COMSM1302 電腦架構概論
第一週完整教材：數位邏輯基礎

布林代數 · 尋找公式 · 邏輯閘 · NAND 與功能完備性

依據 Kira Clements（University of Bristol）課程講義編寫並延伸

2026 年 7 月

本教材的使用方式：本講義整合了第一週四份投影片（1.1 Boolean Algebra、1.2 Finding Formula、1.3 Logic Gates、1.4 NAND）的全部內容，並補充了背景知識、詳細推導、歷史脈絡與延伸練習。每章結尾附有「本章重點」整理；第 5 章為綜合練習題，附完整詳解。建議先讀懂每個定義與例題，再自行完成練習題後對照解答。

目錄

1　布林代數（Boolean Algebra）

1.1　二進位表示：一切從「兩種狀態」開始

每一種數位裝置都使用**二進位（binary）**資料表示法，也就是「兩種狀態」：

狀態	邏輯意義	物理意義
1	TRUE（真）	「開（on）」，電流正在流動
0	FALSE（假）	「關（off）」，電流沒有流動

電腦內部的一切——數字、文字、圖片、程式——最終都被編碼成這兩種狀態的序列。若要研究這些訊號如何被**操作與轉換**，我們需要一套數學工具：**布林代數（Boolean algebra）**。

歷史小知識：布林代數得名於英國數學家 George Boole，他在 1854 年的著作《The Laws of Thought》中將邏輯推理系統化為代數運算。1938 年，Claude Shannon 在其碩士論文中證明布林代數可以完整描述繼電器開關電路的行為，從此奠定了數位電路設計的理論基礎——這篇論文常被譽為「二十世紀最重要的碩士論文」。

1.2　命題（Propositions）

定義

命題（proposition）是一個**非真即假**的敘述句。判斷一個敘述是否為命題的好方法：在句子前面加上「it is true that…（以下敘述為真：）」，若加上後語意通順、可以判斷真假，它就是命題。

例如，對一個三邊為 a、b、c 的三角形，下列都是命題：

命題	命題變數
邊 a 與邊 b 等長	x
邊 b 與邊 c 等長	y
邊 c 與邊 a 等長	z

每個命題可以用一個**命題變數（propositional variable）**代替，變數的值就是該命題的真假值。變數名稱可以任取，但依慣例（也為了簡潔）使用字母。

非命題的例子與其「命題化」的方式：

非命題	$\Longrightarrow$	修改成命題
「這是什麼三角形?」（疑問句）		「它是正三角形」
「邊 a + 邊 b」（只是個算式）		「邊 a + 邊 b = 邊 c」

1.3　基本邏輯運算子（Logical Operators）

三個**基本運算子（basic operators）**足以組合出任何邏輯表達式：

	$\neg A$	$A \wedge B$	$A \vee B$
正式名稱	否定（Negation / Complement）	合取（Conjunction）	析取（Disjunction）
自然語言	not A（非 A）	A and B（A 且 B）	A or B（A 或 B）
替代寫法	$\bar{A}$	$A \cdot B$	$A + B$
C 語言	`!A`	`A && B`	`A || B`

注意：$A \vee B$ 是**包含性的或（inclusive or）**——「A 或 B 或兩者皆真」時輸出為真。這與日常語言中常見的「二選一」（排他性的或）不同，後者是 $\oplus$（XOR），稍後介紹。

1.4　真值表（Truth Tables）

定義

真值表（truth table）列出一個邏輯公式**所有可能的輸入組合**與其對應的輸出值。n 個輸入變數會產生 2^n 列。

三個基本運算子的真值表：

A	B	$\neg A$
0	0	**1**
0	1	**1**
1	0	**0**
1	1	**0**

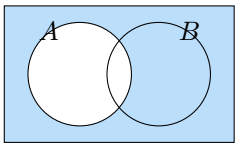
A	B	$A \wedge B$
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

A	B	$A \vee B$
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

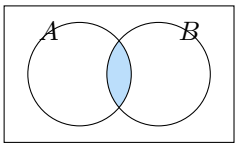
記憶要點：$\wedge$（AND）「全部為 1 才是 1」；$\vee$（OR）「至少一個 1 就是 1」。

1.5　文氏圖（Venn Diagrams）

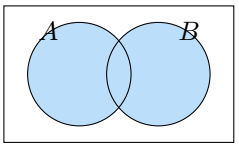
文氏圖用「面積」表達邏輯運算：矩形代表所有可能情況，圓 A 內部代表「A 為真」的情況，塗色區域代表「公式輸出為真」的情況。



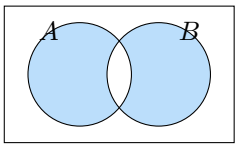
$\neg A$：A 為假的所有情況



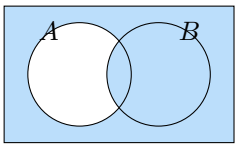
$A \wedge B$：A、B 皆真



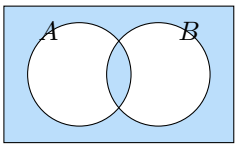
$A \vee B$：至少一者為真



$A \oplus B$：恰有一者為真



$A \to B$：排除「A 真 B 假」



$A \leftrightarrow B$：真假值相同

1.6　組合運算子：三角形的例子

回到三角形的三個命題（x：$a = b$；y：$b = c$；z：$c = a$），我們可以用邏輯運算子把它們組合成**複合命題（compound propositions）**。必要時用括號標明運算順序。

例題 1.1：三角形分類

用 x、y、z 寫出下列判斷式：

(a) 正三角形（三邊皆等長）？

(b) 不等邊三角形（沒有任何兩邊等長）？

(c) 等腰三角形（**恰好**兩邊等長）？

解答

(a) $x \wedge y \wedge z$ ——三個等長關係必須全部成立。

(b) $\neg x \wedge \neg y \wedge \neg z$ ——三個等長關係必須全部不成立。

(c) $(x \vee y \vee z) \wedge \neg(x \wedge y \wedge z)$ ——「至少有一組邊等長」**且**「不是三邊全等長」。這裡有個微妙之處：對實際的三角形而言，若 x、y、z 中任兩個為真，第三個必然為真（等長關係有遞移性），所以「至少一組等長且非全部等長」就等價於「恰好一組等長」。

1.7　更多邏輯運算子

除了三個基本運算子外，還有三個常用運算子。重要的是：**它們都可以用基本運算子表達**。

	$A \oplus B$	$A \to B$	$A \leftrightarrow B$
正式名稱	互斥析取（XOR）	蘊含（Implication）	等價（Equivalence）
自然語言	exclusively A or B	A implies B	A is equivalent to B
基本運算子表示	$(\neg A \vee \neg B) \wedge (A \vee B)$	$\neg A \vee B$	$(\neg A \vee B) \wedge (\neg B \vee A)$

六個運算子的完整真值表：

A	B	$\neg A$	$A \wedge B$	$A \vee B$	$A \oplus B$	$A \to B$	$A \leftrightarrow B$
0	0	1	0	0	0	1	1
0	1	1	0	1	1	1	0
1	0	0	0	1	1	0	0
1	1	0	1	1	0	1	1

幾個幫助理解的觀察：

- **XOR（$\oplus$）**：「恰好一個為真」。也可以想成「$A \neq B$」或「不進位的二進位加法」。$A \oplus B \equiv (A \vee B) \wedge \neg(A \wedge B)$ 是另一個常見的等價寫法。
- **蘊含（$\to$）**：$A \to B \equiv \neg A \vee B$。唯一為假的情形是「$A$ 真而 B 假」。注意當 A 為假時整個式子必為真（前提不成立，承諾自動成立）。用自然語言帶入命題有助於直覺：「如果下雨（A），那麼地會濕（B）」——沒下雨時，不論地濕不濕，這句話都沒有被違反。
- **等價（$\leftrightarrow$）**：$A \leftrightarrow B \equiv (A \to B) \wedge (B \to A)$，即「互相蘊含」。輸出為真恰好在 A、B 真假值相同時；它正是 XOR 的否定：$A \leftrightarrow B \equiv \neg(A \oplus B)$。

1.8　完成複雜公式的真值表

要計算複雜布林表達式的真值表，只要**把每個變數換成該列的輸入值，然後逐步化簡**。

例題 1.2：計算 $(A \wedge B) \vee \neg C$ 的真值表

逐列代入並化簡：

A	B	C	代入化簡過程	輸出
0	0	0	$((0 \wedge 0) \vee \neg 0) \equiv (0 \vee 1)$	1
0	0	1	$((0 \wedge 0) \vee \neg 1) \equiv (0 \vee 0)$	0
0	1	0	$((0 \wedge 1) \vee \neg 0) \equiv (0 \vee 1)$	1
0	1	1	$((0 \wedge 1) \vee \neg 1) \equiv (0 \vee 0)$	0
1	0	0	$((1 \wedge 0) \vee \neg 0) \equiv (0 \vee 1)$	1
1	0	1	$((1 \wedge 0) \vee \neg 1) \equiv (0 \vee 0)$	0
1	1	0	$((1 \wedge 1) \vee \neg 0) \equiv (1 \vee 1)$	1
1	1	1	$((1 \wedge 1) \vee \neg 1) \equiv (1 \vee 0)$	1

1.9　邏輯等價（Logical Equivalence）

定義

每個邏輯公式**恰有一張**真值表與之對應；但每張真值表卻有**無限多個**邏輯公式與之對應。真值表相同的公式稱為**邏輯等價（logically equivalent）**的公式，記作 $\equiv$。

例如下列真值表（即 OR）：

A	B	輸出
0	0	0
0	1	1
1	0	1
1	1	1

對應公式包括：

$$A \lor B$$
$$\neg(\neg A \land \neg B)$$
$$B \lor A$$
$$\neg(\neg A \land \neg B) \land (A \lor \neg A)$$

真值表有兩大用途：

1. **驗證邏輯等價**：兩個公式的真值表逐列比對，全部相同即等價。
2. **描述未知公式的行為**：先列出所有輸入組合的期望輸出，再想辦法找出符合的公式——這正是第 2 章的主題。

1.10　布林代數定律總表（延伸補充）

以下定律都可以用真值表逐一驗證，之後化簡公式時會反覆用到。注意每條定律都有 $\land$ 與 $\lor$ 的**對偶**版本：

布林代數基本定律

定律名稱	$\land$ 版本	$\lor$ 版本
同一律（Identity）	$A \land 1 \equiv A$	$A \lor 0 \equiv A$
支配律（Domination）	$A \land 0 \equiv 0$	$A \lor 1 \equiv 1$
冪等律（Idempotent）	$A \land A \equiv A$	$A \lor A \equiv A$
交換律（Commutative）	$A \land B \equiv B \land A$	$A \lor B \equiv B \lor A$
結合律（Associative）	$(A \land B) \land C \equiv A \land (B \land C)$	$(A \lor B) \lor C \equiv A \lor (B \lor C)$
分配律（Distributive）	$A \land (B \lor C) \equiv (A \land B) \lor (A \land C)$	$A \lor (B \land C) \equiv (A \lor B) \land (A \lor C)$
吸收律（Absorption）	$A \land (A \lor B) \equiv A$	$A \lor (A \land B) \equiv A$
互補律（Complement）	$A \land \neg A \equiv 0$	$A \lor \neg A \equiv 1$
德摩根定律（De Morgan）	$\neg(A \land B) \equiv \neg A \lor \neg B$	$\neg(A \lor B) \equiv \neg A \land \neg B$
雙重否定（Double negation）	$\neg\neg A \equiv A$	

1.11　為什麼要學這些?

布林邏輯遍布所有程式語言。學會建構與**化簡**布林表達式，能幫助你把真實問題轉譯成簡潔的程式。例如：

```
// 化簡前：巢狀條件，難讀又容易出錯
if (isWeekend) {
    if (!isRaining) { goHiking(); }
} else {
    if (isHoliday && !isRaining) { goHiking(); }
}
```

```
// 用布林代數化簡：(W && !R) || (!W && H && !R)
//   = !R && (W || (!W && H))      [提出公因子 !R]
//   = !R && (W || H)              [吸收律變形：W || (!W && H) = W || H]
if (!isRaining && (isWeekend || isHoliday)) { goHiking(); }
```

在硬體層面，化簡表達式代表**更少的邏輯閘**、更低的能耗與更快的運算（見第 3 章）。

本章重點

- 數位裝置以二進位（1/TRUE/通電、0/FALSE/斷電）表示資料。
- 命題是非真即假的敘述；命題變數承載其真假值。
- 基本運算子：$\neg$、$\wedge$、$\vee$；衍生運算子：$\oplus$、$\rightarrow$、$\leftrightarrow$ 都能用基本運算子改寫。
- 真值表是公式行為的完整描述：一式一表，但一表多式（邏輯等價）。
- 德摩根定律與分配律等基本定律是化簡與證明等價的核心工具。

2　尋找公式（Finding Formula）

上一章結尾提出了關鍵問題：**給定一張期望輸出的真值表，如何產生對應的邏輯公式？**本章介紹兩種系統化方法（DNF 與 CNF），以及兩種化簡工具（代數定律與卡諾圖）。

2.1　析取範式（DNF, Disjunctive Normal Form）

定義

析取範式（DNF）：由一個或多個「合取項」（以 $\wedge$ 連接的變數組）用 $\vee$ 連接而成的公式，其中 $\neg$ 只能直接出現在變數前面。形式上：$(\cdots \wedge \cdots) \vee (\cdots \wedge \cdots) \vee \cdots$

方法：針對真值表中每一個**輸出為 1** 的列：

1. 把該列的輸入變數用 $\wedge$ 連接（值為 0 的變數前面加 $\neg$），構成一個合取項——此項**只在**該列的輸入組合下為真；
2. 把所有列的合取項用 $\vee$ 連接起來。

例題 2.1：從真值表求 DNF

A	B	C	$?$
0	0	0	**1**
0	0	1	**1**
0	1	0	**1**
0	1	1	0
1	0	0	**1**
1	0	1	**1**
1	1	0	0
1	1	1	0

輸出為 1 的列共五列，每列各產生一個合取項：

$$? \equiv (\neg A \wedge \neg B \wedge \neg C) \vee (\neg A \wedge \neg B \wedge C) \vee (\neg A \wedge B \wedge \neg C) \vee (A \wedge \neg B \wedge \neg C) \vee (A \wedge \neg B \wedge C)$$

這就是 DNF——正確，但看起來非常複雜！

2.2　合取範式（CNF, Conjunctive Normal Form）

定義

合取範式（CNF）：由一個或多個「析取項」（以 $\vee$ 連接的變數組）用 $\wedge$ 連接而成的公式，其中 $\neg$ 只能直接出現在變數前面。形式上：$(\cdots \vee \cdots) \wedge (\cdots \vee \cdots) \wedge \cdots$

方法：針對每一個**輸出為 0** 的列：

1. 把該列的輸入變數用 $\wedge$ 連接（值為 0 的變數加 $\neg$）後**整體取否定**——意思是「排除這一種輸入組合」；
2. 把所有這些否定項用 $\wedge$ 連接。

沿用例題 2.1 的真值表，輸出為 0 的列只有三列（011、110、111）：

$$? \equiv \neg(\neg A \wedge B \wedge C) \wedge \neg(A \wedge B \wedge \neg C) \wedge \neg(A \wedge B \wedge C)$$

因為 0 比 1 少，這個公式比 DNF 短——但它**還不是** CNF：$\neg$ 出現在整個括號前，而不是只在變數前。

經驗法則：真值表中 1 少就用 DNF、0 少就用 CNF，得到的初始公式較短。

2.3　用德摩根定律完成 CNF

德摩根定律（De Morgan's Laws）

$$\neg(A \wedge B) \equiv \neg A \vee \neg B \qquad \neg(A \vee B) \equiv \neg A \wedge \neg B$$

口訣：**否定往括號裡推：$\wedge$ 與 $\vee$ 互換，每個變數加上／去掉否定**。定律可直接推廣到多個變數，例如 $\neg(A \wedge B \wedge C) \equiv \neg A \vee \neg B \vee \neg C$。

把德摩根定律套用到上面三個項：

$$\begin{aligned} \neg(\neg A \wedge B \wedge C) &\equiv (A \vee \neg B \vee \neg C) \\ \neg(A \wedge B \wedge \neg C) &\equiv (\neg A \vee \neg B \vee C) \\ \neg(A \wedge B \wedge C) &\equiv (\neg A \vee \neg B \vee \neg C) \end{aligned}$$

於是得到真正的 CNF：

$$? \equiv (A \vee \neg B \vee \neg C) \wedge (\neg A \vee \neg B \vee C) \wedge (\neg A \vee \neg B \vee \neg C)$$

2.4　用分配律化簡

有了兩種「找公式」的方法之後，下一個任務是**化簡**。分配律是理想工具：

分配律（Distributivity）

$$(A \wedge B) \vee (A \wedge C) \equiv A \wedge (B \vee C) \qquad (A \vee B) \wedge (A \vee C) \equiv A \vee (B \wedge C)$$

類比於數學中的**提出公因式**；反方向使用則是**展開**。

例題 2.2：化簡 CNF

把剛才的 CNF 一步步化簡（每一步右側註明使用的定律）：

$$
\begin{aligned}
& (A \vee \neg B \vee \neg C) \wedge (\neg A \vee \neg B \vee C) \wedge (\neg A \vee \neg B \vee \neg C) \\
\equiv\ & \neg B \vee \big((A \vee \neg C) \wedge (\neg A \vee C) \wedge (\neg A \vee \neg C)\big) \quad \text{【三項都含 } \neg B\text{，用分配律提出】} \\
\equiv\ & \neg B \vee \big((A \vee \neg C) \wedge (\neg A \vee (C \wedge \neg C))\big) \quad \text{【後兩項提出 } \neg A\text{】} \\
\equiv\ & \neg B \vee \big((A \vee \neg C) \wedge \neg A\big) \quad \text{【}C \wedge \neg C \equiv 0\text{（互補律），}\neg A \vee 0 \equiv \neg A\text{（同一律）】} \\
\equiv\ & \neg B \vee \big((A \wedge \neg A) \vee (\neg C \wedge \neg A)\big) \quad \text{【分配律展開，讓變數有機會相消】} \\
\equiv\ & \neg B \vee (\neg C \wedge \neg A) \quad \text{【}A \wedge \neg A \equiv 0\text{，}0 \vee X \equiv X\text{】}
\end{aligned}
$$

最終結果：$\boxed{\neg B \vee (\neg A \wedge \neg C)}$，遠比原本的五項 DNF 或三項 CNF 精簡。

驗算：把化簡後的公式代回真值表，確認與原表一致：

A	B	C	$\neg B \vee (\neg A \wedge \neg C)$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

✓ 與例題 2.1 完全一致

德摩根定律與分配律也可用來**證明邏輯等價**：若能把一個公式一步步變形成另一個，兩者必然等價。但這套代數方法有個缺點：**很難判斷是否已化到最簡**——這正是卡諾圖要解決的問題。

2.5　卡諾圖（Karnaugh Maps）

定義

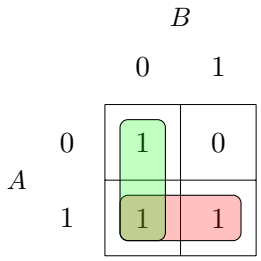
卡諾圖（Karnaugh map，K-map）是把真值表重新排列成二維格子的工具，用「圈選相鄰的 1」找出邏輯表達式的**最簡形式（minimal form）**。

卡諾圖由貝爾實驗室的 Maurice Karnaugh 於 1953 年提出，是 Veitch 圖的改良版。它把「代數化簡」轉化成「視覺圖形辨識」，在 4 個變數以內特別好用。

建構方式：把真值表的輸出填入格子，列與欄分別由部分輸入變數索引。以兩變數為例：

A	B	?
0	0	1
0	1	0
1	0	1
1	1	1

$\Longrightarrow$



求最簡式的步驟：

1. 把相鄰的 1 圈成邊長為 2^n（1、2、4、8……）的**矩形方框**；
   - 方框**可以跨越邊界繞回**（wrap around）——上下邊相連、左右邊相連；
   - 方框**可以（且應該）重疊**——框越大，化出的項越簡單；
   - 不允許對角線圈選；方框內不可含 0。
2. 對每個方框：把「框內**值不變**的變數」用 $\wedge$ 連接（值恆為 0 的變數加 $\neg$）；
3. 把所有方框的表達式用 $\vee$ 連接。

上例中：下排整列（$A=1$，框內 B 有變、A 不變）給出項 A；左欄整欄（$B=0$）給出項 $\neg B$。答案：$A \vee \neg B$。

2.6　格雷碼（Gray Code）

當一側有兩個以上變數時，輸入值必須用**二進位反射格雷碼（binary-reflected Gray code）**排列：**相鄰的值只差一個位元**。

定義

格雷碼生成法（講義版）：從全 0 開始，每次**翻轉最右邊的、能產生新字串的位元**：

$$00 \to 01 \to 11 \to 10$$

（另一個等價方法是「反射法」：把 n 位元格雷碼上下鏡射，上半段前面補 0、下半段前面補 1，就得到 $n+1$ 位元格雷碼。）

為什麼一定要用格雷碼？卡諾圖的化簡原理是布林恆等式 $(X \wedge Y) \vee (X \wedge \neg Y) \equiv X$：兩個「只差一個變數」的項可以合併並消去那個變數。格雷碼保證**圖上相鄰的格子恰好只差一個位元**，所以「圈起相鄰格子」就等於「用代數合併相鄰項」。若欄位照一般二進位順序 $00, 01, 10, 11$ 排列，01 與 10 差兩個位元，圖就失效了。

2.7　四變數卡諾圖範例

例題 2.3：四變數 K-map（講義 Example 1）

列以 AB 索引、欄以 CD 索引，皆用格雷碼順序 $00, 01, 11, 10$：

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

三個方框的讀法：

- **左欄整欄**（4 格）：框內 A、B 都有變化，$C = 0$、$D = 0$ 不變 $\Rightarrow \neg C \wedge \neg D$；
- **中間 2×2 方塊**（$AB \in \{01, 11\}$、$CD \in \{00, 01\}$）：$B = 1$、$C = 0$ 不變 $\Rightarrow B \wedge \neg C$；
- **第一列左右兩端**（wrap around 繞回相接，2 格）：$A = 0$、$B = 0$、$D = 0$ 不變 $\Rightarrow \neg A \wedge \neg B \wedge \neg D$。

最簡式：

$$(\neg C \wedge \neg D) \vee (B \wedge \neg C) \vee (\neg A \wedge \neg B \wedge \neg D)$$

例題 2.4：用 K-map 驗證例題 2.2（講義 Example 2）

回到例題 2.1 ／ 2.2 的三變數真值表（1 的位置：$000, 001, 010, 100, 101$），列以 A 索引、欄以 BC 索引：

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

- **左邊 2×2 方塊**（$BC \in \{00, 01\}$、兩列）：只有 $B = 0$ 不變 $\Rightarrow \neg B$；
- **第一列兩端**（$BC = 00$ 與 $BC = 10$，繞回相接）：$A = 0$、$C = 0$ 不變 $\Rightarrow \neg A \wedge \neg C$。

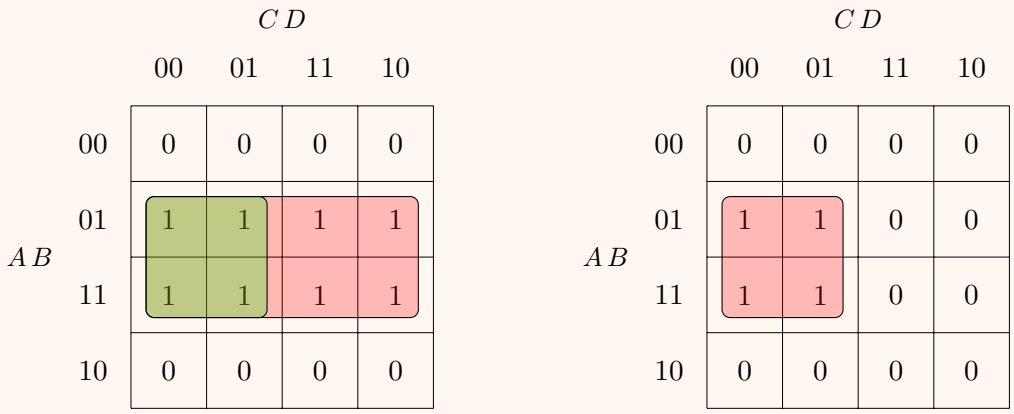
最簡式：$\neg B \vee (\neg A \wedge \neg C)$ ——與例題 2.2 用代數化簡得到的結果完全相同，但 K-map 讓我們

確信這已是最簡形式!

2.8　五個以上的變數

五變數最常見的做法：畫**兩張四變數卡諾圖**，分別代表第五個變數（E）的兩個值，然後在兩張圖之間「對齊／疊合」方框來化簡邏輯。

例題 2.5：五變數 K-map



$E = 0$　　$E = 1$

- $E = 0$ 圖中，中間兩列全為 1：$B = 1$ 不變、且此框只存在於 $E = 0 \Rightarrow B \wedge \neg E$；
- **兩張圖都有**的 2×2 方塊（$B = 1$、$C = 0$）可以「疊合」：因為它同時出現在 $E = 0$ 與 $E = 1$，變數 E 被消去 $\Rightarrow B \wedge \neg C$。

最簡式：$(B \wedge \neg E) \vee (B \wedge \neg C)$。（還可再用分配律寫成 $B \wedge (\neg E \vee \neg C)$。）

本章重點

- **DNF**：對每個輸出 1 的列做 $\wedge$ 項，再全部 $\vee$ 起來；**CNF**：對每個輸出 0 的列做「排除項」，再全部 $\wedge$ 起來（需德摩根整理）。
- 德摩根定律：否定推進括號時 $\wedge \leftrightarrow \vee$ 互換。
- 分配律 = 提公因式／展開，是代數化簡的主力；但代數法難以確認「已達最簡」。
- K-map：格雷碼排列 + 圈 2^n 大小的方框（可繞回、可重疊）$\Rightarrow$ 系統性地找出最簡式。
- 5 變數：兩張 4 變數圖疊合；更多變數則需要其他演算法（如 Quine–McCluskey 法，屬延伸閱讀）。

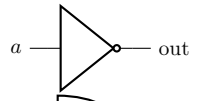
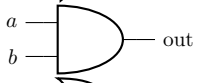
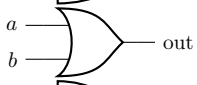
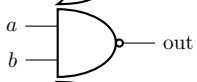
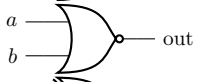
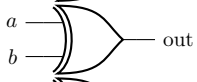
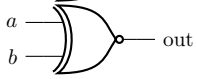
3　邏輯閘（Logic Gates）

3.1　什麼是邏輯閘?

定義

邏輯閘（logic gate）是實作單一簡單布林函數（例如 $A \wedge B$）的元件。它是邏輯函數的**抽象視圖**：目前我們採取**黑盒子（black box）**觀點——只關心它的輸入輸出行為，不深究內部如何運作（實體上它們由電晶體構成，本課程稍後會再回到這個主題）。

3.2　閘符號一覽

閘	符號	布林式	說明	輸出（輸入 ab）			
				00	01	10	11
NOT		$\neg a$	反相器	$a=0 \Rightarrow 1$；			$a=1 \Rightarrow 0$
AND		$a \wedge b$	全 1 才 1	0	0	0	1
OR		$a \vee b$	有 1 就 1	0	1	1	1
NAND		$\neg(a \wedge b)$	NOT-AND	1	1	1	0
NOR		$\neg(a \vee b)$	NOT-OR	1	0	0	0
XOR		$a \oplus b$	恰一個 1	0	1	1	0
XNOR		$\neg(a \oplus b)$	相同才 1	1	0	0	1

符號記憶要訣：

- **AND** 是平頭 D 形；**OR** 是彎月尖頭形；
- 輸出端的**小圓圈（bubble）**代表「取否定」：AND + 圓圈 = NAND，OR + 圓圈 = NOR，XOR + 圓圈 = XNOR；
- **XOR** 在 OR 的輸入側多一道弧線。

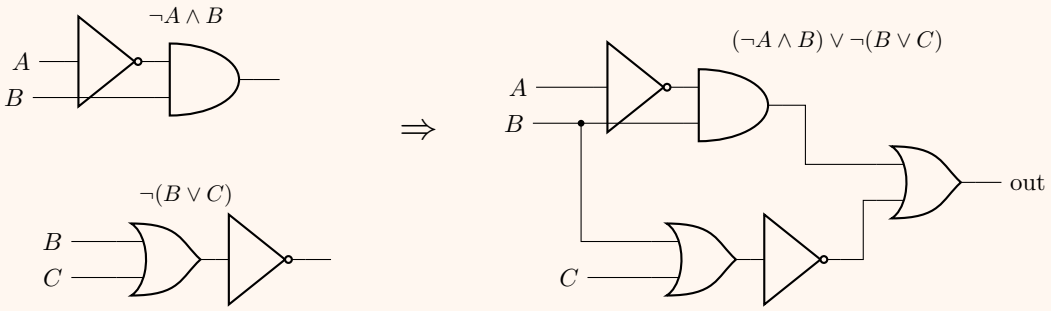
NAND 與 NOR 特別重要：它們各自都是**功能完備 (functionally complete)** 的——任何布林表達式都可以只用其中一種閘實作出來，因此是絕佳的建構基石（詳見第 4 章）。XOR 與 XNOR 較不常用（它們容易由前面的閘組合出來），但仍需能夠辨認。

3.3　用電路實作邏輯

把邏輯表達式中的**每個運算子換成對應的閘**，即可建構出實作該式的電路。複雜電路可由簡單電路組合而成；一條訊號線可以**分支（branch）**複製訊號送往多處。

例題 3.1：實作 $(\neg A \wedge B) \vee \neg(B \vee C)$

先分別實作兩個子電路，再組合：



注意 B 的訊號在進入 AND 閘之前**分支**（圖中的實心圓點），同一份訊號同時送給 AND 閘與 OR 閘——不需要兩個 B 輸入。

3.4　最佳化：用最少的閘

同一個邏輯函數可以有很多種電路實作，我們追求**用最少的閘**：

閘越少 ＝ 能耗越低、運算越快！

比較下列兩個**輸出完全相同**的電路：

$$(\neg A \vee (\neg B \wedge \neg C)) \wedge (B \vee (\neg B \wedge \neg C)) \qquad (\neg A \wedge B) \vee \neg(B \vee C)$$

共 8 個閘（若不分支則更多）　　只需 4 個閘

左邊的電路有兩大問題：

1. **公式本身沒化簡**：$(\neg A \vee (\neg B \wedge \neg C)) \wedge (B \vee (\neg B \wedge \neg C))$ 可用分配律（把 $\neg B \wedge \neg C$ 視為一個整體提出）化簡成 $(\neg A \wedge B) \vee (\neg B \wedge \neg C)$，再用德摩根把 $\neg B \wedge \neg C$ 改寫成 $\neg(B \vee C)$——後者只需 2 個閘（OR + NOT），比前者（2 個 NOT + 1 個 AND）省 1 個閘。
2. **重複的閘沒有共用**：$(\neg B \wedge \neg C)$ 出現兩次，卻蓋了兩份相同的子電路——把第一份的輸出**分支**即可，能再省 3 個閘。

實作流程建議：先用第 2 章的方法（代數定律或 K-map）把公式化到最簡，再畫電路；畫完後檢查有沒有重複的子電路可以用分支共用。

3.5　Logisim：電路模擬工具

實體電路（尤其大規模生產時）非常昂貴，因此業界在實作前都會用**模擬工具**驗證設計。本課程使用教育用模擬器 **Logisim** 設計並驗證電路。重點操作整理：

功能	說明
放置邏輯閘	左側探索窗格（explorer pane）的 **Gates** 資料夾；常用閘也放在上方工具列。先放好所有閘，搭出電路「骨架」。
接線	用**編輯工具（edit tool）**從元件接點拖曳到另一點；拖曳既有導線可延長、縮短或分支。
輸入／輸出腳位	工具列上有**輸入腳位（方形）**與**輸出腳位（圓形）**，接到電路的對應導線上。
測試	用**戳戳工具（poke tool）**點擊輸入腳位切換 0/1，觀察輸出是否符合預期。
屬性表	左下方的屬性表（attributes table）可修改元件設定：閘的輸入數量、元件標籤、元件方向、腳位為輸入或輸出等。
子電路	**Project > Add Circuit...** 建立新電路；在探索窗格**單擊**可把自訂電路當元件使用，**雙擊**可編輯它。也能編輯元件外觀、重新排列輸入輸出腳位。

導線顏色的意義（除錯時非常重要）：

顏色	意義
藍色	**未知**的 1 位元值（通常表示尚未接上輸入）
深綠色	值為 **0** 的 1 位元訊號
淺綠色	值為 **1** 的 1 位元訊號
紅色	**錯誤**（例如兩個輸出互相衝突、短路）

完整的 Logisim 使用手冊：`http://www.cburch.com/logisim/docs.html`

本章重點

- 邏輯閘 = 單一布林函數的黑盒子實作；輸出端小圓圈代表否定。
- 七個基本閘：NOT、AND、OR、NAND、NOR、XOR、XNOR；NAND 與 NOR 是功能完備的。
- 電路實作 = 運算子逐一換成閘；訊號可分支共用。
- 目標是最少閘數：先化簡公式，再共用重複的子電路。
- Logisim 用於實作前的設計驗證；記住四種導線顏色的意義。

4 NAND 與功能完備性

4.1 功能完備（Functionally Complete）

定義

一組運算子（或閘）是**功能完備（functionally complete）**的，若任何布林表達式都能只用這組運算子表達。NAND **單獨一個**就是功能完備的——任何電路都可以只用 NAND 閘重新實作！

這帶來重大的實務優勢：**只需要量產一種閘**，就能製造任何數位電路，使實體生產更有效率。

NAND（與 NOR）的功能完備性由 Henry M. Sheffer 於 1913 年在數學上證明，因此 NAND 運算也稱為「Sheffer stroke」。NOR 同樣功能完備，但**業界標準是 NAND**，因為其電晶體結構速度更快：在 CMOS 製程中，NAND 把（較快的）nMOS 電晶體串聯、（較慢的）pMOS 並聯，NOR 則相反，所以 NAND「結構上比 NOR 快」。知名教材《The Elements of Computing Systems》（nand2tetris）正是從單一個 NAND 閘出發，一路蓋出一台完整的電腦。

4.2 雙重否定：加負號的自由

雙重否定（Double Negation）

$$\neg\neg A \equiv A$$

兩個否定互相抵消。反過來用：可以**隨時為任何表達式加上一對否定**而不改變其意義。

雙重否定與德摩根定律結合，就成為把任何布林函數翻譯成「純 NAND 表達式」的強大工具。

例題 4.1：OR 閘 = 輸入反相的 NAND

$$A \vee B \xRightarrow{\text{雙重否定}} \neg\neg(A \vee B) \xRightarrow{\text{德摩根}} \neg(\neg A \wedge \neg B)$$

最外層的 $\neg(\cdots\wedge\cdots)$ 正是一個 NAND！所以 OR 閘等價於「兩個輸入都先反相的 NAND 閘」。

例題 4.2：AND 閘 = 輸出反相的 NAND

$$A \wedge B \equiv \neg\neg(A \wedge B)$$

內層的 $\neg(A\wedge B)$ 是 NAND，外層再加一個 NOT——即 AND 等價於「NAND 接一個反相器」。

4.3 NOT 也是 NAND

上面兩個實作都還含有「額外的」否定。沒關係——**否定本身就是 NAND**：把同一個輸入接到 NAND 的兩隻腳即可：

$$\neg A \equiv \neg(A \wedge A)$$

（因為 $A \wedge A \equiv A$，冪等律。）

因此例題 4.2 的 AND 若要「明確地」全用 NAND 寫出，就是：

$$A \wedge B \equiv \neg\big(\neg(A \wedge B) \wedge \neg(A \wedge B)\big)$$

三個基本閘的 NAND 實作總覽：

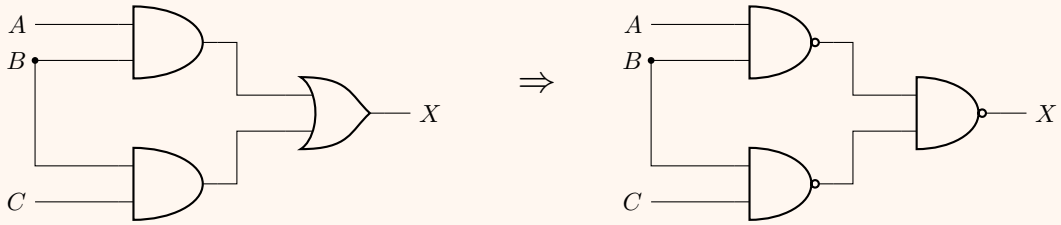
目標	純 NAND 寫法	需要的 NAND 數
$\neg A$	$\neg(A \wedge A)$	1
$A \wedge B$	$\neg(\neg(A \wedge B) \wedge \neg(A \wedge B))$	2
$A \vee B$	$\neg(\neg(A \wedge A) \wedge \neg(B \wedge B))$	3

數 NAND 的技巧：把表達式整理成「否定只以 NAND 形式出現」之後，**數否定符號的個數**就是所需 NAND 閘的數量。每個 $\neg(\cdots \wedge \cdots)$ 是一個 NAND；單獨的 $\neg X$ 是一個「輸入複製」的 NAND。

例題 4.3：把 $(A \wedge B) \vee (B \wedge C)$ 轉成純 NAND 電路

$$\begin{aligned} & (A \wedge B) \vee (B \wedge C) \\ \equiv\ & \neg\neg\big((A \wedge B) \vee (B \wedge C)\big) \quad \text{【雙重否定】} \\ \equiv\ & \neg\big(\neg(A \wedge B) \wedge \neg(B \wedge C)\big) \quad \text{【德摩根】} \end{aligned}$$

數一數否定：外層一個 $\neg(\cdots\wedge\cdots)$、內層兩個 $\neg(\cdots\wedge\cdots)$ ——共 **3 個 NAND**。電路如下（原電路 2 AND + 1 OR，NAND 版同樣 3 個閘）：



4.4　推泡泡（Bubble Pushing）

從**視覺**的角度看 NAND 實作：把每個否定視為一顆**泡泡（bubble）**——就是 NOT、NAND、NOR 閘尖端的小圓圈。

定義

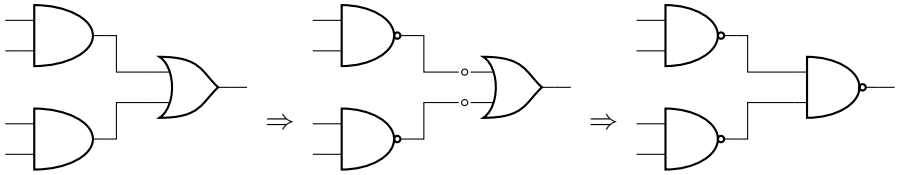
推泡泡（bubble pushing）是把德摩根定律**直接套用在電路圖上**的技巧：

$$\underbrace{\neg(A \wedge B) \equiv \neg A \vee \neg B}_{\text{NAND} \equiv \text{輸入反相的 OR}} \qquad \underbrace{\neg(A \vee B) \equiv \neg A \wedge \neg B}_{\text{NOR} \equiv \text{輸入反相的 AND}}$$

把泡泡從閘的輸出端「推」到輸入端（或反向）時，$\wedge$ 閘與 $\vee$ 閘互換。

圖示：NAND 的兩種畫法（左）與 NOR 的兩種畫法（右）：

使用方法：已有一個只含 AND、OR、NOT 的化簡電路時，**從輸出端開始**往回走，在適當處推泡泡或成對加泡泡（雙重否定），即可合成等價的 NAND 電路。以 $(A \wedge B) \vee (C \wedge D)$ 為例：



步驟解讀：(1) 在兩個 AND 的輸出與 OR 的輸入之間**各加一對泡泡**（雙重否定，不改變功能）——AND+ 泡泡成為 NAND；(2)「輸入帶泡泡的 OR」依德摩根定律**就是 NAND**，直接改畫成 NAND。完成：3 個 NAND，與例題 4.3 的代數推導一致。

4.5　NAND 電路的化簡

NAND 化最好**在公式階段先化簡**；但電路畫好後仍有兩招可用：

1. **移除連續兩個作為雙重否定的 NAND**：兩個串接的「NOT 型 NAND」（輸入綁在一起）互相抵消，可整段拿掉。
2. **合併作用相同的 NAND**：兩個輸入完全相同的 NAND 是重複的——留一個，把它的輸出**分支**給所有用到的地方。

4.6　NAND 板（NAND Boards）：實驗室硬體

實驗課將把電路設計轉移到實體的 **NAND 板**上。板子的構造：

元件	說明
4 個輸入開關	每個開關未按下時輸出 0；各附 **4 個複製輸出腳位**，方便把同一輸入送到多個閘。
16 個 NAND 閘	每個 NAND 有 A、B 輸入腳位（各附複製腳位）、一個顯示輸出值的 **LED**，以及 **4 個複製輸出腳位**。
16 個常數 1 腳位	提供恆為 1 的訊號（例如把 NAND 當 NOT 用時可派上用場：$\neg(A \wedge 1) \equiv \neg A$）。

實作忠告：上板子之前**先在紙上規劃好**哪個邏輯 NAND 對應板上哪個實體 NAND、訊號怎麼走線——否則很快就會變成一團理不清的電線。建議先畫好 NAND 電路圖並為每個閘編號（NAND 1、NAND 2、…），再對應到板上的實體位置。第一次實驗課前請先觀看 NAND 板介紹影片。

本章重點

- NAND 功能完備：$\neg A \equiv \neg(A \wedge A)$（1 個），$A \wedge B$（2 個），$A \vee B$（3 個）。
- 雙重否定讓我們「免費」加負號；德摩根把 $\neg(\vee)$ 與 $\neg(\wedge)$ 互換——兩者聯手可把任何公式改寫成純 NAND 形式。
- 整理好的公式中，**否定的個數 = 所需 NAND 的個數**。
- 推泡泡 = 在電路圖上做德摩根：泡泡穿過閘時 AND $\leftrightarrow$ OR 互換。
- NAND 是業界標準（比 NOR 結構上更快）；實驗課的 NAND 板有 4 個開關、16 個 NAND、16 個常數 1 腳位。

5　綜合練習題（附詳解）

練習 1：命題判斷

下列哪些是命題？(a)「$3+4=8$」　(b)「現在幾點？」　(c)「把門關上。」　(d)「所有偶數都能被 2 整除。」　(e)「$x+1$」

解答

(a) ✓ 是命題（雖然是假的——命題不必為真，只需能判斷真假）。(b) ✗ 疑問句無真假值。(c) ✗ 祈使句無真假值。(d) ✓ 是命題（且為真）。(e) ✗ 只是運算式，沒有斷言任何事；改成「$x+1=5$」才是（在 x 給定後的）命題。

練習 2：用真值表驗證 $A \to B \equiv \neg A \vee B$

解答

A	B	$A \to B$	$\neg A$	$\neg A \vee B$
0	0	1	1	1
0	1	1	1	1
1	0	0	0	0
1	1	1	0	1

第 3 欄與第 5 欄完全相同，故兩式邏輯等價。✓

練習 3：完成真值表

計算 $\neg(A \oplus B) \vee C$ 的完整真值表。

解答

先算 $A \oplus B$，取否定後再與 C 做 OR：

A	B	C	$A \oplus B$	$\neg(A \oplus B)$	$\neg(A \oplus B) \vee C$
0	0	0	0	1	1
0	0	1	0	1	1
0	1	0	1	0	0
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	0	1	1
1	1	1	0	1	1

技巧：$\neg(A \oplus B)$ 就是 XNOR——「A、B 相同」時為 1。

練習 4：DNF 與 CNF

給定真值表：輸出在 $(A,B)=(0,1)$ 與 $(1,0)$ 時為 1，其餘為 0。分別寫出 DNF 與 CNF。

解答

（這正是 XOR！）

DNF（看輸出 1 的列）：$(\neg A \wedge B) \vee (A \wedge \neg B)$

CNF（看輸出 0 的列，即 00 與 11）：

$$\neg(\neg A \wedge \neg B) \wedge \neg(A \wedge B) \xrightarrow{\text{德摩根}} (A \vee B) \wedge (\neg A \vee \neg B)$$

兩式都等價於 $A \oplus B$——與第 1 章「$(\neg A \vee \neg B) \wedge (A \vee B)$」的寫法一致（交換律）。

練習 5：代數化簡

用布林代數定律化簡 $(A \wedge B) \vee (A \wedge \neg B) \vee (\neg A \wedge B)$。

解答

$$\begin{aligned}
& (A \wedge B) \vee (A \wedge \neg B) \vee (\neg A \wedge B) \\
\equiv\ & \big(A \wedge (B \vee \neg B)\big) \vee (\neg A \wedge B) \quad \text{【前兩項提出 } A \text{（分配律）】} \\
\equiv\ & A \vee (\neg A \wedge B) \quad \text{【互補律 } B \vee \neg B \equiv 1\text{；同一律 } A \wedge 1 \equiv A\text{】} \\
\equiv\ & (A \vee \neg A) \wedge (A \vee B) \quad \text{【}\vee \text{ 對 } \wedge \text{ 的分配律】} \\
\equiv\ & A \vee B \quad \text{【互補律；同一律】}
\end{aligned}$$

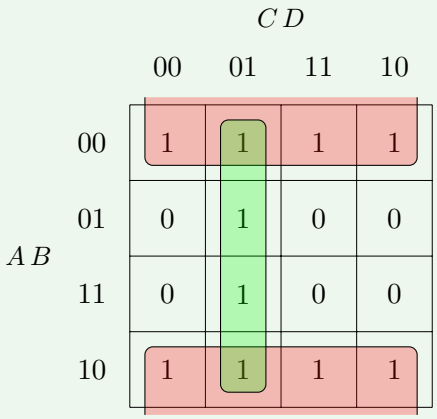
答案：$A \vee B$。（直覺檢查：原式涵蓋「A 真」的兩種情形與「B 真而 A 假」——合起來就是 $A \vee B$。）

練習 6：卡諾圖化簡

用 K-map 求下列真值表的最簡式（輸出 1 的輸入組合：$ABCD \in \{0000, 0001, 0011, 0010, 1000, 1001, 1011, 1010, 0101, 1101\}$）。

解答

填圖（列 AB、欄 CD，格雷碼順序）：



- 第一列（$AB = 00$）與最後一列（$AB = 10$）**上下繞回**合成 2×4 大框（8 格）：框內 A、C、D 都有變化，只有 $B = 0$ 不變 $\Rightarrow \neg B$；
- $CD = 01$ **整欄**（4 格，含 0101 與 1101）：框內 A、B 都有變化，$C = 0$、$D = 1$ 不變 $\Rightarrow \neg C \wedge D$。注意這一欄與上面的大框**重疊**了兩格——重疊讓兩個框都能取到最大，正是「方框應盡量大、可重疊」的威力。

最簡式：$\boxed{\neg B \vee (\neg C \wedge D)}$。

練習 7：閘數計算

(a) 直接按照公式 $(\neg A \wedge \neg B) \vee (\neg A \wedge C)$ 畫電路需要幾個閘？(b) 先化簡再畫呢？

解答

(a) 直接實作：$\neg A$ 出現兩次（分支共用算 1 個 NOT）、$\neg B$ 1 個 NOT、2 個 AND、1 個 OR，共 **5 個閘**。

(b) 用分配律提出 $\neg A$：$\neg A \wedge (\neg B \vee C)$ ——1 個 NOT（A）+ 1 個 NOT（B）+ 1 個 OR + 1 個 AND = **4 個閘**。

練習 8：NAND 化

把 $\neg A \wedge B$ 改寫成只用 NAND 的表達式，並數出所需 NAND 數。

解答

$$\begin{aligned} & \neg A \wedge B \\ \equiv\ & \neg\neg(\neg A \wedge B) \quad \text{【雙重否定】} \\ \equiv\ & \neg\underbrace{\big(\neg(\neg A \wedge B)\big)}_{\text{NAND}} \end{aligned}$$

逐層拆解：$\neg A$ 是 1 個 NAND（$\neg(A \wedge A)$）；$\neg(\neg A \wedge B)$ 是第 2 個 NAND；最外層的 $\neg X$（X 為前式）是第 3 個 NAND（$\neg(X \wedge X)$）。完整寫法：

$$\neg\Big(\neg\big(\neg(A \wedge A) \wedge B\big) \wedge \neg\big(\neg(A \wedge A) \wedge B\big)\Big)$$

共 **3 個 NAND**——恰等於整理後否定符號的個數。✓

練習 9：NOR 的功能完備性

仿照 NAND 的做法，證明 NOR（$\neg(A \vee B)$）也是功能完備的。

解答

只需用 NOR 做出 NOT、OR、AND 三個基本閘：

NOT：	$\neg A \equiv \neg(A \vee A)$	（1 個 NOR，冪等律）
OR：	$A \vee B \equiv \neg\neg(A \vee B) \equiv \neg\big(\neg(A \vee B) \vee \neg(A \vee B)\big)$	（2 個 NOR：NOR + NOT）
AND：	$A \wedge B \equiv \neg\neg(A \wedge B) \equiv \neg(\neg A \vee \neg B)$	（3 個 NOR：兩個 NOT + NOR）

由於 $\{\neg, \wedge, \vee\}$ 功能完備（DNF/CNF 保證任何真值表都可表達），而三者都能用 NOR 構成，故 NOR 單獨功能完備。■（與 NAND 恰成對偶：NAND 做 AND 要 2 個、做 OR 要 3 個；NOR 相反。）

練習 10：用 NAND 實作 XOR

求 $A \oplus B$ 的純 NAND 實作，目標 4 個 NAND。

解答

從等價式出發（令 $N = \neg(A \wedge B)$，即第 1 個 NAND 的輸出）：

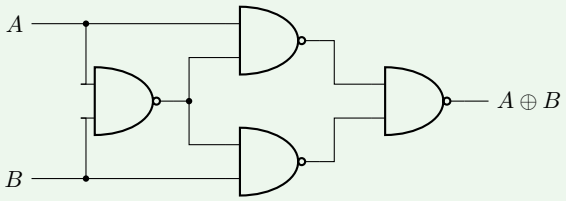
$$A \oplus B \equiv (A \vee B) \wedge \neg(A \wedge B) \equiv (A \wedge \neg(A \wedge B)) \vee (B \wedge \neg(A \wedge B))$$

（第二步：把 $\neg(A \wedge B)$ 分配進 $(A \vee B)$；$A \wedge \neg(A \wedge B) \equiv A \wedge \neg B$、$B \wedge \neg(A \wedge B) \equiv B \wedge \neg A$，正是 XOR 的 DNF。）
再套雙重否定與德摩根：

$$\equiv \neg\big(\neg(A \wedge N) \wedge \neg(B \wedge N)\big)$$

四個 NAND 分別是：

1. $N = \neg(A \wedge B)$
2. $P = \neg(A \wedge N)$
3. $Q = \neg(B \wedge N)$
4. 輸出 $= \neg(P \wedge Q)$



驗證（真值表）：$A{=}B$ 時輸出 0；$A{\neq}B$ 時輸出 1。✓

A 附錄：速查表

A.1 名詞中英對照

英文	中文	英文	中文
proposition	命題	disjunctive normal form (DNF)	析取範式
propositional variable	命題變數	conjunctive normal form (CNF)	合取範式
negation / complement	否定／補	distributivity	分配律
conjunction	合取（AND）	Karnaugh map (K-map)	卡諾圖
disjunction	析取（OR）	Gray code	格雷碼
exclusive or (XOR)	互斥或	minimal form	最簡形式
implication	蘊含	logic gate	邏輯閘
equivalence	等價	functionally complete	功能完備
truth table	真值表	bubble pushing	推泡泡
compound proposition	複合命題	branching (a signal)	（訊號）分支
logically equivalent	邏輯等價	double negation	雙重否定
De Morgan's laws	德摩根定律	propagation	傳遞

A.2 六個運算子速查

A	B	$\neg A$	$A \wedge B$	$A \vee B$	$A \oplus B$	$A \to B$	$A \leftrightarrow B$
0	0	1	0	0	0	1	1
0	1	1	0	1	1	1	0
1	0	0	0	1	1	0	0
1	1	0	1	1	0	1	1

衍生運算子的基本式改寫：$A \oplus B \equiv (\neg A \vee \neg B) \wedge (A \vee B)$；　$A \to B \equiv \neg A \vee B$；　$A \leftrightarrow B \equiv (\neg A \vee B) \wedge (\neg B \vee A)$。

A.3 NAND 實作速查

目標	NAND 寫法	NAND 數
$\neg A$	$\neg(A \wedge A)$	1
$A \wedge B$	$\neg(\neg(A \wedge B) \wedge \neg(A \wedge B))$	2
$A \vee B$	$\neg(\neg(A \wedge A) \wedge \neg(B \wedge B))$	3
$A \oplus B$	見練習 10	4

A.4 參考資料

- Kira Clements, *COMSM1302 Week 1 lecture slides*（1.1–1.4），University of Bristol.

- Nisan & Schocken, *The Elements of Computing Systems*（nand2tetris），Ch. 1 Boolean Logic.
- Wikipedia：*NAND logic*、*Karnaugh map*、*Functional completeness*.
- All About Circuits，*Logic Simplification With Karnaugh Maps*.
- Logisim 使用手冊：`http://www.cburch.com/logisim/docs.html`
- Sheffer, H. M.（1913），NAND 功能完備性的原始證明.