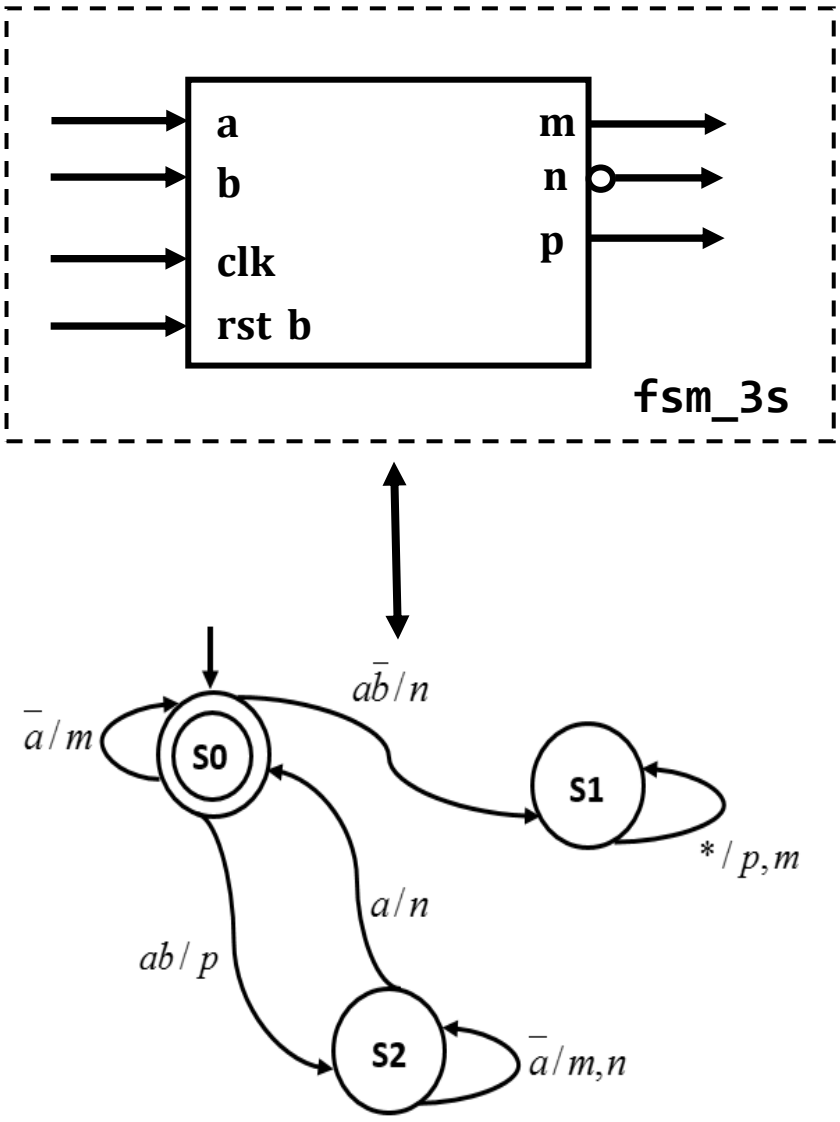
Week 6 – Mealy Machines

Implementing Finite State Machines in Verilog

P.6.1 Implement, using Verilog language, the following architecture of a Finite State Machine (FSM) with 3 states. The module of the FSM should be called ***fsm_3s***.

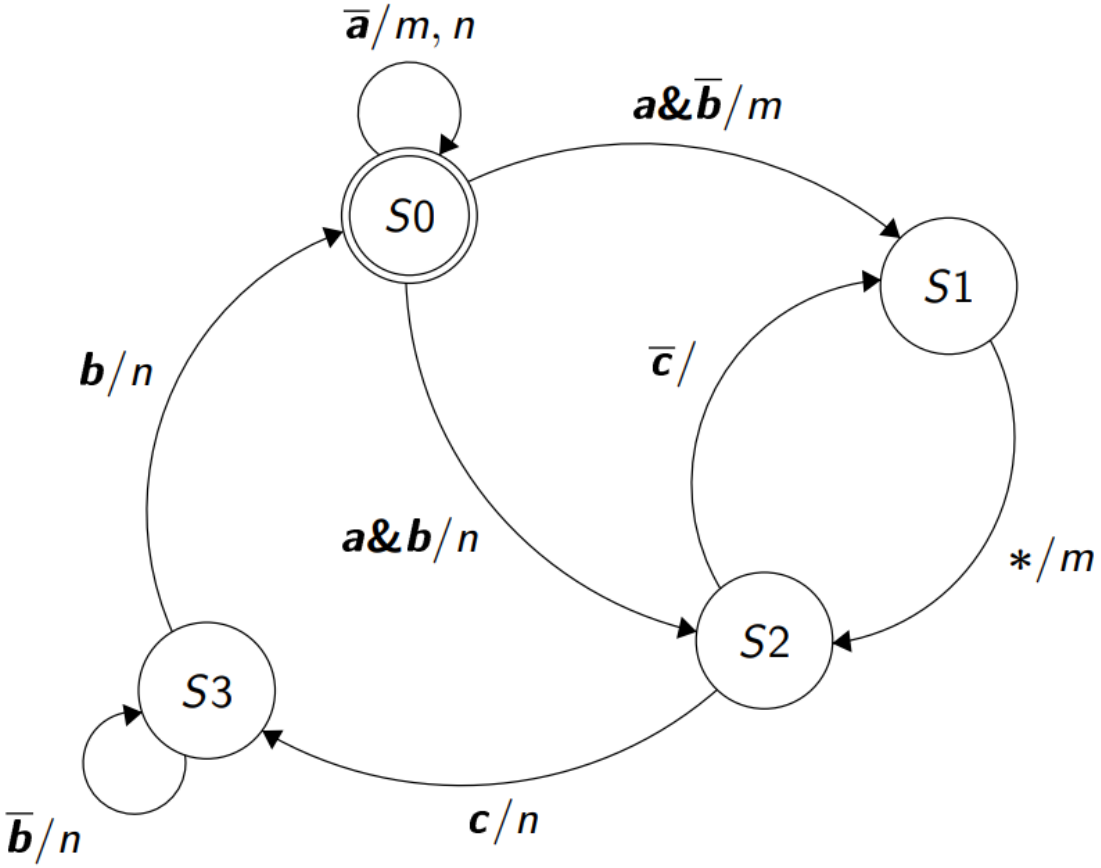


```
module fsm_3s  ( input clk, rst_b,
                 input a, b,
                 output reg m, n, p );
// Step 1
localparam S0 = 2'd0;
localparam S1 = 2'd1;
localparam S2 = 2'd2;
// Step 2
reg [1:0] st;
reg [1:0] st_next;
// Step 3
always @ (*) begin
st_next = S0;
  case (st)
  S0: if (!a)
      st_next = S0;
      else
      if (!b)
      st_next = S1;
  S1: st_next = S1;
  S2: if (a)
      st_next = S0;
      else
      st_next = S2;
endcase
end
```

```
// Step 4
always @ (*) begin
  m = 1'd0 ;
  n = 1'd1 ;
  p = 1'd0 ;
  case (st)
S0: if (!a)
       m = 1'd1 ;
      else
      if (!b)
       n = 1'd0 ;
S1: begin
       m = 1'd1 ;
       p = 1'd1 ;
    end
S2: if (!a) begin
       m = 1'd1 ;
       n = 1'd0 ;
end
      else
       n = 1'd0 ;
endcase
end
```

```
// Step 5
always @ (posedge clk, negedge rst_b)  begin
  if (!rst_b)
    st <= S0 ;
  else
    st <= st_next ;
end
endmodule
```

P.6.2 Construct, in a similar way, the following FSM with 4 states. The module's name will be ***fsm_4s***.



```
module fsm_4s  ( input clk, rst_b, a, b, c,
                 output reg m,n,p );
localparam S0 = 2'd0;
localparam S1 = 2'd1;
localparam S2 = 2'd2;
localparam S3 = 2'd3;
reg [1:0] st, st_next;
always @ (*)
  case (st)
  S0: if (!a)   st_next = S0;  else      if (b)    st_next = S2;
      else      st_next = S1;
  S1:           st_next = S2;  else                st_next = S1;
  S2: if (c)    st_next = S3;  else                st_next = S1;
  S3: if (c)    st_next = S3;  else                st_next = S1;
  endcase
always @ (*) begin
m = 1'd0; n = 1'd0;
  case (st)
  S0: if (!a)   {m,n} = 2'b11; else      if (b)    n = 1'd1;
      else      m = 1'd1;
  S1:           m = 1'd1;
  S2: if (c)    n = 1'd1;
  endcase
            end
always @ (posedge clk, negedge rst_b)  begin
  if (!rst_b)   st <= S0 ;     else                st <= st_next;
                                          end
endmodule
```