VHDL versus Verilog

Factor	**VHDL**	**Verilog**
Alphabets	Case insensitive	Case sensitive
Data types	many	Limited
Based on the language	Pascal	C
Library files	Can be included	Lacks library management
IEEE standard's year	In 1987	In 1995
Design size	Large structures are possible	Cannot handle very large designs
Editing style	More verbose	Concise
Program style	entity ckt is	module ckt (a,b,c);
	end ckt;	endmodule
	architecture c1 of ckt is	
	end c1;	
Delimiter	Semicolon is not put at the beginning	Semicolon is inserted at the
	of the entity and architecture; it is	beginning of the module; it is not
	inserted at the end of both of them.	put while ending the module.
Variable (single)	a:in std_logic;	input a;
	b:out std_logic;	output b;
Variable (vector)	a:in std_logic_vector(3 downto 0);	input [3:0] a;
	b:out std_logic_vector(7 downto 0);	output [7:0] b;
Logic vectors	Parentheses are used; e.g. ( )	Brackets are used; e.g. [ ]
Logic values	Nine: 0,1,X,L,H,W,Z,U,-	Four: 0,1,x,z
Sequential execution	process (a,b)	always @(a,b)
	begin	begin

	end process;	end
Assignment	c<=a and b;	assign c=a&b;
(concurrent)	d<= a nor b;	assign d=~(a|b);
Assignment (sequential)		
when "00"=>dout<="0001";		2'b00: dout=4'b0001;
Assignment symbols	Signal assignment <=	Non-blocking assignment
	Variable assignment :=	Blocking assignment
Bitwise operators	not, and, nand, or, nor, xor, xnor	~,&,|,^
Logical operators	---------	!, ||, &&
Conditional equation	if (a='0') then ----;	if (a==0) ----;
	else ----;	else ----;
	end if;	
If loop syntax	elsif	else if
	case din is	case (din)
Case	end case;	endcase
	Signal (concurrent),	Wire (concurrent),
Temporary declarations	variable (sequential)	reg (sequential)
Gate level and transistor-level	Built-in	Not available
primitives		

COMMONLY USED SYMBOLS

Sl.no.	Symbol	Name
1	~	Tilde
2	'	Tick (Grave)
3	`	Back-tick (Acute)
4	"	Quote
5	&	Ampersand
6	^	Caret
7	#	Cross-hash (Sharp)
8	*	Asterisk
9	(	Parenthesis
10	[	Bracket
11	{	Brace
12	<	Angle bracket
13	/	Slash (Solidus)
14	\	Back-slash
15	|	Pipe (Bar)
16	-	Hyphen (Dash)
17	@	Astatine (At)
18	:	Colon
19	’	Comma (Apostrophe)

20	…	Ellipsis
21	!	Exclamation (Bang)
22	_	Underscore
23	.	Period
24	;	Semicolon

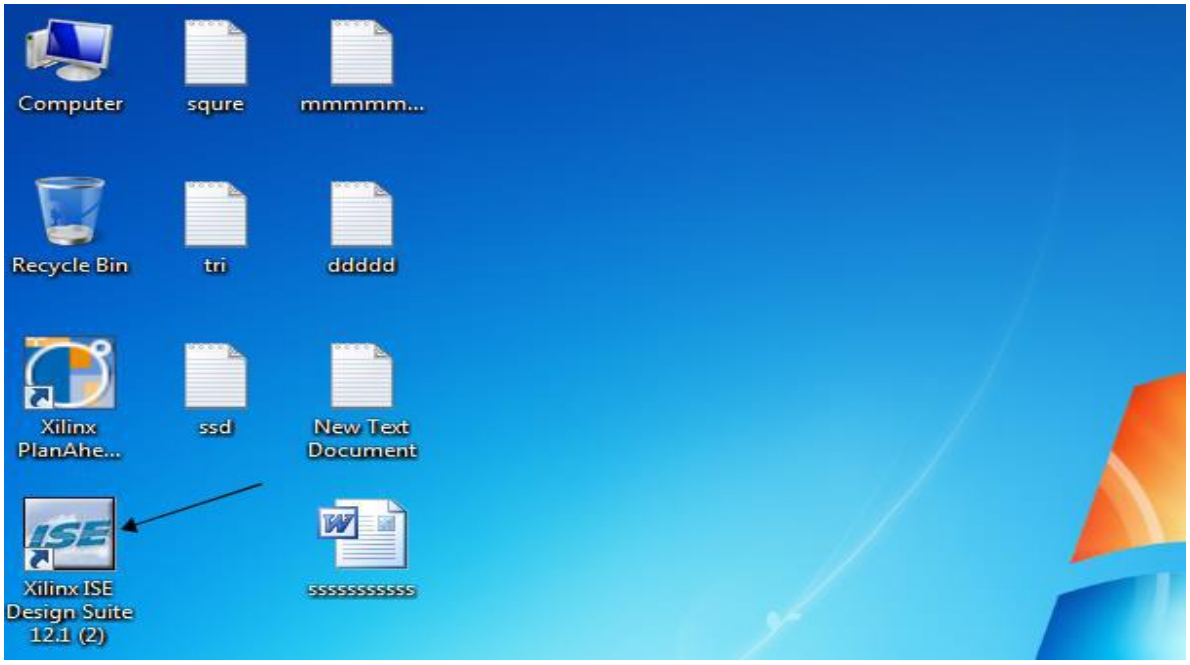
ABBREVIATIONS

CPLD	Complex Programmable Logic D
FPGA	Field Programmable Gate Array
GAL	Generic Array Logic
HDL	Hardware Description Language
IEEE	Institute of Electrical & Electronic Engineers
IP	Intellectual Property
ILA	Integrated Logic Analyzer
ISE	Integrated Software Environment
ISP	In-System Programming
JKFF	Jack-Kilby Flip Flop
JTAG	Joint Test Action Group
LEC	Logic Equivalence Checker

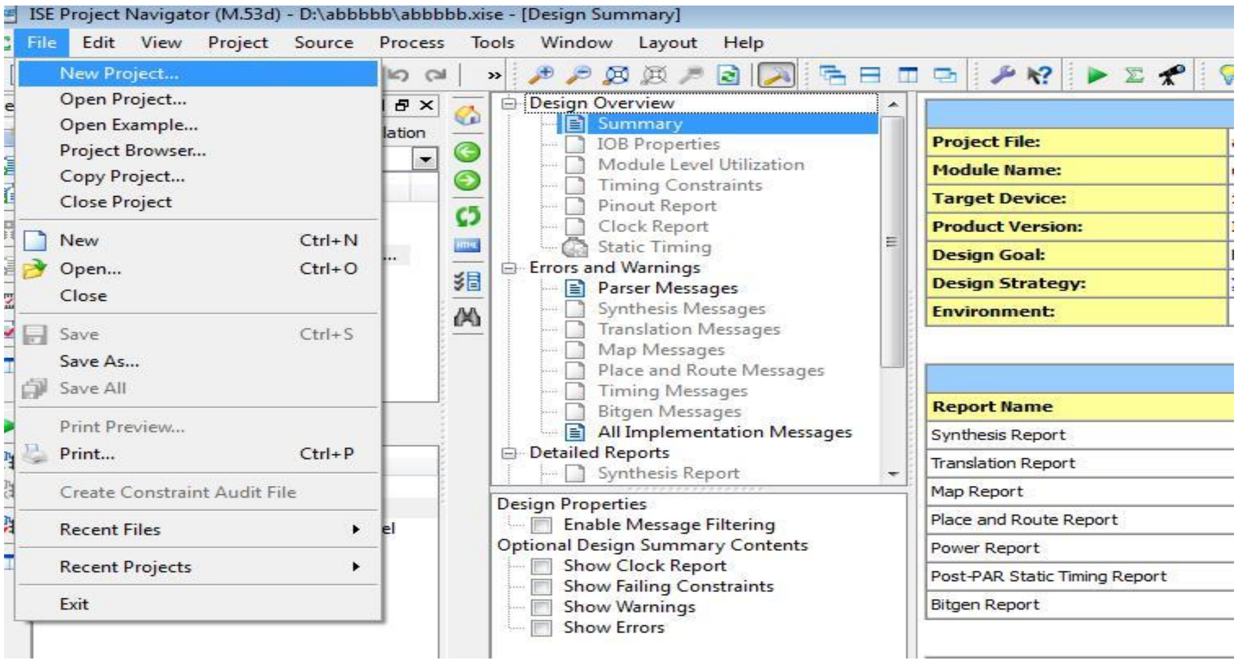
LMG	Logic Modeling Group
LUT	Look-Up Table
NGC	Native Generic Compiler
OTP	One-Time Programmable
PACE	Pin-out and Area Constraints Editor
PAL	Programmable Array Logic
PCI	Peripheral Component Interconnect
PLA	Programmable Logic Array
TBW	Test-Bench Waveform
UCF	User Constraints File
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
XST	Xilinx Synthesis Technology

Steps for Executing VHDL/Verilog Programs

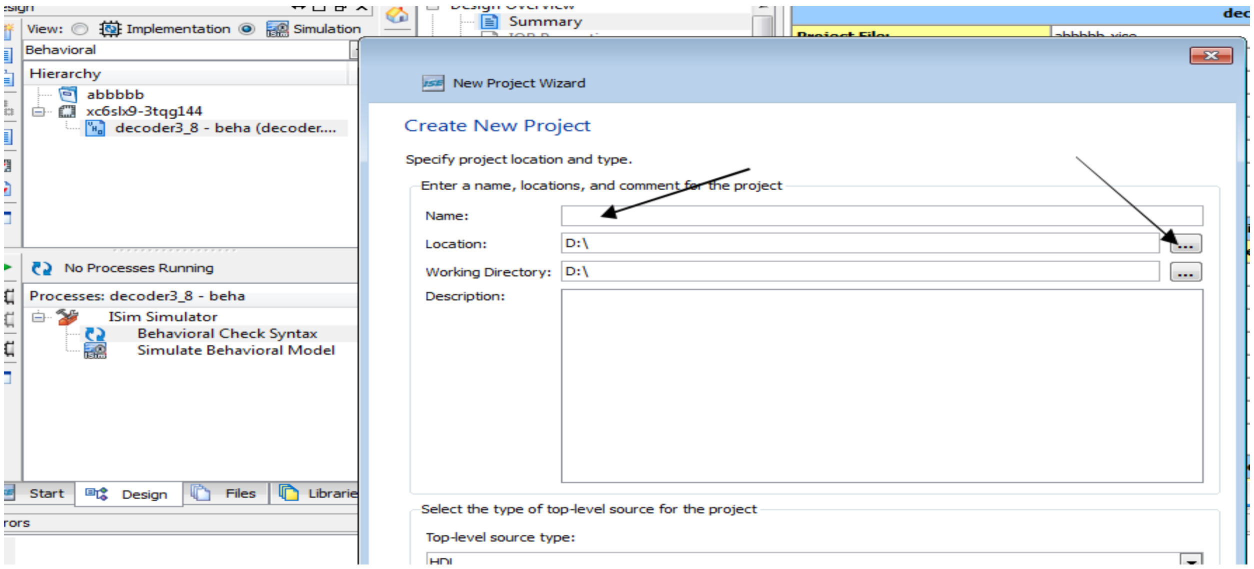
1. Go to start menu. Click on Xilinx Design suit 12.1, then software is opening



2. Through the Software window, Go to file menu, Select new project



3.New project Wizard is opening ,Give project name and choose project location and give Next.



Create New Project

Specify project location and type.

Enter a name, locations, and comment for the project

Name:	
Location:	D:\
Working Directory:	D:\
Description:	

Select the type of top-level source for the project

Top-level source type:

HDL

4. Project setting window is opening there made particular changes given below.

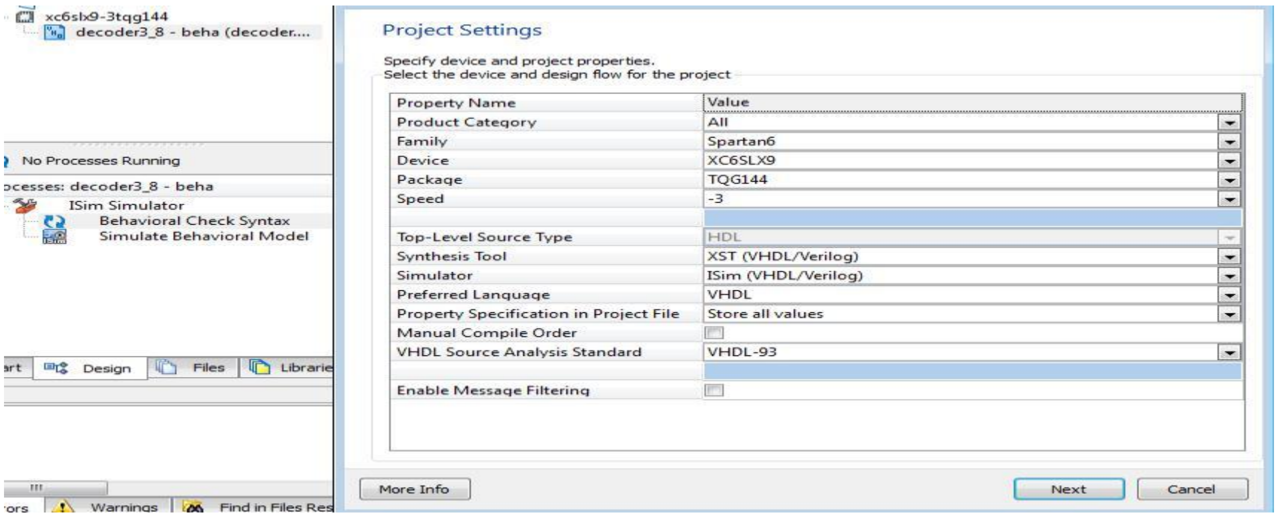
1) Device family ……….. SPARTAN 6

2) Device …………. XC6LX9.

3) Package …………. Tq 144.

4) Simulator …………… ISE Simulator

5) Generated simulation language:VHDL for vhdl program/Verilog for verilog program.



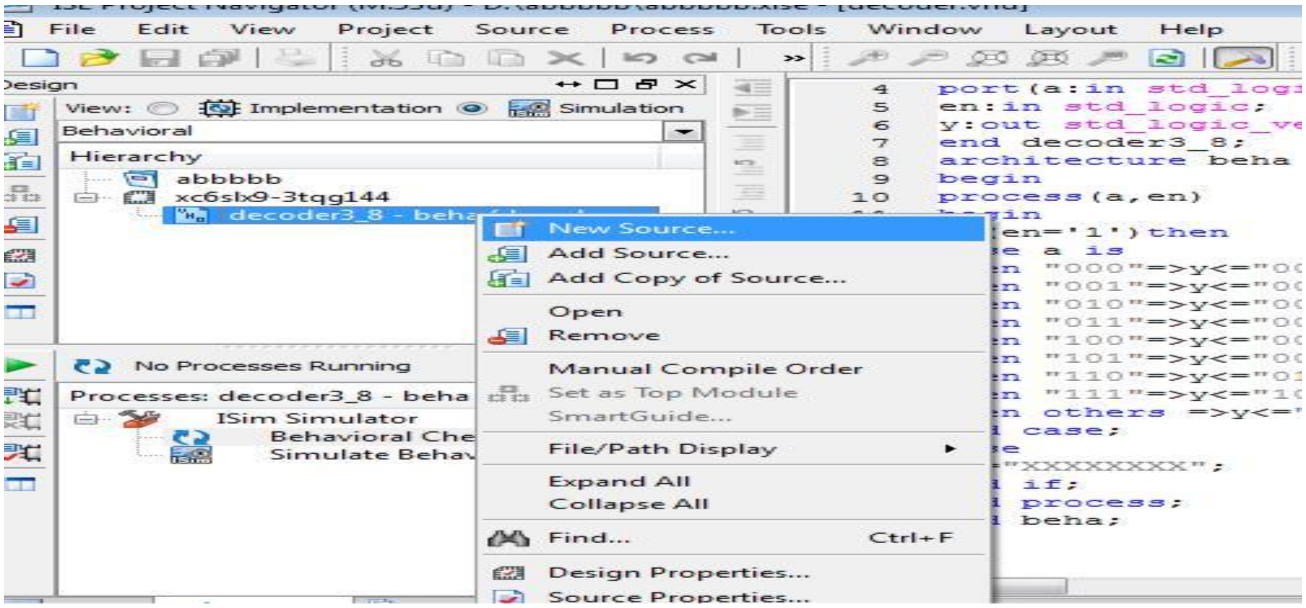
Project Settings

Specify device and project properties.
Select the device and design flow for the project

Property Name	Value
Product Category	All
Family	Spartan6
Device	XC6SLX9
Package	TQG144
Speed	-3
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	VHDL
Property Specification in Project File	Store all values
Manual Compile Order	☐
VHDL Source Analysis Standard	VHDL-93
Enable Message Filtering	☐

4. After typing the program on edit window save the program with (.vhd for vhdl and .v for verilog) suitable extension and add the program on source window for compilation. For clearing the errors double click on behavioral simulation that shows errors on transcript window .After clearing the errors create a Test bench for assigning input values given below.

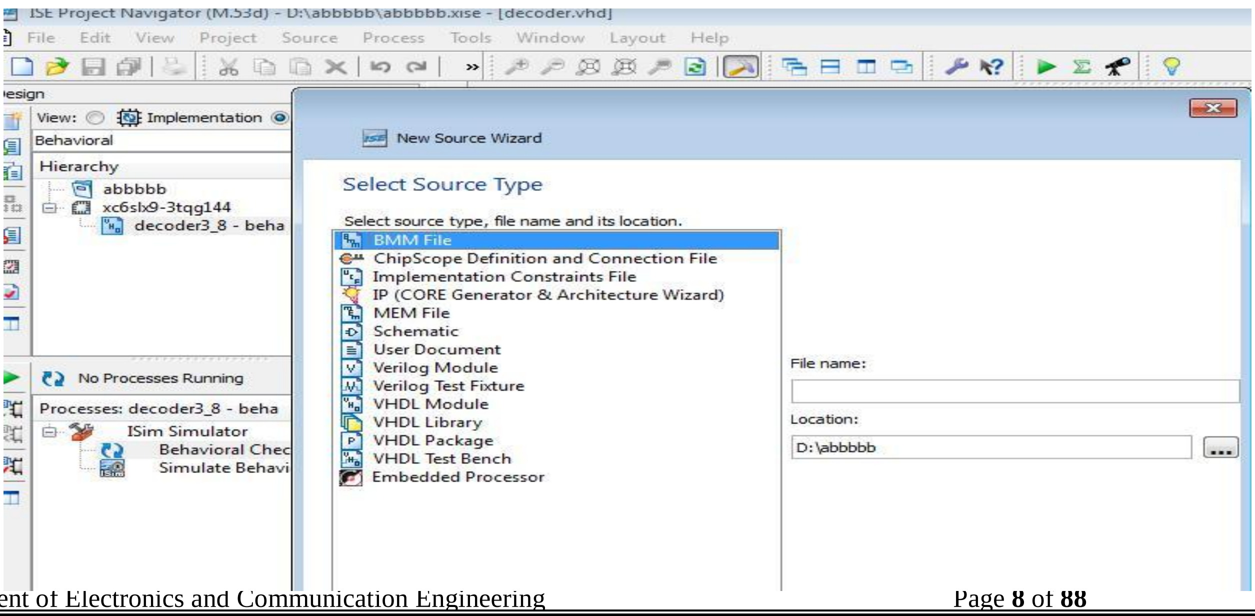
Write click on the program and go to new source



```
port(a:in std_logi
en:in std_logic;
y:out std_logic_ve
end decoder3_8;
architecture beha
begin
process(a,en)
```

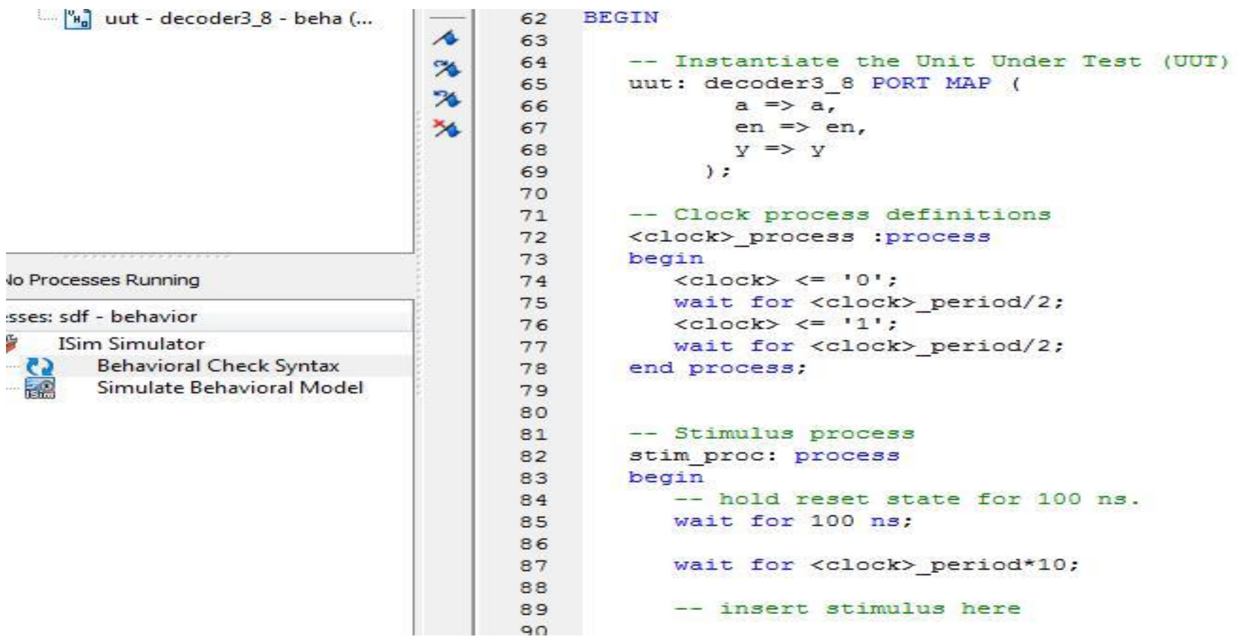
6) New source wizard is opening here, select vhdl test bench for vhdl program and verilg

text fixture for verilog program and give a file name on the box given.



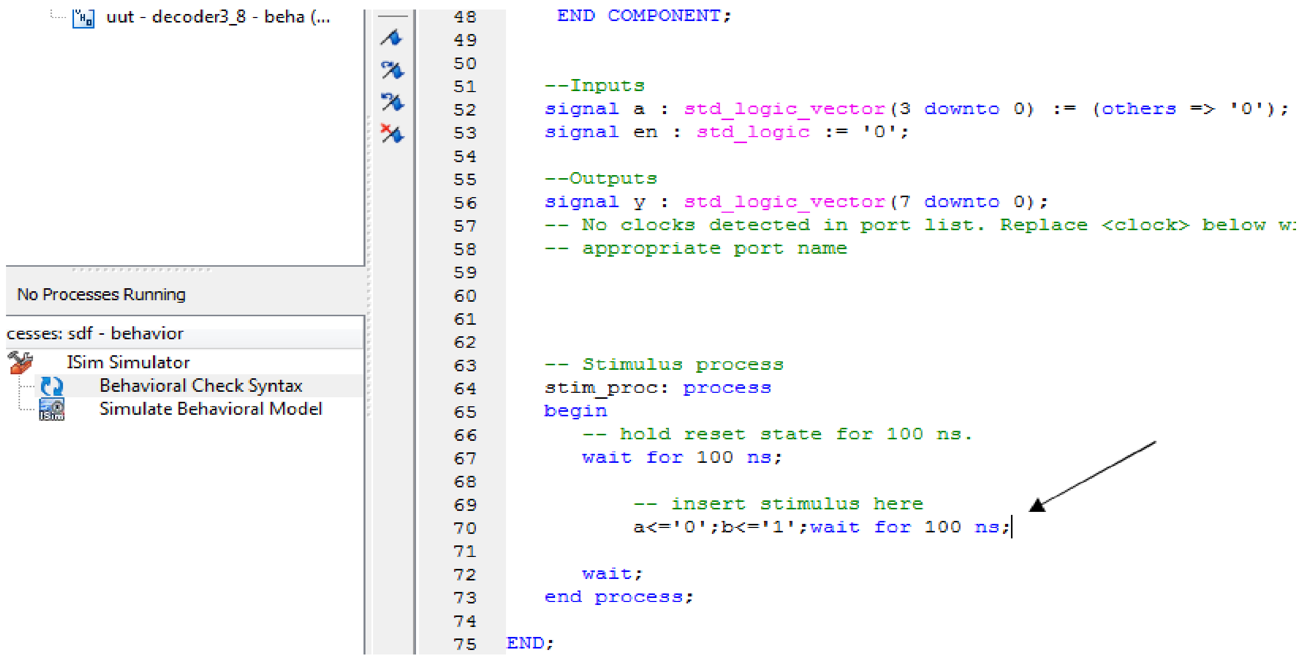
Location: D:\abbbbb

7) After appearing test bench remove all clock related part from test bench (this is for without clock related programs execution).For verilog this process is not required.



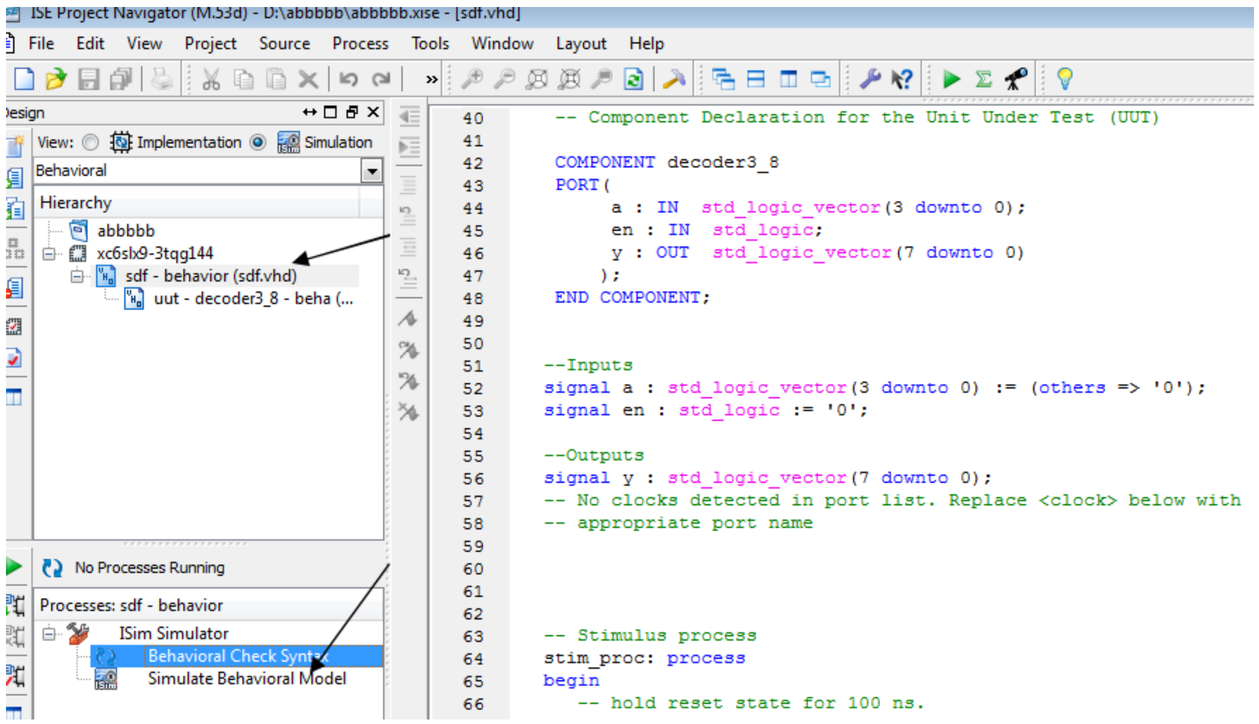
```
BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: decoder3_8 PORT MAP (
          a => a,
          en => en,
          y => y
        );

   -- Clock process definitions
   <clock>_process :process
   begin
      <clock> <= '0';
      wait for <clock>_period/2;
      <clock> <= '1';
      wait for <clock>_period/2;
   end process;


   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
      wait for 100 ns;

      wait for <clock>_period*10;

      -- insert stimulus here
```

8) After made all the changes add input values below “insert stimulus “here given on the particular test bench window shown below.



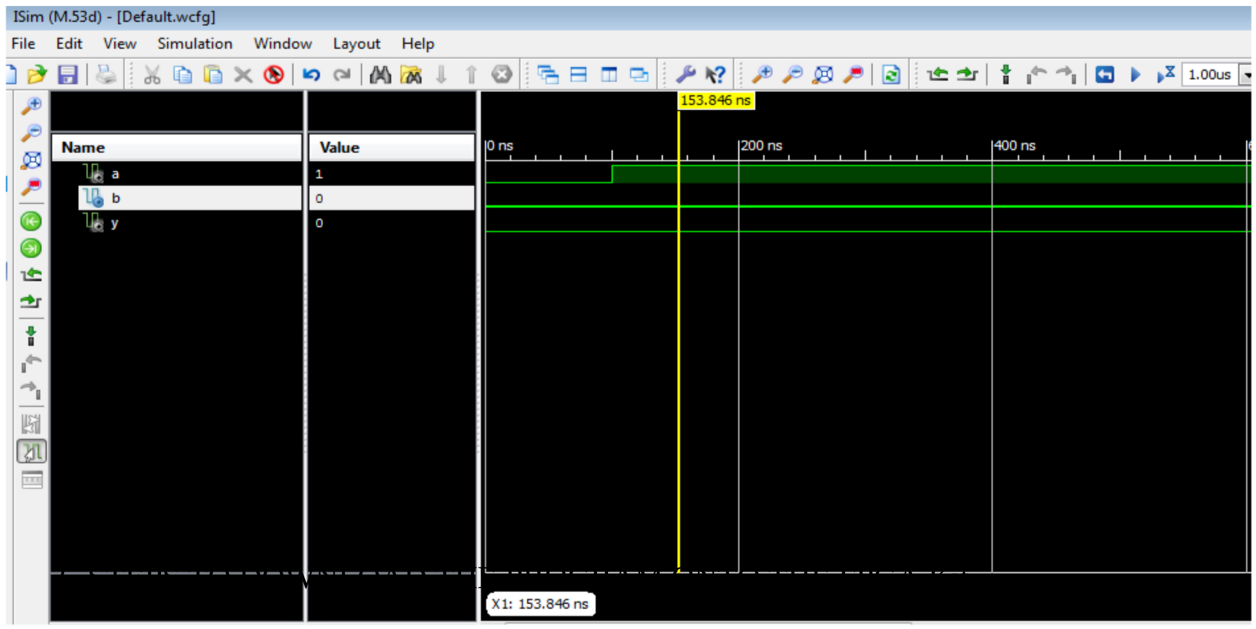
```
    END COMPONENT;


   --Inputs
   signal a : std_logic_vector(3 downto 0) := (others => '0');
   signal en : std_logic := '0';

   --Outputs
   signal y : std_logic_vector(7 downto 0);
   -- No clocks detected in port list. Replace <clock> below w:
   -- appropriate port name




   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
      wait for 100 ns;

         -- insert stimulus here
         a<='0';b<='1';wait for 100 ns;

      wait;
   end process;

END;
```

9)After giving the input go to source window and select test bench on the source window and double click on simulate behavioral model on process window Shown below.

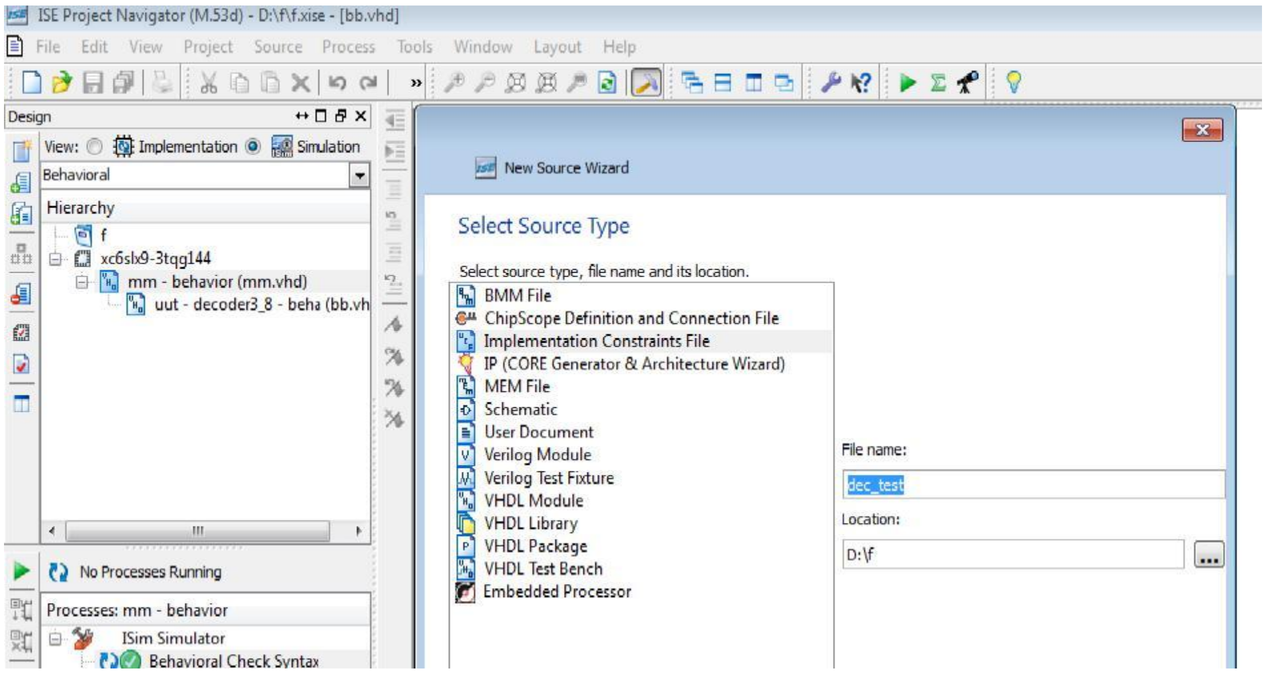

ISE Project Navigator (M.53d) - D:\abbbbb\abbbbb.xise - [sdf.vhd]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
abbbbb
xc6slx9-3tqg144
sdf - behavior (sdf.vhd)
uut - decoder3_8 - beha (...
No Processes Running
Processes: sdf - behavior
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model

```
 -- Component Declaration for the Unit Under Test (UUT)

 COMPONENT decoder3_8
 PORT(
      a : IN  std_logic_vector(3 downto 0);
      en : IN  std_logic;
      y : OUT  std_logic_vector(7 downto 0)
     );
 END COMPONENT;


--Inputs
signal a : std_logic_vector(3 downto 0) := (others => '0');
signal en : std_logic := '0';

--Outputs
signal y : std_logic_vector(7 downto 0);
-- No clocks detected in port list. Replace <clock> below with
-- appropriate port name



-- Stimulus process
stim_proc: process
begin
   -- hold reset state for 100 ns.
```

10) Proper output window is generating shown below.


ISim (M.53d) - [Default.wcfg]
File Edit View Simulation Window Layout Help
1.00us
153.846 ns
Name
Value
a
1
b
0
y
0
0 ns
200 ns
400 ns
X1: 153.846 ns

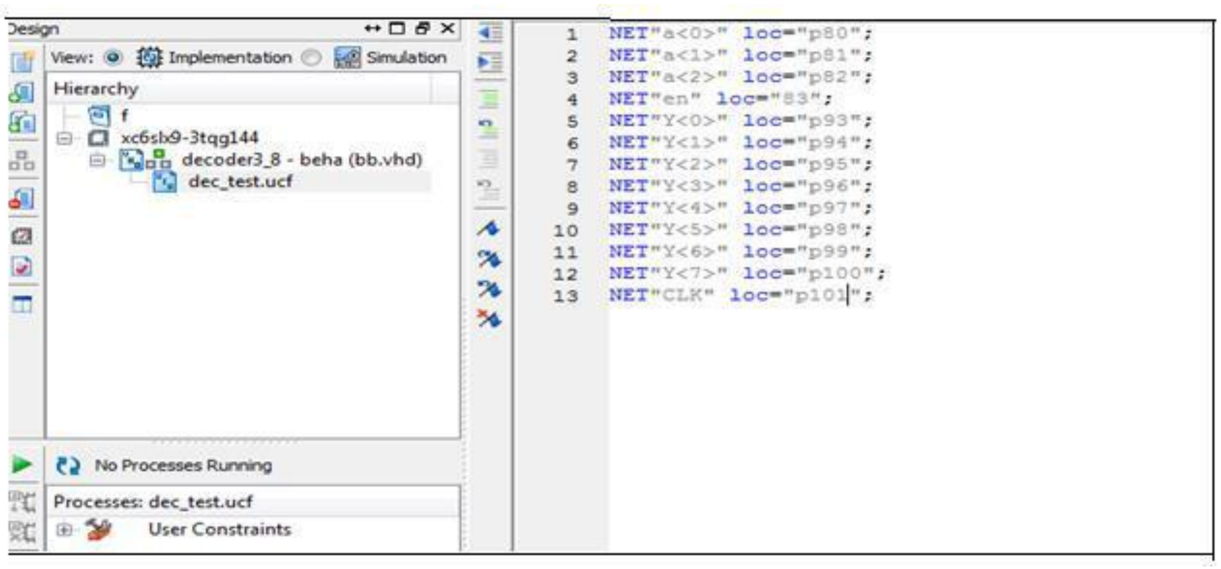
1. Go to source in project window, right click on Source file. Select the New source, Select Implementation Constraints file, give the file name and click next.


ISE Project Navigator (M.53d) - D:\f\f.xise - [bb.vhd]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
f
xc6slx9-3tqg144
mm - behavior (mm.vhd)
uut - decoder3_8 - beha (bb.vh
No Processes Running
Processes: mm - behavior
ISim Simulator
Behavioral Check Syntax
New Source Wizard
Select Source Type
Select source type, file name and its location.
BMM File
ChipScope Definition and Connection File
Implementation Constraints File
IP (CORE Generator & Architecture Wizard)
MEM File
Schematic
User Document
Verilog Module
Verilog Test Fixture
VHDL Module
VHDL Library
VHDL Package
VHDL Test Bench
Embedded Processor

File name: dec_test

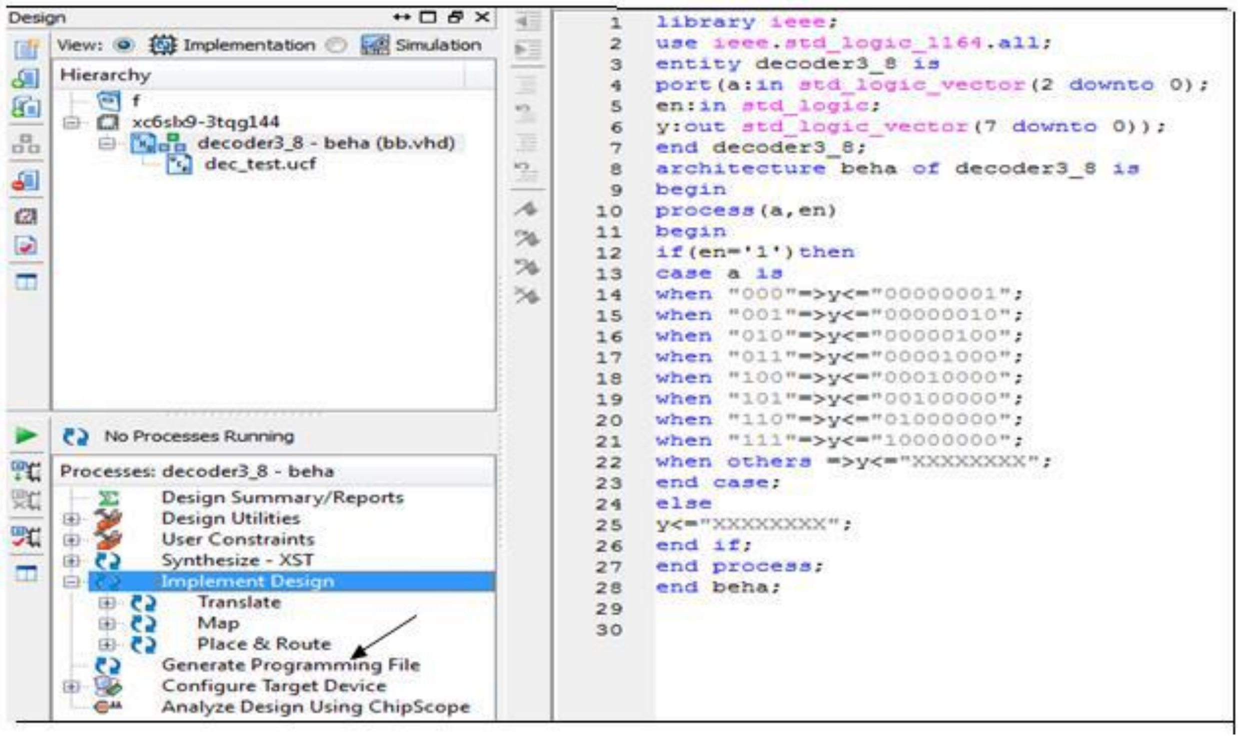
Location: D:\f

2. one blank edit window will open, there you type particular pin numbers, save and exit.



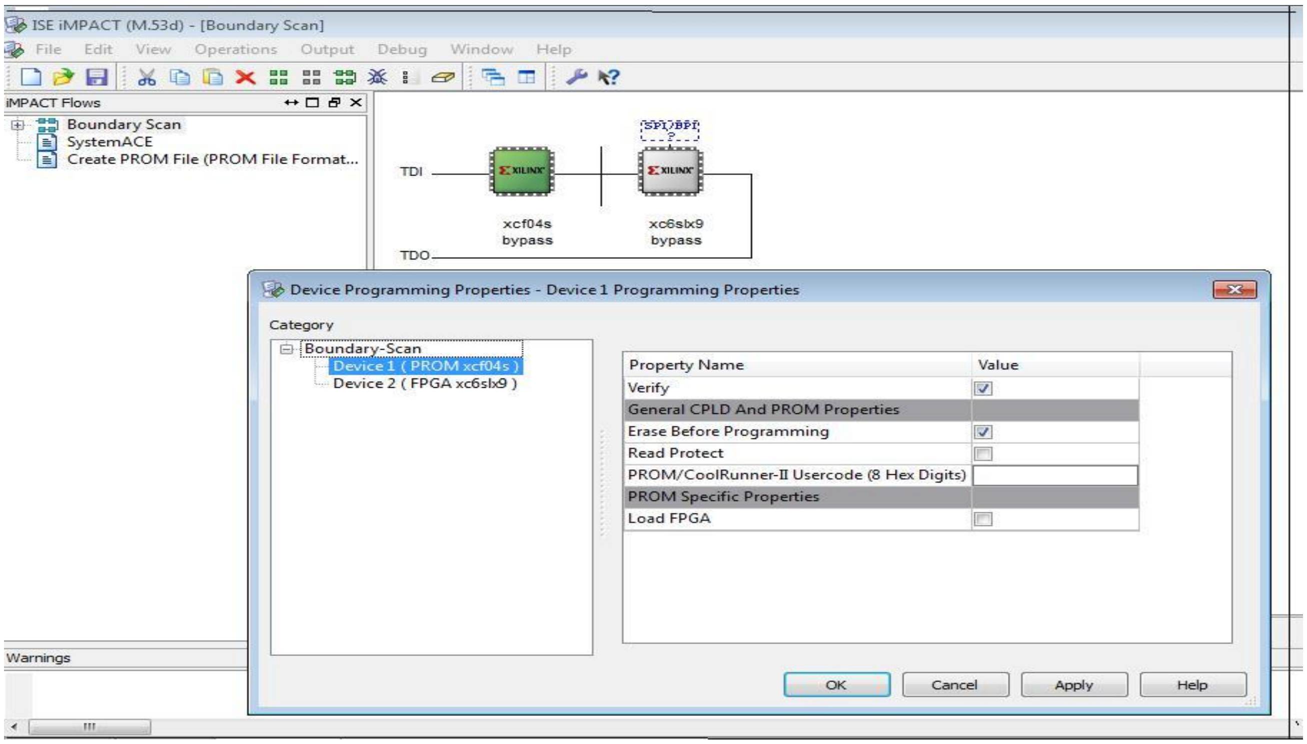
```
NET"a<0>" loc="p80";
NET"a<1>" loc="p81";
NET"a<2>" loc="p82";
NET"en" loc="83";
NET"Y<0>" loc="p93";
NET"Y<1>" loc="p94";
NET"Y<2>" loc="p95";
NET"Y<3>" loc="p96";
NET"Y<4>" loc="p97";
NET"Y<5>" loc="p98";
NET"Y<6>" loc="p99";
NET"Y<7>" loc="p100";
NET"CLK" loc="p101";
```

3. Go to process for source window, double click, on the implementation design, and double on generating programming file.



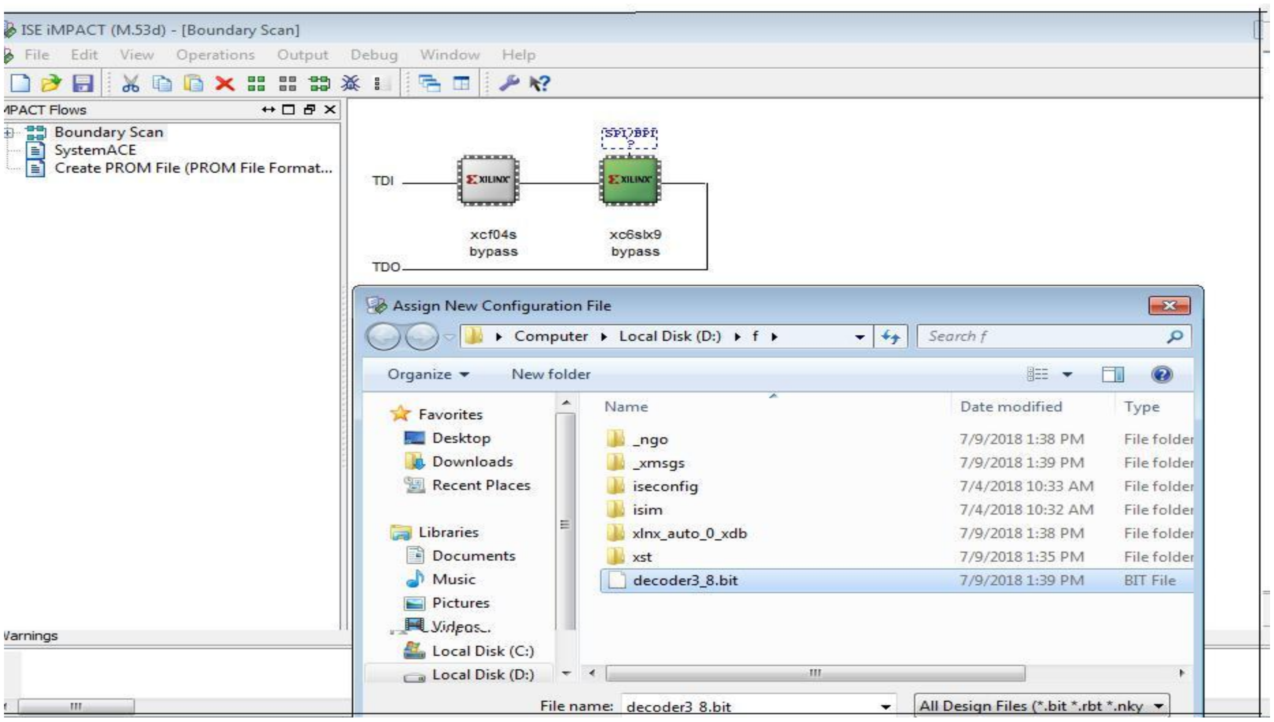
```
library ieee;
use ieee.std_logic_1164.all;
entity decoder3_8 is
port(a:in std_logic_vector(2 downto 0);
en:in std_logic;
y:out std_logic_vector(7 downto 0));
end decoder3_8;
architecture beha of decoder3_8 is
begin
process(a,en)
begin
if(en='1')then
case a is
when "000"=>y<="00000001";
when "001"=>y<="00000010";
when "010"=>y<="00000100";
when "011"=>y<="00001000";
when "100"=>y<="00010000";
when "101"=>y<="00100000";
when "110"=>y<="01000000";
when "111"=>y<="10000000";
when others =>y<="XXXXXXXX";
end case;
else
y<="XXXXXXXX";
end if;
end process;
end beha;
```

4. click ok and double click on boundary scan and select the device and click ok.


ISE iMPACT (M.53d) - [Boundary Scan]
File Edit View Operations Output Debug Window Help
iMPACT Flows
Boundary Scan
SystemACE
Create PROM File (PROM File Format...
SPI/BPI ?
TDI
TDO
XILINX
xcf04s bypass
xc6slx9 bypass
Warnings
Device Programming Properties - Device 1 Programming Properties
Category
Boundary-Scan
Device 1 (PROM xcf04s)
Device 2 (FPGA xc6slx9)
OK
Cancel
Apply
Help

Property Name	Value
Verify	✓
General CPLD And PROM Properties	
Erase Before Programming	✓
Read Protect	☐
PROM/CoolRunner-II Usercode (8 Hex Digits)	
PROM Specific Properties	
Load FPGA	☐

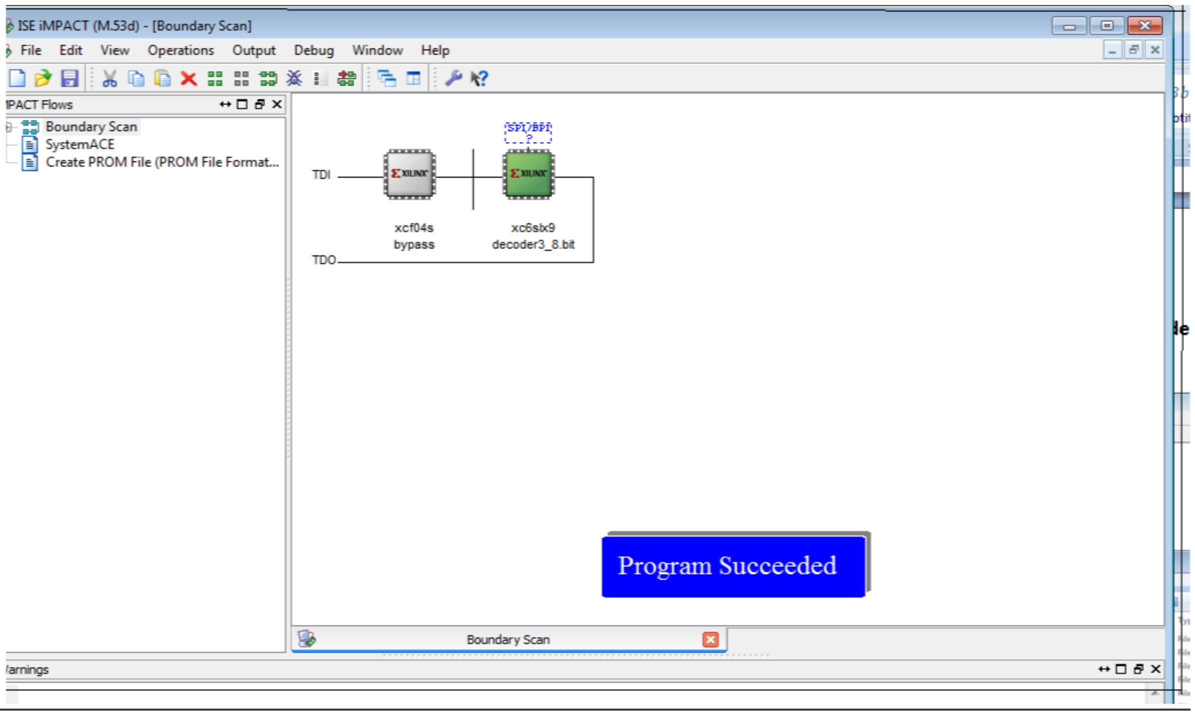
5. Then it showing identity succeeded and right click on the target device and select program.



Name	Date modified	Type
_ngo	7/9/2018 1:38 PM	File folder
_xmsgs	7/9/2018 1:39 PM	File folder
iseconfig	7/4/2018 10:33 AM	File folder
isim	7/4/2018 10:32 AM	File folder
xlnx_auto_0_xdb	7/9/2018 1:38 PM	File folder
xst	7/9/2018 1:35 PM	File folder
decoder3_8.bit	7/9/2018 1:39 PM	BIT File

File name: decoder3_8.bit
All Design Files (*.bit *.rbt *.nky

6. Then it asking flash prom required or not, select NO, Right click on target device and double click on program. It showing program succeeded.


ISE iMPACT (M.53d) - [Boundary Scan]
File Edit View Operations Output Debug Window Help
iMPACT Flows
Boundary Scan
SystemACE
Create PROM File (PROM File Format...
TDI
TDO
xcf04s
bypass
xc6slx9
decoder3_8.bit
Program Succeeded
Boundary Scan
Warnings

EXPERIMENT NO. :1

Write an HDL code to realize all logic gates.

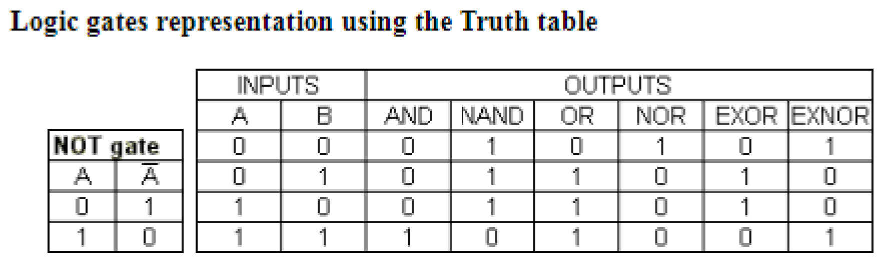
AIM

To write an HDL code to realize all logic gates. Synthesize and view the RTL schematic.

Software Required : Xilinx 12.1 , ISim simulator

Theory:

Digital systems are said to be constructed by using logic gates. These gates are the AND, OR, NOT, NAND, NOR, EXOR and EXNOR gates. The basic operations are described below with the aid of truth tables



Logic gates representation using the Truth table

NOT gate	
A	$\overline{A}$
0	1
1	0

INPUTS		OUTPUTS					
A	B	AND	NAND	OR	NOR	EXOR	EXNOR
0	0	0	1	0	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	0	1	0	0	1

a) VHDL program:

```
library ieee;
use ieee.std_logic_1164.all;
entity gates is
port(a,b: in std_logic;
c,d,e,f,g,h,i: out std_logic);
end gates;
architecture g1 of gates is
begin
        c<=not a;
        d<=a or b;

        e<=a nor b;
        f<=a and b;
        g<=a nand b;
        h<=a xor b;
        i<=a xnor b;
end g1;
```

Test Bench Code (VHDL):-

```
------ insert stimulus here-------

a<='0';b<='1';wait for 100 ns;
a<='1';b<='0';wait for 100 ns;
 a<='1';b<='1';wait for 100 ns;

 a<='0';b<='0';wait for 100 ns;
```

b) Verilog program:

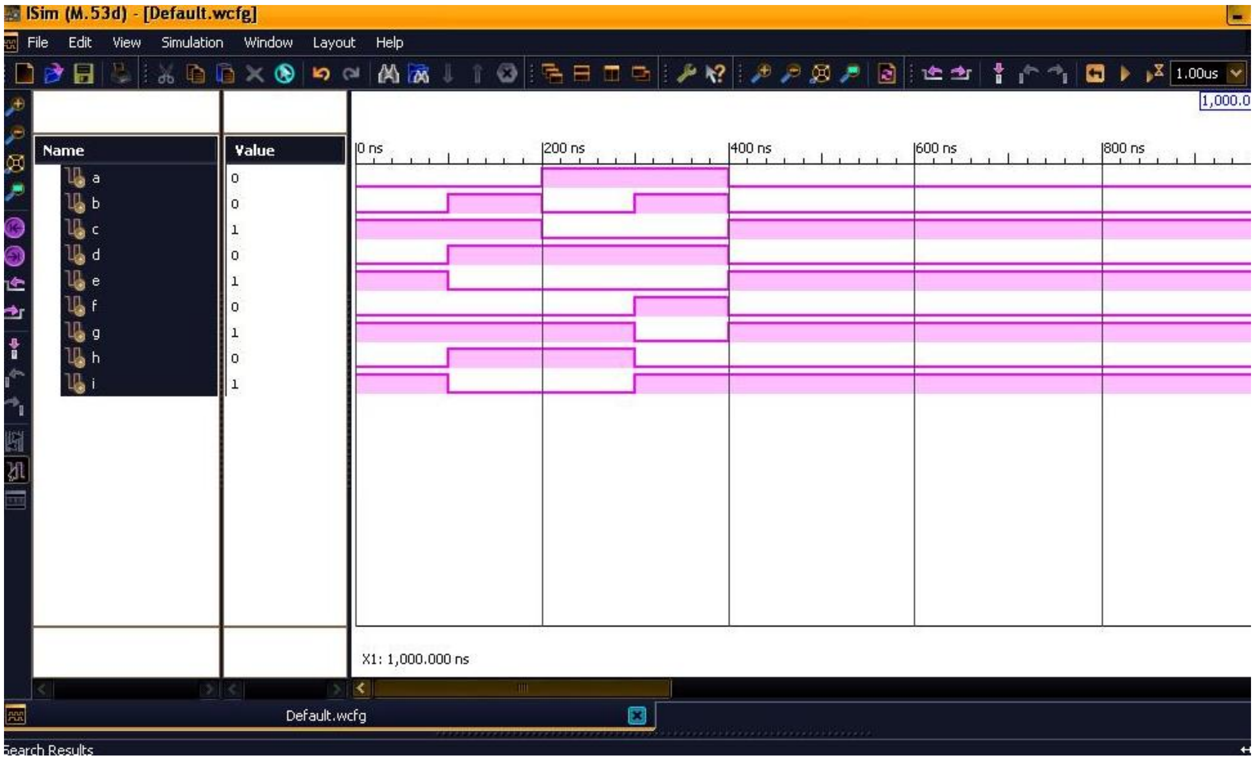
```
module gates(a,b,c,d,e,f,g,h,i);
input a,b;
output c,d,e,f,g,h,i;
assign c=~a;
assign d=a|b;
assign e=~(a|b);
assign f=a&b;
assign g=~(a&b);
assign h=a^b;
assign i=~(a^b);

endmodule
```

Test Bench Code (Verilog):-

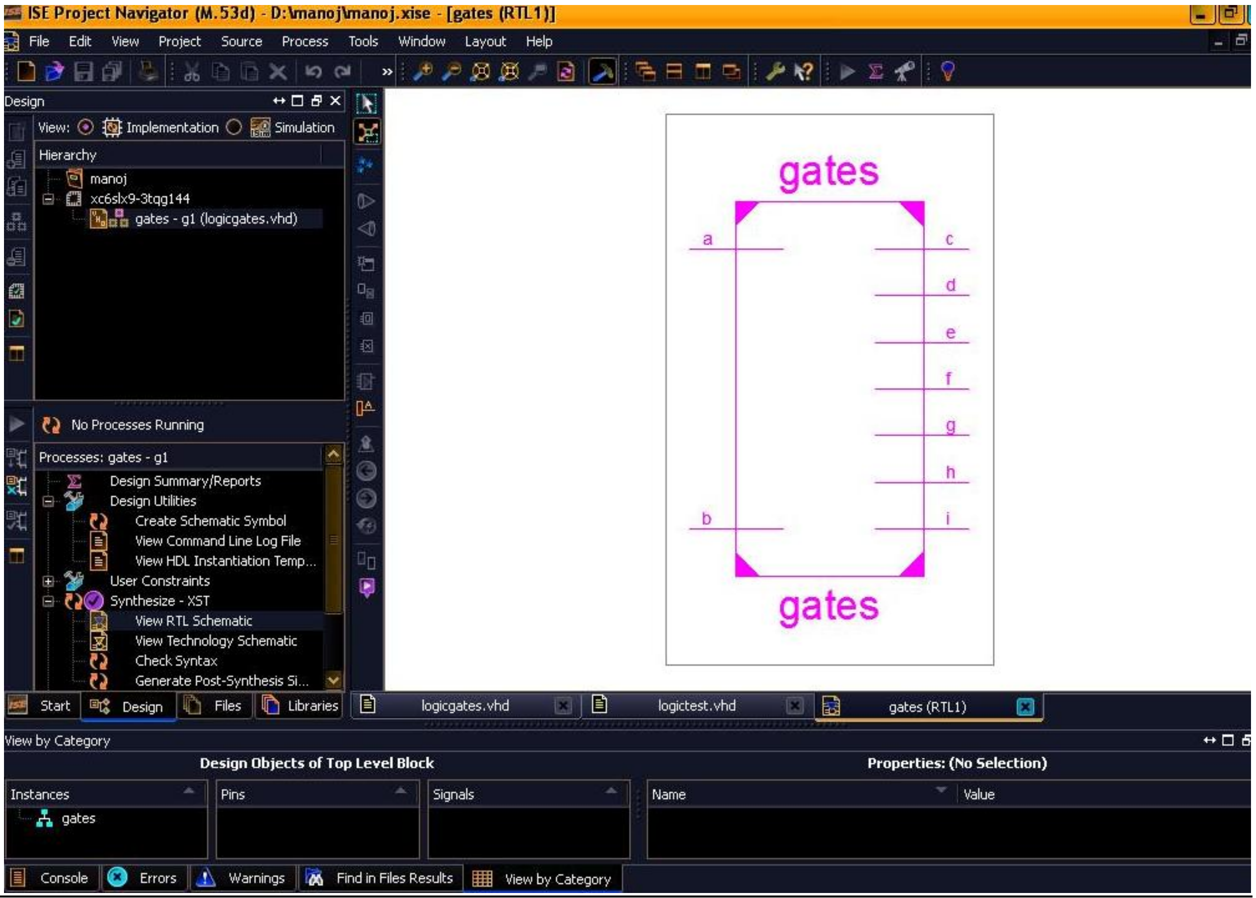
```
/////// Add stimulus here///////

            a = 0;b = 1; #100;
            a = 1;b = 0; #100;
            a = 1;b = 1; #100;
            a = 0;b = 0; #100;
```

Results and Simulation Output :

The HDL code for basic gates has been written and verified using test bench.



RTL SCHEMATIC


ISE Project Navigator (M.53d) - D:\manoj\manoj.xise - [gates (RTL1)]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
manoj
xc6slx9-3tqg144
gates - g1 (logicgates.vhd)
No Processes Running
Processes: gates - g1
Design Summary/Reports
Design Utilities
Create Schematic Symbol
View Command Line Log File
View HDL Instantiation Temp...
User Constraints
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Si...
gates
a
b
c
d
e
f
g
h
i
gates
Start Design Files Libraries
logicgates.vhd
logictest.vhd
gates (RTL1)
View by Category
Design Objects of Top Level Block
Properties: (No Selection)
Instances
gates
Pins
Signals
Name
Value
Console Errors Warnings Find in Files Results View by Category

EXPERIMENT NO.: 2

Write an HDL code to describe the functions of a Full Adder using three modeling styles.

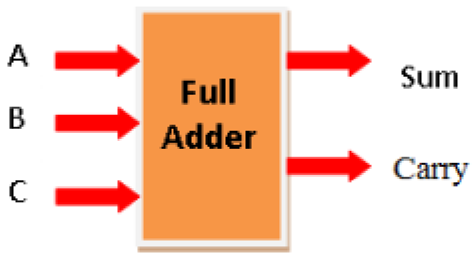
AIM:

To Write an HDL code to describe the functions of a Full Adder using three modeling styles.

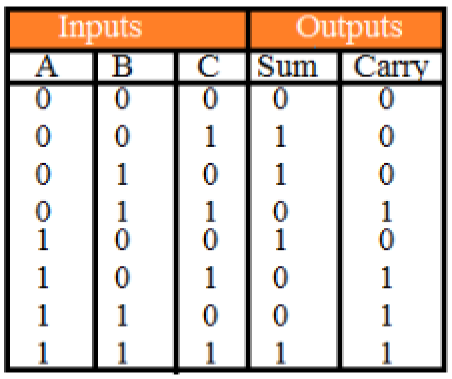
Software Required : Xilinx ISE 12.1 , ISim simulator

Theory:

A full adder is a combinational circuit that adds two one bit numbers along with a carry and produces a sum and carry output. It is shown in the truth table.



Full Adder



Inputs			Outputs	
A	B	C	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Truth Table of Full Adder

i) DATAFLOW MODEL

a) VHDL program:

```
library ieee;
use ieee.std_logic_1164.all;
entity FA_DFM is
port(a,b,ci:in std_logic;
s,co:out std_logic);
end FA_DFM;
architecture f1 of FA_DFM is
begin
        s<=a xor b xor ci;
        co<= (a and b) or (b and ci) or (a and ci);
end f1;
```

Test Bench Code for Dataflow Modeling(VHDL):-

```
------- insert stimulus here--------

   a<='0';b<='0';ci<='1'; wait for 100 ns;
   a<='0';b<='1';ci<='0'; wait for 100 ns;
   a<='0';b<='1';ci<='1'; wait for 100 ns;
   a<='1';b<='0';ci<='0'; wait for 100 ns;
   a<='1';b<='0';ci<='1'; wait for 100 ns;
   a<='1';b<='1';ci<='0'; wait for 100 ns;
   a<='1';b<='1';ci<='1'; wait for 100 ns;
```

b) Verilog program:

```
module fa_dfm(a,b,ci,s,co);
input a,b,ci;
output s,co;
assign s=a^b^ci;

assign co=(a&b)|(b&ci)|(a&ci);
endmodule
```

Test Bench Code for Dataflow modeling (Verilog):-

```
///////// Add stimulus here///////////////

                    a = 0;b = 0;ci = 1; #100;
                    a = 0;b = 1;ci = 0; #100;
                    a = 0;b = 1;ci = 1; #100;
                    a = 1;b = 0;ci = 0; #100;
                    a = 1;b = 0;ci = 1; #100;
                    a = 1;b = 1;ci = 0; #100;
                    a = 1;b = 1;ci = 1; #100;
```

ii) BEHAVIORAL MODEL

a) VHDL program:

```
library ieee;
use ieee.std_logic_1164.all;
```

```
entity FA_BM is

port(din:instd_logic_vector(2 downto 0);
dout:outstd_logic_vector(1 downto 0));
end FA_BM;
architecture f2 of FA_BM is
begin
        process (din)
        begin
                case din is
                        when "000"=>dout<="00";
                        when "001"=>dout<="10";
                        when "010"=>dout<="10";
                        when "011"=>dout<="01";
                        when "100"=>dout<="10";
                        when "101"=>dout<="01";
                        when "110"=>dout<="01";
                        when "111"=>dout<="11";
                        when others=>null;
                end case;
        end process;
end f2;
```

<u>Test Bench Code for Behavioral Modeling(VHDL):-</u>

```
------- insert stimulus here--------

din<="001"; wait for 100 ns;
        din<="010"; wait for 100 ns;
                 din<="011"; wait for 100 ns;
                 din<="100"; wait for 100 ns;
                 din<="101"; wait for 100 ns;
                 din<="110"; wait for 100 ns;
                 din<="111"; wait for 100 ns;
```

b) <u>Verilog program:</u>

```
module fa_bm(din,dout);
input [2:0] din;
output reg [1:0] dout;
always @(din)
```

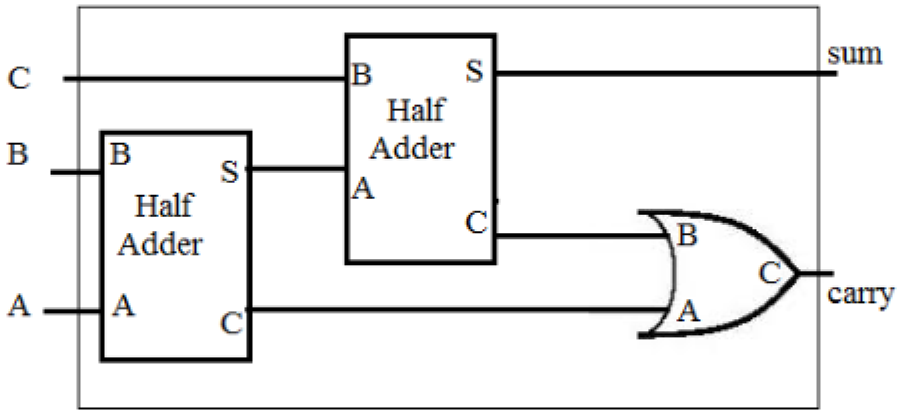
```
begin
          case (din)
                    3'b000:dout=2'b00;
                    3'b001:dout=2'b10;
                    3'b010:dout=2'b10;
                    3'b011:dout=2'b01;
                    3'b100:dout=2'b10;
                    3'b101:dout=2'b01;

                    3'b110:dout=2'b01;
                    3'b111:dout=2'b11;
          endcase
end
endmodule
```

Test Bench Code for Structural modeling (Verilog):-

///////// Add stimulus here///////////////

```
                    din = 3'b001; #100;
                    din = 3'b010; #100;
                    din = 3'b011; #100;
                    din = 3'b100; #100;

                    din = 3'b101; #100;
                    din = 3'b110; #100;
                    din = 3'b111; #100;
```

iii) STRUCTURAL MODEL



Full Adder Using half Adders

a) VHDL program:

```
library ieee;
use ieee.std_logic_1164.all;
entity HA_DF is
port (x,y:in std_logic;
sm,cy:out std_logic);
end HA_DF;
architecture f4 of HA_DF is
begin
        sm<=x xor y;
        cy<=x and y;
end f4;

library ieee;
use ieee.std_logic_1164.all;
entity OR_E is
port (p,q:in std_logic;
     r:out std_logic);
end OR_E;
architecture OR_A of OR_E is

begin
        r<=p or q;
end OR_A;

library ieee;
use ieee.std_logic_1164.all;

entity FA_SM is
port (a,b,ci:in std_logic;
s,co:out std_logic);

end FA_SM;
architecture f3 of FA_SM is
component HA_DF is
port (x,y:in std_logic;
sm,cy:out std_logic);
end component;
component or_e is
port (p,q:in std_logic;
    r:out std_logic);
end component;

signal t1,t2,t3:std_logic;
begin
        A1: HA_DF port map(a,b,t1,t2);
        A2: HA_DF port map(t1,ci,s,t3);
        A3: OR_E port map (co,t2,t3);
end f3;
```

<u>Test Bench Code for Structural Modeling(VHDL):-</u>

```
 ------- insert stimulus here--------

   a<='0';b<='0';ci<='1'; wait for 100 ns;
   a<='0';b<='1';ci<='0'; wait for 100 ns;
   a<='0';b<='1';ci<='1'; wait for 100 ns;
   a<='1';b<='0';ci<='0'; wait for 100 ns;
   a<='1';b<='0';ci<='1'; wait for 100 ns;
   a<='1';b<='1';ci<='0'; wait for 100 ns;
   a<='1';b<='1';ci<='1'; wait for 100 ns;
```

b) <u>Verilog program:</u>

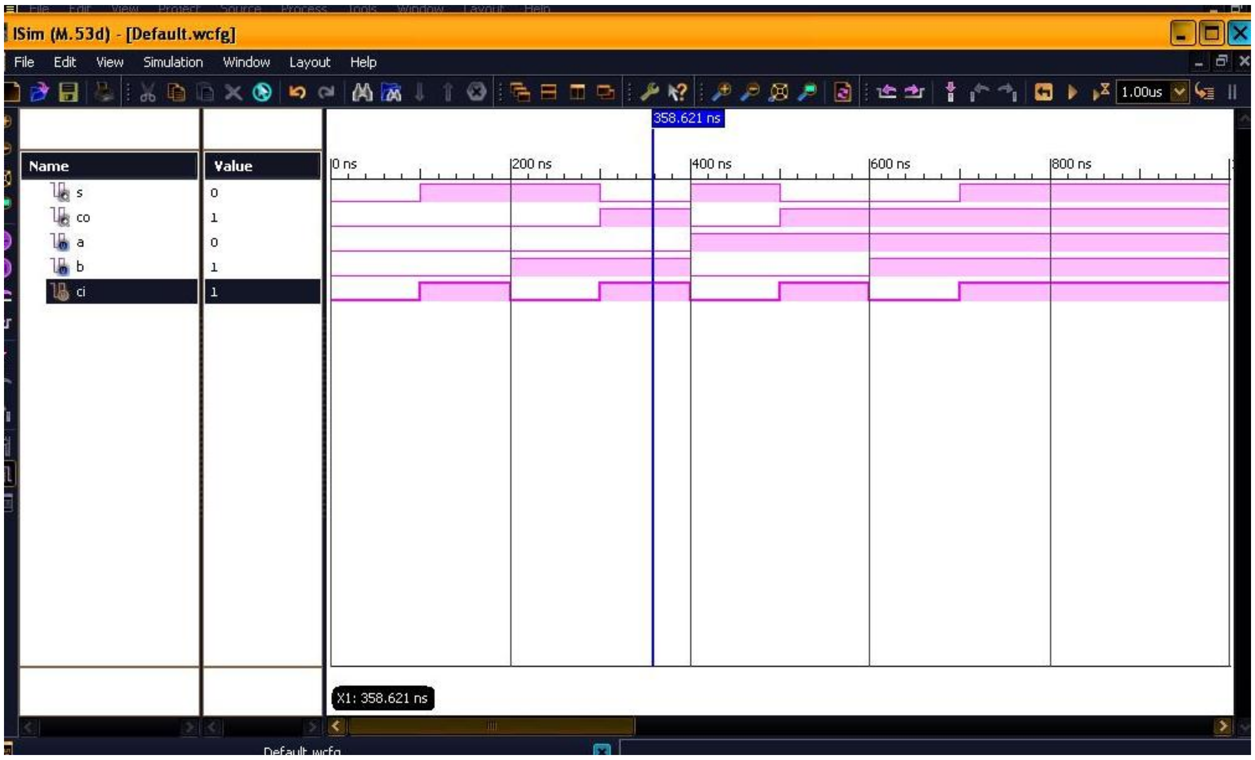
```
module ha(x,y,sm,cy);
input x,y;
output sm,cy;
assign sm=x^y;
assign cy=x&y;
endmodule
module or_m (p,q,r);
input p,q;
output r;

assign r=p | q;
endmodule
module fa(a,b,ci,s,co);

input a,b,ci;
output s,co;
wire t1,t2,t3;
```

```
ha i1(a,b,t1,t2);
ha i2(t1,ci,s,t3);
or_m i3(t2,t3,co);
endmodule
```

Test Bench Code for Structural modeling (Verilog):-

```
///////// Add stimulus here///////////////

a = 0;b = 0;ci = 1; #100;
                    a = 0;b = 1;ci = 0; #100;
                    a = 0;b = 1;ci = 1; #100;
                    a = 1;b = 0;ci = 0; #100;
                    a = 1;b = 0;ci = 1; #100;
                    a = 1;b = 1;ci = 0; #100;
                    a = 1;b = 1;ci = 1; #100;
```

Results and Simulation Outputs: The Full adder circuit is designed and verified using test bench.



EXPERIMENT NO. :3

Write a model for 16 bit ALU using the 4bit opcodes; the requisite functions can be defined for the chosen opcodes.

AIM

To Write a model for 16 bit ALU using the 4bit opcodes; the requisite functions can be defined for the chosen opcodes.

Software Required : Xilinx ISE 12.1 , ISim simulator

Theory:

The component that performs the arithmetic and logical operations is known as the Arithmetic Logic Unit, or ALU. The ALU is one of the most important components in a microprocessor. An ALU should be able to perform functions like

– logical and function
– logical or function
– arithmetic add function
– arithmetic subtract function etc.

a) VHDL program:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity alu16 is
port(a,b:instd_logic_vector(15 downto 0);
en:in std_logic;
op:instd_logic_vector(3 downto 0);
alu:outstd_logic_vector(31 downto 0));
end alu16;
architecture a1 of alu16 is
begin
process(a,b,en,op)
variable t1,t2:std_logic_vector(31 downto 0);

begin
        t1:=(others=>'0');
        t2:=(others=>'0');
        t1(15 downto 0):=a;
        t2(15 downto 0):=b;
        if (en='1') then
                case (op) is
                        when "0001"=>alu<=t1+t2;
                        when "0010"=>alu(15 downto 0)<=a-b;
                        when "0011"=>alu(15 downto 0)<=not a;
                        when "0100"=>alu<=a*b;
                        when "0101"=>alu(15 downto 0)<=a and b;
                        when "0110"=>alu(15 downto 0)<=a or b;
                        when "0111"=>alu(15 downto 0)<=a nand b;
                        when "1000"=>alu(15 downto 0)<=a xor b;
                        when others=>null;
                end case;
        else alu<=(others=>'Z');
        end if;
end process;
end a1;
```

<u>Test Bench Code (VHDL):-</u>

```
-------- insert stimulus here----------

                a <=" 0 000 0 000 0 000 0111";
                b <= "0000 0000 000 00101"; wait for 100 ns;

        en<= '1';op <= "0001"; wait for 100 ns;
                en<= '1';op <= "0010"; wait for 100 ns;
                en<= '1';op <= "0011"; wait for 100 ns;
                en<= '1';op <= "0100"; wait for 100 ns;
                en<= '1';op <= "0101"; wait for 100 ns;
                en<= '1';op <= "0110"; wait for 100 ns;
                en<= '1';op <= "0111"; wait for 100 ns;
                en<= '1';op <= "1000"; wait for 100 ns;
```

b) <u>Verilog program:</u>

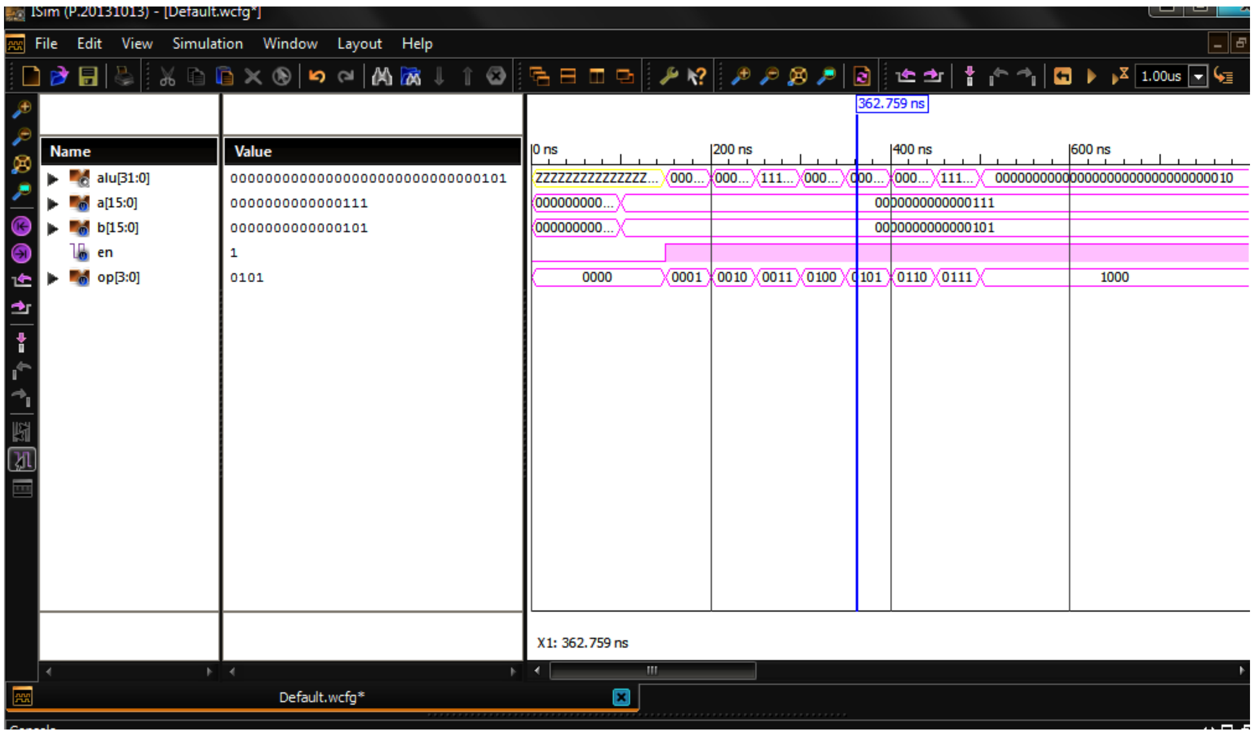
```
module alu16(a,b,en,op,alu);
input [15:0] a,b;
input en;
input [3:0] op;
output reg [31:0] alu;

always @(a,b,en,op)
begin
        if (en==1'b1)
                case (op)
                        4'b0001:alu=a+b;
                        4'b0010:alu=a-b;
                        4'b0011:alu=~a;
                        4'b0100:alu=a*b;
                        4'b0101:alu=a&b;
                        4'b0110:alu=a|b;
                        4'b0111:alu=~(a&b);
                        4'b1000:alu=a^b;
                        default:alu=32'b0;
                 endcase
        else alu=32'bZ;
end
endmodule
```

Test Bench Code for Structural modeling (Verilog):-

```
    --Add stimulus here--

                a = 16'b0000000000000000111;
                b = 16'b0000000000000000101; #50;

                en = 1;op = 4'b0001; #50;
                en = 1;op = 4'b0010; #50;
                en = 1;op = 4'b0011; #50;
                en = 1;op = 4'b0100; #50;
                en = 1;op = 4'b0101; #50;
                en = 1;op = 4'b0110; #50;
                 en = 1;op = 4'b0111; #50;
                en = 1;op = 4'b1000; #50;
```

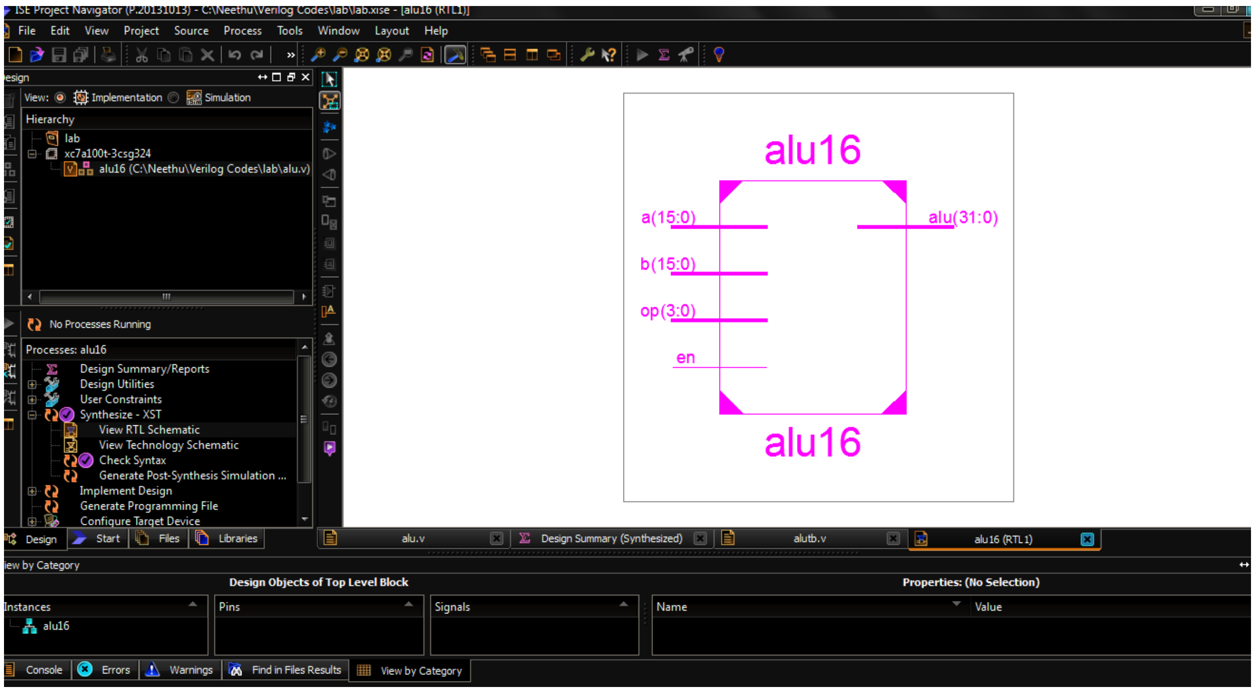
Results and Simulation Outputs

The 16 bit ALu has been designed using HDL code and verified by using test bench.


ISim (P.20131013) - [Default.wcfg*]
File Edit View Simulation Window Layout Help
1.00us
Name
Value
362.759 ns
0 ns
200 ns
400 ns
600 ns
X1: 362.759 ns
Default.wcfg*

```
alu[31:0]   00000000000000000000000000000101
a[15:0]     0000000000000111
b[15:0]     0000000000000101
en          1
op[3:0]     0101
```

RTL Schematic


ISE Project Navigator (P.20131013) - C:\Neethu\Verilog Codes\lab\lab.xise - [alu16 (RTL1)]
File Edit View Project Source Process Tools Window Layout Help
View: Implementation Simulation
Hierarchy
lab
xc7a100t-3csg324
alu16 (C:\Neethu\Verilog Codes\lab\alu.v)
No Processes Running
Processes: alu16
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation ...
Implement Design
Generate Programming File
Configure Target Device
alu16
a(15:0)
b(15:0)
op(3:0)
en
alu(31:0)
alu16
Design Start Files Libraries
alu.v
Design Summary (Synthesized)
alutb.v
alu16 (RTL1)
Design Objects of Top Level Block
Properties: (No Selection)
Instances
alu16
Pins
Signals
Name
Value
Console Errors Warnings Find in Files Results View by Category

EXPERIMENT NO. 4

Write an HDL program for the following designs:

a. Decoder
b. Encoder (without priority and with priority)
c. Multiplexer and De multiplexer

AIM

To write an HDL program in behavioural model for a 8:3 encoder, 3:8 decoder, 4:1 MUX and 1:4 DEMUX and to verify the output.

Software Required : Xilinx ISE 12.1 , ISim simulator

Theory:

An encoder is a combinational circuit that converts binary information in the form of a 2N input lines into N output lines, which represent N bit code for the input. For simple encoders, it is assumed that only one input line is active at a time.

A priority encoder is an encoder circuit in which inputs are given priorities. When more than one inputs are active at the same time, the input with higher priority takes precedence and the output corresponding to that is generated.

A decoder does the opposite job of an encoder. It is a combinational circuit that converts n lines of input into 2n lines of output.

MUX is a combinational circuit which have many data inputs and single output depending on control or select inputs. For N input lines, log n (base2) selection lines, or we can say that for 2n input lines, n selection lines are required.

A demultiplexer (or demux) is a device that takes a single input line and routes it to one of several digital output lines. A demultiplexer of 2n outputs has n select lines, which are used to select which output line to send the input. A demultiplexer is also called a data distributor.

1) DECODER

a) VHDL program

```
library ieee;
use ieee.std_logic_1164.all;
entity decoder3_8 is

port(a:in std_logic_vector(2 downto 0);
en:in std_logic;
y:out std_logic_vector(7 downto 0));
end decoder3_8;
```

```
architecture  beha of decoder3_8 is
 begin
process(a,en)
begin
if(en='1')  then
case a is

when "000"=>y<="00000001"; when "001"=>y<="00000010";
when "010"=>y<="00000100"; when "011"=>y<="00001000";
when "100"=>y<="00010000"; when "101"=>y<="00100000";
 when "110"=>y<="01000000"; when "111"=>y<="10000000";
when others=>y<="XXXXXXXX";
end case;

else y<="XXXXXXXX”; end if;

end process;
end beha;
```

Test Bench Code (VHDL):-

```
------ insert stimulus here-------
                    en<='1';a<=”000”; wait for 100 ns;
                    en<='1';a<=”001”; wait for 100 ns;
                    en<='1';a<=”110”; wait for 100 ns;
                    en<='1';a<=”011”; wait for 100 ns;
                    en<='1';a<=”100”; wait for 100 ns;
                    en<='1';a<=”110”; wait for 100 ns;
                    en<='1';a<=”111”; wait for 100 ns;
                    en<='0';a<=”011”; wait for 100 ns;
```

b) Verilog program:

```
module dec3to8(en,a,y);
input en;
input [3:0]a;
output [7:0]y;
reg[7:0]y;

always @(a,en)
begin
if(en)
```

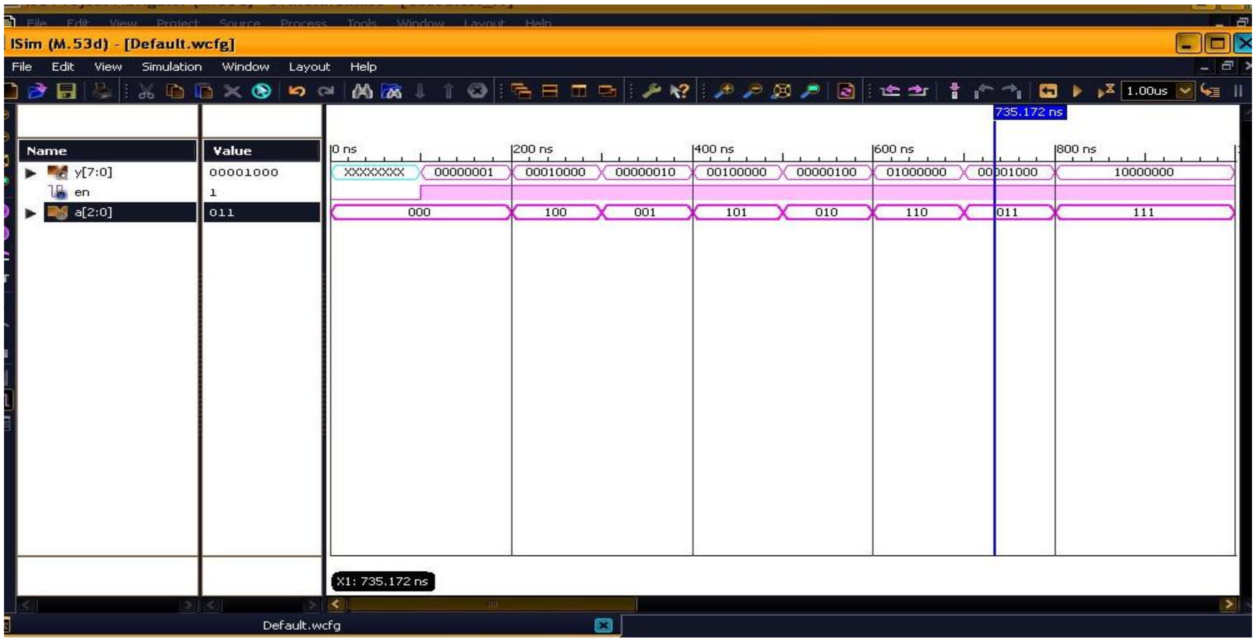
```
case(a)

3’b000: y = 8’b00000001;
3’b001: y = 8’b00000010;
3’b010: y = 8’b00000100;
3’b011: y = 8’b00001000;
3’b100: y = 8’b00010000;
3’b101: y = 8’b00100000;
3’b110: y = 8’b01000000;
3’b111: y = 8’b10000000;
default :y=8’bXXXXXXXX;
endcase
else
y= 8’bXXXXXXXX;
end
endmodule
```

Test Bench Code (Verilog):-

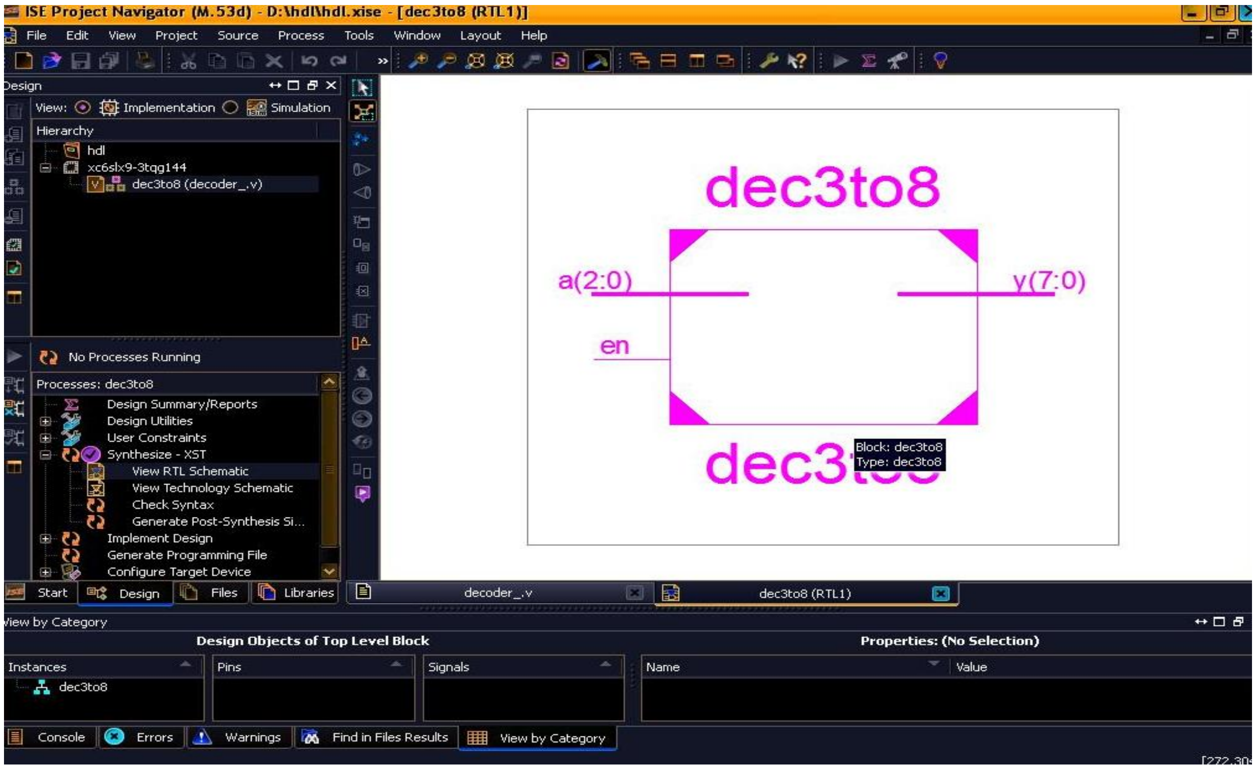
[redacted]

```
//////// Add stimulus here////////
                    en=1;a=3'b000; #100;            en=1;a=3'b100; #100;
                    en=1;a=3'b001; #100;            en=1;a=3'b101; #100;
                    en=1;a=3'b010; #100;            en=1;a=3'b110; #100;

                              en=1;a=3'b011; #100;  en=1;a=3'b111; #100;
```

Simulation Output:



RTL Schematic



2) <u>ENCODER (8:3)</u>

VHDL program:

```
library ieee;
use ieee.std_logic_1164.all;
entity encoder8_3 is
port(a:in std_logic_vector(7 downto 0);
en:in std_logic;
y:out std_logic_vector(2 downto 0));
end encoder8_3;
architecture beha of encoder8_3 is
begin
  process(a,en)
begin
if(en='1')then
case a is
when "00000001"=>y<="000";
when "00000010"=>y<="001";
when "00000100"=>y<="010";
when "00001000”=>y<="011";
when "00010000"=>y<="100";
when "00100000"=>y<="101";
when "01000000"=>y<="110";
when "10000000"=>y<="111";
when others =>y<="XXX";
end case;
else
y<="XXX";
end if;
end process;
end beha;
```

Test Bench Code (VHDL):-

```
-------- insert stimulus here--------
                  en<='1';a<=”000000001”;wait for 100 ns;
                  en<='1';a<=”000000010”;wait for 100 ns;
                  en<='1';a<=”000000100”;wait for 100 ns;
                  en<='1';a<=”000001000”;wait for 100 ns;
                  en<='1';a<=”0000100000”;wait for 100 ns;
                  en<='1';a<=”0001000000”;wait for 100 ns;
                  en<='1';a<=”001000000”;wait for 100 ns;
                  en<='1';a<=”010000000”;wait for 100 ns;
                  en<='1';a<=”100000000”;wait for 100 ns;
```

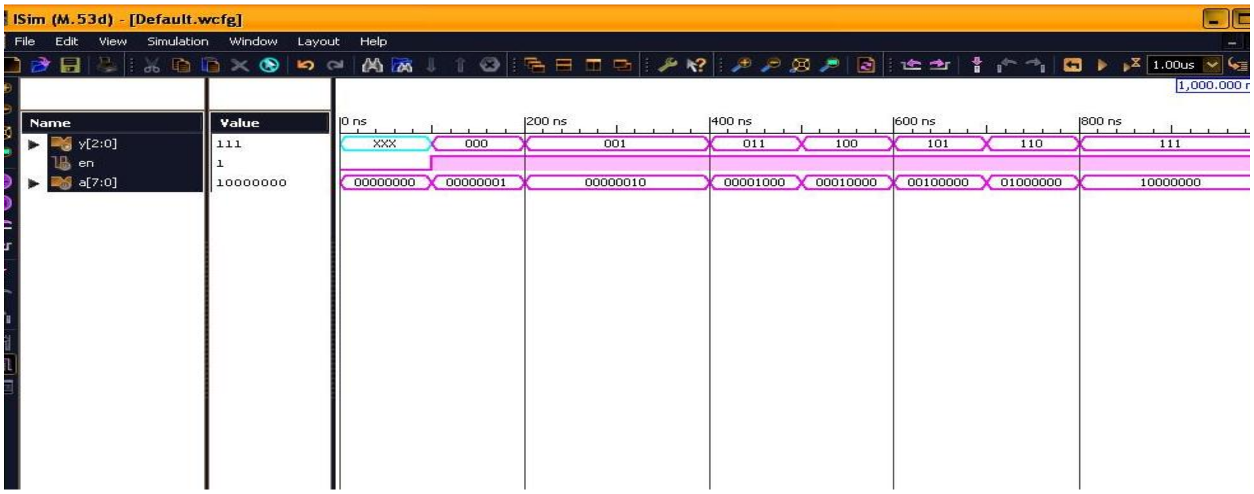
b) <u>Verilog program:</u>

```
module encoder8_3(en,a,y);
input en;
input [7:0]a;
output [2:0]y;
reg[2:0]y;
always @(a,en)
begin
if(en)
case(a)
8’b00000001: y = 3’b000;
8’b00000010: y = 3’b001;
8’b00000100: y = 3’b010;
8’b00001000: y = 3’b011;
8’b00010000: y = 3’b100;
8’b00100000: y = 3’b101;
8’b01000000: y = 3’b110;
8’b10000000: y = 3’b111;
default:y=3’bXXX;
endcase
else
Y=3’bXXX;
end
endmodule
```

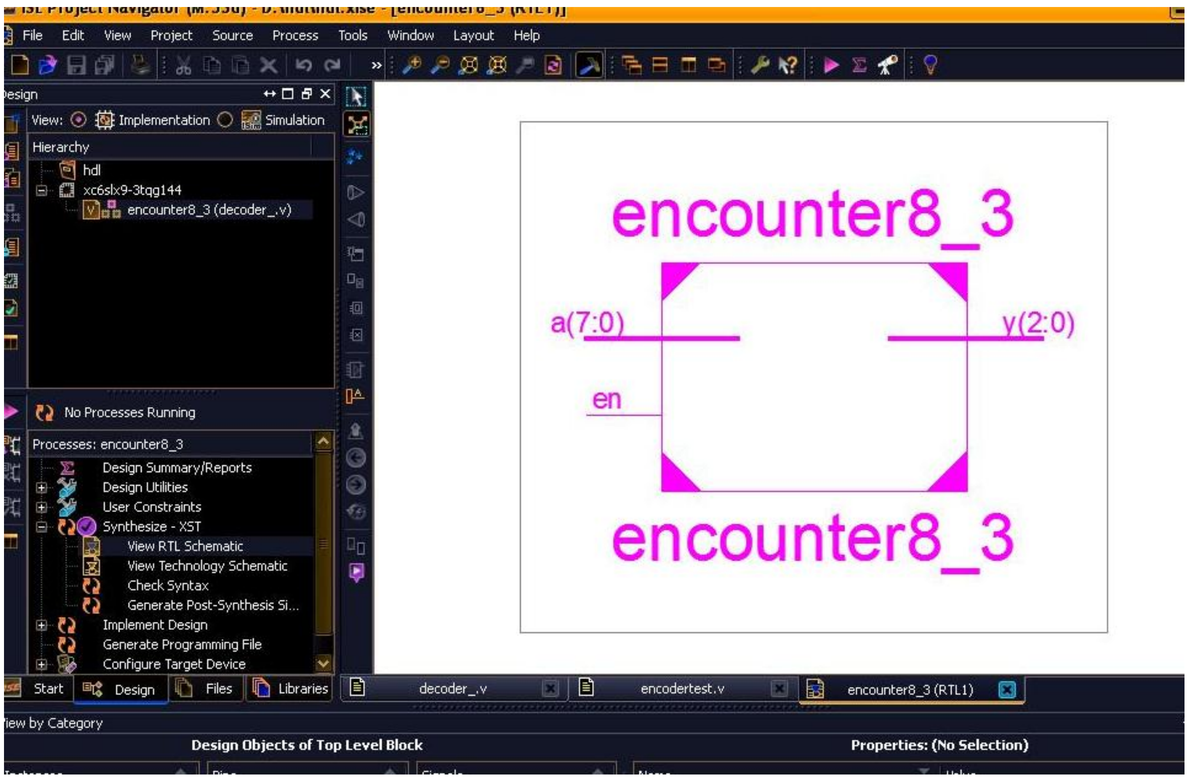
<u>Test Bench Code (Verilog):-</u>

```
//////// Add stimulus here////////

        en=1;a=8'b00000001; #100;
        en=1;a=8'b00000010; #100;
        en=1;a=8'b000000100; #100;
        en=1;a=8'b00001000; #100;
        en=1;a=8'b00010000; #100;
        en=1;a=8'b00100000; #100;
        en=1;a=8'b01000000; #100;
        en=1;a=8'b10000000; #100;
```

Simulation Output:



RTL Schematic



3) PRIORITY ENCODER (8:3)

VHDL CODE

library ieee;
 use ieee.std_logic_1164.all;
entity priencoder is
port(a:in std_logic_vector(7 downto 0);
en:in std_logic;

```
y:out std_logic_vector(2 downto 0));
end priencoder;
architecture beha of priencoder is
begin
 process(a,en)
begin
if (a(7)=’1’)then
y<=”111”;
elsif(a(6)=’1’)then
y<=”110”;
elsif(a(5)=’1’)then
y<=”101”;
elsif(a(4)=’1’)then
y<=”100”;
elsif(a(3)=’1’)then
y<=”011”;
elsif(a(2)=’1’)then
y<=”010”;
elsif(a(1)=’1’)then
y<=”001”;
elsif(a(0)=’1’)then
y<=”000”;
else
y<=”000”;
else
y<=”XXX”;
end if;
end process;
end beha;
```

Test Bench Code (VHDL):-

```
-------- insert stimulus here--------
en<='1';a<=”000000001”;wait for 100 ns;
en<='1';a<=”001000010”;wait for 100 ns;
en<='1';a<=”100000100”;wait for 100 ns;
en<='1';a<=”000001000”;wait for 100 ns;
en<='1';a<=”0000100000”;wait for 100 ns;
en<='1';a<=”0001011000”;wait for 100 ns;
en<='1';a<=”001000000”;wait for 100 ns;
en<='1';a<=”010000000”;wait for 100 ns;
en<='1';a<=”100001000”;wait for 100 ns;
```

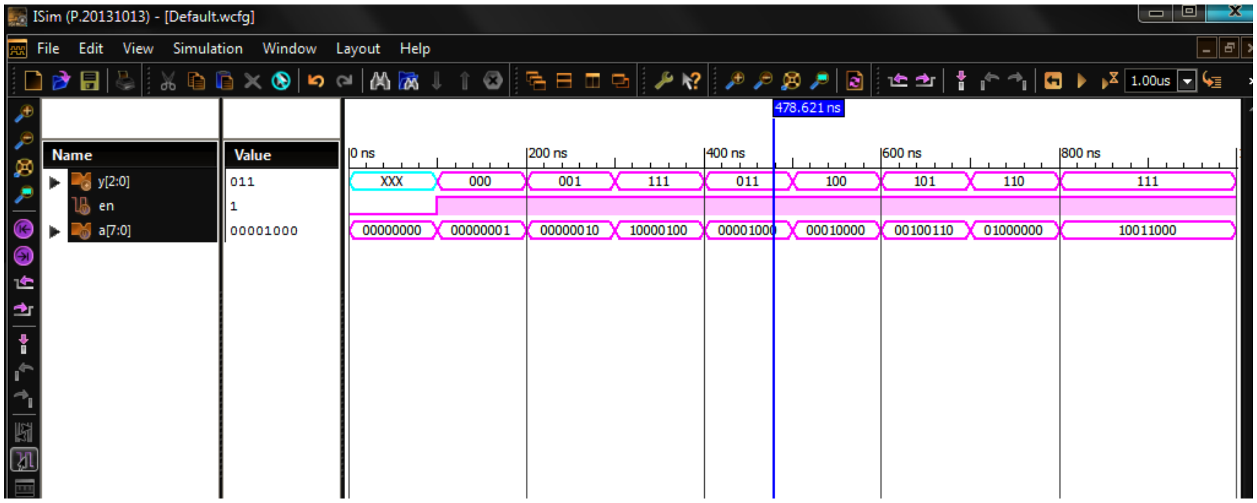
b) <u>Verilog program priority encoder</u>

```
module priencoder(en,a,y);
input en;
input [7:0]a; output [2:0]y; reg[2:0]y;
always @(a,en)
begin
 if(en==1)
 begin
 if(a[7]==1) y=3'b111;
else if(a[6]==1) y=3'b110;
else if(a[5]==1) y=3'b101;
else if(a[4]==1) y=3'b100;
 else if(a[3]==1) y=3'b011;
else if(a[2]==1) y=3'b010;
else if(a[1]==1) y=3'b001;
else if(a[0]==1) y=3'b000;
else y=3'bXXX;
 end
 else
 y=3'bXXX;
 end
 endmodule
```

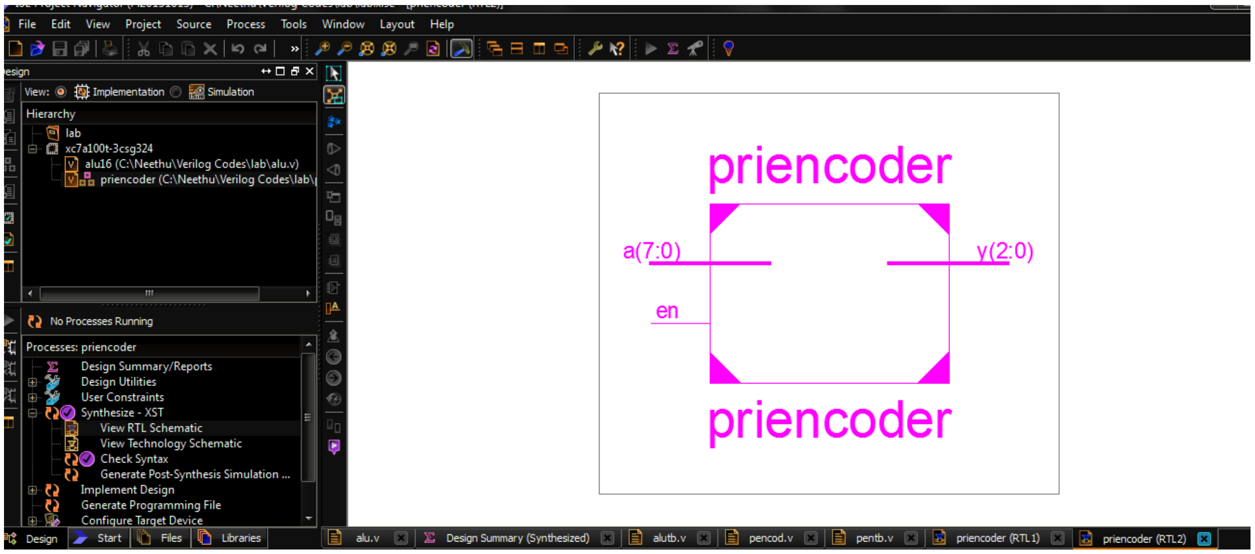
<u>Test Bench Code (Vlog):-</u>

```
en=1;a=8'b00000001; #100;
en=1;a=8'b00000010; #100;
en=1;a=8'b110000100; #100;
en=1;a=8'b00001000; #100;
en=1;a=8'b00010000; #100;
en=1;a=8'b00100110; #100;
en=1;a=8'b01000000; #100;
en=1;a=8'b10011000; #100;
```

Simulation Output:



RTL Schematic:



4) 4:1 MULTIPLEXER

a) VHDL program:

```
library ieee;
use ieee.std_logic_1164.all;
entity mux is
port(i3,i2,i1,i0 : in std_logic;
  sel:in std_logic_vector(1 downto 0);
y:out std_logic);
end mux;
architecture m1 of mux is
begin
```

```
 process(i0,i1,i2,i3,sel)
 begin
 case sel is
  when”00”=>y<=i0;
when”01”=>y<=i1;
when”10”=>y<=i2;
when”11”=>y<=i3;
when others=>y<=’X’;
end case;
end process;
end m1;
```

<u>*Test Bench Code (VHDL):-*</u>

```
-------- insert stimulus here----------
              i3<='1';i2<='0';i1<='1';i0<='0';wait for 100 ns;
              sel<=”00” wait for 100 ns;
              sel<=”01” wait for 100 ns;
              sel<=”10” wait for 100 ns;
              sel<=”11” wait for 100 ns;
```

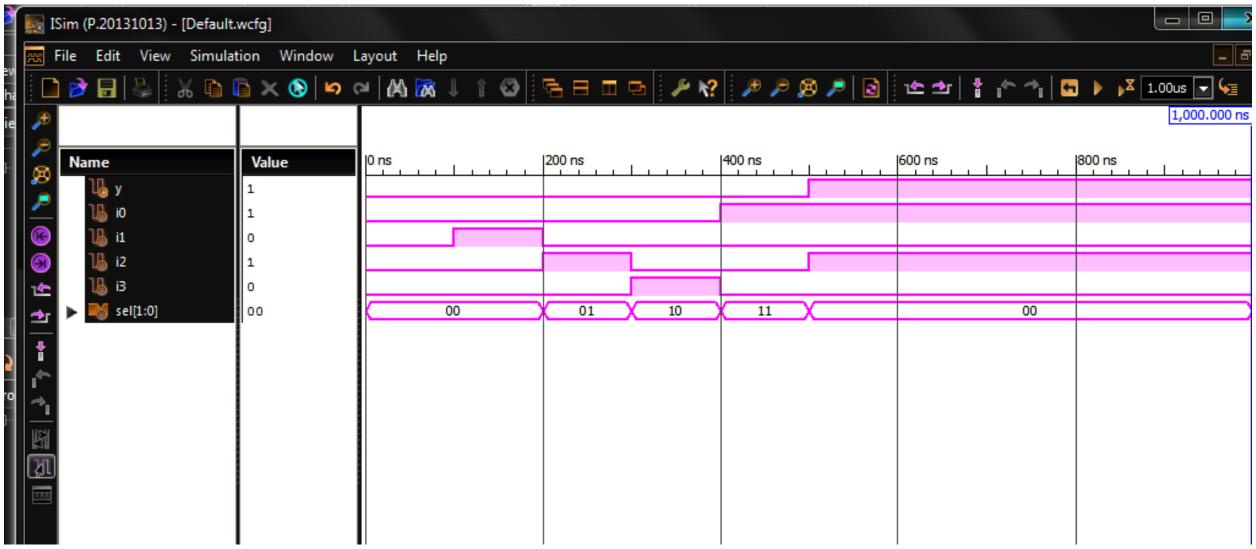
b) <u>Verilog program:</u>

```
module mux4_1(i0,i1,i2,i3,sel,y);
input i0,i1,i2,i3;
input[1:0]sel;
output reg y;

always @(i0,i1,i2,i3,sel)
begin
case(sel)
2’b00:y=i0;
2’b01:y=i1;
2’b10:y=i2;
2’b11:y=i3;
endcase
end
endmodule
```

Test Bench Code (Verilog):-

```
///////// Add stimulus here///////////

            sel=2'b00;i3 = 0;i2 = 0;i1 = 1;i0 = 0; #100;
            sel=2'b01;i3 = 0;i2 = 1;i1 = 0;i0 = 0; #100;
            sel=2'b10;i3 = 1;i2 = 0;i1 = 0;i0 = 0; #100;
            sel=2'b11;i3 = 0;i2 = 0;i1 = 0;i0 = 1; #100;
            sel=2'b00;i3 = 0;i2 = 1;i1 = 0;i0 = 1; #100;
```

Simulation Output and Result:



5) DEMULTIPLEXER (1:4)

a) VHDL program:

```
library ieee;
use ieee.std_logic_1164.all;

entity demux is
port (din:in std_logic;
  sel : std_logic_vector(1 downto 0);
  d : out std_logic_vector(3 downto 0));
end demux;
architecture dm1 of demux is
begin
process(din,sel)
begin
```

```
case (sel)is

 when “00”=> d(0)<=din;
             d(1)<=’X’;
             d(2)<=’X’;
             d(3)<=’X’;
when “01”=> d(0)<=’X’;
              d(1)<=din;
             d(2)<=’X’;
             d(3)<=’X’;
when “10”=> d(0)<=’X’;
             d(1)<=’X’;
              d(2)<=din;
             d(3)<=’X’;
when “11”=> d(0)<=’X’;
             d(1)<=’X’;
              d(2)<=’X’;
             d(3)<=din;
end case;
end process;
end dm1;
```

<u>Test Bench Code (VHDL):-</u>

-------- *insert stimulus here*---------

```
              din<=’1’;din<=’0’;din<=’0’;din<=’0’;
              Sel<= “00”; wait for 100 ns;
              Sel <=”01”; wait for 100 ns;
              Sel <=”10”;wait for 100 ns;
              Sel<=” 11”; wait for 100 ns;
```

b) <u>Verilog program:</u>

```
module demux(i,sel,y3,y2,y1,y0);
input i;
input [1:0]sel;
output y3,y2,y1,y0;
reg y0,y1,y2,y3;
always @(i,sel)
begin
```

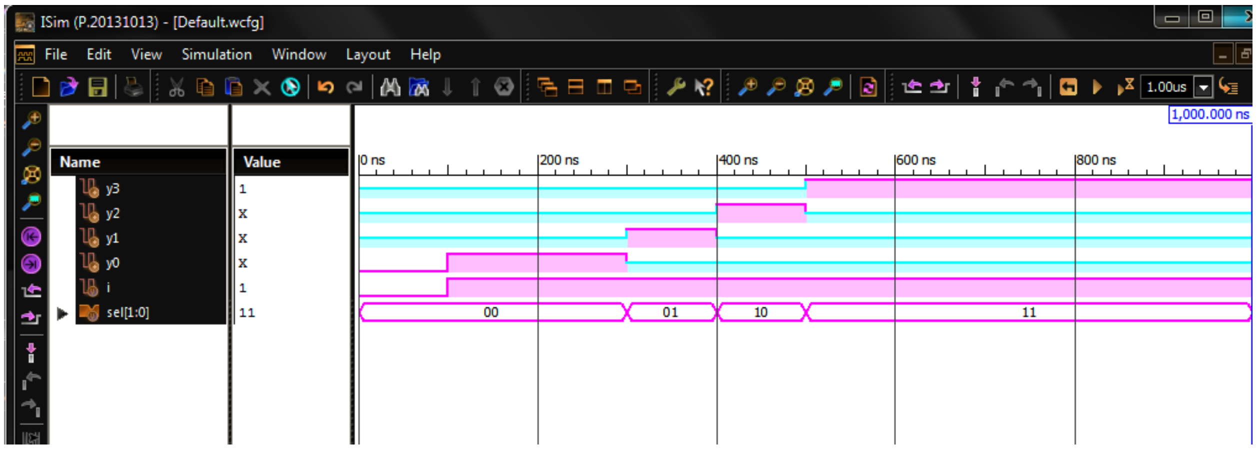
```
case(sel)
2'b00:begin
y0=i;y1=1'bX;y2=1'bX;y3=1'bX;end
2'b01:begin
y0=1'bX;y1=i;y2=1'bX;y3=1'bX;end
2'b10:beginy0=1'bX;y1=1'bX;y2=i;y3=1'bX;end
2'b11:beginy0=1'bX;y1=1'bX;y2=1'bX;y3=i;end
default: begin y0=1'bX;y1=1'bX;y2=1'bX;y3=1'bX; end
endcase
end
endmodule
```

Test Bench Code (Verilog):-

///////// Add stimulus here///////////////

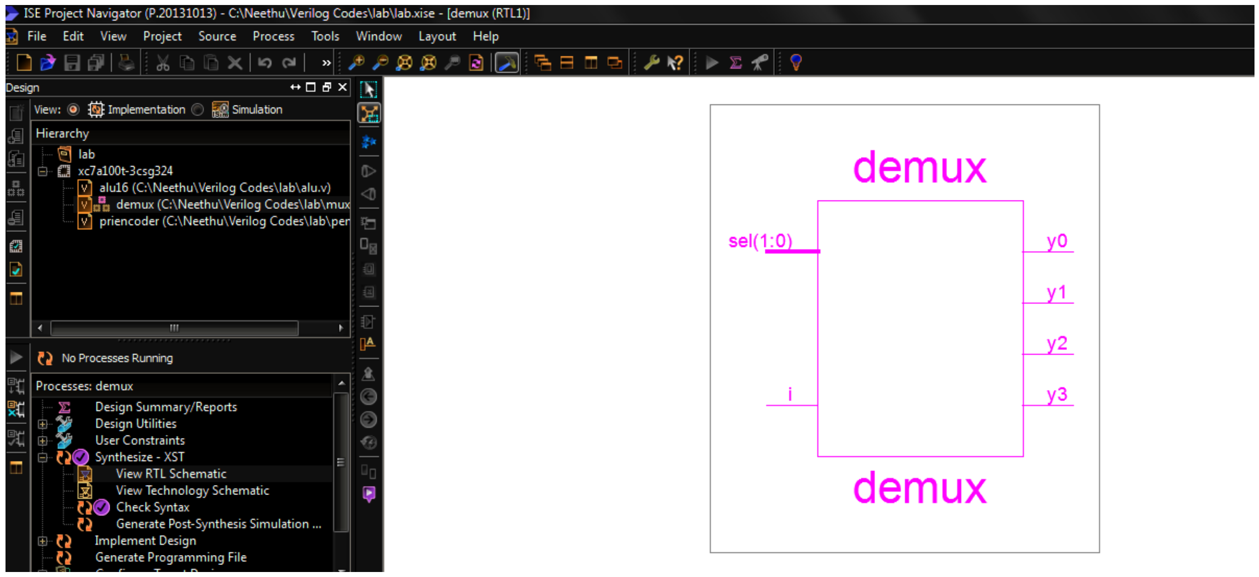
```
        i = 1; #100;
        sel=00;#100;
        sel = 01;#100;
        sel = 10;#100;
        sel = 11;#100;
```

Simulation Output and Results:


ISim (P.20131013) - [Default.wcfg]
File Edit View Simulation Window Layout Help
1.00us
1,000.000 ns
0 ns
200 ns
400 ns
600 ns
800 ns
00
01
10
11

Name	Value
y3	1
y2	X
y1	X
y0	X
i	1
sel[1:0]	11

RTL Schematic:


ISE Project Navigator (P.20131013) - C:\Neethu\Verilog Codes\lab\lab.xise - [demux (RTL1)]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
lab
xc7a100t-3csg324
alu16 (C:\Neethu\Verilog Codes\lab\alu.v)
demux (C:\Neethu\Verilog Codes\lab\mux
priencoder (C:\Neethu\Verilog Codes\lab\per
No Processes Running
Processes: demux
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation ...
Implement Design
Generate Programming File
demux
sel(1:0)
i
y0
y1
y2
y3
demux

Result:

The combinational circuits decoder, encoder, MUX and DEMUX have been designed and verified using simulation.

EXPERIMENT NO :5

Write an HDL program for the following designs:

a. 4 bit Binary to Gray converter

b. 4-bit Binary Comparator

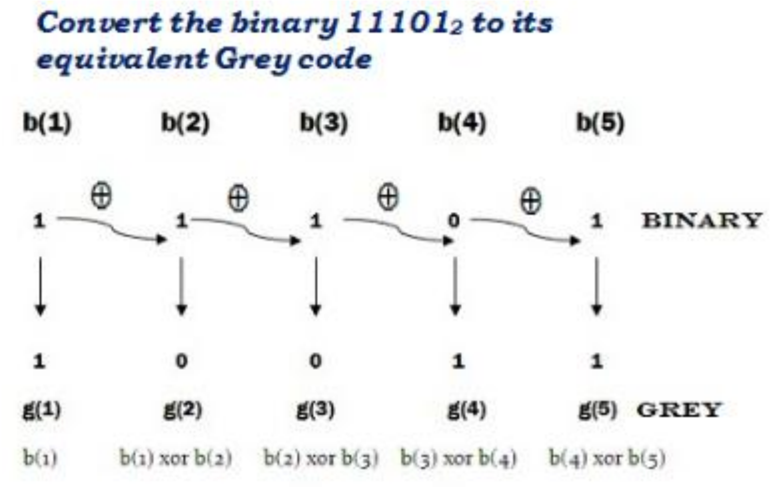
AIM

Write an HDL program for 4 bit Binary to Gray code converter and 4-bit Binary Comparator using dataflow modeling style.

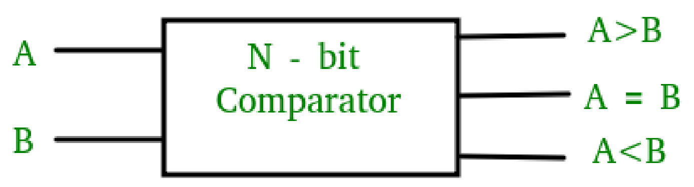
Software Required : Xilinx ISE 12.1 , ISim simulator

Theory:

Binary to gray code conversion method strongly follows the EX-OR gate operation between binary bits.


Convert the binary 11101₂ to its equivalent Grey code
b(1) b(2) b(3) b(4) b(5)
1 1 1 0 1 BINARY
1 0 0 1 1
g(1) g(2) g(3) g(4) g(5) GREY
b(1) b(1) xor b(2) b(2) xor b(3) b(3) xor b(4) b(4) xor b(5)

A magnitude digital Comparator is a combinational circuit that compares two digital or binary numbers in order to find out whether one binary number is equal, less than or greater than the other binary number. We logically design a circuit for which we will have two inputs one for A and other for B and have three output terminals, one for A > B condition, one for A = B condition and one for A < B condition.



1) 4 bit Binary to Gray converter

<u>VHDL program:</u>

```
library ieee;
use ieee.std_logic_1164.all;
entity btog is
port(b:instd_logic_vector(3 downto 0);
        g:out std_logic_vector(3 downto 0));
end btog;
architecture bg1 of btog is
begin
        g(3)<=b(3);
        g(2)<=b(3) xor b(2);
        g(1)<=b(2) xor b(1);
        g(0)<=b(1) xor b(0);

end bg1;
```

<u>Test Bench Code (VHDL):-</u>

```
------- insert stimulus here--------
                b<="1010"; wait for 100 ns;
                b<="1101"; wait for 100 ns;
                b<="1001"; wait for 100 ns;
                b<="1000"; wait for 100 ns;
```

b) <u>Verilog program:</u>

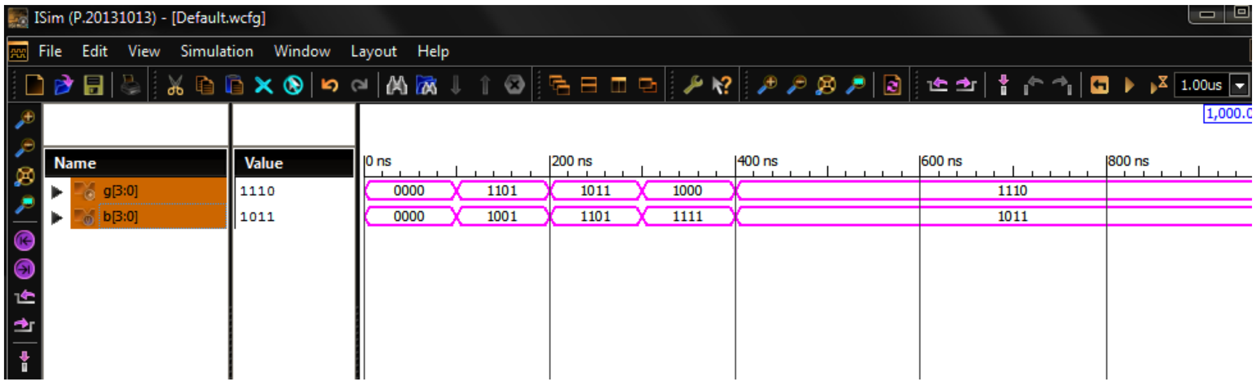
```
module btog(b,g);
input [3:0] b;
output [3:0] g;
assign g[3]=b[3];
assign g[2]=b[3]^b[2];
assign g[1]=b[2]^b[1];
assign g[0]=b[1]^b[0];
endmodule
```

<u>Test Bench Code (Verilog):-</u>

```
///////// Add stimulus here///////////////
```

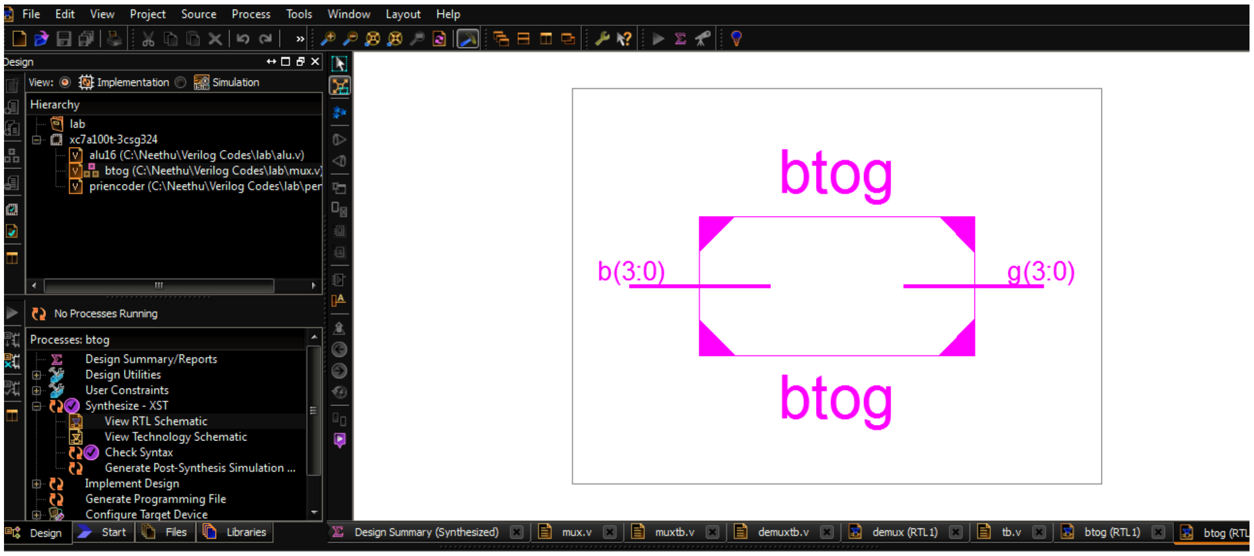
```
b=4'b1001; #100;
b=4'b1101; #100;
b=4'b1111; #100;
b=4'b1011; #100;
```

Simulation Output and Results:


ISim (P.20131013) - [Default.wcfg]
File Edit View Simulation Window Layout Help
1.00us
Name
Value
g[3:0]
b[3:0]
0 ns
200 ns
400 ns
600 ns
800 ns

Name	Value	0 ns				400 ns –
g[3:0]	1110	0000	1101	1011	1000	1110
b[3:0]	1011	0000	1001	1101	1111	1011

RTL Schematic:


File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
lab
xc7a100t-3csg324
alu16 (C:\Neethu\Verilog Codes\lab\alu.v)
btog (C:\Neethu\Verilog Codes\lab\mux.v)
priencoder (C:\Neethu\Verilog Codes\lab\per
No Processes Running
Processes: btog
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation ...
Implement Design
Generate Programming File
Configure Target Device
Design Start Files Libraries
Design Summary (Synthesized) mux.v muxtb.v demuxtb.v demux (RTL1) tb.v btog (RTL1) btog (RTL
btog
b(3:0)
g(3:0)
btog

2) 4 bit BINARY COMPARATOR

a) VHDL program:

```
library ieee;
use ieee.std_logic_1164.all;
entity comp is
```

```
port(a,b:instd_logic_vector(3 downto 0);
altb, aeqb, agtb:out std_logic);
end comp;

architecture c1 of comp is
signal x3,x2,x1,x0 : std_logic;
begin
x3 <= (a(3) xnor b(3));
x2 <= (a(2) xnor b(2));
x1 <= (a(1) xnor b(1));
x0 <= (a(0) xnor b(0));
aeqb<= (x3 and x2 and x1 and x0);
agtb<= ((a(3) and (not b(3))) or (a(2) and (not b(2)) and x3) or (a(1) and (not b(1)) and x3 and
x2) or (a(0) and (not b(0)) and x3 and x2 and x1));
altb<= (((not a(3)) and b(3)) or ((not a(2)) and b(2) and x3) or((not a(1)) and b(1) and x3 and x2)
or ((not a(0)) and b(0) and x3 and x2 and x1));

end c1;
```

Test Bench Code (VHDL):-

```
------- insert stimulus here--------

                a<="1010";b<="1100";wait for 100 ns;
                a<="1010";b<="1000";wait for 100 ns;
                a<="1100";b<="1100";wait for 100 ns;

                a<="1110";b<="1100";wait for 100 ns;
                a<="1000";b<="1100";wait for 100 ns;
                a<="1010";b<="1010";wait for 100 ns;
```

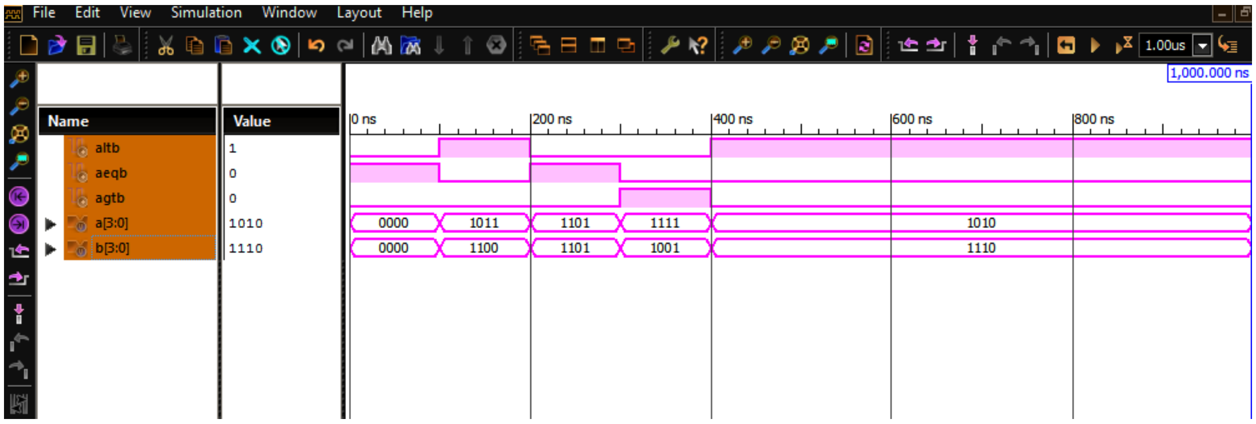
b) Verilog program:

```
module comp(a,b,altb,aeqb,agtb);
input [3:0] a,b;
output altb,aeqb,agtb;
wire x3,x2,x1,x0;

assign x3 = ~(a[3] ^ b[3]);
assign x2 = ~(a[2] ^ b[2]);
assign x1 = ~(a[1] ^ b[1]);
assign x0 = ~(a[0] ^ b[0]);

assign aeqb = (x3 & x2 & x1 & x0);
assign agtb = ((a[3] & (~ b[3])) | (a[2] & (~ b[2]) & x3) | (a[1] & (~ b[1]) & x3 & x2) | (a[0] &
(~b[0]) & x3 & x2 & x1));
assign altb = (((~ a[3]) & b[3]) | ((~ a[2]) & b[2] & x3) | ((~ a[1]) & b[1] & x3 & x2) |((~ a[0])
&b[0] & x3 & x2 & x1));

endmodule
```

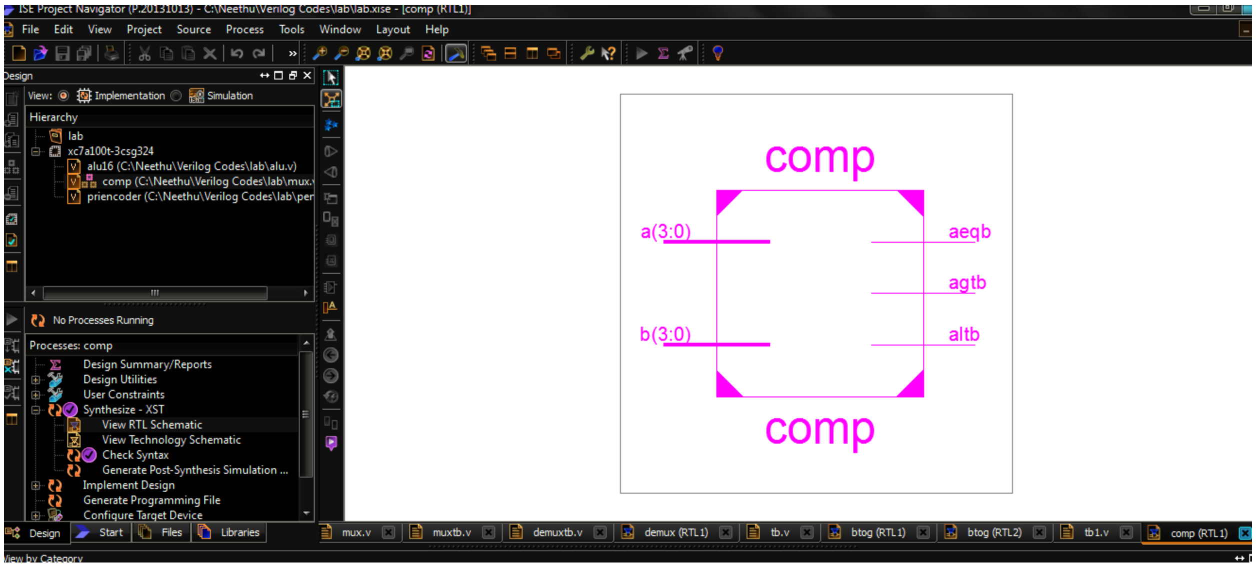
Test Bench Code (Verilog):-

```
///////// Add stimulus here///////////////

          a=4'b1011;b=4'b1100; #100;
          a=4'b1101;b=4'b1101; #100;
          a=4'b1111;b=4'b1001; #100;
          a=4'b1010;b=4'b1110; #100;
```

Simulation Output:



RTL Schematic:


ISE Project Navigator (P.20131013) - C:\Neethu\Verilog Codes\lab\lab.xise - [comp (RTL1)]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
lab
xc7a100t-3csg324
alu16 (C:\Neethu\Verilog Codes\lab\alu.v)
comp (C:\Neethu\Verilog Codes\lab\mux.v
priencoder (C:\Neethu\Verilog Codes\lab\per
No Processes Running
Processes: comp
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation ...
Implement Design
Generate Programming File
Configure Target Device
Design Start Files Libraries
mux.v muxtb.v demuxtb.v demux (RTL1) tb.v btog (RTL1) btog (RTL2) tb1.v comp (RTL1)
View by Category
comp
a(3:0)
b(3:0)
aeqb
agtb
altb
comp

Result

The HDL program for 4 bit Binary to Gray code converter and 4-bit Binary Comparator using dataflow model is written and verified using simulation.

EXPERIMENT NO. : 6

Develop the HDL code for the following flipflops: T, D, SR, JK

AIM

To write a HDL code for the flipflops: T, D, SR, JK using behavioural model.

Software Required : Xilinx ISE 12.1 , ISim simulator

Theory:

A flip-flop or latch is a circuit that has two stable states and can be used to store state information – a bistable multivibrator. The circuit can be made to change state by signals applied to one or more control inputs and will have one or two outputs. Flip-flops and latches are fundamental building blocks of digital electronics systems used in computers, communications etc. The truth table of various flip flops are given below:

S	**R**	**Action**
0	0	No change; random initial
1	0	Q = 1
X	1	Q = 0

J	**K**	**Q_{next}**	**Comment**
0	0	Q	No change
0	1	0	Reset
1	0	1	Set
1	1	Q	Toggle

Clock	D	Q_{next}
Rising edge	0	0
Rising edge	1	1
Non-rising	X	Q

T	Qn	Qn+1	Comment
0	0	0	Hold state (no clock)
0	1	1	Hold state (no clock)
1	0	1	Toggle
1	1	0	Toggle

i) <u>SR FLIP FLOP</u>

a) <u>VHDL program:</u>

```
library ieee;
use ieee.std_logic_1164.all;

entity srff_e is
port (clk, rst:in std_logic;
sr: in std_logic_vector(1 downto 0);
q,qbar: buffer std_logic);
end;
architecture srff_a of srff_e is
```

```
          begin

          process(clk, sr, rst)
                    begin
                    if(rst = '1') then
                    q<= '0';
                    qbar<= '1';

                    elsif(rising_edge(clk)) then
                              case (sr) is
                                        when "00" => q <= q; qbar<= qbar;
                                        when "01" => q <= '0'; qbar<= '1';
                                        when "10" => q <= '1'; qbar<= '0';
                                        when "11" => q <= 'X'; qbar<= 'X';
                                        when others => null;
                              end case;

                    end if;
          end process;
end srff_a;
```

Test Bench Code (VHDL):-

------- *insert stimulus here*--------

```
                    rst<='1';wait for 100 ns;rst<='0';
                    sr<="00"; wait for 100 ns;
                    sr<="01"; wait for 100 ns;
                    sr<="10"; wait for 100 ns;
                    sr<="11"; wait for 100 ns;
```

b) Verilog program:

```
module srff_m (sr,rst,clk,q,qbar);
input [1:0]sr;
input rst,clk;
output reg q,qbar;

always@ (posedge clk)
begin
```

```
        if (rst==1)
        begin q=1'b0; qbar=1'b1; end
        else
        case (sr)
        2'b00 : begin q=q;qbar=qbar; end
        2'b01 : begin q=1'b0; qbar=1'b1; end
        2'b10 : begin q=1'b1; qbar=1'b0; end
        2'b11 : begin q=1'bX; qbar=1'bX; end
        default : begin q=1'b0; qbar=1'b1; end
        endcase
end
endmodule
```

Test Bench Code (Verilog):-

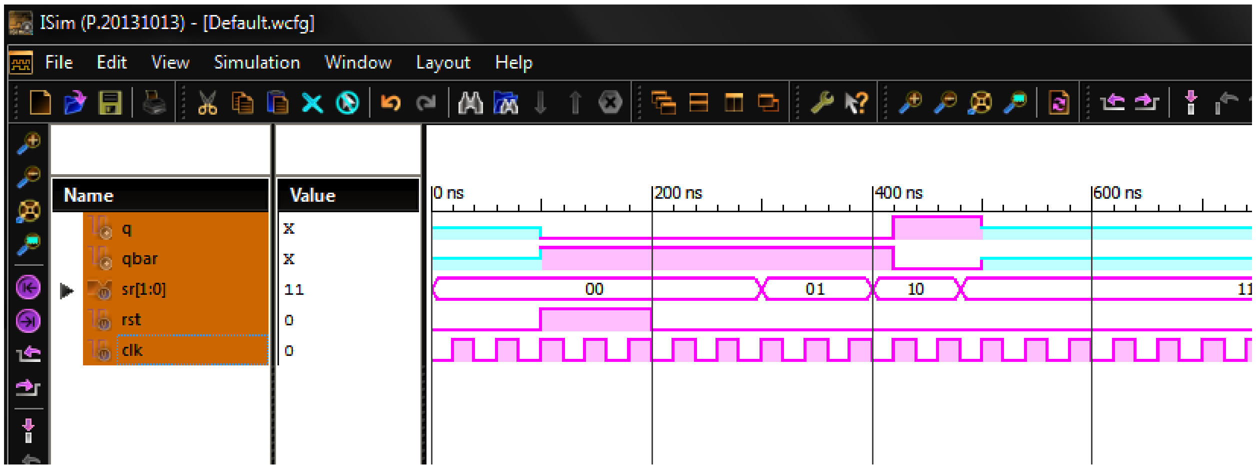
-------*Define the clock before the Initial Begin statement*--------

```
        always begin
        #20 clk=~clk;
        end
```

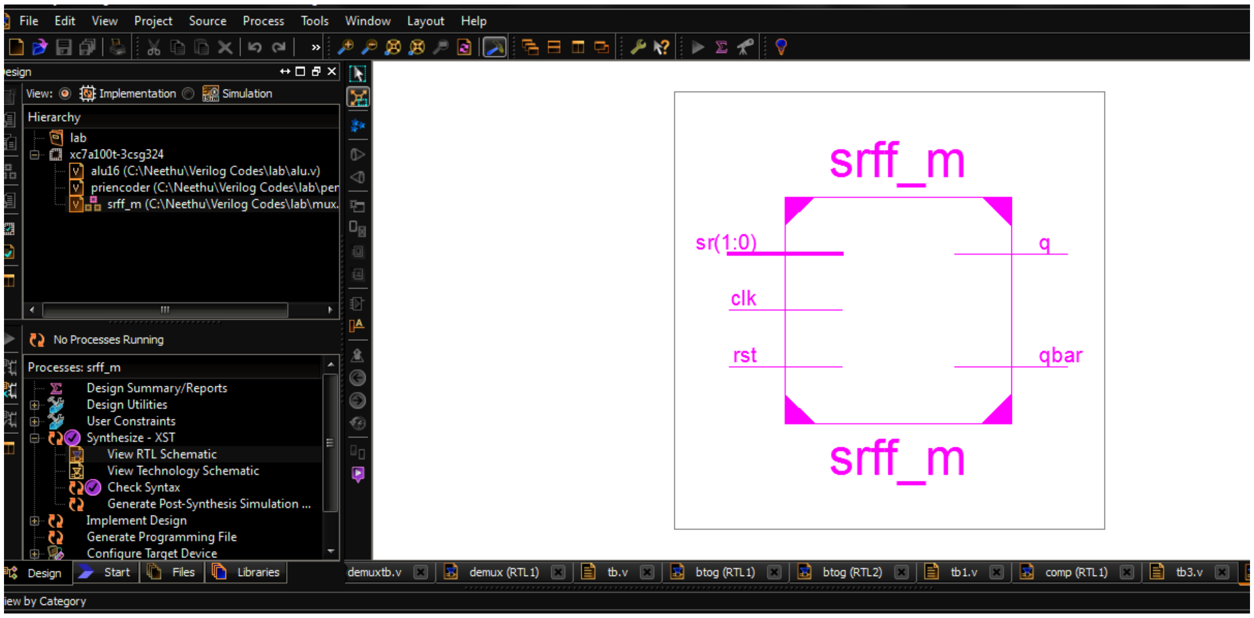
-- *Add stimulus here*--

```
                rst=1;#100; rst = 0;
                sr=2'b00;#100;
                sr=2'b01;#100;
                sr=2'b10;#80;
                sr=2'b11;#100;
```

Simulation Output:


ISim (P.20131013) - [Default.wcfg]
File Edit View Simulation Window Layout Help
Name
Value
q
qbar
sr[1:0]
rst
clk
X
X
11
0
0
0 ns
200 ns
400 ns
600 ns
00
01
10
11

RTL Schematic:


File Edit View Project Source Process Tools Window Layout Help
View: Implementation Simulation
Hierarchy
lab
xc7a100t-3csg324
alu16 (C:\Neethu\Verilog Codes\lab\alu.v)
priencoder (C:\Neethu\Verilog Codes\lab\per
srff_m (C:\Neethu\Verilog Codes\lab\mux.
No Processes Running
Processes: srff_m
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation ...
Implement Design
Generate Programming File
Configure Target Device
Design Start Files Libraries
demuxtb.v demux (RTL1) tb.v btog (RTL1) btog (RTL2) tb1.v comp (RTL1) tb3.v
View by Category
srff_m
sr(1:0)
clk
rst
q
qbar
srff_m

ii) <u>JK FLIP FLOP</u>

a) <u>VHDL program:</u>

```
library ieee;
use ieee.std_logic_1164.all;
entity jkff_e is
port (clk, rst:in std_logic;
jk: in std_logic_vector(1 downto 0);
q,qbar: buffer std_logic);
end;

architecture jkff_a of jkff_e is


        begin
        process(clk, jk, rst)
                begin
                if(rst = '1') then
                q<= '0';
                qbar<= '1';
                elsif(rising_edge(clk)) then
                        case (jk) is
                                when "00" => q <= q; qbar<= qbar;
                                when "01" => q <= '0'; qbar<= '1';
```

```
                                        when "10" => q <= '1'; qbar<= '0';
                                        when "11" => q <= not q; qbar<= not qbar;
                                        when others => null;
                                end case;
                        end if;
                end process;
end jkff_a;
```

Test Bench Code (VHDL):-

```
-------- insert stimulus here--------

rst<='1';wait for 100 ns; rst<='0';
                jk<="00";wait for 100 ns;
                jk<="01";wait for 100 ns;
                jk<="10";wait for 100 ns;
                jk<="11";wait for 100 ns;
```

b) Verilog program:

```
module jkff_m (jk,rst,clk,q,qbar);
input [1:0]jk;
input rst,clk;
output reg q,qbar;

always@ (posedge clk)
begin
        if (rst==1)
        begin q=1'b0; qbar=1'b1; end
        else
        case (jk)
        2'b00 : begin q=q;qbar=qbar; end
        2'b01 : begin q=1'b0; qbar=1'b1; end
        2'b10 : begin q=1'b1; qbar=1'b0; end
        2'b11 : begin q=~ q; qbar=~ qbar; end
        default : begin q=1'b0; qbar=1'b1; end
        endcase
end
endmodule
```

Test Bench Code (Verilog):-

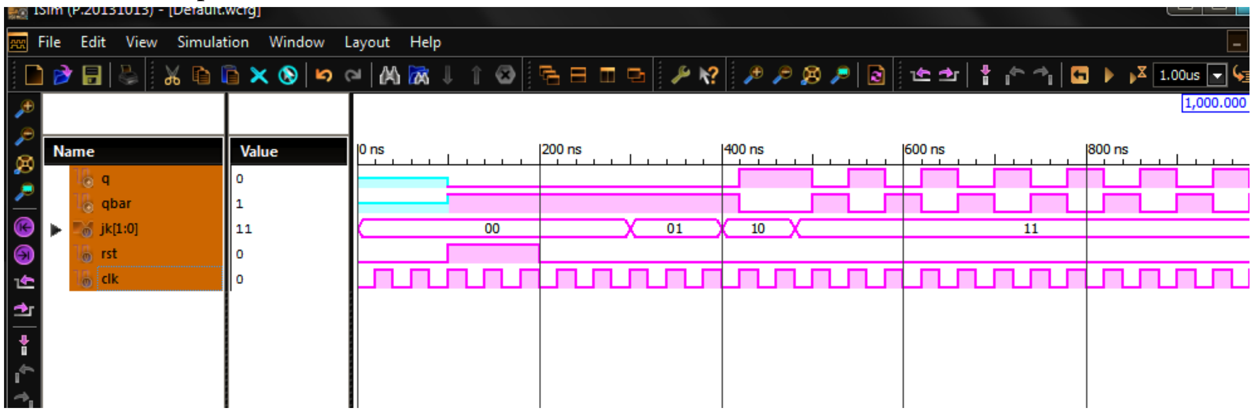
-------*Define the clock before the Initial Begin statement*--------

```
always begin
#20 clk=~clk;
end
```

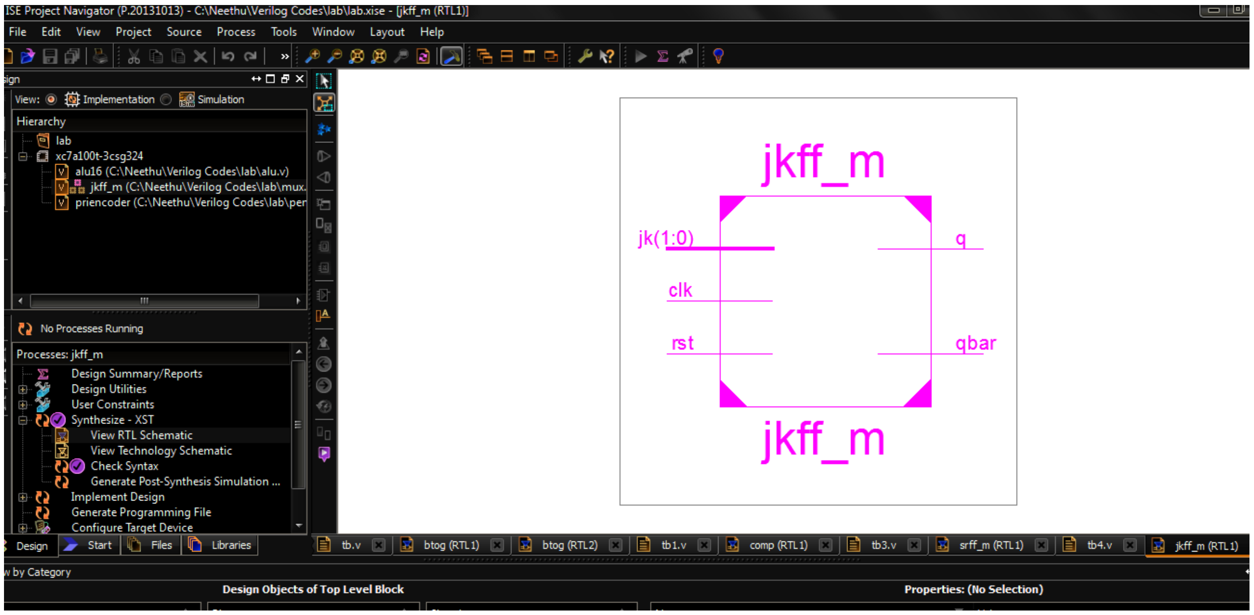
-- *Add stimulus here*--

```
rst=1;#100;rst = 0;
        jk=2'b00;#100;
        jk=2'b01;#100;
        jk=2'b10;#80;
        jk=2'b11;#100;
```

Simulation Output:



RTL Schematic


ISE Project Navigator (P.20131013) - C:\Neethu\Verilog Codes\lab\lab.xise - [jkff_m (RTL1)]
File Edit View Project Source Process Tools Window Layout Help
View: Implementation Simulation
Hierarchy
lab
xc7a100t-3csg324
alu16 (C:\Neethu\Verilog Codes\lab\alu.v)
jkff_m (C:\Neethu\Verilog Codes\lab\mux.
priencoder (C:\Neethu\Verilog Codes\lab\per
No Processes Running
Processes: jkff_m
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation ...
Implement Design
Generate Programming File
Configure Target Device
Design Start Files Libraries
tb.v btog (RTL1) btog (RTL2) tb1.v comp (RTL1) tb3.v srff_m (RTL1) tb4.v jkff_m (RTL1)
by Category
Design Objects of Top Level Block
Properties: (No Selection)
jkff_m
jk(1:0)
clk
rst
q
qbar
jkff_m

iii) <u>D FLIP FLOP</u>

a) VHDL program:

```
library ieee;
use ieee.std_logic_1164.all;
entity dff_e is
port(d,rst,clk:in std_logic;
q,qbar:buffer std_logic);
end dff_e;

architecture dff_a of dff_e is
begin
        process (d,rst,clk)
        begin
                if (rst='1') then
                q<='0'; qbar<='1';
        elsif (rising_edge(clk)) then
        q<=d; qbar<=not d;
        end if;
        end process;
end dff_a;
```

Test Bench Code (VHDL):-

```
------------ insert stimulus here------------

rst<='1';wait for 100 ns;rst<='0';
                        d<='1'; wait for 100 ns;
                        d<='0'; wait for 100 ns;
```

b) Verilog program:

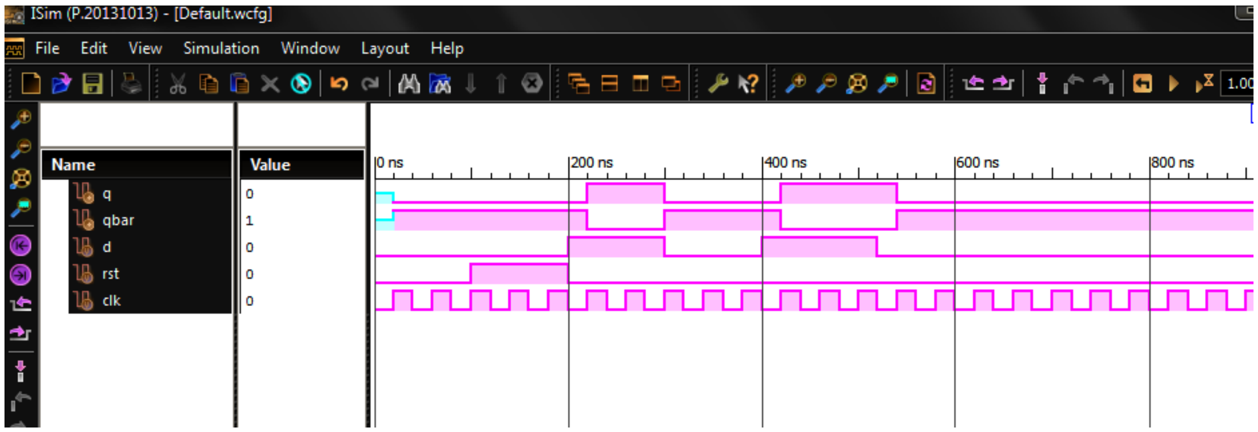
```
module dff_m (d,rst,clk,q,qbar);
input d,rst,clk;
output regq,qbar;
always@ (posedge clk)
begin
if (rst==1)
        begin q=1'b0; qbar=1'b1; end
        else
        begin q=d; qbar=~d; end
```

```
end
endmodule
```

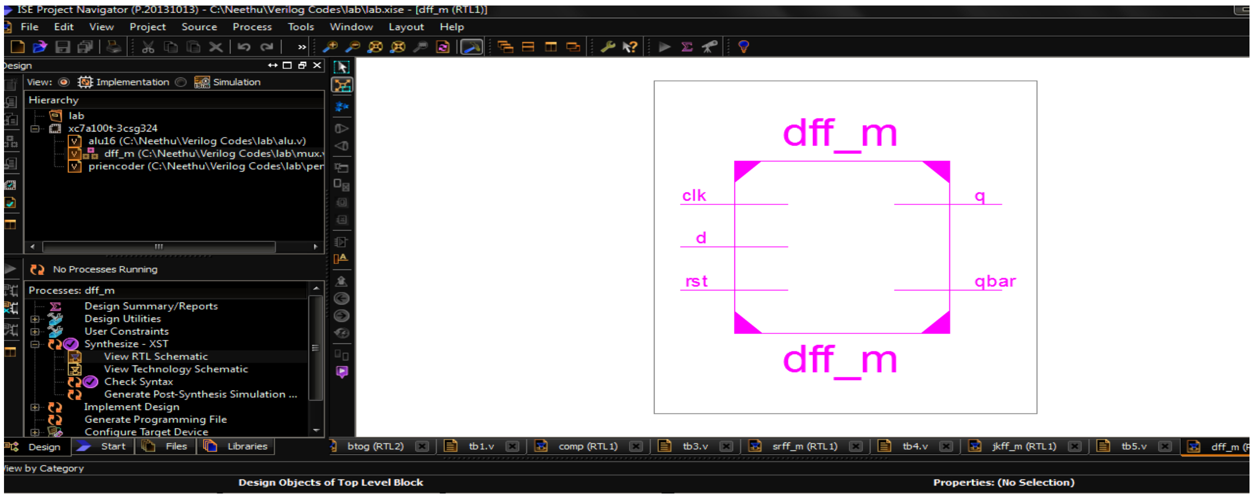
Test Bench Code (Verilog):-

```
-------Define the clock before the Initial Begin statement--------
          always begin
          #20 clk=~clk;
          end
-- Add stimulus here--
                    rst=1;#100;rst=0;
                    d=1; #100;
                    d=0; #100;
                    d=1;#120;
                    d=0;#100;
```

Simulation output:


ISim (P.20131013) - [Default.wcfg]
File Edit View Simulation Window Layout Help
Name
Value
q 0
qbar 1
d 0
rst 0
clk 0
0 ns
200 ns
400 ns
600 ns
800 ns

RTL Schematic:


ISE Project Navigator (P.20131013) - C:\Neethu\Verilog Codes\lab\lab.xise - [dff_m (RTL1)]
File Edit View Project Source Process Tools Window Layout Help
Hierarchy
lab
xc7a100t-3csg324
alu16 (C:\Neethu\Verilog Codes\lab\alu.v)
dff_m (C:\Neethu\Verilog Codes\lab\mux.
priencoder (C:\Neethu\Verilog Codes\lab\per
No Processes Running
Processes: dff_m
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation ...
Implement Design
Generate Programming File
Configure Target Device
dff_m
clk
d
rst
q
qbar
dff_m
Design Objects of Top Level Block
Properties: (No Selection)

iv) T FLIP FLOP

a) <u>VHDL program:</u>

```
library ieee;
use ieee.std_logic_1164.all;
entity tff_e is
port(t,rst,clk:in std_logic;
q,qbar:buffer std_logic);
end tff_e;
architecture tff_a of tff_e is
begin
process (t,rst,clk)
begin
        if (rst='1') then
        q<='0'; qbar<='1';
        elsif (rising_edge(clk)) then
        if (t='1') then q<=not q; qbar<=not qbar;
        end if;
        end if;
end process;
end tff_a;
```

<u>Test Bench Code (VHDL):-</u>

-------- *insert stimulus here*--------

```
        rst<='1';wait for 100 ns;rst<='0';
        t<='0'; wait for 100 ns;
         t<='1'; wait for 100 ns;
```

b) <u>Verilog program:</u>

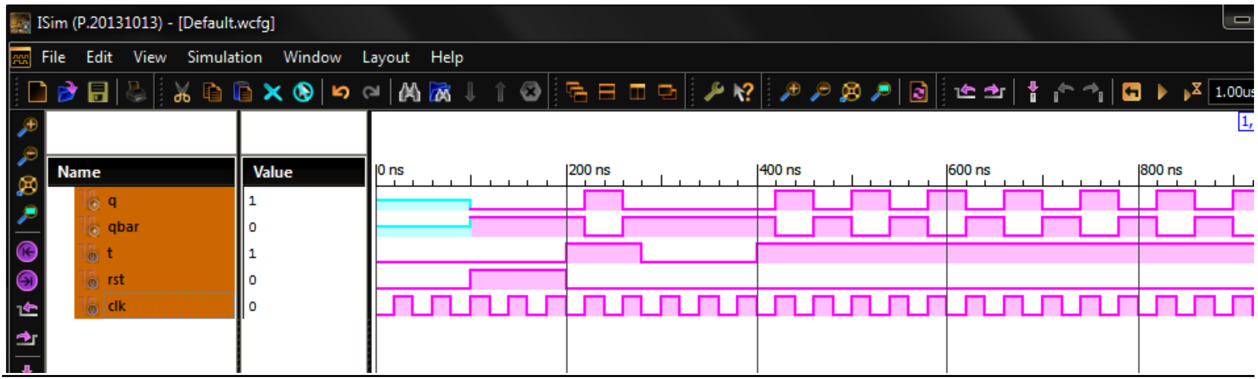
```
module tff_m(t,rst,clk,q,qbar);
input t,rst,clk;
output reg q,qbar;
always@ (posedgeclk)
begin
 if (rst==1)
    begin q=1'b0; qbar=1'b1; end
    else if (t==1)
    begin q=~q; qbar=~qbar; end
```

```
end
endmodule
```

<u>Test Bench Code (Verilog):-</u>

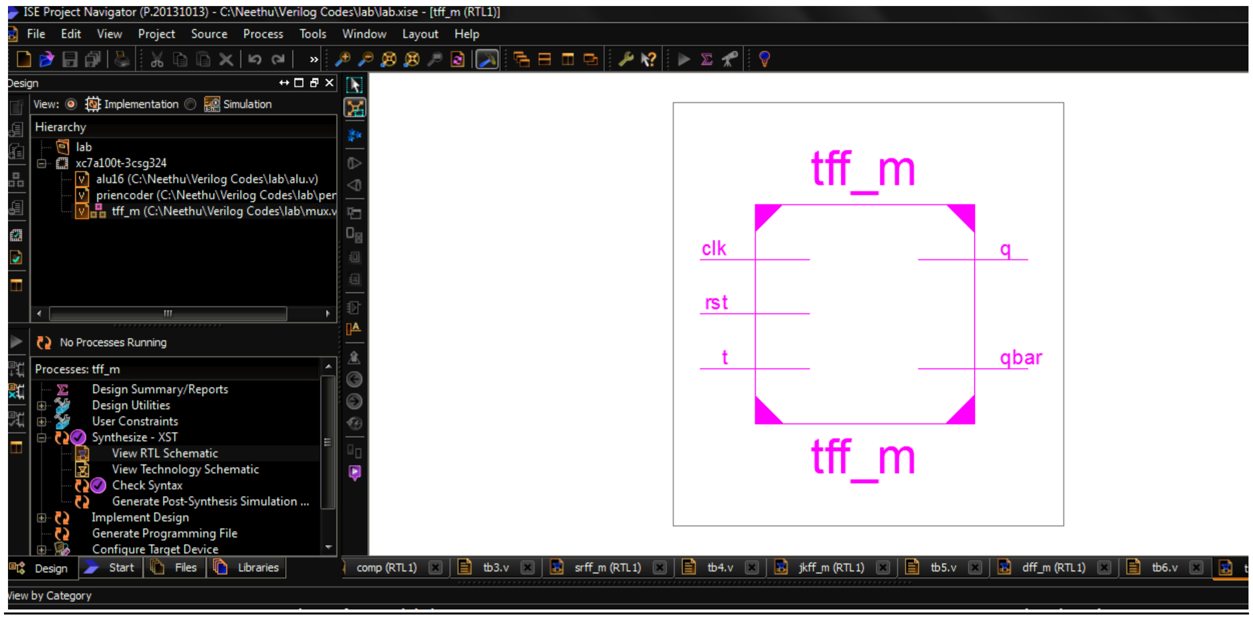
```
-------Define the clock before the Initial Begin statement--------
            always begin
            #20 clk=~clk;
            end
///////// Add stimulus here///////////////
rst=1;#100;rst=0;
 t=1;#80;
 t=0;#120;
t=1;#100;
```

Simulation output:


ISim (P.20131013) - [Default.wcfg]
File Edit View Simulation Window Layout Help
Name
Value
0 ns
200 ns
400 ns
600 ns
800 ns
1.00us

Name	Value
q	1
qbar	0
t	1
rst	0
clk	0

<u>RTL Schematic:</u>


ISE Project Navigator (P.20131013) - C:\Neethu\Verilog Codes\lab\lab.xise - [tff_m (RTL1)]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
lab
xc7a100t-3csg324
alu16 (C:\Neethu\Verilog Codes\lab\alu.v)
priencoder (C:\Neethu\Verilog Codes\lab\per
tff_m (C:\Neethu\Verilog Codes\lab\mux.v
No Processes Running
Processes: tff_m
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation ...
Implement Design
Generate Programming File
Configure Target Device
Design Start Files Libraries
comp (RTL1) tb3.v srff_m (RTL1) tb4.v jkff_m (RTL1) tb5.v dff_m (RTL1) tb6.v
View by Category
tff_m
clk
rst
t
q
qbar
tff_m

Result

The flip flops T, D, SR and JK in behavioural model have been designed using HDL and verified by simulation.

EXPERIMENT NO. 7

Design 4bit Binary and BCD counters (Synchronous reset and Asynchronous reset and "any sequence" counters)

AIM
To write a verilog and VHDL code for 4bit Binary and BCD counters (Synchronous reset and Asynchronous reset and "any sequence" counters)

Software Required: Xilinx ISE 12.1, ISim simulator

Theory:
In digital logic and computing, a counter is a device which stores (and sometimes displays) the number of times a particular event or process has occurred, often in relationship to a clock .Binary counters are counters that go through a binary sequence and an n-bit binary counter is made of "n" number of flip-flops counting from 0 to 2^n-1. BCD counters follow a sequence of ten states and count using BCD numbers from 0000 to 1001 and then returns to 0000 and repeats.

Programs:

i) BINARY COUNTER (SYNCHRONOUS RESET)

a) VHDL program:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity sbc_e is
port(clk,rst:in std_logic;
count:bufferstd_logic_vector(3 downto 0));
end sbc_e;

architecture sbc_a of sbc_e is
begin
        process(rst,clk)
        begin
                if (rising_edge(clk)) then
                        if (rst='1') then
                                count<="0000";
                        else
                                count<=count+1;
```

```
                    end if;
            end if;
    end process;

end sbc_a;
```

Test Bench Code (VHDL):-

```
-------- insert stimulus here--------

rst<='1';wait for 100 ns;rst<='0';
```

b) Verilog program:

```
module sbc_m (clk,rst,count);
input clk,rst;
output reg [3:0] count;
always @(posedge clk)
begin
    if (rst==1)
            count=4'b0000;
    else
            count=count+1;
end
endmodule
```

Test Bench Code (Verilog):-

```
-------Define the clock before the Initial Begin statement--------
    always begin
    #20 clk=~clk;
    end

-- Add stimulus here--

  rst = 1; #100; rst=0;
```

ii) BINARY COUNTER (ASYNCHRONOUS RESET)

a) VHDL program:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity abc_e is
port(clk,rst:in std_logic;
count:bufferstd_logic_vector(3 downto 0));
end abc_e;
architecture abc_a of abc_e is
begin
        process(rst,clk)
        begin
                if (rst='1') then
                        count<="0000";
                elsif (rising_edge(clk)) then

                        count<=count+1;
          end if;
        end process;
end abc_a;
```

<u>*Test Bench Code (VHDL):-*</u>

```
-------- insert stimulus here--------

rst<='1';wait for 100 ns;
                rst<='0';wait for 63 ns;
                rst<='1';wait for 53 ns;
                rst<='0';wait for 25 ns;
```

b) <u>Verilog program:</u>

```
module abc_m(clk,rst,count);
input clk,rst;
output reg [3:0] count;
always @(posedge rst, posedge clk)
begin
        if (rst)
                count=4'b0000;
        else if (clk)
                count=count+1;
```

```
end
endmodule
```

Test Bench Code (Verilog):-

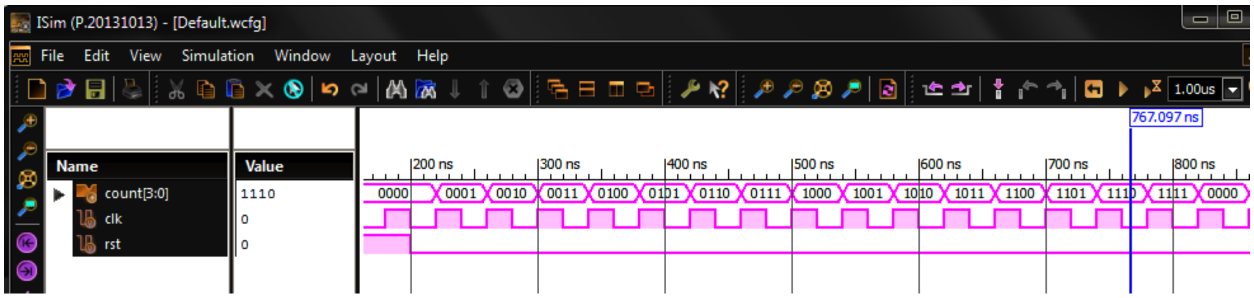
-------*Define the clock before the Initial Begin statement*--------

```
        always begin
        #20 clk=~clk;
        end
```

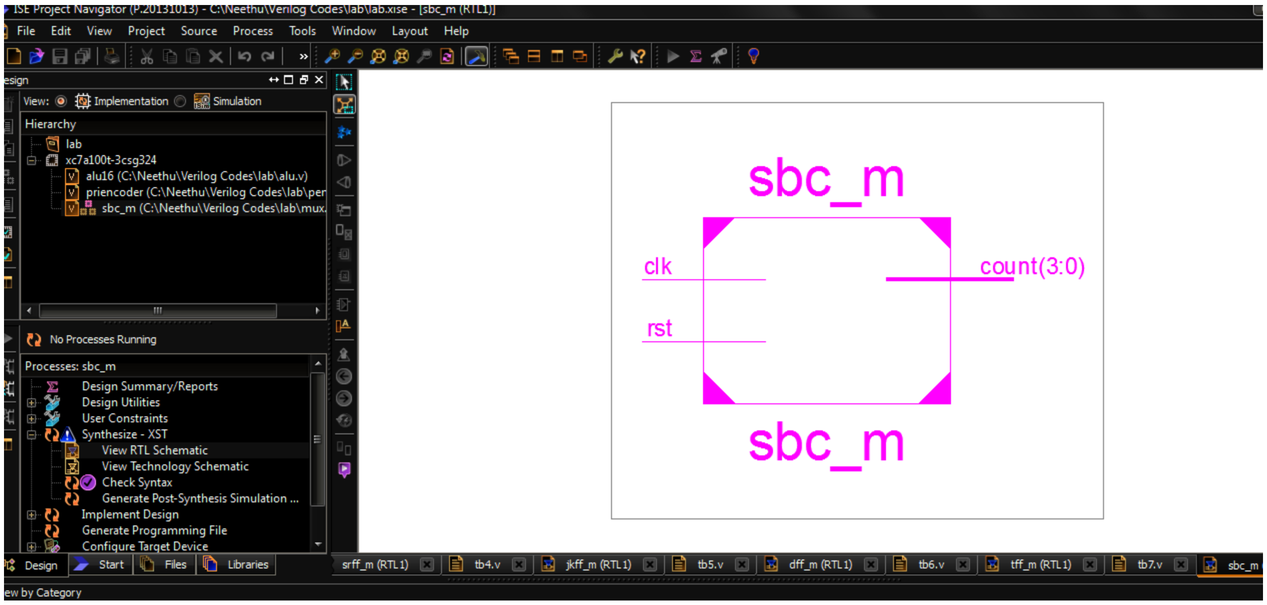
-- *Add stimulus here*--

```
rst = 1;#100;rst=0;
```

Simulation Output (Binary counter with syn. reset):



RTL Schematic:


ISE Project Navigator (P.20131013) - C:\Neethu\Verilog Codes\lab\lab.xise - [sbc_m (RTL1)]
File Edit View Project Source Process Tools Window Layout Help
View: Implementation Simulation
Hierarchy
lab
xc7a100t-3csg324
alu16 (C:\Neethu\Verilog Codes\lab\alu.v)
priencoder (C:\Neethu\Verilog Codes\lab\per
sbc_m (C:\Neethu\Verilog Codes\lab\mux.
No Processes Running
Processes: sbc_m
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation ...
Implement Design
Generate Programming File
Configure Target Device
Design Start Files Libraries
srff_m (RTL1) tb4.v jkff_m (RTL1) tb5.v dff_m (RTL1) tb6.v tff_m (RTL1) tb7.v sbc_m
sbc_m
clk
rst
count(3:0)
sbc_m

iii) BCD COUNTER (SYNCHRONOUS RESET)

a) VHDL program:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity sbcdc_e is
port(clk,rst:in std_logic;
count:bufferstd_logic_vector(3 downto 0));
end sbcdc_e;
architecture sbcdc_a of sbcdc_e is
begin
        process(rst,clk)

        begin
                if (rising_edge(clk)) then

                        if (rst='1' or count="1001") then
                                count<="0000";
                        else
                                count<=count+1;
                        end if;
                end if;
        end process;
end sbcdc_a;
```

Test Bench Code (VHDL):-

-------- *insert stimulus here*--------

```
rst<='1';wait for 100 ns;rst<='0';
```

b) Verilog program:

```
module sbcdc_m(clk,rst,count);
input clk,rst;
output reg [3:0] count;
always @(posedge clk)
```

```
begin
        if (rst==1 | count==4'b1001)

                count=4'b0000;
        else
                count=count+1;
end
endmodule
```

Test Bench Code (Verilog):-

-------*Define the clock before the Initial Begin statement*--------

```
        always begin
        #20 clk=~clk;
        end
rst=1;#100;rst=0;
```

iii) BCD COUNTER (ASYNCHRONOUS RESET)

a) VHDL program:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity asbcdc_e is
port(clk,rst:in std_logic;
count:bufferstd_logic_vector(3 downto 0));
end asbcdc_e;
architecture asbcdc_a of asbcdc_e is
begin
        process(rst,clk)
        begin
                if (rst='1') then
                        count<="0000";
                elsif (rising_edge(clk)) then
                        if (count="1001") then
                        count<="0000";
                        else
                        count<=count+1;
                        end if;
                end if;
        end process;
end asbcdc_a;
```

<u>Test Bench Code (VHDL):-</u>

-------- *insert stimulus here*--------

```
rst<='1';wait for 100 ns;rst<='0';
```

b) <u>Verilog program:</u>

```
module abcdc_m(clk,rst,count);
input clk,rst;
output reg [3:0] count;
always @(posedge rst, posedge clk)

begin
        if (rst)
        count=4'b0000;
        else if (clk) begin
                if (count==4'b1001)
                count=4'b0000;
                 else
count=count+1;
        end
end
endmodule
```

<u>Test Bench Code (Verilog):-</u>

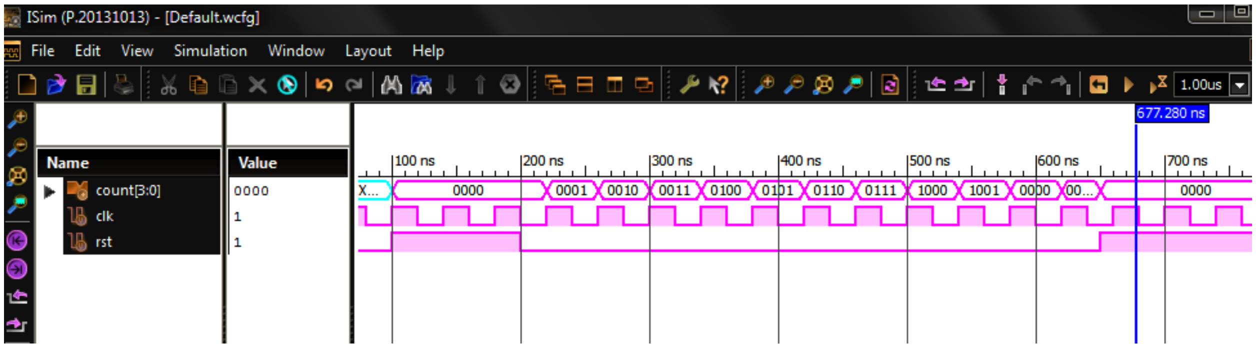
-------*Define the clock before the Initial Begin statement*--------

```
        always begin
        #20 clk=~clk;
        end
```

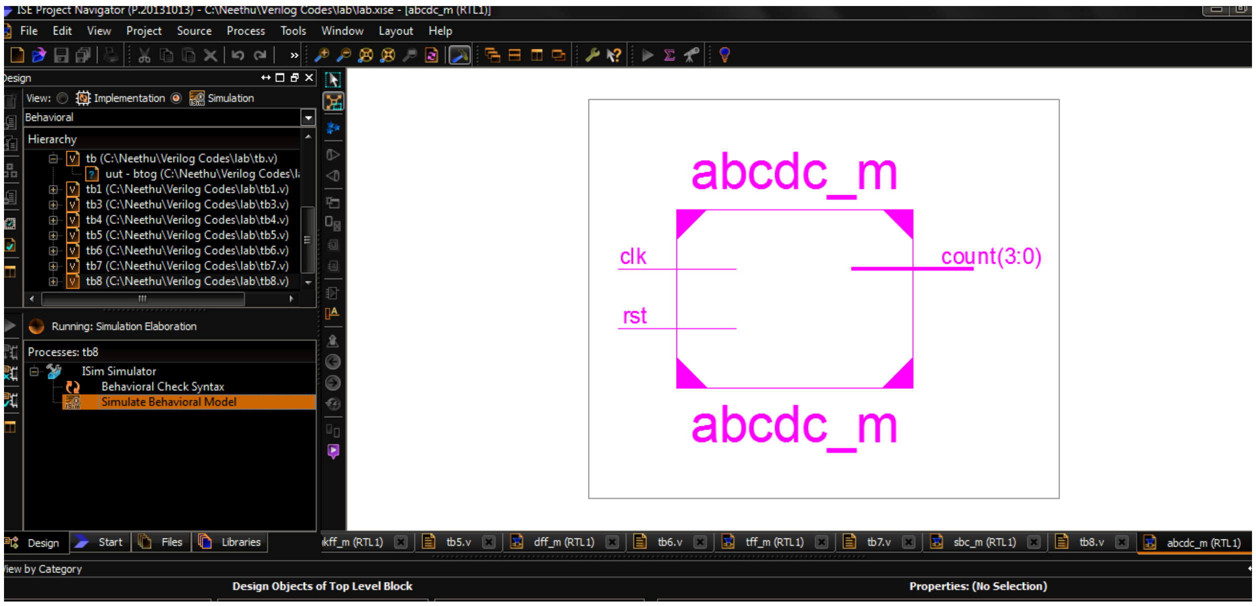
///////// Add stimulus here///////////////

```
          rst=1;#100;rst=0;
```

Simulation Output (Asynchronous reset BCD counter):



RTL Schematic:


ISE Project Navigator (P.20131013) - C:\Neethu\Verilog Codes\lab\lab.xise - [abcdc_m (RTL1)]
File Edit View Project Source Process Tools Window Layout Help
View: Implementation Simulation
Behavioral
Hierarchy
tb (C:\Neethu\Verilog Codes\lab\tb.v)
uut - btog (C:\Neethu\Verilog Codes\l
tb1 (C:\Neethu\Verilog Codes\lab\tb1.v)
tb3 (C:\Neethu\Verilog Codes\lab\tb3.v)
tb4 (C:\Neethu\Verilog Codes\lab\tb4.v)
tb5 (C:\Neethu\Verilog Codes\lab\tb5.v)
tb6 (C:\Neethu\Verilog Codes\lab\tb6.v)
tb7 (C:\Neethu\Verilog Codes\lab\tb7.v)
tb8 (C:\Neethu\Verilog Codes\lab\tb8.v)
Running: Simulation Elaboration
Processes: tb8
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
abcdc_m
clk
rst
count(3:0)
abcdc_m
Design
Start
Files
Libraries
dff_m (RTL1)
tb5.v
dff_m (RTL1)
tb6.v
tff_m (RTL1)
tb7.v
sbc_m (RTL1)
tb8.v
abcdc_m (RTL1)
View by Category
Design Objects of Top Level Block
Properties: (No Selection)

iv) <u>“ANY SEQUENCE” COUNTER (synchronous reset) 0,8,5,13,7</u>

a) <u>VHDL program:</u>

```
library ieee;
use ieee.std_logic_1164.all;
entity asc_e is
port(clk,rst:in std_logic;
count:buffer std_logic_vector(3 downto 0));
end asc_e;
architecture asc_a of asc_e is
begin
        process(clk,rst)
        begin
                if (rising_edge(clk)) then
                        if (rst='1') then count<="0000";
                        else
                        case count is
                        when "0000" => count <= "1000";
                        when "1000" => count <= "0101";
                        when "0101" => count <= "1101";
                        when "1101" => count <= "0111";
                        when "0111" => count<= "0000";
                        when others => null;
                        end case;
                        end if;
                end if;
        end process;
end asc_a;
```

__Test Bench Code (VHDL):-__

```
-------- insert stimulus here--------
rst<='1';wait for 100 ns;rst<='0';
```

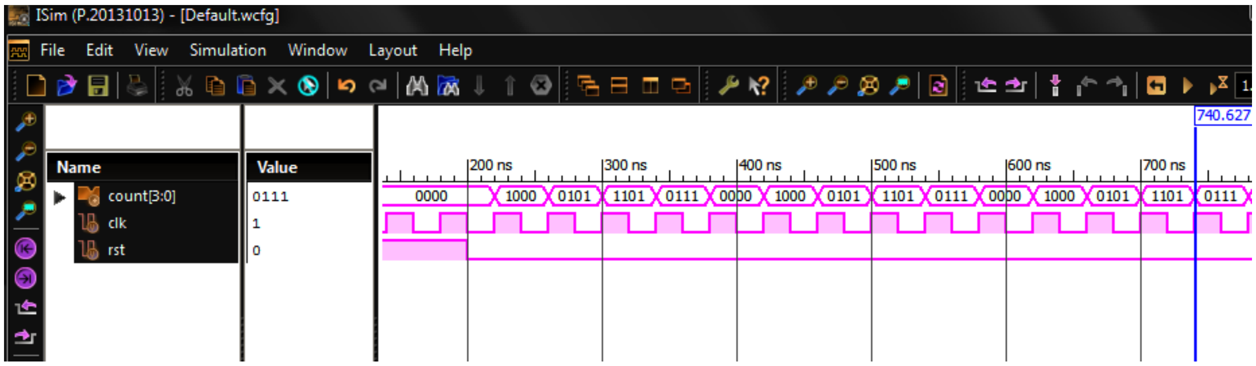
b) __Verilog program:__

```
module asc_m(clk,rst,count);
input clk,rst;
output reg [3:0] count;
always @(posedge clk)
begin
        if(rst) count= 4'b0000;
        else
         case (count)
                4'b0000 : count = 4'b1000;
                4'b1000 : count = 4'b0101;
                4'b0101 : count = 4'b1101;
                4'b1101 : count = 4'b0111;
                4'b0111 : count = 4'b0000;
endcase
end
endmodule
```

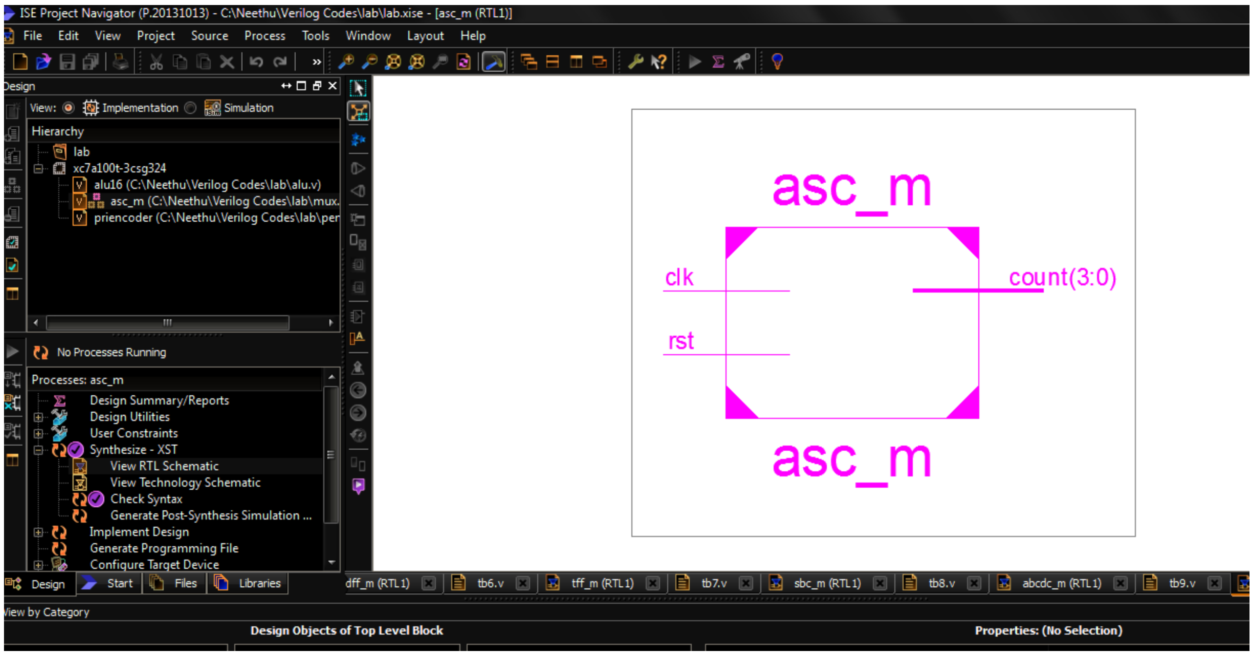
Test Bench Code (Verilog):-

```
-------Define the clock before the Initial Begin statement--------
        always begin
        #20 clk=~clk; end

         //enter stimulus

         rst=1;#100; rst=0;
```

Simulation Output:



RTL Schematic:


ISE Project Navigator (P.20131013) - C:\Neethu\Verilog Codes\lab\lab.xise - [asc_m (RTL1)]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
lab
xc7a100t-3csg324
alu16 (C:\Neethu\Verilog Codes\lab\alu.v)
asc_m (C:\Neethu\Verilog Codes\lab\mux.
priencoder (C:\Neethu\Verilog Codes\lab\per
No Processes Running
Processes: asc_m
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation ...
Implement Design
Generate Programming File
Configure Target Device
Design Start Files Libraries
dff_m (RTL1) tb6.v tff_m (RTL1) tb7.v sbc_m (RTL1) tb8.v abcdc_m (RTL1) tb9.v
View by Category
Design Objects of Top Level Block
Properties: (No Selection)
asc_m
clk
rst
count(3:0)
asc_m

Result

The HDL code for 4bit Binary and BCD counters (Synchronous reset and Asynchronous reset and “any sequence” counters) have been written and verified the output.

CYCLE II

INTERFACING EXPERIMENTS USING XILINX SPRTAN-6

Device: Xilinx spartan 6.
Device family: XC6SLX9
Simulator: ISE simulator.
Package :144 pin package

EXPERIMENT NO. :8

Write an HDL code to display messages on the given seven segment display

AIM

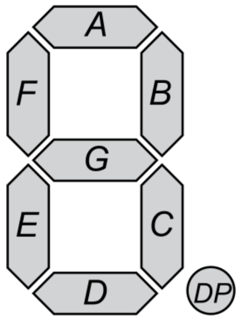
To Write an HDL code to display messages on the given seven segment display by interfacing using FPGA.

Equipments Required:

Xilinx ISE 12.1 , JTAG , FPGA Kit etc

Theory

A seven-segment display is a form of electronic display device for displaying decimal numerals that is an alternative to the more complex dot matrix displays. Seven-segment displays are widely used in digital clocks, electronic meters, basic calculators, and other electronic devices that display numerical information. The individual segments of a seven-segment display are shown below:



Verilog Code:

```
Module hexkeypad(
Input clk,
input [3:0] read,
output reg [3:0] scan,
output [3:0] dispn,
output reg [6:0] disp);
reg [1:0] count=2'b00;

always@(posedge clk)
begin
count = count + 1;
```

```
end

always@(count)
begin
case(count)
2'b00: scan= 4'b0001;
2'b01: scan= 4'b0010;
2'b10: scan= 4'b0100;
2'b11: scan= 4'b1000;
default:scan=4'b0001;
endcase
end

assign dispn = 4'b1110; // use the last display

always@(scan, read)
begin
case(scan)
4'b0001:        case(read)
                        4'b0001: disp= 7'b1111110;//0
                        4'b0010: disp= 7'b0110011;//4
                        4'b0100: disp= 7'b1111111;//8
                        4'b1000: disp= 7'b1001110;//C
                        default: disp= 7'b0000000;
                        endcase
4'b0010:        case(read)
                        4'b0001: disp= 7'b0110000;//1
                        4'b0010: disp= 7'b1011011;//5
                        4'b0100: disp= 7'b1111011;//9
                        4'b1000: disp= 7'b0111101;//D
                        default: disp= 7'b0000000;
                        endcase
4'b0100:        case(read)
                        4'b0001: disp= 7'b1101101;//2
                        4'b0010: disp= 7'b1011111;//6
                        4'b0100: disp= 7'b1110111;//A
                        4'b1000: disp= 7'b1001111;//E
                        default: disp= 7'b0000000;
                        endcase
4'b1000:        case(read)
                        4'b0001: disp= 7'b1111001;//3
```

```
                4'b0010: disp= 7'b1110000;//7
                4'b0100: disp= 7'b0011111;//B
                4'b1000: disp= 7'b1000111;//F
                default: disp= 7'b0000000;
                endcase
default: disp =7'b0000000;

endcase
end

endmodule
```

<u>User constraint file for seven segment display</u>

```
net " disp<0>" loc = "p22";
 net " disp<1>" loc = "p21";
net " disp<2>" loc = "p17";
net " disp<3>" loc = "p15";
 net " disp<4>" loc = "p14";
net " disp<5>" loc = "p12";
net " disp<6>" loc = "p1";
net " dispn<0>" loc = "p30";
 net " dispn<1>" loc = "p29";
 net " dispn<2>" loc = "p27";
net " dispn<3>" loc = "p26";
net " read<0>" loc = "p121";
net " read<1>" loc = "p124";
 net " read<2>" loc = "p127";
net " read<3>" loc = "p126";
net " scan<0>" loc = "p132";
net " scan<1>" loc = "p140";
net " scan<2>" loc = "p139";
net " scan<3>" loc = "p143";
 net " clk" loc = "p55";
```

Result:

The verilog code to display messages on the given seven segment display has been written and verified by interfacing using Spartan 6 FPGA.

EXPERIMENT NO. :9

Write the HDL code to control speed, direction of dc and stepper motor.

Aim
To Write the HDL code to control speed, direction of dc and stepper motor and verify the same by interfacing with these motors using Spartan 6 FPGA.

Equipments Required:

Xilinx ISE 12.1 , JTAG , FPGA Kit etc

Theory
DC Motor is a two wire continuous rotation motor and the two wires are power and ground. When the supply is applied, a DC motor will start rotating until that power is detached. Most of the DC motors run at high revolutions per minute (RPM), examples are; fans being used in computers for cooling or car wheels controlled by a radio. The DC motor speed can be controlled by using PWM (pulse width modulation) technique, a technique of fast pulsing the power ON & OFF. The percentage of time consumed cycling the ON/OFF ratio defines the motor's speed.

A stepper motor is fundamentally a servo motor that uses a different method of motorization. stepper motors utilizes multiple notched electromagnets arranged around a central equipment to describe the position.Stepper motor needs an exterior control circuit to separately energize each electromagnet and make the motor shaft ON. When the electromagnet is power-driven it attracts the equipment's teeth and supports them, somewhat offset from the next electromagnet 'B'. When 'A' is turned off, and 'B' turned on, the apparatus rotates slightly to align with 'B', and everywhere the circle, with each electromagnet around the apparatus energizing and de-energizing in turn to make a rotation. Each revolution from one electromagnet to the next is named a "step", and therefore the motor can be activated by exact pre-defined step angles through a full 360^o rotation. Stepper motors are slow, easy setup, precise rotation, and control – Advantage over other motors like servo motors in controlling of a position. Suitable for 3D printers and related devices where the position is essential.

Verilog Program

i) DC MOTOR'S SPEED AND DIRECTION CONTROL

```
module dcmotor_v(clk,rst,dir,row,en,mtr1,mtr2);
input clk,rst,dir;
input [3:0]row;
output reg mtr1,mtr2;
output en;
```

```
reg [25:0]div;
wire clkd,tick;
reg [7:0]counter;
reg [7:0]dutycycle;

always@(posedge clk)
begin
div=div+1;
end
assign clkd = div[8];
assign tick = row[0] & row[1] & row[2] & row[3];
assign en =1'b1;
always@(posedge clkd)
begin
counter= counter+1;

end
always@(negedge tick)
begin
case(row)
4'b1110:dutycycle<=250;
4'b1101:dutycycle<=190;
4'b1011:dutycycle<=100;
4'b0111:dutycycle<=40;
default:dutycycle<=30;
endcase
end
always@(rst,dir)
begin
if (rst==0)

        begin
        mtr1<=0;
        mtr2<=0;
        end
else
        if (dir==0)
        begin
        mtr2<=0;
        if (counter<=dutycycle)
        mtr1<=1;
        else
        mtr1<=0;
        end
        else// if direction is 1

        begin
        mtr1<=0;
        if (counter<=dutycycle)
        mtr2<=1;
        else
        mtr2<=0;
        end
       end
endmodule
```

ii) STEPPER MOTOR

```
Module stepper_motor(
Input clk,
input reset,
input dir,
input [1:0] row,
output reg [3:0] dout
   );
reg [25:0]divclk;
regi ntclk;
reg [3:0]register;

always@(posedge clk)
begin
divclk= divclk+1;
end

always@(row, divclk)
begin
case(row)
2'b00:intclk= divclk[21];
2'b01:intclk= divclk[19];
2'b10:intclk= divclk[17];
2'b11:intclk= divclk[15];
default:intclk= divclk[21];
```

```
endcase
end

always@(posedge intclk, dir)
begin
if(!reset)
     register=4'b1001;
else
begin
            if(!dir)
                              register= {register[0], register[3:1]};
            else
                             register= {register[2:0],register[3]};
     end

            end
endmodule
```

<u>User constraint file for DC motor</u>

```
net”clk” loc =”p55”;
net “ en” loc = “p5”;
net “ mtr1” loc = “p6”;
net “ mtr2” loc = “p38”;
net “ row<0>” loc = “p57”;
net “ row<1>” loc = “p59”;
net “ row<2>” loc = “p62”;
net “ row<3>” loc = “p67”;
net “ dir” loc = “p81”;
net “ rst<2>” loc = “p80”;
```

<u>User constraint file for stepper motor</u>

```
     net “ row<0>” loc = “p82”;
     net “ row<1>” loc = “p84”;
     net “ dir” loc = “p81”;
     net “ dout<0>” loc = “p38”;
     net “ dout<1>” loc = “p2”;
     net “ dout<2>” loc = “p5”;
     net “ dout<3>” loc = “p6”;
     net “ rst” loc = “p80”;
```

```
net “ clk” loc = “p55”;
```

Result

The HDL code to control speed, direction of dc and stepper motor has been written and verified by interfacing the motors using Spartan 6 FPGA.

EXPERIMENT NO. 10

Write the HDL code to generate different waveforms (sawtooth, sine wave, square, triangle etc) using DAC and FPGA kit**.**

AIM

To Write the HDL code to generate different waveforms (sawtooth, sine wave, square, triangle etc) using DAC and FPGA kit.

Equipments Required:

Xilinx ISE 12.1, JTAG, FPGA, DAC, CRO etc.

Theory

A digital-to-analog converter (DAC or D/A) is an electronic circuit that converts digital data (0's & 1's) to an analog signal. DACs are commonly used for analog waveform generation applications such as audio/music players, video players, TVs, and various electronic systems.

Verilog Code

i) SQUARE WAVE GENERATION

```
module squarewave(clk,rst,dac);
input clk,rst;
output reg[7:0]dac;
reg[7:0]cntr;
reg mode= 1'b0;

always @ (posedge clk,posedge rst)
begin
if (rst==0)
cntr=0;
else
begin
if(mode==0)
begin
cntr=cntr+1'b1;
dac=8'b 11111111;
if(cntr==8'b 11001000)
begin
mode=1;
cntr=0;
```

```
end
end
else
begin
cntr=cntr+1'b1;
dac=8'b00000000;
if(cntr==8'b11001000)
begin
mode=0;
cntr=0;
end
end
end
end
endmodule
```

ii) SAWTOOTH WAVE GENERATION

```
Module sawtooth(clk.rst,dac);
input clk,rst;
output[0:7]dac;
reg[3:0]temp=4’b0000;
reg[0:7]cntr;
always@(posedge clk)
begin
temp=temp+1’ b1;
end
always@(posedge temp[1])

begin
if(rst==0)
cntr=8’b00000000;
else
cntr=cntr+1’b1;
end
assign dac=cntr;
endmodule
```

iii) TRIANGULAR WAVE GENERATION

```
module triangular(clk,rst,dac);
input clk,rst;
output reg[0:7]dac;
reg[3:0]temp=4'b0000;
reg[0:8] cntr;
always @ (posedge clk)
begin
temp=temp+1'b1;
end
always @ (posedge temp[3])
begin
if(rst==0) cntr=9’b000000000;
else
begin
cntr=cntr+1'b1;
if(cntr[0]==1)
dac <= cntr[1:8]
else
dac <= ~(cntr[1:8]);
end
end
endmodule
```

iv) SINE WAVE GENERATION

```
module sinewave(clk,rst,dac);
input clk,rst;
output reg[0:7]dac;
reg[7:0]ce;
integer  i;
reg[7:0] sine[0:36];
initial
begin
i=0;
sine[0]=127;
sine[1]=149;
sine[2]=170;
sine[3]=190;
sine[4]=209;
```

```
sine[5]=224;
sine[6]=237;
sine[7]=246;
sine[8]=252;
sine[9]=254;
sine[10]=252;
sine[11]=246;
sine[12]=237;
sine[13]=224;
sine[14]=209;
sine[15]=190;
sine[16]=170;
sine[17]=149;
sine[18]=127;
sine[19]=104;
sine[20]=83;
sine[21]=63;
sine[22]=45;
sine[23]=29;
sine[24]=17;
sine[25]=7;
sine[26]=2;
sine[27]=0;
sine[28]=2;
sine[29]=7;
sine[30]=17;
sine[31]=29;
sine[32]=45;
sine[33]=63;
sine[34]=83;
sine[35]=104;
sine[36]=127;
end
always @(posedge clk, posedge rst)
begin
if(rst)
ce<=8'b0;
else
ce<=ce+1'b1;
end
always@(posedge ce[37])
```

```
begin
dac<=sine[i];
if(i == 8'd36 )
i<=8'b0;
else
i<=i+1;
end
endmodule
```

<u>User constraint file for waveform generation</u>

```
 net “ dac<0>” loc = “p24”;
net “ dac<1>” loc = “p22”;
net “ dac<2>” loc = “p21”;
net “ dac<3>” loc = “p17”;
net “ dac<4>” loc = “p15”;
net “ dac<5>” loc = “p14”;
net “ dac<6>” loc = “p12”;
net “ dac<7>” loc = “p1”;
net “ rst” loc = “p80”;
net “ clk” loc = “p55”;
```

Result

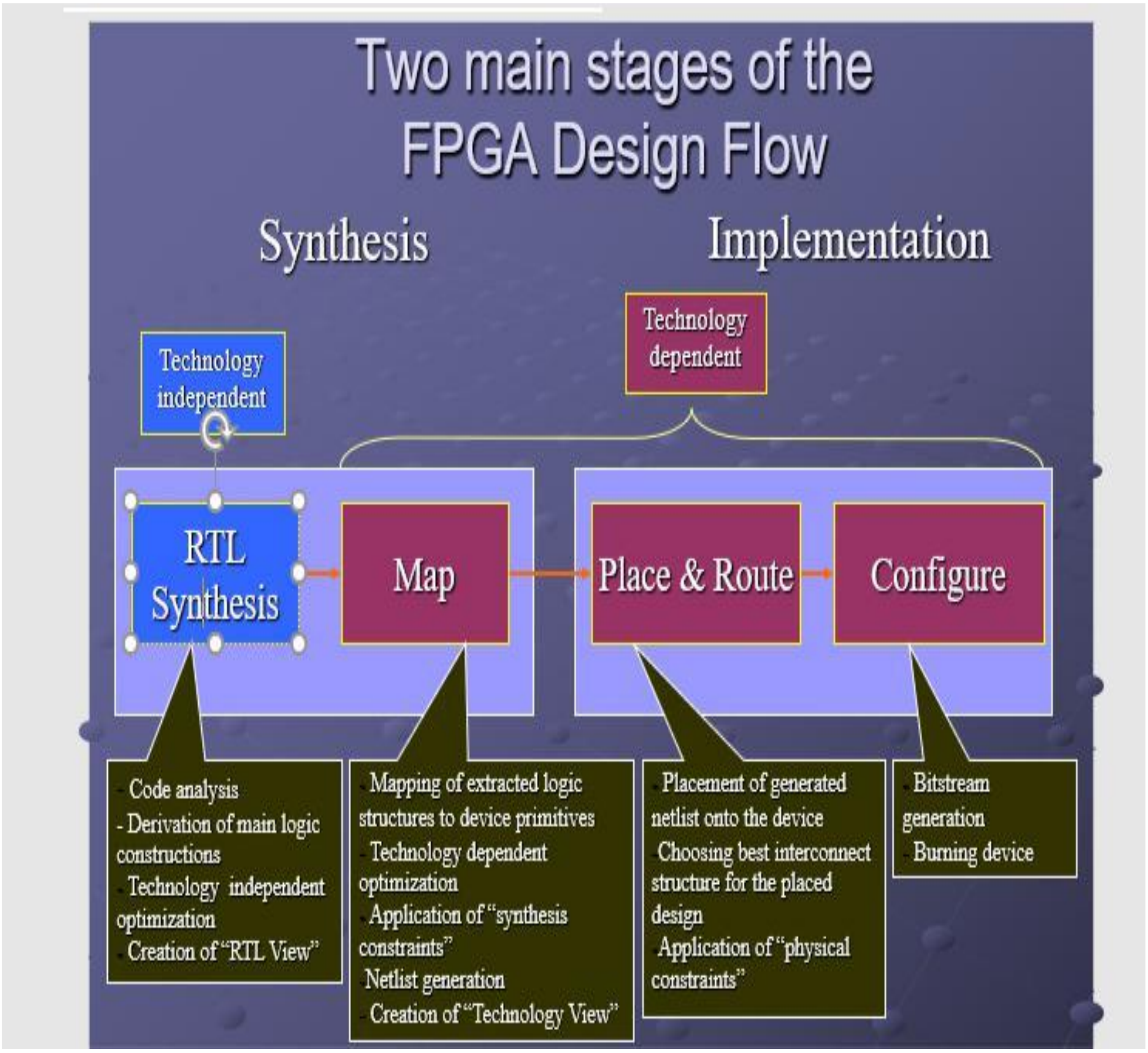
The HDL code to generate different waveforms like sawtooth, sine wave, square, and triangle has been written and verified using DAC and FPGA kit**.**

Frequently Asked VIVA Questions

1. Expand VHDL. What is the difference between VHDL and Verilog?
2. What is the difference between (i) signal and variable (ii) generic & Parameter (iii) function & procedure (iv) task & function (v) always & initial (vi) register & variable (vii) signal & wire
3. What are the different styles of models in VHDL and Verilog?
4. What are the operators in VHDL & Verilog?
5. Which is an operator is having most priority?
6. What is meant by sensitivity list?
7. Give the Following syntax in HDL (i) if, for, function, procedure (ii) while, case.
8. What is the operating frequencyof your FPGA?
9. Expand FPGA & ASIC
10. What are the data types in VHDL?
11. What are the data types in Verilog?What is delta time?
12. What is the difference between (0 to 3) & (3 downto 0)?
13. What is meant by synthesis?
14. Write the flow chart for synthesis process?
15. What is the difference between combination circuit & sequential circuit?
16. What is the difference between latch & Flip flop? Write a Verilog code to swap contents of two registers with and without a temporary register?
17. In a pure combinational circuit is it necessary to mention all the inputs in sensitivity list? If yes, why?What is the difference between wire and reg?
18. Build a 4:1 mux using only 2:1 mux?What are shift operators in HDL?
19. What are the logical operators in VHDL & Verilog?
20. What is the gate density of your FPGA?

21. What is data flow model, structural model, behavioral model?
22. What is the difference between synchronous reset & Asynchronous reset?
23. What is the difference between stepper motor & DC motor?
24. What is the step size of stepper motor?
25. What is mux and demux?
26. What is encoder and decoder?
27. What is the difference between encoder & priority encoder?
28. What is the difference between "bit" and "std_logic"?
29. What are the applications of dc motor and stepper motor?
30. Write the syntax for casex and casez.

APPENDIX


Two main stages of the FPGA Design Flow
Synthesis
Implementation
Technology independent
Technology dependent
RTL Synthesis
Map
Place & Route
Configure
- Code analysis
- Derivation of main logic constructions
- Technology independent optimization
- Creation of "RTL View"
- Mapping of extracted logic structures to device primitives
- Technology dependent optimization
- Application of "synthesis constraints"
- Netlist generation
- Creation of "Technology View"
- Placement of generated netlist onto the device
- Choosing best interconnect structure for the placed design
- Application of "physical constraints"
- Bitstream generation
- Burning device