CS220 Assignment 3

1. Sequence Detector (1010):

A sequential detector is a sequential state machine that takes an input string of bits and generates an output 1 whenever the target sequence (1010) is detected. It is implemented by constructing a Finite State Machine (FSM) which allows for overlapping too, i.e, the last bit of one sequence becomes first bit of the next sequence.

We are implementing it through a Mealy machine.

In a Mealy machine, output depends on the present state as well as the external input (X).

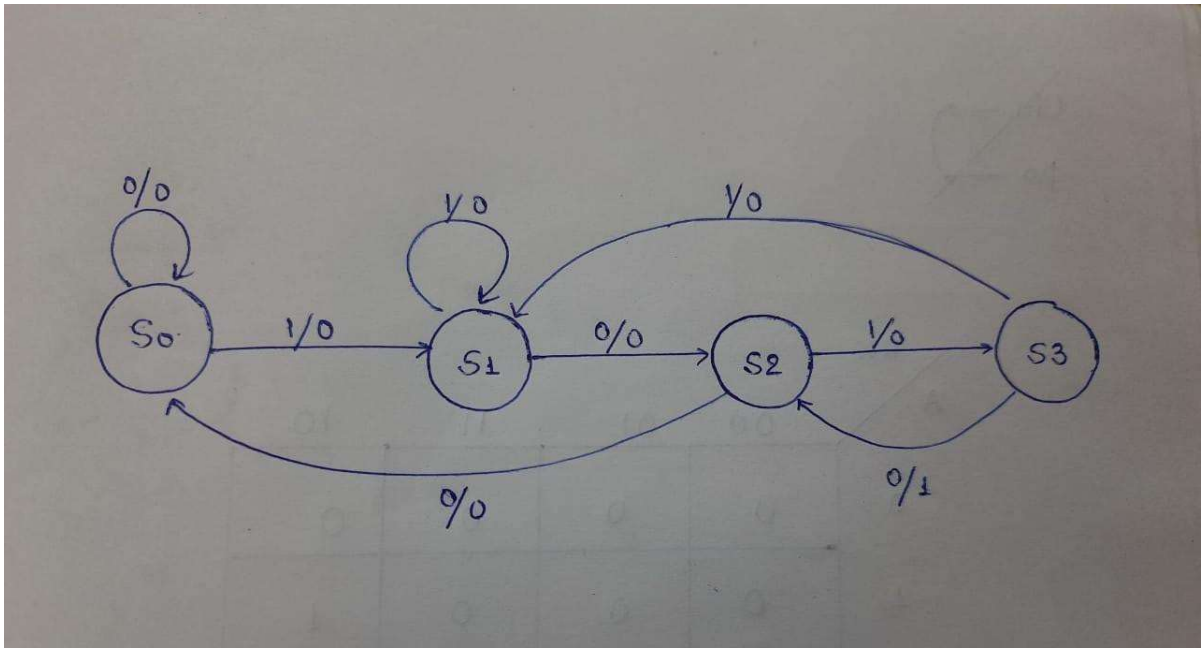
FSM construction of Sequence Detector :

- The FSM is implemented using three always blocks, one block for changing the state on getting input (combinational logic), one for changing the current state whenever the clock (clk) has positive edge (sequential logic) and one for giving output (output logic).
- Four states are defined, IDLE, state1, state2, and state3.

```
parameter IDLE = 2'b00,//initially
state1 = 2'b01, //state when it gets 10 from idle state
state2= 2'b10, //state when it gets 101
state3= 2'b11; //state when it gets 1010
```

- Initially, the clock is set 0 and reset is also set 0. Then 8-bit input is given which sends one-bit to the module seq_detector each time the edge becomes positive in the clock.
- Initially, the state is defined as IDLE and depending upon the input the next state is determined.
- As the state changes, an one-bit output is given. If sequence 1010 gets detected, output is given as 1 else 0.
- Again, when posedge arrives, the next 1-bit input is send, the state changes and the sequence is determined.

❖ State Diagram :



Here S0 is IDLE as stated above.

- Output Table :

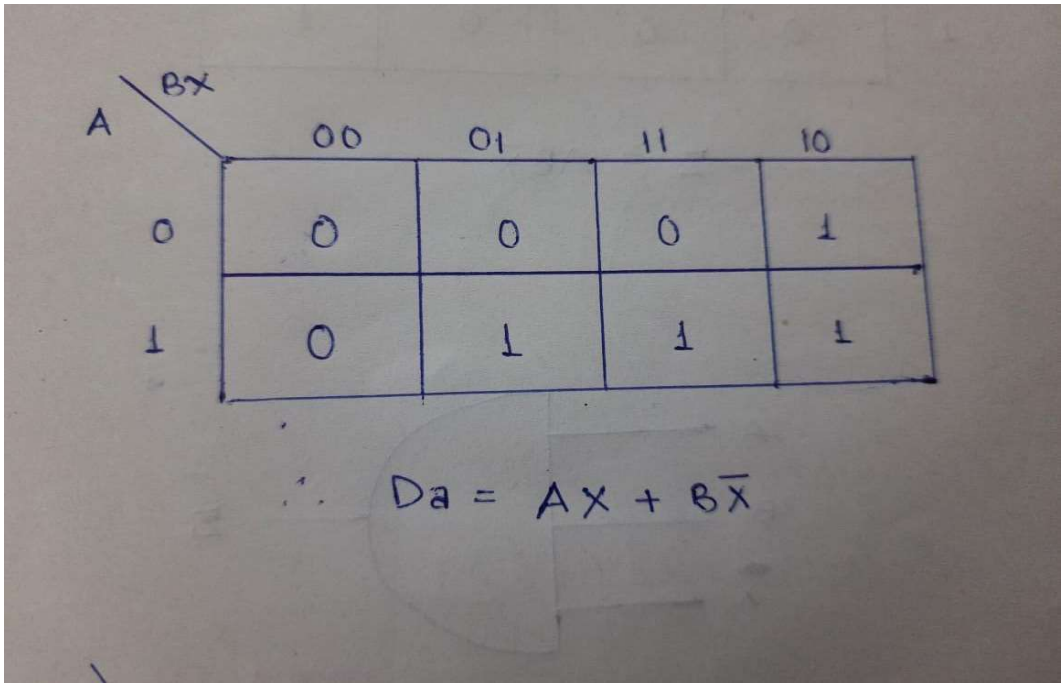
Present State	Next State		Output	
	X=0	X=1	X=0	X=1
00	00	01	0	0
01	10	01	0	0
10	00	11	0	0
11	10	11	1	0

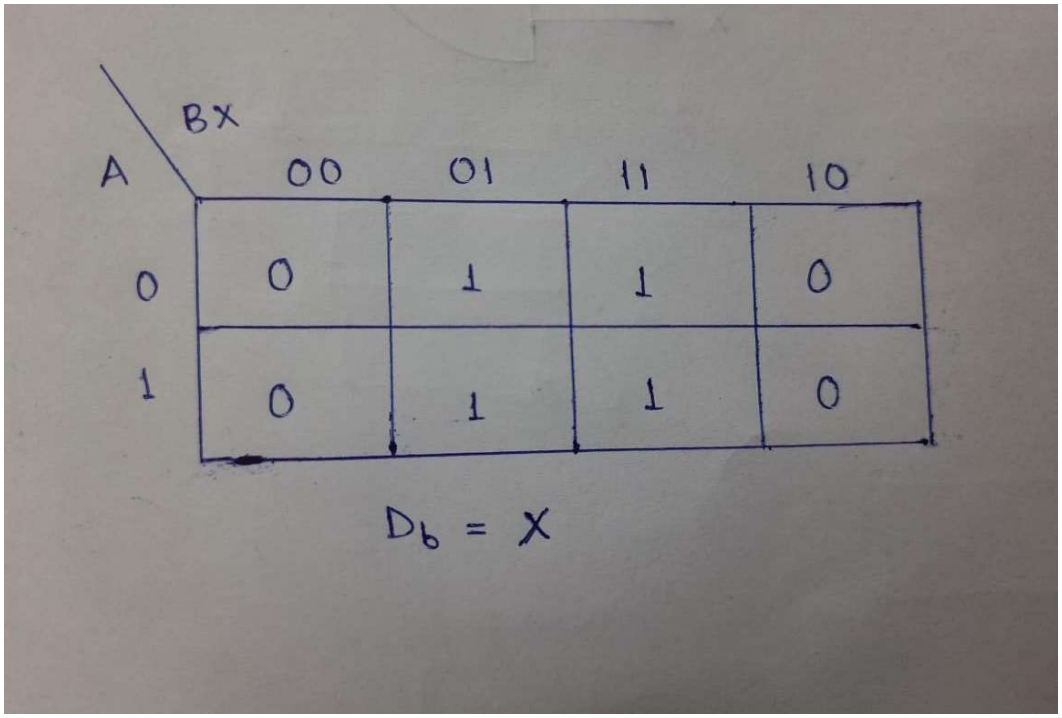
- Excitation Table :

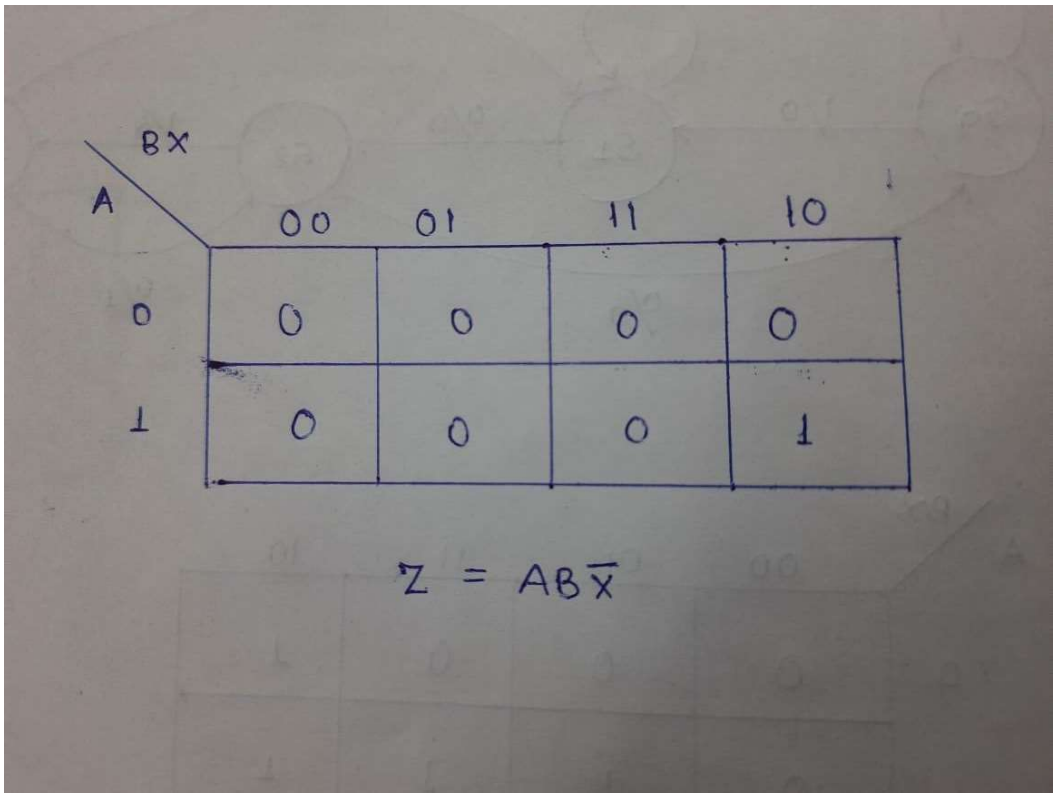
Present State		Input	Next State		Flip-flop excitations		Output
A	B	X	A’	B’	Da	Db	Z
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0
0	1	0	1	0	1	0	0

0	1	1	0	1	0	1	0
1	0	0	0	0	0	0	0
1	0	1	1	1	1	1	0
1	1	0	1	0	1	0	1
1	1	1	1	1	1	1	0

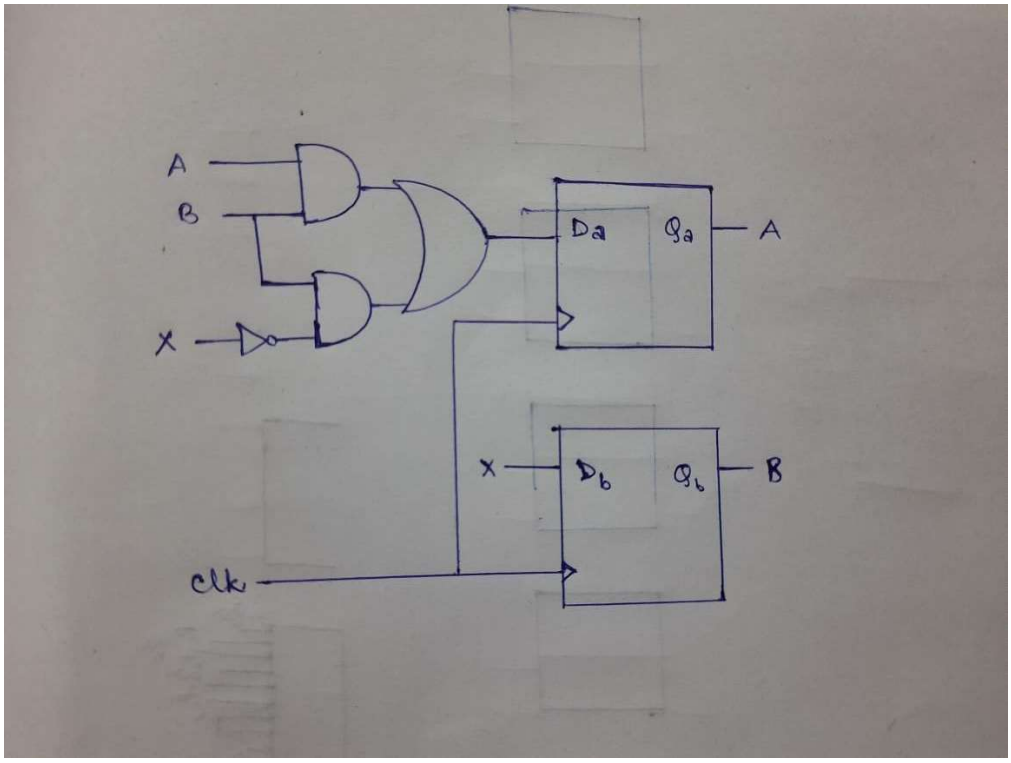
- K-map

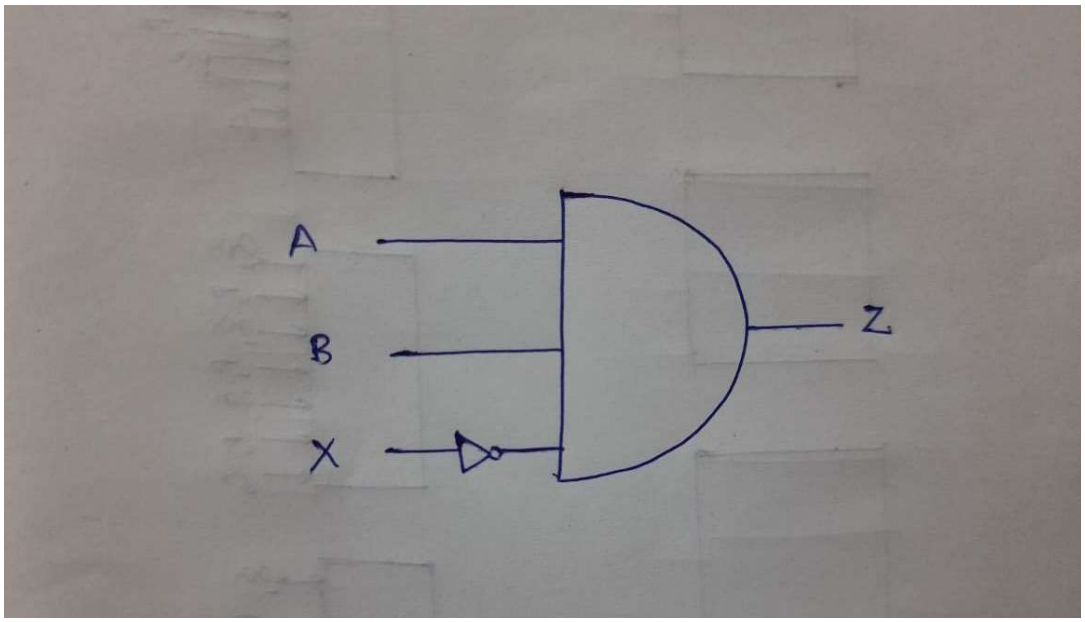






- Circuit / Logic Diagram





- Verilog Code

```
module seq_detector(clk,reset,in,out);

input clk,reset,in;
output out;
```

```
wire clk,reset;
wire in; //1-bit input
reg out; //1-bit output
parameter SIZE = 2; //setting parameter SIZE as 2

//setting parameters as different states
parameter IDLE = 2'b00,//initially
state1 = 2'b01, //state when it gets 10 from idle state
state2= 2'b10, //state when it gets 101
state3= 2'b11; //state when it gets 1010


reg [SIZE-1:0] state; //current state
reg [SIZE-1:0] next_state; //next state
initial begin
  state = IDLE;
end

always @(state or in) begin
    next_state= IDLE;
    case (state)

        //when it gets input after it is at IDLE state
        IDLE: if (~in) begin

            next_state= IDLE;
        end
        else begin
          next_state = state1;
        end

        //when it gets input after it is at state1
        state1: if(~in) begin

          next_state = state2;
        end
        else begin
          next_state= state1;
```

```
        end

        //when it gets input after it is at state2
        state2: if(~in) begin

          next_state = IDLE;
        end
        else begin
          next_state = state3;
        end

        //when it gets input after it is at state3
        state3: if(~in) begin
          next_state= state2;
        end
        else begin
          next_state = state1;
        end

        default:  begin
        next_state = IDLE;
        end


    endcase

end

always @(posedge clk) begin

    if(reset == 1'b1) begin

      state = IDLE; //new (8-bit) input
    end
    else begin

      state = next_state; //continuation of the input
    end

end
```

```
always @(state or in) begin

    if(reset == 1'b1) begin
      out = 1'b0;
    end

    else begin
      if(state == state3) begin
        if(in == 1'b0) begin
          out = 1'b1; //output when it finds the sequence of
1010
        end
        else begin
          out = 1'b0; ////output when it is at any other state
        end
      end
      else begin
        out = 1'b0;
      end
    end

end




endmodule
```

★ Testbench :

```
`include "A3_Q1_seq_detector.v"

module seq_tb();

reg clk, reset, inp;
reg [7:0] in; //reg to store 7-bit input
reg [7:0]out; //reg to store 7-bit output
wire outp; //to store 1-bit after each 1-bit input
```

```
integer i;



always #2 clk=~clk; //oscillation in clock

seq_detector x(clk,reset,inp,outp);

initial begin
  clk= 1'b0; //setting clock to 0 initially
  reset = 1'b0; //setting reset to 0 initially and after each input
case changing to 1 to get again IDLE state
  in = 8'b10101010; #35 $display("input = %b output= %b",in,out);
  reset = 1'b1;
  #5 reset =1'b0;
  in = 8'b01110110; #35 $display("input = %b output= %b",in,out);
  reset = 1'b1;
  #5 reset =1'b0;
  in = 8'b01010010; #35 $display("input = %b output= %b",in,out);
  reset = 1'b1;
  #5 reset =1'b0;
  in = 8'b10100010; #35 $display("input = %b output= %b",in,out);
  reset = 1'b1;
  #5 reset =1'b0;
  in = 8'b01011010; #35 $display("input = %b output= %b",in,out);
  reset = 1'b1;
  #5 reset =1'b0;
  in = 8'b01010110; #35 $display("input = %b output= %b",in,out);
  reset = 1'b1;
  #5 reset =1'b0;
  in = 8'b00010010; #35 $display("input = %b output= %b",in,out);
  reset = 1'b1;
  #5 reset =1'b0;
  in = 8'b00010110; #35 $display("input = %b output= %b",in,out);
  reset = 1'b1;
  #5 reset =1'b0;
  in = 8'b11001010; #35 $display("input = %b output= %b",in,out);
  reset = 1'b1;
  #5 reset =1'b0;
  in = 8'b01110011; #35 $display("input = %b output= %b",in,out);
  reset = 1'b1;
```

```
  #5 reset =1'b0;
  in = 8'b01110100; #35 $display("input = %b output= %b",in,out);
  reset = 1'b1;
  #5 reset =1'b0;
  in = 8'b11010010; #35 $display("input = %b output= %b",in,out);
  reset = 1'b1;
  #5 reset =1'b0;
  in = 8'b01010110; #35 $display("input = %b output= %b",in,out);
  reset = 1'b1;
  #5 reset =1'b0;
  in = 8'b00010010; #35 $display("input = %b output= %b",in,out);
  reset = 1'b1;
  #5 reset =1'b0;
  in = 8'b01010000; #35 $display("input = %b output= %b",in,out);

  #5 $finish;


end


//giving input bit-wise whenever the 7-bit input changes
always @(in) begin
  for(i=7;i>=0;i-=1) begin
      #2
      inp= in[i];
      #2
      out[i]= outp;
    end

end

endmodule
```