Chap 5
Synchronous Sequential Logic (4)

Hanwool Jeong

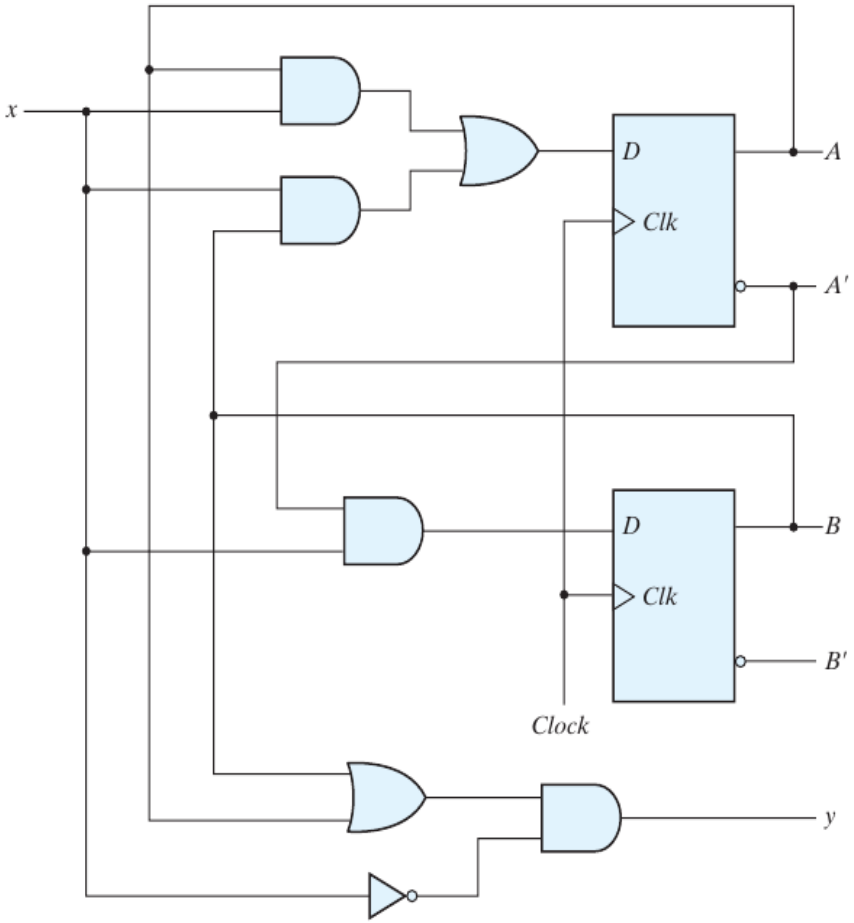
hwjeong@kw.ac.kr

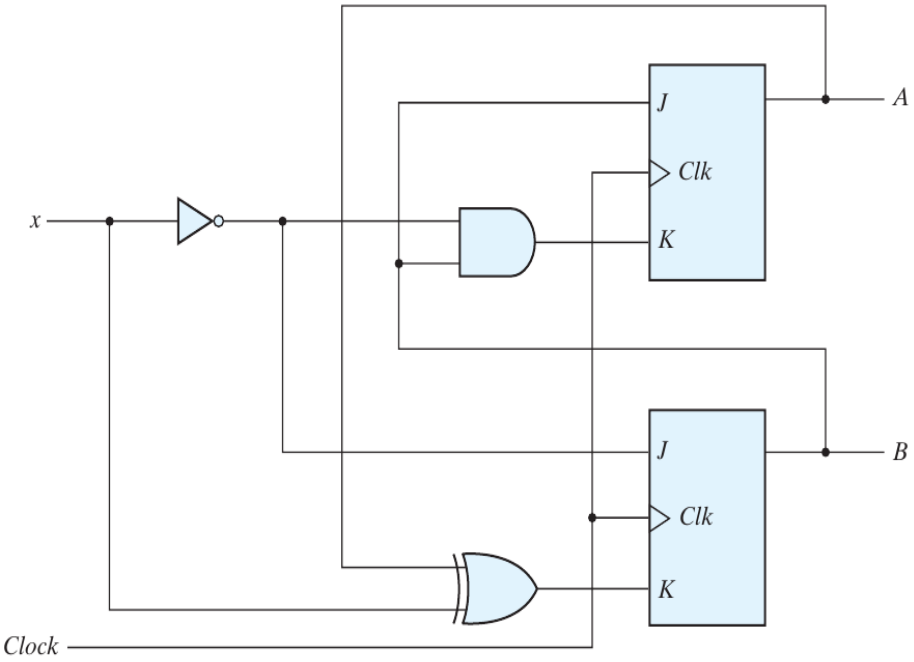
What We have Learned vs. What We will Learn

- What we have learned
  - Operation ➔ Circuit
  - Circuit ➔ Operation
- What we will learn
  - When we do Operation ➔ Circuit, there are somethings to learn more for more delicate circuit design. That is,
  - Mealy machine vs. Moore Machine
  - Reset in Flip-flop
  - State Reduction
  - Systematic flow

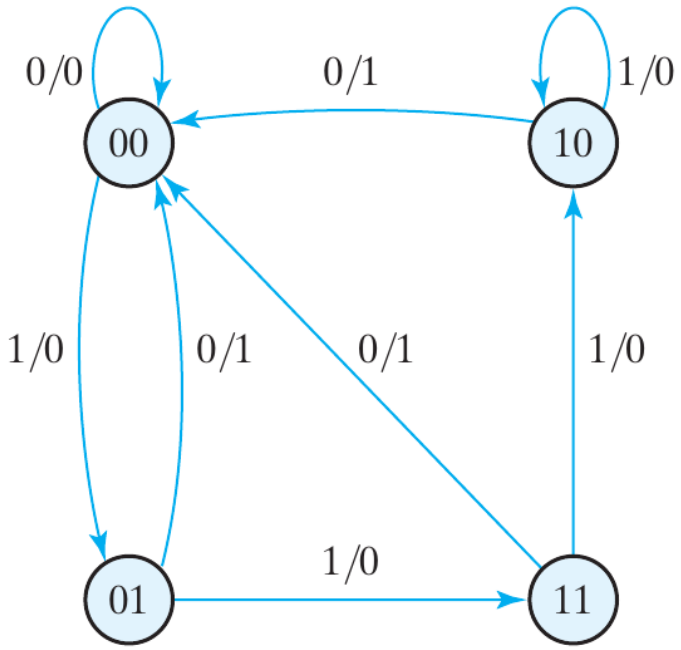
Revisit Two Circuits in Prev. Lecture

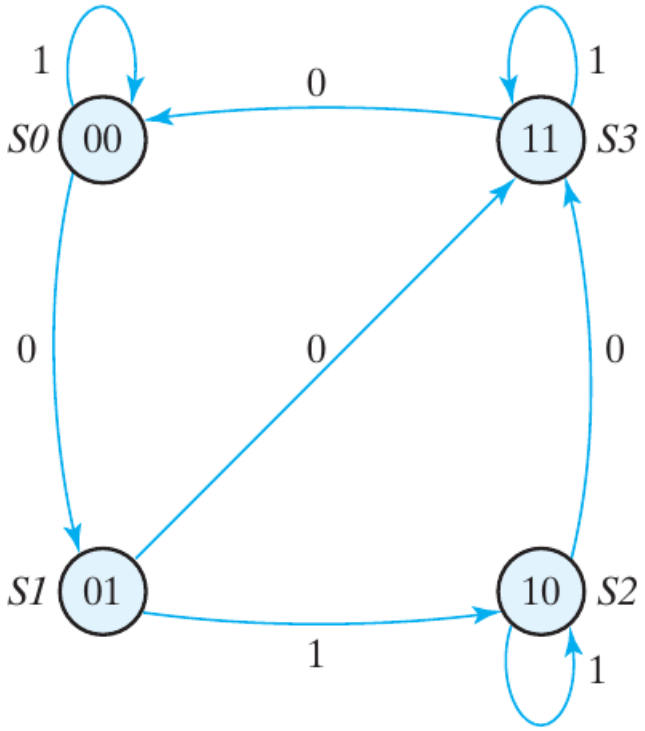
- Differ in the way the output is generated





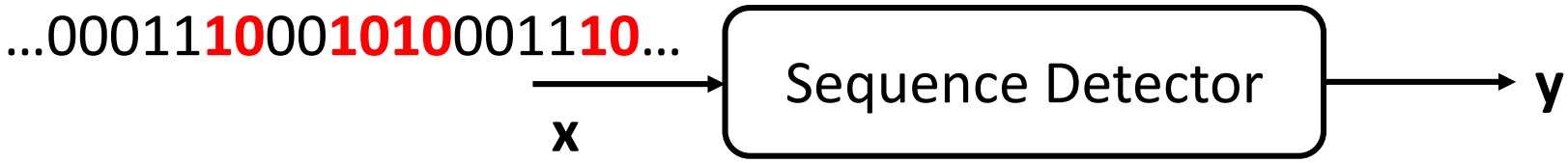
State Diagram?





Imagine “10” Sequence Detector

- Suppose that you design a “10” sequence detector
- Is it sequential logic? Do we define initial states of circuits?

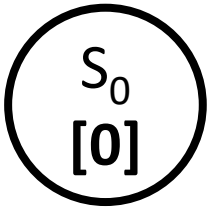
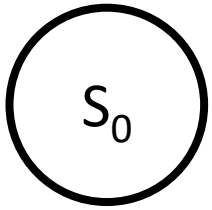
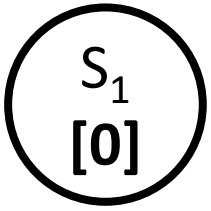
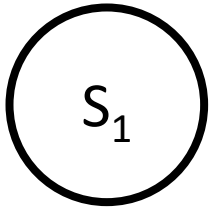


- Sure, we should define finite number of states

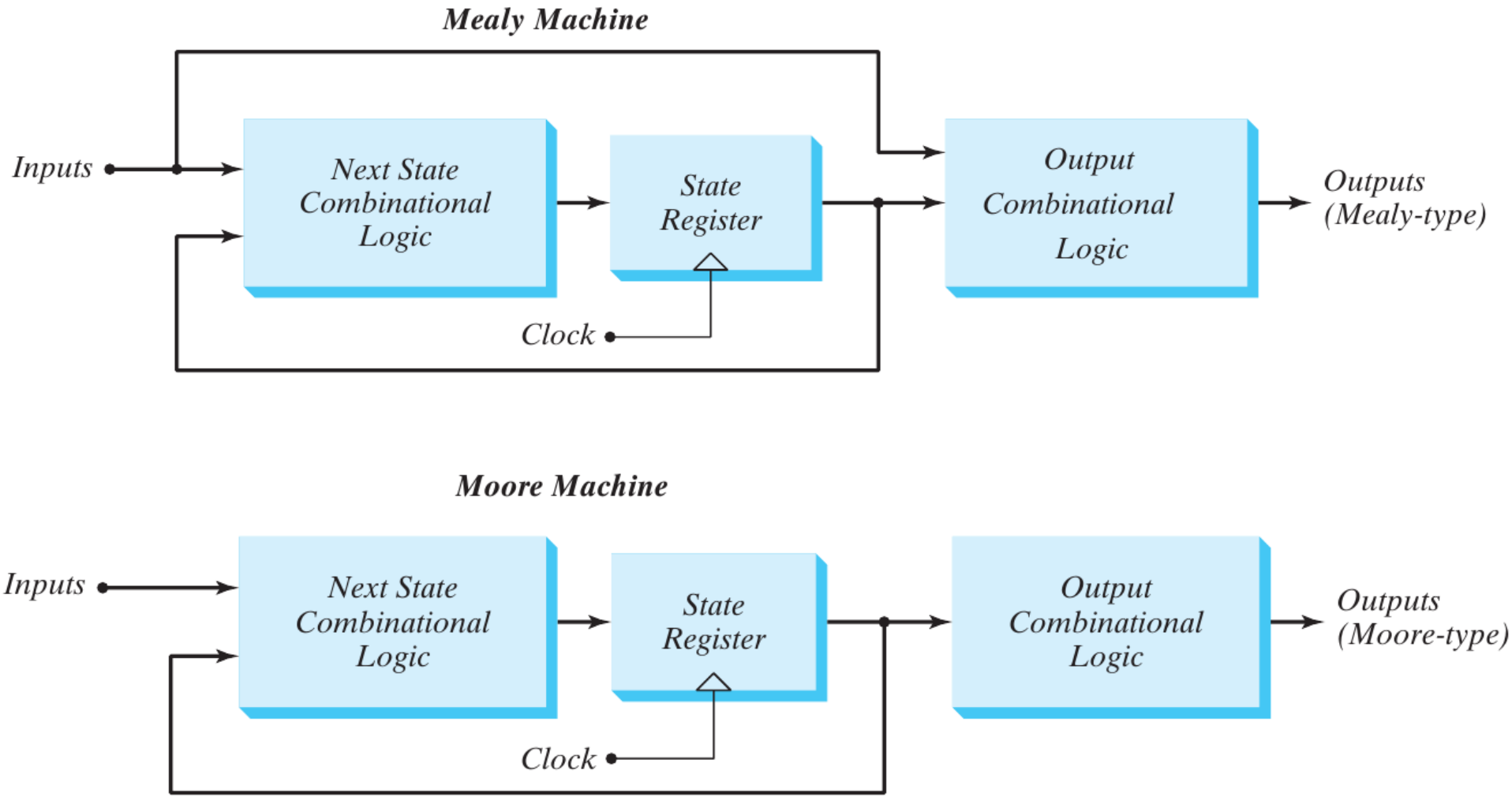
➔ Finite state machine (FSM)

“10” Detector State Diagram

- Two ways to realize the sequence detector

State = Output	Using Separate output
 “1” not received	 “1” not received
 “1” received but not “0” yet	 “1” received but not “0” yet
 “10” is received	**What if “0” is received?**
Moore Machine	**Mealy Machine**

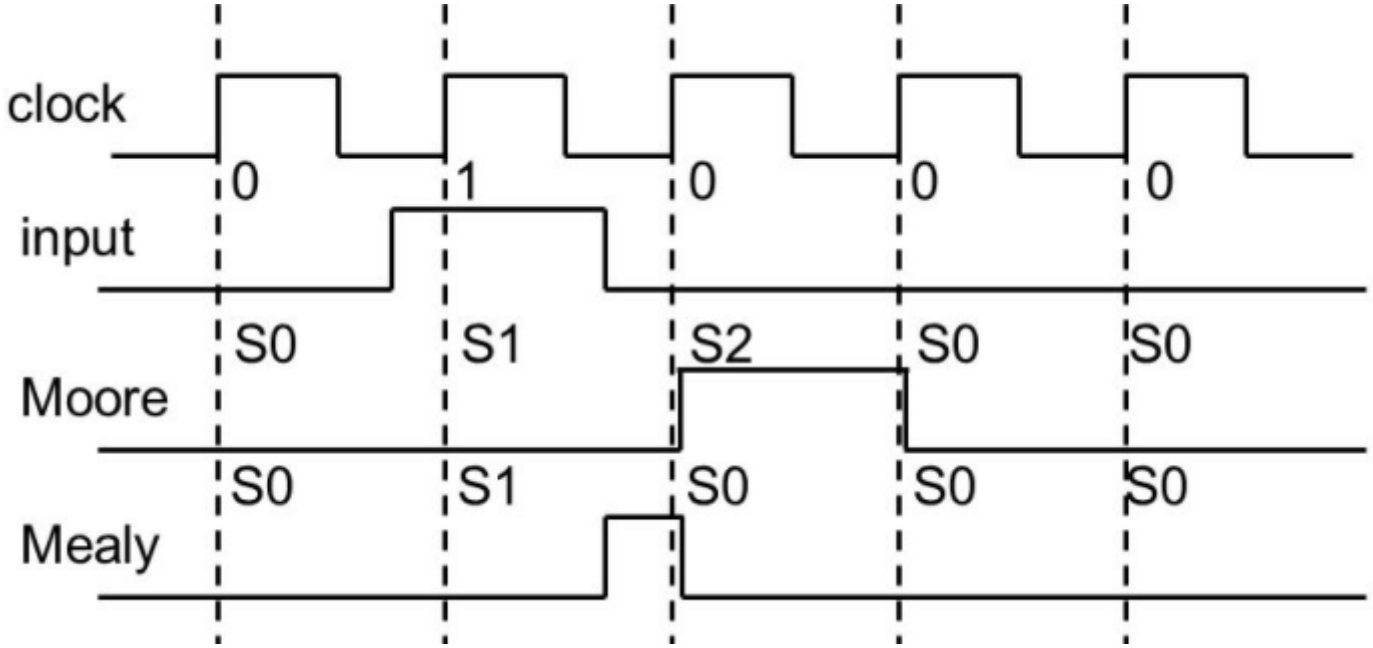
Mealy and Moore Finite State Machine



Requirement for Mealy Machine

- Moore : output is synchronized with the clock (why?)
- Mealy
  - Output changes with input change during clock cycle
  - Output may be momentarily false due to input to FF output delay
- Thus,
  - Inputs of the seq. circuit must be synchronized with clock and the outputs must be sampled immediately before (at) the clock edge
  - The output of Mealy machine is the value that is present immediately before the edge of the clock.

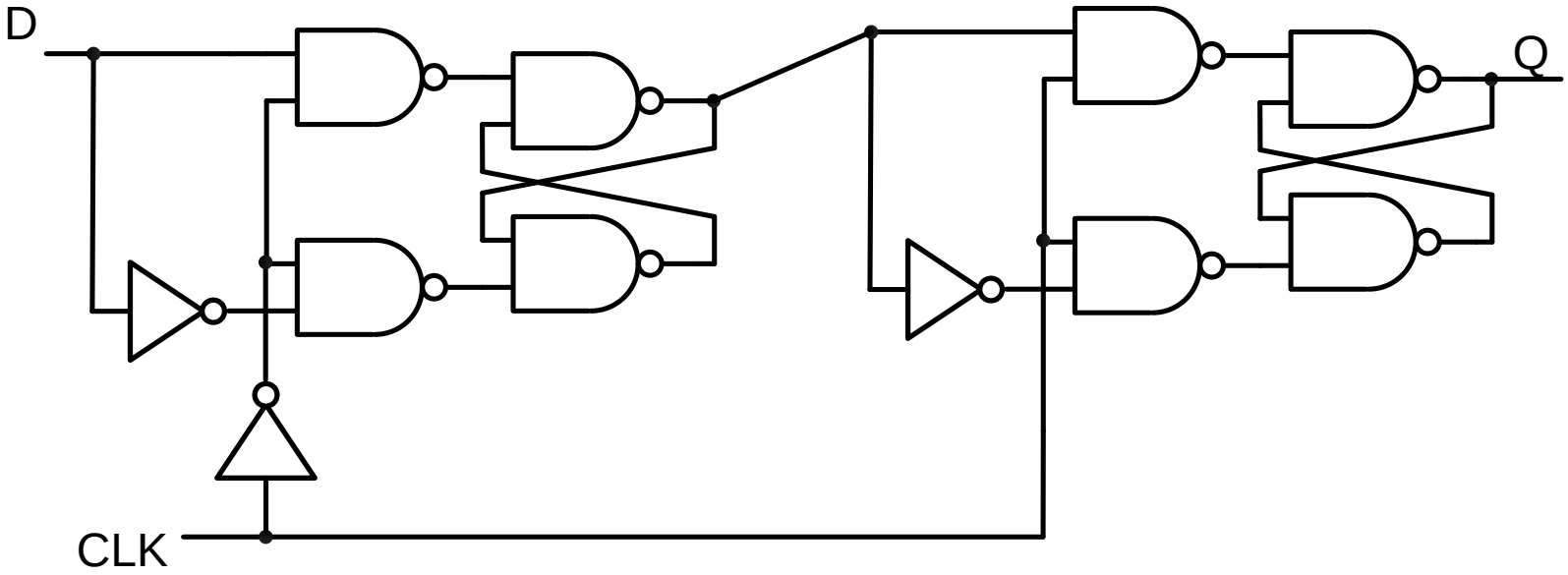
“10” Detector Example



Synchronous Mealy Machine

Direct input to FF

- Employing “asynchronous” reset signal
- Why do we need reset signal?

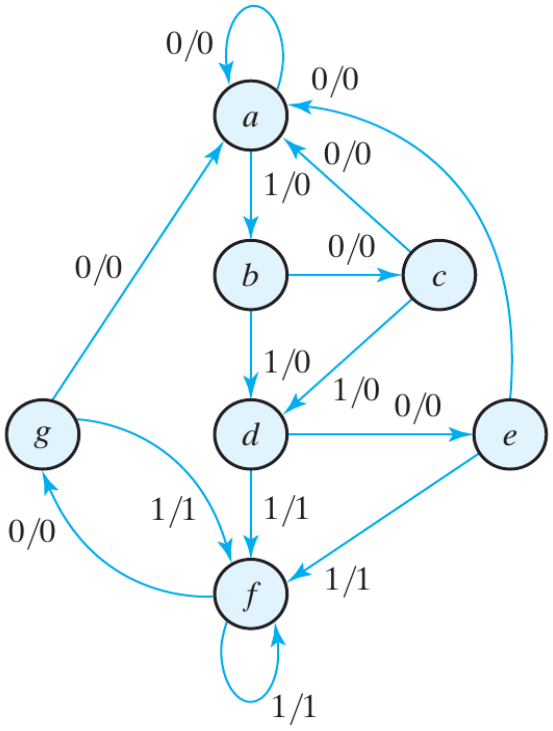


Motivation for State Reduction

- Two sequential circuits may exhibit the same input–output behavior, but have a different number of internal states in their state diagram
  ➔ When it comes to circuit implementation, the number of flip-flop can be reduced.
- Is always good?

Condition of Equivalent Circuit?

- The states are of second importance.
- Only output sequence caused by input sequence matters
  ➔ If identical input sequences are applied to the two circuits and identical outputs occur for all input sequences, then the two circuits are said to be equivalent



state	*a*	*a*	*b*	*c*	*d*	*e*	*f*	*f*	*g*	*f*	*g*
input	0	1	0	1	0	1	1	0	1	0	0
output	0	0	0	0	0	1	1	0	1	0	0

Strategy for Finding Equivalent States in the State Table

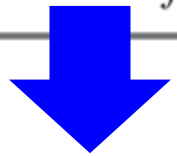
- Two states are equivalent if, for each member of the set of inputs, **they give exactly the same output and send the circuit either to the same state or to an equivalent state**

➔ Then one of the two can removed; See e and g states

State Table

	Next State		Output	
Present State	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	e	f	0	1



Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

State Assignment

- How can you “encode” the state to binary form?

Three Possible Binary State Assignments

State	Assignment 1, Binary	Assignment 2, Gray Code	Assignment 3, One-Hot
a	000	000	00001
b	001	001	00010
c	010	011	00100
d	011	010	01000
e	100	110	10000

Transition of the node
Requires time/power

- Binary example

	Next State		Output	
Present State	$x = 0$	$x = 1$	$x = 0$	$x = 1$
000	000	001	0	0
001	010	011	0	0
010	000	011	0	0
011	100	011	0	1
100	000	011	0	1

Design Procedure

1. From the word description and specifications of the desired operation, derive a state diagram for the circuit.
2. Reduce the number of states if necessary.
3. Assign binary values to the states.
4. Obtain the binary-coded state table.
5. Choose the type of flip-flops to be used.
6. Derive the simplified flip-flop input equations and output equations.
7. Draw the logic diagram.

➔ The part of the design that follows a well-defined procedure is referred to as synthesis

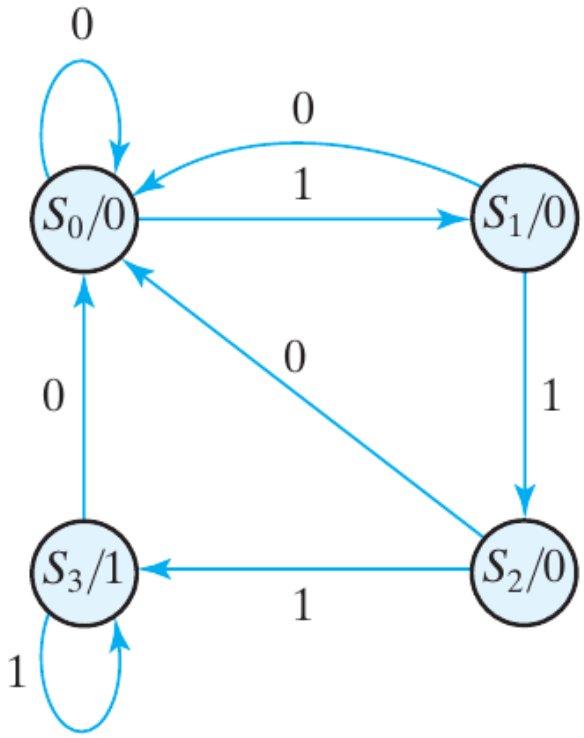
Can You Perform the 1st Step?

1. From the word description and specifications of the desired operation, derive a state diagram for the circuit.

- Suppose we wish to design a circuit that detects a sequence of three or more consecutive 1's in a string of bits coming through an input line ➔ Sequence detector

Synthesis Using D-FF

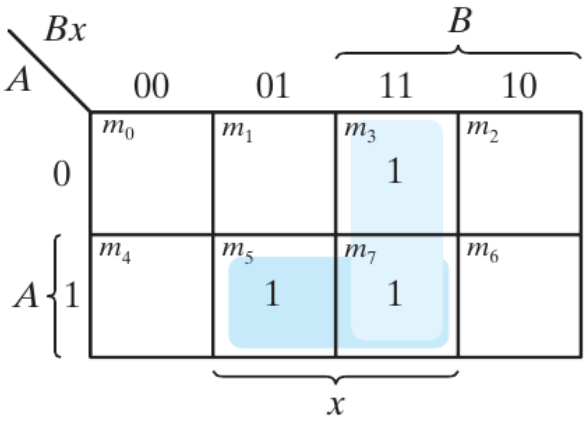
- We need state table to establish equations



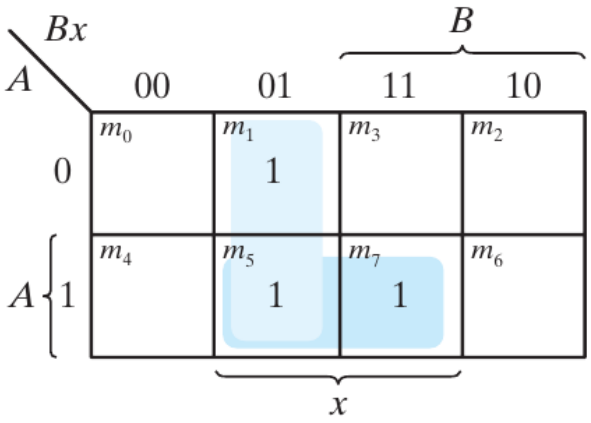
State Table for Sequence Detector

Present 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

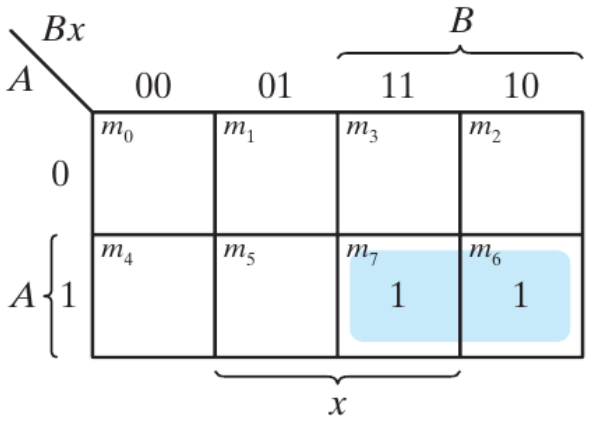
Using K-Map



$D_A = Ax + Bx$



$D_B = Ax + B'x$



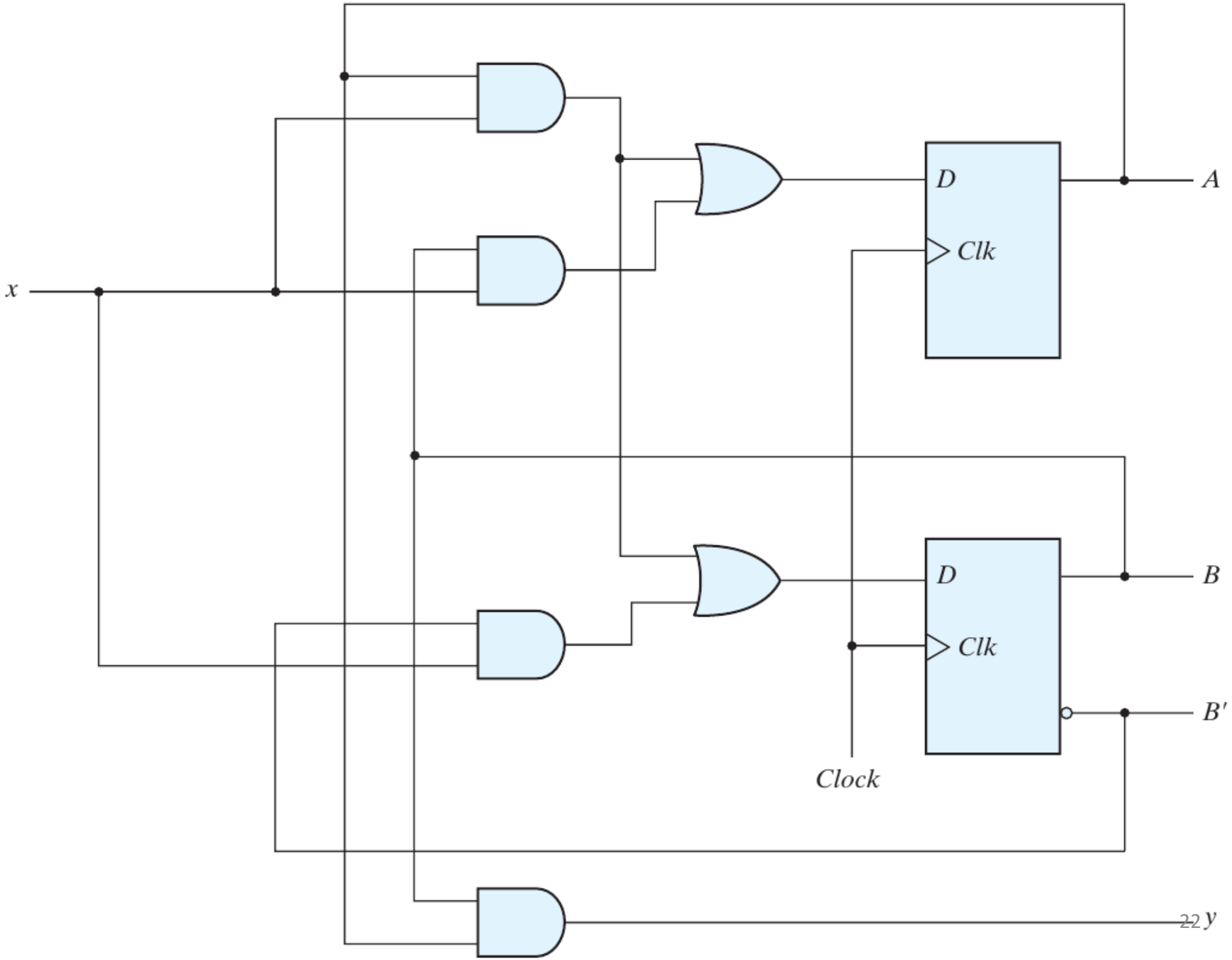
$y = AB$

Can You Design Circuit Diagram?

- With D-FF, it is quite simple. Why?

$$A(t+1) = D_A(A, B, x) = \Sigma(3, 5, 7)$$
$$B(t+1) = D_B(A, B, x) = \Sigma(1, 5, 7)$$
$$y(A, B, x) = \Sigma(6, 7)$$

$$D_A = Ax + Bx$$
$$D_B = Ax + B'x$$
$$y = AB$$


x
D
Clk
A
D
Clk
B
B′
Clock
y

Excitation Tables

- For D-FF, input equations are determined from the next state.
- Not for JK-FFand T-FF. For these FFs, a functional relationship b/w the state table and the input equations is needed.
- During the design, we know a certain transition and wish to know what FF input condition is to achieve the req'd transition. ➔ Table listing state transition vs. req'd input is needed, which is called excitation table

JK Flip-Flop

J	K	Q(t + 1)	
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$Q'(t)$	Complement

T Flip-Flop

T	Q(t + 1)	
0	$Q(t)$	No change
1	$Q'(t)$	Complement

Flip-Flop Excitation Tables

Q(t)	Q(t = 1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Q(t)	Q(t = 1)	T
0	0	0
0	1	1
1	0	1
1	1	0

Synthesis Using JK-FF; State Table w/ JK Inputs

- We need FF Input parts for the table as

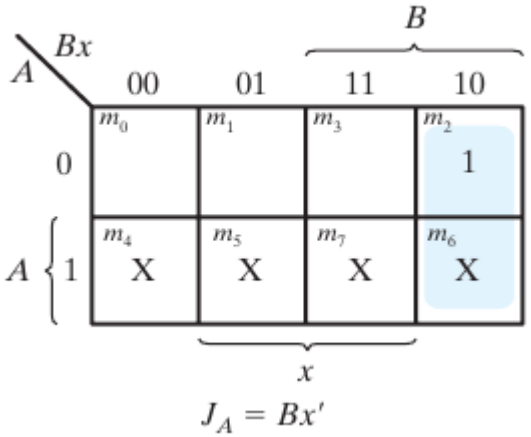
Flip-Flop Excitation Tables

$Q(t)$	$Q(t+1)$	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

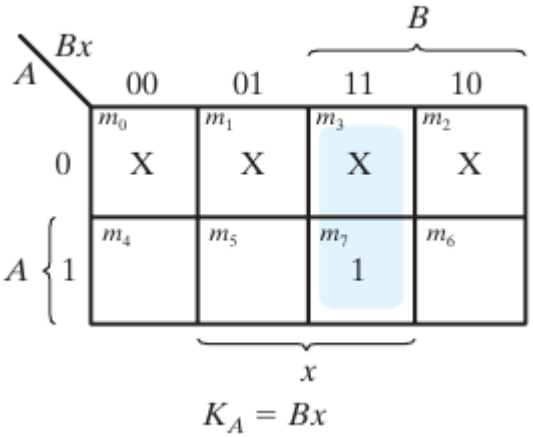
State Table and JK Flip-Flop Inputs

Present State		Input	Next State		Flip-Flop Inputs			
A	B	x	A	B	J_A	K_A	J_B	K_B
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1

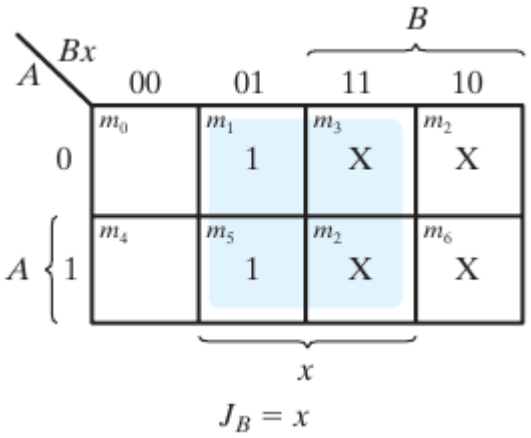
Synthesis Using JK-FF; State Table w/ JK Inputs



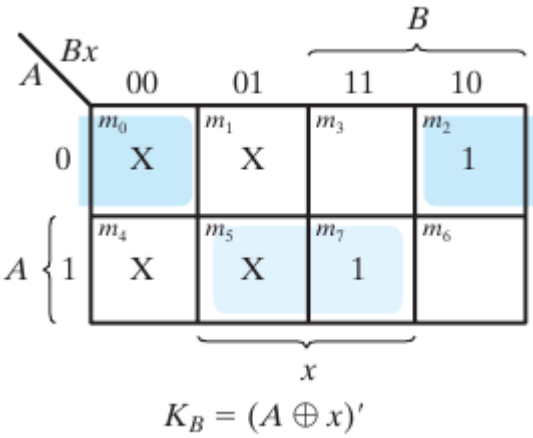
$J_A = Bx'$



$K_A = Bx$

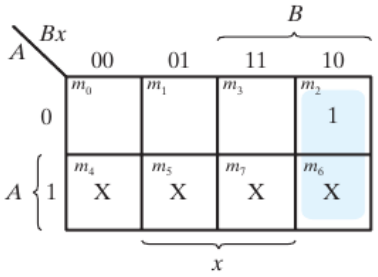


$J_B = x$

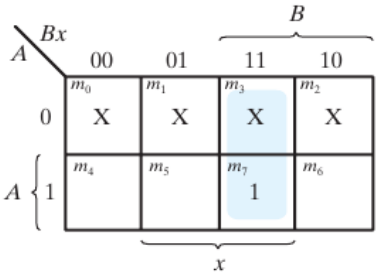


$K_B = (A \oplus x)'$

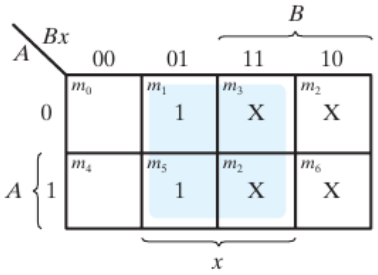
Can you Derive Circuit Diagram?



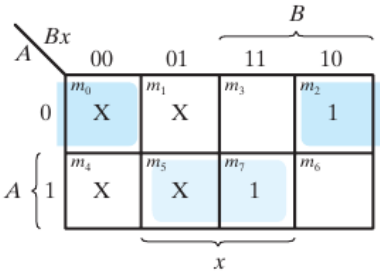
$J_A = Bx'$



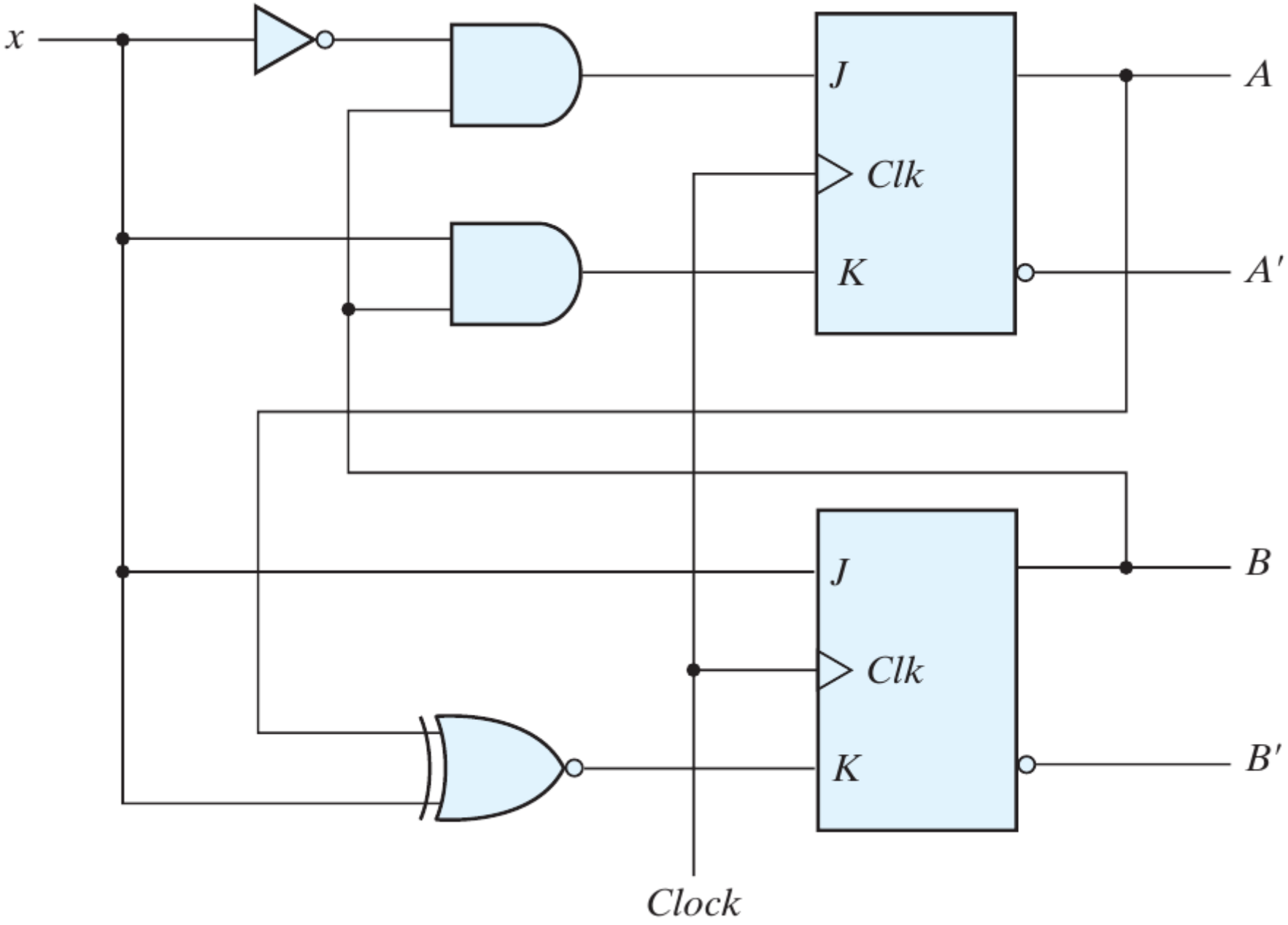
$K_A = Bx$



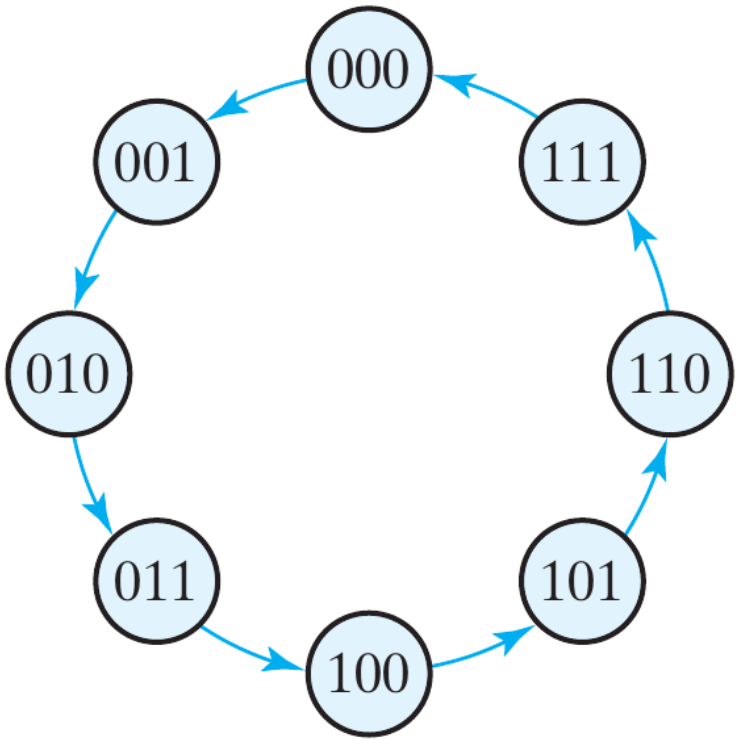
$J_B = x$



$K_B = (A \oplus x)'$


x
J
Clk
K
A
A′
J
Clk
K
B
B′
Clock

Three Bit Counter Synthesis Using T-FF



Present State			Next State		
A_2	A_1	A_0	A_2	A_1	A_0
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

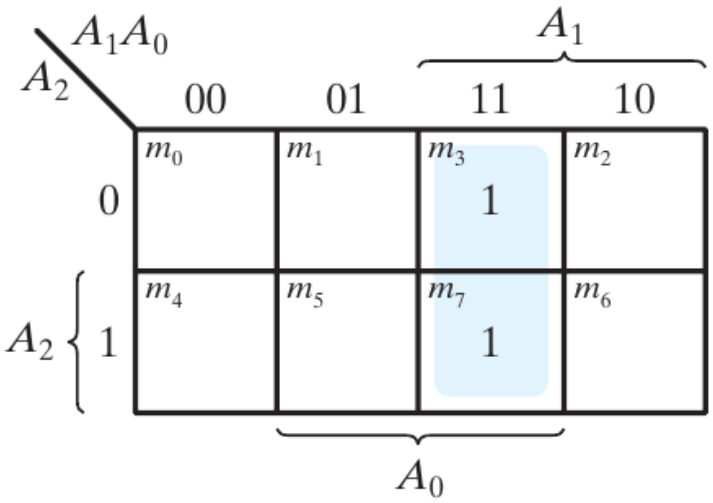
State Table w/ FF Input

$Q(t)$	$Q(t = 1)$	T
0	0	0
0	1	1
1	0	1
1	1	0

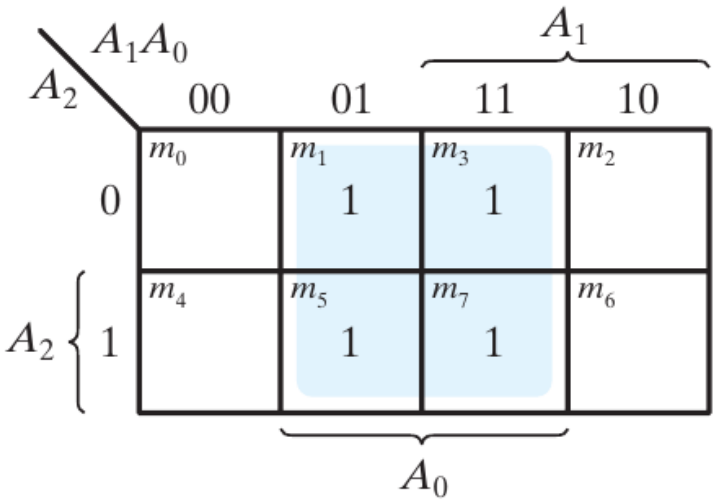
State Table for Three-Bit Counter

Present State			Next State			Flip-Flop Inputs		
A_2	A_1	A_0	A_2	A_1	A_0	T_{A2}	T_{A1}	T_{A0}
0	0	0	0	0	1			
0	0	1	0	1	0			
0	1	0	0	1	1			
0	1	1	1	0	0			
1	0	0	1	0	1			
1	0	1	1	1	0			
1	1	0	1	1	1			
1	1	1	0	0	0			

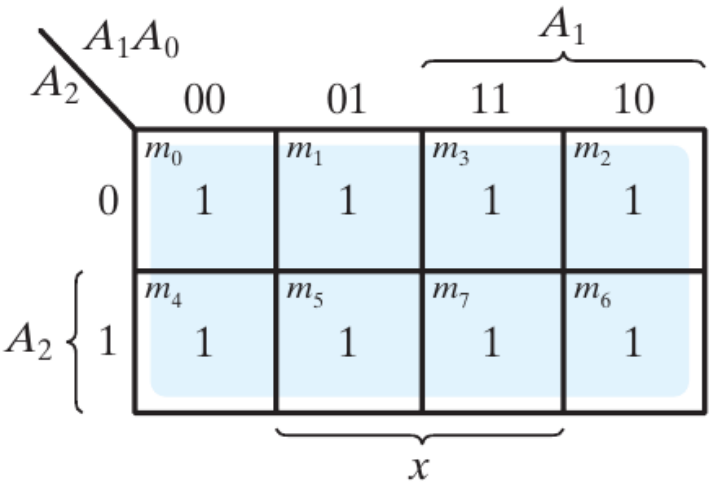
K-Map



$T_{A2} = A_1A_0$



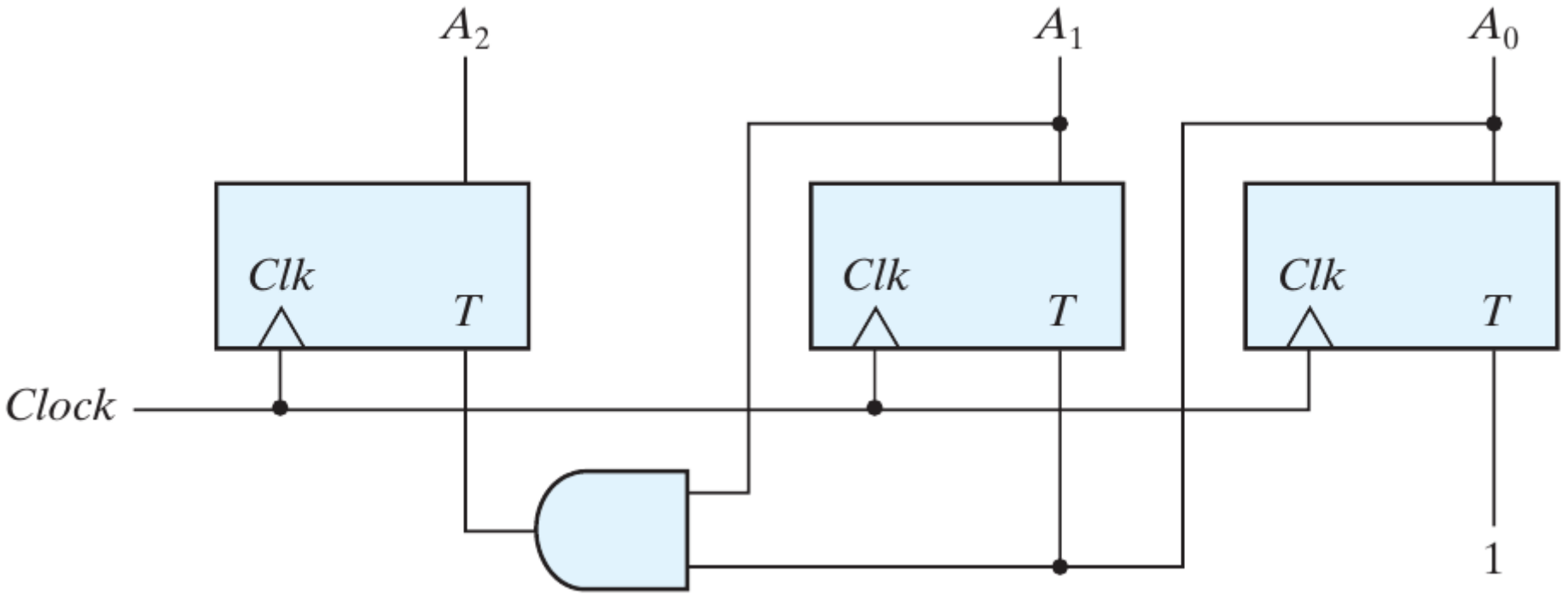
$T_{A1} = A_0$



$T_{A0} = 1$

Circuit

$$T_{A2} = A_1 A_0 \qquad T_{A1} = A_0 \qquad T_{A0} = 1$$


A_2
A_1
A_0
Clk
T
Clk
T
Clk
T
Clock
1

What We Have Learned

- Mealy machine vs. Moore Machine
- Reset in Flip-flop & State Reduction
- Circuit Synthesis Examples