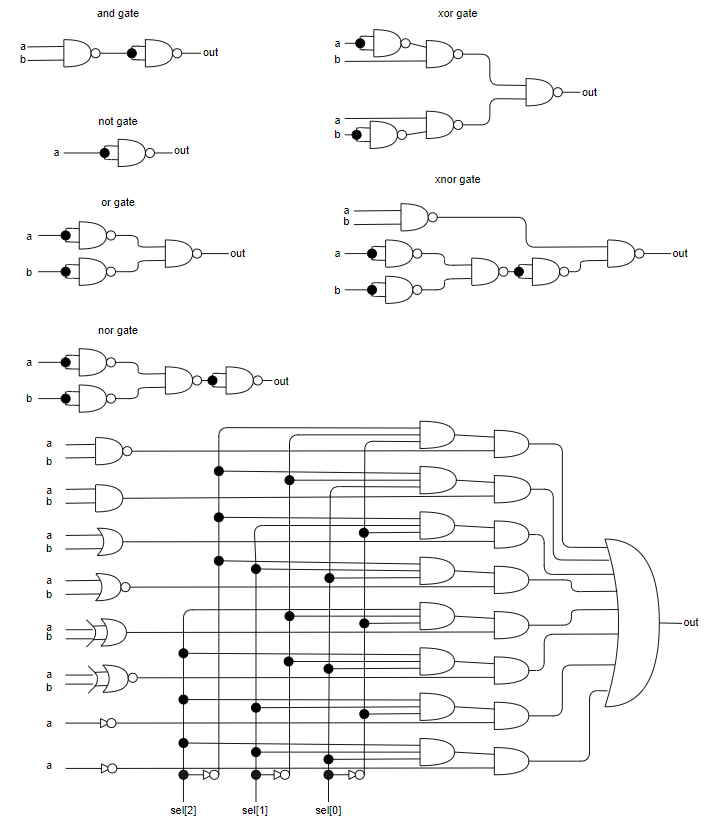
Lab 2 Team 20 report

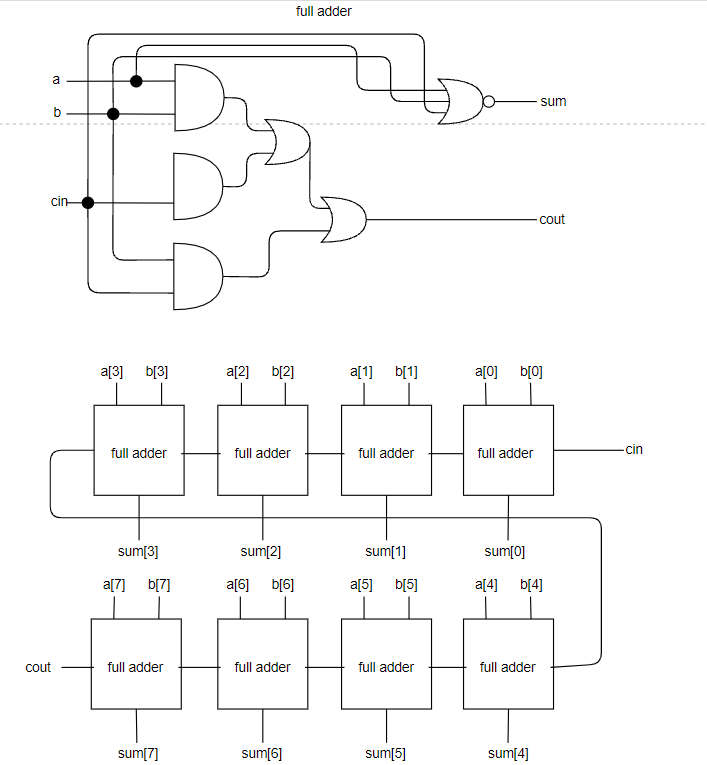
109062222 徐嘉徽

109062119 李佳栩

Basic\_1:

Basic\_3:Half adder是兩個1-bit的input相加，獲得一個sum跟carry

Full adder則是將三個1-bit的input相加，獲得一個sum跟carry

Advanced\_1:

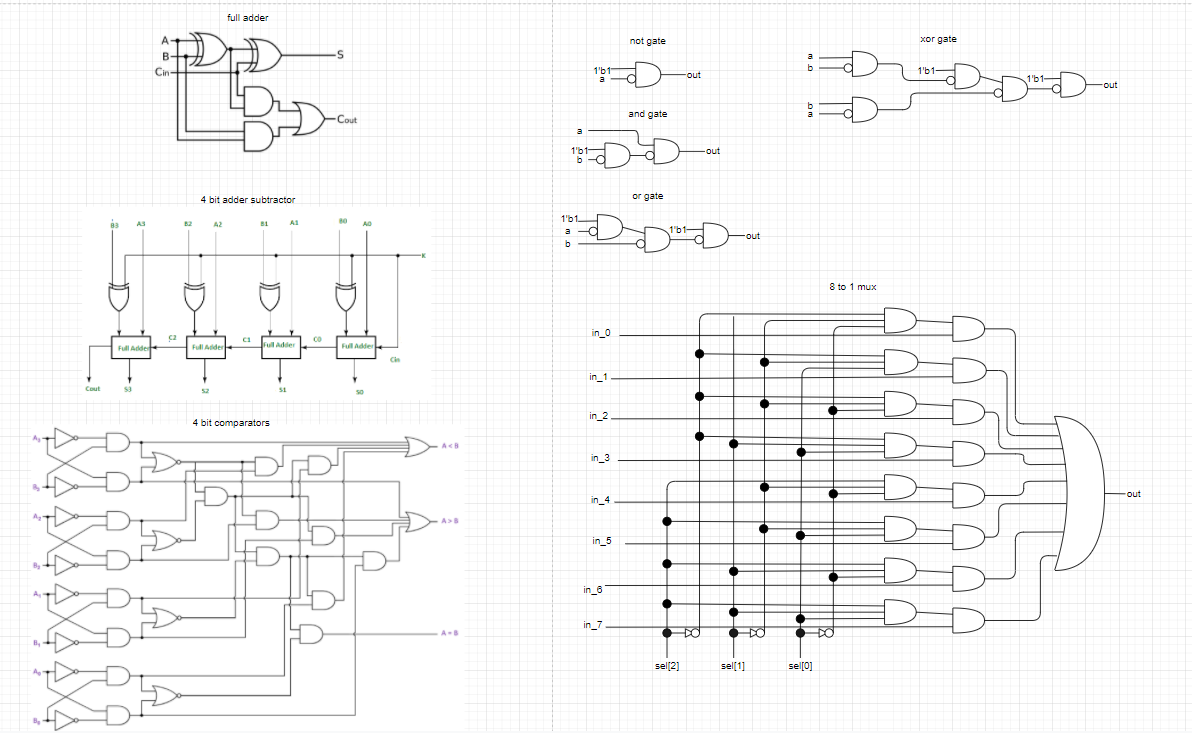
我們先依照basic 3的full adder module做出full adder。

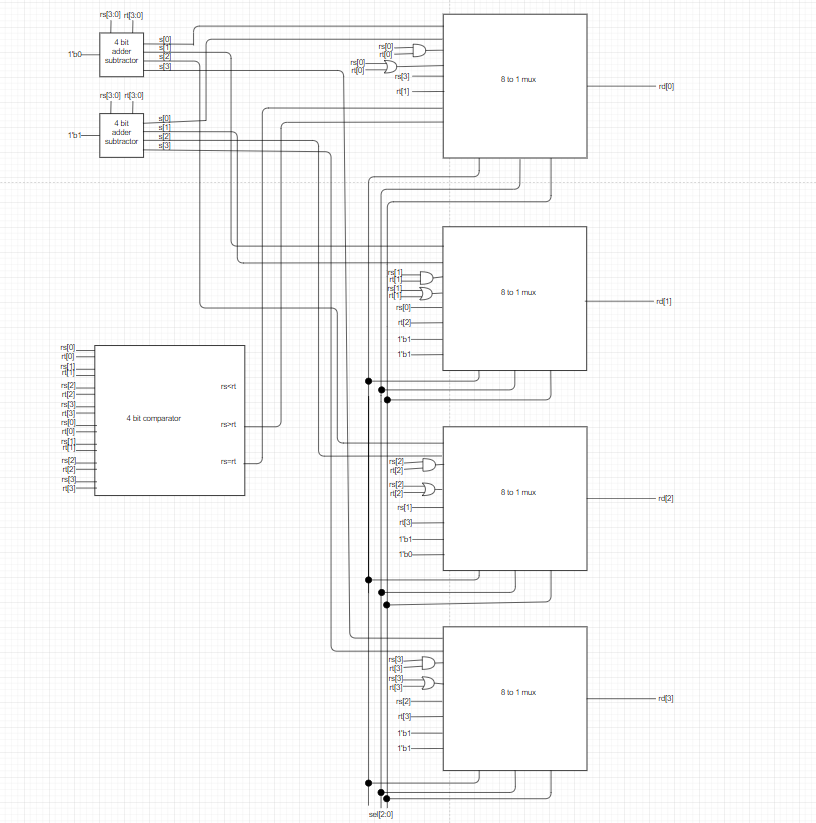
之後再照著pdf上給的圖照著接線，便是8-bit ripple carry adder。

以下便是我們testbench所測試的幾個數據:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| a[3:0] | 10000000(80) | 10000000(80) | 01010101(55) | 01010101(55) | 01010101(55) |
| b[3:0] | 00000001(01) | 10101010(aa) | 10101010(aa) | 00001111(0f) | 01010101(55) |
| cin | 0 | 0 | 1 | 0 | 0 |
| sum[3:0] | 10000001(81) | 00101010(2a) | 00000000(00) | 01100100(64) | 10101010(aa) |
| cout | 0 | 1 | 1 | 0 | 0 |

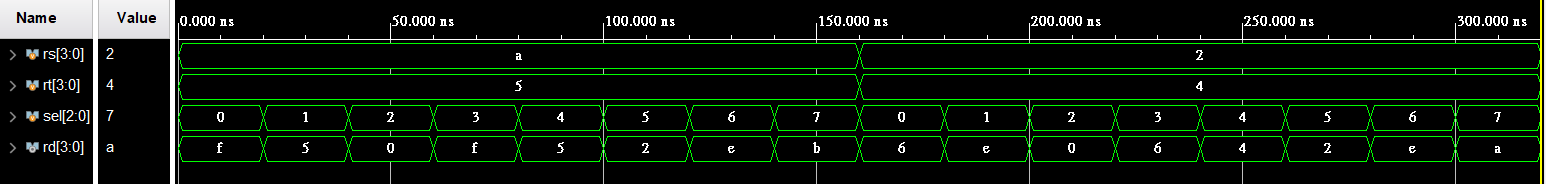
由此可知，達到8-bit ripple carry adder的作用

Advanced\_2:

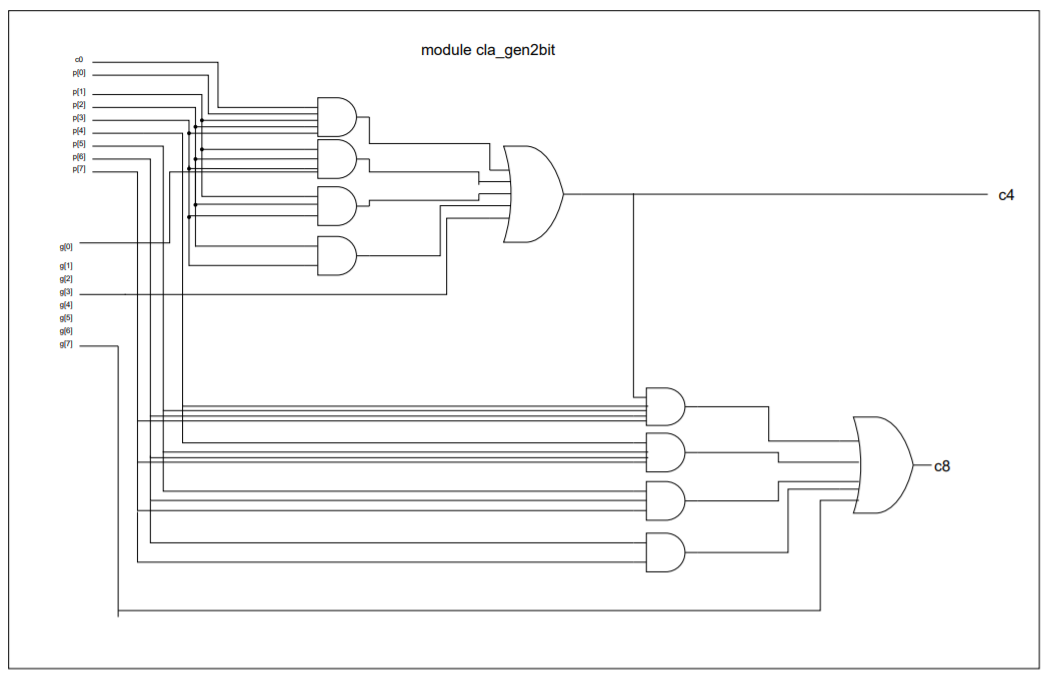


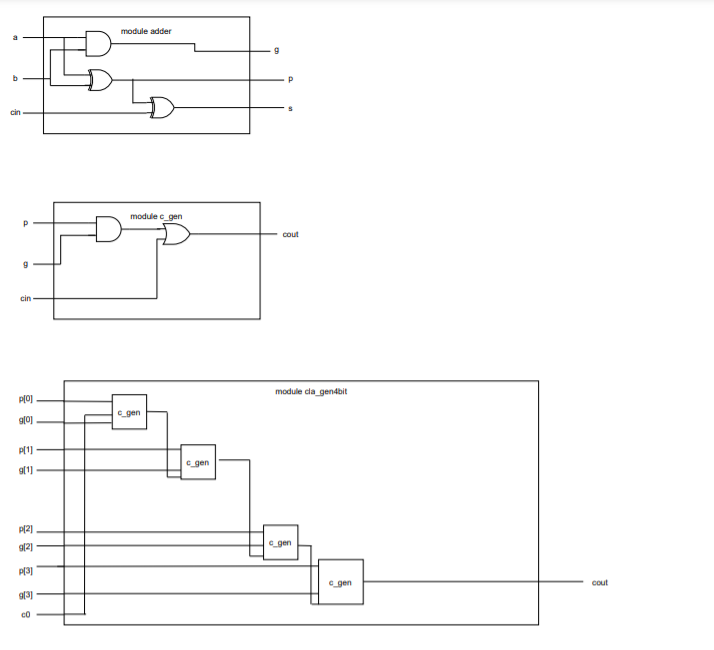
首先先利用universal gate做出and、not、or、xor這些basic logic gate。為了完成ADD跟SUB的功能，利用剛剛做出的gate寫出4-bit adder subtractor。Bitwise and則是將rs與rt的每個bit各自and起來，Bitwise or也是將rs與rt的每個bit各自or起來。Left shift跟right shift則是照著rd={rs[2:0],rs[3]},rd={rt[3],rt[3:1]}。最後為了完成rs==rt跟rs>rt，寫出一個4 bit comparator以達成比大小的功能。接著利用4個8 to 1 mux接上selection sel[2:0]，並各自在mux input接上每一種function output的第0 bit,第1 bit,第2 bit及第 3 bit。都是接第0 bit的mux輸出便是rd[0]，以此類推便會得到rd[3:0]。

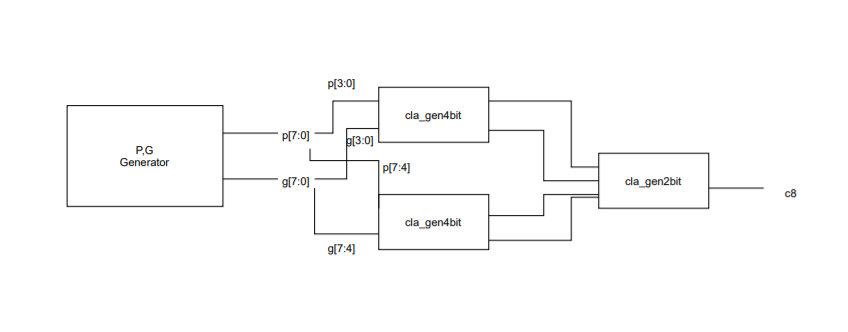
我們將初始值設為rs=1010,rt=0101,sel=000，並每經過20ns就將sel+1，於是每過160ns sel就會重新回到0。前160ns的rs=1010,rt=0101，後160ns為rs=0010,rt=0100。



Advance\_3:



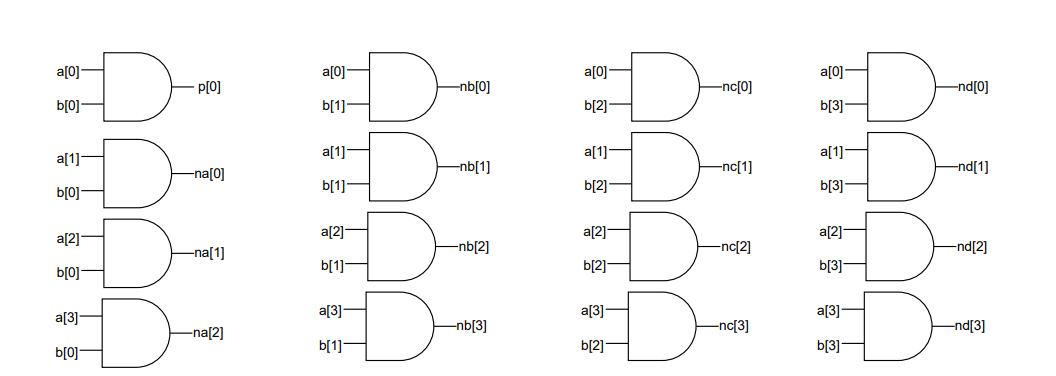




Advancd\_4:

這題是要我們實作一個4個bit的乘法器，根據助教提供的圖，我先把每一個位元互乘的答案先記錄下來，也因為二進位的乘法，兩個值如果都是1那麼乘積才會是1，其他都是0，所以這點我們可以用and gate 來實現。

下圖便是我把每一個值記錄下來。



那我們知道在做乘法的時候每往下一格，位數就多一個，每兩層相加的進位便可以直接加到下一層，以下是範例

a2 a1 a0

X b2 b1 b0



---------------------------------------------------------------------

na[3] na[2] na[1] na[0]

+ nb[3] nb[2] nb[1] nb[0]

----------------------------------------------------------------------

Cout s3 s2 s1 s0

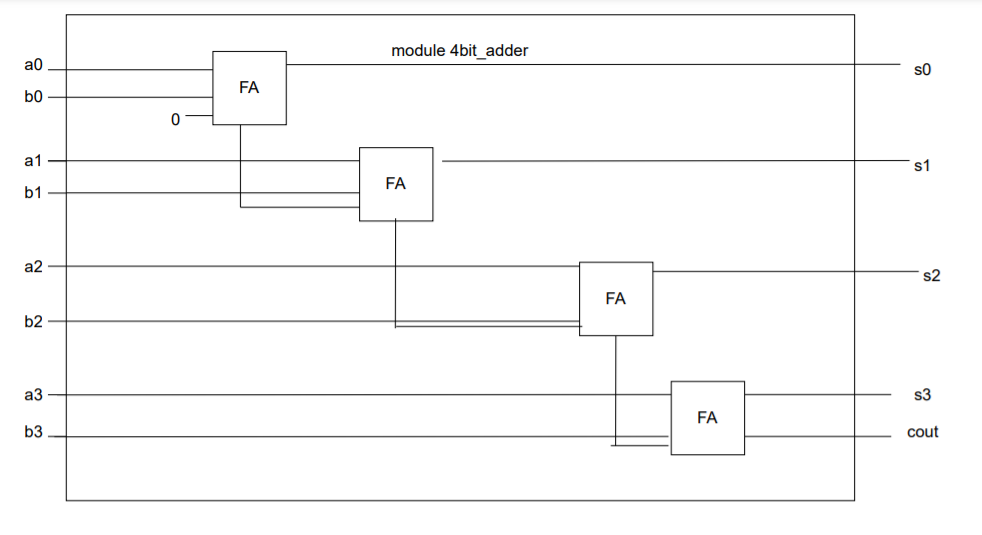
+ nc[3] nc[2] nc[1] nc[0] p[0]

------------------------------------------------------------------------

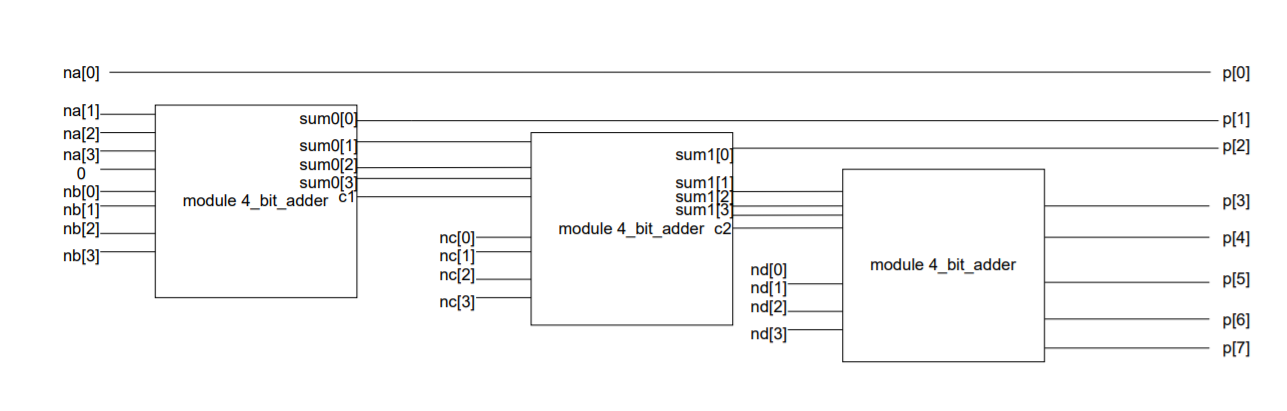
p[5] p[4] p[3] p[2] p[1]

每一次必須把最小位數傳給output，carry out則加到下一層的最高位。

為了方便我寫了一個4個bit的adder。

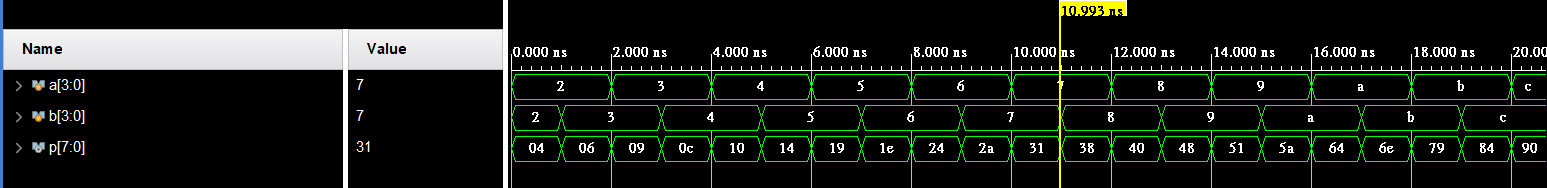


每做一次加法的最小位數就直接傳給output，剩下的位數加上carry out則加到下一層，重複三次即可。



測試：

我先把a跟b的數字設成一樣，每隔一奈秒a+1，每隔兩奈秒b+1，看答案有沒有符合a\*b。



Advance\_5:

這題是要我們實作一個能夠自己找出錯誤的testbench，因為之前可能都是用肉眼看，如果電路很複雜其實很容易看錯，所以我們直接在testbench裡面設定如果算出來的答案不符合我們真正的答案，會有一個訊號告訴我們，結束後也會有一個信號告訴我們。

那我是利用repeat這個寫法，我們知道a和b都是4個bit，所以總共有2的8次方，還加上cin，利用loop的寫法可以簡短的寫出來，然後在迴圈裏面寫一個if-else，如果不符合答案error就會變成1，反之，則變成0，那這題比較需要細心的地方就是每隔5奈秒會有一個input，那收到input的一奈秒後，必須檢查是否有error，在時間上面問題可能會比較多，不過靠著助教的波形圖，我也成功寫出來了。

FPGA:

這題是要利用第二題所做的decode and excute來在FPGA上實現，我先設一個7個bit的陣列當作我的7個燈管。一開始我的疑問是只有4個bit要怎麼implement在7段顯示器上，後來我想到的辦法是把這個rd丟到4-16的decoder，這樣就可以有一個16個bit的陣列，可以用這個陣列來操控我的7個燈管。

首先我先從A這個燈管觀察，總共有12個數字需要用到，所以我把這12個bit用or gate連起來，再連到屬於這個燈管的陣列，這樣不管我選到這12個數字中的哪個數字，A燈管一定會亮，其他以此類推，最後再燒到FPGA上就完成了，那我這題也是用gate-level寫出來的，不知不覺中對verilog也更熟悉了一點。

分工：

徐嘉徽負責1.2.3題

李佳栩負責4.5跟fgpa。

那我們report的分工也是跟自己的題目一樣。

心得：

李佳栩：這次的lab比上次的更有難度，需要更細心且對於這些電路有一定的了解，像是乘法器，以前完全沒有碰過，這也讓我學到看到新的題目時必須要從不同角度下去思考，如果拘泥於一種方法，很容易浪費時間，像是FPGA那題就讓我困擾了很久，希望下次的lab可以寫快一點，也終於渡過只能寫gate-level的階段了。