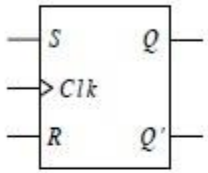
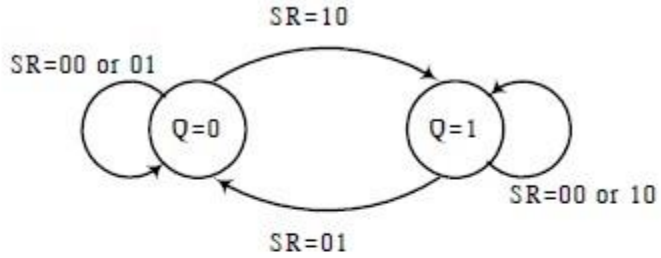
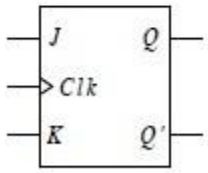
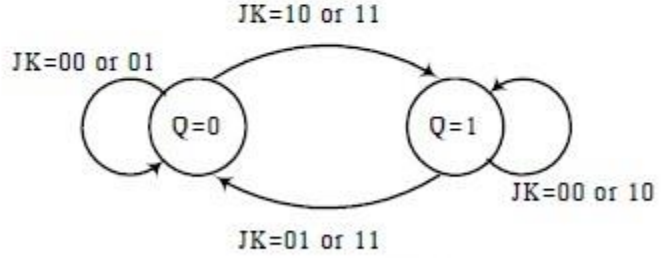
ITI1100 Section Z
Digital Systems I

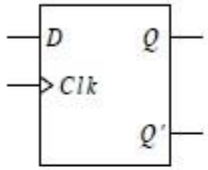
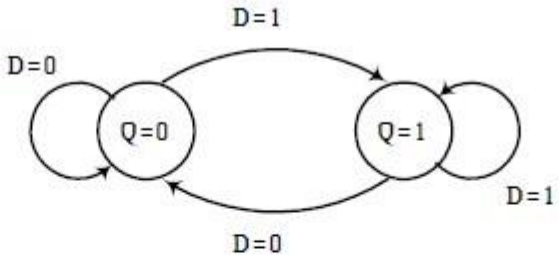
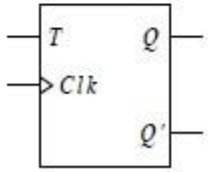
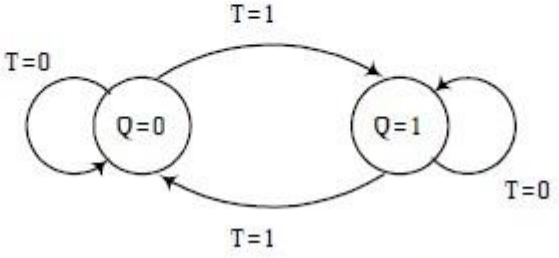
Chapter 5: Synchronous Sequential Logic (2)

Prof. Mohammad Alja'Afreh

Flip-Flop Full Description

Name / Symbol	Characteristic (Truth) Table	State Diagram / Characteristic Equations	Excitation Table
SR  	S R Q Q_{next} 0 0 0 0 0 0 1 1 0 1 0 0 0 1 1 0 1 0 0 1 1 0 1 1 1 1 0 × 1 1 1 ×	 $Q_{next} = S + R'Q$	Q Q_{next} S R 0 0 0 × 0 1 1 0 1 0 0 1 1 1 × 0
JK  	J K Q Q_{next} 0 0 0 0 0 0 1 1 0 1 0 0 0 1 1 0 1 0 0 1 1 0 1 1 1 1 0 1 1 1 1 0	 $Q_{next} = J'K'Q + JK' + JKQ'$ $= J'K'Q + JK'Q + JK'Q' + JKQ'$ $= K'Q(J'+J) + JQ'(K'+K)$ $= K'Q + JQ'$	Q Q_{next} J K 0 0 0 × 0 1 1 × 1 0 × 1 1 1 × 0

Flip-Flop Full Description (cont'd)

Name / Symbol	Characteristic (Truth) Table	State Diagram / Characteristic Equations	Excitation Table
D  	D Q Q_{next} 0 × 0 1 × 1	 $Q_{next} = D$	Q Q_{next} D 0 0 0 0 1 1 1 0 0 1 1 1
T  	T Q Q_{next} 0 0 0 0 1 1 1 0 1 1 1 0	 $Q_{next} = TQ' + T'Q = T \oplus Q$	Q Q_{next} T 0 0 0 0 1 1 1 0 1 1 1 0

Analysis of Synchronous Sequential Circuits

- **Analysis** describes what **a given circuit will do under certain conditions**
- **Behavior** of synchronous sequential circuit can be **determined** from
  - Its inputs
  - Its outputs
  - Flip-Flop states

Analysis of Synchronous Sequential Circuits (cont'd)

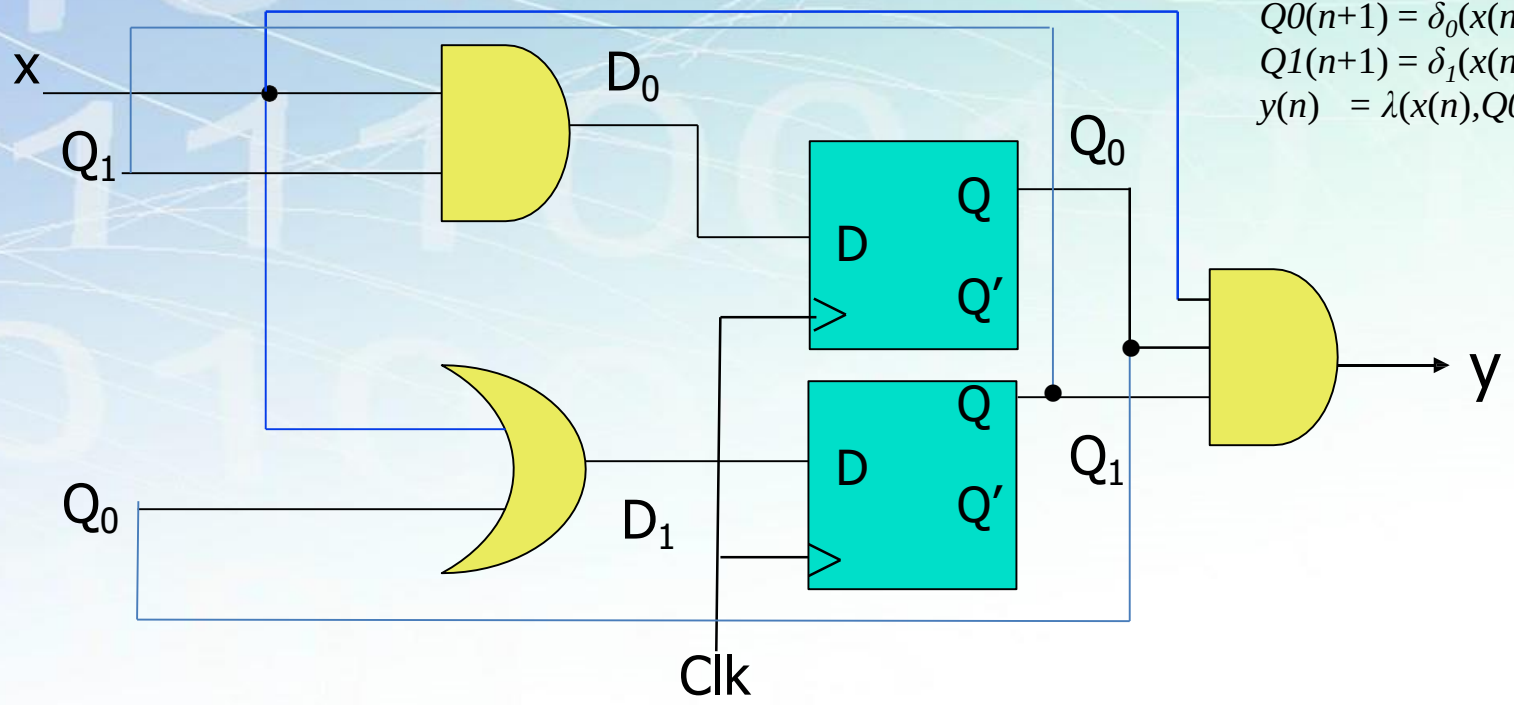
- Example of clocked sequential circuit:

from circuit = find

$Q0(n+1) = \delta_0(x(n), Q0(n), Q1(n))$

$Q1(n+1) = \delta_1(x(n), Q0(n), Q1(n))$

$y(n) = \lambda(x(n), Q0(n), Q1(n))$



Output equation → $y(t) = x(t)Q_1(t)Q_0(t)$

FF input equations →
$D_0(t) = x(t)Q_1(t)$
$D_1(t) = x(t) + Q_0(t)$

State equations →
$Q_0(t+1) = D_0(t)$
$Q_1(t+1) = D_1(t)$

State Table

- **Sequence** of **outputs**, **inputs**, and **flip-flop states** **enumerated** in **state table**
- **Present state** indicates **current value of flip-flops**
- **Next state** indicates **state after next clock edge**
- **Output** is **output value** on **current clock edge**

State Table

Present State	Next State		Output	
	x=0	x=1	x=0	x=1
0 0	00	10	0	0
0 1	10	10	0	0
1 0	00	11	0	0
1 1	10	11	0	1

$Q_1(t)\ Q_0(t)$ $Q_1(t+1)\ Q_0(t+1)$

$y(t) = x(t)Q_1(t)Q_0(t)$
$Q_0(t+1) = D_0(t) = x(t)Q_1(t)$
$Q_1(t+1) = D_1(t) = x(t) + Q_0(t)$

State Table (cont'd)

- All possible **input** combinations enumerated
- All possible **state** combinations enumerated
- Separate columns for **each output value**
- Sometimes easier to **designate a symbol for each state**

Let:

$s_0 = 00$

$s_1 = 01$

$s_2 = 10$

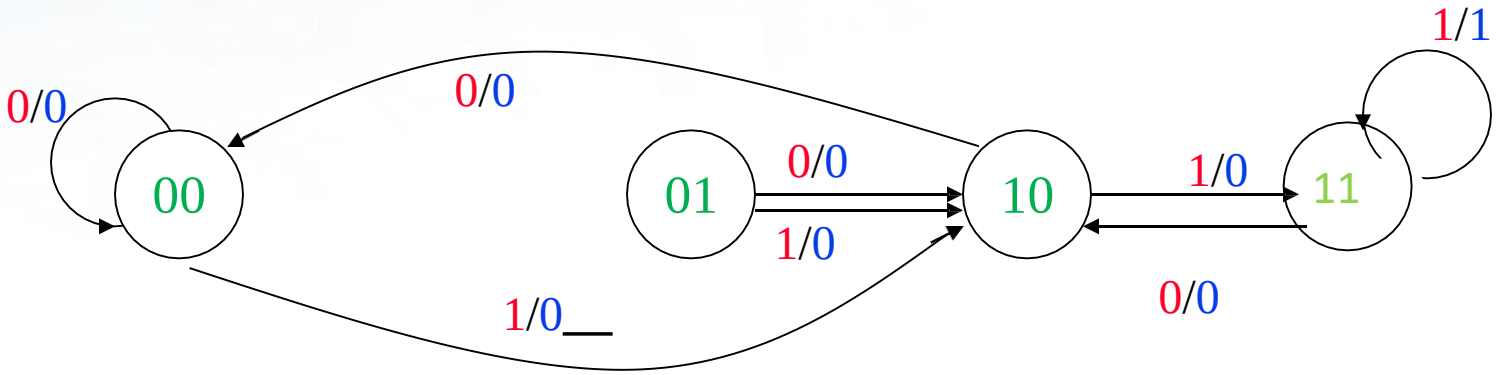
$s_3 = 11$

Present State	Next State		Output	
	x=0	x=1	x=0	x=1
s_0	s_0	s_2	0	0
s_1	s_2	s_2	0	0
s_2	s_0	s_3	0	0
s_3	s_2	s_3	0	1

State Diagram

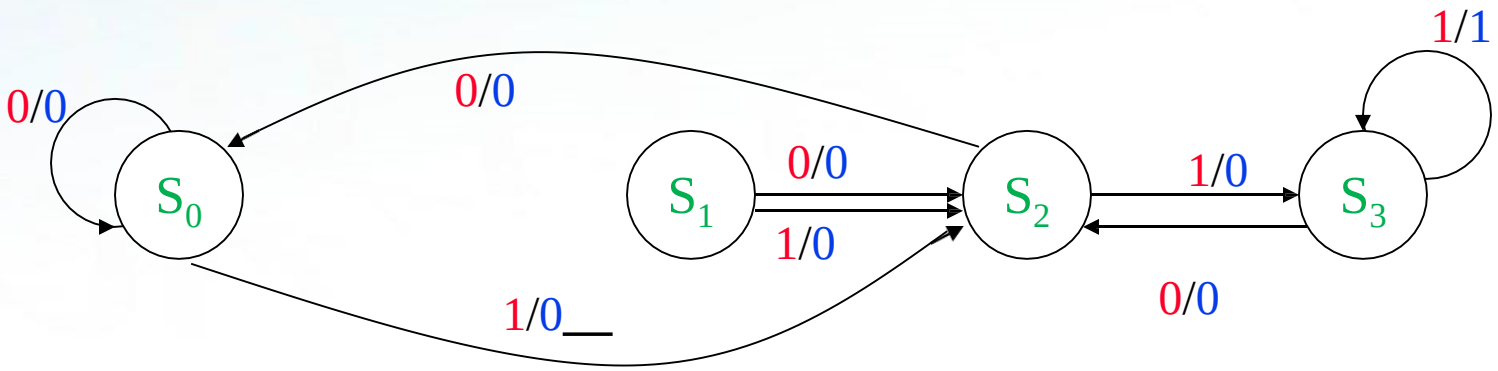
- **Circles** indicate **current state**
- **Arrows** point to **next state**
- For x/y, x is **input** and y is **output**

Present State	Next State		Output	
	x=0	x=1	x=0	x=1
0 0	00	10	0	0
0 1	10	10	0	0
1 0	00	11	0	0
1 1	10	11	0	1



State Diagram (cont'd)

- Each state has two arrows leaving
  - One for **x = 0** and one for **x = 1**
- Unlimited arrows can enter a state
- Use of state names is easier to identify



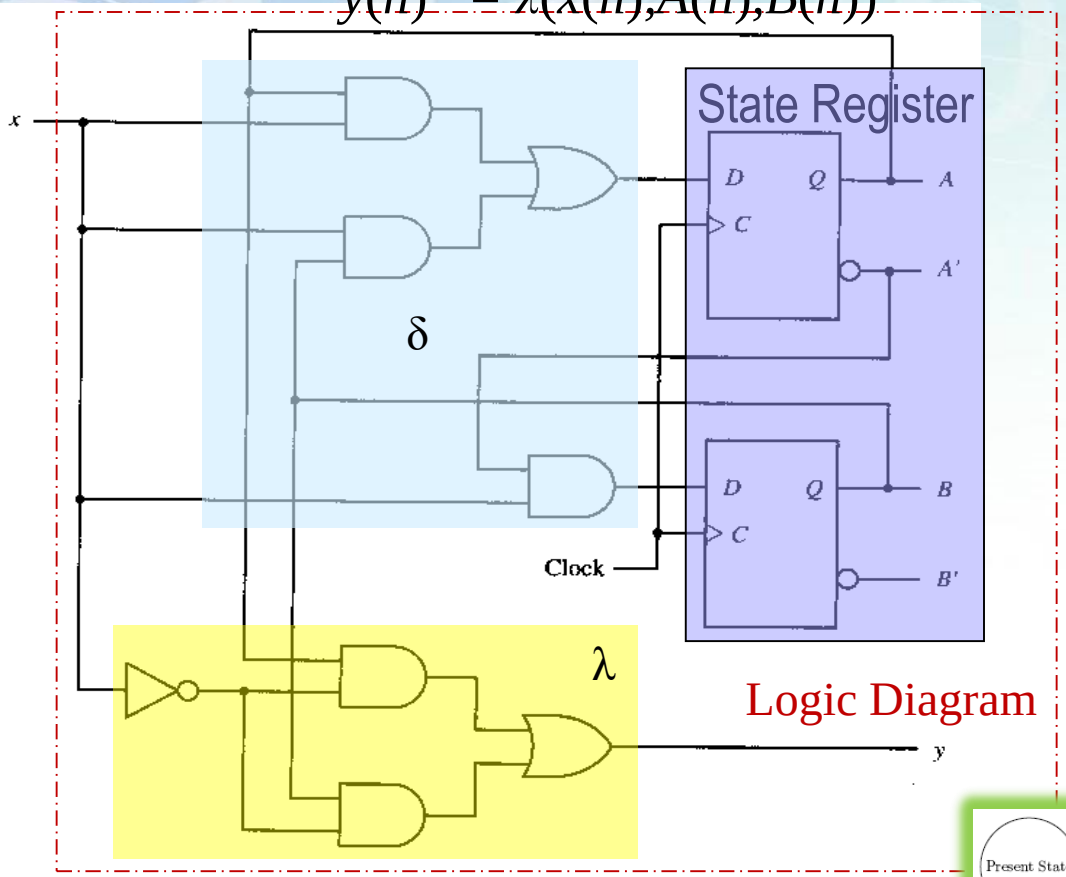
SEQUENTIAL CIRCUITS ANALYSIS

from circuit to state diagram = find

$$A(n+1) = \delta_A(x(n),A(n),B(n))$$
$$B(n+1) = \delta_B(x(n),A(n),B(n))$$
$$y(n) = \lambda(x(n),A(n),B(n))$$

$D_A = x\,A + x\,B = A^+$
$D_B = x\,A' = B^+$
} **Transition function** δ

$y = x'A + x'B$ **Output function** λ

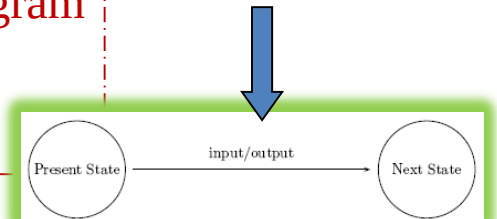


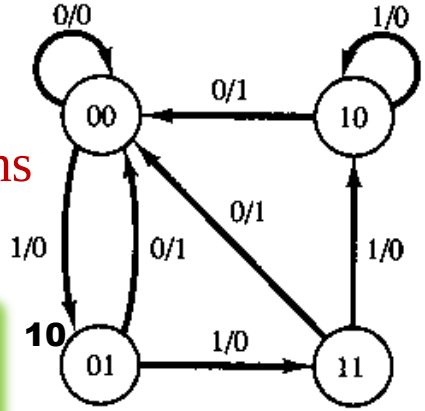
Logic Diagram

State Table

Present State		In	Next State		Out
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
- nodes = states
- arcs = transitions



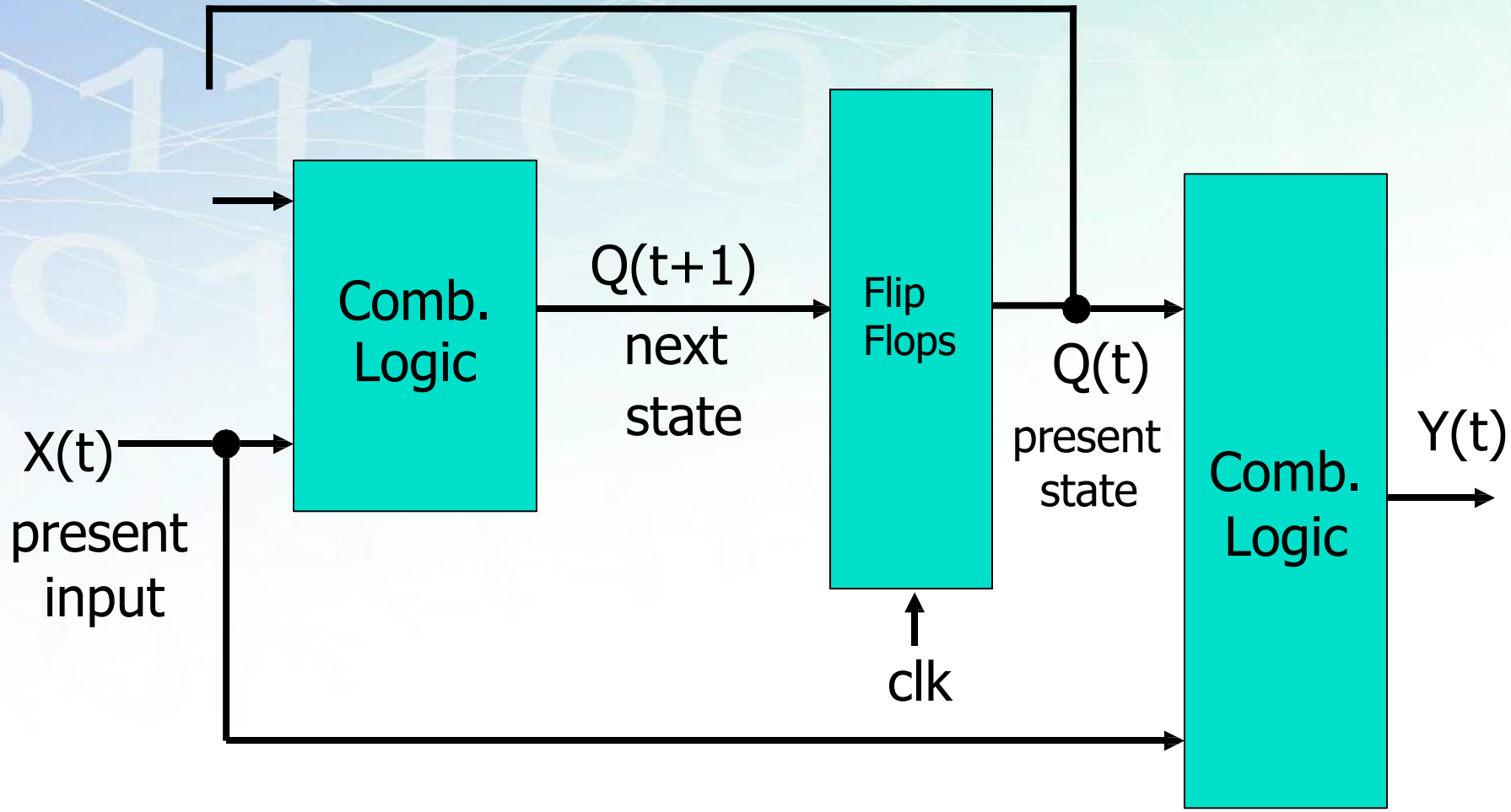


Mealy and More Models

- A general model of a Sequential Circuit has
  - **Inputs**
  - **Outputs**
  - **Internal States**
- There are two "**standard models**"
  - **Mealy Model**, known as Mealy Finite State Machine (FSM) (or Mealy machine)
  - **Moore Model** known as Moore FSM (or Moore Machine)

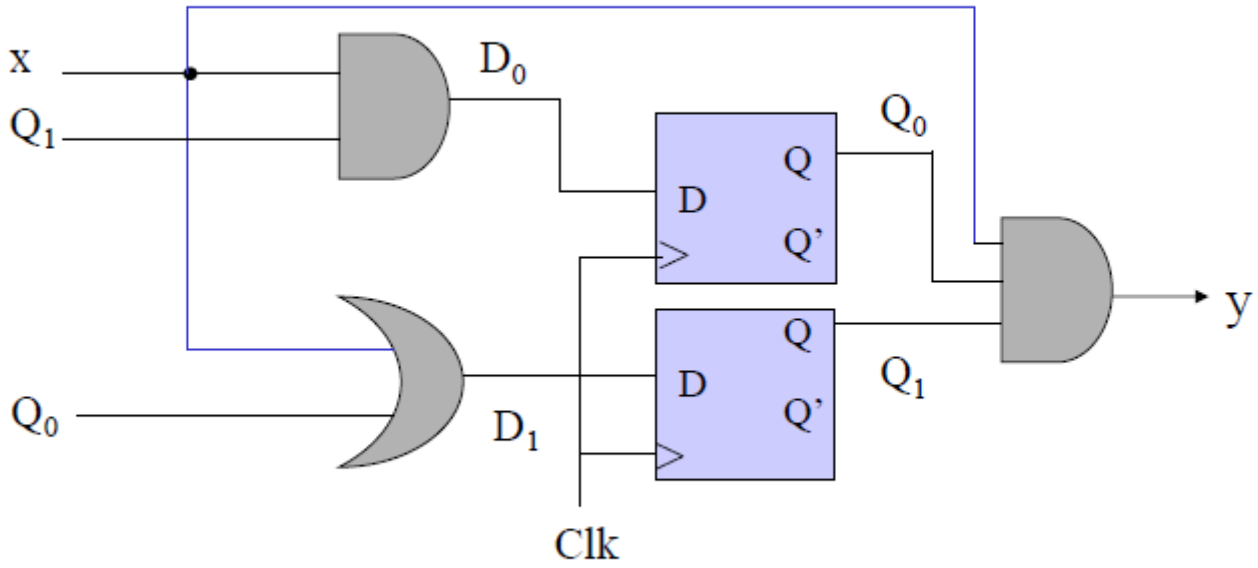
Mealy Machine

- **Output** based on **state** and **present input**



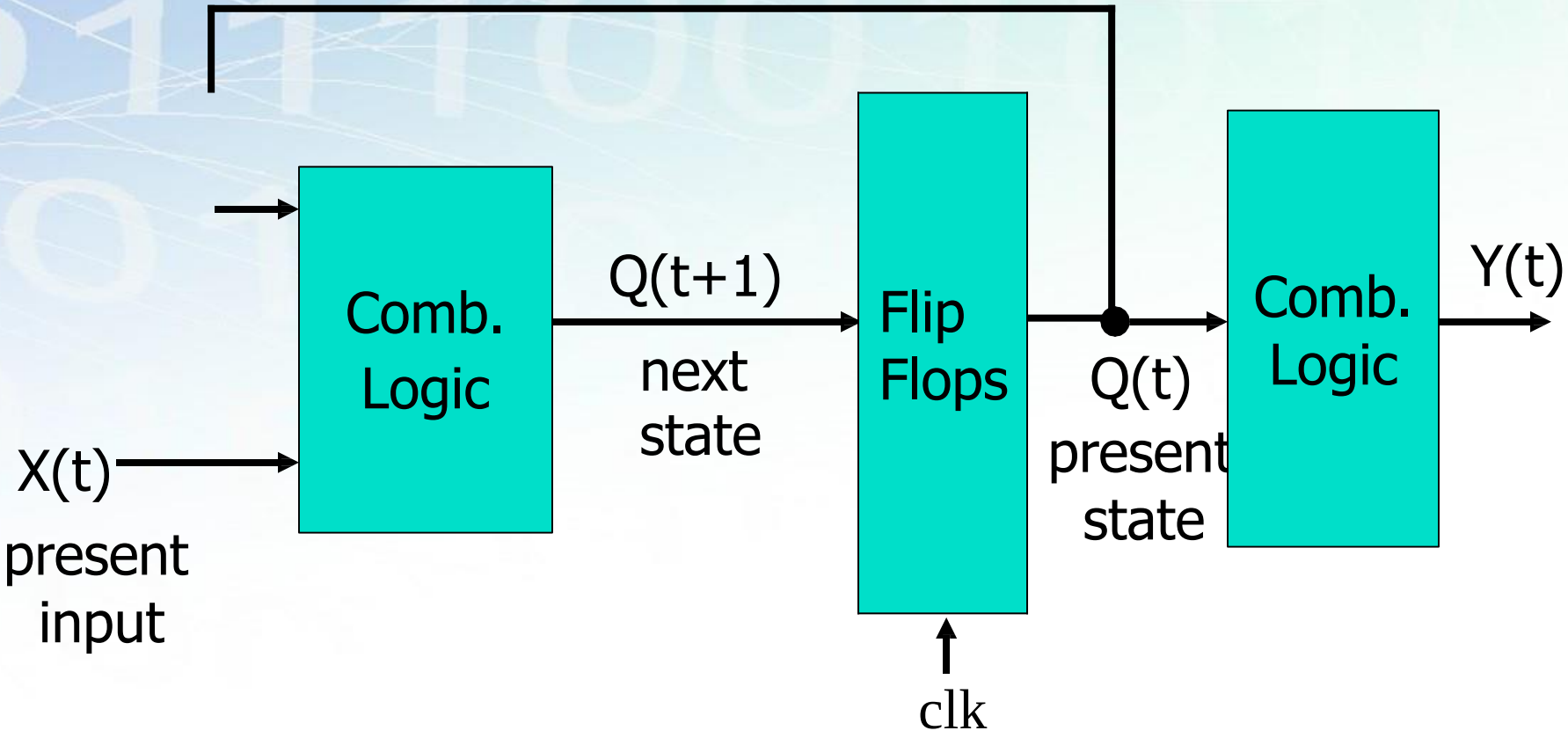
Mealy Machine (cont'd)

- Example:



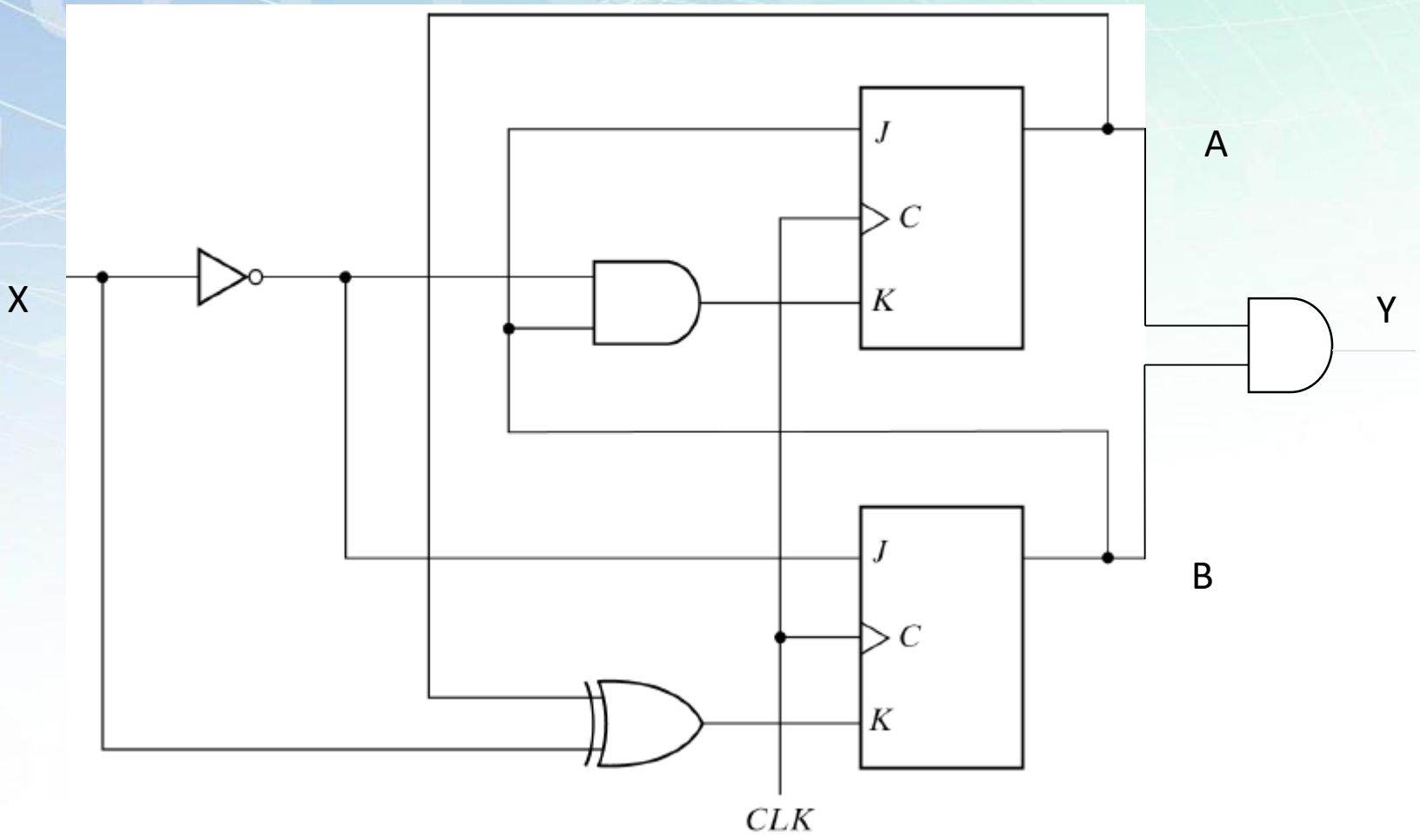
Moore Machine

- **Output** based on **state** only



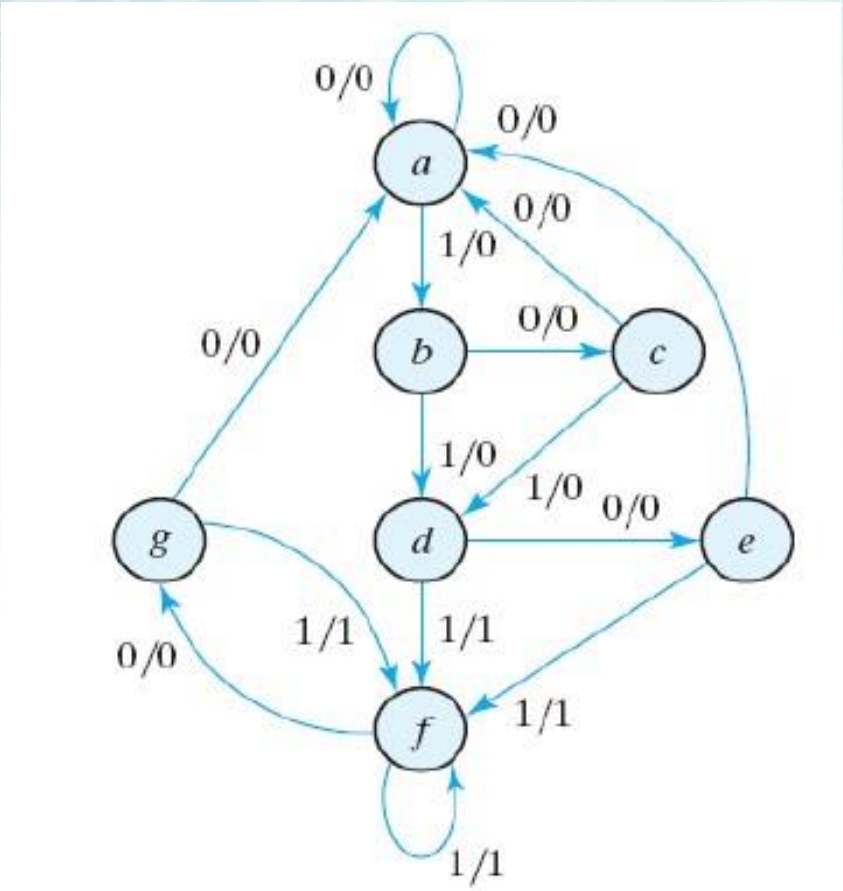
Moore Machine (cont'd)

- Example:



State Reduction

- In general, **reducing the number of states reduces the number of gates and flip-flops**
- Consider a sequential circuit with the following **State diagram**:



State Reduction (cont'd)

- Obtaining the State table from State diagram of the previous slide

State e and g are equivalent → have the same next states and same output when x=0, x=1

State Table

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

Reducing the State Table

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

Remove g and replace it by e in remaining next states

State Reduction (cont'd)

Reducing the State Table

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

f and d are equivalent

Reduced State Table

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

Sequential Circuit Design Procedure

Step 0: From the **problem specification,** create the **state diagram** which shows all of the states that the **FSM** can be in, and how to switch from one state to another

Step 1: State Table - transition (δ) and output (λ) functions

Make a **state table** based on the problem specification or state diagram. It may show the next states S^{n+1} and outputs Z^n as **functions** of present states S^n and inputs X^n (transition function: $S^{n+1} = \delta(S^n, X^n)$, output function $Z_n = \lambda(S_n, X_n)$.

State minimization = reduce the number of states in the table

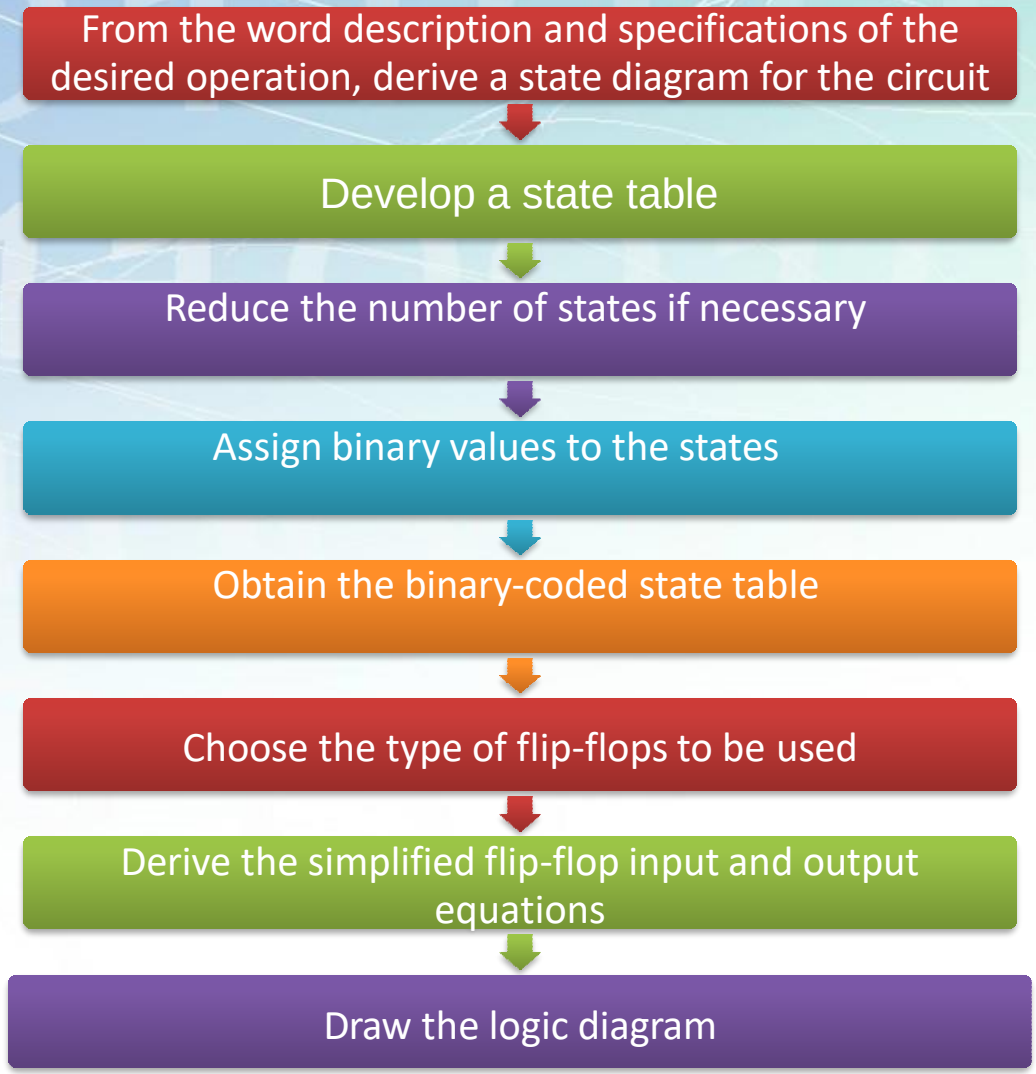
Step 2: **State Assignment** (Encoding). Assign binary codes to the states in the state table, if they were not assigned already. If you have N states, your binary codes will have at least $\lceil \log_2 N \rceil$ digits, and your circuit will have at least $\lceil \log_2 N \rceil$ flip-flops ➔ **Transition table** (i.e., binary-coded state table) = a state table with encoded states.

Step 3: Excitation Table and Equations. Choose the type of flip-flops to be used. For each flip-flop and each row of your transition table, find the flip-flop input values that are needed to generate the next state from the present state. You may use flip-flop excitation tables.

Step 4: Find simplified equations for the flip-flops inputs & the circuit outputs Z.

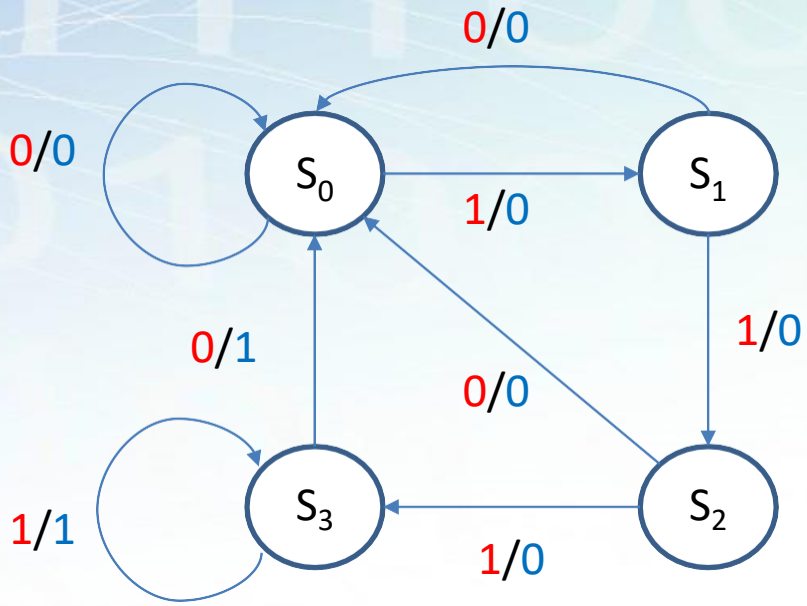
Step 5: Build the circuit!

Design Procedure



Example 1

- Design a circuit that **detects** a **sequence** of **3 or more consecutive 1's** in a string of bits coming through an input line



State diagram

- **State S_0** : zero 1s detected
- **State S_1** : one 1 detected
- **State S_2** : two 1s detected
- **State S_3** : ≥ three 1s detected
- Each state has 2 output arrows
- **x**: input
- **y**: output

Example 1 (cont'd)

Present State	Next State		y	
	x=0	x=1	x=0	x=1
S_0	S_0	S_1	0	0
S_1	S_0	S_2	0	0
S_2	S_0	S_3	0	0
S_3	S_0	S_3	1	1

State table

- Sequence of outputs, inputs, and flip-flop states enumerated in State table
- **Present state:** indicates current value of flip-flops
- **Next state:** indicates state after next clock edge
- **Output:** is output value on current clock edge

No equivalent states ⇒ no reduction in State diagram

Example 1 (cont'd)

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

Binary-coded State table

- Assign binary values to states, 2 bits needed to encode state:

AB
$S_0 = 00$
$S_1 = 01$
$S_2 = 10$
$S_3 = 11$

Example 1 (cont'd)

- Using **D flip-flops**: **2 D flip-flops** to represent the **4 states**

Present State		Input	Next State		FF Inputs		Output
A	B	x	A	B	D_A	D_B	y
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0
0	1	0	0	0	0	0	0
0	1	1	1	0	1	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	1	1	0
1	1	0	0	0	0	0	1
1	1	1	1	1	1	1	1

State table and D flip-flop inputs

D flip-flop Excitation Table

Q	Q_{next}	D
0	0	0
0	1	1
1	0	0
1	1	1

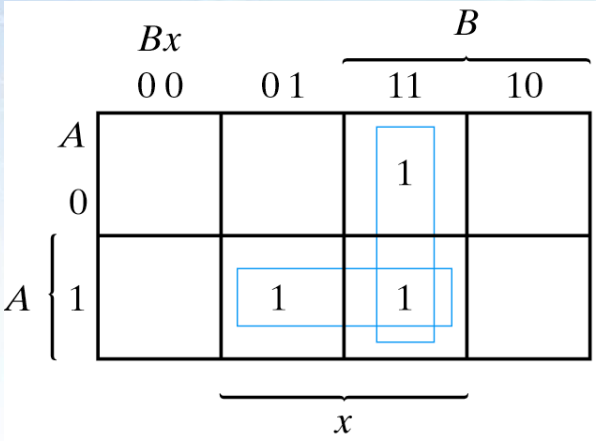
$A(t+1)=D_A(A,B,x)=\Sigma m(3,5,7)$

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

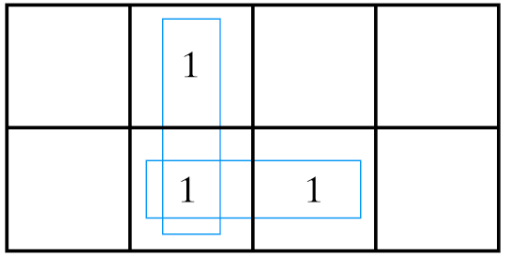
State or Characteristic equations

Example 1 (cont'd)

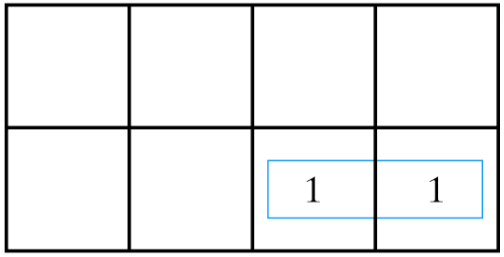
- Create K-map directly from State table (3 columns = 3 K-maps)
- Minimize K-maps to find SOP representations
- Separate circuit for each next state and output value



$D_A = Ax + Bx$



$D_B = Ax + B'x$



$y = AB$

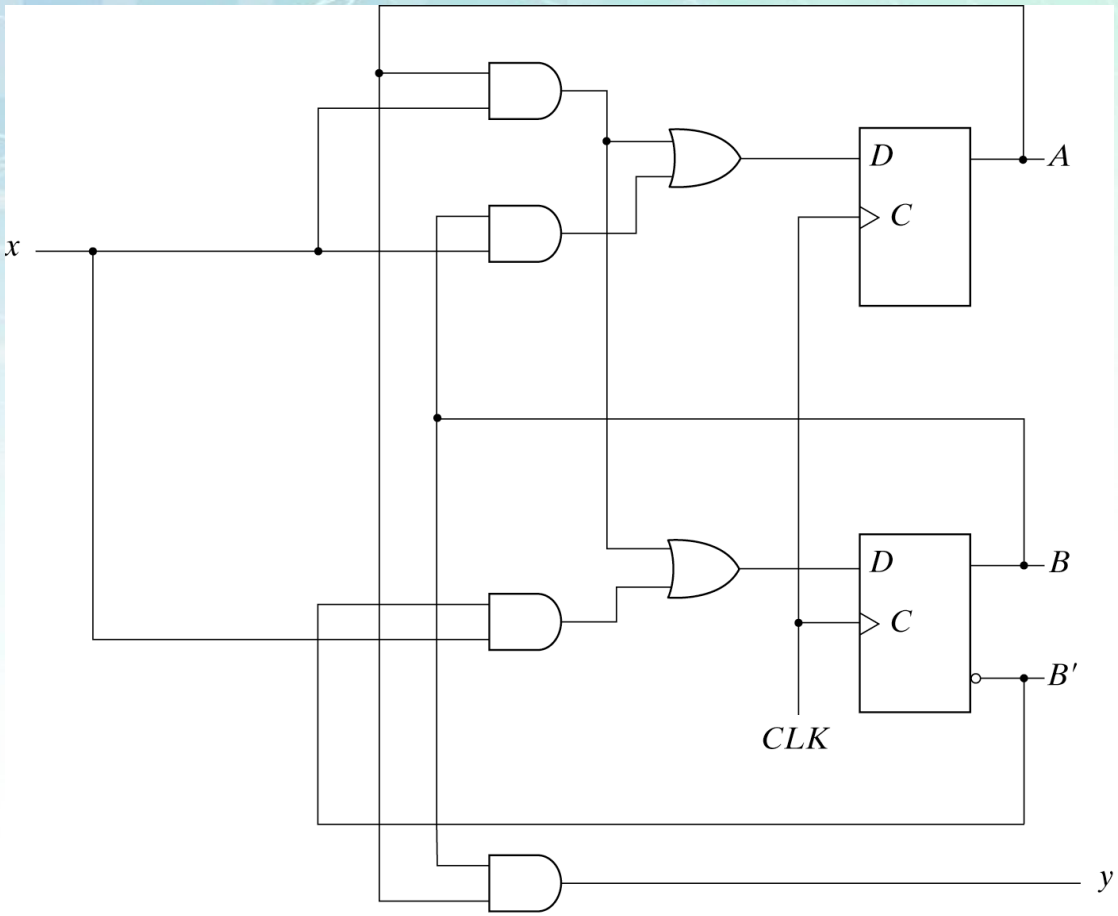
FF input equations Maps for Sequence Detector

Output equation

K-maps for sequence detector

Example 1 (cont'd)

- Output value (**y**) is **function of state**
  - This is a **Moore machine**



Logic Diagram of Sequence Detector