**Lab 1: Verilog HDL**

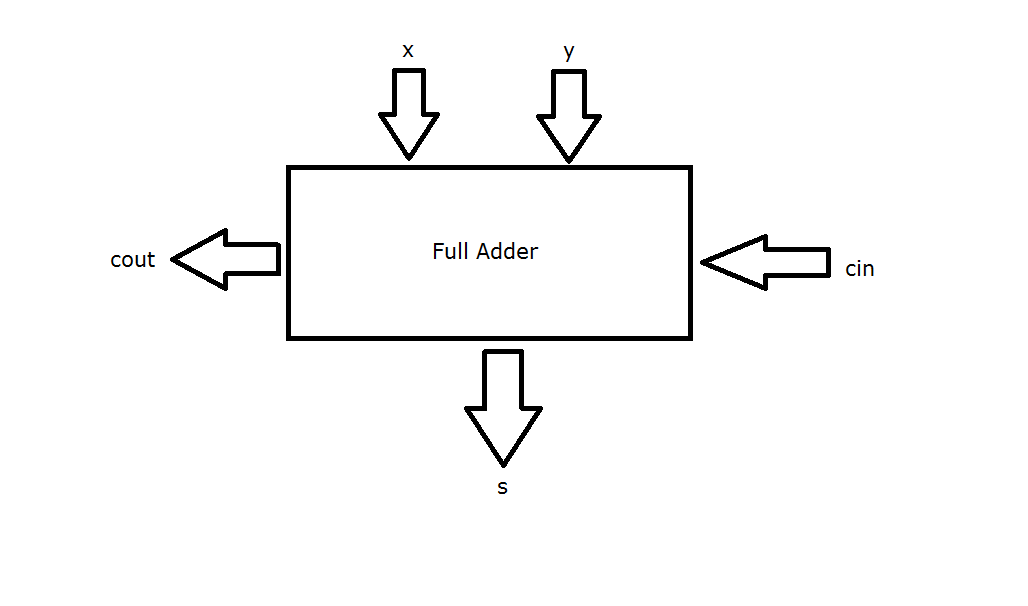
**1. Full Adder**

**Design Specification**

(1) Input: x, y, cin.

(2) Output: cout, s.

(3) Block diagram:

****

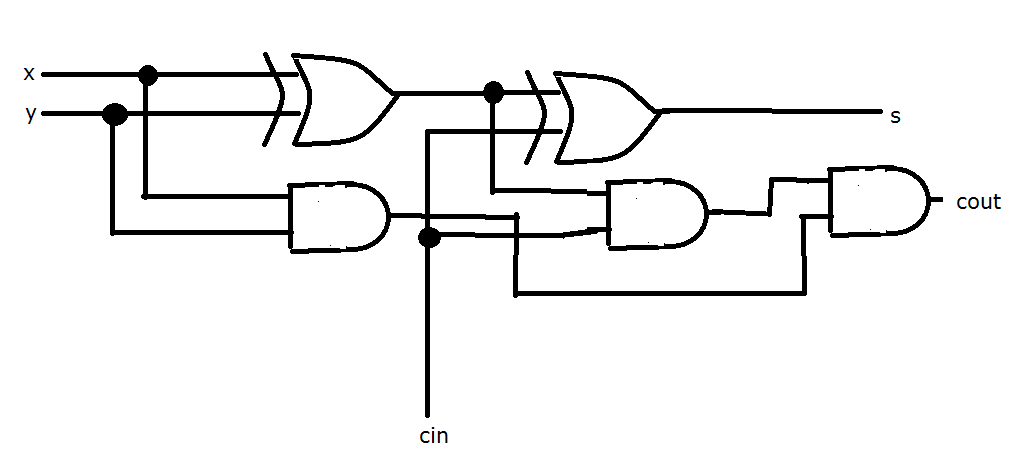
**Design Implementation**

(1) Logic function:

s = x^y^cin;

cout = (x&y) | (cin&(x^y));

(2) Logic Diagram:



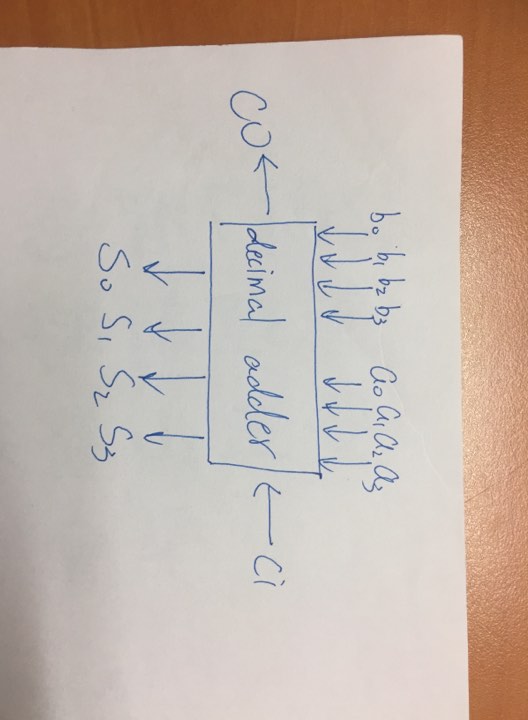
**2. single digit decimal adder**

**Design Specification**

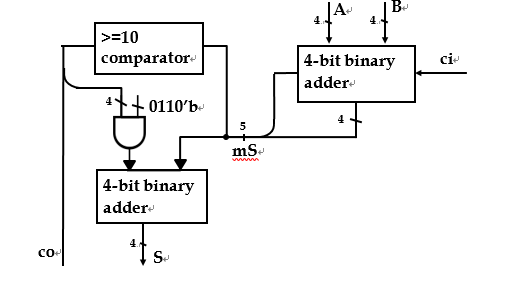
(1) Input: a[3:0], b[3:0], ci.

(2) Output: s[3:0], co.

(3) Block diagram:



詳細的block diagram，說明請見Discussion



**Design Implementation**

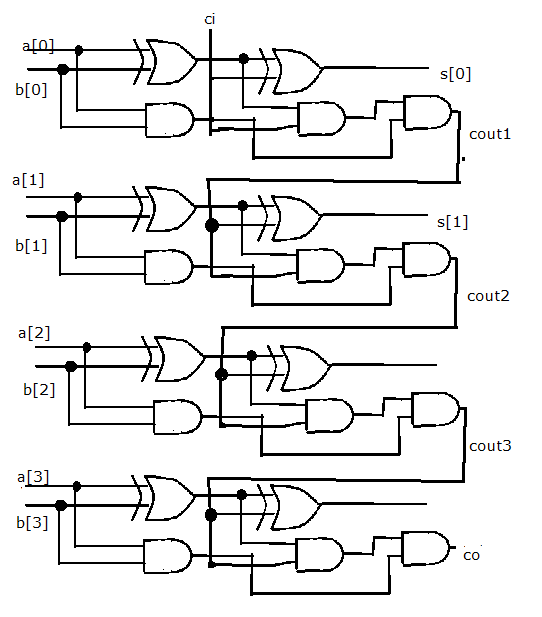
(1) Logic Function:

我用Ripple Carry Adder實作本題，而每個Full Adder輸出中有一進位值，在我的Verilog程式碼中是以wire儲存此值

s[i] = a[i]^b[i]^co[i];

co = (a[3]&b[3]) | (ci[3]&(a[3]^b[3]));

(2) Logic Diagram: (Full adder部分)



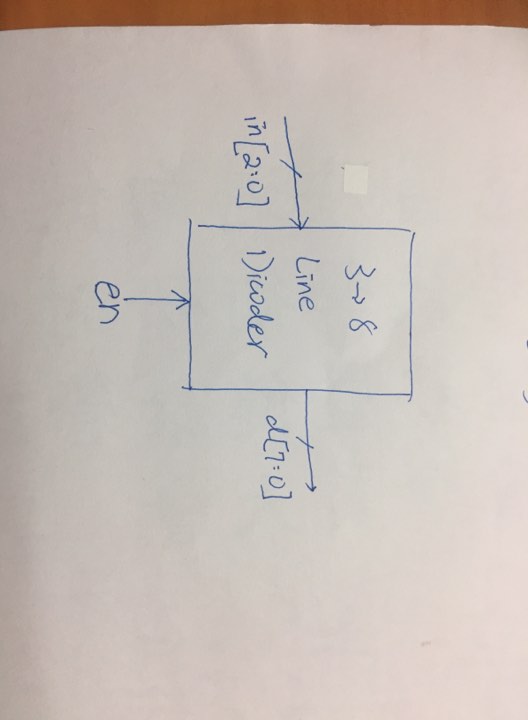
**3. 3-to-8-line decoder with enable**

**Design Specification**

(1) input in[2:0], en.

(2) Output d[7:0].

(3) Block diagram:



**Design Implementation**

(1) Logic function

d[0] = en&(~in[2])&(~in[1])&(~in[0]);

d[1] = en&(~in[2])&(~in[1])&(in[0]);

d[2] = en&(~in[2])&(in[1])&(~in[0]);

d[3] = en&(~in[2])&(in[1])&(in[0]);

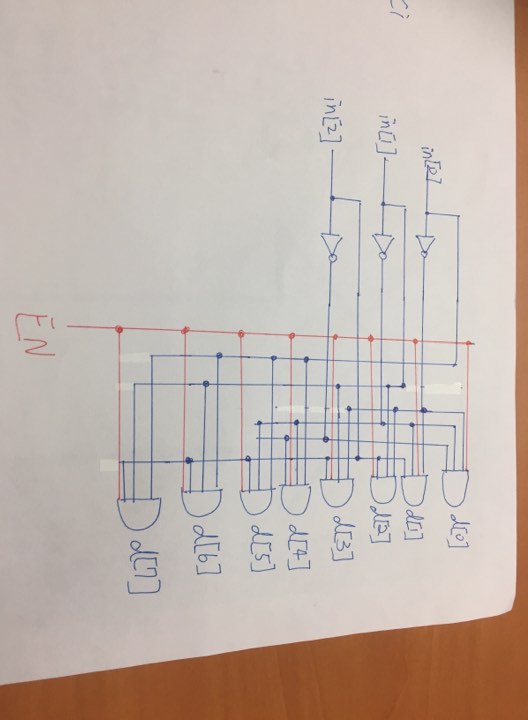
d[4] = en&(in[2])&(~in[1])&(~in[0]);

d[5] = en&(in[2])&(~in[1])&(in[0]);

d[6] = en&(in[2])&(in[1])&(~in[0]);

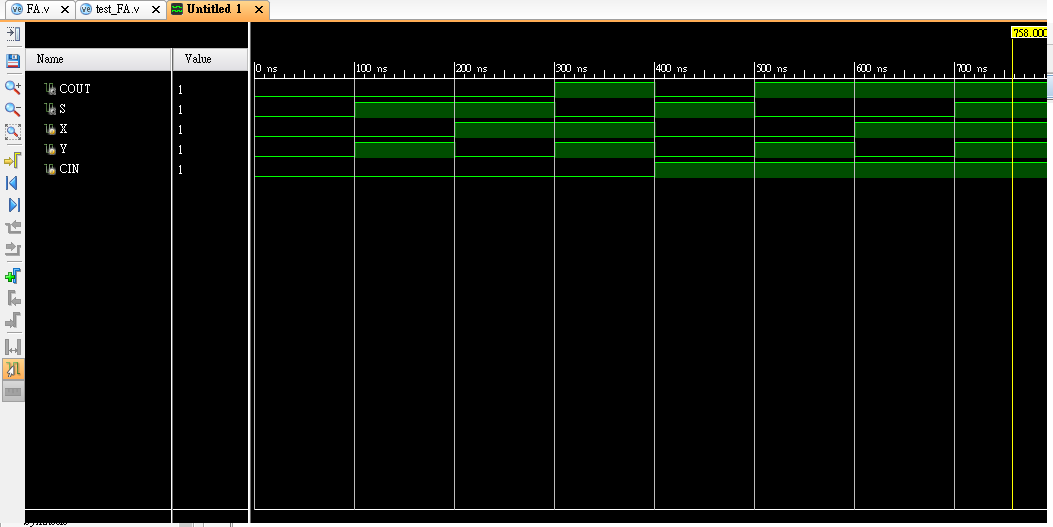
d[7] = en&(in[2])&(in[1])&(in[0]);

(2) Logic diagram

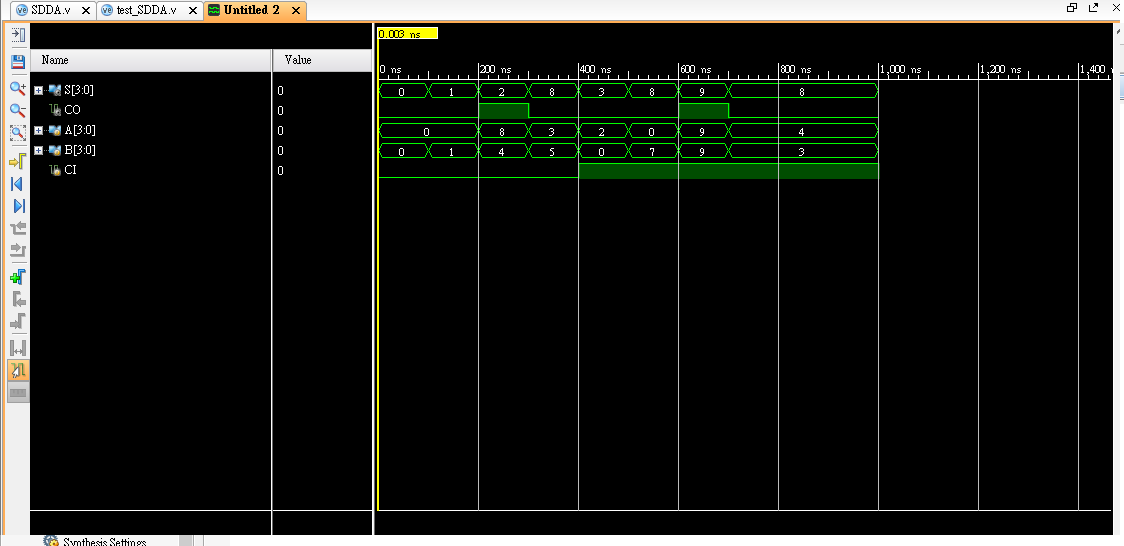
****

**Discussion**

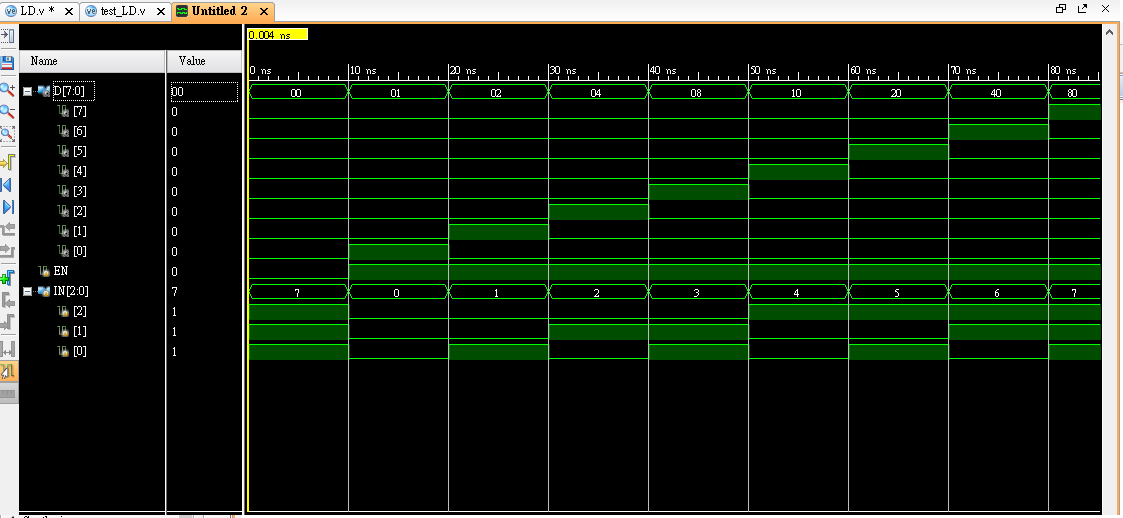
實驗一跑了好幾次波形都是Z跟X，最後幸好有熱心助人的李佳璋同學幫忙debug才發現問題在testbench跟module的接線對不上,以及一些分號忘記打。波形圖如下：



　　實驗二的加法器：先把兩數相加，為了避免溢位把輸出設成五個數(即code中的mS)。接著讓和經過一個大於比較器，若大於10就把co接成4個bit並與-10(以2補數表示即0110)逐位元過and閘，並與原數相加，就相當於減10的過程了。反之若co小於10就維持輸出不變。波形圖如下所示：



實驗三是很基本的組合邏輯複習，一個decoder就是各個最小項之和。接著把致能接到每個and閘上就可以了。上學期的講義內有code。經模擬後波形圖如下；



**Conclusion**

初學Verilog，我會建議與我做一些相同的事情：

1. 買一本中文書來看:因為Verilog不像C語言是一種程式語言，而是電路描述語言，而且老師上課所教的不一定能包含到實作時種種問題。所以我會建議買一本中文的書來看。目前品質好的中文書為數不少，而且相較原文的便宜。

2. 與同學或其他人討論: 不會的地方就問助教或同學，那怕真正的bug只是很細節的東西(像port名子打錯或忘記加分號)。尤其學長姐給的家傳的code一定要讀懂，理解他人的程式碼對練習或創意的發想很有幫助。

如果你很幸運認識就讀高職電子電機科的朋友,問爆他們就對了。

**References**

《Verilog硬體描述語言數位電路》，鄭信源，儒林圖書公司(2016)。