Finite State Machines

Basic State Machine

Why FSMs?

- Models different behavoiur at different times (states)

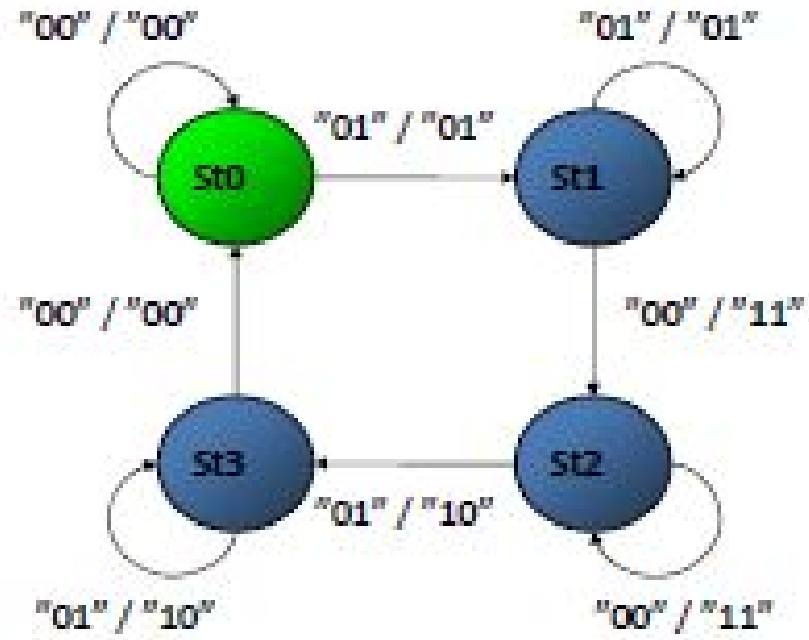
A state machine requires:

- An initial state (Reset)
- Transitions with stable states
- Default values (Case statement)

Realizes:

- Datapath
- Controller
- Datapath+Controller

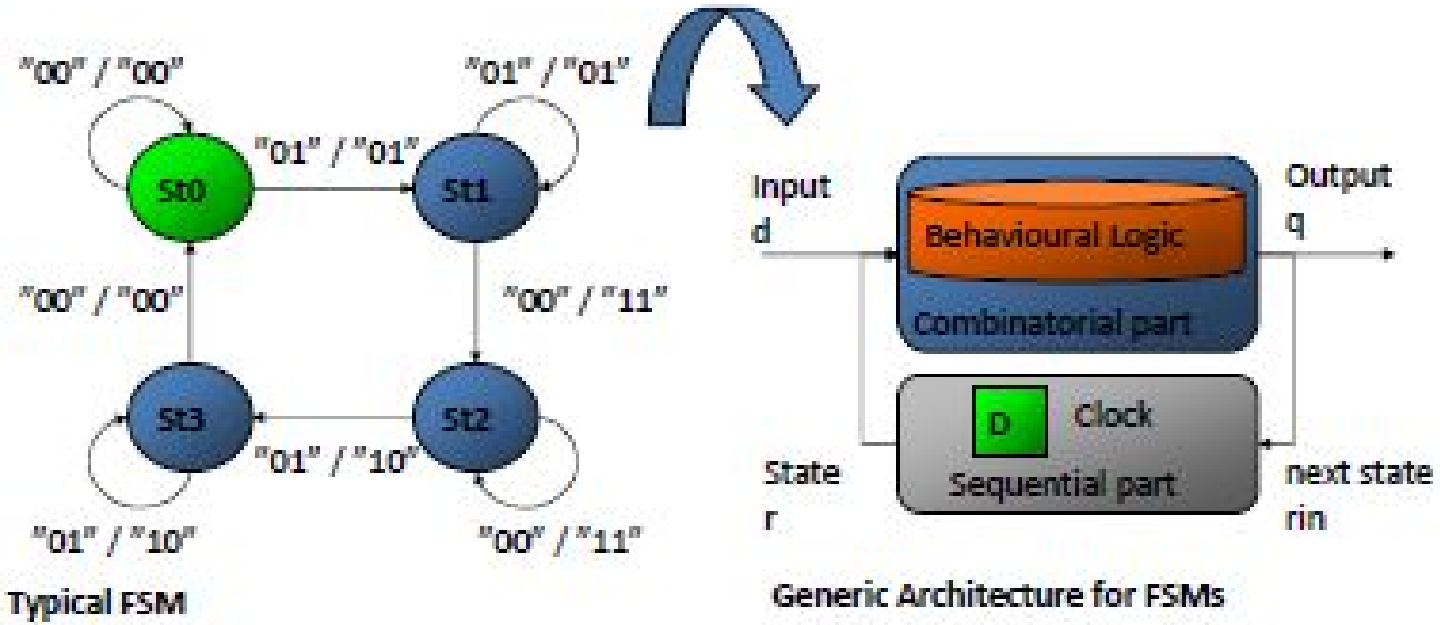
Basic State Machine



A Typical state machine

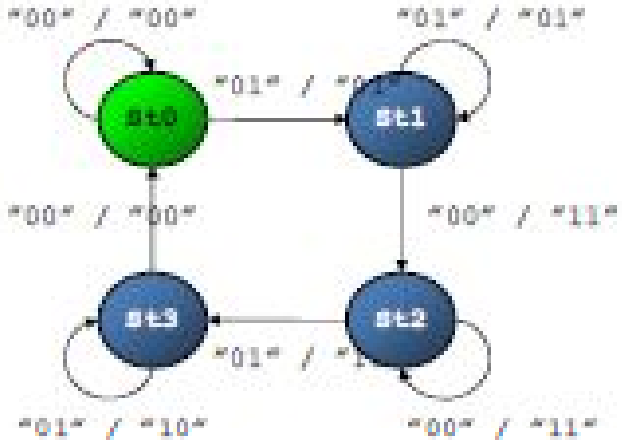
Output of a Mealy machine is state and input dependent

Transforming a State Machine



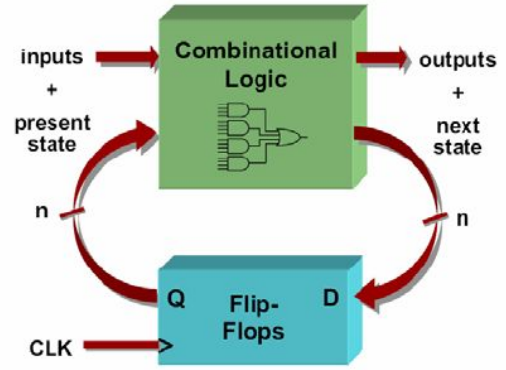
Realization of FSM

```
case (state) is -- Current state and input dependent
      when st0 => if (input = '01') then
                        next_state <= st1;
                        next_output <= "01"
                  end if;
      when ....
      when others =>
                  next_state <= next_state; -- Default
                  next_output <= "00";
   end case;
```



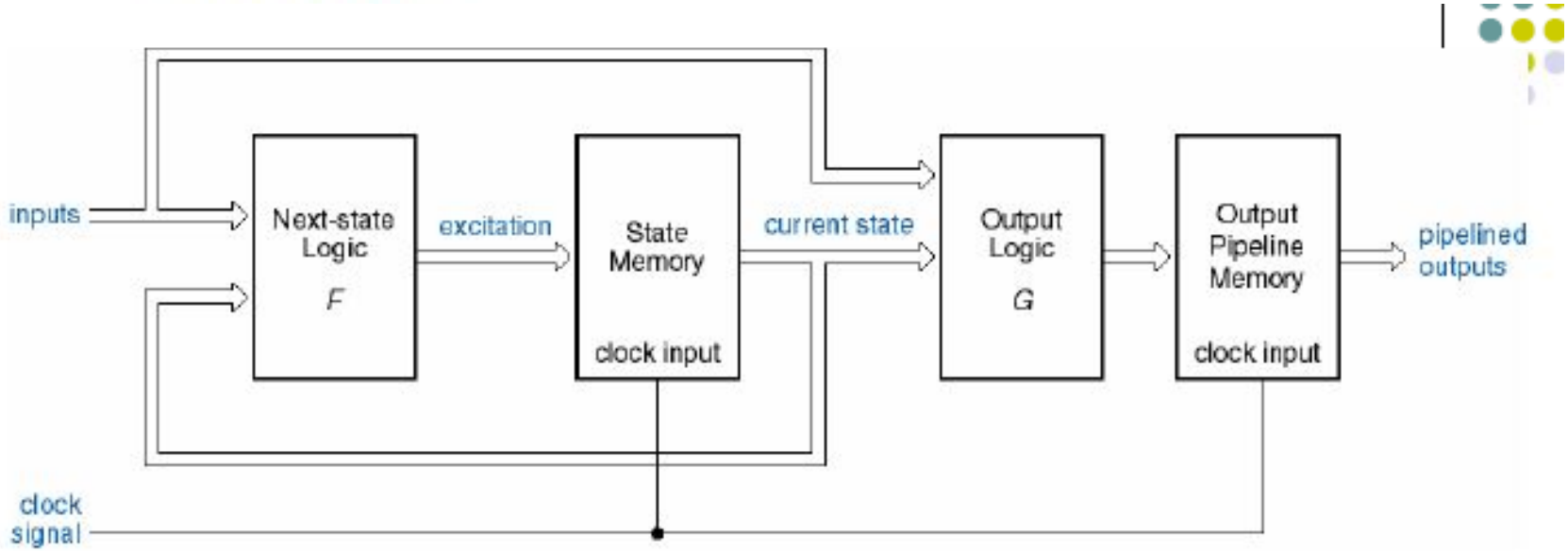
Sequential part:

```
if clk'event and clk = '1' then
  if reset = '1' then
     state <= st0;
     output <= "00";
  else
     state <= next_state;
     output <= next_output;-- regis
  end if;
end if;
```



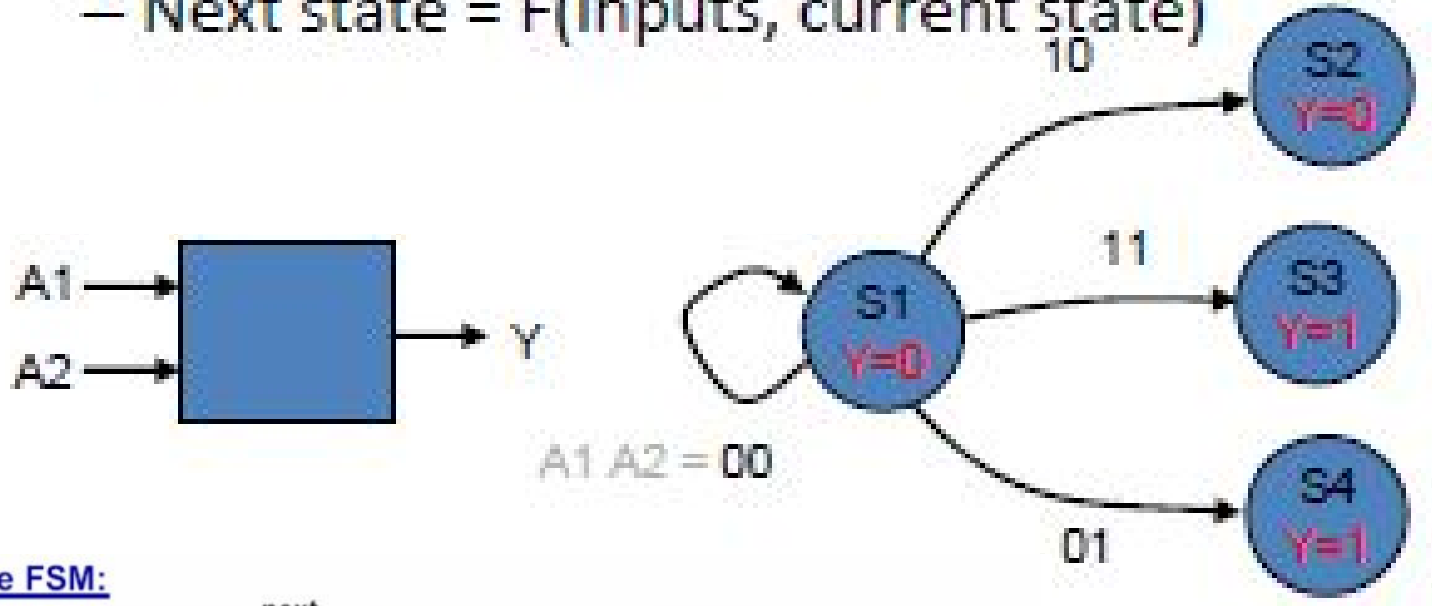
FSM Structure

- A FSM can be split in three parts:
  - State Transition Logic block
  - State Memory block (register)
  - Output logic

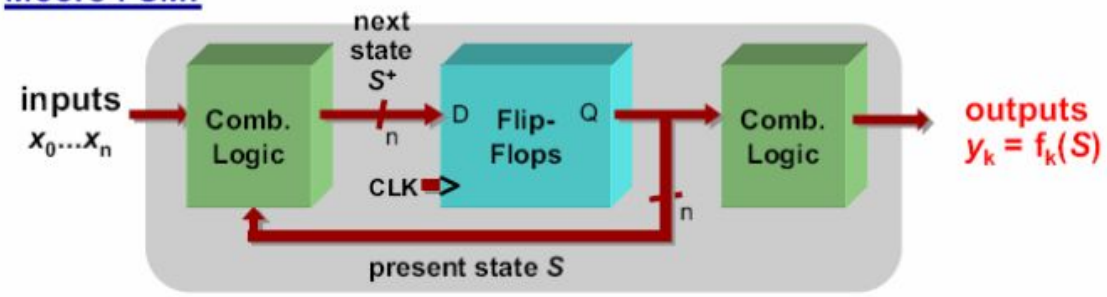


Mealy vs. Moore

- Moore
  - Out = F(Current state)
  - Next state = F(Inputs, current state)

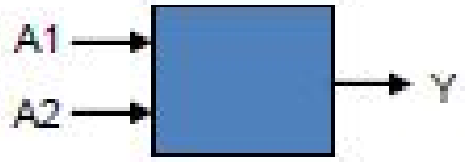


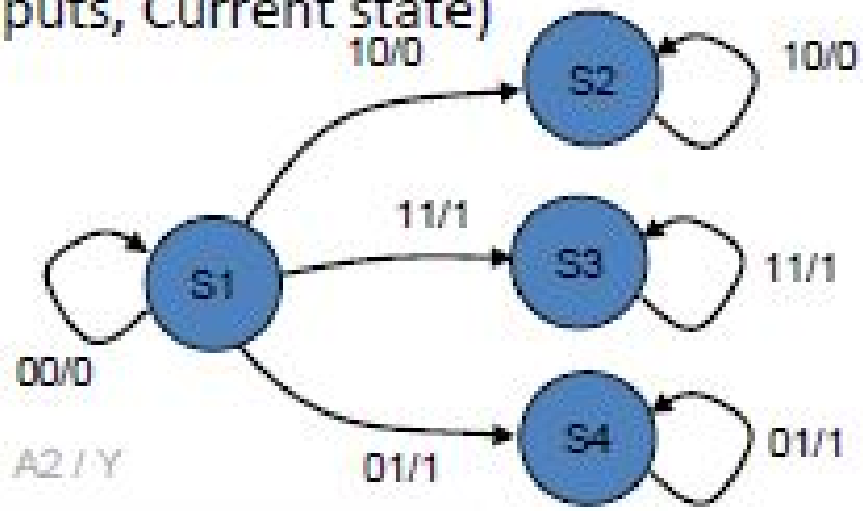
Moore FSM:



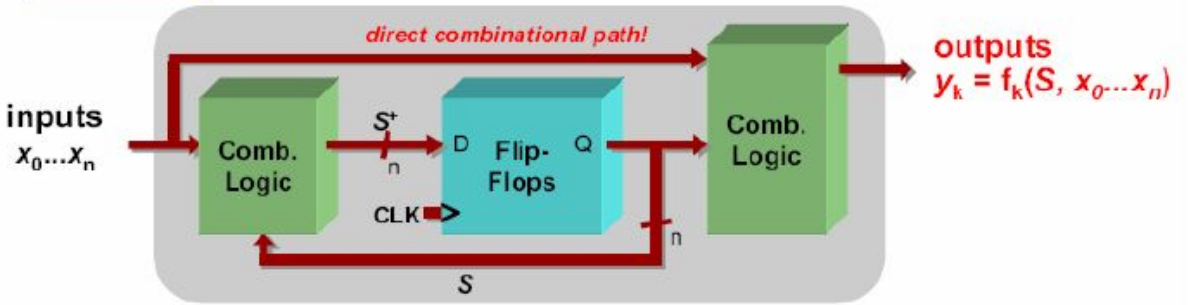
Mealy vs. Moore

- Mealy
  - Out = F(Inputs, Current state)
  - Next state = F(Inputs, Current state)





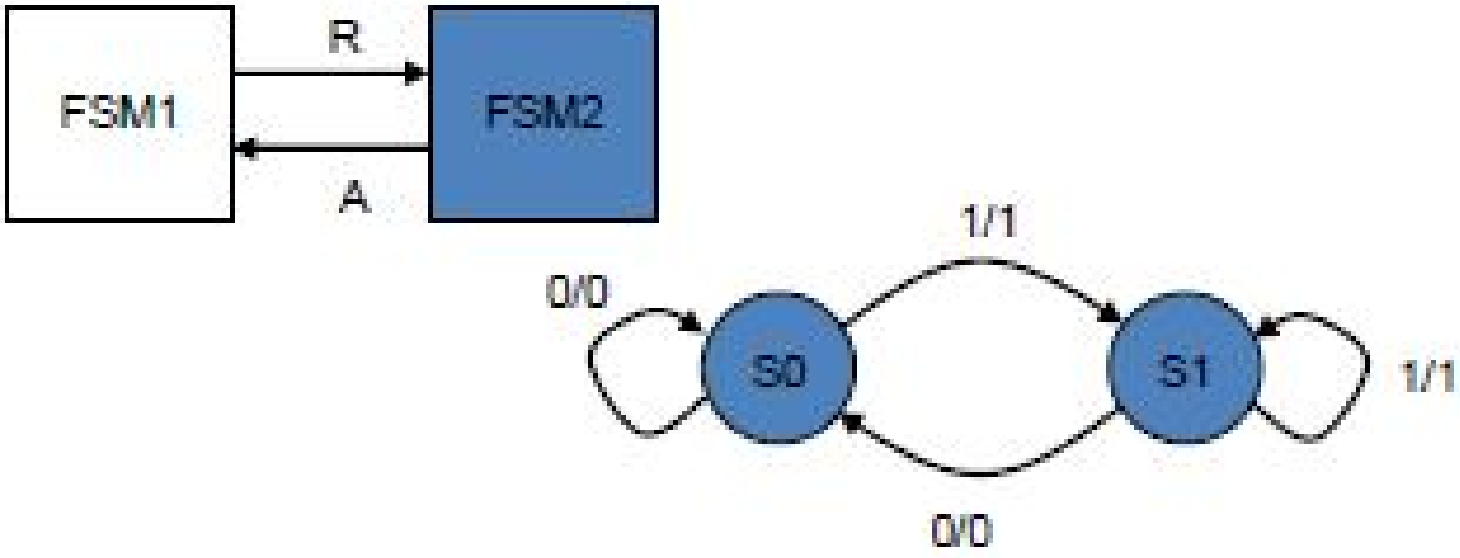
Mealy FSM:



Mealy vs. Moore

- In some situations a Mealy machine can be specified and implemented using less states because it is capable of producing different outputs in a given state.
- In some situations a system using a Mealy machine can be faster because an output may be produced immediately instead of at the next clock tick.
- A Moore machine produces glitch free outputs.
- The outputs from a Moore machine are available to its environment for almost a clock cycle, and in some situations this may allow using a faster cock.

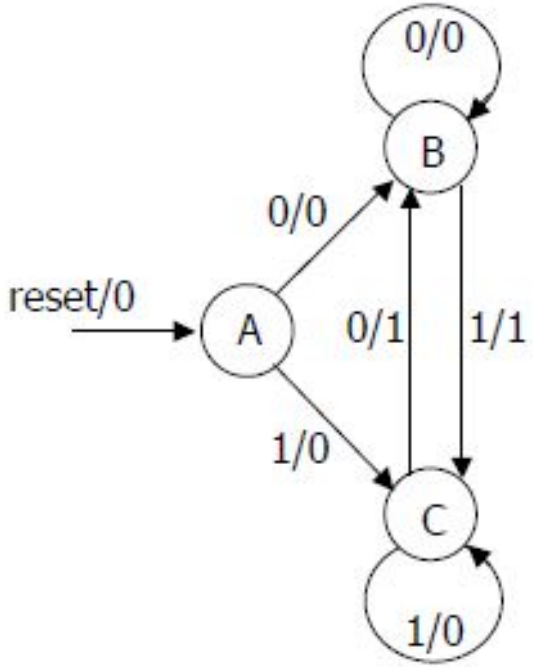
Mealy



- When in s0, a Mealy machine may produce A->1 immediately in response to R->1

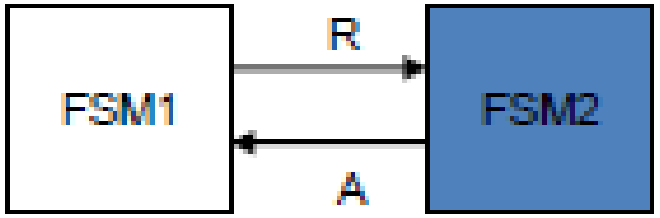
Specifying Outputs for a Mealy Machine

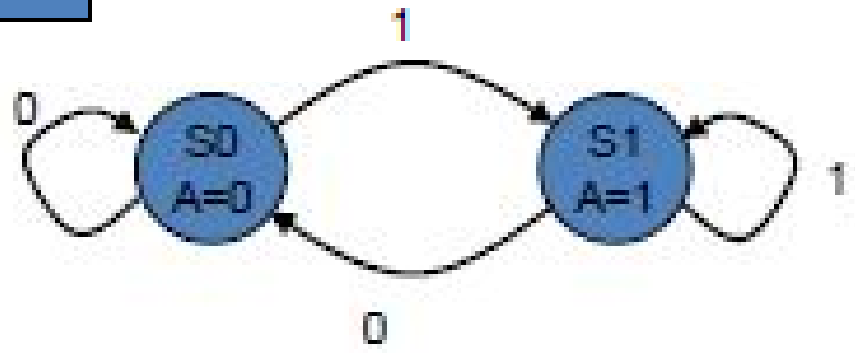
- Output is function of state and inputs
  - Specify output on transition arc between states
  - Example: sequence detector for 01 or 10



reset	input	current state	next state	output
1	–	–	A	0
0	0	A	B	0
0	1	A	C	0
0	0	B	B	0
0	1	B	C	1
0	0	C	B	1
0	1	C	C	0

Moore

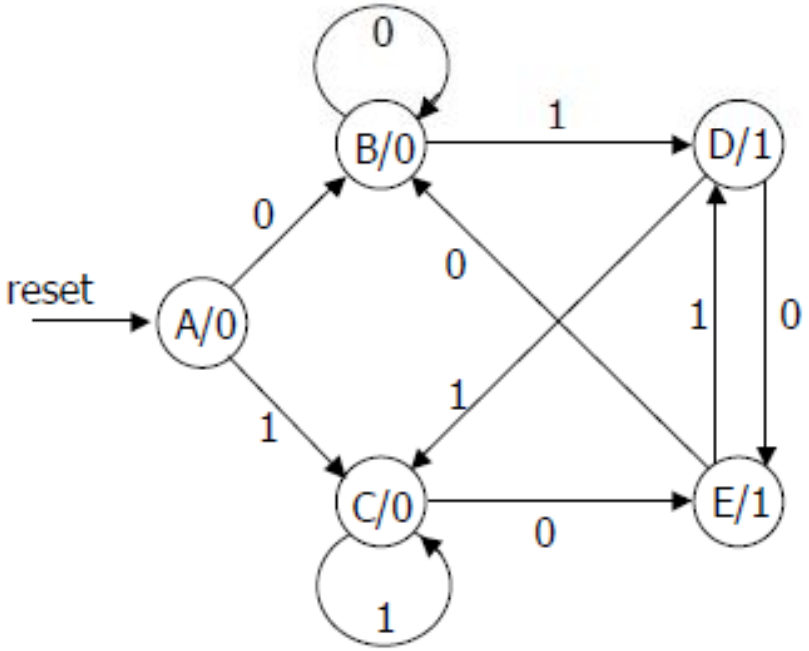




- … a Moore machine is not able to produce A->1 until the next clock when it enters s1

Specifying Outputs for a Moore Machine

- Output is only function of state
  - Specify in state bubble in state diagram
  - Example: sequence detector for 01 or 10

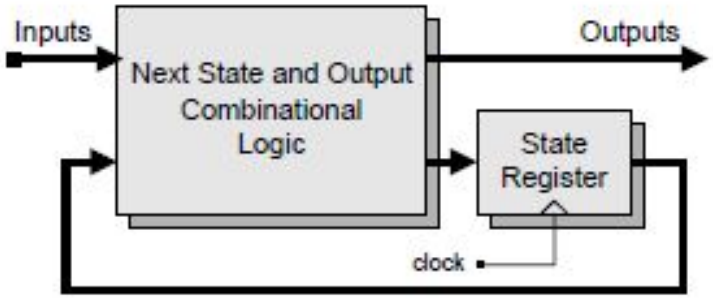


reset	input	current state	next state	output
1	–	–	A	
0	0	A	B	0
0	1	A	C	0
0	0	B	B	0
0	1	B	D	0
0	0	C	E	0
0	1	C	C	0
0	0	D	E	1
0	1	D	C	1
0	0	E	B	1
0	1	E	D	1

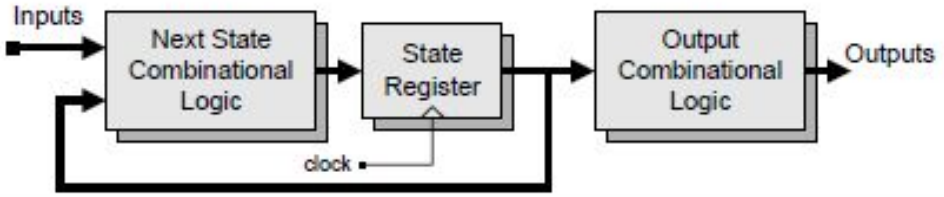
Comparison of Mealy and Moore Machines

- Mealy Machines tend to have less states
  - Different outputs on arcs (n^2) rather than states (n)
- Moore Machines are safer to use
  - Outputs change at clock edge (always one cycle later)
  - In Mealy machines, input change can cause output change as soon as logic is done – a big problem when two machines are interconnected – asynchronous feedback
- Mealy Machines react faster to inputs
  - React in same cycle – don't need to wait for clock
  - In Moore machines, more logic may be necessary to decode state into outputs – more gate delays after

Mealy machine



Moore machine

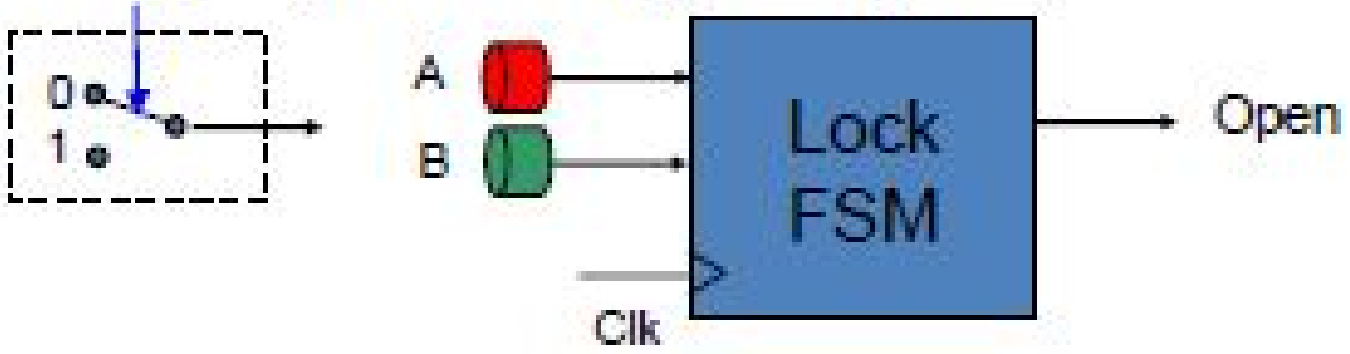


State Machine Analysis Steps

- Assumption: Starting point is a logic diagram.
- 1. Determine next-state function and output function
- 2. Construct state (transition) table
  - For each state/input combination, determine the excitation value.Using the characteristic equation, determine the corresponding next-state values (trivial with DFF's)
- 3. Construct output table
  - For each state/input combination, determine the output value (Can be combined with state table.)
- 4. Draw state diagram

State Graph for lock-FSM

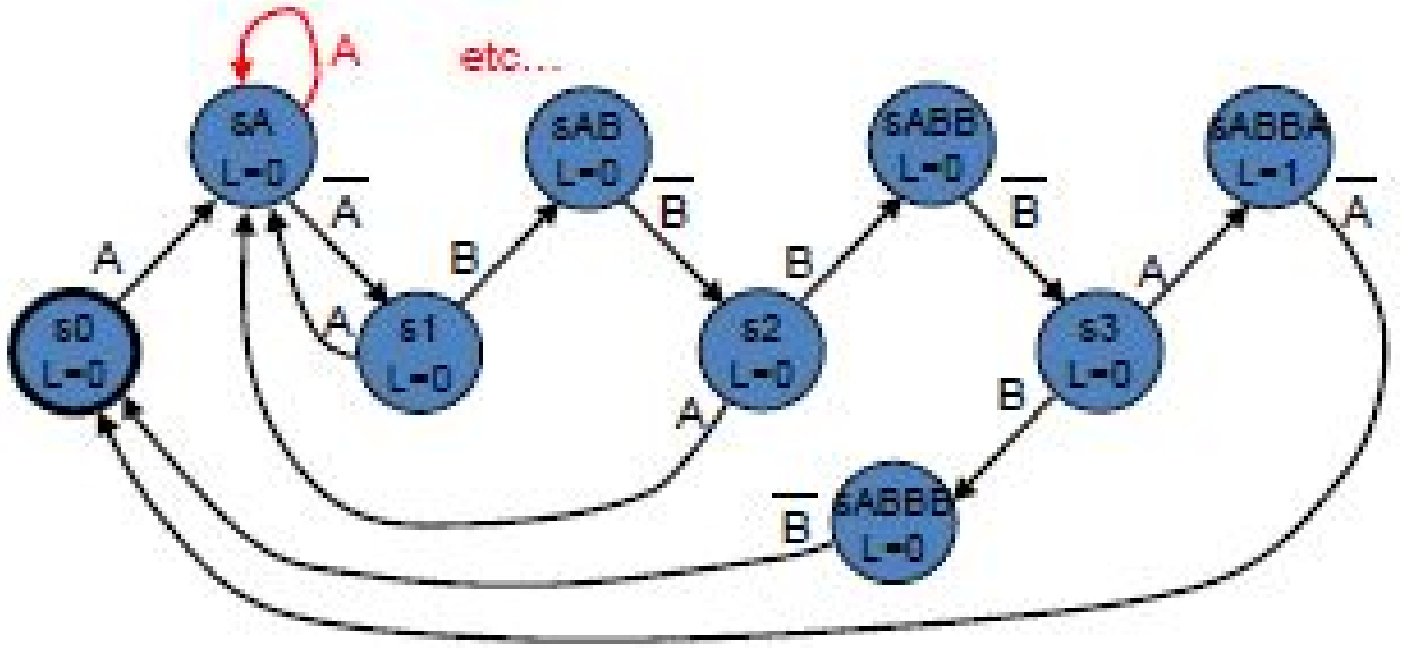
- Pushing: * { A; B; B; A } => Open



- Draw a state graph for the Lock-FSM

State Graph for lock-FSM

Assuming that A and B are never pressed at the same time …



Hmmm: Is this a Mealy FSM or a Moore FSM?

Example

- Sequence Detector
  - We will create a sequence detector for bit a given sequence. We will develop a sequence detector using Mealy/Moore machine model.
  - As an illustrative example a sequence detector for bit sequence ‘1011’ is described. Every clock-cycle a value will be sampled, if the sequence ‘1011’ is detected a ‘1’ will be produced at the output for 1 clock-cycle.

Mealy FSM Sequence Detector

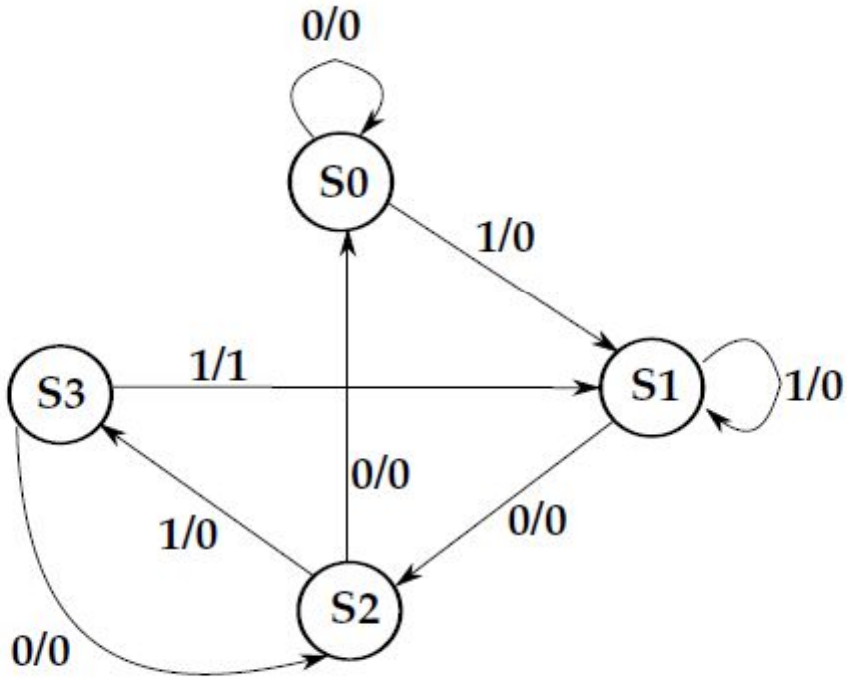


Figure 1: Mealy State Machine for Detecting a Sequence of '1011'

- When in initial state the machine gets the input of ‘1’ it jumps to the next state with the output equal to ’0’. If the input is ‘0’ it stays in the same state.
- When in 2nd state the machine gets an input of ‘0’ it jumps to the 3rd state with the output equal to ‘0’. If it gets an input of ‘1’ it stays in the same state.
- When in the 3rd state the machine gets an input of ‘1’ it jumps to the 4th state with the output equal to ‘0’. If the input received is ‘0’ it goes back to the initial state.
- When in the 4th state the machine gets an input of ‘1’ it jumps back to the 2nd state, with the output equal to ‘1’. If the input received is ‘0’ it goes back to the 3rd state.

Moore FSM Sequence Detector

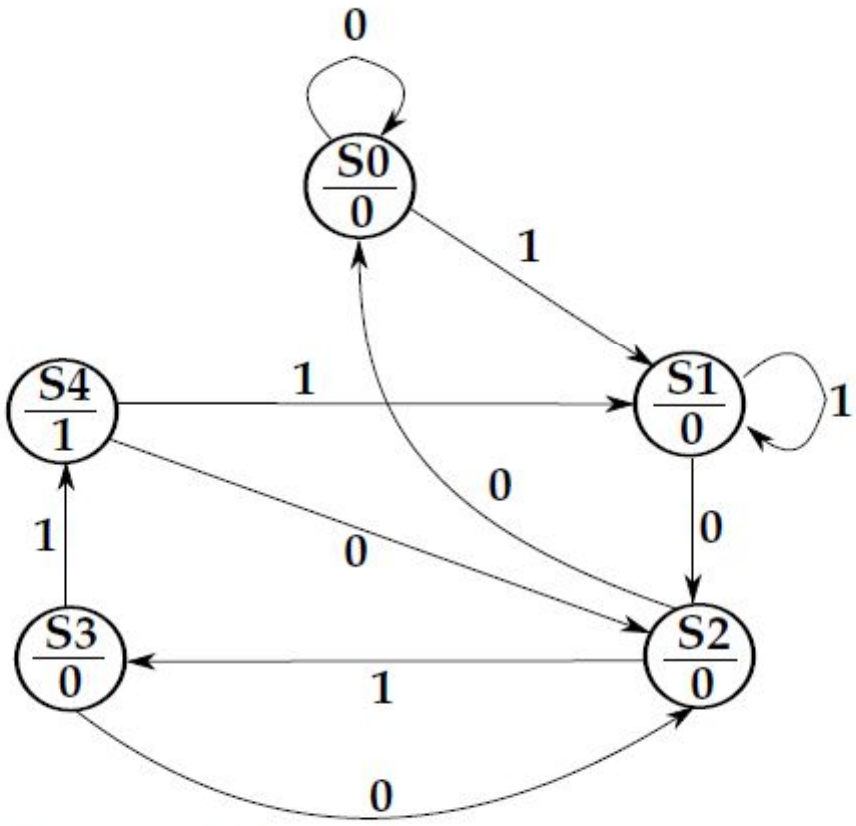


Figure 2: Moore State Machine for Detecting a Sequence of '1011'

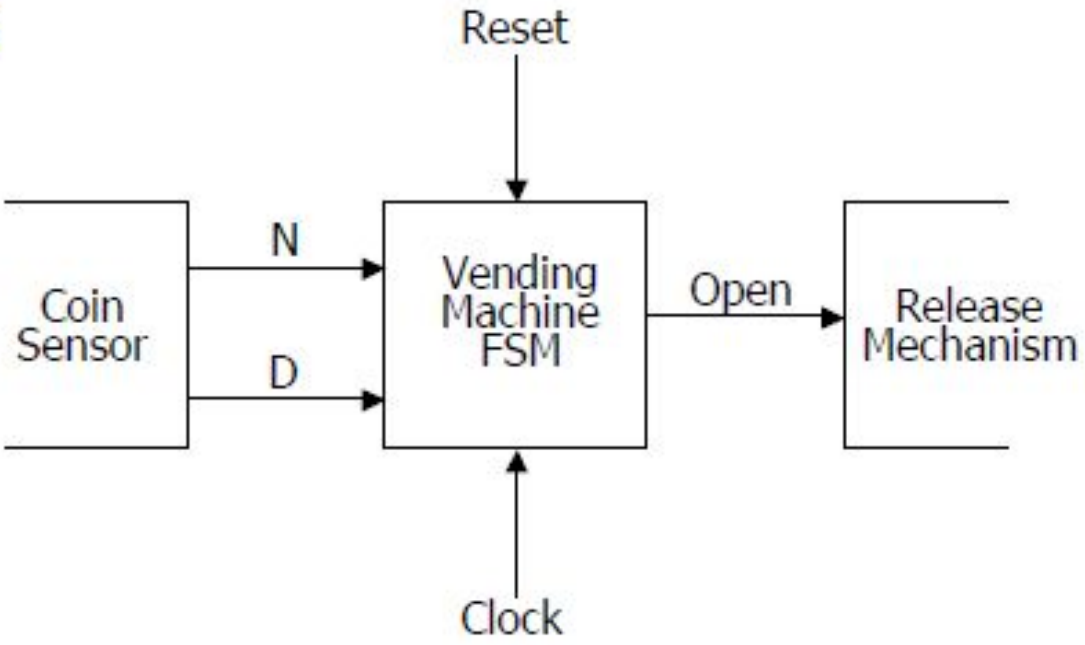
- In initial state the output of the detector is ‘0’. When machine gets the input of ‘1’ it jumps to the next state. If the input is ‘0’ it stays in the same state.
- In 2nd state the output of the detector is ‘0’. When machine gets an input of ‘0’ it jumps to the 3rd state. If it gets an input of ‘1’ it stays in the same state.
- In the 3rd state the output of the detector is ‘0’. When machine gets an input of ‘1’ it jumps to the 4th state. If the input received is ‘0’ it goes back to the initial state.
- In the 4th state the output of the detector is ‘0’. When machine gets an input of ‘1’ it jumps to the 5th state. If the input received is ‘0’ it goes back to the 3rd state.
- In the 5th state the output of the detector is ‘1’. When machine gets an input of ‘0’ it jumps to the 3rd state, otherwise it jumps to the 2nd state.

FSM Exercise(Assignment)

- Write a verilog code with testbench in sequence detector both in Mealy and Moore.
- Show the simulation of the design.

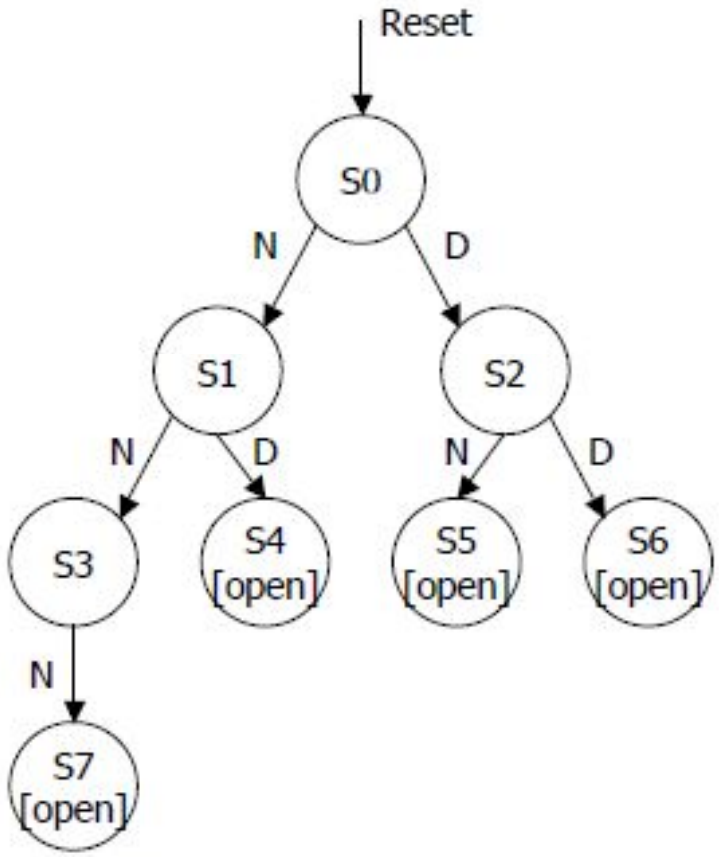
Example: Vending Machine

- Release item after 15 cents are deposited
- Single coin slot for dimes, nickels
- No change



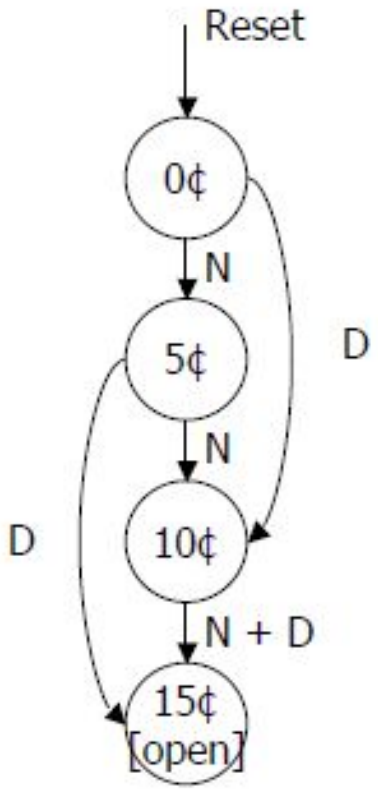
Example: Vending Machine (cont'd)

- Suitable Abstract Representation
  - Tabulate typical input sequences:
    - 3 nickels
    - nickel, dime
    - dime, nickel
    - two dimes
  - Draw state diagram:
    - Inputs: N, D, reset
    - Output: open chute
  - Assumptions:
    - Assume N and D asserted for one cycle
    - Each state has a self loop for N = D = 0 (no coin)



Example: Vending Machine (cont'd)

- Minimize number of states - reuse states whenever possible



present state	inputs D	inputs N	next state	output open
0¢	0	0	0¢	0
	0	1	5¢	0
	1	0	10¢	0
	1	1	–	–
5¢	0	0	5¢	0
	0	1	10¢	0
	1	0	15¢	0
	1	1	–	–
10¢	0	0	10¢	0
	0	1	15¢	0
	1	0	15¢	0
	1	1	–	–
15¢	–	–	15¢	1

symbolic state table

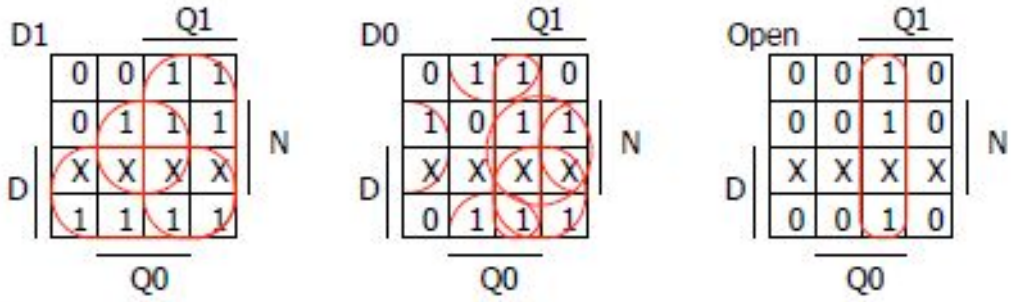
Example: Vending Machine (cont'd)

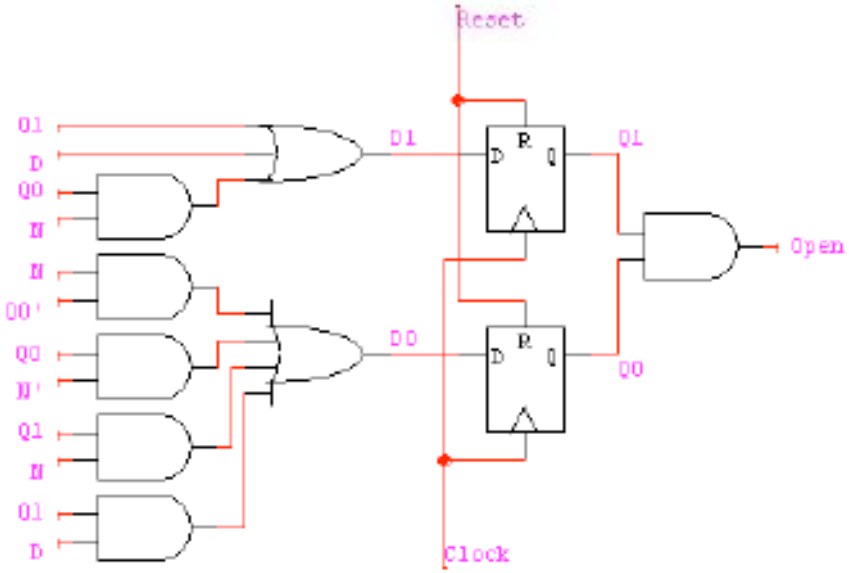
- Uniquely Encode States

present state Q1	Q0	inputs D	N	next state D1	D0	output open
0	0	0	0	0	0	0
		0	1	0	1	0
		1	0	1	0	0
		1	1	–	–	–
0	1	0	0	0	1	0
		0	1	1	0	0
		1	0	1	1	0
		1	1	–	–	–
1	0	0	0	1	0	0
		0	1	1	1	0
		1	0	1	1	0
		1	1	–	–	–
1	1	–	–	1	1	1

Example: Vending Machine (cont'd)

- Mapping to Logic





$D1 = Q1 + D + Q0\ N$

$D0 = Q0' N + Q0 N' + Q1 N + Q1 D$

$OPEN = Q1\ Q0$

Example: Vending Machine (cont'd)

- One-hot Encoding

present state Q3	Q2	Q1	Q0	inputs D	N	next state D3	D2	D1	D0	output open
0	0	0	1	0	0	0	0	0	1	0
				0	1	0	0	1	0	0
				1	0	0	1	0	0	0
				1	1	-	-	-	-	-
0	0	1	0	0	0	0	0	1	0	0
				0	1	0	1	0	0	0
				1	0	1	0	0	0	0
				1	1	-	-	-	-	-
0	1	0	0	0	0	0	1	0	0	0
				0	1	1	0	0	0	0
				1	0	1	0	0	0	0
				1	1	-	-	-	-	-
1	0	0	0	-	-	1	0	0	0	1

D0 = Q0 D′ N′

D1 = Q0 N + Q1 D′ N′

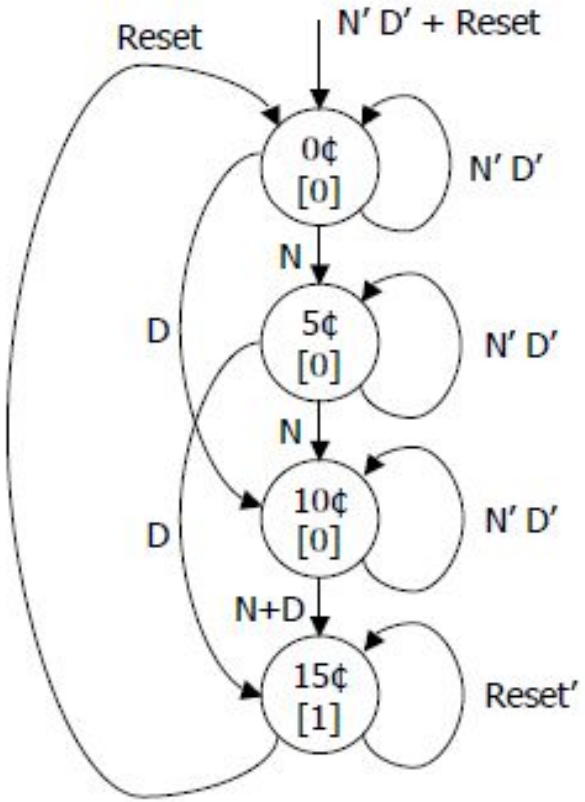
D2 = Q0 D + Q1 N + Q2 D′ N′

D3 = Q1 D + Q2 D + Q2 N + Q3

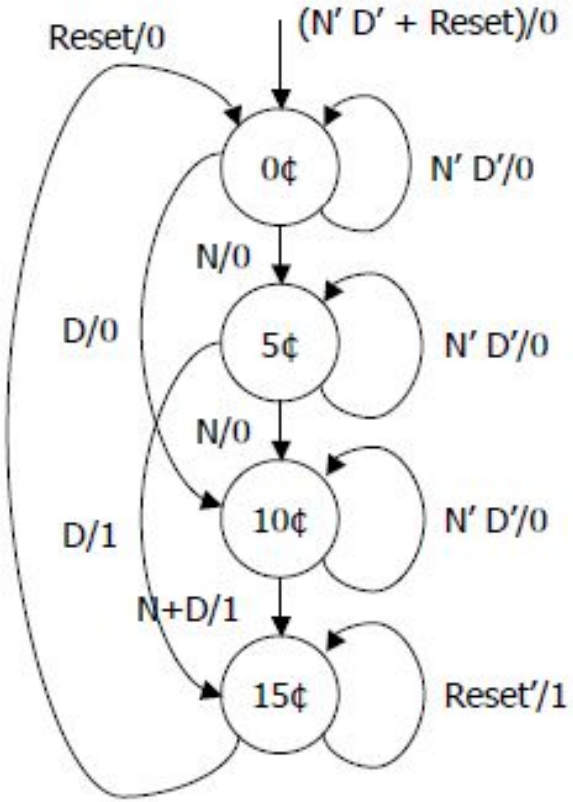
OPEN = Q3

Equivalent Mealy and Moore State Diagrams

- Moore machine
  - outputs associated with state

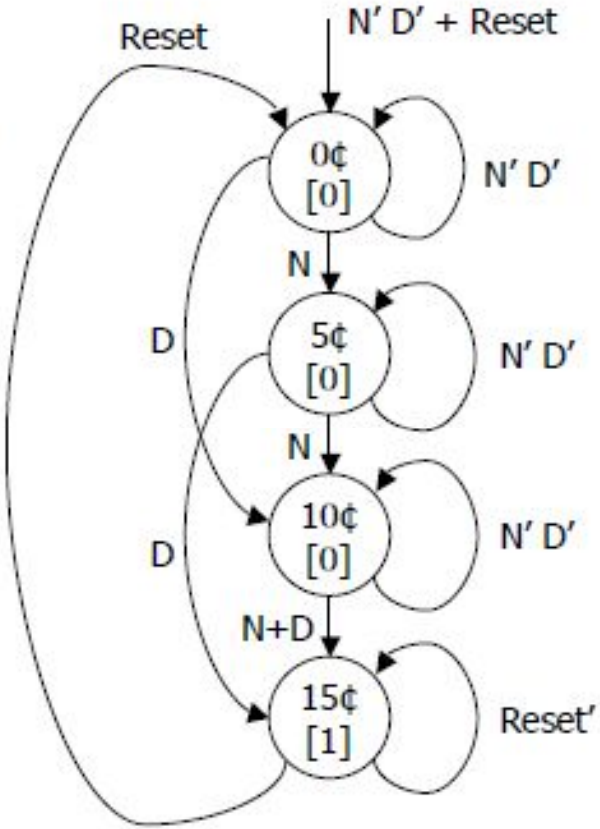


Mealy machine
outputs associated with transitions



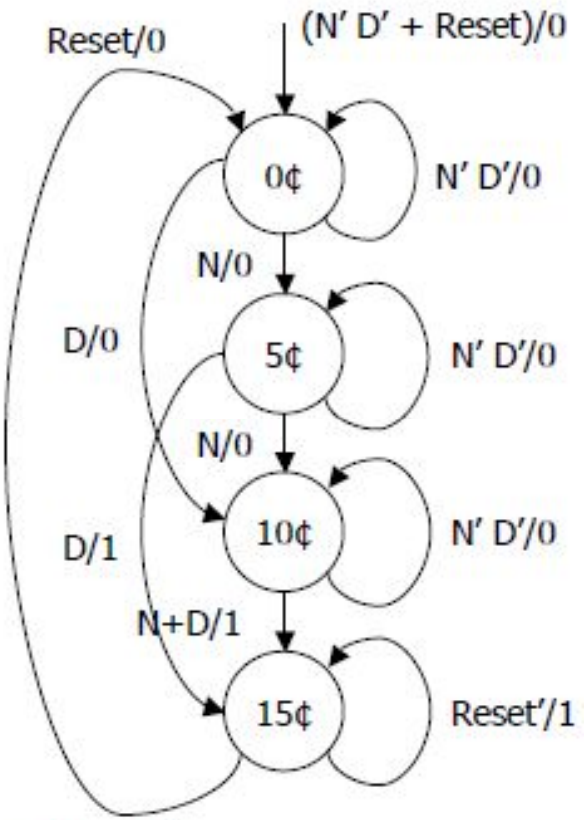
Moore Verilog FSM for Vending Machine

```
module vending (open, Clk, Reset, N, D);
  input Clk, Reset, N, D; output open;
  reg open; reg state;     // state register
  reg next_state;
  parameter zero = 0, five = 1, ten = 2, fifteen = 3;

  always @(N or D or state)
    case (state)
      zero: begin
              if (D)       next_state = five;
              else if (N)  next_state = ten;
              else         next_state = zero;
              open = 0;
            end
      …
      fifteen: begin
              if (!Reset)  next_state = fifteen;
              else         next_state = zero;
              open = 1;
            end
 endcase

  always @(posedge clk)
    if (Reset || (!N && !D))  state <= zero;
    else                      state <= next_state;

  endmodule
```



Mealy Verilog FSM for Vending Machine

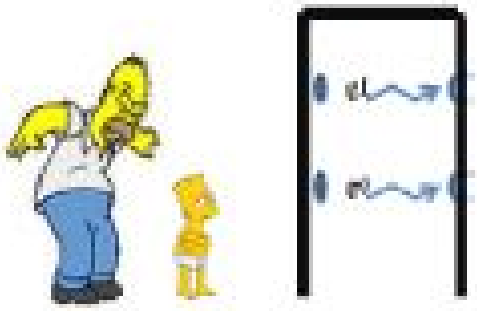
```
module vending (open, Clk, Reset, N, D);
  input Clk, Reset, N, D; output open;
  reg open; reg state;    // state register
  reg next_state; reg next_open;
  parameter zero = 0, five = 1, ten = 2, fifteen = 3;

  always @(N or D or state)
    case (state)
      zero: begin
              if (D) begin
                next_state = ten; next_open = 0;
              end
              else if (N) begin
                next_state = five; next_open = 0;
              end
              else begin
                next_state = zero; next_open = 0;
              end
            end
      …
    endcase

  always @(posedge clk)
    if (Reset || (!N && !D)) begin state <= zero; open <= 0; end
    else begin state <= next_state; open <= next_open; end
  endmodule
```



FSM Exercise(Project)

Marge wants to install an alarm that triggers as soon as somebody enters the kitchen. The alarm should have several alert levels.

- level0: Neither Homer nor Bart is in the kitchen
- level1: Bart but not Homer is in the kitchen
- level2: Homer but not Bart is in the kitchen
- level3: Homer and Bart are in the kitchen



To detect who enters or leaves the kitchen 2 sensors g1 and g0 are installed in the door frame as depicted. The sensors emit a '1' as soon as their reflection is interrupted. If Bart enters the kitchen only g0 will emit a '1'. Homer is always leaning forward when he is entering the kitchen, and, thus, g1 will always be interrupted before g0. Once they have decided to go into the kitchen they will go through the door. However, if they are in the kitchen they always can leave, e.g., level3 changes to level2. The size of Homers hips and belly prevent them from entering the kitchen simultaneously. The clock frequency is 1MHz.