CME 2003
Digital Logic

ALGORITHMIC STATE MACHINES

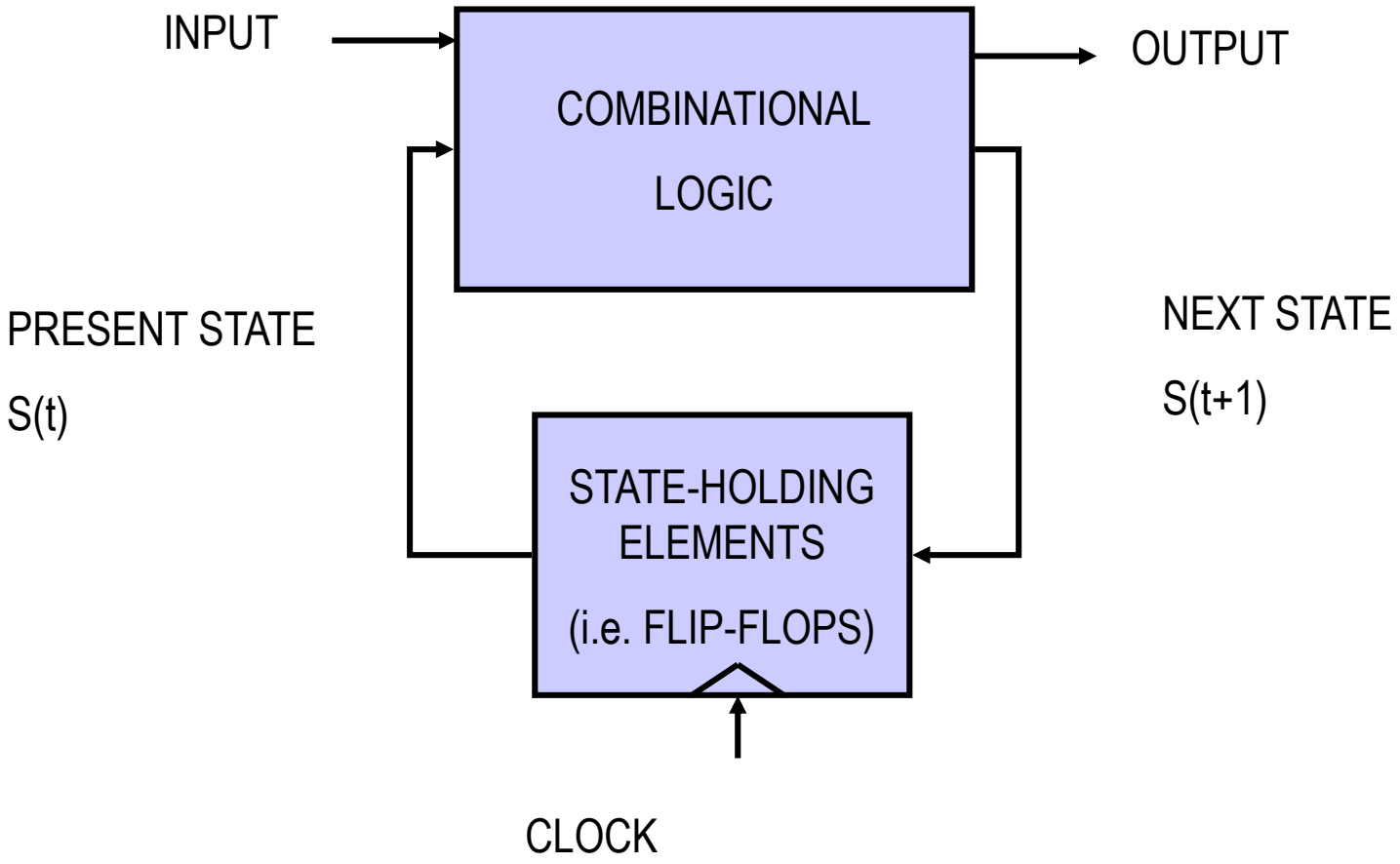
Şerife YILMAZ

Introduction to Sequential Logic

- Output depends on current as well as past inputs
  - Depends on the history
  - Have “memory” property
- Sequential circuit consists of
    - Combinational circuit
    - Feedback circuit
  - Past input is encoded into a set of state variables
    - Uses feedback (to feed the state variables)
      - Simple feedback
      - Uses flip flops

Introduction…

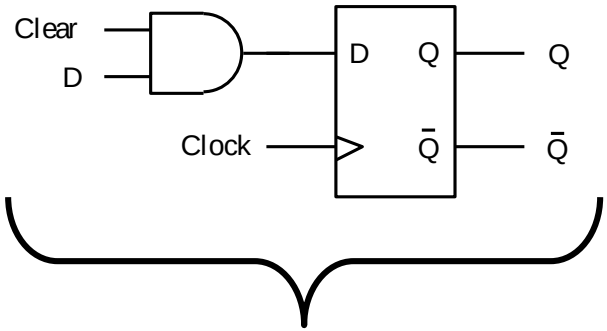
Main components of a typical synchronous sequential circuit (synchronous = uses a clock to keep circuits in lock step)

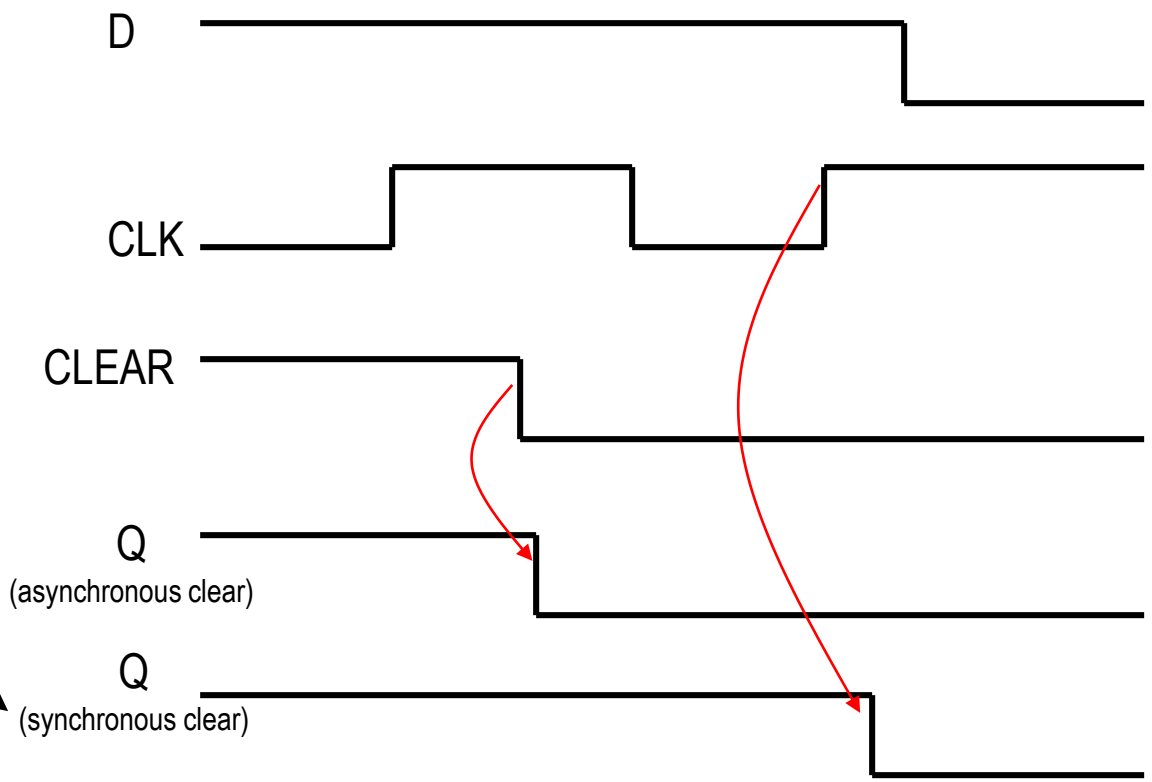


State-Holding Memory Elements

- Latch versus Flip Flop
  - Latches are level-sensitive: whenever clock is high, latch is transparent
  - Flip-flops are edge-sensitive: data passes through (i.e. data is sampled) only on a rising (or falling) edge of the clock
  - Latches cheaper to implement than flip-flops
  - Flip-flops are easier to design with than latches
- In this course, primarily use D flip-flops

D Flip-Flop with Synchronous Clear

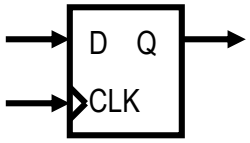




- Asynchronous active-low clear: Q immediately clears to 0
- Synchronous active-low clear: Q clears to 0 on rising-edge of clock

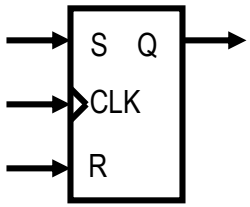
Other Types of Flip-Flops

D Flip-Flop



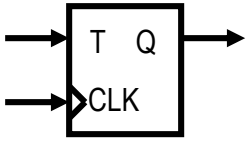
D	Q(t+1)
0	0
1	1

Set-Reset (SR) Flip-Flop



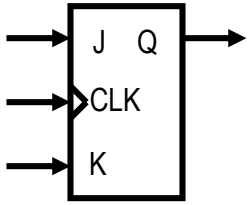
S	R	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	not allowed

Toggle (T) Flip-Flop



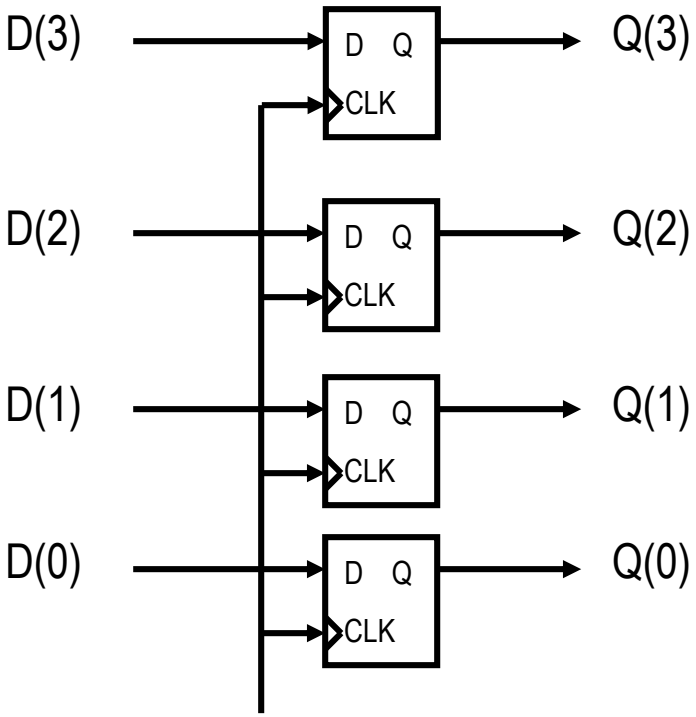
T	Q(t+1)
0	Q(t)
1	$\overline{Q(t)}$

JK Flip-Flop



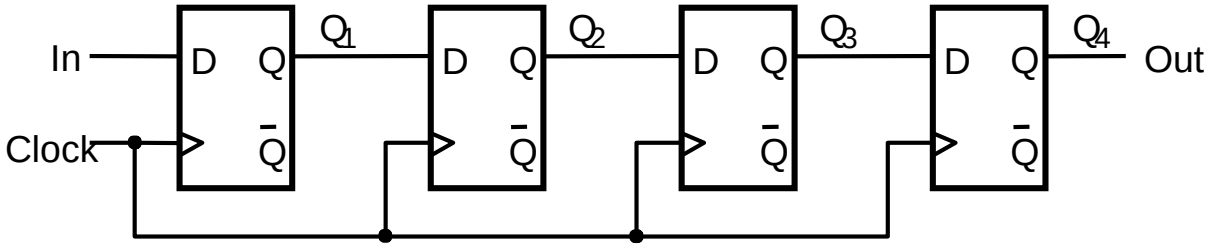
J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	$\overline{Q(t)}$

Register



- In typical nomenclature, a register is a name for a collection of flip-flops used to hold a bus.

Shift Register

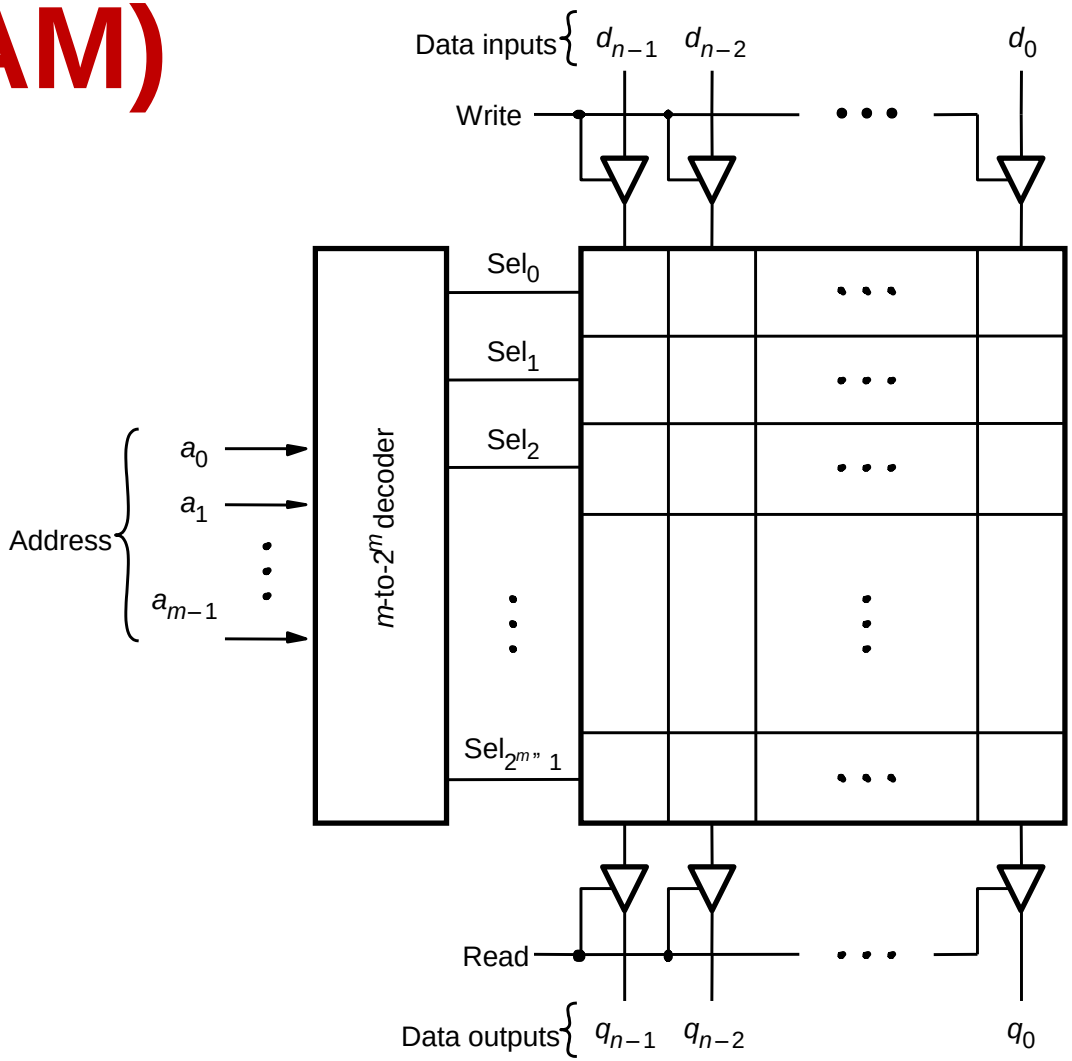


(a) Circuit

	In	Q_1	Q_2	Q_3	Q_4 = Out
t_0	1	0	0	0	0
t_1	0	1	0	0	0
t_2	1	0	1	0	0
t_3	1	1	0	1	0
t_4	1	1	1	0	1
t_5	0	1	1	1	0
t_6	0	0	1	1	1
t_7	0	0	0	1	1

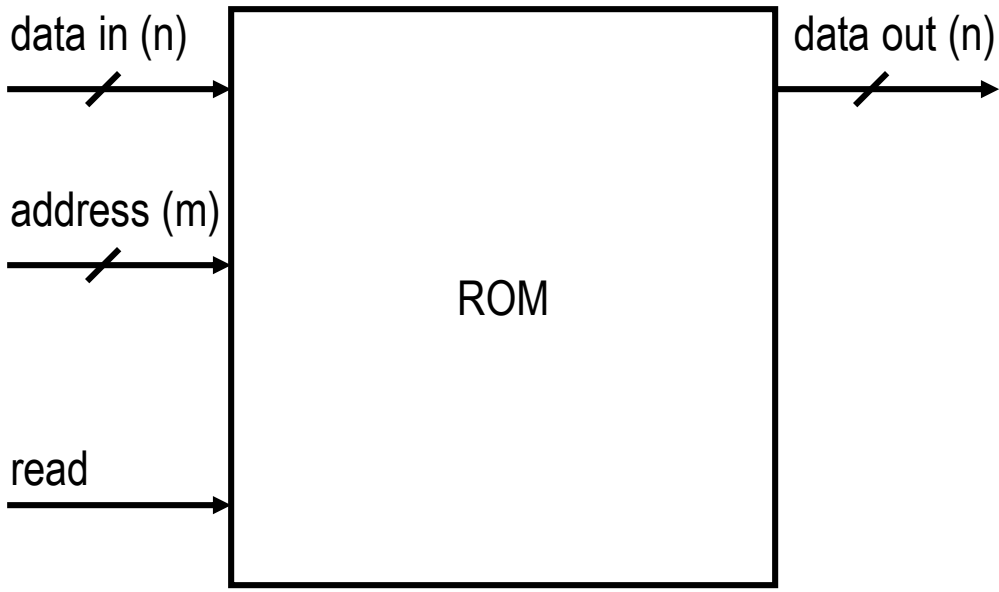
(b) A sample sequence

Random Access Memory (RAM)

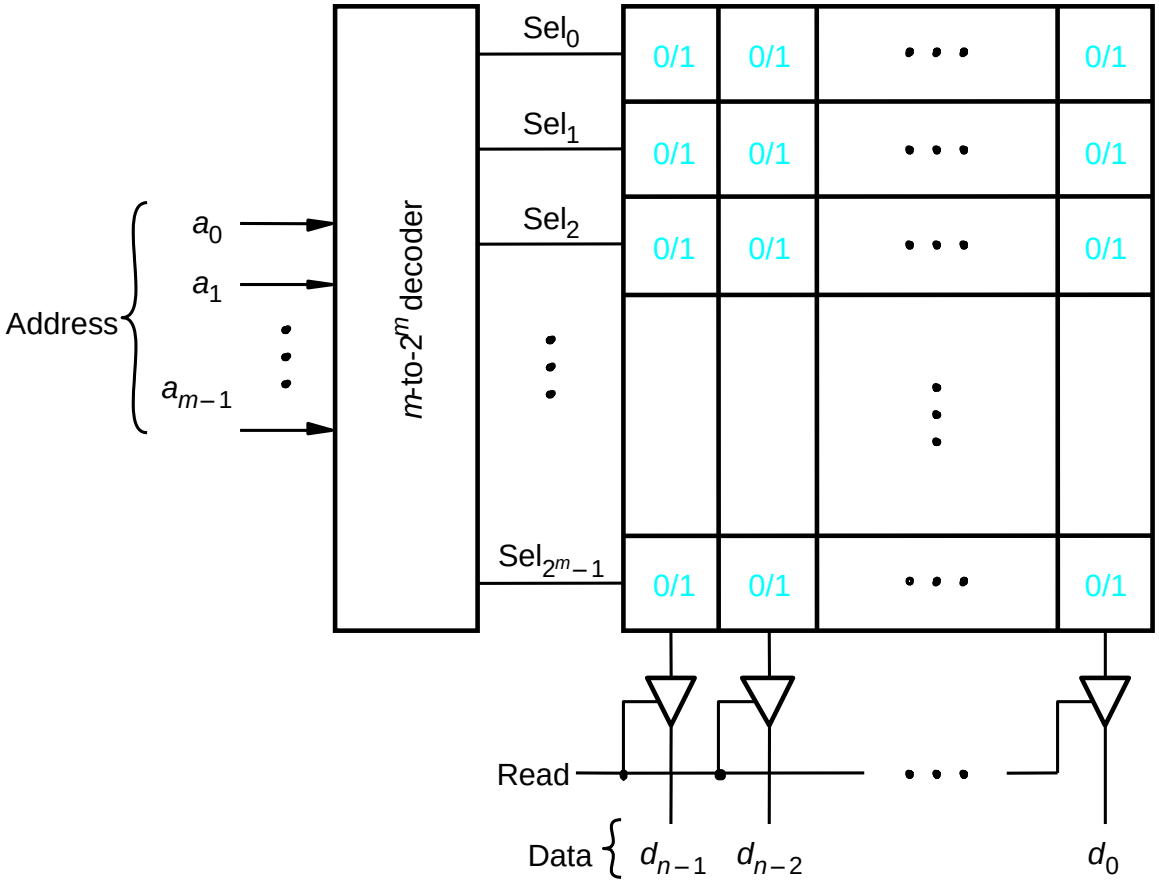


Read Only Memory (ROM)

- Similar to RAM except read only
- Addressable memory
- Can be synchronous (with clock) or asynchronous (no clock)



Read-Only Memory (ROM)



Finite State Machines (FSMs)

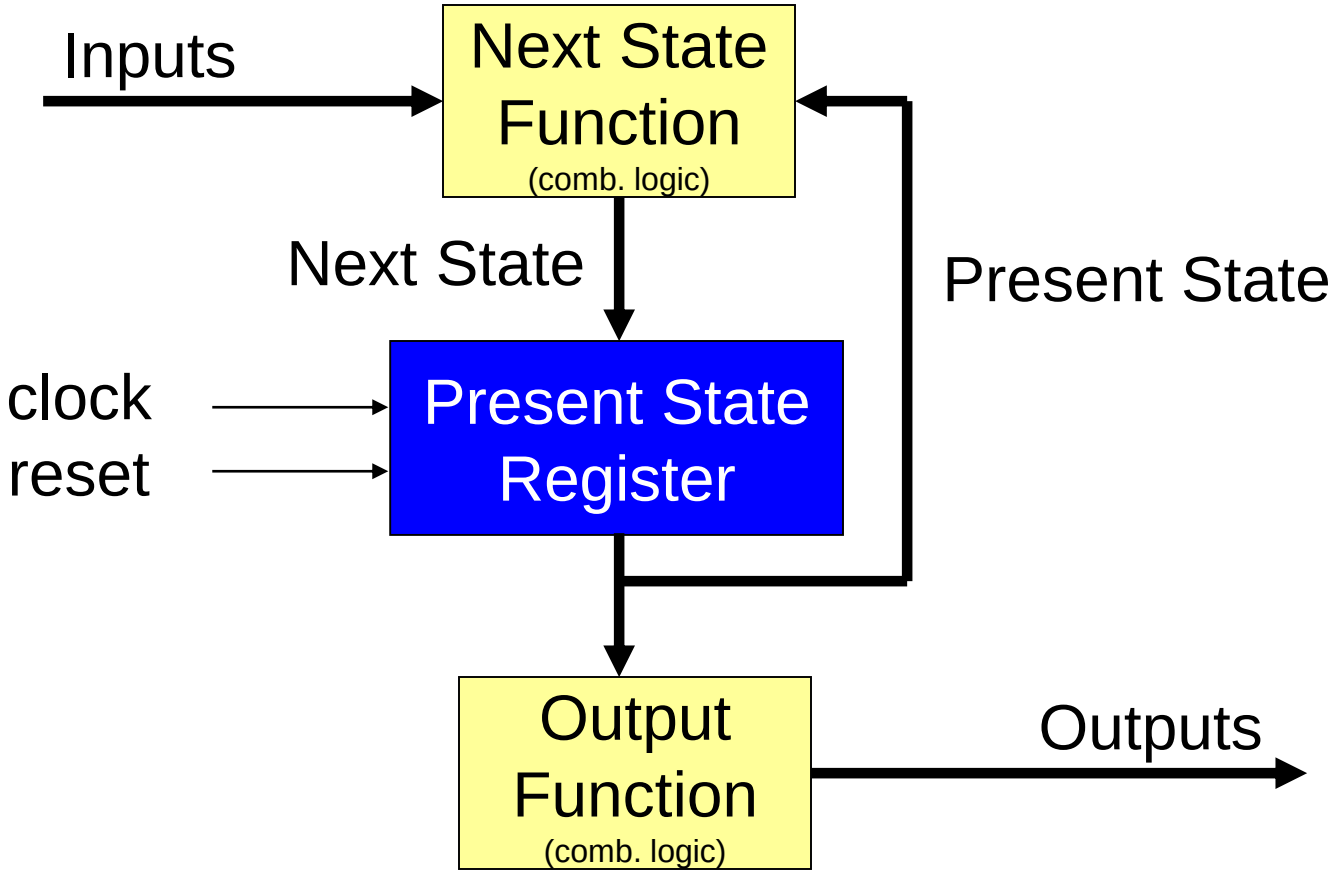
- Any Circuit with Memory Is a Finite State Machine
  - Even computers can be viewed as huge FSMs
- Design of FSMs Involves
  - Defining states
  - Defining transitions between states
  - Optimization / minimization
- Above Approach Is Practical for Simple FSMs Only

Mealy vs. Moore State Machines

- Finite State Machines (FSM) are of two types:
- Moore Machines
  - Next State = Function(Input, Present State)
  - Output = Function(Present State)
- Mealy Machines
  - Next State = Function(Input, Present State)
  - Output = Function(Input, Present State)

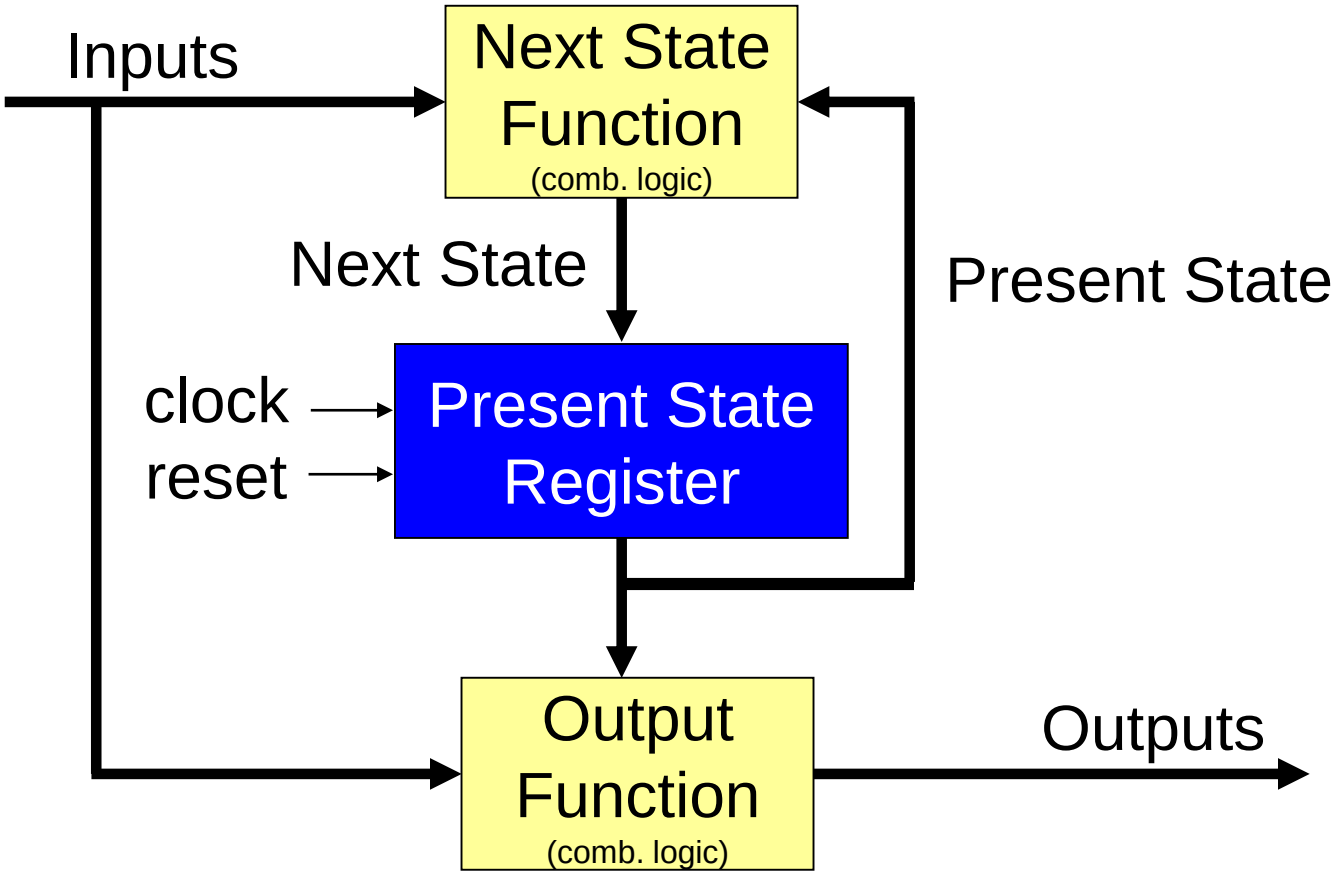
Moore FSM

- Output Is a Function of a Present State Only

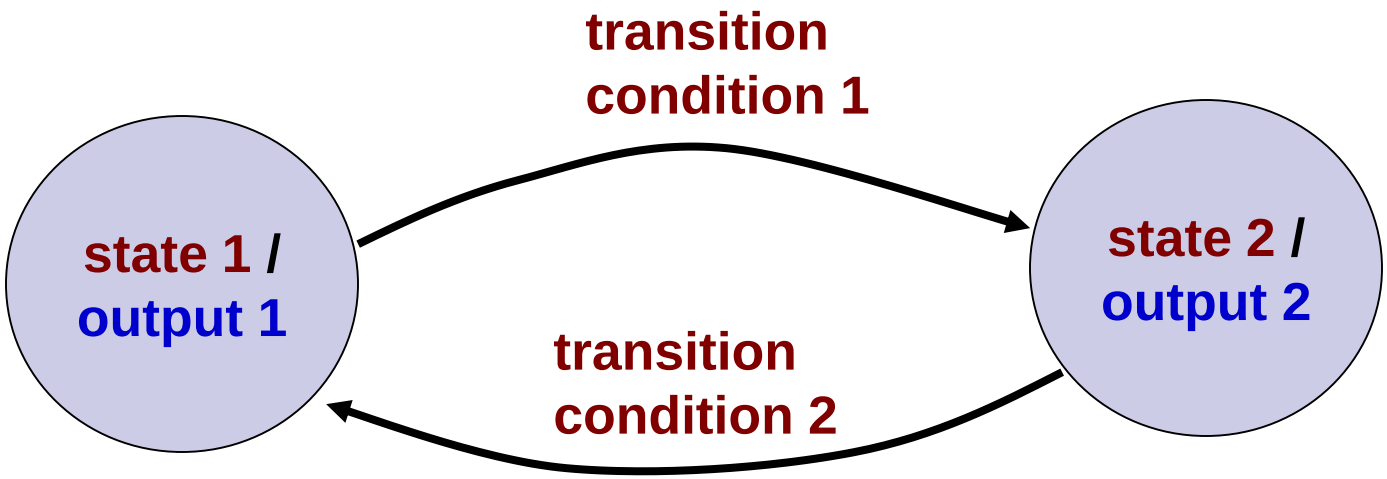


Mealy FSM

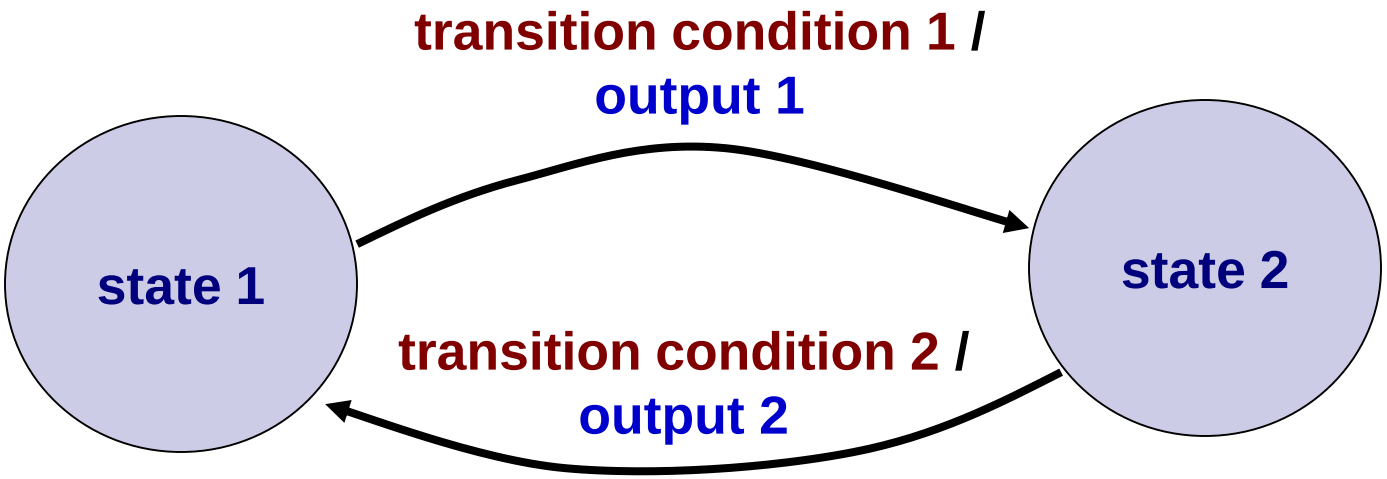
- Output Is a Function of a Present State and Inputs



Moore Machine



Mealy Machine



Moore vs. Mealy FSM

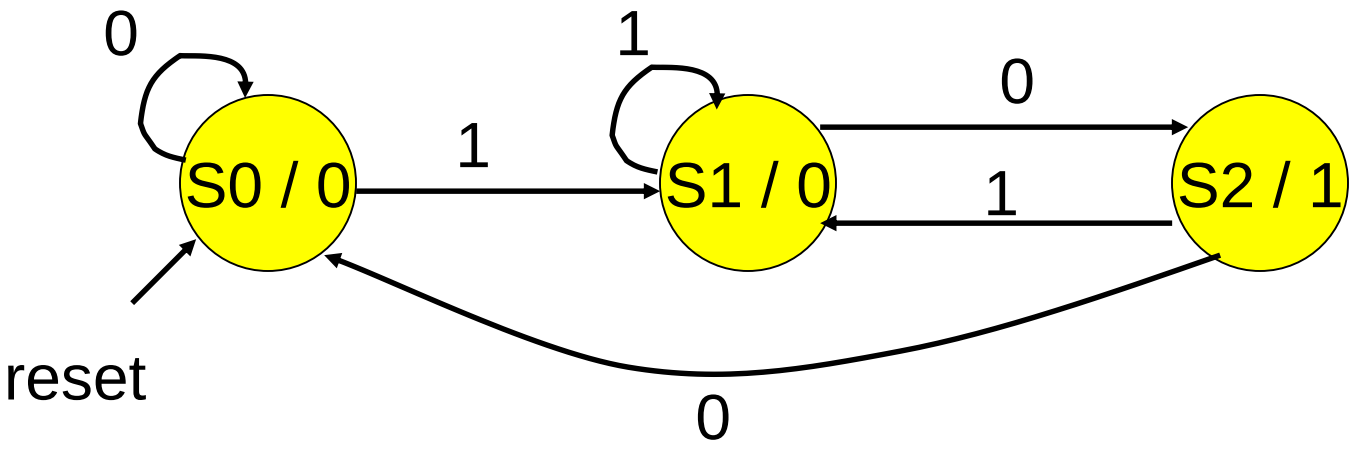
- Moore and Mealy FSMs can be functionally equivalent
  - Equivalent Mealy FSM can be derived from Moore FSM and vice versa
- Mealy FSM Has Richer Description and Usually Requires Smaller Number of States
  - Smaller circuit area

Moore vs. Mealy FSM…

- Mealy FSM computes outputs as soon as inputs change
  - Mealy FSM responds one clock cycle sooner than equivalent Moore FSM
- Moore FSM Has no combinational path between inputs and outputs
  - Moore FSM is more likely to have a shorter critical path
  - Moore outputs synchronized with clock; Mealy outputs may not be (may have race conditions, timing issues, etc.)

Example: Moore FSM

- Moore FSM that Recognizes Sequence “10”



Meaning of states:

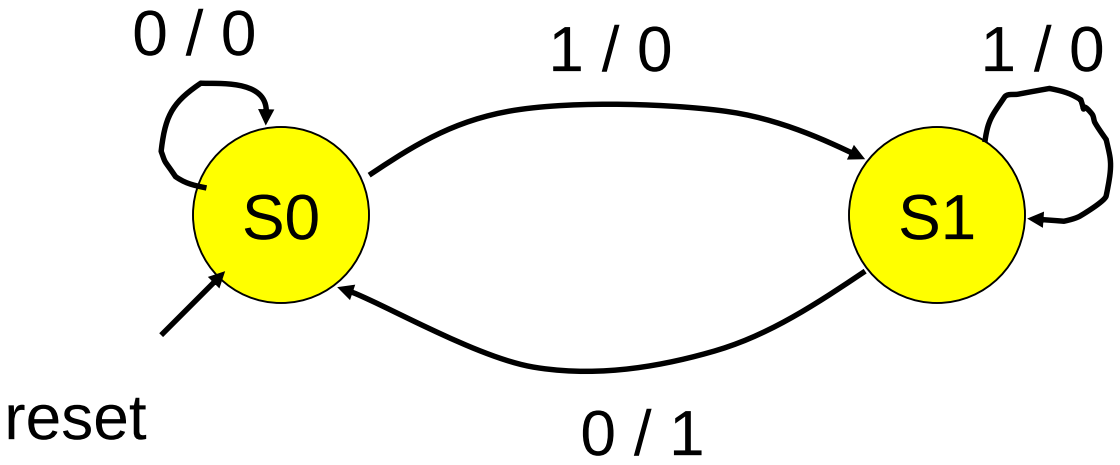
S0: No elements of the sequence observed

S1: “1” observed

S2: “10” observed

Example: Mealy FSM

- Mealy FSM that Recognizes Sequence “10”

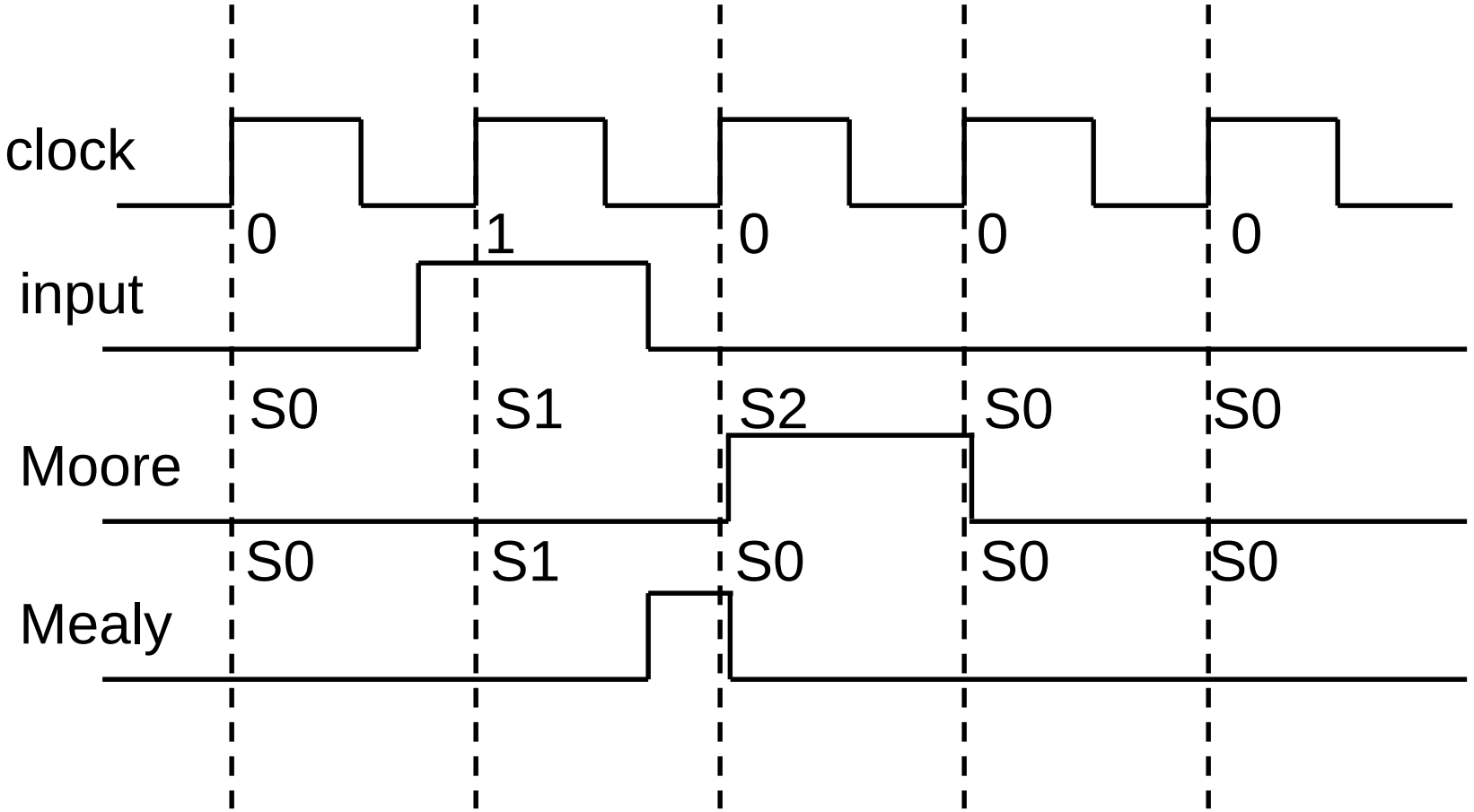


Meaning of states:

S0: No elements of the sequence observed

S1: “1” observed

Example:Moore & Mealy FSMs



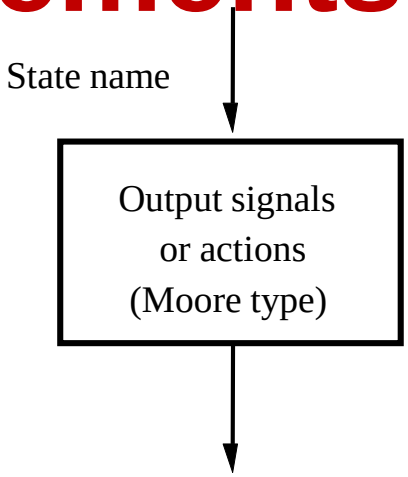
FSM Limitations

- Simple finite state machines (those expressed using state diagrams and state tables) good only for simple designs
  - Many inputs and many outputs make it awkward to draw state machines
  - Often only one input affects the next change of state
  - Most outputs remain the same from state to state
- Instead use algorithmic state machines (ASM)

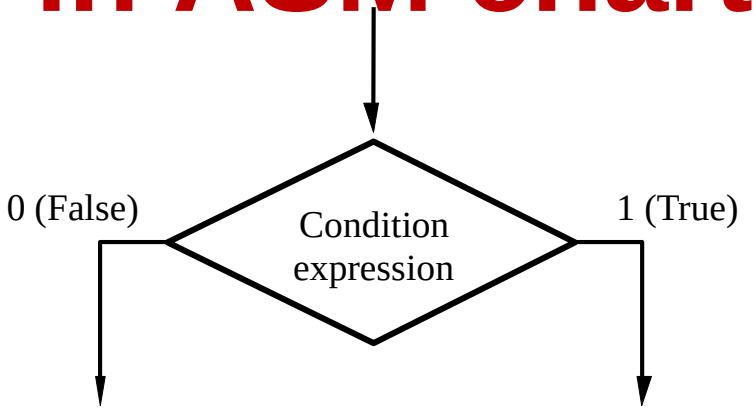
Algorithmic State Machine (ASM)

- Complex digital systems can be represented by algorithmic state machines
- Simple finite state machines (expressed using state diagrams and state tables) good only for simple designs
- Algorithmic State Machines (ASM) are
  - flow-chart type diagrams to represent finite state machines
  - suitable for a larger number of inputs and outputs compared to simple FSMs

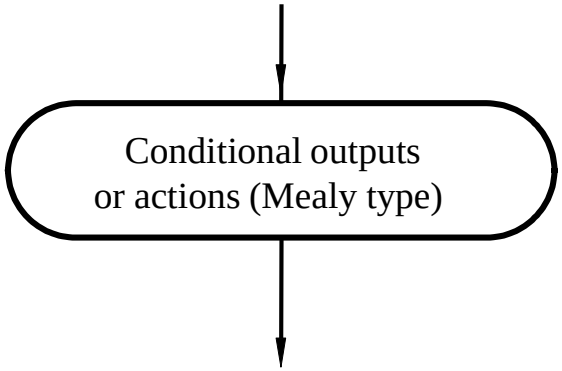
Elements used in ASM charts



(a) State box



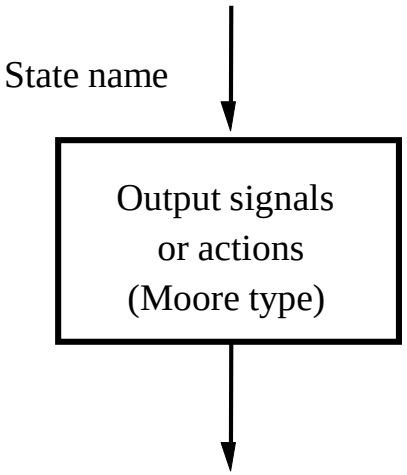
(b) Decision box



(c) Conditional output box

State Box

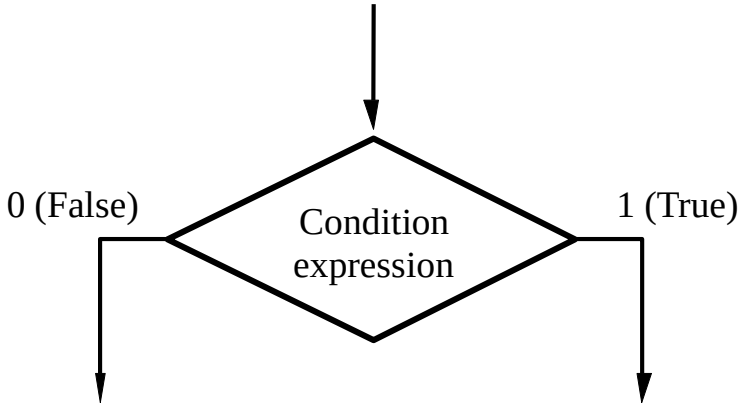
- **State box** – represents a state.
- Equivalent to a node in a state diagram or a row in a state table.
- Contains register transfer actions or output signals
- **Moore-type outputs are listed inside of the box.**
- It is customary to write only the name of the signal that has to be asserted in the given state, e.g., z instead of z=1.
- Also, it might be useful to write an action to be taken, e.g., count = count + 1, and only later translate it to asserting a control signal that causes a given action to take place.



Decision Box

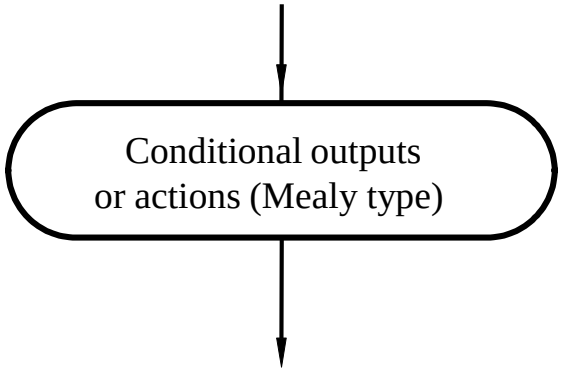
- **Decision box** – indicates that a given condition is to be tested and the exit path is to be chosen accordingly

  The condition expression consists of one or more inputs to the FSM.



Conditional Output Box

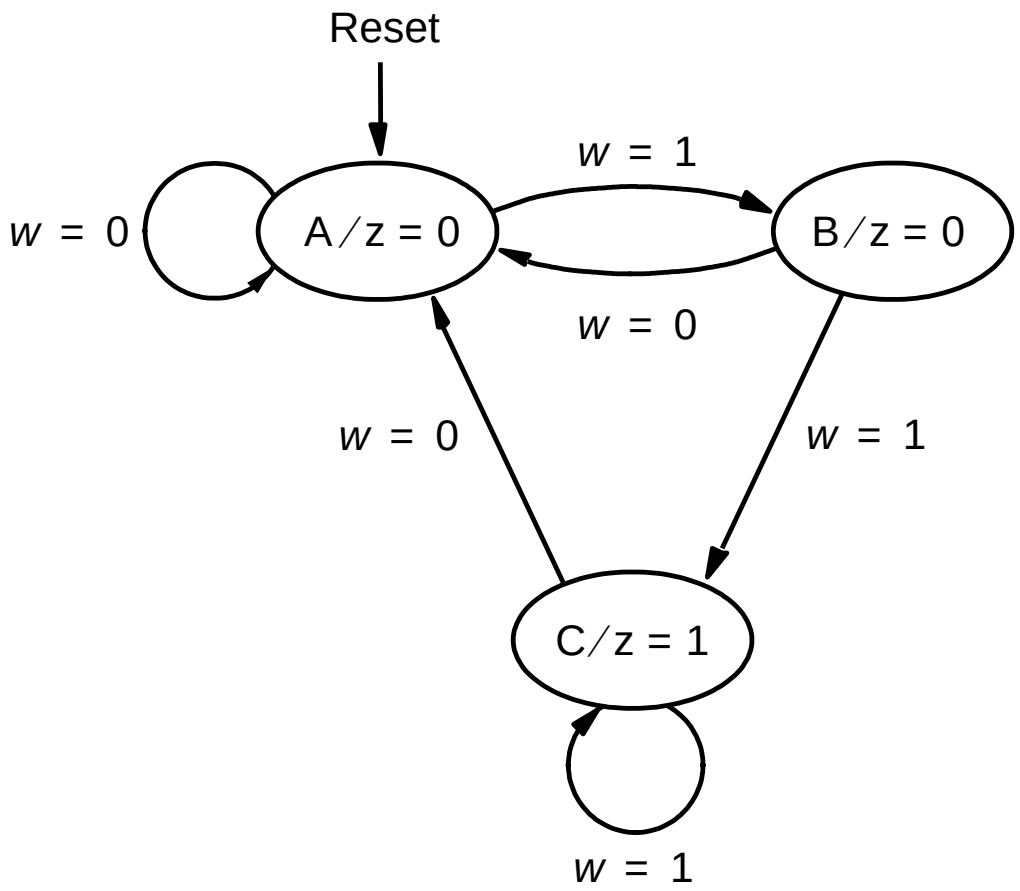
- **Conditional output box**
- Denotes output signals that are of the Mealy type.
- The condition that determines whether such outputs are generated is specified in the decision box.



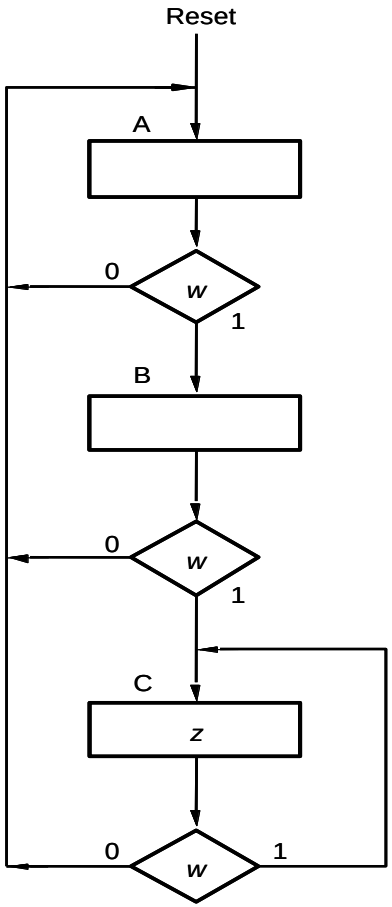
ASMs representing simple FSMs

- Algorithmic state machines can model both Mealy and Moore simple finite state machines

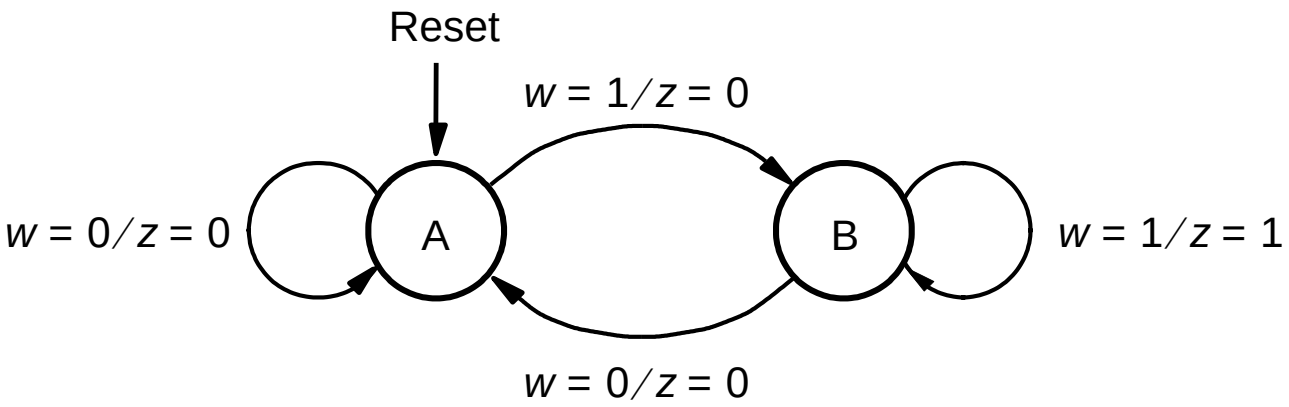
Example 1: Moore FSM –State diagram



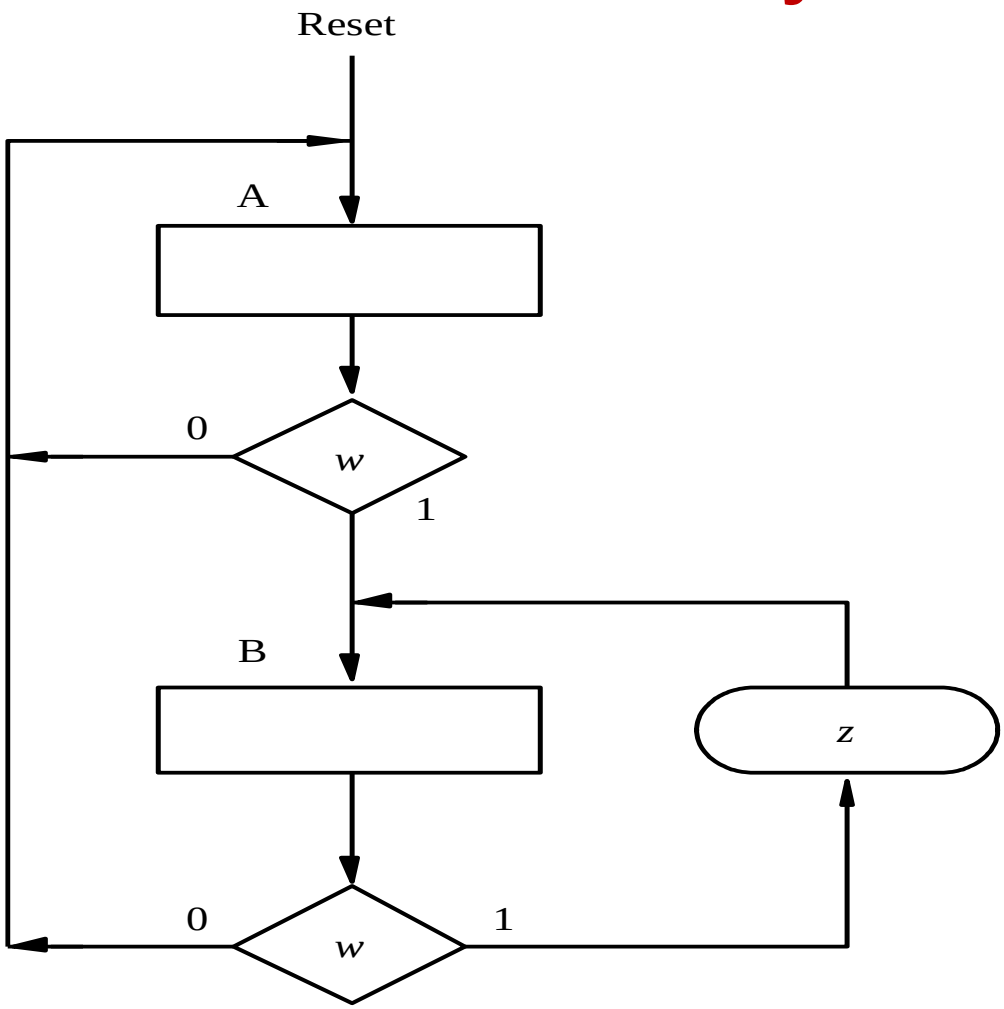
Example 1: ASM Chart for Moore FSM



Example 2: Mealy FSM –State diagram



Example 2: ASM Chart for Mealy FSM



Example 3 – ASM Complex Counter Problem

A sync. 3 bit counter has a mode control M. When M = 0, the counter counts up in the binary sequence. When M = 1, the counter advances through the Gray code sequence.

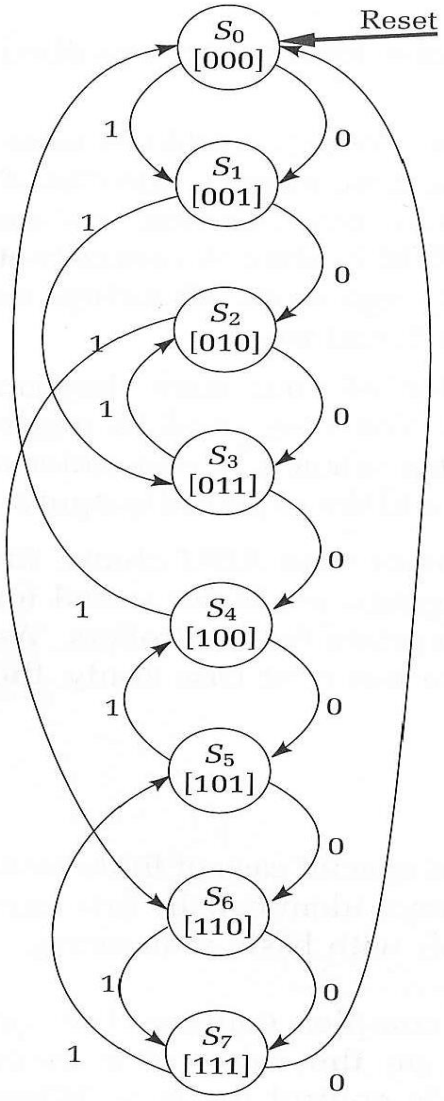
Binary: 000, 001, 010, 011, 100, 101, 110, 111
Gray: 000, 001, 011, 010, 110, 111, 101, 100

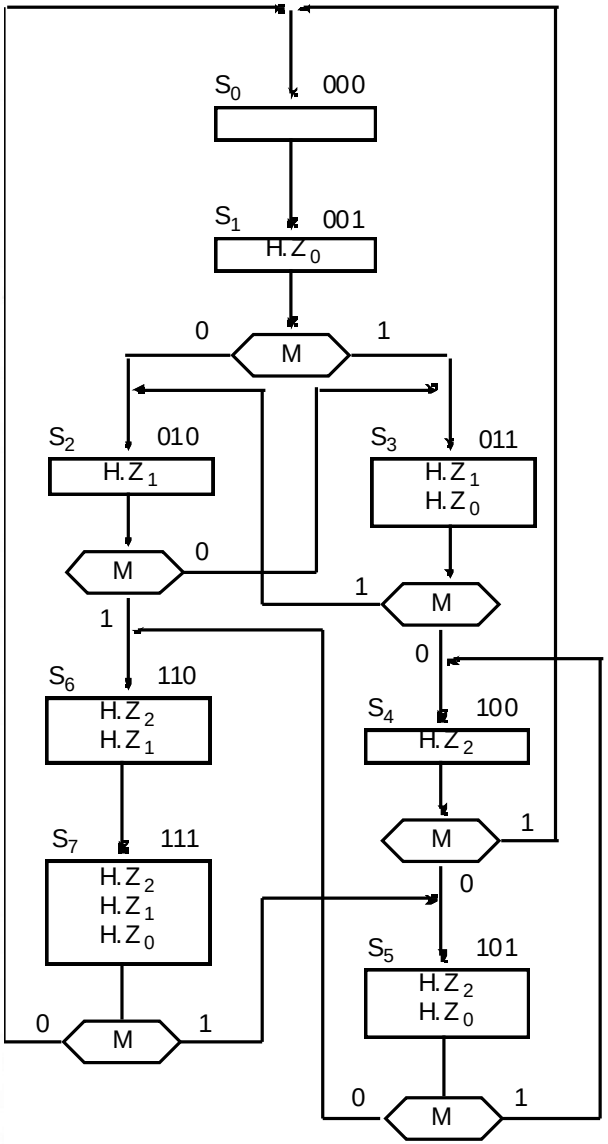
Valid I/O behavior:

Mode Input M	Current State	Next State (Z2 Z1 Z0)
0	000	001
0	001	010
1	010	110
1	110	111
1	111	101
0	101	110
0	110	111

Example- Finite State Machine

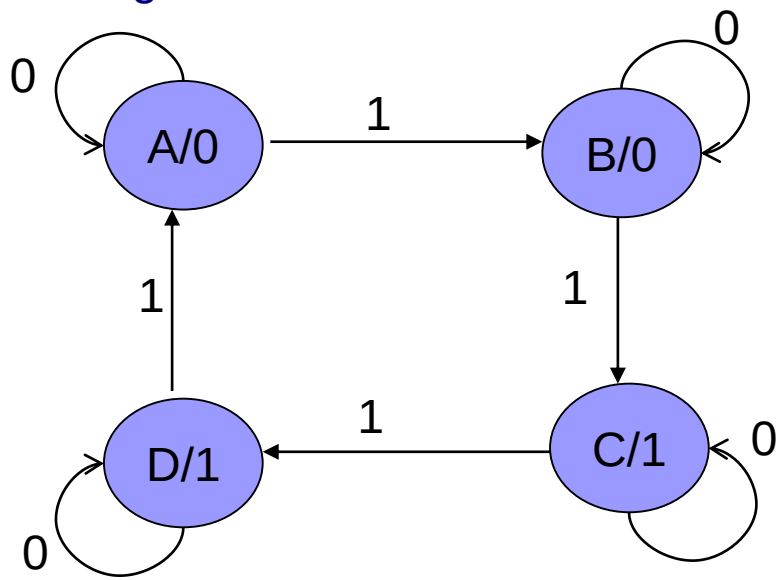
One state for each output combination
Add appropriate arcs for the mode control





Example – State Diagram

State Diagram



State Table

Present State	Next State		Output (z)
	Input (x)		
	0	1	
A	A	B	0
B	B	C	0
C	C	D	1
D	D	A	1

Example – State Table

State assignments:

00 → A

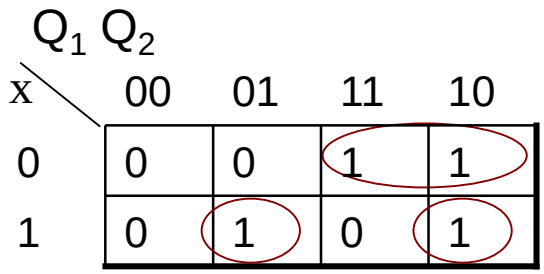
01 → B

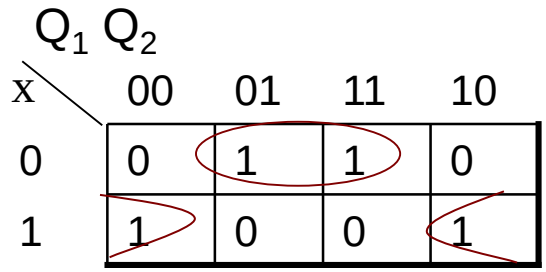
10 → C

11 → D

Input	Present State		Next State		Output (z)
x	Q_1	Q_2	Q^+_1	Q^+_2	
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	1
0	1	1	1	1	1
1	0	0	0	1	0
1	0	1	1	0	0
1	1	0	1	1	1
1	1	1	0	0	1

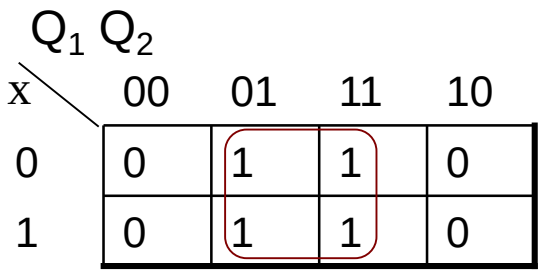
Example – Karnaugh Map





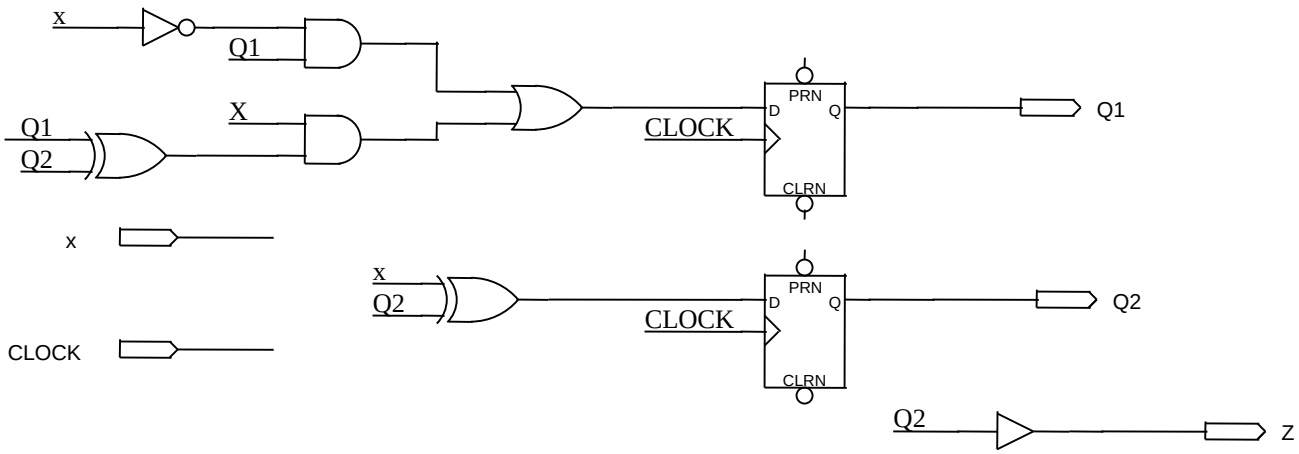
$$Q_1 = \bar{x}Q_1 + x(Q_1 \oplus Q_2)$$

$$Q_2 = \bar{x}Q_2 + x\bar{Q}_2 = x \oplus Q_2$$



$$z = Q_2$$

Circuit Implementation



Wave Form