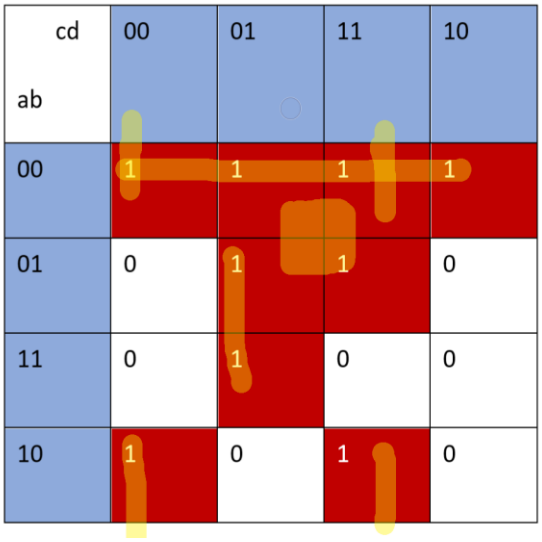
Lab 1

交大電物 0612129 (b123276353) 郭家佑

1. *講解*

cd / ab	00	01	11	10
00	0	1	2	3
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10



由上圖繪製之 K-map 可得知:

(~a ~b) + (~a d) + (~b c d) + (~b ~c ~d) + (b ~c d)

並以此製作 truth table

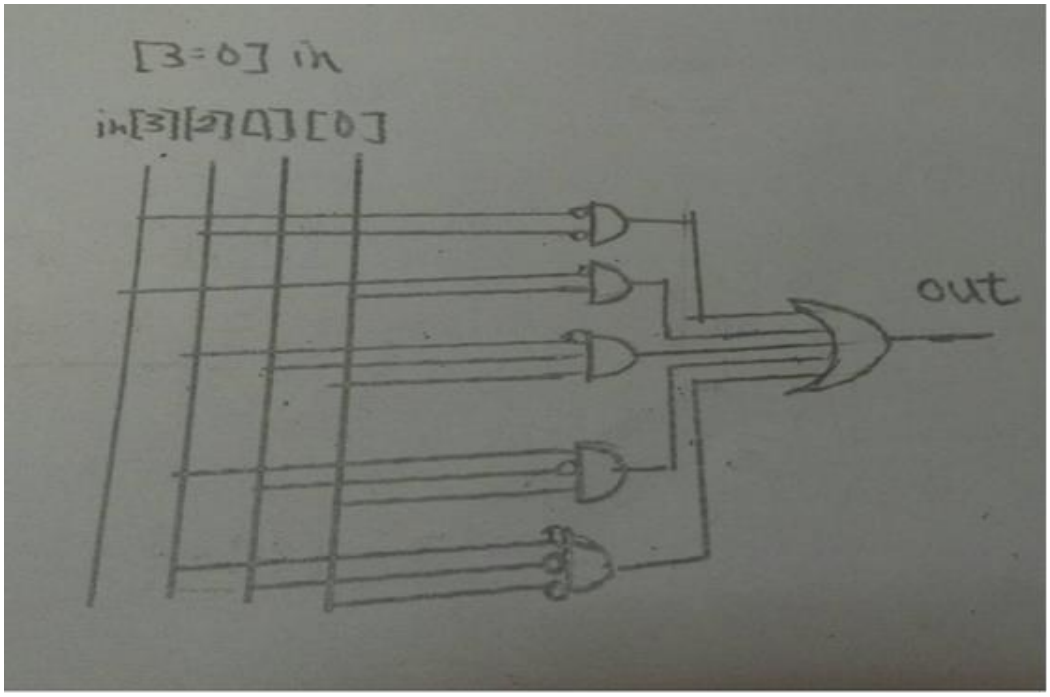
Truth table:

in[3]	in[2]	in[1]	in[0]		~a ~b	~a d	~b c d	b ~c d	~b ~c ~d		代表數字	out
0	0	0	0		1	0	0	0	1		0	1
0	0	0	1		1	1	0	0	0		1	1
0	0	1	0		1	0	0	0	0		2	1
0	0	1	1		1	1	1	0	0		3	1
0	1	0	0		0	0	0	0	0		4	0
0	1	0	1		0	1	0	1	0		5	1
0	1	1	0		0	0	0	0	0		6	0
0	1	1	1		0	1	0	0	0		7	1

1	0	0	0		0	0	0	0	1		8	1
1	0	0	1		0	0	0	0	0		9	0
1	0	1	0		0	0	0	0	0		10	0
1	0	1	1		0	0	1	0	0		11	1
1	1	0	0		0	0	0	0	0		12	0
1	1	0	1		0	0	0	1	0		13	1
1	1	1	0		0	0	0	0	0		14	0
1	1	1	1		0	0	0	0	0		15	0

可以此表與模擬結果對照以判斷是否有 coding 時發生的錯誤。

2. *電路設計圖:*



3. *模擬結果:*

```
time= 50,in=0000,out_G=1,out_D=1,out_B=1
time=100,in=0001,out_G=1,out_D=1,out_B=1
time=150,in=0010,out_G=1,out_D=1,out_B=1
time=200,in=0011,out_G=1,out_D=1,out_B=1
time=250,in=0100,out_G=0,out_D=0,out_B=0
time=300,in=0101,out_G=1,out_D=1,out_B=1
time=350,in=0110,out_G=0,out_D=0,out_B=0
time=400,in=0111,out_G=1,out_D=1,out_B=1
time=450,in=1000,out_G=1,out_D=1,out_B=1
time=500,in=1001,out_G=0,out_D=0,out_B=0
time=550,in=1010,out_G=0,out_D=0,out_B=0
time=600,in=1011,out_G=1,out_D=1,out_B=1
time=650,in=1100,out_G=0,out_D=0,out_B=0
time=700,in=1101,out_G=1,out_D=1,out_B=1
time=750,in=1110,out_G=0,out_D=0,out_B=0
time=800,in=1111,out_G=0,out_D=0,out_B=0
```

4. *問題討論:*

Verilog 除了每個程式一開始學習時都會遇到的語法、每個字的功能，幾乎都會遇到的適應期，雖然有友人表示和 C 有很大的差異，但對我好像沒有太大的困難，仔細想想後似乎和我學過一堆莫名的程式有點關聯，例如 Fortran 也有 integer 與 parameter 這幾個單詞。遇到最大的困難反而是 begin-end 的理解花了我許多的時間，結果似乎與 { } 概念差不多，對於 Fortran 或 Python 這種沒有使用 { } 的程式不算難上手。說起來不少部分與 Fortran 很相似，不過有些很相似，但不一樣，如 Fortran 內中 end module 這兩個詞是可以分開輸入的，另外，在宣告陣列時的寫法也是 arr[n:m](可為負，且宣告時 [ ] 要至於後方)。

還有一大問題是對於 wire 與 reg 使用上的差別，對於一開始的介紹只有說 reg 有記憶性，則 wire 則無，後來實際操作後發現目前使用上並無太多差異，只有在 always/initial 的區塊中需要賦值的時候才要使用 reg，對於實際上的瞭解還是要多實作和查詢。

實作中[3:0]in 在宣告時可以一同宣告(如:in = 4'b0000)，與一開始覺得的他是等同於 in_a,in_b…分成 4 個且要慢慢宣告，或是以{ }宣告，這也是看別人做的才知道的，在宣告時用 10 進位與 2 進位也似乎都是相同的。

~~編譯與執行時間較 gcc 相差很不少，也就是說想一個個測試功能可能需要更多的時間，何況仍需要考慮 delay 部分。~~(優化過後變十分的快速了)

基本上有問題都已經問過助教或自行查詢了，真的要說問題的話，就是為甚麼工作站速度優化過後差異如此多呢…?

5. 參考資料

Reg & Wire:

https://read01.com/zh-tw/oLnEdz.html#.WqODi-huZPY

Level of abstraction:

https://www.quora.com/What-is-the-difference-between-gate-level-data-flow-and-behavioural-modelling-in-Verilog