University of Massachusetts Amherst
Computer Systems Lab 2 (ECE 354)
Spring 2007

Lab 1: Using Nios 2 processor for code execution on FPGA

Objectives: After the completion of this lab:

1. You will understand the typical design flow in an Embedded System design as illustrated by the Altera CAD tools: Quartus 2, SOPC, Nios IDE.
2. You will begin to appreciate the differences between traditional microcontrollers and soft core processors.
3. You will learn how to use on chip memory and SDRAM in your hardware designs
4. Gain more insight into Quartus 2, learn to edit BDF files, assigning pins of FPGA etc.

This lab consists of two parts. First you will echo "hello from Nios 2" on your PC screen in the Nios 2 IDE console window. Next, you will implement a counter program and display the value of count on LCD, LEDs or the 7 segment Hexadecimal Displays on DE2.

These labs will primarily teach you the use of System on Programmable Chip Development tool (SOPC). There is no programming in Verilog or C involved at this stage, which we will leave for the labs to follow :).

Before starting this lab, it is **mandatory** that you read the following document which describes a typical Quartus 2 design. Use this document to refresh your understanding in: Creating a new project, compiling it, assigning pins to FPGA and programming the FPGA.

The document is located at:

http://www.altera.com/education/univ/materials/manual/labs/tut_quartus_intro_verilog.pdf

After you have read this document, you can proceed with the lab.

Part 1: Hello from Nios 2.

1. Create a new project in Quartus 2, taking care to select the proper FPGA (Which FPGA are we using ?). Do not add any files at this stage to project, we will do it later.
2. Go to Tools -> SOPC builder. You will see a new tool (SOPC) with a pop up window in the center of your screen asking you to enter the system name. Although you can specify any name, I advise you to use the name of your project which you created in Quartus 2. This is the safest way and avoids any discrepancies in your top level settings later. Check verilog under the target HDL option. SOPC can generate files in either Verilog or VHDL depending on this option and does not favor any language. Press OK.
3. Maximize the SOPC window and observe the workspace. You will see settings about the board on top under the "Target" option. Change it to "unspecified board" and device family to Cyclone 2. Leave the clock settings unchanged, which reads 50 Mhz.
4. On the left hand side of screen, you will find a number of hardware components present under the "System Contents" tab. These components are available to you for use. Double clicking on any individual component adds it to your hardware. We will be adding Nios 2 processor, on chip memory, JTAG UART and a timer to make our hardware.
5. Under Avalon Components, double click on Nios 2 processor. In the window which opens up, you will see three different kinds of Nios 2. These processors vary from smallest to largest and present you a size vs performance tradeoff. Select the center Nios 2/S processor which is a good combination of speed and small size. Click on finish to add the processor to your hardware.
6. You would see that your workspace now contains Nios 2. Observe that the SOPC automatically assigned a base address to it. Also, you will see that IRQ's are also determined automatically by the SOPC. Dont worry about the error messages which you see on the bottom of the screen, they will go away after we finish adding all the components.
7. Next, we will add some on chip memory to store our program. This memory will be instantiated on the FPGA. To do this, select On- Chip Memory (RAM or ROM) present under the Memory components. You will first have to expand the memory tab to see what other components are present under it. Use 20 KB of on chip memory. Leave the other settings unaltered and click on finish.
8. Next we will add a JTAG UART which reduces the number of pins needed to talk to FPGA by the PC. This is present under the communications components. Click on finish in the pop up box and leave all the settings unaltered.
9. The Avalon switch fabric which glues all these components together, needs an internal clock to work. To provide this, double click on the interval timer present under the "other" components. Use the default settings and click on finish.

10. SOPC assigns IRQ's and base addresses automatically which is very convenient. However you will have to change the priority of the interrupts manually to make this system work properly. By default, the JTAG UART is assigned a higher priority (lower IRQ means higher preference) than the timer_0. As the Avalon Bus is constantly under use while the JTAG UART is needed only to configure the FPGA, the timer_0 should be assigned a higher priority. To do this, click on the rectangular box under the IRQ column in the row of timer_0. Change the IRQ to 0, which is the highest. Similarly change the IRQ of JTAG UART to 16.
11. You are now ready to generate the system, click on the “generate” button on the bottom of the screen. This operation will take a few seconds to complete.
12. After the system is generated, go back to quartus 2 where your project is still open. Do not close the SOPC builder window.
13. We will now work in Quartus 2 for some time. The SOPC generated the files in Verilog for our hardware system (we specified Verilog as the target HDL in the beginning) which we need to add to our project. In Quartus 2 go to project-> Add/Remove files in project.. option. In the dialog box which opens, click on add all and then click on OK.
14. Click on “Start Analysis and Synthesis” button present on the task bar where you also have the buttons for “Start Compilation”, “Settings”, “Assignment Editor” etc.
15. If you get an error message which says that your top level entity is undefined, this is due to the fact that you gave a different name for the system in SOPC. To fix this, click on settings button on the task bar on top and go to “general”. Change the top level entity to whatever name you gave for your system in SOPC. Repeat step 14 which now completes without any problem.
16. Next, you need to assign pins to the FPGA. Use the assignment editor and pin details from DE 2 manual for doing this. You must again read the document from Altera which explains it in detail (The link is given at the beginning of this document). Assign any toggle switch to the reset_n pin and provide the 50 Mhz clock to clk pin.
17. Now do a full compilation by clicking the “Start Compilation” button on the task bar.
18. Program the FPGA with the sof file contained in our project directory. To do this, click on the “Programmer” button on the taskbar. A pop up tells you about the time limited feature of our design. Click on OK. Next, check the program/configure option in the programmer window and click on start. Make sure that you have USB blaster installed and you are using JTAG mode. DO NOT click on the cancel button in the “time remaining” pop up window which appears after programming is successful.
19. After programming the FPGA, go to the SOPC builder again and click on the “System Generation” tab on top. Click on the “Run Nios 2 IDE” button.

20. You will see that Nios 2 IDE is launched and you are asked to specify a workspace. You can use this as your default workspace or specify another folder depending on your convenience.
21. The Nios 2 IDE presents you with a blue welcome screen. Close this screen by clicking on the close button (looks like X) located on top left hand corner, near the word “welcome”.
22. Go to file->new-> C/C++ application to create a new program which we will run on our Nios.
23. The “New Project” window presents you with a number of options. You can select any template from the ones present or create a new one. We will be using the default Hello World template which is already selected. This program in C has already been written and we will be running it on our board. We also need to specify the SOPC builder system which is a ptf file the Nios 2 toolchain uses to determine the hardware present. If you follow the steps in strict order and open the Nios 2 IDE from SOPC, you should have the SOPC builder system already specified as <your SOPC system name>.ptf file. Otherwise you can browse to your project directory and specify the SOPC builder system. Click on finish to complete the project settings.
24. Right click on hello_world_0 under C/C++ projects and select the “System Library Properties” option from the menu which opens. Uncheck “Clean exit(flush buffers)” and check “Small C library” options. Click on OK. This is done to limit the size of executable file as we have only 20 KB of memory with us.
25. Go to run-> run as->Nios 2 hardware. Make sure that your reset switch is logic HIGH (Reset_n is active when it is low) otherwise your processor will be reset all the time.
26. You should see “Hello from Nios 2 “ on your screen after sometime if everything is properly done :).

To make your life simpler, you must note the following:

1. Keep the project name in Quartus 2 and the system name in SOPC same.
2. You can add the components in SOPC in any order, it does not matter.
3. Take particular notice of the IRQ's and change the preference to Timer over JTAG UART.
4. The pins must be assigned properly. Note what is logic High and Low for switches. Refer to DE 2 manual
5. Specify the correct SOPC builder system and project template in Nios 2 IDE.
6. If you dont see things working out, it is a good idea to switch the workspace.

After completing this part of the first lab, you are ready to design a more complex system.

Part 2: Implementing a counter and directing its output to LEDs, LCD or Seven Segment Displays.

In this lab, you have to implement a counter using a C program called "count binary" which is available in the Nios 2 IDE as a template(like Hello world in previous part). The primary task is to create a SOPC builder system which can support this program, which includes SDRAM. This program generates a counter capable of counting from 0x00 to 0xFF on Nios 2. The user can see the count value either on LCD, Two Seven Segment Displays or the 8 green LEDs or on all of them simultaneously. The selection of display is made by pressing on of the four blue keys which you see on DE2 board according to the following rule:

Key 0 pressed: LEDs display the count value in binary.

Key 1 pressed: Two Seven Segment Displays show the count value in decimal.

Key 2 pressed: LCD shows the count value in octal.

Key 3 pressed: All the above show count value.

To complete this lab, you are given some general guidelines and I assume that you have completed part 1 successfully. Before starting this part of the lab, try to understand some sections of the count binary program which is given as a template in Nios 2. Pay particular attention to the way the input is accepted by the switches. How is the interrupt generated ?

1. In your hardware using SOPC builder, include the following components: Nios 2, JTAG UART, Interval Timer as in part 1.
2. Since we will be using external off chip SDRAM, we need to include the SDRAM controller present under the memory components. Change the data width to 16 bits and leave all other values unchanged.
3. To use the LCD, include the character LCD present under Display components.
4. LEDs, Hexadecimal Seven Segment Displays and Keys are examples of parallel input output hardware, which can be found under "other"

components. You will have to include parallel I/O thrice, one for LED, one for the Keys and another for the Seven Segment Display. While adding each parallel I/O, you can specify their width and whether they are input, output or both. There are other options for input type I/Os which let you control their triggering nature (edge vs level) and let you decide their interrupt generation. Think of how many LEDs you need to display the binary count value. You will have to instantiate the same number of LEDs, one for each bit. Hence the width of parallel I/O should be equal to the number of LEDs needed, one output for each bit. These parallel I/O will be connected to the actual LEDs by pin assignments later. In a similar manner, include another set of 4 parallel I/Os for keys which the user will press. Lastly, include support for Seven Segment Display by using a third set of parallel I/Os. You have to figure out the width in this case by reading the DE 2 manual which describes the 7 Segment Displays. Note that you will need two seven segment displays to show the count value. While using these I/Os, you may want to read the program "count binary" more carefully to determine any special needs they might have.

5. Read the requirements given in the count binary program for the hardware. It expects the inputs and outputs to be named in a certain way. Rename the components in your hardware by right clicking them (after you have included them) and selecting rename.
6. Lastly, include a PLL to send a clock to the SDRAM on the board. The idea is to have some pin of FPGA driving the SDRAM clock input. This would ensure that Nios 2 and SDRAM work at the same clock frequencies. Thus the PLL needs to be transparent for most of its design and simply act as means to generate a clock for SDRAM by accepting the global clock which is also being fed to Nios. The clock which is sent to PLL and the clock which comes out, must have the same frequency. You can find PLL under "other" components. To avoid any warning messages later on, make sure that the PLL operation mode is set to "Zero delay buffer". You can find this option under Operation mode settings. Leave all other settings unaltered and click on finish to instantiate the PLL in your hardware.
7. After generating the hardware, you must follow an approach which is different from what you did in previous part which is more convenient and gives you more control over pin assignments. Go to Quartus 2 and select file->new->Block Diagram/Schematic file. Click on OK. You will see that a white grid opens which looks like the surface of a breadboard. Right click anywhere on the white grid and select insert->symbol. In the pop up window which opens, expand the project directory. You will see the name of your SOPC system present there. Select it and click OK. Go to the grid and click anywhere, which inserts the symbol there.
8. The symbol which you included shows you the top level entity as a block. You can see the inputs and outputs. You will need to assign pins to these I/Os carefully. To do this in an easier manner, you will need to rename the signals to the ones used by DE 2. For each input/output, connect a pin by right clicking on the grid, selecting

insert->symbol and then expanding the altera library. Under the expanded library go to primitives->pin and select the type which you need(input or output) .Then rename the wire which joins the pin to the block to the one used by DE 2 for that particular signal. For example zs_cas_n_from_the_sdram_0 should be renamed to DRAM_CAS_N. Take particular note of how the seven segment pins are assigned by DE 2 and rename the signal array accordingly (its important). After you rename the signals, go to assignments->import assignments. Browse to the DE 2 folder on your PC and change the type of file to comma seperated value file (csv). You will find a file called DE2_PIN.txt. Select it and press OK.

9. After completing "Analysis and synthesis", you will find that all the signals in BDF have become connected to the respective pins on DE 2 automatically. You can also go to the assignment editor and verify that all the pins have been connected properly. Finally do a full compilation and program the FPGA by the sof file.
10. Go to Nios 2 IDE and use the SOPC builder system from this project directory. Select the count binary template and run the program on Nios 2 Hardware. You do not need to limit the size of executable this time as you have lot of SDRAM with you (How much?). See how the count value display unit changes when you press the appropriate keys.
11. Now you need to modify the binary counter program to do the following: Add the digits of your individual student ID s. Then add the sums you obtained, lets say you got X as the result after adding the sums. The next step is to do X mod 100, lets say Y is the final result. You have to make the DE2 count to Y in one minute. And you need to display the count on the LCD in octal, on the Seven Segments in decimal and on LEDs in binary. For example, say the student IDs of two members in a group are 21435101 and 23013416. Now 2+1+4+3+5+1+0+1=17 and 2+3+0+1+3+4+1+6=20, adding both the sums we get 17+20=37. Then 37 mod 100 results in 37.Hence this group would display count from 00 to 37 in octal on LCD , in decimal on the Seven Segment Display and on LEDs (in binary of course), all in one minute.