國立中山大學電機工程系

實用數位系統設計

Homework 1-2

學生：徐崇皓

學號：C110152334（外校）

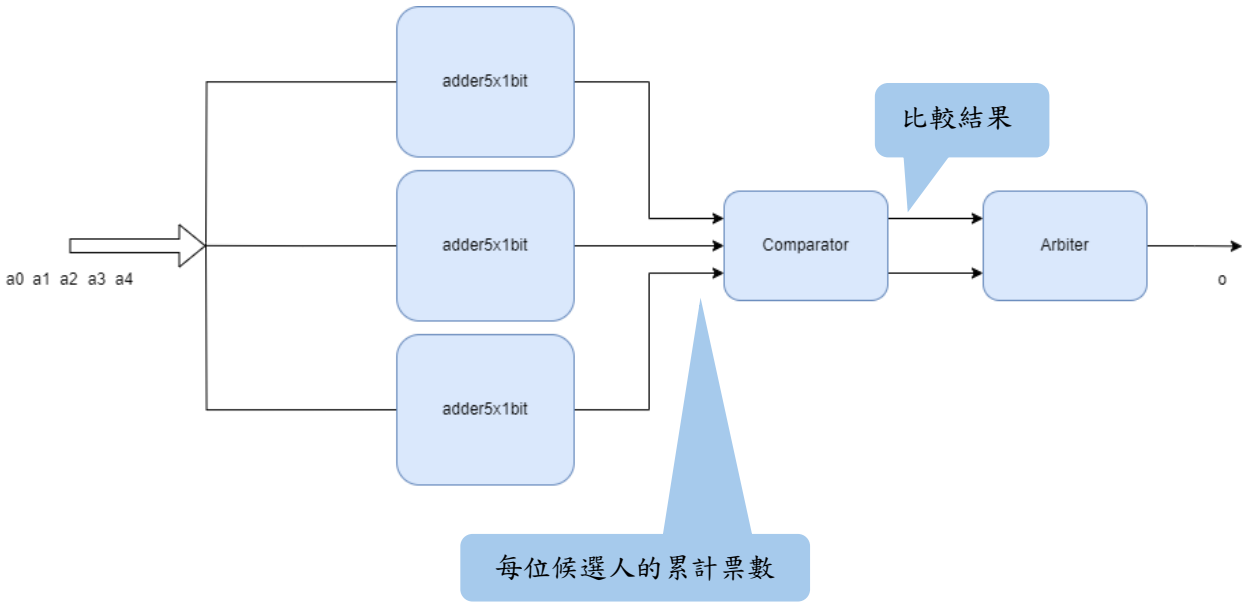
一、 投票機 voting_v1：

1. 演算法：

首先，我們會統計每位候選人獲得的票數，然後進行比較，找出票數最多的候選人，最後將其輸出。接下來，我們可以將電路分成幾個部分進行設計。首先是統計每位候選人的票數。由於輸入是以 one-hot encoding 的方式提供，因此我們只需將每個輸入的每個位元傳送給對應的計數器。在這題，有三個輸入，因此需要三個計數器。接著是比較器，找出票數最多的候選人，然後輸出相應的結果。

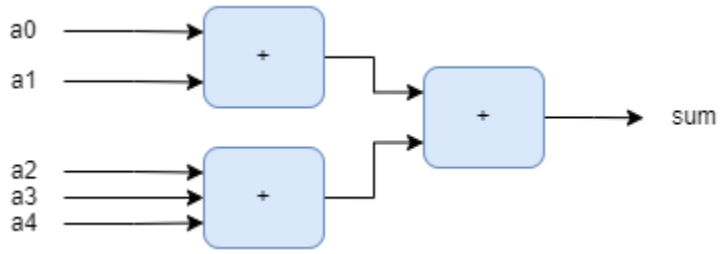
2. 電路架構：

i. 使用 Top Down 的方式剖析，我們可以將電路拆解成如下圖，先將每位候選人的得票數統計，然後給後面的比較器做比較，最後依照權重輸出最大數值的候選人。



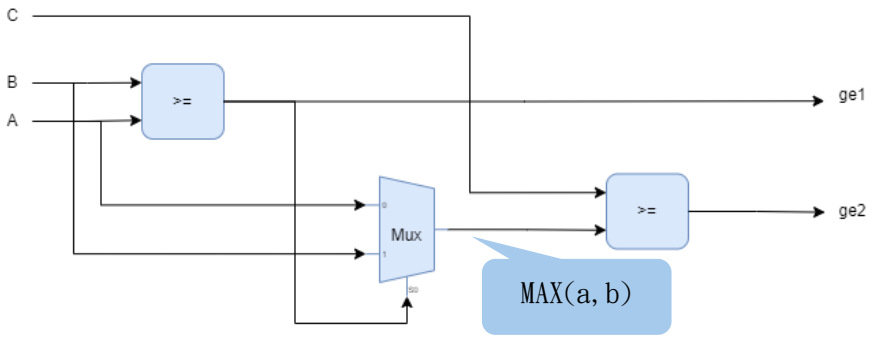
ii. Adder5x1bit

有兩種時做方法，第一種是製作一個 MUX 透過選擇線，選擇對應的輸出，速度相當的快速，缺點也很明顯就是面積非常的大，所以我選擇使用第二種，使用巢狀結構來組成 5 位元 1bit 的加法器，第一級分別為 2 輸入的 HA 與 3 輸入的 FA，第二級由一個 2bit 的 Adder 把前一級結果加總。



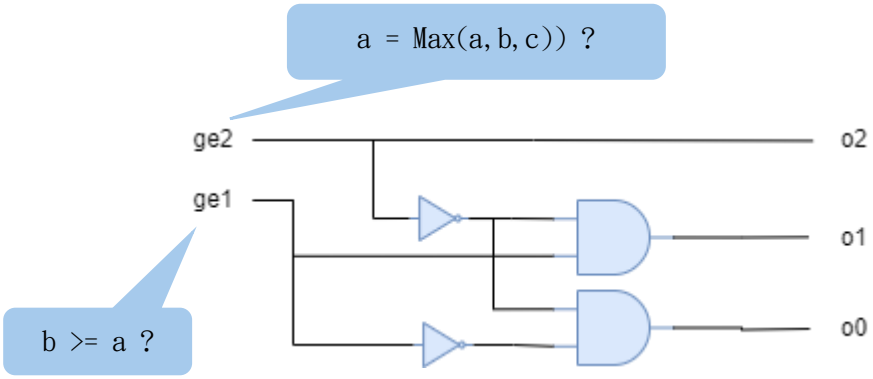
iii. Comparator

使用兩個比較器(透過合成工具)完成，結構如下圖，A 與 B 做比較完後將比較大的值傳入下一級做比較，這樣就可以知道他們的比較關係，同時為了要確保權重，所以將 C 放置最後一級(C 權重最高)且把判斷是增加成大於等於，這樣權重就會變成 C>B>A。



iv. Arbiter

為了確保輸出只有一個結果，所以要把前一級的比較結果，經過邏輯運算後輸出。



3. 電路分析：

i. Area

這部分使用 Desgin Compiler 使用 TSMC 0.13um 的製程下去做合成最後計算 Gate Count(Total Area/NAND Area)來評估此電路的大小。

	um^2	Gate Count
NAND	5.0922	1
voting	383.612392	≅75.33

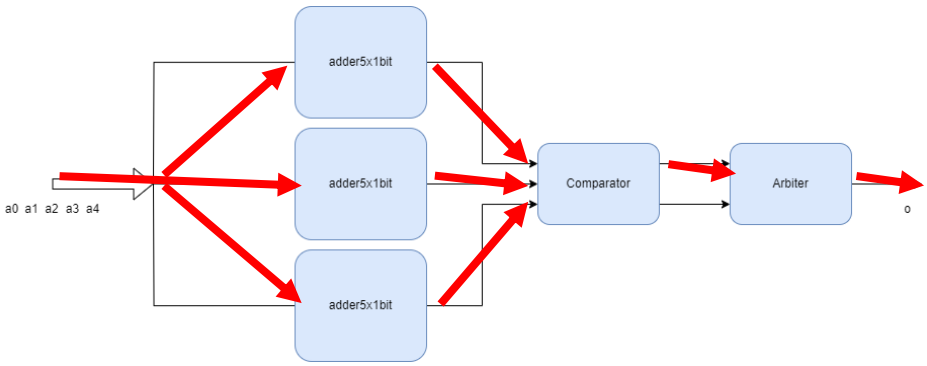
ii. Speed

Critical Path(i1 and i2) :

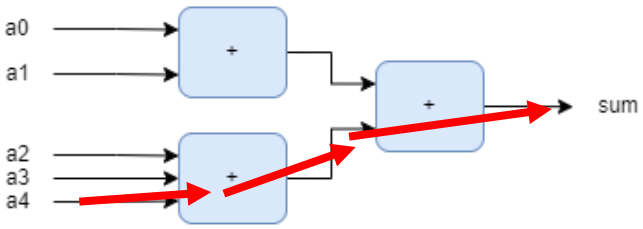
Adder5x1bit -> Comparator -> Arbiter

FA -> comparator3bit->MUX-> comparator3bit ->not -> and gate

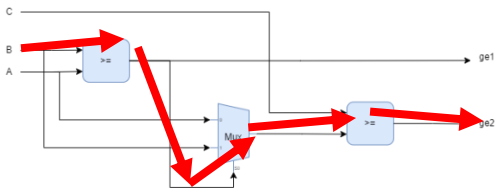
voting



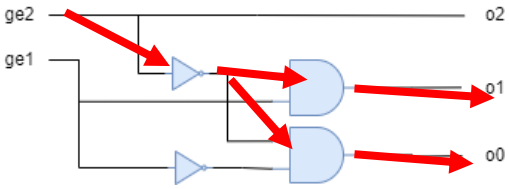
Adder5x1:



Comparator:



Arbiter



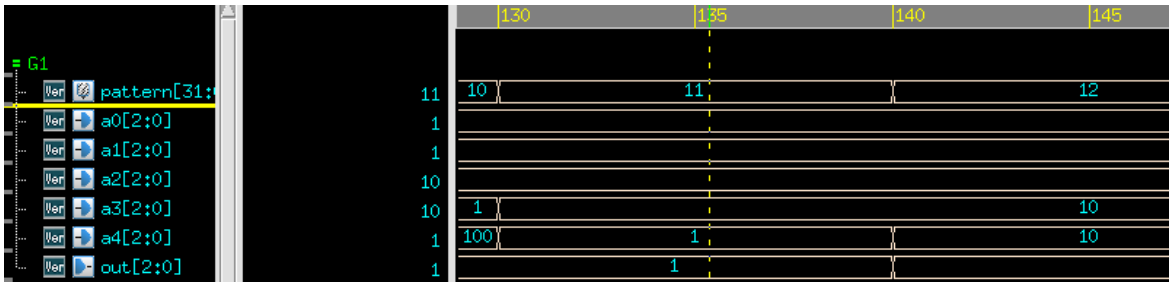
4. 電路模擬

因為這個電路輸入種類特別少只有 243 種(3^5)，所以把每個輸入全部都驗證一次確保電路的功能。我們舉五種常見的狀況舉例

1. 票數為 3:2:0

a0:001 a1:001 a2:010 a3:010 a4:001

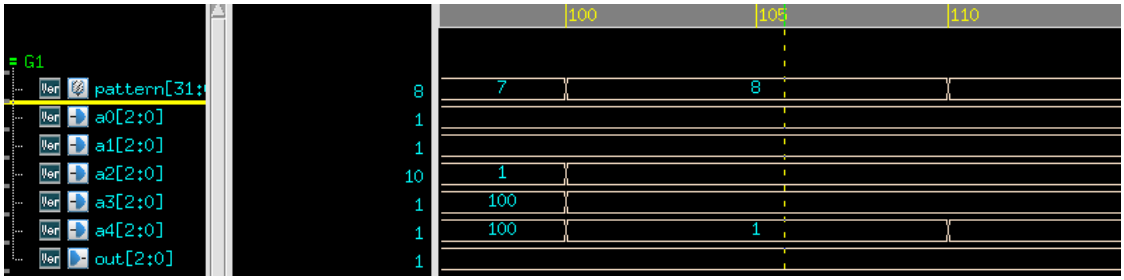
比分 0:2:3 輸出 001



2. 情況票數為 4:1:0

a0:001 a1:001 a2:010 a3:001 a4:001

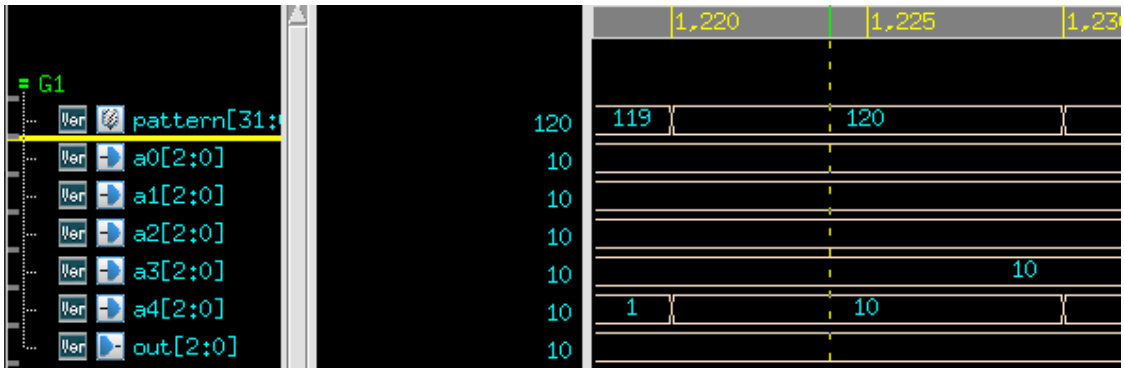
比分 0:2:3 輸出 001



3. 票數為 5:0:0

   a0:001 a1:001 a2:010 a3:001 a4:001

   比分 0:5:0 輸出 010

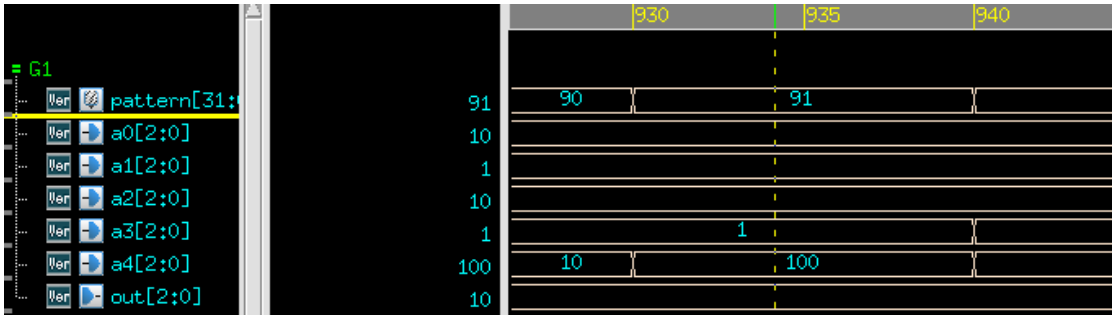


4. 票數為 2:2:1

   a0:010 a1:001 a2:010 a3:001 a4:100

   比分 1:2:2 當比分相同時輸出較高權重的項目。

   輸出 010



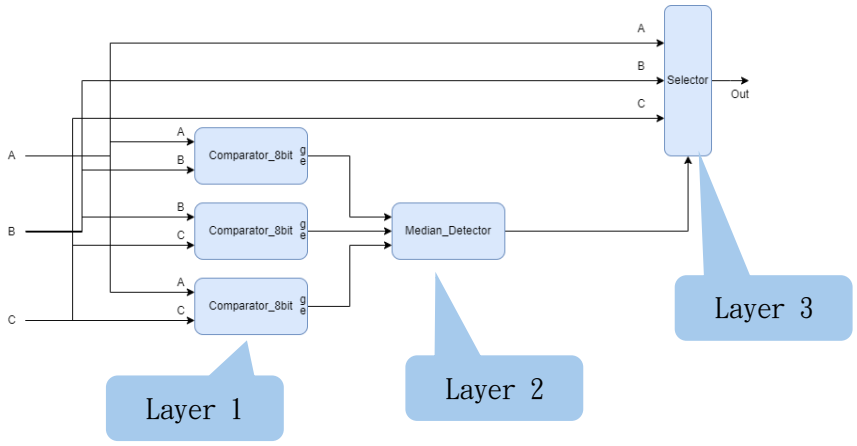
二、 中位數 median_v1

1. 演算法：

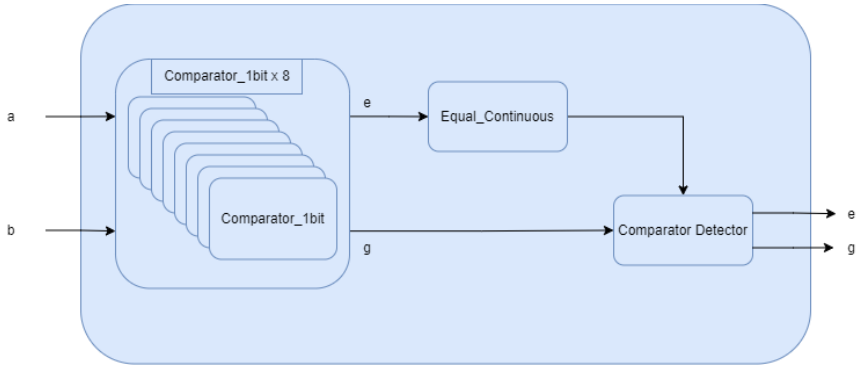
在三個數字中我們要判斷中位數相當的簡單，我們需要把每一個數值拿去比較，假設三個數字為 abc，判斷是否為 a >= b >= c 中的 b，如果符合，這個數為三個數中的中位數，由於這個做法是直接將輸入是為一個數值去做所以他除了能接收 One Hot encoding 之外，還可以接受一般 8bit 表示的無號數。

2. 電路架構：

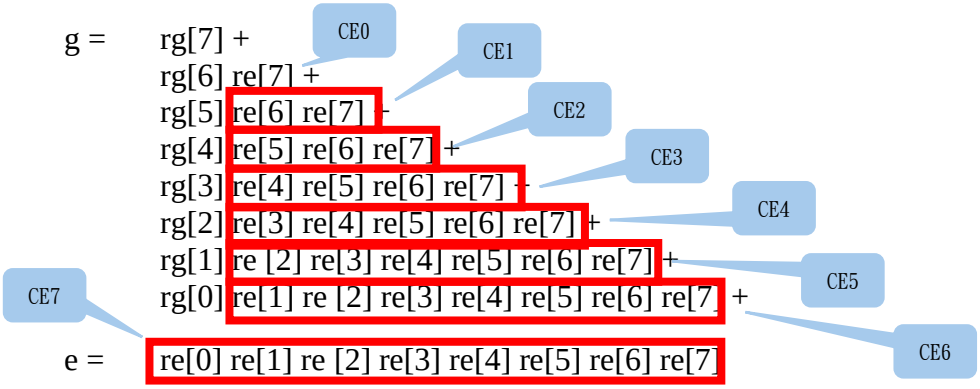
i. 因為判斷條件可以重複利用，所以 Layer 1 先將比較結果全都運算完，Layer 2 使用比較結果判斷三個輸入中的哪一項是中位數，Layer 3 選擇輸出結果。



ii. Comparator_8bit

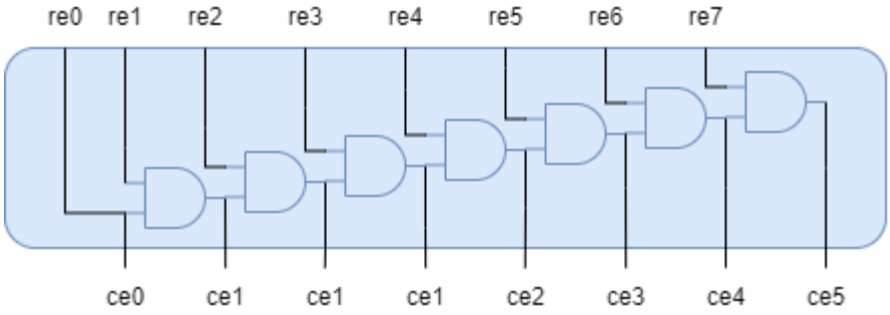


由三個部分組成，分別是 8 個 1bit Comparator、連續相等判斷與輸出比較這三個部分，先由前面的 8 個 bit 比較完後再輸出給後面的偵測器，假設輸出的 Comparator 叫做 re(e)與 rg(g)，我們觀察他的 function，分別可以看到會出現要判斷連續的 re，因此我們製作了一個模組叫做 Equal Continuous 來計算連續的 re，將計算完後的值丟給 Comparator_Detection 做後續的計算。

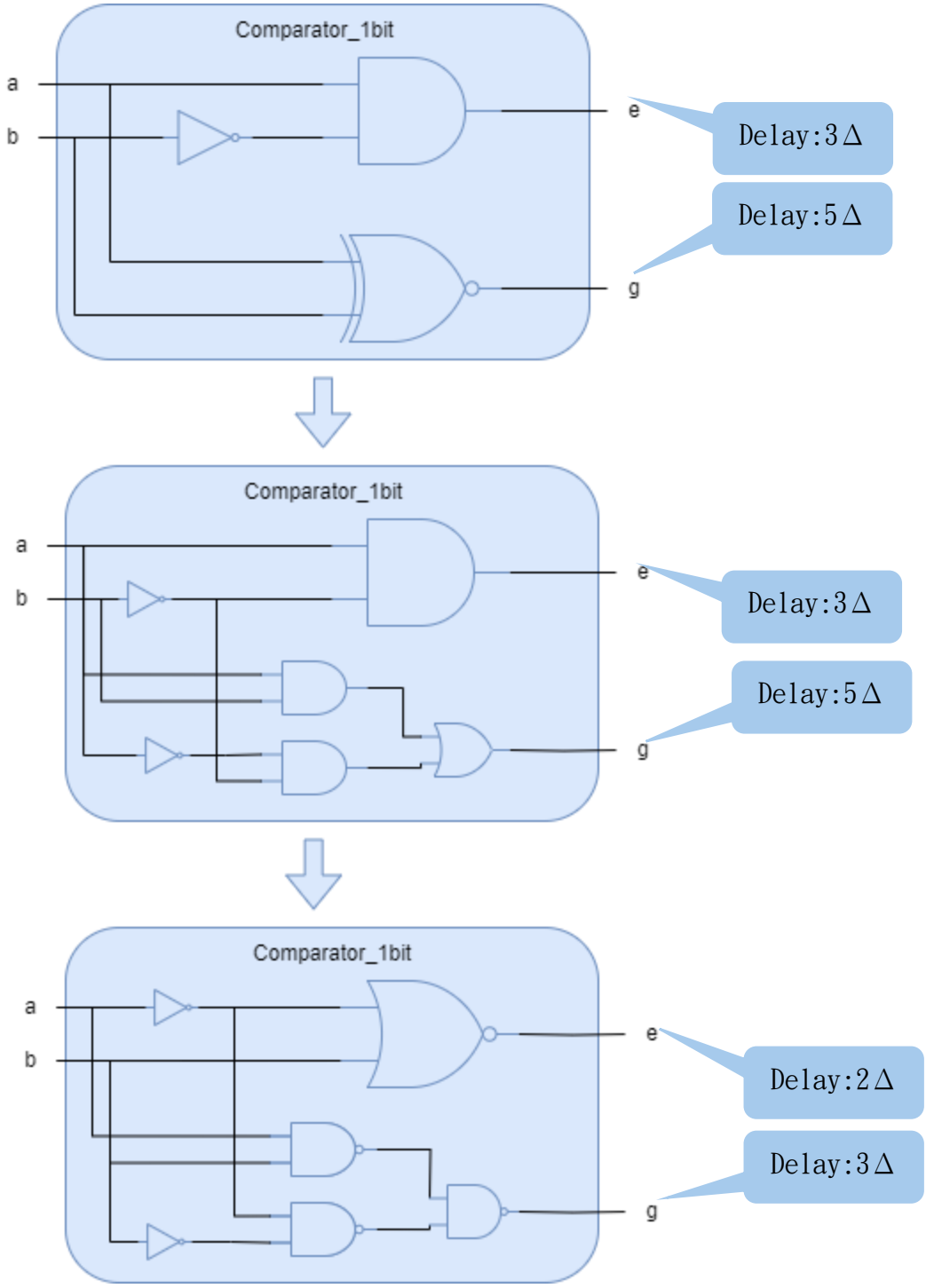


Equal Continuous

使用串聯 and 來完成這項功能，雖然需要一級一級的傳遞延遲相當大，但是面積相當的小，可以有效控制面積。

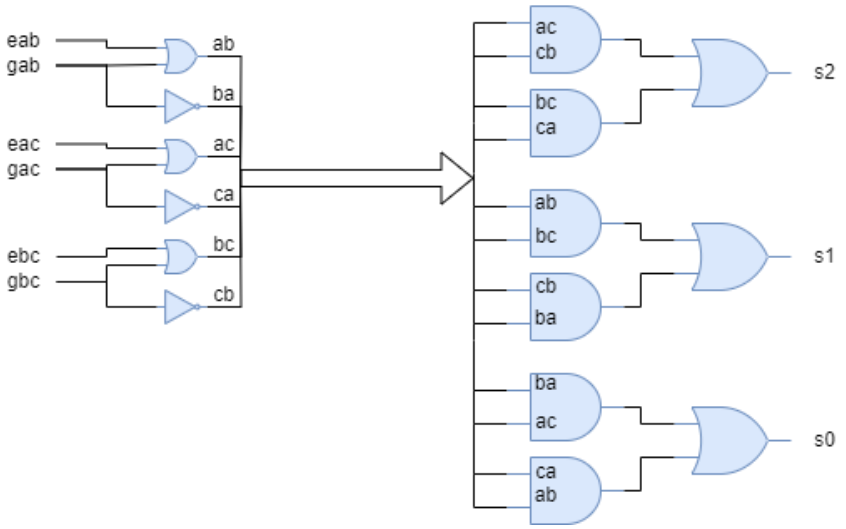


Comparator_1bit



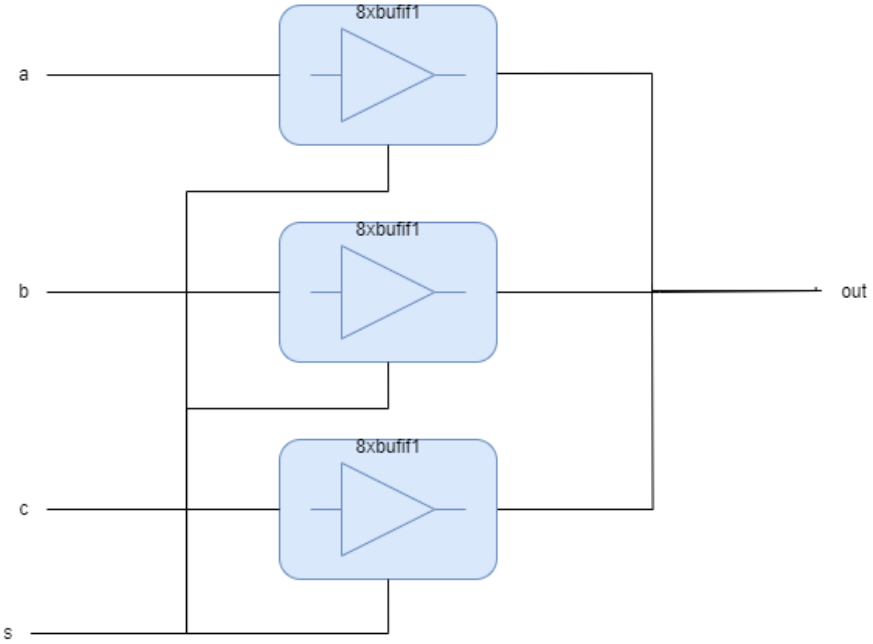
iii. Median_detection

我們有比較結果後就可以運用邏輯運算來判斷是否為中位數，由於我們只前一級只計算了 g(greater)與 e(equal)，當要判斷大於等於時只需要將 g 與 e 做 or 即可，當要做小於等於時用只要計算~g 即可，我們不一定要特別計算 l(lower)來完成。



iv. Selector

將最後結果丟入 Selection，由於最後結果是採用 One Hot encoding 的方式，所以我們只需要把對應的地方輸出就好，使用 buffer 來控制他的輸出，雖然可能會有 buffer 同時對一條線啟動的情況，但是同時起動的情況是訊號相等的，訊號是一樣的並不會造成錯誤。



3. 電路分析：

i. Area

這部分使用 Desgin Compiler 使用 TSMC 0.13um 的製程下去做合成最後計算 Gate Count(Total Area/NAND Area)來評估此電路的大小。

	um^2	Gate Count
NAND	5.0922	1
Median_v1	768.922186	≅151

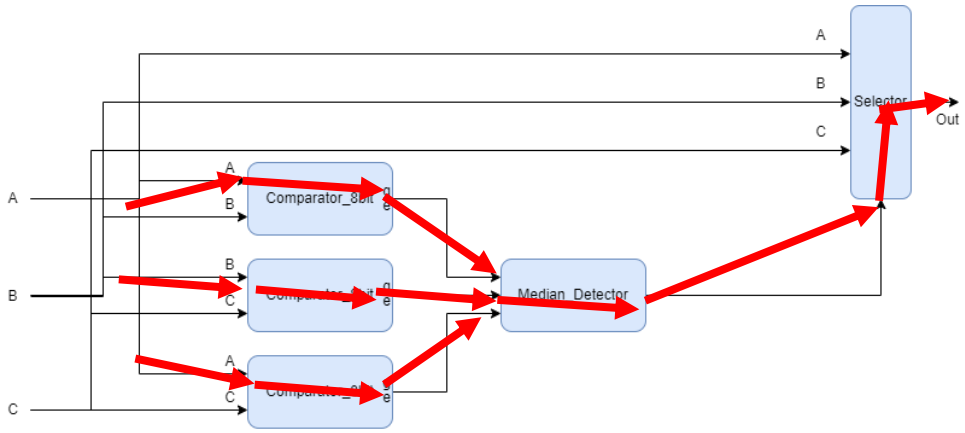
ii. Speed

Max Delay：29Δ

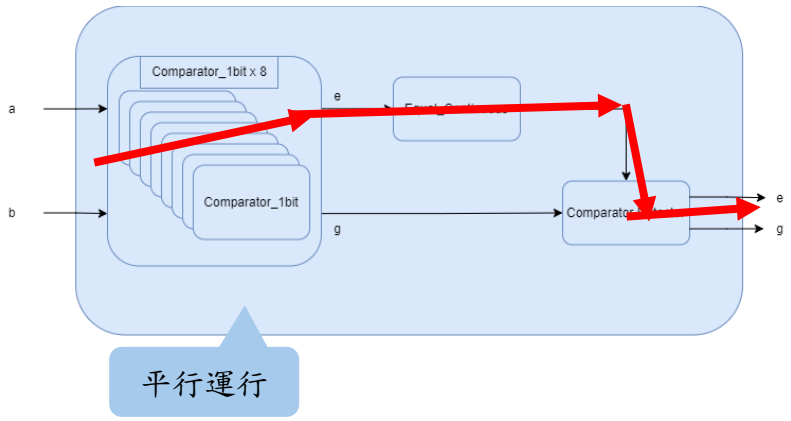
Critical Path(o) :

Comparator_8bit(21Δ) -> Median_Detector(6Δ) -> Selector(2Δ)

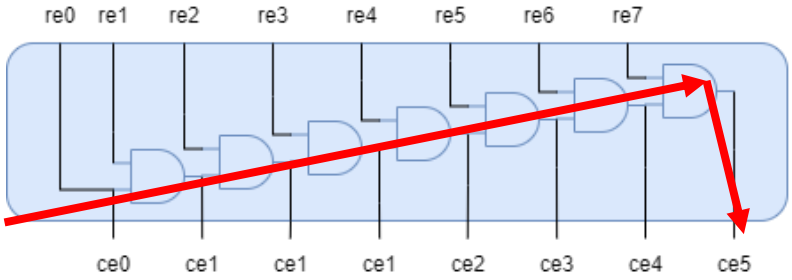
median



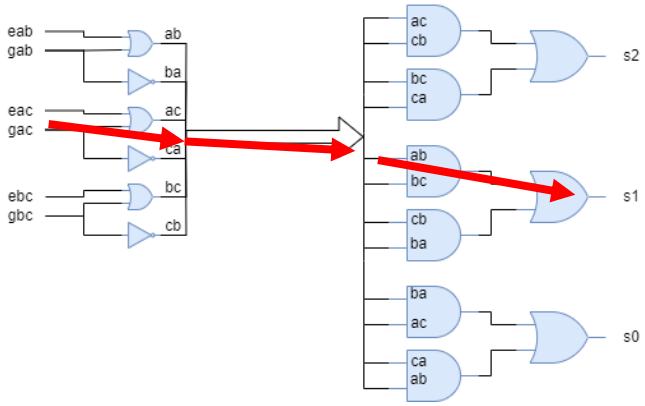
Comparator_8bit



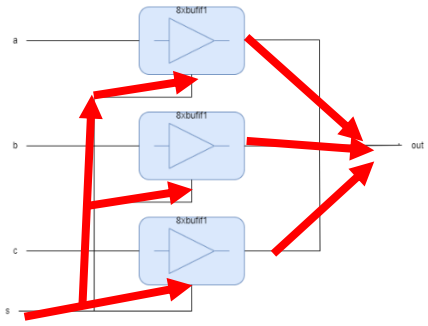
Equal continuous



Median Detector



Selector



三、 中位數 median_v2

(與組員討論出來的作法)

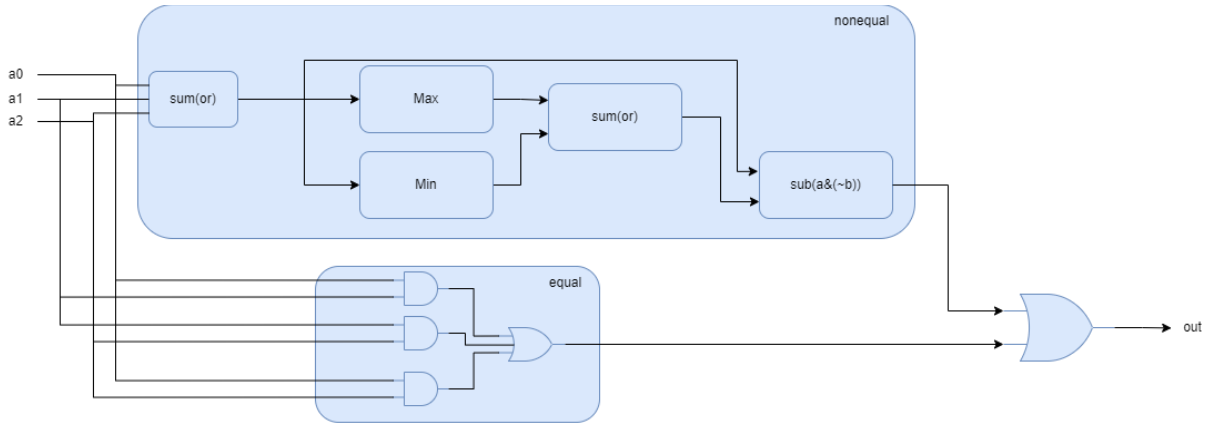
1. 演算法：

因為只有三個數，所以找尋最大與最小，剩下的就是中位數，所以最一開始的想法是把他相加起來扣除掉 max 與 min，但這個做法需要先將輸入 decode 然後做兩次加法，再減除 max 與 min 非常的耗時與費大量面積，所以想到一個方法是在三個數沒有任何數相等時成立的，先把訊號 or 起來(one hot 意義不變只是多個 bit 表示加總數值)，然後找出 max 與 min，把兩者加起來做反向，把先前 or 起來數值去除其餘保留，若有相等情況額外判斷，若出現相等代表一定等於其數值一定是中位數，可以直接輸出該數值，而這種情況發生另一個 or 起來去除掉 max 與 min 的數值皆被扣除數值為 0，所以輸出只需要做 or 運算即可。

out = ((a0 | a1 | a2) & ~(max|min)) | equal

equal = a0&a1 | a1&a2 | a0&a2

2. 電路架構：



i. 根據演算法此電路分為兩個區塊分別為相等與不相等，相等時不相等區塊輸出 0，不相等時相等區塊輸出 0，只需將兩個結果 or 起來即可。

ii. 不相等時我們首先第一步把三個數值加再一起(or)，然後找出 max 與 min，由於是 one hot encoding 的關係，所以只需使用兩個優先編碼器(使用 casez 合成)，即可找出最大與最小數值，接下來加總後減去即可邏輯運算為 sum&~(max|min)。

iii. 相等時判斷也非常簡單僅需考慮 a0==a1 、a1==a2 與 a0==a2，只要其中一個狀況發生即成立，依照下方布林代數即可完成計算。

$$equal = (a0==a1) \mid (a1==a2) \mid (a0==a2)$$
$$= a0\&a1 \mid a1\&a2 \mid a0\&a2$$

3. 電路分析：

i. Area

這部分使用 Desgin Compiler 使用 TSMC 0.13um 的製程下去做合成最後計算 Gate Count(Total Area/NAND Area)來評估此電路的大小。

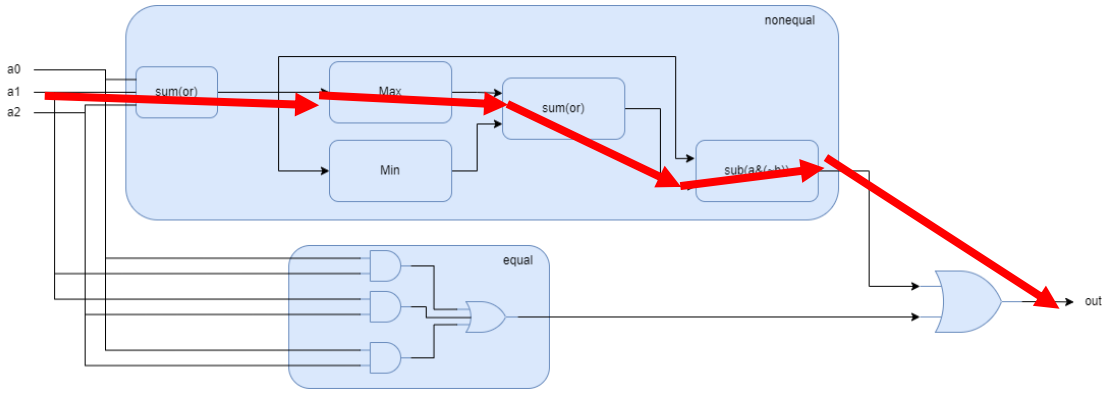
	um^2	Gate Count
NAND	5.0922	1
median_v1	768.922186	≅151
median_v2	475.272001	≅93

ii. Speed

Max Delay：22Δ

Critical Path(out) :

Sum (2Δ) -> Priority encoder (7*2Δ) -> Sum(2Δ) -> Sub(2Δ) -> OR(2Δ)



iii. Version 1 vs Version 2

v2 這個做法面積小於 v1，且速度也比 v1 快，看似比 v2 比 v1 好上不少可是，但當今天要拿這個電路充當一般 8bit 的篩選時，這時候 v1 就派上用場，v2 使用情況只侷限在 One Hot encoding。

	速度	面積	其他
Median_v1	慢	大	可充當一般 8bit 比較器
Median_v2	快	小	

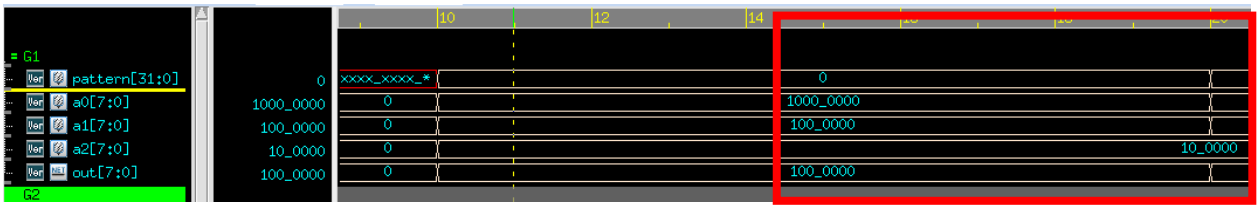
4. 電路模擬

這個電路總共會有 4096 種輸入，相較起上一個電路的測試大很多，所以這裡只輸入幾種 corner case，分別為以下幾種。

Pattern 1. a0 >= a1 >= a1

a0 = 10000000 a1 = 01000000 a2 = 00100000

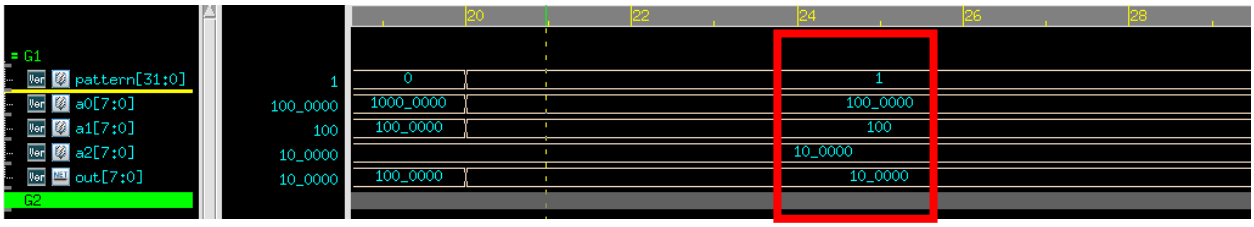
out = 01000000



Pattern 2. a0 >= a2 >= a1

a0 = 01000000 a1 = 00000100 a2 = 00100000

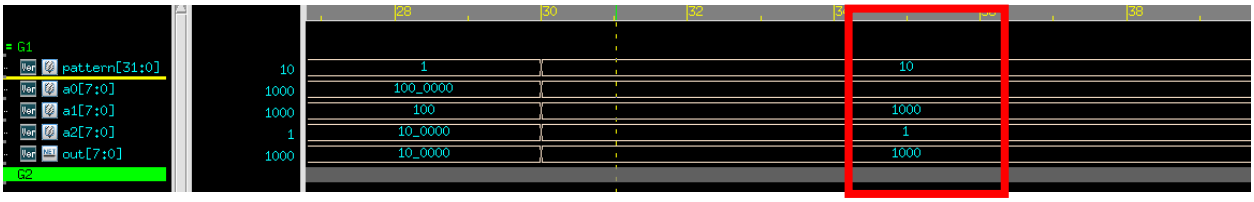
out = 00100000



Pattern 3. a0 與 a1 相等時

a0:00001000 a1:00001000 a2:00000001

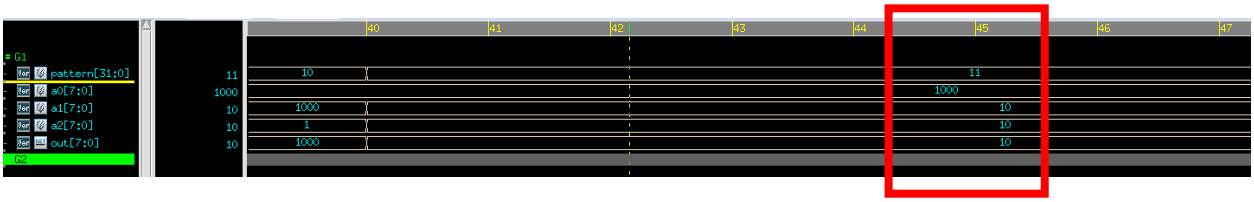
out = 00001000



Pattern 4. a1 與 a2 相等時

a0:00001000 a1:00000010 a2:00000010

out = 00000010



四、心得

這次的作業確實更有挑戰性，我花了比上次更多的時間來完成。我和組員一起討論，試圖找到更好的解決方案，尤其專注於面積和速度的優化。在這次的作業中，我嘗試使用 Design Compiler 來進行合成，以計算 Gate Count 來評估性能。同時我也採用了 Behavior 和 Dataflow level 描述組合邏輯，但為了確保結構與我心目中的樣子是一樣的，我不斷地打開 Design Compiler 查看 Schematic，確認是否與想法一致。然而很多次並不如我所想，只好一一手打每個邏輯閘，耗費了相當多的時間。

另外，以往的 Testbench 通常只是隨便丟幾個數值下去檢測，但這次由於某些電路採取了不同的架構，容易在某些點出現問題。例如，在第二題中，我特地增加了相等的情況來觀察輸出結果，並檢查是否如預期地執行了相等所執行區塊，以及不相等輸出的區塊是否輸出為 0。這讓我更加明白了設計 corner test 的重要性。

這次作業的困難點在於演算法。以往修過的其他數位硬體課程很少有機會涉及到演算法的更改，通常只是在架構上稍作調整以滿足規定的面積或速度要求。然而，這次情況大不相同。要寫出不一樣的演算法來製作電路，這對我來說是一項極大的挑戰。我希望以後自己可以變得更聰明一點，並提出更多不同的演算法來進行設計和比較。