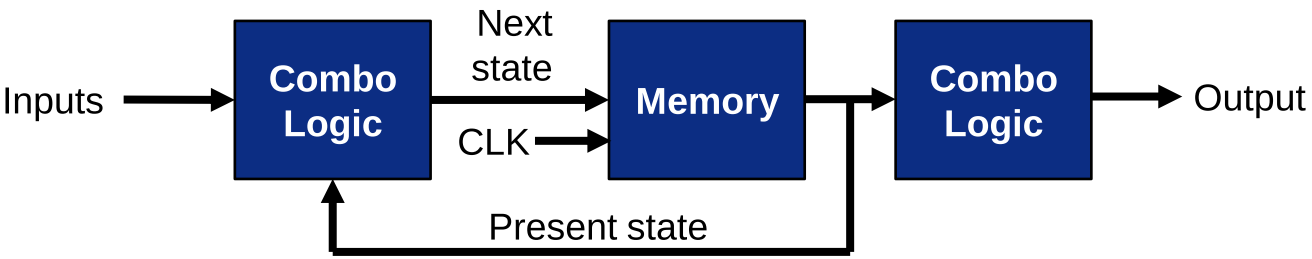
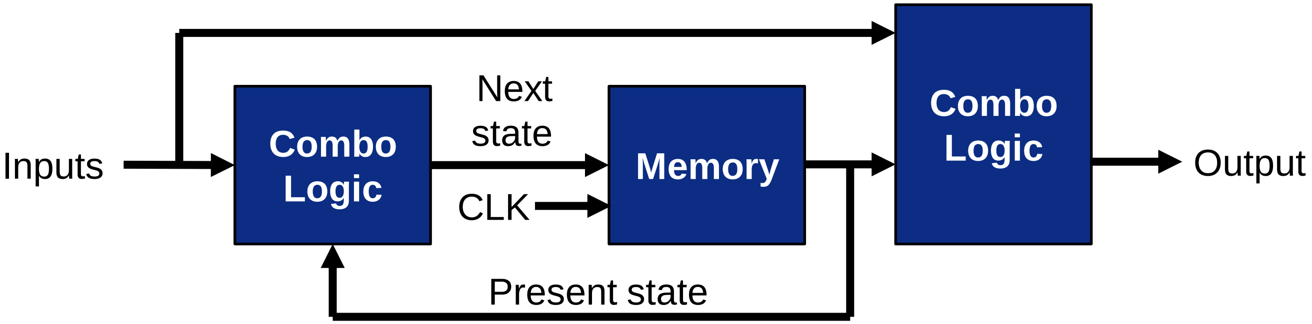
Finite State Machines

Structured two ways:

1. **Moore machine**



2. **Mealy machine**



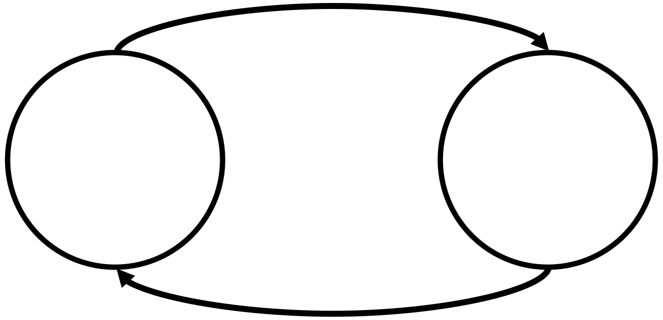
Modern FSM designs use Field Programmable Gate Arrays (FPGAs) which are programmed using VHDL or Verilog

State Transition Diagrams

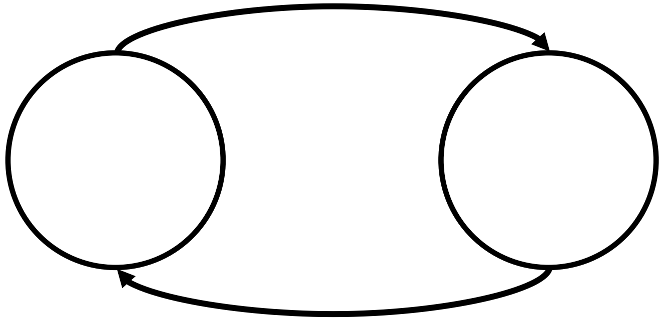
- Model FSMs using **state transition diagrams**

- These consist of:
  1. : all possible configurations of a system
  2. : connections between states
  3. : actions that trigger a transition

Moore machine example:



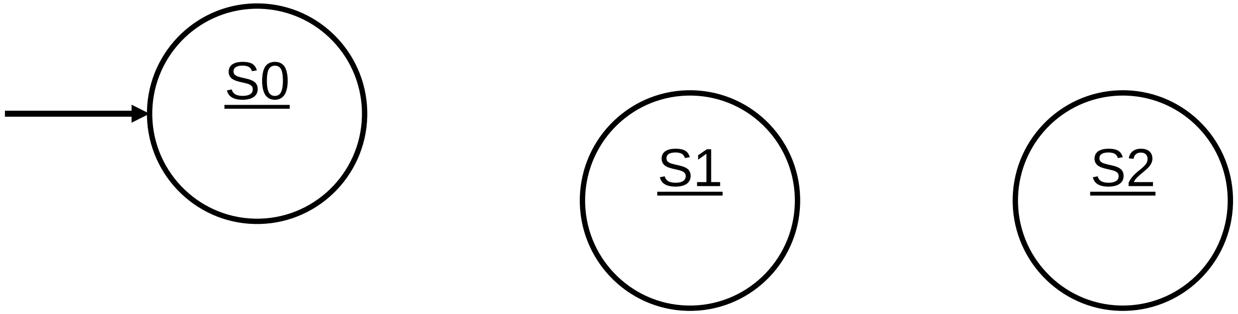
Mealy machine example:



State Transition Diagrams

Example 1: Design a Moore FSM to recognize the binary sequence '10'.

Output = 0:
1:



Meaning of states:

S0	S1	S2
elements of sequence observed	element observed	observed

State Transition Diagrams

Process for developing a solution:

1. Define/ the states (binary representation)

   /encode the input(s)

3. Define/encode the
4. Create a state transition
5. Create a mapping present state/input to the and
6. Express and as logic equations of the present states and
7. Implement with the required and

State Transition Diagrams

1. 3 states, so need 2 bits

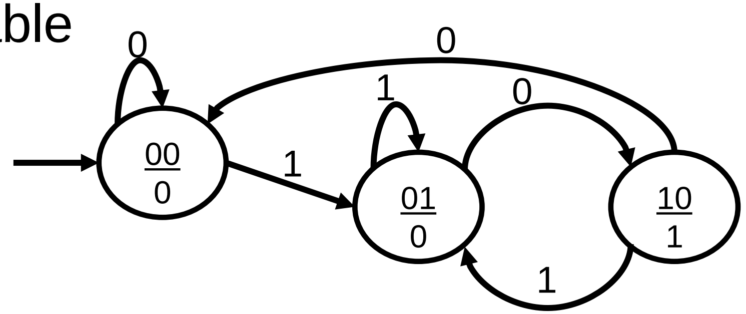
   2^{bits} = max # of states → 2^2 = 4 states

   S0 = 00 S1 = 01 S2 = 10

2. Input is already binary, x = 0 or 1

   (what if it was x = 1, 2, 3, 4?)

3. Define/encode outputs
4. Create the state transition diagram – done!
5. Create the table



Present State		Input	Next State		Output
q_1	q_0	x	q_1^+	q_0^+	z
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

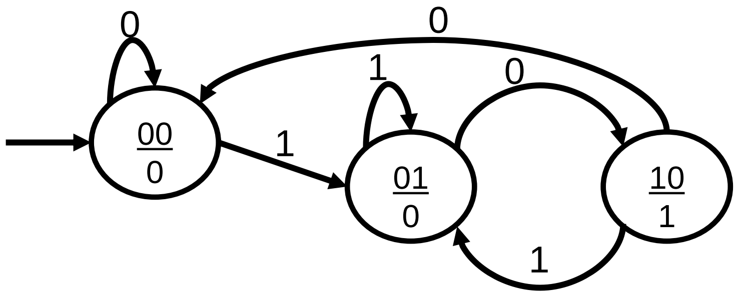
State Transition Diagrams

6. Create combinational logic for q_1^+, q_0^+, and z
   - Use SOP or POS based on situation

$q_1^+ =$

$q_0^+ =$

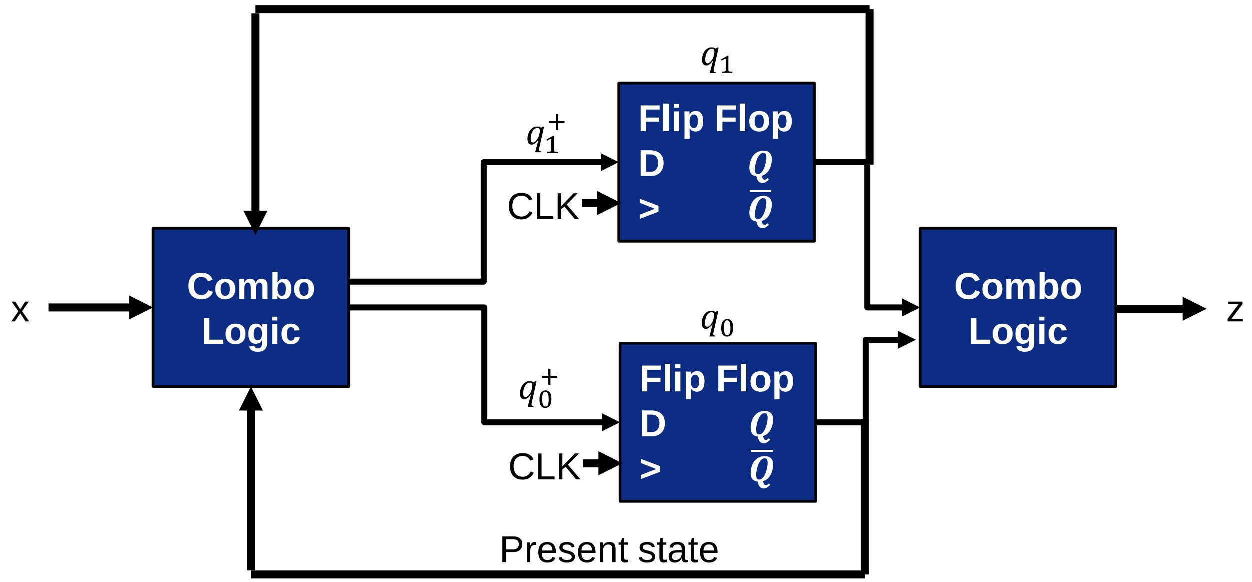
$z =$



Present State		Input	Next State		Output
q_1	q_0	x	q_1^+	q_0^+	z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	0	0	1
1	0	1	0	1	1
1	1	0	x	x	x
1	1	1	x	x	x

State Transition Diagrams

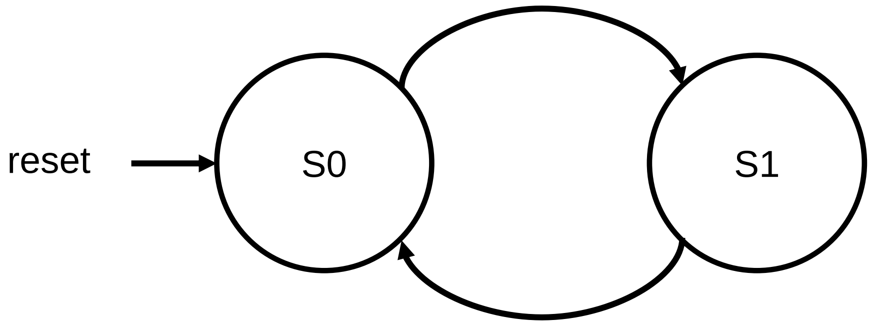
7. Implement with the required flip flop memory and logic gates



The D flip flop will pass its input to the output (Q) on either the rising edge or falling edge of the CLK.

State Transition Diagrams

- What would be an equivalent Mealy diagram?



Present	Input	Next	Output
q_0	x	q_0^+	z
0	0	0	0
0	1	1	0
1	0	0	1
1	1	1	0

Meaning of states:

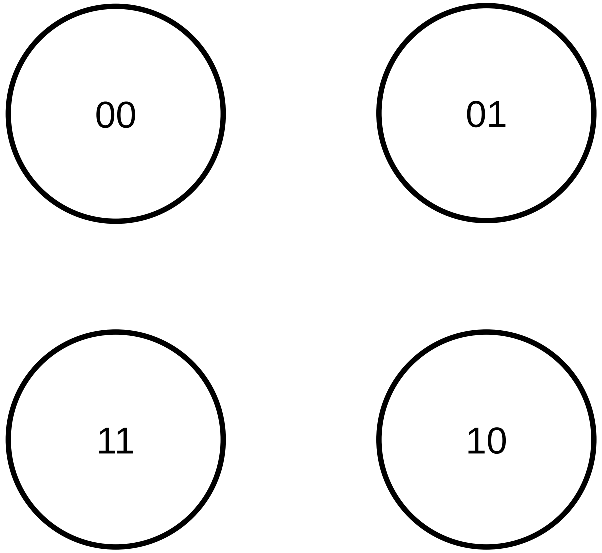
S0
elements of sequence observed

S1
element observed

Mealy implementation *could* have fewer states and use less area. However, a Moore implementation is generally "safer".

Example 2

- Implement a 2-bit counter from 0 to 3 that will count up when it receives a ‘1’ and holds the value when it receives a ‘0’
  - How many states? **4** How many bits? **4=2² → 2 bits**



Present State		Input	Next State		Output	
q_1	q_0	x	q_1^+	q_0^+	z_1	z_0
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

$q_1^+ =$

$z_1 =$

$z_0 =$

* Output = state, so this is a machine

* Same implementation idea as in previous example

Example 3

- A vending machine requires 20¢ to dispense. You can enter a N = nickel, D = dime, or Q = quarter and then get change. For change you can only get a nickel, a dime, or a nickel and a dime. All other additional change is credited. Design the FSM.

1. What are the states?

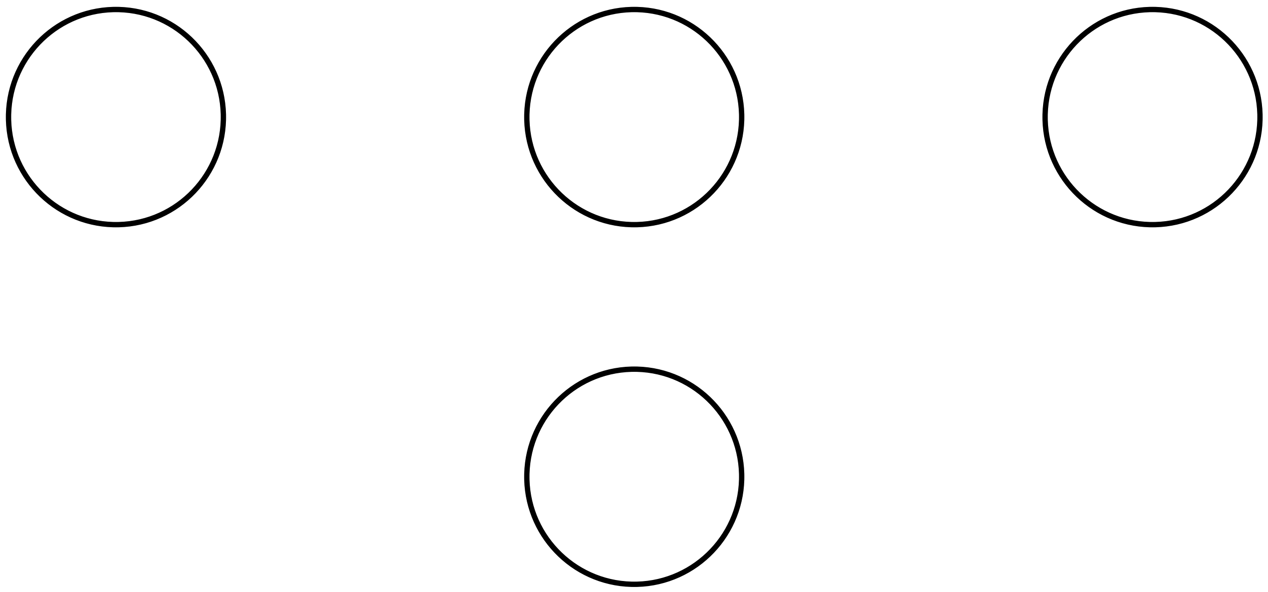
2. What are the inputs?

3. What are the outputs?

Example 3

- A vending machine requires 20¢ to dispense. You can enter a N = nickel, D = dime, or Q = quarter and then get change. For change you can only get a nickel, a dime, or a nickel and a dime. All other additional change is credited. Design the FSM.

4. Create a state transition diagram: INPUT/OUTPUT = Insert / Dispense Return Return



Example 3

- A vending machine requires 20¢ to dispense. You can enter a N = nickel, D = dime, or Q = quarter and then get change. For change you can only get a nickel, a dime, or a nickel and a dime. All other additional change is credited. Design the FSM.

5. Create a state transition table

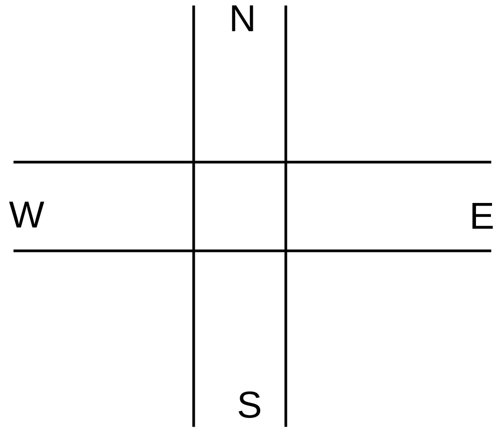
… we'll skip Steps 6-7.

	Present State		Input		Next State		Output		
	q_1	q_0	x_1	x_2	q_1^+	q_0^+	z_2	z_1	z_0
0	0	0	0	0					
1	0	0	0	1					
2	0	0	1	0					
3	0	0	1	1					
4	0	1	0	0					
5	0	1	0	1					
6	0	1	1	0					
7	0	1	1	1					
8	1	0	0	0					
9	1	0	0	1					
10	1	0	1	0					
11	1	0	1	1					
12	1	1	0	0					
13	1	1	0	1					
14	1	1	1	0					
15	1	1	1	1					

Example 4

- Example: We need to design an FSM for a traffic light intersection with sensors that tell us:

Input - X	
00	No traffic
01	N-S traffic
10	E-W traffic
11	Both traffic

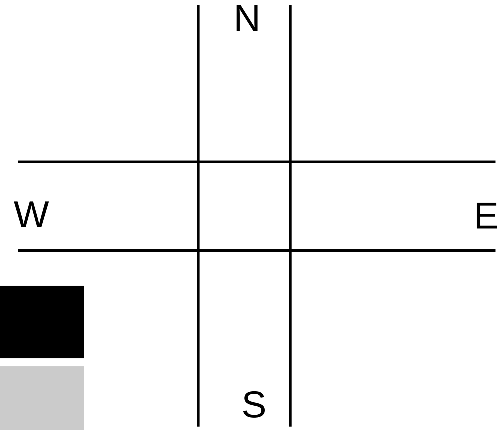


- We want N-S to stay green if there is No traffic or N-S traffic (reassess every 5 seconds)
- Otherwise, we want to transition to Yellow for 1 sec
- Then go to N-S red and E-W green and stay if there is E-W traffic (reassess every 5 seconds
- Otherwise transition to yellow for 1 sec
- Then go to N-S green and E-W red
- Repeat

Example 4

- Example: We need to design an FSM for a traffic light intersection with sensors that tell us:

Input - X	
00	No traffic
01	N-S traffic
10	E-W traffic
11	Both traffic



States	
S0 – 00	NS green, EW red
S1 – 01	NS yellow, EW red
S2 – 10	NS red, EW green
S3 – 11	NS red, EW yellow

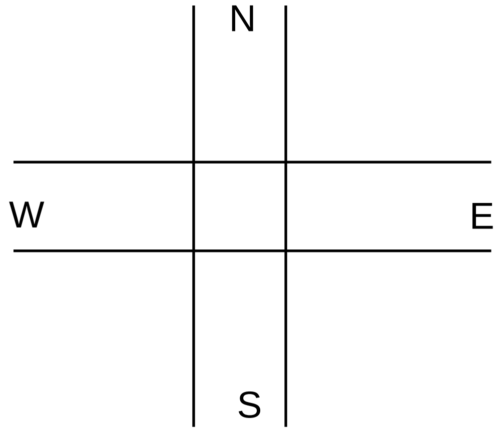
- Step 1: Define/encode the states.
  - Need 4 states
- Steps 2-3: Define/encode the inputs and outputs
  - Outputs are the states (so… Moore or Mealy?)

Example 4

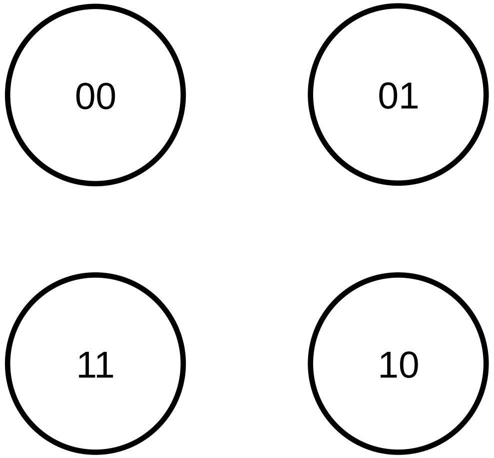
- Example: We need to design an FSM for a traffic light intersection with sensors that tell us:

Input - X	
00	No traffic
01	N-S traffic
10	E-W traffic
11	Both traffic

States	
S0 – 00	NS green, EW red, Delay 5s
S1 – 01	NS yellow, EW red, Delay 1s
S2 – 10	NS red, EW green, Delay 5s
S3 – 11	NS red, EW yellow, Delay 1s



- Step 4: Create State Transition Diagram



Example 4

- Step 5: Create State Transition Table
- Step 6: Express next states and outputs as expressions of the present states and inputs.

$q_1^+ =$

$q_0^+ =$

	Present State		Input		Next State		Output	
	q_1	q_0	x_1	x_0	q_1^+	q_0^+	z_1	z_0
0	0	0	0	0				
1	0	0	0	1				
2	0	0	1	0				
3	0	0	1	1				
4	0	1	0	0				
5	0	1	0	1				
6	0	1	1	0				
7	0	1	1	1				
8	1	0	0	0				
9	1	0	0	1				
10	1	0	1	0				
11	1	0	1	1				
12	1	1	0	0				
13	1	1	0	1				
14	1	1	1	0				
15	1	1	1	1				

Example 4

- Step 7: Implement with gates and flip flops
  - Drawing the logic gates for the combo logic will get messy fast, so we'll just go with "Combo Logic" boxes representing all of it.