EELE 475
HARDWARE AND SOFTWARE ENGINEERING FOR EMBEDDED SYSTEMS

Lab #4
Introduction to Altera's Qsys

Assignment Date: 10/6/15

Submission Date: 10/13/15

Lab Description

Today's lab will introduce you to use Altera's *Qsys System Integration Tool*. This will require that you use the following Altera tools:

- **Quartus II** (v13.0 – *Not a later version since later versions don't support the cyclone II*) - To compile the VHDL that creates your processor hardware.
- **Programmer** - To download your bit stream to the DE2 board. (QuartusII ->Tools -> Programmer)
- **Qsys** - To create your processor system. (Quartus II -> Tools -> Qsys)
- **NIOS II IDE** - To write, compile, and download your C program. (Qsys -> Tools -> Nios II IDE)

Note: When you create your project, it needs to be in a *directory where **the path name does not contain any spaces***. A suggested location would be C:\users\EE475\<your initials>\DE2_system.

Part 1 – Create your VHDL Project

1. Create a directory called DE2_system.

2. Create a Quartus II Project. Name your project DE2_system and have it point to the directory \DE2_system. Name the top-level design entity DE2_Board_top_level. When creating the project for the DE2 FPGA you need to target the following FPGA device on the DE2 board:

   a. FPGA Family : Cyclone II
   b. FPGA Device : EP2C35F672C6

3. Copy the file: *DE2_Board_top_level_template_all_signals.vhd* from the web site (under Lab Files and Resources) and **rename it** as *DE2_Board_top_level.vhd* and import into your Quartus II

project (In Project Navigator window, click in the Files Tab, right click on Files, and select Add/Remove Files in Project) .

4. Right click on *DE2_Board_top_level.vhd* in the File Tab window and Set as Top Level Entity (Quartus expects a file with the same name as the project as the top level entity).

5. Import the DE2 Board pin assignments by importing the file *DE2_Board_pin_assignments.csv* found on the D2L website . (In Quartus II, click *Assigments* at the top, and select *Import Assignments…*, then browse to the *DE2_Board_pin_assignments.csv* file. Deselect Copy existing assignments into…. and then click OK. In the message window you should see that 425 assignments were written.)

6. Open the file file *DE2_Board_top_level.vhd* and follow notes 2-4 in the header section of the file.

7. You can reopen your project by double clicking DE2_System.qpf where .qpf stands for a quartus project file.

<u>Part 2 – Create your Qsys Design</u>

8. We will follow the instructions found in the Altera Tutorial: *Introduction to the Altera SOPC Builder Using VHDL Designs* (found on the D2L website) but will use Qsys instead. Start going through the tutorial, but **note the Qsys specific notes below**.

   a. The tutorial refers to lights as the project name, just keep in mind that you named it DE2_system.

   b. Read the tutorial from the beginning, but start following the instructions found on <u>Section 3, step 2</u>. Open Qsys rather than SOPC Builder (Tools -> Qsys).

   c. <u>Step 4.</u> Click the clock settings tab. You will notice that clk_0 is already set to External and 50 MHz.

   d. <u>Step 5.</u> The processor is found at Processors -> Nios II Processor. Select Nios II/e. Ignore any errors for the time being. Click Finish.

   e. <u>Step 6</u>. The memory component is found at: Memories and Memory Controllers ->On-Chip->On-Chip Memory (RAM or ROM). **Make the memory size 8192 bytes** (data width is 32 bits. 4K won't work when using the NIOS II IDE).

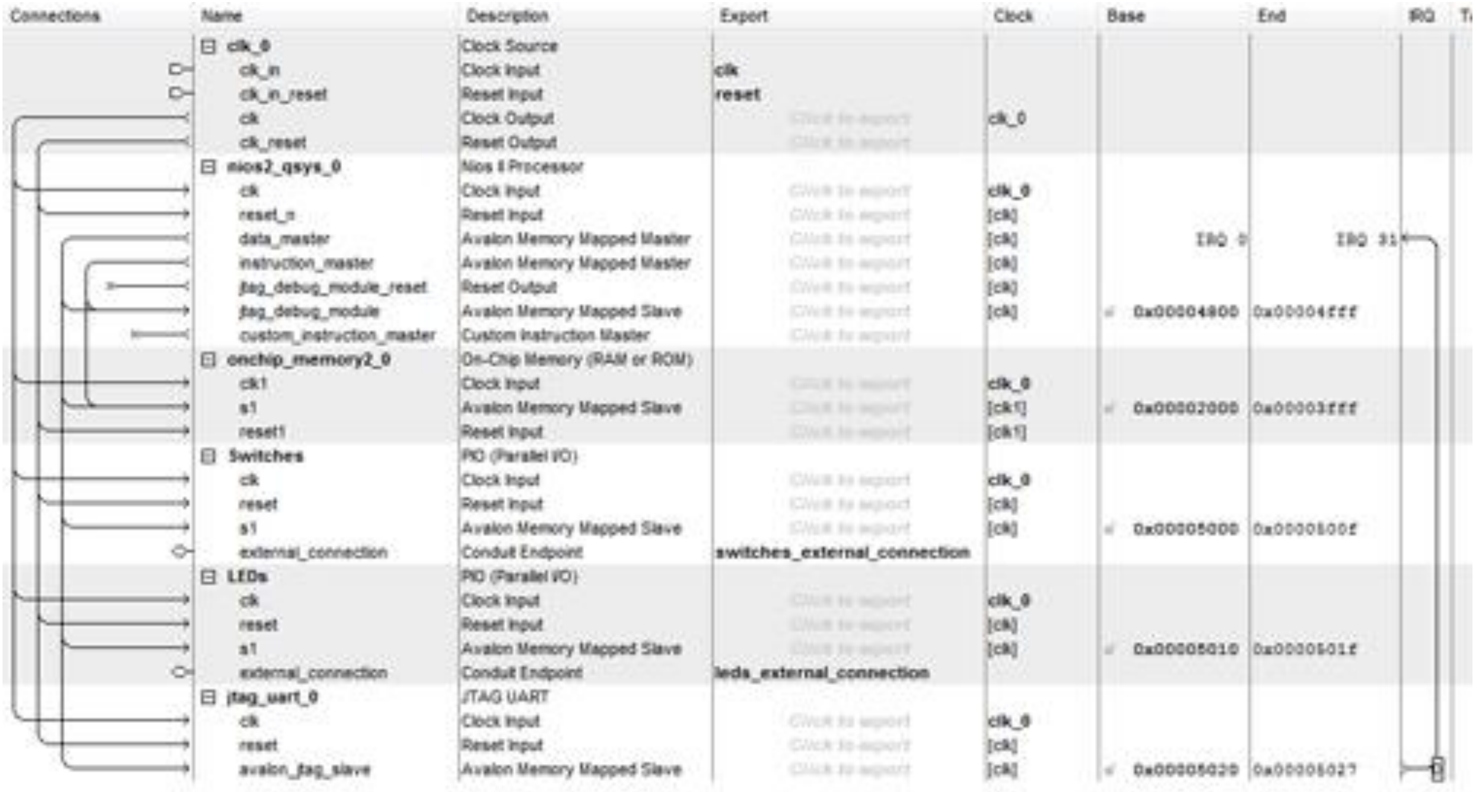
   f. S<u>teps 7 and 8</u>. The PIO is found at: Peripherals->Microcontroller Peripherals->PIO (Parallel I/O). Make the PIOs 18 bits instead of 8 bits (so that all the red LEDs and switches can be exercised). Rename the input PIO to *Switches* and the output PIO to *LEDs* (by right clicking on the name and selecting rename).

g. <u>Step 9</u>. The JTAG UART is found at: Interface Protocols->Serial->JTAG UART.

h. In the connections column, connect up all the clock and reset connections and the peripherals to the Nios processor by clicking on the small bubbles to connect the appropriate signals.

i. In the IRQ, connect the up the jtag interrupt.

Your Qsys system should like something like:



Connections	Name	Description	Export	Clock	Base	End	IRQ
	⊟ clk_0	Clock Source					
	clk_in	Clock Input	clk				
	clk_in_reset	Reset Input	reset				
	clk	Clock Output	Click to export	clk_0			
	clk_reset	Reset Output	Click to export				
	⊟ nios2_qsys_0	Nios II Processor					
	clk	Clock Input	Click to export	clk_0			
	reset_n	Reset Input	Click to export	[clk]			
	data_master	Avalon Memory Mapped Master	Click to export	[clk]	IRQ 0	IRQ 31	
	instruction_master	Avalon Memory Mapped Master	Click to export	[clk]			
	jtag_debug_module_reset	Reset Output	Click to export	[clk]			
	jtag_debug_module	Avalon Memory Mapped Slave	Click to export	[clk]	0x00004800	0x00004fff	
	custom_instruction_master	Custom Instruction Master	Click to export				
	⊟ onchip_memory2_0	On-Chip Memory (RAM or ROM)					
	clk1	Clock Input	Click to export	clk_0			
	s1	Avalon Memory Mapped Slave	Click to export	[clk1]	0x00002000	0x00003fff	
	reset1	Reset Input	Click to export	[clk1]			
	⊟ Switches	PIO (Parallel I/O)					
	clk	Clock Input	Click to export	clk_0			
	reset	Reset Input	Click to export	[clk]			
	s1	Avalon Memory Mapped Slave	Click to export	[clk]	0x00005000	0x0000500f	
	external_connection	Conduit Endpoint	switches_external_connection				
	⊟ LEDs	PIO (Parallel I/O)					
	clk	Clock Input	Click to export	clk_0			
	reset	Reset Input	Click to export	[clk]			
	s1	Avalon Memory Mapped Slave	Click to export	[clk]	0x00005010	0x0000501f	
	external_connection	Conduit Endpoint	leds_external_connection				
	⊟ jtag_uart_0	JTAG UART					
	clk	Clock Input	Click to export	clk_0			
	reset	Reset Input	Click to export	[clk]			
	avalon_jtag_slave	Avalon Memory Mapped Slave	Click to export	[clk]	0x00005020	0x00005027	0

j. Export the Switches and LEDs connections by clicking on the appropriate signal for these peripherals in the Export column.

k. Remember to update the base addresses of the peripherals after any component change. (System->Auto-Assign Base Addresses). This is easy to forget and will cause problems for your C program if you don't update the base addresses.

l. Save the system as *Nios_Qsys* under File->Save As.

m. Before you click generate, go back and set the Reset Vector and Exception Vector to the on-chip memory found on the first page of the NIOS processor customization page, otherwise you will get an error when you try to generate your system. You can do this by double clicking the nios processor component.

n. Instead of <u>Section 4.1</u>, follow Header Note 5 found in *DE2_Board_top_level.vhd*
Get the Nois_Qsys component and instantiation block from Qsys by clicking on the tab *HDL Example* and then cutting & pasting these into your Quartus II design. Set the signals in the instantiated nios_qsys block (port map) to:

```
reset_reset_n                           => KEY(0),
 clk_clk                                => CLOCK_50,
 switches_external_connection_export    => SW,
 leds_external_connection_export        => LEDR
```

Comment out the line(i.e. put -- in front of it for those new to VHDL) :

```
    -- LEDR <= (others => '0');  -- 18 Red LEDs  '1' = ON,  '0' = OFF
```

since we can’t have two drivers driving the LEDs.
The line will be located at about line number ~275.

o. Add the file \Nios_Qsys\synthesis\Nios_Qsys.qip to your design so that Quartus knows about the Nios_Qsys system.

p. You will need to generate timing constraints for Quartus II by using the TimeQuest Timing analyzer to create a .sdc file. (Tools->TimeQuest Timing Analyzer). See section 6 (page 28) of the tutorial *Quartus II Introduction for VHDL Users* that you did in Lab 1 to remind yourself on how to do this. There is also a .sdc file in D2L under lab 4 currently named lab4_sdc.txt. Download this file and then rename to lab4.sdc and add this file to your project.

q. Compile your VHDL code in Quartus II. Verify that your design can run at 50MHz.

r. Ask the instructor for help if you get compiler errors.

s. Skip Section 5. We will be using the Eclipse based software build tools for NIOS II instead (Part 3 below).

<u>Part 3 – Software</u>

9. Create a subdirectory called \Eclipse_Workspace and a subdirectory called \Software under your \DE2_System directory.
10. Open up the NOIS II IDE (you can do this from within Qsys -> Tools-> *Nios II Software Build Tools for Eclipse*)
11. Set the workspace to the directory Eclipse_Workspace that you created.

12. Select File->New->*Nios II Application and BSP from Template*.

   a. For the SOPC Information file name, browse to the file Nios_Qsys.sopcinfo found in the DE2_System directory (or where you created your project).

   b. Name your Project “lights”.

   c. Select the *Blank Project* Template. Click next.

   d. Click finish

13. Open the Nios II BSP Editor (Eclipse->Nios II->BSP Editor).

   a. Select File->open

   b. Browse to your settings.bsp file that is found in\Software\lights_bsp

14. We want a minimal software footprint since we are targeting a very limited amount of memory, so in the Main tab and in the **hal** window

   a. Select enable_reduced_device_drivers

   b. Select enable_small_c_library

   c. Click Generate

   d. Click Exit

15. Right click on *lights* in the Project Explorer and then select New->Source File. Name it lights.c click finish.

16. Copy the contents of lights.c file from the web and paste it into this file in the editor.

17. Now look at the system in Qsys and find what the BASE ADDRESSES are for the switches and LEDs and update the lights.c code with these addresses since they will most likely be different. The Base Addresses can be seen in the Base column under the *System Contents* tab, or by clicking on the *Address Map* tab.

18. Compile the project. Select Project->Build All. This may take a bit of time as it builds all the device driver code. Pay attention to any messages or errors. You need to have it generate a .elf file. You will want to see the message [BSP build complete] to know it compiled correctly.

<u>Part 4 – Run the Software</u>

19. Turn on the DE2 board. Open the Quartus II -> tools -> programmer and download the Quartus II bit stream. Note: if you make any changes to the Qsys system, you will need to recompile the Quartus project first. (A common mistake is running software on the wrong system).

20. In Eclipse select Run->Run and select run as Nios II Hardware (you may have to wait a bit while it looks and finds the device). If this doesn't work and complains about not seeing the .elf file, there was something wrong with your project.

21. The Red LEDs (above the switches) should light up when the switch is pushed forward.

Deliverables

- No lab report
- Turn in the signed Instructor Verification Sheet
- Submit the VHDL code to D2L (the part that you modified).
- Submit your C code to D2L.

Instructor Verification Sheet

Lab #4
Introduction to Altera's Qsys

EELE 475
HARDWARE AND SOFTWARE ENGINEERING
FOR EMBEDDED SYSTEMS

Name : ________________________

Demo #1 : C code running the NIOS II processor that allows the slide switches to turn on & off the red LEDs.

Verified: ___________________________________ Date:____________________