# EEC 180

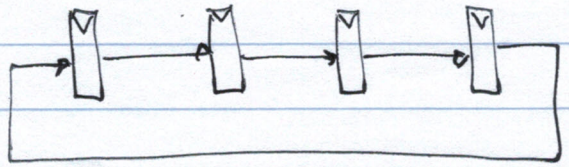
April 29, 2021

## Counters

a) count up

b) count down

c) one-hot
- one FF for each state
- ring counter: → 0 - 1 - 2 - 3 →



| | | | | | |
|---|---|---|---|---|---|
| initialize | 1 | 0 | 0 | 0 | t |
| | 0 | 1 | 0 | 0 | t+1 |
| | 0 | 0 | 1 | 0 | t+2 |
| | 0 | 0 | 0 | 1 | t+3 |

state 3

+ no adder required!
- 1 FF per state
+ fully decoded states

# Finite State Machines

State:

1) Minimal binary coding

$$\#\text{ of state bits} = \lceil \log_2(\#\text{ of states}) \rceil$$

Ex: 5 states
→ 3 bits

8 possible states 000 – 111

BAD: not optimize which 5 to choose

2) One-hot

Ex: 5 states
→ 5 FFs
no optimizing

+ zero-time state decode logic
+ fast state transitions
- not practical for large # of states

Outputs

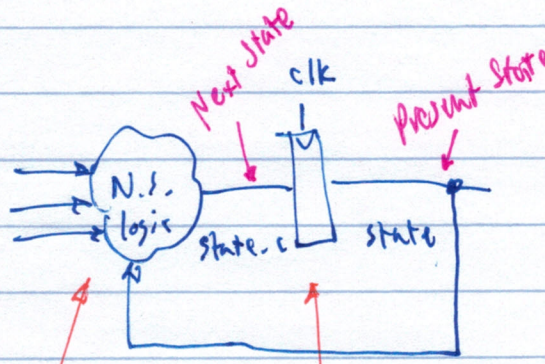
Moore: output is a function of state

Mealy: " " " " " and inputs

Counter: " " the state

Major Components

1) State register
2) Next state logic
3) output logic




3 key signals

1) inputs
2) state + counters
3) outputs

reg
99.9% +
- inputs
- outputs

reg
- inputs
- outputs

## 5 Characteristics

1) Default in N.S. logic
   state_c = state;

2) case (state) // present state

3) STATE : begin // for each state
   ≡
   end

4) if (reset == 1'b1) // at end of N.S. logic

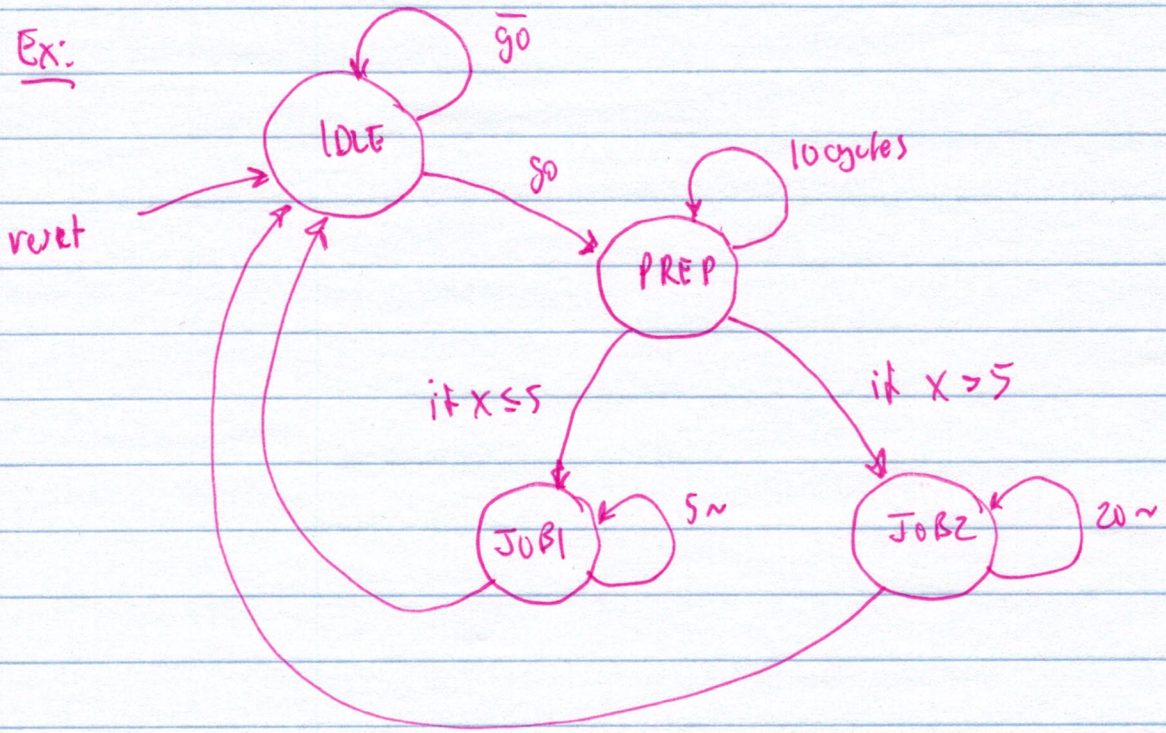
5) instantiate FF's in separate always block

counters inside a FSM

a) // default
count_c = count + 8'h01; // incr.
count_c = count - 8'h01; // decr.

b) // save power = $CV^2f$ f = switching freq.
// hold counter value when unused
count_c = count;
count_c = 8'h00;

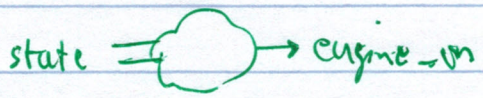
Ex:



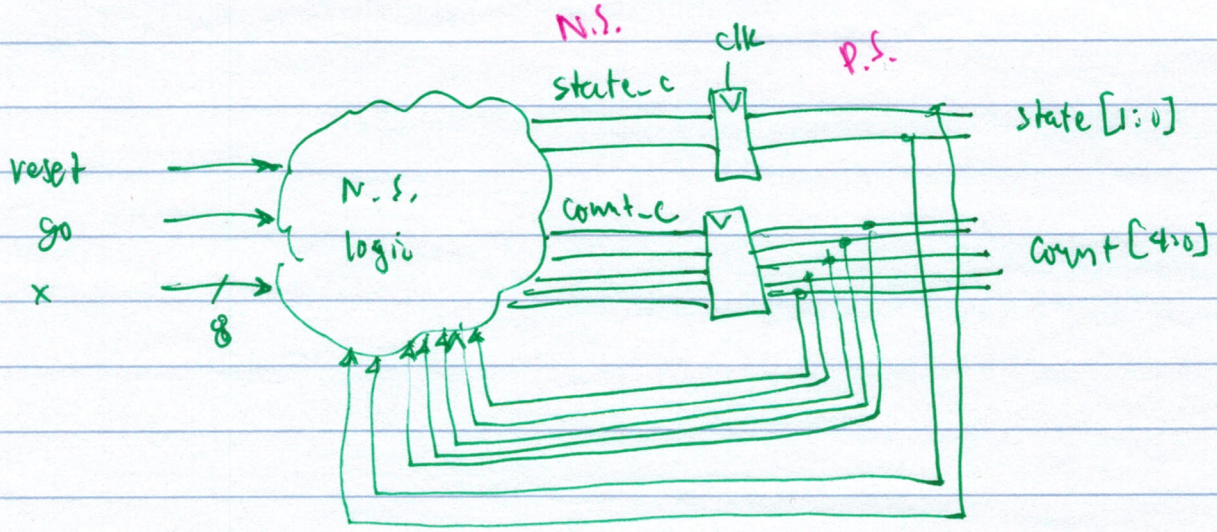

3 bits 000
001

state:
1) state register - choose 2 bits
00
01
10
11
2) counter
a) 3 counters
* b) 1 counter - max count = 20 - 5 bits count down

- inputs : reset, go, x[7:0]
- FFs : state[1:0]
  count[4:0]
- outputs not specified, calculated from state




Circuit diagram




Code

```
parameter IDLE  = 2'h0;
"         PREP  = 2'h1;
"         JOB1  = 2'h2;
"         JOB2  = 2'h3;

                FF          count logic
reg [1:0] state, state_c;
reg [4:0] count, count_c;
```

```
always @(reset or go or x or state or count) begin
    // defaults
    count_c = count - 5'b00001;
    state_c = state;

    case (state)
        IDLE: begin
            if (go == 1'b1) begin
                state_c = PREP;
                count_c = 5'd10;
            end
            else begin
                count_c = count;
            end
        end
        PREP: begin
            if (count == 5'b00000) begin
                if (x <= 8'd005) begin      // goto JOB1
                    state_c = JOB1;
                    count_c = 5'd05;
                end
                else begin                  // JOB2
                    state_c = JOB2;
                    count_c = 5'd20;
                end
            end
        end
```