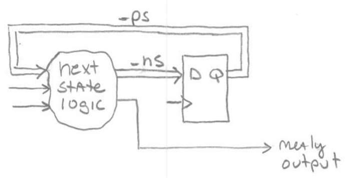
Mealy State Machines

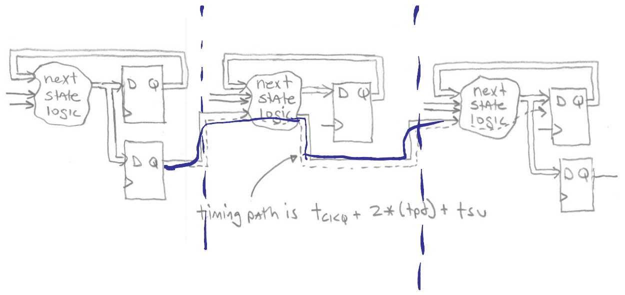
- Mealy state machine outputs are formed by present state and input



- The structure and operation is identical to a Moore machine except for how the output is formed
- The machine may also have Moore-type outputs

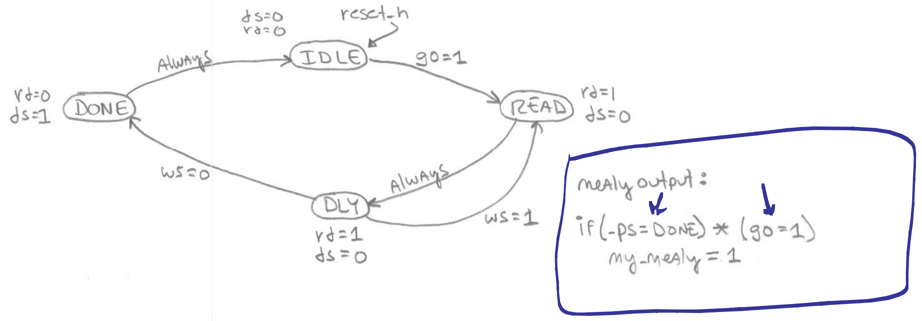
Mealy State Machines

- Mealy outputs are best to stay away from unless a special situation requires it
- Timing can be tricky to track as flip-flop Q outputs may go through two stages of combo logic prior to the next D-input.
- At synthesis time, special care must be taken to constrain this special signal



Mealy State Machines

- Mealy state machine diagrams best show output formation outside the state diagram
- Simply write the mealy equation below the state diagram and clearly indicate that it is a mealy output

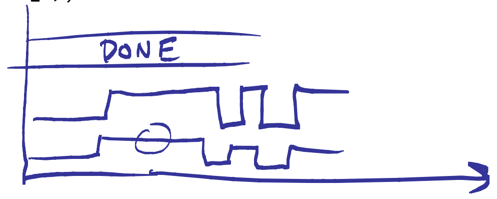


```
mealy output:
if (_ps=DONE) * (go=1)
    my_mealy = 1
```

Mealy State Machines

- Mealy state machine coding
  - Mealy outputs may be formed inside most any state machine
  - The Mealy output is formed in the case/if expression
  - The output is asserted under a state qualified by an if

```
module mealy_noglitch ( output rd, ds,
                        output reg my_mealy,
                        input  go, ws, clk, reset_n);
  enum reg [3:0]{ .......
  always_ff @(posedge clk, .......
  always_comb begin
    no_glitch_ns = XX;
    my_mealy = 1'b0;    //default output
    case (no_glitch_ps)
      IDLE : if (go)   no_glitch_ns = READ;
      else             no_glitch_ns = IDLE;
      READ :           no_glitch_ns = DLY;
      DLY  : if (!ws)  no_glitch_ns = DONE;
      else             no_glitch_ns = READ;
      DONE : begin     no_glitch_ns = IDLE;
                       if (go) my_mealy = 1'b1;
             end
    endcase
  end
```



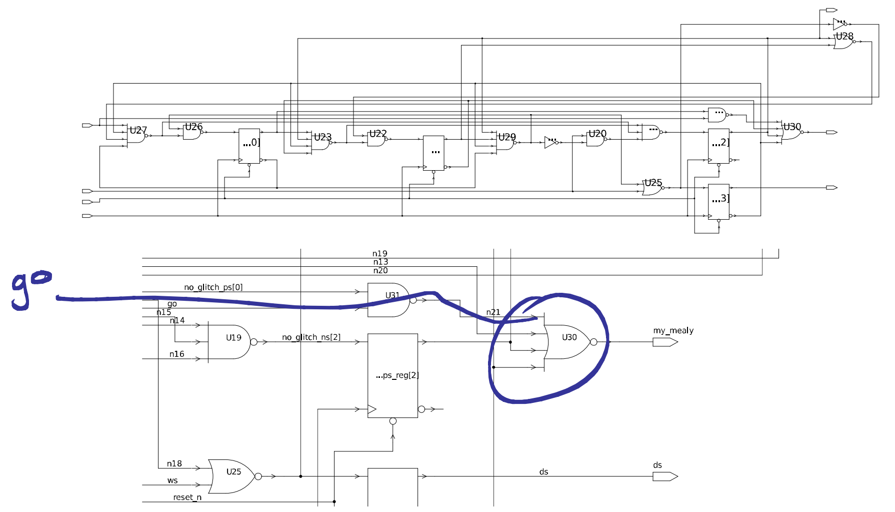
Mealy State Machines

- The whole enchilada

```
module mealy_noglitch ( output rd, ds,
                        output reg my_mealy,
                        input  go, ws, clk, reset_n);

  enum reg [3:0]{
    IDLE = 4'b00_00,
    READ = 4'b01_01,
    DLY  = 4'b01_10,
    DONE = 4'b10_11,
    XX   = 'x} no_glitch_ns, no_glitch_ps;

  always_ff @(posedge clk, negedge reset_n)
    if (!reset_n)      no_glitch_ps <= IDLE;
    else               no_glitch_ps <= no_glitch_ns;

  always_comb begin
    no_glitch_ns = XX;
    my_mealy = 1'b0;    //default output
    case (no_glitch_ps)
      IDLE : if (go)   no_glitch_ns = READ;
      else             no_glitch_ns = IDLE;
      READ :           no_glitch_ns = DLY;
      DLY  : if (!ws)  no_glitch_ns = DONE;
      else             no_glitch_ns = READ;
      DONE : begin     no_glitch_ns = IDLE;
                       if (go) my_mealy = 1'b1;
             end
    endcase
  end
  assign {ds,rd} = no_glitch_ps[3:2];
endmodule
```

Mealy State Machines

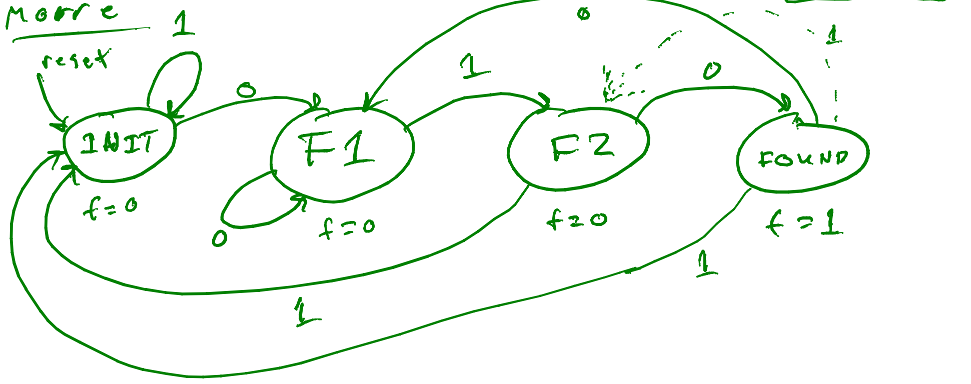
- Synthesis results



Write both a mealy and moore machine to find the pattern "0 1 0" in a serial string of binary inputs.

Input : 0 0 1 1 1 0 1 0 1 0 1

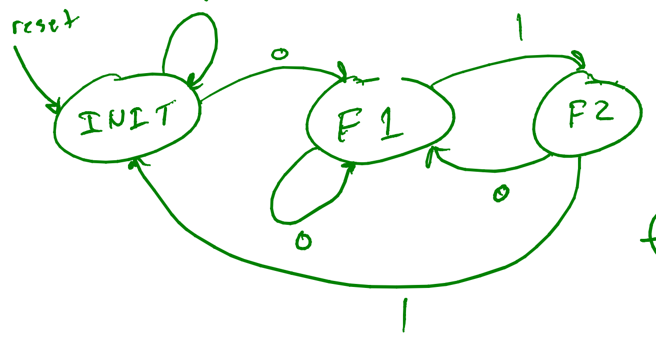
Moore

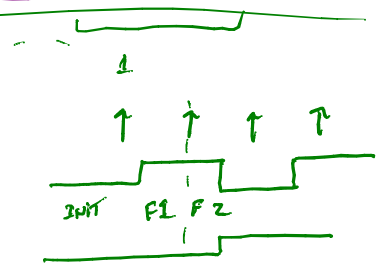


Write both a mealy and moore machine to find the pattern "0 1 0" in a serial string of binary inputs.

Input : 0 0 1 1 1 $\underbrace{\overset{\downarrow}{0}\ 1\ \overset{\downarrow}{0}}$ 1 0 1

Mealy





f = F2 & !in