THE AMERICAN UNIVERSITY IN CAIRO

الجـامـعـة الأمـريكيـة بالقـاهـرة

Department of Computer Science and Engineering

CSCE2302:Digital Design Lab
Vivado Tutorial

Fall 2018

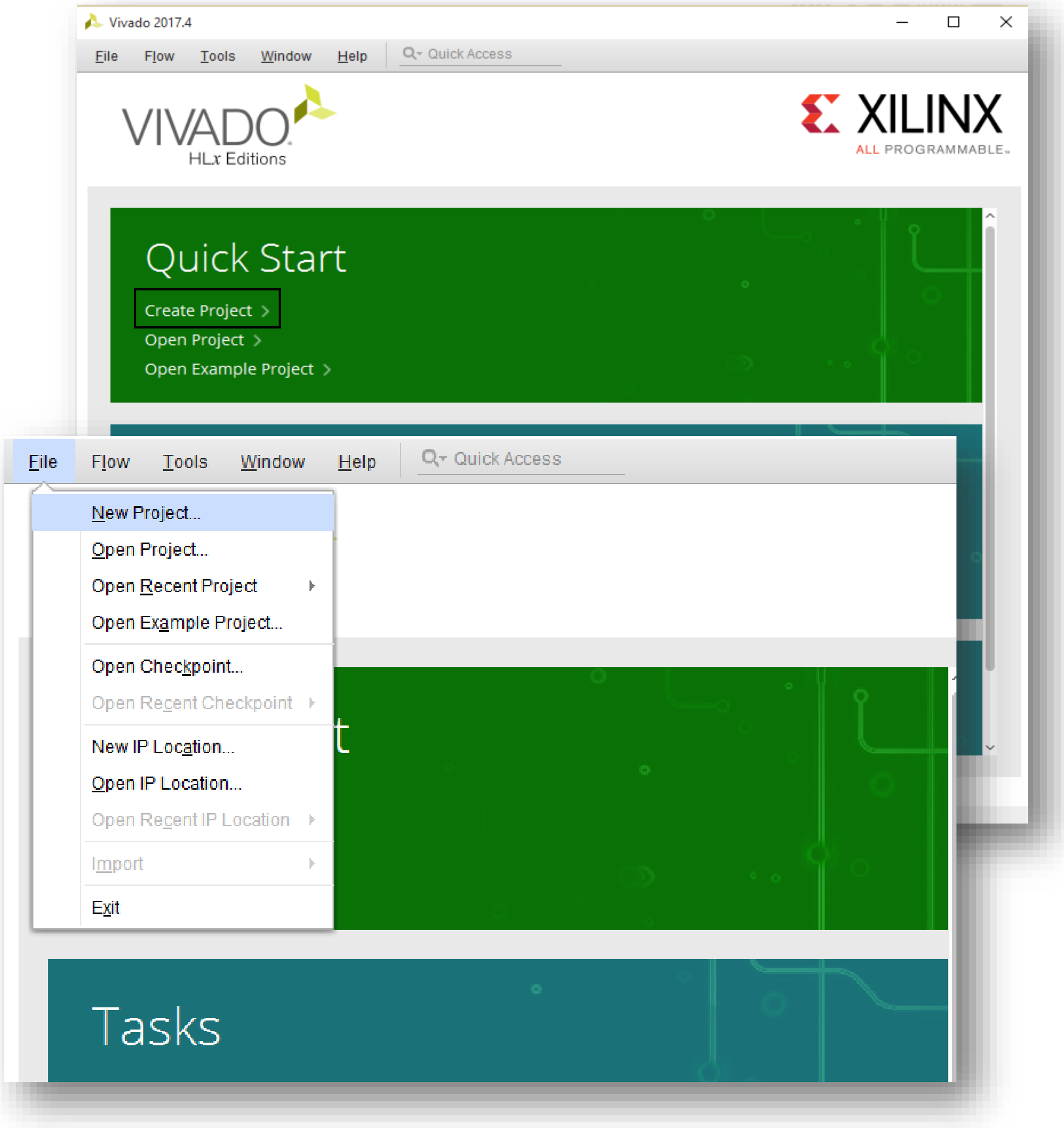
Contents

Create a New Project

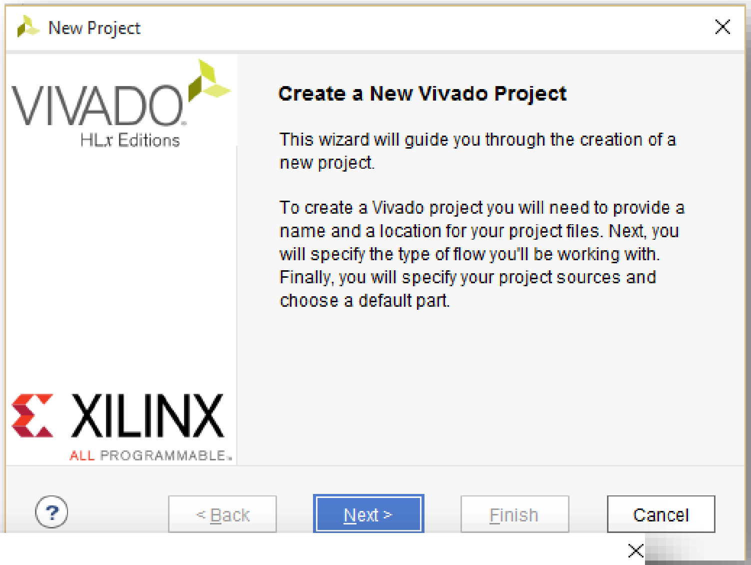
1. In the Vivado IDE Getting started page, click Create Project

Or alternatively

1. Click on **File** and then **New Project**



2. A dialog box will appear → click Next >

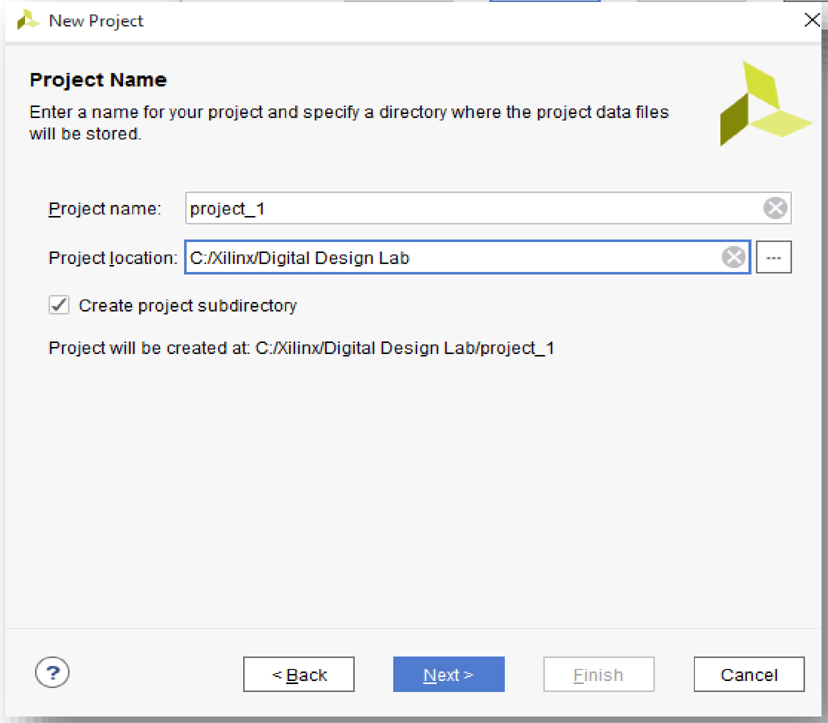

New Project

Create a New Vivado Project

This wizard will guide you through the creation of a new project.

To create a Vivado project you will need to provide a name and a location for your project files. Next, you will specify the type of flow you'll be working with. Finally, you will specify your project sources and choose a default part.

3. In the **New Project** dialog box:
   a. Change your **Project name** as needed
   b. Choose where you want to save your project in **Project location**.
   c. Project name, as well as the project location, must not have any spaces in it
   d. Click Next >


New Project

Project Name

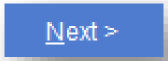
Enter a name for your project and specify a directory where the project data files will be stored.

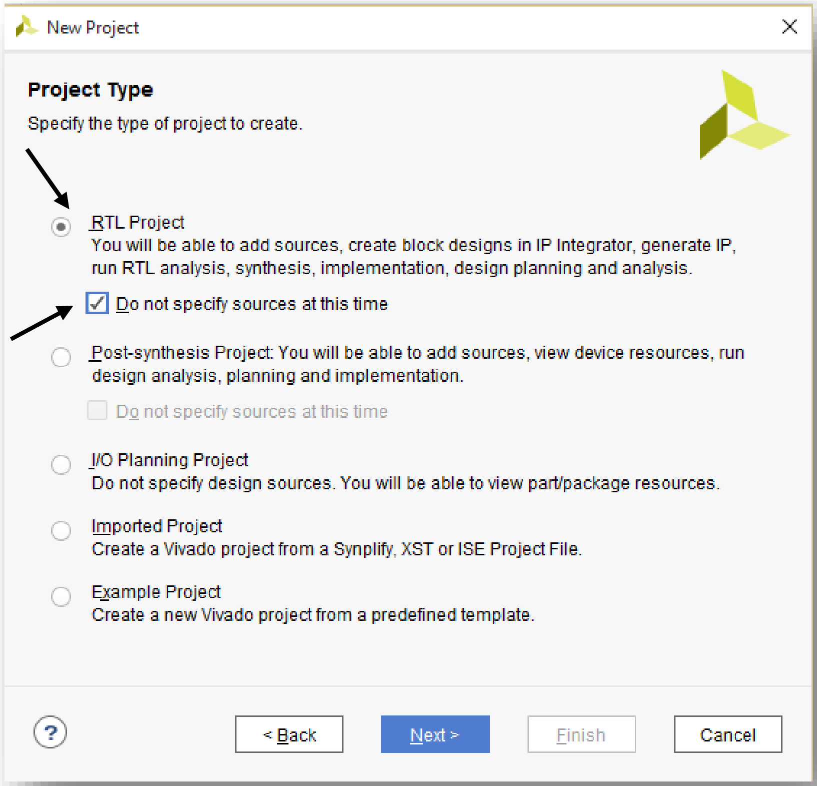
Project name: project_1

Project location: C:/Xilinx/Digital Design Lab

☑ Create project subdirectory

Project will be created at: C:/Xilinx/Digital Design Lab/project_1

4. In the **Project Type** dialog box:
   a. Select **RTL Project**
   b. Check the box: "Do not specify sources at this time"
   c. Click 



New Project

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.

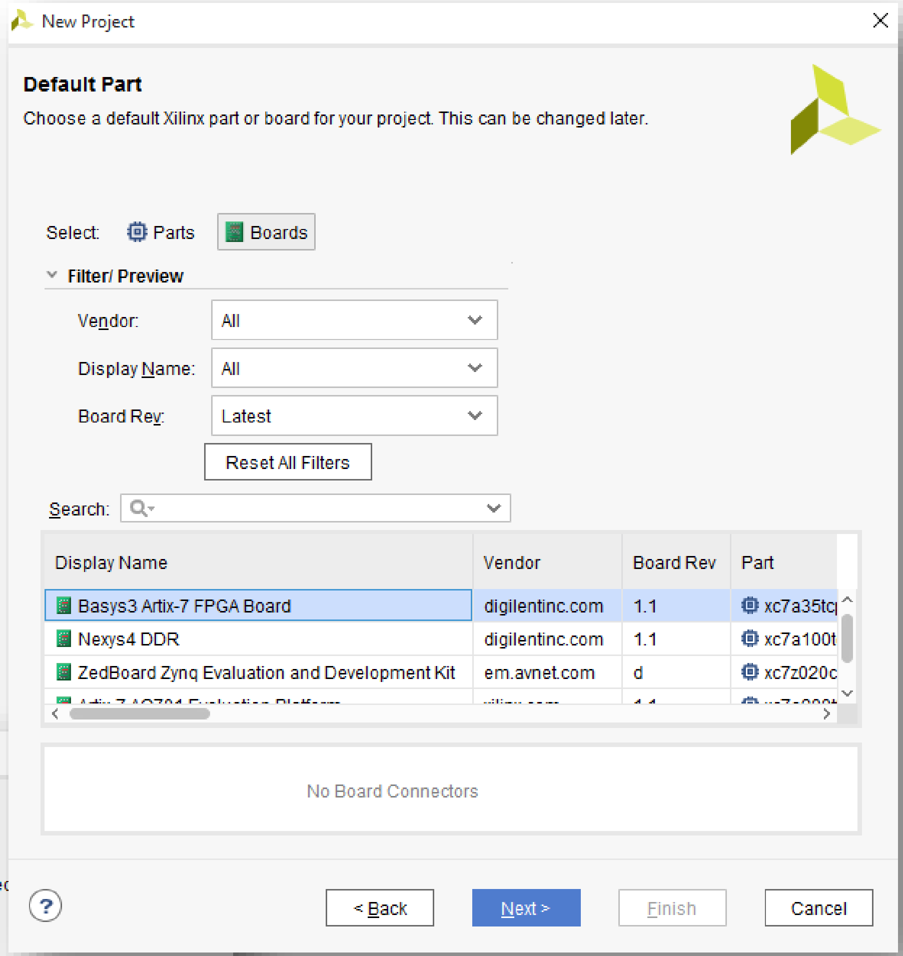
Do not specify sources at this time

I/O Planning Project
Do not specify design sources. You will be able to view part/package resources.

Imported Project
Create a Vivado project from a Synplify, XST or ISE Project File.

Example Project
Create a new Vivado project from a predefined template.

5. In the **Default Part** dialog box:
   a. Click on Boards
   b. Choose Basys3 Artix-7 FPGA Board
   c. Click Next >


New Project

Default Part

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

Select: Parts Boards

Filter/ Preview

Vendor: All

Display Name: All

Board Rev: Latest

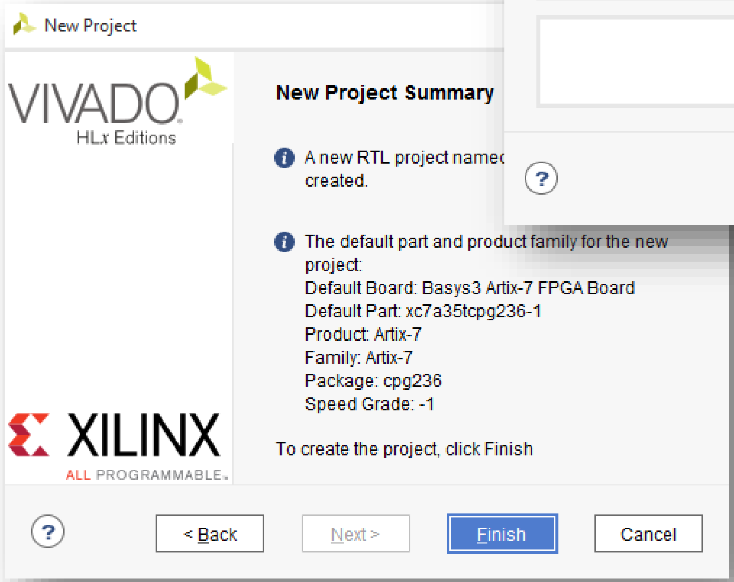
Reset All Filters

Search:

Display Name	Vendor	Board Rev	Part
Basys3 Artix-7 FPGA Board	digilentinc.com	1.1	xc7a35tc
Nexys4 DDR	digilentinc.com	1.1	xc7a100t
ZedBoard Zynq Evaluation and Development Kit	em.avnet.com	d	xc7z020c

No Board Connectors

6. In the new project Summary, click Finish


New Project

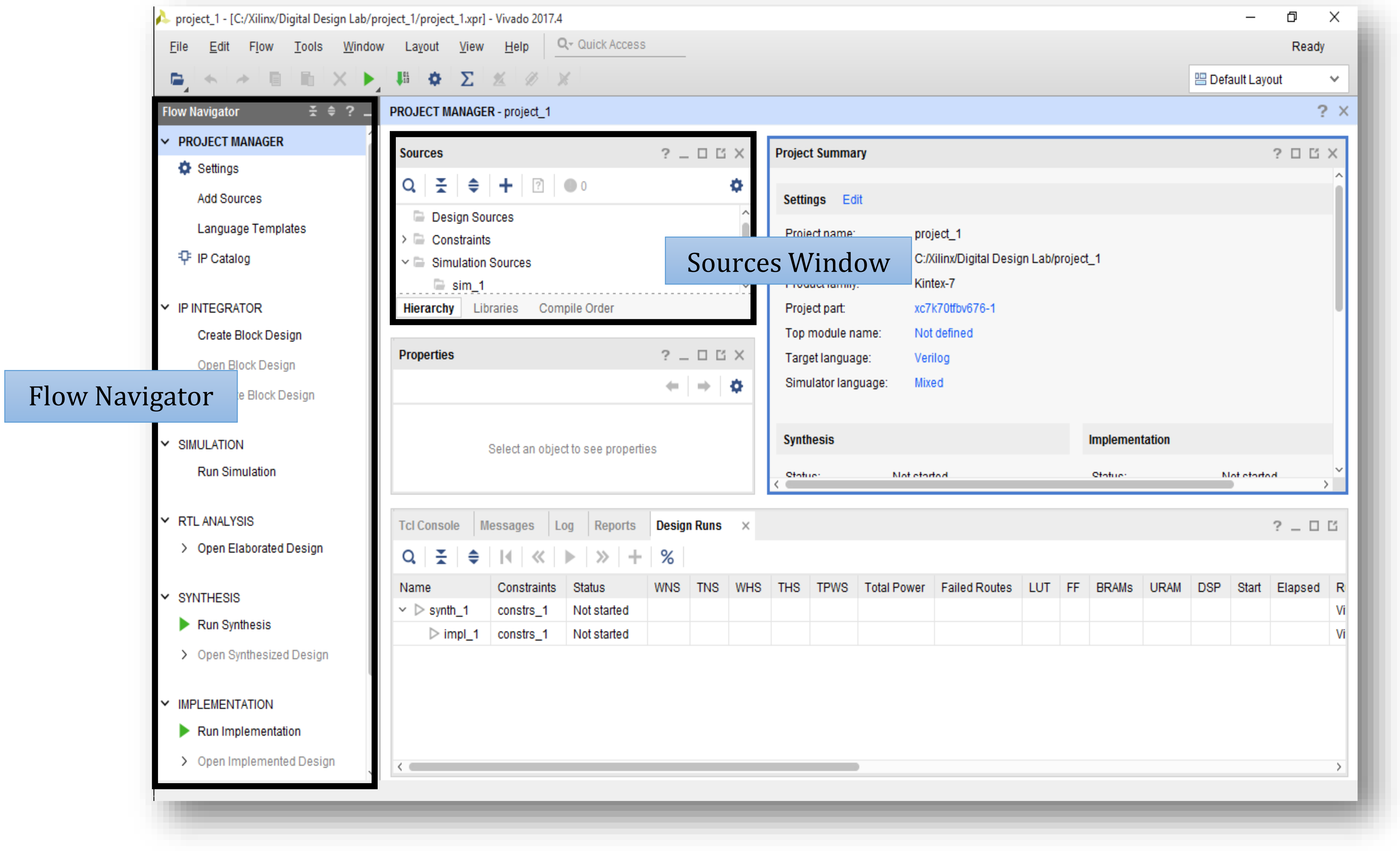
New Project Summary

A new RTL project named created.

The default part and product family for the new project:
Default Board: Basys3 Artix-7 FPGA Board
Default Part: xc7a35tcpg236-1
Product: Artix-7
Family: Artix-7
Package: cpg236
Speed Grade: -1

To create the project, click Finish

- After creating the project, Vivado opens the new project in the default view layout.



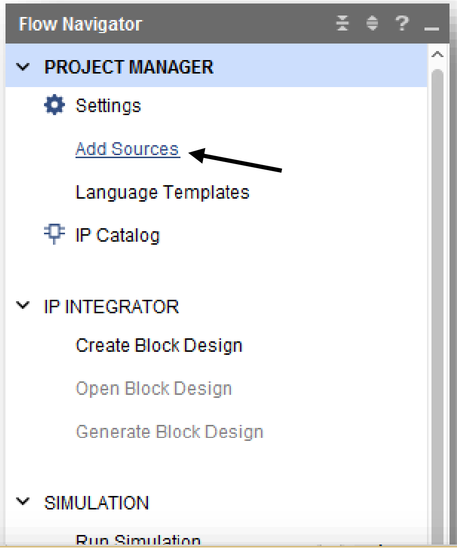
Project name: project_1
Project location: C:/Xilinx/Digital Design Lab/project_1
Product family: Kintex-7
Project part: xc7k70tfbv676-1
Top module name: Not defined
Target language: Verilog
Simulator language: Mixed

Synthesis Status: Not started
Implementation Status: Not started

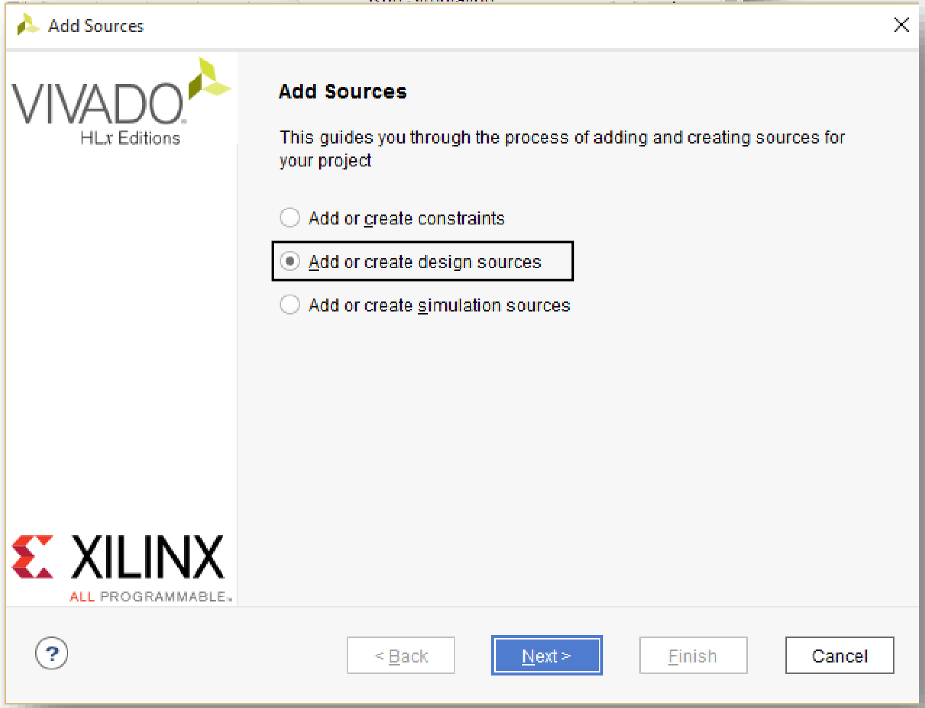
Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP	Start	Elapsed	R
synth_1	constrs_1	Not started															Vi
impl_1	constrs_1	Not started															Vi

Create a new Source File

1. In the **Project Manager** Window, press on **Add sources**



2. In the **Add Sources** Window, mark "Add or create design sources then click Next >


Add Sources

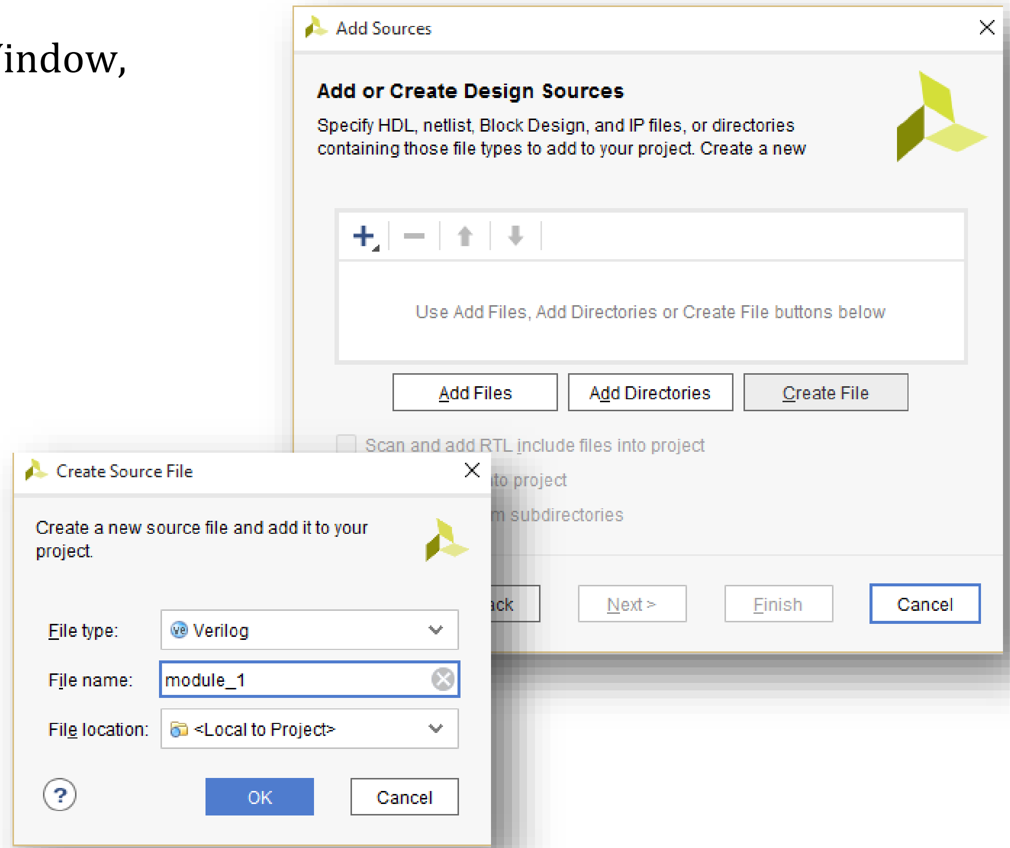
Add Sources

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

3. In the **Add or Create Design Sources** Window, Click on "Create File"

4. Adjust the properties of the **Create Source File** window as follows:
   - **File type**: Verilog
   - **File name**: <your_module_name> must not have any spaces
   - **File location**: Local to Project



Add Sources

Add or Create Design Sources

Specify HDL, netlist, Block Design, and IP files, or directories containing those file types to add to your project. Create a new

Use Add Files, Add Directories or Create File buttons below

Add Files | Add Directories | Create File

Scan and add RTL include files into project

to project

m subdirectories

ck | Next > | Finish | Cancel

Create Source File

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

File type: Verilog

File name: module_1

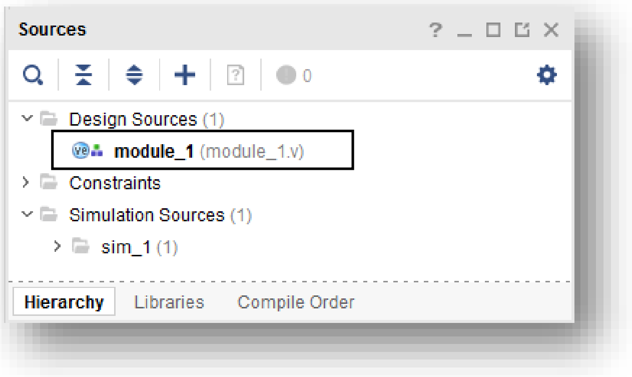
File location: <Local to Project>

OK | Cancel

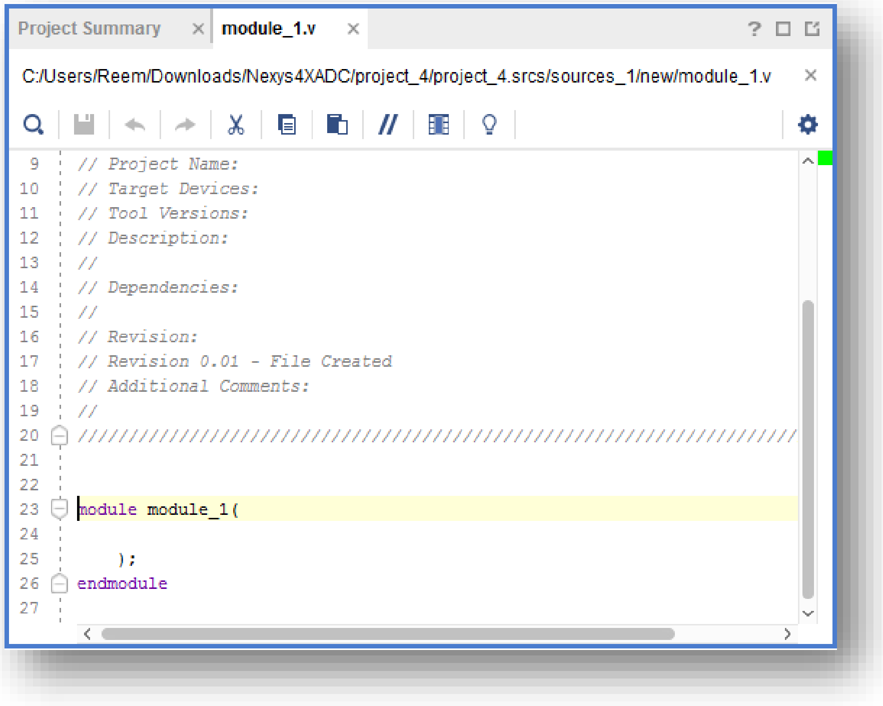
5. Click Finish

6. A Define Module window will appear where you can optionally define your inputs and outputs. We will skip this now. So, Press OK and then Yes

7. In the **Sources** Window, your module will appear



8. Double click on the module, A file with your module name and extension .v will appear where you can write your module

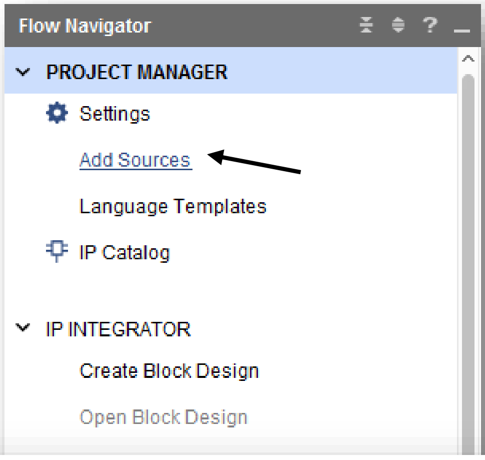

Project Summary
module_1.v
C:/Users/Reem/Downloads/Nexys4XADC/project_4/project_4.srcs/sources_1/new/module_1.v

```
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////


module module_1(

    );
endmodule
```

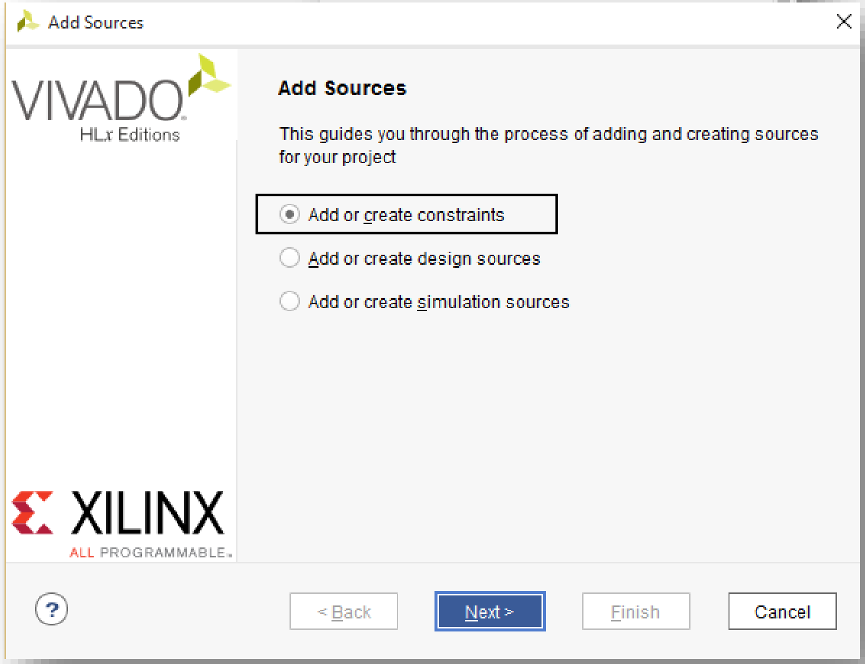
Write a Constraint File

A constraint file is used to link the input and output signals in yout module to the certain pins in the FPGA
You cannot connect an internal signal (a wire/ a reg) to the FPGA

1. Create a new Constraint File
   a. In the **Project Manager** Window, press on **Add sources**



   b. In the **Add Sources** Window, mark "Add or create Constraints" then click Next >

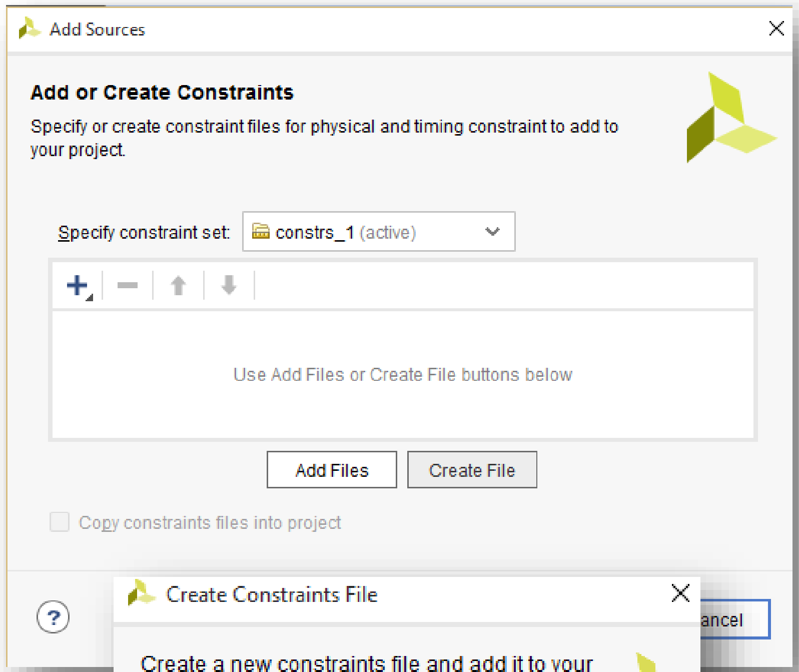


Add Sources

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

c. In the **Add or Create Constraints** Window Click on "Create File"



Add Sources

Add or Create Constraints

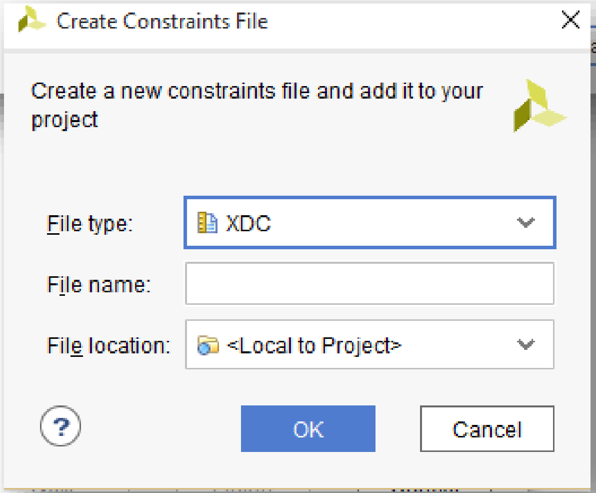
Specify or create constraint files for physical and timing constraint to add to your project.

Specify constraint set: constrs_1 (active)

Use Add Files or Create File buttons below

d. Adjust the properties of the **Create Constraints File** window as follows:

- **File type**: XDC
- **File name**: <your_file_name> must not have any spaces
- **File location**: Local to Project



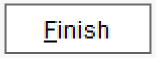
Create Constraints File

Create a new constraints file and add it to your project

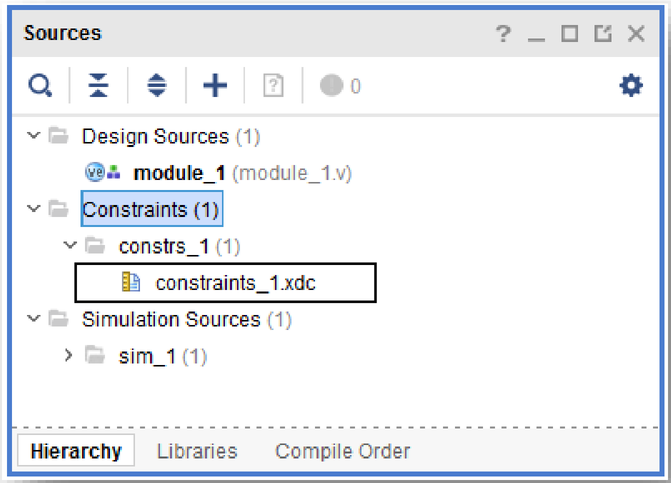
File type: XDC

File name:

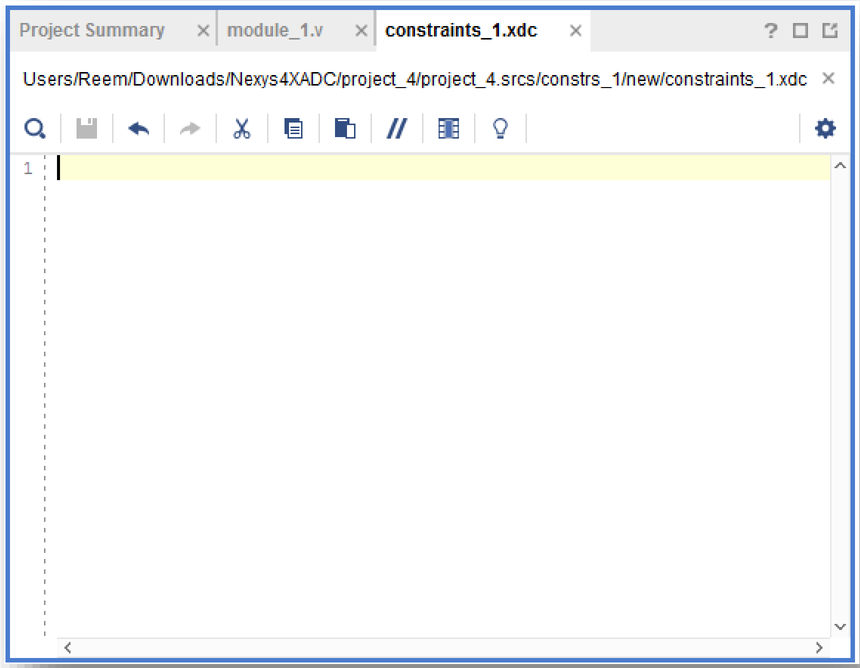
File location: <Local to Project>

e. Click 

f. Your constraint file will appear in the **Sources** window under the **Constraints** folder with extension .xdc



g. Double click on the .xdc, a window with your constraints file name and extension .xdc will appear where you can write your constraints (mapping from your Verilog module and the FPGA pins)


Project Summary
module_1.v
constraints_1.xdc
Users/Reem/Downloads/Nexys4XADC/project_4/project_4.srcs/constrs_1/new/constraints_1.xdc

2. Write Constraints

Syntax:

```
set_property PACKAGE_PIN <pin_number> [get_ports <signal>]
set_property IOSTANDARD LVCMOS33 [get_ports <same_signal>]
set_property PACKAGE_PIN <pin_number> [get_ports {<vector_element>}]
set_property IOSTANDARD LVCMOS33 [get_ports {same_vector_element}]
```

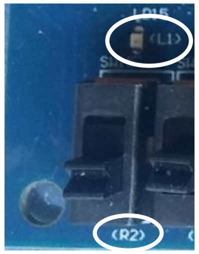
Example: Suppose you have a module:

```
module module_1 (input a, output b,output [1:0]c); // some code
endmodule
```

In the constraint file, write:

```
set_property PACKAGE_PIN R2 [get_ports a]
set_property IOSTANDARD LVCMOS33 [get_ports a]
set_property PACKAGE_PIN L1 [get_ports b]
set_property IOSTANDARD LVCMOS33 [get_ports b]
set_property PACKAGE_PIN P1 [get_ports {c[0]} ]
set_property IOSTANDARD LVCMOS33 [get_ports {c[0]} ]
set_property PACKAGE_PIN N3 [get_ports {c[1]} ]
set_property IOSTANDARD LVCMOS33 [get_ports {c[1]} ]
```

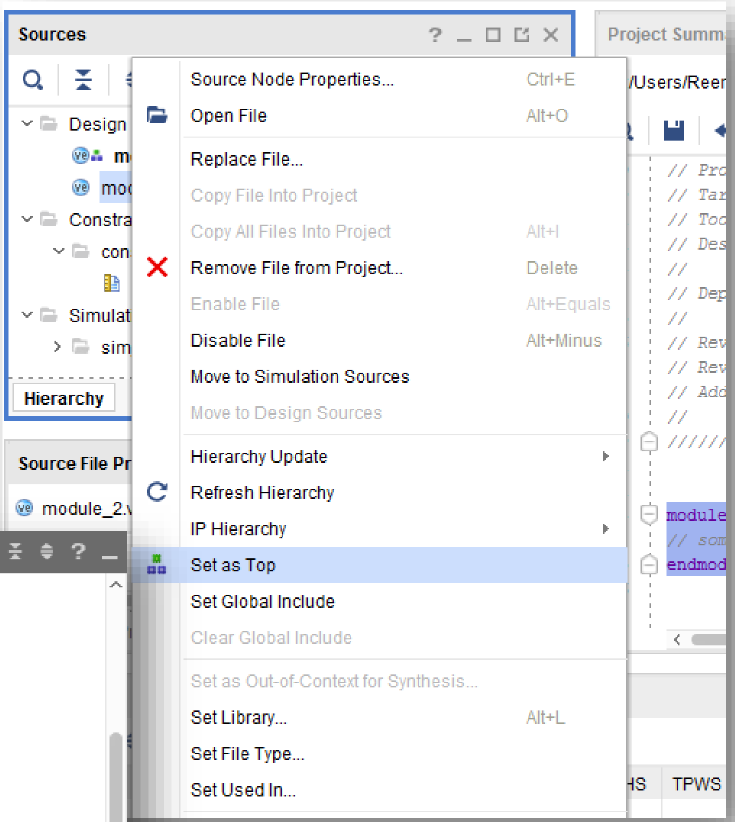
This mean signal a is connected to switch named **R2** and signal b is connected to the LED named **L1**
and the 2-bit signal c is connected to the LEDs named **P1** and **N3**

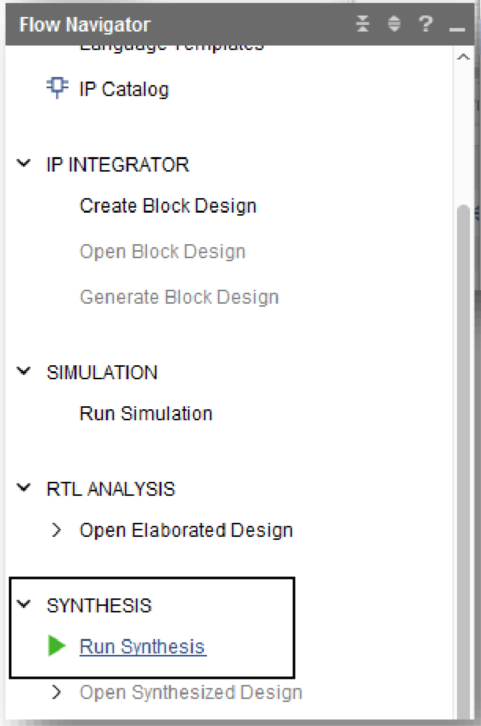


Implement a Design on the FPGA

1. Set a module as a top module
   a. Right click on the module that you want to implement on the FPGA
   b. In the drop-down menu, click on “Set as Top”
   c. The top module is marked by this sign

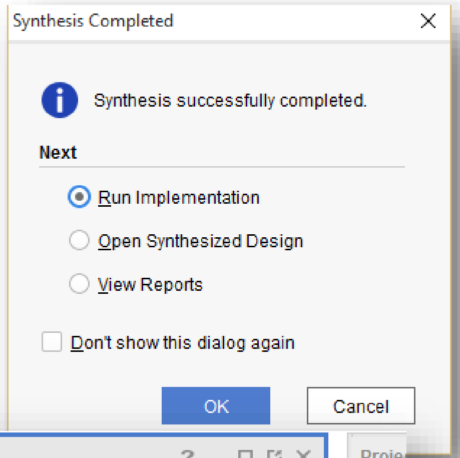
2. Synthesize your top module
   a. In the **Flow Navigator**, click on “SYNTHESIS”
   b. Click on “Run Synthesis”





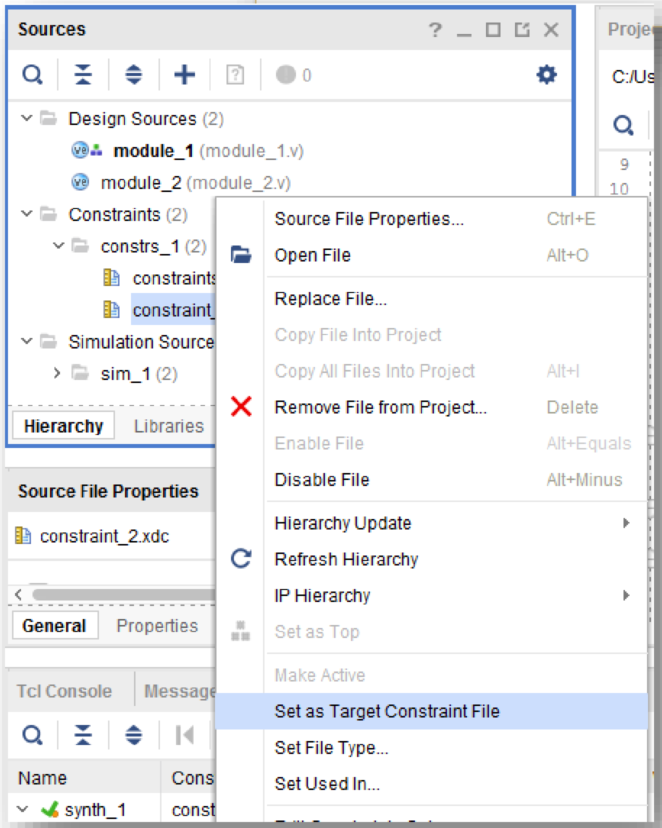
3. The synthesis process might take up to a few seconds

4. After the synthesis is complete with no error, a message will appear "Synthesis successfully completed". In this message click Cancel

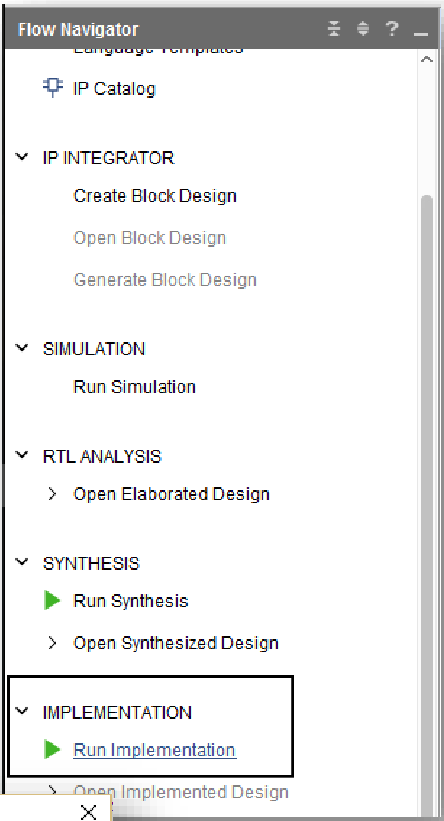

Synthesis Completed

Synthesis successfully completed.

5. If you have more than one constraint file, you need to choose one of them
   a. In the **Sources** window, right click on the constraint file that corresponds to your top module
   b. Choose as "Set as Target Constraint File"

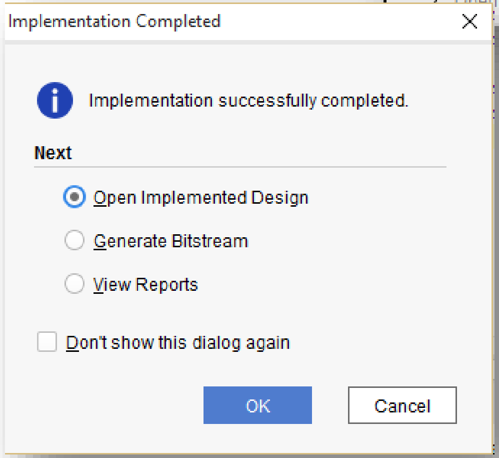


6. Implement your design
    a. In the **Flow Navigator**, under“IMPLEMENTATION”, click on “Run Implementation”



7. Connect the FPGA to the computer and switch it on using its power switch

8. After the implementation is complete with no error, a message will appear “Implementation successfully completed”. In this message, click Cancel


Implementation Completed

Implementation successfully completed.

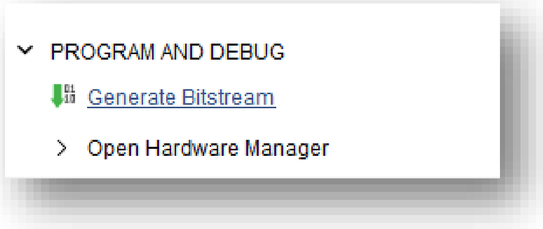
Next

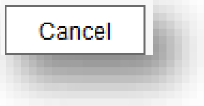
[✓ Open Implemented Design]
Generate Bitstream
View Reports

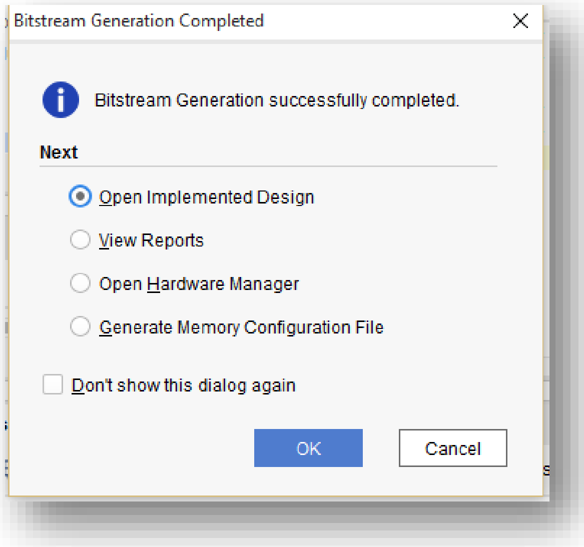
Don't show this dialog again

OK Cancel

9. In the **Flow Navigator**, under PROGRAM AND DEBUG, click Generate Bitstream.



10. After you get " Bitstream Generation successfully completed", click 


Bitstream Generation Completed

Bitstream Generation successfully completed.

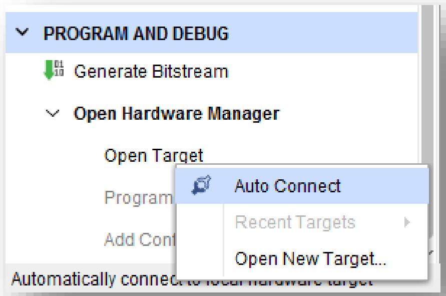
Next

[✓ Open Implemented Design]
View Reports
Open Hardware Manager
Generate Memory Configuration File

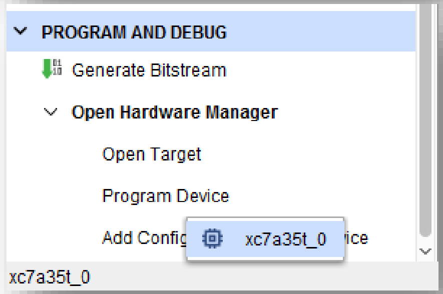
Don't show this dialog again

11.Under PROGRAM AND DEBUG, click **Open Hardware Manager**

a. Click on Open Target, then Auto Connect



b. Click on Program Device, then xc7a35t_0



Notes

- If you change your Verilog Code, you need to synthesize your design, implement it and generate the bitstream
- If you change your Constraint file, you need to implement your design and generate the bitstream

Synthesize: means to convert your Verilog code to the optimum digital circuits

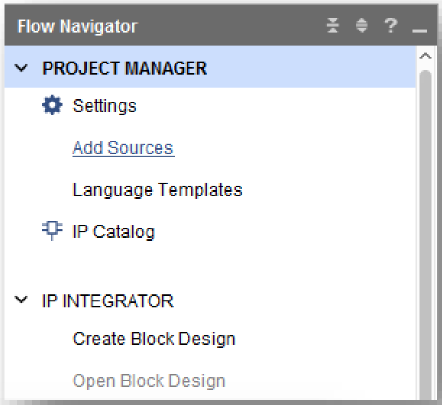
Implement: means to map the synthesized circuit to the FPGA

Generate Bitstream: coverts your implementation to 1’s and 0’s

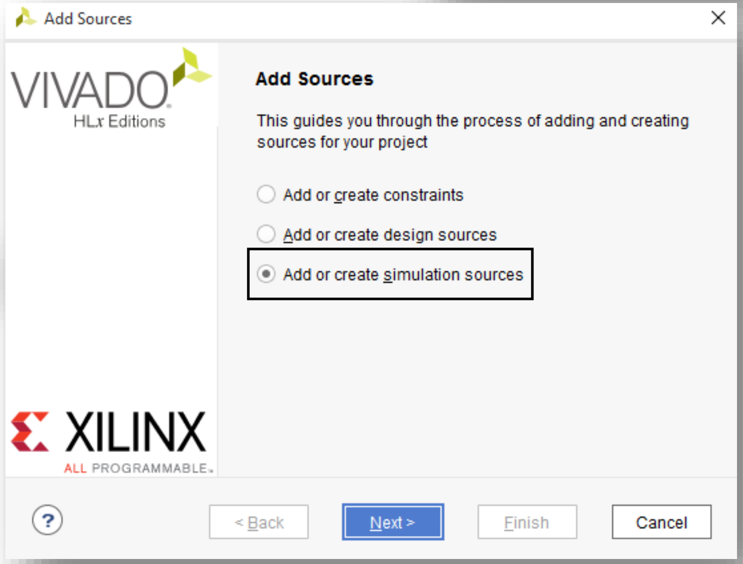
Run Simulation

- Simulation is used to verify your Verilog module and that it is functioning the way you want before you implement it on the FPGA

1. In the **Flow Navigator**, under **PROJECT MANAGER**, click Add Sources



2. In the Add Sources window, choose "Add or Create Simulation Sources" and then click Next >


Add Sources

Add Sources

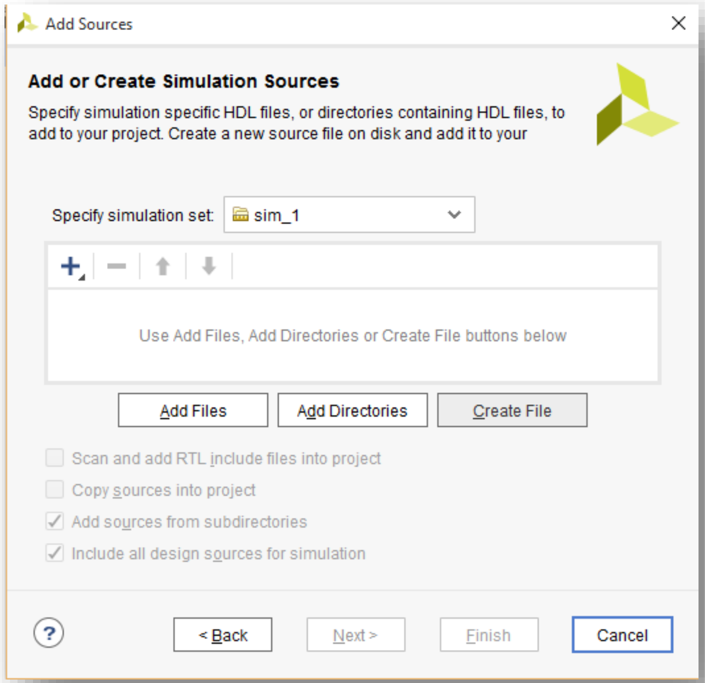
This guides you through the process of adding and creating sources for your project

Add or create constraints

Add or create design sources

[✓ Add or create simulation sources]

3. In the **Add or Create Simulation Sources** Window, click Create File


Add Sources

Add or Create Simulation Sources

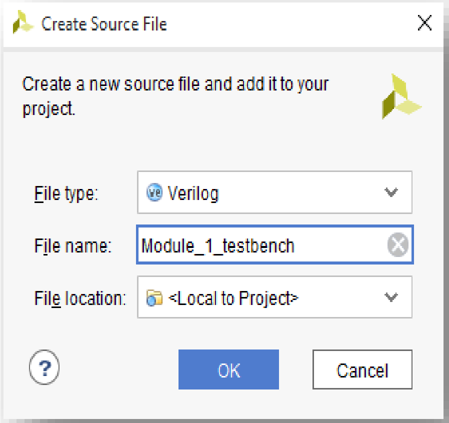
Specify simulation specific HDL files, or directories containing HDL files, to add to your project. Create a new source file on disk and add it to your

Specify simulation set: sim_1

Use Add Files, Add Directories or Create File buttons below

- [ ] Scan and add RTL include files into project
- [ ] Copy sources into project
- [x] Add sources from subdirectories
- [x] Include all design sources for simulation

4. Adjust the properties of the **Create Constraints File** window as follows:
   - **File type**: Verilog
   - **File name**: <your_file_name> must not have any spaces
   - **File location**: Local to Project

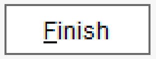
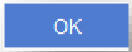

Create Source File

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

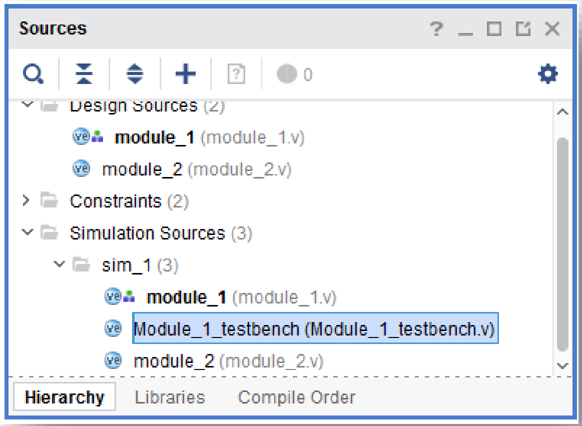
File type: Verilog

File name: Module_1_testbench

File location: <Local to Project>

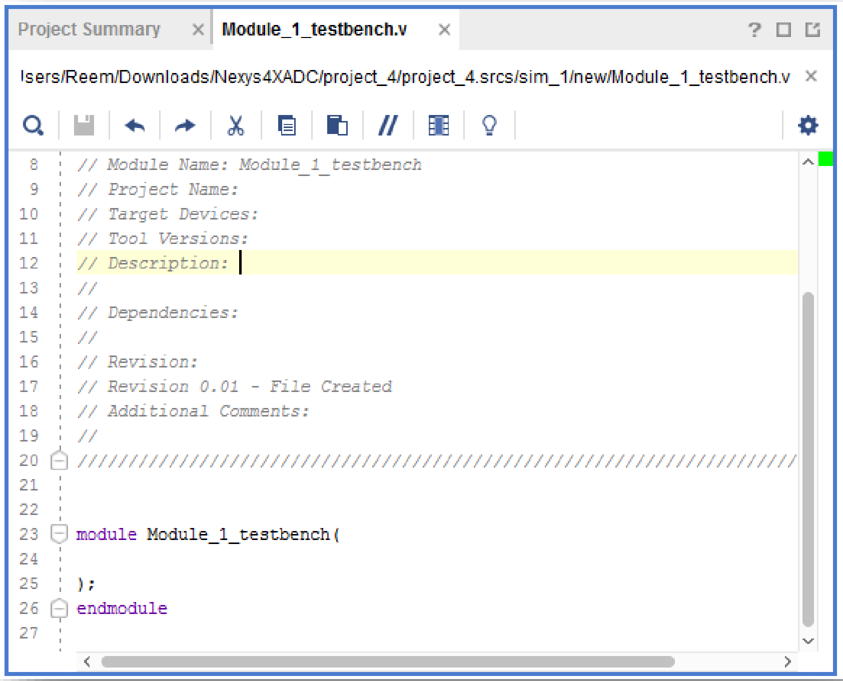
5. Click , then  and

6. In the **Sources** Window, under Simulation Sources, you will find the file you have just created



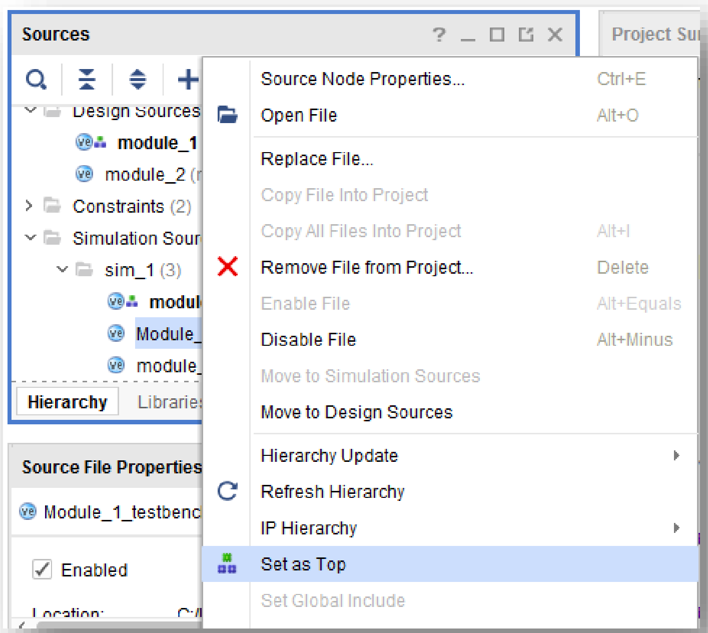
7. Double click on the file that you have created, a window with your file name and extension .v will appear where you can write your testbench.

** to know how to write a testbench → click here

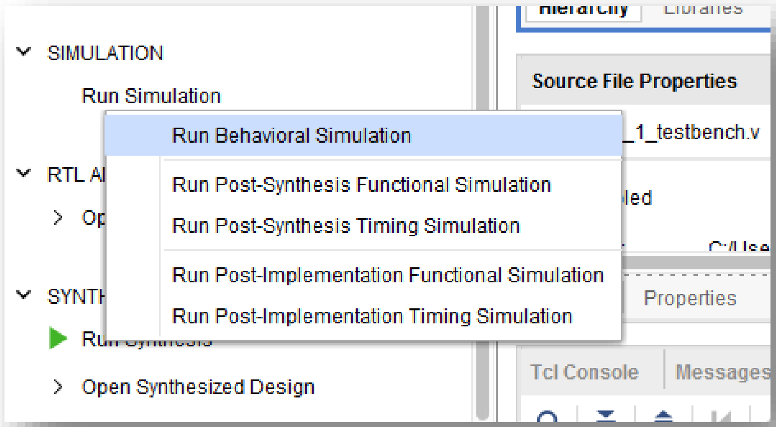

Project Summary
Module_1_testbench.v
!sers/Reem/Downloads/Nexys4XADC/project_4/project_4.srcs/sim_1/new/Module_1_testbench.v

```
// Module Name: Module_1_testbench
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////


module Module_1_testbench(

);
endmodule
```

8. After you write a testbench, right click on the simulation file in the **Sources** window. Click “Set as Top”



9. In the Flow Navigator, under SIMULATION, click Run Simulation then click Run Behavioural Simulation


SIMULATION
Run Simulation
Run Behavioral Simulation
Run Post-Synthesis Functional Simulation
Run Post-Synthesis Timing Simulation
Run Post-Implementation Functional Simulation
Run Post-Implementation Timing Simulation
RTL ANALYSIS
SYNTHESIS
Run Synthesis
Open Synthesized Design
Source File Properties
_1_testbench.v
Properties
Tcl Console
Messages

Block Memory Generator

Vivado provides a catalog for ready-made modules that you can use directly (similar to libraries in C). These are called IPs. The complixity of the IPs can range from simple arithmetic operators such as adders, accumulators, and multipliers, to system-level building blocks such as filters and memories.

1. Create a Coefficient File

a. Open a Notepad (or any other text editor)

a. Specify the base with witch you are writing your instructions by typing

```
memory_initialization_radix = 16;  // Your instructions are in hexadecimal
                                   // Use 2 for binary or 8 for octal
```

b. Specify your instructions by typing

```
memory_initialization_vector = 80001,  // the 1st instruction in hexadecimal
00120,                                 // the 2nd intruction in hexadecimal
```

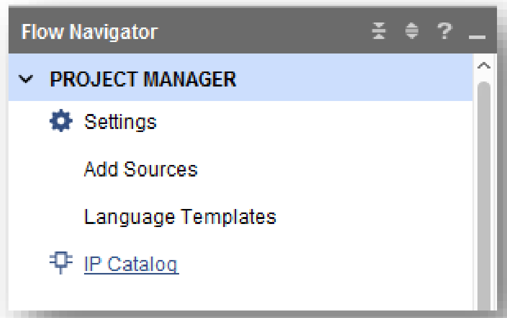
And so on …

c. Add a semicolon at the end of the last instruction

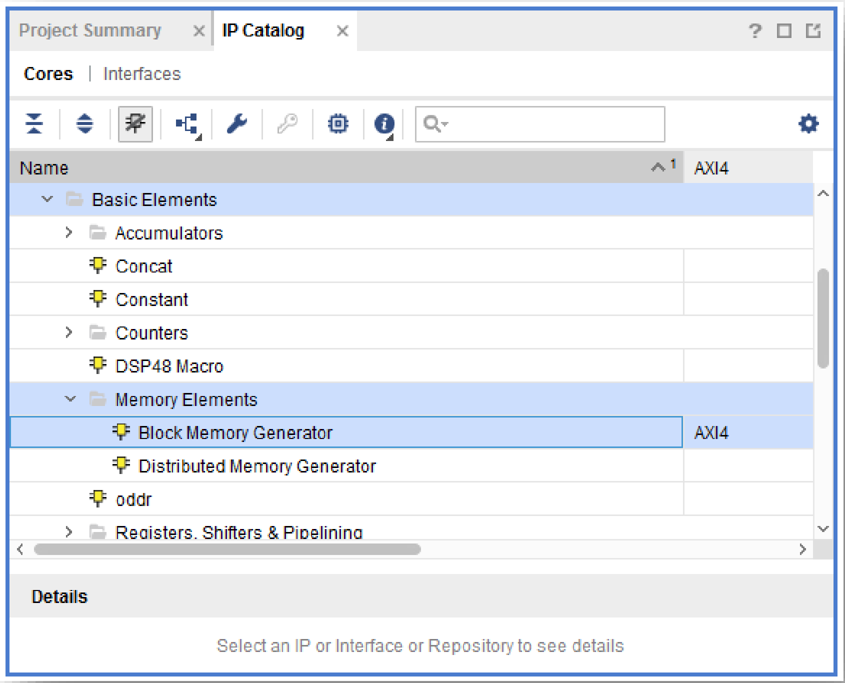
d. Save the file with extension .coe (make sure to remove .txt extension) and close it.

2. Generate a Core Memory

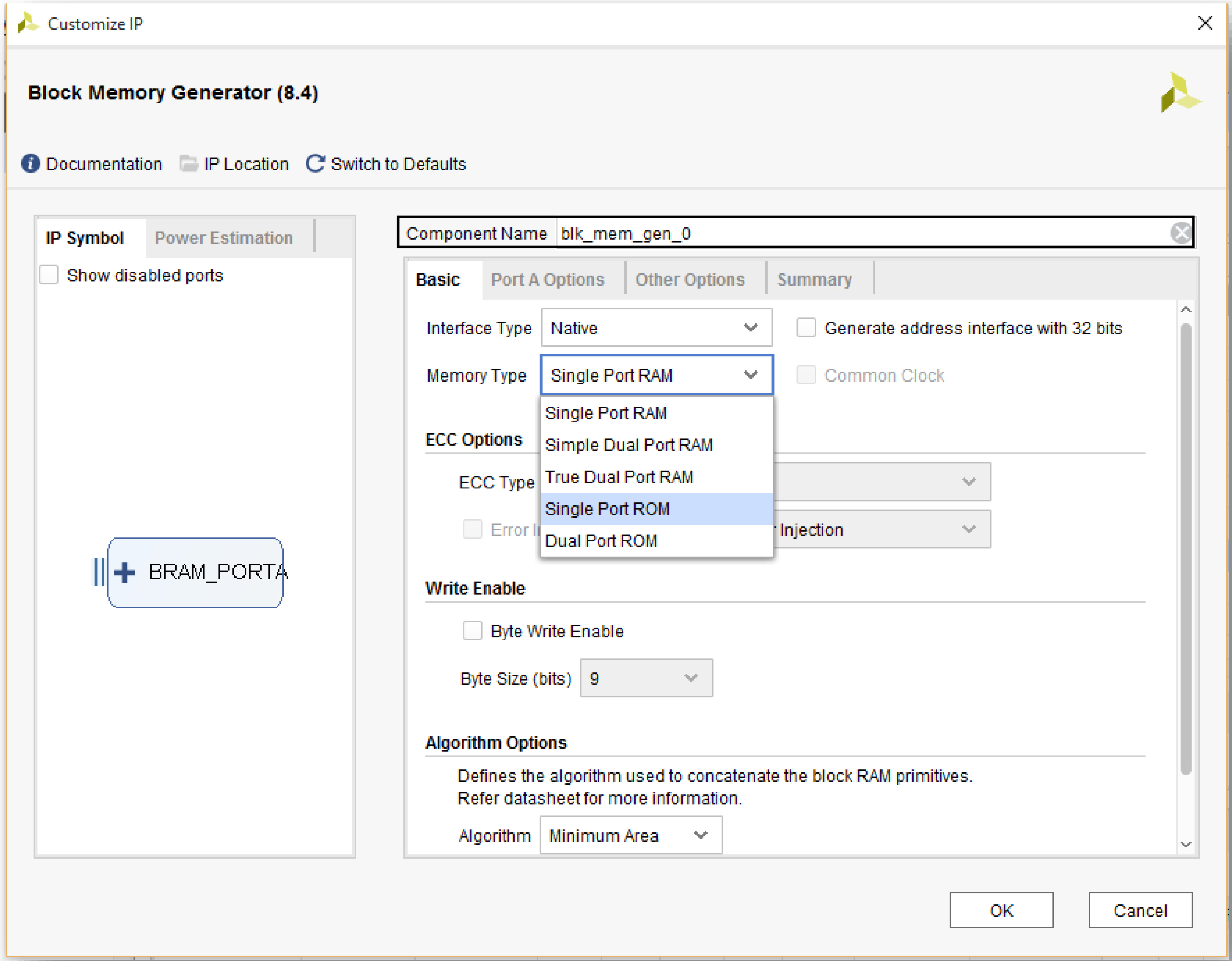
a. In the Flow Navigator, under PROJECT MANAGER, click IP Catalog



b. A window called IP Catalog will appear. Under Basic Elements, choose Memory Elements. Then, double click on Block Memory Generator



c. In the Customize IP tab, type the name of the memory module that will be generated for you. It must not have any spaces.


Customize IP

Block Memory Generator (8.4)

Documentation   IP Location   Switch to Defaults

IP Symbol   Power Estimation

Show disabled ports

Component Name blk_mem_gen_0

Basic   Port A Options   Other Options   Summary

Interface Type: Native

Generate address interface with 32 bits

Memory Type: Single Port RAM

Common Clock

ECC Options

ECC Type

Error I… Injection

Write Enable

Byte Write Enable

Byte Size (bits): 9

Algorithm Options

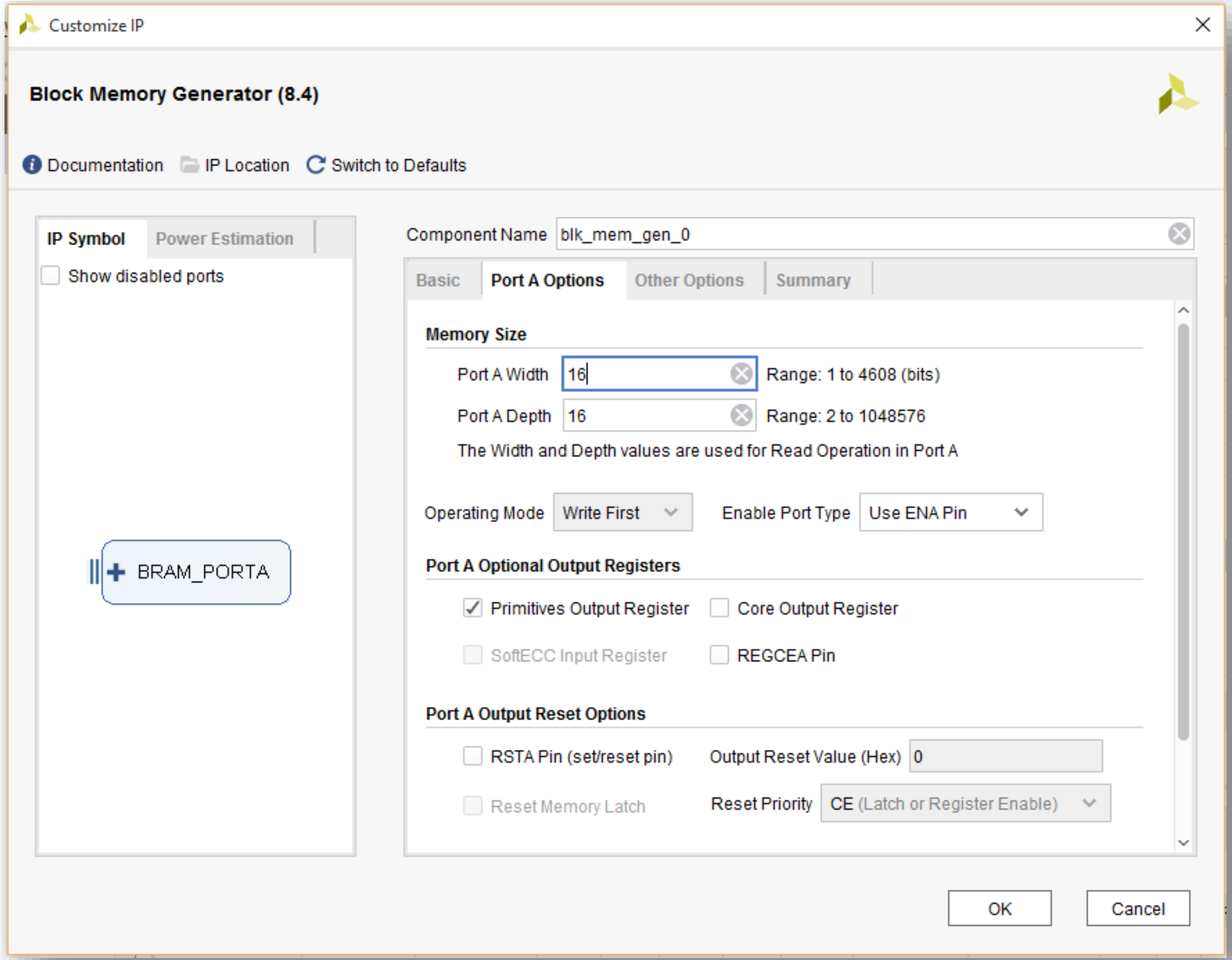
Defines the algorithm used to concatenate the block RAM primitives.
Refer datasheet for more information.

Algorithm: Minimum Area

OK   Cancel

d. In the **Basic** tab, choose a Memory Type: Single Port ROM.

e. In the Port A options tab, specify
  - Port A Width: your memory word length
  - Port A Depth: the size of your memory (number of memory words)


Customize IP

Block Memory Generator (8.4)

Documentation   IP Location   Switch to Defaults

IP Symbol   Power Estimation

Show disabled ports

BRAM_PORTA

Component Name: blk_mem_gen_0

Basic   Port A Options   Other Options   Summary

Memory Size

Port A Width: 16   Range: 1 to 4608 (bits)

Port A Depth: 16   Range: 2 to 1048576

The Width and Depth values are used for Read Operation in Port A

Operating Mode: Write First   Enable Port Type: Use ENA Pin

Port A Optional Output Registers

[✓] Primitives Output Register   [ ] Core Output Register

[ ] SoftECC Input Register   [ ] REGCEA Pin

Port A Output Reset Options

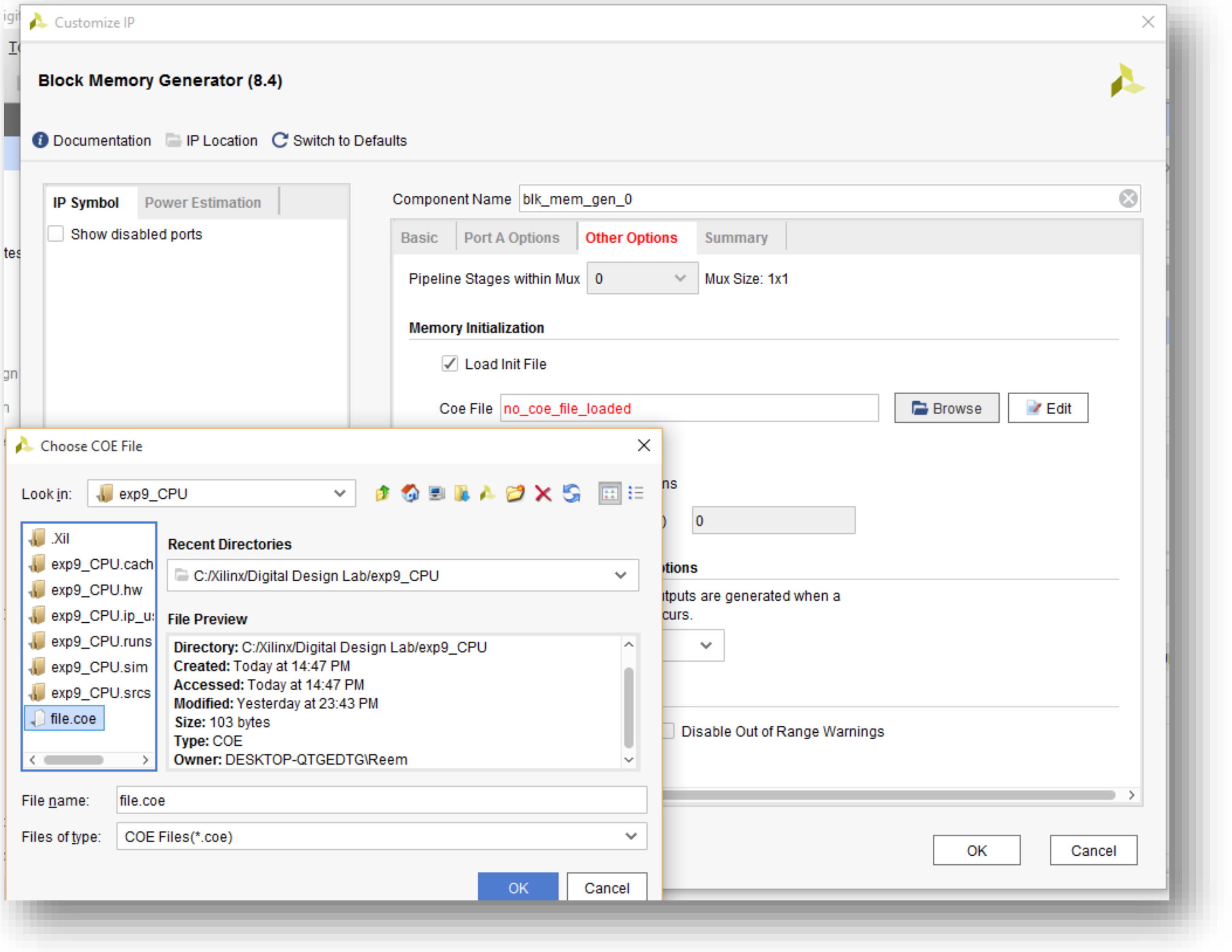
[ ] RSTA Pin (set/reset pin)   Output Reset Value (Hex): 0

[ ] Reset Memory Latch   Reset Priority: CE (Latch or Register Enable)

OK   Cancel

f. In the **other Options** tab, check the box load Init File, browse your computer where you saved the .coe file and then click OK and OK

g. Now Vivado will generate a block Rom module. The Rom is storing the instructions inside the .coe file.



Component Name: blk_mem_gen_0

Pipeline Stages within Mux: 0 Mux Size: 1x1

Memory Initialization

✓ Load Init File

Coe File: no_coe_file_loaded

Look in: exp9_CPU

.Xil
exp9_CPU.cach
exp9_CPU.hw
exp9_CPU.ip_u
exp9_CPU.runs
exp9_CPU.sim
exp9_CPU.srcs
file.coe

Recent Directories: C:/Xilinx/Digital Design Lab/exp9_CPU

File Preview

Directory: C:/Xilinx/Digital Design Lab/exp9_CPU
Created: Today at 14:47 PM
Accessed: Today at 14:47 PM
Modified: Yesterday at 23:43 PM
Size: 103 bytes
Type: COE
Owner: DESKTOP-QTGEDTG\Reem

File name: file.coe

Files of type: COE Files(*.coe)

Disable Out of Range Warnings

h. To create an instance of the Rom, use the following syntax:

```
Mem m(.clka(clk),  1, .addra(pc), .douta(mem_word));
```

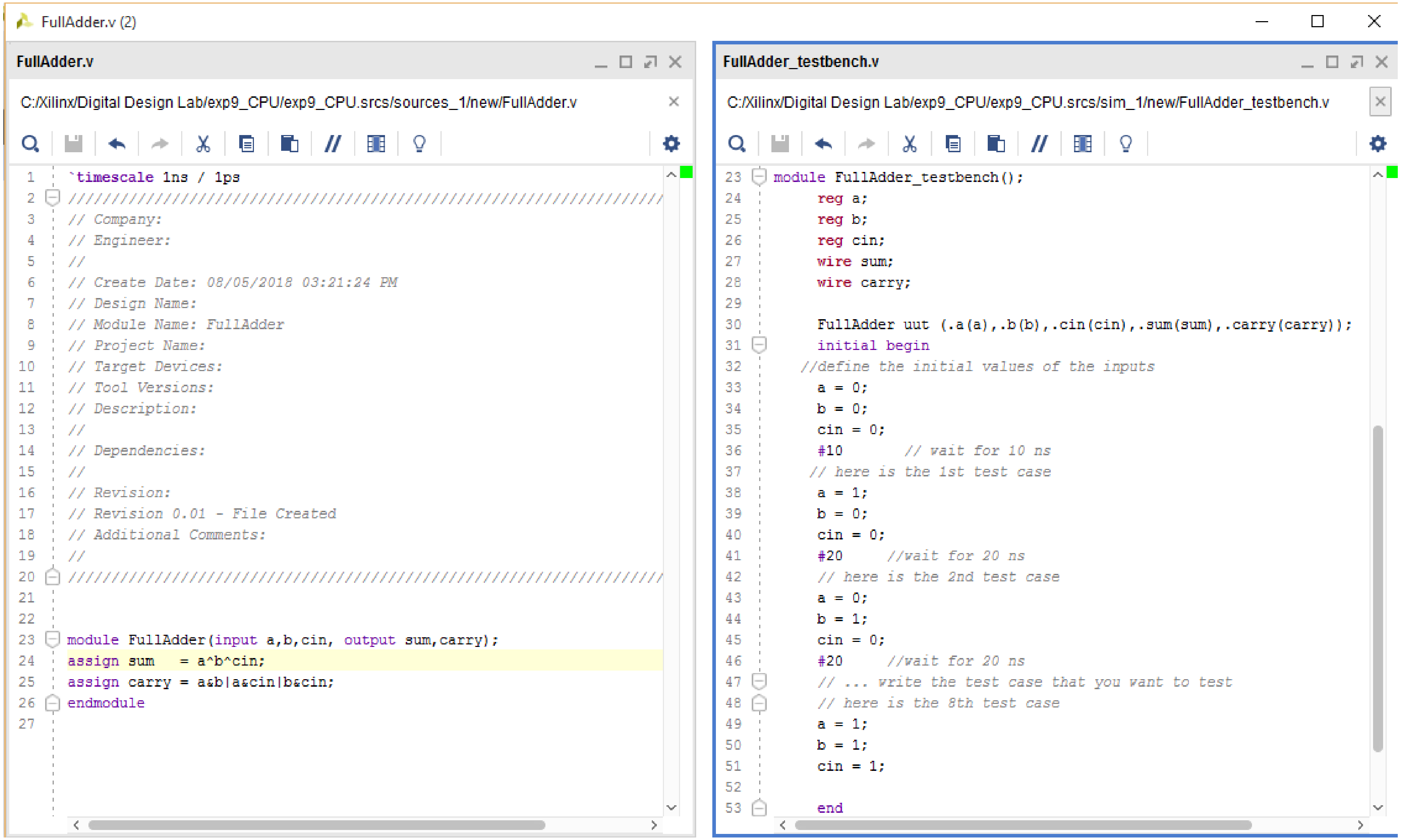
Where:
`Mem`: should be replaced with the module name (the one you chose in step c)
`m`: should be replaced with the instance name
`clk`: should be replaced with your input clk signal
`pc`: should be replaced with your program counter signal
`mem_word`: should be replaced with your memory word signal

Write a Testbench

A testbench is a module that is used to test another module which we will call the target module.
The test bench has the following specifications:

- It has no inputs or outputs
- It has the inputs of the target module defined as `reg` and the outputs of the target module defined as `wire`
- It has an instance of the target module
- It has an initial block where you define your test cases.

Code Example: A module with a testbench



FullAdder.v (2)

FullAdder.v

C:/Xilinx/Digital Design Lab/exp9_CPU/exp9_CPU.srcs/sources_1/new/FullAdder.v

```verilog
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 08/05/2018 03:21:24 PM
// Design Name:
// Module Name: FullAdder
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////


module FullAdder(input a,b,cin, output sum,carry);
assign sum   = a^b^cin;
assign carry = a&b|a&cin|b&cin;
endmodule

```

FullAdder_testbench.v

C:/Xilinx/Digital Design Lab/exp9_CPU/exp9_CPU.srcs/sim_1/new/FullAdder_testbench.v

```verilog
module FullAdder_testbench();
     reg a;
     reg b;
     reg cin;
     wire sum;
     wire carry;

     FullAdder uut (.a(a),.b(b),.cin(cin),.sum(sum),.carry(carry));
     initial begin
   //define the initial values of the inputs
     a = 0;
     b = 0;
     cin = 0;
     #10       // wait for 10 ns
    // here is the 1st test case
     a = 1;
     b = 0;
     cin = 0;
     #20      //wait for 20 ns
     // here is the 2nd test case
     a = 0;
     b = 1;
     cin = 0;
     #20      //wait for 20 ns
     // ... write the test case that you want to test
     // here is the 8th test case
     a = 1;
     b = 1;
     cin = 1;

     end
```