正確學會 改訂新版 ディジタル回路と Verilog HDL

Verilog 的16堂課

第 7 章

序向電路 I

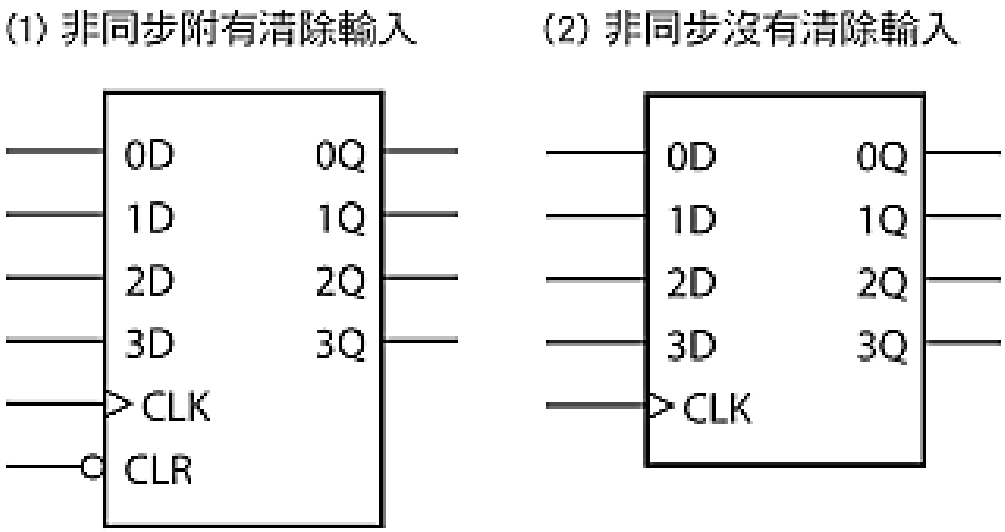
本章重點

計數器區分有非同步與同步計數器，前者的電路與動作較為容易理解，而後者在合成電路時會比較有效率

- 7.1 暫存器
- 7.2 移位暫存器
- 7.3 非同步計數器
- 7.4 同步計數器

7.1 暫存器

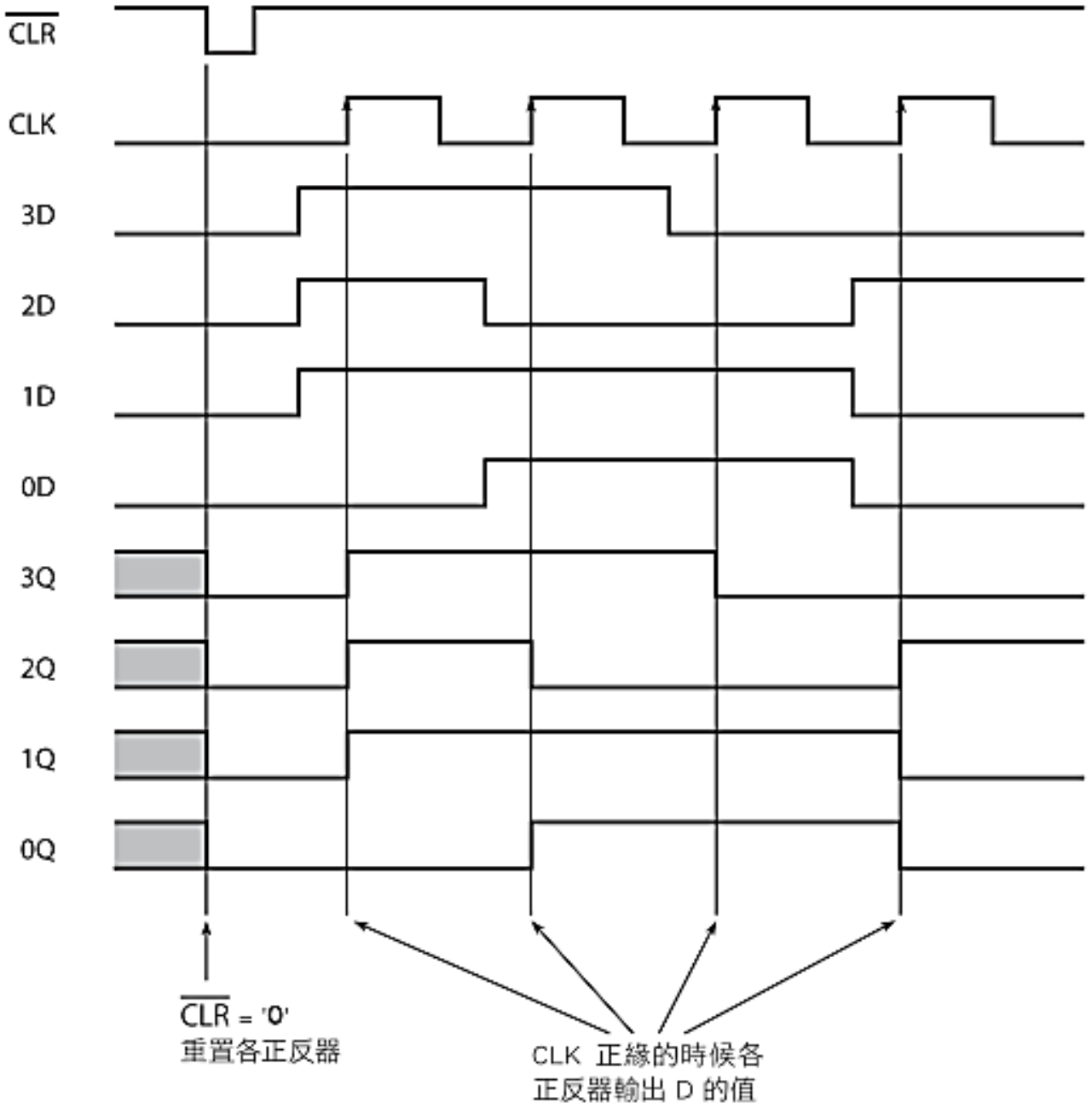
- 正反器是只能記憶 1 位元的情報記憶單元,要記憶 2 位元以上的話就多個正反氣構成記憶電路,也就是暫存器(register)
- 暫存器一般都是由「同步 D 型正反器」或「非同步 R+同步 D 型正反器」所構成



▲ 圖 7.1　4 位元暫存器

- 暫存器內部電路有以下的特徵
  - 輸出只有各位元的 Q
  - 由多數的正反器所組成
  - 各正反器的 CLK 由同一個信號來統一產生
  - 各正反器的 R 是由同一個信號來統一清除
  - 有附加清除輸入的暫存器電路符號是非同步 R + 同步 D 型正反器
  - 沒有清除輸入的就只有同步 D 型正反器
  - 多數的暫存器都是「正緣觸發」
  - 有內部使用到閂鎖的暫存器

- 用時序圖表示暫存器動作, CLR (清除) 為 ON 把各個正反器重置, CLK (時脈) 為正緣把正反器的 Q 設為 D 的值
- 扇入 (fan in) 是指邏輯元件輸入訊號端的數量
- 此處 CLK、CLR 訊號輸入要提供 4 個正反器使用，要加上緩衝器電路提高電路負載能力



▲圖 7.3　4 位元暫存器的時序圖

7.1.2 Verilog HDL 描述

程式 7.1　4 位元暫存器的測試平台　LST7_1.V

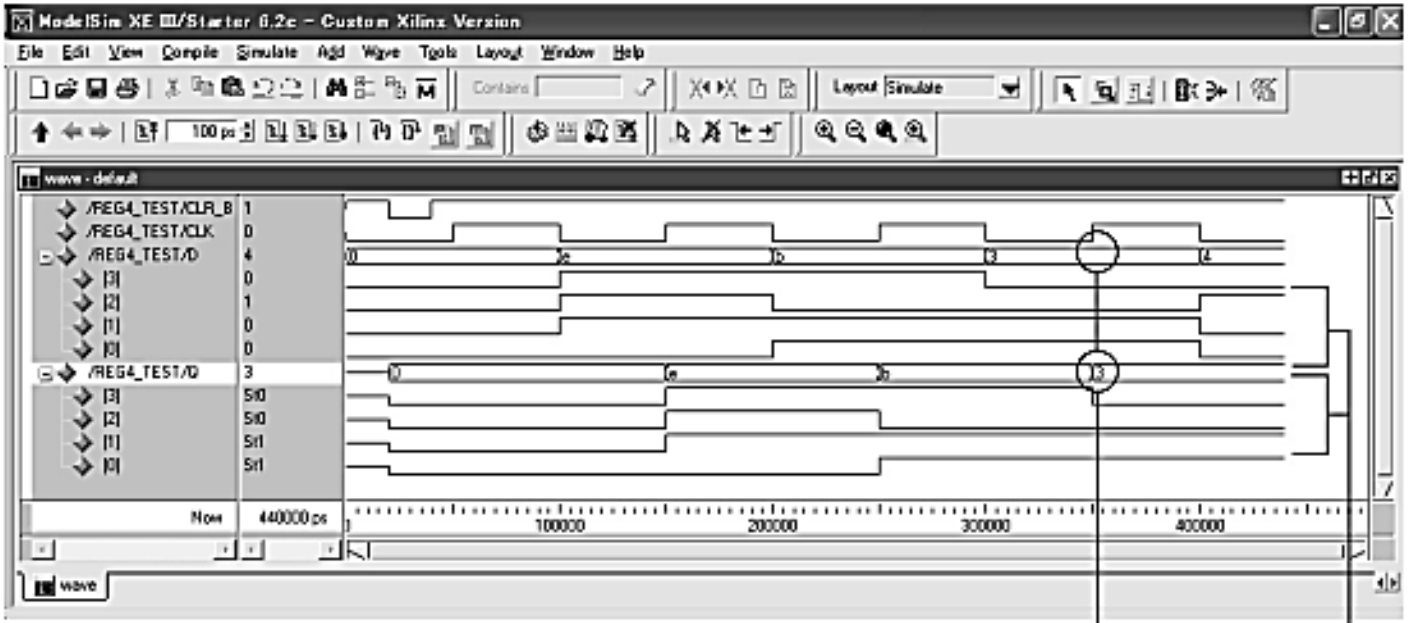
```
`timescale      1ns/1ns
module      REG4_TEST;
reg         CLR_B, CLK;
reg         [3:0] D;
wire        [3:0] Q;
parameter   STEP = 100;

            REG4    REG4    ( CLR_B, D, CLK, Q );
            always  #( STEP/2 )     CLK     = ~CLK;
            initial
                    begin
                                    CLR_B = 1; D = 0; CLK = 0;
                    #( STEP/5 )     CLR_B = 0;
                    #( STEP/5 )     CLR_B = 1;
                    #( STEP*3/5 )   D     = 4'b1110;
                    #STEP           D     = 4'b1011;
                    #STEP           D     = 4'b0011;
                    #STEP           D     = 4'b0100;
                    #( STEP/2-10 )  $finish;
            end
endmodule
```

*1 4 位元的匯流排形式

*2 2 進制 4 位元

程式 7.2　4 位元暫存器的 Verilog HDL 描述　LST7_2.V

```
/*          REG4              */
module      REG4              ( CLR _ B,  D,  CLK,  Q );
input       CLR _ B,  CLK;
input       [3:0]  D;
output      [3:0]  Q;          *4
reg         [3:0]  Q;  ←——— 暫存器也是 4 位元
            always             @( posedge CLK or negedge CLR _ B   )
                               Q <= ( !CLR _ B )?      0:  D;
endmodule
                                        *3 CLR _ B= '0' 的話 Q<=0;
                                           CLR _ B= '1' 的話 Q<=D;
```



▲ 圖 7.4　4 位元暫存器的模擬結果

16 進制匯流排表示　　逐位元表示

7.1.3 暫存器的 Verilog HDL 階層設計

程式 7.3　用階層設計觀念寫的 4 位元暫存器的 Verilog HDL 描述　LST7_3.V

```
/*          REG4        */
module      REG4        ( CLR_B, D, CLK, Q );
input       CLR_B, CLK;
input       [3:0] D;
output      [3:0] Q;
wire [      3:0] Q_B;
R_SYDFF     R_SYDFF0 ( CLR_B, D[0], CLK, Q[0], Q_B[0] ),
            R_SYDFF1 ( CLR_B, D[1], CLK, Q[1], Q_B[1] ),
            R_SYDFF2 ( CLR_B, D[2], CLK, Q[2], Q_B[2] ),
            R_SYDFF3 ( CLR_B, D[3], CLK, Q[3], Q_B[3] );
endmodule
```

元件名稱改變

輸入 D 的最低位元

*1

*2

不用 reg 而用 wire 來宣告 Q_B

輸入 D 的最高位元

不是分號而是逗號

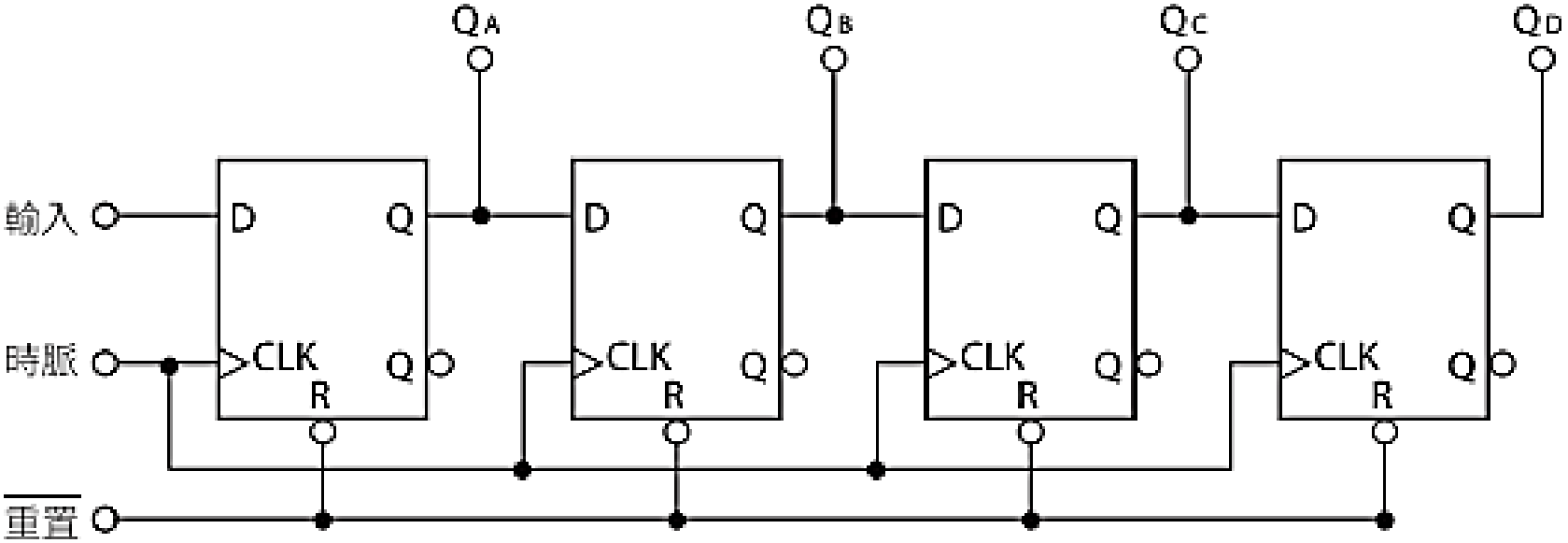
- 呼叫模組的語法原則上是「模組名 元件名稱 (參數群);」，不過因為這邊多次呼叫出同一個模組，所以語法變成像程式一樣用 ',' 來區別各個元件名稱,當然參數群的描述順序要跟模組「R_SYDFF」對的起來才行。
- 因為各位元的正反器存在於簡碼化後的「R_SYDFF」裡面，所以 reg 變數宣告要在「R_SYDFF」裡面做才是正確的。

7.2 移位暫存器

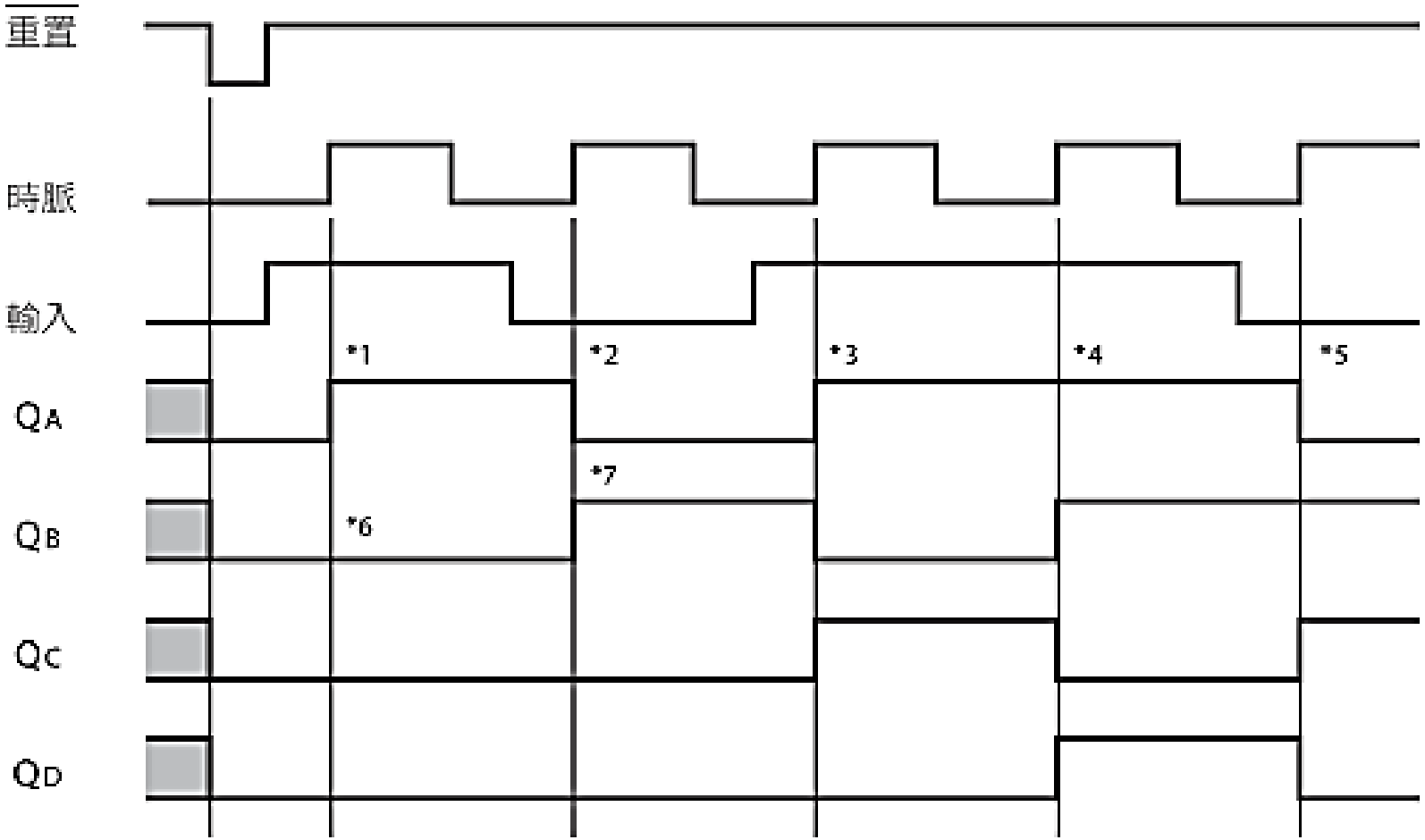
- 移位暫存器 (shift register) 是「根據時脈的輸入，把現在正反器記憶的數位資料以 1 位元為單位左移或右移的電路」，主要用在信號的串列並列變換

串列輸入並列輸出移位暫存器

- 此處使用非同步 R + 同步 D 型正反器的 4 位元串列輸入並列輸出移位暫存器



- 重置為 ON 會重置各正反器,之後 Q_A 會發生以下動作
  - 時脈正緣時 D 為 ' 1', 所以輸出為 '1‘
  - 時脈正緣時 D 為 ' 0', 所以輸出為 '0‘
- 「Q_B 」則是以下動作
  - 時脈正緣之前 D的值就由 Q 輸出

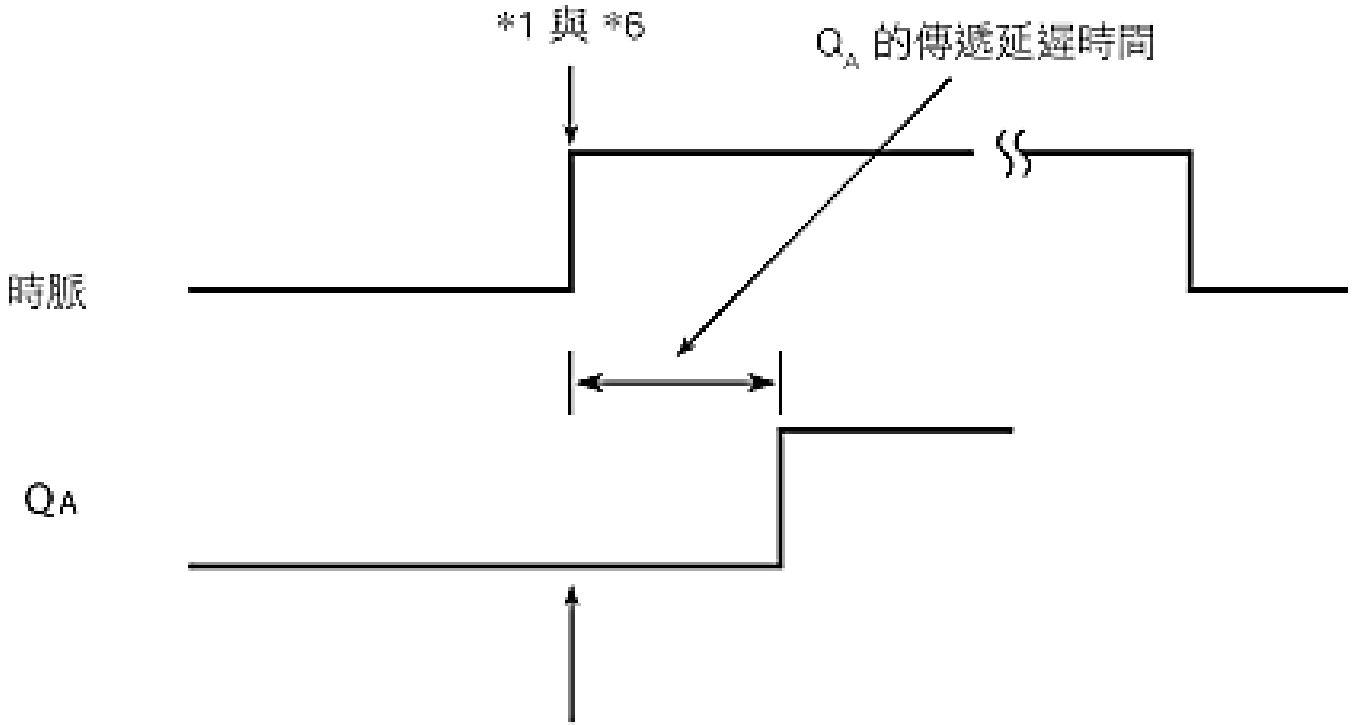


▲ 圖 7.5　4 位元串列輸入並列輸出移位暫存器

- 串列 (serial)：一次處理 1 位元的數位情報
- 並列 (parallel)：一次處理多個位元的數位情報

考慮正反器的傳遞延遲時間

- 考量正反器的傳遞延遲時間來解決剛剛Q_B的問題

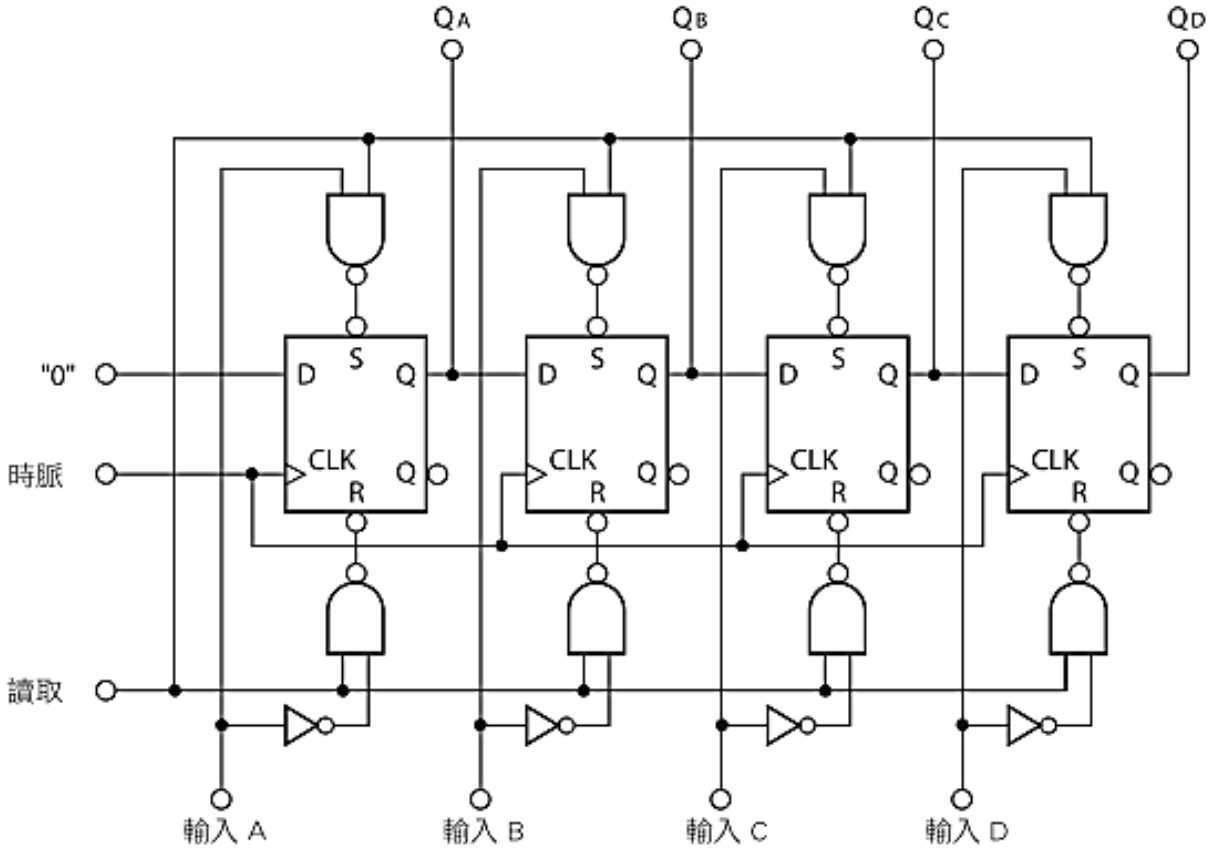


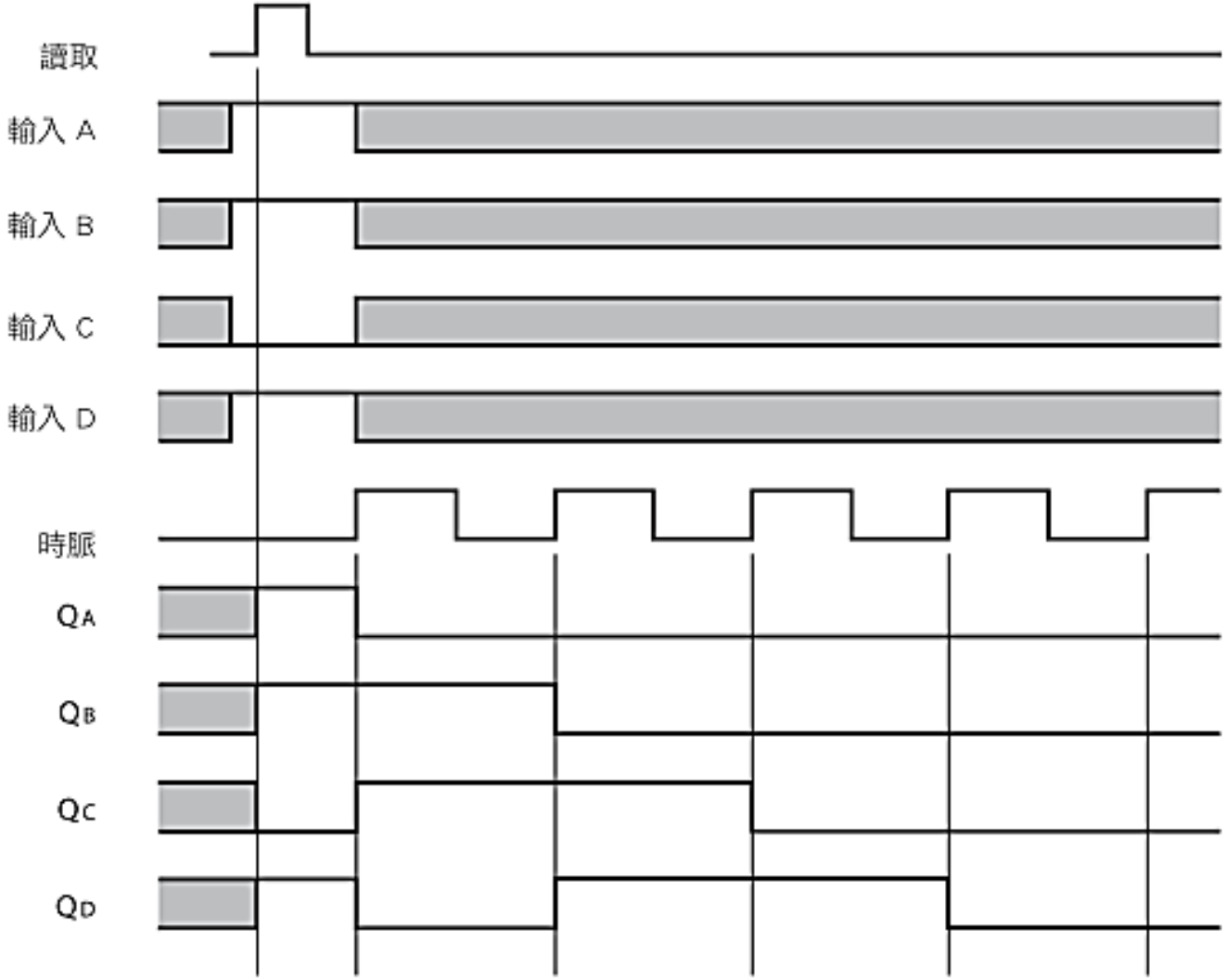
Q_B 是根據時脈正緣時候的輸入 D（也就是 Q_A）來決定輸出。
因為有傳遞延遲時間的存在，這時候 Q_A 還是 '0'。

▲ 圖 7.6　正反器的傳遞延遲時間

並列輸入串列輸出移位暫存器

- 此處使用非同步 RS + 同步 D 型正反器 4 位元並列輸入串列輸出移位暫存器 (parallel in serial out shift register)





▲ 圖 7.7　4 位元並列輸入串列輸出移位暫存器

- 讀取輸入與非同步 RS 設定好各正反器的初始值之後，即開始移位的動作

$\overline{S}$	$\overline{R}$	CLK	D	Q	Q'	
0	0	—	—	—	—	
0	1	—	—	—	1	禁止輸入
1	0	—	—	—	0	設置
1	1	1	—	1	1	重置
1	1	1	—	0	0	
1	1	0	—	1	1	
1	1	0	—	0	0	
1	1	↑	1	—	1	$\overline{S}$= '1', $\overline{R}$= '1' 且 CLK 為正緣時, D 的值由 Q 輸出
1	1	↑	0	—	0	

▲ 表 7.1　非同步 RS + 同步 D 型正反器的特性表

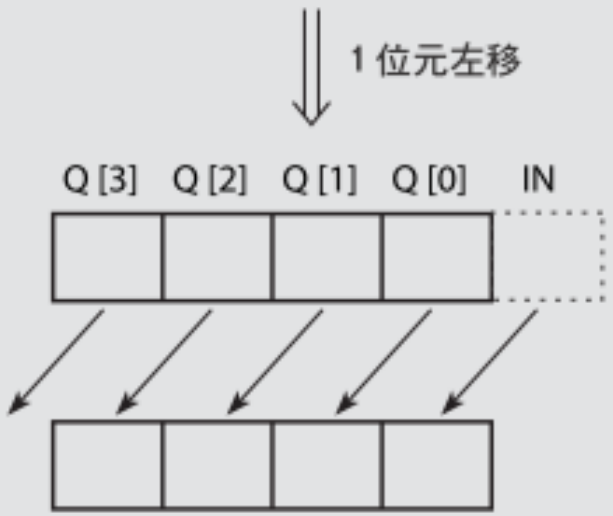
7.2.2 移位暫存器的 Verilog HDL 描述

串列輸入並列輸出移位暫存器

- 4 位元串列輸入並列輸出移位暫存器的 Verilog HDL重點為狀態的描述
- 根據「{Q, IN}」的連結，IN 會移位到 Q 的最低位元「Q[0]」
- 「Q[3:0]」會全部往左移位 1 位元成為「Q[4:1]」

程式 7.5　4 位元串列輸入並列輸出移位暫存器的 Verilog HDL 描述　LST7_5.V

```
/*       SIN _ POUT _ SHIFT      */
module   SIN _ POUT _ SHIFT      ( RESET _ B, IN, CLK, Q );
input    RESET _ B, CLK, IN;
output   [3:0] Q;
reg      [3:0] Q;
         always       @( posedge CLK or negedge RESET _ B  )
                      Q <= ( !RESET _ B )?   0: { Q, IN };
endmodule
```

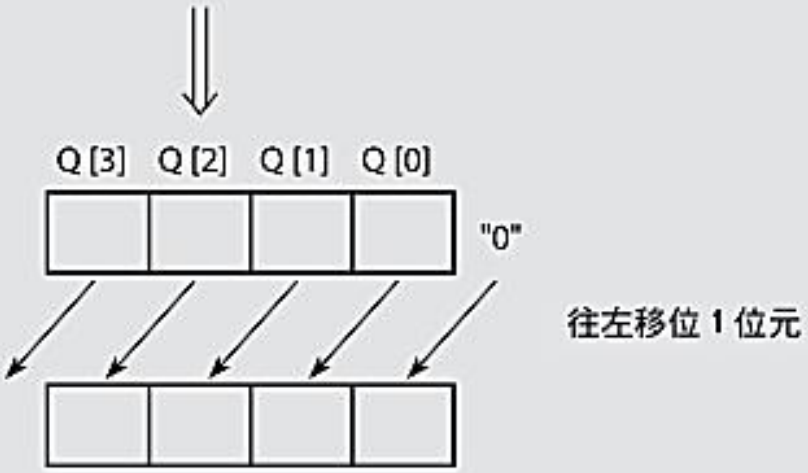


並列輸入串列輸出移位暫存器

- 狀態描述用「{Q, 0}」或用移位運算子「<<」做連接

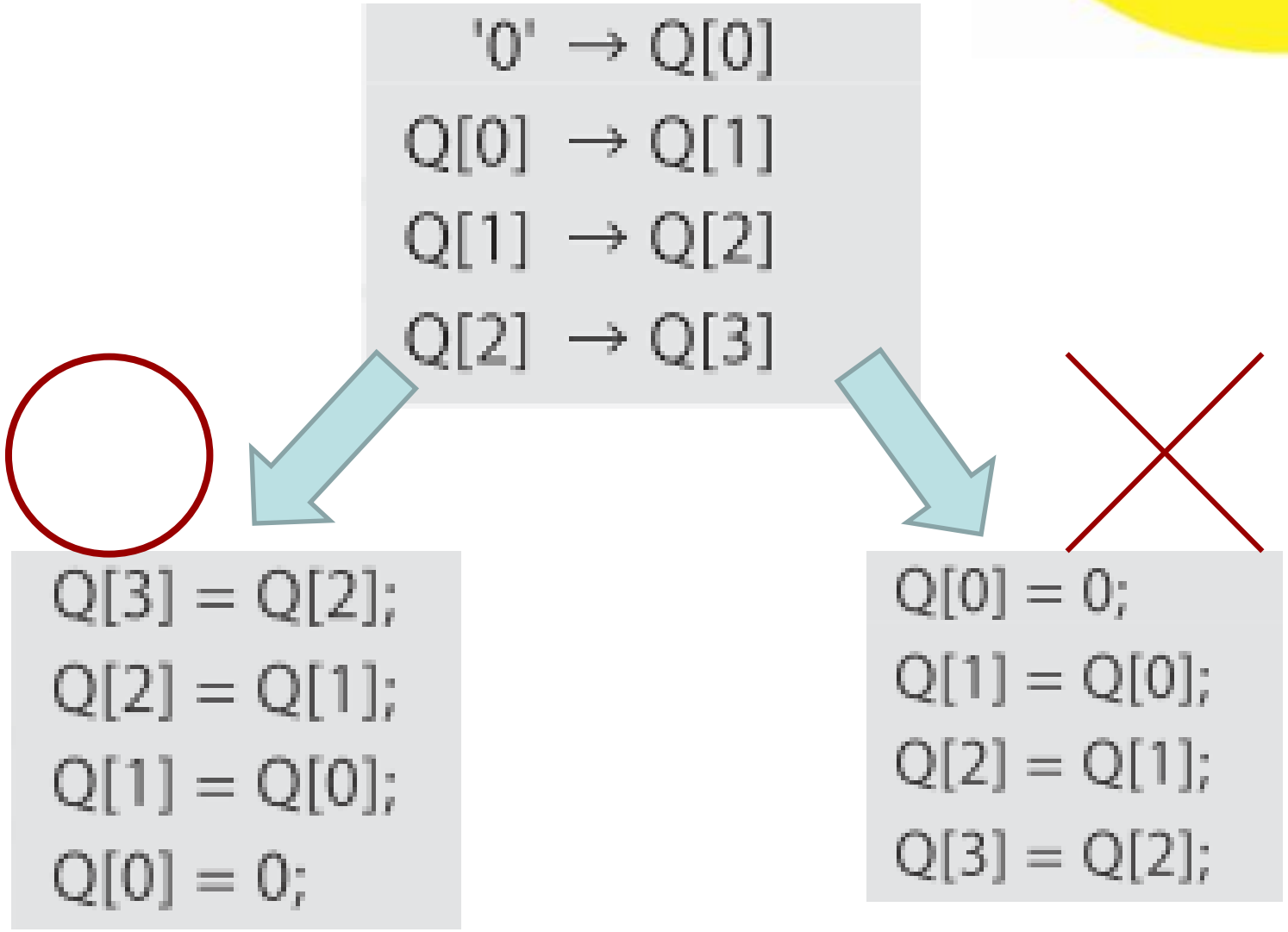
程式 7.8　4 位元並列輸入串列輸出移位暫存器的 Verilog HDL 描述　LST7_8.V

```
/*        PIN_SOUT_SHIFT          */
module    PIN_SOUT_SHIFT          ( LOAD, IN, CLK, Q );
input     LOAD, CLK;
input     [3:0] IN;
output    [3:0] Q;
reg       [3:0] Q;
          always   @( posedge CLK or posedge LOAD  )
                   if   ( LOAD )
                        Q <= IN;
                   else
                   Q <= Q << 1;              // Q <= { Q, 0 } 也可以
endmodule
```



'=' 與 '<='

- 到目前為止的應用容易產生 assign 用 '=', always 用 '<=' 這樣的錯覺
  - '=' 為阻礙指定, 是直接把右側 (RHS) 的值代入到左側 (LHS)
  - <= 為無阻礙指定, 不會馬上把右側 (RHS) 的值代入到左側 (LHS) ,只會先做排程的動作, 然後實行下面的程式


'0' → Q[0]
Q[0] → Q[1]
Q[1] → Q[2]
Q[2] → Q[3]
Q[3] = Q[2];
Q[2] = Q[1];
Q[1] = Q[0];
Q[0] = 0;
Q[0] = 0;
Q[1] = Q[0];
Q[2] = Q[1];
Q[3] = Q[2];

7.3 非同步計數器

- 計數器 (counter) 是每當信號輸入的時候, 把目前電路記憶的值 +1 (遞增) 或 -1 (遞減) 的電路
- 遞增動作的電路稱為上數計數器
- 遞減動作的電路稱為下數計數器
- 可切換遞增與遞減的電路則稱為上下數計數器

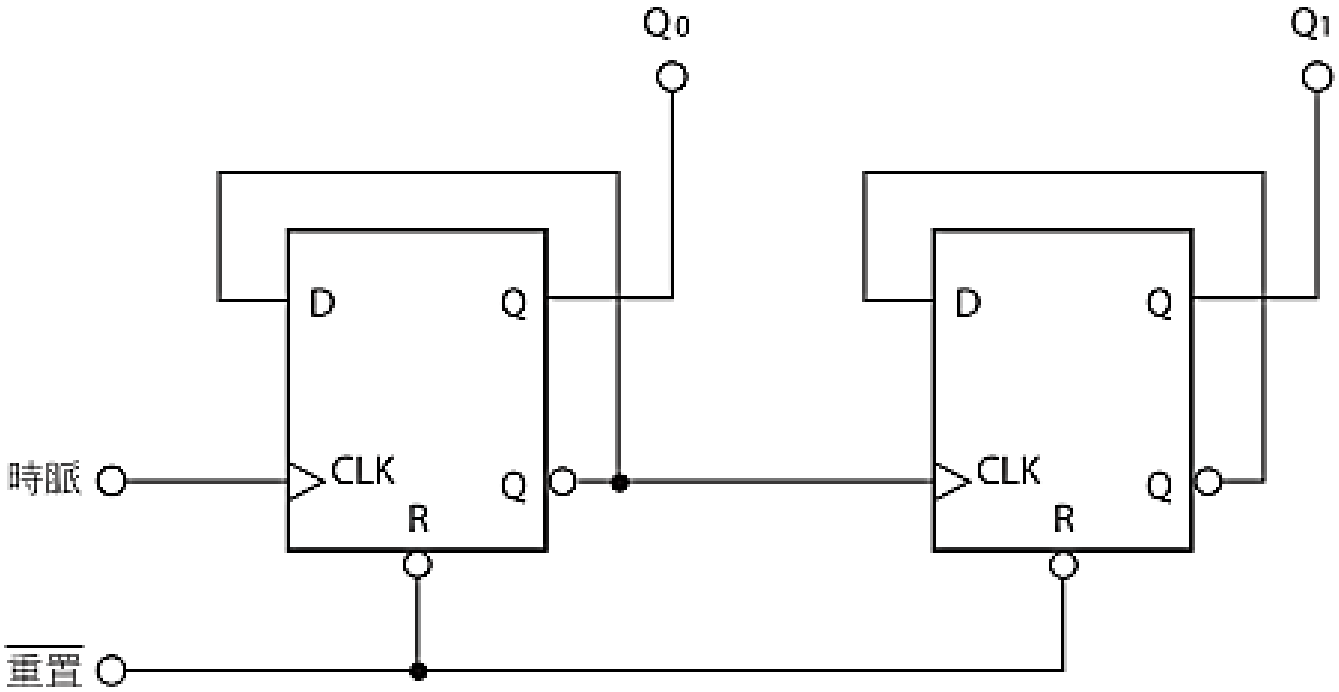
7.3.1 非同步 mode-2 ^{n} 計數器

mode-2 計數器

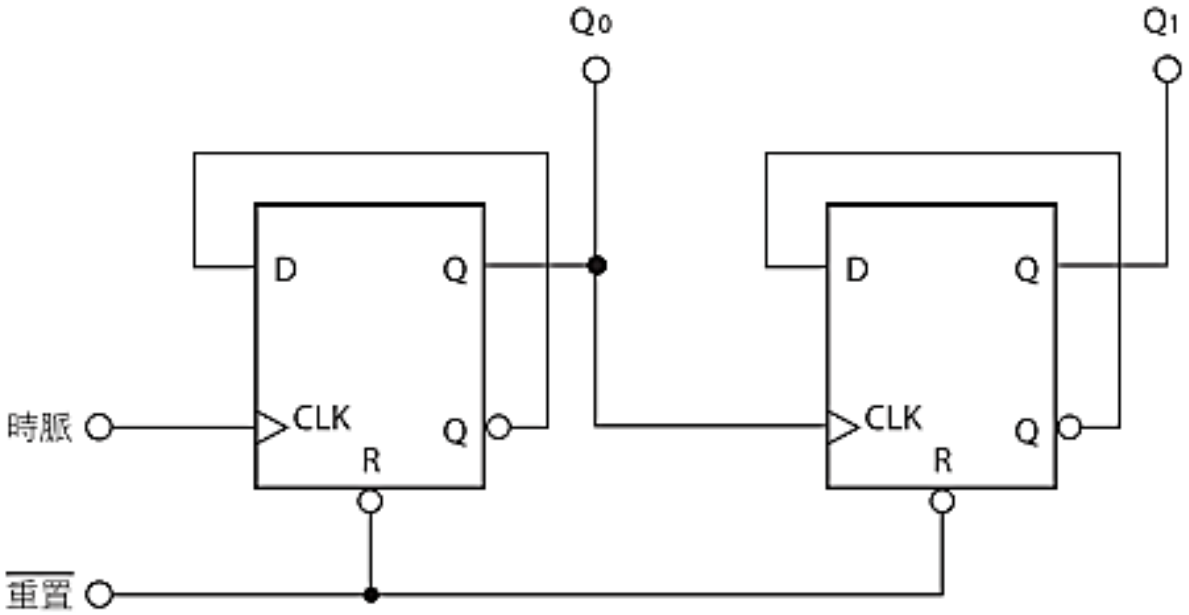
- mode-2 計數器稱為二進位計數器，時脈觸發輸出就會有 '0'→ '1'→ '0'… 的變化
- mode-2 計數器可以由許多不同的正反器所構成
- mode-2 計數器的輸出變化為 '0'→'1'→'0'…，所以不管是上數計數器或者下數計數器都是同樣的電路。負緣觸發的正反器也會是同樣的電路。

mode-4 計數器

- 兩個 mode-2 計數器接在一起, 可組成 mode-4 計數器
- 當時脈變化, 輸出的兩個位元, [Q 1 , Q 0] 會有 [0, 0]→[0, 1]→[1, 0]→[1, 1]→[0, 0]…的變化



- mode-4 下數計數器的接到 Q_1 的 CLK 變成是 Q_0
- mode-4 下數計數器的[Q_1, Q_0] 根據時脈的輸入會有 [0, 0]→[1, 1]→[1, 0]→[0, 1]→[0, 0]… 的變化, 用 10 進制來看就是 '0'→'3'→'2'→'1'→'0'…

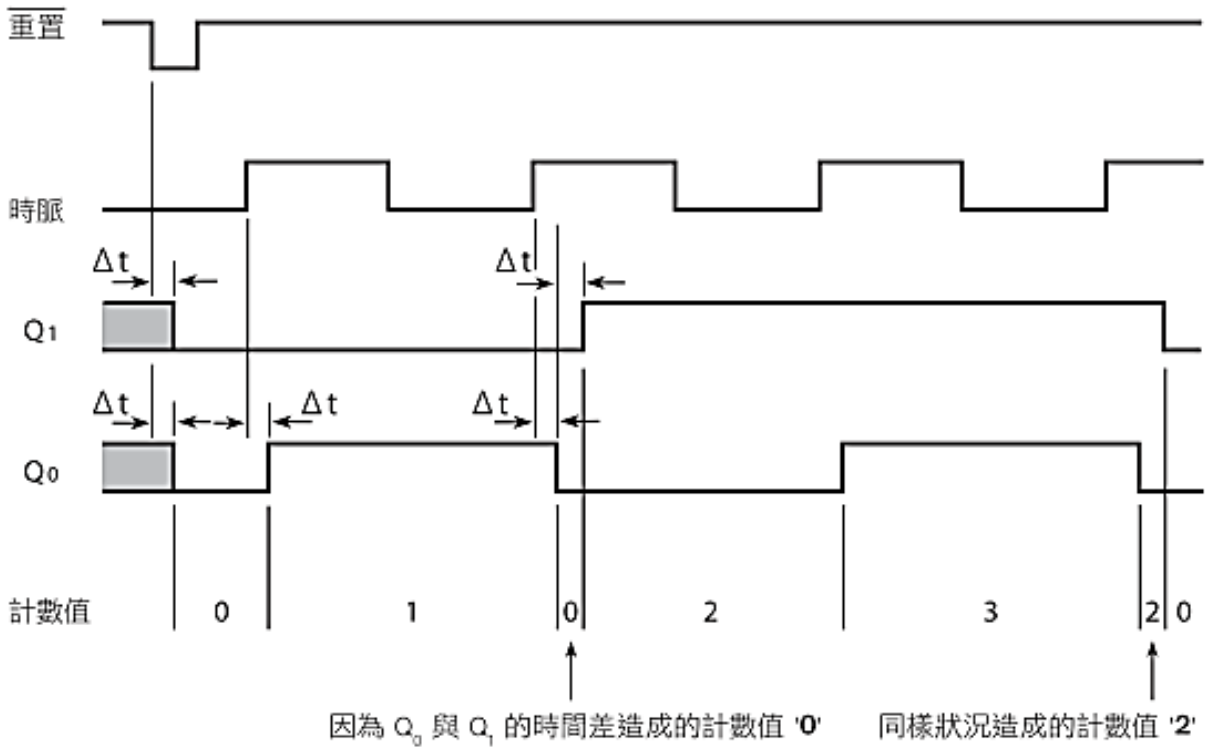


mode-2^n 計數器的構成

- 兩個 mode-2 計數器構成 mode-4 計數器, 三個構成 mode-8 計數器, 4 個構成 mode-16 計數器, n 個就可以構成 mode-2^n 計數器

非同步計數器的特徵

- 只要連接多個 mode-2 計數器就可以很容易的構成 mode-2 n 計數器。
- 正反器的輸出會接到下一級的 CLK, 所以正反器的傳遞延遲時間會一直累加。



非同步計數器的 Verilog HDL 描述

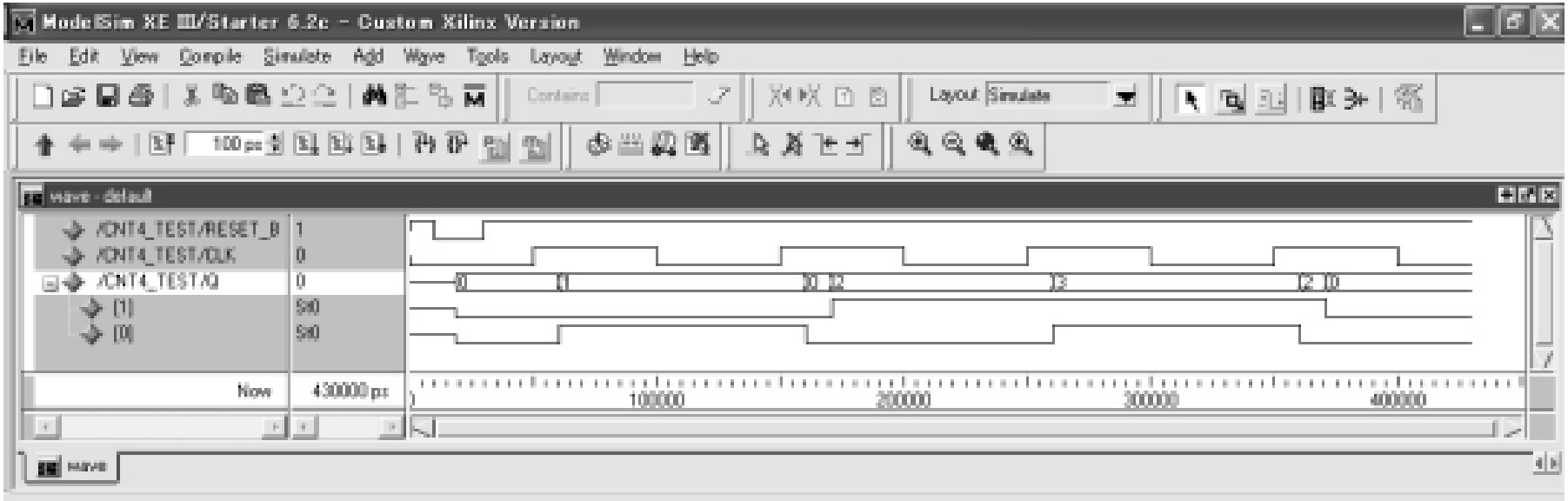
程式 7.11　非同步 mode-16 上數計數器的 Verilog HDL 描述　LST7_11.V

```
/*          CNT16           */
module      CNT16           ( RESET_B, CLK, Q );
input       RESET_B, CLK;
output      [3:0] Q;
wire        [3:0] Q_B;                                          *1      *2
            R_SYDFF         R_SYDFF0  ( RESET_B, Q_B[0],        CLK,    Q[0], Q_B[0] ),
                            R_SYDFF1  ( RESET_B, Q_B[1],  Q_B[0],       Q[1], Q_B[1] ),
                            R_SYDFF2  ( RESET_B, Q_B[2],  Q_B[1],       Q[2], Q_B[2] ),
                            R_SYDFF3  ( RESET_B, Q_B[3],  Q_B[2],       Q[3], Q_B[3] );
endmodule
```

- 模組「R_SYDFF」裡面宣告把各個 D 與各個正反器的 $\overline{Q}$ 接在一起
- CLK 跟上一級正反器的 $\overline{Q}$ 接在一起

用 Verilog HDL 來確認傳遞延遲時間的累加

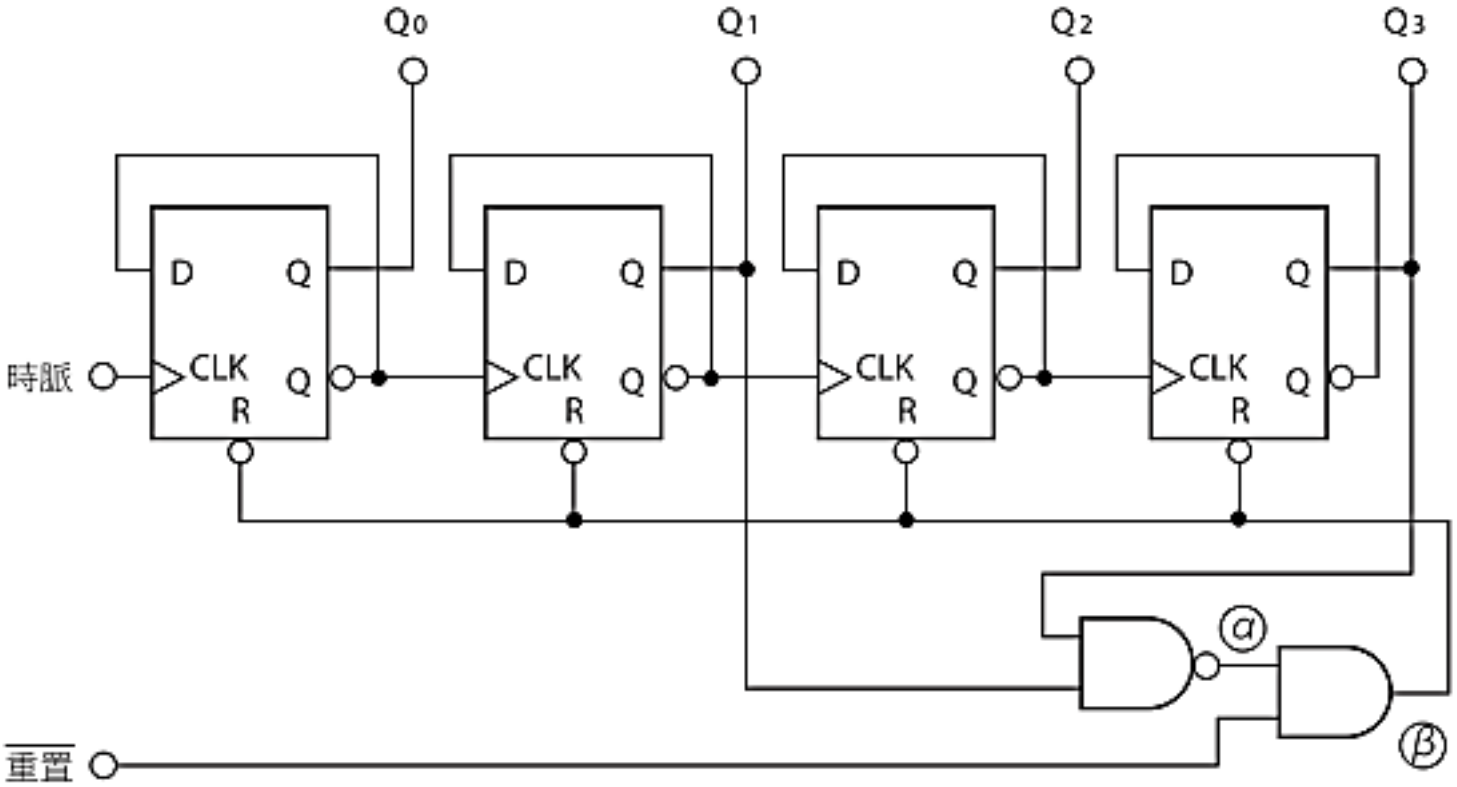
- 把 delay 值放入到正反器的描述裡，確認傳遞延遲時間的累加與計數值變化的狀況



▲ 圖 7.18 考慮傳遞延遲時間的非同步 mode-4 上數計數器的模擬結果

7.3.2 非同步 mode-N 計數器
mode-10 上數計數器

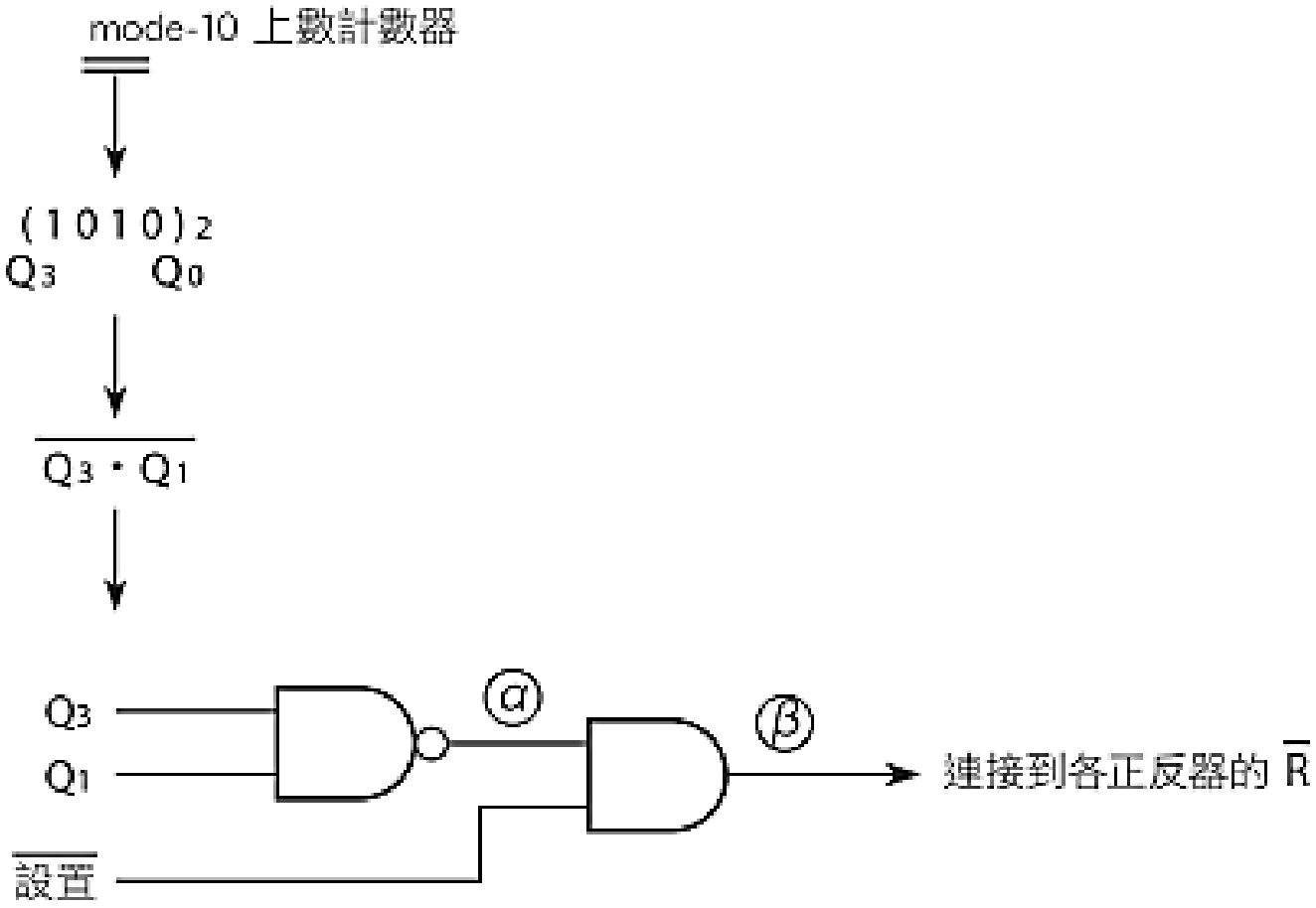
- mode-10 計數器是根據時脈每次的正緣會從 '0'→'1'→'2'→… 一直上數, 到 '9' 之後就會再一次回到 '0'



- 輸入「重置」的時候計數值變為 '0'。這是因為「重置」的 '0' 透過 AND 電路讓輸出的 β 也為 '0', 所以各個正反器都被重置了
- 之後每當時脈正緣的時候計數值就 +1, 會一直遞增到 ' 9'。到這邊為止的動作都跟 mode-16 上數計數器一樣
- 再一次的時脈正緣則會讓計數值回到 ' 0'

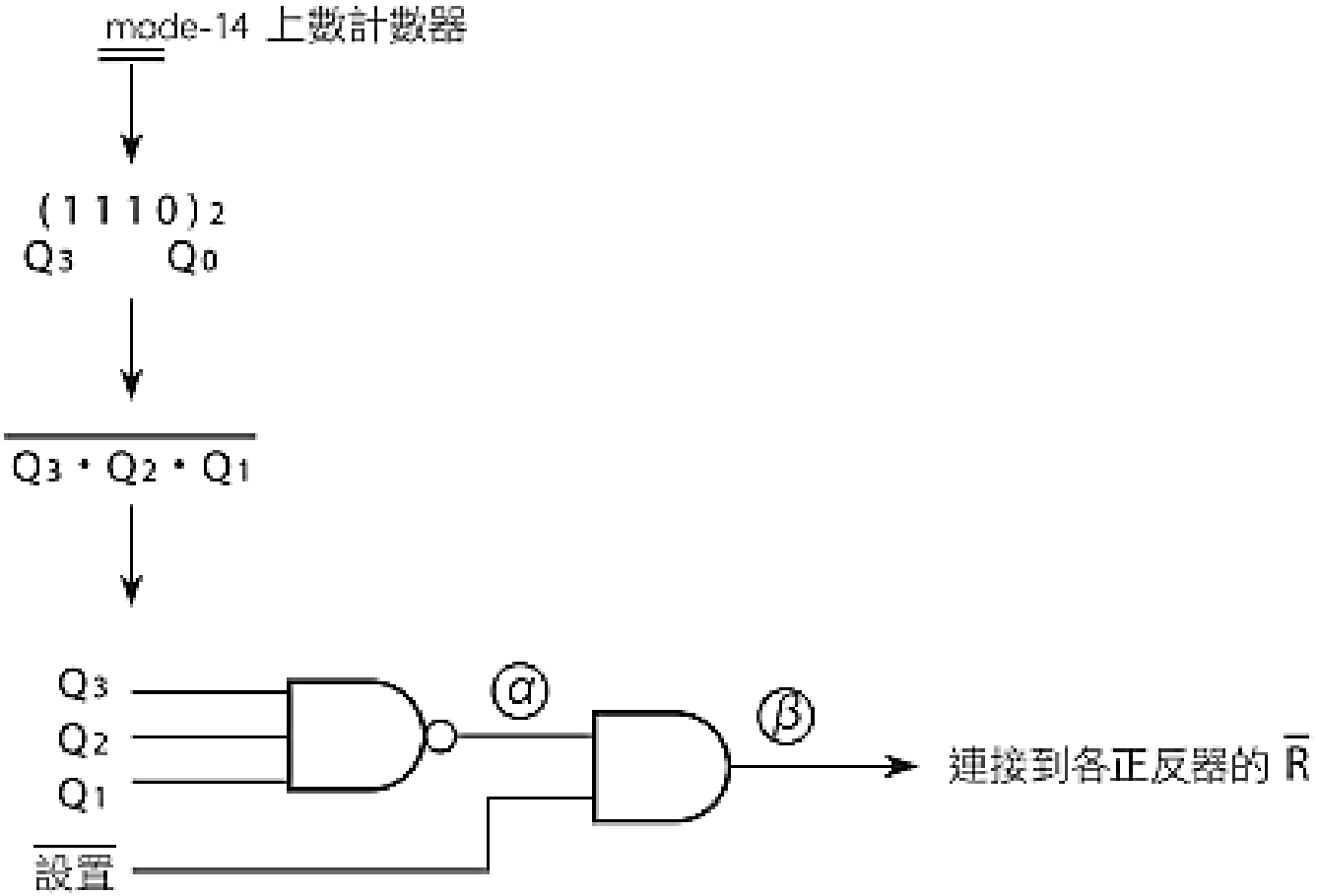
mode-N 計數器的構成

- 非同步 mode-10 上數計數器的構成：
  - mode-10 上數計數器的「10」用 2 進制表示為 →[1010] (2)
  - 用 2 進制裡面 '1' 的位元透過 NAND 的結果來輸出α
  - α 與「 $\overline{\text{重置}}$」的 AND 結果來輸出 β



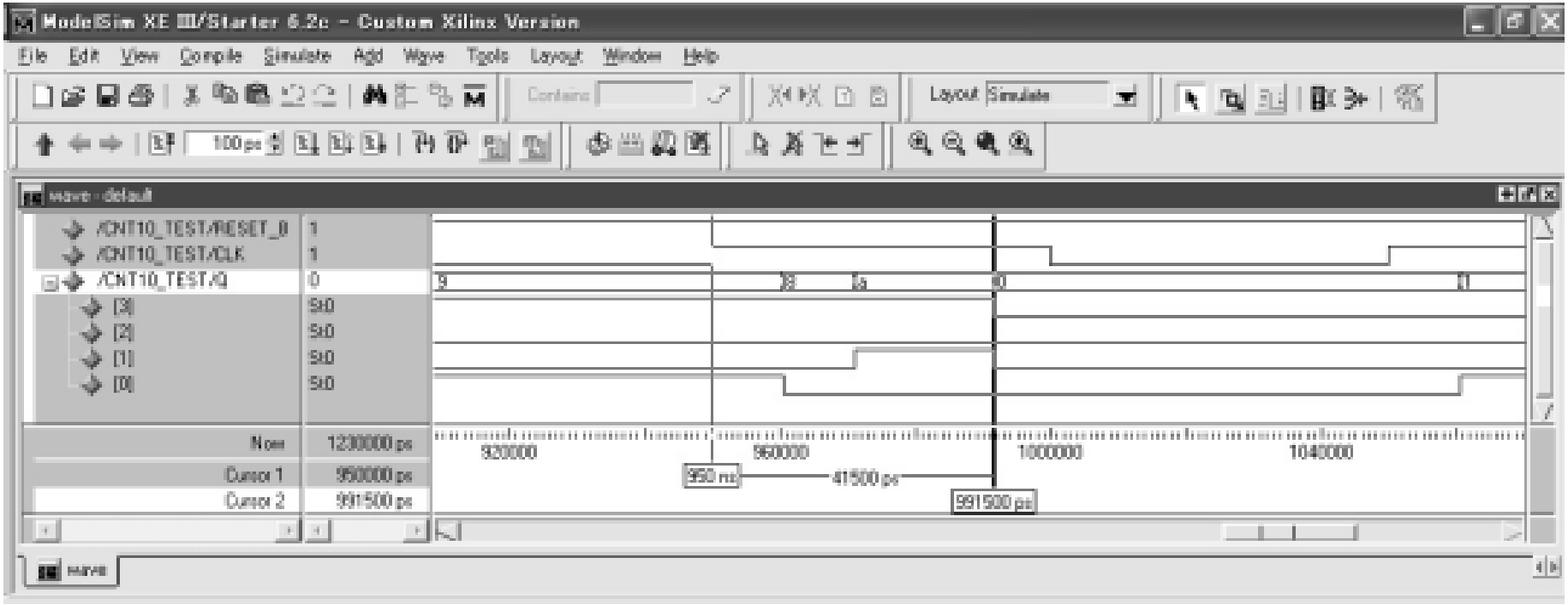
▲ 圖 7.21 mode-10 上數計數器的構成

- 同理, mode-14 上數計數器的 α, β 的構成如下



▲ 圖 7.22 mode-14 上數計數器的構成

非同步 mode-10 上數計數器的 Verilog HDL 描述



▲ 圖 7.24　非同步上數計數器的模擬結果

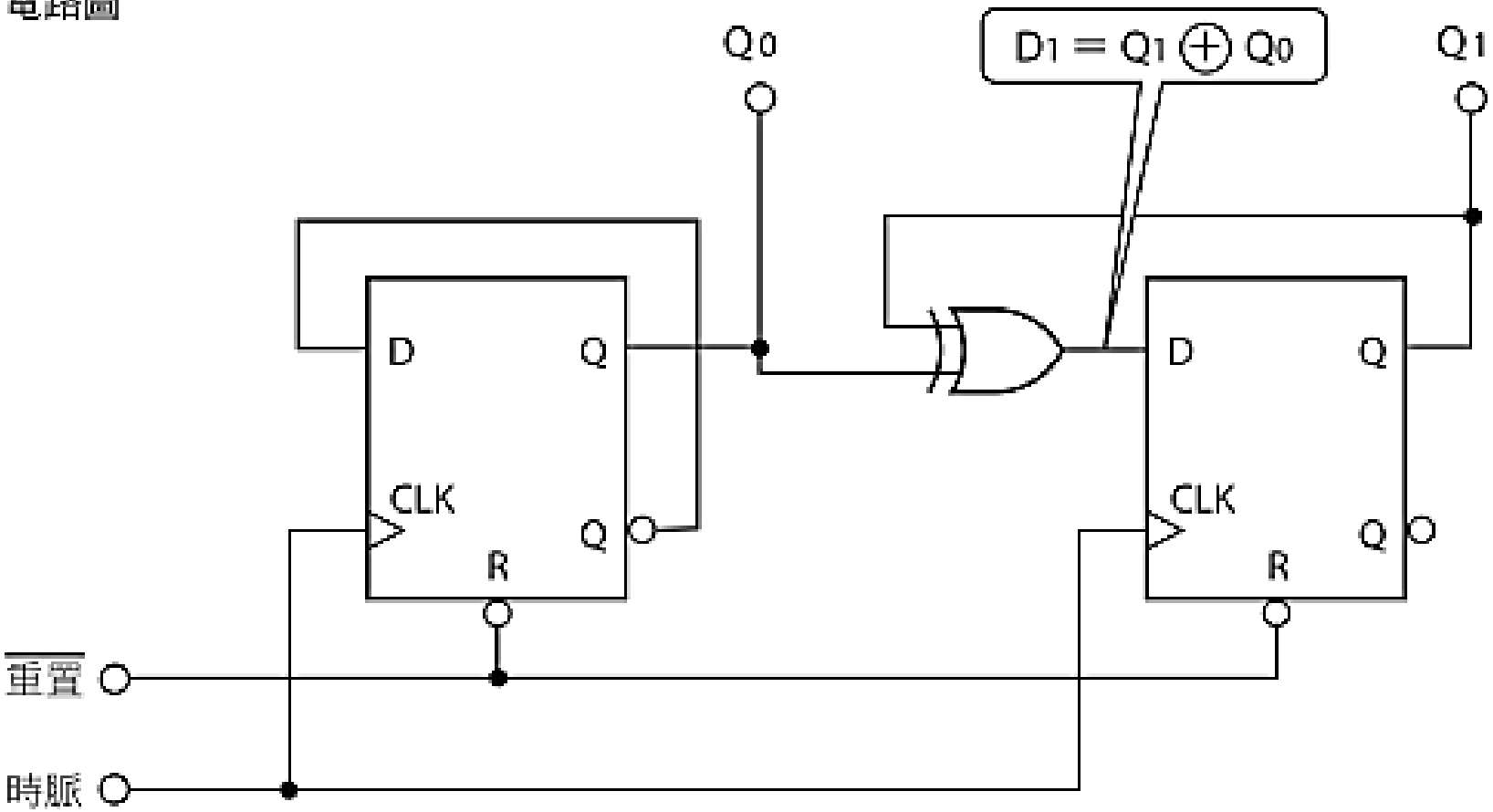
7.4 同步計數器

- 同步電路的特性如下,只要其中一個不符合就是非同步式：
  - 所有正反器的 CLK 都跟時脈接在一起
  - 所有正反器的 S 都不使用
  - 所有正反器的 R, 不是不使用就是接到重置

7.4.1 同步 mode-2^n 計數器

- mode-4上數計數器

(a) 電路圖



同步計數器的特徵

- 正反器的傳遞延遲時間不會累加
- 計數值不會有多餘的數值跑出來

同步計數器的 Verilog HDL 描述

- 根據電路圖的邏輯式把各正反器的 D 接在一起
- 同步電路正反器的 CLK 跟 R 都是各自全部接在一起, 所以也可以不呼叫模組「R_SYDFF」
- 程式 7.17 用「電路圖印象的描述」，程式 7.18 則是用各個正反器 D 的「邏輯式描述」, 後者設計上的重點為：
  - 模組「CNT4」的暫存器宣告
  - 用模組「CNT4」在 always 語法中把狀態帶入到 Q

7.4.2 同步 mode-N 計數器

- 跟非同步的電路相比，同步的較為複雜
- 同步電路的時序圖可以想成：
  - 考慮 mode-10 上數計數器的動作，把 Q_3~Q_0 標記在時序圖上
  - 從電路圖推出 Q_3~Q_0 的各輸入 D 的邏輯式

$$D_3 = Q_2 \cdot Q_1 \cdot Q_0 + Q_3 \cdot \overline{Q_0}$$
$$D_2 = Q_2 \cdot \overline{Q_1} + Q_2 \cdot \overline{Q_0} + \overline{Q_2} \cdot Q_1 \cdot Q_0$$
$$D_1 = Q_1 \cdot \overline{Q_0} + \overline{Q_3} \cdot \overline{Q_1} \cdot Q_0$$
$$D_0 = \overline{Q_0}$$

  - 根據 Q_3~Q_0 與邏輯式標記 D_3~D_0