

## <<2 進制轉 16 進制>>

$1010110110.110101_{(2)}$

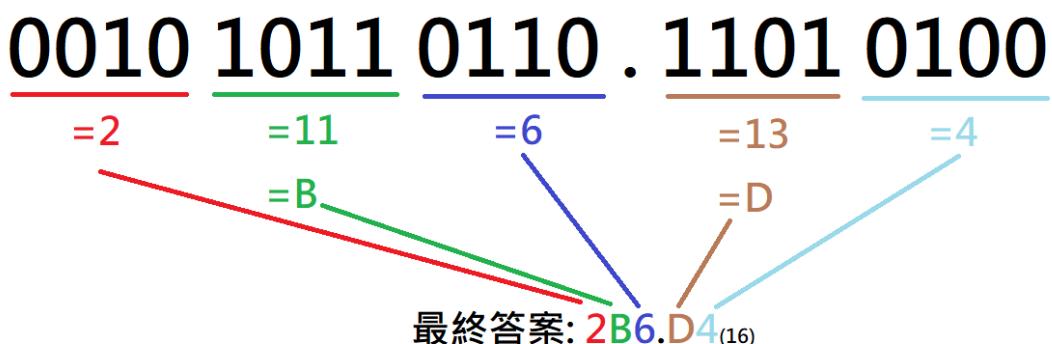
步驟 1:由小數點作為起點，將每 4bit 分成一組。

$10\textcolor{red}{10110110}.\textcolor{blue}{110101}$

步驟 2:如果組別不足 4 個位，則需要補上"0"

$0010\textcolor{magenta}{1011}\textcolor{red}{0110}.\textcolor{cyan}{1101}\textcolor{red}{0100}$

步驟 3:以每 4bit 黎計



## <<任何進制轉 10 進制>>



$$\sum_{x=-\infty}^{\infty} d_x = \sum_{x=-\infty}^{\infty} d \times n^x$$

$d_x$  = 第  $x$  位置的數值，例如上圖  $d_0$  是 1。

$n$  = 進制值，例如上圖是 2 進制， $n=2$ 。

$$\blacksquare (2409.87)_{10} = 2 \times 10^3 + 4 \times 10^2 + 9 + 8 \times 10^{-1} + 7 \times 10^{-2}$$

$$\blacksquare (1101.1001)_2 = 2^3 + 2^2 + 2^0 + 2^{-1} + 2^{-4} = 13.5625$$

$$\blacksquare (703.64)_8 = 7 \times 8^2 + 3 + 6 \times 8^{-1} + 4 \times 8^{-2} = 451.8125$$

$$\blacksquare (A1F.8)_{16} = 10 \times 16^2 + 16 + 15 + 8 \times 16^{-1} = 2591.5$$

$$\blacksquare (423.1)_5 = 4 \times 5^2 + 2 \times 5 + 3 + 5^{-1} = 113.2$$

$$\blacksquare (263.5)_6 \quad \text{Digit 6 is NOT allowed in radix 6}$$

## <<十進制整數轉二進制整數(短除法)>>

$2\boxed{100}.....0$	↑
$2\boxed{50}.....0$	
$2\boxed{25}.....1$	
$2\boxed{12}.....0$	
$2\boxed{6}.....0$	
$2\boxed{3}.....1$	
1	
	↓
$2\boxed{100}.....0$	
$2\boxed{50}.....0$	
$2\boxed{25}.....1$	
$2\boxed{12}.....0$	
$2\boxed{6}.....0$	
$2\boxed{3}.....1$	
1	

=110 0100  
=0110 0100

$2\lfloor 123 \dots \dots 1$	$2\lfloor 123 \dots \dots 1$
$2\lfloor 61 \dots \dots 1$	$2\lfloor 61 \dots \dots 1$
$2\lfloor 30 \dots \dots 0$	$2\lfloor 30 \dots \dots 0$
$2\lfloor 15 \dots \dots 1$	$2\lfloor 15 \dots \dots 1$
$2\lfloor 7 \dots \dots 1$	$2\lfloor 7 \dots \dots 1$
$2\lfloor 3 \dots \dots 1$	$2\lfloor 3 \dots \dots 1$
1	$2\lfloor 3 \dots \dots 1$

=111 1011  
=0111 1011

<<十進制小數轉換為二進制小數(無限乘法)>>

$$(0.8125)_{10}$$

$$\begin{aligned} &= 2 \times (0.8125)_{10} = 1.625 \cdots \text{取出整數，把小數再繼續計算：1} \\ &= 2 \times (0.625)_{10} = 1.25 \cdots \text{取出整數，把小數再繼續計算：1} \\ &= 2 \times (0.25)_{10} = 0.5 \cdots \text{取出整數，把小數再繼續計算：0} \\ &= 2 \times (0.5)_{10} = 1.0 \cdots \text{取出整數，把小數再繼續計算：1} \end{aligned}$$

最後把取出的 0 與 1 以順序排，答案為  $(0.1101)_2$ ，因此也就等於  $(0.8125)_{10}$

\*\*\*如果數值是又有整數又有小數，其實也是分開整數以整數方式去計，小數以小數方式去計

<<十進制小數轉換為 8 進制小數(無限乘法)>>

$$(0.8125)_{10}$$

$$\begin{aligned} &= 8 \times (0.8125)_{10} = 6.5 \cdots \text{取出整數，把小數再繼續計算：6} \\ &= 8 \times (0.5)_{10} = 4.0 \cdots \text{取出整數，把小數再繼續計算：4} \end{aligned}$$

最後把取出數以順序排，答案為  $(0.64)_8$ ，因此也就等於  $(0.8125)_{10}$

$$(0.8665)_{10}$$

$$\begin{aligned} &= 8 \times (0.8665)_{10} = 6.932 \cdots \text{取出整數，把小數再繼續計算：6} \\ &= 8 \times (0.932)_{10} = 7.456 \cdots \text{取出整數，把小數再繼續計算：7} \\ &= 8 \times (0.456)_{10} = 3.648 \cdots \text{取出整數，把小數再繼續計算：3} \\ &= 8 \times (0.648)_{10} = 5.184 \cdots \text{取出整數，把小數再繼續計算：5} \\ &= 8 \times (0.184)_{10} = 1.472 \cdots \text{取出整數，把小數再繼續計算：1} \\ &= 8 \times (0.472)_{10} = 3.776 \cdots \text{取出整數，把小數再繼續計算：3} \\ &= 8 \times (0.776)_{10} = 6.208 \cdots \text{取出整數，把小數再繼續計算：6} \\ &= 8 \times (0.208)_{10} = 1.664 \cdots \text{取出整數，把小數再繼續計算：1} \\ &= 8 \times (0.664)_{10} = 5.312 \cdots \text{取出整數，把小數再繼續計算：5} \\ &= 8 \times (0.312)_{10} = 2.496 \cdots \text{取出整數，把小數再繼續計算：2} \\ &= 8 \times (0.496)_{10} = 3.968 \cdots \text{取出整數，把小數再繼續計算：3} \\ &= 8 \times (0.968)_{10} = 7.744 \cdots \text{取出整數，把小數再繼續計算：7} \\ &= 8 \times (0.744)_{10} = 5.952 \cdots \text{取出整數，把小數再繼續計算：5} \\ &= 8 \times (0.952)_{10} = 7.616 \cdots \text{取出整數，把小數再繼續計算：7} \\ &= 8 \times (0.616)_{10} = 4.928 \cdots \text{取出整數，把小數再繼續計算：4} \\ &= 8 \times (0.928)_{10} = 7.424 \cdots \text{取出整數，把小數再繼續計算：7} \\ &= 8 \times (0.424)_{10} = 3.392 \cdots \text{取出整數，把小數再繼續計算：3} \end{aligned}$$

:

:

約為  $0.67351361523757473311$

## <<十進制小數轉任何進制小數(公式)>>

簡介:其實只是從無限乘法來推演出的公式而已，本質上仍是無限乘法。

1. 十進制轉  $\beta$  進制時，把第  $n$  位小數以  $\beta$  進制的次方顯示，本質上仍然 10 進值

$$(y_n)_{10} = [\beta \times \{M_{n-1}\}] \times \beta^{-(n)}$$

2. 十進制轉  $\beta$  進制時，所有小數以  $\beta$  進制的次方及加總方式顯示，本質上仍然 10 進數值

$$f(x)_{10} = \sum_{n=1}^{\infty} (y_n)_{10} = (y_1)_{10} + (y_2)_{10} + \cdots + (y_n)_{10} = \sum_{n=1}^{\infty} [\beta \times \{M_{n-1}\}] \times \beta^{-(n)}$$

3. 十進制轉  $\beta$  進制時，只顯示該  $\beta$  進制的的值，本質上已轉為  $\beta$  進制數值

$$(y_n)_{\beta} = [\beta \times \{M_{n-1}\}]$$

$\beta$	需要的進制值
{}	高斯記號，只取小數，不取整數
[]	高斯記號，只取整數，不取小數
$M_n$	$M$ 代表第 $n$ 次的題目數值，也即第 $n$ 次答案值，例如第 0 次是 0.8665，第 1 次就是 6.932，第 2 次就是 7.456
$(y_n)_{\beta}$	$\beta$ 進制的第 $n$ 位小數值， $y$ 是該位置的數值

$$(y_1)_8 = [8 * \{0.8665\}] * 8^{-1}$$

$$(y_1)_8 = [8 * 0.8665] * 8^{-1}$$

$$(y_1)_8 = [6.932] * 8^{-1}$$

$$(y_1)_8 = [6] * 8^{-1}, M_1 = 6.932$$

$$(y_2)_8 = [8 * \{6.932\}] * 8^{-2}$$

$$(y_2)_8 = [8 * 6.932] * 8^{-2}$$

$$(y_2)_8 = [7.456] * 8^{-2}$$

$$(y_2)_8 = [7] * 8^{-2}, M_2 = 7.456$$

延伸:

因為:

$$(M_n) = \beta \times \{M_{n-1}\}$$

$$(M_1) = \beta \times \{M_0\}$$

$$(M_2) = \beta \times \{M_1\}$$

$$(M_3) = \beta \times \{M_2\}$$

所以:

$$(M_3) = \beta \times \{\beta \times \{\beta \times \{M_0\}\}\}$$

高斯記號抽取小數:

$$(M_n) = \beta \times \{\beta \times \{\beta \times \cdots \times \{M_0\}\}\} = \{\beta^n \times M_0\}$$

反向推論:

$$\sum_{n=1}^{\infty} (y_n)_{\beta} = \sum_{n=1}^{\infty} [\beta \times \{M_{n-1}\}] \times \beta^{-(n)} = \sum_{n=1}^{\infty} [\beta \times \{\beta^{n-1} \times M_0\}] \times \beta^{-(n)}$$

$\beta$  進制第  $n$  位小數值:

$$(y_n)_{\beta} = [\beta \times \{\beta^{n-1} \times M_0\}]$$

優化後的公式:

4. 十進制轉  $\beta$  進制時，把第  $n$  位小數以  $\beta$  進制的次方顯示，本質上仍然 10 進值

$$(y_n)_{10} = [\beta \times \{\beta^{n-1} \times M_0\}] \times \beta^{-(n)}$$

5. 十進制轉  $\beta$  進制時，所有小數以  $\beta$  進制的次方及加總方式顯示，本質上仍然 10 進數值

$$f(x)_{10} = \sum_{n=1}^{\infty} (y_n)_{10} = (y_1)_{10} + \cdots + (y_n)_{10} = \sum_{n=1}^{\infty} [\beta \times \{\beta^{n-1} \times M_0\}] \times \beta^{-(n)}$$

6. 十進制轉  $\beta$  進制時，只顯示該  $\beta$  進制的的值，本質上已轉為  $\beta$  進制數值

$$(y_n)_{\beta} = [\beta \times \{\beta^{n-1} \times M_0\}]$$

例子:

$$A = (0.24)_{10}$$

$$(y_1)_2 = [2 \times \{2^0 \times 0.24\}] = [2 \times \{0.24\}] = [2 \times 0.24] = [0.48] = 0$$

$$(y_2)_2 = [2 \times \{2^1 \times 0.24\}] = [2 \times \{0.48\}] = [2 \times 0.48] = [0.96] = 0$$

$$(y_3)_2 = [2 \times \{2^2 \times 0.24\}] = [2 \times \{0.96\}] = [2 \times 0.96] = [1.84] = 1$$

$$(y_4)_2 = [2 \times \{2^3 \times 0.24\}] = [2 \times \{1.92\}] = [2 \times 0.92] = [1.68] = 1$$

...(餘下繼續)

驗證:  $(0.24)_{10} = (0.0011110)_2$  (con to 7 fig sig)

## <<二進制負數值轉換>>

在二進制中表示負數的方式有 3 種，第一是 Signed Magnitude，第二種是 1's complement，第三種是 2's complement

正數				負數			
十進位	二進位 ( 4 bit 為例 )			十進位	二進位 ( 4 bit 為例 )		
	Signed Magnitude	1's Complement	2's Complement		Signed Magnitude	1's Complement	2's Complement
0	0000	0000	0000	0	1000	1111	0000
1	0001	0001	0001	-1	1001	1110	1111
2	0010	0010	0010	-2	1010	1101	1110
3	0011	0011	0011	-3	1011	1100	1101
4	0100	0100	0100	-4	1100	1011	1100
5	0101	0101	0101	-5	1101	1010	1011
6	0110	0110	0110	-6	1110	1001	1010
7	0111	0111	0111	-7	1111	1000	1001
8	X	X	X	-8	X	X	1000

<https://medium.com/@tiffanychen1118>

### 1. Signed Magnitude

Signed Magnitude 其實就是在第一個 bit 以 0 表示為正，以 1 表示為負，但這樣會浪費掉第一個 bit，而且會令到可表示的最大值減少一半，例如 0~255 會變成 -128~127，

### 2. 1's Complement

1's Complement 用法是與原數顛倒，這種方式可以用哂全部 bit，例如 0~255 可以用哂做 -255~255。但要注意「0」會用「0000」和「1111」表示，這兩者都是 0，因此會造成一些混淆。為解決混淆，因此 2's complement 就出現了。另外注意與原數顛倒意思是根據原有的 bit 數作顛倒，即係 1001，顛倒為 0110，不應自行補上“0000 1001”。

### 3. 2's complement

計算方法是先進行 1's Complement 再加 1，而這個方法可以令到「0」的表示方法只有「0000」一種，不再有兩種表達方法，避免混淆。而且可表示的數值多返 1 個，即 -256~255。

### <<Complement 加減法>>

使用 complement 的處是可以方便地使用減法，例如  $z = A - B$ ，可使用  $A + (-B)$ ，這樣就可以以加法來完成減法。

- i. 已知  $A-B$  得出結果會是正數情況下：

重點一：2 個值要對齊大家的 bit 數量

重點二：如果題目本身在最前面有 0，則需保留前面的 0 值。

重點三：如果出現溢位則丟棄

例子一：

例如  $A=(0111010101111)_2$  有 13 位 bit， $B=(1010011011)_2$  有 10 個位 bit，因此 B 要補至與 A 的 bit 數相同，即  $B=(0001010011011)_2$ 。 $A=(0\ 1110\ 1010\ 1111)_2 + B=(0\ 0010\ 1001\ 1011)_2$  後等於  $(10\ 1100\ 0001\ 0100)_2$ ，多出第 14 個位需要丟棄。即  $(0\ 1100\ 0001\ 0100)_2$

例子二：

$A=(101\ 0100)_2 = (84)_{10}$ ， $B=(100\ 0011)_2 = (67)_{10}$

如果要  $A-B$ ，可以先將 B 轉倒數，成為 1's complement， $-B=(011\ 1100)_2 = -67$ ，再轉 2's complement， $-B = (011\ 1101)_2 = -67$ ，之後相加  $A+(-B)=(1001\ 0001)$ ，但要注意，這裡只用返相對應 bit 數，即 7 個 bit，因此多出的一個 bit 需要移除，所以相加  $A+(-B)=(001\ 0001) = (17)_{10}$ 。

$$\begin{array}{r} x & \quad 1010100 \\ 2's \text{ complement of } y & \quad + \quad 0111101 \\ \text{Discard carry} \rightarrow & \quad \boxed{1}0010001 = (17)_{10} \end{array}$$

ii. 已知 A-B 得出結果會是負數情況下:

重點一: 2 個值要對齊大家的 bit 數量

重點二: 如果題目本身在最前面有 0，則需保留前面的 0 值。

重點三: 不會出現溢位

重點四: 答案是負數，需要逆向 Complement 先可以還原成正數值

如果 B-A，則為 B+(-A)，將 A 顛倒=(010 1011)<sub>2</sub> 即係 1's complement，之後再加"1" 成為 2's complement = (010 1100)<sub>2</sub>，結果為(110 1111)<sub>2</sub>，由於這個結果是負數值，因此需要將他轉返為正值才可以知道正數部份是多少，可想像為逆向的 2's complement，首先是將結果減 1 再顛倒，即(110 1111)<sub>2</sub> 減一，得出(110 1110)<sub>2</sub>，再顛倒為(001 0001)=(17)<sub>10</sub>。

$$\begin{array}{r} \text{Y} & \quad 1000011 \\ \text{2's complement of } x & + \quad 0101100 \\ \hline & 1101111 = (-17)_{10} \end{array}$$

反思: 如果要預先知道答案計出黎是正與負，是否沒有 2'complement 的需要? 答案是需要的，因為 2'complement 作用是優化運算過程，而非優化運算結果，因此在事前需要知道 A-B 結果是正還是負值，這又該如何判斷? 實際上好簡單，只需由 MSB 開始進行比較。

<<2 進制相加>>

$$\begin{array}{r} 10101 & 21 \\ + 10011 & 19 \\ \hline 101000 & 40 \end{array}$$

<<2 進制相乘>>

$$4.75 = (100.11)_2 \text{ and } 3.625 = (11.101)_2$$

$$\begin{array}{r} 100.110 \\ \times 11.101 \\ \hline 100110 \\ 000000 \text{ X} \\ 100110 \text{ X X} \\ 100110 \text{ X X X} \\ \hline +) 100110 \text{ X X X X} \\ \hline 10001.001110 (= 17.21875_{10}) \end{array}$$

共6位小數

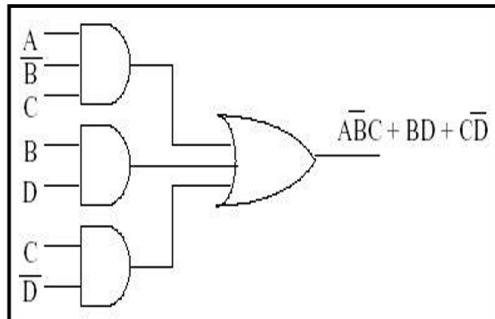
在第6位小數

<<2 進制相除>>

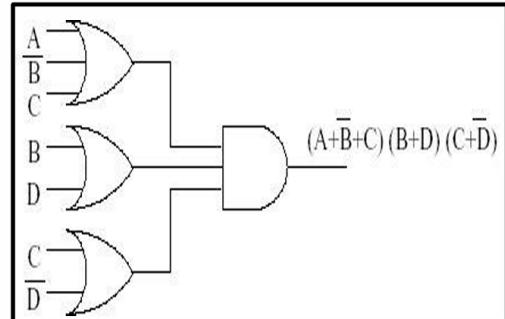
Divisor	1 1 0 1	0 0 0 0 1 0 1 0 1	Quotient
		<u>1 0 0 0 1 0 0 1 0</u>	Dividend
		- 1 1 0 1	1 1 1 1 + 1 - 1 1 0 1 = 0 0 1 0 + 1 + 1 = 1 0 0
		1 0 0 0 0	
		- 1 1 0 1	
		1 1 1 0	
		- 1 1 0 1	
		1	Remainder

<<SOP 與 POS>>

● Sum-of-Products (SOP)



● Product-of-Sums (POS)



Sum of product (SOP)

$$X = \bar{A}\bar{B}\bar{C} + A\bar{B} + \bar{A}BC$$

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

編寫一個 truth table，取出 X 結果為 1 與 ABC 之間邏輯。

Product of sum (POS)

$$X = (\bar{A}+B+\bar{C})(A+\bar{B})(\bar{A}+B+C)$$

A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

編寫一個 truth table，取出 X 結果為 0 與 ABC 之間邏輯。

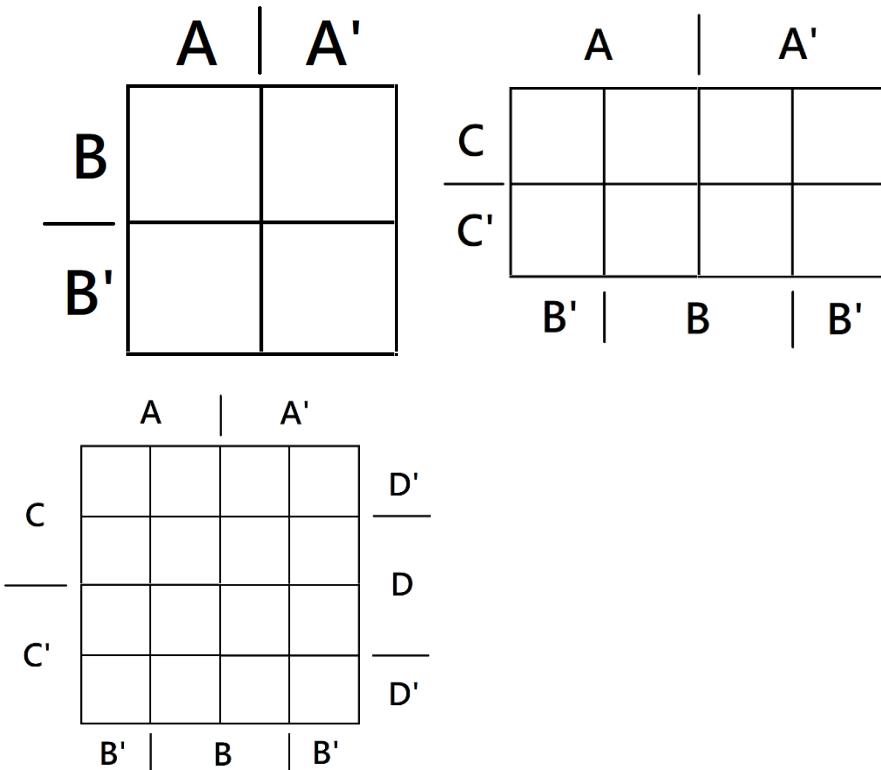
## <<K-Map>>

目的:根據 input 的數據和 output 的 F 數值反推論出這些 input output 的 logic gate 結構, 用在找出 SOP 或 POS。

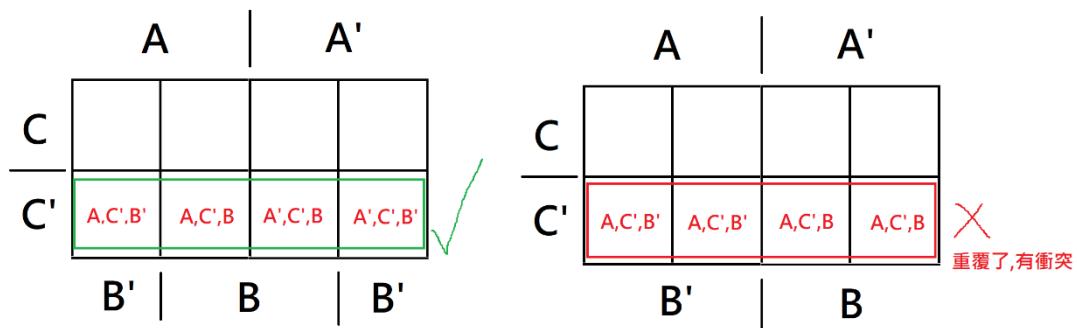
### 1. 定義 K-Map 格數與排位規則:

i. 定義 K-Map 大小, 2 個 input 將有  $2^2$  格數的 k-map, 3 個 input 將有  $2^3$  格數的 K-map。

以下是各 input 數量的排位, 以這種排位方式目的是令每個 item 不會衝突:



如果不以這種形式畫便會出現:



2. 繪製已知的 Truth table 結果，把相應結果填入 K-Map

A	B	C	F(output)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

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

3. 根據規則圈出相連的"1"值，用作找出 SOP

- i. 盡可能以最大的圈來圈出
- ii. 只可以是圓形的圈，不可以例如是十字形相連的"1"
- iii. K-Map 沒有邊界，因此可以是出現半月形與另一邊的半月形相連
- iv. 任何一個"1"也需要圈
- v. 圈出既圈大小只有 1, 2, 4, 8, 16 等等的 size，不可以例如 3 個相連

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

未圈唔



A		A'	
C		1	
C'		1	
B'		B	
B'		B'	

不可以非圓形



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

不是1,2,4,8,16



A		A'	
C		1	
C'		1	
B'		B	
B'		B'	



4. 把單個圈起的位置相乘，把所有圈相加

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

$$F = ACB' + ABCC' + CBAA'$$

$$F = C(A+B) + AB$$

以上就是 SOP 答案，而如果是找 POS，也是使用用樣方式，但就是根據規則圈出相連的“0”值。

可以直接背左佢：

		A		A'				A		A'	
		12	13	5	4	C'			A	A'	
B		14	15	7	6		B		i	ii	C'
		10	11	3	2	C					
B'		8	9	1	0	C'	B'		iii	i	C'
		D'	D	D'	D'		D'	D	D'	D'	

<<Boolean algebra 化簡公式表>>

公式	化簡
A+0	A
A+1	1
A*0	0
A*1	A
A+A	A
A+A'	1
A*A	A
A*A'	0
A''	A
A+AB	A
A+A'B	A+B
(A+B)(A+C)	A+BC
(X+Y)'	X'*Y'
(X+Y+...+Z)'	X'*Y'*...Z'
(X*Y)'	X'+Y'
(X*Y*...*Z)'	X'+Y'+...Z'

<<簡化公式技巧>>

1.無中生有法：

由於( $a+a'$ )是等於"1"，因此在公式中可以無時無刻加入"1"。

而在答問題技巧上，我們有時為左想還原做 truth table 的每個答案，我們將可以加返該項本身有既值，例如：

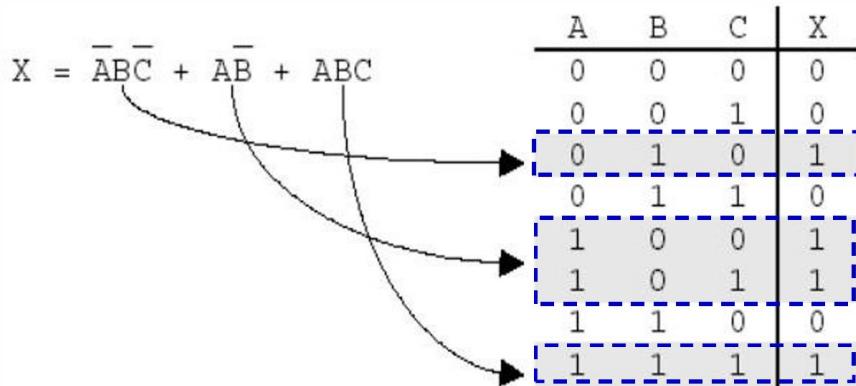
$$\begin{aligned} & \bar{a}\bar{b}\bar{c} + \bar{a}b\bar{c} + a\bar{b} + \bar{b}c \\ &= \bar{a}\bar{b}\bar{c} + \bar{a}b\bar{c} + a\bar{b}(c + \bar{c}) + (a + \bar{a})\bar{b}c \end{aligned}$$

這樣就可以令公式還原出  $f(a,b,c)$  的 8 個可能性中所有會出"1"的值，而非簡化版的公式。

2.如題目限制不能使用 de morgan theorem，可以先使用無中生有來擴大還原所有可能性，而且盡可能不以"抽數出黎"既方式去做。

$$\begin{aligned} & \bar{a}\bar{b}\bar{c} + \bar{a}b\bar{c} + a\bar{b}c + a\bar{b}\bar{c} + a\bar{b}c + \bar{a}\bar{b}c \\ & \bar{a}\bar{b}(\bar{c} + c) + \bar{a}b\bar{c} + a\bar{b}c + a\bar{b}\bar{c} \\ & \bar{a}\bar{b} + \bar{a}b\bar{c} + a\bar{b}c + a\bar{b}\bar{c} \\ & \bar{a}(\bar{b} + b\bar{c}) + a\bar{b}c + a\bar{b}\bar{c} \\ & \bar{a}(\bar{b} + \bar{c}) + a\bar{b}c + a\bar{b}\bar{c} \\ & \bar{a}\bar{b} + \bar{a}\bar{c} + a\bar{b}(c + \bar{c}) \\ & \bar{a}\bar{b} + \bar{a}\bar{c} + a\bar{b} \\ & (\bar{a} + a)\bar{b} + \bar{a}\bar{c} \\ & \bar{b} + \bar{a}\bar{c} \end{aligned}$$

<<Converting SOP to TruthTable>>



#### The Standard SOP Expression

- All variables appear in each product term.
- Each of the product term in the expression is called **minterm**.
- Example:  $f(A,B,C) = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + A\bar{B}C$

Minterm	Minterm Code	Minterm Number
$\bar{A}\bar{B}\bar{C}$	010	$m_2$
$A\bar{B}\bar{C}$	110	$m_6$
$A\bar{B}C$	011	$m_3$

- In compact form (*namely Canonical Form*),  $f(A,B,C)$  may be written as:

$$f(A,B,C) = m_2 + m_3 + m_6$$

$$f(A,B,C) = \sum m(2,3,6)$$

<<Converting POS to Truth Table>>

A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

$X = (\bar{A}+B+\bar{C}) \cdot (A+\bar{B}) \cdot (A+B+C)$

#### ◆ The standard POS Expression

- All variables appear in each product term.
- Each of the sum term in the expression is called a **maxterm**.
- Example:  $f(A,B,C) = (\bar{A}+B+\bar{C}) \cdot (A+\bar{B}) \cdot (\bar{A}+B+C)$

Maxterm	Maxterm Code	Maxterm Number
$\bar{A}+B+\bar{C}$	101	M <sub>5</sub>
$A+\bar{B}+\bar{C}$	001	M <sub>1</sub>
$\bar{A}+B+C$	100	M <sub>4</sub>

- In compact form,  $f(A,B,C)$  may be written as

$$f(A,B,C) = M_1 M_4 M_5$$

$$f(A,B,C) = \prod M(1,4,5)$$

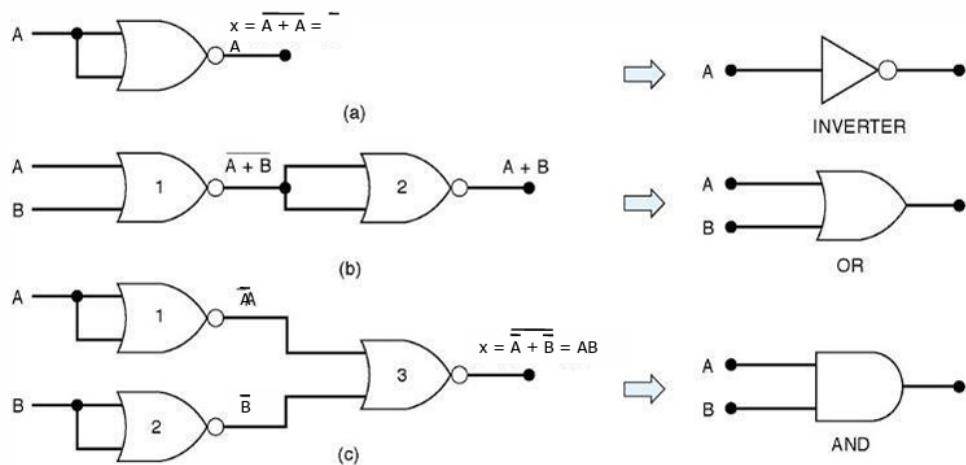

---

## <<Universal Logic Gates 通用邏輯閘>>

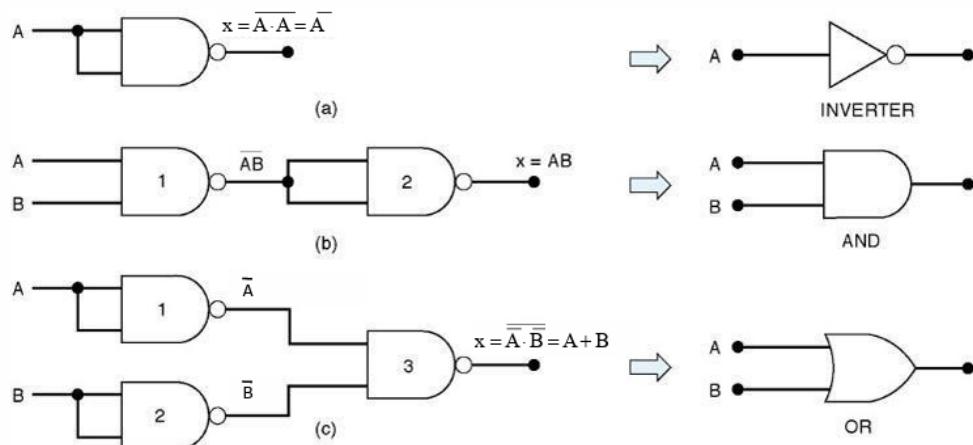
意思：是可以採用同 NOR gate 與 NAND 做到唔不同其他 logic gate 的效果，因為就可以只用一款 IC 就足夠。

由於如果要做唔同 function 而用好多 logic gate，這樣會令到現實上需要買唔同類型既 IC，因此在大規模採購或製造 IC 時會不夠統一而出現例如銷量不同但製造量相同而浪費掉，另外也可能會因為有多個種類 IC 而令到安裝工作人員增加錯誤既機率，因為當統一用同一款 IC，咁就不會因為選擇錯了而造成安裝錯誤。但是也有不好的地方，就是會將電路複雜左及浪費多左電流。

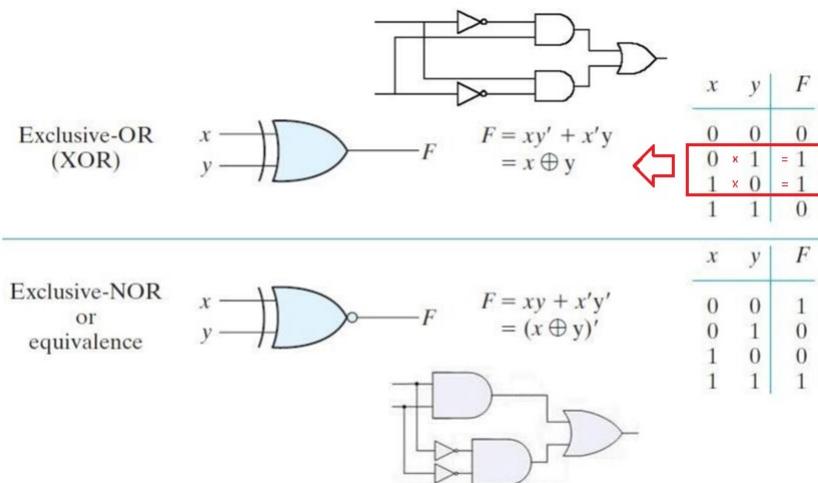
### • NOR Gate



### • NAND Gate



## <<Exclusive gate>>



意思：當 2 者不一樣時才會有 output

### <<帶未知數的 K-Map>>

原理：與 K-map 一樣，但有些結果是未知數(x)，因此在計算時會有不一樣方式。

“x”不像 1 和 0，他是在特定情況下才可以把 x 也圈起來，

規則如下：

- i. 不能單獨把“x”圈起
- ii. 盡可能圈出的“1”連用“x”擴大，就可以把他圈起
- iii. 擴大時，圈出的“x”，這時就可以被當作為“1”

以下是一些可被圈起的例子：

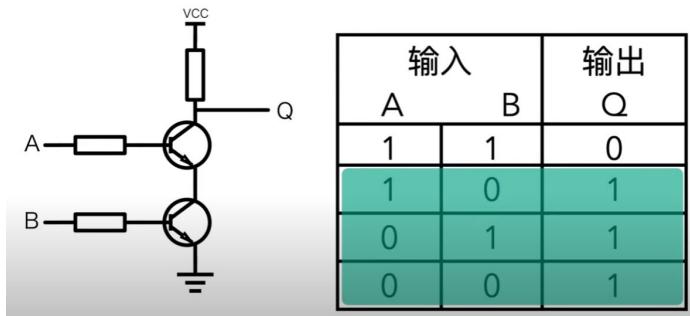
		A	A'	
		C	D'	
		D		
x	x	1	1	
x	x	0	0	
0	x	0	0	
1	x	1	1	
B'	B		B'	

		A	A'	
		C	D'	
		D		
0	1	0	0	
x	1	1	x	
1	x	1	1	
0	x	0	0	
B'	B		B'	

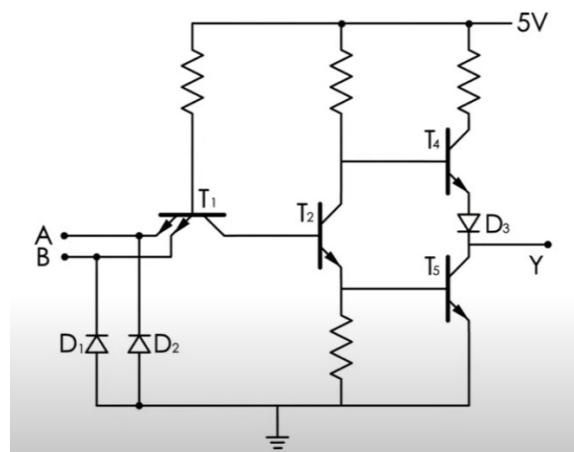
		A	A'	
		C	D'	
		D		
0	1	0	0	
0	x	0	0	
0	1	0	0	
0	0	0	0	
B'	B		B'	

<<傳統 logic gate>>

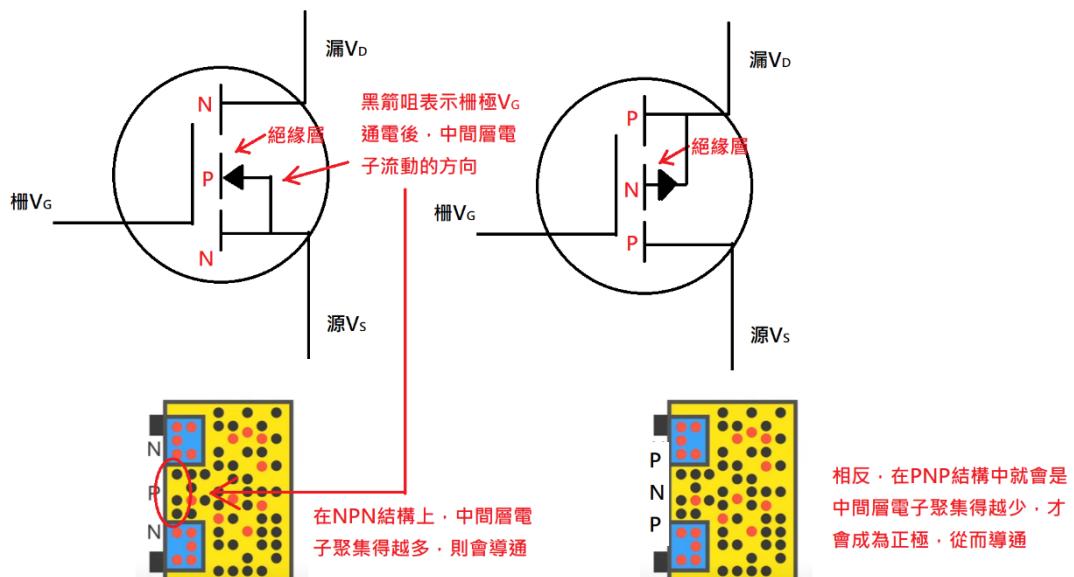
### 1.RTL (Resistor Transistor Logic)



### 2. TTL(Transistor-Transistor Logic)



<<CMOS 管>>



N-MOS管

P-MOS管

CMOS 分為 N 型與 P 型，N-MOS 是指導通後整個晶體管也是 N 狀態，而 P-MOS 是指導通後為 P 狀態，所以 N-MOS 是以 NPN 組成，而 P-MOS 是以 PNP 組成。

也就是 gate 輸入為高電平時，N-MOS 中間層會吸引電子，令中間層也成為 N 從而導通。而 P-MOS 也是吸引電子，中間層也是 N，但是由於 P-MOS 初始是 PNP，因此輸入高電平時仍然會是 PNP 結構，所以不會導通。

	箭咀方向	意思	Gate 輸入
N-MOS	指入	吸引電子時導通	高電壓吸引電子
P-MOS	指出	排走電子時導通	低電壓排走電子

在 CMOS 之前其實是以三極管來組成 TTL(Transistor-Transistor Logic)晶片，但三極管是以電流作驅動，耗電量高，而 CMOS 因為驅動只需使用電壓(中間有絕緣層)，因此只有極少量電流。

### <<N-MOS logic gate>>

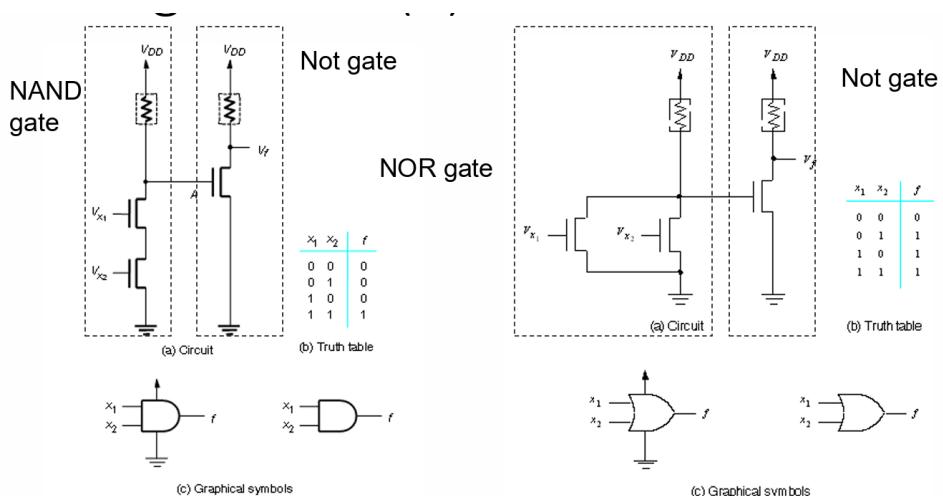
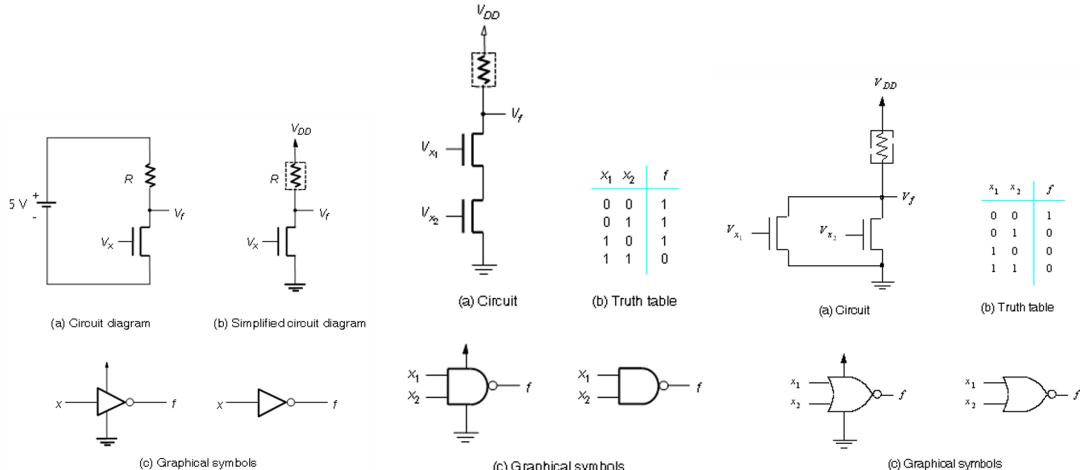
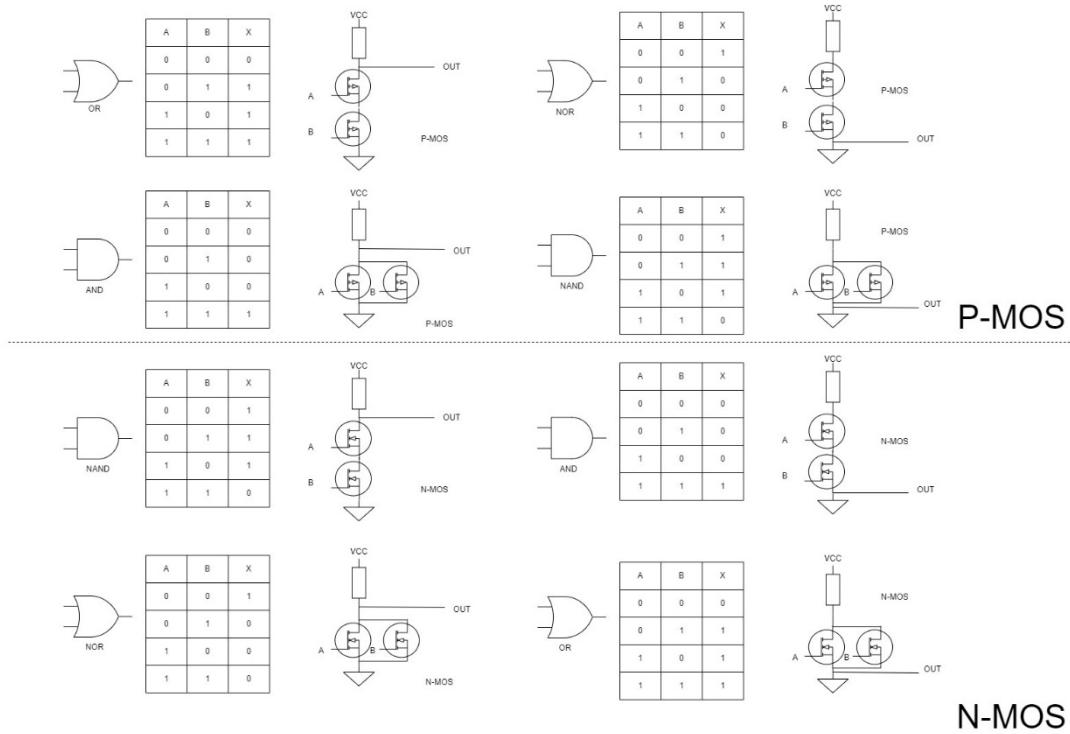


Figure 3.8. NMOS realization of an AND gate.

Figure 3.9. NMOS realization of an OR gate.

## <<P/N-MOS logic gate>>



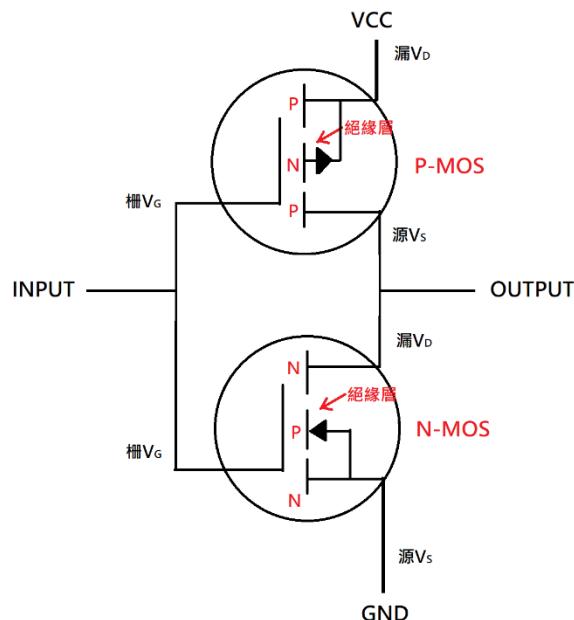
剛好地 P-MOS 的 AND gate 與 N-MOS 的 OR gate 結果是相反的，因此在設計 C-MOS 時就可以用 P-MOS 加 N-MOS 的方式組成，但對方需要是相反電路，即串聯並聯相反，也就是 logic gate 相反。

## <<C-MOS logic gate>>

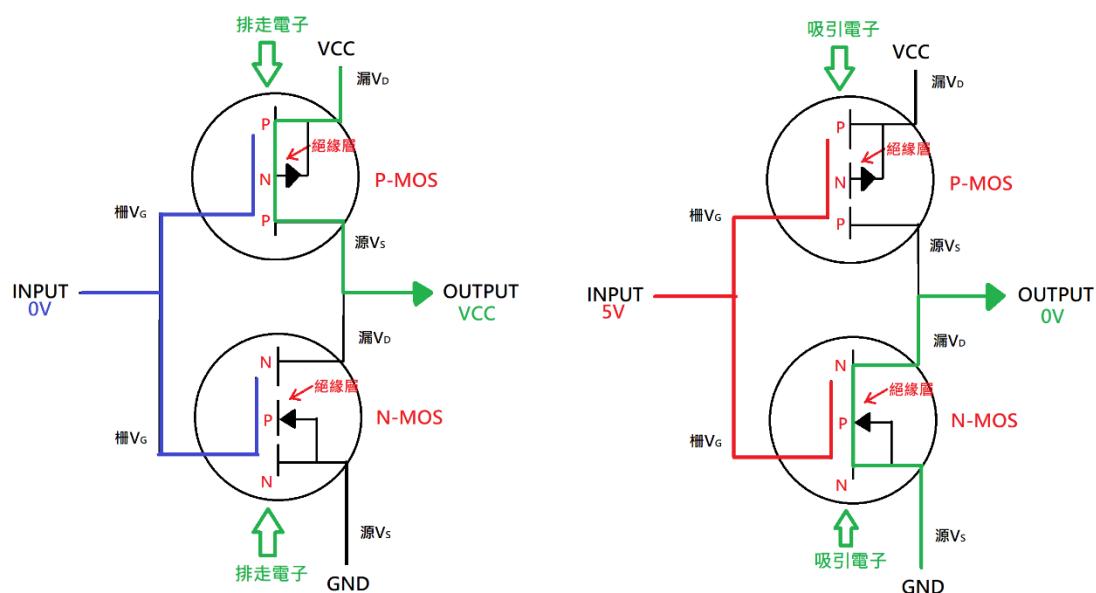
C-MOS 由 N-MOS 加 P-MOS 組成，而由於通常上方是接駁正壓，下方接駁 GND，因此通常擺放 P-MOS 在上方，N-MOS 在下方。

由於普通的單 P-MOS/C-MOS 需要加入電阻，這會因電流而功耗提高，而 C-MOS 則沒有電流。

INPUT = 5V : P-MOS 不通，N-MOS 通
INPUT = 0V : P-MOS 通，N-MOS 不通



2.導通情況就如同 NOT gate。

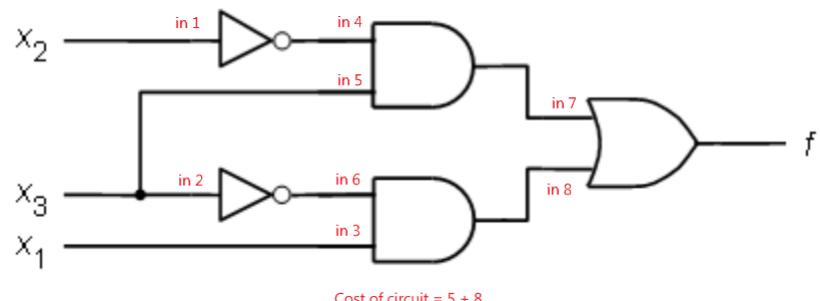


MOSFET and JFET circuit symbols

P-channel					
N-channel					
	JFET	MOSFET enhancement mode	MOSFET enhancement mode (no bulk)	MOSFET depletion mode	CSDN @hungtaowu

<<Cost of circuit>>

The cost of network = total logic gate + total input of all logic gate

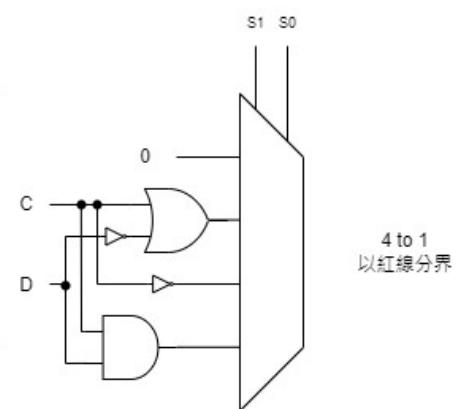


## <<multiplexers>>

例子:

$$Y(A, B, C, D) = \bigcup M(0, 1, 2, 3, 5, 10, 11, 12, 13, 14)$$

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1



解釋:

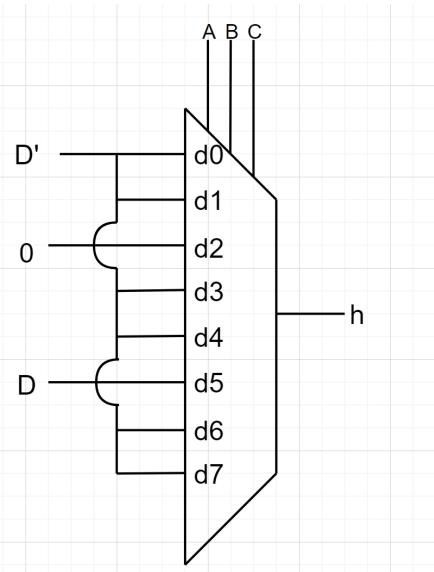
- 1.16 除 4 則為每 4 個答案為一組，得出 4 個答案，稱為 4 to 1
- 2.允許使用 logic gate
- 3.根據該組的 truth 答案判斷該 input 由什麼方法可組成。
- 4.selector 數目等於  $2^n = \text{input}$  數量時的 n 值，例如把 truth 組合成 4 個 input 就等於 1 個 input 時，input 得返 4 個，而 selector 就是 2 個
- 5.input 由最低睇起，即係 D，而 selector 由最高值睇起，即係 A。

<<multiplexers clock table>>

例子:

$$h(a, b, c, d) = \sum m(0, 2, 6, 8, 11, 12, 14)$$

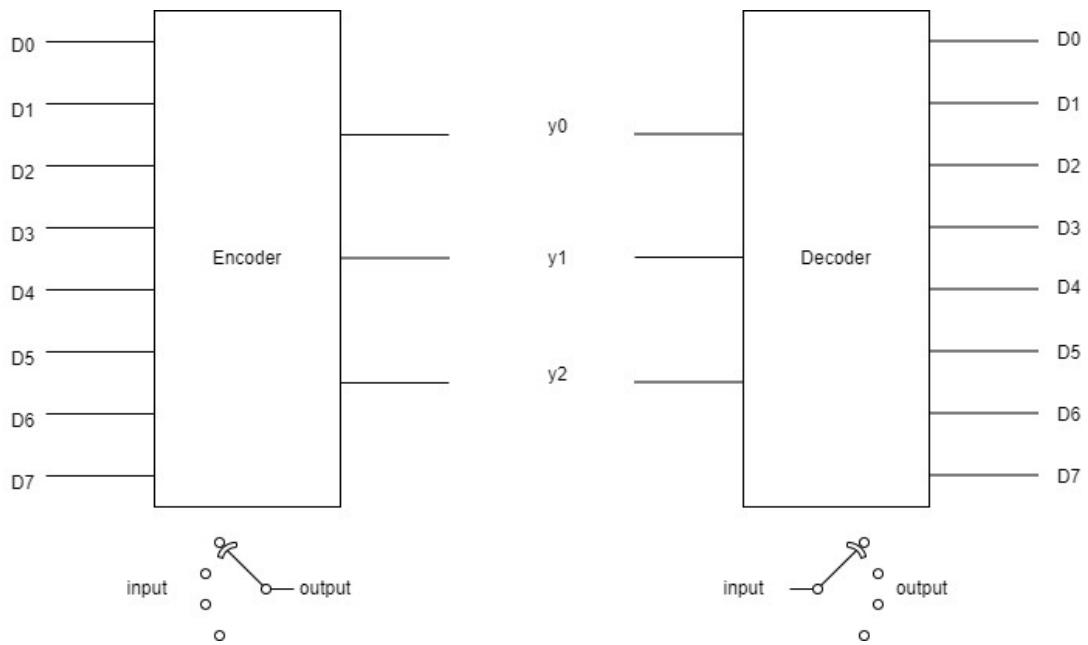
Clock	1	2	3	4	5	6	7	8	9	10
A	0	0	1	0	1	0	1	0	1	0
B	1	0	0	1	0	0	1	0	1	1
C	1	1	1	0	0	0	1	0	1	0
D	0	0	1	1	0	0	1	1	0	0



h	1	1	1	0	1	1	0	0	1	0
---	---	---	---	---	---	---	---	---	---	---

## <<Encoder / Decoder>>

Encoder 只支持單一個 input 為 high，之後就會以較少 bit 數表達返 input source。

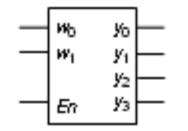


$w_3$	$w_2$	$w_1$	$w_0$	$y_1$	$y_0$
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

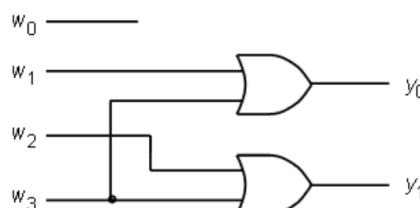
(a) Truth table

$En$	$w_1$	$w_0$	$y_0$	$y_1$	$y_2$	$y_3$
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1
0	x	x	0	0	0	0

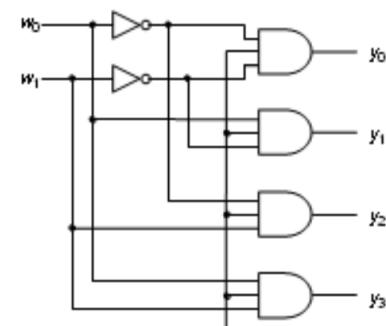
(a) Truth table



(b) Graphical symbol



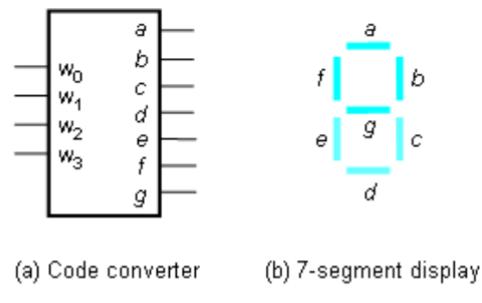
(b) Circuit



(c) Logic circuit

### <<Code converter>>

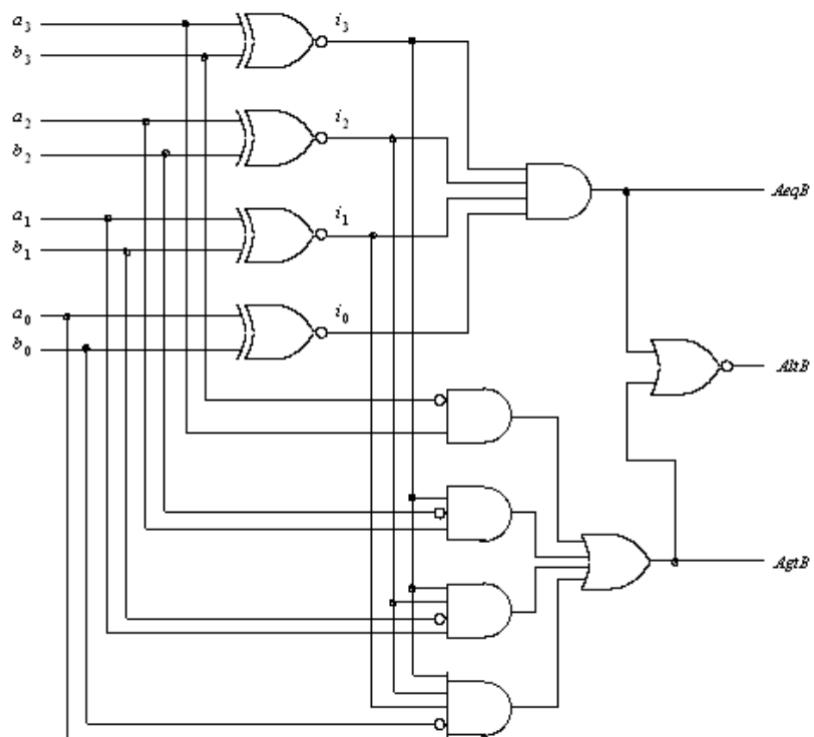
類似 Encoder / Decoder，但不只是支持單一的 input



(a) Code converter      (b) 7-segment display

$w_3 \quad w_2 \quad w_1 \quad w_0$	$a$	$b$	$c$	$d$	$e$	$f$	$g$
0 0 0 0	1	1	1	1	1	1	0
0 0 0 1	0	1	1	0	0	0	0
0 0 1 0	1	1	0	1	1	0	1
0 0 1 1	1	1	1	1	0	0	1
0 1 0 0	0	1	1	0	0	1	1
0 1 0 1	1	0	1	1	0	1	1
0 1 1 0	1	0	1	1	1	1	1
0 1 1 1	1	1	1	0	0	0	0
1 0 0 0	1	1	1	1	1	1	1
1 0 0 1	1	1	1	1	0	1	1

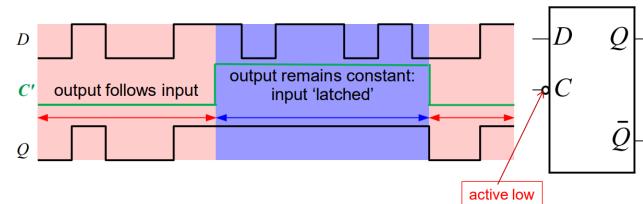
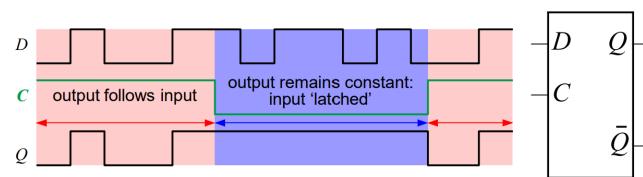
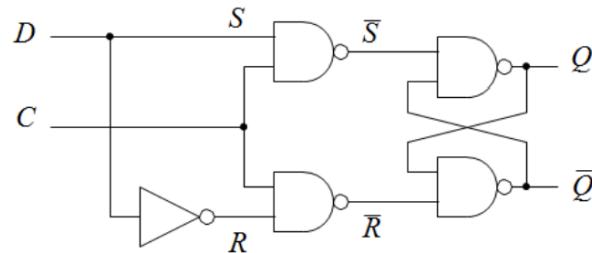
### <<Arithmetic Comparison Circuit 比較器>>



<<Sequential Circuit>>

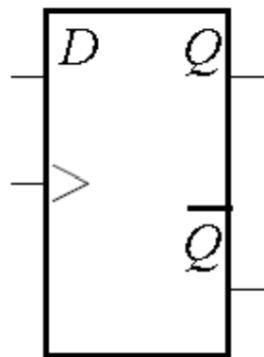
1. D-flip-flop with enable

## D Flip-Flop



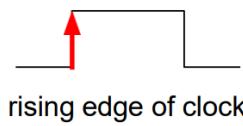
原理：當 C 是 active 時，Q 就會跟隨 D

## 2. D-flip-Flop with edge trigger

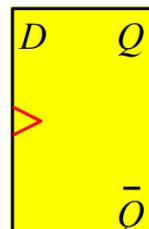


D	Next
0	0 (clear)
1	1 (set)

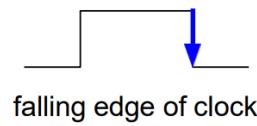
**Positive edge triggered**



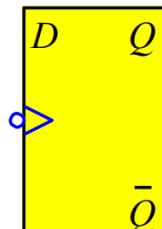
wedge shows **positive** edge triggering



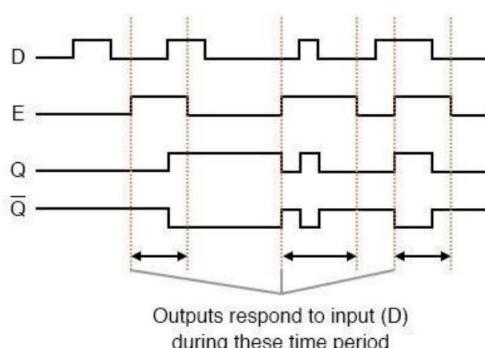
**Negative edge triggered**



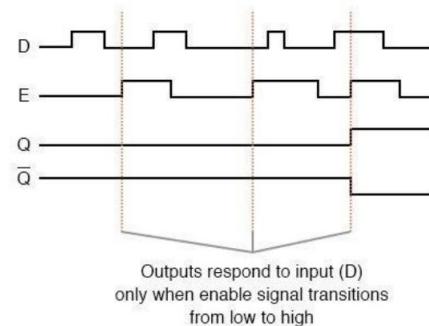
additional of a circle means that there is **negative** edge triggering



Regular D-latch response

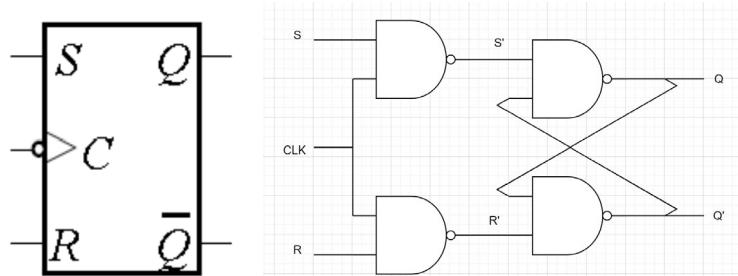


Positive edge-triggered D-latch response



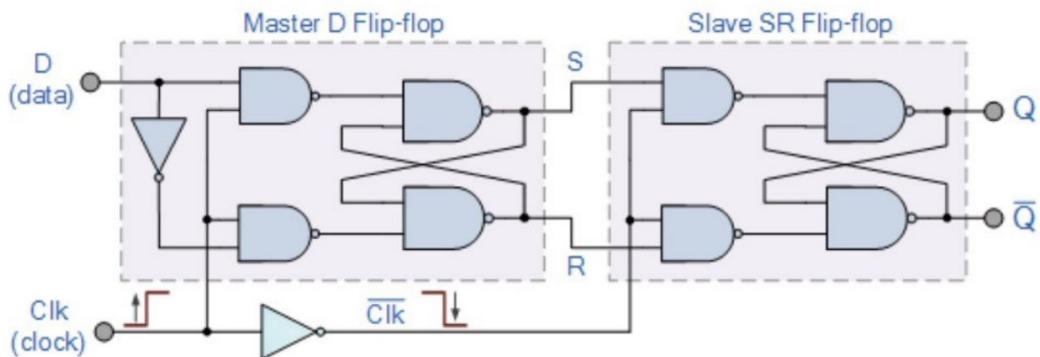
原理：只在邊緣觸發

### 3. SR-Latch

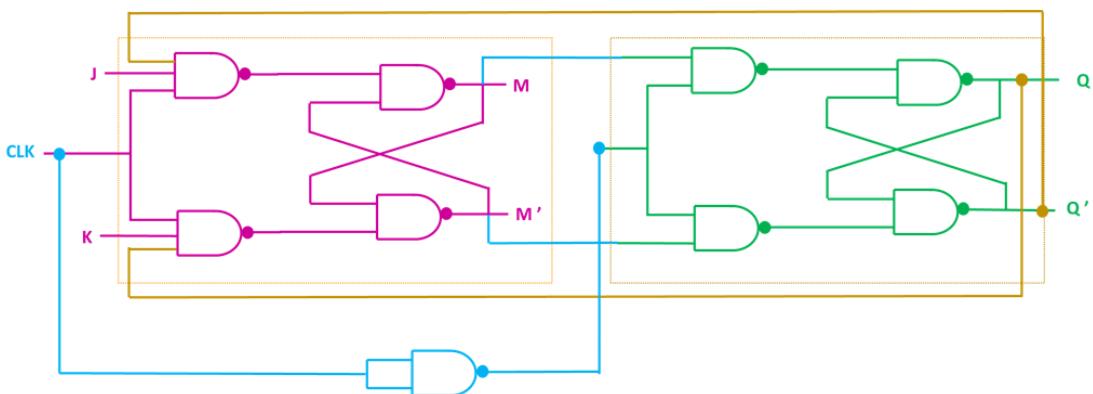


S	R	Next state
0	0	Keep
0	1	0 (clear)
1	0	1 (set)
1	1	? (random)

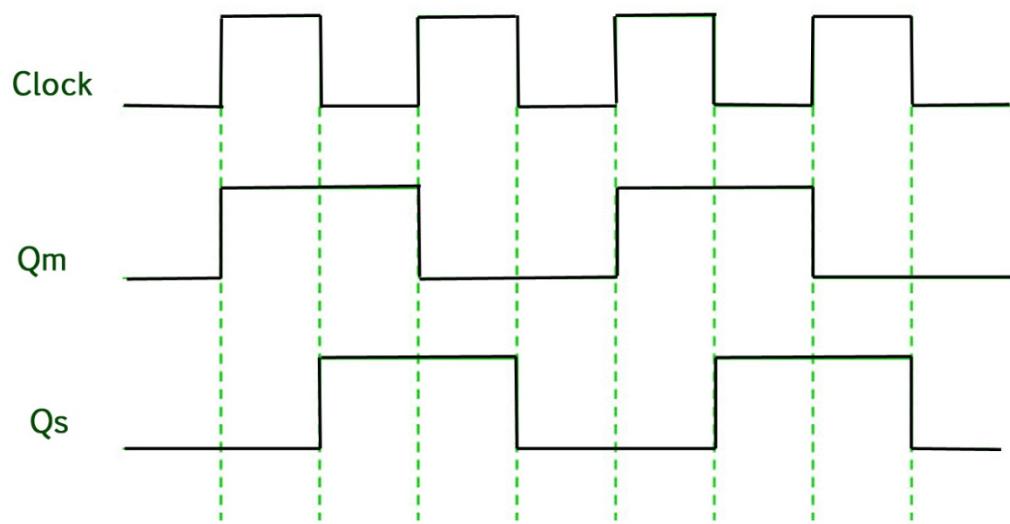
### 4. Transparent Latches



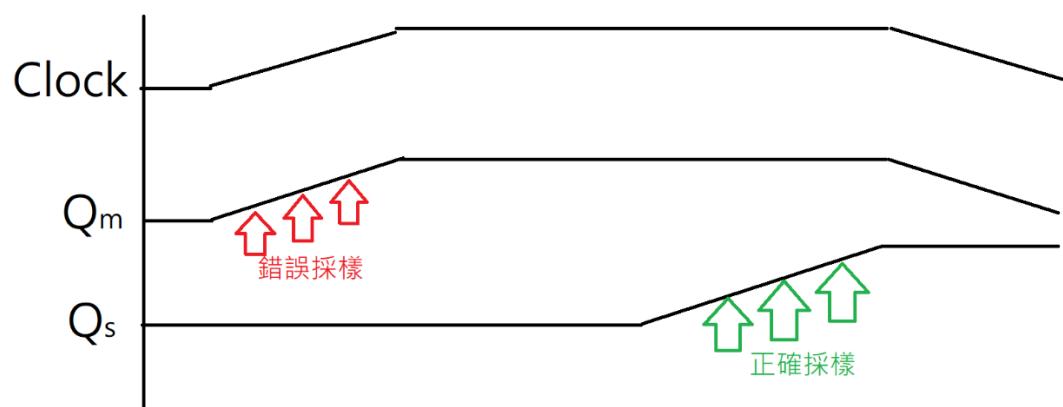
原理: 這是由 Master D Flip-flop 與 slave SR Flip-flop 組成，其中 Master D Flip-flop 是為防止出現 S 與 R 都為 1 的情況，而 output 就是不斷反轉。



原理: 這是 Master slave 版的 JK Flip-flop，由 master JK Flip-flop 及 Slave SR Flip-flop 組成， output 就是不斷反轉。

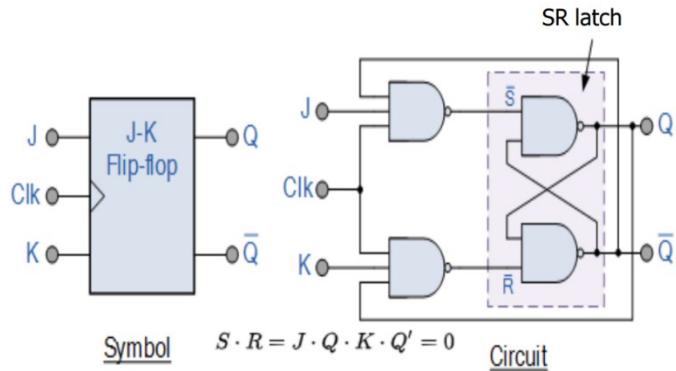


放大後:



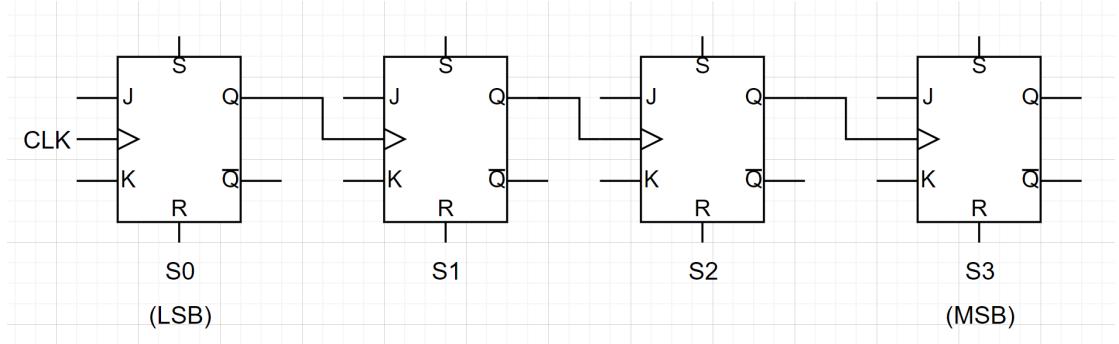
其用途是令到 output 穩定，以免因採樣時得錯誤的信號。

## 5. J-K Flip-Flop

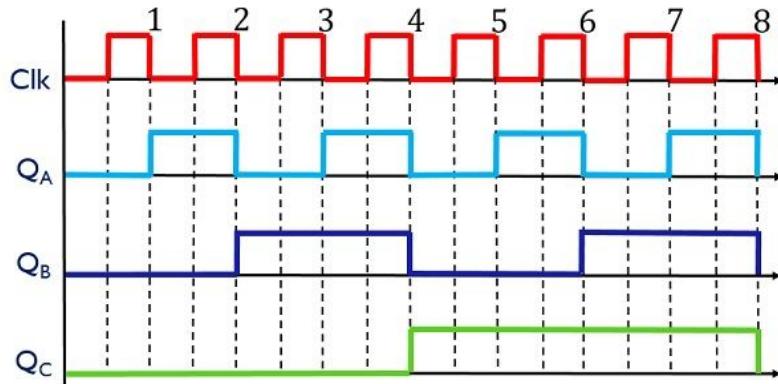


J	K	Next state
0	0	Keep
0	1	0 (clear)
1	0	1 (set)
1	1	Toggle (Change)

### i. Async J-K Flip-Flop (非同步)



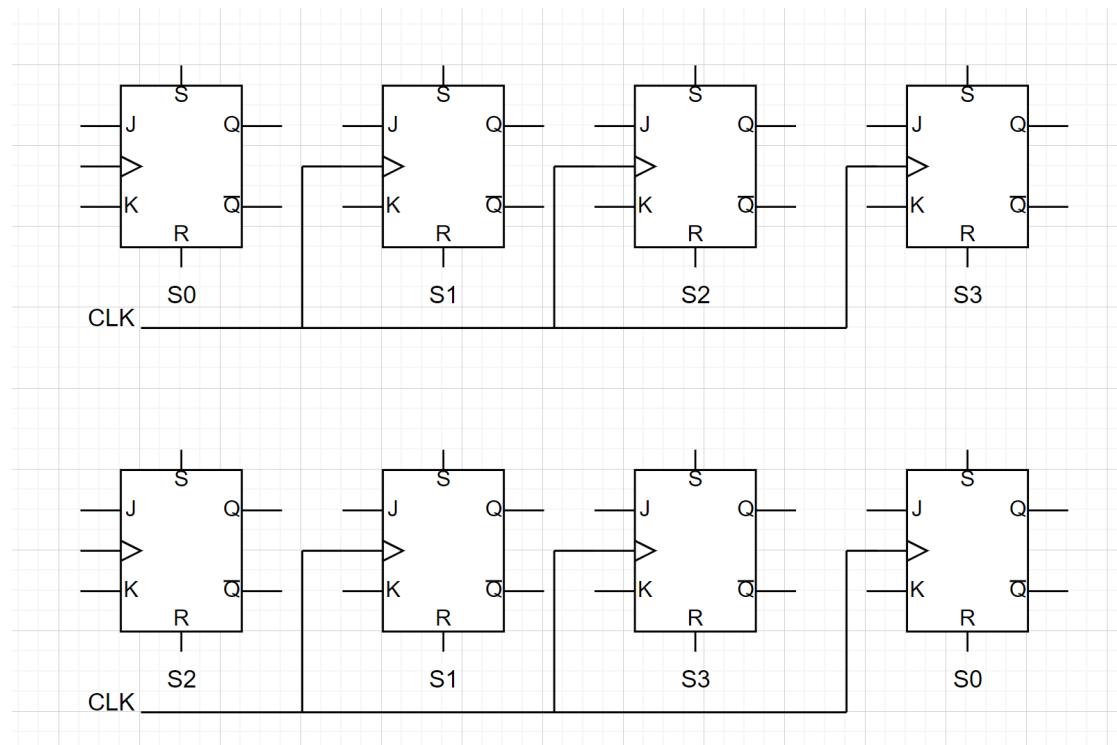
特點: 左手邊 Clock 接駁的一定是 LSB ( $J_0$ )，主要也取決 JK 的 input



Timing Diagram of 3-bit Synchronous Counter

Electronics Coach

ii. Sync J-K Flip-Flop (同步)

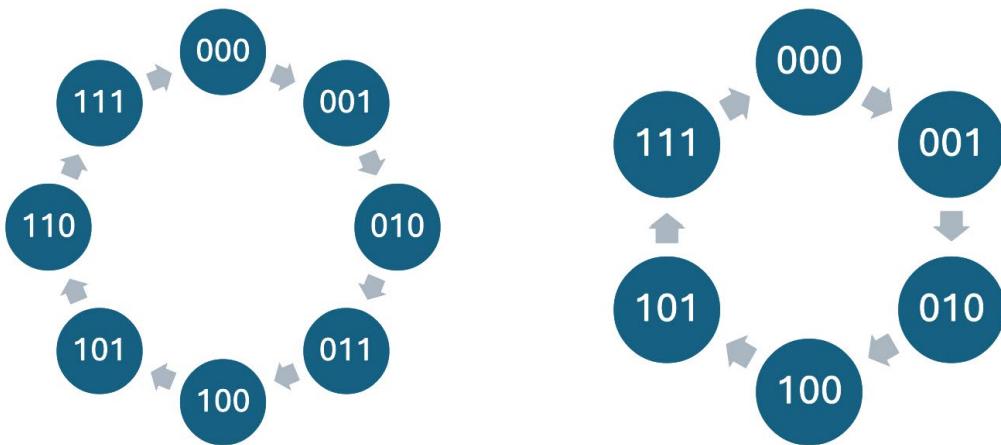


特點:

由於是同步所以不需要把由最細既 Bit 進行排列，任何排列放也是允許，主要也取決 JK 的 input

<< Async Up-Counter>>

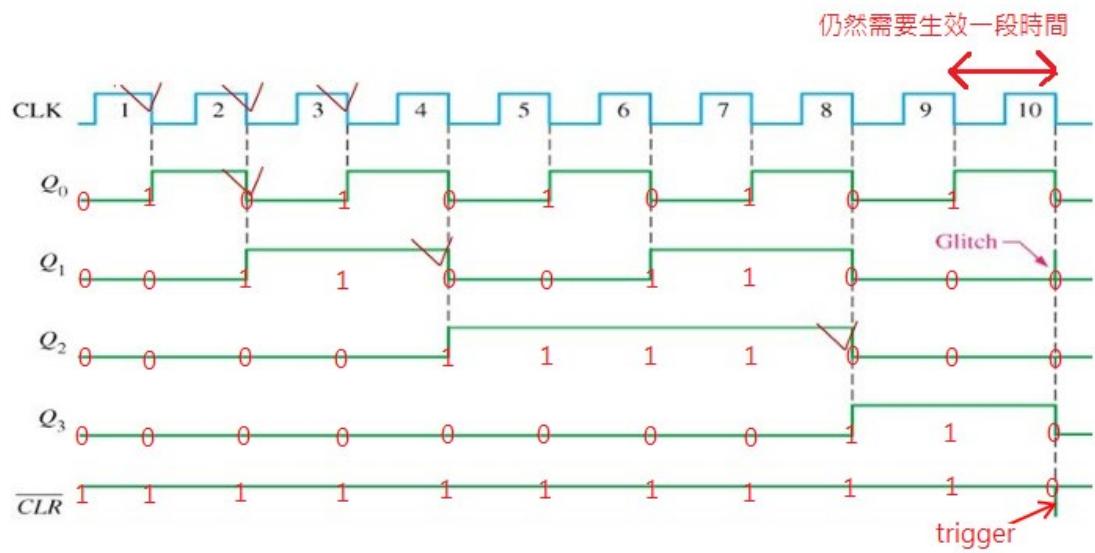
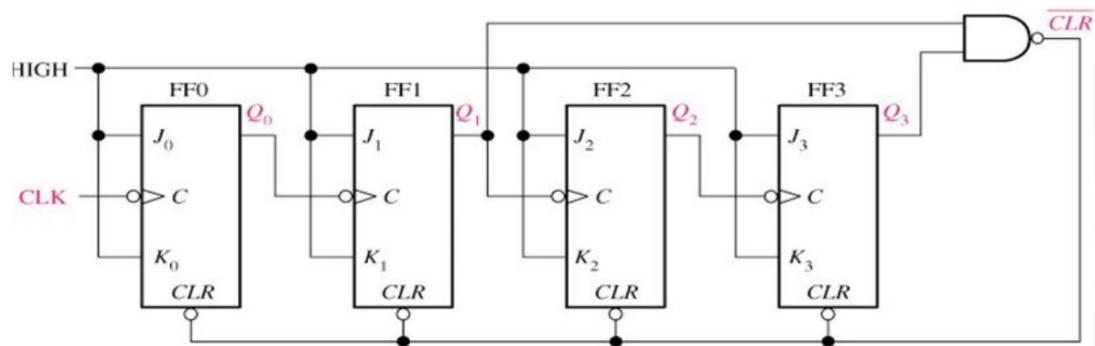
i. 0~7 up-counter & 0,1,2,4,5,7 up-counter , state diagram:



解釋:

用作表達 counter 的順序，而上圖是一個使用 3bit 的 up-counter

## ii. J-K Flip-Flop 10 step counter



解釋:

一個普通的 J-K Flip-Flop async counter 要做到數 10 下，即  $0(0000)_2$  數到  $9(1001)_2$ ，但當到了  $(1010)_2$  時（即數到第 11 下），就會觸發 CLR clean 所有 bit 到  $0000$ ，為何不是  $(1001)_2$  就觸發呢？這是因為數到第 9  $(1001)_2$  時，也需要等佢出現一陣，而不是一到第 9 就馬上還原。

<< Sync up-counter (inverse J-K flip-flop logic) >>

Present	Next	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

原理:

這是一個同步的 J-K flip-flop，因此沒有 async 呀方便，要透過 J-K Excitation Table 及 K-MAP 反推返 J-K input 與 Q output 關係。

例子:

Up-Counter Question : sequence = 0,1,2,4,5,7,0,1,2....

i. J-K Flip-Flop State counter transition table，先列晒 Q output 與 J-K input 的所有情況可能：

Present			Next								
$Q_C$	$Q_B$	$Q_A$	$Q_C$	$Q_B$	$Q_A$	$J_C$	$K_C$	$J_B$	$K_B$	$J_A$	$K_A$
0	0	0	0	0	1	0	x	0	x	1	X
0	0	1	0	1	0	0	x	1	x	X	1
0	1	0	1	0	0	1	x	x	1	0	x
0	1	1	x	X	x	X	x	x	x	x	X
1	0	0	1	0	1	x	0	0	x	1	X
1	0	1	1	1	1	x	0	1	x	x	0
1	1	0	x	x	X	x	x	x	x	x	x
1	1	1	0	0	0	x	1	x	1	x	1

ii. K-map 找出 J-K input 與 Q output 關係:

A	$A'$
B	x x
$B'$	x 1
B	x 1
$B'$	x 0

$$J_A = B'$$

A	$A'$
B	1 x
$B'$	0 x
B	1 x
$B'$	x x

$$J_A = C' + B$$

A	$A'$
B	x x
$B'$	1 0
B	1 0
$B'$	x x

$$J_B = A$$

A	$A'$
B	1 x
$B'$	x x
B	x x
$B'$	x 1

$$K_B = 1$$

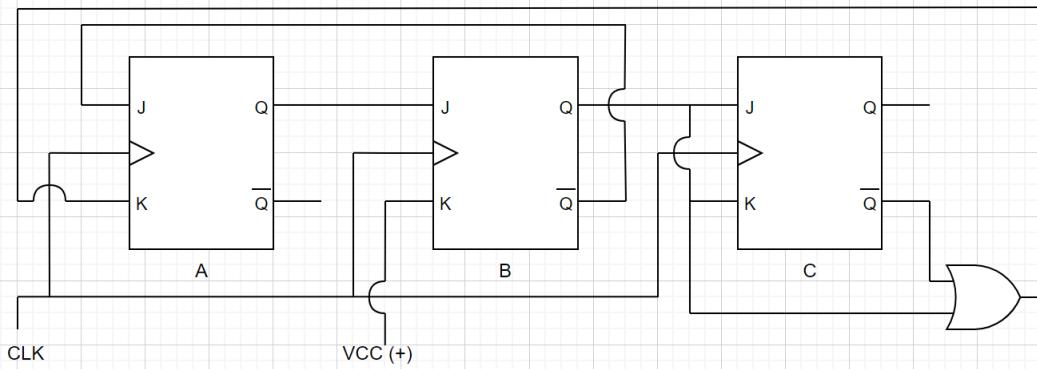
A	$A'$
B	x x
$B'$	x x
B	x x
$B'$	x 1

$$J_C = B$$

A	$A'$
B	1 x
$B'$	0 0
B	0 0
$B'$	x x

$$K_C = B$$

### iii.sync J-K Flip-Flop State counter transition table



<<ADC (analog to digital)>>

$$V_{LSB} = \frac{V_{ref}}{2^n}$$

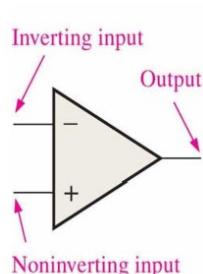
符號	解釋
n	Number of bit
V <sub>LSB</sub>	Voltage of the LSB
V <sub>ref</sub>	Resolution

e.g

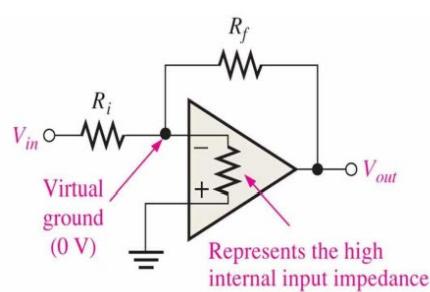
$$V_{LSB} = 2.44mV = \frac{10}{2^{12}}$$

$$\% V_{ref} = 0.0244\%$$

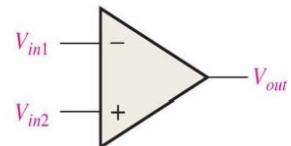
<<Op-Amp>>



(a) Op-amp symbol

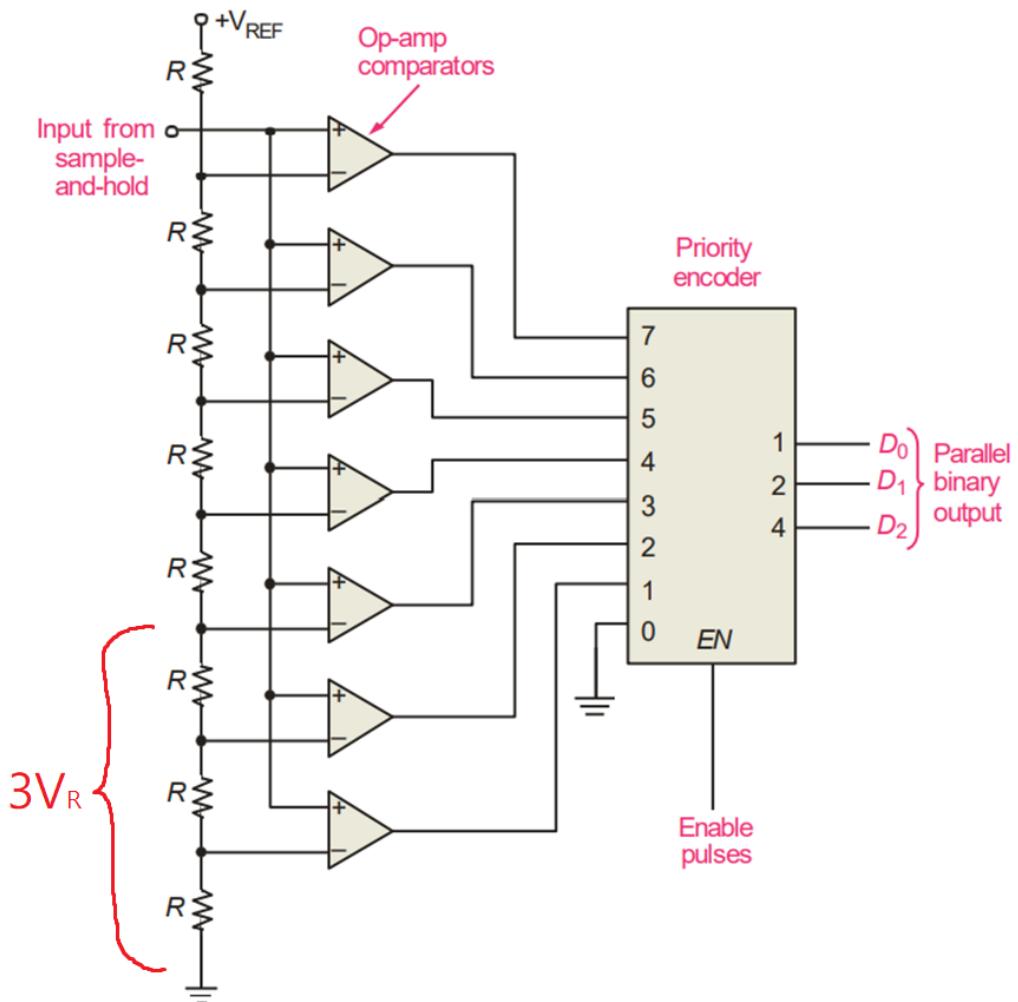


(b) Op-amp as an inverting amplifier  
with gain of  $R_f/R_i$



(c) Op-amp as a comparator

<<Flash ADC>>



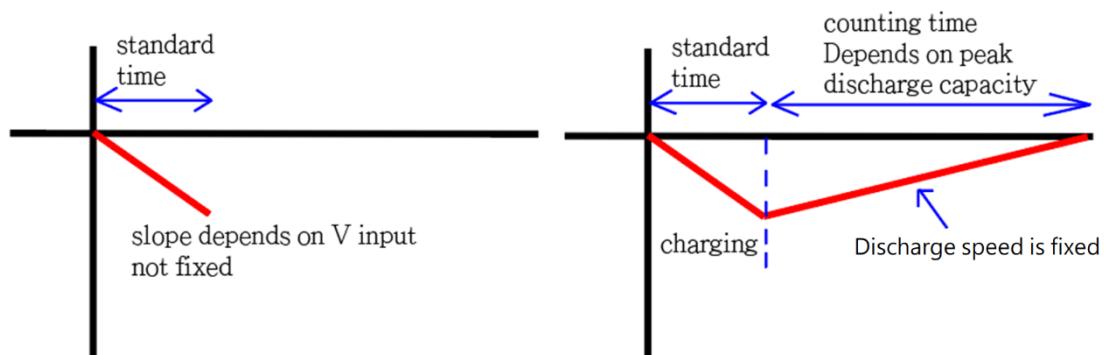
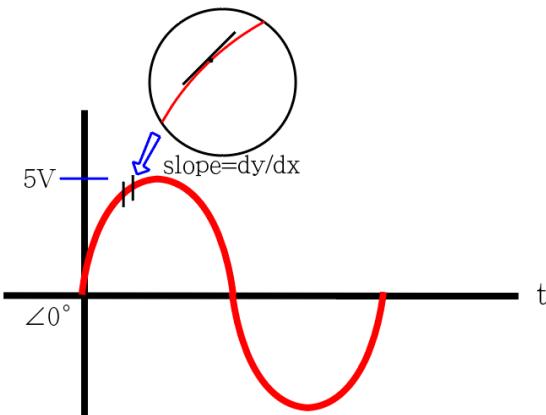
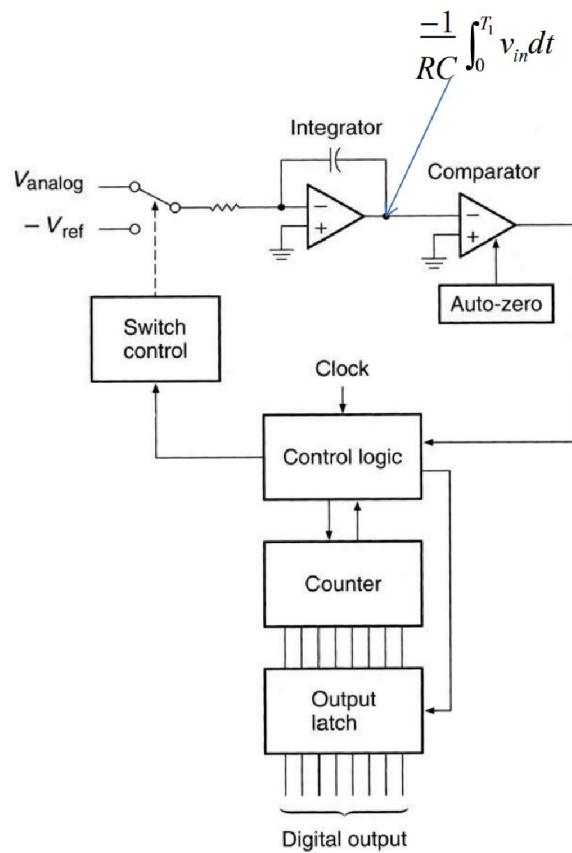
原理:

Op-amp 用作比較器，一層比一層電壓低，之與用比較器比較電壓，睇下電壓去到邊個 level。

好處: 快

壞處: 好易受干擾

<<Dual-slope ADC>>



原理: 截取在模擬信號中的一少段範圍中的電壓值，基本上這段範圍的電壓值也是一樣的，而這個電壓值對 **integrator** 電容進行充電，充電時間是固定的。

第二步，充電到固定時間後會進行放電到 **comparator**，而放電時間就不再是固定，但放電速度就是固定的，在這情況下就會出現不同的放電時間，因此放電時間就 **depends on** 在固定時間內充電的峰值。

而另外，在放電過程中，**Counter** 會開始運作，隨放電時間越長，輸出的 **digital output** 數值則越大。

$$V_{\text{output for comparator}} = \frac{-1}{RC} \int_0^{T_1} V_{in} dt$$

$V_{in}$  = analog 信號某一短暫的輸入電壓值

R = 電阻

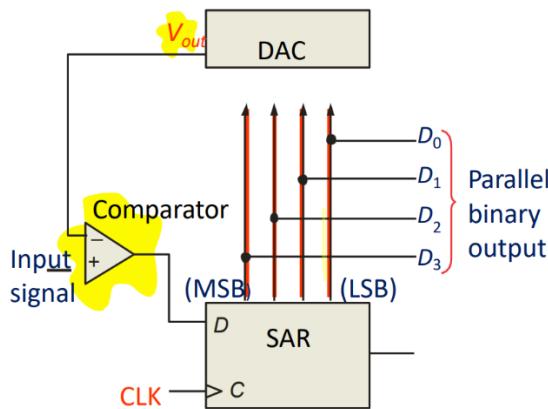
C = 電容

$V_{\text{output for comparator}}$  = 經第一個 **integrator** 後的 **output**，作為比較器的輸入

好處: 得益 **slope** 充電形式，假設突然有一刻受到電壓干擾而提升或降低電壓，但由於只是一瞬而已，**Dual-slope** 是在宏觀上根據一段時間來累積計算，因此影響 **slope** 值不大，從而也就是有良好的抗干擾性。

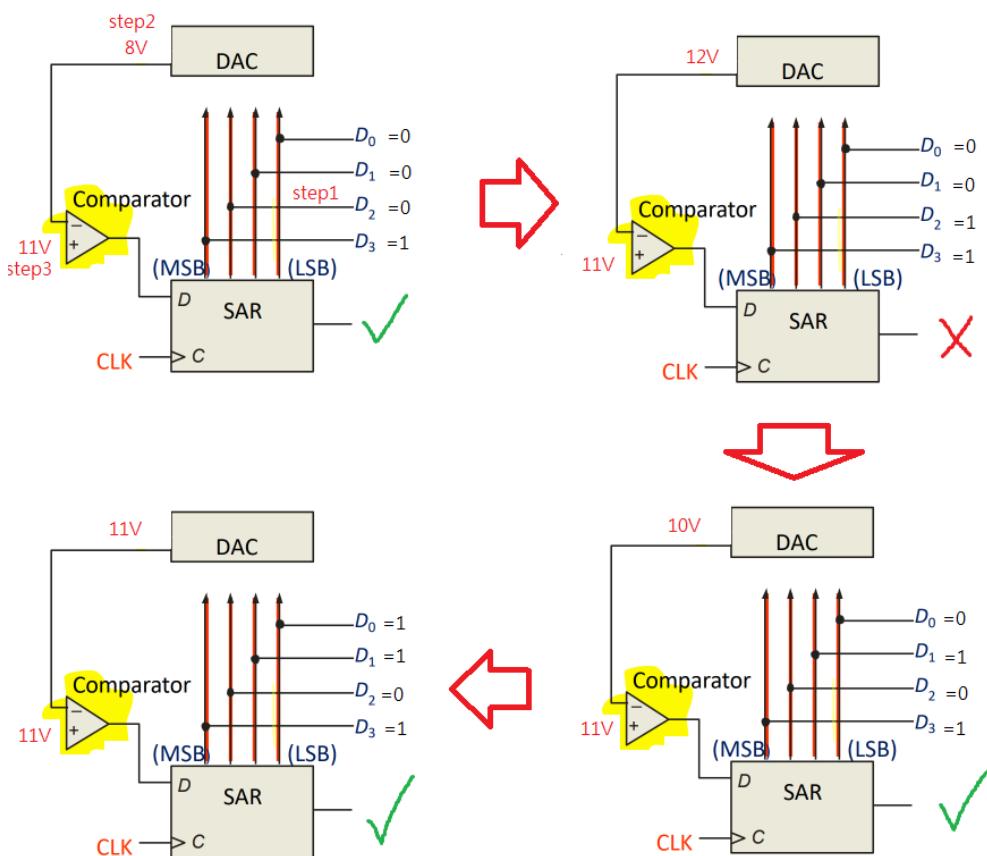
壞處 : 慢

## <<Successive-approximation ADC>>



原理:

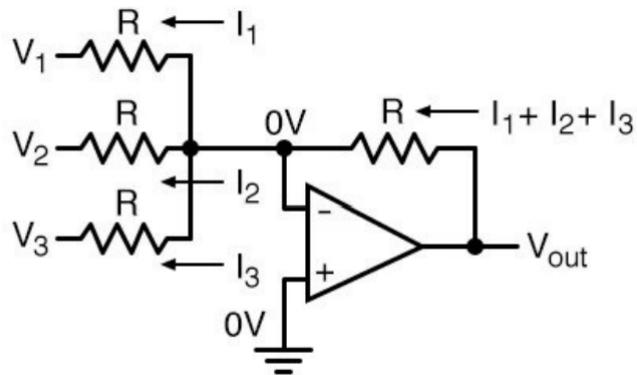
這種 ADC 是組合了 ADC + DAC 才能運作，首先在 Op-Amp 的正極位置輸入 analog 信號的某一刻的電壓值，之後 SAR 寄存器會先發送出最大的 digital 值到 DAC，例如 SAR 發送 1111 到 DAC，DAC 就會發送最大的 analog 值，這裡假設為 5V，由 DAC 產生的 5V 就會 output 到 Comparator 與 Input signal 進行比較，若 輸入較大，則保留該位元；否則重設 (0)，因此這是一種不斷循環過程。



好處: 中等速度, 抗干擾

<<DAC (digital to analog)>>

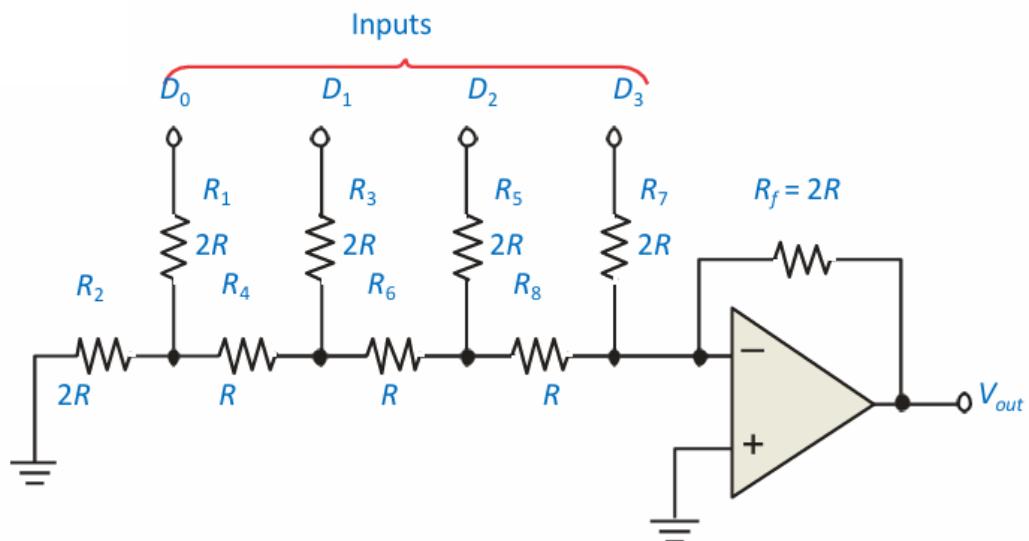
Inverting Summer Circuit



$$V_{out} = -(V_1 + V_2 + V_3)$$

$$\text{Resolution} = \frac{\text{Full - scale voltage}}{2^n - 1}$$

<< DAC R-2R >>



簡介:由於一般的 DAC 所需的電阻值不是統一，而且更可能出現不是常見的電阻值，造成匹配困難，因此就出現 R-2R 形式的電阻電路，目標是盡可能採用同一類型的電阻值電阻。

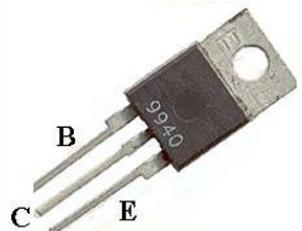
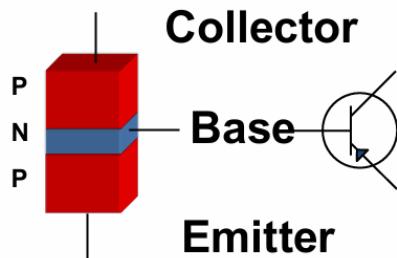
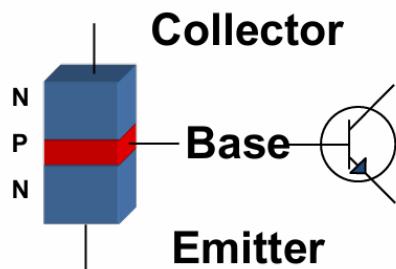
只計算某一 bit(e.g. D0)亮起時所產生的 output 電壓:

$$V_{out \text{ for one bit}} = - \frac{V_s}{2^{n-i}} \left( \frac{R_f}{R_{TH}} \right)$$

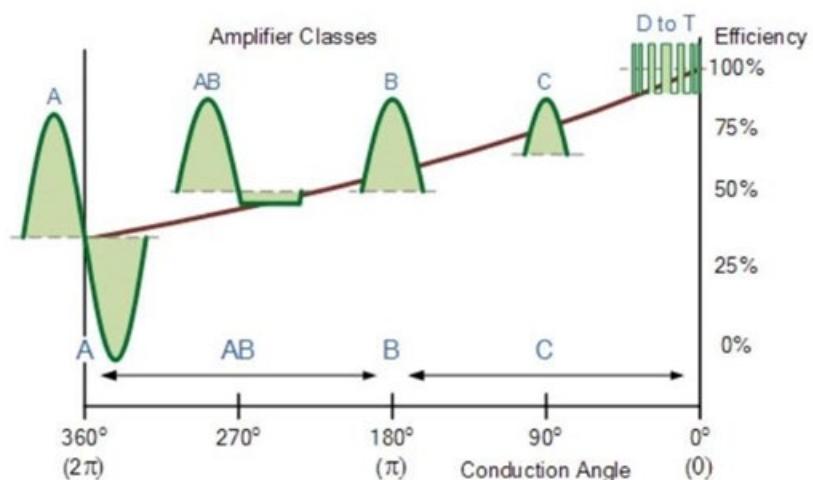
計算所有亮起的 bit(e.g. D<sub>0</sub>, D<sub>2</sub> 都亮起，即 1010)所產生的總 output 電壓:

$$V_{total\ out} = \sum -\frac{V_s}{2^{n-i}} \left( \frac{R_f}{R_{TH}} \right)$$

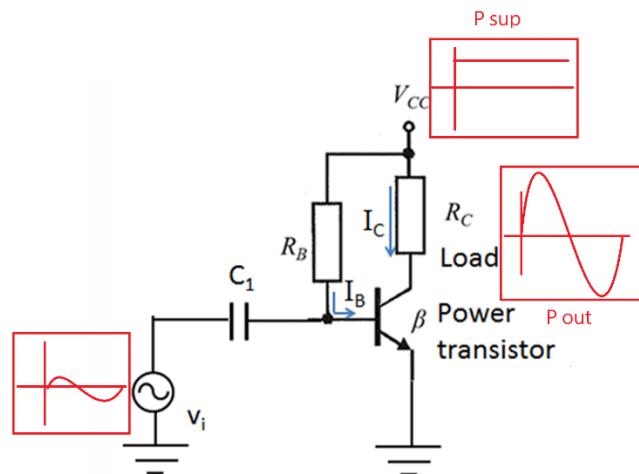
<<Power Amplifiers>>



## Classes of Power Amplifiers



<<Class A Amplifier>>



特點: 與 transistor 類似，但 C 與 B 之間加多左個  $R_B$ 。

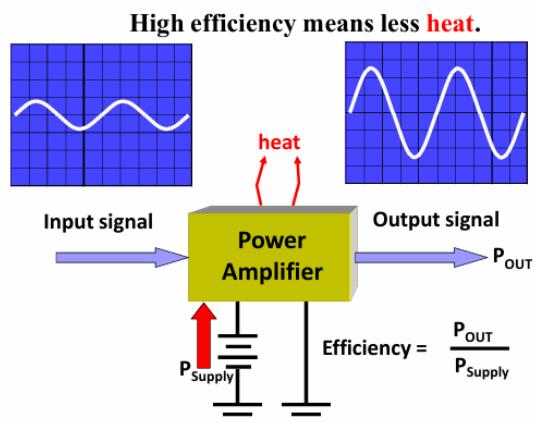
$$V_{cc} = 0.7 + I_B R_B$$

$$V_{cc} = V_{CE} + I_c R_c$$

$$\beta = \frac{I_C}{I_B}$$

$$V_{rms} = \frac{V_p}{\sqrt{2}} = \frac{V_{pp}}{2\sqrt{2}}$$

$$P_{out} = \frac{V_p^2}{2R_c} = \frac{\left(\frac{V_{pp}}{2}\right)^2}{2R_c} = \frac{V_{pp}^2}{8R_c}$$



$$\eta = \frac{P_{out}}{P_{sup}} \times 100\%$$

P <sub>out</sub>	真正放大後的 analog power
P <sub>sup</sub>	放大器供應的 power
V <sub>rms</sub>	V out analog 信號的平均值
V <sub>p</sub>	V out analog 信號的 peak
V <sub>pp</sub>	V out analog 信號的 peak to peak/loading)
$\eta$	功率的效率
V <sub>CC</sub>	放大器輸入電壓
I <sub>BQ</sub>	
I <sub>CQ</sub> = I <sub>C</sub>	
I <sub>B</sub> = I <sub>analog</sub> + I <sub>BQ</sub>	

<<Class B/AB Amplifier>>

