EGC220
Class Notes
4/28/2023

Baback Izadi
Division of Engineering Programs
bai@engr.newpaltz.edu

Flip-Flop Excitation Tables

PRESENT STATE	NEXT STATE	S	R
Q(t)	Q(t+1)		
0	0	0	X
0	1	1	0
1	0	0	1
1	1	x	0

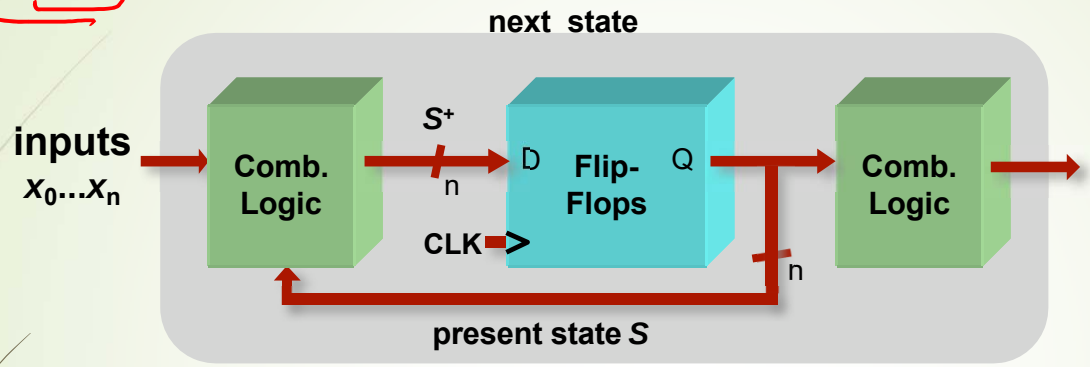
Q(t)	Q(t+1)	T
0	0	0
0	1	1
1	0	1
1	1	0

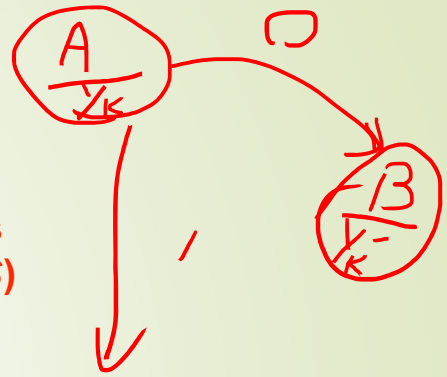
PRESENT STATE	NEXT STATE	J	K
Q(t)	Q(t+1)		
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

Q(t)	Q(t+1)	D
0	0	0
0	1	1
1	0	0
1	1	1

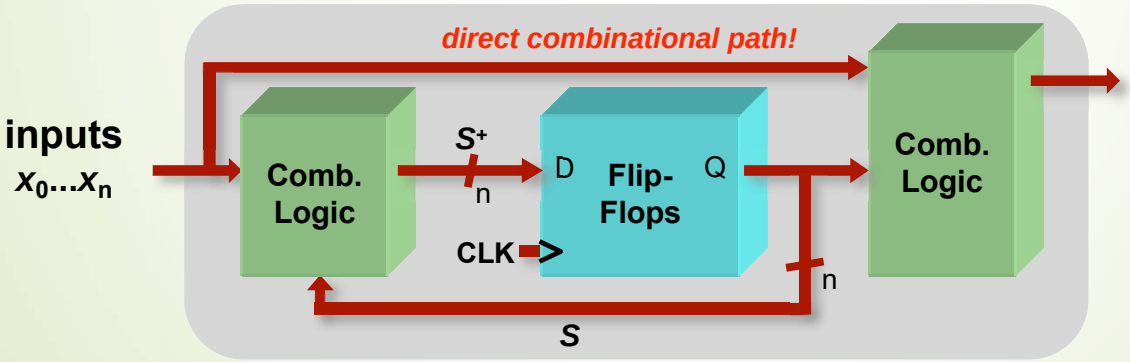
Moore Vs Mealy FSMs: different output generation

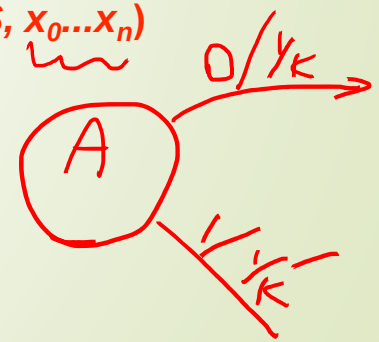
- **Moore FSM:**





- **Mealy FSM:**





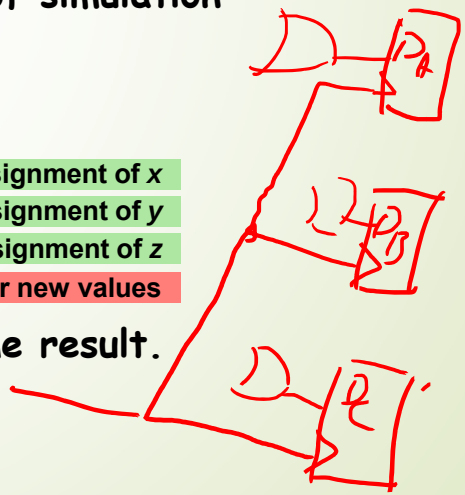
Blocking vs. Nonblocking Assignments

- **Verilog supports two types of assignments within `always` blocks, with subtly different behaviors.**
- ***Blocking assignment:* evaluation and assignment are immediate**

```
always @ (a or b or c)
begin
   x = a | b;          1. Evaluate a | b, assign result to x
   y = a ^ b ^ c;      2. Evaluate a^b^c, assign result to y
   z = b & ~c;         3. Evaluate b&(~c), assign result to z
end
```

- ***Nonblocking assignment:* all assignments deferred until all right-hand sides have been evaluated (end of simulation timestep)**

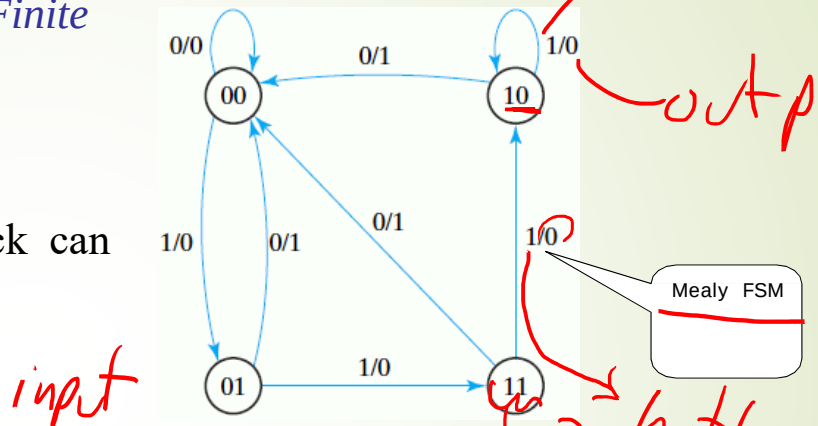
```
always @ (a or b or c)
begin
   x <= a | b;         1. Evaluate a | b but defer assignment of x
   y.<= a ^ b ^ c;     2. Evaluate a^b^c  but defer assignment of y
   z <= b & ~c;        3. Evaluate b&(~c) but defer assignment of z
end                    4. Assign x, y, and z with their new values
```

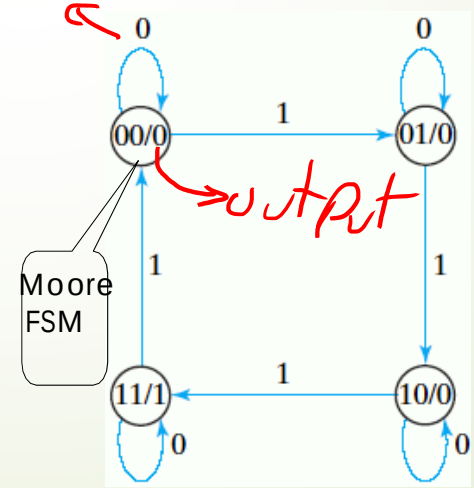


- **Sometimes, as above, both produce the same result. Sometimes, not!**

Finite State Machines Example

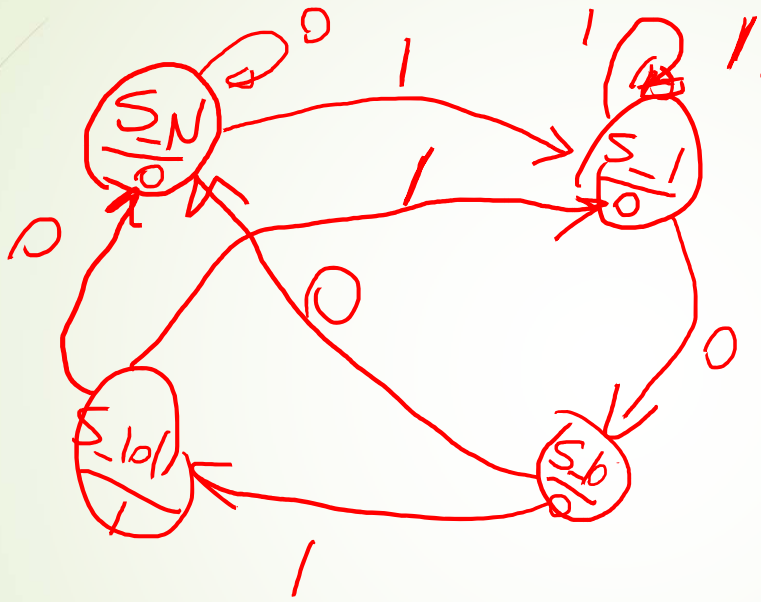
- State diagrams are representations of *Finite State Machines* (FSM)
- Mealy FSM
  - Output depends on input and state
  - Output is not synchronized with clock can have temporarily unstable output
- Moore FSM
  - Output depends only on state





Problem 1

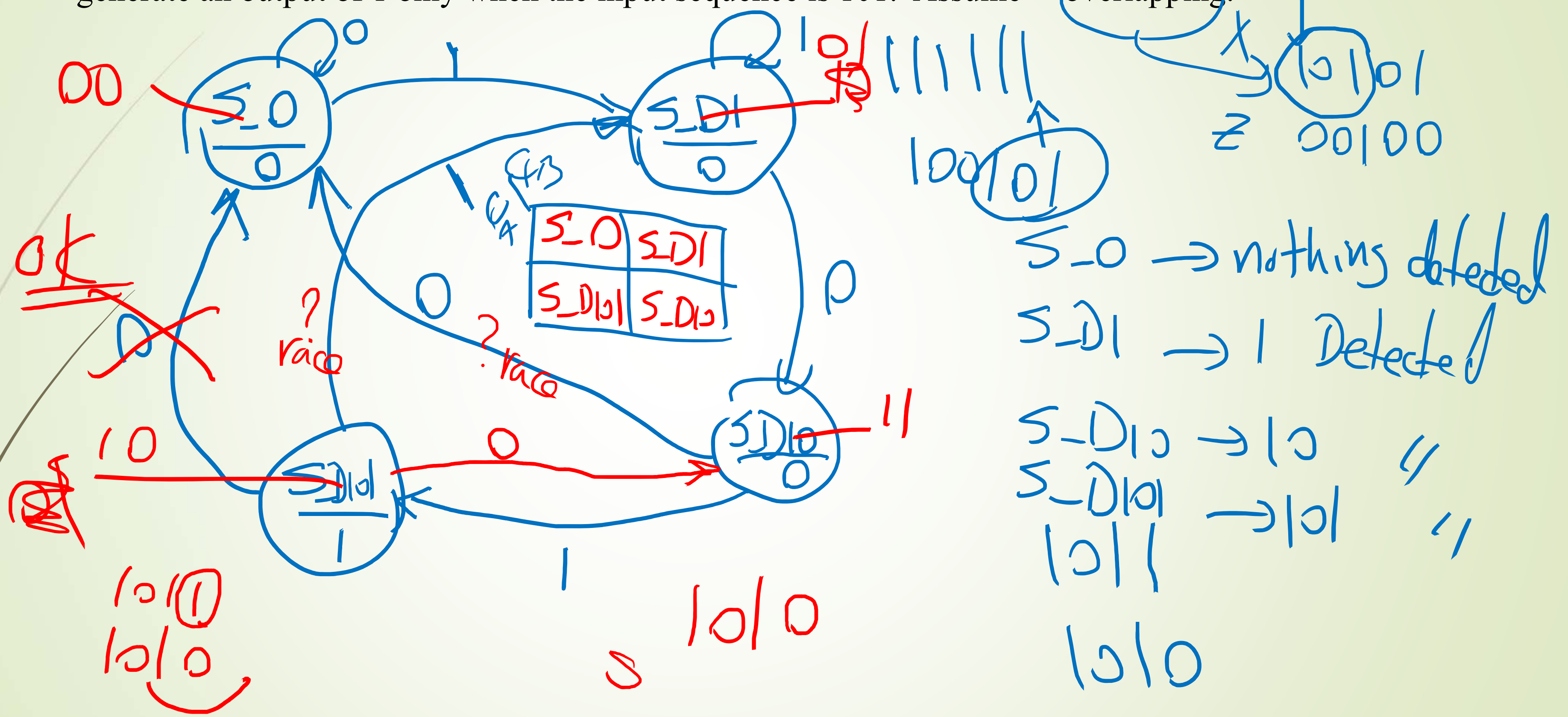
Using D flip-flops, design a Moore based sequence detector with one input and one output, which would generate an output of 1 only when the input sequence is 101. Assume no overlapping.

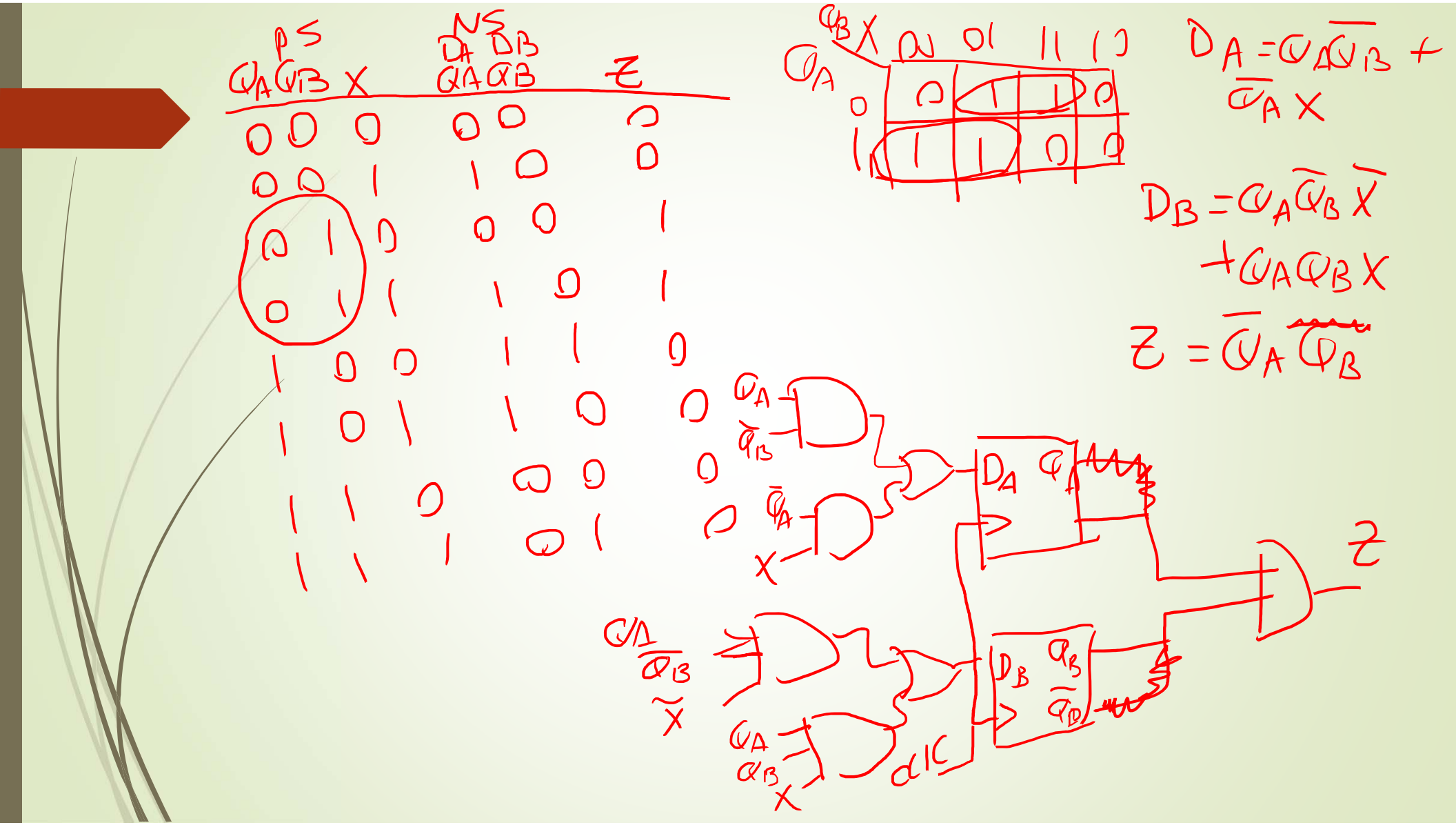


S_N	S_1
S_101	S_10

Problem 1b

Using D flip-flops, design a Moore based sequence detector with one input and one output, which would generate an output of 1 only when the input sequence is 101. Assume overlapping.

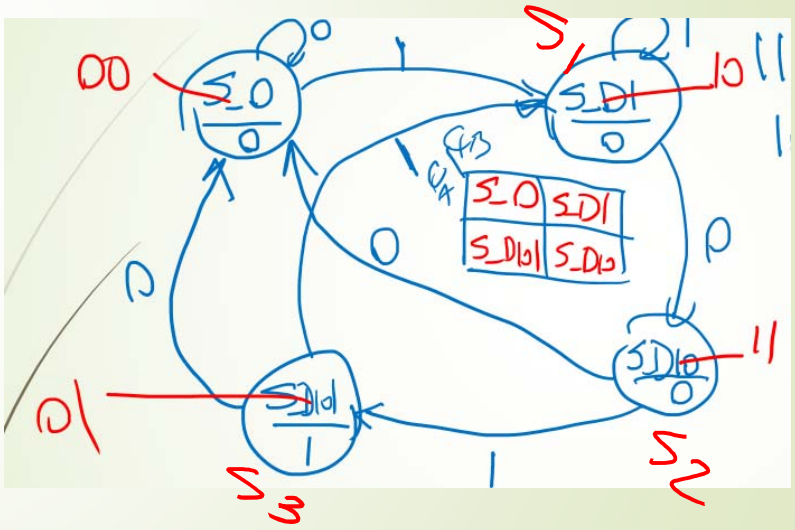



PS
NS
$Q_A Q_B$ X
$D_A D_B$
$Q_A Q_B$
Z
Q_B X
Q_A
00 01 11 10
$D_A = Q_A\bar{Q}_B + \bar{Q}_A X$
$D_B = Q_A\bar{Q}_B\bar{X} + Q_A Q_B X$
$Z = \bar{Q}_A\bar{Q}_B$
Q_A
$\bar{Q}_B$
$\bar{Q}_A$
X
D_A
Q_A
Q_A
$\bar{Q}_B$
$\bar{X}$
Q_A
Q_B
X
clk
D_B
Q_B
$\bar{Q}_B$
Z

Problem 2

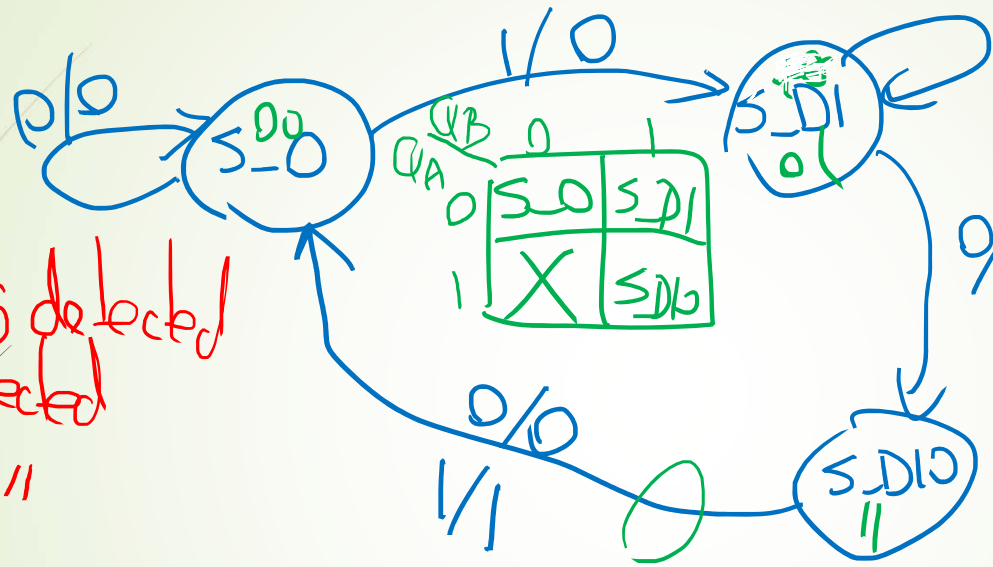
Write a Verilog code for Problem 1

```
module seq3_detect_moore(x,clk, y);
// Moore machine for a sequence 101
  input x, clk;
output y;
reg [1:0] state;
  parameter S0=2'b00, S1=2'b10, S2=2'b11,  S3=2'b01;
// Define the sequential block
always @(posedge clk)
 case (state)
    S0: if (x) state <= S1;
        elsestate <= S0;
    S1: if (x) state <= S1;
          elsestate <= S2;
      S2: if (x) state <= S3;
          else state <= S0;
   S3: if (x) state <= S1;
          else state <= S0;
 endcase
// Define output during S3
assign y = (state == S3);
endmodule
```



Problem 3

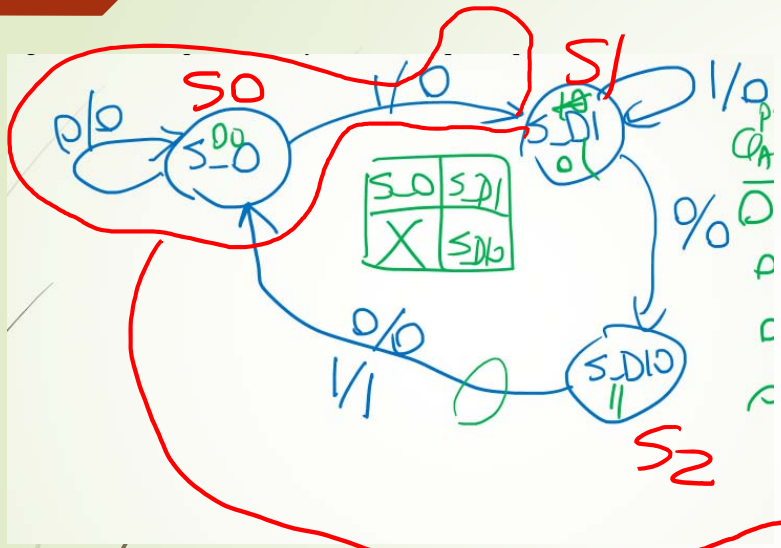
Using JK flip-flops, design a Mealy based sequence detector with one input and one output, which would generate an output of 1 only when the input sequence is 101. Assume no overlapping.



PS Q_A	Q_B	X	NS Q_A	Q_B	$J_A K_A$	$J_B K_B$	Z
0	0	0	0	0	0x	0x	0
0	0	1	0	1	1x	1x	0
0	1	0	1	0	1x	0x	0
0	1	1	0	1	0x	1x	0
1	0	0	x	x	xxx	xxx	x
1	0	1	x	x	xxx	xxx	x
1	1	0	0	0	1x	1x	0
1	1	1	0	0	1x	1x	1

Problem 4

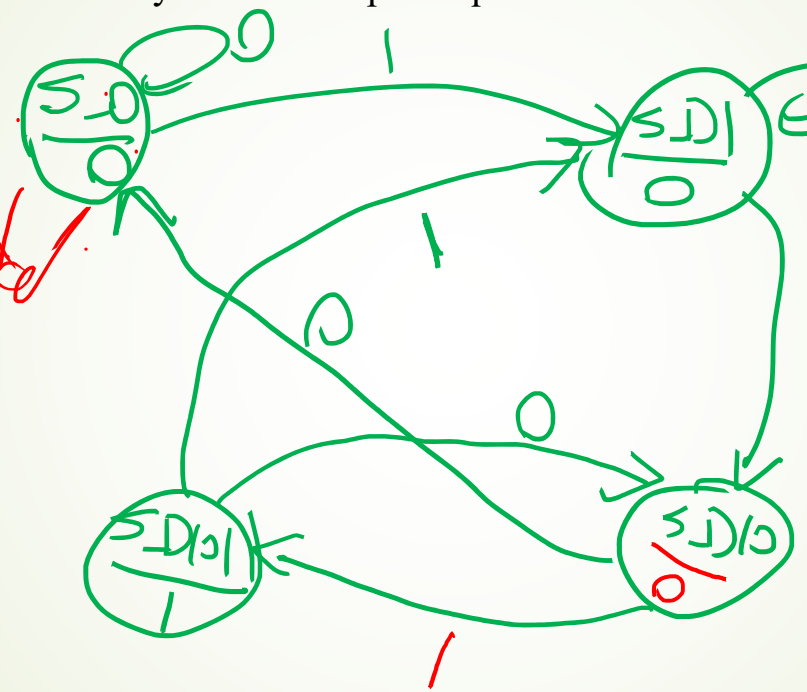
Write a Verilog code for Problem 3. Use default flip-flop given by Verilog.




```
module seq3_detect_mealy(x,clk, y);

// Mealy machine for a three-1s sequence detection
 input x, clk;

 output y;  reg y;

 parameter S0=2'b00, S1=2'b01, S2=2'b11;

// Next state and output combinational logic

// Use blocking assignments "="

always @(x or pstate)

   case (pstate)

     S0: if (x) begin nstate = S1; y = 0; end

       else begin nstate = S0; y = 0; end

     S1: if (x) begin nstate = S1; y = 0; end

       else begin nstate = S2; y = 0; end

     S2: if (x) begin nstate = S0; y = 1; end

       else begin nstate = S0; y = 0; end

   endcase

// Sequential logic, use nonblocking assignments "<="

      always @(posedge clk)

          pstate <= nstate;

     endmodule
```

Problem 5

Using D flip-flops, design a Moore based sequence detector with one input and one output, which would generate an output of 1 only when the input sequence is 101. Assume overlapping.



S0: Nothing detected

S1: 1 Detected

S10: 10 =

S101: 101 =

101

1011

Problem 7

Using JK flip-flops, design a Mealy based sequence detector with one input and one output, which would generate an output of 1 only when the input sequence is 101. Assume overlapping.