Intel Stratix 10 Configuration User Guide

Updated for Intel® Quartus® Prime Design Suite: **17.1**

UG-S10CONFIG | 2017.11.09
Latest document on the web: PDF | HTML

Contents

1 Intel® Stratix® 10 Configuration Overview

Table 1. Configuration Schemes and Features Overview in Intel® Stratix® 10 Devices

Scheme	Data Width (bits)	Max Clock Rate (MHz)(1)	Max Data Rate(1)	Device Security	Partial Reconfiguration(2)	Remote System Update
Avalon-ST	32	125	4 Gbps	Yes	Yes	No
	16	125	2 Gbps	Yes	Yes	No
	8	125	1 Gbps	Yes	Yes	No
Active Serial (AS)(3)	4	133(4)	532 Mbps	Yes	Yes	Yes
SD/MMC	8	50	400 Mbps	Yes	Yes	Yes
NAND	8	50	400 Mbps	Yes	Yes	Yes
JTAG	1	30	30 Mbps	Yes	Yes	No
Configuration via Protocol (CvP)	x1, x2, x4, x8 lanes	250	5 Gbps(5)	Yes	Yes(6)	No

Note: The compression feature is enabled by default for all configuration scheme and cannot be disabled.

Related Links

- Intel FPGA Parallel Flash Loader II IP Core on page 37
- Intel Stratix 10 GX and SX Device Family Pin Connection Guidelines
- Creating a Partial Reconfiguration Design chapter of the Handbook Volume 1: Design and Compilation

(1) The max clock rate and max data rate are preliminary.

(2) You can perform partial reconfiguration after the device is fully configured. Refer to the *Creating a Partial Reconfiguration Design* chapter of the *Intel® Quartus® Prime Pro Edition Handbook Volume 1: Design and Compilation* for more information.

(3) Intel Stratix® 10 devices support configuration from EPCQ-L devices only.

(4) The maximum clock rate when using external configuration clock source is 133MHz. The maximum clock rate reduces if you use the internal oscillator as the configuration clock source, during SmartVID operation, or when the device is in user mode.

(5) The PCIe protocol overhead also limits the maximum rate.

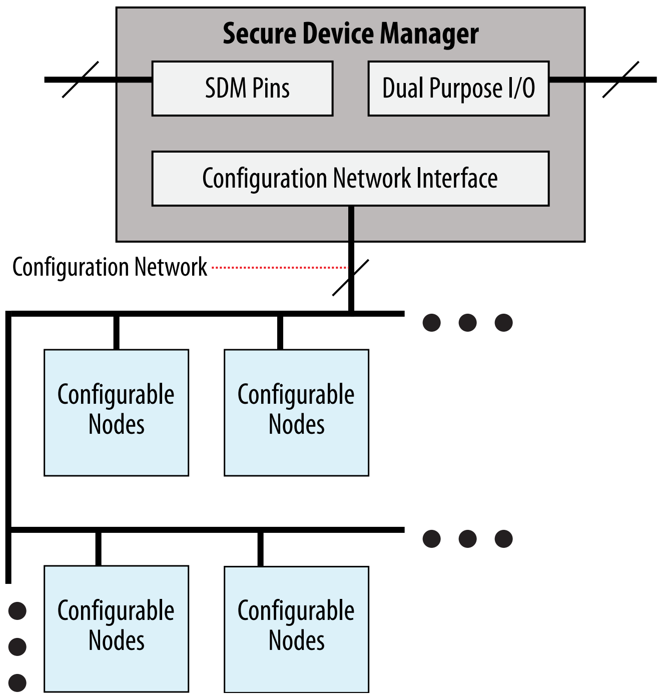
(6) Partial reconfiguration over PCIe requires additional soft logic.

1.1 Intel Stratix 10 Configuration Architecture

The Intel Stratix 10 device configuration system consists of the following components:

- Secure device manager (SDM)
- Configuration network
- Configurable nodes

Figure 1. Intel Stratix 10 Configuration Architecture Block Diagram



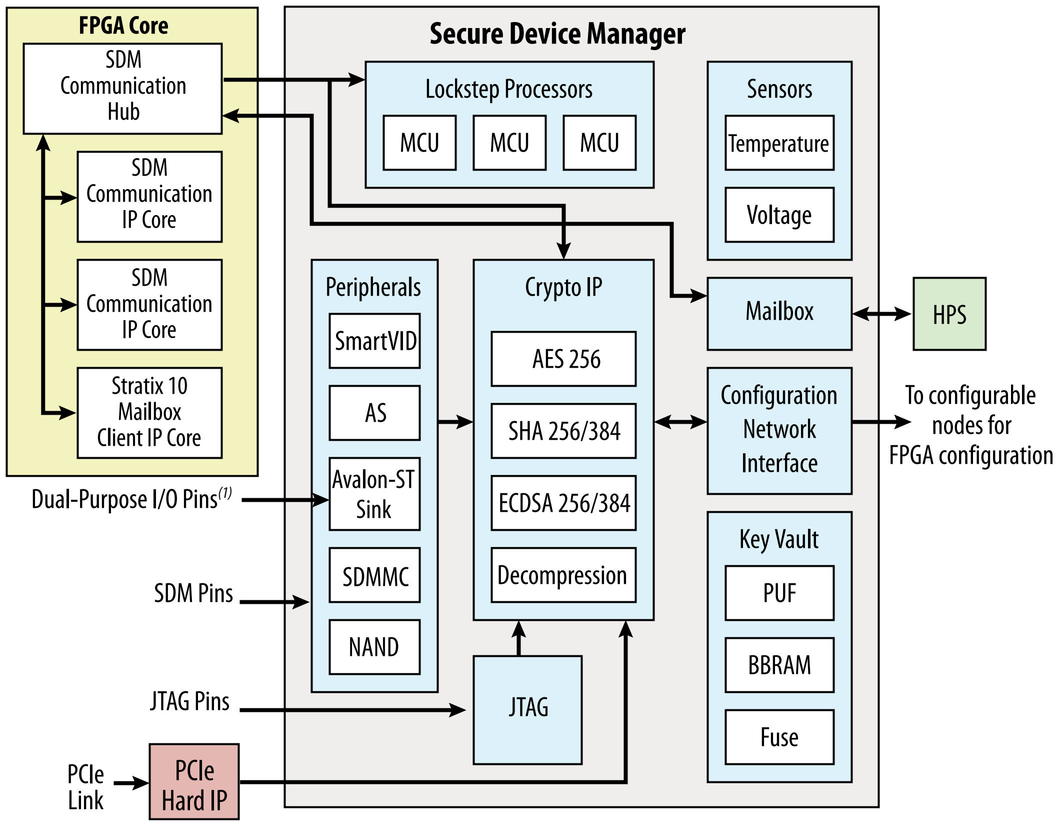
1.1.1 Secure Device Manager

SDM is a triple-redundant processor-based block that manages the following Intel Stratix 10 device processes:

- FPGA configuration
- Hard processor system (HPS) secure boot process (applicable to Intel Stratix 10 SoC devices only)

The SDM performs authentication, decryption, and decompression on the configuration data. Subsequently, the SDM sends the data over to the configurable nodes through the configuration network.

Figure 2. SDM Block Diagram



Note:
(1) Dedicated SDM pins are used for Avalon-ST x8 while the general purpose I/O pins are used for Avalon-ST x16 and x32 configuration scheme. Refer device pinout for more information.

The SDM is the point of entry to the FPGA for device configuration and to the HPS for booting using one of the following sources:

- Avalon-ST data source
- EPCQ-L configuration device via active serial interface
- NAND flash memory
- SD and MMC flash cards
- PCI Express interface
- JTAG interface

The external sources are connected to the Intel Stratix 10 device through either JTAG, SDM, or dual-purpose I/O pins. The SDM has SD/MMC, NAND flash, CFI flash, and EPCQ-L controllers to interface with external flash memories for configuration. Internal sources that are connected to the SDM include HPS and FPGA core.

Related Links

- Intel Stratix 10 Device Pinouts
- Intel Stratix 10 GX and SX Device Family Pin Connection Guidelines

2 Intel Stratix 10 Configuration Details

2.1 Configuration Sequence

Figure 3. Configuration Sequence in Intel Stratix 10 Devices

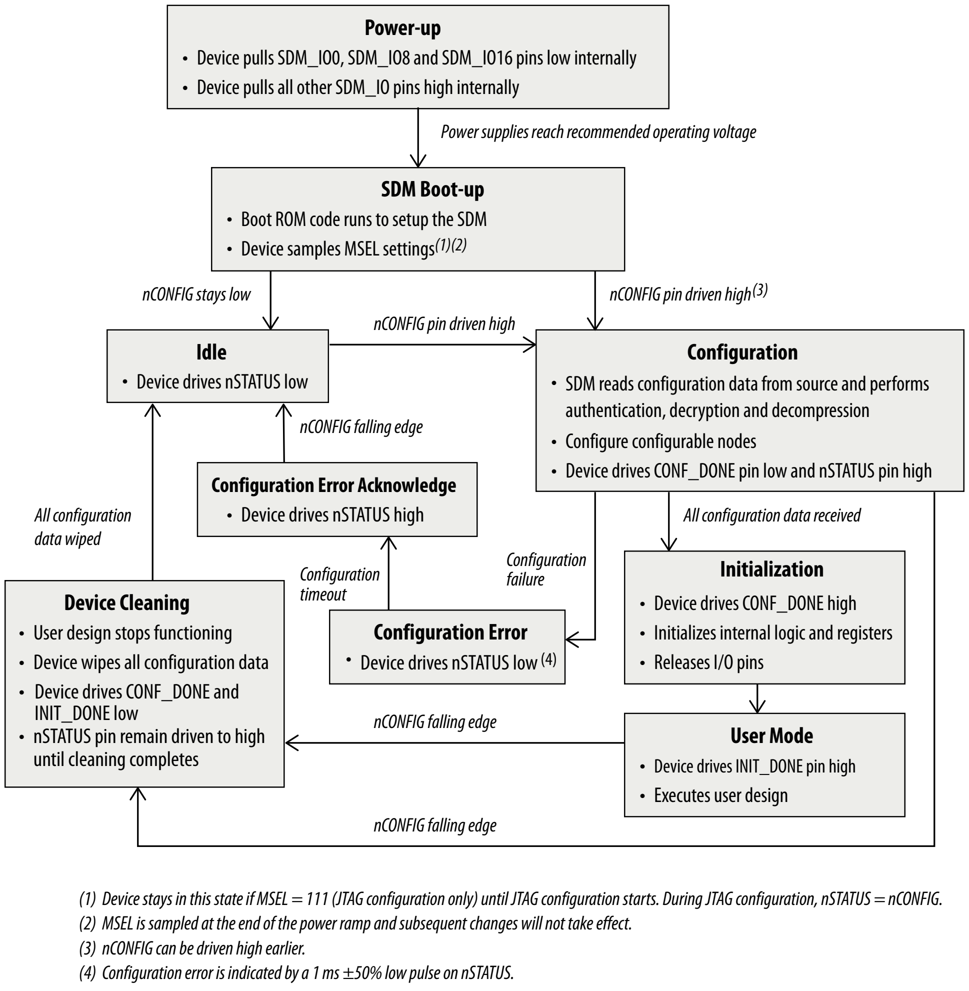
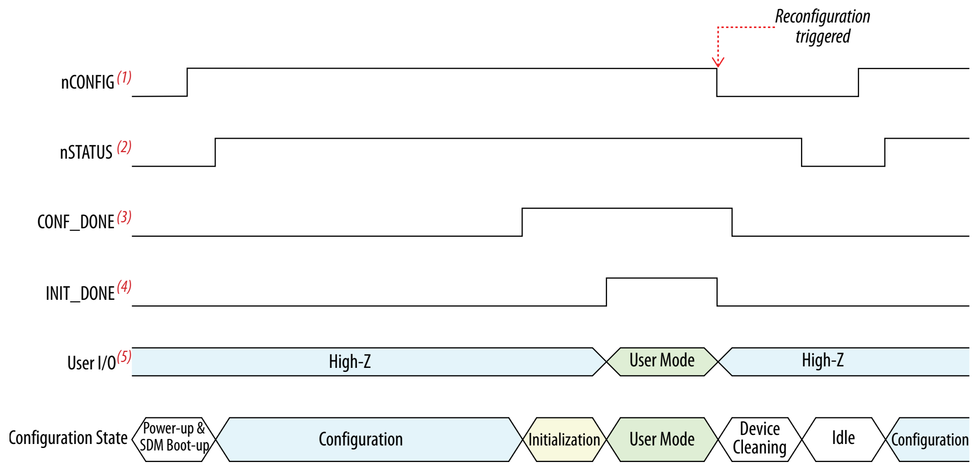

Power-up
• Device pulls SDM_IO0, SDM_IO8 and SDM_IO16 pins low internally
• Device pulls all other SDM_IO pins high internally
Power supplies reach recommended operating voltage
SDM Boot-up
• Boot ROM code runs to setup the SDM
• Device samples MSEL settings(1)(2)
nCONFIG stays low
nCONFIG pin driven high(3)
nCONFIG pin driven high
Idle
• Device drives nSTATUS low
Configuration
• SDM reads configuration data from source and performs authentication, decryption and decompression
• Configure configurable nodes
• Device drives CONF_DONE pin low and nSTATUS pin high
nCONFIG falling edge
Configuration Error Acknowledge
• Device drives nSTATUS high
All configuration data wiped
All configuration data received
Configuration timeout
Configuration failure
Device Cleaning
• User design stops functioning
• Device wipes all configuration data
• Device drives CONF_DONE and INIT_DONE low
• nSTATUS pin remain driven to high until cleaning completes
Configuration Error
• Device drives nSTATUS low (4)
Initialization
• Device drives CONF_DONE high
• Initializes internal logic and registers
• Releases I/O pins
nCONFIG falling edge
User Mode
• Device drives INIT_DONE pin high
• Executes user design
nCONFIG falling edge
(1) Device stays in this state if MSEL = 111 (JTAG configuration only) until JTAG configuration starts. During JTAG configuration, nSTATUS = nCONFIG.
(2) MSEL is sampled at the end of the power ramp and subsequent changes will not take effect.
(3) nCONFIG can be driven high earlier.
(4) Configuration error is indicated by a 1 ms ±50% low pulse on nSTATUS.

Figure 4. General Configuration Timing Diagram



(1) nCONFIG must only be driven from low to high when nSTATUS is low and from high to low when nSTATUS is high.
(2) Configuration error is indicated by a 1 ms ±50% low pulse on nSTATUS.
(3) When the CONF_DONE is implemented using the recommended SDM pin, it will be tri-stated and pulled-down internally during the power-up and SDM boot-up state.
(4) When the INIT_DONE is implemented using the recommended SDM pin, it will be tri-stated and pulled-down internally during the power-up and SDM boot-up state.
(5) All user I/O will be tri-stated when device is not in user mode.

Non-JTAG Configuration Scheme

You can identify the configuration states during device configuration by observing the behavior of the configuration pins. Based on your configuration scheme selection, the device can either receive configuration data from an external source, or read the data from external memory devices.

During power-up until after the device exit power-on-reset (POR), the device samples the MSEL pin settings to select the configuration scheme. The device goes into the SDM Boot-up state after power-up. In this state, the device runs the boot ROM code to setup the SDM system.

The device stays in the Idle state when the `nCONFIG` signal is low. A rising edge of the `nCONFIG` signal starts the configuration based on the desired scheme selected through the MSEL setting. You can use the `nCONFIG` signal to delay a device from configuring. You must only change the `nCONFIG` signal value if it is in the same state as the nSTATUS signal. Failure to do this can cause the device to miss sampling an edge on the `nCONFIG` signal. To synchronize with the configuration system, drive the `nCONFIG` signal low and wait until the nSTATUS signal goes low. Then, drive the `nCONFIG` signal high. The device starts a new configuration when the nSTATUS signal is high.

During the configuration state, the behavior of the device depends on the configuration scheme you selected. For a passive configuration scheme like Avalon-ST, the device accepts and processes the configuration data. However, for active configuration schemes like AS, the device initiates the configuration and reads data from flash memory or source device and use the data for configuration. During this state, a firmware that is part of the configuration data is loaded into the Intel Stratix 10 device initially. The SDM continues to process the rest of the configuration data subsequently. The device drives the `CONF_DONE` signal high to indicate it has received all configuration data and enters the initialization state where it performs the last configuration steps. The device drives the INIT_DONE signal high indicating the device has entered user mode. Your design starts functioning in the user mode.

The device drives the nSTATUS signal low for 1ms ± 50% if a configuration error occurs. The data source or external host may trigger a new configuration by using the nCONFIG signal.

A falling edge on the nCONFIG signal makes the device leave user mode, it wipes the user design and goes to the idle state. The device then drives the nSTATUS signal low when it goes into the idle state after the device cleaning is done and is ready to accept a new configuration.

To program a flash device that is directly connected to the device, hold the nCONFIG signal low, then use the Intel Quartus Prime Programmer to program the device. Alternatively, if you pull up all MSEL pins at power-on, the device tri-states the flash configuration pins. You can then program the flash by connecting it directly to a programmer.

JTAG Configuration Scheme

You can perform JTAG configuration anytime. The device cancels previous configuration and accepts the reconfiguration data from the JTAG interface. The nCONFIG signal must be held in a stable state during JTAG configuration as a falling edge on the nCONFIG signal cancels the JTAG configuration and makes the device configure from the MSEL selected boot source.

From any state, you can perform reconfiguration by driving the nCONFIG signal from high to low. The device reconfigures based on the configuration scheme selected by the MSEL setting when you power up the device. Driving the nCONFIG signal from high to low in any state puts the device in device cleaning state where the previous configuration data are wiped. The device then go into the idle state once the device is ready to accept new configuration.

Related Links

- Intel Stratix 10 Device Pinouts
- SDM Pin Mapping on page 10
- AS Configuration Timing on page 24
- Avalon-ST Configuration Timing on page 21
- Configuration Specifications in Intel Stratix 10 Device Datasheet
- Avalon Streaming Interface Specification

2.2 Intel Stratix 10 Configuration Pins

Intel Stratix 10 uses SDM pins for device configuration. The SDM pins perform various functions according to the configuration scheme selected.

2.2.1 SDM Pin Mapping

Table 2. SDM Pin Mapping

Note: You can use SDM pins for configuration and other functions; for example, power management. Refer to the *Intel Stratix 10 Device Pinouts* and *Intel Stratix 10 Pin Connection Guidelines* for more details on other functions.

SDM Pins	MSEL Function	Configuration Source Function				Other Functions
		Avalon-ST x8	AS	SD/MMC	NAND Flash	
SDM_IO0	—	—	—	—	—	INIT_DONE
SDM_IO1	—	AVSTx8_DATA2	AS_DATA1	SDMMC_CFG_DATA1	NAND_RE_N	—
SDM_IO2	—	AVSTx8_DATA0	AS_CLK	SDMMC_CFG_DATA0	NAND_ADQ0	—
SDM_IO3	—	AVSTx8_DATA3	AS_DATA2	SDMMC_CFG_DATA2	NAND_ADQ2	—
SDM_IO4	—	AVSTx8_DATA1	AS_DATA0	SDMMC_CFG_CMD	NAND_ADQ1	—
SDM_IO5	MSEL0	—	AS_nCSO0	SDMMC_CFG_CCLK	NAND_WE_N	CONF_DONE(7), INIT_DONE(8)
SDM_IO6	—	AVSTx8_DATA4	AS_DATA3	SDMMC_CFG_DATA3	NAND_ADQ3	—
SDM_IO7	MSEL1	—	AS_nCSO2	—	NAND_ALE	—
SDM_IO8	—	AVST_READY(9)	AS_nCSO3	SDMMC_CFG_DATA4	NAND_RB	—
SDM_IO9	MSEL2	—	AS_nCSO1	—	NAND_CLE	—
SDM_IO10	—	AVSTx8_DATA7	—	SDMMC_CFG_DATA7	NAND_ADQ5	—
SDM_IO11	—	AVSTx8_VALID	—	—	NAND_ADQ6	—
SDM_IO12	—	—	—	—	NAND_WP_N	—
SDM_IO13	—	AVSTx8_DATA5	—	SDMMC_CFG_DATA5	NAND_CE_N	—
SDM_IO14	—	AVSTx8_CLK	—	—	NAND_ADQ7	—
SDM_IO15	—	AVSTx8_DATA6	—	SDMMC_CFG_DATA6	NAND_ADQ4	—
SDM_IO16	—	—	—	—	—	CONF_DONE, INIT_DONE

Related Links

- Intel Stratix 10 Device Pinouts
- Intel Stratix 10 GX and SX Device Family Pin Connection Guidelines

(7) You can set CONF_DONE to SDM_IO5 when using Avalon-ST x8 and x32 schemes only.

(8) You can set INIT_DONE to SDM_IO5 when using Avalon-ST x8 and x32 schemes only.

(9) AVST_READY is applicable in Avalon-ST x8, x16 and x32 configuration schemes.

2.2.2 MSEL Settings

MSEL pins are used to set the configuration scheme for Intel Stratix 10 devices. You must pull-up to V_{CCIO_SDM} or pull-down to GND these pins through a 4.7-kΩ resistor depending on your configuration scheme. The device samples the MSEL after the device powers up and reaches the recommended operating voltage. If you switch the MSEL setting on the fly, you must power-down then power-up the device to the recommended operating voltage once again for the device to sample the new MSEL setting. During the SDM boot up stage, the SDM pins used for MSEL setting are sampled to determine the configuration scheme.

Externally pull the SDM pins with MSEL function high or low through a 4.7-kΩ resistor to select the desired configuration scheme according to the following below. You must also select the configuration scheme in the **Configuration** page of the **Device and Pin Options** dialog box in the Intel Quartus Prime software. The SDM pins usage are set accordingly in the programming file based on your selection.

Table 3. MSEL Settings for Each Configuration Scheme of Intel Stratix 10 Devices

Configuration Scheme	MSEL[2:0]
Avalon-ST (x32)	000
Avalon-ST (x16)	101
Avalon-ST (x8)	110
AS (Fast mode – for CvP)(10)	001
AS (Normal mode)	011
NAND x8	010
SD/MMC x4/x8	100
JTAG only(11)	111

Note: Refer to the *Intel Stratix 10 Device Pin Connection Guidelines* and *Intel Stratix 10 Device Datasheet* for more information.

Related Links

- Intel Stratix 10 GX and SX Device Family Pin Connection Guidelines
- POR Specifications in Intel Stratix 10 Device Datasheet

(10) If you use AS Fast mode and are not concerned about 100ms PCIe linkup, you must still ramp the VCCIO_SDM supply within 18ms. This ramp-up requirement is ensure that the QSPI device is within its operating voltage range when the Intel Stratix 10 device begins to access it.

(11) JTAG configuration works with MSEL settings for other configuration schemes, unless disabled for security

2.2.3 Device Configuration Pins

Table 4. Intel Stratix 10 Device Configuration Pins

- The configuration pins listed are based on the configuration schemes. Some of the schemes share the same physical pins on the device. The configuration functions of the physical pins are determined based on the configuration scheme selected by MSEL and the options selected in the Intel Quartus Prime software.
- There are no dedicated PR_REQUEST, PR_ERROR, PR_DONE, CvP_CONFDONE and SEU_ERROR pins. You can use the unused SDM IO pins for CvP_CONFDONE and SEU_ERROR pins while you can only use general purpose I/O for PR_REQUEST, PR_ERROR and PR_DONE pins by setting them in the Intel Quartus Prime software and connecting them to the Intel Stratix 10 Partial Reconfiguration AVST Controller IP core.

Configuration Function	Configuration Scheme	Input/Output	User Mode	Powered by
TCK	JTAG	Input	—	V_{CCIO_SDM}
TDI	JTAG	Input	—	V_{CCIO_SDM}
TMS	JTAG	Input	—	V_{CCIO_SDM}
TDO	JTAG	Output	—	V_{CCIO_SDM}
nSTATUS	All schemes	Output	—	V_{CCIO_SDM}/pull-up
nCONFIG	All schemes	Input	—	V_{CCIO_SDM}/pull-up
MSEL[2:0](12)	All schemes	Input	—	V_{CCIO_SDM}/pull-up/pull-down
CONF_DONE(13)	All schemes	Output	—	V_{CCIO_SDM}
INIT_DONE(14)	All schemes	Output	—	V_{CCIO_SDM}
OSC_CLK_1	All schemes	Input	—	V_{CCIO_SDM}
AS_nCSO[3:0]	AS	Output	—	V_{CCIO_SDM}
AS_DATA[3:0]	AS	Bidirectional	—	V_{CCIO_SDM}
AS_CLK	AS	Output	—	V_{CCIO_SDM}
AVST_READY	Avalon-ST x8/x16/32	Output	—	V_{CCIO_SDM}
AVSTx8_DATA[7:0]	Avalon-ST x8	Input	—	V_{CCIO_SDM}
AVSTx8_VALID	Avalon-ST x8	Input	—	V_{CCIO_SDM}
AVSTx8_CLK	Avalon-ST x8	Input	—	V_{CCIO_SDM}
AVST_DATA[31:0](15)	Avalon-ST x16/x32	Input	I/O	V_{CCIO}
AVST_VALID(15)	Avalon-ST x16/x32	Input	I/O	V_{CCIO}

continued...

(12) MSEL pins are sampled on device power-up when power-supplies reached recommended operating voltage. During configuration, the pins will have other functions based on the configuration scheme selected.

(13) You must enable the CONF_DONE pin function in the Intel Quartus Prime Software. This pin is required if you are using PFL II to configure the device for Avalon-ST configuration scheme.

(14) You must enable the INIT_DONE pin function in the Intel Quartus Prime Software. This pin is optional for all configuration scheme.

(15) Dual purpose configuration pins. You can use these pins as user I/O during user mode.

Configuration Function	Configuration Scheme	Input/Output	User Mode	Powered by
AVST_CLK[15]	Avalon-ST x16/x32	Input	I/O	V_{CCIO}
NAND_RE_N	NAND	Output	—	V_{CCIO_SDM}
NAND_WE_N	NAND	Output	—	V_{CCIO_SDM}
NAND_CE_N	NAND	Output	—	V_{CCIO_SDM}
NAND_RB	NAND	Input	—	V_{CCIO_SDM}
NAND_ALE	NAND	Output	—	V_{CCIO_SDM}
NAND_CLE	NAND	Output	—	V_{CCIO_SDM}
NAND_ADQ[7:0]	NAND	Bidirectional	—	V_{CCIO_SDM}
SDMMC_CFG_CMD	SD/MMC	Output	—	V_{CCIO_SDM}
SDMMC_CFG_DATA[7:0]	SD/MMC	Bidirectional	—	V_{CCIO_SDM}
SDMMC_CFG_CCLK	SD/MMC	Output	—	V_{CCIO_SDM}

2.2.3.1 Configuration Pins I/O Standard and Drive Strength

Table 5. Intel Stratix 10 Configuration Pins I/O Standard and Drive Strength

Configuration Pin	Type	I/O Standard	Drive Strength (mA)
TDO	Output	1.8-V LVCMOS	8
TMS	Input	Schmitt Trigger Input	—
TCK	Input	Schmitt Trigger Input	—
TDI	Input	Schmitt Trigger Input	—
nSTATUS	Output	1.8-V LVCMOS	8
OSC_CLK_1	Input	Schmitt Trigger Input	—
nCONFIG	Input	Schmitt Trigger Input	—
SDM_IO[0:16]	Input/Output	Schmitt Trigger Input or 1.8-V LVCMOS	8
All other configuration pins	Input/Output	Schmitt Trigger Input or 1.8-V LVCMOS	8

2.2.4 Additional Configuration Pin Functions

In addition to the configuration pins used in the configuration scheme, there are other configuration functions which you can assign to SDM pins. These configuration pins are implemented using unused SDM pins and can be set in the Intel Quartus Prime software.

Table 6. Additional Configuration Pins

- SDM pins are also used for SmartVID power management feature. Refer to the *Intel Stratix 10 Power Management User Guide* for more information about the pin assignments.
- Intel recommends that you include an external weak pull-down resistors for CONF_DONE and INIT_DONE pins.

Pin Function	Possible Settings	Recommended Settings	Functional Description
CONF_DONE	• **SDM_IO5**(16) • **SDM_IO16**	**SDM_IO16**	Allows you to monitor if device configuration is completed. During power-up, the SDM boot-up, and configuration stages, the pin is pulled low. Upon successful configuration, the pin is driven high by the Intel Stratix 10 device.
INIT_DONE	• **SDM_IO0** • **SDM_IO16** • **SDM_IO5**(17)	**SDM_IO0**	Allows you to monitor if device initialization is completed. During power-up, the SDM boot-up, configuration, and initialization stages, the pin is pulled low. Upon successful initialization, the pin is driven high by the Intel Stratix 10 device.

Related Links

Intel Stratix 10 Power Management User Guide User Guide

2.2.5 Setting Additional Configuration Pins

You must enable and assign the SDM pins for CONF_DONE and INIT_DONE functions in the Intel Quartus Prime software.

To set the additional configuration pins, perform the following steps:

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. Refer to Table 6 on page 14 for more information.
4. Click **OK** to confirm and close the **Configuration Pin** dialog box.

2.2.6 Enabling Dual-Purpose Pins

The AVST_CLK, AVST_DATA[15..0], AVST_DATA[31..16] and AVST_VALID are dual-purpose pins. You can set the dual purpose pins to function either as a regular I/O pin or an input tri-state, when the device enters into user mode. The V_{CCIO} of this I/O bank must be powered at 1.8V and assigned to 1.8V I/O standard if these pins are used as regular I/O pin.

To set the dual-purpose pins, perform the following steps:

1. On the **Assignments** menu, click **Device**.
2. In the **Device and Pin Options**, 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**.

(16) You can set CONF_DONE to SDM_IO5 when using Avalon-ST x8 and x32 schemes only.

(17) You can set INIT_DONE to SDM_IO5 when using Avalon-ST x8 and x32 schemes only.

2.3 Configuration Clocks

2.3.1 OSC_CLK_1 Clock Input

Intel Stratix 10 devices contain internal oscillator as the clock source for configuration. When your design uses internal oscillator, the configuration process runs between 170MHz and 230MHz. Optionally, you can feed an external clock source to the OSC_CLK_1 pin to increase the configuration process throughput. OSC_CLK_1 is used as the reference clock source for the PLL in SDM to generate 250MHz clock for configuration process.

Note:

- You must use external clock source for CvP implementation to meet the PCIe 100ms power-up time requirement. When using this clock, ensure that the supplied clock is stable.
- Unstable OSC_CLK_1 clock source may lead to potential functional failure and power cycle is needed to reconfigure the Intel Stratix 10 device.

If you are using the OSC_CLK_1, use the following clock source speed:

- 25-MHz
- 100-MHz
- 125-MHz

Set the frequency of the clock feeding the OSC_CLK_1 pin in the Intel Quartus Prime software before compiling your design. The clock is internally multiplied within the Intel Stratix 10 device to generate a 250-MHz clock for the configuration process.

2.3.2 Setting Configuration Clock Source

You must specify the configuration clock source by selecting either the internal oscillator or OSC_CLK_1 with the specific supported frequency.

To select the configuration clock source, perform the following steps:

1. On the **Assignments** menu, click **Device**.
2. In the **Device and Pin Options** select the **General** category.
3. Select the desired configuration clock source from the **Configuration clock source** drop down menu.
4. Click **OK** to confirm and close the **Device and Pin Options**.

2.4 Configuration and Programming Files

The Intel Stratix 10 configuration and external flash programming involves multiple file types and tools.

Figure 5. Overview of Intel Quartus Prime Supported Files and Tools for Configuration and Programming

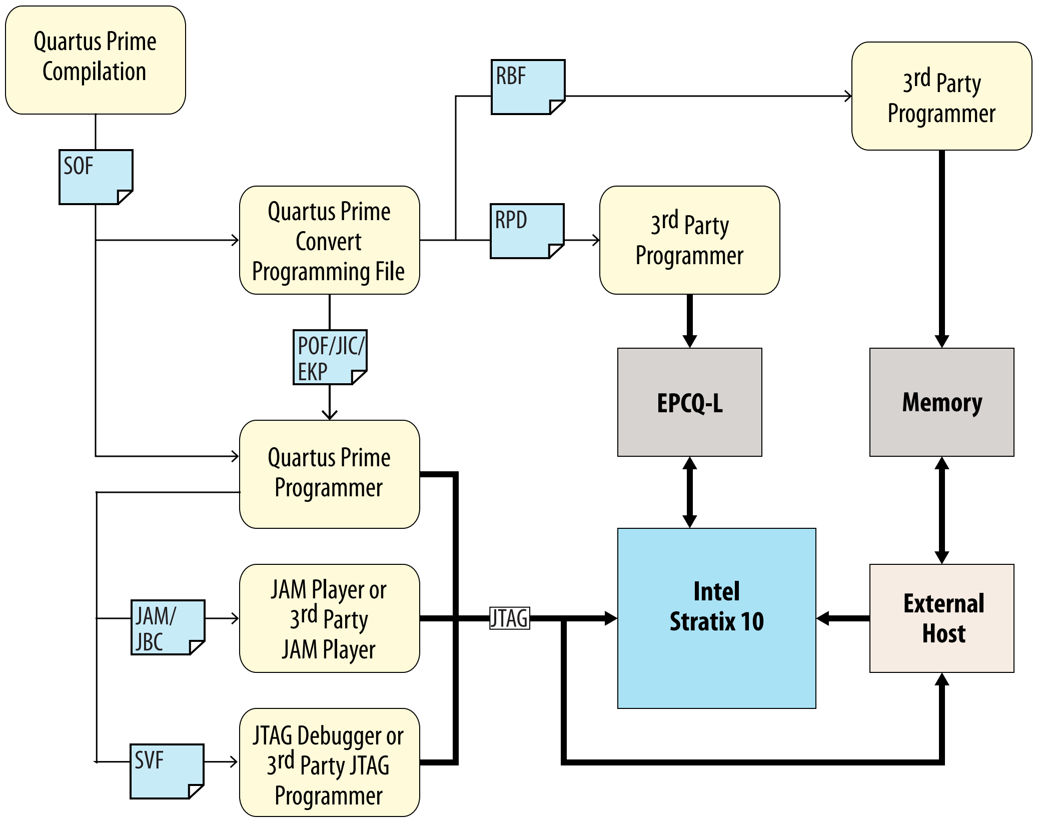

Quartus Prime Compilation
SOF
Quartus Prime Convert Programming File
RBF
RPD
3rd Party Programmer
3rd Party Programmer
POF/JIC/EKP
EPCQ-L
Memory
Quartus Prime Programmer
JAM/JBC
JAM Player or 3rd Party JAM Player
JTAG
Intel Stratix 10
External Host
SVF
JTAG Debugger or 3rd Party JTAG Programmer

Table 7. Supported Programming and Configuration File Format

File Format	Description
SRAM Object File (`.sof`/SOF)	Configuration file for JTAG configuration
Raw Binary File (`.rbf`/RBF)	Configuration file for use with a third party data source, CvP or HPS data source
Programming Object File (`.pof`/POF)	EPCQ-L and external flash programming file for AS and Avalon-ST configuration using Intel Quartus Prime Programmer
JTAG Indirect Configuration File (`.jic`/JIC)	EPCQ-L programming file for AS configuration using Intel Quartus Prime Programmer
Raw Programming Data File (`.rpd`/RPD)	EPCQ-L programming file for AS configuration using 3rd-party programmer
JAM Standard Test and Programming Language Format (`.jam`/JAM)	Configuration file for third-party JTAG host
JAM Byte Code (`.jbc`/JBC)	
Serial Vector Format (`.svf`/SVF)	

3 Intel Stratix 10 Configuration Schemes

3.1 Avalon-ST Configuration

The Avalon-ST configuration scheme uses an external host, such as a microprocessor, MAX® II, MAX V, or Intel MAX 10 device. 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 core 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.

During power-up and until the power-supply is stable, or during reconfiguration, the host drives the nCONFIG signal low. When the host is ready to configure, the Intel Stratix 10 device responds by asserting the nSTATUS signal low. When the host senses nSTATUS is asserted low, it can drive nCONFIG high. The Intel Stratix 10 device then drives nSTATUS high, and asserts AVST_READY high when it is ready to receive data.

The AVST_CLK must be continuously running throughout the configuration until the CONF_DONE goes high. 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 high when the device is ready to accept data. 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. The host :

- Must synchronize the AVST_READY signal to the AVSTx8_CLK signal or AVST_CLK signal using a 2-stage register synchronizer.
- Must de-assert the AVST_VALID signal within 6 valid cycles after the de-assertion of AVST_READY pin.
- Can send up to 6 additional data words after the de-assertion of the AVST_READY signal including the delay incur by the 2-stage register synchronizer.

To monitor for configuration error, ensure the host monitors the nSTATUS signal every 500µs.

You must properly constraint the AVST_CLK and AVST_DATA signal at 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 *Intel Stratix 10 Device Datasheet* for information about the timing specifications.

Note: The AVSTx8_CLK or AVST_CLK signals must be running continuously throughout configuration. This condition is required for the Intel Stratix 10 device to assert the AVST_READY signal.

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 completed.

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 into 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 additionally use dual-purpose I/O pins that can be used as general-purpose IO pins after configuration.

Related Links

- Intel FPGA Parallel Flash Loader II IP Core on page 37
- Intel Stratix 10 Device Pinouts
- SDM Pin Mapping on page 10
- Avalon-ST Configuration Timing in Intel Stratix 10 Device Datasheet
- Avalon Streaming Interface Specification

3.1.1 Avalon-ST Single-Device Configuration

Figure 6. Connection Setup for Avalon-ST x8 Single-Device Configuration

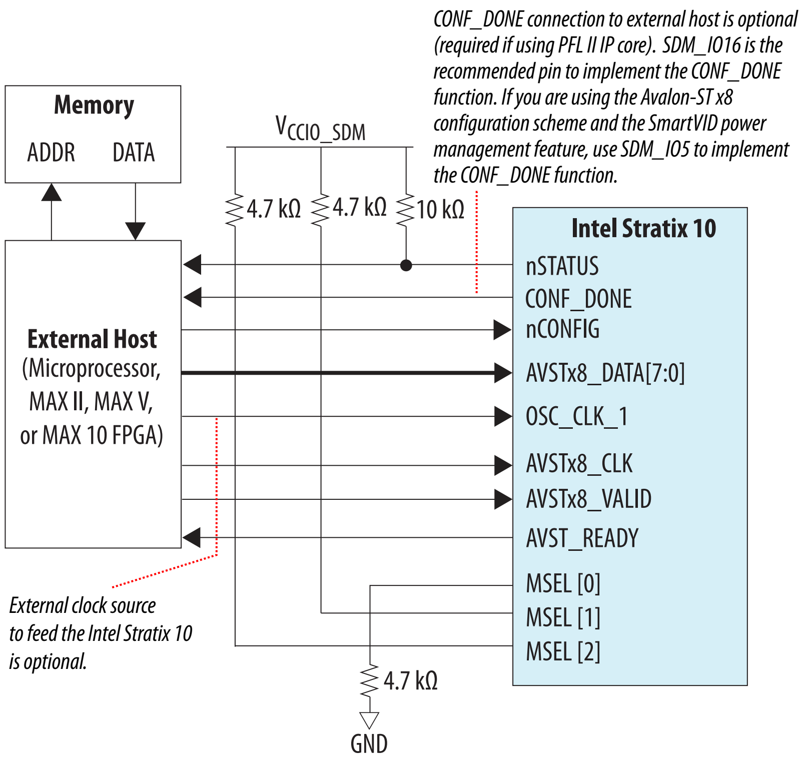

CONF_DONE connection to external host is optional (required if using PFL II IP core). SDM_IO16 is the recommended pin to implement the CONF_DONE function. If you are using the Avalon-ST x8 configuration scheme and the SmartVID power management feature, use SDM_IO5 to implement the CONF_DONE function.
Memory
ADDR
DATA
V_{CCIO_SDM}
4.7 kΩ
4.7 kΩ
10 kΩ
Intel Stratix 10
nSTATUS
CONF_DONE
nCONFIG
AVSTx8_DATA[7:0]
OSC_CLK_1
AVSTx8_CLK
AVSTx8_VALID
AVST_READY
MSEL [0]
MSEL [1]
MSEL [2]
External Host
(Microprocessor, MAX II, MAX V, or MAX 10 FPGA)
External clock source to feed the Intel Stratix 10 is optional.
4.7 kΩ
GND

Figure 7. Connection Setup for Avalon-ST x16 Single-Device Configuration

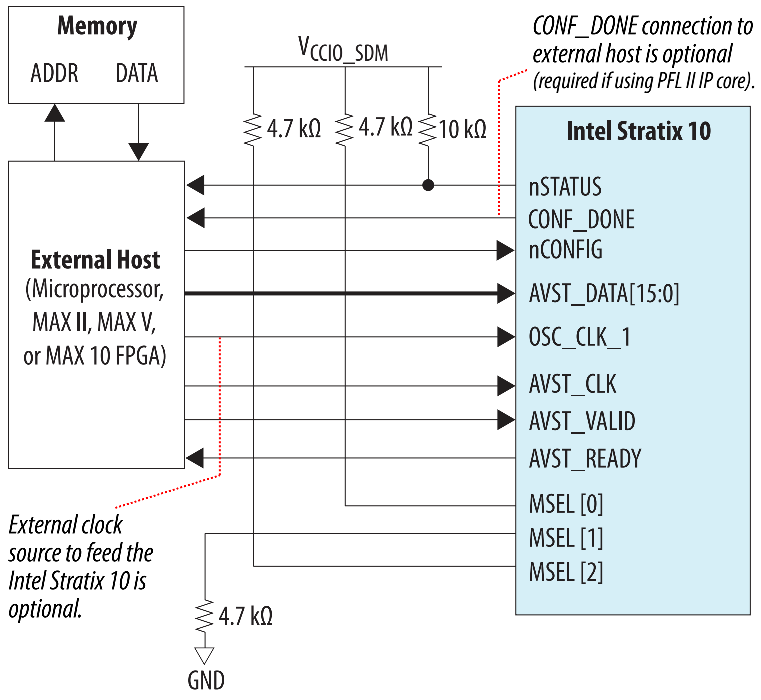
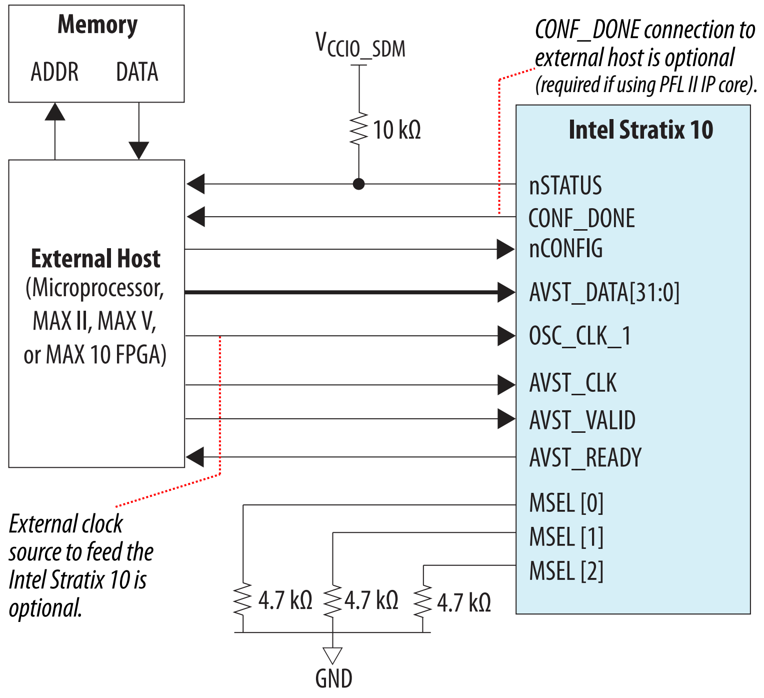

Memory
ADDR
DATA
VCCIO_SDM
4.7 kΩ
4.7 kΩ
10 kΩ
CONF_DONE connection to external host is optional (required if using PFL II IP core).
Intel Stratix 10
nSTATUS
CONF_DONE
nCONFIG
AVST_DATA[15:0]
OSC_CLK_1
AVST_CLK
AVST_VALID
AVST_READY
MSEL [0]
MSEL [1]
MSEL [2]
External Host
(Microprocessor, MAX II, MAX V, or MAX 10 FPGA)
External clock source to feed the Intel Stratix 10 is optional.
4.7 kΩ
GND

Figure 8. Connection Setup for Avalon-ST x32 Single-Device Configuration


Memory
ADDR
DATA
VCCIO_SDM
10 kΩ
CONF_DONE connection to external host is optional (required if using PFL II IP core).
Intel Stratix 10
nSTATUS
CONF_DONE
nCONFIG
AVST_DATA[31:0]
OSC_CLK_1
AVST_CLK
AVST_VALID
AVST_READY
MSEL [0]
MSEL [1]
MSEL [2]
External Host
(Microprocessor, MAX II, MAX V, or MAX 10 FPGA)
External clock source to feed the Intel Stratix 10 is optional.
4.7 kΩ
4.7 kΩ
4.7 kΩ
GND

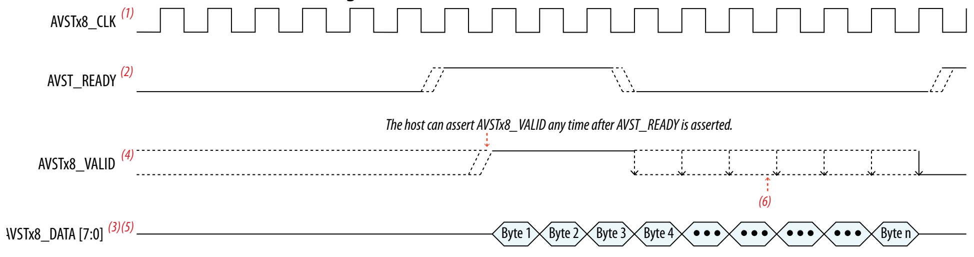
Related Links

Intel FPGA Parallel Flash Loader II IP Core on page 37

3.1.2 Avalon-ST Configuration Timing

Figure 9. Avalon-ST Bus Timing Waveform

Figure describes the Avalon-ST bus timing waveform in detail.



1. For Avalon-ST x16 and x32, this signal is AVST_CLK. These clocks must be running throughout the configuration (until CONF_DONE goes high).
2. AVST_READY is an asynchronous signal and valid only when nSTATUS is high.
3. AVSTx8_DATA signals can be in any state when device configuration is not in progress.
4. For Avalon-ST x16 and x32, this signal is AVST_VALID.
5. For Avalon-ST x16 and x32, this signal is AVST_DATA[15:0] and AVST_DATA[31:0] respectively.
6. Host may send up to 6 more data including the delay incurred by the 2-stage register synchronizer after AVST_READY is de-asserted.

Related Links

- Configuration Sequence on page 7
- Configuration Specifications in Intel Stratix 10 Device Datasheet
- Avalon-ST Configuration Timing in Intel Stratix 10 Device Datasheet
- Avalon Streaming Interface Specification

3.1.3 RBF Configuration File Format

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

The data in `.rbf` file are in little-endian format. For example, 95h 48h 29h 62h are the first 4 bytes of data sequence in `.rbf` file, you must send the data in the following manner to the AVST_DATA[]interfaces:

- `AVST_DATA [7:0]` = 95h
- `AVST_DATA [15:8]` = 48h
- `AVST_DATA [23:16]` = 29h
- `AVST_DATA [31:24]` = 62h

3.2 AS Configuration

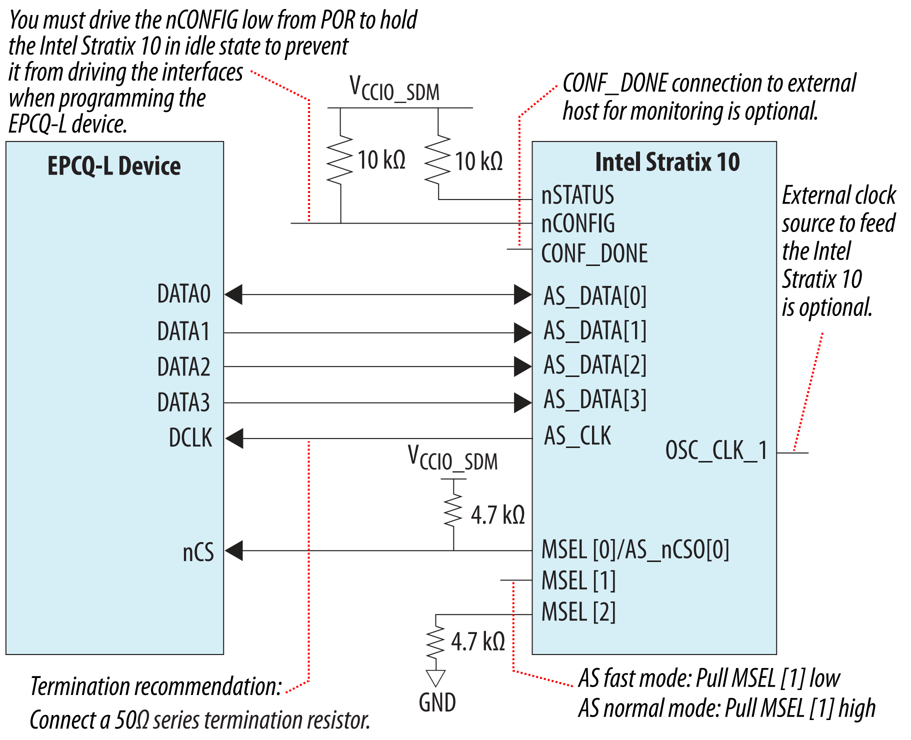
In the AS configuration scheme, the SDM block in the Intel Stratix 10 device controls the configuration process and interface. The EPCQ-L configuration devices stores the configuration data. The AS configuration scheme supports AS x4 (4-bit data width) mode only.

Note: If an HPS is present, you can use it to access the flash after initial configuration.

3.2.1 AS Configuration Setup

3.2.1.1 AS Single-Device Configuration

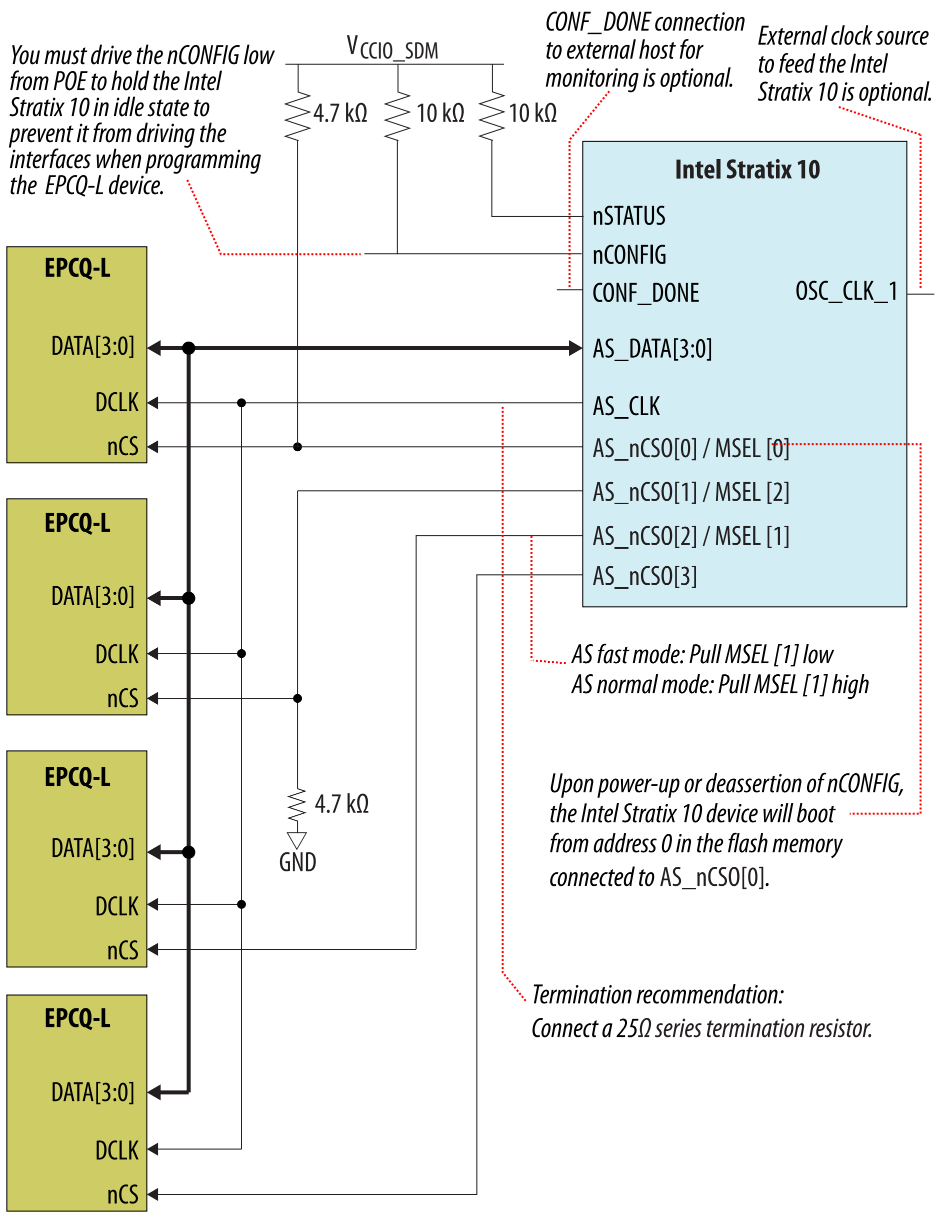
Figure 10. Connections for AS x4 Single-Device Configuration


You must drive the nCONFIG low from POR to hold the Intel Stratix 10 in idle state to prevent it from driving the interfaces when programming the EPCQ-L device.
VCCIO_SDM
10 kΩ
10 kΩ
CONF_DONE connection to external host for monitoring is optional.
EPCQ-L Device
Intel Stratix 10
nSTATUS
nCONFIG
CONF_DONE
External clock source to feed the Intel Stratix 10 is optional.
DATA0
DATA1
DATA2
DATA3
DCLK
AS_DATA[0]
AS_DATA[1]
AS_DATA[2]
AS_DATA[3]
AS_CLK
OSC_CLK_1
VCCIO_SDM
4.7 kΩ
nCS
MSEL [0]/AS_nCSO[0]
MSEL [1]
MSEL [2]
4.7 kΩ
GND
Termination recommendation: Connect a 50Ω series termination resistor.
AS fast mode: Pull MSEL [1] low
AS normal mode: Pull MSEL [1] high

3.2.1.2 AS Configuration with Multiple EPCQ-L Devices

Intel Stratix 10 devices support up to four EPCQ-L devices for configuration and remote system upgrade. You can program the EPCQ-L devices using the Intel Quartus Prime software. Each EPCQ-L device gets a dedicated AS_nCS0 pin, but shares other pins, as shown in the following figure.

Figure 11. Connection Setup for AS Configuration with Multiple EPCQ-L Devices


You must drive the nCONFIG low from POE to hold the Intel Stratix 10 in idle state to prevent it from driving the interfaces when programming the EPCQ-L device.
VCCIO_SDM
4.7 kΩ
10 kΩ
10 kΩ
CONF_DONE connection to external host for monitoring is optional.
External clock source to feed the Intel Stratix 10 is optional.
Intel Stratix 10
nSTATUS
nCONFIG
CONF_DONE
OSC_CLK_1
AS_DATA[3:0]
AS_CLK
AS_nCSO[0] / MSEL [0]
AS_nCSO[1] / MSEL [2]
AS_nCSO[2] / MSEL [1]
AS_nCSO[3]
EPCQ-L
DATA[3:0]
DCLK
nCS
EPCQ-L
DATA[3:0]
DCLK
nCS
EPCQ-L
DATA[3:0]
DCLK
nCS
EPCQ-L
DATA[3:0]
DCLK
nCS
4.7 kΩ
GND
AS fast mode: Pull MSEL [1] low
AS normal mode: Pull MSEL [1] high
Upon power-up or deassertion of nCONFIG, the Intel Stratix 10 device will boot from address 0 in the flash memory connected to AS_nCSO[0].
Termination recommendation:
Connect a 25Ω series termination resistor.

Note: For more information the Intel Stratix 10 pin connections, refer to the *Intel Stratix 10 Device Pin Connection Guideline*.

Advantages of connecting up to four EPCQ-L devices are:

- Ability to store multiple design files for remote system upgrade.
- Increase storage beyond the largest single EPCQ-L device available.
- Remaining space in the EPCQ-L can be used as additional storage for user application.

3.2.2 AS Configuration Timing

Figure 12. AS Configuration Serial Output Timing Diagram

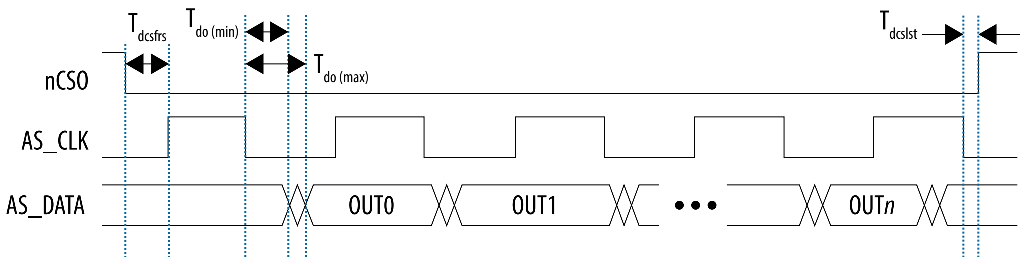
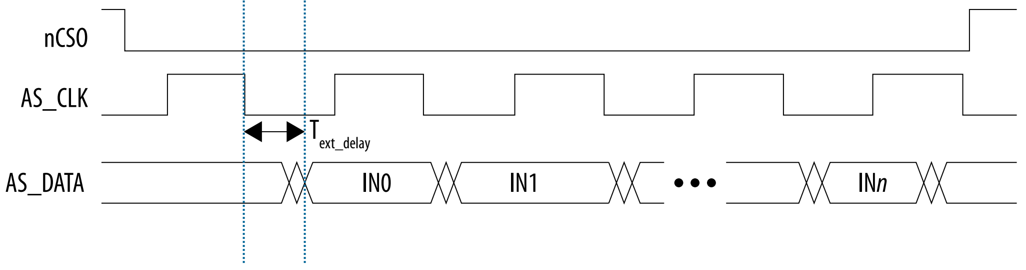


Figure 13. AS Configuration Serial Input Timing Diagram



Note: For more information about the timing parameters, refer to the *Intel Stratix 10 Device Datasheet*.

Related Links

- Configuration Sequence on page 7
- Configuration Specifications in Intel Stratix 10 Device Datasheet

3.2.2.1 AS_CLK

The Intel Stratix 10 device drives AS_CLK to the EPCQ-L device. The AS_CLK is generated from the internal oscillator or from the external clock that feeds the OSC_CLK_1 pin. 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 133MHz. Set the maximum frequency you want the AS_CLK pin to run at in the Intel Quartus Prime software. The pin runs at or below your selected frequency.

Table 8. Supported configuration clock source and AS_CLK Frequencies in Intel Stratix 10 Devices

Configuration Clock Source	AS_CLK Frequency (MHz)
Internal oscillator	• 115 • 77 • 58
OSC_CLK_1	• 133.3 • 125 • 108 • 100 • 80 • 50

Intel Stratix 10 devices use the internal oscillator to load the first section of the bitstream (approximately 200Kbytes). The device loads the remaining bitstream with a faster clock, if an external clock feeds the OSC_CLK_1 pin and is enabled in the Intel Quartus Prime software.

3.2.2.2 Estimating the Active Serial Configuration Time

The AS configuration time is determined by the time it takes to transfer the configuration data from an EPCQ-L device to the Intel Stratix 10 device.

Use the following equations to estimate the configuration time:

.rpd Size x (0.9 x minimum AS_CLK period / 4 bits per AS_CLK cycle) = estimated minimum configuration time.

3.2.3 EPCQ-L Configuration Devices

3.2.3.1 Controlling EPCQ-L Devices

The Intel Stratix 10 device's AS_nCS0 pin connects to the chip select (nCS) pin of the EPCQ-L device. During configuration, the Intel Stratix 10 device enables the EPCQ-L device by driving the AS_nCS0 output pin low. Intel Stratix 10 devices use the AS_CLK and AS_DATA0 pins to send operation commands and read address signals to the EPCQ-L device. The EPCQ-L device provides data on its serial data output (DATA[]) pins, which connect to the AS_DATA[] input of the Intel Stratix 10 devices.

3.2.3.2 Programming EPCQ-L Devices

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

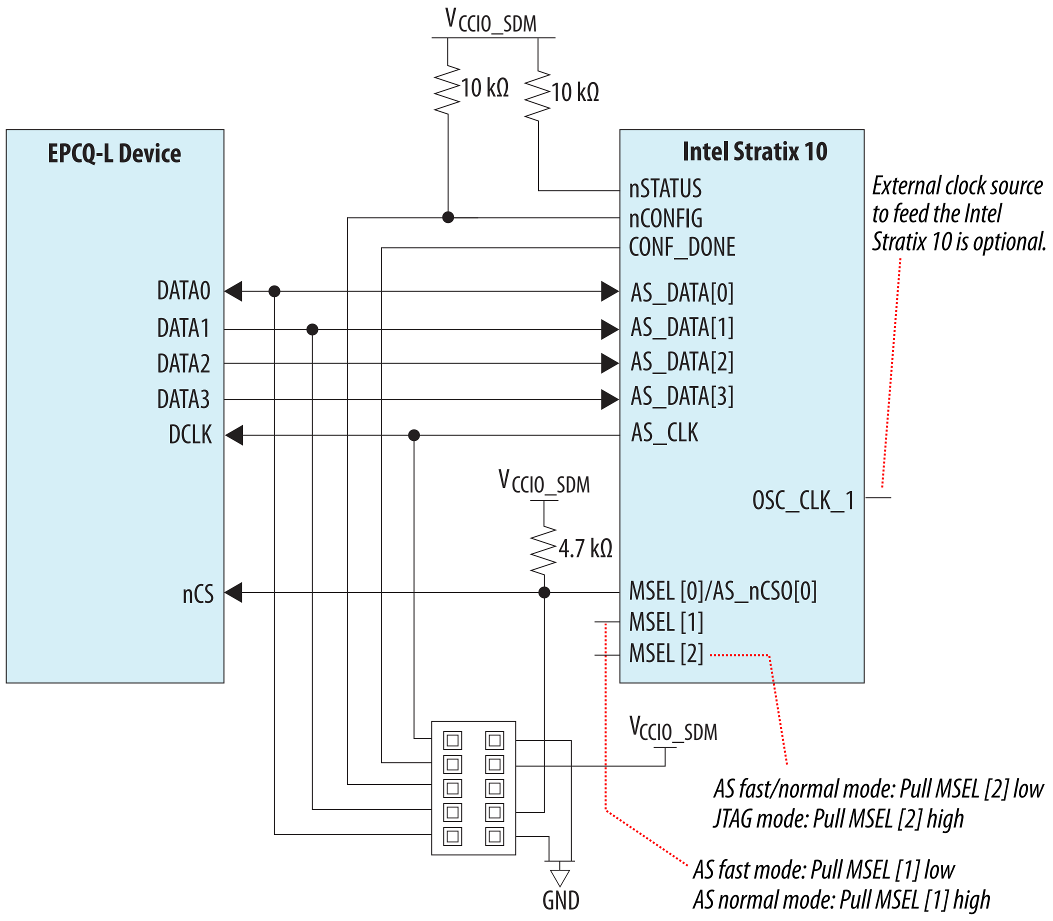
Table 9. EPCQ-L Device In-System Programming Options

You can program the EPCQ-L devices using in-system programming (ISP) through an AS programming interface or a JTAG interface.

ISP Method	Description
AS programming interface	The Intel Quartus Prime software or any supported third-party software programs the configuration data directly into the EPCQ-L device.
JTAG interface	The Intel Quartus Prime programmer interfaces with the SDM of the Intel Stratix 10 device through JTAG interface and programs the EPCQ-L device.

3.2.3.2.1 Programming EPCQ-L Devices using the Active Serial Interface

Figure 14. Connection Setup for Programming the EPCQ-L Device using the AS Interface

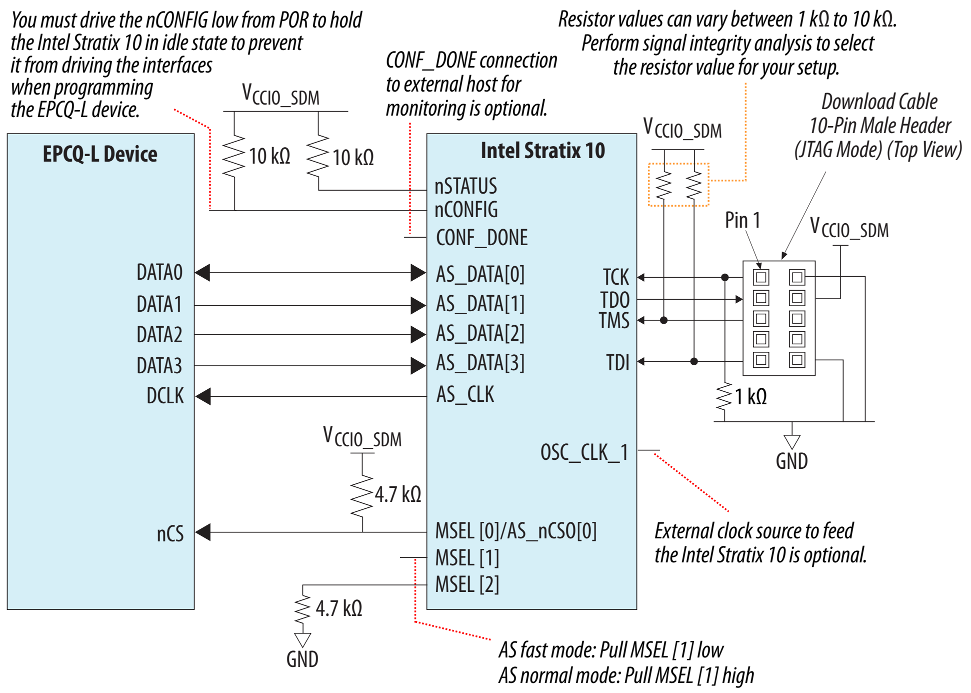


During the EPCQ-L device programming using the download cable through the AS header, the programmer serially transmits programming data, operation command, and address information to the EPCQ-L device on DATA0. During the EPCQ-L device verification using the download cable, DATA1 transfers the programming data back to the download cable.

When programming the EPCQ-L devices through the AS interface, ensure that the Intel Stratix 10 device does not drive or start the device configuration on the AS interface pins. In order to do that, your system must have the capability to change the MSEL setting to 111(JTAG configuration scheme) and re-power up the device if the device is powered up initially. After programming completes, you must change the MSEL setting to AS configuration scheme and then re-power up the device before the device configures itself.

3.2.3.2.2 Programming EPCQ-L Devices using the JTAG Interface

Figure 15. Connection Setup for Programming the EPCQ-L Devices using the JTAG Interface


You must drive the nCONFIG low from POR to hold the Intel Stratix 10 in idle state to prevent it from driving the interfaces when programming the EPCQ-L device.
CONF_DONE connection to external host for monitoring is optional.
Resistor values can vary between 1 kΩ to 10 kΩ. Perform signal integrity analysis to select the resistor value for your setup.
Download Cable 10-Pin Male Header (JTAG Mode) (Top View)
VCCIO_SDM
10 kΩ
10 kΩ
EPCQ-L Device
Intel Stratix 10
nSTATUS
nCONFIG
CONF_DONE
DATA0
DATA1
DATA2
DATA3
DCLK
nCS
AS_DATA[0]
AS_DATA[1]
AS_DATA[2]
AS_DATA[3]
AS_CLK
TCK
TDO
TMS
TDI
Pin 1
1 kΩ
GND
OSC_CLK_1
4.7 kΩ
MSEL [0]/AS_nCSO[0]
MSEL [1]
MSEL [2]
External clock source to feed the Intel Stratix 10 is optional.
AS fast mode: Pull MSEL [1] low
AS normal mode: Pull MSEL [1] high

3.2.3.2.3 Multiple EPCQ-L Devices Support

The Programmer File Generator tool in the Intel Quartus Prime software indicates the number of EPCQ-L devices required to fit the configuration file generated.

Requirements for using multiple EPCQ-L devices are:

- Only identical EPCQ-L devices are supported for the multiple EPCQ-L devices feature.
- Multiple EPCQ-L device feature supports up to 4 devices only.
- Each section of your configuration data must fit within a single EPCQ-L device.

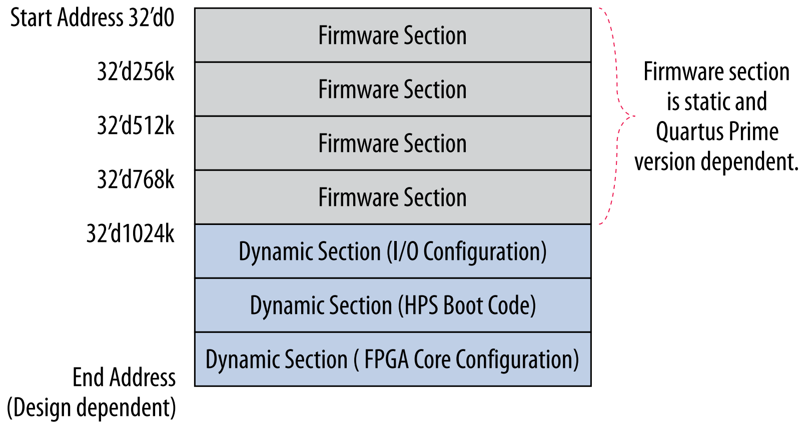
Note: The Intel Quartus Prime Programmer File Generator tool creates a `.jic` file based on the setting you set. The configuration fails if the wrong EPCQ-L device type is selected in the Programmer File Generator tool. However, the configuration can complete successfully if your design contains more EPCQ-L devices compared to your settings in Programmer File Generator tool.

3.2.3.3 EPCQ-L Memory Layout

EPCQ-L devices store the configuration data in sections.

The following diagram illustrates each section of the Intel Stratix 10 configuration data mapping in EPCQ-L memory sections.

Figure 16. EPCQ-L Memory Layout Diagram


Start Address 32'd0
32'd256k
32'd512k
32'd768k
32'd1024k
End Address (Design dependent)
Firmware Section
Firmware Section
Firmware Section
Firmware Section
Dynamic Section (I/O Configuration)
Dynamic Section (HPS Boot Code)
Dynamic Section (FPGA Core Configuration)
Firmware section is static and Quartus Prime version dependent.

- Using `.rpd` file for third-party programmer—you must ensure that the configuration data are stored starting from address 0 of the EPCQ-L device.
- Using `.jic` or `.pof` files for Intel Stratix 10 Programmer—the Intel Stratix 10 Programmer automatically programs the configuration data starting from address 0 of the EPCQ-L device.

3.2.4 Active Serial Configuration Software Settings

You must 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, perform 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. Turn on the **Use configuration device** and select your EPCQ-L device from the drop-down list.
   c. Select the desired AS clock frequency from the **Active serial clock source** drop-down list.
   d. Select **Auto** or **1.8 V** in the **Configuration device I/O voltage** drop-down list.
3. Click **OK** to confirm and close the **Device and Pin Options**.

3.2.5 Configuring Intel Stratix 10 Devices using AS Configuration

You must perform the following steps prior to configuring the Intel Stratix 10 using AS configuration scheme:

1. Generate `.pof`, `.jic`, or `.rpd` programming files using Convert Programming Files
2. Program the `.pof`, `.jic`, or `.rpd` file into the EPCQ-L

Note: You can use the Intel Quartus Prime Programmer to program the `.pof` or `.jic` file into the EPCQ-L device through an AS header or JTAG interface respectively. Alternatively, you can use third-party programmer to program the `.rpd` file into the EPCQ-L device.

Related Links

SDM Pin Mapping on page 10

3.2.5.1 Generating Programming Files using Convert Programming Files

The `.pof`, `.jic`, and `.rpd` files are generated from a `.sof` file using the Intel Quartus Prime Convert Programming Files tool.

To convert the programming files, perform the following steps:

1. On the **File** menu, click **Convert Programming Files**.
2. Under **Output programming file**, select **Programmer Object File (.pof)**, **JTAG Indirect Configuration File (.jic)**, or **Raw Programming Data File (.rpd)** in the Programming filetype list.
3. In the **Mode** list, select **Active Serial x4**.
4. Click **Option/Boot Info**. In the **Options setting** dialog box, set the RPD File Endianness to **Big Endian**

   Note: This step is applicable if you are generating `.rpd` only.
5. In the **File name** field, specify the file name for the programming file you want to create.
6. To generate a Memory Map File (`.map`), turn on **Create Memory Map File (Auto generate output_file.map)**.
7. To generate a Raw Programming Data (`.rpd`), turn on **Create config data RPD (Generate output_file_auto.rpd)**.
8. The `.sof` can be added through **Input files to convert** list.
9. Click **Generate** to generate related programming file.

3.2.5.2 Programming .pof files into EPCQ-L Device

To program the `.pof` into the EPCQ-L 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. To enable the real-time ISP mode, turn-on the **Enable real-time ISP to allow background programming**.
7. Click **Start** to start programming.

3.2.5.3 Programming .jic files into EPCQ-L Device

To program the `.jic` into the EPCQ-L 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.

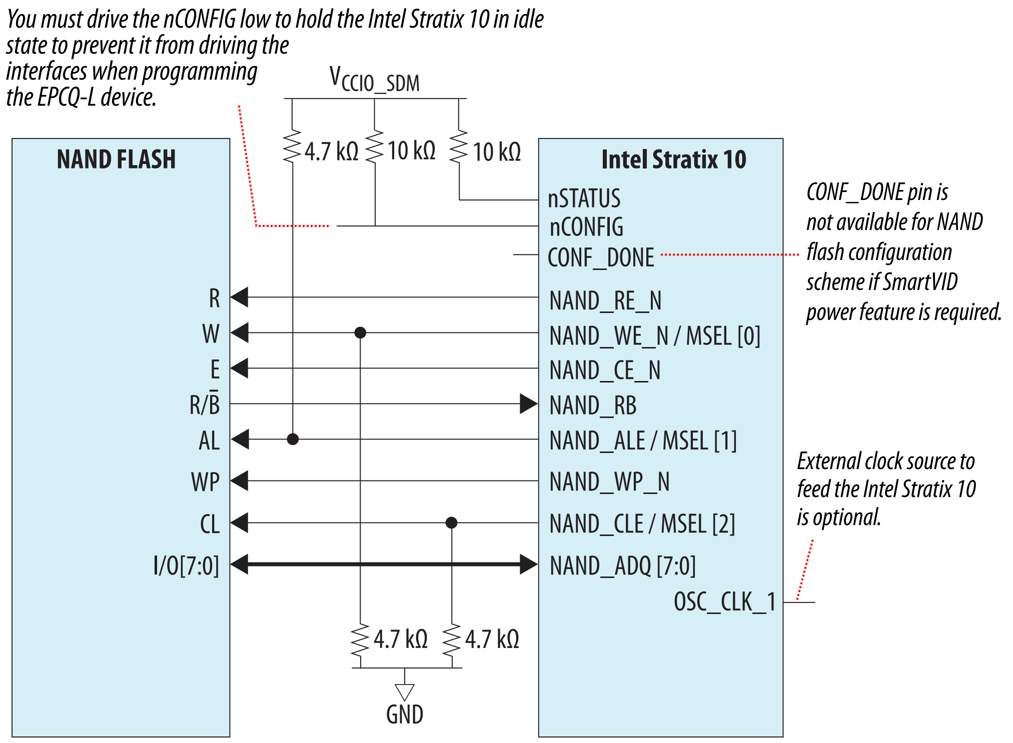
3.3 Configuration from NAND Flash

In the configuration scheme using NAND flash memories, configuration data is stored in the external NAND flash memory device. The SDM block in the Intel Stratix 10 device uses the on-chip NAND flash controller to interface with the NAND flash memory and read the configuration data from the NAND flash memory for the configuration process. The configuration scheme using NAND flash supports x8 (8-bit data width) NAND flash memories.

Note: If an HPS is present, you can use it to access the flash when the device is in user mode.

3.3.1 NAND Flash Single-Device Configuration

Figure 17. Connection Setup for NAND Flash Single-Device Configuration


You must drive the nCONFIG low to hold the Intel Stratix 10 in idle state to prevent it from driving the interfaces when programming the EPCQ-L device.
VCCIO_SDM
4.7 kΩ
10 kΩ
10 kΩ
NAND FLASH
R
W
E
R/B̄
AL
WP
CL
I/O[7:0]
Intel Stratix 10
nSTATUS
nCONFIG
CONF_DONE
NAND_RE_N
NAND_WE_N / MSEL [0]
NAND_CE_N
NAND_RB
NAND_ALE / MSEL [1]
NAND_WP_N
NAND_CLE / MSEL [2]
NAND_ADQ [7:0]
OSC_CLK_1
CONF_DONE pin is not available for NAND flash configuration scheme if SmartVID power feature is required.
External clock source to feed the Intel Stratix 10 is optional.
4.7 kΩ
4.7 kΩ
GND

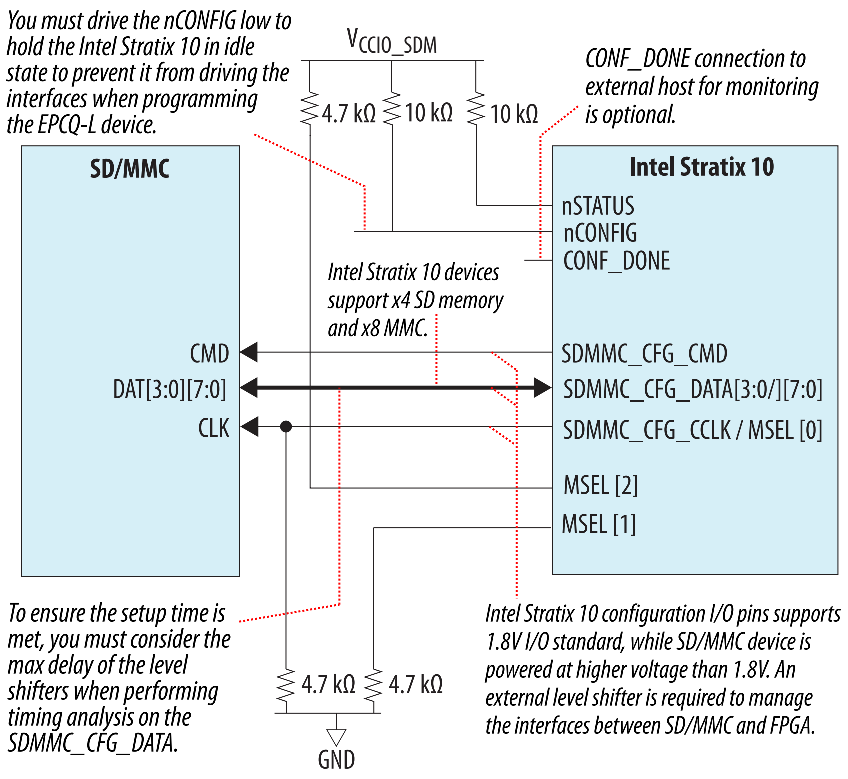
3.4 Configuration from SD/MMC

In the configuration scheme using SD memory cards, or MMC, configuration data is stored in the memory cards. The SDM block in the Intel Stratix 10 device uses the on-chip SD/MMC controller to interface with 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.

Note: If an HPS is present, you can use it to access the flash when the device is in user mode.

3.4.1 SD/MMC Single-Device Configuration

Figure 18. Connection Setup for SD/MMC Single-Device Configuration


You must drive the nCONFIG low to hold the Intel Stratix 10 in idle state to prevent it from driving the interfaces when programming the EPCQ-L device.
V_{CCIO_SDM}
4.7 kΩ
10 kΩ
10 kΩ
CONF_DONE connection to external host for monitoring is optional.
SD/MMC
Intel Stratix 10
nSTATUS
nCONFIG
CONF_DONE
Intel Stratix 10 devices support x4 SD memory and x8 MMC.
CMD
DAT[3:0][7:0]
CLK
SDMMC_CFG_CMD
SDMMC_CFG_DATA[3:0/][7:0]
SDMMC_CFG_CCLK / MSEL [0]
MSEL [2]
MSEL [1]
To ensure the setup time is met, you must consider the max delay of the level shifters when performing timing analysis on the SDMMC_CFG_DATA.
4.7 kΩ
4.7 kΩ
GND
Intel Stratix 10 configuration I/O pins supports 1.8V I/O standard, while SD/MMC device is powered at higher voltage than 1.8V. An external level shifter is required to manage the interfaces between SD/MMC and FPGA.

3.5 JTAG Configuration

JTAG-chain device programming is ideal during development. Intel Stratix 10 devices can be reconfigured with a new design faster than programming that design into flash memory. JTAG can also be used to reprogram a corrupted flash memory that prevents the Intel Stratix 10 device from configuring using its normal configuration scheme. The Intel Quartus Prime software generates an SRAM Object File (`.sof`) that you can use for JTAG configuration using a download cable in the Intel Quartus Prime software programmer. Use the Intel FPGA download cables to configure the Intel Stratix 10 devices through its JTAG interface. The Intel FPGA Download Cable II and Intel FPGA

Ethernet Cable can support the VCCIO_SDM supply at 1.8 V. Alternatively, you can use the JAM™ Standard Test and Programming Language (STAPL) Format File (`.jam`) or JAM Byte Code File (`.jbc`) with other third-party programmer tools.

Intel Stratix 10 devices supports configuration data compression in JTAG configuration scheme.

Related Links

- Programming Support for Jam STAPL Language
- Intel FPGA Download Cable II
- Intel Stratix 10 Device Datasheet
- Intel Stratix 10 Configuration Pins on page 9

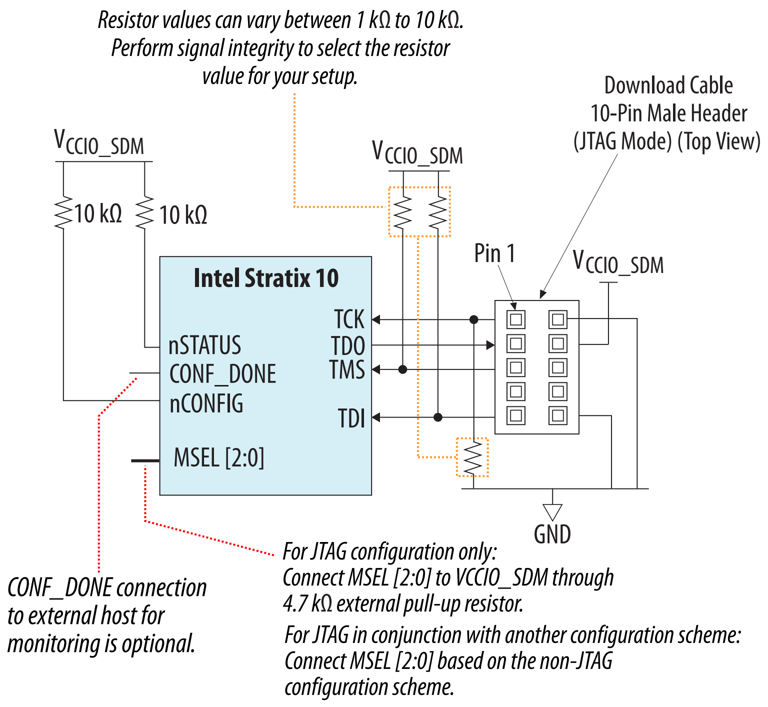
3.5.1 JTAG Single-Device Configuration

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.

3.5.1.1 JTAG Single-Device Configuration using Download Cable Connections

Figure 19. Connection Setup for JTAG Single-Device Configuration using Download Cable

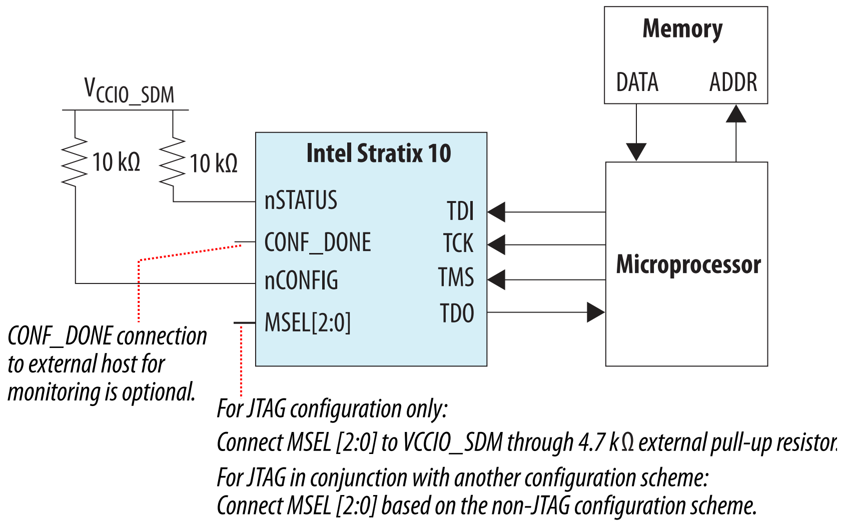

Resistor values can vary between 1 kΩ to 10 kΩ. Perform signal integrity to select the resistor value for your setup.
Download Cable 10-Pin Male Header (JTAG Mode) (Top View)
VCCIO_SDM
10 kΩ
10 kΩ
VCCIO_SDM
Pin 1
VCCIO_SDM
Intel Stratix 10
TCK
nSTATUS
TDO
CONF_DONE
TMS
nCONFIG
TDI
MSEL [2:0]
GND
For JTAG configuration only: Connect MSEL [2:0] to VCCIO_SDM through 4.7 kΩ external pull-up resistor.
For JTAG in conjunction with another configuration scheme: Connect MSEL [2:0] based on the non-JTAG configuration scheme.
CONF_DONE connection to external host for monitoring is optional.

Related Links

Intel FPGA Download Cable II

3.5.1.2 JTAG Single-Device Configuration using a Microprocessor

Figure 20. Connection Setup for JTAG Single-Device Configuration using a Microprocessor


Memory
DATA
ADDR
VCCIO_SDM
10 kΩ
10 kΩ
Intel Stratix 10
nSTATUS
CONF_DONE
nCONFIG
MSEL[2:0]
TDI
TCK
TMS
TDO
Microprocessor
CONF_DONE connection to external host for monitoring is optional.
For JTAG configuration only:
Connect MSEL [2:0] to VCCIO_SDM through 4.7 kΩ external pull-up resistor.
For JTAG in conjunction with another configuration scheme:
Connect MSEL [2:0] based on the non-JTAG configuration scheme.

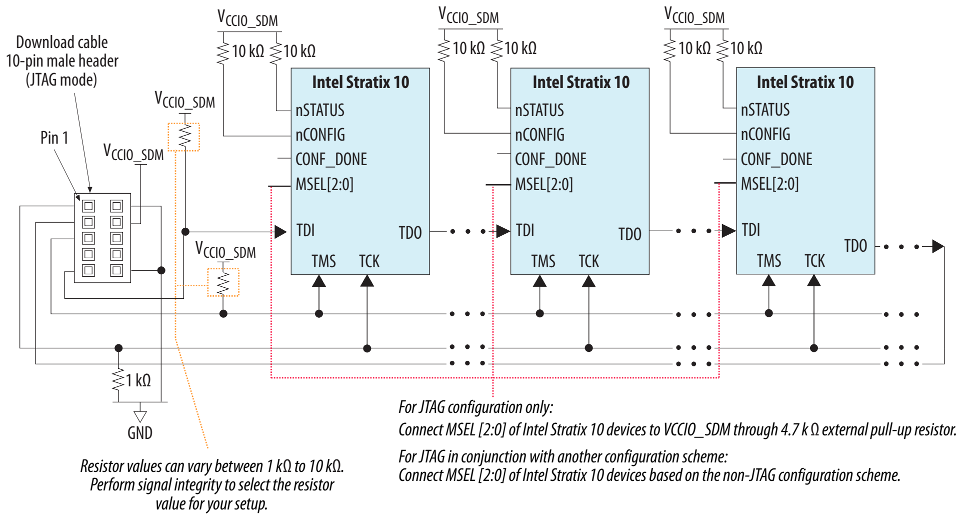
3.5.2 JTAG Multi-Device Configuration

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

- One JTAG-compatible header is connected to several devices in a JTAG chain. The number of devices in the chain is limited only by the drive capability of the download cable.
- 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.

3.5.2.1 JTAG Multi-Device Configuration using Download Cable

Figure 21. Connection Setup for JTAG Multi Device Configuration using Download Cable


Download cable
10-pin male header
(JTAG mode)
Pin 1
VCCIO_SDM
10 kΩ
1 kΩ
GND
Intel Stratix 10
nSTATUS
nCONFIG
CONF_DONE
MSEL[2:0]
TDI
TDO
TMS
TCK
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 Stratix 10 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 Stratix 10 devices based on the non-JTAG configuration scheme.

4 Intel Stratix 10 Configuration Features

4.1 Remote System Upgrade

Intel Stratix 10 devices contain dedicated remote system upgrade circuitry. You can use this feature to upgrade your system from a remote location. You can design your system to manage remote upgrades of the application configuration images in the configuration device. When an error occurs, the circuitry detects the error, reverts to a safe configuration image that is the factory image, and provides error status to your design.

You can also create one or more application images for the device depending on the storage space of your flash device. An application image contains selected functionalites to be implemented in the target device.

4.2 Device Security

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

- 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 intrusion mitigation
- Anti-tampering response

4.3 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 particular 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.

For more information, refer to the *Creating a Partial Reconfiguration Design* chapter of the *Intel Quartus Prime Pro Edition Handbook Volume 1: Design and Compilation*.

Related Links

Creating a Partial Reconfiguration Design chapter of the Handbook Volume 1: Design and Compilation

4.4 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.

5 Intel FPGA Parallel Flash Loader II IP Core

You can use the PFL II IP core with an external host such as the MAX II, MAX V, or Intel MAX 10 devices to program configuration data into a flash memory device using JTAG interface and to configure the Intel Stratix 10 device with Avalon-ST configuration scheme from the flash memory device.

Related Links

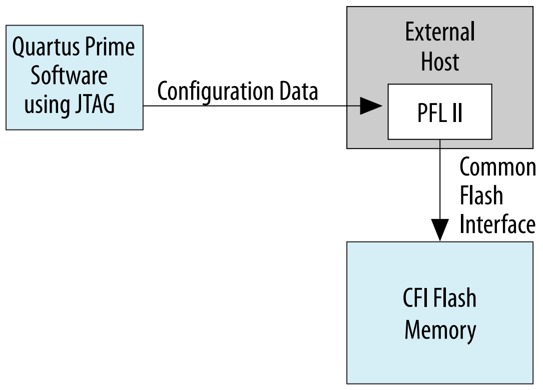
- Avalon-ST Configuration on page 17
- Avalon-ST Single-Device Configuration on page 19

5.1 Functional Description

5.1.1 Programming CFI Flash

The external host's JTAG block interfaces directly with the logic array in a special JTAG mode. This mode brings the JTAG chain through the logic array instead of the host boundary-scan cells (BSCs). The PFL II IP core provides JTAG interface logic to convert the JTAG stream provided by the Intel Quartus Prime software and to program the CFI flash memory devices connected to the host I/O pins.

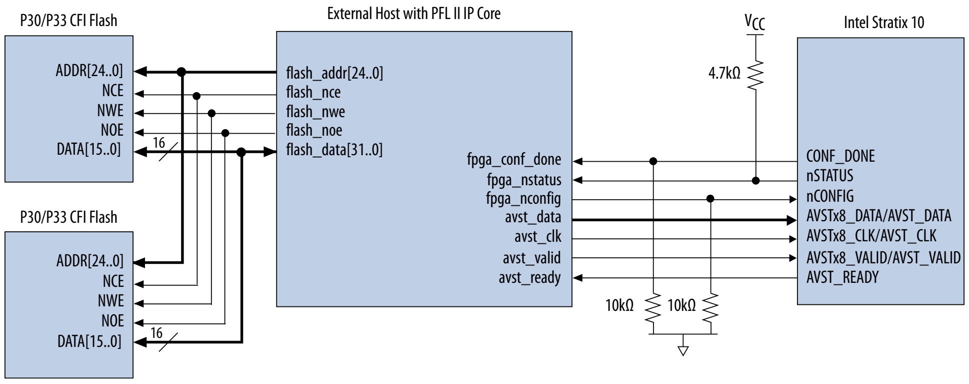
Figure 22. Programming the CFI Flash Memory with the JTAG Interface



The PFL II IP core supports dual P30 or P33 CFI flash memory devices in burst read mode to achieve faster configuration time. Two identical P30 or P33 CFI flash memory devices connect 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 23. PFL II IP core with Dual P30 or P33 CFI Flash Memory Devices

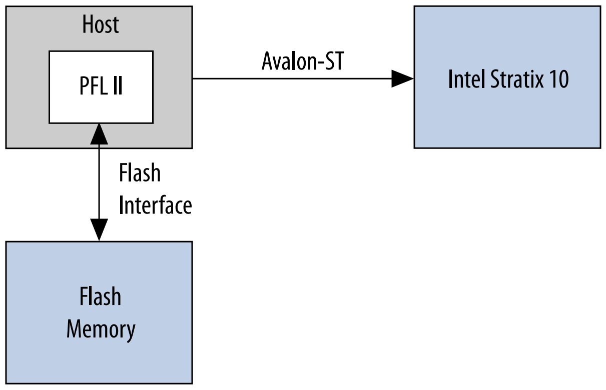
The flash memory devices in the dual P30 or P33 CFI flash solution must have the same memory density from the same device family and manufacturer.



5.1.2 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 using the Avalon-ST configuration scheme.

Figure 24. FPGA Configuration with Flash Memory Data



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

- You want to use fewer LEs.
- 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, the flash memory device contains initialization storage for an application specific standard product (ASSP). You can use the PFL II IP core to program the flash memory device with the initialization data and also create your own design source code to implement the read and initialization control with the host logic.

Creating Separate PFL II Functions

To create separate PFL II functions, follow these steps:

1. To create a PFL II instantiation, select **Flash Programming Only** mode.
2. Assign the pins appropriately.
3. Compile and generate a `.pof` for the flash memory device. Ensure that you tri-state all unused I/O pins.
4. To create another PFL II instantiation, select **Configuration Control Only mode**.
5. Instantiate this configuration controller into your production design.
6. Whenever you must program the flash memory device, program the CPLD with the flash memory device `.pof` and update the flash memory device contents.
7. Reprogram the host with the production design `.pof` that includes the configuration controller.

Note: All unused pins are set to ground by default. 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 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.

5.1.3 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 25. Micron J3 Flash Memory in 8-Bit Mode

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

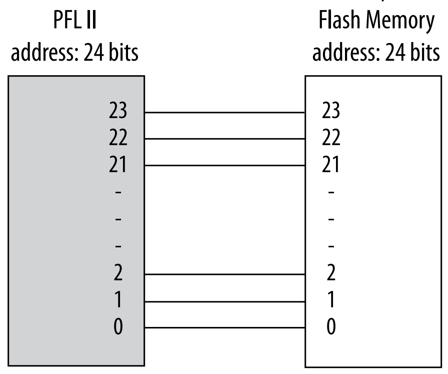


Figure 26. Micron J3, P30, and P33 Flash Memories in 16-Bit Mode

The flash memory addresses in Micron J3, P30, and P33 16-bit flash memory shift one bit down in comparison with the flash addresses in PFL II IP core. The flash address in the Micron J3, P30, and P33 flash memory starts from bit 1 instead of bit 0.

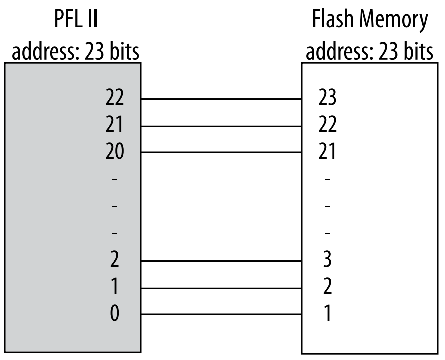


Figure 27. Cypress and Micron M28, M29 Flash Memory in 8-Bit Mode

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.

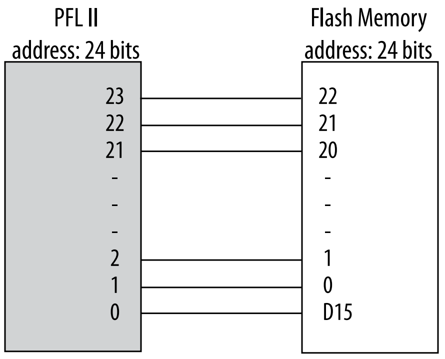
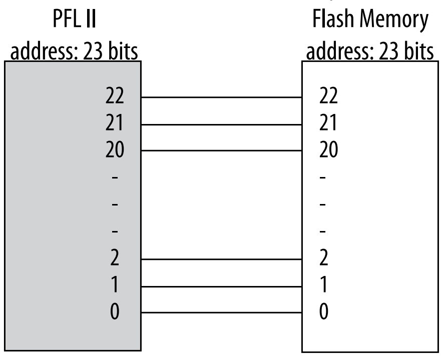


Figure 28. Cypress and Micron M28, M29 Flash Memory in 16-Bit Mode

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



5.1.4 Implementing Page in the Flash .pof

The PFL II IP core stores configuration data in a maximum of eight pages in a flash memory block. Each page holds the configuration data for a single FPGA chain.

The total number of pages and the size of each page depends on the density of the flash. These pages allow you to store designs for different FPGA chains or different designs for the same FPGA chain in different pages.

Use the generated `.sof` files to create a flash memory device `.pof`. When converting these `.sof` files to a `.pof`, use the following address modes to determine the page address:

- Block mode—allows you to specify the start and end addresses for the page.
- Start mode—allows you to specify only the start address. You can locate the start address for each page on an 8-KB boundary. If the first valid start address is `0×000000`, the next valid start address is an increment of `0×2000`.
- Auto mode—allows the Intel Quartus Prime software to automatically determine the start address of the page. The Intel Quartus Prime software aligns the pages on a 128-KB boundary; for example, if the first valid start address is `0×000000`, the next valid start address is an increment of `0×20000`.

5.1.4.1 Storing Option Bits

The PFL II IP core requires you to allocate space in the flash memory device for option bits. The option bits sector contains information about the start address for each page, the `.pof` version used for flash programming, and the Page-Valid bits. You must specify the options bits sector address in the flash memory device when converting the `.sof` files to a `.pof` and creating a PFL II design.

Table 10. Option Bits Sector Format

Offset address `0x80` stores the `.pof` version required for programming flash memory. This `.pof` version applies to all eight pages of the configuration data. The PFL II IP core requires the `.pof` version to perform a successful FPGA configuration process.

Sector Offset	Value
0x00–0x03	Page 0 start address
0x04–0x07	Page 0 end address
0x08–0x0B	Page 1 start address
0x0C–0x0F	Page 1 end address
0x10–0x13	Page 2 start address
0x14–0x17	Page 2 end address
0x18–0x1B	Page 3 start address
0x1C–0x1F	Page 3 end address
0x20–0x23	Page 4 start address
0x24–0x27	Page 4 end address
0x28–0x2B	Page 5 start address
0x2C–0x2F	Page 5 end address
0x30–0x33	Page 6 start address
0x34–0x37	Page 6 end address
0x38–0x3B	Page 7 start address

continued...

Sector Offset	Value
0x3C–0x3F	Page 7 end address
0x40–0x7F	Reserved
0x80[18]	.pof version
0x81-0xFF	Reserved

The Intel Quartus Prime Convert Programming File tool generates the information for the `.pof` version when you convert the `.sof` files to `.pof` files.

The value for the `.pof` version for Intel Stratix 10 is `0x05`.

Caution: Do not overwrite any information in the option bits sector to prevent the PFL II IP core from malfunctioning, and always store the option bits in unused addresses in the flash memory device.

5.1.4.1.1 Restoring Option Bit Start and End Address

You can restore 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.

The value for bit `[31:0]` for the start address of a page consists from the following format. The value for bit `[31:0]` for the end address of a page represents the 32 bits addressable end address.

Table 11. Start Address Bit Content

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

Table 12. End Address Bit Content

Bit	Width	Description
31:0	32	Addressable end address

To restore the addresses:

- Start address—append 13 bits of `0` to the addressable start address
- End address—append 2 bits of `1` to the addressable end address

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

You have a converted a `.pof` file that has two page address with the following values in the option bit sector offset:

Sector Offset	Value
0x00 – 0x03	0x00004000
0x04 – 0x07	0x00196E30
0x08 – 0x0B	0x001C0000
0x0C – 0x0F	0x00352E30

Page 0 start address = Bit[31:11] appends with 0000000000000

= 00000000000000000010000000000000000

= 0x10000

Page 0 end address = 0x00196E30 appends with 2'b11

= 00011001011011100011000011

= 0x65B8C3

Page 1 start address = Bit[31:11] appends with 0000000000000

= 00000000000111000000000000000000000

= 0x700000

Page 1 end address = 0x00352E30 appends with 2'b11

= 00000000001101010010111000011000011

= 0xD4B8C3

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

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

Figure 29. Implementing Page Mode and Option Bits in the CFI Flash Memory Device

- The end address depends on the density of the flash memory device. For the address range for devices with different densities, refer Byte Address Range table.
- You must specify the byte address for the option bits sector.

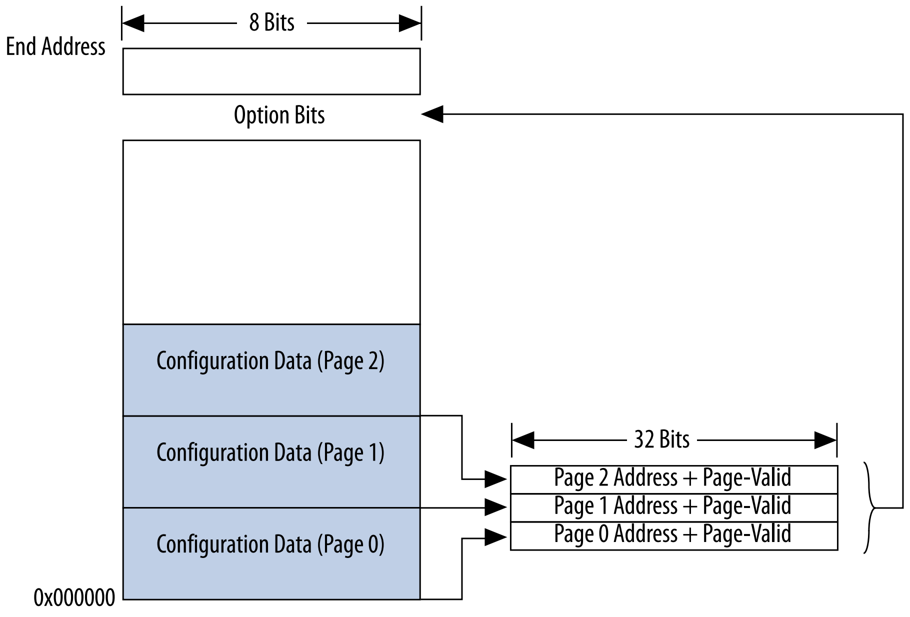

8 Bits
End Address
Option Bits
Configuration Data (Page 2)
Configuration Data (Page 1)
Configuration Data (Page 0)
0x000000
32 Bits
Page 2 Address + Page-Valid
Page 1 Address + Page-Valid
Page 0 Address + Page-Valid

Figure 30. Page Start Address, End Address, and Page-Valid Bit Stored as Option Bits

Bits 0 to 12 for the page start address are set to zero and are not stored as option bits. The Page-Valid bits indicate whether each page is successfully programmed. The PFL II IP core programs the Page-Valid bits after successfully programming the pages.

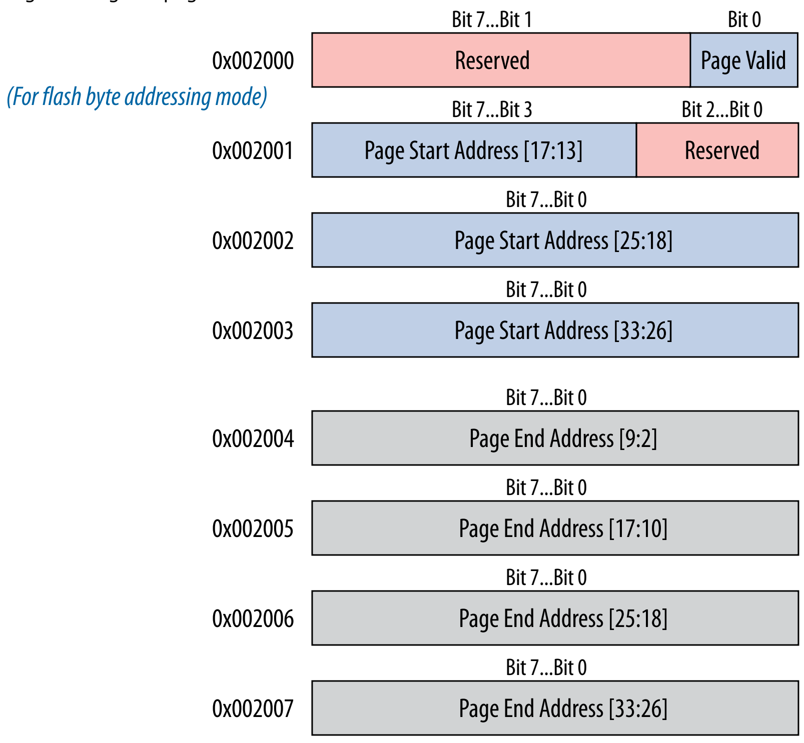


Table 13. Byte Address Range for CFI Flash Memory Devices with Different Densities

CFI Device (Megabit)	Address Range
8	0x0000000–0x00FFFFF
16	0x0000000–0x01FFFFF
32	0x0000000–0x03FFFFF
64	0x0000000–0x07FFFFF
128	0x0000000–0x0FFFFFF
256	0x0000000–0x1FFFFFF
512	0x0000000–0x3FFFFFF
1024	0x0000000–0x7FFFFFF

5.2 Using PFL II IP Core

5.2.1 Converting .sof to .pof File

To convert the `.sof` file to a `.pof`, follow these steps:

1. On the **File** menu, click **Convert Programming Files**.
2. For **Programming file type**, specify **Programmer Object File** (`.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 (Mb) capacity.
4. For **Mode**, select the configuration mode that matched to the `.sof` file. The available configuration modes are AvSTx8/AvSTx16/AvSTx32.
5. To add the configuration data, under **Input files to convert**, select **SOF Data**.
6. Click **Add File** and browse to the `.sof` files you want to add.

   You can place more than one `.sof` in the same page if you intend to configure a chain of FPGAs. The order of the `.sof` files must follow the order of the devices in the chain. If you want to store the data from other `.sof` files in a different page, click **Add SOF page**. Add the `.sof` files to the new page.
7. Select **SOF Data** and click **Properties** to set the page number and name. Under **Address mode for selected pages**, 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, or select Start to specify the start address only and click OK.
8. You can also store Hexadecimal (Intel-Format) File (`.hex`) user data in the flash memory device:
   a. In the **Input files to convert** sub-window of the **Convert Programming Files**, select **Add Hex Data**.
   b. In the **Add Hex Data** dialog box, select either absolute or relative addressing mode.
      - If you select absolute addressing mode, the data in the `.hex` is programmed in the flash memory device at the exact same address location listed in the `.hex`.
      - If you select relative addressing mode, specify a start address. The data in the `.hex` is programmed into the flash memory device with the specific start address, and the differences between the addresses are kept. If no address is specified, the software selects an address.

   Note: You can also add other non-configuration data to the `.pof` by selecting the `.hex` that contains your data when creating the flash memory device `.pof`.
9. Click **Options** to specify the start address to store the option bits.

This start address must be identical to the address you specify when creating the PFL II IP core. Ensure that the option bits sector does not overlap with the configuration data pages and that the start address resides on an 8-KB boundary.

10. To generate programming files with the enhanced bitstream compression feature, turn on the **Enable enhanced bitstream-compression** when available in the **Options** dialog box and click **OK**.
11. Click **Generate** to create the `.pof`.

5.2.2 Programming CPLDs and Flash Memory Devices

You can either program the CPLD and the flash memory concurrently or separately.

5.2.2.1 Programming CPLDs and Flash Memory Devices Concurrently

To program concurrently, first program the CPLD, then the flash memory device. Follow these steps:

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 density of the flash memory device to be programmed.
4. Right-click the necessary flash memory device density and click **Change File**.
5. Select the `.pof` generated for the flash memory device. The `.pof` for the flash memory device is attached to the `.pof` of the CPLD.
6. Add other programming files if your chain has other 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.

5.2.2.2 Programming CPLDs and Flash Memory Devices Separately

To program the CPLD and the flash memory devices separately, follow these steps:

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 P30 or P33, the programmer window shows a dual P30 or P33 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 necessary flash memory device density and click **Change File**.

Note: You must select the density that is equivalent to the sum of the density of two CFI flash memory devices. For example, if you require two 512-Mb CFI flash memory devices, then select CFI 1 Gbit.

7. Select the `.pof` generated for the flash memory device. The `.pof` for the flash memory device is attached 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: 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, provided the CPLD contains the PFL II IP core. 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.

5.2.3 Defining New CFI Flash Device

The PFL II IP core supports Intel-compatible 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 memory device feature.

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

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

Table 14. Functions of the Define CFI Flash Device Feature

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

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

Table 15. Parameter Settings for New CFI Flash Device

Parameter	Description
CFI flash device name	Define the CFI flash name
CFI flash device ID	Specify the CFI flash identifier code
CFI flash manufacturer ID	Specify the CFI flash manufacturer identification number

continued...

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

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

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

5.3 Supported CFI Flash Memory Devices

Table 16. CFI Flash Memory Devices Supported by PFL II IP Core

Manufacturer	Product Family	Data Width	Density (Megabit)	Device Name[19]
Micron	C3	16	8	28F800C3
			16	28F160C3
			32	28F320C3
			64	28F640C3
	J3	8 or 16	32	28F320J3
			64	28F640J3
			128	28F128J3
		16	256	JS29F256J3
	P30	16	64	28F640P30
			128	28F128P30
			256	28F256P30
			512	28F512P30
			1000	28F00AP30
			2000	28F00BP30
	P33	16	64	28F640P33
			128	28F128P33
			256	28F256P33

continued...

(19) The PFL II IP core supports top and bottom boot block of the flash memory devices. For Micron flash memory devices, the PFL II IP core supports top, bottom, and symmetrical blocks of flash memory devices.

Manufacturer	Product Family	Data Width	Density (Megabit)	Device Name[19]
			512	28F512P33
			1000	28F00AP33
			2000	28F00BP33
	M29EW	8 or 16	256	28F256M29EW
			512	28F512M29EW
			1000	28F00AM29EW
	M29W	8 or 16	16	M28W160CT
				M28W160CB
				M29W160F7
				M29W160FB
			32	M29W320E
				M29W320FT
				M29W320FB
			64	M29W640F
				M29W640G
			128	M29W128G
			256	M29W256G
	M29DW	8 or 16	32	M29DW323DT
				M29DW323DB
	G18	16	512	MT28GU512AAA1EGC-0SIT
			1024	MT28GU01GAAA1EGC-0SIT
	M58BW	32	16	M58BW16FT
				M58BW16FB
			32	M58BW32FT
		16 or 32	32	M58BW32FB
Cypress	GL-P[20]	8 or 16	128	S29GL128P
			256	S29GL256P
			512	S29GL512P
			1024	S29GL01GP
	AL-D	8 or 16	16	S29AL016D
			32	S29AL032D

continued...

(19) The PFL II IP core supports top and bottom boot block of the flash memory devices. For Micron flash memory devices, the PFL II IP core supports top, bottom, and symmetrical blocks of flash memory devices.

(20) Supports page mode.

Manufacturer	Product Family	Data Width	Density (Megabit)	Device Name[19]
	AL-J	8 or 16	16	S29AL016J
	AL-M	8 or 16	16	S29AL016M
	JL-H	8 or 16	32	S29JL032H
			64	S29JL064H
	WS-N	16	128	S29WS128N
	GL-S	16	128	S29GL128S
			256	S29GL256S
			512	S29GL512S
			1024	S29GL01GS
Macronix	MX29LV	16	16	MX29LV160D
			32	MX29LV320D
			64	MX29LV640D
				MX29LV640E
	MX29GL	16	128	MX29GL128E
			256	MX29GL256E
Eon Silicon Solution	EN29LV	16	16	EN29LV160B
	EN29GL	16	32	EN29LV320B
			128	EN29GL128

5.4 Parameters

Table 17. PFL II General Parameters

Options	Value	Description
Operating mode	• Flash Programming and FPGA Configuration • Flash Programming • FPGA Configuration	Specifies the operating mode of flash programming and FPGA configuration control in one IP core or separate these functions into individual blocks and functionality.
Targeted flash device	• CFI Parallel Flash	Specifies the flash memory device connected to the PFL II IP core.
Tri-state flash bus	• On • Off	Allows the PFL II IP core to tri-state all pins interfacing with the flash memory device when the PFL II IP core does not require an access to the flash memory.

(19) The PFL II IP core supports top and bottom boot block of the flash memory devices. For Micron flash memory devices, the PFL II IP core supports top, bottom, and symmetrical blocks of flash memory devices.

Table 18. PFL II Flash Interface Setting Parameters

Options	Value	Description
Number of flash devices used	• CFI Parallel Flash: 1–16	Specifies the number of flash memory devices connected to the PFL II IP core.
Largest flash density	• CFI Parallel Flash: 8 Mbit–2 Gbit	Specifies the density of the flash memory device to be programmed or used for FPGA configuration. If you have more than one flash memory device connected to the PFL II IP core, specify the largest flash memory device density. For dual P30/P33 CFI flash, select the density that is equivalent to the sum of the density of two flash devices. For example, if you use two 512-Mb CFI flashes, you must select **CFI 1 Gbit**.
Flash interface data width	CFI Parallel Flash: • 8 • 16 • 32	Specifies the flash data width in bits. The flash data width depends on the flash memory device you use. For multiple flash memory device support, the data width must be the same for all connected flash memory devices. Select the flash data width that is equivalent to the sum of the data width of two flash devices. For example, if you are targeting dual P30 or P33 solution, you must select **32 bits** because each CFI flash data width is 16 bits.
User 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 device, connect this pin to the RESET# pin of the flash device.

Table 19. PFL II Flash Programming Parameters

Options	Value	Description
Flash programming IP optimization	• 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 less LEs, but the flash programming time is longer.
FIFO size	—	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 20. PFL II FPGA Configuration Parameters

Options	Value	Description
External clock frequency	—	Specifies the user-supplied clock frequency for the IP core to configure the FPGA. The clock frequency must not exceed two times the maximum clock (AVST_CLK) frequency acceptable by the FPGA for configuration. The PFL II IP core can divide the frequency of the input clock maximum by two.
Flash access time	—	Specifies the access time of the flash. You can get the maximum access time that a flash memory device requires from the flash datasheet. Intel recommends specifying a flash access time that is the same as or longer than the required time.

continued...

Options	Value	Description
		For CFI parallel flash, the unit is in ns and for NAND flash, the unit is in us. NAND flash uses page instead of byte, and requires more access time. This option is disabled for quad SPI flash.
Option bits byte address	—	Specifies the start address in which the option bits are stored in the flash memory. The start address must reside on an 8-KB boundary. See related for more information about option bits.
FPGA configuration scheme	• Avalon-ST x8 • Avalon-ST x16 • Avalon-ST x32	Select the FPGA configuration scheme.
Configuration failure response options	• Halt • Retry same page • Retry from fixed address	Configuration behavior after configuration failure. • If you select **Halt**, the FPGA configuration stops completely after failure. • If you select **Retry same page**, after failure, the PFL II IP core reconfigures the FPGA with data from the same page of the failure. • If you select **Retry from fixed address**, the PFL II IP core reconfigures the FPGA with data from a fixed address in the next option field after failure.
Byte address to retry from on configuration failure	—	If you select **Retry from fixed address** for configuration failure option, this option specifies the flash address for the PFL II IP core to read from the reconfiguration for a configuration failure.
Include input to force reconfiguration	• On • Off	Includes an optional reconfiguration input pin (`pfl_nreconfigure`) to enable a reconfiguration of the FPGA.
Watchdog timer	• On • Off	Enables a watchdog timer for remote system upgrade support. Turning on this option enables the `pfl_reset_watchdog` input pin and `pfl_watchdog_error` output pin, and specifies the time period before the watchdog timer times out. This watchdog timer is a time counter which runs at the `pfl_clk frequency`.
Time period before the watchdog timer times out	—	Specifies the time out period of the watchdog timer. The default time out period is 100 ms
Use advance read mode	• Normal Mode • Intel Burst Mode (P30 or P33) • Cypress Page Mode (GL) • Micron Burst Mode (M58BW)	An option to improve the overall flash access time for the read process during the FPGA configuration. • Normal mode—Applicable for all flash memory • Intel Burst mode—Applicable for Micron P30 and P33 flash memory only. Reduces sequential read access time • Cypress page mode—Applicable for Cypress GL flash memory only • Micron burst mode—Applicable for Micron M58BW flash memory only For more information about the read-access modes of the flash memory device, refer to the respective flash memory data sheet.
Latency count	• 3 • 4 • 5	Specify the latency count for Intel Burst Read mode. Only available when enable Intel Burst Mode.

5.5 Signals

Table 21. PFL II Signals

Pin	Type	Weak Pull-Up	Function
pfl_nreset	Input	—	Asynchronous reset for the PFL II IP core. Pull high to enable FPGA configuration. To prevent FPGA configuration, pull low when you do not use the PFL II IP core. This pin does not affect the flash programming functionality of the PFL II IP core.
pfl_flash_access_granted	Input	—	Used for system-level synchronization. This pin is driven by a processor or any arbitrator that controls access to the flash. This active-high pin is connected permanently high if you want the PFL II IP core to function as the flash master. Pulling the `pfl_flash_access_granted` pin low prevents the JTAG interface from accessing the flash and FPGA configuration.
pfl_clk	Input	—	User input clock for the device. Frequency must match the frequency specified in the IP core and must not be higher than the maximum DCLK frequency specified for the specific FPGA during configuration. This pins are not available for the flash programming option in the PFL II IP core.
fpga_pgm[]	Input	—	Determines the page for the configuration. This pins are not available for the flash programming option in the PFL II IP core.
fpga_conf_done	Input	10-kW Pull-Up Resistor	Connects to the CONF_DONE pin of the FPGA. The FPGA releases the pin high if the configuration is successful. During FPGA configuration, this pin remains low. This pins are not available for the flash programming option in the PFL II IP core.
fpga_nstatus	Input	10-kW Pull-Up Resistor	Connects to the nSTATUS pin of the FPGA. This pin must be released high before the FPGA configuration and must stay high throughout 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 pins are not available for the flash programming option in the PFL II IP core.
pfl_nreconfigure	Input	—	A low signal at this pin initiates FPGA reconfiguration. You can reconnect this pin to a switch for more flexibility to set this input pin high or low to control FPGA reconfiguration. When FPGA reconfiguration is initiated, the `fpga_nconfig` pin is pulled low to reset the FPGA device. The `pfl_clk`. pin registers this signal. This pins are not available for the flash programming option in the PFL II IP core.
pfl_flash_access_request	Output	—	Used for system-level synchronization. When necessary, this pin connects to a processor or an arbitrator. The PFL II IP core drives this pin high when the JTAG interface accesses the flash or the PFL II IP core configures the FPGA. This output pin works in conjunction with the `flash_noe` and `flash_nwe` pins.
flash_addr[]	Output	—	Address inputs for memory addresses. The width of the address bus line depends on the density of the flash memory device and the width of the

continued...

Pin	Type	Weak Pull-Up	Function
			`flash_data` bus. The output of this pin depends on the setting of the unused pins if you did not select the PFL II interface tri-state option when the PFL II is not accessing the flash memory device.
`flash_data[]`	Input or Output (bidirectional pin)	—	Data bus to transmit or receive 8- or 16-bit data to or from the flash memory in parallel. The output of this pin depends on the setting of the unused pins if you did not select the PFL II interface tri-state option when the PFL II is not accessing the flash memory device. (21)
`flash_nce[]`	Output	—	Connects to the `nCE` pin of the flash memory device. A low signal enables the flash memory device. Use this pin for multiple flash memory device support. The `flash_nce` pin is connected 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.
`flash_nwe`	Output	—	Connects to the `nWE` pin of the flash memory device. A low signal enables write operation to the flash memory device.
`flash_noe`	Output	—	Connects to the `nOE` pin of the flash memory device. A low signal enables the outputs of the flash memory device during a read operation.
`flash_clk`	Output	—	Used 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 single CFI flash. In dual P30 or P33 CFI flash solution, the `flash_clk` frequency runs at a quarter of the `pfl_clk` frequency. Use this pin for burst mode only. Do not connect these pins from the flash memory device to the host if you are not using burst mode.
`flash_nadv`	Output	—	Used for burst mode. Connects to the address valid input pin of the flash memory device. Use this signal for latching the start address. Use this pin for burst mode only. Do not connect these pins from the flash memory device to the host if you are not using burst mode.
`flash_nreset`	Output	—	Connects to the reset pin of the flash memory device. A low signal resets the flash memory device.

continued...

(21) Intel recommends not inserting logic between the PFL II pins and the host I/O pins, especially on the flash_data and fpga_nconfig pins.

Pin	Type	Weak Pull-Up	Function
`fpga_nconfig`	Open Drain Output	10-kW Pull-Up Resistor	Connects to the `nCONFIG` pin of the FPGA. A low pulse resets the FPGA and initiates configuration. This pins are not available for the flash programming option in the PFL II IP core. (21)
`pfl_reset_watchdog`	Input	—	A toggle signal to reset the watchdog timer before the watchdog timer times out. Hold the signal high or low for at least two clock cycles of the `pfl_clk` frequency to correctly reset the watchdog timer.
`pfl_watchdog_error`	Output	—	A high signal indicates an error to the watchdog timer.

Related Links

Avalon Streaming Interface Specification

6 Document Revision History for Intel Stratix 10 Configuration User Guide

Date	Version	Changes
November 2017	2017.11.09	• Removed link to the *Configuration via Protocol (CvP) Implementation User Guide*. • Updated titles for *Device Security*,*Partial Reconfiguration*, and *Configuration via Protocol*.
November 2017	2017.11.06	• Updated *Option Bits Sector Format* table. • Updated a step in *Setting Additional Configuration Pins*. • Added *Converting .sof to .pof File* and *Programming CPLDs and Flash Memory Devices*. • Updated the .pof version value in *Storing Option Bits*. • Added information about restoring start and end address for option bits in *Restoring Option Bit Start and End Address*. • Added note about pull-down resistor is recommended for `CONF_DONE` and `INIT_DONE` pins in *Additional Configuration Pin Functions*. • Added new subsection *Multiple EPCQ-L Devices Support*. • Added *Configuration Pins I/O Standard and Drive Strength* table. • Updated information about maximum additional data words when using 2-stage register synchronizer. • Updated the equation for minimum AS configuration time estimation. • Added *RBF Configuration File Format* section explaining the format of the .rbf file. • Updated *Configuration Sequence* to state that a firmware which is part of the configuration data if loaded in the device initially. • Updated description for **Number of flash devices used** parameter in the *PFL II Flash Interface Setting Parameters* table. • Updated *Configuration via Protocol* overview and added link to the Configuration via Protocol (CvP) Implementation User Guide. • Updated *Partial Reconfiguration* overview and added link to the*Creating a Partial Reconfiguration Design chapter of the Handbook Volume 1: Design and Compilation*. • Updated *Design Security Overview* descriptions. • Added note for Partial Reconfiguration feature and link to Partial Reconfiguration Solutions IP User Guide in *Intel Stratix 10 Configuration Overview*. • Removed SDM pin notes in *Intel Stratix 10 Configuration Overview*. • Updated internal oscillator's AS_CLK frequency in *Supported configuration clock source and AS_CLK Frequencies in Intel Stratix 10 Devices* table.
May 2017	2017.05.22	• Updated *Connection Setup for Programming the EPCQ-L Device using the AS Interface* figure. • Updated guideline to program the EPCQ-L device in *Programming EPCQ-L Devices using the Active Serial Interface*.

continued...

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