Porting the Nios GERMS Monitor to Work with Different Flash Memories

September 2003 - 1.0 Application Note 322

Introduction

The Nios® development kit is supplied with an example boot monitor program called GERMS. Among other things, the GERMS monitor allows easy erasure and programming of flash devices on any of the Altera Nios development boards. The GERMS monitor provides flash drivers that are specific to the flash device on the Nios development board, although these drivers may work for other flash devices in the same family. When creating a custom system, a user may want to use a different flash device not supported by the GERMS monitor. This document describes the process to create a new SOPC Builder flash component and to port the GERMS monitor for an arbitrary flash device.

Requirements

You will need the data sheet describing the hardware interface and the erase/write routines for the target flash device. In addition, you should be familiar with:

- Programming C code
- The Quartus® II software
- The SOPC Builder tool and its Interface to User Logic wizard

Hardware Design

The first stage of the component creation process is to generate the hardware interface to the flash memory device. To do this, perform the following steps:

1. Launch the Quartus II software and create a new project.

   or

   Open an existing project.

2. Launch SOPC Builder.

3. Select **Interface To User Logic** from the pool of available modules in the left-hand pane of the SOPC Builder System Contents page.

4. Click **Add**. The **Interface to User Logic** dialog box opens.

5. For the **Bus Interface Type,** select Avalon Memory Slave.

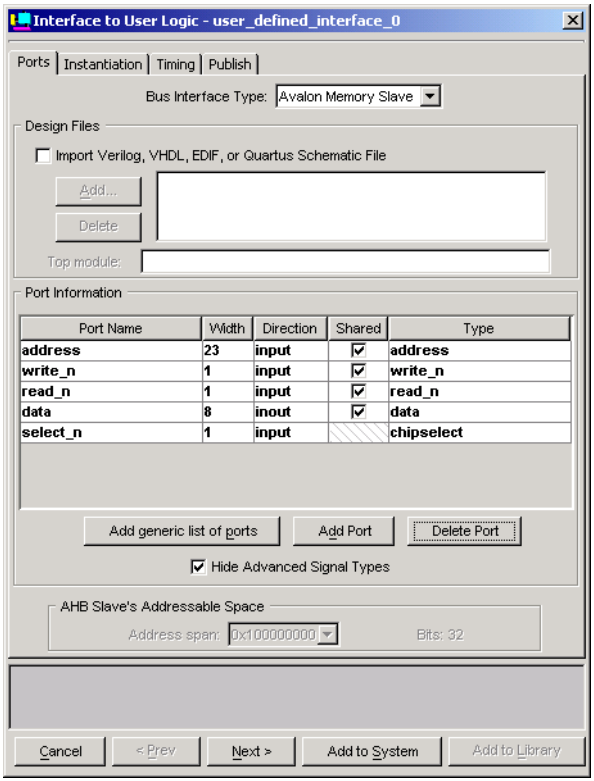
6. In the Design Files section, ensure that the **Import** check box is not turned on.

7. Use the **Add Port** and **Delete Port** buttons to configure the hardware interface to the flash device in accordance with the flash device data sheet and the system requirements (see Figure 1).

   A typical flash device will require the following ports:

   - `address` (input bus)
   - `data` (input bus)
   - `read` or `read_n` (input)
   - `write` or `write_n` (input)
   - `chipselect` or `chipselect_n` (input)

Figure 1. Specifying Port Information in the Interface to User Logic Dialog Box

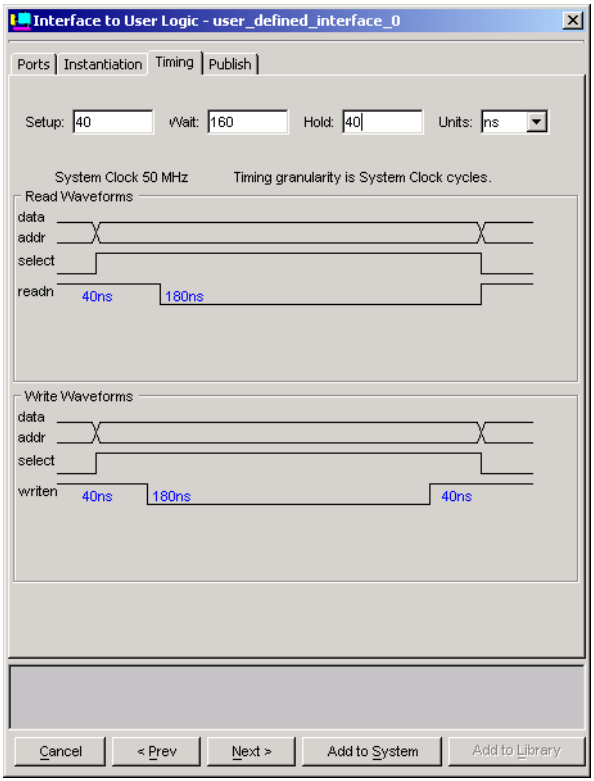


8. When completed, click **Next**. The **Instantiation** tab displays and shows just one instantiation option, **Export bus ports**.

9. Click **Next** to accept the **Export bus ports** setting. The **Timing** tab displays.

10. Enter the timing requirements for the flash device in accordance with the flash device data sheet. Specify the timing requirements in ns rather than cycles, because SOPC Builder then automatically adds the correct number of wait states for the device regardless of system clock frequency (see Figure 2).

Figure 2. Specifying Timing in the Interface to User Logic Dialog Box



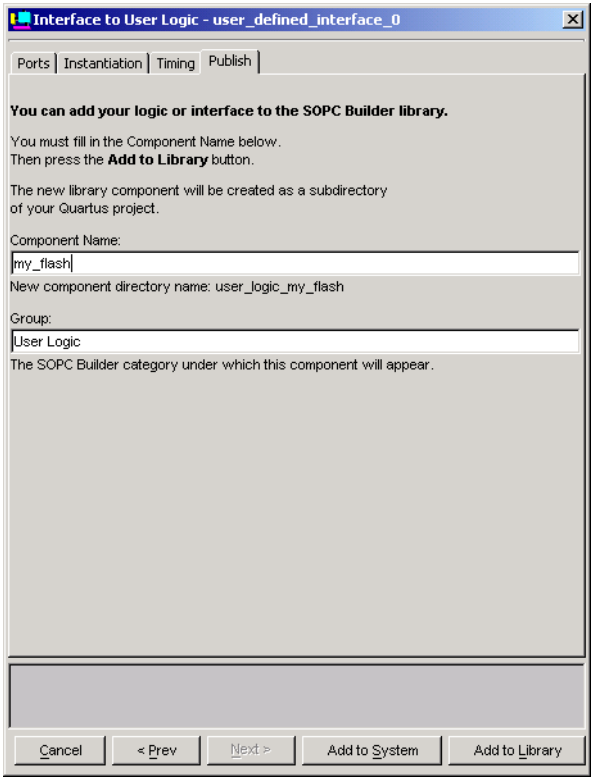
The timing specified in this wizard is relative to signals internal to the Nios system module on the FPGA. It is important to account for additional time for T_{CO}, T_{SU}, and T_H incurred by routing to the FPGA I/O pins. These values are design dependant and should be confirmed with the Quartus II timing analyzer.

11. Click **Next** when the timing requirements are complete. The **Publish** tab displays.

    The flash component can now be published so that it appears in the pool of available modules in the SOPC Builder System Contents page.

12. Enter a name for the new component ensuring that it ends in `_flash` (see Figure 3). SOPC Builder recognizes the `_flash` suffix to identify a component as a flash device. When accessing `_flash` components, GERMS uses the appropriate flash driver routines rather than using standard memory read and write operations.

Figure 3. Publishing a Component with the Interface to User Logic Dialog Box



You must use underscores rather than spaces in the name of the component. Failure to do so will result in a system generation error when creating a SOPC system containing this component.

13. Click **Add to Library**. This publishes the component and creates a new folder with the prefix `user_logic_` in the Quartus working directory. The contents of this folder describe the newly-created component.

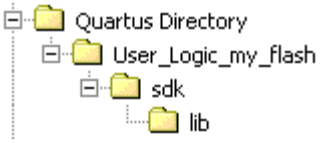
After the component is published, it will appear in the pool of available modules in the System Contents page of SOPC Builder. You can add your new interface to flash as easily as adding any Altera-provided component to a system.

Software Design

In order for the GERMS monitor to correctly erase and write to flash, it may be necessary to re-write the flash drivers for your custom system. This can be done by modifying the existing drivers, or by creating a new file. This section describes the process of modifying the existing drivers. To modify the existing drivers, perform the following steps:

1. Create a directory structure within the published hardware component directory as described in Step 13 above and shown in Figure 4.

Figure 4. SDK Directory Structure for the Custom Flash Component



2. Copy the file **flash_AMD29LV065d.c** found in the **<*Nios development kit installation*>\sdk\lib** directory and place it the **lib** directory created in Step 1 above.

3. Rename the file to identify your specific flash device driver, and then edit the file and modify the drivers to comply with the target flash device's data sheet.

Not all of the routines in this file are referenced by the GERMS monitor. If the driver is only to be used by the GERMS monitor, then you only need to modify the following routines and any private routines that they call:

- `nr_flash_erase_sector`
- `nr_flash_write`

In their current implementations, these routines call the following subroutines within the same driver file:

- `await_erase_complete`
- `wait_at_least_1_us`
- `await_write_complete`
- `amd29lv065d_flash_write_byte`

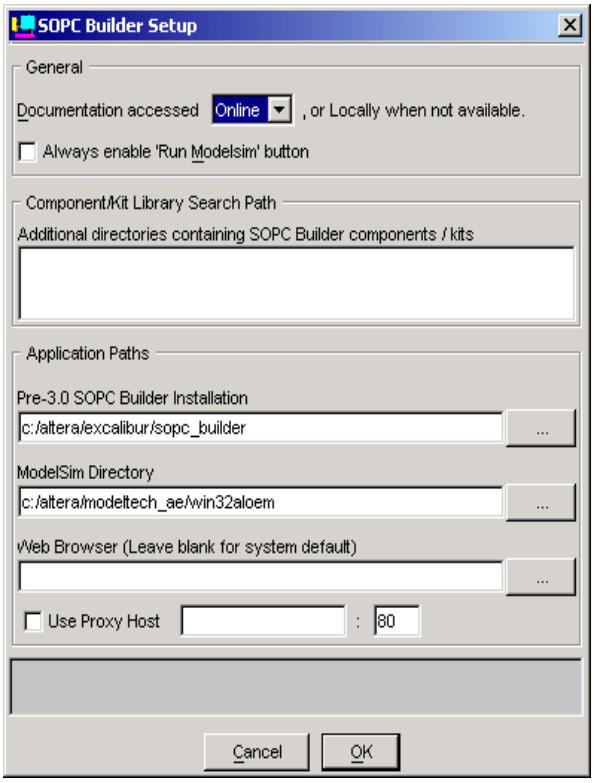
Using the New Flash Component

Once you have completed and verified the porting effort, you can make the custom flash component available to all members on your development team. For example, you can place the new component directory on a network drive to which all Nios developers have access.

Developers can access this component from SOPC Builder by specifying the path to it's parent directory. To specify this path, perform the following steps from within SOPC Builder:

1. Select **SOPC Builder Setup** (File menu).
2. Specify the path in the **Additional directories containing SOPC Builder components/kits** field (see Figure 5 on page 7).

Figure 5. Specifying a Network Path to SOPC Builder Components



Testing

The methodology used in this document was tested by implementing a custom interface to the AMD 29LV065d device on the Nios Development Board, Stratix Edition. A new user interface to the flash device was created as described in this document. The driver routines were copied to the new component directory and unused routines were deleted. Hardware testing was then accomplished by downloading software to flash and running it from flash.

References

The following documents provide background information on concepts related to the Nios embedded processor and SOPC Builder:

- *SOPC Builder User Guide*
- *Nios Hardware Development Tutorial*

- *Nios Software Development Tutorial*
- *Nios Software Developer's Reference Manual*

Revision History

First publication in September 2003.

101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
www.altera.com
Applications Hotline:
(800) 800-EPLD
Literature Services:
lit_req@altera.com