實用數位系統設計

HW1　8 bits adder

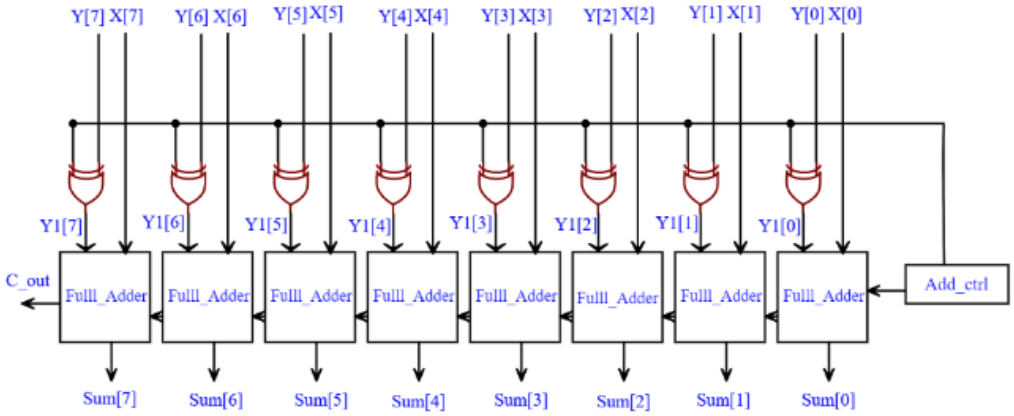
學號:B103012002

姓名:林凡皓

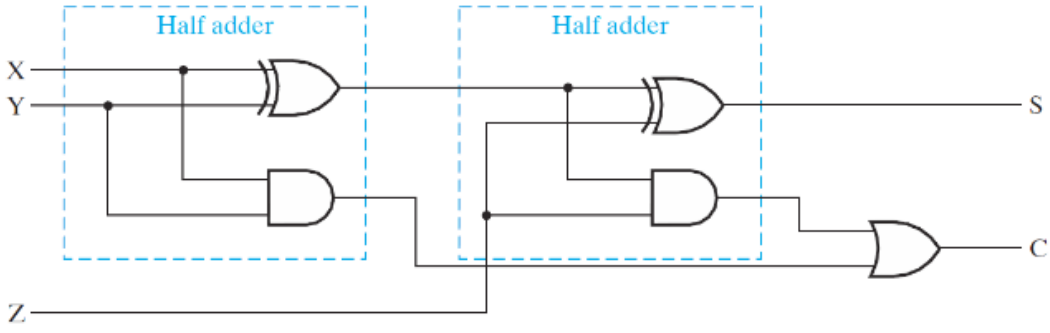
一、 設計原理

(一) Ripple Carry Adder:

一個 8 bits ripple carry adder 可以先拆成八個 full adder(如圖(一)) ，而一個 full adder 可以拆成兩個 half adder(如圖(二))。



圖(一)



圖(二)

先由 half adder 開始設計，利用 gate level 的設計方式將邏輯閘組合在一起即可完成設計。完成 half adder 後，將兩個 half adder 依照圖(二)的

電路組合在一起即可完成一個 full adder。最後再將八個 full adder 利用圖(一)之電路組合在一起，就可以完成一個八位元的加法器。從圖(一)中我們可以發現，Y 的每個位元要輸入進 full adder 時，都會和 Add_ctrl(即 code 當中的 op_mode)一起先經過一個 XOR，才會輸入進 full adder，這是為了要控制執行加法還是減法所設計的。當 Add_ctrl 為 1 時，要執行減法，所以每一個 Y 的位元都會反向(1 變 0，0 變 1)，而為了滿足二補數的結果(一補數加一)，Add_ctrl 要接到圖(二)中的 Z 腳位，來達到加一的效果；當 Add_ctrl 為 0 時，要執行加法，此時 Y 的每一個位元會維持原來的訊號，而結果就會是我們所預期的加法。另外，為了要偵測是否 overflow，所以有一個額外的訊號 Overflow=c6^C_out 被加到 code 當中，1 代表有溢位，0 代表沒有溢位。

(二) Carry Lookahead Adder:

Carry lookahead adder 和 ripple carry adder 最大

的不同在於 carry lookahead adder 不用等待上一個位元的進位就能繼續算下一個位元，所以 carry lookahead adder 的速度會比 ripple carry adder 快，但是 carry lookahead adder 的面積就會比 ripple carry adder 大。

由於 carry lookahead adder 的邏輯閘較多的關係，如果要一次設計出一個八位元的加減法器想必會很複雜，所以我的想法是先設計出一個四位元的加減法器，再將兩個四位元的加減法器組合起來變成一個八位元的加減法器。四位元的加減法器如下:

$p_i = X_i \wedge Y_i$, $g_i = X_i \& Y_i$, for i = 0 , 1 , 2 , 3

c0 = g0+p0cin

c1 = g1+p1g0+p1p0cin

c2 = g2+p2g1+p2p1g0+p2p1p0cin

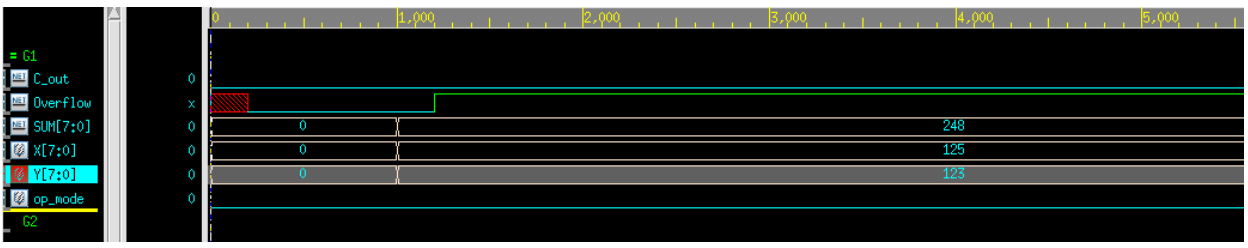
c3 = g3+p3g2+p3p2g1+p3p2p1g0+p3p2p1p0cin

二、 結果分析

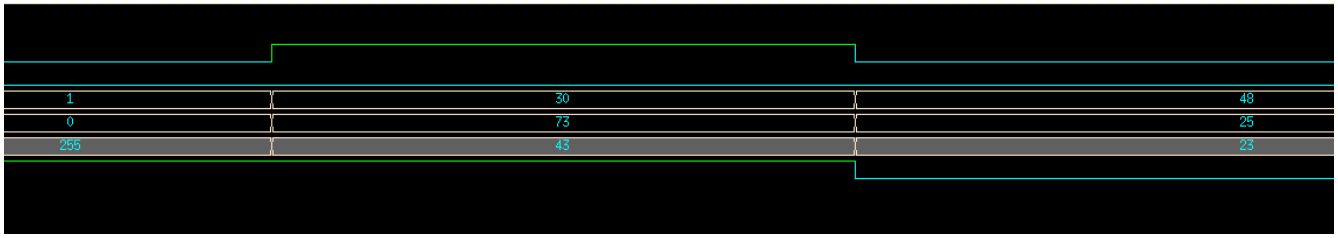
(一) Ripple Carry Adder:

設計 testbench 時，我的想法是先測試 Overflow 的

功能是否正常，我選擇使用 125+123，雖然這兩個數相加後的結果看似正常，但是在有號數的世界裡 248(11111000)其實是負的，所以 Overflow 為 1。從圖(三)中我們還可以注意到產生 Overflow 的訊號會有 delay，這是因為在產生結果後還需要在經過一個 XOR gate 才會產生 Overflow 的訊號，也是因為這個原因，在 0ns 時 Overflow 會是 unknown。測試完 Overflow 之後是測試減法，我將減法分成 X>Y 和 X<Y 兩種狀況，當 X<Y(0-255)，產生的結果為 1，這是因為-255 取二補數之後的值為 00000001；當 X>Y(73-43)，產生的結果為 30，也是正確的答案，此外，如果將 43 取二補數後為 11010101，73 的二進位為 01001001，將兩數相加後為 101011110，有進位，所以 C_out 為 1。最後測試的是加法(25+23)，結果為 48，由於沒有溢位，所以 Overflow 的訊號為 0，這樣的結果也是正確的。



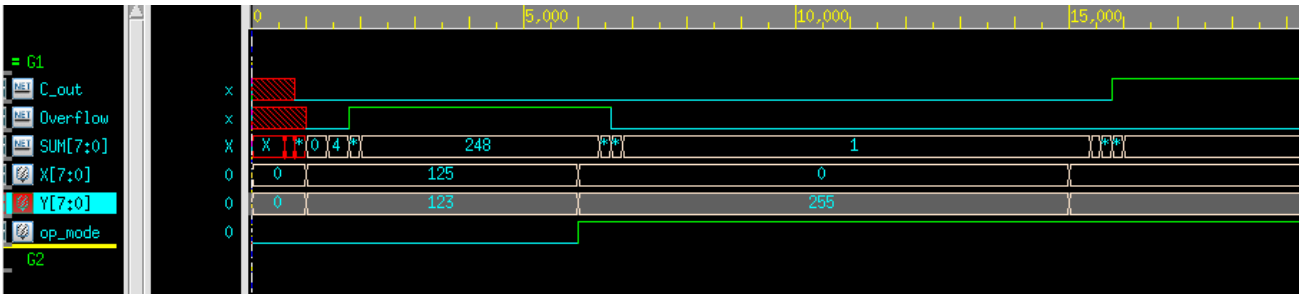
圖(三)



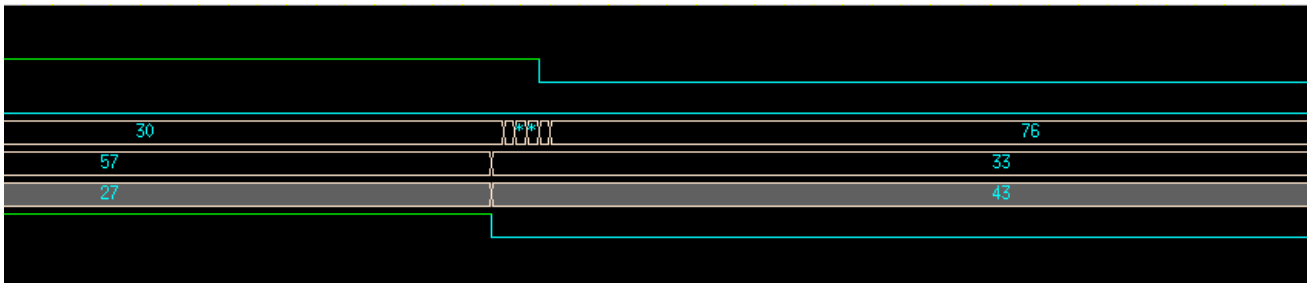
圖(四)

(二) Carry Lookahead Adder:

測試 Carry lookahead adder 的想法和測試 ripple carry adder 的想法一致，先從 Overflow 和 C_out 開始測試，我們也一樣能從圖(五)中注意到 Overflow 的訊號會有一段時間的延遲。接著測試減法，當 X<Y(0-255)時，因為取二補數的關係所以產生的結果為 1；當 X>Y(57-27)時，產生的結果為 30，為正確答案，此外因為 57 的二進為加上 27 的二補數有進位，所以 C_out 為 1。最後測試加法(33+43)，結果為 76，為正確結果。圖(五)和圖(六)中的 SUM 有一些*，這些是計算過程的一些結果，由於格子太小的關係，所以結果就被省略成*。



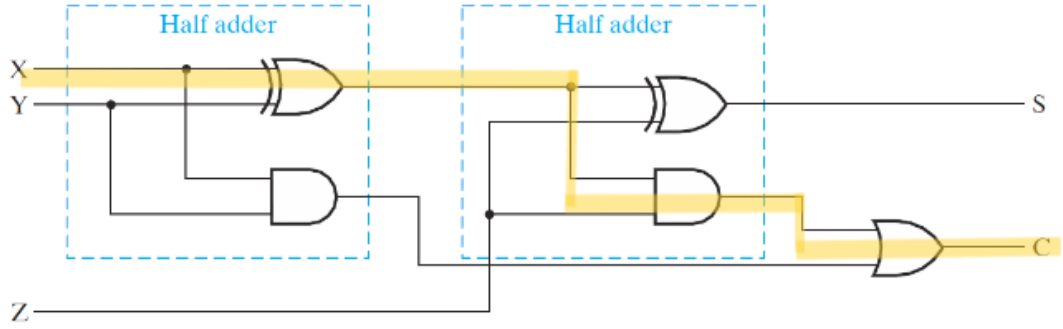
圖(五)



圖(六)

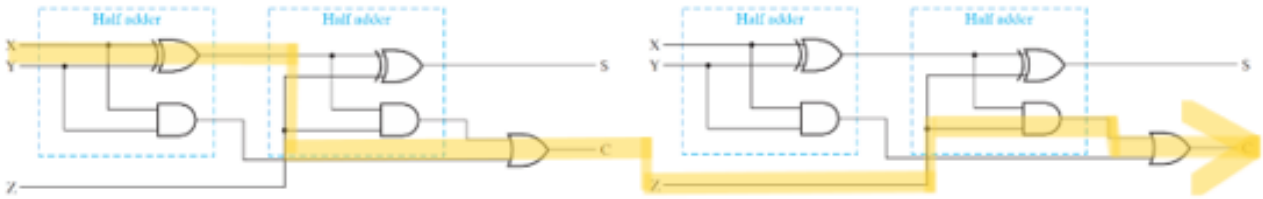
三、 問題與討論

(一) Critical path and maximum delay:



圖(七)

由圖(七)我們能算出當一個 full adder 獨立使用時的最長路徑需經過三個邏輯閘。

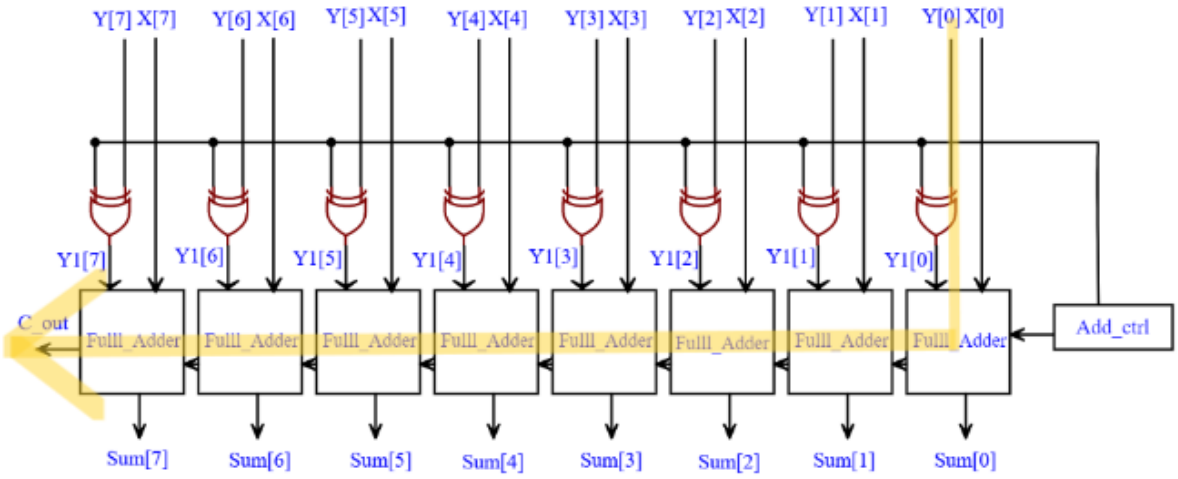


圖(八)

如果我們將兩個 full adder 串接在一起的話，如圖(八)，其最長路徑會變成需要經過 3+2=5 個邏輯閘，也就是當我們多串接一個 full adder，最長路經

需經過的邏輯閘數目就要加二。

一個八位元加減法器的最長路徑如圖(九)所示，除了經過八個 full adder 之外，在輸入的地方還要再經過一個 XOR gate，所以總供需要經過 1+3+(2*7)=18 個邏輯閘，由於我將每一個邏輯閘的延遲假設成 2ns，所以最長的延遲為 36ns。如果考慮到 Overflow 偵測功能，那最長路徑會多經過一個 XOR gate，也就是 delay 會變成 38ns。



圖(九)

(二) How many logic gates are used:

(1) Ripple carry adder:

8 bits Ripple carry adder 是由八個 full adder 和八個 XOR gate 所組成，而一個 full adder 含有五個邏輯閘，所以 8 bits ripple carry adder 有

8*5+8=48 個邏輯閘。如果考慮到 Overflow 偵測的功能，那邏輯閘數目就要多加上一個 XOR gate。

(2) Carry lookahead adder:

8 bits carry lookahead adder 是由兩個 4 bits carry lookahead adder 和八個 XOR gate 組成，一個 4 bits carry lookahead adder 有 27 個邏輯閘，所以 8 bits carry lookahead adder 有 27*2+8=62 個邏輯閘。

四、 心得討論

這次作業是我第一次使用工作站，過程中也發生了很多很多的問題，印象比較深刻的就是一開始設定時有一個步驟是要輸入 xclock 並確認是否有跳出時鐘，我嘗試了很久才發現是沒有連上 Xming server 造成的。在這次作業中除了讓我比較熟悉工作站的使用之外，我覺得更重要的是讓我更深刻的瞭解到 ripple carry adder 和 carry lookahead adder 之間的差別，雖然這兩種加法器在大一的數位系統設計的課程當中都學過了，但是這次作業是我第一次將這兩個電路實現

出來，並且進行一些比較，從這些分析與比較中也證實了當初所學習到的知識，例如:之前學過 carry lookahead adder 的面積會比 ripple carry adder 還要大，在這次作業中我計算出的邏輯閘總數分別是 62 個邏輯閘和 48(或 49)個邏輯閘，結果確實跟我所學的一樣。除此之外，在這次實驗中我也發現到自己對 testbench 非常不熟悉，像是自動化的 testbench 還有該如何設計測試 pattern 才能夠最有效的將電路完整的測試我在這次作業中都沒有做到，我感覺 testbench 會是我在未來的時間需要去多練習的項目，希望經過這學期的練習我能夠更加熟悉 testbench 的寫法。