AN 370: Using the Intel FPGA Serial Flash Loader with the Intel Quartus Prime Software

Online Version
Send Feedback

AN-370

ID: 683299
Version: 2019.02.18

Contents

1. Using the Intel® FPGA Serial Flash Loader IP Core with the Intel® Quartus® Prime Software

The Intel® FPGA Serial Flash Loader IP core is an in-system programming (ISP) solution for Intel FPGA serial configuration devices. In-system programming offers you the option to program your serial configuration devices using a JTAG interface.

To use the JTAG interface to program your serial configuration device, you must create an instance of the Intel FPGA Serial Flash Loader IP core into your FPGA to form a bridge between the FPGA JTAG hard logic and the FPGA active serial memory interface (ASMI) hard logic. You can then use a download cable (Intel FPGA Parallel Port Cable, Intel FPGA Download Cable), production tester, and other tools that support JTAG to program your serial configuration device.

1.1. Features

The Intel FPGA Serial Flash Loader IP core allows you to:

- Configure your FPGA and program your serial configuration devices using the same JTAG interface.
- Correctly interpret extra padding bits introduced by third-party programmer tools to ensure successful serial configuration device programming using the **Use enhanced mode SFL** parameter.

Extended Features for Intel Arria® 10 and Intel Cyclone® 10 GX Devices

The Intel FPGA Serial Flash Loader IP core supports Intel Arria® 10 and Intel Cyclone® 10 GX devices with the following features:

- Multiple serial configuration devices—up to three cascaded identical serial configuration devices can be used to store a single configuration file.
- Multiple-die serial configuration devices—up to four stacked-die serial configuration device can be used to store a single configuration file.
- Boot page selection—specify the boot page for design multiple SRAM object file (`.sof`).

1.2. Overview

Figure 1. In-System Programming Method

This figure shows the in-system programming method using the Intel FPGA Serial Flash Loader IP core.

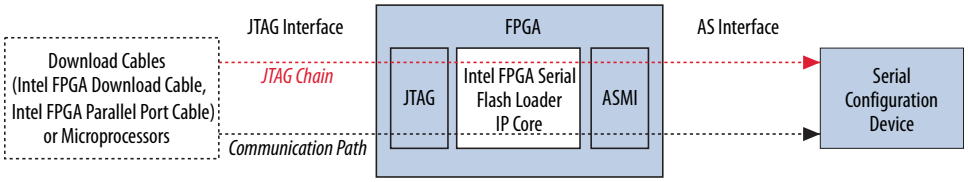


Table 1. In-System Programming Method Blocks

This table lists the blocks in the in-system programming method using the Intel FPGA Serial Flash Loader IP core.

Block	Description
JTAG	This block refers to the FPGA internal JTAG hard logic.
Intel FPGA Serial Flash Loader IP core	The IP core instantiates serial flash loader (SFL) image into your design to bridge the JTAG and ASMI interfaces. This feature allows you to perform SFL programming without resetting your design in the FPGA.
ASMI	This block is the FPGA internal ASMI hard logic.
Serial Configuration Device	This block refers to the following serial configuration devices: • EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128. • EPCQ16, EPCQ32, EPCQ64, EPCQ128, and EPCQ256. • EPCQL256, EPCQL512, and EPCQL1024. • EPCQ4A, EPCQ16A, EPCQ32A, EPCQ64A, EPCQ128A, and EPCQ512A.

Related Information

Device Configuration Support Center
Provides more information on the latest configuration devices supported by Intel. For details, refer to the ***Intel Supported Configuration Devices*** tab in the ***Device Configuration Support Center*** page.

1.3. Programming Single and Multiple Serial Configuration Devices with the Intel FPGA Serial Flash Loader IP Core

To program serial configuration devices using the Intel FPGA Serial Flash Loader IP core, set up your board in AS mode and then follow these steps:

1. To bridge the JTAG interface to the active serial interface, configure the SFL image into the FPGA.

   Note: Bypass this step if the SFL image exists in the FPGA.

2. Program the serial configuration device or devices through the JTAG-ASMI bridge of the SFL.

   Note: Programming serial configuration device uses one data line only regardless of the active serial configuration mode chosen.

3. Reconfigure your FPGA with the new configuration data you programmed into the serial configuration device in Step 2 on page 4. This replaces the SFL image with the new configuration data. To reconfigure the FPGA with the new configuration data, pull the `nConfig` pin low and then release the pin.

Note: You can include the Intel FPGA Serial Flash Loader IP core in your new configuration data to allow programming the serial configuration device when your FPGA is in user mode.

Figure 2. Programming Serial Configuration Devices with the Intel FPGA Serial Flash Loader IP Core Programming Flow

This figure shows the general programming flow to program serial configuration devices with the Intel FPGA Serial Flash Loader IP core.

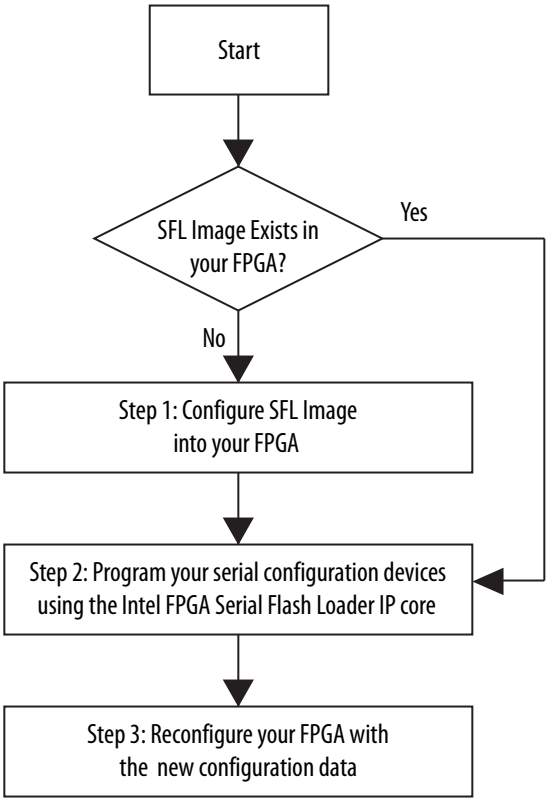


Figure 3. Programming a Single Serial Configuration Device with the Intel FPGA Serial Flash Loader IP Core Programming Flow

This figure shows the programming flow to program a single serial configuration device with the Intel FPGA Serial Flash Loader IP core.

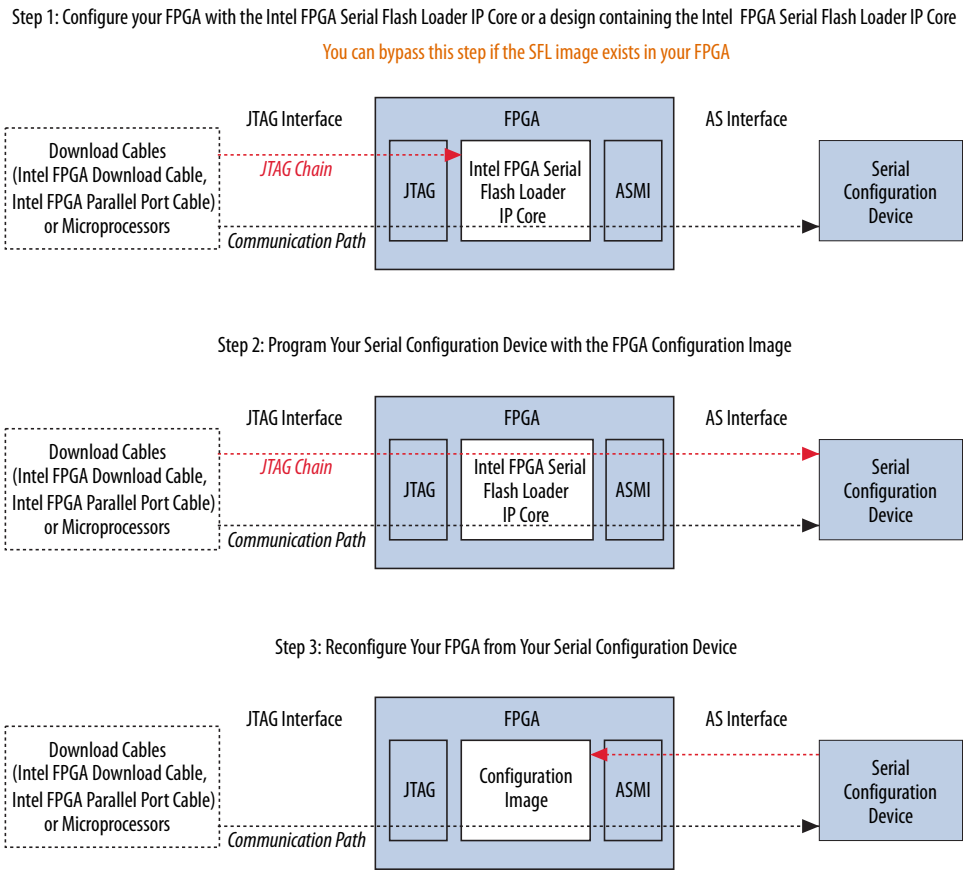
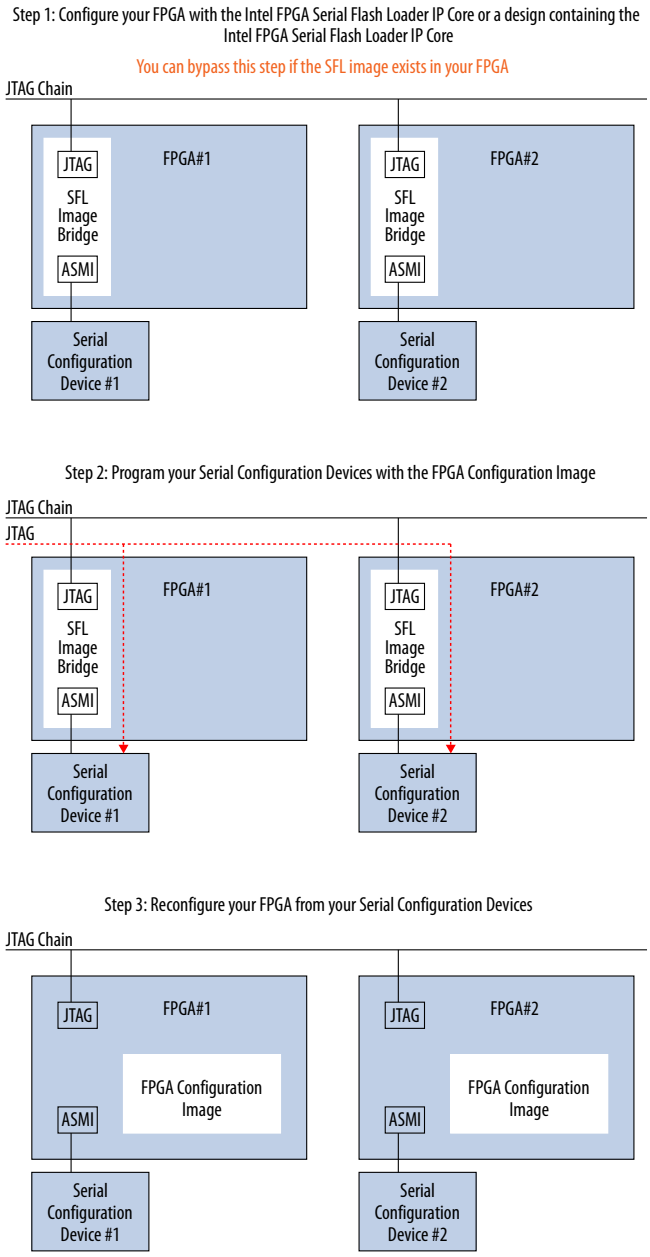


Figure 4. Programming Multiple Serial Configuration Devices with the Intel FPGA Serial Flash Loader IP Core Programming Flow

This figure shows the programming flow to program multiple serial configuration devices with the Intel FPGA Serial Flash Loader IP core.



1.4. Using the Intel FPGA Serial Flash Loader IP Core in the Intel Quartus® Prime Software

Use the Intel FPGA Serial Flash Loader IP core to create a dedicated FPGA image that you use only when programming a serial configuration device. For example, configure your FPGA with this dedicated FPGA image only when you want to program the

configuration device. After programming the serial configuration device, reconfigure your FPGA with your FPGA image that does not contain an Intel FPGA Serial Flash Loader IP core.

Alternatively, you can instantiate the Intel FPGA Serial Flash Loader IP core in your FPGA image. This allows you to program your serial configuration device at any time without interrupting your design. The internal logic can access and read and/or write the serial configuration device at any time using the **Share ASMI interface with your design** parameter in the Intel FPGA Serial Flash Loader IP core.

1.4.1. Instantiating the Intel FPGA Serial Flash Loader IP Core

You must create an instance of the Intel FPGA Serial Flash Loader IP core in your FPGA top-level design. To create the instance of the Intel FPGA Serial Flash Loader IP core, follow these steps:

1. In the IP Catalog window, search for and click **Intel FPGA Serial Flash Loader**. Click **Add**. The IP Parameter Editor appears.
2. In the **New IP Instance** dialog box, specify your top-level file name.
3. Turn on the **Share ASMI interface in your design** parameter if you must share the ASMI interface with your design. This option provides additional control pins for controlling the ASMI interface to access external serial configuration device from core logic.
4. Click **Finish** to instantiate the Intel FPGA Serial Flash Loader IP core.

   Note: The Intel FPGA Serial Flash Loader IP core does not have any timing or simulation model. Therefore, you cannot simulate the IP core.

1.5. Generating .jic and .jam Programming Files in the Intel Quartus® Prime Software

You can program serial configuration devices with either JTAG Indirect Configuration File (`.jic`), Jam File `.jam`, or Jam Byte Code `.jbc` programming files using the Intel Quartus® Prime Programmer. To generate `.jic` or `.jam` programming files with the Intel Quartus Prime software, you must first generate a user-specified SRAM object file (`.sof`), which is the input file. Next, you must convert the `.sof` to a `.jic` file using the **Convert Programming File** dialog box. Alternatively, if you prefer to use `.jam` programming files, you must convert the `.jic` file to a `.jam` file using the Intel Quartus Prime Programmer.

1.5.1. Converting .sof to .jic Files in the Intel Quartus Prime Software

To convert a **`.sof`** to a **`.jic`** file, perform the following steps:

1. On the File menu, select **Convert Programming Files**.
2. In the **Convert Programming Files** dialog box, select **JTAG Indirect Configuration File (.jic)** from the **Programming file type** drop down menu.
3. In the **Configuration device** field, specify the targeted serial configuration device.
4. In the **File name** field, browse to the target directory and specify an output file name.

5. Highlight the **SOF Data** in the Input files to convert window.
6. Click **Add File**.
7. Select the **.sof** file that you want to convert to a **.jic** file.
8. Click **OK**.
9. Highlight **Flash Loader** and click **Add Device**.
10. In the **Select Devices** dialog box, select the targeted FPGA that you are using to program the serial configuration device.
11. Click **OK**.
12. Click **Generate**.

    Note: The **Memory Map File** check box is checked by default. The Intel Quartus Prime programmer generates the memory allocation mapping file along with the **.jic** file. You can turn off this option by turning off the check box.

Related Information

Programming Serial Configuration Devices with the Intel Quartus Prime Programmer on page 10

1.5.2. Converting .jic Files to .jam Files in the Intel Quartus Prime Software

To convert a **.jic** to a **.jam** file in the Intel Quartus Prime software, perform the following steps:

1. On the Tools menu, select **Programmer**.
2. Click **Add File**. The **Select Programming File** dialog box appears.
3. Browse to the **.jic** file you created in "Converting .sof to .jic Files in the Intel Quartus Prime Software" on page 11. Add more **.jic** files if you are programming multiple serial configuration devices.
4. Click **Open**.
5. Select **Create/Update**. In the File menu, scroll to **Create JAM, SVF, or ISC File**.
6. The **Create JAM, SVF, or ISC File** dialog box appears.
7. Click **OK**.

   To program the serial configuration device or devices with the **.jam** file that you created, add the file to the Intel Quartus Prime programmer window and perform the steps in "Programming Serial Configuration Devices Using the Intel Quartus Prime Programmer and .jam Files" on page 20.

Note: With the same steps outlined above, you can generate a **.jbc** or **.svf** file from the **.jic** file.

1.6. Programming Serial Configuration Devices with the Intel Quartus Prime Programmer

You can use the Intel Quartus Prime programmer to generate serial configuration device programming files. The Intel Quartus Prime programmer can generate both **.jic** and **.jam** files with the factory default enhanced SFL image that is run directly from the Intel Quartus Prime programmer.

As long as the JTAG interface of the FPGA is accessible for programming, you can use the factory default enhanced SFL image that is run directly from the Intel Quartus Prime programmer for your application. If you enable tamper protection in design security feature, JTAG configuration is disabled. However, the serial configuration device is still accessible from JTAG interface if the FPGA is configured with encrypted configuration image that contains Intel FPGA Serial Flash Loader IP core.

1.6.1. Programming Serial Configuration Devices Using the Intel Quartus Prime Programmer and .jic Files

To program serial configuration devices with **.jic** files, you must perform the following steps:

1. When the **.sof**-to-**.jic** file conversion is complete, add the **.jic** file to the Intel Quartus Prime programmer window:
   a. In the Tools menu, choose **Programmer**. The **Chain1.cdf** dialog box appears.
   b. Click **Add File**. In the **Select Programming File** dialog box, browse to the **.jic** file.
   c. Click **Open**.
2. Configure the FPGA with the SFL image by turning on the FPGA **Program/Configure** box. This process corresponds to Step 1 on page 4. After the **Program/Configure** box is turned on, the Intel Quartus Prime programmer automatically invokes the factory default enhanced SFL image.

   Note: The **Check block CRCs or perform on-chip verification to accelerate PFL/SFL verification when available** check box under **Tools > Options > Programmer** tab is another relevant option for the enhanced mode SFL solution. This check box is turned on by default to speed up the EPCS image verification process using CRC method. The verification takes place when you turn on the **Verify** check box in the Intel Quartus Prime programmer. If you do not use this option, turn off the check box.
3. Program the serial configuration device by turning on the corresponding **Program/Configure** box, and then click **Start**. This process corresponds to Step 2 on page 4.

Note: If the **Program/Configure** check boxes are not specified, the Intel Quartus Prime programmer bypasses the request. Also, if the FPGA does not have the SFL image when the serial configuration device data is programmed through the JTAG interface, the programming process fails.

You can program multiple serial configuration devices by including more than one **.jic** file in the Intel Quartus Prime programmer.

Note: Your FPGA must be in active serial configuration mode to enable the Intel FPGA Serial Flash Loader IP core to program.

1.6.2. Programming Serial Configuration Devices Using the Intel Quartus Prime Programmer and .jam Files

When programming with **.jam** files, the Intel Quartus Prime programmer requires configuring the FPGA and programming the serial configuration device in one step.

To program serial configuration devices with **.jam** file, perform the following steps:

1. When the **.jic**-to-**.jam** file conversion is complete, add the **.jam** file to the Intel Quartus Prime programmer window:
   a. In the Tools menu, select **Programmer**. The **Chain1.cdf** dialog box appears.
   b. Click **Add File**. In the **Select Programming File** dialog box, browse to the **.jam** file.
   c. Click **Open**.
2. Configure the FPGA with the SFL image, and program the serial configuration device by turning on the FPGA **Program/Configure** check box. This process corresponds to Step 1 and Step 2 of Figure 3 on page 6.
3. Click **Start**.

   Note: The **.jam** file is generated from the **.jic** file through the chain description file (**.cdf**). For more information, refer to Intel Quartus Prime Help.

   You can program multiple serial configuration devices with one **.jam** file in the Intel Quartus Prime programmer.

1.7. Features for Intel Arria 10 and Intel Cyclone 10 GX Devices

1.7.1. Multiple Configuration Devices Support

You are unable to select the number of configuration devices. Based on the configuration device you selected in Convert Programming File tool, the Intel Quartus Prime software indicates the number of configuration devices required to fit the configuration file generated. The following examples explain the multiple flash support:

Table 2. Example of Multiple Configuration Devices Requirements

- Only identical configuration devices are supported for the multiple configuration devices feature.
- Multiple configuration device feature supports up to 3 devices only.

Configuration Device Selected	Configuration Data Size	Configuration Device Required
EPCQ-L256	Larger than 256 Mbit and smaller than 512 MBit	2
EPCQ-L256	Larger than 512 Mbit and smaller than 768 MBit	3

Note: The Intel Quartus Prime Convert Programming File tool creates a .jic file based on the setting you set. Configuration will fail if the wrong configuration device type selected in the Convert Programming File tool. However, configuration will work if the configuration devices on your board are more than to the configuration devices required by the generated configuration file.

The Intel Quartus Prime programmer sees multiple configuration devices as a big storage unit. It spans across the flash boundary automatically when the content to be stored exceeds a particular flash capacity.

For example, in Table 2 on page 11, only a single JIC file will be generated. For RPD file generation, multiple RPD files will be generated because the RPD files is programmed directly to the flash with other tools, such as third-party programmer. You must manage the RPD files and determine the right RPD to be programmed into each flash.

1.7.2. Boot Page Selection

For design with more than one SOF page, you can select to boot from any pages available. To select the boot page, click the **Option/Boot Info** button in Convert Programming File toll and select the page available from the drop down menu. By default, the page number will be set at page_0.

1.8. Intel FPGA Serial Flash Loader IP Core Parameter

Table 3. Intel FPGA Serial Flash Loader IP Core Parameter

Parameter	Value	Description
Share ASMI interface with your design	Turn On, Turn Off	Turn on the **Share ASMI interface in your design** parameter if you must share the ASMI interface with your FPGA design. This option provides additional control pins for controlling the ASMI interface to access external serial configuration device from core logic.
Use enhanced mode SFL	Turn On, Turn Off	The **Use enhanced mode SFL** parameter is enabled by default. This option provides more flexibility for JTAG cascading environment and the usage of the SFL with a third-party programmer tool. Turn off the **Use enhanced mode SFL** parameter if you do use enhanced SFL. For Arria V, Arria V GZ, Intel Arria 10, Intel Cyclone 10 GX, Cyclone V, and Stratix® V devices, you cannot disable this parameter.

1.9. Intel FPGA Serial Flash Loader IP Core Signals

Table 4. Intel FPGA Serial Flash Loader IP Core Signals

Signal	Direction	Width (Bits)	Description
`dclk_in` (1)	Input	1	Clock signal from your FPGA design to the external DCLK pin through the ASMI hard logic. The clock frequency of the `dclk_in` signal depends on your design and the flash frequency. Both inputs and output must be synchronous to `dclk_in`.
`ncso_in`(1)	Input	1	Control signal from your FPGA design to the nCS0 pin. A low signal enables the serial configuration device. *Note:* This signal is 3 bits width for Intel Arria 10 and Intel Cyclone 10 GX devices.

continued...

(1) Available for all device families when you turn on the **Share ASMI interface with your design** parameter.

Signal	Direction	Width (Bits)	Description
asdo_in [2]	Input	1	Control signal from your FPGA design to the ASDO pin for sending data into the serial configuration device.
noe_in	Input	1	Control signal to enable the Intel FPGA Serial Flash Loader IP core. A low signal enables the IP core. The IP core tri-states the ASMI interface when the IP core is disabled. If you do not access ASMI interface externally, for example, from a microprocessor, leave this signal tied to GND. This signal is available for all device families.
asmi_access_granted[1]	Input	1	Control signal to allow the Intel FPGA Serial Flash Loader IP core to access the following pins using the ASMI interface: • dclk_in • ncso_in • asdo_in • data0_out • data_in • data_oe • data_out A high signal allows the Intel Serial Flash Loader IP core to access the ASMI interface. A low signal allows your FPGA design to access the ASMI interface. Always keep this signal low. The user logic must always monitor the asmi_access_request signal. If the asmi_access_request signal is asserted, the user logic may assert the asmi_access_granted signal to allow the JTAG interface to access the Intel FPGA Serial Flash Loader IP core. The user logic must continue to drive the asmi_access_granted signal until the Intel FPGA Serial Flash Loader IP core deasserts the asmi_access_request signal. This signal is not synchronous with the dclk_in signal. *Note:* The user interface is the master and the JTAG interface is the slave.
data0_out[2]	Output	1	Signal from the DATA0 pin to your FPGA design.
asmi_access_request[1]	Output	1	A high signal indicates that the Intel FPGA Serial Flash Loader IP core is requesting ASMI interface access. The Intel FPGA Serial Flash Loader IP core starts accessing the ASMI interface when the ASMI_ACCESS_GRANTED is high. The asmi_access_request signal stays high until the Intel FPGA Serial Flash Loader IP core operation ends, such as Program/Configure, Verify, Blank Check, Examine, Erase and Auto-detect. If the asmi_access_granted signal is not asserted five seconds after the

continued...

(2) Available for Arria II, Intel Cyclone 10 LP, Cyclone IV, and Stratix IV device families when you turn on the **Share ASMI interface with your design** parameter.

Signal	Direction	Width (Bits)	Description
			asmi_access_request signal goes high, the Intel FPGA Serial Flash Loader IP core operation fails. This signal is not synchronous with the dclk_in signal.
data_in[][3]	Input	4	Control signal from your FPGA design to the AS data pin for sending data into the serial configuration device.
data_out[][3]	Output	4	Signal from the AS data pin to your FPGA design.
data_oe[][3]	Input	4	Controls data pin either as input or output because the dedicated pins for active serial data is bidirectional. To set the AS data pin as input, set the desired data pin oe to 0. To set the AS data pin as output, set the desired data pin oe to 1.

1.10. Document Revision History for AN 370: Using the Intel FPGA Serial Flash Loader IP Core with the Intel Quartus Prime Software

Document Version	Changes
2019.02.18	Added reference to the supported configuration devices in the *Overview* section.

Date	Version	Changes
December 2017	2017.12.18	• Rebranded as Intel. • Renamed the document as *Using the Intel FPGA Serial Flash Loader IP Core with the Intel Quartus Prime Software*. • Added support for Intel Cyclone 10 GX device family. • Updated Table: In-System Programming Method Blocks. • Updated the *Features for Intel Arria 10 and Intel Cyclone 10 GX Devices* topic. • Updated the *Multiple Configuration Devices Support* topic. • Made minor text edits.
May 2016	2016.05.02	• Added note to *Programming Single and Multiple Serial Configuration Devices with the Altera Serial Flash Loader IP Core* mentioning data width during configuration device programming.
July 2015	2015.07.01	• Updated figures *Programming Serial Configuration Devices with the Altera Serial Flash Loader IP Core Programming Flow* by resizing it. • Removed redundant table description in *Table 2: Altera Serial Flash Loader IP Core Parameter* and *Table 3: Altera Serial Flash Loader IP Core Signals*. • Corrected typo's and defining .jic, .jam and .jbc files.

continued...

(3) Available for Arria V, Arria V GZ, Intel Arria 10, Intel Cyclone 10 GX, Cyclone V, and Stratix V devices only when you turn on the **Share ASMI interface with your design** parameter.

Date	Version	Changes
June 2015	2015.06.15	• Updated megafunction to IP core. • Added information about setting program length count in to prevent early and late CONF_DONE assertion in 'Converting .sof to .jic Files in the Quartus II Software'. • Added Arria 10 device support in the IP core parameters and signals. • Added multiple flash and multiple stacked die flash support for Arria 10 devices.
August 2014	2014.08.18	• Updated `ncso_in` port width. • Added EPCQL256, EPCQL512, and EPCQL1024.
July 2014	2014.07.04	• IP core name changed from SFL megafunction to Altera Serial Flash Loader IP core to reflect the change in 14.0 release. • Listed IP core parameter settings. • Listed Arria V, Cyclone V, and Stratix V specific signals. • Updated figures to reflect the correct JTAG-ASMI bridge. • Included EPCQ devices information. • Updated content to reflect the IP Catalog changes. • Rewrite content to improve readability. • Updated template. • Removed outdated screen shots.
October 2012	3.2	Updated "Programming Single and Multiple Serial Configuration Devices with the SFL Solution" on page 3 to fix step error. Updated "Programming Serial Configuration Devices Using the Quartus II Programmer and .jic Files" on page 17 to fix step error. Updated template.
April 2009	3.1	• Updated the "Introduction" section • Updated the "Using the SFL Megafunction in the Quartus II Software" section • Updated Figure 5 and Figure 7 in the "Instantiating SFL Megafunction in the Quartus II Software" • Updated Figure 8, Figure 9, Figure 10, and Figure 11 in the "Converting .sof to .jic Files in the Quartus II Software" section • Updated Figure 12 and Figure 13 in the "Converting .jic Files to .jam Files in the Quartus II Software" section • Updated Figure 14 and Figure 15 in the "Programming Serial Configuration Devices Using the Quartus II Programmer and .jic Files" section • Updated Figure 17 in the"Programming Serial Configuration Devices Using the Quartus II Programmer and .jam Files" section • Updated the "Instantiating SFL Megafunction in the Quartus II Software" section • Updated Figure 6 in "Instantiating SFL Megafunction in the Quartus II Software" section • Updated Table 2 in the "Instantiating SFL Megafunction in the Quartus II Software" section • Added handnote to the "Converting .sof to .jic Files in the Quartus II Software" section • Added handnote to the "Converting .jic Files to .jam Files in the Quartus II Software" section • Updated the "Programming Serial Configuration Devices with the Quartus II Programmer" section • Updated the "Programming Serial Configuration Devices Using the Quartus II Programmer and .jic Files" section • Updated the "Programming Single and Multiple Serial Configuration Devices with the SFL Solution" section
July 2006	3.0	Updated the first paragraph in the "Introduction" section Updated the first column of Table 1

continued...

Date	Version	Changes
		Updated the forth paragraph in the "Introduction" section Updated the bulleted list in the "Introduction" section Added Note to Step 2 of the "Steps for Programming Single and Multiple Serial Configuration Devices with the SFL Solution"section Added Figure 2 to the"Steps for Programming Single and Multiple Serial Configuration Devices with the SFL Solution" section Added notes to Figure 3 and Figure 4 Added the "Using the SFL Megafunction in the Quartus II Software" section Added a note to Figure 11 and after Figure 11
June 2008	2.0	Updated the first and forth paragraph and the bulleted list in the "Introduction"section Updated column one of Table 1 Updated steps 2 and 3 in the "Steps for Programming Single and Multiple Serial Configuration Devices with the SFL Solution" section