EX. NO: 02	MULTIPLEXERS AND DEMULTIPLEXERS
DATE :	

AIM: To simulate Multiplexers and Demultiplexers in behavior, structural, and dataflow model of Verilog HDL.

SOFTWARE USED: Xilinx Vivado

HARDWARE USED: Basys3 FPGA Board

MULTIPLEXERS:

VERILOG HDL CODES:

4:1 Multiplexer in behavioral model:

```
module four_mux_behav(data_in,sel,data_out);
   input [3:0] data_in;
   input [1:0] sel;
   output     data_out;
   reg data_out;
   always@(*)
   begin
    if (sel == 2'b00)
    begin data_out=data_in[0]; end
     else if (sel == 2'b01)
    begin data_out=data_in[1]; end
     else if (sel == 2'b10)
    begin data_out= data_in[2];end
     else if (sel == 2'b11)
    begin data_out=data_in[3]; end
    end
endmodule
```

4:1 Multiplexer in structural model:

```
module  four_mux_struct( data_in,sel,data_out);
   input [0:3] data_in;
   input [1:0] sel;
   output     data_out;
   wire not_sel0, not_sel1;
   wire x, y, z,w;
   not (not_sel0, sel[0]);
   not (not_sel1, sel[1]);
   and(x, data_in[0], not_sel1, not_sel0);
   and (y, data_in[1], not_sel1, sel[0]);
   and (z, data_in[2], sel[1], not_sel0);
   and (w,data_in[3], sel[1], sel[0]);
   or (data_out, x, y, z,w);
endmodule
```

<u>4:1 Multiplexer in dataflow model</u>:

```
module four_mux_data (data_in,sel,data_out);
   input [3:0] data_in;
   input [1:0] sel;
   output     data_out;
assign data_out = ((sel == 2'b00)& data_in[0] |(sel == 2'b01)& data_in[1] |(sel == 2'b10)&
data_in[2] |(sel == 2'b11)&data_in[3]);
endmodule
```

<u>Testbench of 4:1 Multiplexer</u>:

```
module tb_mux4to1;
   reg [3:0] data_in;
   reg [1:0] sel;
   wire data_out;
  four_mux_data FM1(.data_in(data_in), .sel(sel), .data_out(data_out));
  //four_mux_behav FM2(.data_in(data_in), .sel(sel), .data_out(data_out));
 // four_mux_struct FM3(.data_in(data_in), .sel(sel), .data_out(data_out));
   initial begin
      data_in = 4'b1010;
      sel = 2'b00;
      #10 sel = 2'b01;
      #10 sel = 2'b10;
      #10 sel = 2'b11;
      #10;
      data_in = 4'b1110;
      sel = 2'b00;
      #10 sel = 2'b01;
      #10 sel = 2'b10;
      #10 sel = 2'b11;
      #10;
      data_in = 4'b1100;
      sel = 2'b00;
      #10 sel = 2'b01;
      #10 sel = 2'b10;
      #10 sel = 2'b11;
      #10 $stop;
      $monitor(" sel = %b, data_in = %b, data_out = %b", sel, data_in,  data_out);
   end

endmodule
```

8:1 Multiplexer: (behavioral model)

```
module eight_mux_behav(data_in,sel,data_out);
    input [7:0] data_in;
    input [2:0] sel;
    output data_out;
    reg data_out;
    always@(*)
    begin
     if (sel == 3'b000)
     begin data_out=data_in[0]; end
      else if (sel == 3'b001)
     begin data_out=data_in[1]; end
      else if (sel == 3'b010)
     begin data_out= data_in[2];end
      else if (sel == 3'b111)
     begin data_out=data_in[3]; end
      if (sel == 3'b100)
     begin data_out=data_in[4]; end
      else if (sel == 3'b101)
     begin data_out=data_in[5]; end
      else if (sel == 3'b110)
     begin data_out= data_in[6];end
      else if (sel == 3'b111)
     begin data_out=data_in[7]; end
     end
endmodule
```

Testbench of 8:1 Multiplexer:

```
module tb_mux8to1;
    reg [7:0] data_in;
    reg [2:0] sel;
    wire data_out;
   // eight_mux_data FM1(.data_in(data_in), .sel(sel), .data_out(data_out));
    eight_mux_behav FM2(.data_in(data_in), .sel(sel), .data_out(data_out));
   // eight_mux_struct FM3(.data_in(data_in), .sel(sel), .data_out(data_out));
    initial begin
        data_in = 8'b11001010;
        sel = 3'b000;
        #10 sel = 3'b001;
        #10 sel = 3'b010;
        #10 sel = 3'b011;
        #10 sel = 3'b100;
        #10 sel = 3'b101;
        #10 sel = 3'b110;
        #10 sel = 3'b111;
        #10;

        data_in = 8'b00011100;
          sel = 3'b000;
        #10 sel = 3'b001;
        #10 sel = 3'b010;
        #10 sel = 3'b011;
        #10 sel = 3'b100;
        #10 sel = 3'b101;
        #10 sel = 3'b110;
        #10 sel = 3'b111;

        #10; $stop;
        $monitor(" sel = %b, data_in = %b, data_out = %b", sel, data_in,  data_out);
    end
endmodule
```

<u>16:1 Multiplexer using 4:1 Multiplexer</u>:

```
module  four_mux_structral( data_in,sel,data_out);
   input [3:0] data_in;
   input [1:0] sel;
   output      data_out;
   wire not_sel0, not_sel1;
   wire x, y, z,w;
   not (not_sel0, sel[0]);
   not (not_sel1, sel[1]);
   and(x, data_in[0], not_sel1, not_sel0);
   and (y, data_in[1], not_sel1, sel[0]);
   and (z, data_in[2], sel[1], not_sel0);
   and (w,data_in[3], sel[1], sel[0]);
   or (data_out, x, y, z,w);
endmodule

module sixteen_mux(data_in,sel,data_out);
   input [15:0] data_in;
   input [3:0] sel;
   output data_out;
   wire w,x,y,z;
   wire [3:0] s;
   assign s[0]=w;
   assign s[1]=x;
   assign s[2]=y;
   assign s[3]=z;

   four_mux_structral FM1( data_in[3:0],sel[1:0],w);
   four_mux_structral FM2( data_in[7:4],sel[1:0],x);
   four_mux_structral FM3( data_in[11:8],sel[1:0],y);
   four_mux_structral FM4( data_in[15:12],sel[1:0],z);
   four_mux_structral FM5( s,sel[3:2],data_out);
endmodule
```

Testbench of 16:1 Multiplexer:

```
module sixteen_mux_test;
   reg [15:0] data_in;
   reg [3:0] sel;
   wire data_out;
   sixteen_mux SM(.data_in(data_in), .sel(sel), .data_out(data_out));
   initial begin

      data_in = 16'b1100001010110101;
      sel = 4'b0000;
      #10 sel = 4'b0001;
      #10 sel = 4'b0010;
      #10 sel = 4'b0011;
      #10 sel = 4'b0100;
      #10 sel = 4'b0101;
      #10 sel = 4'b0110;
      #10 sel = 4'b0111;
      #10 sel = 4'b1000;
      #10 sel = 4'b1001;
      #10 sel = 4'b1010;
      #10 sel = 4'b1011;
      #10 sel = 4'b1100;
      #10 sel = 4'b1101;
      #10 sel = 4'b1110;
      #10 sel = 4'b1111;

     #10 data_in = 16'b1000011010101010;
       sel = 4'b0000;
      #10 sel = 4'b0001;
      #10 sel = 4'b0010;
      #10 sel = 4'b0011;
      #10 sel = 4'b0100;
      #10 sel = 4'b0101;
      #10 sel = 4'b0110;
      #10 sel = 4'b0111;
      #10 sel = 4'b1000;
      #10 sel = 4'b1001;
      #10 sel = 4'b1010;
      #10 sel = 4'b1011;
      #10 sel = 4'b1100;
      #10 sel = 4'b1101;
      #10 sel = 4'b1110;
      #10 sel = 4'b1111;
      #10; $stop;
      $monitor(" sel = %b, data_in = %b, data_out = %b", sel, data_in,  data_out);
   end

endmodule
```

32:1 Multiplexer using 8:1 Multiplexer and 4:1 Multiplexer:

```
`timescale 1ns / 1ps
module four_mux_data1 (data_in,sel,data_out);
    input [3:0] data_in;
    input [1:0] sel;
    output     data_out;
assign data_out = ((sel == 2'b00)& data_in[0] |(sel == 2'b01)& data_in[1] |(sel == 2'b10)&
data_in[2] |(sel == 2'b11)&data_in[3]);
endmodule

module eight_mux_behav1(data_in,sel,data_out);
    input [7:0] data_in;
    input [2:0] sel;
    output data_out;
    reg data_out;
    always@(*)
    begin
     if (sel == 3'b000)
     begin data_out=data_in[0]; end
      else if (sel == 3'b001)
     begin data_out=data_in[1]; end
      else if (sel == 3'b010)
     begin data_out= data_in[2];end
      else if (sel == 3'b111)
     begin data_out=data_in[3]; end
      if (sel == 3'b100)
     begin data_out=data_in[4]; end
      else if (sel == 3'b101)
     begin data_out=data_in[5]; end
      else if (sel == 3'b110)
     begin data_out= data_in[6];end
      else if (sel == 3'b111)
     begin data_out=data_in[7]; end
     end
endmodule

module mux32to1(data_in,sel,data_out);
 input [31:0] data_in;
 input [4:0] sel;
 output data_out;
 wire w,x,y,z;
    wire [3:0] s;
    assign s[0]=w;
    assign s[1]=x;
    assign s[2]=y;
    assign s[3]=z;

   eight_mux_behav1 FM1( data_in[7:0],sel[2:0],w);
   eight_mux_behav1 FM2( data_in[15:8],sel[2:0],x);
   eight_mux_behav1 FM3( data_in[23:16],sel[2:0],y);
   eight_mux_behav1 FM4( data_in[31:24],sel[2:0],z);
   four_mux_data1 FM5( s,sel[4:3],data_out);
endmodule
```

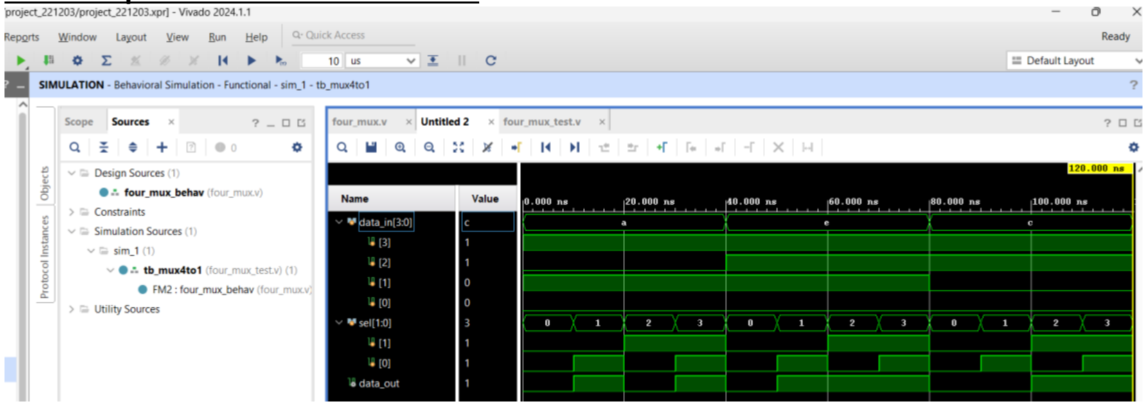
<u>Testbench of 32:1 Multiplexer</u>:

```
module mux32to1_test;
   reg [31:0] data_in;
   reg [4:0] sel;
   wire data_out;
   mux32to1 SM(.data_in(data_in), .sel(sel), .data_out(data_out));
   initial begin
      data_in = 32'b11000010101101011100001010110101;
      sel = 5'b00000;
      #10 sel = 5'b00001;
      #10 sel = 5'b00010;
      #10 sel = 5'b00011;
      #10 sel = 5'b00100;
      #10 sel = 5'b00101;
      #10 sel = 5'b00110;
      #10 sel = 5'b00111;
      #10 sel = 5'b01000;
      #10 sel = 5'b01001;
      #10 sel = 5'b01010;
      #10 sel = 5'b01011;
      #10 sel = 5'b01100;
      #10 sel = 5'b01101;
      #10 sel = 5'b01110;
      #10 sel = 5'b01111;
      #10 sel = 5'b10000;
      #10 sel = 5'b10001;
      #10 sel = 5'b10010;
      #10 sel = 5'b10011;
      #10 sel = 5'b10100;
      #10 sel = 5'b10101;
      #10 sel = 5'b10110;
      #10 sel = 5'b10111;
      #10 sel = 5'b11000;
      #10 sel = 5'b11001;
      #10 sel = 5'b11010;
      #10 sel = 5'b11011;
      #10 sel = 5'b11100;
      #10 sel = 5'b11101;
      #10 sel = 5'b11110;
      #10 sel = 5'b11111;
      #10

       data_in = 32'b11000010111000010101101010110101;
      sel = 5'b00000;
      #10 sel = 5'b00001;
      #10 sel = 5'b00010;
      #10 sel = 5'b00011;
      #10 sel = 5'b00100;
      #10 sel = 5'b00101;
      #10 sel = 5'b00110;
      #10 sel = 5'b00111;
      #10 sel = 5'b01000;
```

```
        #10 sel = 5'b01001;
        #10 sel = 5'b01010;
        #10 sel = 5'b01011;
        #10 sel = 5'b01100;
        #10 sel = 5'b01101;
        #10 sel = 5'b01110;
        #10 sel = 5'b01111;
        #10 sel = 5'b10000;
        #10 sel = 5'b10001;
        #10 sel = 5'b10010;
        #10 sel = 5'b10011;
        #10 sel = 5'b10100;
        #10 sel = 5'b10101;
        #10 sel = 5'b10110;
        #10 sel = 5'b10111;
        #10 sel = 5'b11000;
        #10 sel = 5'b11001;
        #10 sel = 5'b11010;
        #10 sel = 5'b11011;
        #10 sel = 5'b11100;
        #10 sel = 5'b11101;
        #10 sel = 5'b11110;
        #10 sel = 5'b11111;
        #10; $stop;
        $monitor(" sel = %b, data_in = %b, data_out = %b", sel, data_in,  data_out);
    end
endmodule
```

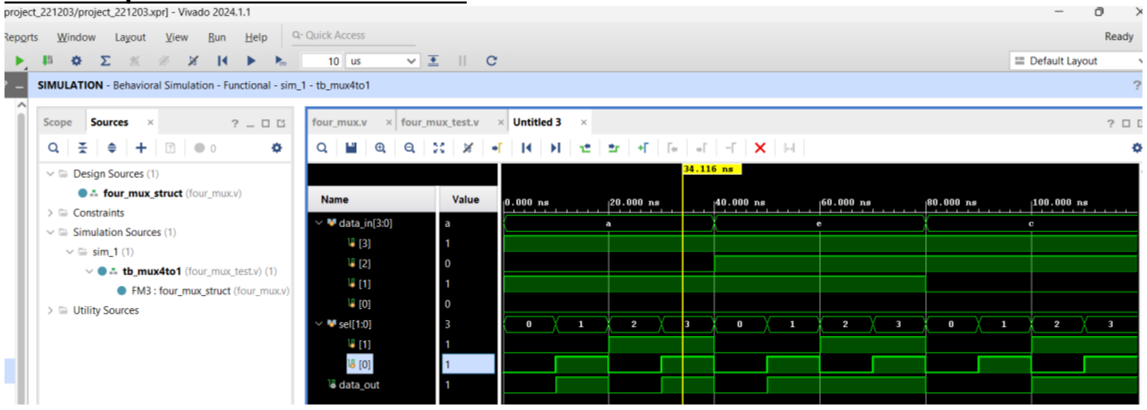
<u>SIMULATION WAVEFORMS</u>:

<u>4:1 Multiplexer in behavioral model</u>:



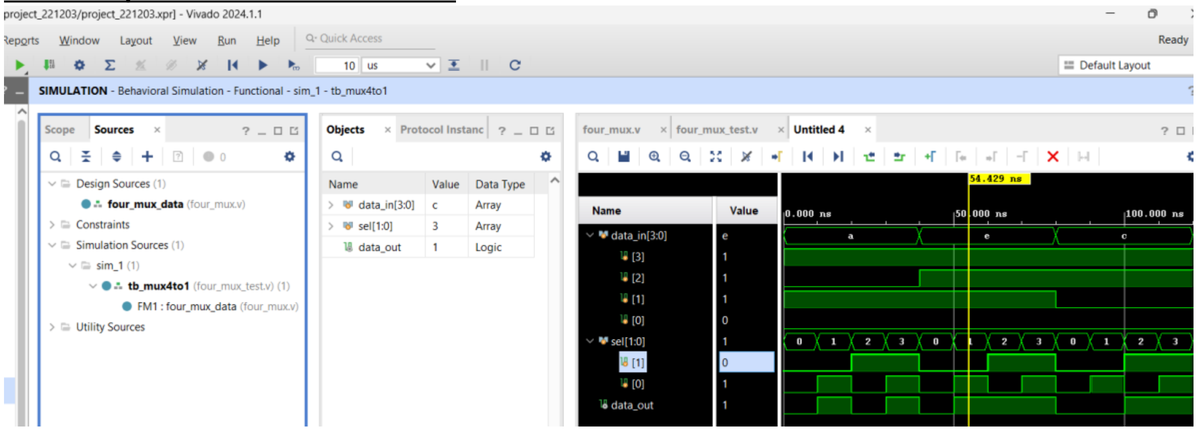
project_221203/project_221203.xpr] - Vivado 2024.1.1
Reports Window Layout View Run Help
Quick Access
Ready
Default Layout
SIMULATION - Behavioral Simulation - Functional - sim_1 - tb_mux4to1
Scope Sources
Design Sources (1)
four_mux_behav (four_mux.v)
Constraints
Simulation Sources (1)
sim_1 (1)
tb_mux4to1 (four_mux_test.v) (1)
FM2 : four_mux_behav (four_mux.v)
Utility Sources
four_mux.v Untitled 2 four_mux_test.v
Name Value
data_in[3:0] c
[3] 1
[2] 1
[1] 0
[0] 0
sel[1:0] 3
[1] 1
[0] 1
data_out 1
120.000 ns
0.000 ns 20.000 ns 40.000 ns 60.000 ns 80.000 ns 100.000 ns
a e c
0 1 2 3 0 1 2 3 0 1 2 3

<u>4:1 Multiplexer in structural model</u>:



project_221203/project_221203.xpr] - Vivado 2024.1.1
Reports Window Layout View Run Help
Quick Access
Ready
Default Layout
SIMULATION - Behavioral Simulation - Functional - sim_1 - tb_mux4to1
Scope Sources
Design Sources (1)
four_mux_struct (four_mux.v)
Constraints
Simulation Sources (1)
sim_1 (1)
tb_mux4to1 (four_mux_test.v) (1)
FM3 : four_mux_struct (four_mux.v)
Utility Sources
four_mux.v four_mux_test.v Untitled 3
Name Value
data_in[3:0] a
[3] 1
[2] 0
[1] 1
[0] 0
sel[1:0] 3
[1] 1
[0] 1
data_out 1
34.116 ns
0.000 ns 20.000 ns 40.000 ns 60.000 ns 80.000 ns 100.000 ns
a e c
0 1 2 3 0 1 2 3 0 1 2 3

4:1 Multiplexer in dataflow model:

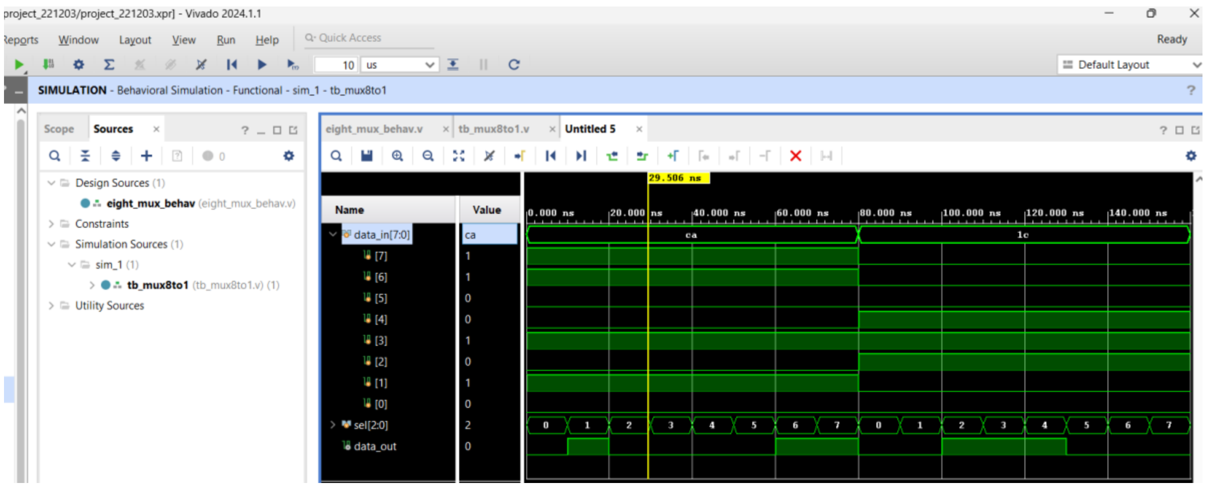

project_221203/project_221203.xpr] - Vivado 2024.1.1

SIMULATION - Behavioral Simulation - Functional - sim_1 - tb_mux4to1

Name	Value	Data Type
data_in[3:0]	c	Array
sel[1:0]	3	Array
data_out	1	Logic

Name	Value
data_in[3:0]	e
[3]	1
[2]	1
[1]	1
[0]	0
sel[1:0]	1
[1]	0
[0]	1
data_out	1

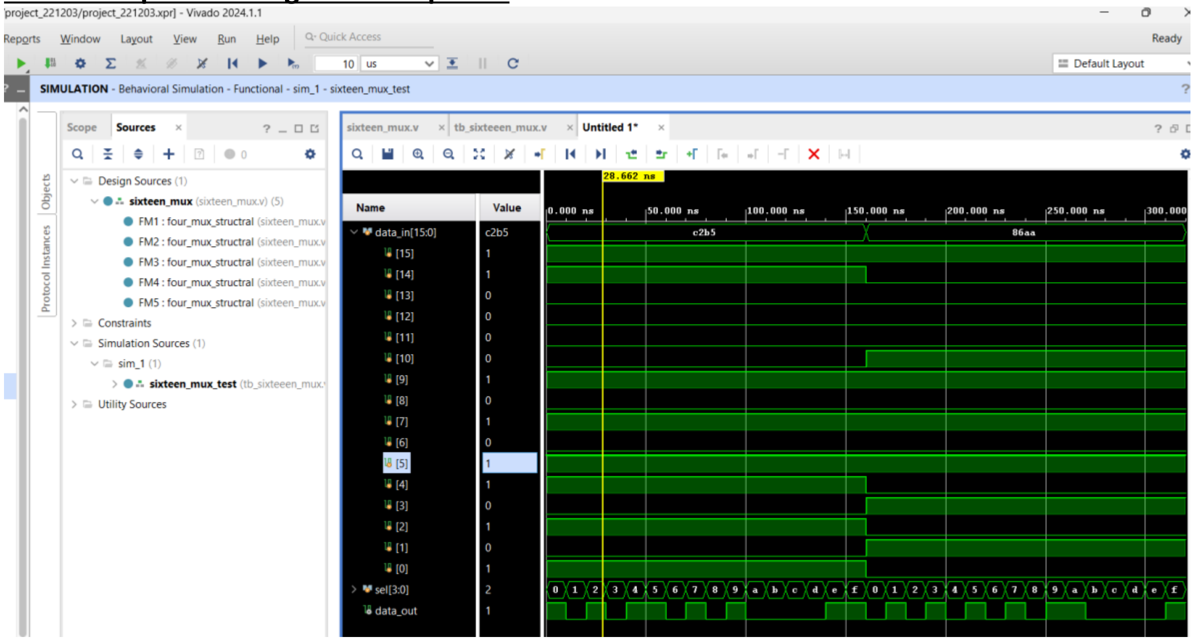
8:1 Multiplexer: (behavioral model)


project_221203/project_221203.xpr] - Vivado 2024.1.1

SIMULATION - Behavioral Simulation - Functional - sim_1 - tb_mux8to1

Name	Value
data_in[7:0]	ca
[7]	1
[6]	1
[5]	0
[4]	0
[3]	1
[2]	0
[1]	1
[0]	0
sel[2:0]	2
data_out	0

16:1 Multiplexer using 4:1 Multiplexer:

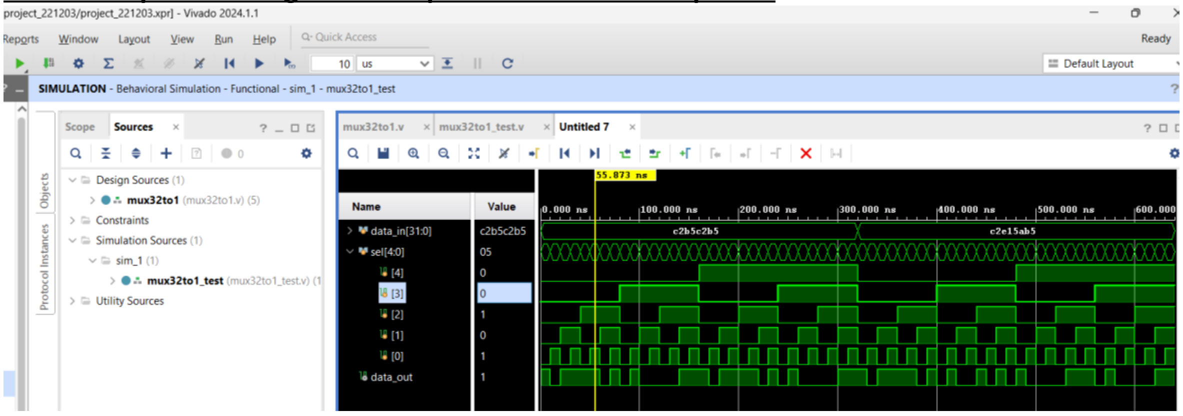

project_221203/project_221203.xpr] - Vivado 2024.1.1

SIMULATION - Behavioral Simulation - Functional - sim_1 - sixteen_mux_test

Reports Window Layout View Run Help
Quick Access
Ready
10 us
Default Layout
Scope Sources
Objects
Protocol Instances
Design Sources (1)
sixteen_mux (sixteen_mux.v) (5)
FM1 : four_mux_structral (sixteen_mux.v
FM2 : four_mux_structral (sixteen_mux.v
FM3 : four_mux_structral (sixteen_mux.v
FM4 : four_mux_structral (sixteen_mux.v
FM5 : four_mux_structral (sixteen_mux.v
Constraints
Simulation Sources (1)
sim_1 (1)
sixteen_mux_test (tb_sixteeen_mux.
Utility Sources
sixteen_mux.v tb_sixteeen_mux.v Untitled 1*
28.662 ns
0.000 ns
50.000 ns
100.000 ns
150.000 ns
200.000 ns
250.000 ns
300.000
c2b5 86aa
0 1 2 3 4 5 6 7 8 9 a b c d e f 0 1 2 3 4 5 6 7 8 9 a b c d e f

Name	Value
data_in[15:0]	c2b5
[15]	1
[14]	1
[13]	0
[12]	0
[11]	0
[10]	0
[9]	1
[8]	0
[7]	1
[6]	0
[5]	1
[4]	1
[3]	0
[2]	1
[1]	0
[0]	1
sel[3:0]	2
data_out	1

32:1 Multiplexer using 8:1 Multiplexer and 4:1 Multiplexer:



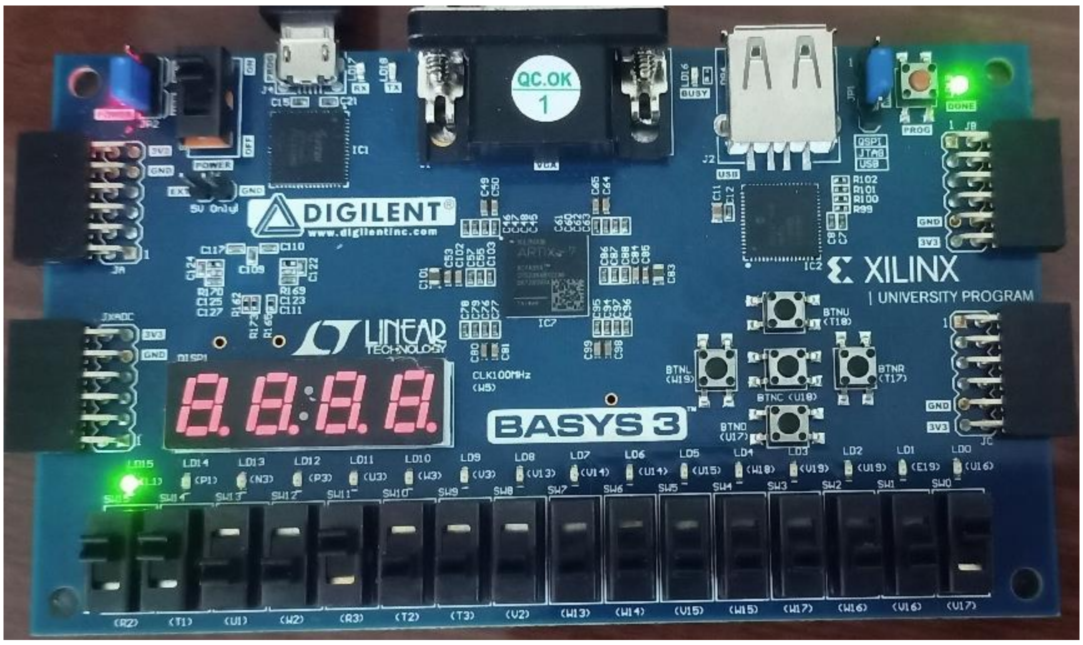
HADWARE OUTPUT:

4:1 Multiplexer:

I/O ports:

data_in[3]: R3	data_in[0]: V2	data_out: L1
data_in[2]: T2	sel[1]: R2	
data_in[1]: T3	sel[0]: T1	

For data_in[3]=1, data_in[2]=0, data_in[1]=0, data_in[0]=0
sel[1]=1,sel[0]=1

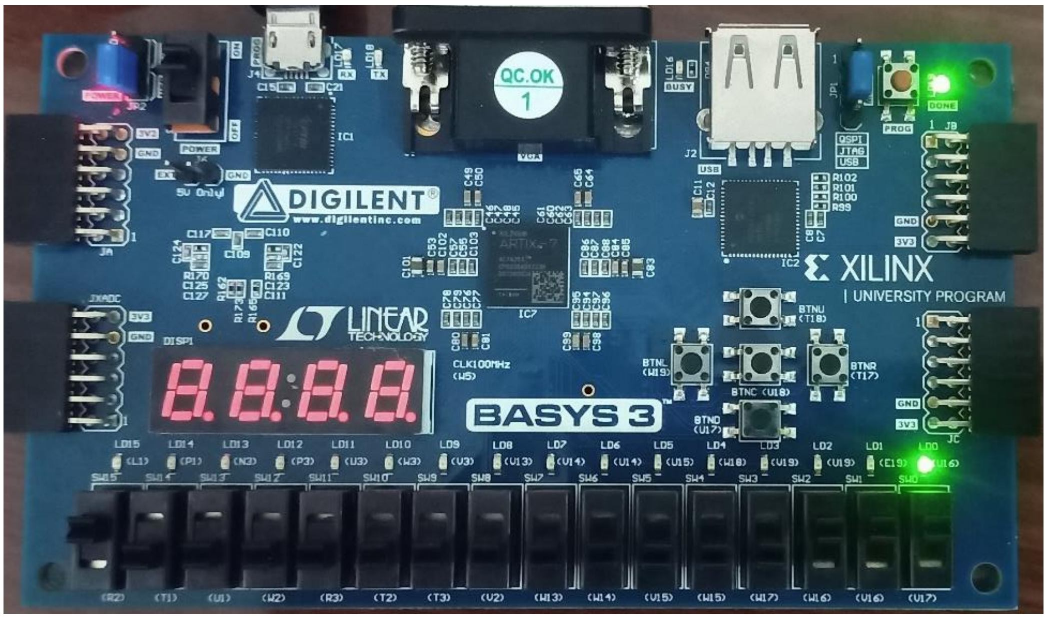


8:1 Multiplexer:

I/O ports:

data_in[7]: R2	data_in[3]: R3	sel[2]: W16
data_in[6]: T1	data_in[2]: T2	sel[1]: V16
data_in[5]: U1	data_in[1]: T3	sel[0]: V17
data_in[4]:W2	data_in[0]: V2	data_out: U16

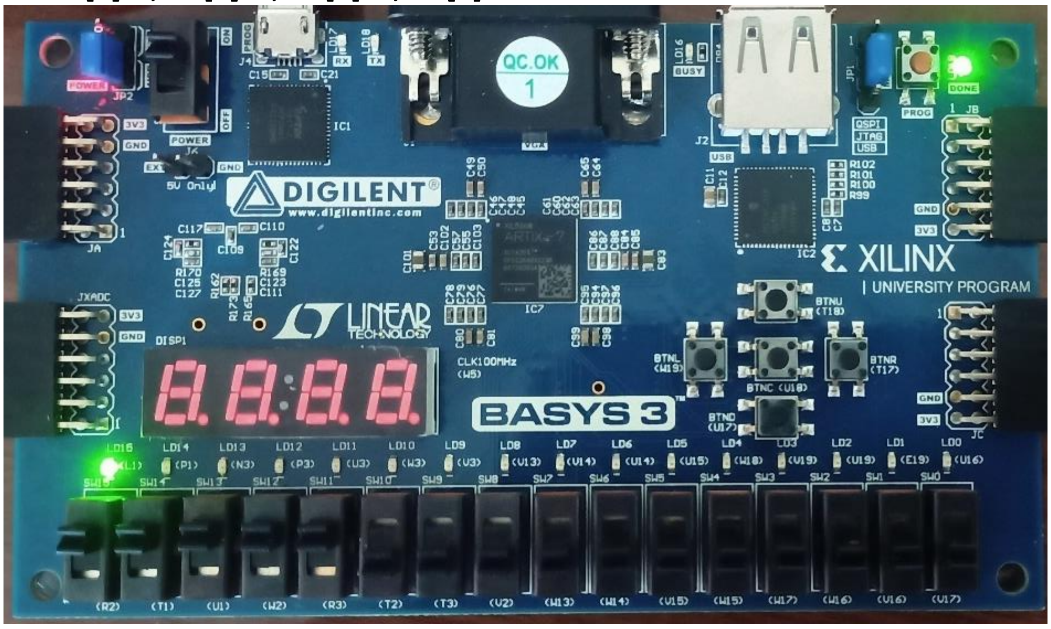
**For data_in[7]=1, data_in[6]=0, data_in[5]=0, data_in[4]=0
data_in[3]=0, data_in[2]=0, data_in[1]=0, data_in[0]=0
sel[2]=1, sel[1]=1, sel[0]=1**



<u>16:1 Multiplexer</u>:
<u>I/O ports</u>:

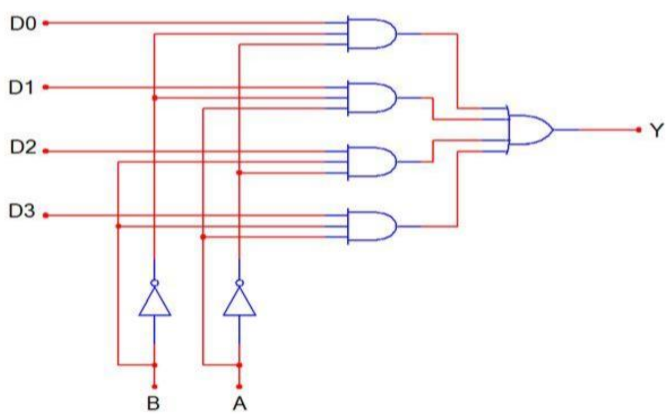
data_in[15]: R3	data_in[7]: W17	
data_in[14]: T2	data_in[6]: W16	sel[3]: R2
data_in[13]: T3	data_in[5]: V16	sel[2]: T1
data_in[12]: V2	data_in[4]: V17	sel[1]: U1
data_in[11]: W13	data_in[3]: W19	sel[0]: W2
data_in[10]: W14	data_in[2]: U18	data_out: L1
data_in[9]: V15	data_in[1]: T17	
data_in[8]: W15	data_in[0]: T18	

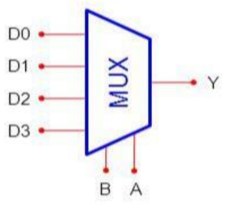
**For data_in[15]=1, data_in[14]=0, data_in[13]=0, data_in[12]=0
data_in[11]=0, data_in[10]=0, data_in[9]=0, data_in[8]=0
data_in[7]=0, data_in[6]=0, data_in[5]=0, data_in[4]=0
data_in[3]=0, data_in[2]=0, data_in[1]=0, data_in[0]=0
sel[3]=1, sel[2]=1, sel[1]=1, sel[0]=1**



CIRCUIT DIAGRAMS AND TRUTH TABLES:

4:1 Multiplexer:

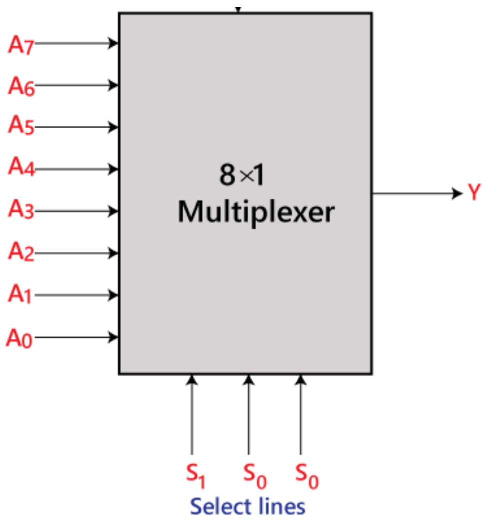




B	A	Y
0	0	D0
0	1	D1
1	0	D2
1	1	D3

Y=A’B’D0+AB’D1+A’BD2+ABD3

8:1 Multiplexer:



INPUTS			Output
S_2	S_1	S_0	Y
0	0	0	A_0
0	0	1	A_1
0	1	0	A_2
0	1	1	A_3
1	0	0	A_4
1	0	1	A_5
1	1	0	A_6
1	1	1	A_7

Y=A0S2’S1’S0’+A1S2’S1’S0+A2S2’S1S0’+A3S2’S1S0+A4S2S1’S0’+A5S2S1’S0+A6S2S1S0’+ A7S2S1S0

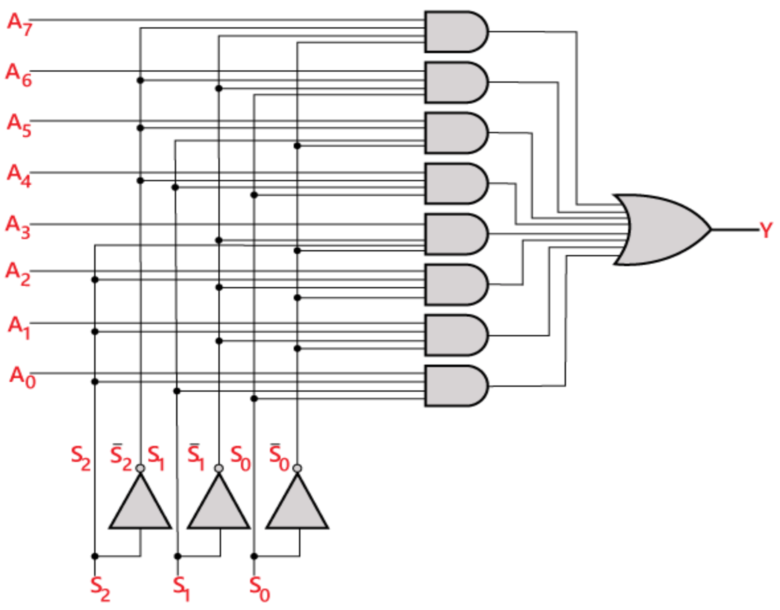
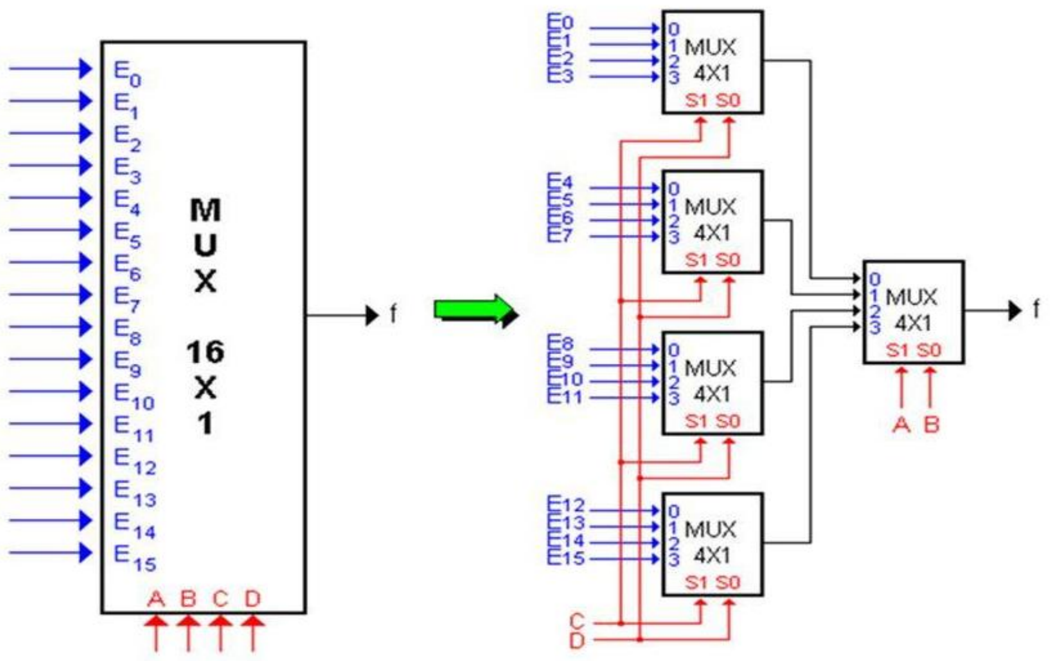
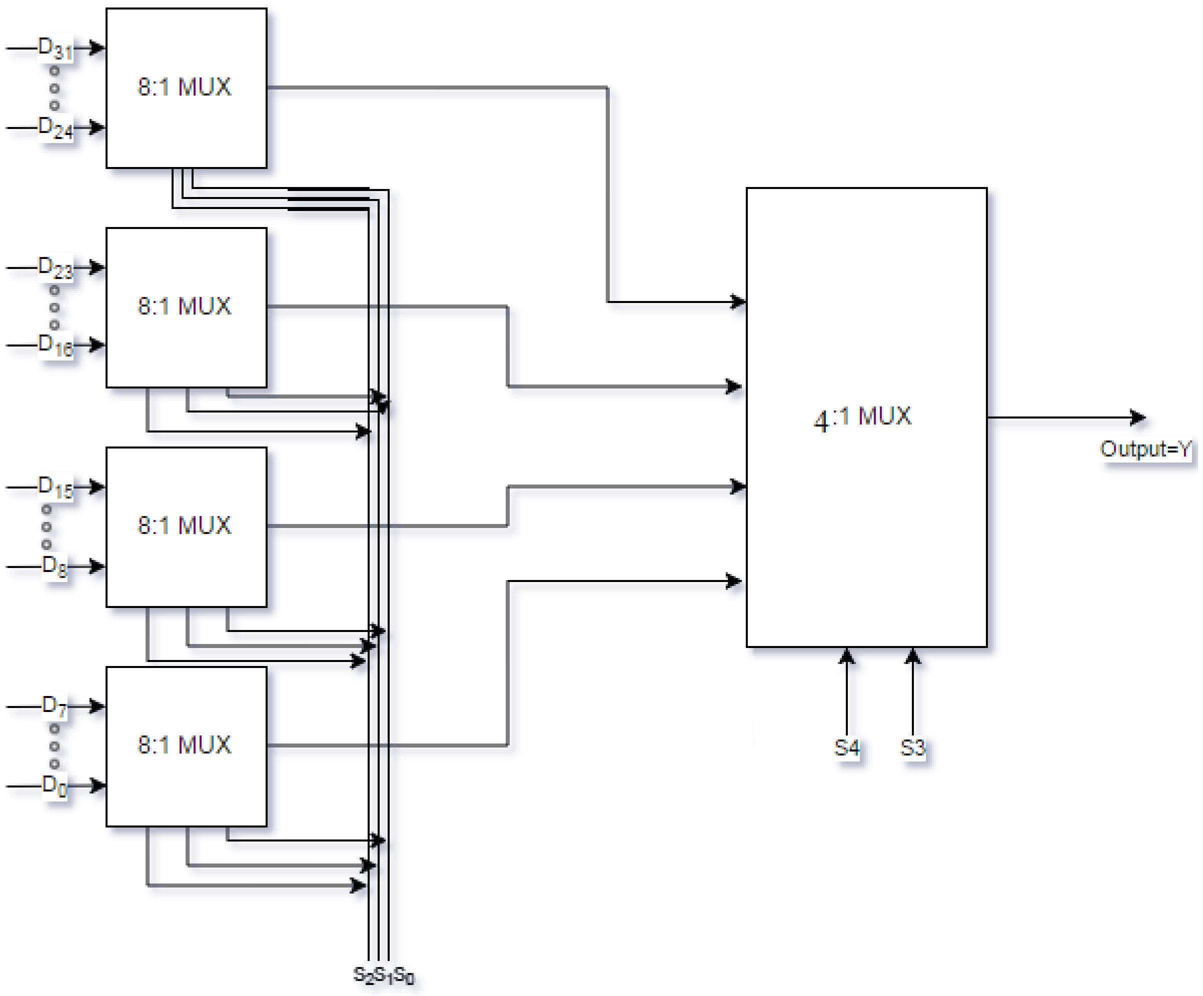


Fig. 8:1 multiplexer

16:1 Multiplexer using 4:1 Multiplexer:



32:1 Multiplexer:



DEMULTIPLEXERS:

VERILOG HDL CODES:

1:4 Demultiplexer in behavioral model:

```
module demux1to4_behav(data_in,sel,data_out);
input data_in;
input [1:0] sel;
output reg [3:0] data_out;
always@(*)
    begin
     if(sel==2'b00)
        begin data_out={3'b0,data_in}; end
    else if(sel==2'b01)
        begin data_out={2'b0,data_in, 1'b0}; end
     else if(sel==2'b10)
        begin data_out={1'b0,data_in,2'b00}; end
     else if(sel==2'b11)
        begin data_out={data_in,3'b000}; end
      else
        begin  data_out = 4'b0000; end
     end
endmodule
```

1:4 Demultiplexer in structural model:

```
module demux1to4_struct(data_in,sel,data_out);
input data_in;
input [1:0] sel;
output [3:0] data_out;
wire w1,w2;
not (w1, sel[0]);
not (w2, sel[1]);
and (data_out[0],w1,w2,data_in);
and (data_out[1],sel[0],w2,data_in);
and (data_out[2],w1,sel[1],data_in);
and (data_out[3],sel[0],sel[1],data_in);
endmodule
```

1:4 Demultiplexer in dataflow model:

```
module demux1to4_data(data_in,sel,data_out);
input data_in;
input [1:0] sel;
output [3:0] data_out;
assign data_out[0]=((sel==2'b00)& data_in);
assign data_out[1]=((sel==2'b01)& data_in);
assign data_out[2]=((sel==2'b10)& data_in);
assign data_out[3]=((sel==2'b11)& data_in);
endmodule
```

Testbench of 1:4 Demultiplexer:

```
module tb_demux;
   reg data_in;
   reg [1:0] sel;
   wire [3:0] data_out;
   //demux1to4_behav DM1 (.data_in(data_in), .sel(sel), .data_out(data_out));
   //demux1to4_data DM2 (.data_in(data_in), .sel(sel), .data_out(data_out));
   demux1to4_struct DM3 (.data_in(data_in), .sel(sel), .data_out(data_out));
   initial begin
      $monitor("sel = %b, data_in = %b, data_out = %b", sel, data_in, data_out);
      data_in = 1'b1;
       sel = 2'b00;
      #10 sel = 2'b01;
      #10 sel = 2'b10;
      #10 sel = 2'b11;
      #10 data_in = 1'b0;
      sel = 2'b00;
      #10 sel = 2'b01;
      #10 sel = 2'b10;
      #10 sel = 2'b11;
      #10 $stop;
   end
endmodule
```

1:8 Demultiplexer:

```
module demux1to8_data(data_in,sel,data_out);
input data_in;
input [2:0] sel;
output [7:0] data_out;
assign data_out[0]=((sel==3'b000)& data_in);
assign data_out[1]=((sel==3'b001)& data_in);
assign data_out[2]=((sel==3'b010)& data_in);
assign data_out[3]=((sel==3'b011)& data_in);
assign data_out[4]=((sel==3'b100)& data_in);
assign data_out[5]=((sel==3'b101)& data_in);
assign data_out[6]=((sel==3'b110)& data_in);
assign data_out[7]=((sel==3'b111)& data_in);
endmodule
```

Testbench of 1:8 Demultiplexer:

```
module tb_demux1to8;
   reg data_in;
   reg [2:0] sel;
   wire [7:0] data_out;
   demux1to8_data DM2 (.data_in(data_in), .sel(sel), .data_out(data_out));
   initial begin
      $monitor("sel = %b, data_in = %b, data_out = %b", sel, data_in, data_out);
      data_in = 1'b1;
         sel = 3'b000;
      #10 sel = 3'b001;
      #10 sel = 3'b010;
      #10 sel = 3'b011;
      #10 sel = 3'b100;
```

```
        #10 sel = 3'b101;
        #10 sel = 3'b110;
        #10 sel = 3'b111;

        #10 data_in = 1'b0;
        sel = 3'b000;
        #10 sel = 3'b001;
        #10 sel = 3'b010;
        #10 sel = 3'b011;
        #10 sel = 3'b100;
        #10 sel = 3'b101;
        #10 sel = 3'b110;
        #10 sel = 3'b111;
        #10 $stop;
    end
endmodule
```

1:16 Demultiplexer using 1:4 Demultiplexer:

```
module demux1to16(data_in,sel,data_out);
input data_in;
input [3:0] sel;
output [15:0] data_out;
wire [3:0]a;
demux1to4_struct DM1(data_in,sel[3:2],a[3:0]);
demux1to4_struct DM2(a[0],sel[1:0],data_out[3:0]);
demux1to4_struct DM3(a[1],sel[1:0],data_out[7:4]);
demux1to4_struct DM4(a[2],sel[1:0],data_out[11:8]);
demux1to4_struct DM5(a[3],sel[1:0],data_out[15:12]);
endmodule

module demux1to4_struct(data_in,sel,data_out);
input data_in;
input [1:0] sel;
output [3:0] data_out;
wire w1,w2;
not (w1, sel[0]);
not (w2, sel[1]);
and (data_out[0],w1,w2,data_in);
and (data_out[1],sel[0],w2,data_in);
and (data_out[2],w1,sel[1],data_in);
and (data_out[3],sel[0],sel[1],data_in);
endmodule
```

Testbench of 1:16 Demultiplexer:

```
module tb_demux1to16;
    reg data_in;
    reg [3:0] sel;
    wire [15:0] data_out;
    demux1to16 DM2 (.data_in(data_in), .sel(sel), .data_out(data_out));
    initial begin
        $monitor("sel = %b, data_in = %b, data_out = %b", sel, data_in, data_out);
        data_in = 1'b1;
            sel = 4'b0000;
        #10 sel = 4'b0001;
        #10 sel = 4'b0010;
        #10 sel = 4'b0011;
        #10 sel = 4'b0100;
        #10 sel = 4'b0101;
        #10 sel = 4'b0110;
        #10 sel = 4'b0111;
        #10 sel = 4'b1000;
        #10 sel = 4'b1001;
        #10 sel = 4'b1010;
        #10 sel = 4'b1011;
        #10 sel = 4'b1100;
        #10 sel = 4'b1101;
        #10 sel = 4'b1110;
        #10 sel = 4'b1111;
        data_in = 1'b0;
        #10 sel = 4'b0000;
        #10 sel = 4'b0001;
        #10 sel = 4'b0010;
        #10 sel = 4'b0011;
        #10 sel = 4'b0100;
        #10 sel = 4'b0101;
        #10 sel = 4'b0110;
        #10 sel = 4'b0111;
        #10 sel = 4'b1000;
        #10 sel = 4'b1001;
        #10 sel = 4'b1010;
        #10 sel = 4'b1011;
        #10 sel = 4'b1100;
        #10 sel = 4'b1101;
        #10 sel = 4'b1110;
        #10 sel = 4'b1111;
        #10 $stop;
    end
endmodule
```

1:32 Demultiplexer using 1:8 Demultiplexer and 1:4 Demultiplexer:

```
module demux1to32(data_in,sel,data_out);
input data_in;
input [4:0] sel;
output [31:0] data_out;
wire [3:0]a;
demux1to4 DM1(data_in,sel[4:3],a[3:0]);
demux1to8 DM2(a[0],sel[2:0],data_out[7:0]);
demux1to8 DM3(a[1],sel[2:0],data_out[15:8]);
demux1to8 DM4(a[2],sel[2:0],data_out[23:16]);
demux1to8 DM5(a[3],sel[2:0],data_out[31:24]);
endmodule

module demux1to8 (data_in,sel,data_out);
input data_in;
input [2:0] sel;
output [7:0] data_out;
```

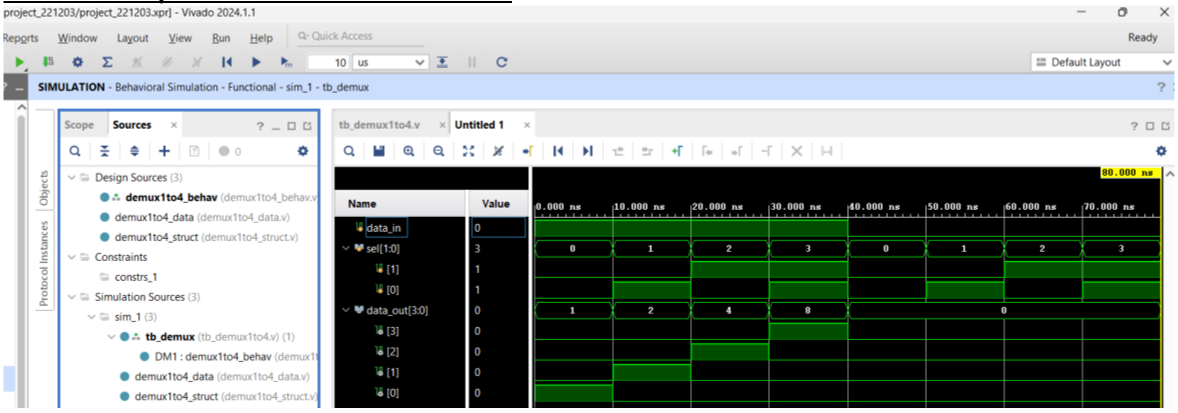
```
assign data_out[0]=((sel==3'b000)& data_in);
assign data_out[1]=((sel==3'b001)& data_in);
assign data_out[2]=((sel==3'b010)& data_in);
assign data_out[3]=((sel==3'b011)& data_in);
assign data_out[4]=((sel==3'b100)& data_in);
assign data_out[5]=((sel==3'b101)& data_in);
assign data_out[6]=((sel==3'b110)& data_in);
assign data_out[7]=((sel==3'b111)& data_in);
endmodule

module demux1to4(data_in,sel,data_out);
input data_in;
input [1:0] sel;
output [3:0] data_out;
wire w1,w2;
not (w1, sel[0]);
not (w2, sel[1]);
and (data_out[0],w1,w2,data_in);
and (data_out[1],sel[0],w2,data_in);
and (data_out[2],w1,sel[1],data_in);
and (data_out[3],sel[0],sel[1],data_in);
endmodule
```

<u>Testbench of 1:32 Demultiplexer</u>:

```
module tb_demux1to32;
   reg data_in;
   reg [4:0] sel;
   wire [31:0] data_out;
   demux1to32 DM2 (.data_in(data_in), .sel(sel), .data_out(data_out));
   initial begin
      $monitor("Time = %0t, sel = %b, data_in = %b, data_out = %b", $time, sel, data_in,
data_out);
      data_in = 1'b1;
      for (integer i = 0; i < 32; i = i + 1) begin
         sel = i;
         #10;
      end
      #10 $stop;
   end
endmodule
```

<u>SIMULATION WAVEFORM</u>:

<u>1:4 Demultiplexer in behavioral model</u>:



project_221203/project_221203.xpr] - Vivado 2024.1.1

SIMULATION - Behavioral Simulation - Functional - sim_1 - tb_demux

Design Sources (3)
- demux1to4_behav (demux1to4_behav.v
- demux1to4_data (demux1to4_data.v)
- demux1to4_struct (demux1to4_struct.v)

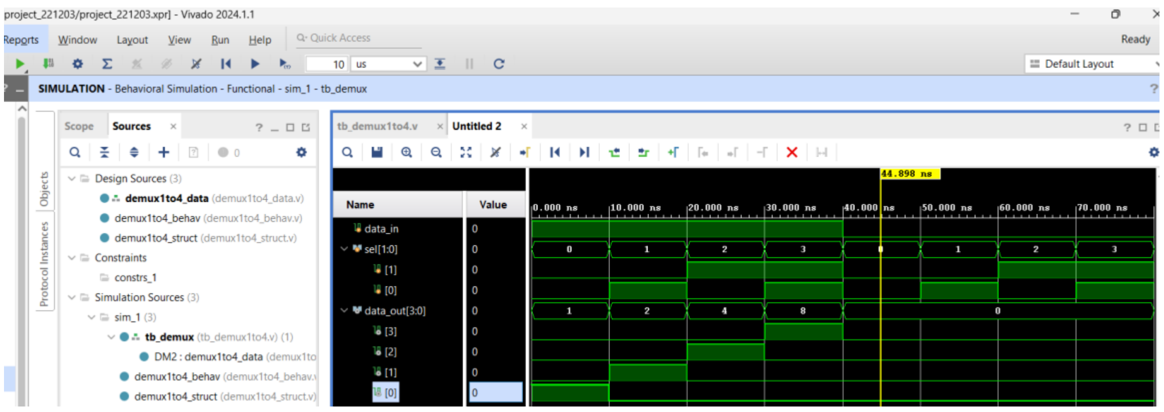
Constraints
- constrs_1

Simulation Sources (3)
- sim_1 (3)
  - tb_demux (tb_demux1to4.v) (1)
    - DM1 : demux1to4_behav (demux1t
  - demux1to4_data (demux1to4_data.v)
  - demux1to4_struct (demux1to4_struct.v)

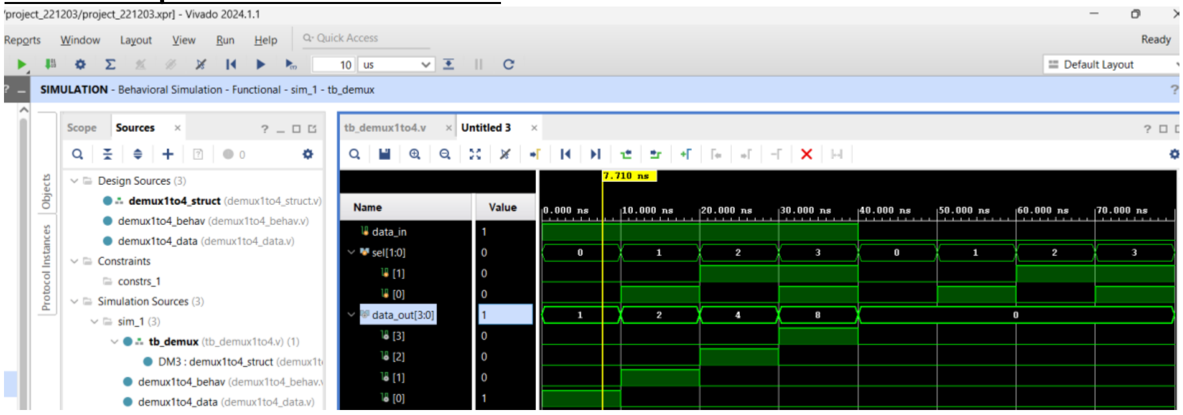
Name	Value
data_in	0
sel[1:0]	3
[1]	1
[0]	1
data_out[3:0]	0
[3]	0
[2]	0
[1]	0
[0]	0

Reports Window Layout View Run Help
Quick Access
Ready
10 us
Default Layout
Scope Sources
tb_demux1to4.v Untitled 1
Objects
Protocol Instances
80.000 ns
0.000 ns 10.000 ns 20.000 ns 30.000 ns 40.000 ns 50.000 ns 60.000 ns 70.000 ns
0 1 2 3 0 1 2 3
1 2 4 8 0

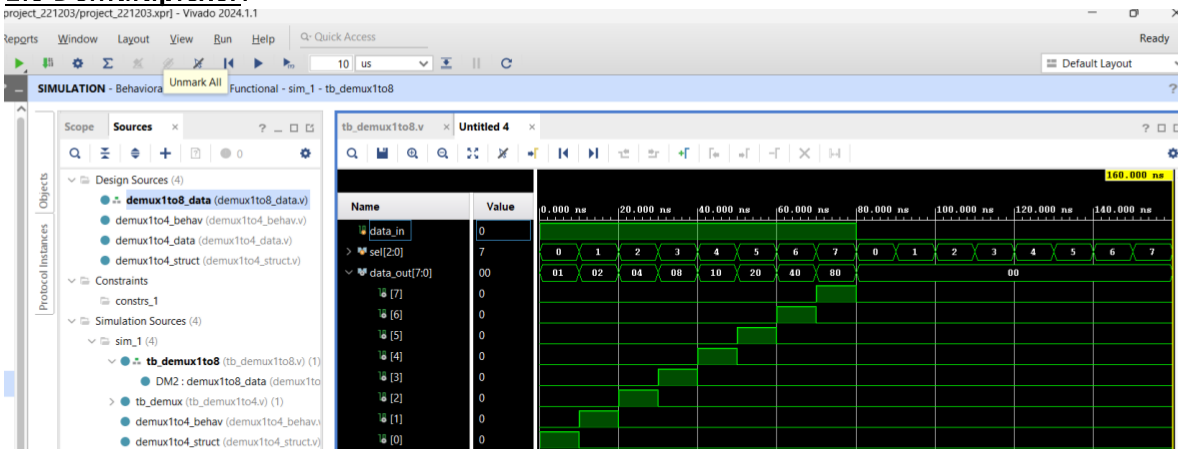
1:4 Demultiplexer in dataflow model:


project_221203/project_221203.xpr] - Vivado 2024.1.1
SIMULATION - Behavioral Simulation - Functional - sim_1 - tb_demux
Design Sources (3)
demux1to4_data (demux1to4_data.v)
demux1to4_behav (demux1to4_behav.v)
demux1to4_struct (demux1to4_struct.v)
Constraints
constrs_1
Simulation Sources (3)
sim_1 (3)
tb_demux (tb_demux1to4.v) (1)
DM2 : demux1to4_data (demux1to
tb_demux1to4.v
Untitled 2
44.898 ns
Name
Value
data_in 0
sel[1:0] 0
[1] 0
[0] 0
data_out[3:0] 0
[3] 0
[2] 0
[1] 0
[0] 0
0 1 2 3 0 1 2 3
1 2 4 8 0

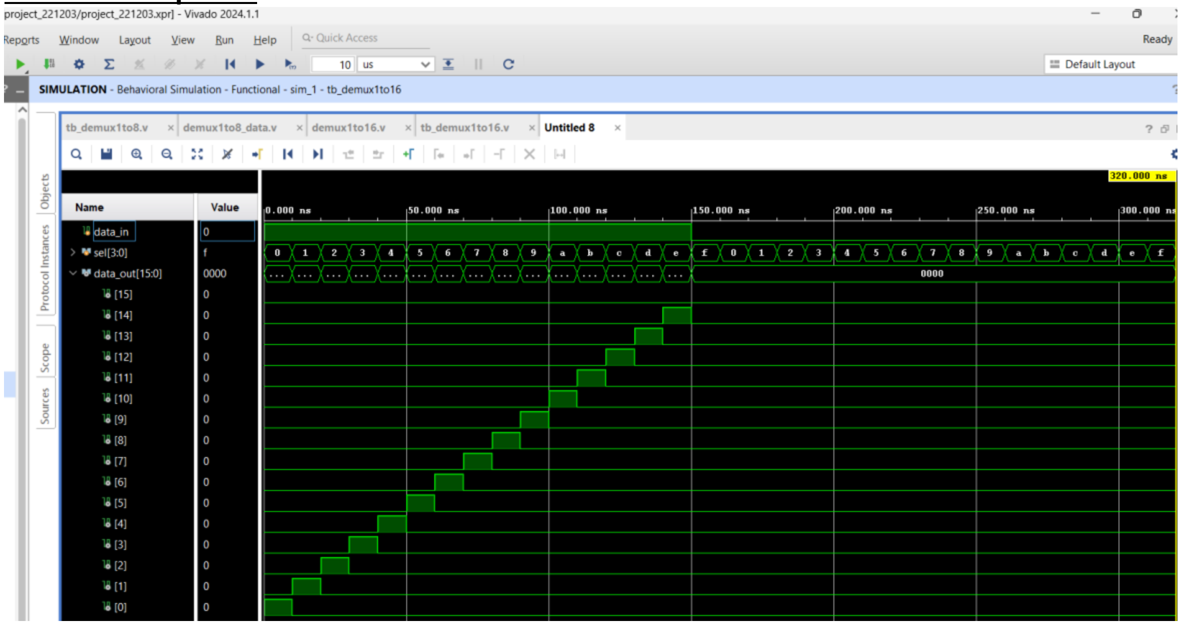
1:4 Demultiplexer in structural model:


project_221203/project_221203.xpr] - Vivado 2024.1.1
SIMULATION - Behavioral Simulation - Functional - sim_1 - tb_demux
Design Sources (3)
demux1to4_struct (demux1to4_struct.v)
demux1to4_behav (demux1to4_behav.v)
demux1to4_data (demux1to4_data.v)
Constraints
constrs_1
Simulation Sources (3)
sim_1 (3)
tb_demux (tb_demux1to4.v) (1)
DM3 : demux1to4_struct (demux1t
tb_demux1to4.v
Untitled 3
7.710 ns
Name
Value
data_in 1
sel[1:0] 0
[1] 0
[0] 0
data_out[3:0] 1
[3] 0
[2] 0
[1] 0
[0] 1
0 1 2 3 0 1 2 3
1 2 4 8 0

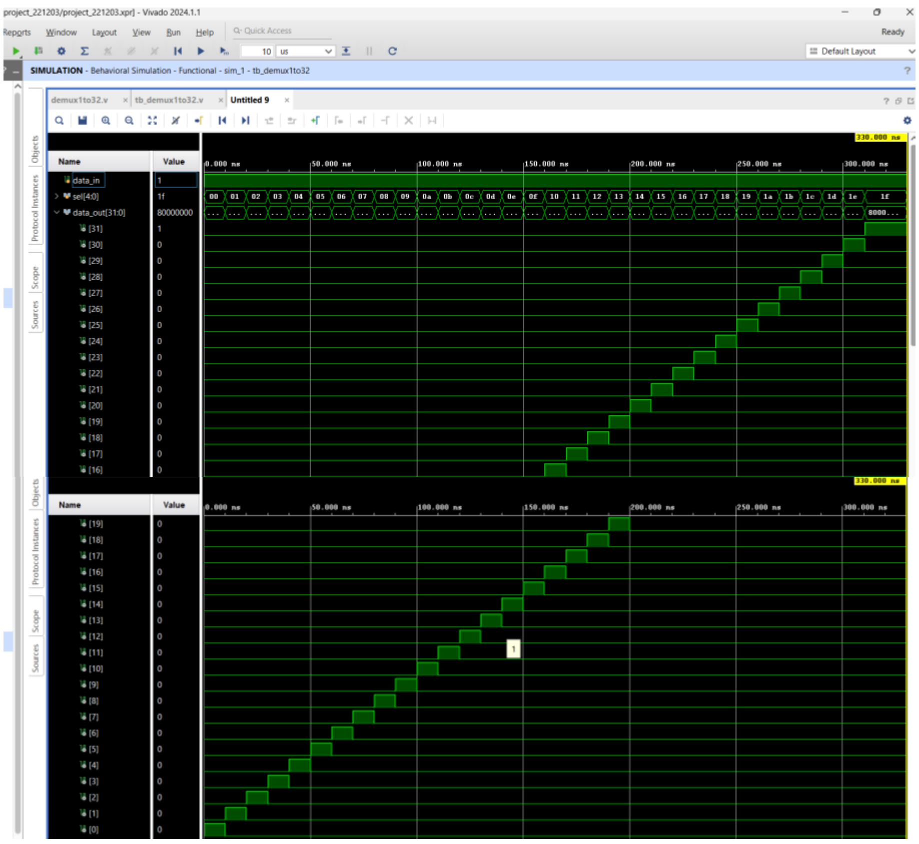
1:8 Demultiplexer:


project_221203/project_221203.xpr] - Vivado 2024.1.1
SIMULATION - Behavioral Simulation - Functional - sim_1 - tb_demux1to8
Unmark All
Design Sources (4)
demux1to8_data (demux1to8_data.v)
demux1to4_behav (demux1to4_behav.v)
demux1to4_data (demux1to4_data.v)
demux1to4_struct (demux1to4_struct.v)
Constraints
constrs_1
Simulation Sources (4)
sim_1 (4)
tb_demux1to8 (tb_demux1to8.v) (1)
DM2 : demux1to8_data (demux1to
tb_demux (tb_demux1to4.v) (1)
tb_demux1to8.v
Untitled 4
160.000 ns
Name
Value
data_in 0
sel[2:0] 7
data_out[7:0] 00
[7] 0
[6] 0
[5] 0
[4] 0
[3] 0
[2] 0
[1] 0
[0] 0
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
01 02 04 08 10 20 40 80 00

1:16 Demultiplexer:



1:32 Demultiplexer:


project_221203/project_221203.xpr] - Vivado 2024.1.1
SIMULATION - Behavioral Simulation - Functional - sim_1 - tb_demux1to32
demux1to32.v
tb_demux1to32.v
Untitled 9
Name
Value
data_in
sel[4:0]
data_out[31:0]
80000000
330.000 ns

HARDWARE OUTPUT:

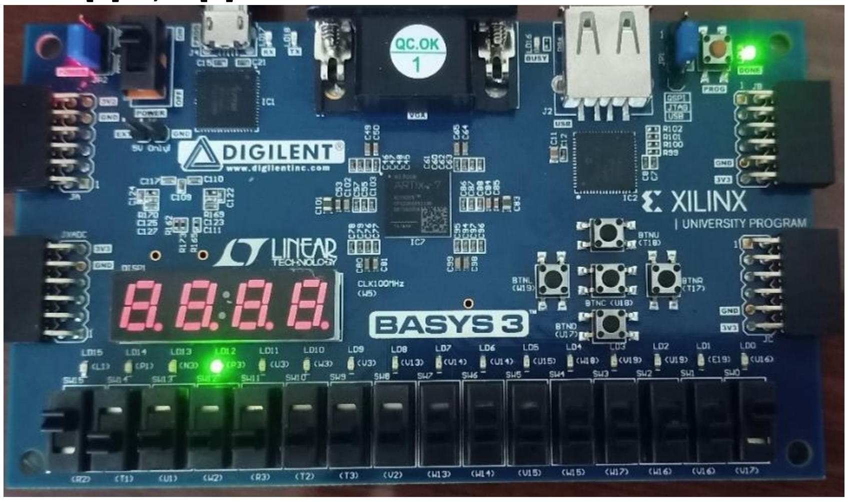
1:4 Demultiplexer:

I/O ports:

data_out[3]: L1	data_out[0]: P3	sel[1]: T1
data_out[2]: P1		sel[0]: U1
data_out[1]: N3		data_in: R2

For data_in=1

sel[1]=0,sel[0]=0



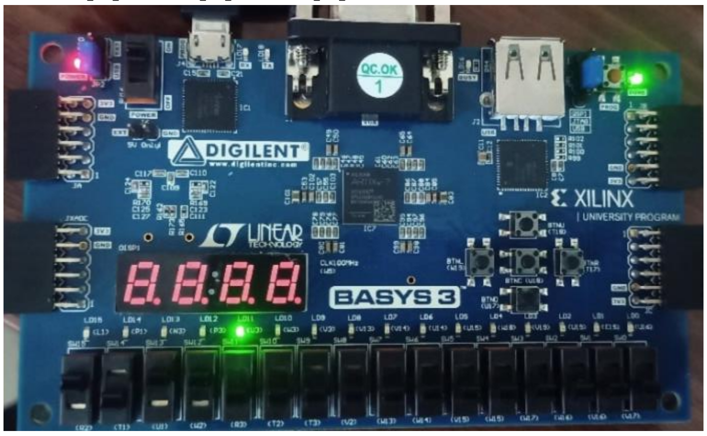
1:8 Demultiplexer:

I/O ports:

data_out[7]:L1	data_out[3]: U3	sel[2]: T1
data_out[6]: P1	data_out[2]: W3	sel[1]: U1
data_out[5]: N3	data_out[1]: V3	sel[0]: W2
data_out[4]: P3	data_out[0]: V13	data_in: R2

For data_in=1

sel[2]=0 sel[1]=1,sel[0]=1



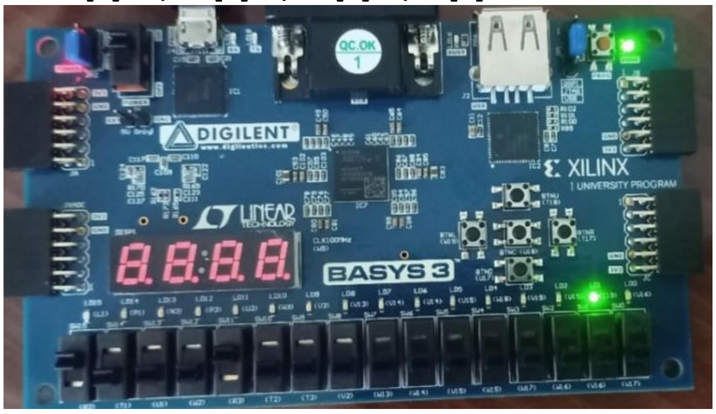
1:16 Demultiplexer:

I/O ports:

data_out[15]:L1	data_out[8]: V13	data_out[1]: E19
data_out[14]: P1	data_out[7]: V14	data_out[0]: U16
data_out[13]: N3	data_out[6]: U14	sel[3]: T1
data_out[12]: P3	data_out[5]: U15	sel[2]: U1
data_out[11]: U3	data_out[4]: W18	sel[1]: W2
data_out[10]: W3	data_out[3]: V19	sel[0]: R3
data_out[9]: V3	data_out[2]: U19	data_in: R2

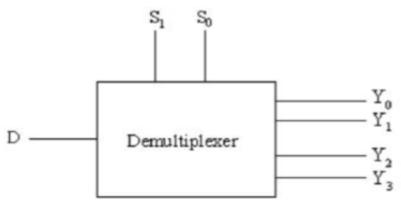
For data_in=1

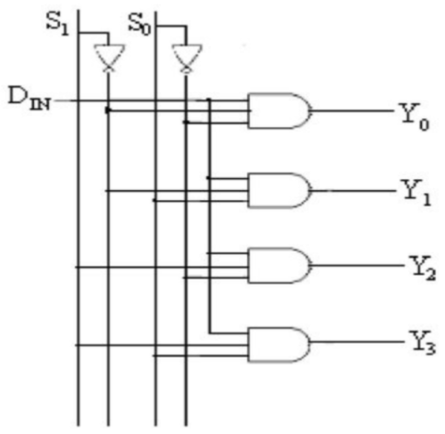
sel[3]=0 , sel[2]=0, sel[1]=0,sel[0]=1



CIRCUIT DIAGRAMS AND TRUTH TABLES:

1:4 Demultiplexer:





Inputs		Output
S_1	S_0	
0	0	Y_0=D
0	1	Y_1=D
1	0	Y_2=D
1	1	Y_3=D

Y0 = DS1’S0’

Y1 = DS1S0’

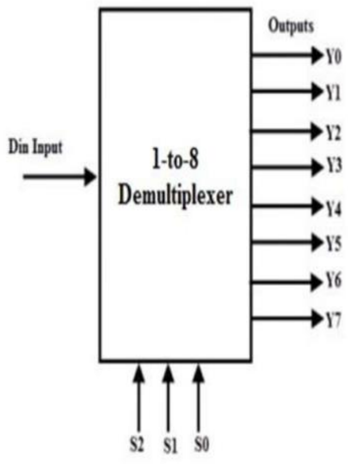
Y2 = DS1’S0

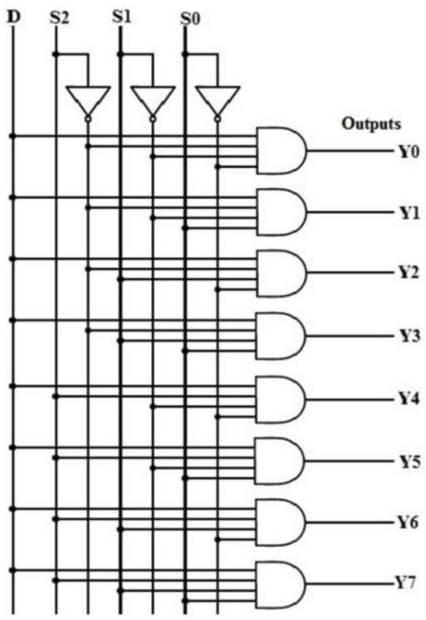
Y3 = DS1S0

1:8 Demultiplexer:

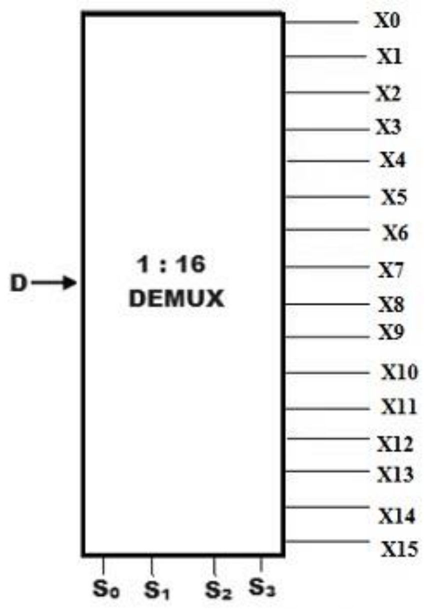
S2	S1	S0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	0	0	0	0	0	0	0	0	0	D
0	0	1	0	0	0	0	0	0	D	0
0	1	0	0	0	0	0	0	D	0	0
0	1	1	0	0	0	0	D	0	0	0
1	0	0	0	0	0	D	0	0	0	0
1	0	1	0	0	D	0	0	0	0	0
1	1	0	0	D	0	0	0	0	0	0
1	1	1	D	0	0	0	0	0	0	0

Y0 = DS2'S1'S0'
Y1 = DS2'S1'S0
Y2 = DS2'S1S0'
Y3 = DS2'S1S0
Y4 = DS2S1'S0'
Y5 = DS2S1'S0
Y6 = DS2S1S0'
Y7 = DS2S1S0





1:16 Demultiplexer:



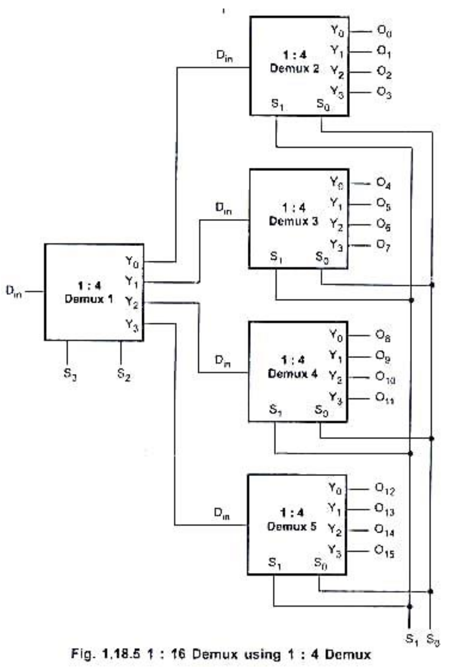
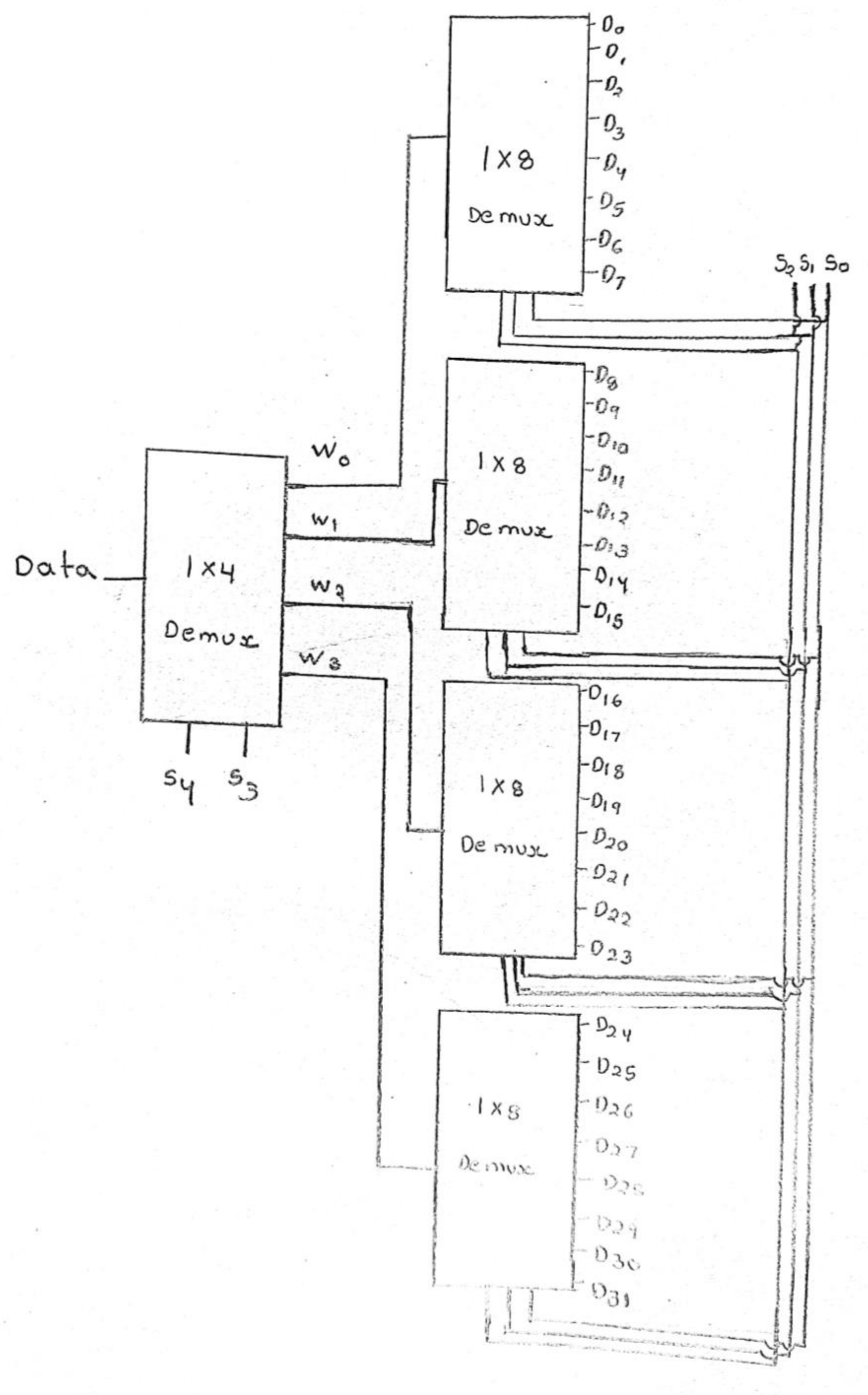


Fig. 1.18.5 1 : 16 Demux using 1 : 4 Demux

1:32 Demultiplexer:

Using 1:4 demultiplexer and 1:8 demultiplexer



RESULT:

Thus, the logic circuits for the multiplexers and demultiplexers are designed in Verilog HDL and the output combinations are verified using Xilinx Vivado software on the Basys3 FPGA board.