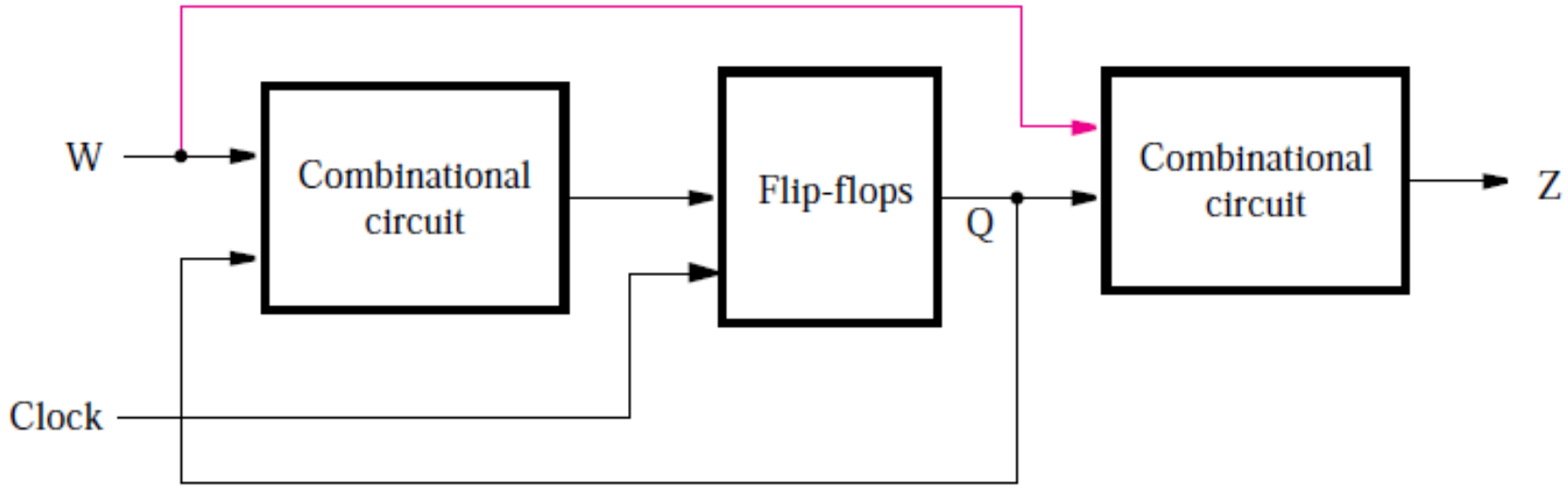
Synchronous Sequential Circuits

- Part I: Analysis
  - State Table
  - State Diagram
  - Finite State Machine (FSM)
  - Mealy & Moore State Model
- Part 2: Synthesis (Design)
  - General Procedure
  - Write a State Diagram, State Assignment
  - State Table
  - State Equations
  - Choice of Flip-Flop

General Form

- Combinational circuit for both input and output.
- Flip-flops for “memory” function
- Clock signal for “synchronization”



Analysis of Sequential Circuits

- Analysis is describing what a given circuit will do.
- The behavior of a clocked (synchronous) sequential circuit is determined from the inputs, the output, and the states of FF

❖Steps:

- Obtain state equations
  - FF input equations
  - Output equations
- Fill the state table
  - Put all combinations of inputs and current states
  - Fill the next state and output
- Draw the state diagram

Analysis of Combinational vs Sequential Circuits

Combinational :	**Sequential :**
➢Boolean Equations	✓State Equations
➢Truth Table	✓State Table
	✓State Diagram
➢Output as a function of inputs	✓Output as a function of input and current state
	✓Next state as a function of inputs and current state.

State Equations

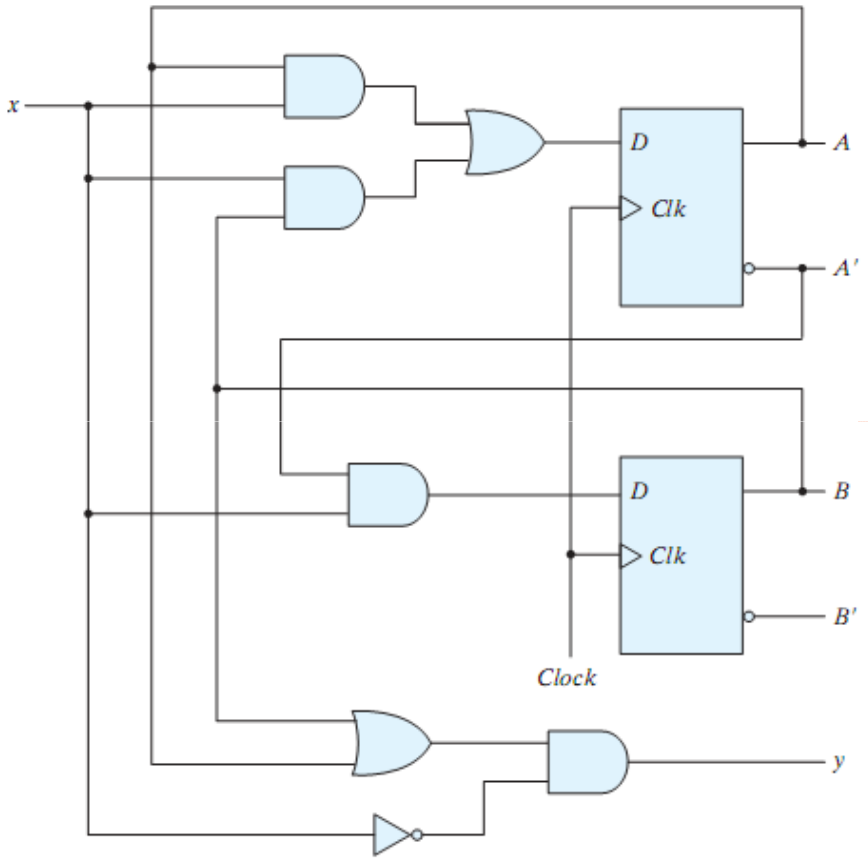
- A **state equation** is a Boolean expression which specifies the next state and output as a function of the present state and inputs.

- **Example:**
  - The shown circuit has two D-FFs (A,B), an input x and output y.
  - The D input of a FF determines the next state

    A(t+1) = A(t)x+B(t)x = Ax+Bx

    B(t+1) = A'(t)x = A'x
  - Output:

    y = (A+B)x'

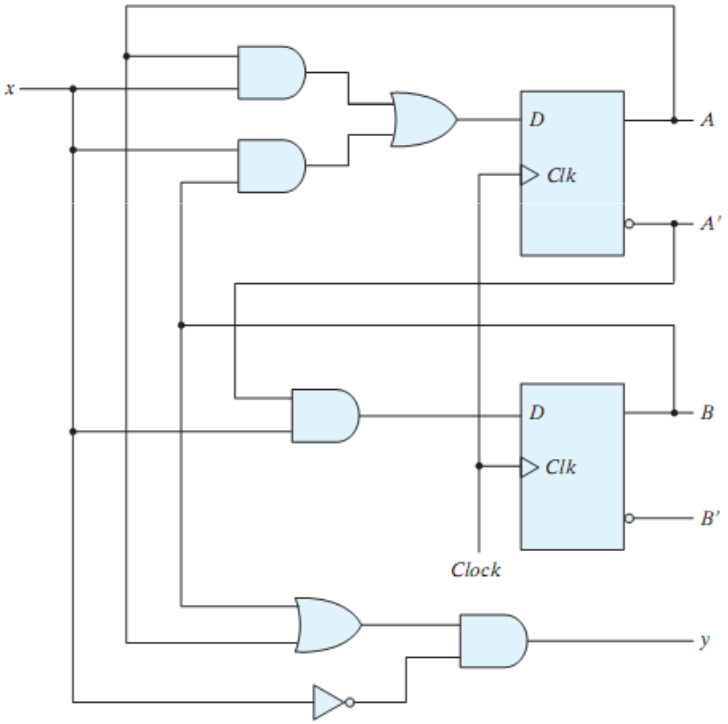


State Table

- A **state table** is a table enumerating all present states, inputs, next states and outputs.
  - Present state, inputs: list all combinations
  - Next states, outputs: derived from state equations

4 sections

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	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

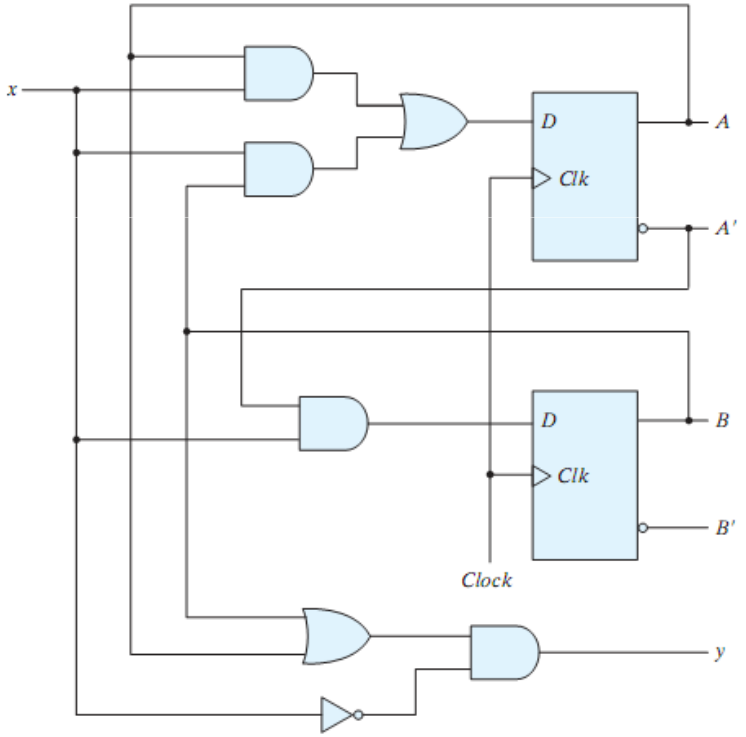


State Table

- A **state table** is a table enumerating all present states, inputs, next states and outputs.
  - Present state, inputs: list all combinations
  - Next states, outputs: derived from state equations

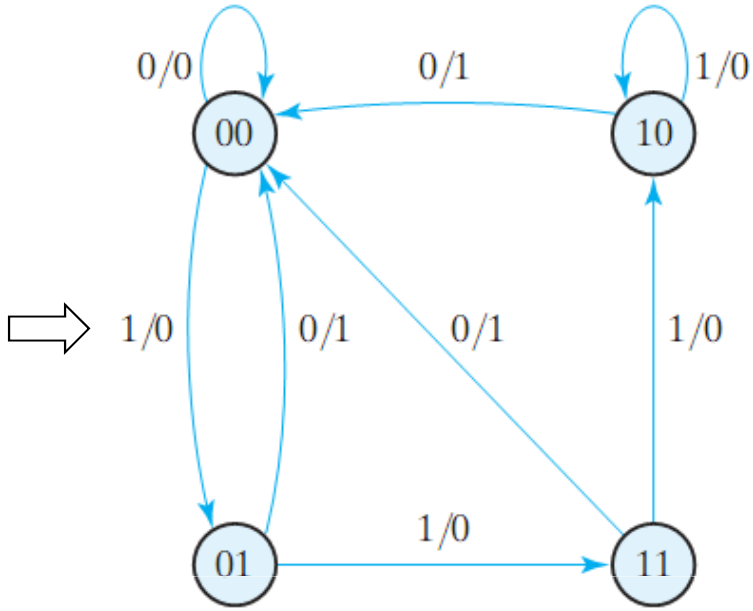
2-D Form

Present State		Next State				Output	
		$x = 0$		$x = 1$		$x = 0$	$x = 1$
A	B	A	B	A	B	y	y
0	0	0	0	0	1	0	0
0	1	0	0	1	1	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0



State Diagram

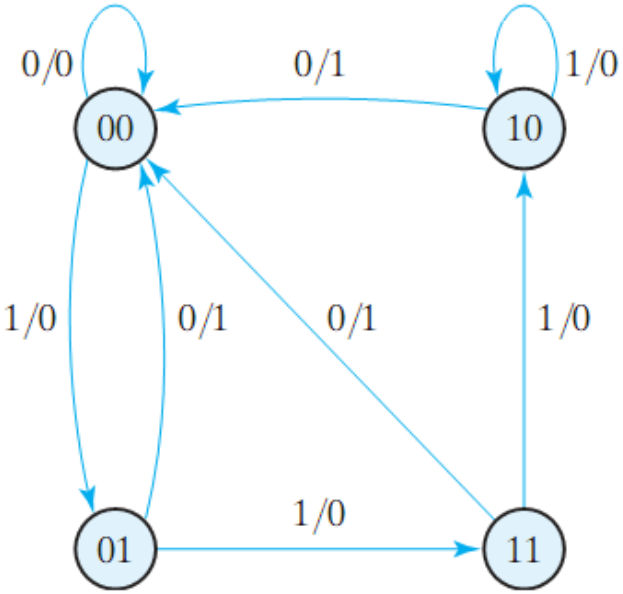
Present State		Input	Next State		Output
A	B	X	A	B	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0



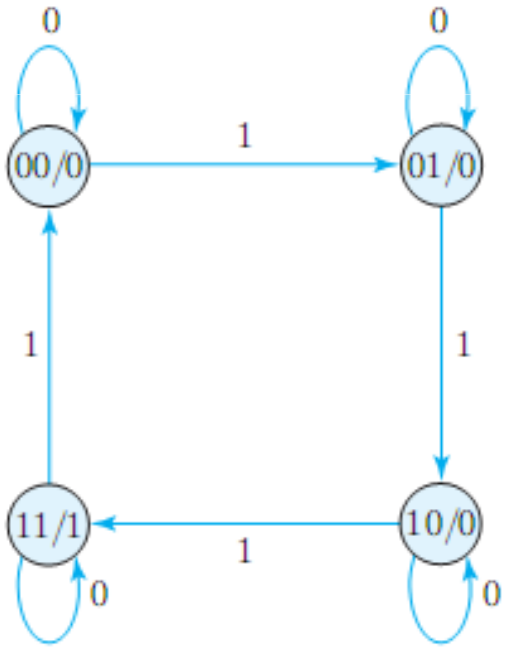
- The **state diagram** is a graphical representation of a state table (provides same information)
- Circles are states (FFs), Arrows are transitions between states
- Labels of arrows represent inputs and outputs
- State diagrams are used to represent "**finite state machine (FSM)**", which is a mathematical model of computation used to design both sequential circuits and computer programs.
- As the name implies, FSM consists of **finite** number of states.

Mealy vs Moore Finite State Machine (FSM)

- **Mealy FSM:**
  - Output depends on current state and input
  - Output is not synchronized with the clock



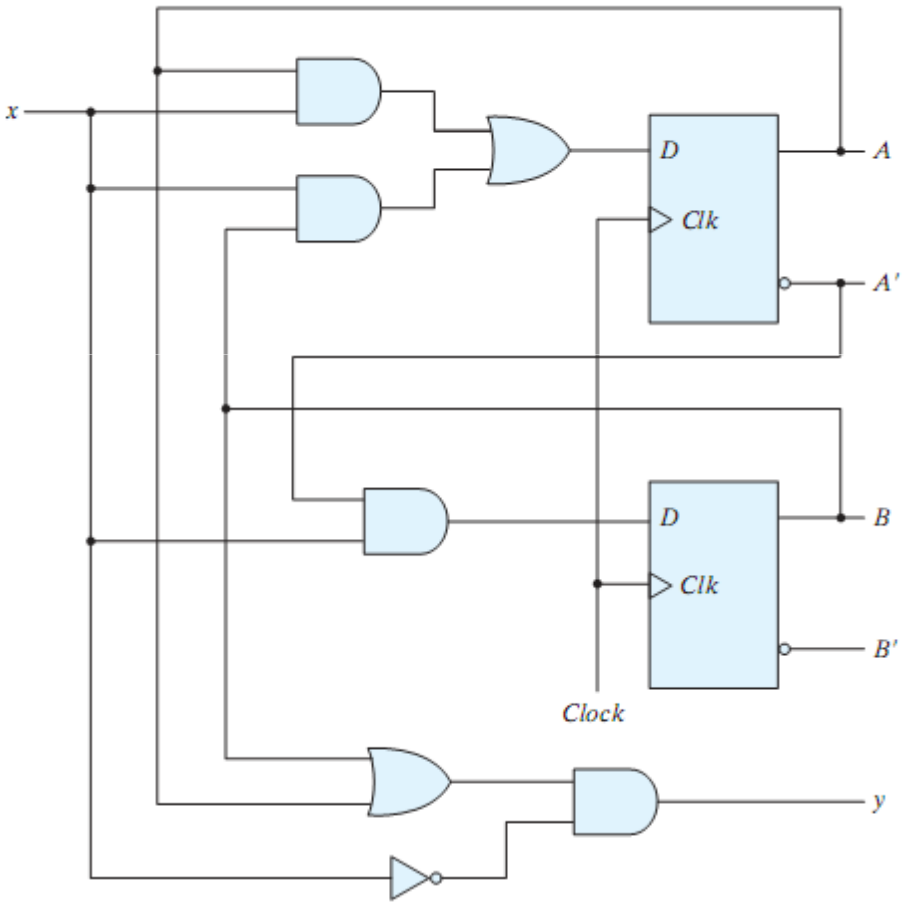
- **Moore FSM:**
  - Output depends on current state only



EXAMPLE 1

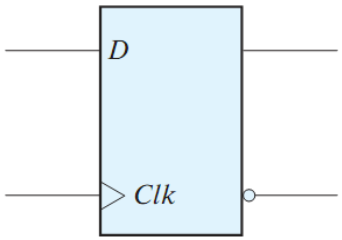
❖Analyze this circuit?

- Is this a sequential circuit? Why?
- How many inputs?
- How many outputs?
- How many states?
- What type of memory?

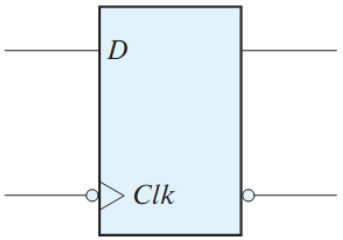


EXAMPLE 1 (CONT.)

D Flip Flop (review)



(a) Positive-edge



(a) Negative-edge

Characteristic Tables and Equations

Q(t)	D	Q(t+1)
0	0	0
0	1	1
1	0	0
1	1	1

D	Q(t+1)
0	0
1	1

Q(t+1) = D

EXAMPLE 1 (CONT.)

- **State equations:**

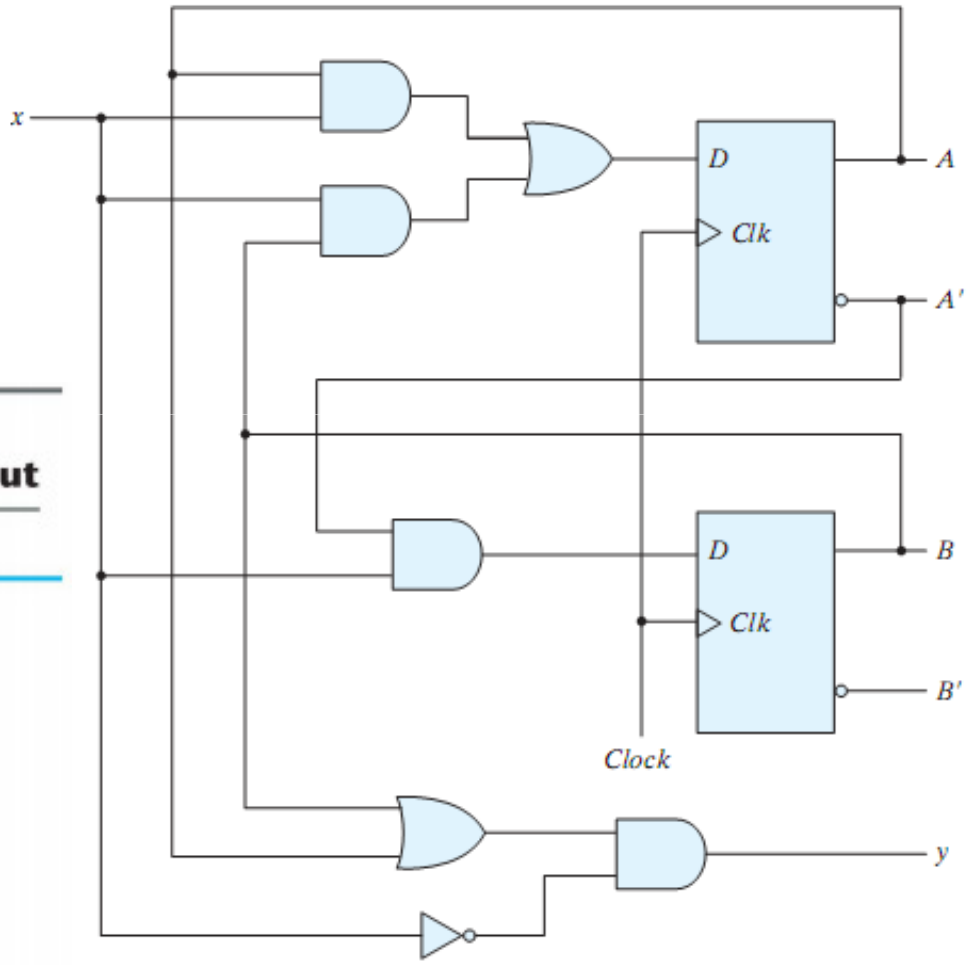
  $D_A = AX + BX$

  $D_B = A' X$

  $Y = (A + B) X'$

- **State table:**

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	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



EXAMPLE 1 (CONT.)

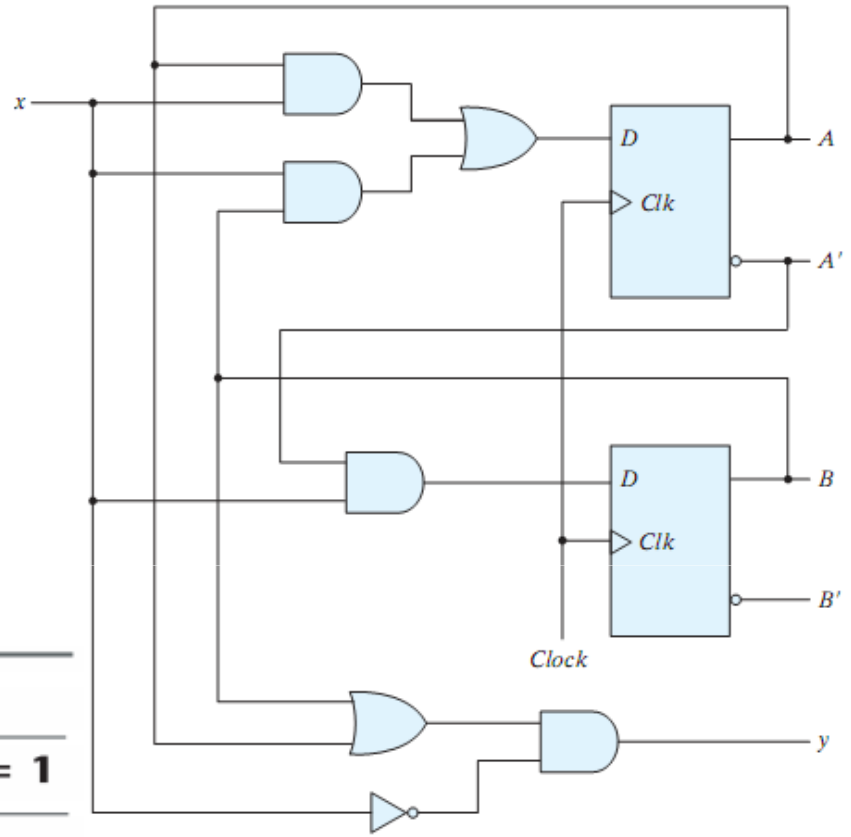
- **State equations:**

  $D_A = AX + BX$

  $D_B = A'X$

  $Y = (A + B)X'$

- **State table (2D):**



Present State		Next State				Output	
		$x = 0$		$x = 1$		$x = 0$	$x = 1$
A	B	A	B	A	B	y	y
0	0	0	0	0	1	0	0
0	1	0	0	1	1	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0

EXAMPLE 1 (CONT.)

- **State equations:**

  $D_A = AX + BX$

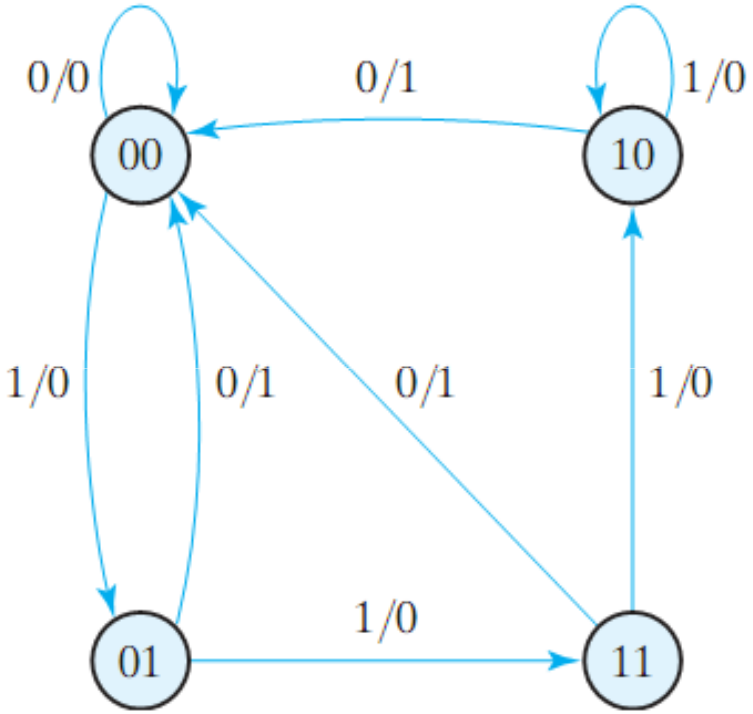
  $D_B = A'X$

  $Y = (A + B)X'$

- **State table:**

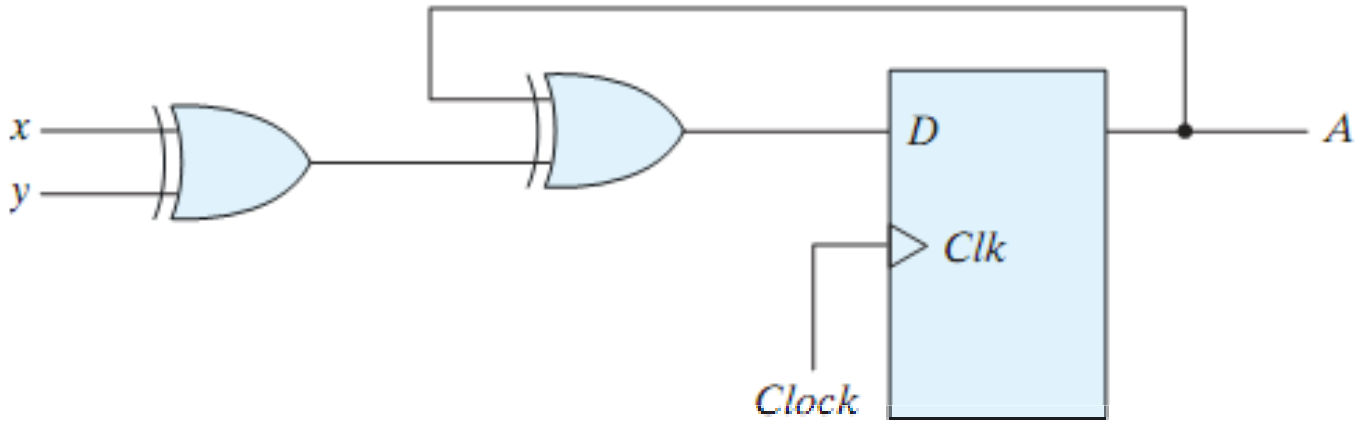
Present State		Input	Next State		Output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

State diagram:



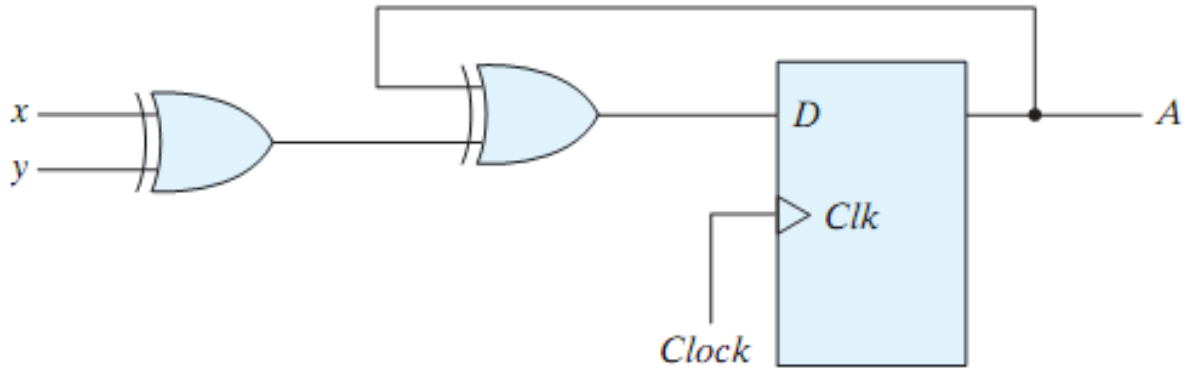
Mealy-type FSM

EXAMPLE 2



- Analyze this circuit.
- What about the output?
- This circuit is an example of a **Moore machine** (output depends **only** on current state)
- **Mealy machines** is the other type (output depends on **both** inputs and current states)

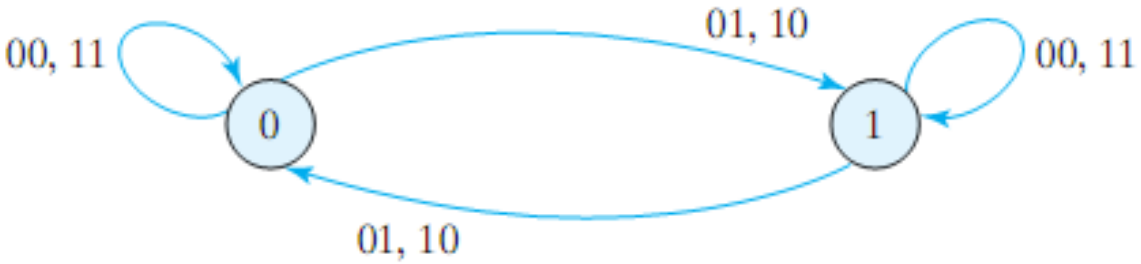
EXAMPLE 2 (CONT.)



Present state	Inputs		Next state
A	*x*	*y*	*A*
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:

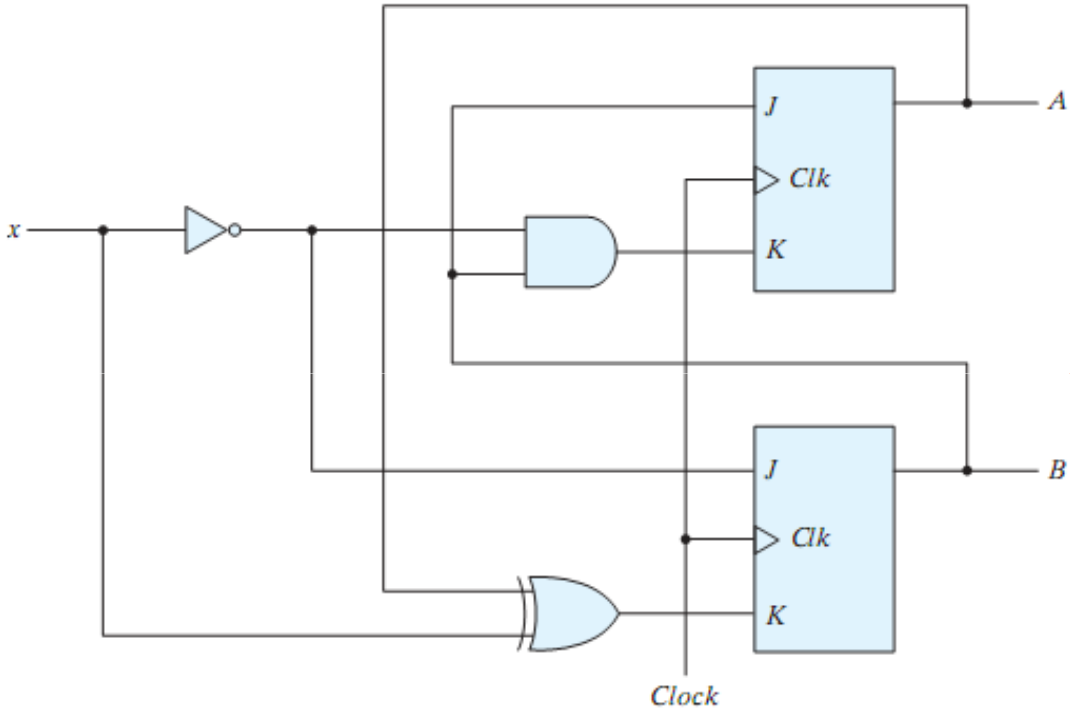
$D_A = A \oplus X \oplus Y$



EXAMPLE 3

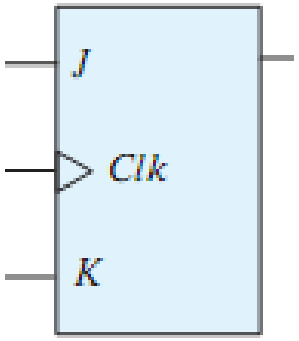
❖Analyze this circuit?

- Is this a sequential circuit? Why?
- How many inputs?
- How many outputs?
- How many states?
- What type of memory?



EXAMPLE 3 (CONT.)

JK Flip Flop (review)

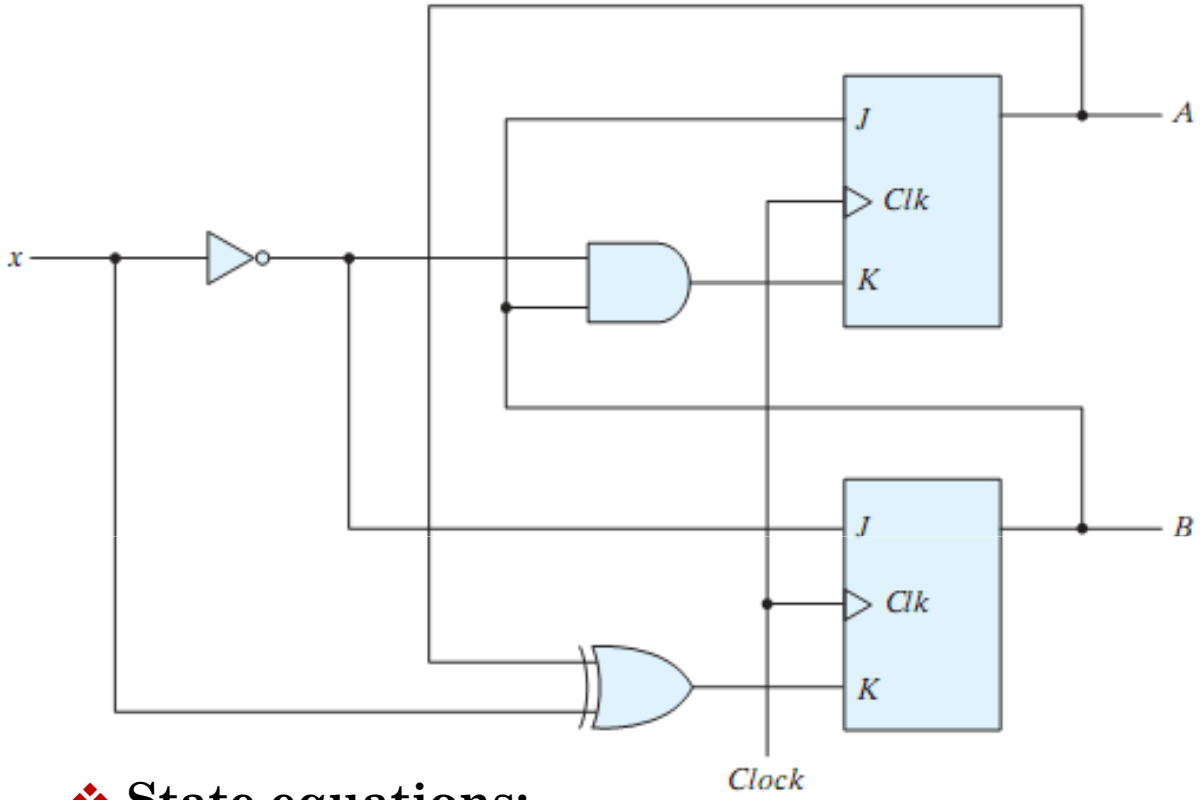


Characteristic Tables and Equations

J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	Q'(t)

$$Q(t+1)=Q^{+} = JQ' + K'Q$$

EXAMPLE 3 (CONT.)



- **State equations:**

  $J_A = B, K_A = B X'$

  $J_B = X', K_B = A \oplus X$

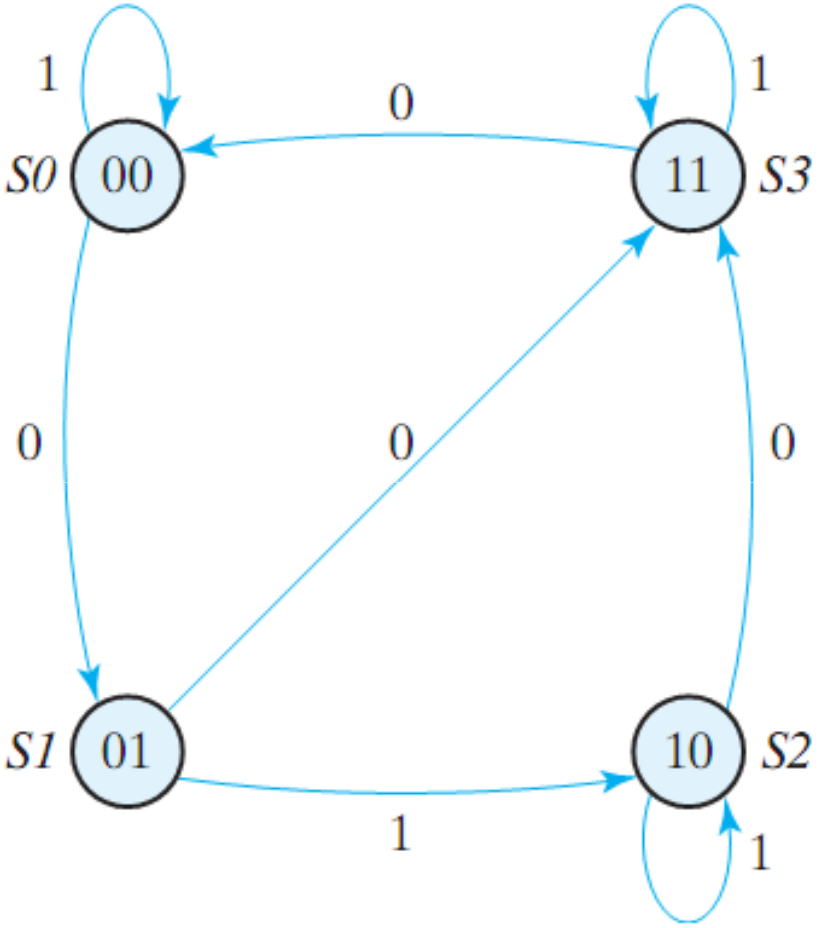
- **by substitution:**

  $A(t+1) = J_A A' + K_A' A = A' B + A B' + A X$

  $B(t+1) = J_B B' + K_B' B = B' X' + A B X + A' B X'$

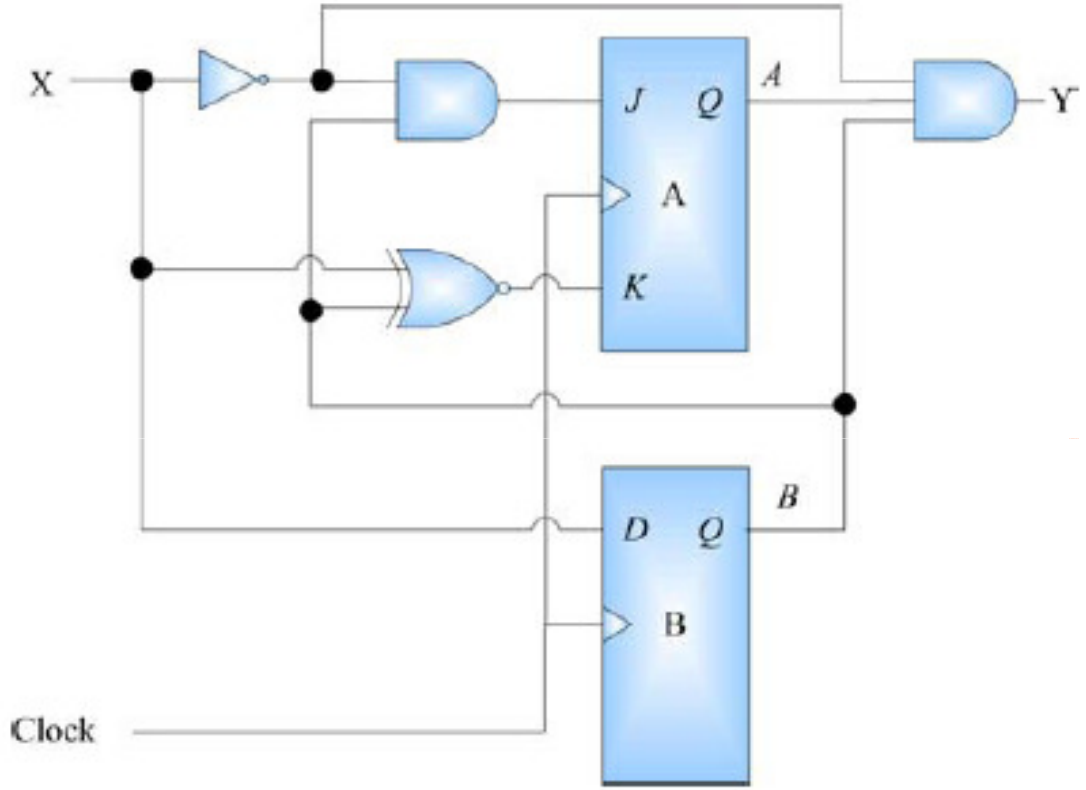
EXAMPLE 3 (CONT.)

Present State		Input	Next State	
A	B	x	A	B
0	0	0	0	1
0	0	1	0	0
0	1	0	1	1
0	1	1	1	0
1	0	0	1	1
1	0	1	1	0
1	1	0	0	0
1	1	1	1	1



EXAMPLE 4

- Analyze this circuit?
- Is this a sequential circuit? Why?
- How many inputs?
- How many outputs?
- How many states?
- What type of memory?



EXAMPLE 4 (CONT.)

State equations:

$J_A = BX'$

$K_A = BX' + B'X$

$D_B = X$

$Y = X'AB$

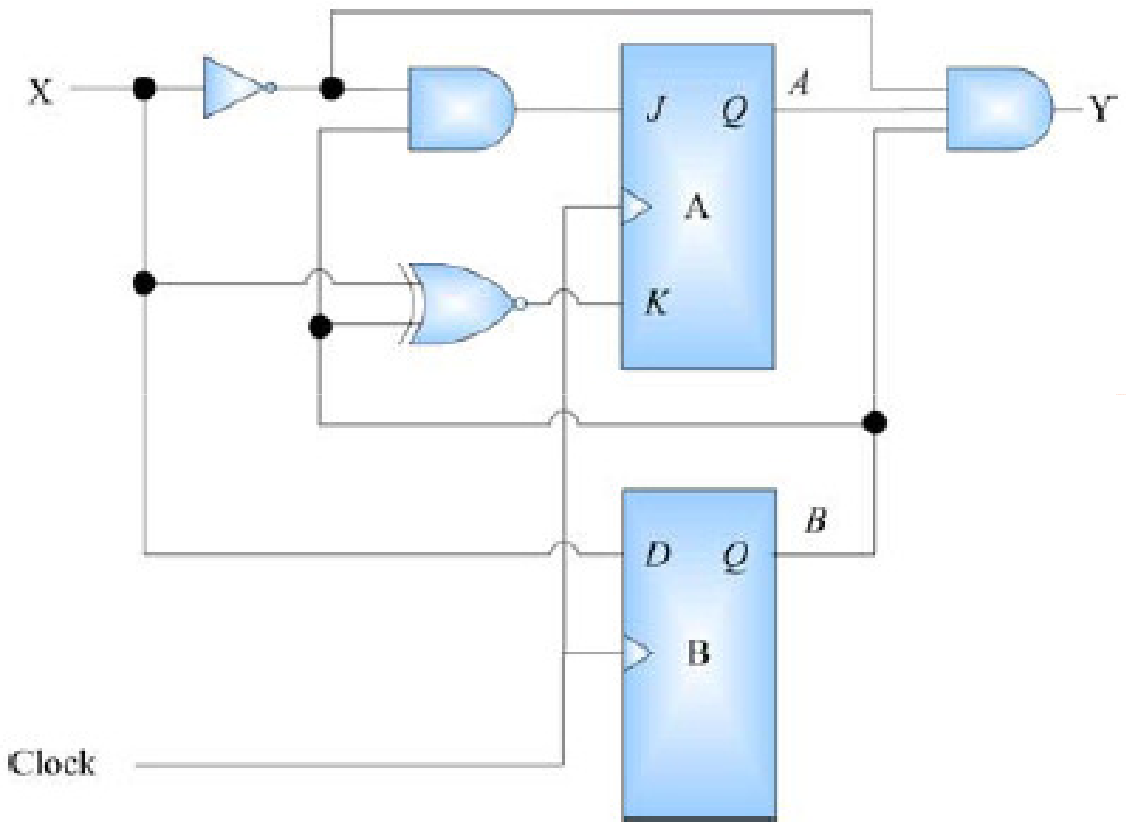
by substitution:

$A(t+1) = J_A A' + K_A' A$

$= A'BX' + A(B'+X)(B+X')$

$= A'BX'+A(BX+B'X')$

$B(t+1)=X$



EXAMPLE 4 (CONT.)

State equations:

$J_A = BX'$

$K_A = BX' + B'X$

$D_B = X$

$Y = X'AB$

by substitution:

$A(t+1) = J_A A' + K_A' A$

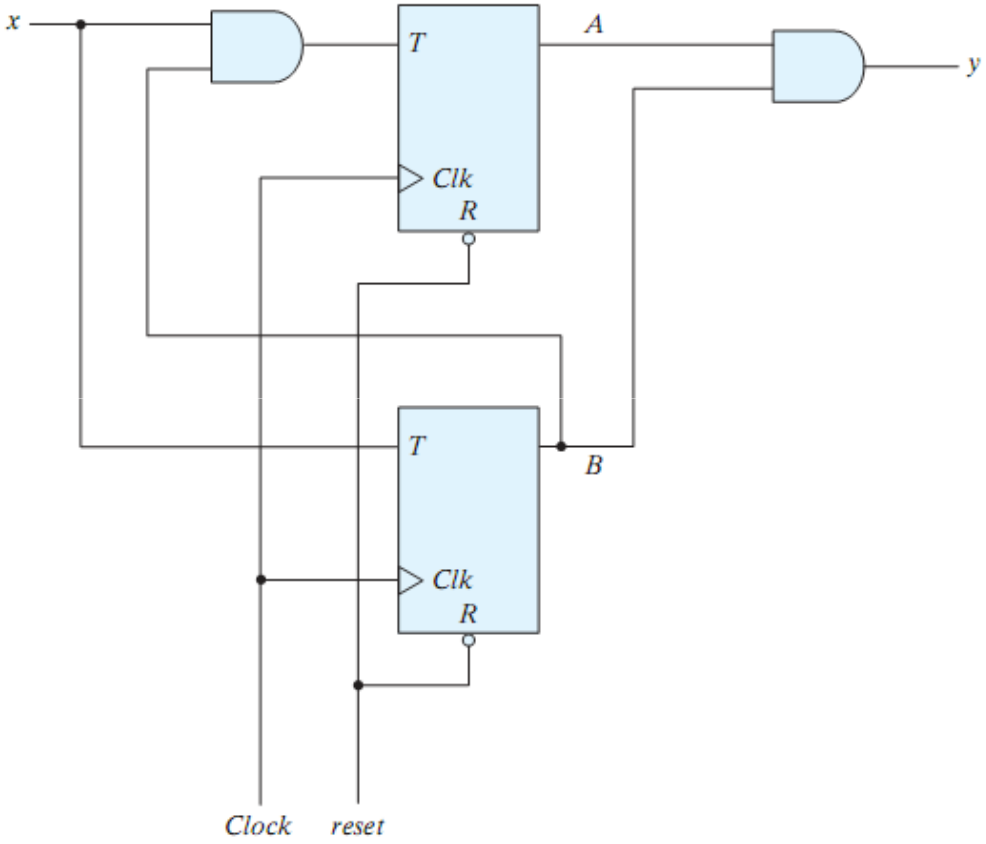
$= A'BX'+A(BX+B'X')$

$B(t+1)=X$

Current State		*Input*	*Next State*		*Output*
***A*(*t*)**	***B*(*t*)**	***X***	***A*(*t*+1)**	***B*(*t*+1)**	***Y***
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	0	1	0
1	1	0	0	0	1
1	1	1	1	1	0

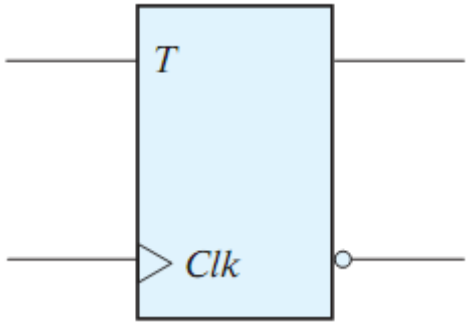
EXAMPLE 5

- Analyze this circuit?
- Is this a sequential circuit? Why?
- How many inputs?
- How many outputs?
- How many states?
- What type of memory?



EXAMPLE 5 (CONT.)

T Flip Flop (review)



Characteristic Tables and Equations

T	Q(t+1)
0	Q(t)
1	Q'(t)

$$Q(t+1) = TQ' + T'Q$$

EXAMPLE 5 (CONT.)

State equations:

$T_A = BX$

$T_B = X$

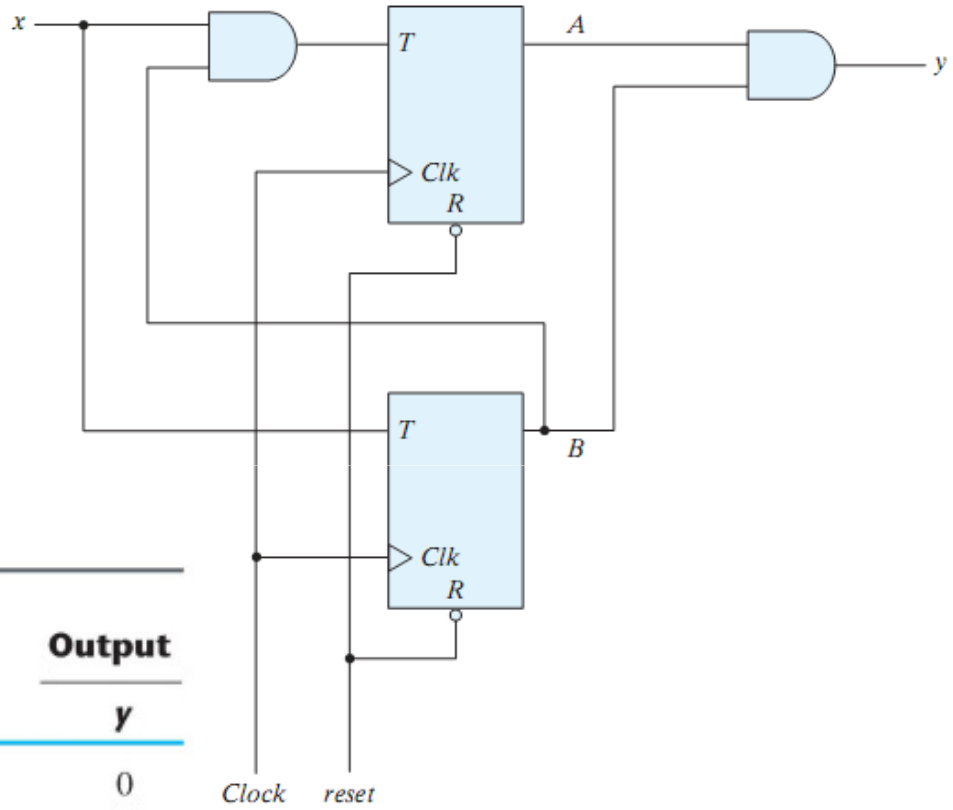
$Y = AB$

by substitution:

$A(t+1) = T_A A' + T_A' A$

$= A'BX + A(B'+X')$

$B(t+1) = B'X + BX'$



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	1	0
0	1	1	1	0	0
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	1

EXAMPLE 5 (CONT.)

State equations:

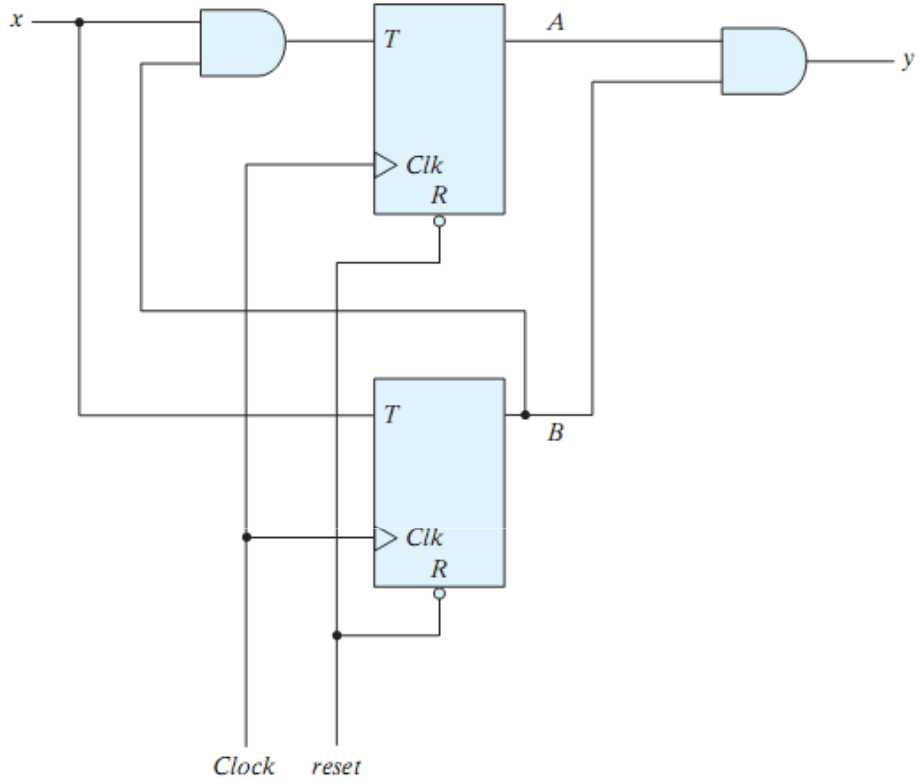
$T_A = BX$

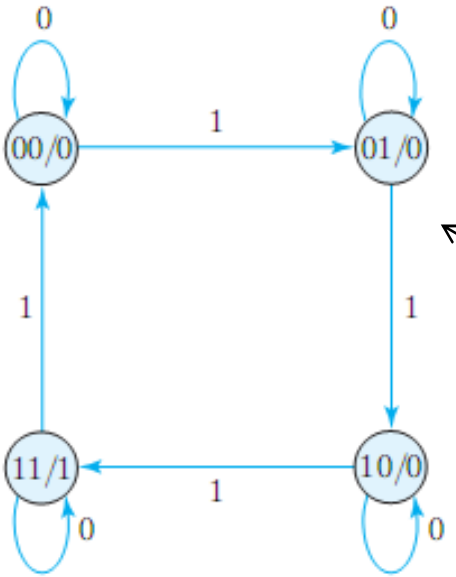
$T_B = X$

$Y = AB$

by substitution:

$A(t+1) = T_A A' + T_A' A$





The output depends only on current state.
This is a **Moore** machine

What does this circuit do?

Analysis Summary

- To analyze a sequential circuit:
  - Obtain state equations
    - FF input equations
    - Output equations
  - Fill the state table
    - Put all combinations of inputs and current states
    - Fill the next state and output
    - For the next state use characteristic table/equation
  - Draw the state diagram
- Two types of synchronous sequential circuits (Mealy and Moore)

Design (Synthesis) of Synchronous Sequential Circuits

- The **design** of a clocked sequential circuit starts from a set of specifications and ends with a logic diagram (Analysis reversed!)
- Building blocks: flip-flops, combinational logic
- Need to choose type and number of flip-flops
- Need to design combinational logic together with flip-flops to produce the required behavior
- The combinational part is
  - flip-flop input equations
  - output equations

Design of Synchronous Sequential Circuits

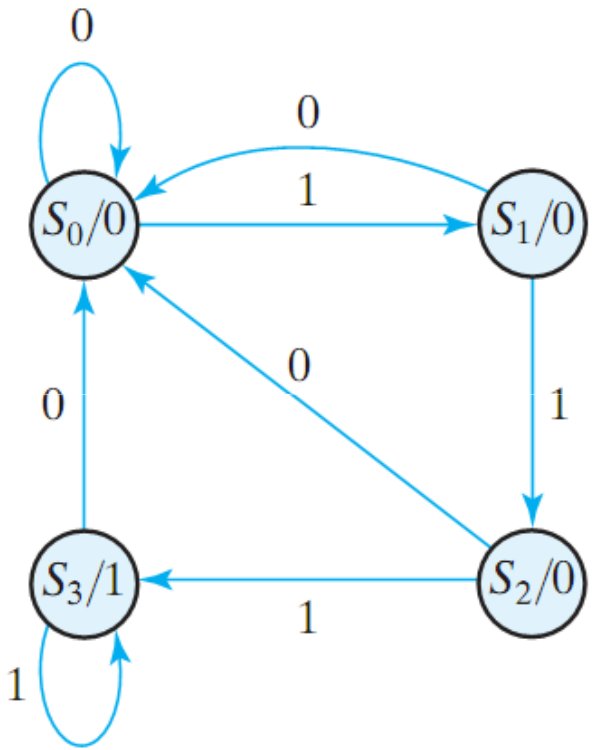
❖ Design Procedure:

- Obtain a state diagram from the specification
  - State reduction if necessary
- Obtain State Table
  - State Assignment
  - Choose type of flip-flops
  - Use FF's excitation table to complete the table
- Derive state equations
  - Obtain the FF input equations and the output equations
  - Use K-Maps to function simplification
- Draw the circuit diagram

STEP1: OBTAINING THE STATE DIAGRAM

- A very important step in the design procedure.
- Requires experience!

- **Example:** Design a circuit that detects a sequence of three consecutive 1's in a string of bits coming through an input line (serial bit stream)



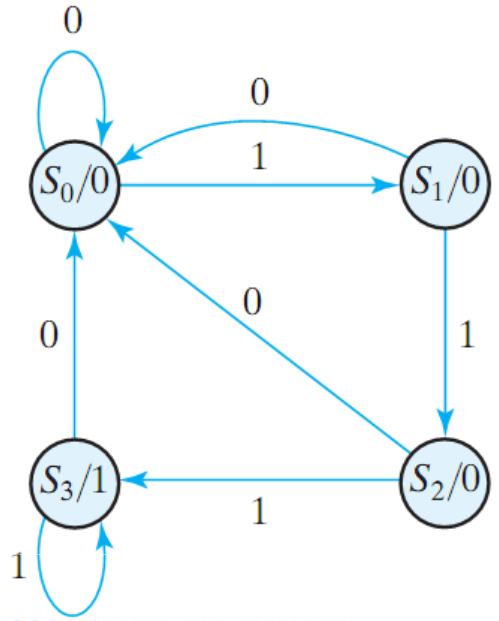
STEP2: OBTAINING THE STATE TABLE

- Assign binary codes for the states

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

- We choose 2 D-FF (simplest!)
  - Next state specifies what should be the input to each FF

- **Example:** Design a circuit that detects a sequence of three consecutive 1's in a string of bits coming through an input line (serial bit stream)



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

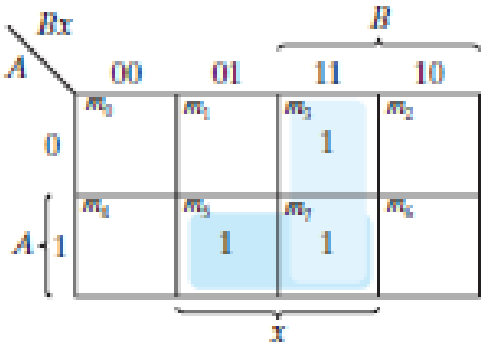
STEP3: OBTAINING THE STATE EQUATIONS

- Input Equations : $A(t+1) = D_A = \sum(3,5,7) = A\,x + B\,x$,

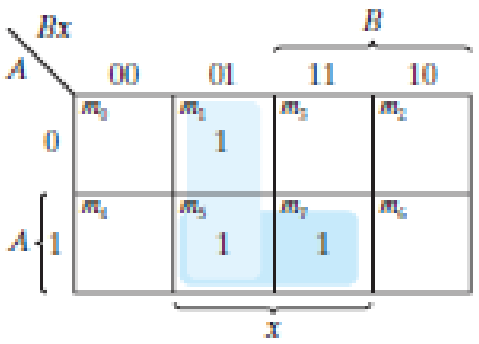
$B(t+1) = D_B = \sum(1,5,7) = A\,x + B'\,x$, $y = \sum(6,7) = A\,B$

State Table for Sequence Detector

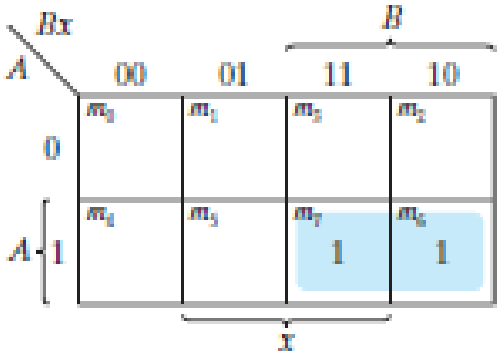
Present State		Input	Next State		Output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1



$D_A = A\,x + B\,x$

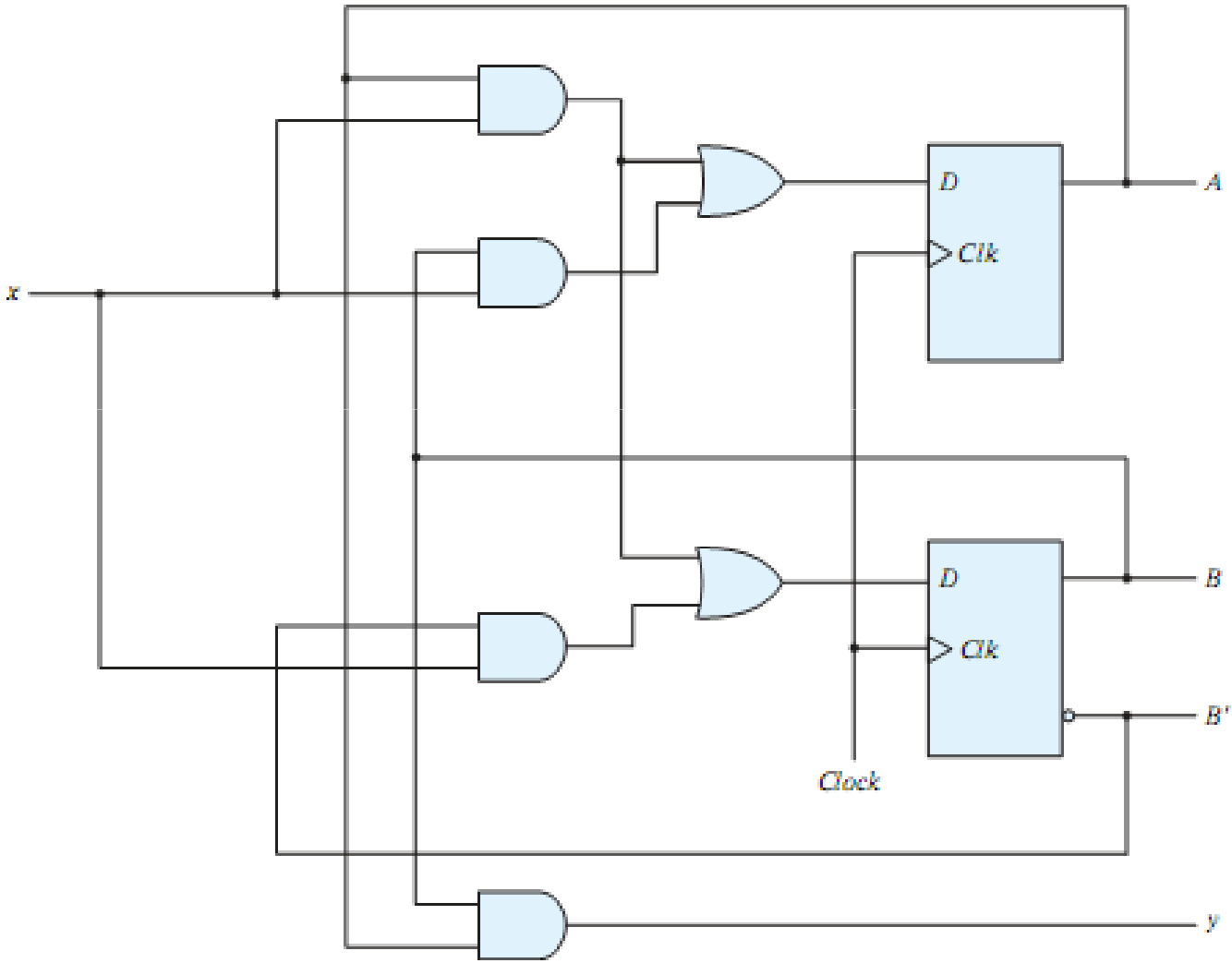


$D_B = A\,x + B'\,x$



$y = AB$

STEP4: DRAW CIRCUITS



DESIGN WITH OTHER TYPES OF FF

- In designing with D-FFs, the input equations are obtained from the next state (simple!)
- It is not the case when using JK-FF and T-FF !
- **Excitation Table:** Lists the required inputs that will cause certain transitions.
  - Characteristic tables used for analysis, while excitation tables used for design

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*

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

(b) *T*

$$Q(t+1)=JQ'+K'Q$$

$$Q(t+1)=TQ'+T'Q$$

State Table for JK Flip-Flop Case

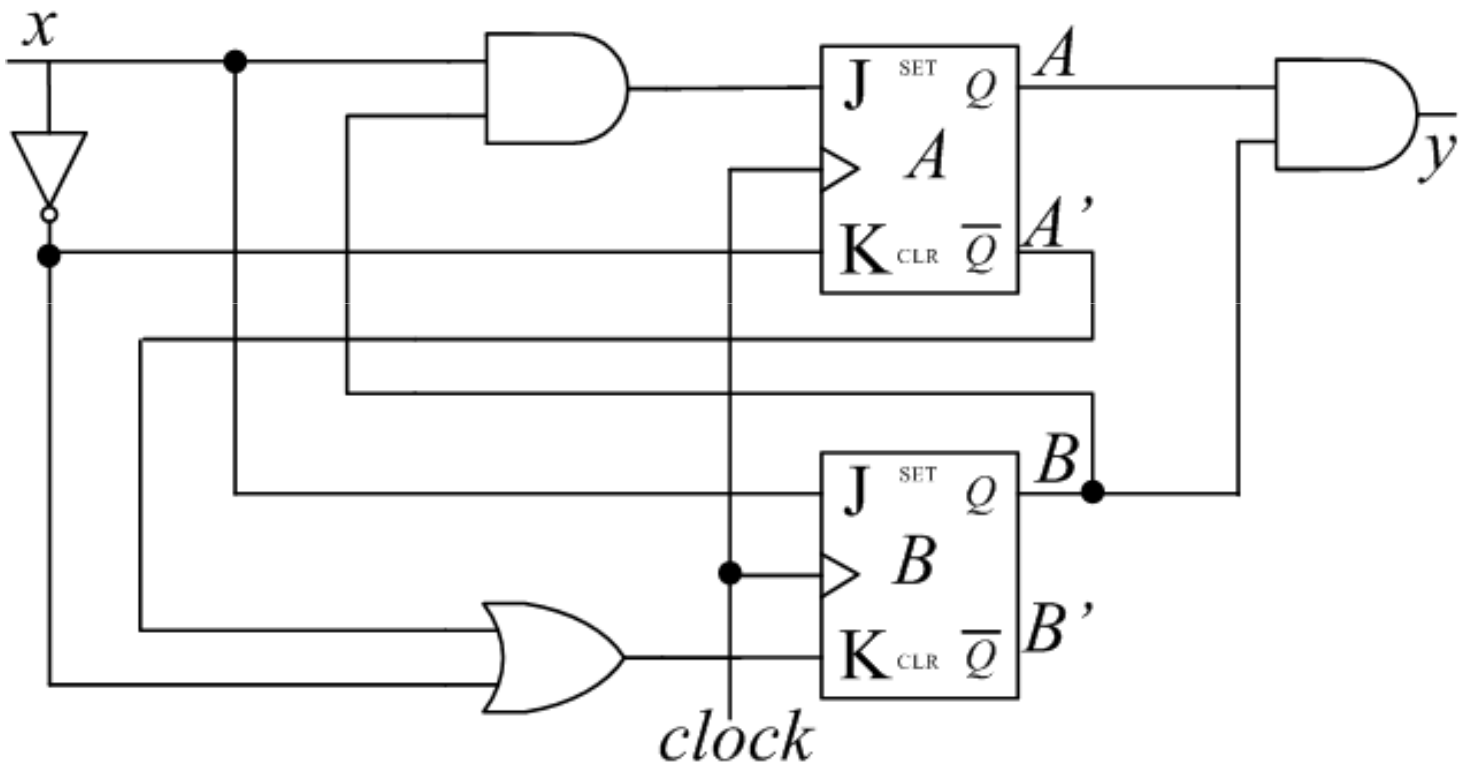
- Use the previous state assignment.

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

State Equations and Circuit

$$A(t+1) = J_A A' + K_A' A;$$

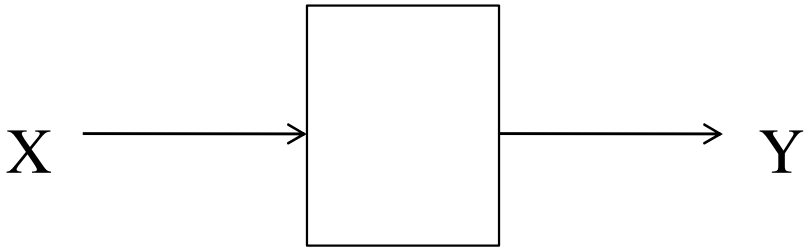
$$J_A = Bx, K_A = x', J_B = x, K_B = A' + x'; y = AB$$



Note: Fewer logic gates required.

EXAMPLE 1

- **Problem: Design of A Sequence Recognizer**
- Design a circuit that reads as inputs continuous bits, and generates an output of '1' if the sequence (1011) is detected

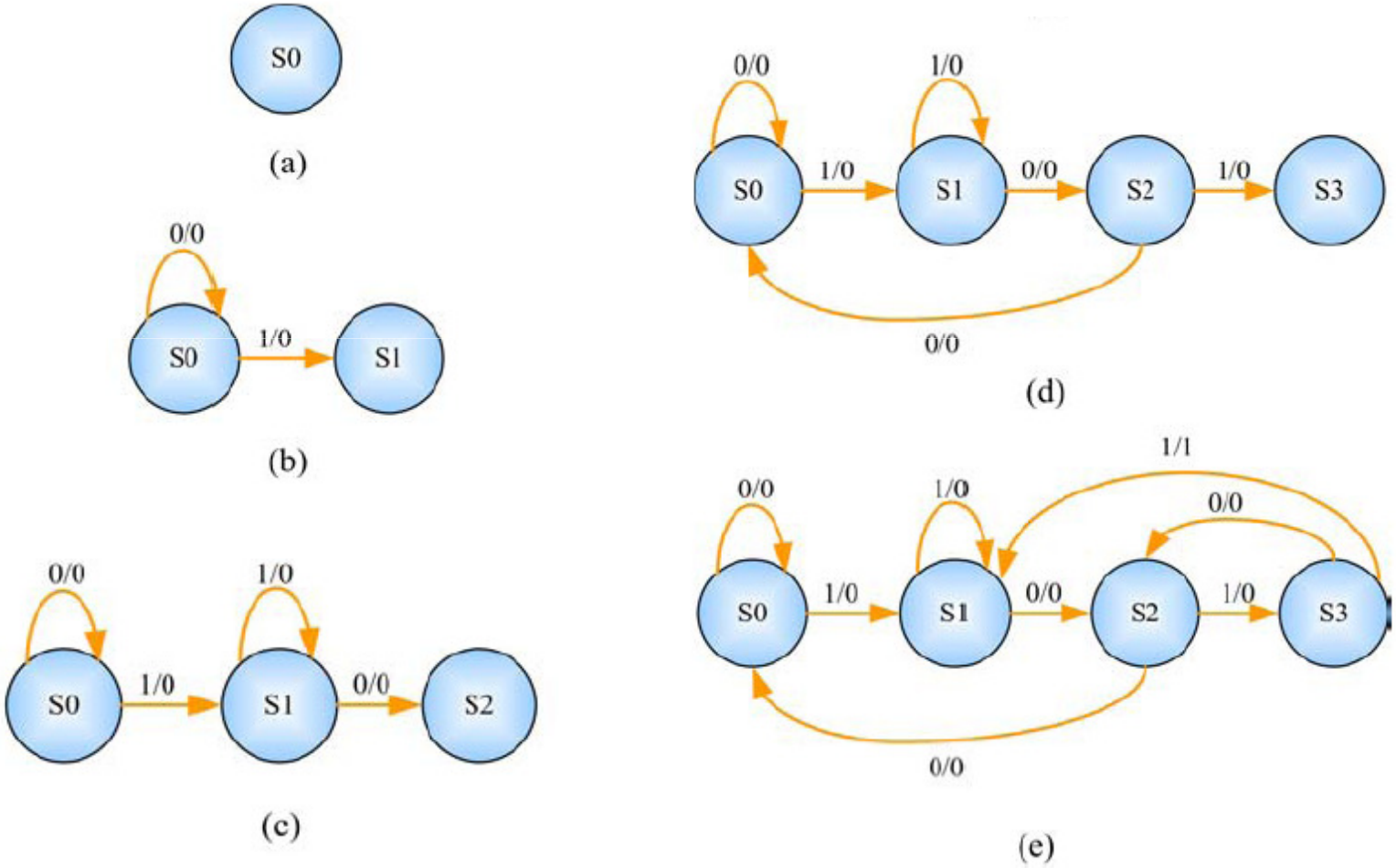


Input	1	1	1	0	0	1	0	0	1	0	0	1	1	0	1	1	0	1	0	1	1	0	1	1	1	1	1	1	1
Output	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	**1**	0	0	**1**	0	0	0	0

EXAMPLE 1 (CONT.)

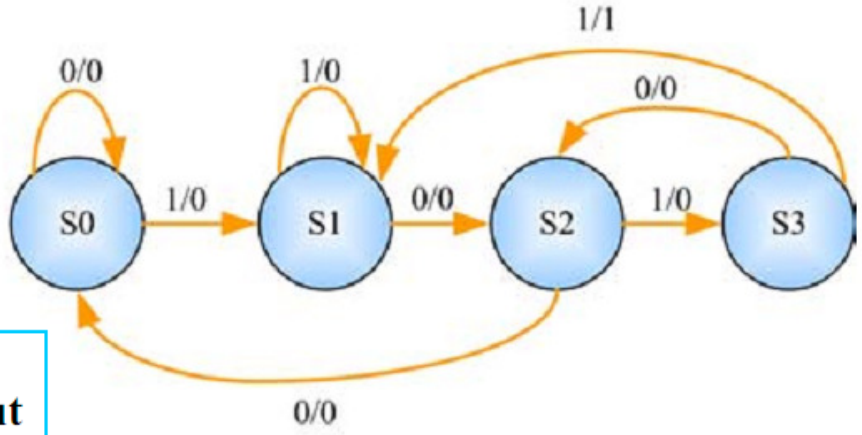
❖ **Step1: State Diagram**

Sequence to be detected:1011



EXAMPLE 1 (CONT.)

❖**Step 2: State Table**



Inputs of Combinational Circuit		Next State	Output
Present State	Input		
S0	0	*S0*	0
S0	1	*S1*	0
S1	0	*S2*	0
S1	1	*S1*	0
S2	0	*S0*	0
S2	1	*S3*	0
S3	0	*S2*	0
S3	1	*S1*	1

OR

Present State	Next State		Output	
	X=0	X=1	X=0	X=1
S0	*S0*	*S1*	0	0
S1	*S2*	*S1*	0	0
S2	*S0*	*S3*	0	0
S3	*S2*	*S1*	0	1

EXAMPLE 1 (CONT.)

- **Step 2: State Table**
- ***state assignment***

State	*Assignment*
S0	00
S1	01
S2	10
S3	11

Q: How many FF?

$\lceil \log_2(\text{no. of states}) \rceil$

Inputs of Combinational Circuit		Next State	Output
Present State	Input		
S0	0	*S0*	0
S0	1	*S1*	0
S1	0	*S2*	0
S1	1	*S1*	0
S2	0	*S0*	0
S2	1	*S3*	0
S3	0	*S2*	0
S3	1	*S1*	1

Inputs of Combinational Circuit		Next State	Output
Present State	Input		
A B	*X*	*A B*	*Y*
0 0	0	0 0	0
0 0	1	0 1	0
0 1	0	1 0	0
0 1	1	0 1	0
1 0	0	0 0	0
1 0	1	1 1	0
1 1	0	1 0	0
1 1	1	0 1	1

EXAMPLE 1 (CONT.)

- **Step 2: State Table**
- ***choose FF***
- **In this example, lets use JK–FF for A and D-FF for B**

Inputs of Combinational Circuit		Next State	Output
Present State	Input		
A B	*X*	*A B*	*Y*
0 0	0	0 0	0
0 0	1	0 1	0
0 1	0	1 0	0
0 1	1	0 1	0
1 0	0	0 0	0
1 0	1	1 1	0
1 1	0	1 0	0
1 1	1	0 1	1

EXAMPLE 1 (CONT.)

Present State		*Input*	*Next State*		*output*	*Flip-flops Inputs*		
A	*B*	*X*	*A*	*B*	*Y*	J_A	K_A	D_B
0	0	0	0	0	0	0	X	0
0	0	1	0	1	0	0	X	1
0	1	0	1	0	0	1	X	0
0	1	1	0	1	0	0	X	1
1	0	0	0	0	0	X	1	0
1	0	1	1	1	0	X	0	1
1	1	0	1	0	0	X	0	0
1	1	1	0	1	1	X	1	1

D–FF excitation table

Q(t +1)	D	Operation
0	0	Reset
1	1	Set

JK–FF excitation table

Q(t)	Q(t +1)	J	K	Operation
0	0	0	X	No change
0	1	1	X	Set
1	0	X	1	Reset
1	1	X	0	No Change

EXAMPLE 1 (CONT.)

❖ **Step 3: State Equations**

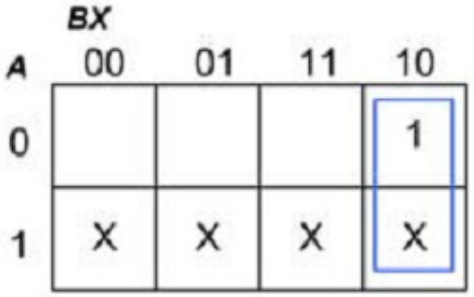
❖ ***use k-map***

$J_A = BX'$

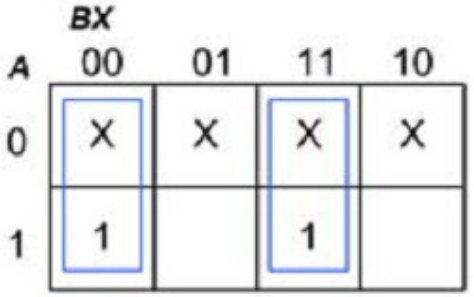
$K_A = BX + B'X'$

$D_B = X$

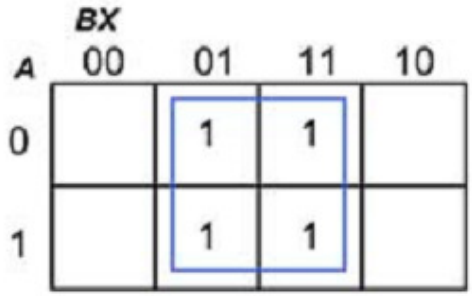
$Y = ABX'$



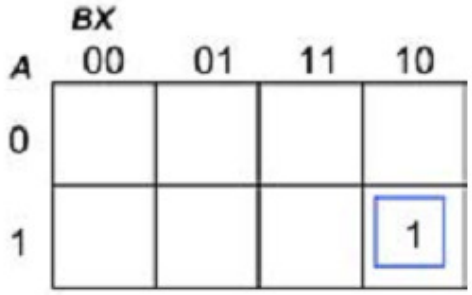
JA = BX'



KA = BX + B'X'



DB = X



Y = ABX'

EXAMPLE 1 (CONT.)

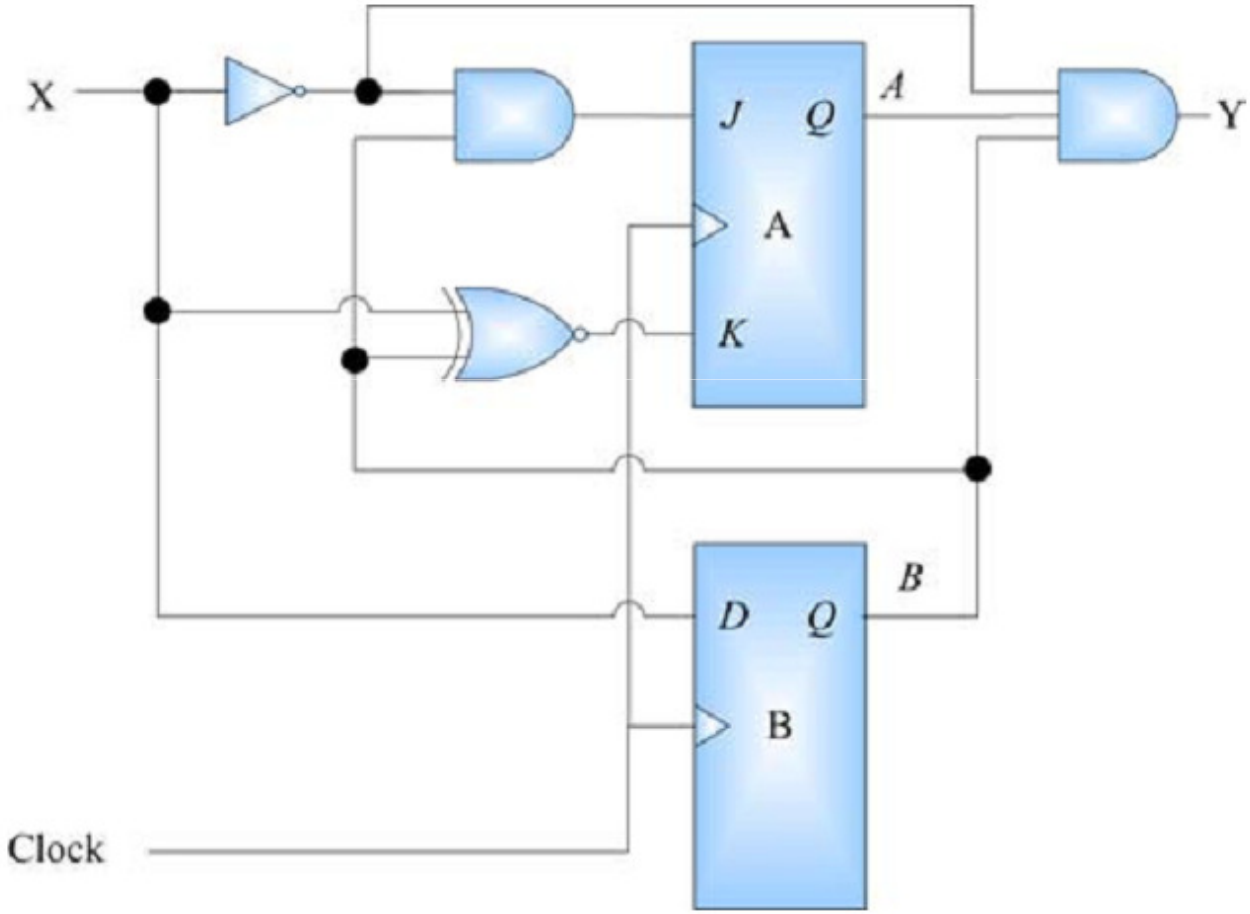
❖**Step 4: Draw Circuit**

$J_A = BX'$

$K_A = BX + B'X'$

$D_B = X$

$Y = ABX'$



EXAMPLE 2

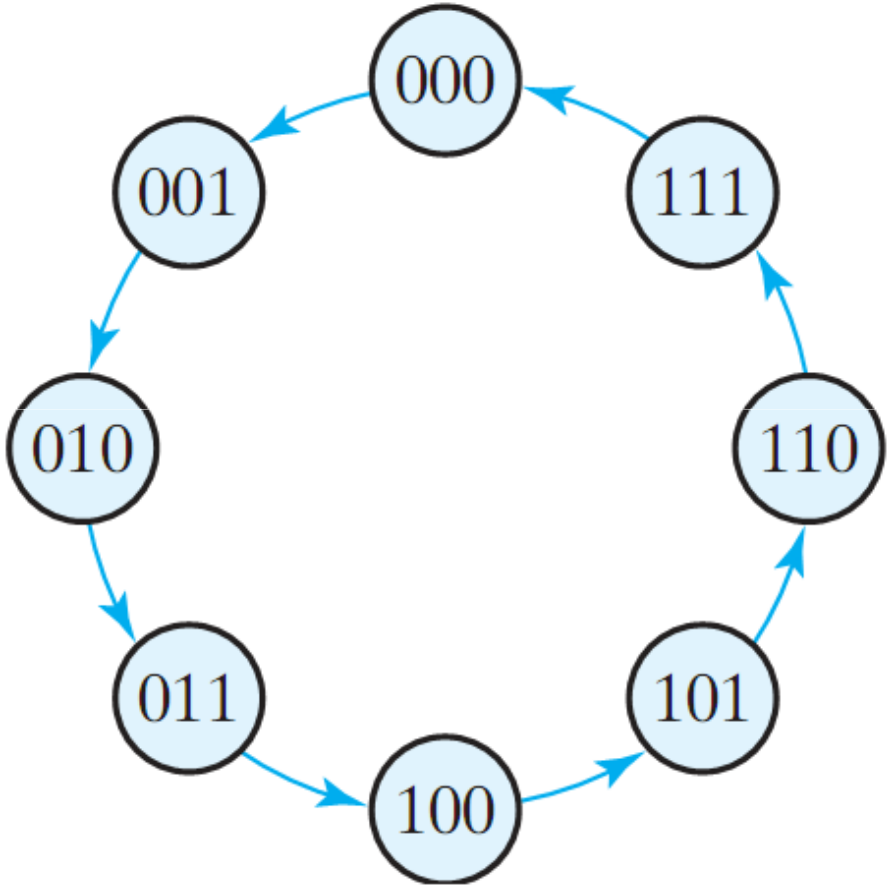
Problem: Design of A 3-bit Counter

Design a circuit that counts in binary form as follows
000, 001, 010, … 111, 000, 001, …

EXAMPLE 2 (CONT.)

❖**Step1: State Diagram**

- The outputs = the states
- Where is the input?
- What is the type of this sequential circuit?



EXAMPLE 2 (CONT.)

❖**Step2: State Table**

❖*We choose T-FF*

T–FF excitation table

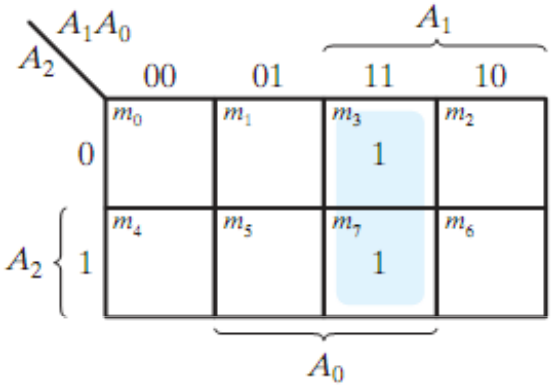
Q(t +1)	T	Operation
$Q(t)$	0	No change
$\overline{Q}(t)$	1	Complement

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

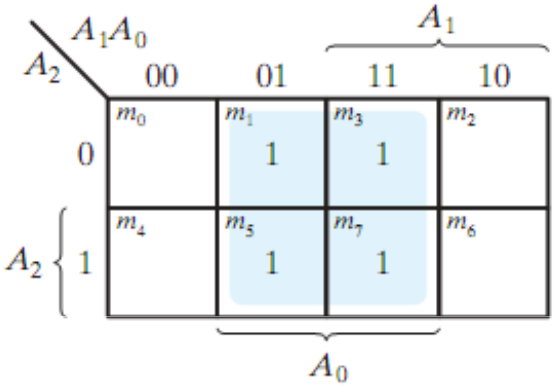
0

EXAMPLE 2 (CONT.)

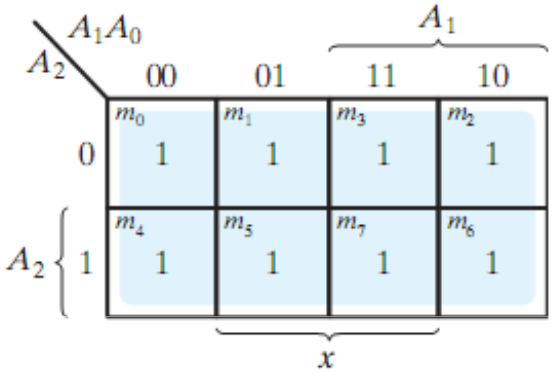
- **Step3: State Equations**



$T_{A2} = A_1A_0$



$T_{A1} = A_0$



$T_{A0} = 1$

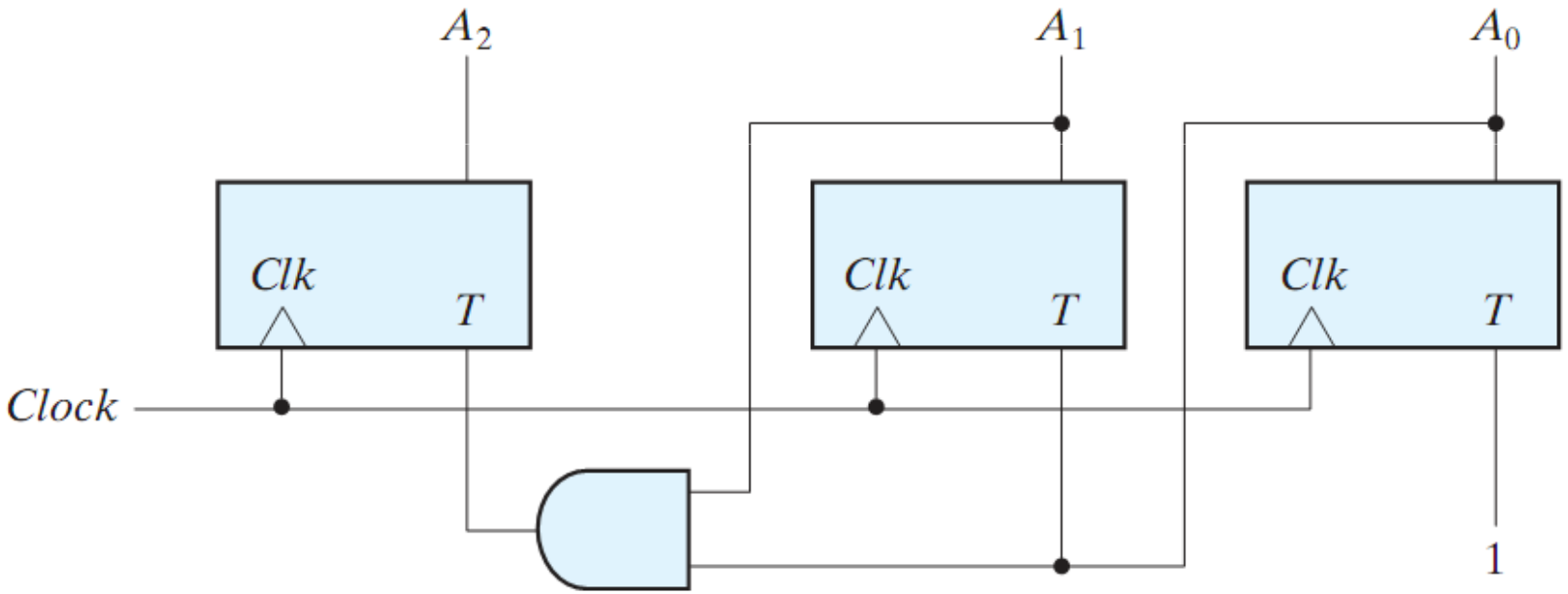
EXAMPLE 2 (CONT.)

❖**Step4: Draw Circuit**

$T_{A0} = 1$

$T_{A1} = A_0$

$T_{A2} = A_1A_0$



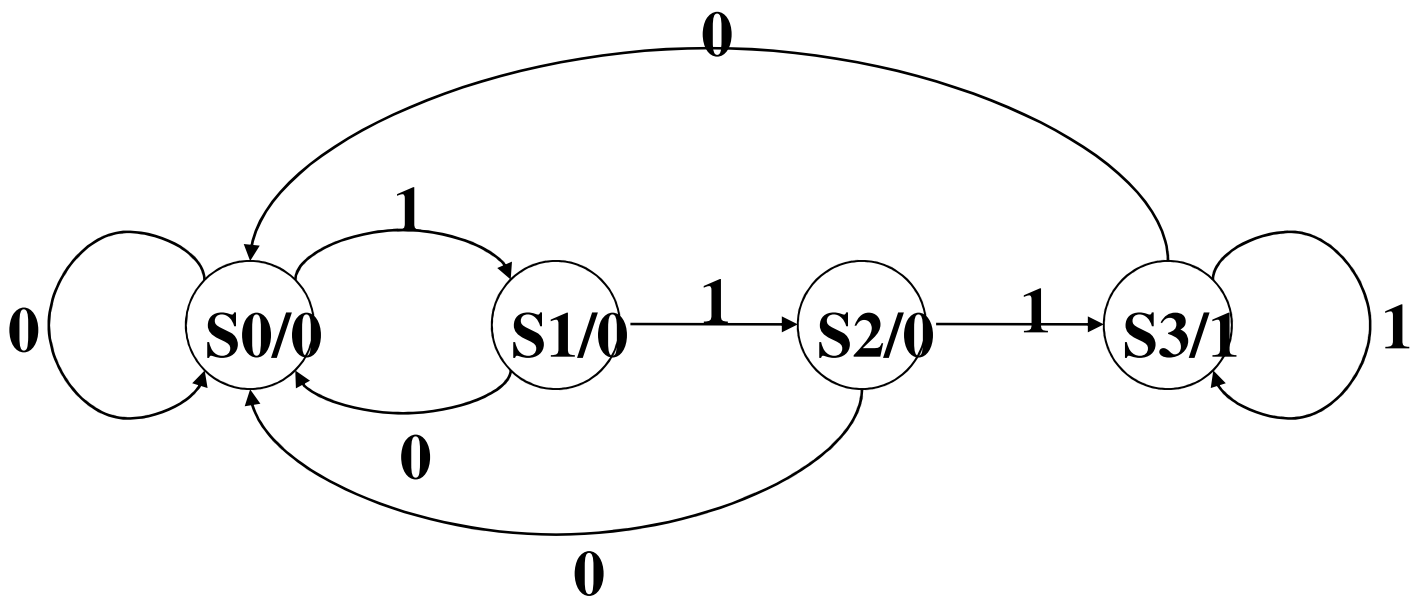
EXAMPLE 3

Problem: Design of A Sequence Recognizer

Design a **Moore** machine to detect the sequence (111). The circuit has one input (X) and one output (Z).

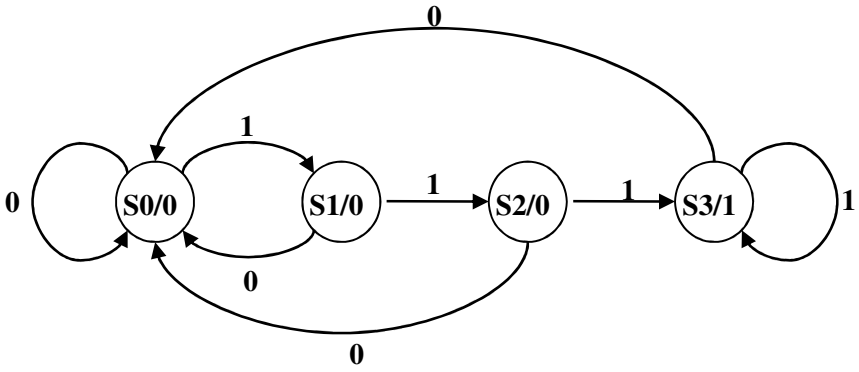
- **Step1: State Diagram**

Sequence to be detected:111



EXAMPLE 3 (CONT.)

- **Step2: State Table**
- ***Use binary encoding***
- ***Use JK-FF and D-FF***



Inputs of Comb.Circuits			Next State		Outputs of Comb.Circuit			Output
Present State		Input			Flip-flop Inputs			
A	B	X	A	B	J_A	K_A	D_B	Z
0	0	0	0	0	0	X	0	0
0	0	1	0	1	0	X	1	0
0	1	0	0	0	0	X	0	0
0	1	1	1	0	1	X	0	0
1	0	0	0	0	X	1	0	0
1	0	1	1	1	X	0	1	0
1	1	0	0	0	X	1	0	1
1	1	1	1	1	X	0	1	1

EXAMPLE 3 (CONT.)

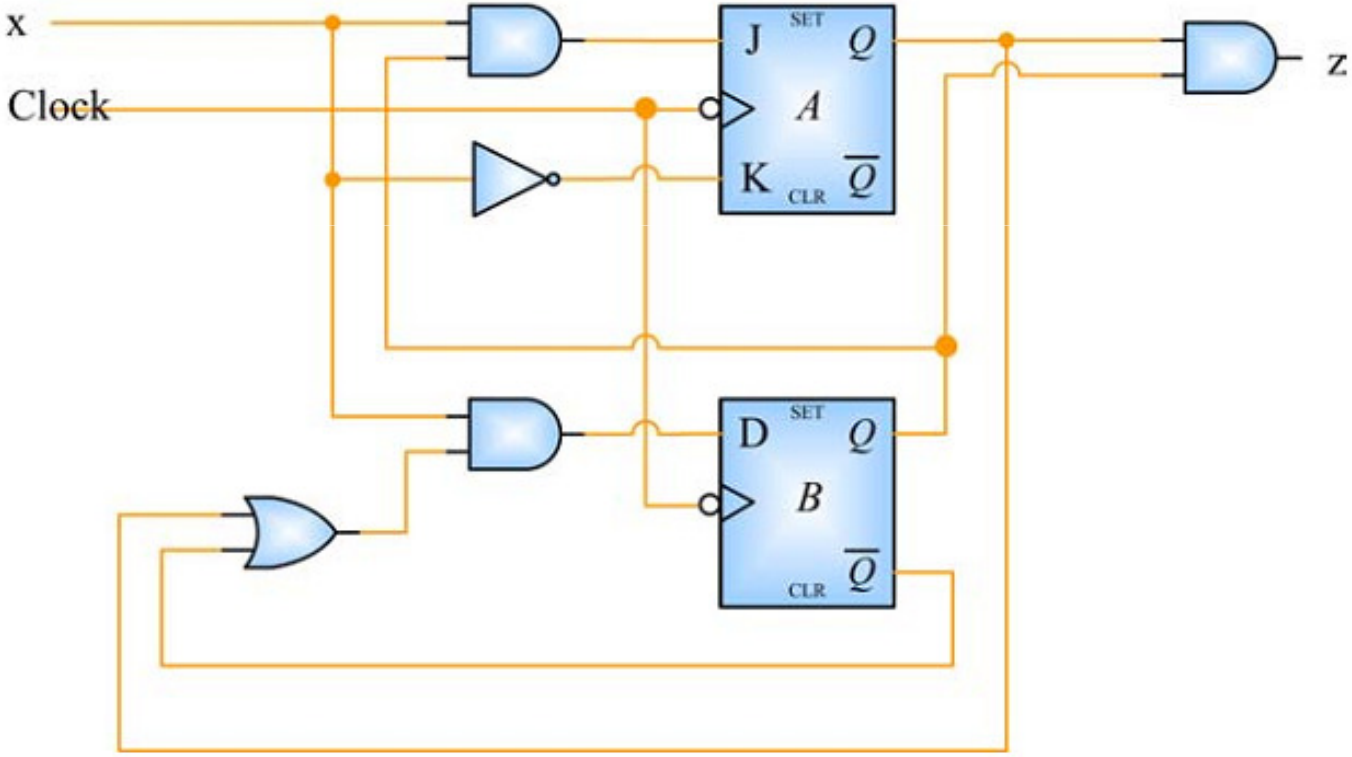
- **Step4: Draw Circuit**
- ***For step3, use k-maps as usual***

$J_A = XB$

$K_A = X'$

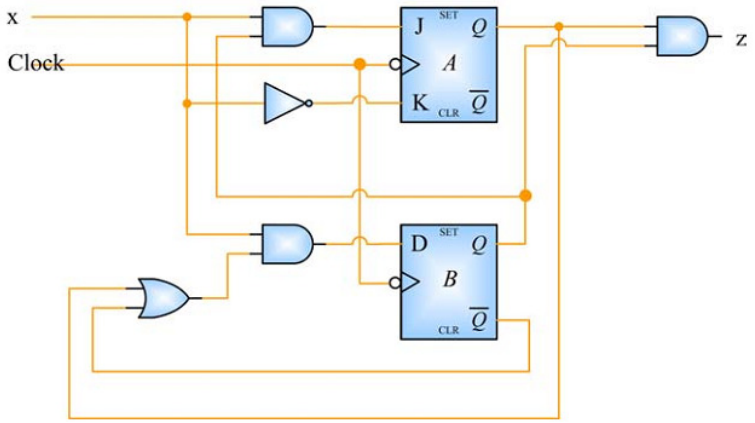
$D_B = X(A+B)$

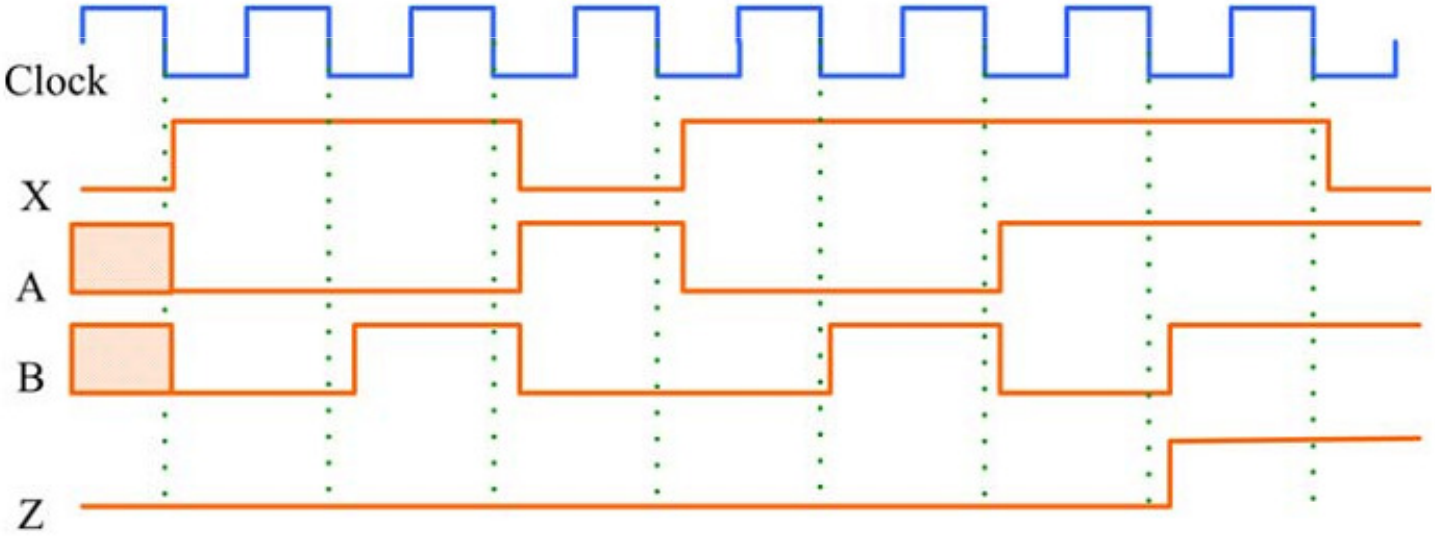
$Z = A.B$



EXAMPLE 3 (CONT.)

- **Timing Diagram (verification)**
- ***Question: Does it detect 111 ?***

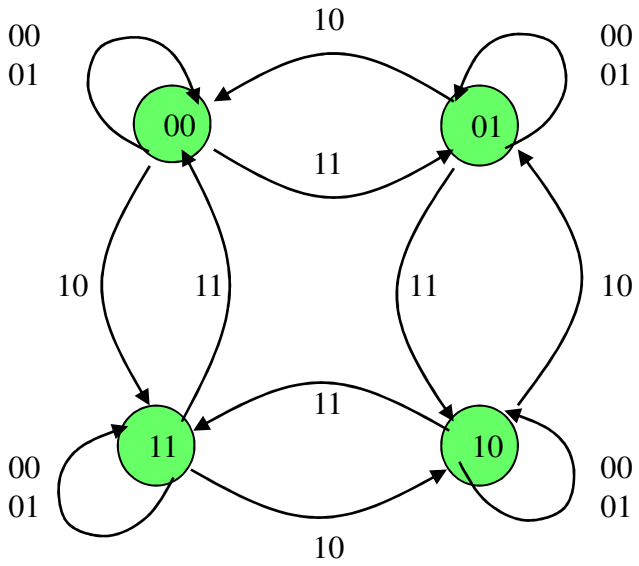




EXAMPLE 4

- **Problem: Design Up/Down counter with Enable**
- Design a sequential circuit with two JK flip-flops A and B and two inputs X and E. If E = 0, the circuit remains in the same state, regardless of the input X. When E = 1 and X = 1, the circuit goes through the state transitions from 00 to 01 to 10 to 11, back to 00, and then repeats. When E = 1 and X = 0, the circuit goes through the state transitions from 00 to 11 to 10 to 01, back to 00 and then repeats.

EXAMPLE 4 (CONT.)



Present State		Inputs		Next State		FF Inputs			
A	B	E	X	A	B	J_A	K_A	J_B	K_B
0	0	0	0	0	0	0	X	0	X
0	0	0	1	0	0	0	X	0	X
0	0	1	0	1	1	1	X	1	X
0	0	1	1	0	1	0	X	1	X
0	1	0	0	0	1	0	X	X	0
0	1	0	1	0	1	0	X	X	0
0	1	1	0	0	0	0	X	X	1
0	1	1	1	1	0	1	X	X	1
1	0	0	0	1	0	X	0	0	X
1	0	0	1	1	0	X	0	0	X
1	0	1	0	0	1	X	1	1	X
1	0	1	1	1	1	X	0	1	X
1	1	0	0	1	1	X	0	X	0
1	1	0	1	1	1	X	0	X	0
1	1	1	0	1	0	X	0	X	1
1	1	1	1	0	0	X	1	X	1

EXAMPLE 4 (CONT.)

EX / AB	00	01	11	10
00	0	0	0	1
01	0	0	1	0
11	x	x	x	x
10	x	x	x	x

$J_A = BEX + B'EX'$

EX / AB	00	01	11	10
00	x	x	x	x
01	x	x	x	x
11	0	0	1	0
10	0	0	0	1

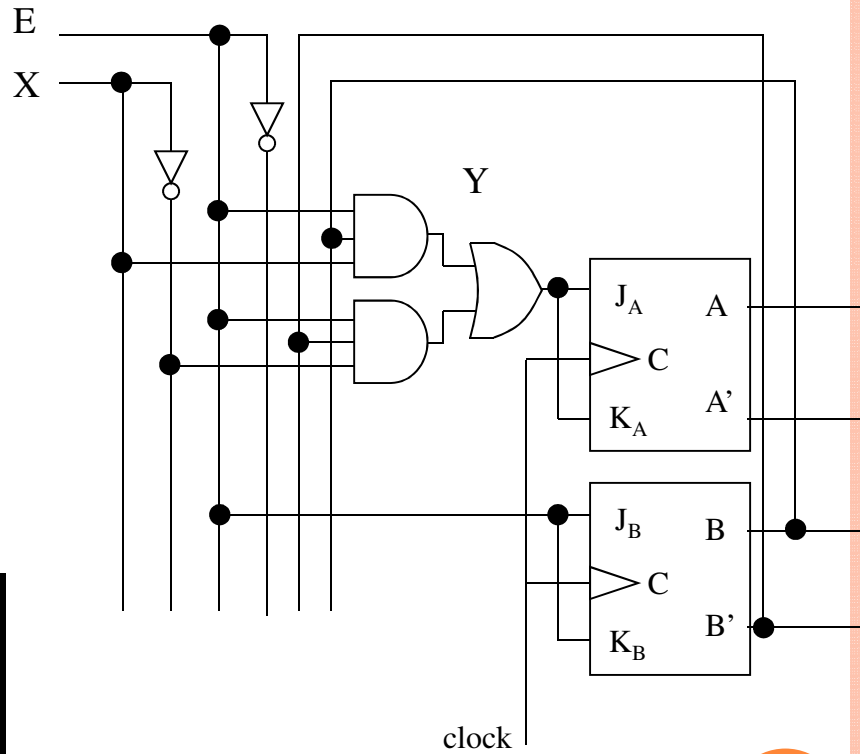
$K_A = BEX + B'EX'$

EX / AB	00	01	11	10
00	0	0	1	1
01	x	x	x	x
11	x	x	x	x
10	0	0	1	1

$J_B = E$

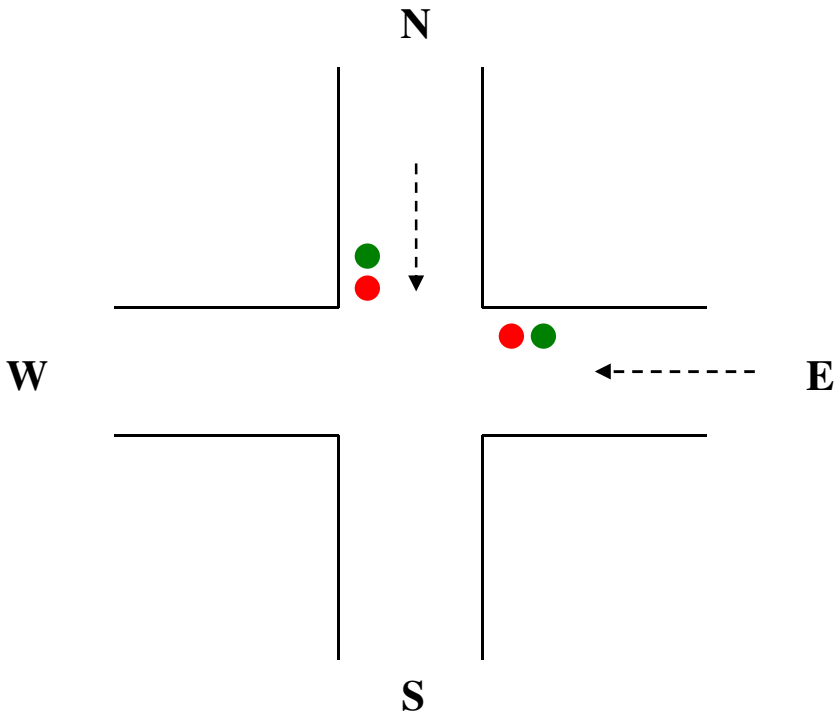
EX / AB	00	01	11	10
00	x	x	x	x
01	0	0	1	1
11	0	0	1	1
10	x	x	x	X

$K_B = E$



EXAMPLE 5

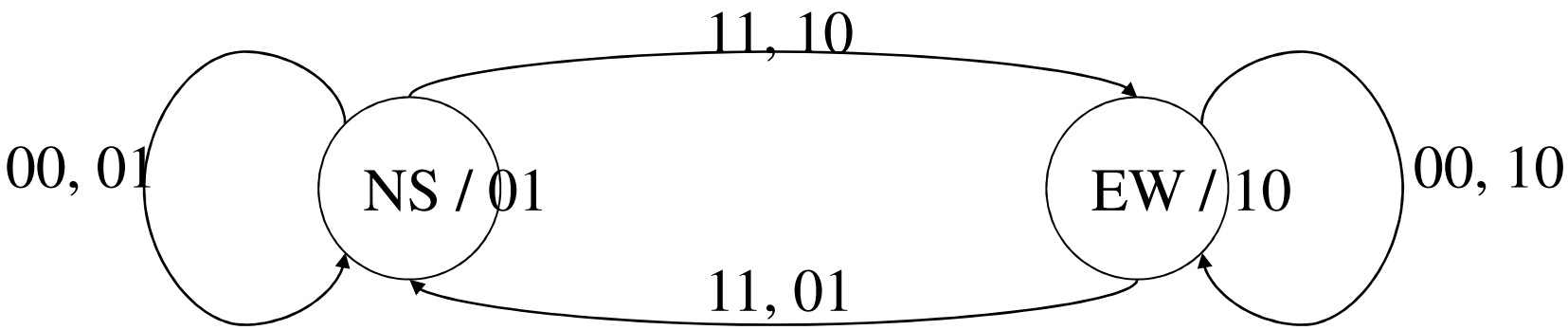
- **Problem:** Design a traffic light controller for a 2-way intersection. In each way, there is a sensor and a light



Traffic	Action
EW only	**EW** Signal green **NS** Signal red
NS only	**NS** Signal green **EW** Signal red
EW & **NS**	Alternate
No traffic	Previous state

EXAMPLE 5 (CONT.)

❖**Step1: State Diagram**



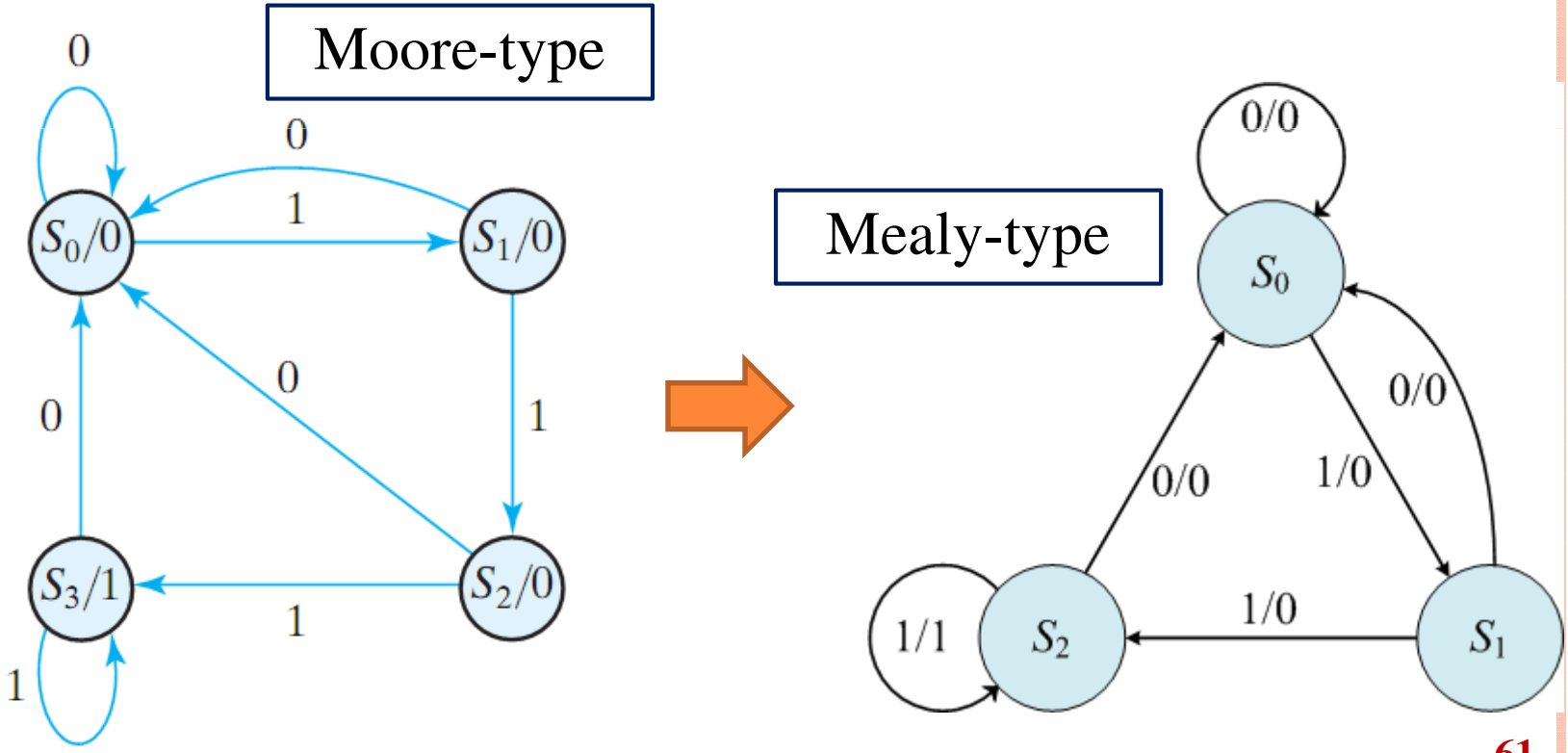
STATES	INPUTS	OUTPUTS
• **NS:** NS is green • **EW**: EW is green	• Sensors X_1, X_0 X_0: car coming on NS X_1: car coming on EW	• Light S_1, S_0 S_0 : NS is green S_1 : EW is green

EXAMPLE 5 (CONT.)

- **Exercise:** Complete the design using:
  - D-FF
  - JK-FF
  - T-FF

CHOICE OF FSM (MOORE VS MEALY)

- In general, Moore-type is easier to design, but may require more states → more FFs.
- Mealy-type is more flexible and may to state reduction → simpler implementation.
- 3 consecutive 1 detector example



State Table & Circuit

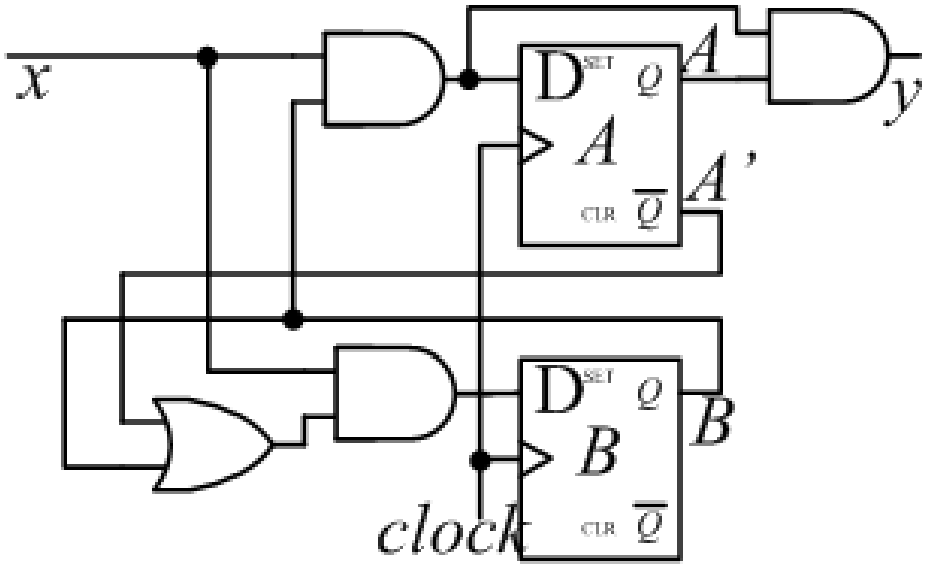
State Assignment:

$S_0 \rightarrow 00$

$S_1 \rightarrow 01$

$S_2 \rightarrow 11$

Present State A	Present State B	Input x	Next State A	Next State B	Output y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	1	0
1	1	0	0	0	0
1	1	1	1	1	1



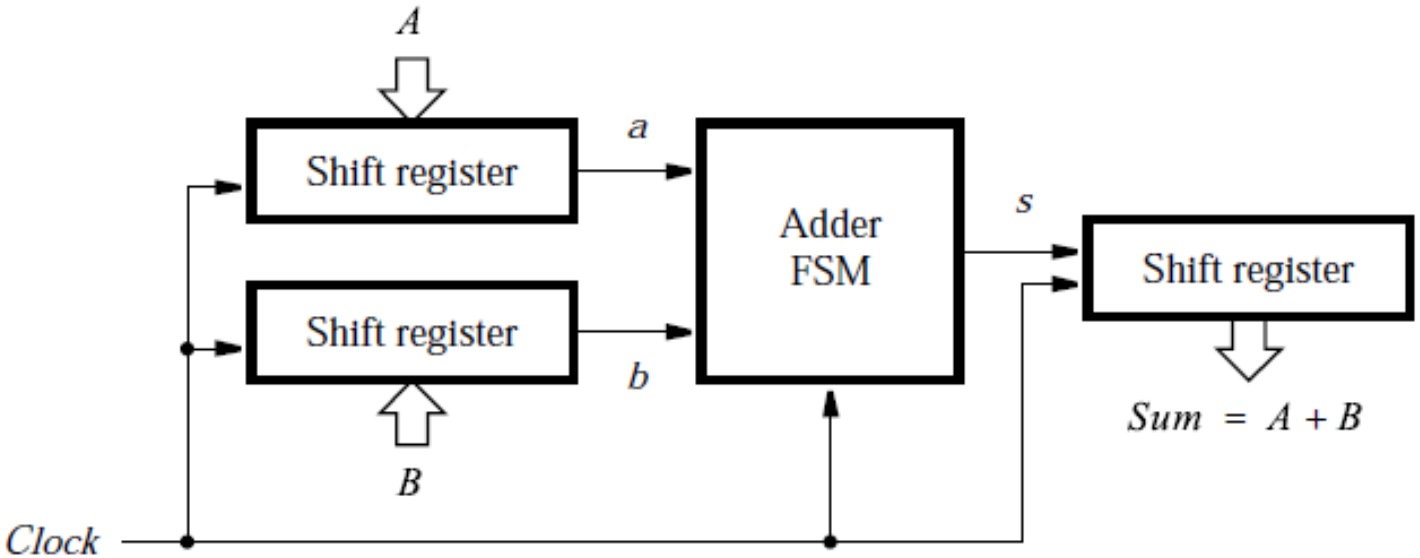
State Equations:

$$D_A = Bx, D_B = (A'+B)x,$$

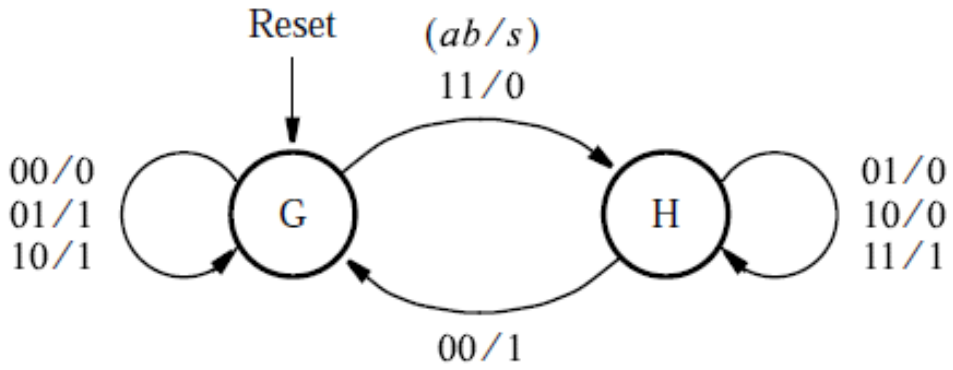
$$y = ABx$$

Mealy-type FSM for Serial Adder

- Input: $A=a_{n-1}a_{n-2}\ldots a_0$, $B=b_{n-1}b_{n-2}\ldots b_0$
- Output: $Sum=s_{n-1}s_{n-2}\ldots s_0$.
- 3 Shift register for storing A, B and Sum.
- Addition performed "sequentially".



State Diagram & Table



G: carry-in = 0
H: carry-in = 1

Mealy-type FSM for serial adder

Present state	Next state				Output s			
	ab = 00	01	10	11	00	01	10	11
G	G	G	G	H	0	1	1	0
H	G	H	H	H	1	0	0	1

State Table

Present state	Next state				Output			
	ab = 00	01	10	11	00	01	10	11
y	Y				s			
0	0	0	0	1	0	1	1	0
1	0	1	1	1	1	0	0	1

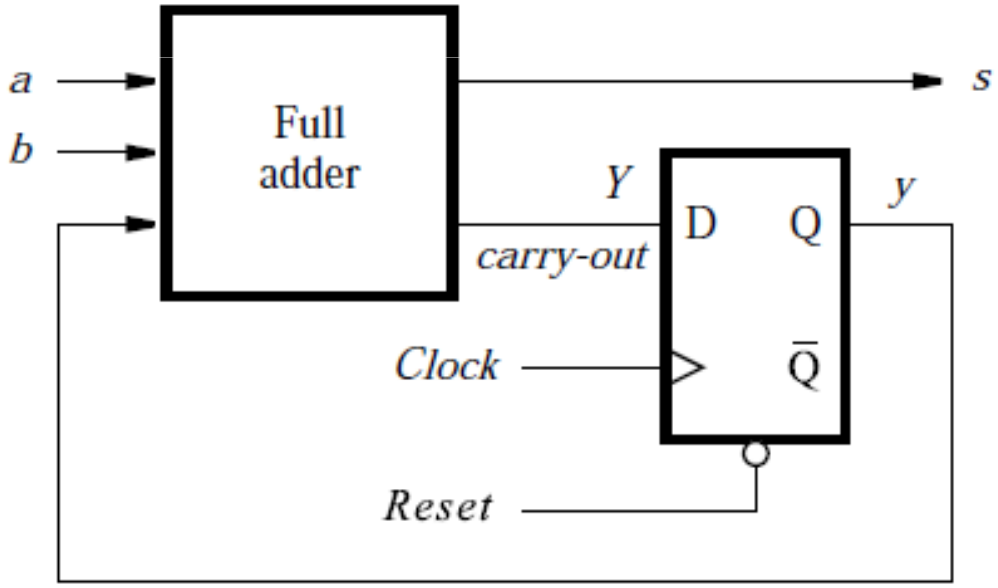
State-assigned Table

CIRCUIT

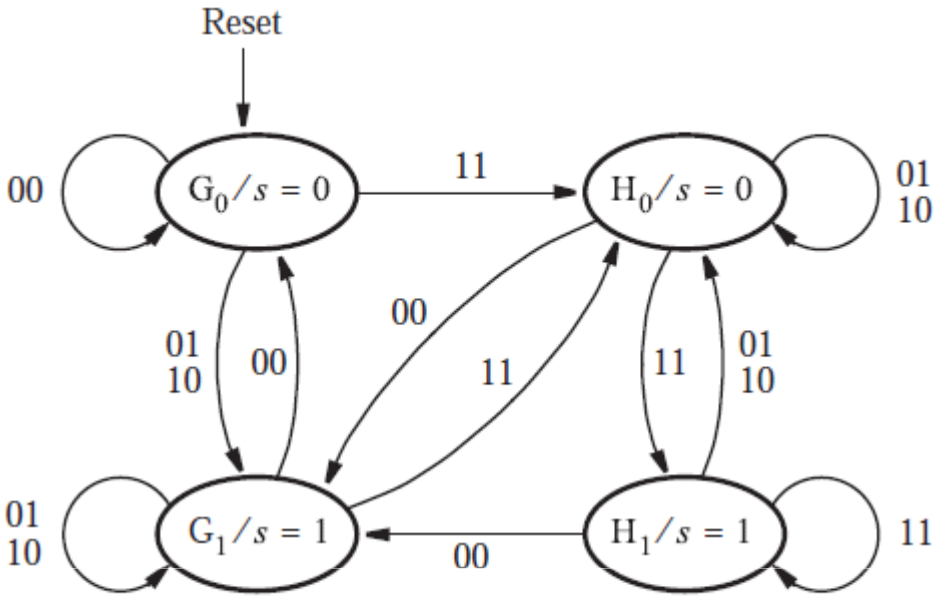
State Equations:

$$D = Y = ab + by + ay,$$

$$s = a \oplus b \oplus y$$



MOORE-TYPE STATE DIAGRAM & TABLE



Moore-type FSM for serial adder

Present state	Next state				Output
	$ab = 00$	01	10	11	s
G_0	G_0	G_1	G_1	H_0	0
G_1	G_0	G_1	G_1	H_0	1
H_0	G_1	H_0	H_0	H_1	0
H_1	G_1	H_0	H_0	H_1	1

State Table

Present state	Next state				Output
	$ab = 00$	01	10	11	s
y_2y_1	Y_2Y_1				
0 0	0 0	0 1	0 1	1 0	0
0 1	0 0	0 1	0 1	1 0	1
1 0	0 1	1 0	1 0	1 1	0
1 1	0 1	1 0	1 0	1 1	1

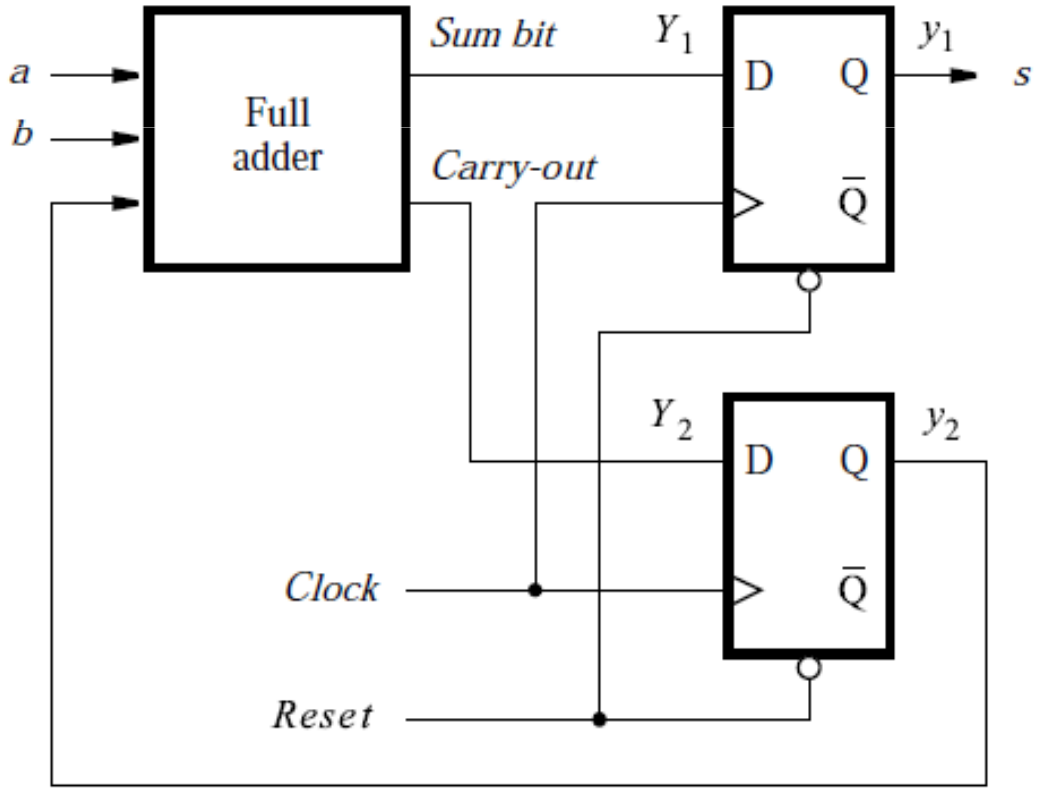
State-assigned Table

CIRCUIT

State Equations:

$$D_1 = Y_1 = a \oplus b \oplus y_2, D_2 = Y_2 = ab + by_2 + ay_2,$$

$$s = y_1$$



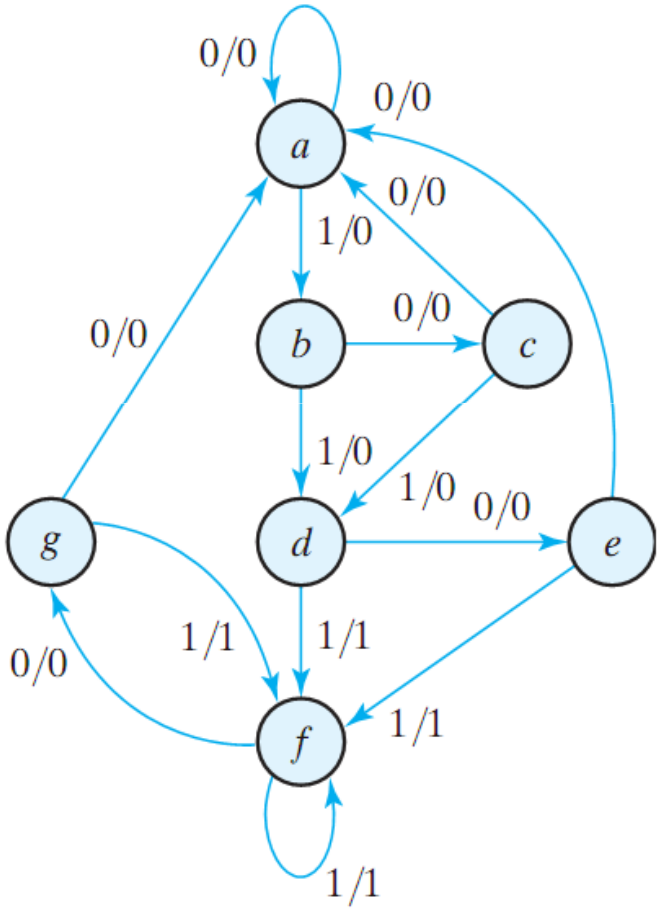
State Reduction

- Two sequential circuits may exhibits the same input-output behavior, but have a different number of states
- **State Reduction:** The process of reducing the number of states, while keeping the input-output behavior unchanged.
- It results in fewer Flip flops
- It may increase the combinational logic!

State Reduction (Example)

- **Is it possible to reduce this FSM?**
- **How many states?**
- **How many input/outputs?**

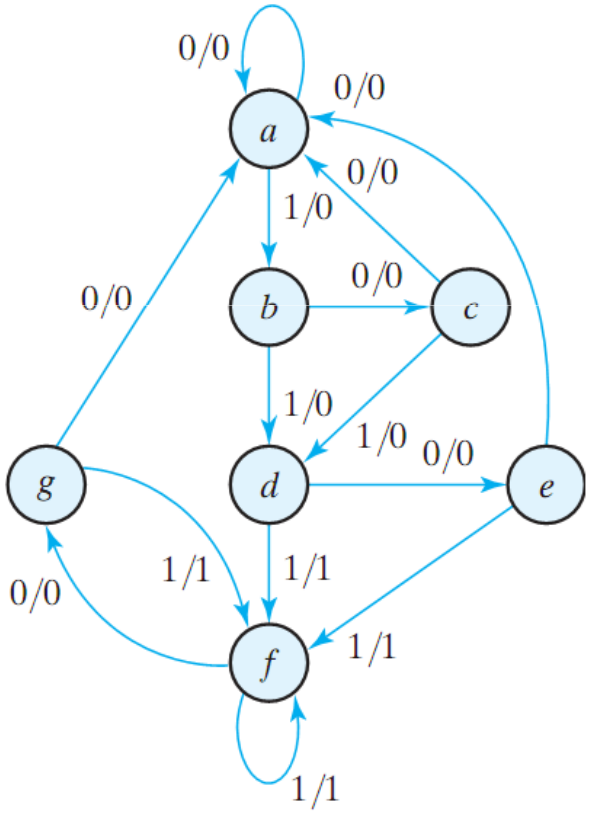
- **Notes:**
- we use letters to denote states rather than binary codes
- we only consider input/output sequence and transitions



State Reduction (Example)

- **Step 1: get the state table**

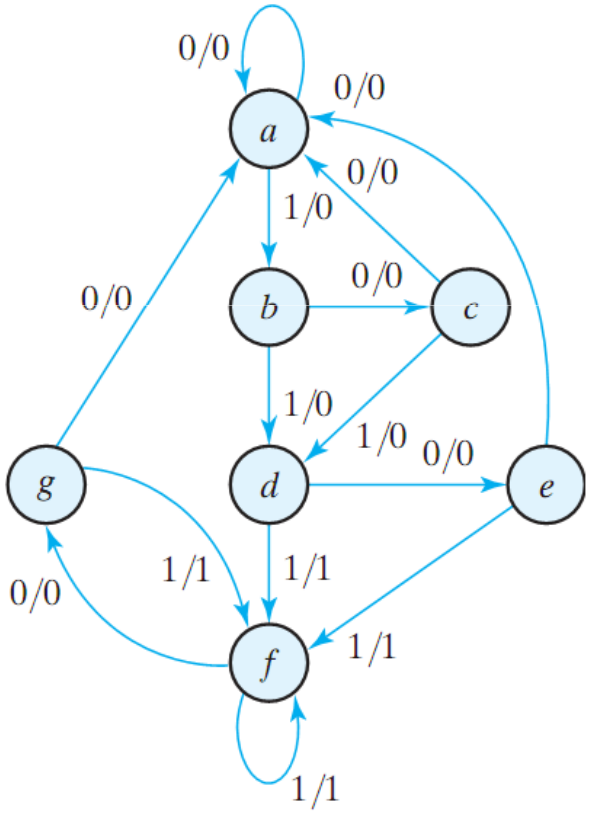
Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1



State Reduction (Example)

- **Step 1: get the state table**
- **Step 2: find similar states**

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

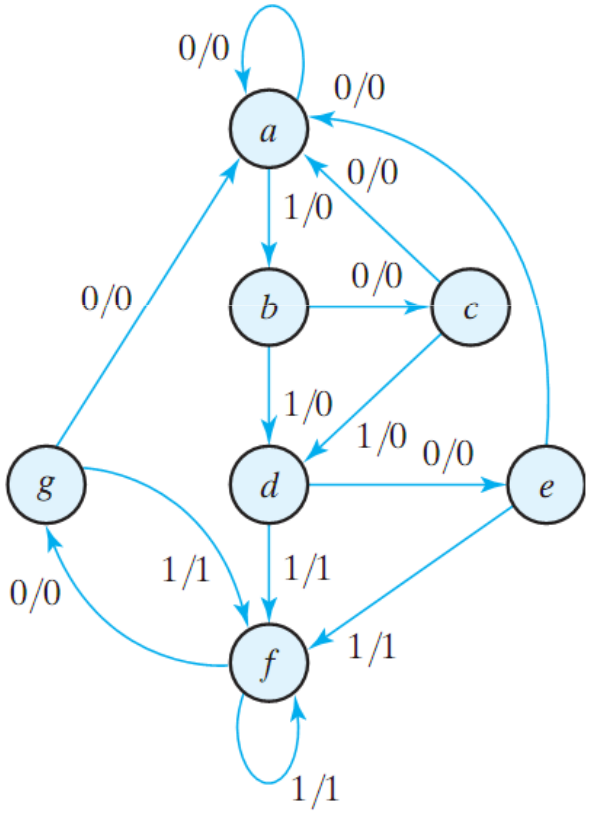


- ***e*** and ***g*** are equivalent states
- remove ***g*** and replace it with ***e***

State Reduction (Example)

- **Step 1: get the state table**
- **Step 2: find similar states**

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

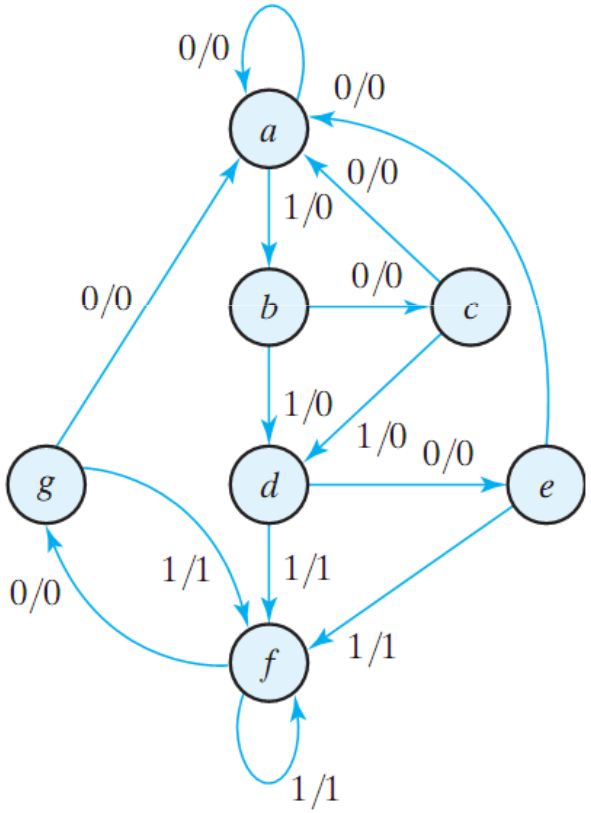


- ***e*** and ***g*** are equivalent states
- remove ***g*** and replace it with ***e***

State Reduction (Example)

- **Step 1: get the state table**
- **Step 2: find similar states**

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

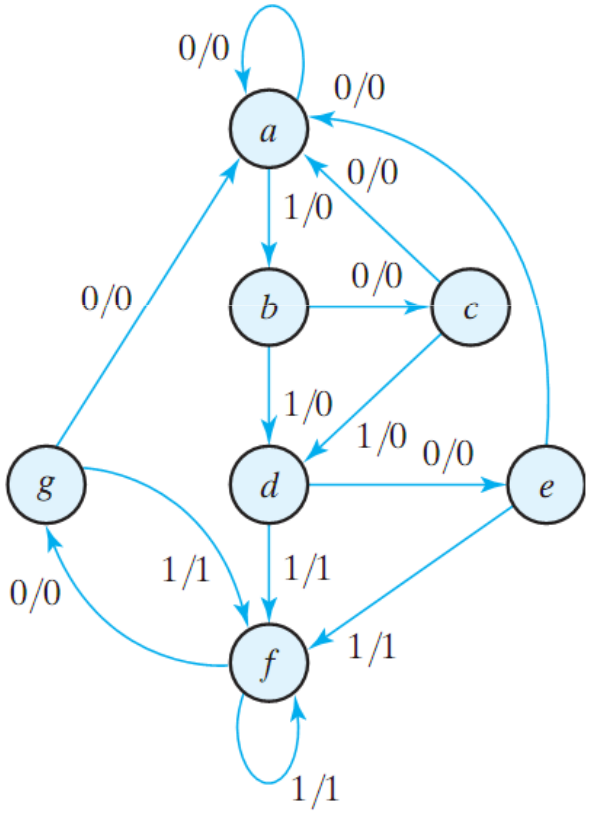


- ***d*** and ***f*** are equivalent states
- remove ***f*** and replace it with ***d***

State Reduction (Example)

- **Step 1: get the state table**
- **Step 2: find similar states**

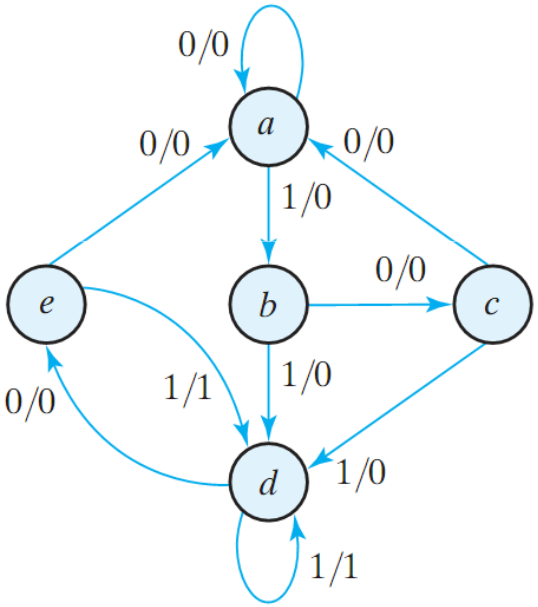
Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

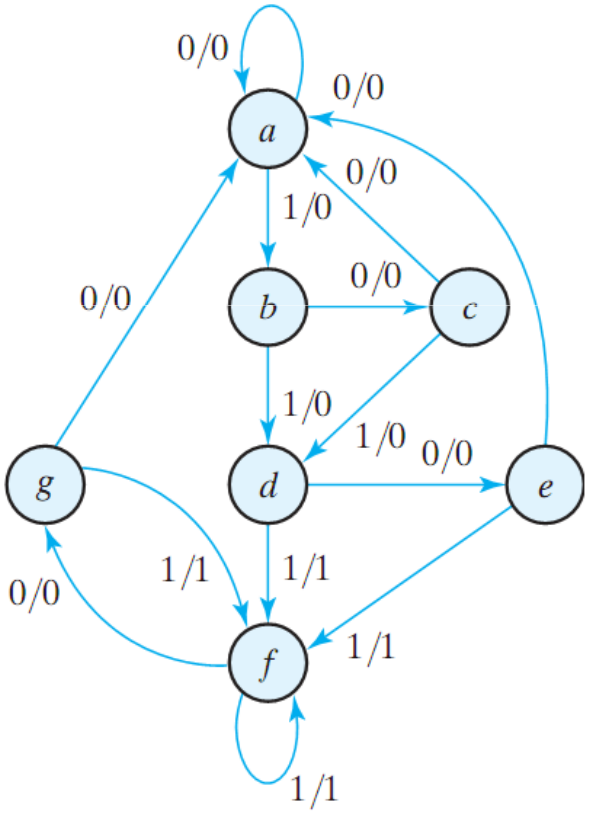


- ***d*** and ***f*** are equivalent states
- remove ***f*** and replace it with ***d***

State Reduction (Example)

- **Reduced FSM**





Verify sequence:

State	a	a	b	c	d	e	f	f	g	f
input	0	1	0	1	0	1	1	0	1	
output	0	0	0	0	0	1	1	0	1	

PARTITIONING MINIMIZATION PROCEDURE

Definition 1 *Two states S_i and S_j are said to be equivalent if and only if for every possible input sequence, the same output sequence will be produced regardless of whether S_i or S_j is the initial state.*

- *k-successor of S_i*: The next state with input=*k*.
- If S_i and S_j are equivalent, then their corresponding *k-successors* for all *k* are also equivalent.

Definition 2 *A partition consists of one or more blocks, where each block comprises a subset of states that may be equivalent, but the states in a given block are definitely not equivalent to the states in other blocks.*

EXAMPLE

- State table of this Moore-type FSM shown below

Step 1: P_1=(*ABCDEFG*)

Step 2: A, B, D -> output1, else-> output 0

P_2=(*ABD*)(*CEFG*)

Step 3: 0-successors of (ABD) = (BDB),

1-successors = (CFG)

0-successors of (CEFG)= (FFEF),1-suc=(ECDG)

P_3=(*ABD*)(*CEG*)(*F*)

Present state	Next state		Output z
	$w = 0$	$w = 1$	
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

EXAMPLE (CONT)

Step 4: F is not in the same block as CEG

$P_4=(AD)(B)(CEG)(F)$

Step 5: Check k-successors of AD and CEG

AD: 0-suc=BB,1-suc=CG

CEG: 0-suc=FFF,1-suc=ECG

Thus, $P_5=(AD)(B)(CEG)(F)$

(AD)->A, (CEG)->C

Present state	Next state		Output z
	$w=0$	$w=1$	
A	B	C	1
B	A	F	1
C	F	C	0
F	C	A	0

Present state	Next state		Output z
	$w=0$	$w=1$	
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

Previous Example

P_1=(abcdefg)

P_2=(abc)(defg)

P_3=(a)(bc)(df)(eg)

P_4=(a)(b)(c)(df)(eg)

P_5=(a)(b)(c)(df)(eg)

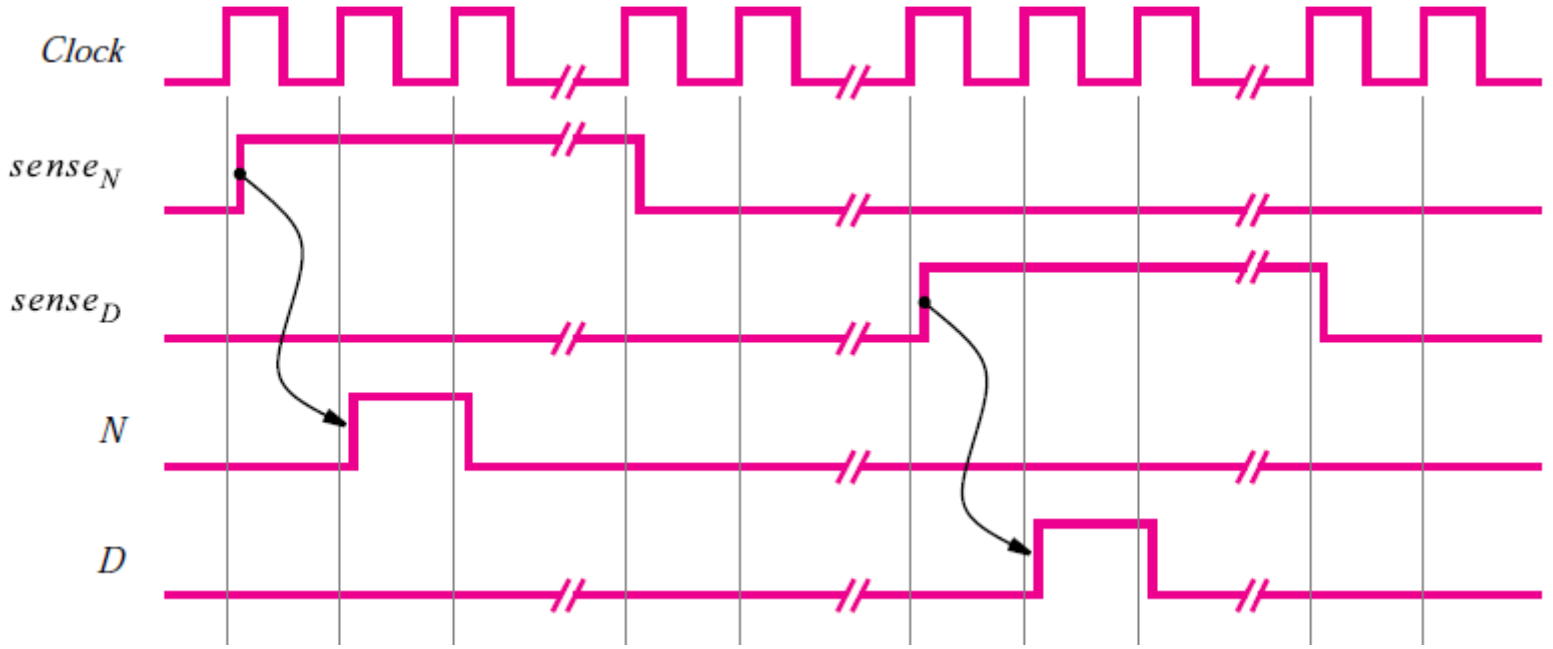
Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

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	d	0	1
e	a	d	0	1

A Vending Machine

Specification A candy vending machine with

- accepts only nickels and dimes
- 15 cents for 1 candy
- If 20 cents are deposited, no change, but 5 cents are credited for next purchase.



A Vending Machine (2)

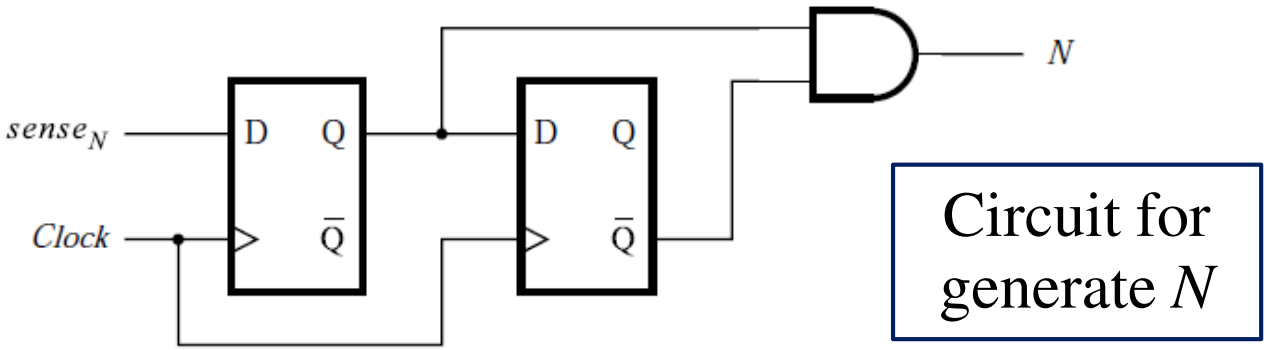
Input D, N : indicate whether a dime or a nickel is deposited.

D, N cannot be 1 at the same time.

($sense_N$, $sense_D$: Sensor output)

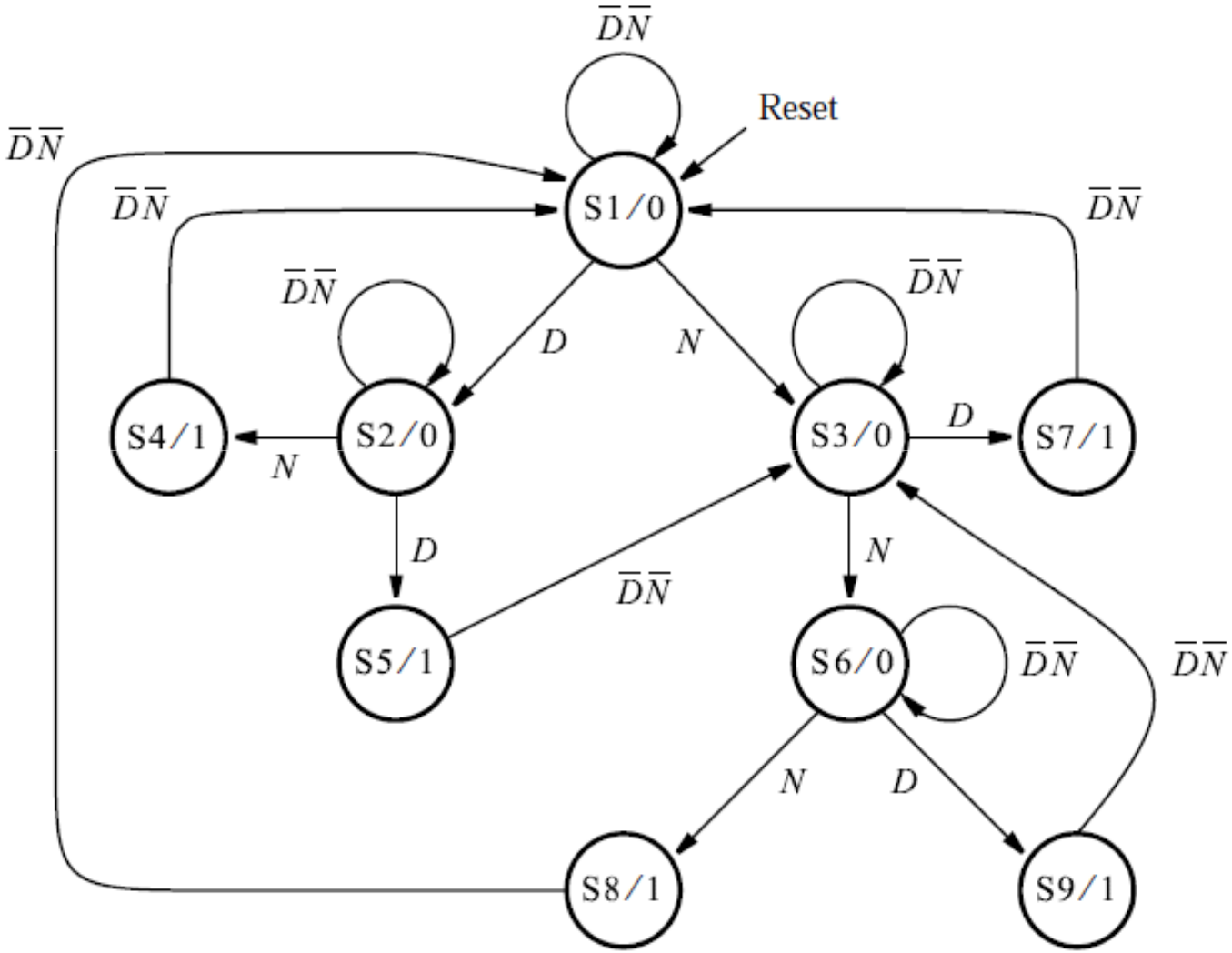
Output z : indicate whether a candy is released or not.

Clock period : 100 ns



Circuit for generate N

A Vending Machine (3)



State Table & Minimization

1: P_1=(123456789)

2: P_2=(1236)(45789)

3: P_3=(1)(3)(26)(45789)

4: P_4=(1)(3)(26)(478)(59)

5: P_5=(1)(3)(26)(478)(59)

New States:

(1)→S1

(26)→S2

(3)→S3

(478)→S4

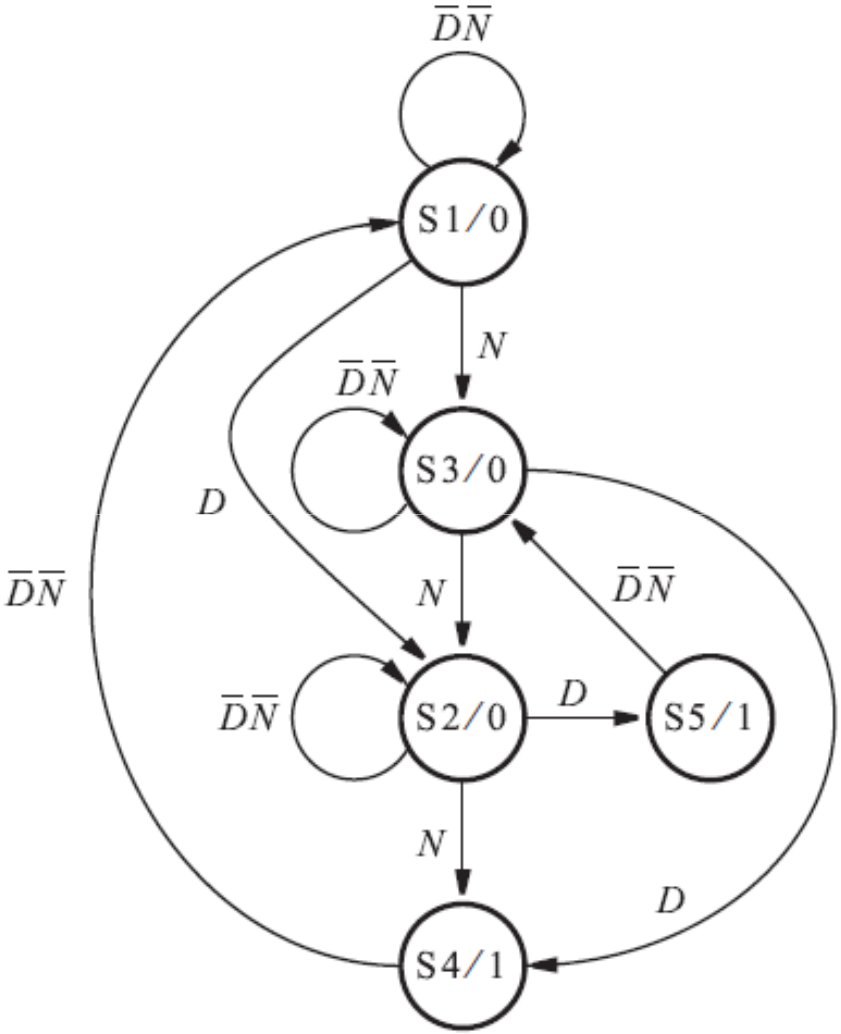
(59) →S5

Present state	Next state				Output z
	DN = 00	01	10	11	
S1	S1	S3	S2	–	0
S2	S2	S4	S5	–	0
S3	S3	S6	S7	–	0
S4	S1	–	–	–	1
S5	S3	–	–	–	1
S6	S6	S8	S9	–	0
S7	S1	–	–	–	1
S8	S1	–	–	–	1
S9	S3	–	–	–	1

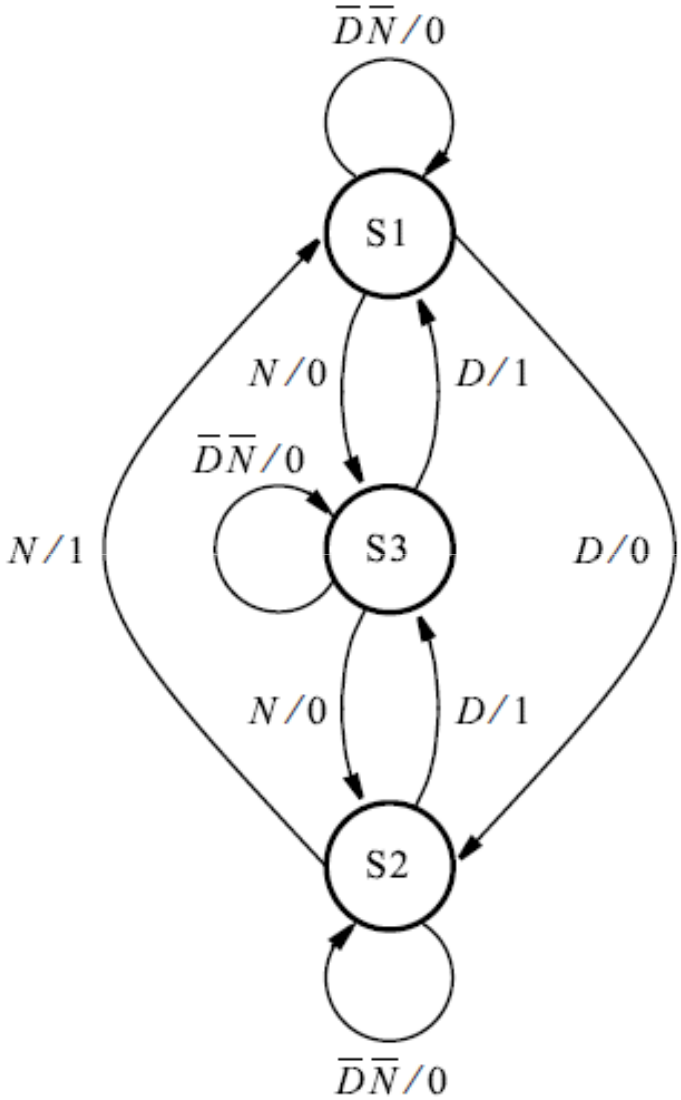
Present state	Next state				Output z
	DN = 00	01	10	11	
S1	S1	S3	S2	–	0
S2	S2	S4	S5	–	0
S3	S3	S2	S4	–	0
S4	S1	–	–	–	1
S5	S3	–	–	–	1

Note that this is an example of *Incompletely specified state table.*

FSM



Moore-type



Mealy-type

EQUIVALENT STATES

- Theorem: *Two states p and q are equivalent iff for every single input X, the outputs are the same and the next states are equivalent, that is,*

$\lambda(p, X) = \lambda(q, X)$ *and* $\delta(p, X) \equiv \delta(q, X)$

where

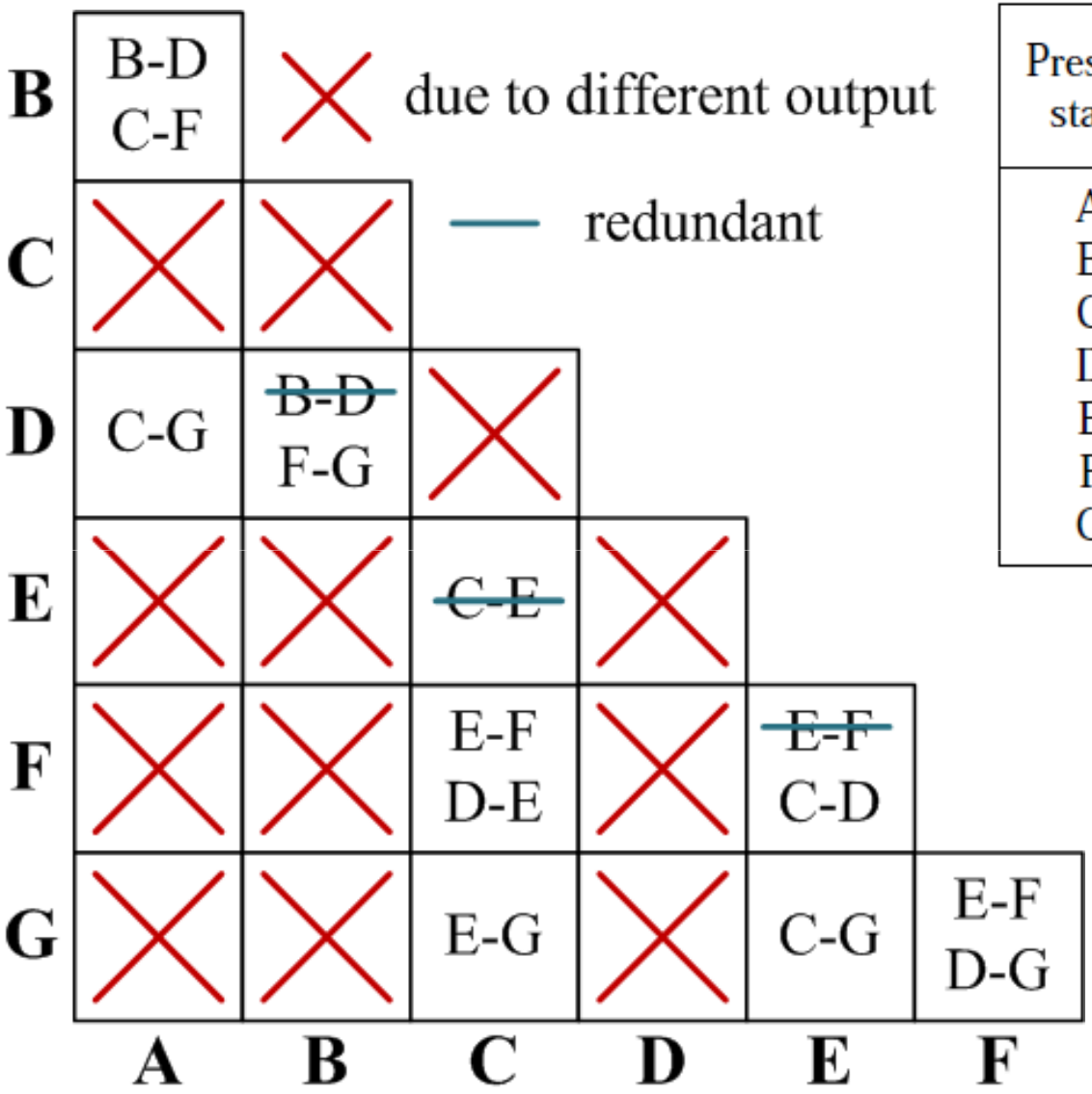
$\lambda(p, X)$: Output given "present" state p and input X

$\delta(p, X)$: Next state given p and X

IMPLICATION TABLE PROCEDURE

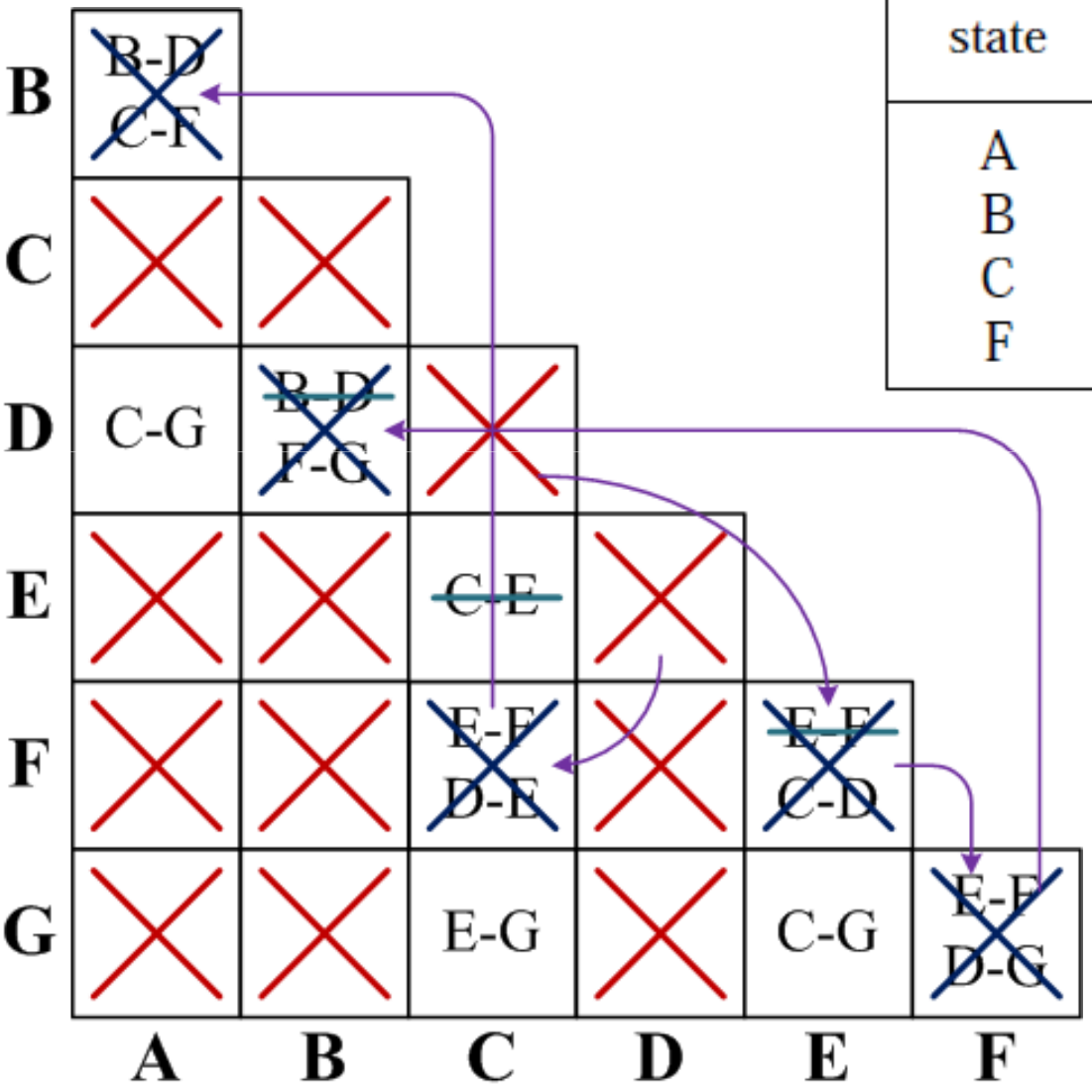
1. Construct a chart which contains a square for each pair of states.
2. Compare each pair of rows in the state table. If the outputs associated with states *i* and *j are different, place an* X *in square i-j to indicate that* $i \not\equiv j$. *If the outputs are the* same, place the implied pairs in square *i-j. (If the next states of i and j are m and n* for some input *x, then m-n is an implied pair.) If the outputs and next states are the* same (or if *i-j only implies itself), place a check* (√) *in square i-j to indicate that* $i \equiv j$.
3. Go through the table square-by-square. If square *i-j contains the implied pair m-n,* and square *m-n contains an* X, *then* $i \not\equiv j$, *and an* X *should be placed in square i-j.*
4. If any X's were added in step 3, repeat step 3 until no more X's are added.
5. For each square *i-j which does not contain an X,* $i \equiv j$.

IMPLICATION TABLE EXAMPLE



Present state	Next state		Output
	$w = 0$	$w = 1$	z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

Implication Table Example (2)



Present state	Next state		Output
	$w = 0$	$w = 1$	z
A	B	C	1
B	A	F	1
C	F	C	0
F	C	A	0

EQUIVALENT SEQUENTIAL CIRCUITS

Definition: Sequential circuit N_1 is equivalent to sequential circuit N_2 if for each state p in N_1, there is a state q in N_2 such that $p \equiv q$, and conversely, for each state s in N_2, there is a state t in N_1 such that $s \equiv t$, i.e.,

$$\forall p \in P, \exists q \in Q, p \equiv q \rightarrow N_1 \equiv N_2$$

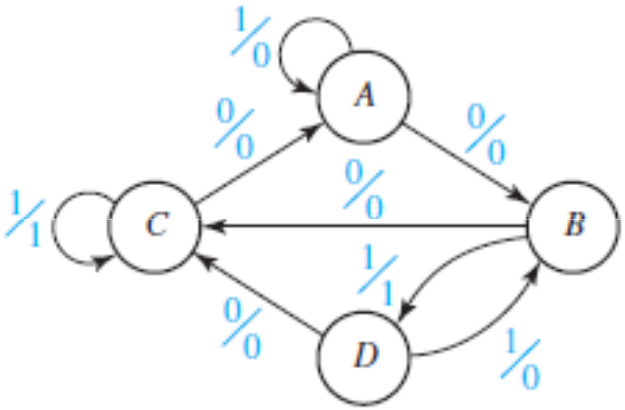
$$\forall s \in Q, \exists t \in Q, s \equiv t \rightarrow N_2 \equiv N_1$$

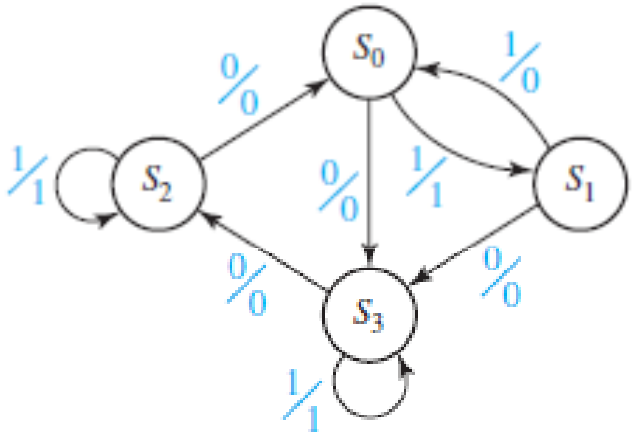
where P, Q : states of N_1, N_2

EXAMPLE $N_1 \equiv N_2$

	N_1			
	$X = 0$	1	$X = 0$	1
A	B	A	0	0
B	C	D	0	1
C	A	C	0	1
D	C	B	0	0

	N_2			
	$X = 0$	1	$X = 0$	1
S_0	S_3	S_1	0	1
S_1	S_3	S_0	0	0
S_2	S_0	S_2	0	0
S_3	S_2	S_3	0	1





	A	B	C	D
S_0	×	C–S_3 D–S_1	A–S_3 C–S_1	×
S_1	B–S_3 A–S_0	×	×	C–S_3 B–S_0
S_2	B–S_0 A–S_2	×	×	C–S_0 B–S_2
S_3	×	C–S_2 D–S_3	A–S_2 C–S_3	×

	A	B	C	D
S_0	×	C–S_3 D–S_1	~~A–S_3~~ ~~C–S_1~~	×
S_1	~~B–S_3~~ ~~A–S_0~~	×	×	C–S_3 B–S_0
S_2	B–S_0 A–S_2	×	×	~~C–S_0~~ ~~B–S_2~~
S_3	×	~~C–S_2~~ ~~D–S_3~~	A–S_2 C–S_3	×

STATE ASSIGNMENT

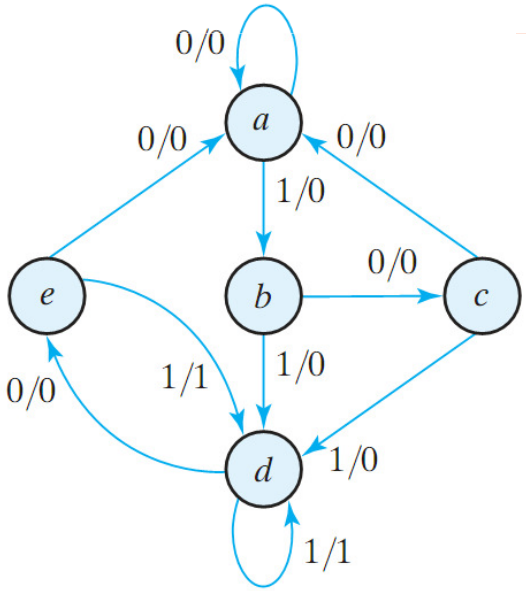
State Assignment: Assign unique binary codes to the states

• For m states, we need $\lceil \log_2 m \rceil$ bits (FF)

Example

• **Three Possible 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



One-Hot Encoding (State Assignment)

- Each state has only one 1, where the state variable whose value is 1 is regarded "hot", e.g., 001, 010, 100.
- Feature
  - Often lead to simpler output expressions.
  - Consequently, faster circuit.
- Useful when applied to CPLD or FPGA with a flip-flop in each cell, i.e., many FFs available.

One-Hot Encoding Example (3 consecutive 1 detector)

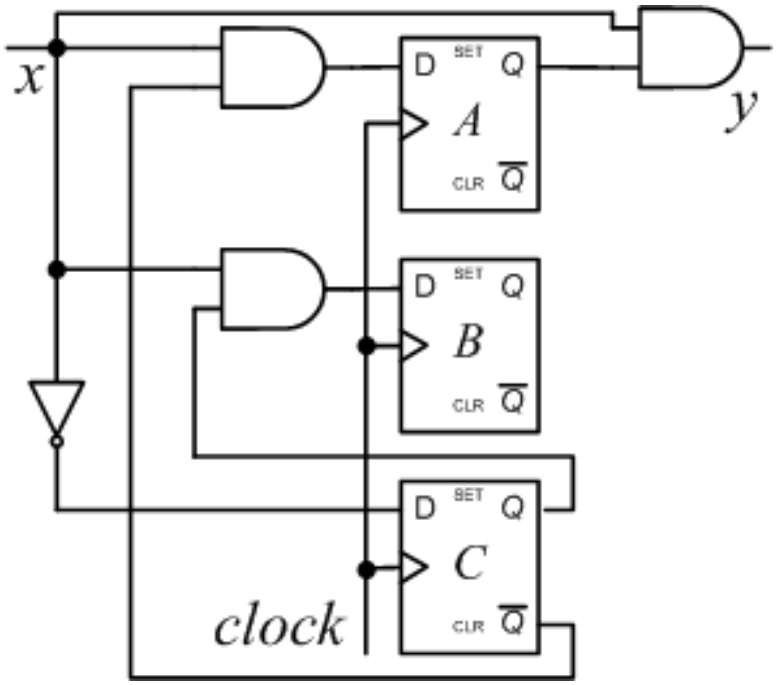
State Assignment:

$S_0 \rightarrow 001$

$S_1 \rightarrow 010$

$S_2 \rightarrow 100$

Present State			Input	Next State			Output
A	B	C	x	A	B	C	y
0	0	1	0	0	0	1	0
0	0	1	1	0	1	0	0
0	1	0	0	0	0	1	0
0	1	0	1	1	0	0	0
1	0	0	0	0	0	1	0
1	0	0	1	1	0	0	1



Input Equations:

$$D_A = C'x,$$

$$D_B = Cx,$$

$$D_C = x',$$

$$y = Ax$$

Design Summary

- To design a synchronous sequential circuit:
  - Obtain a state diagram
    - State reduction if necessary
  - Obtain State Table
    - State Assignment
    - Choose type of flip-flops
    - Use FF's excitation table to complete the table
  - Derive state equations
    - Use K-Maps
    - Obtain the FF input equations and the output equations
  - Draw the circuit diagram