Lab01 (109061256 陳立萍)

1. lab1_1: a BCD-to-Excess-3 code converter

Design Specification

input: abcd(MSB to LSB)

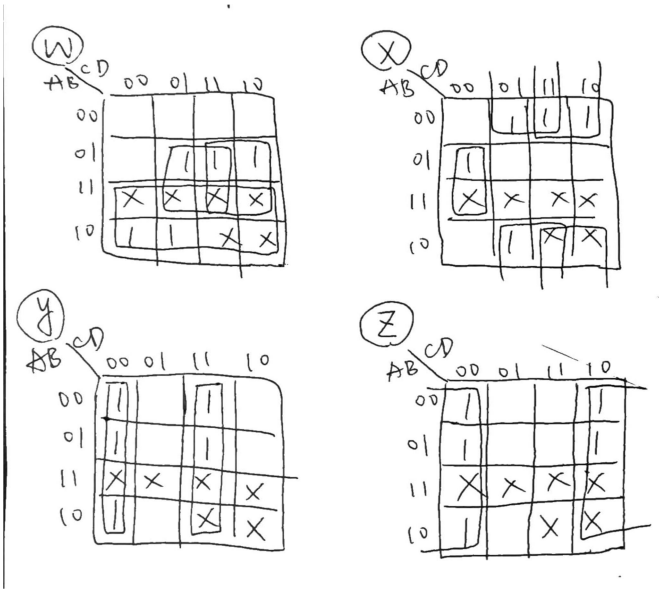
output: wxyz(MSB to LSB)

abcd: 0000 ~ 1001

1.1 Write the Boolean function/logic equation.

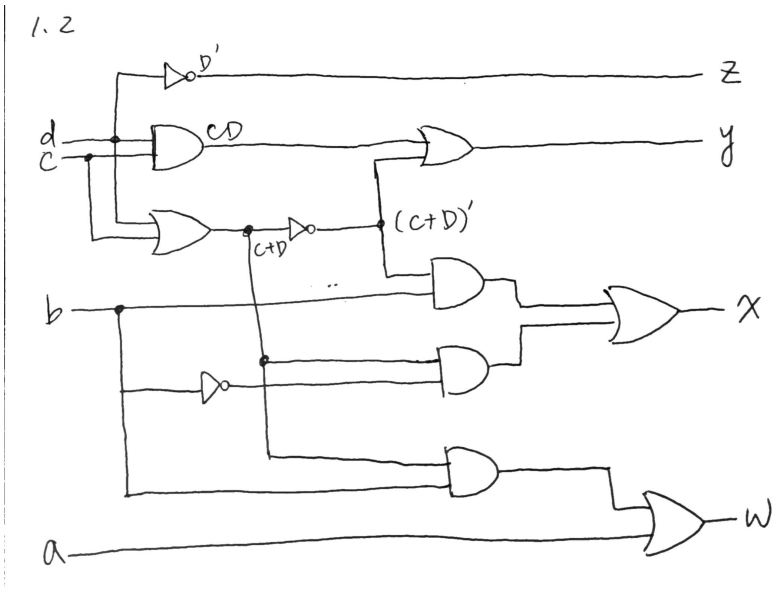
$wxyz = abcd + 0011$

a	b	c	d	w	x	y	z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X



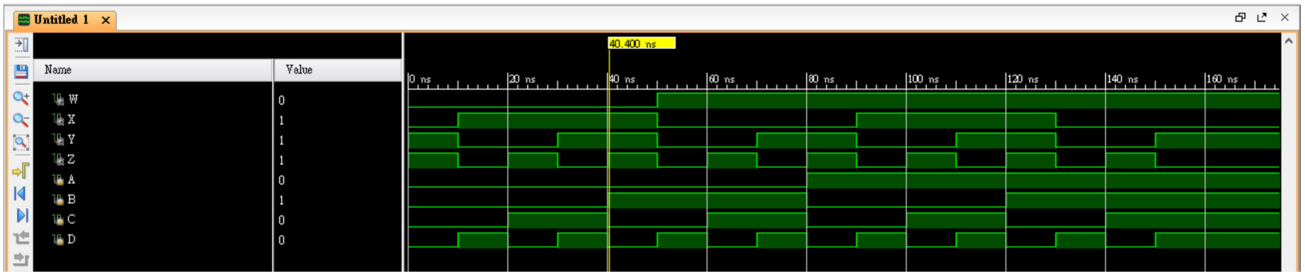
$$w = a + bc + bd$$
$$x = b'c + b'd + bc'd'$$
$$y = cd + c'd'$$
$$z = d'$$

1.2 logic diagram



Design Implementation

1.3 wave diagram



設計方法：

BCD 轉換成 Excess-3 的式子如上 1.1 ，wxyz = abcd + 0011，接著寫出對應的 truth table，再用 k-map 化簡 output (wxyz)，得到 wxyz 各自的 Boolean function.

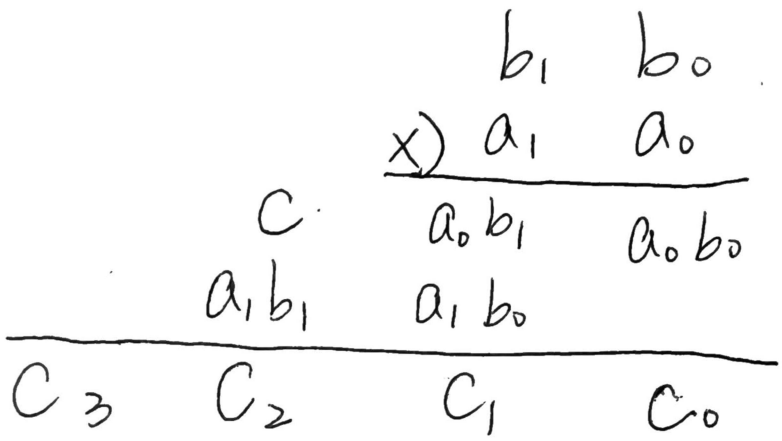
2. lab1_2: an unsigned 2-bit x 2-bit binary multiplier

Design Specification

input: a1, a0, b1, b0

output: c3, c2, c1, c0, c

2.1



2.1 $C_0 = a_0b_0$

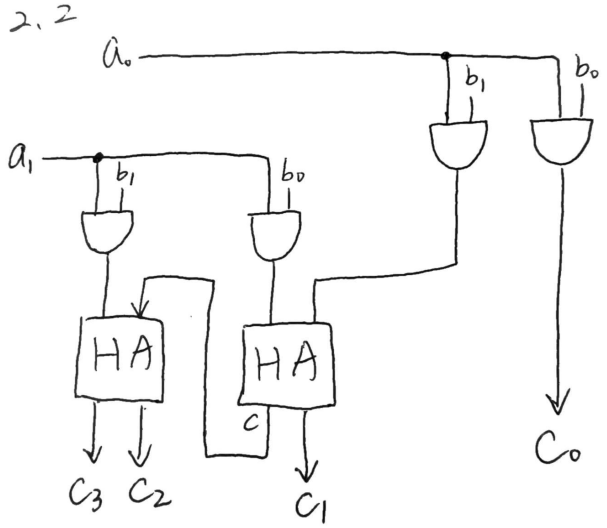
$C_1 = a_0b_1 \oplus a_1b_0$

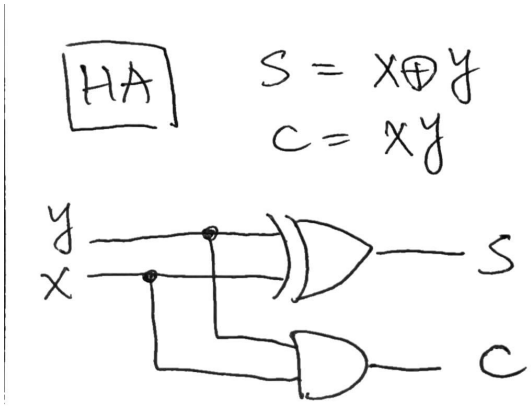
$C = a_0b_1 a_1b_0$

$C_2 = a_1b_1 \oplus C$

$C_3 = a_1b_1C$

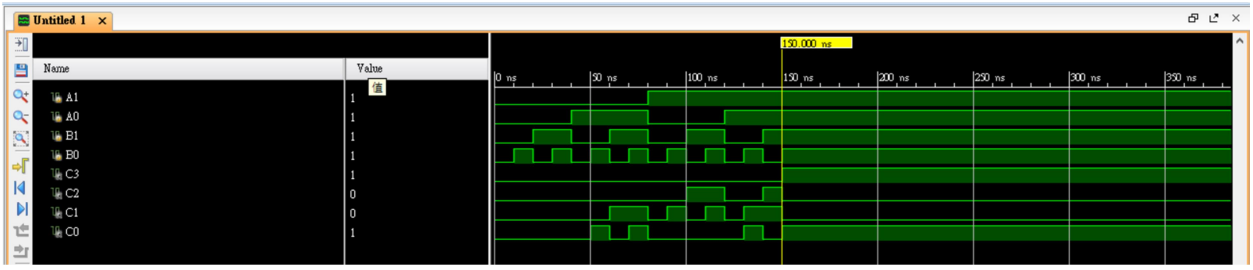
2.2 logic diagram





Design Implementation

2.3 wave diagram



設計方法：

經由乘法直式計算（在 2.1）可以得到 input 與 output 之間的關係，而其中 c 為 a0b1 與 a1b0 做加法後所得之進位，而其中的加法運算利用 half adder 作加法，得到的每一個 output cx 即為每次加法所的到的 S，c0 為最低位，c3 為最高位。

3. lab1_3: a 3-bit binary adder/subtractor

Design Specification

input: a[2:0], b[2:0], m

output: s[2:0], c[2:0], v

3.1

M	Function	
0	A+B	加法
1	A+B'+1	減法

M	B⊕M
0	B⊕0=B
1	B⊕1=B'

3.1

$S_0 = a_0 \oplus (b_0 \oplus m) \oplus m$

$C_0 = a_0(b_0 \oplus m) + m a_0 + m(b_0 \oplus m)$

$S_1 = a_1 \oplus (b_1 \oplus m) \oplus C_0$

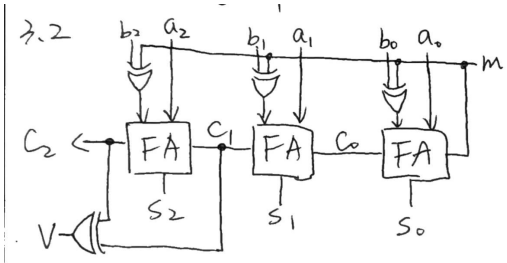
$C_1 = a_1(b_1 \oplus m) + C_0 a_1 + C_0(b_1 \oplus m)$

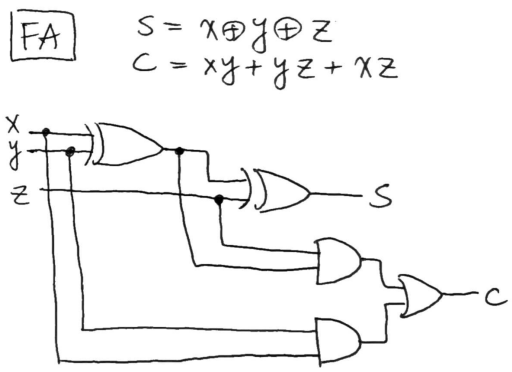
$S_2 = a_2 \oplus (b_2 \oplus m) \oplus C_1$

$C_2 = a_2(b_2 \oplus m) + C_1 a_2 + C_1(b_2 \oplus m)$

$V = C_2 \oplus C_1$

3.2 logic diagram





Design Implementation

3.3 wave diagram

設計方法：

使用 exclusive or 作加數與補數的轉換，如 3.1 表格所示，在每次作運算前，都先將 bx 的與 m 作 exclusive or 的動作，使其化為加數或是補數，再與 a 的每一個位元用 full adder 相加，每一級加法所產生的 s 即為答案，而 c 用來記住每一級加法所產生的 carry，另外，v 為 overflow indicator 為(c2 exclusive c1)

Bonus. Lab1_bonus: for two 3-bit unsigned numbers a(a2a1a0) and b(b2b1b0), build a logic circuit to output o as the smaller number.

Design Specification

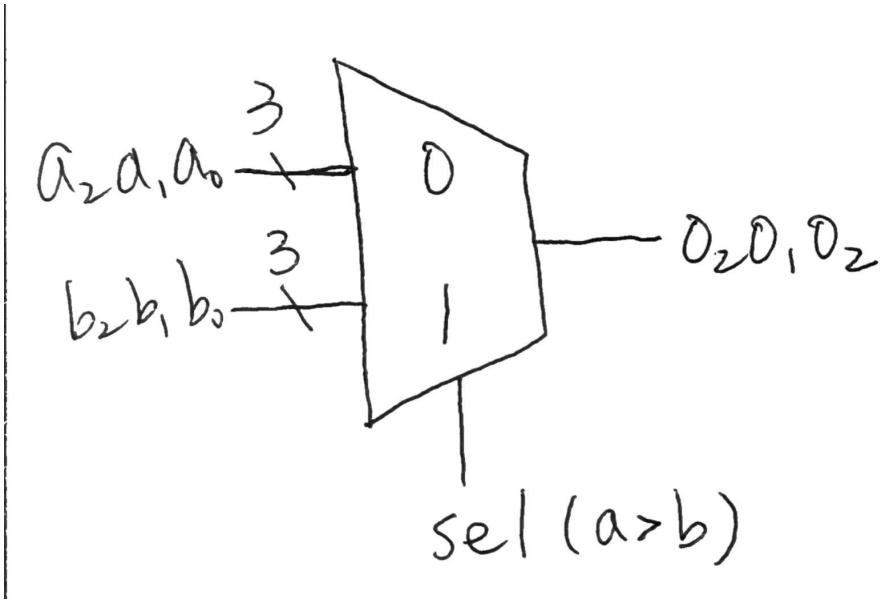
input: a2, a1, a0, b2, b1, b0

output:　o2, o1, o0, sel

bonus

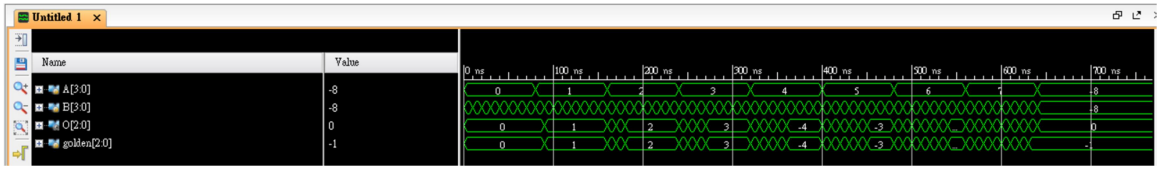
$$sel = a_2 b_2' + \overline{a_2 \oplus b_2} \cdot a_1 b_1' + \overline{a_2 \oplus b_2} \cdot \overline{a_1 \oplus b_1} \cdot a_0 b_0'$$

logic diagram



Design Implementation

wave diagram



設計方法：

狀況一：a2 = 1, b2 = 0

如果 a>b 要為 true，則可得 boolean function: (a2 & (~b2))

狀況二：a2 = b2, a1 = 1, b1 = 0

如果 a>b 要為 true，則可得 boolean function: ((~(a2 ^ b2)) & a1 & (~b1))

狀況三：a2 = b2, a1 = b1, a0 = 1, b0 = 0

如果 a>b 要為 true，則可得 boolean function: ((~(a2 ^ b2)) & (~(a1 ^ b1)) & a0 & (~b0))

綜合以上三點，將三狀況 OR 在一起，即可表達所有 a>b 的狀況，故表達 a > b (a2, a1, a0, b2, b1, b0)的 Boolean function 如下: (a2 & (~b2)) | ((~(a2 ^ b2)) & a1 & (~b1)) | ((~(a2 ^ b2)) & (~(a1 ^ b1)) & a0 & (~b0))。

再用 MUX 以 sel 做選擇，如果 sel 為 1，亦即 a>b 為 true，那麼 b 會被選出並 assign 給 o，若 sel 為 0，亦即 a>b 為 false，那麼 a 會被選出並 assign 給 o。

Discussion

Lab1_1

這個題目非常簡單，就是寫出每一個 bit 的 boolean function 再用 k-map 化簡即可得到簡化之答案，再一一 assign 給 output 即可。

Lab1_2

這題在將乘法直式寫出來後，就非常清楚該如何寫出 boolean function，而結果也與預期的相同。

Lab1_3

在做這題時，起初忘記 b 的每一個 bit 要 exclusive or m，因此導致輸出結果一直無法正確，除錯除了好久才發現是這個原因。

在 Simulation 的波形圖上可以調數值的顯示要是 signed 或是 unsigned ，而電腦預設為 unsigned，導致一開始跑的時候，第一組數值是 A=4,B=4,S=0

但是 testbench 上給的數值是 A=-4,B=-4,S=0，一開始一直以為是程式寫錯，後來才發現旁邊可以調數值的有號與無號。

調整完後的輸出便與 testbench 給的值相符。

Lab1_bonus

做這題的時候原先我用 if else 的語法一一比較 a 與 b 的每一個 bit，但是在 assign 給 o 的每一個 bit 的時候，卻無法將單一一個 ax 或是 bx assign 給 ox(ox 宣告為 output 型態)，因為宣告型態的問題，所以後來就改了方法，用 mux 下去寫，然而整個程式變得簡短許多，跑出來的波形圖也為正確答案。

Conclusion

經過 lab01 後上學期忘記的東西大部分都回來了。不過光是 lab01 就交了很多新的語法，像是 always 與 if else 和 case 等語法，都還在可以理解的範圍。希望之後可以學到更多不一樣的，尤其是一些設計想法，也希望自己可以理解這些新的知識。

References

教授授課頭影片：語法運用，符號運用。