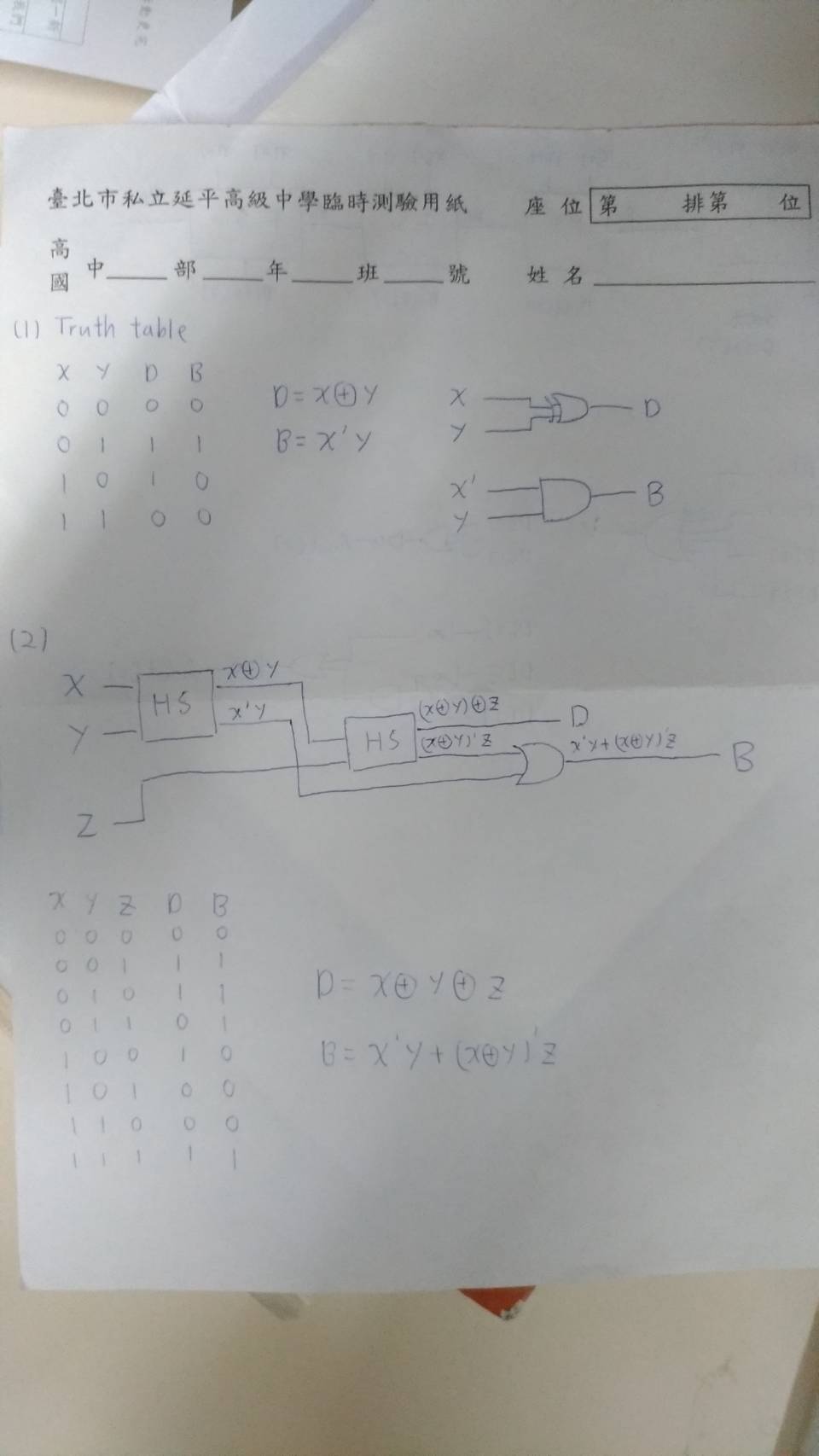
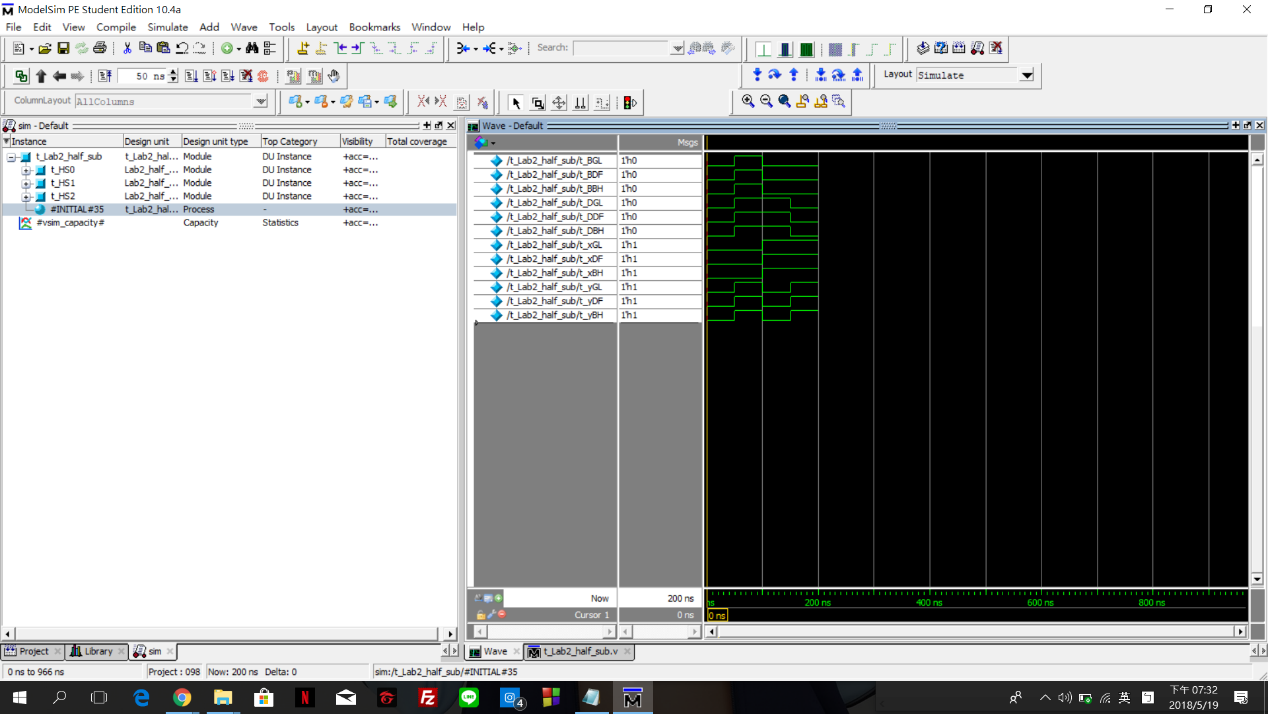
(1)

從Truth table利用k-map或透過觀察推倒得出布林代數式後，及利用講義上的half-adder可推出half-substractor。



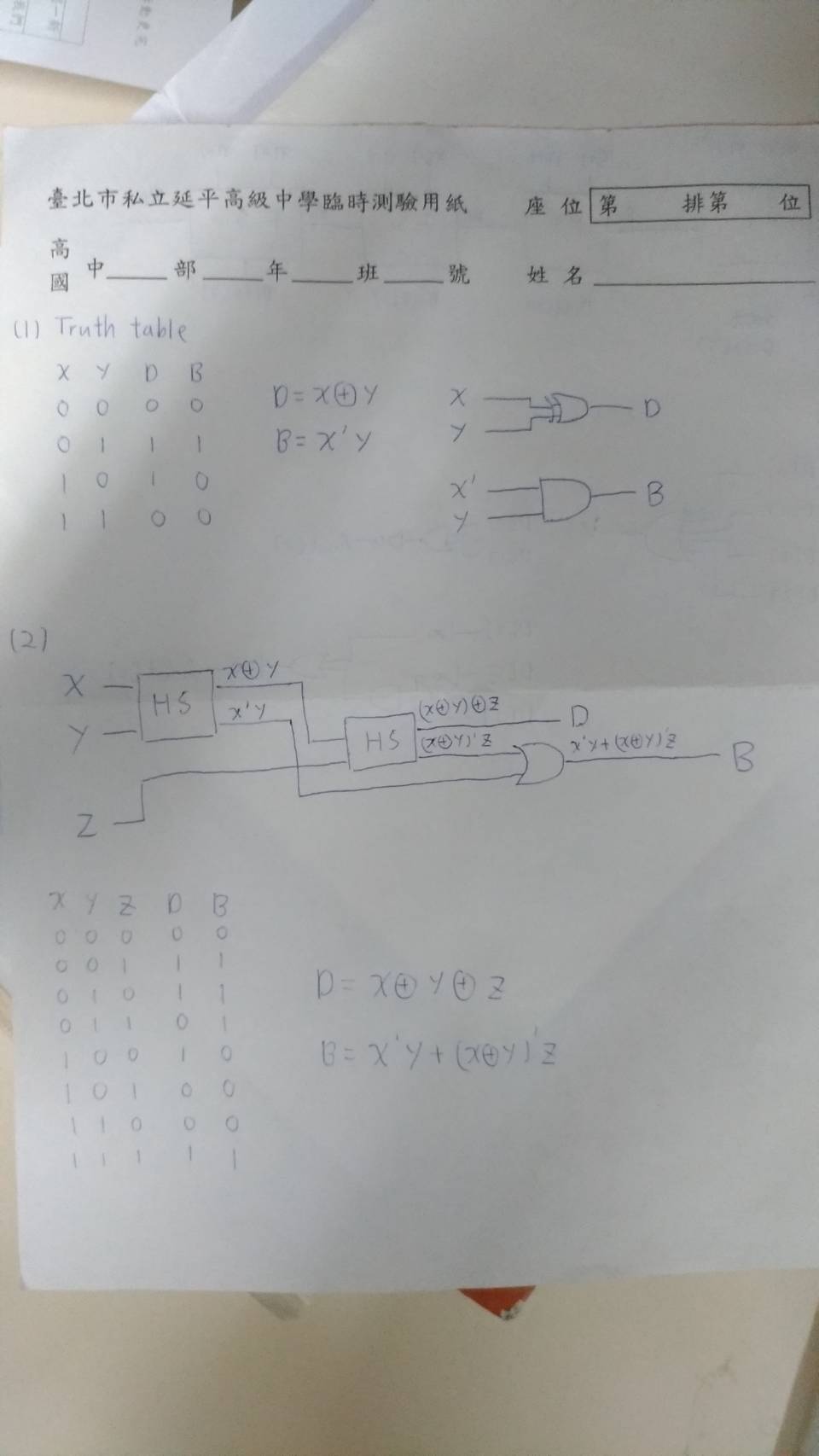
2A(a)iv (半減器)之模擬結果波形圖



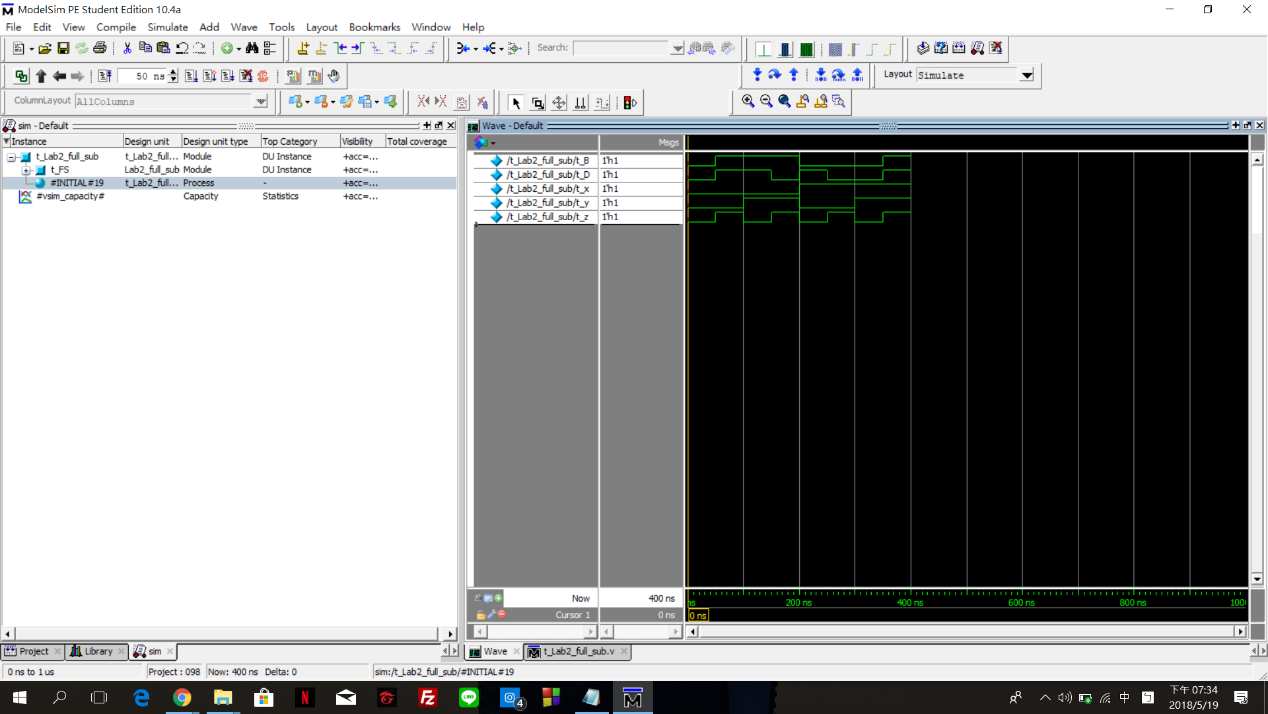
Gate level ， dataflow ， behavior所模擬出的波形皆一致，因此是沒有差異的，而它們及我算出的真值表都一樣，所以應該是正確。

(2)

從Truth table利用k-map或透過觀察推倒得出布林代數式後，及利用講義上的full-adder可推出full-substractor，也能先畫基本電路後，發現有些基本電路可替換成half-substractor。



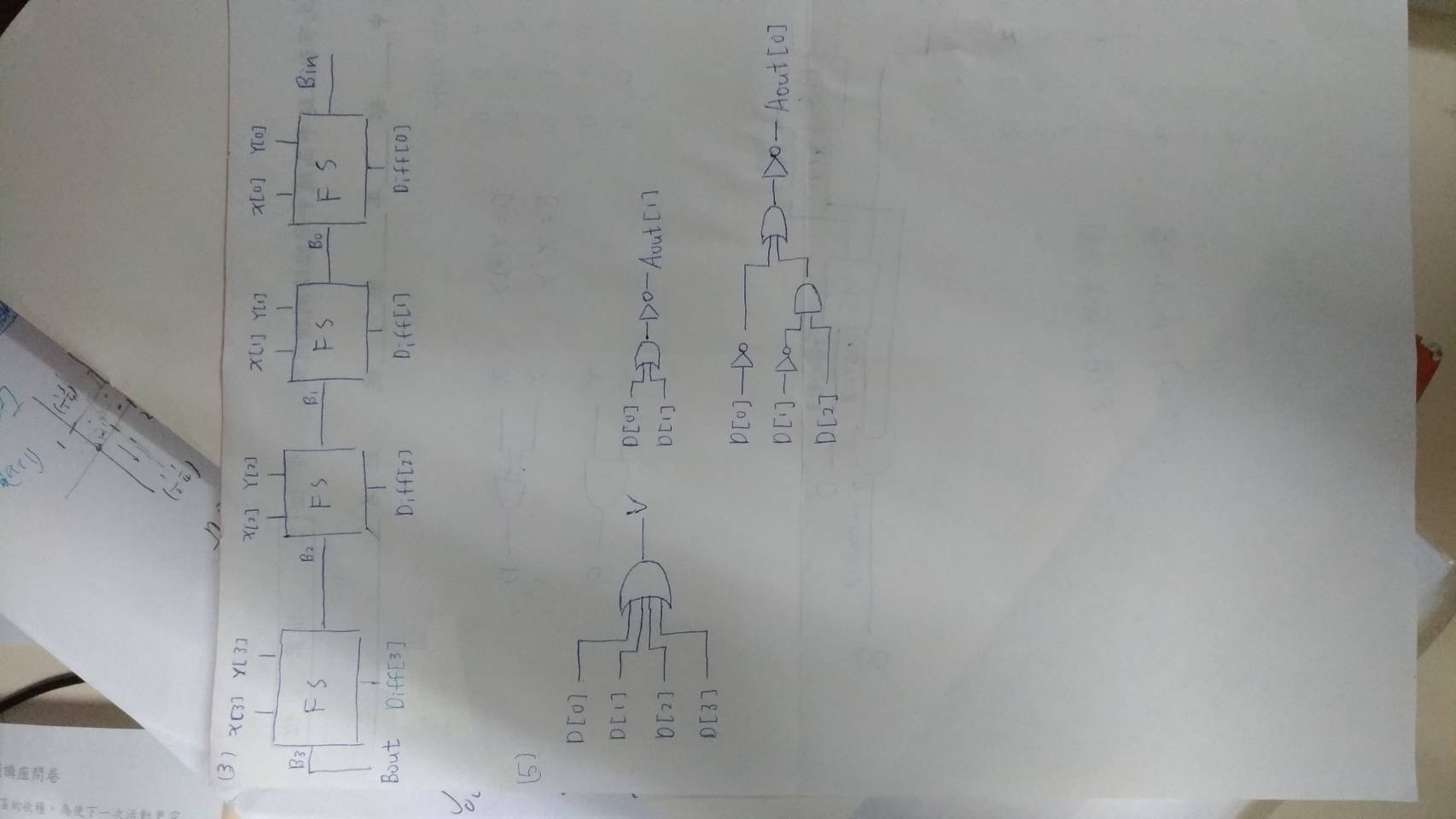
2A(b)ii (全減器)之模擬結果波形圖



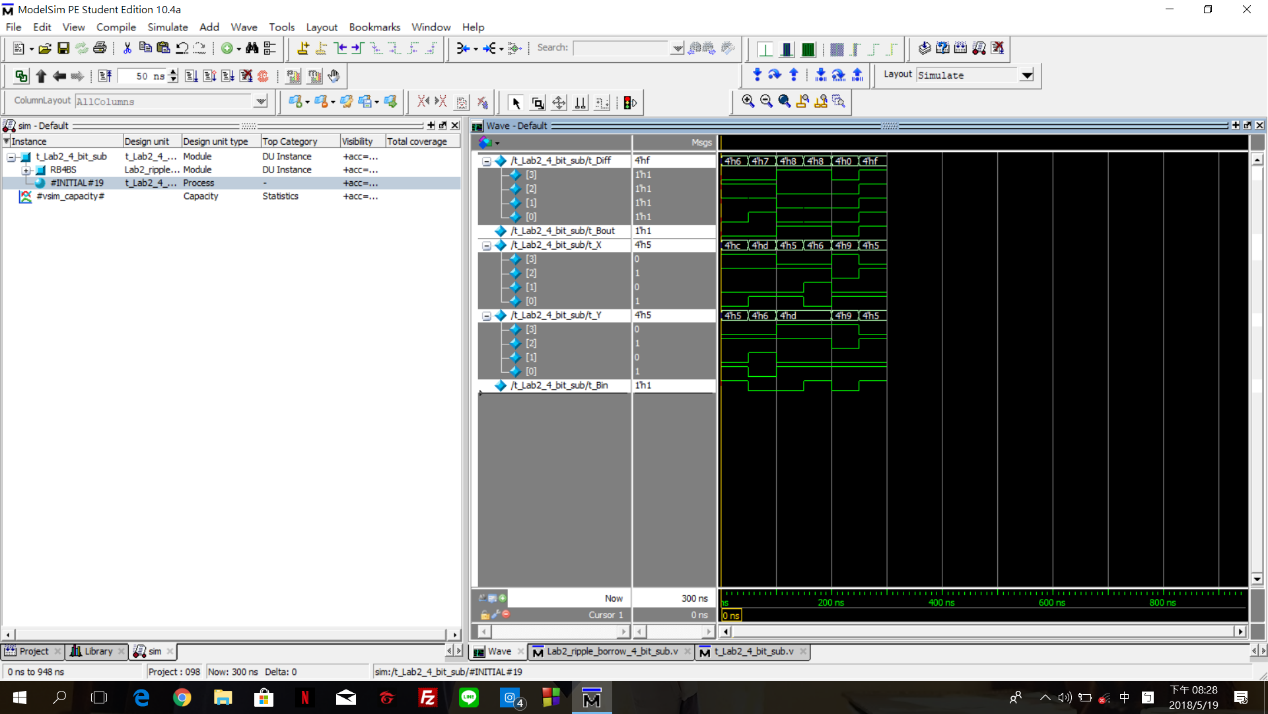
因為波形與我求得的值相同，且程式是依照電路圖設計，因此應該是對的。

(3)

參考講義的4-bit ripple-carry adder及full-substractor的觀念可推出4-bit ripple-borrow subtractor。



2A(c)ii (4-bit RBS)之模擬結果波形圖



X 1100 1101 0101 0110 1001 0101

Y 0101 0110 1101 1101 1001 0101

Bin - 1 - 0 - 0 - 1 - 0 - 1

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

Diff[3][2][1][0] 0110 0111 1000 1000 0000 1111

Bout 0 0 1 1 0 1

結果與波形相同，故答案是對的。

(4)

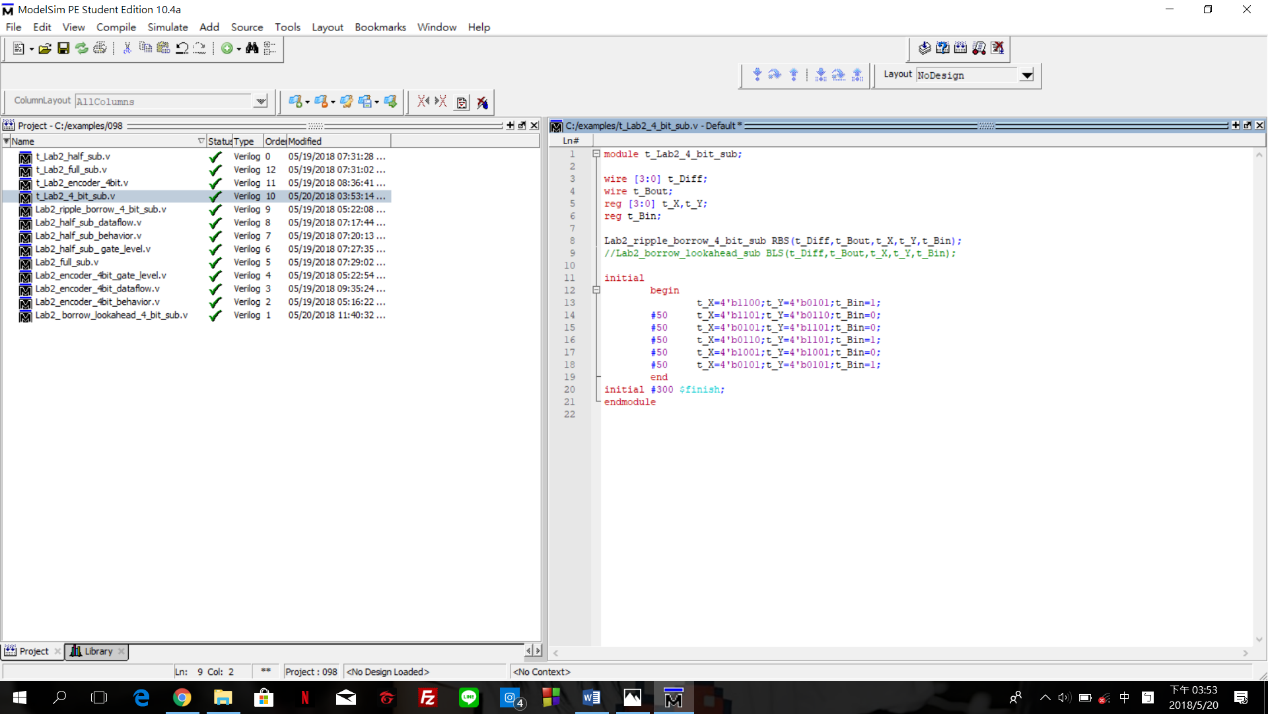
參考講義的Carry Lookahead Adder (CLS)及其觀念，再思考P跟G的電路，配合講義的4-bit ripple-carry adder及full-substractor的觀念推出的4-bit ripple-borrow subtractor。即可推出4-bit borrow lookahead subtractor (BLS)。

P[i]=X[i]xorY[i] G[i]= (!X[i])orY[i]

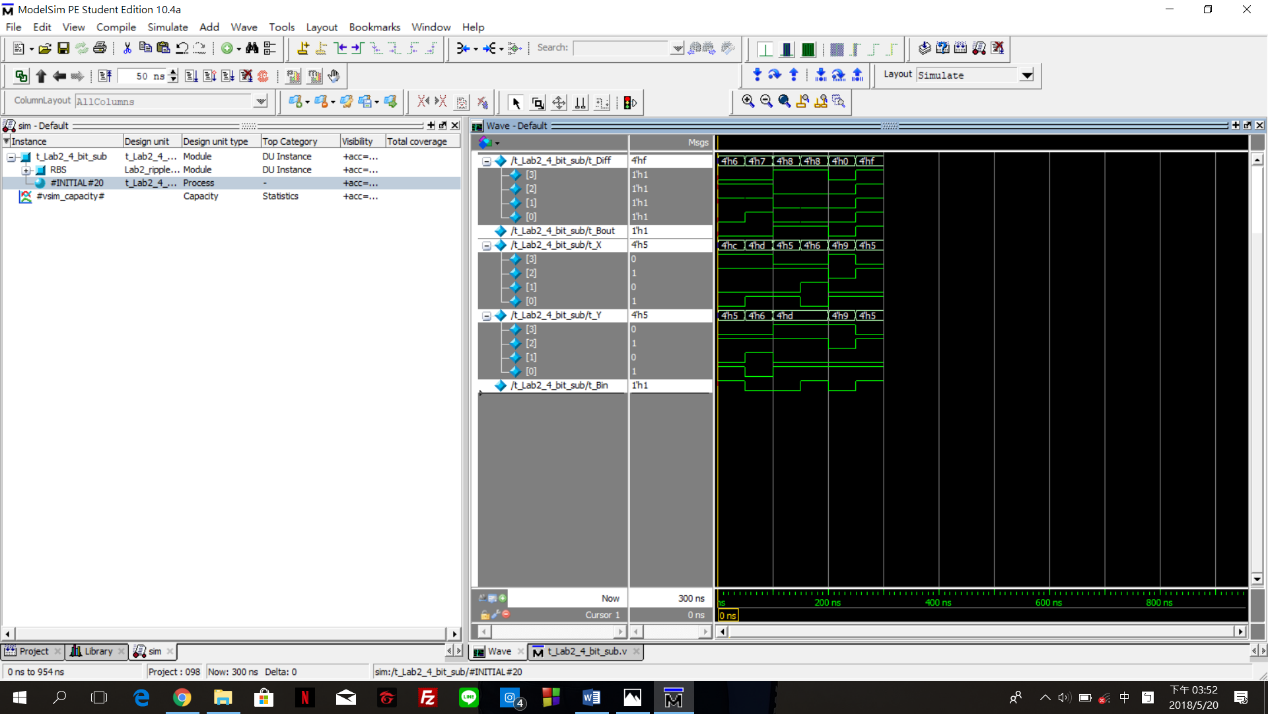
D[i]= P[i]xorB[i] B[i+1]=G[i]or((!P[i])andB[i])

找出Pi、Gi、Bi、Di的布林代數式後，即可推出BLS。

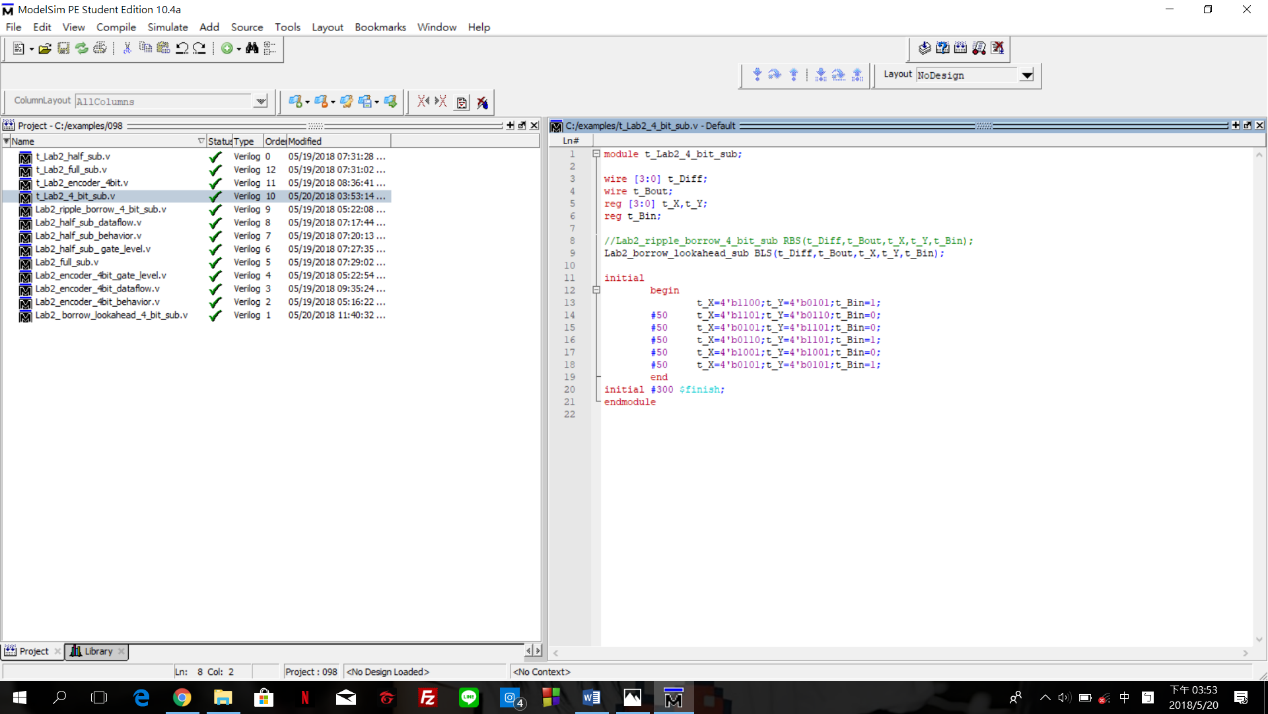
RBS

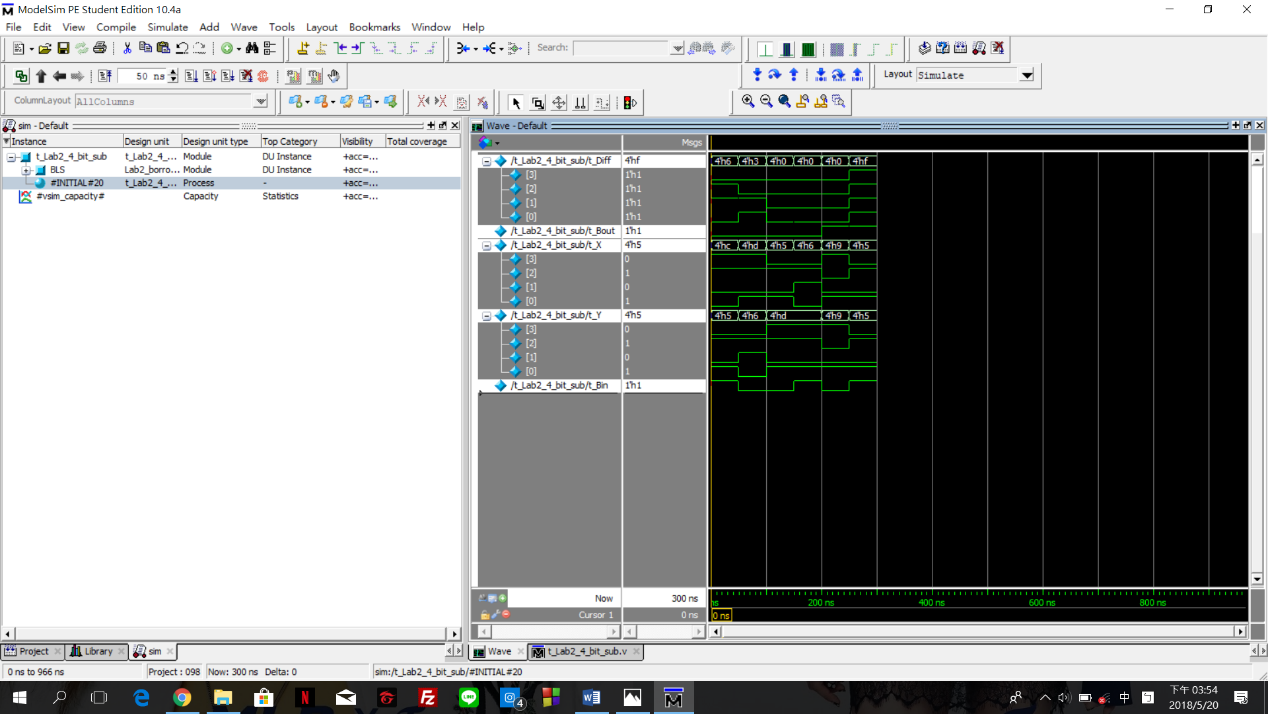


2A(c)ii (4-bit RBS)之模擬結果波形圖



BLS

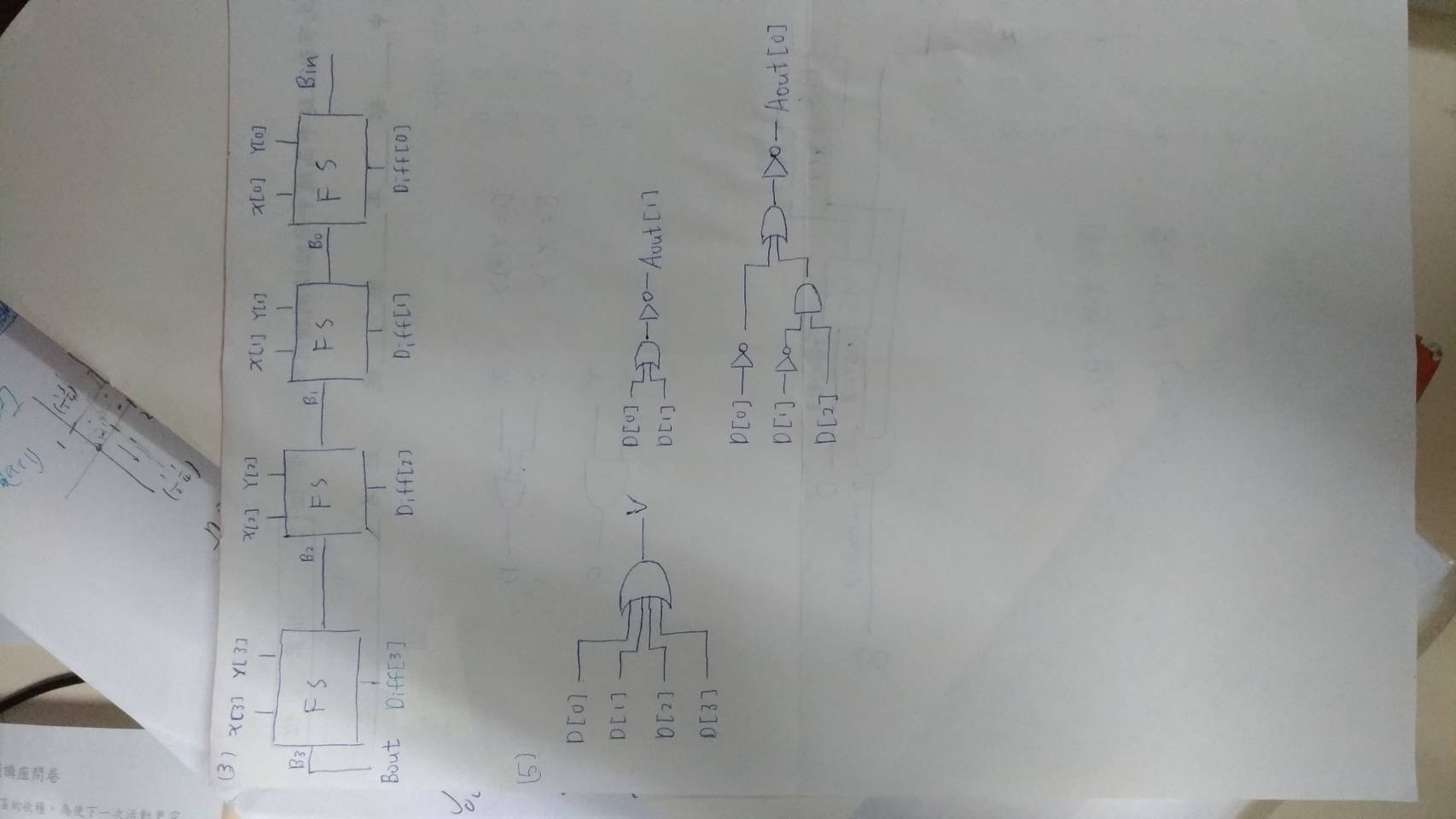


2A(d)ii (4-bit BLS)之模擬結果波形圖

因為BLS與RBS的波形皆相同，而(3)中說明RBS是正確的，因此BLS也是正確的。

(5)

參考講義的Priority Encoder並由Truth table利用k-map或透過觀察推倒得出布林代數式，或將講義上的Priority Encoder的output改成complement且改變input的Priority order即可推出4-input priority encoder。

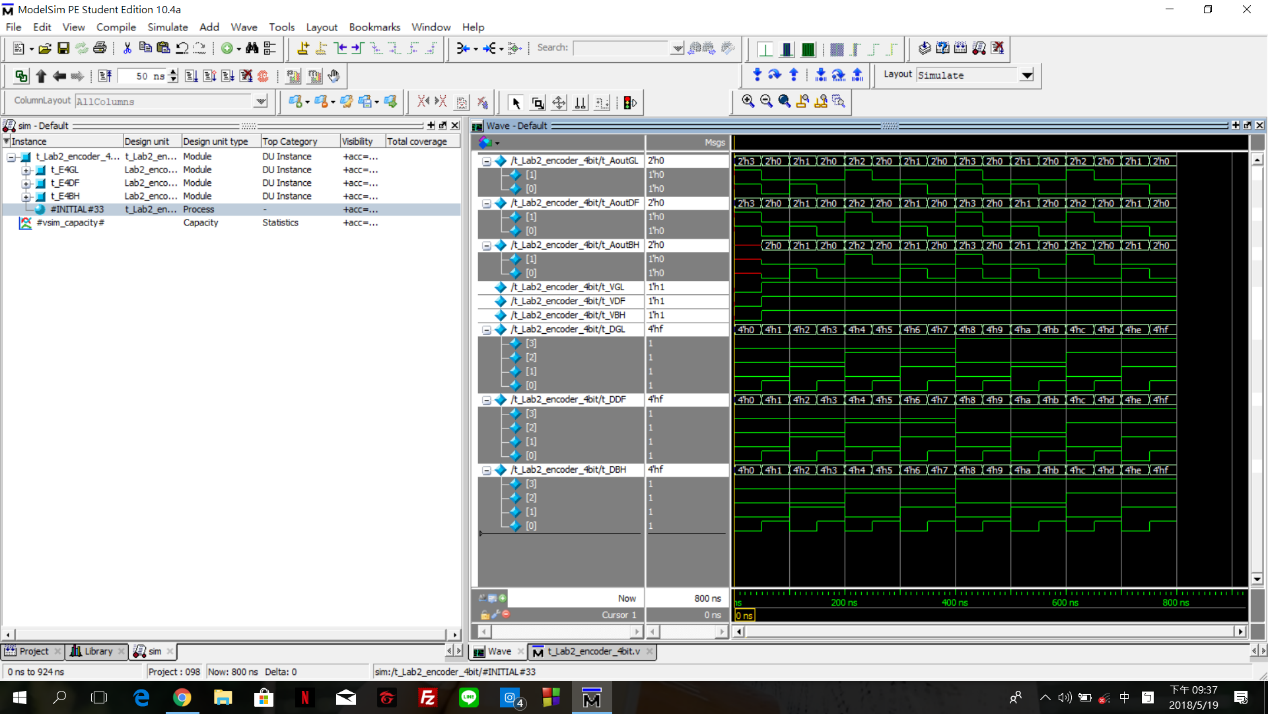


V=D[0]orD[1]orD[2]orD[3]

Aout[1]=!(D[0]orD[1])

Aout[0]=!(D[0]or((!D[1])andD[2]))

2B. iv (四輸入優先編碼器)之模擬結果



波形三個方法(Gate level ， dataflow ， behavior)皆一致，與我畫出的電路圖原理相同，且答案與代入布林代數式同，故答案應該正確。

(6)心得與感想、及遭遇到的問題或困難:

這次的作業邏輯比上次難，需要注意的語法也比上次多，必須一直猜想編譯邏輯才有可能compile成功，而且這次的作業規模更大也更複雜比較有寫程式的感覺，但有太多東西要靠自己摸了，希望提示還能多一點，因為有講義，其實有些只是加法改成減法就好，而一些其實沒得參考就比較難。這次自己摸程式的時間真的很多，尤其是因為有多bits和behavior，而且同個變數套用多個函式有問題，只能改用比較瑣碎的方法，真的花很久時間才debug完，這次真的有難，所以寫的不順。