



University of
Massachusetts
Amherst

Engin112 – Lectures 27-28

Sequential Circuits

Analysis and Design

Maciej Ciesielski
Department of Electrical and Computer Engineering
11/07-09/2011

Analysis of Sequential Circuits

- Behavior of clocked sequential circuit determined by
 - Inputs
 - Outputs
 - State of flip-flops
- Analysis process
 - Consider all combinations of
 - » Inputs
 - » Flip-flop states
 - Determine next state and output of circuit
- Concept of a Finite State Machine (FSM)
- Methods
 - State equations
 - State table
 - State diagram

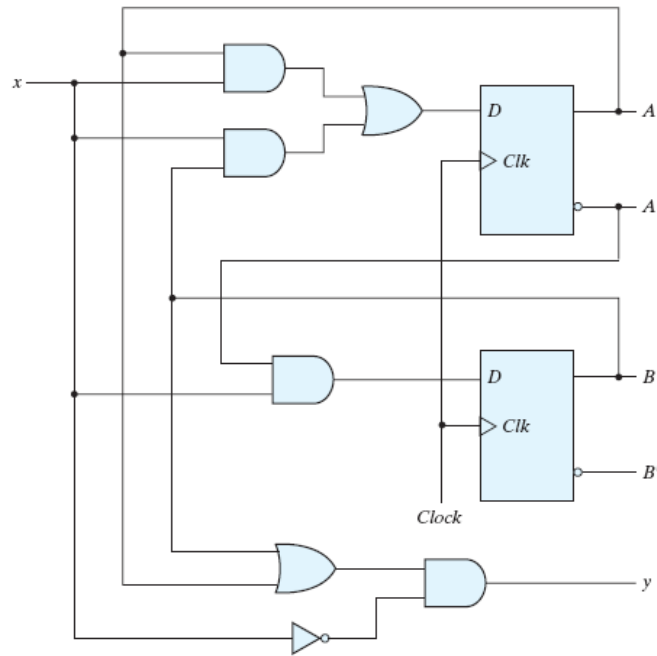
Example Circuit

- Sequential circuit

- Input: x
- Output: y
- Flip-flops:
 - » 2 D-type A and B

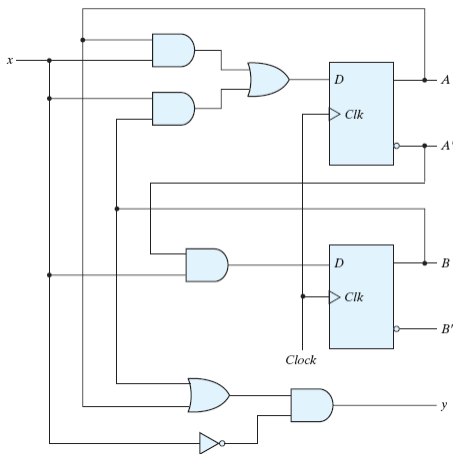
- When is $y=1$?

- Very difficult to answer
- Systematic analysis necessary

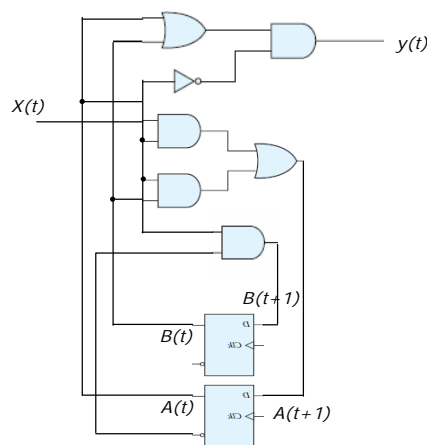


State and Output Equations

Circuit view



FSM view



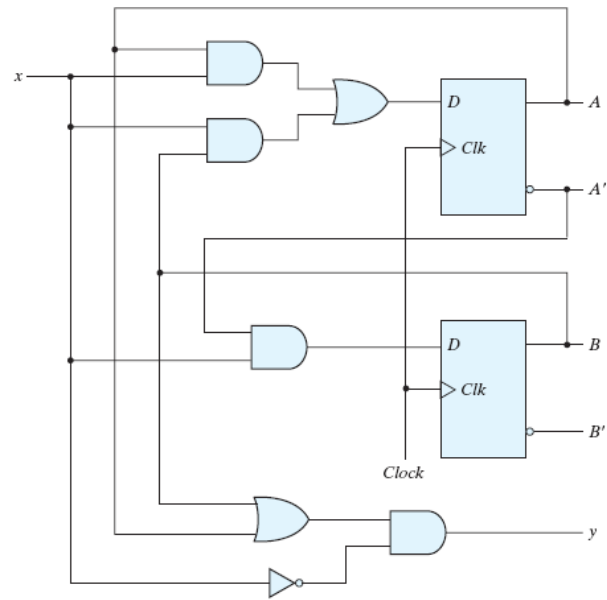
$$A(t+1) = A(t) x(t) + B(t) x(t)$$

$$B(t+1) = A'(t) x(t)$$

$$y(t) = (A(t) + B(t)) x'(t)$$

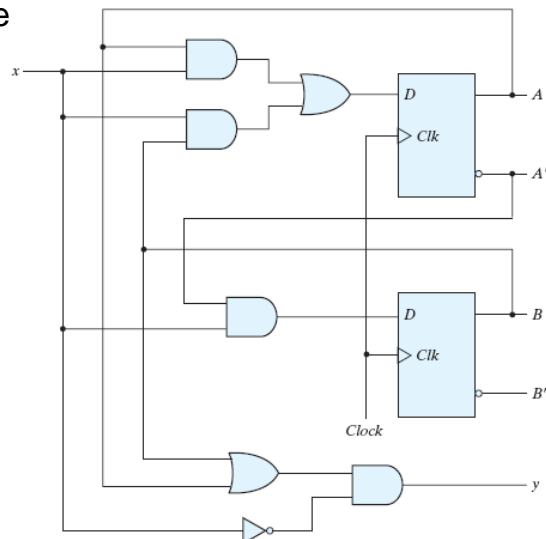
State and Output Equations

- State equation specifies next state
 - Function of current state and inputs
- State equation for flip-flops:
 - $A(t+1) = A(t) x(t) + B(t) x(t)$
 - $B(t+1) = A'(t) x(t)$
- Output expression:
 - $y(t) = (A(t) + B(t)) x'(t)$
- Simplified:
 - $A(t+1) = A x + B x$
 - $B(t+1) = A' x$
 - $y = (A+B) x'$
- How to derive these equations ?



Flip-flop Input Equations

- Similar to state equations
 - Specifies type of flip-flop used
 - In case of D-FFs they are the same as state equations.
- Example:
 - $D_A = A x + B x$
 - $D_B = A' x$
 - $y = (A + B) x'$



State Table

- What needs to be considered in table?
 - Inputs
 - State of flip-flops
 - Next state of flip-flops
 - Outputs

} "left side" of table
 } "right side" of table

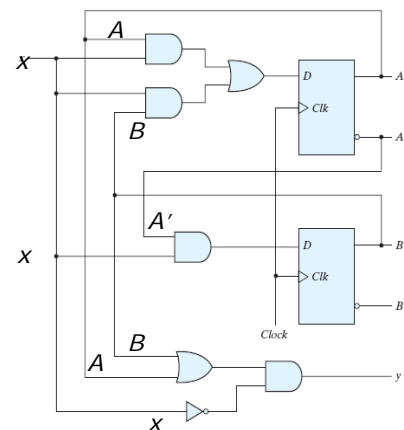
- How many entries in state table?
 - n inputs
 - m states
 - Total of 2^{m+n} entries

- For every entry
 - Determine flip-flop change by input and current state
 - » State equation
 - Determine the output
 - » Output equation

State Table

- State table

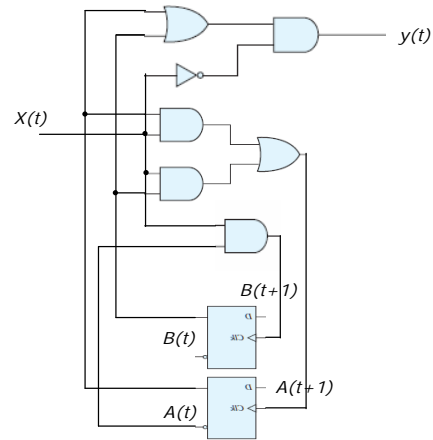
current state		input	next state		output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0



State Table

- Alternate form:

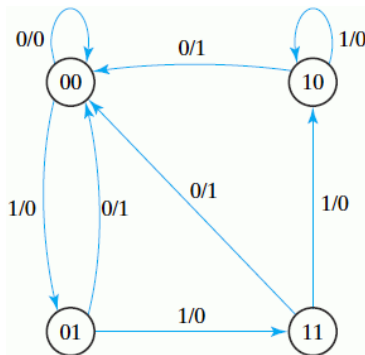
current state	next State		Output	
	x=0	x=1	x=0	x=1
AB	AB	AB	y	y
00	00	01	0	0
01	00	11	1	0
10	00	10	1	0
11	00	1	0	



- Note that state combinations can be concatenated
 - $AB = 00$ instead of $A = 0$ and $B = 0$

State Diagram

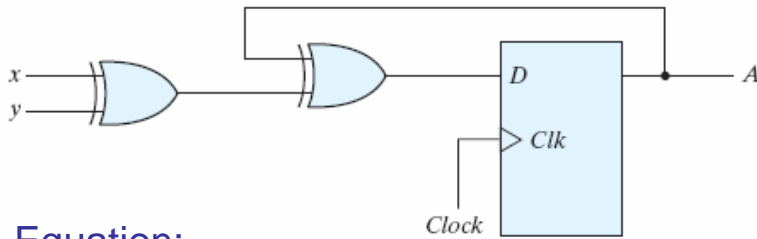
- State transitions represented as graph
 - Vertices indicate states
 - Edges represent transitions
 - » Edge annotation: "x/y" meaning input is x and output is y
- Easiest generated from state table



current state	next state		output	
	x=0	x=1	x=0	x=1
AB	AB	AB	y	y
00	00	01	0	0
01	00	11	1	0
10	00	10	1	0
11	00	10	1	0

Analysis Example 2

▪ **Circuit:**



Present state	Inputs		Next state
	A	x y	
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

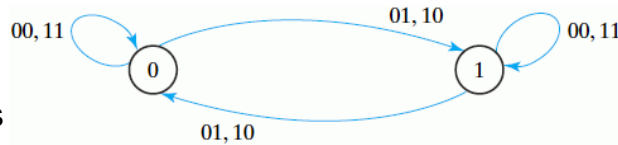
▪ **Equation:**

- $A(t+1) = A \oplus x \oplus y$ (state equation)
- $D_A(t+1) = A \oplus x \oplus y$ (flip-flop equation)

▪ **State table:**

▪ **State diagram:**

- Note: no outputs



Recap from last lecture

▪ **Sequential circuit analysis**

- State equations
- State table
- State diagram

▪ **Today's lecture**

- Finite state machines (FSM)
 - » Mealy
 - » Moore
- Sequential circuit design procedure

Finite State Machines

- State diagrams are representations of *Finite State Machines* (FSM)

- Two flavors of FSMs:
 - » Mealy FSM
 - » Moore FSM

- Difference:**

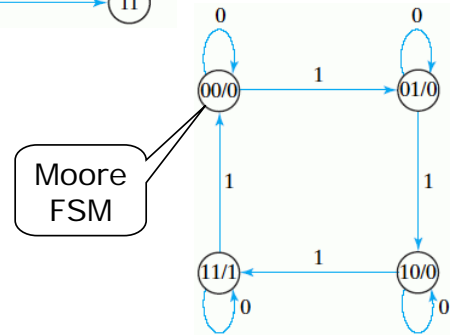
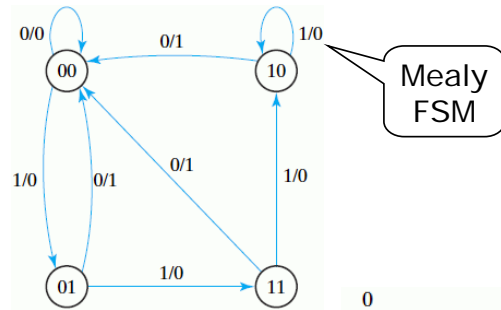
- How output is determined

- Mealy FSM**

- Output depends on input and state
- Output is not synchronized with clock
 - » can have temporarily unstable output

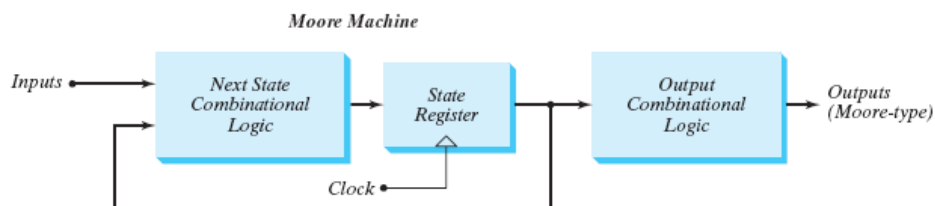
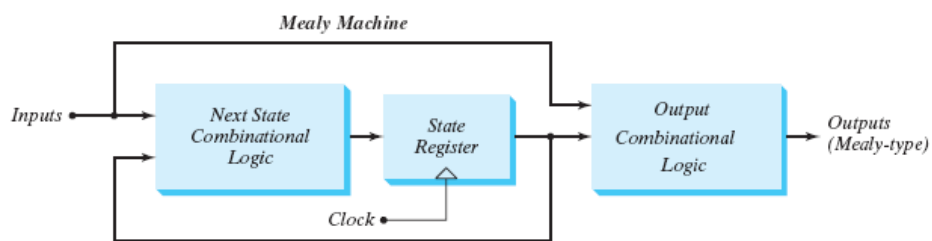
- Moore FSM**

- Output depends only on state



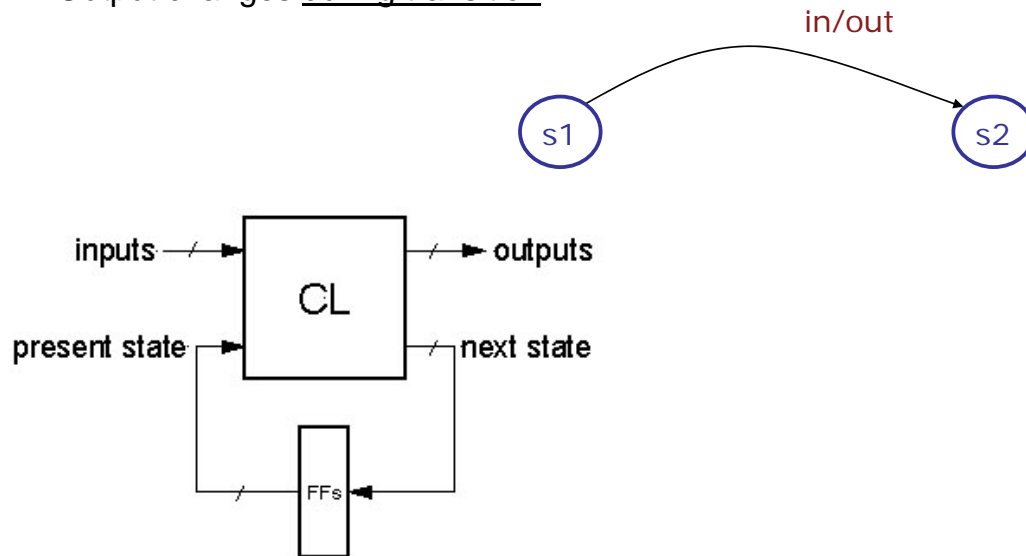
Two Flavors of FSM

- Mealy vs Moore machine



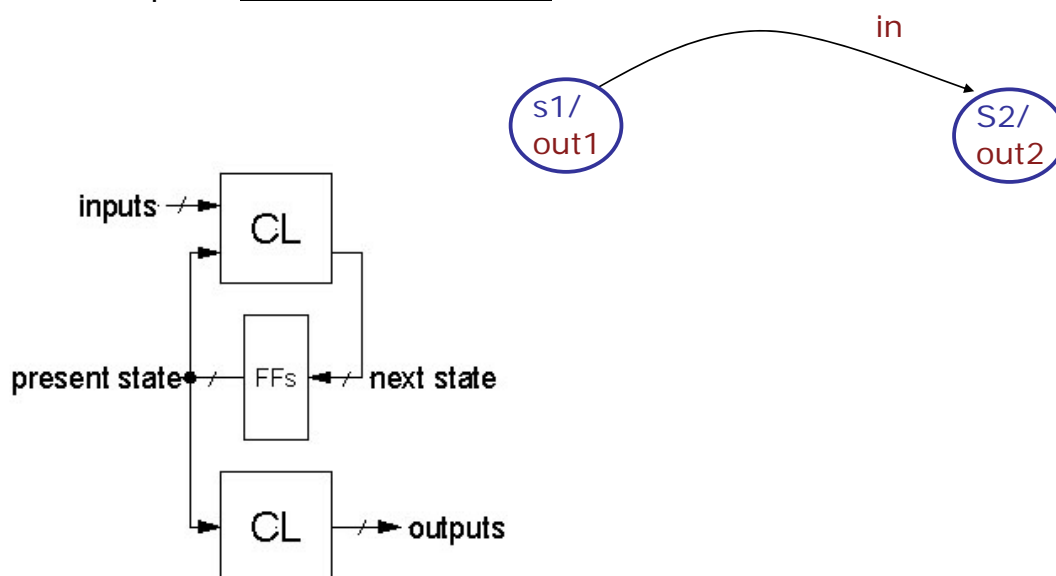
Mealy Machine

- Output based on state and present input
 - Output changes during transition



Moore Machine

- Output based on state only
 - Output is associated with state



Design of Sequential Circuits

- How can we design a sequential circuit?
 - E.g., circuit that detects 3 or more consecutive 1's in input
- Design procedure:
 1. Derive state diagram from description
 2. Reduce number of states if necessary
 3. Assign binary values to states
 4. Obtain binary coded state table (transition table)
 5. Choose type of flip-flops
 6. Derive flip-flop input equations and output equations
 7. Draw logic diagram
- Steps 1 & 3 require insight
- Steps 2, 4–7 can be automated
 - Design that follows well-defined procedure called synthesis

Next
Lecture

Canonical form of Sequential Circuits

- Graphs are hard to compare
 - Arbitrary state names
 - Arbitrary coding
- Graphs are generally very difficult to deal with
 - Determining if two graphs are identical is the “graph isomorphism problem”
 - » No known algorithm exists that can determine isomorphism in polynomial time for arbitrary graphs
 - » Problem reduces to trying every possible matching
 - » Requires exponential time (very, very long for large graphs)
- Canonical form of sequential circuit would require solution to graph isomorphism problem
- No canonical form for sequential circuits exists

State Assignment

- States are represented by flip-flop values in circuit
 - Need to encode state in binary
- What is the minimum number of flip-flops that are necessary to encode m states?
 - Need at least $\lceil \log_2(m) \rceil$ bits ($\lceil \rceil$ is “ceiling” function)

- Many possible encodings:

State	Binary	Gray	One-hot
a	000	000	00001
b	001	001	00010
c	010	011	00100
d	011	010	01000
e	100	110	10000

- Terminology:

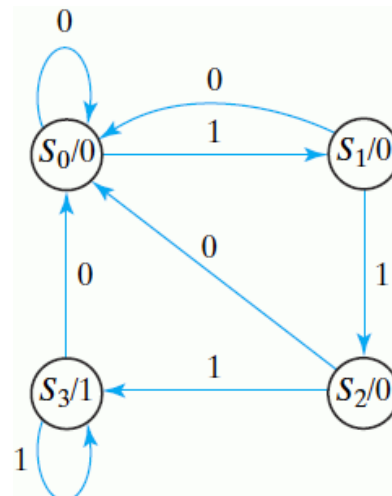
- “State table” uses uncoded states
- “Transition table” uses coded states

Example 1: Sequence Detector

- Circuit specification:
 - Design a circuit that outputs a 1 when three consecutive 1s have been applied to input, and 0 otherwise.”

- Step 1: derive state diagram

- What should a state represent?
 - » E.g., “number of 1’s seen so far”
- Moore or Mealy FSM?
 - » Both possible
 - » Chose Moore to simplify diagram
- State diagram:
 - » State S0: zero 1s detected
 - » State S1: one 1 detected
 - » State S2: two 1s detected
 - » State S3: three 1s detected



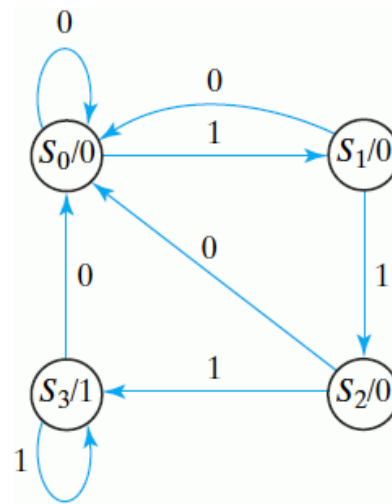
Example 1: Sequence Detector

Step 2: reduce number of states

- State table:

current state	next state		output
	x=0	x=1	
S ₀	S ₀	S ₁	0
S ₁	S ₀	S ₂	0
S ₂	S ₀	S ₃	0
S ₃	S ₀	S ₃	1

- Which states are equivalent?
 - » None – no state reduction possible



Step 3: state assignment

- Two flip-flops
- Binary state coding

Example 1: Sequence Detector

Step 4: Binary coded state table

- Name flip-flops A and B

current state		next state				output
		x=0		x=1		
A	B	A	B	A	B	
0	0	0	0	0	1	0
0	1	0	0	1	0	0
1	0	0	0	1	1	0
1	1	0	0	1	1	1

Step 5: Choose type of flip-flops

- E.g., D flip-flop
- Characteristic equation: $Q(t+1)=D_Q$

Example 1: Sequence Detector

Step 6: derive flip-flop input equations and output equation

- Use state table

- $A(t+1) = D_A(A,B,x)$
 $= \Sigma(3,5,7)$

- $B(t+1) = D_B(A,B,x)$
 $= \Sigma(1,5,7)$

- $y(A,B,x) = \Sigma(6,7)$
or $y(A,B) = \Sigma(3)$

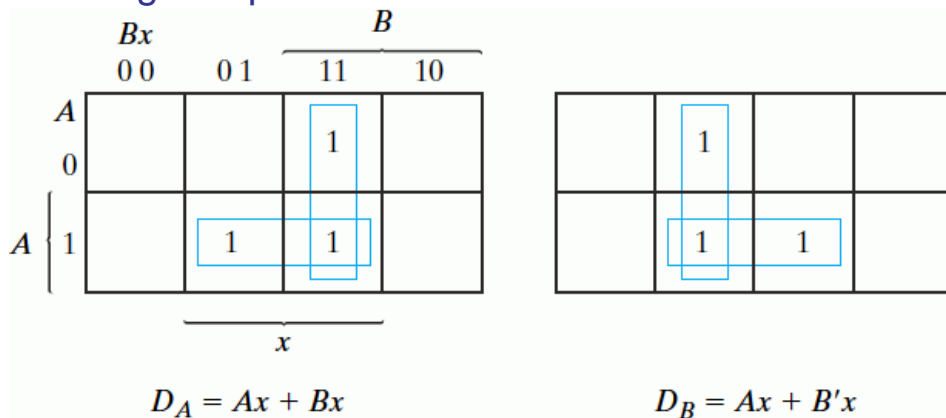
current state		input	next state		output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

Example 1: Sequence Detector

Step 6b: minimize equations

- $A(t+1) = \Sigma(3,5,7)$
- $B(t+1) = \Sigma(1,5,7)$
- $y(A,B) = \Sigma(3)$ – easy: $y = AB$

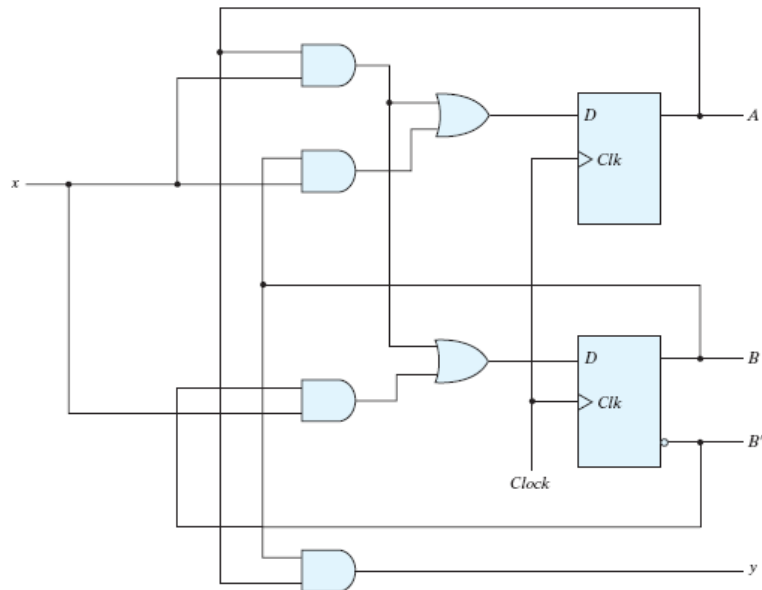
Karnaugh maps for A and B:



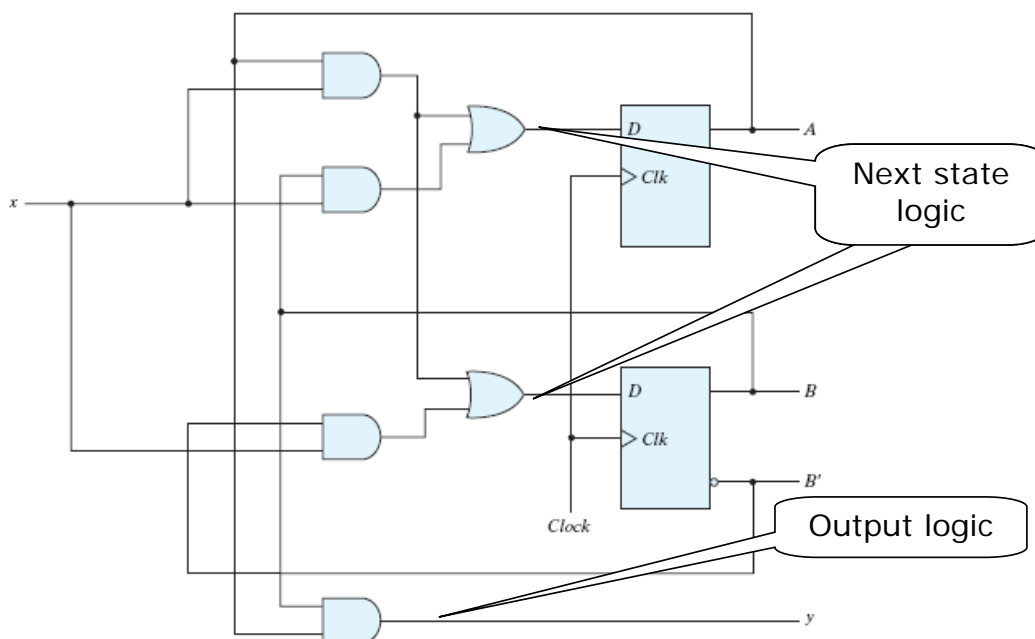
Example 1: Sequence Detector

Step 7: Circuit diagram

- $D_A = Ax + Bx$
- $D_B = Ax + B'x$
- $Y = AB$



Terminology



Summary

- Finite state machines form the basis of many digital systems
- Designs often start from clear specifications
- Develop state diagram and state table
- Optimize using combinational design techniques
 - Output logic
 - Next state logic
- Mealy or Moore implementations possible