VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Digital Logic Design
CSE1003 LAB

Digital Simulation Using Verilog in Modelsim

Experiment 9

BIMAL PARAJULI
(20BDS0405)

7/18/21

CSE – 1003 (LAB)

Bimal parajuli (20BDS0405)

DIGITAL LOGIC DESIGN (CSE1003)

Exp#9 – Digital Simulation using ModelSim

Objectives:

- To write the Verilog HDL code for various digital circuits.
- Simulate them using the ModelSim Altera Software
- Test them using appropriate test benches.

Softwares Used:

- ModelSim Altera 10.1d (Quartus II 13.0sp1)

Theory:

Verilog is a hardware description language (HDL) used to model electronic systems. It is most commonly used in the design and verification of digital circuits at the register-transfer level of abstraction. It is also used in the design and verification of analog, digital and mixed-signal circuits.

ModelSim is a Software environment developed by Mentor Graphics for the simulation of hardware description languages such as VHDL, Verilog, etc.

In this experiment, we use ModelSim Software Environment to design, model and test various digital Circuits.

Bimal parajuli (20BDS0405)

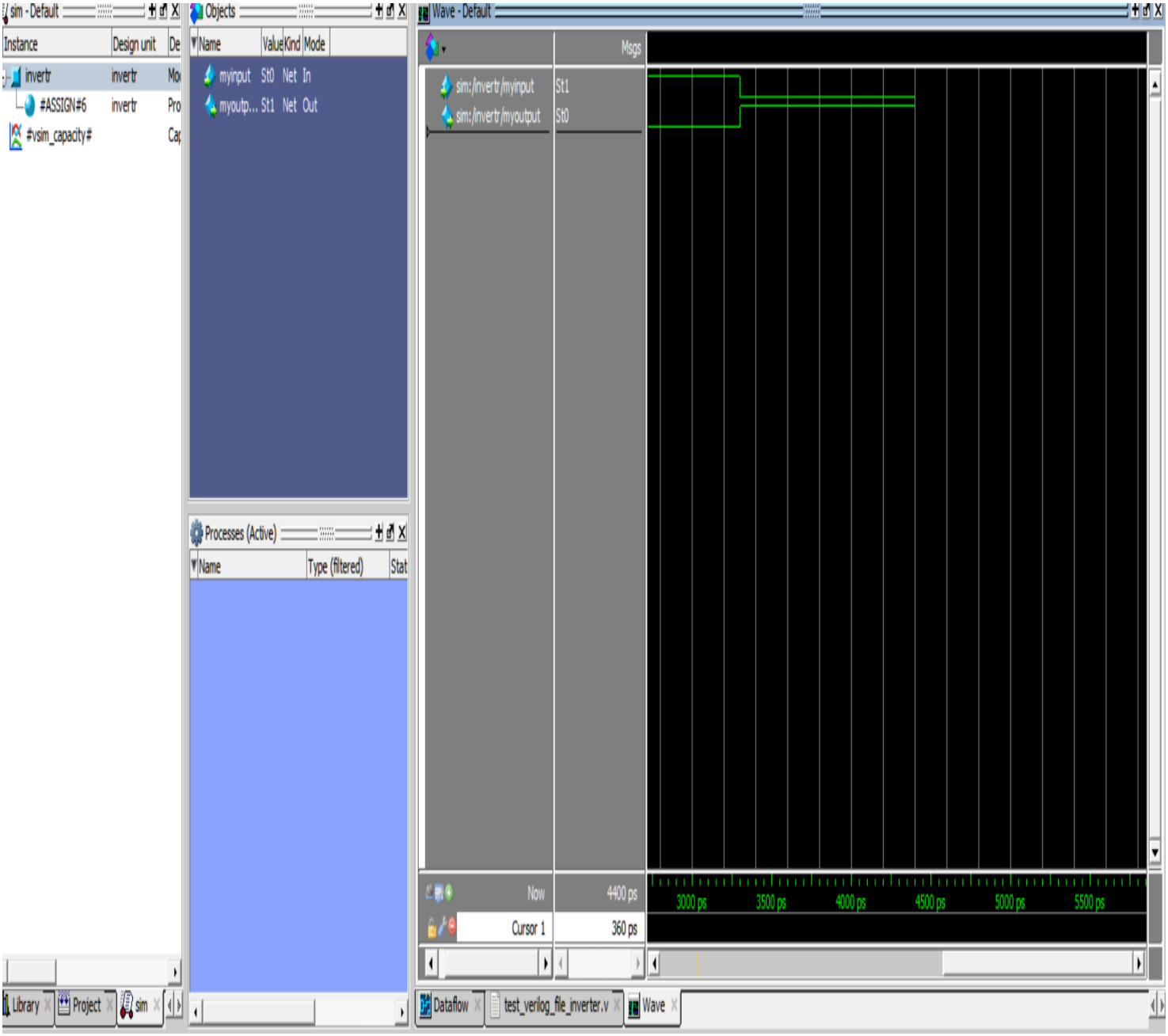
1. Verilog Simulation of a basic Inverter:

C:\altera\13.0sp1\test_verilog_file_inverter.v (/invertr) - Default

```
//model an inverter logic gate in modelsim

module invertr(myinput, myoutput);
  input myinput;
  output myoutput;
  assign myoutput = ~myinput;
  // if input is 1, output is 0
  // if input is 0, output is 1

endmodule
```

Dataflow | test_verilog_file_inverter.v | Wave

Bimal parajuli (20BDS0405)



Bimal parajuli (20BDS0405)

2. Verilog Simulation of a half Adder:

C:/altera/13.0sp1/half_adder.v (/ha_tb)

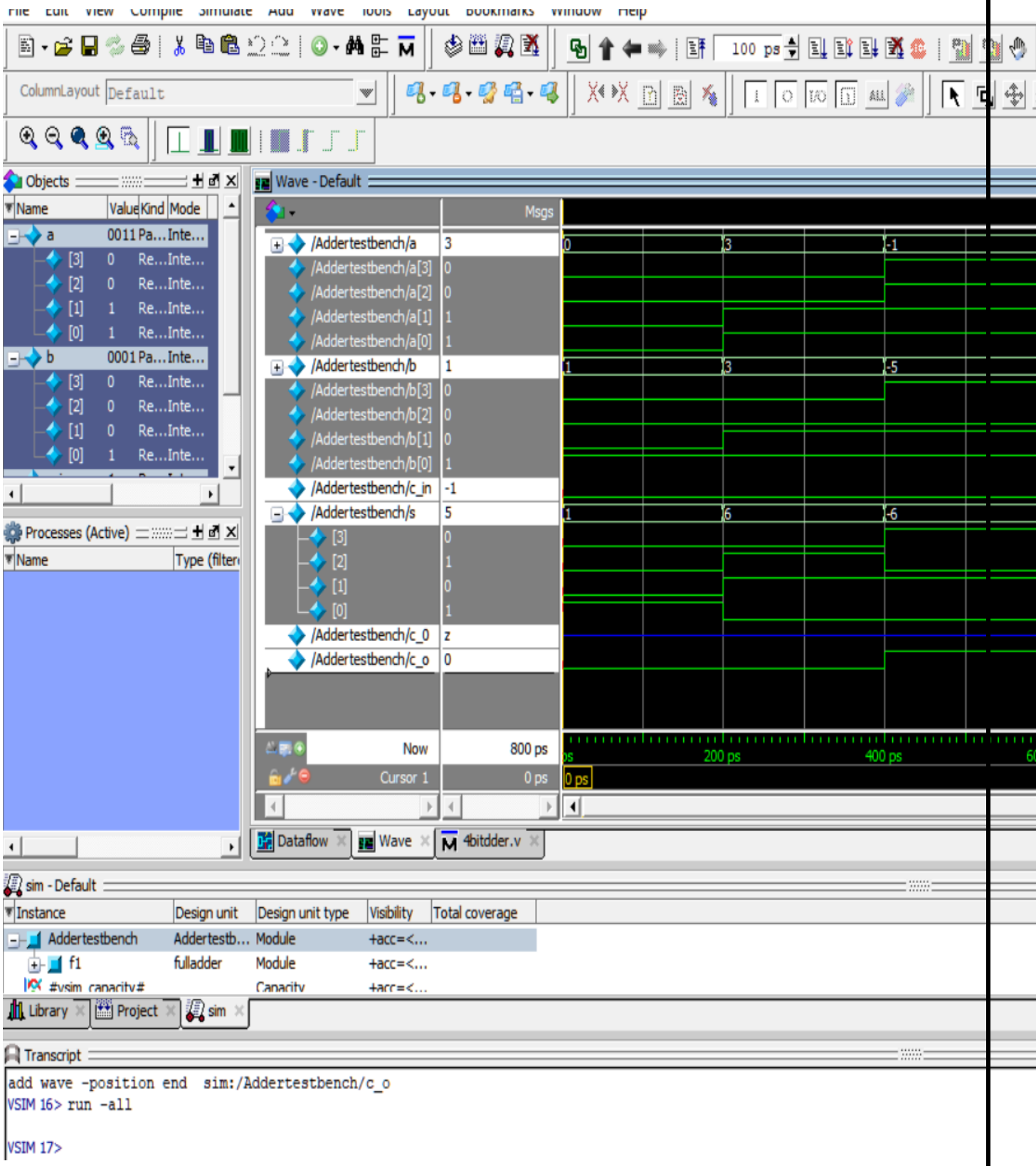
File Edit View Tools Bookmarks Window Help

C:/altera/13.0sp1/half_adder.v (/ha_tb) - Default

Ln#

```
////////////////////////////////////////////
//////Half adder Verilog Code////////////


module half_adder(A,B,Sum,Cout);

  input A,B;
  output Sum, Cout;
  xor(Sum,A,B);
  and(Cout,A,B);

endmodule



//////////////////////////////////////////////////
//////Test bench for half adder Circuit//////

module ha_tb();
  reg a,b;
  wire s,c;

  half_adder a1(a,b,s,cout);
initial
    begin
      a=1'b0; b=1'b0;
      #100

      a=1'b1; b=1'b0;
      #100

      a=1'b0; b=1'b1;
      #100

      a=1'b1; b=1'b1;

    end
endmodule
```

Bimal parajuli (20BDS0405)



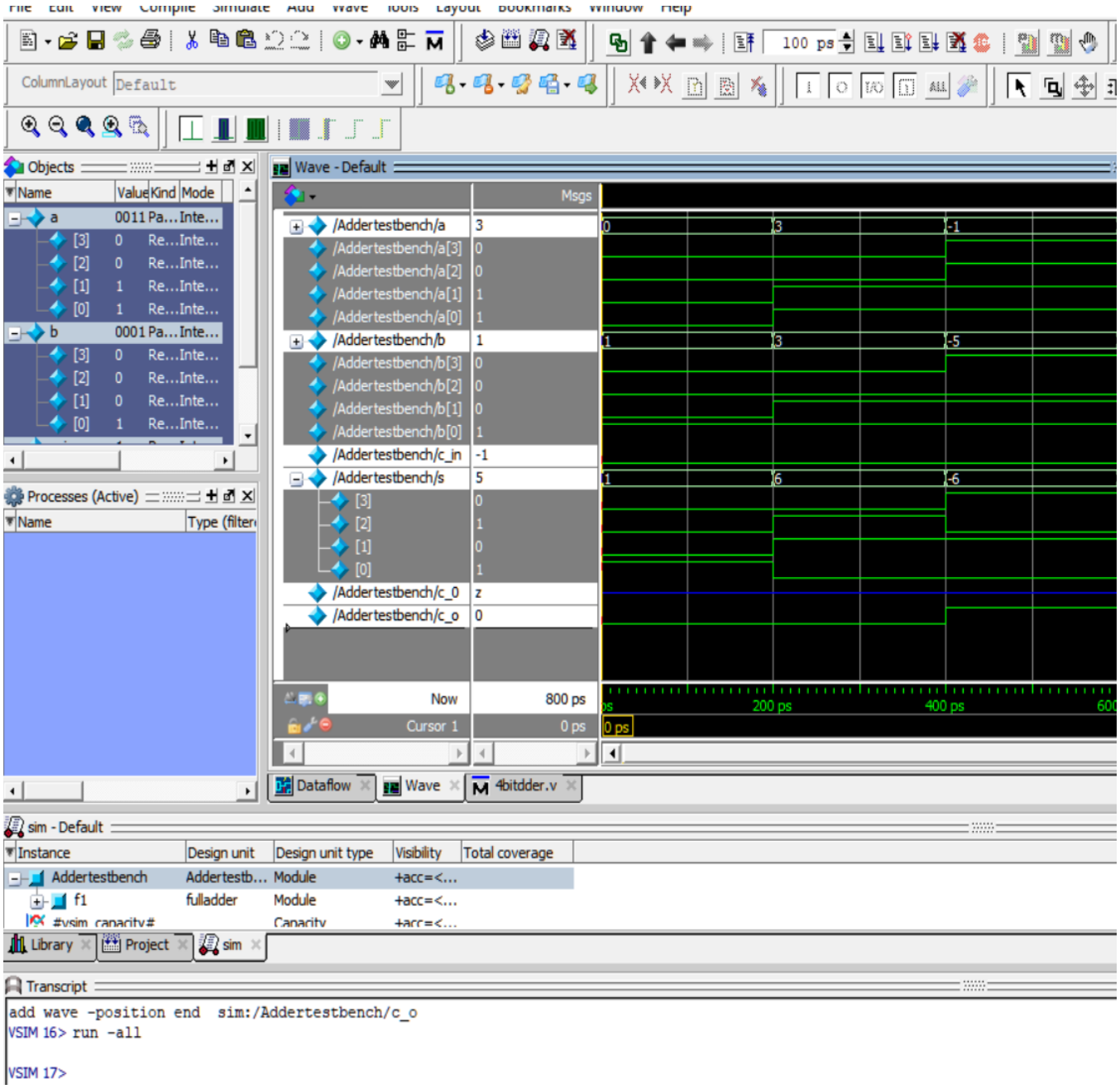
Bimal parajuli (20BDS0405)

3. Verilog Simulation of a Full Adder:

C:/altera/13.0sp1/4bitdder.v (/Addertestbench) - Default

```
//////////////////////////////
module fulladder(A,B,Cin,Sum,Cout);

  input [3:0] A, B;
  input Cin;

  output [3:0] Sum;
  output Cout;

  //Specifying the function of a full adder
  assign {Cout,Sum} = A + B + Cin;
endmodule

//////////////////////////////////

//////////////////////////////////

///////Test bench for above Module///
module Addertestbench();
  reg[3:0] a,b;
  reg c_in;
  wire [3:0] s;
  wire c_0;

  fulladder f1(.A(a),.B(b),.Cin(c_in),.Sum(s),.Cout(c_o));

initial
  begin
      a= 4'b0000; b=4'b0001; c_in =1'b0;
      #200;

      a= 4'b0011; b=4'b0011; c_in =1'b0;
      #200;

      a= 4'b1111; b=4'b1011; c_in =1'b0;
      #200;

      a= 4'b0011; b=4'b0001; c_in =1'b1;
      #200;
    end
endmodule
```

Bimal parajuli (20BDS0405)



Bimal parajuli (20BDS0405)

4. Verilog Simulation of a 2 to 4 decoder:

C:/altera/13.0sp1/2_to_4_decoder.v (/decoder)

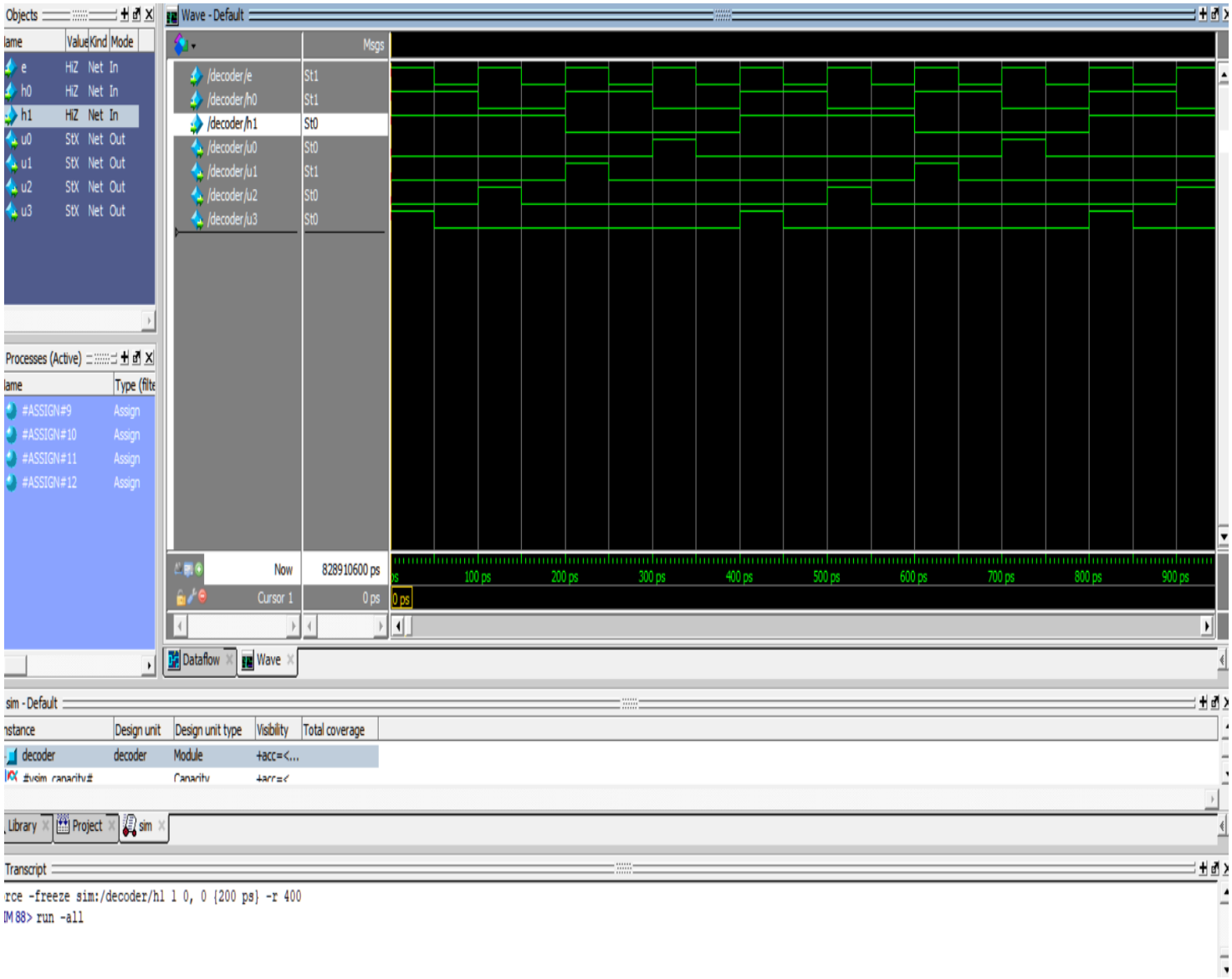
File Edit View Tools Bookmarks Window Help

C:/altera/13.0sp1/2_to_4_decoder.v (/decoder) - Default

Ln#

```
1   ////////////////////////////////////////////////////////////////////
2   ////////////////////2 to 4 decoder in verilog ///////////////////////
3
4   module decoder(u0,u1,u2,u3,e,h0,h1);
5
6       input e,h0,h1;
7       output u0, u1, u2, u3;
8
9       assign u0= (e & ~h1 & ~h0);
10      assign u1= (e & ~h1 & h0);
11      assign u2= (e & h1 & ~h0);
12      assign u3= (e & h1 & h0);
13
14    endmodule
15
16    ////////////////////////////////////////////////////////////////////
17    /////////////Test bench for 2 to 4 decoder ///////////////////////
18
19    module decoder_tb();
20
21      reg e,h0,h1;
22      wire u0,u1,u2,u3;
23
24      decoder d1(u0, u1, u2, u3, e, h0, h1);
25
26      initial
27        begin
28          e=0; h0=1; h1=0;
29          #100;
30          e=1; h0=0; h1=0;
31          #100;
32          e=1; h0=0; h1=1;
33          #100;
34          e=1; h0=1; h1=0;
35          #100;
36          e=1; h0=1; h1=1;
37        end
38      endmodule
39
```

Bimal parajuli (20BDS0405)



Bimal parajuli (20BDS0405)

5. Verilog Simulation of 2421 to 53-1-1 Code Converter (DA_2).

C:/altera/13.0sp1/code_Converter.v (/converter_tb)

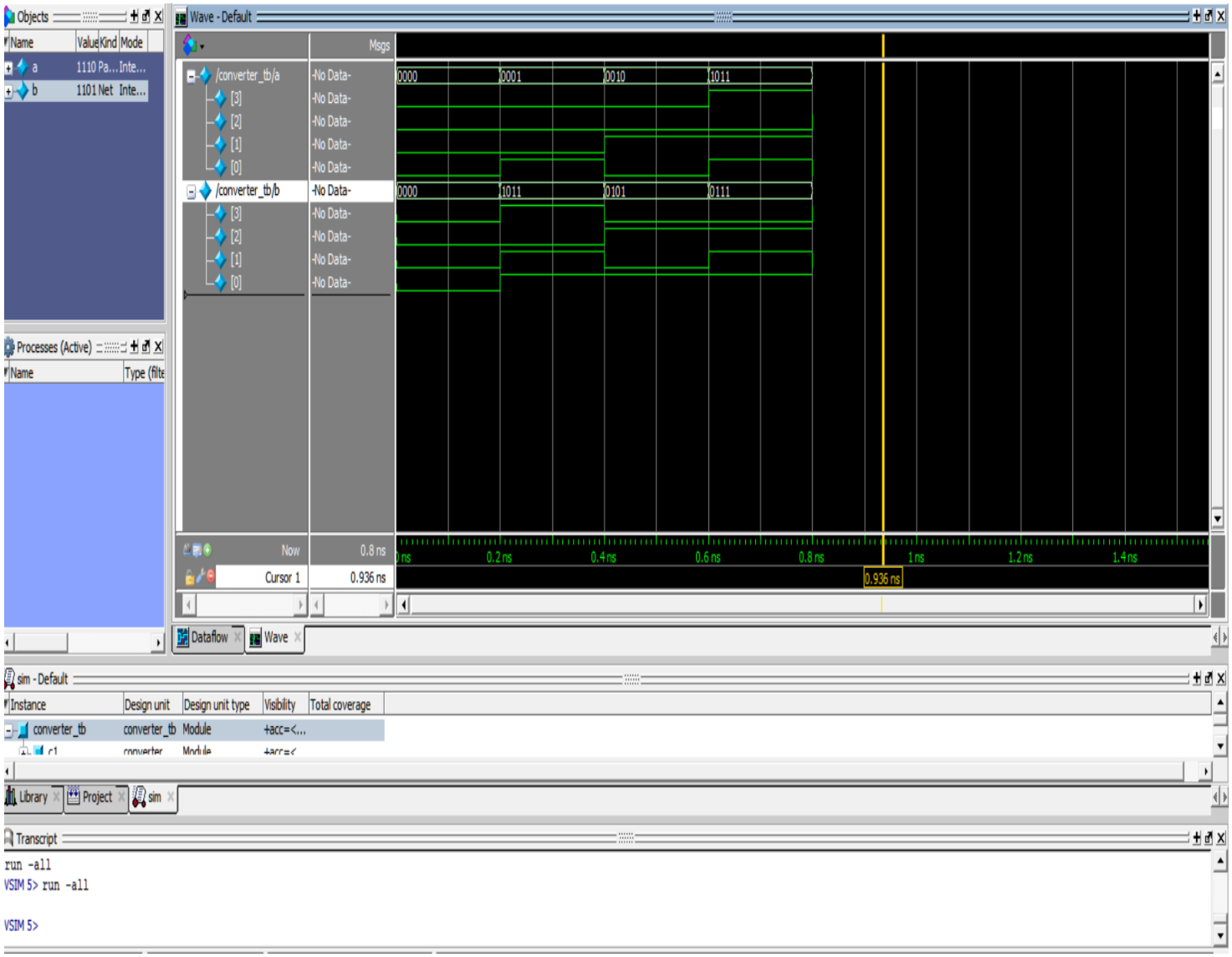
File Edit View Tools Bookmarks Window Help

C:/altera/13.0sp1/code_Converter.v (/converter_tb) - Default

Ln#

```
1   ////////////////////////////////////////////////////////////////////////////////
2   ///Verilog Code to design a module for converting 4 bit 2421 to 53-1-1 code///////
3
4   module converter(A,B);
5
6     input [3:0] A;
7     output [3:0] B;
8
9         assign B[0] = A[1] | A[0] | (A[2] & A[3]);
10        assign B[1] = (~A[2] & A[3]) | (A[2] & ~A[3]) | (~A[2] & A[0]);
11        assign B[2] = (A[1] & ~A[2]) | (~A[2] & A[3] ) | (~A[0] & A[2]);
12        assign B[3] = (A[0] & ~A[2] & ~A[3]) | ( ~A[0] & A[3] );
13  endmodule
14
15
16  ////////////////////////////////////////////////////
17  ///////Test bench for above code converte/////////
18
19  module converter_tb();
20    reg[3:0] a;
21    wire [3:0] b;
22
23    converter c1(.A(a), .B(b));
24
25    initial
26        begin
27            a= 4'b0000;
28            #200;
29            a= 4'b0001;
30            #200;
31            a= 4'b0010;
32            #200;
33            a= 4'b1011;
34            #200;
35            a= 4'b1110;
36        end
37  endmodule
38
```

Bimal parajuli (20BDS0405)



CONCLUSION/ INFERENCES:

From the above experiments, we can observe that any digital circuits can be simulated by writing a proper Verilog(HDL) code of the corresponding circuit. HDL compilers like ModelSim help to convert the code into it's circuit equivalent which can further be tested with a test bench and the design can be further implemented in FPGAs if required.