CE/CZ3001

LAB-1

In this lab you need to find the area and time complexity of different 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 will be 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 for different bit-widths. You will experiment with the functionality of each arithmetic circuit as well as the area and time complexity of multiplier for different bit-widths.

ARITHMETIC CIRCUITS DESCRIPTION

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

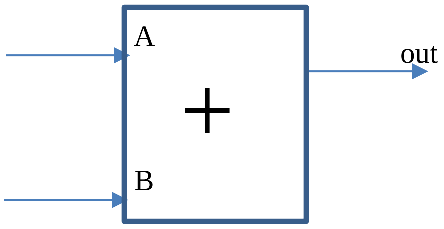


Figure-1: Adder circuit diagram.

2. **Multiplier :** The functionality of the multiplier is to multiply two inputs A, B to result in output product (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.

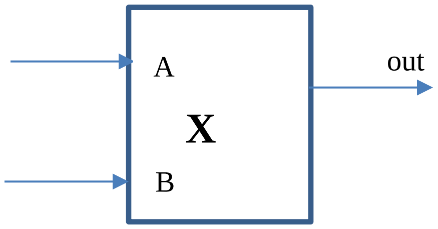


Figure-2: Multiplier circuit diagram.

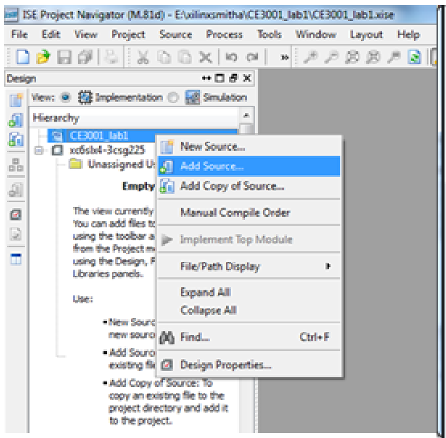
ARITHMETIC CIRCUITS SYNTHESIS, TESTING AND ANALYSIS

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-width. 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.

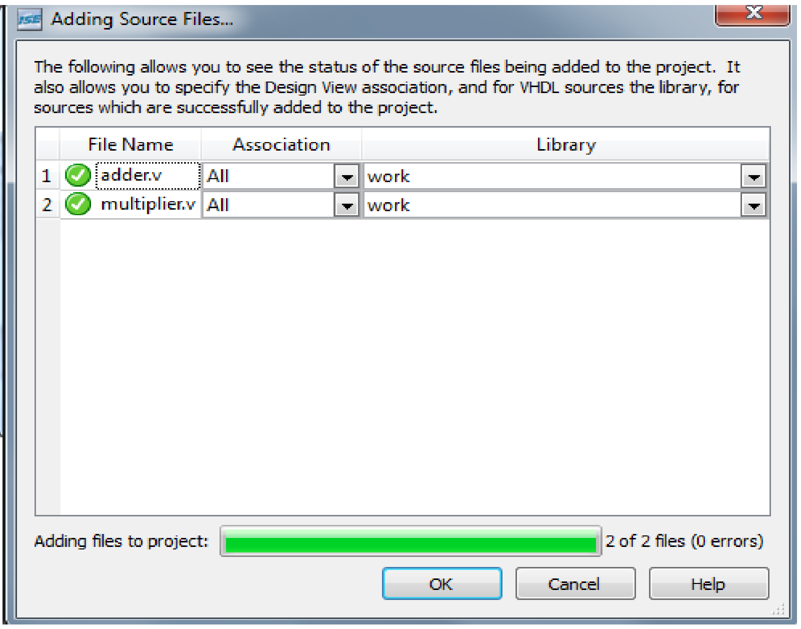
   A. Open Project and Download Source Verilog files.

   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 ‘multipler.v’ to this folder.
   4) Once the project is open, we need to ‘Add source’ by adding ‘adder_v’, and ‘multipler.v’ file.


ISE Project Navigator (M.81d) - E:\xilinxsmitha\CE3001_lab1\CE3001_lab1.xise

The view currently ... You can add files t... using the toolbar a... from the Project m... using the Design, F... Libraries panels.

Use:
- New Sourc... new sourc...
- Add Sourc... existing fil...
- Add Copy of Source: To copy an existing file to the project directory and add it to the project.

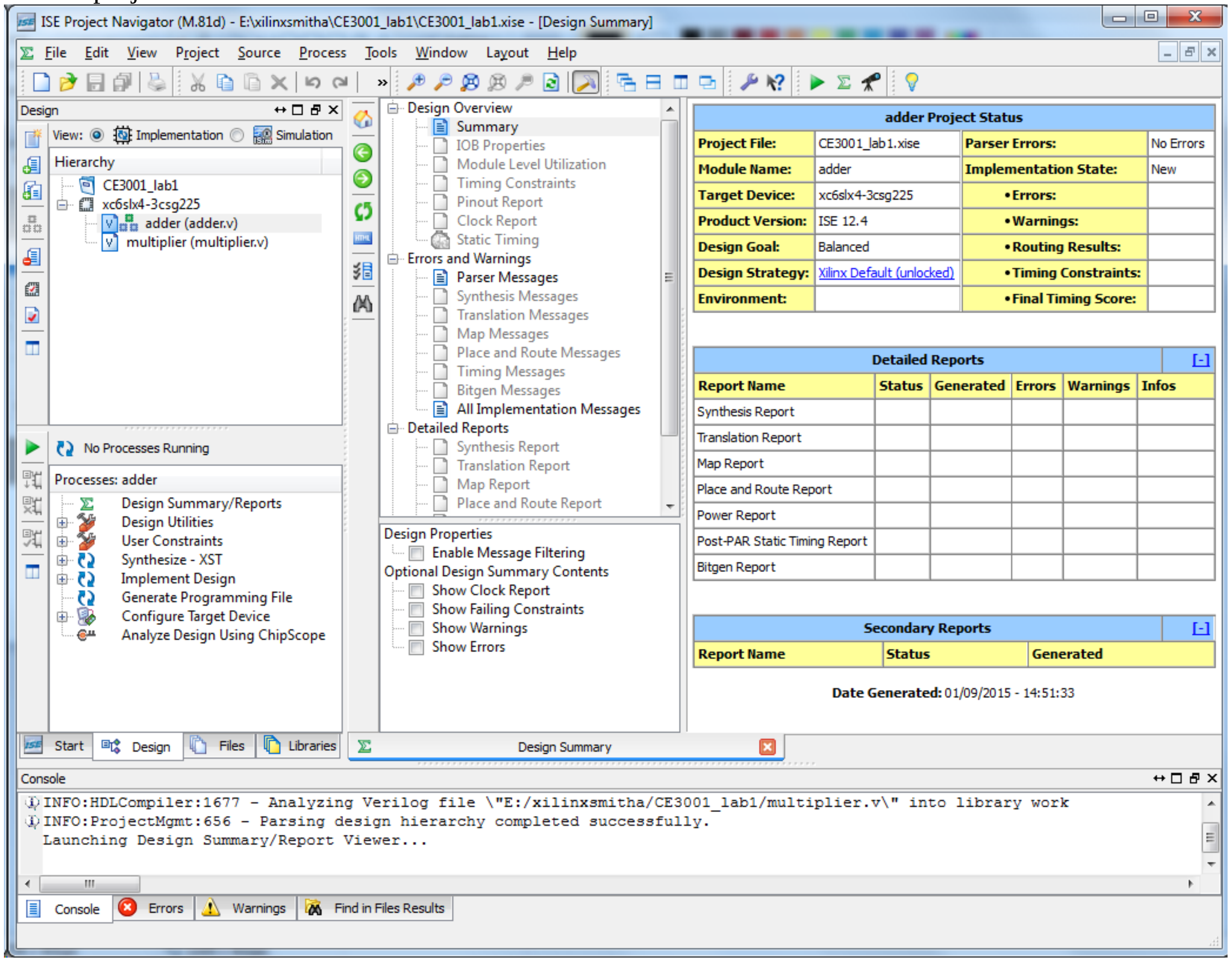

Adding Source Files...

The following allows you to see the status of the source files being added to the project. It also allows you to specify the Design View association, and for VHDL sources the library, for sources which are successfully added to the project.

	File Name	Association	Library
1	✓ adder.v	All	work
2	✓ multiplier.v	All	work

Adding files to project: 2 of 2 files (0 errors)

5) Click ‘OK’ for adding the source files ‘‘adder_v’ and ‘multipler.v’ will be added to the project “CE3001lab1”.



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

Detailed Reports [-]

Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report					
Translation Report					
Map Report					
Place and Route Report					
Power Report					
Post-PAR Static Timing Report					
Bitgen Report					

Secondary Reports [-]

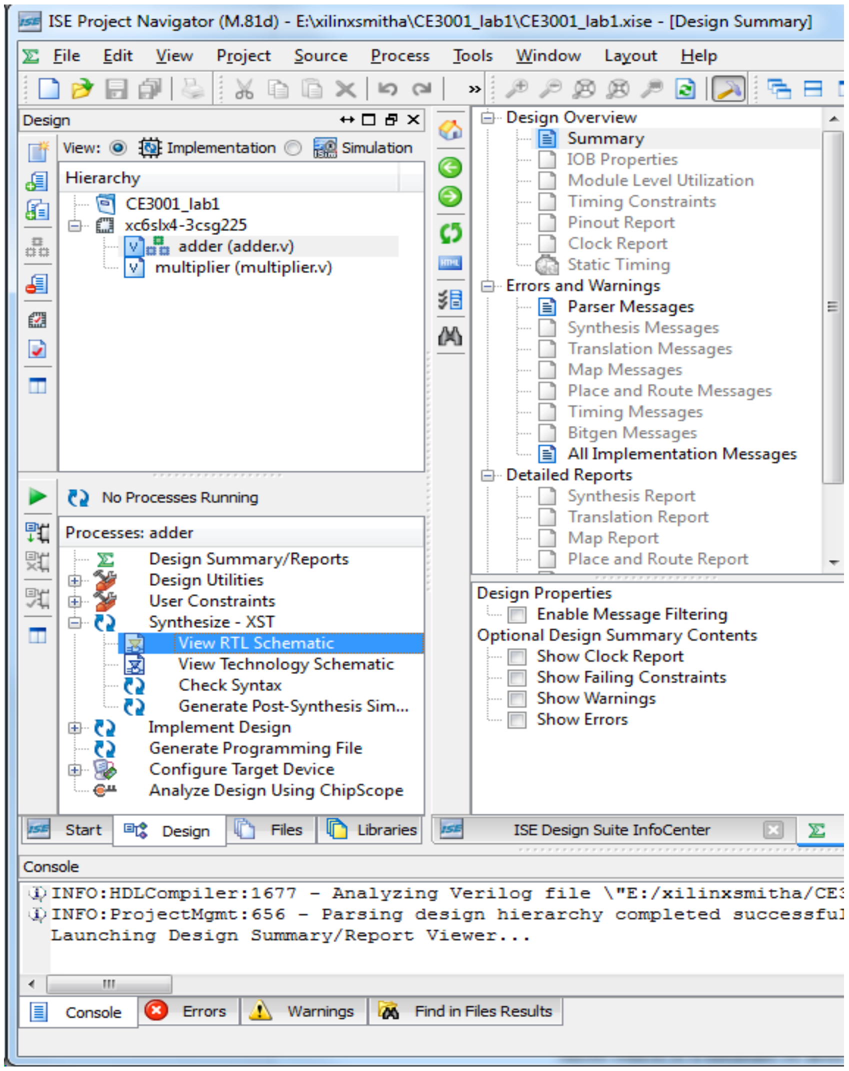
Report Name	Status	Generated

Date Generated: 01/09/2015 - 14:51:33

```
INFO:HDLCompiler:1677 - Analyzing Verilog file \"E:/xilinxsmitha/CE3001_lab1/multiplier.v\" into library work
INFO:ProjectMgmt:656 - Parsing design hierarchy completed successfully.
Launching Design Summary/Report Viewer...
```

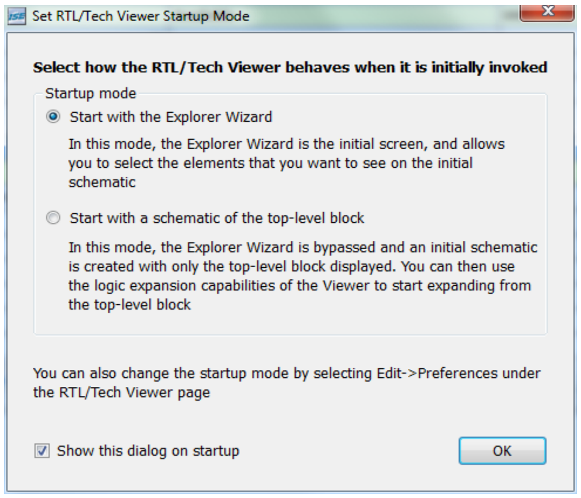
6) Double click the adder module and go through your ‘adder.v’ code to understand the functionality of the full adder. In order to synthesize ‘adder.v’ right click ‘adder’ and select ‘Set as Top Module’.

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’.



```
INFO:HDLCompiler:1677 - Analyzing Verilog file \"E:/xilinxsmitha/CE3
INFO:ProjectMgmt:656 - Parsing design hierarchy completed successful
Launching Design Summary/Report Viewer...
```

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.


Set RTL/Tech Viewer Startup Mode

Select how the RTL/Tech Viewer behaves when it is initially invoked

Startup mode

[✓ Start with the Explorer Wizard]

In this mode, the Explorer Wizard is the initial screen, and allows you to select the elements that you want to see on the initial schematic

Start with a schematic of the top-level block

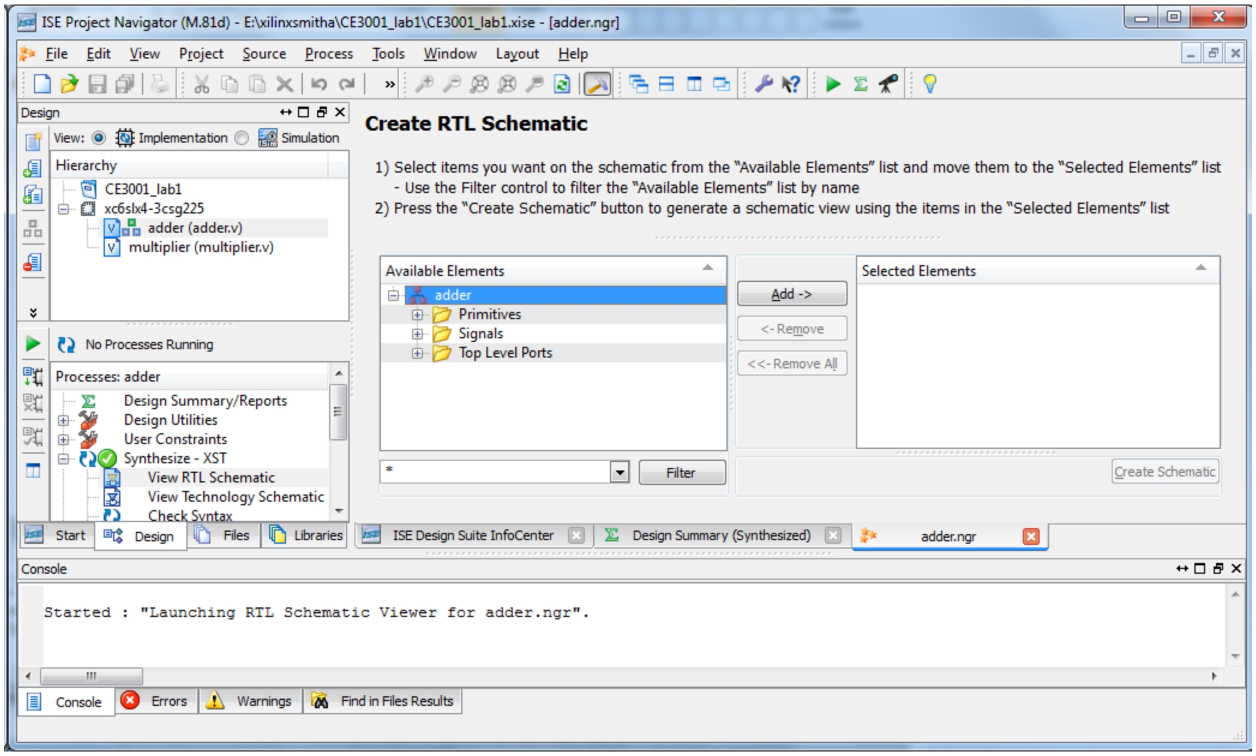
In this mode, the Explorer Wizard is bypassed and an initial schematic is created with only the top-level block displayed. You can then use the logic expansion capabilities of the Viewer to start expanding from the top-level block

You can also change the startup mode by selecting Edit->Preferences under the RTL/Tech Viewer page

[✓ Show this dialog on startup]

OK

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

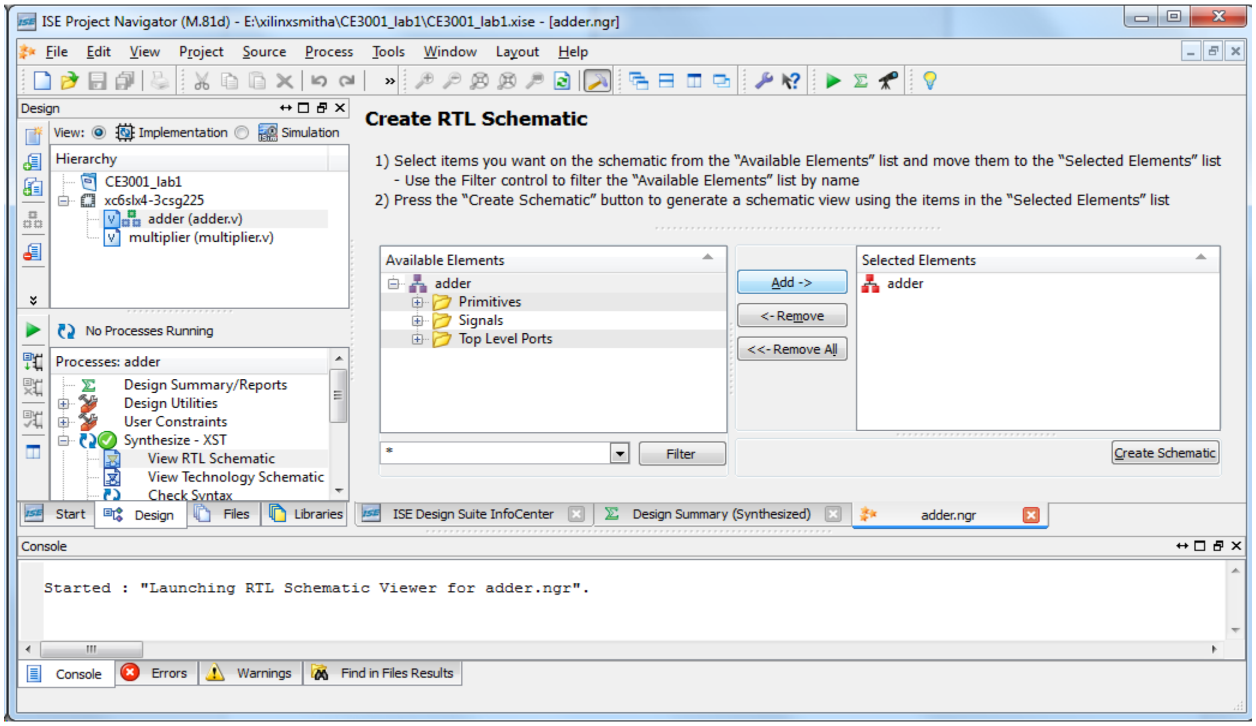

ISE Project Navigator (M.81d) - E:\xilinxsmitha\CE3001_lab1\CE3001_lab1.xise - [adder.ngr]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
CE3001_lab1
xc6slx4-3csg225
adder (adder.v)
multiplier (multiplier.v)
No Processes Running
Processes: adder
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Available Elements
adder
Primitives
Signals
Top Level Ports
Add ->
<- Remove
<<- Remove All
Selected Elements
Filter
Create Schematic
Start Design Files Libraries
ISE Design Suite InfoCenter
Design Summary (Synthesized)
adder.ngr
Console
Errors Warnings Find in Files Results

Create RTL Schematic

1) Select items you want on the schematic from the "Available Elements" list and move them to the "Selected Elements" list
 - Use the Filter control to filter the "Available Elements" list by name
2) Press the "Create Schematic" button to generate a schematic view using the items in the "Selected Elements" list

```
Started : "Launching RTL Schematic Viewer for adder.ngr".
```

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


ISE Project Navigator (M.81d) - E:\xilinxsmitha\CE3001_lab1\CE3001_lab1.xise - [adder.ngr]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
CE3001_lab1
xc6slx4-3csg225
adder (adder.v)
multiplier (multiplier.v)
No Processes Running
Processes: adder
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Available Elements
adder
Primitives
Signals
Top Level Ports
Add ->
<- Remove
<<- Remove All
Selected Elements
adder
Filter
Create Schematic
Start Design Files Libraries
ISE Design Suite InfoCenter
Design Summary (Synthesized)
adder.ngr
Console
Errors Warnings Find in Files Results

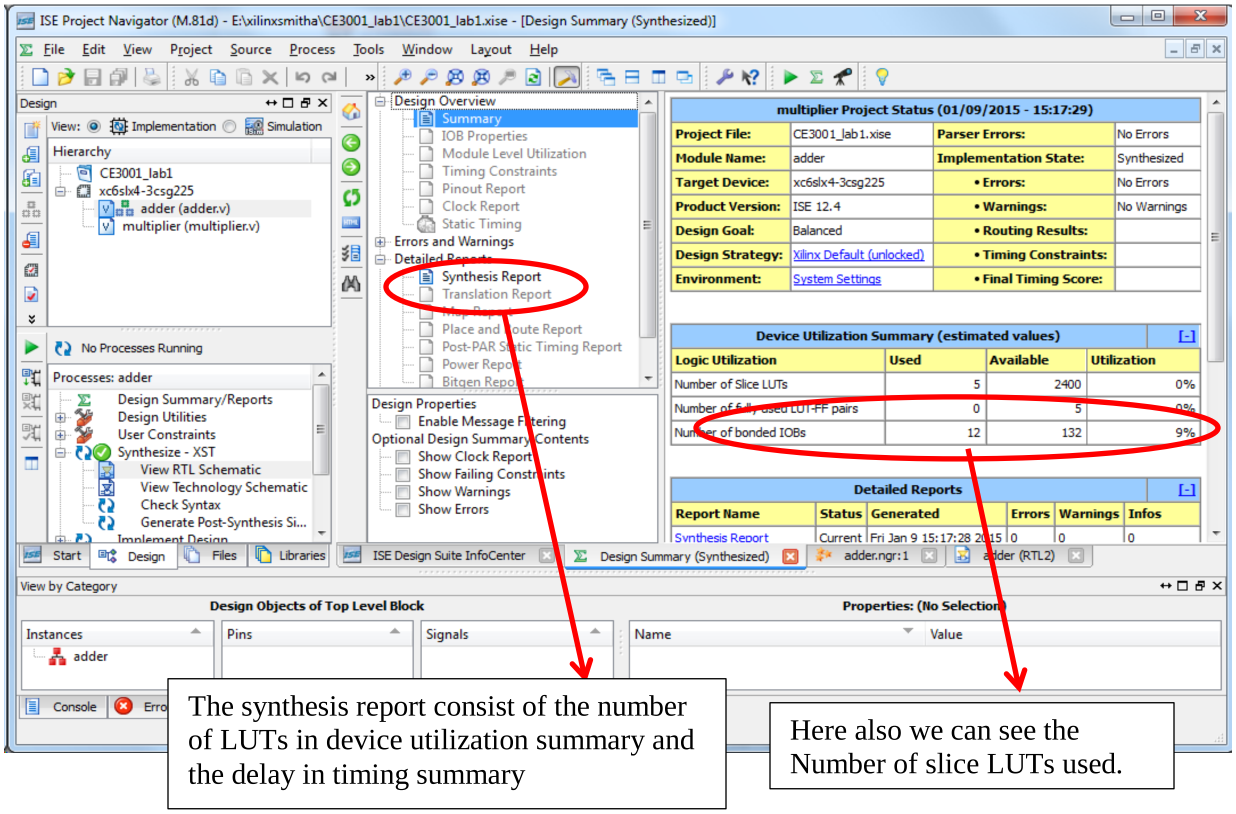
Create RTL Schematic

1) Select items you want on the schematic from the "Available Elements" list and move them to the "Selected Elements" list
 - Use the Filter control to filter the "Available Elements" list by name
2) Press the "Create Schematic" button to generate a schematic view using the items in the "Selected Elements" list

```
Started : "Launching RTL Schematic Viewer for adder.ngr".
```

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.



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%
Number of fully used LUT-FF pairs	0	5	0%
Number of bonded IOBs	12	132	9%

Detailed Reports [-]

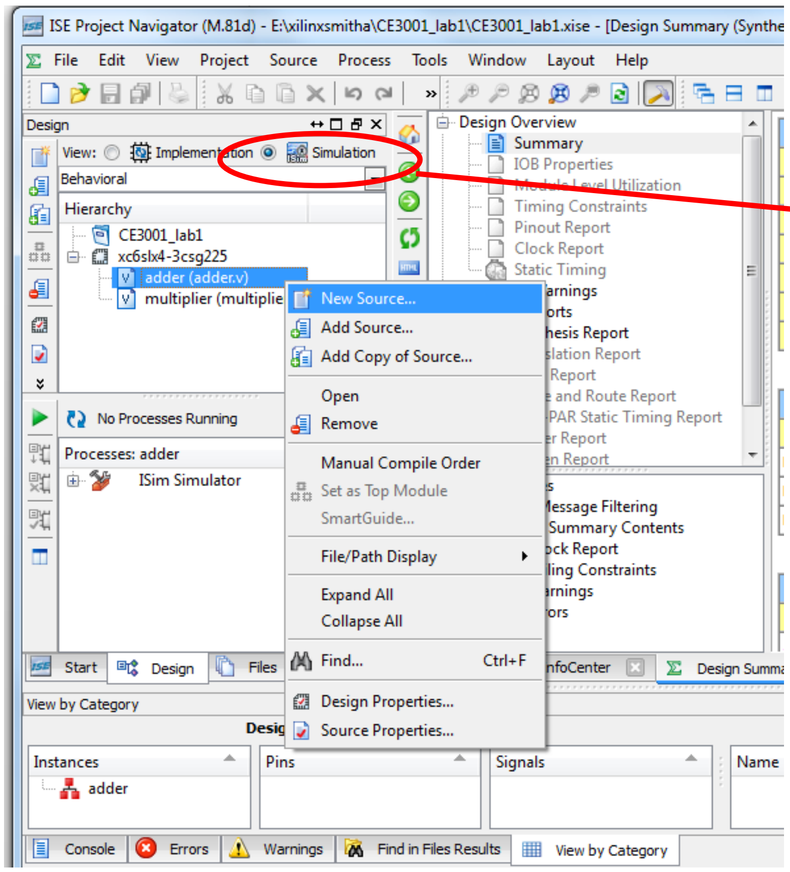
Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Fri Jan 9 15:17:28 2015	0	0	0

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

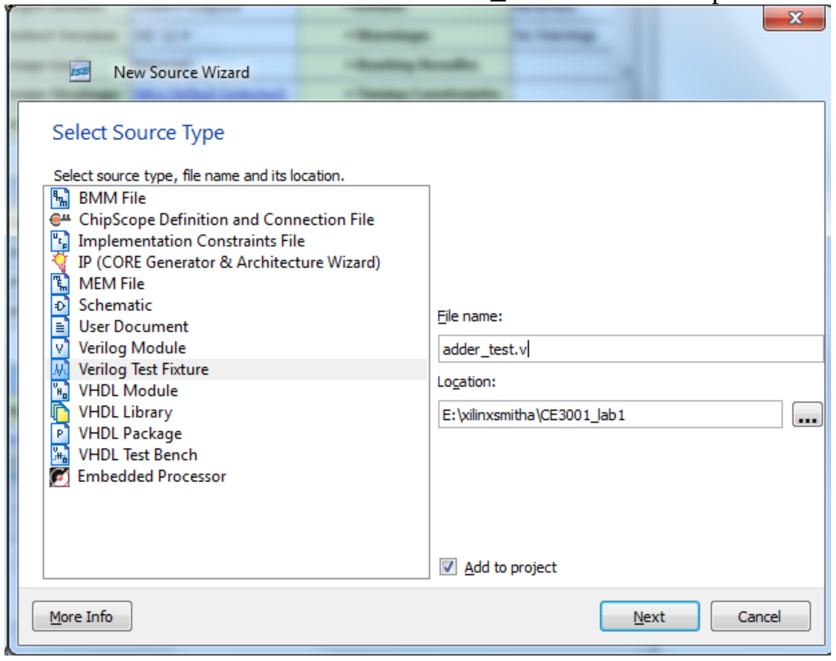
Here also we can see the Number of slice LUTs used.

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

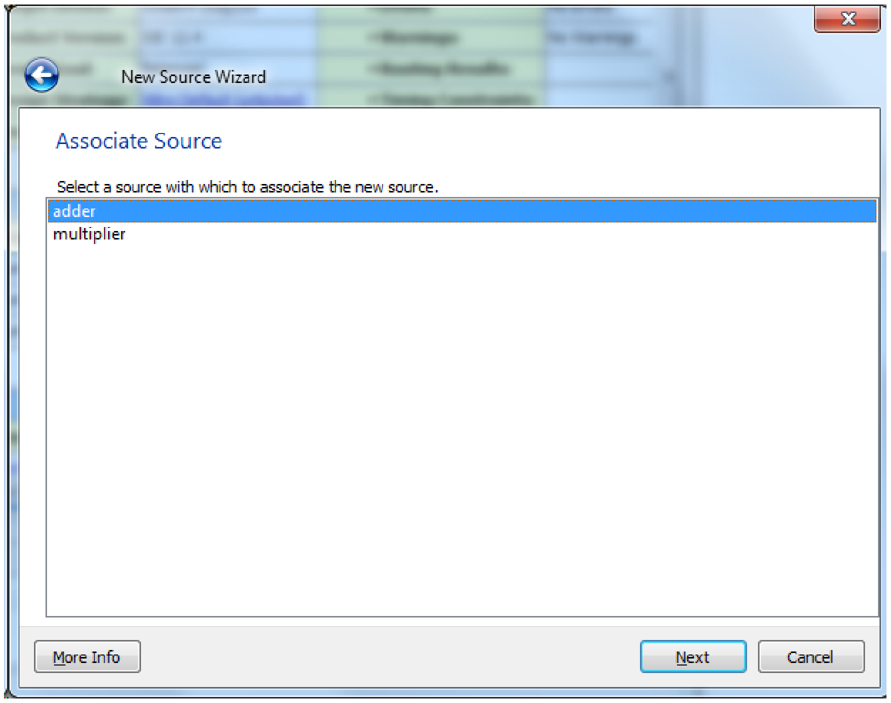
- In the Design Panel, right-click on the icon adder and select >New Source to move to the page Select Source Type



- 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



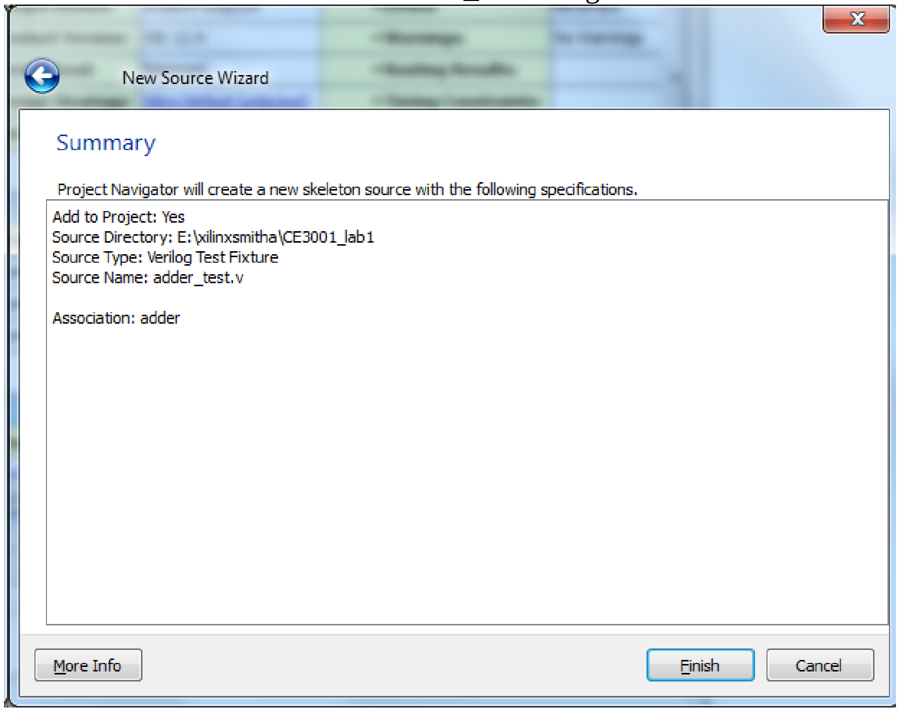
Associate Source

Select a source with which to associate the new source.

adder
multiplier

- 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.

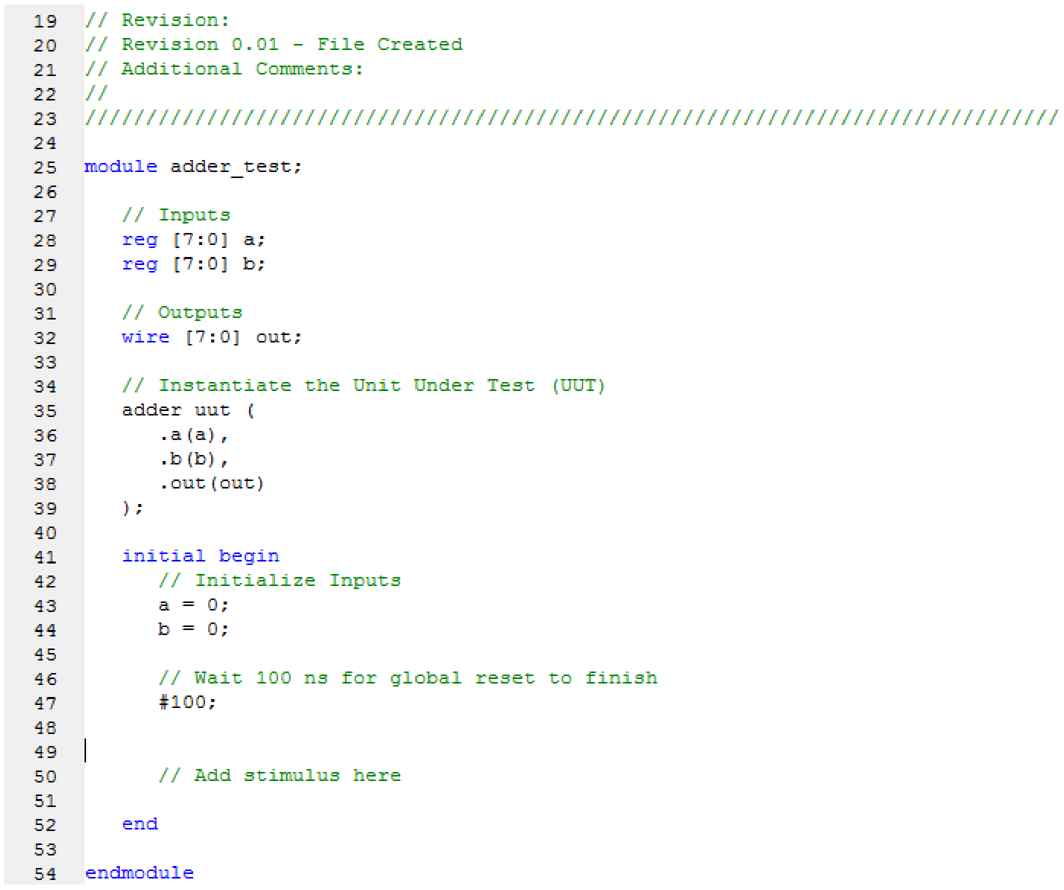


Summary

Project Navigator will create a new skeleton source with the following specifications.

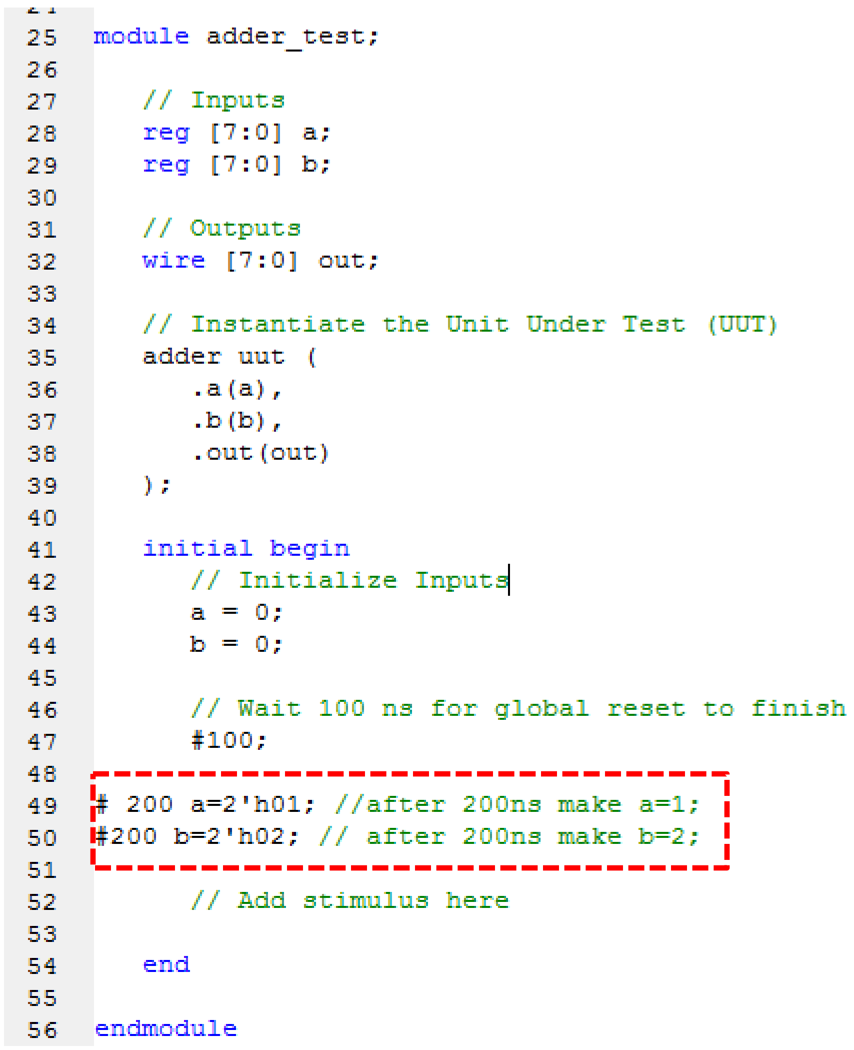
Add to Project: Yes
Source Directory: E:\xilinxsmitha\CE3001_lab1
Source Type: Verilog Test Fixture
Source Name: adder_test.v

Association: adder



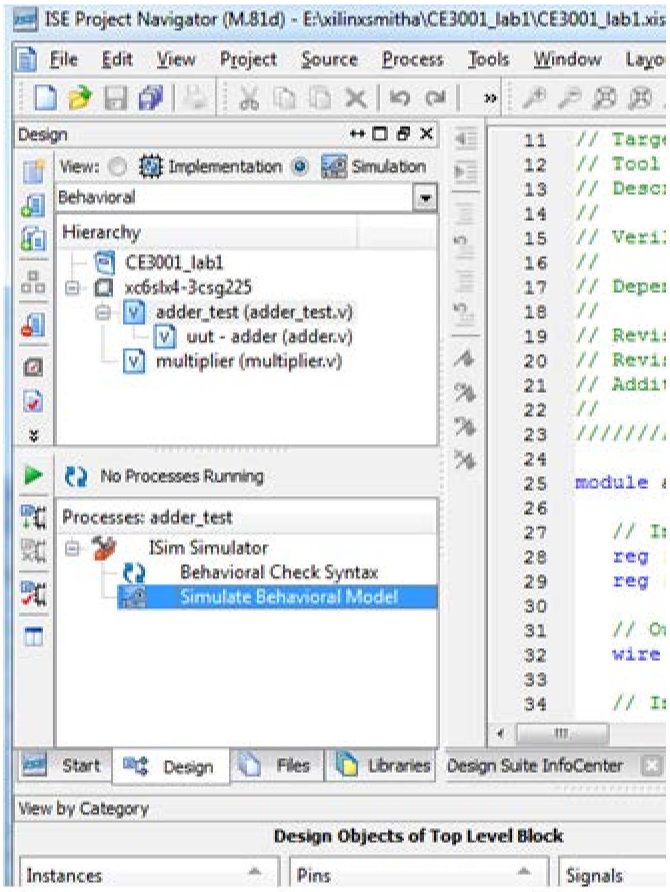
```
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////

module adder_test;

   // Inputs
   reg [7:0] a;
   reg [7:0] b;

   // Outputs
   wire [7:0] out;

   // Instantiate the Unit Under Test (UUT)
   adder uut (
      .a(a),
      .b(b),
      .out(out)
   );

   initial begin
      // Initialize Inputs
      a = 0;
      b = 0;

      // Wait 100 ns for global reset to finish
      #100;


      // Add stimulus here

   end

endmodule
```

12) Add the following inputs below the ‘//Add stimulus here’ section in the program.
    # 200 a=2’h01; //after 200ns make a=1;
    #200 b=2’h02; // after 200ns make b=1;



```
module adder_test;

   // Inputs
   reg [7:0] a;
   reg [7:0] b;

   // Outputs
   wire [7:0] out;

   // Instantiate the Unit Under Test (UUT)
   adder uut (
      .a(a),
      .b(b),
      .out(out)
   );

   initial begin
      // Initialize Inputs
      a = 0;
      b = 0;

      // Wait 100 ns for global reset to finish
      #100;

# 200 a=2'h01; //after 200ns make a=1;
#200 b=2'h02; // after 200ns make b=2;

      // Add stimulus here

   end

endmodule
```

13) Click ‘Simulate Behavioural model’ below ‘ISIM simulator’ for the processes ‘adder_test’ to generate the output waveform. Verify whether the functionality of full adder is correctly done.

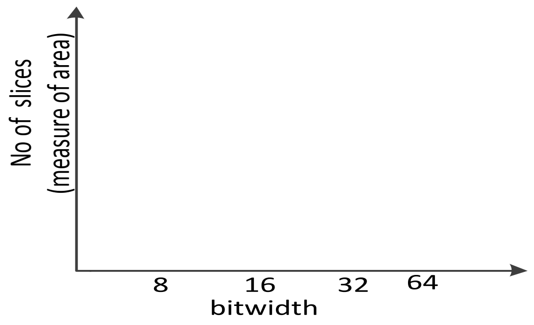

ISE Project Navigator (M.81d) - E:\xilinxsmitha\CE3001_lab1\CE3001_lab1.xis
File Edit View Project Source Process Tools Window Layo
Design
View: Implementation Simulation
Behavioral
Hierarchy
CE3001_lab1
xc6slx4-3csg225
adder_test (adder_test.v)
uut - adder (adder.v)
multiplier (multiplier.v)
No Processes Running
Processes: adder_test
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
Start
Design
Files
Libraries
Design Suite InfoCenter
View by Category
Design Objects of Top Level Block
Instances
Pins
Signals

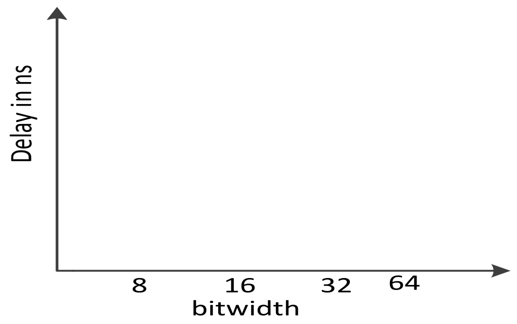
B: The **<u>adder</u>** code is parametrizable. That means you can change the bit-width of the adder by changing the value for parameter **DSIZE**. If parameter **DSIZE** is set to 8, then it's an eight bit adder. If the parameter **DSIZE** =64, then the bit-width of the operands in adder is 64 bit.

1) Change the parameter **DSIZE** from 8, 16, 32 and 64 and note the number of slices used and delay (in ns). 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		





Area in slices (vs) bit-width

delay (vs) bit-width

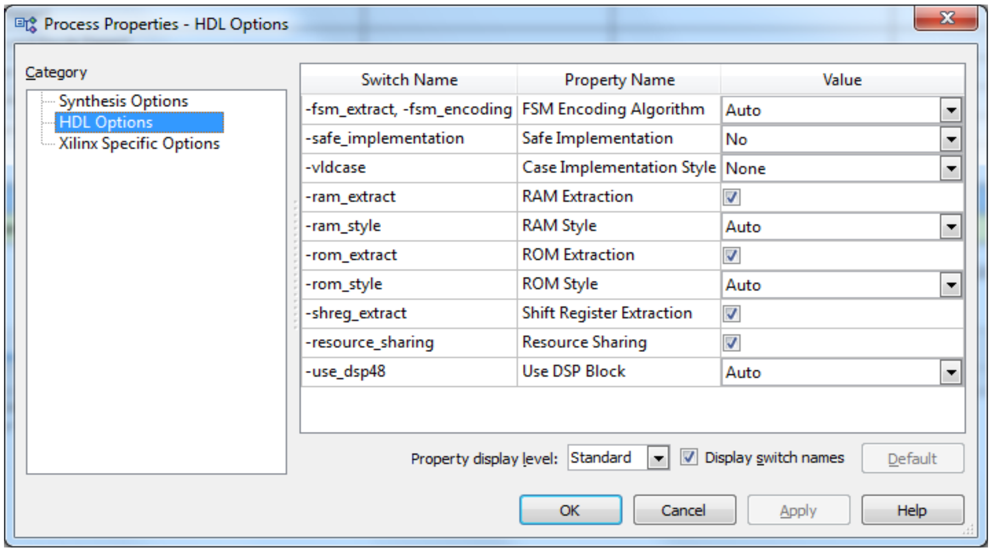
To synthesize and to see its circuit diagram make it as the topmodule and **repeat from steps 6-13.** You may have to add respective input values to the testbench program for adder in-order to verify the same. Do note that the bit-width of the inputs is same as the bit-width parameter that you have set 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 which reflects the increased bit-width of the architecture.

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 is 64 bit.

1) Change the parameter **DSIZE** from 8, 16, 32 and 64 and note the number of slices occupied and delay in terms of (ns). 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 widths of the inputs is 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 architecture.

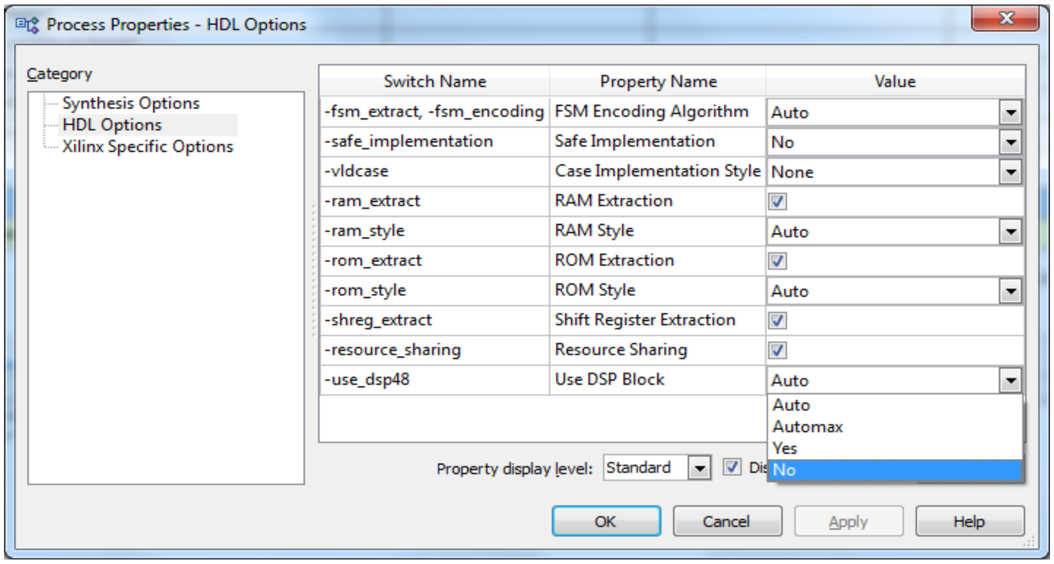
3) <u>Before doing step 7(the synthesis), click the 'HDL option' button in the 'Process Properties' tab.</u> Note that there is a 'Switch Name- use_dsp48' as the last option.


Process Properties - HDL Options

Switch Name	Property Name	Value
-fsm_extract, -fsm_encoding	FSM Encoding Algorithm	Auto
-safe_implementation	Safe Implementation	No
-vldcase	Case Implementation Style	None
-ram_extract	RAM Extraction	✓
-ram_style	RAM Style	Auto
-rom_extract	ROM Extraction	✓
-rom_style	ROM Style	Auto
-shreg_extract	Shift Register Extraction	✓
-resource_sharing	Resource Sharing	✓
-use_dsp48	Use DSP Block	Auto

Property display level: Standard ✓ Display switch names

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


Process Properties - HDL Options

Switch Name	Property Name	Value
-fsm_extract, -fsm_encoding	FSM Encoding Algorithm	Auto
-safe_implementation	Safe Implementation	No
-vldcase	Case Implementation Style	None
-ram_extract	RAM Extraction	✓
-ram_style	RAM Style	Auto
-rom_extract	ROM Extraction	✓
-rom_style	ROM Style	Auto
-shreg_extract	Shift Register Extraction	✓
-resource_sharing	Resource Sharing	✓
-use_dsp48	Use DSP Block	Auto

Property display level: Standard ✓ Dis

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		

Area in slices (vs) bit-width

delay (vs) bit-width

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 Lab, we consider a simple ALU that performs the computation for eight arithmetic and logical operation. The eight operations are: ADD, SUB, AND, XOR, SLL, SRL, COM, and MUL 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 format
SUB	A-B	Subtraction	Subtraction of A and B, where both A and B are in 2's complement format
AND	A&B	Logical AND	Bit-wise AND of A, B
XOR	A^B	Logical XOR	Bit-wise XOR of A, B
SLL	A<<B	Shift left logical	Shift left operand A shift to left by B
SRL	A>>B	Shift right logical	Shift right operand A shift to right by B
COM	A << = B	Comparison	If A<=B then it results "true" (output 1) else "false (output=0) A and B are in 2's complement format
MUL	A*B	Multiplication	Multiplication of A and B. A and B are in 2's complement

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 8 possible instructions which the ALU can perform. If an ALU has to perform more than 8 (but not more than 16) operations then it would need 4 bit control. The information is also listed in Table 2. The encoding of the 8 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	Selects which operation to be performed (8 operations)
Out	Output	Output of the operation

Table 3 - ALU operation encoding

Operation	**'op' value**
AND	000
SUB	001
AND	010
XOR	011
SLL	100
SRL	101
COM	110
MUL	111

ALU Implementation, Testing and Analysis

For this assignment,

A. You will be given the Verilog code as well as the test bench for ALU. You have to test whether the ALU gives correct desired results.

B. You need to set the input bit-width to 8, 16, 32 and 64 and find out the number of slices used and the maximum combinational path delay. You need to plot area (vs) bit-width as well as delay (vs) bit-width for the ALU module. In FPGA you can find area directly so instead of area you can take number of slices, which would be proportional to the area.

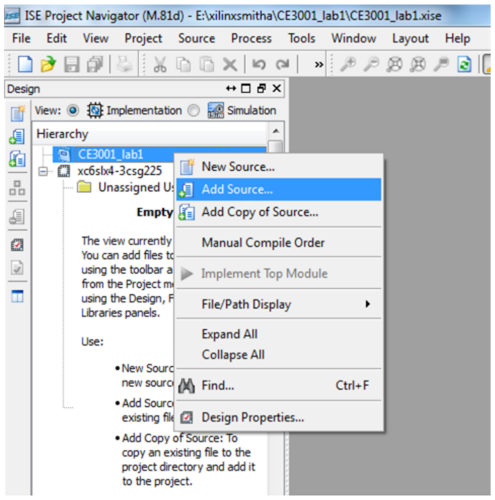
Details

A. The ALU code given 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.

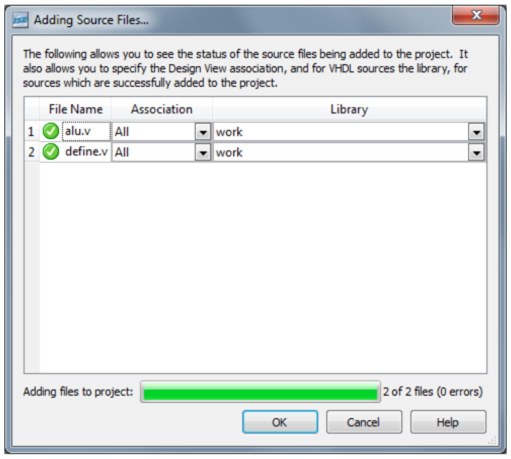


ISE Project Navigator (M.81d) - E:\xilinxsmitha\CE3001_lab1\CE3001_lab1.xise

The view currently
You can add files to
using the toolbar a
from the Project m
using the Design, F
Libraries panels.

Use:

• New Sourc
new sourc
• Add Sourc
existing file
• Add Copy of Source: To copy an existing file to the project directory and add it to the project.



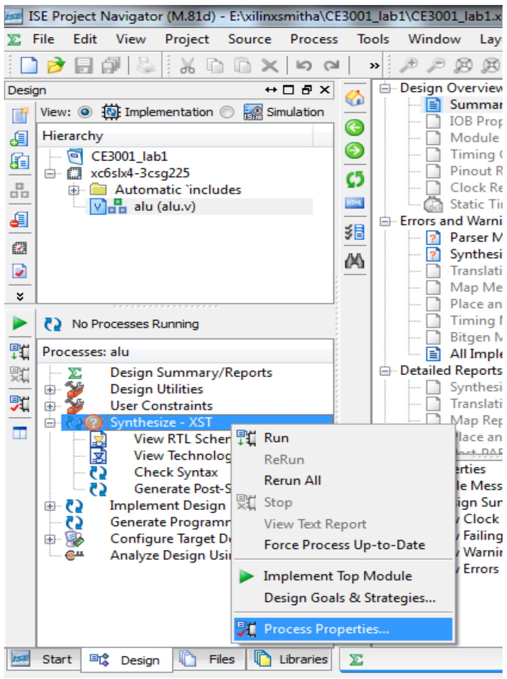
Adding Source Files...

The following allows you to see the status of the source files being added to the project. It also allows you to specify the Design View association, and for VHDL sources the library, for sources which are successfully added to the project.

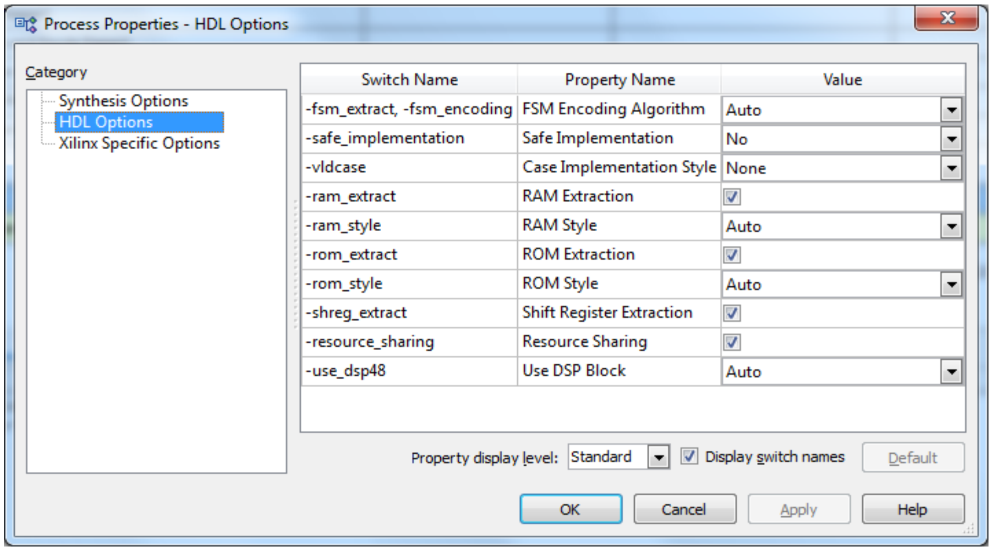
	File Name	Association	Library
1	✓ alu.v	All	work
2	✓ define.v	All	work

Adding files to project: 2 of 2 files (0 errors)

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’.



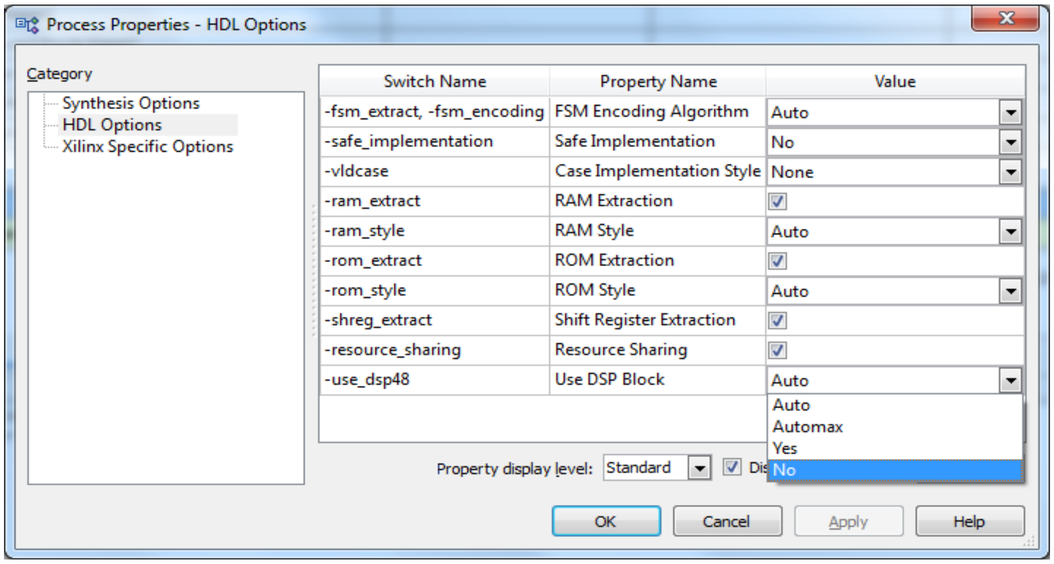
8) In the ‘Process Properties’ tab, click the ‘HDL option’ button. We can note that there is a ‘Switch Name- use_dsp48’ as the last option.



Switch Name	Property Name	Value
-fsm_extract, -fsm_encoding	FSM Encoding Algorithm	Auto
-safe_implementation	Safe Implementation	No
-vldcase	Case Implementation Style	None
-ram_extract	RAM Extraction	✓
-ram_style	RAM Style	Auto
-rom_extract	ROM Extraction	✓
-rom_style	ROM Style	Auto
-shreg_extract	Shift Register Extraction	✓
-resource_sharing	Resource Sharing	✓
-use_dsp48	Use DSP Block	Auto

Property display level: Standard ✓ Display switch names

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

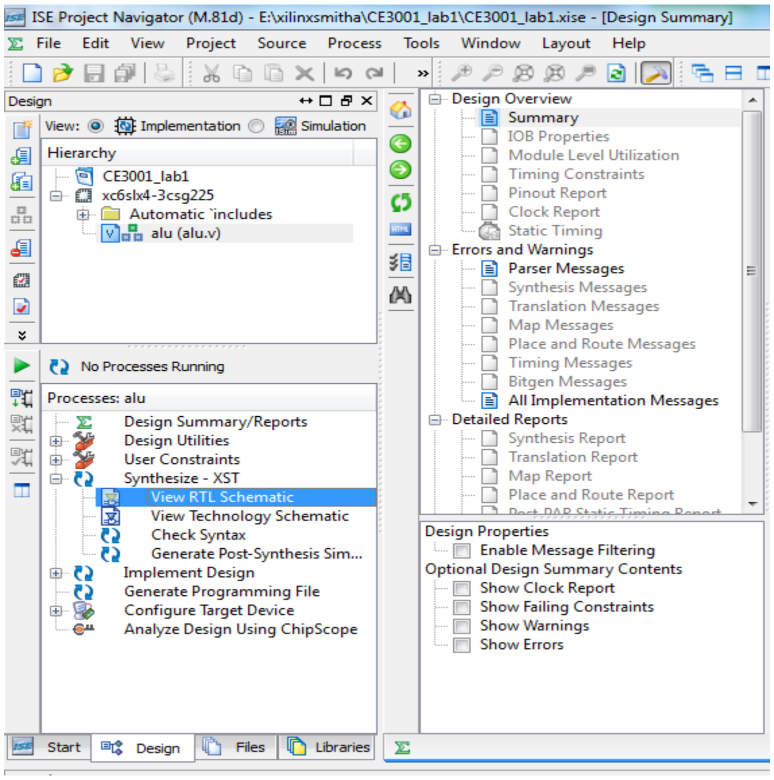


Process Properties - HDL Options

Category: Synthesis Options, HDL Options, Xilinx Specific Options

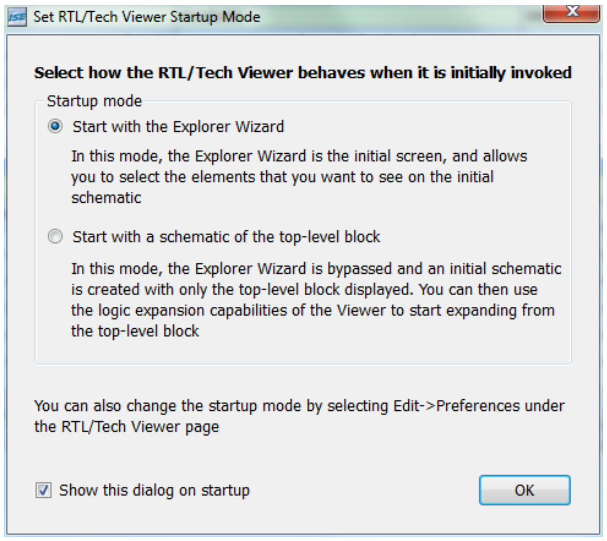
Switch Name	Property Name	Value
-fsm_extract, -fsm_encoding	FSM Encoding Algorithm	Auto
-safe_implementation	Safe Implementation	No
-vldcase	Case Implementation Style	None
-ram_extract	RAM Extraction	✓
-ram_style	RAM Style	Auto
-rom_extract	ROM Extraction	✓
-rom_style	ROM Style	Auto
-shreg_extract	Shift Register Extraction	✓
-resource_sharing	Resource Sharing	✓
-use_dsp48	Use DSP Block	Auto

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



ISE Project Navigator (M.81d) - E:\xilinxsmitha\CE3001_lab1\CE3001_lab1.xise - [Design Summary]

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.



Set RTL/Tech Viewer Startup Mode

Select how the RTL/Tech Viewer behaves when it is initially invoked

Startup mode

(●) Start with the Explorer Wizard

In this mode, the Explorer Wizard is the initial screen, and allows you to select the elements that you want to see on the initial schematic

( ) Start with a schematic of the top-level block

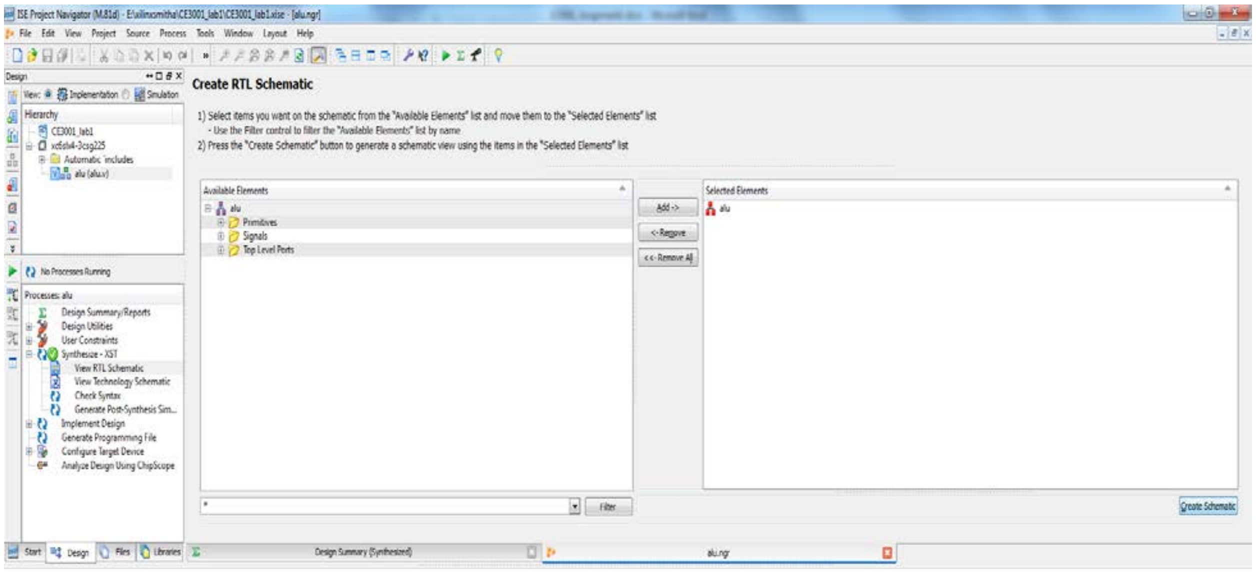
In this mode, the Explorer Wizard is bypassed and an initial schematic is created with only the top-level block displayed. You can then use the logic expansion capabilities of the Viewer to start expanding from the top-level block

You can also change the startup mode by selecting Edit->Preferences under the RTL/Tech Viewer page

[✓] Show this dialog on startup

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.

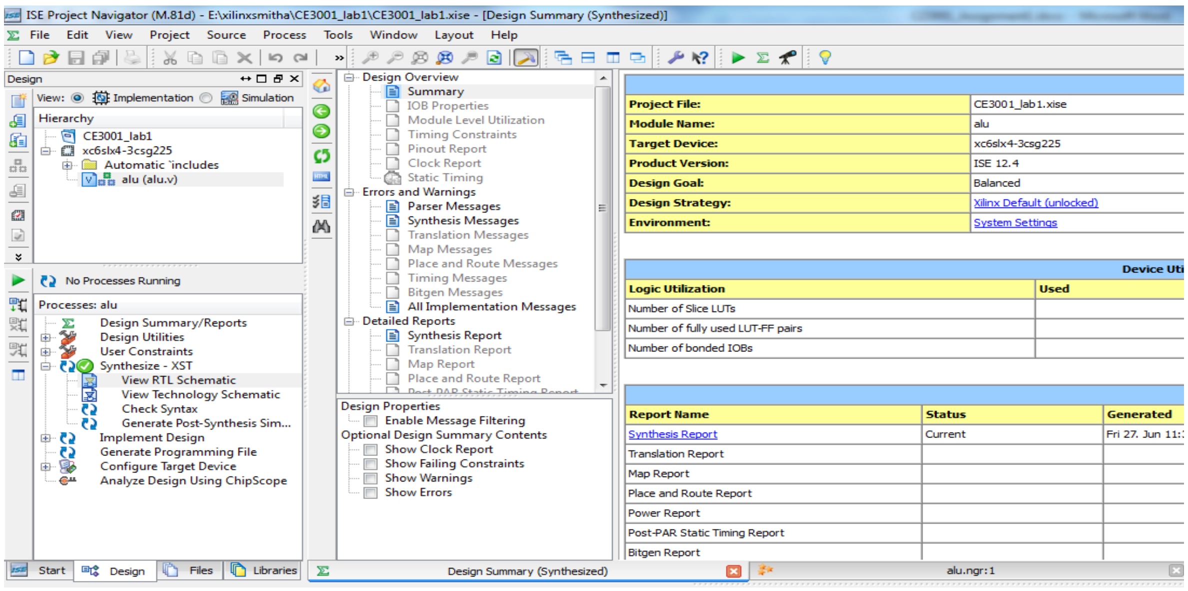


Create RTL Schematic

1) Select items you want on the schematic from the "Available Elements" list and move them to the "Selected Elements" list
 - Use the Filter control to filter the "Available Elements" list by name
2) Press the "Create Schematic" button to generate a schematic view using the items in the "Selected Elements" list

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.



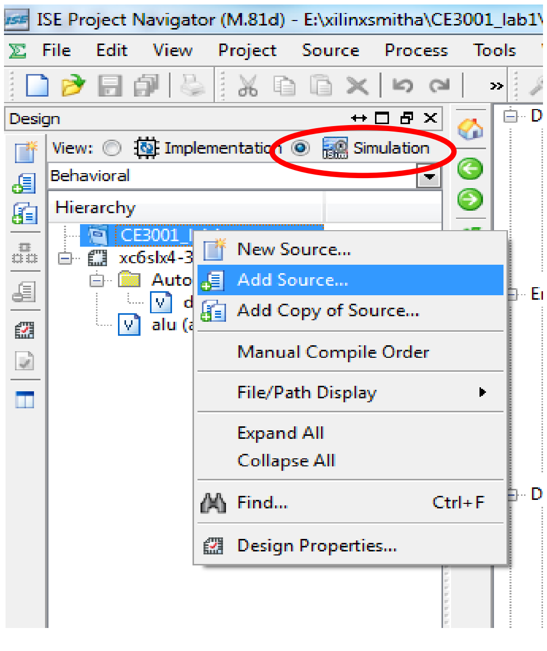
Project File:	CE3001_lab1.xise
Module Name:	alu
Target Device:	xc6slx4-3csg225
Product Version:	ISE 12.4
Design Goal:	Balanced
Design Strategy:	Xilinx Default (unlocked)
Environment:	System Settings

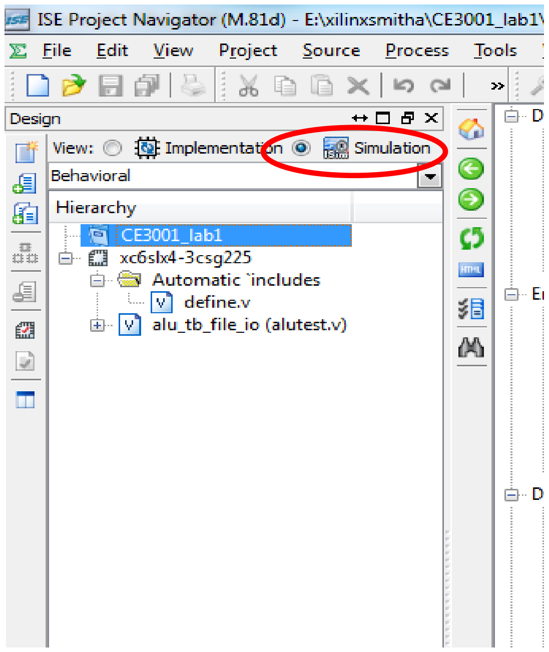
Device Uti

Logic Utilization	Used
Number of Slice LUTs	
Number of fully used LUT-FF pairs	
Number of bonded IOBs	

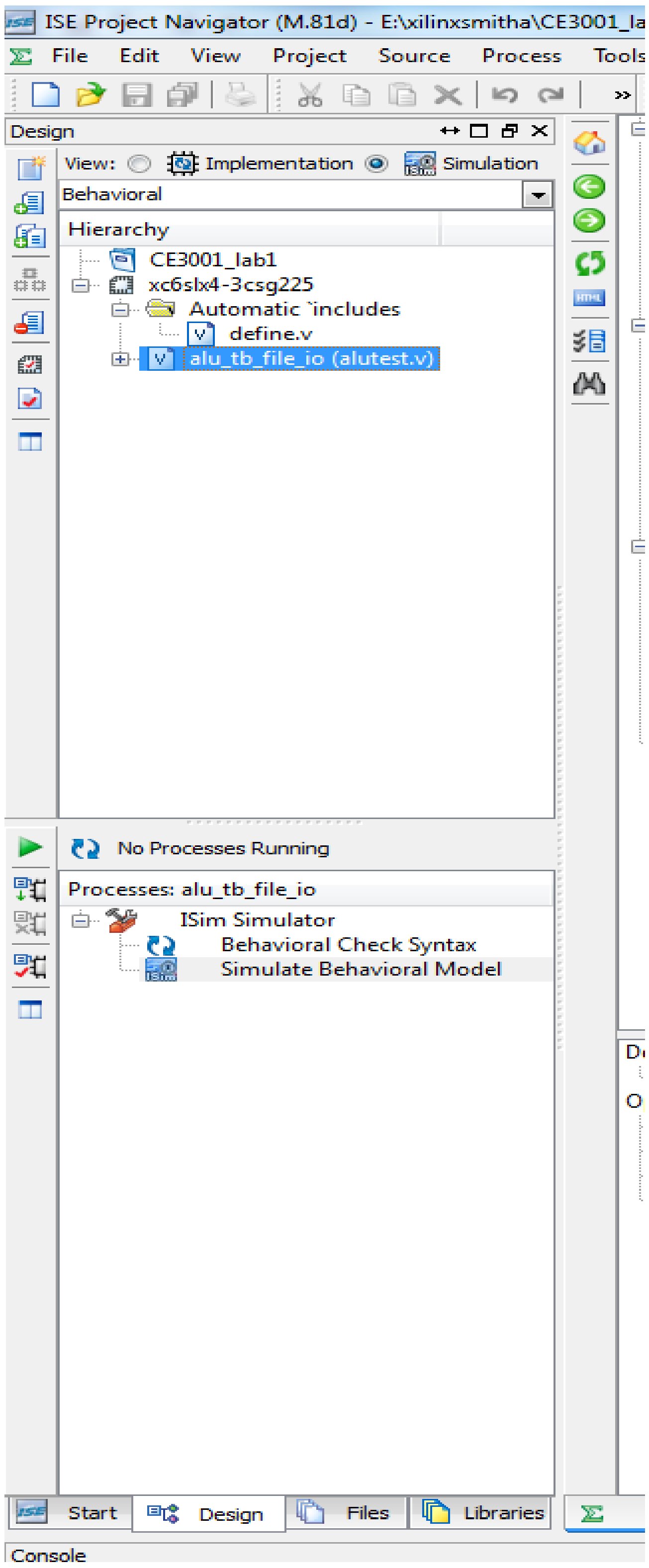
Report Name	Status	Generated
Synthesis Report	Current	Fri 27. Jun 11:
Translation Report		
Map Report		
Place and Route Report		
Power Report		
Post-PAR Static Timing Report		
Bitgen Report		

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.




ISE Project Navigator (M.81d) - E:\xilinxsmitha\CE3001_lab1\
File Edit View Project Source Process Tools
Design
View: Implementation Simulation
Behavioral
Hierarchy
CE3001_lab1
xc6slx4-3csg225
Automatic `includes
define.v
alu_tb_file_io (alutest.v)

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’.


ISE Project Navigator (M.81d) - E:\xilinxsmitha\CE3001_la

CE3001_lab1
xc6slx4-3csg225
Automatic `includes
define.v
alu_tb_file_io (alutest.v)

No Processes Running

Processes: alu_tb_file_io
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral 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 in 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 in ALU is 8 bit.
If parameter DSIZE=64, then the bit-width of the operands in ALU is 64 bit.

1) Change the parameter DSIZE from 8, 16, 32 and 64 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		

Area in slices (vs) bit-width

Delay (vs) bit-width

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.