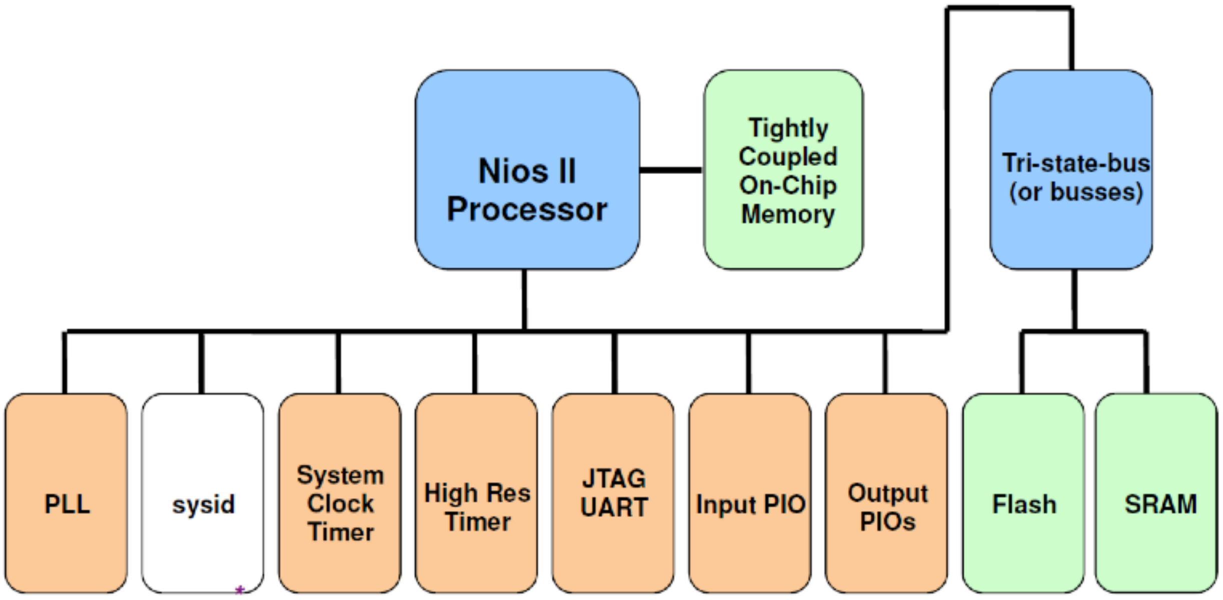
ECE 178 - Embedded Systems
Assignment Three – Due: Demonstration on 02/25/2015

The purpose of this exercise is to learn how to design an embedded system in FPGA device and using assembly language to develop some applications program on it. FPGA devices contain both memory and logic elements that can be used along with intellectual property (IP) components to implement a system-on-a-programmable-chip (SoPC) for embedded appications. Designing system with embedded processors requires both hardware and software design elements. Use the Quartus II and Qsys Builder CAD tools to design the hardware for a fully functional, customize, soft- processor to include the followings:



- Nios II CPU
- Tightly-coupled on-chip instruction memory
- External SRAM memory controller
- External Flash memory controller
- JTAG UART periperals
- PIO peripherals, configured to interface to LEDs and switches
- High resolution times
- Phase-locked loop (PLL), and more

Nios II Processor Hardware Design

First we create a Quartus II project.

1. Start Quartus II
2. Create a new project (File > New Project Wizard)

Select a working folder and name the project and top entity "NiosII"

Note: Do NOT use a folder path with blank spaces

- Select the Cyclone II EP2C35F672C6 device
- Leave the EDA tool settings unchanged
- Review the summary, select Verilog and if all settings are correct click "Finish" to create the project

3. Create a schematic file (File > New > Block Diagram/Schematic File)
4. Save the schematic file as niosII.bdf (File > Save As)

We now have an empty project within which we shall create a Nios II processor. Use Qsys to combine any IP blocks with an Avalon Bus Structure. The Nios II processor is equipped with an Avalon Bus and is therefore instantiated within Qsys Builder.

5. Start Qsys: From the Quartus II software Tools menu choose Qsys (Tools > QSYS Builder)
   Then from within Qsys, go to the File menu, and choose Save as …, and enter the system name **niosII** and press save. The Qsys should be empty with the exception of the Clock Source component.
6. Check the initial clock setting for the Qsys project: Go to the Clock Settings tab, and ensure that the External clock is set to 50 MHz to match the oscillator on your board. Then double-click on the clock name, and change it to clk by typing directly over the clk_0. Press enter when you are done.
7. Go to the **Project Settings** tab, and ensure that the Device Family matches the FPGA family you are using. Qsys should have pulled the right device information from your Quartus II .qsf file.
8. Then return to the Qsys **System Contents** tab to add the embedded required to construct your Nios II system in Qsys.

9. Add PLL Component to System. Add the "**Avalon ALTPLL**" component to the project from the **PLL** folder in the Qsys **Library** pick-list. To do this, double-click the **Avalon ALTPLL** component to add to your system. Note: you will accept most of the default PLL setting in the Megawizard. The specific setting that you have to make are described in the next few steps.
10. On the **Parameter Settings** tab, set the frequency to match that of your board (50 MHz).
11. Navigate through the pages of the PLL MegaWizard using **Next** button until you get to the **Output Clocks** page. Set the PLL **Output Clock** properties as shown below:

	Output Tap Settings	
	C0	C1
Cyclone II with Input Frequency of 50MHz	Mult. Factor = 17	Mult. Factor = 17
	Div. Factor = 10	Div. Factor = 10
	Phase Shift = 0	Phase Shift = -3.32 ns

Note: to enable a second PLL output tap in the MegaWizard, click on the "**Use this clock**" check box on the **clk c1 Output Clocks** page of the MegaWizard. A -3.22 ns phase shift is added to the C1 output for those boards with SSRAMs to meet timing.

12. After you configure the PLL to match the setting described previously, press **Finish** repeatedly until you exit the MegaWizard. This component is now added to the system.
13. Then re-name the component to **pll** by right-clicking on it, selecting rename, and typing **pll** over top of the highlighted component name. Press enter when you are done.
14. Next, label the PLL output taps for easy identification later. Return to the **Qsys Clcok Setting** tab. Double-click on **pll_c0** in the **Name** column, and type over the text to rename it **sys_clk**, and press enter. Then re-name **pll_c1** to **sram_clk**.
15. Set the Qsys connections to drive the **pll** from the **Clock Source** block: Back on the **System Contents** tab, connect the Clock Source's **clk** (i.e. **Clock Output**) interface on to the pll's **inclk_interface** (i.e. **Clock Input**) interface by moving the mouse into the **Connections** panel and clicking on the appropriate connection dot. By doing this, you are feeding the FPGA input clock through to the PLL via the Clock Source block.

Tip: If you click the mouse on either the master or slave interface inside a component, it will highlight the pipes in the connection grid and may make this process a little easier to visualize.

16. Then click in the **Export** column next to pll **Clock Output C1**, and type **sram_clk**. This PLL clock output will be used to drive the SRAM clock on the development board.
17. **Add Nios II Processor and System ID Components to System**. From the Qsys Library pick-list look in the **Processors** folder, and select Nios II Processor, and then click Add. Select **Nios II/s** for the processor core. Accept the default Hardware multiplication type for your device.
18. Click on the **Cache and Memory Interface** tab, set the **Instruction Cache** size to **4 Kbytes a**nd include 1 **tightly coupled memory instruction master port**.
19. Now, click on the **JTAG Debug Module** tab. Select the **Level 3 JTAG Target Connection Download** option.
20. Click **Finish**. This will add the Nios II processor to the Qsys system.
21. **Renam**e the processor by right clicking on it and selecting Rename. Type in **cpu** and hit enter.
22. Then connect the **Clock Input** interface on the cpu to **sys_clk** by either choosing **sys_clk** in the drop-down menu in the **Clock** column beside the Nios II Processor or by connecting the **c0** output of the Avalon ALTPLL component to the cpu **Clock Input** interface directly in the connection panel in Qsys. (Both actions achieve the same result.)
23. Then connect the cpu **data_master** interface to the **pll_slave** interface on the PLL.
24. Add a **System ID** peripheral to the system, and accept the default setting. It is located in the **Pheripherals>Debug and Performance** folder in the Qsys Library pick-list.
25. Connect its **clk** interface to **sys_clk** (i.e. pll output tap **c0**) and its **control_slave** interface to the cpu **data_master** interface.
26. Rename the component, **sysid**.
27. **Add Tightly Coupled On-Chip Memory to System**. Now add a tightly on-chip memory to attach to the processor. From the Qsys **Library** pick list, look inside the **Memories and Memory Controllers > On-Chip** folder, and select **On-Chip Memory (Ram or Rom)**, and click **Add**.

28. Keep the **memory width at 32 bits**, and ensure the **Total Memory** size to **4096** bytes. Finally, enable **Dual-Port Access**. Accept all other defaults.
29. Click **Finish** and rename the component **tightly_coupled_instruction_memory**.
30. Set the clock driving both Avalon Slave interfaces to **sys_clk** (i.e. pll output tap **c0**).
31. Next, ensure that the appropriate master/slave connections are made between the cpu and the Tightly Coupled Memory.
    That is, **tightly_coupled_instruction_master_0** interface on the cpu should connect to the **tightly_coupled_instruction_memory s1** interface on the On_Chip Memory, while the **data_master** interface on the cpu should connect to the **s2** interface on the On-Chip memory.
32. **Add the Interval Timer to System**. Most control system use a timer component to enable precise calculation of time. Select the **Interval Timer** from the **Peripherals > Microcontroller Peripherals** folder and click **Add**. Select the defaults and in the Presets list select Full-feautured, and click **Finish**. After the timer has been added to your system, rename it to **sys_clk_timer**.
33. Connect the timer to **sys_clk (**i.e. pll output tap **c0**), and wire its **s1** interface to the cpu **data_master**.
34. **Add another Interval Timer to the system**. This time, change the period from ms to micro-second. Then click Finish.
35. **Rename** this timer to **high_res_timer**, and connect it to **sys_clk** (i.e. pll output tap **c0**), and write its **s1** interface to the cpu **data_master** just like you did with the sys_clk_timer.
36. **Add JTAG UART to System**. The JTAG UART provides a convenient way to communicate character data with the Nios II processor through the USB-Blaster download cable. From the left hand window pane find the **JTAG UART** from the **Interface Protocols > Serial** folder, and click **Add**. Accept the defaults, and press **Finish**.
37. **Rename** the component to **jtag_uart**.
38. Connect the jtag_uart to **sys_clk** (i.e. pll output tap **c0**), and wire up the **Avalon_jtag_slave** interface to the cpu data master.
39. Tip: By hovering your mouse over the dots on the connection grid, the master to which that connection dot it associated will appear as a tool tip in the Qsys GUI. By getting in

the habit of taking a quick look at the message in the tool tip before making a connection, you reduce your chances of making an incorrect connection.

40. **Add a PIO**

    To control the Seven Segment Display

    Enter a width of 16 bits, with output ports only. Rename this peripheral **seven_seg_pio**.

41. **Add an 8-bit PIO input circuit**

    To be connected to the toggle switches. Rename this peripheral to **switches**.

42. **Add another PIO for Pushbuttons**

    Set the width of the interface to 4 bits and set the direction to input port only. Rename it **pushbuttons**

43. **An 8-bit PIO output circuit**

    To be connected to green LEDs. Rename it to **gled**

44. **An 8-bit PIO output circuit**

    To be connected to red LEDs. Rename it to **rled**.

45. **A one-bit PIO circuit**

    It will serve as a status flag, which will be connected to the pushbutton key *KEY*3. Configure it to be an input port that is one bit wide. Also, in the Input Options tab select the following:

    a. Synchronously capture feature activated by the Falling edge for the *Edge capture* register.
    b. Generate IRQ interrupt on Edge

    Rename it to **status_flag**.

46. **Establish IRQ Priorities**

    Assign the IRQ numbers: Go to the system menu and select Auto-Assign IRQs to establish some basic IRQ assignments. Then, take a look at the IRQ values that result. Edit them to ensure that the sys_clk_timer gets priority 0 (highest priority) and JTAG_UART gest priority 2 (lowest priority).

47. **Connect the Reset and Assign the Base Addresses**. Connect all the reset interfaces for the design in one shot: go to the Qsys System menu, and select **Create Global Reset Network.**

48. Auto assigns the base addresses for all the components so that there are no overlapping addresses. Return to the **System** menu, and select **Assign Based Addresses**.
49. This should get rid of most of the Qsys warning and error messages.
50. **Set the Nios II Processor Reset and Exception Handler Locations**. Finally, set the reset and exception addresses for the CPU: Double-click on the **cpu** component to re-open it
51. **Set System Pipelining Level for the Qsys Interconnect**. Go to the **Project Settings** tab, and change the pipelining limit from 1 to 2 stages to help increase system fmax.
52. **Generate Qsys System**. In the System Contents tab Message box, check to see if you have any remaining memory addresses violations. If so, fix them by auto-assigning the base addresses (see System menu).
53. Then go to the **Generation** tab, and **uncheck** the **Create block system file** (.bsf) checkbox.
54. Save your Qsys system (**File > Save**).
55. Press the **Generate button**. Qsys will now create the parameterized hardware system for you. The generation process take several minutes.
56. Add the nios II system output files to the Quartus II project: After Qsys has finished generating your embedded sub-system, close the Qsys system generation dialog box, and go to Quartus II. Add the **niosII.qip** file to the project (**Project** menue > **Add/Remove Files in Project)**.
57. Browse to the **niosII/synthesis** folder by pressing the … button beside the **File Name** field in the Quartus II **Settings** page. When you locate the **synthesis** directory, be sure that the browsing filter is set to find files of **All Files**, or you will not see the .qip file.
58. Select the **niosII.qip** file, and press **Open.** Then press **Add** on the Quartus II **Settings** page.
59. Click **OK** to close the setting page.
60. Start compilation in Quartus II by selecting **Processing>Start Compilation**.

At this time you have finished creating a Nios II based embedded system system. In the next phase you will use the Altera Monitor program to develop and implement application software onto your designed embedded system.