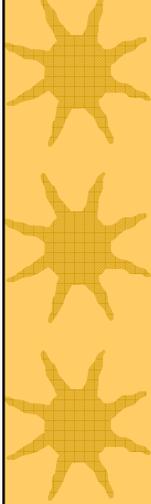




第五章 同步序向邏輯



★5-1 序向電路

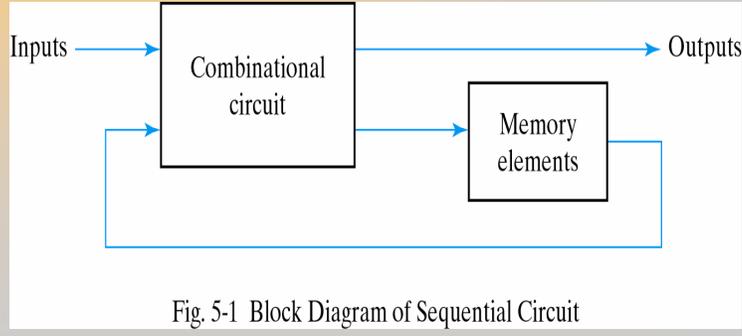
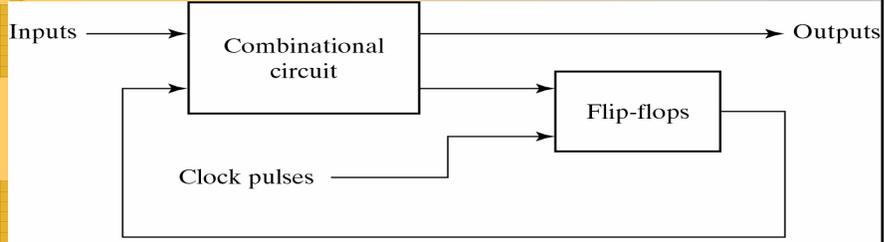
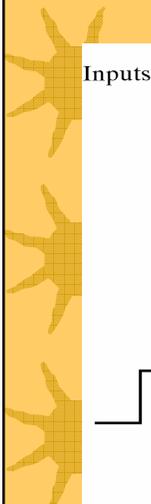


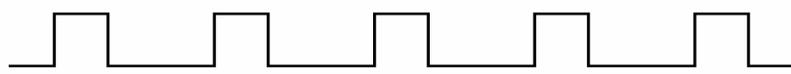
Fig. 5-1 Block Diagram of Sequential Circuit



同步時脈序向電路



(a) Block diagram



(b) Timing diagram of clock pulses

Fig. 5-2 Synchronous Clocked Sequential Circuit



5-2 門鎖器

★SR門鎖器 (SR Latch) :由NOR閘所構成之SR門鎖器

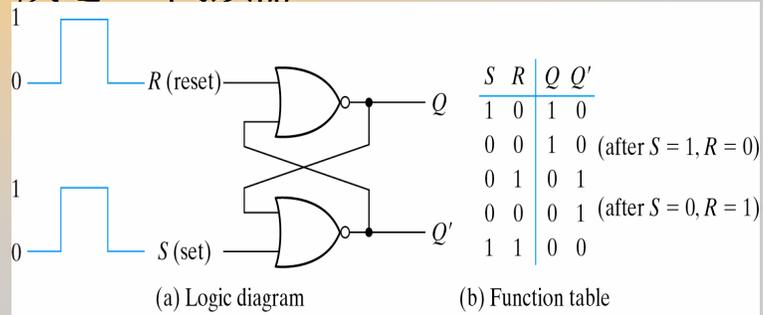


Fig. 5-3 SR Latch with NOR Gates



由NAND閘所構成之SR門鎖器

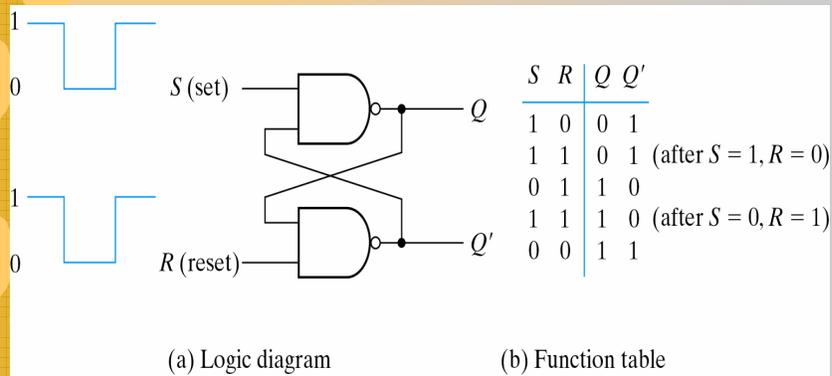
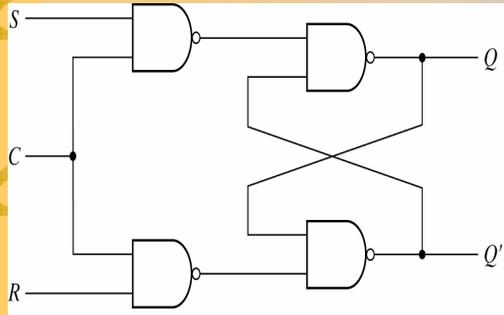


Fig. 5-4 SR Latch with NAND Gates



具有控制輸入之SR門鎖器



(a) Logic diagram

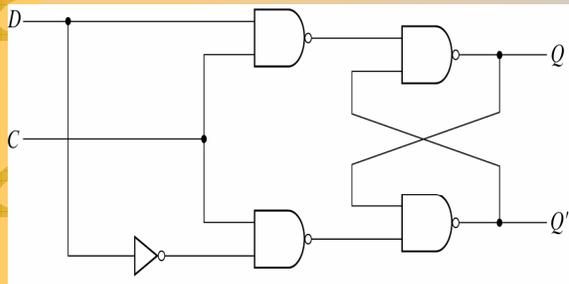
C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	Q = 0; Reset state
1	1	0	Q = 1; set state
1	1	1	Indeterminate

(b) Function table

Fig. 5-5 SR Latch with Control Input



D型門鎖器(D Latch)

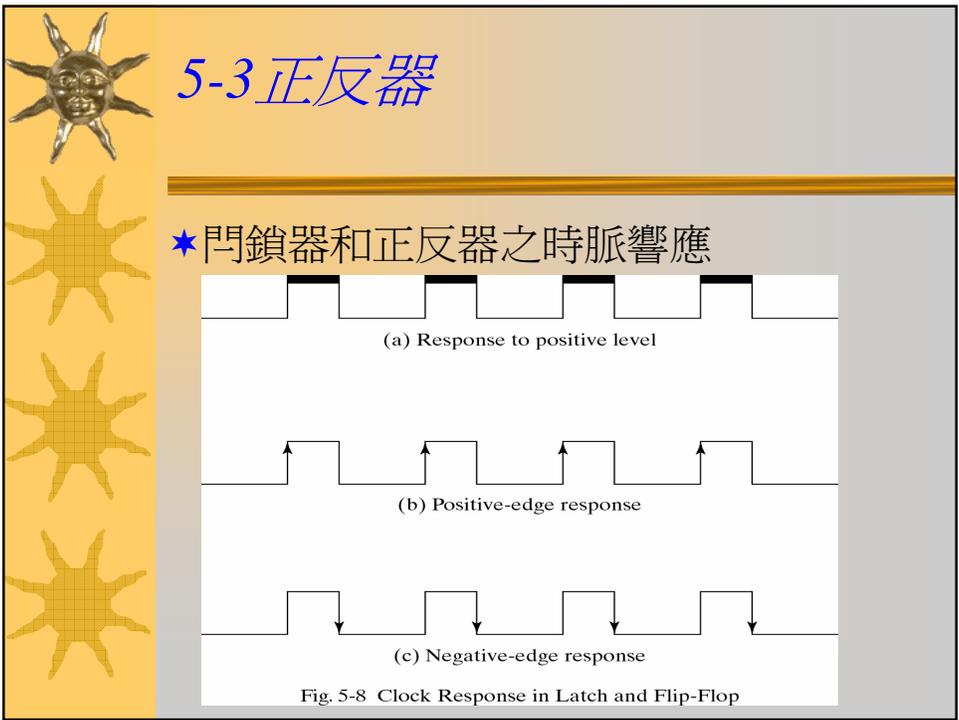
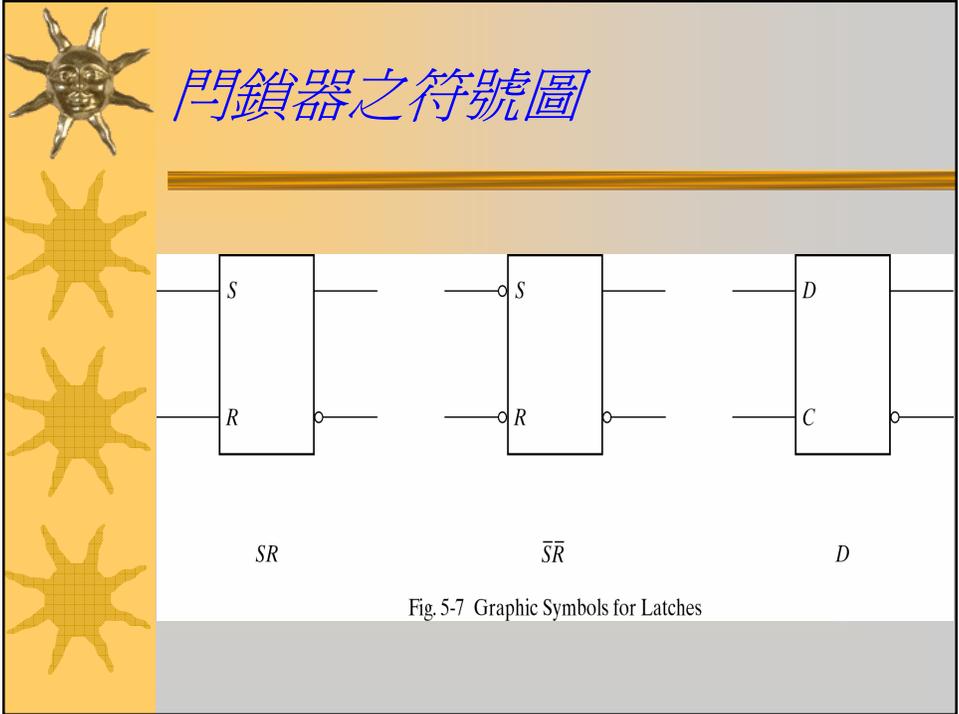


(a) Logic diagram

C	D	Next state of Q
0	X	No change
1	0	Q = 0; Reset state
1	1	Q = 1; Set state

(b) Function table

Fig. 5-6 D Latch





邊緣觸發D型正反器 (Edge-Triggered D Flip-Flop)

★D型主僕正反器

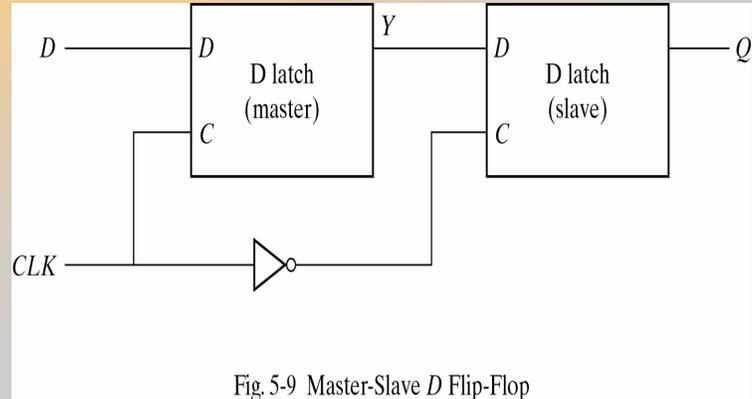


Fig. 5-9 Master-Slave D Flip-Flop



D型正緣觸發正反器

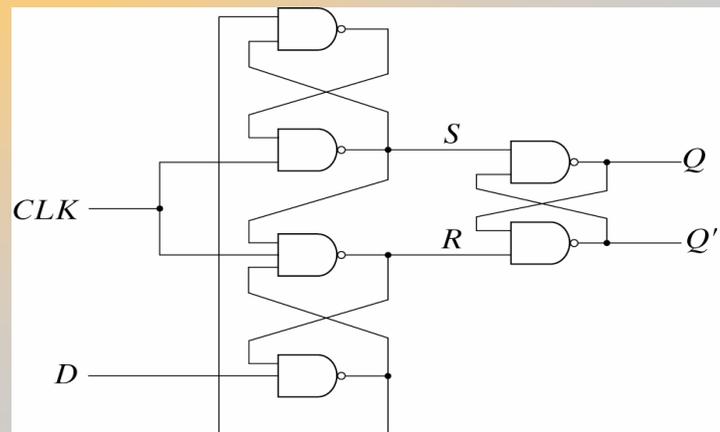
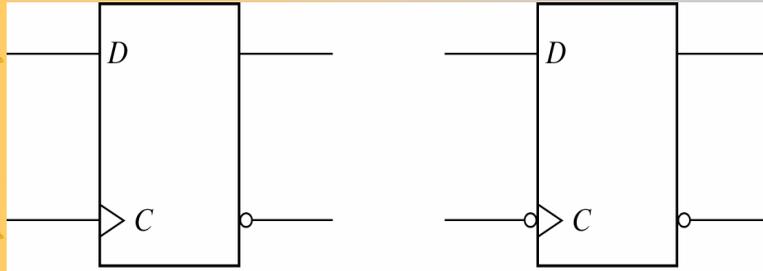


Fig. 5-10 D-Type Positive-Edge-Triggered Flip-Flop



D型邊緣觸發正反器之符號圖



(a) Positive-edge

(a) Negative-edge

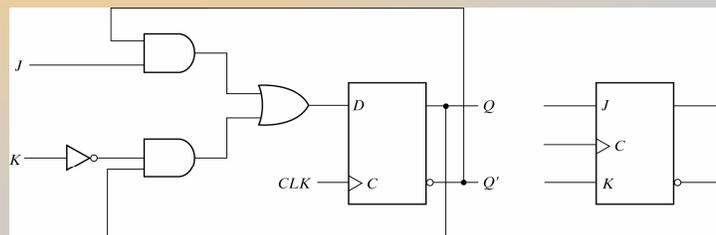
Fig. 5-11 Graphic Symbol for Edge-Triggered D Flip-Flop



JK正反器

★圖5-12(a)之D輸入端之電路方程式為

$$D = JQ' + K'Q$$



(a) Circuit diagram

(b) Graphic symbol

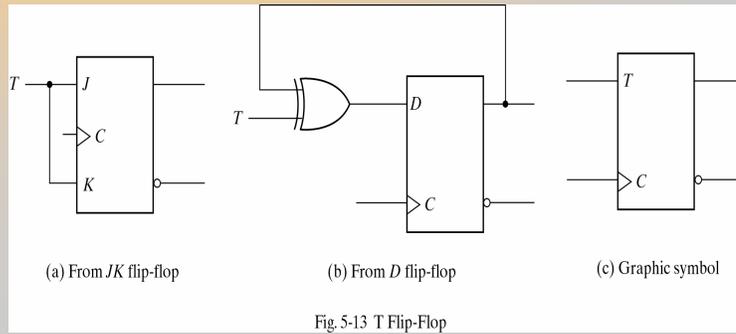
Fig. 5-12 JK Flip-Flop



T型正反器：屬互補式之正反器

★圖5-13(b) T型正反器之D輸入端表示式為

$$D = T \oplus Q = TQ' + T'Q$$



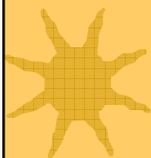
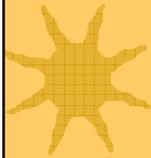
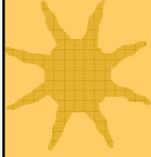
特性表

★表5-1 正反器的特性表

JK正反器			
J	k	Q(t+1)	
0	0	Q(t)	狀態未改變
0	1	0	重置為0
1	0	1	設置為1
1	1	Q'(t)	補數輸出



正反器之特性表

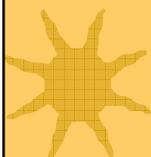


D型正反器	
D	Q(t+1)
0	0 重置為0
1	1 設置為1

T型正反器	
T	Q(t+1)
0	Q(t) 狀態未改變
1	Q'(t) 補數輸出



特性方程式



- ★ D型正反器之特性方程式為

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

$$Q(t+1) = JQ' + K'Q$$

- ★ JK正反器之特性方程式為

$$Q(t+1) = JQ' + K'Q$$

- ★ T型正反器之特性方程式為

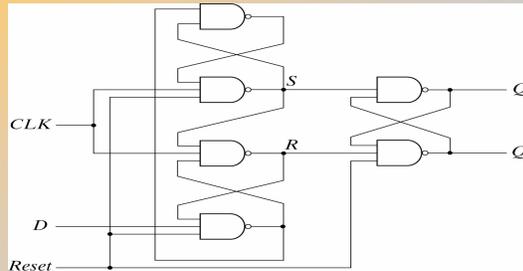
$$Q(t+1) = T \oplus Q = TQ' + T'Q$$



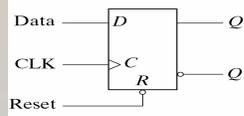
直接輸入: 用來強制正反器變成特殊狀態且與時脈無關!!
包括

(a) 預先設置(PRESET) 或 直接輸入(direct set)

(b) 清除(clear) 或 直接重置(direct reset)



(a) Circuit diagram



(b) Graphic symbol

R	C	D	Q	Q'
0	X	X	0	1
1	↑	0	0	1
1	↑	1	1	0

(b) Function table

Fig. 5-14 D Flip-Flop with Asynchronous Reset



圖5-15是由兩個D型正反器及邏輯閘所構成之時控序向電路

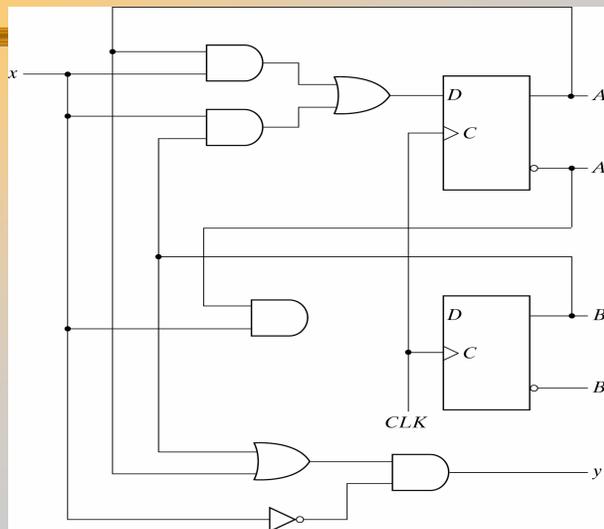
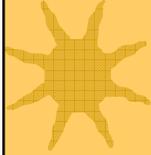


Fig. 5-15 Example of Sequential Circuit



5-4時控序向電路分析



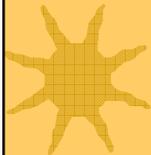
★狀態方程式:

★圖5-15電路之狀態方程式為

$$A(t+1) = A(t)x(t) + B(t)x(t)$$

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

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

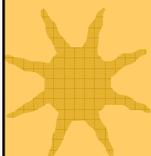


★或表示為

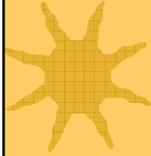
$$A(t+1) = Ax + Bx$$

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

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

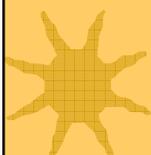
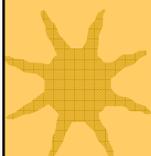


狀態表



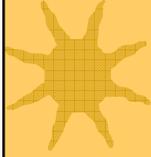
★圖5-15的狀態表

目前狀態		輸入	次一狀態		輸出
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





另一種形式的狀態表

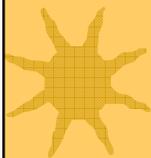


★表5-3 狀態表的第二種形式

目前狀態	次一狀態		輸出	
	x=0	x=1	x=0	x=1
AB	AB	AB	y	y
00	00	01	0	0
01	00	11	1	0
10	00	10	1	0
11	00	10	1	0



狀態圖



★表5-3的結果可用狀態圖表示 如圖5-16

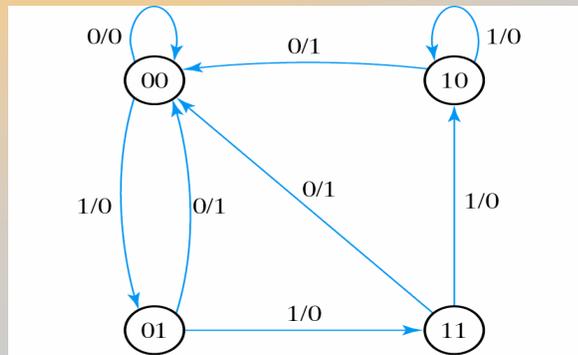
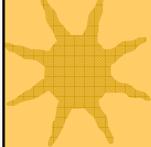


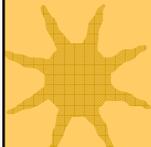
Fig. 5-16 State Diagram of the Circuit of Fig. 5-15



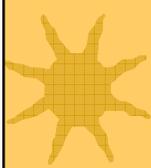
正反器輸入方程式或稱為輸入函數, 係採用正反器的輸入符號代表輸入方程式的變數而下標則表示正反器輸出的名稱



★ 舉例而言, 敘述一個具有輸入x和y的OR 閘連接到正反器的輸入D, 而它的輸出標示為Q, 其輸入方程式表示為



★
$$D_Q = x + y$$



★ 圖5-15電路之輸入方程式及輸出方程式可表示為

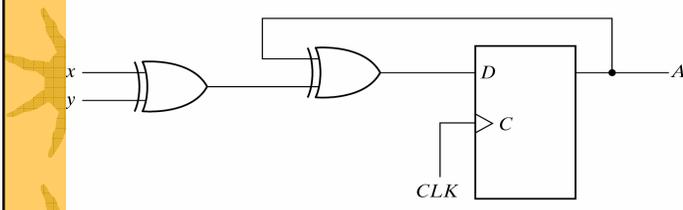
$$D_A = Ax + Bx$$

$$D_B = A'x$$

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



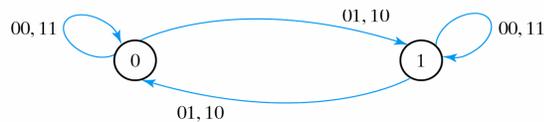
圖5-17 具有D型正反器的序向電路



(a) Circuit diagram

Present state	Inputs		Next state
	A	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



(c) State diagram

Fig. 5-17 Sequential Circuit with D Flip-Flop



D型正反器的分析

★圖5-17之輸入及輸出方程式為

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

$$A(t+1) = A \oplus x \oplus y$$

- ★因為對D型正反器而言, 其次一狀態與輸入D相同



JK正反器的分析

一個使用JK或T正反器的序向電路，其次態值可由下列程序獲得：

- ★1、用現態和輸入變數的觀點決定正反器的輸入方程式。
- ★2、列出每一個輸入方程式的二元值。
- ★3、使用相對應的正反器特性表決定狀態表中的次態值。

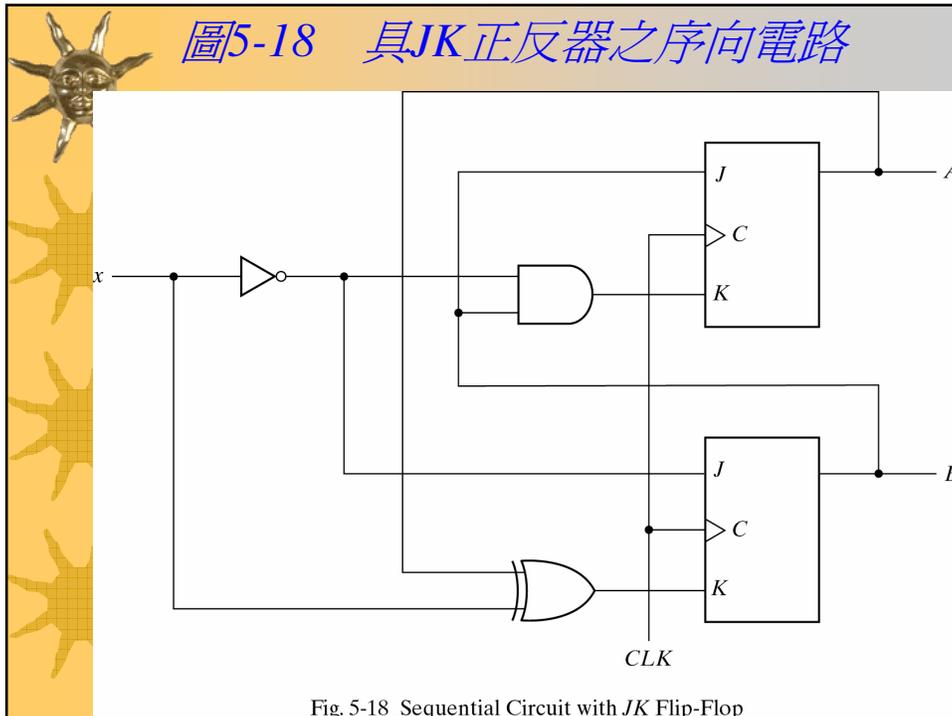


圖5-18之輸入及狀態方程式

- ★ 如圖5-18所示。電路的輸入方程式為

$$J_A = B$$

$$K_A = Bx'$$

$$J_B = x'$$

$$K_B = A'x + Ax' = A \oplus x$$
- ★ 正反器的特性程式可藉由將A，B取代Q的名稱而得

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

$$B(t+1) = JB' + K'B$$
- ★ 將 J_A, K_A 代入則A的狀態方程式為：

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

$$B(t+1) = x'B' + (A \oplus x)'B = B'x' + ABx + A'Bx'$$

圖5-18電路之狀態圖

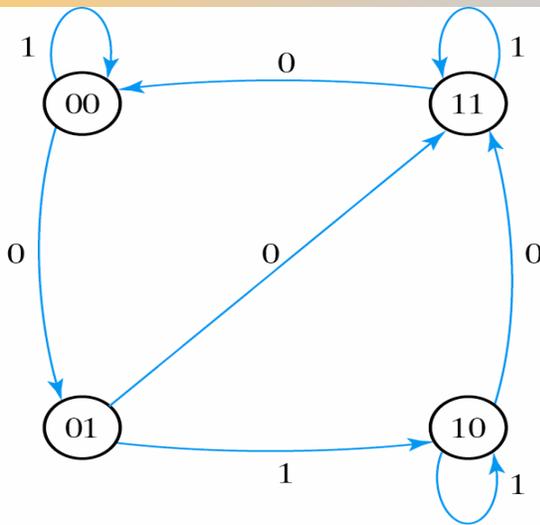
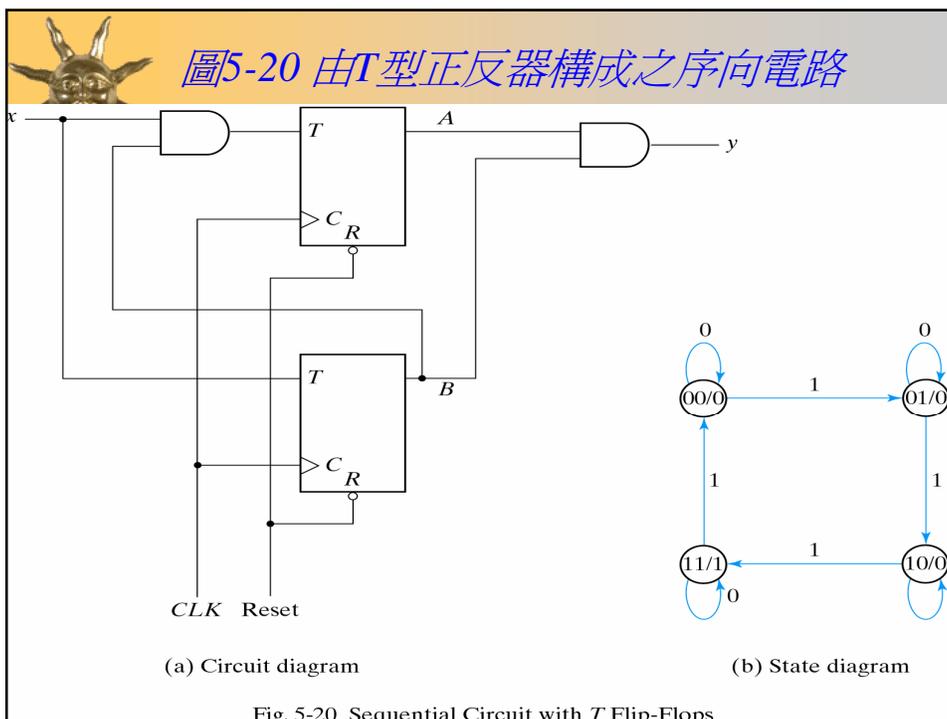


Fig. 5-19 State Diagram of the Circuit of Fig. 5-18

表5-4 圖5-18電路之狀態表

表5-4

目前狀態		輸入	次一狀態		正反器輸入			
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



T型正反器分析

- ★ T型正反器之特性方程式為

$$Q(t+1) = T \oplus Q = T'Q + TQ'$$
- ★ 圖5-20之輸入及輸出方程式為

$$T_A = Bx \quad T_B = x$$

$$y = AB$$
- ★ 而其次態值可由狀態方程式獲得

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

$$B(t+1) = x \oplus B$$



表5-5 圖5-20序向電路之狀態表

目前狀態		輸入	次一狀態		輸出
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



** 密利和莫爾模型

★在密利模型裡，輸出值是現態和輸入兩者的函數；但在莫爾模型中，其輸出值僅是現態的函數。討論上述模型時，有些書籍會稱這兩種序向電路為有限狀態機器（finite state machine，縮寫為FSM），屬密利模型的序向電路被稱為密利FSM或密利機，屬莫爾模型的序向電路被稱為莫爾FSM或莫爾機。

★圖5-15為密利機範例

★圖5-18及圖5-20則為莫耳機範例



5-5 序向電路的硬體描述語言

★動作模式 (behavioral modeling) :

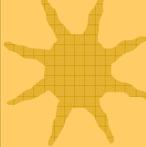
- ★在Verilog HDL中，有兩種動作敘述：**initial**和**always**。Initial動作在時間t=0開始執行，**always**動作則是重複地執行直到模擬完成為止。
- ★在一個模組中，可使用關鍵字**initial**和**always**來宣告動作，伴隨在一個敘述 (statement) 或一個區塊 (block) 敘述前後的關鍵字為**begin**和**end**。



- ★一個**initial**敘述只執行一次，它在模擬開始時動作並且在所有敘述執行完成後結束。

- ★以下是兩種產生任意運作時脈 (free-running clock) 的可能方式，

★ initial	initial
★ begin	begin
★ clock=1'b0 ;	clock=1'b0 ;
★ repeat (30)	#300
\$finish ;	
★ #10 clock= ~ clock ;	end
★ end	always
★	#10 clock= ~
clock ;	



HDL範例 5-1

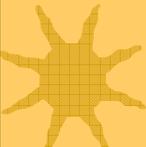
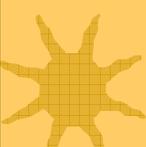
//Description of D latch (see Fig.5-6)



```

module D-latch ( Q,D,control ) ;
  output Q ;
  input D,control ;
  reg Q ;
always @ ( control or D )
if ( control ) Q=D ;    //same as : if
  ( control ==1 )
endmodule

```

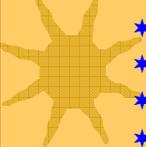
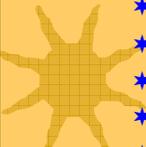




HDL 範例 5-2

```

* //D flip-flop
* module D-FF ( Q , D , CLK ) ;
*   output Q ;
*   input D , CLK ;
*   reg Q ;
* always @ ( posedge CLK )
*   Q=D ;
* endmodule
* // D flip-flop with asynchronous reset.
* module DFF ( Q , D , CLK , RST ) ;
*   output Q ;
*   input D , CLK ,RST ;
*   reg Q ;
* always @ ( posedge CLK or negedge RST )
*   if ( ~RST ) Q=1'b0 ;    //same as : if ( RST ==0 )
*   else Q=D ;
* endmodule

```


```

★ HDL 範例 5-3
★ //T flip-flop from D flip-flop and gates
★ module TFF ( Q ,T ,CLK , RST ) ;
★   output Q ;
★   input T ,CLK , RST ;
★   wire DT ;
★   assign DT = Q ^ T
★ //Instantiate the D flip-flop
★   DFF TF1 ( Q ,DT ,CLK , RST ) ;
★ Endmodule
★ //JK flip-flop from D flip-flop and gates
★ module JKFF ( Q , J, K, CLK, RST ) ;
★   output Q ;
★   input J, K, CLK, RST ;
★   wire JK ;
★   assign JK= ( J & ~Q ) | ( ~K & Q )
★

```

```

HDL 範例 5-3:
//Instantiate the D flip-flop
DFF JK1 ( Q ,JK ,CLK , RST ) ;
endmodule

// D flip-flop
module DFF ( Q , D, CLK, RST ) ;
output Q ;
input D, CLK, RST ;
reg Q ;
always @ (posedge CLK or negedge RST)
if ( ~RST ) Q = 1'b0 ;
else Q = D ;
endmodule

```



★ HDL 範例 5-4:

```

★ //Functional description of JK flip-flop
★ module JK-FF ( J ,K ,CLK , Q , Qnot ) ;
★   output Q ,Qnot ;
★   input J , K, CLK ;
★   reg Q ;
★   assign Qnot =~ Q ;
★   always @ ( posedge CLK )
★       case ( {J,K} )
★       2'b00 : Q=Q ;
★       2'b01 : Q=1'b0 ;
★       2'b10 : Q=1'b1 ;
★       2'b11 : Q=~Q ;
★       endcase
★ endmodule

```



★ HDL 範例 5-5:

```

★ //Mealy state diagram (Fig.5-16)
★ module Mealy-mdl ( x , y , CLK, RST ) ;
★   input x , CLK, RST ;
★   output y ;
★   reg y ;
★   reg [ 1: 0 ] Prstate, Nxtstate ;
★   parameter S0 =2'b00, S1 =2'b01, S2 =2'b10, S3 =2'b11 ;
★   always @ ( posedge CLK or negedge RST )
★       if ( ~RST ) Prstate = S0 ; // Initialize to state S0
★       else Prstate = Nxtstate ; //Clock operations
★   always @ ( Prstate or x ) //Determine next state
★       case ( Prstate )

```

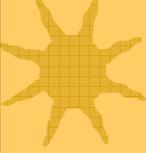
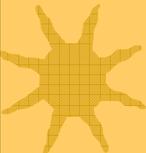
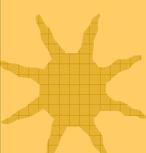


★ HDL 範例 5-5~:

```

★      S0 : if ( x ) Nxtstate = S1 ;
★          else Nxtstate = S0 ;
★      S1 : if ( x ) Nxtstate = S3 ;
★          else Nxtstate = S0 ;
★      S2 : if ( ~x ) Nxtstate = S0 ;
★          else Nxtstate = S2 ;
★      S3 : if ( ~x ) Nxtstate = S2 ;
★          else Nxtstate = S0 ;
★      endcase
★ always @ ( Prstate or x ) //Evaluate output
★      case ( Prstate )
★          S0 : y=0 ;
★          S1 : if ( x ) y=1'b0 ; else y = 1'b1 ;
★          S2 : if ( x ) y=1'b0 ; else y = 1'b1 ;
★          S3 : if ( x ) y=1'b0 ; else y = 1'b1 ;
★      endcase
★ endmodule

```

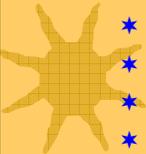
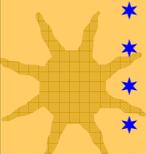





★ HDL 範例 5-6:

```

★ // Moore state diagram (Fig. 5-19)
★ module Moore-mdl ( x , AB, CLK, RST ) ;
★     input x , CLK, RST ;
★     output [ 1 : 0 ] AB ;
★     reg [ 1 : 0 ] state ;
★     parameter S0 =2'b00, S1 =2'b01, S2 =2'b10, S3 =2'b11 ;
★     if ( ~RST ) state = S0 ; // Initialize to state S0
★     always @ ( posedge CLK or negedge RST )
★     else
★     case ( state )
★         S0 : if ( x ) state = S1 ; else state = S0 ;
★         S1 : if ( x ) state = S2 ; else state = S3 ;
★         S2 : if ( ~x ) state = S3 ; else state = S2 ;
★         S3 : if ( ~x ) state = S0 ; else state = S3 ;
★     endcase
★     assign AB = state // output of flip-flops
★ endmodule

```




★ HDL 範例 5-7:

```
★ // Structural description of sequential circuit
★ // See fig. 5-20(a)
★ module Tcircuit ( x , y, A, B, CLK, RST ) ;
★   input  x , CLK, RST ;
★   output y, A, B ;
★   wire  TA, TB ;
★ // Flip-flop input equations
★   assign TB = x,
★           TA = x & B ;
★ // Output equation
★   assign y = A & B ;
★ // Instantiate T flip-flops
★   T_FF BF ( B, TB, CLK, RST ) ;
★   T_FF AF ( A, TA, CLK, RST ) ;
★ endmodule
```



★ HDL 範例 5-7~:

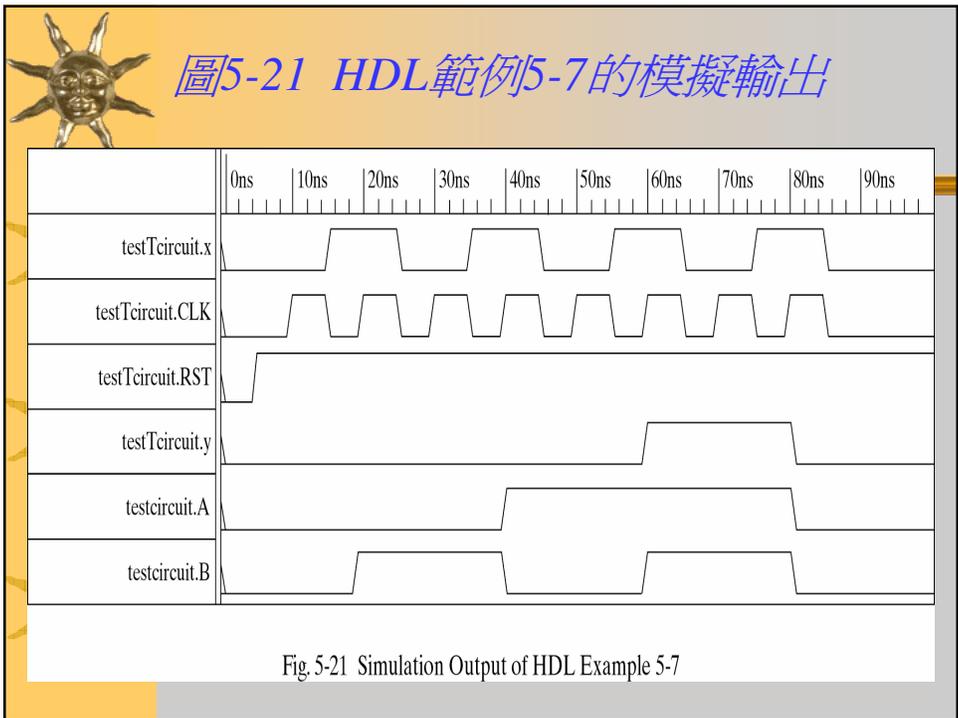
```
★ // T flip-flop
★ module T_FF ( Q, T, CLK, RST )
★   output Q ;
★   input  T, CLK, RST ;
★   reg Q ;
★   always @ ( posedge CLK or negedge RST )
★     if ( ~RST ) Q = 1'b0 ;
★     else Q = Q ^ T ;
★ endmodule
★
★ // Stimulus for testing sequential circuit
★ module testTcircuit ;
★   reg x, CLK, RST ; // inputs for circuit
★   wire y, A, B ; // output from circuit
★   Tcircuit TC ( x, y, A, B, CLK, RST ) ; // instantiate circuit
```

★ HDL 範例 5-7~~:

```

★ initial
★   begin
★     RST = 0 ;
★     CLK = 0 ;
★     #5 RST = 0 ;
★     repeat ( 16 )
★       #5 CLK = ~CLK ;
★     end
★ initial
★   begin
★     x = 0 ;
★     #15 x = 1 ;
★     repeat ( 8 )
★       #10 x = ~x ;
★     end
★ end module

```





5-6 狀態簡化與指定

★圖5-22 狀態圖

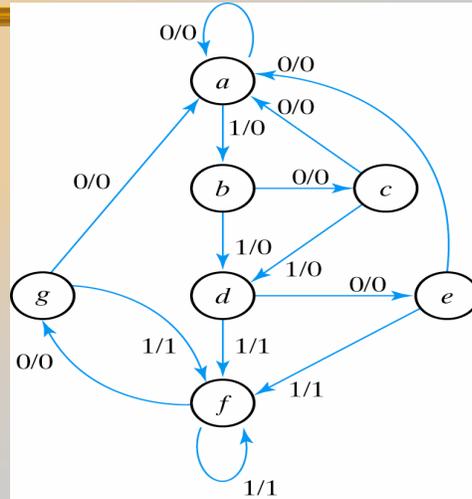


Fig. 5-22 State Diagram



★表5-6 圖5-22狀態圖之狀態表

目前 狀態	次一狀態		輸出	
	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



- ★由表5-6可知狀態 g及e為相等狀態,
- ★故圖5-22可簡化為5個狀態,如圖5-23所示

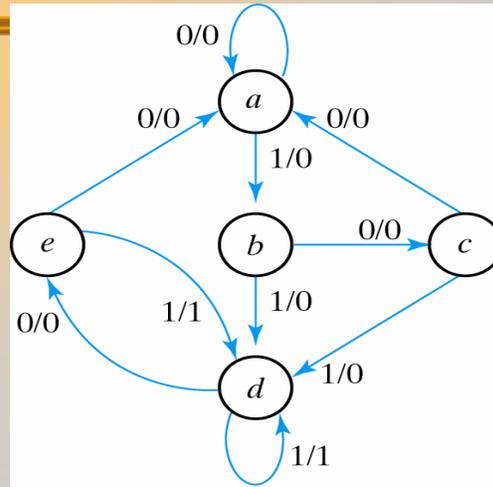


Fig. 5-23 Reduced State Diagram



- ★狀態表簡化
- ★由圖5-23可推得表5-7

目前 狀態	次一狀態		輸出	
	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



★表5-8簡化後之狀態表

目前 狀態	次一狀態		輸出	
	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

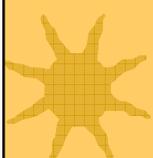
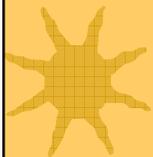
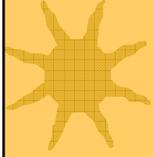


★表5-9 三種可能的二進位狀態指定

目前 狀態	第一種指定	第二種指定	第三種指定
	二進位	葛雷碼	One-shot
a	000	000	00001
b	001	001	00010
c	010	011	00100
d	011	010	01000
e	100	110	10000



★表5-10 將第一種狀態指定用於簡化後之狀態表



目前狀態	次一狀態		輸出	
	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

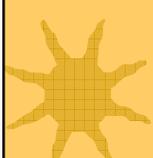
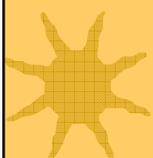
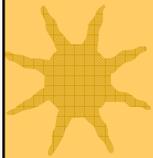


5-7 設計程序

★ 同步序向電路的設計程序可摘要如下列步驟

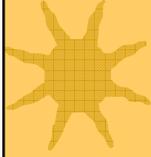
★ :

- ★ 1、從文字敘述及所需要的操作規格，獲得電路的狀態圖。
- ★ 2、如果需要，簡化狀態數量。
- ★ 3、指定狀態的二元值。
- ★ 4、獲得二元編碼的狀態表。
- ★ 5、選擇欲使用的正反器型式。
- ★ 6、推導出已簡化的輸入方程式及輸出方程式。
- ★ 7、繪製邏輯圖。





- ★設計一個電路可偵測出一串位元中有
- ★三個或更多個連續的1出現



★圖5-24 順序偵測器之狀態圖

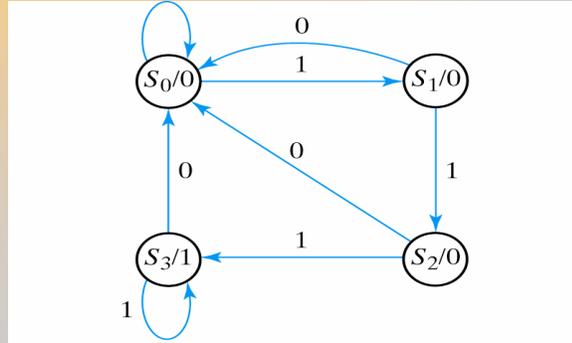
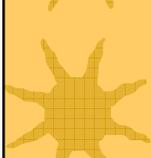
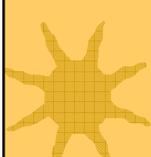
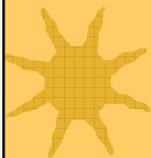


Fig. 5-24 State Diagram for Sequence Detector



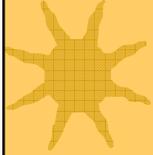
★表5-11 順序偵測器之狀態表

目前狀態		輸入	次一狀態		輸出
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	1	0	0
0	1	1	0	0	0
1	0	0	0	1	0
1	0	1	1	1	0
1	1	0	0	1	1
1	1	1	1	0	1





★ 為設計此偵測器電路，選擇兩個D正反器來描述四個狀態，而且它們的輸出分別為A與B。有一個輸入x及一個輸出y。



★ 將表5-11化簡如下：

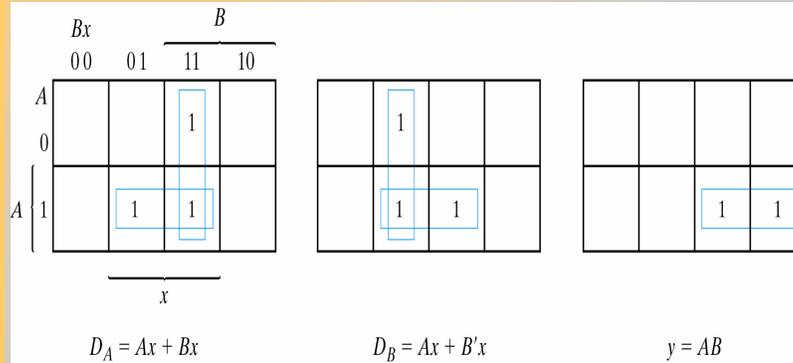
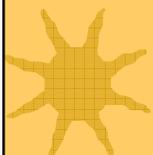
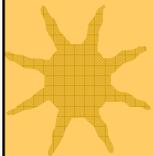


Fig. 5-25 Maps for Sequence Detector



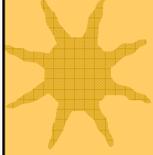
★ 正反器的輸入方程式可直接由A與B的次態行獲得，並可表示成最小項的和為



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

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

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

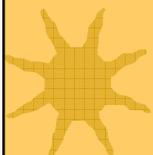


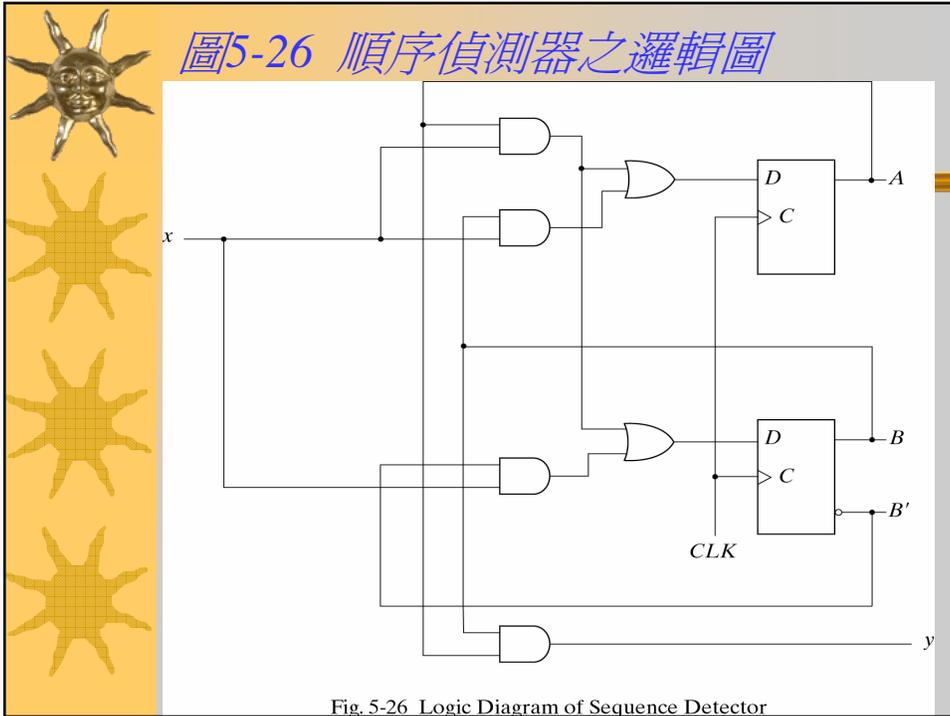
★ 簡化後之方程式為

$$D_A = Ax + Bx$$

$$D_B = Ax + B'x$$

$$y = AB$$





激勵表 (Excitation Tables)

- ★ 在設計程序中，我們通常知道從現態到
- ★ 次態的轉變，並且希望能找到造成如此
- ★ 轉變的正反器輸入條件。有鑑於此，對
- ★ 一已知的狀態改變，我們需要一個表
- ★ 來列出必要的輸入條件，這樣的表稱之
- ★ 為激勵表。



表5-12 正反器之激勵表

★ JK正反器之激勵表

★ T正反器之激勵表

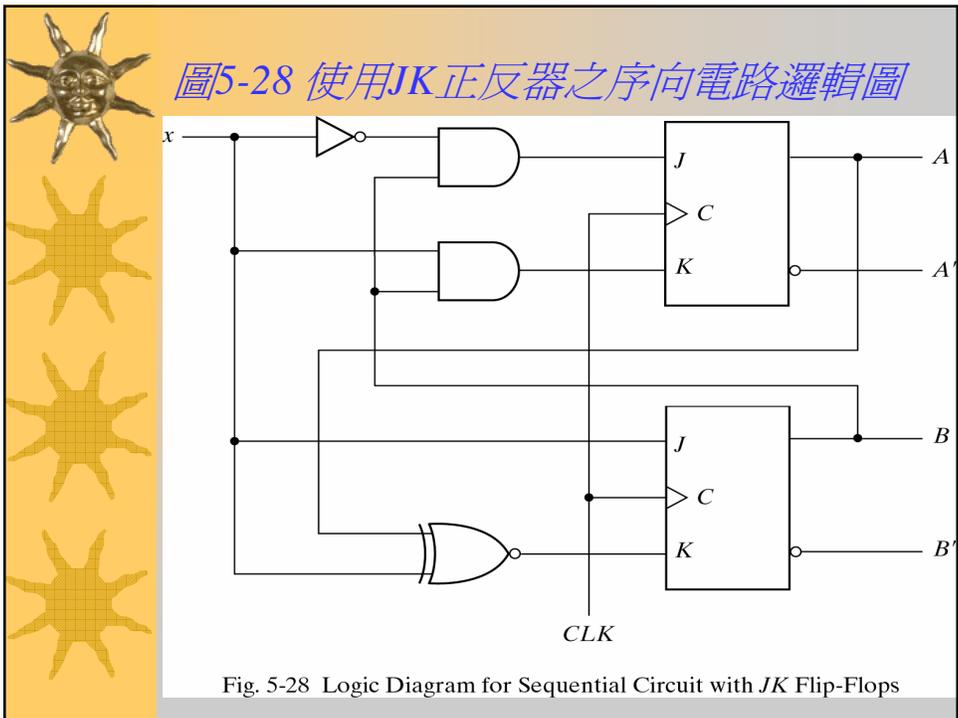
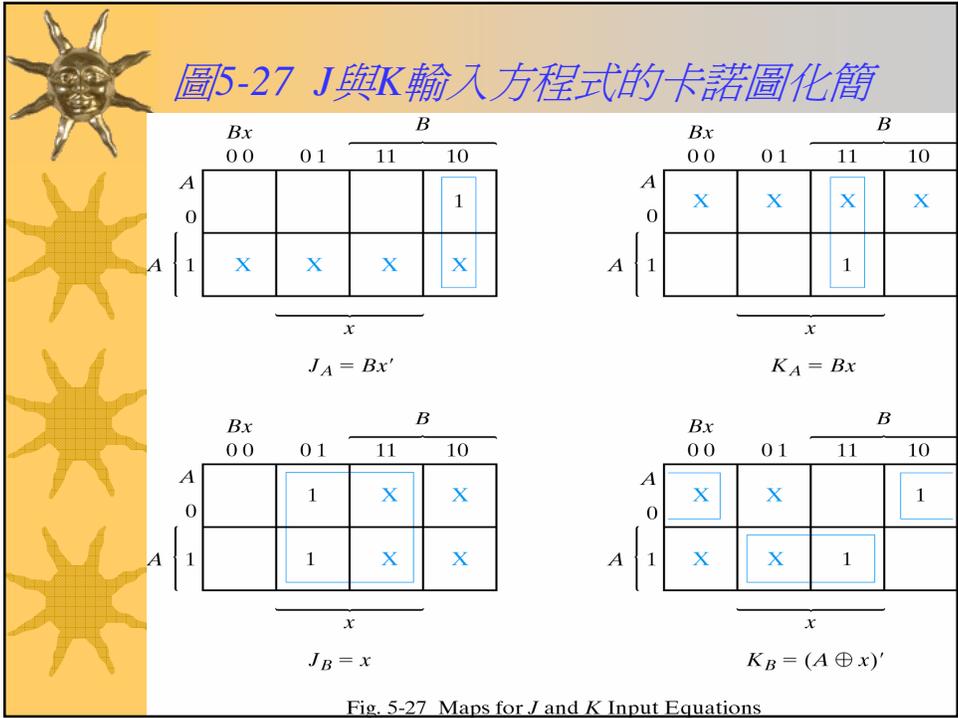
Q(t)	Q(t+1)	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

Q(t)	Q(t+1)	T
0	0	0
0	1	1
1	0	1
1	1	0



表5-13 狀態表及JK正反器之輸入

目前狀態		輸入 x	次一狀態		正反器輸入			
A	B		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





使用T正反器設計3位元二進位計數器

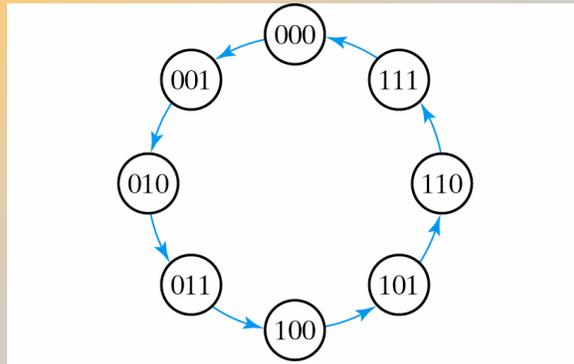
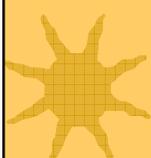
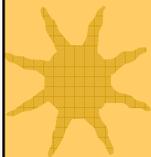
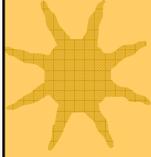


Fig. 5-29 State Diagram of 3-Bit Binary Counter



★表5-14 3位元二進位計數器之狀態表

目前狀態			次一狀態			正反器輸入		
A ₂	A ₁	A ₀	A ₂	A ₁	A ₀	T _{A2}	T _{A1}	T _{A0}
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	1	1
1	1	1	0	0	0	1	1	1

