COMSM1302 電腦架構概論

Lab Sheet 3 完整詳解

R-S 閂鎖 · D 閂鎖 · D 正反器 · 傳播延遲 · 階層式 RAM · T ／ JK 正反器

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 上組裝標準記憶元件（R-S 閂鎖、D 閂鎖、D 正反器、n 字組 RAM）；
2. 改造標準記憶元件，使其為主動低／主動高，或正緣／負緣觸發；
3. 操縱標準記憶元件的輸入，讓它儲存想要的值。

所需工具：Logisim 與 NAND 板。本週是課程的分水嶺：電路第一次有了**記憶**——輸出不再只由當前輸入決定，還取決於**過去**。一切的根源是一個危險又美妙的動作：**把輸出接回輸入**。

目錄

1　閂鎖到正反器（Latches to Flip-Flops）

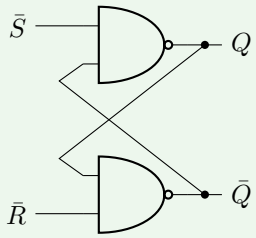
題目

用**雙輸入 NAND 閘**在 Logisim 上建出以下六個電路，並測試你知道如何把儲存值更新為 0 與 1：主動低 R-S 閂鎖、主動高 R-S 閂鎖、主動高 D 閂鎖、主動低 D 閂鎖、正緣觸發 D 正反器、負緣觸發 D 正反器。再於 NAND 板上實體驗證（先規劃接線）。提示：其中一半的設計在課堂上明確講過！

解答

(1) 主動低 R-S 閂鎖（2 個 NAND）——課堂講過。

兩個 NAND 交叉耦合（各自的輸出接到對方的一個輸入）：



「主動低」= 控制輸入平常放 1、**拉到 0 才動作**：

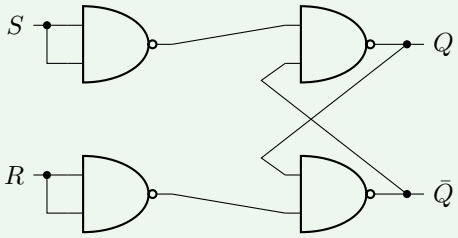
$\bar{S}$	$\bar{R}$	Q	$\bar{Q}$	意義
1	1	保持	保持	**記憶**（維持上一狀態）
0	1	1	0	設定（set）
1	0	0	1	重設（reset）
0	0	1	1	**禁止**（$Q = \bar{Q}$ 矛盾；同時放開會進入不定狀態）

原理：NAND 只要任一輸入為 0 就輸出 1。$\bar{S} = 0$ 強迫 $Q = 1$，經交叉回饋使 $\bar{Q} = 0$，即使 $\bar{S}$ 放回 1，回饋會把狀態**鎖住**。

操作練習（存 1 再存 0）：$\bar{S}$ 短暫拉 0（$Q \to 1$）、放回 1（保持）；$\bar{R}$ 短暫拉 0（$Q \to 0$）、放回 1（保持）。

(2) 主動高 R-S 閂鎖（4 個 NAND）。

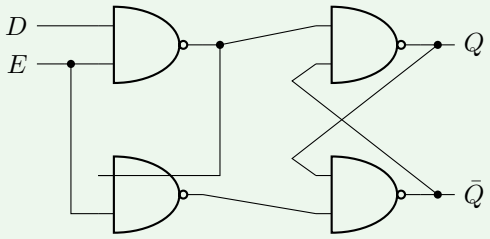
「主動高」= 平常放 0、拉到 1 才動作。在兩個控制輸入前各加一個 NAND 作 NOT 即可（S 反相成 $\bar{S}$、R 反相成 $\bar{R}$）：



行為表隨之反轉：$S = R = 0$ 保持、$S = 1$ 設定、$R = 1$ 重設、$S = R = 1$ 禁止。

(3) 主動高 D 閂鎖（4 個 NAND）——課堂講過。

R-S 閂鎖的禁止狀態很危險。D 閂鎖用**一條資料線** D **+ 一條致能線** E 消除它：$E = 1$ 時**透明**（Q 跟隨 D）、$E = 0$ 時**鎖住**。



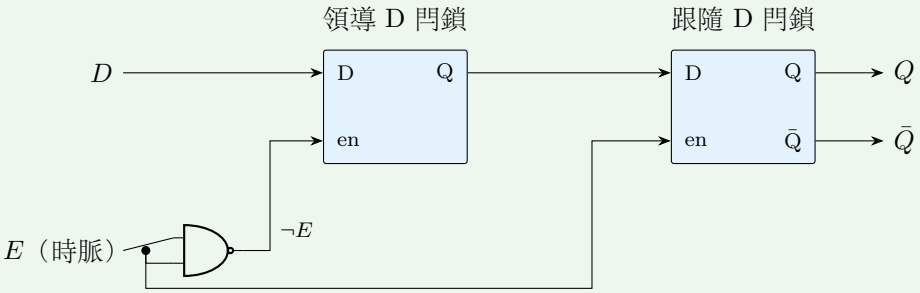
結構解讀：左側兩個 NAND 產生 $\bar{S} = D \barwedge E$、$\bar{R} = \bar{D} \barwedge E$（注意 $\bar{D}$ 直接取自第一個 NAND 的輸出——省一個閘），右側就是 (1) 的主動低 R-S 核心。$E = 1$ 時 $\bar{S}$、$\bar{R}$ 恰好互補（禁止態不可能出現）；$E = 0$ 時兩者皆 1（保持）。

(4) 主動低 D 閂鎖（5 個 NAND）。

要求 $E = 0$ 透明、$E = 1$ 鎖住：在 (3) 的 E 前面加一個 NAND 作 NOT 即可（輸入 E 兩腳短接）。共 $4 + 1 = 5$ 個 NAND。

(5) 正緣觸發 D 正反器（9 個 NAND）——課堂講過。

領導—跟隨（leader–follower）結構：兩個 D 閂鎖串聯、致能互補：



運作原理（**兩扇門永遠不同時開**）：

- $E = 0$：領導透明（追蹤 D）、跟隨鎖住（輸出不變）；
- E 上升瞬間：領導**鎖住**「邊緣前一刻的 D」、跟隨**打開**，把這個值送到 Q；
- $E = 1$ 期間：D 再怎麼變都被領導擋住。

結果：Q **只在** E 的**上升緣**更新——邊緣觸發。NAND 計數：主動低 D 閂鎖需要「$E = 0$ 透明」……更精確地：領導用「E 先反相再進主動高 D 閂鎖」的形式（$4 + 1$），跟隨用主動高 D 閂鎖（4），反相 NAND 已算在領導裡，共 **9** 個。

(6) 負緣觸發 D 正反器（9 個 NAND）。

把反相器**搬到跟隨那邊**（領導 en $= E$、跟隨 en $= \neg E$）：$E = 1$ 時領導追蹤 D，E **下降**瞬間領導鎖住、跟隨打開——Q 只在下降緣更新。

NAND 板驗證要點：

- 板上 16 個 NAND：R-S（2 或 4）、D 閂鎖（4 或 5）都輕鬆放得下；D 正反器要 9 個——超過兩顆晶片，**先在紙上把每個 NAND 對應到晶片腳位再接線**；
- 交叉耦合的回饋線是最容易接錯的地方：建議先接好核心 R-S 對、用按鈕驗證 set/reset，再往前加轉向閘（steering gates）；

- 記住板子的特性：未接線輸入經下拉電阻預設為 0、每個 NAND 輸出有 LED——可以逐級檢查訊號；
- 主動低輸入平常要**接常數 1**（板上有常數 1 腳位），測試時才短暫改接 0（或斷開 = 0）。

挑戰題

挑戰：把你的 D 閂鎖與其他同學的板子串成移位暫存器（shift register）?（需要 USB 分接器共用電源。）

天真的接法會失敗：把大家的主動高 D 閂鎖 $Q \to D$ 串成一列、共用同一條 E——$E = 1$ 時**整條鏈同時透明**，輸入值會像水流一樣「唰」地衝過**所有**閂鎖，每拍移動的格數取決於 E 高電位的時間長短——不可控。

正確做法（兩相時脈 / 領導—跟隨原理）：讓相鄰的閂鎖用**相反極性**的致能——奇數位置用主動高、偶數位置用主動低（或反之），全部接同一條 E：

- $E = 1$：奇數閂鎖打開接收前一級的值，偶數鎖住（擋水閘）；
- $E = 0$：偶數打開接收，奇數鎖住。

每個完整時脈週期，資料**恰好前進一級**。一對「主動高 + 主動低」閂鎖就是一個 D 正反器——這個挑戰其實是讓你們用人數湊出「n 位元移位暫存器 = n 個正反器串聯」的事實。**共用電源的原因：**兩塊板若各用各的 USB 電源，接地電位可能不同，高低電位的判讀基準就不一致，訊號跨板後可能被誤判——所以必須用 USB 分接器共用同一電源。

2　Logisim 的背叛（Treachery of Logisim）

題目

本節只在 NAND 板上做，不要用 Logisim。把你的 D 正反器中「連接兩個致能輸入之間、負責反相的那個 NAND」換成**五個** NAND（每個都接成 NOT）。驗證正反器行為「跟原來一模一樣」——畢竟 $\neg\neg x \equiv x$，對吧？……對吧？……哪裡出錯了？

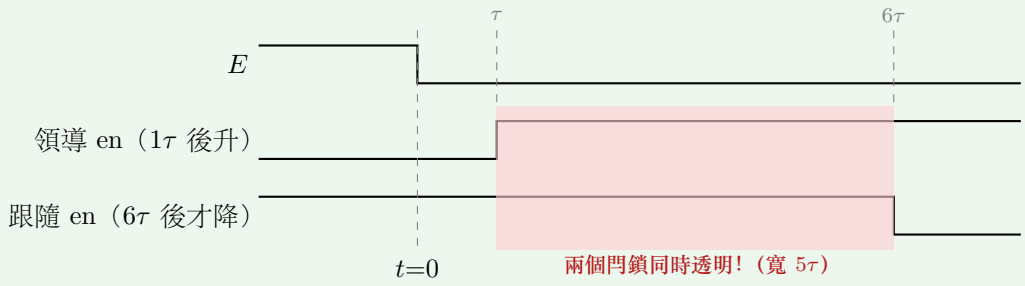
接著把那五個串聯的「NOT 閘」頭尾相接成**環**，你能解釋觀察到的行為嗎？

解答

實驗一：五重反相的正反器——為什麼壞掉？

邏輯上，五個 NOT 串聯 ≡ 一個 NOT （$\neg^5 x = \neg x$），電路的真值表完全沒變。但**物理上**，每個真實的 NAND 閘有**傳播延遲** τ（74HC 系列約 8～15 ns）——換成五個閘後，跟隨閂鎖的致能訊號比原來**晚了約** 4τ 才到。正反器的邊緣觸發性質正是敗在這裡。

回顧結構：領導 en = $\neg E$（經 1 個閘）、跟隨 en = ¬(領導 en)（原本經 1 個閘，現在經 5 個閘）。看 E **下降**的瞬間（設此刻為 $t = 0$）：

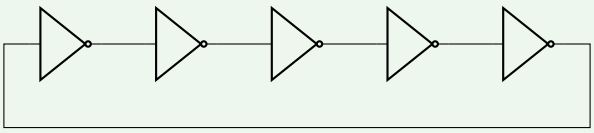


- $t = \tau$：領導 en 升起——**領導打開**，開始追蹤當前的 D；
- $t = 6\tau$：跟隨 en 才降下——在 $[\tau, 6\tau]$ 這段窗口內**兩個閂鎖同時透明**；
- 新的 D 值只需約 $2\tau \sim 3\tau$ 就能穿過領導閂鎖——比 5τ 的窗口短——於是它**一路衝到** Q（race-through，直衝）。

可觀察的症狀：正緣觸發正反器的 Q 應該**只在上升緣**改變。但現在：E 拉高後改變 D（照理 Q 不動），再把 E **放開（下降緣）**——Q 竟然跟著更新了！正反器彷彿「兩個邊緣都觸發」，邊緣觸發的保證徹底失效。

為什麼原本一個 NAND 就沒事? 原設計的窗口只有約 1τ（領導在 τ 打開、跟隨在 2τ 關閉），而資料穿過領導需要 $2\tau \sim 3\tau$——**來不及**穿過去窗口就關了。邊緣觸發的正確性其實是一場**賽跑**：「致能反相的延遲」必須**短於**「資料穿過領導的延遲」。一個 NAND 贏得了這場賽跑；五個 NAND 輸掉了。Logisim 預設把所有閘當成理想元件，不會如實模擬這場賽跑——所以這個實驗**只能**在真板子上做。

實驗二：五個 NOT 接成環——環形振盪器（ring oscillator）。



觀察到的現象：五個 LED **全部半亮**（既不是穩定的亮也不是穩定的暗）。

解釋：設環上某點的訊號為 x。繞環一圈經過**奇數**（5）次反相，回到原點時要求 $x = \neg x$——**沒有任何穩定狀態能滿足這個方程式**。電路唯一的出路是**永不停止地振盪**：一個「翻轉波前」沿著環不停繞圈，每繞一圈把每個節點翻轉一次。振盪週期 = 波前繞兩圈的時間（一圈翻成反相、再一圈翻回來）：

$$T = 2n\tau \quad \Rightarrow \quad f = \frac{1}{2n\tau} \approx \frac{1}{2 \times 5 \times 10\,\text{ns}} = 10\,\text{MHz}.$$

LED 每秒閃爍上千萬次，遠超人眼（約 60 Hz）的分辨極限，時間平均下來就是**半亮**。（對照：**偶數**個反相器成環要求 $x = x$——有兩個穩定狀態，那就是第 1 節的閂鎖！奇數環振盪、偶數環記憶，一體兩面。）

在 Logisim 上做這兩個實驗會怎樣? 五重反相的正反器會「正常運作」（因為模擬器不會輸掉那場賽跑）；反相器環則會讓 Logisim 直接報錯（“Oscillation apparent”）或以模擬器的節拍假振盪。**這就是標題「Logisim 的背叛」：當傳播延遲開始重要，模擬器給你的答案就不可信了**。所以課程規定：在 Logisim 中把 D 正反器當積木用時，一律用 memory 函式庫的**預建版本**——它內建額外邏輯來防止這類錯誤。

延伸知識：環形振盪器不是「壞掉的電路」——它是真實晶片裡的重要工具：製程監控（用振盪頻率量測該批晶片的實際閘延遲）、PLL 的壓控振盪器（改變供電電壓即可調頻）、甚至硬體亂數產生器（振盪相位的抖動是真隨機源）。同樣地，「直衝」問題正是現代 VLSI 時序分析（setup/hold time 檢查）要防範的核心對象。

3　記憶體（Memory）

題目

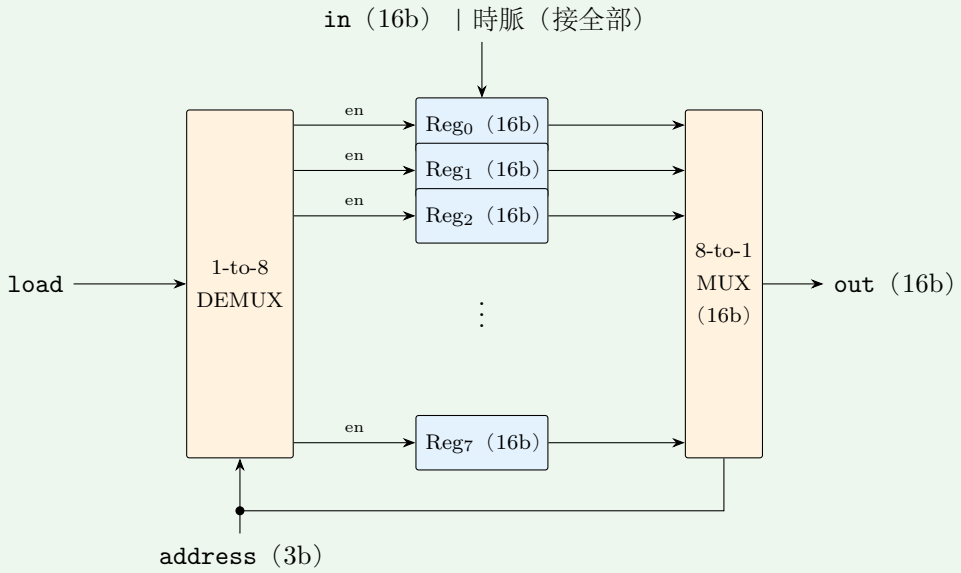
Hack CPU 有 32KB 記憶體，分成 16,384 個字組。用抽象化的力量逐步建出來：

1. 用 memory 函式庫的預建 **16 位元暫存器**建 **8 字組 RAM**：輸入 `in`（16 位元）、`address`（3 位元）、`load`（1 位元）、時脈；輸出 `out`（16 位元）。上升緣時 `out` 應為 `address` 處的值；若 `load` 為高則該處更新為 `in`；
2. 用 8 字組 RAM 作子電路建 **64 字組 RAM**（`address` 改為 6 位元）。提示：address 要拆成兩部分；
3. 用 64 字組 RAM 作子電路建 **32KB RAM**（14 位元位址）。提示：可再多建幾層子電路。

解答

核心觀念：RAM = 暫存器陣列 + 位址解碼。「讀」用**多工器**把被選中暫存器的輸出導到 `out`；「寫」用**解多工器**把 `load` 訊號只送給被選中的暫存器——上週的 Plexers 在這裡登場。

(1) 8 字組 RAM。



組裝細節：

- 8 顆 16 位元暫存器（memory 函式庫，Data Bits = 16）；`in` 匯流排**同時**接到全部 8 顆的 D 輸入、時脈也接到全部 8 顆——「大家都聽得到，但只有被點名的人動筆」；

- **寫入路徑**：1-to-8 解多工器（Plexers 函式庫，Select Bits = 3，Data Bits = 1），輸入接 `load`、選擇線接 `address`——只有第 `address` 顆暫存器的 enable 收到 `load`，其餘收到 0；
- **讀取路徑**：8-to-1 多工器（Select Bits = 3，**Data Bits = 16**），八個輸入接八顆暫存器的輸出、選擇線接 `address`、輸出即 `out`——讀取是組合邏輯，隨時反映被選暫存器的當前值，自然滿足「上升緣後 `out` 是該位址的值」；
- 驗證：對位址 3 寫入 `0x00FF`（`load`=1、觸發時脈）、改讀位址 5（應為 0）、讀回位址 3（應為 `0x00FF`）；`load`=0 時觸發時脈，任何值都不該改變。

(2) 64 字組 RAM——遞迴的第一步。
$64 = 8 \times 8$：放 **8 個 8 字組 RAM 子電路**，用 splitter 把 6 位元 `address` 拆成兩部分：

- **高 3 位 `address[5:3]`**：選「哪一塊」——接 1-to-8 DEMUX 分派 `load`，並接 8-to-1 MUX（16 位元）選哪塊的 `out`；
- **低 3 位 `address[2:0]`**：選「塊內哪一格」——**同時**接到全部 8 塊的 `address` 輸入；
- `in` 與時脈照樣廣播到全部 8 塊。

結構與 (1) **一模一樣**——只是「暫存器」換成了「8 字組 RAM」。這就是題目說的抽象化的力量。
(3) 32KB（16,384 字組）RAM——把遞迴走完。
$16384 = 8 \times 8 \times 8 \times 8 \times 4$。沿同一模式逐層往上蓋：

層	容量	位址位元	組成
RAM8	8 字組	3	8 顆暫存器
RAM64	64 字組	$3+3=6$	8 個 RAM8
RAM512	512 字組	$3+6=9$	8 個 RAM64
RAM4K	4096 字組	$3+9=12$	8 個 RAM512
RAM16K	**16384 字組**	$\mathbf{2+12=14}$	**4** 個 RAM4K

每一層的接法完全複製 (2)：高位選塊（DEMUX 分 load、MUX 合 out）、低位下傳、`in` ／時脈廣播。唯一的變化在最頂層：$16384 = 4 \times 4096$，所以高位只有 **2 位元**，用 **1-to-4** DEMUX 與 **4-to-1** MUX（上週挑戰題做過的那兩個！）。這正是 nand2tetris 教材中 RAM8 → RAM64 → RAM512 → RAM4K → RAM16K 的標準建法。

容量檢算：16384 字組 × 16 位元/字組 = 262144 位元 = 32768 位元組 = 32 KB ✓。位址位元檢算：$2^{14} = 16384$ ✓。**觀察**：讀取延遲隨層數線性增加（每層多過一顆 MUX）——真實記憶體用二維行列解碼（第四週的 wordline ／ bitline）而非純樹狀結構，但「解碼器選格子」的原理相同。

4　其他正反器（Other Flip-Flops）

題目

從預建 D 正反器出發，加上少量邏輯建出：

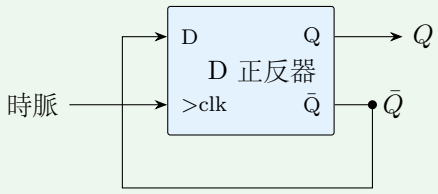
- **T 正反器**：唯一輸入是時脈；輸出 Q 與 $Q' = \neg Q$，且 Q 是頻率為輸入時脈**一半**的時脈訊號（T = toggle）；
- **JK 正反器**：輸入 J、K 與時脈。上升緣時：J=K=0 保持、J=0, K=1 重設（Q=0）、J=1, K=0 設定（Q=1）、J=K=1 翻轉。

建好後用 wiring 函式庫的 clock 輸入測試（Ctrl+T 手動觸發時脈緣）。

解答

(1) T 正反器：一條回饋線就夠（0 個額外閘）。

把 D 正反器的 $\bar{Q}$ **接回** D：



原理：每個上升緣，正反器把「當前 Q 的相反值」存進去（$Q_{next} = \bar{Q}$）——**每拍翻轉一次**。Q 完成一個完整週期（$0 \to 1 \to 0$）需要**兩個**時脈上升緣，所以 Q 的頻率恰是時脈的一半 ✓。這也是**二進位計數器**的最低位：把 n 個 T 正反器串起來（前一級的 Q 當下一級的時脈）就是除 2^n 分頻器。

為什麼**必須**用正反器而不能用閂鎖？若用透明閂鎖，$E = 1$ 期間 $Q \to \bar{Q} \to Q \to \cdots$ 會**不受控地連續振盪**（正是第 2 節環形振盪器的翻版）；邊緣觸發保證每個邊緣**恰好翻一次**。

(2) JK 正反器。

先把行為表翻譯成「下一個 Q」的布林式。逐列考察 Q_{next} 與 J、K、Q 的關係：

J	K	Q_{next}	說明
0	0	Q	保持
0	1	0	重設
1	0	1	設定
1	1	$\bar{Q}$	翻轉

四列合併成一條**特徵方程式**：

$$\boxed{Q_{next} \;=\; (J \wedge \bar{Q}) \;\vee\; (\neg K \wedge Q)}$$

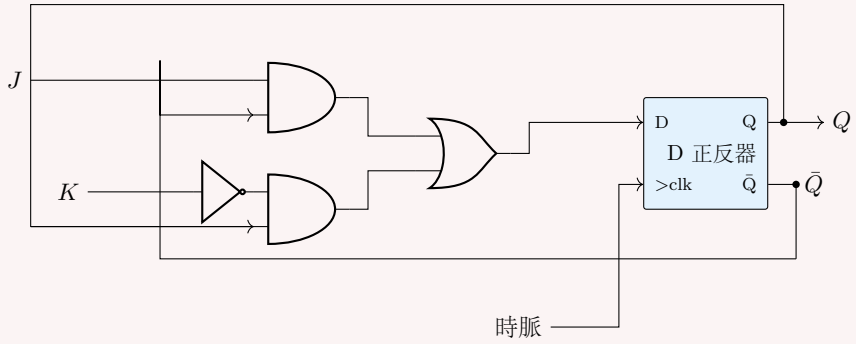
驗證：J=K=0：$0 \vee Q = Q$ ✓；J=0, K=1：$0 \vee 0 = 0$ ✓；J=1, K=0：$\bar{Q} \vee Q = 1$ ✓；J=K=1：$\bar{Q} \vee 0 = \bar{Q}$ ✓。

把這條式子接到 D 輸入即可，時脈**原封不動**直通 D 正反器（題目的提示）。

挑戰題

挑戰：只用四個 AND/OR/NOT 閘＋ D 正反器實作 JK。

數一數特徵方程式需要的閘：$\bar{K}$（NOT ×1）、$J \wedge \bar{Q}$（AND）、$\bar{K} \wedge Q$（AND）、最後 OR——**恰好 4 個**。$\bar{Q}$ **不用**閘：D 正反器本來就提供 $\bar{Q}$ 輸出！



接線總結：$D = (J \wedge \bar{Q}) \vee (\bar{K} \wedge Q)$，其中 Q、$\bar{Q}$ 取自正反器自身的兩個輸出（回饋），時脈直通。**為什麼時脈必須直通？**若試圖把時脈「閘控」（例如 J、K 與時脈先 AND），會製造出毛刺（glitch）與加閘延遲，破壞邊緣觸發時序——讓 D 正反器全權負責「何時」、外部組合邏輯只負責「什麼值」，是同步設計的基本紀律。

Logisim 測試：接 wiring 函式庫的 clock 元件，按 Ctrl+T 逐緣觸發：J=1, K=0 打一拍 $\rightarrow Q = 1$；J=0, K=1 打一拍 $\rightarrow Q = 0$；J=K=1 連打數拍 $\rightarrow Q$ 每拍翻轉（變成 T 正反器）；J=K=0 連打數拍 $\rightarrow Q$ 不動。**歷史註腳：**JK 是「萬能正反器」——四種模式涵蓋 SR（無禁止態）、D（$K = \bar{J}$）與 T（J=K=1）；TTL 時代（如 7476）曾是標準元件，現代設計則幾乎只用 D 正反器＋組合邏輯（正如本題的建法）。

A 附錄：速查表

A.1 本 Lab 的 NAND 計數

電路	NAND 數	備註
主動低 R-S 閂鎖	2	交叉耦合核心
主動高 R-S 閂鎖	4	核心 + 2 個輸入反相
主動高 D 閂鎖	4	轉向閘共用 $\bar{D}$
主動低 D 閂鎖	5	加 1 個致能反相
正緣觸發 D 正反器	9	領導（5）+ 跟隨（4）
負緣觸發 D 正反器	9	反相器移到跟隨側
T 正反器	+0	$\bar{Q} \to D$ 回饋線
JK 正反器	+4（AND/OR/NOT）	時脈直通

A.2 行為對照表

元件	何時更新	特徵
R-S 閂鎖	控制輸入動作期間	有禁止態
D 閂鎖	E 有效期間（透明）	無禁止態、位準觸發
D 正反器	僅時脈邊緣	$Q_{next} = D$
T 正反器	僅時脈邊緣	$Q_{next} = \bar{Q}$（分頻 2）
JK 正反器	僅時脈邊緣	$Q_{next} = J\bar{Q} \vee \bar{K}Q$

A.3 關鍵公式與數字

主題	內容
環形振盪器頻率	$f = 1/(2n\tau)$；$n = 5$、$\tau \approx 10\,\text{ns}$ 時 $\approx 10\,\text{MHz}$
奇偶反相環	奇數環：無穩態（振盪）；偶數環：雙穩態（記憶）
直衝條件	致能反相延遲 > 資料穿越領導閂鎖的延遲
RAM 遞迴	8^k 字組 RAM = 8 個 8^{k-1} 字組 RAM + DEMUX/MUX
Hack RAM	$2^{14} = 16384$ 字組 $\times 16$ 位元 $= 32\,\text{KB}$
JK 特徵方程式	$Q_{next} = J\bar{Q} \vee \bar{K}Q$

A.4 參考資料

- COMSM1302 第三週講義（R-S 閂鎖、D 閂鎖與正反器、暫存器與 RAM、傳播延遲與亞穩態），University of Bristol.
- Nisan & Schocken, *The Elements of Computing Systems*（nand2tetris），Ch. 3（Memory）——RAM8 → RAM16K 的階層建法.
- Ring oscillator（Wikipedia）——奇數反相環的振盪原理與工程應用.
- Logisim User Guide（memory 函式庫：register、D flip-flop；wiring 函式庫：clock、splitter）.