.

---

### Document Version | Intel Quartus Prime Version | Changes
---|---|---
---|---|---
... | ... | ...

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Made the following corrections:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Corrected the following statement: Because Intel Stratix 10 devices operate at 1.8 volt and all SD MMC I/Os operate between 2.7 - 3.6 volts, an intermediate voltage level translator is necessary for SD cards. This statement is only true for SD cards.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Corrected the value of MSEL for Avalon-ST x16 configuration in Table 1 and Table 9. The correct value is 101.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Corrected PFL II IP core with Dual P30 or P33 CFI Flash Memory Devices figure. The nCONFIG signal should not have a pulldown resistor.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Removed the statement that Remote system upgrade cannot use partial reconfiguration (PR) images for the application image from the Remote System Upgrade Using AS Configuration topic. Remote system upgrade does support PR.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- In the Mailbox Client Intel Stratix 10 FPGA IP Command List and Description table, for CONFIG_STATUS, corrected the size of MSEL. MSEL is 3 bits.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Corrected the end address in step 14a of Generating a Standard RSU Image. It should be 0x00523FF.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Corrected the definition of QSPI_ERASE. The number of words to erase must be a multiple of 4000 (hexadecimal) words.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Changed Number of Commands and Number of Responses to Command Length and Response Length in the Mailbox Client Intel Stratix 10 FPGA IP Command List and Descriptions (RSU Functions for Non-HPS Variants) table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Corrected the fields of the RSU_STATUS command. The Last failing image field should be called the First failing image. This field reports the flash offset of the first failing application image.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- In the Remote System Upgrade Flash Memory Layout table, changed the amount of reserved flash memory image from 64k to 256k.</td>
</tr>
<tr>
<td>2018.11.02</td>
<td>18.1</td>
<td>Updated Figure 39: Intel Stratix 10 Modules and Interfaces to Implement RSU Using Images Stored in Flash Memory to exclude SD and MMC memory. These memory types are not supported in the current release.</td>
</tr>
<tr>
<td>2018.10.23</td>
<td>18.1</td>
<td>Added the following statement to the description of Avalon-ST Configuration Timing topic: The AVST_READY signal is only valid when the nSTATUS pin is high.</td>
</tr>
<tr>
<td>2018.10.10</td>
<td>18.1</td>
<td>Made the following changes:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Changed the number of remote system upgrade images supported from more than 500 to 507 in Remote System Upgrade Configuration Images.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Updated the last two entries in the Configuration Firmware Pointer Block Format table.</td>
</tr>
<tr>
<td>2018.10.04</td>
<td>18.1</td>
<td>Made the following changes:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Corrected statement in the Remote System Upgrade topic. A command to the Mailbox Client Intel Stratix 10 FPGA Mailbox Client IP Core initiates reconfiguration.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Corrected the Intel Stratix 10 Remote System Upgrade Components figure and Related Information link. The mailbox component is the Mailbox Client Intel Stratix 10 FPGA IP Core.</td>
</tr>
<tr>
<td>2018.09.21</td>
<td>18.1</td>
<td>Made the following changes:</td>
</tr>
</tbody>
</table>

*continued*
<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>• Added new chapter, Remote System Upgrade</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added new chapter, Intel Stratix 10 Debugging Guide</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added separate Debugging Guidelines topics in the Avalon-ST, AS, and JTAG configuration scheme sections.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Significantly expanded Stratix 10 Configuration Overview Configuration Overview chapter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added Additional Clock and SmartVID Requirements for Transceivers, HPS, PCIe, High Bandwidth Memory (HBM2) and SmartVID topic.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Expanded OSC_CLK_1 Clock Input topic to include additional usage requirements.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added AS Using Multiple Serial Flash Devices topic.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added numerous screenshots illustrating Intel Quartus Prime Pro Edition procedures.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Improved many figures illustrating configuration schemes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added the fact that you must have system administrator privileges to define a new flash device in the Defining New CFI Flash Memory Device topic.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added MT28EW to the list of PFL II flash devices supported.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Moved almost all of the material describing the PFL II flash from an appendix to the Intel Stratix 10 Configuration Schemes chapter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Edited entire document for clarity and style.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Corrected minor errors and typos.</td>
</tr>
<tr>
<td>2018.05.07</td>
<td>18.0</td>
<td>• Removed Estimating the Active Serial Configuration Time section.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated the OSC_CLK_1 supported frequency.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added selecting flash loader step to Generating Programming Files using Convert Programming Files.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added a note to TCK, TDI, TMS, and TDO stating that they are available for HPS JTAG chaining in SoC devices.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Removed instruction to drive nCONFIG low from POR in the following diagrams:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>— Connections for AS x4 Single-Device Configuration</td>
</tr>
<tr>
<td></td>
<td></td>
<td>— Connection Setup for AS Configuration with Multiple EPCQ-L Devices</td>
</tr>
<tr>
<td></td>
<td></td>
<td>— Connection Setup for Programming the EPCQ-L Devices using the JTAG Interface</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added a note in OSC_CLK_1 Clock Input stating that reference clocks to EMIF and PCIe IP cores must be stable and free running.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Removed .ekp file from Overview of Intel Quartus Prime Supported Files and Tools for Configuration and Programming figure.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated the Configuring Intel Stratix 10 Devices using AS Configuration section title to Generating and Programming AS Configuration Programming Files.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated Configuration Schemes and Features Overview in Intel Stratix 10 Devices table:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>— Added a note stating to contact sales representative for more information about support readiness.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>— Added a note stating to contact sales representative for more information about flash support other than EPCQ-L devices.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Removed NAND configuration support.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated Configuration Sequence in Intel Stratix 10 Devices figure by adding a looped flow arrow during Idle state.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>• Updated the MSEL note in <em>Intel Stratix 10 Device Configuration Pins</em> table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added a note to recommend <code>OSC_CLK_1</code> for configuration clock source in <code>OSC_CLK_1 Clock Input</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated CvP data width and maximum data rate in <em>Configuration Schemes and Features Overview in Intel Stratix 10 Devices</em> table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Removed the multiple EPCQ-L configuration device support.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>November 2017</td>
<td>2017.11.09</td>
<td>• Removed link to the <em>Configuration via Protocol (CvP) Implementation User Guide</em>.</td>
</tr>
<tr>
<td>November 2017</td>
<td>2017.11.06</td>
<td>• Updated <em>Option Bits Sector Format</em> table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated a step in <em>Setting Additional Configuration Pins</em>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added <em>Converting .sof to .pof File and Programming CPLDs and Flash Memory Devices</em>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated the .pof version value in <em>Storing Option Bits</em>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added information about restoring start and end address for option bits in <em>Restoring Option Bit Start and End Address</em>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added note about pull-down resistor is recommended for <code>CONF_DONE</code> and <code>INIT_DONE</code> pins in <em>Additional Configuration Pin Functions</em>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added new subsection <em>Multiple EPCQ-L Devices Support</em>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added <em>Configuration Pins I/O Standard and Drive Strength</em> table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated information about maximum additional data words when using 2-stage register synchronizer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated the equation for minimum AS configuration time estimation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added <em>RBF Configuration File Format</em> section explaining the format of the <code>.rbf</code> file.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated <em>Configuration Sequence</em> to state that a firmware which is part of the configuration data if loaded in the device initially.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated description for <em>Number of flash devices used</em> parameter in the <em>PFL II Flash Interface Setting Parameters</em> table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated <em>Configuration via Protocol overview</em> and added link to the <em>Configuration via Protocol (CvP) Implementation User Guide</em>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated <em>Partial Reconfiguration overview</em> and added link to the <em>Creating a Partial Reconfiguration Design chapter of the Handbook Volume 1: Design and Compilation</em>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated <em>Design Security Overview descriptions</em>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added note for Partial Reconfiguration feature and link to Partial Reconfiguration Solutions IP User Guide in <em>Intel Stratix 10 Configuration Overview</em>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Removed SDM pin notes in <em>Intel Stratix 10 Configuration Overview</em>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated internal oscillator's <code>AS_CLK</code> frequency in <em>Supported configuration clock source and AS_CLK Frequencies in Intel Stratix 10 Devices</em> table.</td>
</tr>
</tbody>
</table>

continued...

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>May 2017</td>
<td>2017.05.22</td>
<td>• Updated Connection Setup for Programming the EPCQ-L Device using the AS Interface figure.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated guideline to program the EPCQ-L device in Programming EPCQ-L Devices using the Active Serial Interface.</td>
</tr>
<tr>
<td>April 2017</td>
<td>2017.04.10</td>
<td>• Updated note for AS Fast Mode in MSEL Settings for Each Configuration Scheme of Devices table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added note to Configuration via Protocol recommending user to use AS x4 fast mode for CvP application.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated instances of Spansion to Cypress.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated note and description in Configuration Overview.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Removed AS x1 support.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added Connection Setup for SD/MMC Single-Device Configuration figure.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated Connections for AS x4 Single-Device Configuration, Connection Setup for AS Configuration with Multiple EPCQ-L Devices, Connection Setup for Programming the EPCQ-L Devices using the JTAG Interface, Connection Setup for NAND Flash Single-Device Configuration, and Connection Setup for SD/MMC Single-Device Configuration to include note about nCONFIG test point.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added note in Avalon-ST Configuration stating that AVST_CLK should be continuous.</td>
</tr>
<tr>
<td>February 2017</td>
<td>2017.02.13</td>
<td>• Updated Configuring Stratix 10 Devices using AS Configuration section and subsections to include .jic for AS configuration scheme.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added Programming .jic files into EPCQ-L Device.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated the SDM description.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated SDM block diagram by adding Mailbox block and note for Avalon-ST x8 configuration scheme.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated Configuration Sequence Diagram.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated configuration sequence descriptions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated Avalon-ST Bus Timing Waveform figure.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added note to Avalon-ST in Stratix 10 Configuration Overview table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated ASx4 max data rate in Stratix 10 Configuration Overview table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Removed Configurable Node subsection.</td>
</tr>
<tr>
<td>December 2016</td>
<td>2016.12.09</td>
<td>• Updated max data rate for ASx1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated the Configuration Sequence in Stratix 10 Devices figure.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated configuration sequence description.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added JTAG configuration sequence description.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added Parallel Flash Loader II IP core.</td>
</tr>
<tr>
<td>October 2016</td>
<td>2016.10.31</td>
<td>Initial release</td>
</tr>
</tbody>
</table>