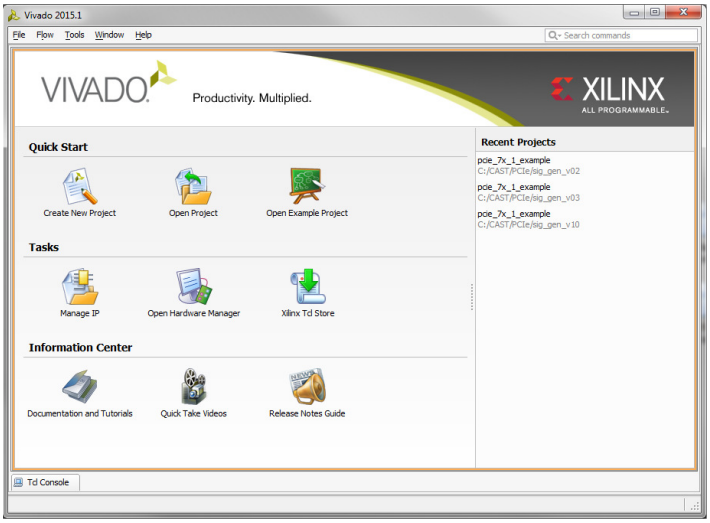
BASYS3 board tutorial

(Decoder design using Vivado 2015.1)

Note: you will need the Xilinx Vivado Webpack version installed on your computer (or you can use the department systems).

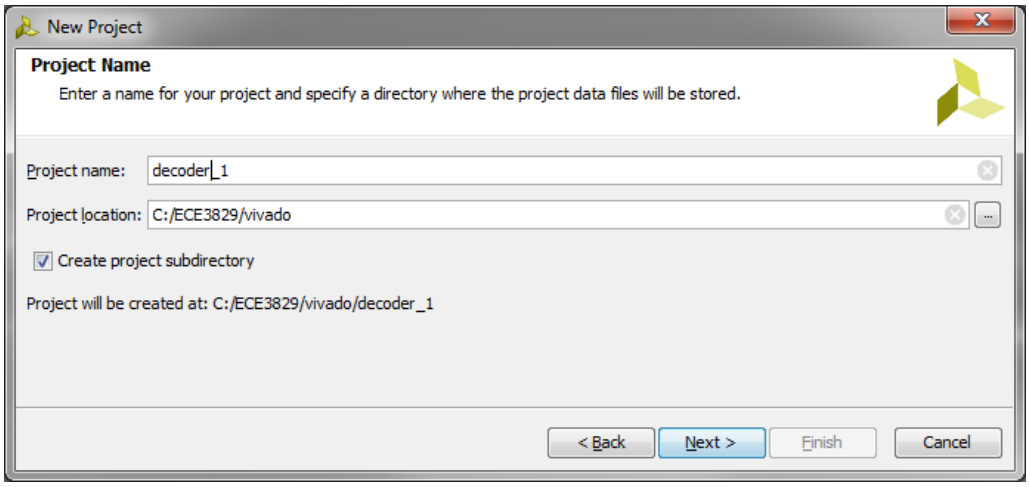
This tutorial shows how to create a simple combinational design (a 3 to 8 decoder using the slider switches and leds) that can be implemented on the Basys3 board.

Start Vivado Design Suite:

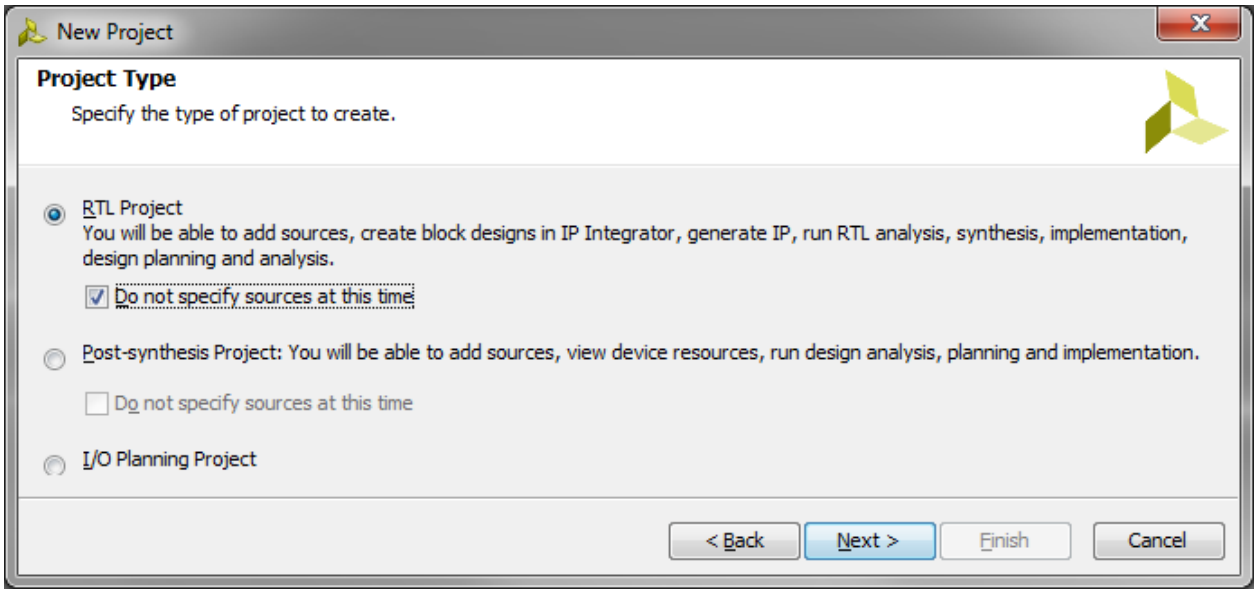


Select Create New Project.

Click Next and then enter a Project name and location for your project:

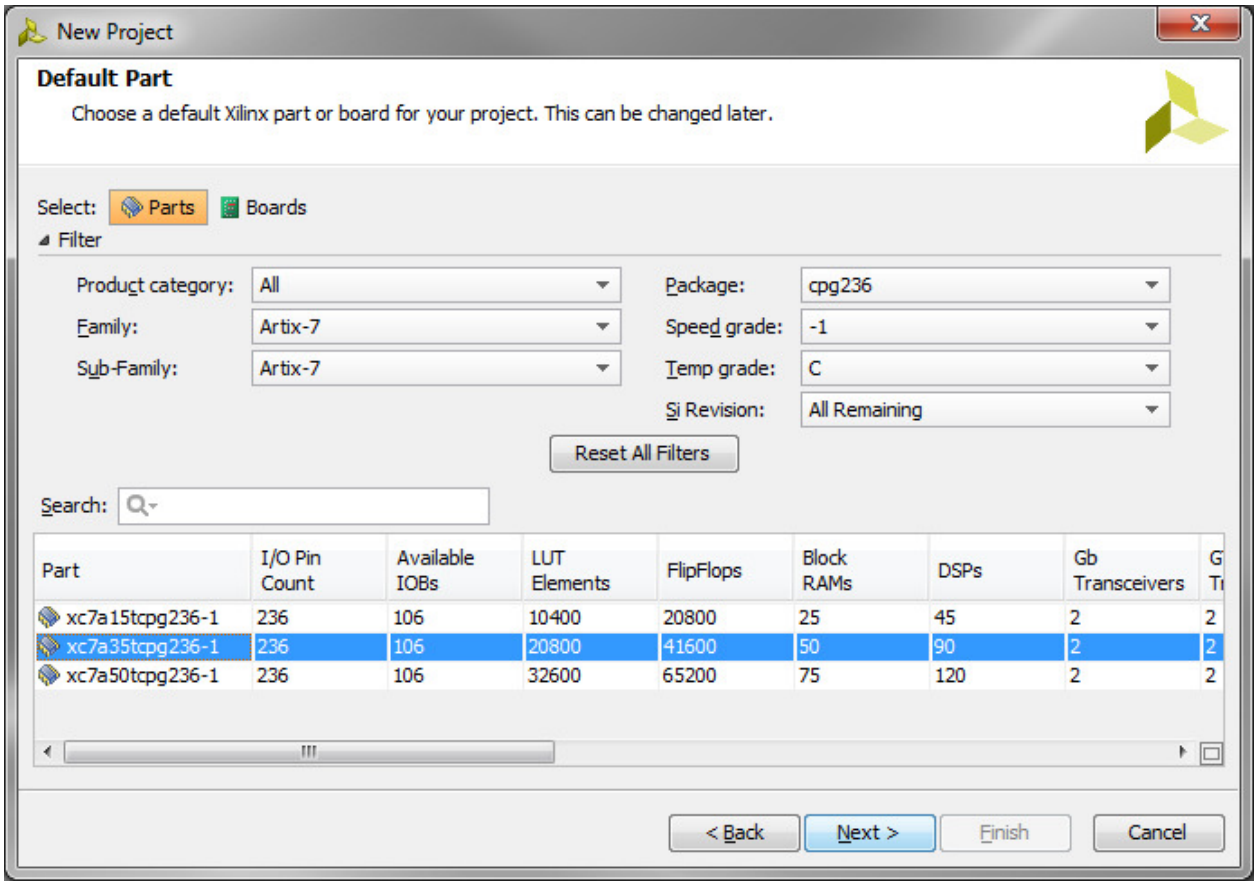


Click Next and select the RTL project type:

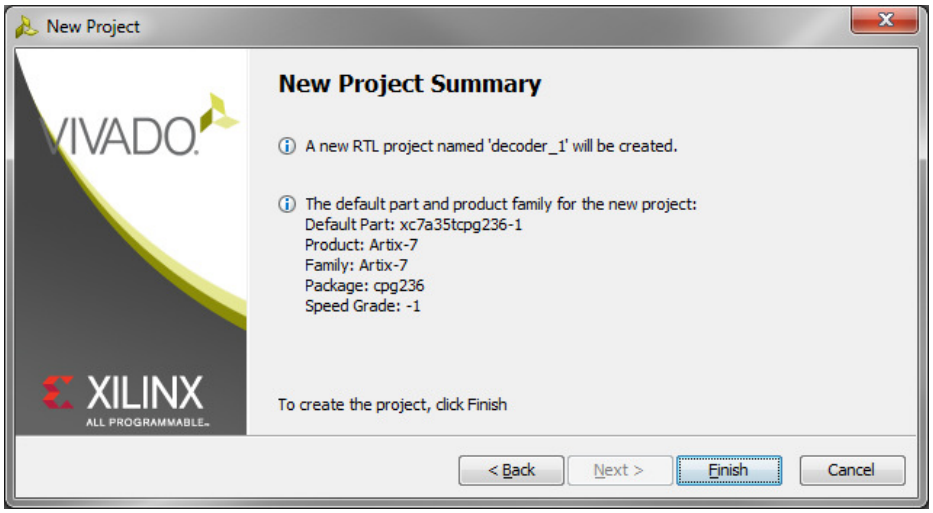


Check the "Do not specify courses at this time" box and click Next:

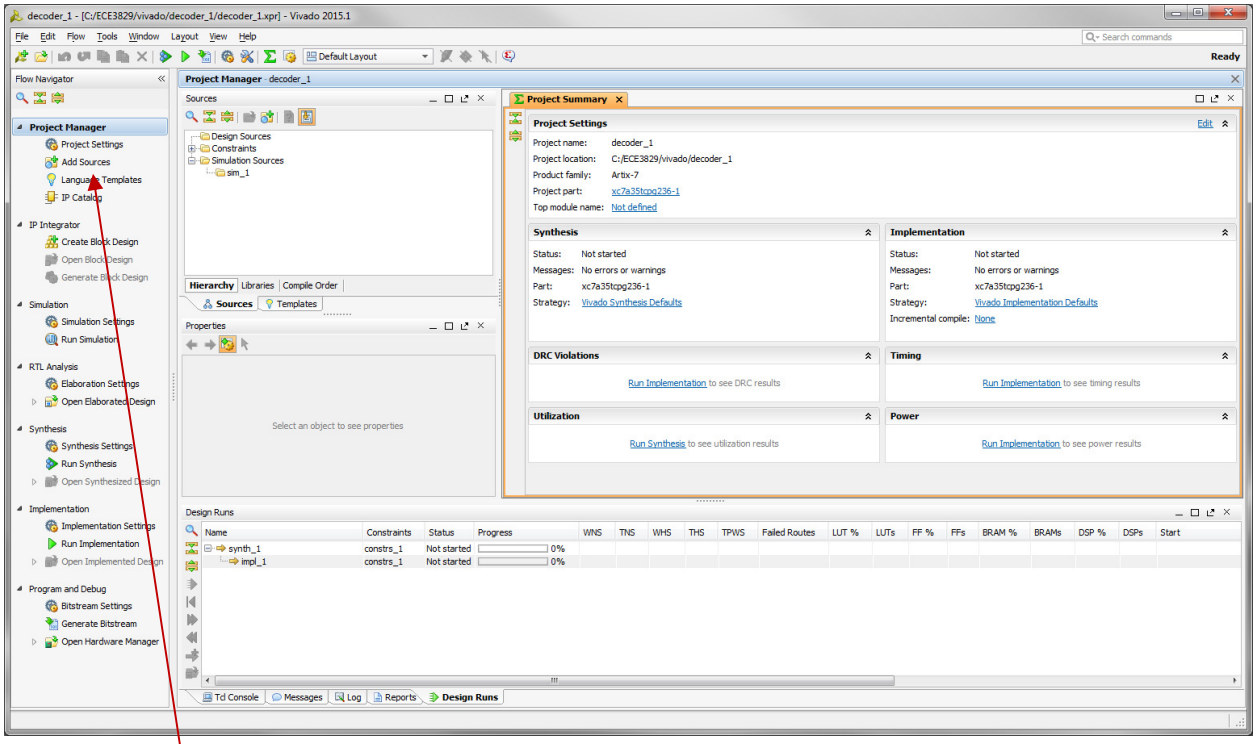
Select the corect Xilinx FPGA that is on the Basys3 board (XC7A35T-1CPG236C)



Click Next, and then Finish:



The Project window opens:

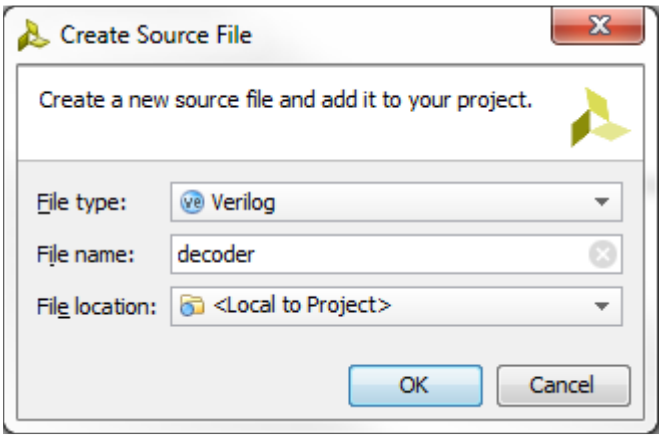


We now need to add a Verilog design source to describe our decoder operation.

Click on Add Sources in the left Project Manager window (or select the menu item File => Add Sources:

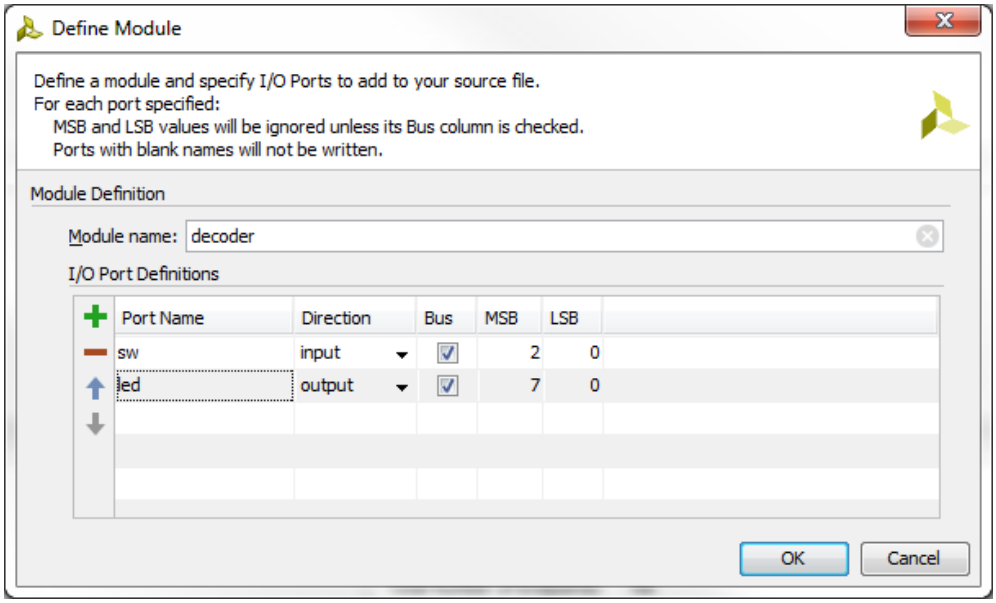
Select Next,

And then select Create File (click on the + symbol) and enter decoder for the file name:



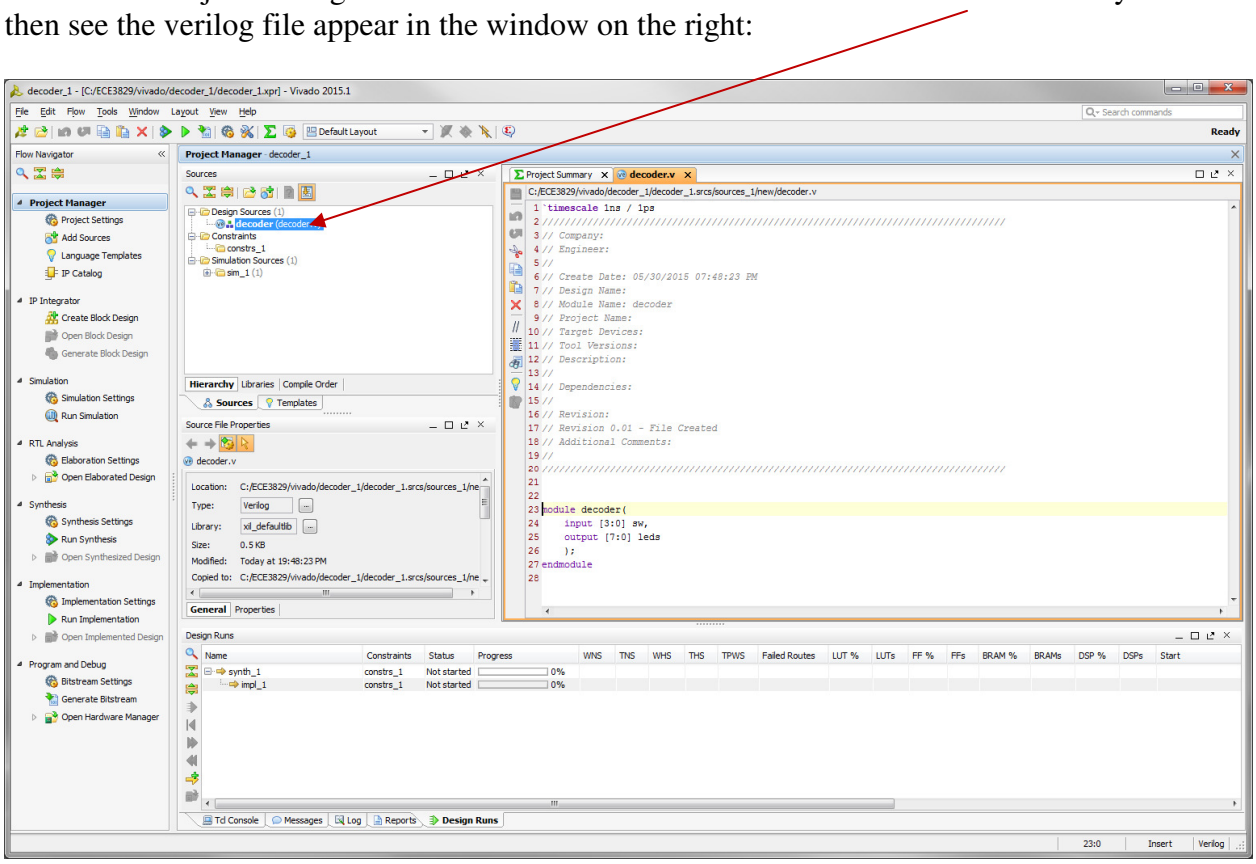
Then click OK and Finish.

We can now specify the inputs and outputs to create our 3 to 8 decoder (we will use three switches and eight LEDs):



Click OK.

Back in the Project Manager Sources window double-click the new decoder.v file and you will then see the verilog file appear in the window on the right:



You should add your name and a description of this file to the header description.

We can now add the verilog statements to design our 3 to 8 decoder.

There are anumber of ways to design the decoder, below is an example using a case statament:

decoder.v

C:/ECE3829/vivado/decoder_1/decoder_1.srcs/sources_1/new/decoder.v

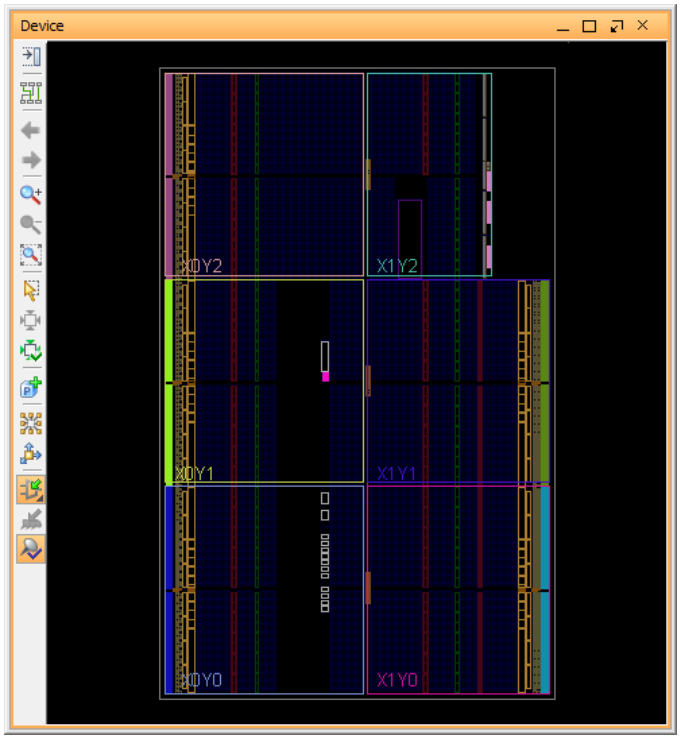
```
23 module decoder(
24     input       [2:0] sw,
25     output  reg [7:0] led  // reg type required
26     );
27
28     always @ (sw)   // we will use a case statement for this decoder
29         case (sw)
30             3'b000: led = 8'b00000001;
31             3'b001: led = 8'b00000010;
32             3'b010: led = 8'b00000100;
33             3'b011: led = 8'b00001000;
34             3'b100: led = 8'b00010000;
35             3'b101: led = 8'b00100000;
36             3'b110: led = 8'b01000000;
37             3'b111: led = 8'b10000000;
38         endcase
39
40 endmodule
```

Now we can synthesize the design.

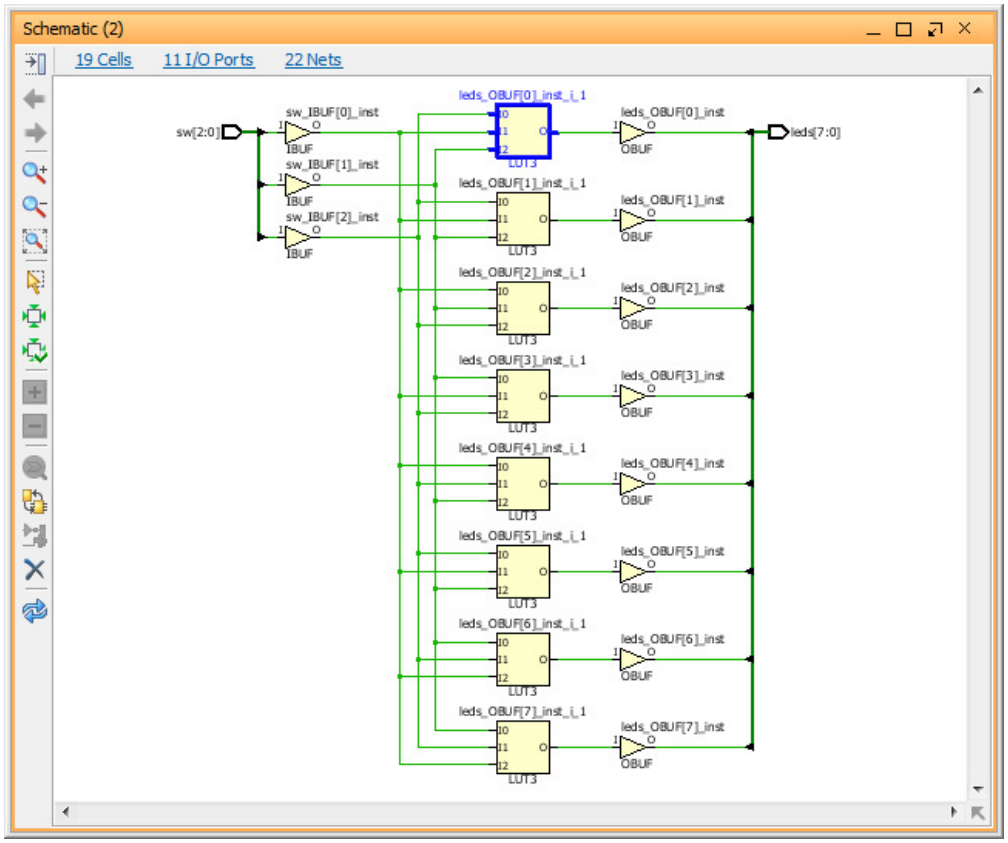
Click Run Synthesis in the Project Manager window.

After synthesis is complete there should be no errors or warnings reported.

If you open the synthesized design you can see a device level representation (this is mostly empty since we just have a very simple design that only uses a small fraction of the available FPGA resources):



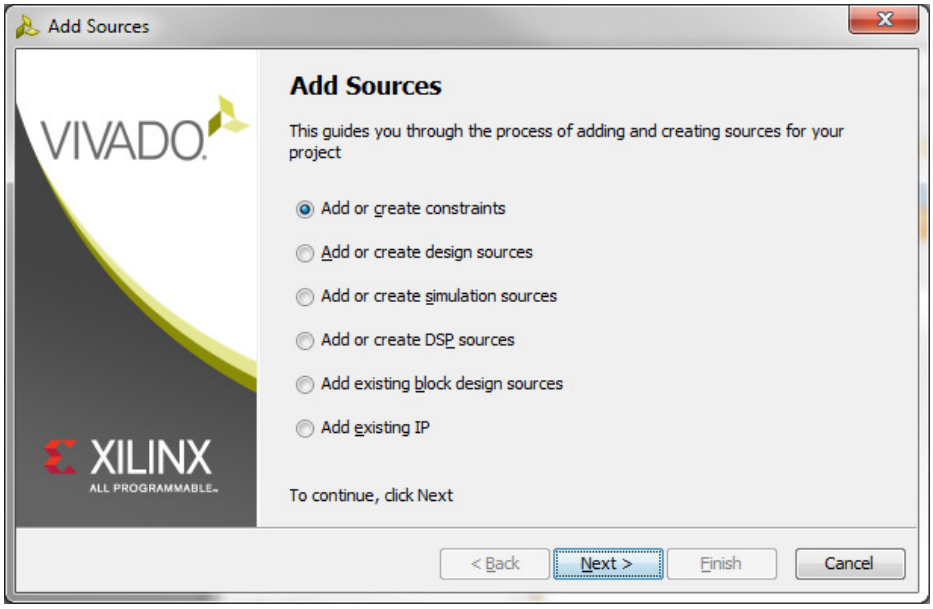
But you can also look at a schematic representation to see the input and output buffers and the LUTs used:



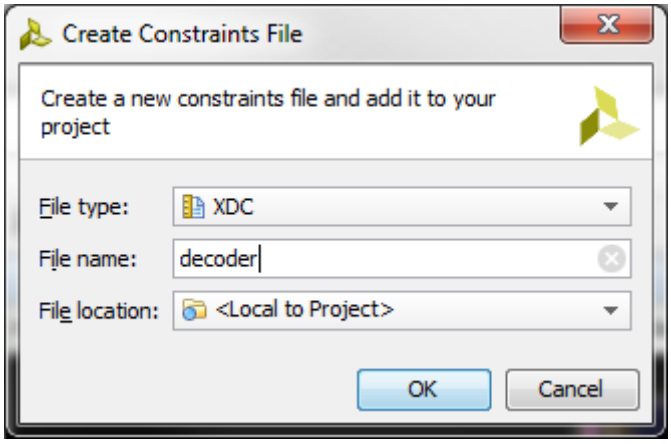
Before we can implement the design we need to specify the FPGA pins that will be used for the SW inputs and LED outputs.

Look at the Basys3 manual to determine the FPGA pins.

Next click Add Sources and select 'Add or create constraints':



We can name his decoder



In the souces window window select the constraints file by the hierarchy Constraints => constrs_1 => decoder.xdc. We can then add location constraints for all the inputs and outputs (you can download a copy of the Basys3 XDC constraints from the Digilent website – just copy the pins you are using for the design):

These constraints specify the pins to use for each signal and what type of interface.

decoder.xdc

C:/ECE3829/vivado/decoder_1/decoder_1.srcs/constrs_1/new/decoder.xdc

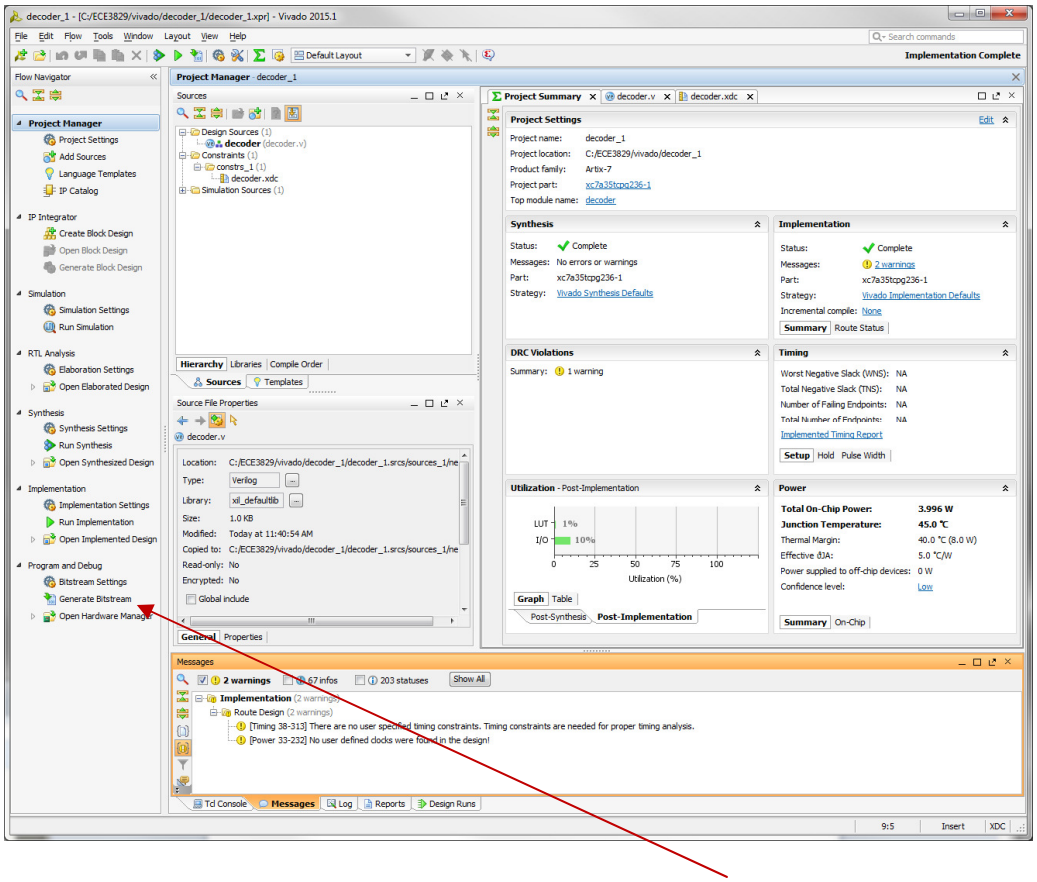
```
# Switches
set_property PACKAGE_PIN V17 [get_ports {sw[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sw[0]}]
set_property PACKAGE_PIN V16 [get_ports {sw[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sw[1]}]
set_property PACKAGE_PIN W16 [get_ports {sw[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sw[2]}]

# LEDs
set_property PACKAGE_PIN U16 [get_ports {led[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {led[0]}]
set_property PACKAGE_PIN E19 [get_ports {led[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {led[1]}]
set_property PACKAGE_PIN U19 [get_ports {led[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {led[2]}]
set_property PACKAGE_PIN V19 [get_ports {led[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {led[3]}]
set_property PACKAGE_PIN W18 [get_ports {led[4]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {led[4]}]
set_property PACKAGE_PIN U15 [get_ports {led[5]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {led[5]}]
set_property PACKAGE_PIN U14 [get_ports {led[6]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {led[6]}]
set_property PACKAGE_PIN V14 [get_ports {led[7]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {led[7]}]
```

Now we have specified the correct pins to use for the design (so it matches the Basys3 board layout) we can implement the design.

Click Run Implementation in the Project Manager window.

You will find there are two warnings after implementation. This is because we have not specified any timing constraints. For this simple combinational circuit we can ignore this.



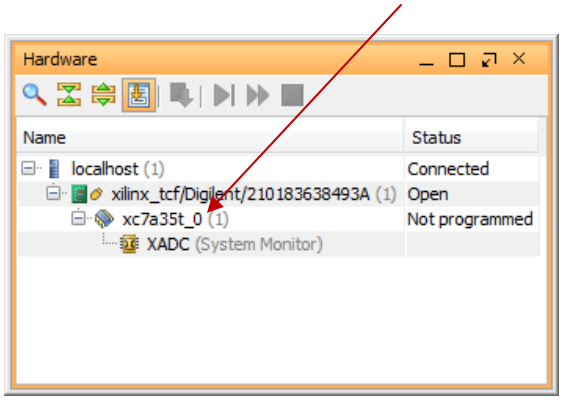
We can now generate the bitstream. Click on Generate Bitstream in the Project Manager window.

Next step is to program the FPGA with the bitsream.

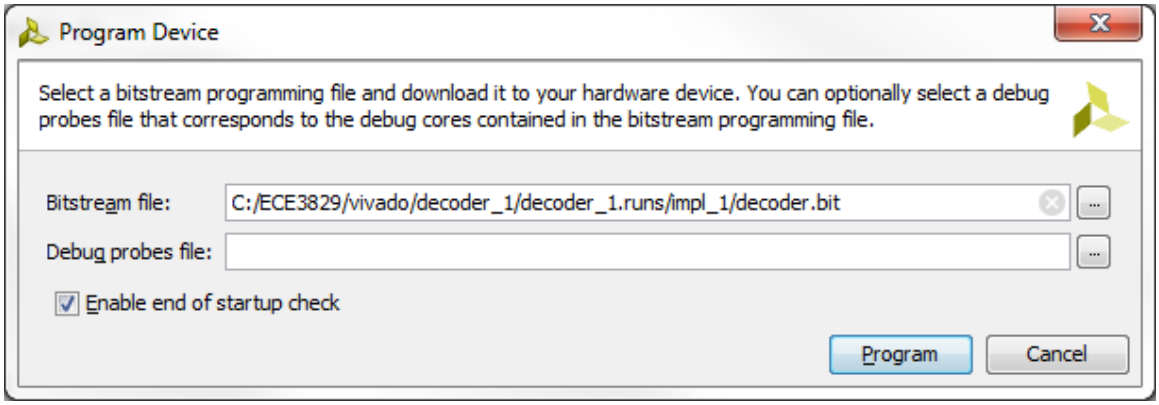
On the Basys3 board make sure that JP1 is set to the JTAG mode and connect the USB cable to the board and turn on the power.

Select the Hardware Manager in the Project Manager and select Open Target and then Auto Connect.

You should see the Xilinx xc7a35t_0 in the Hardware Window:



Click on Program Device and select the decoder.bit bitstream (automatically filled in):



Then select Program (ignore the warning about the missing debug core and the rule violation).

Programming should just take a few seconds and then the Done led on the Basys3 board will turn on.

You should now find that your decoder is implemented on the FPGA.
Change the three slider switches (SW2, SW1, SW0) through all eight combinations and verify the correct corresponding LED turns on.

Congratulations!

You have entered a design and then Synthesized, Implemented, and programmed the FPGA with the generated bit file. This was a simple design example but the same steps are just repeated for any design.

Close the Hardware manager to go back to the Project Manager window.

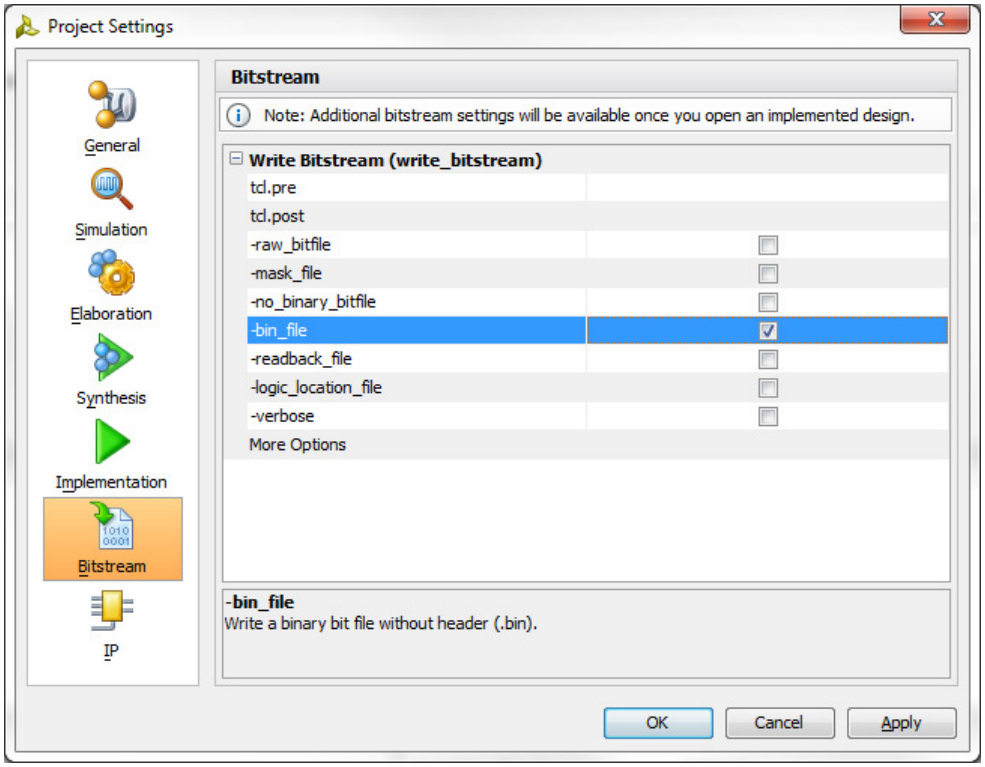
Programming the Serial Flash

The FPGA is a volatile device and so the bit file will not be present after power cycling the board. We can load the QSPI serial flash on the Basys3 board so it loads the bit file from the flash on power up.

First we need to create a bin file to be able to program the serial flash.

Click on the Bitstream Settings in the Program and Debug section of the Project manager.

Select the bin_file option:

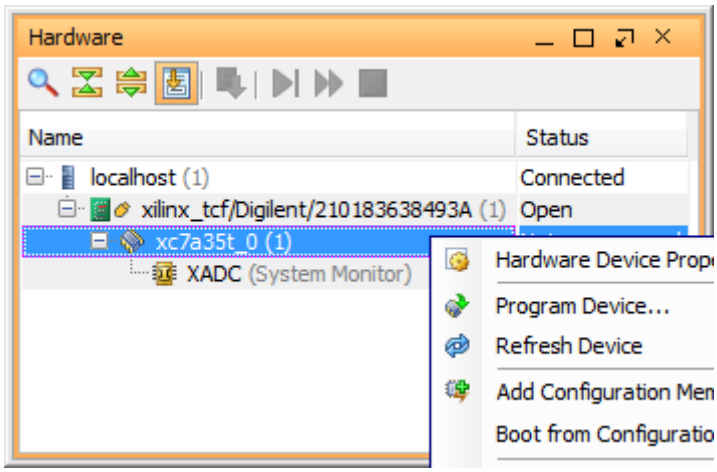


Click OK.

Click on Generate Bitstream.

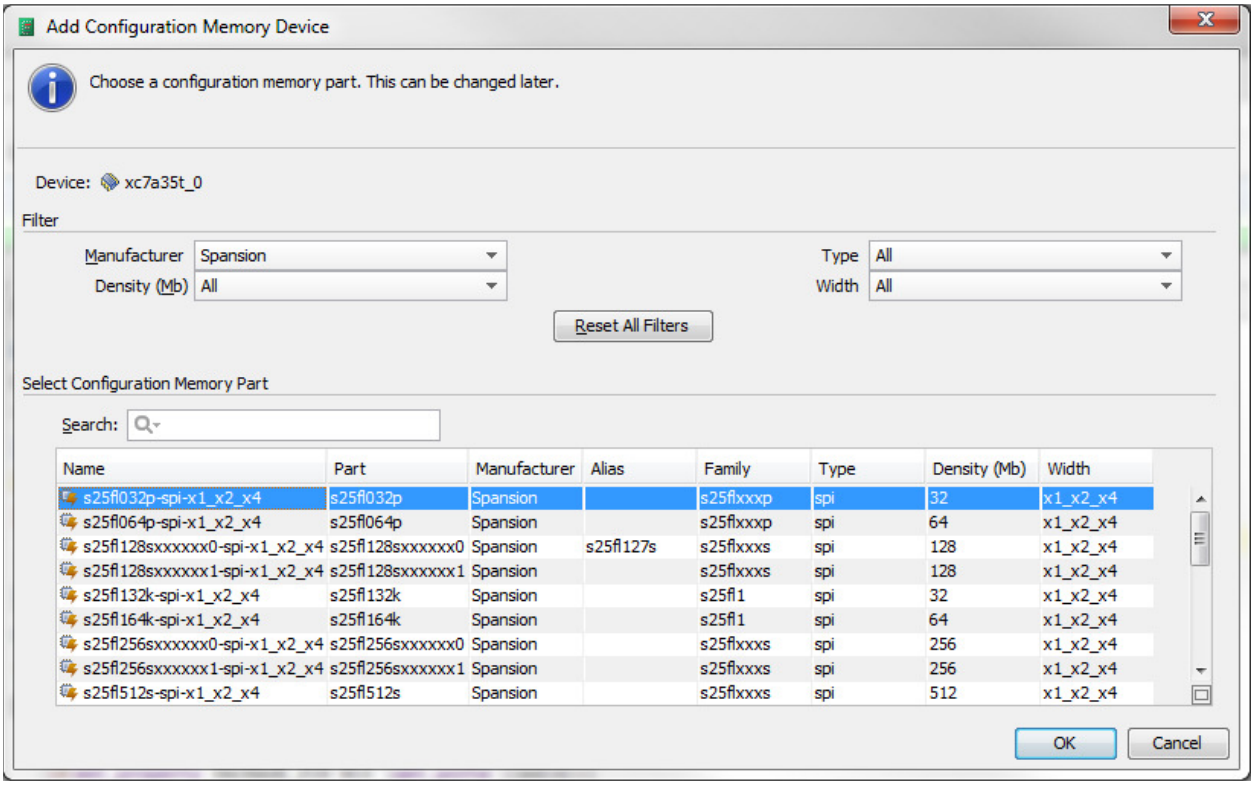
After generation, if you look in the decoder_1 => decoder_1.runs => impl_1 directory you will see a decoder.bit and a decoder.bin file.

Use the Hardware Manager to connect to the Basys3 board then right-mouse click on the FPGA and select Add Configuration Memory Device:

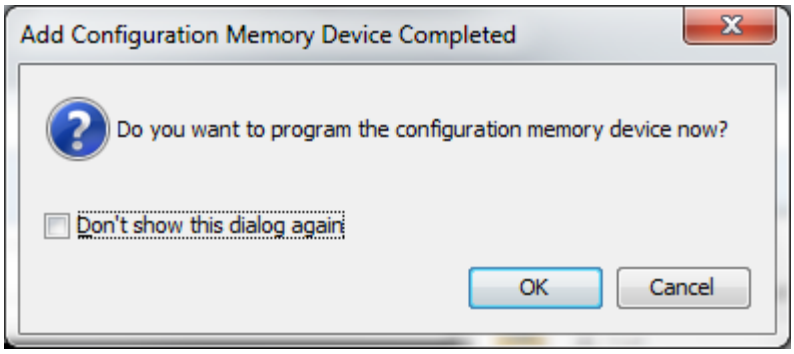


We now need to select the serial flash device that is on the Basys3 board.

Select Spansion as the Manufacturer, then select the 32Mb device:

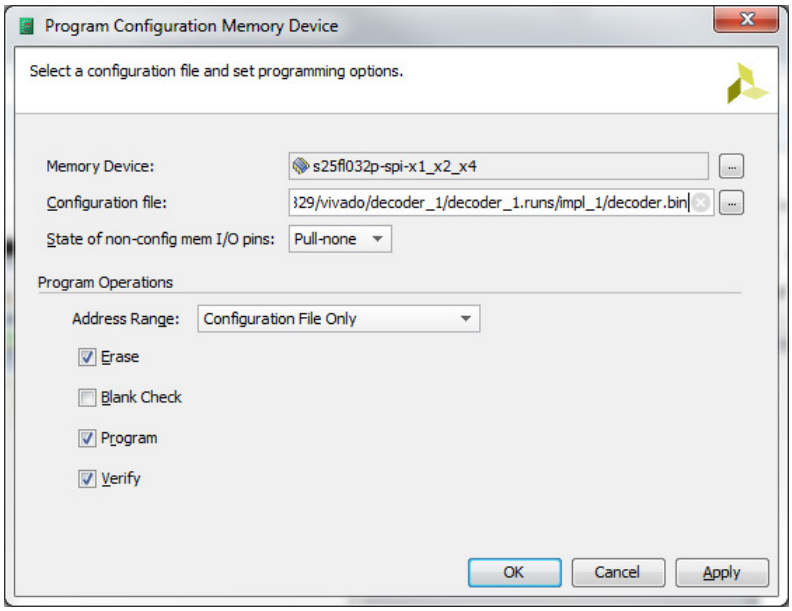


Click OK



Click OK

Select the Configuration File (decoder.bin) in the impl_1 directory.



Click OK.

Note: this will erase any existing design in the QSPI flash (including the configuration file shipped with the Basys3 board).

The QSPI Flash will now be erased and then programmed with the decoder.bin file.

Once programmed, you can power off the Basys3 board and change the JP1 jumper mode from JTAG to QSPI.

Power back on the board and after a few seconds the Done LED will turn on indicating that your decoder design has been automatically loaded into the FPGA from the serial flash. You can verify the decoder design by moving the slider switches and observing the leds as before.