COMSM1302 電腦架構概論

Lab Sheet 2 完整詳解

半加器 · 全加器 · 加減法器 · NAND 加法器 · 多工器 · Hack ALU · 進位制轉換

逐題解析（含全部挑戰題）

2026 年 7 月

本實驗的學習目標：完成本 Lab 後，你應該能夠——

1. 在 Logisim 中建立**子電路（subcircuit）**並在其他設計中當作元件使用；
2. 使用**分線器（splitter）**組裝處理多位元輸入輸出的電路；
3. 設計能執行算術運算（特別是加法與減法）的電路；
4. 在電路設計中使用**多工器**與**解多工器**，依輸入做出計算決策。

所需工具：Logisim（電路模擬；需要 Java）與 NAND 板（實體驗證）。子電路用法：在電路名稱上單擊、再點擊要放置的位置。分線器與常數腳位都在 wiring 函式庫中。

目錄

1　半加器（Half Adder）與 4 位元增量器

題目

在 Logisim 上組裝 1 位元**半加器**，只能用兩個以下輸入的閘（NOT 閘只有一個輸入，其他閘一律用雙輸入版本）。行為由右側真值表定義。接著用半加器作為子電路，建立 **4 位元增量器**：從 4 位元輸入腳位取值、加 1（來自常數 1 位元腳位）、經 4 位元輸出腳位輸出。

A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

解答

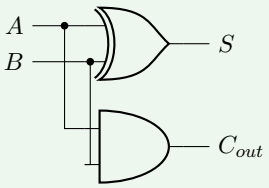
第一步：從真值表讀出兩個輸出的布林式。

逐欄觀察：

- S 在「恰好一個輸入為 1」時為 1——這正是 **XOR**：$S = A \oplus B$；
- C_{out} 只在兩個輸入都為 1 時為 1——這是 **AND**：$C_{out} = A \wedge B$。

直覺：一位元加法 $A + B$ 的結果最大是 $1 + 1 = 2 = 10_2$——S 是結果的個位、C_{out} 是進位。

半加器電路（2 個閘）：

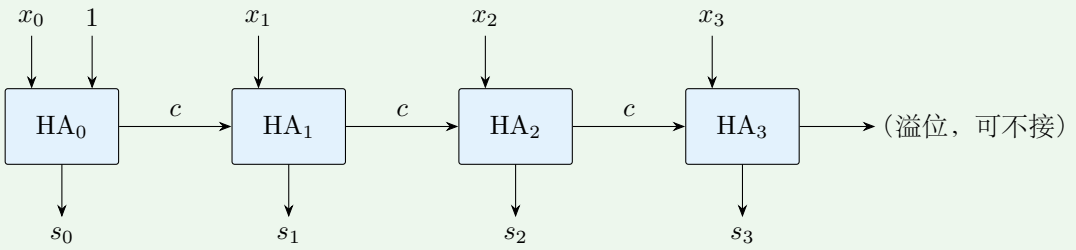


Logisim 操作提示：若閘的清單沒有 XOR，可用 $S = (A \vee B) \wedge \neg(A \wedge B)$ 展開（OR、AND、NOT 各一，加上共用的 AND 共 4 個閘），但 Logisim 的 Gates 函式庫本來就有 XOR，直接用即可（記得屬性面板把輸入數設成 2）。

第二步：4 位元增量器。

「加 1」= 把 4 位元輸入 $x_3x_2x_1x_0$ 與常數 0001 相加。因為第二個運算元只有最低位是 1，**不需要**完整加法器——半加器鏈就夠：

- 第 0 位：$x_0 + 1 \to$ 半加器 HA_0（輸入 x_0 與常數 1）；
- 第 i 位（$i \geq 1$）：x_i+ 前一級的進位 $\to \text{HA}_i$；
- 每級的 S 就是輸出位 s_i，C_{out} 餵給下一級。



Logisim 組裝要點：

1. 輸入腳位屬性 Data Bits = 4；接一個 **splitter**（Fan Out = 4，Bit Width In = 4）拆成 4 條 1 位元線；
2. 放 4 份半加器子電路；常數腳位（Constant，值 1）接 HA_0 的第二輸入；
3. 另一個 splitter 反向使用（4 條 1 位元合成 4 位元）接到 4 位元輸出腳位；
4. 最高位的 C_{out} 懸空即可（$1111+1$ 會繞回 0000 ——這正是模 2^4 算術，也是 2 補數運作的基礎）。

驗證：輸入 0111（7）應輸出 1000（8）；輸入 1111（15）應輸出 0000（繞回）。

2　全加器（Full Adder）與 4 位元加法器

題目

組裝 1 位元**全加器**（雙輸入以下的閘），行為由右表定義。接著用全加器作子電路建立 **4 位元加法器**：兩個 4 位元輸入腳位相加、4 位元輸出，另含 1 位元進位輸入與 1 位元進位輸出腳位。

C_{in}	A	B	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

解答

第一步：讀出布林式。

S 在「1 的個數為奇數」時為 1（第 2、3、5、8 列）：

$$S = A \oplus B \oplus C_{in}.$$

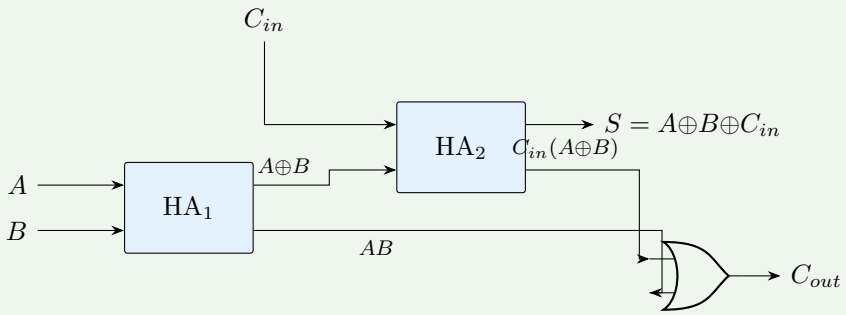
C_{out} 在「至少兩個輸入為 1」時為 1（第 4、6、7、8 列）。標準化簡：

$$C_{out} = (A \wedge B) \vee \big(C_{in} \wedge (A \oplus B)\big).$$

（也可寫成 $AB + AC_{in} + BC_{in}$；上式的好處是能**重用**半加器結構。）

第二步：用兩個半加器＋一個 OR 組裝（最經濟的做法）。

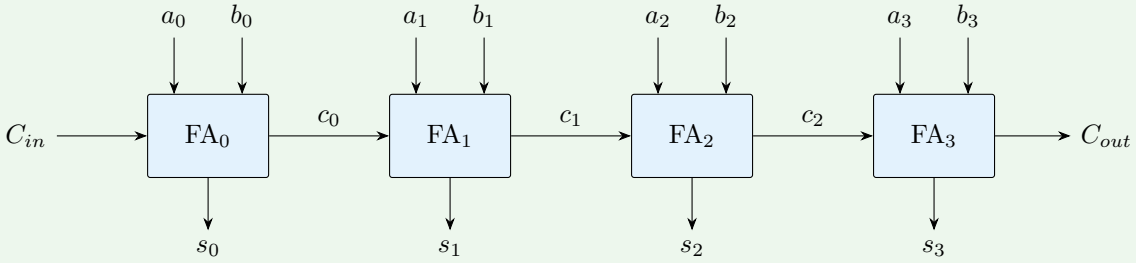
把加法拆成兩步：先算 $A + B$（半加器 1），再把其和與 C_{in} 相加（半加器 2）；兩步只要**任一步**產生進位，C_{out} 就是 1（兩步不可能同時進位，OR 即可）：



閘數統計：每個半加器 2 個閘（XOR、AND）＋ 1 個 OR $= 2 \times 2 + 1 =$ **5 個閘**。在 Logisim 中：放兩份半加器子電路＋一個 OR 閘即可——這正是「子電路」威力的示範。

第三步：4 位元漣波進位加法器（ripple-carry adder）。

四份全加器子電路串成一列，進位像漣波一樣從低位傳向高位：



組裝要點與增量器相同：兩個 4 位元輸入各接一個 splitter、輸出經 splitter 合併，另加 1 位元 C_{in} 輸入腳位與 C_{out} 輸出腳位。驗證建議：$3+5 = 8$（$0011+0101 = 1000$）、$9+9 = 18 = 10010_2$（輸出 0010、$C_{out} = 1$——溢位可見）、$C_{in} = 1$ 時 $0 + 0$ 應輸出 0001。

挑戰題

挑戰：能否改造 4 位元加法器，讓它既能加也能減？（提示：需要一個多工器，以及對 2 補數的理解！）

核心觀念： 2 補數下

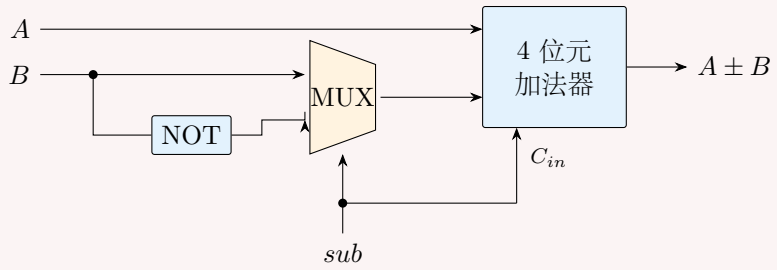
$$A - B \;=\; A + (-B) \;=\; A + \underbrace{\neg B + 1}_{\text{2 補數取負}}.$$

所以減法 ＝「把 B 逐位取反」＋「把 C_{in} 設為 1」。新增一條控制線 sub：

- $sub = 0$：加法——B 原樣通過、$C_{in} = 0$；

- $sub = 1$：減法——B 取反、$C_{in} = 1$（正好補上 +1，一石二鳥）。

做法一（題目提示的做法）：放一個 NOT 閘（4 位元）算出 $\neg B$，再用一顆 **4 位元 2-to-1 多工器**（Plexers 函式庫，Select 接 sub）在 B 與 $\neg B$ 之間選擇；sub 同時接到加法器的 C_{in}：



做法二（更省元件）：用 4 個 **XOR** 閘取代 NOT + MUX——$B_i \oplus sub$ 在 $sub = 0$ 時等於 B_i、$sub = 1$ 時等於 $\neg B_i$，即「可控反相器」。兩種做法邏輯完全等價；本題指定用多工器，但值得知道實際硬體幾乎都用 XOR 版。

驗證：$sub = 1$ 時測 $5 - 3 = 2$、$3 - 5 = -2$（輸出 1110，2 補數的 -2 ✓）、$7 - 7 = 0$。注意 $3 - 5$ 時 $C_{out} = 0$、$5 - 3$ 時 $C_{out} = 1$：減法時進位輸出的意義是「**無借位**」。

3　NAND 加法器

題目

只用**雙輸入 NAND 閘**在 Logisim 上組裝：(1) 1 位元半加器；(2) 1 位元全加器。並在 NAND 板上實體驗證（先想好接線布局）。選做：與另一位同學的板子串接全加器（兩板需用 USB 分接器共用同一電源）。

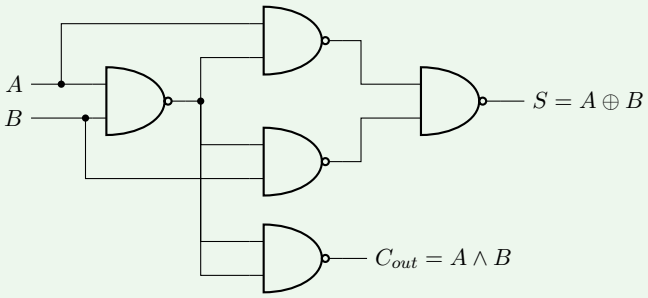
解答

回憶第一週的 NAND 建構塊：

$$\neg A = A \barwedge A, \qquad A \wedge B = (A \barwedge B) \barwedge (A \barwedge B),$$
$$A \vee B = (A \barwedge A) \barwedge (B \barwedge B), \qquad A \oplus B = 4 \text{ 個 NAND（見下）}.$$

半加器：5 個 NAND。XOR 的 4-NAND 經典結構，關鍵是第一個 NAND 的輸出 $N_1 = A \barwedge B$ 可**同時**用來造 C_{out}：

$$N_1 = A \barwedge B,$$
$$N_2 = A \barwedge N_1, \quad N_3 = B \barwedge N_1,$$
$$S = N_2 \barwedge N_3 = A \oplus B,$$
$$C_{out} = N_1 \barwedge N_1 = \neg N_1 = A \wedge B.$$

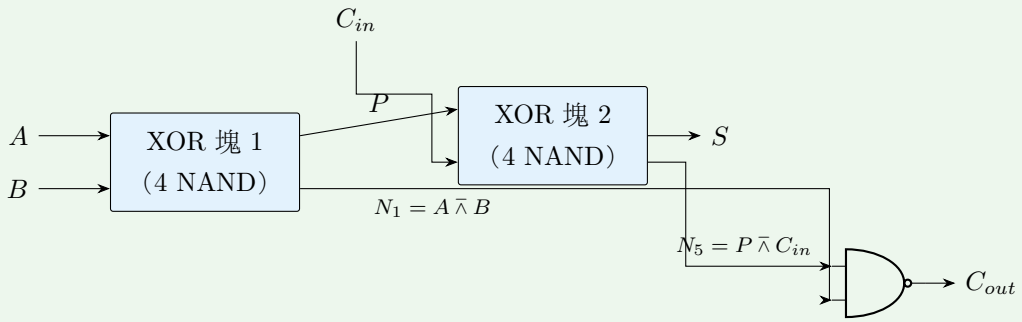


為什麼 4 個 NAND 就是 XOR? 展開驗證：$N_2 = \neg(A \wedge \neg(AB)) = \neg A \vee AB$、$N_3 = \neg B \vee AB$，$S = \neg(N_2 \wedge N_3)$；代入化簡得 $S = A\bar{B} + \bar{A}B$ ✓（或直接列真值表驗證四列）。

全加器：9 個 NAND。沿用第 2 節「兩個半加器 + OR」的結構，但把 OR 也化成 NAND。妙處在於：半加器內部 $C_{out} = \neg N_1$，而 $X \vee Y = \neg X \barwedge \neg Y$——所以**直接拿兩個半加器的 N_1（尚未反相的進位）NAND 起來**就是 C_{out}，連反相器都省了：

$$\begin{aligned} &\text{第一級（4 個）：} P = A \oplus B, \quad N_1 = A \barwedge B \\ &\text{第二級（4 個）：} S = P \oplus C_{in}, \quad N_5 = P \barwedge C_{in} \\ &\text{進位（1 個）：} C_{out} = N_1 \barwedge N_5 = (A \wedge B) \vee (P \wedge C_{in}) \ \checkmark \end{aligned}$$

合計 $4 + 4 + 1 = \mathbf{9}$ **個 NAND**。



NAND 板實作要點：

- 板上共 16 個 NAND（4 顆晶片 × 4 閘）——半加器用 5 個、全加器用 9 個，都放得下；
- **先規劃再接線**：把 XOR 塊 1 配置在晶片 1、XOR 塊 2 在晶片 2、進位 NAND 用晶片 3，訊號從左往右流，可大幅減少跨板飛線；
- 輸入用按鈕（按下 = 1）：半加器用 2 顆、全加器用 3 顆；輸出直接讀對應 NAND 的 LED；
- 未接線的輸入因下拉電阻預設為 0——測試時要留意；
- **串接兩塊板**（選做）：把本板全加器的 C_{out} LED 引腳接到對方板全加器的 C_{in} 輸入，即得 2 位元漣波加法器。兩板必須**共用同一 USB 電源**（用 USB 分接器）——否則兩板的邏輯高低電位參考點（接地）不同，訊號無法正確判讀。

4　多工器與解多工器（Plexers）

題目

為下列定義各找出等價的布林表達式（用 $\wedge$、$\vee$、$\neg$；所有訊號皆為 1 位元），並在 Logisim 上用雙輸入以下的閘組裝驗證：

- **2-to-1 多工器**：讓兩個輸入訊號之一通過到單一輸出，由一個選擇輸入控制；
- **1-to-2 解多工器**：把單一輸入訊號送到兩個輸出之一，由一個選擇輸入控制。

解答

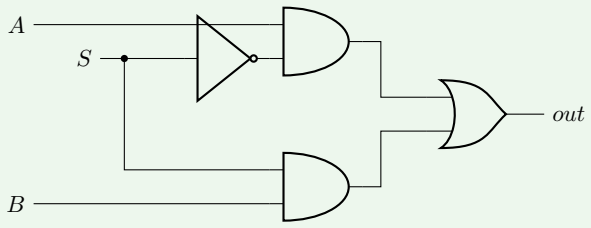
2-to-1 多工器。輸入 A、B 與選擇線 S（$S=0$ 選 A、$S=1$ 選 B）。真值表（8 列）：

S	A	B	out	
0	0	0	0	$S=0$：out 跟著 A
0	0	1	0	
0	1	0	1	
0	1	1	1	
1	0	0	0	$S=1$：out 跟著 B
1	0	1	1	
1	1	0	0	
1	1	1	1	

從 DNF 取 1 的列化簡（或直接由「$S=0$ 取 A、$S=1$ 取 B」的語意寫出）：

$$\boxed{out = (\neg S \wedge A) \vee (S \wedge B)}$$

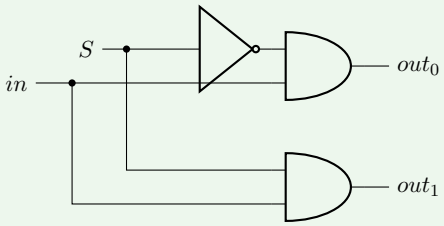
3 種閘、共 4 個（NOT、AND×2、OR）：



1-to-2 解多工器。輸入 in 與選擇線 S（$S=0$ 送到 out_0、$S=1$ 送到 out_1；沒被選中的輸出為 0）：

S	in	out_0	out_1
0	0	0	0
0	1	1	0
1	0	0	0
1	1	0	1

$$\boxed{out_0 = \neg S \wedge in, \qquad out_1 = S \wedge in}$$

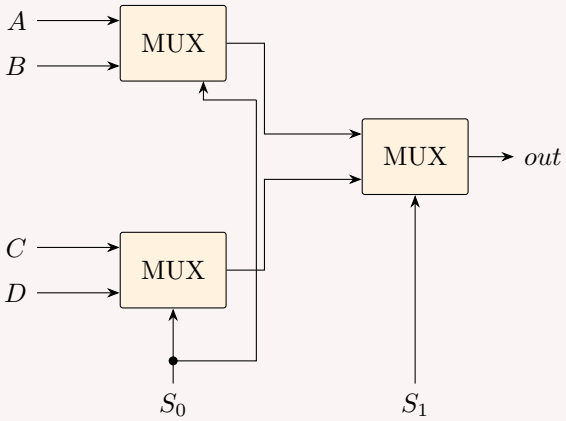


觀察：MUX 與 DEMUX 是「**鏡像**」——MUX 用 AND 閘**擋住**沒被選中的輸入（強制為 0）再 OR 合併；DEMUX 用 AND 閘擋住沒被選中的**輸出**。Logisim 驗證：窮舉所有輸入組合（MUX 8 組、DEMUX 4 組），對照真值表。

挑戰題

挑戰：組裝 4-to-1 多工器與 1-to-4 解多工器。

方法一：樹狀組合（最優雅，重用子電路）。4 選 1 需要 2 條選擇線 S_1S_0。用 **3 個 2-to-1 MUX** 排成兩層：第一層用 S_0 從 (A, B) 與 (C, D) 各選一個，第二層用 S_1 在兩個中選者之間做最終決定：



對應布林式（把樹展開）：

$$out = \big(\neg S_1 \wedge (\neg S_0 A \vee S_0 B)\big) \vee \big(S_1 \wedge (\neg S_0 C \vee S_0 D)\big).$$

方法二：平坦式（一層解碼）。每個輸入配一個「三輸入 AND」（用兩個雙輸入 AND 串聯），選中條件是 S_1S_0 的對應組合，再全部 OR 起來：

$$out = \bar{S}_1\bar{S}_0 A \;\vee\; \bar{S}_1 S_0 B \;\vee\; S_1\bar{S}_0 C \;\vee\; S_1 S_0 D.$$

1-to-4 解多工器完全對偶：3 個 1-to-2 DEMUX 的樹（第一層用 S_1 分上下、第二層用 S_0 分左右），或平坦式

$$out_0 = \bar{S}_1\bar{S}_0\, in, \quad out_1 = \bar{S}_1 S_0\, in, \quad out_2 = S_1\bar{S}_0\, in, \quad out_3 = S_1 S_0\, in.$$

一般化：2^n-to-1 MUX 需要 $2^n - 1$ 個 2-to-1 MUX、n 條選擇線——這正是 Hack ALU 與記憶體定址電路的積木。

5　算術邏輯單元（Hack ALU）

題目

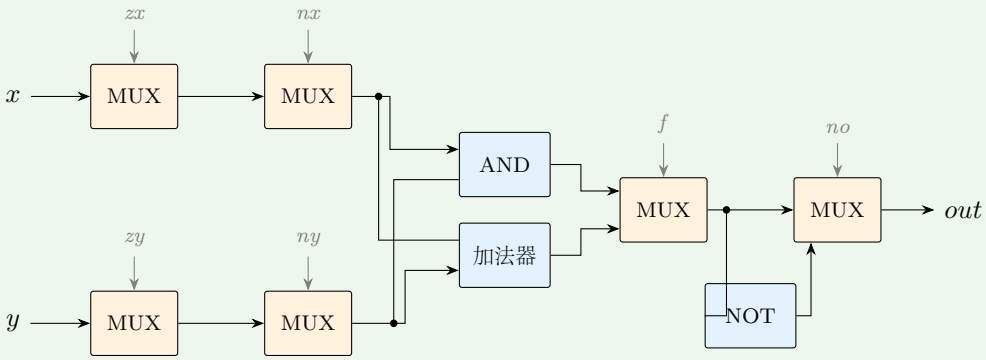
在 Logisim 上建造 Hack ALU：包含 6 顆預建多工器，控制位如下（x、y、out 皆為 4 位元，用你的 4 位元加法器作子電路）：

1. **zx**：$zx = 0$ 則 $x = x$，否則 $x = 0$
2. **nx**：$nx = 0$ 則 $x = x$，否則 $x = \neg x$
3. **zy**：$zy = 0$ 則 $y = y$，否則 $y = 0$
4. **ny**：$ny = 0$ 則 $y = y$，否則 $y = \neg y$
5. **f**：$f = 0$ 則 $out = x \,\&\, y$，否則 $out = x + y$
6. **no**：$no = 0$ 則 $out = out$，否則 $out = \neg out$

再接上兩個 1 位元比較輸出：**zr**（$out = 0$ 時為 1）與 **ng**（$out < 0$ 時為 1）。用題目給的表（18 列控制位組合）測試設計。

解答

整體結構：兩條前處理管線＋運算選擇＋後處理。



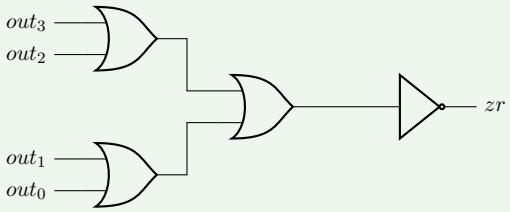
每一級的細節（全部用「多工器做決策」的同一招）：

1. **zx 級**：4 位元 MUX，輸入 0 接 x、輸入 1 接常數 0000，Select 接 zx；
2. **nx 級**：4 位元 MUX，輸入 0 接上一級輸出、輸入 1 接其 NOT（4 位元 NOT 閘），Select 接 nx；
3. **zy ／ ny 級**：與 1、2 完全相同，作用在 y；
4. **f 級**：AND（4 位元）與你的 4 位元加法器（$C_{in} = 0$，C_{out} 懸空）**同時**計算，MUX 用 f 選擇哪個結果通過——輸入 0 接 AND、輸入 1 接加法器；
5. **no 級**：MUX 在「原樣」與「NOT」之間選擇，Select 接 no。

共 6 顆 MUX ✓。注意順序：**先歸零、再取反**（所以 $zx = nx = 1$ 得到 $\neg 0 = 1111$，不是 0）；輸出端**先運算、再取反**。

zr 與 ng：

- ng：2 補數的負數 = MSB 為 1，所以 $ng = out_3$——用 splitter 拆出最高位直接接出，零個閘；
- zr：$out = 0000$ 表示「所有位都是 0」，即 $zr = \neg(out_3 \vee out_2 \vee out_1 \vee out_0)$ ——三個雙輸入 OR 串成樹再接一個 NOT（4 個閘）。



測試：把 18 列控制位逐一撥入，用幾組 x、y 值（如 $x = 0011 = 3$、$y = 0101 = 5$）對照表中預期輸出。例如 $x{+}y$ 列應得 $1000 = 8$、$x{-}y$ 列應得 $1110 = -2$、$x\&y$ 列應得 0001。

挑戰題

挑戰：解釋為什麼每一列的計算會產生表中定義的輸出。

全部推導只需要一條 2 補數恆等式：

$$\boxed{\neg a = -a - 1 \quad \Longleftrightarrow \quad -a = \neg a + 1}$$

（逐位取反 = 取負再減一。）以下把 18 列分成六組，x、y 表示前處理**後**進入 f 級的值。

組 1：常數（不看輸入）

輸出	控制位效果	推導
0	$x{\to}0$，$y{\to}0$，加法	$0 + 0 = 0$
1	$x{\to}\neg 0{=}{-1}$，$y{\to}{-1}$，加法，取反	$\neg(-1 + -1) = \neg(-2) = 2 - 1 = 1$
-1	$x{\to}{-1}$，$y{\to}0$，加法	$-1 + 0 = -1$

關鍵：歸零**在前**、取反**在後**，所以 $zx = 1, nx = 1$ 製造出 $\neg 0 = 1111 = -1$——ALU 憑空生出常數 -1 的手法。

組 2：直通與位元取反

輸出	控制位效果	推導
x	$y{\to}{-1}{=}1111$，AND	$x \,\&\, 1111 = x$（AND 的單位元素）
y	$x{\to}1111$，AND	$1111 \,\&\, y = y$
$!x$	$y{\to}1111$，AND，取反	$\neg(x\&1111) = \neg x$
$!y$	$x{\to}1111$，AND，取反	$\neg y$

關鍵：1111 是 AND 的**單位元素**（相當於乘以 1），用它「墊住」另一邊，就能讓單一運算元穿過雙運算元的 ALU。

組 3：算術取負

輸出	控制位效果	推導
$-x$	$y{\to}{-1}$，加法，取反	$\neg(x + (-1)) = \neg(x - 1) = -(x - 1) - 1 = -x$
$-y$	$x{\to}{-1}$，加法，取反	同理 $= -y$

組 4：加一與減一

輸出	控制位效果	推導
$x{+}1$	$x{\to}\neg x$，$y{\to}{-1}$，加法，取反	$\neg(\neg x-1)=-(\neg x-1)-1=-\neg x=x+1$
$y{+}1$	$x{\to}{-1}$，$y{\to}\neg y$，加法，取反	同理 $=y+1$
$x{-}1$	$y{\to}{-1}$，加法	$x+(-1)=x-1$
$y{-}1$	$x{\to}{-1}$，加法	$y-1$

（$-\neg x=-(-x-1)=x+1$——同一條恆等式再用一次。）

組 5：加法與減法

輸出	控制位效果	推導
$x{+}y$	全 0，加法	直接相加
$x{-}y$	$x{\to}\neg x$，加法，取反	$\neg(\neg x+y)=-(-x-1+y)-1=x-y$
$y{-}x$	$y{\to}\neg y$，加法，取反	對稱 $=y-x$

組 6：位元運算

輸出	控制位效果	推導
$x\&y$	全 0，AND	直接 AND
$x\|y$	$x{\to}\neg x$，$y{\to}\neg y$，AND，取反	$\neg(\neg x\,\&\,\neg y)=x\vee y$（**De Morgan**）

總結：六個看似簡陋的控制位能產生 18 種有用運算，靠的是三個代數事實——(1) $\neg a=-a-1$（2 補數）；(2) 1111 是 AND 的單位元素、0000 是加法的單位元素；(3) De Morgan 定律。這正是 Hack ALU 設計的精髓：**硬體極簡，把複雜度外移到控制位的組合上。**

6　進位制轉換（Number Representations）

題目

把下列每個數轉換成二進位、八進位、十進位與十六進位（原本的進位制除外）。適用時可假設使用 8 位元與 2 補數表示法。

10101101_2	205_8	123_{10}	92_{16}
01110101_2	312_8	-3_{10}	$7F_{16}$
11011100_2	051_8	-84_{10}	DA_{16}

解答

方法總覽（8 位元一律先化成位元組再轉換）：

- **二 → 十六**：從右往左**每 4 位**一組查表；**二 → 八**：每 **3 位**一組（不足左補 0）；
- **八／十六 → 二**：每個數字獨立展開成 3 ／ 4 個位元；
- **二 → 十**：MSB 為 0 直接按權重加總；MSB 為 1 且採 2 補數時，值 = 無號值 -256（或：取反加一得絕對值）；

- **負十進位 → 二**：先寫出絕對值、再取反加一。

完整答案表（粗體為題目給定；十進位欄以 8 位元 2 補數解讀，括號內為無號值）：

#	二進位	八進位	十進位	十六進位
1	**10101101**	255	−83（173）	AD
2	**01110101**	165	117	75
3	**11011100**	334	−36（220）	DC
4	10000101	**205**	−123（133）	85
5	11001010	**312**	−54（202）	CA
6	00101001	**051**	41	29
7	01111011	173	**123**	7B
8	11111101	375	−3	FD
9	10101100	254	−84	AC
10	10010010	222	−110（146）	**92**
11	01111111	177	127	**7F**
12	11011010	332	−38（218）	**DA**

代表性的完整推導（每類示範一題，其餘同法）：

第 1 題 10101101_2：

- 十六進位：$\underbrace{1010}_{A}\underbrace{1101}_{D} \Rightarrow AD$；
- 八進位：左補 0 成 9 位 $\underbrace{010}_{2}\underbrace{101}_{5}\underbrace{101}_{5} \Rightarrow 255_8$；
- 十進位：MSB $= 1$，是負數。無號值 $= 128+32+8+4+1 = 173$，2 補數值 $= 173-256 = -\mathbf{83}$。驗算：取反 $01010010 = 82$、加一 $= 83$，故 -83 ✓。

第 4 題 205_8：每個八進位數字展開 3 位：2→010、0→000、5→101，拼起來 010000101，取低 8 位 $= 10000101_2$。十六進位：$1000\,0101 \Rightarrow 85$。十進位：無號 $128+4+1 = 133$；以 2 補數解讀 $133-256 = -\mathbf{123}$。

第 8 題 -3_{10}：$3 = 00000011$；取反 11111100；加一 **11111101**。十六進位 $1111\,1101 = FD$；八進位 $011\,111\,101 = 375_8$。（快速驗算：$-1 = 11111111$、$-2 = 11111110$、$-3 = 11111101$——從全 1 倒著數。）

第 9 題 -84_{10}：$84 = 64+16+4 = 01010100$；取反 10101011；加一 **10101100**。十六 $= AC$、八 $= 254_8$。

第 10 題 92_{16}：9→1001、2→0010 $\Rightarrow 10010010_2$。八進位 $010\,010\,010 = 222_8$。十進位：無號 $128+16+2 = 146$；2 補數 $146-256 = -\mathbf{110}$。

十進位欄的兩種讀法：題目說「適用時假設 8 位元 2 補數」。MSB 為 0 的數（第 2、6、7、11 題）兩種讀法相同；MSB 為 1 的數（第 1、3、4、5、8、9、10、12 題）若當**無號數**讀就是括號內的值，當 **2 補數**讀就是負值——兩者恆差 256。八進位與十六進位欄位單純轉錄**位元組樣式**，不帶正負號。

挑戰題

挑戰：把每個二進位數改用符號大小（signed magnitude）與 1 補數（1's complement）解讀，求其十進位值。

規則：

- **符號大小**：MSB 是正負號（1 = 負），其餘 7 位是**絕對值**；
- **1 補數**：MSB 為 1 時，值 = −（逐位取反後的無號值）；等價地：1 補數值 = 2 補數值 +1（限負數）。

位元組	2 補數（對照）	符號大小	1 補數
10101101	-83	$-(0101101_2) = -45$	$-(01010010_2) = -82$
01110101	117	117	117
11011100	-36	$-(1011100_2) = -92$	$-(00100011_2) = -35$
10000101	-123	-5	-122
11001010	-54	-74	-53
00101001	41	41	41
01111011	123	123	123
11111101	-3	-125	-2
10101100	-84	-44	-83
10010010	-110	-18	-109
01111111	127	127	127
11011010	-38	-90	-37

示範推導 (10101101): 符號大小——符號位 1（負），絕對值 $0101101_2 = 32+8+4+1 = 45 \Rightarrow -45$。1 補數——MSB 為 1，取反得 $01010010 = 82 \Rightarrow -82$。

觀察三種表示法的規律：**正數（MSB=0）三種讀法完全相同**；負數時三者各不相同，且恆有「1 補數值 = 2 補數值 +1」。這也解釋了為什麼 2 補數勝出：符號大小與 1 補數都有 +0 ／ −0 兩個零（00000000 與 10000000 ／ 11111111），硬體要對「兩個零」做特判；2 補數只有一個零，加法器不分正負一體適用（第二週講義的核心論點）。

挑戰題

額外挑戰：把每個位元組解讀為浮點數。格式：1 個符號位、2 個指數位、5 個尾數位；指數採**偏移**儲存（2 位指數的偏移量為 1）、尾數**正規化**（隱含前導 1）。

解讀公式： 位元組 $s\,e_1e_0\,m_4m_3m_2m_1m_0$ 的值為

$$(-1)^s \times 1.m_4m_3m_2m_1m_{0(2)} \times 2^{E-1}, \qquad E = e_1e_{0(2)} \in \{0,1,2,3\}.$$

尾數的權重：$m_4 = \frac{1}{2}$、$m_3 = \frac{1}{4}$、$m_2 = \frac{1}{8}$、$m_1 = \frac{1}{16}$、$m_0 = \frac{1}{32}$。

位元組	s	E（2^{E-1}）	尾數 $1.m$	計算	值
1 01 01101	1	1（×1）	1.40625	-1.40625×1	-1.40625
0 11 10101	0	3（×4）	1.65625	1.65625×4	6.625
1 10 11100	1	2（×2）	1.875	-1.875×2	-3.75
1 00 00101	1	0（$\times\frac{1}{2}$）	1.15625	$-1.15625/2$	-0.578125
1 10 01010	1	2（×2）	1.3125	-1.3125×2	-2.625
0 01 01001	0	1（×1）	1.28125	1.28125×1	1.28125
0 11 11011	0	3（×4）	1.84375	1.84375×4	7.375
1 11 11101	1	3（×4）	1.90625	-1.90625×4	-7.625
1 01 01100	1	1（×1）	1.375	-1.375×1	-1.375
1 00 10010	1	0（$\times\frac{1}{2}$）	1.5625	$-1.5625/2$	-0.78125
0 11 11111	0	3（×4）	1.96875	1.96875×4	7.875
1 10 11010	1	2（×2）	1.8125	-1.8125×2	-3.625

示範推導（10101101）：切成 $s=1$、$e=01$、$m=01101$。指數 $E=01_2=1$，實際冪次 $=E-1=0$，比例 $2^0=1$。尾數 $1.01101_2=1+\frac{1}{4}+\frac{1}{8}+\frac{1}{32}=1.40625$。值 $=-1.40625\times1=\mathbf{-1.40625}$。

示範推導（10010010）：$s=1$、$e=00$、$m=10010$。$E=0$，冪次 $=-1$，比例 $\frac{1}{2}$。尾數 $1.10010_2=1+\frac{1}{2}+\frac{1}{16}=1.5625$。值 $=-1.5625\times\frac{1}{2}=\mathbf{-0.78125}$。

與真正 IEEE 754 的差異：本題的簡化格式把**所有**指數樣式都當正規數處理。真正的 IEEE 754 會保留兩端：$e=00$ 作**次正規數**（隱含前導 0、固定冪次），$e=11$ 作**無窮大／ NaN**（尾數全 0 為 $\pm\infty$，否則 NaN）。若按 IEEE 規則，上表中 $e=00$ 的兩列（-0.578125、-0.78125）與 $e=11$ 的四列（6.625、7.375、-7.625、7.875）都會另有解讀——題目的提示明說「正規化尾數 + 偏移 1」，所以此處採用簡化解讀。

觀察：同一個位元組 10101101，四種解讀得到四個不同的值——無號 173、2 補數 -83、符號大小 -45、1 補數 -82、浮點 -1.40625。**位元本身沒有意義，意義來自約定的解讀方式**——這是本週（也是整門課）最重要的一課。

A　附錄：速查表

A.1　本 Lab 的閘數統計

電路	一般閘	只用 NAND
半加器	2（XOR、AND）	5
全加器	5（半加器 ×2 + OR）	9
4 位元增量器	半加器 ×4	NAND ×20
4 位元加法器	全加器 ×4	NAND ×36
2-to-1 MUX	4（NOT、AND×2、OR）	4
1-to-2 DEMUX	3（NOT、AND×2）	—
4-to-1 MUX	2-to-1 MUX ×3	—
zr	4（OR×3、NOT）	—
ng	0（直接取 MSB）	—

A.2　關鍵公式

主題	公式
半加器	$S = A \oplus B$，$C_{out} = A \wedge B$
全加器	$S = A \oplus B \oplus C_{in}$，$C_{out} = AB \vee C_{in}(A \oplus B)$
2 補數取負	$-a = \neg a + 1$；$\neg a = -a - 1$
減法	$A - B = A + \neg B + 1$（XOR 反相 + $C_{in} = 1$）
2-to-1 MUX	$out = \neg S A \vee S B$
1-to-2 DEMUX	$out_0 = \neg S \cdot in$，$out_1 = S \cdot in$
De Morgan	$\neg(\neg x \wedge \neg y) = x \vee y$
浮點解讀	$(-1)^s \times 1.m \times 2^{E-\text{bias}}$，本題 bias $= 1$

A.3　進位制對照（4 位元）

十	二	八	十六	十	二	八	十六
0	0000	0	0	8	1000	10	8
1	0001	1	1	9	1001	11	9
2	0010	2	2	10	1010	12	A
3	0011	3	3	11	1011	13	B
4	0100	4	4	12	1100	14	C
5	0101	5	5	13	1101	15	D
6	0110	6	6	14	1110	16	E
7	0111	7	7	15	1111	17	F

A.4　參考資料

- COMSM1302 第二週講義（2.1 數的表示法、2.2 二進位加法、2.3 二進位減法、2.4 ALU），University of Bristol.

- Nisan & Schocken, *The Elements of Computing Systems*（nand2tetris），Ch. 2（Boolean Arithmetic）——Hack ALU 的原始出處.
- Logisim User Guide（wiring 函式庫：splitter、constant；Plexers 函式庫：multiplexer、demultiplexer）.