ECSE-325
Digital Systems

Lab #3 – *Digital System Design using Qsys* **Winter 2023**

In this lab you will create a digital system using Qsys that will implement the ARCCOS circuit as a peripheral module on the CycloneV FPGA fabric. This will be accessed by a program running on the embedded Arm HPS cpu on the CycloneV chip.

Note: It is best to use the Altera Quartus v15.1 installed on the computers in TR4180 for this lab.

Newer versions of Quartus Prime (v19 and above) can be used on your own computers, but you will have to install WSL v1.0 (Windows Subsystem Linux) if it is not already. Otherwise, the compilation in Platform Designer will give some errors.
see
https://www.intel.com/content/www/us/en/support/programmable/articles/000074066.html

Creating a New Project

In order to have your design working, you should implement the following steps carefully.

1. You are provided with a preset file on myCourses. Create a new folder for your project, and copy the file:

Altera_Cyclone_V SOC_Development_Kit_HPS_Presets.qprs

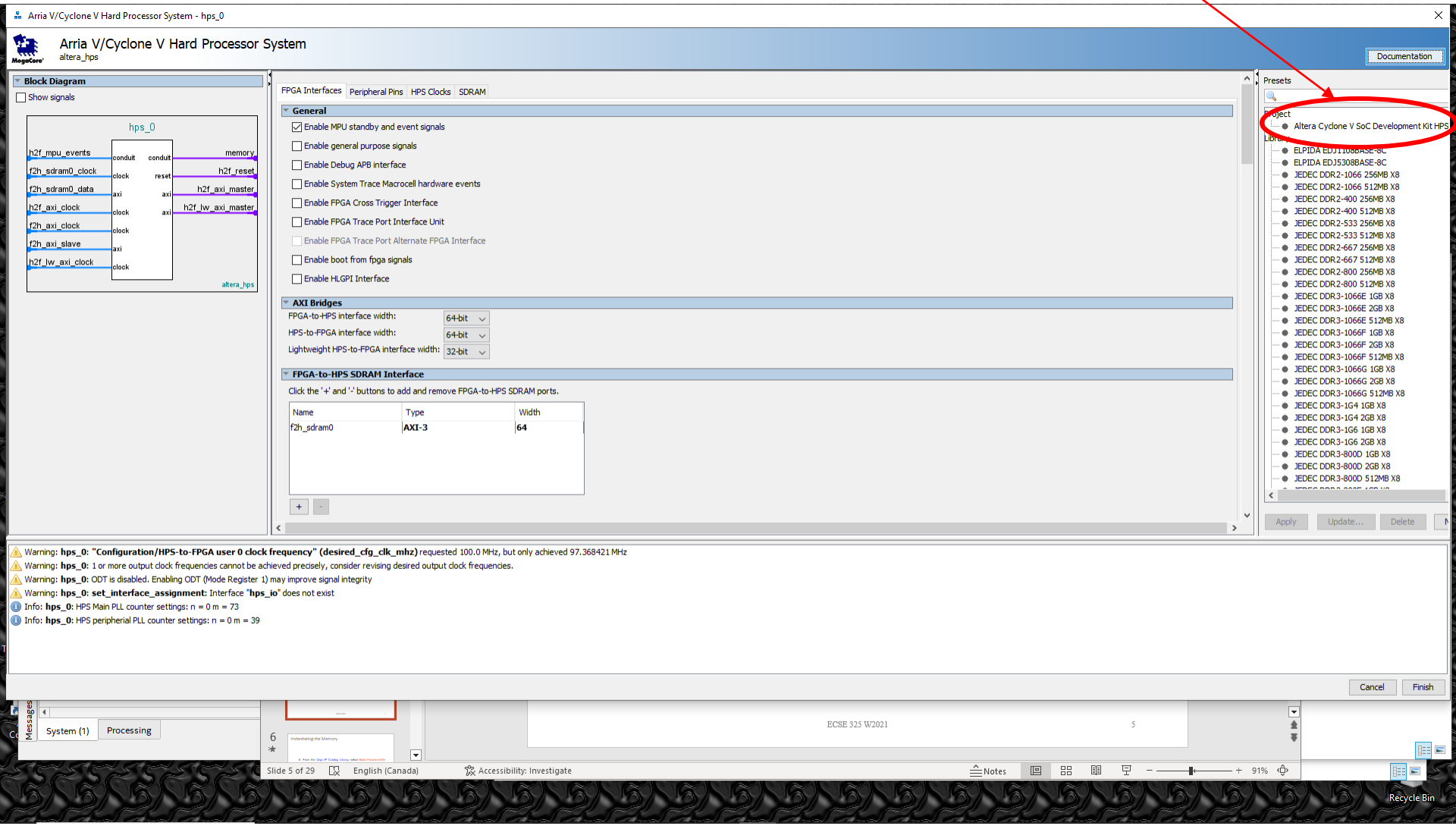
to your project directory.

1. Start Quartus. Using the ”New Project Wizard”, name your project name as ”qsys_lab”. Do not add any files to your project. Choose the same device as in the previous labs.

Instantiating the Processor

3. Start Qsys (or Platform Designer) from the Quartus Tools menu. Under Qsys Tools/Options add the project directory to the IP Search Path. Click Finish. You shouldnt get any warnings and errors at this point.
4. From the Qsys IP Catalog Library, in the Library section, select the Processors and Peripherals/Hard Processor System/Arria V/Cyclone V Hard Processor System and click ”Add”.
5. In the window that pops up, select the Preset Altera_Cyclone_V_SOC_Development_Kit_HPS_Presets. This will load in various settings for the hard processor device. Click Apply, then click Finish. There will be a bunch of red error messages. Ignore these, they will be fixed later.

Altera_Cyclone_V_SOC_Development_Kit_HPS_Presets



Instantiating I/Os

6. From the Qsys IP Catalog Library, select Processors and Peripherals/Peripherals/PIO (Parallel I/O) and click "Add". Set the bit-width to 10. Set direction to OUTPUT. This will be used to turn on/off the single element LEDs on the board. Click "Finish". Right-click on the pio_0 label at the top of the module, and select "Rename" and rename the component to "LEDS".
7. Add another PIO component, but this time with direction set to INPUT. Set the bit width to 10. Rename to "SWITCHES". This will be used to read the state of the slide switches on the board.

Instantiating I/Os

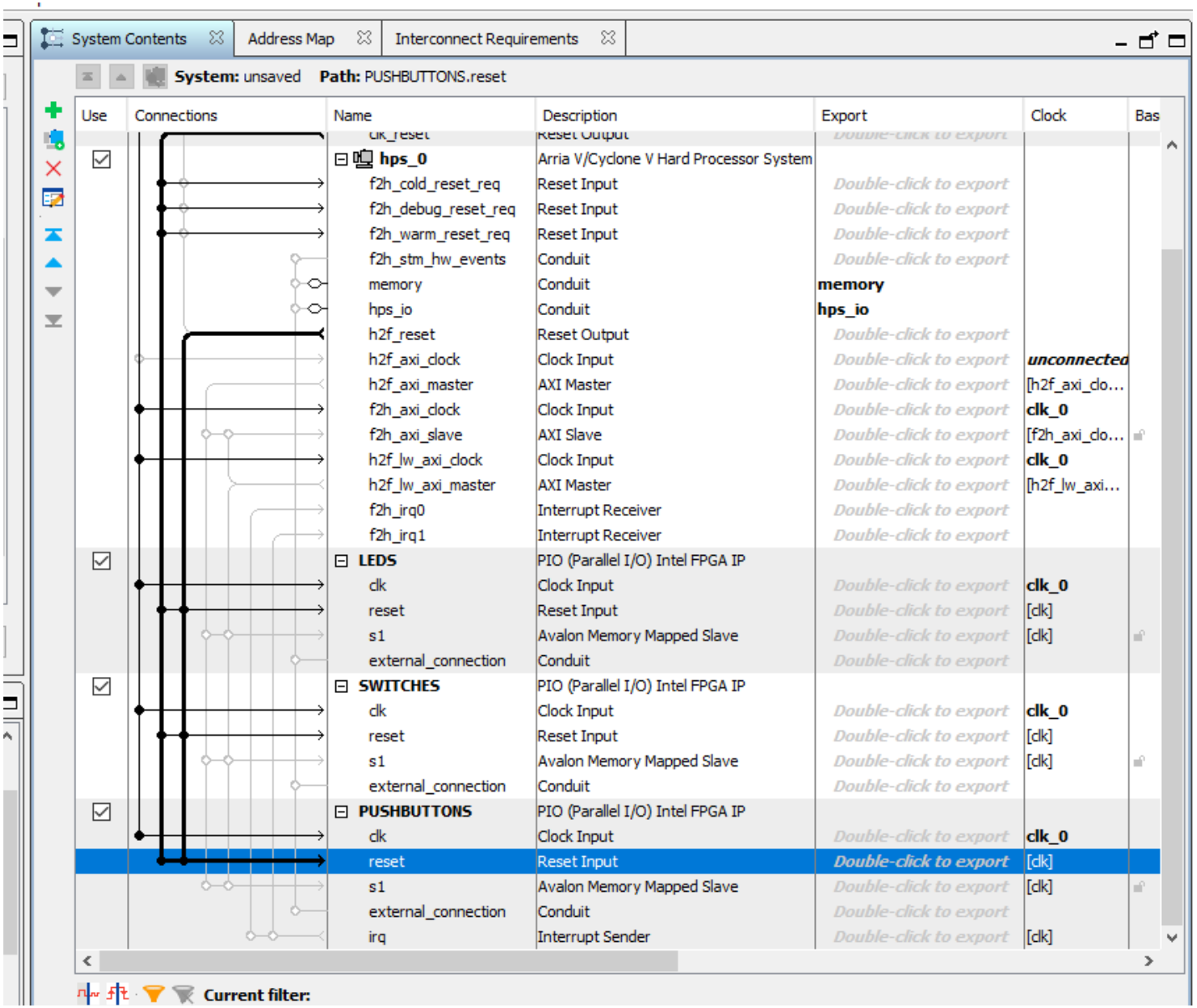
9. Add another PIO component, but this time with direction set to INPUT. Set the bit width to 4. Check the Synchronously Capture box, and set Edge Type to FALLING. Check the Generate IRQ box and set IRQ Type to EDGE. In Qsys rename to ”PUSHBUTTONS”. This will be used to read the state of the pushbuttons on the board.

Making the Connections – CLK, RESET

10. In the Systems Contents pane of the Qsys window, make the connections between the components. Connect all of the clk inputs of the added components to the clk clock output signal of the clk 0 component by clicking on the appropriate bubbles in the Connections column. Also connect this clk to the h2f_axi_clock, f2h_axi_clock and h2f_lw_axi_clock inputs.
11. Connect the clk_reset signal of the clk 0 component to the f2h_cold_reset_req, f2h_debug_reset_req, and f2h_warm_reset_req of the hps 0 component and to the reset inputs of the other added components. Do the same with the h2f_reset output of the hps0 component (just to the other component reset inputs). This will OR the two reset signals together.

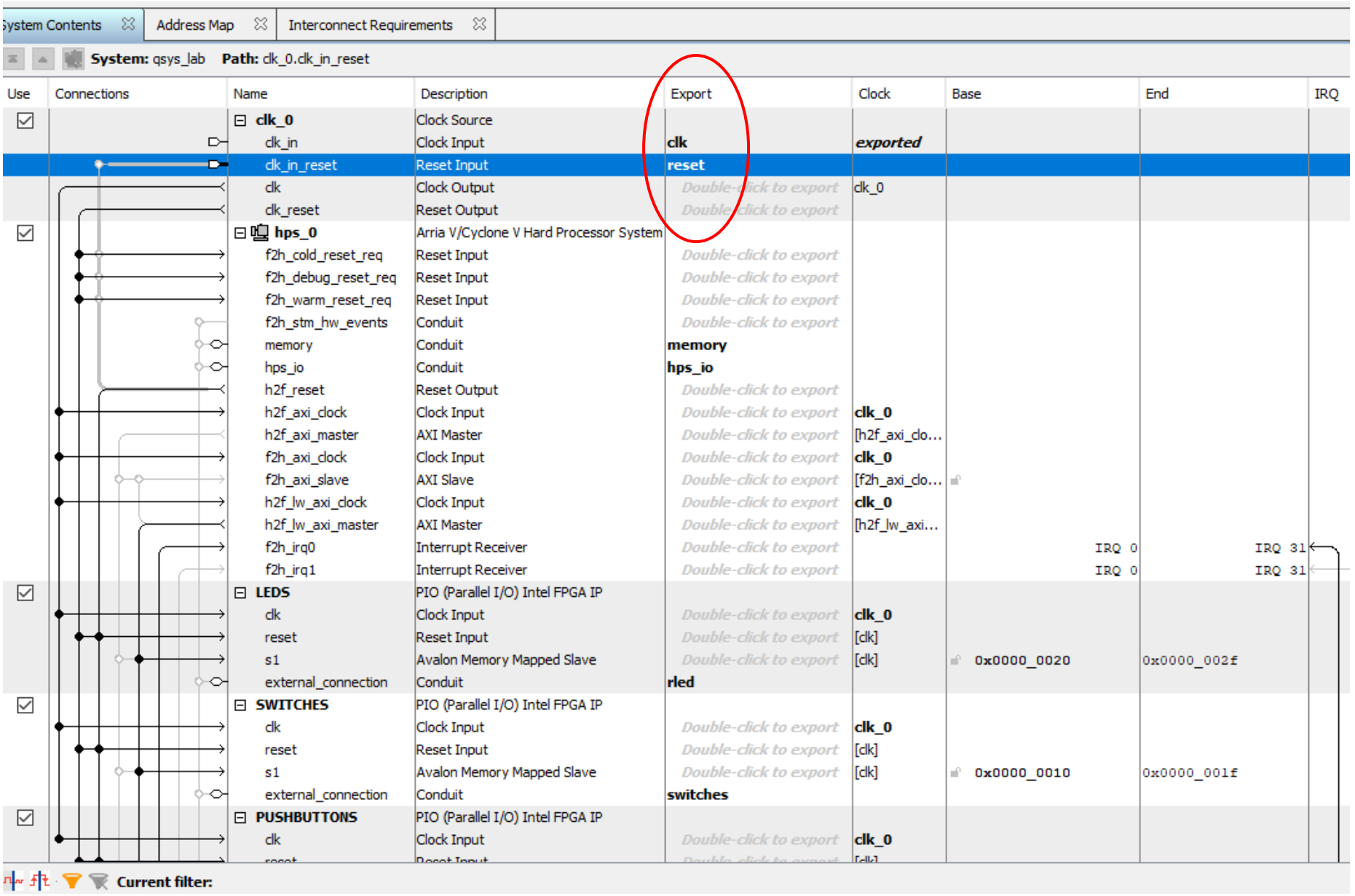
Making the Connections – CLK, RESET

The Qsys System Contents window should look like this at this point:



Making the Connections – CLK, RESET

11. Also, make sure that the *clk_in* and *clk_in_reset* lines are exported. Give the exports labels such as *clk* and *reset*. Once you do that, the signals *clk* and *reset* will show up in the Pin Planner, and you can assign them to FPGA pins. Connect the *clk* signal to the pin connected to the 50MHz clock (as you did in lab 2) and connect the *reset* signal to the pin connected to pushbutton(0). Do not connect the *clk_in* and *clk_in_reset* ports (on the left side) to any of the other Avalon port signals.

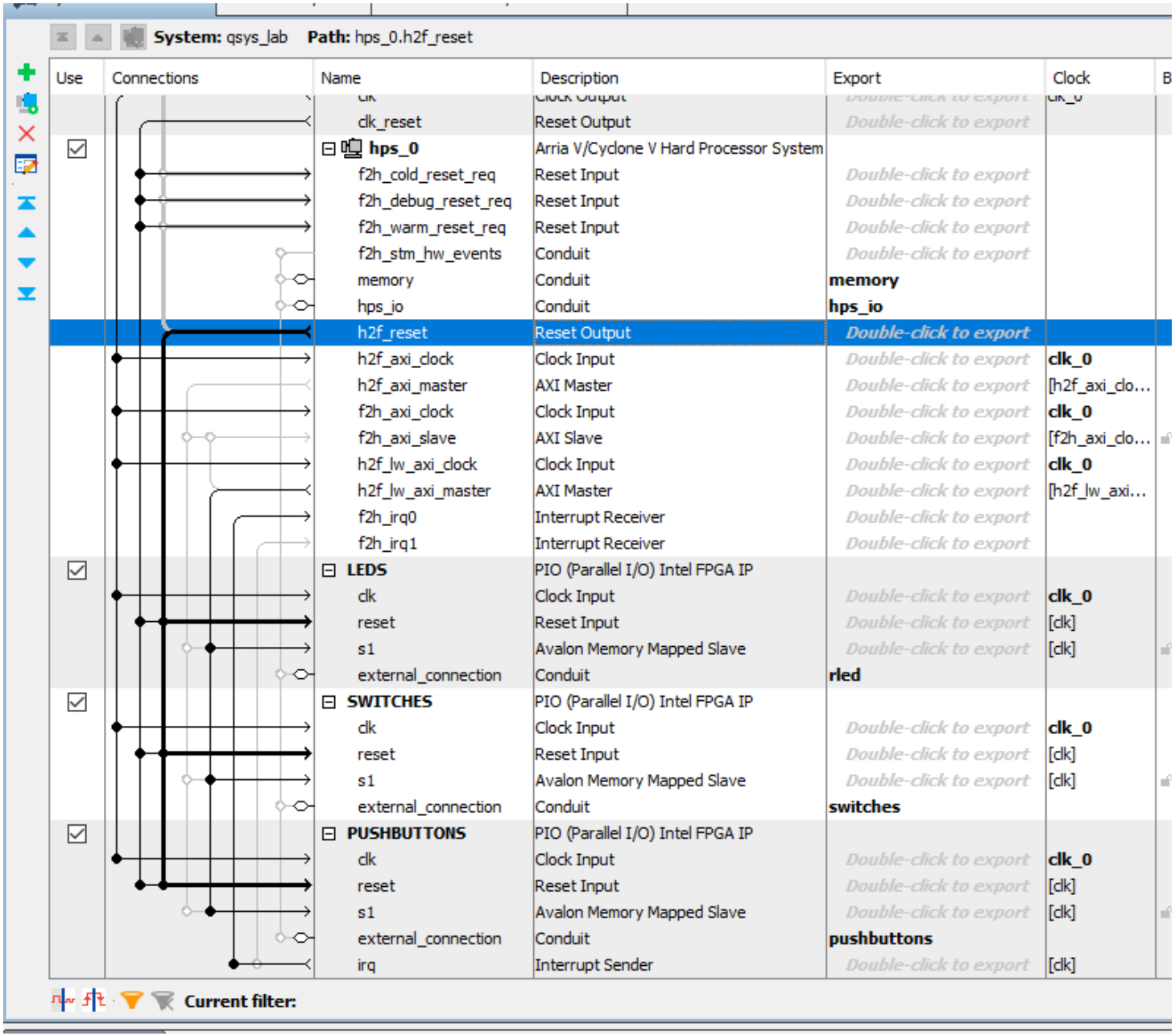

System Contents
Address Map
Interconnect Requirements
System: qsys_lab Path: clk_0.clk_in_reset
Use
Connections
Name
Description
Export
Clock
Base
End
IRQ
clk_0
Clock Source
clk_in
Clock Input
clk
exported
clk_in_reset
Reset Input
reset
clk
Clock Output
Double-click to export
clk_0
clk_reset
Reset Output
hps_0
Arria V/Cyclone V Hard Processor System
f2h_cold_reset_req
f2h_debug_reset_req
f2h_warm_reset_req
f2h_stm_hw_events
Conduit
memory
hps_io
h2f_reset
h2f_axi_clock
h2f_axi_master
AXI Master
[h2f_axi_clo...
f2h_axi_clock
f2h_axi_slave
[f2h_axi_clo...
h2f_lw_axi_clock
h2f_lw_axi_master
[h2f_lw_axi...
f2h_irq0
Interrupt Receiver
IRQ 0
IRQ 31
f2h_irq1
LEDS
PIO (Parallel I/O) Intel FPGA IP
reset
[clk]
s1
Avalon Memory Mapped Slave
0x0000_0020
0x0000_002f
external_connection
rled
SWITCHES
0x0000_0010
0x0000_001f
switches
PUSHBUTTONS
Current filter:

Making the Connections

12. Connect the irq signal of the PUSHBUTTONS component to the f2h_irq0 of the hps 0 component.
13. Connect the Avalon Memory Mapped Slave (bus) signals (these are labeled s1) of all the pio components to the h2f_lw_axi_master on the hps 0 component.
14. Double click on the "Double-click-to-export" text in the "Export" column for the port named external connection for the LEDS component. Type in "rled". This tells Qsys that this component will have some signals that will be connected externally (in this case to the red LEDs on the board). Similarly, export an external connection for the SWITCHES and PUSHBUTTONS components (name these switches and pushbuttons, respectively).

Making the Connections

The Qsys System Contents window should look like this at this point:

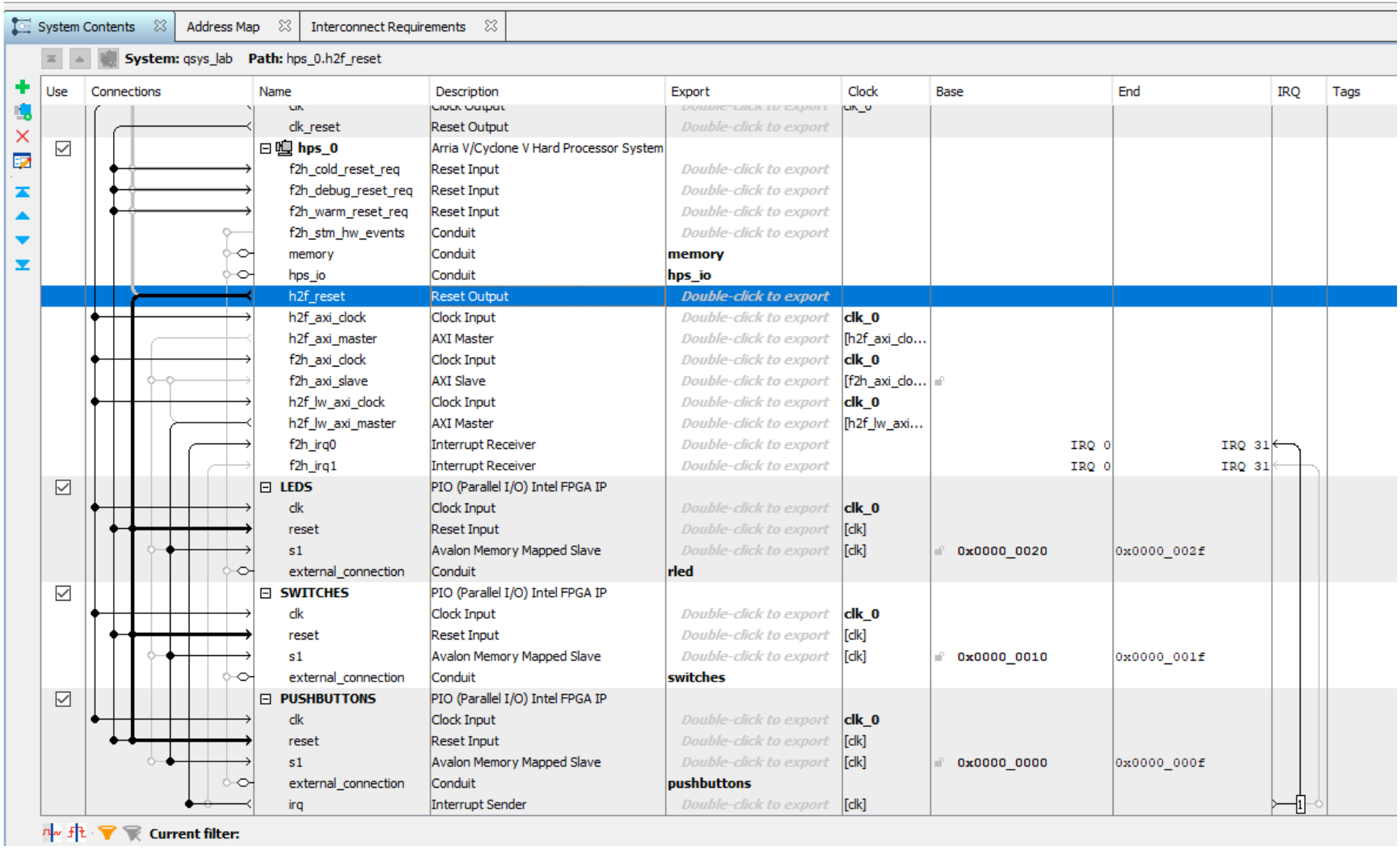


Making the Connections

15. In the IRQ column, change the pushbuttons IRQ value to 1.
16. Notice the Base column, these entries show the base addresses of the Avalon interface for each component. Currently they are all set to the same value, 0x00000000. We need to make them different. To do this, select System/Assign Base Addresses from the Qsys menu bar. This will change the values to ones that do not overlap.
17. There should be no more error messages in the Qsys Message pane. There may be some warnings.
18. Save the Qsys settings file as "qsys_lab.qsys". Close the save system window when it finishes saving.

Making the Connections

The Qsys System Contents window now should look like this:



Generating VHDL Files

19. Click on “Generate HDL” in the “Generate” tab. In the “Create HDL design files for synthesis” box select “VHDL”. Uncheck the “Create timing. . . ” box and select “None” for “Create simulation model”, since we will not be doing any simulation.
    Uncheck the “Create block symbol file” box.

The Output Directory Path should be your project directory.
Click on “Generate”. This will produce the VHDL file which basically instantiates the components specified in your Qsys system.

Exit Qsys when finished.

Project Setup

20. In Quartus, select Project/Add/Remove Files in Project. Then, add the qsys_lab.qip file in the directory qsys_lab/synthesis (which was created when you ran Generate in Qsys).
21. In Quartus, open the file /synthesis/qsys_lab.vhd (which should have been generated by Qsys). In the /Project menu tab, first select "Add Current File to Project", then select "Set as Top- Level Entity".
22. We will need to map the FPGA pins to the DE1-SoC board connections. Before doing this, you should run the Analysis and Synthesis (under Processing/Start/Start Analysis and Synthesis/). This will determine what pins are unassigned and need to be mapped.

Pin Assignments

23. When Qsys generates any HPS component, Qsys also generates the pin assignment TCL Script File (.tcl) to perform pin assignments for the memory connected to the HPS. The script file name and location is at:

    ../qsys_lab/synthesis/submodules/hps_sdram_p0_pin assignments.tcl

24. Run this script to assign constraints to the SDRAM component. To do so select Tools/Tcl Scripts… Once the script has run, we can assign the rest of the standard DE1-SoC pins.

25. To assign the rest of the pins, select Assignments/Import Assignments and choose the file qsys_tutorial.qsf to load in some pre-defined assignments (otherwise it would be quite tedious to enter them all yourself using the Pin Planner). Note that in this assignment, KEY(0) (the rightmost pushbutton) is mapped to a reset signal, and thus will not be accessible by the HPS via the Avalon bus (i.e. the signal pushbuttons[0] is not connected to anything).

26. Re-compile the design. There should be no errors.

Intel FPGA Monitor Program

27. Start up the Intel FPGA Monitor Program. This allows programming of the HPS (ARM CPU) on the Cyclone V device. (this can be found in the Altera 15.1 UP directory)
28. Select File/New Project, to startup the new project wizard. Enter the path to the Project Directory, and set the name as qsys_lab. Under the "Select a processor architecture" select "ARM Cortex-A9". Under "Specify a system" select the system "select <Custom System>". For the System description file choose "qsys lab.sopcinfo" (which should have been generated for you by qsys), and for the FPGA programming file choose "output files/qsys lab.sof". In the "Specify a program type" page, choose "C Program". In the "Specify program details" page select the file "test_leds.c". This is a demo program to get you started. Click on "Finish". Do not download the system file yet, unless you have already connected the DE1-SoC board.

Testing the LEDs

29. Open the test_leds.c program file and take a look at it. It consists of some timer code and some code to read the DE1-SoC board switches and write to the DE1-SoC board LEDs (using the HPS-FPGA bridges/Avalon busses that were set up with Qsys).
30. Connect the DE1-SoC board to the desktop computer via a USB cable (use the USB connector on the left side of the board, next to the red power button). Turn on the board power.
31. Download the System settings to the board using /Actions/Download System. This will download the .sof file generated when you compiled the qsys_lab.vhd top-level file. You might have problems here if the USB cable is not plugged in, or the board power is not on. You should otherwise get a "Download System Success" message window popup.

Testing the LEDs

32. Compile the program with /Actions/Compile. If there are no errors an ".sre" binary file will be generated that can be downloaded to the HPS over the USB cable. To do this, run /Actions/Load. You will know that this works when the debugger icons (just under the File/Edit/Actions bar) become active (not grayed out). Click on the "Continue" icon (or select /Actions/Continue). This will kick the HPS into action. You should now see the red LEDS reflecting the state of the slide switches on the board.

Observing the C Code for LEDs

33. Play around with the c-code and make some changes. For example, you could use the timer code provided to make a counter to change the value sent to the LEDs (instead of using the switches).
34. Next, you will use Qsys to add a new component of your own design. This will also be connected to the Avalon bus and communicate with the HPS via the light-weight bridge (h2f_lw_axi).

Creating a Custom Slave Module

35. First, make a new VHDL design entity called "qsys_lab_custom_component.vhd". This entity will contain a component instantiation for the high level **gNN_ARCCOS_DISPLAY** (your name may be different) design entity made in lab #2 (the one that includes the 7-segment decoder components). We do it this way so that any changes will mainly be done to the second file. If the first file does not change, we will not require any re-generations with Qsys (after the initial one to tell it about the new component). The custom component design entity should provide the necessary signals for connecting as a slave to an Avalon memory-mapped master. As such, it needs the following port signals described in the VHDL code:
    - clock (clock signal)
    - resetn (active low reset)
    - address (8 bit address to be used internally by the component to direct data)
    - writedata (32 bit data to be sent to the component – only 8bits will be used)
    - write (active when a write transaction is to be performed)
    - chipselect (active when a transaction is being performed)
    - segments1, segments2, segments3, segments4 (conduits to the 7-seg LEDS)

The custom module should contain the gNN_ARCCOS_DISPLAY entity as a component, and properly handle sending data to and from the Avalon slave ports to the gNN_ARCCOS_DISPLAY circuit.

Creating the Custom Hardware in Qsys

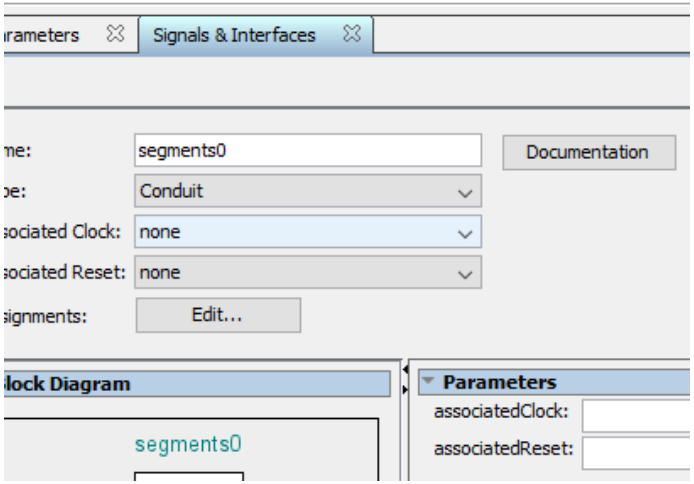
36. Start Qsys again and load the qsys_lab.qsys file.
37. In the IP Catalog select "New Component" and click on "Add". An information window will appear, in which you will provide details about the new component. Give it the name "qsys_lab_custom component" with Display name "custom_component". Leave the Group blank. For the description enter "qsys lab custom component". Enter your group student names in the Create by tab.
38. Select the "Files" tab in the Component Editor window. In the "Synthesis Files" section click on "Add File". Select the top-level VHDL file you just created (qsys_lab_custom_component.vhd). Then click on "Analyze Synthesis Files". This will look at the design entity descriptions to see what the input and output ports are and match them to Avalon interface signals. (if you get a "*Info: Error: No modules found when analyzing null.*" message then you probably have a typo or syntax error in your VHDL file).

Creating the Custom Hardware in Qsys

39. There will be some error messages in the Messages pane. This is because we still have some work to do!
40. Click on the "Signals & Interfaces" tab at the top of the Component Editor window. This shows the signals and interfaces in your top level component, and what Qsys thinks they are for relative to the Avalon bus interface. Most of these guesses are wrong, hence the errors in the Message window. Under the clock[1] signal, click on "add interface". Select "new Clock Input". The entry will change to "clock sink". Drag the clock[1] signal to move it under the clock sink interface. In the right hand part of the window, change Signal Type by selecting "clk". Drag the resetn signal from the Avalon slave 0 interface to be under the "clock reset" interface. Change the Signal Type to "reset n".

Connecting the Custom Hardware in Qsys

41. For your 7-segment display output signals, change the signal type to "Conduit".
42. There will still be a few remaining error messages. On the left, click on "avalon slave 0", and under "Associated Clock" select "clock sink" and under "Associated Reset" select clock reset. Finally, click on the "clock reset" interface. Then set its Associated Clock to "clock sink". All the error messages should now be gone!
43. Click Finish and choose "Yes, Save" to save the qsys related component description files. The new component will now show up in the IP Catalog, in the Project group.



Creating the VHDL Components

44. We can now add the new component to our Qsys system, just like we did with the other components earlier. Connect its ports just as you did for the PUSHBUTTONS component.
45. The address map will now overlap the other component addresses, so we have to redo the /System/Assign Base Addresses command. The addresses may have changed for the components, so you should make sure to check what the new addresses are so that you can change your C-code if needed. Note the memory address range for the new component. It actually has a 10-bit address range (0000-03FF) as the address indicates 8-bit words, whereas the data width is 32-bits.
46. Export the four 7-segments signals. You will need to add connections to these pins in the pin planner after compilation in Quartus.

Generating the new VHDL

47. Re-run the Generate HDL in Qsys. Then you can exit Qsys.
48. Open the Intel FPGA Monitor Program. Modify the C-program test file (save it under a new name) to test the functionality of your new component. Keep in mind that the base addresses will (probably) need to be changed. Also remember that the addresses in the c-program must be in multiples of 4. The address passed to your component will be (c-code address base address)/4. If the address is not a multiple of 4 the HPS will likely hang.
49. Open the Intel FPGA Monitor Program. Modify the c-program test file (save it under a new name) and test the functionality of your new component by sending some test values (you should try the same values you used in your functional simulation and board tests in the earlier labs).

If you finish the lab early you can try and change your custom component and/or the C-program to do other actions. For example, make a timer and a counter to sequence through all 256 possible values.

Writeup the Lab Report .

Write up a report describing the ***Qsys system design***. This report should be submitted in pdf format.

The report must include the following items:

- A header listing the group number, the names and student numbers of each group member.
- A title, giving the name (e.g. ***gNN_Qsys_System***) of the circuit.
- The Qsys block diagram of the final custom system. (Systems Content pane)
- Listing for the c-program for the test of your ARCCOS_DISPLAY circuit.
- Results obtained for some test cases using the c-program to validate the system.

The report is due on Thursday April 13, at 11:59 PM (as this is the last day of classes there will be no extensions).

Submit the Lab Report to myCourses .

The lab report, and all associated design files (*the design files are the vhdl (.vhd) files, as well as the Qsys related files and c programs*) must be submitted, as an assignment to the myCourses site. Only one submission need be made per group (all students in the group will receive the same grade).

Combine all of the files that you are submitting into one *zip* file and name the zip file gNN_LAB_3.zip (where NN is your group number).