

ITI 1100/section A Winter 2011

DIGITAL SYSTEM I

Lectures:

Monday, 11:30 – 13:00 room: ***STE-G0103***

Thursday, 13: 00 –14:30 room: ***STE-G0103***

TUTORIAL: **Thursday,** 17:30 - 19:00 ***MNT 203***

LAB

LAB 1 **Thursday,** 19:00 - 22:00 ***CBY B302***

LAB 2 **Wednesday,** 14:30 - 17:30 ***CBY B302***

Professor : Dr. A. Karmouch, office ***CBY A508***

Mid-term exam date: Saturday March 5, 2011
(10:00-11:30)

Chapter 5

Synchronous Sequential Logic

- A digital system is composed of
 - registers
 - data processing units
- Binary information is transferred from one set of registers into another
- Transfer can be done directly or via a processing unit to perform an operation

Registers

- A register is a group of *Binary cells*
- a register with n cells can store n bits

Example:

- Register with 8 cells can store 8 bits and can be in one of 2^8 possible states

0	0	0	0	0	0	1	1
---	---	---	---	---	---	---	---

The content is function of the interpretation given to the information stored in to it. Here it's the positive integer (+3)

Registers

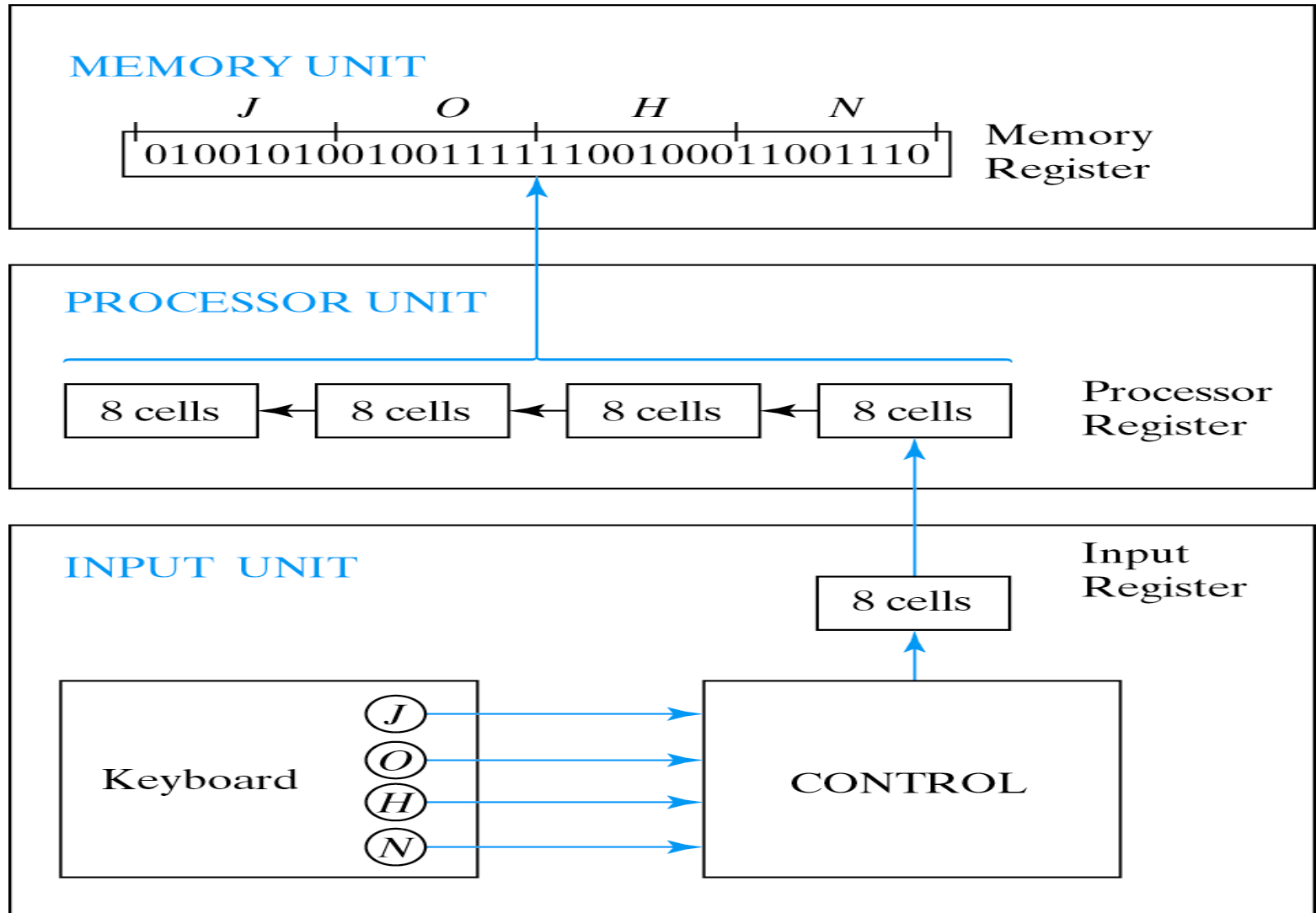


Fig. 1-1 Transfer of information with registers

Registers

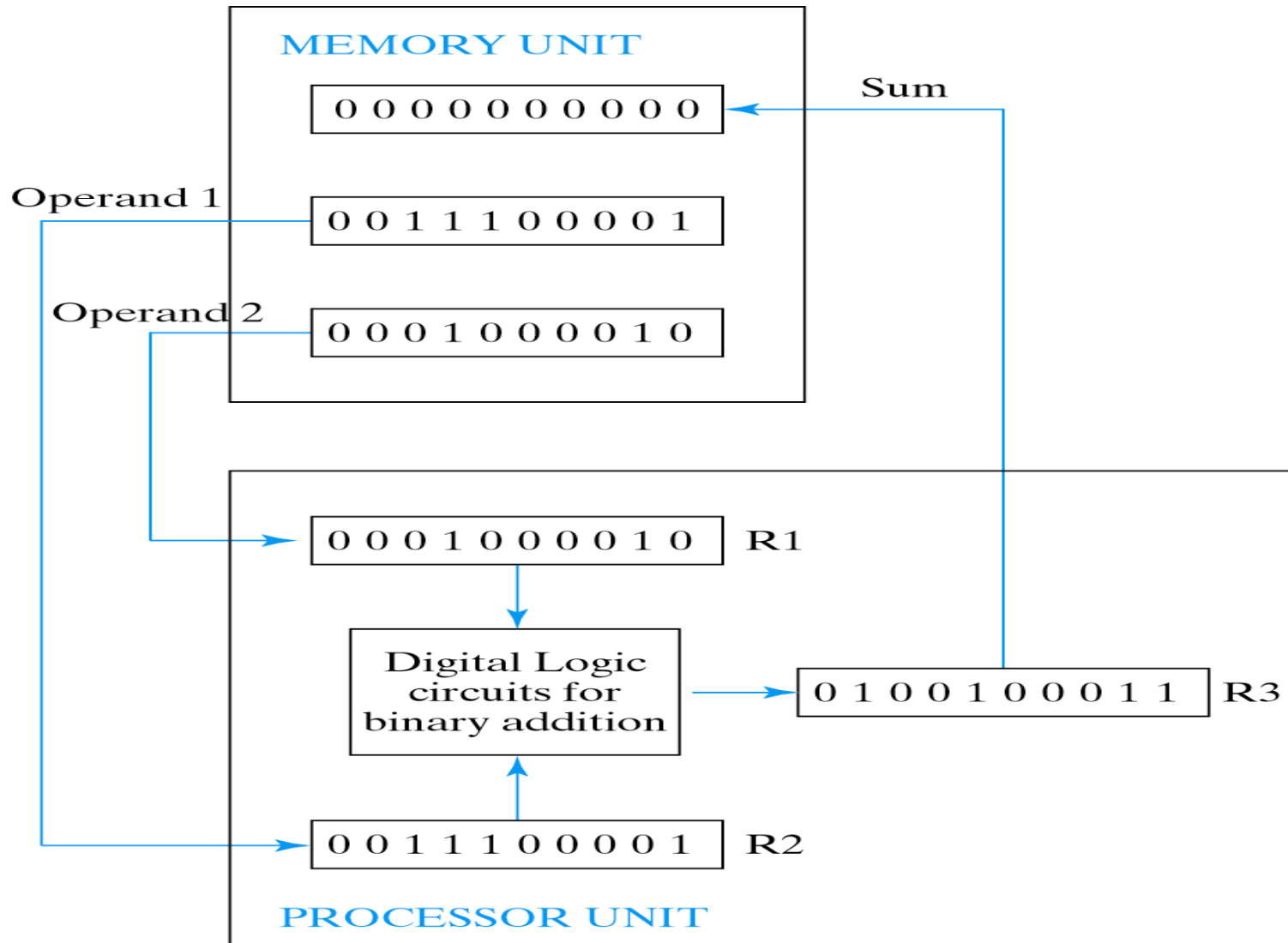


Fig. 1-2 Example of binary information processing
1111200A -A. Kamilouci

Sequential Circuits

- outputs logical functions of inputs and previous history of circuit (memory)
- after changed inputs, new outputs appear in the next clock cycle
- feedback loops

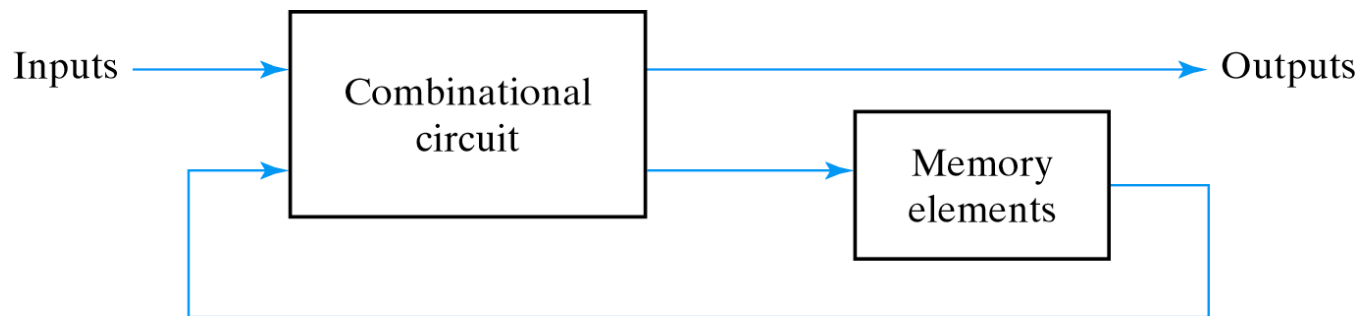
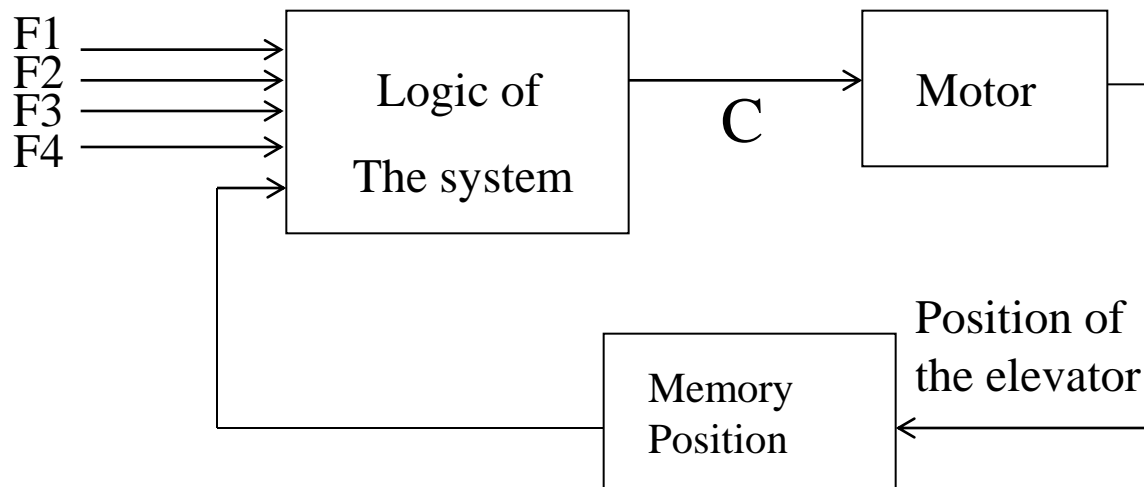


Fig. 5-1 Block Diagram of Sequential Circuit

Sequential logic circuits -Example

-- Elevator Control --

- The buttons are the input variables of the system
- The position is also an input variable which represents the current state of the elevator.
- in the figure the command control 'C' depends on the button selection AND and the position of the elevator (I.e. the previous state of the system: before the activation of the buttons)



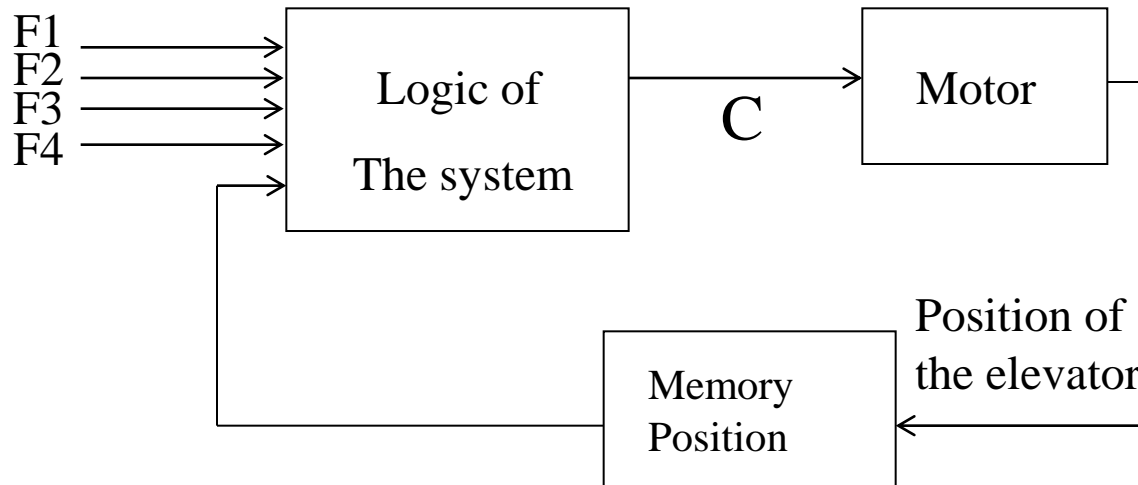
Sequential logic circuits –Example (2)

-- Elevator Control --

This example represents a **Sequential** logic system That is the output C depends on combination of

1 – **the input variables**

2 – **AND the previous states** (internal states).



Sequential circuits: time and memory

→ *In contrast to the combinational circuits* two new parameters are to be considered

1 - *Time*

We have to consider the propagation delay of the circuits

2 – *Memory Element*

We have to keep the previous results and impose a predetermined order.

→ Generally speaking, the function performed by the memory element is to store information in a binary form.

Sequential logic circuits : Flip-flop

- The memory cell that stores one "bit" of information is termed Flip-flop.
- A set of Flip-flop can be used to store several bits (one each) in a REGISTER.
- two different modes:
 - 1- Asynchronous
 - 2- Synchronous

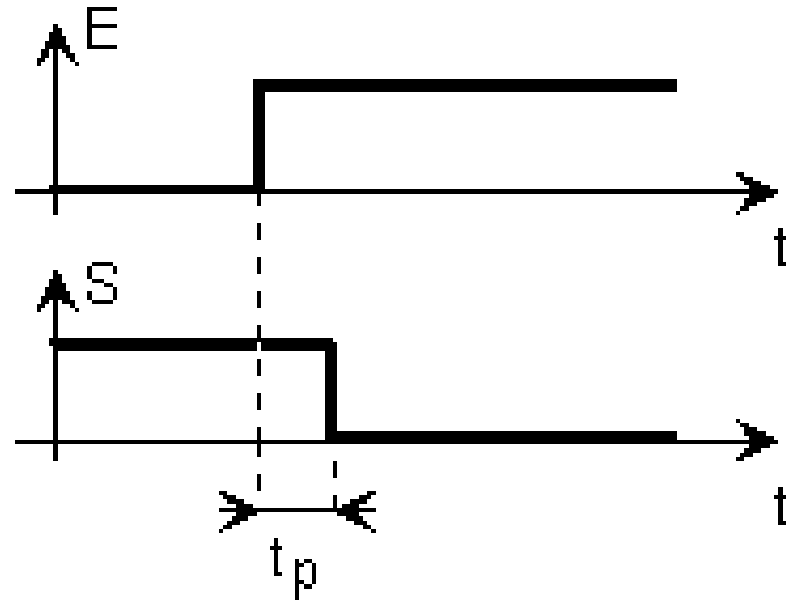
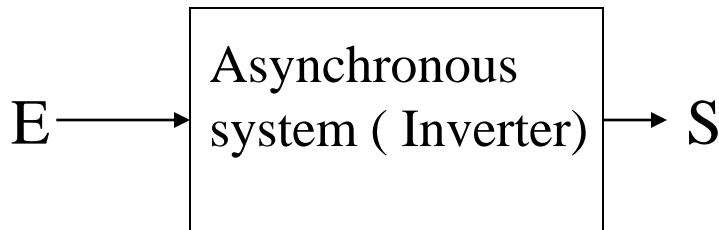
Sequential logic circuits

1- Asynchronous

- The output reacts "immediately" to the changes of the input variables

→ Propagation delay of the gates are to be considered

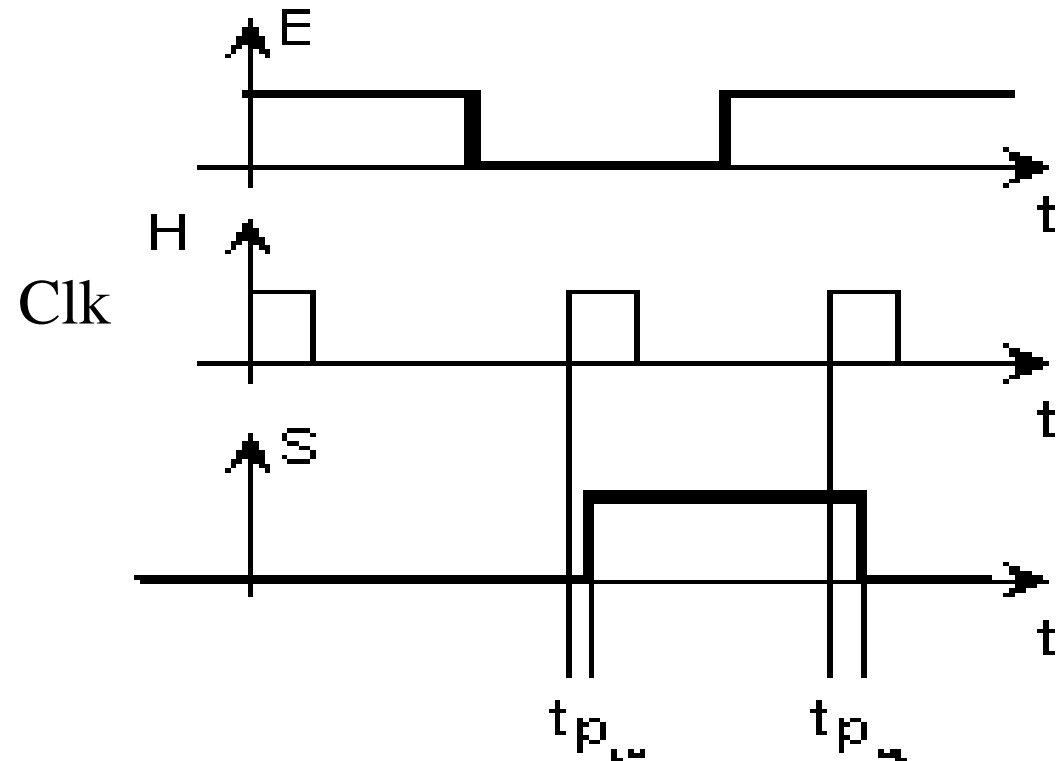
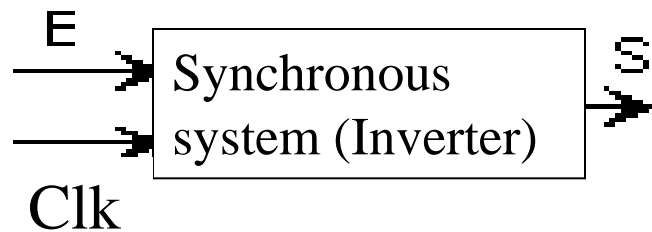
--- > t_p

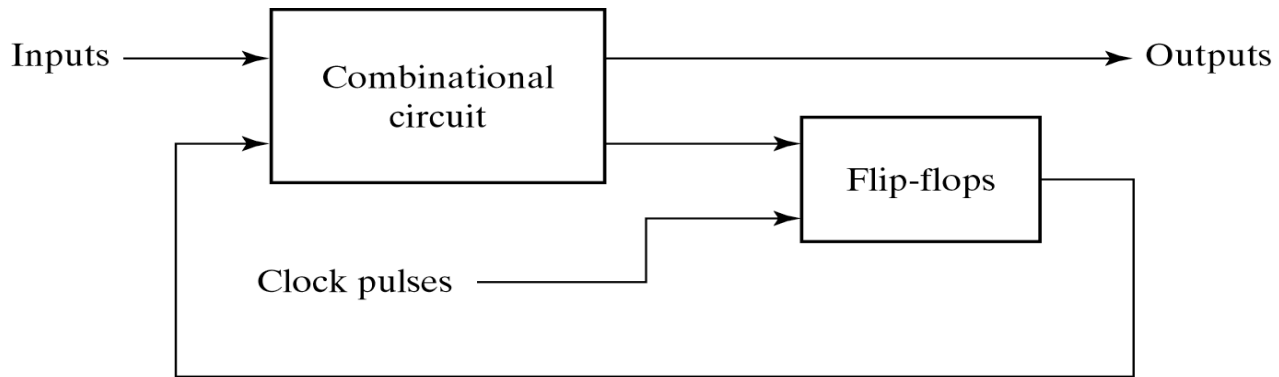


Sequential logic circuits

2- Synchronous

- In this mode of operation, changes of the input variables will be become **EFFECTIVE** when the clock input is active.





(a) Block diagram



(b) Timing diagram of clock pulses

Fig. 5-2 Synchronous Clocked Sequential Circuit

Clock signal : definitions

- Clock is used in synchronous logic circuits to trigger the circuit (**Flip-flop**) allowing it to switch its state.
- Clock signal can trigger a flip-flop in three ways:
 - **During Positive level**
 - **During Positive transition (or positive edge)**
 - **During Negative transition (or negative edge)**

Timing Diagram –Input and output signals

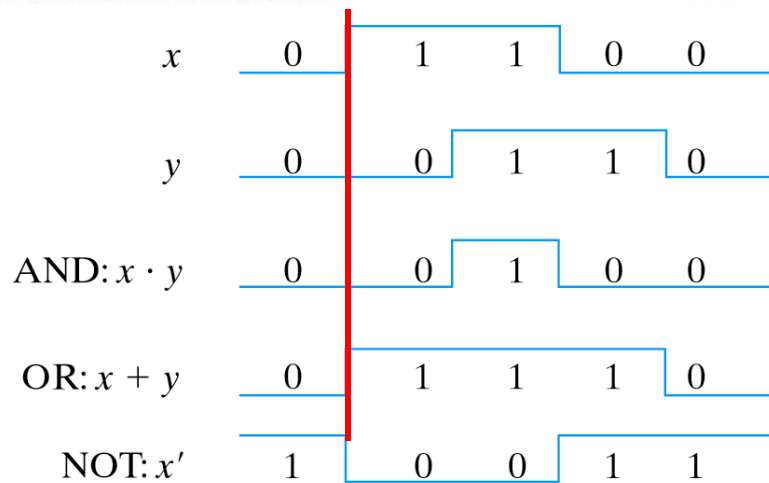
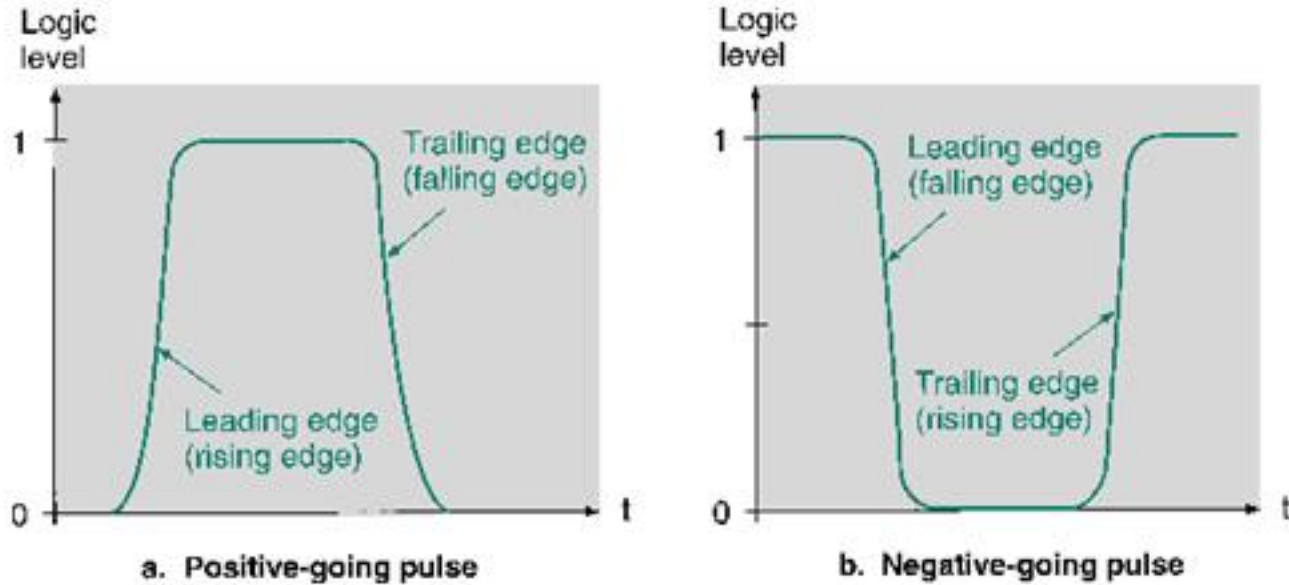
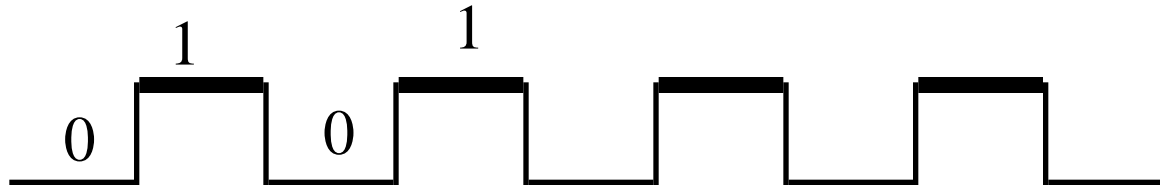


Fig. 1-5 Input-output signals for gates

Clock signal : definitions

1- Positive level

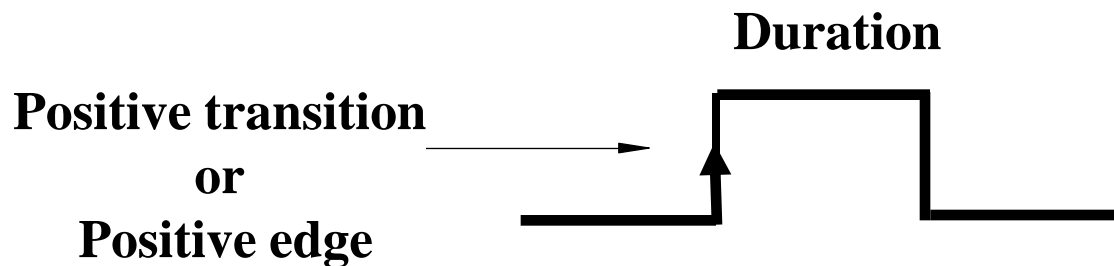
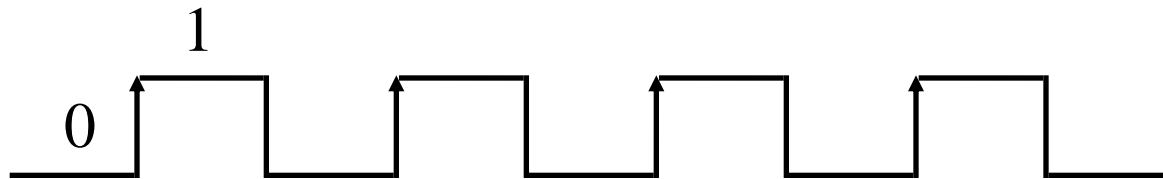
- The flip-flop is triggered while the clock pulse is at logic '1'
- the logic '0' is therefore inactive state where changes to the flip-flop (state) are not allowed.



Clock signal : definitions (2)

2- Positive transition (or positive edge)

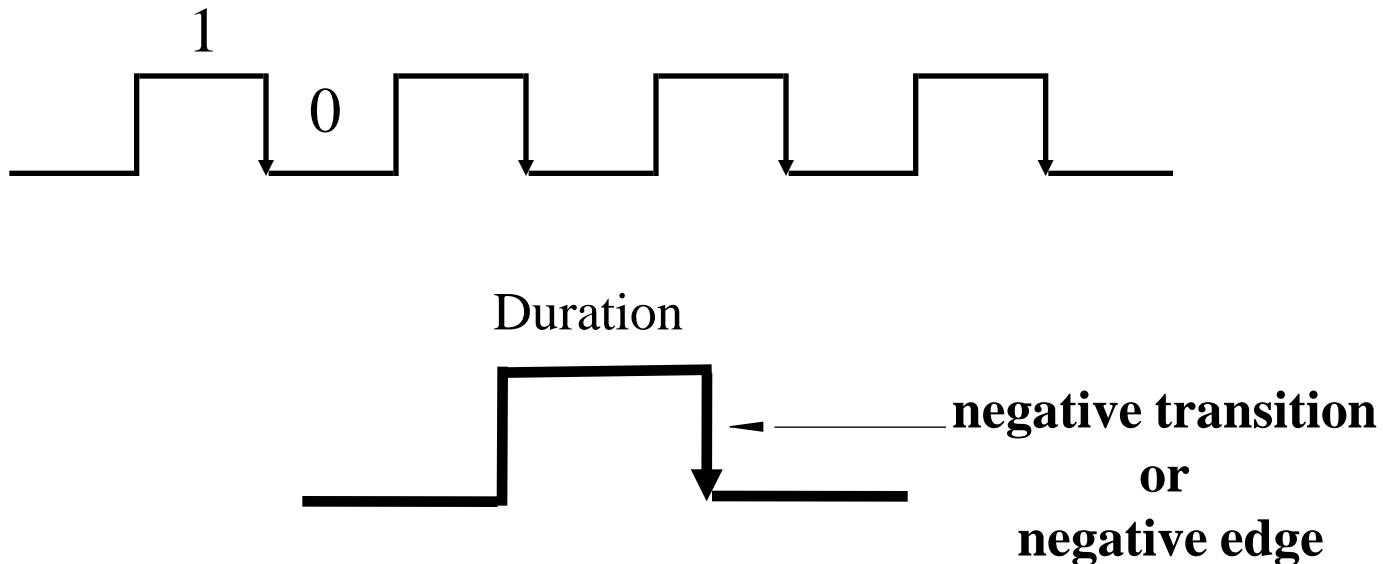
The flip flop is triggered only during the signal transition from 0 to 1 , this transition is defines as *positive edge*



Clock signal : definitions (4)

3- negative transition (or negative edge)

The flip flop is triggered only during the signal transition from **1 to 0**, this transition is defines as *negative edge*



Memory Element (Latch)

- **A flip-flop** can maintain a binary state indefinitely until directed by an input signal to switch states
- The most basic types of flip-flops operate with signal levels and are referred to as **Latches**.
- **Latches** are basic circuits from which all flip-flops are constructed

Memory Element (SR Latch)

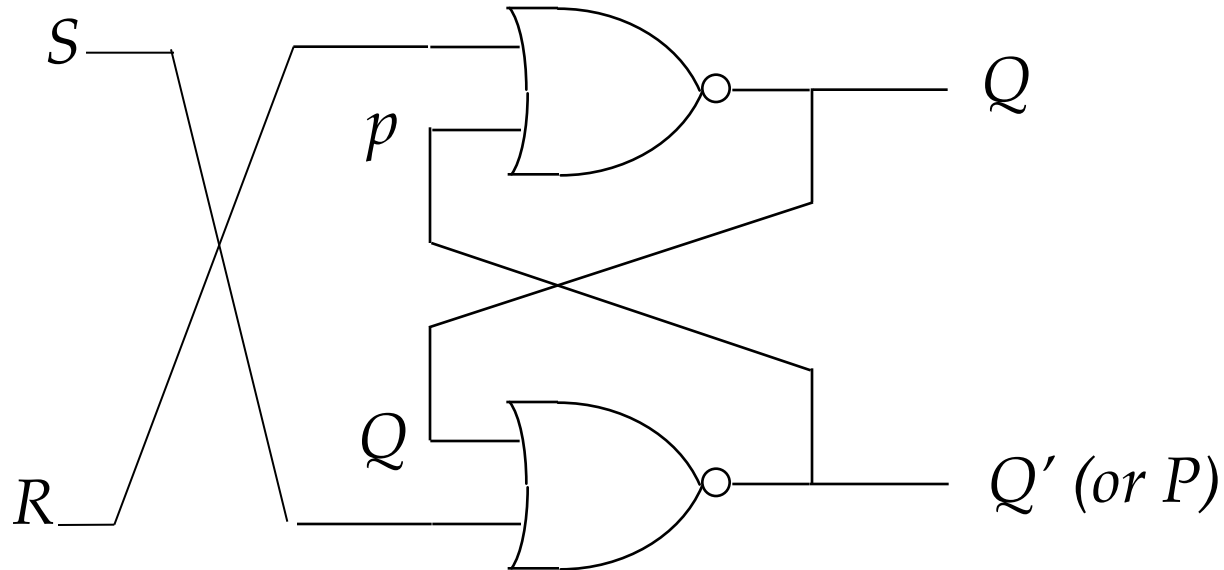
- **Flip-flop** (Latch) SR has two input variables that are defined as S and R
 - S for SET
 - R for RESET
- It also has two outputs Q (**normal state**) and Q' (**complement state**)

Symbol



Memory Element (SR Latch)

1- NOR Implementation



$$Q = (R+P)' = R'P'$$

$$P = (S+Q)' = S'Q'$$

Memory Element (SR Latch)



Symbol

Function table for RS with NOR gates

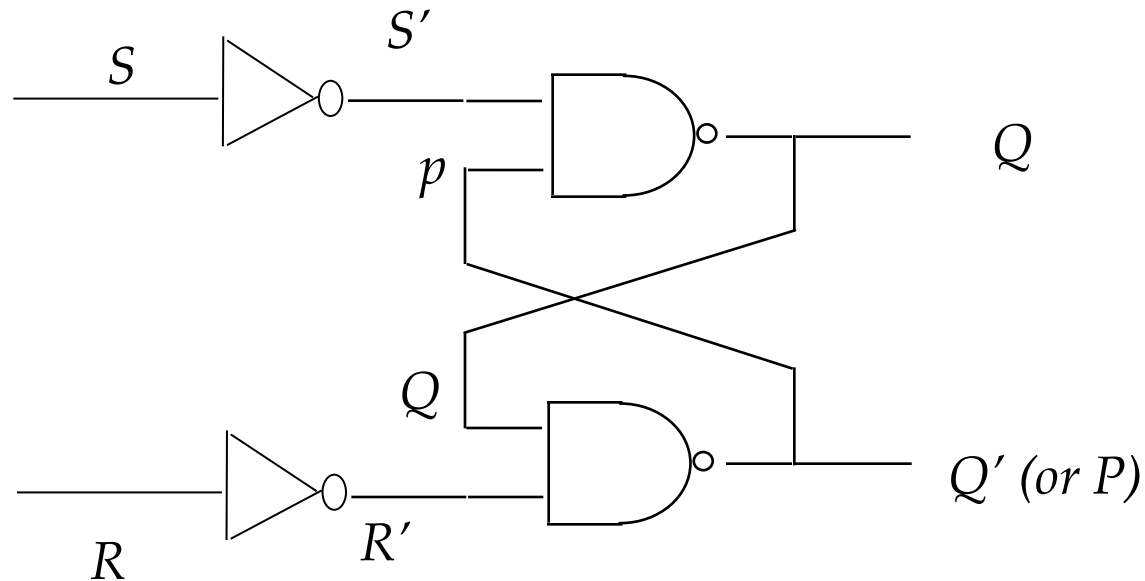
S	R	Q	Q'	
1	0	1	0	Set to "1"
0	1	0	1	Set to "0"
0	0	1	0	unchanged (after SR = 10)
0	0	0	1	unchanged (after SR = 01)
1	1	?	?	Not allowed (0 0)

Memory Element (SR Latch)

2- NAND Implementation

$$Q = (S'.P)' = S + P'$$

$$P = (R'.Q)' = R + Q'$$



Memory Element (SR Latch)

2- NAND Implementation (3)

Function table for RS with NAND gates:

S	R	Q	Q'	
1	0	0	1	set to « 0 »
1	1	0	1	unchanged (after SR = 10)
0	1	1	0	set to « 1 »
1	1	1	0	unchanged (after SR = 01)
0	0	?	?	Not allowed! (1 1)

Memory Element (Latch)

- **A flip-flop** can maintain a binary state indefinitely until directed by an input signal to switch states
- The most basic types of flip-flops operate with signal levels and are referred to as **Latches**.
- **Latches** are basic circuits from which all flip-flops are constructed

Synchronous Sequential Circuits

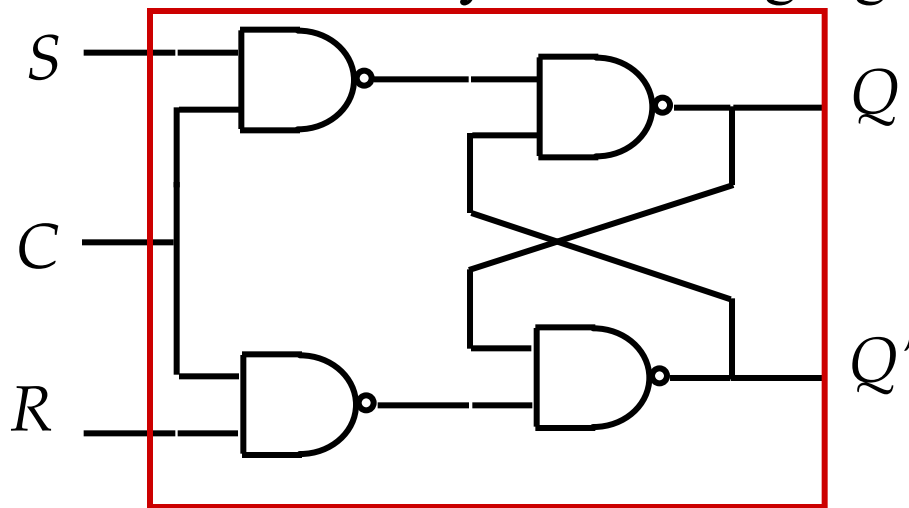
- If we add to the memory element a **synchronization signal** (i.e. clock impulse or control input) we would obtain a synchronous SR latch (i.e. a Flip-Flop)
- We will examine the following synchronous flip-flops
 - SR
 - D and T
 - JK

SR Latch with Control Input

- The time when a latch is allowed to change state is regulated.
- Change of state is regulated by a control signal called ENABLE.
- **Circuit is a NAND latch controlled by steering gates.**

Latch with Control Input: SR Latch

- Used in two principal ways.
 - as an ON/OFF signal.
 - as a synchronizing signal.



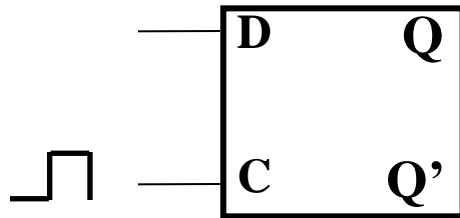
C	S	R	Q _{t+1}	Q' _{t+1}	Function
1	0	0	Q _t	Q' _t	No change
1	0	1	0	1	RESET
1	1	0	1	0	SET
1	1	1	0	0	Forbidden
0	X	X	Q _t	Q' _t	Inhibited.

- Sometimes it's useful to avoid latch changes
 - When C = 0 → disables all latch state changes*
- Control signal *enables* state change when C = 1
- Right side of circuit same as ordinary S-R latch with NAND.

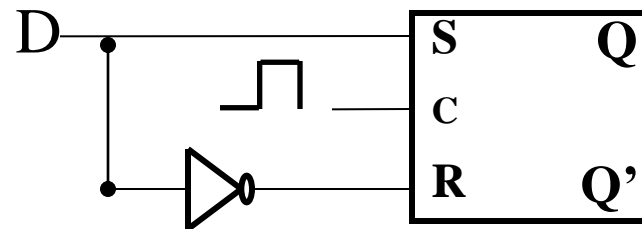
Latches with Control Input: D latch

- D for Data. Data is transferred to the Latch

Symbol



Implementation with
SR latch



When C is high, D passes from input to output (Q)

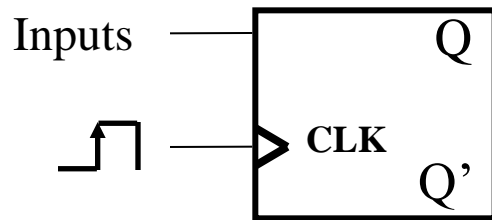
D	C	Q	Q'
0	1	0	1
1	1	1	0
X	0	Q_n	Q_n'

Edge Sensitive Flip-Flop

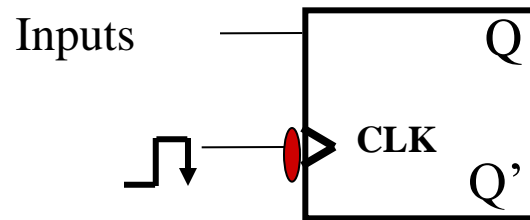
- Latches respond to trigger levels on control inputs
Example: If $C = 1$, input is reflected at output
- Flip flops store data on a positive or negative edge.
Example: control input transitions from 0 to 1, data input available at output
- Data remains stable in the flip flop until next positive /negative edge.
- Different types of flip flops are used for specific functions

Edge Sensitive Flip-Flop :definitions

- Latch with a clock input.
- The sequential circuit output changes when its CLOCK input detects an edge. **Edge-sensitive** instead of **level-sensitive**.
- **Symbol is a triangle on the CLK (clock) input of a flip-flop.**
- Flip flop can be triggered on **positive** or **negative edge**
- **Bubble before Clock (CLK) input indicates negative edge trigger**

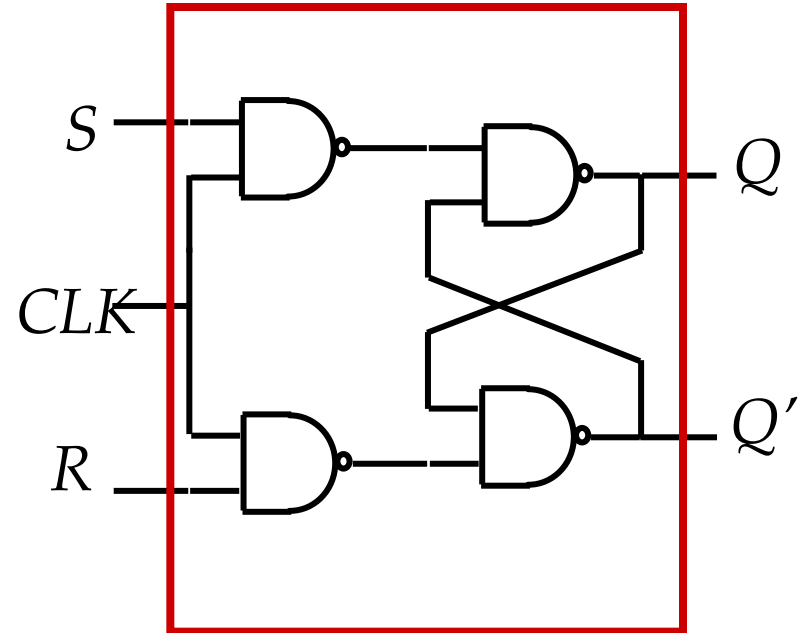
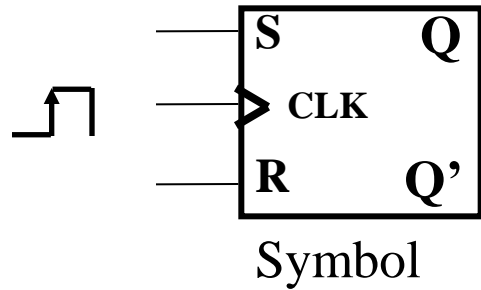


Symbol with positive edge



Symbol with negative edge

Edge Sensitive SR

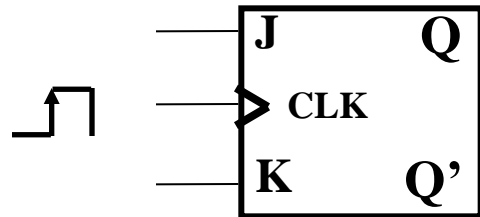


S	R	Q_{n+1}	Function
0	0	Q_n	No change
0	1	0	RESET
1	0	1	SET
1	1	?	Forbidden

- Q_n = state *before* positive edge
- Q_{n+1} = state *after* positive edge

JK Flip flop

Symbol



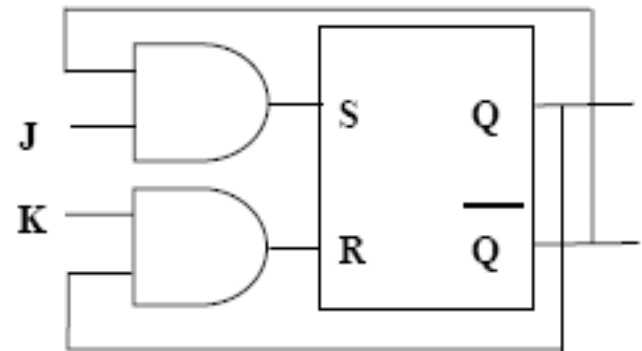
J	K	Q_{n+1}	Function
0	0	Q_n	No change
0	1	0	RESET
1	0	1	SET
1	1	Q'_n	complement (also Toggle)

Same as SR except for
 $K=J= 1$ the JK flip flop will
 Output the opposite state of
 Q_n .

- Q_n = state *before* positive edge
- Q_{n+1} = state *after* positive edge

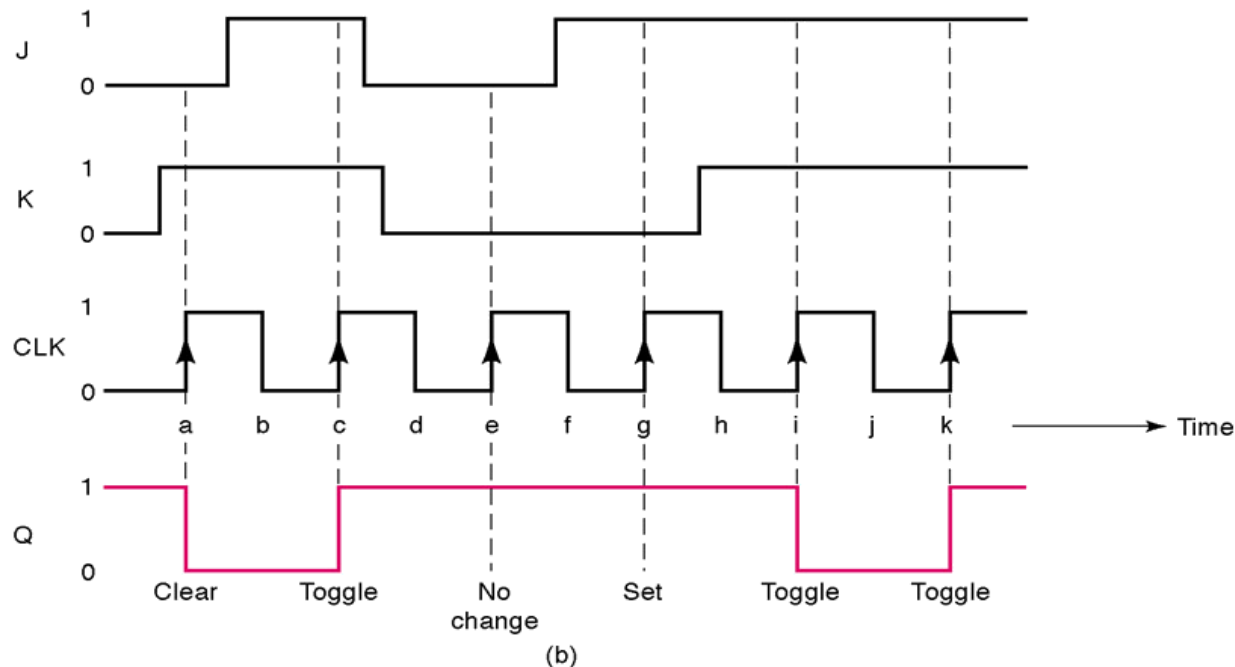
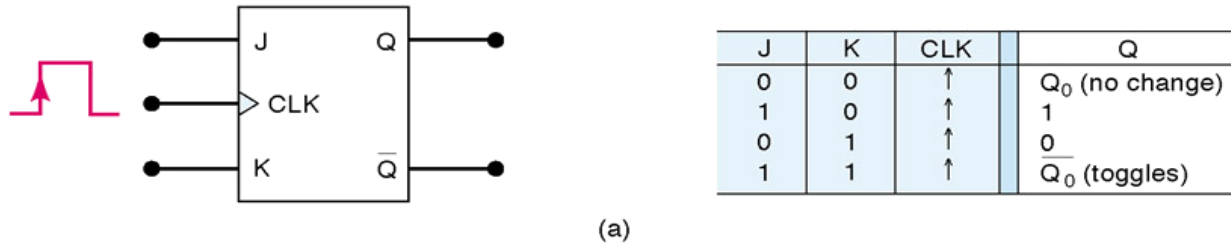
If $Q_n = 1$ then $Q_{n+1} = 0$

If $Q_n = 0$ then $Q_{n+1} = 1$



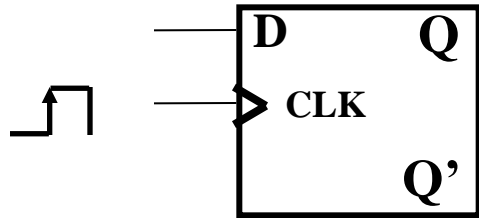
JK Flip flop

- Two data inputs, J and K
- J -> set, K -> reset, if J=K=1 then toggle (complement) output



D Flip flop

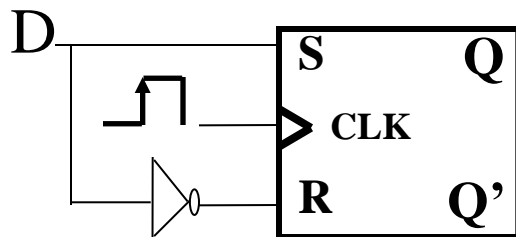
- Output changes only on the clock transition



Symbol

D	Q_{n+1}
0	0
1	1

D	CLK	Q	Q'
0	↑	0	1
1	↑	1	0
X	0	Q_0	Q_0'



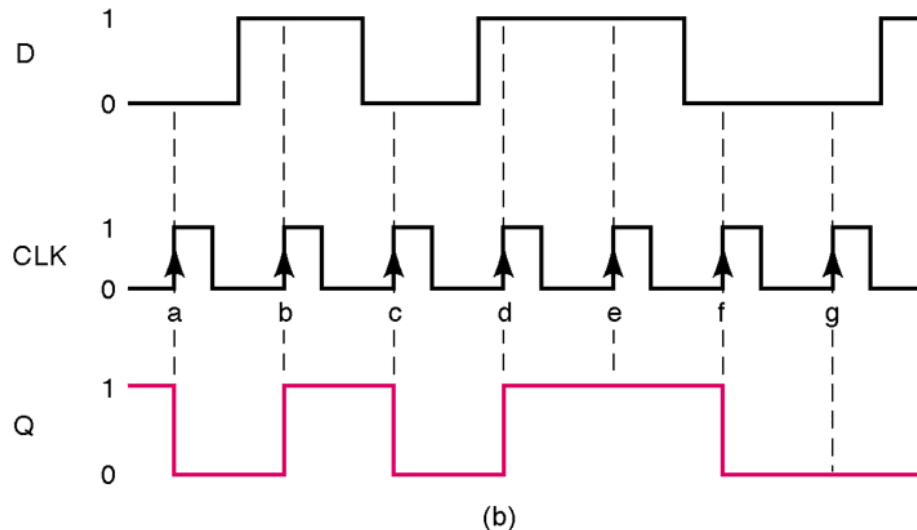
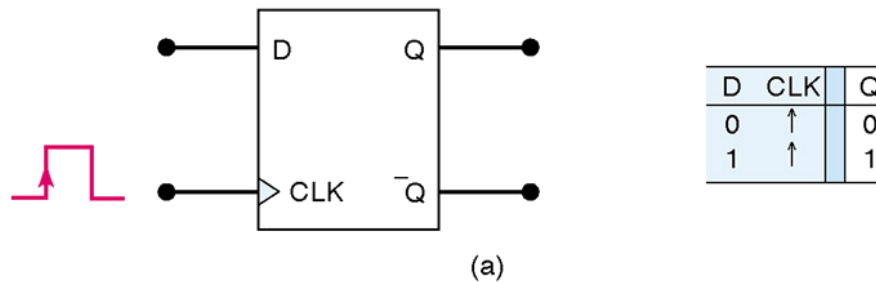
D Flip flop can be implemented using SR

$$S = D$$

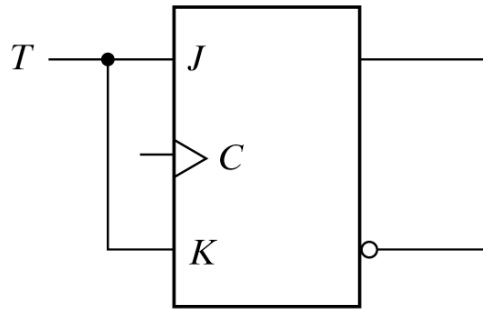
$$R = D'$$

D Flip flop

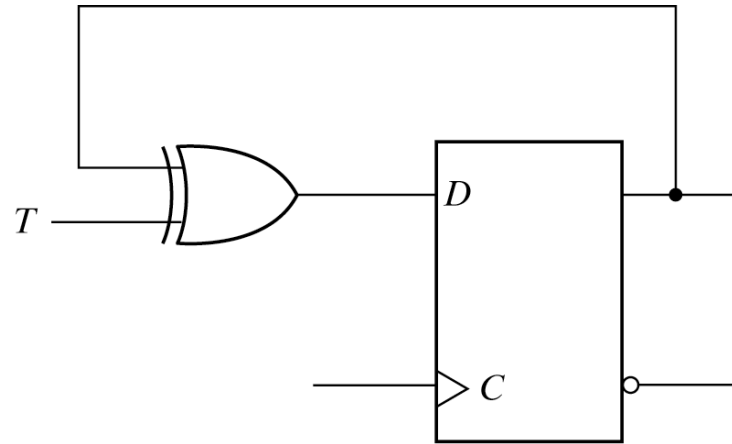
- Stores a value on the positive edge of *Clock*
- Input changes at other times have no effect on output



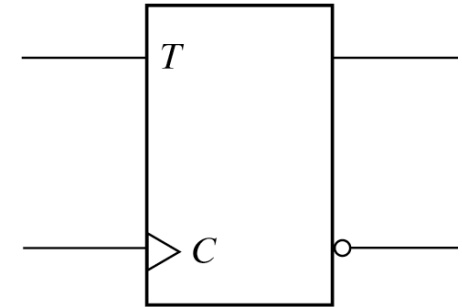
T Flip-Flop



(a) From JK flip-flop



(b) From D flip-flop



(c) Graphic symbol

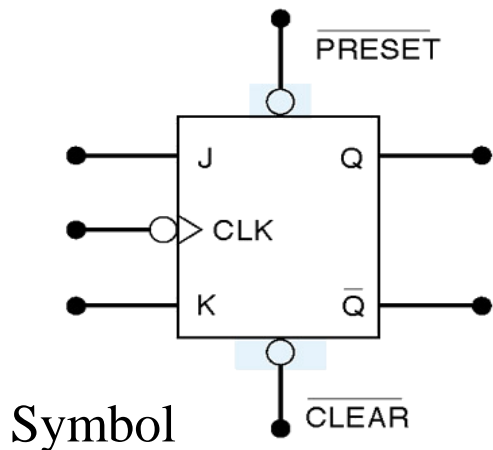
Fig. 5-13 T Flip-Flop

°Can be Created from JK or D flip flop

T	C	Q	Q'
0		Q_n	Q_n'
1		<i>Complement (toggle)</i>	

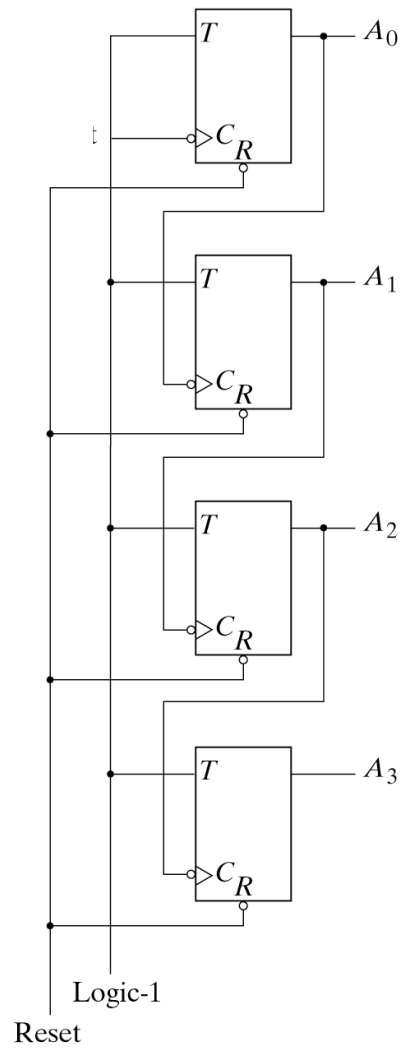
Asynchronous Inputs

- So far we have only considered **synchronous inputs** (e.g. D, S, R, J, K) → Their Effects on the output are synchronized with the *CLK* input.
- **Flip flops have asynchronous inputs.** They operate independently of the synchronous inputs and clock
 - used to Set the Flip flop to 1 or 0 state *at any time*.

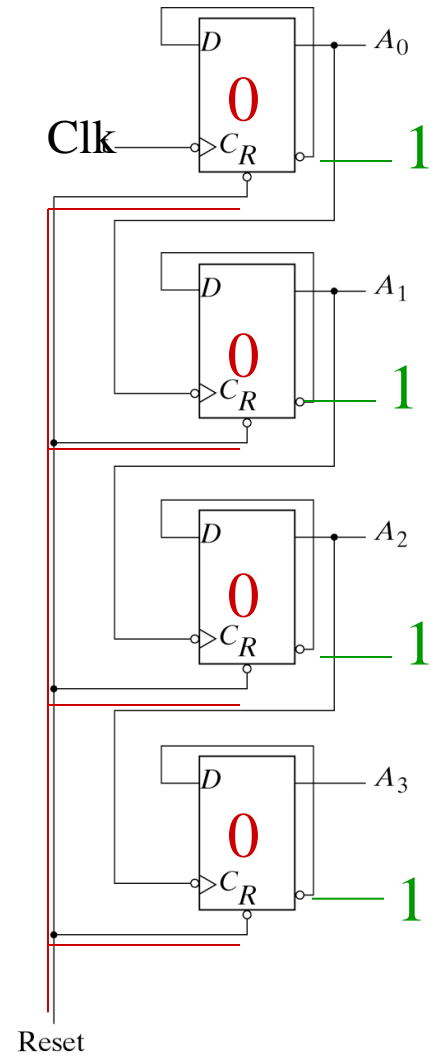


PRESET	CLEAR	FF response
1	1	Clocked operation*
0	1	Q = 1 (regardless of CLK)
1	0	Q = 0 (regardless of CLK)
0	0	Not used

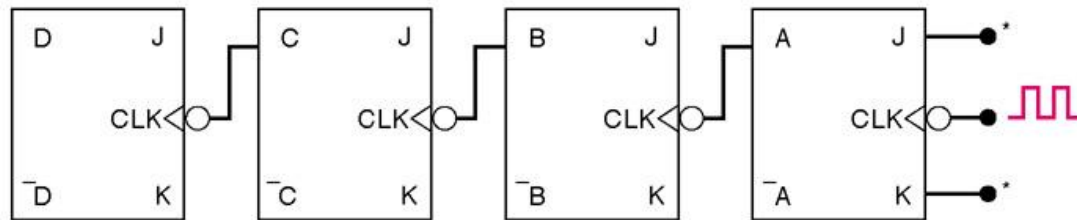
*Q will respond to J, K, and CLK



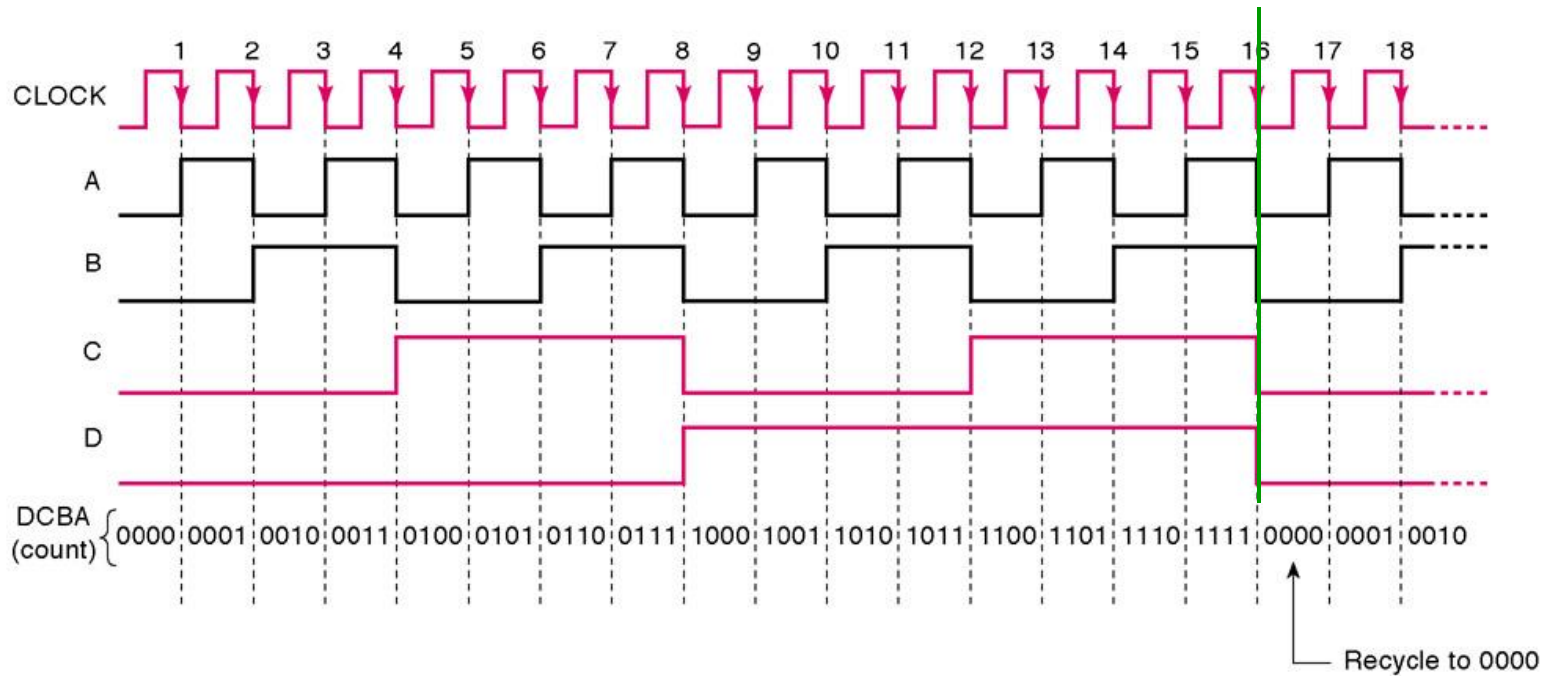
(a) With T flip-flops



(b) With D flip-flops



*All J and K inputs assumed to be 1.



- Similar to T flop example on previous slide

Analysis of synchronous sequential circuits

→ Analysis describes what a given circuit will do under certain conditions

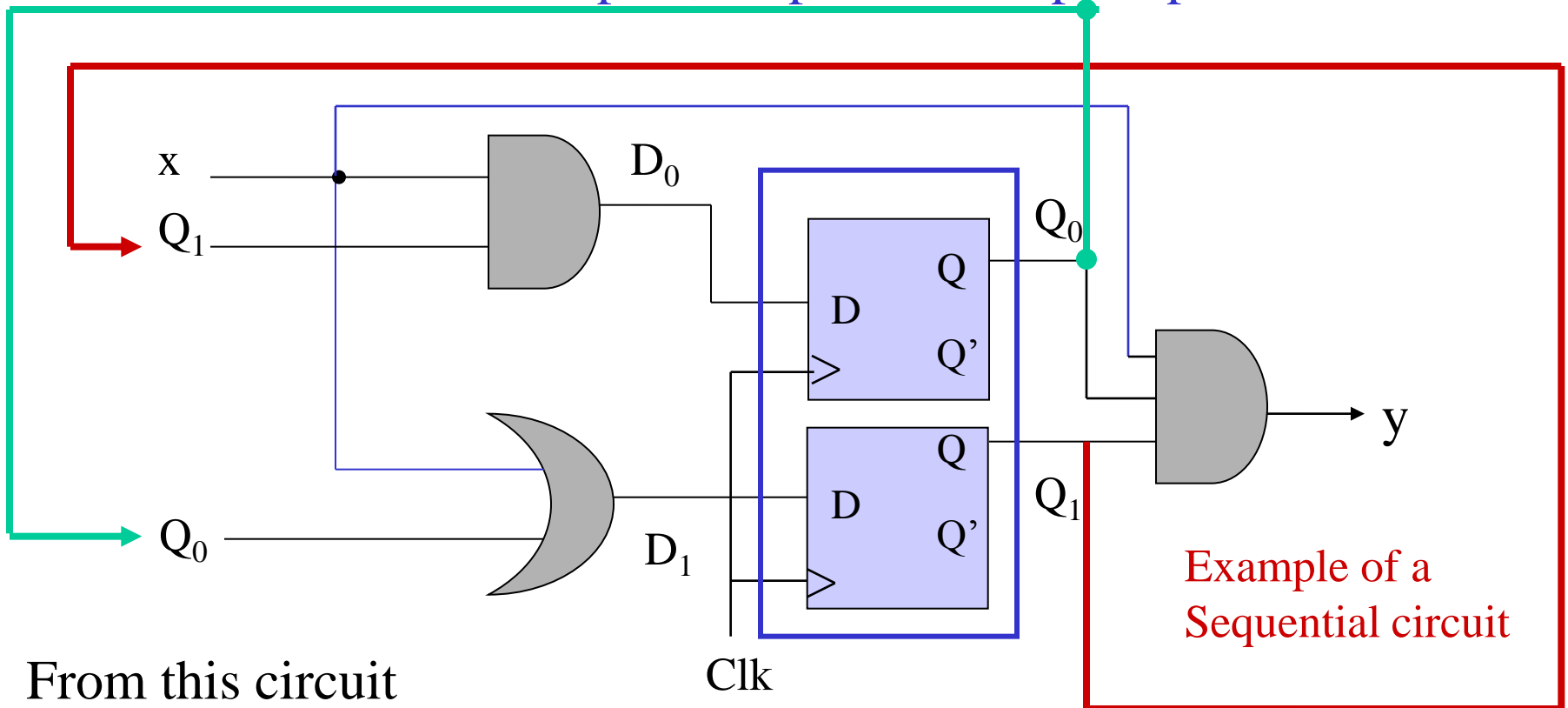
→ Behavior of synchronous sequential circuit can be determined from

- Its inputs,

- Its outputs and its Flip Flop state

Flip Flop State

- Behavior of synchronous sequential circuit can be determined from inputs, outputs and Flip Flop state



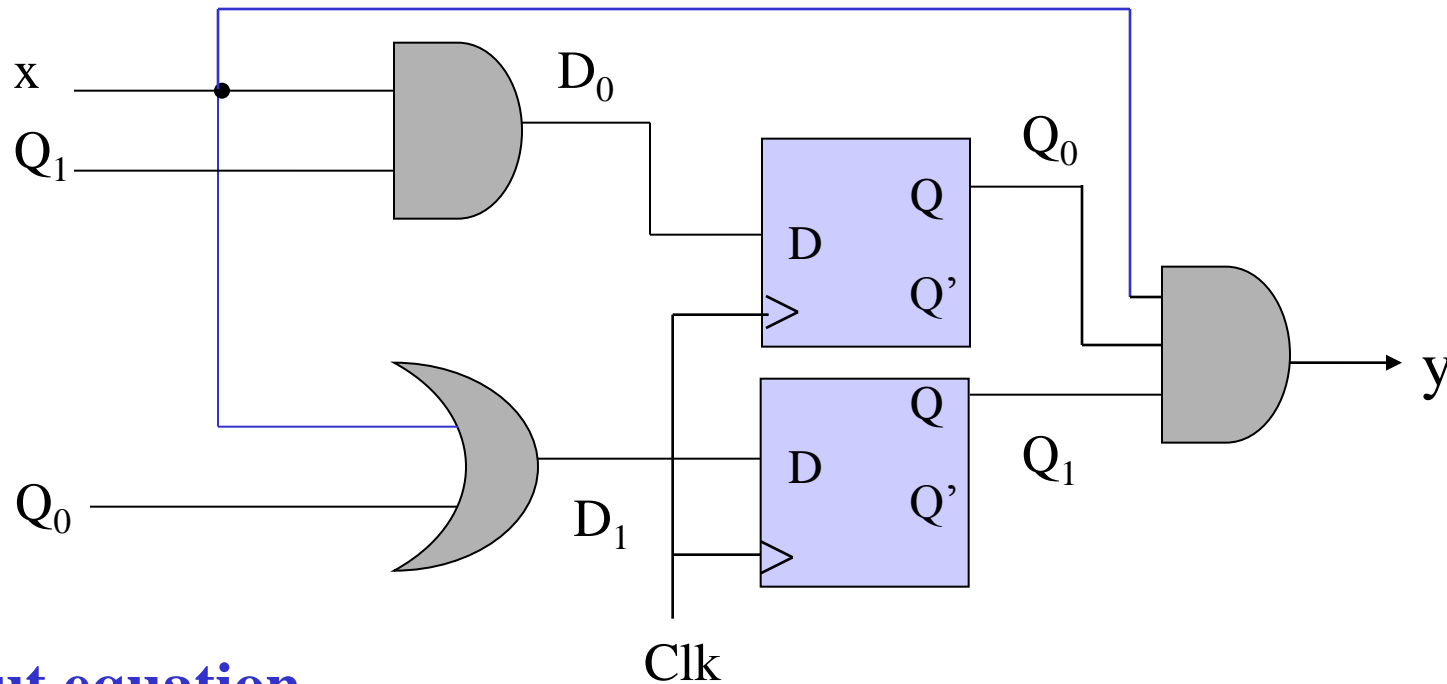
$$y(t) = x(t)Q_1(t)Q_0(t)$$

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

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

Output and State Equations

- Next state dependent on previous state.



Output equation

$$y(t) = x(t)Q_1(t)Q_0(t)$$

State equations

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

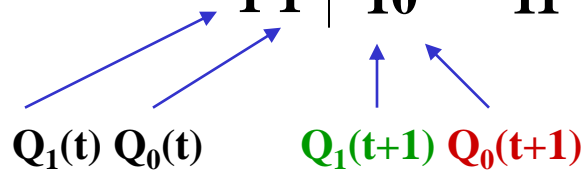
$$Q_1(t+1) = D_1(t) = x(t) + Q_0(t)$$

State Table

- Sequence of outputs, inputs, and flip flop states are listed in a table called “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



$$y(t) = x(t)Q_1(t)Q_0(t)$$

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

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

State Table

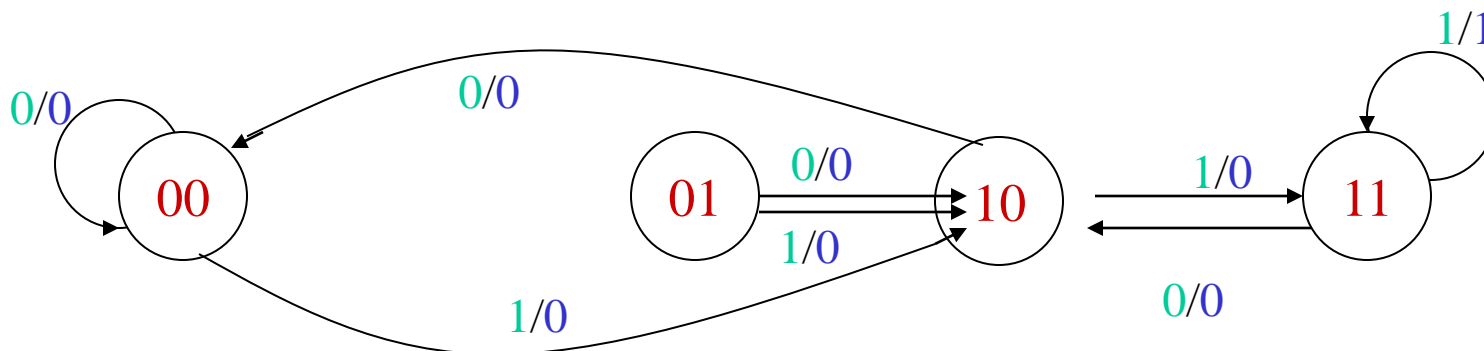
- All possible input combinations are listed
- All possible state combinations are listed
- Separate columns for each output value.
- Easier to use a symbol for each state.

Let:	Present State	Next State		Output	
		x=0	x=1	x=0	x=1
$s_0 = 00$	s_0	s_0	s_2	0	0
$s_1 = 01$	s_1	s_2	s_2	0	0
$s_2 = 10$	s_2	s_0	s_3	0	0
$s_3 = 11$	s_3	s_2	s_3	0	1

State Diagram

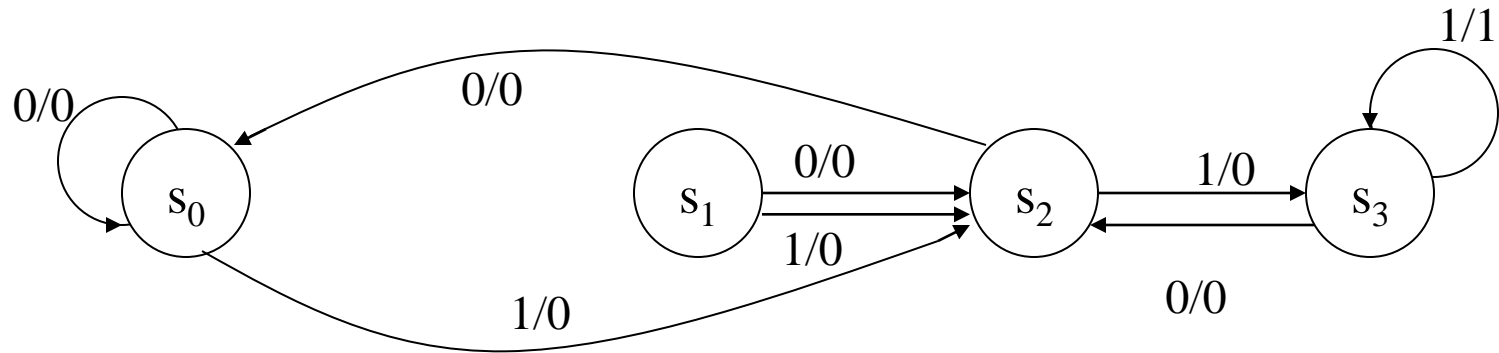
- **Circles** indicate current state
- Arrows point to **next state**
- For **x/y**, **x** is input and **y** is output

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

- 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 in this example
 - Easier to identify



Analysis of synchronous sequential circuits

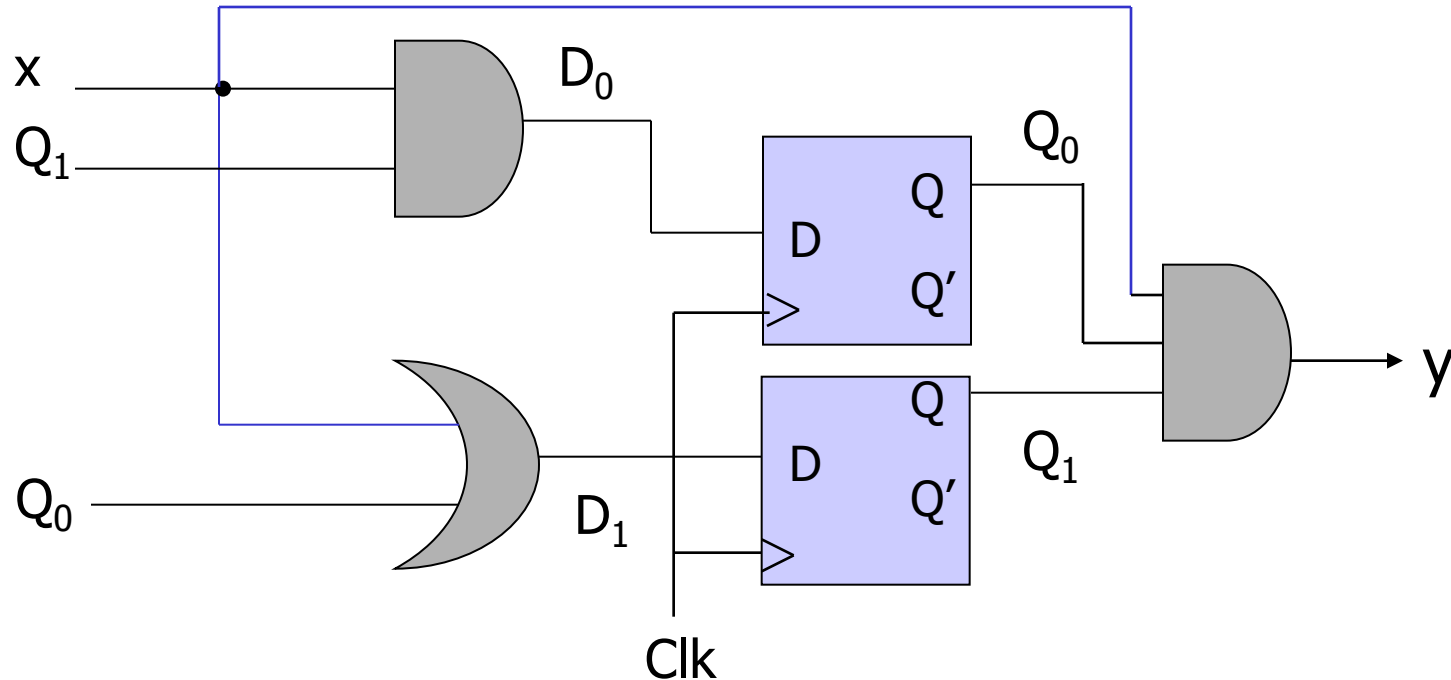
→ Analysis describes what a given circuit will do under certain conditions

→ Behavior of synchronous sequential circuit can be determined from

- Its inputs,
- Its outputs and its Flip Flop state

Flip Flop Input Equations

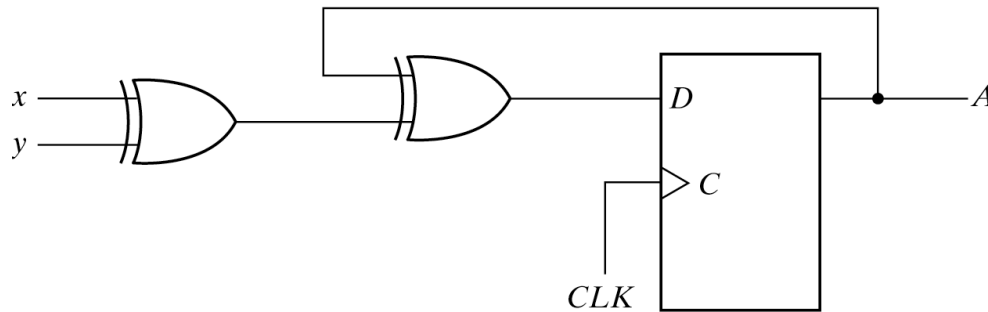
- Boolean expressions which indicate the input to the flip flops.



$$D_{Q_0} = xQ_1$$
$$D_{Q_1} = x + Q_0$$

Analysis with D Flip-Flops

- Identify flip flop input equations $D_A = A \oplus x \oplus y$
- Identify output equation (no output in this example)

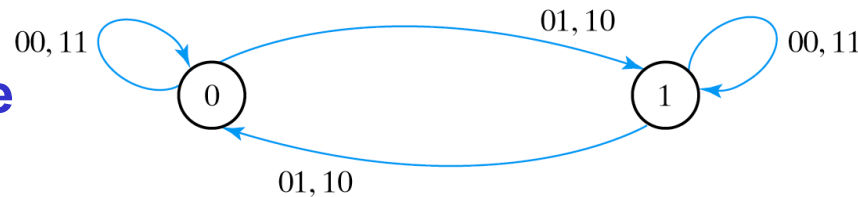


(a) Circuit diagram

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

(b) State table

Note: this example has no output



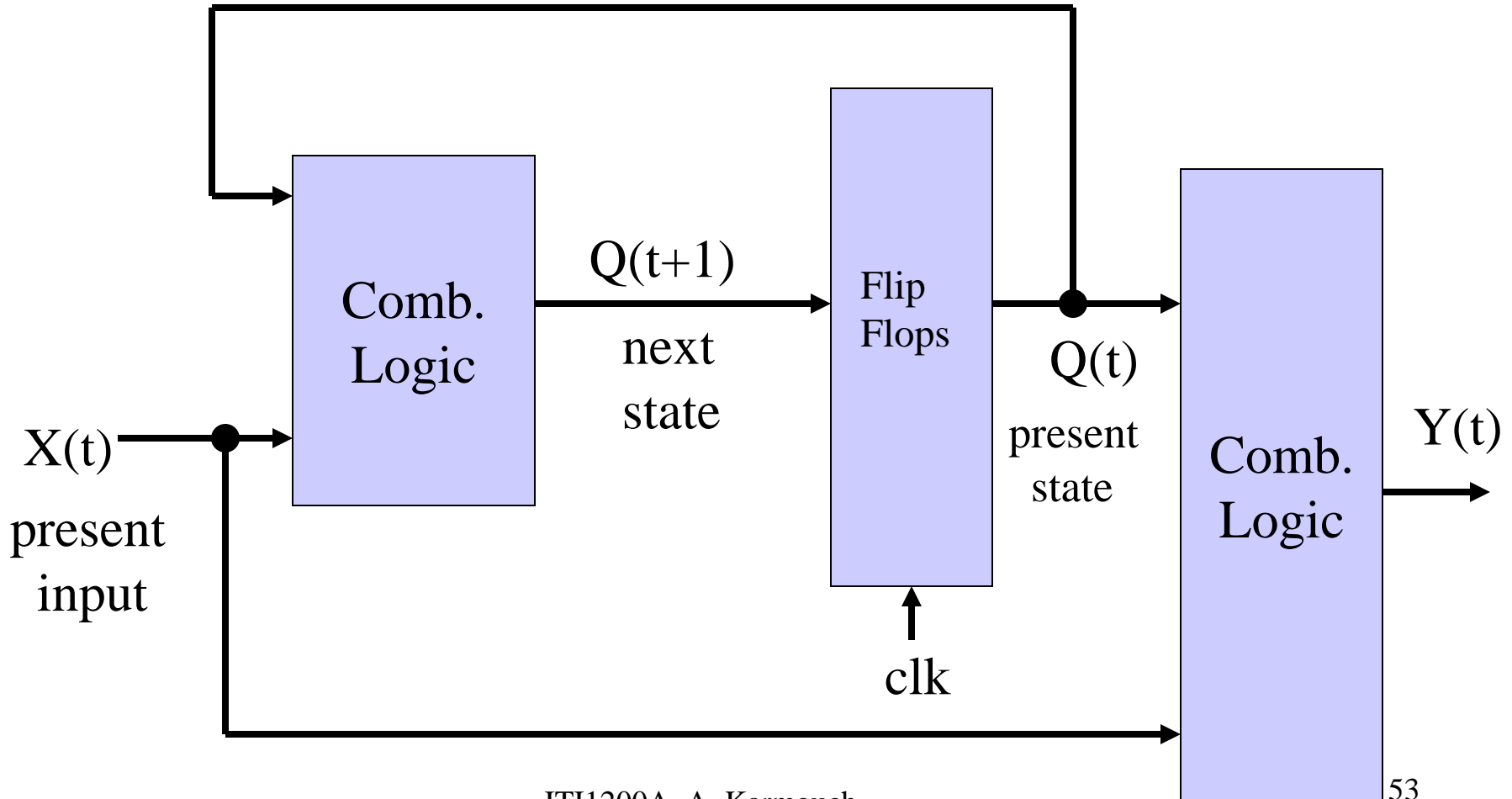
(c) State diagram

Mealy and More Models

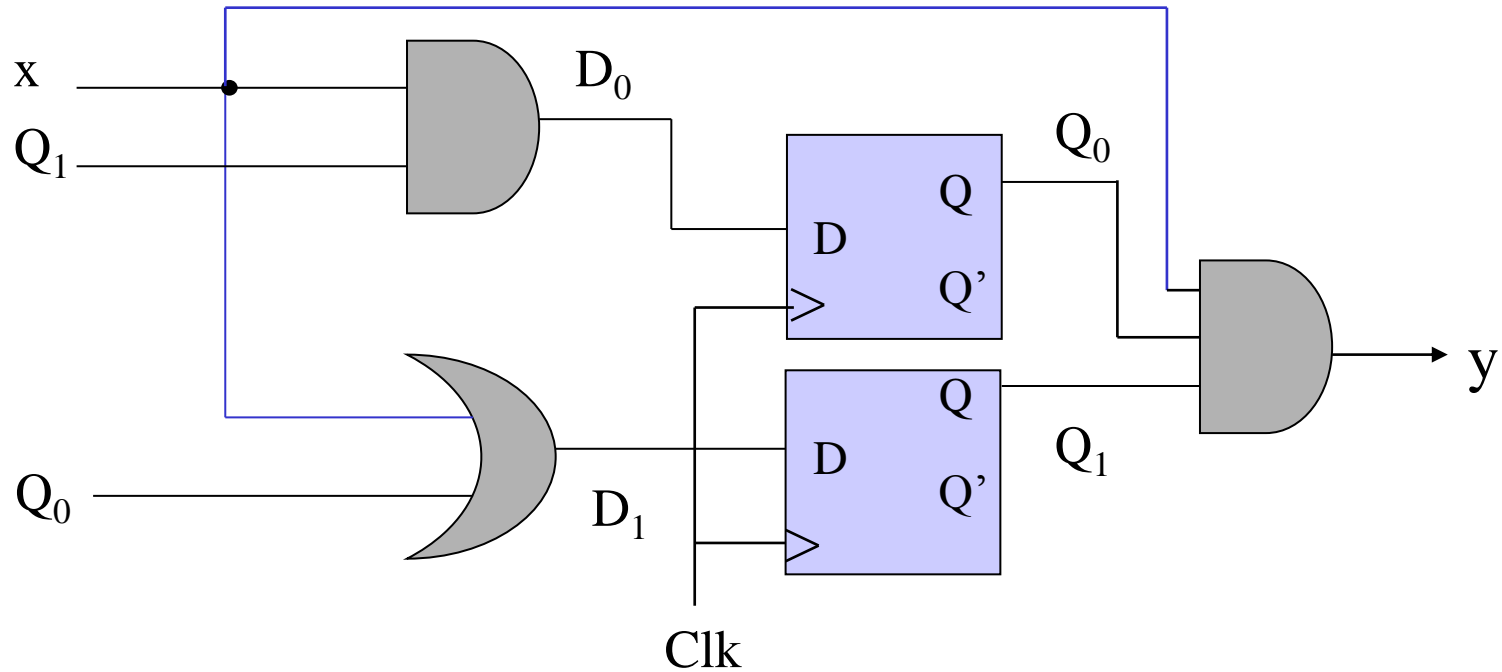
- A general model of a Sequential Circuit has
 - Inputs
 - Outputs
 - Internal States
- There are two “standard models”
 - Mealy Model, known as Mealy Finite State Machine (or Mealy machine)
 - Moore Model known as Moore Finite State Machine (or Moore Machine)

Mealy Machine

- Output based on state and present input

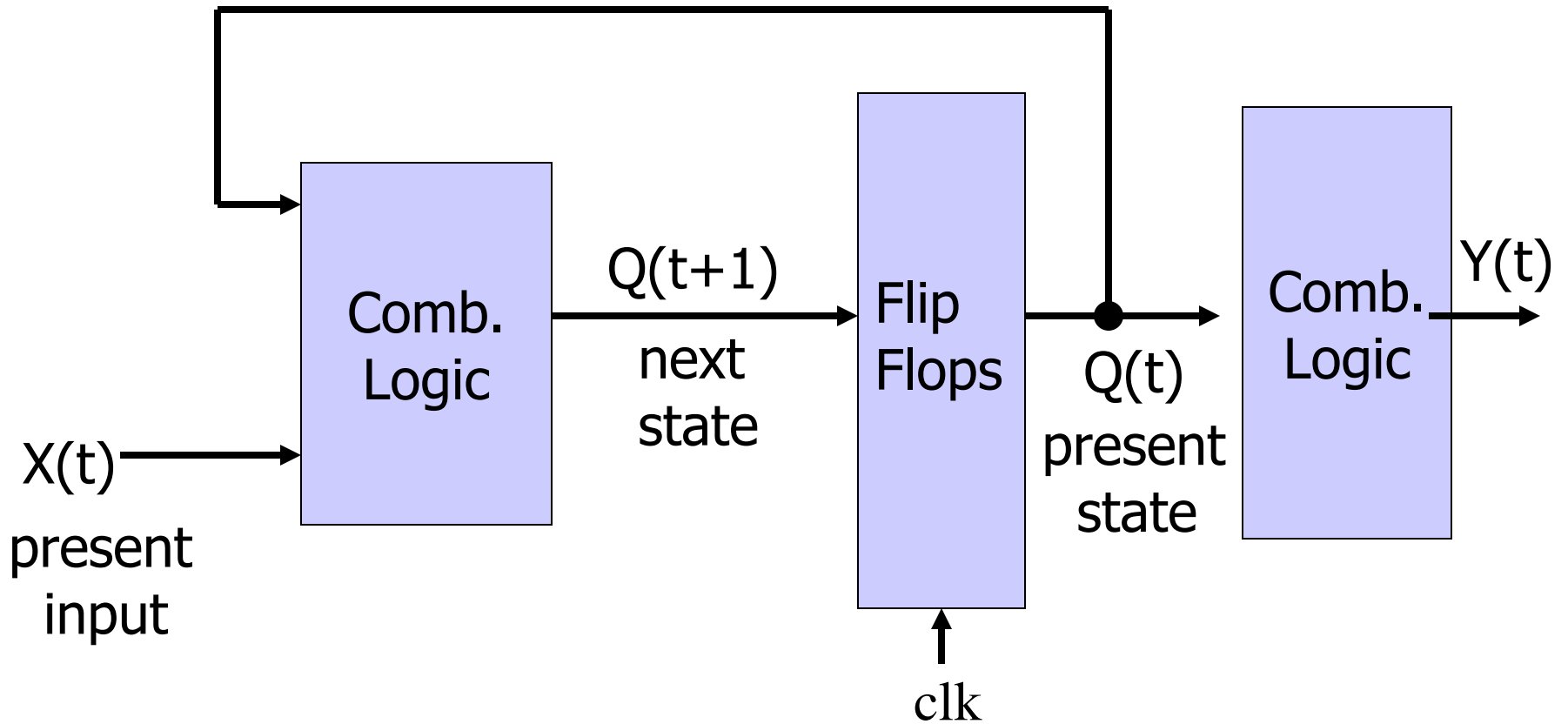


Example of Mealy Machine



Moore Machine

- Output based on state only



Example of Moore Machine

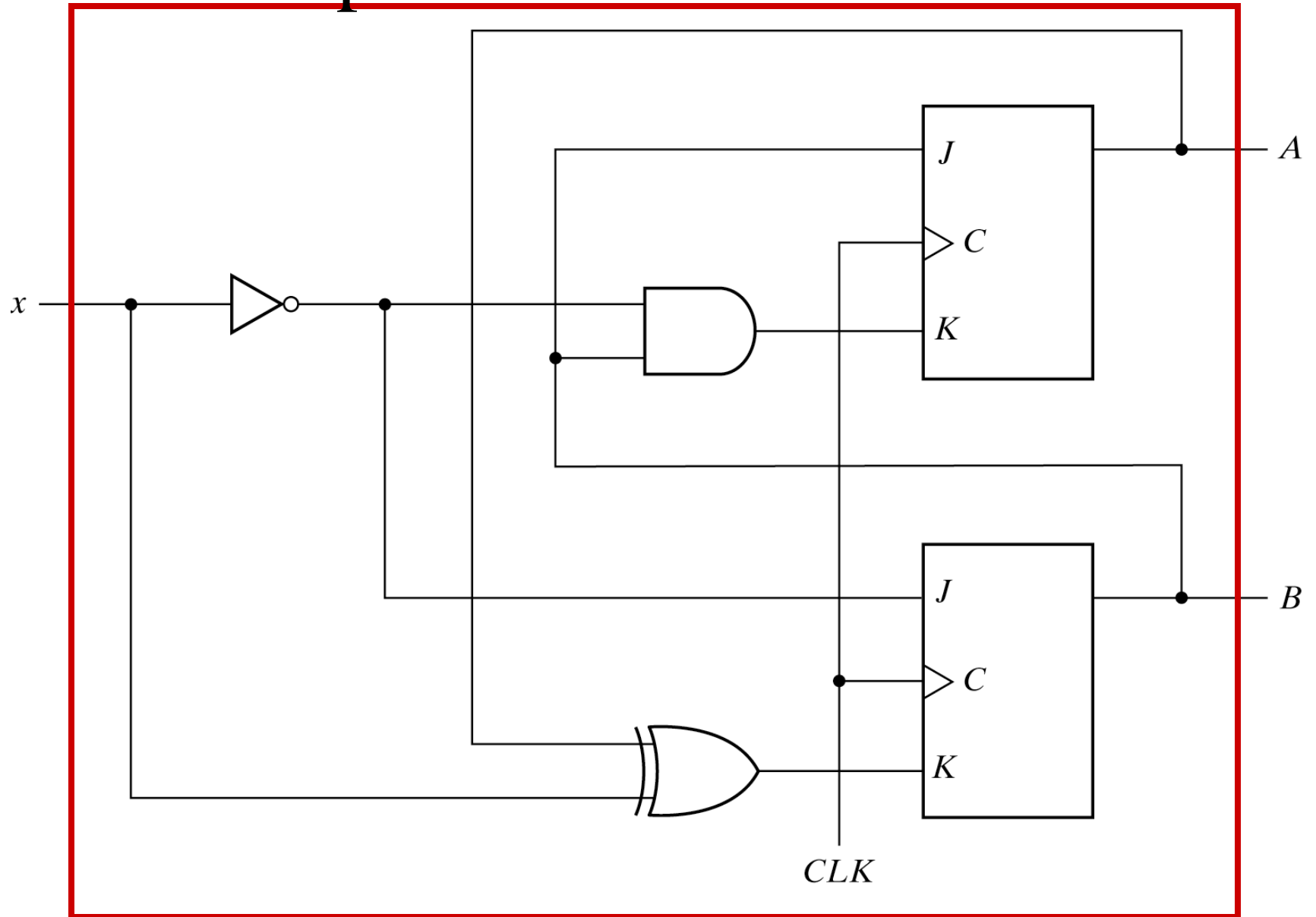
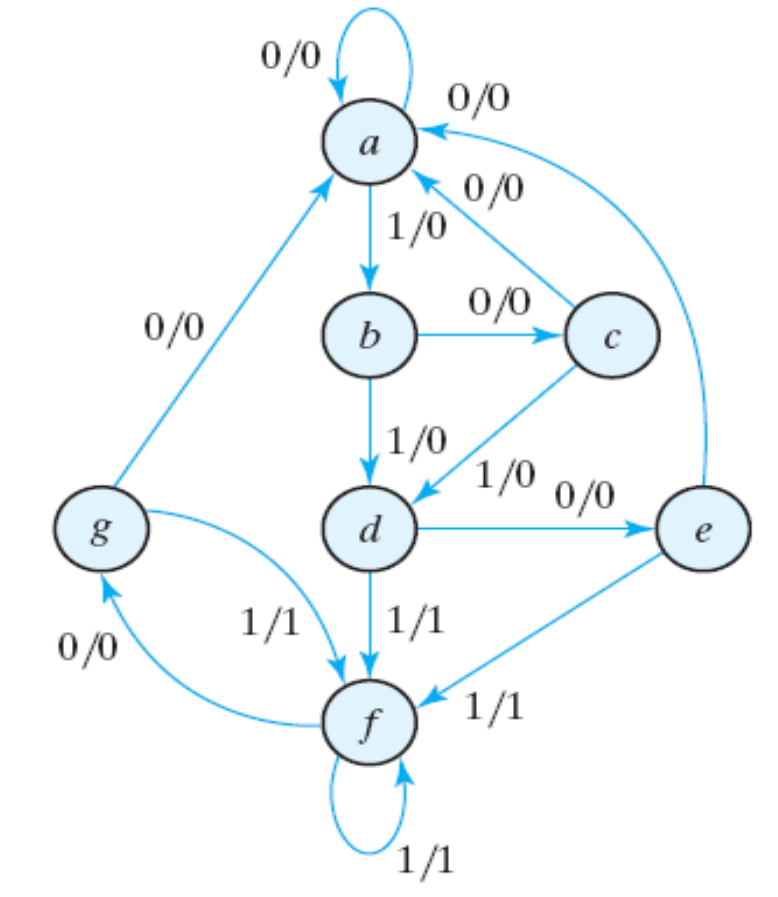


Fig. 5-18 Sequential Circuit with JK Flip-Flop

State Reduction (1)

- Consider a sequential circuit with the following diagram



State Reduction (2)

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

State **e** and **f** are equivalent \rightarrow have the same next states and same output when $x=0$, $x=1$

State Table

Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
<i>a</i>	<i>a</i>	<i>b</i>	0	0
<i>b</i>	<i>c</i>	<i>d</i>	0	0
<i>c</i>	<i>a</i>	<i>d</i>	0	0
<i>d</i>	<i>e</i>	<i>f</i>	0	1
\rightarrow <i>e</i>	<i>a</i>	<i>f</i>	0	1
<i>f</i>	<i>g</i>	<i>f</i>	0	1
\rightarrow <i>g</i>	<i>a</i>	<i>f</i>	0	1

Reducing the State Table

Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
<i>a</i>	<i>a</i>	<i>b</i>	0	0
<i>b</i>	<i>c</i>	<i>d</i>	0	0
<i>c</i>	<i>a</i>	<i>d</i>	0	0
<i>d</i>	<i>e</i>	<i>f</i>	0	1
<i>e</i>	<i>a</i>	<i>f</i>	0	1
<i>f</i>	<i>e</i>	<i>f</i>	0	1

\swarrow Remove **g** and replace it by **e** in remaining next states

State Reduction (3)

Reducing the State Table

Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
<i>a</i>	<i>a</i>	<i>b</i>	0	0
<i>b</i>	<i>c</i>	<i>d</i>	0	0
<i>c</i>	<i>a</i>	<i>d</i>	0	0
→ <i>d</i>	<i>e</i>	<i>f</i>	0	1
<i>e</i>	<i>a</i>	<i>f</i>	0	1
→ <i>f</i>	<i>e</i>	<i>f</i>	0	1

Reduced State Table

Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
<i>a</i>	<i>a</i>	<i>b</i>	0	0
<i>b</i>	<i>c</i>	<i>d</i>	0	0
<i>c</i>	<i>a</i>	<i>d</i>	0	0
<i>d</i>	<i>e</i>	<i>d</i>	0	1
<i>e</i>	<i>a</i>	<i>d</i>	0	1

f and *d* are
equivalent

END of Chapter 5

Note: several examples are discussed in the class