DLD LAB WIN 2021

LAB 4

Aim: To simulate different types of logic gates DeMUX-1to4 and MUX-6to1

Tools used: vivado software

Truth tables:

DeMux-1to4:

Data Input	Select Inputs		Outputs			
D	S_1	S_0	Y_3	Y_2	Y_1	Y_0
D	0	0	0	0	0	D
D	0	1	0	0	D	0
D	1	0	0	D	0	0
D	1	1	D	0	0	0

MUX-16to1:

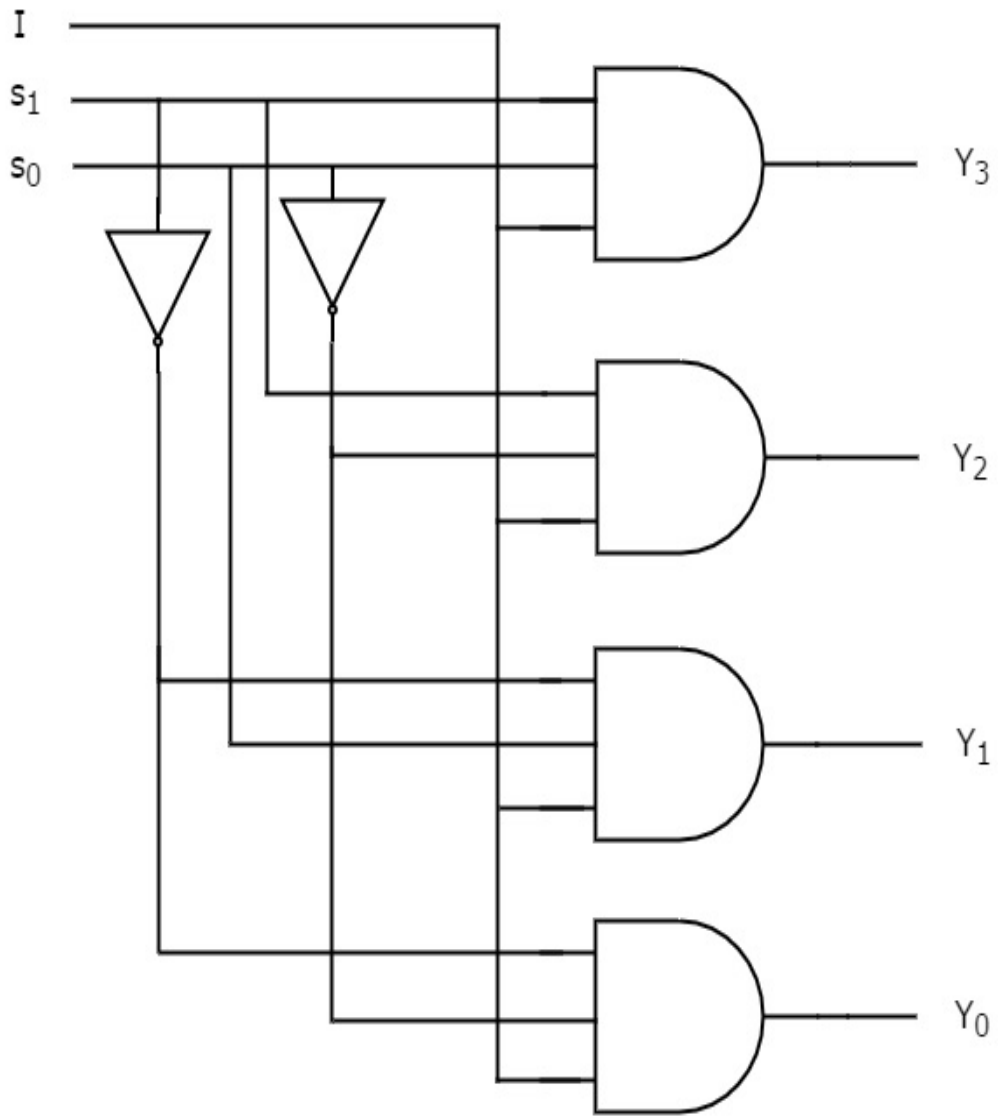
TRUTH TABLE

S0	S1	S2	S3	$\overline{E}$	SELECTED CHANNEL
X	X	X	X	1	None
0	0	0	0	0	0
1	0	0	0	0	1
0	1	0	0	0	2
1	1	0	0	0	3
0	0	1	0	0	4
1	0	1	0	0	5
0	1	1	0	0	6
1	1	1	0	0	7
0	0	0	1	0	8
1	0	0	1	0	9
0	1	0	1	0	10
1	1	0	1	0	11
0	0	1	1	0	12
1	0	1	1	0	13
0	1	1	1	0	14
1	1	1	1	0	15

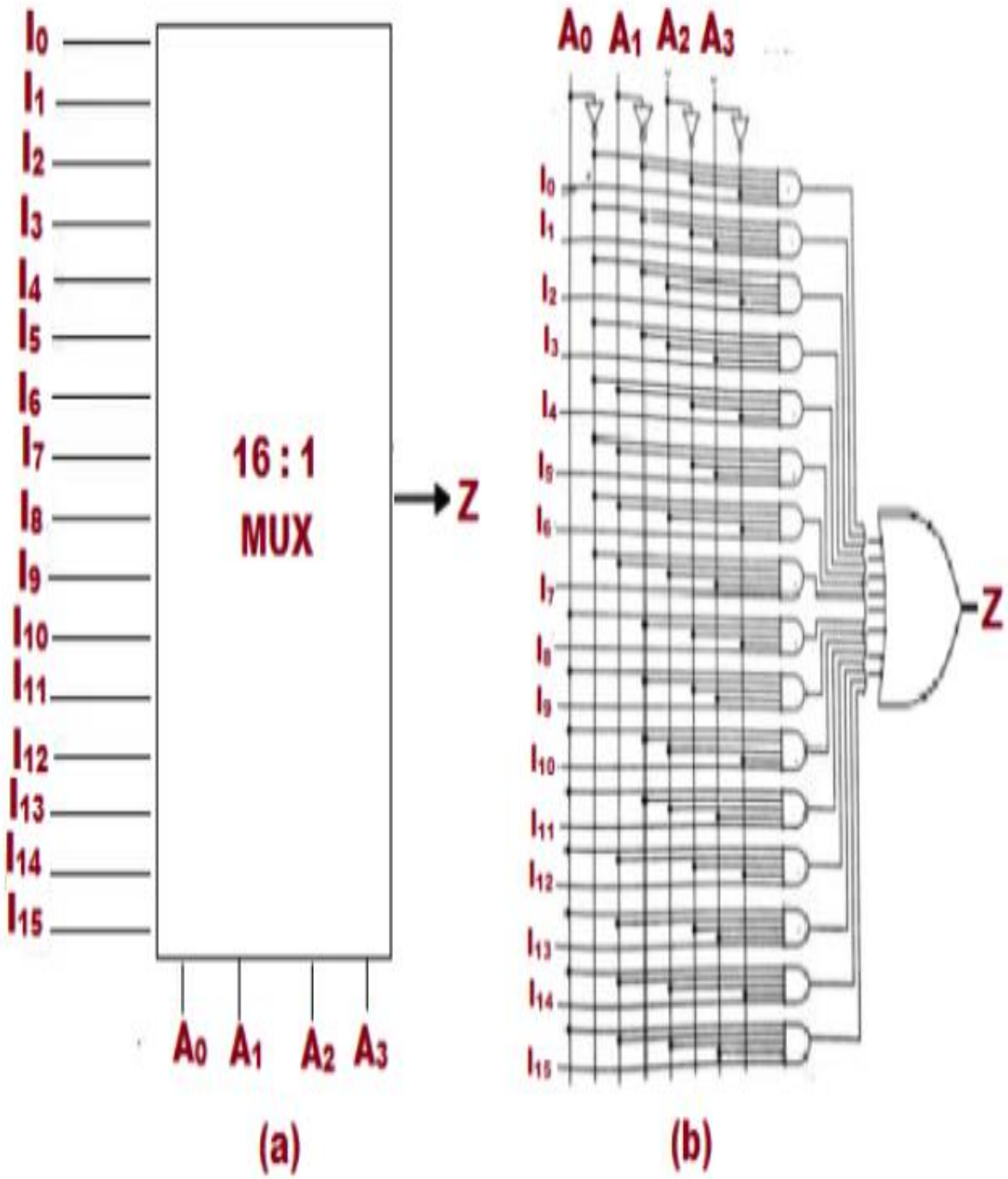
H= High Level
L= Low Level
X= Don't Care

Circuit diagrams

Demux-1to4:



Mux 16to1:


I0
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
I12
I13
I14
I15
16 : 1
MUX
Z
A0 A1 A2 A3
(a)
A0 A1 A2 A3
Z
(b)

Codes:

Demux-1to4:

```
19  //
20  //////////////////////////////////////////////////////////////////
21
22  module dem_1_4(i0,i1,i2,i3,s0,s1,z);
23  input s0,s1,z;
24  output i0,i1,i2,i3;
25  and3 f4((~s0),(~s1),z,i0);
26  and3 f2((~s0),s1,z,i1);
27  and3 f3(s0,(~s1),z,i2);
28  and3 f1(s0,s1,z,i3);
29  endmodule
30  module and3(a,b,c,z);
31  input a,b,c;
32  output z;
33  assign z = a & b & c;
34  endmodule
35
36
```

Test bench code demux 4:1:

```
module tb_dem_1_4;
reg s0,s1,z;
wire i0,i1,i2,i3;

dem_1_4 a2(i0,i1,i2,i3,s0,s1,z);
initial
begin
    z = 1'b1;
    s0 = 1'b0;s1 = 1'b0;
#10 s0 = 1'b0;s1 = 1'b1;
#10 s0 = 1'b1;s1 = 1'b0;
#10 s0 = 1'b1;s1 = 1'b1;
end
endmodule
```

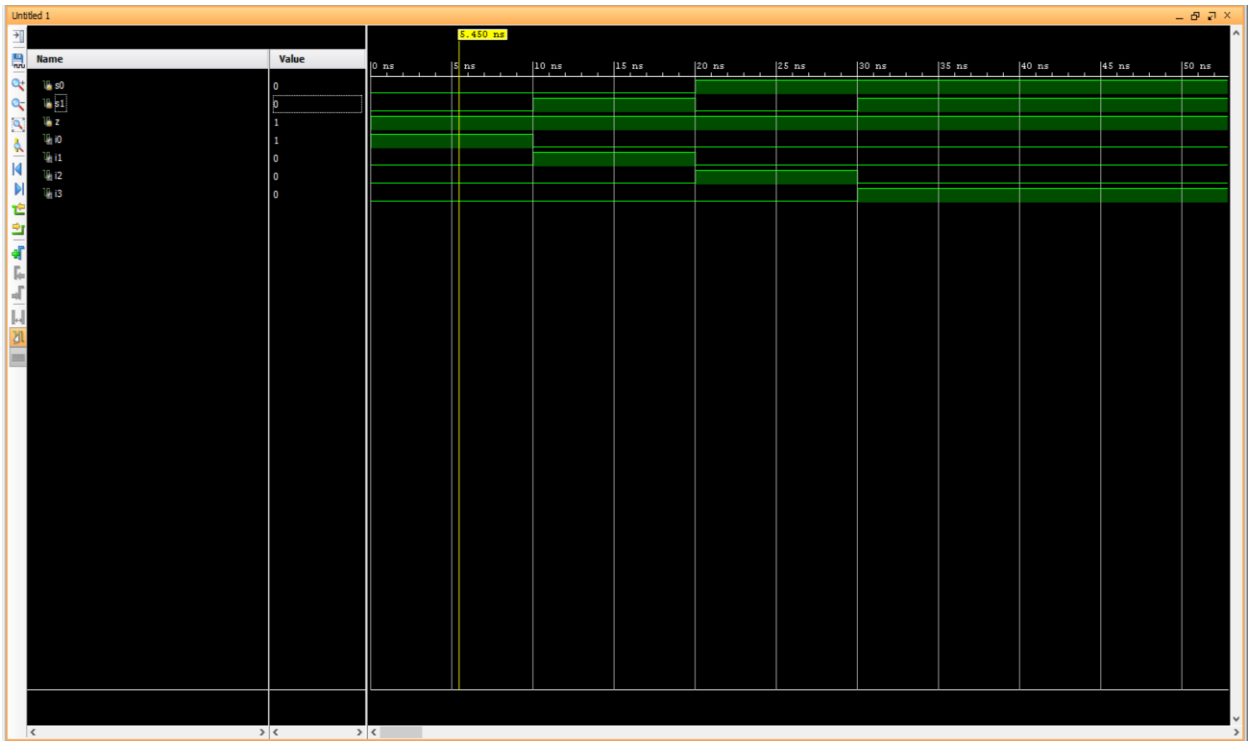
Mux 16:1

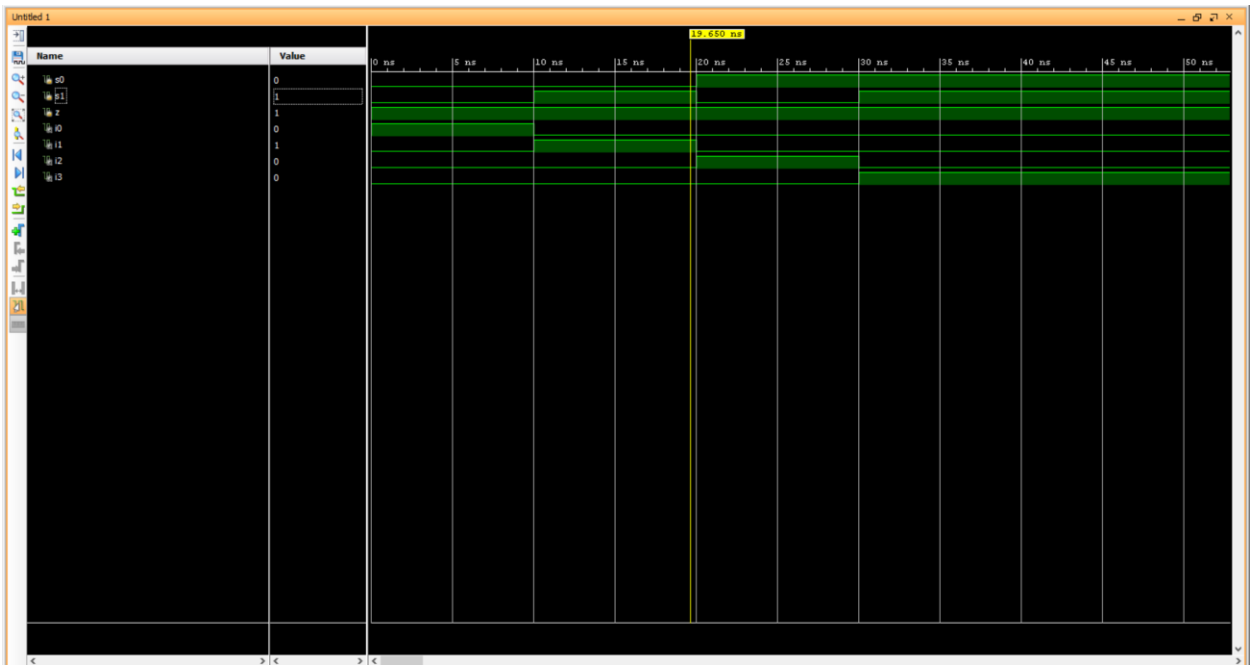
```
22
23    module mux16_1(i0,i1,i2,i3,i4,i5,i6,i7,i8,i9,i10,i11,i12,i13,i14,i15,s0,s1,s2,s3,z);
24    input i0,i1,i2,i3,i4,i5,i6,i7,i8,i9,i10,i11,i12,i13,i14,i15,s0,s1,s2,s3;
25    output reg z;
26    always@(i0,i1,i2,i3,i4,i5,i6,i7,i8,i9,i10,i11,i12,i13,i14,i15,s0,s1,s2,s3)
27    begin
28    if(s0 == 1'b0 && s1 == 1'b0 && s2 == 1'b0 && s3 == 1'b0)
29       z = i0;
30    else if(s0 == 1'b0 && s1 == 1'b0 && s2 == 1'b0 && s3 == 1'b1)
31       z = i1;
32    else if(s0 == 1'b0 && s1 == 1'b0 && s2 == 1'b1 && s3 == 1'b0)
33       z = i2;
34    else if(s0 == 1'b0 && s1 == 1'b0 && s2 == 1'b1 && s3 == 1'b1)
35       z = i3;
36    else if(s0 == 1'b0 && s1 == 1'b1 && s2 == 1'b0 && s3 == 1'b0)
37       z = i4;
38    else if(s0 == 1'b0 && s1 == 1'b1 && s2 == 1'b0 && s3 == 1'b1)
39       z = i5;
40    else if(s0 == 1'b0 && s1 == 1'b1 && s2 == 1'b1 && s3 == 1'b0)
41       z = i6;
42    else if(s0 == 1'b0 && s1 == 1'b1 && s2 == 1'b1 && s3 == 1'b1)
43       z = i7;
44    else if(s0 == 1'b1 && s1 == 1'b0 && s2 == 1'b0 && s3 == 1'b0)
45       z = i8;
46    else if(s0 == 1'b1 && s1 == 1'b0 && s2 == 1'b0 && s3 == 1'b1)
47       z = i9;
48    else if(s0 == 1'b1 && s1 == 1'b0 && s2 == 1'b1 && s3 == 1'b0)
49       z = i0;
50    else if(s0 == 1'b1 && s1 == 1'b0 && s2 == 1'b1 && s3 == 1'b1)
51       z = i11;
52    else if(s0 == 1'b1 && s1 == 1'b1 && s2 == 1'b0 && s3 == 1'b0)
53       z = i12;
54    else if(s0 == 1'b1 && s1 == 1'b1 && s2 == 1'b0 && s3 == 1'b1)
55       z = i13;
56    else if(s0 == 1'b1 && s1 == 1'b1 && s2 == 1'b1 && s3 == 1'b0)
57       z = i14;
58    else
59       z = i15;
60    end
61    endmodule
62
```

Testbench code mux 16:1

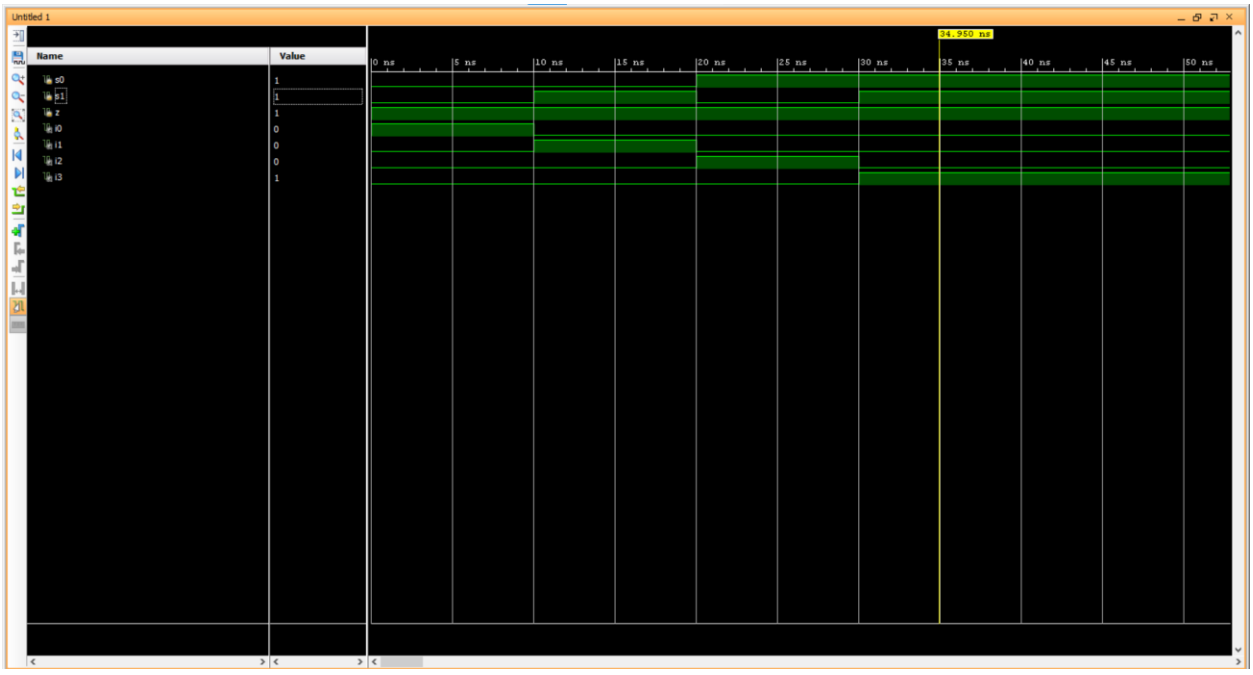
```
module tb_mux16_1;
reg i0,i1,i2,i3,i4,i5,i6,i7,i8,i9,i10,i11,i12,i13,i14,i15,s0,s1,s2,s3;
wire z;

mux16_1 x3(i0,i1,i2,i3,i4,i5,i6,i7,i8,i9,i10,i11,i12,i13,i14,i15,s0,s1,s2,s3,z);
initial
begin

    i0 = 1'b0;i1 = 1'b1;i2 = 1'b0;i3 = 1'b1;
    i4 = 1'b0;i5 = 1'b1;i6 = 1'b0;i7 = 1'b1;
    i8 = 1'b0;i9 = 1'b1;i10 = 1'b0;i11 = 1'b1;
    i12 = 1'b0;i13 = 1'b1;i14 = 1'b0;i15 = 1'b1;
    s0 = 1'b0;s1 = 1'b0;s2 = 1'b0;s3 = 1'b0;
#10 s0 = 1'b0;s1 = 1'b0;s2 = 1'b0;s3 = 1'b1;
#10 s0 = 1'b0;s1 = 1'b0;s2 = 1'b1;s3 = 1'b0;
#10 s0 = 1'b0;s1 = 1'b0;s2 = 1'b1;s3 = 1'b1;
#10 s0 = 1'b0;s1 = 1'b1;s2 = 1'b0;s3 = 1'b0;
#10 s0 = 1'b0;s1 = 1'b1;s2 = 1'b0;s3 = 1'b1;
#10 s0 = 1'b0;s1 = 1'b1;s2 = 1'b1;s3 = 1'b0;
#10 s0 = 1'b0;s1 = 1'b1;s2 = 1'b1;s3 = 1'b1;
#10 s0 = 1'b1;s1 = 1'b0;s2 = 1'b0;s3 = 1'b0;
#10 s0 = 1'b1;s1 = 1'b0;s2 = 1'b0;s3 = 1'b1;
#10 s0 = 1'b1;s1 = 1'b0;s2 = 1'b1;s3 = 1'b0;
#10 s0 = 1'b1;s1 = 1'b0;s2 = 1'b1;s3 = 1'b1;
#10 s0 = 1'b1;s1 = 1'b1;s2 = 1'b0;s3 = 1'b0;
#10 s0 = 1'b1;s1 = 1'b1;s2 = 1'b0;s3 = 1'b1;
#10 s0 = 1'b1;s1 = 1'b1;s2 = 1'b1;s3 = 1'b0;
#10 s0 = 1'b1;s1 = 1'b1;s2 = 1'b1;s3 = 1'b1;
end
endmodule
```

Output Demux 4to1:

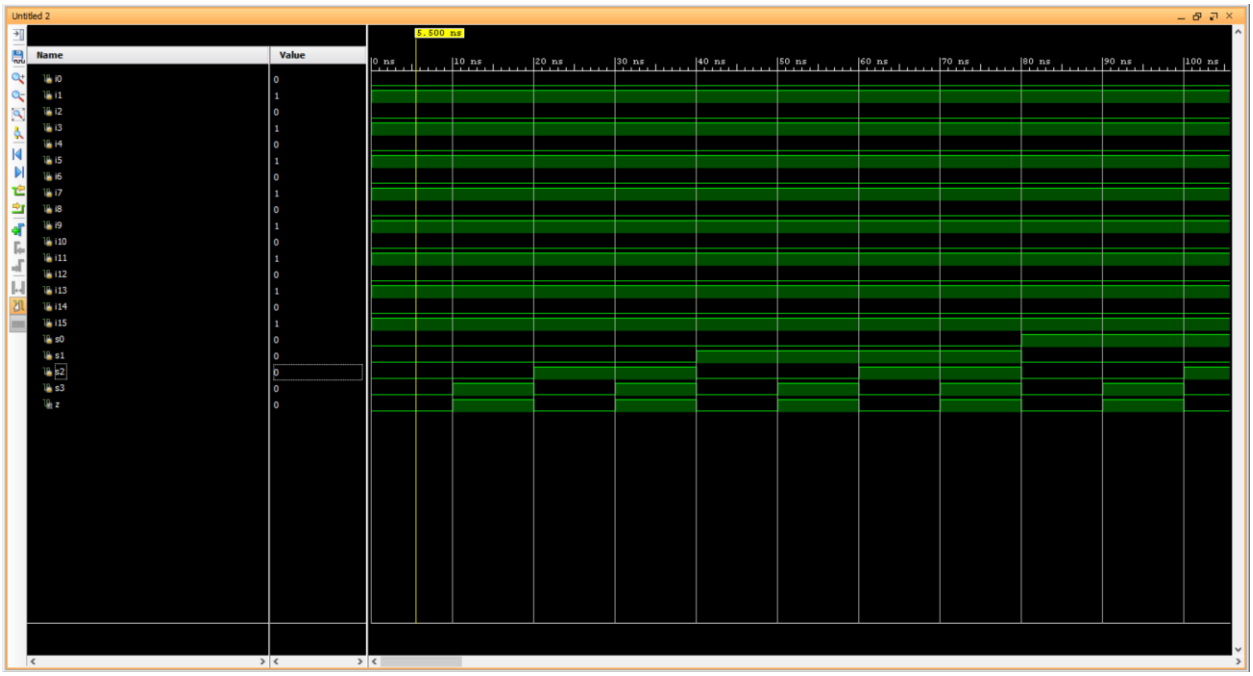


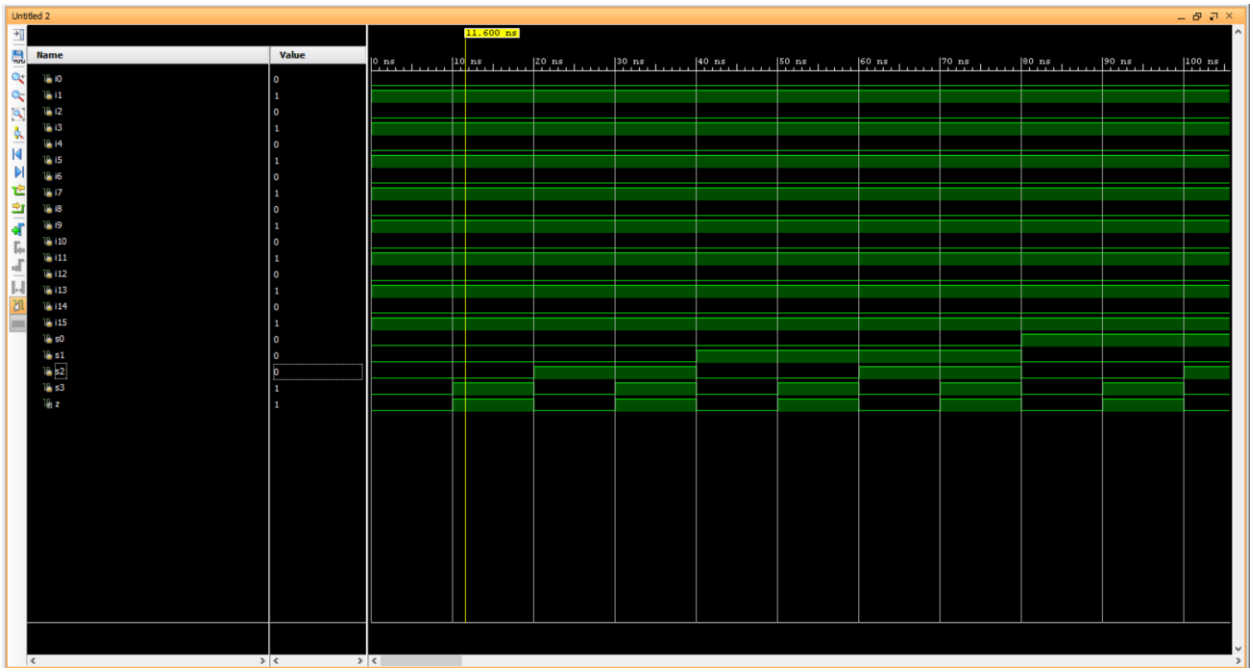



Untitled 1
Name
Value
s0
s1
z
i0
i1
i2
i3
1
0
1
0
0
1
0
28.450 ns
0 ns
5 ns
10 ns
15 ns
20 ns
25 ns
30 ns
35 ns
40 ns
45 ns
50 ns

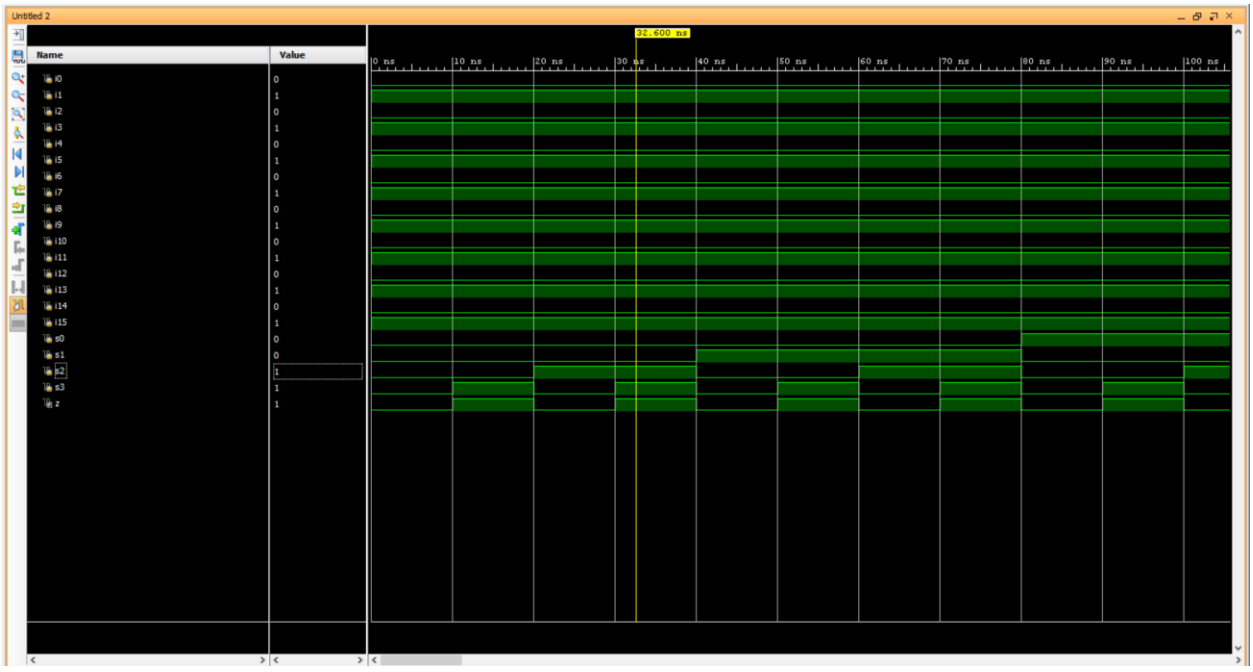

Untitled 1
Name
Value
s0
s1
z
i0
i1
i2
i3
1
1
1
0
0
0
1
34.950 ns
0 ns
5 ns
10 ns
15 ns
20 ns
25 ns
30 ns
35 ns
40 ns
45 ns
50 ns

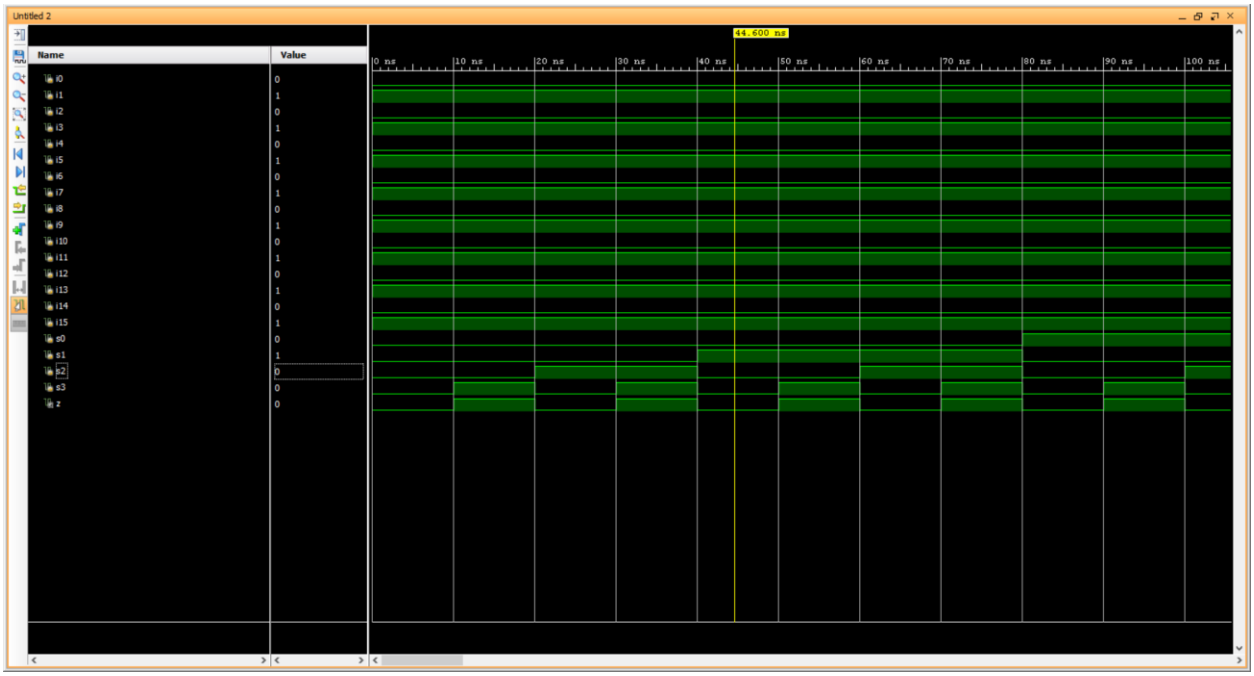
Mux 16:1 outputs:



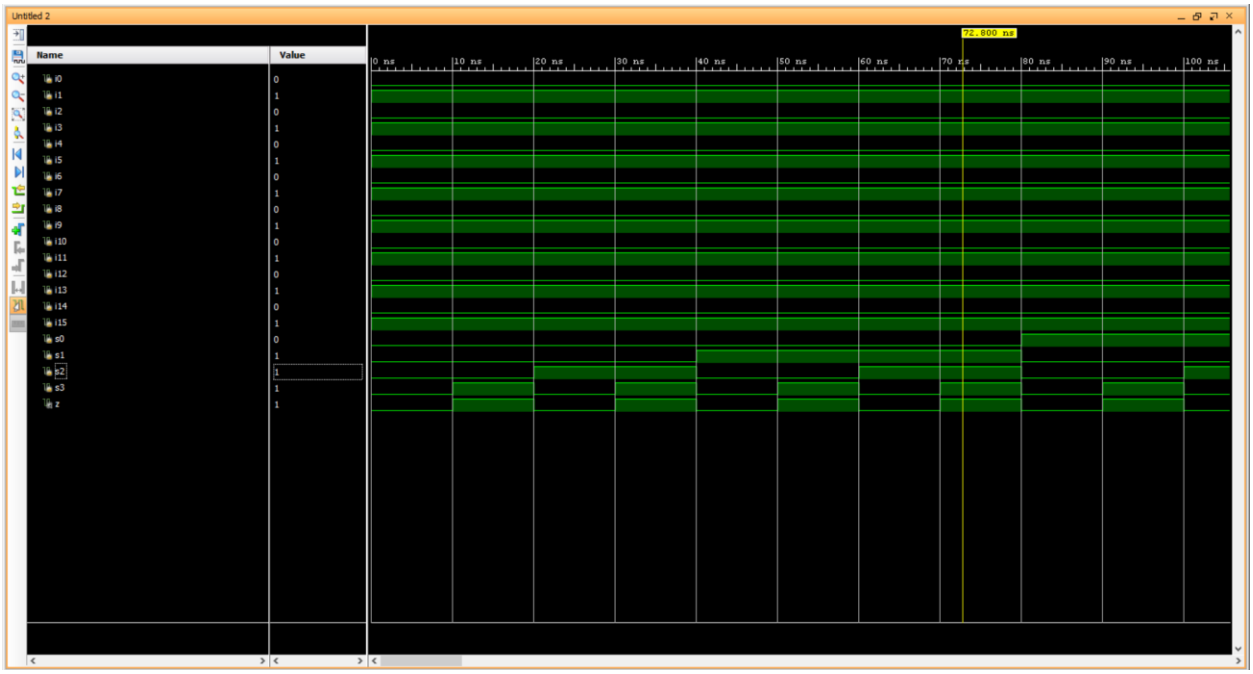


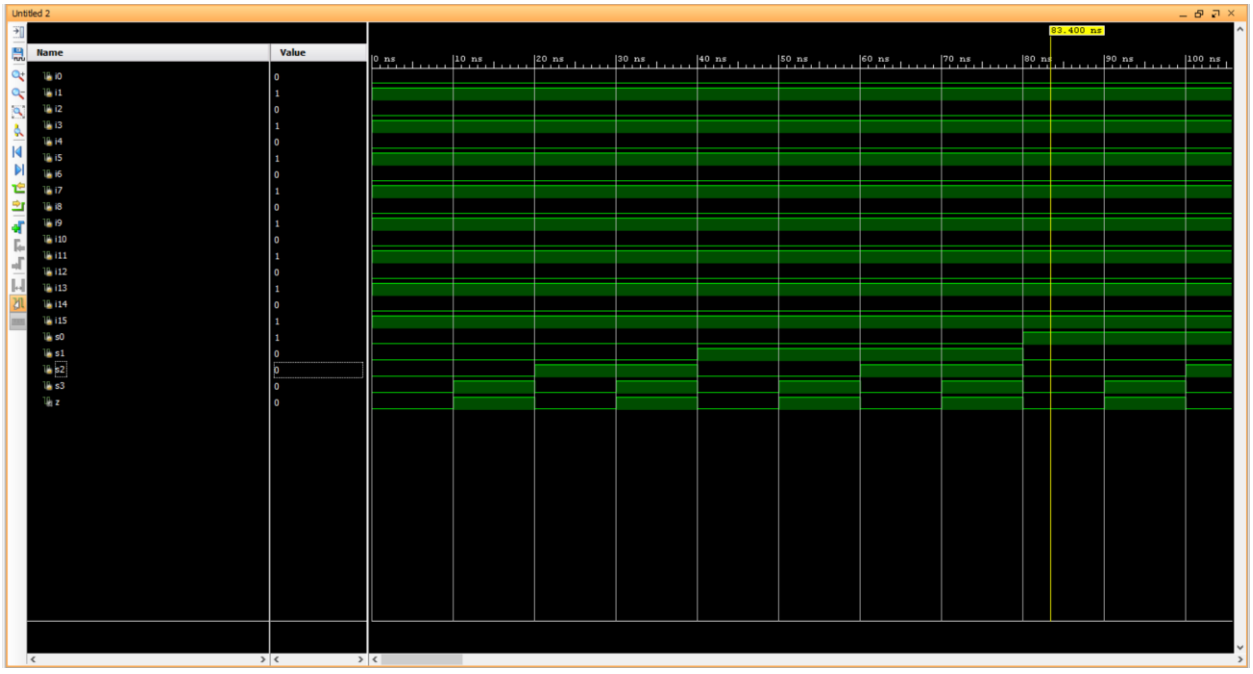

Untitled 2
Name
Value
i0 0
i1 1
i2 0
i3 1
i4 0
i5 1
i6 0
i7 1
i8 0
i9 1
i10 0
i11 1
i12 0
i13 1
i14 0
i15 1
s0 0
s1 0
s2 1
s3 0
z 0
26.400 ns
0 ns
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
70 ns
80 ns
90 ns
100 ns

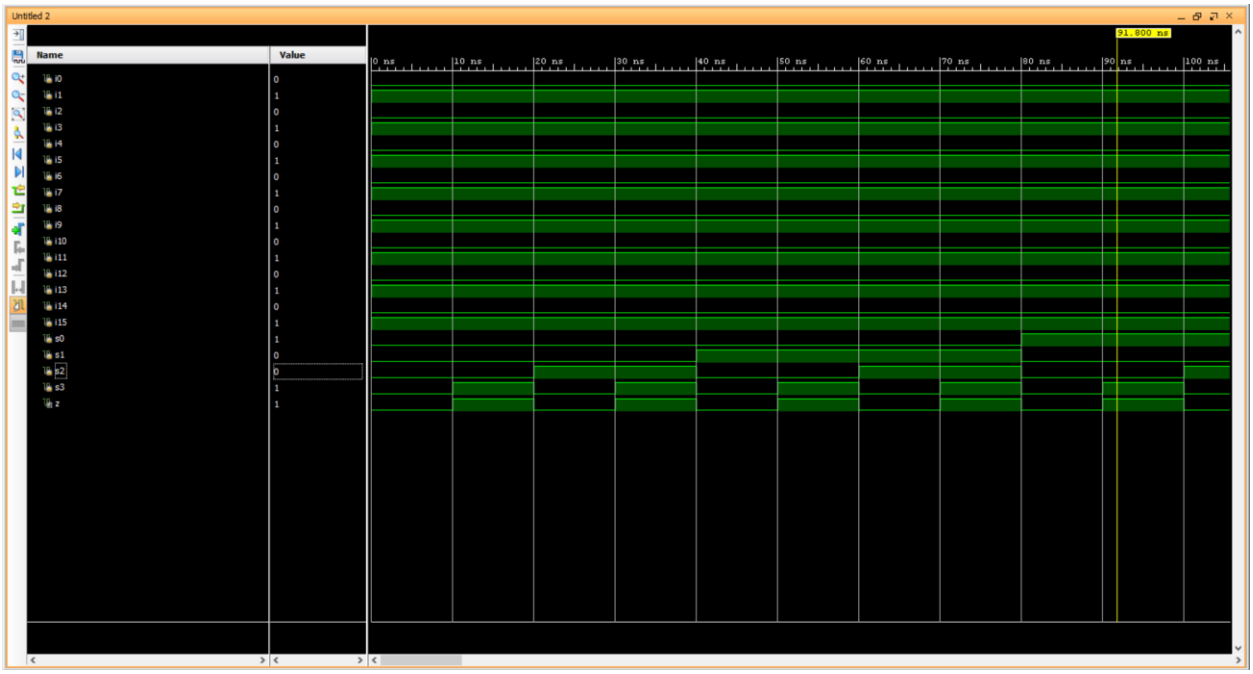

Untitled 2
Name
Value
i0 0
i1 1
i2 0
i3 1
i4 0
i5 1
i6 0
i7 1
i8 0
i9 1
i10 0
i11 1
i12 0
i13 1
i14 0
i15 1
s0 0
s1 0
s2 1
s3 1
z 1
32.600 ns
0 ns
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
70 ns
80 ns
90 ns
100 ns

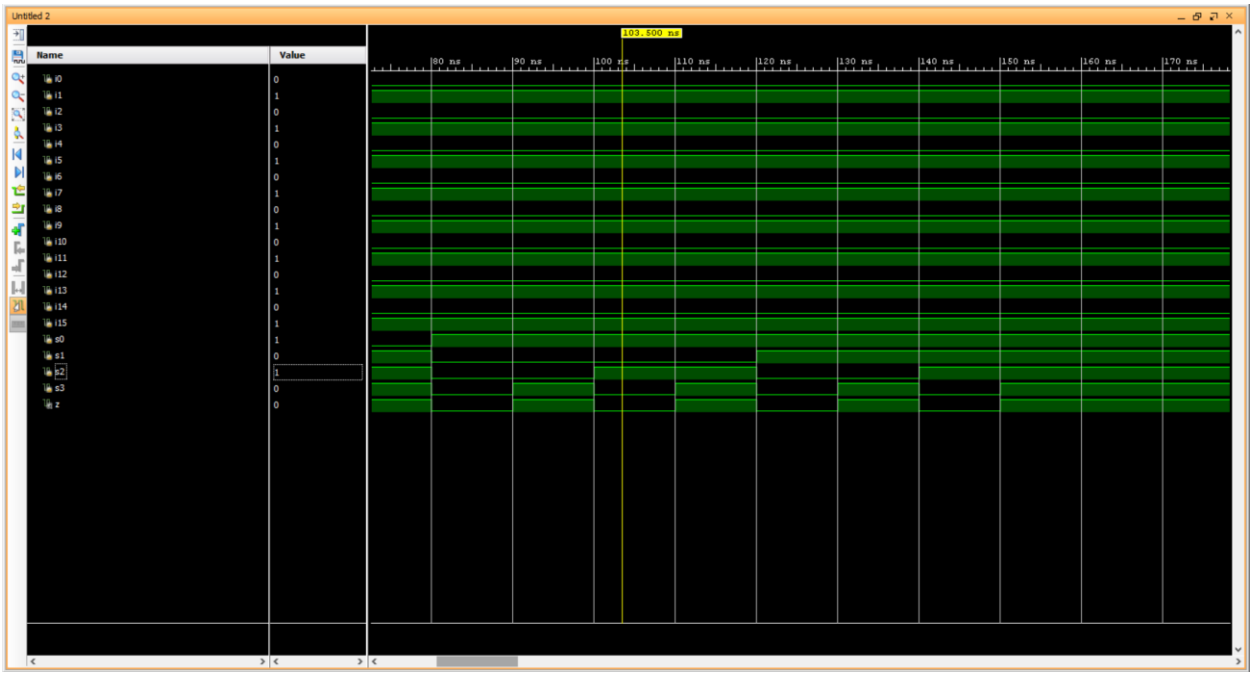

Untitled 2
Name
Value
i0 0
i1 1
i2 0
i3 1
i4 0
i5 1
i6 0
i7 1
i8 0
i9 1
i10 0
i11 1
i12 0
i13 1
i14 0
i15 1
s0 0
s1 1
s2 0
s3 0
z 0
44.600 ns
0 ns
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
70 ns
80 ns
90 ns
100 ns

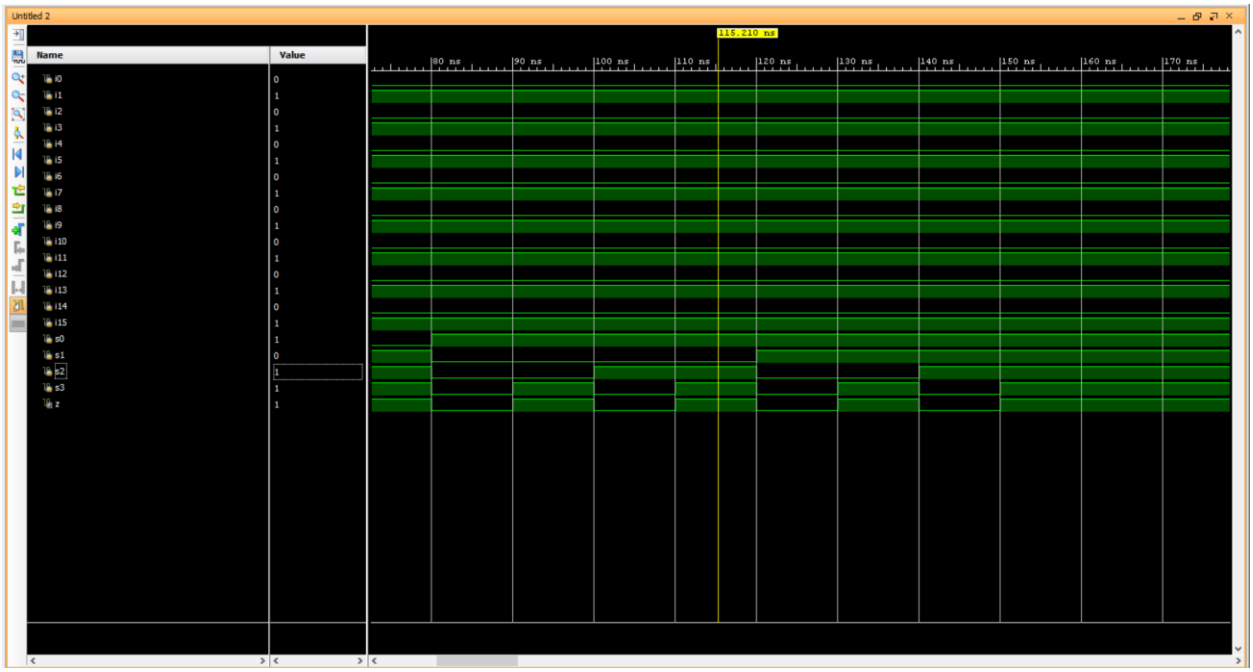

Untitled 2
Name
Value
i0 0
i1 1
i2 0
i3 1
i4 0
i5 1
i6 0
i7 1
i8 0
i9 1
i10 0
i11 1
i12 0
i13 1
i14 0
i15 1
s0 0
s1 1
s2 0
s3 1
z 1
54.800 ns
0 ns
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
70 ns
80 ns
90 ns
100 ns

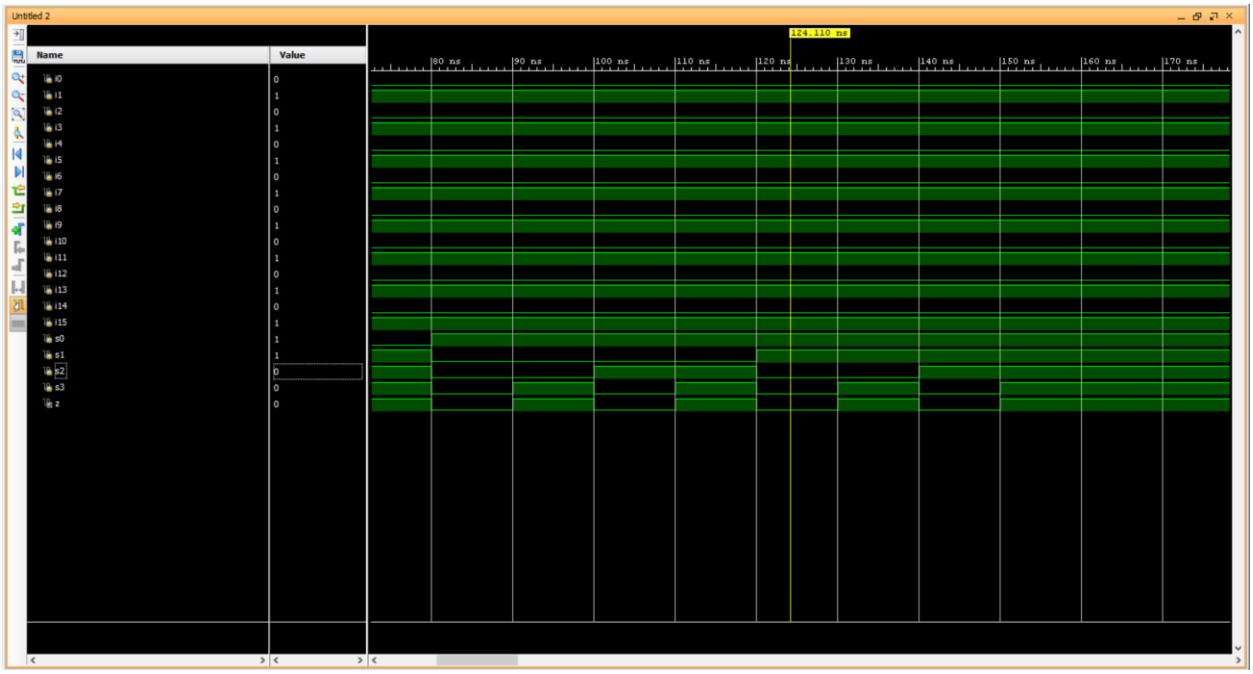

Untitled 2
Name
Value
i0 0
i1 1
i2 0
i3 1
i4 0
i5 1
i6 0
i7 1
i8 0
i9 1
i10 0
i11 1
i12 0
i13 1
i14 0
i15 1
s0 0
s1 1
s2 1
s3 1
z 1
72.800 ns
0 ns
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
70 ns
80 ns
90 ns
100 ns

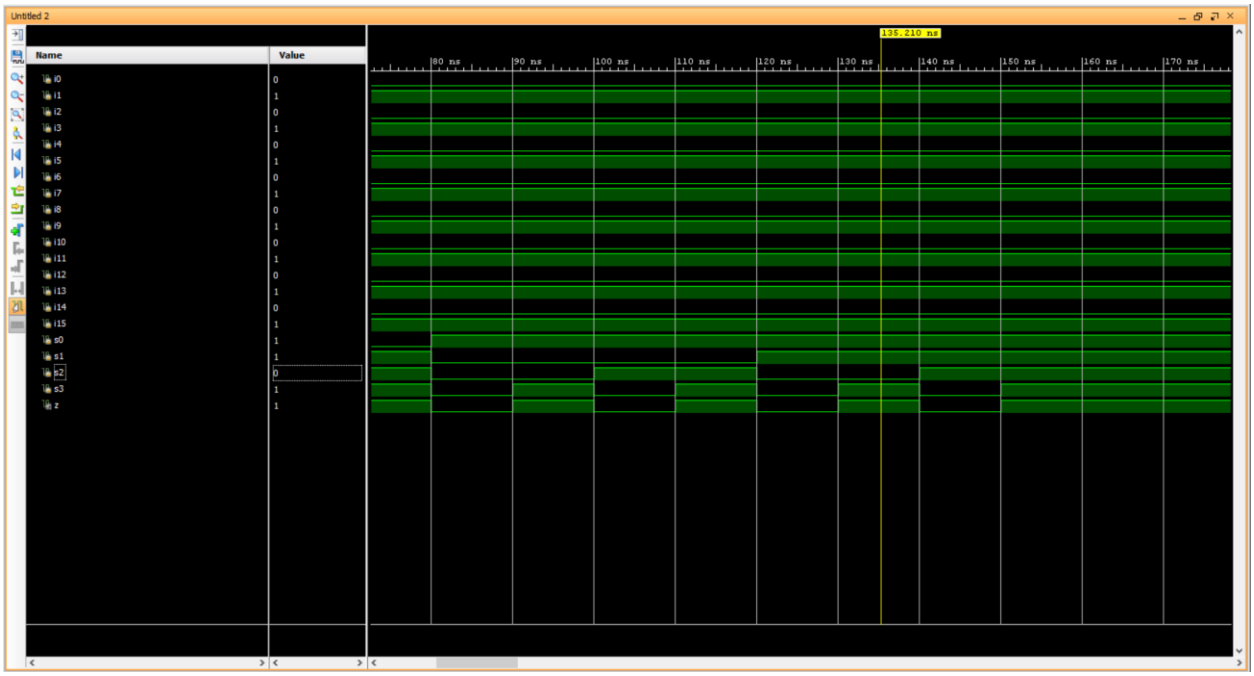

Untitled 2
Name
Value
i0 0
i1 1
i2 0
i3 1
i4 0
i5 1
i6 0
i7 1
i8 0
i9 1
i10 0
i11 1
i12 0
i13 1
i14 0
i15 1
s0 1
s1 0
s2 0
s3 0
z 0
83.400 ns
0 ns
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
70 ns
80 ns
90 ns
100 ns


Untitled 2
Name
Value
i0 0
i1 1
i2 0
i3 1
i4 0
i5 1
i6 0
i7 1
i8 0
i9 1
i10 0
i11 1
i12 0
i13 1
i14 0
i15 1
s0 1
s1 0
s2 0
s3 1
z 1
91.800 ns
0 ns
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
70 ns
80 ns
90 ns
100 ns

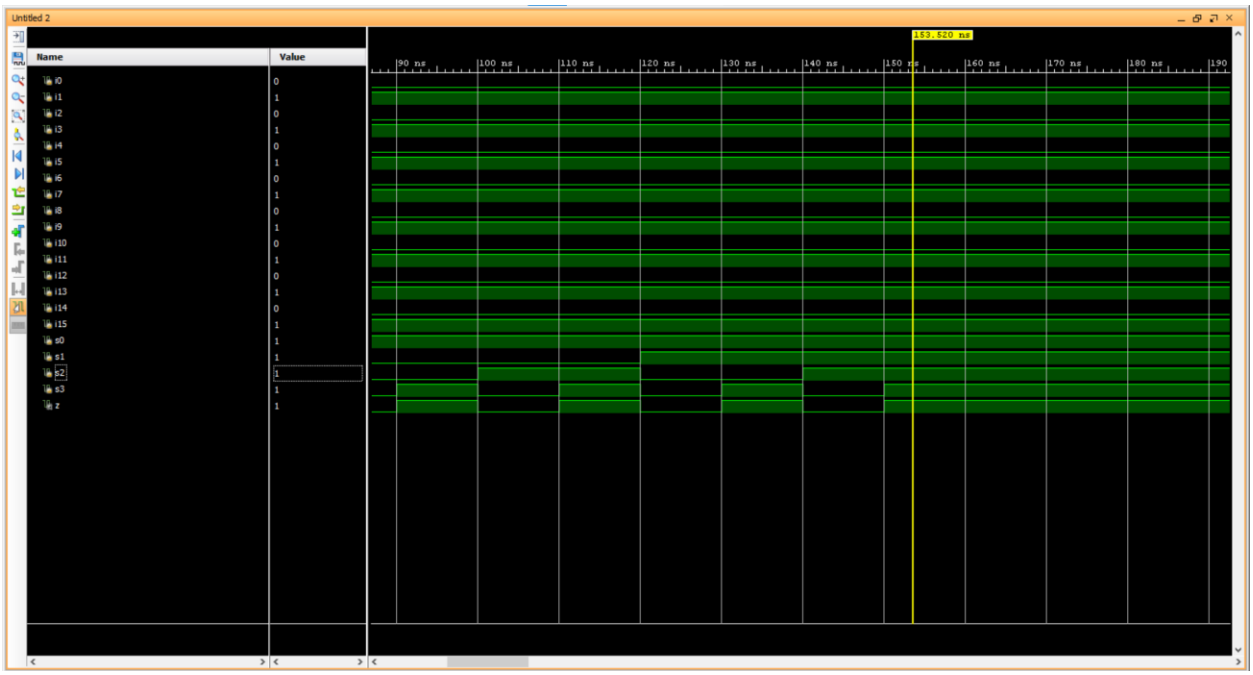

Untitled 2
Name
Value
i0 0
i1 1
i2 0
i3 1
i4 0
i5 1
i6 0
i7 1
i8 0
i9 1
i10 0
i11 1
i12 0
i13 1
i14 0
i15 1
s0 1
s1 0
s2 1
s3 0
z 0
103.600 ns
80 ns
90 ns
100 ns
110 ns
120 ns
130 ns
140 ns
150 ns
160 ns
170 ns


Untitled 2
Name
Value
i0 0
i1 1
i2 0
i3 1
i4 0
i5 1
i6 0
i7 1
i8 0
i9 1
i10 0
i11 1
i12 0
i13 1
i14 0
i15 1
s0 1
s1 0
s2 1
s3 1
z 1
115.210 ns
80 ns
90 ns
100 ns
110 ns
120 ns
130 ns
140 ns
150 ns
160 ns
170 ns


Untitled 2
Name
Value
i0 0
i1 1
i2 0
i3 1
i4 0
i5 1
i6 0
i7 1
i8 0
i9 1
i10 0
i11 1
i12 0
i13 1
i14 0
i15 1
s0 1
s1 1
s2 0
s3 0
z 0
124.110 ns
80 ns
90 ns
100 ns
110 ns
120 ns
130 ns
140 ns
150 ns
160 ns
170 ns


Untitled 2
Name
Value
i0 0
i1 1
i2 0
i3 1
i4 0
i5 1
i6 0
i7 1
i8 0
i9 1
i10 0
i11 1
i12 0
i13 1
i14 0
i15 1
s0 1
s1 1
s2 0
s3 1
z 1
135.210 ns
80 ns
90 ns
100 ns
110 ns
120 ns
130 ns
140 ns
150 ns
160 ns
170 ns


Untitled 2
Name
Value
i0 0
i1 1
i2 0
i3 1
i4 0
i5 1
i6 0
i7 1
i8 0
i9 1
i10 0
i11 1
i12 0
i13 1
i14 0
i15 1
s0 1
s1 1
s2 1
s3 0
z 0
143.520 ns
90 ns
100 ns
110 ns
120 ns
130 ns
140 ns
150 ns
160 ns
170 ns
180 ns
190



Conclusion:

Now I got full knowledge about the mutliplexer and demultiplexer