

# 數位邏輯實習

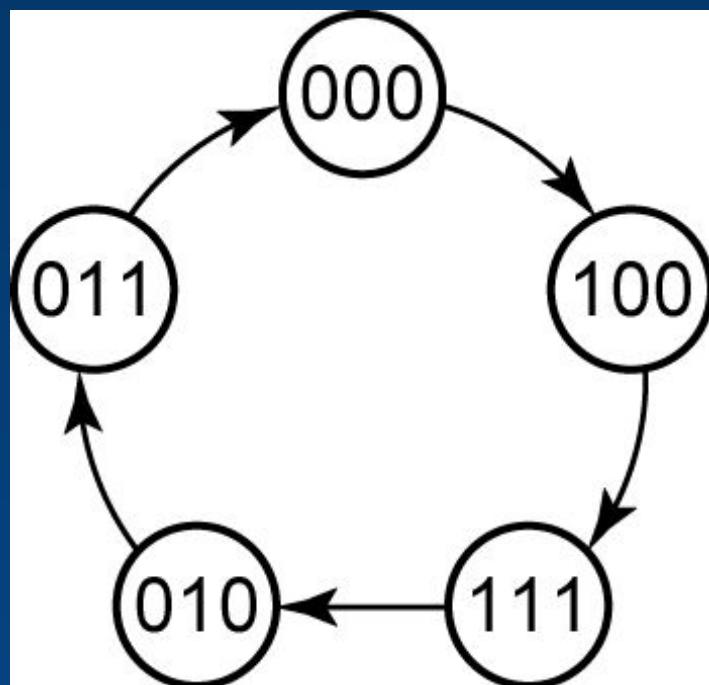
計數器、狀態機

國立台北科技大學  
電機工程系  
吳昭正

# Outline

- 計數器
- 狀態機
- VHDL語法介紹
- 作業題

1. Create a state graph to count in the desired sequence.
2. Create a state table from the state graph created in (1).  
We need one flip-flop per bit. Ex: if we need to count from 0 to 7, we need 3 bits, therefore we should use three flip-flops.
3. Derive Karnaugh maps from the state table created in (2) and solve for the inputs to each flip-flop.



**Table 12-3: State Table for Figure 12-21**

C	B	A	C <sup>+</sup>	B <sup>+</sup>	A <sup>+</sup>
0	0	0	1	0	0
0	0	1	-	-	-
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1	-	-	-
1	1	0	-	-	-
1	1	1	0	1	0

*Figure 12-21: State Graph for Counter*

We could derive  $T_C$ ,  $T_B$ , and  $T_A$  directly from the state table, but it is often more convenient to plot next-state maps showing  $C^+$ ,  $B^+$ , and  $A^+$  as functions of  $C$ ,  $B$ , and  $A$ , and then derive  $T_C$ ,  $T_B$ , and  $T_A$  from these maps.

# 計數器設計(4/7)

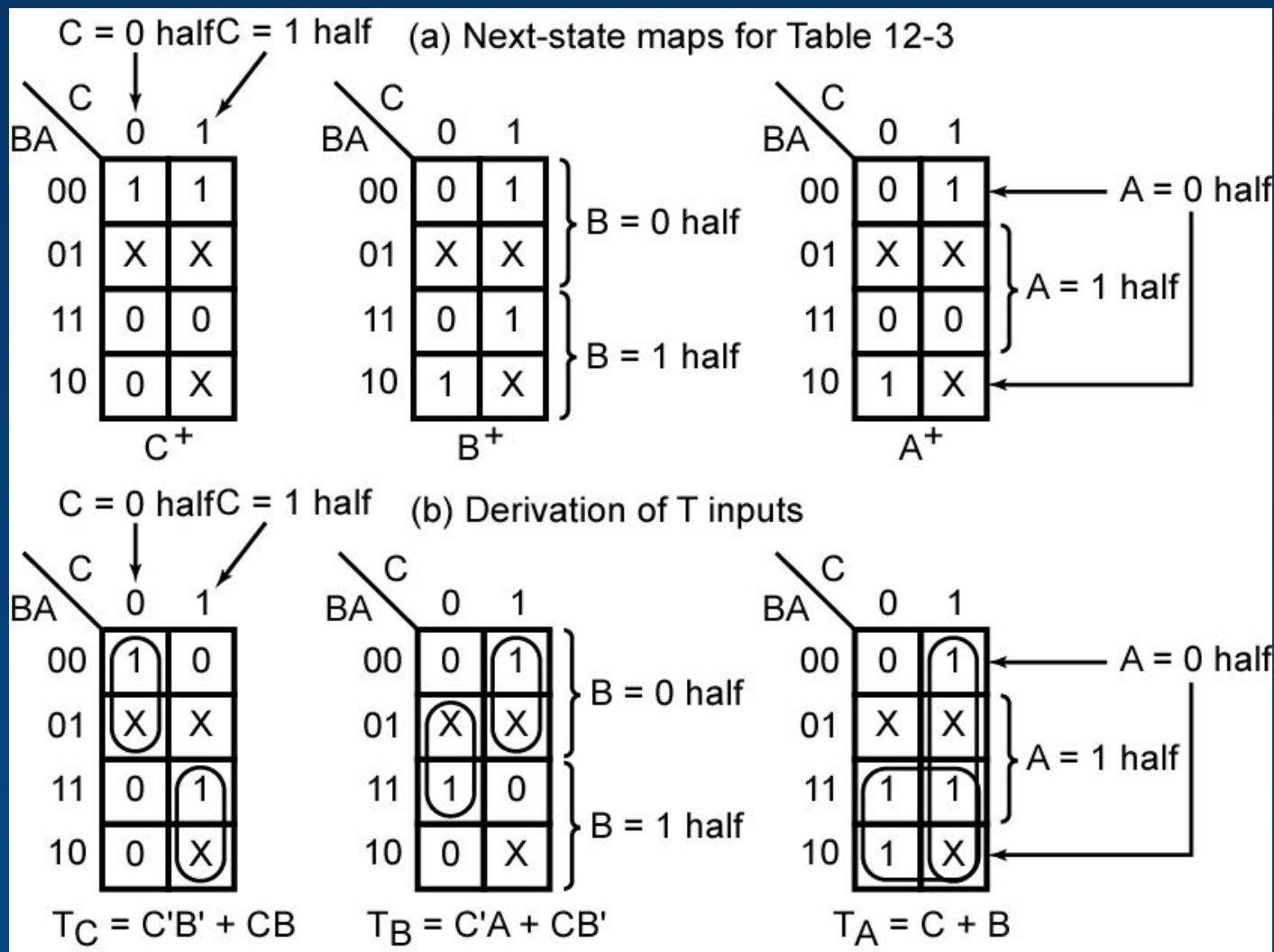


Figure 12-22

# 計數器設計(5/7)

Table 12-4. Input for T Flip-Flop

$Q$	$Q^+$	$T$
0	0	0
0	1	1
1	0	1
1	1	0

$$T = Q^+ \oplus Q$$

Given the present state of a T flip-flop ( $Q$ ) and the desired next state ( $Q^+$ ), the  $T$  input must be a 1 whenever a change in state is required. Thus,  $T = 1$  whenever  $Q^+ \neq Q$ .

# 計數器設計(6/7)

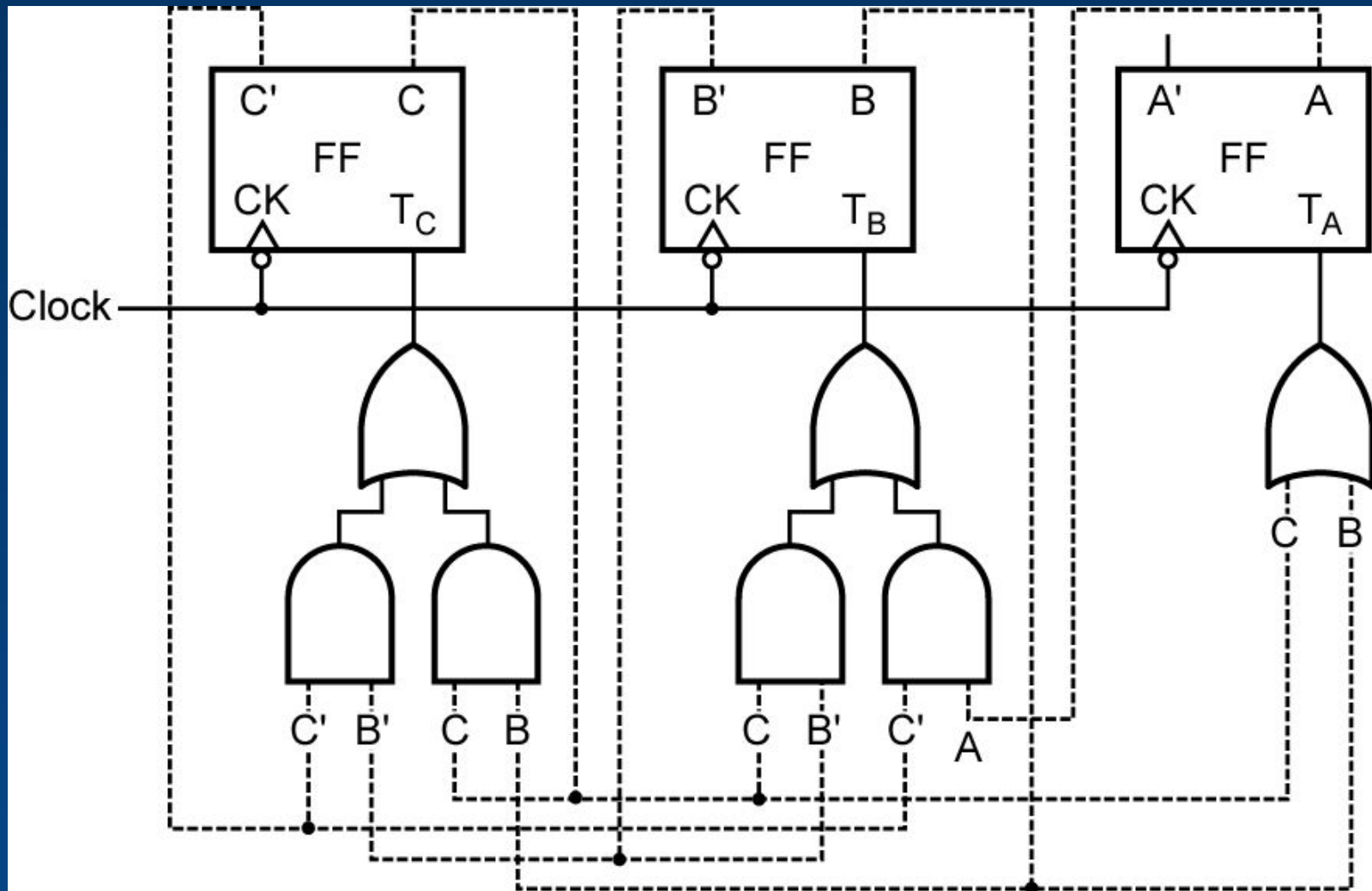


Figure 12-23: Counter Using T Flip-Flops



Table 12-9. Determination of Flip-Flop Input Equations from Next-State Equations Using Karnaugh Maps

Type of Flip-Flop	Input	Q = 0		Q = 1		Rules for Forming Input Map From Next-State Map*	
		Q <sup>+</sup> = 0	Q <sup>+</sup> = 1	Q <sup>+</sup> = 0	Q <sup>+</sup> = 1	Q = 0 Half of Map	Q = 1 Half of Map
Delay	<i>D</i>	0	1	0	1	no change	no change
Toggle	<i>T</i>	0	1	1	0	no change	complement
Set-Reset	<i>S</i>	0	1	0	X	no change	replace 1's with X's**
	<i>R</i>	X	0	1	0	replace 0's with X's**	complement
J-K	<i>J</i>	0	1	X	X	no change	fill in with X's
	<i>K</i>	X	X	1	0	fill in with X's	complement

Q<sup>+</sup> means the next state of Q

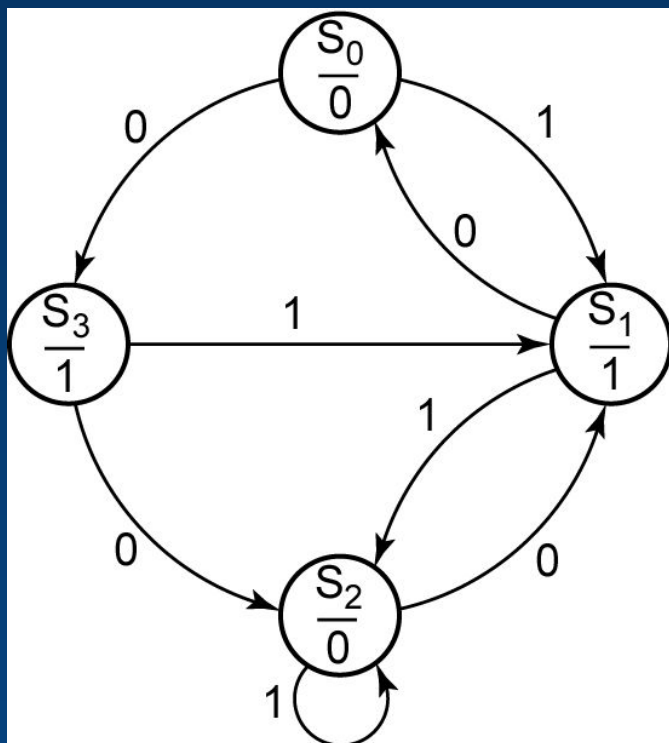
X is a don't-care

\*Always copy X's from the next-state map onto the input maps first.

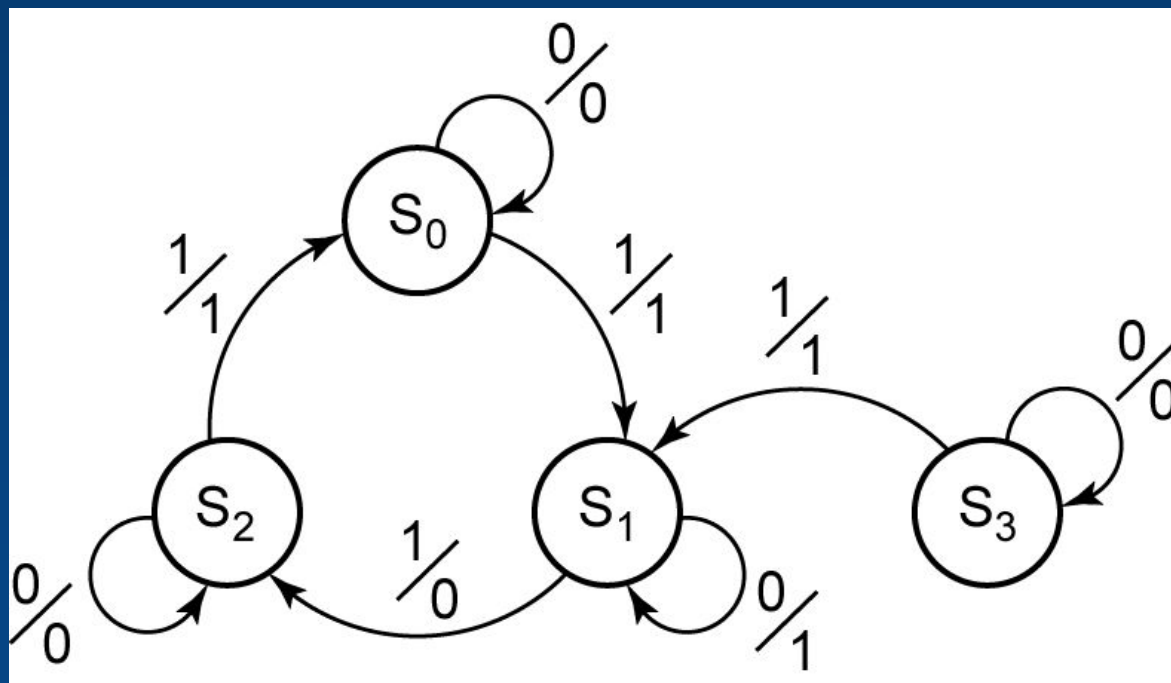
\*\*Fill in the remaining squares with 0's.

# 狀態機基本概念(1/13)

## Moore or Mealy State Machine

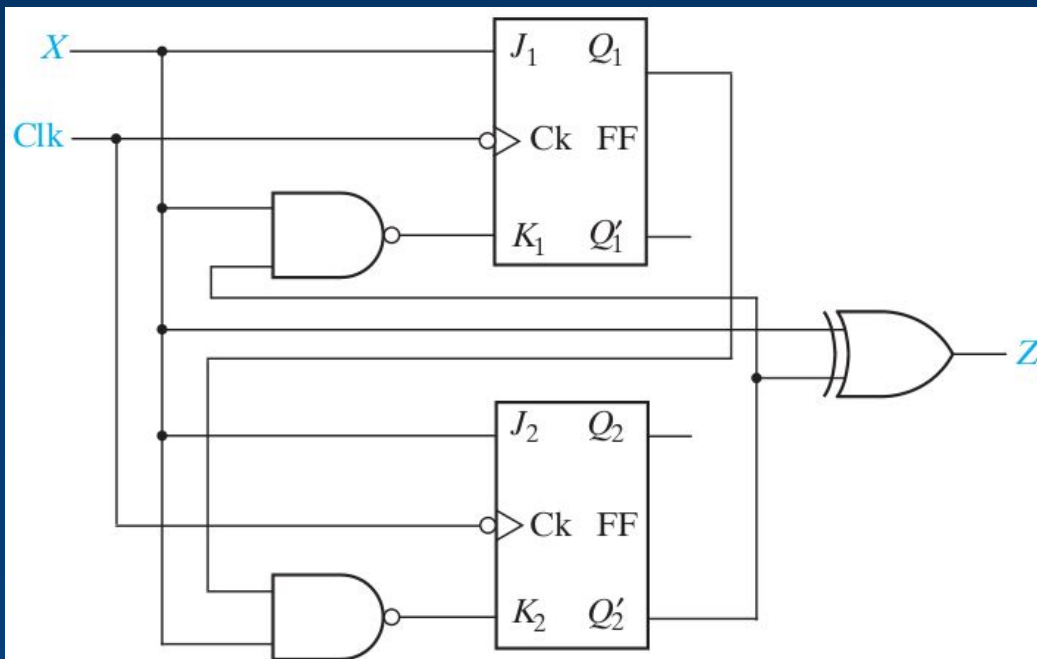


Moore or Mealy State Machine?

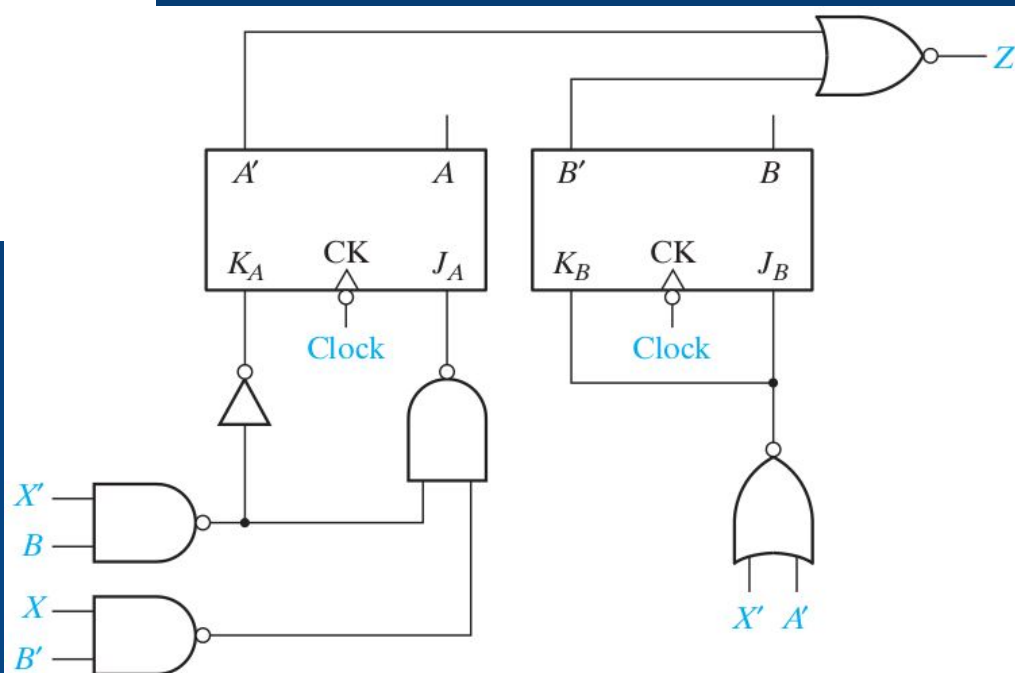


# 狀態機基本概念(2/13)

## Moore or Mealy State Machine



Moore or Mealy State Machine?



# 狀態機基本概念(3/13)

## 從電路推導狀態圖

Although constructing timing charts is satisfactory for small circuits and short input sequences, the construction of state tables and graphs provides a more systematic approach which is useful for the analysis of larger circuits and which leads to a general synthesis procedure for sequential circuits.

The state table specifies the next state and output of a sequential circuit in terms of its present state and input.

**Section 13.3 (p. 401)**

# 狀態機基本概念(4/13)

## 從電路推導狀態圖

The following method can be used to construct the state table:

1. Determine the flip-flop input equations and the output equations from the circuit.
2. Derive the next-state equation for each flip-flop from its input equations, using one of the following relations:

$$\text{D flip-flop} \quad Q^+ = D \quad (13-1)$$

$$\text{D-CE flip-flop} \quad Q^+ = D \cdot CE + Q \cdot CE' \quad (13-2)$$

$$\text{T flip-flop} \quad Q^+ = T \oplus Q \quad (13-3)$$

$$\text{S-R flip-flop} \quad Q^+ = S + R'Q \quad (13-4)$$

$$\text{J-K flip-flop} \quad Q^+ = JQ' + K'Q \quad (13-5)$$

# 狀態機基本概念(5/13)

## 從電路推導狀態圖

3. Plot a next-state map for each flip-flop.
4. Combine these maps to form the state table. Such a state table, which gives the next state of the flip-flops as a function of their present state and the circuit inputs, is frequently referred to as a transition table.

# 狀態機基本概念(6/13)

## 從電路推導狀態圖

As an example of this procedure, we will derive the state table for the circuit of Figure 13-5:

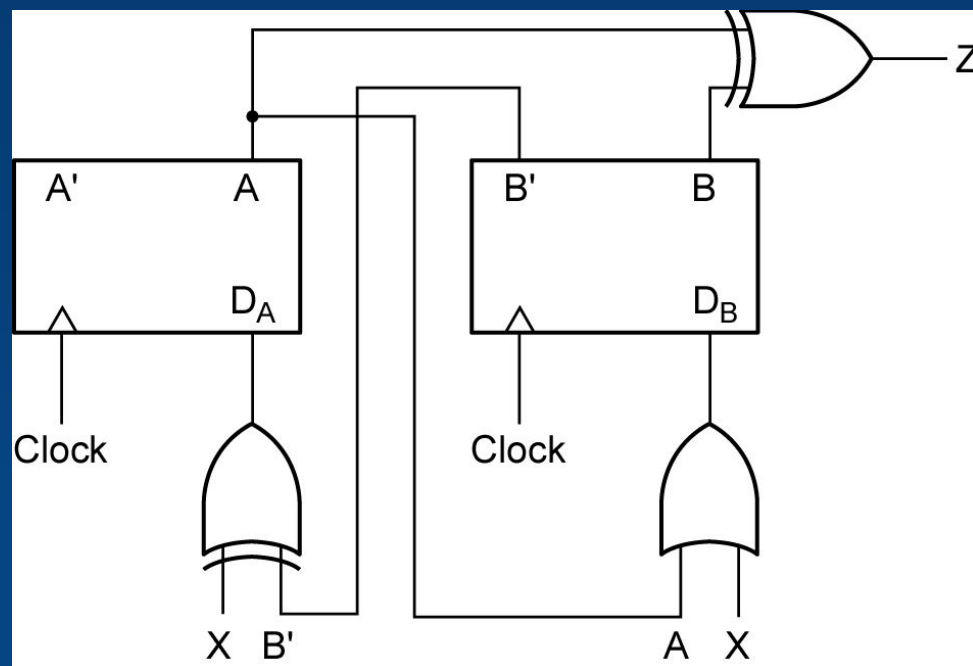
1. The flip-flop input equations and output equation are

$$D_A = X \oplus B' \quad D_B = X + A \quad Z = A \oplus B$$

2. The next-state equations for the flip-flops are

$$A^+ = X \oplus B' \quad B^+ = X + A$$

**Figure 13-5**



# 狀態機基本概念(7/13)

## 從電路推導狀態圖

3. The corresponding maps are

		$X$	
		0	1
$AB$	00	1	0
	01	0	1
	11	0	1
	10	1	0
		$A^+$	

		$X$	
		0	1
$AB$	00	0	1
	01	0	1
	11	1	1
	10	1	1
		$B^+$	



# 狀態機基本概念(8/13)

## 從電路推導狀態圖

4. Combining these maps yields the transition table in Table 13-2(a), which gives the next state of both flip-flops ( $A^+B^+$ ) as a function of the present state and input. The output function  $Z$  is then added to the table. In this example, the output depends only on the present state of the flip-flops and not on the input, so only a single output column is required.

Table 13-2. Moore State Tables for Figure 13-5

AB	$A^+B^+$		Z	Present State	Next State		Present Output(Z)
	X=0	X=1			X = 0	X = 1	
00	10	01	0	$S_0$	$S_3$	$S_1$	0
01	00	11	1	$S_1$	$S_0$	$S_2$	1
11	01	11	0	$S_2$	$S_1$	$S_2$	0
10	11	01	1	$S_3$	$S_2$	$S_1$	1

(a)

(b)

# 狀態機基本概念(9/13)

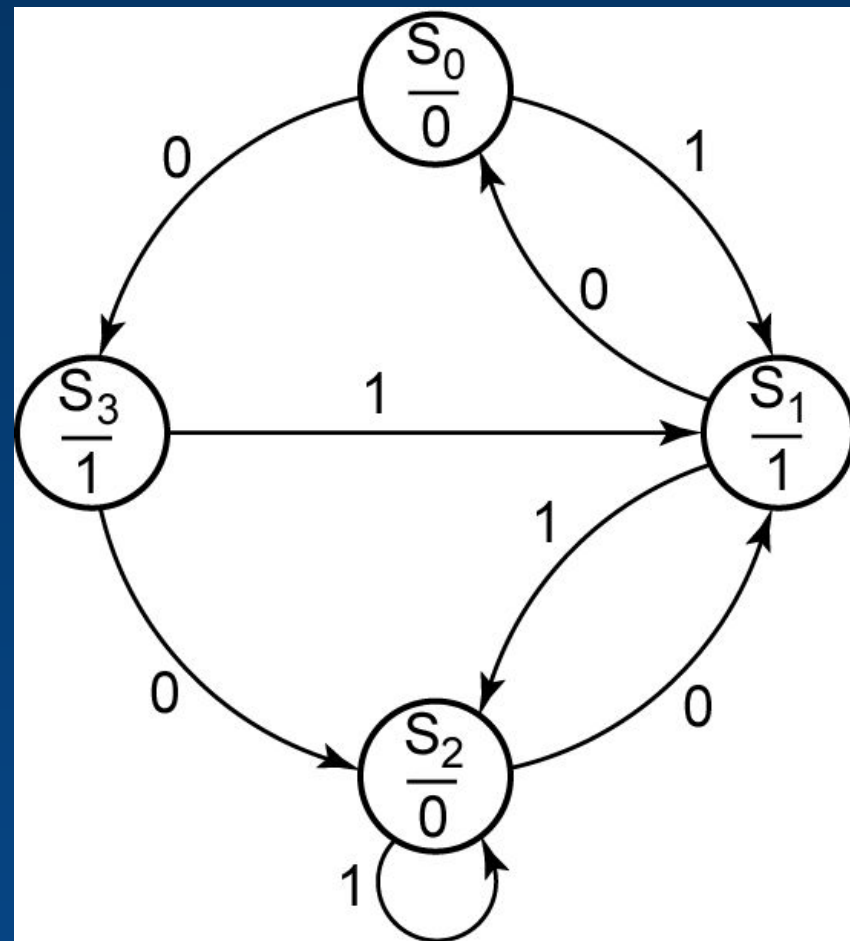
## 從電路推導狀態圖

**Table 13-2**

Present State	Next State		Present Output(Z)
	X = 0	X = 1	
S <sub>0</sub>	S <sub>3</sub>	S <sub>1</sub>	0
S <sub>1</sub>	S <sub>0</sub>	S <sub>2</sub>	1
S <sub>2</sub>	S <sub>1</sub>	S <sub>2</sub>	0
S <sub>3</sub>	S <sub>2</sub>	S <sub>1</sub>	1

(b)

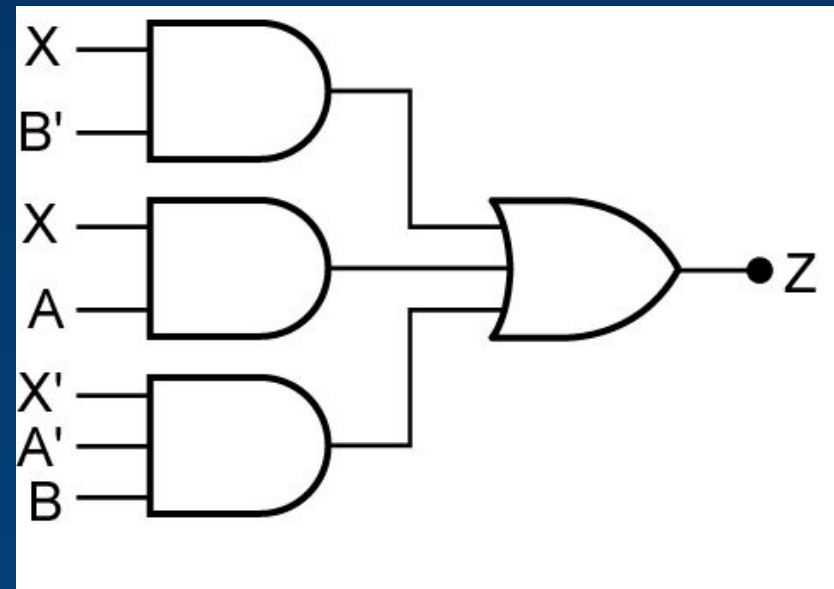
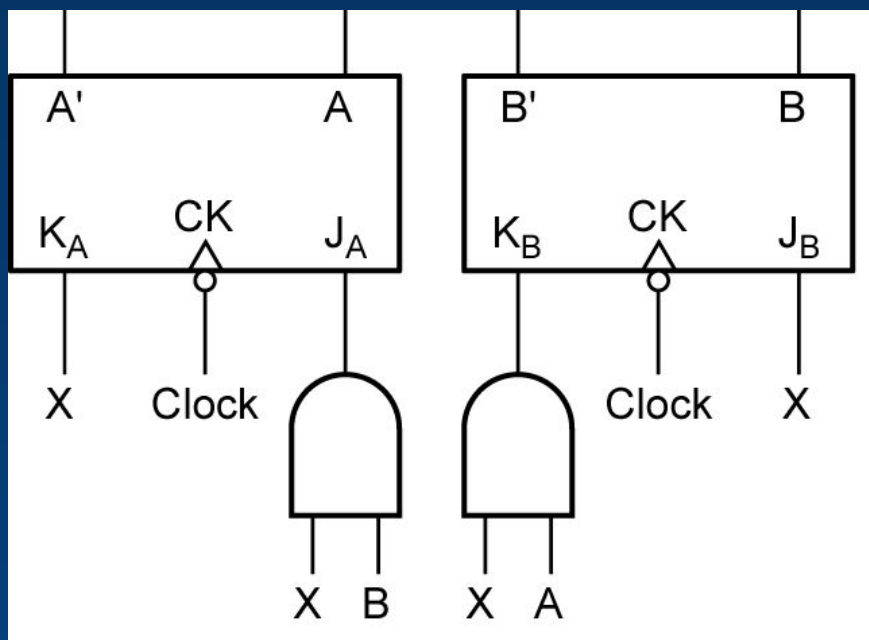
Each node in the graph represents a state in the circuit.



*Figure 13-9: Moore State Graph for Figure 13-5*

# 狀態機基本概念(10/13)

## 從電路推導狀態圖



**Figure 13-7**

We can construct the next-state and output equations from the circuit diagram:

$$A^+ = J_A A' + K'_A A = XBA' + X'A$$

$$B^+ = J_B B' + K'_B B = XB' + (AX)'B = XB' + X'B + A'B$$

$$Z = X'A'B + XB' + XA$$

# 狀態機基本概念(11/13)

## 從電路推導狀態圖

Next, we can plot Karnaugh maps for  $A^+$ ,  $B^+$ , and  $Z$ . We can then use these maps to derive the transition table.

AB \ X		0	1
		0	1
00	0	0	
01	0	1	
11	1	0	
10	1	0	
		$A^+$	

AB \ X		0	1
		0	1
00	0	1	
01	1	1	
11	1	0	
10	0	1	
		$B^+$	

AB \ X		0	1
		0	1
00	0	1	
01	1	0	
11	0	1	
10	0	1	
		$Z$	

Figure 13-10

# 狀態機基本概念(12/13)

## 從電路推導狀態圖

(a)

$AB$	$A^+B^+$		$Z$	
	$X = 0$	1	$X = 0$	1
00	00	01	0	1
01	01	11	1	0
11	11	00	0	1
10	10	01	0	1

(b)

Present State	Next State		Present Output	
	$X = 0$	1	$X = 0$	1
$S_0$	$S_0$	$S_1$	0	1
$S_1$	$S_1$	$S_2$	1	0
$S_2$	$S_2$	$S_0$	0	1
$S_3$	$S_3$	$S_1$	0	1

Table 13-3. Mealy State Tables for Figure 13-7

# 狀態機基本概念(13/13)

## 從電路推導狀態圖

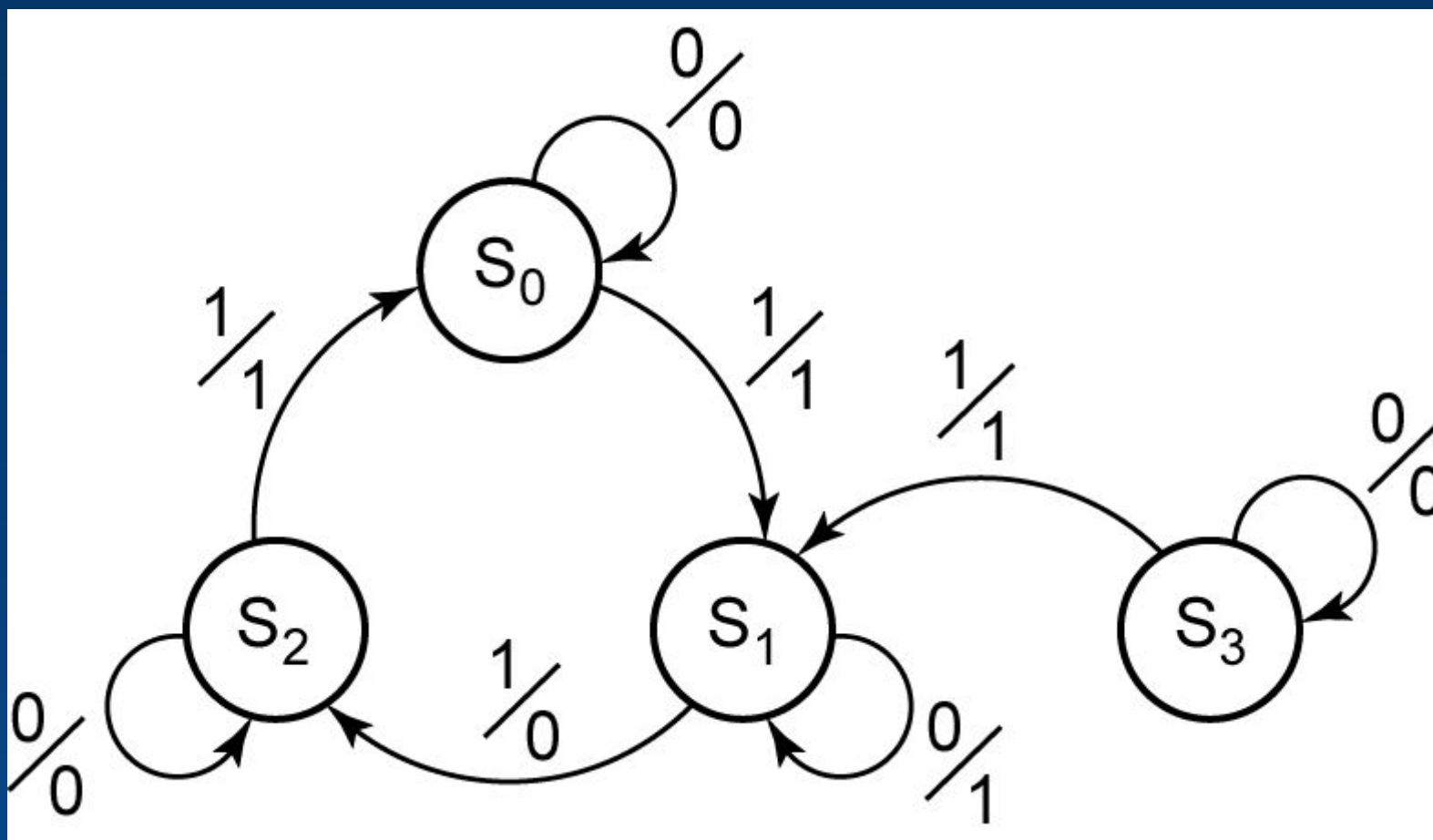
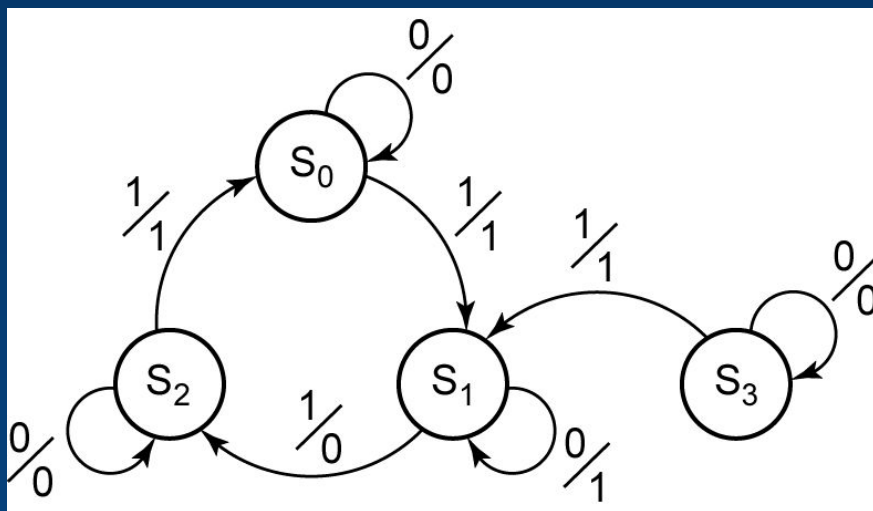


Figure 13-11: Mealy State Graph for Figure 13-7

# 狀態機設計(1/5)

## 從狀態圖推導電路

### 1. 從狀態圖建立狀態表



(b)

Present State	Next State		Present Output	
	X = 0	1	X = 0	1
S <sub>0</sub>	S <sub>0</sub>	S <sub>1</sub>	0	1
S <sub>1</sub>	S <sub>1</sub>	S <sub>2</sub>	1	0
S <sub>2</sub>	S <sub>2</sub>	S <sub>0</sub>	0	1
S <sub>3</sub>	S <sub>3</sub>	S <sub>1</sub>	0	1

# 狀態機設計(2/5)

## 從狀態圖推導電路

2. 把狀態表中的 $S_0$ 、 $S_1$ 、...、 $S_n$ 改成二進位數字

(b)

Present State	Next State		Present Output	
	$X = 0$	1	$X = 0$	1
$S_0$	$S_0$	$S_1$	0	1
$S_1$	$S_1$	$S_2$	1	0
$S_2$	$S_2$	$S_0$	0	1
$S_3$	$S_3$	$S_1$	0	1



(a)

$AB$	$A^+B^+$		$Z$	
	$X = 0$	1	$X = 0$	1
00	00	01	0	1
01	01	11	1	0
11	11	00	0	1
10	10	01	0	1




# 狀態機設計(3/5)

## 從狀態圖推導電路

### 3.從狀態表建立Next-State Map

(a)

AB	$A^+B^+$		Z	
	X = 0	1	X = 0	1
00	00	01	0	1
01	01	11	1	0
11	11	00	0	1
10	10	01	0	1



AB \ X	0		1	
	0	1	0	1
00	0	0	0	1
01	0	1	1	1
11	1	0	1	0
10	1	0	0	1

$A^+$        $B^+$

AB \ X	0		1	
	0	1	0	1
00	0	1	0	1
01	1	0	0	1
11	0	1	0	1
10	0	1	0	1

Z

# 狀態機設計(4/5)

## 從狀態圖推導電路

### 4. 從Next-State Map轉換成卡諾圖

		X	
		0	1
AB	00	0	0
	01	0	1
	11	1	0
	10	1	0
		$A^+$	

Type of Flip-Flop	Input	Q = 0		Q = 1		Rules for Forming Input Map From Next-State Map*	
		$Q^+ = 0$	$Q^+ = 1$	$Q^+ = 0$	$Q^+ = 1$		
Delay	D	0	1	0	1	no change	no change
Toggle	T	0	1	1	0	no change	complement
Set-Reset	S	0	1	0	X	no change	replace 1's with X's**
	R	X	0	1	0	replace 0's with X's**	complement
J-K	J	0	1	X	X	no change	fill in with X's
	K	X	X	1	0	fill in with X's	complement

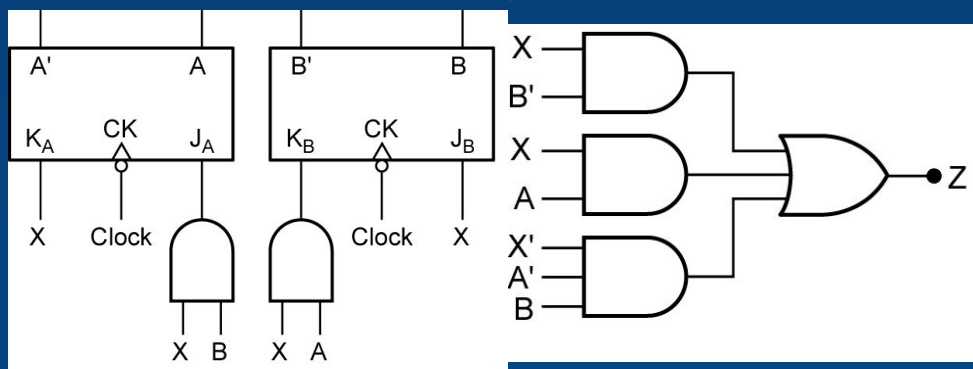
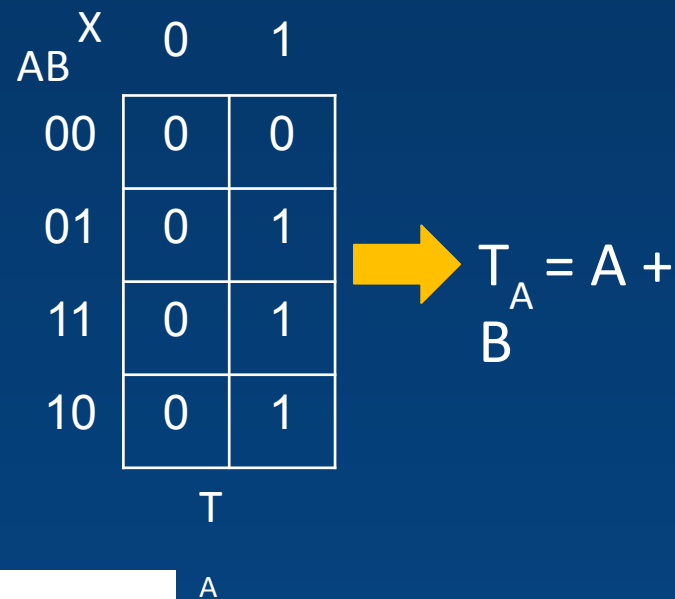
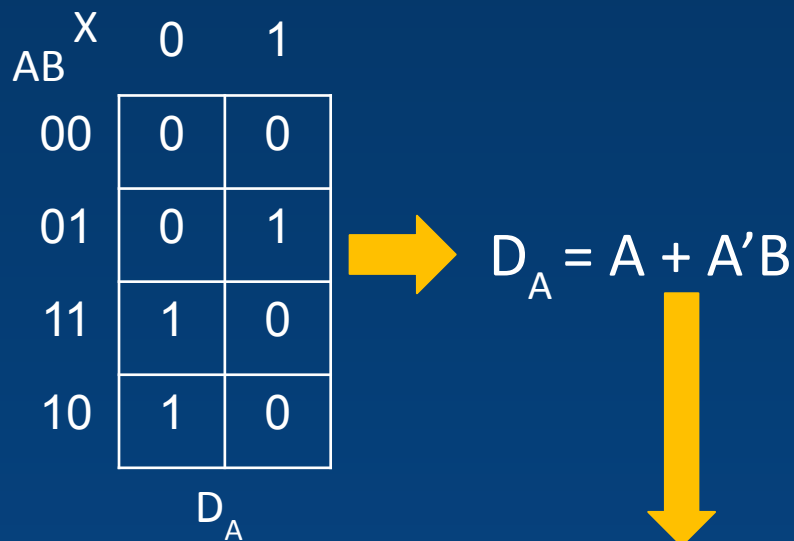
		X	
		0	1
AB	00	0	0
	01	0	1
	11	1	0
	10	1	0
		$D_A$	

		X	
		0	1
AB	00	0	0
	01	0	1
	11	0	1
	10	0	1
		T	

# 狀態機設計(5/5)

## 從狀態圖推導電路

### 5.從卡諾圖推導最簡布林代數式



範例圖  
與上方布林代數式無關

# VHDL設計程序(1/5)

## Modeling State Machine

### 1. 插入library

```
library ieee;
```

### 2. 插入USE

```
use ieee.std_logic_1164.all;
```

# VHDL設計程序(2/5)

## Modeling State Machine

### 3. 宣告entity

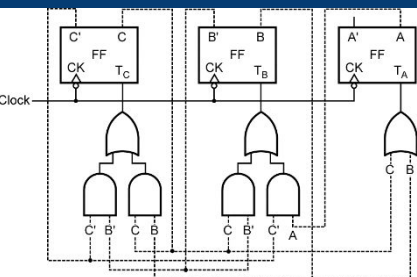
```
entity test is
  port (
    -- Input ports
    clk : in std_logic;

    -- Output ports
    a_out, b_out, c_out : out std_logic );
end test;
```

# VHDL設計程序(3/5)

## Modeling State Machine

### 4.1. 宣告architecture決定設計方式 (component)



原圖請見投影片第8頁

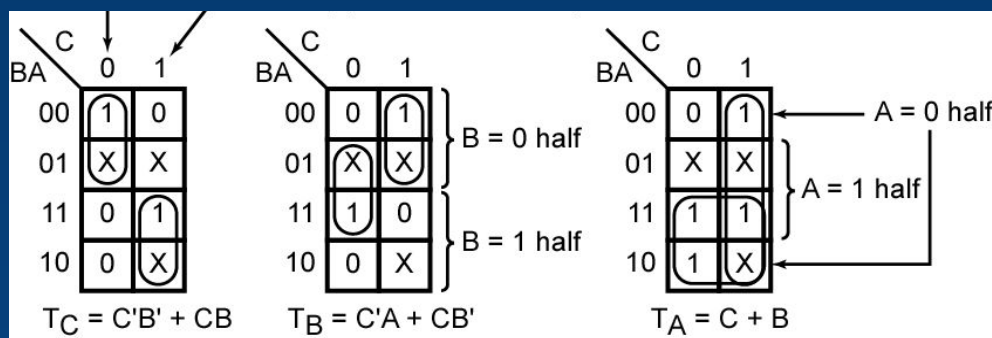
```
architecture test_arch of test is
    component my_TFF
        port (
            Ck, T: in std_logic;
            Q    : out std_logic );
    end component;
    signal a, b, c : std_logic := '0';
begin
    TFFA : my_TFF port map (Clk, (c or b), a);
    TFFB : my_TFF port map (Clk, (a and not(c)) or (not(b) and c), b);
    TFFC : my_TFF port map (Clk, (b and c) or (not(b) and not(c)), c);

    a_out <= a;
    b_out <= b;
    c_out <= c;
end test_arch;
```

# VHDL設計程序(4/5)

## Modeling State Machine

### 4.2. 宣告architecture決定設計方式 (布林代數式)



原圖請見投影片第6頁

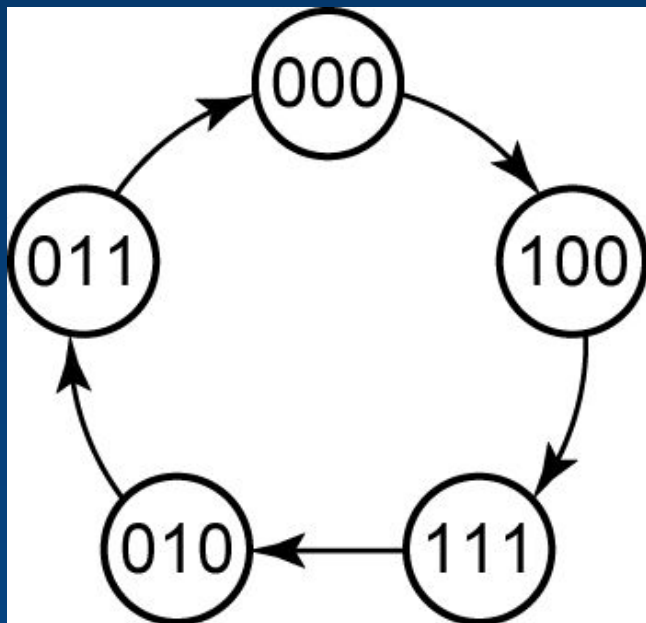
```
architecture test_boolean_arch of test_boolean is
    signal Ta, Tb, Tc : std_logic;
    signal a, b, c : std_logic := '0';
begin
    Ta <= c or b;
    Tb <= (a and not(c)) or (c and not(b));
    Tc <= (b and c) or (not(b) and not(c));
    a_out <= a;
    b_out <= b;
    c_out <= c;

    process(clk) is
    begin
        if(rising_edge(clk)) then
            a <= Ta xor a;
            b <= Tb xor b;
            c <= Tc xor c;
        end if;
    end process;
end test_boolean_arch;
```

# VHDL設計程序(5/5)

## Modeling State Machine

### 4.3. 宣告architecture決定設計方式 (狀態圖)



原圖請見投影片第4頁

```

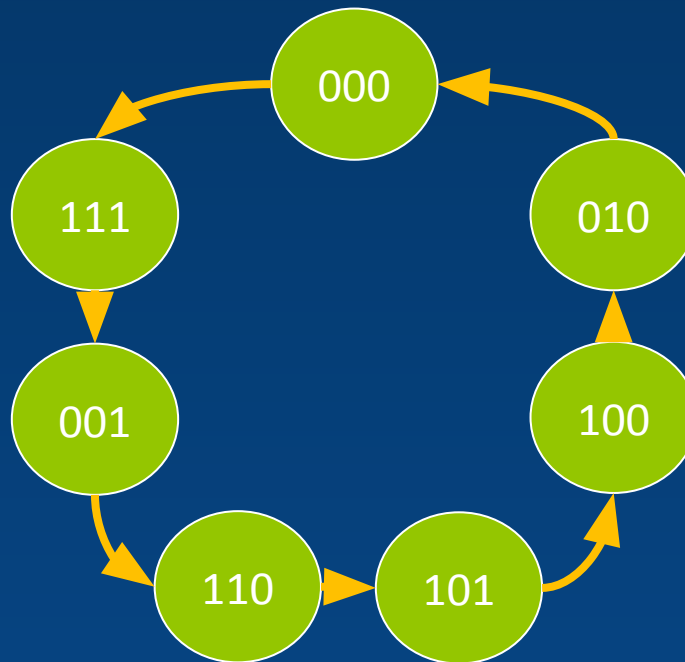
architecture state_graph of test_state_graph is
    signal z : std_logic_vector(2 downto 0) := "000";
Begin
    a_out <= z(0);
    b_out <= z(1);
    c_out <= z(2);
    process(clk) is
    begin
        if(rising_edge(clk)) then
            if(z = "000") then z <= "100";
            elsif(z = "100") then z <= "111";
            elsif(z = "111") then z <= "010";
            elsif(z = "010") then z <= "011";
            else z <= "000";
            end if;
        end if;
    end process;
end state_graph;
  
```



# 作業題(1/4)

- 作業題1:利用 $J$ - $K$ 正反器推導出下頁狀態圖的計數器電路。
- 挑選一組設計模式完成本題的計數器。
  - 第一組: 投影片32頁的設計方式 + 拉電路
  - 第二組: 投影片32頁的設計方式 + 挑選投影片30-31頁的一種設計方式

# 作業題(2/4)



# 作業題(3/4)

- 作業題2:利用  $T$  正反器推導出下頁狀態圖的狀態機電路。
- 挑選一組設計模式完成本題的狀態機。
  - 第一組: 投影片32頁的設計方式 + 拉電路
  - 第二組: 投影片32頁的設計方式 + 挑選投影片30-31頁其中一種設計方式

# 作業題(4/4)

