**Lab 1 110060027朱豐蔚**

1. **exp\_1**

**Design Specification**

graycoder

a

b

c

d

w

x

y

z

Input: a, b, c, d.

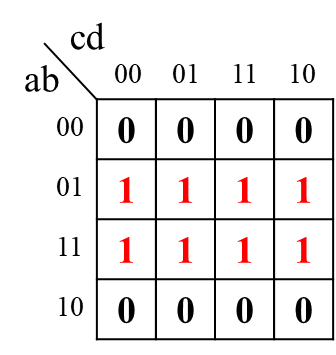
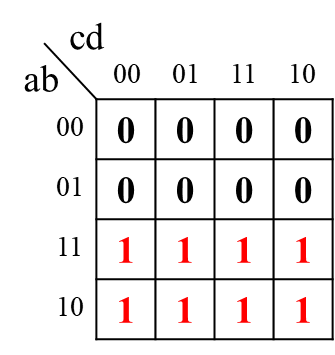
Output: w, x, y, z.

**Design Implementation**

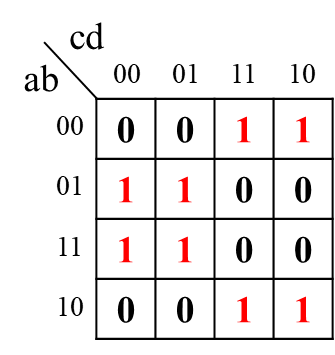
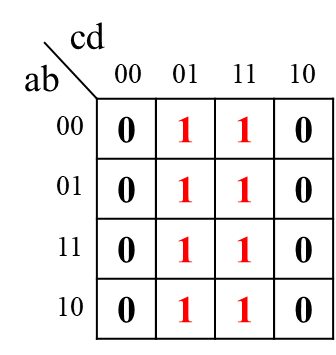
Use truth table and K-map to design the functions:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a | b | c | d |  | w | x | y | z |
| 0 | 0 | 0 | 0 |  | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |  | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |  | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 |  | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |  | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 |  | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 |  | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |  | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 |  | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 |  | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |  | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 |  | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 |  | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |  | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 |  | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 |  | 1 | 0 | 0 | 0 |

w = a x = ab’+a’b = a⊕b



y = bc’+b’c = b⊕c z = cd’+c’d = c⊕d



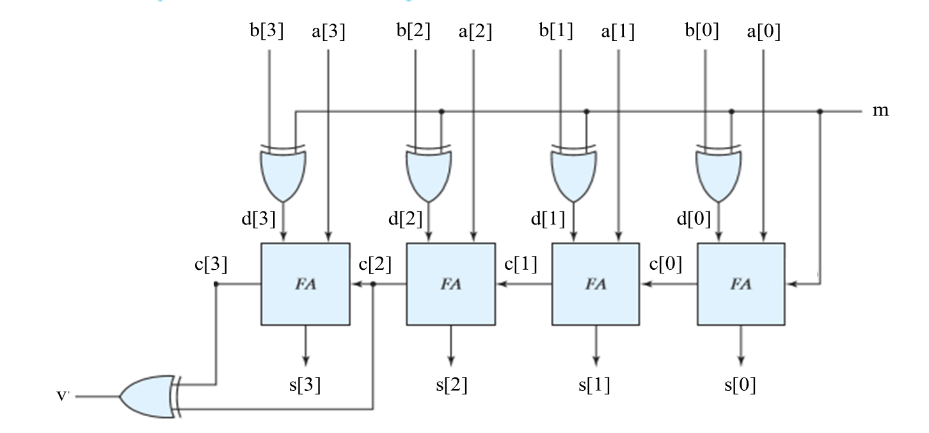
Code:

assign w = a;

assign x = a ^ b;

assign y = b ^ c;

assign z = c ^ d;



1. **exp\_2**

**Design Specification**

Input: a[3:0], b[3:0], m.

Output: s[3:0], v.

Wire: c[3:0], d[3:0].

**Design Implementation**

增加c[3:0], d[3:0]以方便撰寫程式，其中:

c[i]為進行第i個FA運算後的進位

d[i] = b[i]⊕m以方便撰寫後續程式

再套用FA公式:

s[i] = a[i]⊕d[i]⊕c[i-1]

c[i] = a[i]d[i] + c[i-1](a[i]⊕d[i])

v = c[3]⊕c[2]

Code:

assign d[0] = b[0] ^ m;

assign d[1] = b[1] ^ m;

assign d[2] = b[2] ^ m;

assign d[3] = b[3] ^ m;

assign s[0] = a[0] ^ d[0] ^ m;

assign c[0] = (a[0] \* d[0]) | ((a[0] ^ d[0]) \* m);

assign s[1] = a[1] ^ d[1] ^ c[0];

assign c[1] = (a[1] \* d[1]) | ((a[1] ^ d[1]) \* c[0]);

assign s[2] = a[2] ^ d[2] ^ c[1];

assign c[2] = (a[2] \* d[2]) | ((a[2] ^ d[2]) \* c[1]);

assign s[3] = a[3] ^ d[3] ^ c[2];

assign c[3] = (a[3] \* d[3]) | ((a[3] ^ d[3]) \* c[2]);

assign v = c[3] ^ c[2];

1. **exp\_3**

**Design Specification**

Input: a[2:0], b[2:0].

Output: o[2:0]**.**

experiment\_3\_design

a[2:0]

b[2:0]

o[2:0]

3

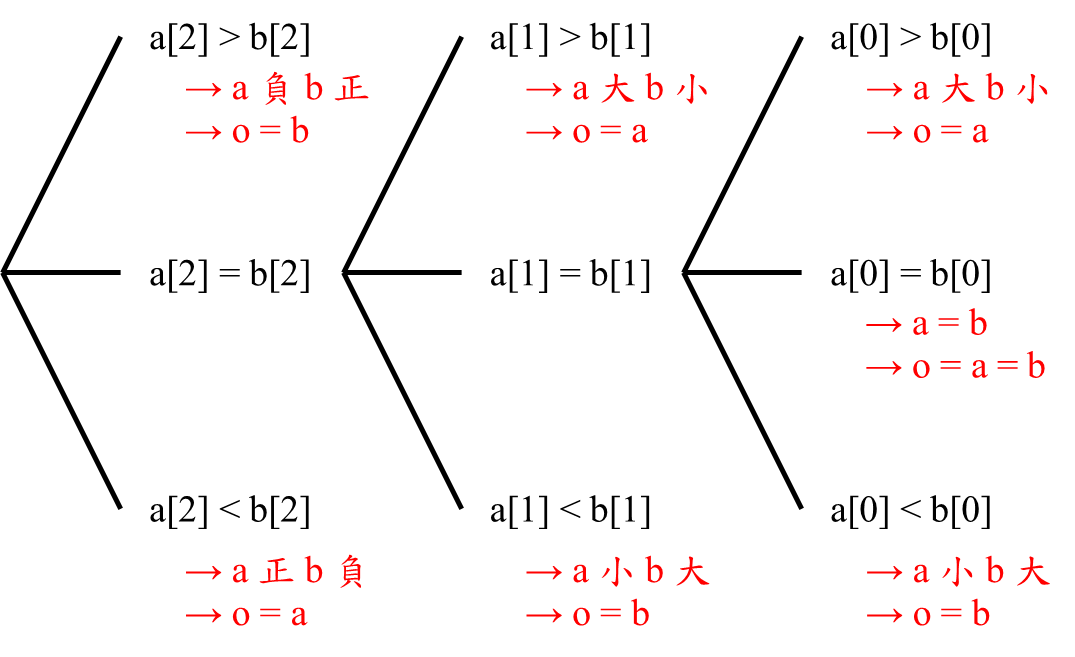
3

3

**Design Implementation**

為使用if來做撰寫，先畫出樹狀圖整理o和a, b之間的關係:

(題目要求a, b為signed numbers，需特別注意a[2], b[2]的大小關係)



Code:

reg [2:0] o;

always @(a or b)

begin

if (a[2] > b[2])

o[2:0] = b[2:0];

else if (a[2] < b[2])

o[2:0] = a[2:0];

else

if (a[1] > b[1])

o[2:0] = a[2:0];

else if (a[1] < b[1])

o[2:0] = b[2:0];

else

if (a[0] > b[0])

o[2:0] = a[2:0];

else

o[2:0] = b[2:0];

end

**======================================================================**

**Discussion**

由於此三題都來自於以往的邏輯設計課，所以在設計的思路上，我沒有遇到太多的障礙，反而是在第二題Adder/Substraterk的code撰寫中，我發現了以往紙面上書寫的function，因為需要電腦打入而變得更加冗長和難以閱讀，因此，我加入了d[3:0]的想法，也成功簡化撰寫，使程式更加明朗，debug上也更加順利。

在實驗結果分析中，一開始由於不熟悉操作，因此在結果的判讀上遇到了很大的障礙，例如說第三題中的數字，電腦自動顯示是unsigned numbers，因此數字的比大小就會不明朗清楚，這一度讓我以為是結果錯誤，而後來詢問助教，也才學會透過改變模擬結果中的參數radix，來達到更好判讀結果的方法，這也是我這次實驗的收穫之一。

**Conclusion**

在Lab 0當中，只是讓我們照著已經寫好的程式去打，試著去熟悉Vivado的操作，而在Lab 1很不一樣的是我們必須從頭去自行設計，這個差別就讓第一次自行接觸設計Verilog的我們在看到繁雜的電腦介面時無從下手。

而經過了這三題的演練，我已經學會了最基本的操作，不會像第一次自行設計時那樣徬徨，我想這是我在Lab 1中最大的收穫。另外一方面，題目的選擇大多是來自上學期的邏輯設計課中，也讓我能複習邏輯設計的內容，以做好面對這學期實驗課的準備。

**References**

Adder/Substrater圖片參考來源(有自行修改): <https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.answersproject.com%2FExpertAnswers%2Fverilog-write-code-following-question-4-bit-adder-subtractor-overflow-detection-cascading-&psig=AOvVaw0UUt5byfhPkIjvEOwzJbOW&ust=1646795320956000&source=images&cd=vfe&ved=0CAsQjRxqFwoTCKD5zebEtfYCFQAAAAAdAAAAABAD>