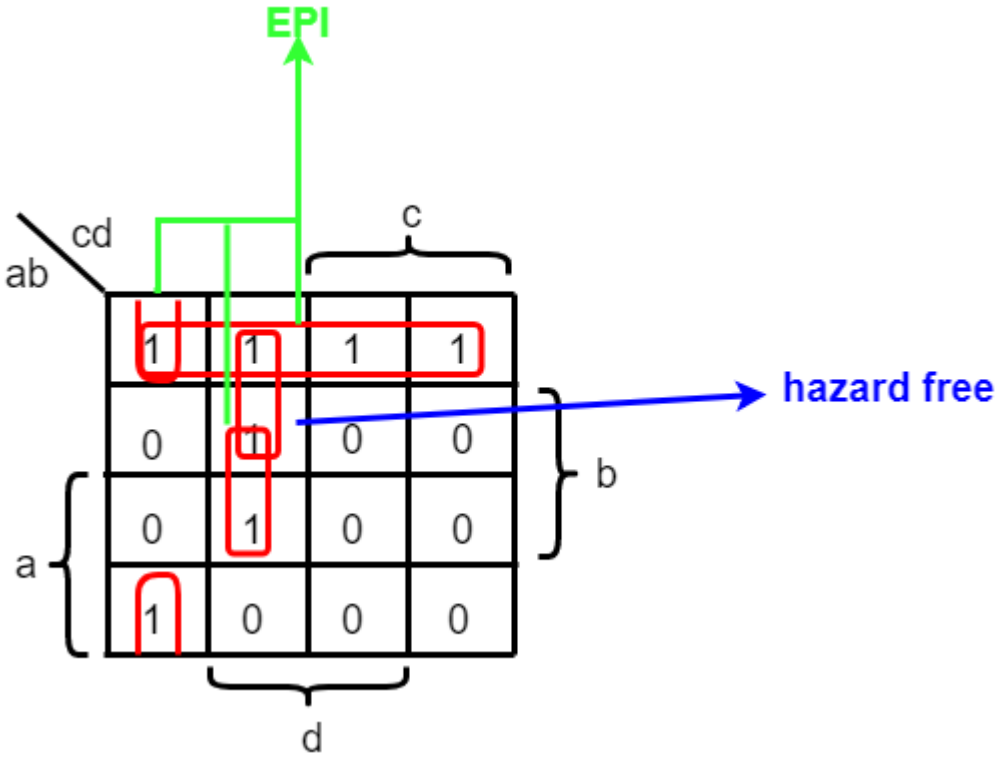
Logic Design

210510210 詹其佐

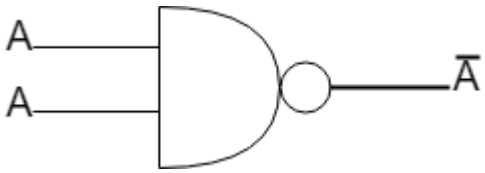
Lab1

K-map :



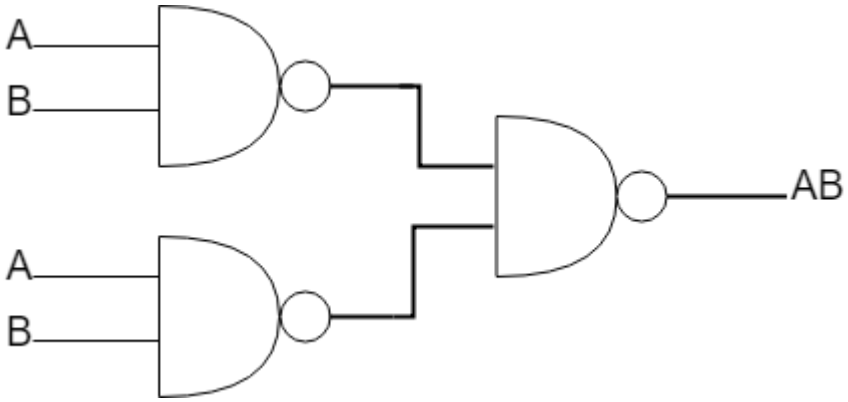
這是費式數列的 K-map{0, 1, 2, 3, 5, 8, 13}，K-map 大括號旁的英文字母表示，那兩行或兩列為 1 的值。有三個綠色的 EPI 原本我只用這三個 EPI 做，後來詢問助教才了解到，每塊都必須有相連的，否則在他們信號轉換的過程中就會跑出 hazard free 的問題。

My_not gate:



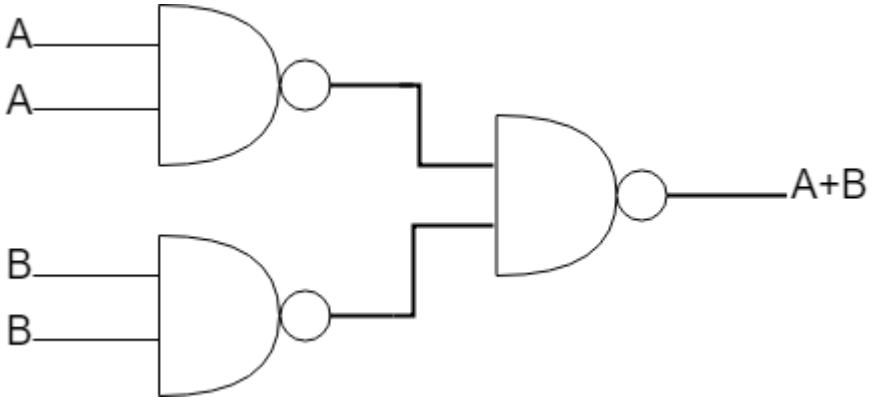
我用了一個 nand gate 做出 My_not gate。

My_and gate:



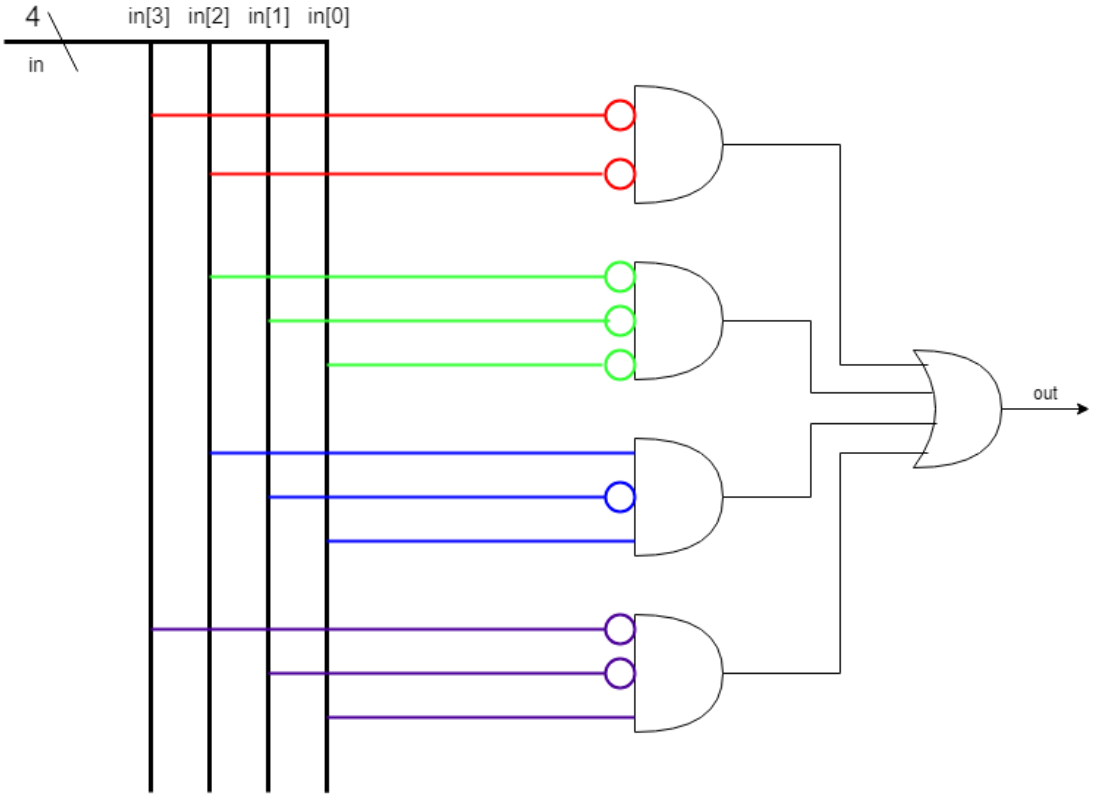
這個圖其實就等於先對 AB 做一次 nand 之後，再傳入我的 not gate。

My_or gate:



這個跟上面的很像，只是他會對非 A 和非 B 做 nand 的動作，結果我有運算，等於 A+B。

電路圖 ：



這是用 4 個 and 和 1 個 out 組合成的 4-input 電路圖，最後紫色的是處理 hazard free 所加上的。

原本我的電路圖就長這樣，幸好有同學提問助教是否電路圖也要用 nand gate 來表示，我才補了上面第二頁。所以上面那張圖的 not gate, and gate, or gate 都是用 my_not gate, my_and gate, my_or gate。我在寫這項作業的時候，也是先畫出電路圖，之後再想怎麼用 nand gate 來完成其他的 gate，在思考的過程中發現碰到瓶頸，是看了助教給的網頁才豁然開朗。

模擬結果:

```
ncsim> source /usr/cad/cadence/INCISIV/cur/tools/inca/files/ncsimrc
ncsim> run
time=   5,in=0000,out_G=1,out_D=1,out_B=1
time=  10,in=0001,out_G=1,out_D=1,out_B=1
time=  15,in=0010,out_G=1,out_D=1,out_B=1
time=  20,in=0011,out_G=1,out_D=1,out_B=1
time=  25,in=0100,out_G=0,out_D=0,out_B=0
time=  30,in=0101,out_G=1,out_D=1,out_B=1
time=  35,in=0110,out_G=0,out_D=0,out_B=0
time=  40,in=0111,out_G=0,out_D=0,out_B=0
time=  45,in=1000,out_G=1,out_D=1,out_B=1
time=  50,in=1001,out_G=0,out_D=0,out_B=0
time=  55,in=1010,out_G=0,out_D=0,out_B=0
time=  60,in=1011,out_G=0,out_D=0,out_B=0
time=  65,in=1100,out_G=0,out_D=0,out_B=0
time=  70,in=1101,out_G=1,out_D=1,out_B=1
time=  75,in=1110,out_G=0,out_D=0,out_B=0
time=  80,in=1111,out_G=0,out_D=0,out_B=0
All pass!! subarashii
Simulation complete via $finish(1) at time 80 NS + 0
./fib_tb.v:29             $finish;//program end here
```

在跑模擬結果的時候，第一次是錯的，原因是我裡面有一項原本包含非 B，結果我打成 B。後來又碰上許多問題，包括整行都是 Z 的情況，在後來下載了新的 testbench 之後就解決了。而且我發現 testbench 很好，我是先完成 G 就讓他跑了，他還是可以順利結束，不會因為其他沒做好就在第一步跳停。

程式碼部分截圖:

```
module my_not(in, out);
    input in;
    output out;

    nand nand1(out, in, in);

endmodule

module my_and(in[0], in[1], out);
    input [1:0]in;
    output out;

    wire nand_a;
    nand nand1(nand_a, in[0], in[1]);
    my_not not1(nand_a, out);

endmodule

module my_or(in[0], in[1], out);
    input [1:0]in;
    output out;

    wire not_a, not_b;
    my_not not1(in[1], not_a);
    my_not not2(in[0], not_b);
    nand nand1(out, not_a, not_b);

endmodule
```

結論

這次的 lab1 在實作的過程中碰到很多困難，從一開始不熟悉 verilog，到後來不會用工作站，都是慢慢理解網路的講義還有爬文，之後每次討論區基本上都會問出我的問題，回去看才發現又錯了，像之前就沒發現我的 G 整行都是 Z，但看了同學的問題，回去檢查發現到這個問題。最後又看到同學問了電路圖的問題，所以我又再補上自己的 gate，可以說這次的 lab1 是一直到最後都還在修改。要問助教的問題在實作過程中都已經得到解決了，所以目前沒有問題，謝謝助教。