

Project Report: Design and Implementation of Digital Circuits on FPGA using Verilog HDL

Submitted by
VADNAM SWAYAM PRAKASH (ECE/64/13)

AKASH GUPTA (ECE/69/13)

SUJEET KUMAR (ECE/61/13)

(DEPARTMENT OF E&C ENGINEERING, NIT SRINAGAR - J&K)

UNDER GUIDANCE OF
Dr. MANISHA PATTANAİK
(ASSOCIATE PROFESSOR)



विश्वजीवनामृतं ज्ञानम्

**ABV-Indian Institute of Information Technology & Management
Gwalior-474015 (MADHYA PRADESH)
January-February 2016**



ABV- INDIAN INSTITUTE OF INFORMATION TECHNOLOGY
& MANAGEMENT, GWALIOR
MADHYA PRADESH, INDIA- 474015

CERTIFICATE

This is to certify that the report entitled “**Design and Implementation of Digital Circuitry on FPGA using Verilog HDL**”, submitted by **Vadnam Swayam Prakash (Enroll. No. ECE/64/13)**, **Akash Gupta (Enroll. No. ECE/69/13)** and **Sujeet Kumar (Enroll. No. ECE/61/13)** as a part of B.Tech winter internship in partial fulfilment of the requirements for the award of the **Bachelor of Technology in Electronics and Communication Engineering, National Institute of Technology, Srinagar** is an authentic record of our own work carried out at ABV-Indian Institute of Information Technology and Management, Gwalior during a period from 1st January 2016 to 2nd February 2016.

The matter presented in this report has not been submitted by us for the award of any other degree elsewhere.

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Date:

Dr. Manisha Pattanaik
(Associate Professor)
ABV-IITM, Gwalior
Madhya Pradesh

ACKNOWLEDGEMENTS

We would like to articulate our deep gratitude to **Dr. MANISHA PATTANAIK** (Associate Professor), ABV-IIITM, Gwalior for giving us this chance and providing us with constant support and invaluable guidance throughout the duration of the winter internship program. We would also like to convey our sincerest gratitude and indebtedness to **Mr. Singaluri Vijaysairam** and **Mr. Vikash Choudhary** as our mentors at ABV-IIITM, Gwalior who showed their great efforts and guidance at required times without which it would have been very difficult to carry out the project work. Moreover, an assemblage of this nature could never have been attempted with our reference to the works of others whose details are mentioned in the references section at the last. We acknowledge our indebtedness to all of them. Furthermore, we would like to take the name of our parents and God who directly or indirectly encouraged and motivated us during this dissertation.

ABSTRACT

XILINX Tools is a suite of software tools used for the design of digital circuits implemented using Xilinx Field Programmable Gate Array (FPGA) or Complex Programmable Logic Device (CPLD). The CAD tools enable to design combinational and sequential circuits starting with Verilog HDL design specifications. Verilog HDL is one of the two most common Hardware Description Languages (HDL) used by Integrated Circuit(IC) designers. HDLs allow the design to be simulated earlier in the design cycle in order to correct errors or experiment with different architectures. Designs described in HDL are technology-independent, easy to design and debug, and are usually more readable than schematics, particularly for large circuits.

This report involves basic knowledge of FPGA and XILINX ISE Design Suite Software. For Synthesis and Verification of circuits, the FPGA kit used is XILINX SPARTAN-6 XC6SLX45 with special features like...45nm Technology , 6 input Look-Up Tables, High-speed GTP serial transceivers in the LXT FPGAs, Data rates up to 800 Mb/s (12.8 Gb/s peak bandwidth).

Basic Digital Logic Circuits like AND, OR, NOT, NAND, XOR etc. and Application-Based circuits like Traffic Light Controller, Magnitude Comparator, ALU are also designed and analysed in XILINX ISE Design Suite Software.

Various important parameters like Individual Gate Time Delay, Power Consumed, Critical Delay, Area covered which decides the performance of the digital circuits are also discussed and explained in this report.

Analysis of these above mentioned circuits is made by referring to the RTL Schematic, Technology Schematic and Timing Diagrams generated by XILINX ISE tool. This tool provides the utility for High Level Synthesis for Front End Design.

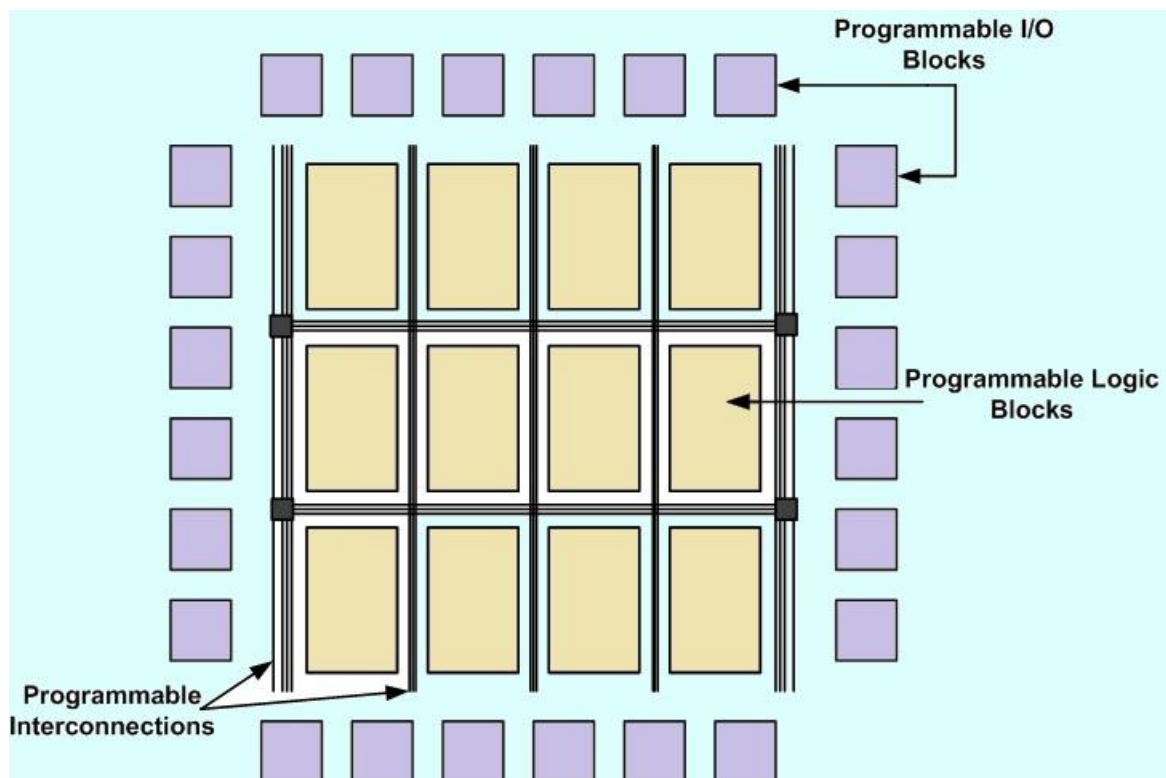
CONTENTS

1. Introduction to FPGA.....	1
1.1 Features of FPGA.....	1
1.2 Advantages.....	3
2. Basic Xilinx ISE Design Suite Software.....	4
2.1 Features of XILINX SPARTAN-6 XC6SLX45.....	8
3. Steps to write a Verilog module in Xilinx.....	9
3.1 Check syntax	
3.2 RTL design	
3.3 Synthesis	
3.4 Test bench	
3.5 Simulation	
3.6 Timing diagram	
3.7 UCF through Plan Ahead	
3.8 Implementation	
3.9 Generating bit file	
3.10 Program burnt on Spartan kit	
4. Some Functions Implementation Using Verilog.....	20
5. Designs and Analysis of Circuits.....	30
5.1 Ripple Carry Adder.....	30
5.2 Ripple Carry Counter.....	32
5.3 Asynchronous D Flip-Flop.....	34
5.4 Synchronous D Flip-Flop.....	36
5.5 Magnitude Comparator	38
5.6 Arithmetic Logic Unit.....	43
5.7 Traffic Light Controller.....	45
6. Conclusion.....	53
References	

1. INTRODUCTION TO FPGA

FPGAs are programmable semiconductor devices that are based around a matrix of Configurable Logic Blocks (CLBs) connected through programmable interconnects. As opposed to Application Specific Integrated Circuits (ASICs), where the device is custom built for the particular design, FPGAs can be programmed to the desired application or functionality requirements. Although One-Time Programmable (OTP) FPGAs are available, the dominant type are SRAM-based which can be reprogrammed as the design evolves. FPGAs allow designers to change their designs very late in the design cycle– even after the end product has been manufactured and deployed in the field. In addition, Xilinx FPGAs allow for field upgrades to be completed remotely, eliminating the costs associated with re-designing or manually updating electronic systems.

1.1 FPGAs COMMON FEATURES



FPGAs have evolved far beyond the basic capabilities present in their predecessors, and incorporate hard (ASIC type) blocks of commonly used functionality such as RAM, clock management and DSP.

The following are the basic components in a FPGA:-

- **CLB:**

The Configurable Logic Block is the basic logic unit in a FPGA. Exact numbers and features vary from device to device, but every CLB consists of a configurable switch matrix with 4 or 6 inputs, some selection circuitry (MUX, etc), and flip-flops. The switch matrix is highly flexible and can be configured to handle combinatorial logic, shift registers or RAM. More architectural details can be found in the applicable device's data sheet.

- **INTERCONNECT:**

While the CLB provides the logic capability, flexible interconnect routing routes the signals between CLBs and to and from I/Os. Routing comes in several flavors, from that designed to interconnect between CLBs to fast horizontal and vertical long lines spanning the device to global low-skew routing for Clocking and other global signals. The design software makes the inter-connect routing task hidden to the user unless specified otherwise, thus significantly reducing design complexity.

- **SELECTIO (IOBs):**

Today's FPGAs provide support for dozens of I/O standards thus providing the ideal interface bridge in your system. I/O in FPGAs is grouped in banks with each bank independently able to support different I/O standards. Today's leading FPGAs provide over a dozen I/O banks, thus allowing flexibility in I/O support.

- **MEMORY:**

Embedded Block RAM memory is available in most FPGAs, which allows for on-chip memory in your design. These allow for on-chip memory for your design. Xilinx FPGAs provide up to 10Mbits of on-chip memory in 36kbit blocks that can support true dual-port operation.

- **CLOCK MANAGEMENT:**

Digital clock management is provided by most FPGAs in the industry (all Xilinx FPGAs have this feature). The most advanced FPGAs from Xilinx offer both digital clock management and phase-looped locking that provide precision clock synthesis combined with jitter reduction and filtering.

1.2 ADVANTAGES OF FPGA

- **PERFORMANCE:**

Taking advantage of hardware parallelism, FPGAs exceed the computing power of digital signal processors (DSPs) by breaking the paradigm of sequential execution and accomplishing more per clock cycle. BDTI, a noted analyst and benchmarking firm, released benchmarks showing how FPGAs can deliver many times the processing power per dollar of a DSP solution in some applications. Controlling inputs and outputs (I/O) at the hardware level provides faster response times and specialized functionality to closely match application requirements.

- **TIME TO MARKET:**

FPGA technology offers flexibility and rapid prototyping capabilities in the face of increased time-to-market concerns. You can test an idea or concept and verify it in hardware without going through the long fabrication process of custom ASIC design. You can then implement incremental changes and iterate on an FPGA design within hours instead of weeks. Commercial off-the-shelf (COTS) hardware is also available with different types of I/O already connected to a user-programmable FPGA chip. The growing availability of high-level software tools decreases the learning curve with layers of abstraction and often offers valuable IP cores (prebuilt functions) for advanced control and signal processing.

- **COST:**

The non-recurring engineering (NRE) expense of custom ASIC design far exceeds that of FPGA-based hardware solutions. The large initial investment in ASICs is easy to justify for OEMs shipping thousands of chips per year, but many end users need custom hardware functionality for the tens to hundreds of systems in development. The very nature of programmable silicon means you have no fabrication costs or long lead times for assembly. Because system requirements often change over time, the cost of making incremental changes to FPGA designs is negligible when compared to the large expense of re-spinning an ASIC.

- **RELIABILITY:**

While software tools provide the programming environment, FPGA circuitry is truly a “hard” implementation of program execution. Processor-based systems often involve several layers of abstraction to help schedule tasks and share resources among multiple processes. The driver layer controls hardware resources and the OS manages memory and processor bandwidth. For any given processor core, only one instruction can execute at a time, and processor-based systems are continually at risk of time-critical tasks preempting one another. FPGAs, which do not use OSs, minimize reliability concerns with true parallel execution and deterministic hardware dedicated to every task.

- **LONG-TERM MAINTENANCE:**

As mentioned earlier, FPGA chips are field-upgradable and do not require the time and expense involved with ASIC redesign. Digital communication protocols, for example, have specifications that can change over time, and ASIC-based interfaces may cause maintenance and forward-compatibility challenges. Being reconfigurable, FPGA chips can keep up with future modifications that might be necessary. As a product or system matures, you can make functional enhancements without spending time redesigning hardware or modifying the board layout.

2. Basic XILINX ISE Design Suite

Xilinx ISE(Integrated Synthesis Environment) is a software tool produced by Xilinx for synthesis and analysis of HDL designs, enabling the developer to synthesize ("compile") their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmer. Xilinx tools is a suite of software tools used for the design of digital circuits implemented using Xilinx field programmable gate array(FPGA) or complex programmable logic device (CPLD).

The design procedure consist of:-

1. Design entry
2. Synthesis and Implementation of the design
3. Functional Simulation
4. Testing and Verification

Digital designs can be entered in various ways using the above CAD tools: using a schematic entry tool, using a hardware description languages (HDL) - Verilog or VHDL or a combination of both. In this lab we will only use the design flow that involves the use of Verilog HDL. The CAD tools enable you to design combinational and sequential circuits starting with Verilog HDL design specifications.

The steps of this design procedure are listed below :-

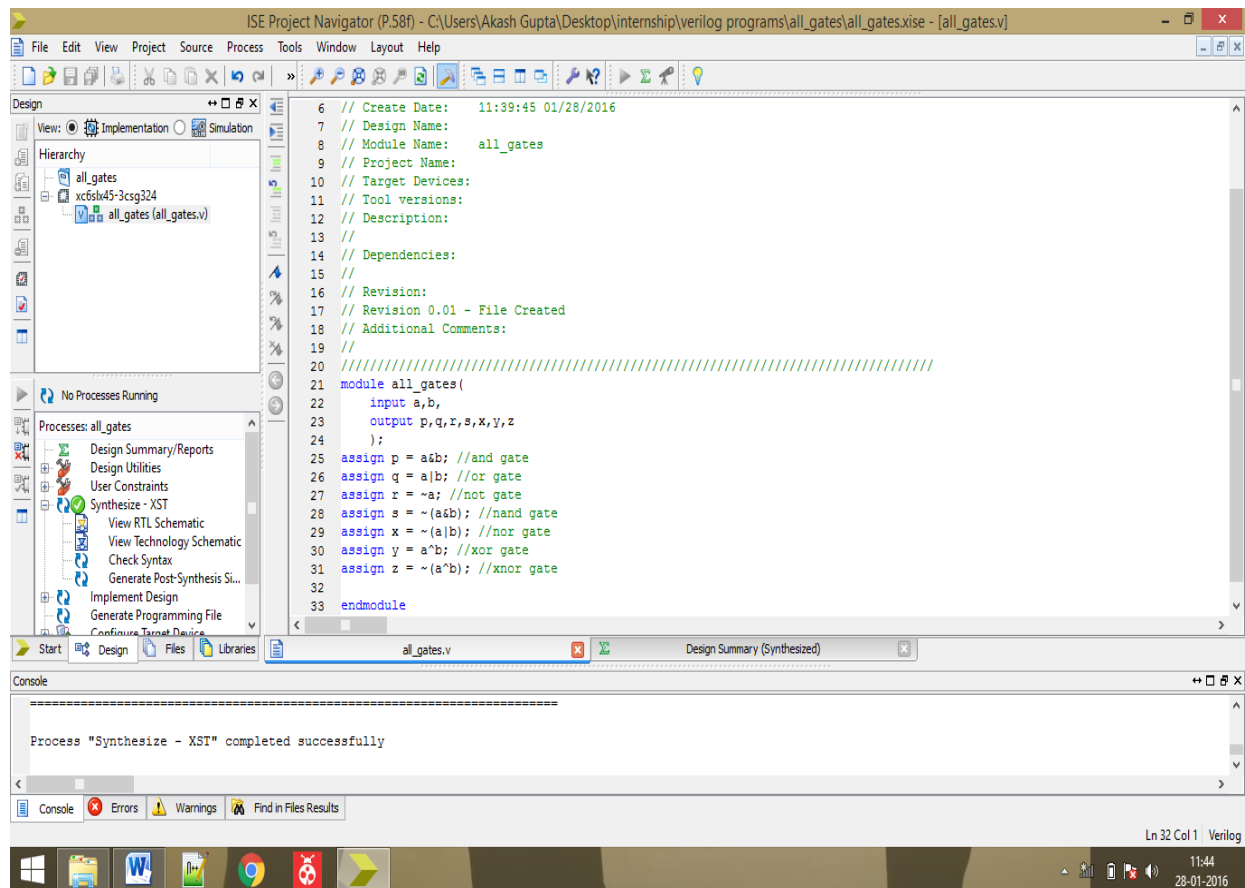
1. Create verilog design input file(s) using template driven editor.
2. Compile and implement the verilog design file(s).
3. Create the test vectors and simulate the design (functional simulation) without using a PLD (FPGA or CPLD).
4. Assign input/output pins to implement the design on a target device.
5. Download bit stream to an FPGA or CPLD device.

A Verilog input file in the Xilinx software environment consists of the following segments -

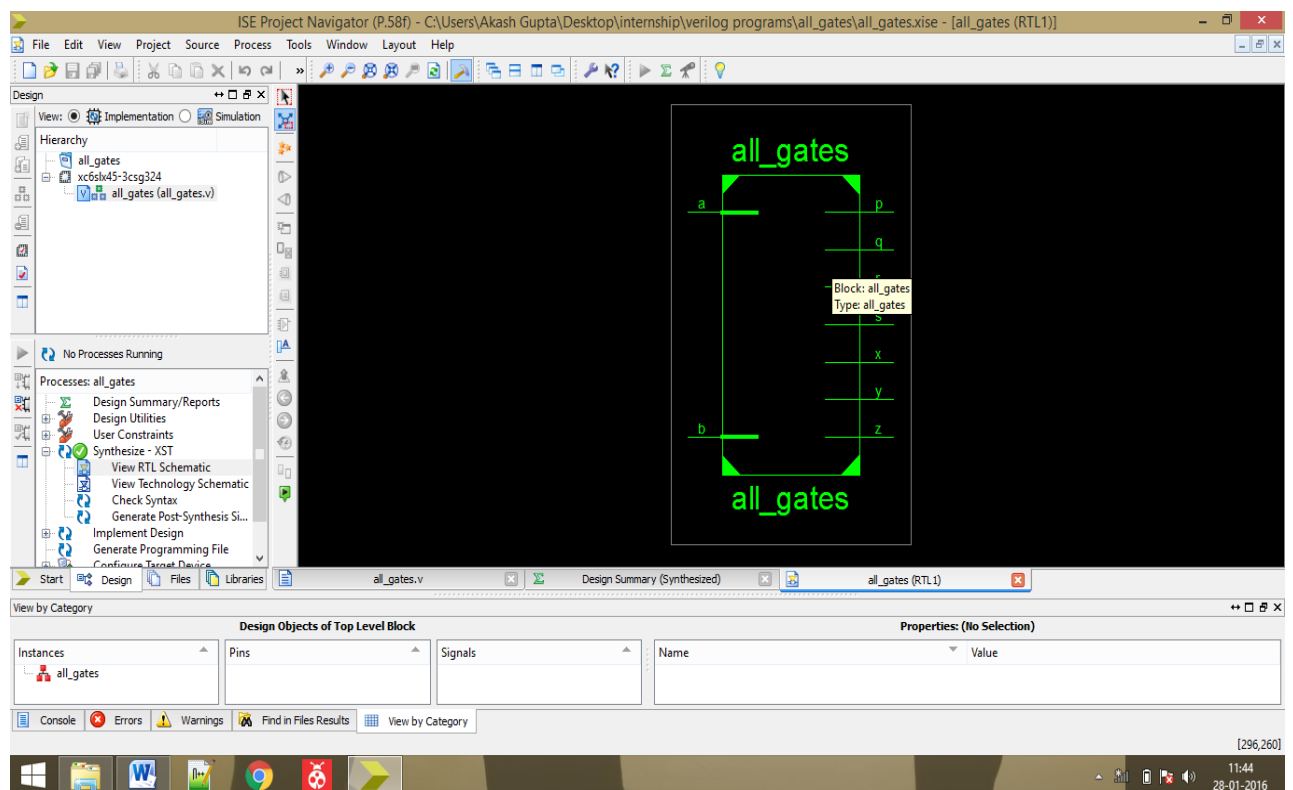
- Header – module name, list of input and output ports.
- Declarations – input and output ports, registers and wires.
- Logic descriptions – equations, state machines and logic functions.
- End – endmodule.

EXAMPLE USING VERILOG -

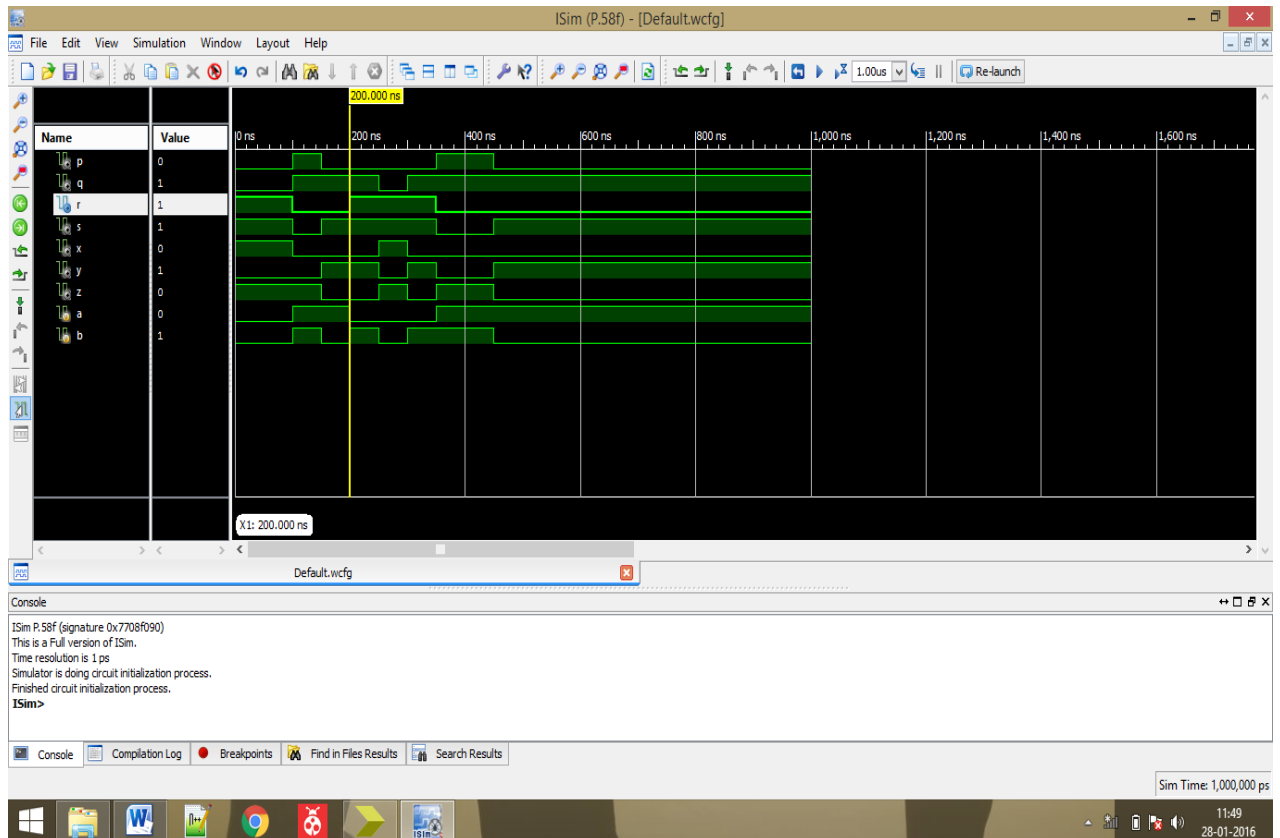
Realization of all logic gates– with RTL schematic and delay table



RTL Schematic-



Simulation of Behavioral Model –



Data Sheet report:

* All values displayed in nanoseconds (ns)

Source pad	Designation pad	Delay
A	p	4.068
A	q	4.072
A	s	3.912
A	x	3.910
A	y	4.048
A	z	3.798
B	p	3.913
b	q	3.917
b	r	3.983
B	s	3.764
B	x	3.762
B	y	3.896
B	z	3.654

*Power leakage of above function is 0.042 watt.

2.1. Features of XILINX SPARTAN-6 XC6SLX45

- 45nm Technology
- 6 input Look-Up Tables
- Designed for low cost
- Low static and dynamic power
- Multi-voltage, multi-standard Select IO interface banks
- High-speed GTP serial transceivers in the LXT FPGAs
- Integrated Endpoint block for PCI Express designs (LXT)
- Low-cost PCI® technology support compatible with the 33 MHz, 32- and 64-bit specification.
- Efficient DSP48A1 slices
- Integrated Memory Controller blocks
- Data rates up to 800 Mb/s (12.8 Gb/s peak bandwidth)
- Multi-port bus structure with independent FIFO to reduce design timing issues.
- Abundant logic resources with increased logic capacity
- Block RAM with a wide range of granularity
- Clock Management Tile (CMT) for enhanced performance
- Simplified configuration, supports low-cost standards
- Enhanced security for design protection

3. STEPS TO WRITE A VERILOG MODULE IN XILINX

1. Open Xilinx ISE tool from desktop icon
2. Open New Project from File – New Project and the project in the desired location.
3. Add source to create a Verilog Module .

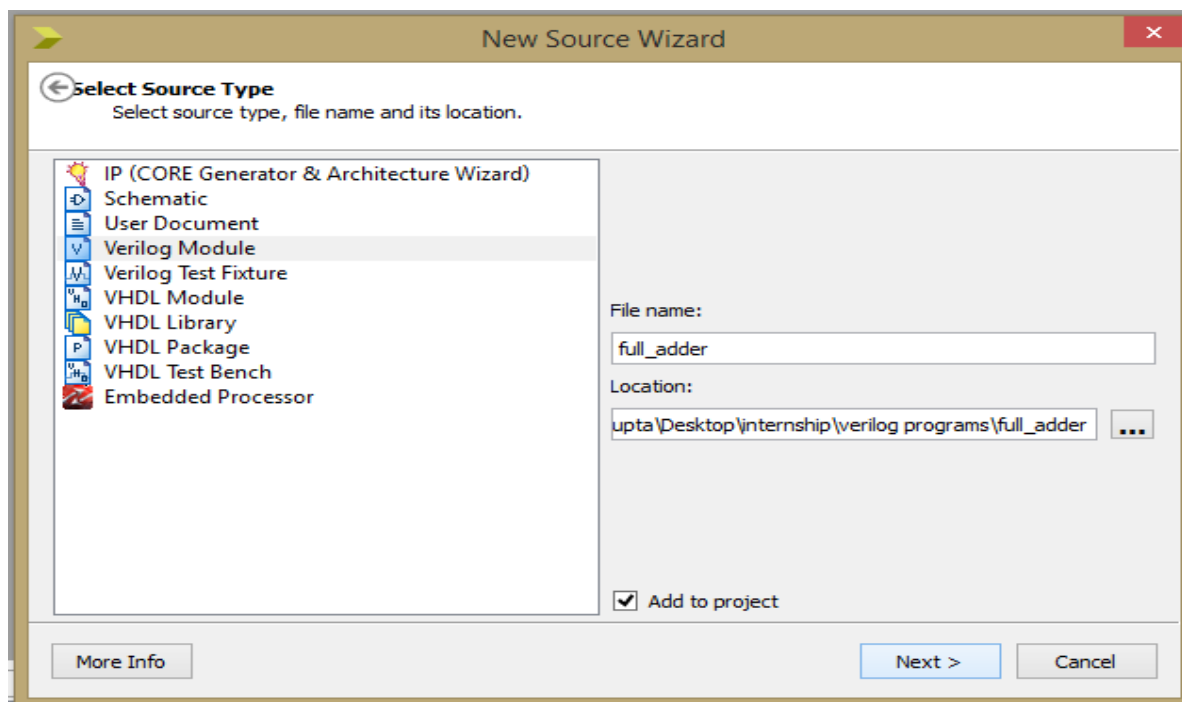
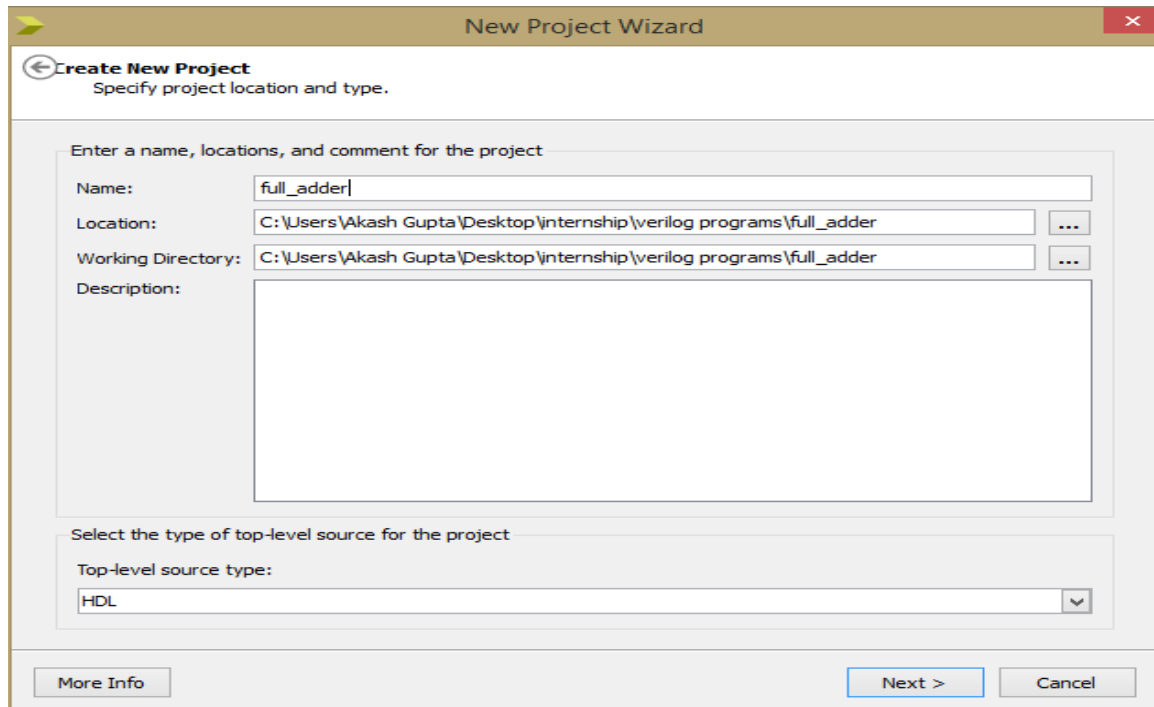


Fig 1 : Adding Verilog Module Source

4. Give the module name, inputs and outputs and press next.

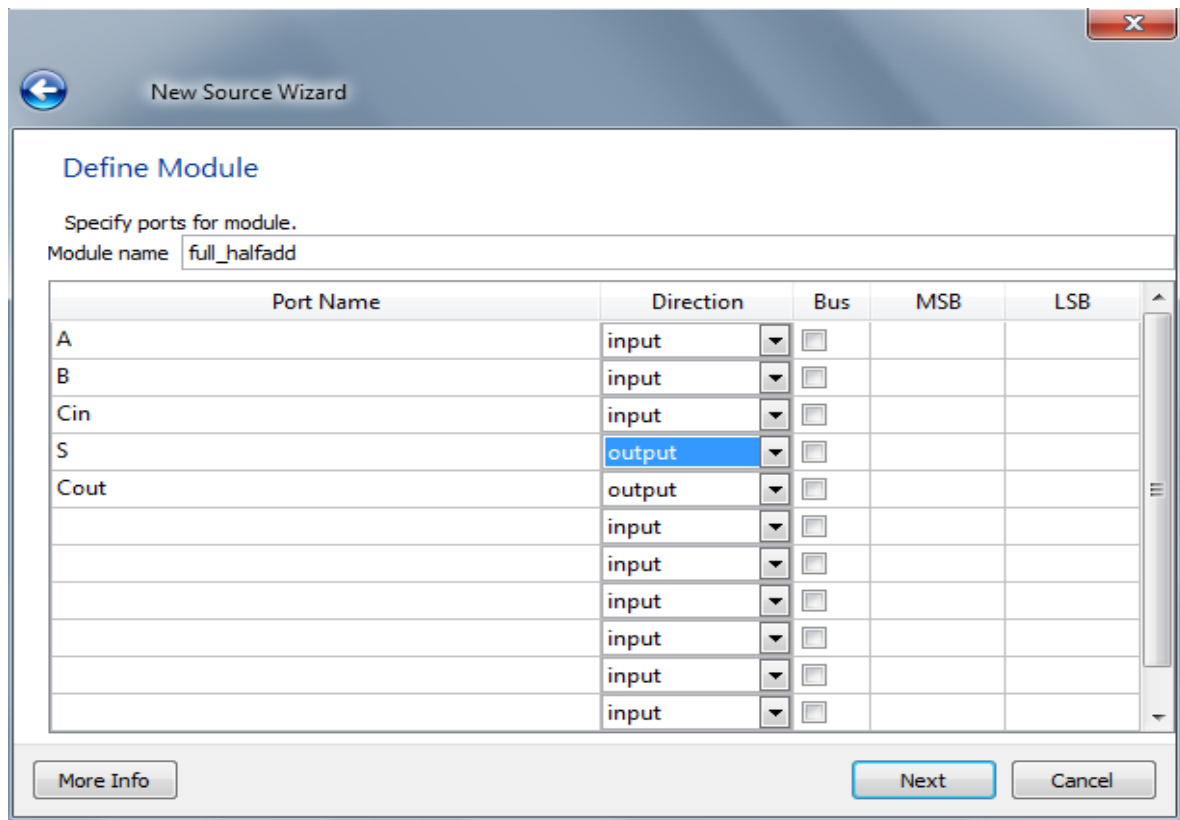
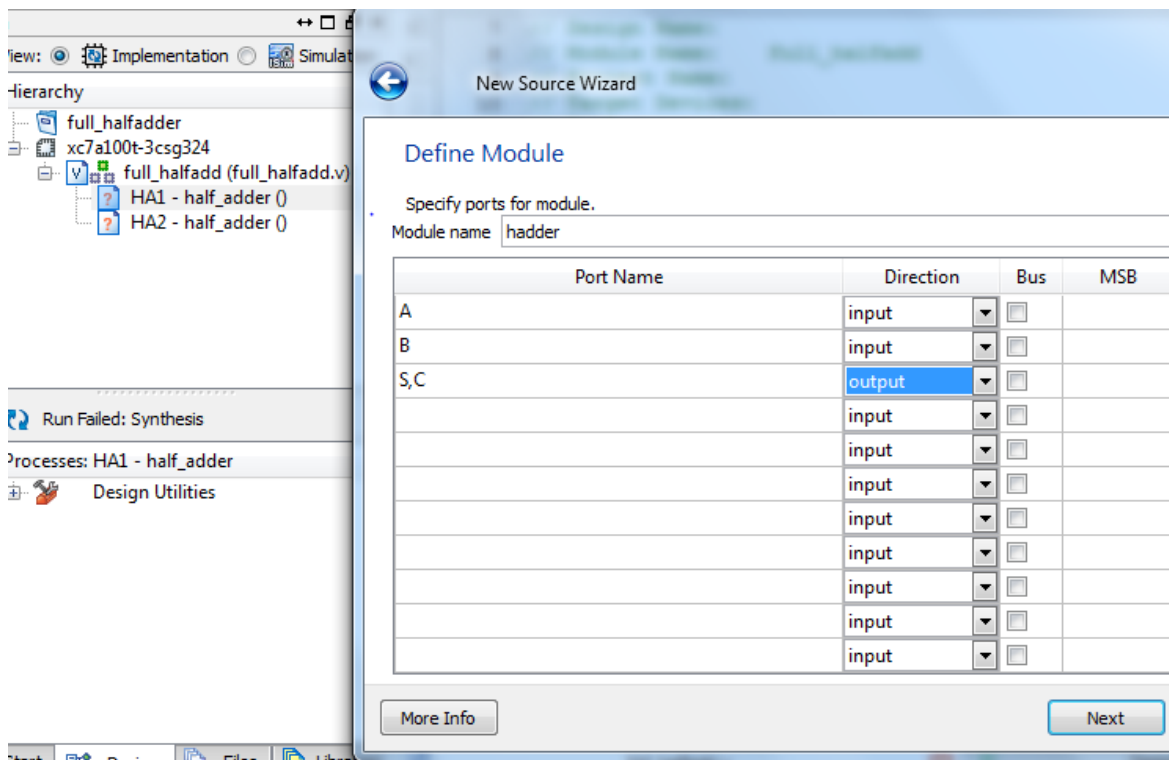


Fig2: Adding Inputs and Outputs to a Verilog Module

5. Write the Verilog code and Synthesize the code.



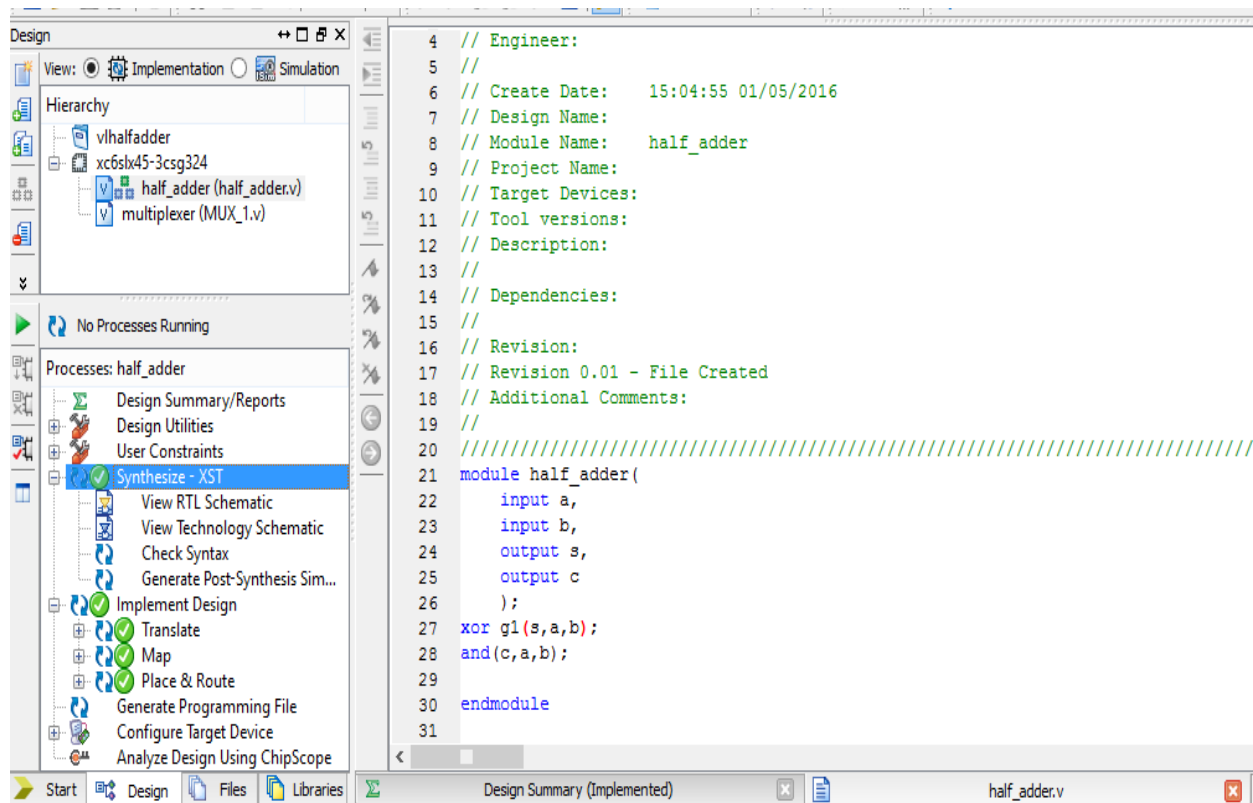


Fig3: Verilog code of half adder

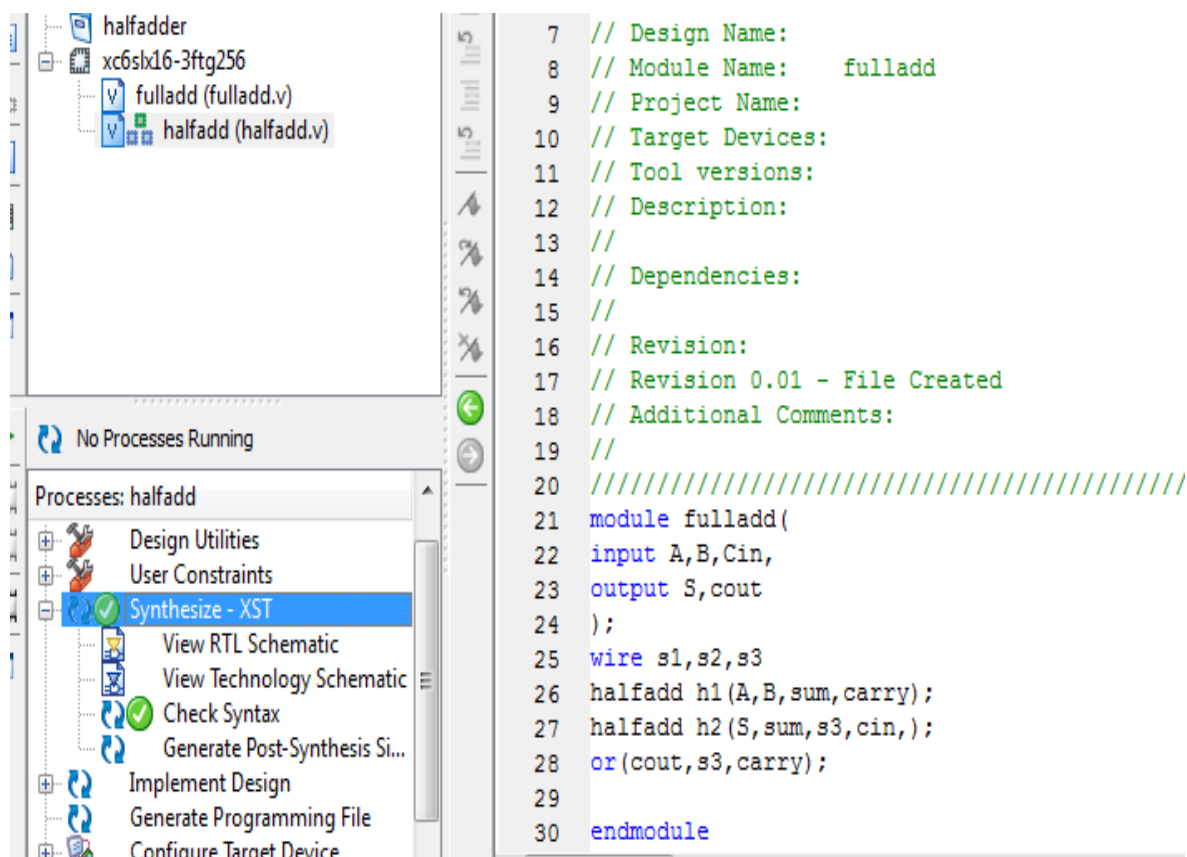


Fig4: Verilog code and synthesis of full adder

6. View the RTL schematic –

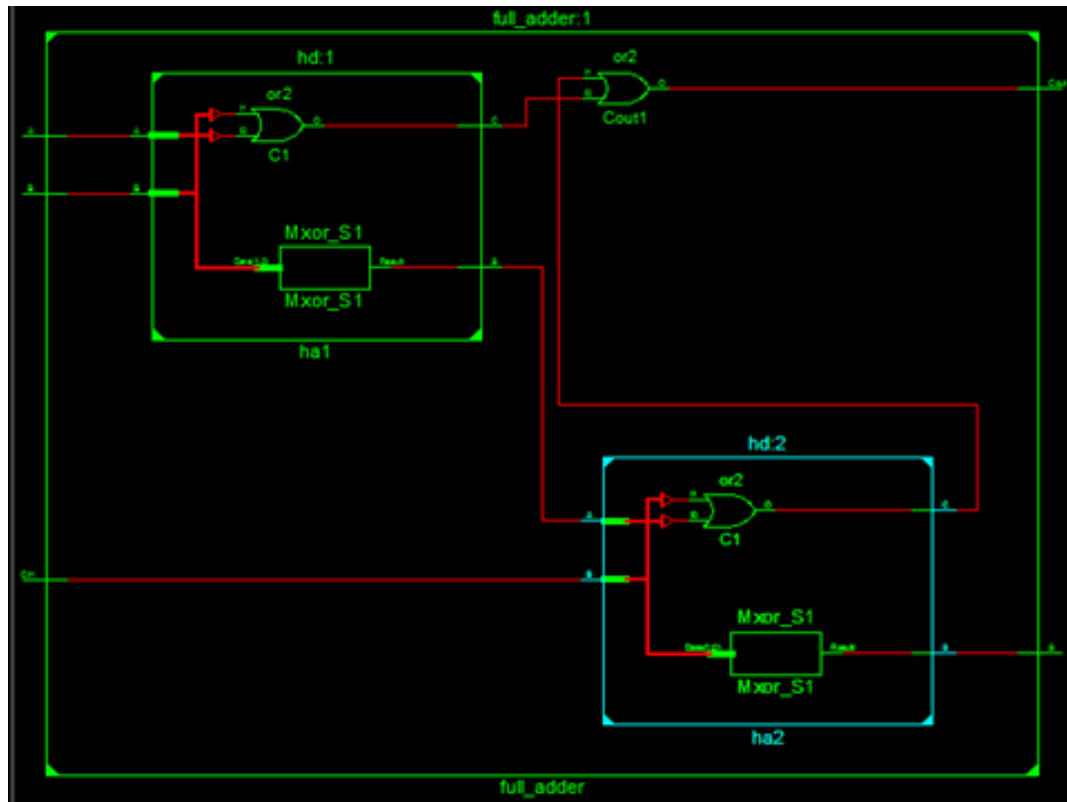


Fig5: RTL schematic of full adder

7. Add New Source for test bench for the full adder

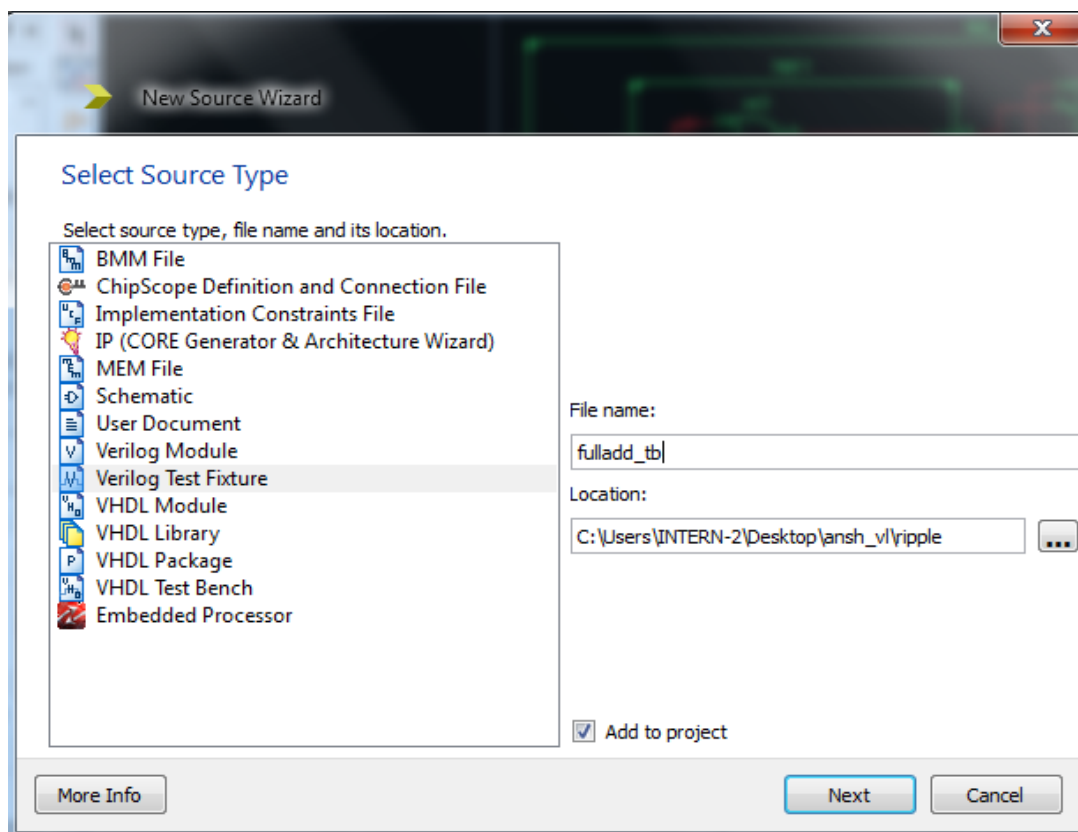


Fig6: Adding Verilog Test Fixture Source

8. Write the test bench for the full adder

```

44
45  initial begin
46      // Initialize Inputs
47      A = 0;
48      B = 0;
49      Cin = 0;
50
51      // Wait 100 ns for global reset to finish
52      #400;
53      // Add stimulus here
54      A = 0;
55      B = 1;
56      Cin = 0;
57
58      #10
59      A = 1;
60      B = 0;
61      Cin = 1;
62
63      #10
64      A = 1;
65      B = 1;
66      Cin = 1;
67  end
68  initial begin
69      $monitor("time=" , $time , , "A=%b B=%b Cin=%b : S=%b Cout=%b ", A,B,Cin,S,Cout);
70  end
71 endmodule

```

Fig7: Test bench code for full adder

9. Go to Simulation – ISim Simulator - Behavioral Check Syntax

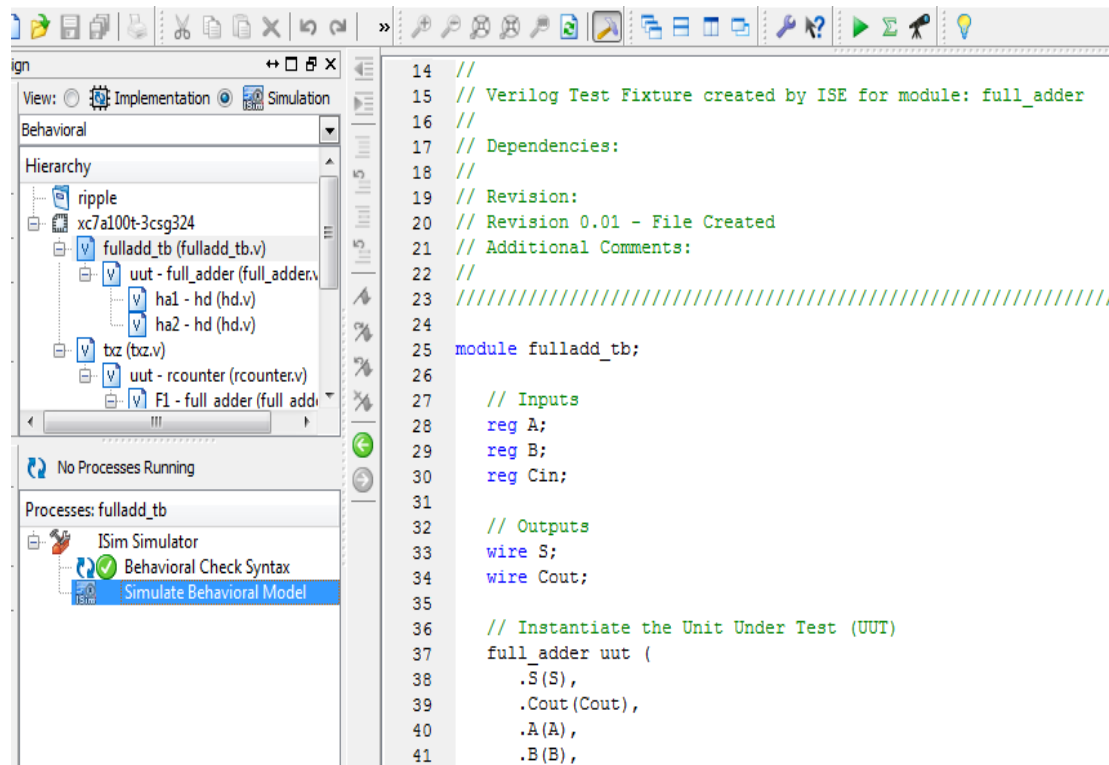


Fig8 : Behavioral Syntax Check

10. Now Simulate Behavioural Model

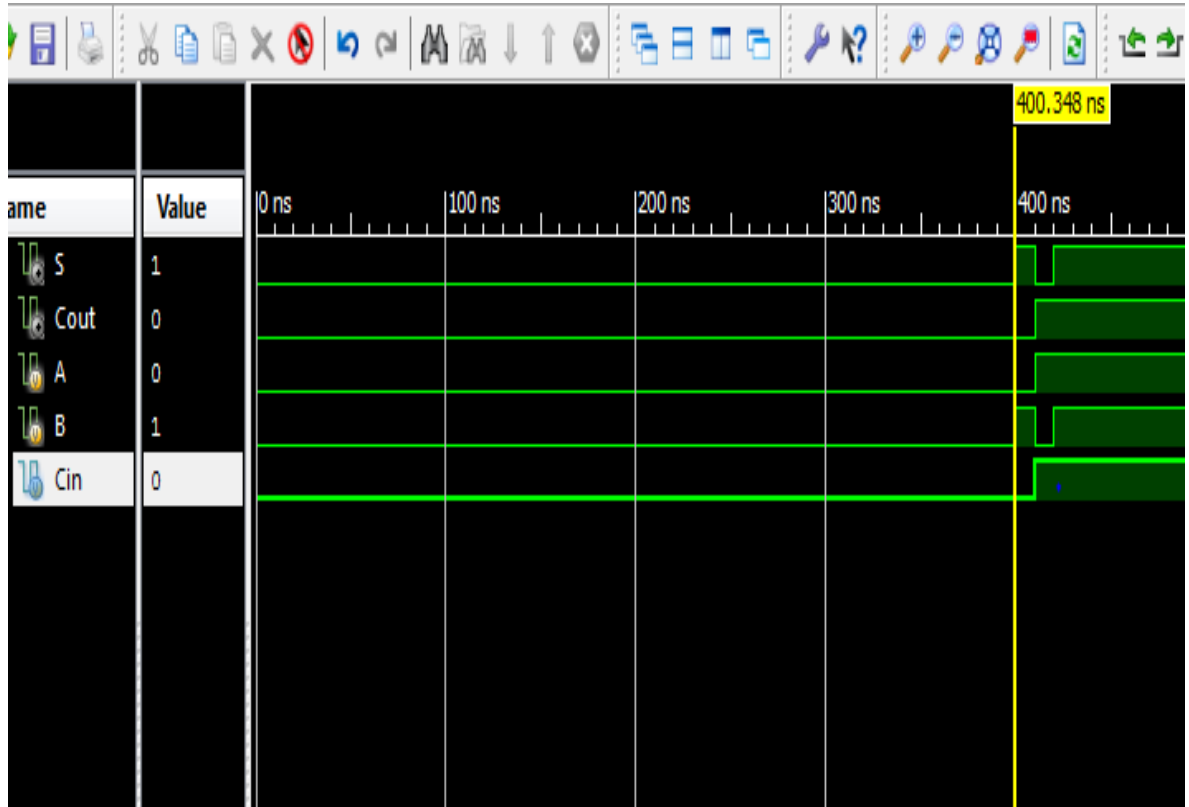


Fig9: Waveforms Generated for full adder

11. Add New Source for Constraint File and give the pin specifications for various inputs and outputs for Spartan 6 FPGA kit.

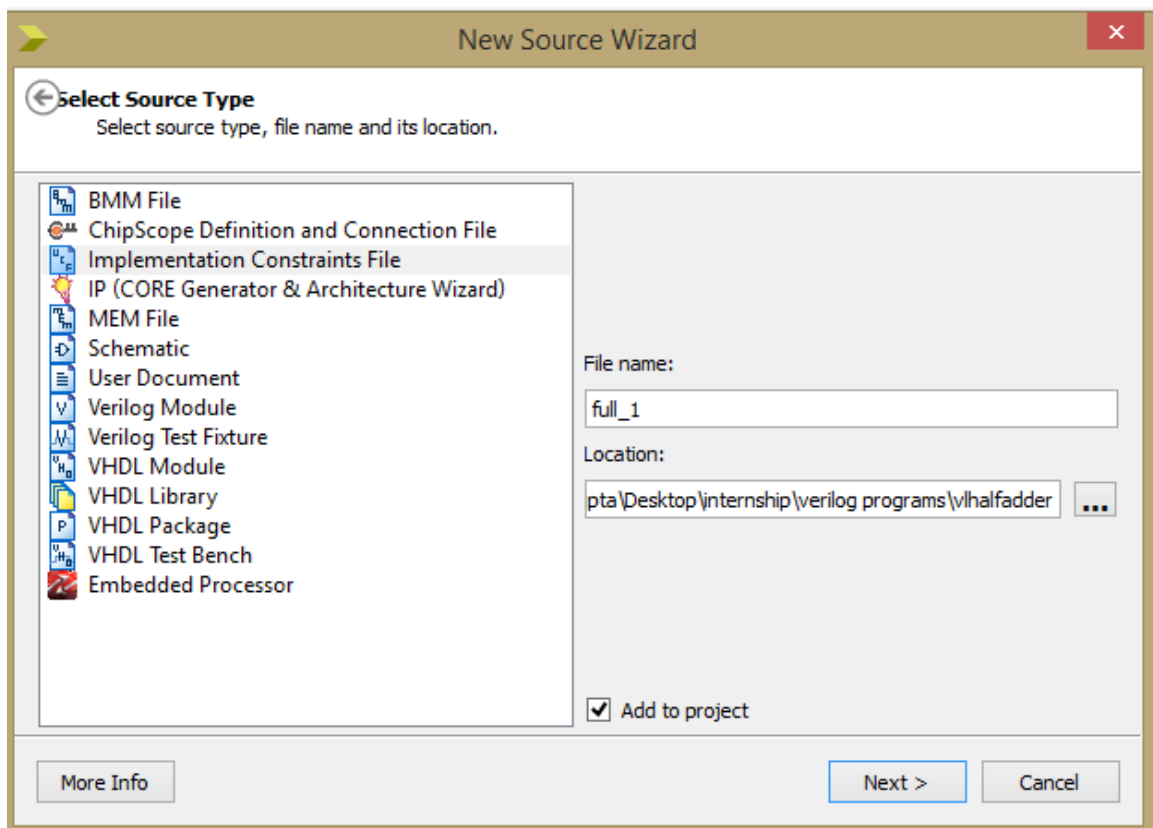


Fig10(a): New source for UCF

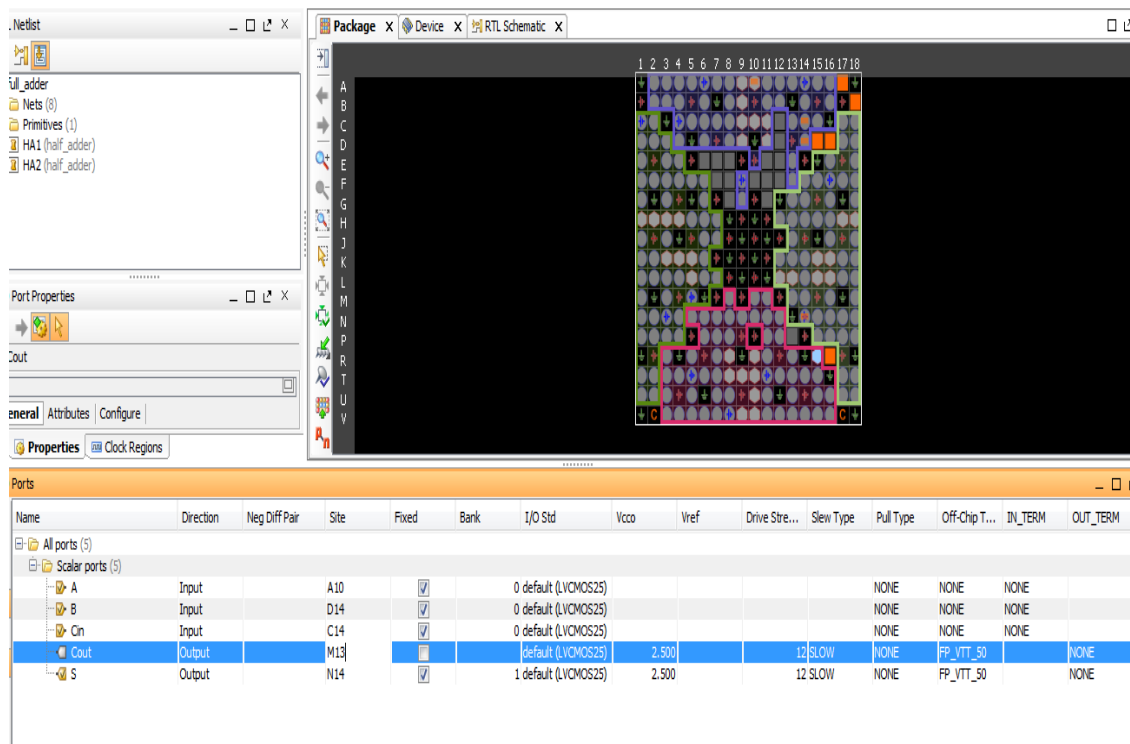


Fig10(b): Giving pin specifications for various inputs & outputs using plan ahead

12. Implement the Design and Generate Bit File.

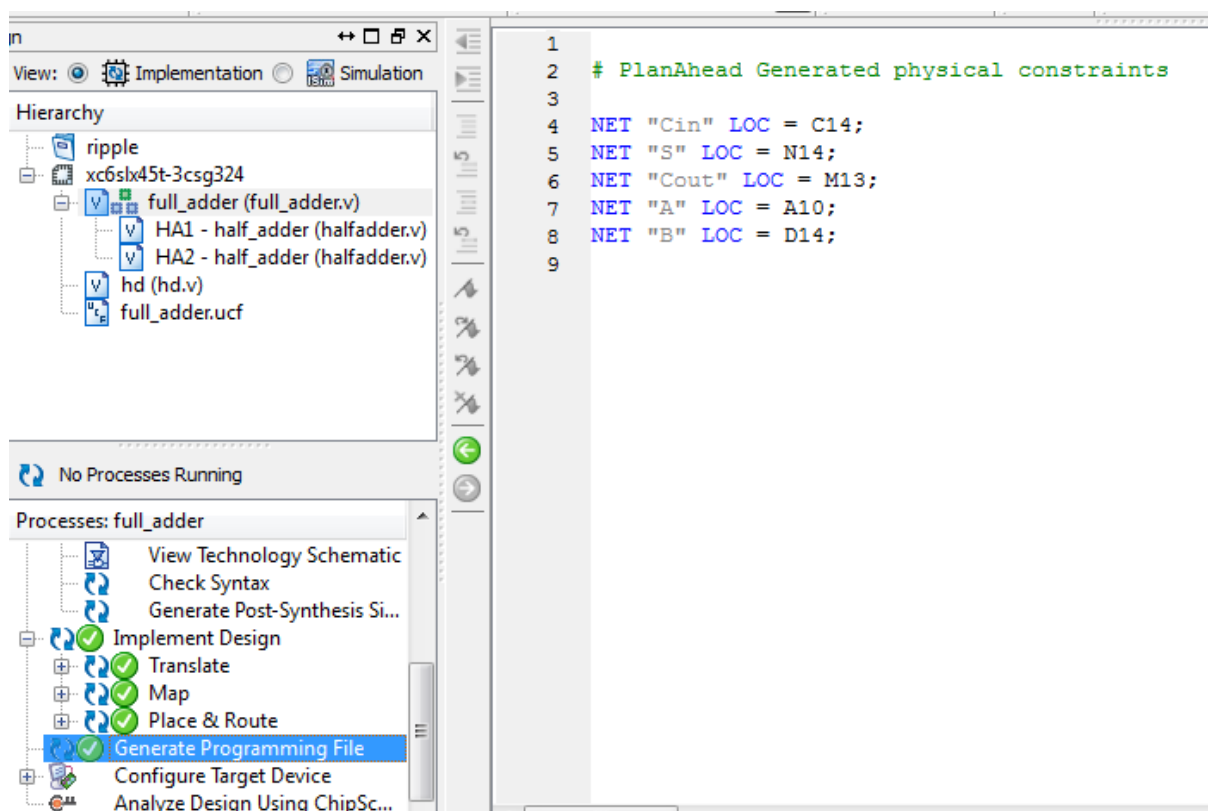


Fig11: Implementation Constraint File

13. Now Open ISE iMPACT tool from Tool - iMPACT

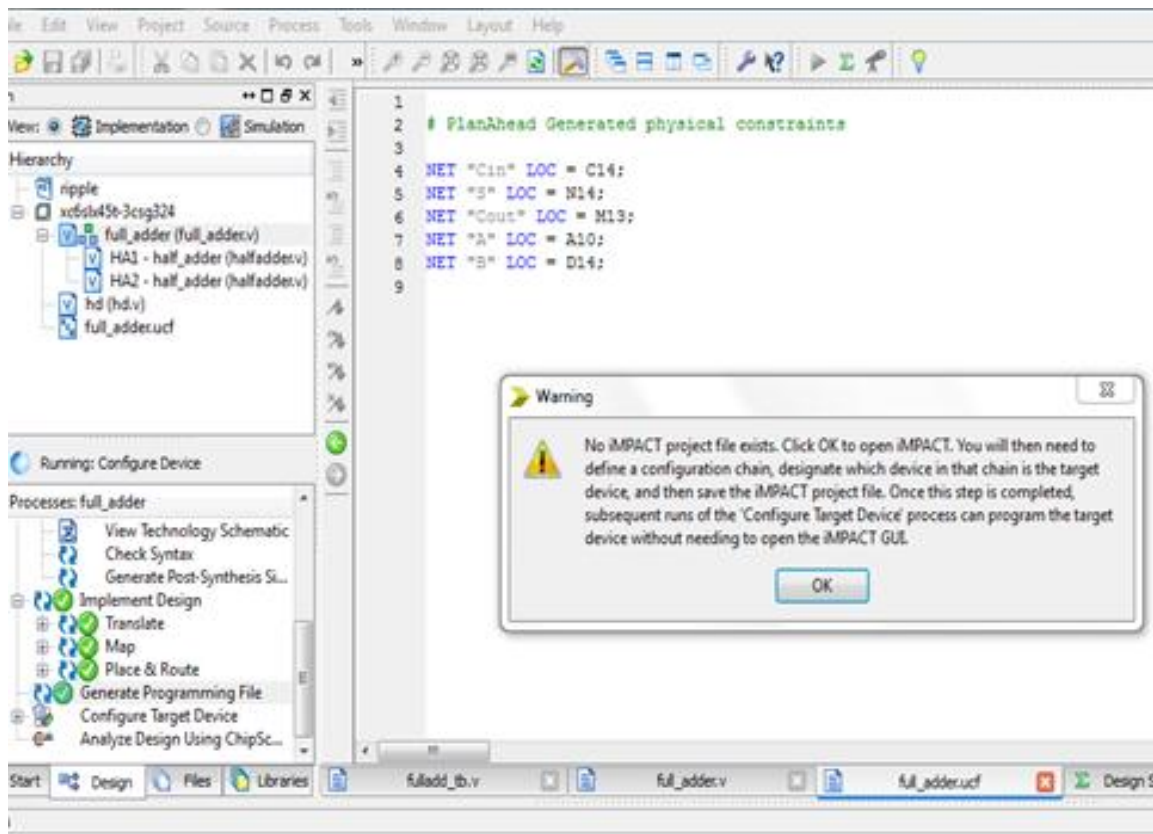
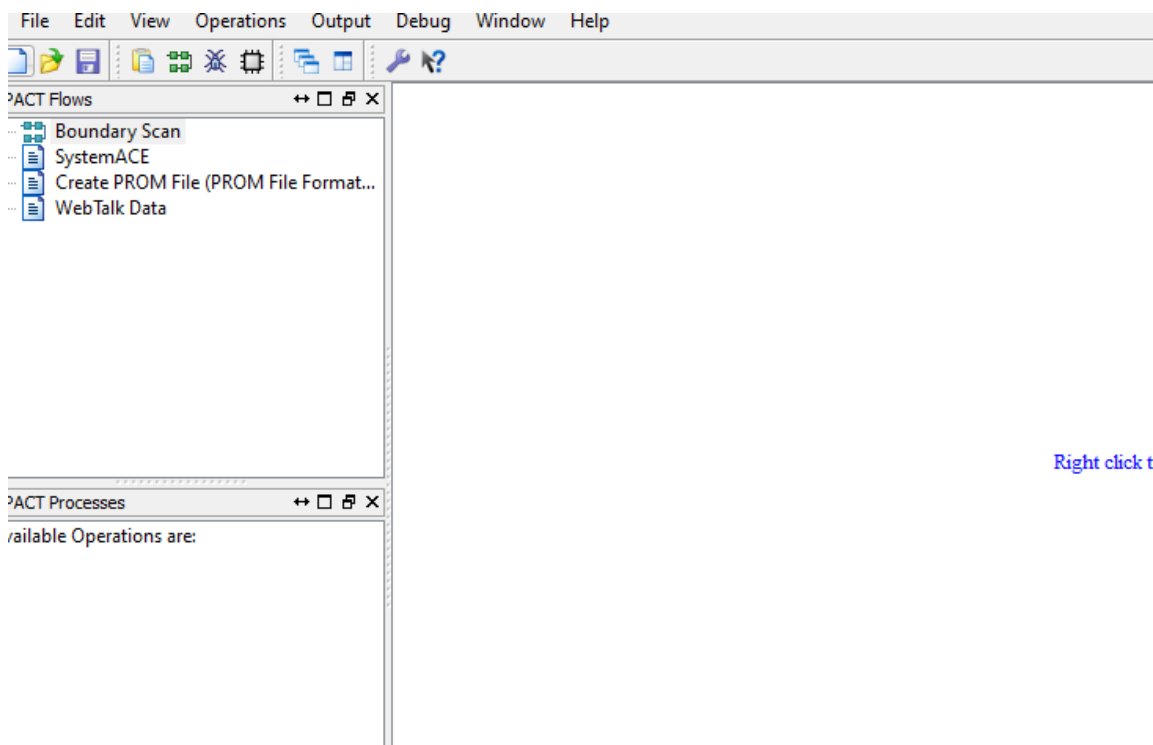


Fig12: Impact tool after generating bit file



14. Add the bit file from File – Open Project – full_adder.bit

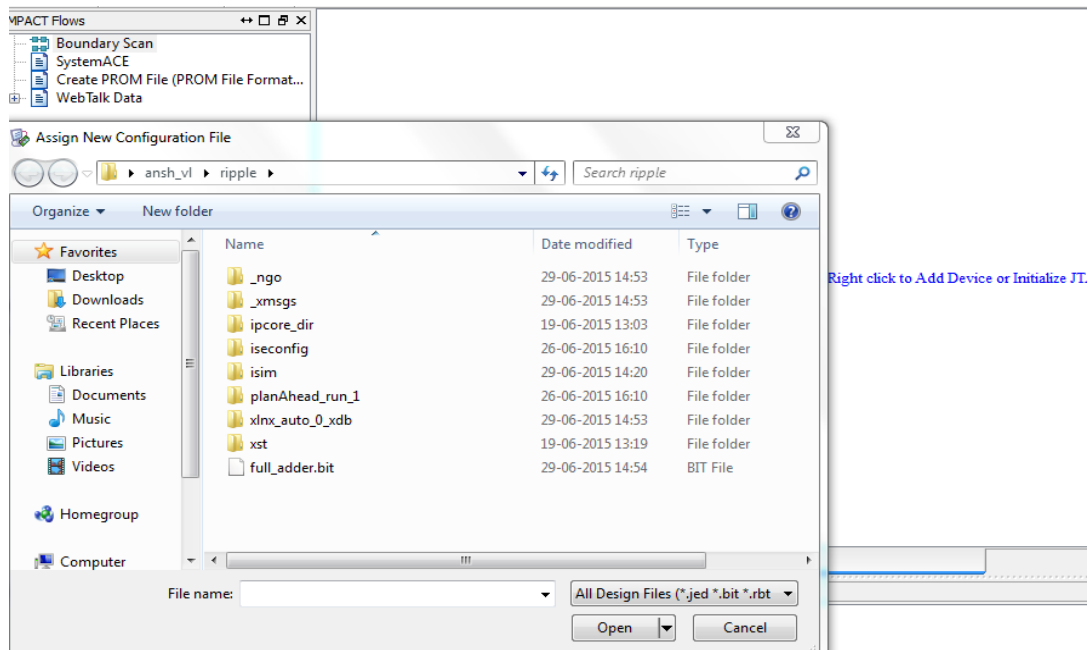


Fig13: Adding bit file to iMPACT Tool

15. Double click on boundary scan and execute.

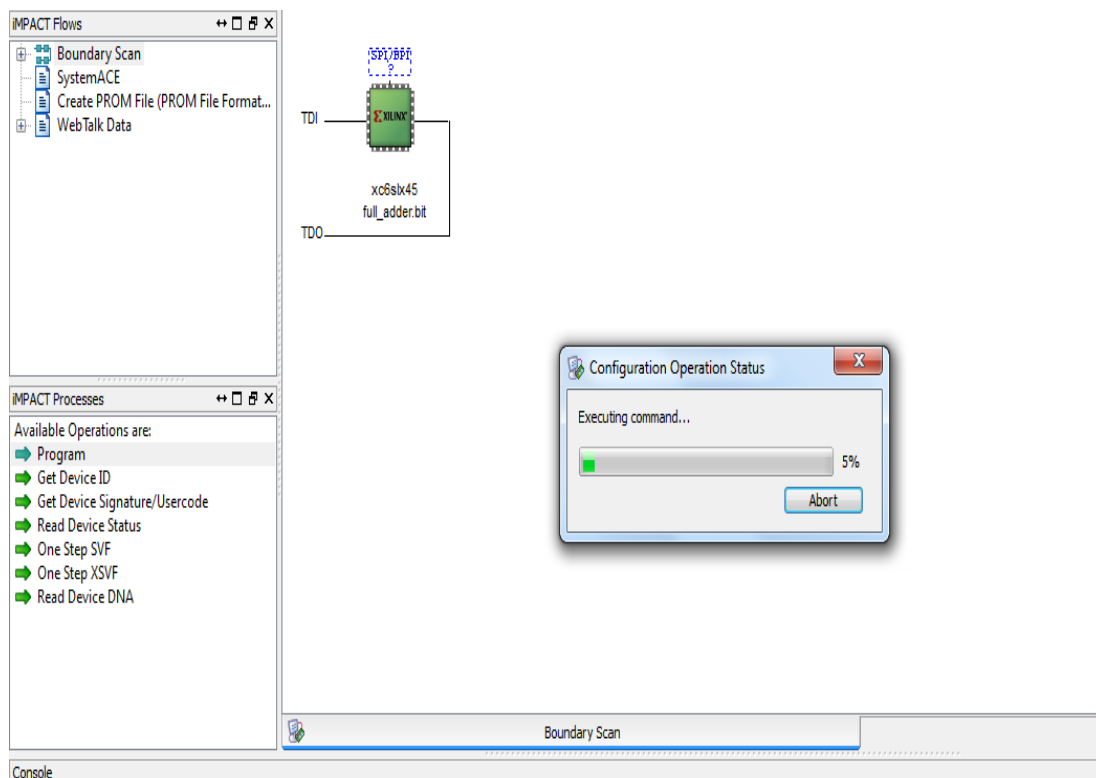


Fig 14: Boundary Scan and execution

16. Connect the Spartan 6 FPGA kit and dump the code into it.

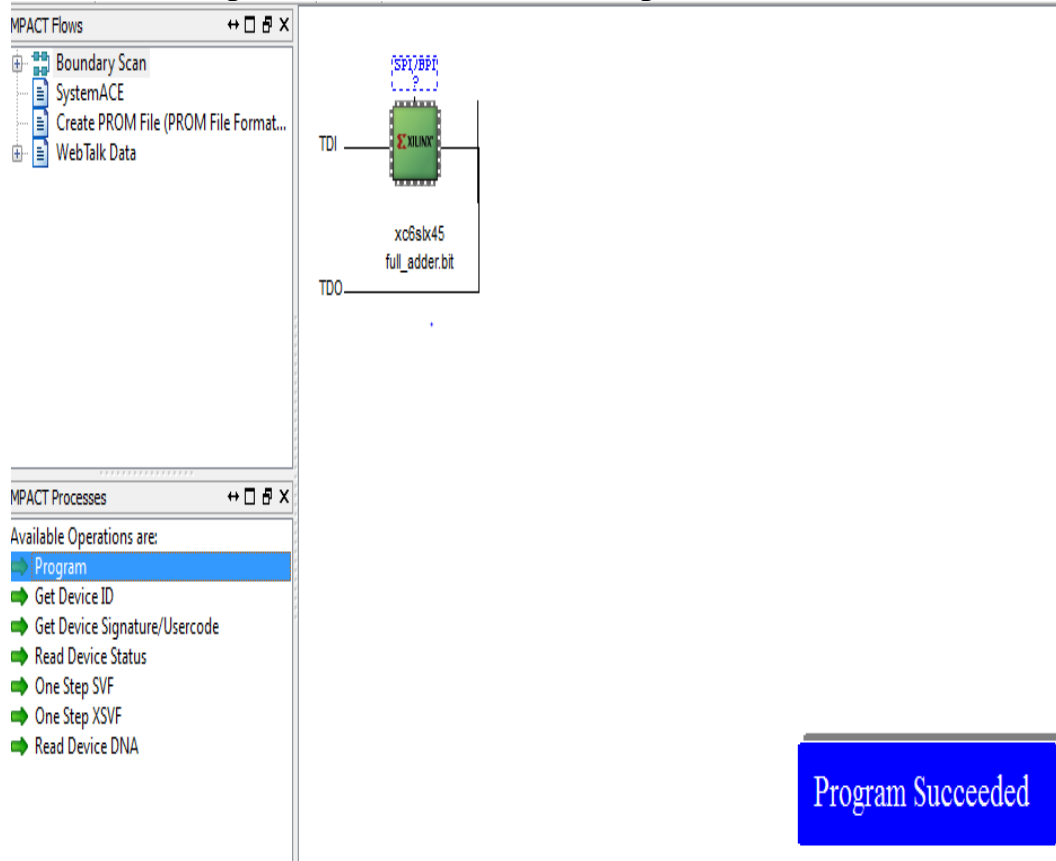


Fig 15: Program Successfully Dumped into FPGA Spartan 6 Kit

17. Verify the results from the FPGA kit.

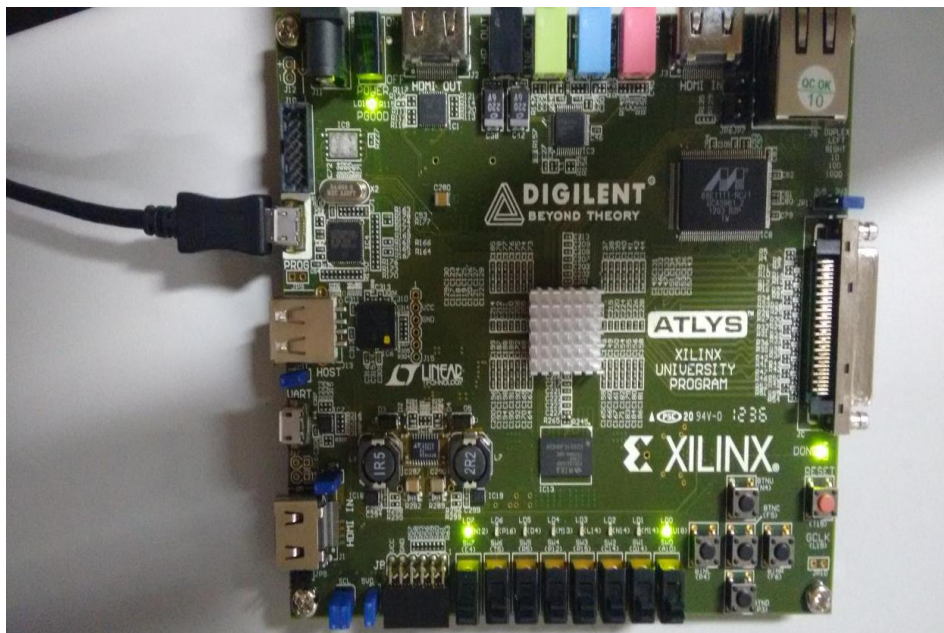


Fig 16. Output

PARAMETERS

AREA COVERED BY FULL ADDER IN FPGA CHIP:

Report : Area
Design : full_adder
Version: H-2013.03-SP5-3
Date : Tue Jan 05 15:04:55 2016

Number of ports: 5
Number of nets: 8
Number of cells: 3
Number of combinational cells: 1
Number of sequential cells: 0
Number of macros/black boxes: 0
Number of buf/inv: 0
Number of references: 3
Combinational area: 49.766400
Buf/Inv area: 0.000000
Noncombinational area: 0.000000
Macro/Black Box area: 0.000000
Net Interconnect area: 1.070661

Total cell area: 49.766400

Total area: 50.837061

REPORT: POWER-ANALYSIS EFFORT LOW

Design : full_adder
Version: H-2013.03-SP5-3
Date : Tue Jan 05 15:04:55 2016

Design	Wire Load Model	Library
full_adder	ForQA	saed90nm_typ_ht
half_adder_0	ForQA	saed90nm_typ_ht
half_adder_1	ForQA	saed90nm_typ_ht

Global Operating Voltage = 1.2

POWER-SPECIFIC UNIT INFORMATION :

Voltage Units = 1V
Capacitance Units = 1.000000ff
Time Units = 1ns
Dynamic Power Units = 1uW (derived from V,C,T units)
Leakage Power Units = 1pW

Cell Internal Power = 13.4923 uW (91%)
 Net Switching Power = 1.3539 uW (9%)
 Total Dynamic Power = 14.8462 uW (100%)

Cell Leakage Power = 994.9086 nW

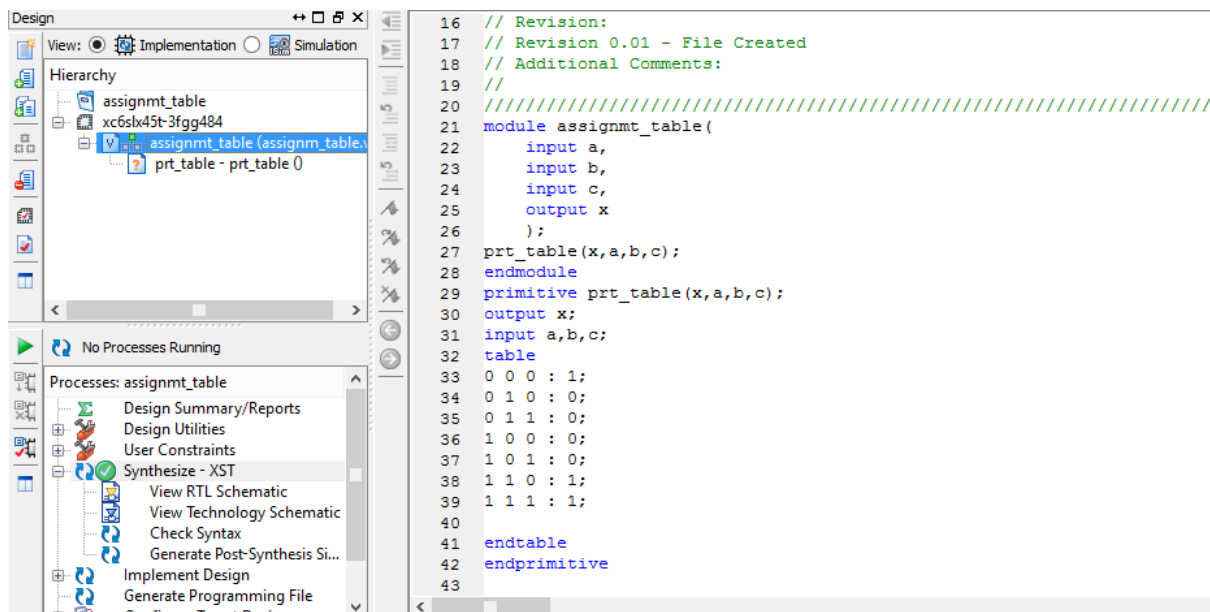
Internal Power Group	Switching Power	Leakage Power	Total Power	Power(%)	Attrs
combinational	13.4923	1.3539	9.9491e+05	15.8411(100.00%)	

Total 13.4923 uW 1.3539 uW 9.9491e+05 pW 15.8411 uW

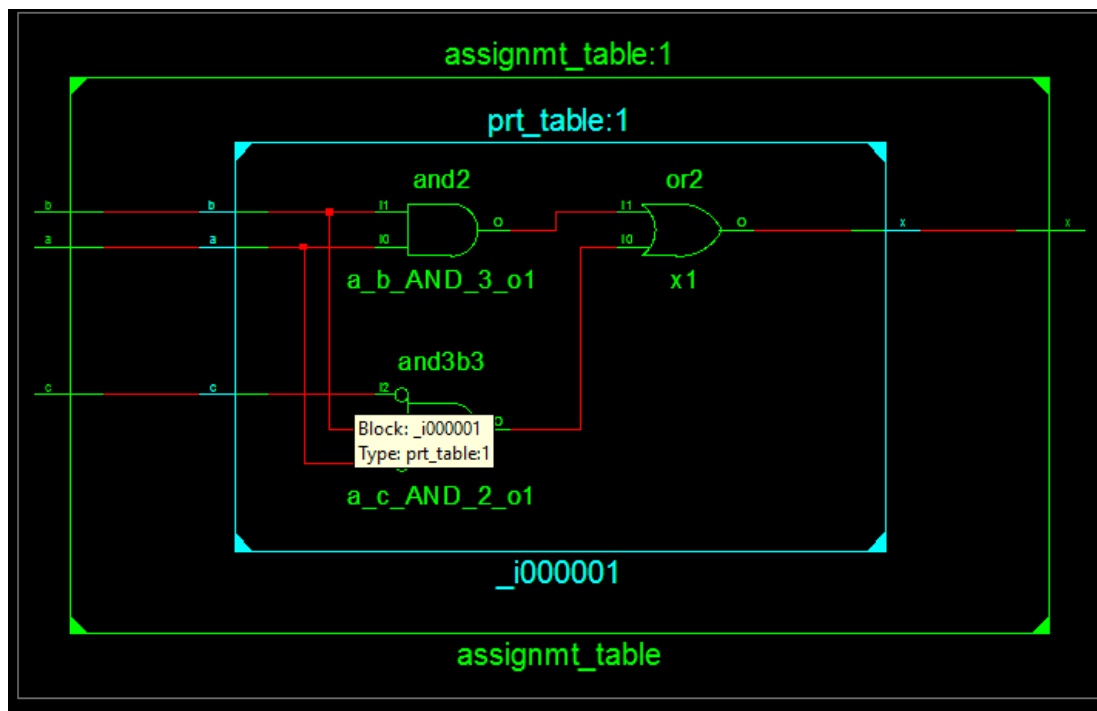
4. SOME FUNCTIONS IMPLEMENTATION USING VERILOG

4.1. Verilog code to implement the function expressed in the following truth table-

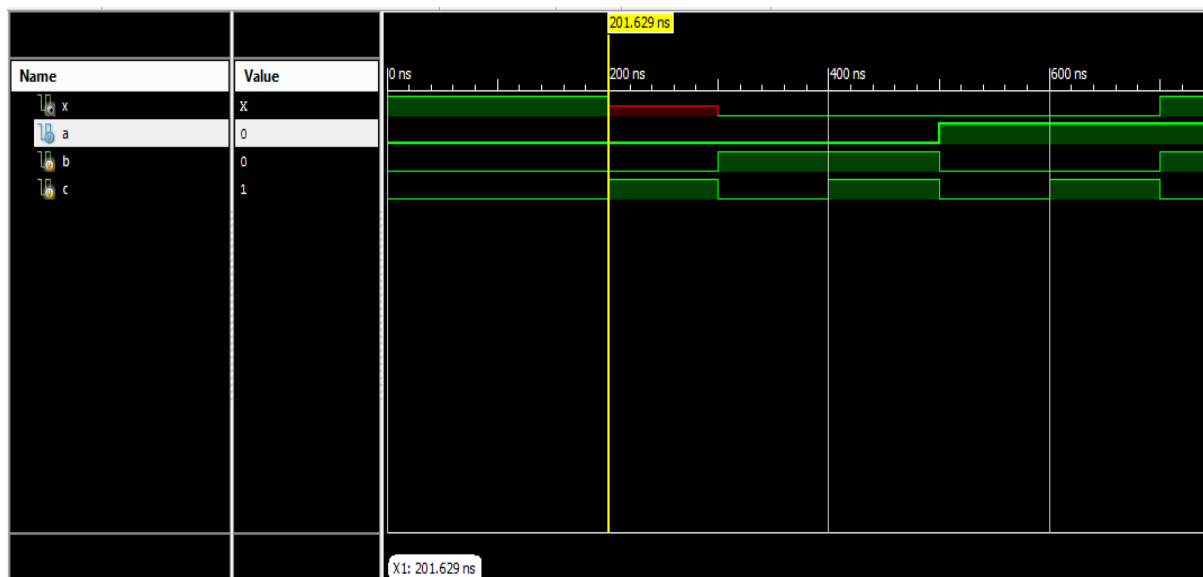
A	B	C	Y
0	0	0	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



RTL Schematic-



Waveforms Generated-



Data Sheet report:

*All values displayed in nanoseconds (ns)

Delay: 5.385 ns (Levels of Logic = 3)

Source: a (PAD)

Destination: x (PAD)

Data Path: a to x

Cell:in->out	fanout	Gate Delay	Net Delay	Logical Name (Net Name)
IBUF:I->O	1	1.222	0.808	a_IBUF (a_IBUF)
LUT3:I0->O	1	0.205	0.579	_i000001/x1 (x_OBUF)
OBUF:I->O		2.571		x_OBUF (x)

Total	5.385ns (3.998ns logic, 1.387ns route) (74.2% logic, 25.8% route)			

*Power leakage of above question is 0.042 watt

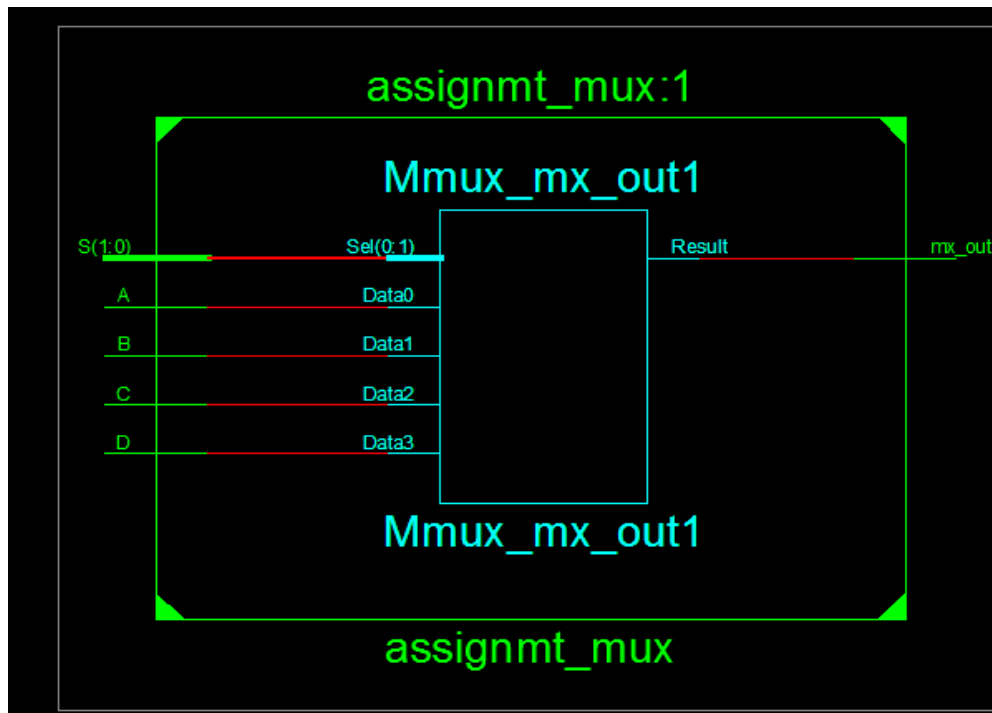
4.2. Implementation of 4:1 MUX using a signal conditional assignment statements. The inputs to the MUX are data inputs and a bus sel. The single output is mx_out.

```

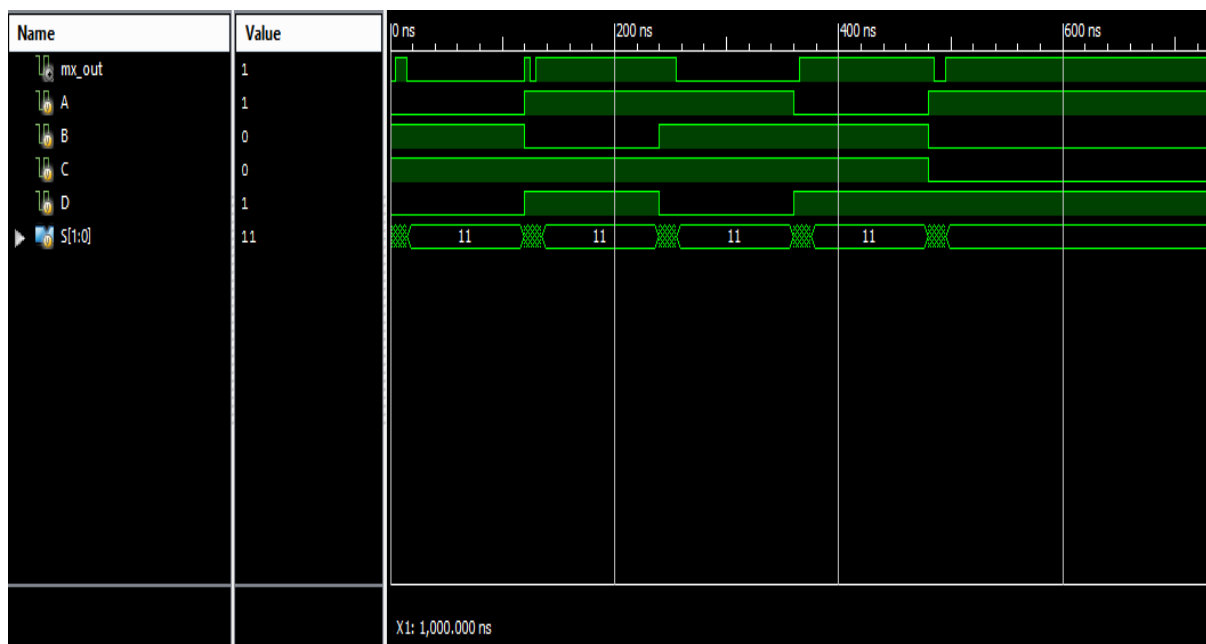
3  //////////////////////////////////////
4  // Company:
5  // Engineer:
6  //
7  // Create Date:    12:27:07 01/12/2016
8  // Design Name:
9  // Module Name:    assignmt_mux
10 // Project Name:
11
12 module assignmt_mux(
13     output mx_out,
14     input A,B,C,D,
15     input [1:0] S
16 );
17
18 reg mx_out;
19 always @( A,B,C,D,S )
20 begin
21     case(S)
22     2'b00: mx_out=A;
23     2'b01: mx_out=B;
24     2'b10: mx_out=C;
25     2'b011: mx_out=D;
26     default : mx_out=1'bx;
27     endcase
28 end
29
30 endmodule

```

RTL Schematic-



Waveforms Generated-



Data Sheet report:

*All values displayed in nanoseconds (ns)

Delay: 5.519ns (Levels of Logic = 3)

Source: S<0> (PAD)

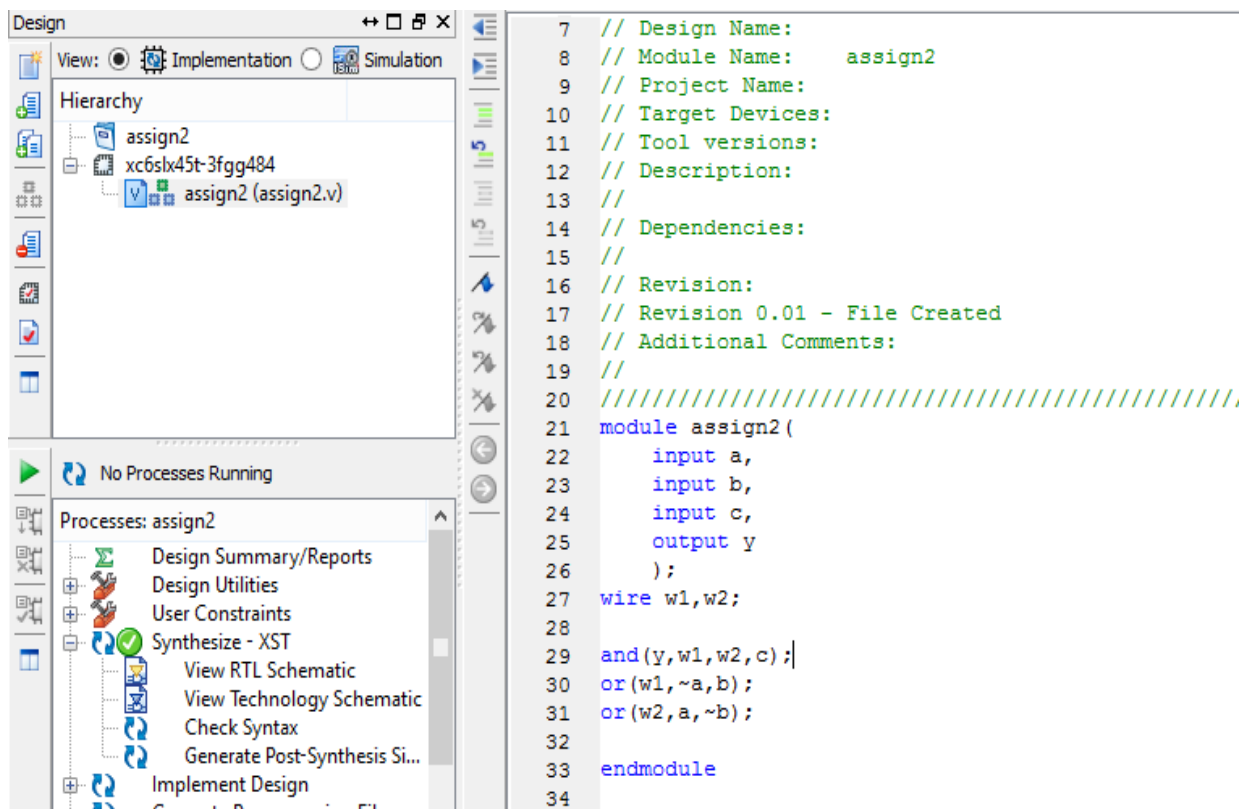
Destination: mx_out (PAD)

Data Path: S<0> to mx_out

Cell:in->out	fanout	Gate Delay	Net Delay	Logical Name (Net Name)
IBUF:I->O	1	1.222	0.944	S_0_IBUF (S_0_IBUF)
LUT6:I0->O	1	0.203	0.579	Mmux_mx_out11(mx_out_OBUF)
OBUF:I->O		2.571		mx_out_OBUF (mx_out)
Total				(3.996ns logic, 1.523ns route) (72.4% logic, 27.6% route)

4.3. Realize the following function using Verilog?

Giving logic representation for the required function-



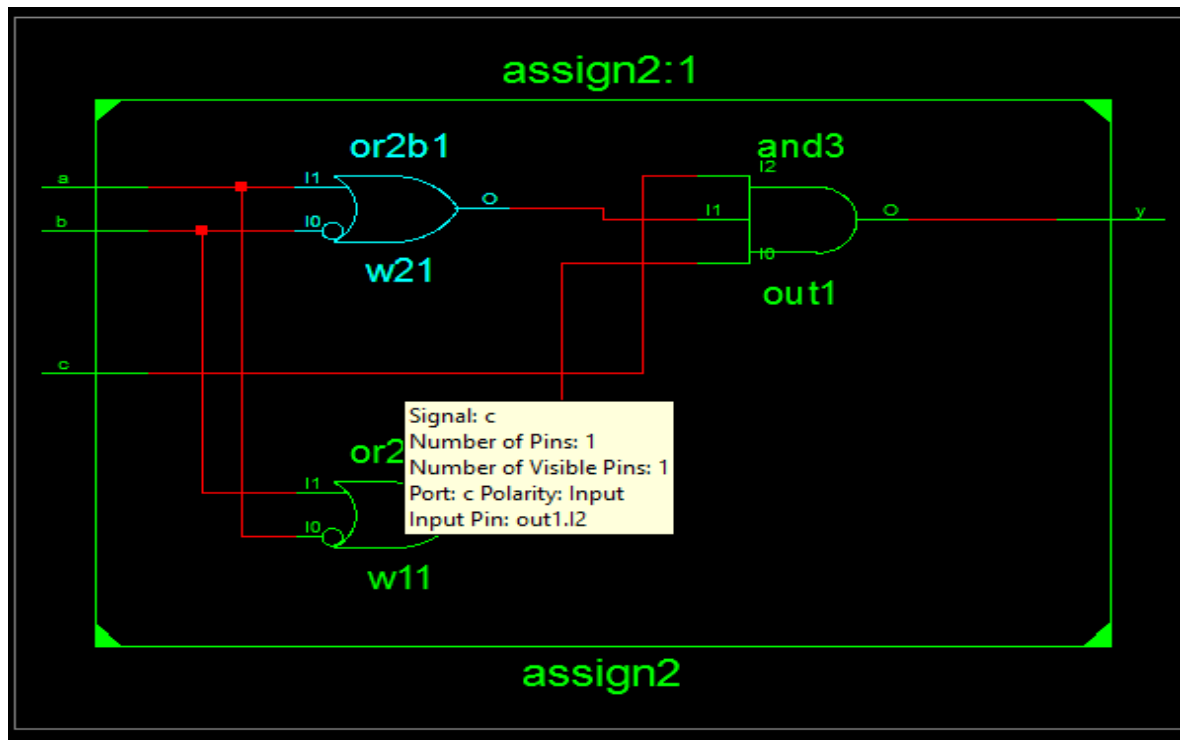
The screenshot displays the Xilinx ISE IDE interface. On the left, the 'Hierarchy' pane shows the project structure: 'assign2' (xc6slx45t-3fgg484) containing 'assign2 (assign2.v)'. Below it, the 'Processes' pane lists various tasks, with 'Synthesize - XST' currently selected. The main editor on the right shows the Verilog code for the 'assign2' module.

```

7 // Design Name:
8 // Module Name:  assign2
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 module assign2(
22     input a,
23     input b,
24     input c,
25     output y
26 );
27 wire w1,w2;
28
29 and(y,w1,w2,c);
30 or(w1,~a,b);
31 or(w2,a,~b);
32
33 endmodule
34

```

RTL Schematic-



- $Y = (AB + A)(C + E) + D$

No Processes Running

Processes: ANSSSSS

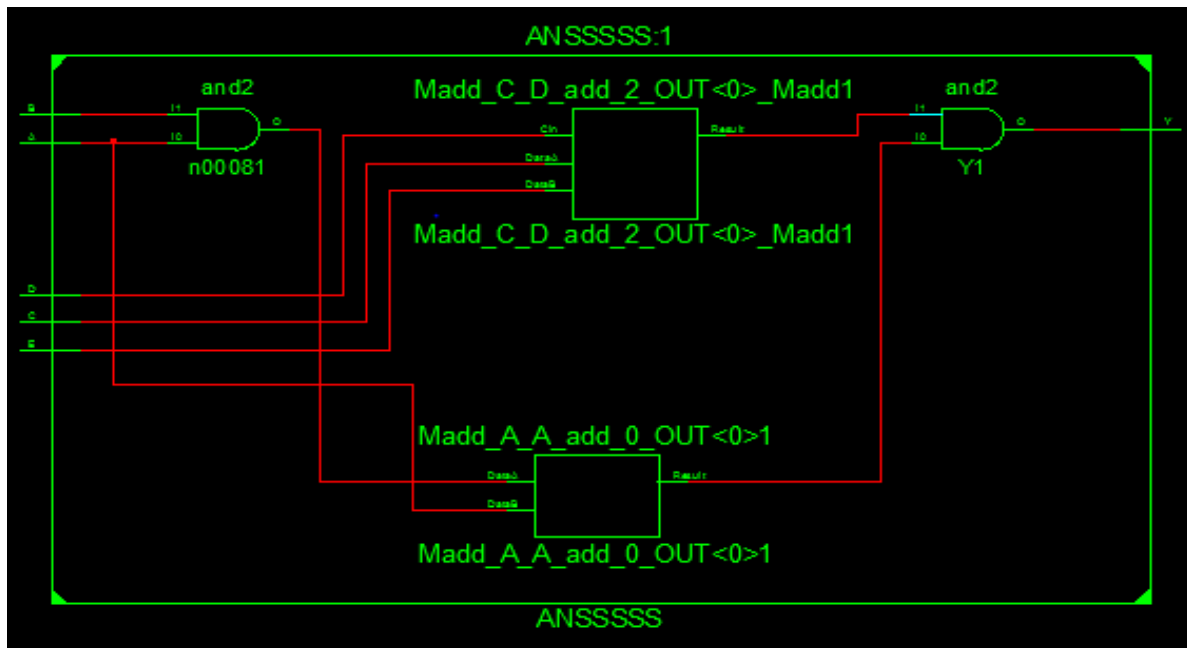
- Floorplan Area/IO/Logic (...)
- Synthesize - XST**
- View RTL Schematic
- View Technology Schematic
- Check Syntax
- Generate Post-Synthesis Si...
- Implement Design
- Generate Programming File

```

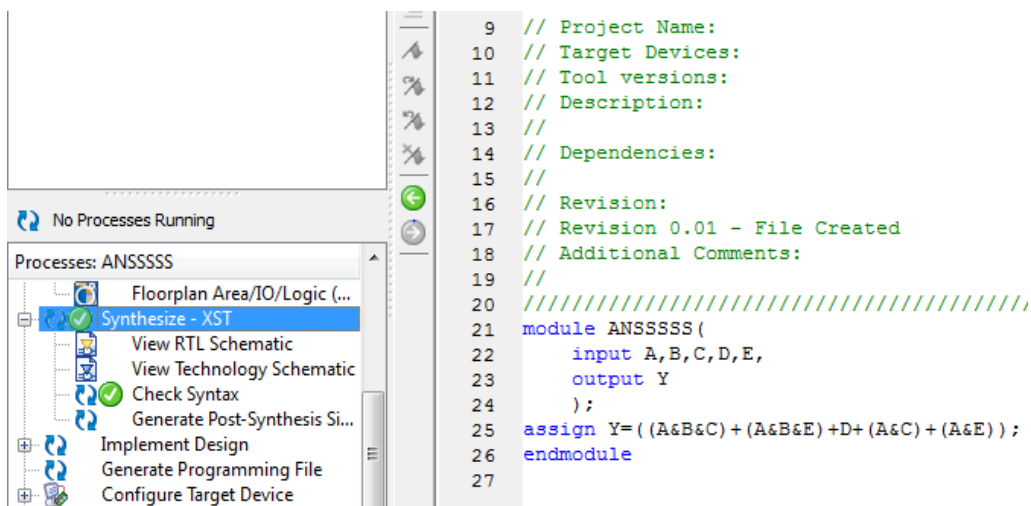
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////
21 module ANSSSSS(
22     input A,B,C,D,E,
23     output Y
24 );
25 assign Y= ( (A&B) +A) & ( (C+E) +D) ;
26 endmodule
27

```

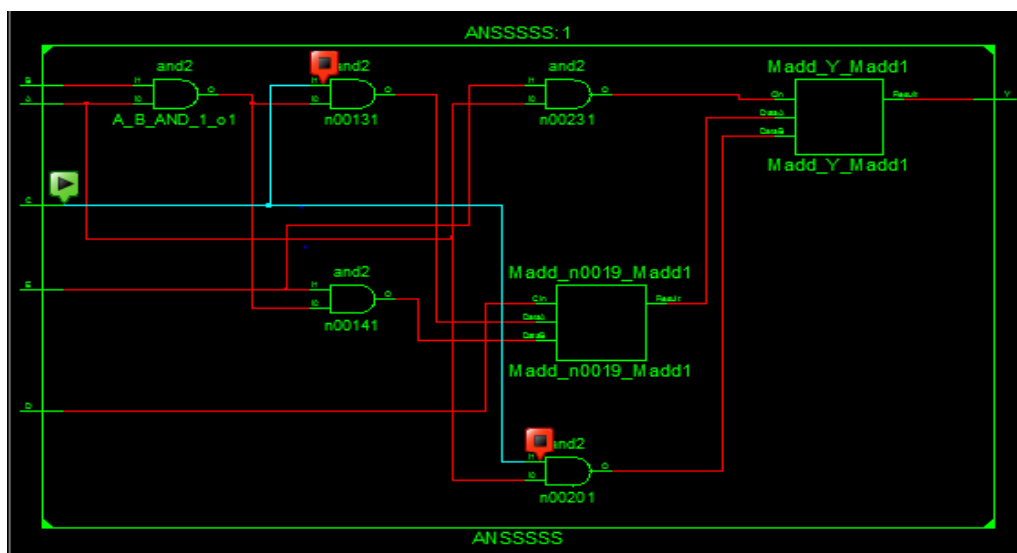
RTL Schematic -



- $Y = ABC + ABE + D + AC + AE$



RTL Schematic-



Data Sheet report:

* All values displayed in nanoseconds (ns)

Source Pad	Destination Pad	Delay
A	Y	6.382
B	Y	6.381
C	Y	6.500
D	Y	6.709
E	Y	6.952

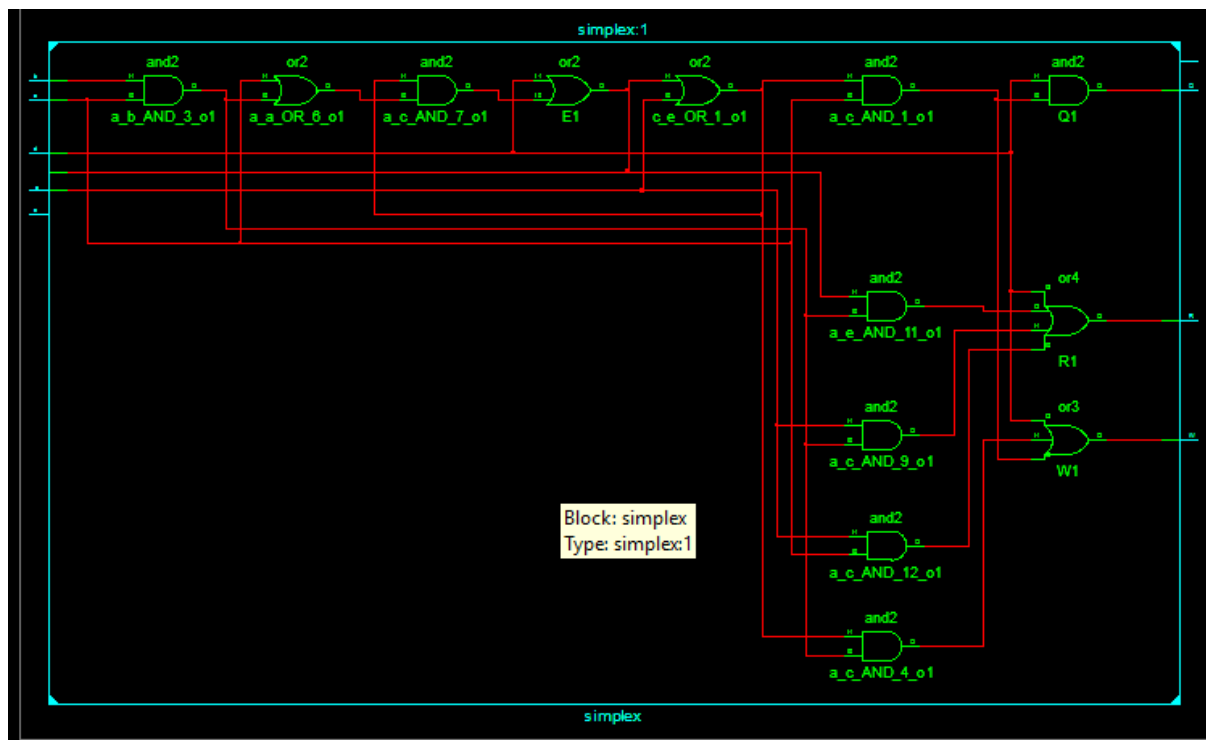
*Power leakage of above is 0.020 watt.

4.4. Write the verilog code to realize –

Logic Description for the required functionality-

```
2  ////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date:    14:41:36 01/11/2016
7  // Design Name:
8  // Module Name:    simplex
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 module simplex(
22     input a,
23     input b,
24     input c,
25     input d,
26     input e,
27     output Q,
28     output W,
29     output E,
30     output R
31 );
32 assign Q= a&(c|e)&d;
33 assign W= ((a&b)&(c|e)) | d | (a&(c|e)); assign E= (((a&b)|a)&(c|e)) | d ;
34 assign R= (a&b&c) | (a&b&e) | d | (a&c) ;
35
36 endmodule
37
```


RTL Schematic-



4.5. Write a verilog code to realize a 3 input Nand gate. The three input signal names are A, B, and C and output is F.

Logic Description for the module dlink-

hierarchy

- dlk
- xc6sbl6-3ftg256
- dlk (dlink.v)

Processes: dlink

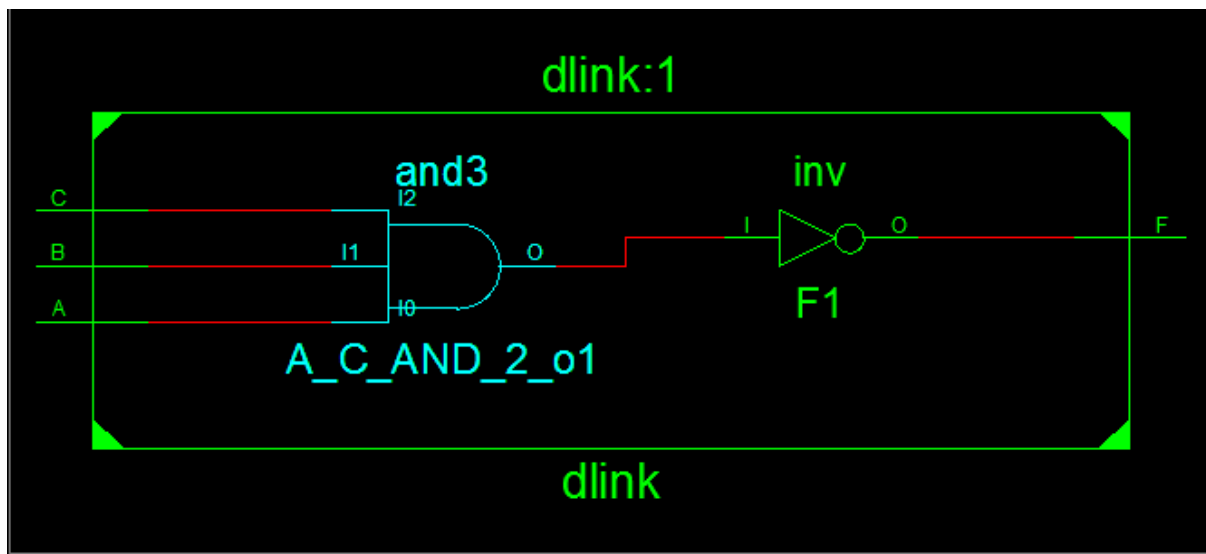
- Design Summary/Reports
- Design Utilities
- User Constraints
- Synthesize - XST
- View RTL Schematic
- View Technology Schematic
- Check Syntax
- Generate Post-Synthesis Si...

```

3 // Company:
4 // Engineer:
5 //
6 // Create Date:      11:52:57
7 // Design Name:
8 // Module Name:      dlink
9 // Project Name:
10 // Target Devices:
11 // Tool versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Crea
18 // Additional Comments:
19 //
20 ///////////////////////////////////////////////////
21 module dlink(
22     input A,B,C,
23     output F
24 );
25     assign F = ~(A&B&C);
26 endmodule
27

```

RTL Schematic-



Data Sheet report:

* All values displayed in nanoseconds (ns)

Source Pad	Destination Pad	Delay
A	F	5.860
B	F	6.009
C	F	5.953

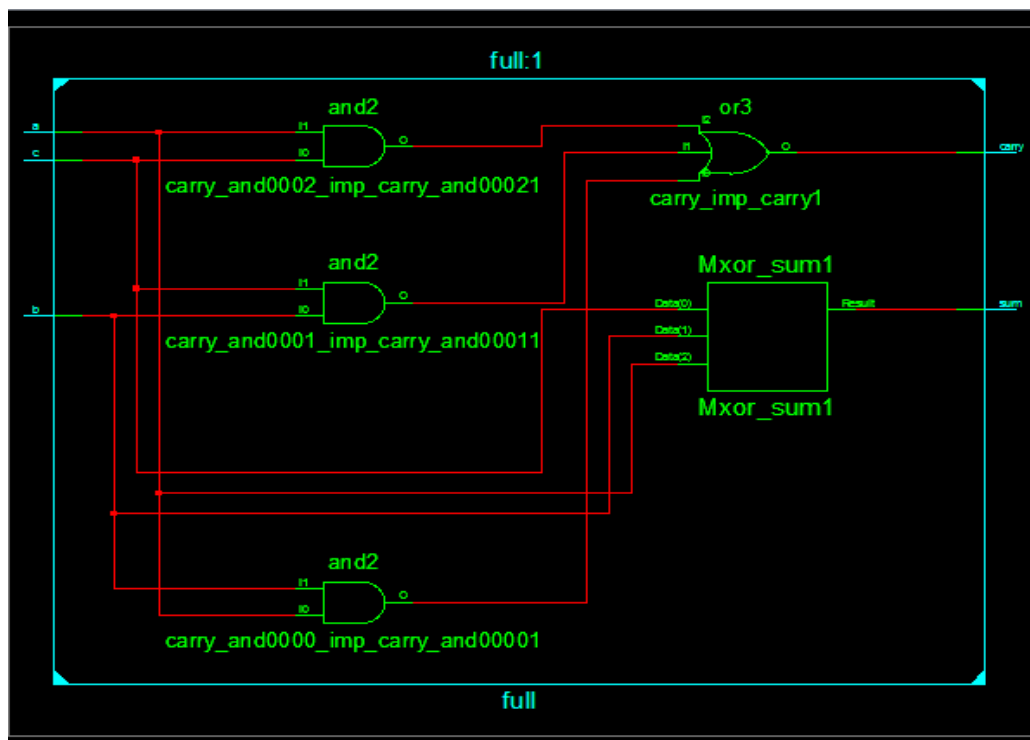
*Power leakage of above function is 0.020 watt.

5. DESIGNS & ANALYSIS OF CIRCUITS

5.1. RIPPLE CARRY ADDER

```
3 // Company:
4 // Engineer:
5 //
6 // Create Date:    16:07:15 06/15/2015
7 // Design Name:
8 // Module Name:    Ripple
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 module ripple_adder_4bit(
22     output [3:0] sum,
23     output Cout,
24     input [3:0] A,
25     input [3:0] B,
26     input carry
27 );
28     wire c1,c2 ,c3;
29     full FA1(A[0],B[0],carry, sum[0],c1);
30
31     full FA2(A[1],B[1],c1,sum[1],c2);
32     full FA3(A[2],B[2],c2,sum[2],c3);
33     full FA4(A[3],B[3],c3,sum[3],Cout);
34
35 endmodule
36
```

RTL Schematic-
Ripple.v



Test Bench of Ripple Carry Adder-

```

2  module ripple_bit;
3  // Inputs
4  reg [3:0] A;
5  reg [3:0] B;
6  reg carry;
7
8  // Outputs
9  wire [3:0] sum;
10 wire Cout;
11 // Instantiate the Unit Under Test (UUT)
12 ripple_adder_4bit uut (
13     .sum(sum),
14     .Cout(Cout),
15     .A(A),
16     .B(B),
17     .carry(carry)
18 );
19 initial begin
20     // Initialize Inputs
21     // Wait 100 ns for global reset to finish
22     A[0]=0;
23     B[0]=1;
24     carry=0;
25     #100;
26
27     A[1]=1;
28     B[1]=1;
29     #100;
30     A[2]=0;
31     B[2]=1;
32     #100;
33     A[3]=1;
34     B[3]=1;
35     #100;
36     // Add stimulus here
37 end
38 endmodule

```

Data Sheet report:

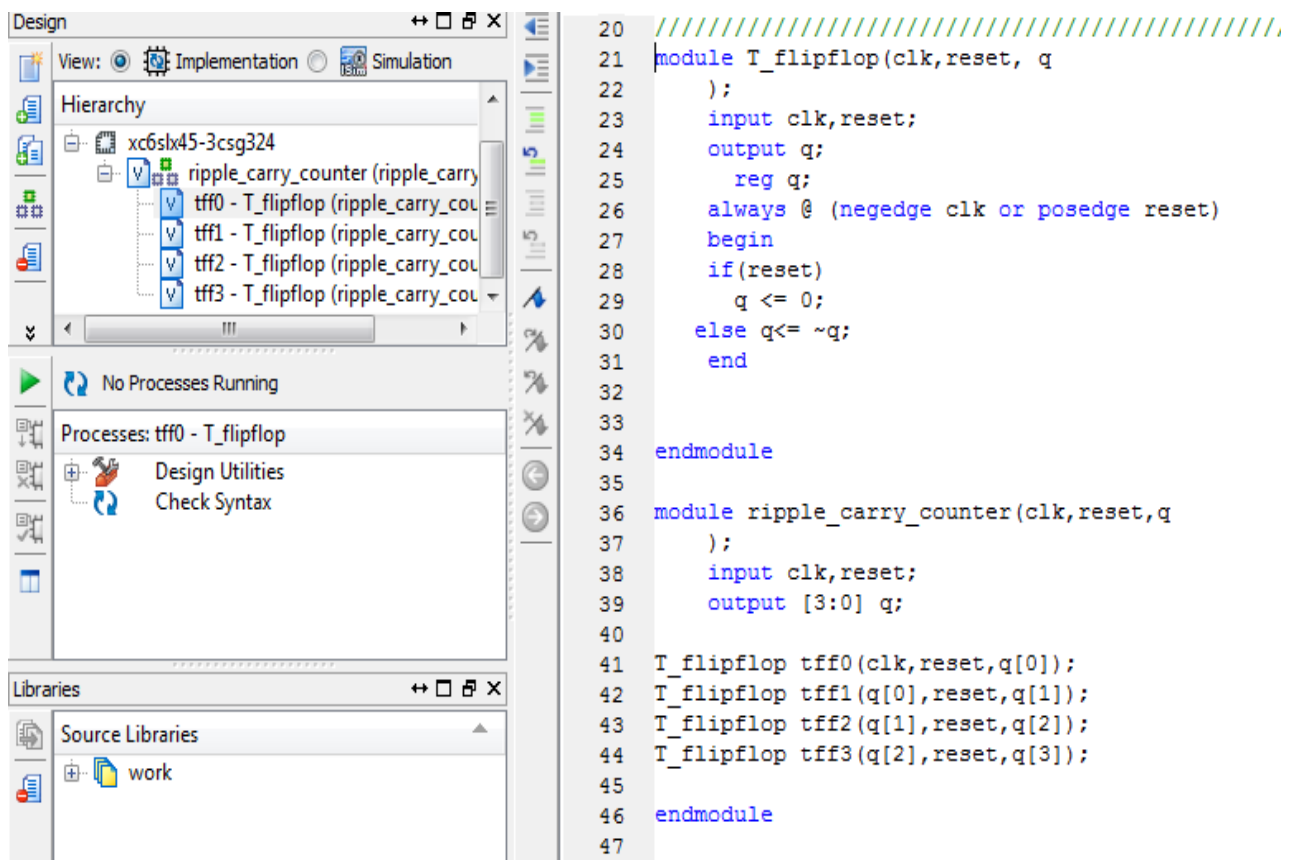
*All values displayed in nanoseconds (ns)

Source Pad	Destination Pad	Delay
A<0>	cout	4.493
A<0>	sum<0>	3.687
A<0>	sum<1>	3.837
A<0>	sum<2>	3.623
A<0>	sum<3>	4.355
A<1>	cout	4.717
A<1>	sum<1>	4.077
A<1>	sum<2>	3.851
A<1>	sum<3>	4.579
A<2>	cout	4.058
A<2>	sum<2>	3.993
A<2>	sum<3>	3.914
A<3>	cout	4.204
A<3>	sum<3>	4.070
B<0>	cout	4.424

B<0>	sum<0>	3.672
B<0>	sum<1>	3.776
B<0>	sum<2>	3.609
B<0>	sum<3>	4.286
B<1>	cout	4.605
B<1>	sum<1>	3.951
B<1>	sum<2>	3.814
B<1>	sum<3>	4.467
B<2>	cout	4.022
B<2>	sum<2>	3.998
B<2>	sum<3>	3.876
B<3>	cout	4.201
B<3>	sum<3>	4.071

*Power leakage of above Ripple Carry Adder is 0.42 watt.

5.2. RIPPLE CARRY COUNTER



The screenshot displays a Verilog HDL editor interface. On the left, the 'Design' pane shows the hierarchy of the project 'xc6slx45-3csg324'. It includes a 'ripple_carry_counter' module which contains four instances of 'T_flipflop' (tff0, tff1, tff2, tff3). Below this, the 'Processes' pane shows 'tff0 - T_flipflop' and 'Design Utilities' with 'Check Syntax' selected. The 'Libraries' pane shows 'Source Libraries' with a 'work' library.

The right pane shows the Verilog code for the counter. It starts with a comment line (20) and defines the 'T_flipflop' module (21-34). The 'T_flipflop' module has inputs 'clk' and 'reset', and an output 'q'. It uses a 'reg q' and an 'always' block triggered by 'negedge clk or posedge reset' to implement a D flip-flop. The 'endmodule' for 'T_flipflop' is at line 34.

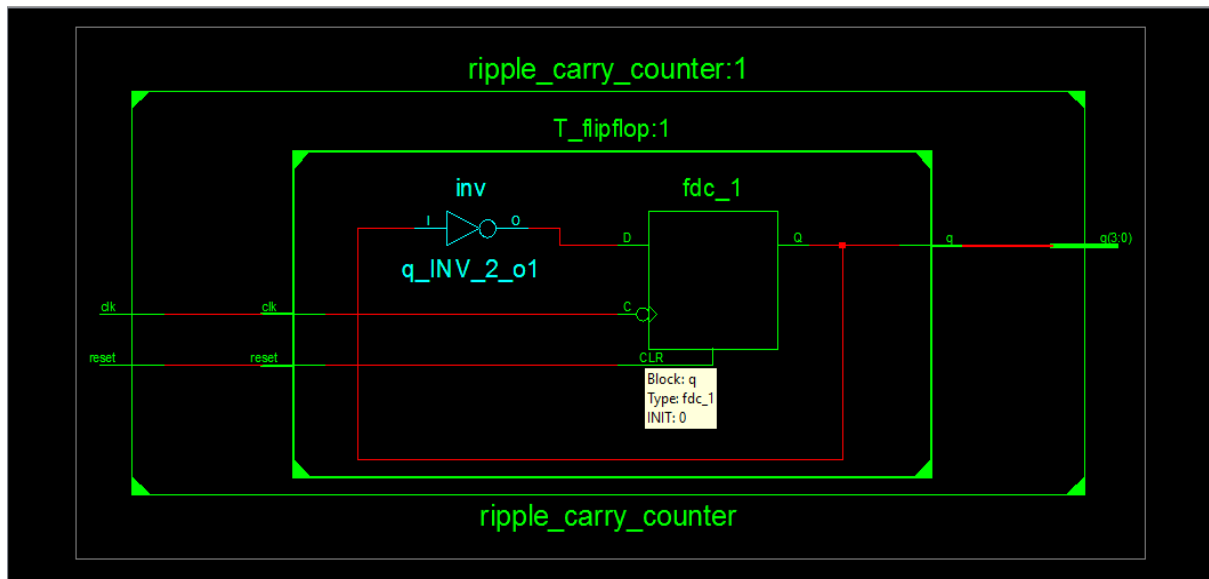
Next, the 'ripple_carry_counter' module is defined (36-46). It has inputs 'clk' and 'reset', and a 4-bit output 'q' (3:0). It instantiates four 'T_flipflop' modules: 'tff0' (41), 'tff1' (42), 'tff2' (43), and 'tff3' (44). The 'endmodule' for 'ripple_carry_counter' is at line 46.

```

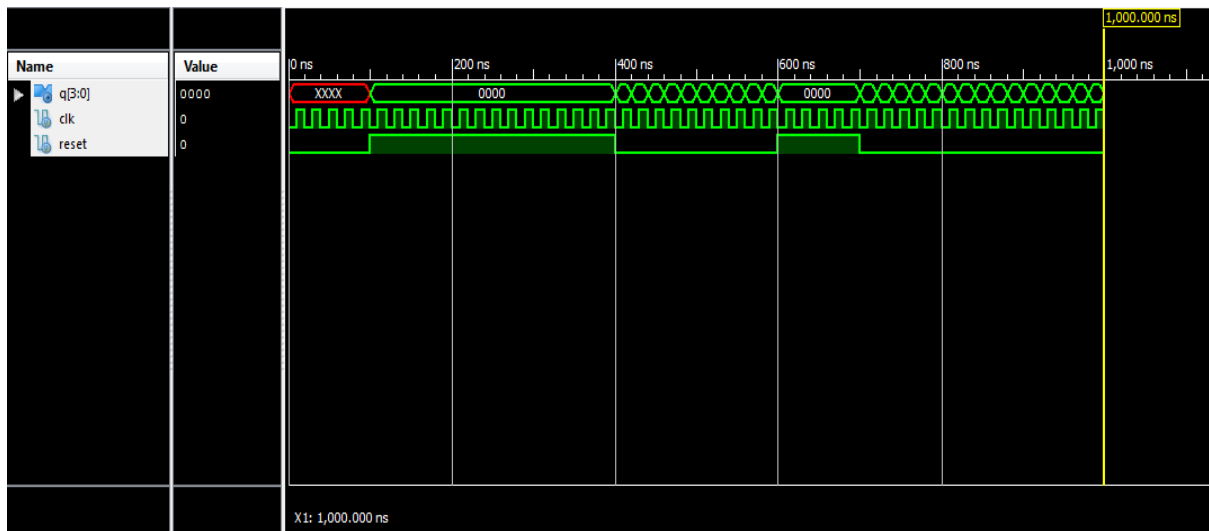
20 ///////////////////////////////////////////////////
21 module T_flipflop(clk,reset, q
22 );
23     input clk,reset;
24     output q;
25     reg q;
26     always @ (negedge clk or posedge reset)
27     begin
28         if(reset)
29             q <= 0;
30         else q<= ~q;
31     end
32
33
34 endmodule
35
36 module ripple_carry_counter(clk,reset,q
37 );
38     input clk,reset;
39     output [3:0] q;
40
41     T_flipflop tff0(clk,reset,q[0]);
42     T_flipflop tff1(q[0],reset,q[1]);
43     T_flipflop tff2(q[1],reset,q[2]);
44     T_flipflop tff3(q[2],reset,q[3]);
45
46 endmodule
47

```

RTL Schematic-



Waveform Generated-



Data Sheet report:

*All values displayed in nanoseconds (ns)

Timing constraint: Default period analysis for Clock 'tff0/q'

Clock period: 1.984ns (frequency: 504.007MHz)

Total number of paths / destination ports: 1 / 1

Delay: 1.984ns (Levels of Logic = 1)

Source: tff1/q (FF)

Destination: tff1/q (FF)

Source Clock: tff0/q falling

Destination Clock: tff0/q falling

Data Path: tff1/q to tff1/q

Cell:in->out	fanout	Gate Delay	Net Delay	Logical Name (Net Name)
FDC_1:C->Q	3	0.447	0.650	tff1/q (tff1/q)
INV:I->O	1	0.206	0.579	tff1/q_INV_2_o1_INV_0 (tff1/q_INV_2_o)
FDC_1:D		0.102	tff1/q	

Total		1.984ns	(0.755ns logic, 1.229ns route) (38.1% logic, 61.9% route)	

5.3. Asynchronous D Flip-Flop

Data Sheet report:

*All values displayed in nanoseconds (ns)

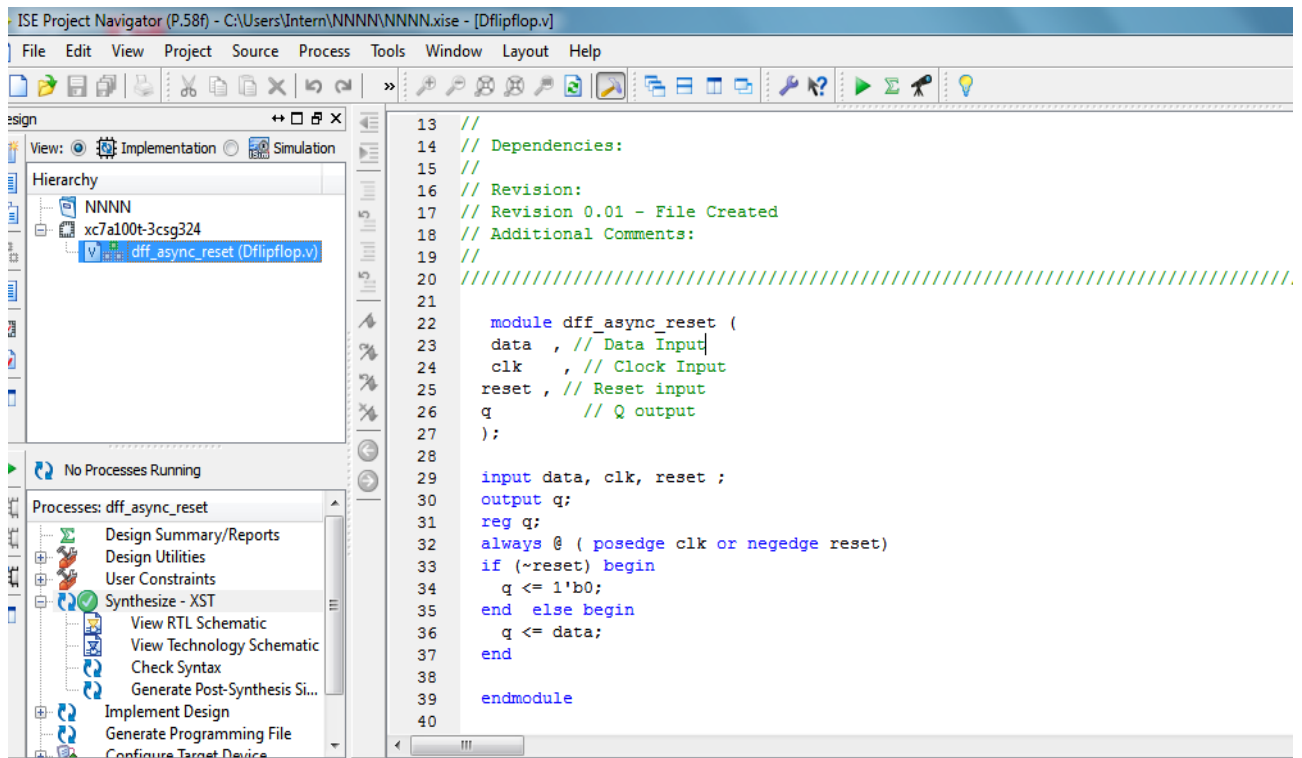
Setup/Hold to clock clk

Max Setup to		Process		Max Hold to		Process	
Clock	Source	clk(edge)		Corner		clk (edge)	
Corner	Internal Clock(s)	Phase	data				
-3.350(R)	FAST	3.862(R)	SLOW	clk_BUFGP		0.000	
reset	-2.151(R)	SLOW	2.854(R)	FAST			
clk_BUFGP	0.000						

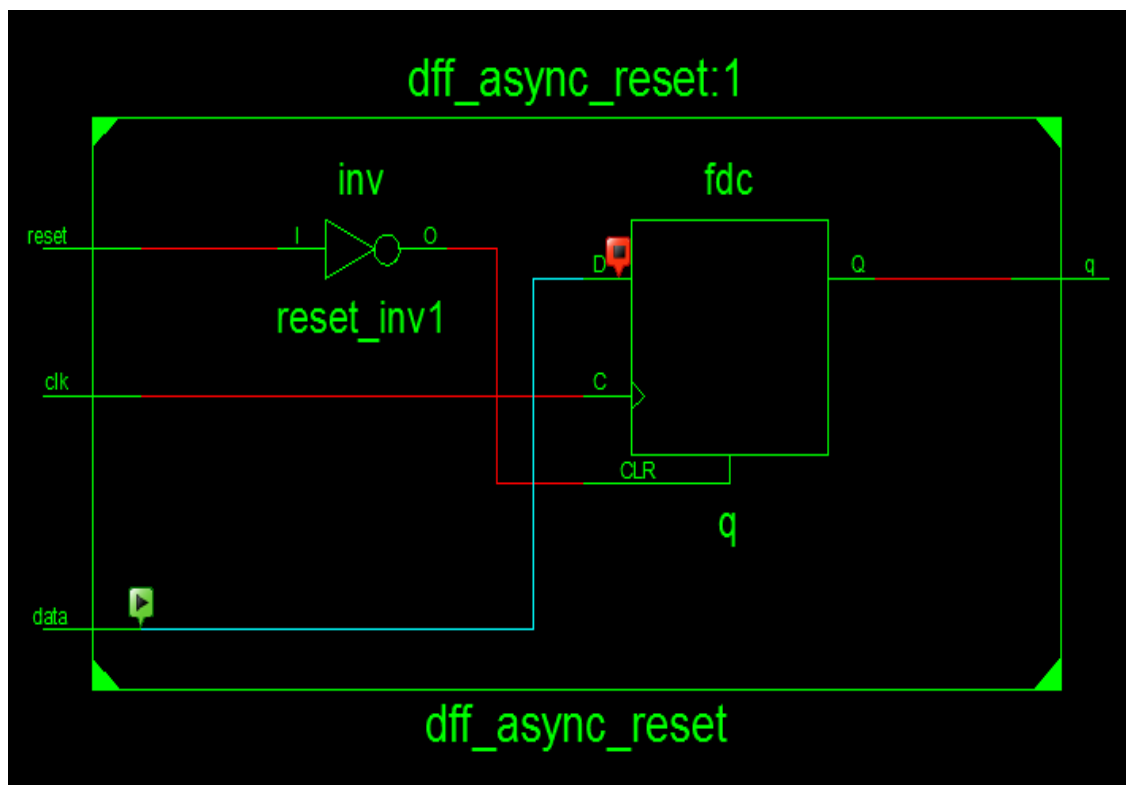
*Power leakage of above function is 0.042 watt.

Clock clk to Pad

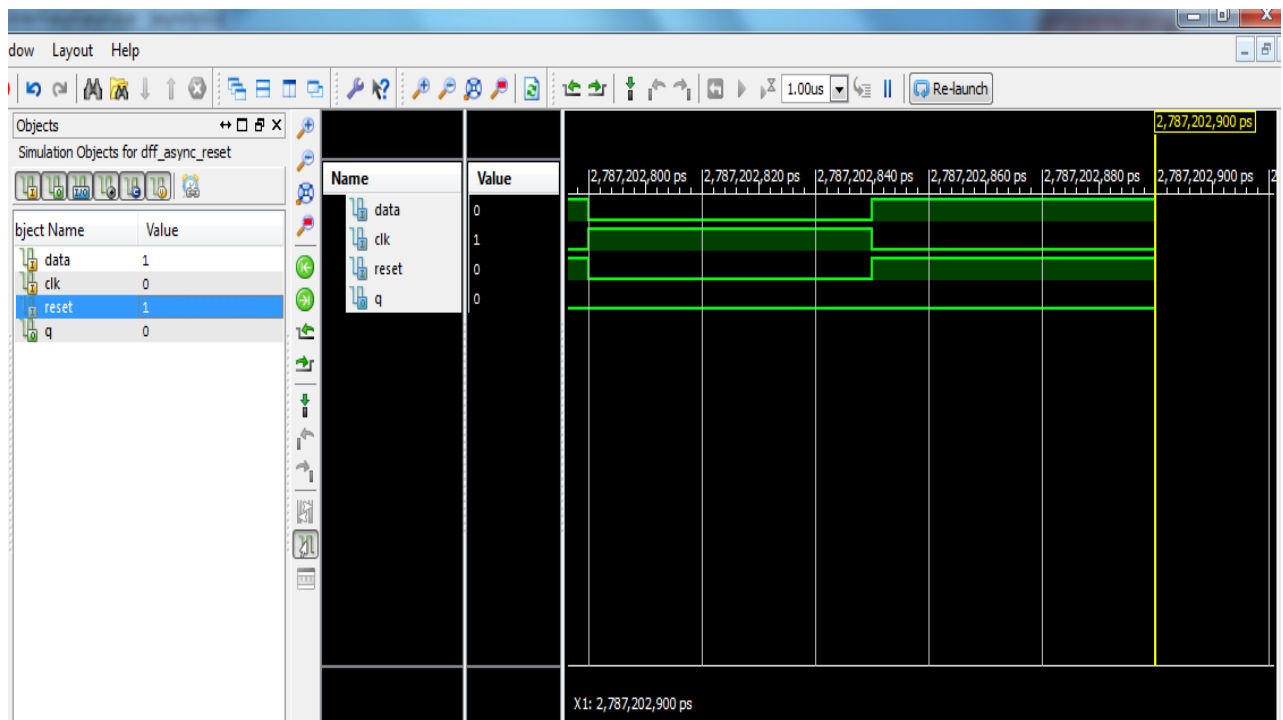
Destinatio n	Max (slowest) clk (edge) to PAD	Proces s Corner	Min(fastest)cl k (edge) to PAD	Proces s corner	Internal clk(s)	Cloc k Phase
Q	7.437(R)	SLOW	5.402(R)	FAST	clk_BUFG P	0.000



RTL Schematic-



Waveform Generated-

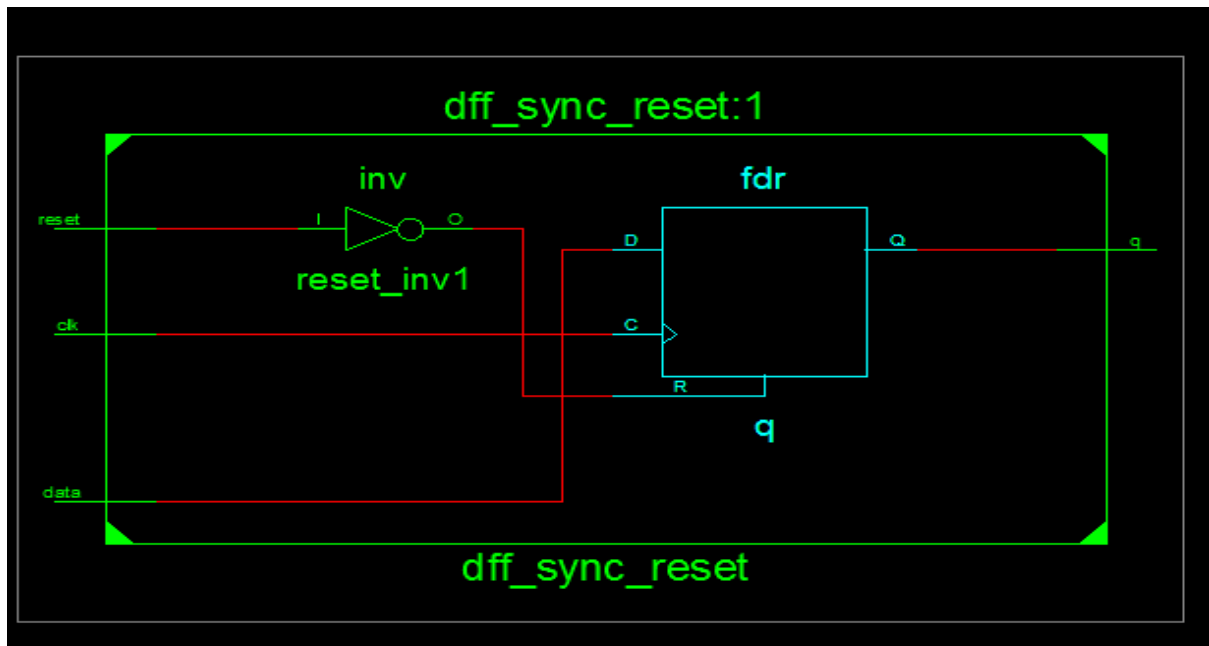


5.4. Synchronous D Flip Flop

```

13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 ///////////////////////////////////////////////////////////////////
21
22 module dff_sync_reset (
23     data , // Data Input
24     clk   , // Clock Input
25     reset , // Reset input
26     q      // Q output
27 );
28
29 input data, clk, reset ;
30 output q;
31 //-----Internal Variables-----
32 reg q;
33 //-----Code Starts Here-----
34 always @ ( posedge clk)
35 if (~reset) begin
36     q <= 1'b0;
37 end else begin
38     q <= data;
39 end
40 endmodule //End Of Module dff_sync_reset

```



Data Sheet report:

* All values displayed in nanoseconds (ns)

- Setup/Hold to clock clk

Max Setup to		Process	Max Hold to		Process
Clock	Source	clk(edge)	Corner	clk(edge)	
Corner	Internal Clock(s)	Phase	data		

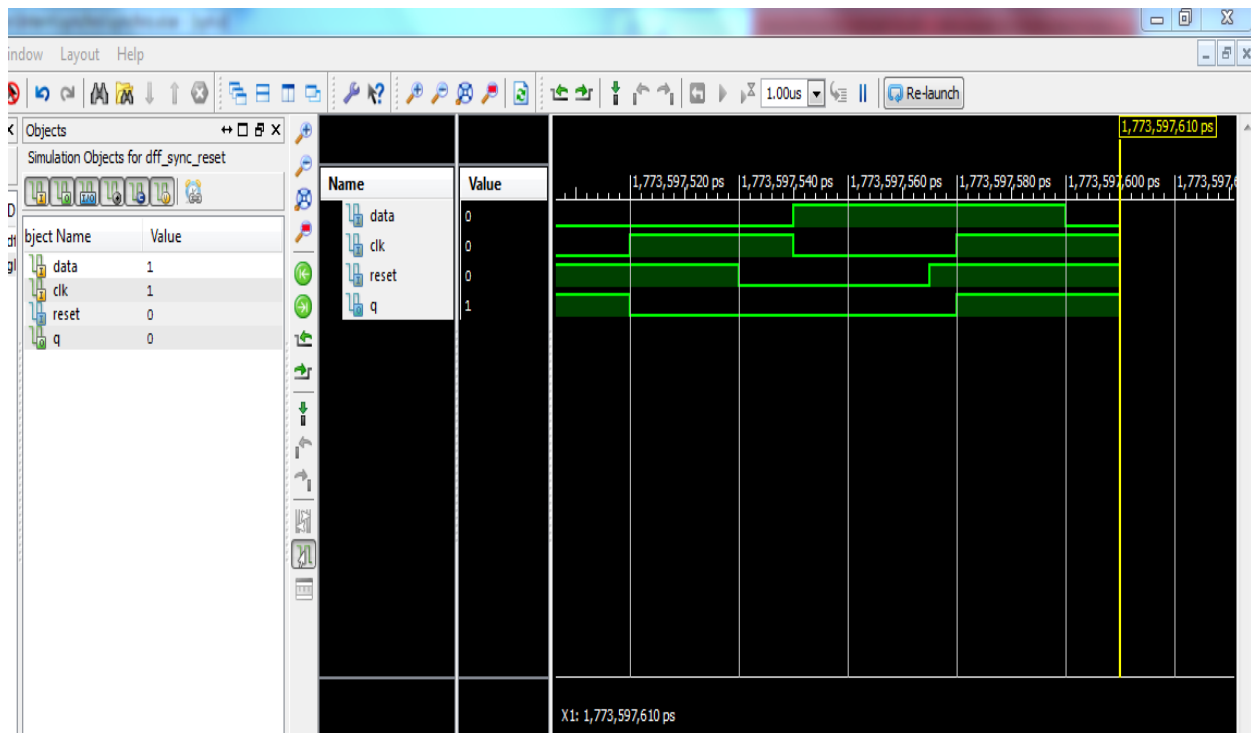
-3.350(R)		FAST		3.862(R)	SLOW
clk_BUFGP	0.000	reset		-2.135(R)	SLOW
3.422(R)	FAST	clk_BUFGP		0.000	

*Power leakage of above function is 0.042 watt.

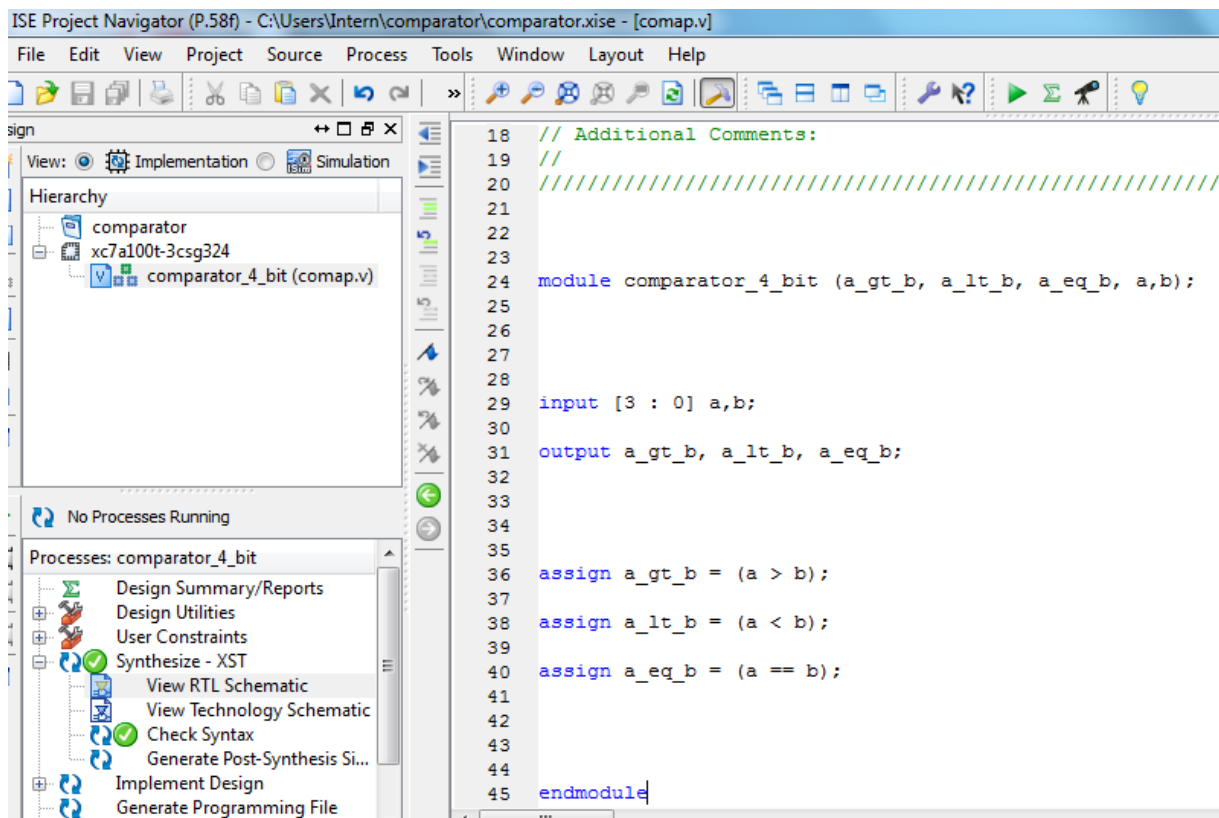
- Clock clk to Pad

Destination	Max (slowest) clk (edge) to PAD	Process Corner	Min(fastest)clk (edge) to PAD	Process corner	Internal clk(s)	Clock Phase
Q	7.437(R)	SLOW	5.402(R)	FAST	clk_BUFGP	0.000

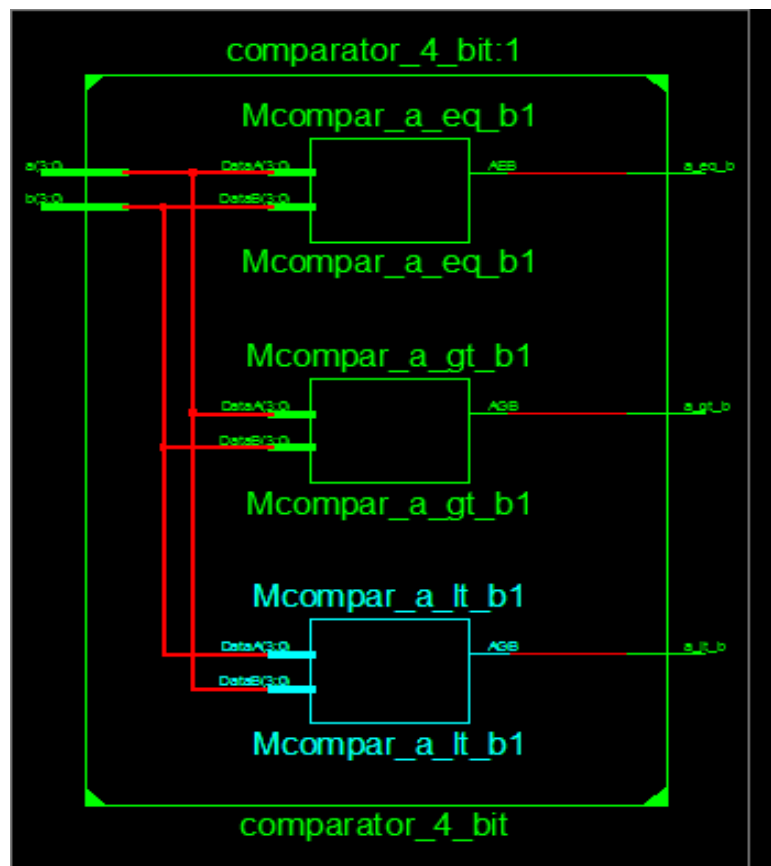
Waveform Generated-



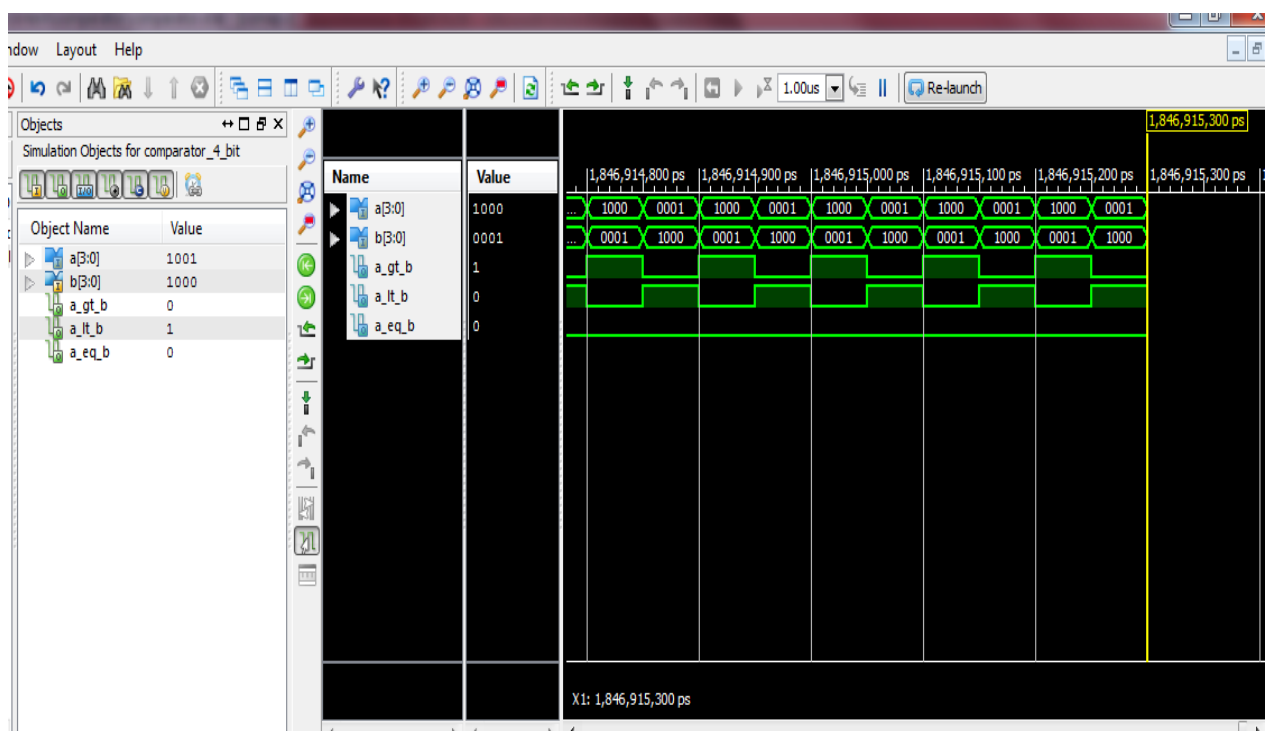
5.5. Magnitude Comparator (4 Bit)



RTL Schematic-

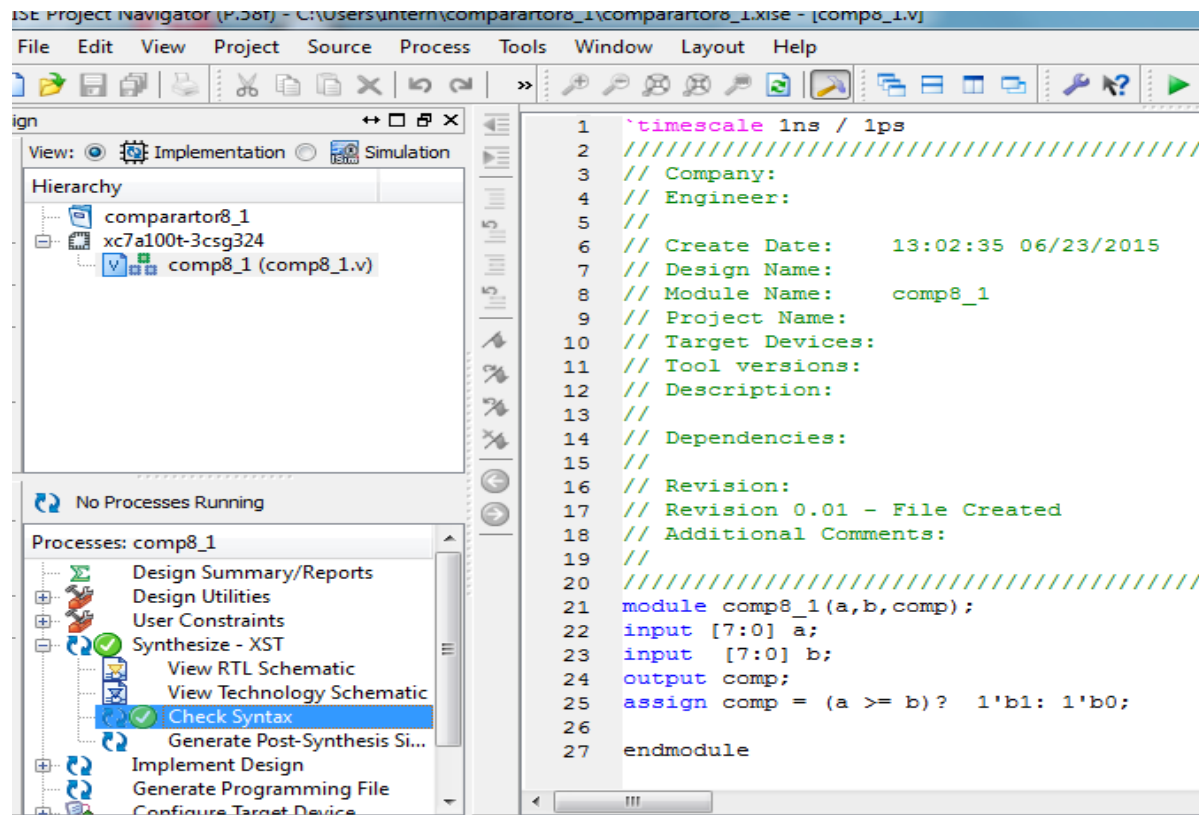


Waveform Generated-

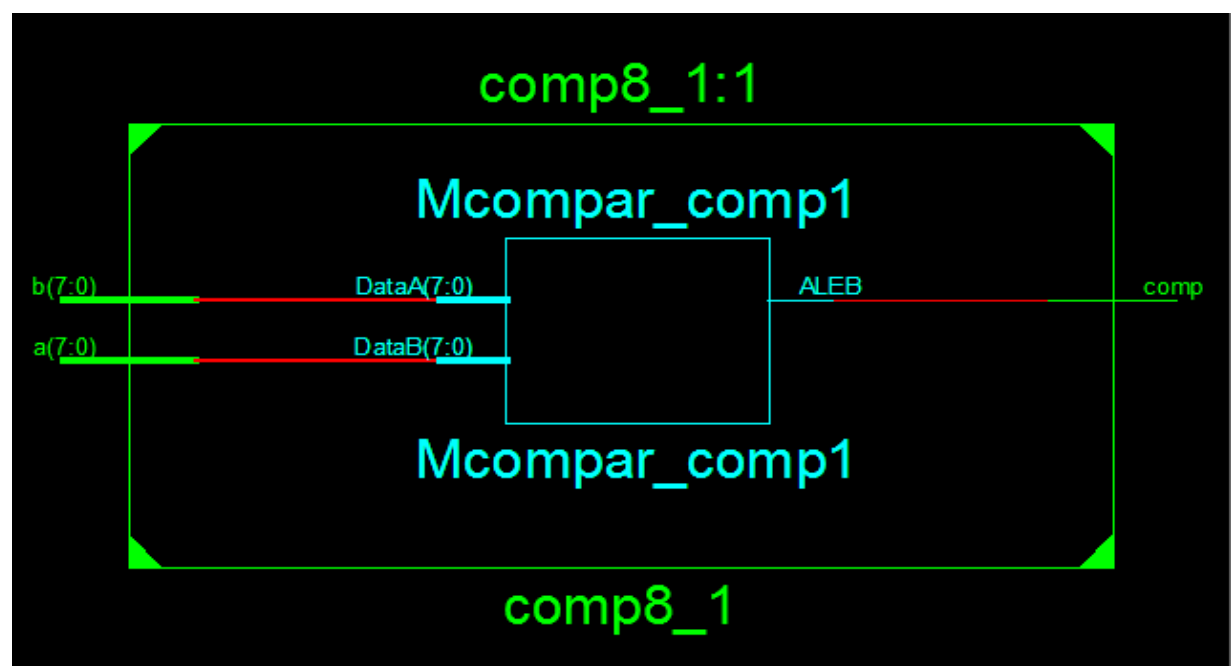


Magnitude Comparator (8 Bit)

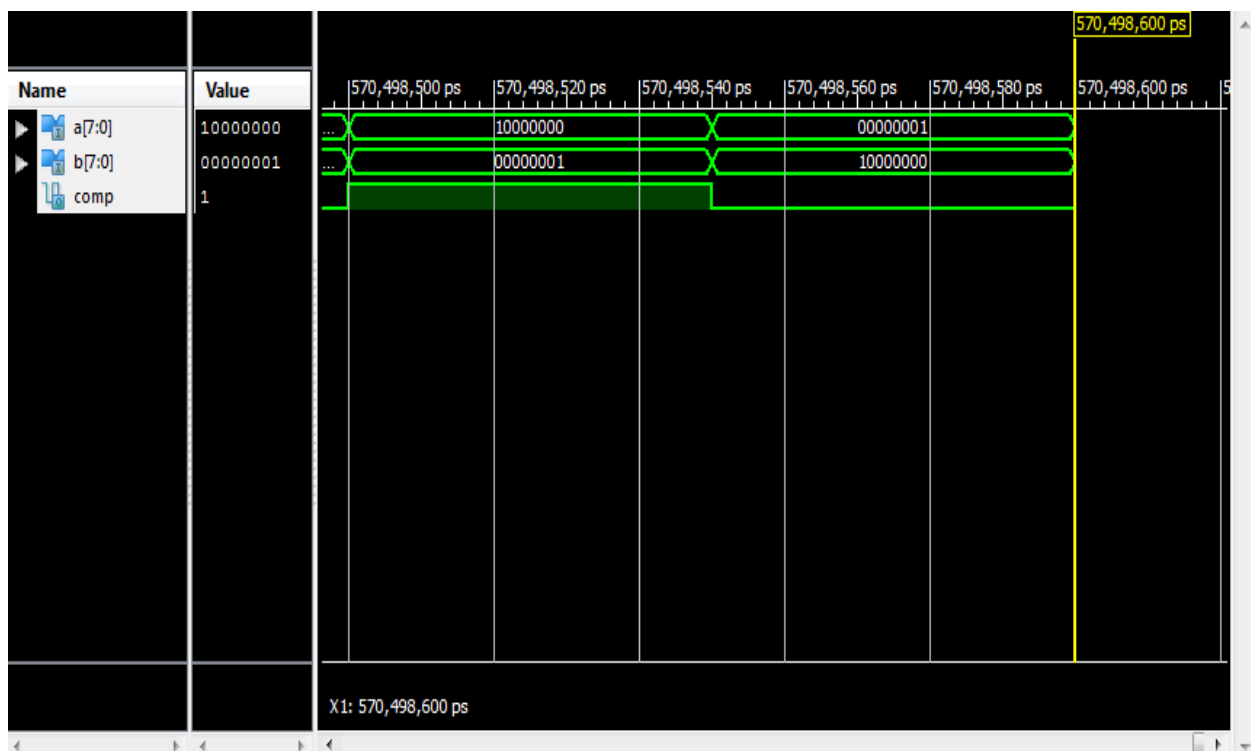
Required Logic Description for the 8-bit comparator-



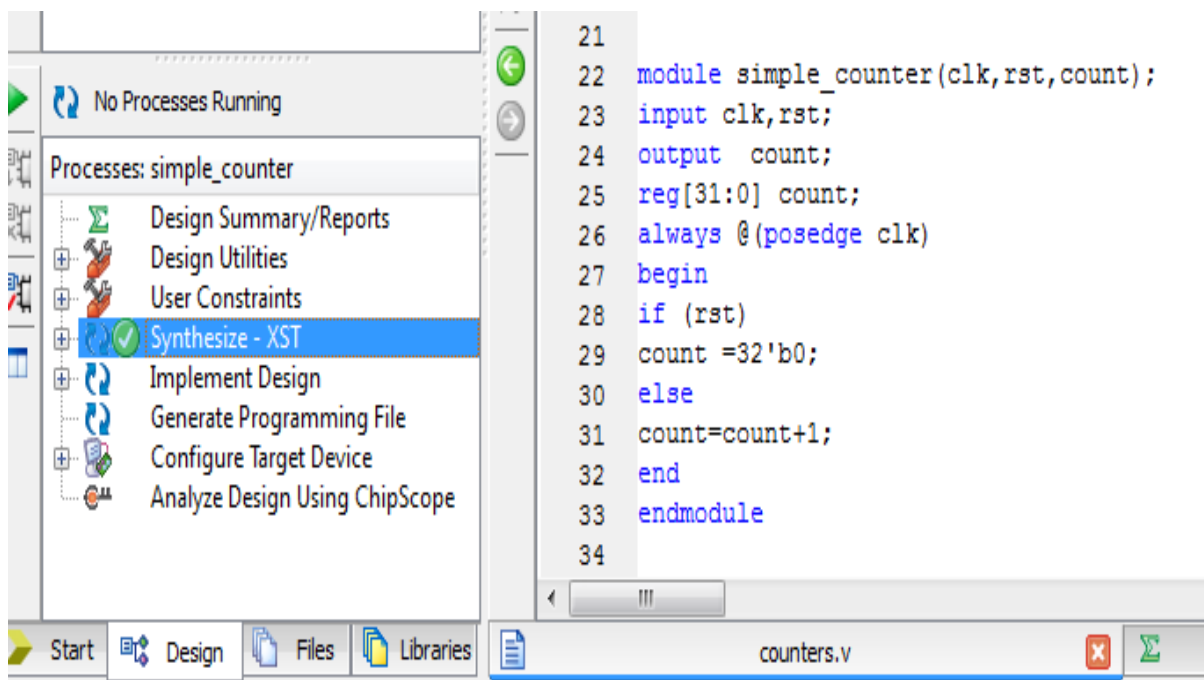
RTL Schematic-



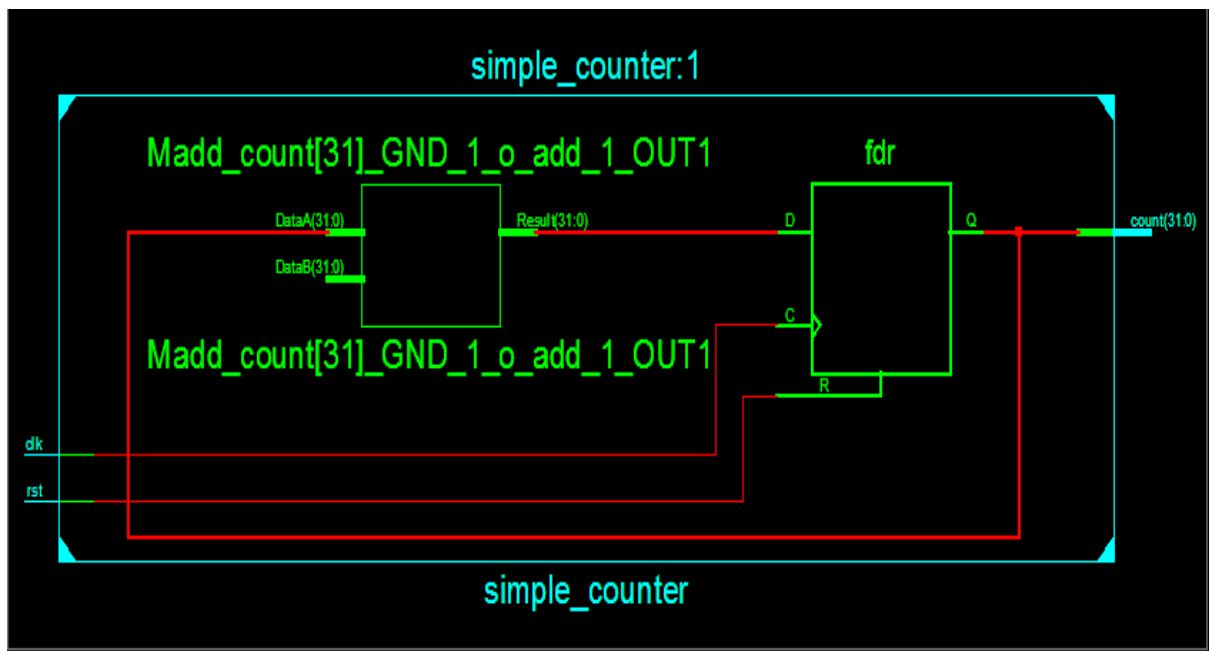
Waveform Generated-



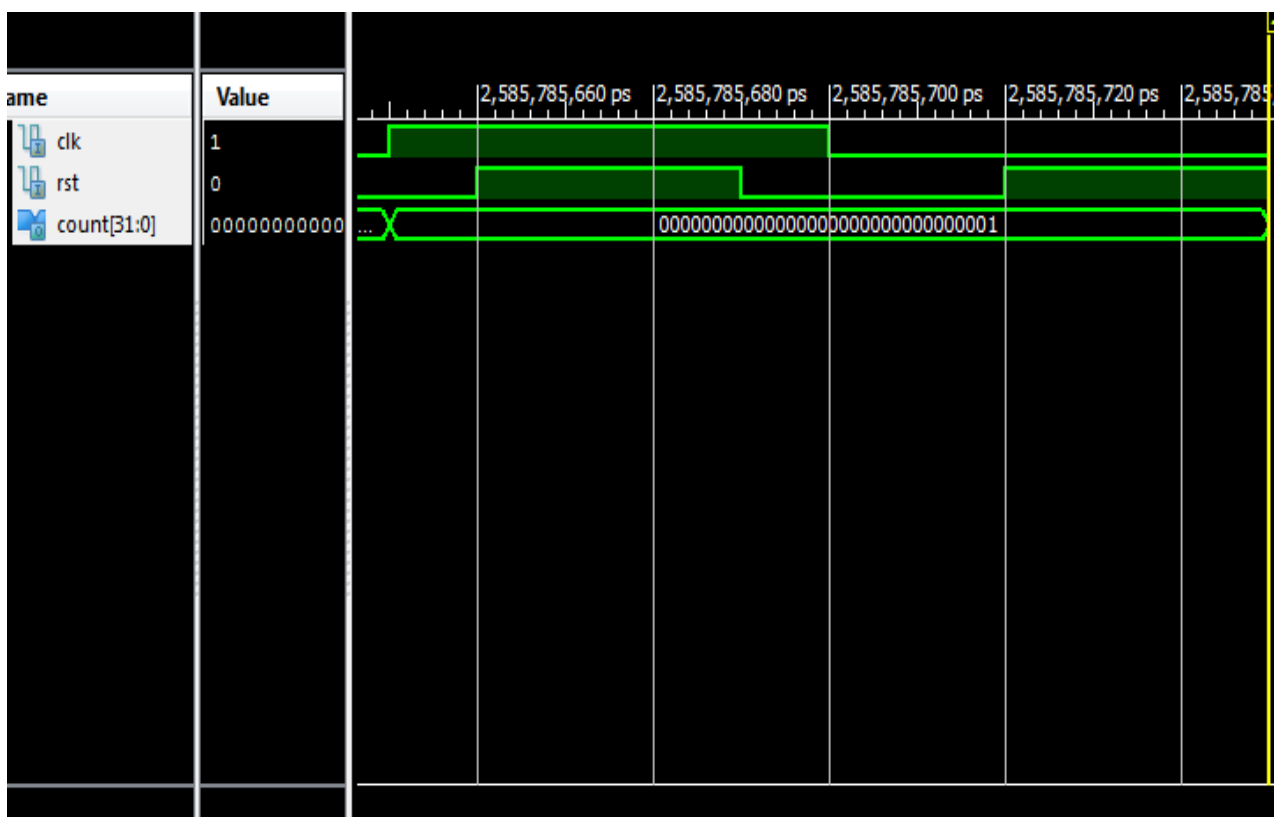
Counters



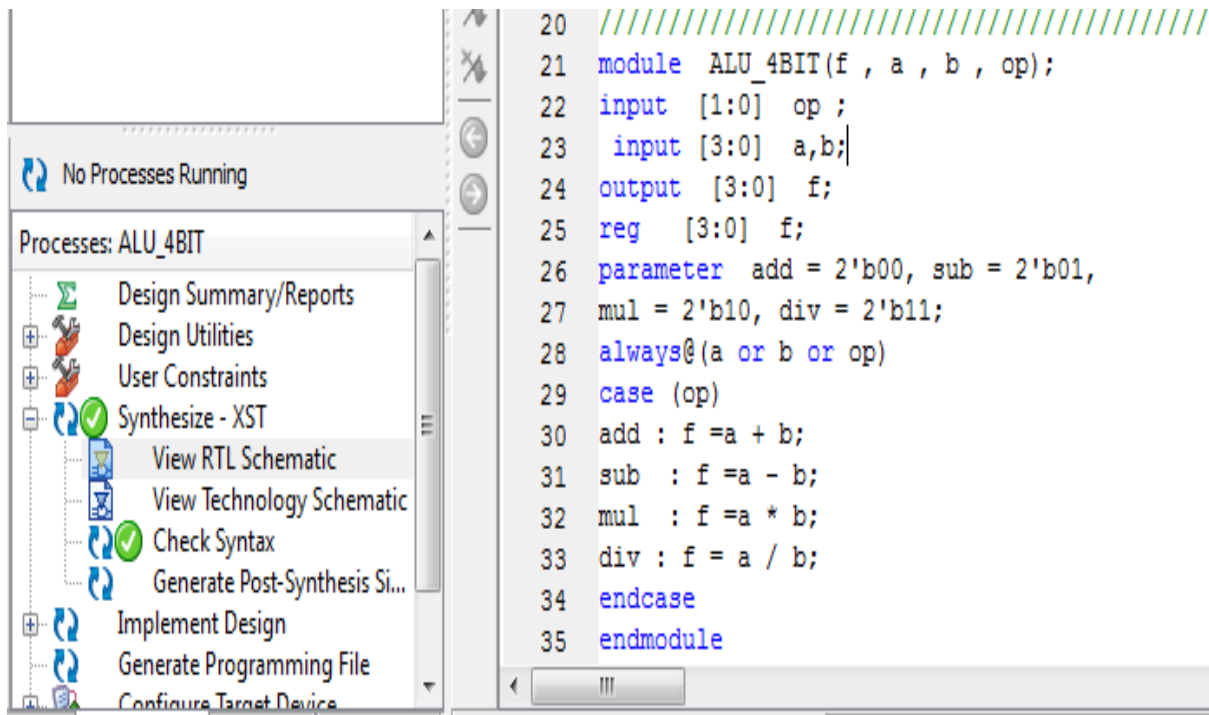
RTL Schematic-



Waveform Generated-



5.6. Arithmetic Logic Unit

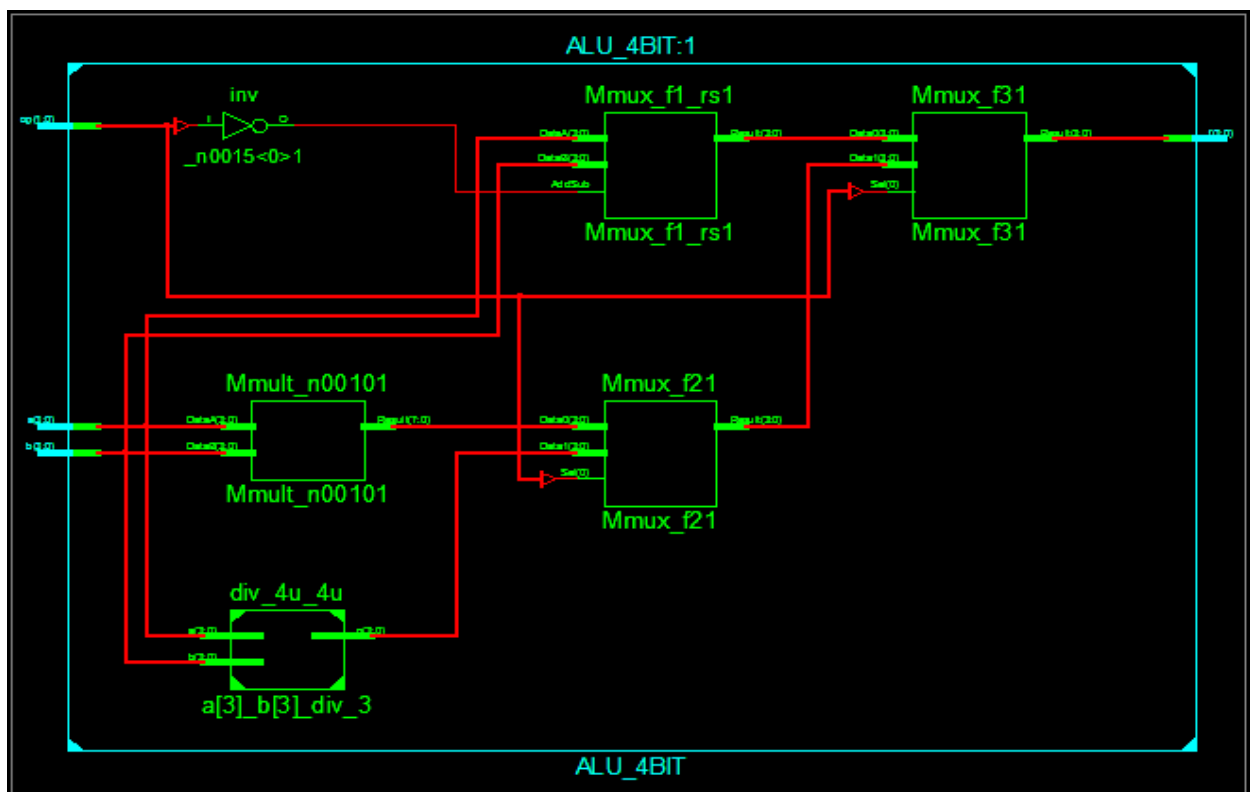


```

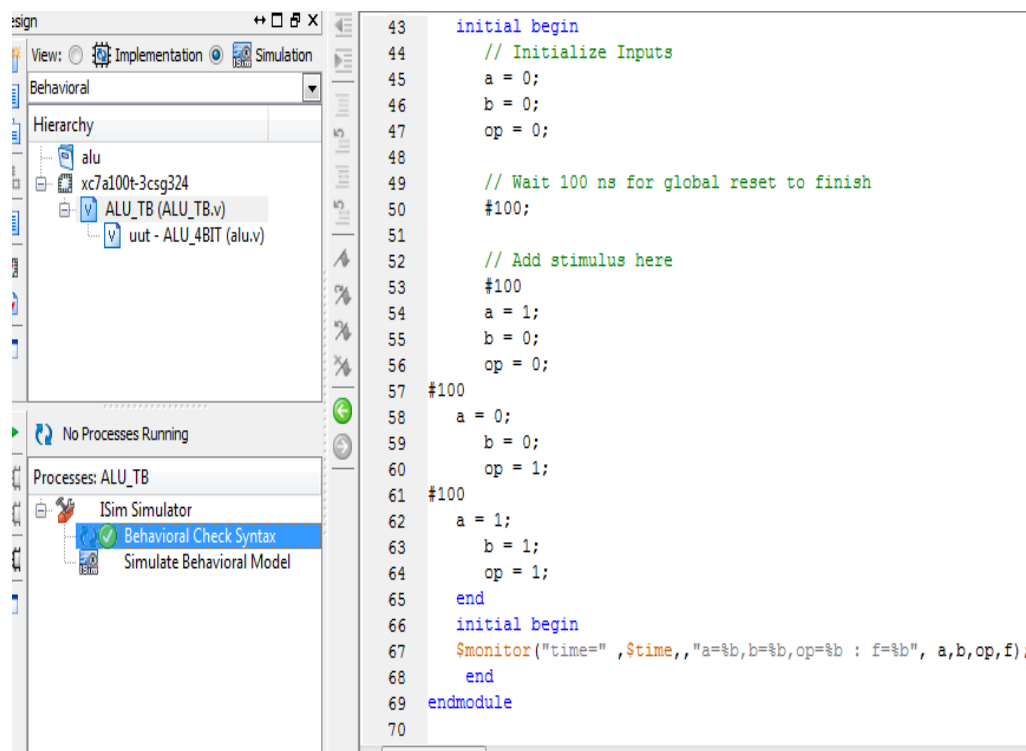
20 //////////////////////////////////////////////////
21 module ALU_4BIT(f , a , b , op);
22 input [1:0] op ;
23 input [3:0] a,b;
24 output [3:0] f;
25 reg [3:0] f;
26 parameter add = 2'b00, sub = 2'b01,
27 mul = 2'b10, div = 2'b11;
28 always@(a or b or op)
29 case (op)
30 add : f = a + b;
31 sub : f = a - b;
32 mul : f = a * b;
33 div : f = a / b;
34 endcase
35 endmodule

```

RTL Schematic-



Test Bench for Arithmetic Logic Unit-

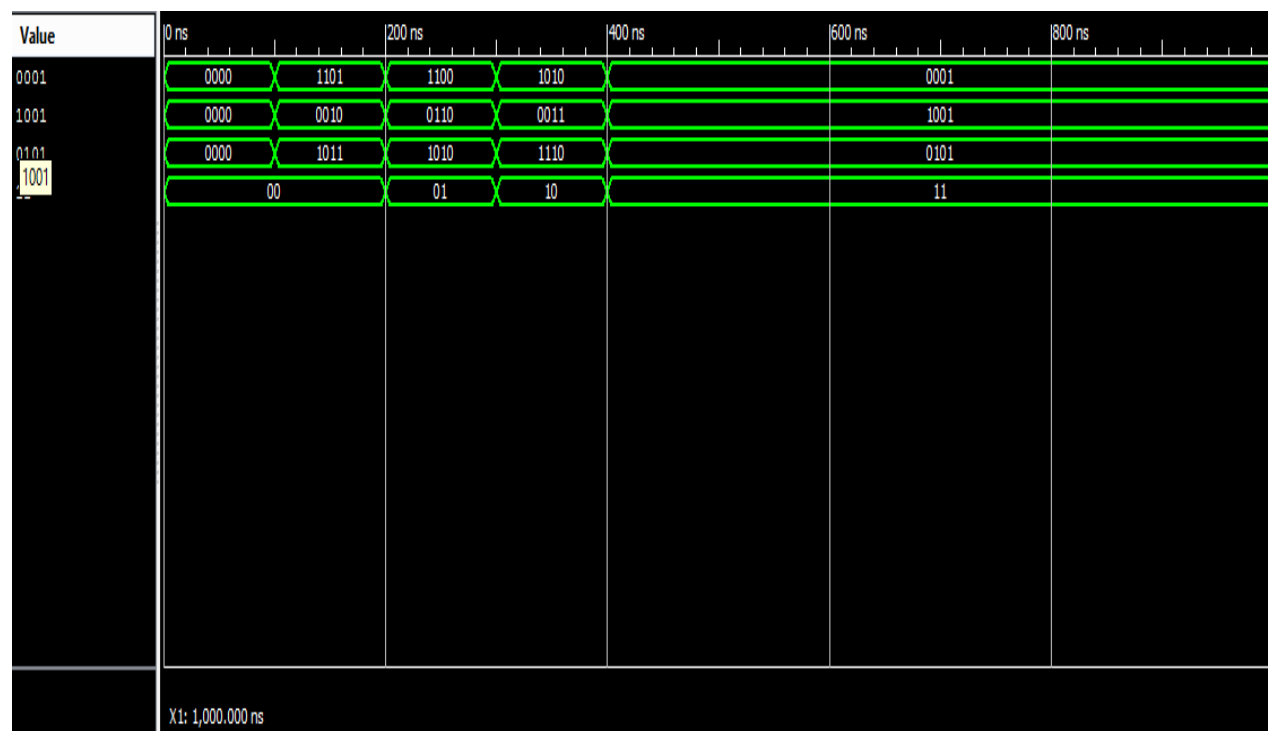


```

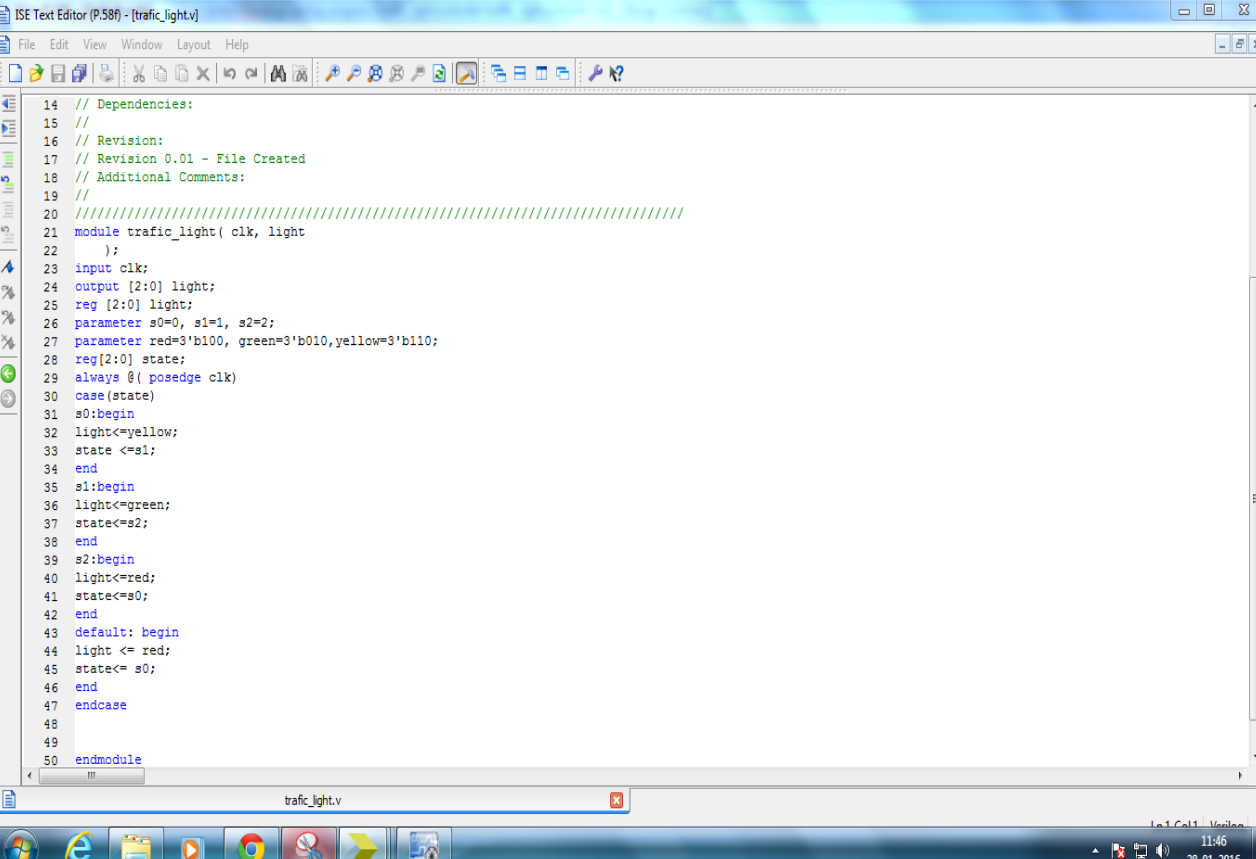
43  initial begin
44      // Initialize Inputs
45      a = 0;
46      b = 0;
47      op = 0;
48
49      // Wait 100 ns for global reset to finish
50      #100;
51
52      // Add stimulus here
53      #100
54      a = 1;
55      b = 0;
56      op = 0;
57  #100
58      a = 0;
59      b = 0;
60      op = 1;
61  #100
62      a = 1;
63      b = 1;
64      op = 1;
65  end
66  initial begin
67      $monitor("time=", $time, "a=%b,b=%b,op=%b : f=%b", a,b,op,f);
68  end
69  endmodule
70

```

Waveform Generated-



5.7. Traffic Light Controller

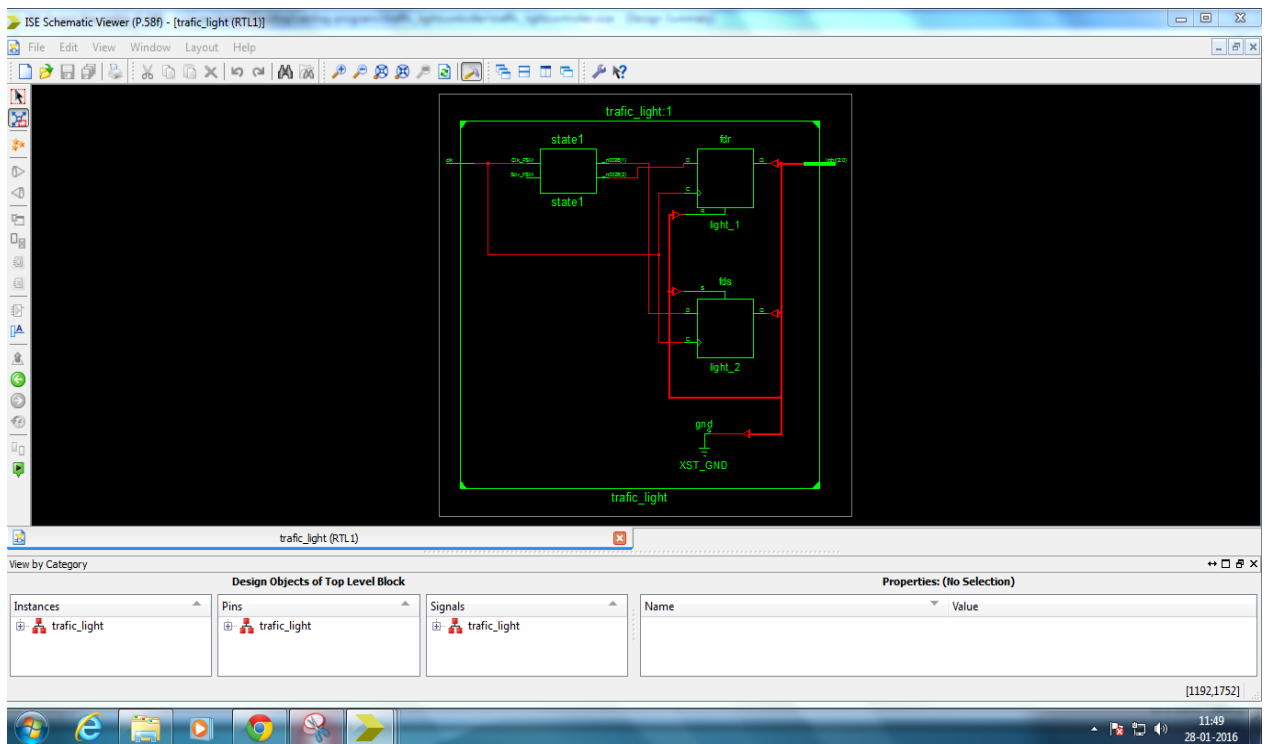


```

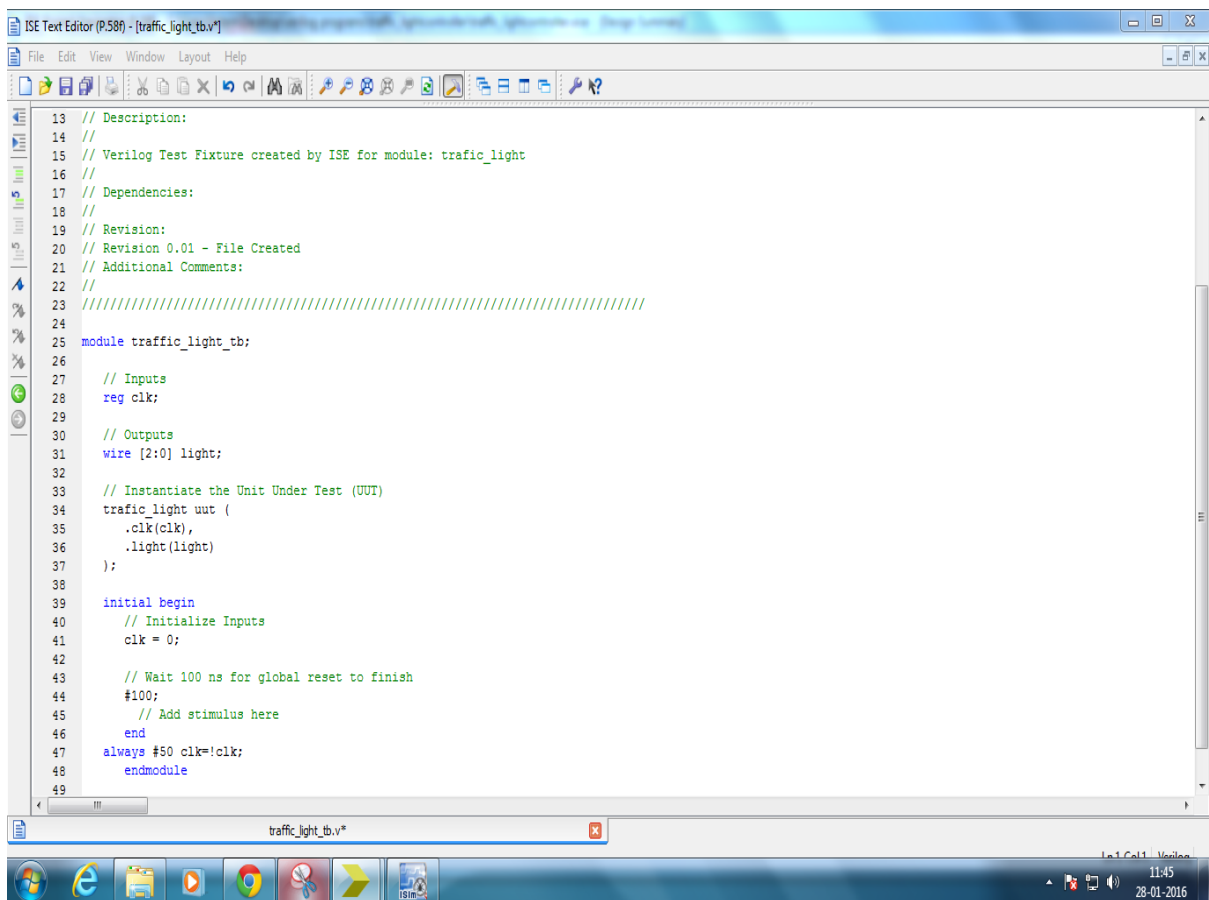
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 ///////////////////////////////////////////////////
21 module traffic_light( clk, light
22 );
23 input clk;
24 output [2:0] light;
25 reg [2:0] light;
26 parameter s0=0, s1=1, s2=2;
27 parameter red=3'b100, green=3'b010, yellow=3'b110;
28 reg[2:0] state;
29 always @( posedge clk)
30 case(state)
31 s0:begin
32 light<=yellow;
33 state <=s1;
34 end
35 s1:begin
36 light<=green;
37 state<=s2;
38 end
39 s2:begin
40 light<=red;
41 state<=s0;
42 end
43 default: begin
44 light <= red;
45 state<= s0;
46 end
47 endcase
48
49
50 endmodule

```

RTL Schematic-



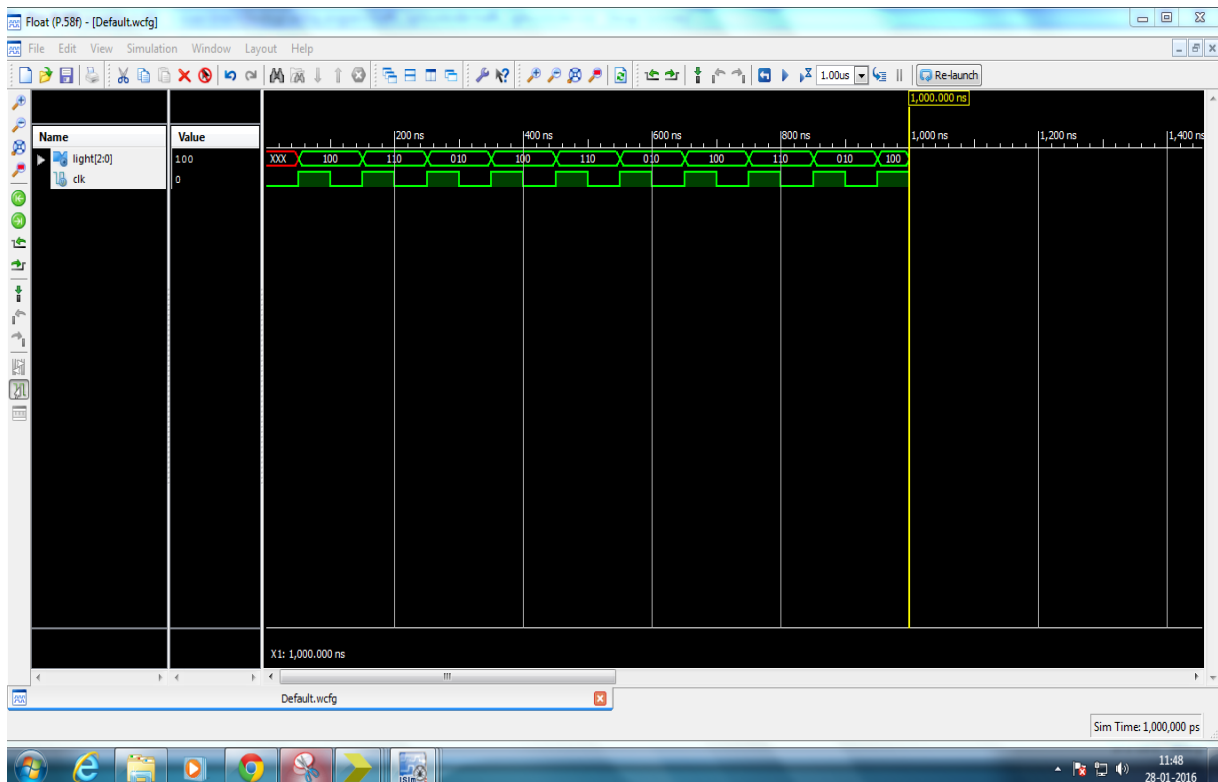
Test Bench for Traffic Light Controller-



The screenshot shows the ISE Text Editor window with the file name "traffic_light_tb.v". The code is a Verilog test bench for a traffic light controller. It includes a module declaration, input/output declarations, instantiation of the Unit Under Test (UUT), and an initial block for initialization and stimulus generation.

```
13 // Description:
14 //
15 // Verilog Test Fixture created by ISE for module: traffic_light
16 //
17 // Dependencies:
18 //
19 // Revision:
20 // Revision 0.01 - File Created
21 // Additional Comments:
22 //
23 ///////////////////////////////////////////////////////////////////
24
25 module traffic_light_tb;
26
27     // Inputs
28     reg clk;
29
30     // Outputs
31     wire [2:0] light;
32
33     // Instantiate the Unit Under Test (UUT)
34     traffic_light uut (
35         .clk(clk),
36         .light(light)
37     );
38
39     initial begin
40         // Initialize Inputs
41         clk = 0;
42
43         // Wait 100 ns for global reset to finish
44         #100;
45         // Add stimulus here
46     end
47     always #50 clk=!clk;
48 endmodule
49
```

Waveform Generated-



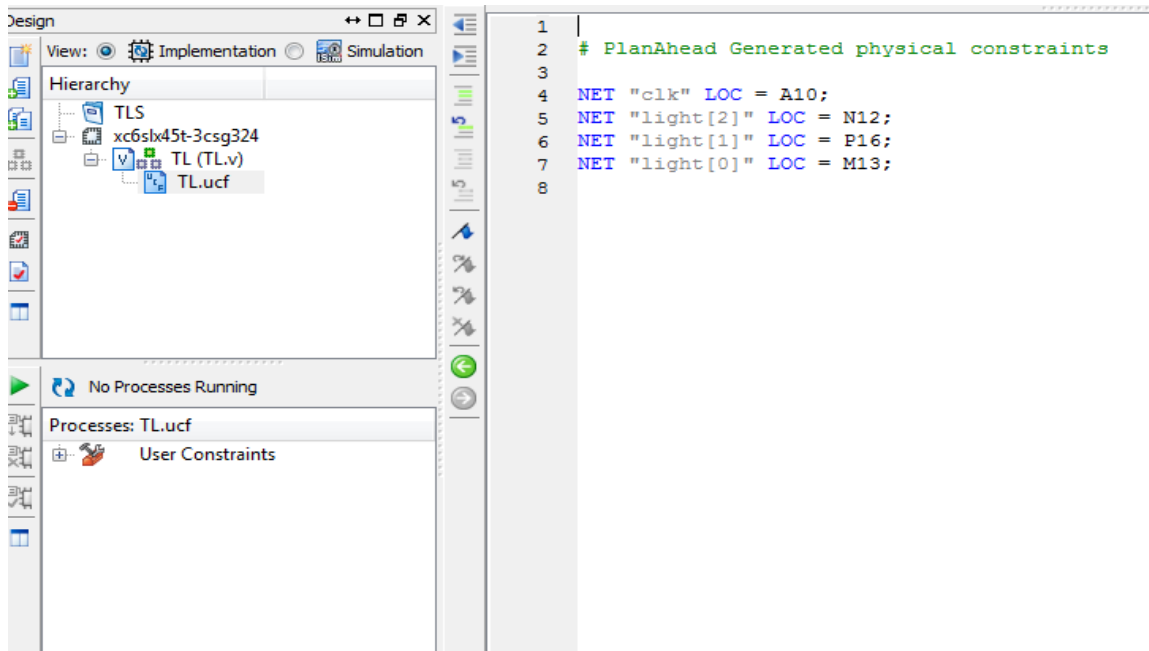
• TIMING ANALYSIS

Clock clk to Pad						
Destination	Max (slowest) clk (edge) to PAD	Process Corner	Min (fastest) clk (edge) to PAD	Process Corner	Internal Clock(s)	Clock Phase
light<0>	8.138 (R)	SLOW	6.339 (R)	FAST	clk_BUFGP	0.000
light<1>	7.903 (R)	SLOW	6.113 (R)	FAST	clk_BUFGP	0.000
light<2>	8.070 (R)	SLOW	6.280 (R)	FAST	clk_BUFGP	0.000
Clock to Setup on destination clock clk						
Source Clock	Src:Rise	Src:Fall	Src:Rise	Src:Fall	Dest:Rise	Dest:Fall
clk	1.285					

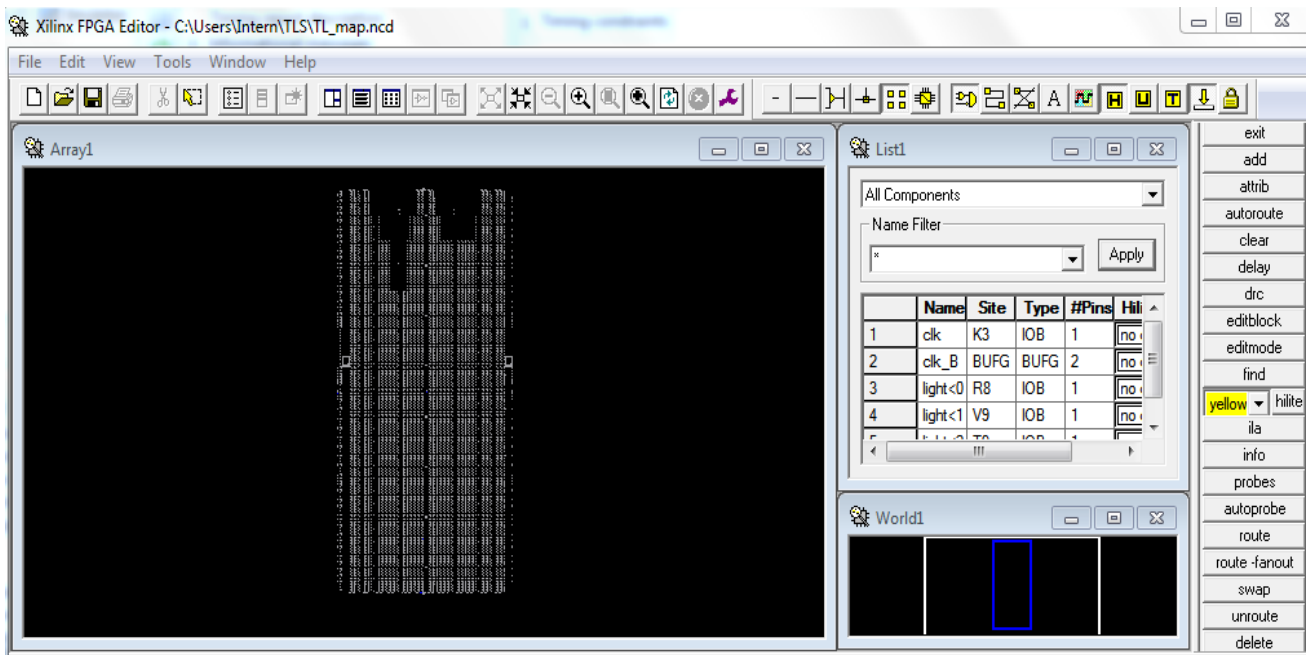
• POWER ANALYSIS

A	B	C	D	E	F	G	H	I	J	K	L	M	N
Device		On-Chip	Power (W)	Used	Available	Utilization (%)			Supply	Summary	Total	Dynamic	Quiescent
Family	Spartan6	Clocks	0.000	1	--	--			Source	Voltage	Current (A)	Current (A)	Current (A)
Part	xc6sxc45t	Logic	0.000	2	27288	0			Vccint	1.200	0.016	0.000	0.015
Package	csg324	Signals	0.000	3	--	--			Vccaux	2.500	0.005	0.000	0.005
Temp Grade	C-Grade	IOs	0.000	4	190	2			Vcco25	2.500	0.002	0.000	0.002
Process	Typical	Leakage	0.036										
Speed Grade	-3	Total	0.037										
Environment		Thermal Properties	Effective TJA	Max Ambient	Junction Temp				Supply Power (W)		Total	Dynamic	Quiescent
Ambient Temp (C)	25.0		(C/W)	(C)	(C)						0.037	0.000	0.036
Use custom TJA?	No		22.6	84.2	25.8								
Custom TJA (C/W)	NA												
Airflow (LFM)	0												
Heat Sink	None												
Custom TSA (C/W)	NA												
Characterization													
Production	v1.3,2011-05-04												

- For Implementing on the FPGA chip, first create Constraints File for declaring the I/O pins specifications as required by the design-

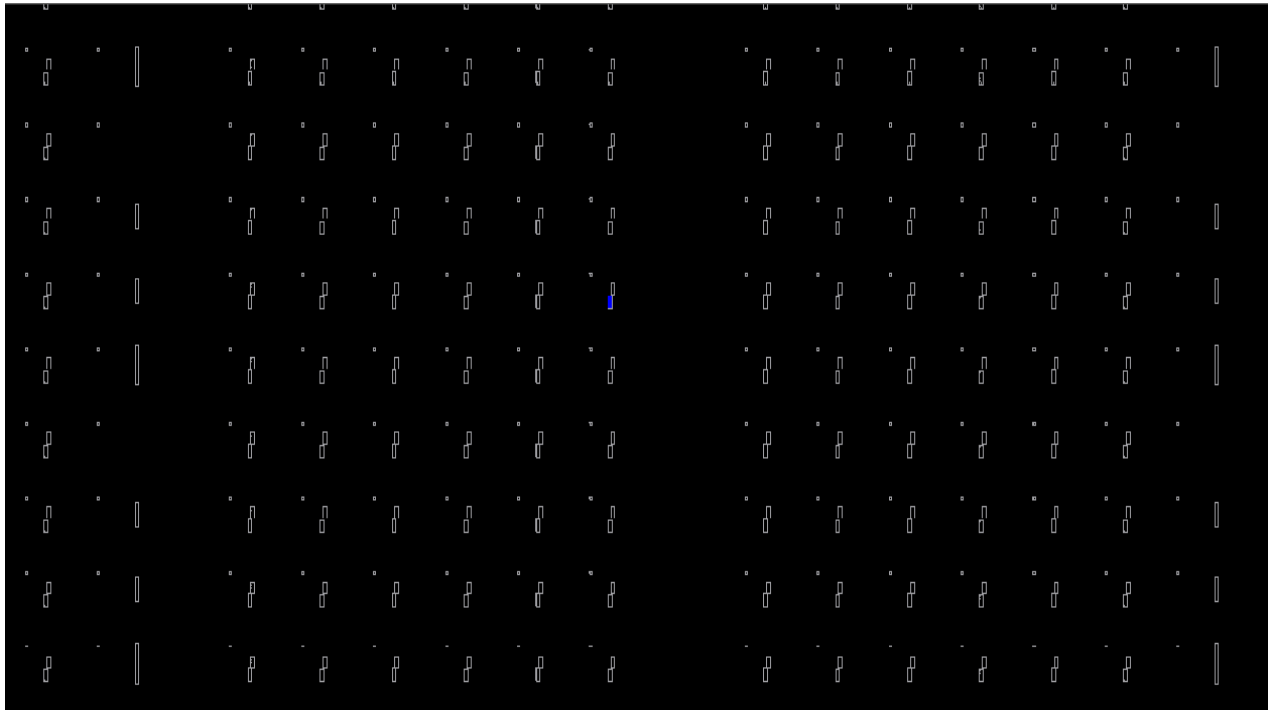


- The below figure depicts the mapping of all the ports given to the Traffic Light design for the proper testing and verification. This can be done and seen by using XILINX FPGA Editor.



- This figure gives the exact locations of the Look-up tables used by the particular design and can be seen by specifying the desired color to the LUT by using the required options provided by the FPGA Editor.

And the blue-colored one is the LUT used by the above design.



- The below elaborated design is used for the pins specifications of the particular design having multiple input-output ports, registers etc. Here it is showing the I/O pins, registers used by the design.

Elaborated Design * 3

Elaborated Design is out-of-date. Constraints were modified. [more info](#) [Reload](#)

RTL Netlist

Package x Device x RTL Schematic x

I/O Port Properties

light[2]

General Attributes Configure

Properties Clock Regions

I/O Ports

Name	Direction	Neg Diff Pair	Site	Fixed	Bank	I/O Std	Vcco	Vref	Drive Stre...	Slew Type	Pull Type	Off-Chip T...	IN_TERM	OUT_TERM
light (3)														
light[0]	Output		N12	<input checked="" type="checkbox"/>		2 default (LVCMOS25)	2.500		12 SLOW	NONE	FP_VTT_50		NONE	
light[1]	Output		M13	<input checked="" type="checkbox"/>		1 default (LVCMOS25)	2.500		12 SLOW	NONE	FP_VTT_50		NONE	
light[2]	Output		P16	<input checked="" type="checkbox"/>		1 default (LVCMOS25)	2.500		12 SLOW	NONE	FP_VTT_50		NONE	
Scalar ports (1)														
clk	Input		A10	<input checked="" type="checkbox"/>										

Net delays:

Delay: 2.772ns (Levels of Logic = 2)

Source: light_2 (FF)

Destination: light_1 (FF)

Source Clock: clk rising

Destination Clock: clk rising

Data Path: light_2 to light_1

		Gate	Net	
Cell:in->out	fanout	Delay	Delay	Logical Name (Net Name)
FDS:C->Q	2	0.447	0.616	light_2 (light_2)
INV:I->O	2	0.206	0.616	light<2>_inv1_INV_0 (state_FSM_FFd1)
INV:I->O	1	0.206	0.579	state__n0026<2>1_INV_0 (_n0026<2>)
FDR:D		0.102		light_1
Total		2.772ns		(0.961ns logic, 1.811ns route) (34.7% logic, 65.3% route)

DESIGN SUMMARY:

Top Level Output File Name : traffic_light.ngc

Primitive and Black Box Usage:

BELS : 5
 # GND : 1
 # INV : 3
 # VCC : 1
 # Flip-Flops/Latches : 3

# FDE	: 1
# FDR	: 1
# FDS	: 1
# Shift Registers	: 1
# SRLC16E	: 1
# Clock Buffers	: 1
# BUFGP	: 1
# IO Buffers	: 3
# OBUF	: 3

Device utilization summary:

Selected Device : XC6SLX45- CSG32 -3

Slice Logic Utilization:

Number of Slice Registers:	3 out of 54576	0%
Number of Slice LUTs:	4 out of 27288	0%
Number used as Logic:	3 out of 27288	0%
Number used as Memory:	1 out of 6408	0%
Number used as SRL:	1	

Slice Logic Distribution:

Number of LUT Flip Flop pairs used:	7	
Number with an unused Flip Flop:	4 out of 7	57%
Number with an unused LUT:	3 out of 7	42%
Number of fully used LUT-FF pairs:	0 out of 7	0%
Number of unique control sets:	2	

IO Utilization:

Number of IOs:	4	
Number of bonded IOBs:	4 out of 218	1%

Specific Feature Utilization:

Number of BUFG/BUFGCTRL/BUFHCEs: 1 out of 16 6%

- The below given figure is the pictorial representation of all the above mentioned details of the used Slice Registers, LUTs, bonded IOBs etc.

traffic_light Project Status (01/26/2016 - 15:24:29)					
Project File:	traffic_lightcontroller.xise	Parser Errors:	No Errors		
Module Name:	traffic_light	Implementation State:	Synthesized		
Target Device:	xc6slx45-3csg324	•Errors:	No Errors		
Product Version:	ISE 14.5	•Warnings:	2 Warnings (2 new)		
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 Registers	3	54576	0%	
Number of Slice LUTs	4	27288	0%	
Number of fully used LUT-FF pairs	0	7	0%	
Number of bonded IOBs	4	218	1%	
Number of BUFG/BUFGCTRL/BUFHCEs	1	16	6%	

Detailed Reports						[-]
Report Name	Status	Generated	Errors	Warnings	Infos	
Synthesis Report	Current	Tue 26. Jan 15:30:19 2016	0	2 Warnings (2 new)	1 Info (1 new)	
Translation Report						

CONCLUSION

Many Combinational and Sequential circuits were implemented in Xilinx using Verilog HDL. Synthesis and Verification of circuits was done on the FPGA kit, XILINX_SPARTAN-6 XC6SLX45. Digital circuits like and, or, not, nand, xor etc. and various application based circuits like Traffic Light Controller, Magnitude Comparator, ALU are also designed and analysed in XILINX ISE Design Suite Software. Analysis of characteristics parameters of a digital circuit were made through the RTL Schematic, Timing Diagrams generated by using the XILINX tools.

It was observed that Critical Time Delay of a complete digital circuit depends on number of gates and nets used. For example, the total time delay for the traffic light controller circuit was 2.772ns and for the ripple carry counter, it was 1.984ns, so it concludes that the number of gates and nets used, is the major factor that decides the critical time taken by the circuit.

While loading the program onto the chip, a particular .bit or .ise File Format is used. This file is generated by the XILINX tool itself. The file contains the object code for a particular design which makes the chip work according to the desired program. Hence the verification is done by uploading .bit file of any particular digital circuit onto the chip.

While using the utility tools, many errors were easily corrected in the later design process. But many complex systems are harder to work with.

Using Verilog HDL, it is easier to organize the hierarchical building blocks and manage complexity of a digital design. With care, it is well-suited for both verification and synthesis.

REFERENCES

[1] *Verilog Digital Design Synthesis* by **Samir Palnitkar**

SunSoft press 1996

[2] Xilinx ISE 14.5 Tutorials

[3] www.xilinx.com/FPGA

[4] *Nptel.ac.in/Verilog*

[5] *ECE 232: Hardware Organization and Design -*

Verilog Tutorial by University of Massachusetts