EG3205 Part II. Multiprocessor and Multicore Systems

Laboratory Exercise 1

Contents

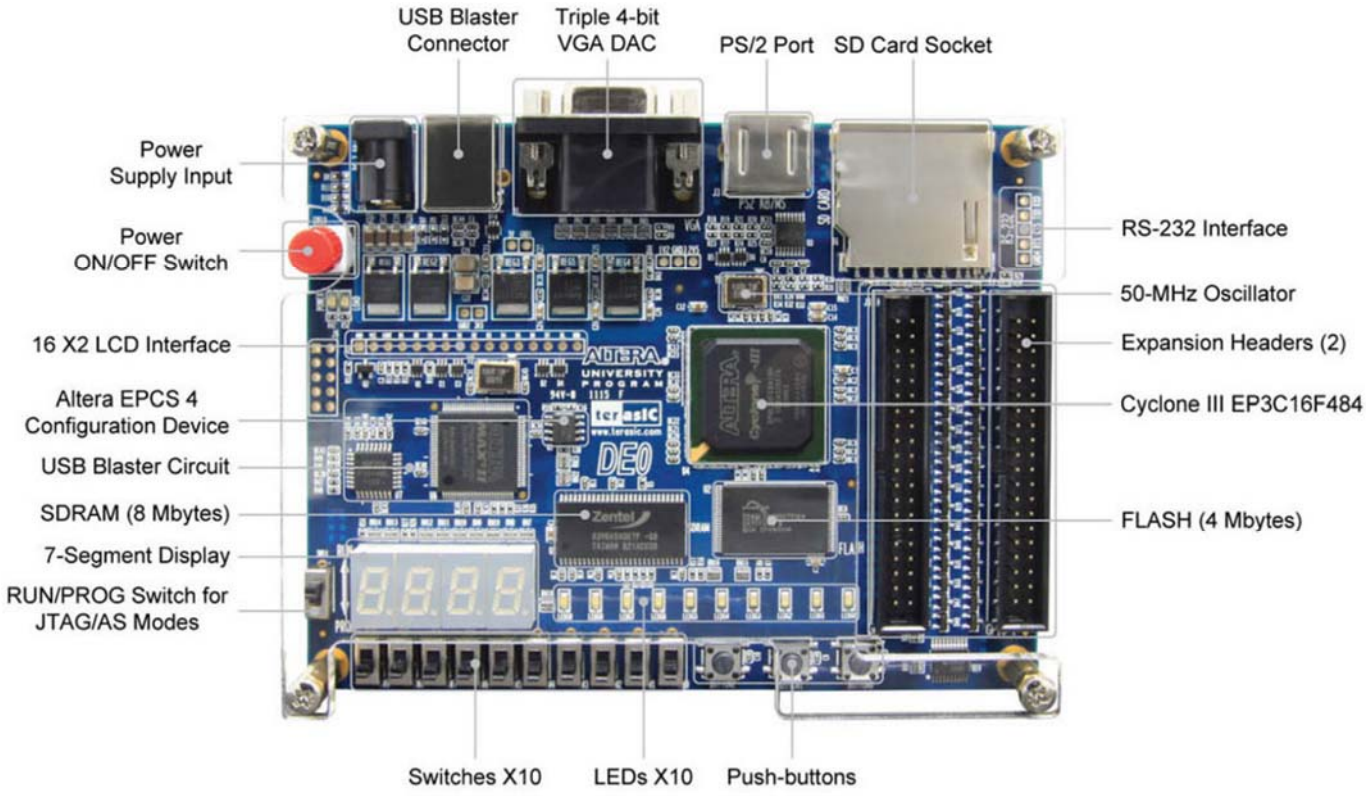
Introduction to Lab 1 1
Description of Lab 1 2
Implementing a simple Nios II processor 3
Programming your simple processor ("Super Loop" architecture) 12
Demonstrate Understanding 16

Introduction to Lab 1

The purpose of this first lab is to familiarise yourself with

- Field Programmable Gate Arrays (FPGAs),
- the DE0 FPGA prototyping board,
- the Intel ECAD Quartus II software suite of tools,
- the Nios II processor design environment.

Your goal is to implement a Nios II "soft" processor (IP core) using Quartus II ver. 13.1 and to run a Superloop scheduler on the Cyclone III FPGA of the DE0 board.



Description of Lab 1

You are expected to complete the tasks below using a provided C-code for a Superloop scheduler:

Task 1:

Familiarise yourself with the Intel /Altera Nios II design environment, which supports the hardware and software design process of a Nios II based programmable system-on-a-chip system.

You will achieve this by following the step-by-step guide below.

Use the provided `basic_nios_II_flashing_led_superloop` scheduler that is available on Blackboard.

For further information on the Qsys integration tool and the Nios II processor refer to the Altera manuals provided in the Supplementary reading folder on Blackboard.

Task 2:

Design the hardware configuration of the Nios II based system using the Qsys integration tool. Use the "Implementing a simple Nios II processor" walkthrough

Task 3:

Implement the provided Superloop scheduler via the Eclipse based Software Build Tool (SBT) on the DE0 FPGA board and demonstrate correct operation. Use the "Programming your simple processor" walkthrough.

Task 4:

Demonstrate understanding of the previous tasks by briefly writing in a word document:

- An explanation of the tasks performed by the various components in your Qsys design
- An explanation of the functions performed by the Superloop code, including the main functions, inputs and outputs
- About how the code to points to the correct pins on the DE0 board

Note: Please make sure that you create your Quartus II project in a root directory and name it Suitably, e.g. "NiosCore". Z:\NiosCore. It is important that your directory does not contain spaces, this will matter later on, use underscores instead.

Files provided:

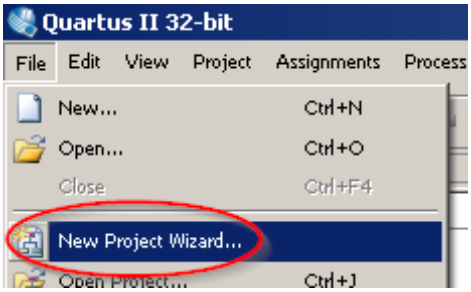
Please download the `basic_nios_II_flashing_led_superloop.zip` file from blackboard.

Acknowledgements:
The first version of this exercise was created by Dr Keith Athaide (March 2012).
The second version of this exercise was developed by Dr. M. Fayyaz(June 2015).
This version of the exercise was developed by Sam Kennedy (October 2018).

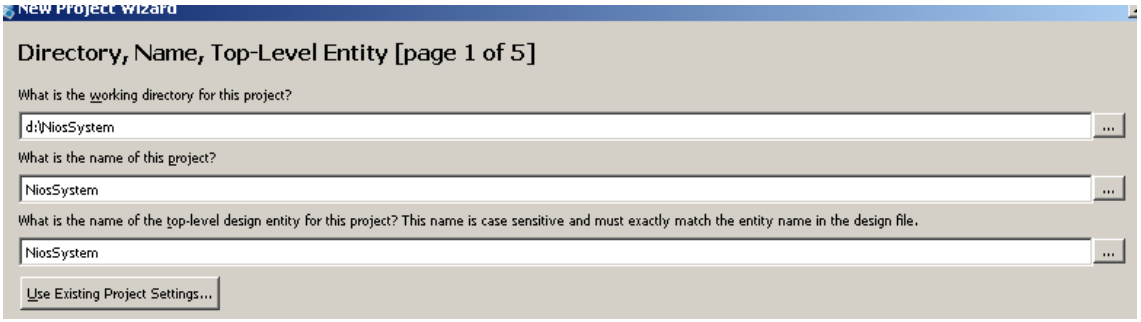
Implementing a simple Nios II processor

Quartus IDE

1. Open Quartus II 13.1 and create a new project by selecting File > New Project Wizard.



2. Enter an appropriate working directory (all project files will be stored in this directory) and project name and click “Next”.

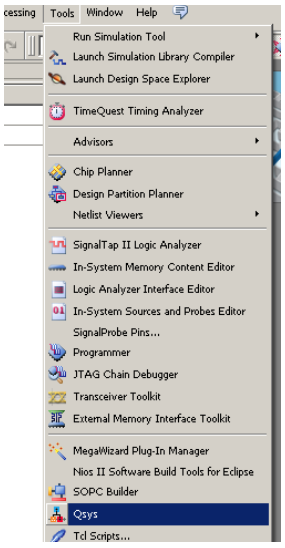


3. Skip past the “Add Files” page by clicking “Next”.

4. Select the FPGA device on the DE0 development board by first setting the family to “Cyclone III” and then selecting the EP3C16F484C6.

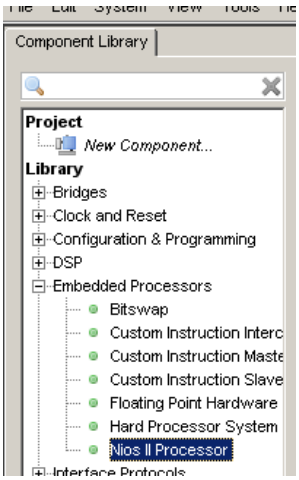
5. Click the “Finish” button.
6. Open the Qsys designer from Tools > Qsys.

Qsys

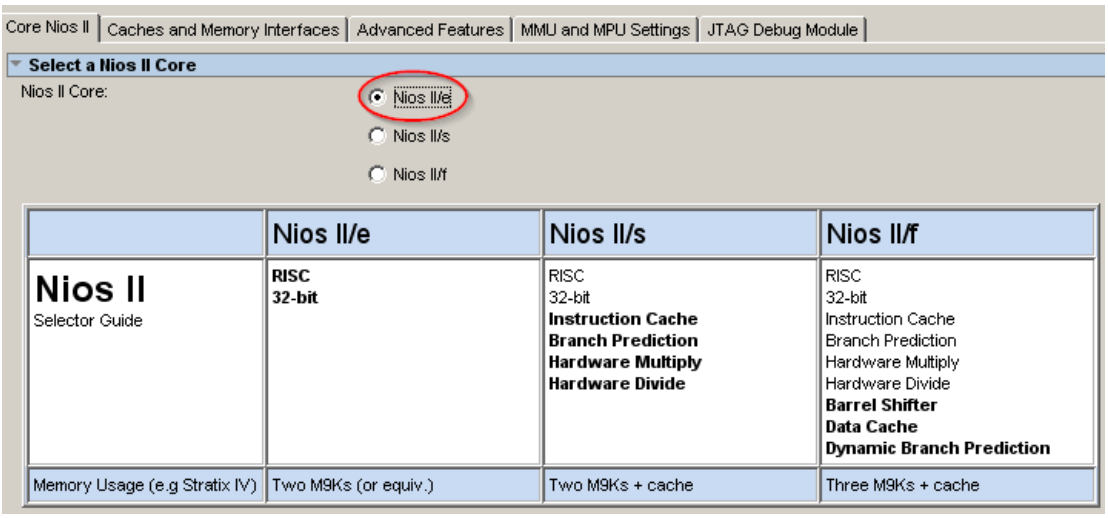


7. Add a Nios II Processor by double clicking on Nios II Processor under “Embedded Processors” in the Component Library.

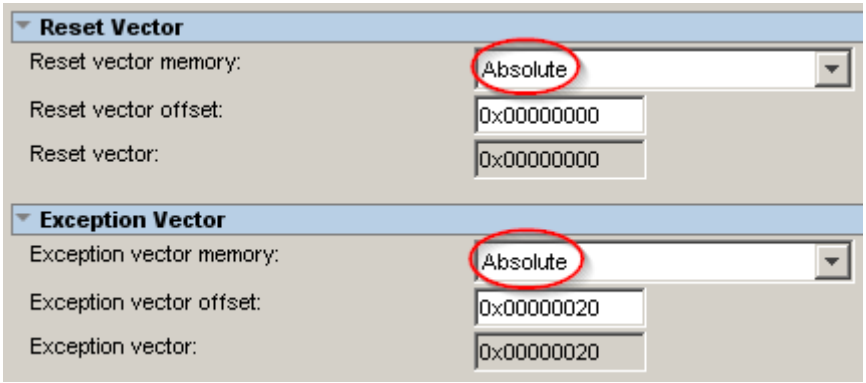
[The component's context menu (right-click) provides access to the relevant datasheet.]



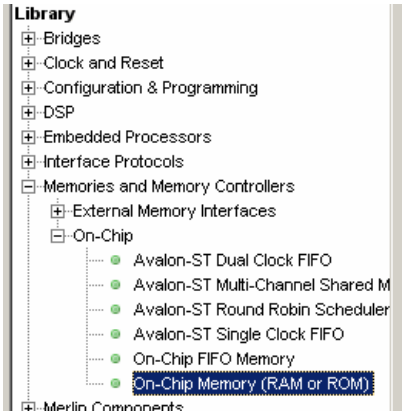
8. In the configuration dialog, select the Nios II/e core.



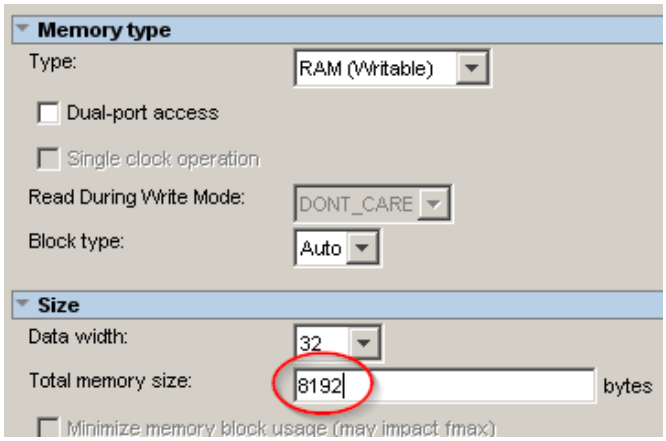
9. Scroll further down, set both the Reset & Exception vector memories to "Absolute" and click "Finish".



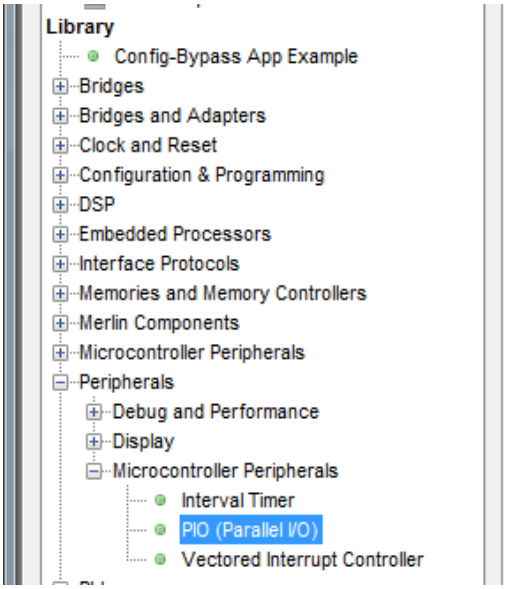
10. Now add some on-chip memory to store program data for the Nios II core.



11. Set the size of the memory to 8192 bytes and click “Finish”.

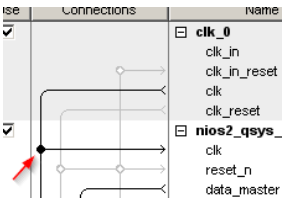


12. Add a PIO (Parallel I/O) microcontroller peripheral.

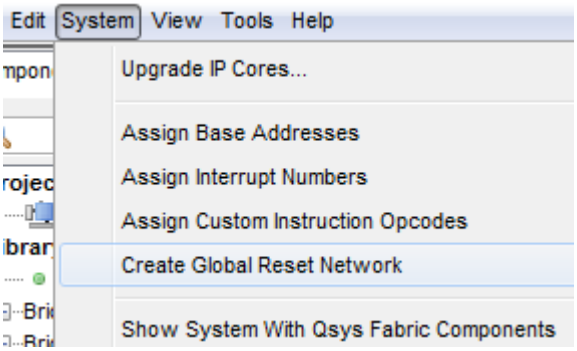


13. Accept all the defaults by clicking “Finish” (this will give you an 8-bit port).

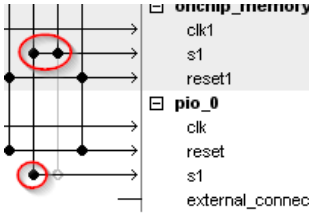
14. Connect the clocks of the Nios II core, the PIO peripheral and the on-chip memory by clicking at the appropriate points in the “Connections” column.



15. Connect the resets by choosing “Create Global Reset Network” from the “System” menu.

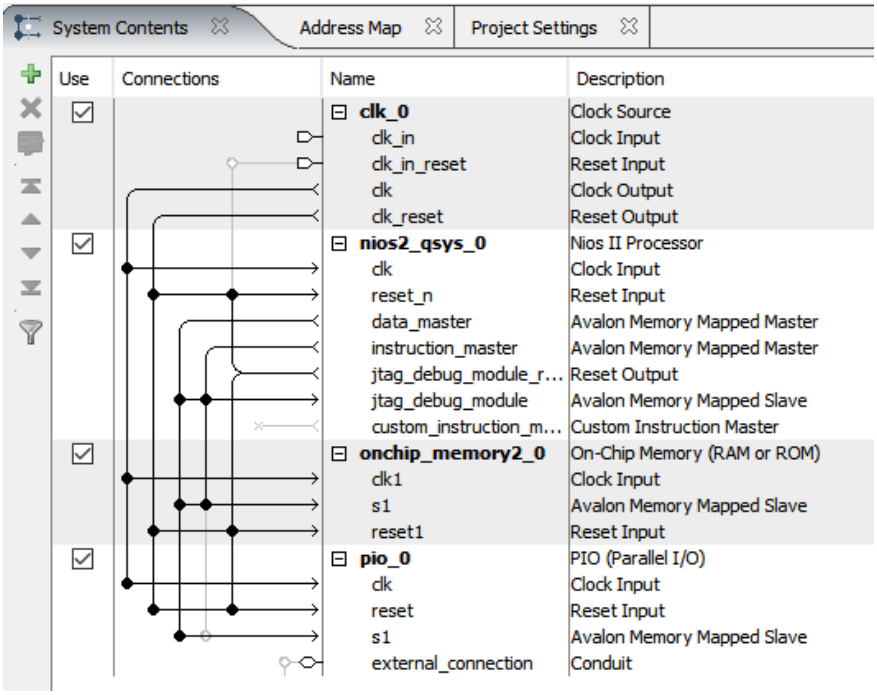


16. In a similar manner to the clocks, connect the data master of the Nios II core to the PIO peripheral and to the on-chip memory;

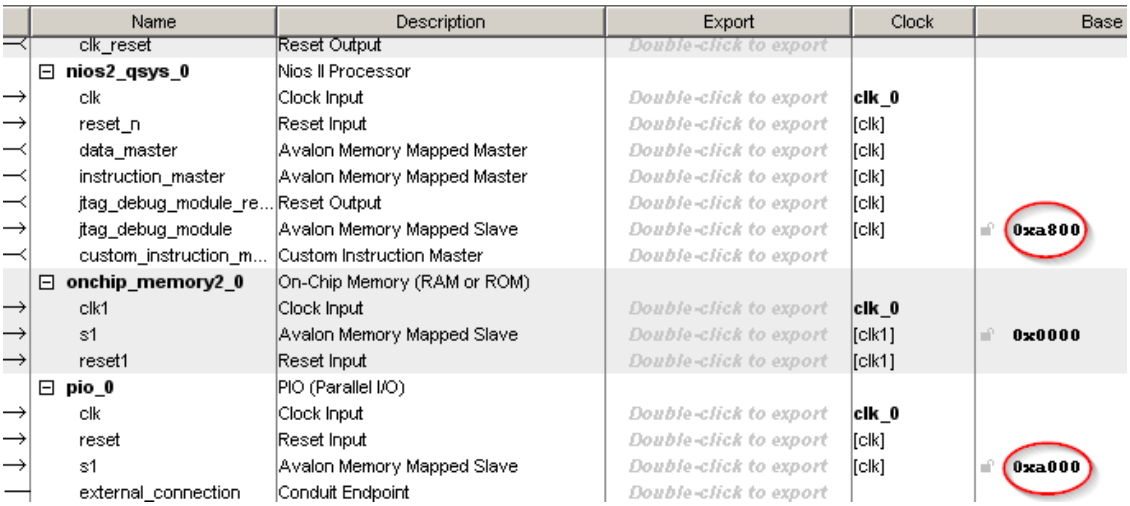


17. Do the same to connect the instruction master of the core to the on-chip memory (only).

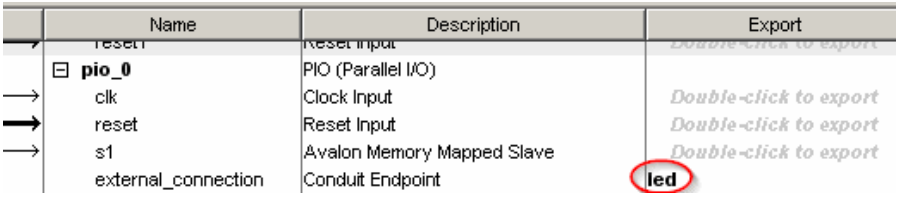
    Your connections should now look like this:



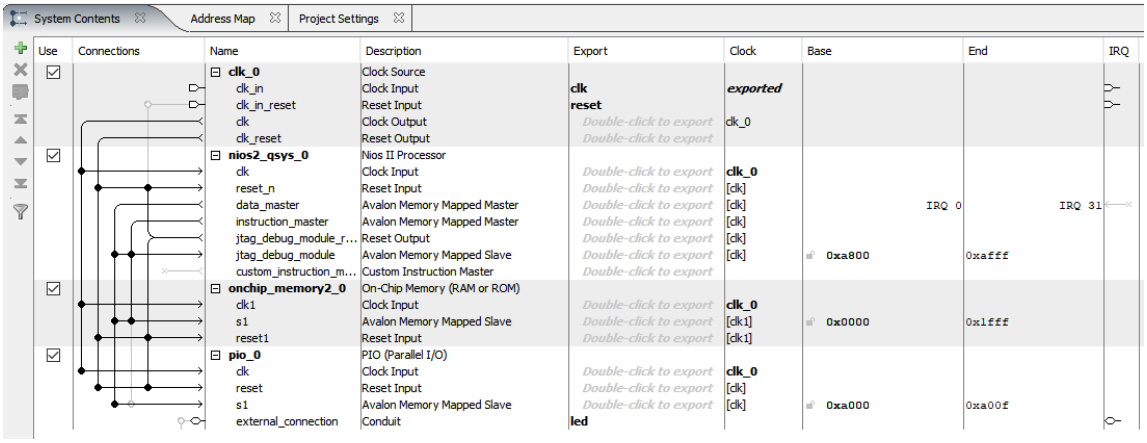
18. Set the base address of the “jtag_debug_module” on the Nios II core to 0xa800 and that of the PIO peripheral to 0xa000 to avoid address space conflicts with each other and the on-chip memory.



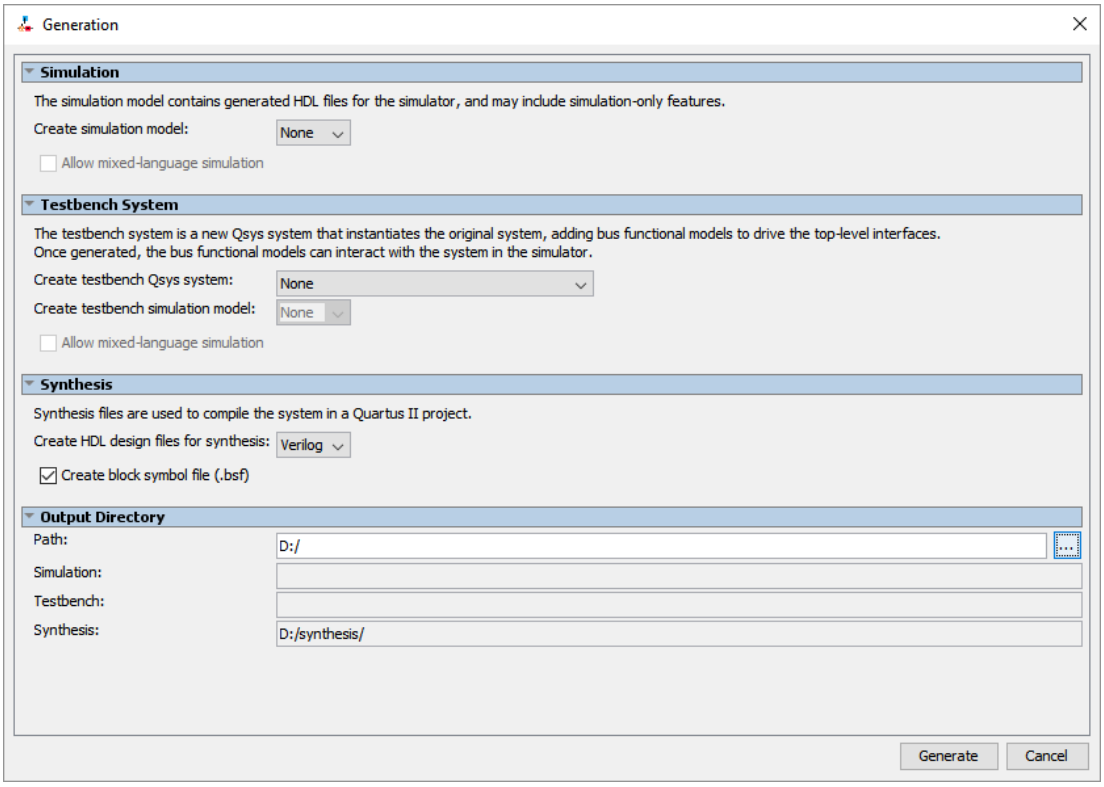
19. Export the “external_connection” of the PIO peripheral by double-clicking in the “Export” column and typing “led”.



20. Save the Qsys file as “nios_system”. Your final design should look as the image below

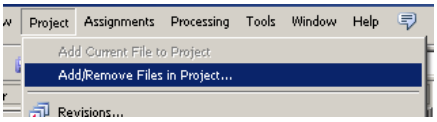


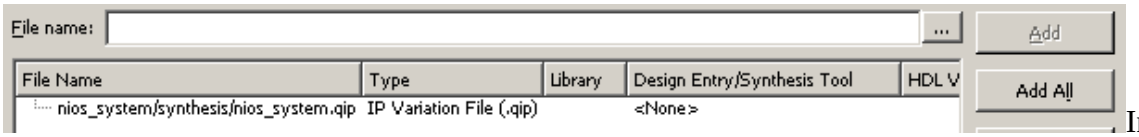
21. Switch to the “Generation” tab and click “Generate”. Qsys will now generate the files required to synthesise the system and the process should complete with neither errors nor warnings.



Quartus IDE

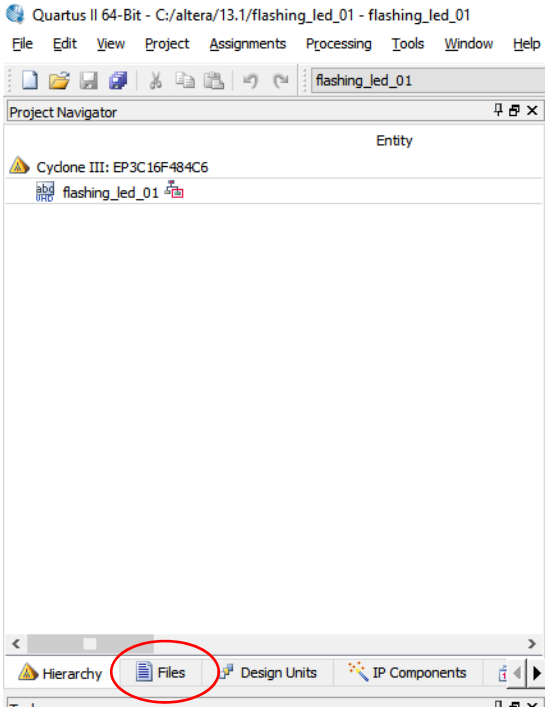
22. Switch to the Quartus IDE and add the QIP file generated by Qsys by selecting “Add/Remove Files in Project” under the “Project” menu. The file will be under “nios_system/synthesis”. The file type in the “Select File” dialog may have to be set to “IP Variation Files”.

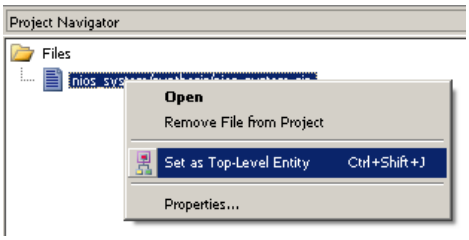




In Quartus, files are added by first browsing for them with the "…" button and then using the "Add" button so that they appear in the list. Then select "OK" from the bottom right to close the window and return back to quartus

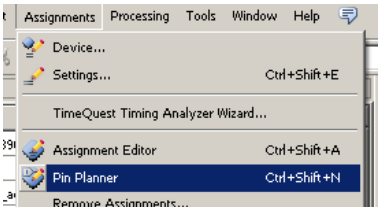
23. In the "Project Navigator" box on the top left of Quartus, select the "Files" tab, right-click on the recently selected QIP file and choose "Set as Top-Level Entity".





24. Run Analysis & Synthesis by clicking on the appropriate button on the toolbar. This will allow Quartus to build a default pin assignment configuration that will be used in the next step.

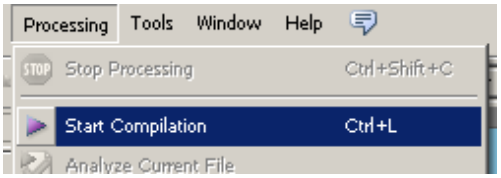
25. Open “Pin Planner” from the “Assignments” menu.



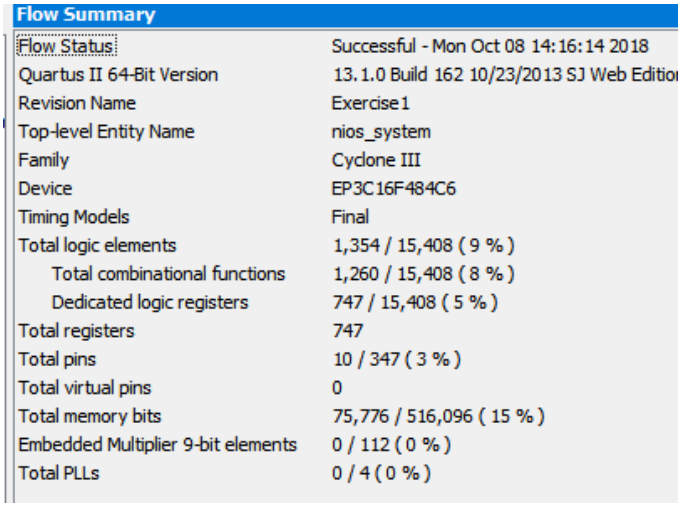
26. Enter locations for “clk_clk”, “led_export” and “reset_reset_n” according to the table below, as extracted from the DE0 user manual (page # 25, 29, & 31) so that the LEDs are assigned to the first 8 green LEDs, the clock to a 50 MHz signal and the reset to the “PUSH BUTTON 3”.

Node Name	Direction	Location
clk_clk	Input	PIN_G21
led_export[7]	Output	PIN_C2
led_export[6]	Output	PIN_C1
led_export[5]	Output	PIN_E1
led_export[4]	Output	PIN_F2
led_export[3]	Output	PIN_H1
led_export[2]	Output	PIN_J3
led_export[1]	Output	PIN_J2
led_export[0]	Output	PIN_J1
reset_reset_n	Input	PIN_F1

27. Close the “Pin Planner” and complete compilation by choosing “Start Compilation” from the “Processing” menu.



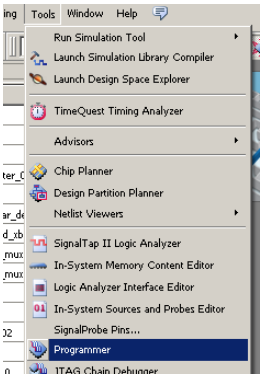
28. The compilation report should display similar to the one below



Flow Summary	
Flow Status	Successful - Mon Oct 08 14:16:14 2018
Quartus II 64-Bit Version	13.1.0 Build 162 10/23/2013 SJ Web Editio
Revision Name	Exercise1
Top-level Entity Name	nios_system
Family	Cyclone III
Device	EP3C16F484C6
Timing Models	Final
Total logic elements	1,354 / 15,408 (9 %)
Total combinational functions	1,260 / 15,408 (8 %)
Dedicated logic registers	747 / 15,408 (5 %)
Total registers	747
Total pins	10 / 347 (3 %)
Total virtual pins	0
Total memory bits	75,776 / 516,096 (15 %)
Embedded Multiplier 9-bit elements	0 / 112 (0 %)
Total PLLs	0 / 4 (0 %)

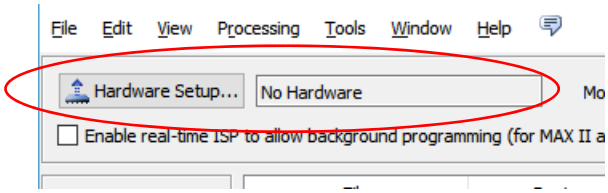
29. Once the compilation is complete, open the “Programmer” from the “Tools” menu. At this point please make sure that your DE0 board is connected to the computer and turned on.

Programmer

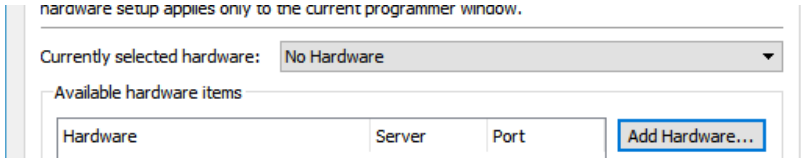


31. Ensure that “USB-Blaster” is selected in the top left next to the “Hardware Setup” button. If it is, skip to step 33.

32. If it reads “No Hardware” click the “Hardware Setup” button to open a new page.

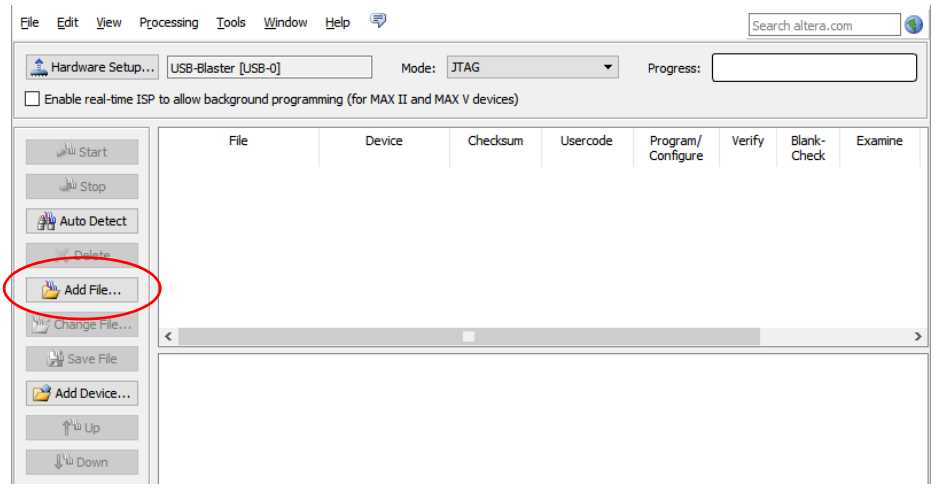


33. Click the drop down next to “Currently Selected Hardware” and select “USB Blaster” and then close the screen to return to the Programmer.

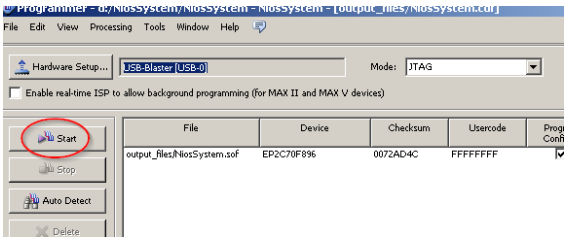


- If there is no hardware available to select in the dropdown menu then check your board is connected and on. If it is then please let a demonstrator know as this means the drivers required are not installed

34. Select “Add File” on the left hand side and locate the .sof file in the output_files folder



35. Press the “Start” button to download the recently compiled RTL to the FPGA development board (volatile version – load the .sof file, leave the board in Run mode).

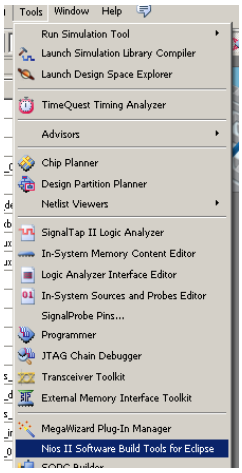


36. If successful, the board should display only a solid blue light power light. This is because there is no software yet running on the processor we have just created.

Programming your simple processor (“Super Loop” architecture)

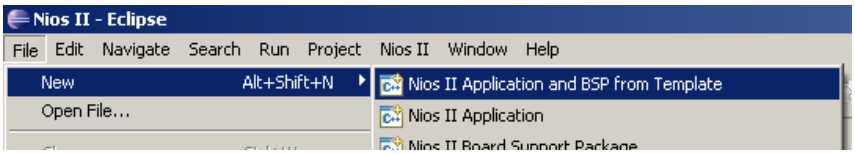
1. Open the “Nios II Software Build Tools for Eclipse” from the “Tools” menu.

Eclipse IDE

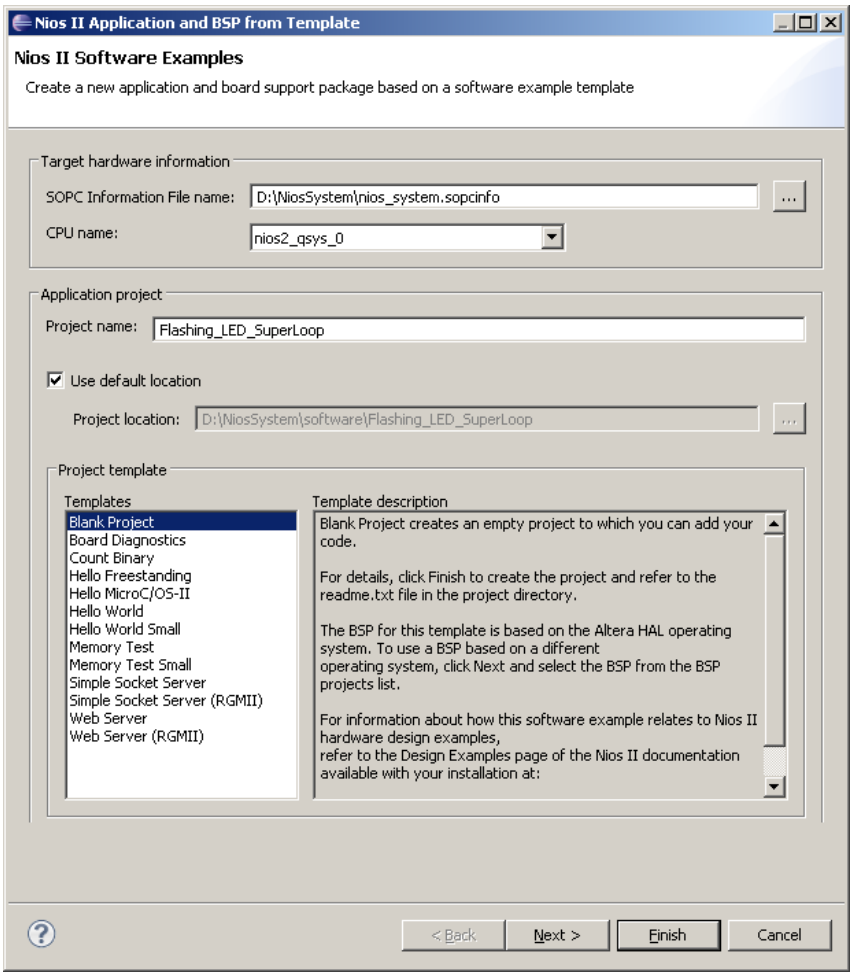


2. When prompted for the workspace directory, pick a “software” folder under your working directory, e.g. “D:\NiosSystem\software”.

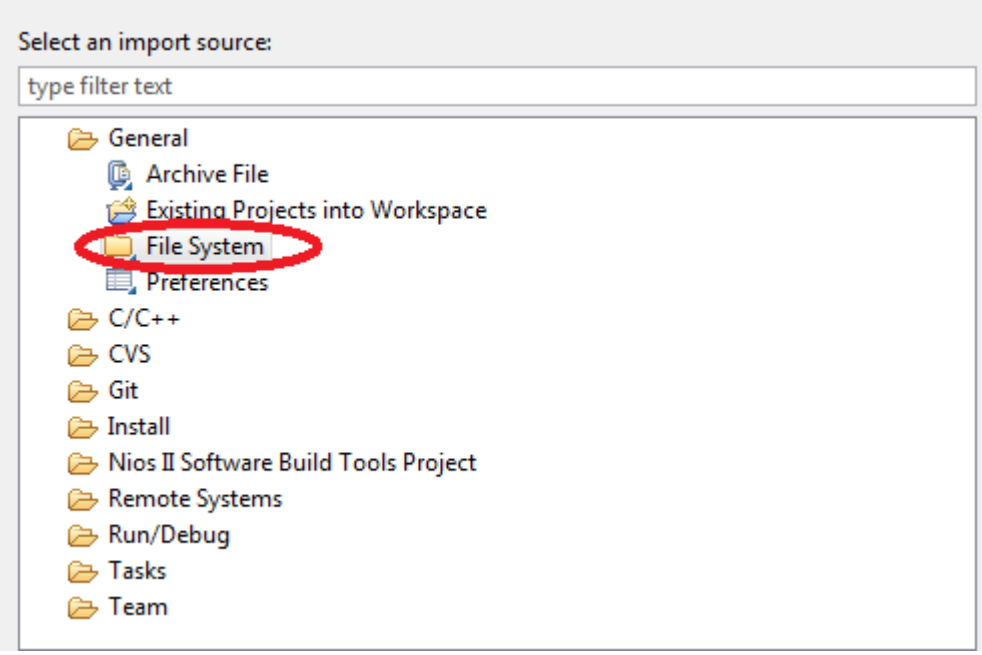
3. Select the “Nios II Application and BSP from Template” from the “New” submenu under the “File” menu.



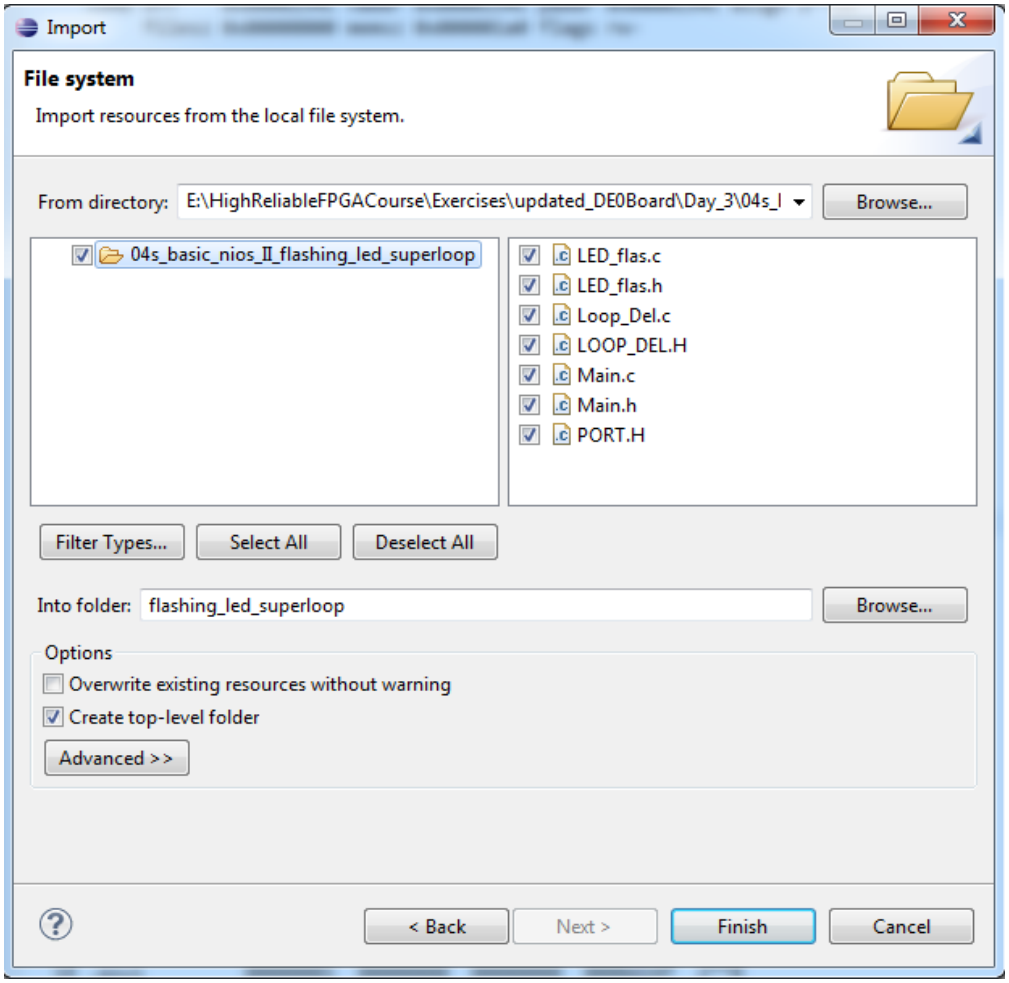
4. The “SOPC Information File name” will be in the working directory e.g. “D:\NiosSystem\software”; use a “Blank Project” as the creation template and name the project accordingly (e.g. “Flashing_LED_SuperLoop”), then click “Finish”.



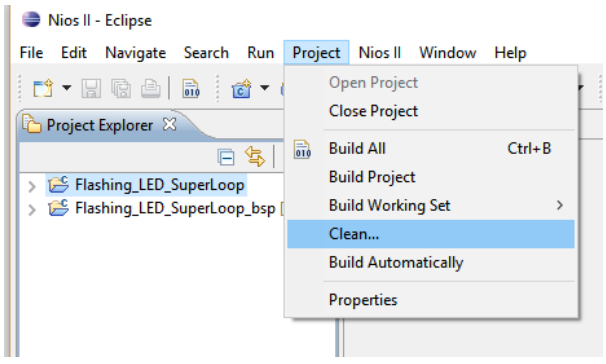
5. Select the “Import” from the “File” menu. Then expand “General” and select “File System”.



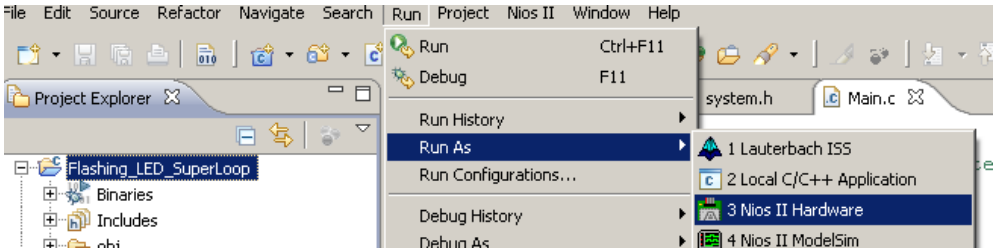
6. A “File System” dialog will open. Click browse to locate the source folder that includes the super loop program files. Click on ‘Select All’ to include all files into the project. Make sure that ‘Create top-level folder’ is checked. Click finish to complete the process.



7. Once the folder is added, clean and build the project.

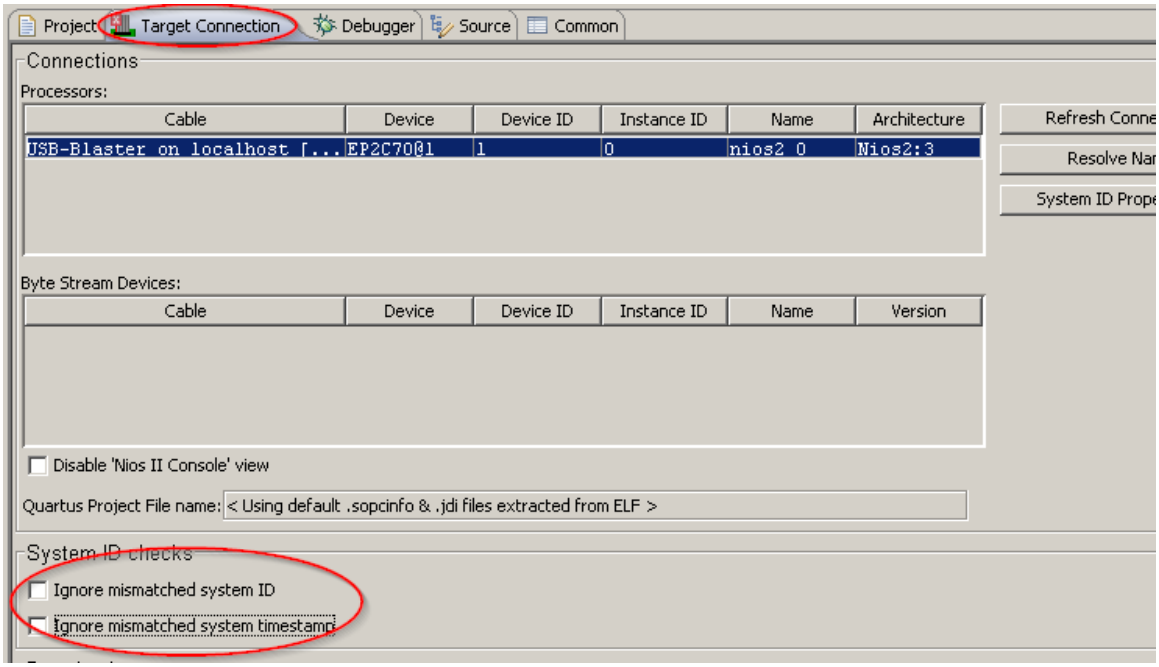


8. Select the project and choose “Nios II Hardware” under the “Run As” submenu on the “Run” menu (the project can be debugged, if needed, by using “Debug As” instead).



The BSP and main projects will be compiled and downloaded to the hardware after which one of the green LEDs will flash.

Note: If you get an error about "Connected system ID hash not found on target at expected base address" while downloading code to the board, then check both "Ignore mismatched system ID" and "Ignore mismatched system timestamp" on the "Target Connection" tab of the launch configuration dialog and retry.



Demonstrate Understanding

Using Eclipse, navigate through the code for the Superloop. Examine the various source and header files. Analyse the functions implemented and the code they contain. What does the Loop_Delay function do? How is the LED turned on and off?

The code is primarily taken from the Patterns for Time-Triggered Embedded Systems book by Michael Pont, which is available on Blackboard.

Prof. Tanya Vladimirova
06/11/2018