

Vivado Tutorial and Lab
ECE 366- Computer Organization

February 10, 2025

1. Tasks to be done in this lab.

1. Design and implement a half-adder (using a data flow approach).
2. Using the half-adders designed in step 1 to implement a full adder.
3. Implement a 4-bit adder/subtractor (adds/subtracts two 4-bit inputs) using the full adders designed in step 2.

2. Download

- a. Windows: [AMD Unified Installer Windows](#)
 - i. Double click on the downloaded exe file and follow on-screen installation instructions.
- b. Linux: [AMD Unified Installer Linux](#)
 - i. `cd <folder_in_which_you_downloaded_the_bin_file>`
 - ii. `chmod +x <bin_file>`
 - iii. `./<bin_file>`
 - iv. follow on-screen installation instructions.
- c. Detailed User Guide and Installation Link from AMD: <https://docs.amd.com/r/en-US/ug973-vivado-release-notes-install-license/Navigating-Content-by-Design-Process>

3. Step-by-step guide for implementing half-adder.

Follow the steps shown in the figures one by one.

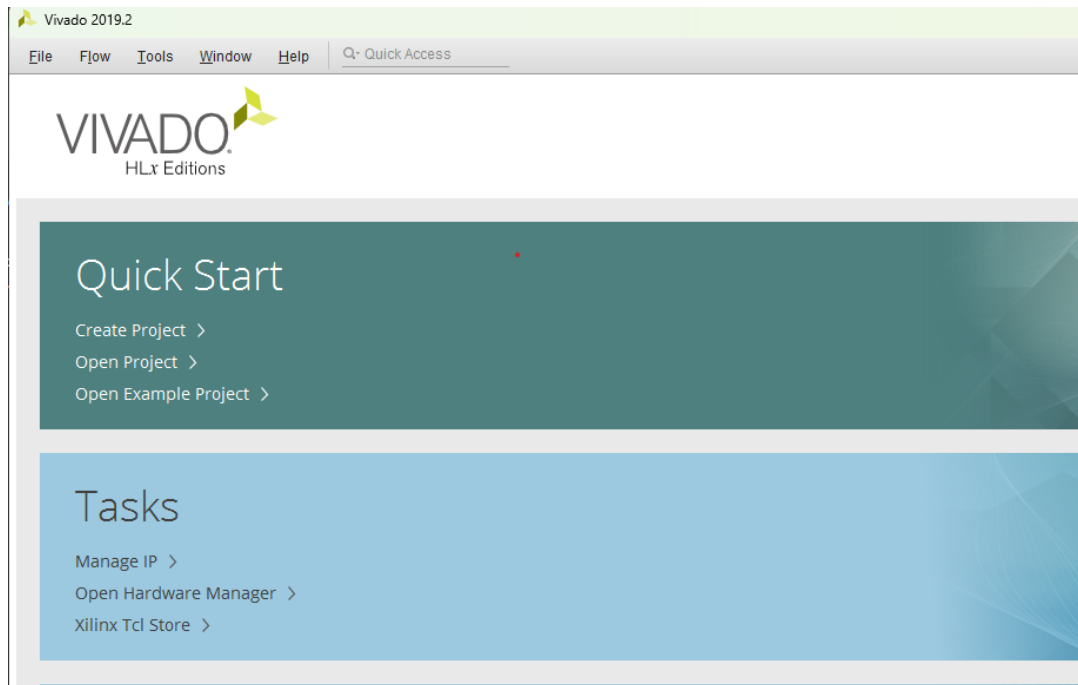


Figure 1: Create New Project

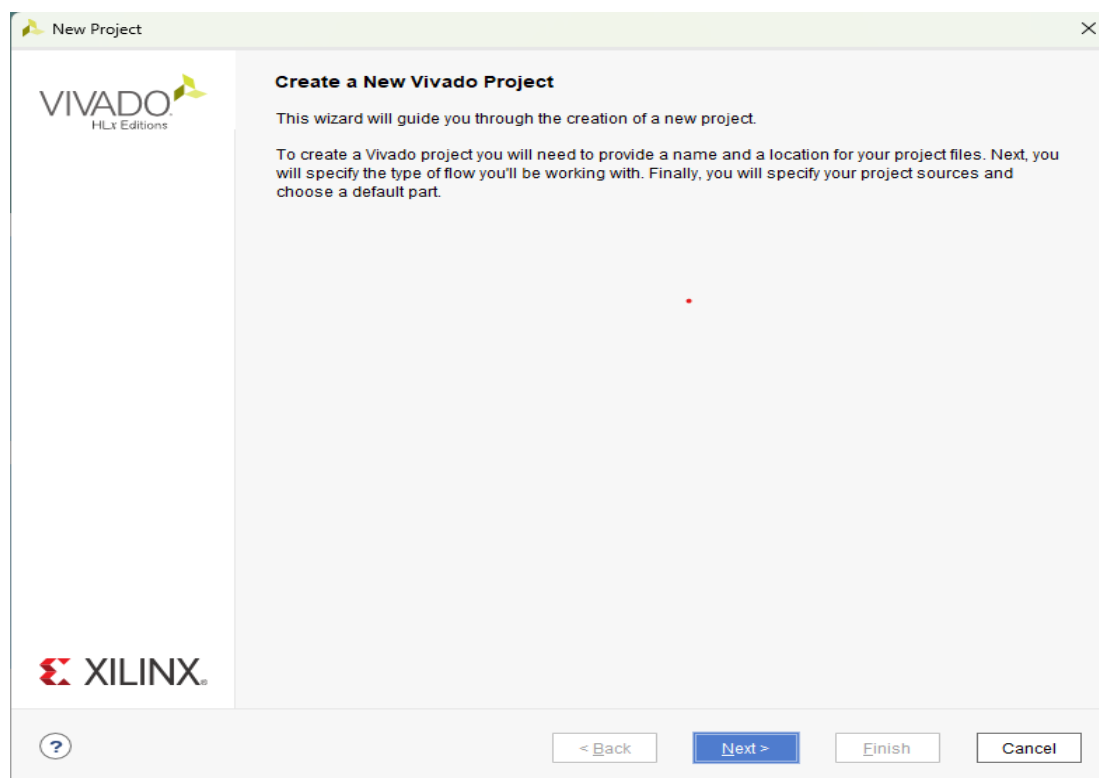


Figure 2: Click Next

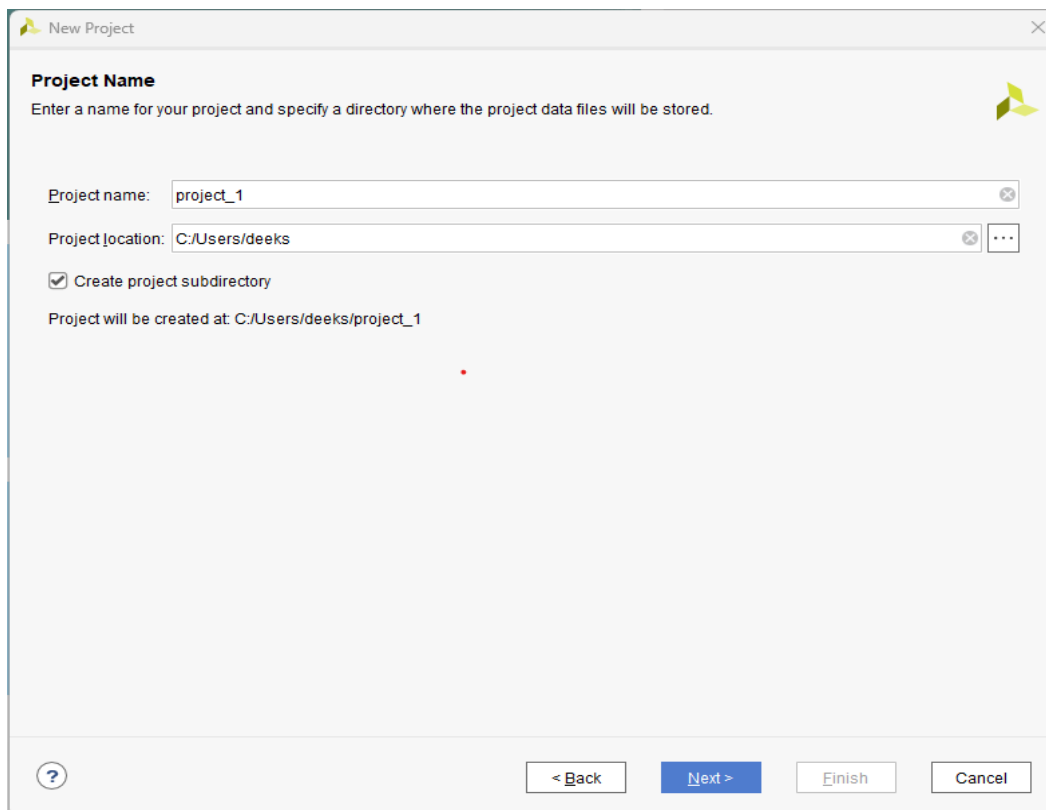


Figure 3: Give your project a name and set the location of your project

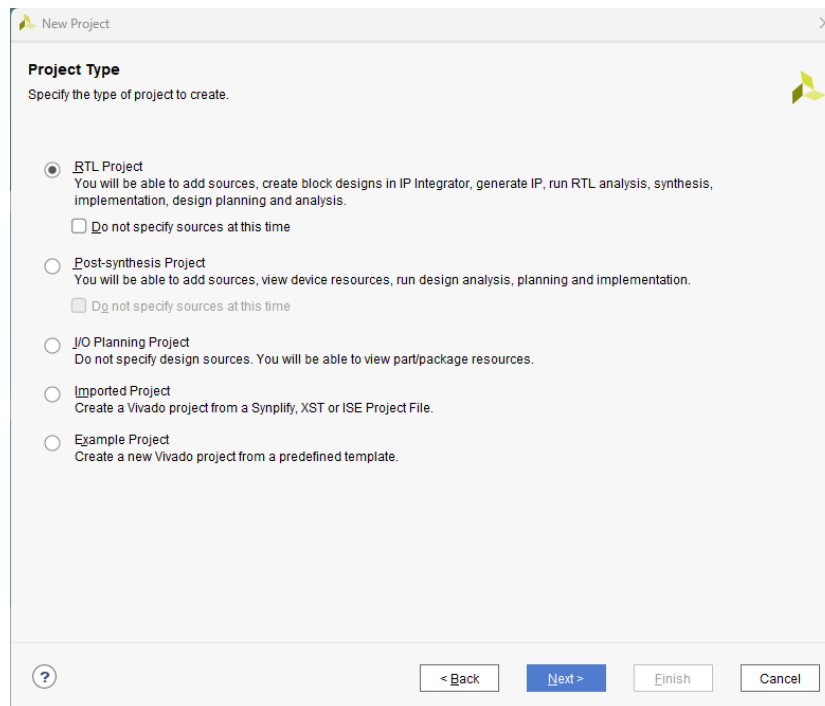


Figure 4: Select the type of project you wish to create (RTL Project in our case, uncheck the option "Do not specify.....")

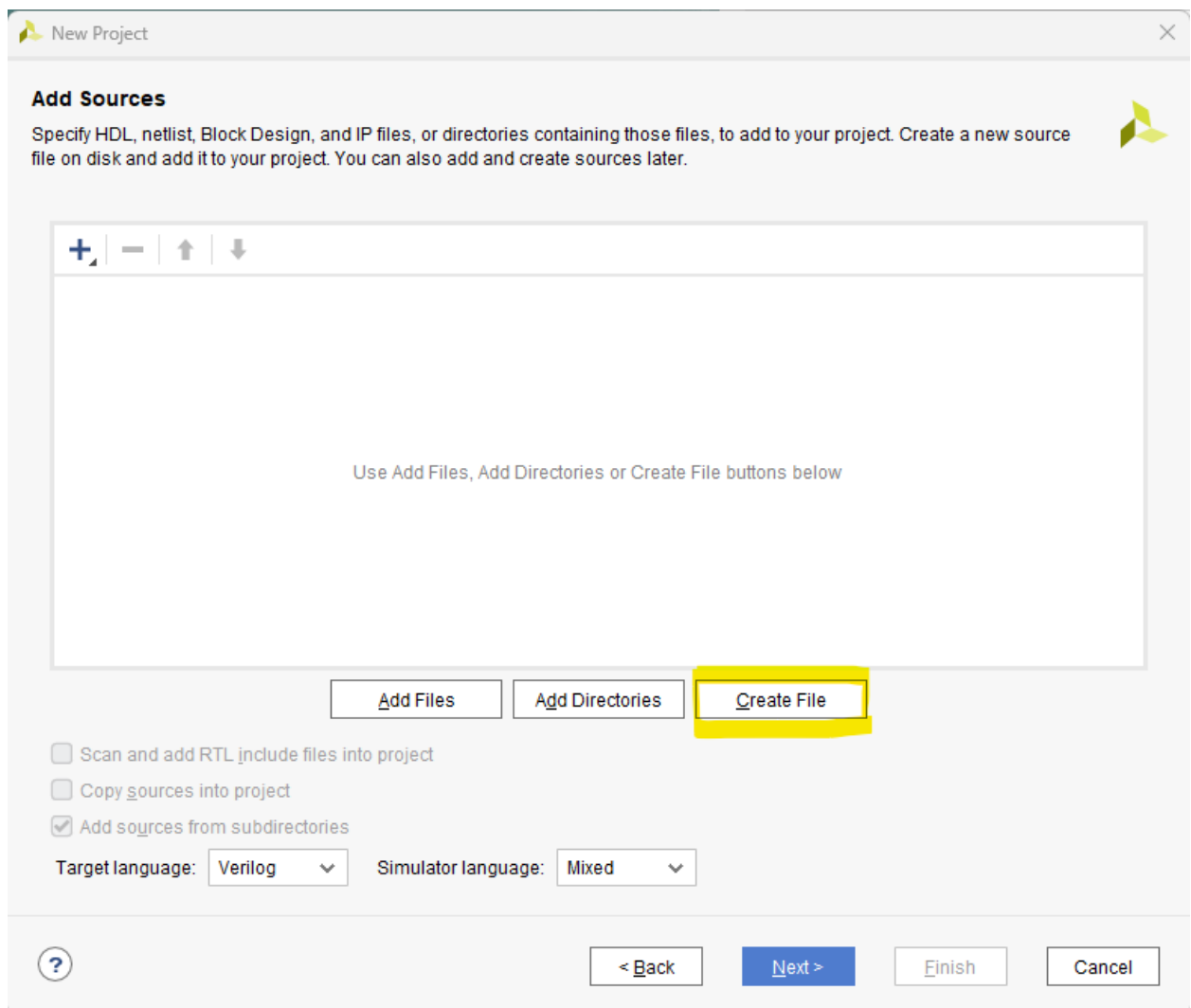


Figure 5: In Add Sources, click on Create File to create new files

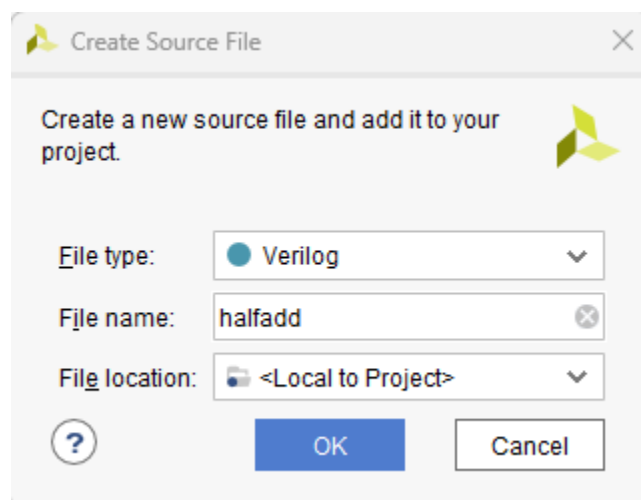


Figure 6: Give a suitable name to your verilog file and click OK.

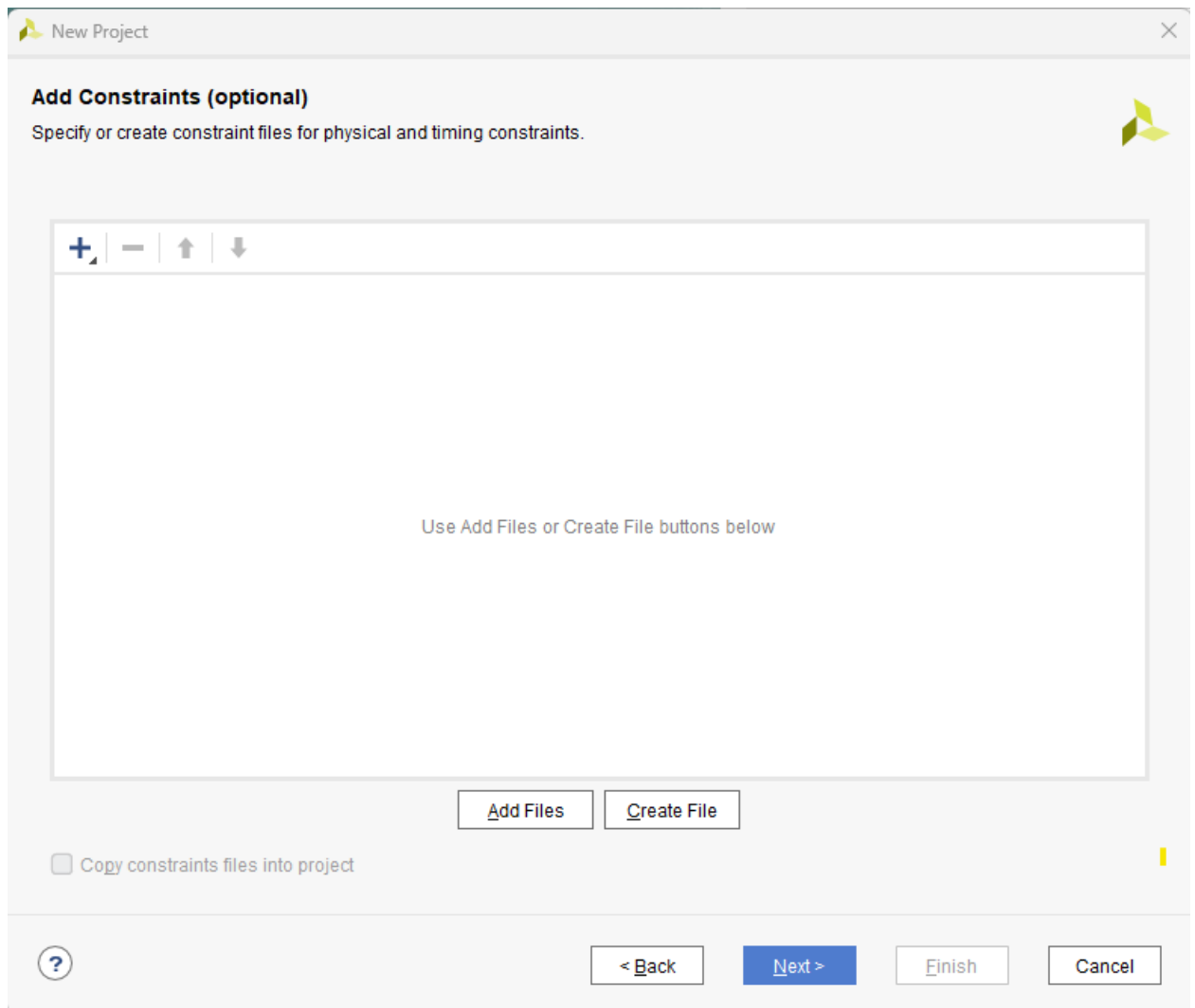


Figure 7: Add Constraints if required (click Next for our case)

These settings remain the same for any ECE 366 project.

New Project

Default Part
Choose a default Xilinx part or board for your project.

Parts | Boards

[Reset All Filters](#)

Category: All Package: cpg236 Temperature: All Remaining
Family: Artix-7 Speed: -1 Static power: All Remaining

Search: Q-

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	Gt
xc7a15tcpg236-1	236	106	10400	20800	25	0	45	2
xc7a35tcpg236-1	236	106	20800	41600	50	0	90	2
xc7a50tcpg236-1	236	106	32600	65200	75	0	120	2

< Back Next > Finish Cancel

Figure 8: Select the target FPGA device

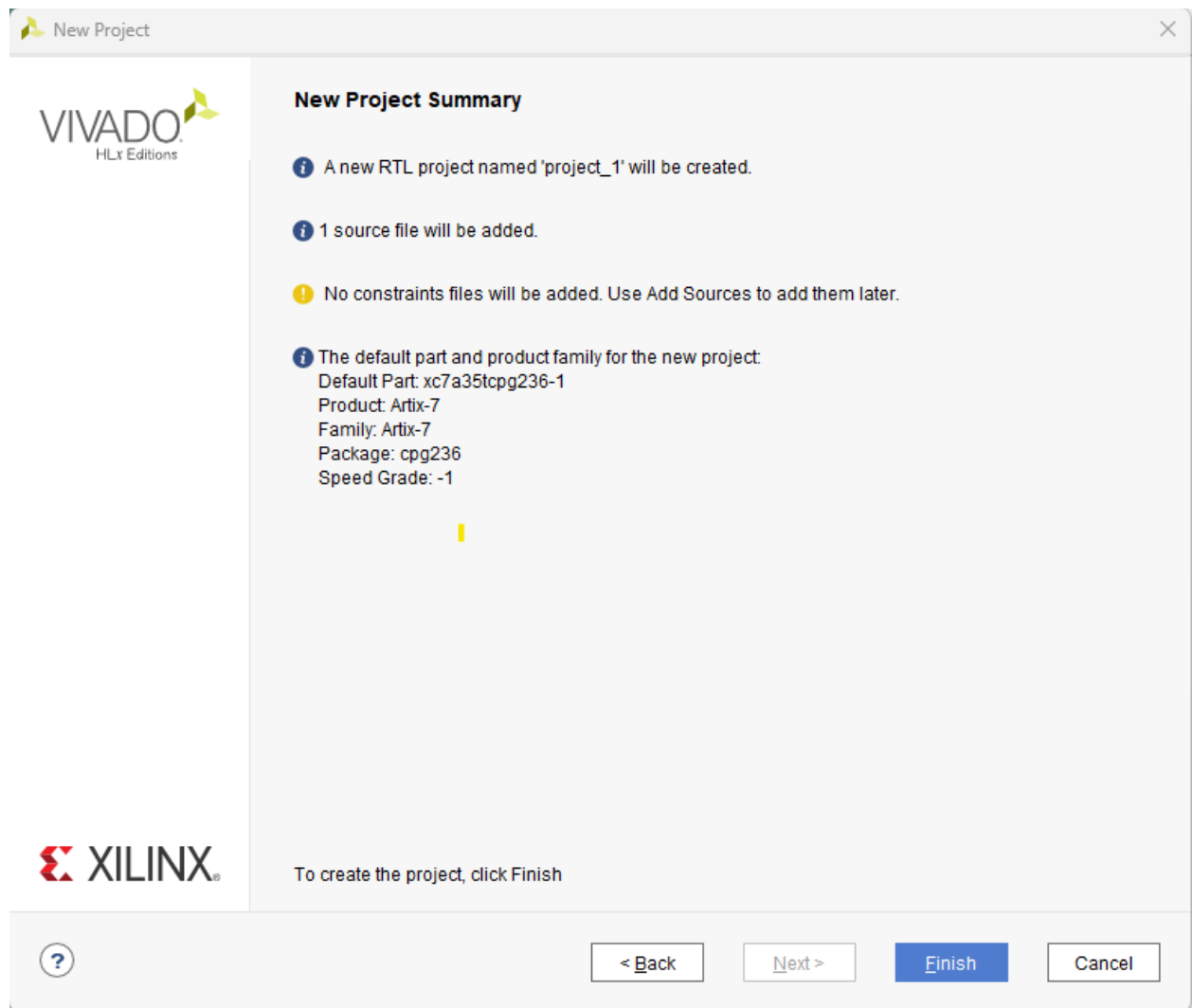


Figure 9: Check the project summary, click Finish.

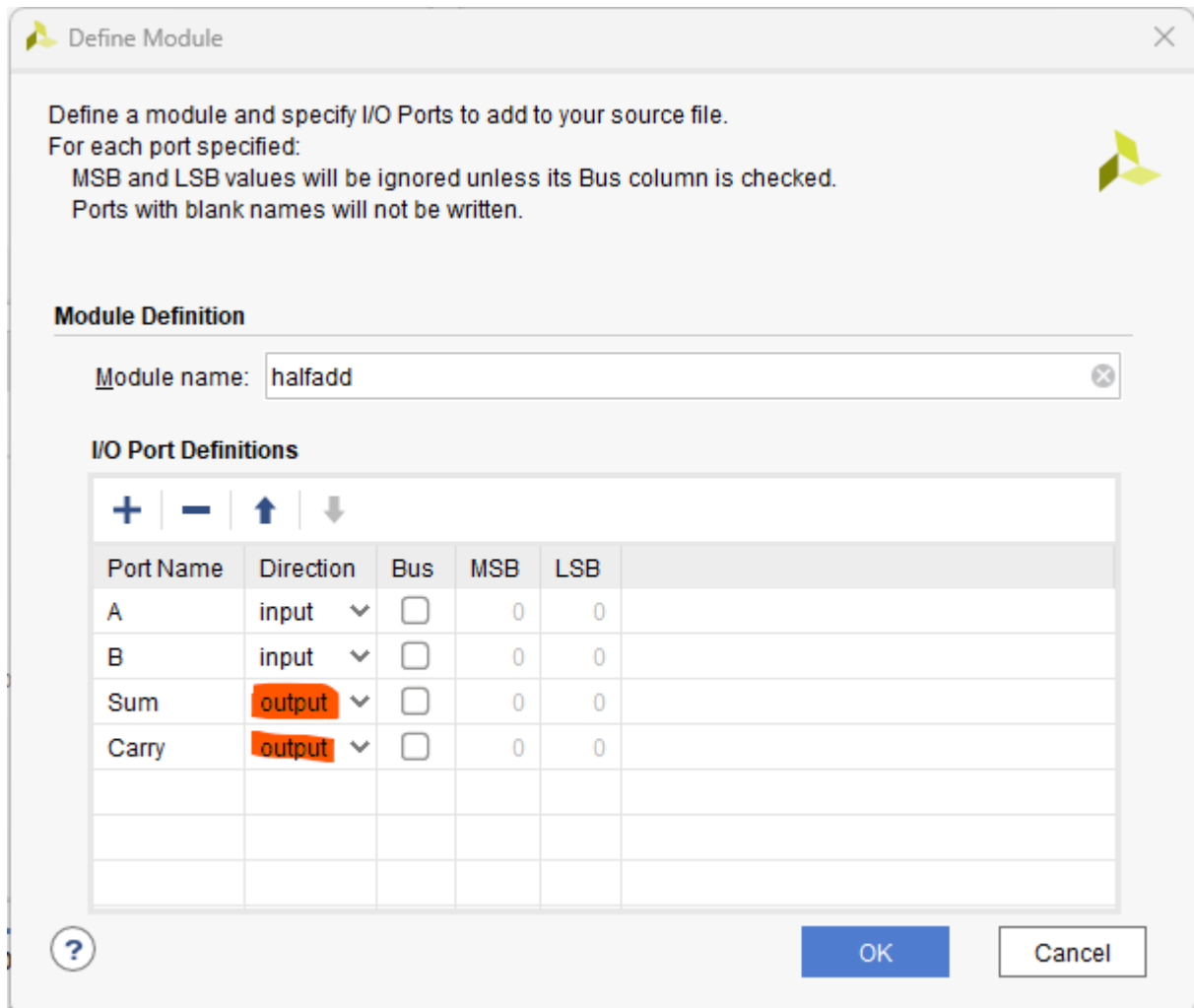
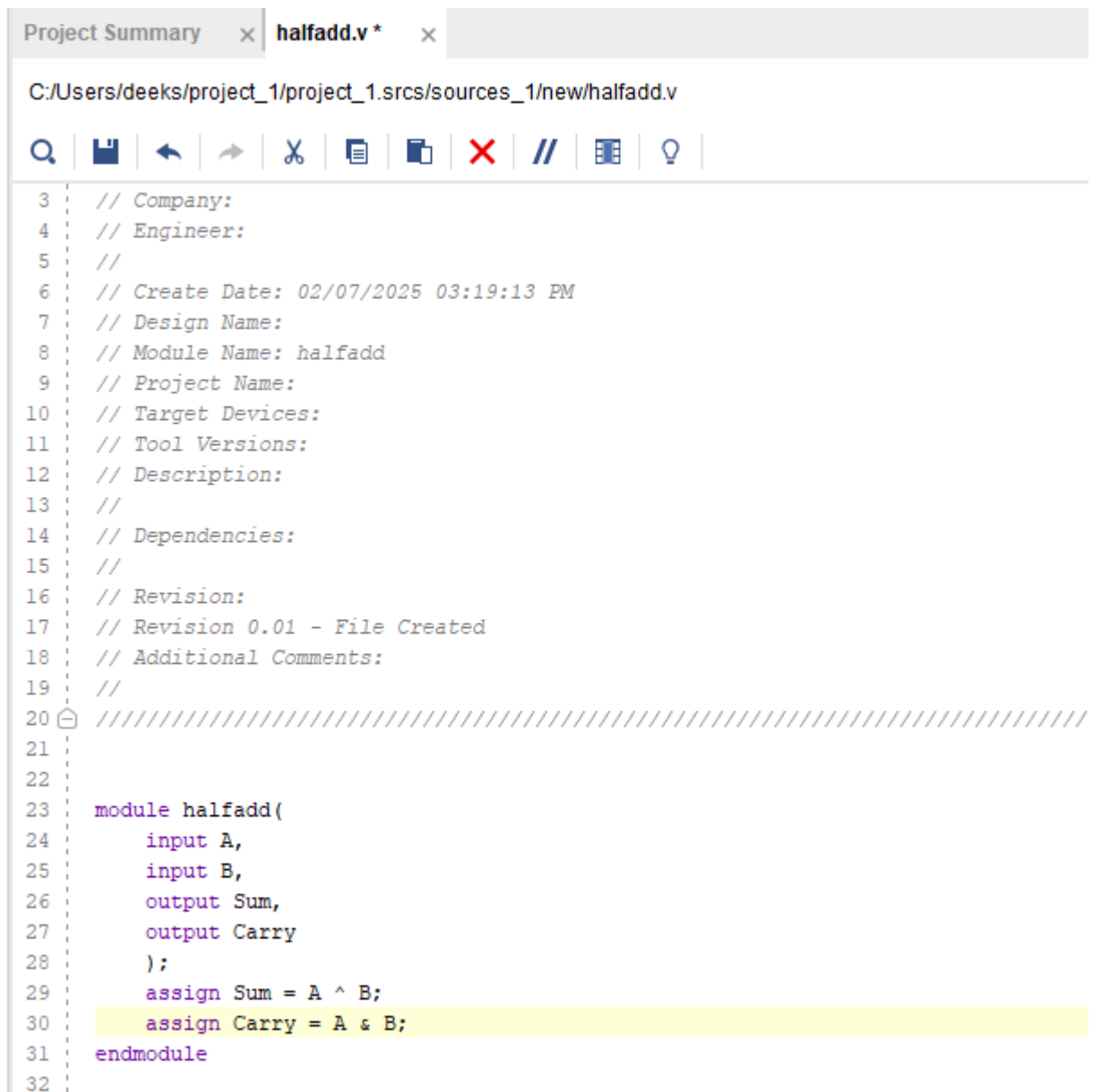


Figure 10: Mention the IOs of the verilog files created



Figure 11: You can see the files added in Sources window. Now open the files and code the required logic.



```
3 // Company:
4 // Engineer:
5 //
6 // Create Date: 02/07/2025 03:19:13 PM
7 // Design Name:
8 // Module Name: halfadd
9 // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 ///////////////////////////////////////////////////////////////////
21
22
23 module halfadd(
24     input A,
25     input B,
26     output Sum,
27     output Carry
28 );
29     assign Sum = A ^ B;
30     assign Carry = A & B;
31 endmodule
32
```

Figure 12: Half adder code added in the halfadd.v file and save the file.

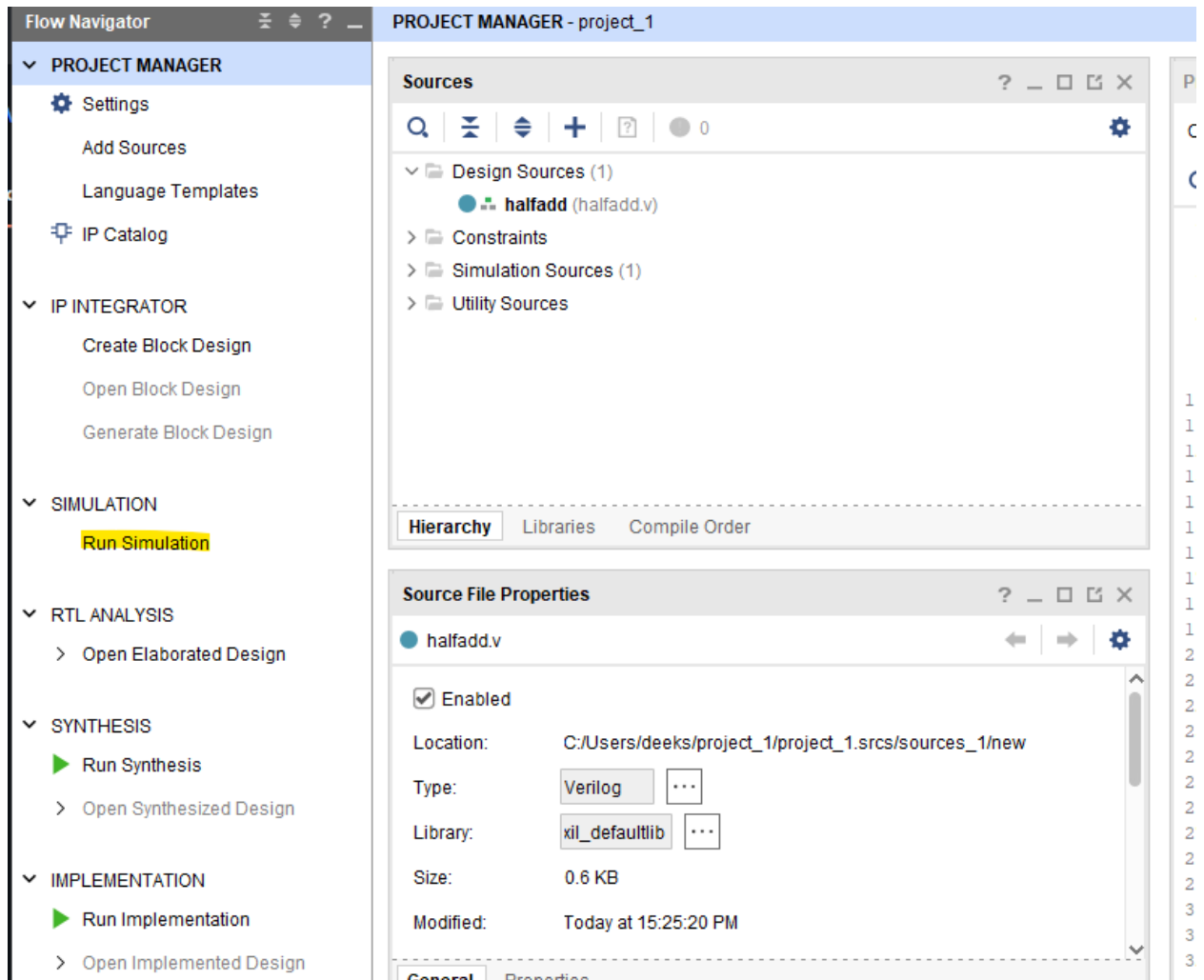


Figure 13: Click on Run Simulation and select Run Behavioral Simulation.

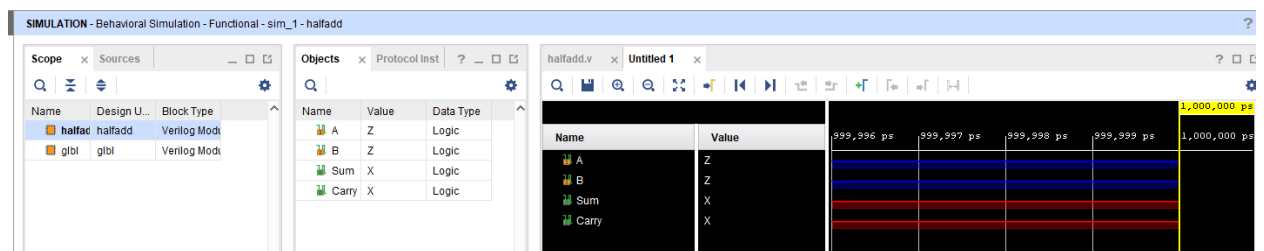


Figure 14: Behavioral Simulation will open with the following windows.

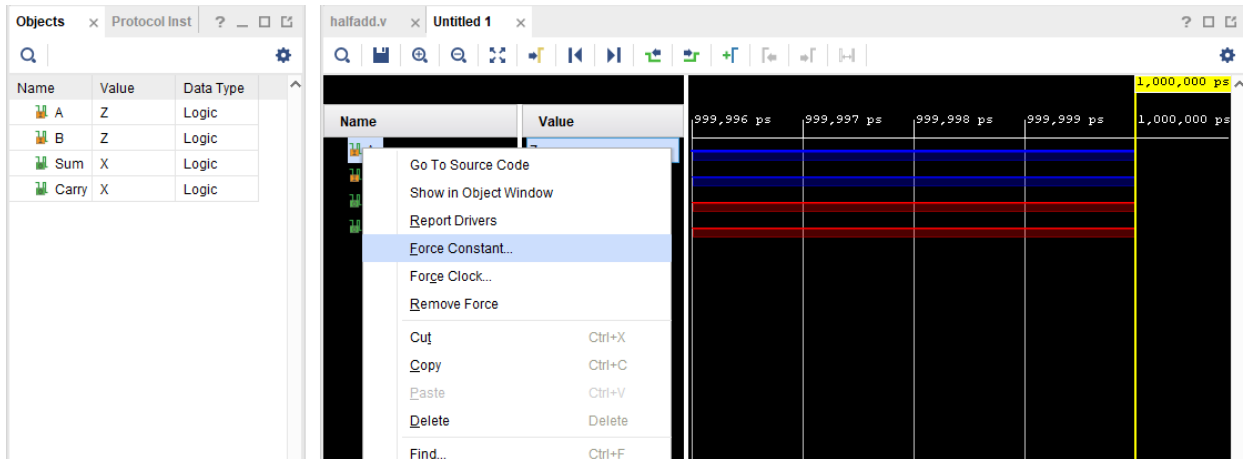


Figure 15: Force constant or clocks to the Inputs to generate various test sequences.

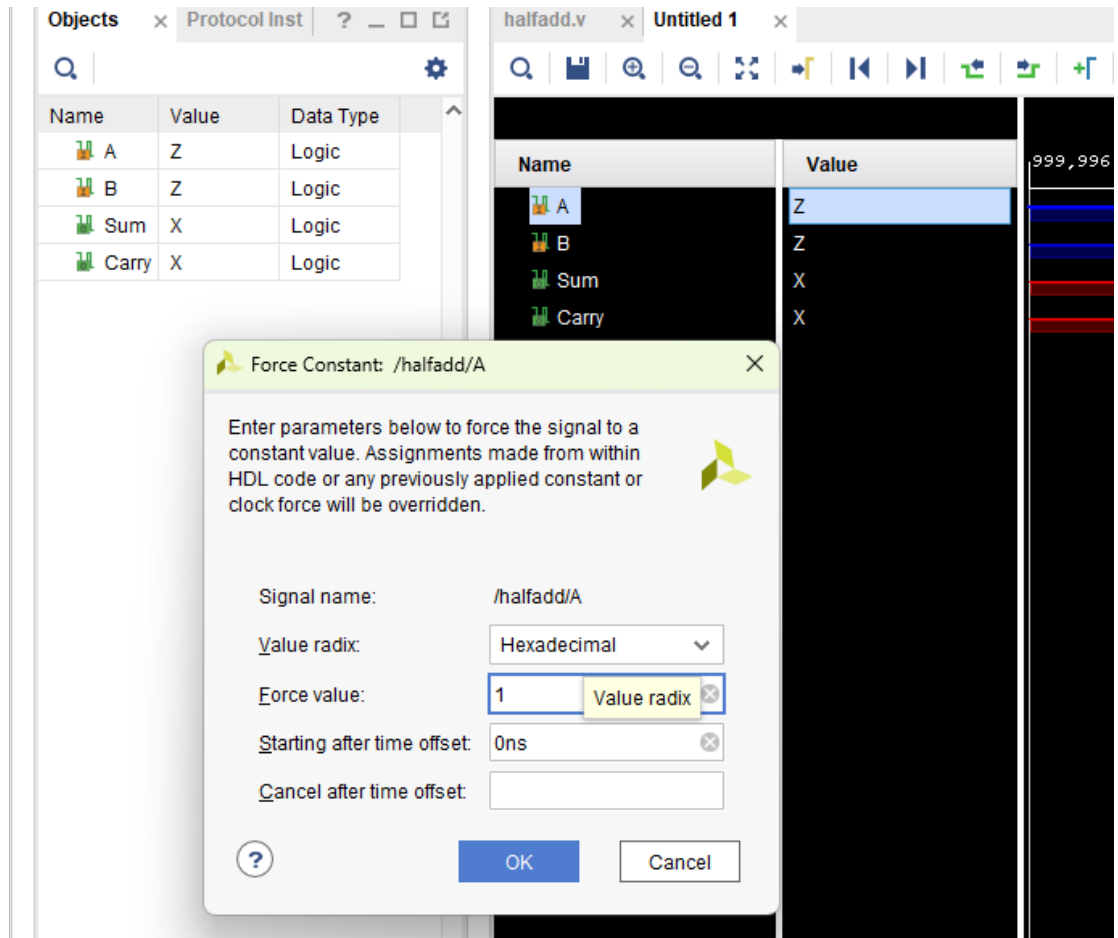


Figure 16: Give values in the Force value.

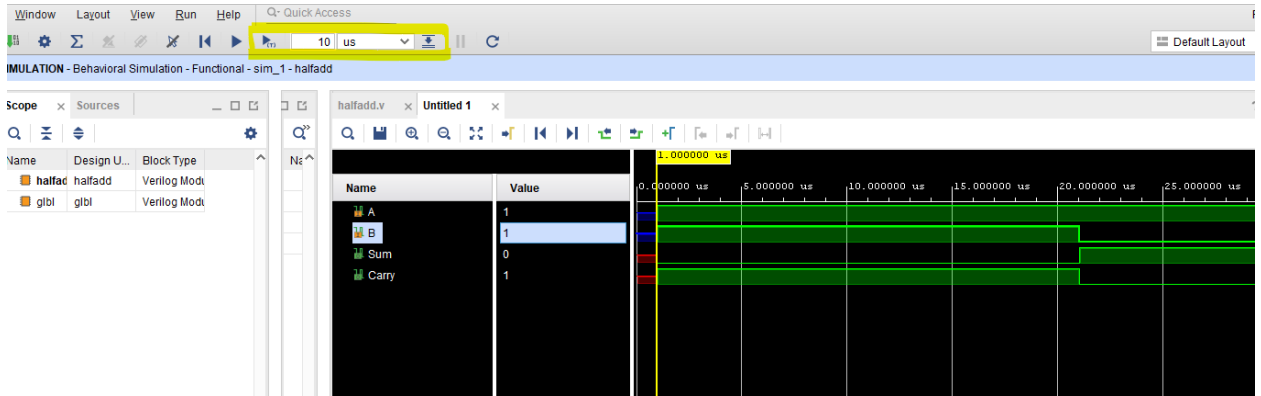


Figure 17: Verify the outputs generated for corresponding Inputs. Click on the Run option for the runs.

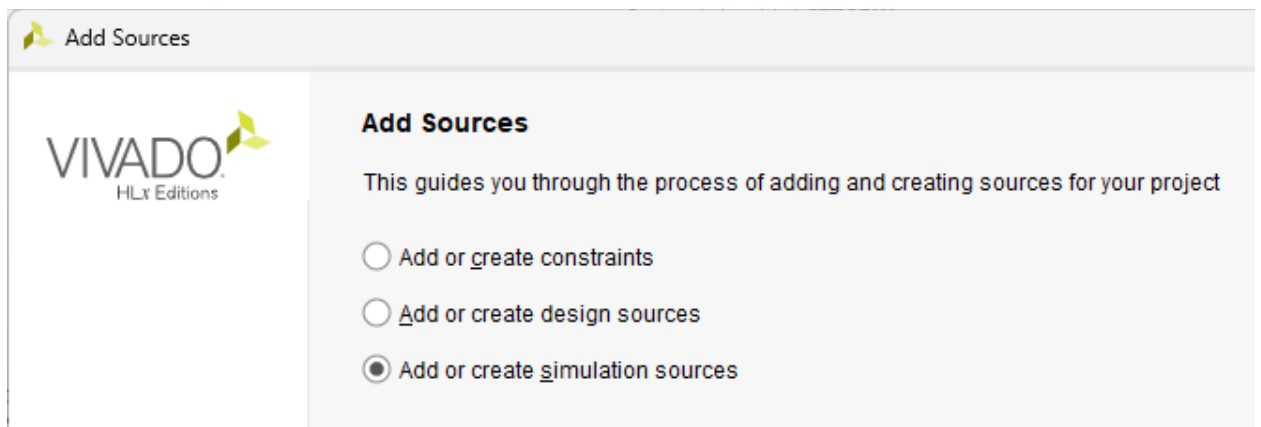


Figure 18: Another method is to run with the testbench file. Use the Add or create simulation sources option to add the testbench file.

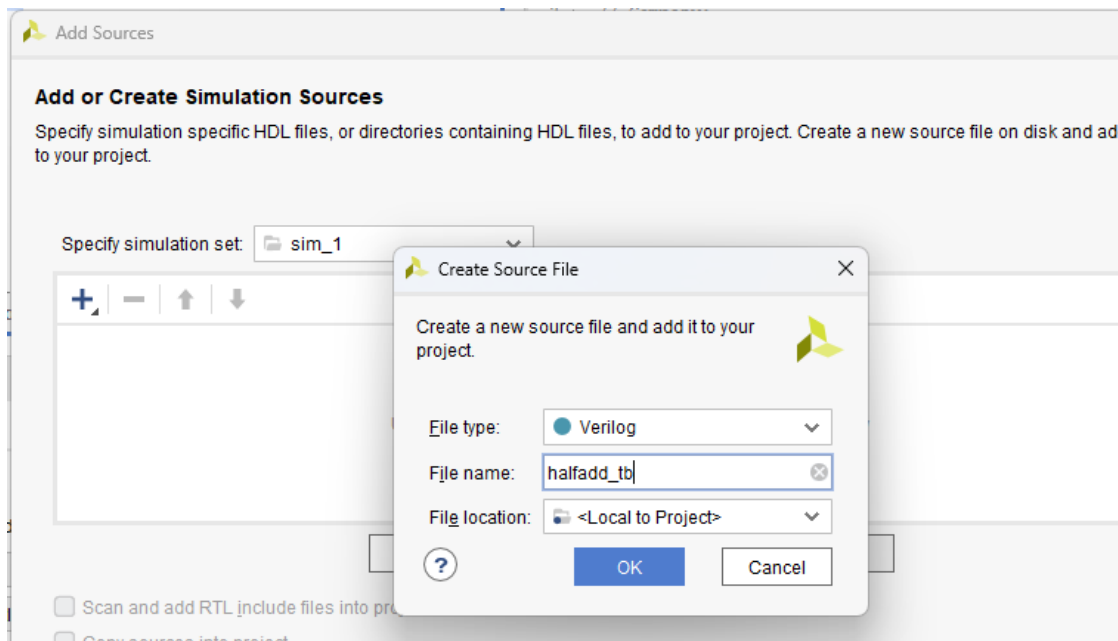


Figure 20: Name the file and click OK.

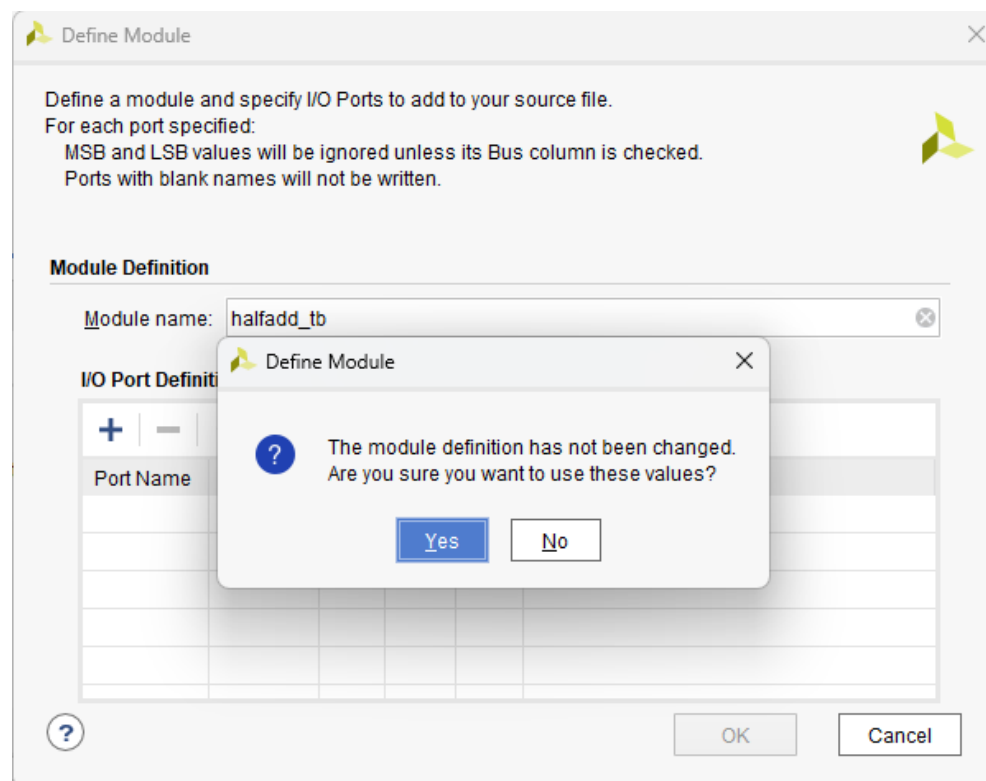
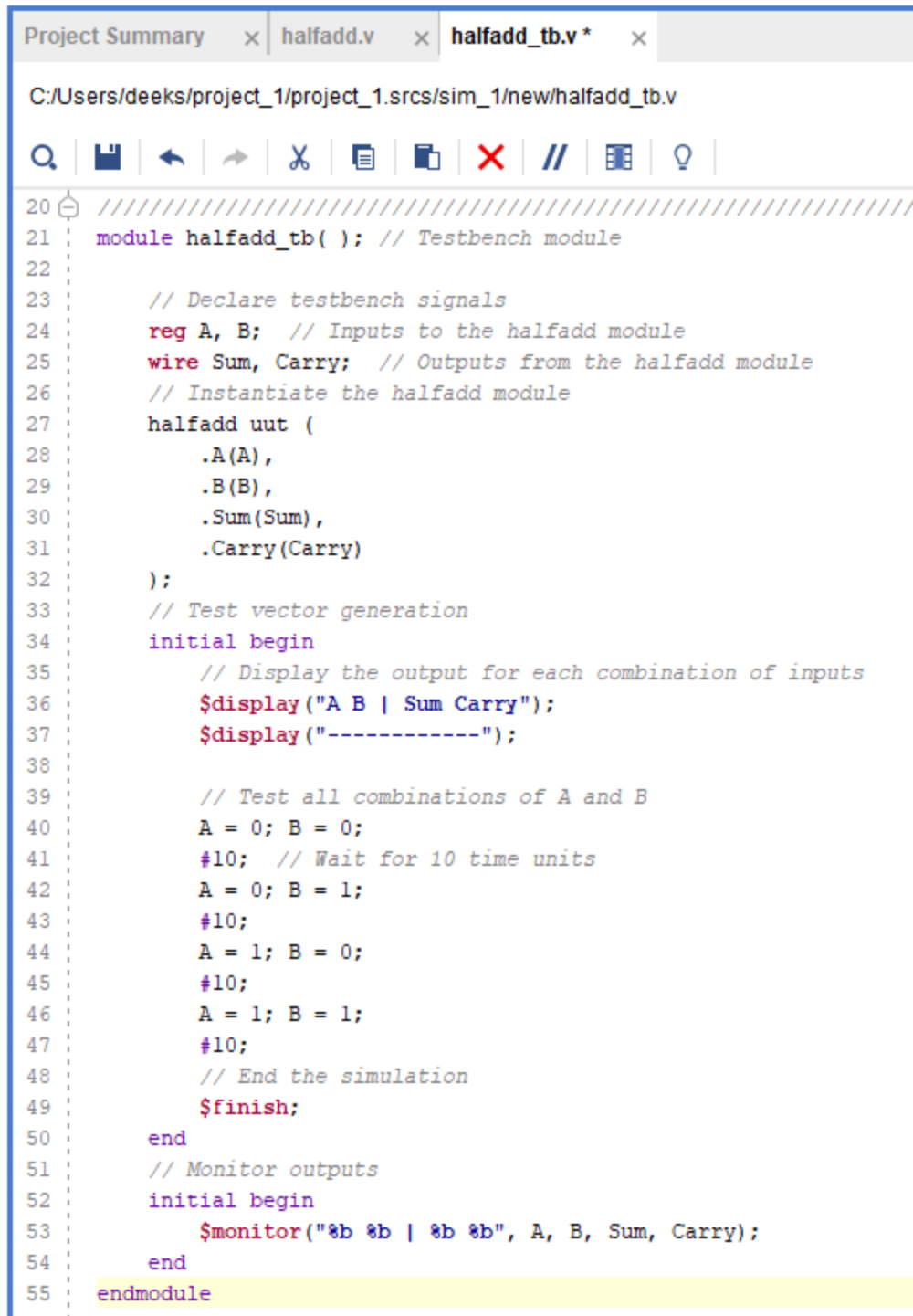


Figure 20: Click Yes and then OK.

The image shows a screenshot of a text editor window with three tabs: 'Project Summary', 'halfadd.v', and 'halfadd_tb.v *'. The active tab is 'halfadd_tb.v *', which contains Verilog testbench code. The code is for a module named 'halfadd_tb' and includes signal declarations, module instantiation, and test vector generation. The code is color-coded: keywords are in purple, comments in grey, and identifiers in red. Line numbers 20 through 55 are visible on the left margin. The code ends with 'endmodule' on line 55, which is highlighted in yellow.

```
20 ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
21 module halfadd_tb( ); // Testbench module
22
23     // Declare testbench signals
24     reg A, B; // Inputs to the halfadd module
25     wire Sum, Carry; // Outputs from the halfadd module
26     // Instantiate the halfadd module
27     halfadd uut (
28         .A(A),
29         .B(B),
30         .Sum(Sum),
31         .Carry(Carry)
32     );
33     // Test vector generation
34     initial begin
35         // Display the output for each combination of inputs
36         $display("A B | Sum Carry");
37         $display("-----");
38
39         // Test all combinations of A and B
40         A = 0; B = 0;
41         #10; // Wait for 10 time units
42         A = 0; B = 1;
43         #10;
44         A = 1; B = 0;
45         #10;
46         A = 1; B = 1;
47         #10;
48         // End the simulation
49         $finish;
50     end
51     // Monitor outputs
52     initial begin
53         $monitor("%b %b | %b %b", A, B, Sum, Carry);
54     end
55 endmodule
```

Figure 21: Add the test bench code as shown above and save the file.

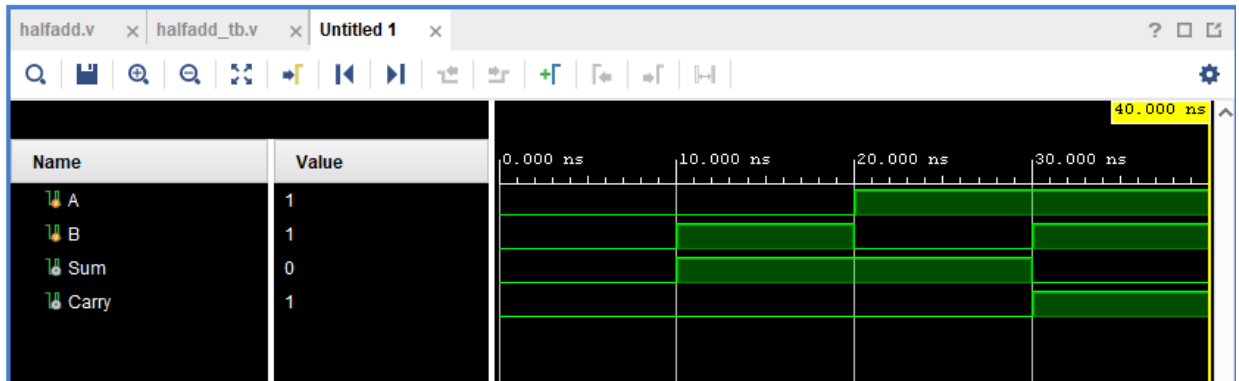


Figure 22: Click on Simulation and Run Behavioral Simulation. Behavioral Simulation will open with the following windows.

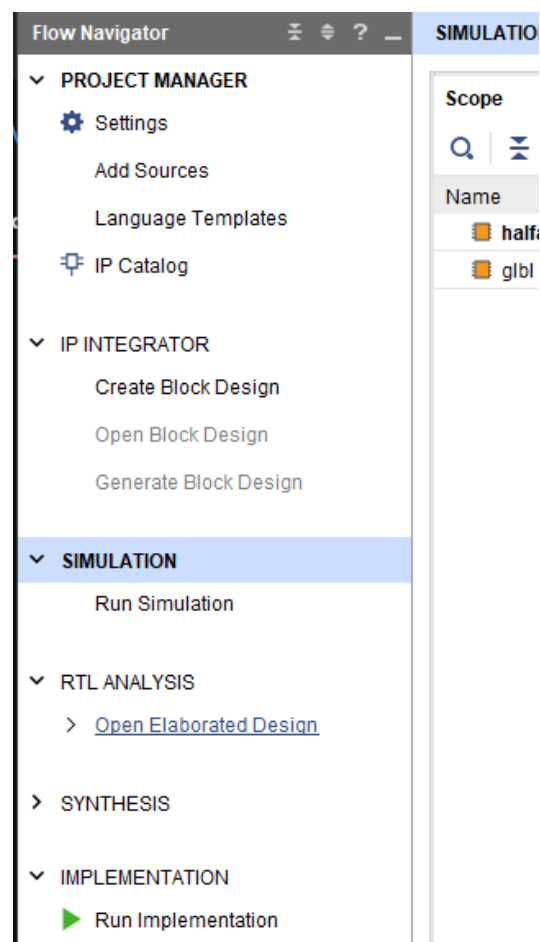


Figure 23: Now click on Open Elaborated Design from RTL ANALYSIS.

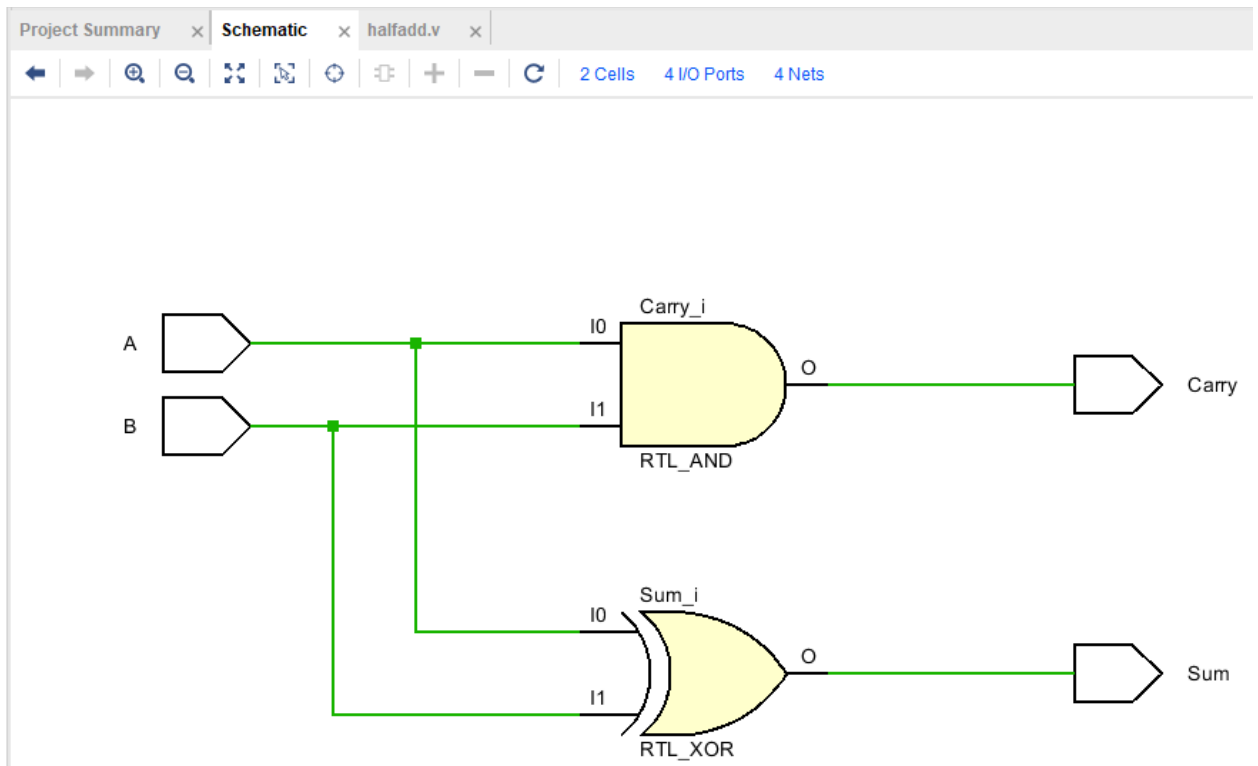


Figure 24: Schematic will be displayed.

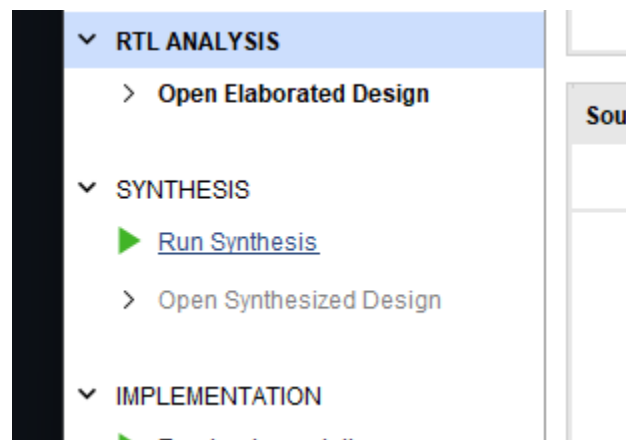


Figure 25: Now click on Run Synthesis.

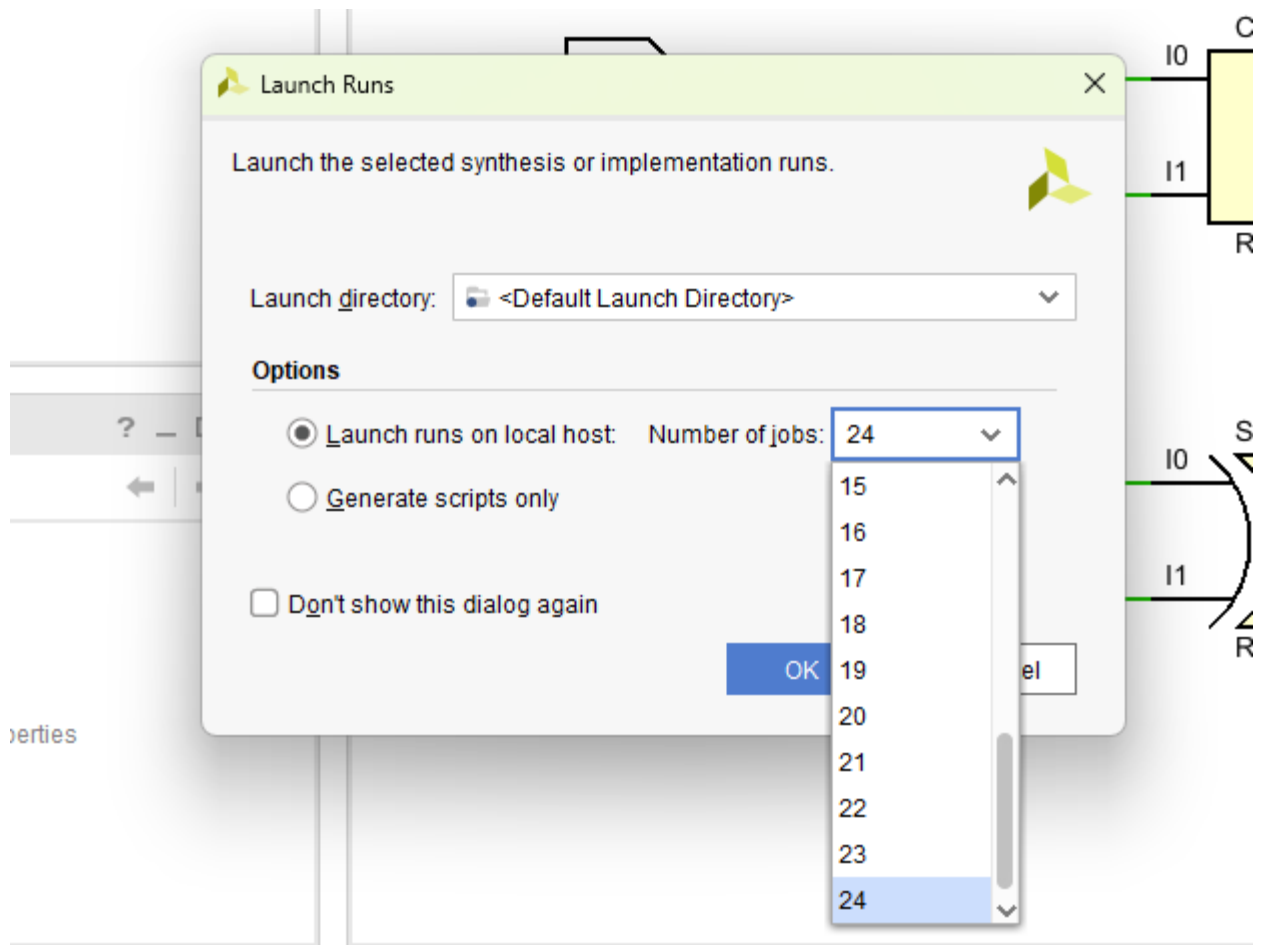


Figure 26: Select the Number of jobs according to your machine and click OK.

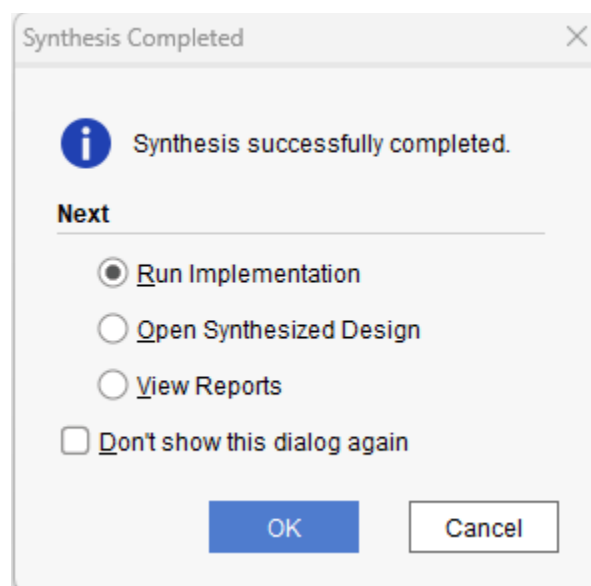


Figure 27: After successful synthesis, run the Implementation.

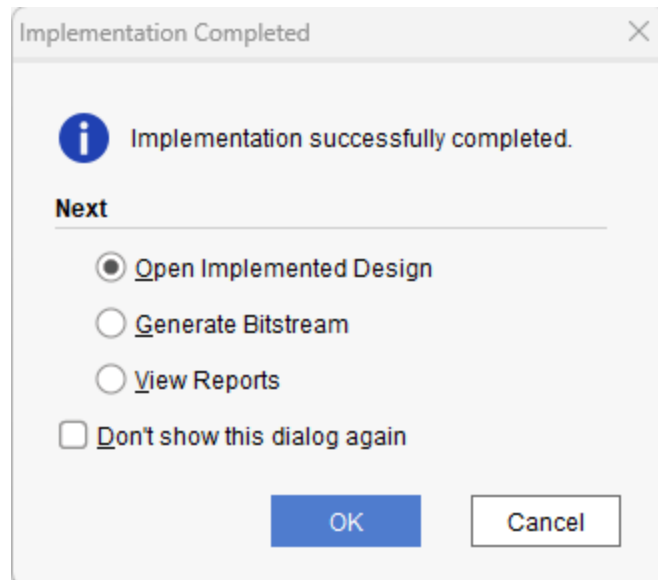


Figure 28: Open the Implemented Design after successful Implementation.

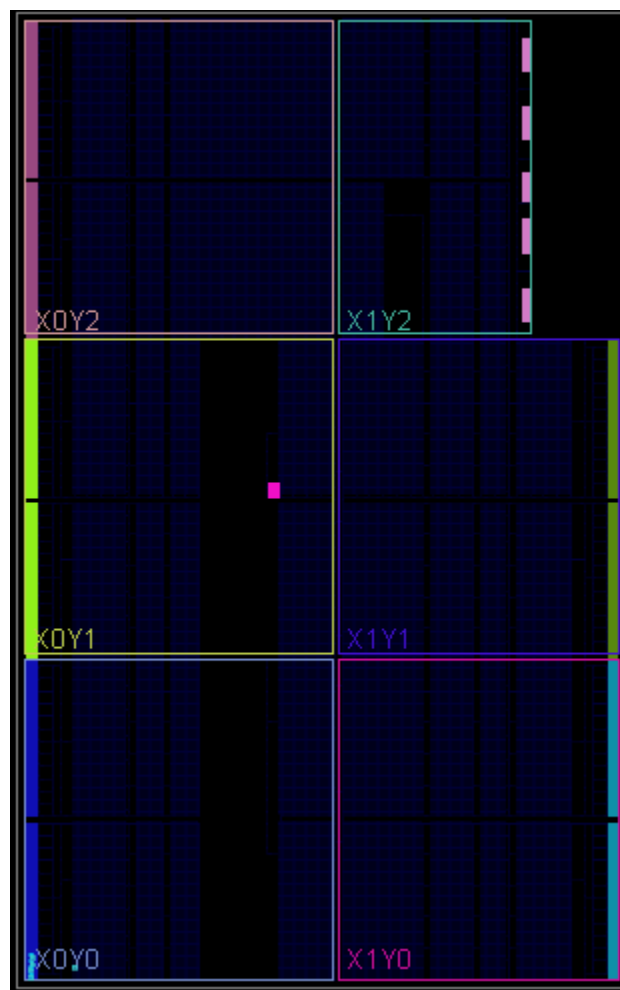


Figure 29: The Artix7 FPGA architecture

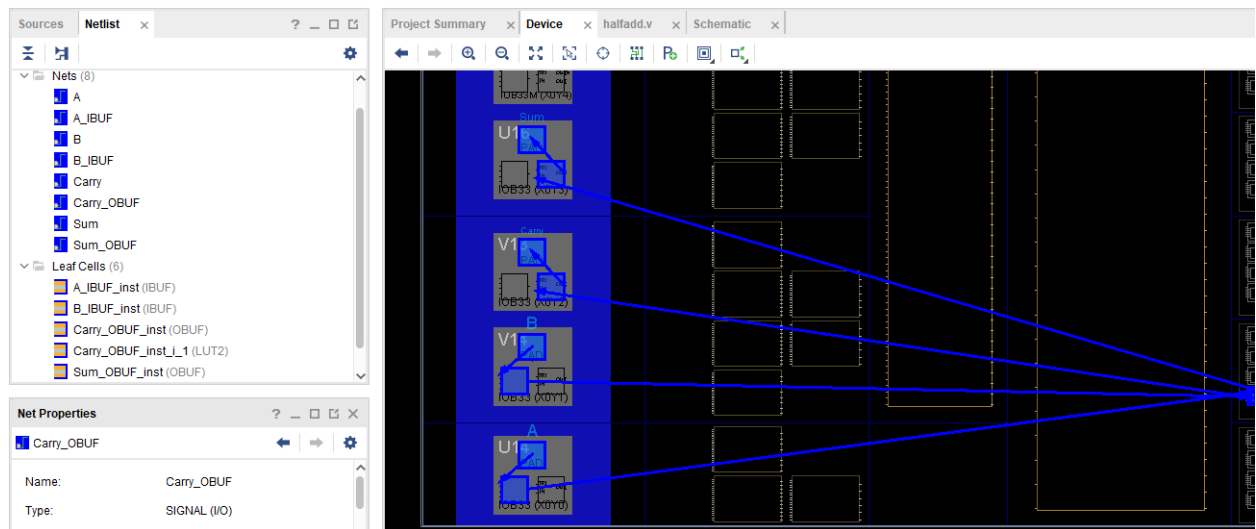


Figure 30: Zoom In to see the implemented Logic, and select different Nets to visualise the connections.

Project Summary x Device x halfadd.v x Schematic x

Overview | Dashboard

Status: ✔ Complete

Messages: ! 1 warning

Part: xc7a35tcpg236-1

Strategy: [Vivado Synthesis Defaults](#)

Report Strategy: [Vivado Synthesis Default Reports](#)

Incremental synthesis: [None](#)

Status: ✔ Complete

Messages: ! 7 warnings

Part: xc7a35tcpg236-1

Strategy: [Vivado Implementation Defaults](#)

Report Strategy: [Vivado Implementation Default Reports](#)

Incremental implementation: [None](#)

DRC Violations

Summary: ! 2 critical warnings
! 1 warning

[Implemented DRC Report](#)

Timing Setup | Hold | Pulse Width

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-Synthesis | Post-Implementation

Graph | Table

Resource	Utilization	Available	Utilization %
LUT	1	20800	0.01
IO	4	106	3.77

Power Summary | On-Chip

Total On-Chip Power: 0.733 W

Junction Temperature: 28.7 °C

Thermal Margin: 56.3 °C (11.2 W)

Effective θ_{JA} : 5.0 °C/W

Power supplied to off-chip devices: 0 W

Confidence level: [Low](#)

[Implemented Power Report](#)

Figure 31: Project Summary gives details on resource utilization, timing and other details.

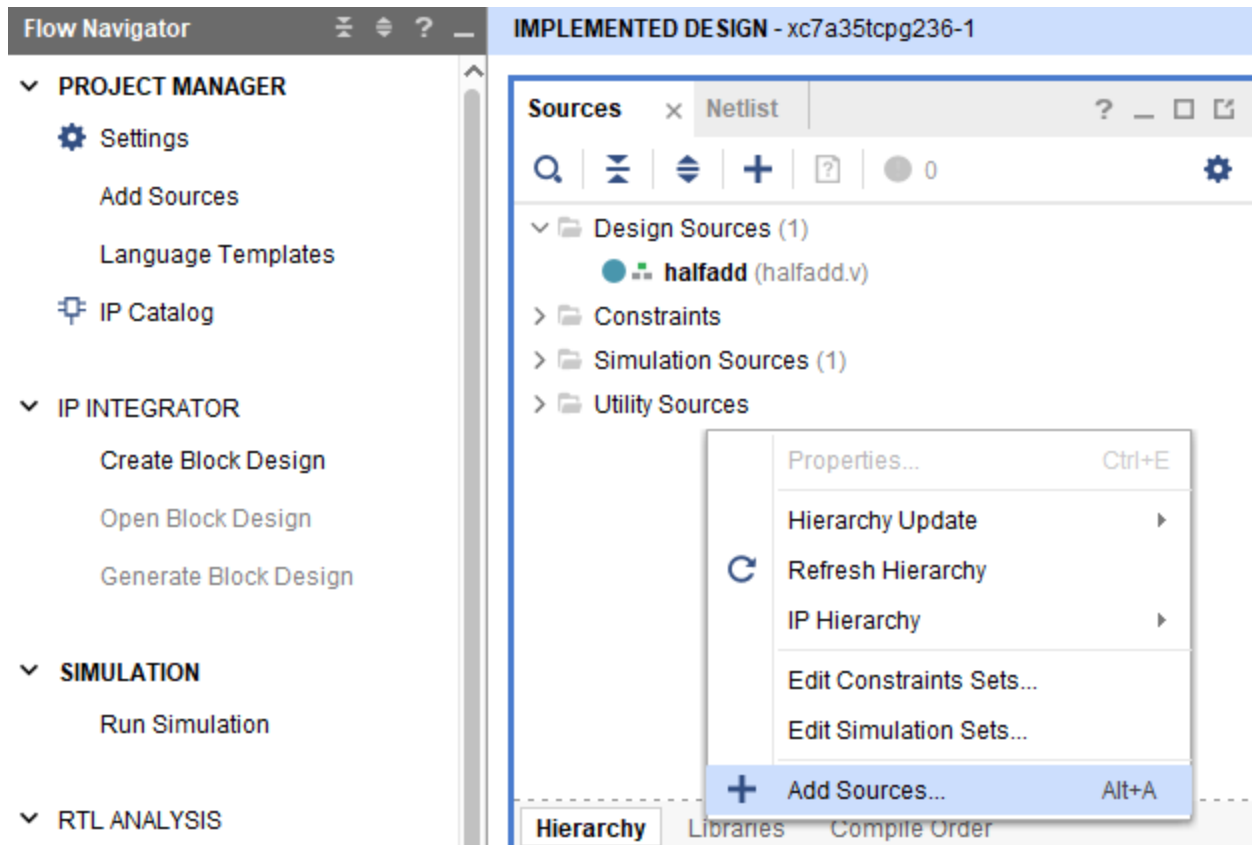


Figure 32: After successful implementation add the .xdc file into the project.

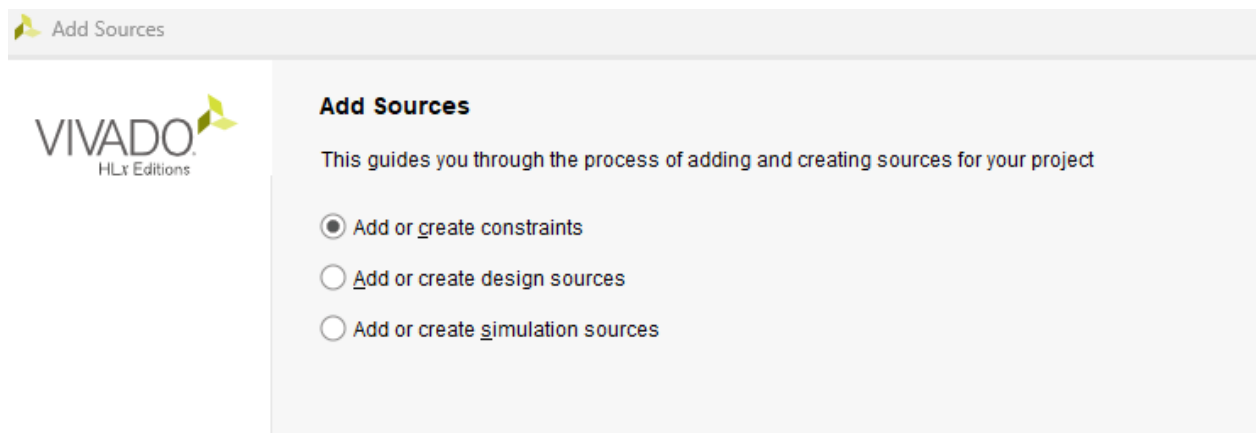


Figure 33: Select Add constraints option.

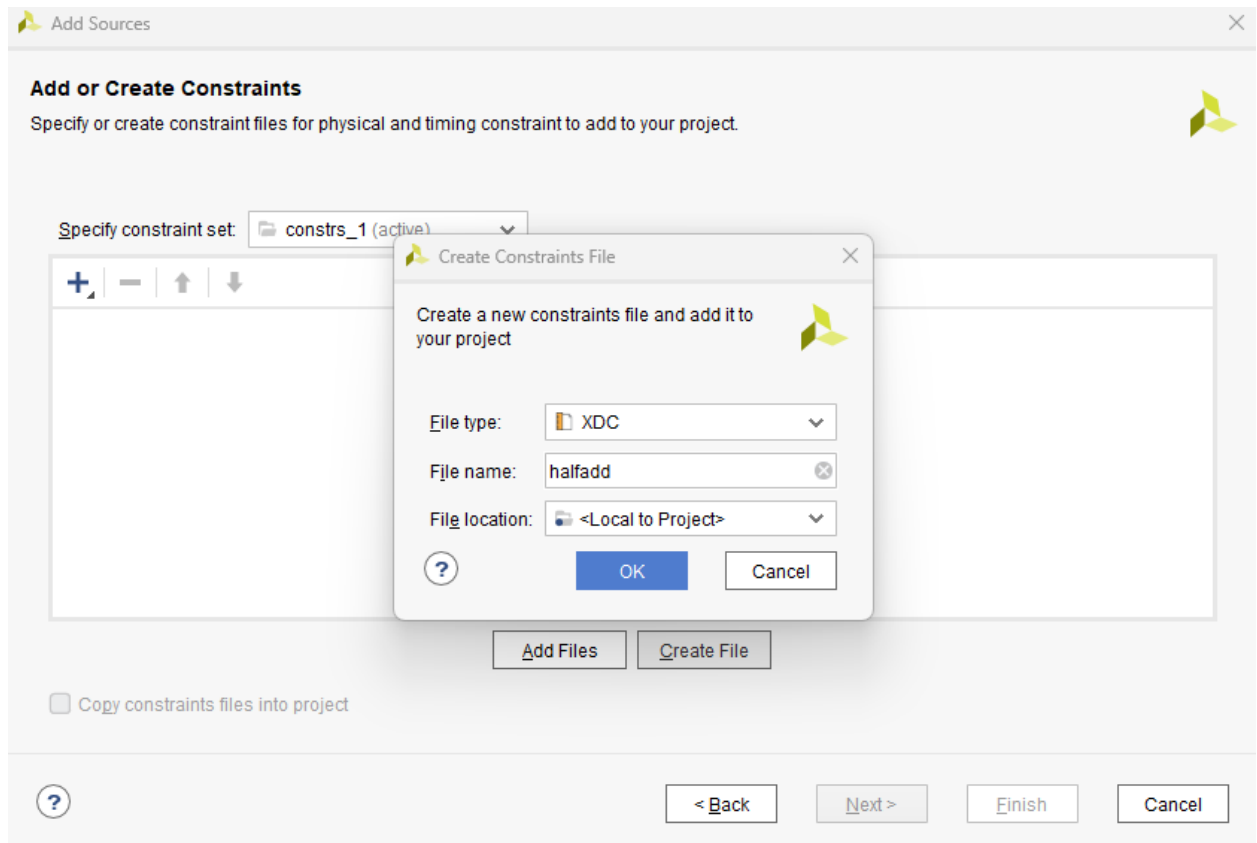


Figure 34: Create File and give a name to your .xdc file

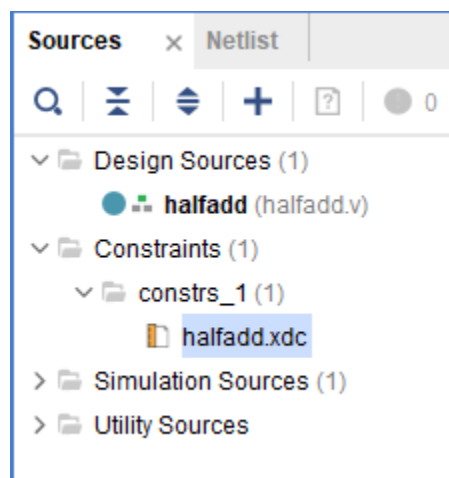


Figure 35: .xdc file shows up in the Sources window under Constraints. Now open the file and write the constraints to make connections between the IOs and the peripherals on the board. (set_property PACKAGE_PIN <pin> [get_ports <port>];
set_property IOSTANDARD LVCMOS33 [get_ports <port>])

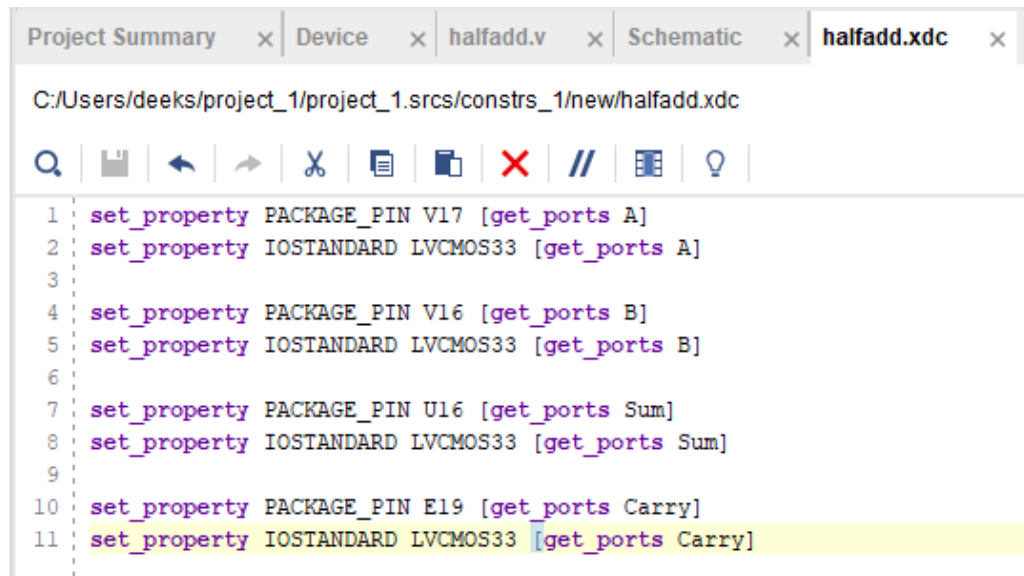


Figure 36: The constraint file is added according to the basys3 board input and output pins. ([Source](#))

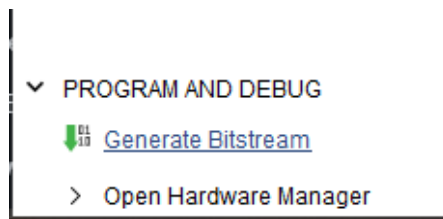


Figure 37: After completing the .xdc file, click on Generate Bitstream.

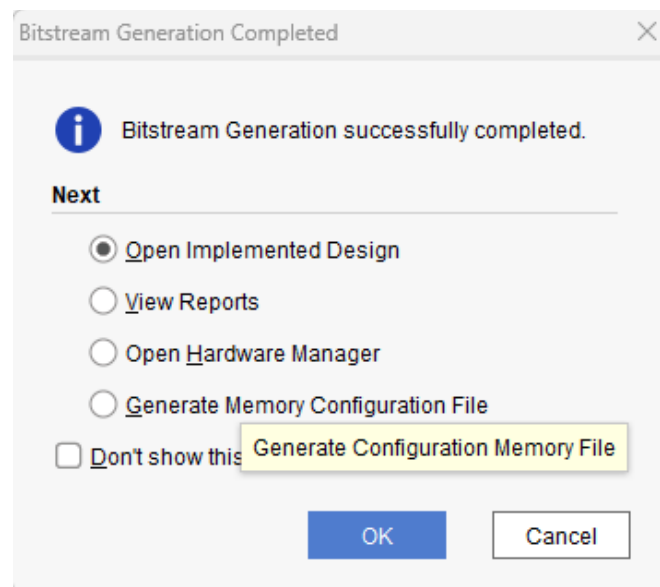


Figure 38: The above window will appear on successful Bitstream Generation.