Lab 2 report

組員：110062221 李品萱

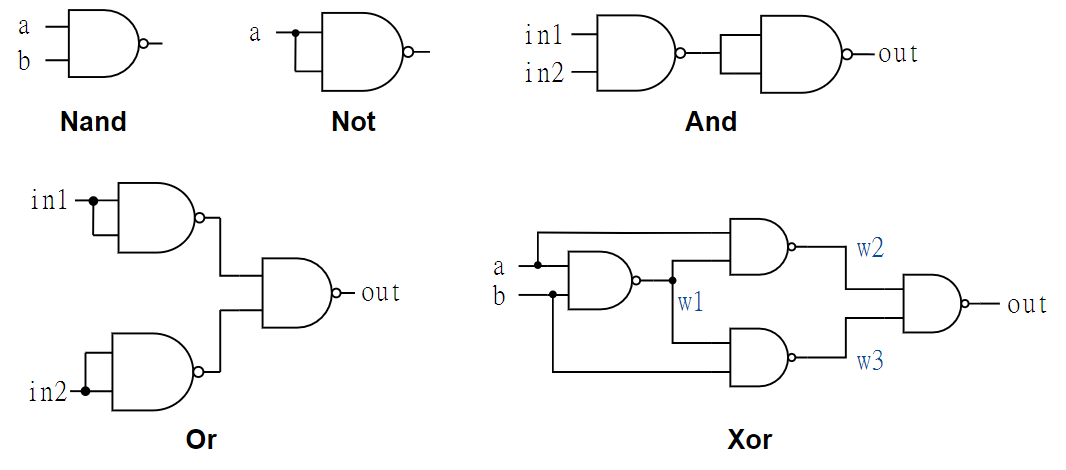
110062213 唐翊雯

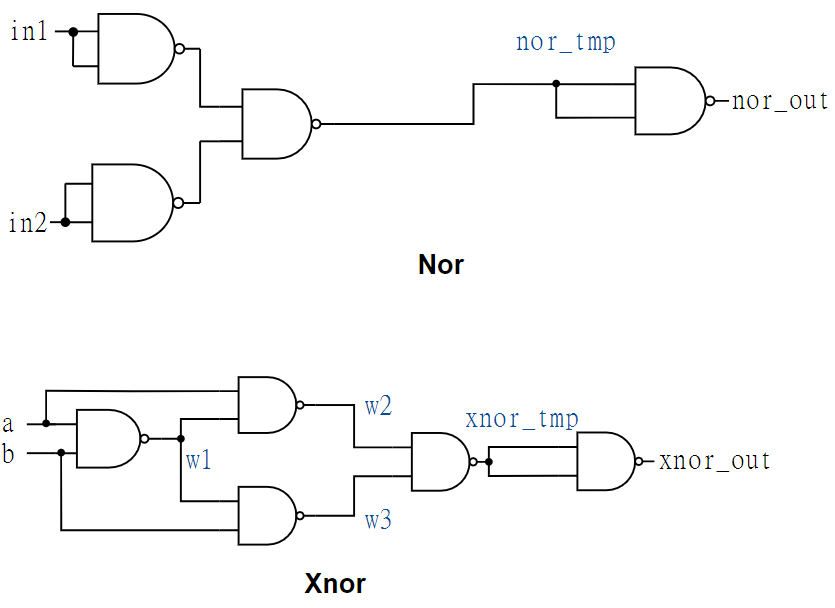
1. **(Gate Level) 8-bit ripple carry adder (RCA)**

首先，這題我們使用了Basic question 1中以NAND Gate實作的basic logic gates，以及在Basic question 3中以NAND Gate實作的Full Adder。

* Basic question 1：

以下為用NAND Gate實作的Basic Logic Gates:

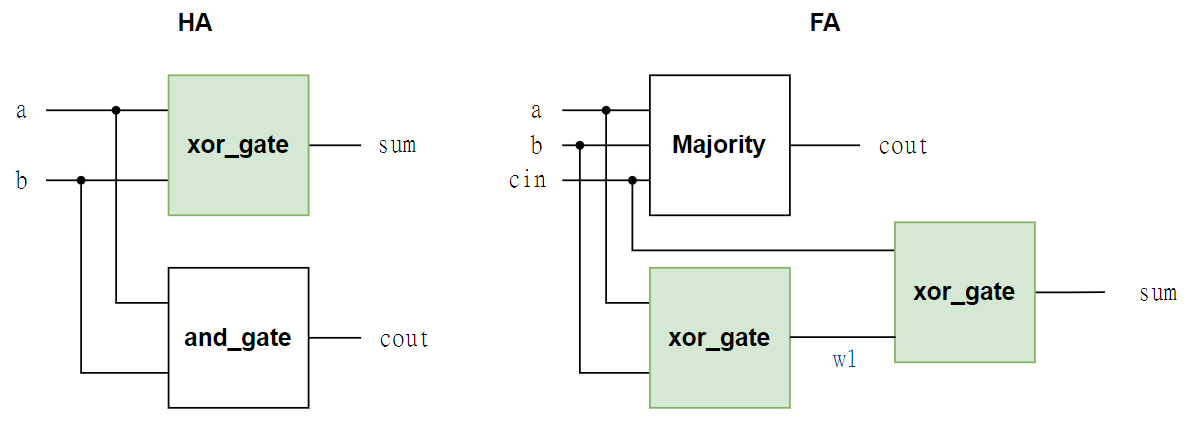


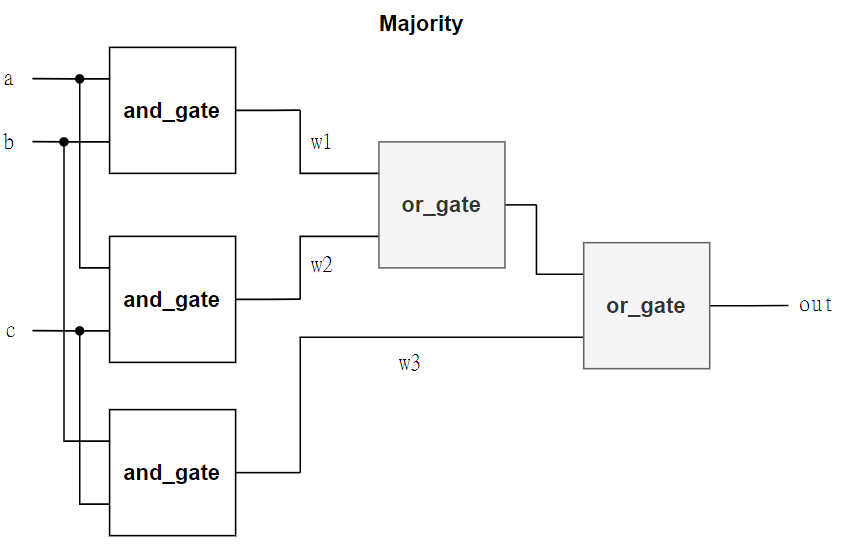


Select:

* Basic question 3：

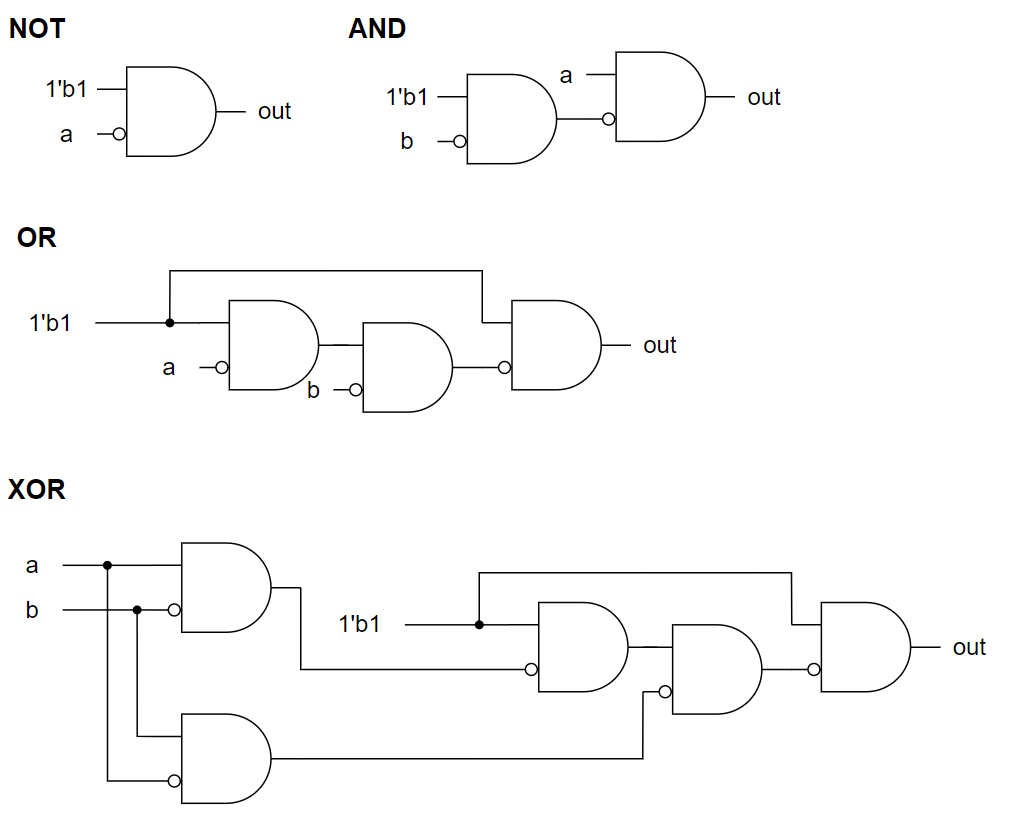
Half Adder、Full Adder的Circuit分別如下：

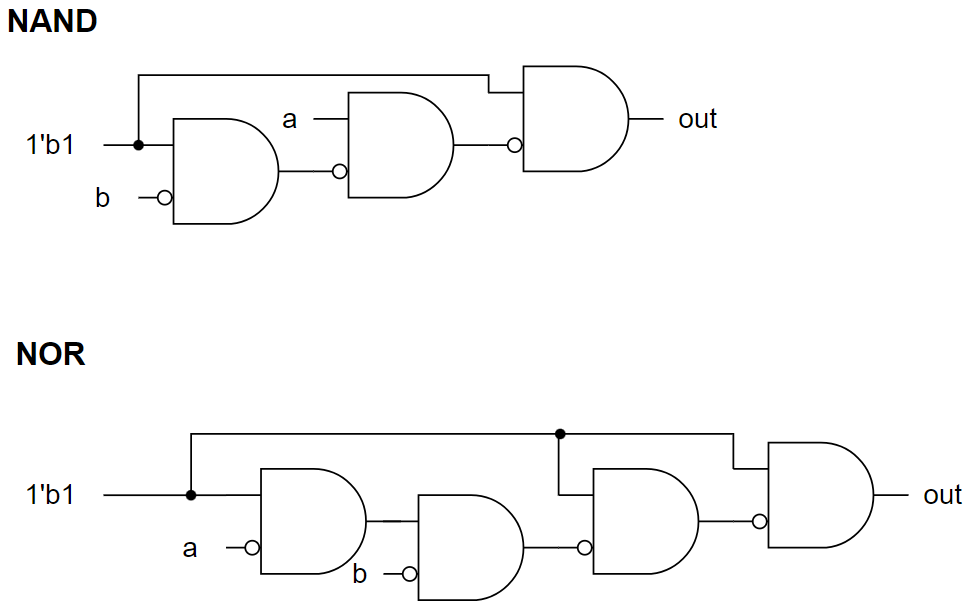




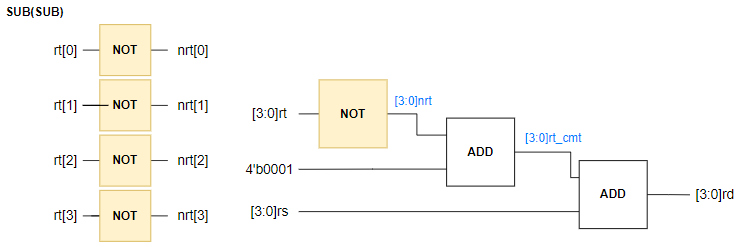
1. **(Gate Level) Decode and execute**

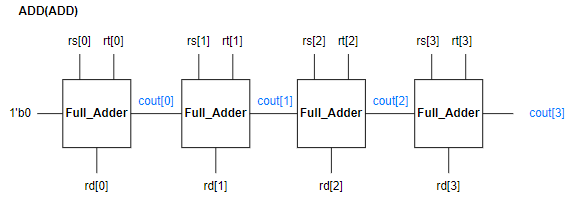
這題首先要使用給定的universal gate，實作出其他所有的basic logic gate，它們的circuit分別如下：

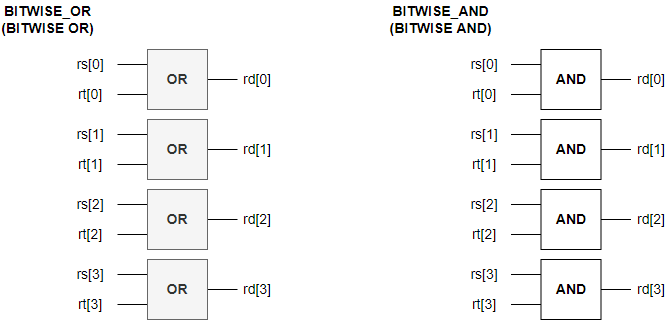


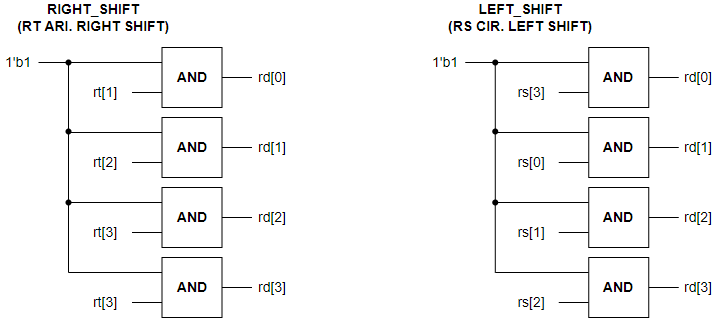


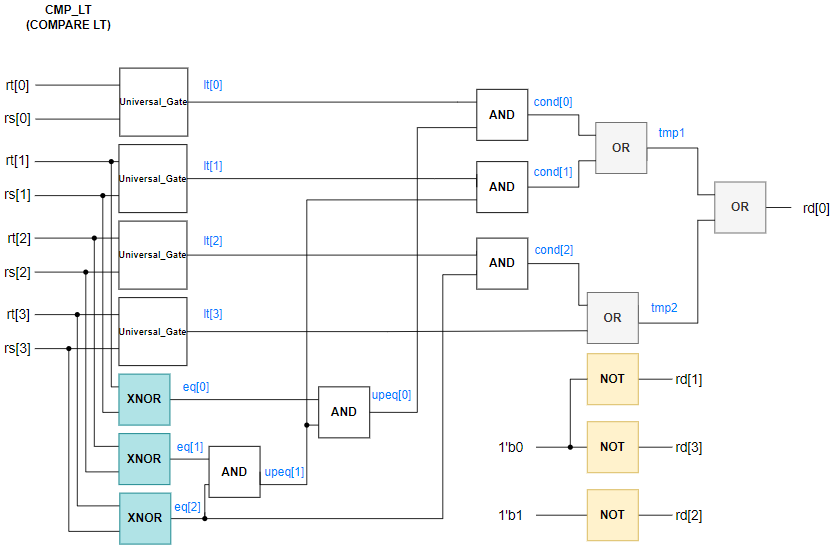
我們基於這些basic logic gate，實作了題目所要求的各個function，它們的circuit分別如下：

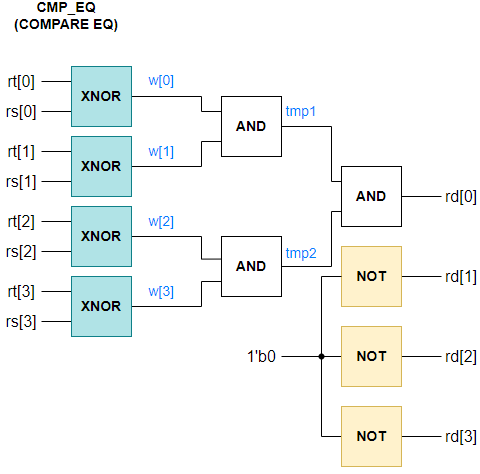












SUB 的實作使用到了二補數：首先，這個function等同於。因此我們先對rt取負號，即計算它的二補數，接著再利用ADD這個module將rs與rt相加，得到我們要的輸出。計算二補數的方法是將所有的bit都倒過來再加一，因此我們先使用之前實作的NOT gate，對rt所有的bit都取not，再將它與4’b0001，一樣利用ADD這個module將它們相加起來。

ADD這個function的實作則是使用到了4個Full Adder。基本上，每一個Full Adder的實作的方式與basic question 3的實作相同，只是把所有basic logic gates的實作從使用NAND改為使用題目規定的universal gate。如上圖把4個Full Adder接在一起，便可以實作出這個function。

BITWISE OR的實作相對單純：將rs與rt的每一個bit分別作OR，便是這個function要的輸出。BITWISE AND的實作也很相似：將rs與rt的每一個bit分別作AND即可。

RT ARI. RIGHT SHIFT的部分，題目已經說明rd的每一個bit分別要是rt哪一個bit的值，這邊我們的做法是對這些值與1’b1取AND，輸出即為所求。RS CIR. LEFT SHIFT也很相似，一樣，題目已經說明rd的每一個bit分別要是rs哪一個bit的值，對這些值與1’b1取AND，輸出即為所求。

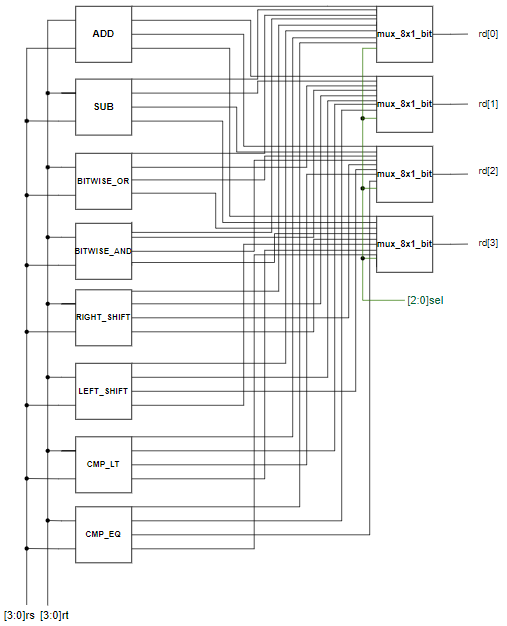
COMPARE LT的部分，我們實作的想法如下：比較兩個二進位數字的大小，首先要比較最高位的bit，若不同則該bit比較出來的大小關係即為兩數的大小關係。若相同，則繼續往下一個較低位的bit比較，依此類推。因此若要符合，必須符合下列條件的其中一項：

因此，我們分別用logic gate得到上述關係式的結果，再將所有結果取OR，output即為rd[0]。 rd[3:1] 的部分，題目已經指定每個bit的值，這邊我們直接使用三個NOT gate：輸入分別放題目要求的值的相反；輸出分別放rd[0]、rd[1]、rd[2]，便能達到題目要求。

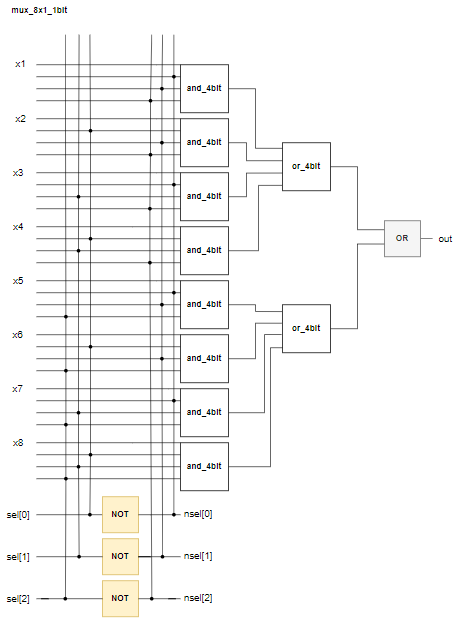
COMPARE EQ的實作，我們的想法如下：比較兩個二進位數字的是否相等，要比較兩數的每一個bit是否都相同。也就是說，若，下列的值皆須為true：

判斷相等的部分我們使用XNOR實作，若兩bit作XNOR的值為true，即代表兩bit相等。最後把上述4個值取AND，即為rd[0] 的值。rd[3:1] 的部分與COMPARE LT的部分相同，使用三個NOT gate給rd[3:1] 題目所要求的值。

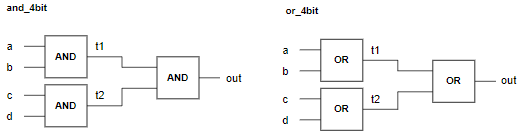
最後，根據sel的值，我們要給出相對應的output。在得到各個function的output之後，我們使用了4個8 to 1的MUX (1 bit)，分別得到output的每個bit的值，circuit如下：



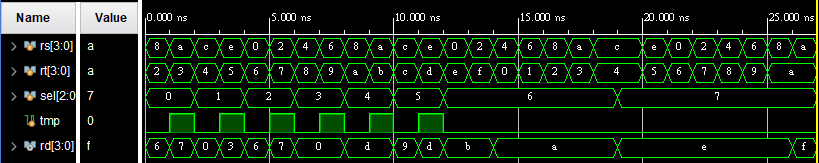
而8 to 1的MUX (1 bit)，其circuit如下：

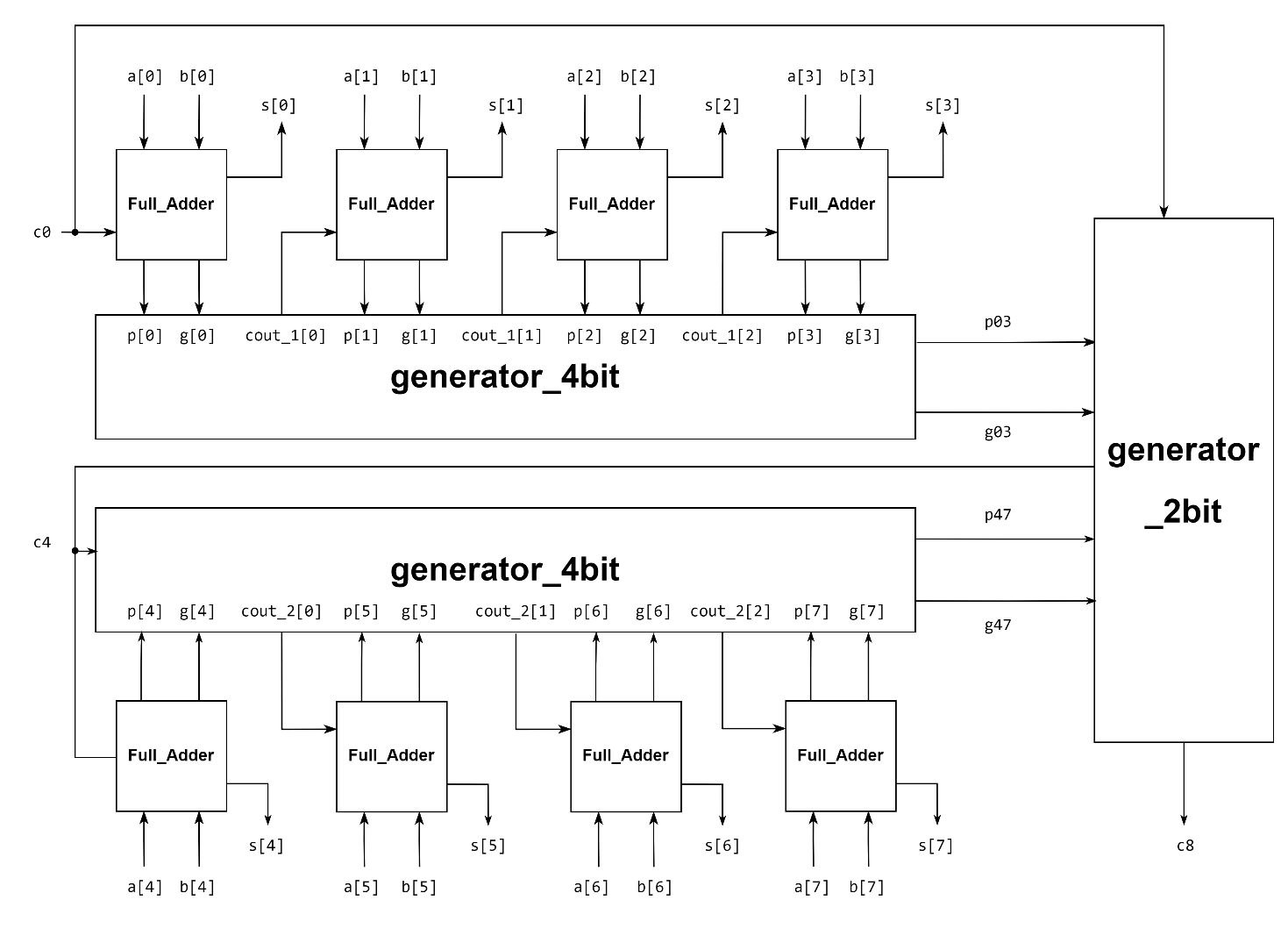


其中使用到了4 bit的AND以及4 bit的OR，circuit分別如下：

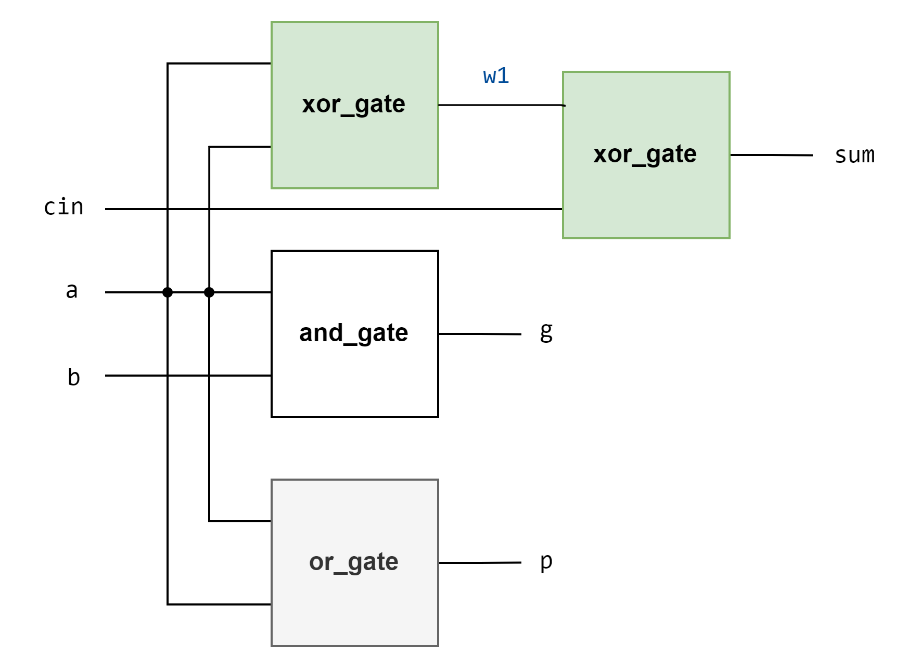


Testbench的部分，對於前6個function，我們讓rs的值每次加2、讓rt的值每次加1，並賦予不同的初始值下去確認每個function的output是否正確。而對COMPARE LT的部分，我們多跑了幾次，以確認在、以及的情況下，output 都會是正確的。COMPARE EQ的部分，我們也特別assign了rs與rt的值，以確保在以及的情況下，output都是正確的。

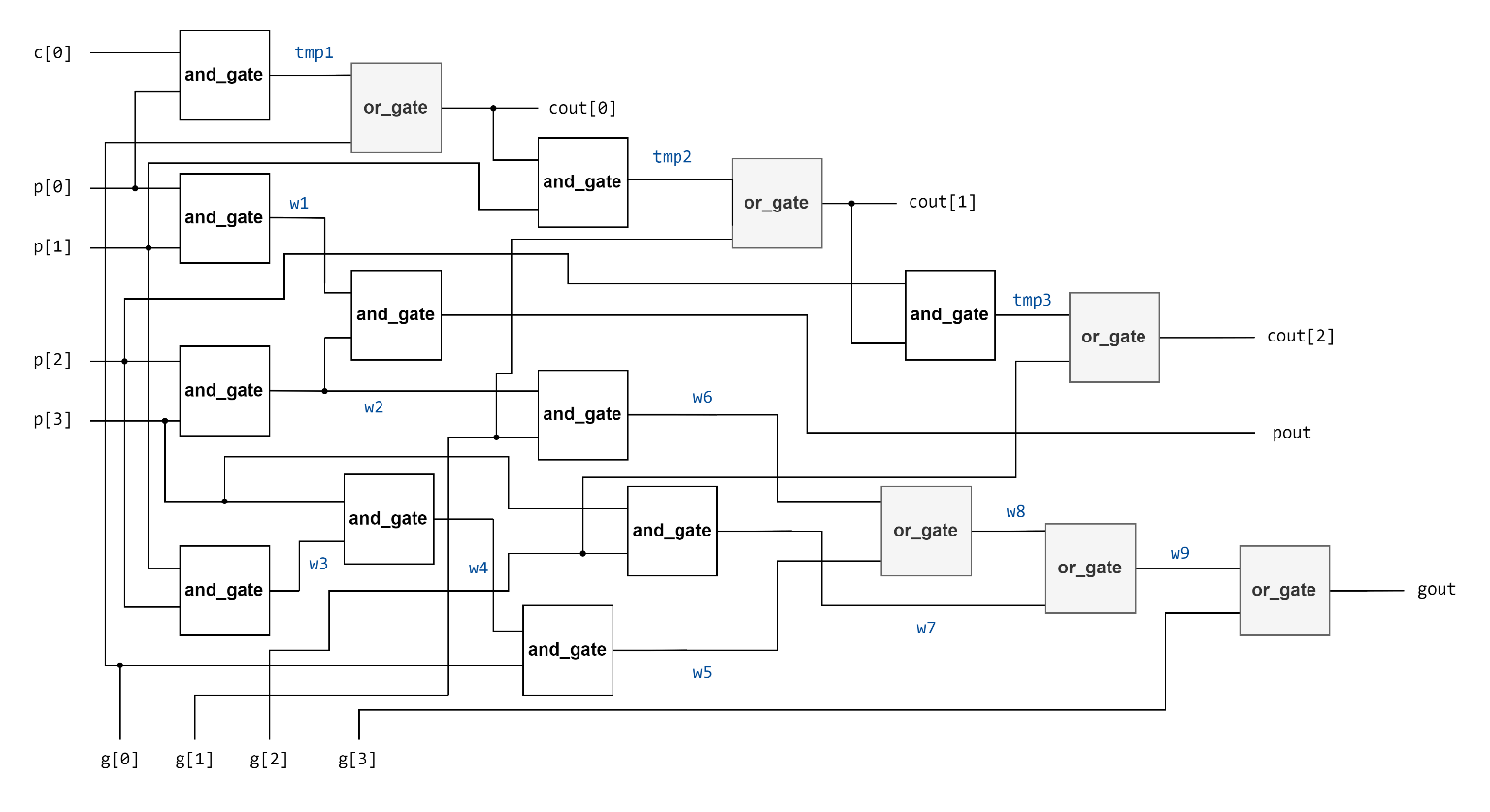


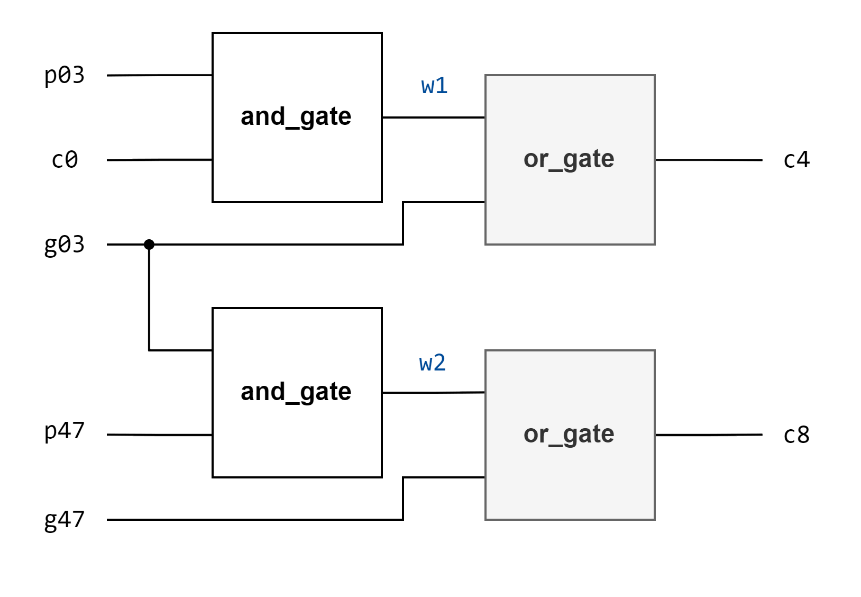
1. **(Gate Level) 8-bit carry-lookahead (CLA) Adder**

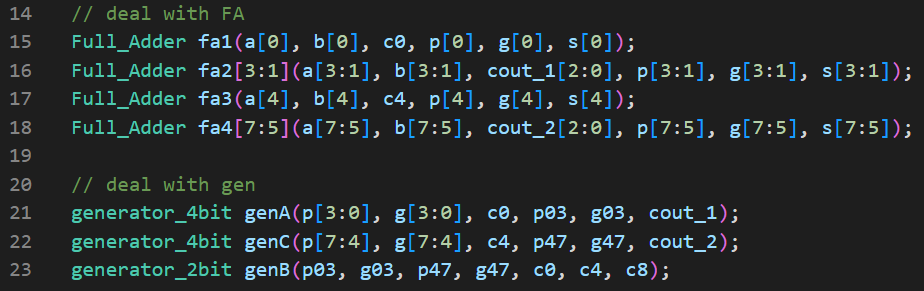
如上圖，這題我們需要八個Full Adder、兩個4-bit Carry-Look-Ahead Generator與一個2-bit Carry-Look-Ahead Generator。CLA特別的地方在於它的p, g及generator，讓它可以提前知道兩數相加時的carry，且每個bit的Full Adder可以直接拿自己的carry in同時進行運算，而不用像RCA需要等待上一個bit做完之後才知道carry out的值，能夠減少延遲時間，也不須依賴前面的計算結果來獲得值，因此較為有效率。

為了實作generator，我們需要先做出p與g兩個訊號，p代表的是propagate，負責將兩個bit做or，也就是說如果兩個bit中至少有一個bit為1就會是1，此時當carry in 是1時就會將此carry往下傳遞；g代表的是generate，負責將兩個bit做and，意即當兩個bit都是1時必會出現carry out。而在設計的部分我們將input丟進Full Adder中讓它output, p, g, sum再將p, g傳入generator中，這部分的Full Adder如下圖。

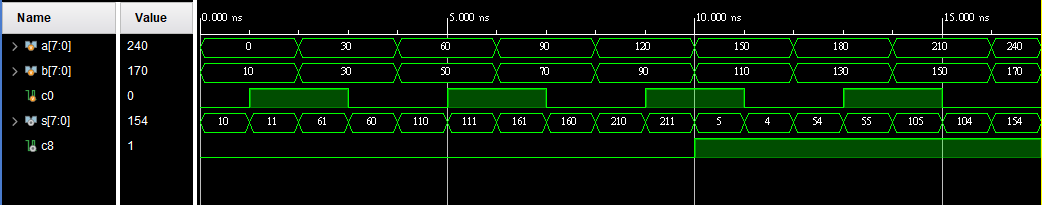
在計算進位的部分，它的equation為：，經過計算可得到原式 ，其中 即為 ，而 即為 ， 及 同理，於是我們可以知道c4即為，而c8即為 。因此在code的部分，我們傳入4bit的p及g，依照上面equation的框架做出 、 及cout，4bit generator的circuit diagram如下。

而後，4bit generator的pout及gout會再傳入2bit generator做上述c4及c8的equation再output出c4及c8，如下圖。

在code的部分，考慮到c4與其它cout接出來的地方不同，為了寫code上的方便，我們單獨處理c4，並將cout分為cout\_1及cout\_2兩部分，各為4bits，下圖為我們在Carry\_Look\_Ahead\_Adder\_8bit這個module中呼叫Full Adder及generator的部分。



在test的部份我們用clock延遲的方式讓c0會在a與b沒有變動時出現0跟1兩種情況，以便確認carry這部分沒有問題，如圖。而當c8為1時，可以發現s的數值無法完整呈現a、b相加該有的結果，這是因為s只有8bits，因此最大值已時進位表示為255，超過則會發生溢位的狀況，此時若將c8與s cascade起來就會得到正確的結果。

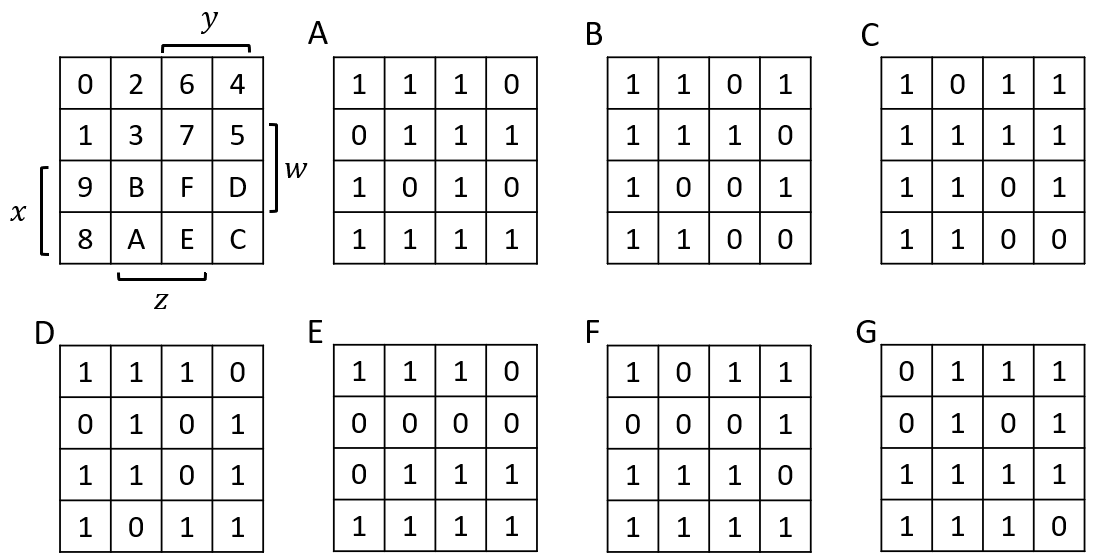


1. **(Gate Level) 4-bit multiplier**
2. **An exhaustive testbench design**
3. **FPGA: (Gate Level) Decode and execute**

這題是要沿用Advance question 2的module，並將結果顯示在FPGA板上。首先對於0F的每一個數字，它們對二進位以及7-segment的對應分別如下：

|  |  |  |
| --- | --- | --- |
| **Hex** | **Binary** | **7-segment** |
| 0 | 0000 | ABCDEF |
| 1 | 0001 | BC |
| 2 | 0010 | ABDEG |
| 3 | 0011 | ABCDG |
| 4 | 0100 | BCFG |
| 5 | 0101 | ACDFG |
| 6 | 0110 | ACDEFG |
| 7 | 0111 | ABC |
| 8 | 1000 | ABCDEFG |
| 9 | 1001 | ABCDFG |
| A | 1010 | ABCEFG |
| B | 1011 | CDEFG |
| C | 1100 | ADEF |
| D | 1101 | BCDEG |
| E | 1110 | ADEFG |
| F | 1111 | AEFG |

我們對7-segment的每一個線段，去紀錄它們分別在output為哪些值時會亮，並使用k-map去寫出function。下列的k-map，1表示要亮、0表示不亮。x、y、z、w 分別代表 rd[3]、rd[2]、rd[1]、rd[0] 的值：



將每個k-map算出來可以得到如下的算式：

1. **Summary**
2. **Contributions**

* **Code:**

**(Gate Level) 8-bit ripple carry adder (RCA)** by 李品萱

**(Gate Level) Decode and execute** by 李品萱

**(Gate Level) 8-bit carry-lookahead (CLA) Adder** by 唐翊雯

**(Gate Level) 4-bit multiplier** by 唐翊雯

**An exhaustive testbench design** by 唐翊雯

**FPGA: (Gate Level) Decode and execute** by李品萱

* **Report:**