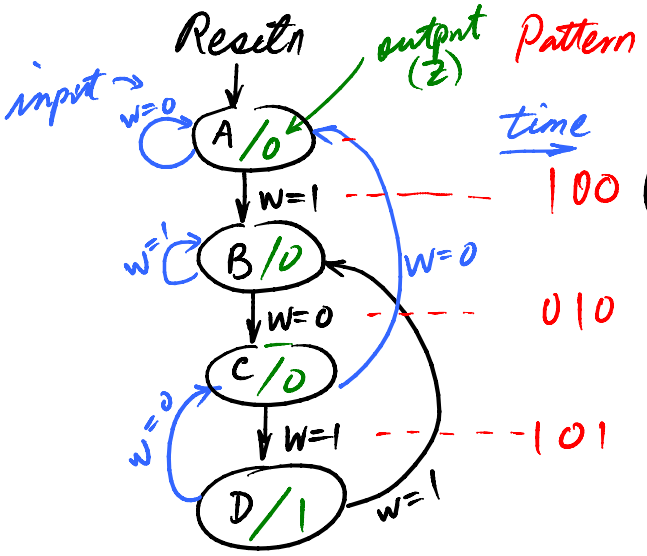
FSM (cont.)

Design example = "101" pattern recognition

1. State diagram (2nd version)

meaning of each state:
- A ← seeing "0" only
- B ← seeing 1st "1" in pattern
- C ← seeing "10" in pattern
- D ← seeing "101" in pattern



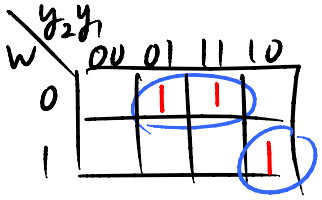
2. State Table

Present State	Next State w=0	Next State w=1	output z
A	A	B	0
B	C	B	0
C	A	D	0
D	C	B	1

3. # of FFs = use y_2y_1 = 00 (A), 01 (B), 10 (C), 11 (D)

P.S y_2y_1	N.S w=0 Y_2Y_1	N.S w=1 Y_2Y_1	output z
0 0	0 0	0 1	0
0 1	1 0	0 1	0
1 0	0 0	1 1	0
1 1	1 0	0 1	1

4. Derive Comb. ckt A & B.

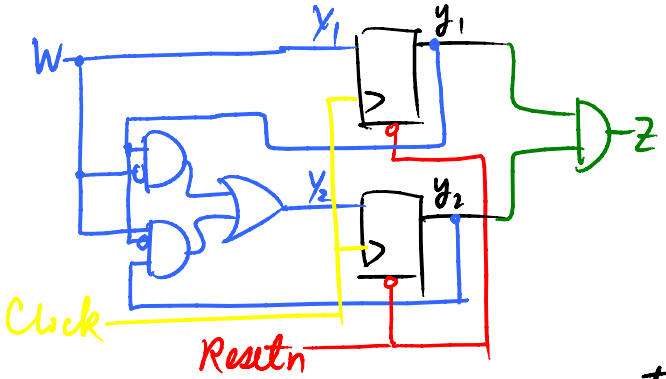


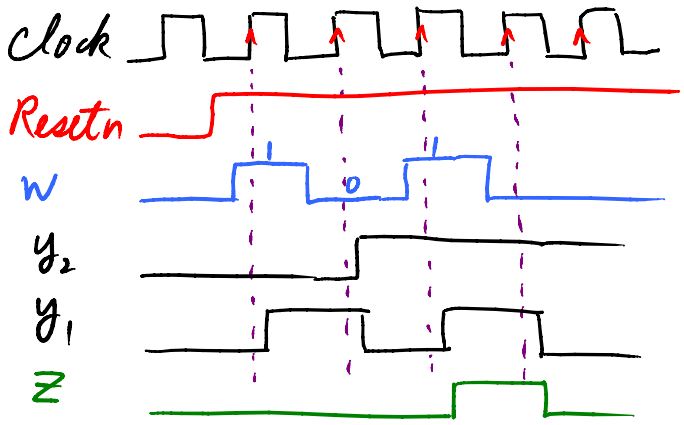
$$Y_2 = y_1\bar{w} + y_2\bar{y}_1w$$

$$Y_1 = w$$

$$z = y_2y_1$$

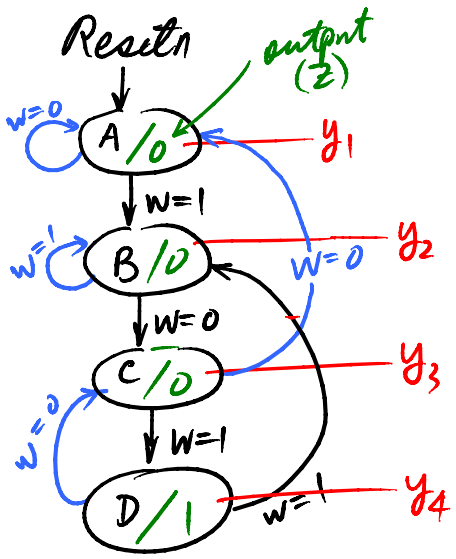
5. Draw FSM ckt.





— Verilog code for a FSM can be written by using three blocks:
Comb. cct. A, FFs, Comb. cct B.

```
Module seq101 (input Clock, w, Resetn, output z);
  reg [2:1] y, Y; // y2, y1 are the FFs present state
                  // Y2, Y1 are the next state.
  parameter A=2'b00, B=2'b01, C=2'b10, D=2'b11;
  // Comb. cct A describing state transitions
  always@ (w, y)
    case (y)
      A: if (!w) Y=A;  (if input w=0, stay in state A)
         else Y=B;     (if input w=1, transition to state B)
      B: if (!w) Y=C;
         else Y=B;
      C: if (!w) Y=A;
         else Y=D;
      D: if (!w) Y=C;
         else Y=B;
    end case
  // FFs
  always@ (posedge Clock)
    if (Resetn==0)      } synchronous
      y<=A;             } reset
    else
      y<=Y;
  // Comb. cct B
  assign z=(y==D);  { assign 1 to z if it's in state D
end module
```



Now, we can redo step 3 (chose # of FFs = 4)

(one FF for each state)

$y_4 y_3 y_2 y_1 = 0001$ (A)
0010 (B)
0100 (C)
1000 (D)

} one-hot state coding

consider all links pointing into A
(State transition into A)

1. when you are in state A and w=0
2. when you are in state C and w=0

$$Y_1 = y_1\bar{w} + y_3\bar{w} = (y_1 + y_3)\bar{w}$$
(1) (2)

let's do it for state B

$$Y_2 = y_1 w + y_2 w + y_4 w = (y_1 + y_2 + y_4) w$$

for state C ⟶ $Y_3 = y_2\bar{w} + y_4\bar{w} = (y_2 + y_4)\bar{w}$

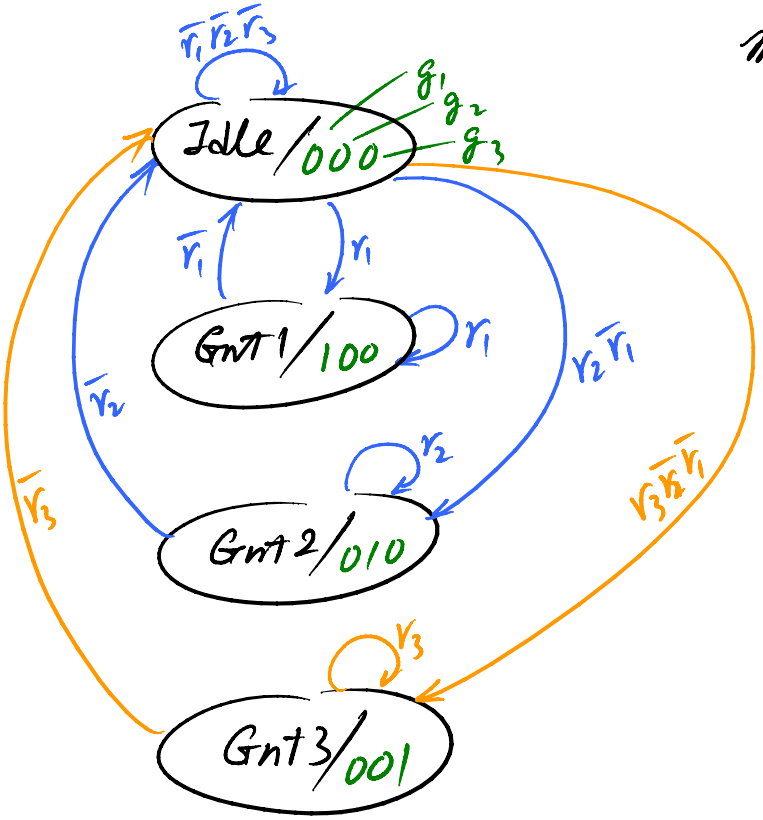
for state D ⟶ $Y_4 = y_3 w$

for output $z = y_4$ (z=1 whenever you are in state D)

Design example (Arbiter)

— design a FSM that controls access to a shared resources by three devices. Each device requests use of the resources by asserting r_1, r_2 or r_3. The arbiter decides which device "gets" the resource, and sets its grant signal g_1, g_2 or g_3. There is a priority scheme: $r_1 > r_2 > r_3$.

1. State Diagram



2. use 2 FFs

```
Module arbiter(input R[1:3],
    input Clock, Resetn, output [1:3]G);
    reg [2:1] y, Y;
    parameter Idle=2'b00, Gnt1=2'b01,
        Gnt2=2'b10, Gnt3=2'b11;
// State Diagram
always@(y, R)
  case (y)
    Idle: if (R[1]) Y=Gnt1;
          else if (R[2]) Y=Gnt2;
          else if (R[3]) Y=Gnt3;
          else Y=Idle;
    Gnt1: if (R[1]) Y=Gnt1;
          else Y=Idle;
    Gnt2: if (R[2]) Y=Gnt2;
          else Y=Idle;
    Gnt3: if (R[3]) Y=Gnt3;
          else Y=Idle;
    default: if (R[3]) Y=Gnt3;
          else Y=Idle;
// FFs
always@(posedge Clock)
    if (!Resetn) y<=Idle;
    else y<=Y;
// outputs
    assign G[1]=(y==Gnt1);
    assign G[2]=(y==Gnt2);
    assign G[3]=(y==Gnt3);
end module
```

not necessary here, but needed when some unused states exist. → default

synchronous reset