14:332:231
DIGITAL LOGIC DESIGN

Ivan Marsic, Rutgers University
Electrical & Computer Engineering
Fall 2013

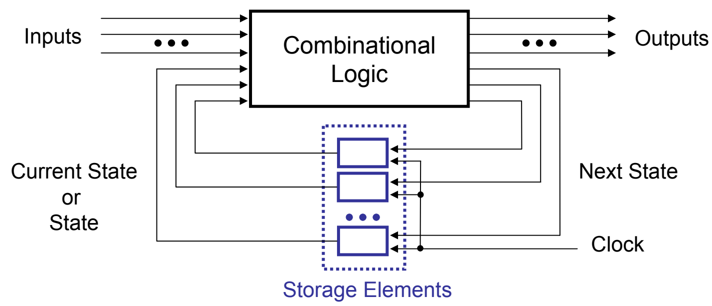
Lecture #17: Clocked Synchronous State-Machine Analysis

Clocked Synchronous Sequential Circuits

- Also known as "finite state machines"
  - Finite refers to the fact that the number of states the circuit can assume is finite
- Use edge-triggered *flip-flops*
- "Clocked" = all storage elements use a clock input (i.e. all storage elements are flip-flops)
- "Synchronous" = all flip-flops use the same clock signal
  - All flip-flops are triggered from the same master clock signal, and therefore all change their state together

Clocked Synchronous FSM Structure

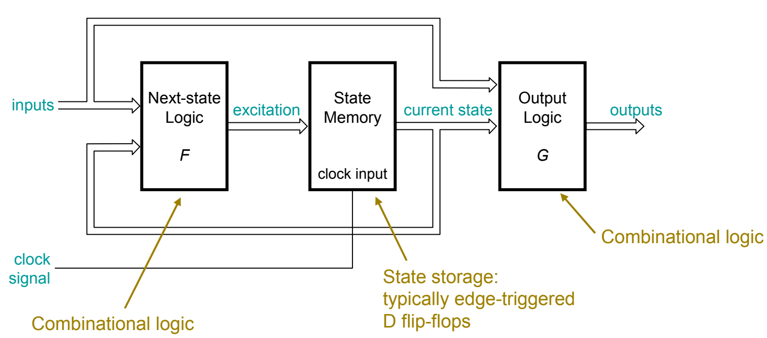
- **State**: determined by possible values in sequential storage elements
- **Transition**: change of state
- **Clock**: controls when state can change by controlling storage elements



State-machine Structure (Mealy)

- **Mealy machine**
  output depends on state and current input:
  Next state = *F* (current state, input)
  Output = *G* (current state, input)

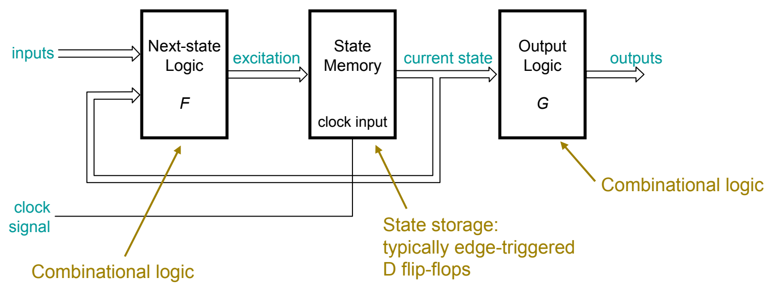
State storage = set of *n* flip-flops that store the state of the machine (2^n states)



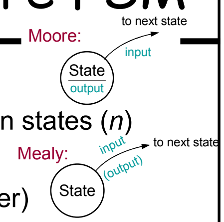
State-machine Structure (Moore)

- **Moore machine**
  output depends only on current state:

  Output = *G* (current state)



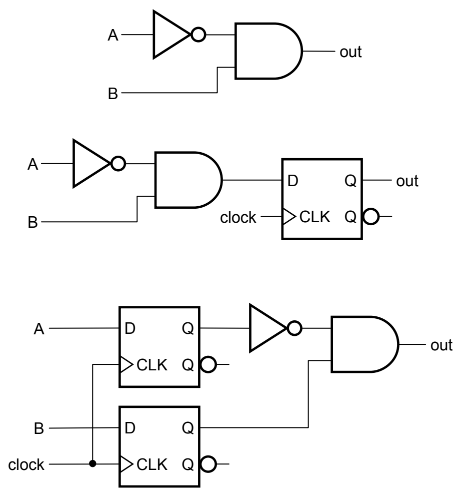
Comparison of Mealy & Moore FSM



- Mealy machines usually have less states
  - outputs are shown on transitions ($n \times n$) rather than in 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 *may* occur if one isn't careful
- Mealy machines react faster to inputs
  - react in the same cycle—don't need to wait for clock
  - outputs *may* be considerably shorter than the clock cycle
  - but, asynchronous outputs and asynchronous are hazardous
  - in Moore machines, more logic *may* be necessary to decode state into outputs—there *may* be more gate delays after clock edge

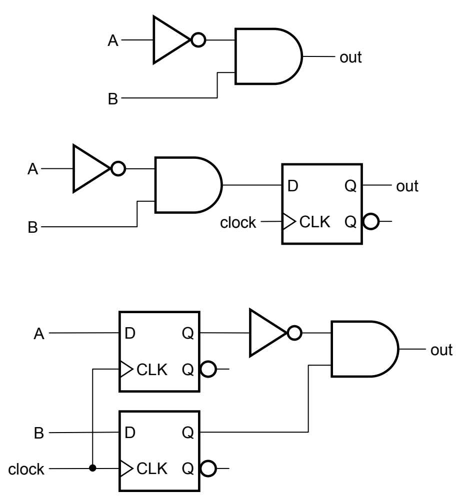
Mealy and Moore Example

- Mealy or Moore?



Mealy and Moore Example

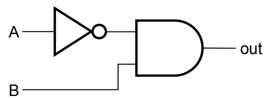
- Mealy or Moore?



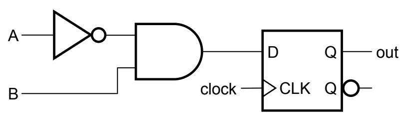
Not a state machine

Mealy and Moore Example

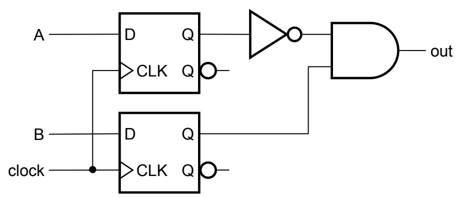
- Mealy or Moore?



Not a state machine

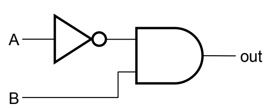


Moore:
output = Γ(state)
[no directly feeding input to output logic]

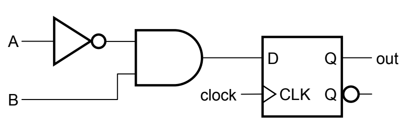


Mealy and Moore Example

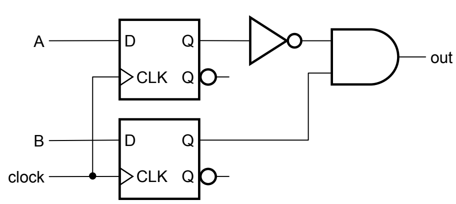
- Mealy or Moore?



Not a state machine



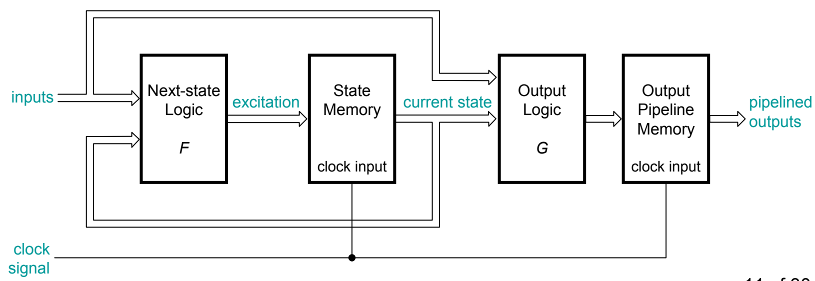
Moore:
output = Γ(state)
[no directly feeding input to output logic]



Moore:
output = Λ(state)
[no directly feeding input to output logic]

Mealy Machine with Pipelined Outputs

- Outputs of a Mealy machine can be kept constant within a clock period by using output flip-flops
- Often used in programmable logic device (PLD) based state machines
  - Output taken directly from flip-flops, valid sooner after clock edge
  - But the "output logic" must determine output value *one clock tick sooner* ("pipelined")
  - Drawback: output changes are delayed by as much as one clock cycle



Notation, Characteristic Equations

- Q* means "the next value of Q" ("next state")
- "Excitation" is the *input* applied to a device that determines the next state
- "Characteristic equation" specifies the next state of a device as a function of its excitation (inputs)

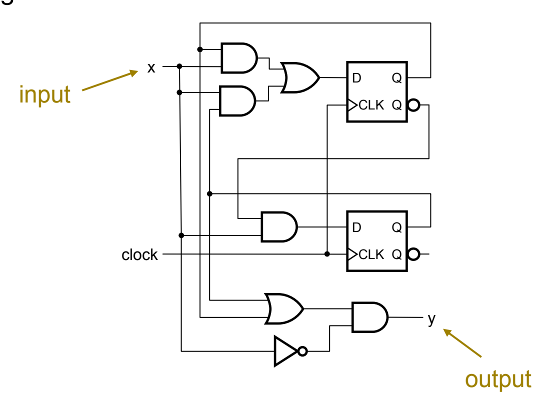
Device Type	Characteristic Equation
S-R latch	$Q* = S + R' \cdot Q$
D latch	$Q* = D$
Edge-triggered D flip-flop	$Q* = D$
...	...
Edge-triggered J-K flip-flop	$Q* = J \cdot Q' + K' \cdot Q$
T flip-flop	$Q* = Q'$
T flip-flop with enable	$Q* = EN \oplus Q = EN \cdot Q' + EN' \cdot Q$

Clocked Synchronous State Machine Analysis

- Clocked synchronous state machines can be described in many ways:
  - circuit schematic
  - state and state/output tables
  - transition and transition/output tables
  - state diagrams (flowcharts)
  - ASM (algorithmic state machine) charts
  - HDL (hardware description languages)
  - programming languages
- A description that can be given to a CAD system for simulation and synthesis is preferred. Usually these are text descriptions, but drawing tools exist

Example Sequential Circuit Analysis

- Is this a Moore or Mealy machine?
- What does it do?
- How do the outputs change when an input arrives?

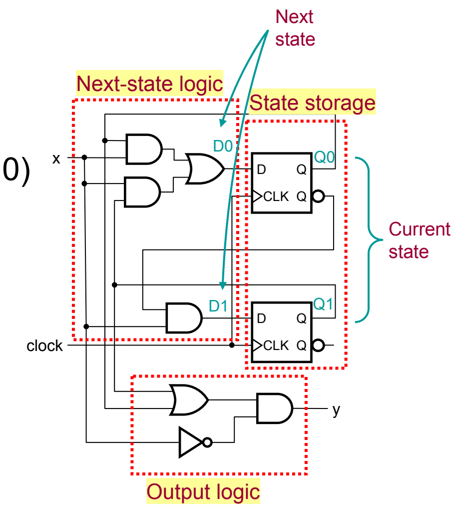


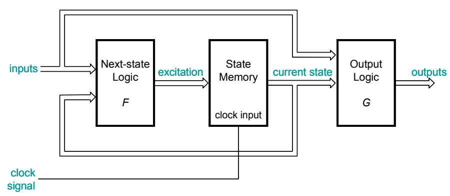
Example Sequential Circuit Analysis

- Input: x(t)
- Output: y(t)
- State: (Q0(t), Q1(t))

  Example: (Q0 Q1)= (01), (10)
- Next State:

  (D0(t), D1(t)) =
  (Q0(t+1), Q1(t+1))





State-Machine Analysis Steps

- Assumption: Starting point is a logic diagram

1. Determine next-state function $F(\cdot)$ and output function $G(\cdot)$

2a. Construct state table
- – For each state/input combination, determine the excitation value
- – Using the characteristic equation, determine the corresponding next-state values (*trivial* with D flip-flops)

2b. Construct output table
- – For each state/input combination, determine the output value (can be combined with state table)

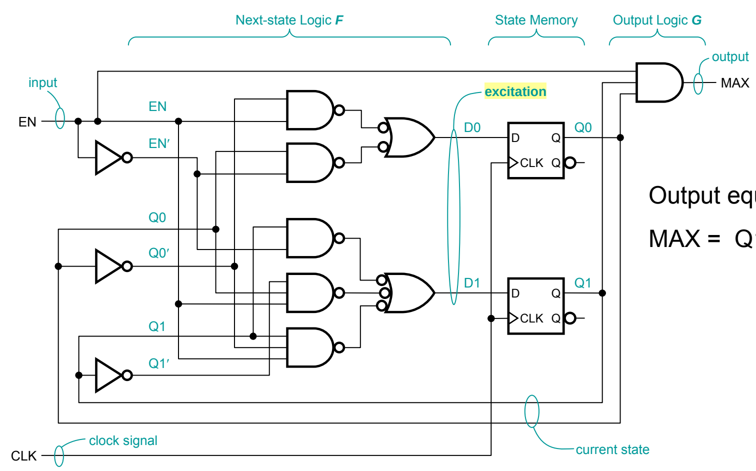
3. Draw the state diagram (optional)

Some Definitions

- *Excitation* = input signals for D flip-flops at each clock tick
- *Excitation equation* = next-state logic *F*(·) of the state machine
- *Characteristic equation* = specifies the flip-flop's next state as a function(current-state, inputs)
- *Transition equation* = specifies the state machine's next state as a function(current-state, inputs); essentially same as *F*(·)
- *Transition table* = created by evaluating the transition equations for very input/state combination
- *Output equation* = output behavior *G*(·) of the state machine

Example State Machine

- Clocked synchronous state machine example
  - Using positive-edge triggered D flip-flops



Output equation:

MAX = Q1 · Q0 · EN

… it is a Mealy Machine

- The flip-flops are positive-edge-triggered D flip-flops
- *State-to-state transitions* occur when the state memory (flip-flops) is loaded with new next-state values
  – state-to-state transitions can only occur on the CLK edge

The flowchart for the analysis:

excitation equation → characteristic equation → transition equation → transition table → output equation → state/output table → state diagram

Transition Equations

- Excitation equations:

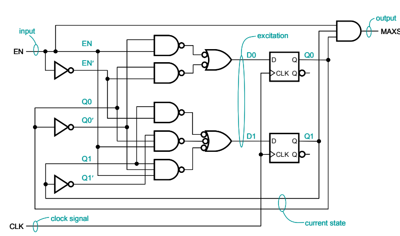
$$D0 = Q0 \cdot EN' + Q0' \cdot EN$$

$$D1 = Q1 \cdot EN' + Q1' \cdot Q0 \cdot EN + Q1 \cdot Q0' \cdot EN$$

- Characteristic equations:

$$Q0^* = D0$$

$$Q1^* = D1$$



- Substitute excitation equations into characteristic equations to obtain ***transition equations***:

$$Q0^* = Q0 \cdot EN' + Q0' \cdot EN$$

$$Q1^* = Q1 \cdot EN' + Q1' \cdot Q0 \cdot EN + Q1 \cdot Q0' \cdot EN$$

Transition and State Tables

- Transition equations:
  $Q0^* = Q0 \cdot EN' + Q0' \cdot EN$
  $Q1^* = Q1 \cdot EN' + Q1' \cdot Q0 \cdot EN + Q1 \cdot Q0' \cdot EN$
- Output equation:
  $MAX = Q1 \cdot Q0 \cdot EN$

	EN	
Q1 Q0	***0***	***1***
00	00	01
01	01	10
10	10	11
11	11	00
	Q1* Q0*	

transition table

	EN	
S	***0***	***1***
A	A	B
B	B	C
C	C	D
D	D	A
	S*	

state table
A = 00
B = 01
C = 10
D = 11

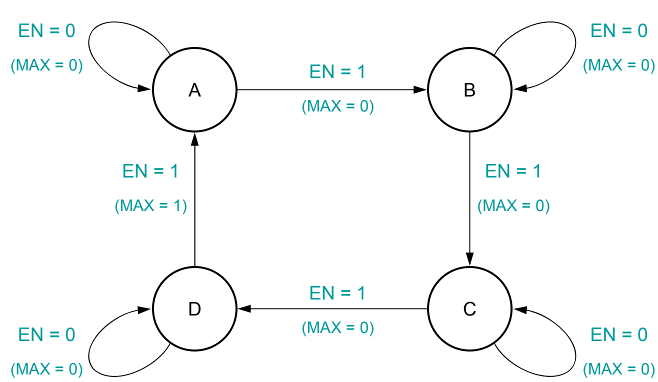
	EN	
S	***0***	***1***
A	A, 0	B, 0
B	B, 0	C, 0
C	C, 0	D, 0
D	D, 0	A, 1
	S*, MAX	

state/output table

<u>Function of the example machine</u>
2-bit binary *counter* with enable input EN:
- When EN=0, maintains current count
- When EN=1, the count advances by 1 at each clock tick; rolling over to 00 after 11

State Diagram

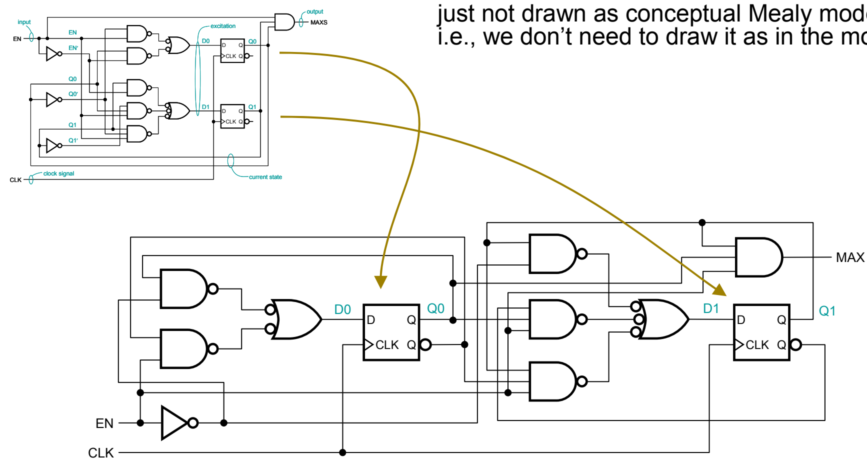
- Graphical representation of the state/output table
- Ovals for states
- Arrows for transitions (annotated by the output)



Redrawing of the Example Synchronous State Machine

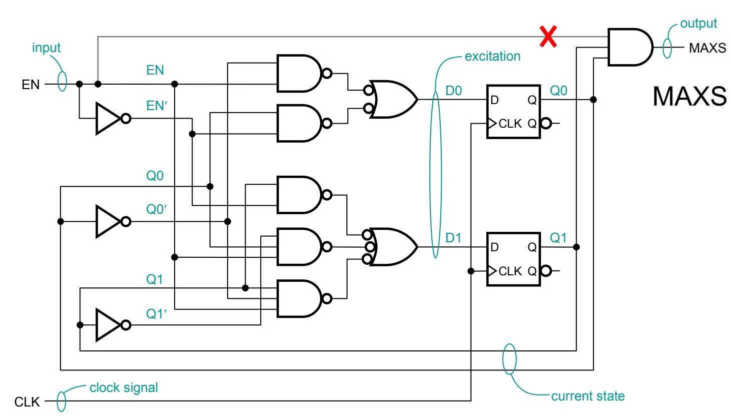
- Excitation equations and the state variables are placed slightly differently (also QN is used)

… but it is the same state machine
just not drawn as conceptual Mealy model
i.e., we don't need to draw it as in the model



Modified State Machine

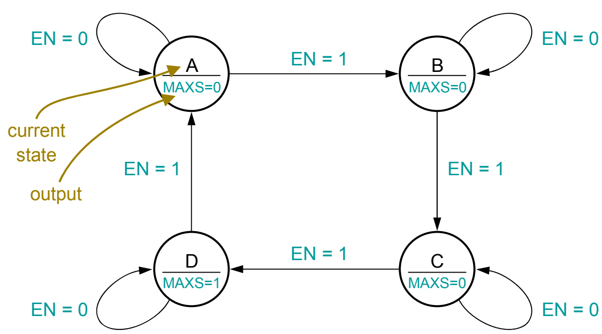
- *Moore* machine, the output depends only on the state



$$MAXS = Q0 \cdot Q1$$

Modified State Machine

- Moore state diagram and state/output table
- Moore type output depends *only on state*
  - Mealy type output depends on *state* ***and*** *input*

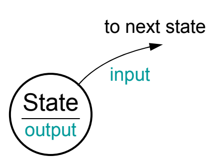


	EN		
S	***0***	***1***	***MAXS***
A	A	B	0
B	B	C	0
C	C	D	0
D	D	A	1
	S*		

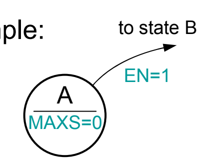
State Diagram Convention

Moore Machine:

- output depends only on state

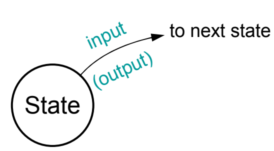


Example:

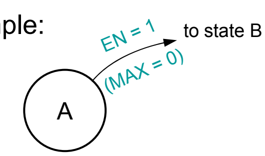


Mealy Machine:

- output depends on state and input

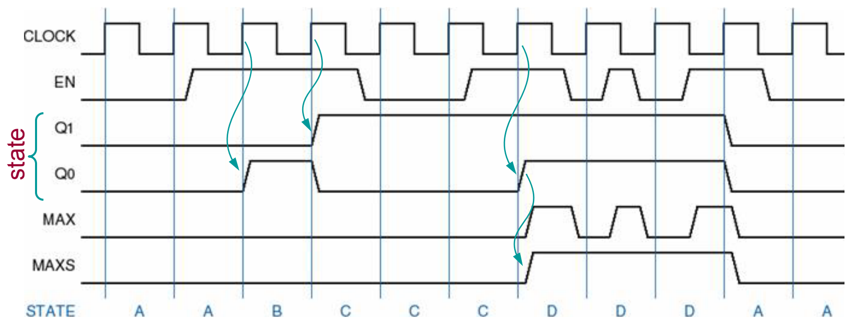


Example:



Timing Diagram for State Machine(s)

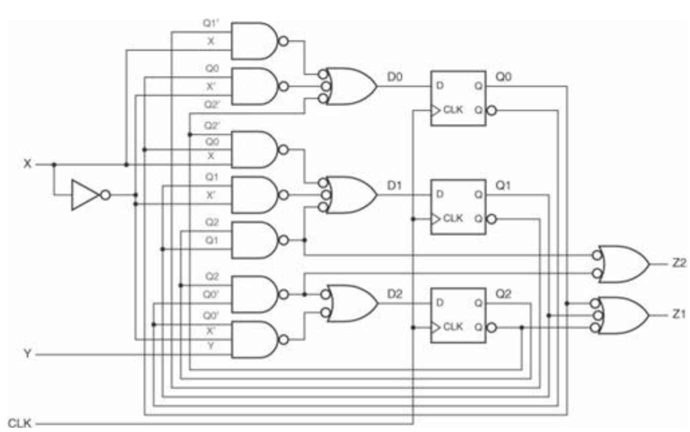
- Timing diagram shows example behavior, starting with a given initial state of 00 (A)
- NOT a complete description of machine behavior because it neglects timing constraints



- States: A = 00 | B = 01 | C = 10 | D = 11
- The counter counts only if EN=1 at the rising edge of CLOCK

Another Example State Machine

- A clocked synchronous state machine with three flip-flops and eight states



Example State Machine Analysis

- Excitation equations:
  - $D0 = Q1' \cdot X + Q0 \cdot X' + Q2$
  - $D1 = Q2' \cdot Q0 \cdot X + Q1 \cdot X' + Q2 \cdot Q1$
- Transition equations:
  - $Q0^{*} = Q1' \cdot X + Q0 \cdot X' + Q2$
  - $Q1^{*} = Q2' \cdot Q0 \cdot X + Q1 \cdot X' + Q2 \cdot Q1$
  - $Q2^{*} = Q2 \cdot Q0' + Q0' \cdot X' \cdot Y$
- Output equations:
  - $Z1 = Q2 + Q1' + Q0'$
  - $Z2 = Q2 \cdot Q1 + Q2 \cdot Q0'$

	X Y				
Q2 Q1 Q0	00	01	10	11	Z1 Z2
000	000	100	001	001	10
001	001	001	011	011	10
010	010	110	000	000	10
011	011	011	010	010	00
100	101	101	101	101	11
101	001	001	001	001	10
110	111	111	111	111	11
111	011	011	011	011	11
	Q2* Q1* Q0*				

transition/output table

	X Y				
S	00	01	10	11	Z1 Z2
A	A	E	B	B	10
B	B	B	D	D	10
C	C	G	A	A	10
D	D	D	C	C	00
E	F	F	F	F	11
F	B	B	B	B	10
G	H	H	H	H	11
H	D	D	D	D	11
	S*				

state/output table

Moore machine

State Diagram of the Example State Machine

- **Transition expression** = a transition is taken for inputs for which the transition expression is “1”
- Transition expressions on arcs *leaving* a state must be mutually exclusive and all-inclusive
- Transitions labeled “1” are always taken
- The sum of the leaving transition expressions must be one
- For a given (current-state, next-state) a transition expression can be written as a *sum of minterms* for the input combinations that cause that transition