

MOOC Version: Migrating the VGA demo in HDL to the Qsys environment with IP

Soft Processor Design for FPGAs

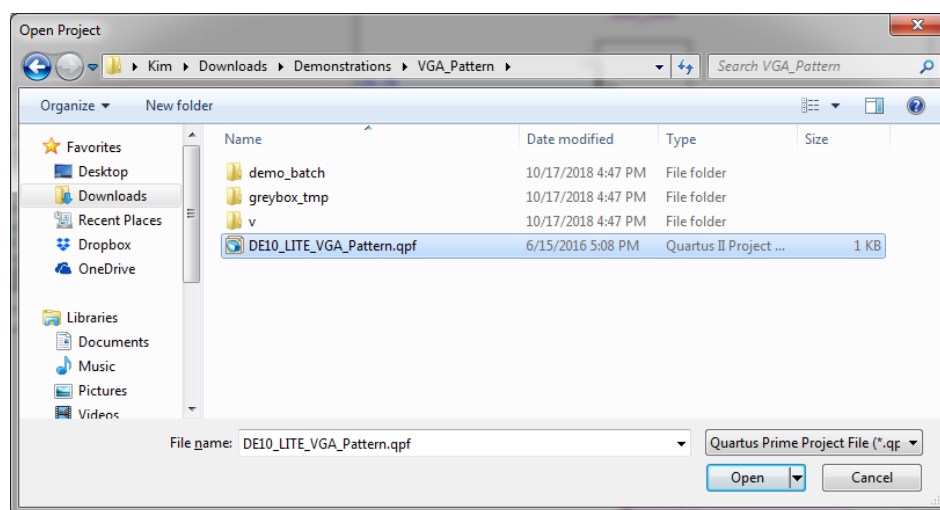
University of Colorado Boulder



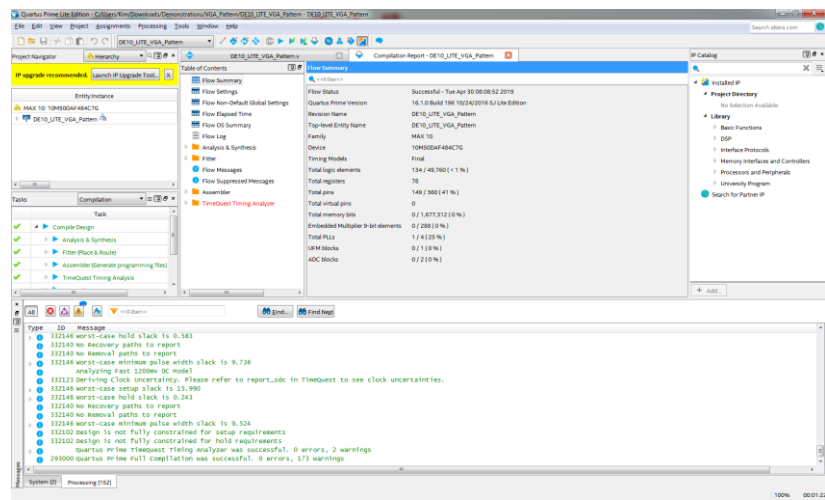
Items needed: DE10-Lite board, USB programming cable, VGA cable

Procedure:

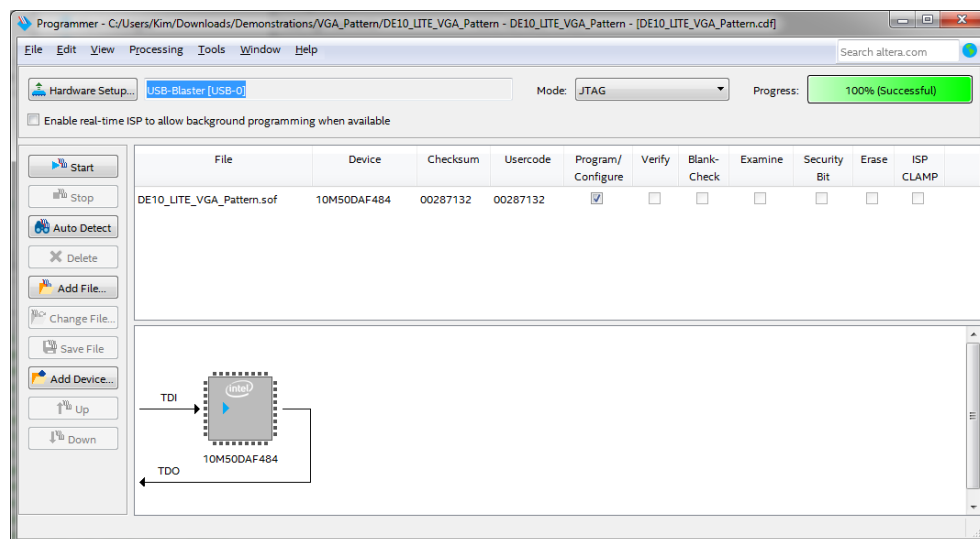
1. Find the VGA demo project in the DE10-Lite demonstrations folder.
2. Launch the project in Quartus.



3. Compile the project and check the report to see the size of the implementation.



4. Program the board and connect the VGA output to a monitor.



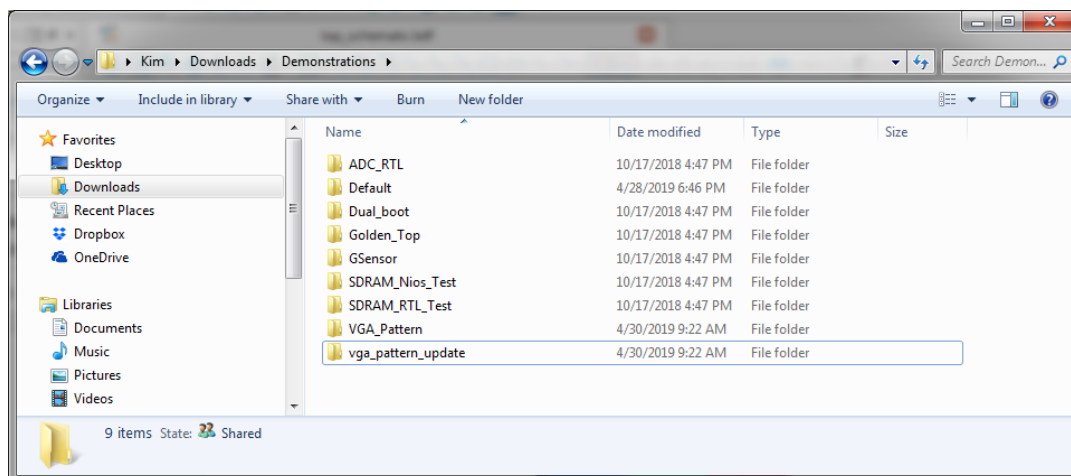
5. The display should look like the image below.



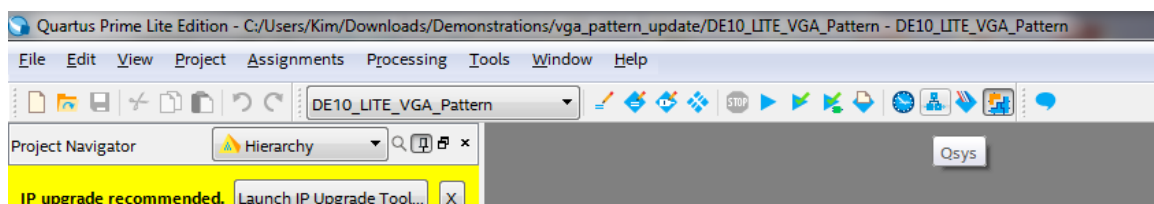
CHECK POINT : POST YOUR PROGRESS FOR THIS PART TO THE FORUMS

6. Next step is to modify the project to implement the design in Qsys using the Test Pattern Generator IP and the Clocked Video Output IP components along with the NIOS II CPU.

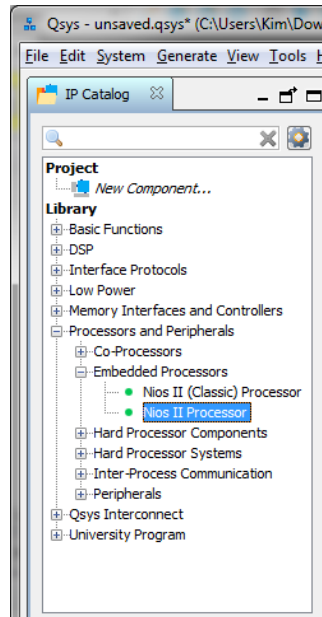
7. Close the Quartus environment and make a copy of the demo into a new folder called vga_pattern_update.



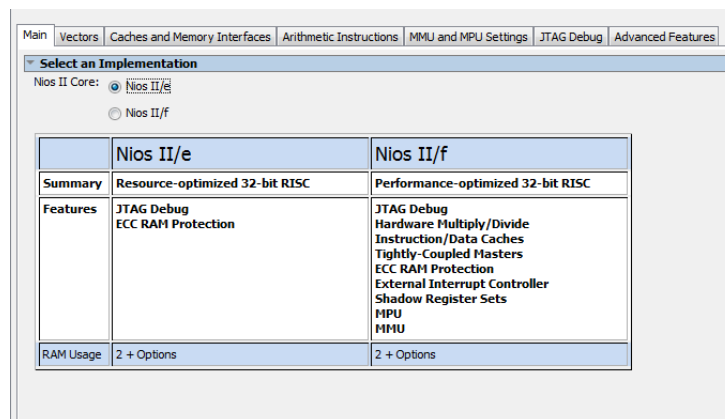
8. Load the project in the new folder and launch Qsys by selecting the icon show in the figure below.



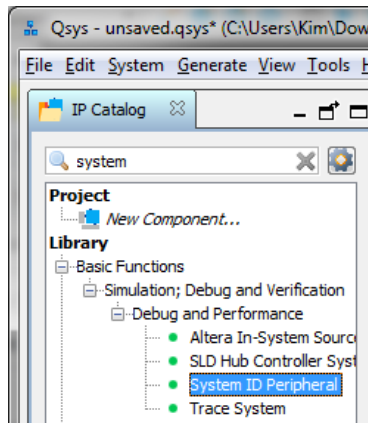
9. Once Qsys is launched, the first step is to define all of the components. Select the NIOS II processor to add to the core. See the selection shown on the left hand side of the tool. Double click to add it to the system.



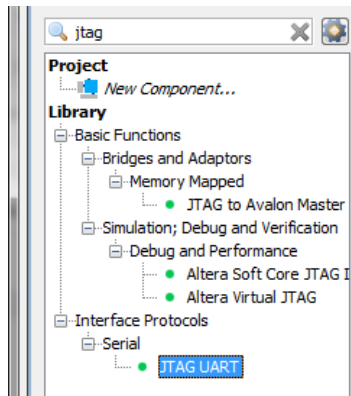
10. Right click on the component to select the edit menu. Select the economy mode on the first setup tab.



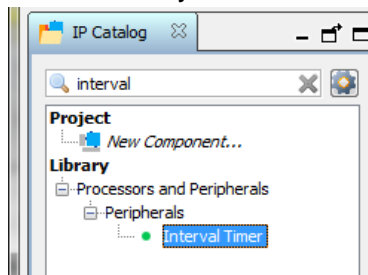
11. Finish the module by renaming it to CPU by right clicking on the component. Add the next component which is the System ID Peripheral. Rename it to sysid.



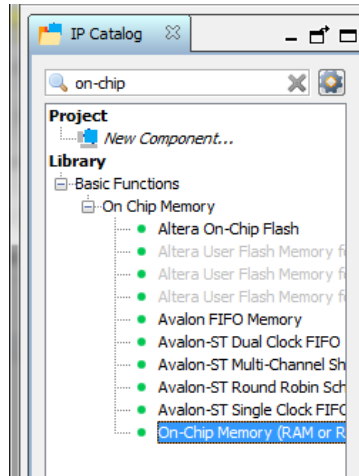
12. Next component is the JTAG UART. Rename it to jtag_uart. Keep the default settings.



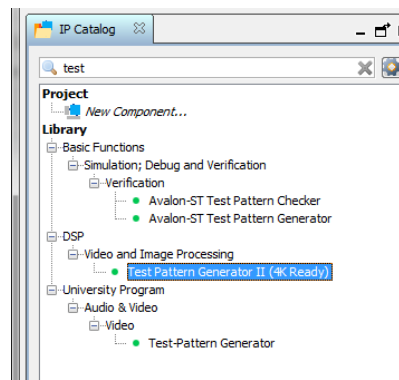
13. Add the Interval Timer next and rename it sys_clk_timer



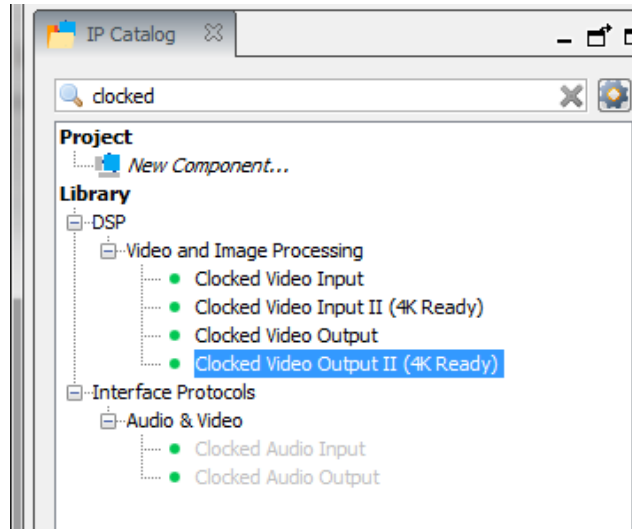
14. Add the On-Chip Memory and rename the component to code_memory. Change the Total memory size from the default 4,096 to 80,000.



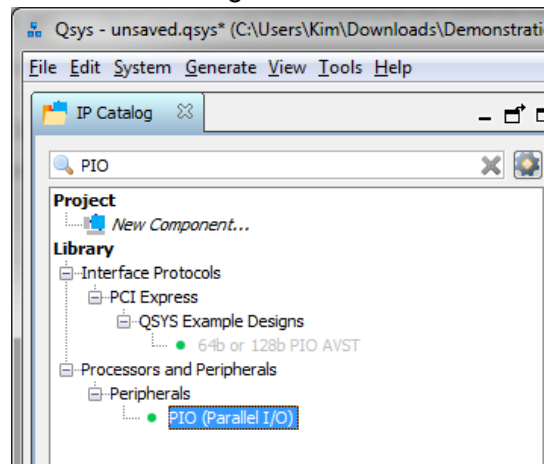
15. Next add the Test Pattern Generator and rename it test_pattern. Configure this module for 4 bits per color sample, run-time control, 640 x 480 frame width and height, RGB color space, 4:4:4 output format, color planes transmitted in parallel, progressive output, 1 pixel in parallel, Uniform background with RGB set for 15 for each value.



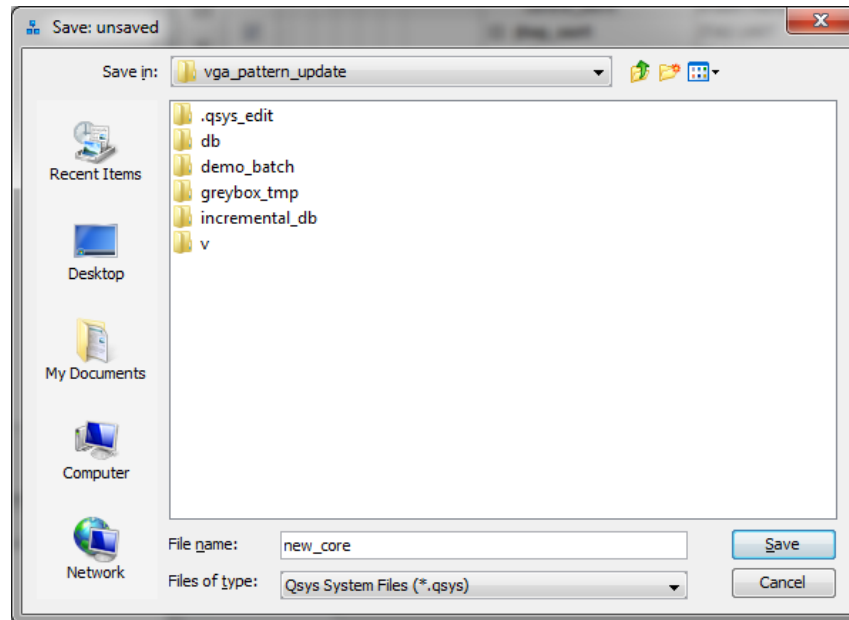
16. Next configure the Clocked Video Output. Rename the component vga_out. Configure the component for 640 x 480 image width and height. Set the bits per color plane to 4 and the number of color planes to 3. Select the color plan transmission format to parallel. Select the sync signals for on separate wires. For Separate Syncs only - Frame / Field 1 set the horizontal sync to 96 pixels, the horizontal front porch to 16 pixels, back porch to 48 pixels, vertical sync to 2 lines, front porch 10 lines, and back porch 33 lines. All other settings in the Syncs configuration should be zero. Set the general parameters for pixel fifo size of 640 and the fifo level to start at 639. Runtime configurable video modes are 1 and the width of the vid_std bus is 1. All other settings in the general parameters are unchecked. Low latency mode is 0.



17. The final component is to add the PIO and rename it switches. Configure the module to a width of 3 and set for input. The other settings will remain the same.



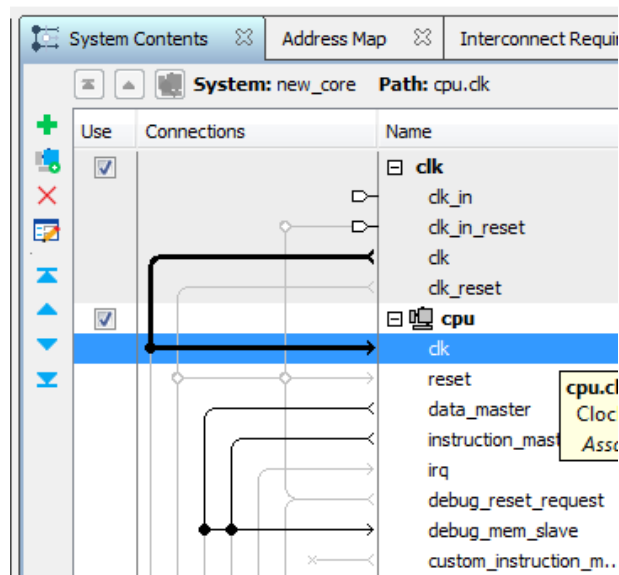
18. Save the system as new_core.



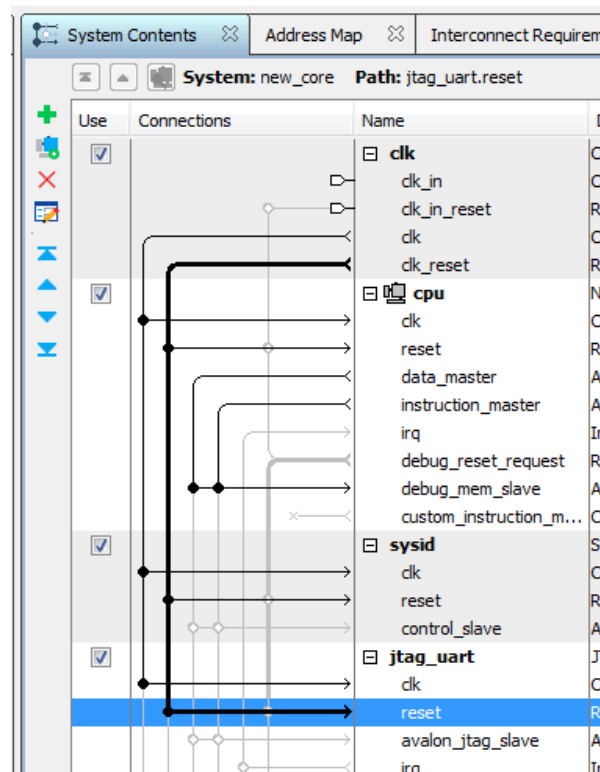
_____ (sign off by student assistant) _____ (date and time)

Part 2:

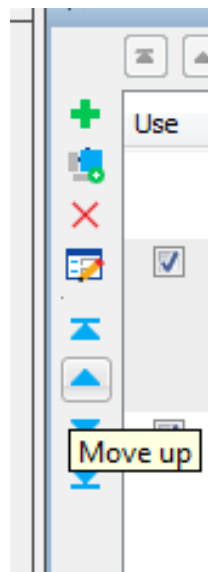
19. Connect the clock and reset signals to each of the components. Click on the connector to attach the clock to each component.

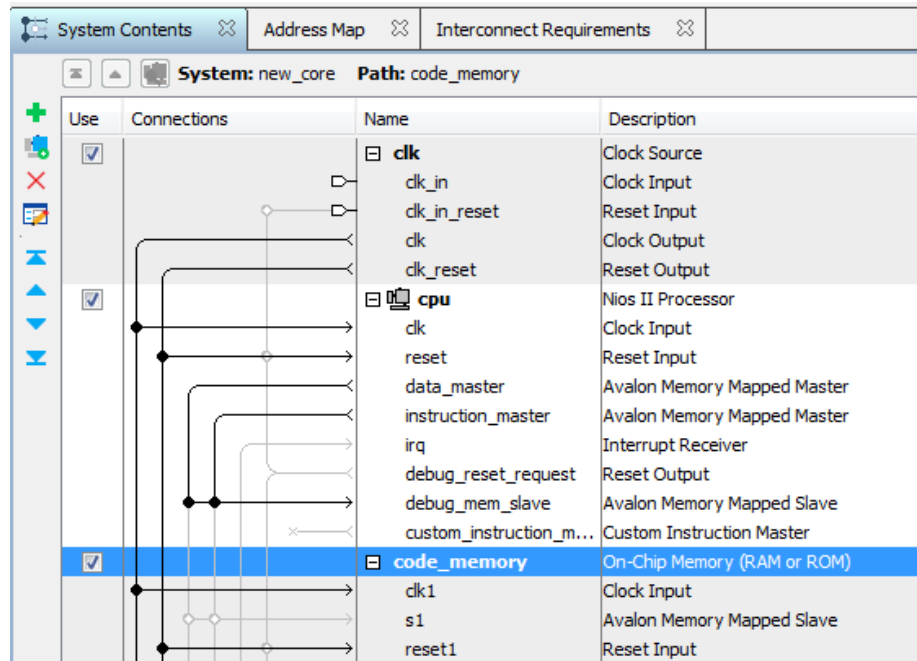


20. Do the same for the reset signal as well.



21. Once all of the components are connected, move the code memory up to right next to the CPU by selecting the component and using the control buttons on the left hand side of the Qsys environment.



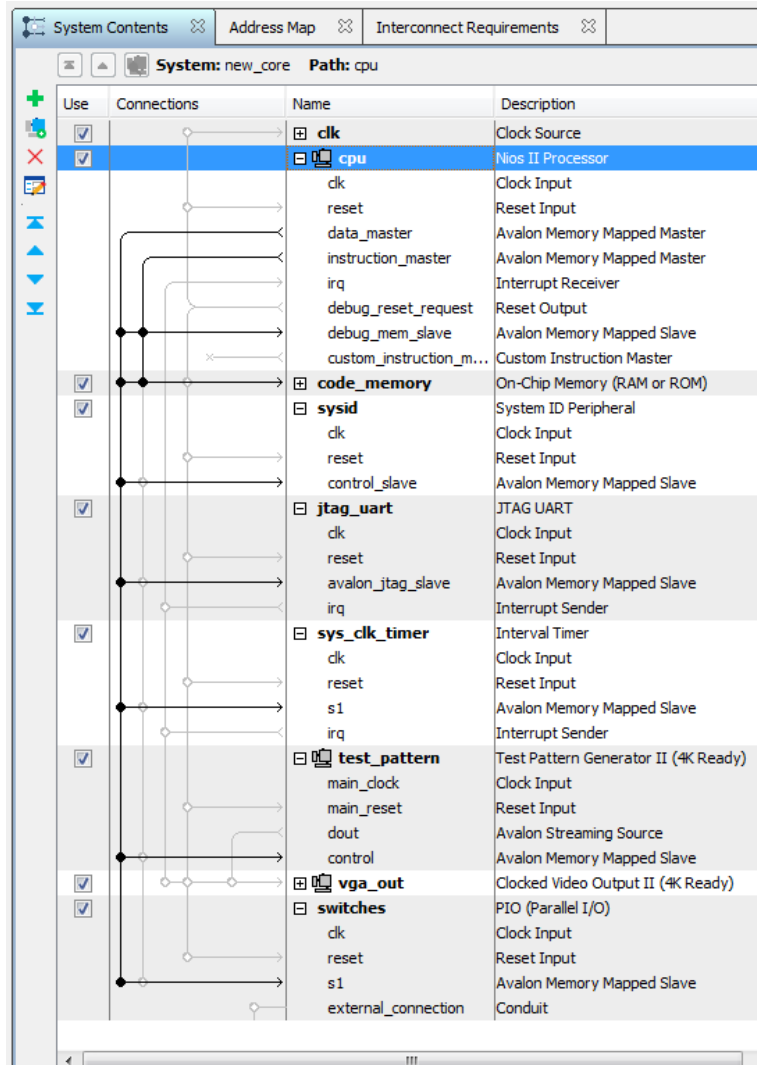


22. Connect the Avalon Memory Mapped Slave on the code_memory to the instruction_master and data_master of the cpu. After the bus signals are connected from the memory to the cpu the vectors can be set.

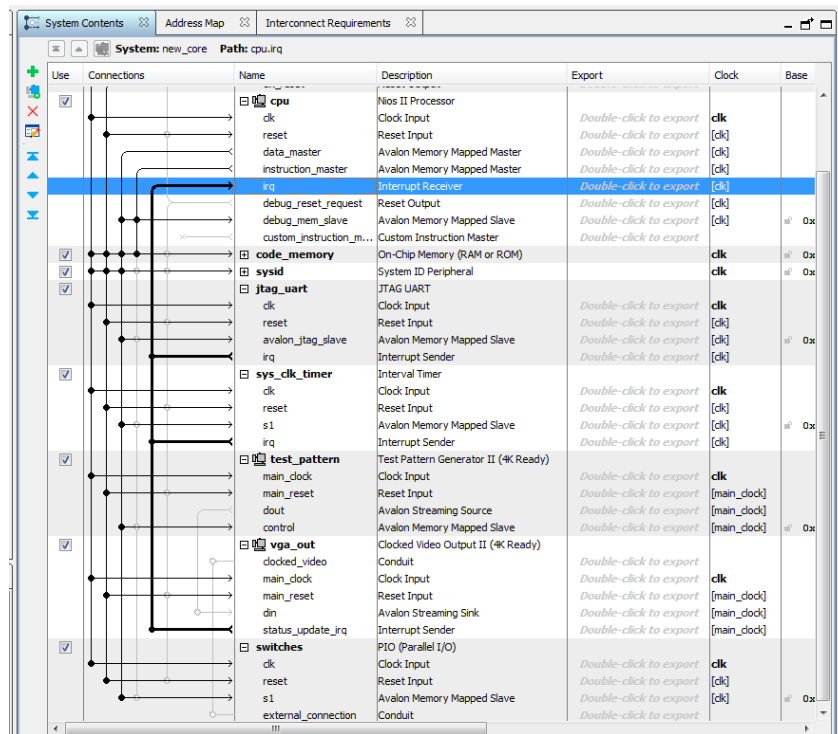
The screenshot shows the 'Vectors' configuration window with the following settings:

Section	Property	Value
Reset Vector	Reset vector memory:	code_memory.s1
	Reset vector offset:	0x00000000
	Reset vector:	0x00000000
Exception Vector	Exception vector memory:	code_memory.s1
	Exception vector offset:	0x00000020
	Exception vector:	0x00000020
Fast TLB Miss Exception Vector	Fast TLB Miss Exception vector memory:	None
	Fast TLB Miss Exception vector offset:	0x00000000
	Fast TLB Miss Exception vector:	0x00000000

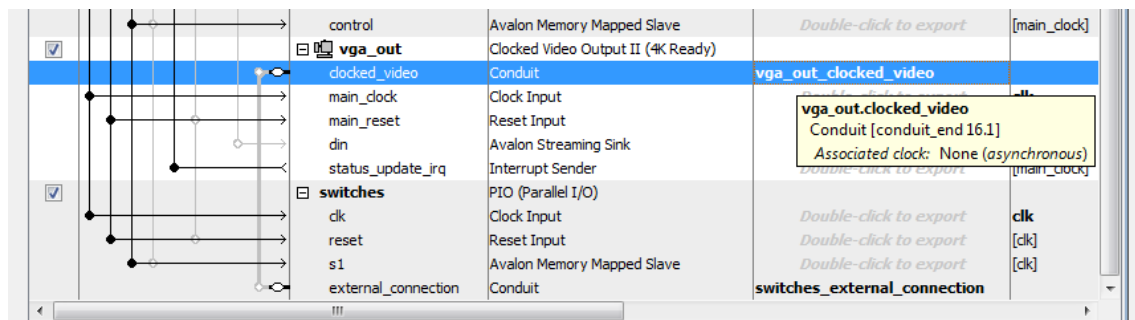
23. The data_master from the CPU should be connected to the sysid component control_slave input, jtag_uart component avalon_jtag_slave input, sys_clk_timer component s1 input, test_pattern component control input, switches component s1 input.



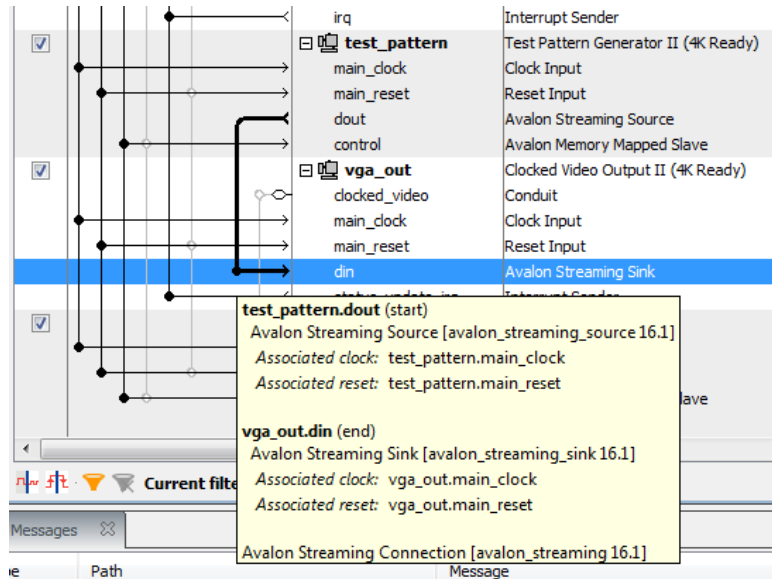
24. Connect the IRQ signals to the CPU for each module.



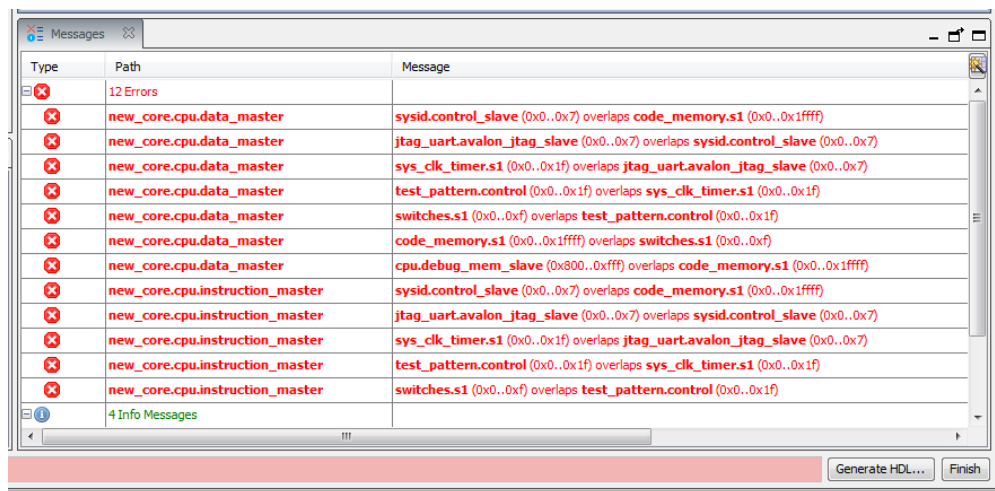
25. Export the signals for the VGA and the switches.



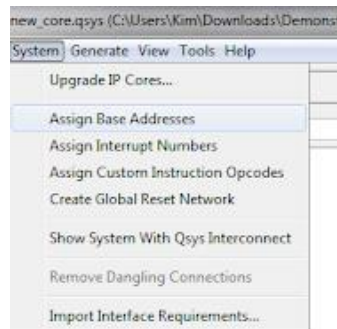
26. Connect the video data path from the TPG to the CVO



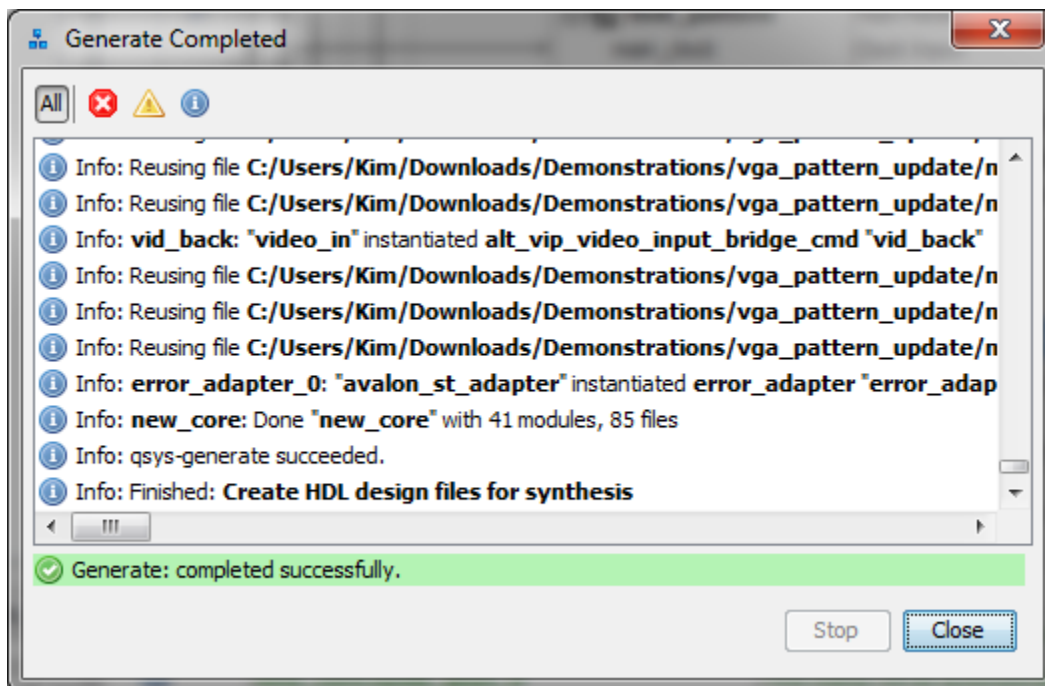
27. Save the new_core file again. There are still errors in the project so the base addresses need to be updated.



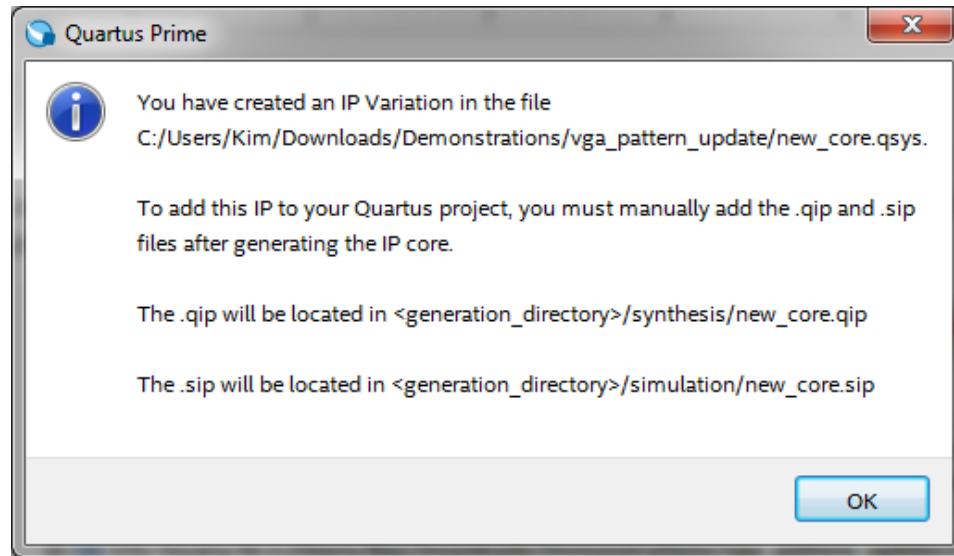
28. Select Assign Base Address from the System menu.



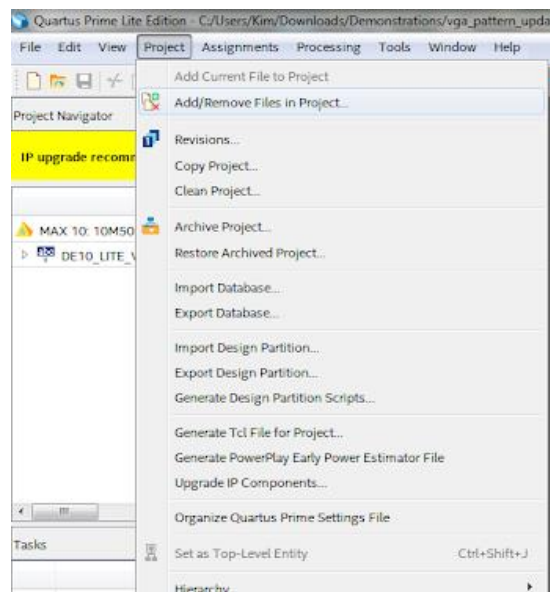
29. The errors should then disappear. Save the project again then select Generate HDL. It should complete with no errors as shown in the screenshot below.



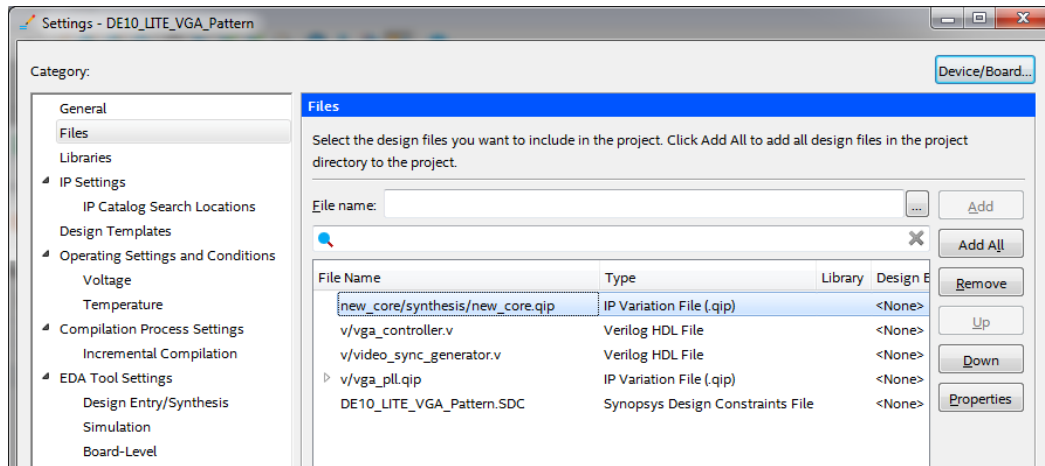
30. The following output will show up after closing the Qsys tool.



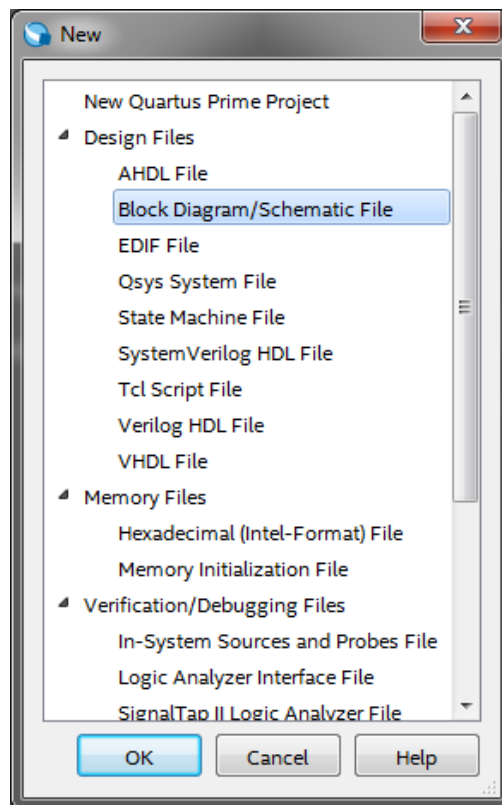
31. Add the .qip file to the project.



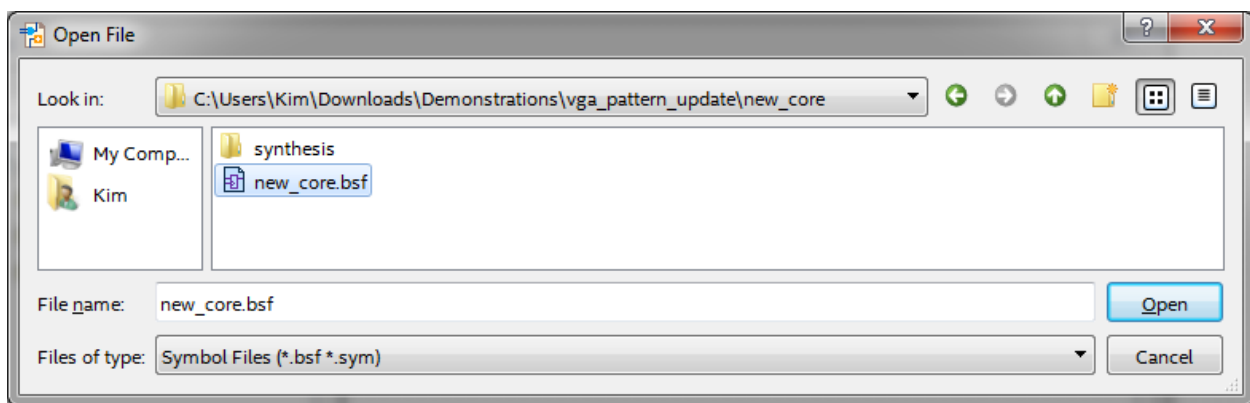
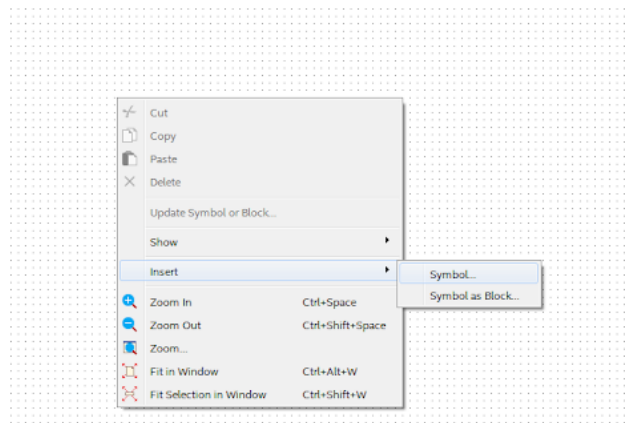
32. From the files menu select the file and add to the project as shown.



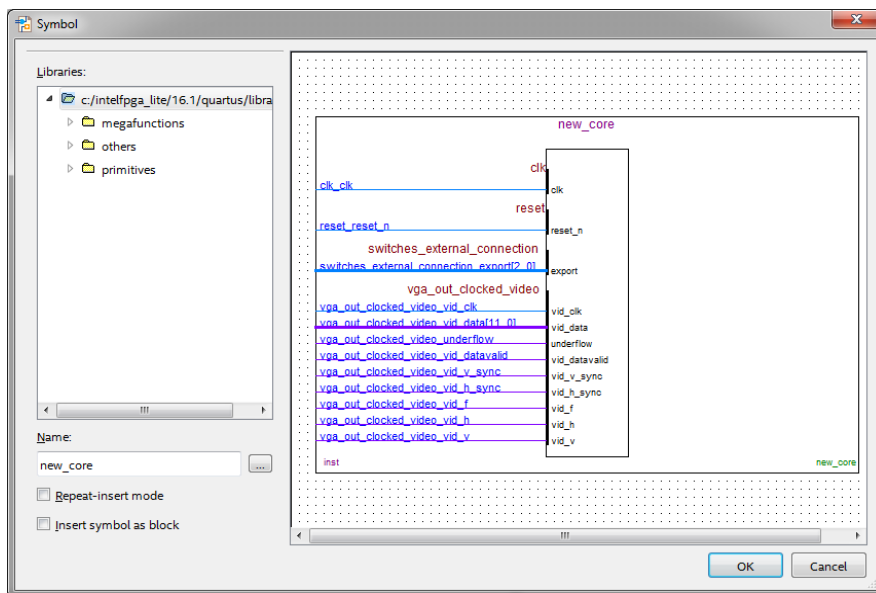
33. Create a new schematic entry file for the project in the top level Quartus environment.



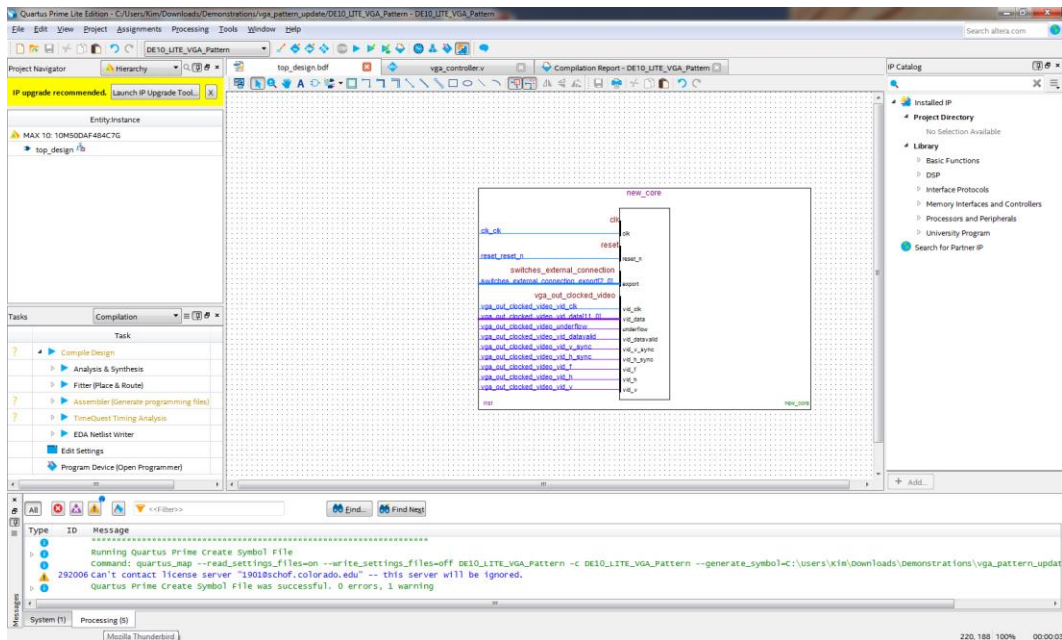
34. Save the file as top_design.bdf. In the project menu add the file to the current project and set it as the top level entity. Right click in the IDE to add the symbol from the Qsys tool.



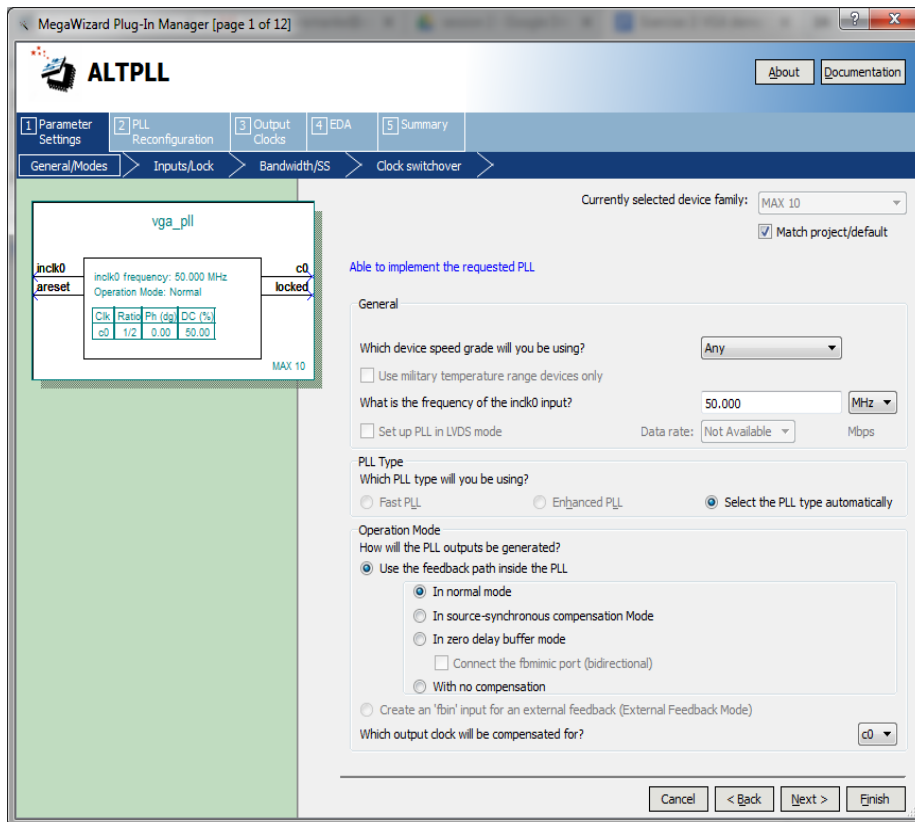
35. The symbol will then appear in the symbol tool



36. Select OK then add it to the schematic file.

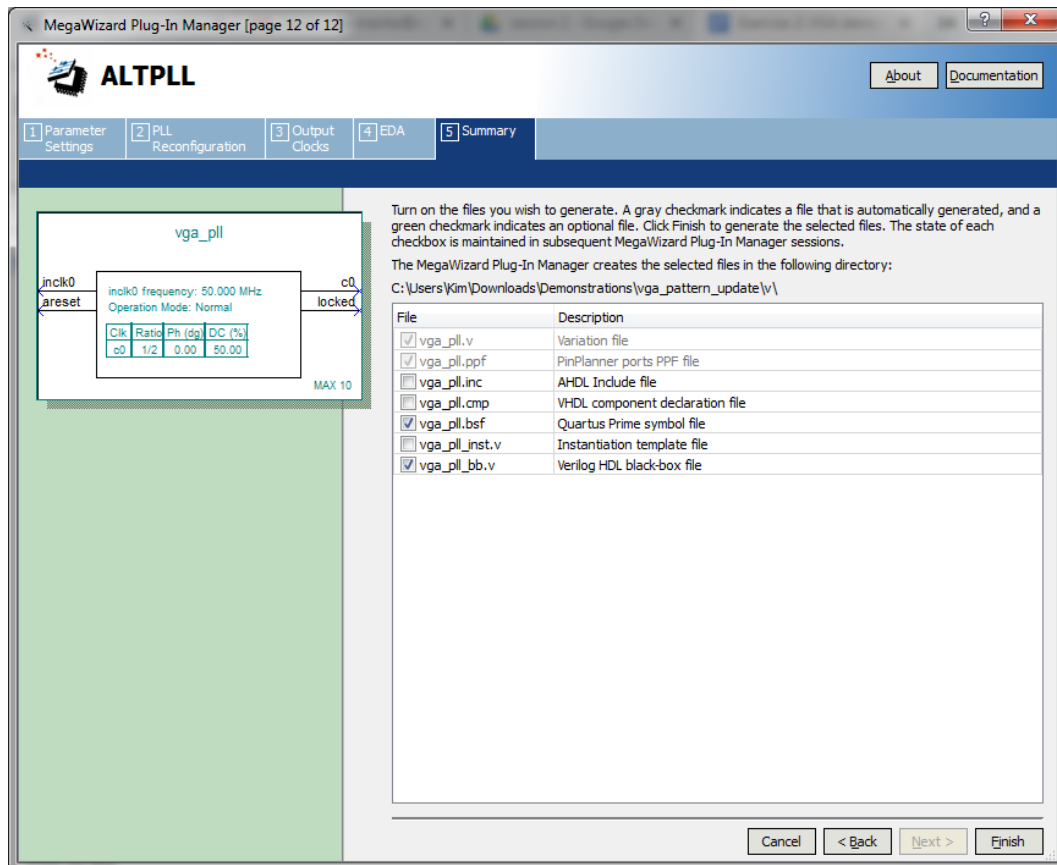


37. The next step is to generate a symbol for the vga_pll input using the MegaWizard Plug-In Manager. Open the vga_pll.v file then generate a symbol file once it is open. It will launch the screen shown below.

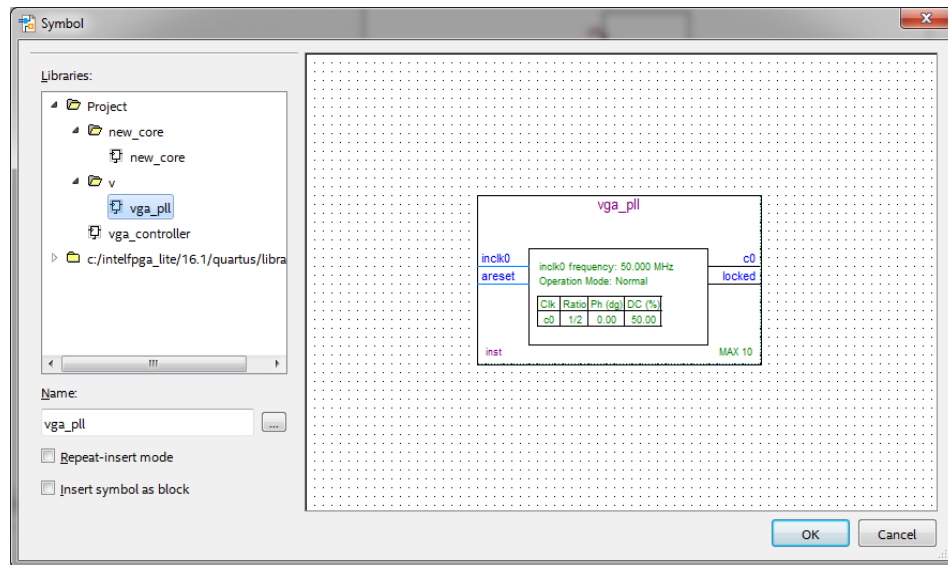


The output clock from the vga_pll is the Pixel Clock for the 640 x 480 VGA signal. It is not the same frequency as the input clock of 50 MHz. When you get to the setting for the Output Clocks, set the frequency to 25.175 MHz. It will round as close as possible to this value when implementing the Pixel Clock signal for the VGA output.

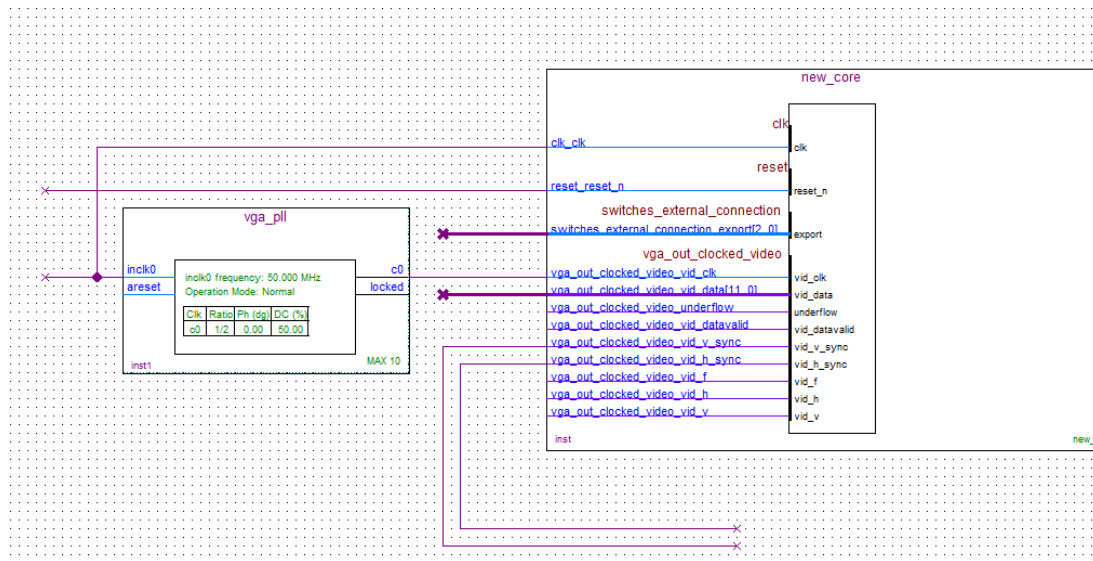
Turn on the Quartus Prime symbol file in the last menu.



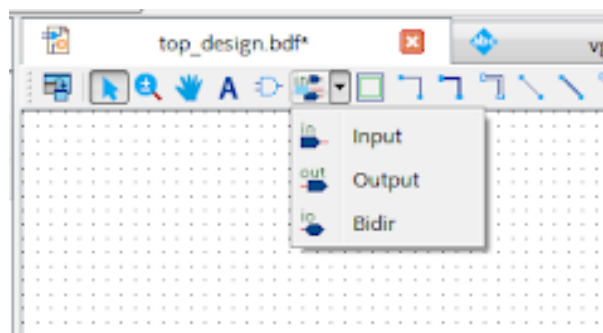
39. Insert the file into the project by locating it under the V folder.



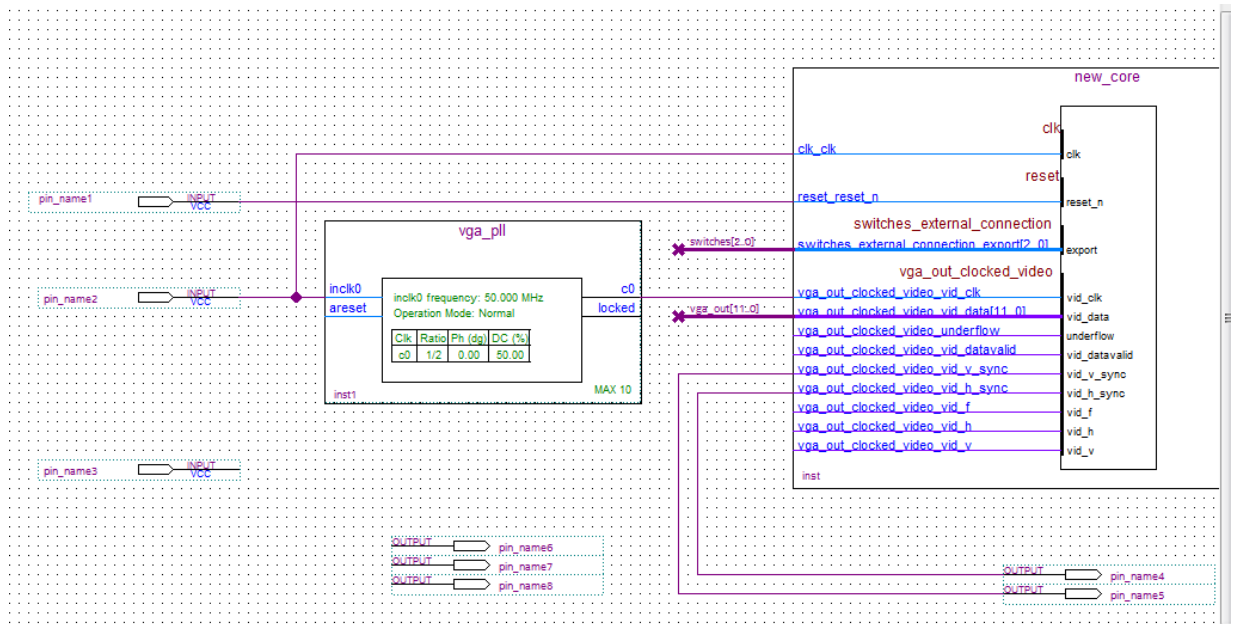
40. Once the symbol is added to the top_design.bdf then the connections need to be made to the I/O and from the PLL to the core. Lines can be drawn from one module to the next by clicking near the position of the symbol and dragging the mouse to the next connection point. Bus lines also will appear when drawing from the component.



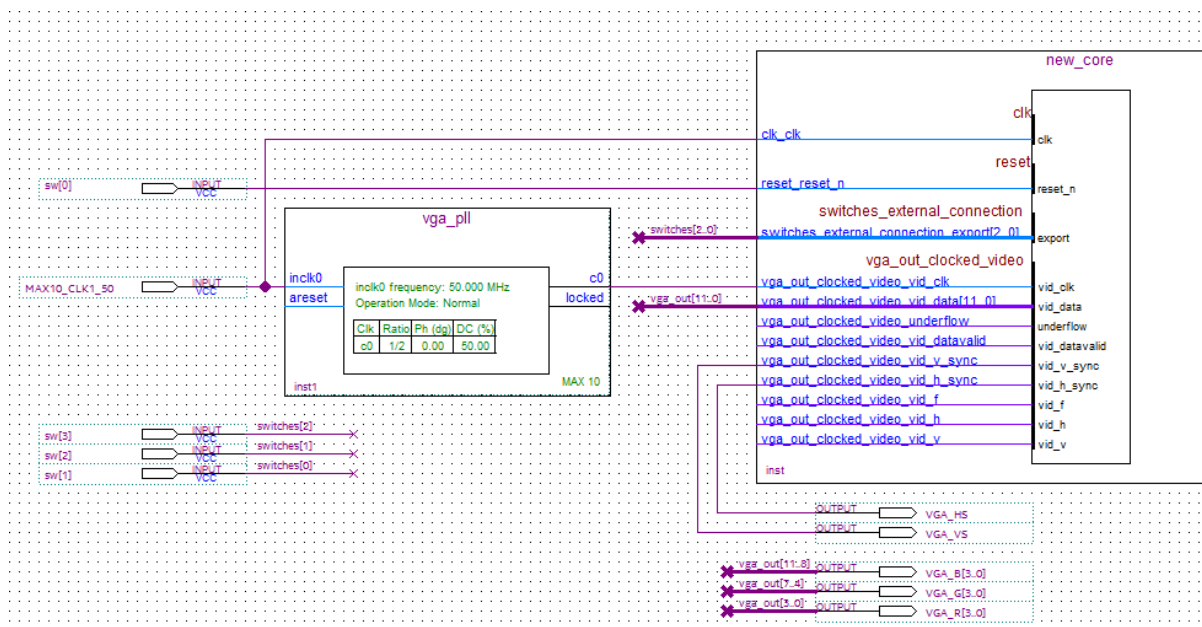
41. Name the bus lines for the switches and the data as switches[2..0] and vga_out[11..0] respectively. Right click on the bus line and select the properties. Next select the I/O connection by selecting the connectors.



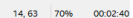
Add the I/O connectors as shown below.



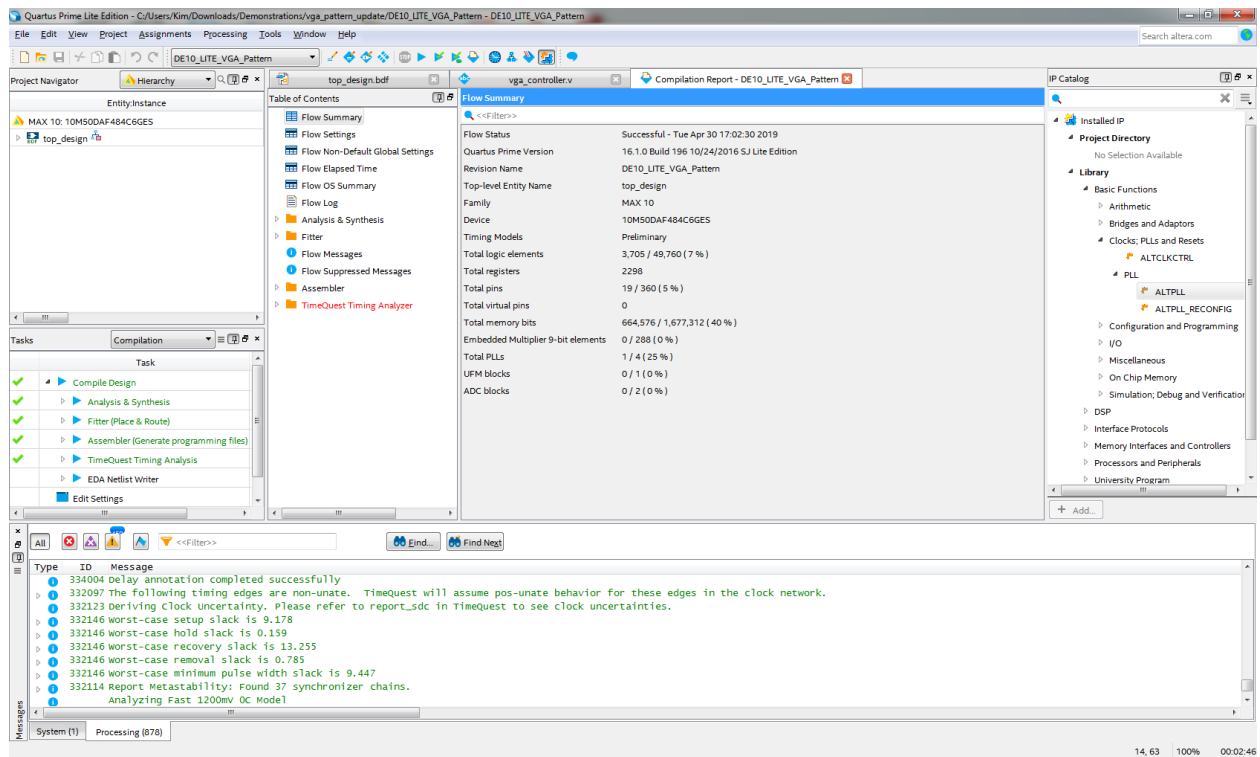
Label the reset signal as sw[0], input to the vga_pll as MAX10_CLK1_50, SW[3], SW[2], SW[1] and label the wires with the switches[2], switches[1], and switches[0]. For the output create labels for vid_v_sync to VGA_VS and vid_h_sync to VGA_HS. Connect the three output signals to VGA_B[3..0] with the bus input property labeled as vga_out[11..8], VGA_G[3..0] with the bus input property labeled as vga_out[7..4], VGA_R[3..0] with the bus input property labels as vga_out[3..0].



Save the top level file then compile.
An error occurs during assembly which needs to be fixed.



Compile the project.

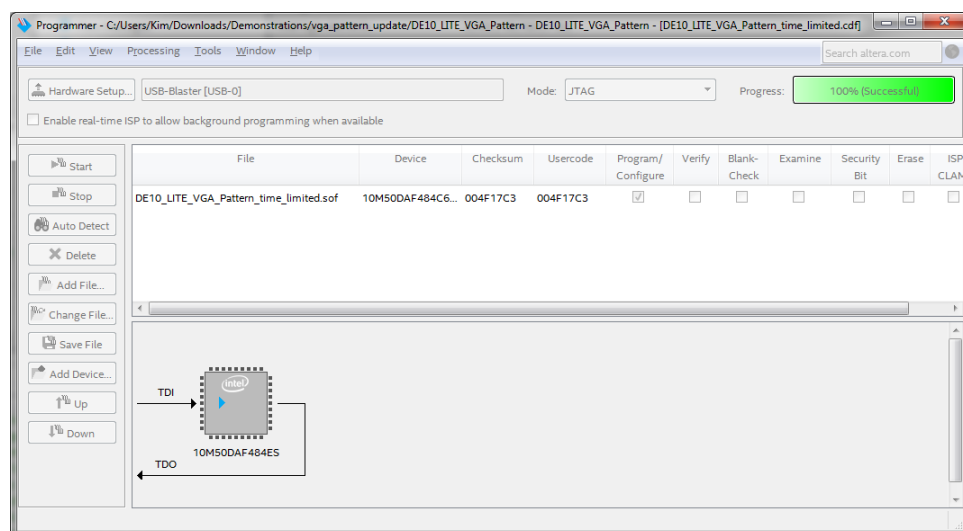


Next step now is to program the FPGA then access the Software Build Tool to control the VGA output.

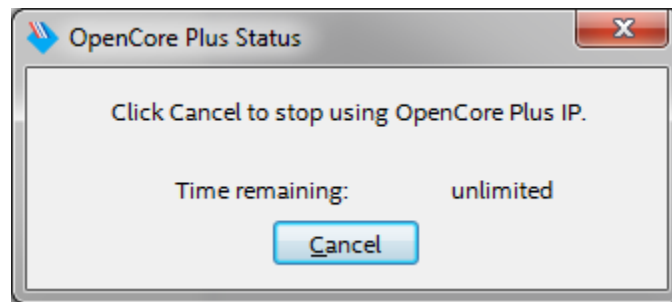
(ONCE THIS PART IS COMPLETED – submit the system.h file for grading)

You can go further here to get the VGA output to display:

A time limited SOF is generated because the Video IP suite is only in evaluation mode.

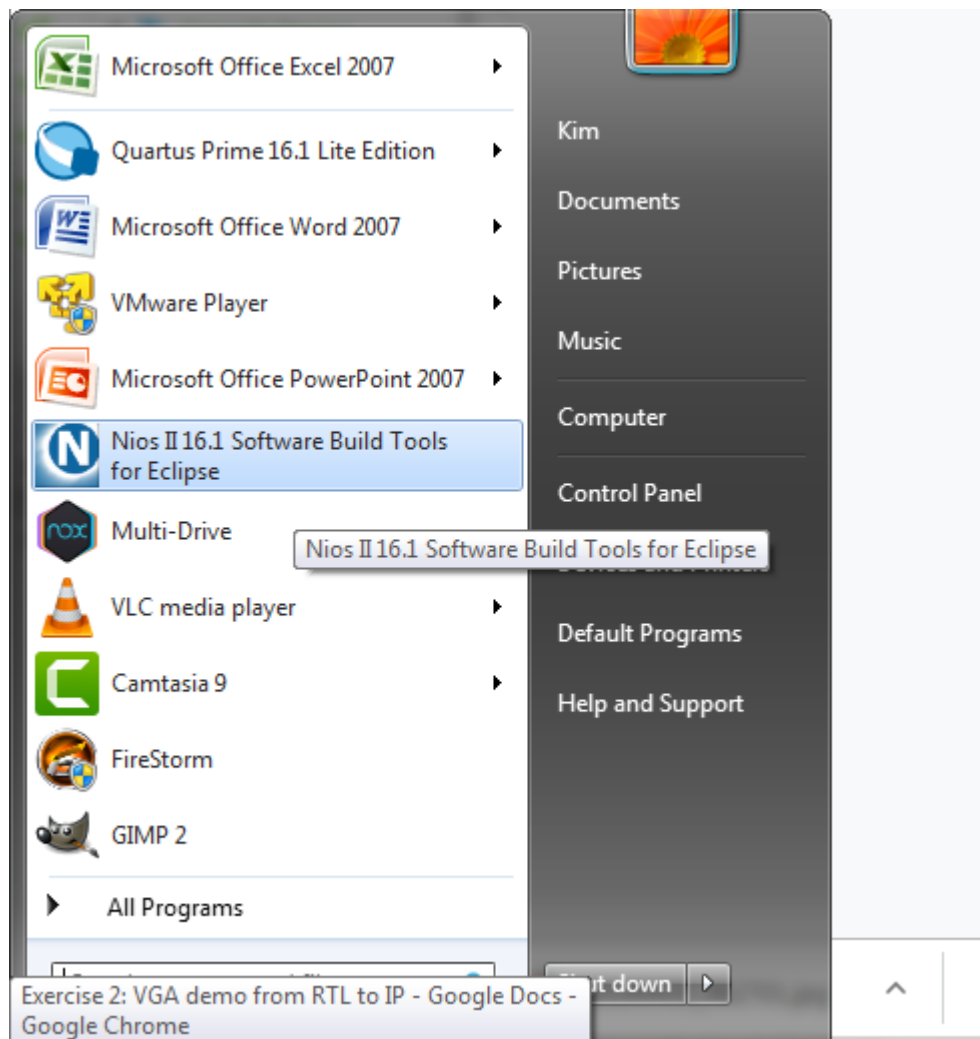


Once it is configured the status of the Open Core must be kept open.

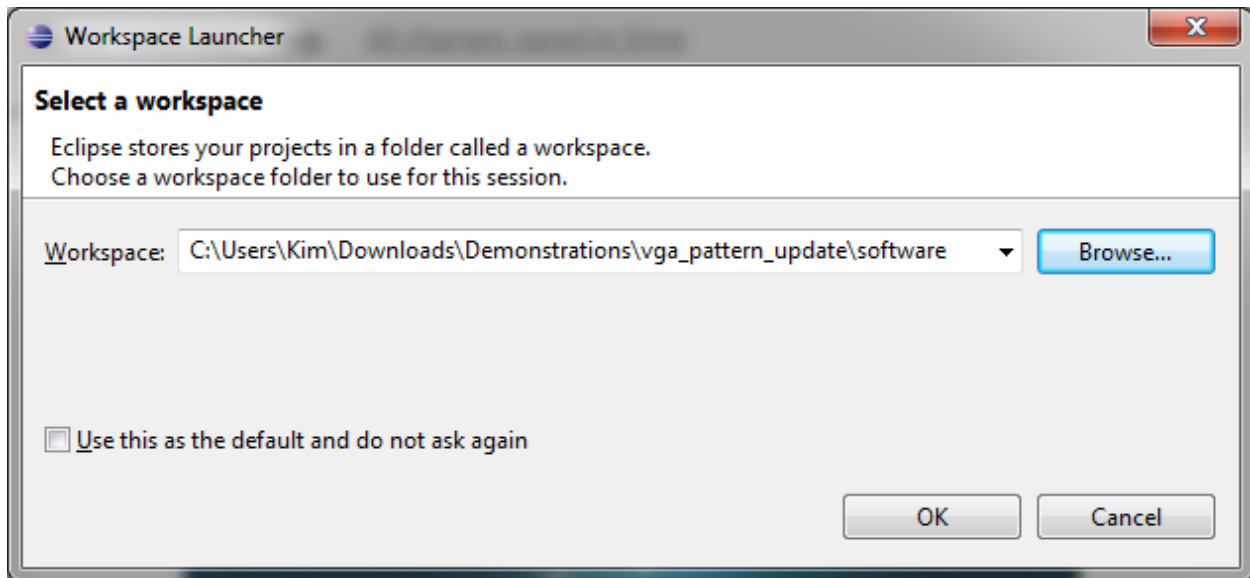


Navigate to the working directory for the FPGA project and make a new folder called software.

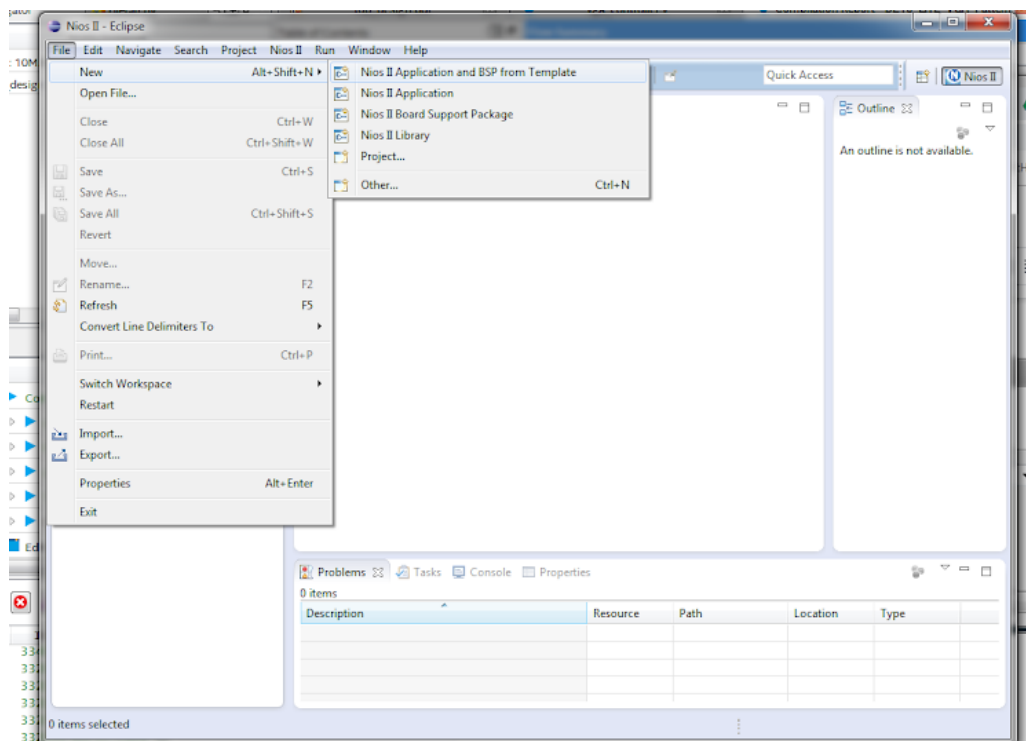
Launch the Eclipse platform.



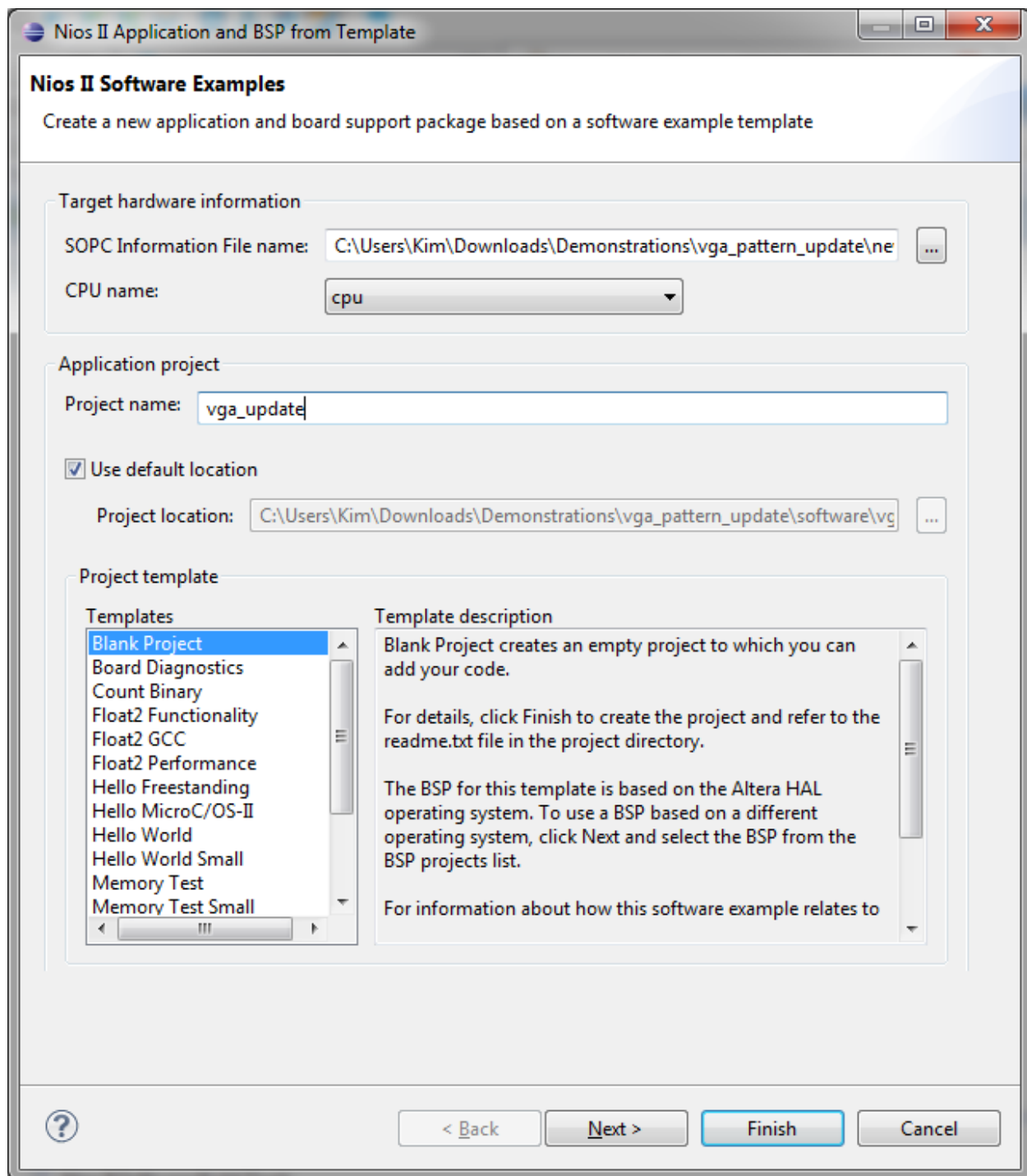
Select the software folder as the workspace.



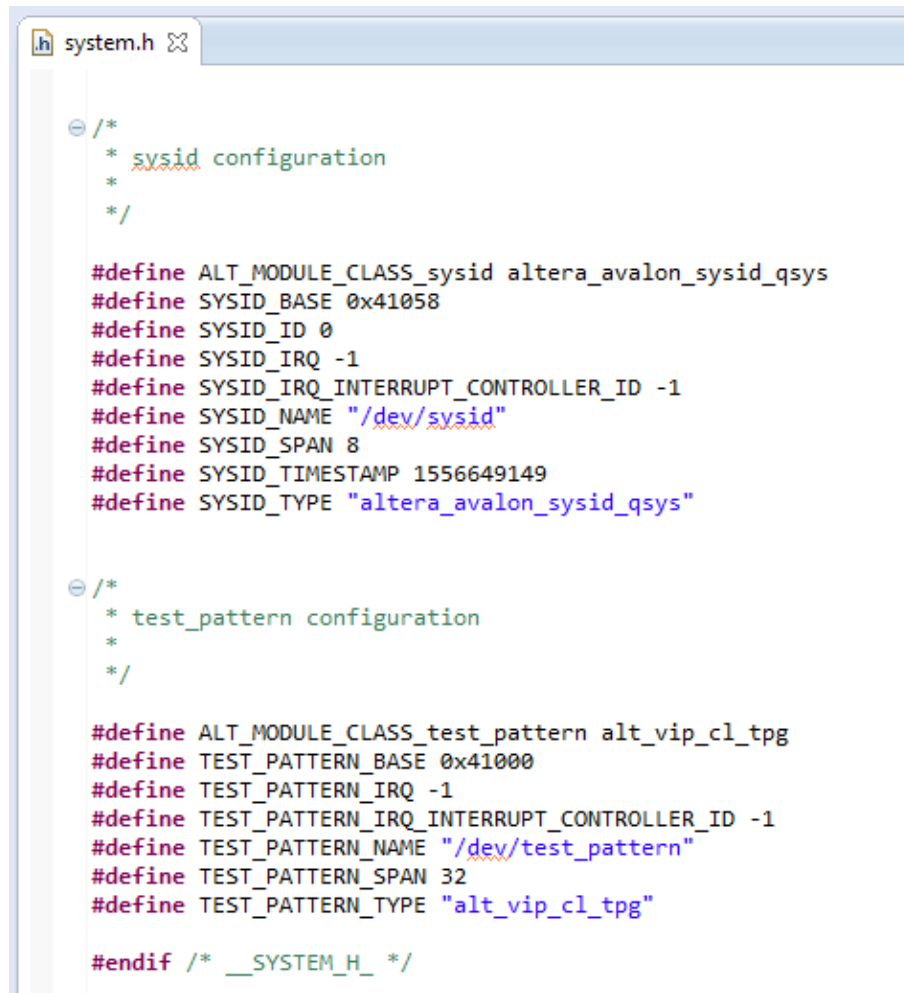
Create a new Nios II Application and BSP from Template.



Use the test_core.sopcinfo file generated by Qsys, name the project vga_update and create a blank project as shown in the screen capture below



The IDE will launch for the software build tool with two folders showing in the project explorer pane. Open the vga_update_bsp folder and navigate to the system.h file. This is generated by the sopcinfo file from Qsys and shows the location and definitions of the key components of the system. You can review this file but don't make any changes.



```
system.h

/*
 * sysid configuration
 */

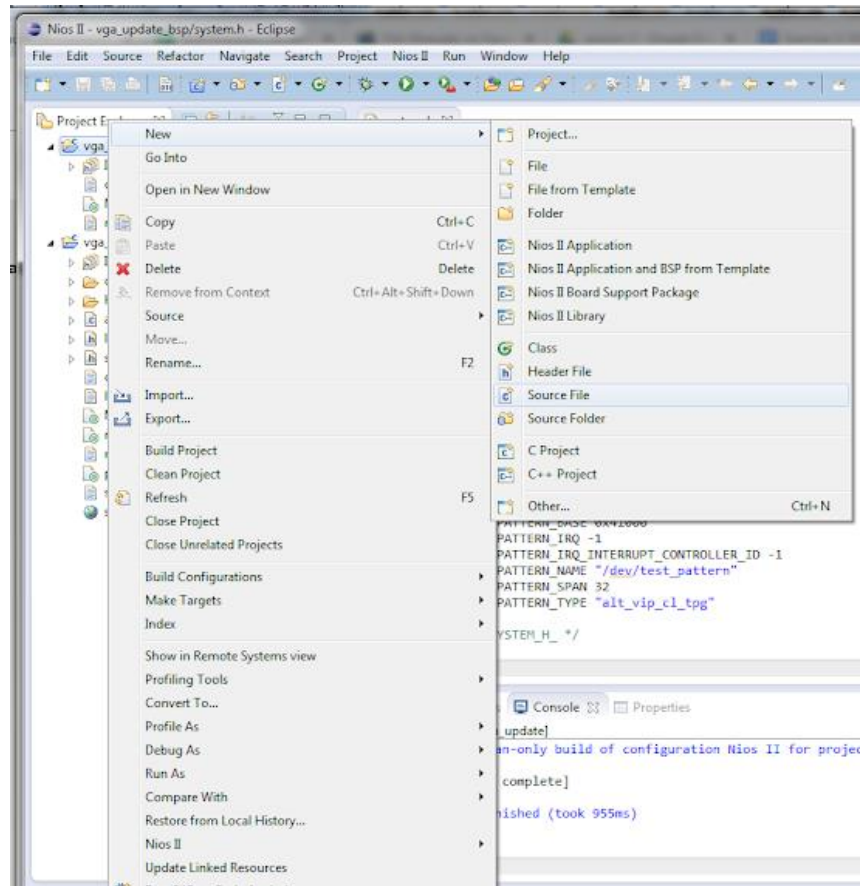
#define ALT_MODULE_CLASS_sysid altera_avalon_sysid_qsys
#define SYSID_BASE 0x41058
#define SYSID_ID 0
#define SYSID_IRQ -1
#define SYSID_IRQ_INTERRUPT_CONTROLLER_ID -1
#define SYSID_NAME "/dev/sysid"
#define SYSID_SPAN 8
#define SYSID_TIMESTAMP 1556649149
#define SYSID_TYPE "altera_avalon_sysid_qsys"

/*
 * test_pattern configuration
 */

#define ALT_MODULE_CLASS_test_pattern alt_vip_cl_tpg
#define TEST_PATTERN_BASE 0x41000
#define TEST_PATTERN_IRQ -1
#define TEST_PATTERN_IRQ_INTERRUPT_CONTROLLER_ID -1
#define TEST_PATTERN_NAME "/dev/test_pattern"
#define TEST_PATTERN_SPAN 32
#define TEST_PATTERN_TYPE "alt_vip_cl_tpg"

#endif /* __SYSTEM_H_ */
```

The test_pattern is to be controlled by the CPU so the base address will be used in this project. Create a new file in the top folder and name it main.c



Once the file is created add the following code to the new main.c file.

```

#include <stdio.h>
#include <unistd.h>
#include "system.h"
#include <io.h>

int main()
{
    // output width
    IOWR(TEST_PATTERN_BASE, 3, 640);
    // output height
    IOWR(TEST_PATTERN_BASE, 4, 480);

    // turn on TPG
    IOWR(TEST_PATTERN_BASE, 0, 1);

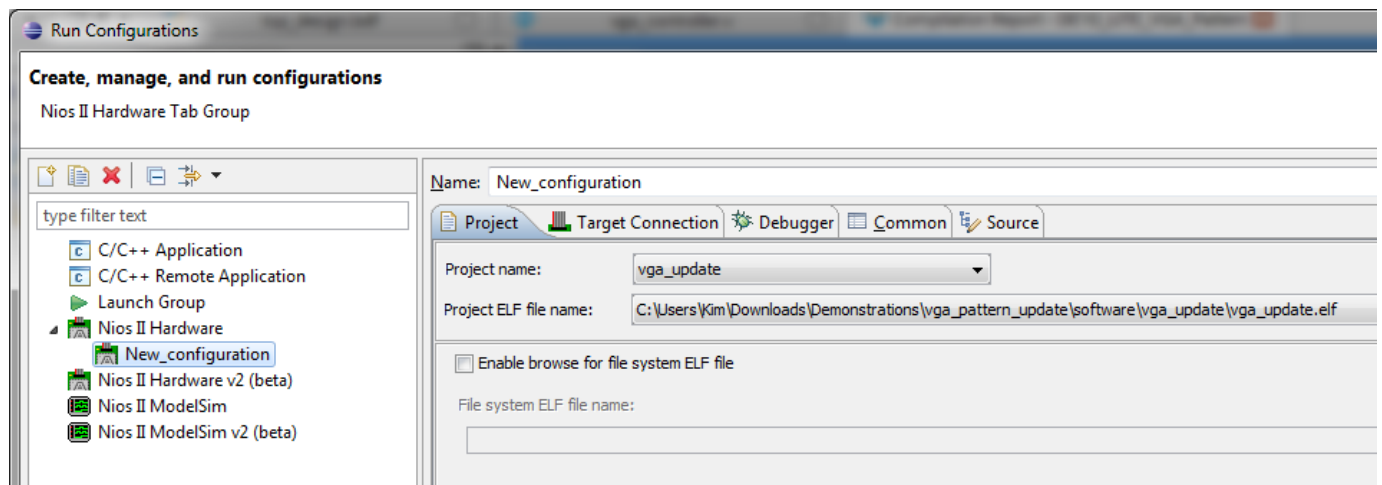
    // Color values
    IOWR(TEST_PATTERN_BASE, 5, 16);
    IOWR(TEST_PATTERN_BASE, 6, 16);
    IOWR(TEST_PATTERN_BASE, 7, 180);

    printf("New color is ???\n");

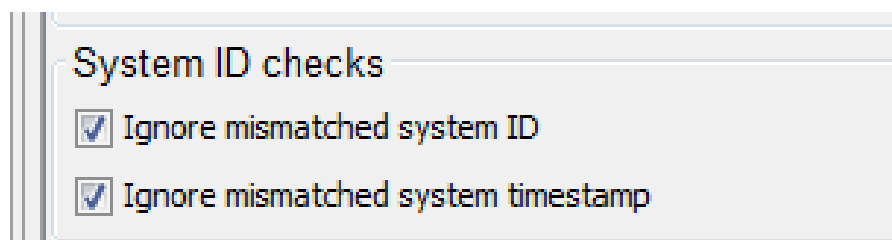
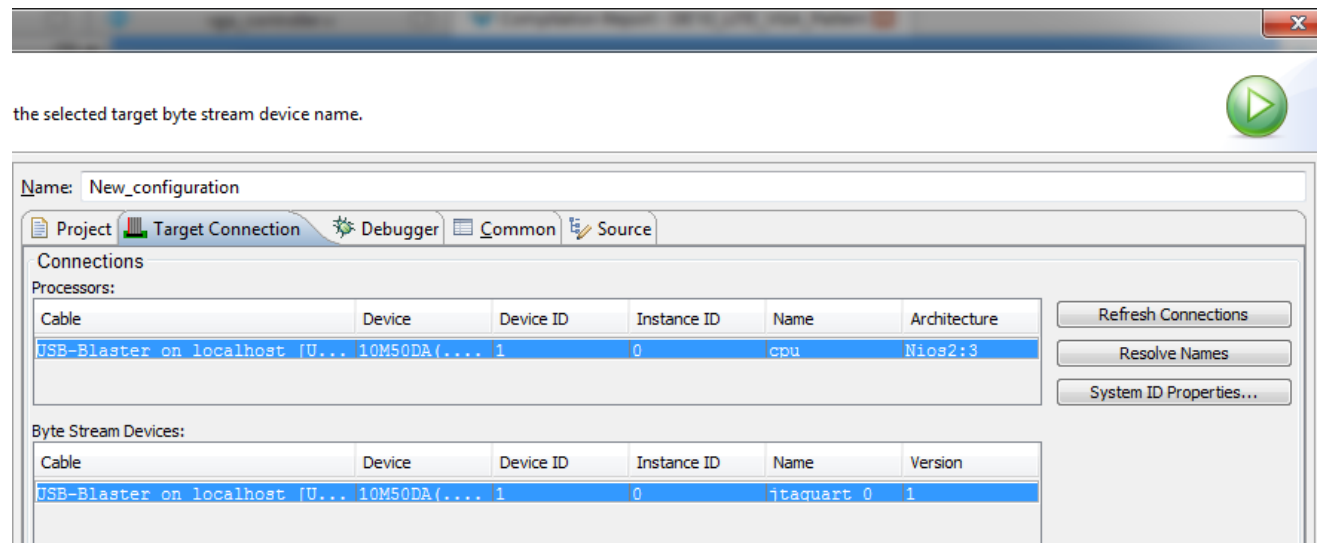
    // infinite loop for testing
    while (1);
}

```

Build the project then setup the Run configuration. From the top menu select Run -> Run Configuration. Select NIOS II hardware then configure as shown below.



On the Target Configuration tab check the boxes for the System ID and timestamp and refresh connections if they are not shown in blue like the one in the screen shots below.



Next run the project and see what appears on the monitor. What color is displayed? _____