實用數位系統設計－HW1-1

學號：B035020026　　姓名：李冠霖

設計原理：

這次加法器我使用前瞻進位加法器實作，使用兩個 4-bit 的前瞻進位加法器構成一個 8-bit 的加法器。前瞻進位需要求出 Generate, Propagate，公式為：

$$\text{Propagate } P_i = A_i \oplus B_i$$

$$\text{Generate } G_i = A_i B_i$$

進而求出 Sum 和 Carry：

$$\text{Sum } S_i = P_i \oplus C_i$$

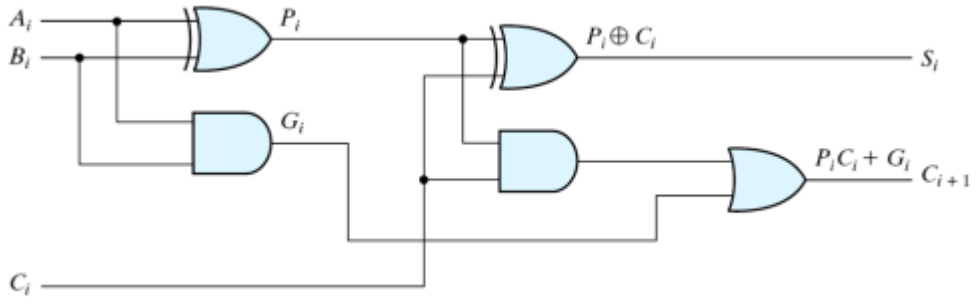
$$\text{Carry } C_{i+1} = G_i + P_i C_i$$

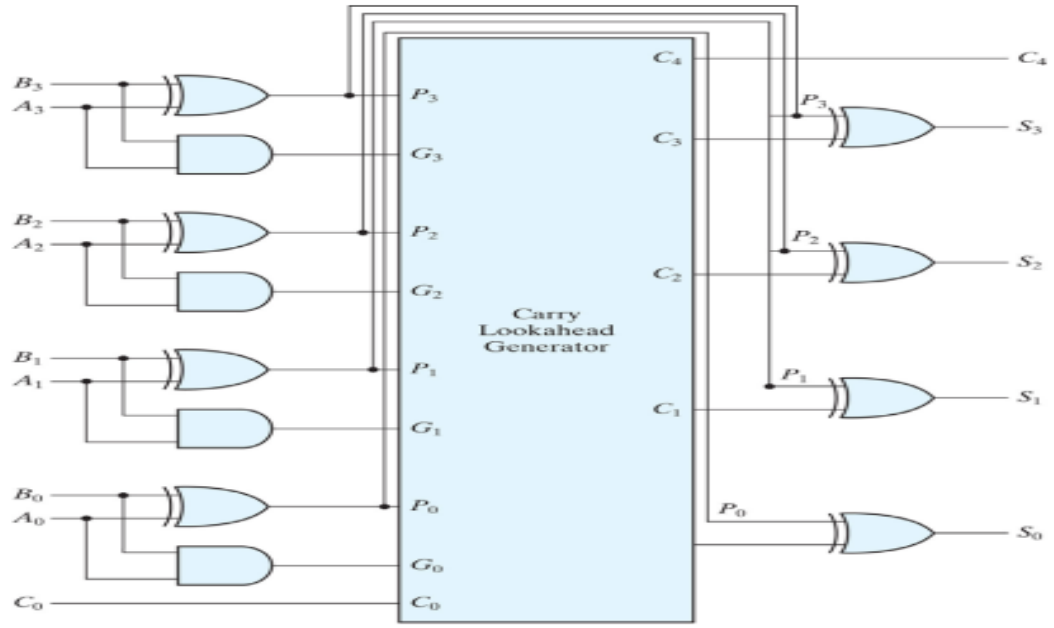
$$\rightarrow C_1 = G_0 + P_0 C_0$$

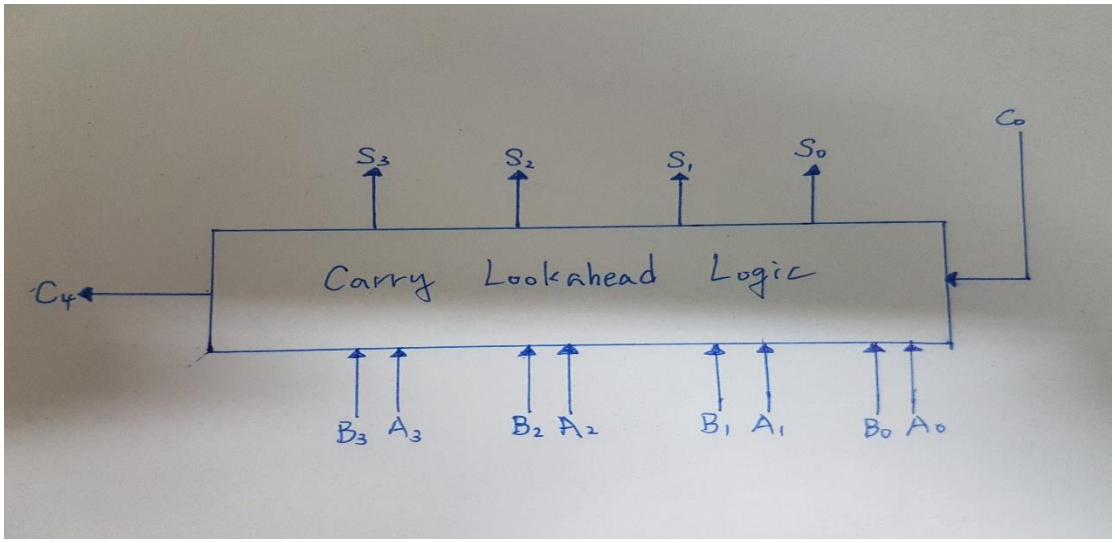
$$C_2 = G_1 + P_1 C_1 = G_1 + P_1(G_0 + P_0 C_0) = G_1 + P_1 G_0 + P_1 P_0 C_0$$

$$C_3 = G_2 + P_2 C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$$

$$\vdots$$







展開後同時計算才得已解決漣波進位加法器延遲的問題，同時因為展開算式，將會大幅增加邏輯閘的使用(使用面積增加)，尤其越高位的計算將會越複雜，也因此為了減少複雜度，又能獲得加速效果，採用了 4+4bits，將低位 4-bit 的 Carry 作為高位 4-bit 的輸入，這邊會產生延遲，但速度已經比漣波進位快上許多！

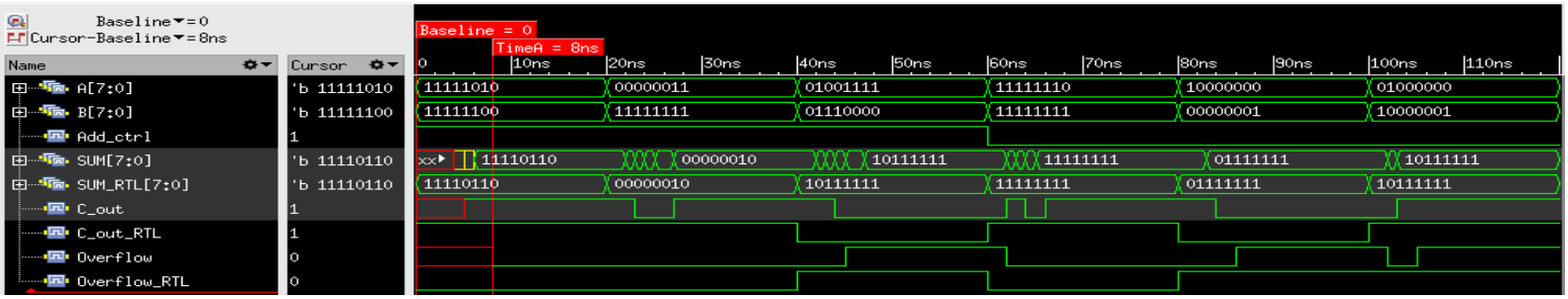
此加法器採用的是有號數運算，在進行加法或減法的部分提供一個 input (Add_ctrl)控制，若進行減法，則先將減數(B)與(~Add_ctrl)進行 xor 後進行相加即為 A-B 的結果。減法的 Carry Out 需再與(~Add_ctrl)進行 xor 後才是正確輸出。

最後還有 Overflow 的檢測，取同號相加結果為異號者(正+正=負)，以及異號相減結果者與被減數異號者(正-負=負)為 1，用卡諾圖化減則為

AB \ PS	00	01	10	11
00		1		
01				1
10			1	
11	1			

A 為被加數最高位，B 為加數最高位，P 為正負號(正=0)，S 為總和最高位。

結果分析：

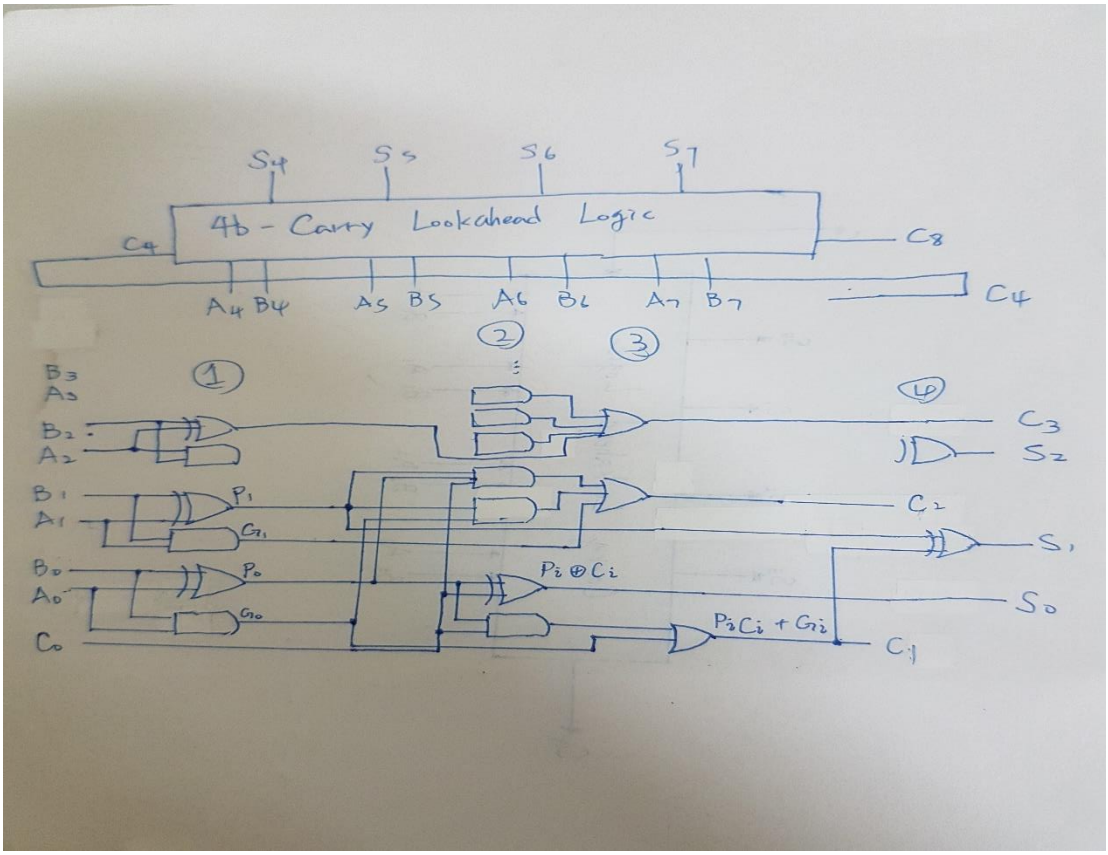


本加減法器可以顯示(-128)～(+127)（signed number）

使用六個輸入進行測試，分別為：

3+(-1)、(-6)+(-4)、79+112、(-2)-(-1)、(-128)-1、64-(-127)

測試同號異號的加減，並同時檢查是否可以正常處理 overflow 的狀況，從圖中可以看出在加入了 delay 的 gate level code 在 overflow 的波形有明顯的延遲後才顯示出正確的答案(RTL 版本未加上 delay)，也同時符合實際運作的情形，加入 delay 的方式為在每個基礎邏輯閘加上#1 的 delay，若不加上 delay 則在 input 進去後 0 秒就直接顯示，不符合實際電路運作狀況。



最長的路徑為 Overflow 的輸出，CLL 產生 3 個 delay，和產生 S 所需要的 xor 閘(1 delay)，以及前面判斷加減法的 not+xor(2 delay)，最後的兩層判斷 Overflow，3+1+2+2=8

心得討論：

對於第一次使用工作站，從架設環境到撰寫程式碼花了不少時間，加上非本科系，過去也沒有實際操作過實體電路，因此對硬體描述語言只停留在理論階段，在實際撰寫程式碼的時候，時常碰到語法錯誤導致到處都是 bug，電路的設計也是經過查詢多個參考資料學習而來的結果。