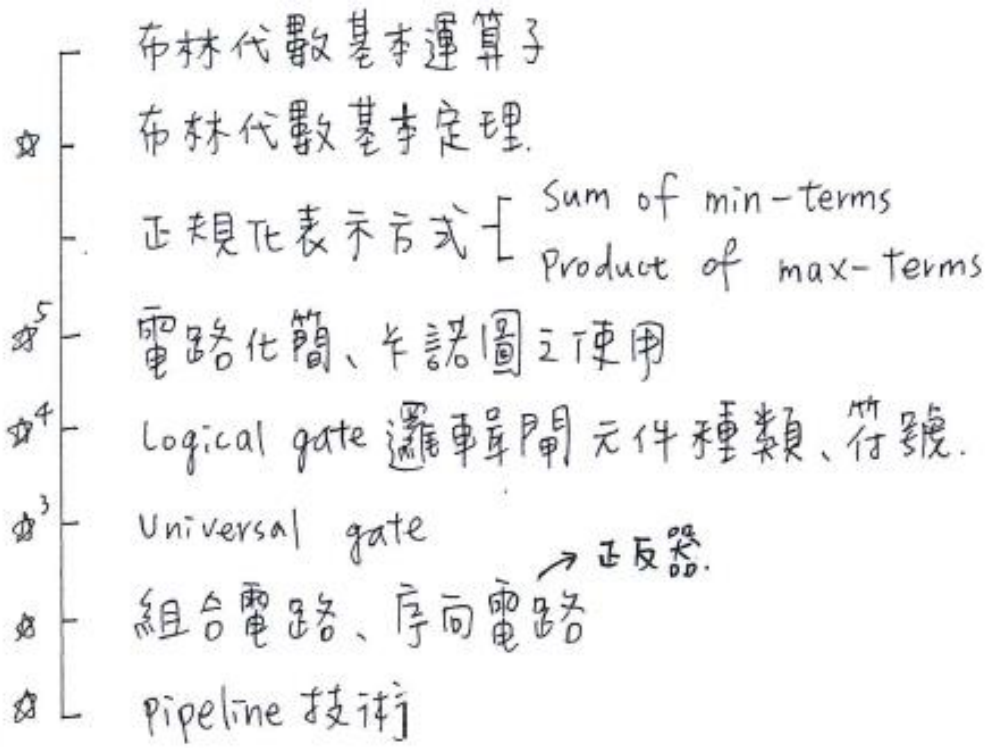
CH 6 數位邏輯



1. 布林代數基本運算

- AND (•)

  ⇨ Input 皆為 1 的時候為 1，否則為 0

A	B	A•B
0	0	0
0	1	0
1	0	0
1	1	1

- OR(+)

  ⇨ Input 皆為 0 的時候為 0，否則為 1(有 1 個為 1 就 1)

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

- NOT

(3) NOT 否定 一，

Def: 1變0，0變1.

A	$\bar{A}$
0	1
1	0

表3.2.2 基本邏輯閘與布林代數關係

邏輯閘	簡稱	布林代數	運算式
反向閘	NOT	NOT	$X=\bar{A}$
及閘	AND	AND	X=A・B
或閘	OR	OR	X=A+B
反及閘	NAND	NOT-AND	$X=\overline{A \cdot B}$
反或閘	NOR	NOT-OR	$X=\overline{A+B}$
互斥或閘	XOR	EX-OR	$X=A \oplus B$
互斥反或閘	XNOR	EX-NOR	$X=\overline{A \oplus B}$

2. 布林代數基本定理(9 個)

(1) 單一律 $A \cdot A = A$

$A + A = A$

(2) 基值律 $A \cdot 1 = A$ $A + 1 = 1$

(基本律) $A \cdot 0 = 0$ $A + 0 = A$

(3) 補數律 $\bar{\bar{A}} = A$ $A + \bar{A} = 1$ $A \cdot \bar{A} = 0$

(4) 交換律 $A+B = B+A$ $A \cdot B = B \cdot A$

(5) 結合律 $(A \cdot B) \cdot C = A \cdot (B \cdot C)$

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

(6) 分配律 $A \cdot (B+C) = (A \cdot B)+(A \cdot C)$

$A+(B \cdot C) = (A+B) \cdot (A+C)$

A	B	$\overline{(A+B)}$	$\bar{A} \cdot \bar{B}$	$\overline{(A \cdot B)}$	$\bar{A}+\bar{B}$
0	0	1	1	1	1
0	1	0	0	1	1
1	0	0	0	1	1
1	1	0	0	0	0

☆(7) 笛摩根定理 對一個布林式子取補成(否定)，則結果為：

- 項次變補
- ・變＋
- ＋變・

eg. $\overline{(A+B)} = \bar{A} \cdot \bar{B}$

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

A	B	A+B	$\overline{A+B}$	$\overline{A}$	$\overline{B}$	$\overline{A}\cdot\overline{B}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

兩者所得結果相同

A	B	A·B	$\overline{A\cdot B}$	$\overline{A}$	$\overline{B}$	$\overline{A}+\overline{B}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

兩者所得結果相同

(8) 吸收定理 eg. $x + xy = x$ Pf: $x\cdot 1 + x\cdot y = x\cdot(1+y) = x\cdot 1 = x$

eg. $x + xy + xz + x\bar{w} + x\alpha\beta\bar{r} = x\cdot(1+y+z+\bar{w}+\alpha\beta\bar{r}) = x\cdot 1 = x$

別: $A+B = A+\overline{A}B$

Pf: $A+B = A+1\cdot B = A+(A+\overline{A})\cdot B = A+AB+\overline{A}B = A+\overline{A}B$
(2)基值律 (3)補救律 (6)分配律 (8)吸收.

(9) 對偶律 用途: 已知某個布林式子成立. 則可以用對偶律轉換得到另一個也成立的布林式子。

轉換方式: {-1變0 -0變1 -·變+ -+變·}

eg. 已知成立
(1) $A+\overline{A}=1 \xrightarrow{\text{對偶.}} A\cdot\overline{A}=0$
(2) $A\cdot 1 = A \longrightarrow A+0=A$

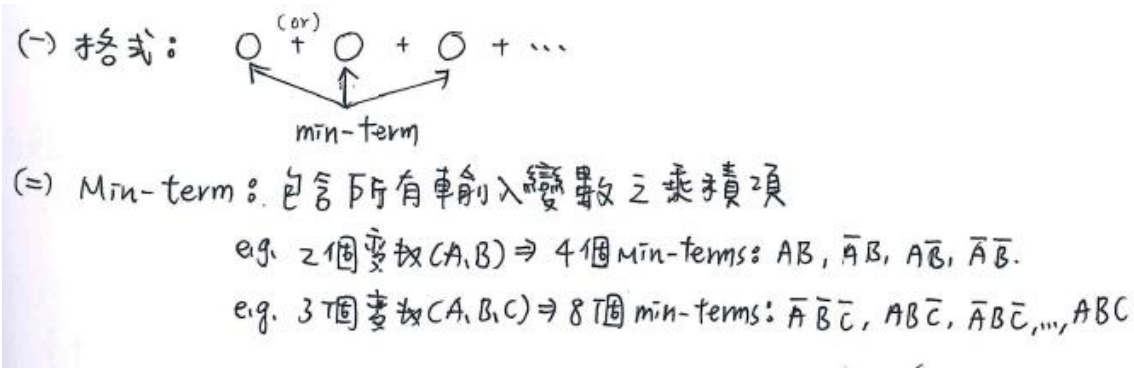
3. 正規化表示法

- 「最小項的和」(sum of minterms)

  ⇨ 將最小項以「OR」運算子結合，也就是用「＋」運算子

結合，便是「最小項的和」

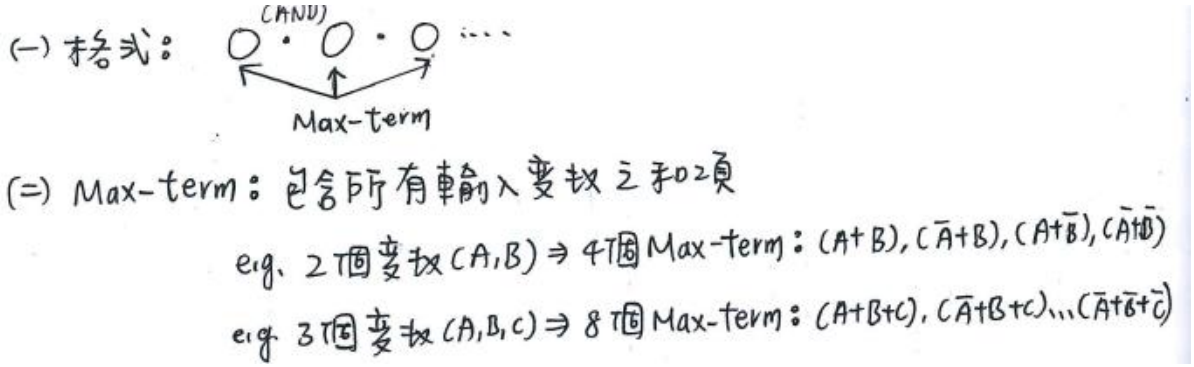
⇨ N 個變數，2^n個組合



- 「最大項的積」(product of maxterms)

⇨ 將最大項以「AND」運算子結合，也就是用「・」運算子

結合，便是「最大項的積」



Example

$F = ABC' + A'B' + BC'$ 3 variables 7 literals

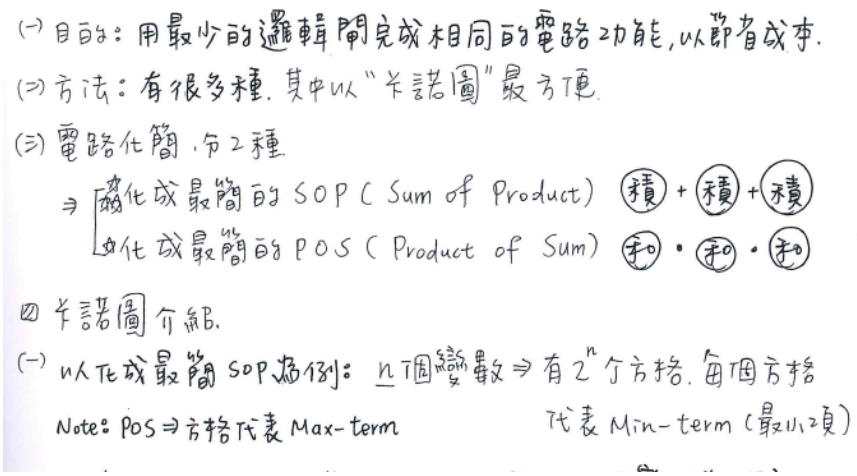
Row No.	A	B	C	Mintems			Maxterms			f	f'
0	0	0	0	A'B'C'	=	m_0	A+B+C	=	M_0	0	1
1	0	0	1	A'B'C	=	m_1	A+B+C'	=	M_1	0	1
2	0	1	0	A'BC'	=	m_2	A+B'+C	=	M_2	0	1
3	0	1	1	A'BC	=	m_3	A+B'+C'	=	M_3	1	0
4	1	0	0	AB'C'	=	m_4	A'+B+C	=	M_4	1	0
5	1	0	1	AB'C	=	m_5	A'+B+C'	=	M_5	1	0
6	1	1	0	ABC '	=	m_6	A'+B'+C	=	M_6	1	0
7	1	1	1	ABC	=	m_7	A'+B'+C'	=	M_7	1	0

$$m_i' = Mi$$

$$\Rightarrow \quad f = A'BC + AB'C' + AB'C + ABC' + ABC$$
$$= m_3 + m_4 + m_5 + m_6 + m_7$$
$$or \quad f(A, B, C) = \sum m(3, 4, 5, 6, 7)$$
$$\min \text{ term} = 1 \quad \Rightarrow \quad f = 1$$

$$f = (A + B + C)(A + B + C')(A + B' + C) = M_0 M_1 M_2$$
$$f(A, B, C) = \prod M(0, 1, 2)$$
$$\text{maxterm} = 0 \quad \Rightarrow \quad f = 0$$

4. 電路化簡

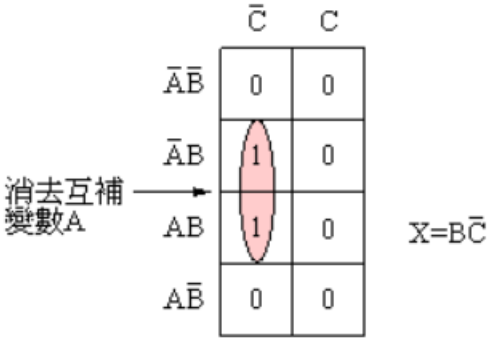


- SOP(卡諾圖)

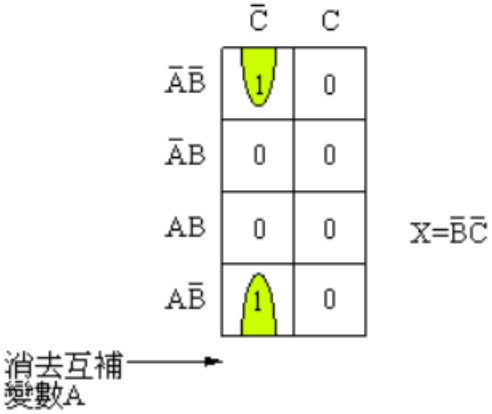
化簡的步驟：

1. 將在真值表中可產生 1 的每個基礎乘積項，對應的填入卡諾圖的空格中，並標記為 1 ，其他的空格則填入 0 。
2. 依序圈出相鄰的 8 個 1 、相鄰的 4 個 1 、相鄰的 2 個 1 ，空格中的 1 可被重複圈選，以便消除最多的變數。
3. 如果還留下獨立的 1 ，也要個別圈選。
4. 觀察圈選的狀況，要讓所有 1 的空格都被圈到，而圈選的組數要愈少愈好。
5. 每一個圈選的結果是一個乘積項，將所有的乘積項OR起來即是化簡後的布林代數式。

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

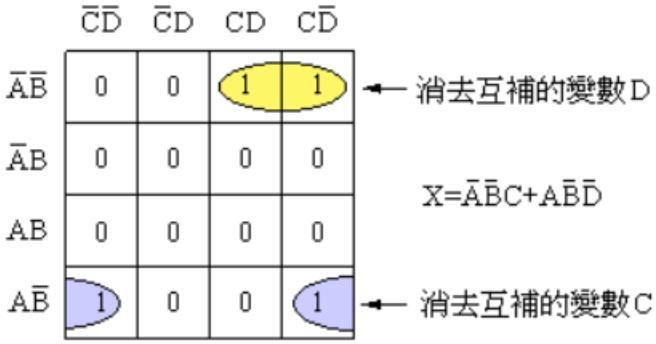


(a) $X=\bar{A}\bar{B}\bar{C}+A\bar{B}\bar{C}$



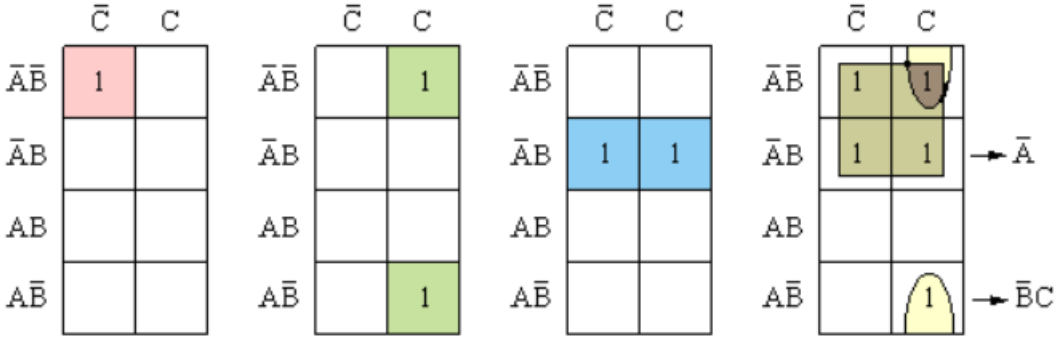
上下兩列亦屬於相鄰的方格

(c) $X=\bar{A}\bar{B}CD+\bar{A}B\bar{C}D+A\bar{B}\bar{C}\bar{D}+AB\bar{C}\bar{D}$



左右兩列亦屬於相鄰的方格

利用卡諾圖化簡 $X=\bar{A}\bar{B}\bar{C}+\bar{B}C+\bar{A}B$ 之布林代數式



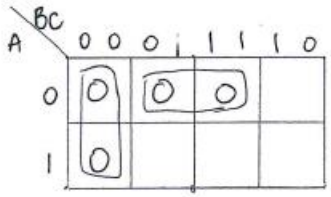
- POS

如何成最簡的 POS (Product of Sum)

步驟相同，差別：
- 每一个方格代表最大項，填"0"
- 合併相鄰方格時，要寫出和項
  （1變補，0不補）
- 最後再乘起來。

例：$F(A,B,C) = \Pi_1(0,1,3,4)$ 成 Simplified POS

000 001 011 100



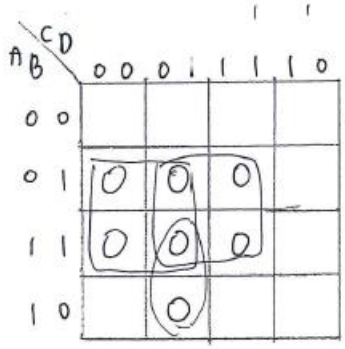
□：$(A+\bar{C})$

□：$(B+C)$

∴ $(A+\bar{C})\cdot(B+C)$ ※

例：$F(A,B,C,D) = (\bar{A}+\bar{B}+\bar{D})\cdot(\bar{A}+C+\bar{D})\cdot(A+\bar{B}+\bar{D})\cdot(\bar{B}+C+D)$

1 1 1　1 0 1　0 1 1　1 0 0



□：$(\bar{B}+C)$

□：$(\bar{B}+\bar{D})$

□：$(\bar{A}+C+\bar{D})$

∴ $(\bar{B}+C)\cdot(\bar{B}+\bar{D})\cdot(\bar{A}+C+\bar{D})$ ※

- 隨意條件(Don't care condition)

  ⇨ 並非所有的輸入狀況皆會發生，其對輸出不重要

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

隨意

	$\bar{C}$	C
$\bar{A}\bar{B}$	0	0
$\bar{A}B$	0	X
AB	X	1
$A\bar{B}$	1	1

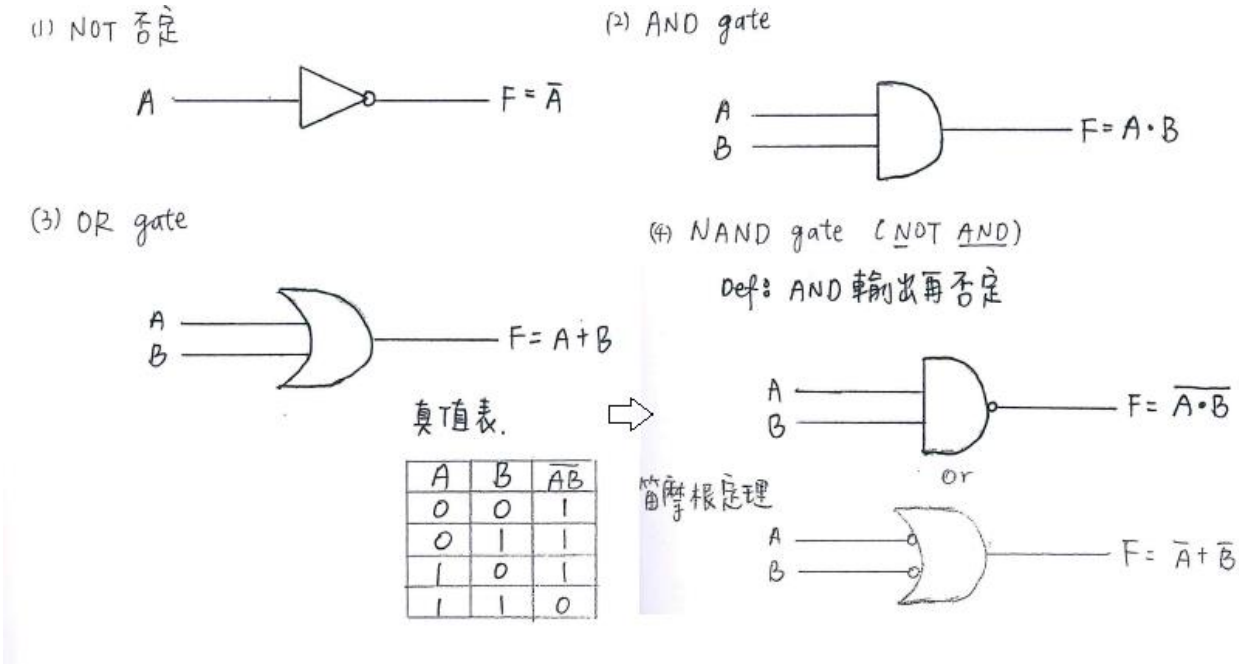
當成0

當成1

	$\bar{C}$	C
$\bar{A}\bar{B}$	0	0
$\bar{A}B$	0	0
AB	1	1
$A\bar{B}$	1	1

Y=A

5. 邏輯閘(Logic Gate)



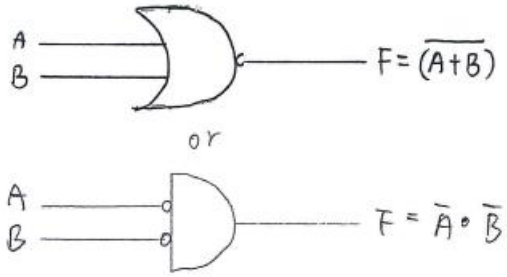
A	B	$\overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0

(5) NOR gate (NOT OR)

Def: OR 輸出再否定

真值表

A	B	$\overline{(A+B)}$
0	0	1
0	1	0
1	0	0
1	1	0

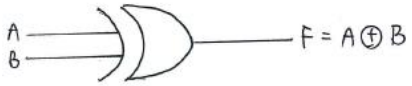


(6) XOR gate (exclusive-OR gate 互斥或)

- 相同為 0，相異為 1。
- 真值表

A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

- $A \oplus B \iff \bar{A}B + A\bar{B}$ 輸流補



(7) XNOR gate (exclusive-NOR 互斥反或)

- 相同為 1，相異為 0。
- 真值表

A	B	A⊙B
0	0	1
0	1	0
1	0	0
1	1	1

- $A \odot B \iff \bar{A}\bar{B} + AB$ 同為補 + 同為不補

例: $\overline{A \oplus B} = A \odot B$

Pf: $\overline{A \odot B} = \overline{(\bar{A}B + A\bar{B})} = \overline{\bar{A}B} \cdot \overline{A\bar{B}} = (A+\bar{B}) \cdot (\bar{A}+B)$

$= A\bar{A} + AB + \bar{B}\bar{A} + \bar{B}B$ ($A\bar{A}$, $\bar{B}B$ = 0)

$= \bar{A}\bar{B} + AB = A \odot B$

例1: $F(A,B,C) = \bar{A}B + A\bar{B}\bar{C} + AB$ 化成 simplified SOP

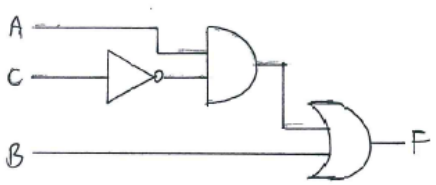
並繪出電路圖 (Circuit Diagram)

Ans:

A \ BC	00	01	11	10
0			1	1
1	1		1	1

$\therefore B + A\bar{C}$

電路圖如下:

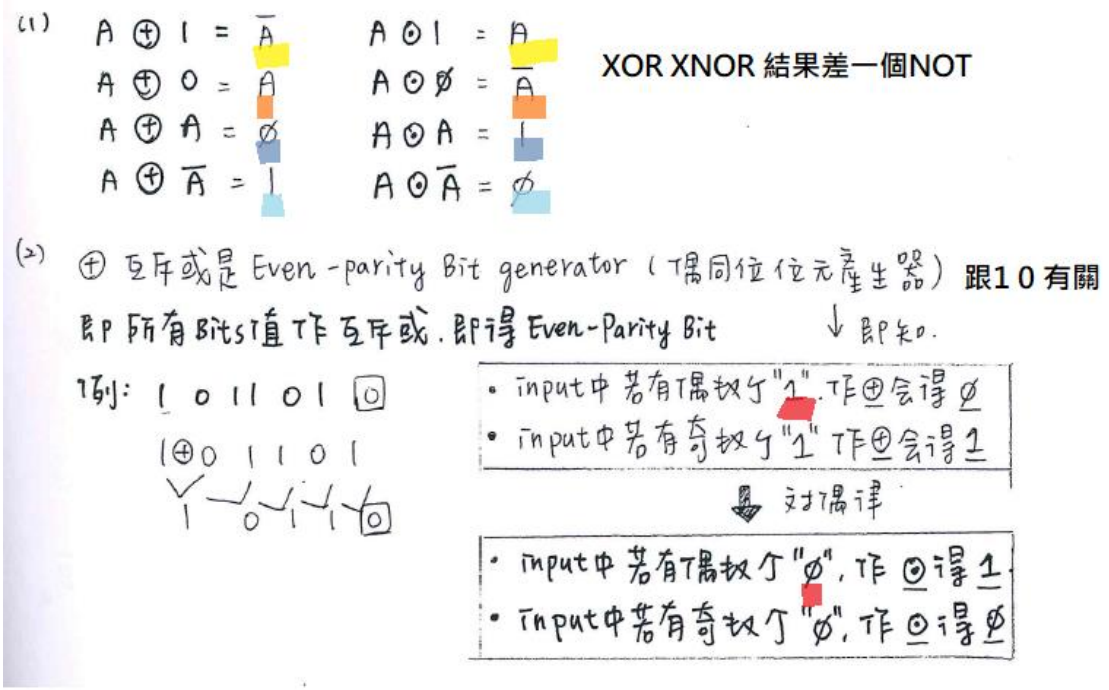


6. Universal Gate C Complete Gate 萬用閘/完整閘

Def: 單用此閘可以完成所有電路 (或其它 gates) 功能。

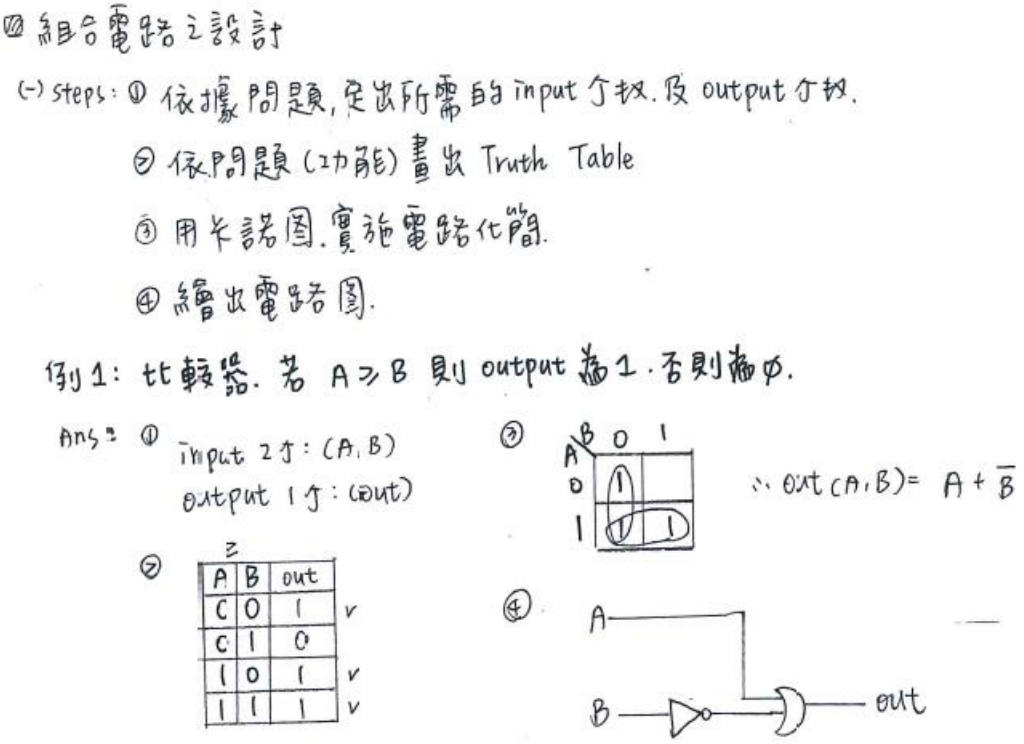
即 NAND 和 NOR gate 皆是。

7. XOR 與 XNOR 特性探討



8. 電路種類

- 組合電路(Combinational Circuit)



▨ 加法器.

(+) 分為2種
- I. 半加器 (Half Adder):
  - 輸入：A、B 2个
  - 輸出：Sum (和) 與 Cout (進位)
- II. 全加器 (Full Adder):
  - 輸入：A、B、C 3个
  - 輸出：Sum 與 Cout

I. 半加器.

+			進位.
A	B	Sum	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

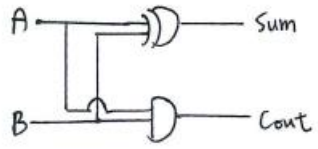
∴ Sum:

A \ B	0	1
0		1
1	1	

$Sum = \bar{A}B + A\bar{B} = A \oplus B$.

Cout: AB ※

∴ 電路:



II. 全加器.

A	B	C	Sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

∴ Sum:

A \ BC	00	01	11	10
0		1		1
1	1		1	

$Sum = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC = A \oplus B \oplus C$

Cout:

A \ BC	00	01	11	10
0			1	
1		1	1	1

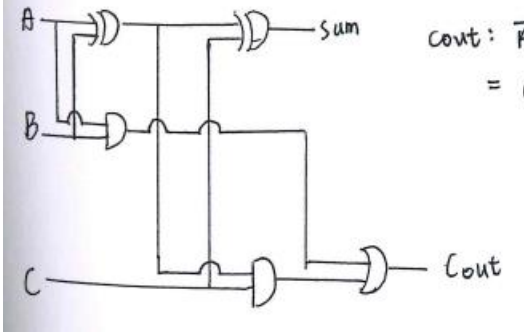
⇒ 在此不follow 卡諾圖原則. 而是要再利用先前某些 ⊕ 成果of sum.

Cout: $\bar{A}BC + A\bar{B}C + AB$

$= C \cdot (\bar{A}B + A\bar{B}) + AB = C \cdot (A \oplus B) + AB$ ※

相當於是2个半加器及1个OR.

∴ 电路图.



☑ 減法器.

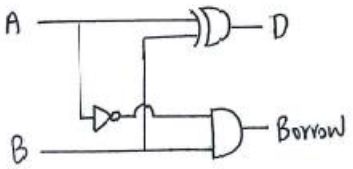
分為 { I. 半減器 { 輸入：A·B (2個) 即 A-B
輸出：D 差值 和 Borrow 下借位.
II. 全減器 { 輸入：A·B·C (3個) 即 A-B-C
輸出：D 和 Borrow

I. 半減器.

A	B	D	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

∴ $D = \bar{A}B + A\bar{B} = A \oplus B$

$Borrow = \bar{A}B$



II. 全減器.

Inputs：A·B·C 3個 (即 A-B-C)

outputs：D 及 Borrow

A	B	C	D	Borrow
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

D：

A \ BC	00	01	11	10
0		1		1
1	1		1	

$D = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC = \underline{A \oplus B \oplus C}$

Borrow：

A \ BC	00	01	11	10
0		①	1 1	①
1				

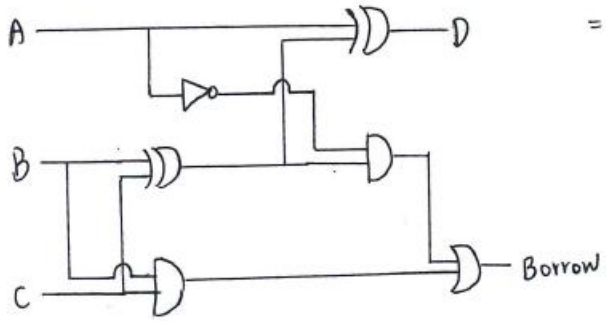
不遵守卡諾圖化簡原則，想要再利用某些 ⊕ 成果 in D

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

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

$= \underline{BC + \bar{A} \cdot (B \oplus C)}$

∴電路圖：

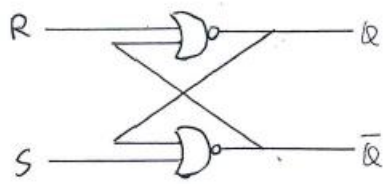


- 序向電路(Sequential Circuit)

  ⇨ 正反器(Flip-Flop)

图：以 NOR gate 为例.

R.S.为 inputs，Q为输出（$\bar{Q}$为Q之补极）



R	S	上次输出 Q	本次输出 Q.	$\bar{Q}$
1	0	0	0	1
1	0	1	0	1
1	1	0	0	0
1	1	1	0	0
0	0	0	0	1
0	0	1	1	0

} 予值，应避免之

※NOR特性：Input只要有1个为1，输出必为0

R	S	Q	$\bar{Q}$	
1	0	0	1	⇒ Reset (R)
0	1	1	0	⇒ Set (S)
0	0	Q_t	$\bar{Q_t}$	⇒ Q_t：上次输出，即维持原态，(上次输出状态) 即记忆功能
1	1	予值，不会发生，宜避免之，一般來說undifined		

- 激励表 Exciting Table

上次 Q_t	本次 Q	R	S	
1	0	1	0	⇒ 改变状态 Reset
1	1	0	X	⇒ { Set：S=1，R=0 / 维持原态：S=0，R=0
0	0	X	0	⇒ { Reset：S=0，R=1 / 维持原态：S=0，R=0
0	1	0	1	⇒ Set

R.S：inputs.

R一定要是0，S1或0不在乎

"X"：don't care.

- 比較

組合電路	序向電路
◦ 电路之輸出只與輸入的組合相関與時序無関.與之一次輸出無関	◦ 輸出會由本次輸入組合.由之次輸出結果共同決定與時序(Timing)相関.
◦ 相同的輸入組合必得相同(一致)之結果.	◦ 相同的輸入組合.不見得得到相同結果.
◦ 不具備回饋(feedback)路徑之电路.	◦ 具備
◦ 不具備記憶(memory)之能力	◦ 具備
◦ 例: 比較器. 編碼器. 加法器. 解碼器. 減法器. 計數器. 多工器. (counting) 解多工器... etc.	◦ 例: 正反器(flip-flop) 記憶体. 漣波計数器. etc.

9. Pineline(管線式)

- 公式

  ⇨ (Instruction count + (pipeline stage -1)) * Clock cycle time

  ⇨ 執行到最後一條指令時，還需等(stage – 1)個 cycle，作後一個指令才會完全做完

(一) 目的

加速多條指令之執行（縮短多條指令之執行時間）

(二) Non- pipeline

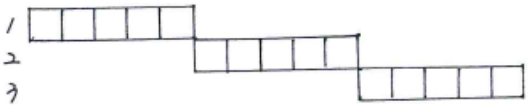
假設机器指令有5個 stages .如下：

IF	DE	EXE	FO	WB

(循序執行)

每个stage 各花 1个 clock cycle . 執行 3條指令. 共花 3*5=15个 clock cycles.

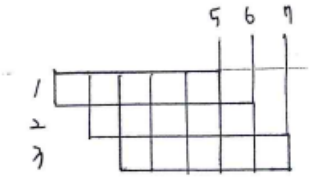
圖：



(三) pipeline 技術

不同指令之不同 stage 可以 overlay execution (重疊/平行執行)

⇒ 加速整体指令之執行.



3個指令共花7個 clock cycles.

☆ pipeline 之公式 記

令 N：指令数目. 則 pipeline time

S：指令 stage 数目. = [N+(S-1)] * MAX{各 stage 之 Time} ⇦ (Instruction count + (pipeline stage -1)) * Clock cycle time

例：机器指令有5個 stages 如下： Ans：

IF	DE	EXE	FO	WM
10ns	5ns	10ns	10ns	5ns

執行10條指令共花？ns

① Non-pipeline ② pipeline

① Non-pipeline

10*(10+5+10+10+5)=10*40=400 ns

② pipeline

[10+(5-1)] * MAX{10, 5, 10, 10, 5}

= 14*10 = 140 ns.

10. 命題演算

(一) $P \Rightarrow Q$ （唸法：若P則Q）

Def：等同於 $\sim P \vee Q$ （$\bar{P}+Q$）

（$\sim$：NOT，$\wedge$：AND，$\vee$：OR）

P	Q	P⇒Q
0	0	1
0	1	1
1	0	0
1	1	1

(二) $P \Leftrightarrow Q$ （P若且唯若Q）

Def：等同於 $(P \Rightarrow Q) \wedge (Q \Rightarrow P)$

$(\sim P \vee Q) \wedge (\sim Q \vee P)$

$= (\bar{P}+Q) \cdot (\bar{Q}+P)$

$= \bar{P}\bar{Q} + \bar{P}\cdot P + Q\bar{Q} + QP$

$= \bar{P}\bar{Q} + PQ = P \odot Q$

（互斥反或）

P	Q	P⇔Q	P⇒Q	Q⇒P	(P⇒Q)∧(Q⇒P)
0	0	1	1	1	1
0	1	0	1	0	0
1	0	0	0	1	0
1	1	1	1	1	1

一樣.

(三) Tautology 恆真.

不論 input 之組合為何，output 一律為 True (1)

Note：若以卡諾圖來看，即每一個方格均填入"1"