Digital Systems Design and Laboratory
[12. Registers and Counters]

Chung-Wei Lin
cwlin@csie.ntu.edu.tw
CSIE Department
National Taiwan University

We would like to thank Prof. Hui-Ru Jiang (NTU) for contributing the lecture material

Sequential Logic Design

- Unit 11: Latches and Flip-Flops
  - Basic unit

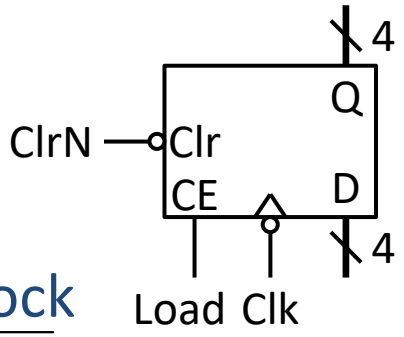
差別：flip–flop 只有在 active edge 才會改變
相同：都有記憶功能

- Unit 12: Registers and Counters
  - Simple sequential circuit
- Units 13--15: Finite State Machines
  - Complex sequential circuit
- Unit 16: Summary
  - Put it all together

Outline

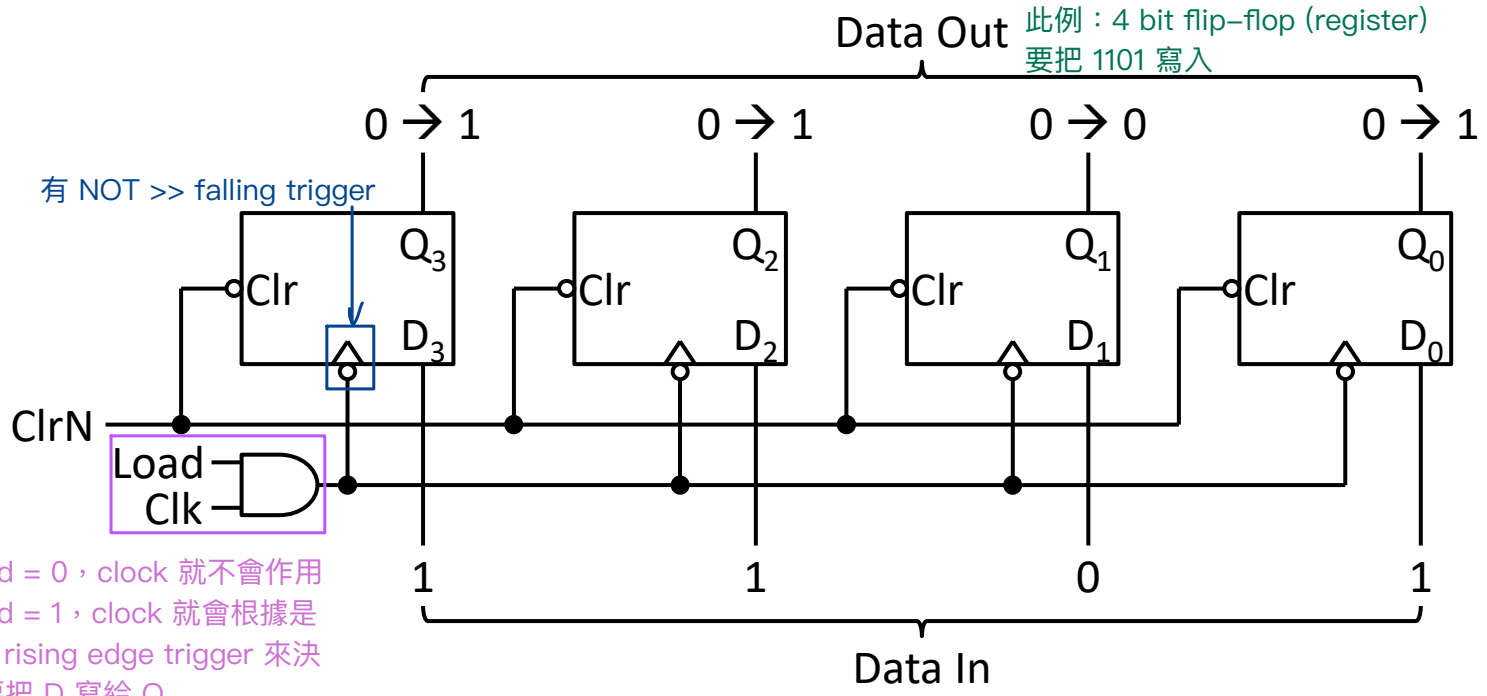
- **Registers and Register Transfers**
- Shift Registers
- Design of Binary Counters
- Counters for Other Sequences
- Counter Design Using S-R and J-K Flip-Flops
- Derivation of Flip-Flop Input Equations

Registers (1/2)



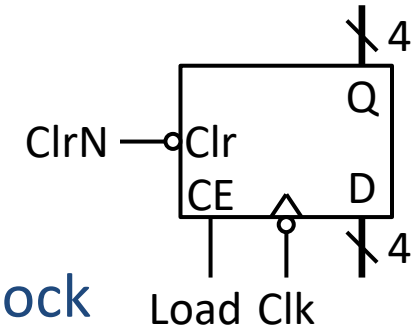
可以想成一對 flip-flop 會接在一起同時運作

- Register: a group of D flip-flops with a common clock
- Example
  - 4-bit D flip-flop registers with Data, Load, Clear (ClrN), Clock (Clk)
- First Implementation: gated clock
  - When Load = 1, load data at D to Q at Clk falling

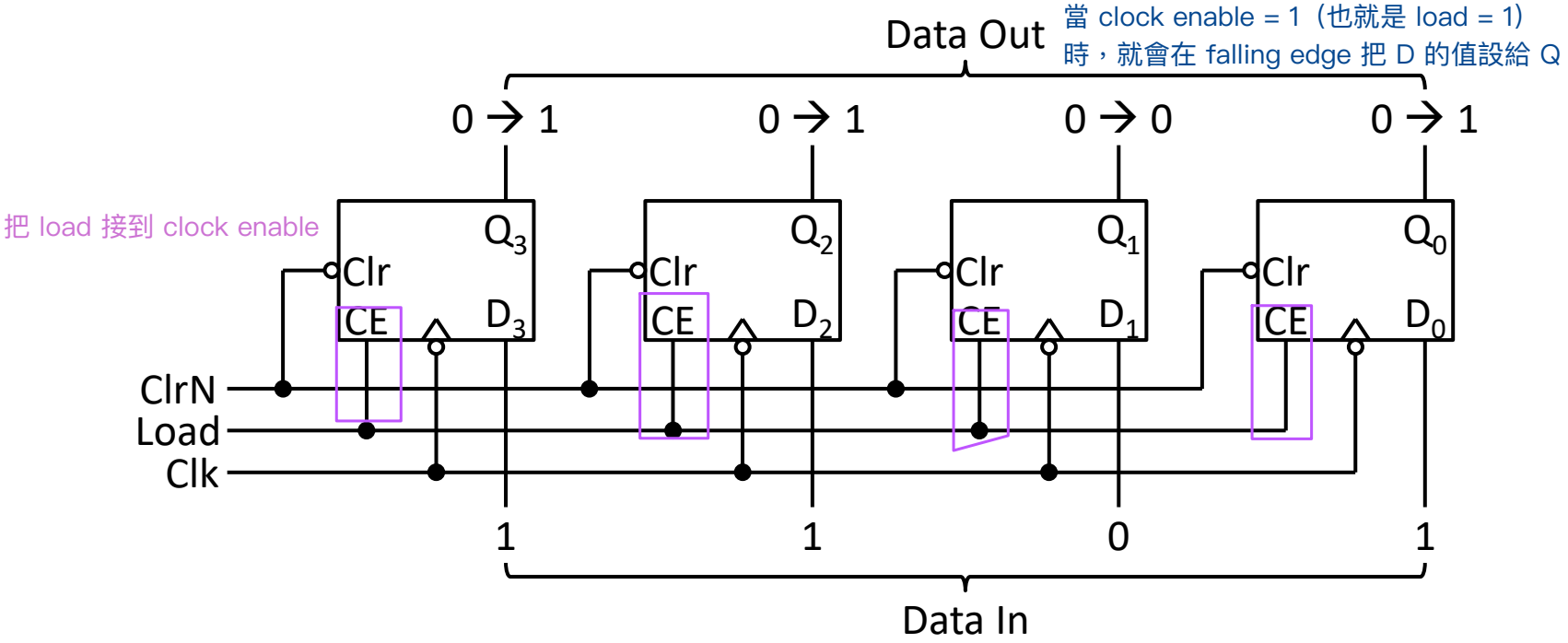


- 如果 load = 0，clock 就不會作用
- 如果 load = 1，clock 就會根據是 falling / rising edge trigger 來決定要不要把 D 寫給 Q

Registers (2/2)



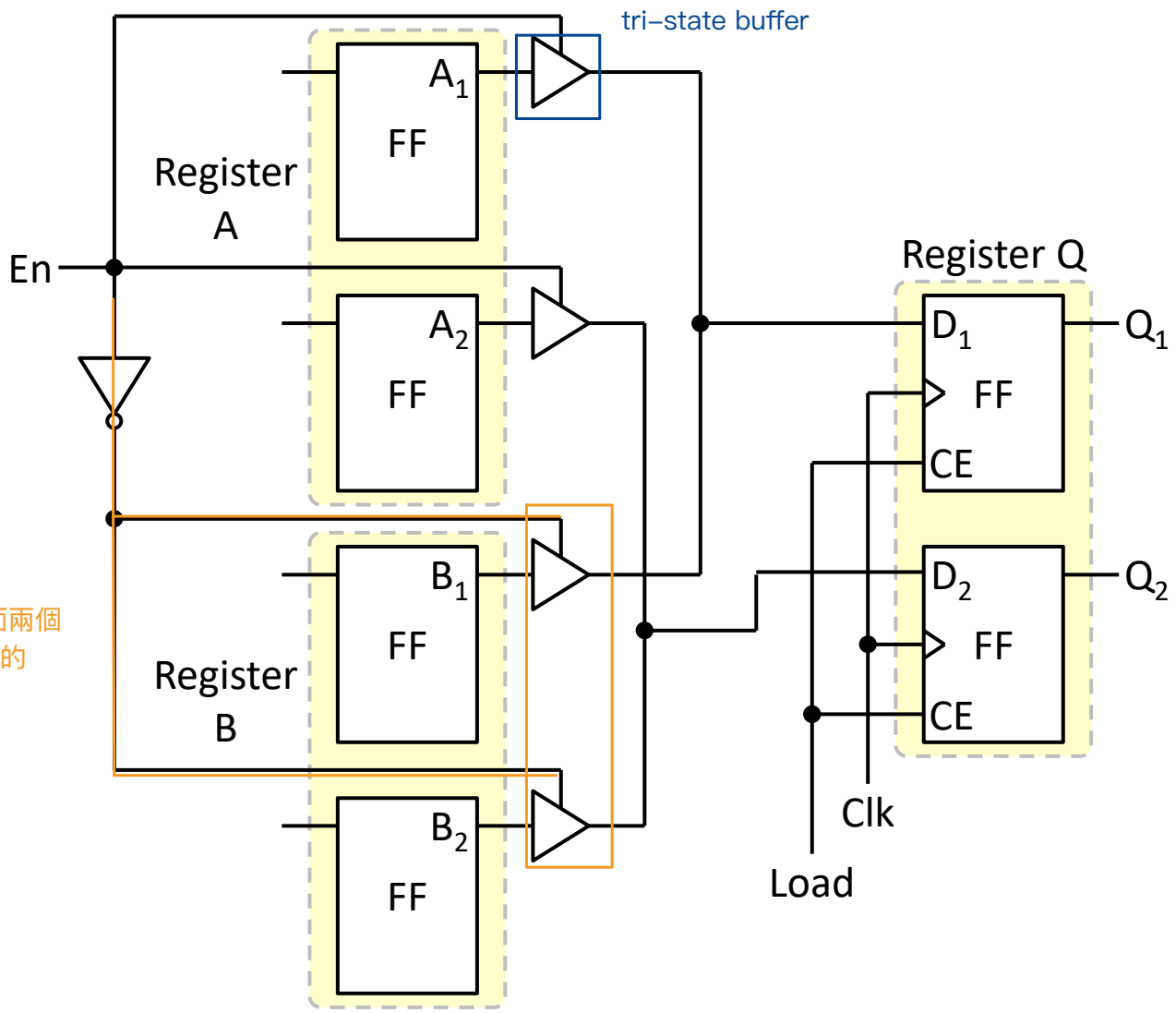
- Register: a group of D flip-flops with a common clock
- Example
  - 4-bit D flip-flop registers with Data, Load, Clear (ClrN), Clock (Clk)
- Second implementation: clock enable
  - When Load = 1, load data at D to Q at Clk falling



Data Transfer between Registers

- 2-to-1 MUX
  - If En = 1, Q = A
  - If En = 0, Q = B

(enable)

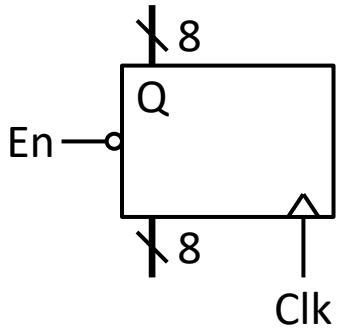


當 enable = 1 時，下面兩個 buffer 被關起來（上面的 buffer 被打開）

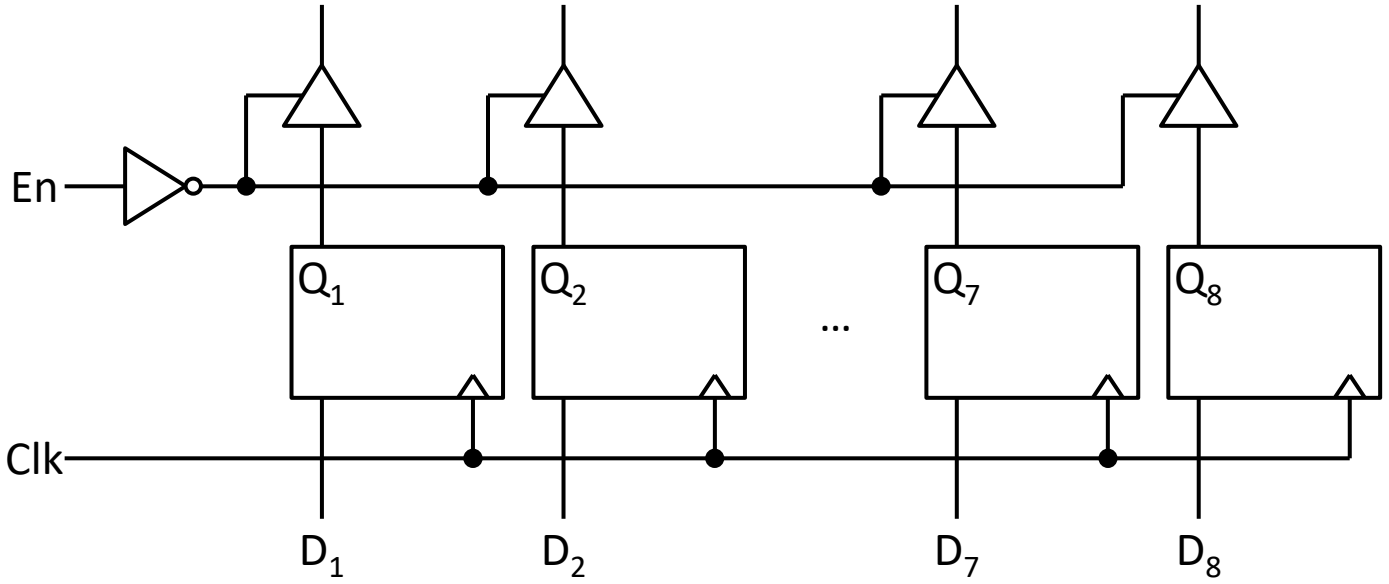
enable = 0 則相反

8-Bit Register with Tri-State Output (1/2)

- Symbol

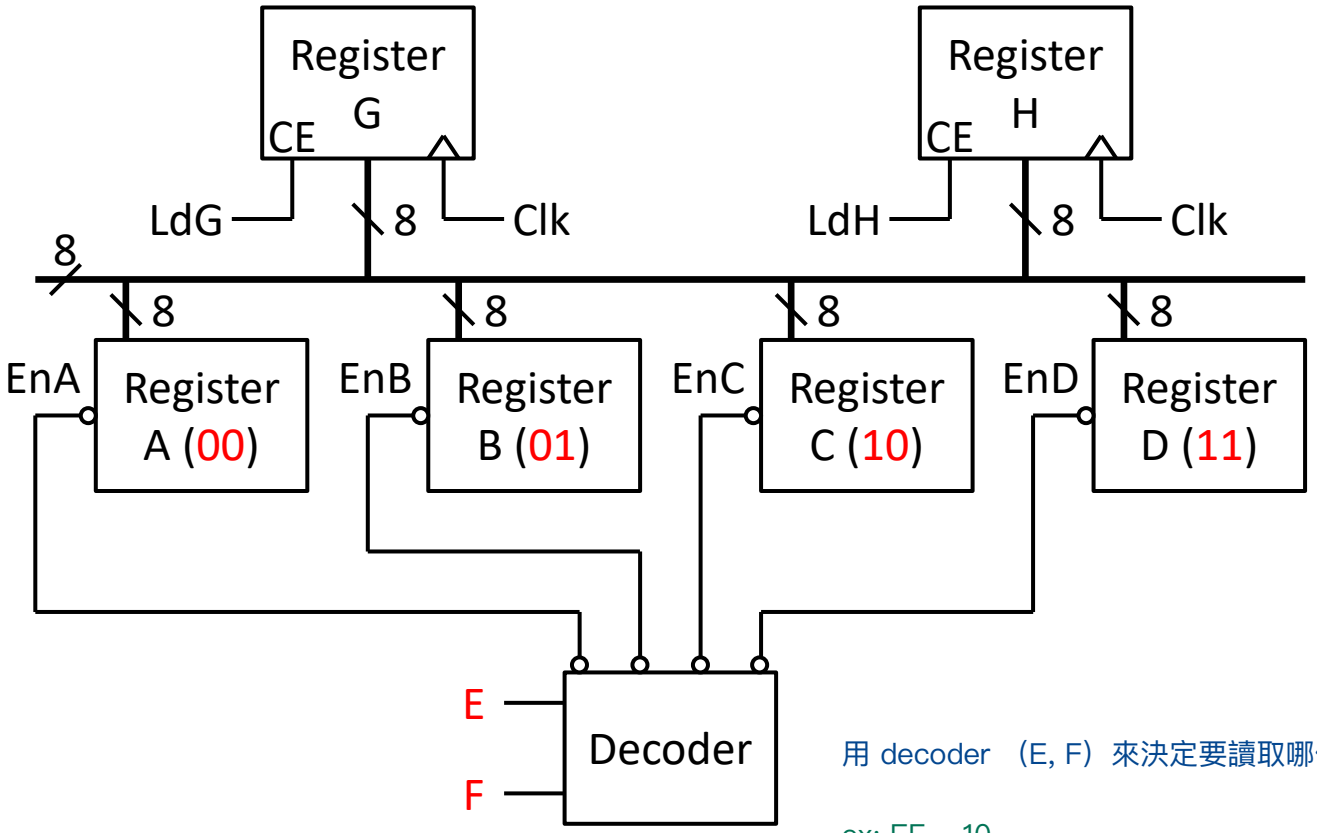


- Logic diagram



8-Bit Register with Tri-State Output (2/2)

- Data transfer



用 decoder （E, F）來決定要讀取哪個 register 的值

ex: EF = 10
>> register C turn on, 其餘 turn off

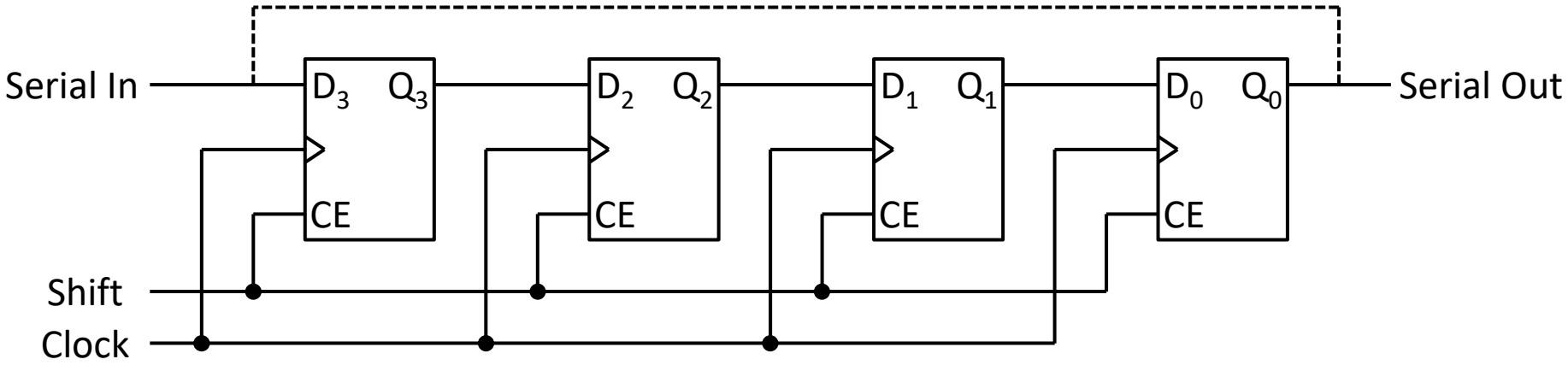
Outline

- Registers and Register Transfers
- **Shift Registers**
- Design of Binary Counters
- Counters for Other Sequences
- Counter Design Using S-R and J-K Flip-Flops
- Derivation of Flip-Flop Input Equations

Shift Registers (1/2)

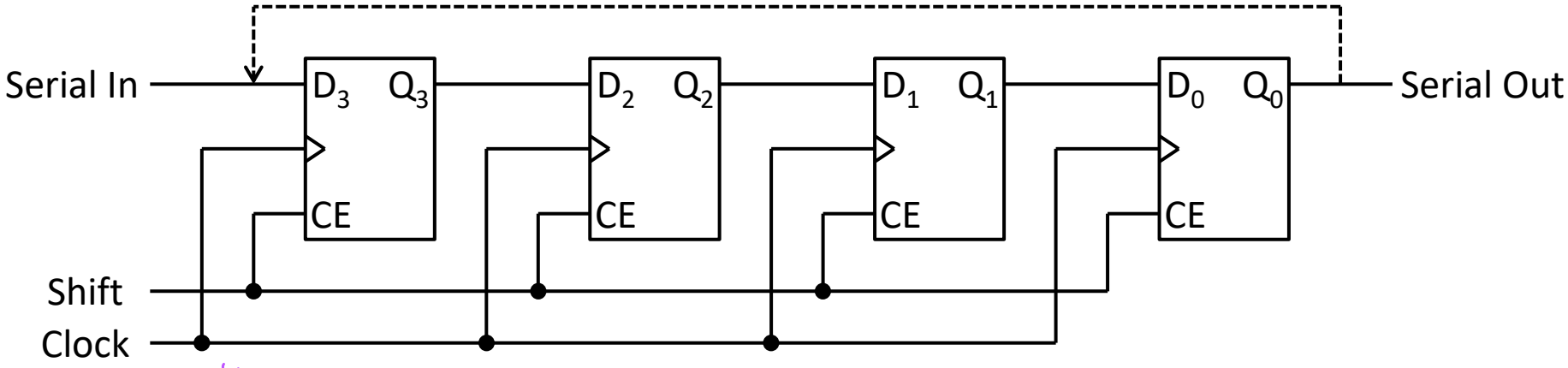
也是 group of flip–flops ，前面的 flip–flop 的 output 會傳給下個 flip–flop

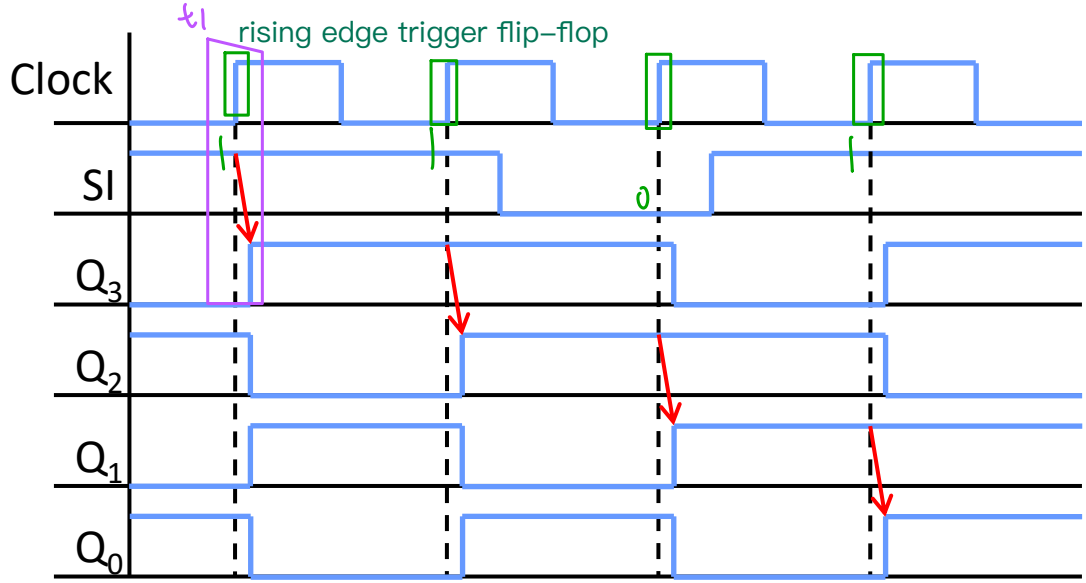
- Shift register: a group of flip-flops where binary data can be stored and shifted left or right when a shift signal is applied
- Example: 4-bit right-shift register



Shift Registers (2/2)

- Timing diagram of a 4-bit right-shift register





Initial $Q_3Q_2Q_1Q_0$ = 0101

SI = 1, 1, 0, 1

$Q_3Q_2Q_1Q_0$ = 0101

t1 1010

1101

0110

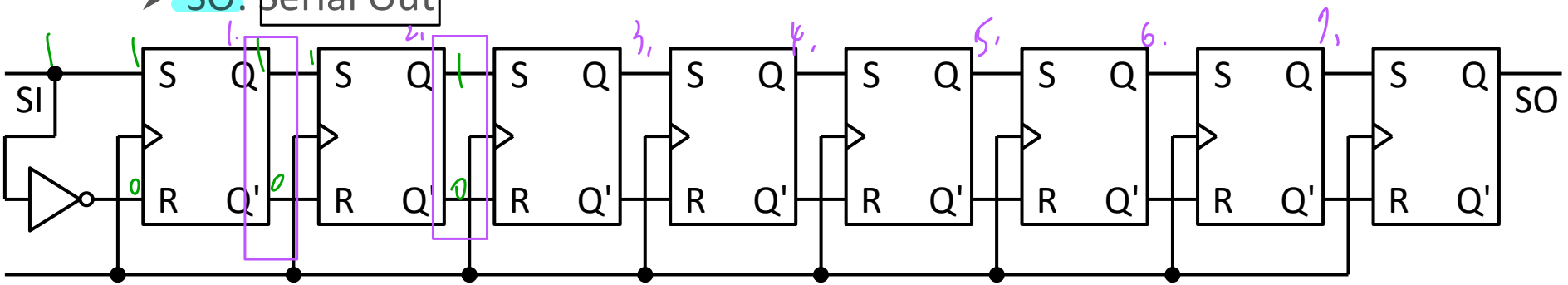
1011

在 t1 的 rising edge 時，SI 的 1 寫入 Q_3，Q_3的值 shift 到 Q_2，Q_2的值 shift 到 Q_1，以此類推，最後一個 Q_0 的 1 被擠掉

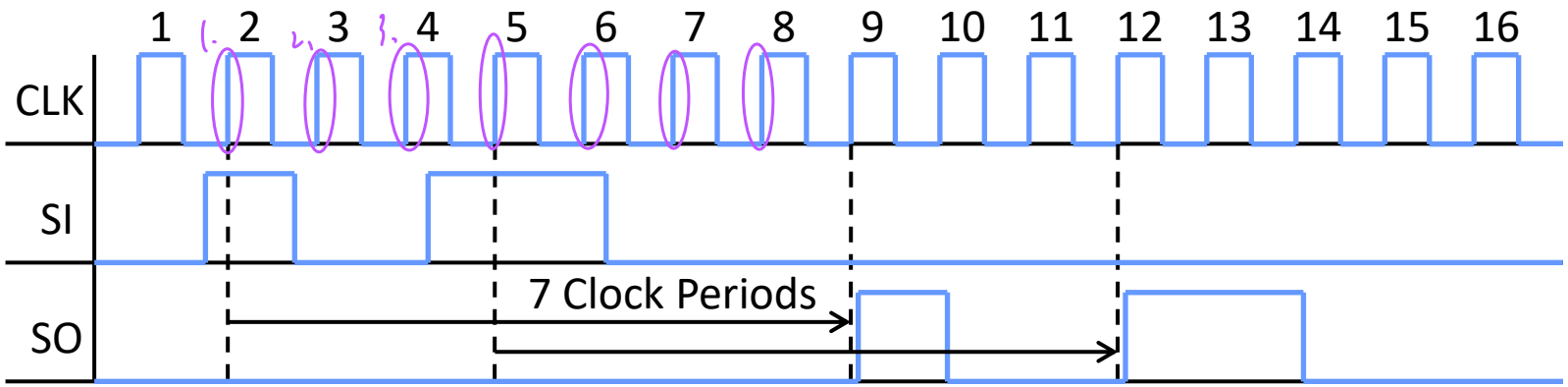
每次 rising edge Q_3 的值就不見了，其餘平移

N-bit Serial-In Serial-Out Shift Registers

- Take (n-1) cycles to output data
  - SI: Serial In
  - SO: Serial Out

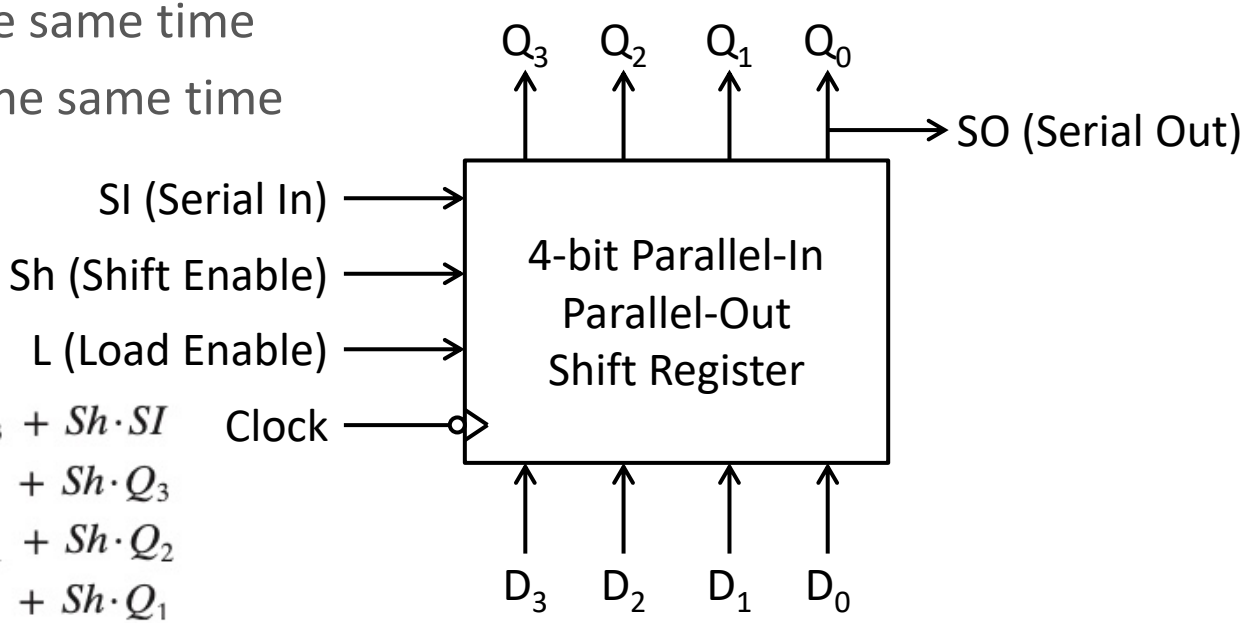


在第二個 rising edge 前把 SI 拉高了，所以第二個 rising edge 時第一個 S >> Q
第三個 rising edge 時反映在第二個 S >> Q



Parallel-In Parallel-Out Right Shift Register (1/2)

- **Parallel-in parallel-out (PIPO)**
  - Load all data at the same time
  - Read out data at the same time



next state equations

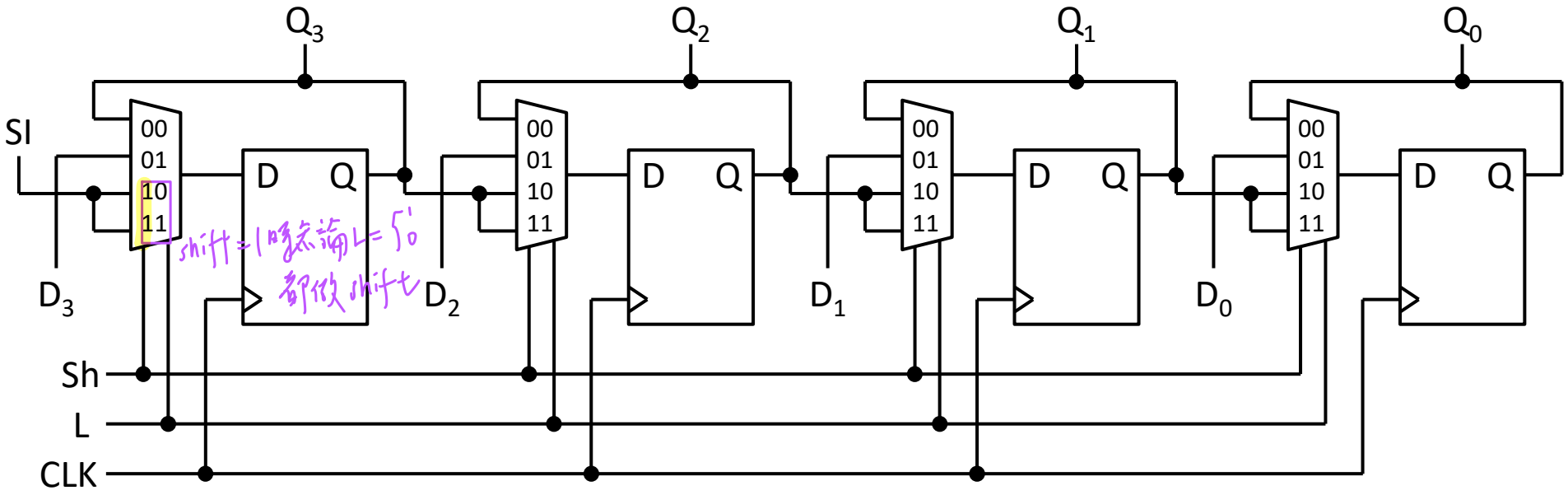
$$Q_3^+ = Sh' \cdot L' \cdot Q_3 + Sh' \cdot L \cdot D_3 + Sh \cdot SI$$
$$Q_2^+ = Sh' \cdot L' \cdot Q_2 + Sh' \cdot L \cdot D_2 + Sh \cdot Q_3$$
$$Q_1^+ = Sh' \cdot L' \cdot Q_1 + Sh' \cdot L \cdot D_1 + Sh \cdot Q_2$$
$$Q_0^+ = Sh' \cdot L' \cdot Q_0 + Sh' \cdot L \cdot D_0 + Sh \cdot Q_1$$

Sh (Shift)	L (Load)	Q_3^+	Q_2^+	Q_1^+	Q_0^+	Action
0	0	Q_3	Q_2	Q_1	Q_0	No Change
0	1	D_3	D_2	D_1	D_0	Load
1	X	SI	Q_3	Q_2	Q_1	Right Shift

shift = 1 時，不管 load 是多少都做平移

Parallel-In Parallel-Out Right Shift Register (2/2)

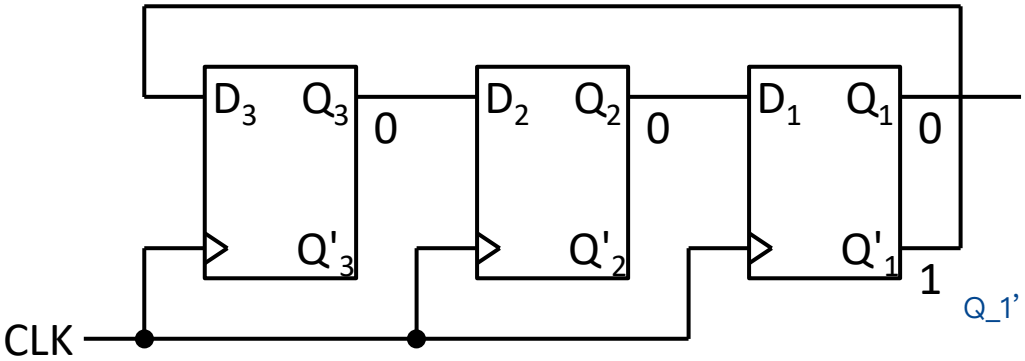
- Implement using flip-flops and MUXes



Sh (Shift)	L (Load)	Q_3^+	Q_2^+	Q_1^+	Q_0^+	Action
0	0	Q_3	Q_2	Q_1	Q_0	No Change
0	1	D_3	D_2	D_1	D_0	Load
1	X	SI	Q_3	Q_2	Q_1	Right Shift

Shift Register with Inverted Feedback

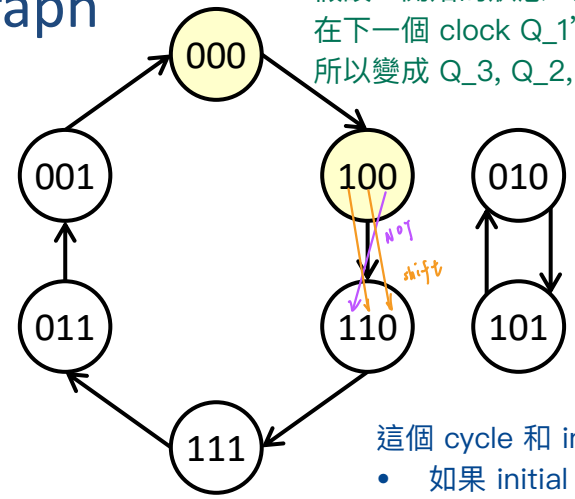
- Johnson counter: a shift register with inverted feedback
  - Counter: a circuit that cycles through a fixed sequence of states



Q_1' 接回 D_3

- State graph

假設一開始的狀態 Q_3, Q_2, Q_1 = 0, 0, 0
在下一個 clock Q_1' = 1 送回 Q_3, Q_3 shift 到 Q_2，Q_2 shift 到 Q_1
所以變成 Q_3, Q_2, Q_1 = 1, 0, 0



這個 cycle 和 initial state 有關

- 如果 initial state = 010，就會在 010, 101 間跳來跳去

Outline

- ❑ Registers and Register Transfers
- ❑ Shift Registers
- ❑ **Design of Binary Counters**
- ❑ Counters for Other Sequences
- ❑ Counter Design Using S-R and J-K Flip-Flops
- ❑ Derivation of Flip-Flop Input Equations

T flip–flop：決定翻轉或不翻轉
D flip–flop：直接給值

T=1 : toggle
T=0 : unchanged

$Q^+ = D$ (when edge is triggered)

Counting 0--7 (1/2)

用 K–map 做 minimization 來決定 T_B, T_C

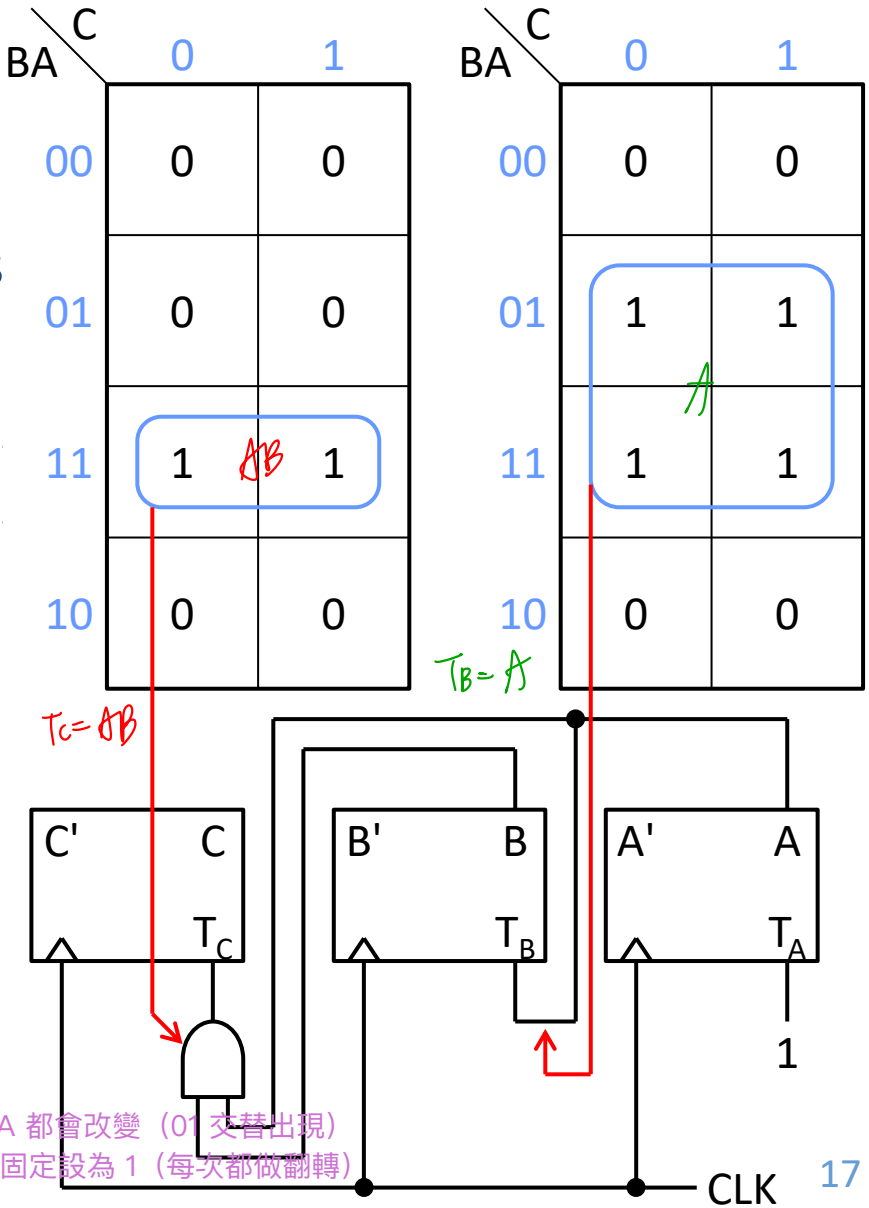
所有 flip–flop 受同個 clock 控制

- **Synchronous** counter: flip-flops are synchronized by a clock
- First implementation: T flip-flops

T = 1：翻轉，T = 0：不變

Present State			Next State			Flip-Flop Inputs (By Observation)		
C	B	A	C^+	B^+	A^+	T_C	T_B	T_A
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	0	1
1	1	1	0	0	0	1	1	1

binary counter：我們想從 0 數到 7，到 7 以後再回到 0



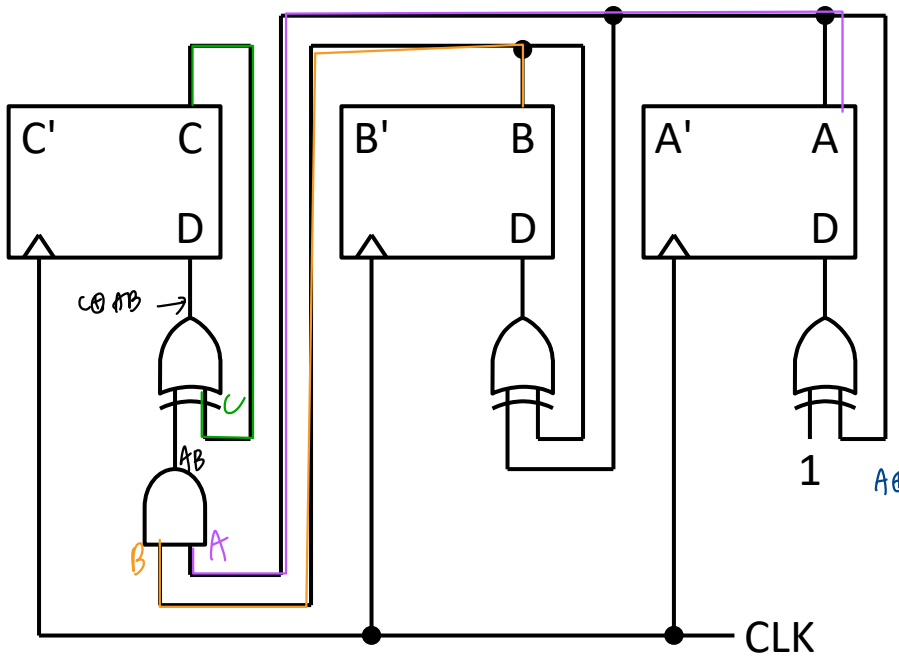
因為每次 A 都會改變（01 交替出現）
所以 T_A 固定設為 1（每次都做翻轉）

Counting 0--7 (2/2)

D 的值給多少，就會在 active edge 時把 D 的值給多少

Second implementation: D flip-flops

- $D_A = A^+ = A'$
- $D_B = B^+ = BA' + B'A = B \oplus A$
  - B changes when A = 1
- $D_C = C^+ = C'BA + CB' + CA' = C \oplus BA$
  - C changes when A = B = 1



Present State			Next State		
C	B	A	C^+	B^+	A^+
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

C'BA
CB'
CA'

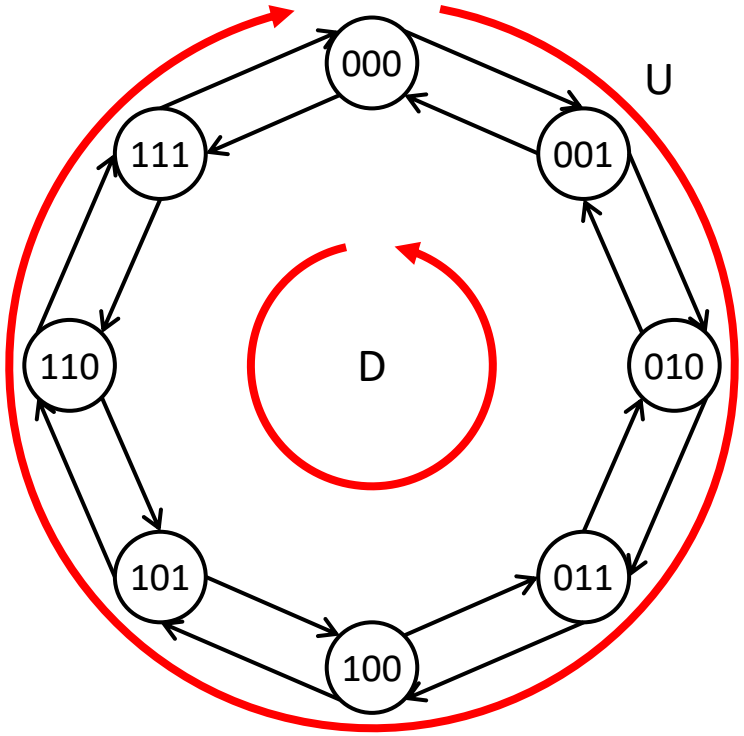
A, B 不同時 B^+ = 1

A⊕1 { A=0 : 0⊕1 = 1 ; A=1 : 1⊕1 = 0

0 變 1，1 變 0

Up-Down Counter

- U and D control "up" and "down"
  - Do not allow U = D = 1
  - $D_A = A^+ = A \oplus (U + D)$
  - $D_B = B^+ = B \oplus (UA + DA')$ 原：BOA
  - $D_C = C^+ = C \oplus (UBA + DB'A')$ 原：COBA



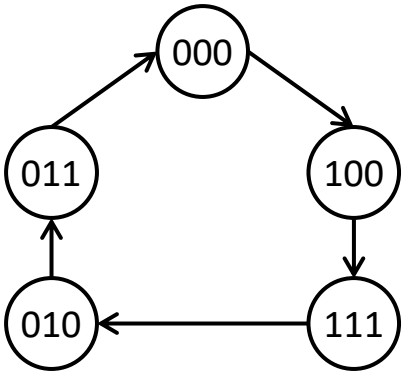
CBA	$C^+B^+A^+$	
	U	D
000	001	111
001	010	000
010	011	001
011	100	010
100	101	011
101	110	100
110	111	101
111	000	110

Outline

- ❑ Registers and Register Transfers
- ❑ Shift Registers
- ❑ Design of Binary Counters
- ❑ **Counters for Other Sequences**
- ❑ Counter Design Using S-R and J-K Flip-Flops
- ❑ Derivation of Flip-Flop Input Equations

State Diagram of Counter

- What if the sequence is not in straight binary order?



Present State			Next State		
C	B	A	C^+	B^+	A^+
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

K-Map Derivation

❑ Next states

記得注意 midterm 順序

BA \ C	0	1
00	1	1
01	X	X
11	0	0
10	0	X

C^+

BA \ C	0	1
00	0	1
01	X	X
11	0	1
10	1	X

B^+

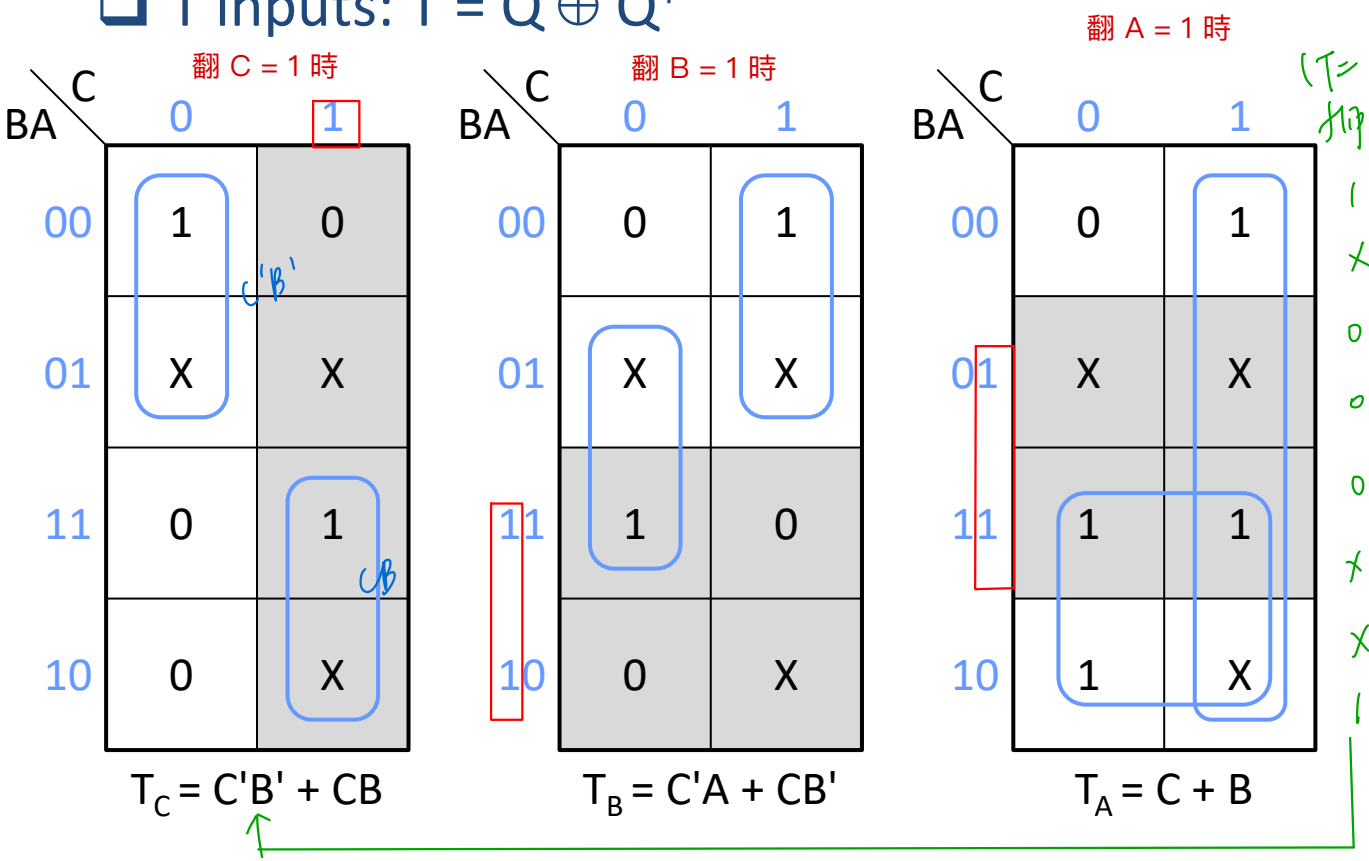
BA \ C	0	1
00	0	1
01	X	X
11	0	0
10	1	X

A^+

C	B	A	C^+	B^+	A^+
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

Implementation: T Flip-Flops (1/2)

- T inputs: $T = Q \oplus Q^+$

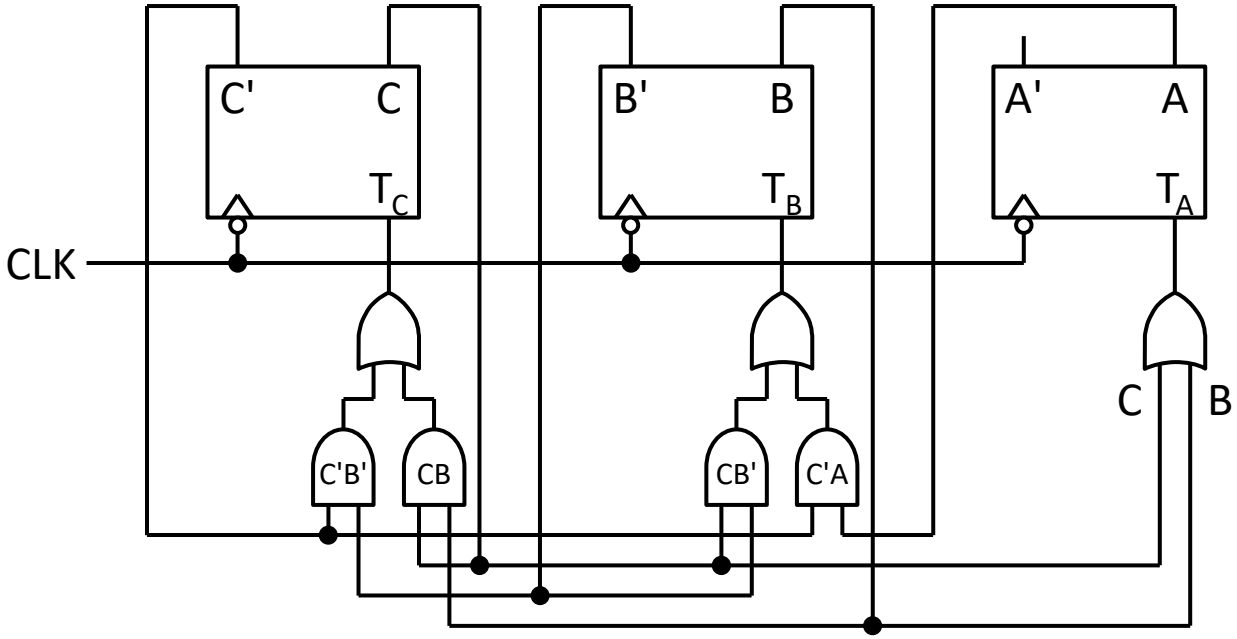


C	B	A	C^+	B^+	A^+
0	0	0	1	0	0
0	0	1	-	-	-
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1	-	-	-
1	1	0	-	-	-
1	1	1	0	1	0

灰底的地方：值改變
白底的地方：值不變

>> T 決定的不是最後的值，而是要不要翻轉

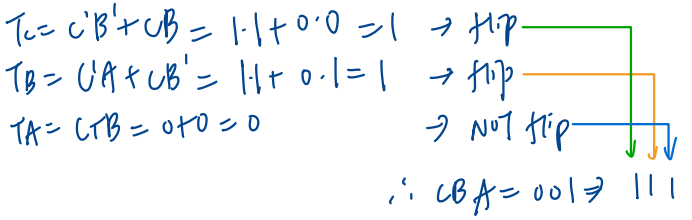
Implementation: T Flip-Flops (2/2)

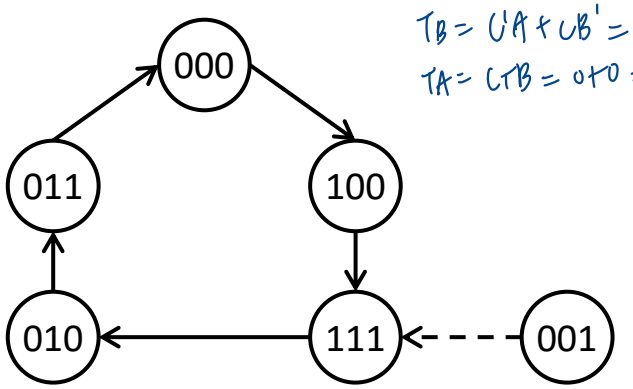


Don't Care States

如果 initial state 是 don’t care

- If flip-flops are initially set to CBA = 001
  - Tracking signals through the network shows that $T_C = T_B = 1$ and $T_A = 0$, so the state changes to 111



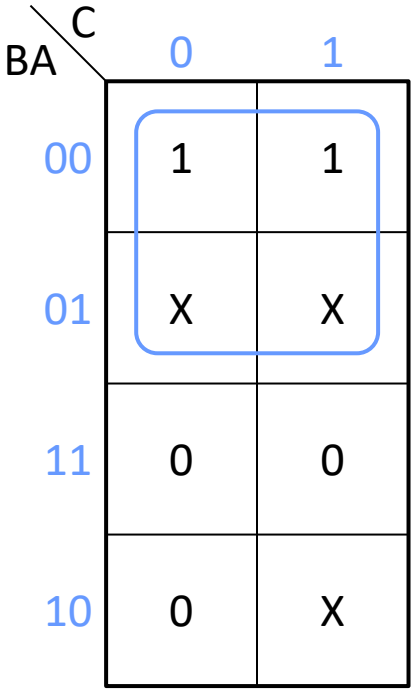


- When the power is turned on, the initial states of all flip-flops are unpredictable!! 在設計時要考慮這些 don’t care state 會不會回到 cycle 中
  - Don't care states should be checked to make sure that they eventually lead into the main counting sequence
  - Or use power-up reset 強迫回到這五個 state 的某個 state

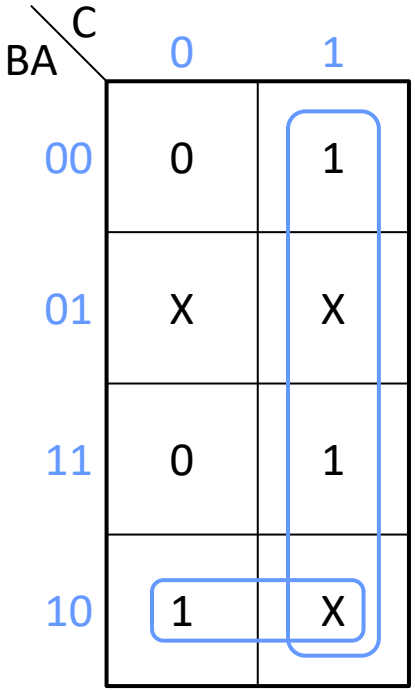
Implementation: D Flip-Flops (1/2)

next state >> 看 D 送什麼值就是什麼值

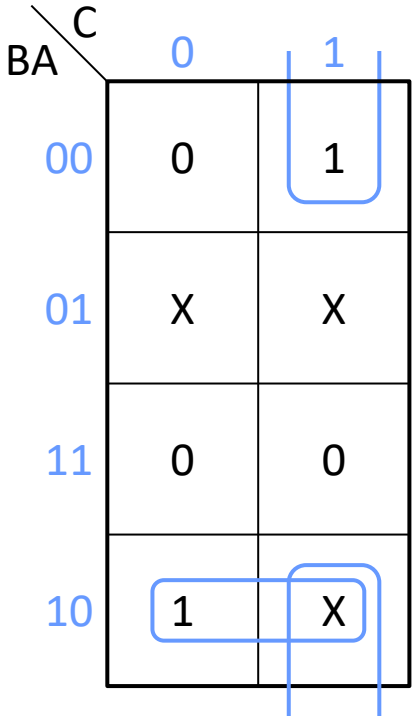
❑ Next states



$D_C = B'$



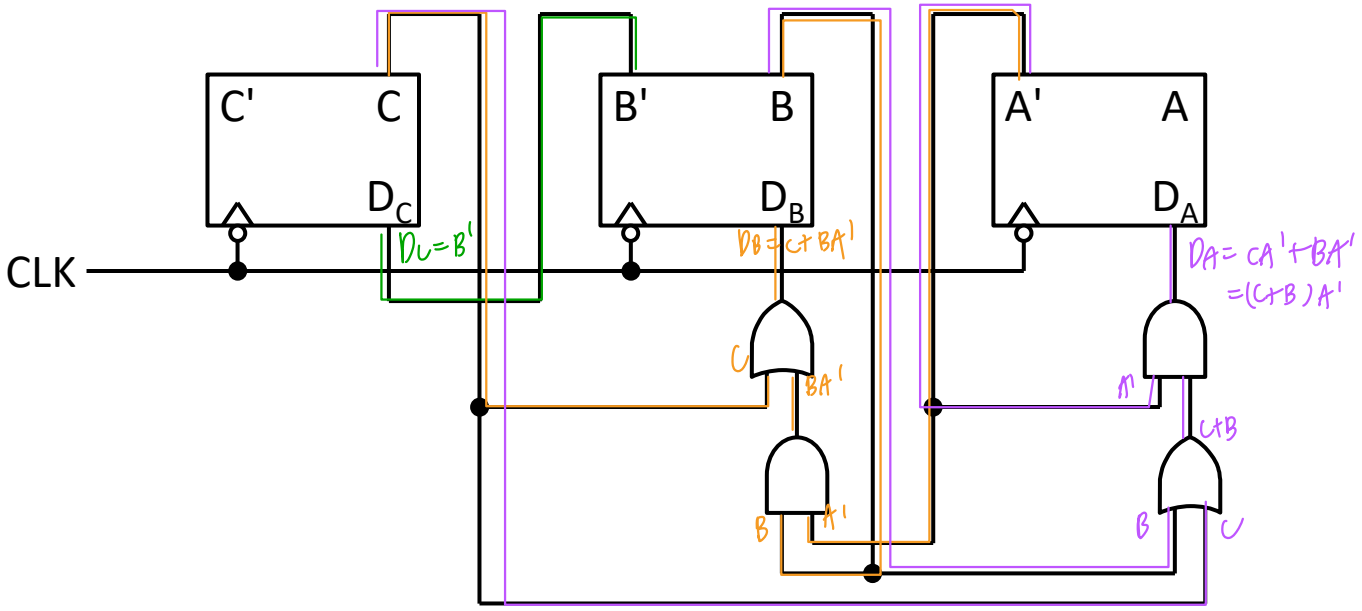
$D_B = C + BA'$



$D_A = CA' + BA'$

C	B	A	C^+	B^+	A^+
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

Implementation: D Flip-Flops (2/2)



Outline

- ❑ Registers and Register Transfers
- ❑ Shift Registers
- ❑ Design of Binary Counters
- ❑ Counters for Other Sequences
- ❑ **Counter Design Using S-R and J-K Flip-Flops**
- ❑ Derivation of Flip-Flop Input Equations

Recap: S-R Flip-Flops

- What is the relation between S, R and Q, Q^+?
  - We do it reversely from Q and Q^+ to S and R

• 因為有兩個 cases 是 not allowed ，所以只有六個 cases

S	R	Q	Q^+	
0	0	0	0	SR = 00 Unchanged
0	0	1	1	
0	1	0	0	SR = 01 Reset to 0
0	1	1	0	
1	0	0	1	SR = 10 Set to 1
1	0	1	1	
1	1	0	X	SR = 11 Inputs Not Allowed
1	1	1	X	

Q	Q^+	S	R
0	0	0 0	0 1
0	1	1	0
1	0	0	1
1	1	0 1	0 0

Q	Q^+	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

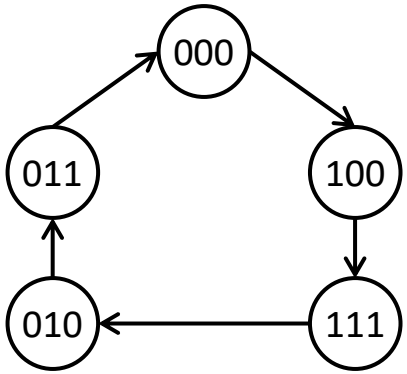
原本的想法：知道 SR ，去決定 Q^+

這邊變成假如知道 Q >> Q^+，問 SR = ?

Excitation Table

Using S-R Flip-Flops (1/2)

- Derive S-R flip-flop inputs from the excitation table



Q	Q^+	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

excitation table

C 從 0 變成 C^+ = 1

>> 查 excitation table，QQ^+ = 01，因此 SR = 10，所以這邊的 S_C = 1, S_R = 0

C	B	A	C^+	B^+	A^+	S_C	R_C	S_B	R_B	S_A	R_A
0	0	0	1	0	0	1	0	0	X	0	X
0	0	1	-	-	-	X	X	X	X	X	X
0	1	0	0	1	1	0	X	X	0	1	0
0	1	1	0	0	0	0	X	0	1	0	1
1	0	0	1	1	1	X	0	1	0	1	0
1	0	1	-	-	-	X	X	X	X	X	X
1	1	0	-	-	-	X	X	X	X	X	X
1	1	1	0	1	0	0	1	X	0	0	1

By Karnaugh maps

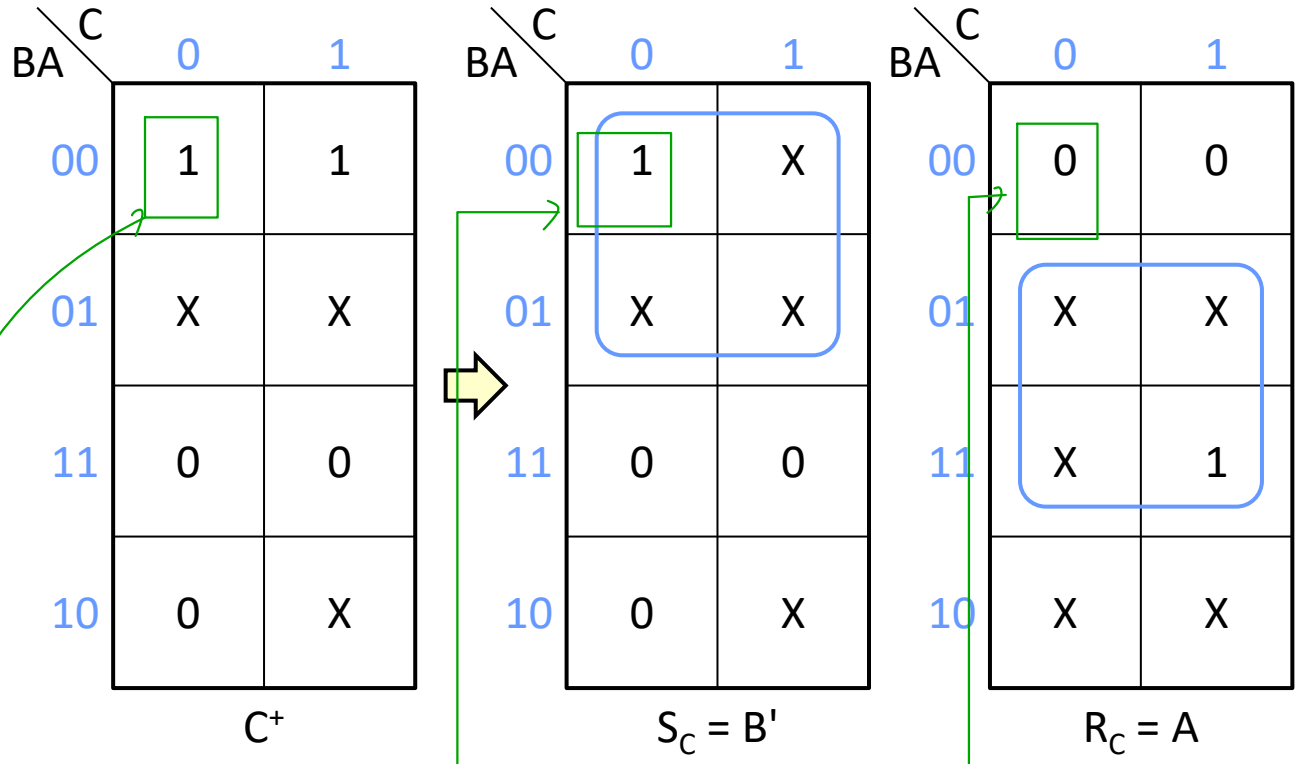
$S_C = B'$, $R_C = A$, $S_B = C$, $R_B = C'A$, $S_A = CA' + BA'$, $R_A = A$

Using S-R Flip-Flops (2/2)

- truth table 和 K–map 其實是同一件事情

❑ Alternative: derive S-R flip-flop inputs with K-maps (faster?)

Q	Q^+	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0



此格代表 C^+ = 1，也就是：CC^+ = 01
>> 查 excitation table 發現 QQ^+ = 01 對應到 SR = 10

Recap: J-K Flip-Flops

- What is the relation between J, K and Q, Q^+?
  - We do it reversely from Q and Q^+ to J and K

如果 Q 要從 0 變成 1，代表 J 必為 1（無論是要用 set to 1 還是 toggle 的方式來變，J 都是 1）

J	K	Q	Q^+	
0	0	0	0	Unchanged
0	0	1	1	
0	1	0	0	Reset to 0
0	1	1	0	
1	0	0	1	Set to 1
1	0	1	1	
1	1	0	1	Toggle（好玩）
1	1	1	0	

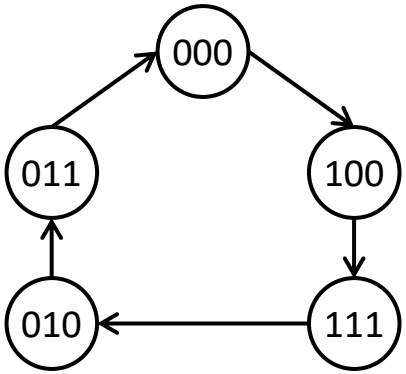
Q	Q^+	J	K
0	0	0	0
		0	1
0	1	1	0
		1	1
1	0	0	1
		1	1
1	1	0	0
		1	0

Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

如果 Q, Q^+ 是什麼值，J, K 是什麼值

Using J-K Flip-Flops

- Derive J-K flip-flop inputs from the excitation table



Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

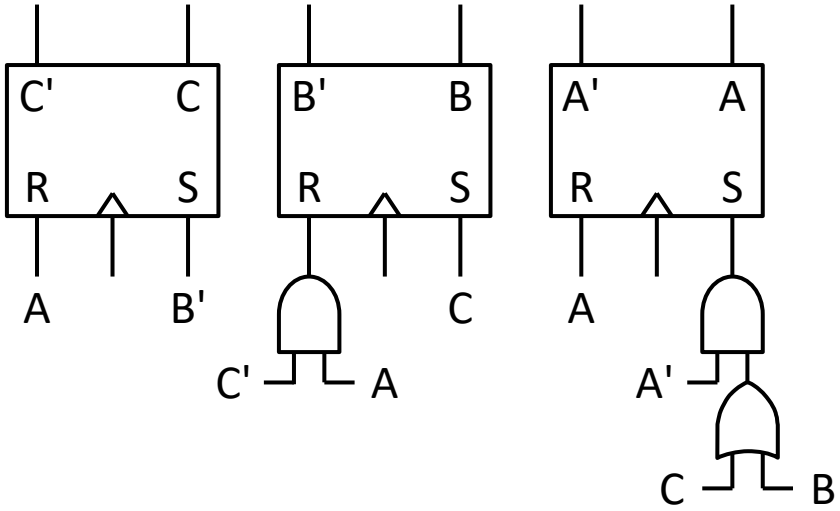
C	B	A	C^+	B^+	A^+	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	1	0	0	1	X	0	X	0	X
0	0	1	-	-	-	X	X	X	X	X	X
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	0	0	0	0	X	X	1	X	1
1	0	0	1	1	1	X	0	1	X	1	X
1	0	1	-	-	-	X	X	X	X	X	X
1	1	0	-	-	-	X	X	X	X	X	X
1	1	1	0	1	0	X	1	X	0	X	1

By Karnaugh maps

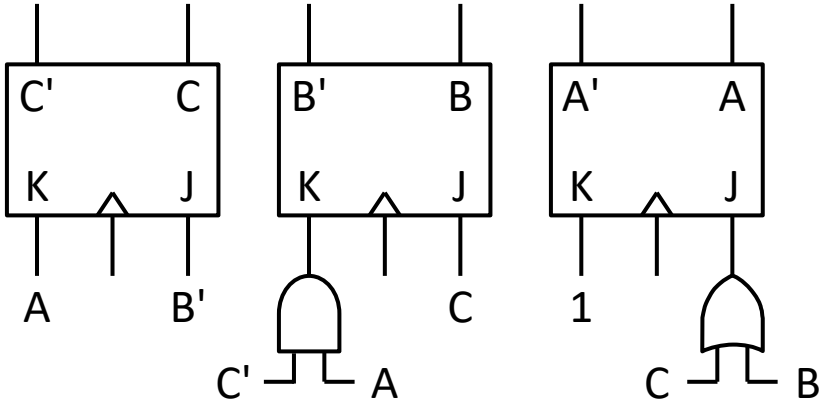
$J_C = B'$, $K_C = A$, $J_B = C$, $K_B = C'A$, $J_A = C + B$, $K_A = 1$

Implementation

- S-R flip-flops



- J-K flip-flops



Outline

- ❑ Registers and Register Transfers
- ❑ Shift Registers
- ❑ Design of Binary Counters
- ❑ Counters for Other Sequences
- ❑ Counter Design Using S-R and J-K Flip-Flops
- ❑ **Derivation of Flip-Flop Input Equations**

Derivation of Flip-Flop Input Equations

- Determine the flip flop input equations from the **next-state equations** using K-maps
  - Always copy X's from next state maps onto input maps first

open book 考試記得必定要帶這個表！！！

Type of FF	Input	Q = 0		Q = 1		Rules for forming input map from next state map	
		$Q^+ = 0$	$Q^+ = 1$	$Q^+ = 0$	$Q^+ = 1$	Q = 0 Half of Map	Q = 1 Half of Map
D	D	0	1	0	1	No change	No change
T	T	0	1	1	0	No change	Complement
S-R	S	0	1	0	X	No change	Replace 1's with X's
	R	X	0	1	0	Replace 0's with X's Replace 1's with 0'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

D 決定 Q^+

T 決定 flip

兩邊 K-map 都不動

SR, JK 不確定的話可以回去看 excitation table

(excitation table)

Important Tables

Q	Q^+	D
0	0	0
0	1	1
1	0	0
1	1	1

D Flip-Flop

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

T Flip-Flop

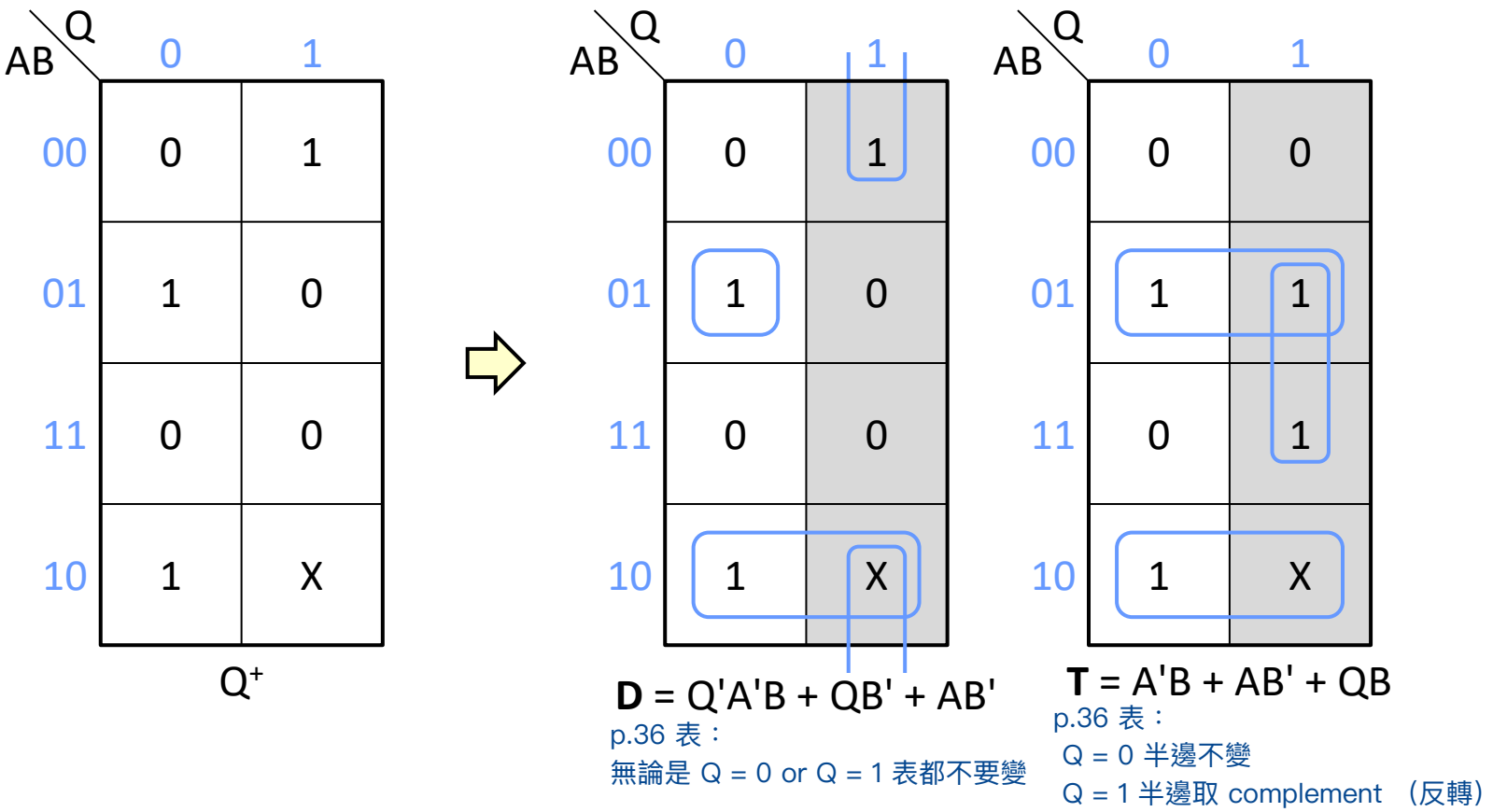
Q	Q^+	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

S-R Flip-Flop

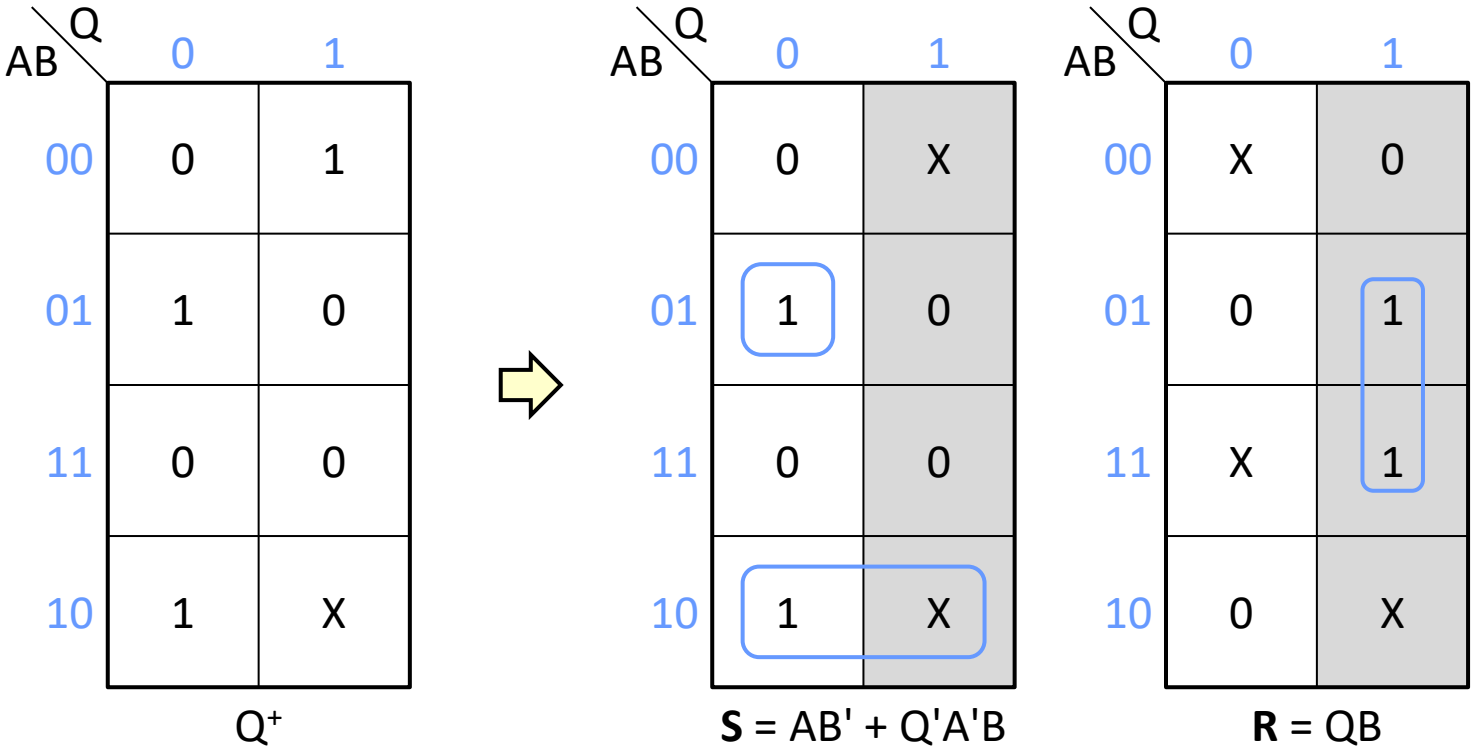
Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

J-K Flip-Flop

3-Variable Example (1/3)



3-Variable Example (2/3)

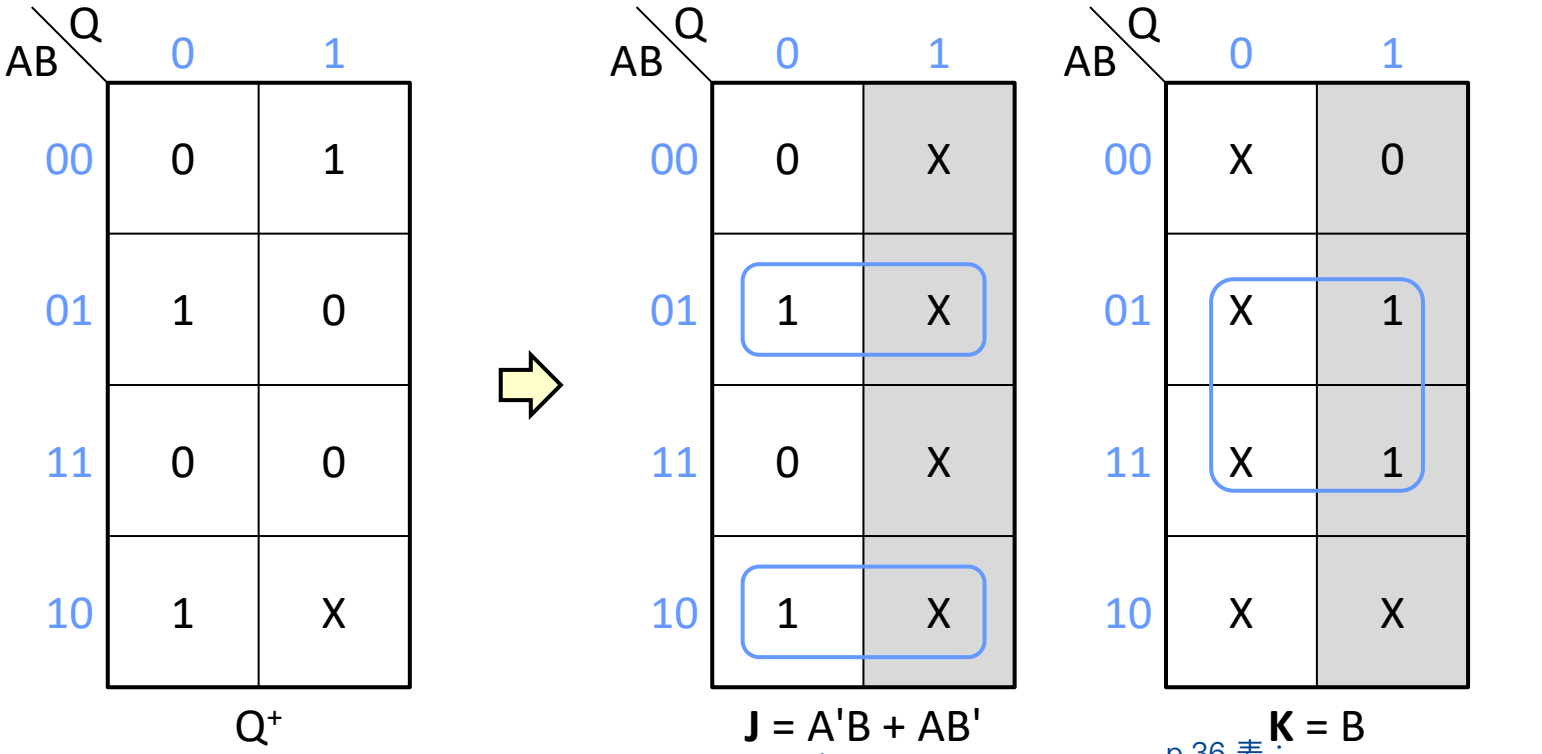


- 用 SR

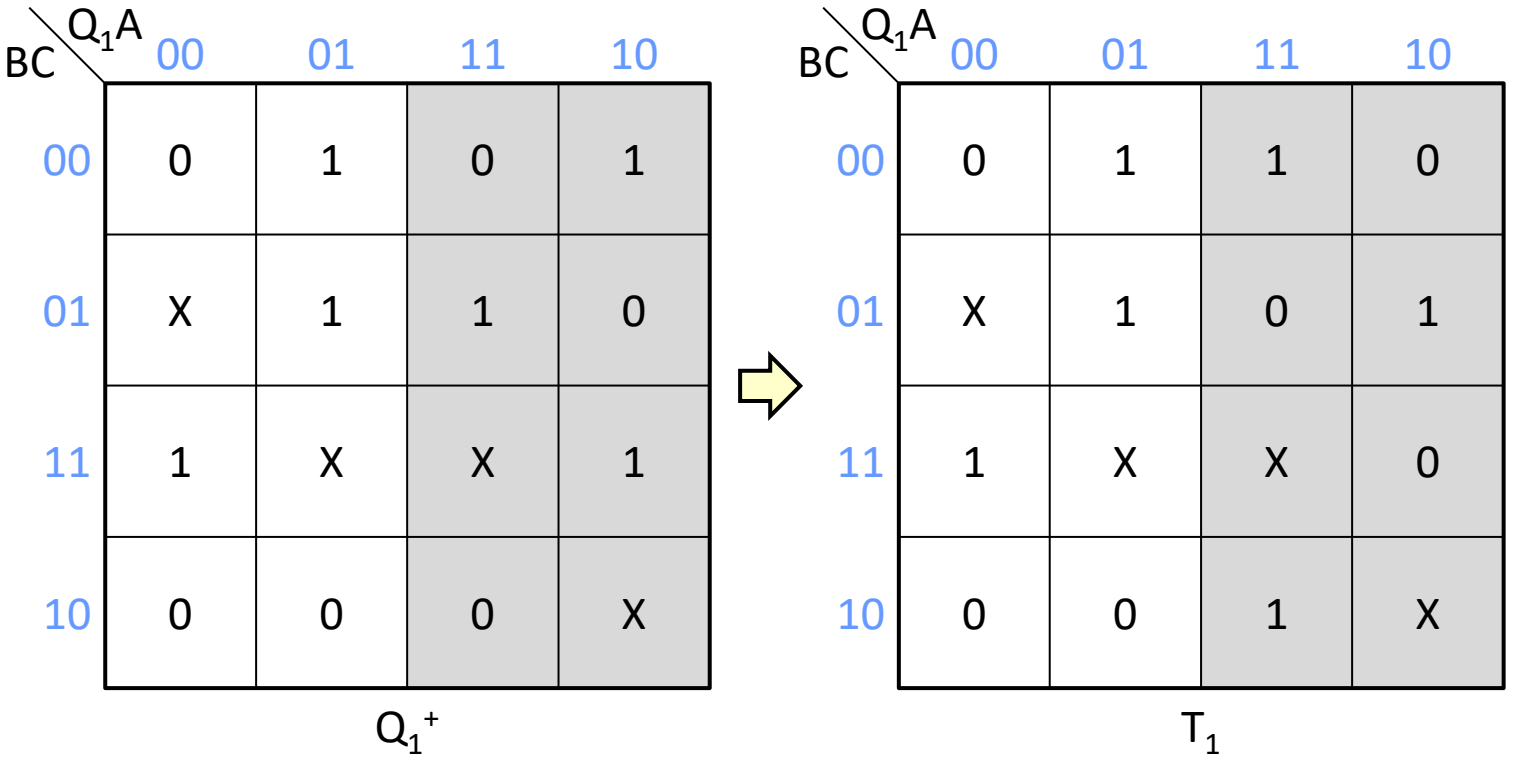
>> 要兩個 K-map（S, R 各一個）

>> 原本用 D flip-flop 或 T flip-flop 都只要一個）

3-Variable Example (3/3)



4-Variable Example (1/3)



灰底處反轉（don't care 仍為 don't care）

4-Variable Ex. (2/3)

p.36 表：
SR flip-flop 的 R 在 Q_2 = 0 時不變

Q2=0

CQ_2 \ AB	00	01	11	10
00	1	X	1	0
01	0	0	X	1
11	1	0	X	1
10	X	0	0	1

Q_2^+

p.36 表：
SR flip-flop 的 R 在 Q_2 = 1 時 1 變 x

Q2=1

CQ_2 \ AB	00	01	11	10
00	1	X	1	0
01	0	0	X	X
11	X	0	X	X
10	X	0	0	1

S_2

CQ_2 \ AB	00	01	11	10
00	0	X	0	X
01	1	1	X	0
11	0	1	X	0
10	X	X	X	0

R_2

4-Variable Ex. (3/3)

Q_3 = 0
>> J no change

(故意換位置將 Q_3 擺在 C 前面)

Q_3 = 1
>> J 用 x 填滿

Q_3C \ AB	00	01	11	10
00	0	0	1	X
01	0	1	X	1
11	X	X	0	0
10	1	1	1	0

Q_3^+

Q_3C \ AB	00	01	11	10
00	0	0	1	X
01	0	1	X	1
11	X	X	X	X
10	X	X	X	X

J_3

Q_3C \ AB	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	X	X	1	1
10	0	0	0	1

K_3

Q&A