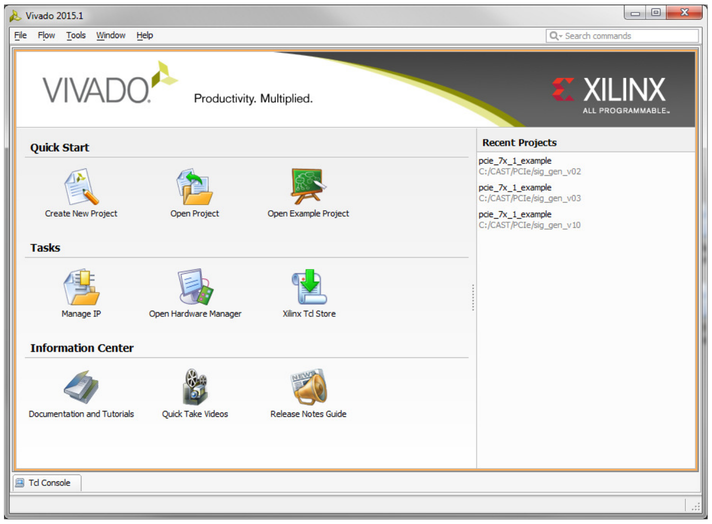
NEXYS4DRR board tutorial

(VHDL 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).

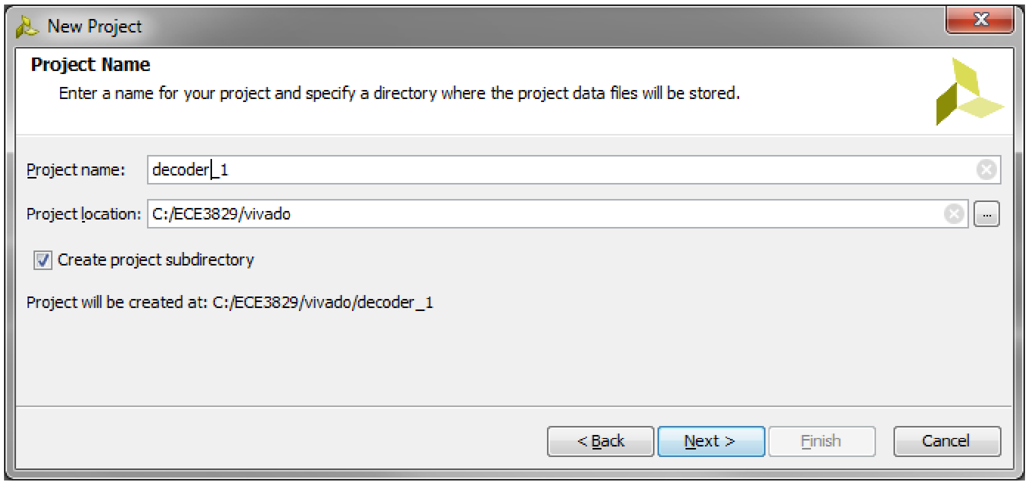
This tutorial shows how to create a simple combinational design (a 3 to 8 decoder using the slider switches and LEDs) that can be implemented on the Nexys4DDR board.

Start Vivado Design Suite:



Select Create New Project.

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



Project Name

Enter a name for your project and specify a directory where the project data files will be stored.

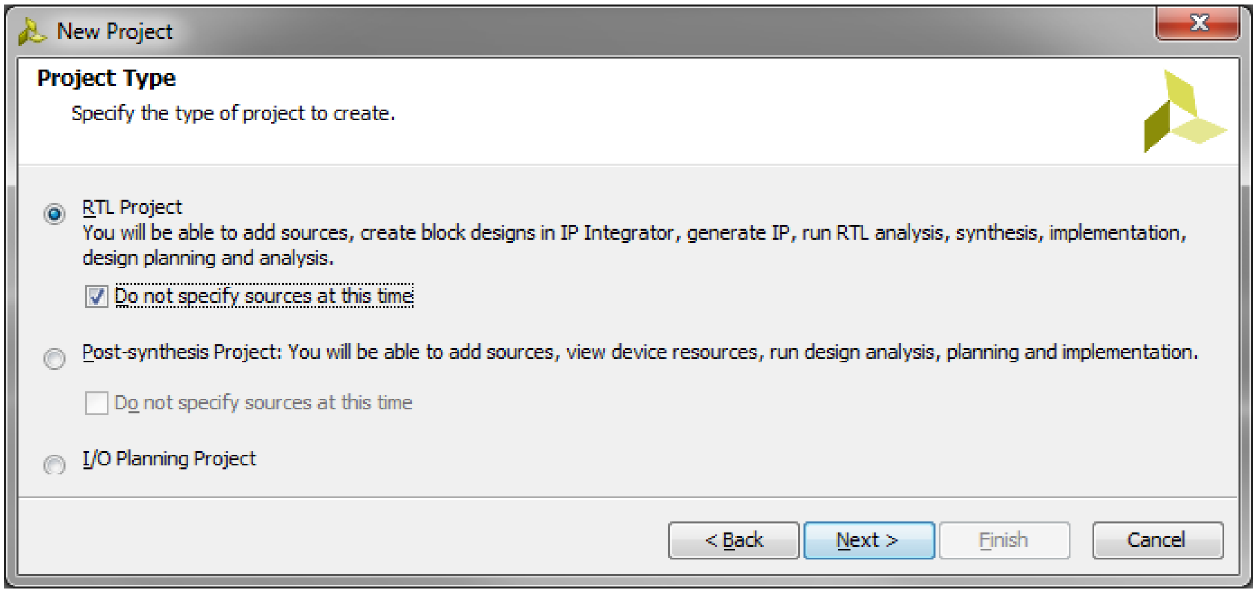
Project name: decoder_1

Project location: C:/ECE3829/vivado

☑ Create project subdirectory

Project will be created at: C:/ECE3829/vivado/decoder_1

Click Next and select the RTL project type:



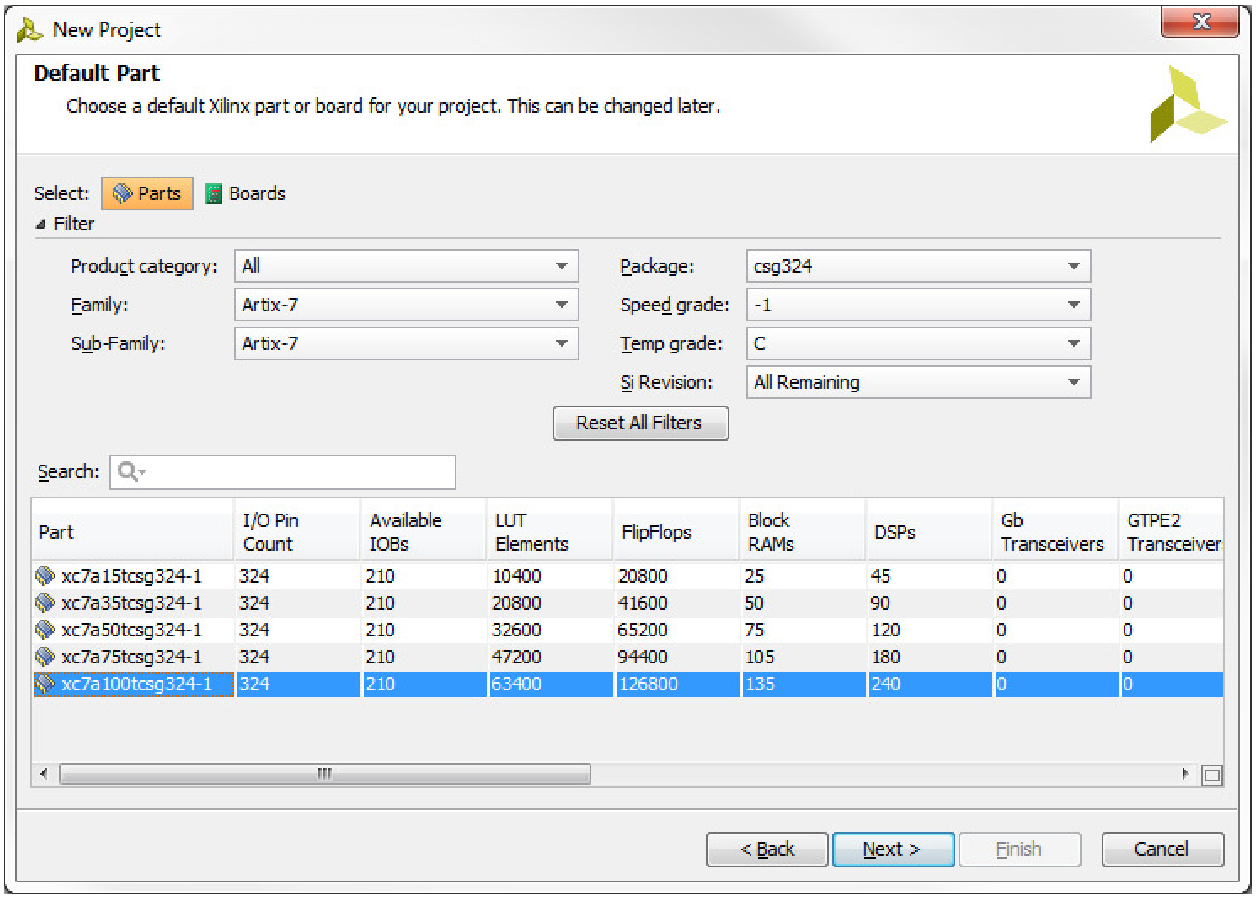
Project Type

Specify the type of project to create.

- ● RTL Project
  You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.
  - ☑ Do not specify sources at this time
- ○ Post-synthesis Project: You will be able to add sources, view device resources, run design analysis, planning and implementation.
  - ☐ Do not specify sources at this time
- ○ I/O Planning Project

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

Select the corect Xilinx FPGA that is on the Nexys4DDR board (XC7A100T-1CSG324C):



Default Part

Choose a default Xilinx part or board for your project. This can be changed later.

Select: [✓ Parts] Boards

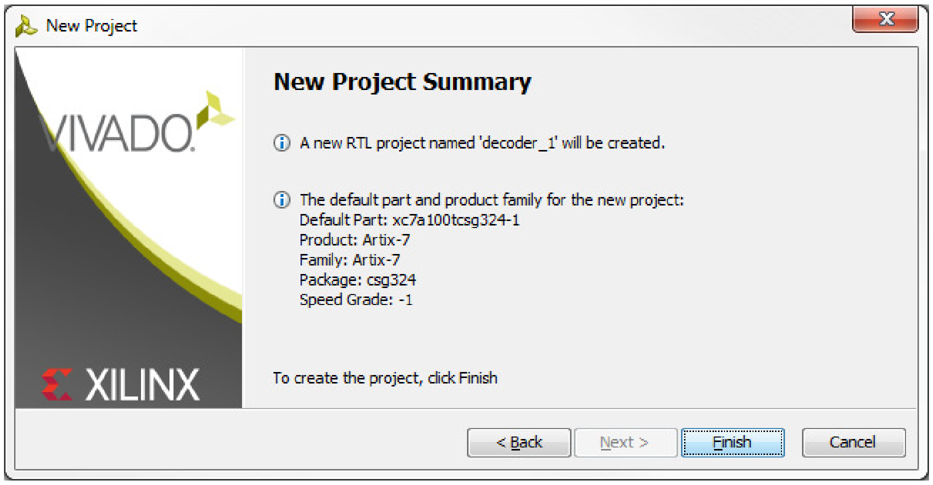
Filter

Product category:	All	Package:	csg324
Family:	Artix-7	Speed grade:	-1
Sub-Family:	Artix-7	Temp grade:	C
		Si Revision:	All Remaining

Search:

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	DSPs	Gb Transceivers	GTPE2 Transceiver
xc7a15tcsg324-1	324	210	10400	20800	25	45	0	0
xc7a35tcsg324-1	324	210	20800	41600	50	90	0	0
xc7a50tcsg324-1	324	210	32600	65200	75	120	0	0
xc7a75tcsg324-1	324	210	47200	94400	105	180	0	0
xc7a100tcsg324-1	324	210	63400	126800	135	240	0	0

Click Next, and then Finish:



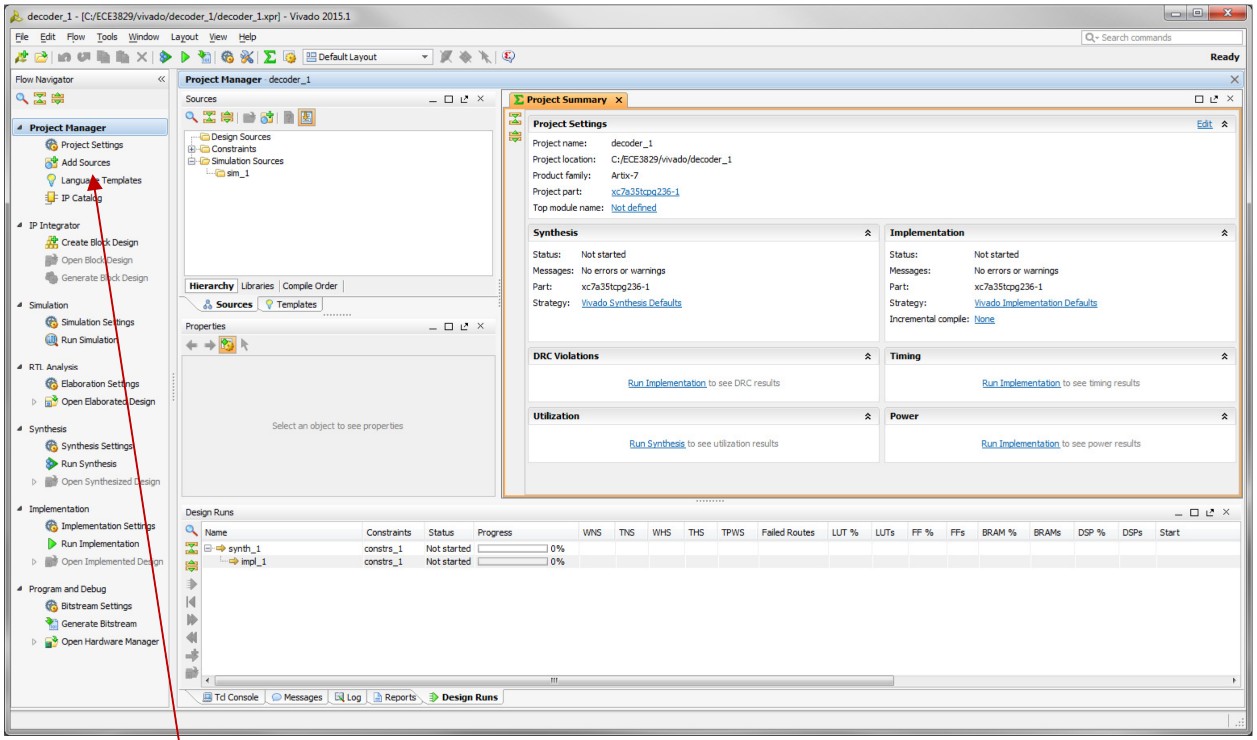
New Project Summary

A new RTL project named 'decoder_1' will be created.

The default part and product family for the new project:
Default Part: xc7a100tcsg324-1
Product: Artix-7
Family: Artix-7
Package: csg324
Speed Grade: -1

To create the project, click Finish

The Project window opens:



Project Settings

Project name: decoder_1
Project location: C:/ECE3829/vivado/decoder_1
Product family: Artix-7
Project part: xc7a35tcpg236-1
Top module name: Not defined

Synthesis

Status: Not started
Messages: No errors or warnings
Part: xc7a35tcpg236-1
Strategy: Vivado Synthesis Defaults

Implementation

Status: Not started
Messages: No errors or warnings
Part: xc7a35tcpg236-1
Strategy: Vivado Implementation Defaults
Incremental compile: None

DRC Violations

Run Implementation to see DRC results

Timing

Run Implementation to see timing results

Utilization

Run Synthesis to see utilization results

Power

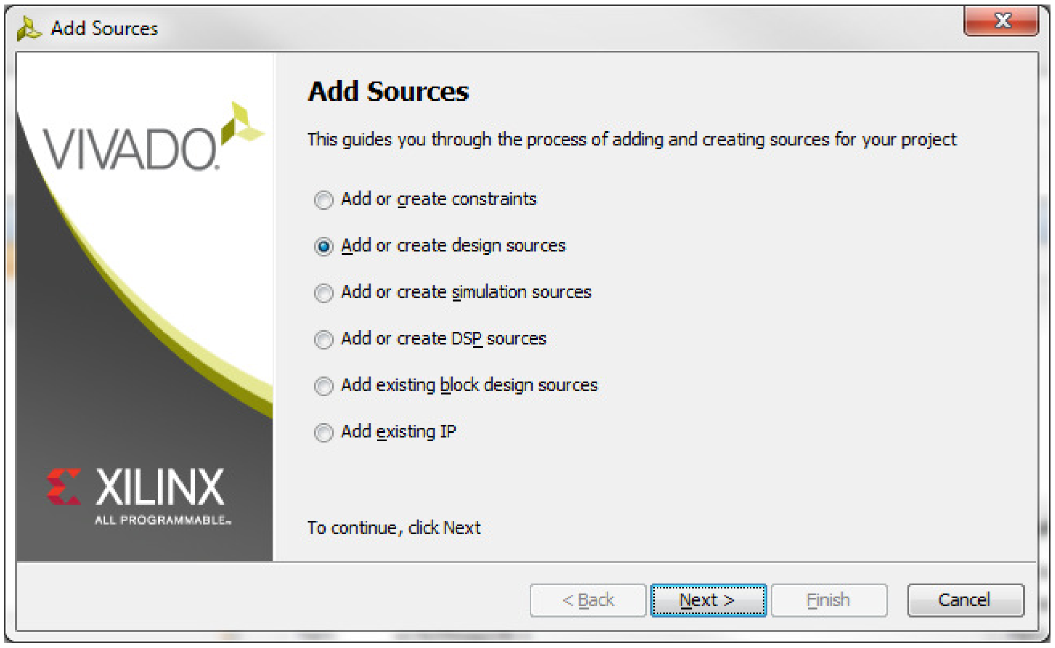
Run Implementation to see power results

Design Runs

Name	Constraints	Status	Progress	WNS	TNS	WHS	THS	TPWS	Failed Routes	LUT %	LUTs	FF %	FFs	BRAM %	BRAMs	DSP %	DSPs	Start
synth_1	constrs_1	Not started	0%															
impl_1	constrs_1	Not started	0%															

We now need to add a VHDLg 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:


Add Sources

This guides you through the process of adding and creating sources for your project

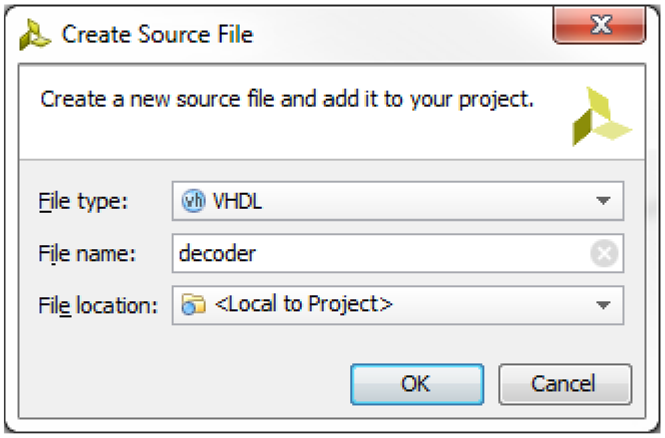
Add or create constraints [✓ Add or create design sources] Add or create simulation sources Add or create DSP sources Add existing block design sources Add existing IP

To continue, click Next

Select Next,

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

Make sure the file type is set to VHDL


Create Source File

Create a new source file and add it to your project.

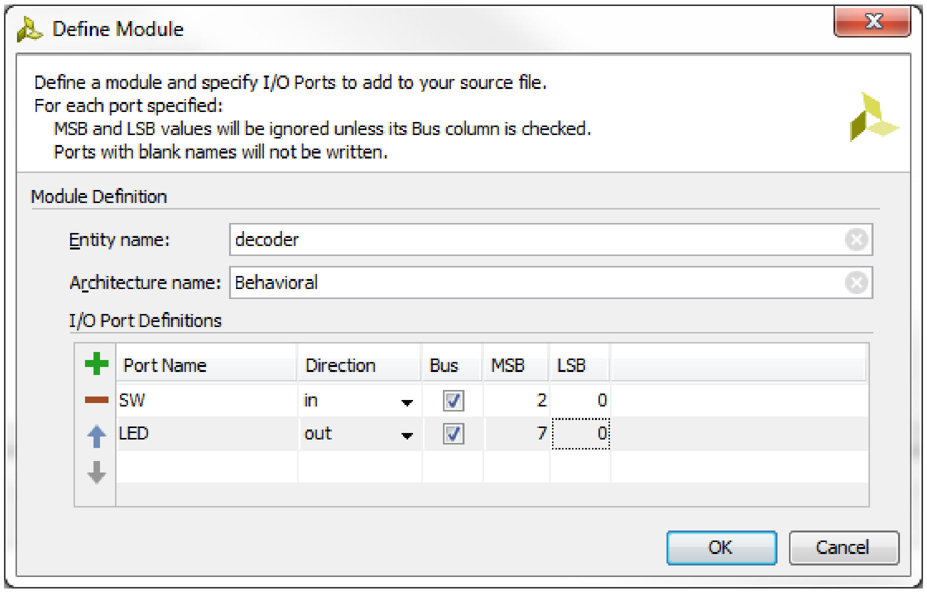
File type: VHDL

File name: decoder

File location: <Local to Project>

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):



Define Module

Define a module and specify I/O Ports to add to your source file.
For each port specified:
  MSB and LSB values will be ignored unless its Bus column is checked.
  Ports with blank names will not be written.

Module Definition

Entity name: decoder

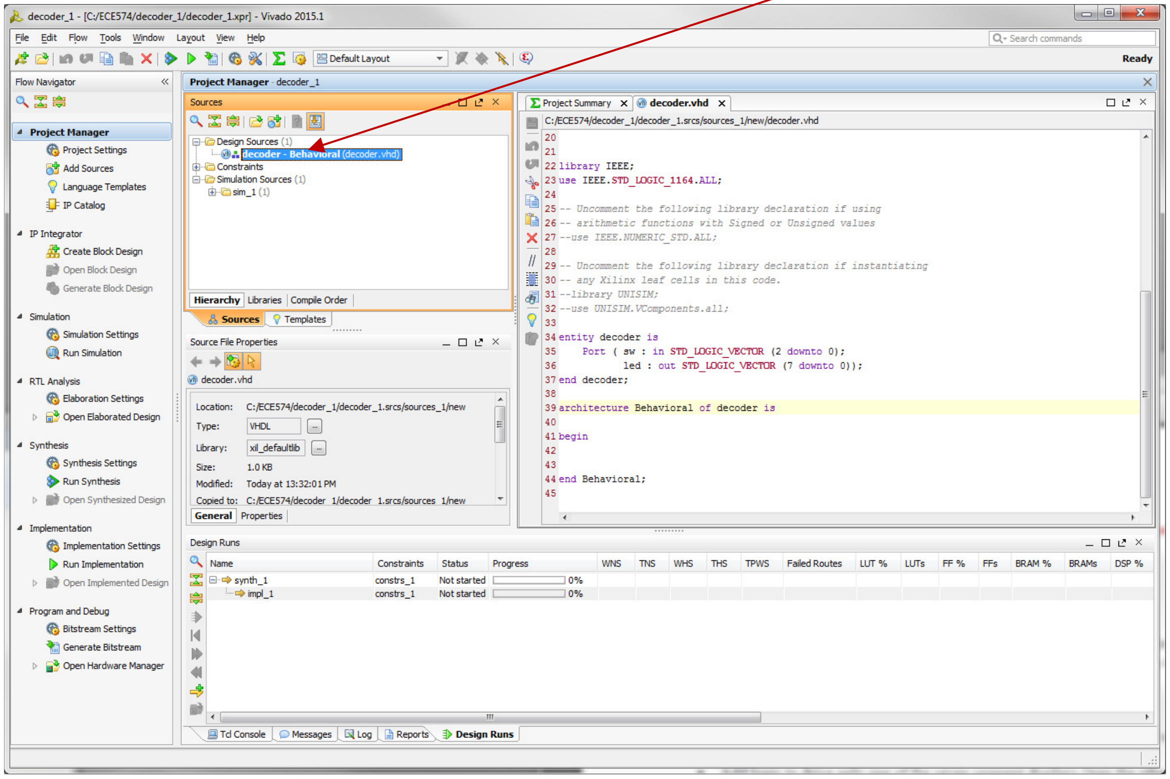
Architecture name: Behavioral

I/O Port Definitions

Port Name	Direction	Bus	MSB	LSB
SW	in	✓	2	0
LED	out	✓	7	0

Click OK.

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



decoder_1 - [C:/ECE574/decoder_1/decoder_1.xpr] - Vivado 2015.1

Design Sources (1)
decoder - Behavioral (decoder.vhd)
Constraints
Simulation Sources (1)
sim_1 (1)

C:/ECE574/decoder_1/decoder_1.srcs/sources_1/new/decoder.vhd

```


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity decoder is
    Port ( sw : in STD_LOGIC_VECTOR (2 downto 0);
           led : out STD_LOGIC_VECTOR (7 downto 0));
end decoder;

architecture Behavioral of decoder is

begin


end Behavioral;

```

Source File Properties

decoder.vhd

Location: C:/ECE574/decoder_1/decoder_1.srcs/sources_1/new
Type: VHDL
Library: xil_defaultlib
Size: 1.0 KB
Modified: Today at 13:32:01 PM
Copied to: C:/ECE574/decoder_1/decoder_1.srcs/sources_1/new

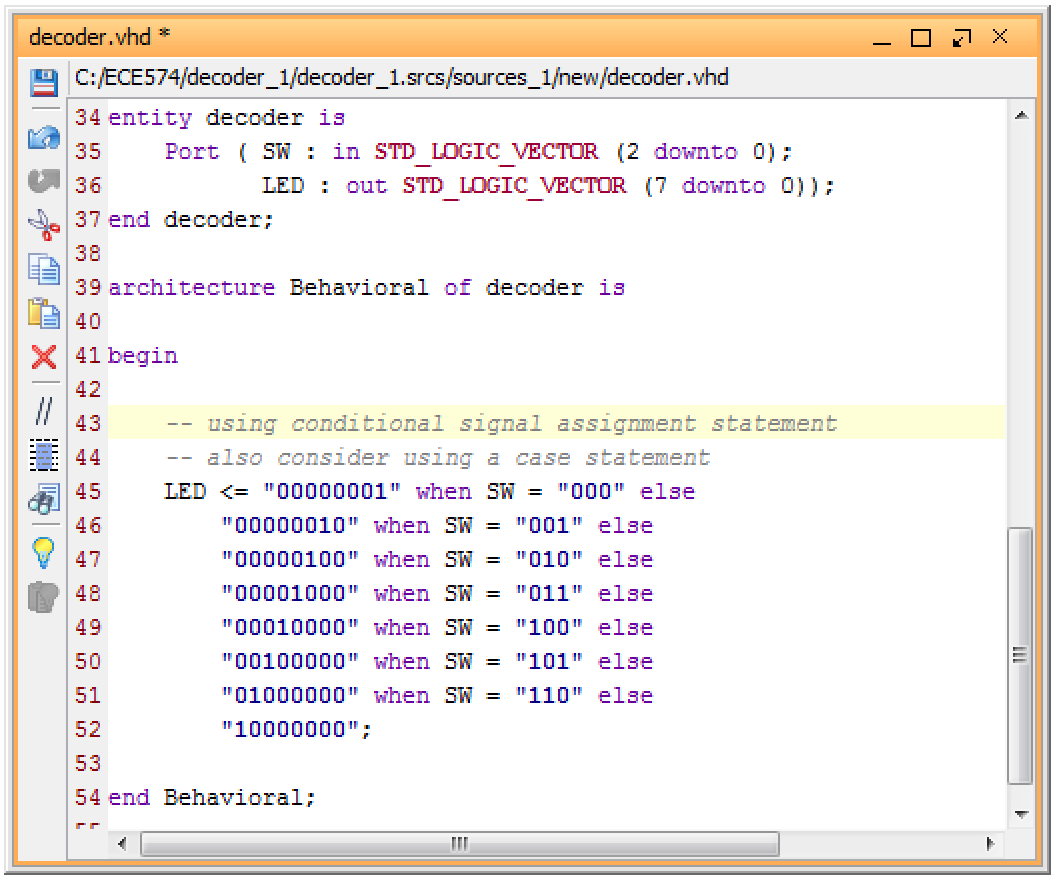
Design Runs

Name	Constraints	Status	Progress	WNS	TNS	WHS	THS	TPWS	Failed Routes	LUT %	LUTs	FF %	FFs	BRAM %	BRAMs	DSP %
synth_1	constrs_1	Not started	0%													
impl_1	constrs_1	Not started	0%													

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 a number of ways to design the decoder in VHDL, below is an example using a conditional signal assigment statement:


decoder.vhd *
C:/ECE574/decoder_1/decoder_1.srcs/sources_1/new/decoder.vhd

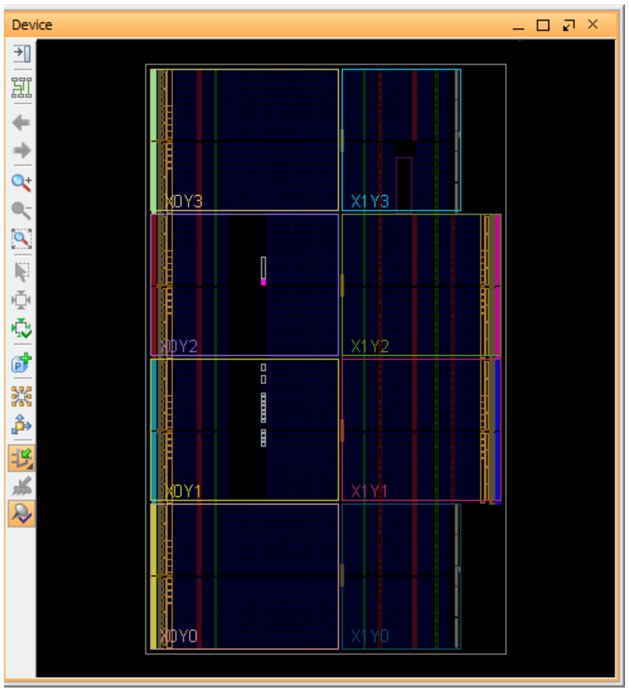
```vhdl
entity decoder is
    Port ( SW : in STD_LOGIC_VECTOR (2 downto 0);
           LED : out STD_LOGIC_VECTOR (7 downto 0));
end decoder;

architecture Behavioral of decoder is

begin

    -- using conditional signal assignment statement
    -- also consider using a case statement
    LED <= "00000001" when SW = "000" else
        "00000010" when SW = "001" else
        "00000100" when SW = "010" else
        "00001000" when SW = "011" else
        "00010000" when SW = "100" else
        "00100000" when SW = "101" else
        "01000000" when SW = "110" else
        "10000000";

end Behavioral;
```

Now we can synthesize the design.

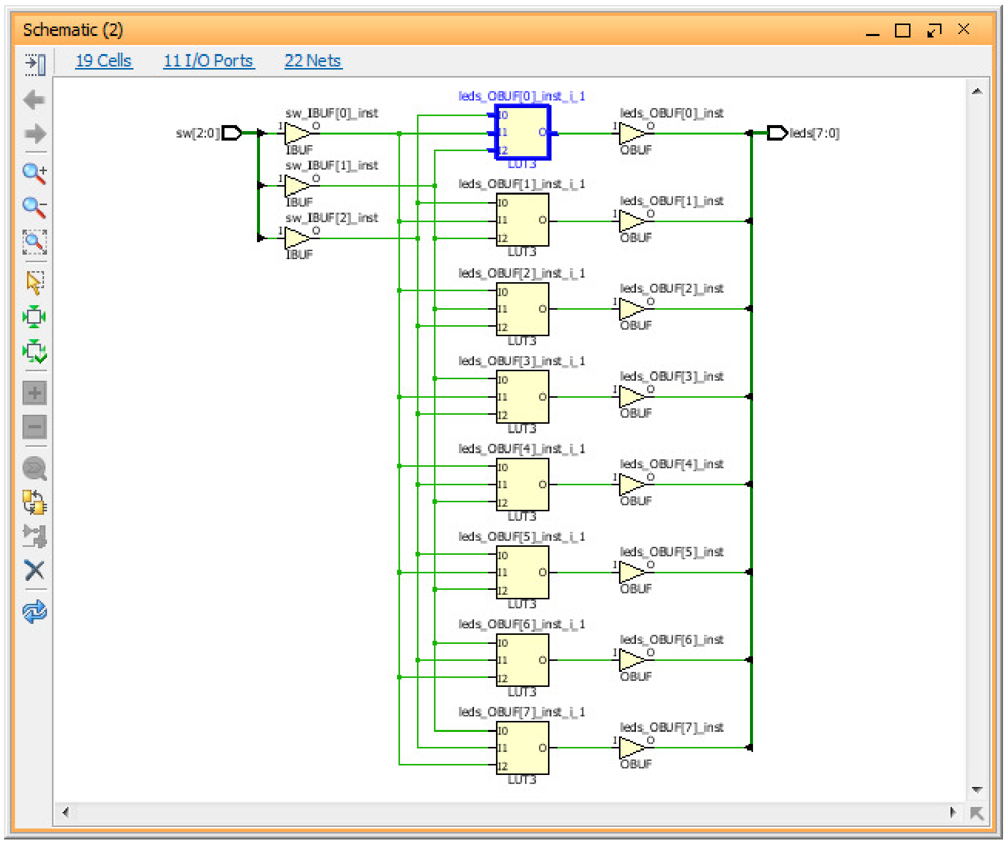
Click Run Synthesis in the Project Manager window.

After synthesis is complete there should be no errors or warnings reported (look at the Porject Summary window).

If you click on Open 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):



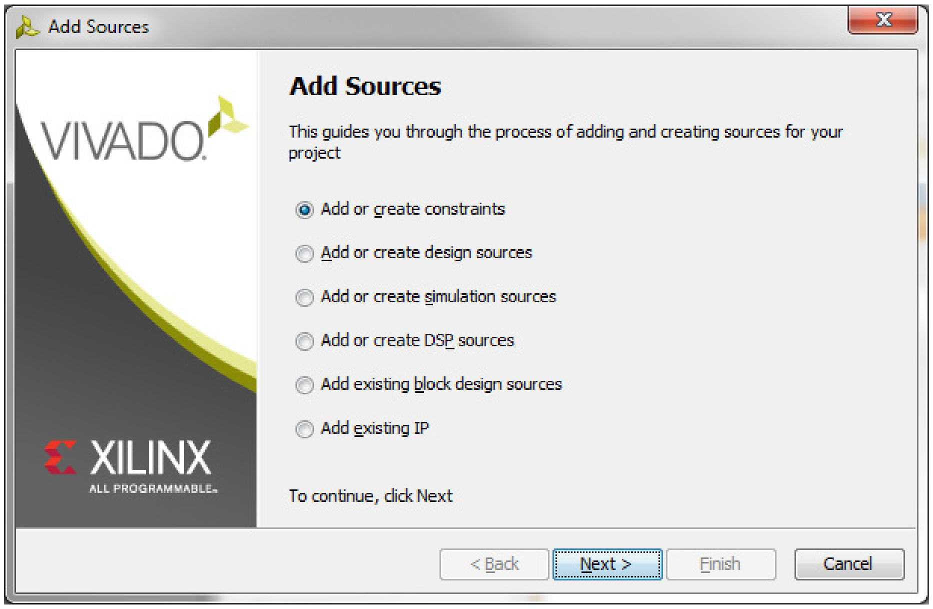
But you can also look at a schematic representation to see the input and output buffers and the LUTs used (click on Schematic under the Synthesized Design):



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 Nexys4DDR manual to determine the FPGA pins.

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

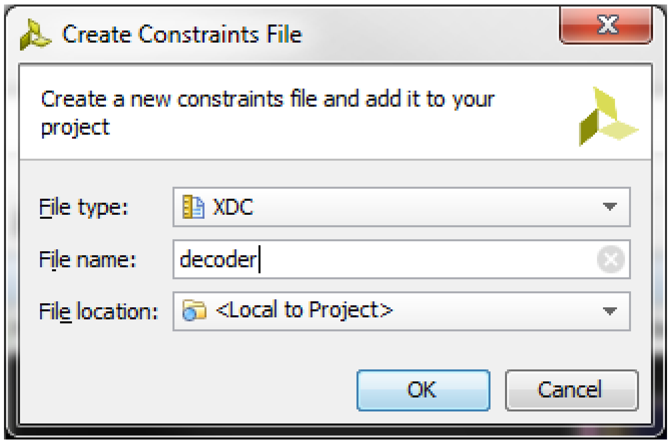

Add Sources

This guides you through the process of adding and creating sources for your project

(●) Add or create constraints
( ) Add or create design sources
( ) Add or create simulation sources
( ) Add or create DSP sources
( ) Add existing block design sources
( ) Add existing IP

To continue, click Next

We can name the constraints file decoder:

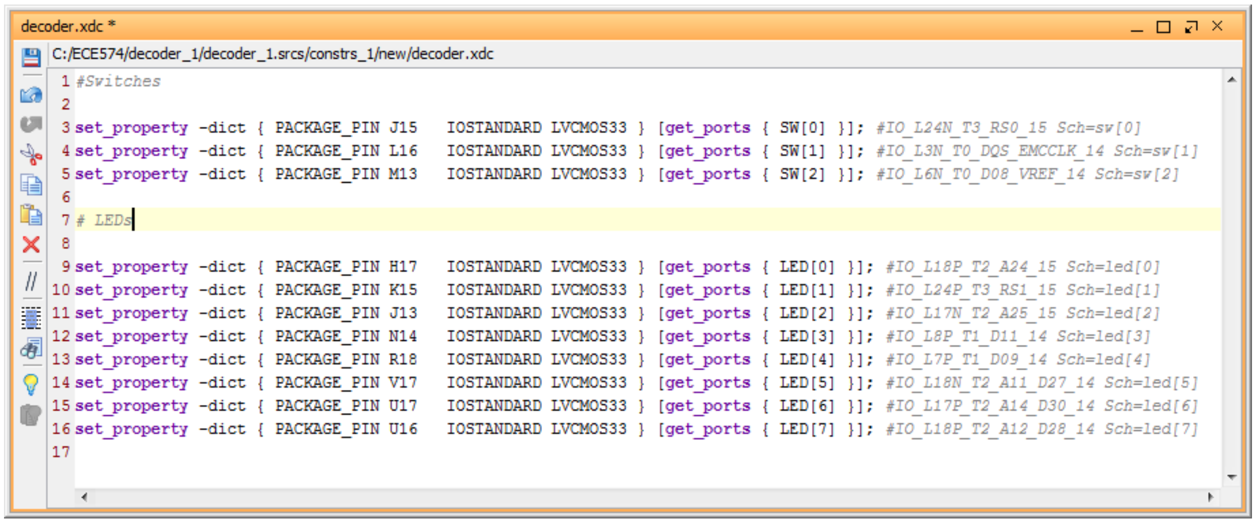

Create Constraints File

Create a new constraints file and add it to your project

File type: XDC
File name: decoder
File location: <Local to Project>

In the Souces 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 Nexys4DDR 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:/ECE574/decoder_1/decoder_1.srcs/constrs_1/new/decoder.xdc

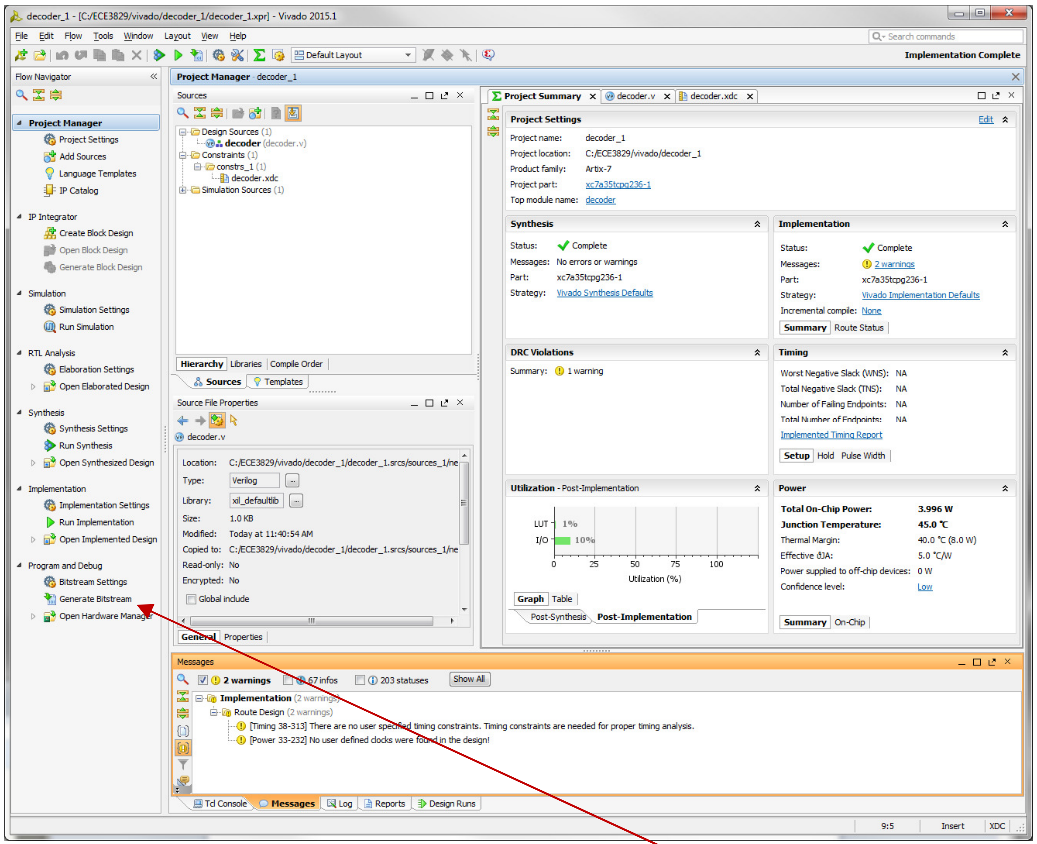
```
#Switches

set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { SW[0] }]; #IO_L24N_T3_RS0_15 Sch=sw[0]
set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 } [get_ports { SW[1] }]; #IO_L3N_T0_DQS_EMCCLK_14 Sch=sw[1]
set_property -dict { PACKAGE_PIN M13   IOSTANDARD LVCMOS33 } [get_ports { SW[2] }]; #IO_L6N_T0_D08_VREF_14 Sch=sw[2]

# LEDs

set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 } [get_ports { LED[0] }]; #IO_L18P_T2_A24_15 Sch=led[0]
set_property -dict { PACKAGE_PIN K15   IOSTANDARD LVCMOS33 } [get_ports { LED[1] }]; #IO_L24P_T3_RS1_15 Sch=led[1]
set_property -dict { PACKAGE_PIN J13   IOSTANDARD LVCMOS33 } [get_ports { LED[2] }]; #IO_L17N_T2_A25_15 Sch=led[2]
set_property -dict { PACKAGE_PIN N14   IOSTANDARD LVCMOS33 } [get_ports { LED[3] }]; #IO_L8P_T1_D11_14 Sch=led[3]
set_property -dict { PACKAGE_PIN R18   IOSTANDARD LVCMOS33 } [get_ports { LED[4] }]; #IO_L7P_T1_D09_14 Sch=led[4]
set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports { LED[5] }]; #IO_L18N_T2_A11_D27_14 Sch=led[5]
set_property -dict { PACKAGE_PIN U17   IOSTANDARD LVCMOS33 } [get_ports { LED[6] }]; #IO_L17P_T2_A14_D30_14 Sch=led[6]
set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports { LED[7] }]; #IO_L18P_T2_A12_D28_14 Sch=led[7]
```

Now we have specified the correct pins to use for the design (so it matches the Nexys4DDR board layout) we can implement the design.

NOTE: although VHDL is not case sensitive the XDC constraints file is case sensitive and the port names need to match the port names in the VHDL entity.

Click Run Implementation under Implementation in the Flow Navigator 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 these warnings.



Project Settings

Project name:	decoder_1
Project location:	C:/ECE3829/vivado/decoder_1
Product family:	Artix-7
Project part:	xc7a35tcpg236-1
Top module name:	decoder

Synthesis

Status:	✓ Complete
Messages:	No errors or warnings
Part:	xc7a35tcpg236-1
Strategy:	Vivado Synthesis Defaults

Implementation

Status:	✓ Complete
Messages:	2 warnings
Part:	xc7a35tcpg236-1
Strategy:	Vivado Implementation Defaults
Incremental compile:	None

DRC Violations

Summary: 1 warning

Timing

Worst Negative Slack (WNS):	NA
Total Negative Slack (TNS):	NA
Number of Failing Endpoints:	NA
Total Number of Endpoints:	NA

Implemented Timing Report

Utilization - Post-Implementation

LUT	1%
I/O	10%

Power

Total On-Chip Power:	3.996 W
Junction Temperature:	45.0 ℃
Thermal Margin:	40.0 ℃ (8.0 W)
Effective ϑJA:	5.0 ℃/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Source File Properties: decoder.v

Location:	C:/ECE3829/vivado/decoder_1/decoder_1.srcs/sources_1/ne
Type:	Verilog
Library:	xil_defaultlib
Size:	1.0 KB
Modified:	Today at 11:40:54 AM
Copied to:	C:/ECE3829/vivado/decoder_1/decoder_1.srcs/sources_1/ne
Read-only:	No
Encrypted:	No

Messages: 2 warnings, 67 infos, 203 statuses

```
Implementation (2 warnings)
  Route Design (2 warnings)
    [Timing 38-313] There are no user specified timing constraints. Timing constraints are needed for proper timing analysis.
    [Power 33-232] No user defined clocks were found in the design!
```

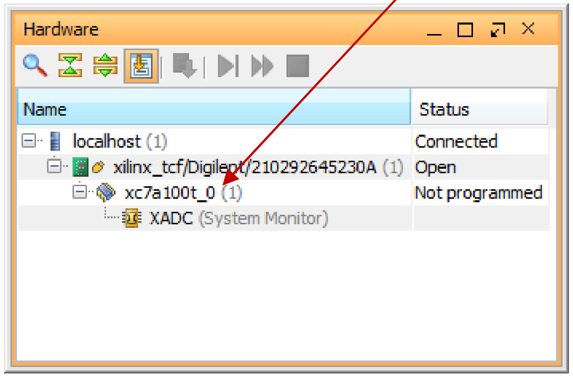
We can now generate the bitstream. Click on Generate Bitstream in the Flow Navigator window.

Next step is to program the FPGA with the bitstream (you will notice another warning message regarding configuration voltage – ignore this for now)

On the Nexys4DDR board make sure that JP1 Mode Jumper is set to the JTAG mode and connect the USB cable to the board and turn on the power.

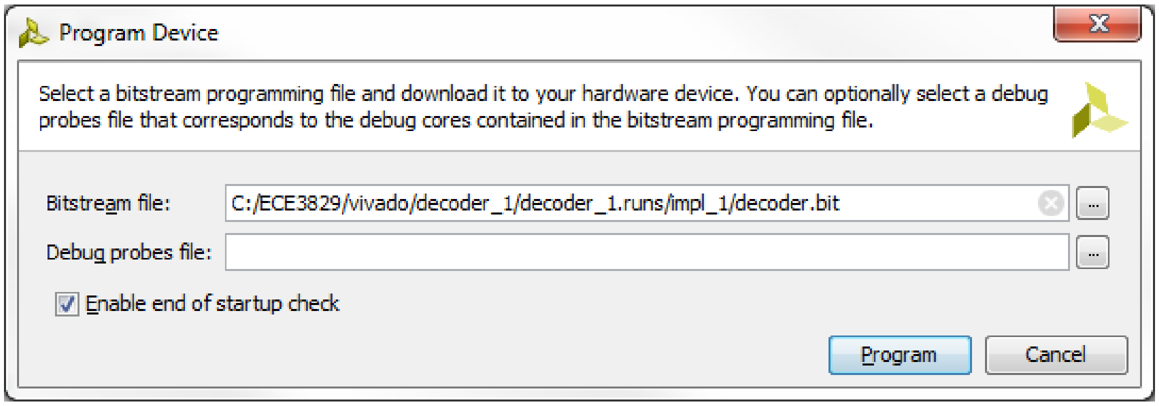
Select the Hardware Manager in the Flow Navigator (under Program and Devug) and select Open Target and then Auto Connect.

You should now see the Xilinx xc7a100t_0 in the Hardware Window:



Name	Status
localhost (1)	Connected
xilinx_tcf/Digilent/210292645230A (1)	Open
xc7a100t_0 (1)	Not programmed
XADC (System Monitor)	

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


Program Device

Select a bitstream programming file and download it to your hardware device. You can optionally select a debug probes file that corresponds to the debug cores contained in the bitstream programming file.

Bitstream file: C:/ECE3829/vivado/decoder_1/decoder_1.runs/impl_1/decoder.bit

Debug probes file:

[✓] Enable end of startup check

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 Nexys4DDR 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.

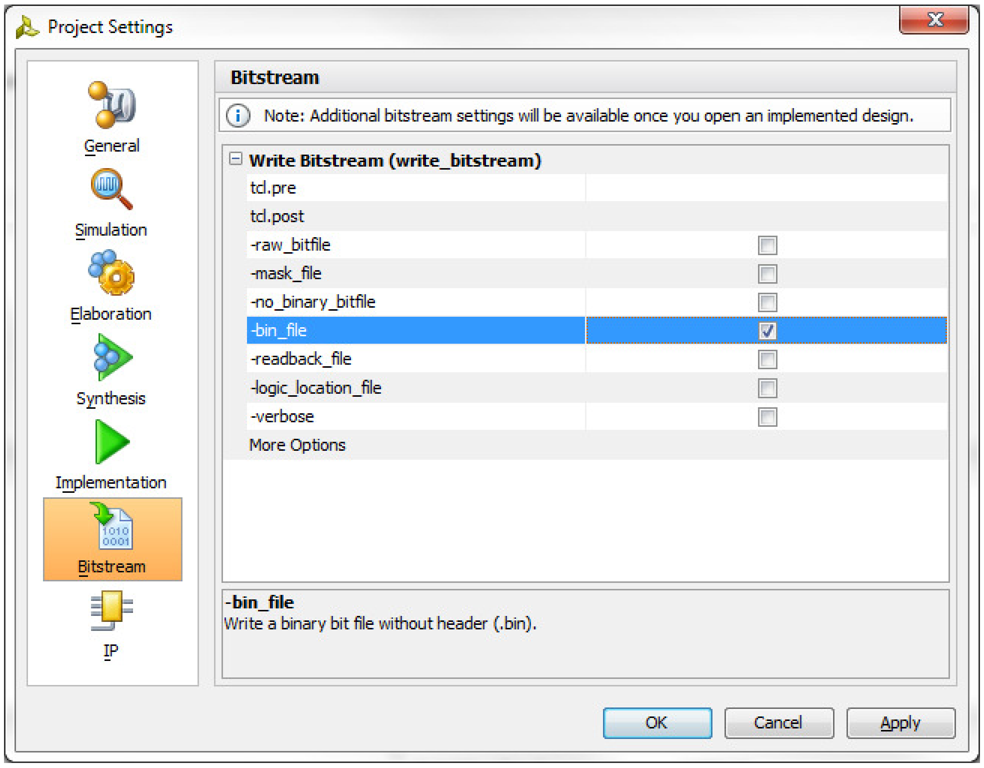
Programming the Serial Flash

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 Nexys4DDR 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 Flow Navigator.

Select the bin_file option:



Bitstream

Note: Additional bitstream settings will be available once you open an implemented design.

Write Bitstream (write_bitstream)

tcl.pre	
tcl.post	
-raw_bitfile	☐
-mask_file	☐
-no_binary_bitfile	☐
-bin_file	✓
-readback_file	☐
-logic_location_file	☐
-verbose	☐
More Options	

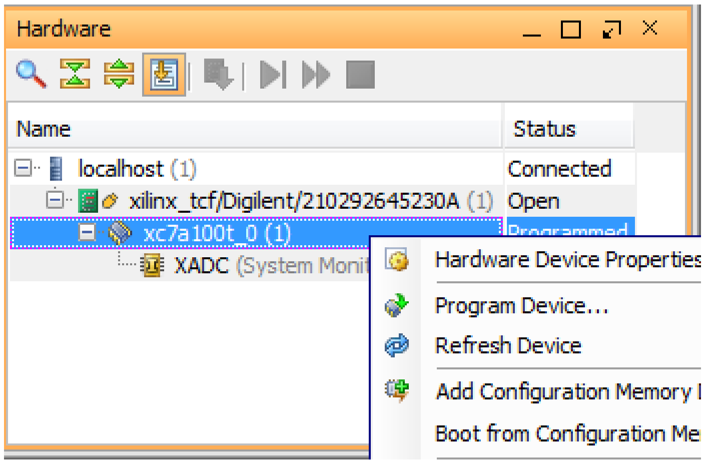
-bin_file
Write a binary bit file without header (.bin).

Click OK.

Click on Generate Bitstream.

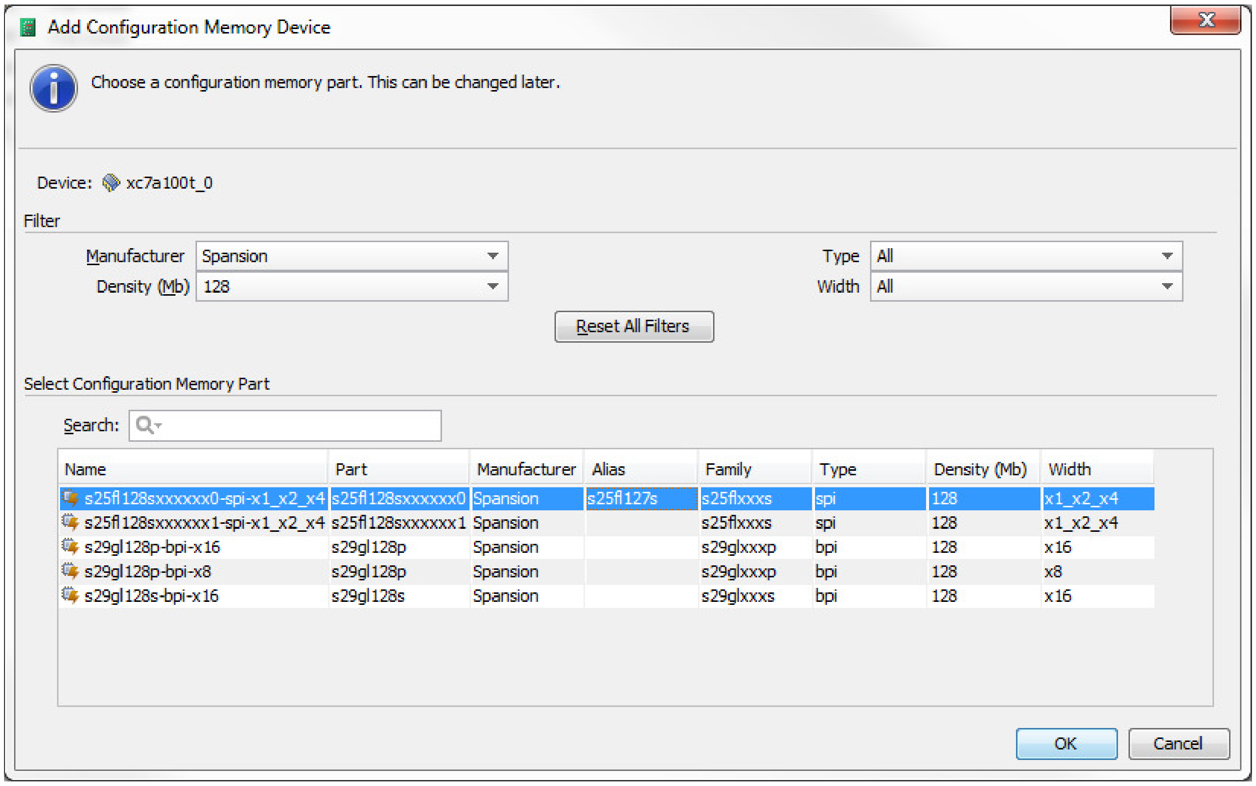
After generation, if you look in the decoder_1 => decoder_1.runs => impl_1 directory you will see a decoder.bit and a decoder.bin file.

Use the Hardware Manager to connect to the Nexys4DDR board then right-mouse click on the FPGA and select Add Configuration Memory Device:


Hardware
Name
Status
localhost (1)
Connected
xilinx_tcf/Digilent/210292645230A (1)
Open
xc7a100t_0 (1)
Programmed
XADC (System Monit
Hardware Device Properties
Program Device…
Refresh Device
Add Configuration Memory I
Boot from Configuration Me

We now need to select the serial flash device that is on theNexys4DDR board.

Select Spansion as the Manufacturer, then select the 128Mb device (S25FL128S):



Choose a configuration memory part. This can be changed later.

Device: xc7a100t_0

Filter

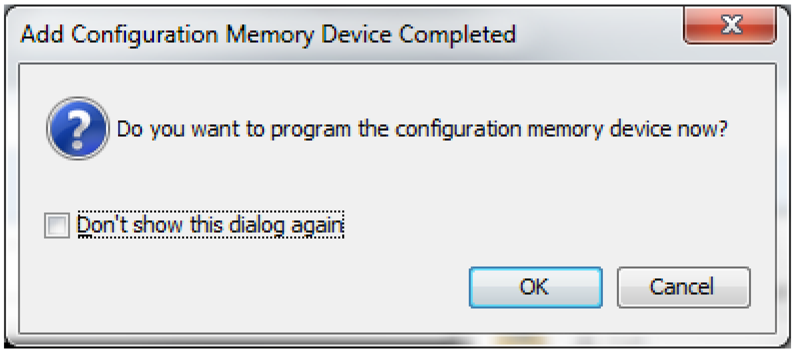
Manufacturer: Spansion
Density (Mb): 128
Type: All
Width: All

Select Configuration Memory Part

Search:

Name	Part	Manufacturer	Alias	Family	Type	Density (Mb)	Width
s25fl128sxxxxxxx0-spi-x1_x2_x4	s25fl128sxxxxxxx0	Spansion	s25fl127s	s25flxxxs	spi	128	x1_x2_x4
s25fl128sxxxxxxx1-spi-x1_x2_x4	s25fl128sxxxxxxx1	Spansion		s25flxxxs	spi	128	x1_x2_x4
s29gl128p-bpi-x16	s29gl128p	Spansion		s29glxxxp	bpi	128	x16
s29gl128p-bpi-x8	s29gl128p	Spansion		s29glxxxp	bpi	128	x8
s29gl128s-bpi-x16	s29gl128s	Spansion		s29glxxxs	bpi	128	x16

Click OK



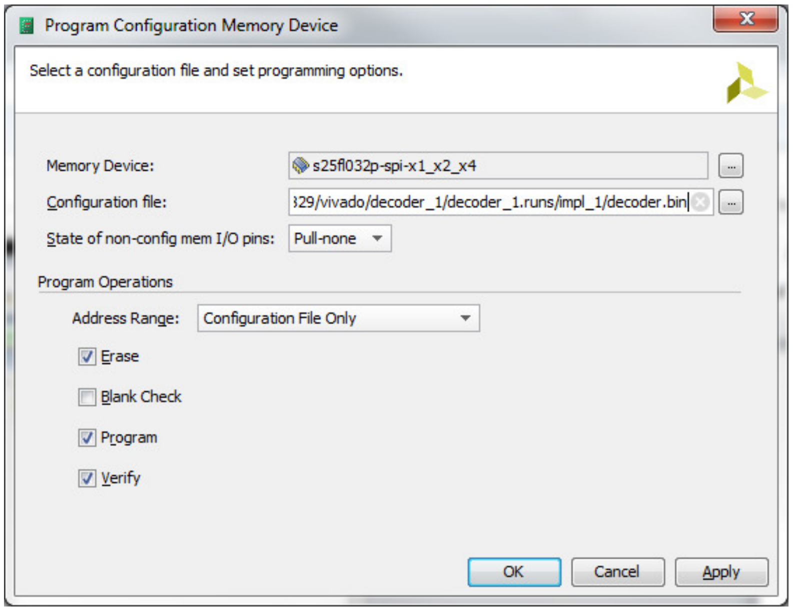
Add Configuration Memory Device Completed

Do you want to program the configuration memory device now?

☐ Don't show this dialog again

Click OK

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



Program Configuration Memory Device

Select a configuration file and set programming options.

Memory Device: s25fl032p-spi-x1_x2_x4

Configuration file: 329/vivado/decoder_1/decoder_1.runs/impl_1/decoder.bin

State of non-config mem I/O pins: Pull-none

Program Operations

Address Range: Configuration File Only

☑ Erase

☐ Blank Check

☑ Program

☑ Verify

Click OK.

Note: this will erase any existing design in the QSPI flash (including the configuration file shipped with the Nexys4DDR board).

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

Once programmed, you can power off the Nexys4DDR 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.