Digital Logic Circuit (SE273 – Fall 2020)

Lecture 8: Sequential Circuits Design

Jaesok Yu, Ph.D. (jaesok.yu@dgist.ac.kr)

Assistant Professor

Department of Robotics Engineering, DGIST

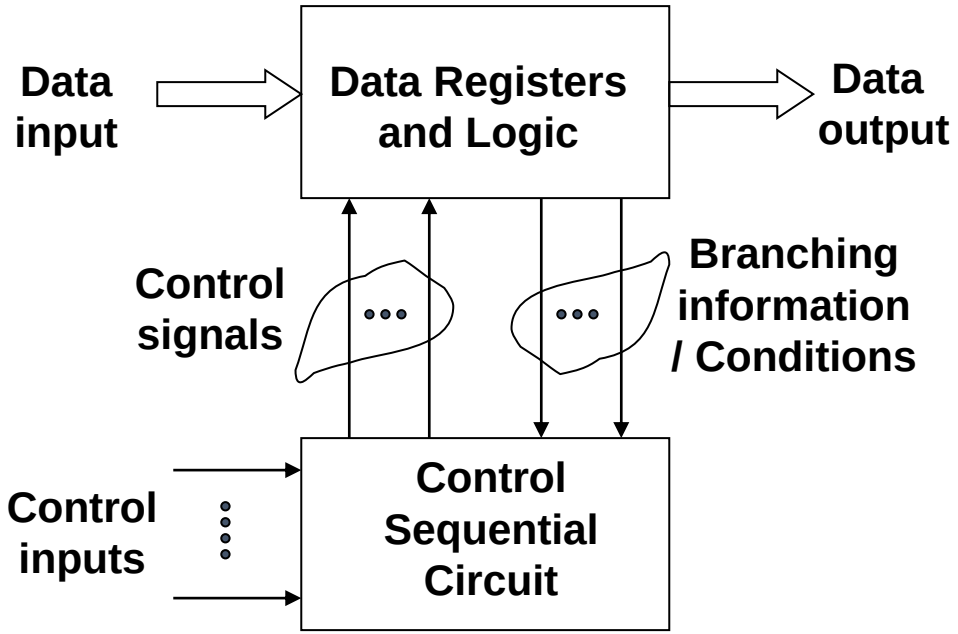
Goal

- Learn how to analyze sequential circuits
  - State equations, state table, state diagram
  - Finite state machines

- Learn how to design sequential circuits
  - Sequential circuit design procedure
  - Design sequential circuits in Verilog

General Digital Systems

- General Digital Systems
  - Most digital systems can be partitioned into a control section and data flow section containing registers and logic
    - Data flow section: Register transfer operations > Register-Transfer-Level (RTL) design
    - Digital sequencer > Finite State Machine (FSM)/Algorithmic State Machine (ASM) design



General Digital Systems

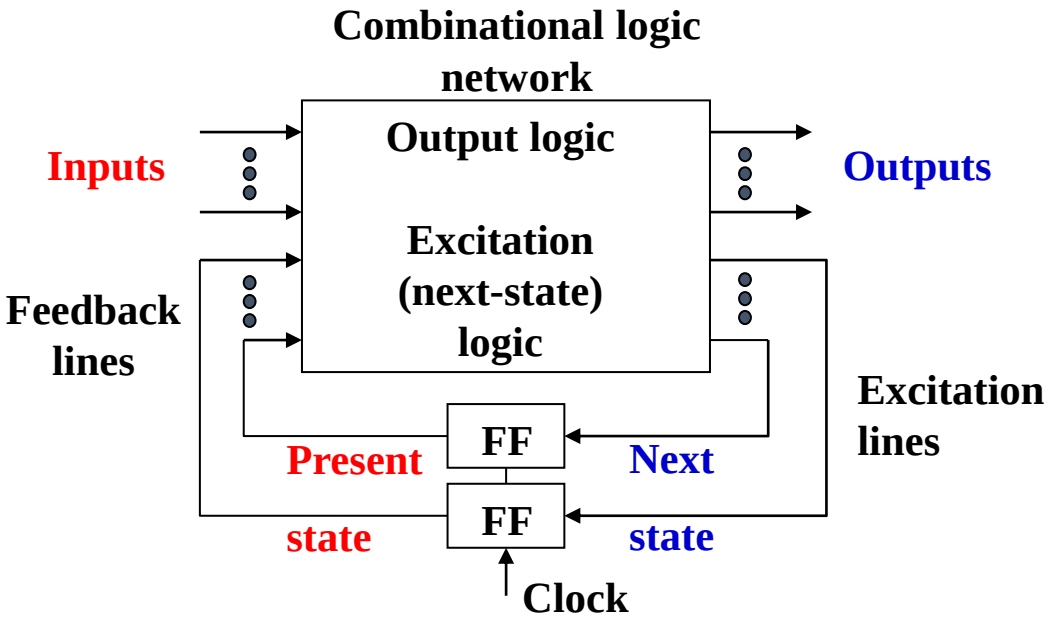
- Model for a general synchronous sequential-logic circuit

Mealy machine

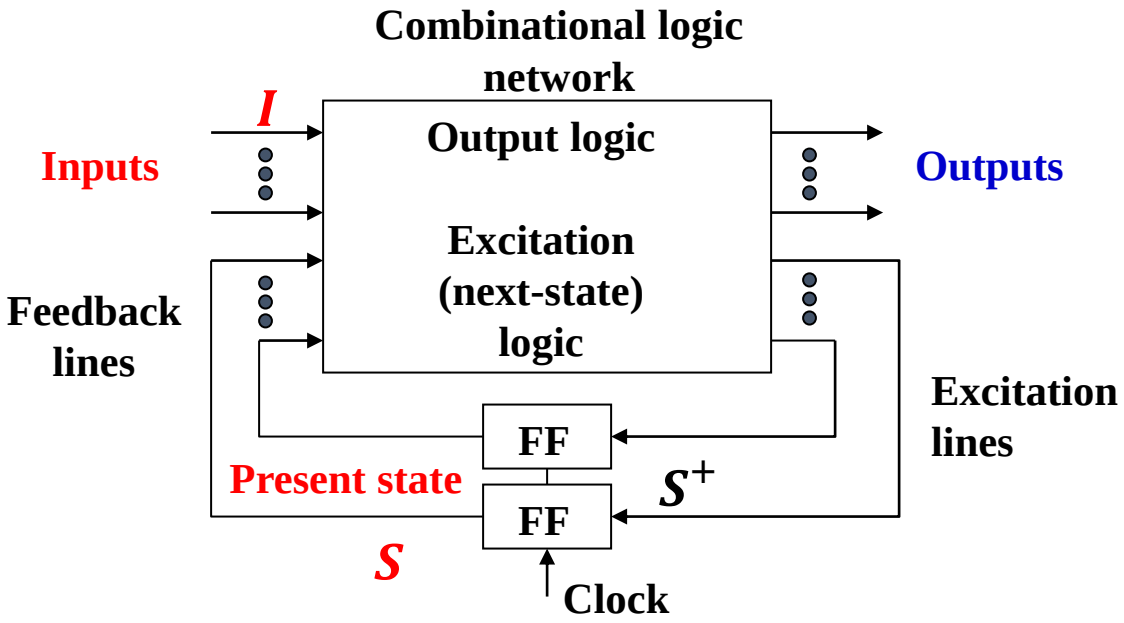
$$Outputs = f(inputs, present\ states)$$

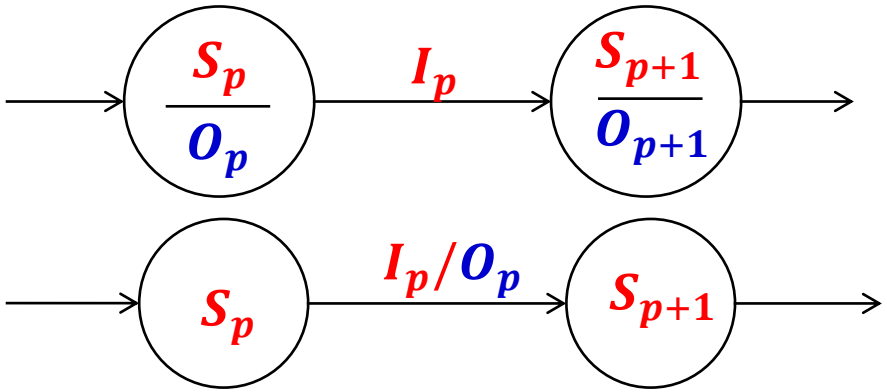
Moore machine

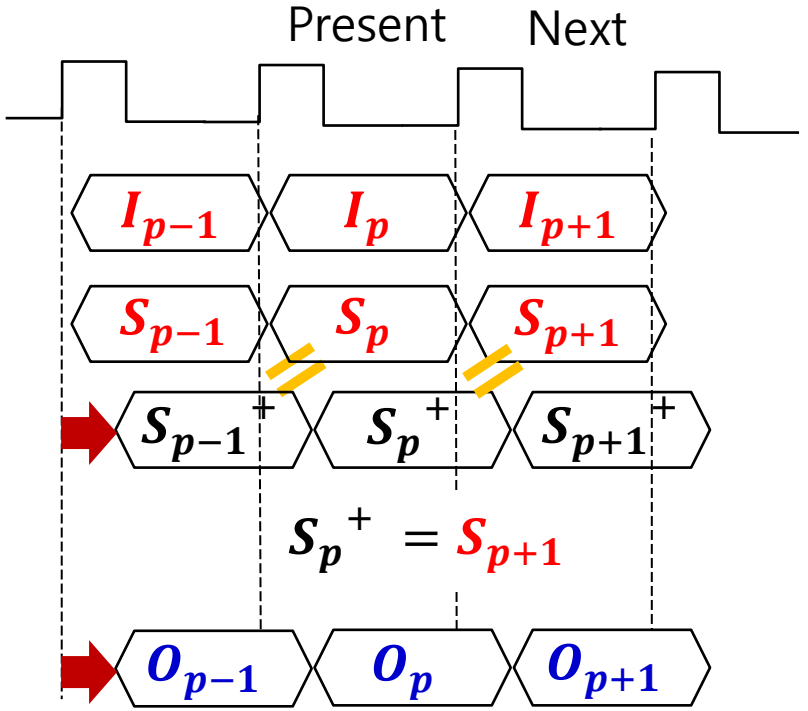
$$Outputs = f(present\ states)$$



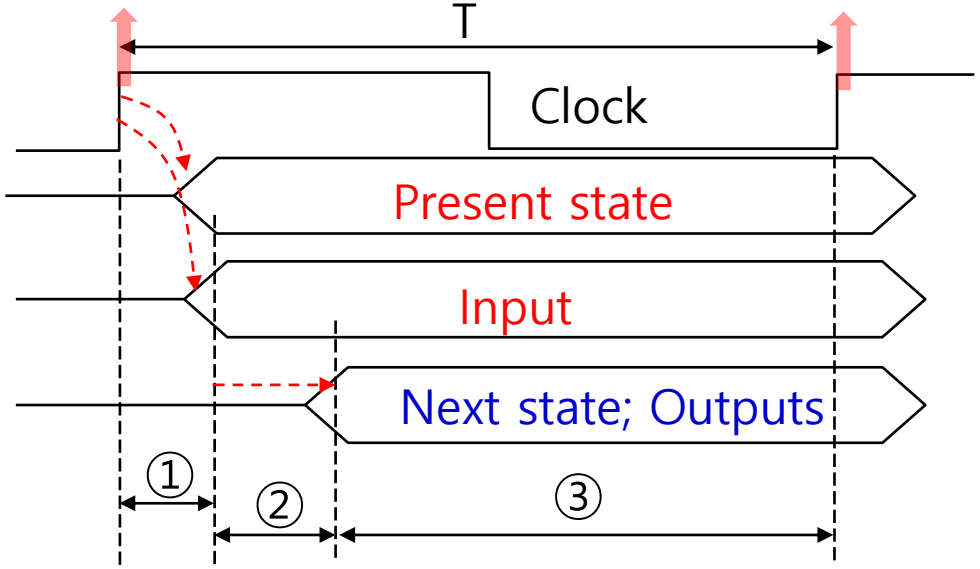
Clock cycle and Update of State / Output







Minimum Cycle Time



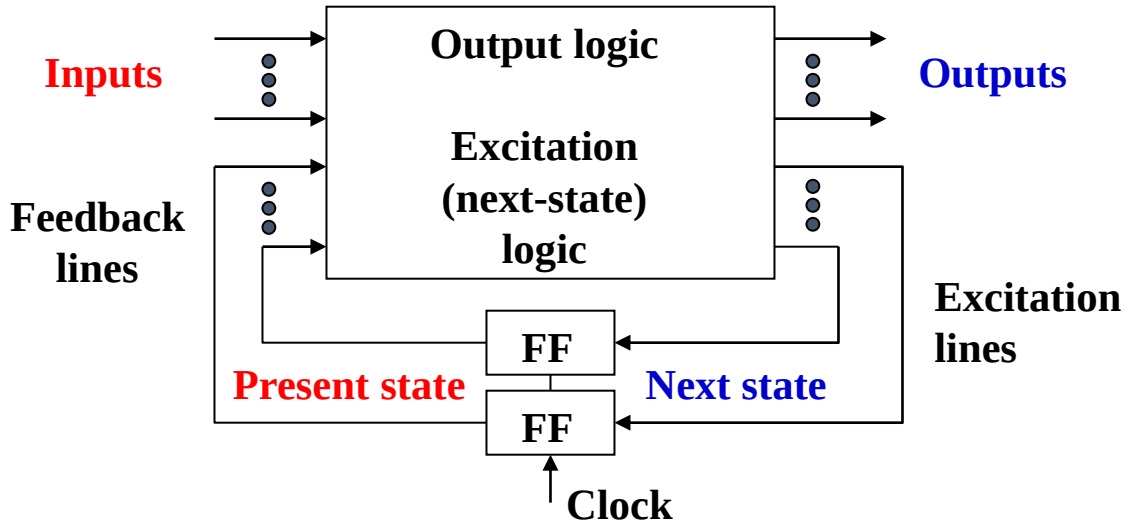
$$T \geq ① + ② + ③$$

① Clock to input stable delay

② Propagation delay of next state generation logic

$$Next\ state = f(inputs, present\ states)$$

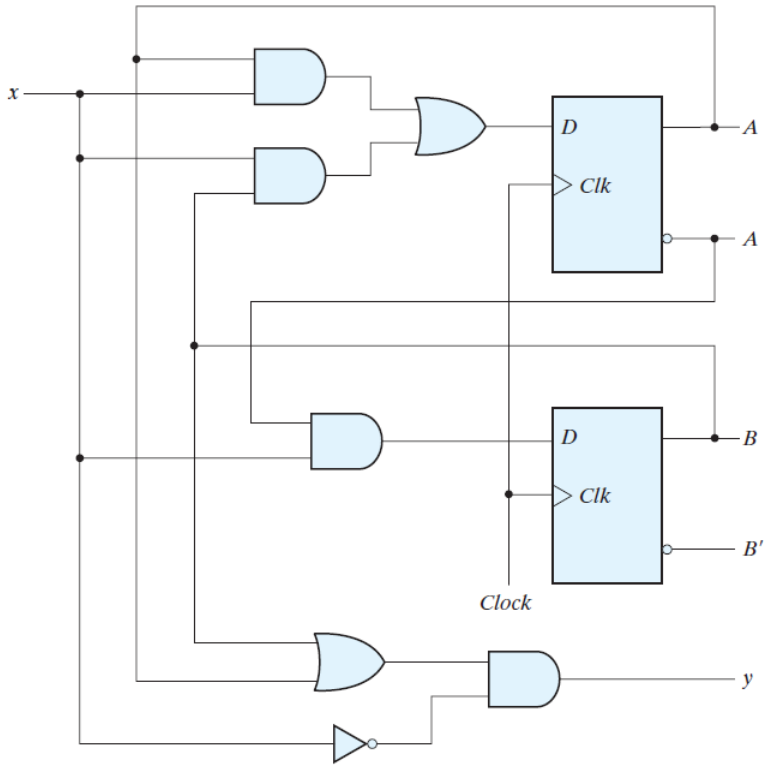
③ Setup time of state memories



Analysis

State Equations

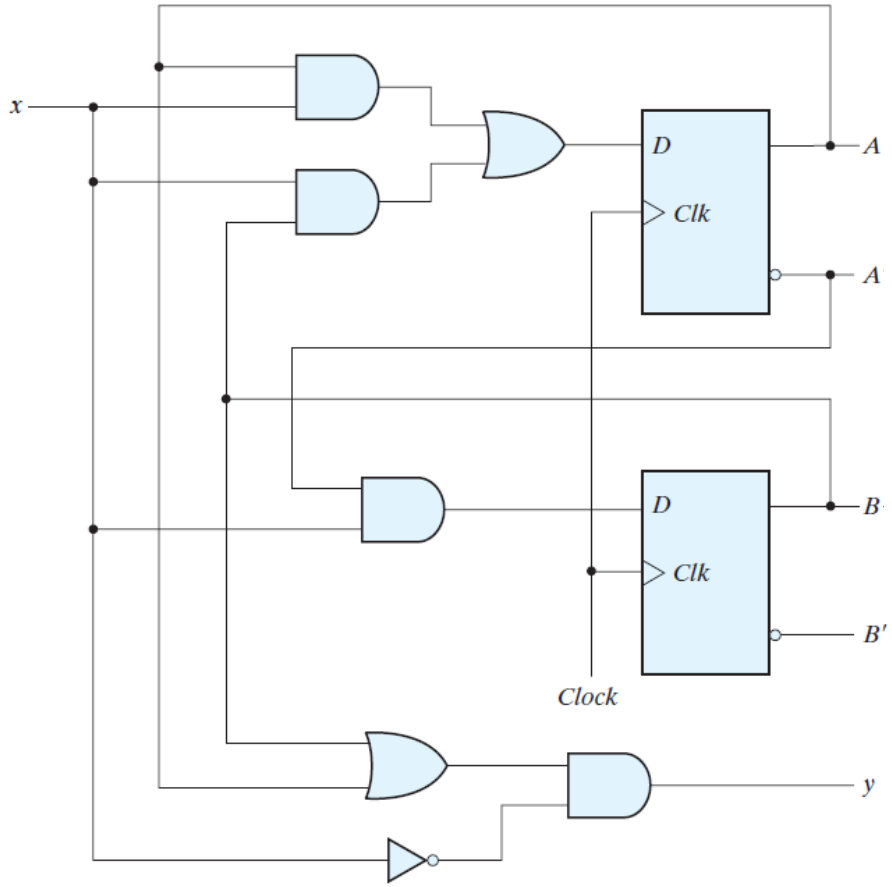
- A clocked sequential circuit can be described by state equations
  - Also called a transition equation
  - It specifies the next state as a function of the present state and inputs



How can we describe this circuit?

0 detector

- Makes the output ‘high’ when a 0 is detected in a stream of 1’s



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

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

State Table

- The time sequences of inputs, outputs, and FF states can be enumerated in a state table (transition table)

$$A(t+1) = Ax + Bx$$

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

$$y = Ax' + Bx'$$

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	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

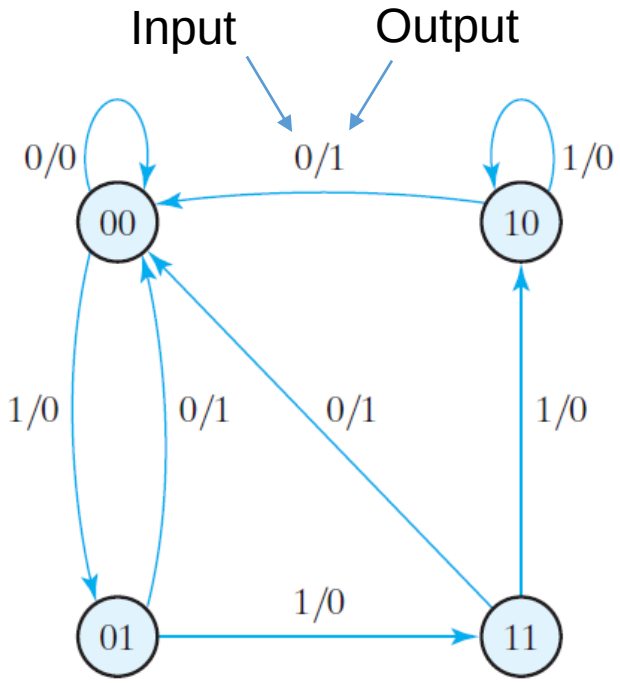
Alternative form of a state table

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

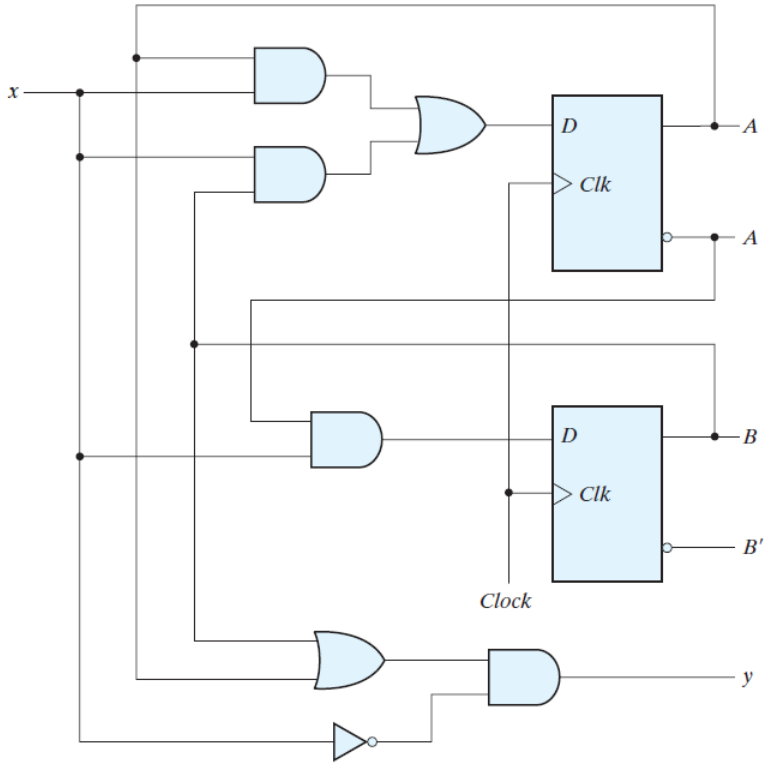
- We can graphically present the sequential circuit as a state diagram
  - State: circle
  - Directed line: transitions btw states

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



Flip-Flop Input Equations

- A set of Boolean equations that generates the inputs to FFs
  - Type of flip-flops
  - A list of Boolean equations for combinational circuits



$$D_A = Ax + Bx$$
$$D_B = A'x$$

$$y = (A + B)x'$$

Analysis with D Flip-Flops

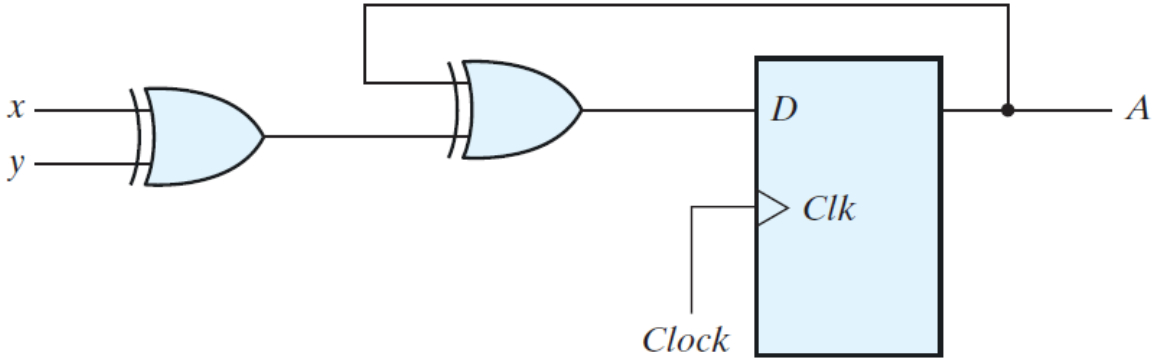
- The circuit we want to analyze is as follows:

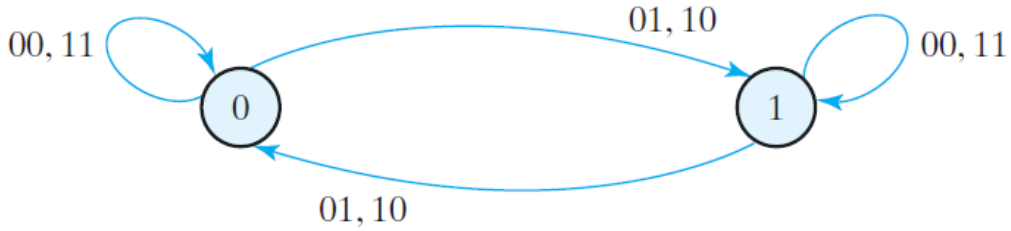
$$D_A = A \oplus x \oplus y$$

D_A symbol implies a D FF with output A

State equation is identical to input equation

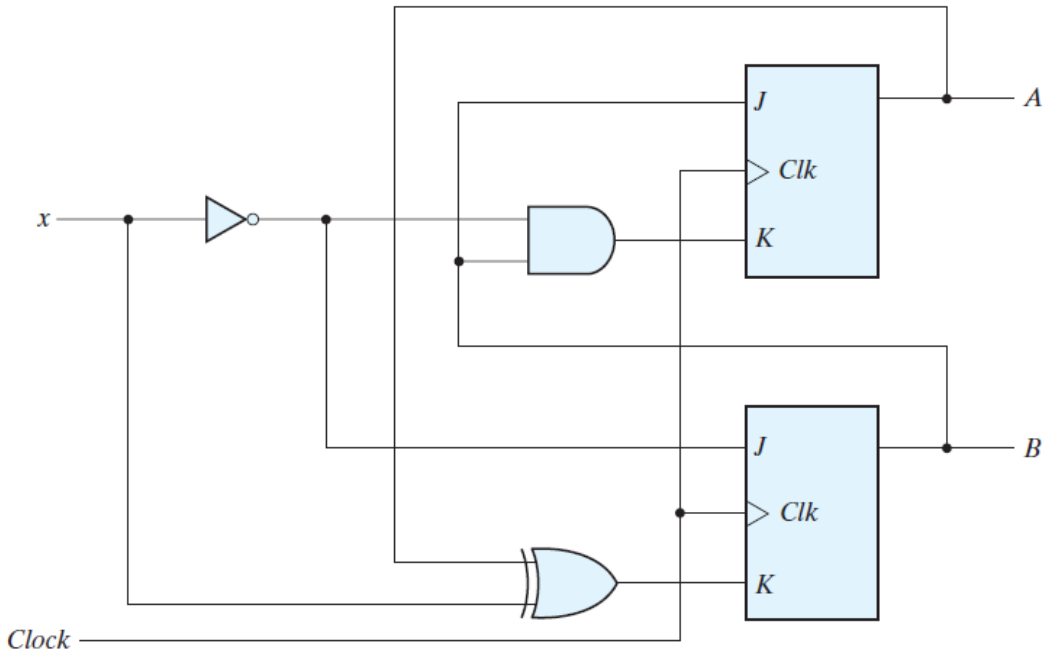
Present state	Inputs		Next state
A	x	y	A
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1





Analysis with JK Flip-Flops

- The circuit we want to analyze is as follows:



Need to refer to the corresponding characteristic table

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

Analysis with JK Flip-Flops

- The circuit we want to analyze is as follows:

$$J_A = B \quad K_A = Bx'$$
$$J_B = x' \quad K_B = A'x + Ax' = A \oplus x$$

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

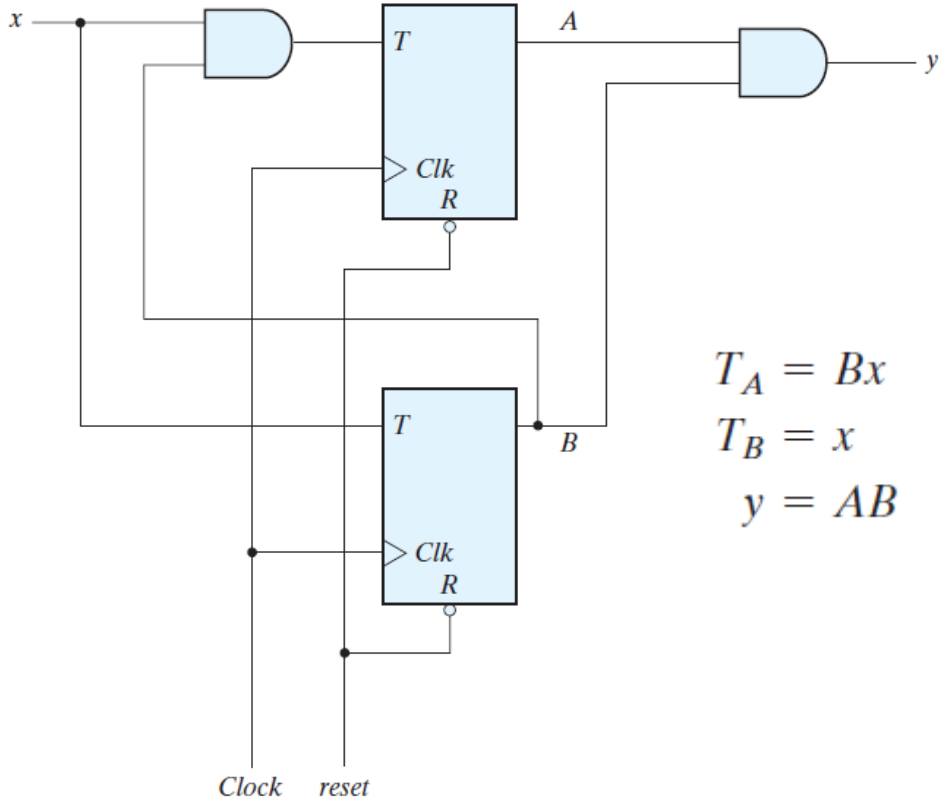
$$A(t+1) = JA' + K'A$$
$$B(t+1) = JB' + K'B$$

$$A(t+1) = A'B + AB' + Ax$$
$$B(t+1) = B'x' + ABx + A'Bx'$$

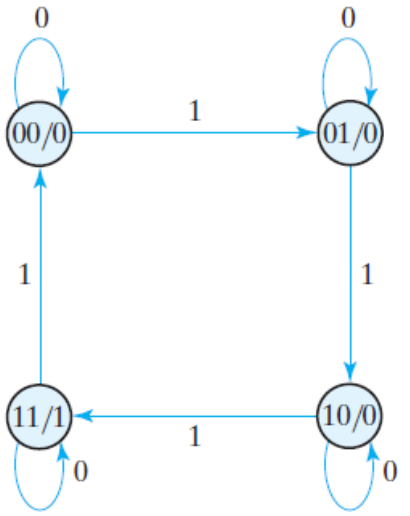
Analysis with T Flip-Flops

- The circuit we want to analyze is as follows:

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



$$T_A = Bx$$
$$T_B = x$$
$$y = AB$$

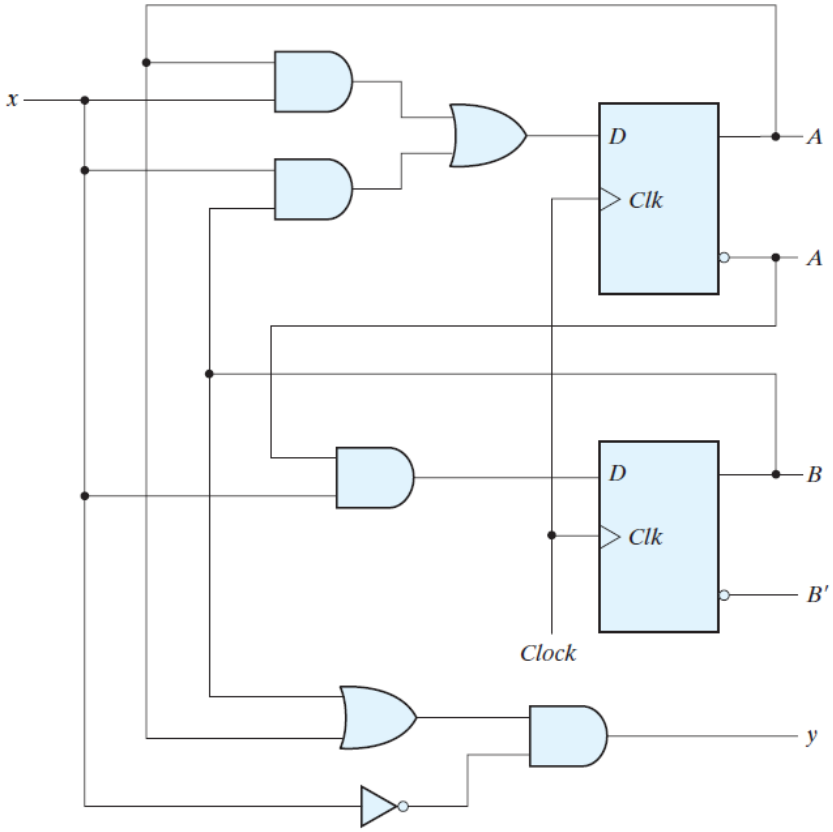


Present State		Input	Next State		Output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	1	0	0
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	1

Finite State Machine (FSM)

- There are two types of FSMs (we've already covered)

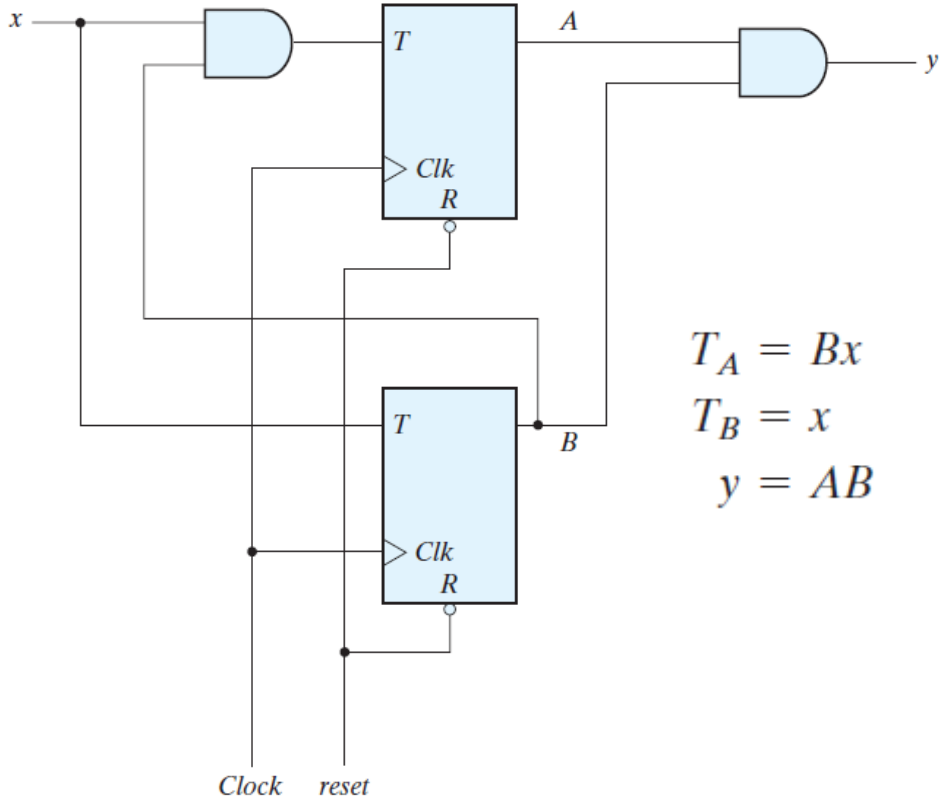
Example w/ D FF



$$D_A = Ax + Bx$$
$$D_B = A'x$$

$$y = (A + B)x'$$

Example w/ T FF



$$T_A = Bx$$
$$T_B = x$$
$$y = AB$$

Design of Sequential Circuits

Overview of Sequential Circuit Design

- For massive ICs, we rely on automated synthesis tools
  - As a sequential building block, synthesis tools use D flip-flop
  - A sequential circuit requires a **state table** for specification (or **state diagram**)
  - It consists of flip-flops and combinational gates
  - # of FFs = determined by the # of states

Design Procedure

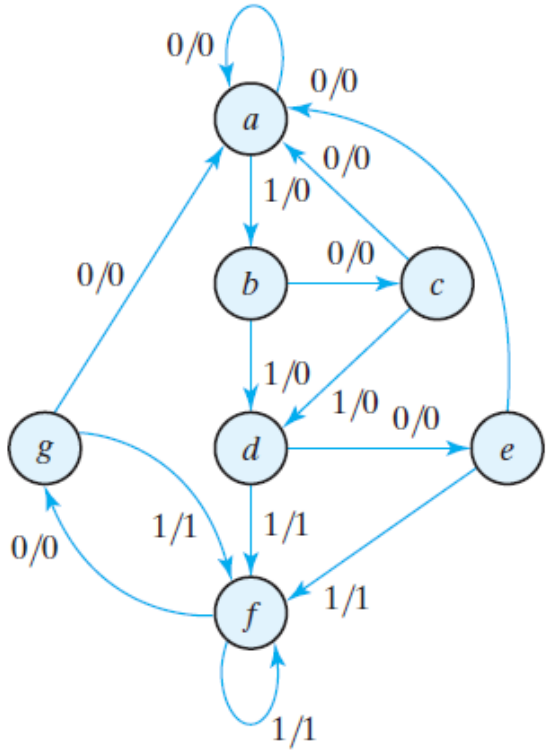
- A list of recommended steps:

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.

State Reduction

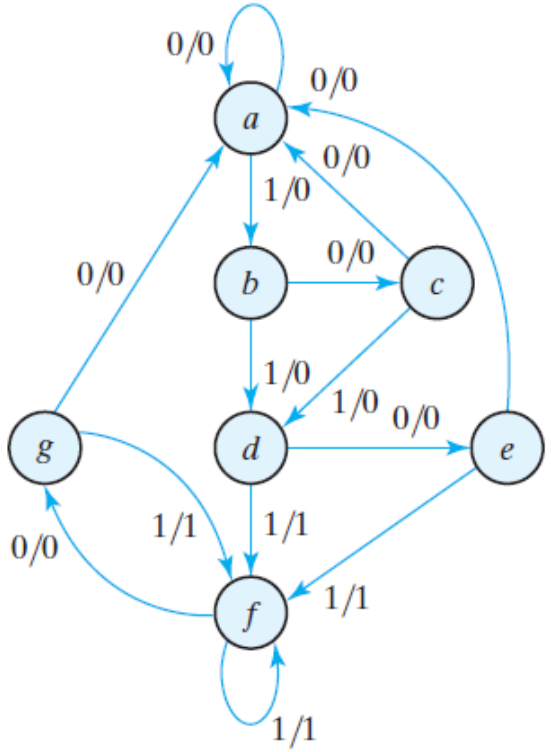
- State reduction allows us to design a seq. circuit w/ fewer flip-flops
  - m flip-flops → 2^m states
  - But sometimes reduced # of FFs may require more combinational gates to realize its next state and/or output

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



State Reduction

- We need the state table for the reduction
  - It is more convenient to look at state table rather than a diagram
  - We look for two present states that go to the same next state and have the same output for both input combinations



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

- We may perform state reduction multiple times

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

Original

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	

Reduced

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	

Design Procedure

- A list of recommended steps:

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.

State Assignment

- To design a sequential circuit w/ physical components, it is necessary to assign **unique binary codes** to the states
  - With n bits, we can assign codes to 2^n states

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

	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*	*d*	0	1
e	*a*	*d*	0	1

	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

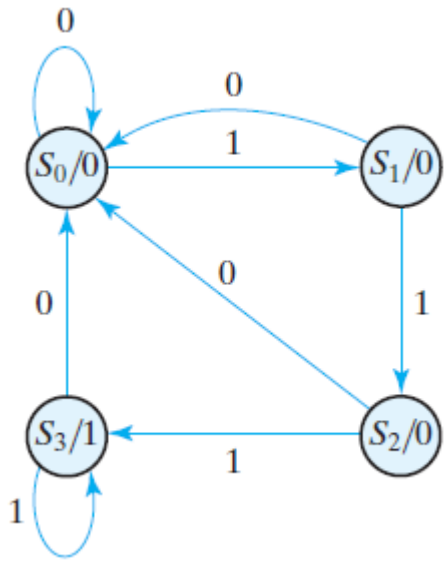
Design Procedure

- A list of recommended steps:

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.

Synthesis Using D Flip-Flops

- Suppose we wish to design a circuit that detects a sequence of three or more consecutive 1's
  - Input is a serial bit stream
  - We can derive a state table from a given state diagram



State Table for Sequence Detector

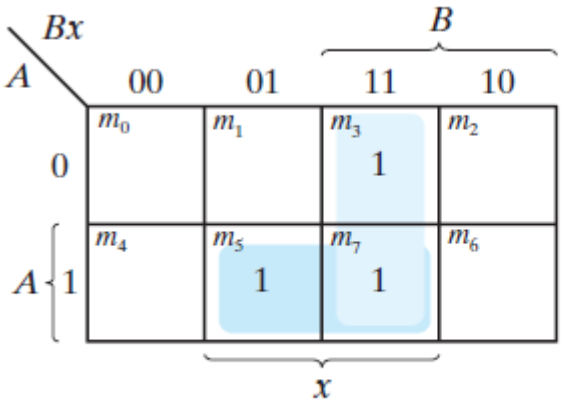
Present State		Input	Next State		Output
A	*B*	*x*	*A*	*B*	*y*
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

We choose two D flip-flops (represent four states)

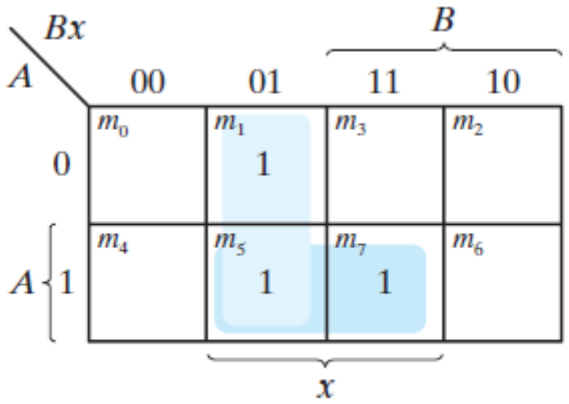
Label outputs as A and B

Flip-Flop Input Equations w/ D FF

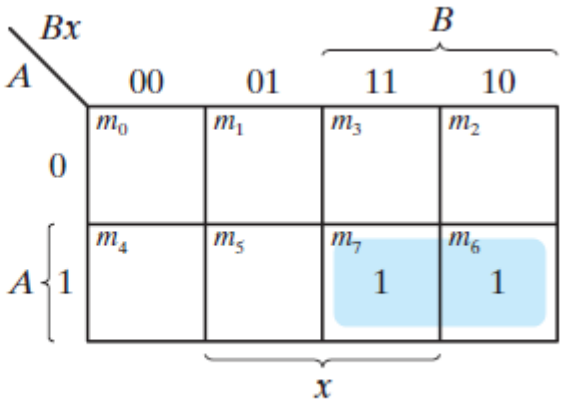
- It can be obtained directly from the next-state columns of A and B
  - Express in sum-of-minterms form



$D_A = Ax + Bx$



$D_B = Ax + B'x$



$y = AB$

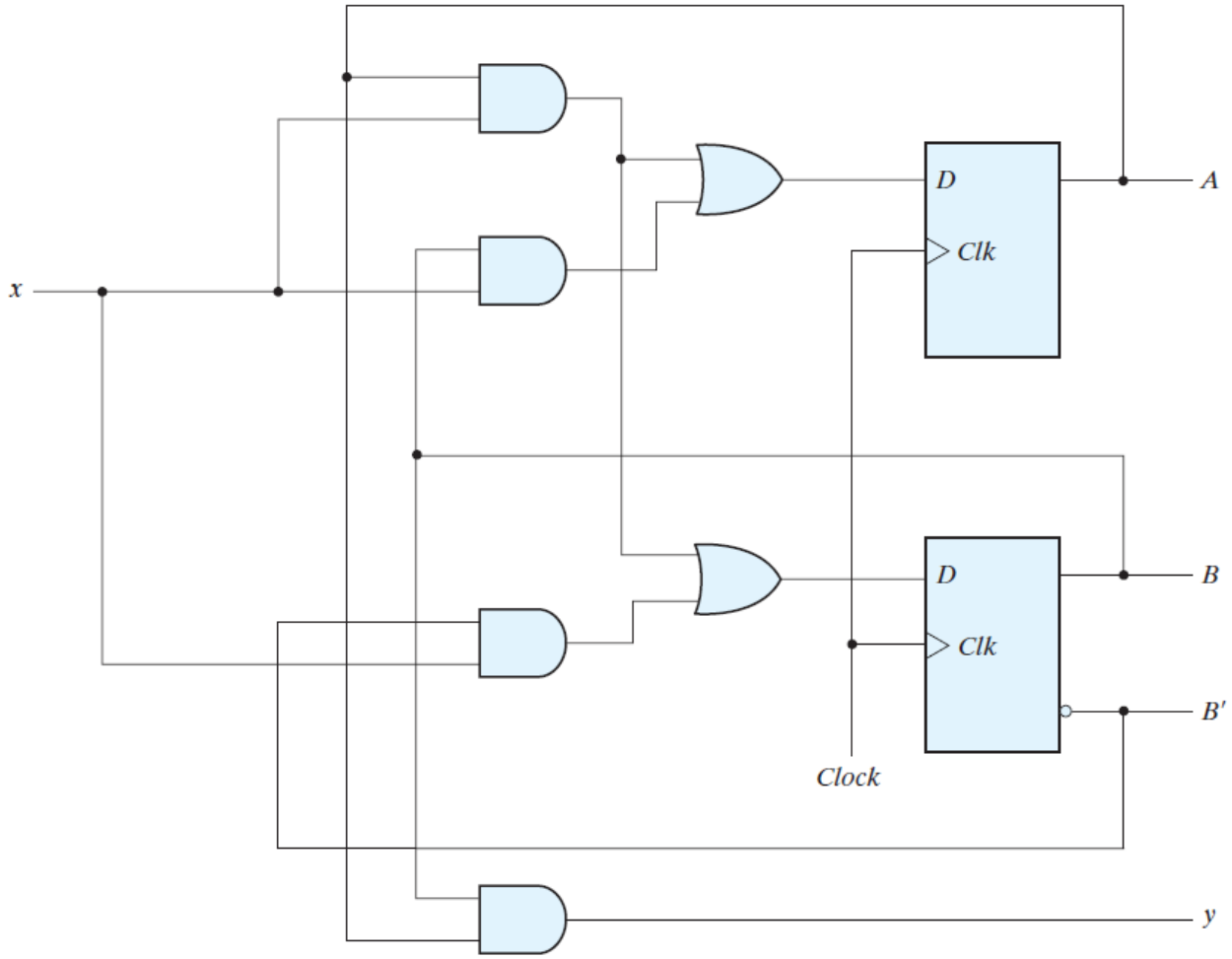
$A(t+1) = D_A(A, B, x) = \Sigma(3, 5, 7)$

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

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

Schematic of a Sequence Detector

- The schematic of the designed sequential circuit is



What type of FSM??

Synthesis Using JK Flip-Flops

- Manual synthesis procedure for sequential circuits w/ JK flip-flops is the same as with D FFs
  - Except that input equations must be evaluated from the present-state to the next-state transition (excitation table)

FF Excitation Table

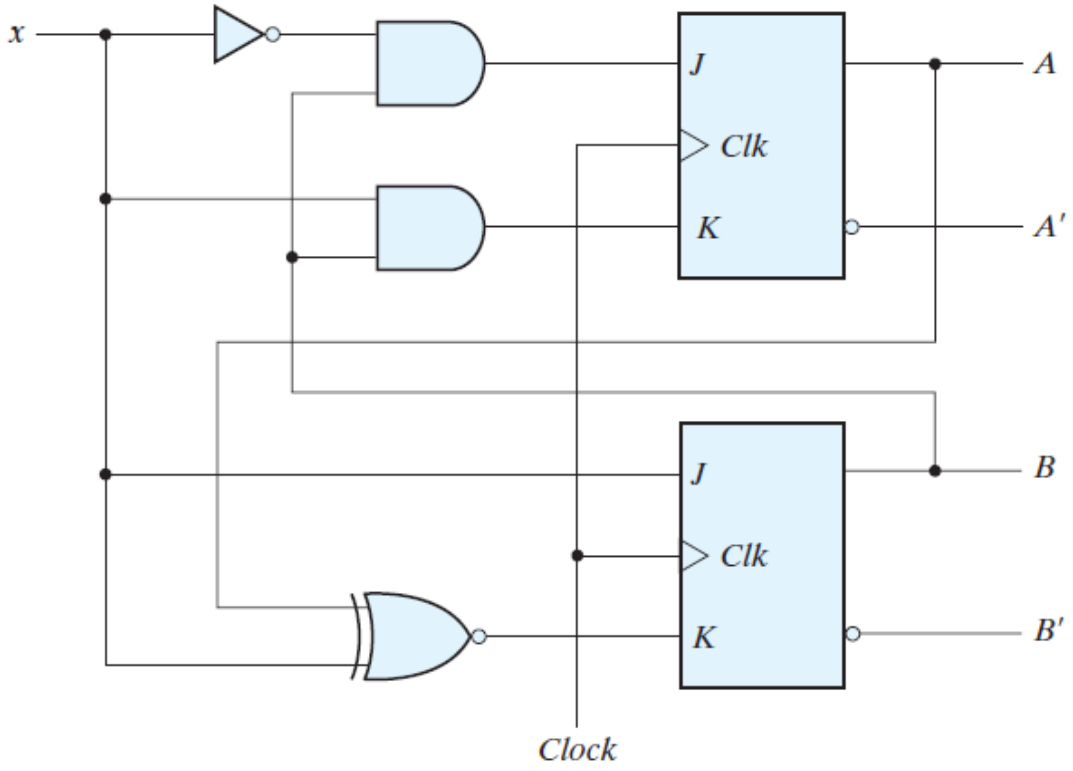
$Q(t)$	$Q(t+1)$	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

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 Input Equations w/ JK FF

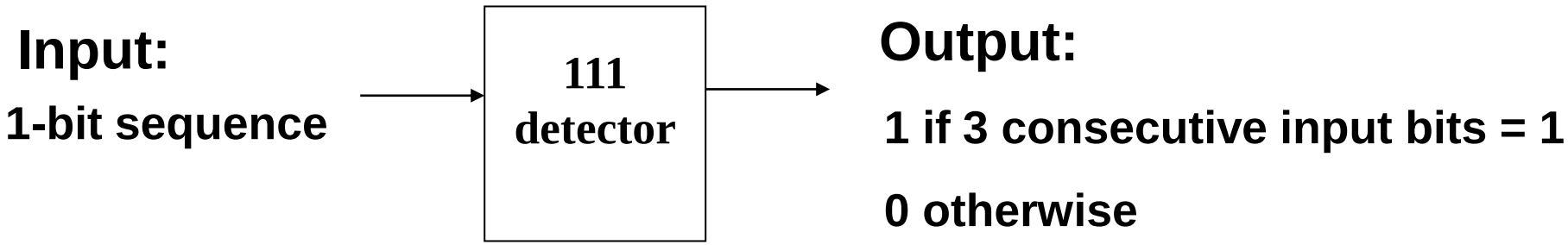
- Now, we can derive input equations as a function of present state A, B, and input x

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

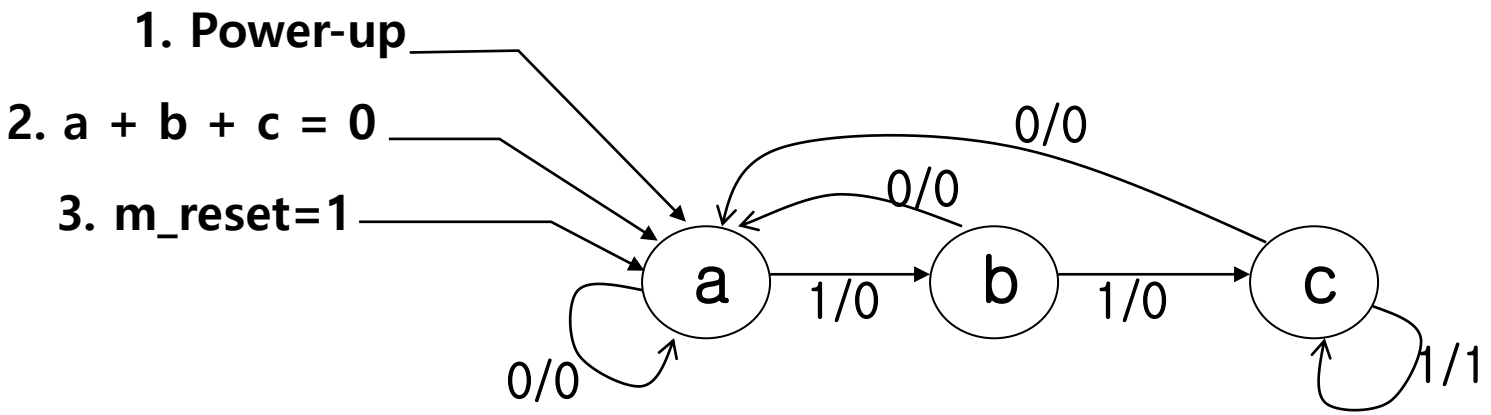


More example: 111 detector

- Design a 111 detector



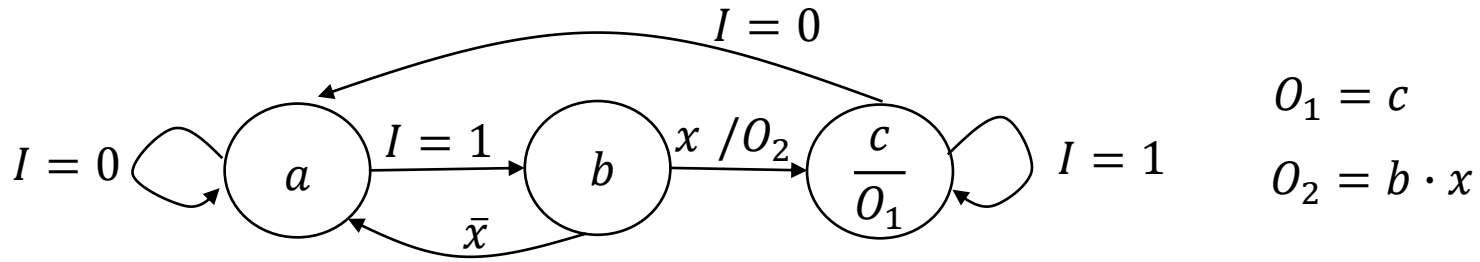
More example: 111 detector



1. Power on Reset
2. Other conditions: Not defined in any states.
   - Initial state
   - Prevent undefined state
3. Reset to initial or pre-defined state
   - Synchronous reset
   - Asynchronous reset

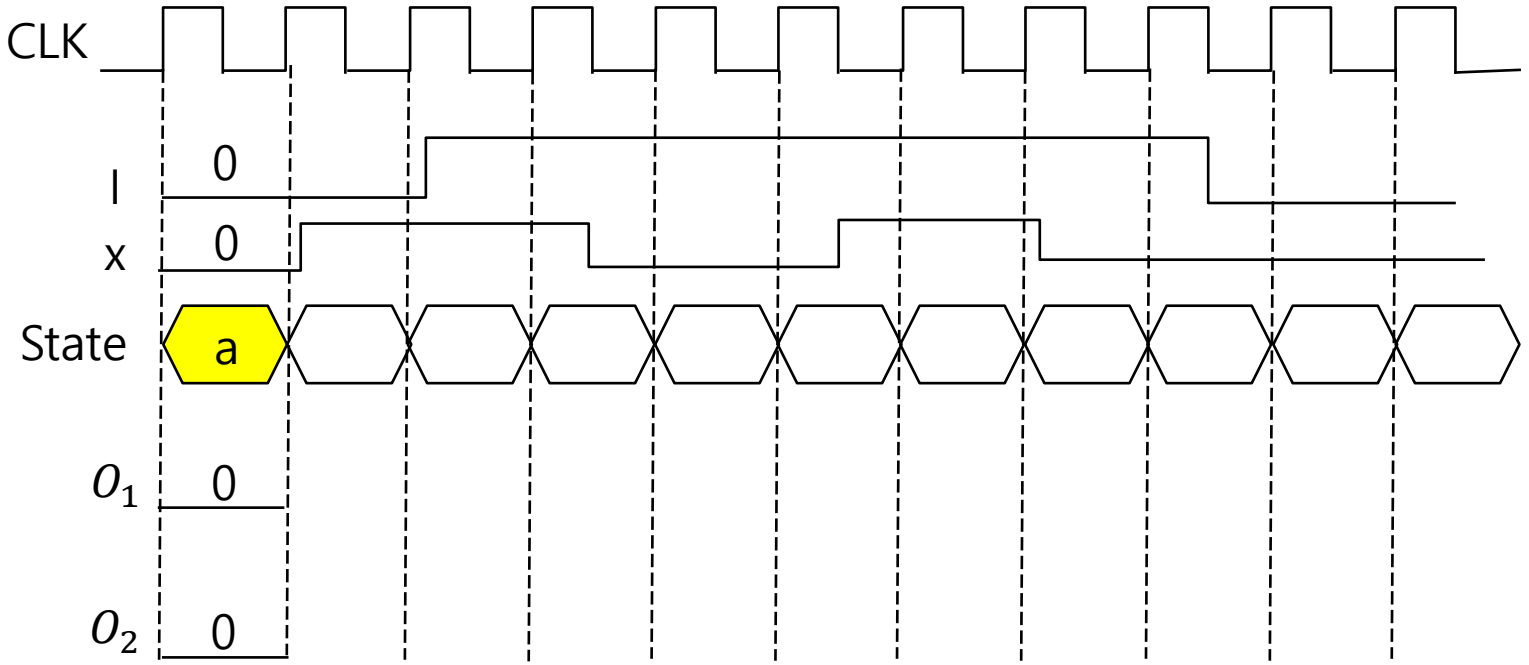
Generally Not Defined In the State diagram

More example: 111 detector

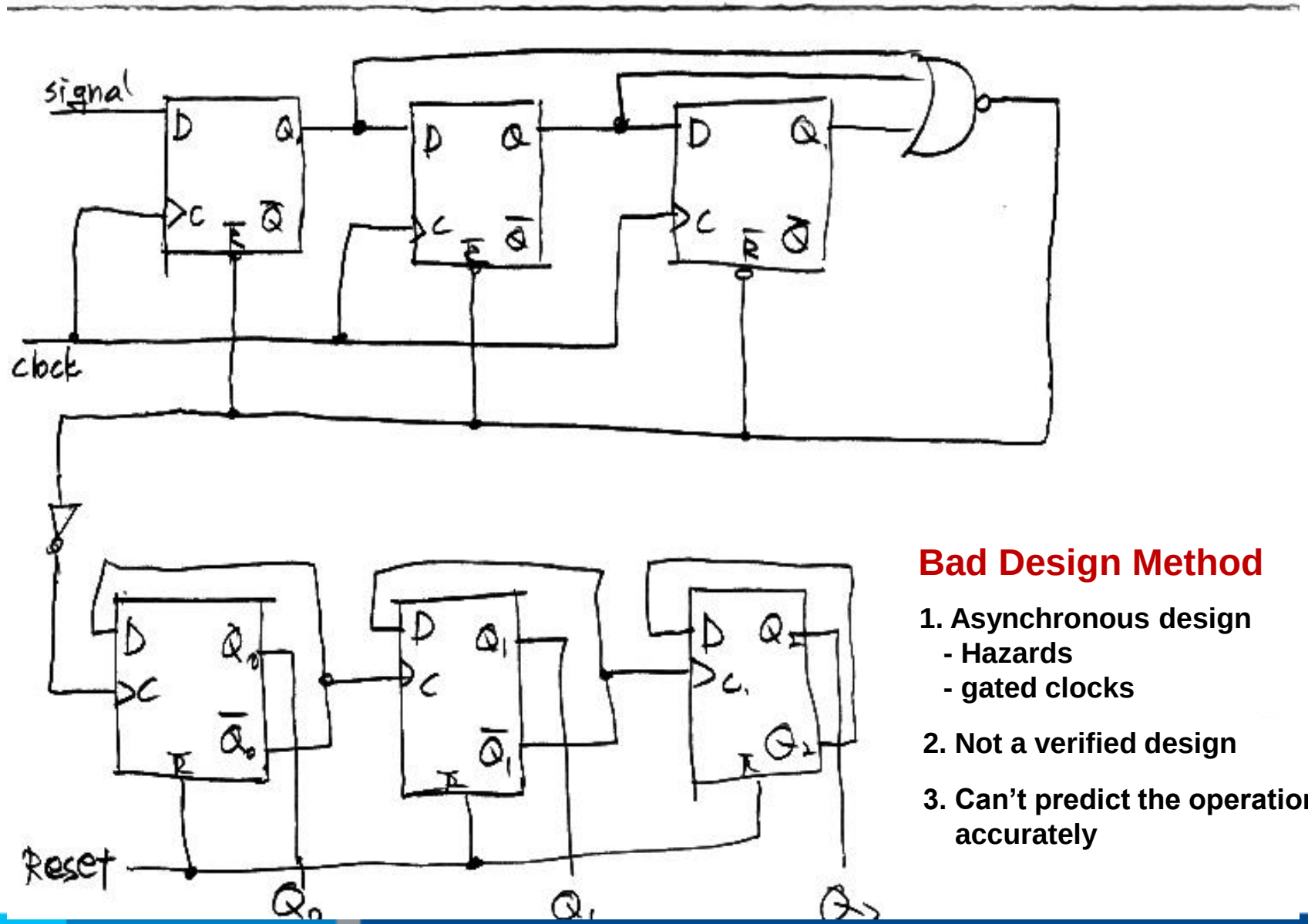


$O_1 = c$

$O_2 = b \cdot x$



Bad design example

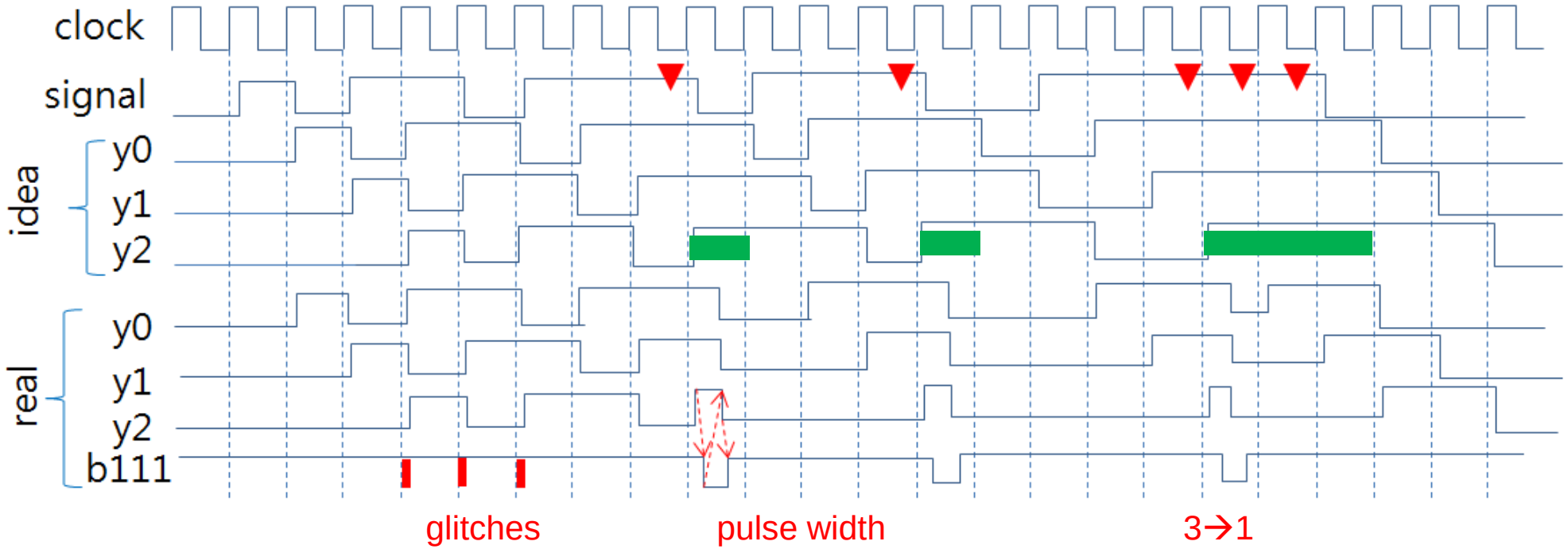


Bad Design Method

1. Asynchronous design
   - Hazards
   - gated clocks
2. Not a verified design
3. Can’t predict the operation accurately

Bad design example

Design verification by checking timing diagram for a test vector



- glitches
- does not work when clock rate is very high
- consecutive 111 patterns are not detected

Design procedure

✓ Draw the state diagram ⟶ **State table reduction**

✓ Write the state table

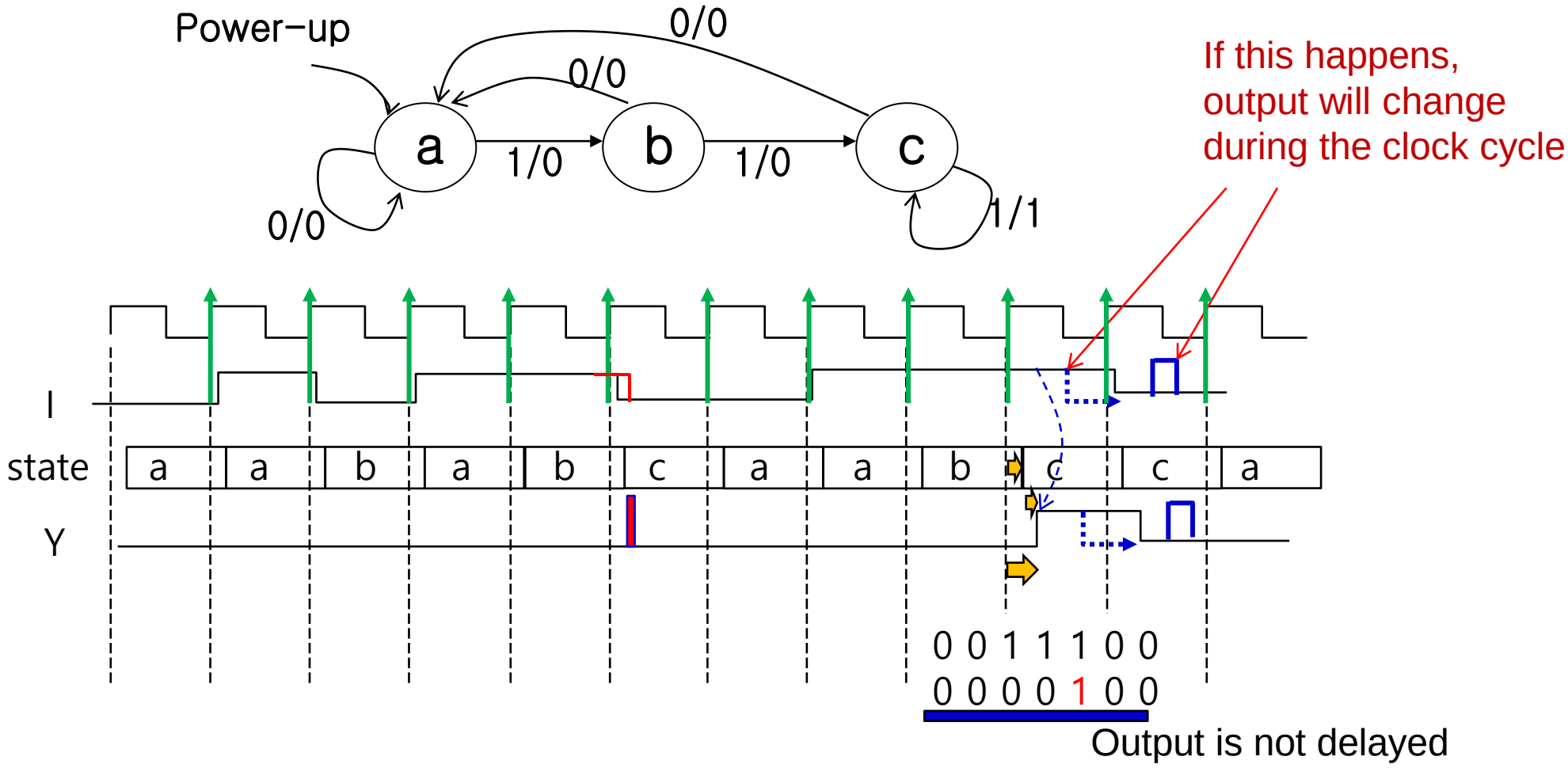
✓ Determine the excitation equations

✓ Finish the design

- Output logic
- Excitation logic

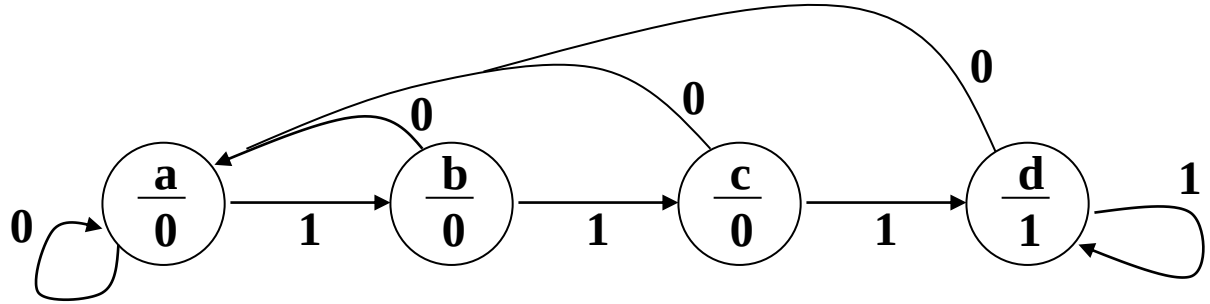
Mealy 111 detector

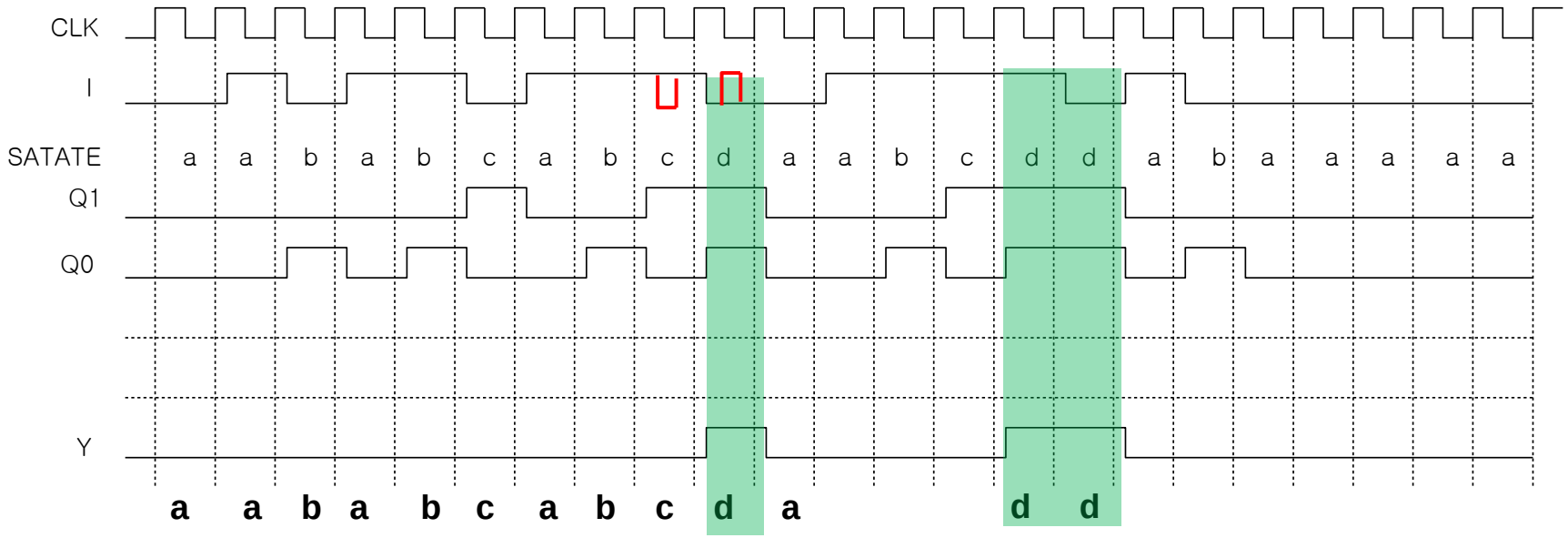
- State diagram



Moore 111 detector

- State diagram





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

Output is stable for a whole clock period even when inputs and other signals change.

Output is delayed by 1 clock cycle

Moore vs Mealy Machine

- ✓Outputs depend only on the present state.
- ✓Easier to design and debug than Mealy machines.
- ✓But often contain more states than equivalent Mealy machines.
- ✓No outputs occur during the transition.
- ✓Cannot respond to an input until the active edge of the clock occurs; this is in contrast to a Mealy circuit.

Summary

- We learned how to analyze sequential circuits
  - State equations, state table, state diagram
  - Finite state machines: Mealy and Moore machines

- Learned how to design sequential circuits
  - Sequential circuit design procedure
  - Design sequential circuits in Verilog