


## Core Overview

The common flash interface controller core with Avalon® interface (CFI controller) allows you to easily connect SOPC Builder systems to external flash memory that complies with the Common Flash Interface (CFI) specification. The CFI controller is SOPC Builder-ready and integrates easily into any SOPC Builder-generated system.

For the Nios® II processor, Altera provides hardware abstraction layer (HAL) driver routines for the CFI controller. The drivers provide universal access routines for CFI-compliant flash memories. Therefore, you do not need to write any additional code to program CFI-compliant flash devices. The HAL driver routines take advantage of the HAL generic device model for flash memory, which allows you to access the flash memory using the familiar HAL application programming interface (API), the ANSI C standard library functions for file I/O, or both.

The Nios II Embedded Design Suite (EDS) provides a flash programmer utility based on the Nios II processor and the CFI controller. The flash programmer utility can be used to program any CFI-compliant flash memory connected to an Altera® device.

 For more information about how to read and write flash using the HAL API, refer to the *Nios II Software Developer's Handbook*. For more information on the flash programmer utility, refer to the *Nios II Flash Programmer User Guide*.

Further information about the Common Flash Interface specification is available at [www.intel.com](http://www.intel.com). As an example of a flash device supported by the CFI controller, see the data sheet for the AMD Am29LV065D-120R, available at [www.amd.com](http://www.amd.com).

The common flash interface controller core supersedes previous Altera flash cores distributed with SOPC Builder or Nios development kits. All flash chips associated with these previous cores comply with the CFI specification, and therefore are supported by the CFI controller.

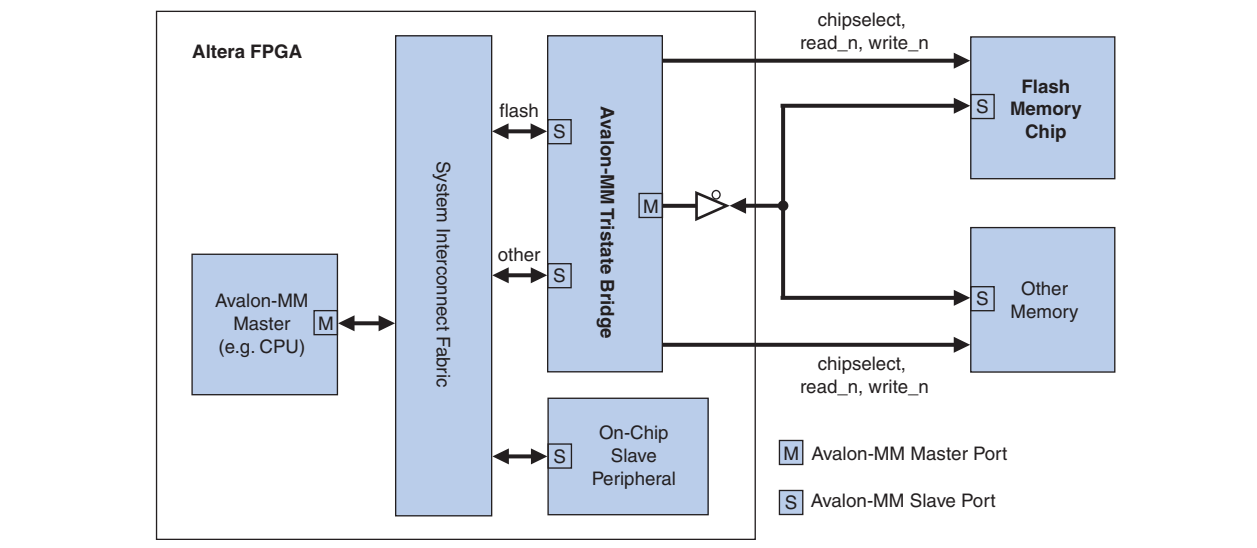
This chapter contains the following sections:

- “Functional Description” on page 3-2
- “Device and Tools Support” on page 3-2
- “Instantiating the Core in SOPC Builder” on page 3-2
- “Software Programming Model” on page 3-4

## Functional Description

Figure 3-1 shows a block diagram of the CFI controller in a typical system configuration. As shown in Figure 3-1, the Avalon Memory-Mapped (Avalon-MM) interface for flash devices is connected through an Avalon-MM tristate bridge. The tristate bridge creates an off-chip memory bus that allows the flash chip to share address and data pins with other memory chips. It provides separate chipselect, read, and write pins to each chip connected to the memory bus. The CFI controller hardware is minimal; it is simply an Avalon-MM tristate slave port configured with waitstates, setup, and hold time appropriate for the target flash chip. This slave port is capable of Avalon-MM tristate slave read and write transfers.

**Figure 3-1.** An SOPC Builder System Integrating a CFI Controller



Avalon-MM master ports can perform read transfers directly from the CFI controller's Avalon-MM port. See [“Software Programming Model”](#) on page 3-4 for more detail on writing/erasing flash memory.

## Device and Tools Support

The CFI controller supports all Altera device families. The CFI controller provides drivers for the Nios II HAL system library.

## Instantiating the Core in SOPC Builder

Hardware designers use the MegaWizard™ interface for the CFI controller in SOPC Builder to specify the core features. The following sections describe the available options.

## Attributes Page

The options on this page control the basic hardware configuration of the CFI controller.

### Presets Settings

The **Presets** setting is a drop-down menu of flash chips that have already been characterized for use with the CFI controller. After you select one of the chips in the **Presets** menu, the wizard updates all settings on both tabs (except for the Board Info setting) to work with the specified flash chip.


The options provided are not intended to cover the wide range of flash devices available in the market. If the flash chip on your target board does not appear in the **Presets** list, you must configure the other settings manually.

### Size Settings

The size setting specifies the size of the flash device. There are two settings:


- **Address Width**—The width of the flash chip's address bus.
- **Data Width**—The width of the flash chip's data bus

The size settings cause SOPC Builder to allocate the correct amount of address space for this device. SOPC Builder will automatically generate dynamic bus sizing logic that appropriately connects the flash chip to Avalon-MM master ports of different data widths.

 For details about dynamic bus sizing, refer to the [Avalon Interface Specifications](#).

## Timing Page

The options on this page specify the timing requirements for read and write transfers with the flash device.

 Refer to the specifications provided with the common flash device you are using to obtain the timing values you need to calculate the values of the parameters on the **Timing** page.

The settings available on the **Timing** page are:

- **Setup**—After asserting `chipselct`, the time required before asserting the `read` or `write` signals. You can determine the value of this parameter by using the following formula:


$$\text{Setup} = t_{\text{CE}} (\text{chip enable to output delay}) - t_{\text{OE}} (\text{output enable to output delay})$$

- **Wait**—The time required for the `read` or `write` signals to be asserted for each transfer. Use the following guideline to determine an appropriate value for this parameter:

The sum of **Setup**, **Wait**, and board delay must be greater than  $t_{\text{ACC}}$ , where:

- Board delay is determined by the  $T_{\text{CO}}$  on the device address pins,  $T_{\text{SU}}$  on the device data pins and propagation delay on the board traces in both directions.
- $t_{\text{ACC}}$  is the address to output delay.

- **Hold**—After deasserting the `write` signal, the time required before deasserting the `chipselct` signal.
- **Units**—The timing units used for the **Setup**, **Wait**, and **Hold** values. Possible values include ns,  $\mu$ s, ms, and clock cycles.


 For more information about signal timing for the Avalon-MM interface, refer to the [Avalon Interface Specifications](#).

## Software Programming Model

This section describes the software programming model for the CFI controller. In general, any Avalon-MM master in the system can read the flash chip directly as a memory device. For Nios II processor users, Altera provides HAL system library drivers that enable you to erase and write the flash memory using the HAL API functions.

### HAL System Library Support

The Altera-provided driver implements a HAL flash device driver that integrates into the HAL system library for Nios II systems. Programs call the familiar HAL API functions to program CFI-compliant flash memory. You do not need to know anything about the details of the underlying drivers.

 The HAL API for programming flash, including C code examples, is described in detail in the [Nios II Software Developer's Handbook](#). The Nios II EDS also provides a reference design called Flash Tests that demonstrates erasing, writing, and reading flash memory.

#### Limitations

Currently, the Altera-provided drivers for the CFI controller support only Intel, AMD and Spansion flash chips.

### Software Files

The CFI controller provides the following software files. These files define the low-level access to the hardware, and provide the routines for the HAL flash device driver. Application developers should not modify these files.

- **altera\_avalon\_cfi\_flash.h, altera\_avalon\_cfi\_flash.c**—The header and source code for the functions and variables required to integrate the driver into the HAL system library.
- **altera\_avalon\_cfi\_flash\_funcs.h, altera\_avalon\_cfi\_flash\_table.c**—The header and source code for functions concerned with accessing the CFI table.
- **altera\_avalon\_cfi\_flash\_amd\_funcs.h, altera\_avalon\_cfi\_flash\_amd.c**—The header and source code for programming AMD CFI-compliant flash chips.
- **altera\_avalon\_cfi\_flash\_intel\_funcs.h, altera\_avalon\_cfi\_flash\_intel.c**—The header and source code for programming Intel CFI-compliant flash chips.

## Referenced Documents

This chapter references the following documents:


- [Avalon Interface Specifications](#)
- [Nios II Flash Programmer User Guide](#)
- [Nios II Software Developer's Handbook](#)

## Document Revision History

Table 3–1 shows the revision history for this chapter.

**Table 3–1.** Document Revision History

Date and Document Version	Changes Made	Summary of Changes
November 2009 v9.1.0	Revised description of the timing page settings.	
March 2009 v9.0.0	No change from previous release.	—
November 2008 v8.1.0	Changed to 8-1/2 x 11 page size. Added description to parameters on <b>Timing</b> page.	—
May 2008 v8.0.0	Updated the CFI controllers supported by Altera-provided drivers.	Updates made to comply with the Quartus II software version 8.0 release.

 For previous versions of the *Quartus II Handbook*, refer to the [Quartus II Handbook Archive](#).

