Lab 3
Combinational Circuits

Objective

Design, model, and simulate the procedural descriptions for combinational blocks such as a multiplexer, decoder, and comparator, ... At the end of this lab, the trainee will build and implement a complete arithmetic logic unit on the FPGA development board.

Discussion

1. Multiplexer

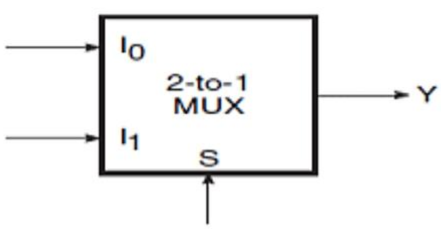
A multiplexer device has multiple inputs and a single-line output. The select lines determine which input is connected to the output, and also to increase the amount of data that can be sent over a network within a certain time. It is also called a data selector.

1.1 2x1 Multiplexer

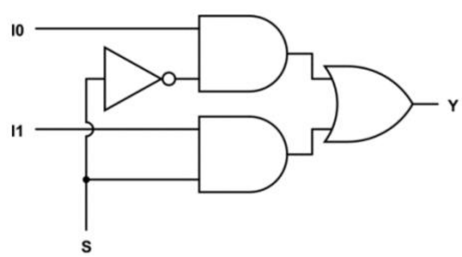
In 2×1 multiplexer, there are only two inputs, i.e., I_0 and I_1, one selection line, i.e., S , and single outputs, i.e., Y. Based on the combination of inputs that are present at the selection line S^0, one of these 2 inputs will be connected to the output. The block diagram of the 2×1 multiplexer is given below.

The boolean expression of the Full 2x1 MUX:

$$\mathbf{Y=S`\,I_0+S\,I_1}$$



Logic Symbol of 2x1 Multiplexer



Circuit Diagram of 2x1 Multiplexer

Code 1: Structural Verilog model for 2x1 Multiplexer

```
module Mux2t01_gatelevel (I0, I1, select, Y);

 input I0;
 input I1;
 input select;
 output Y;
 wire w1;
 wire w2;

 and g1(w1, I0, select);
 and g2(w2, I1, ~select);
 or g3(Y, w1, w2);
endmodule
```

Code 2: Data flow Verilog model for 2x1 Multiplexer

```
module mux2to1_data_flow (I0, I1, select, Y);

 input I0;
 input I1;
 input select;
 output Y;

 assign Y= (~select & I0) | (select & I1);
endmodule
```

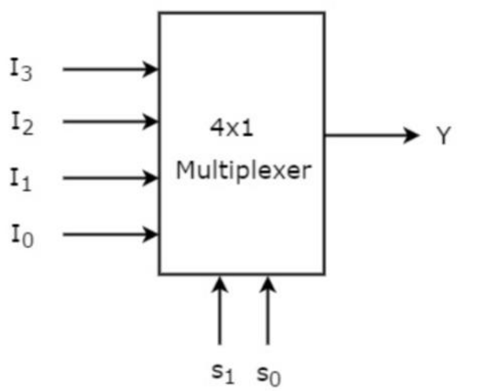
Code 3: Behavioral Verilog model for 2x1 Multiplexer

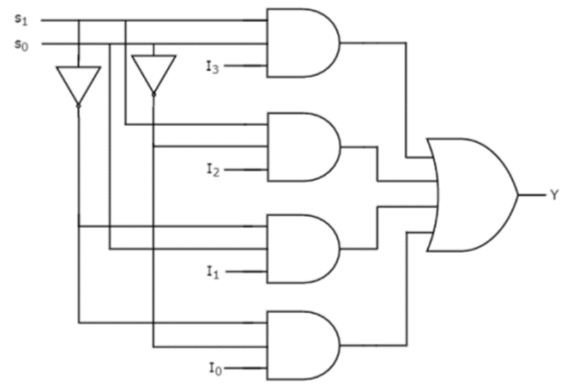
```
module mux2to1_behavioral (I0, I1, select, Y);

 input I0;
 input I1;
 input select;
 output reg Y;
 always @ (*)
 begin
  if (select==0)
     Y=I0;
  else if (select==1)
      Y=I1;
 end
endmodule
```

1.2 4x1 Multiplexer

4x1 Multiplexer has four data inputs I0, I1, I2 & I3, two selection lines S0 & S1 and one output Y. The block diagram of 4x1 Multiplexer is shown in the following figure. One of these 4 inputs will be connected to the output based on the combination of inputs present at these two selection lines. The boolean expression of the 4x1 MUX:

$$Y = \grave{S}_1\grave{S}_0 I_0 + \grave{S}_1 S_0 I_1 + S_1 \grave{S}_0 I_2 + S_1 S_0 I_3$$





Logic symbol of 4x1 Multiplexer **Block diagram of 4x1 Multiplexer**

Code 4: Gate/Structural Verilog model for 4x1 Multiplexer

```
module mux4to1_structural (I, S, Y);


input [3:0] I;
input [1:0] S;
output Y;


wire w1;
wire w2;
wire w3;
wire w4;

and g1(w1, ~S[1], ~S[0], I[0]);
and g2(w2, ~S[1], S[0], I[1]);
and g3(w3, S[1], ~S[0], I[2]);
and g4(w4, S[1], S[0], I[3]);
or g5(Y, w1, w2, w3, w4);

endmodule
```

Code 5: Data flow Verilog model for 4x1 Multiplexer

```
module mux4to1_dataflow (I, S, Y);
 input [3:0] I;
 input [1:0] S;
 output Y;
 assign Y=(~S[1] & ~S[0] & I[0]) | (~S[1] & S[0] & I[1]) | (S[1] & ~S[0] &
I[2]) | (S[1] & S[0] & I[3]);

endmodule
```

Code 6: behavioral Verilog model for 4x1 Multiplexer

```
module mux4to1_behavioral (I, S, Y);
 input [3:0] I;
 input [1:0] S;
 output reg Y;

 always @ (I or S)
   begin
     case (S)
       2`b00 : Y=I[0];
       2`b01 : Y=I[1];
       2`b10 : Y=I[2];
       2`b11 : Y=I[3];
     endcase
   end
endmodule
```

2. Decoder

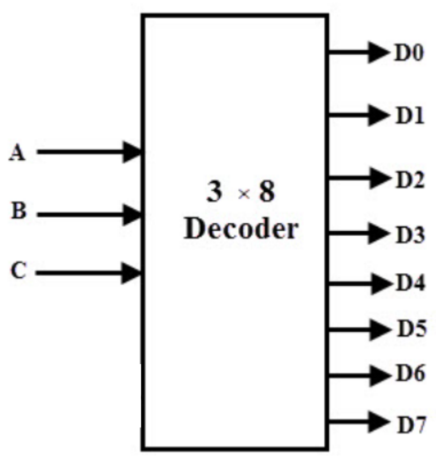
2.1 3 to 8 decoder

A 3 to 8 decoder has three inputs (A, B, C) and eight outputs (D0 to D7). Based on the 3 inputs one of the eight outputs is selected.

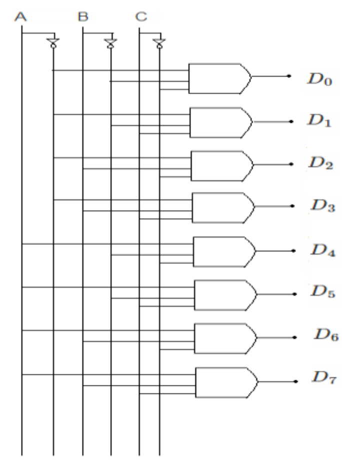
we can write the Boolean functions for each output as

$$D_0 = A`B`C` \qquad D_1 = A`B`C \qquad D_2 = A`BC` \qquad D_3 = A`BC$$

$$D_4 = AB`C` \qquad D_5 = AB`C \qquad D_6 = ABC` \qquad D_7 = ABC$$



Logic symbol of 3X8 Decoder



Block diagram of 3x8 Decoder

Code 7: Data flow Verilog model for 3x8 Decoder

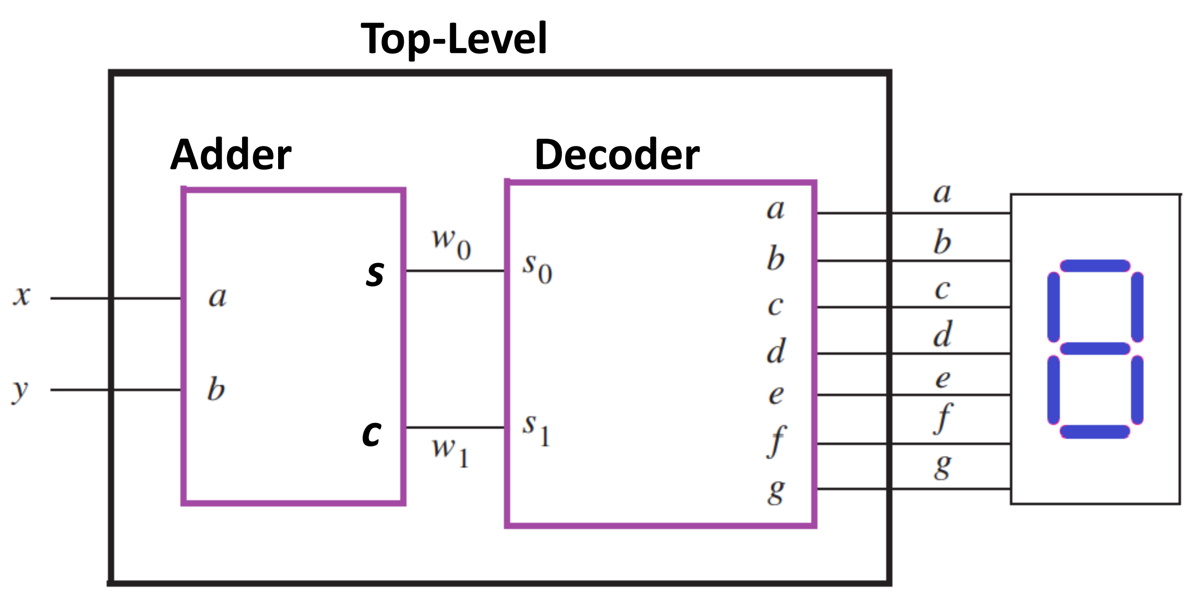
```
module decoder_data_flow (A, B, C, D);

input A;
input B;
input C;
output [7:0]D;

assign D[0]=~A & ~B & ~C;
assign D[1]=~A & ~B &C;
assign D[2]=~A & B & ~C;
assign D[3]=~A & B & C;
assign D[4]=A & ~B & ~C;
assign D[5]=A & ~B & C;
assign D[6]=A & B & ~C;
assign D[7]=A & B & C;

endmodule
```

Code 8: Behavioral Verilog model for 3x8 Decoder

```
module decoder behavioral (I, D);

 input [2:0] I;
 output reg [7:0] D;
always @ (I)
 begin
   case (I)
    3`b000: D=8`b10000000;
    3`b001: D=8`b01000000;
    3`b010: D=8`b00100000;
    3`b011: D=8`b00010000;
    3`b100: D=8`b00001000;
    3`b101: D=8`b00000100;
    3`b110: D=8`b00000010;
    3`b111: D=8`b00000001;
   endcase
 end
endmodule
```

3.Display_Adder

The purpose of the circuit is to generate the arithmetic sum of the two inputs x and y, using the adder module, and then to show the resulting decimal value using the decoder module on the 7-segment display.



a.adder_module

```
module adderx (a, b, s, c);
        input a, b;
        output s, c;

        assign s = a ^ b;   // sum using xor
        assign c =  a & b;  // carry
endmodule
```

b.decoderx_module

```
module decoderx (s0, s1, a, b, c, d, e, f, g);
input s0, s1;
output a, b, c, d, e, f, g;

        assign a = ~s0;
        assign b = 1;
        assign c = ~s1;
        assign d = ~s0;
        assign e = ~s0;
        assign f = ~s1 & ~s0;
        assign g = s1 & ~s0;
endmodule
```

c.Top_module

```
module Display_adder (x, y, a, b, c, d, e, f, g);
    input x, y;
    output a, b, c, d, e, f, g;
    wire w0, w1;

    adderx U1 (x, y, w0, w1);
    decoderx U2 (w0, w1, a, b, c, d, e, f, g);
endmodule
```

4.Comparator

8-bit Magnitude Comparator

An 8-bit magnitude comparator compares the two 8-bit values and produce a 1-bit flag as result, which indicates that the first value is either greater than or less than or equal to the second value. The block diagram of a comparator is shown in Figure 1.

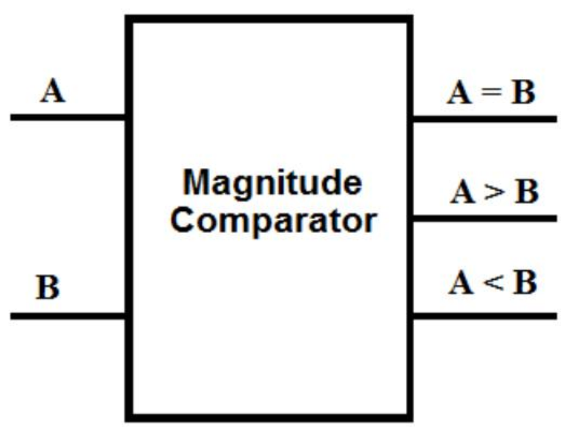


Figure 1. Block Diagram of Magnitude Comparator

Code 10: Verilog Code for 8-bit Magnitude Comparator

```
module magcomp (A,B,Gt,Lt,Eq);
  input [7:0]A,B; //The two 8-bit Inputs A and B
  output   Gt,Lt,Eq; //The Outputs of comparison
  reg   Gt, Lt, Eq;
always @ (A or B) //Check the state of the input lines
    begin
     Gt <= ( A > B )? 1'b1 : 1'b0;
     Lt <= ( A < B )? 1'b1 : 1'b0;
     Eq <= ( A == B)? 1'b1 : 1'b0;
    end
endmodule
```

Assignment:

Write structural, and behavioral models for the following:

4-bit Comparator using subtractor.

Has three outputs denote the following:

Z = 1 if the result is 0. (zero flag)

N = 1 if the result is negative. (negative flag)

V = 1 if arithmetic overflow occurs. (overflow flag)

Bonus :

Write structural, and behavioral models for the following:

BCD Adder.

Deliverables

Verilog codes for all designs with screenshots for simulated waveform including all test cases. Also, screenshots for the RTL and synthesis results of all designs ad RTL netlist viewer.