10910EECS207001 Logic Design Lab

LAB 1：Gate-Level Verilog Report

TEAM 5

組長：108062211 黃子瑋

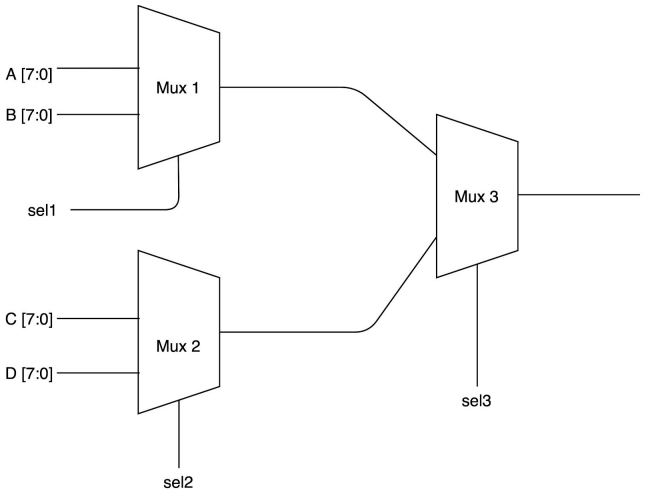
組員：108062213 顏浩昀

2020.09.24

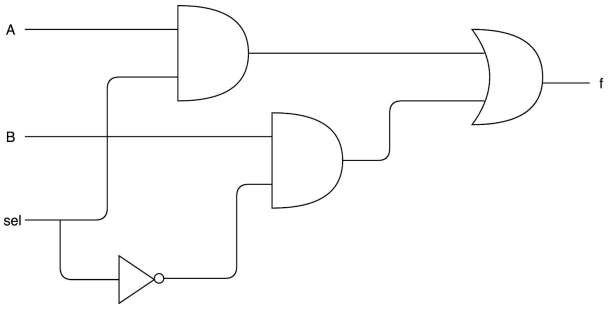
MUX_8bits

I、Logic Design Diagram

(i)完整電路設計圖



(ii)單一mux電路設計圖



II、Explaination

```
input [8-1:0] a, b;
input sel;
output [8-1:0] f;

wire sel0;
wire [8-1:0] cha, chb;
not not1(sel0, sel);

and and1[7:0](cha, a, sel );
and and2[7:0](chb, b, sel0 );

or or1[7:0](f, cha, chb);
```

左圖為單一mux module的設計，當中的兩個AND Gate是選擇輸出為a,b的重要因素，若sel == 1，會選擇a作為輸出；反之則選擇b作為輸出。由於各Logic Gate能判別的bit數為1bit，我們使用gate array的形式製作Mux。

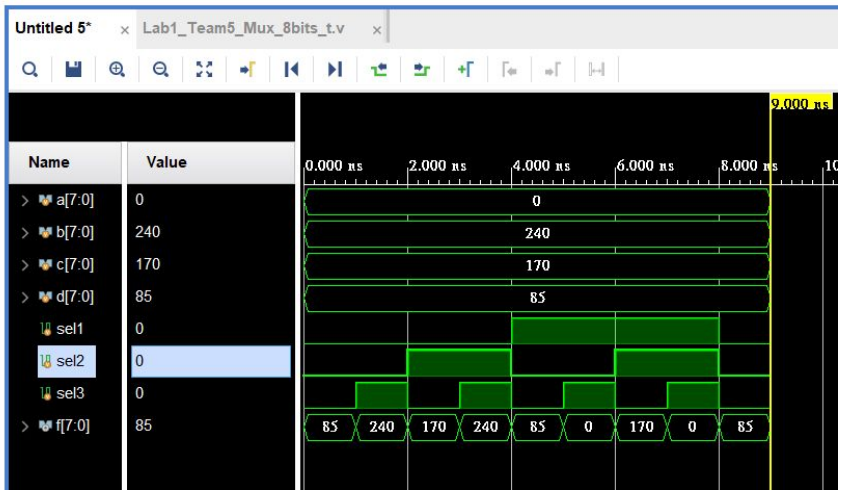
右圖是依照本題題意所寫成，利用已經設計好的Mux，先透過m1選擇a or b，再透過m2選擇c or d，最後是藉由m3選擇使用m1 or m2的output當作最後的f(output)。

```
Mux m1(
    .a(a),
    .b(b),
    .sel(sel1),
    .f(ab)
);
Mux m2(
    .a(c),
    .b(d),
    .sel(sel2),
    .f(cd)
);
Mux m3(
    .a(ab),
    .b(cd),
    .sel(sel3),
    .f(f)
);
```

III、Testbench

```
initial begin
  repeat (2**3) begin
    #1{sel1,sel2,sel3} = {sel1,sel2,sel3} + 1'b1;
  end

  #1 $finish;
end
```

測試時以預設a,b,c,d四數數值，將sel1、sel2、sel3所有可能皆測試一遍，以確認所有排列組合皆無問題。

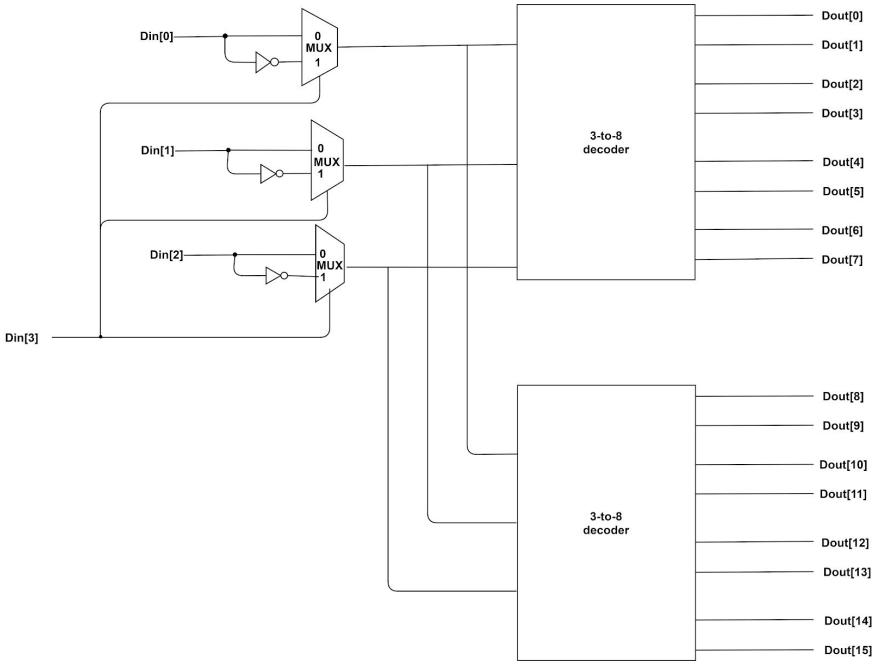


左圖為波形圖結果

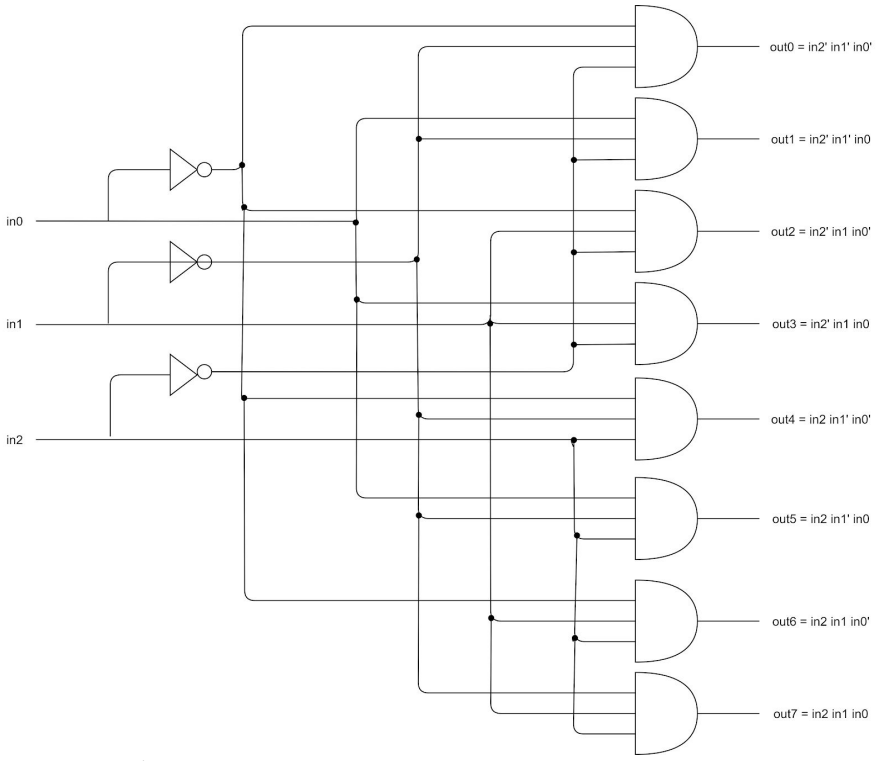
4x16 decoder

I、Logic Design Diagram

(i) 完整電路設計圖



(ii) 3x8 decoder電路設計圖



(iii) MUX電路設計圖

見MUX_8bits附圖。

II、Explaination

Input Din[3:0]	Output Dout[15:0]	Input Din[3:0]	Output Dout[15:0]
1111	0000_0001_0000_0001	0111	1000_0000_1000_0000
1110	0000_0010_0000_0010	0110	0100_0000_0100_0000
1101	0000_0100_0000_0100	0101	0010_0000_0010_0000
1100	0000_1000_0000_1000	0100	0001_0000_0001_0000
1011	0001_0000_0001_0000	0011	0000_1000_0000_1000
1010	0010_0000_0010_0000	0010	0000_0100_0000_0100
1001	0100_0000_0100_0000	0001	0000_0010_0000_0010
1000	1000_0000_1000_0000	0000	0000_0001_0000_0001

我們發現在對應表中16種input中每兩個會有一組相同的output，像是1111以及0000，而output中的前8個bit和後面8個也一定相同，於是我們開始用3X8 decoder去做設計。由於發現擁有相同output的input每一個bit都是01相反，於是我們使用input的第四個bit做判斷，如果為1，則將其餘的每一個bit反轉後再導入3X8 decoder，那對於3X8 decoder來說，就只會有8種input，而3X8 decoder內部則是將input的3bit訊號轉換為one hot的8bit輸出，而我們只要將這個8bit訊號重複兩遍，即可得到題目要求的16bit訊號輸出。

```
not not0 (d0not,din[0]);
not not1 (d1not,din[1]);
not not2 (d2not,din[2]);

wire  [3-1:0] d1input;

Mux_1bit in2(d2not,din[2],din[3],d1input[2]);
Mux_1bit in1(d1not,din[1],din[3],d1input[1]);
Mux_1bit in0(d0not,din[0],din[3],d1input[0]);

Decoder_3X8 d1( d1input , dout[15:8] );
Decoder_3X8 d2( d1input , dout[7:0] );
```

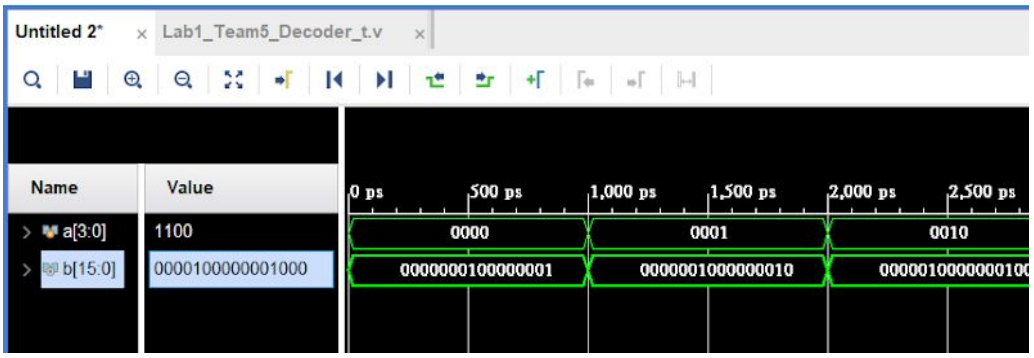
例如:1111的第[3]個bit為1，那我們就將後面的三個bit做翻轉得到000再將其輸入給3X8 decoder，3X8 decoder輸出得到0000_0001，而我們就將這個訊號重複輸出兩遍，即得題意所求。

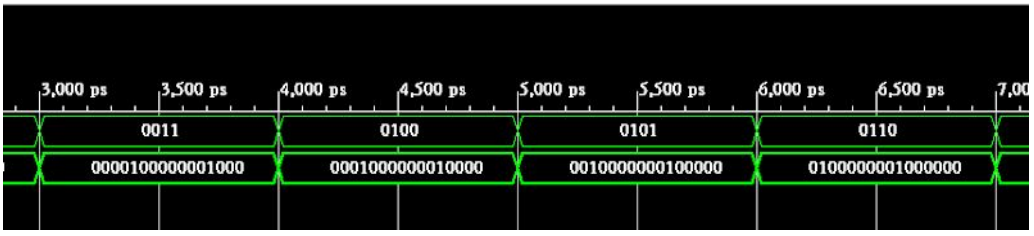
左圖中，MUX前兩個wire為input，第三個為control value，第四個為output。

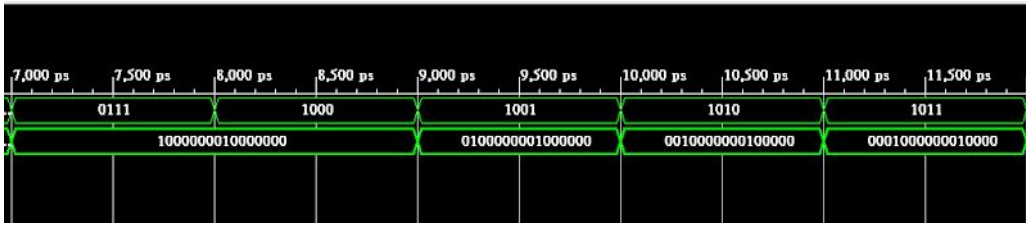
III、Testbench

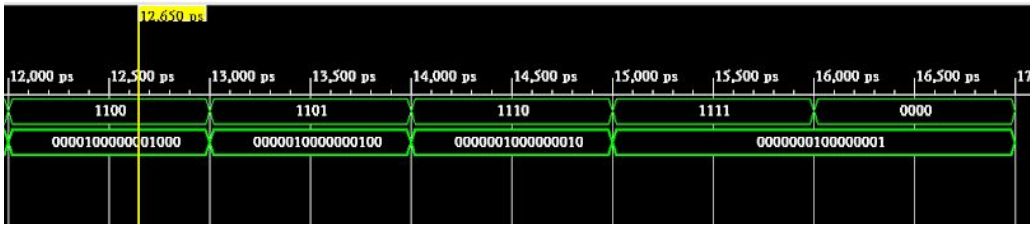
```
initial begin
  repeat (2 ** 4) begin
    #1
    a = a + 3'b1;
  end
  #1 $finish;
end
```

依照題目所給之表格，窮舉所有數值。左圖為測試用部分code，下圖皆為測試波形圖。





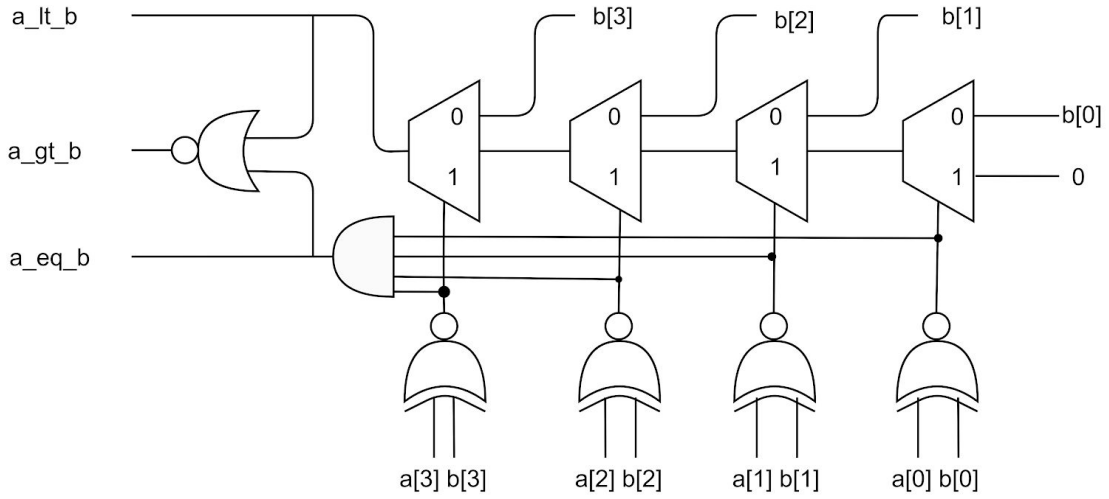




4-bit comparator

I、Logic Design Diagram

(i)完整電路設計圖



(ii)XNOR Gate電路設計圖

同Basic question 2附圖

(iii)MUX電路設計圖

詳見Mux_8bits附圖(ii)

II、Explaination

a_lt_b的訊號是直接將兩個input的bit由高位數到低位數依序比較，若是相比的bit不相等XNOR gate就會輸出0，則MUX就會選擇當下b的bit做輸出，而當位數b!=a、b==1時即為a<b，反之亦然，因此正好可以用來當作a_lt_b訊號的表示方法。

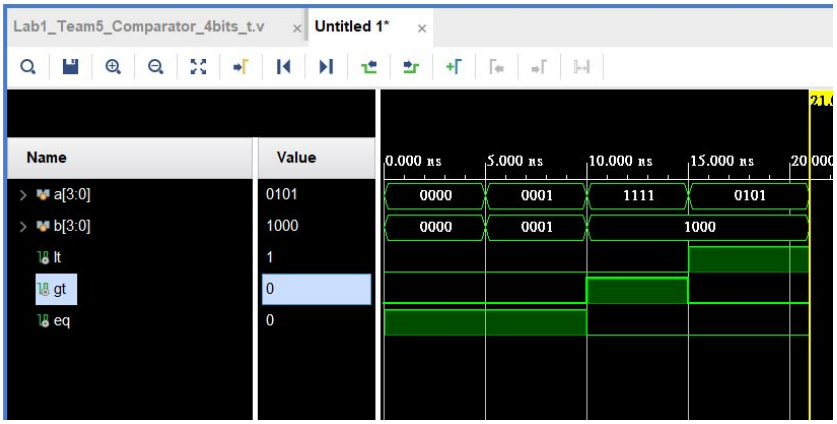
```
XNOR_1 x0(a[0],b[0],x0o);
XNOR_1 x1(a[1],b[1],x1o);
XNOR_1 x2(a[2],b[2],x2o);
XNOR_1 x3(a[3],b[3],x3o);

Mux_1bit m0(0,b[0],x0o,m0o);
Mux_1bit m1(m0o,b[1],x1o,m1o);
Mux_1bit m2(m1o,b[2],x2o,m2o);
Mux_1bit m3(m2o,b[3],x3o,a_lt_b);

and and0 (a_eq_b,x0o,x1o,x2o,x3o);

or or0 (temp,a_lt_b,a_eq_b);
not not0 (a_gt_b,temp);
```

若是a、b在當前的bit相等，則MUX會將訊號交給下一位數的MUX去比較下一位數的bit，以此類推，那到了最後一個bit時若是ab依然相等，那a就不小於b，於是a_lt_b訊號就輸出0，那4個bit都相等的狀況下，4個XNOR gate就會皆輸出1那我們就將這四個訊號and起來，作為a_eq_b的輸出，最後，如果a_lt_b、a_eq_b皆為0則代表a_gt_b，於是將這兩個gate以NOR gate連接起來作為a_gt_b的訊號來源。

III、Testbench

```
initial begin
    #5  // a == b
    a = 1;
    b = 1;
    #5  // a > b
    a = 2**4-1;
    b = 2**3;
    #5 // a < b
    a = 2**2+1;
    b = 2**3 ;
    #5
    #1 $finish;
end
```

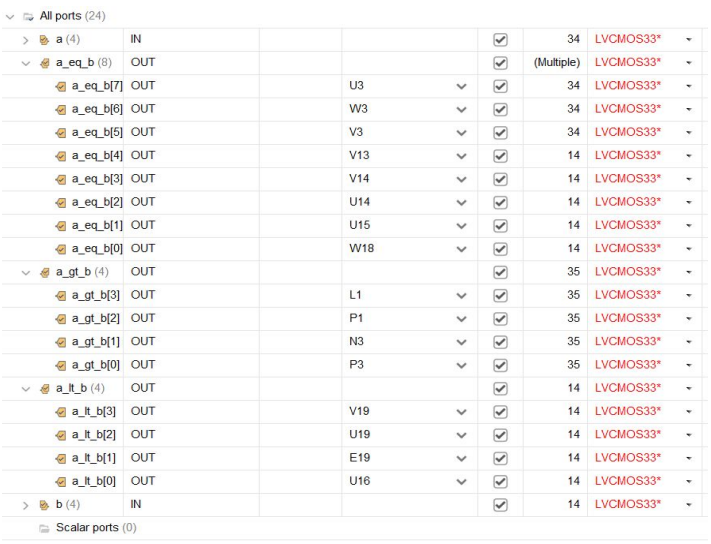
測試a==b、a<b、a>b之情況，下圖為測試波形圖。



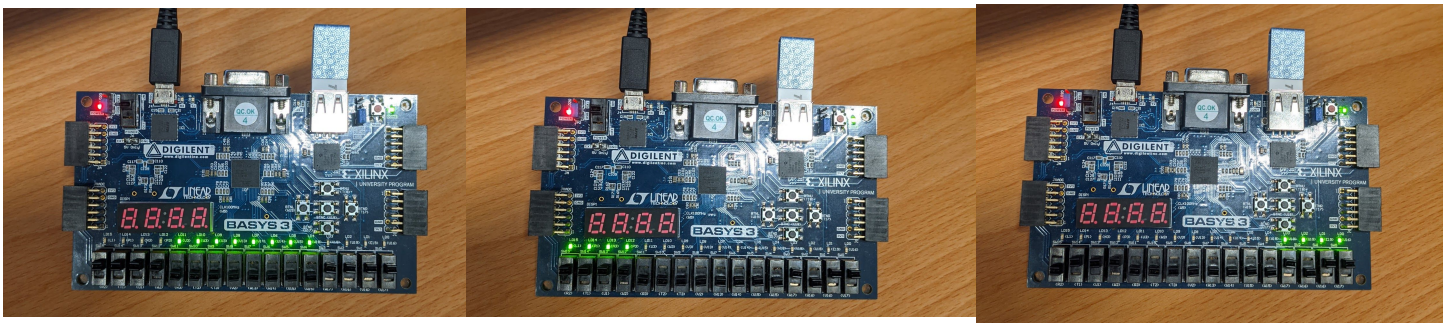
IV、FPGA實作

架構同上，唯將output訊號分別轉為4、8、4個bit，使用fin-out實作，實作方法如左圖。xdc檔案實作方法如下圖。

```
not not1(nlt,lt);
not not2(ngt,gt);
not not3(neq,eq);
not not4(
        a_lt_b[3],
        a_lt_b[2],
        a_lt_b[1],
        a_lt_b[0],
        nlt  );
not not5(
        a_gt_b[3],
        a_gt_b[2],
        a_gt_b[1],
        a_gt_b[0],
        ngt );
not not6(a_eq_b[7],
        a_eq_b[6],
        a_eq_b[5],
        a_eq_b[4],
        a_eq_b[3],
        a_eq_b[2],
        a_eq_b[1],
        a_eq_b[0],
        neq  );
```



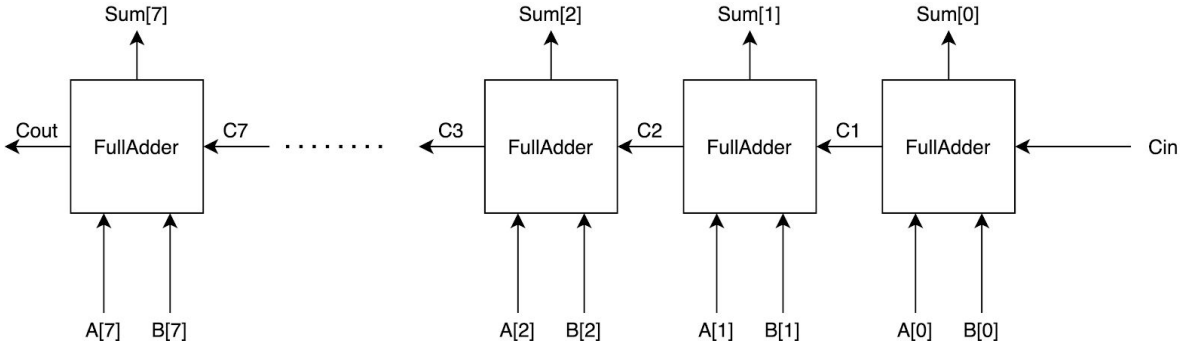
下圖為實作範例



Ripple Carry Adder

I、Logic Design Diagram

(i)完整電路設計圖



(ii)FullAdder電路設計圖

同Basic question 2附圖

II、Explaination

我們在實作Ripple Carry Adder(以下簡稱RCA)使用的是多個Full Adder(以下簡稱FA)。

首先解釋FA的製作方式，每一個FA都有著三個input，分別是a,b及cin，另外有著cout和sum兩個output。第一步要先將a和b透過XNOR Gate產出x1out。接著將x1out與cin接入第二個XNOR Gate，至此透過兩個XNOR Gate可以得到output中的sum。最後，我們使用Mux來做output的cout，在這個Mux中，sel訊號是x1out，a和cin則分別代表Mux1和0的。

在製作完FA的module後，則可以將多個FA相接形成RCA。題目敘述的cin為編號FA1的cin input，同時A[0],B[0]（A,B兩數的第0個bit）分別代表FA1的a,b input，透過FA1我們會得到output sum[0]及c1(cout)，而這個c1同時也是下一個FA(編號FA2)的cin input。同上，FA2的output c2(cout)會成為FA3的cin，依此類推我們連接剩餘的FA成為一個RCA，而最後一個FA(編號FA8)的output cout則為題目所需的cout。

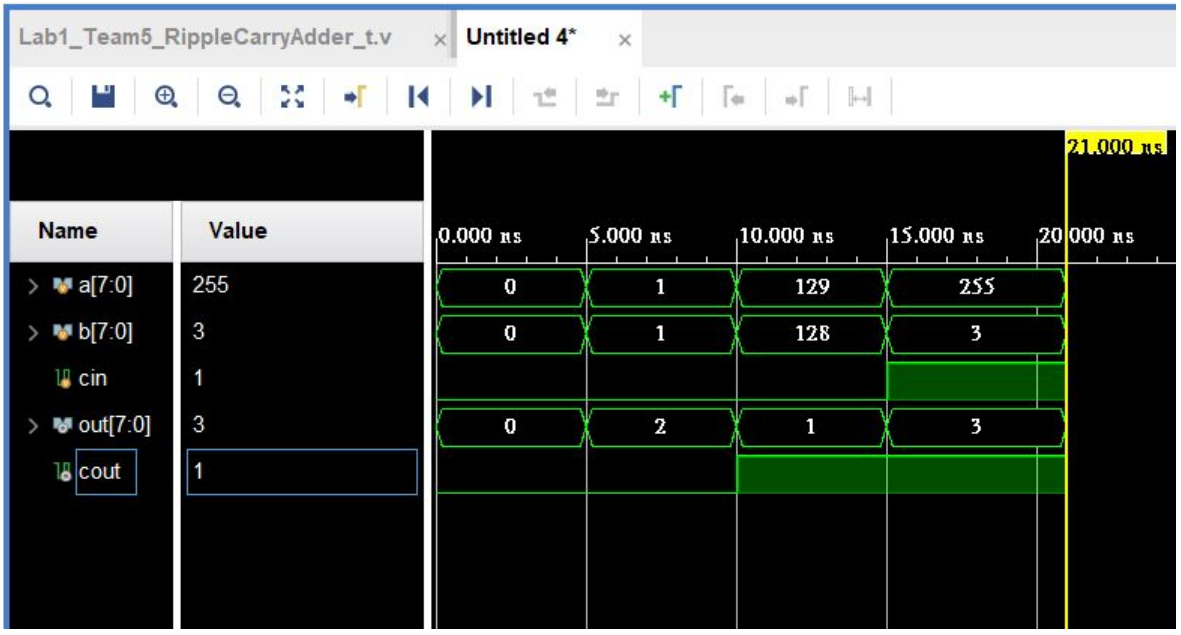
III、Testbench

```
initial begin

    #5  // cin == 0 cout == 0
    a = 1;
    b = 1;
    cin = 0;
    #5  // cin==0 cout==1
    a = 2**7+1;
    b = 2**7;
    cin = 0;
    #5 // cin == 1 cout==1 since cin
    a = 2**8-1;
    b = 3 ;
    cin = 1;
    #5


    #1 $finish;
end
```

我們以特定數據作為測資。第一組測資測試當cin==0，且a, b數字較小，cout不會有進位的情形。第二組測資為cin==0，而a,b相加會讓會有進位使cout==1。第三筆測資為cin==1且a,b,cin相加會進位使cout==1。

第一筆測資希望測得RCA是否能做基本相加，第二筆測資希望測得RCA在相加的進位是否正常運作，第三筆測資希望測得RCA在有cin的情況下是否能正確運作並正確進位。

下圖為測試波形圖。



What are we learned from this Lab

本次Lab使用Vivado進行波形圖模擬以及FPGA實作，熟悉Vivado頁面與使用方式是此次Lab最大收穫。其次，verilog的基礎語法以及Logic Gate的設計雖然在邏輯設計課程已經有過些許接觸，但此次Lab為我們第一次親自設計完整的電路圖與從頭到尾使用verilog寫出自己的module與testbench，更利用Vivado與FPGA實作出comparator。

我們在此次Lab中也遭遇了些許困難：

一、

執行模擬時有少次發生error，後來發現是由於testbench撰寫時有疏忽，所以導致執行模擬時出現錯誤。

二、

FPGA板子在vivado介面抓不到，反覆測試花費不少時間，後來在TA時間向助教求助，經由四個助教反覆確認，應為筆電硬體問題，需要更換電腦，但由於小組組員只有一台windows筆電，因此我們轉向使用將bit檔燒錄至USB硬碟的做法測試，後來在網路尋找方法後成功實作，也因此避免額外購買筆電的費用。

以下是我們實作以USB作為bit_stream來源的方法:

1、一樣在vivado內生成bit_stream。
2、在project資料夾->.runs->impl_1中找到.bit檔案
3、將.bit檔複製至**乾淨USB**根目錄內。
4、將USB插上FPGA版，並將右側jumper設置為USB。

此時，USB左方BUSY燈號會開始閃動，表示FPGA版正在搜尋USB內的檔案，(USB內如有其他檔案則會耗時許久)，待燈號停止閃爍，即可正常使用。

Cooperation

108062213 顏浩昀:

負責8bit_MUX、RCA設計，XNOR、1bit_MUX元件設計，繪製邏輯圖，共同撰寫報告。

108062211 黃子瑋:

負責4bits_comparator、decoder設計，執行vivado模擬，FPGA實作，共同撰寫報告。

本次coding部分，由兩個組員使用vs code插件 live share共同完成，他是一個可以讓多位使用者即時共同編輯同一份程式碼的工具，檔案是存放在其中一位會議主持人的本地端，可以即時儲存，有些類似google文件共同編輯的方式，不同之處在於，vs code本身的文字編輯器對於coding十分方便，以及可以即時交給編譯器做編譯，對於多人實作十分有幫助，特別是我們只有一位組員可以安裝vivado實測code是否正確，藉由兩人實時共同編輯一個人的本地文件，再交由使用windows作業系統的同學測試code是否正確，可以省去中間檔案傳輸的麻煩，是個十分好用的功能。