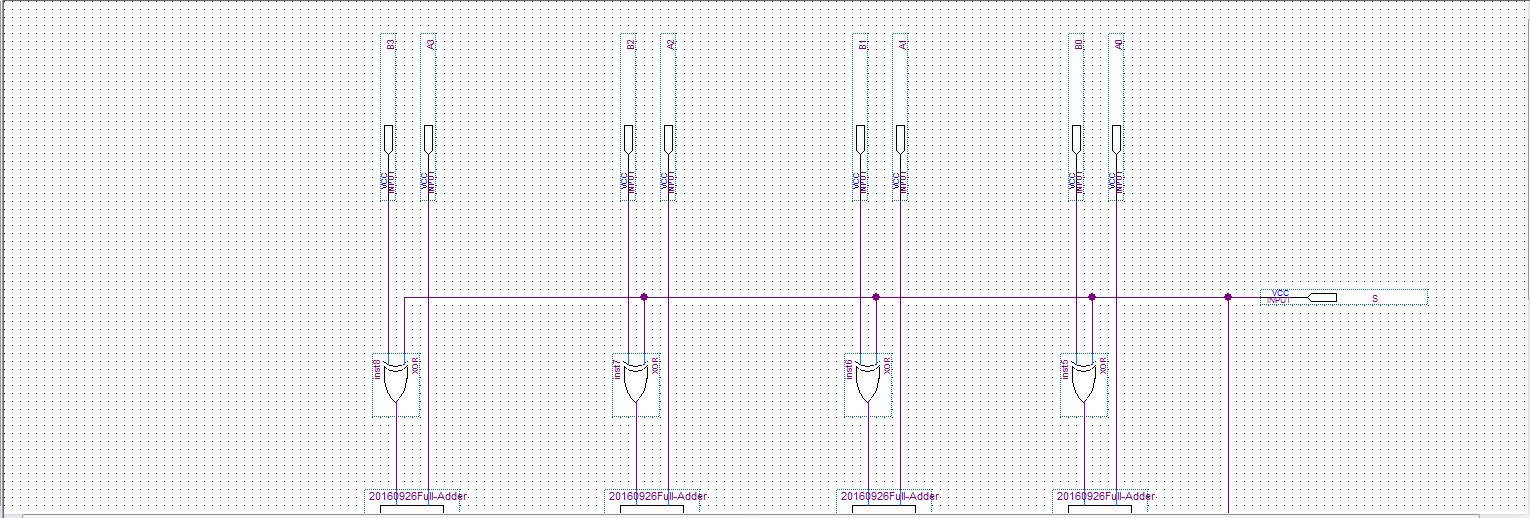
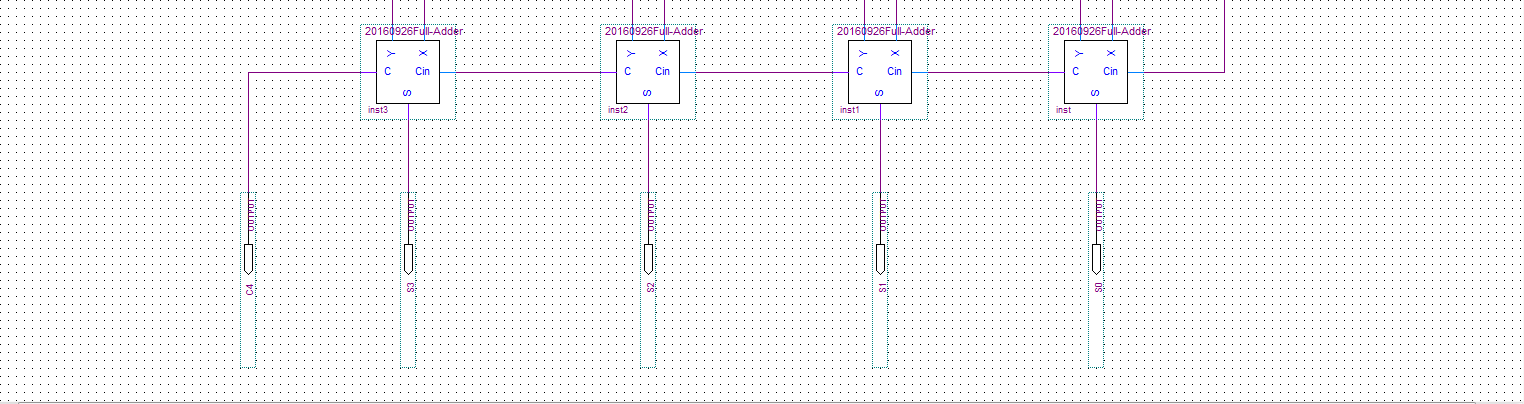
Title:

4 bits signed numbers Adder/Subtractor with an overflow detector



Team No.2

Members:

A1045516 蔡湘俊

A1045505 施彥廷

Abstract

此 4 bits signed numbers Adder/Subtractor with an overflow detector 可以說是4 bits Ripple Carry Adder 的強化版電路，因為在我們所熟悉的數學運算當中最常用的不外乎就是加、減、乘、除，這四個基本的運算建構出今天龐大的數學體系，而在我們電路的設計當中也希望透過電路來去實現這些功能，那這個 4 bits signed numbers Adder/Subtractor with an overflow detector 這個電路就實現了4大基本運算的其中兩個，分別是加法以及減法，那麼在這個電路當中我們設計了針對signed numbers的加減法運算，因為在整個數字體系當中是由正數以及負數來去組成的，所以將signed的方法設計進去會大大的增加此電路的實用度，那麼我們也多增加了overflow detector這一個偵測機制，它最主要的功用就是為了幫我們判斷是否此次的運算已經overflow了，打個比方，像是正數加正數竟然跑出了負數的結果，這很明顯的看出此運算結果一定是錯誤的，這時我們就會將其稱作overflow，多加了此項機制就能去確保運算出來的結果是正確的，由此可以看出此電路針對一般的運算使用是不會有什麼大問題的。

Content

1. methods:

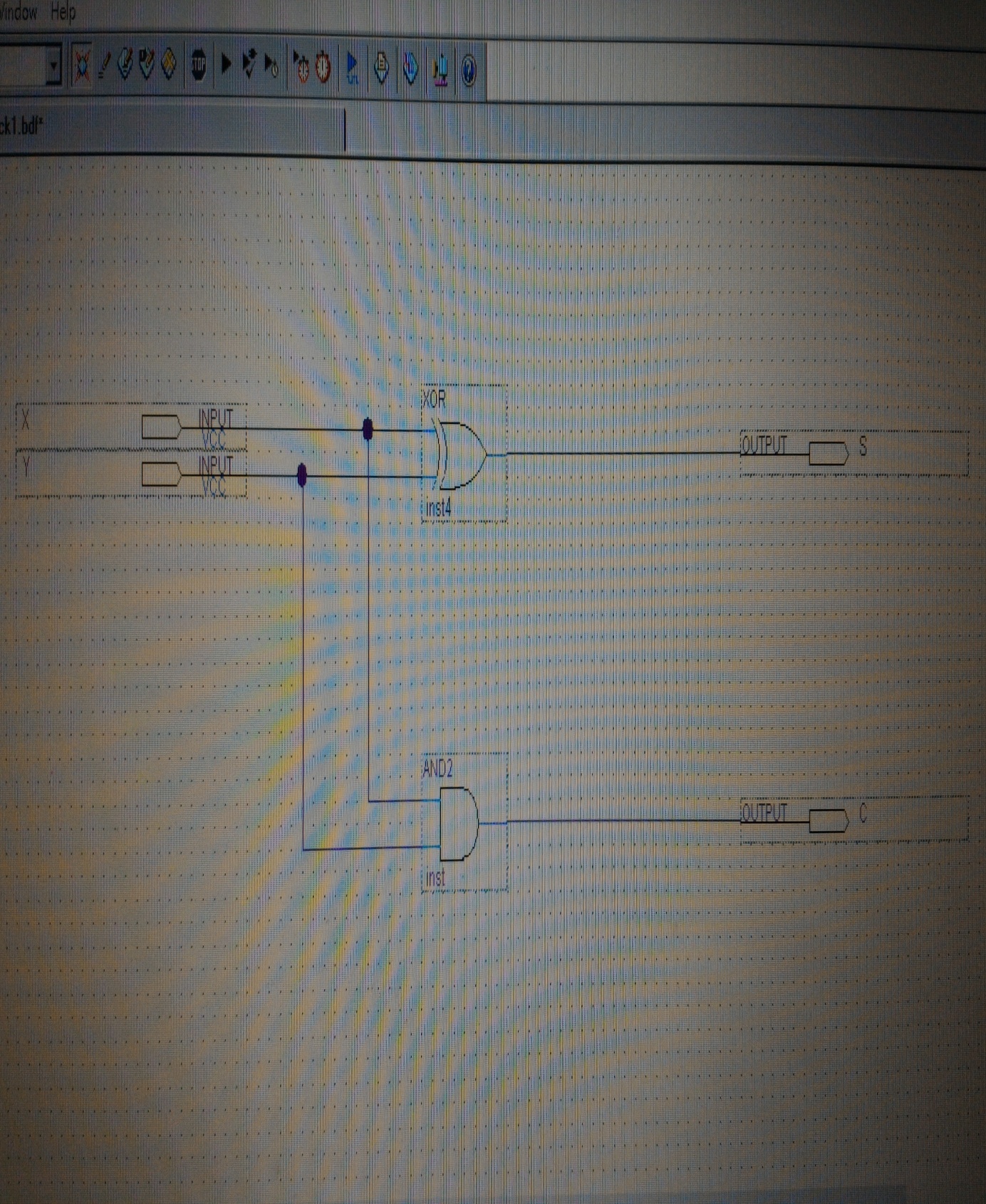
我們製作這個電路的流程分別是:

Half Adder -> Full Adder -> 4 bits Ripple Carry Adder -> 4 bits Adder-Subtractor -> Overflow Detection for Addition and Subtraction -> Finish

1. Half Adder:

這個 Half Adder 是由兩個inputs X 和 Y、一個XOR、一個And以及兩個outputs S 和 C所組成，而這一個電路最主要的功能就是去針對一個bit也就是input X和一個bit也就是input Y的相加，那麼S所代表的含意就是這兩個bit所加出來的Sum，而這個C所代表的含意就是這兩個bit相加後所得出的Carry，下圖所呈列的Truth Table就是四種組合所相加出來的結果:

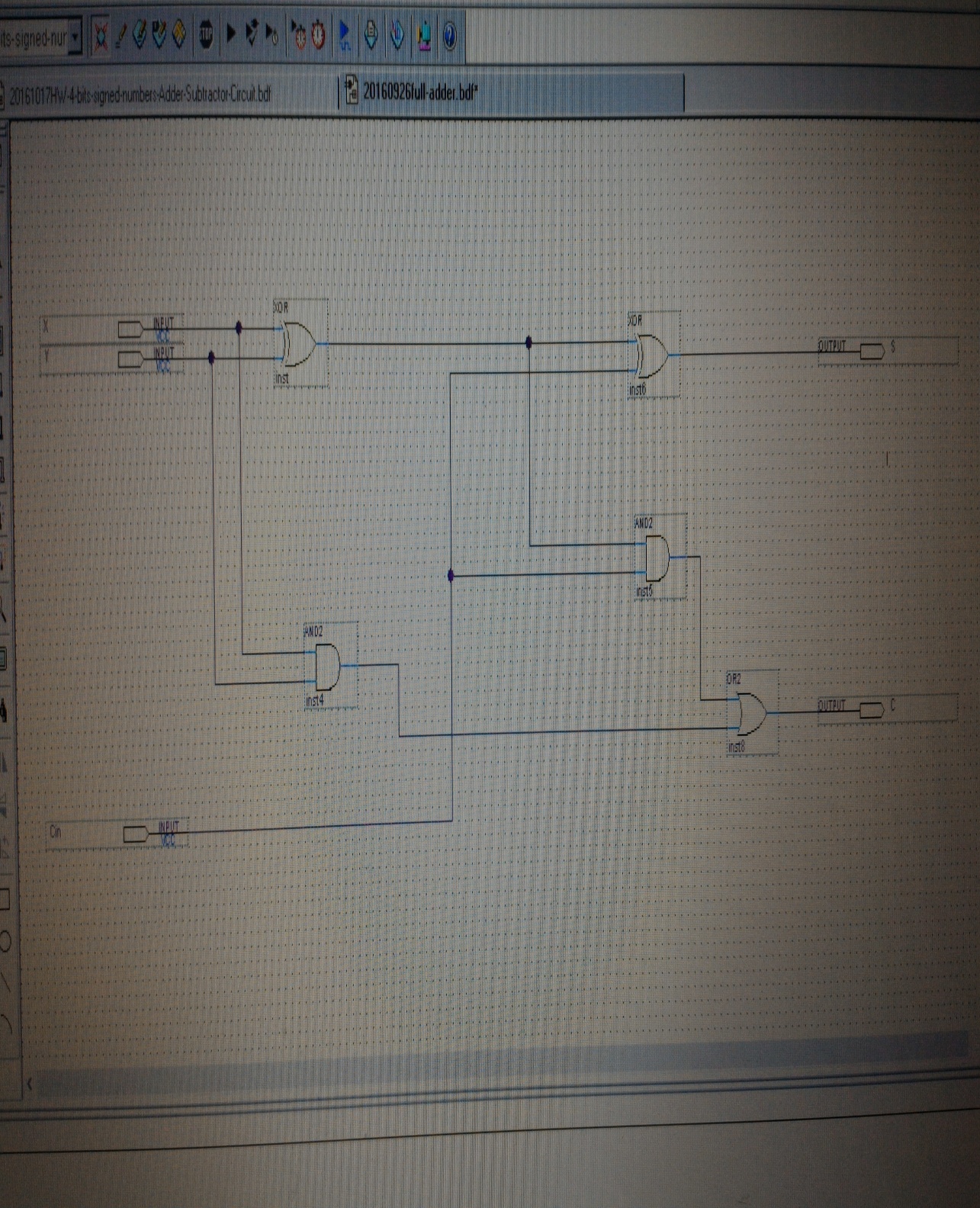
|  |  |  |  |
| --- | --- | --- | --- |
| Inputs | | Outputs | |
| X | Y | C | S |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |



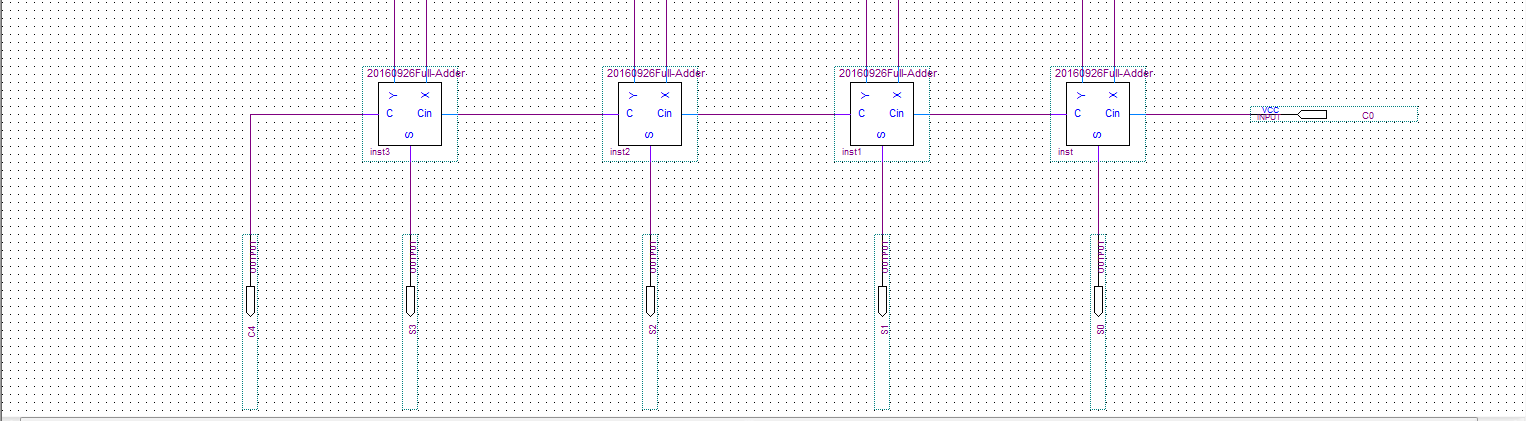
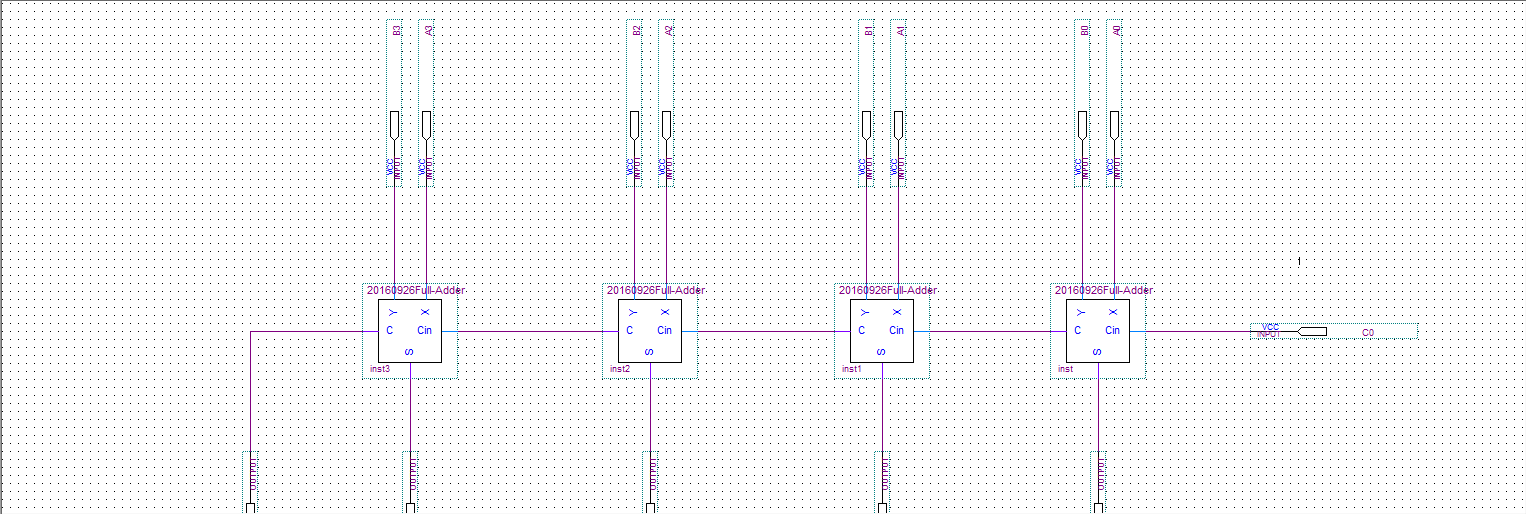
1. Full Adder:

這個Full Adder是由兩個Half Adder來去組成的，所以其內部分別由2個inputs X、Y和Cin、2個XOR、2個AND、1個OR來去構成，這個電路的功能大致與Half Adder是十分的相似，可是其又多了一個input也就是Cin，其所代表的含意就是Carry in，它負責去接受前一個Full Adder所產生的Carry out，來去將其

加入到目前bit的運算當中。

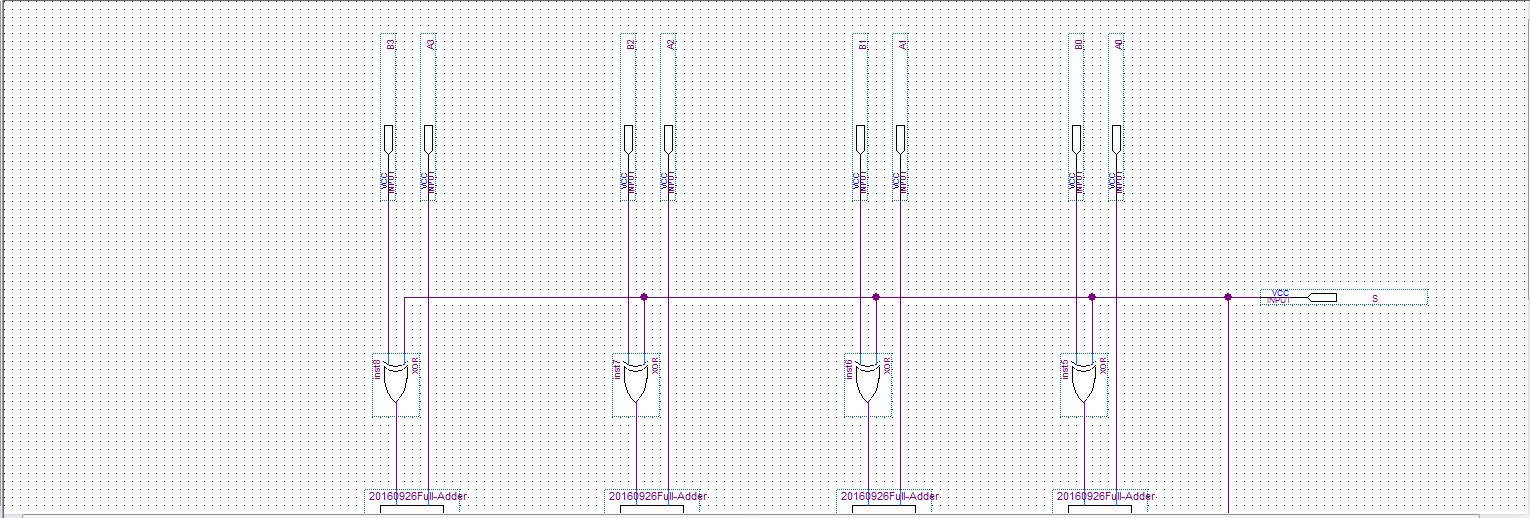
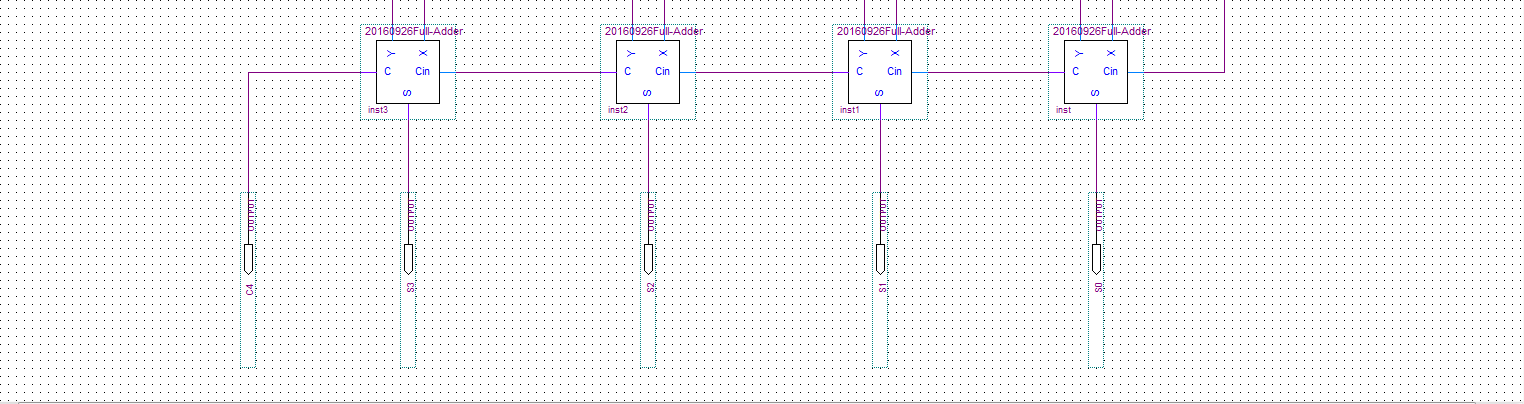


1. 4 bits Ripple Carry Adder:

這個Ripple Carry Adder它是由剛剛上頭所製作的Full Adder所組成的，總共包括4個Full Adder、5個outputs分別是S0、S1、S2、S3以及C4，S0、S1、S2、S3代表每一個bit所加出來的Sum，而C4代表的是最後一個bit的Carry out、9個inputs分別是4 bits的A和4 bits的B以及C0，而4 bits的A和4 bits的B就是我們所要進行運算的4 bits的binary，而C0就是代表一開始的Carry in通常是等於0，這一個Ripple Carry Adder最主要的功能就是將其2個4 bits的binary來去進行相加的動作，並將相加後結果輸出。

1. 4 bits Adder-Subtractor:

這個電路的組成是4 bits Ripple Carry Adder的延伸，它多了4個XOR以及1個input S並且和C0串連在一起，也就是S是多少那麼C0就會是多少，那這個input S最主要的功能就是去判斷目前所要做的運算是加法還是減法，當S=0時，則是要進行加法的運算，也就是此電路變成加法器(Adder)，當S=1且C0=1時，則是要進形減法的運算，也就是此電路變成減法器(Subtractor)，那S會和上頭Full Adder的4 bits的B的每一個bit進行XOR，來去將B進行2s complement的動作，隨後再和另外一組4 bits的A進行運算 ，透過這樣的方法就能夠讓其電路實現加法運算和減法運算的功能。



1. Overflow Detection for Addition and

Subtraction:

那麼在做完4 bits Adder-Subtractor的時候，此4 bits signed numbers Adder/Subtractor with an overflow detector的電路基本上已經是快完成了，就只差Overflow Detection for Addition and Subtraction這一個部份而已，而這個Overflow Detection的做法十分的簡單只需再多增加1個XOR和1個output V，並且將Full Adder的Carry out中的C4以及C3連接到XOR來去判斷是否有Overflow，當V=0時，代表沒有overflow，反之，當V=1時，則代表有Overflow。那對於signed numbers常見的overflow如下所

示(以下的bit都是signed bit):

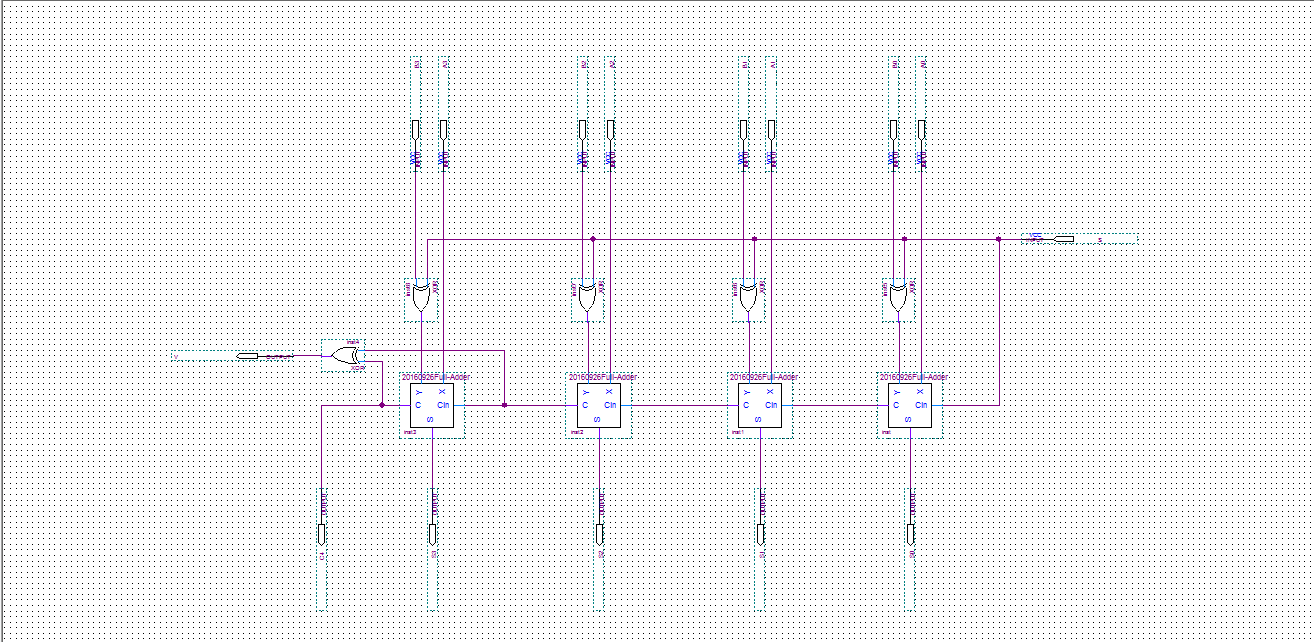
0 0 0 0

+0 -1 -0 +1

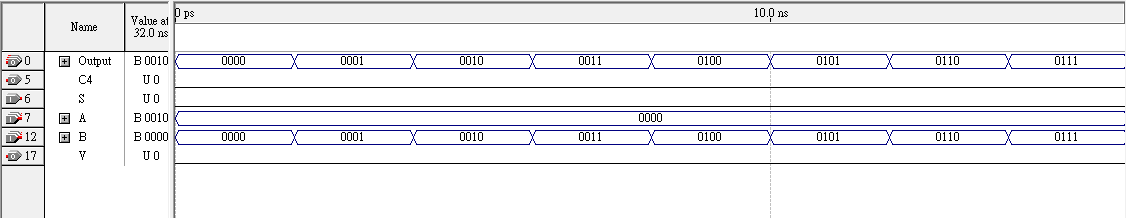
1 1 0 0

那麼經由了以上這些步驟之後就做出了我們所要

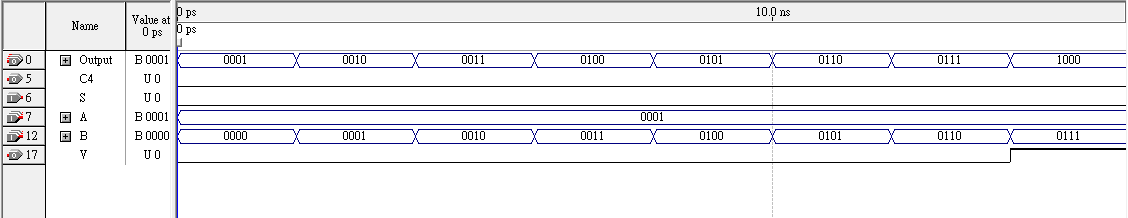
的4 bits signed numbers Adder/Subtractor with an overflow detector

1. circuit diagram
2. simulation results

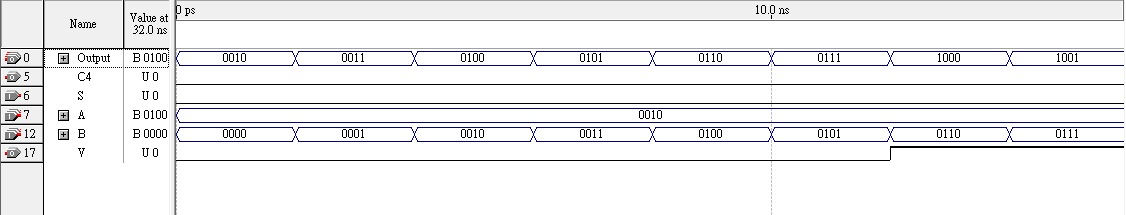
以下為加法器(S = 0)之模擬結果:



上圖為在加法器(S = 0) 情況下A = 0 (0000)加上 B = 0~7 (全為合法輸入)之結果。可以看到無論B之值為何，皆未產生Overflow (V = 1)的情形。

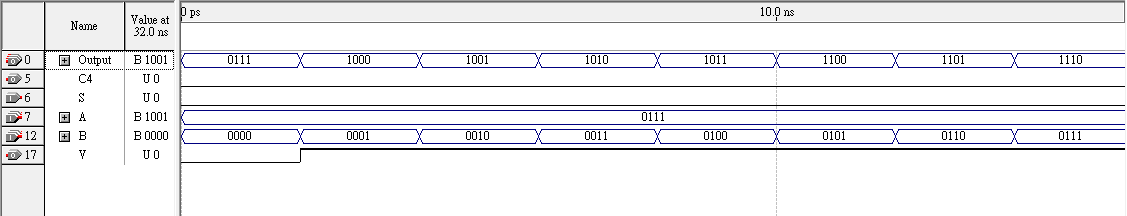


上圖為在加法器(S = 0) 情況下A = 1 (0001)加上 B = 0~7 (全為合法輸入)之結果。可以看到在B = 7(0111)時產生Overflow(V = 1)的情形。



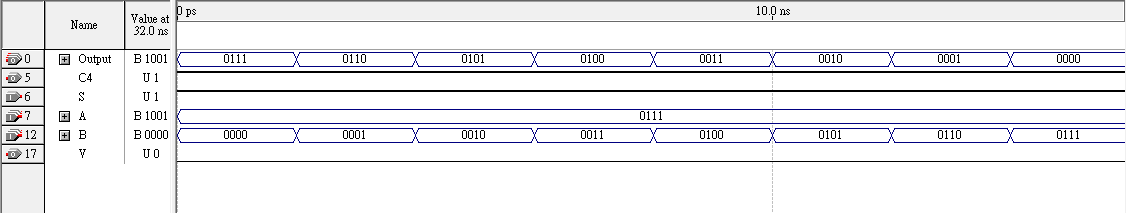
上圖為在加法器(S = 0) 情況下A = 2(0010) 加上 B = 0~7 (全為合法輸入)之結果。可以看到在B = 6和7(0110和0111)時產生Overflow(V = 1)的情形。

依此類推，我們直接觀測A = 7(0111) 加上 B = 0~7 (全為合法輸入)之結果。

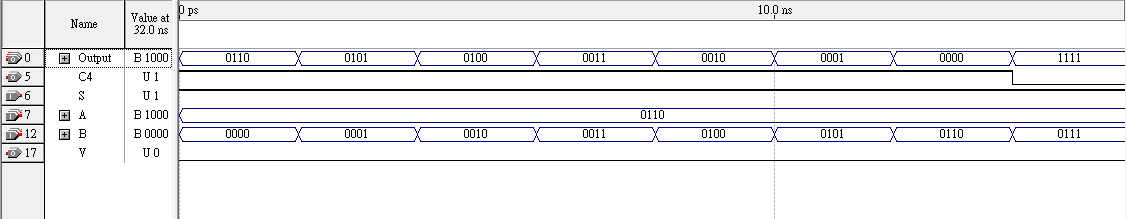


上圖為在加法器(S = 0) 情況下A = 7(0111) 加上 B = 0~7 (全為合法輸入)之結果。可以看到在B = 1到7(0001和0111)時皆產生Overflow (V = 1)的情形。

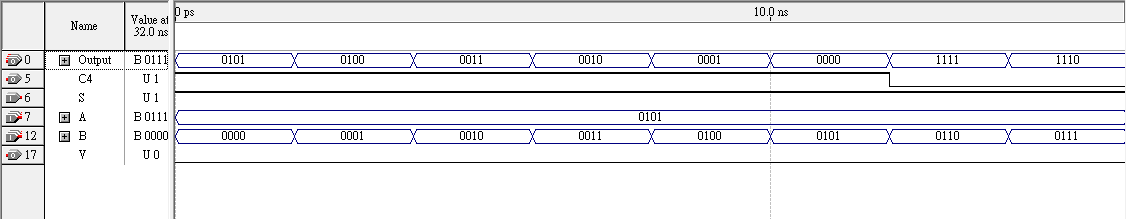
以下為減法器(S = 1)之模擬結果:



上圖為在減法器(S = 1) 情況下A = 7(0111) 減掉 B = 0~7 (全為合法輸入)之結果。可以看到無論B之值為何，皆未產生借位 (C4 = 0)的情形，因為未產生借位，故所有運算皆未產生Overflow。

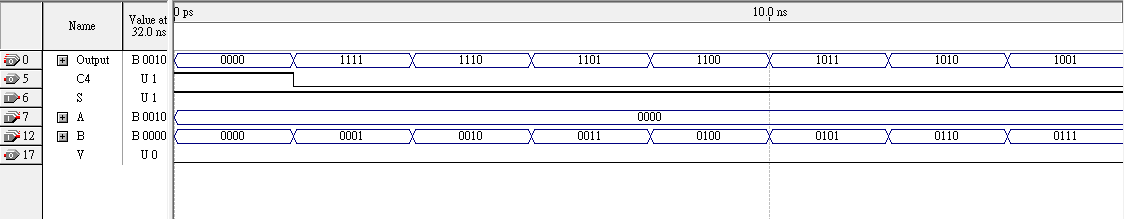


上圖為在減法器(S = 1) 情況下A = 6(0110) 減掉 B = 0~7 (全為合法輸入)之結果。可以看到在B = 7(0111)，產生借位 (C4 = 0)的情形，因為產生借位表most-significant-bit被改變(正負號改變)，因此此運算產生Overflow。



上圖為在減法器(S = 1) 情況下A = 6(0110) 減掉 B = 0~7 (全為合法輸入)之結果。可以看到在B = 6和7(0110和0111)，產生借位 (C4 = 0)的情形，因為產生借位表most-significant-bit被改變(正負號改變)，因此此運算產生Overflow。

依此類推，我們直接觀測A = 0(0000) 減掉 B = 0~7 (全為合法輸入)之結果。



上圖為在減法器(S = 1) 情況下A = 0(0000) 減掉 B = 0~7 (全為合法輸入)之結果。可以看到在B = 1到7(0001到0111)，皆產生借位 (C4 = 0)的情形，因為產生借位表most-significant-bit被改變(正負號改變)，因此此運算產生Overflow。

1. Problems

(1)在設計電路的過程當中，我們遇上最大的困難就是無法理解在Oveflow detect的電路中為何要使用XOR閘，而不是AND閘。因為我們的想法是，如果接出來的兩個皆為1，似乎就無法讓Overflow為1了。在查閱課本以及許多相關資料後，仍然無法理解，所幸在交大開放式課程中找到了我們要的答案。原來根本不會有兩者皆為1的情況發生，因為此電路的most-significant-bit是表示正負號，因此若是採用AND就會產生嚴重誤判，故使用XOR閘才是恰當的選擇。

(2)在進行減法器模擬的時候，Overflow一直為0 (V = 0)的結果讓我們有些許錯愕。原本的想法是以為Overflow應該在減法器中也要可以運作，在經過許多資料的查閱之後，發現到C4的奧妙。C4原本所表示的就是most-significant-bit是否進位，而在減法器中，因為其值和S(控制加法或減法器的關鍵)須一致且必為1。因此，當C4被借位變成0時，表most-significant-bit被改變(正負號改變)，因此此運算產生Overflow。

1. Discussions

階層式設計:此電路設計是透過階層式的設計，由Half Adder -> Full Adder -> 4 bits Ripple Carry Adder -> 4 bits Adder-Subtractor -> Overflow Detection for Addition and Subtraction 。當中若低層的電路有誤，將導致最後電路的失敗，是一個連動性的設計。在了解每個小電路的原理和運作方式後，得到最後的結果，期望在之後更複雜的數位系統實驗裡，能更清楚整體運作原理。

1. References

(1)Logic and Computer Design Fundamentals FIFTH EDITION P.182 FIGURE 3-45

(2)https://goo.gl/UhtX9L