SCHOOL OF ENGINEERING AND TECHNOLOGY

Department of Electronics and Communication Engineering

EC631 – VLSI DESIGN LABORATORY

NAME: PREM KUMAR R

REG. NO: 1960628

YEAR/BRANCH: 3/ECE

SEMESTER: 6th

CHRIST (DEEMED TO BE UNIVERSITY)
SCHOOL OF ENGINEERING AND TECHNOLOGY
Mysore Road, Kanmanike, Bangalore – 560074

LABORATORY CERTIFICATE

This is to certify that **Mr./Ms**. …Prem Kumar R……..…… has satisfactorily completed the course of experiments in **EC631 - VLSI Design Laboratory** prescribed by the department of **Electronics and Communication Engineering, CHRIST (Deemed to be University)** for **VI** semester **Bachelor of Technology** Course in the laboratory of this university during the year **2021 – 2022**.

Signature of Staff-In charge

Signature of Head of the Department

Name of the Candidate …….

Register Number ……...

Examination Center ……...

Date of Practical Examination ……...

Signature of the Examiners:

1.

2.

SYLLABUS

EC631 - VLSI DESIGN LABORATORY

LIST OF EXPERIMENTS:

1) Design Entry and Simulation of Combinational Logic circuits

   a) Basic logic gates

   b) Multiplexer and Demultiplexer

   c) Encoder and Decoder

   d) Half adder and Full adder

   e) Half Subtractor and Full Subtractor

   f) 8 bit adder

   g) 4 bit multiplier

2) Design Entry and Simulation of Sequential Logic Circuits

   a) Flip-Flops

   b) Counters

   c) Registers

3) Synthesis, P&R and Post P&R simulation for all the blocks/codes developed in Expt. No. 1 and No. 2.

4) Design Entry and Simulation of traffic signal controller using Xilinx ISE Design suite and implementing the same on Spartan FPGA.

5) Schematic and Layout of a simple CMOS inverter, parasitic extraction and simulation.

6) Design and simulation of pipelined serial and parallel adder to add/ subtract 8 number of size, 12 bits each in 2's complement.

7) Design and Implement a 4 digit seven segment display.

CONTENTS

S.NO.	DATE	NAME OF THE EXPERIMENT	MARKS OBTAINED	SIGNATURE OF STAFF
1.		Design Entry and Simulation of Combinational Logic circuits		
1. a)	12/01/2022	Basic logic gates		
1. b)	12/01/2022	Multiplexer and De-multiplexer		
1. c)		Encoder and Decoder		
1. d)		Half adder and Full adder		
1. e)		Half subtractor and Full subtractor		
1. f)		8-bit adder		
1. g)		4-bit multiplier		
2.		Design Entry and Simulation of Sequential Logic Circuits		
2. a)		Flip-Flops		
2. b)		Counters		
2. c)		Registers		
3.		Synthesis, P&R and Post P&R simulation for all the blocks/codes developed in Expt. No. 1 and No. 2.		
4.		Design Entry and Simulation of traffic signal controllers using Xilinx Vivado		
5.		Layout of a simple CMOS inverter and simulation.		
6.		Design and Implement a Binary to Seven Segment Display		

Completed/Incomplete Staff In-Charge Signature

EX.NO: 1 DESIGN ENTRY AND SIMULATION OF COMBINATIONAL LOGIC CIRCUITS

DATE: 12/01/2022

AIM:

To verify the functionality and timing of your design or portion of your design. To interpret Verilog code into circuit functionality and displays logical results of the described HDL to determine correct circuit operation and to create and verify complex functions in a relatively small amount of time.

TOOLS REQUIRED:

1. Xilinx Vivado Design Suite: WebEdition

PROCEDURE:

1. Start by clicking **Vivado Design Suite: WebEdition**

2. Go to **File → new project → Enter project name**, select the top level source as **HDL** & click **next**.

3. Enter **device properties** as

Product Category : General Purpose
Family : Kintex 7
Device : xc7k70t
Package :fbg484
Speed : -1
Top Level Source Type : HDL
Synthesis Tool : XST (VHDL/Verilog)
Simulator : ISim (VHDL/Verilog)
Preferred Language : Verilog

& **Click next**.

4. **Right click** the device name (XCS3540) in the source window to create **new**

source.

5. Select **Verilog module** and enter **file name** in the new source window & click **next**.

6. Write the **Verilog code** in the **Verilog editor window**.

7. Write the **testbench** by create **new source simulation source.**

8. Run Check syntax through **Process window→ synthesize→** double click **Behavioral Check Syntax→** and removes error if present, with proper syntax & coding.

9. Click on the symbol of FPGA device and then right click→ click on **new source**.

10. Select the desired parameters for simulating the design. In this case **combinational circuit** and **simulation time** click **finish**.

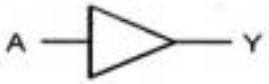
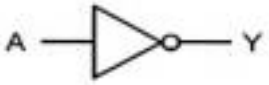
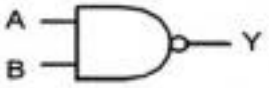
11. Assign **all input signal** (**high or low**) using just **click** on this and save file.

12. From the **source process window**. Click **Behavioral simulation** from drop-down menu.

13. Double click the **Simulation Behavioral Model**.

14. Verify your design in **wave window** by seeing behavior of output signal with respect to input signal.

BASIC LOGIC GATES:

Logic function	Logic symbol	Truth table	Boolean expression
Buffer		A | Y 0 | 0 1 | 1	$Y = A$
Inverter (NOT gate)		A | Y 0 | 1 1 | 0	$Y = \overline{A}$
2-input AND gate		A B | Y 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1	$Y = A \bullet B$
2-input NAND gate		A B | Y 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0	$Y = \overline{A \bullet B}$
2-input OR gate		A B | Y 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1	$Y = A + B$
2-input NOR gate		A B | Y 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0	$Y = \overline{A + B}$
2-input EX-OR gate		A B | Y 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0	$Y = A \oplus B$
2-input EX-NOR gate		A B | Y 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1	$Y = \overline{A \oplus B}$

PROGRAM:

a) BASIC LOGIC GATES:

1) GATE LEVEL MODELING:

```
22  //1960628 prem
23  module basic_gates(
24      input i1,
25      input i2,
26      output not_out,
27      output and_out,
28      output or_out,
29      output xor_out,
30      output nor_out,
31      output nand_out,
32      output xnor_out,
33      output buf_out
34      );
35
36  not no(not_out,i1);
37  and an(and_out,i1,i2);
38  or ort(or_out,i1,i2);
39  xor xort(xor_out,i1,i2);
40  nor nort(nor_out,i1,i2);
41  nand nandt(nand_out,i1,i2);
42  xnor (xnort,i1,i2);
43  buf buffet(buf_out,i1);
44  endmodule
45
```

2) DATAFLOW LEVEL MODELING:

```
47  //DATAFLOW
48  //1960628 PREM
49  module basic_gates(
50      input i1,
51      input i2,
52      output not_out,
53      output and_out,
54      output or_out,
55      output xor_out,
56      output nor_out,
57      output nand_out,
58      output xnor_out,
59      output buf_out
60      );
61
62  assign not_out =~i1;
63  assign and_out =i1&i2;
64  assign or_out=i1|i2;
65  assign xor_out=i1^i2;
66  assign nor_out=~(i1|i2);
67  assign nand_out= ~(i1&i2);
68  assign xnor_out=~(i1^i2);
69  assign buf_out = i1;
70  endmodule
```

BEHAVIORAL LEVEL MODELING:

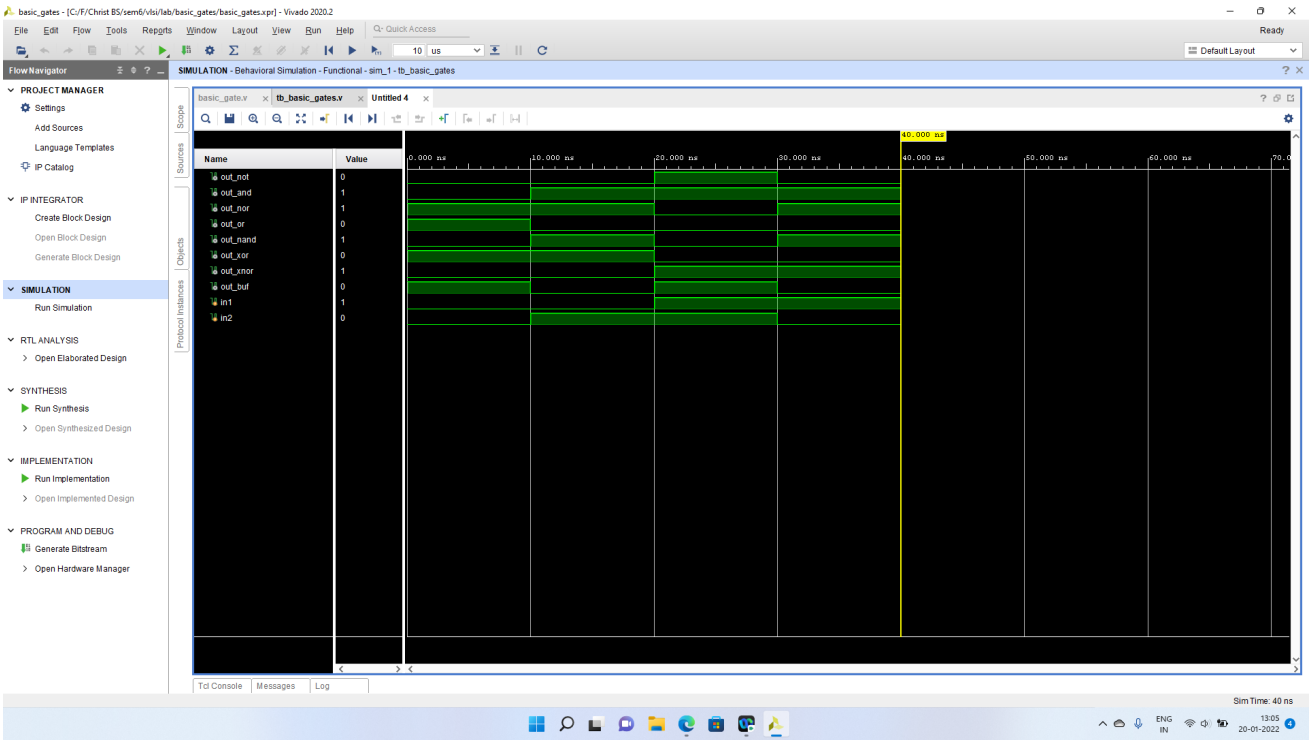
```
77  //1960628 Prem
78  //behavioural basic gate
79  module basic_gate(
80      input in1,
81      input in2,
82      output reg out_and,
83      output reg out_or,
84      output reg out_nand,
85      output reg out_nor,
86      output reg out_xor,
87      output reg out_not,
88      output reg out_buf,                                    _out, buf_out;
89      output reg out_xnor
90
91      );
92
93  always @(in1,in2)
94  begin
95  if(in1==0&&in2==0)
96  begin
97  out_not = 1;
98  out_and = 0;
99  out_nand = 1;
100 out_or = 0;
101 out_nor = 1;
102 out_xor = 0;
103 out_xnor = 1;
104 out_buf = 0;
105 end
106 if(in1==1&&in2==0)
107 begin
108 out_not = 0;
109 out_and = 0;
110 out_nand = 1;
111 out_or = 0;
112 out_nor = 1;
113 out_xor = 0;
114 out_xnor = 1;
115 out_buf = 1;
116 end
```

```
 99  end
100  if(i1==0&&i2==1)
101  begin
102   not_out =1;
103   and_out =0;
104   or_out=0;
105   xor_out=0;
106   nor_out=1;
107   nand_out=1;
108   xnor_out=1;
109   buf_out =0;
110  end
111  if(i1==1&&i2==1)
112  begin
113   not_out =0;
114   and_out =0;
115   or_out=1;
116   xor_out=0;
117   nor_out=0;
118   nand_out=0;
119   xnor_out=1;
120   buf_out =1;
121  end
122  endmodule
```

TEST BENCH:

```
21
22  //1960628 PREM
23  module tb_basic_gates();
24  wire not_out,and_out,or_out,xor_out,nor_out,nand_out,xnor_out, buf_out;
25  reg i1,i2;
26
27  basic_gates I1( not_out,and_out,or_out,xor_out,nor_out,nand_out,xnor_out, buf_out,i1,i2);
28  initial
29  begin
30  i1 = 1'b0;
31  i2 = 1'b0;
32  #10
33  i1 = 1'b0;
34  i2 = 1'b1;
35  #10
36  i1 = 1'b1;
37  i2 = 1'b0;
38  #10
39  i1 = 1'b1;
40  i2 = 1'b1;
41  #10
42  $finish;
43  end
44  endmodule
```

SIMULATION OUTPUT:



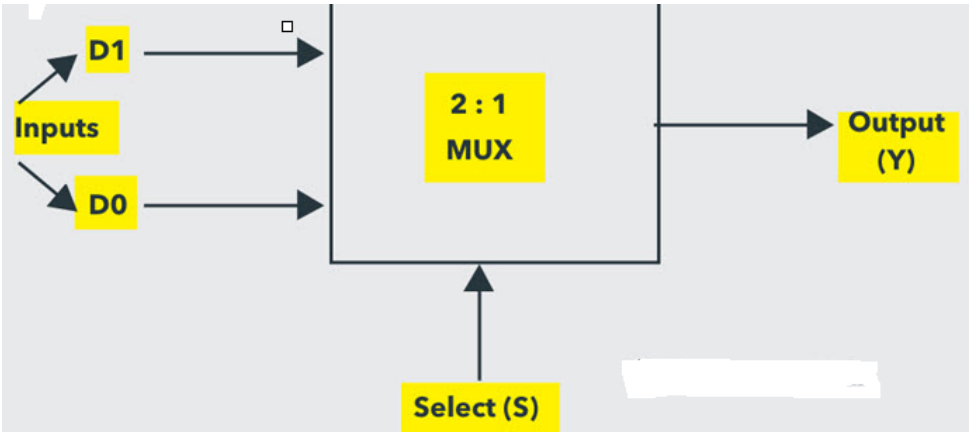
Result:

Thus the Verilog code was scripted, simulated and waveforms were verified using Xilinx Vivado Webpack.

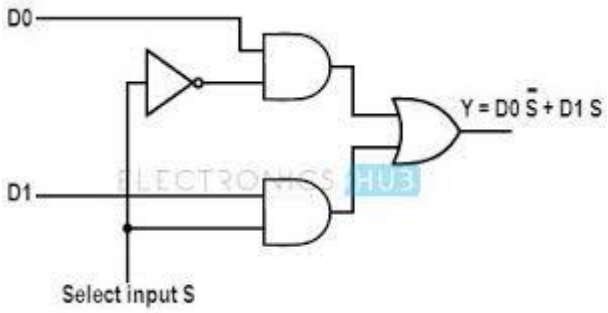
D) Multiplexer and Demultiplexer:

MULTIPLEXER (2x1):

Logic Symbol



Logic Circuit



Truth Table

Select	Inputs		Output
0	0	0	0
0	0	1	1
1	1	0	1
1	1	1	1

PROGRAM

1) GATE LEVEL MODELING:

```
15      //
16      // Revision:
17      // Revision 0.01 - File Created
18      // Additional Comments:
19      //
20      //////////////////////////////////////////////////////////////////////////////////
21
22      //1960628 PREM
23      module mux_demux(d0,d1,s,y);
24
25      input d0,d1,s;
26      output y;
27
28      wire w0,w1,w3;
29      not not_0(w0,s);
30      and and_1(w2,w0,d0);
31      and and_2(w3,d1,s);
32      or out(y,w2,w3);
33
34      endmodule
35
```

2) DATAFLOW LEVEL MODELING:

```
21
22  //1960628 PREM
23  // DATAFLOW MODELLING
24  module mux_demux(d0,d1,s,y);
25  input d0,d1,s;
26  output y;
27  wire w0,w1,w3;
28  assign not_0 = ~(s);
29  assign and_0 = not_0&d0;
30  assign and_1 = d1&s;
31  assign y = and_0 + and_1;
32  endmodule
33
```

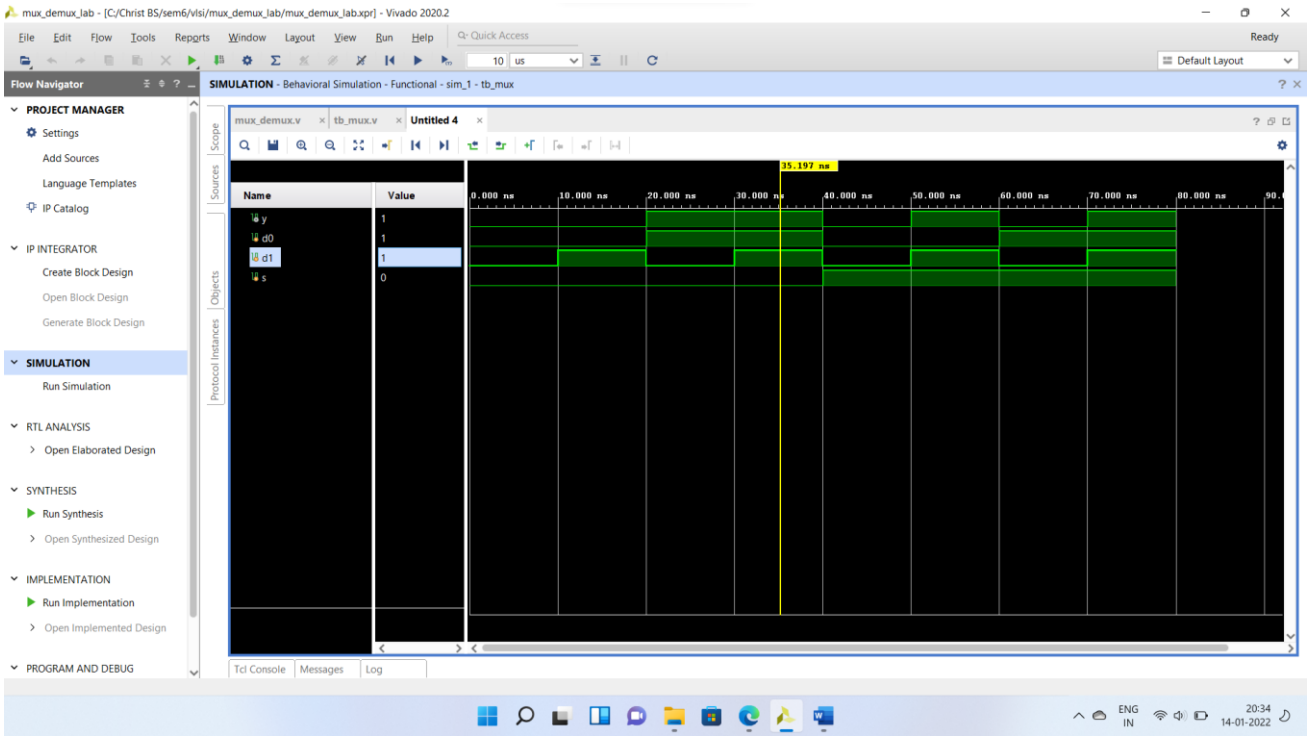
3) BEHAVIORAL LEVEL MODELING:

```
21
22  //1960628 PREM
23  // BEHAVIORAL,DATAFLOW,MODELLING
24  module mux_demux(d0,d1,s,y);
25  input d0,d1,s;
26  output y;
27  /*data flow commented out
28  wire w0,w1,w3;
29  assign not_0 = ~(s);
30  assign and_0 = not_0&d0;
31  assign and_1 = d1&s;
32  assign y = and_0 + and_1;*/
33  assign y = (s==1)?d1:d0;
34  endmodule
35
```

TESTBENCH

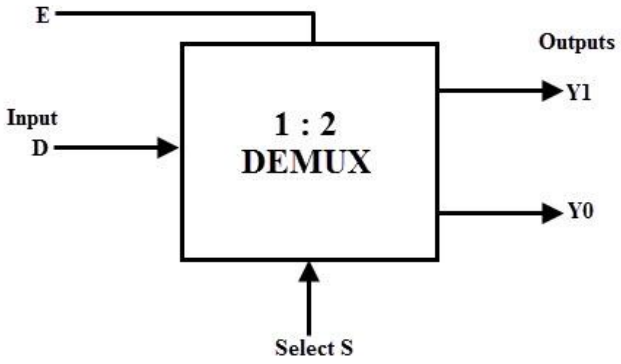
```
21
22      //1960628 PREM
23      module tb_mux();
24      wire y;
25      reg d0,d1,s;
26      mux_demux I1(d0,d1,s,y);
27      initial
28      begin
29      d0 = 1'b0;
30      d1 = 1'b0;
31      s = 1'b0;
32      #10
33      d0 = 1'b0;
34      d1 = 1'b1;
35      s = 1'b0;
36      #10
37      d0 = 1'b1;
38      d1 = 1'b0;
39      s = 1'b0;
40      #10
41      d0 = 1'b1;
42      d1 = 1'b1;
43      s = 1'b0;
44      #10
45      d0 = 1'b0;
46      d1 = 1'b0;
47      s = 1'b1;
48      #10
49      d0 = 1'b0;
50      d1 = 1'b1;
51      s = 1'b1;
52      #10
53      d0 = 1'b1;
54      d1 = 1'b0;
55      s = 1'b1;
56      #10
57      d0 = 1'b1;
58      d1 = 1'b1;
59      s = 1'b1;
60      #10
61      $finish;
62      end
63      endmodule
64
```

SIMULATION OUTPUT

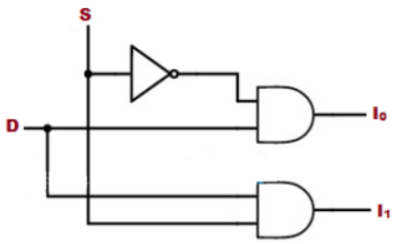


DE-MULTIPLEXER (1x2):

Logic Symbol



Logic Circuit



Truth Table

Select	Input	Outputs	
S	D	Y_2	Y_1
0	0	0	0
0	1	0	1
1	0	0	0
1	1	1	0

PROGRAM

1) **GATE LEVEL MODELING:**

```
//1960628 PREM
//GATE LEVEL MODELLING
module demux_comb(y0,y1,d,s);
input s;
input d;
output y0;
output y1;
wire w0;
not not_0(w0,s);
and and_1(y0,w0,d);
and and_2(y1,d,s);
endmodule
```

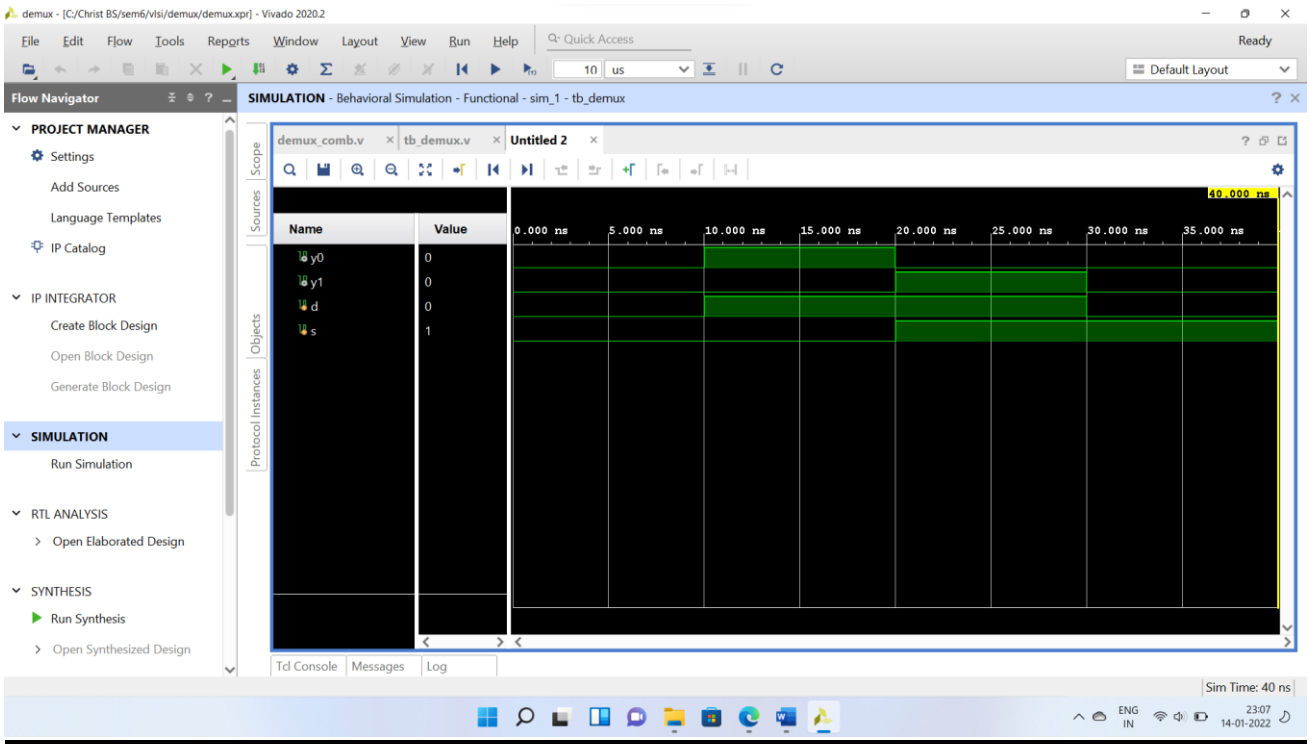
2) DATAFLOW LEVEL MODELING:

```
//DATA FLOW MODELLING
//1960628 PREM
module demux_comb(y0,y1,d,s);
input s;
input d;
output y0;
output y1;
//DATAFLOW
assign not_0 = ~s;
assign y0 = not_0&d;
assign y1 = s&d;
//GATE LEVEL
//wire w0;
//not not_0(w0,s);
//and and_1(y0,w0,d);
//and and_2(y1,d,s);
```

TESTBENCH

```
//1960628 PREM
//TEST BENCH FOR DEMUX 1X2
module tb_demux();
wire y0,y1;
reg d,s;
demux_comb I1(y0,y1,d,s);
initial
begin
d = 1'b0;
s = 1'b0;
#10
d = 1'b1;
s = 1'b0;
#10
d = 1'b1;
s = 1'b1;
#10
d = 1'b0;
s = 1'b1;
#10
$finish;
end
endmodule
```

SIMULATION OUTPUT



RESULT : A Mux and Demux were modelled in the Xilinx Vivado simulator and their output was verified.