

CE/CZ3001

LAB-1

In this lab you need to find the area and time complexity of arithmetic circuits like adders and multipliers and circuits for basic logic operations for different bit-widths. Besides, you will experiment with the functionality of ALU as well as the area and time complexity of ALU of different bit-widths. You need to find out which of the arithmetic or logic circuits affect the area and time complexity of ALU significantly, and the impact of word-length on such complexity. You are provided with parametrizable Verilog code where the bit-width can be changed. You should also understand the Verilog code completely and should develop competence to write the code which would be required in your projects.

PART I: ARITHMETIC CIRCUITS

In this part of the Lab, we consider a simple adder and a multiplier. You will experiment with the functionality of each arithmetic circuit and find their area and time complexity for different bit-widths.

ARITHMETIC CIRCUITS DESCRIPTION

1. **Adder:** The function of an adder is to add two input operands (A and B) to produce output (out). You can change the bit width of inputs to find the area and delay complexity of the adders of different bit-width.

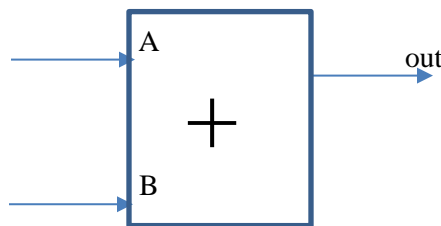


Figure-1: Block diagram of adder.

2. **Multiplier :** The function of the multiplier is to multiply two input operands (A and B) to produce the output (out). Assuming both A and B to be W-bit words in 2's complement representation, we can find that 'out' is a (2W)-bit word in 2's complement representation.

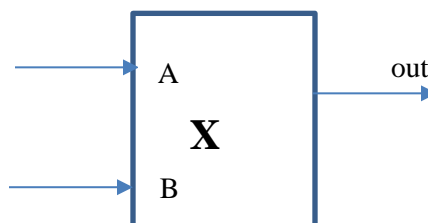


Figure-2: Block diagram of multiplier.

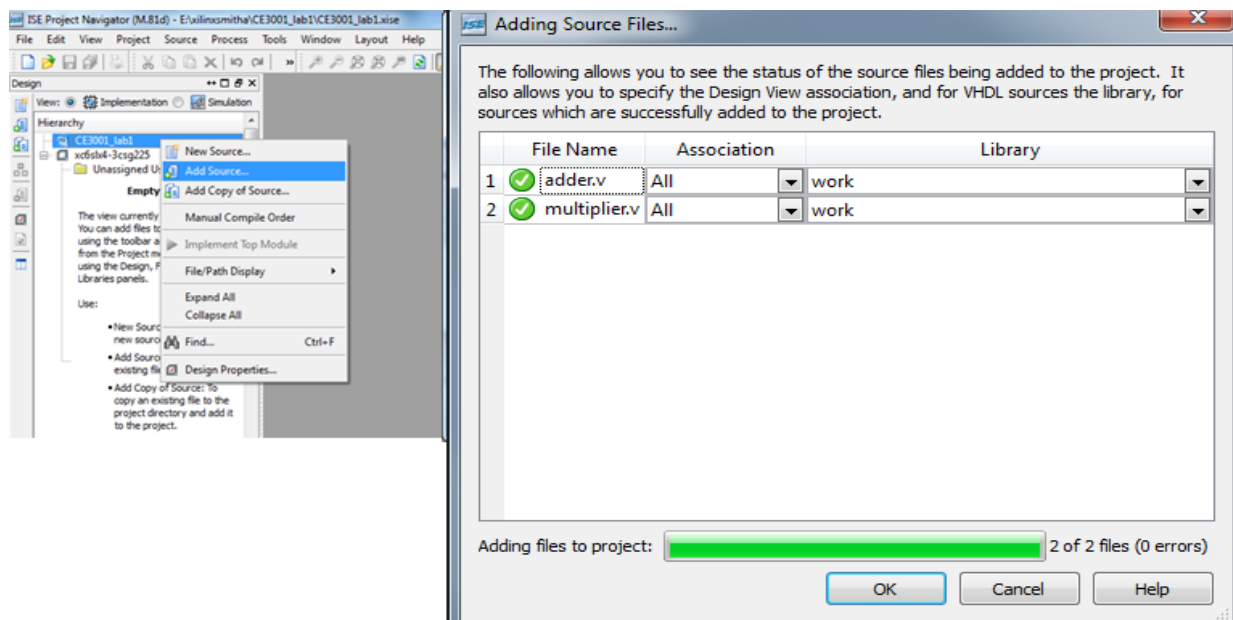
ARITHMETIC CIRCUITS SYNTHESIS, TESTING AND ANALYSIS

- 1) You will be given the Verilog codes of an adder and a multiplier. You have to generate the test bench and test whether the Verilog modules give correct results.
- 2) You need to take the input bit-width 8, 16, 32 and 64 and find out the number of slices used and the maximum combinational path delay of adders and multipliers of different bit-widths. You need to plot area (vs) bit-width as well as delay (vs) bit-width for the adder and the multiplier modules.

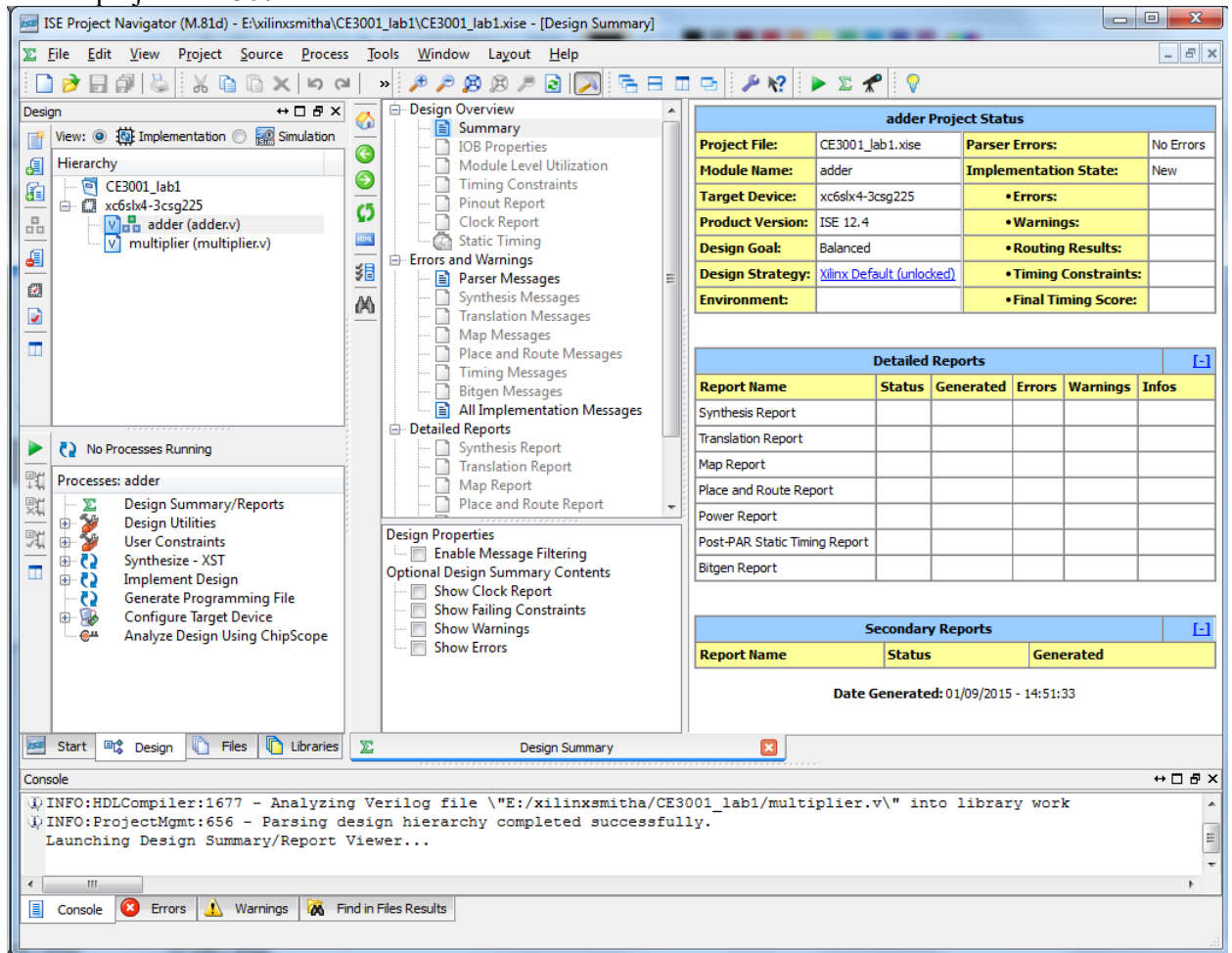
To find out whether the design is functioning correctly you need to synthesise code and to see if that gives the correct results. Besides you need to analyse the increase in complexity (in terms of number of slices and computational delay) of adder and multiplier along with the increase in bit-width. For that you proceed as follows.

A. Open Project and Download Source Verilog files.

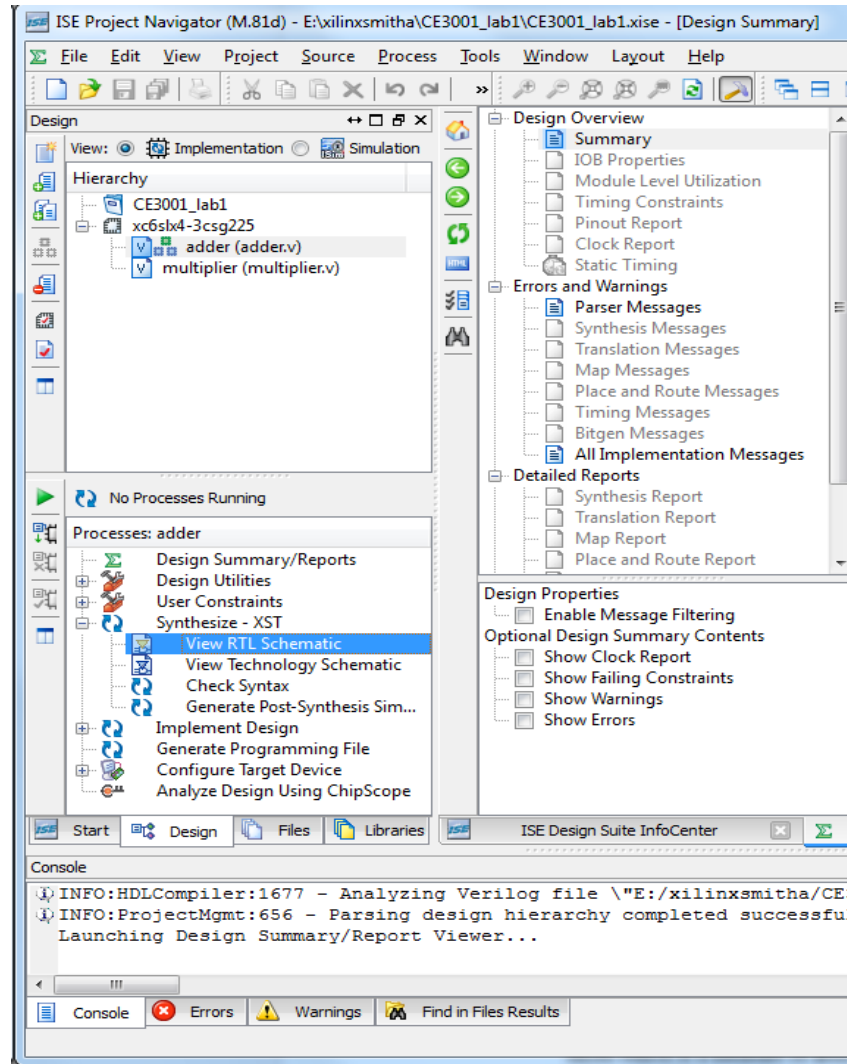
- 1) Open Xilinx ISE Project navigator and start a new project “CE3001_lab1”.
- 2) The project settings are as below
 1. In the field family, select >Spartan6
 2. In the field device, Select > XC6SLX4
 3. In the field package, Select > CSG225
 4. In the field speed, Select > -3
 5. In the field Simulator, Select > ISim(VHDL/Verilog)
 6. In the field Preferred Language, Select > Verilog
 7. Click > Next to move to the page Project Summary
 8. Click > Finish in the page Project Summary
- 3) This will create a folder by the name “CE3001_lab1”. You may copy the files ‘adder_v’, and ‘multiplier.v’ to this folder.
- 4) Once the project is open, we need to ‘Add source’ by adding ‘adder_v’, and ‘multiplier.v’ file.



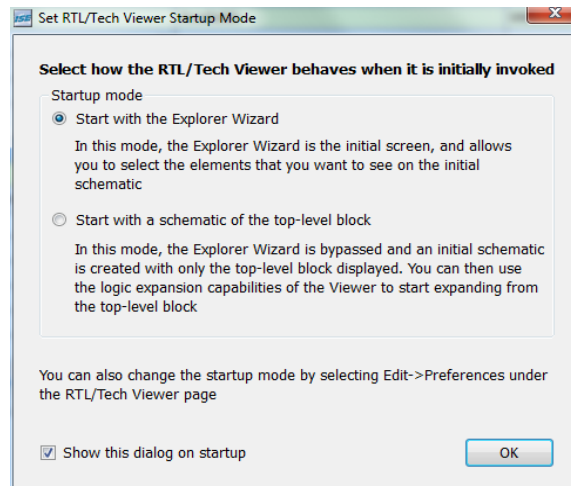
- 5) Click 'OK' for adding the source files "adder.v" and "multiplier.v" will be added to the project "CE3001lab1".



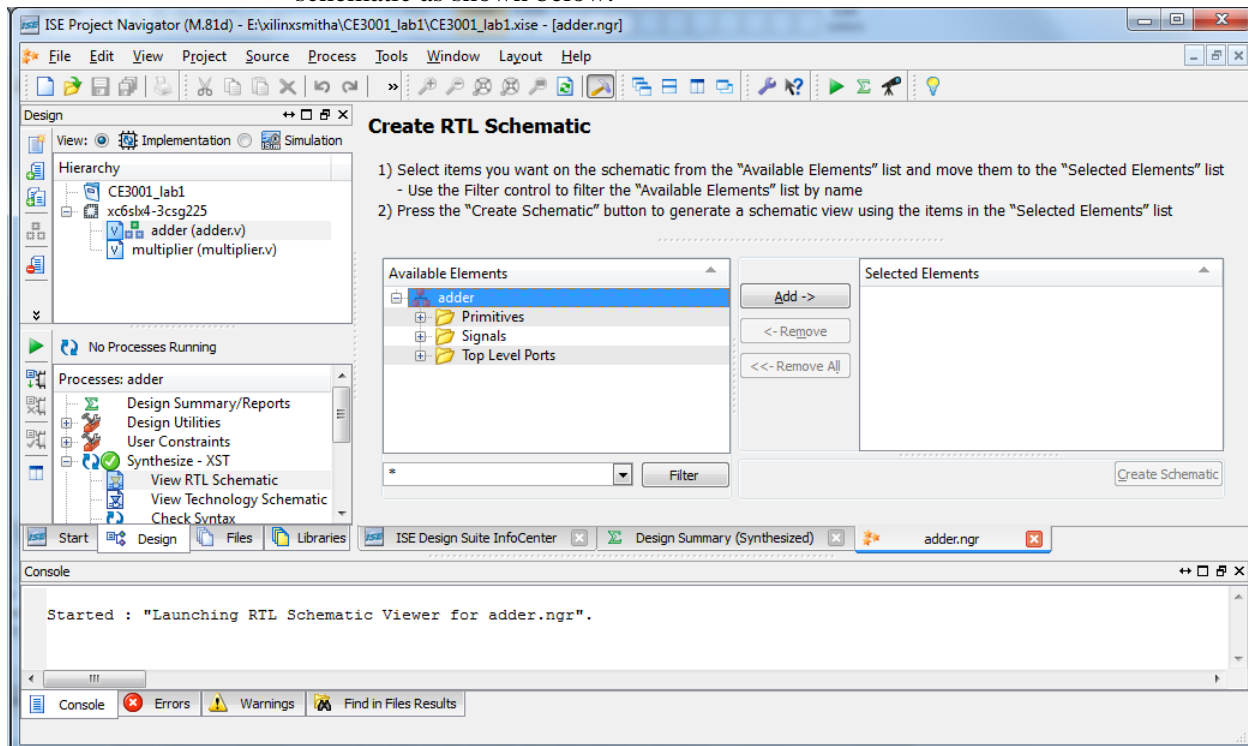
- 6) Double click the adder module and go through your 'adder.v' code to understand that. In order to synthesize 'adder.v' right click 'adder' and select 'Set as Top Module'.
- 7) To synthesize the 'adder.v' code and to see the schematic diagram of the adder, click on to the 'synthesize-XST' button in the 'Design' tab. Now, expand the 'synthesis' button and click on to 'View RTL schematic'.



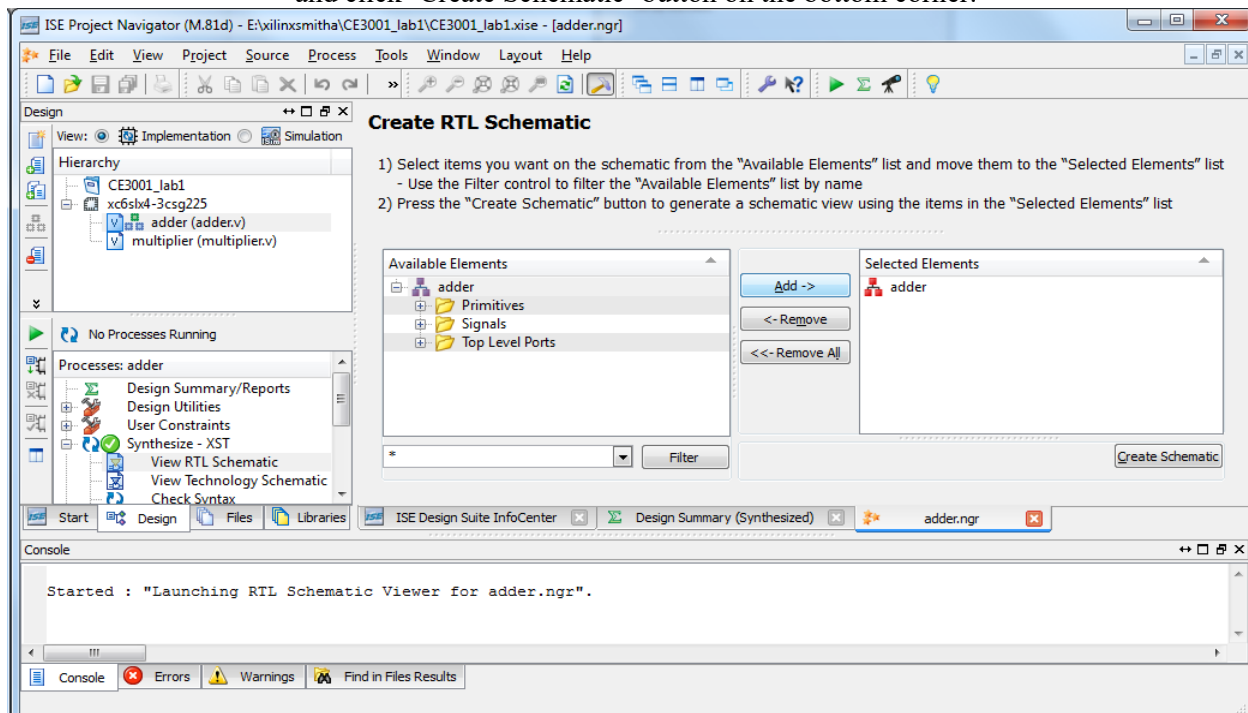
- 8) This process will initiate the synthesis of the code (number of Look up Table (LUT) slices which is a measure of area and delay of the circuit) and also will give the RTL schematic of the code.



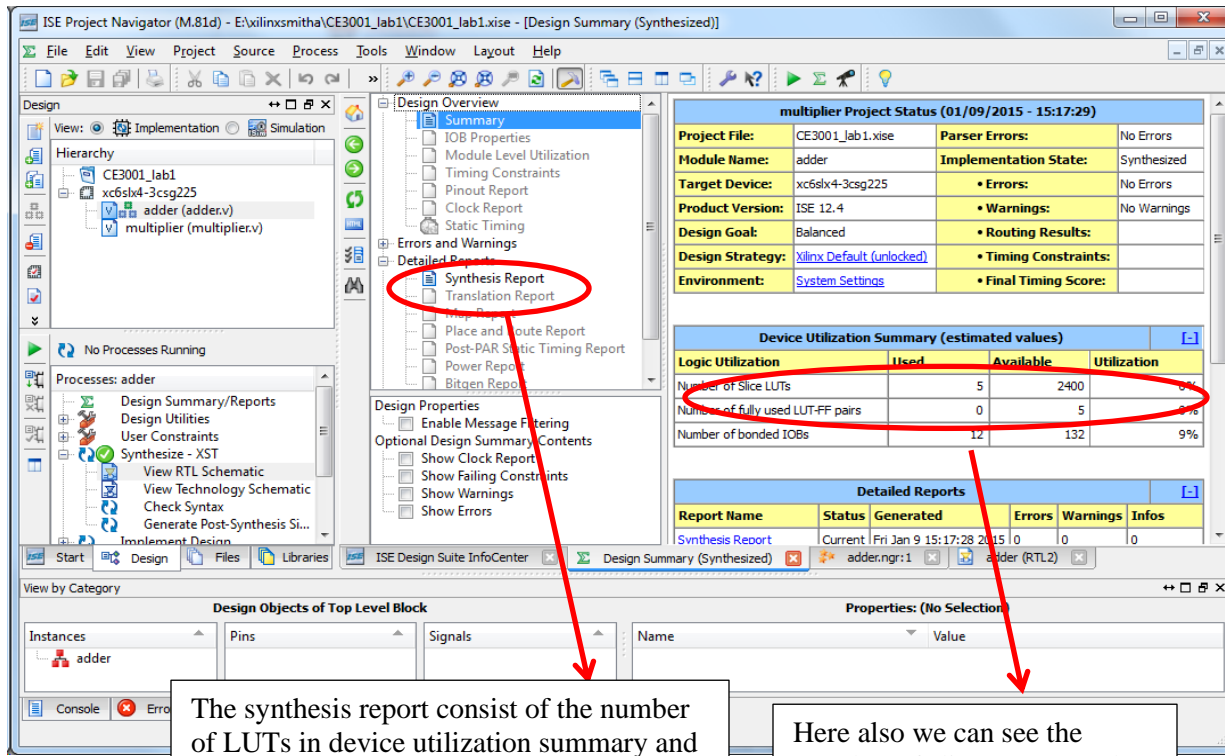
- a. Click Ok. This will bring another window with the adder module to create the schematic as shown below.



- b. Click 'adder' under 'Available Elements' and "ADD" to the 'Selected Elements' and click 'Create Schematic' button on the bottom corner.



- 9) The schematic diagram can be seen. Click into the main module diagram to go into the circuit diagram of adder. You can re-verify the functionality by checking the circuit diagram. You can also know the amount of hardware (LUTs) used by clicking the “View Technology Schematic”
- 10) The synthesis result can be found by clicking the ‘Synthesis report’ in the ‘Design overview- synthesis report’ tab.



The screenshot shows the ISE Project Navigator window for a project named 'multiplier'. The 'Design Overview' tab is active, and the 'Synthesis Report' is selected in the 'Detailed Reports' list. A red circle highlights the 'Synthesis Report' in the list. Another red circle highlights the 'Device Utilization Summary (estimated values)' table. A red arrow points from the 'Synthesis Report' circle to a callout box, and another red arrow points from the 'Device Utilization Summary' circle to another callout box.

multiplier Project Status (01/09/2015 - 15:17:29)

Project File:	CE3001_lab1.xise	Parser Errors:	No Errors
Module Name:	adder	Implementation State:	Synthesized
Target Device:	xc6slx4-3csg225	Errors:	No Errors
Product Version:	ISE 12.4	Warnings:	No Warnings
Design Goal:	Balanced	Routing Results:	
Design Strategy:	Xilinx Default (unlocked)	Timing Constraints:	
Environment:	System Settings	Final Timing Score:	

Device Utilization Summary (estimated values)

Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	5	2400	0.21%
Number of fully used LUT-FF pairs	0	5	0.00%
Number of bonded IOBs	12	132	9.09%

Detailed Reports

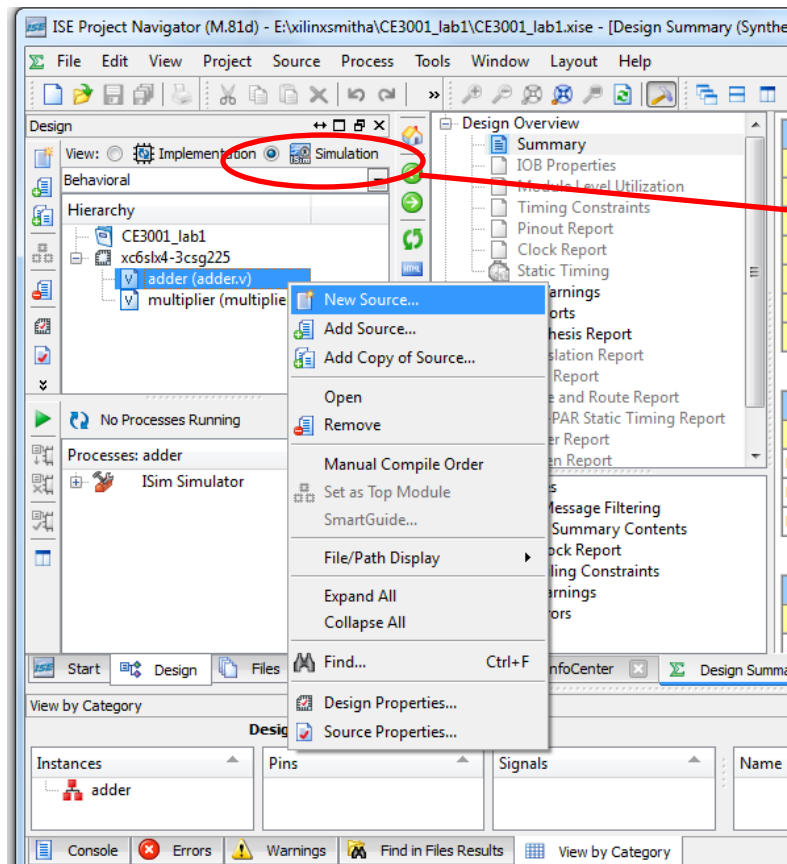
Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Fri Jan 9 15:17:28 2015	0	0	0

The synthesis report consist of the number of LUTs in device utilization summary and the delay in timing summary

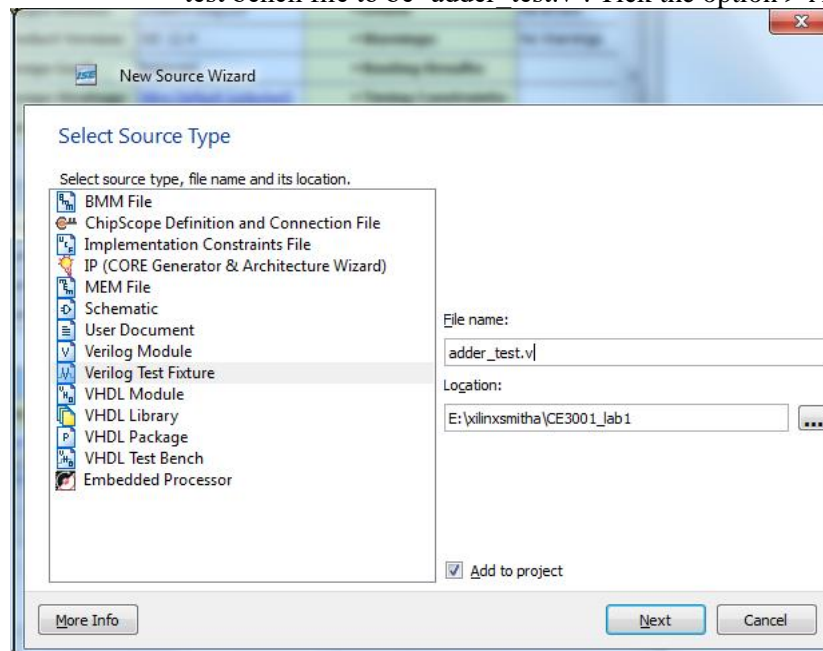
Here also we can see the Number of slice LUTs used.

- 11) To do simulation of the code given we need a test bench. In ISE Design Suite, click ‘View-> Simulation’.

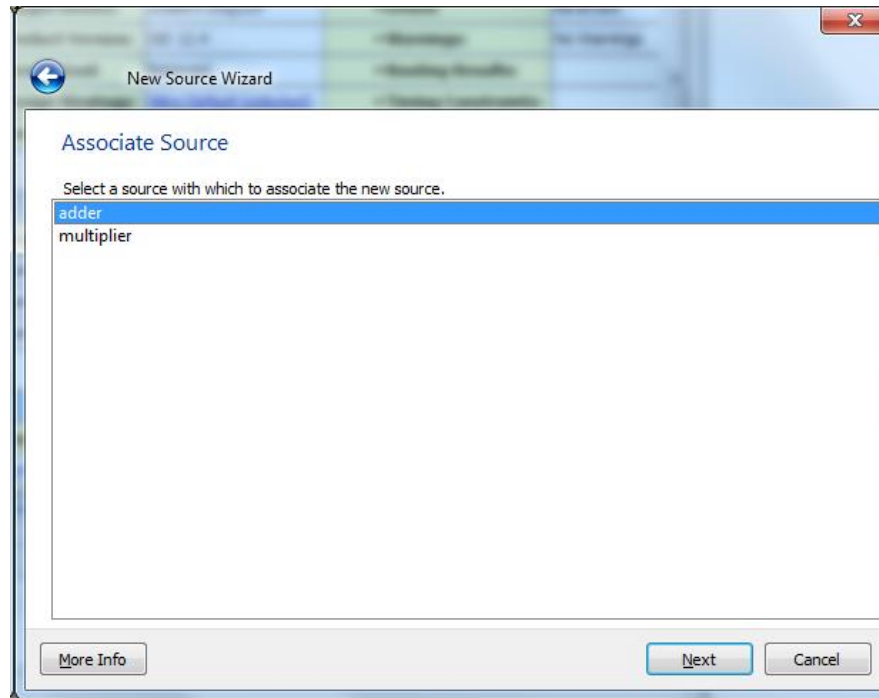
- In the Design Panel, right-click on the icon adder and select >New Source to move to the page Select Source Type



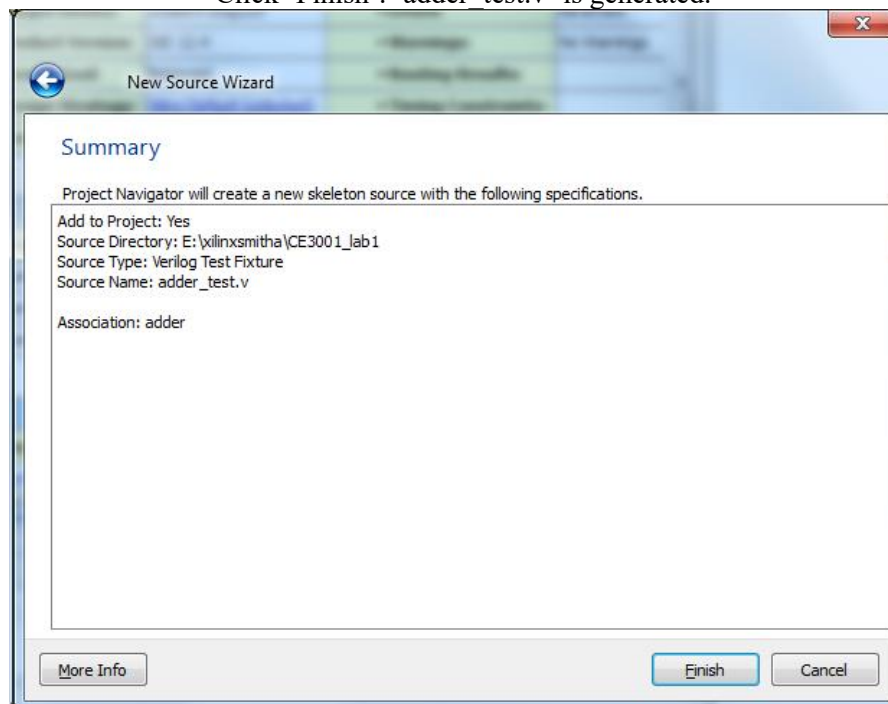
- The page Select Source Type Page ‘Verilog Test Fixture’ and name the test bench file to be ‘adder_test.v’. Tick the option > Add To Project.



- Click > Next to move to the page Associate Source

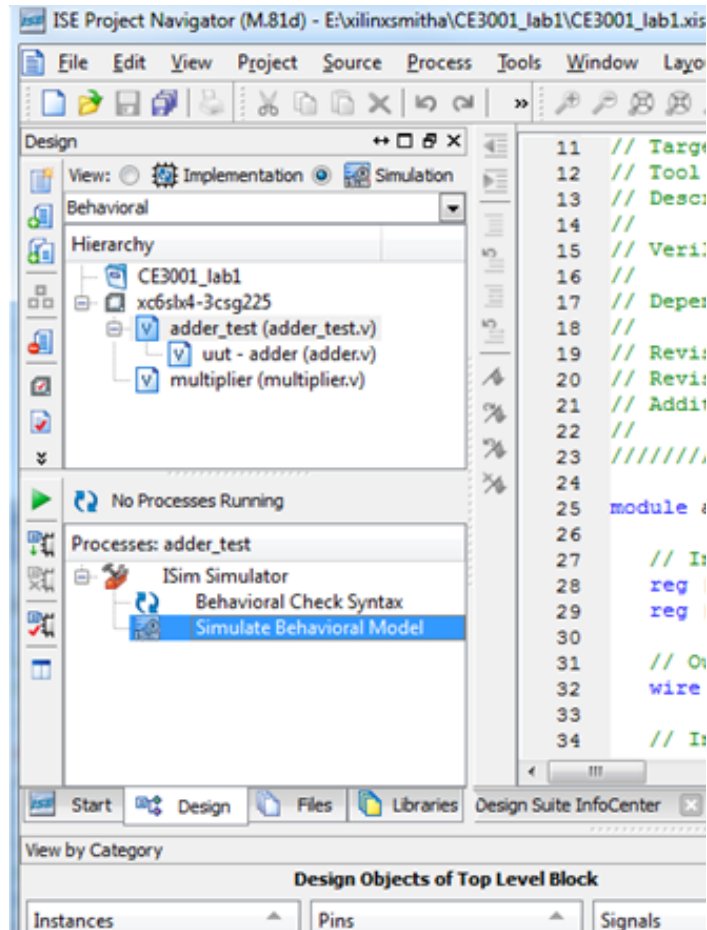


- Click the module that we want to simulate. In the current case 'adder'. Select that and click next.
- Click 'Finish'. 'adder_test.v' is generated.




```
19 // Revision:
20 // Revision 0.01 - File Created
21 // Additional Comments:
22 //
23 ///////////////////////////////////////////////////////////////////
24
25 module adder_test;
26
27     // Inputs
28     reg [7:0] a;
29     reg [7:0] b;
30
31     // Outputs
32     wire [7:0] out;
33
34     // Instantiate the Unit Under Test (UUT)
35     adder uut (
36         .a(a),
37         .b(b),
38         .out(out)
39     );
40
41     initial begin
42         // Initialize Inputs
43         a = 0;
44         b = 0;
45
46         // Wait 100 ns for global reset to finish
47         #100;
48
49         // Add stimulus here
50
51
52     end
53
54 endmodule
```

- 12) Add the following inputs below the ‘//Add stimulus here’ section in the program.
200 a=8’h01; //after 200ns make a=1;
#200 b=8’h02; // after 200ns make b=1;
- 13) Click ‘Simulate Behavioural model’ below ‘ISIM simulator’ for the processes ‘adder_test’ to generate the output waveform. Verify whether the functionality of adder is correct.



B: The **adder** code is parametrizable. That means you can change the bit-width of the adder by changing the value for parameter **DSIZE**. If parameter **DSIZE** is set to 8, then after synthesis an eight bit adder circuit will be created in the FPGA. If the parameter **DSIZE** =64, then the bit-width of the operands of the adder would be 64 bit.

- 1) Set the parameter **DSIZE** 8, 16, 32 and 64 one by one and note the number of slices used and delay (in ns) in different cases. You need to synthesise each time (steps 6-13) after you change the parameter value to find out the number of slices and delay in ns. Plot the graph, area (vs) bit-width as well as delay (vs) bit-width for the adder module.

Table 1: Slices and delay for adder

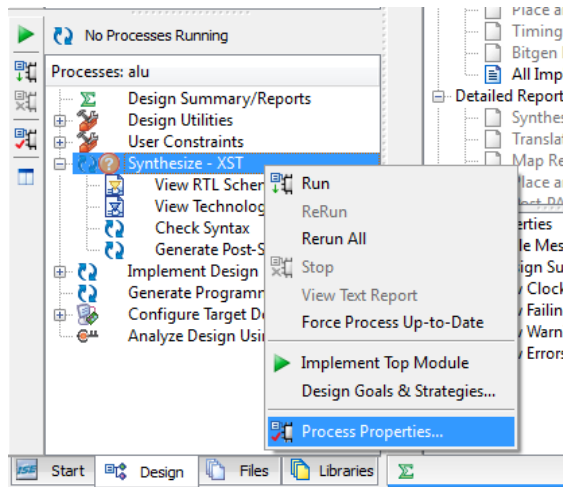
Parameter: DSIZE	BIT-WIDTH	No of LUT slices	Delay in ns
8	8		
16	16		
32	32		
64	64		

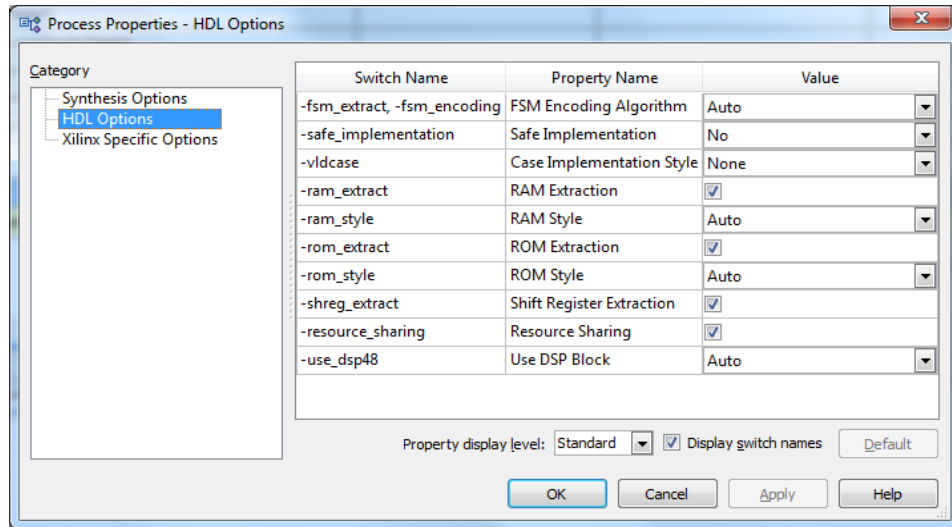
To synthesize and to see its circuit diagram you need to make it as the topmodule and **repeat from steps 6-13**. You may have to provide input values of proper bit-width to the testbench program for adder in-order to verify the same. Note that the bit-width of the inputs is same as the bit-width parameter that you have set in the code. To view the technology schematic follow the step 8, but instead of clicking ‘view RTL schematic’,

click ‘View Technology Schematic’ . You can also note the Technology schematic diagram reflects the increased bit-width of the adder.

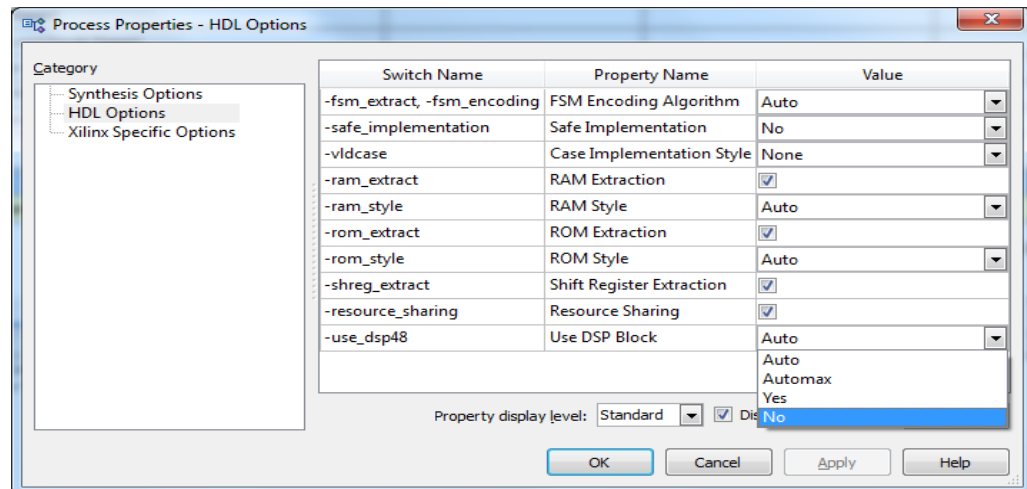
C: The given **Multiplier** code is also parametrizable. That means you can change the bit-width of the operands of multiplier by changing the value for parameter **DSIZE**; If parameter **DSIZE**=8, then the bit-width of the operands in multiplier is 8 bit. If parameter **DSIZE**=64, then the bit-width of the operands of the multiplier would be 64 bit.

- 1) Set the parameter **DSIZE** 8, 16, 32 and 64(one by one) and note the number of slices occupied and delay in terms of (ns) for all cases. You need to synthesise each time (steps 6-13) after you change the parameter value to find out the number of slices and delay in ns. Plot the graph, area (vs) bit-width as well as delay (vs) bit-width for the multiplier module.
- 2) To synthesize and to see its circuit diagram set it as the topmodule (indicated in step 6) and repeat from steps 7-13. You have to add respective input values to the testbench program for multiplier in-order to verify the same. Note that the bit-width of the inputs is the same as the bit-width parameter that you set in the code. To view the technology schematic follow the step 8, but instead of clicking ‘view RTL schematic’, click ‘View Technology Schematic’ to see the increased bit-width of the multiplier.
- 3) Before doing step 7(the synthesis), click the ‘HDL option’ button in the ‘Process Properties’ tab. Note that there is a ‘Switch Name- use_dsp48’ as the last option.





- 4) The use of DSP48 block has to be prevented and hence select 'No' and Click-> 'OK'.



- 5) Continue with the rest of the steps as earlier.

Table 2: Slices and delay for Multiplier

Parameter: DSIZE	BIT-WIDTH	No of LUT slices	Delay in ns
8	8		
16	16		
32	32		
64	64		

EVALUATION -1

- 1) Plot the graph, area (vs) bit-width as well as delay (vs) bit-width for the adder module for DSIZE=8, 16, 32 and 64.
- 2) Plot the graph, area (vs) bit-width as well as delay (vs) bit-width for the multiplier module for DSIZE=8, 16, 32 and 64.

PARTII: ARITHMETIC LOGIC UNIT (ALU)

1. ARITHMETIC LOGIC UNIT (ALU) SPECIFICATIONS

In this part of the Lab, we consider a simple ALU that performs the computation for eight arithmetic and logical operation. The seven operations are: ADD, SUB, AND, XOR, COM, MUL and ADDI as described in Table 1.

Table 1 - Description of ALU Operations

Instruction	Equation	Operation	Description
ADD	$A+B$	Addition	Addition of A and B, where both A and B are in 2's complement representation
SUB	$A-B$	Subtraction	Subtraction of A and B, where both A and B are in 2's complement representation
AND	$A \& B$	Logical AND	Bit-wise AND of A, B
XOR	$A \wedge B$	Logical XOR	Bit-wise XOR of A, B
ORR	$A B$	Logical OR	Bit-wise OR of A, B
PassB	B	Only allow B to pass to output	Used for conditional branch; whether to check whether the content is equal to zero or not

A and B are the data input ports of the ALU to feed maximum of two operands to the ALU. The ALU has 3-bit control input to perform one out of the 6 possible instructions which the ALU can perform. The information is also listed in Table 2. The encoding of the 6 instructions is listed in Table 3.

Table 2 - Port List Specification. The bit-width to be varied from 8 bit to 64 bit.

Port Name	Port Direction	Description
A	Input	First operand
B	Input	Second operand
op	Input	Indicates the ALU about the operation to be performed
Out	Output	Output of the operation

Table 3 - ALU operation encoding

Operation	'op' value
ADD	000
SUB	001
AND	010
XOR	011
ORR	100
PassB	101

ALU IMPLEMENTATION, TESTING AND ANALYSIS

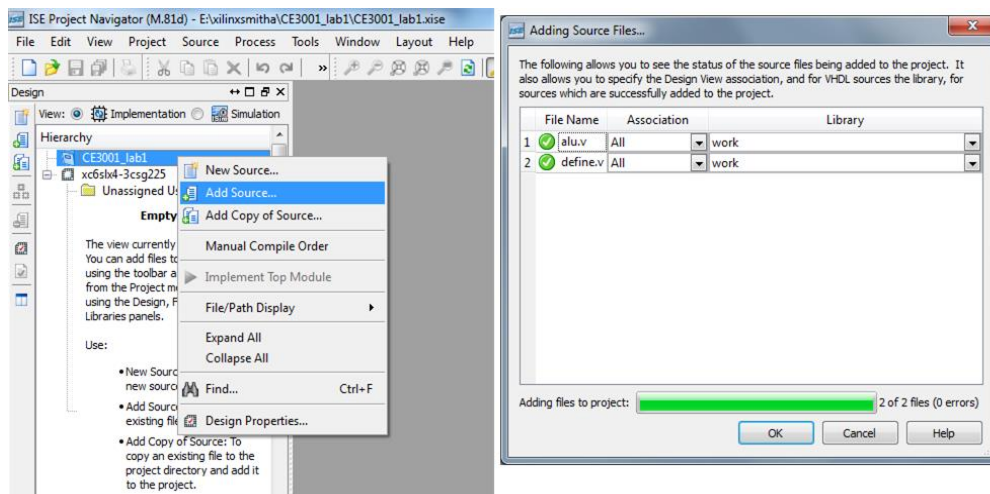
For this assignment,

- You are given the Verilog code as well as the test bench for ALU. You have to test whether the ALU gives correct results.
- You need to set the input bit-width to 8, 16, 32 and 64 (one by one) and find out the number of slices used and the maximum combinational path delay in each case. You need to plot area (vs) bit-width as well as delay (vs) bit-width for the ALU module. In FPGA you cannot find area directly so instead of area you can take number of slices, which would be considered proportional to the area.

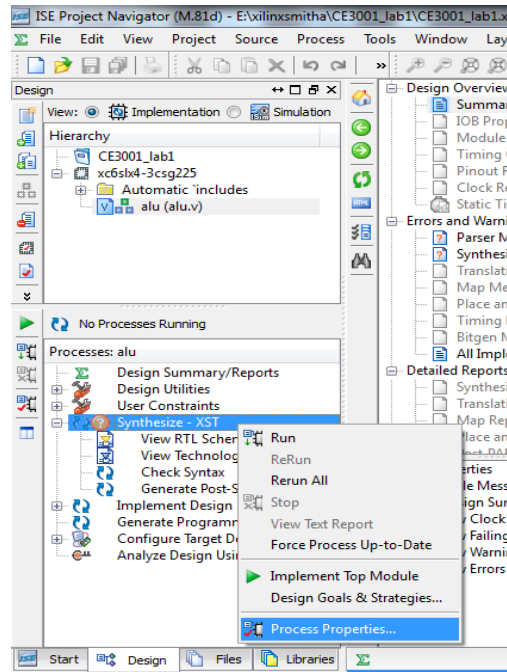
DETAILS

A. The given ALU code is parametrizable but initialized to 8 bits. To find out whether the design is functioning correctly you need to synthesise code and to see if that gives the correct results. For that you can proceed as follows.

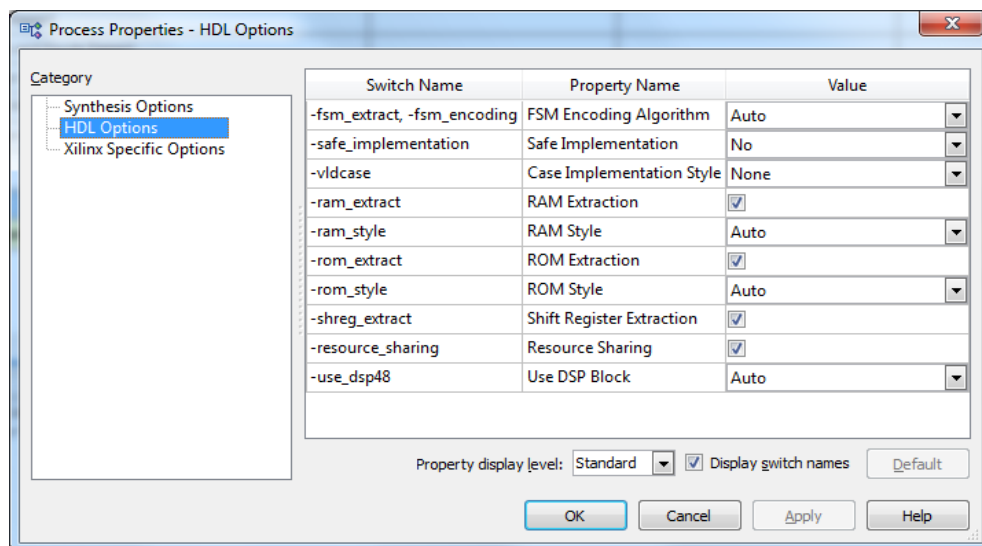
- 1) Open Xilinx ISE Project navigator and start a new project “CE3001_lab1”.
- 2) The project settings are as below
 1. In the field family, select >Spartan6
 2. In the field device, Select > XC6SLX4
 3. In the field package, Select > CSG225
 4. In the field speed, Select > -3
 5. In the field Simulator, Select > ISim(VHDL/Verilog)
 6. In the field Preferred Language, Select > Verilog
 7. Click > Next to move to the page Project Summary
 8. Click > Finish in the page Project Summary
- 3) This will create a folder by the name “CE3001_lab1”. You may copy the files ‘alu.v’, ‘define.v’, ‘alutest.v’, ‘input.txt’ and ‘output.txt’ to this folder. The ‘alu.v’ file has the code for ALU and the ‘define.v’ file has the user variables in ‘alu.v’. The ‘alutest.v’ provides the input test vectors listed in ‘input.txt’ to test the code in ‘alu.v’. The output is being written to ‘output.txt’.
- 4) Once the project is open, we need to add the ‘alu.v’ file and as well as ‘define.v’ file.



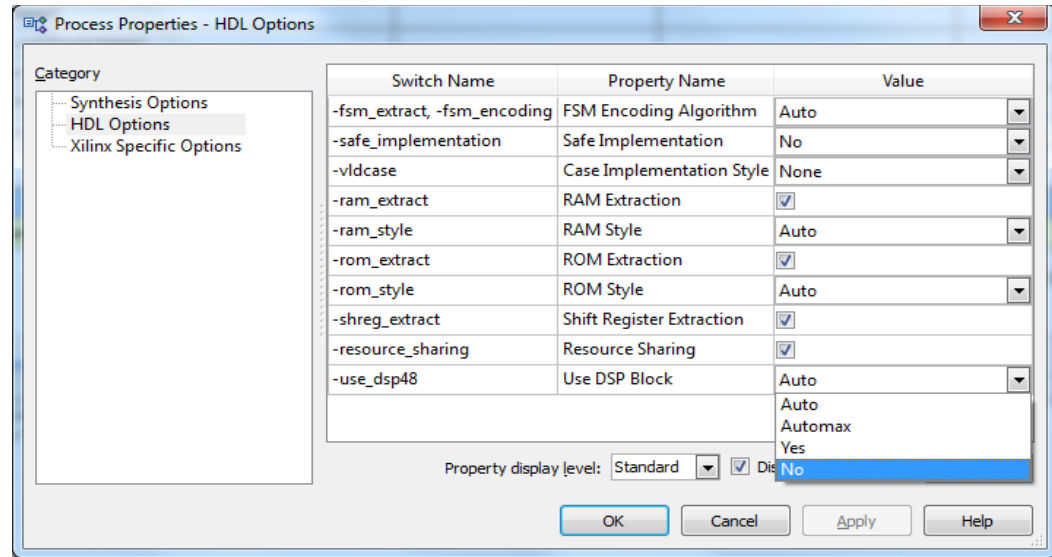
- 5) Click ‘OK’ for adding the source files ‘alu.v’ and ‘define.v’ will be added to the project “CE3001lab1”.
- 6) Go through your ‘alu.v’ code to understand the functionalities provided by the given ALU and how it is achieved.
- 7) To synthesize the code and to see the schematic diagram of the ALU, click on to the ‘synthesize-XST’ button in the ‘Design’ tab for the process ‘alu’. Right click the ‘synthesize-XST’ button and select ‘Process Properties’.



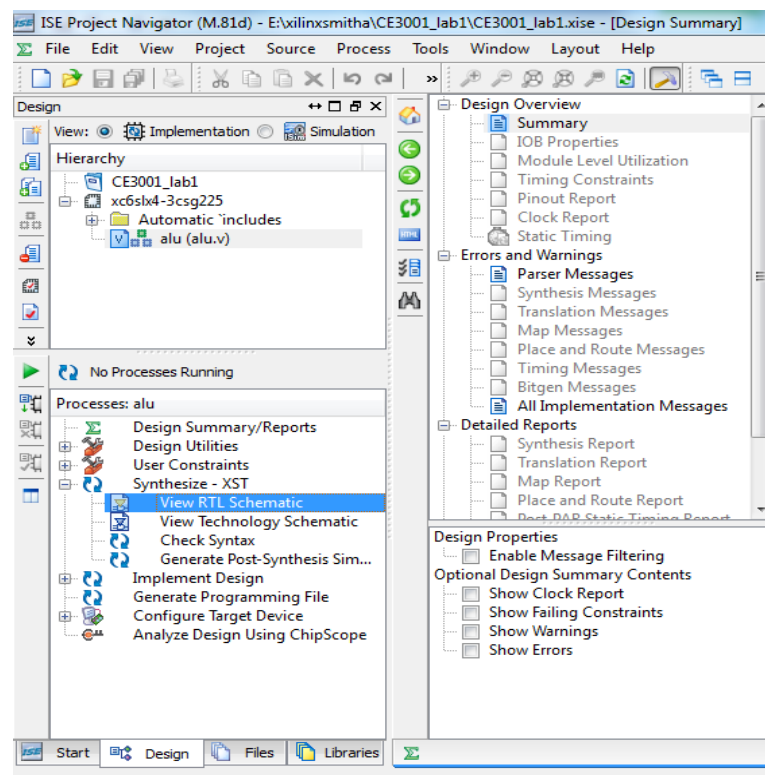
- 8) In the 'Process Properties' tab, click the 'HDL option' button. We can note that there is a 'Switch Name- use_dsp48' as the last option.



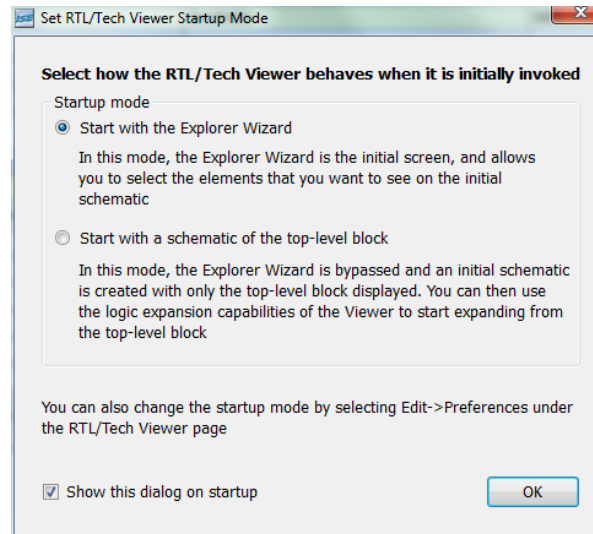
- 9) The use of DSP48 block has to be prevented and hence select 'No' and Click-> 'OK'.



10) Now, expand the 'synthesis' button and click on to 'View RTL schematic'.

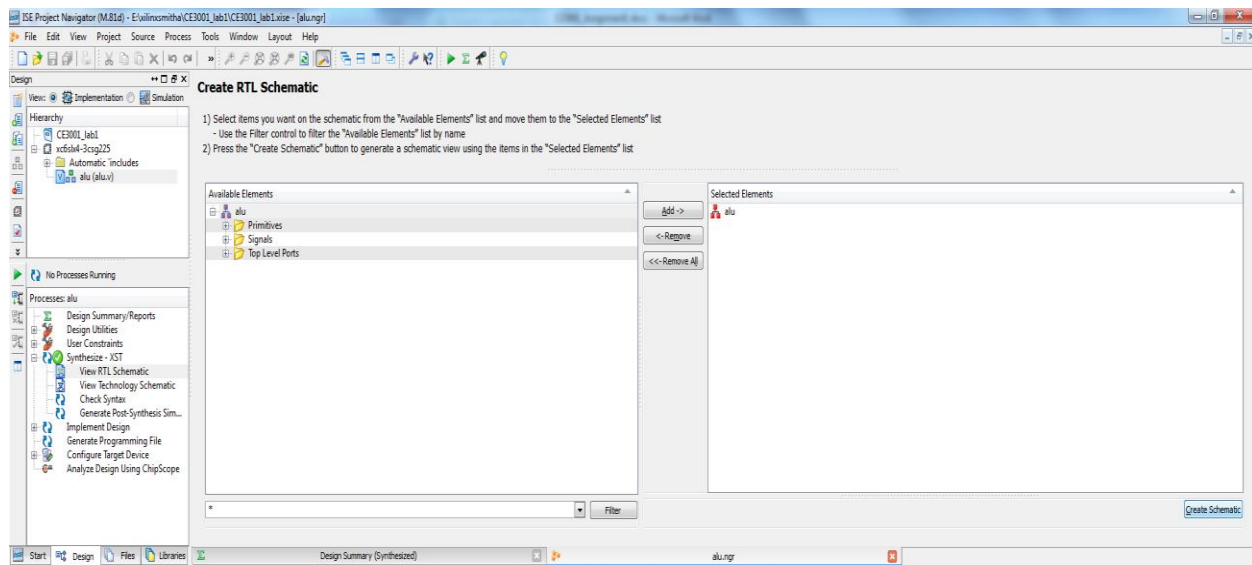


11) This process will initiate the synthesis of the code as can be seen in synthesis report (number of Look up Table (LUT) slices which is a measure of area and delay of the circuit) and also will give the RTL schematic of the code.



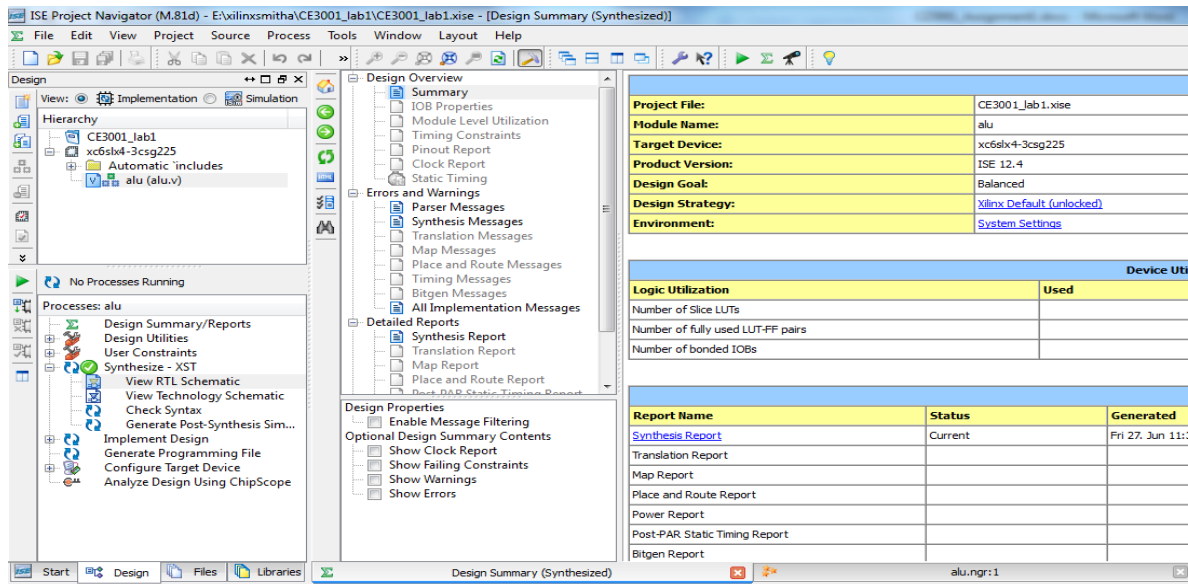
c. Click Ok. This will bring another window with the ALU module to create the schematic as shown below.

d. Click 'alu' under 'Available Elements' and "ADD" to the 'Selected Elements' and click 'Create Schematic' button on the bottom corner.

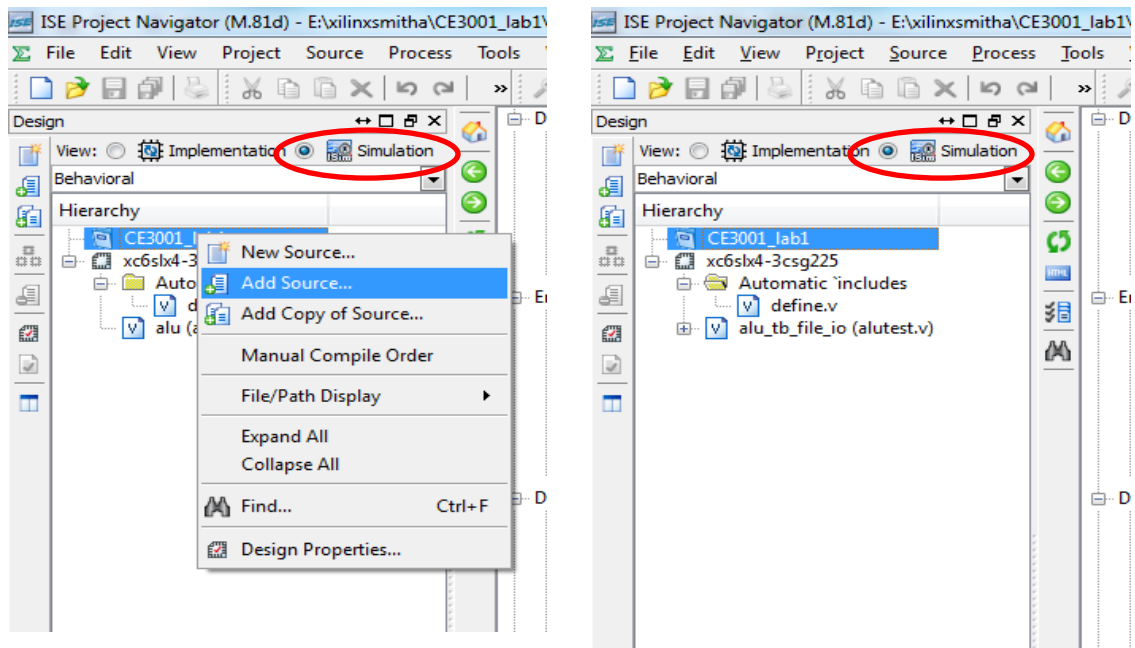


12) The schematic diagram can be seen. To go inside the main module, we need to click the main module schematic.

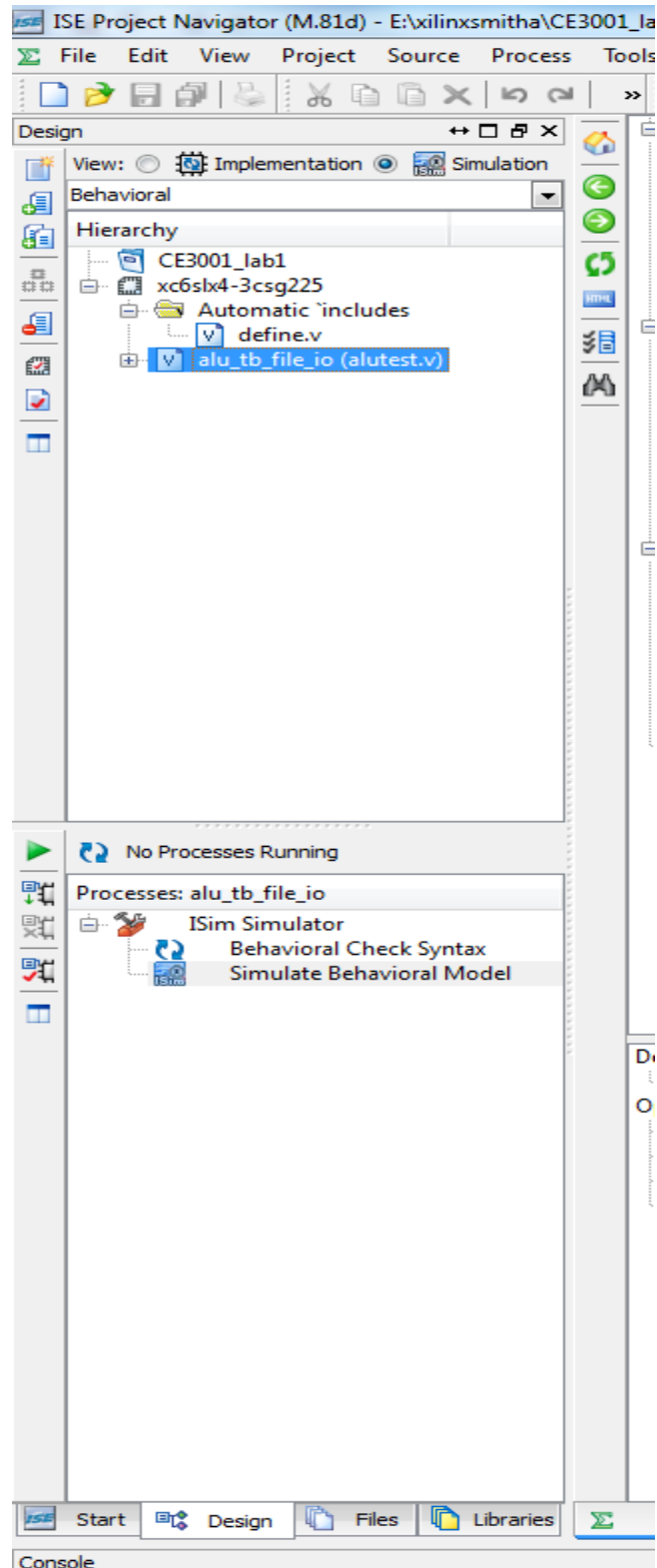
13) The synthesis result can be found by clicking the 'Synthesis report' in the 'Design Summary' tab.



- 14) To do simulation of the code given we need a test bench. Click on to the ‘Simulation’ tab in the design section and add ‘alutest.v’ to test the ALU. The ‘alutest.v’ is initialized to bit-width 4, but can be changed to different bit-widths. Note that you need to change the values in ‘input.txt’ accordingly. As the DSIZE of test bench is 4, the DSIZE in “define.v” should be 4.



- 15) Now click on ‘alutest.v’ added and you can see ‘ISIM simulator’ being selected in the bottom portion of design section. Expand ‘ISIM simulator’ and click on ‘Simulate behavioural model’.



- 16) The simulator will open, but as we had given the input as text file and has programmed the 'alutest.v' to give the output as text file, we can go and check the 'output.txt' file in the folder "CE3001_lab1". You can see the results of the inputs that you have given.

- B. The ALU code given is parametrizable. That means we can change the bit-width of the ALU by changing the value for parameter DSIZE;
 If parameter DSIZE=8, then the bit-width of the operands of ALU is 8 bit.
 If parameter DSIZE=64, then the bit-width of the operands of ALU is 64 bit.

- 1) Change the parameter DSIZE from 8, 16, 32 and 64 (one by one) and note the number of slices occupied and delay in terms of (ns). Plot the graph, area (vs) bit-width as well as delay (vs) bit-width for the ALU module.

Table 3: Slices and delay for ALU module

DSIZE	BIT-WIDTH	No of slices	Delay in ns
8	8		
16	16		
32	32		
64	64		

- 2) You can also note the schematic diagram also reflects the increased bit-width of the architecture.

EVALUATION -2

1. Plot the graph, area (vs) bit-width as well as delay (vs) bit-width for the ALU module for DSIZE =8, 16, 32 and 64.