LAB 1 – Raja Aadhithan

Design - Full adder using Half adder:

Code:

```
module half_adder(input a,b,
            output sum,carry);

   //Understand the Data-flow abstraction
   assign sum   = a ^ b;
   assign carry = a & b;

endmodule
```

```
module full_adder( input a_in,
                   b_in,
          c_in,
          output sum_out,
          carry_out);

wire w1,w2,w3;

half_adder ha1(.a(a_in), .b(b_in), .sum(w1), .carry(w2));
half_adder ha2(.a(c_in), .b(w1), .sum(sum_out), .carry(w3));
or or1(carry_out, w2,w3);



endmodule
```

Test bench:

```
module full_adder_tb();

   //Testbench global variables
   reg   a,b,cin;
   wire  sum,carry;

   //Variable for loop iteration
   integer i;

   //Step1 : Instantiate the full adder with order based port mapping
full_adder DUT(a,b,cin,sum,carry);
```

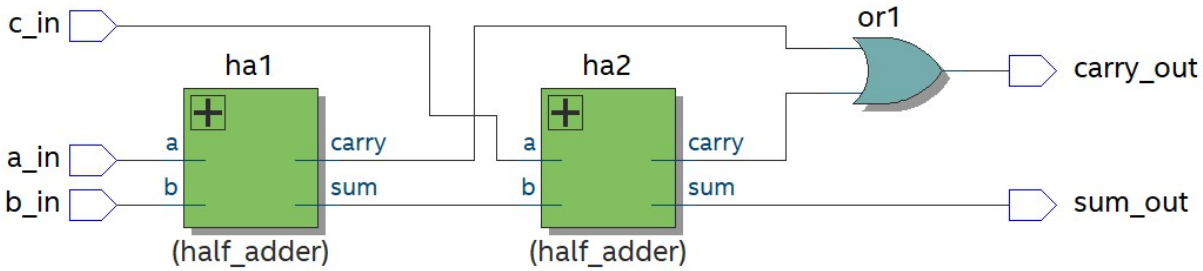
```
   //Process to initialize the variables at 0ns
   initial
      begin
         a   = 1'b0;
    b   = 1'b0;
    cin = 1'b0;
      end

   //Process to generate stimulus using for loop
   initial
      begin
    for (i=0;i<8;i=i+1)
       begin
          {a,b,cin}=i;
          #10;
       end
      end

   //Process to monitor the changes in the variables
   initial
      $monitor("Input a=%b, b=%b, c=%b, Output sum =%b, carry=%b",a,b,cin,sum,carry);

   //Process to terminate simulation after 100ns
   initial #100 $finish;


endmodule
```

Wave:

Output:

```
# Input a=0, b=0, c=0, Output sum =0, carry=0
# Input a=0, b=0, c=1, Output sum =1, carry=0
# Input a=0, b=1, c=0, Output sum =1, carry=0
# Input a=0, b=1, c=1, Output sum =0, carry=1
# Input a=1, b=0, c=0, Output sum =1, carry=0
# Input a=1, b=0, c=1, Output sum =0, carry=1
# Input a=1, b=1, c=0, Output sum =0, carry=1
# Input a=1, b=1, c=1, Output sum =1, carry=1
```

RTL:


c_in
a_in
b_in
ha1
a
b
carry
sum
(half_adder)
ha2
a
b
carry
sum
(half_adder)
or1
carry_out
sum_out

Exercises:

Design : Ripple adder

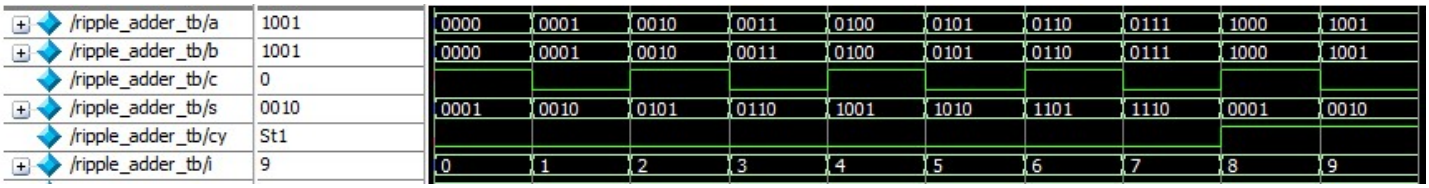
Code:

```
module ripple_adder(input [3:0] a,b, input cin,
             output[3:0] sum,output carry);
wire [2:0] w;

full_adder FA1(a[0],b[0],cin,sum[0],w[0]);
full_adder FA2(a[1],b[1],w[0],sum[1],w[1]);
full_adder FA3(a[2],b[2],w[1],sum[2],w[2]);
full_adder FA4(a[3],b[3],w[2],sum[3],carry);
endmodule
```

Test bench:

```
module ripple_adder_tb();
reg [3:0]a,b;
reg c;
wire [3:0]s;
wire cy;
integer i;
initial begin
a=0;
b=0;
c=0; end

ripple_adder DUT(a,b,c,s,cy);
initial begin
for (i=0;i<16;i=i+1)
begin
a=i;
b=i;
c=!c;
#10;
end
end
initial
      $monitor("Input a=%b, b=%b, c=%b, Output sum =%b, carry=%b",a,b,c,s,cy);

   //Process to terminate simulation after 100ns
   initial #100 $finish;


endmodule
```

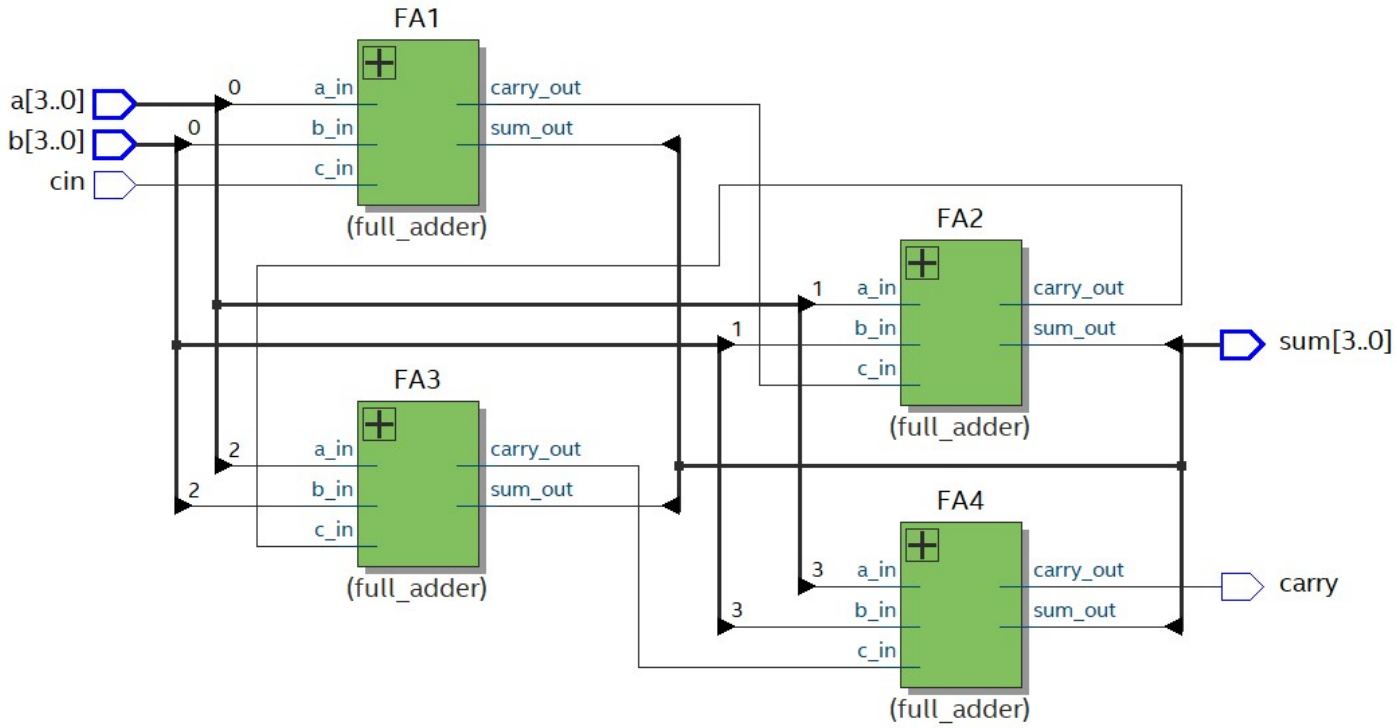
Wave:



Output:

```
VSIM 28> run -all
# Input a=0000, b=0000, c=1, Output sum =0001, carry=0
# Input a=0001, b=0001, c=0, Output sum =0010, carry=0
# Input a=0010, b=0010, c=1, Output sum =0101, carry=0
# Input a=0011, b=0011, c=0, Output sum =0110, carry=0
# Input a=0100, b=0100, c=1, Output sum =1001, carry=0
# Input a=0101, b=0101, c=0, Output sum =1010, carry=0
# Input a=0110, b=0110, c=1, Output sum =1101, carry=0
# Input a=0111, b=0111, c=0, Output sum =1110, carry=0
# Input a=1000, b=1000, c=1, Output sum =0001, carry=1
# Input a=1001, b=1001, c=0, Output sum =0010, carry=1
# ** Note: $finish    : C:/Users/Aadhithan/Documents/Verilog_labs/lab1/sim/ripple_ad
der_tb.v(26)
#    Time: 100 ps  Iteration: 0  Instance: /ripple_adder_tb
```

RTL:



Design : 4:1 mux using 2:1 mux

Code:

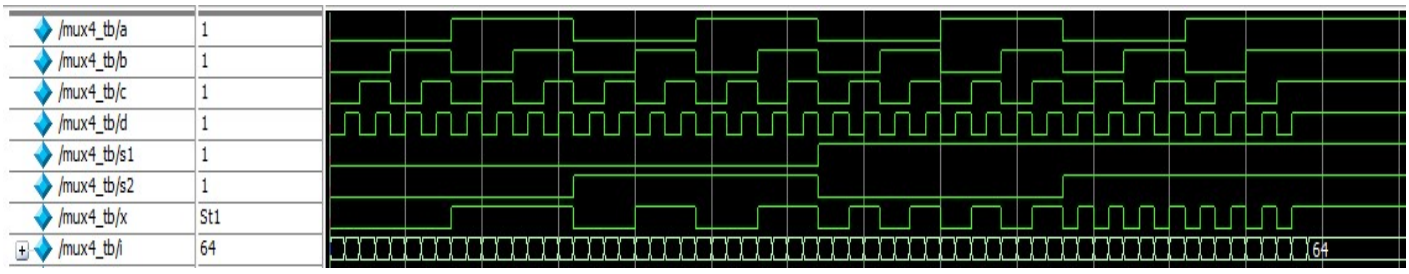
```
module mux2(input a,b,s, output x);
assign x = s ? b : a;
endmodule
```

```
module mux4(input a,b,c,d,s1,s2,output x);
wire w1,w2;
mux2 mu1(a,b,s2,w1);
mux2 mu2(c,d,s2,w2);
mux2 mu3(w1,w2,s1,x);
endmodule
```

Test bench:

```
module mux4_tb();
reg a,b,c,d,s1,s2;
wire x;
integer i;
mux4 DUT(a,b,c,d,s1,s2,x);
initial begin
   a=0;b=0;c=0;d=0;s1=0;s2=0;
end
initial begin
    for(i=0;i<64;i=i+1)
    begin
        {s1,s2,a,b,c,d}=i;
        #10;
end
end
initial #1000 $finish;
endmodule
```

Wave:

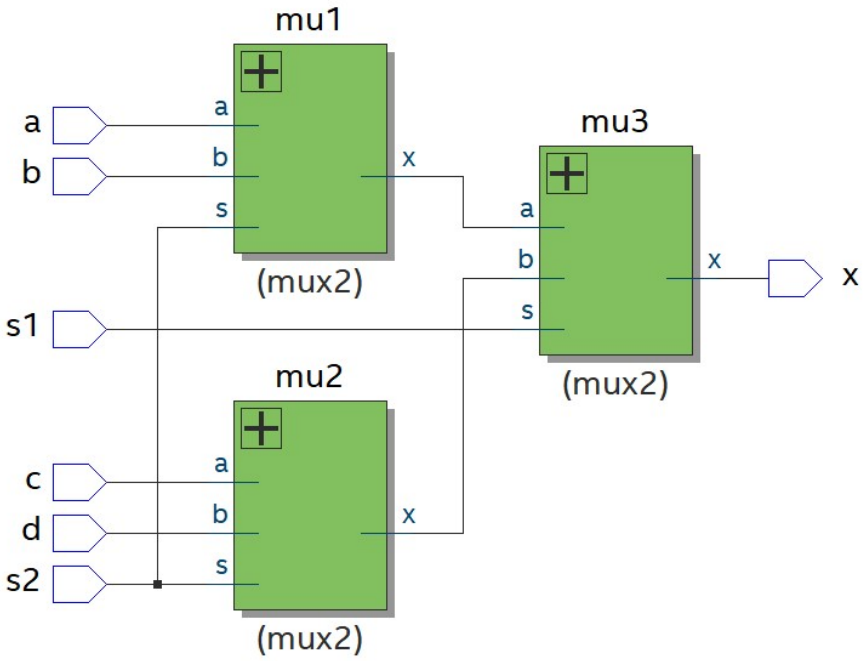


Output:

```
VSIM 34> run -all
# @time:  0- selsct is 00 nad input is 0,0,0,0 for output is 0
# @time: 10- selsct is 00 nad input is 0,0,0,1 for output is 0
# @time: 20- selsct is 00 nad input is 0,0,1,0 for output is 0
# @time: 30- selsct is 00 nad input is 0,0,1,1 for output is 0
# @time: 40- selsct is 00 nad input is 0,1,0,0 for output is 0
# @time: 50- selsct is 00 nad input is 0,1,0,1 for output is 0
# @time: 60- selsct is 00 nad input is 0,1,1,0 for output is 0
# @time: 70- selsct is 00 nad input is 0,1,1,1 for output is 0
# @time: 80- selsct is 00 nad input is 1,0,0,0 for output is 1
# @time: 90- selsct is 00 nad input is 1,0,0,1 for output is 1
# @time:100- selsct is 00 nad input is 1,0,1,0 for output is 1
# @time:110- selsct is 00 nad input is 1,0,1,1 for output is 1
# @time:120- selsct is 00 nad input is 1,1,0,0 for output is 1
# @time:130- selsct is 00 nad input is 1,1,0,1 for output is 1
# @time:140- selsct is 00 nad input is 1,1,1,0 for output is 1
# @time:150- selsct is 00 nad input is 1,1,1,1 for output is 1
# @time:160- selsct is 01 nad input is 0,0,0,0 for output is 0
# @time:170- selsct is 01 nad input is 0,0,0,1 for output is 0
# @time:180- selsct is 01 nad input is 0,0,1,0 for output is 0
# @time:190- selsct is 01 nad input is 0,0,1,1 for output is 0
# @time:200- selsct is 01 nad input is 0,1,0,0 for output is 1
# @time:210- selsct is 01 nad input is 0,1,0,1 for output is 1
# @time:220- selsct is 01 nad input is 0,1,1,0 for output is 1
# @time:230- selsct is 01 nad input is 0,1,1,1 for output is 1
# @time:240- selsct is 01 nad input is 1,0,0,0 for output is 0
# @time:250- selsct is 01 nad input is 1,0,0,1 for output is 0
# @time:260- selsct is 01 nad input is 1,0,1,0 for output is 0
# @time:270- selsct is 01 nad input is 1,0,1,1 for output is 0
# @time:280- selsct is 01 nad input is 1,1,0,0 for output is 1
# @time:290- selsct is 01 nad input is 1,1,0,1 for output is 1
# @time:300- selsct is 01 nad input is 1,1,1,0 for output is 1
# @time:310- selsct is 01 nad input is 1,1,1,1 for output is 1
# @time:320- selsct is 10 nad input is 0,0,0,0 for output is 0
# @time:330- selsct is 10 nad input is 0,0,0,1 for output is 0
```

```
# @time:330- selsct is 10 nad input is 0,0,0,1 for output is 0
# @time:340- selsct is 10 nad input is 0,0,1,0 for output is 1
# @time:350- selsct is 10 nad input is 0,0,1,1 for output is 1
# @time:360- selsct is 10 nad input is 0,1,0,0 for output is 0
# @time:370- selsct is 10 nad input is 0,1,0,1 for output is 0
# @time:380- selsct is 10 nad input is 0,1,1,0 for output is 1
# @time:390- selsct is 10 nad input is 0,1,1,1 for output is 1
# @time:400- selsct is 10 nad input is 1,0,0,0 for output is 0
# @time:410- selsct is 10 nad input is 1,0,0,1 for output is 0
# @time:420- selsct is 10 nad input is 1,0,1,0 for output is 1
# @time:430- selsct is 10 nad input is 1,0,1,1 for output is 1
# @time:440- selsct is 10 nad input is 1,1,0,0 for output is 0
# @time:450- selsct is 10 nad input is 1,1,0,1 for output is 0
# @time:460- selsct is 10 nad input is 1,1,1,0 for output is 1
# @time:470- selsct is 10 nad input is 1,1,1,1 for output is 1
# @time:480- selsct is 11 nad input is 0,0,0,0 for output is 0
# @time:490- selsct is 11 nad input is 0,0,0,1 for output is 1
# @time:500- selsct is 11 nad input is 0,0,1,0 for output is 0
# @time:510- selsct is 11 nad input is 0,0,1,1 for output is 1
# @time:520- selsct is 11 nad input is 0,1,0,0 for output is 0
# @time:530- selsct is 11 nad input is 0,1,0,1 for output is 1
# @time:540- selsct is 11 nad input is 0,1,1,0 for output is 0
# @time:550- selsct is 11 nad input is 0,1,1,1 for output is 1
# @time:560- selsct is 11 nad input is 1,0,0,0 for output is 0
# @time:570- selsct is 11 nad input is 1,0,0,1 for output is 1
# @time:580- selsct is 11 nad input is 1,0,1,0 for output is 0
# @time:590- selsct is 11 nad input is 1,0,1,1 for output is 1
# @time:600- selsct is 11 nad input is 1,1,0,0 for output is 0
# @time:610- selsct is 11 nad input is 1,1,0,1 for output is 1
# @time:620- selsct is 11 nad input is 1,1,1,0 for output is 0
# @time:630- selsct is 11 nad input is 1,1,1,1 for output is 1
# ** Note: $finish    : C:/Users/Aadhithan/Documents/Verilog_labs/lab1/4_2_mux/muv4_
tb.v(16)
#    Time: 1 ns  Iteration: 0  Instance: /mux4_tb
# 1
```

RTL:



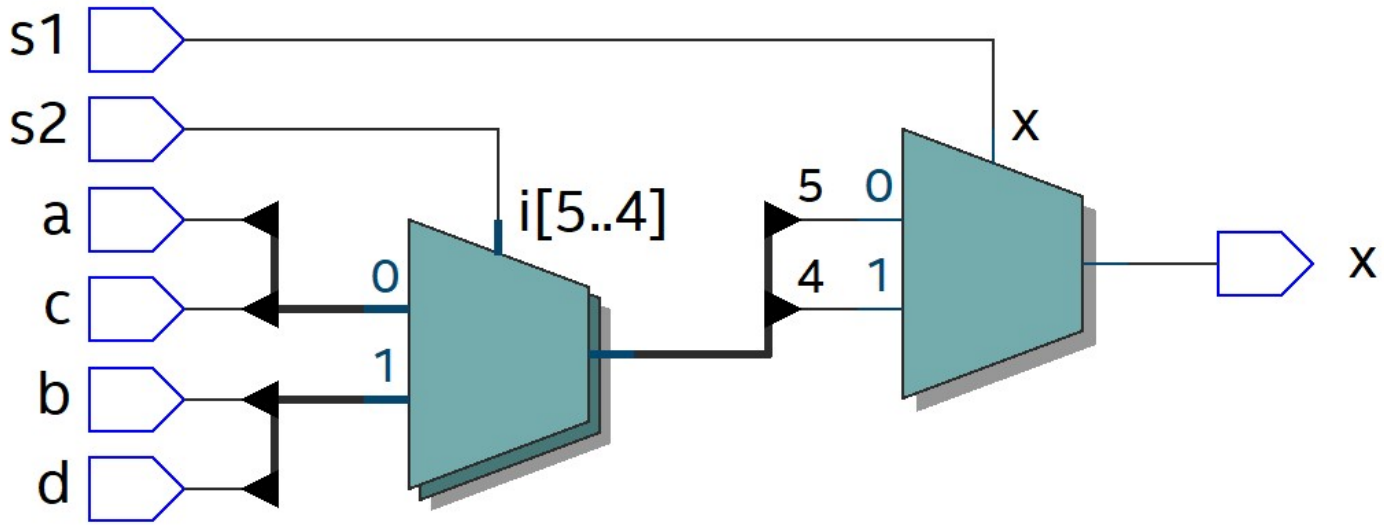
Assignment:

Design: RTL for full adder.

Code:

```
module dataflow_fulladder(input a,b,c,d,s2,s1, output x);
assign x = s1 ? (s2 ? d : c) : (s2 ? b :a );
endmodule
```

RTL:

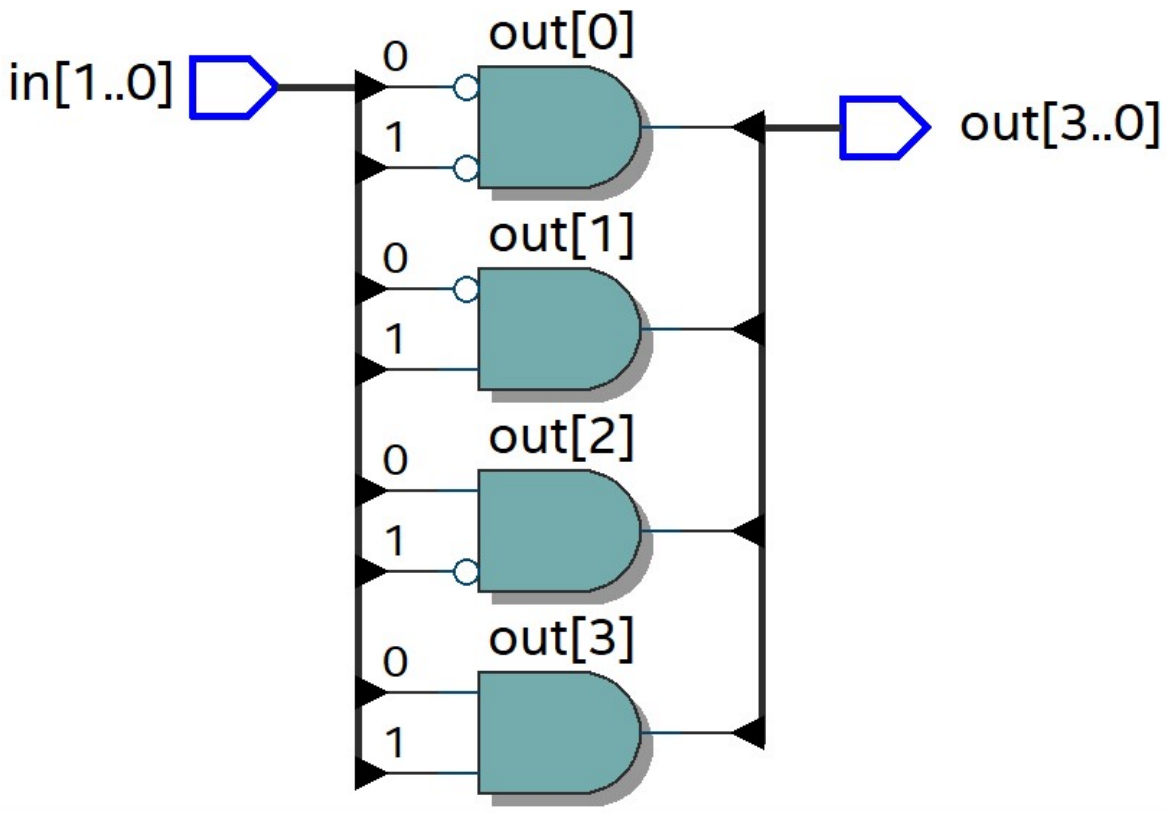


Design: 2x4 decoder.

Code:

```
module decoder(
    input [1:0] in,
    output [3:0] out
    );
  assign out[0] = (~in[1]) & (~in[0]) ;
  assign out[1] = ( in[1]) & (~in[0]) ;
  assign out[2] = (~in[1]) & ( in[0]) ;
  assign out[3] = ( in[1]) & ( in[0]) ;

endmodule
```

RTL:



Design: 8x3 priority encoder.

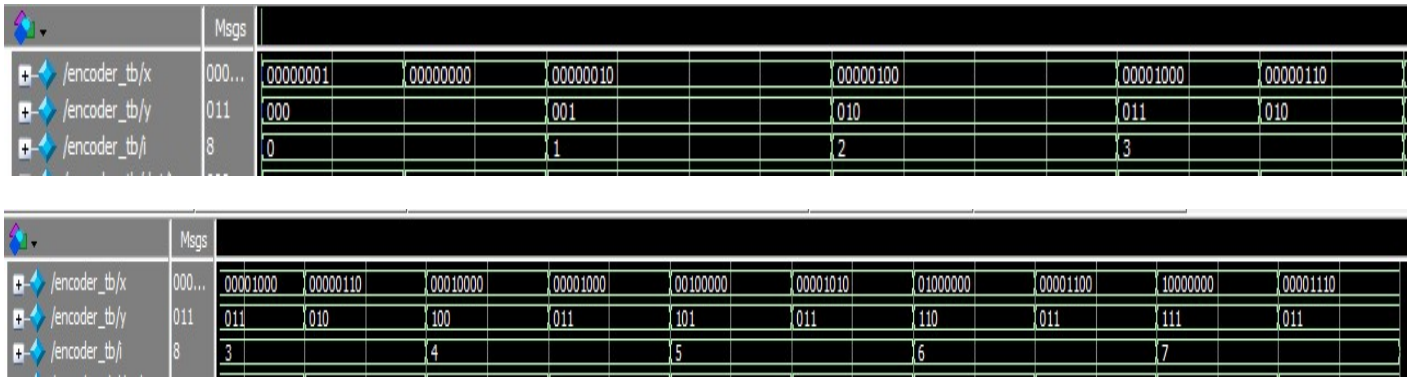
Code:

```
module encoder(input [7:0]in , output [2:0] out);
wire[4:0] x;
wire[5:0] n;
not(n[5],in[5]);
not(n[4],in[4]);
not(n[3],in[3]);
not(n[2],in[6]);
not(n[1],in[2]);
or(out[2],in[7],in[6],in[5],in[4]);
and(x[0],in[3],n[5],n[4]);//!5,!4,3
and(x[1],n[5],n[4],n[3],in[2]);//!5,!4,!3,2
and(x[2],x[0],n[2]);//!6,!5,!4,3
and(x[3],n[2],in[5]);//!6,5
and(x[4],n[2],n[5],n[4],n[3],n[1],in[1]);//!6,!5,!4,!3,!2,1
or(out[1],in[7],in[6],x[0],x[1]);
or(out[0],in[7],x[3],x[4],x[2]);
endmodule
```

Test bench:

```
module encoder_tb();
reg [7:0] x;
wire [2:0] y;
integer i;
encoder dat(x,y);
initial begin
    for(i=0; i<8; i = i+1)
    begin
        x = 2**i;
        #10;
        x=2*i;
        #10;
    end
$finish;
end

initial $monitor("@ time: %3dps the input is %8b output is %3b",$time,x,y);

endmodule
```

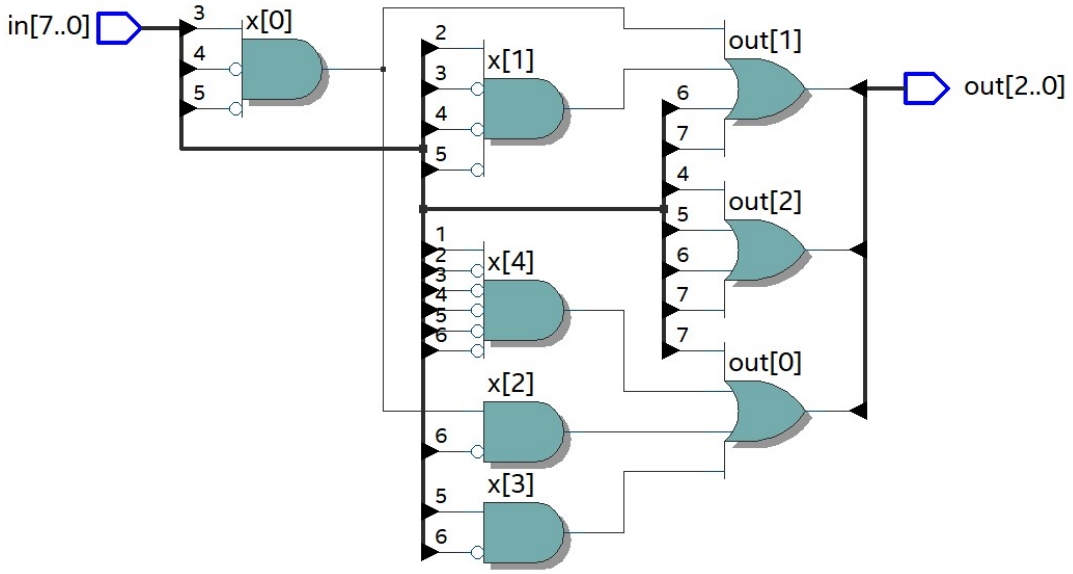
Wave:



Output:

```
VSIM 38> run -all
# @ time:   0ps the input is 00000001 output is 000
# @ time:  10ps the input is 00000000 output is 000
# @ time:  20ps the input is 00000010 output is 001
# @ time:  40ps the input is 00000100 output is 010
# @ time:  60ps the input is 00001000 output is 011
# @ time:  70ps the input is 00000110 output is 010
# @ time:  80ps the input is 00010000 output is 100
# @ time:  90ps the input is 00001000 output is 011
# @ time: 100ps the input is 00100000 output is 101
# @ time: 110ps the input is 00001010 output is 011
# @ time: 120ps the input is 01000000 output is 110
# @ time: 130ps the input is 00001100 output is 011
# @ time: 140ps the input is 10000000 output is 111
# @ time: 150ps the input is 00001110 output is 011
# ** Note: $finish    : C:/Users/Aadhithan/Documents/Verilog_labs/lab1/encoder/encoder_tb.v(14)
#    Time: 160 ps  Iteration: 0  Instance: /encoder_tb
```

RTL:



Design: 4x1 mux.

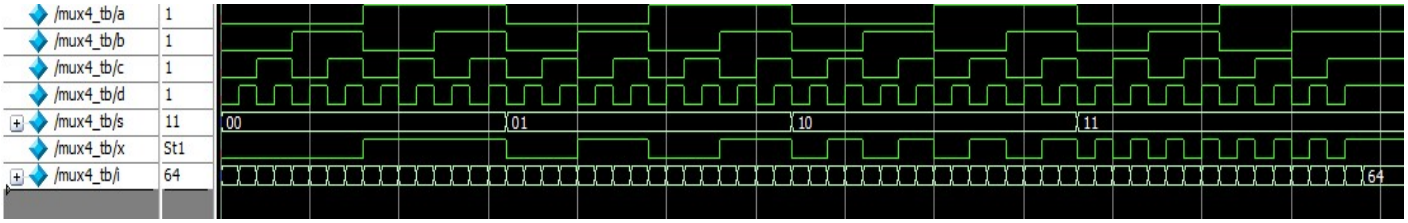
Code:

```
module mux_dec(input A,B,C,D, input [1:0] in, output Y);
wire [3:0] out;
reg x;
  assign out[0] = (~in[1]) & (~in[0]) ;
  assign out[1] = ( in[1]) & (~in[0]) ;
  assign out[2] = (~in[1]) & ( in[0]) ;
  assign out[3] = ( in[1]) & ( in[0]) ;
  always@(*)begin
      case(in)
      2'b00:x <= out[0] ? A : 1'bz;
      2'b01:x <= out[2] ? B : 1'bz ;
      2'b10:x <= out[1] ? C : 1'bz ;
      2'b11:x <= out[3] ? D : 1'bz ;
      endcase
  end
assign Y = x;
endmodule
```

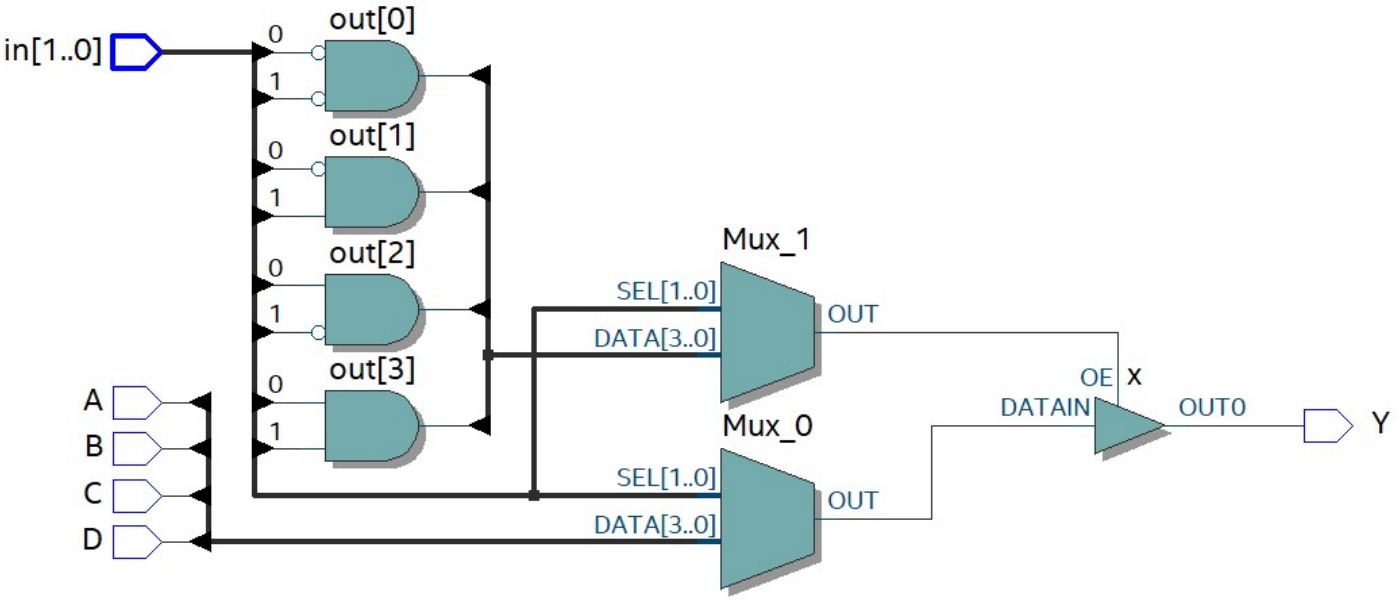
Test bench:

```
module mux4_tb();
reg a,b,c,d;
reg [1:0] s;
wire x;
integer i;
mux_dec DUT(a,b,c,d,s,x);
initial begin
   a=0;b=0;c=0;d=0;s = 0;
end
initial begin
    for(i=0;i<64;i=i+1)
    begin
        {s,a,b,c,d}=i;
        #10;
end
end
initial #1000 $finish;
endmodule
```

Wave:


/mux4_tb/a
/mux4_tb/b
/mux4_tb/c
/mux4_tb/d
/mux4_tb/s
/mux4_tb/x
/mux4_tb/i
1
1
1
1
11
St1
64
00
01
10
11
64

RTL:


in[1..0]
A
B
C
D
0
1
out[0]
out[1]
out[2]
out[3]
Mux_1
SEL[1..0]
DATA[3..0]
OUT
Mux_0
SEL[1..0]
DATA[3..0]
OUT
OE
x
DATAIN
OUT0
Y

Design: Bidirectional buffer

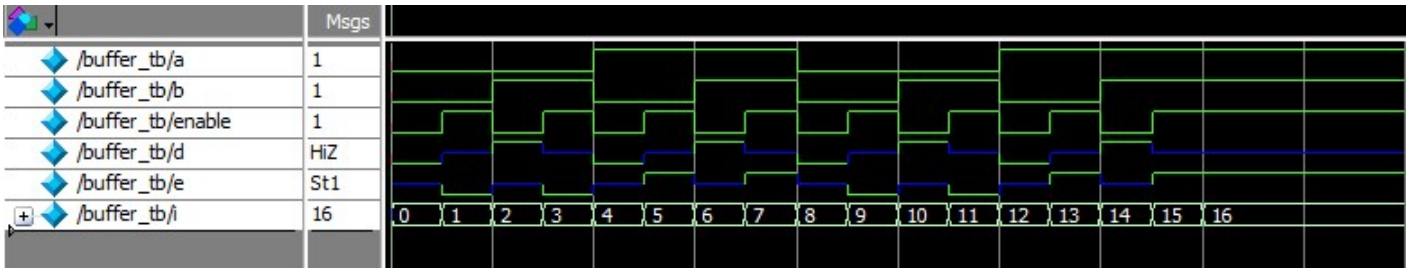
Code:

```
module bufer(inout a,b, input c);
bufif1(b,a,c);
bufif0(a,b,c);
endmodule
```

Testbench:

```
module bufer_tb();
wire a,b;
reg c,ta,tb;
integer i;
bufer dut(a,b,c);
initial begin
for(i=0;i<8;i=i+1)
begin
        {ta,tb,c}=i[2:0];
        #10;
end
end
assign a = c ? ta : 1'bz;
assign b = c ? 1'bz : tb;
endmodule
```

Wave:



Output:

```
VSIM 49> run -all
# @time =   0: The value of a is 0 and b is 0
#  enable is 0
#  output ar is 0 and br is z
# @time =  10: The value of a is 0 and b is 0
#  enable is 1
#  output ar is z and br is 0
# @time =  20: The value of a is 0 and b is 1
#  enable is 0
#  output ar is 1 and br is z
# @time =  30: The value of a is 0 and b is 1
#  enable is 1
#  output ar is z and br is 0
# @time =  40: The value of a is 1 and b is 0
#  enable is 0
#  output ar is 0 and br is z
# @time =  50: The value of a is 1 and b is 0
#  enable is 1
#  output ar is z and br is 1
# @time =  60: The value of a is 1 and b is 1
#  enable is 0
#  output ar is 1 and br is z
# @time =  70: The value of a is 1 and b is 1
#  enable is 1
#  output ar is z and br is 1
# @time =  80: The value of a is 0 and b is 0
#  enable is 0
#  output ar is 0 and br is z
# @time =  90: The value of a is 0 and b is 0
#  enable is 1
#  output ar is z and br is 0
# @time = 100: The value of a is 0 and b is 1
#  enable is 0
#  output ar is 1 and br is z
# @time = 100: The value of a is 0 and b is 1
#  enable is 0
#  output ar is 1 and br is z
# @time = 110: The value of a is 0 and b is 1
#  enable is 1
#  output ar is z and br is 0
# @time = 120: The value of a is 1 and b is 0
#  enable is 0
#  output ar is 0 and br is z
# @time = 130: The value of a is 1 and b is 0
#  enable is 1
#  output ar is z and br is 1
# @time = 140: The value of a is 1 and b is 1
#  enable is 0
#  output ar is 1 and br is z
# @time = 150: The value of a is 1 and b is 1
#  enable is 1
#  output ar is z and br is 1
```

RTL: