

# 計算機結構 (暫存器單元的設計)

陳鍾誠 於金門大學

# 第 10 章

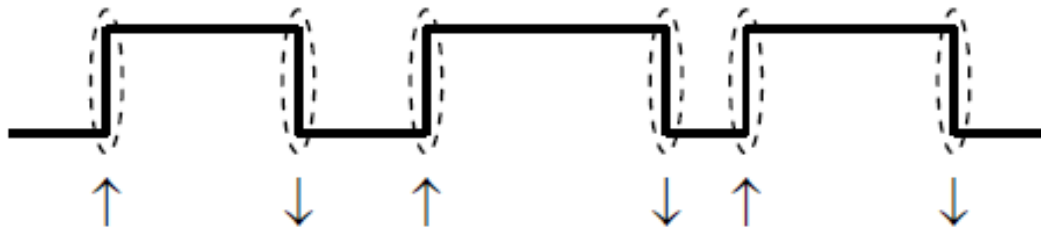
## *Memory Cells ( 記憶單元 )*

Computer Organization and Design Fundamental

書籍作者：David Tarnoff

投影片製作者：陳鍾誠

# 10.1 New Truth Table Symbols



**Figure 10-1** Symbols for Rising Edge and Falling Edge Transitions

## *10.1.1 Edges/Transitions*

## *10.1.2 Previously Stored Values*

If a memory cell is powered, it contains a stored value. We don't know whether that value is a one or a zero, but there is something stored in that cell. If a logic circuit uses the stored value of that cell as an input, we need to have a way of labeling it so that it can be used within a boolean expression.

Just as A, B, C, and D are used to represent inputs and X is used to represent an output, a standard letter is used to represent stored values. That letter is *Q*. To indicate that we are referring to the last value of Q stored, we use *Q<sub>0</sub>*.

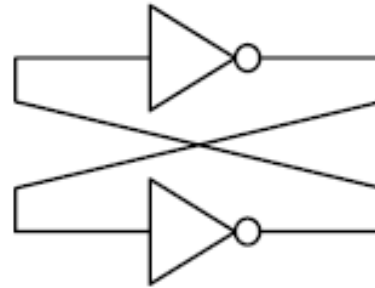
## *10.1.3 Undefined Values*

Door	Magnetron	Light		D	M	L
Closed	Off	Off		0	0	0
Closed	On	On	$\Rightarrow$	0	1	1
Open	Off	On		1	0	1
Open	On	Shouldn't happen		1	1	U

**Figure 10-2** Sample Truth Table Using Undefined Output

# **10.2 The S-R Latch**

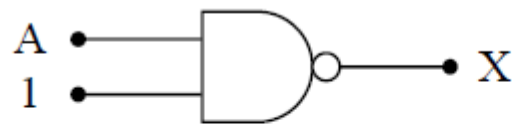
用 not 鎖住位元



**Figure 10-3** Primitive Feedback Circuit using Inverters

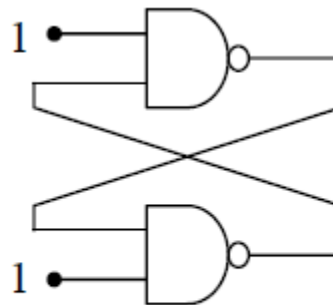


# 用 AND 鎖住位元

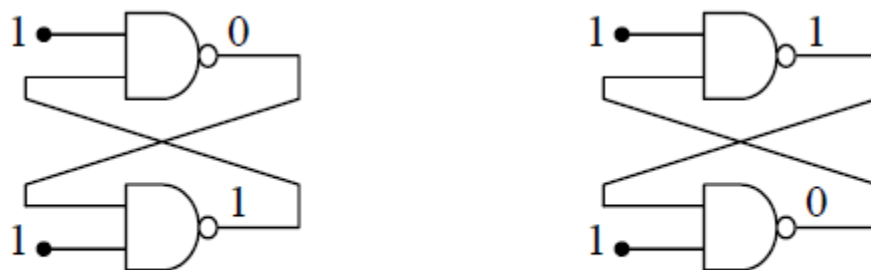


A	1	X
0	1	1
1	1	0

**Figure 10-4** Operation of a NAND Gate with One Input Tied High

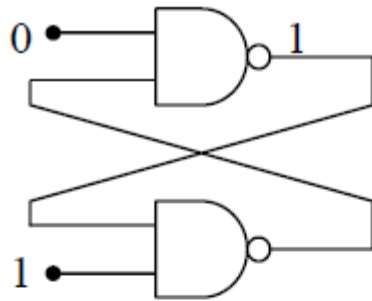


**Figure 10-5** Primitive Feedback Circuit Redrawn with NAND Gates

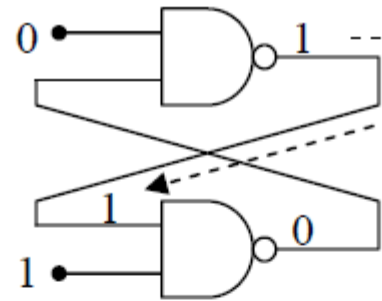


**Figure 10-6** Only Two Possible States of Circuit in Figure 10-5

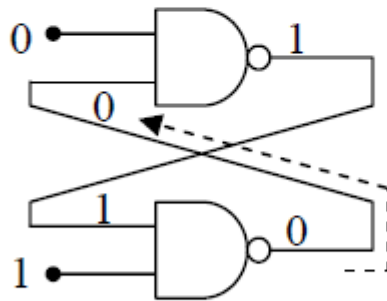
# S-R Latch 的運作原理



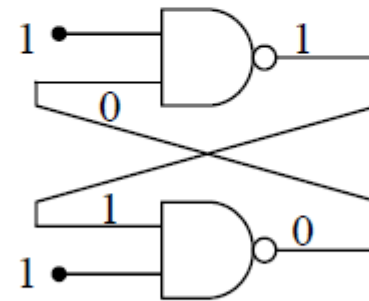
- a.) A zero to the free input of the top NAND gate forces a one to its output



- b.) That one passes to the bottom NAND which in turn outputs a zero



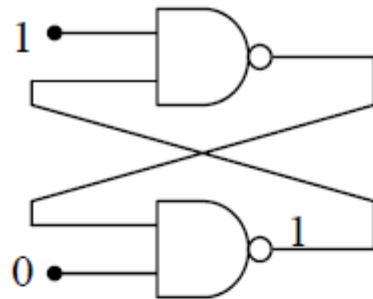
- c.) A zero from the bottom NAND returns to the lower input of the top NAND



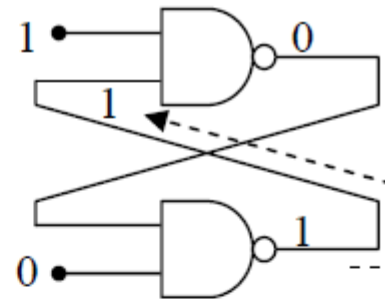
- d.) The second zero at the top NAND holds its output even if the free input returns to 1

Figure 10-7 Operation of a Simple Memory Cell

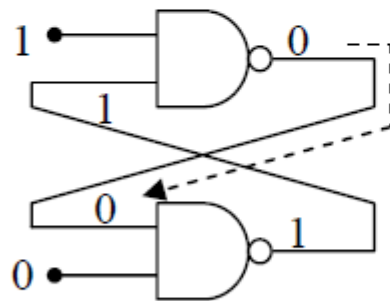
# S-R Latch 的運作原理 (續)



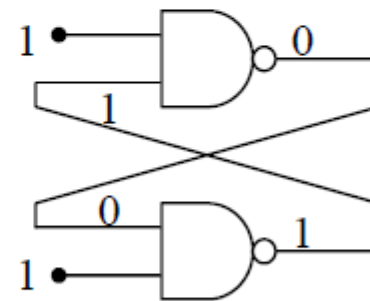
- a.) A zero to the free input of the bottom NAND gate forces a one to its output



- b.) That one passes to the top NAND which in turn outputs a zero



- c.) A zero from the top NAND returns to the lower input of the bottom NAND



- d.) The second zero at the bottom NAND holds its output even if the free input returns to 1

Figure 10-8 Operation of a Simple Memory Cell (continued)

# S-R Latch 的真值表

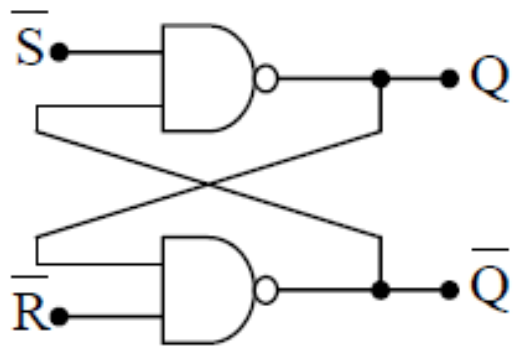


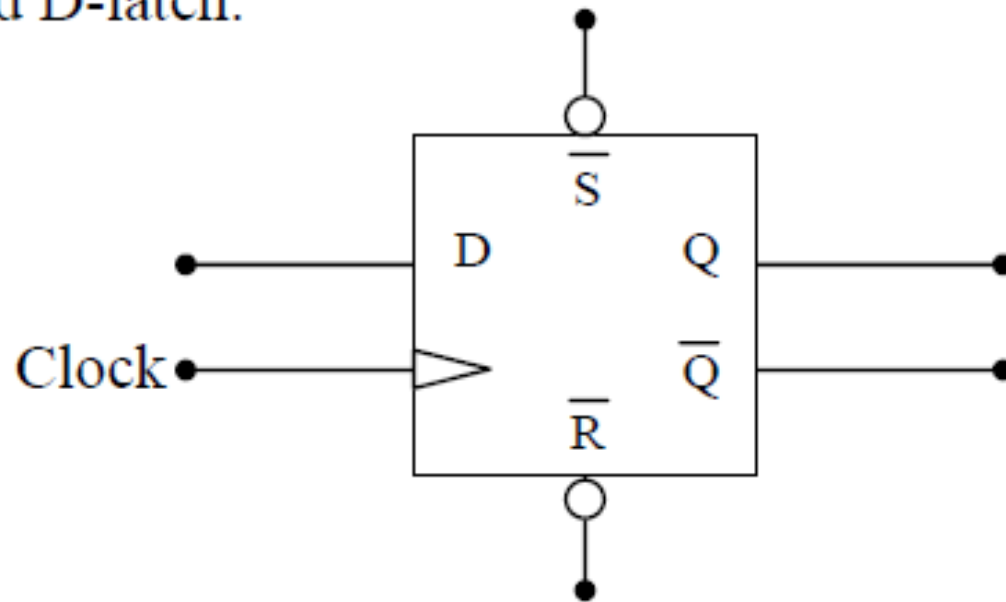
Figure 10-9 S-R Latch

$\overline{S}$	$\overline{R}$	$Q$	$\overline{Q}$
0	0	U	U
0	1	1	0
1	0	0	1
1	1	$Q_0$	$\overline{Q}_0$

Figure 10-10 S-R Latch Truth Table

## 10.3 The D Latch

implemented D-latch.



**Figure 10-11** Block Diagram of the D Latch

# Edge-Triggered **D Latch**

D	Clock	Q	$\overline{Q}$
X	0	$Q_0$	$\overline{Q_0}$
X	1	$Q_0$	$\overline{Q_0}$
X	↓	$Q_0$	$\overline{Q_0}$
0	↑	0	1
1	↑	1	0

a.) Rising Edge

D	Clock	Q	$\overline{Q}$
X	0	$Q_0$	$\overline{Q_0}$
X	1	$Q_0$	$\overline{Q_0}$
X	↑	$Q_0$	$\overline{Q_0}$
0	↓	0	1
1	↓	1	0

b.) Falling Edge

**Figure 10-12** Edge-Triggered D Latch Truth Tables

# Active Low/Active High D-Latch

D	Clock	Q	$\overline{Q}$
X	1	$Q_0$	$\overline{Q_0}$
0	0	0	1
1	0	1	0

a.) Active Low

D	Clock	Q	$\overline{Q}$
X	0	$Q_0$	$\overline{Q_0}$
0	1	0	1
1	1	1	0

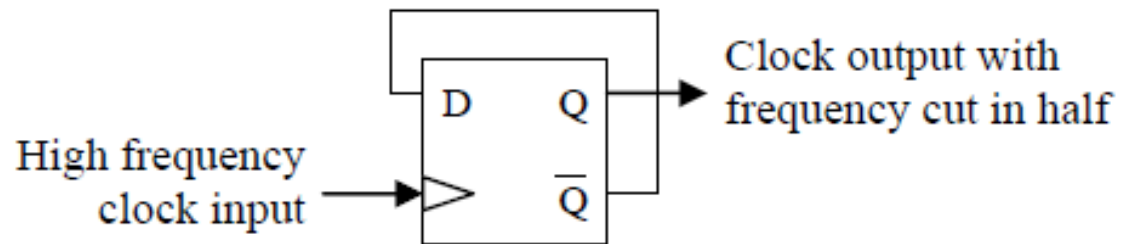
b.) Active High

**Figure 10-13** Transparent D Latch Truth Tables

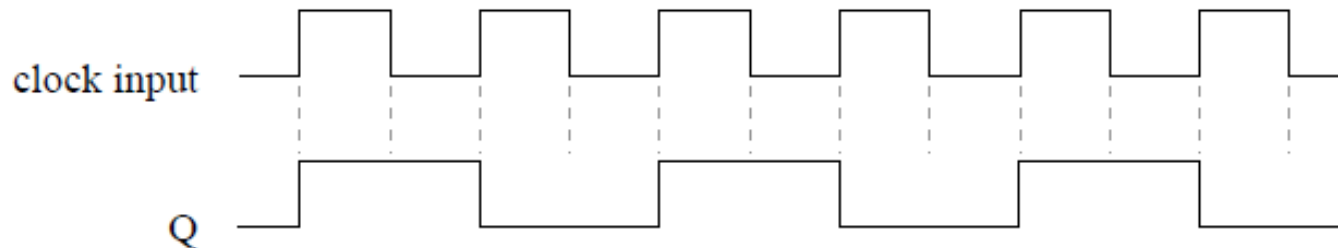


# **10.4 Divide-By-Two Circuit**

# 10.4 Divide-By-Two Circuit

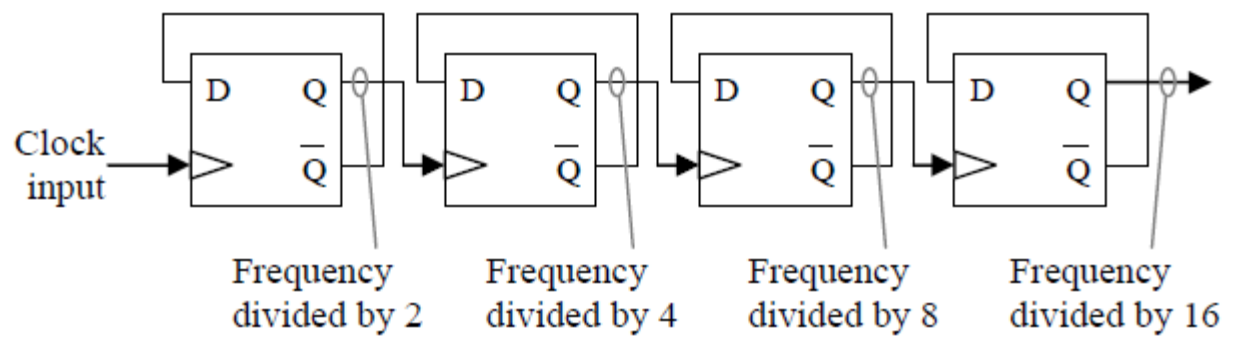


**Figure 10-14** Divide-By-Two Circuit

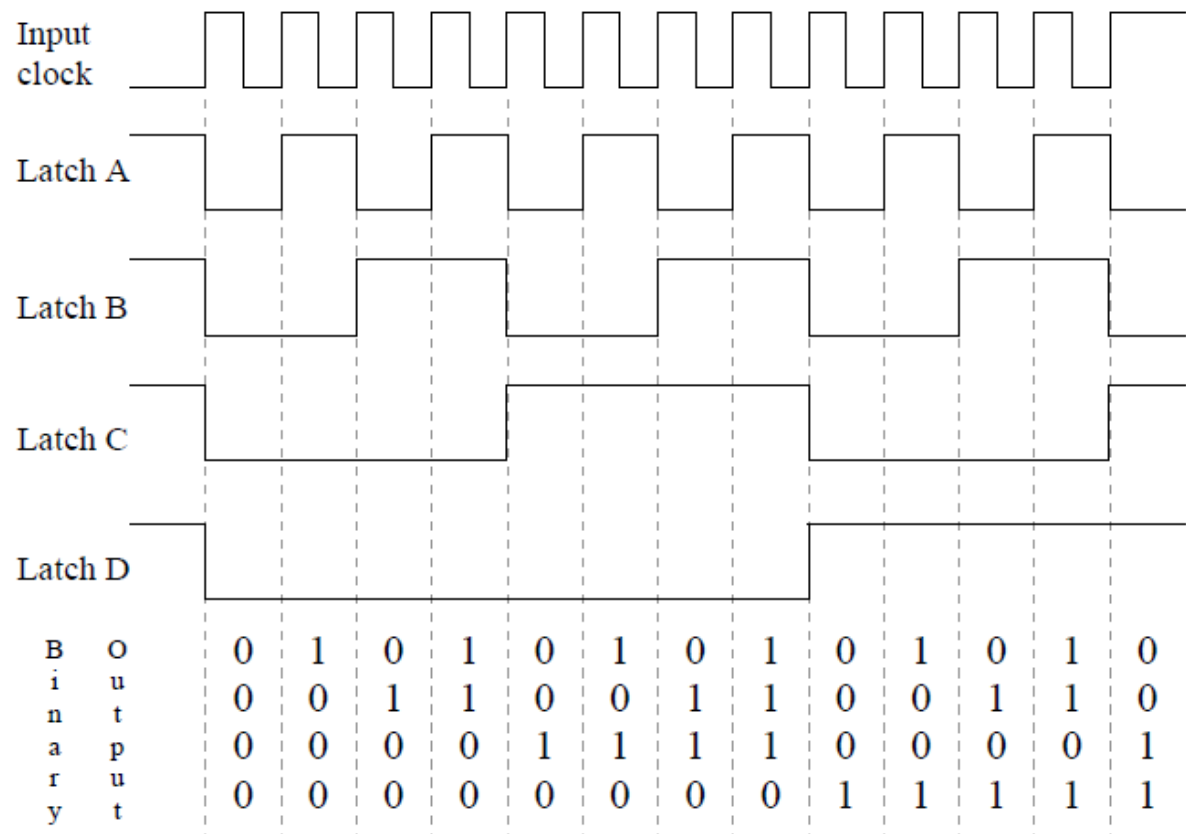


**Figure 10-15** Clock and Output Timing in a Divide-By-Two Circuit

# 10.5 Counter

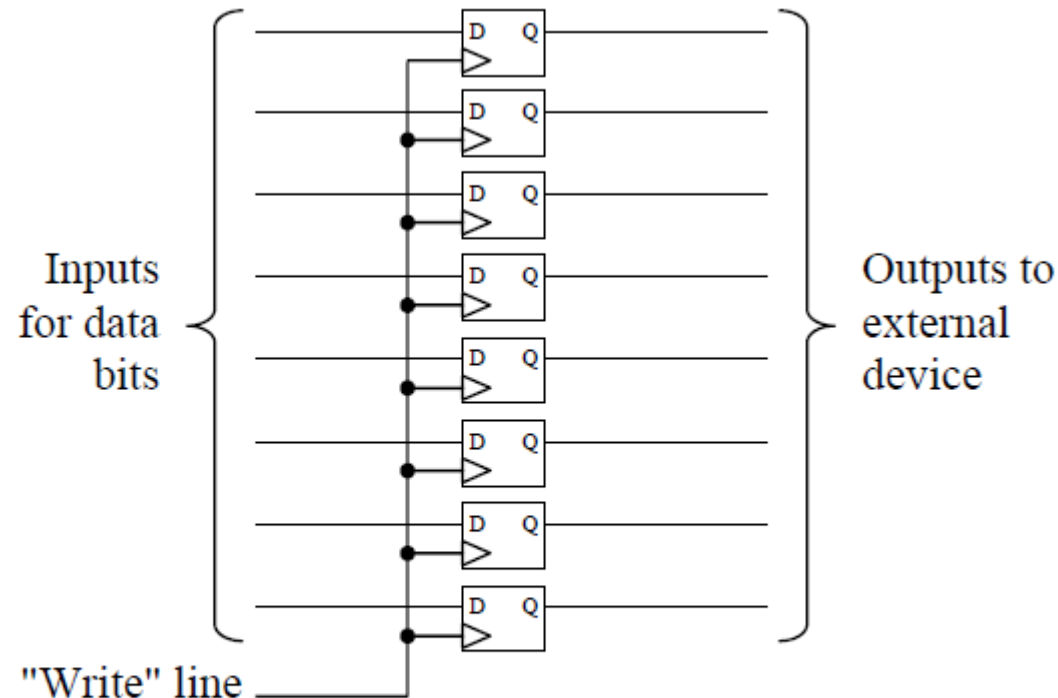


**Figure 10-16** Cascading Four Divide-By-Two Circuits



**Figure 10-18** Output of Binary Counter Circuit

# 10.6 Parallel Data Output



**Figure 10-19** Output Port Data Latch Circuitry

# 1 位元記憶元件

陳鍾誠 於金門大學

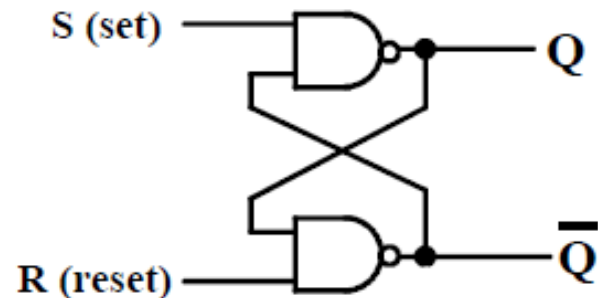
# 門瑣 (Latch)

陳鍾誠 於金門大學

# S-R 閥瑣 (Latch)

## Basic (NAND) $\bar{S} - \bar{R}$ Latch

- “Cross-Coupling”  
two NAND gates gives  
the  $\bar{S} - \bar{R}$  Latch:
- Which has the time  
sequence behavior:



- $S = 0, R = 0$  is  
forbidden as  
input pattern

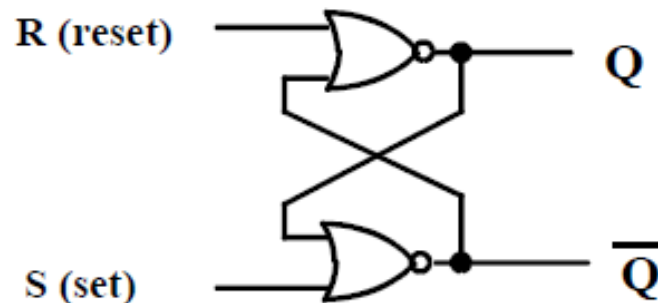
Time ↓

R	S	Q	$\bar{Q}$	Comment
1	1	?	?	Stored state unknown
1	0	1	0	“Set” Q to 1
1	1	1	0	Now Q “remembers” 1
0	1	0	1	“Reset” Q to 0
1	1	0	1	Now Q “remembers” 0
0	0	1	1	Both go high
1	1	?	?	Unstable!



## Basic (NOR) S – R Latch

- Cross-coupling two NOR gates gives the S – R Latch:

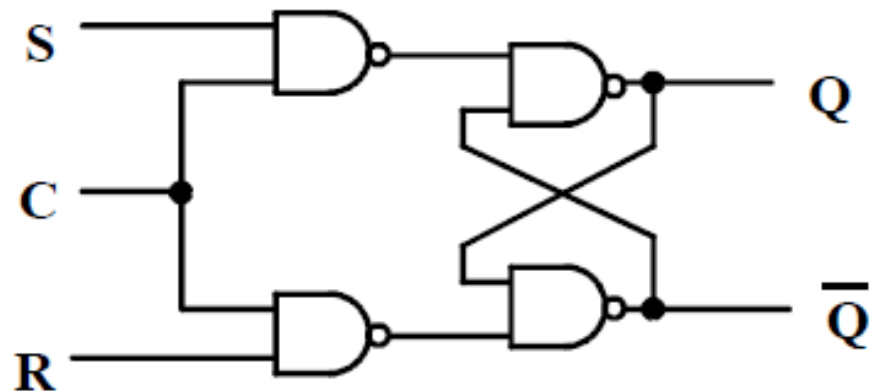


- Which has the time sequence behavior:

Time	R	S	Q	$\bar{Q}$	Comment
	0	0	?	?	Stored state unknown
	0	1	1	0	“Set” Q to 1
	0	0	1	0	Now Q “remembers” 1
	1	0	0	1	“Reset” Q to 0
	0	0	0	1	Now Q “remembers” 0
	1	1	0	0	Both go low
	0	0	?	?	Unstable!

## Clocked S - R Latch

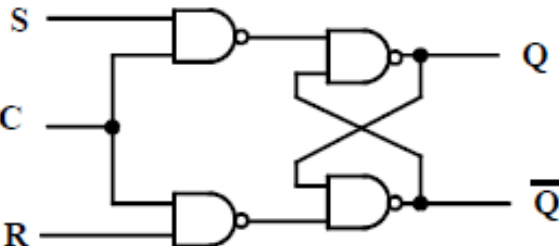
- Adding two NAND gates to the basic  $\overline{S}$  -  $\overline{R}$  NAND latch gives the clocked S - R latch:



- Has a time sequence behavior similar to the basic S-R latch except that the S and R inputs are only observed when the line C is high.
- C means “control” or “clock”.

## Clocked S - R Latch (continued)

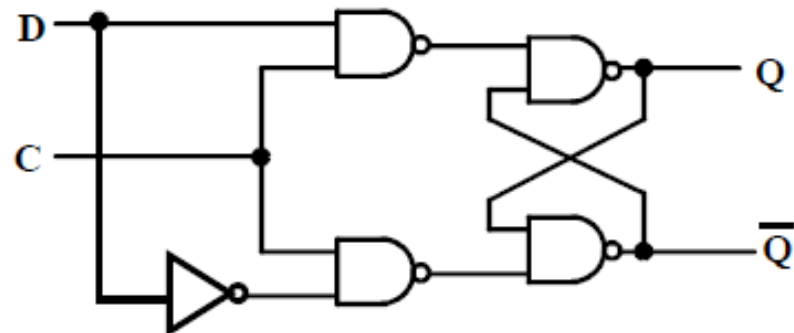
- The Clocked S-R Latch can be described by a table:

				Q(t)	S	R	Q(t+1)	Comment
				0	0	0	0	No change
				0	0	1	0	Clear Q
				0	1	0	1	Set Q
				0	1	1	???	Indeterminate
				1	0	0	1	No change
				1	0	1	0	Clear Q
				1	1	0	1	Set Q
				1	1	1	???	Indeterminate

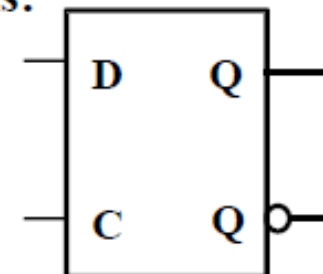
- The table describes what happens after the clock [at time (t+1)] based on:
  - current inputs (S,R) and
  - current state Q(t).

## D Latch

- Adding an inverter to the S-R Latch, gives the D Latch:
- Note that there are no “indeterminate” states!



The graphic symbol for a D Latch is:



Q	D	Q(t+1)	Comment
0	0	0	No change
0	1	1	Set Q
1	0	0	Clear Q
1	1	1	No Change

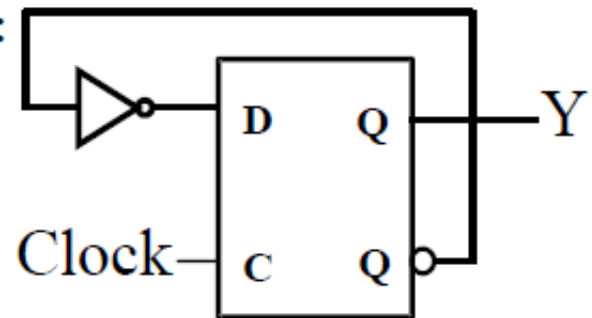
# The Latch Timing Problem

---

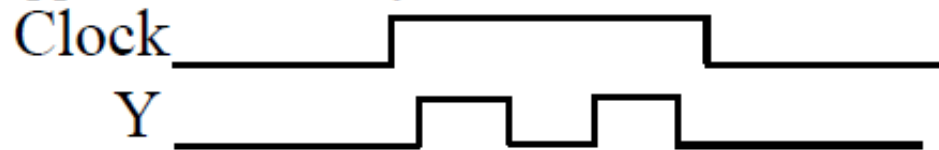
- In a sequential circuit, paths may exist through combinational logic:
  - From one storage element to another
  - From a storage element back to the same storage element
- The combinational logic between a latch output and a latch input may be as simple as an interconnect
- For a clocked D-latch, the output  $Q$  depends on the input  $D$  whenever the clock input  $C$  has value 1

## The Latch Timing Problem (continued)

- Consider the following circuit:



- Suppose that initially  $Y = 0$ .



- As long as  $C = 1$ , the value of  $Y$  continues to change!
- The changes are based on the delay present on the loop through the connection from  $Y$  back to  $Y$ .
- This behavior is clearly unacceptable.
- Desired behavior:  $Y$  changes only once per clock pulse

## The Latch Timing Problem (continued)

---

- A solution to the latch timing problem is to break the closed path from Y to Y within the storage element
- The commonly-used, path-breaking solutions replace the clocked D-latch with:
  - a master-slave flip-flop
  - an edge-triggered flip-flop

# 正反器 (Flip-Flop)

陳鍾誠 於金門大學



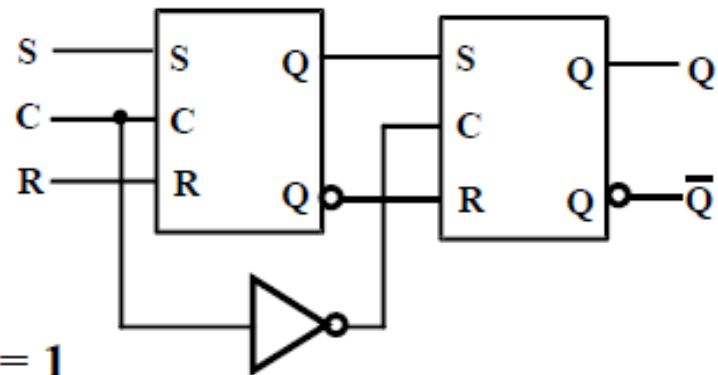
# Flip-Flops

---

- The latch timing problem
- Master-slave flip-flop
- Edge-triggered flip-flop
- Standard symbols for storage elements
- Direct inputs to flip-flops

## S-R Master-Slave Flip-Flop

- Consists of two clocked S-R latches in series with the clock on the second latch inverted
- The input is observed by the first latch with  $C = 1$
- The output is changed by the second latch with  $C = 0$
- The path from input to output is broken by the difference in clocking values ( $C = 1$  and  $C = 0$ ).
- The behavior demonstrated by the example with D driven by Y given previously is prevented since the clock must change from 1 to 0 before a change in Y based on D can occur.



## Flip-Flop Problem

---

- The change in the flip-flop output is delayed by the pulse width which makes the circuit slower or
- S and/or R are permitted to change while  $C = 1$ 
  - Suppose  $Q = 0$  and S goes to 1 and then back to 0 with R remaining at 0
    - The master latch sets to 1
    - A 1 is transferred to the slave
  - Suppose  $Q = 0$  and S goes to 1 and back to 0 and R goes to 1 and back to 0
    - The master latch sets and then resets
    - A 0 is transferred to the slave
  - This behavior is called *1s catching*

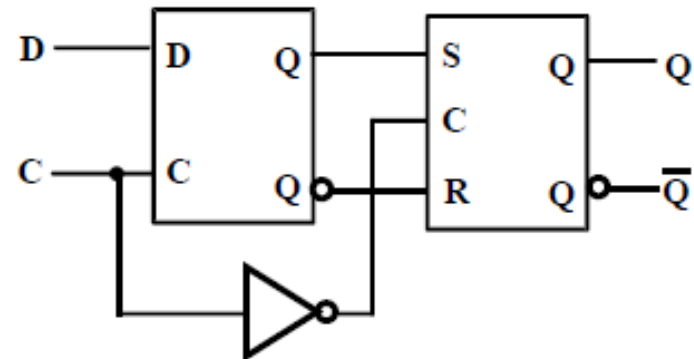
## Flip-Flop Solution

---

- Use edge-triggering instead of master-slave
- An *edge-triggered* flip-flop ignores the pulse while it is at a constant level and triggers only during a transition of the clock signal
- Edge-triggered flip-flops can be built directly at the electronic circuit level, or
- A master-slave D flip-flop which also exhibits edge-triggered behavior can be used.

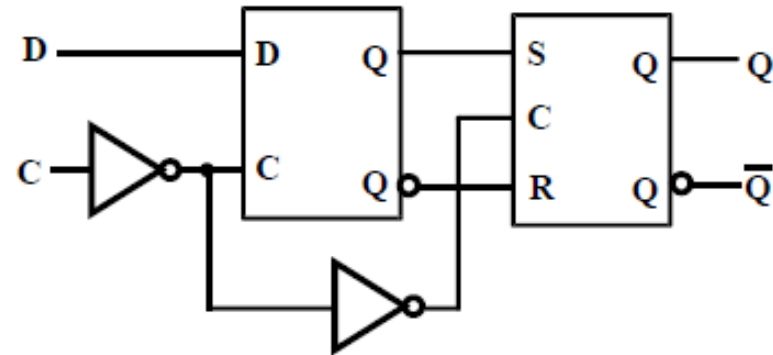
# Edge-Triggered D Flip-Flop

- The edge-triggered D flip-flop is the same as the master-slave D flip-flop
- It can be formed by:
  - Replacing the first clocked S-R latch with a clocked D latch or
  - Adding a D input and inverter to a master-slave S-R flip-flop
- The delay of the S-R master-slave flip-flop can be avoided since the 1s-catching behavior is not present with D replacing S and R inputs
- The change of the D flip-flop output is associated with the negative edge at the end of the pulse
- It is called a *negative-edge triggered flip-flop*



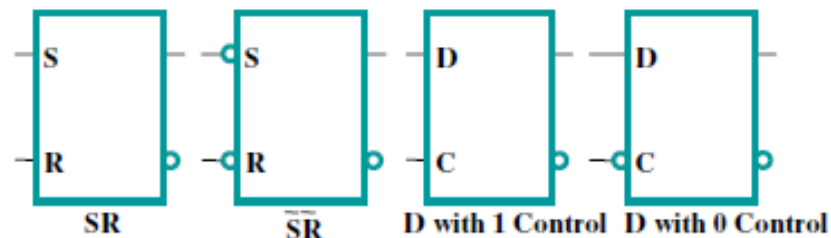
## Positive-Edge Triggered D Flip-Flop

- Formed by adding inverter to clock input



- Q changes to the value on D applied at the positive clock edge within timing constraints to be specified
- Our choice as the standard flip-flop for most sequential circuits

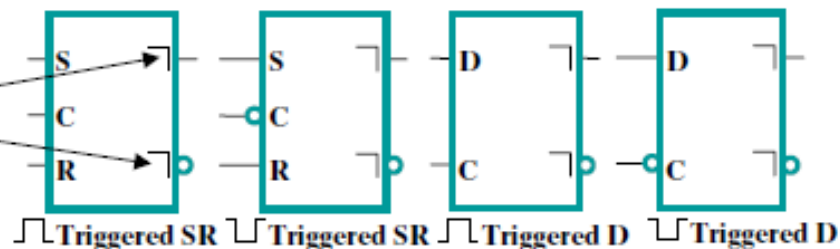
# Standard Symbols for Storage Elements



(a) Latches

- Master-Slave:**

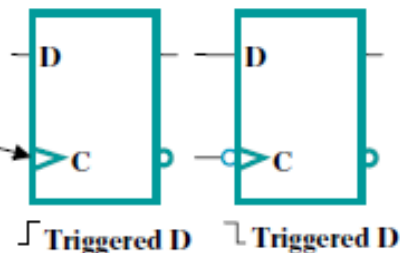
**Postponed output indicators**



(b) Master-Slave Flip-Flops

- Edge-Triggered:**

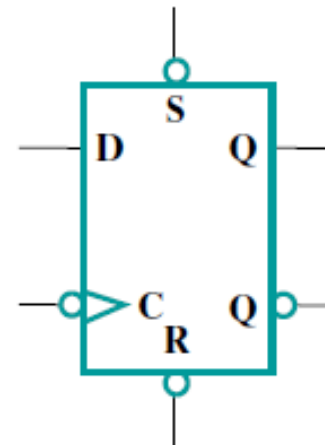
**Dynamic indicator**



(c) Edge-Triggered Flip-Flops

# Direct Inputs

- At power up or at reset, all or part of a sequential circuit usually is initialized to a known state before it begins operation
- This initialization is often done outside of the clocked behavior of the circuit, i.e., asynchronously.
- Direct R and/or S inputs that control the state of the latches within the flip-flops are used for this initialization.
- For the example flip-flop shown
  - 0 applied to  $\overline{R}$  resets the flip-flop to the 0 state
  - 0 applied to  $\overline{S}$  sets the flip-flop to the 1 state





# J-K Flip-flop

---

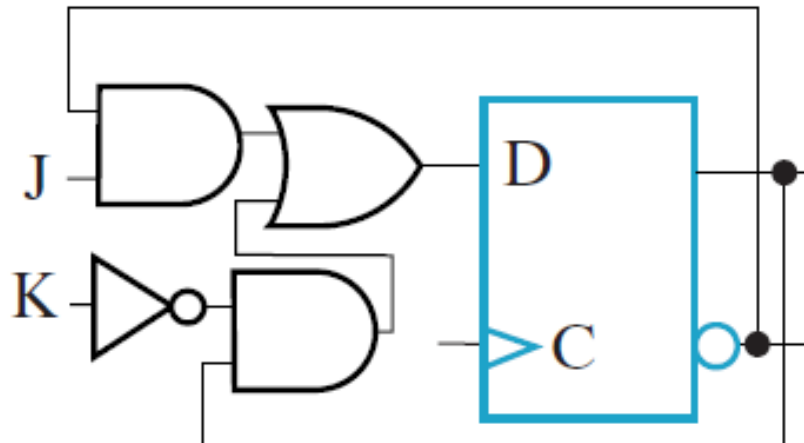
## ■ Behavior

- Same as S-R flip-flop with J analogous to S and K analogous to R
- Except that  $J = K = 1$  is allowed, and
- For  $J = K = 1$ , the flip-flop changes to the *opposite state*
- As a master-slave, has same “1s catching” behavior as S-R flip-flop
- If the master changes to the wrong state, that state will be passed to the slave
  - E.g., if master falsely set by  $J = 1, K = 1$  cannot reset it during the current clock cycle

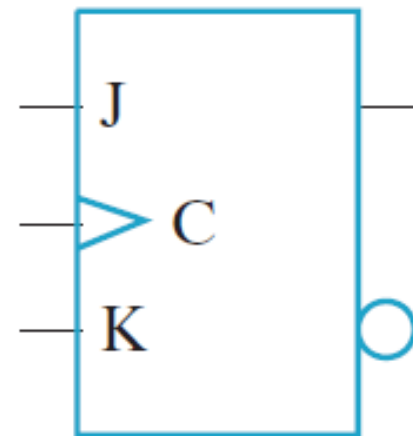
## J-K Flip-flop (continued)

### ■ Implementation

- To avoid 1s catching behavior, one solution used is to use an edge-triggered D as the core of the flip-flop



### ■ Symbol



# T Flip-flop

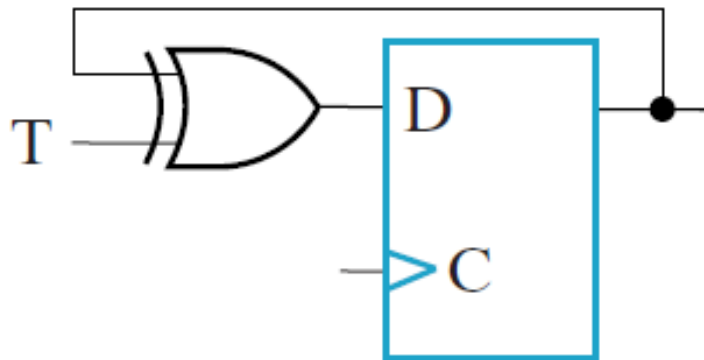
---

- **Behavior**
  - Has a single input T
    - For  $T = 0$ , no change to state
    - For  $T = 1$ , changes to opposite state
- Same as a J-K flip-flop with  $J = K = T$
- As a master-slave, has same “1s catching” behavior as J-K flip-flop
- Cannot be initialized to a known state using the T input
  - Reset (asynchronous or synchronous) essential

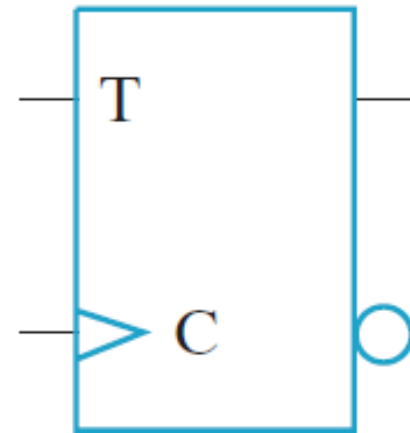
## T Flip-flop (continued)

### ■ Implementation

- To avoid 1s catching behavior, one solution used is to use an edge-triggered D as the core of the flip-flop



### ■ Symbol



# 正反器電路描述

陳鍾誠 於金門大學

# Basic Flip-Flop Descriptors

---

- **Used in analysis**
  - *Characteristic table* - defines the next state of the flip-flop in terms of flip-flop inputs and current state
  - *Characteristic equation* - defines the next state of the flip-flop as a Boolean function of the flip-flop inputs and the current state
- **Used in design**
  - *Excitation table* - defines the flip-flop input variable values as function of the current state and next state

# D Flip-Flop Descriptors

---

- **Characteristic Table**

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

- **Characteristic Equation**

$$Q(t+1) = D$$

- **Excitation Table**

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

# T Flip-Flop Descriptors

---

- **Characteristic Table**

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

- **Characteristic Equation**

$$Q(t+1) = T \oplus Q$$

- **Excitation Table**

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



# S-R Flip-Flop Descriptors

- Characteristic Table

S	R	Q(t+1)	Operation
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	?	Undefined

- Characteristic Equation

$$Q(t+1) = S + \overline{R} Q, S \cdot R = 0$$

- Excitation Table

Q(t)	Q(t+1)	S	R	Operation
0	0	0	X	No change
0	1	1	0	Set
1	0	0	1	Reset
1	1	X	0	No change

# J-K Flip-Flop Descriptors

- Characteristic Table

J	K	Q(t+1)	Operation
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$\bar{Q}(t)$	Complement

- Characteristic Equation

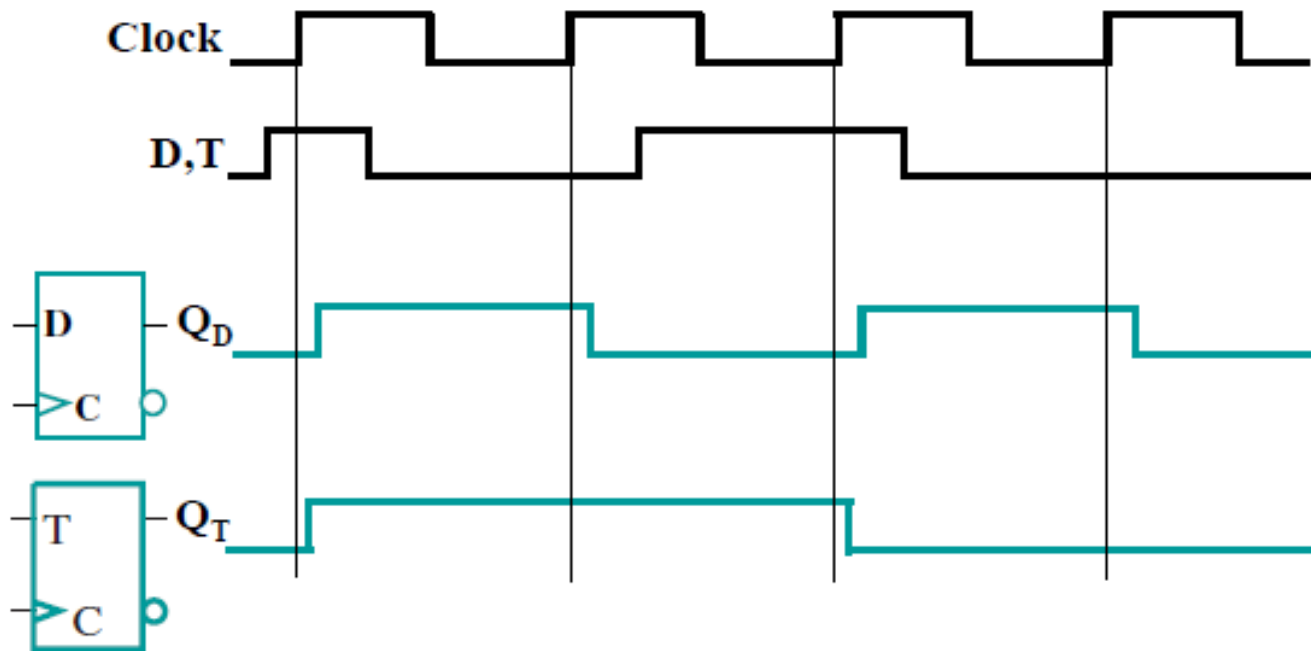
$$Q(t+1) = J \bar{Q} + \bar{K} Q$$

- 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

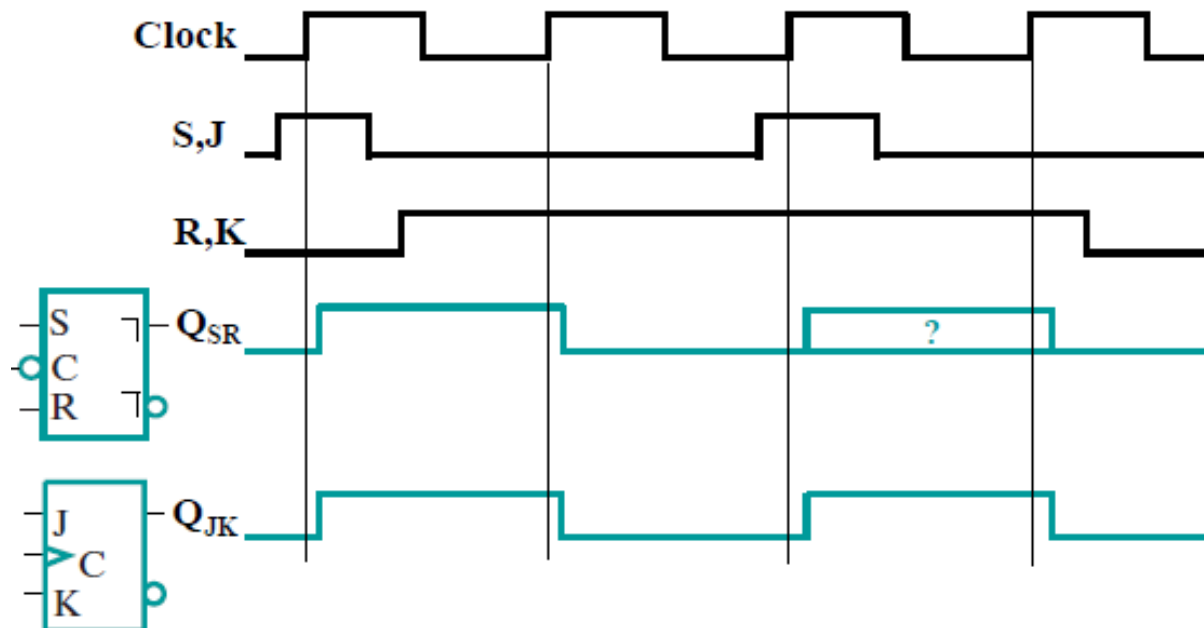
# Flip-flop Behavior Example

- Use the characteristic tables to find the output waveforms for the flip-flops shown:



## Flip-Flop Behavior Example (continued)

- Use the characteristic tables to find the output waveforms for the flip-flops shown:



# 循序電路

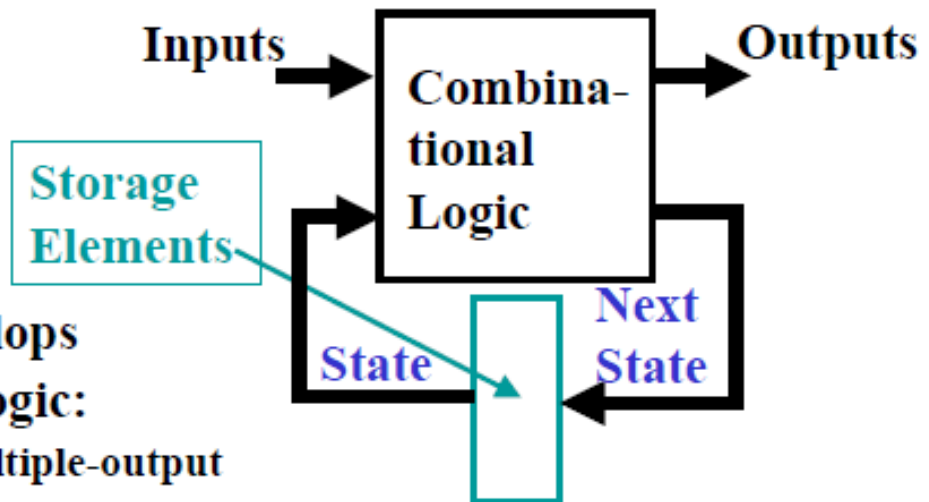
陳鍾誠 於金門大學

# Introduction to Sequential Circuits

---

- A Sequential circuit contains:

- Storage elements: Latches or Flip-Flops
- Combinational Logic:
  - Implements a multiple-output switching function
  - Inputs are signals from the outside.
  - Outputs are signals to the outside.
  - Other inputs, State or Present State, are signals from storage elements.
  - The remaining outputs, Next State are inputs to storage elements.



# Types of Sequential Circuits

---

- Depends on the times at which:
  - storage elements observe their inputs, and
  - storage elements change their state
- Synchronous
  - Behavior defined from knowledge of its signals at discrete instances of time
  - Storage elements observe inputs and can change state only in relation to a timing signal (clock pulses from a clock)
- Asynchronous
  - Behavior defined from knowledge of inputs at any instant of time and the order in continuous time in which inputs change
  - If clock just regarded as another input, all circuits are asynchronous!
  - Nevertheless, the synchronous abstraction makes complex designs tractable!

# 模擬

## Discrete Event Simulation

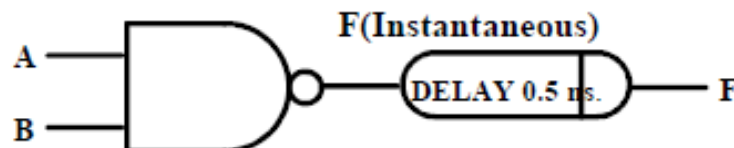
---

- In order to understand the time behavior of a sequential circuit we use discrete event simulation.
- Rules:
  - Gates modeled by an ideal (instantaneous) function and a fixed gate delay
  - Any change in input values is evaluated to see if it causes a change in output value
  - Changes in output values are scheduled for the fixed gate delay after the input change
  - At the time for a scheduled output change, the output value is changed along with any inputs it drives



## Simulated NAND Gate

- Example: A 2-Input NAND gate with a 0.5 ns. delay:



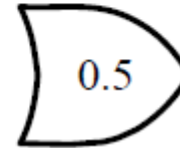
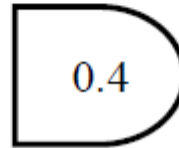
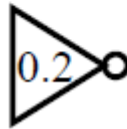
- Assume A and B have been 1 for a long time
- At time  $t=0$ , A changes to a 0 at  $t=0.8$  ns, back to 1.

t (ns)	A	B	F(I)	F	Comment
$-\infty$	1	1	0	0	A=B=1 for a long time
0	$1 \Rightarrow 0$	1	$1 \Leftarrow 0$	0	F(I) changes to 1
0.5	0	1	1	$1 \Leftarrow 0$	F changes to 1 after a 0.5 ns delay
0.8	$1 \Leftarrow 0$	1	$1 \Rightarrow 0$	1	F(Instantaneous) changes to 0
0.13	1	1	0	$1 \Rightarrow 0$	F changes to 0 after a 0.5 ns delay

## Gate Delay Models

---

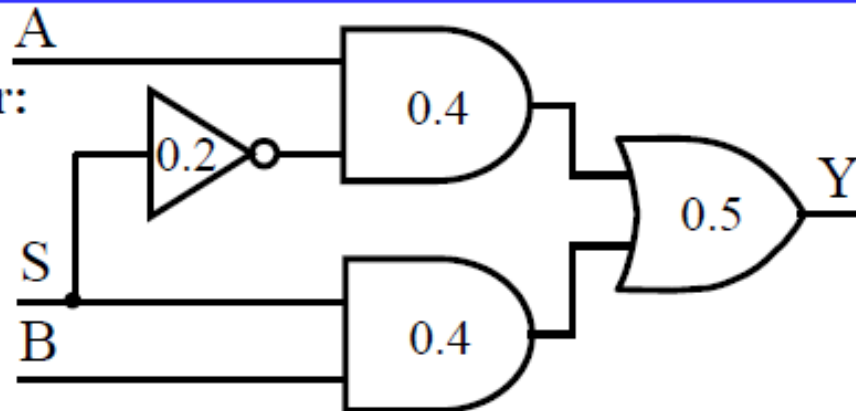
- Suppose gates with delay  $n$  ns are represented for  $n = 0.2$  ns,  $n = 0.4$  ns,  $n = 0.5$  ns, respectively:



# Circuit Delay Model

- Consider a simple 2-input multiplexer:
- With function:

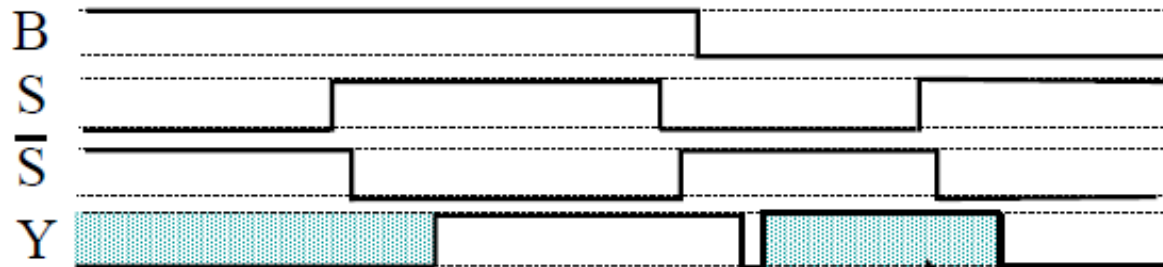
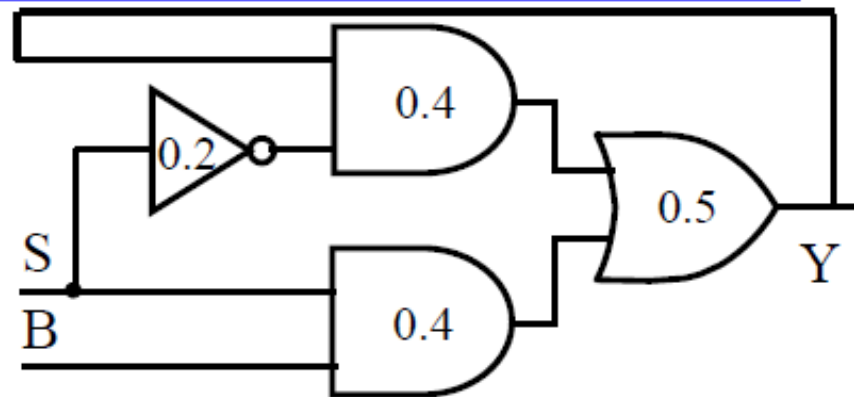
- $Y = A$  for  $S = 1$
- $Y = B$  for  $S = 0$



- "Glitch" is due to delay of inverter

## Storing State

- What if A connected to Y?
- Circuit becomes:
- With function:
  - $Y = B$  for  $S = 1$ , and  $Y(t)$  dependent on  $Y(t - 0.9)$  for  $S = 0$



- The simple combinational circuit has now become a sequential circuit because its output is a function of a time sequence of input signals!

Y is stored value in shaded area

## Storing State (Continued)

---

- Simulation example as input signals change with time. Changes occur every 100 ns, so that the tenths of ns delays are negligible.

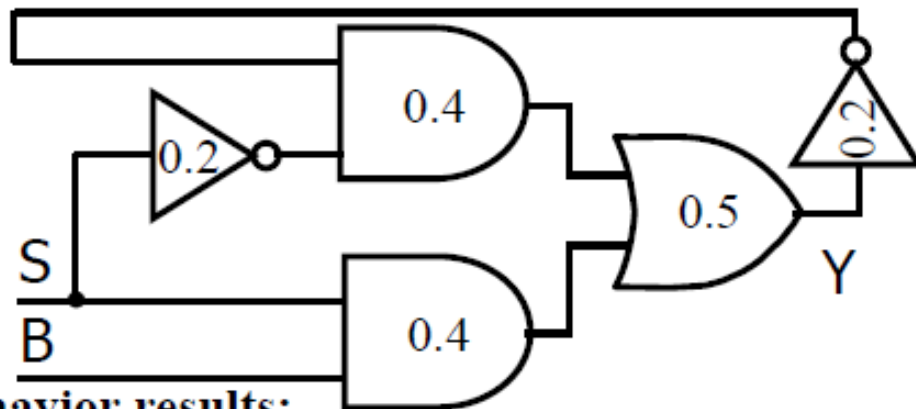
Time	B	S	Y	Comment
↓	1	0	0	Y “remembers” 0
	1	1	1	Y = B when S = 1
	1	0	1	Now Y “remembers” B = 1 for S = 0
	0	0	1	No change in Y when B changes
	0	1	0	Y = B when S = 1
	0	0	0	Y “remembers” B = 0 for S = 0
	1	0	0	No change in Y when B changes

- Y represent the state of the circuit, not just an output.

# 不穩定的電路（振蕩）

## Storing State (Continued)

- Suppose we place an inverter in the “feedback path.”



- The following behavior results:

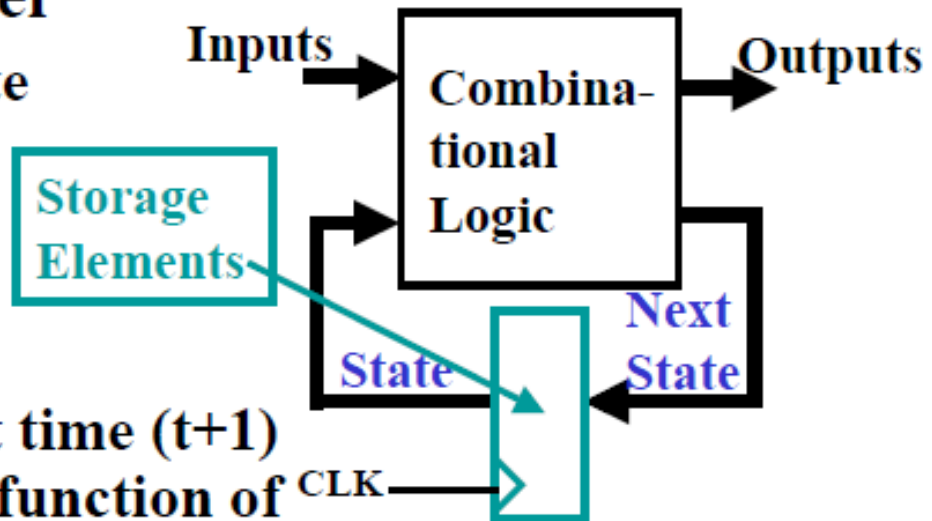
- The circuit is said to be unstable.
- For  $S = 0$ , the circuit has become what is called an *oscillator*. Can be used as crude clock.

B	S	Y	Comment
0	1	0	$Y = B$ when $S = 1$
1	1	1	
1	0	1	Now Y “remembers” A
1	0	0	Y, 1.1 ns later
1	0	1	Y, 1.1 ns later
1	0	0	Y, 1.1 ns later

# Sequential Circuit Analysis

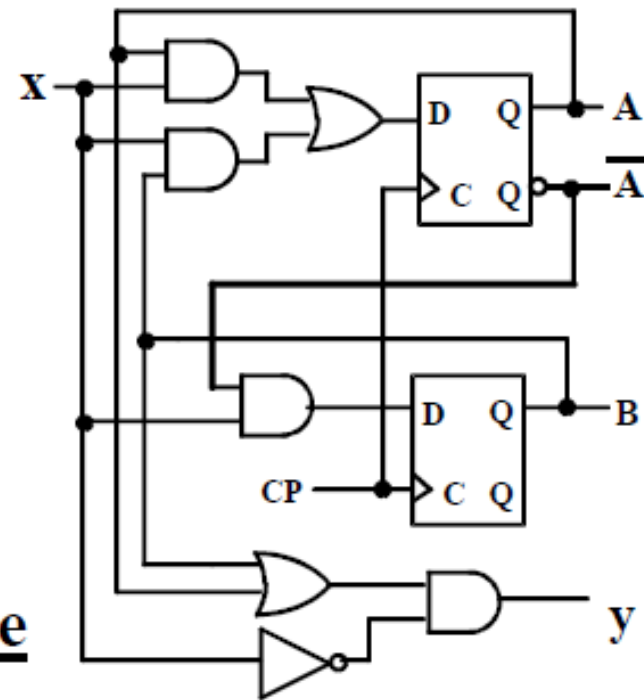
## ■ General Model

- Current State at time (t) is stored in an array of flip-flops.
- Next State at time (t+1) is a Boolean function of State and Inputs.
- Outputs at time (t) are a Boolean function of State (t) and (sometimes) Inputs (t).



## Example 1 (from Fig. 5-15)

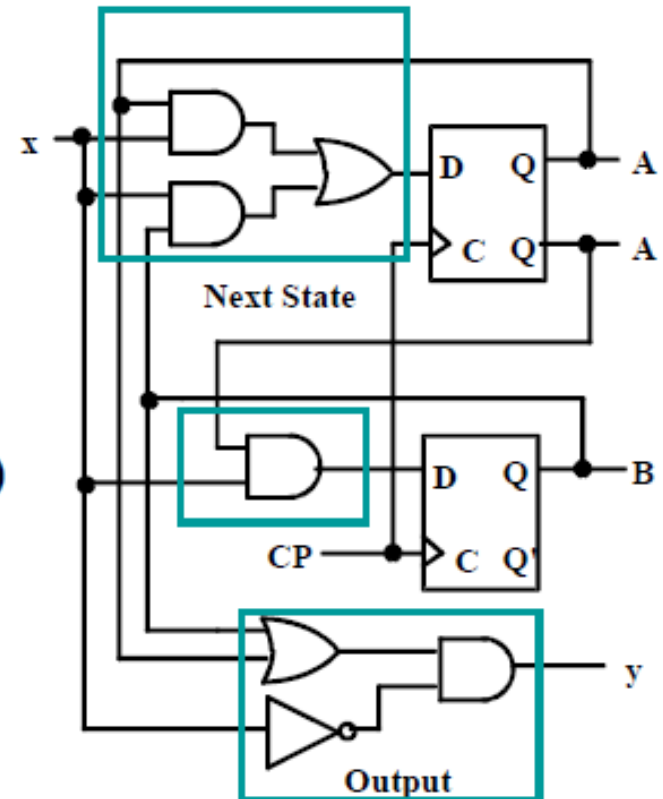
- **Input:**  $x(t)$
- **Output:**  $y(t)$
- **State:**  $(A(t), B(t))$
- **What is the Output Function?**
- **What is the Next State Function?**





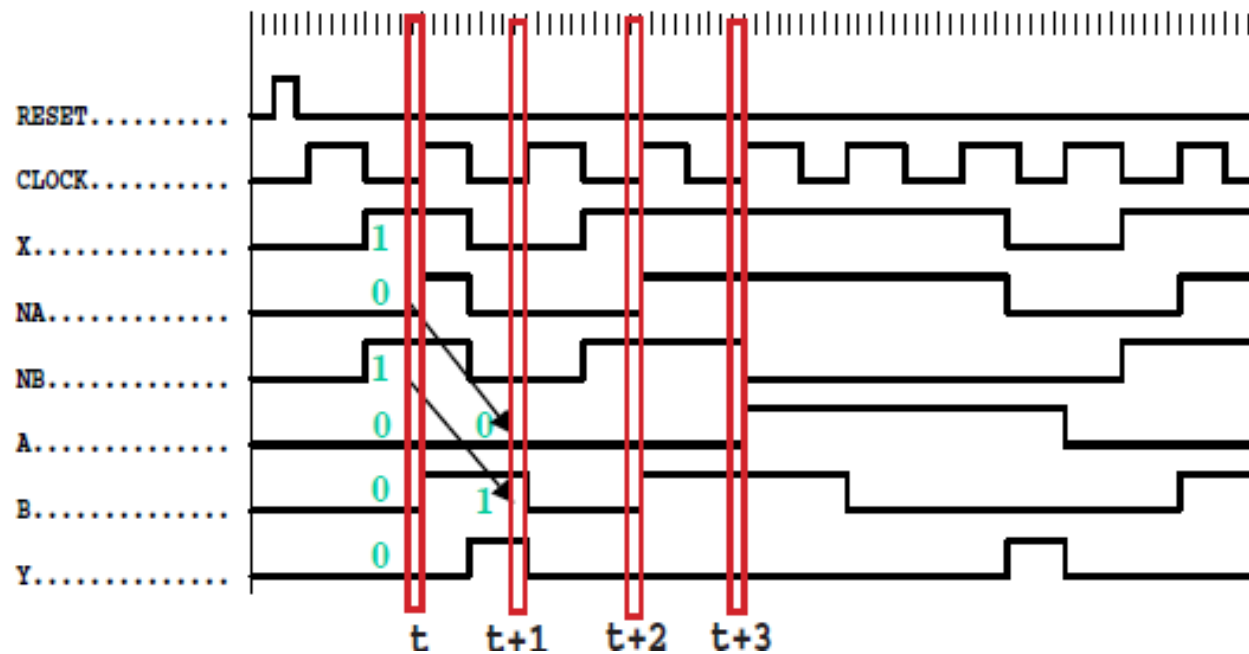
## Example 1 (from Fig. 5-15) (continued)

- Boolean equations for the functions:
  - $A(t+1) = A(t)x(t) + B(t)x(t)$
  - $B(t+1) = \bar{A}(t)x(t)$
  - $y(t) = \bar{x}(t)(B(t) + A(t))$



## Example 1(from Fig. 5-15) (continued)

- Where in time are inputs, outputs and states defined?



# State Table Characteristics

---

- *State table* – a multiple variable table with the following four sections:
  - *Present State* – the values of the state variables for each allowed state.
  - *Input* – the input combinations allowed.
  - *Next-state* – the value of the state at time  $(t+1)$  based on the present state and the input.
  - *Output* – the value of the output as a function of the present state and (sometimes) the input.
- From the viewpoint of a truth table:
  - the inputs are Input, Present State
  - and the outputs are Output, Next State

## Example 1: State Table (from Fig. 5-15)

---

- The state table can be filled in using the next state and output equations:  
 $A(t+1) = A(t)x(t) + \overline{B(t)}x(t)$   
 $B(t+1) = \overline{A(t)}x(t) + B(t)x(t)$   
 $y(t) = x(t)(B(t) + A(t))$

Present State		Input	Next State		Output
A(t)	B(t)	x(t)	A(t+1)	B(t+1)	y(t)
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: Alternate State Table

- 2-dimensional table that matches well to a K-map.  
Present state rows and input columns in Gray code order.
  - $A(t+1) = A(t)x(t) + B(t)x(t)$
  - $B(t+1) = \bar{A}(t)x(t)$
  - $y(t) = \bar{x}(t)(B(t) + A(t))$

Present State A(t) B(t)	Next State		Output	
	x(t)=0 A(t+1)B(t+1)	x(t)=1 A(t+1)B(t+1)	x(t)=0 y(t)	x(t)=1 y(t)
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

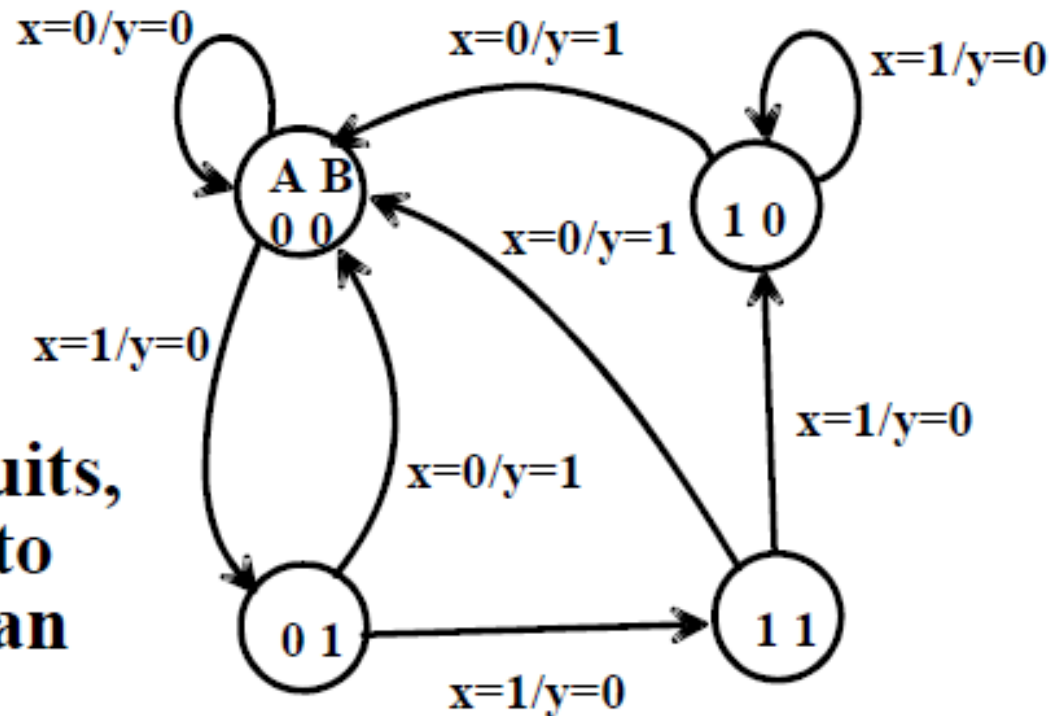
# 狀態機

陳鍾誠 於金門大學

## Example 1: State Diagram

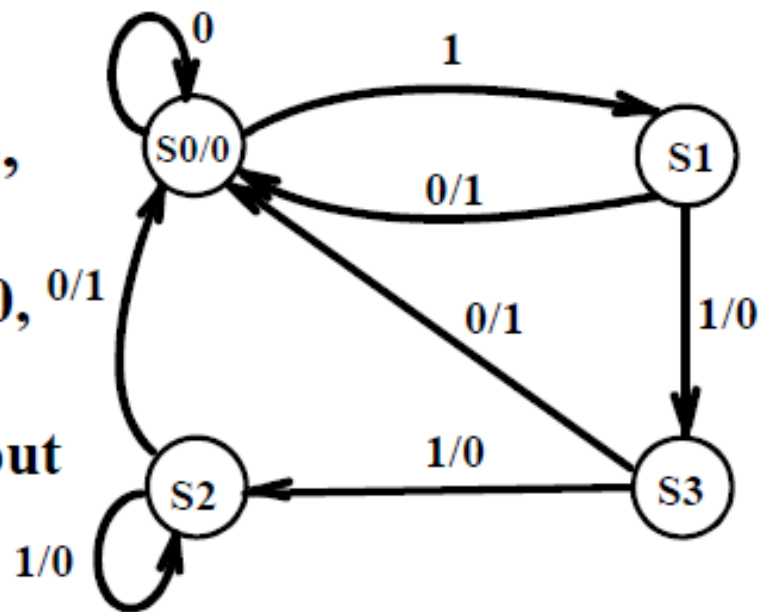
---

- Which type?
- Diagram gets confusing for large circuits
- For small circuits, usually easier to understand than the state table



## Equivalent State Example

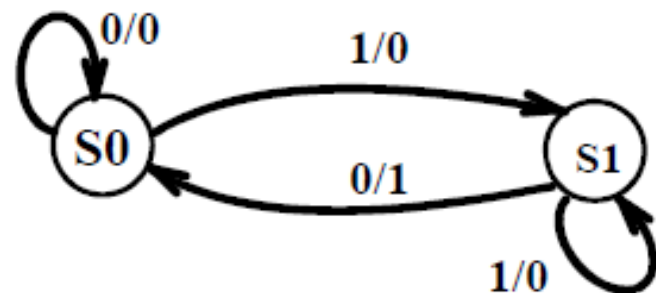
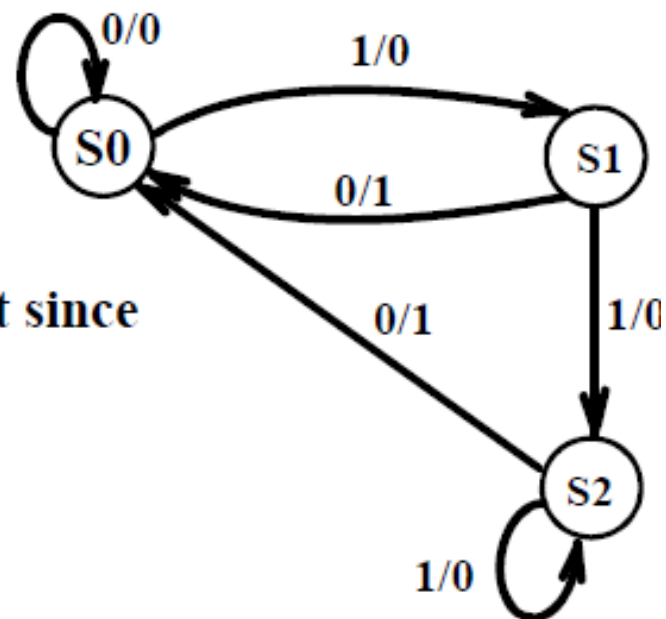
- Text Figure 5-17(a):
- For states S3 and S2,
  - the output for input 0 is 1 and input 1 is 0,  $0/1$  and
  - the next state for input 0 is S0 and for input 1 is S2.
  - By the alternative definition, states S3 and S2 are equivalent.





# Equivalent State Example

- Replacing S3 and S2 by a single state gives state diagram:
- Examining the new diagram, states S1 and S2 are equivalent since
  - their outputs for input 0 is 1 and input 1 is 0, and
  - their next state for input 0 is S0 and for input 1 is S2,
- Replacing S1 and S2 by a single state gives state diagram:



# Moore and Mealy Models

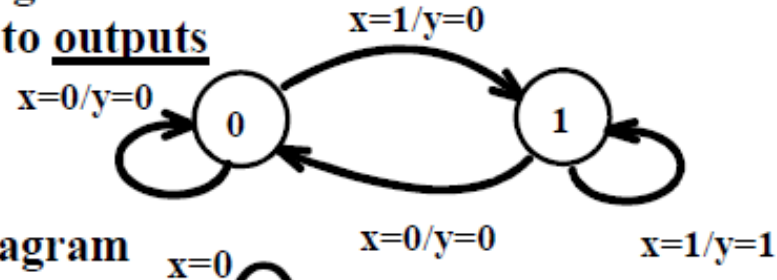
---

- Sequential Circuits or Sequential Machines are also called *Finite State Machines* (FSMs). Two formal models exist:
  - Moore Model
    - Named after E.F. Moore
    - Outputs are a function **ONLY** of states
    - Usually specified on the states.
  - Mealy Model
    - Named after G. Mealy
    - Outputs are a function of inputs **AND** states
    - Usually specified on the state transition arcs.

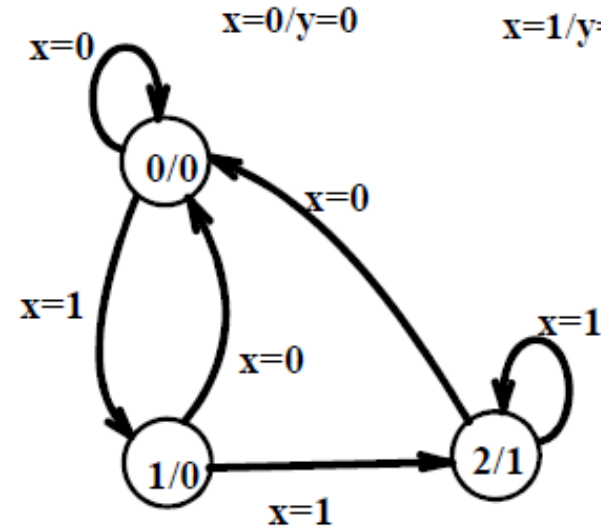
## Moore and Mealy Example Diagrams

---

- Mealy Model State Diagram  
maps inputs and state to outputs



- Moore Model State Diagram  
maps states to outputs



## Moore and Mealy Example Tables

---

- **Moore Model state table maps state to outputs**

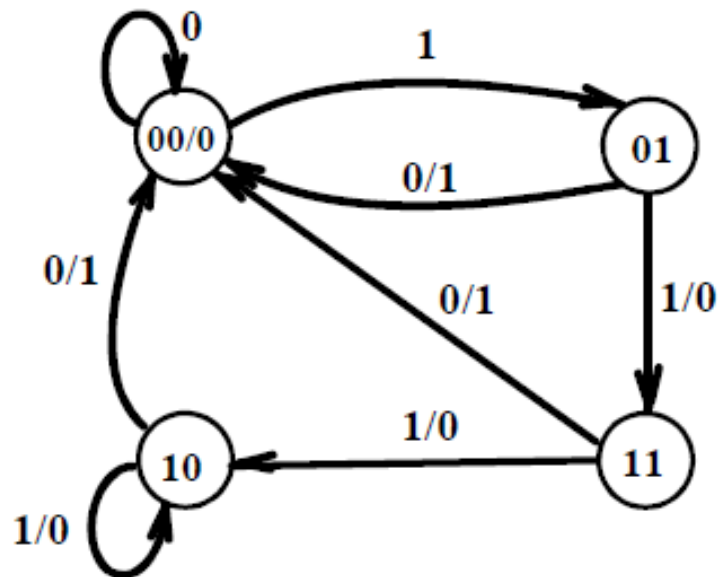
Present State	Next State		Output
	x=0	x=1	
0	0	1	0
1	0	2	0
2	0	2	1

- **Mealy Model state table maps inputs and state to outputs**

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

## Mixed Moore and Mealy Outputs

- In real designs, some outputs may be Moore type and other outputs may be Mealy type.
- Example: Figure 5-17(a) can be modified to illustrate this
  - State 00: Moore
  - States 01, 10, and 11: Mealy
- Simplifies output specification

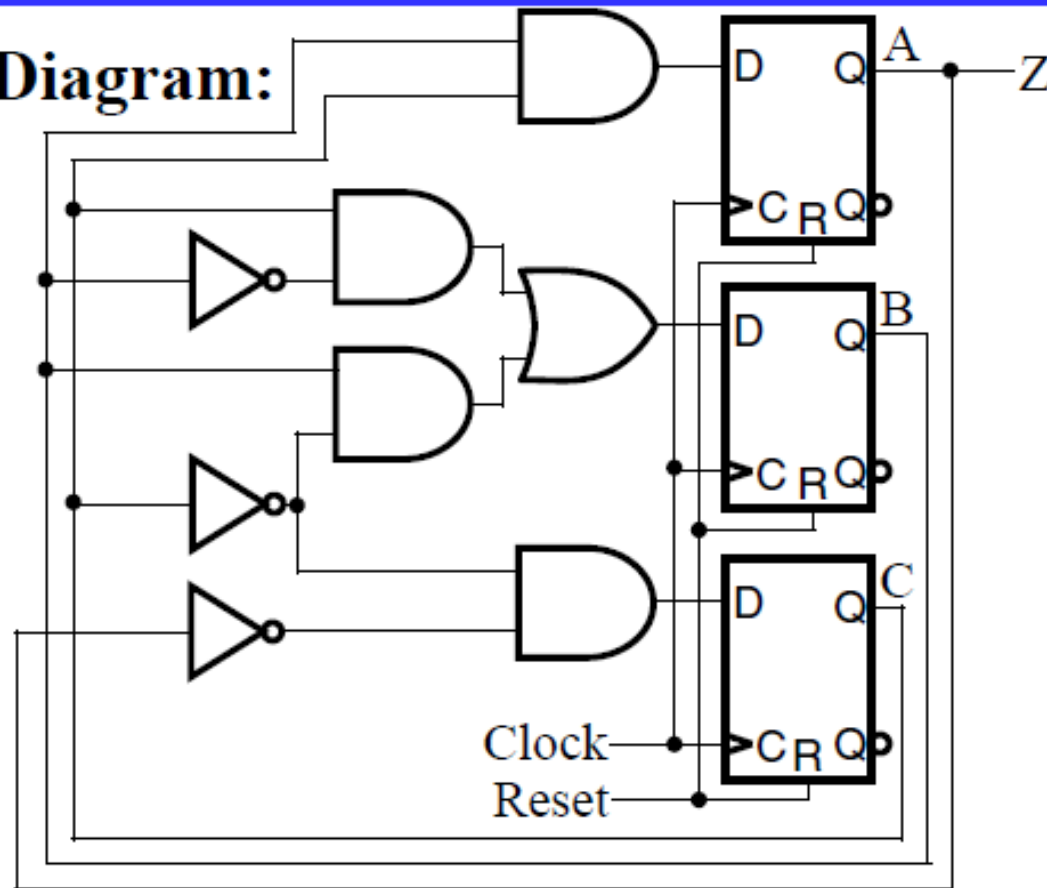


# 循序電路分析

陳鍾誠 於金門大學

## Example 2: Sequential Circuit Analysis

- Logic Diagram:



## Example 2: Flip-Flop Input Equations

---

- **Variables**

- **Inputs: None**
- **Outputs:  $Z$**
- **State Variables:  $A, B, C$**

- **Initialization: Reset to  $(0,0,0)$**

- **Equations**

- $A(t+1) =$   $Z =$
- $B(t+1) =$
- $C(t+1) =$



## Example 2: State Table

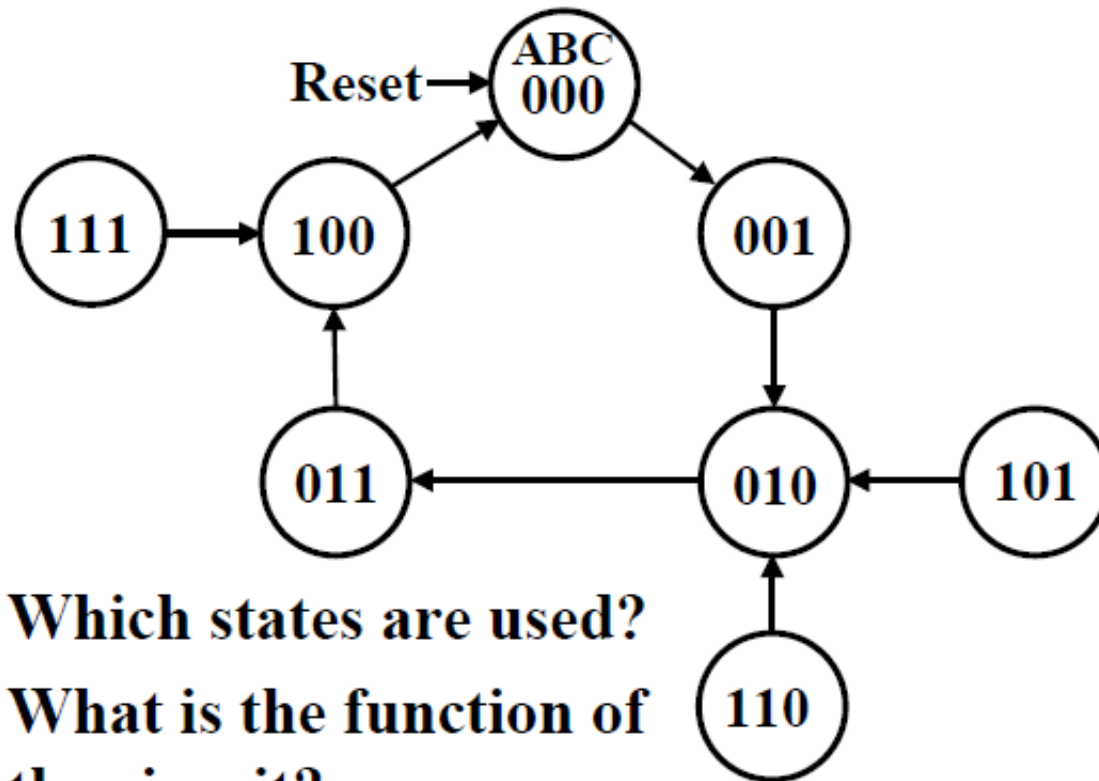
---

$X' = X(t+1)$

A B C	A'B'C'	Z
0 0 0		
0 0 1		
0 1 0		
0 1 1		
1 0 0		
1 0 1		
1 1 0		
1 1 1		

## Example 2: State Diagram

---



- Which states are used?
- What is the function of the circuit?

# 狀態機電路設計

- 參考 LCDF4\_Chap\_05\_P3.pdf

# 授權限制

- 本投影片部份取材自邏輯與計算機設計 之投影片
  - 來源：<http://writphotec.com/mano4/>
  - 使用時請遵照下列條款

## Terms of Use

---

- All (or portions) of this material © 2008 by Pearson Education, Inc.
- Permission is given to incorporate this material or adaptations thereof into classroom presentations and handouts to instructors in courses adopting the latest edition of Logic and Computer Design Fundamentals as the course textbook.
- These materials or adaptations thereof are not to be sold or otherwise offered for consideration.
- This Terms of Use slide or page is to be included within the original materials or any adaptations thereof.