DSD LAB RECORD

Software

SIDAK SINGH (24BCE1460)
AKSHIT MADATHIL (24BCE5304)
NIKETH (24BAI1600)
SHAASVAT R (24BAI1612)
PRAJWAN

Experiment – 1

Verification of Boolean expression

```
add wave -position insertpoint sim:/test_2to1/*
VSIM 7> run
# sel = 0: i0 = 0: i1=0 --> f = x
# sel = 0: i0 = 1: i1=0 --> f = x
# sel = 0: i0 = 0: i1=1 --> f = x
# sel = 0: i0 = 1: i1=1 --> f = x
# sel = 1: i0 = 0: i1=0 --> f = 0
# sel = 1: i0 = 1: i1=0 --> f = 0
# sel = 1: i0 = 0: i1=1 --> f = 1
# sel = 1: i0 = 1: i1=1 --> f = 1

VSIM 8>
```

- ## Aim-

  To verify the given Boolean expression using Model Sim software.

  Y=AB

- ## Tools Required-

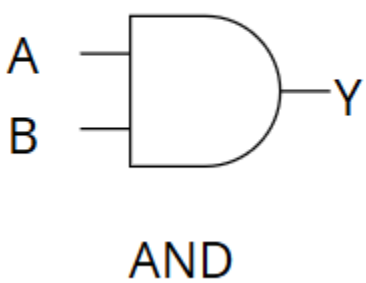
  Model Sim software

- ## Code-

Design block code-

```
Ln#
1   module pgmadd(a,b,y);
2   input a;
3   input b;
4   output y;
5   and (y,a,b);
6   endmodule
7   Test bench code-
8   module pgmadd_tb;
9   reg a;
10  reg b;
11  wire y;
12  pgmadd uut (a,b,y);
13  initial
14  begin
15  a = 0; b = 0;
16  #10 a = 0; b = 1;
17  #10 a = 1; b = 0;
18  #10 a = 1; b = 1;
19  #10 $stop;
20  $monitor($time, "a=%b, b=%b, y=%b", a,b,y);
21  $dumpfile("dump.vcd");
22  $dumpvars();
23  end
24  endmodule
25
26
```

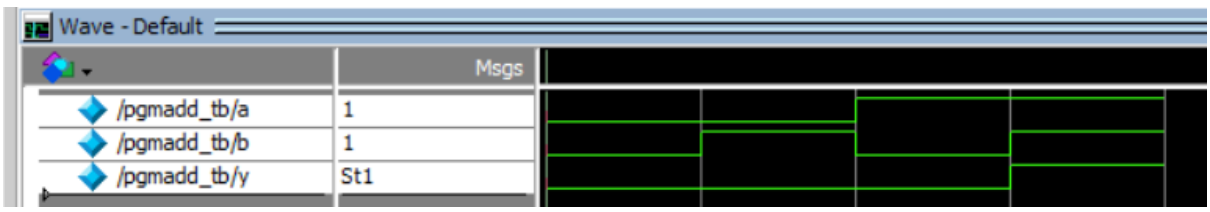
- Circuit Diagram-



- Truth Table-

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

- Output-




```
VSIM 6> run
#                    0 a=0, b=0, y=0
#                   10 a=0, b=1, y=0
#                   20 a=1, b=0, y=0
#                   30 a=1, b=1, y=1
# ** Note: $stop    : C:/Modelsim_Everything/Verify_Boolean_Tb.v(13)
#    Time: 40 ps  Iteration: 0  Instance: /pgmadd_tb
```

- Result-

Hence the given Boolean expression has been verified.

Experiment – 2

Implementation of Adder and subtractor Circuits

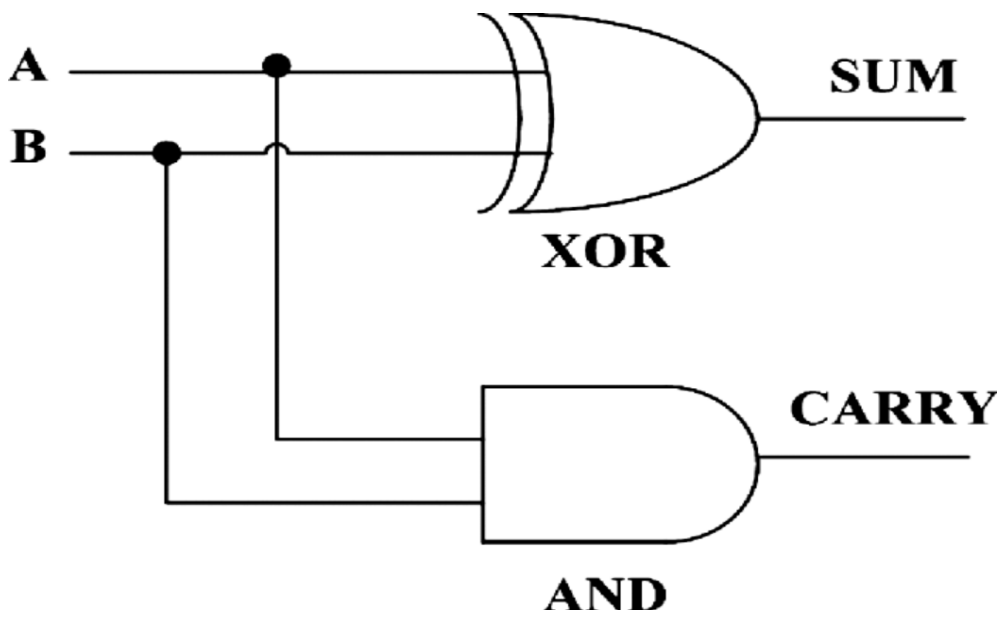
- Aim-

To develop the source code for adders and subtractors by using VERILOG (Modelsim) and obtain the simulation.

- Tools Required- Model Sim software

- Half Adder:

i)circuit diagram:



ii)code:

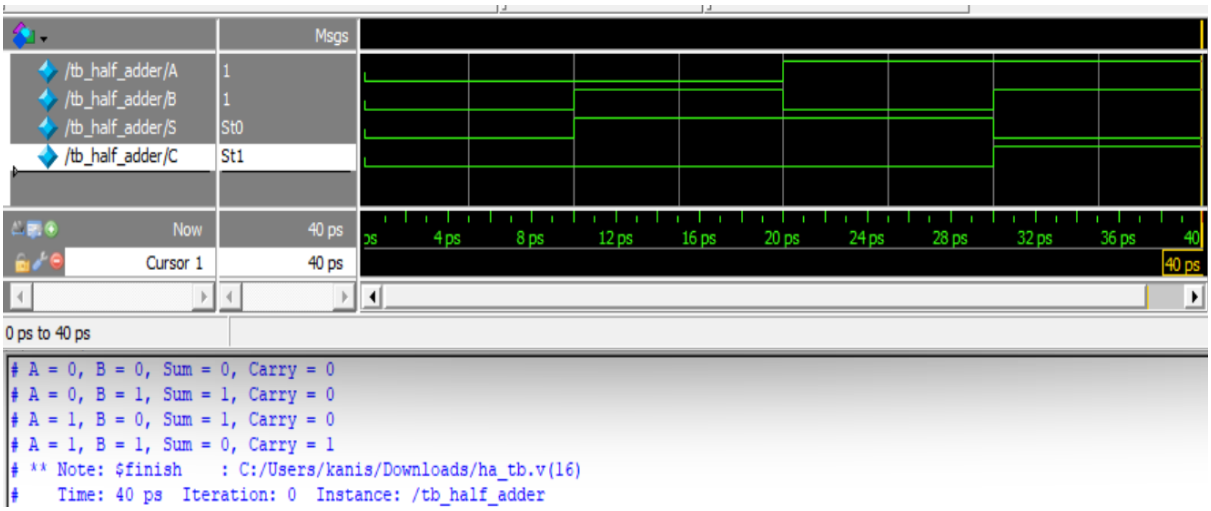
Design block -

```
module half_adder(
    input A,
    input B,
    output S,
    output C
);
    assign S = A ^ B;
    assign C = A & B;
endmodule
```

Test bench -

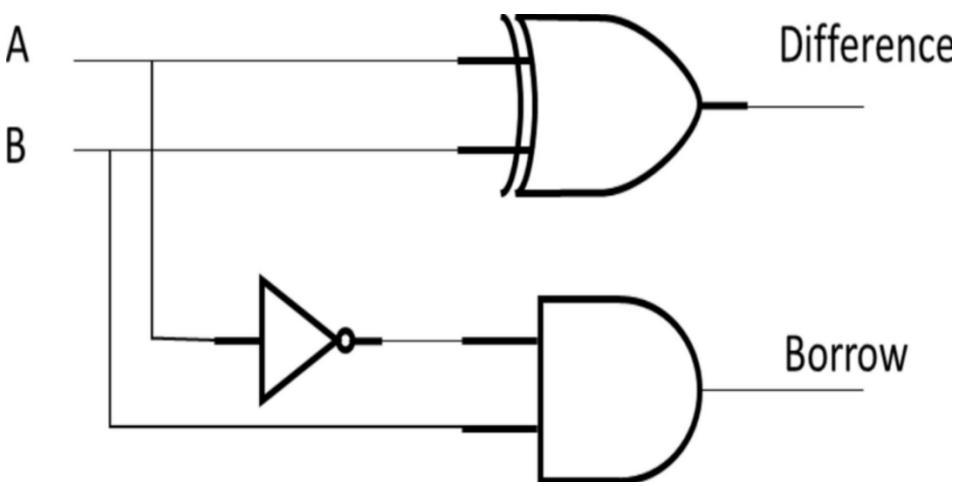
```
module tb_half_adder;
    reg A, B;
    wire S, C;
    half_adder uut (
        .A(A),
        .B(B),
        .S(S),
        .C(C)
    );
    initial begin
        $monitor("A = %b, B = %b, Sum = %b, Carry = %b", A, B, S, C);
        A = 0; B = 0; #10;
        A = 0; B = 1; #10;
        A = 1; B = 0; #10;
        A = 1; B = 1; #10;
        $stop;
    end
endmodule
```

iii)Output:



- Half Subtractor:

i)Circuit diagram:



ii)code:

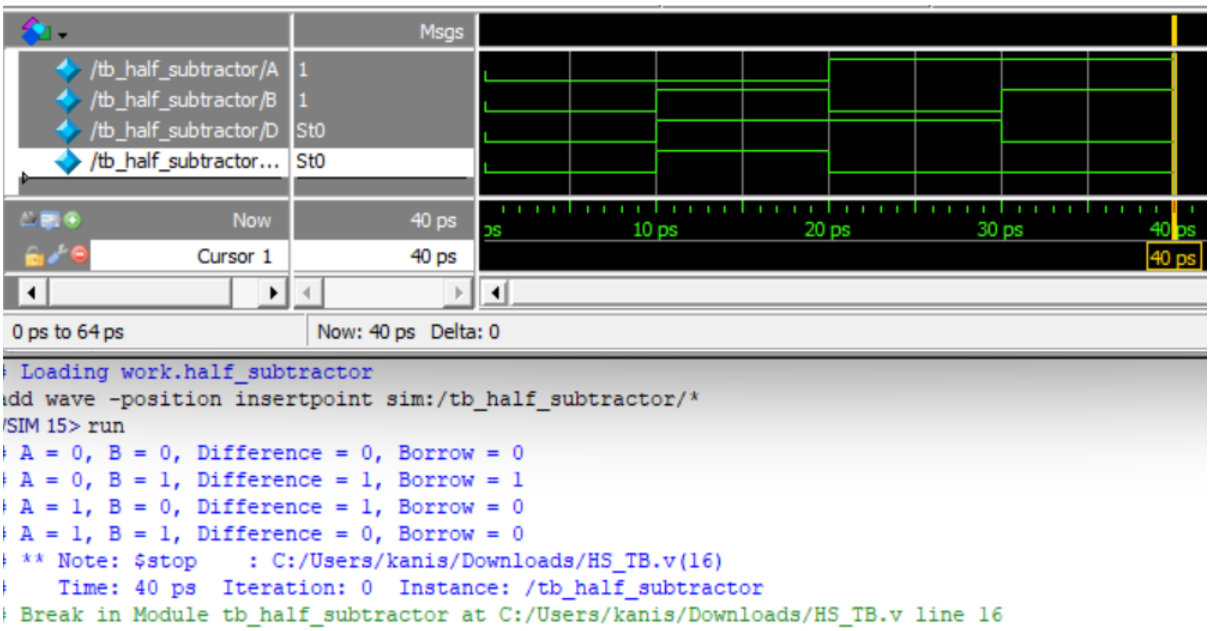
Design block-

```
Ln#
1   module half_subtractor(
2       input A,
3       input B,
4       output D,
5       output B_out
6   );
7       assign D = A ^ B;
8       assign B_out = ~A & B;
9   endmodule
10
11
```

Test bench-

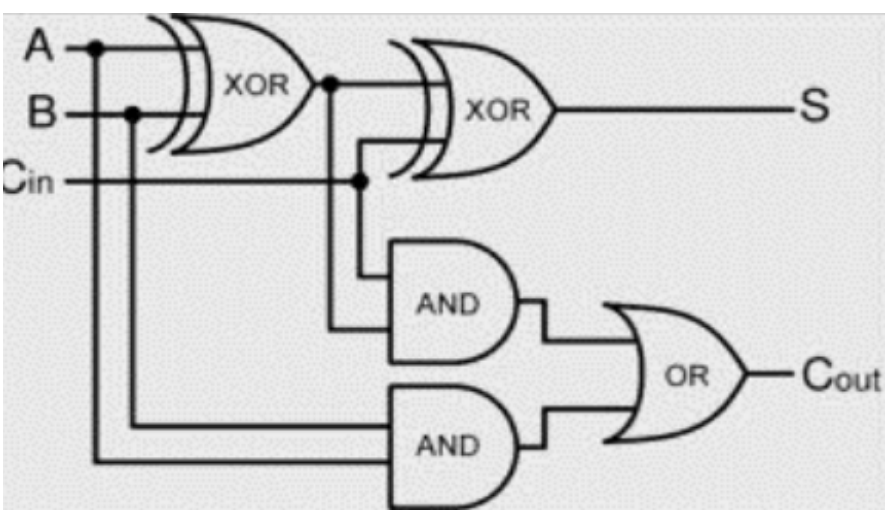
```
Ln#
1   module tb_half_subtractor;
2       reg A, B;
3       wire D, B_out;
4       half_subtractor uut (
5           .A(A),
6           .B(B),
7           .D(D),
8           .B_out(B_out)
9       );
10      initial begin
11          $monitor("A = %b, B = %b, Difference = %b, Borrow = %b", A, B, D, B_out);
12          A = 0; B = 0; #10;
13          A = 0; B = 1; #10;
14          A = 1; B = 0; #10;
15          A = 1; B = 1; #10;
16          $stop;
17      end
18  endmodule
19
20
```

iii)Output:



- Full adder:

i)circuit diagram:



ii)code:

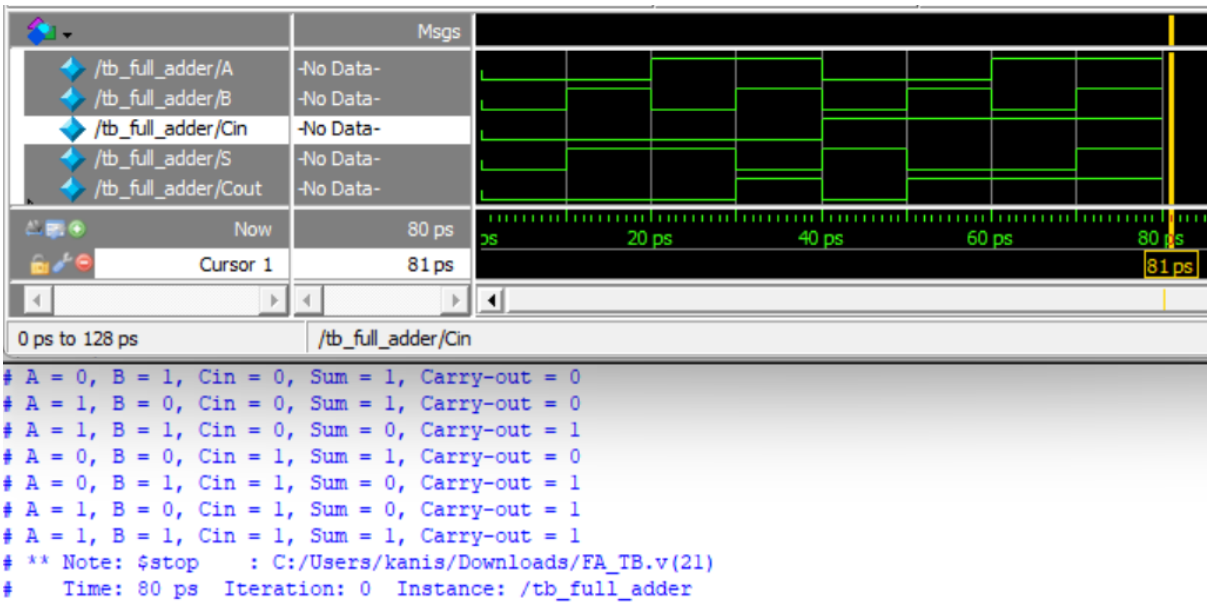
Design block-

```
module full_adder(
    input A,
    input B,
    input Cin,
    output S,
    output Cout
);
    assign S = A ^ B ^ Cin;
    assign Cout = (A & B) | (Cin & (A ^ B));
endmodule
```

Test bench-

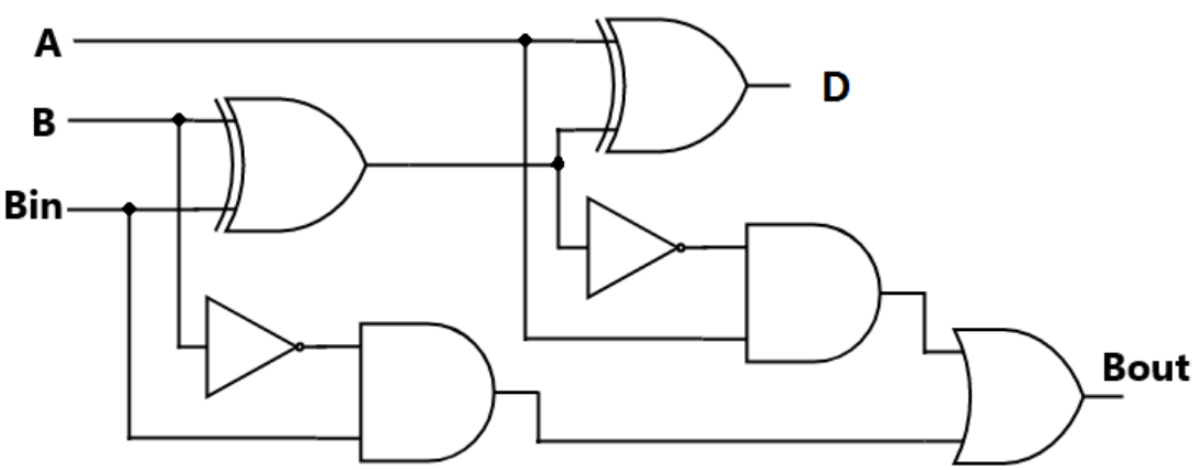
```
module tb_full_adder;
    reg A, B, Cin;
    wire S, Cout;
    full_adder uut (
        .A(A),
        .B(B),
        .Cin(Cin),
        .S(S),
        .Cout(Cout)
    );
    initial begin
        $monitor("A = %b, B = %b, Cin = %b, Sum = %b, Carry-out = %b", A, B, Cin, S, Cout);
        A = 0; B = 0; Cin = 0; #10;
        A = 0; B = 1; Cin = 0; #10;
        A = 1; B = 0; Cin = 0; #10;
        A = 1; B = 1; Cin = 0; #10;
        A = 0; B = 0; Cin = 1; #10;
        A = 0; B = 1; Cin = 1; #10;
        A = 1; B = 0; Cin = 1; #10;
        A = 1; B = 1; Cin = 1; #10;
        $stop;
    end
endmodule
```

iii)Output:



- Full subtractor:

i)circuit diagram:



iii)code:

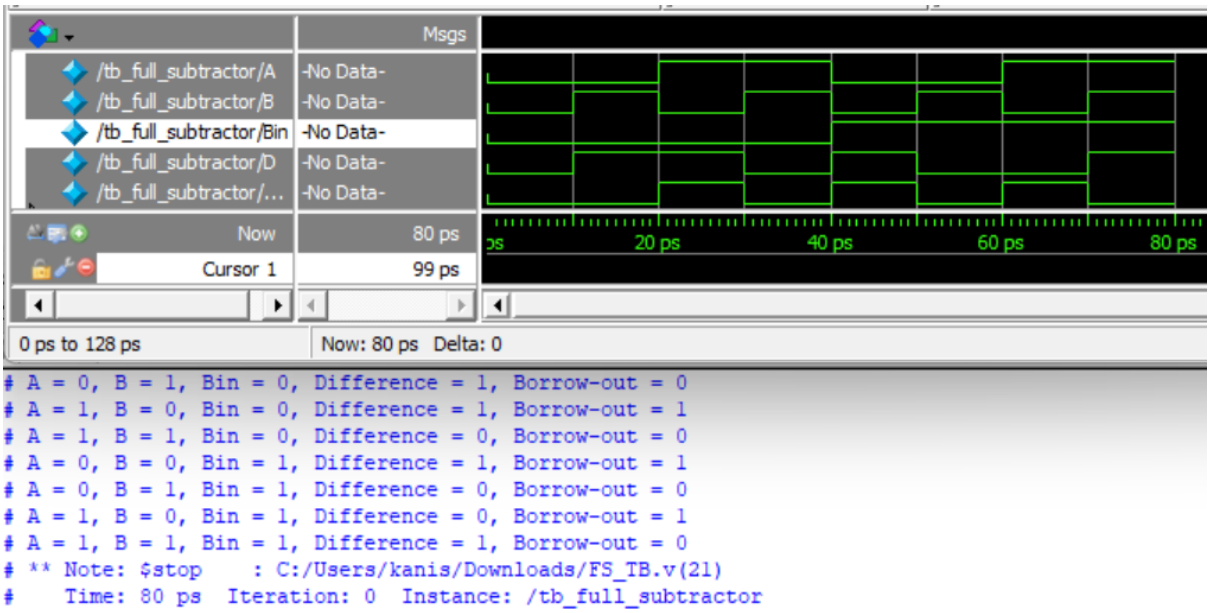
Design block-

```
module full_subtractor(
    input A,
    input B,
    input Bin,
    output D,
    output Bout
);
    assign D = A ^ B ^ Bin;
    assign Bout = (A &~ (B | Bin)) | (~B & Bin);
endmodule
```

Test bench-

```
module tb_full_subtractor;
    reg A, B, Bin;
    wire D, Bout;
    full_subtractor uut (
        .A(A),
        .B(B),
        .Bin(Bin),
        .D(D),
        .Bout(Bout)
    );
    initial begin
        $monitor("A = %b, B = %b, Bin = %b, Difference = %b, Borrow-out = %b", A, B, Bin, D, Bout);
        A = 0; B = 0; Bin = 0; #10;
        A = 0; B = 1; Bin = 0; #10;
        A = 1; B = 0; Bin = 0; #10;
        A = 1; B = 1; Bin = 0; #10;
        A = 0; B = 0; Bin = 1; #10;
        A = 0; B = 1; Bin = 1; #10;
        A = 1; B = 0; Bin = 1; #10;
        A = 1; B = 1; Bin = 1; #10;
        $stop;
    end
endmodule
```

iii)Output:



- **Result-**

  Hence the adders and subtractors have been implemented

Experiment – 3

Design of Decoders

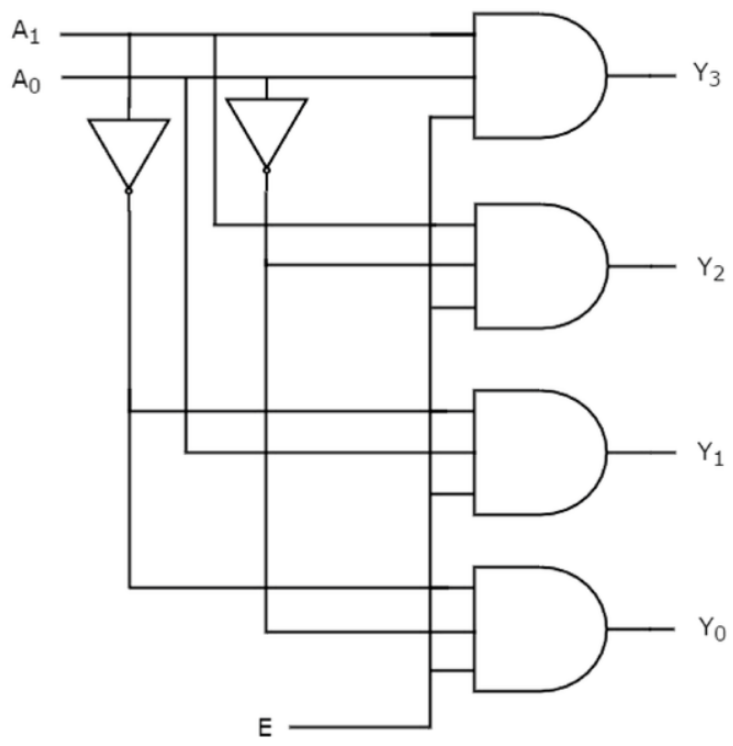
- **Aim:**

To design and implement Decoder circuit using VERILOG(Modelsim) and obtain the simulation.

- **Tools Required:** Model Sim Software.

2:4 Decoder:

i)Circuit Diagram:



ii)Code:

```
module decoder24_g(e,a,b,d);
input e,a,b;
output [3:0]d;
wire ne,na,nb;
not n0(enb,en);
not n1(na,a);
not n2(nb,b);
nand n3(y[0],ne,na,nb);
nand n4(y[1],ne,na,b);
nand n5(y[2],ne,a,nb);
nand n6(y[3],ne,a,b);
endmodule
```

Data Flow:

```
module decoder24_d(e,a,b,d);
input e,a,b;
output [3:0]d;
wire ne,na,nb;
assign ne = ~e;
assign na = ~a;
assign nb = ~b;
assign d[0] = ~(ne&na&nb);
assign d[1] = ~(ne&na&b);
assign d[2] = ~(ne&a&nb);
assign d[3] = ~(ne&a&b);
endmodule
```

Behavioural modelling:

```
Ln#
1   module decoder24_b(e,a,b,d);
2   input e,a,b;
3   output reg [3:0]d;
4   always @(e,a,b)
5       begin
6       if(e==0)
7           begin
8             if(a==1'b0 & b==1'b0) d=4'b1110;
9             else if(a==1'b0 & b==1'b1) d=4'b1101;
10            else if(a==1'b1 & b==1'b0) d=4'b1011;
11            else if(a==1 & b==1) d=4'b0111;
12            else d=4'bxxxx;
13          end
14      else
15        d=4'b1111;
16    end
17  endmodule
18
19
```

Test Bench:

```
Ln#
1   module decoder24_g_tb;
2   reg a,b,e;
3   wire [3:0]d;
4   decoder24_g dut(e,a,b,d);
5     initial
6       begin
7         e=1;a=1'bx;b=1'bx;
8   #5 e=0;a=0;b=0;
9   #5 e=0;a=0;b=1;
10  #5 e=0;a=1;b=0;
11  #5 e=0;a=1;b=1;
12  #5 $stop;
13      end
14  endmodule
15
16
```

iii)Truth Table:

A1	A0	Y0	Y1	Y2	Y3
0	0	D	0	0	0
0	1	0	D	0	0
1	0	0	0	D	0
1	1	0	0	0	D

- Output:

```
VSIM 2> run -all
# Time=0 | e=0 a=0 b=0 => d=1110
# Time=10000 | e=0 a=0 b=1 => d=1101
# Time=20000 | e=0 a=1 b=0 => d=1011
# Time=30000 | e=0 a=1 b=1 => d=0111
# Time=40000 | e=1 a=0 b=0 => d=1111
# Time=50000 | e=1 a=0 b=1 => d=1111
# Time=60000 | e=1 a=1 b=0 => d=1111
# Time=70000 | e=1 a=1 b=1 => d=1111
# ** Note: $finish    : C:/Modelsim_Everything/Decoder_Tb.v(16)
#    Time: 80 ns  Iteration: 0  Instance: /Decoder_tb
```

- <u>Result:</u>

Hence decoder is designed.

Experiment – 4

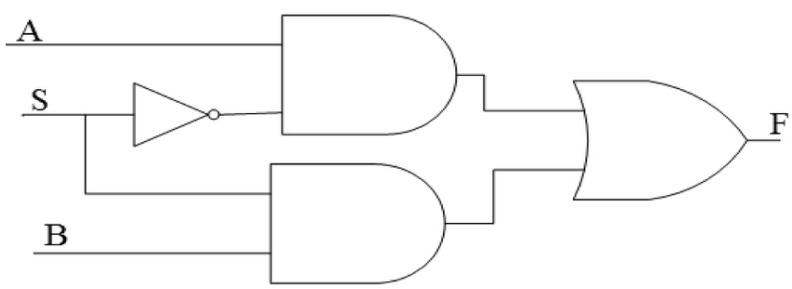
Design and Implementation of Multiplexers

- AIM:

To verify and study the truth table of 2:1 multiplexer circuit using modelism

- SOFTWARE REQUIRED: MODELISM

- CIRCUIT DIAGRAM:



- CODE:

GATE LEVEL MODELING:

```
Ln#
1   module mux2to1(sel, i1, i0, f);
2   i0, i1, sel;
3   output f;
4   wire nsel, w1, w2;
5   not(nsel, sel);
6   and(w1, i0, nsel);
7   and(w2, i1, sel);
8   or(f, w1, w2);
9   Endmodule
10
11
```

TEST BENCH:

```
Ln#
1   module test_mux2to1;
2     reg s, a1, a0;
3     wire q;
4     mux2to1 mux(.sel(s), .i1(a1), .i0(a0), .f(q));
5     initial begin
6       $monitor("sel = %b: i0 = %b, i1 = %b --> f = %b", s, a0, a1, q);
7       s = 1'b0; a1=1'b0; a0=1'b0;
8       #10
9       s = 1'b0; a1=1'b0; a0=1'b1;
10      #10
11      s = 1'b0; a1=1'b1; a0=1'b0;
12      #10
13      s = 1'b0; a1=1'b1; a0=1'b1;
14      #10
15      s = 1'b1; a1=1'b0; a0=1'b0;
16      #10
17      s = 1'b1; a1=1'b0; a0=1'b1;
18      #10
19      s = 1'b1; a1=1'b1; a0=1'b0;
20      #10
21      s = 1'b1; a1=1'b1; a0=1'b1;
22    end
23  endmodule
24
25
```

DATA FLOW: module mux2to1

```
Ln#
1   input i0,i1,sel,
2   output f);
3   assign f = (~sel & i0)|(sel & i1);
4   endmodule
5
6
```

```
add wave -position insertpoint sim:/test_2to1/*
VSIM 7> run
# sel = 0: i0 = 0: i1=0 --> f = x
# sel = 0: i0 = 1: i1=0 --> f = x
# sel = 0: i0 = 0: i1=1 --> f = x
# sel = 0: i0 = 1: i1=1 --> f = x
# sel = 1: i0 = 0: i1=0 --> f = 0
# sel = 1: i0 = 1: i1=0 --> f = 0
# sel = 1: i0 = 0: i1=1 --> f = 1
# sel = 1: i0 = 1: i1=1 --> f = 1

VSIM 8>
```

TRUTH TABLE:

SELECT(S)	INPUT(A)	INPUT(B)	OUTPUT(F)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

- RESULT: The multiplexer was successfully designed and implemented in ModelSim using basic logic gates, and its functionality was verified with the truth table.

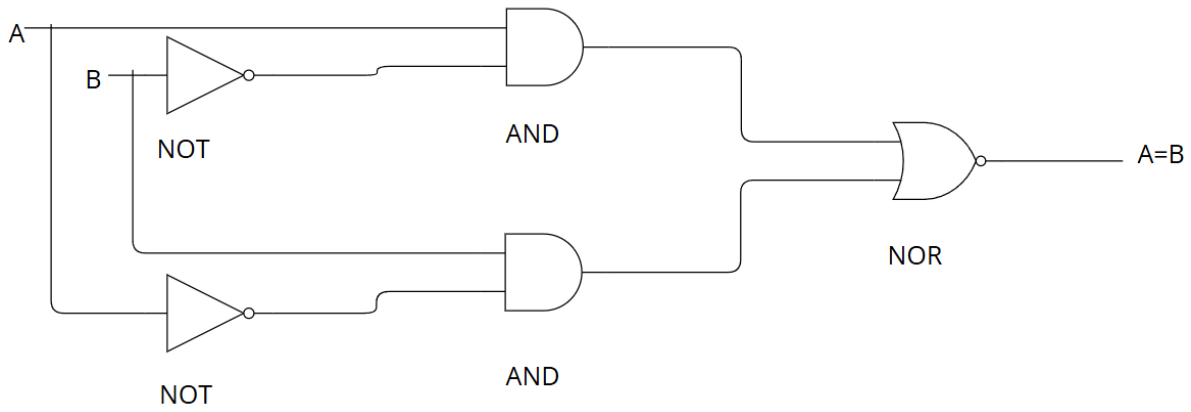
Experiment – 5

1(Bit) magnitude comparator

- Aim- To verify the 1(Bit) magnitude comparator using Modelsim software.

- Tools required- Modelsim software

- Circuit Diagram-



- Code-

Design block code-

```
Ln#
1   module Mag_Comp(a, b, o1, o2, o3);
2   input a, b;
3   output o1, o2, o3;
4   assign o1 = a &(~b);
5   assign o2 = b &(~a);
6   assign o3 = ~ (o1 | o2);
7   endmodule
8
9
```

Test bench-

```
Ln#
1   module tb_Mag_Comp;
2   reg a, b;
3   wire o1, o2, o3;
4   Mag_Comp uut (.a(a),.b(b),.o1(o1),.o2(o2), .o3(o3));
5   initial begin
6                   $display("Time\tA B | A>B A<B A=B");
7                    $display("------------------------");
8                    a = 0; b = 0; #10;
9                   $display("%0t\t%0d %0d | %0d %0d %0d", $time, a, b, o1, o2, o3);
10                  a = 0; b = 1; #10;
11                  $display("%0t\t%0d %0d | %0d %0d %0d", $time, a, b, o1, o2, o3);
12                  a = 1; b = 0; #10;
13                  $display("%0t\t%0d %0d | %0d %0d %0d", $time, a, b, o1, o2, o3);
14                  a = 1; b = 1; #10;
15                  $display("%0t\t%0d %0d | %0d %0d %0d", $time, a, b, o1, o2, o3);
16                  $finish;
17  end
18  endmodule
19
20
```

- Truth Table-

A	B	A>B	A=B	A<B
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

- Output-

```
ModelSim> vsim -gui work.tb_magnitude_comparator
# vsim -gui work.tb_magnitude_comparator
# Start time: 08:56:32 on Feb 13,2025
# Loading work.tb_magnitude_comparator
# Loading work.magnitude_comparator
VSIM 2> run
# A = 0, B = 0, A > B = 0, A = B = 1, A < B = 0
# A = 0, B = 1, A > B = 0, A = B = 0, A < B = 1
# A = 1, B = 0, A > B = 1, A = B = 0, A < B = 0
# A = 1, B = 1, A > B = 0, A = B = 1, A < B = 0
# ** Note: $finish    : E:/24BCE5399/mag_comp_tb.v(43)
#    Time: 40 ps  Iteration: 0  Instance: /tb_magnitude_comparator
# 1
# Break in Module tb_magnitude_comparator at E:/24BCE5399/mag_comp_tb.v line 43
```

- Result-

Thus the 1(Bit) magnitude comparator has been verified.