Using the Altera Serial Flash Loader Megafunction with the Quartus II Software

2014.08.18

AN-370 Subscribe Send Feedback

The Altera Serial Flash Loader megafunction IP core is an in-system programming (ISP) solution for Altera 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 instantiate the Altera 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. This allows you to program your serial configuration device directly using the JTAG interface. You can then use a download cable (ByteBlaster II, USB-Blaster), production tester, and other tools that support JTAG to program your serial configuration device.

Note: Beginning from the Quartus II software version 14.0, the name of this IP core has been changed from Serial Flash Loader to the Altera Serial Flash Loader IP core.

Features

The Altera 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 programmer tools to ensure successful serial configuration device programming with the **Use enhanced mode SFL** parameter.

Overview

Figure 1: In-System Programming Method

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

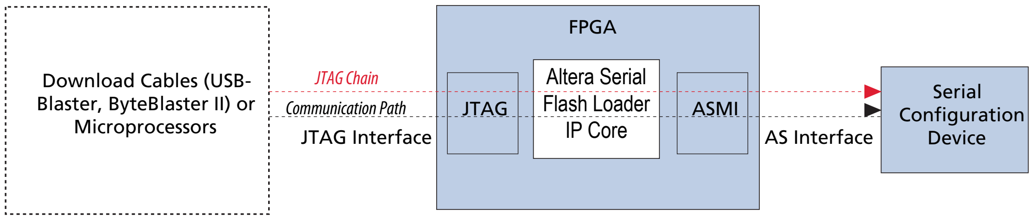

Download Cables (USB-Blaster, ByteBlaster II) or Microprocessors
JTAG Chain
Communication Path
JTAG Interface
FPGA
JTAG
Altera Serial Flash Loader IP Core
ASMI
AS Interface
Serial Configuration Device

Table 1: In-System Programming Method Blocks

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

Block	Description
JTAG	This block refers to the FPGA internal JTAG hard logic.
Altera Serial Flash Loader IP core	This block refers to the Altera Serial Flash Loader IP core. The IP core instantiate a serial flash loader (SFL) image into your design to bridge the JTAG and ASMI interface. 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, EPCQ256 and EPCQ512. • EPCQL256, EPCQL512, and EPCQL1024. **Note:** Beginning from the Quartus II software version 14.0a10, when generating the programming files or when programming EPCQL devices, you must select EPCQ devices. This is because the Programmer and the **Covert Programming Files** dialog box do not display EPCQL devices. This feature will be supported in future releases.

Programming Single and Multiple Serial Configuration Devices with the Altera Serial Flash Loader IP Core

To program serial configuration devices using the Altera 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, instantiate the Altera Serial Flash Loader IP core into your FPGA design.

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

2. Program the serial configuration device or devices via the JTAG-ASMI bridge of the SFL.
3. Reconfigure your FPGA with the new configuration data you programmed into the serial configuration device in **step 2**. 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 Altera 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 Altera Serial Flash Loader IP Core Programming Flow

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

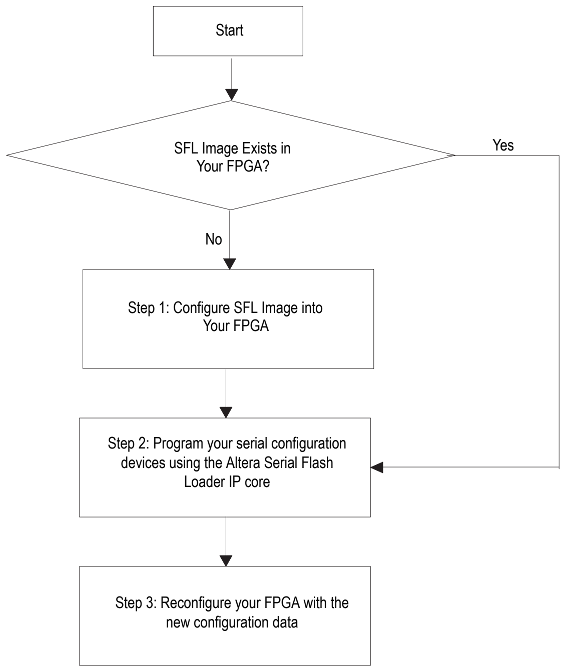

Start
SFL Image Exists in Your FPGA?
Yes
No
Step 1: Configure SFL Image into Your FPGA
Step 2: Program your serial configuration devices using the Altera Serial Flash Loader IP core
Step 3: Reconfigure your FPGA with the new configuration data

Figure 3: Programming a Single Serial Configuration Device with the Altera Serial Flash Loader IP Core Programming Flow

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

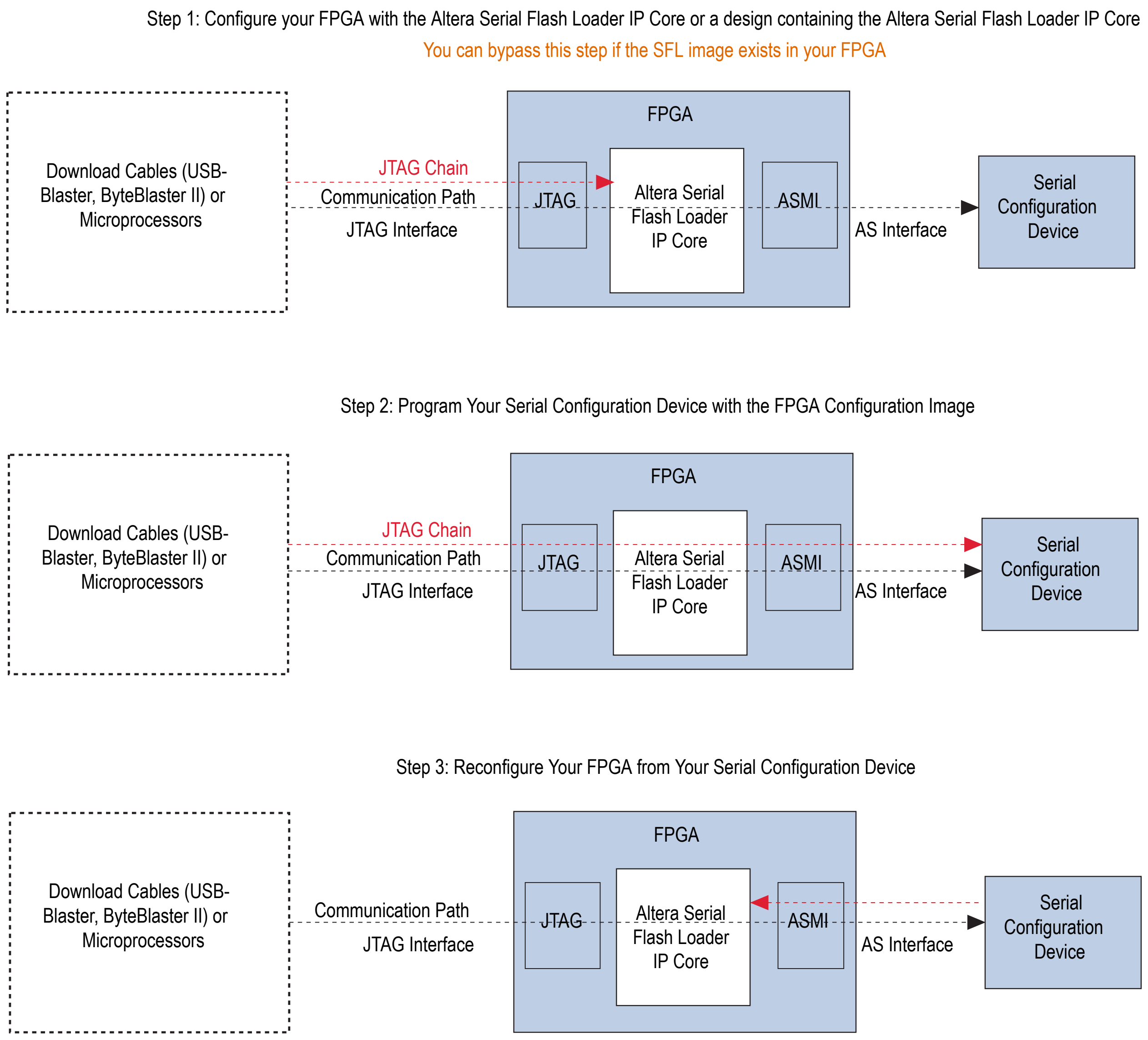
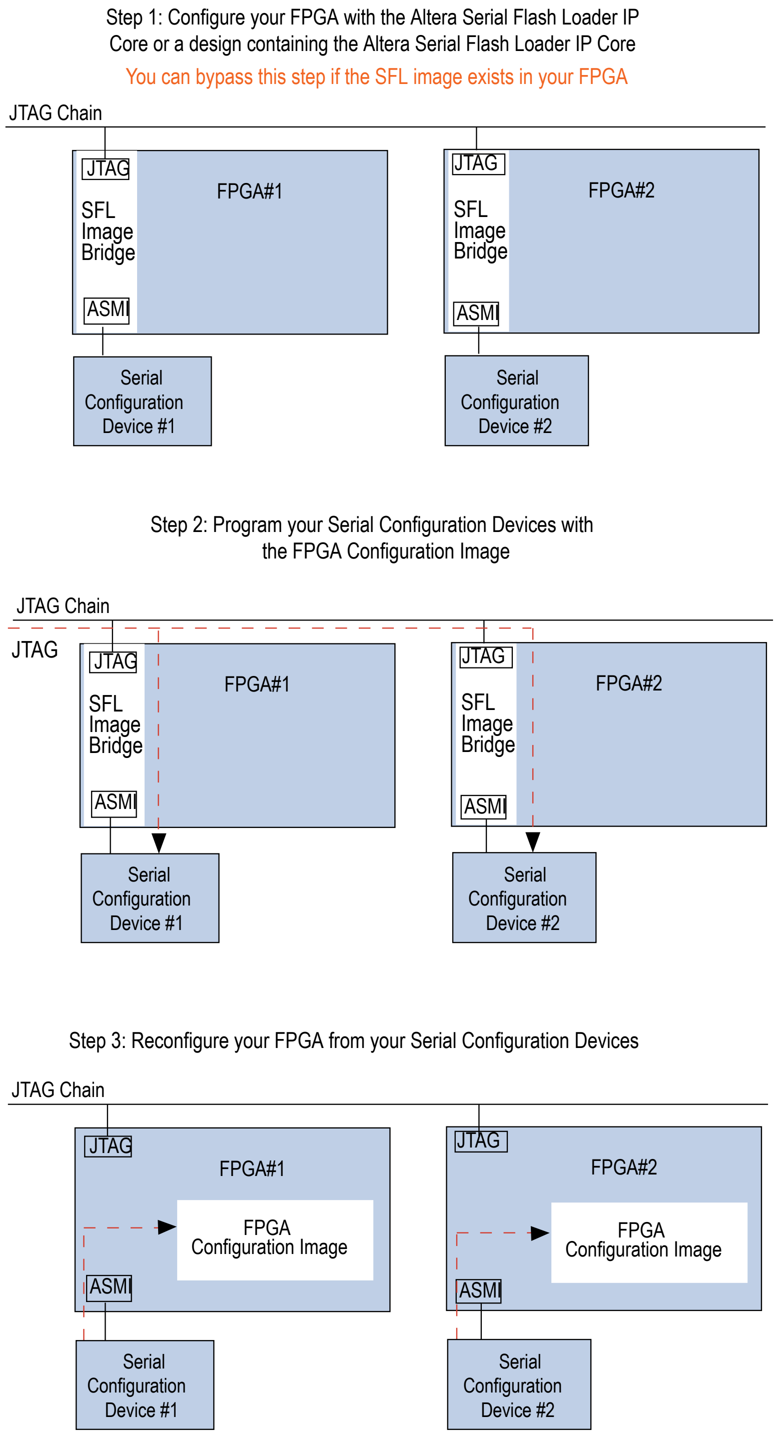

Step 1: Configure your FPGA with the Altera Serial Flash Loader IP Core or a design containing the Altera Serial Flash Loader IP Core
You can bypass this step if the SFL image exists in your FPGA
Download Cables (USB-Blaster, ByteBlaster II) or Microprocessors
JTAG Chain
Communication Path
JTAG Interface
FPGA
JTAG
Altera Serial Flash Loader IP Core
ASMI
AS Interface
Serial Configuration Device
Step 2: Program Your Serial Configuration Device with the FPGA Configuration Image
Download Cables (USB-Blaster, ByteBlaster II) or Microprocessors
JTAG Chain
Communication Path
JTAG Interface
FPGA
JTAG
Altera Serial Flash Loader IP Core
ASMI
AS Interface
Serial Configuration Device
Step 3: Reconfigure Your FPGA from Your Serial Configuration Device
Download Cables (USB-Blaster, ByteBlaster II) or Microprocessors
Communication Path
JTAG Interface
FPGA
JTAG
Altera Serial Flash Loader IP Core
ASMI
AS Interface
Serial Configuration Device

Figure 4: Programming Multiple Serial Configuration Devices with the Altera Serial Flash Loader IP Core Programming Flow

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


Step 1: Configure your FPGA with the Altera Serial Flash Loader IP Core or a design containing the Altera Serial Flash Loader IP Core
You can bypass this step if the SFL image exists in your FPGA
JTAG Chain
JTAG
SFL Image Bridge
ASMI
FPGA#1
FPGA#2
Serial Configuration Device #1
Serial Configuration Device #2
Step 2: Program your Serial Configuration Devices with the FPGA Configuration Image
JTAG Chain
JTAG
Step 3: Reconfigure your FPGA from your Serial Configuration Devices
JTAG Chain
FPGA Configuration Image

Using the Altera Serial Flash Loader IP Core in the Quartus II Software

Use the Altera 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 Altera Serial Flash Loader IP core.

Alternatively, you can instantiate the Altera Serial Flash Loader IP core in your FPGA image. This will allow 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 Altera Serial Flash Loader IP core.

Instantiating the Altera Serial Flash Loader IP Core

You must instantiate the Altera Serial Flash Loader IP core in your FPGA top-level design. To instantiate the Altera Serial Flash Loader IP core, follow these steps:

1. In the IP Catalog window, search for and click **Altera 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. Then, select your FPGA device family from the **Device Family** pull-down list. Then, select the FPGA device from the **Device** pull-down list. Click **OK**.
4. 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.
5. 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 not wish to use enhanced SFL.
6. Click **Finish** to instantiate the Altera Serial Flash Loader IP core.

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

Generating .jic and .jam Programming Files in the Quartus II Software

You can program serial configuration devices with either **.jic**, **.jam**, or **.jbc** programming files using the Quartus II Programmer. To generate **.jic** or **.jam** programming files with the Quartus II 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 Quartus II Programmer.

Converting .sof to .jic Files in the Quartus II 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 **FlashLoader** and click **Add Device**.
10. Click **OK**. The **Select Devices** dialog box appears.
11. Select the targeted FPGA that you are using to program the serial configuration device.
12. Click **OK**. The **Convert Programming Files** dialog box appears.
13. Click **Generate**.

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

To program the serial configuration device or devices with the **.jic** file that you created, add the file to the Quartus II programmer window and perform the steps in **Programming Serial Configuration Devices with the Quartus II Programmer**.

Related Information

Programming Serial Configuration Devices with the Quartus II Programmer on page 7

Converting .jic Files to .jam Files in the Quartus II Software

To convert a **.jic** to a **.jam** file in the Quartus II 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 Quartus II Software**. 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 Quartus II programmer window and perform the steps in **Programming Serial Configuration Devices Using the Quartus II Programmer and .jam Files**.

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

Programming Serial Configuration Devices with the Quartus II Programmer

You can use the Quartus II programmer to generate serial configuration device programming files. The Quartus II programmer can generate both **.jic** and **.jam** files with the factory default enhanced SFL image that is run directly from the Quartus II 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 Quartus II 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 Altera Serial Flash Loader IP Core.

Programming Serial Configuration Devices Using the Quartus II 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 Quartus II 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 2. After the **Program/Configure** box is turned on, the Quartus II programmer automatically invokes the factory default enhanced SFL image.

   Note: By default, the factory default enhanced SFL image invokes directly from the Quartus II programmer after the **Program/Configure** check box is turned on. To revert back to legacy SFL (Factory Default SFL image), turn off the **Use enhanced Serial Flash Loader (SFL) IP as factory default image** check box under **Tools** --> **Options** --> **Programmer** tab.

   Note: The **Check block CRCs 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 Quartus II programmer. If you do not wish to 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 2.

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

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

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

Programming Serial Configuration Devices Using the Quartus II Programmer and .jam Files

When programming with **.jam** files, the Quartus II 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 Quartus II 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**.
3. Click **Start**.

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

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

Altera Serial Flash Loader IP Core Parameter

Table 2: Altera Serial Flash Loader IP Core Parameter

This table lists the parameters for the Altera Serial Flash Loader IP core.

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.

Parameter	Value	Description
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 not wish to use enhanced SFL. However, for Arria V, Arria V GZ, Cyclone V, and Stratix V devices, you do not have the option to disable this parameter.

Altera Serial Flash Loader IP Core Signals

Table 3: Altera Serial Flash Loader IP Core Signals

This table lists the signals for the Altera Serial Flash Loader IP core.

Signal	Direction	Width (Bits)	Description
`dclk_in` [(1)]	Input	1	Clock signal from your FPGA design to the external `DCLK` pin via the ASMI hard logic. The clock frequency of the `dclk_in` signal is depend on your design as well as the flash frequency. Both inputs and output must be synchronous to `dclk_in`.
`ncso_in`[(1)]	Input	1 (3 bits for Arria 10 devices)	Control signal from your FPGA design to the `nCSO` pin. A low signal enables the serial configuration device.
`asdo_in` [(2)]	Input	1	Control signal from your FPGA design to the `ASDO` pin for sending data into the serial configuration device.

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

Signal	Direction	Width (Bits)	Description
`noe_in`	Input	1	Control signal to enable the Altera 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, then 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 Altera 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 Altera 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 Altera Serial Flash Loader IP core. The user logic must continue to drive the `asmi_access_granted` signal until the Altera 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.

Signal	Direction	Width (Bits)	Description
asmi_access_request[1]	Output	1	A high signal indicates that the Altera Serial Flash Loader IP core is requesting ASMI interface access. The Altera 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 Altera 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 asmi_access_request signal goes high, the Altera 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. If you want to connect your Arria V, Arria V GZ, Cyclone V or Stratix V device to the serial configuration device, then set the data_in to the following: • data_in[0] = don't care • data_in[1] = serial configuration device data to your FPGA design via AS_DATA1 pin. • data_in[2] = don't care • data_in[3] = don't care
data_out[][3]	Output	4	Signal from the AS data pin to your FPGA design. If you want to connect your Arria V, Arria V GZ, Cyclone V or Stratix V device to the serial configuration device, then set the data_out to the following: • data_out[0] = FPGA design data to serial configuration device via AS_DATA0 pin. • data_out[1] = 1'b0 • data_out[2] = 1'b1 • data_out[3] = 1'b1

[3] Available for Arria V, Arria V GZ, Cyclone V, and Stratix V devices only when you turn on the **Share ASMI interface with your design** parameter .

Signal	Direction	Width (Bits)	Description
`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`. If you want to connect your Arria V, Arria V GZ, Cyclone V or Stratix V device to the serial configuration device, then set the `data_oe` to the following: • `data_oe[0]` = 1'b1 • `data_oe[1]` = 1'b0 • `data_oe[2]` = 1'b1 • `data_oe[3]` = 1'b1

Document Revision History

Table 4: Document Revision History

Date and Revision	Changes Made	Summary of Changes
2014.08.18	• Updated `ncso_in` port width. • Added EPCQL256, EPCQL512, and EPCQL1024.	—
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 version 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.	—

Date and Revision	Changes Made	Summary of Changes
April 2009 version 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	—

Date and Revision	Changes Made	Summary of Changes
July 2006, version 3.0	Updated the first paragraph in the “Introduction” section Updated the first column of Table 1 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	Changes in response to first user feedback
June 2008, version 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	—