1 – 1010 SEQUENCE DETECTOR MEALY WITHOUT OVERLAP

```
module melfsm(din, reset, clk, y);
input din;
input clk;
input reset;
output reg y;
reg [1:0] cst, nst;
parameter S0 = 2'b00, //all state
          S1 = 2'b01,
          S2 = 2'b10,
          S3 = 2'b11;
always @(cst or din)
 begin
 case (cst)
   S0: if (din == 1'b1)
          begin
         nst = S1;
          y=1'b0;
          end
      else
          begin
          nst = cst;
          y=1'b0;
          end
   S1: if (din == 1'b0)
          begin
        nst = S2;
          y=1'b0;
          end
       else
          begin
          y=1'b0;
           nst = cst;
          end
   S2: if (din == 1'b1)
          begin
         nst = S3;
          y=1'b0;
          end
          else
          begin
          nst = S0;
          y=1'b0;
          end
   S3: if (din == 1'b0)
          begin
         nst = S0;
          y=1'b1;
//output will come
          end
        else
          begin
          nst = S1;
          y=1'b0;
          end
    default: nst = S0;
  endcase
end
always@(posedge clk)
          begin
           if (reset)
             cst <= S0;
           else
             cst <= nst;
          end
endmodule
```

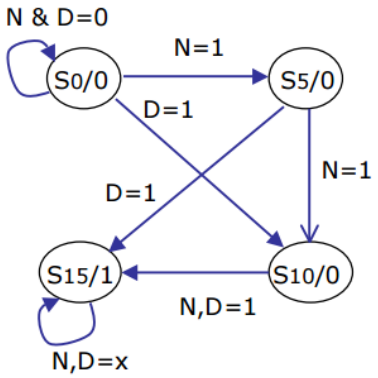
2 – up down counter

```
 7 module up_down_counter    (
 8 out       ,  // Output of the counter
 9 up_down   ,  // up_down control for counter
10 clk       ,  // clock input
11 data      ,  // Data to load
12 reset        // reset input
13 );
14 //----------Output Ports--------------
15 output [7:0] out;

16 //------------Input Ports--------------
17 input [7:0] data;
18 input up_down, clk, reset;

19 //------------Internal Variables--------
20 reg [7:0] out;

21 //-------------Code Starts Here-------
22 always @(posedge clk)
23 if (reset) begin // active high reset
24   out <= 8'b0 ;
25 end else if (up_down) begin
26   out <= out + 1;
27 end else begin
28   out <= out - 1;
29 end
30
31 endmodule
```

3 – Vending Machine

```
module vending_moore(N, D, clk, reset, open);
// Moore FSM for a vending machine
  input N, D, clk, reset;
  output open;
  reg [1:0] state;
  parameter S0=2'b00, S5=2'b01, S10=2'b10,
S15=2'b11;

// Define the sequential block
  always @(posedge reset or posedge clk)
    if (reset) state <= S0;
   else
     case (state)
            S0:  if (N) state <= S5;
                   else if (D) state <= S10;
                        else state <= S0;
            S5:  if (N)            state <= S10;
                   else if (D) state <= S15;
                        else state <= S5;
            S10: if (N) state <= S15;
                   else if (D) state <= S15;
                        else state <= S10;
            S15: state <= S15;
      endcase
// Define output during S3
            assign open = (state == S15);
endmodule
```

Synthesizing Moore FSM directly from state diagram




```
module vending_mealy(N, D, clk, reset, open);
// Mealy FSM for a vending machine
  input N, D, clk, reset;
  output open;
  reg [1:0] pstate, nstate;
  reg open;
  parameter S0=2'b00, S5=2'b01, S10=2'b10, S15=2'b11;

// Next state and ouptut combinational logic
    always @(N or D or pstate or reset)
      if (reset)
           begin nstate = S0; open = 0; end
            else case (pstate)
              S0:  begin open = 0; if (N) nstate = S5;
                     else if (D) nstate = S10;
                        else nstate = S0; end
              S5:  begin open = 0; if (N)nstate = S10;
                     else if (D) nstate = S15;
                        else nstate = S5; end
             S10: if (N | D) begin nstate = S15; open = 0; end
                     else begin nstate = S10; open = 0; end
             S15: begin nstate = S0; open = 1; end
           endcase
// FF logic, use nonblocking assignments "<="
           always @(posedge clk)
                      pstate <= nstate;
endmodule
```

Synthesizing Mealy FSM directly from state diagram