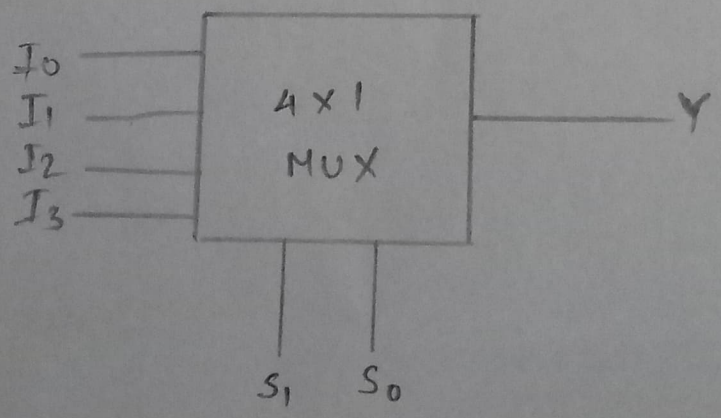
Name- MOITRISH MAITY Roll-48 CSE 2nd Year

EXPERIMENT-4

- **Title**: Combinational circuits
- **Objective**: Implementation of combinational circuit
- **Course Outcome**: CO2
- **Bloom's Level**: Evaluation.

(4.1) 4x1 MUX

- **Problem Statement**: Write a verilog program that implements Multiplexer module (4x1 MUX).
- **Theory**: MUX is a combinational logic circuit designed to switch one of several input lines through a single common output line by appliation of a control signal. It has maximum of 2^n data inputs 'n' selection lines and a single output line. One of these data inputs will be connected to the output line based on the values of the selection lines. Here, we are implementing a 4x1 MUX using Verilog.
- **Logic Diagram**:



Truth Table:

S_1	S_0	Y
L	L	I_0
L	H	I_1
H	L	I_2
H	H	I_3

$S_1, S_0 \rightarrow$ Select lines

$Y \rightarrow$ Output

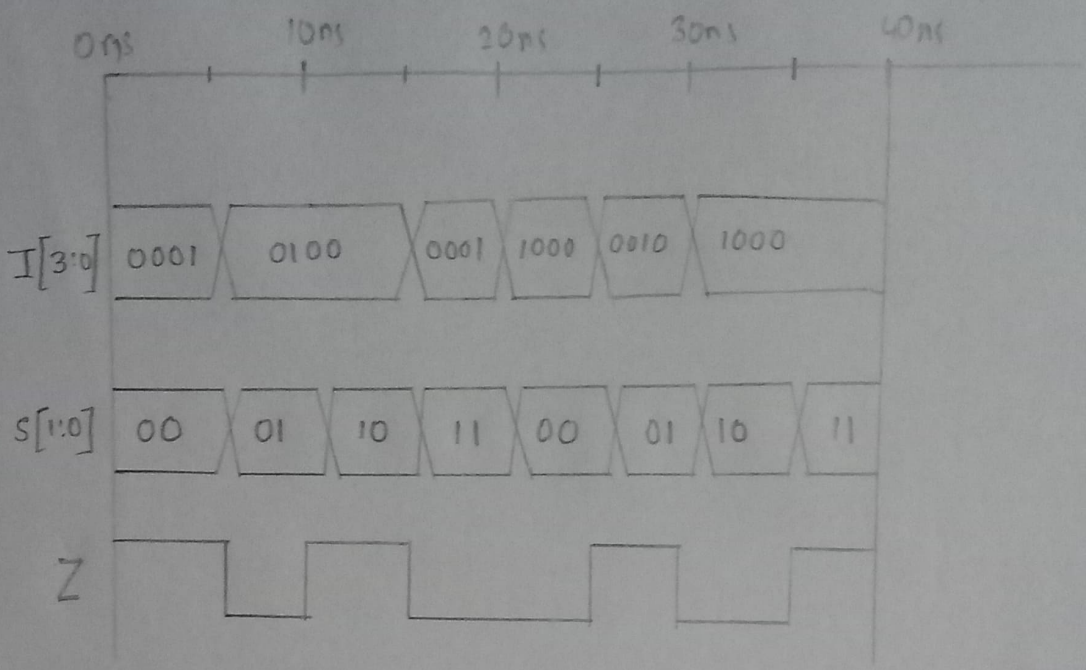
Design Code:

```
module four-mux (input [3:0]i, input[1:0]s, output y);
    assign y=s[1]?(s[0]?a[3]:a[2]):(s[0]?a[1]:a[0]);
endmodule
```

Testbench Code:

```
module four-mux-tb();
    reg [1:0]S; reg[3:0]I;
    wire Y;
    four-mux uut(.s(S),.i(I),.y(Y));
      initial begin;
      $dumpfile("dump.vcd"); $dumpvars;
      S=2'b00; I=4'b0001; #5;
      S=2'b01; I=4'b0100; #5;
      S=2'b10; I=4'b0100; #5;
      S=2'b11; I=4'b0001; #5;
      S=2'b00; I=4'b1000; #5;
      S=2'b01; I=4'b0010; #5;
      S=2'b10; I=4'b1000; #5;
      S=2'b11; I=4'b1000; #5;
      end
endmodule
```

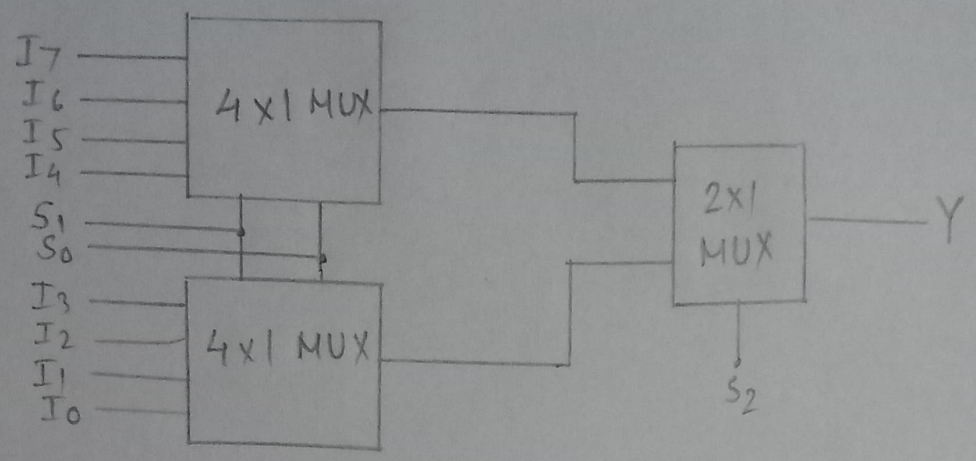
- Timing Diagram :



4.2 8 X 1 MUX:

- Problem Statement : Write a verilog program to include previous module (4X1 MUX) to describe functionality of 8X1 MUX circuit.
- Theory: MUX is a combinational logic circuit designed to switch one of several input lines through a single common output line by applications of a control signal. Here, we are implementing a 8 X1 MUX by using two 4X1 MUX and one 2 X 1 MUX.

Logic Diagram:



Truth Table:

S_2	S_1	S_0	Y
L	L	L	I_0
L	L	H	I_1
L	H	L	I_2
L	H	H	I_3
H	L	L	I_4
H	L	H	I_5
H	H	L	I_6
H	H	H	I_7

$S_2 S_1 S_0$ → Select Lines

Y → output

Design Code:

```
module four_mux (input [3:0] i, input [1:0] s, output y);
    assign y = s[1] ? ( s[0] ? a[3] : a[2]) : (s[0] ? a[1] : a[0]);
endmodule
module two_mux ( input a, b, s , output y);
    assign y = s ? b : a;
endmodule
```

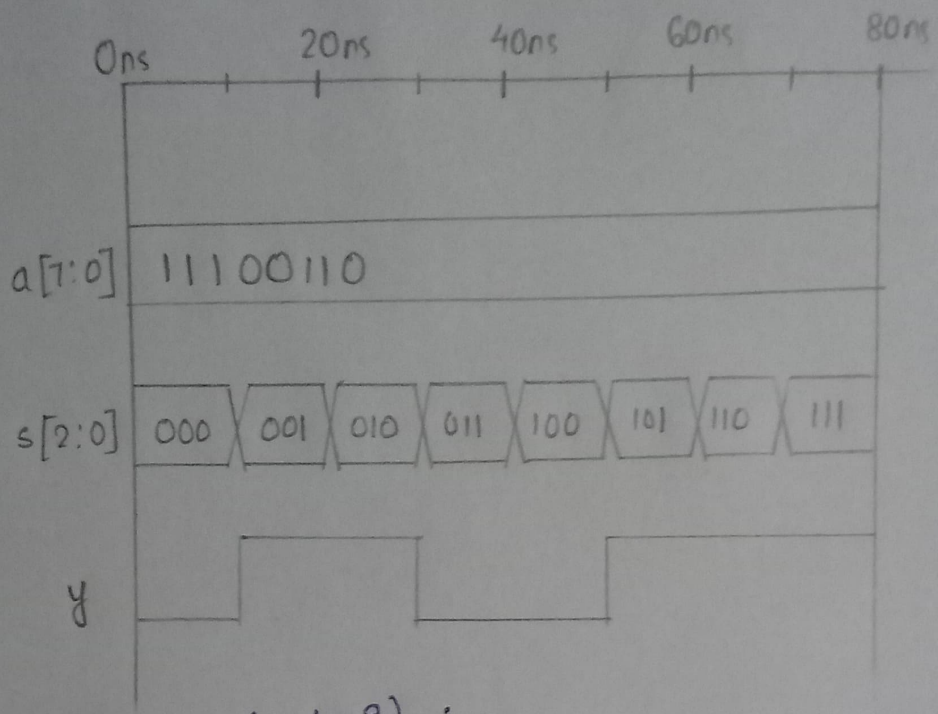
- Testbench Code :

```
module eight-mux-tb();
    reg [7:0]a;
    reg [2:0]s;
    wire y1, y2;

four-mux uut1(.a(a[3:0]), .s(s[1:0]), .y(y1));
four-mux uut2(.a(a[7:4]), .s(s[1:0]), .y(y2));
two-mux uut3(.a(y1), .b(y2), .y(y), .s(s[2]));
  initial begin;
  $dumpfile("dump.vcd");
  $dumpvars;
  a = 10011110;
  s = 000 ; #10;
  s = 001 ; #10;
  s = 010 ; #10;
  s = 011 ; #10;
  s = 100 ; #10;
  s = 101 ; #10;
  s = 110 ; #10;
  s = 111 ; #10;
  end

endmodule.
```

Timing Diagram:

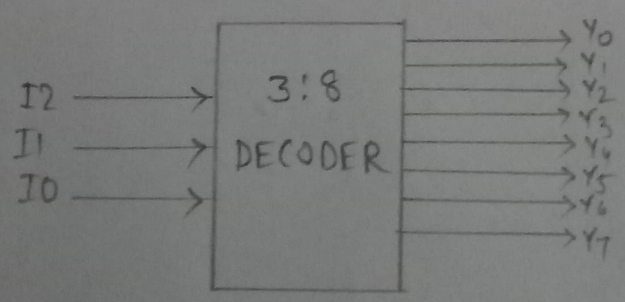


(4.3) Decoder (3 to 8):

Problem Statement: Write a verilog code to design, simulate and test a circuit of 3:8 decoder.

Theory: A decoder is a combinational logic circuit that converts an n-bit binary input code to 2^n output lines such that only one output line is activated for each one of the possible combinations of input. Here, we are trying to implement a 3:8 decoder using Verilog code.

Logic Diagram:



TRUTH Table :

I2	I1	I0	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
L	L	L	H	L	L	L	L	L	L	L
L	L	H	L	H	L	L	L	L	L	L
L	H	L	L	L	H	L	L	L	L	L
L	H	H	L	L	L	H	L	L	L	L
H	L	L	L	L	L	L	H	L	L	L
H	L	H	L	L	L	L	L	H	L	L
H	H	L	L	L	L	L	L	L	H	L
H	H	H	L	L	L	L	L	L	L	H

Design Code :

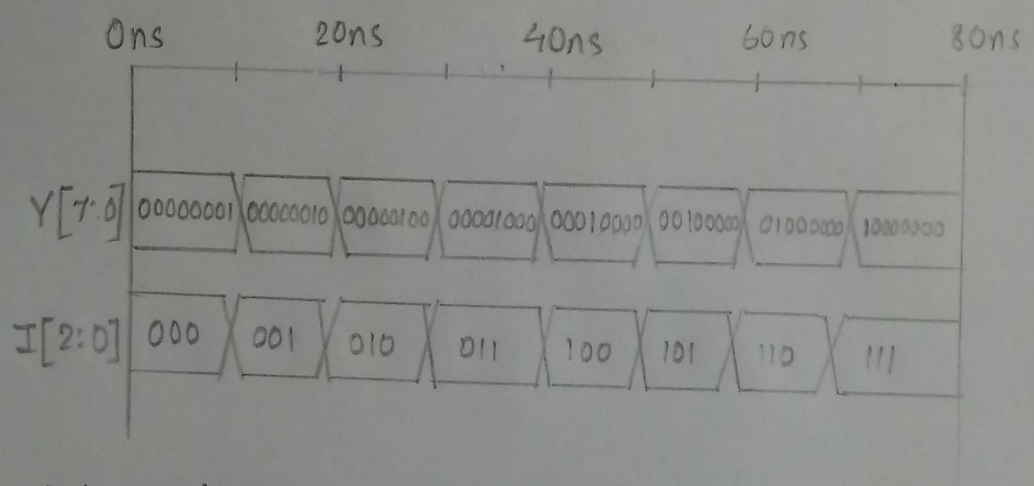
```
module decoder (input [2:0] i, output [7:0] y);
    assign y[0] = (~i[2]) & (~i[1]) & (~i[0]);
    assign y[1] = (~i[2]) & (~i[1]) & (i[0]);
    assign y[2] = (~i[2]) & (i[1]) & (~i[0]);
    assign y[3] = (~i[2]) & (i[1]) & (i[0]);
    assign y[4] = (i[2]) & (~i[1]) & (~i[0]);
    assign y[5] = (i[2]) & (~i[1]) & (i[0]);
    assign y[6] = (i[2]) & (i[1]) & (~i[0]);
    assign y[7] = (i[2]) & (i[1]) & (i[0]);
endmodule
```

Testbench Code :

```
module decoder_tb();
    reg [2:0] I;
    wire [7:0] Y;
decoder uut (.i(I), .y(Y));
    initial begin
        $dumpfile("dump.vcd"); $dumpvars;
```

```
    I= 3'b000 ; #10 ;
    I= 3'b001 ; #10 ;
    I= 3'b010 ; #10 ;
    I= 3'b011 ; #10 ;
    I= 3'b100 ; #10 ;
    I= 3'b101 ; #10 ;
    I= 3'b110 ; #10 ;
    I= 3'b111 ; #10 ;
    end
endmodule
```

Timing Diagram:



Discussion:

In this experiment, we have implemented various combinational circuits like 4x1 MUX, 8x1 MUX, 3:8 Decoder etc. We have also learned various coding terms in Verilog.

CO Justification: In this experiment, we have designed combinational circuits like decoders, multiplexers etc. So, CO2 is justified.

Questionairres:

(4.1) What is the difference between following 2 lines of verilog code?

```
#5a=b
a=#5b
```

Ans> #5a=b;

In this line, it waits 5 time units before doing the action for 'a=b'.

a=#5b;

In this line, the value of b is calculated and stored in an internal temporary register. Then it waits 5 time units and assign stored value to a.

(4.2) What do you mean by continuous assignment?

Ans> Continuous assignments are the most basic assignments in dataflow modelling. Continuous assignments are used to model in combinational logics. It drives values into the nets.

Example:

```
wire out;
assign out = In-A & In-B;
```

Continuous assignment 'out' is a net. Both In-A and In-B are nets.

(4.3) For the following verilog code statement:

```
wire [7:0]A;
wire B;
assign B=^A;
```

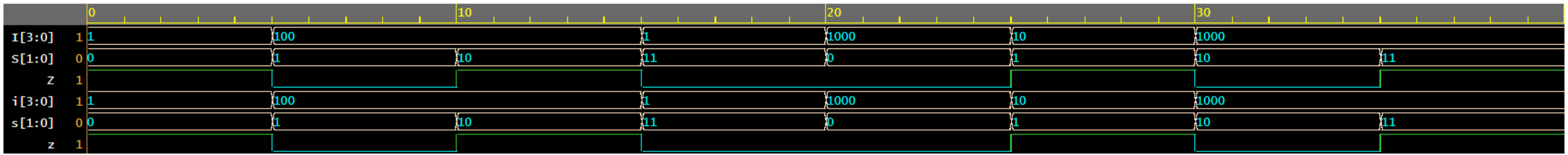
If the value of A is 16'b 10110011. What will be the value of {A[4:3], 3{B}}?

A (a) 5'b10111, as A[4:3]='10', B='1'.

EPWave

From: 0s To: 40s

Get Signals | Radix | 100% | 0s



Note: To revert to EPWave opening in a new browser window, set that option on your user page.

4x1 MUX

EPWave

From: 0s To: 80s

Get Signals | Radix | 100% | 0s

		0	10	20	30	40	50	60	70
D[7:0]	1	1	10	100	1000	10000	100000	1000000	10000000
I[2:0]	0	0	1	10	11	100	101	110	111
d[7:0]	1	1	10	100	1000	10000	100000	1000000	10000000
i[2:0]	0	0	1	10	11	100	101	110	111

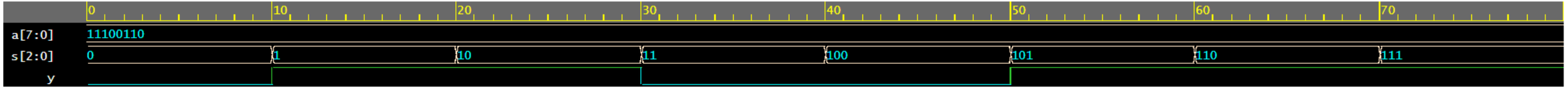
Note: To revert to EPWave opening in a new browser window, set that option on your user page.

3:8 DECODER

EPWave

From: 0s To: 80s

Get Signals | Radix | 100%



Note: To revert to EPWave opening in a new browser window, set that option on your user page.

8x1 MUX