正確學會 改訂新版 ディジタル回路と Verilog HDL

Verilog 的16堂課

第 13 章

乘法器電路設計

本章重點

乘除法電路是適當的使用運算電路與移位暫存器或計數器來實現，本章不只介紹乘法電路, 也會說明「為什麼要這樣構成電路」。

- 13.1 乘法電路
- 13.2 乘法電路的高速化12.3輸入信號的同步化

13.1 乘法電路

13.1.1 電路構成

- 乘法是算出每一個位數的積，之後一邊做進位一邊作加法，來求出最後的積，其運算可推展如下：
  - 部分積的累加，一直到最後的積求出來之前需要暫存這些結果，所以需要暫存器，叫作「ACC」。
  - 需要暫存被乘數的暫存器，「MD」
  - 需要移位暫存器，把乘數從 LSB 開始一個一個位元依序輸出，稱作「MQ」。

各暫存器的機能

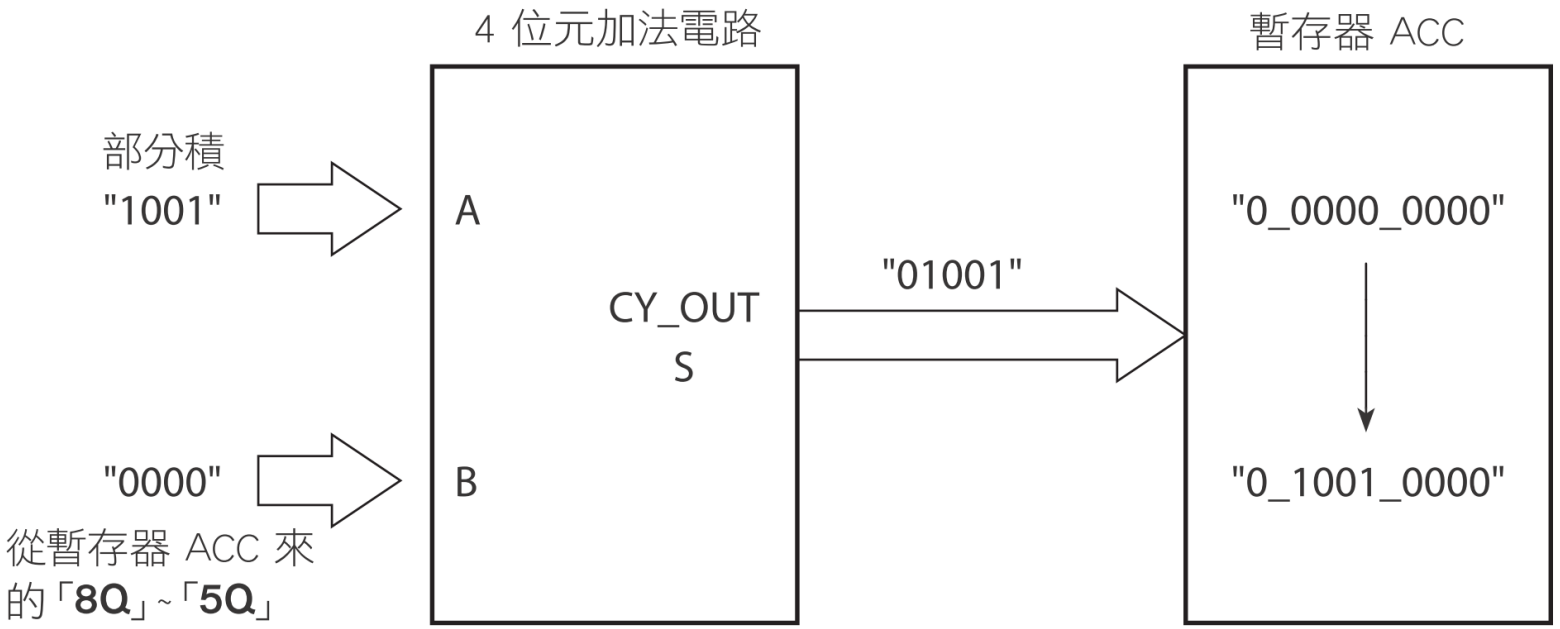
- 暫存器「MD」
  - 用來暫存被乘數 4 位元的暫存器
- 移位暫存器「MQ」
  - 用來暫存乘數 4 位元的暫存器。
  - 乘數的位元會根據「MQ[0]」→「MQ[1]」→「MQ[2]」→「MQ[3]」順序輸出給「SOUT」
- 為了求出部分積需要 4 個 2AND 電路
- 4 位元加法電路
  - 把部分積與 ACC 的值做相加。

- 暫存器「ACC」
  - 主要是用來累加部分積，以及最後會暫存最後積為主要目的
  - 也附有移位暫存器的機能
- 計數器「CNT8」
  - 求出逐位數的部分積，一邊位移一邊相加
- 正反器「JK_FF」
  - 求出逐位數的部分積，一邊位移一邊相加

暫存器「ACC」

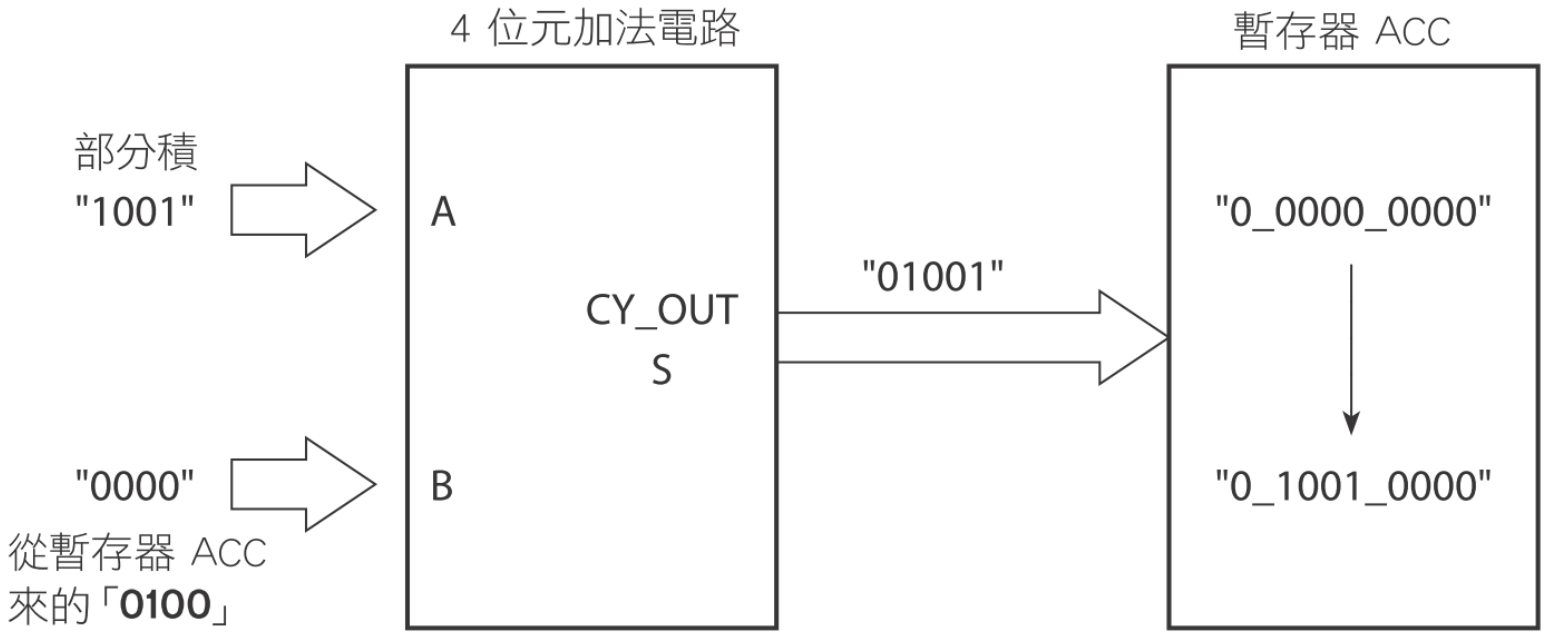
- 暫存器的輸入與輸出的位元關係都會相差一個位元
  - 「8Q」 → 「B3」 → 「S3」 → 「7D」
  - 「7Q」 → 「B2」 → 「S2」 → 「6D」
  - 「6Q」 → 「B1」 → 「S1」 → 「5D」
  - 「5Q」 → 「B0」 → 「S0」 → 「4D」
  - 「4Q」 → 「3D」
  - 「3Q」 → 「2D」
  - 「2Q」 → 「1D」
  - 「1Q」 → 「0D」

- A. 乘數「MQ」LSB 的 '1' 的部分積與暫存器 ACC 做相加



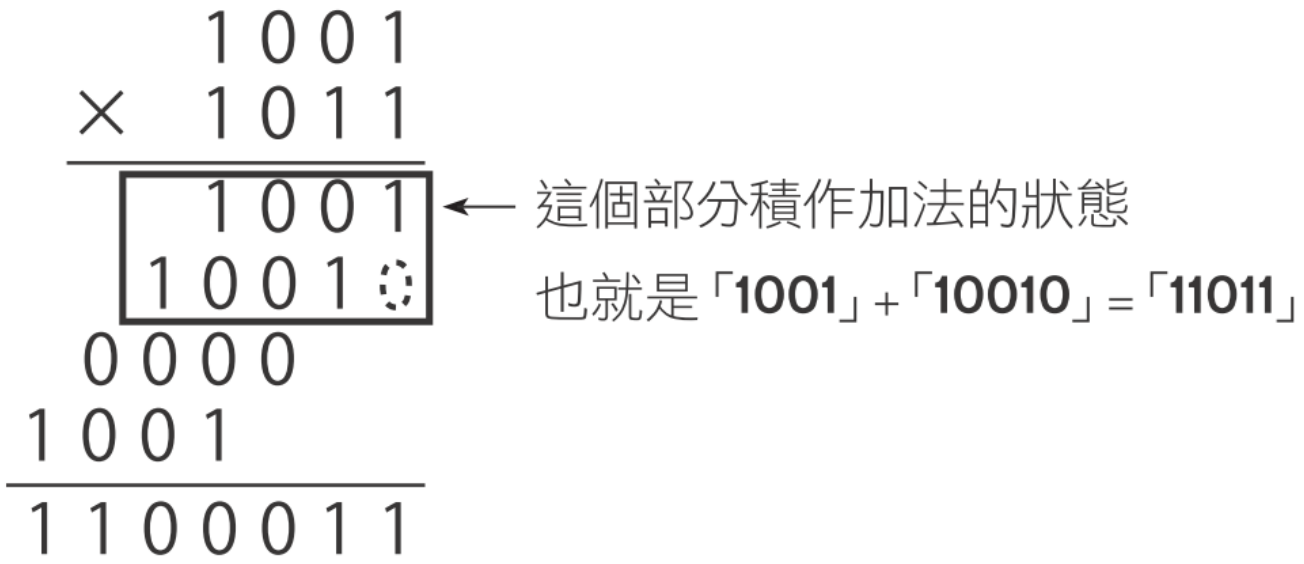
▲圖 13.2　乘數 MQ 的 LSB 部分積與暫存器 ACC 的相加

- B. 乘數 MQ 的第 1 位元的 '1' 的部分積與暫存器 ACC 相加



▲圖 13.3　乘數 MQ 的第 1 位元的部分積與暫存器 ACC 做相加

- 用手算的方式來確認結果：

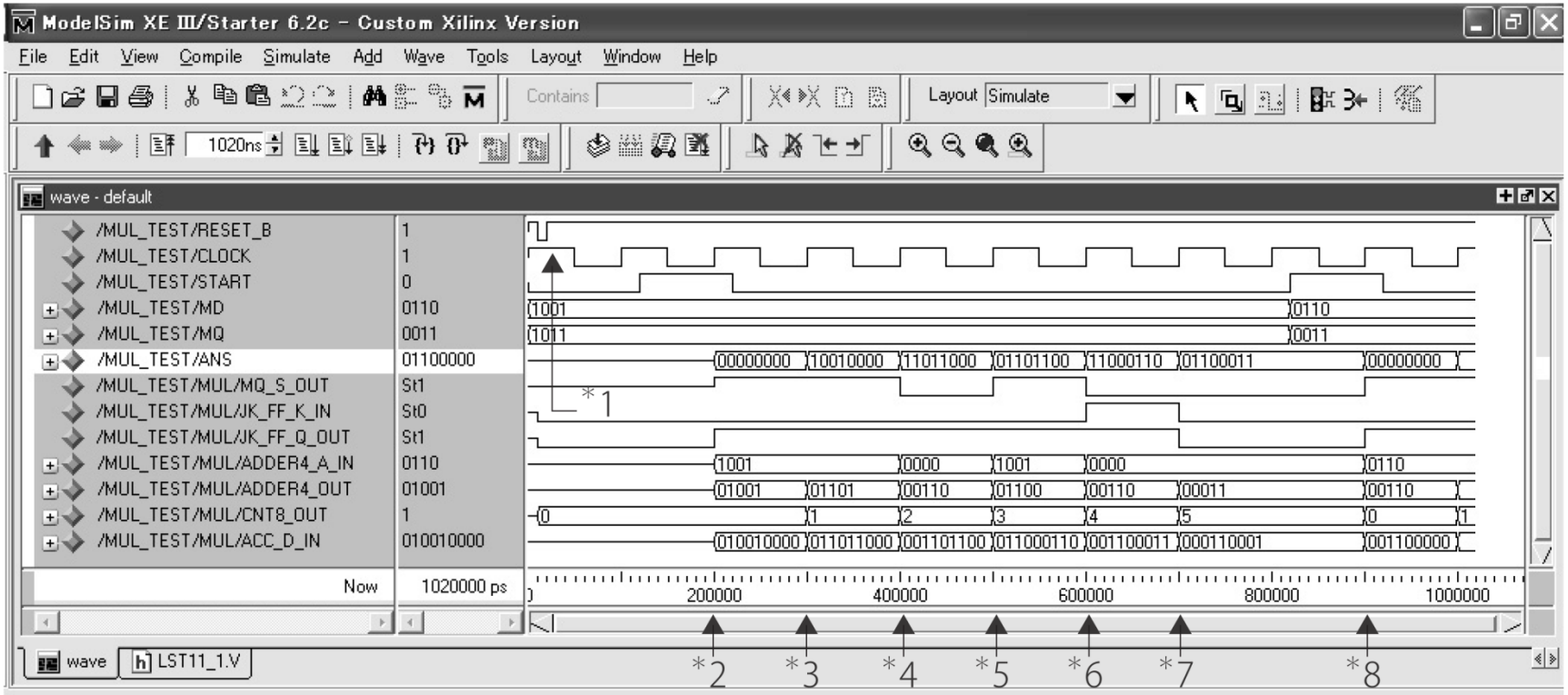


13.1.2 乘法電路的 Verilog HDL 描述

- 請直接取用光碟相關程式
- 電路的構成元件與模組的關係如下：

測試平台	→ MUL_TEST	→ 程式 13.1
乘法電路	→ MUL	→ 程式 13.2
暫存器「MD」	→ EN_REG4	→ 程式 13.3
暫存器「ACC」	→ EN_REG9	→ 程式 13.4
移位暫存器「MQ」	→ EN_PIN_SOUT_SHIFT	→ 程式 13.5
正反器「JK_FF」	→ R_SYJKFF	→ 程式 13.6
計數器「CNT8」	→ EN_CNT8	→ 程式 13.7

13.1.3 模擬結果的驗證



▲圖 13.5 4 位元 × 4 位元乘法電路的模擬結果

說明

- 1.系統重置的「 RESET_B」會清除正反器「 K_FF」與計數器「CNT8」
- 2.當表示乘法運算開始的「START」為 ON 的時候
  - 把 MD 的值設定進暫存器 MD 裡面
  - 把 MQ 的值從移位暫存器 MQ 裡面讀取出來
  - 清除暫存器 ACC
  - 清除計數器「 CNT8」
  - 設置正反器「 JK_FF」
- 3～6：將先前說明 A、B 的動作做 4 次, 且暫存最後的結果

- 7.暫存器 ACC 的輸入「 z 7D」~「0D」的積, 要由「7Q」~「0Q」做輸出
- 8.再一次把「START」變為 ON, 重複 2. 的動作

13.1.4 用乘法運算子 '*' 做的乘法電路的 Verilog HDL 描述

- 乘法電路可以使用乘法運算子 '*' 來做描述,只需要像「ANS = MD * MQ;」描述即可

左邊的位元長度 = 被乘數的位元長度+乘數的位元長度

程式 13.9 4 位元 × 4 位元乘法電路的 Verilog HDL 描述

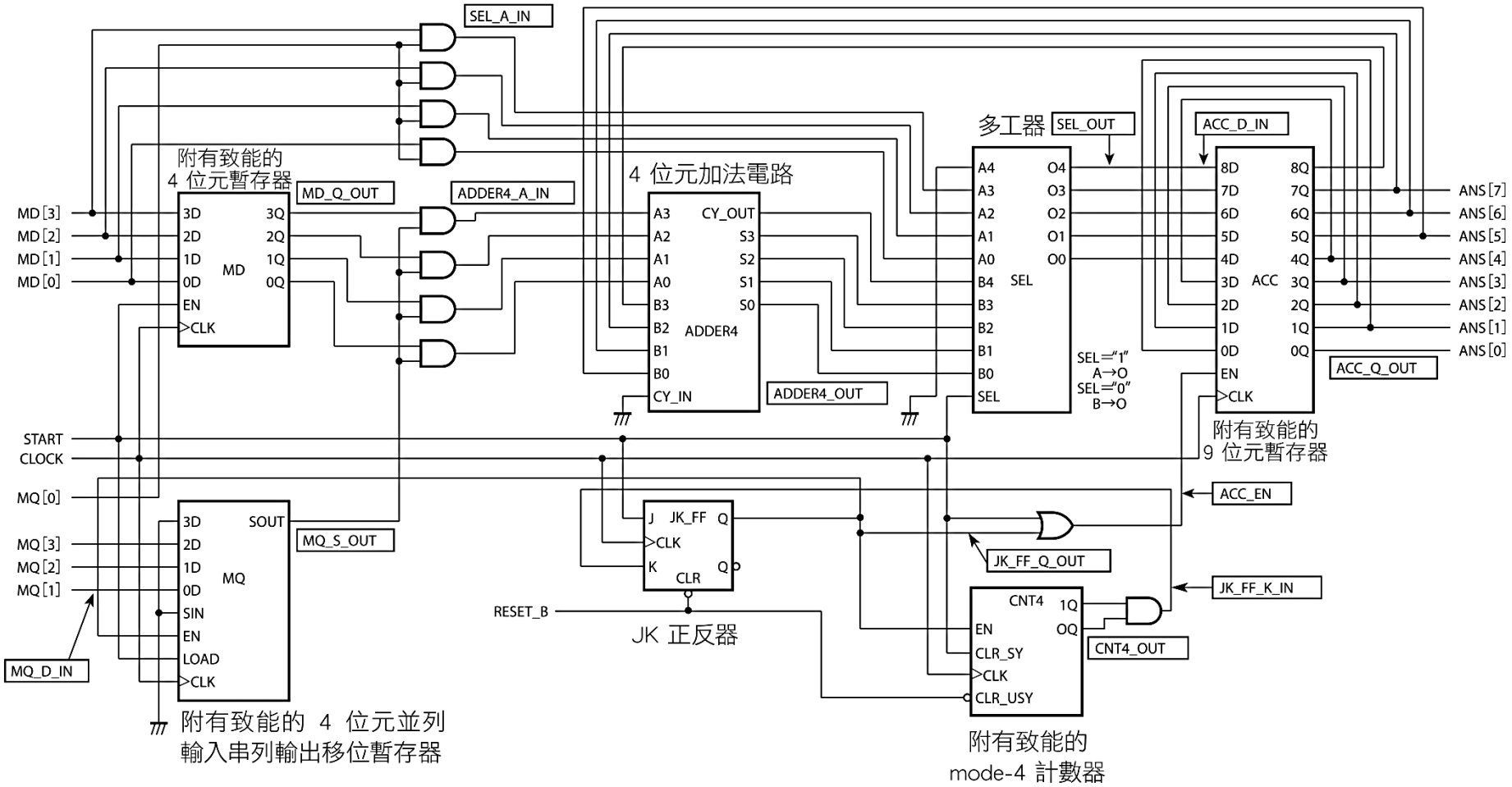
```
/*          MUL          */
module      MUL          ( MD, MQ, ANS );
input       [3:0]        MD, MQ;
output      [7:0]        ANS;

            assign       ANS = MD * MQ;
endmodule
```

13.2 乘法電路的高速化

- 前一節的乘法電路的運算時間為 5 個時脈，這邊讓我們來檢討要怎麼樣縮短這個運算時間

13.2.1 累加器讀取的技巧

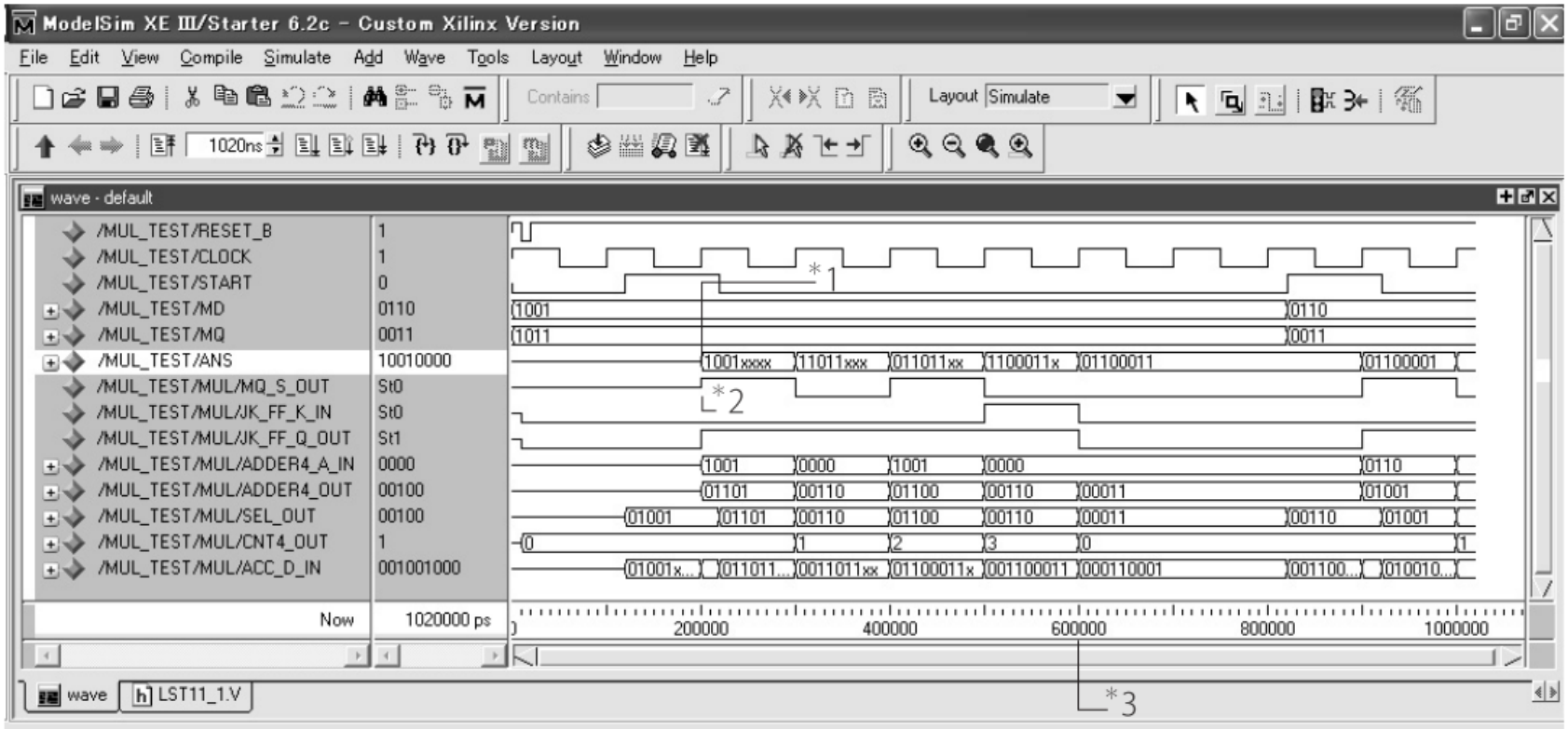
- 與其做重置為 '0' 的這個動作，何不拿這個時間來讀取一開始的部分積
- 暫存器 ACC 在「START」為 ON 的時候不是做清除動作，而是變成做讀取的動作



▲圖 13.6　4 位元 × 4 位元乘法電路(累加器讀取的技巧)

暫存器「ACC」的變更

- 暫存器「ACC」的變更
  - 不需要清除動作，所以把輸入「CLR」給移掉
  - 把輸入「EN」連接上正反器「JK_FF」的 Q 與「START」做 OR 之後的信號
- 追加多工器
- 追加與「MQ[0]」相關的部分積
- 移位暫存器「MQ」的變更
  - 變更為在運算中透過移位暫存器 MQ 的「SOUT」
- 計數器的變更
  - 變成 mode-4 上數計數器
  - 計數器為 3，「JK_FF」的輸入 K 為 OFF



▲圖 13.7　4 位元 × 4 位元乘法電路的模擬結果

說明

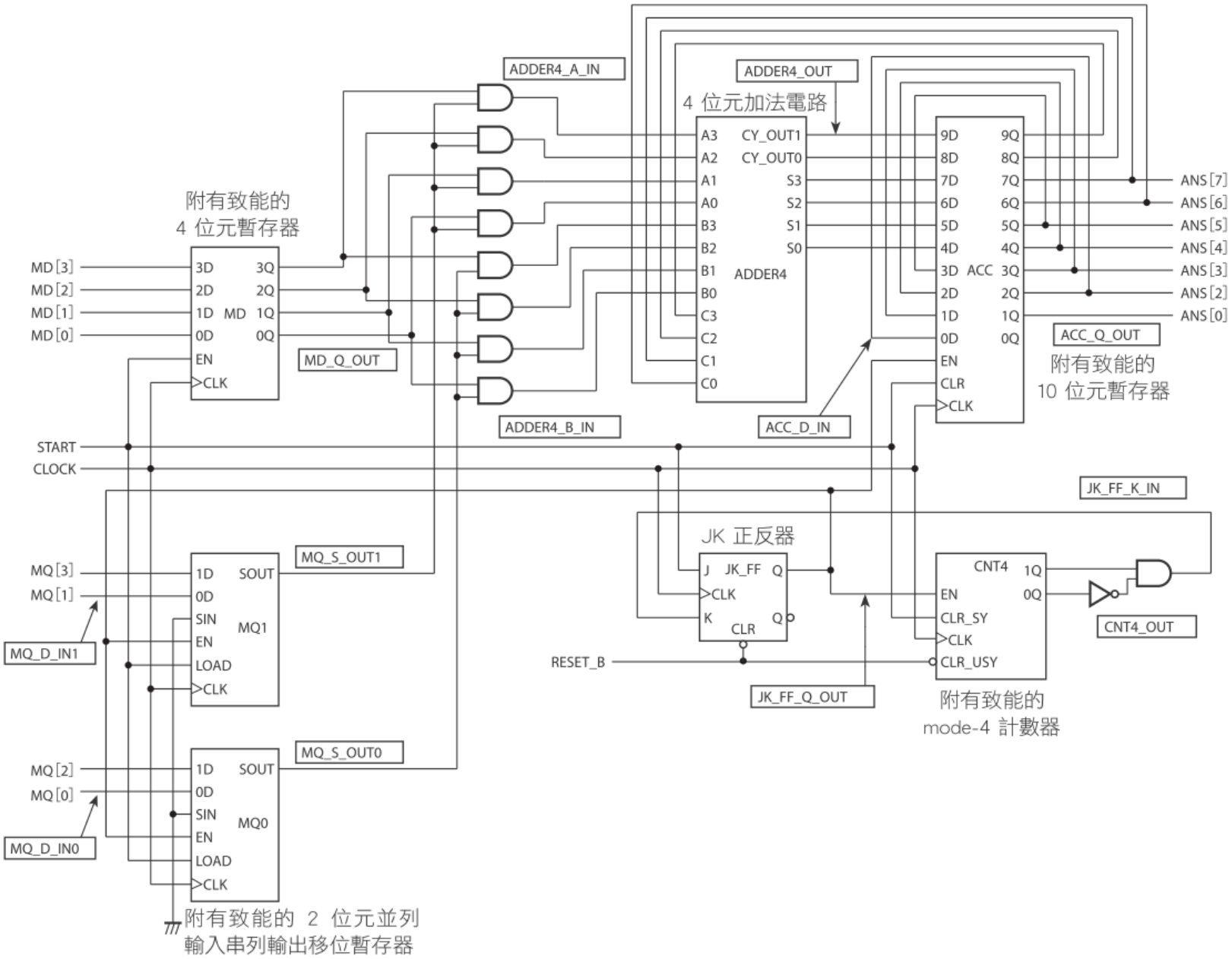
- 1.當「 START」為 ON 的時候, 遇到「CLOCK」的正緣時就會把跟「MQ[0]」相關的部分積由暫存器 ACC 讀取出來
- 2.移位暫存器 MQ 的「 z SOUT」是由「MQ[1]」來輸出, 再來依序為「MQ[2]」→「MQ[3]」來輸出
- 3.運算時間只花了 4 個時脈

補充

- 「START」的時候沒有做清除動作的狀況：下一個運算動作不會正確的實行
- 「 START」的時候有做清除動作的狀況：因為有根據「START」讓計數器的值為 '0', 所以運算動作會正常實行的機率比較高
- 不排除雜訊 (noise) 之類的干擾會讓其他電路受到影響

13.2.2 同時處理 2 位元的乘數

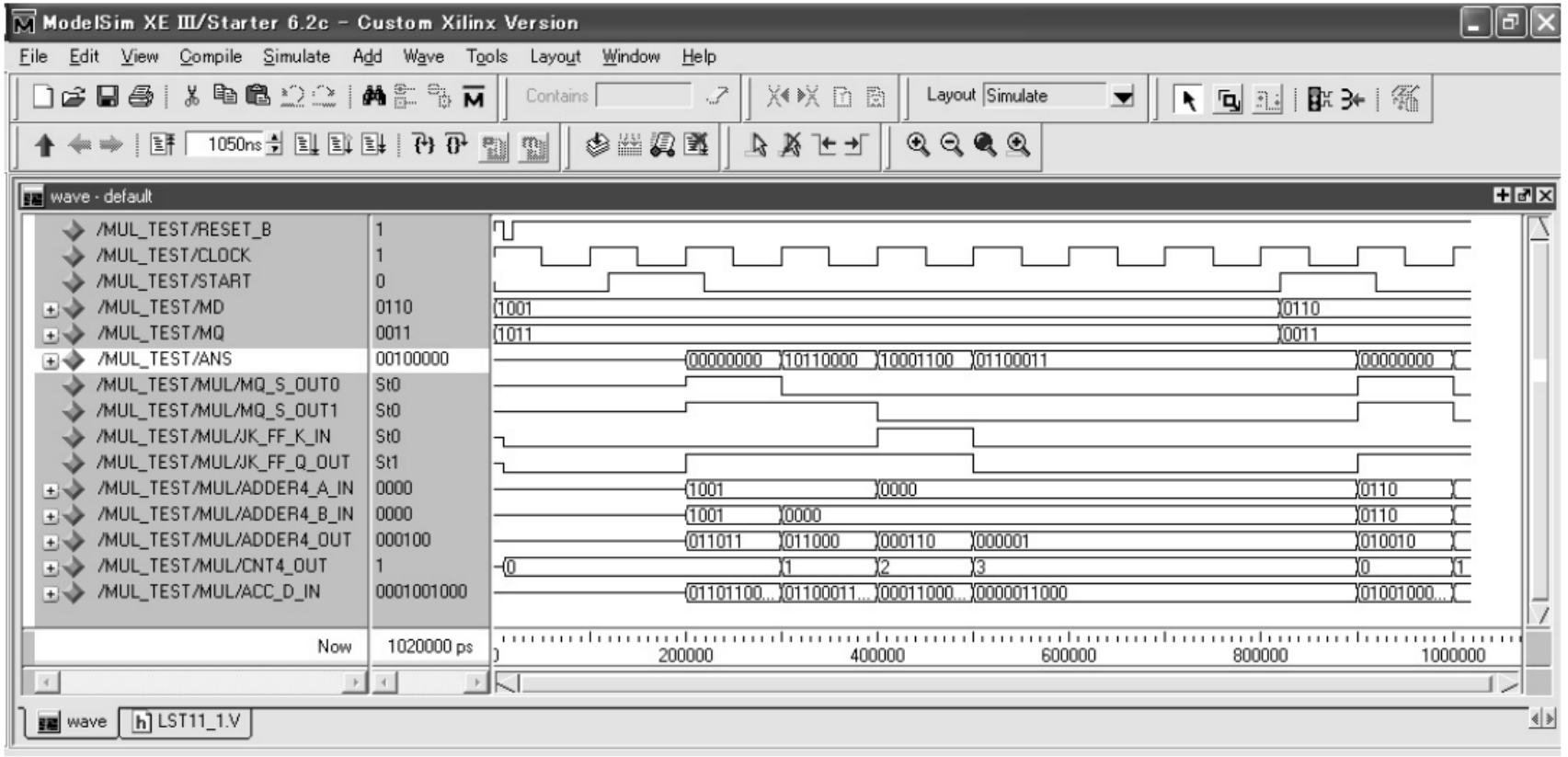
- 一次加兩個, 這樣移位的次數減半, 當然運算時間也減半



▲圖 13.8 4 位元 × 4 位元乘法電路(同時處理乘數的兩個位元)

- 暫存器「ACC」的變更
  - 「9Q」 → 「C3」 → 「S3」 → 「7D」
  - 「8Q」 → 「C2」 → 「S2」 → 「6D」
  - 「7Q」 → 「C1」 → 「S1」 → 「5D」
  - 「6Q」 → 「C0」 → 「S0」 → 「4D」
  - 「5Q」 → 「3D」
  - 「4Q」 → 「2D」
  - 「3Q」 → 「1D」
  - 「2Q」 → 「0D」
- 移位暫存器「MQ」的變更
  - 「MQ[0]」與「MQ[2]」,「MQ[1]」與「MQ[3]」用到個別的移位暫存器

- 為了求出部分積的 8 個 2 AND 電路
- 4 位元加法電路的變更
  - 輸入為 A, B, C 各 4 位元
  - 輸入 A 會變成「A x 2+B+C」
- 計數器的變更
  - 控制「EN」的計數器只需要計數到 '2‘
  - 正反器「CNT4」的值為 '2‘,「JK_FF」的輸入 K 為 OFF



▲圖 13.9　4 位元 × 4 位元乘法電路的模擬結果