Flash Programming the Altera DE2 Board

1.) Add the necessary components to your system.

Open the SOPC builder and add an Avalon-MM Tristate Bridge (under Bridges and Adaptors → Memory Mapped) and a CFI Flash Memory Interface (under Memories and Memory Controllers → Flash). Use default settings for the tristate bridge. In the flash memory interface settings, set “presets” to “custom,” “address width” to 22 bits, and “data width” to 8 bits. Then look under the Timing tab and set the setup, wait, and hold times. Recommended times are setup = 40, wait = 160, and hold = 40. (These depend on the model of flash memory chip installed on the Altera board. Refer to the flash chip’s datasheet for more information.) Adding the System ID component to your system is also recommended, though not necessary.

In the main SOPC builder window, click the “Connection” dots to the left of the tristate_master (under your tristate_bridge component) and cfi_flash components to connect them.

Now you must set up the Nios II processor to run your program from flash. Select your processor component and click “Edit” to open its settings window. Put your flash memory component in the Reset Vector: Memory: drop box. Leave the offset at 0. The Digikey tutorial will tell you to put the exception vector in flash memory as well (with an offset), but DON’T DO IT! Leave the exception vector in RAM.

Reassign base addresses and IRQs, then regenerate the system.

2.) Make appropriate connections in your VHDL code.

Putting the tristate bridge and CFI flash memory in your system will add the following signals to your processor’s component declaration (assuming your flash memory is called cfi_flash_0):

```
signal address_to_the_cfi_flash_0 : OUT STD_LOGIC_VECTOR (21 DOWNTO 0);
signal data_to_and_from_the_cfi_flash_0 : INOUT STD_LOGIC_VECTOR (7 DOWNTO 0);
signal read_n_to_the_cfi_flash_0 : OUT STD_LOGIC;
signal select_n_to_the_cfi_flash_0 : OUT STD_LOGIC;
signal write_n_to_the_cfi_flash_0 : OUT STD_LOGIC;
```

Make sure you add these to your DE2_board_top_level VHDL file. In your component’s instantiation, connect the signals as follows:

```
address_to_the_cfi_flash_0 => FL_ADDR,
data_to_and_from_the_cfi_flash_0 => FL_DQ,
read_n_to_the_cfi_flash_0 => FL_OE_N,
```

select_n_to_the_cfi_flash_0 => FL_CE_N,
write_n_to_the_cfi_flash_0 => FL_WE_N,

where the pin names given are the standard names which can be found in the default assignments file.

Scroll down to the bottom of the VHDL file and comment out any default assignments which are driving the flash memory chip's pins. Pull FL_RST_N high (i.e. assign a '1' to it).

Recompile your system in Quartus II.

3.) Store the FPGA configuration on the DE2 board using Active Serial mode programming.

Before you attempt this step, make sure the copy of Quartus II you are using is appropriately patched. Quartus 9.1 has a bug which prevents it from properly generating some of the files needed for active serial programming. If your copy is not patched, you should see a green button in the lower right corner of Quartus II's main window, encouraging you to download Service Pack 1. Click the button and follow the instructions to download and install the update. If you have an older version of Quartus (e.g. 9.0), you should not need a patch.

Once your software is patched, launch the Programmer tool. Change the programming mode from JTAG to AS (active serial). Quartus will ask if you want to discard the current programming options; click yes. Then click the "Add File" button on the left side of the Programmer, and navigate to your project folder. There should be only one compatible file in the folder; its extension will be ".pof." Selecting this file will add it to the list in the center of the Programmer window. Put a check in the Program/Configure checkbox.

Now turn on your DE2 board and look at the left side of the board, next to the LCD screen. There is a small sliding switch that can be set on "Run" or "Prog." Slide the switch to the "Prog" position. Then click the "Start" button on the Programmer. When the Programmer finishes its operation, slide the switch back to "Run." If the operation was successful, any simple features that you implemented in VHDL should now work. They should continue to work if you cycle power to the board. Connecting the switches to the red LEDs in your VHDL code is a good way to test at this stage (although you will have to give up the use of any PIO component you had connected to the LEDs).

4). Load your C code into flash memory.

Open the NIOS II IDE and load your project. Open the System Library Settings window. On the right side of the window, you will see various segments of your code listed, with a drop box to the right of each, indicating where it should be located in memory. Put the program memory (.text) and the read-only data memory (.rodata) in flash by selecting your flash component from the list in the drop boxes. Leave the others (.rwdata, heap, and stack) in RAM. Click OK to close the window.

Now launch the Flash Programmer (Tools → Flash Programmer). Create a new launch configuration by clicking the "new" button in the upper left corner of the window. Leave all settings and checkboxes in their default state. Click the "Program Flash" button.

Now you're done! If the programming operation was successful, you should see the following (or something similar) in the console window:

```
Programmed 62KB +2KB in 1.3s (49.2KB/s)
Device contents checksummed OK
Leaving target processor paused
```

The next time you cycle power to the board, your program should boot up and begin running within a few seconds.

Common Errors and Problems:
You may see the following errors when using the flash programmer:

"No CFI table found at address flash_base_address":
Your flash memory component is probably not connected properly. Review part 2 of this tutorial and make sure you've done everything correctly.

"make: ***[cfi_flash_0.flash] Error 5":
You've put your processor's reset vector in flash memory, but forgot to edit your C project's system library to put your .text and .rodata in the flash.

"Empty flash content cannot be programmed or verified"
You haven't put your processor's reset vector in the flash.

"Boot copier overlaps data in flash"
You may get this error if you put the processor's exception vector in the flash. Move the exception vector back to RAM.

"There are no NIOS II processors available which match the values specified."
You may get this error if 1) your DE2 board is not programmed with the system that your C code is designed for, 2) you forgot to flip the switch back to "Run" after programming the board, or 3) you have an incomplete or bad configuration build, due to the active serial bug described in part 3 of this tutorial.