CHAPTER-5b

Mealy Model and Moore Model

Digital Design (with an introduction to the Verilog HDL) 6th Edition,
M. Morris Mano, Michael D. Ciletti

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY,
DESIGN AND MANUFACTURING,
KANCHEEPURAM

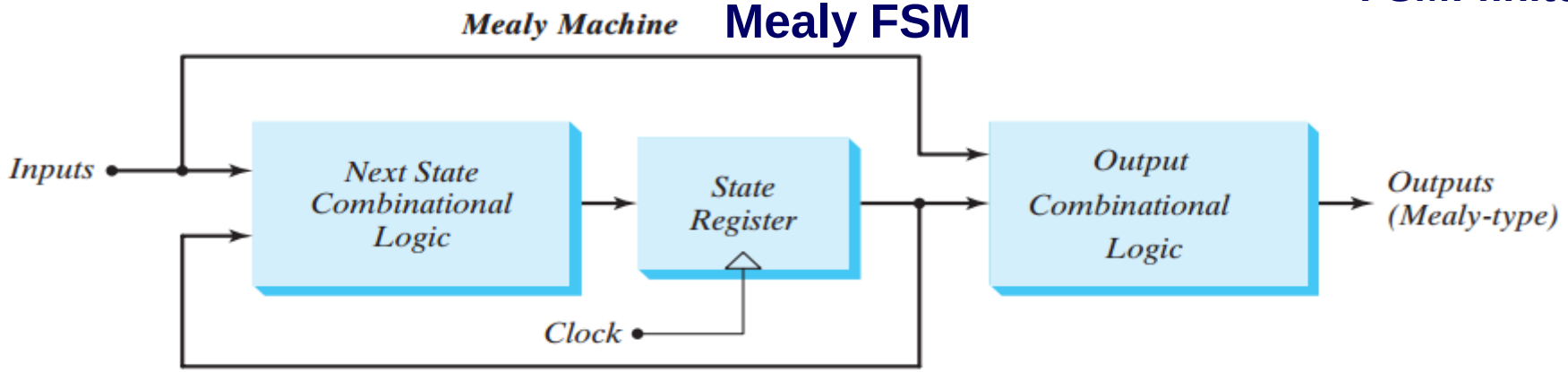
ECE, IIITDM Kancheepuram

Mealy Model and Moore Model

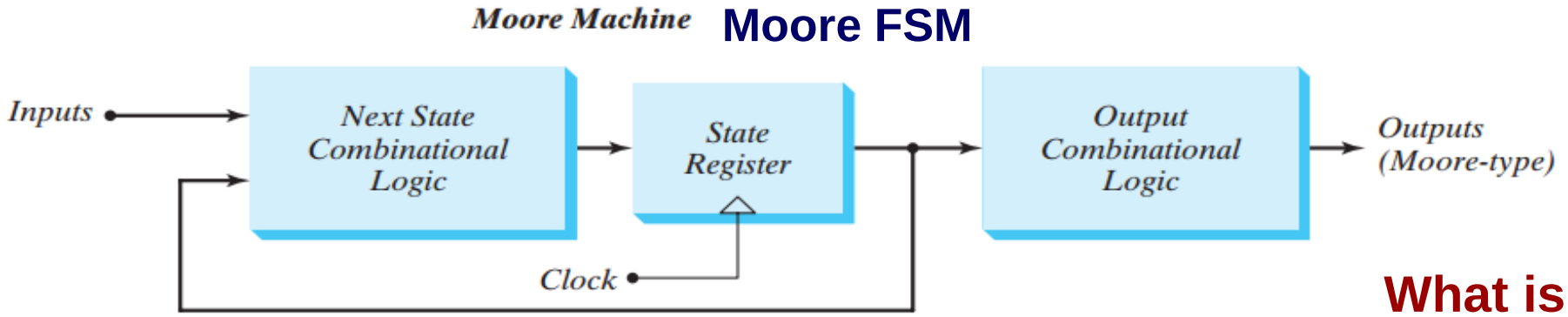
- A **state machine** is a sequential circuit having a limited (finite) number of states occurring in a prescribed order.

FSM: finite state machine

Mealy FSM



Moore FSM



What is the difference?

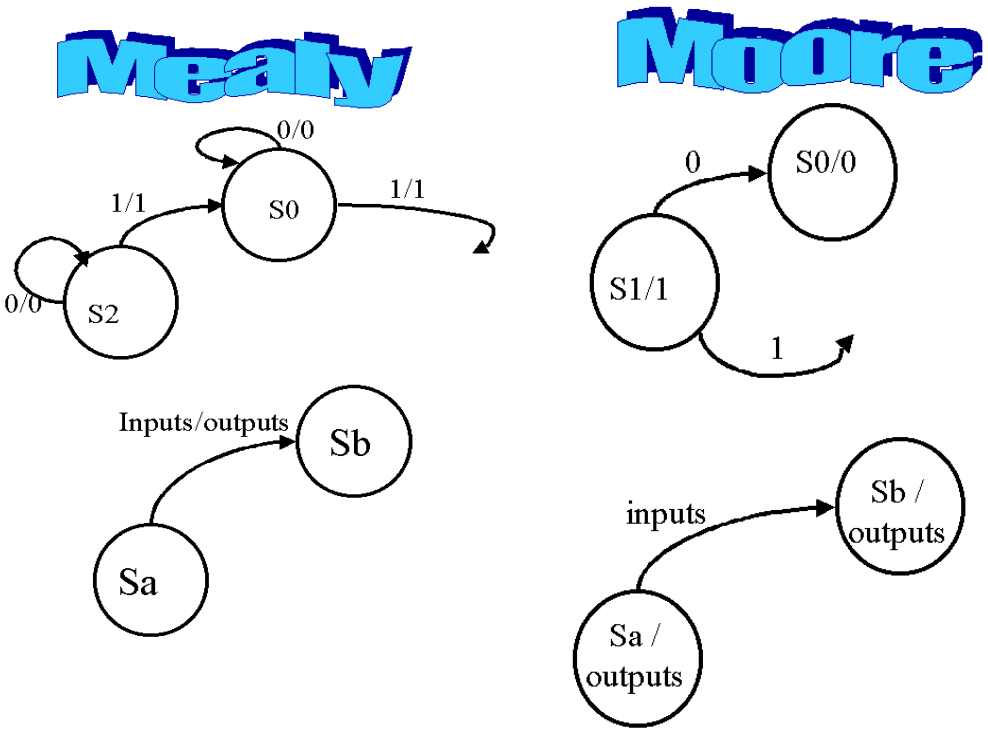
Mealy Model and Moore Model

In the Mealy model

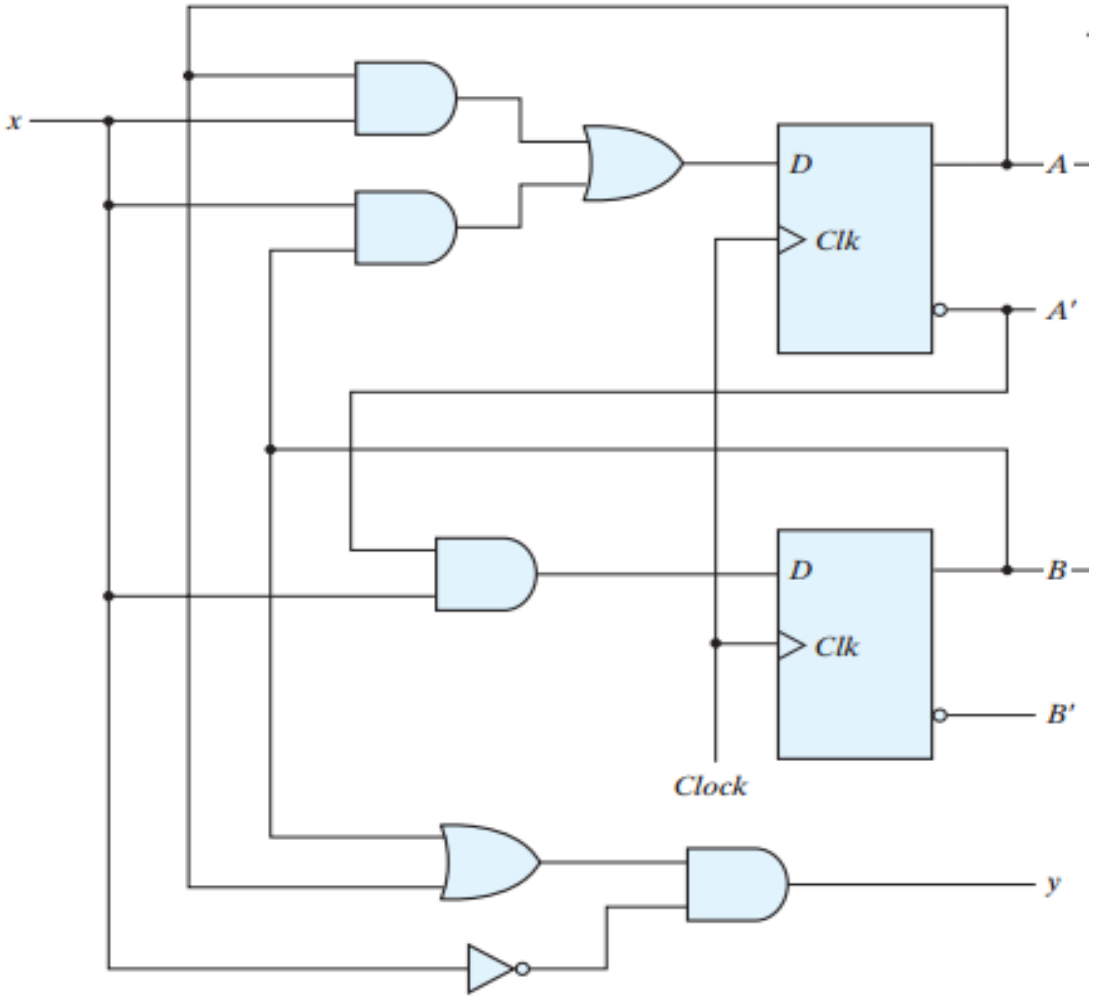
- The output is a function of both the present state and the input.
- The outputs may change if the inputs change during the clock cycle.

In the Moore model

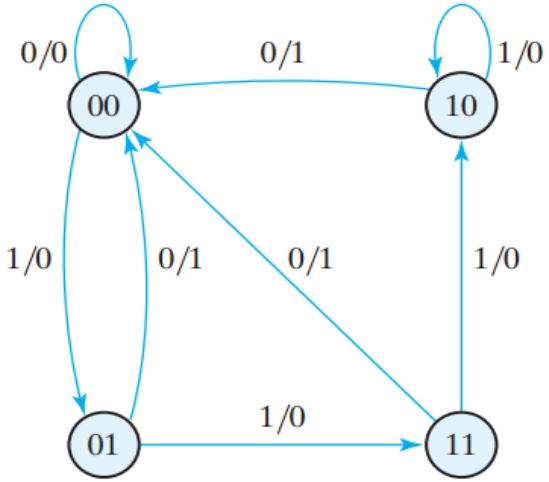
- The output is a function of only the present state.



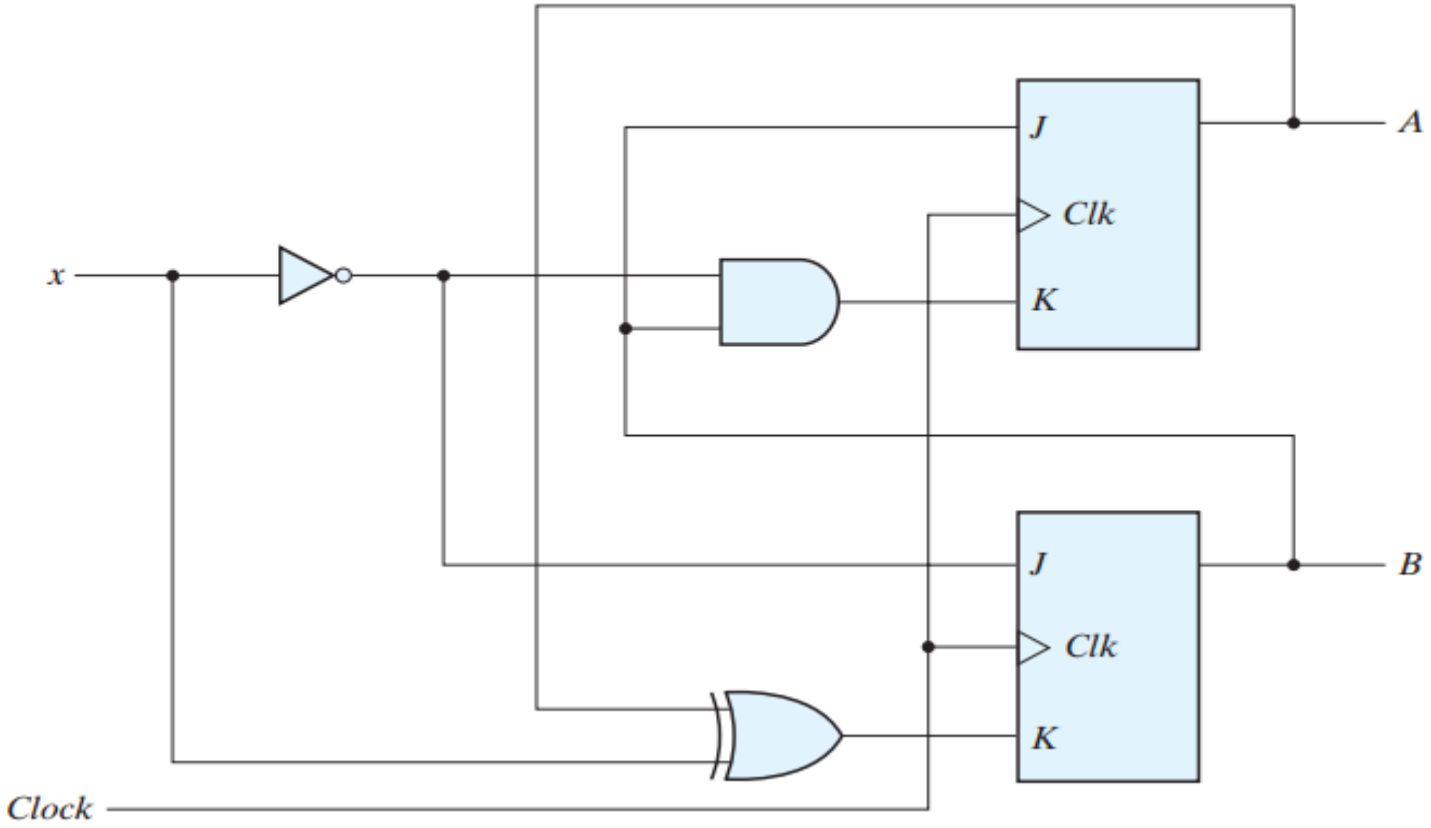
Mealy Machine



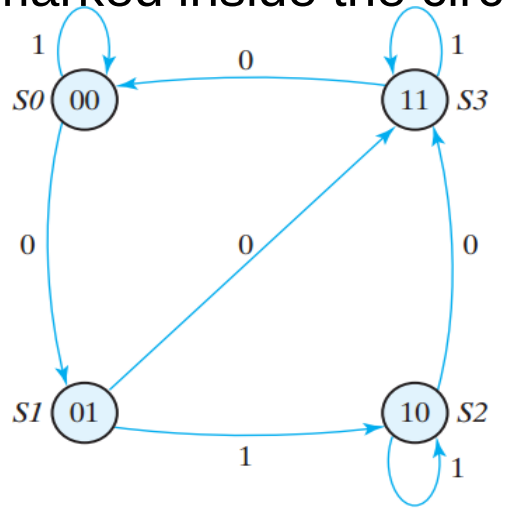
- Output *y* is a function of both input *x* and the present state of *A* and *B* .
- The corresponding state diagram shows both the input and output values, separated by a slash along the directed lines between the states.



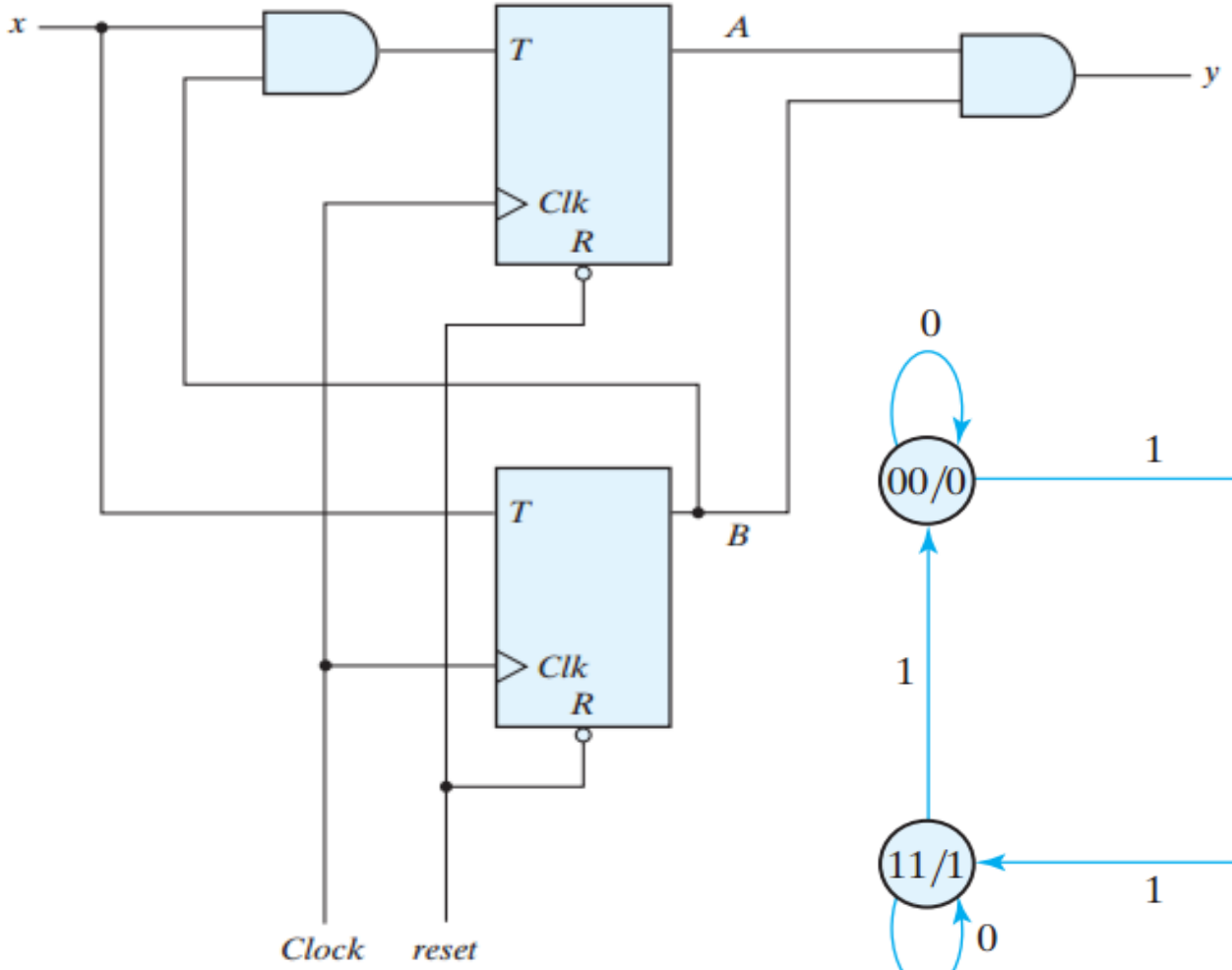
Moore Model



- The output is a function of the present state only.
- The corresponding state diagram has only inputs marked along the directed lines.
- The outputs are the flip-flop states marked inside the circles.



Moore Model



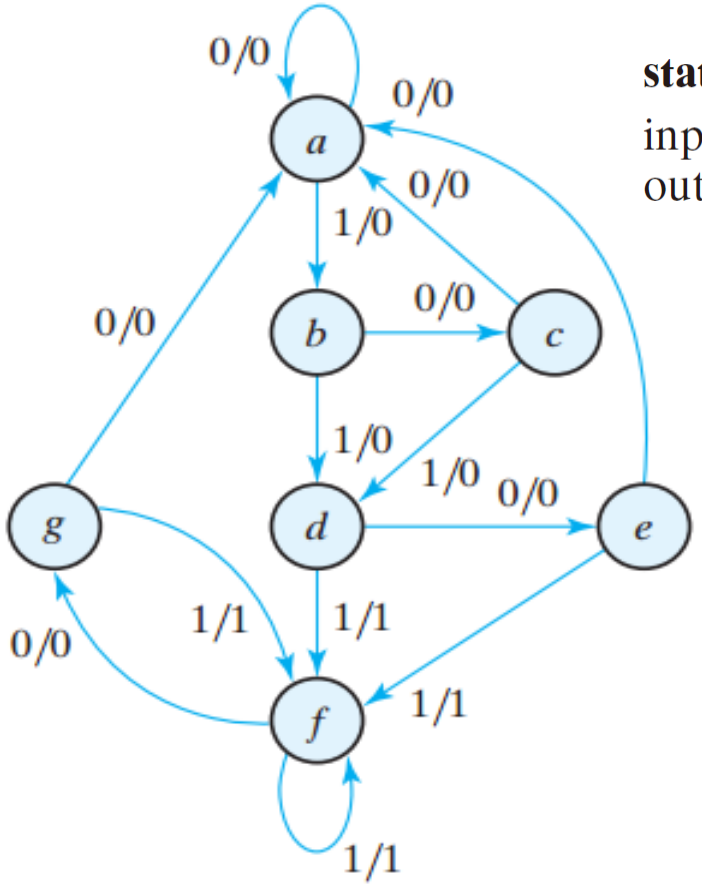
- The output depends only on flip-flop values, and that makes it a function of the present state only.
- The input value in the state diagram is labeled along the directed line, but the output value is indicated inside the circle together with the present state.

State Reduction and Assignment

- *m* flip-flops produce 2^m states, a reduction in the number of states may result in a reduction in the number of flip-flops.
- The reduction in the number of flip-flops in a sequential circuit is referred to as the state-reduction problem.
- If identical input sequences are applied to two circuits and identical outputs occur for all input sequences, then the two circuits are said to be equivalent.
- State reduction is to find ways of reducing the number of states in a sequential circuit without altering the input–output relationships.

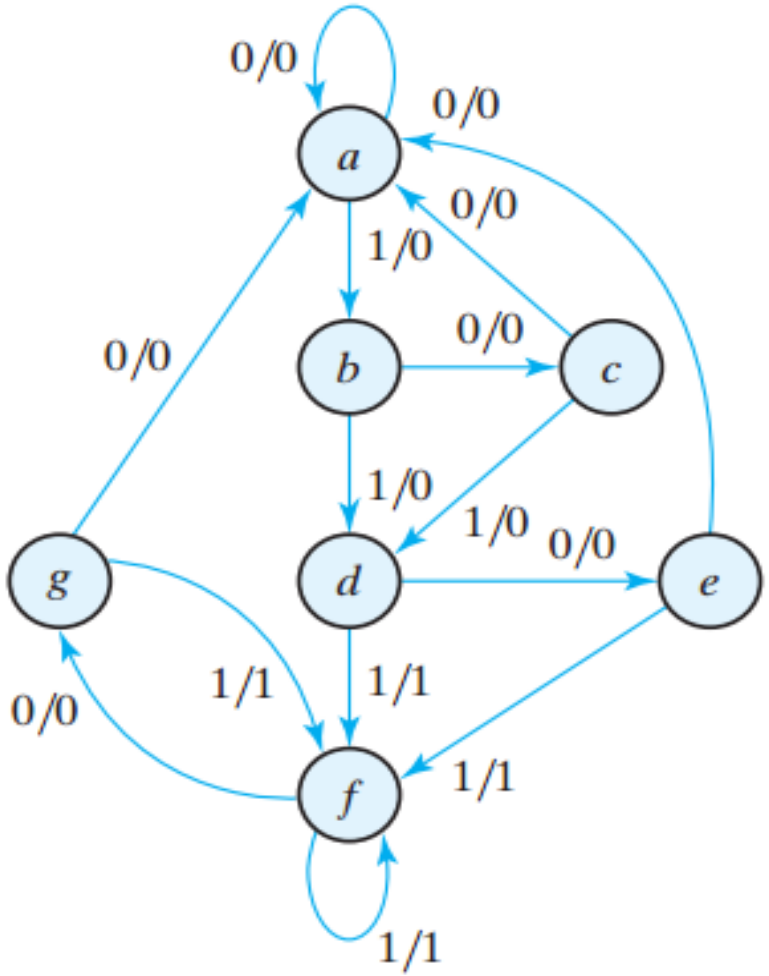
State Reduction

- As an example, consider the input sequence **01010110100** starting from the initial state *a*.



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

State Reduction



Present State	Next State $x=0$	Next State $x=1$	Output $x=0$	Output $x=1$
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

- "*g*" and "*e*" are equivalent.
- Remove the present state "*g*".
- Change next state "*g*" to "*e*".

"Two states are said to be equivalent if, for each member of the set of inputs, they give exactly the same output and send the circuit to the same state."

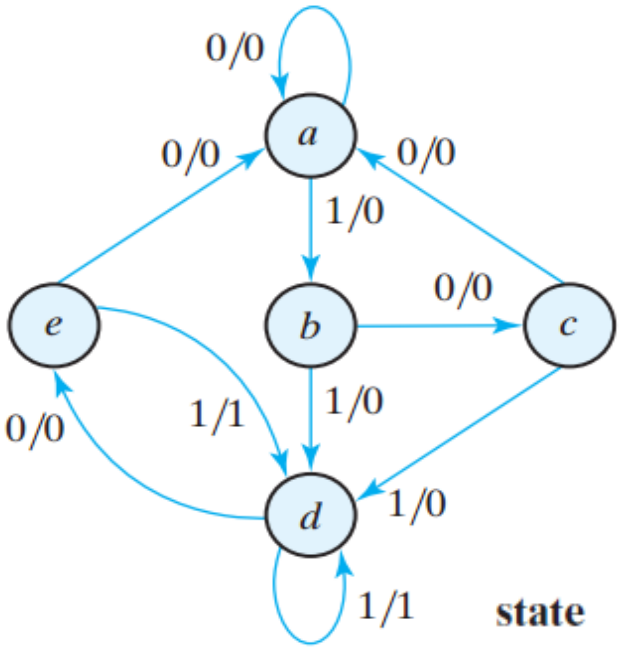
State Reduction

Present State	Next State $x=0$	Next State $x=1$	Output $x=0$	Output $x=1$
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 $x=0$	Next State $x=1$	Output $x=0$	Output $x=1$
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

- "*d*" and "*f*" are equivalent.
- Remove the present state "*f*". Change next state "*f*" to "*d*".

State Reduction



Present State	Next State $x=0$	Next State $x=1$	Output $x=0$	Output $x=1$
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

- This state diagram satisfies the original input–output specifications and will produce the required output sequence for any given input sequence.
- For the input sequence used previously (01010110100), the same output sequence results although the state sequence is different.

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

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

State Assignment

- In order to design a sequential circuit with physical components, it is necessary to assign unique coded binary values to the states.
- Code the states to unique binary values.

Table 5.9
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

State Assignment

- Any binary number assignment is satisfactory as long as each state is assigned a unique number.

Table 5.10
Reduced State Table with Binary Assignment 1

	Present State	Next State $x = 0$	Next State $x = 1$	Output $x = 0$	Output $x = 1$
a	000	000	001	0	0
b	001	010	011	0	0
c	010	000	011	0	0
d	011	100	011	0	1
e	100	000	011	0	1

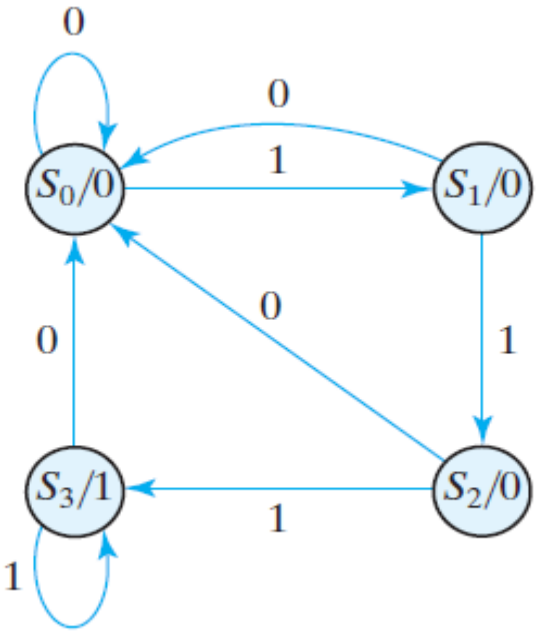
Design Procedure

- We have learned how to derive a state diagram from a sequential circuit: (Sequential Circuit → State Equations → State Table → State Diagram)
- If the specification/state diagram is given, how do we design a sequential circuit?

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.

Design Procedure

- 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 (i.e., the input is a *serial bit stream*).
- It is derived by starting with state S_0, the reset state.

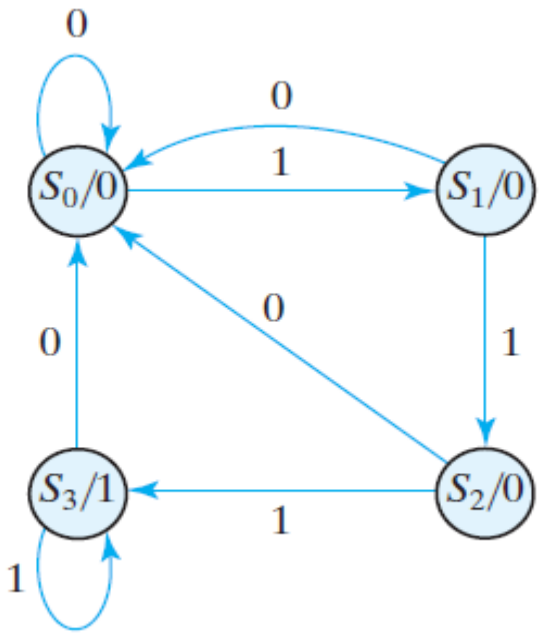


- If the input is 0, the circuit stays in S_0, but if the input is 1, it goes to state S_1 to indicate that a 1 was detected.
- If the next input is 1, the change is to state S_2 to indicate the arrival of two consecutive 1's, but if the input is 0, the state goes back to S_0.
- The third consecutive 1 sends the circuit to state S_3. If more 1's are detected, the circuit stays in S_3. Any 0 input sends the circuit back to S_0.
- In this way, the circuit stays in S_3 as long as there are three or more consecutive 1's received.

Synthesis Using DFF

- **Step 1:** Assign states using binary codes

  S_0= 00, S_1= 01, S_2= 10, S_3= 11

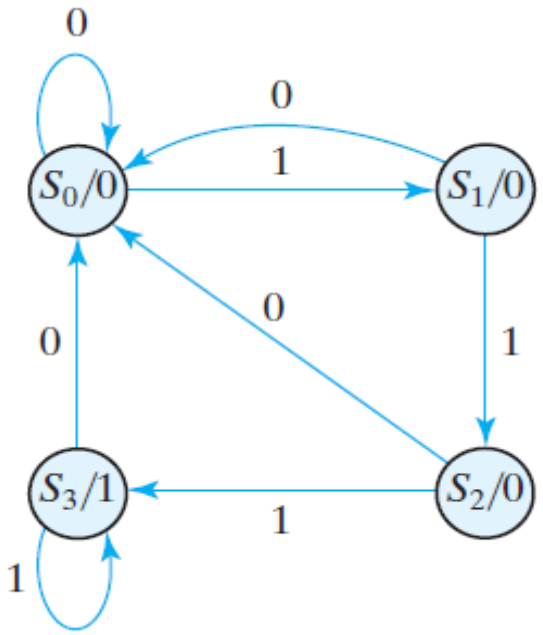


States	*A*	*B*
S_0	0	0
S_1	0	1
S_2	1	0
S_3	1	1

Present State	Next State $x = 0$	Next State $x = 1$	Output
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

Synthesis Using DFF

- **Step 2:** Make a binary-coded state table according to the state diagram



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

Synthesis Using DFF

- **Step 3**: Find minterms of next states $A(t+1)$, $B(t+1)$ and output y (next states $A(t+1)$, $B(t+1)$ are also DFF inputs).

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

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

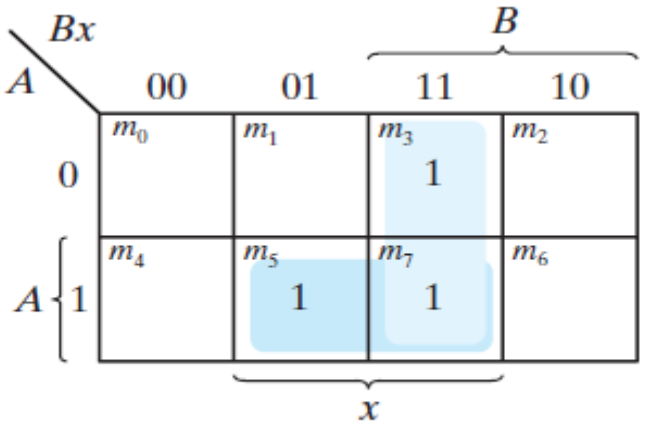
Synthesis Using DFF

- **Step 4**: Simplify the equations D_A, D_B and y, using K-map

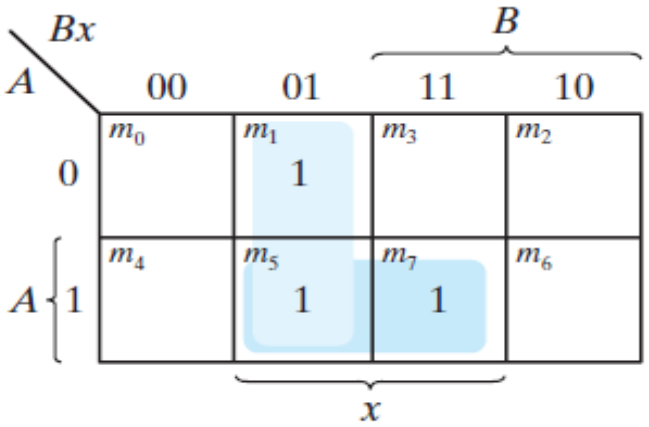
$$D_A(A,B,x) = \Sigma(3,5,7)$$
$$D_B(A,B,x) = \Sigma(1,5,7)$$
$$y(A,B,x) = \Sigma(6,7)$$

$$\Rightarrow$$

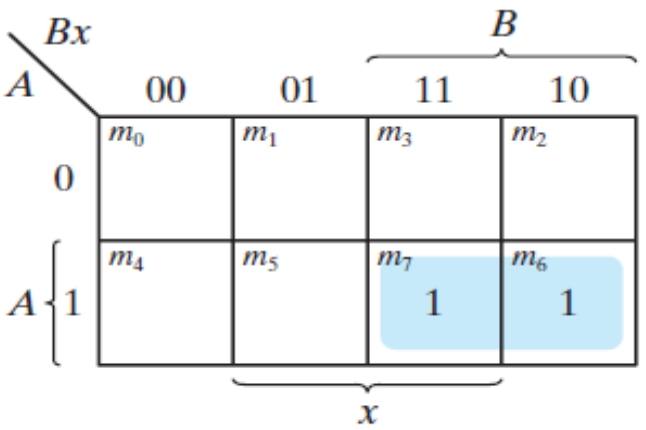
$$D_A = Ax + Bx$$
$$D_B = Ax + B'x$$
$$y = AB$$



$D_A = Ax + Bx$



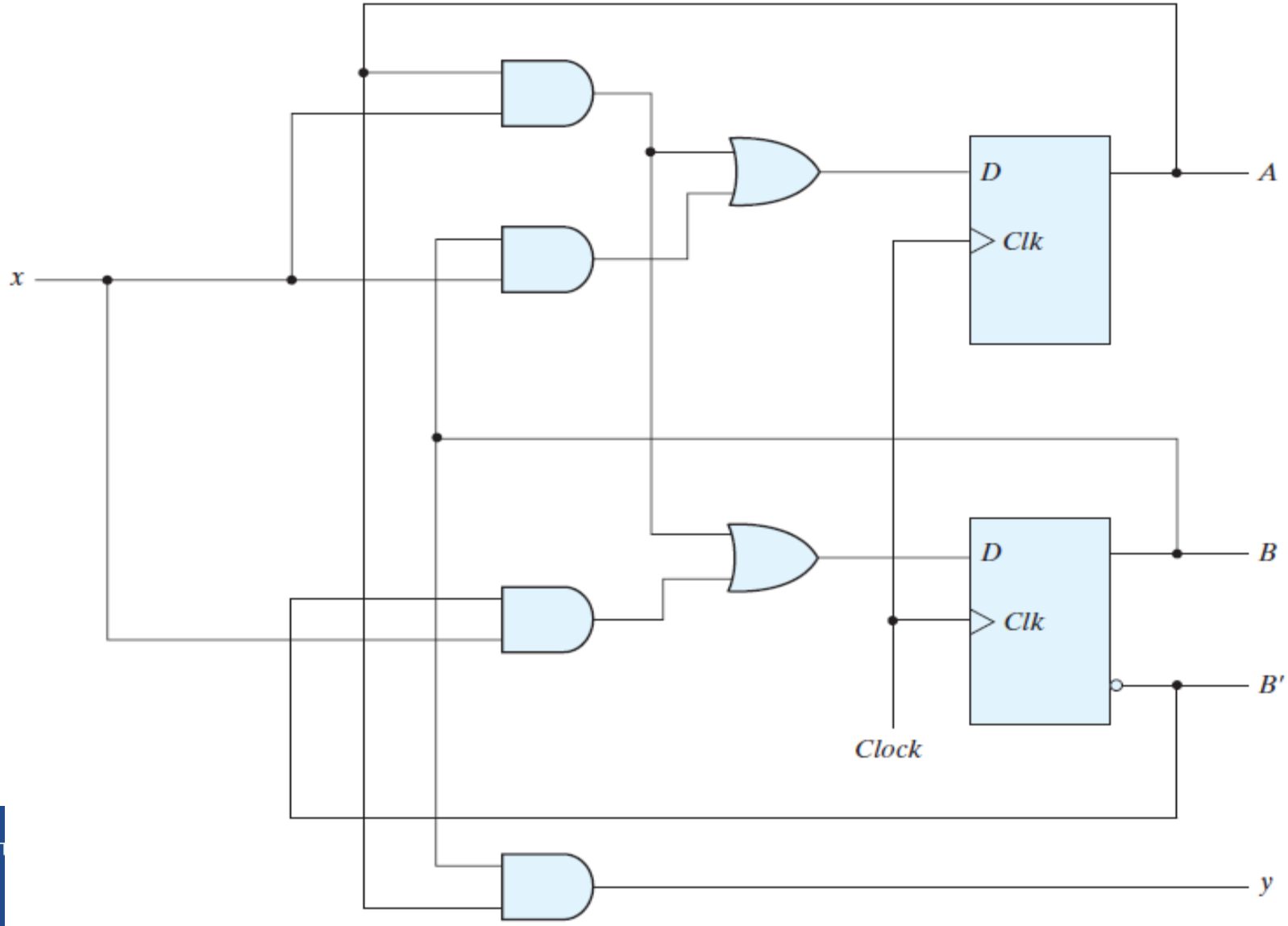
$D_B = Ax + B'x$



$y = AB$

Synthesis Using DFF

- **Step 5:** Draw the circuit from equations.



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

Excitation Tables

- The design of a sequential circuit with flip-flops other than the *D* type is complicated by the fact that the input equations for the circuit must be derived indirectly from the state table.
- During the design process, we usually know the transition from the present state to the next state and wish to find the flip-flop input conditions that will cause the required transition.
- For this reason, we need a table that lists the required inputs for a given change of state. Such a table is called an **excitation table** .

Table 5.12
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

(a) *JK* Flip-Flop

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

(b) *T* Flip-Flop

Synthesis Using JKFF

- Synthesize the sequential circuit specified by Table 5.13

Table 5.13
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

Flip-Flop Excitation Tables

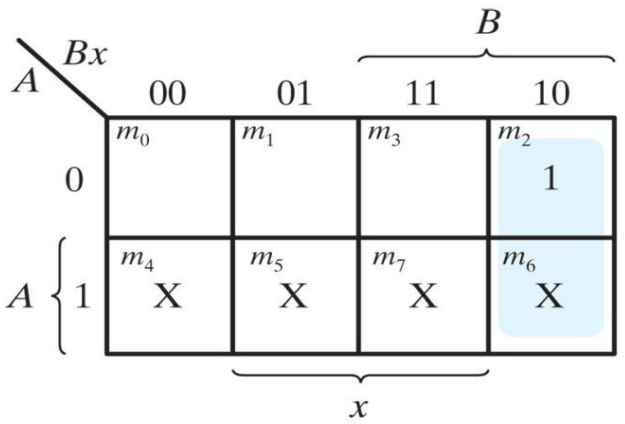
Q(t)	Q(t = 1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

(a) *JK* Flip-Flop

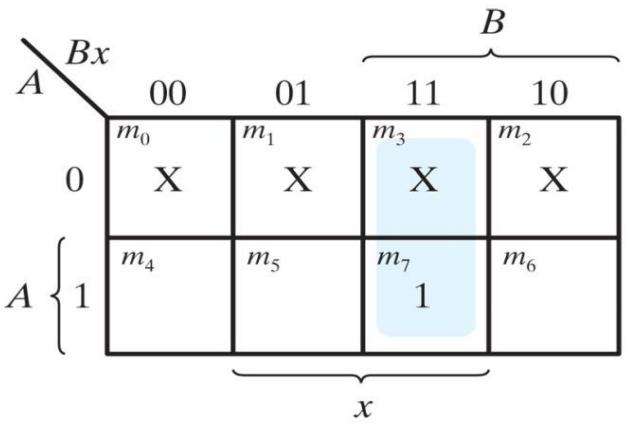
Synthesis Using JKFF

State Table and JK Flip-Flop Inputs

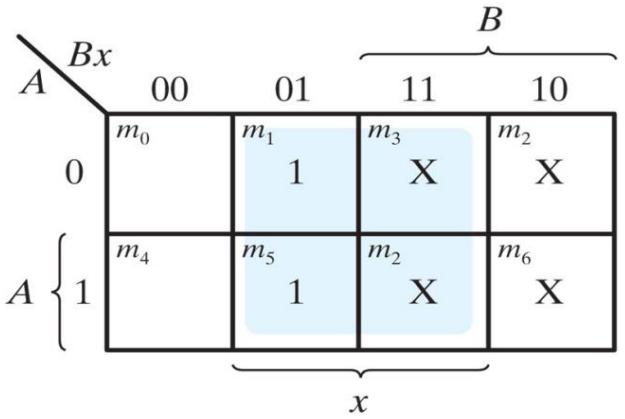
Present State		Input	Next State		Flip-Flop Inputs			
A	**B**	**x**	**A**	**B**	J_A	K_A	J_B	K_B
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1



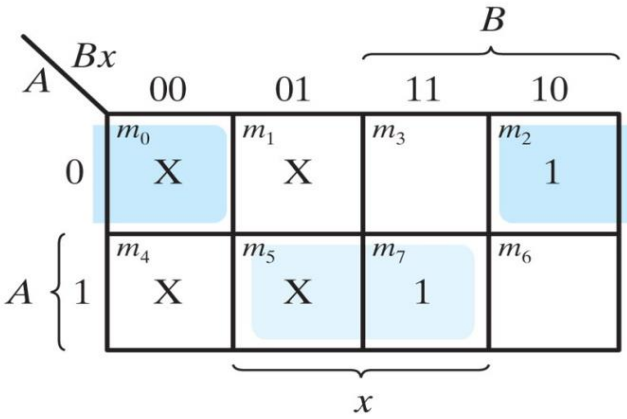
$J_A = Bx'$



$K_A = Bx$



$J_B = x$



$K_B = (A \oplus x)'$

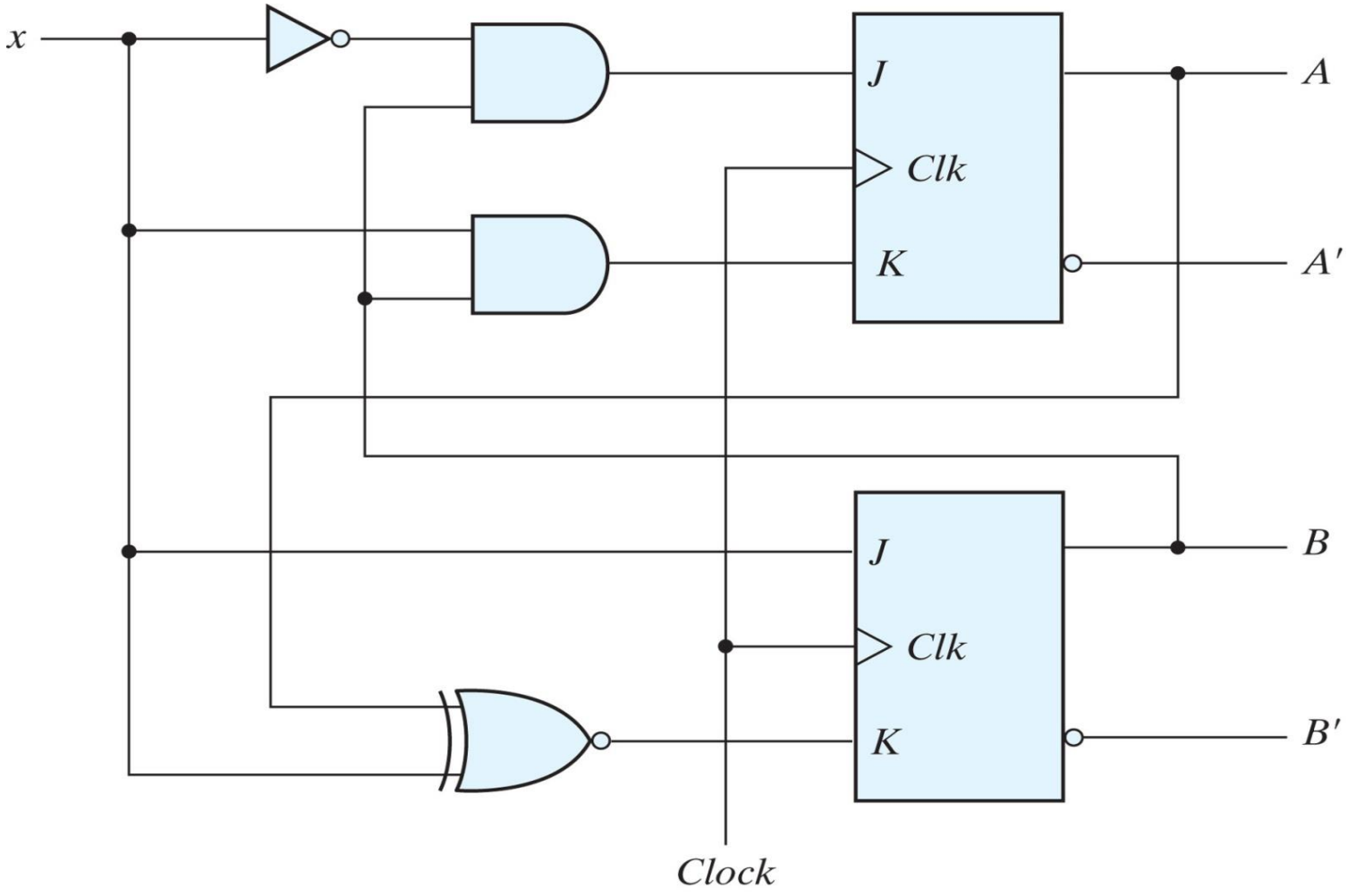
Synthesis Using JKFF

$$J_A = Bx'$$

$$K_A = Bx$$

$$J_B = x$$

$$K_B = (A \oplus x)'$$



Synthesis Using TFF

- Designing a binary counter.
- An *n* -bit binary counter consists of *n* flip-flops that can count in binary from 0 to 2^n - 1.

Q(t)	Q(t + 1)	T
0	0	0
0	1	1
1	0	1
1	1	0

(b) *T* Flip-Flop

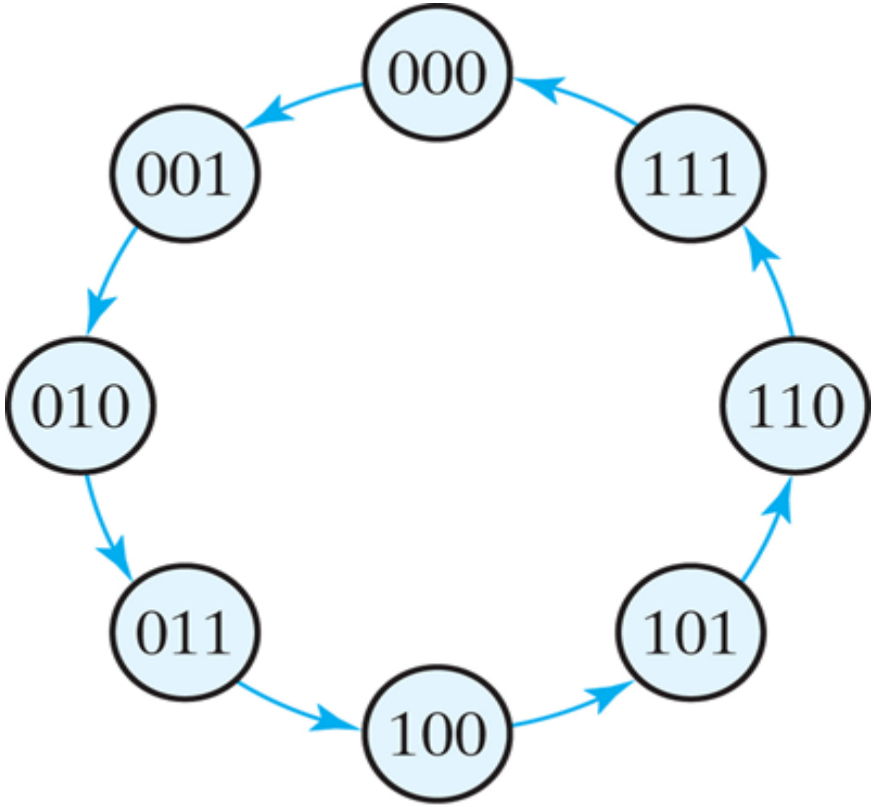


FIGURE 5.32
State diagram of three-bit binary counter

Synthesis Using TFF

Table 5.14
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	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	1	1
1	1	1	0	0	0	1	1	1

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

Synthesis Using TFF

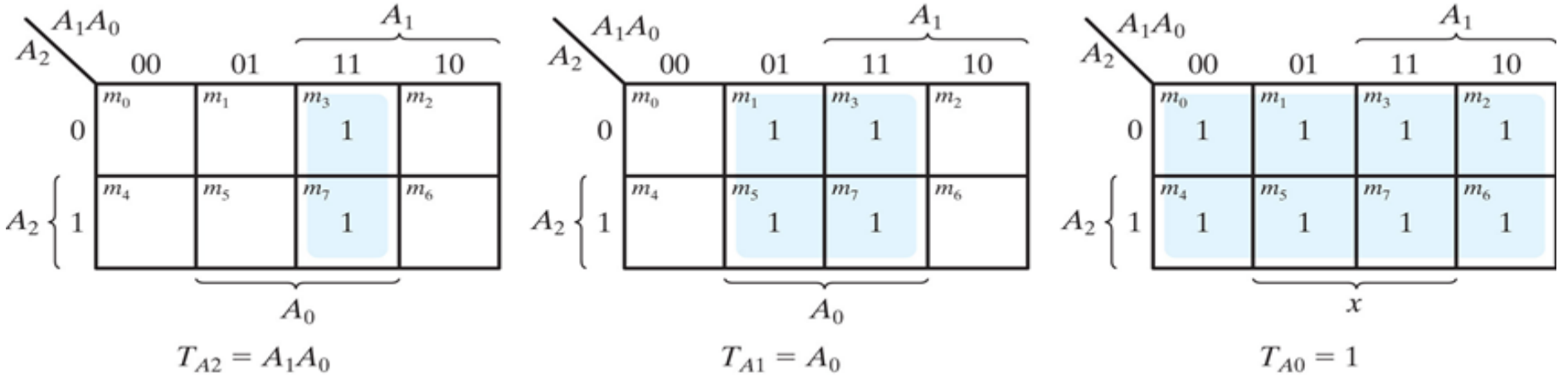


FIGURE 5.33
Maps for three-bit binary counter

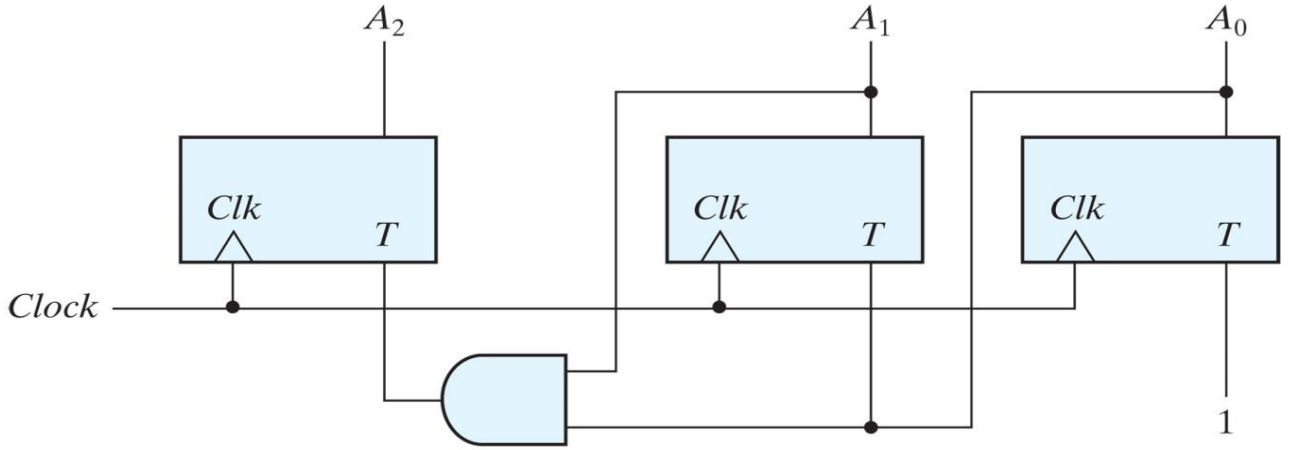


FIGURE 5.34
Logic diagram of three-bit binary counter

The End

Reference:

1. **Digital Design (with an introduction to the Verilog HDL) 6th Edition, M. Morris Mano, Michael D. Ciletti**