積體電路設計 HW4

B07901087 電機四 蔡沛洹

(a)

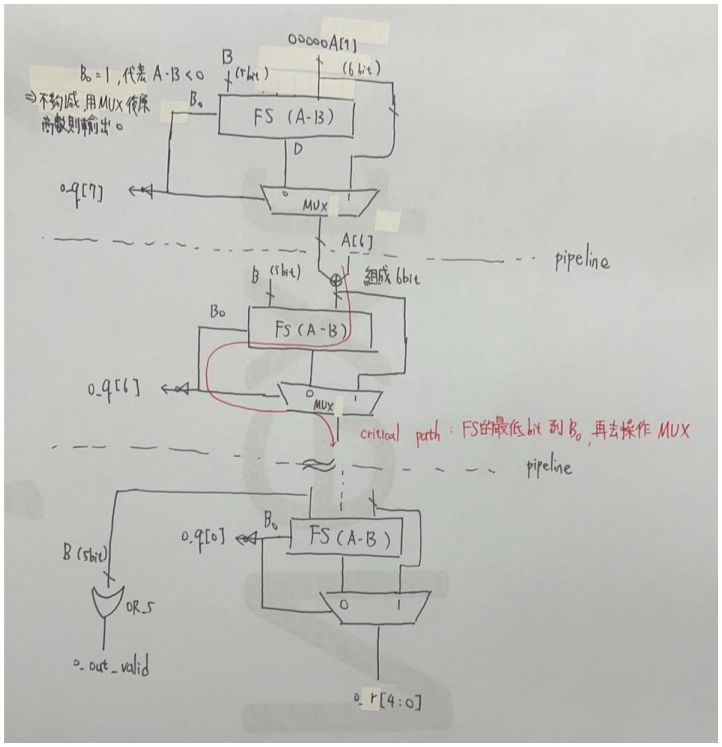
Minimum cycle time: 3.527ns

Strategy: pipeline

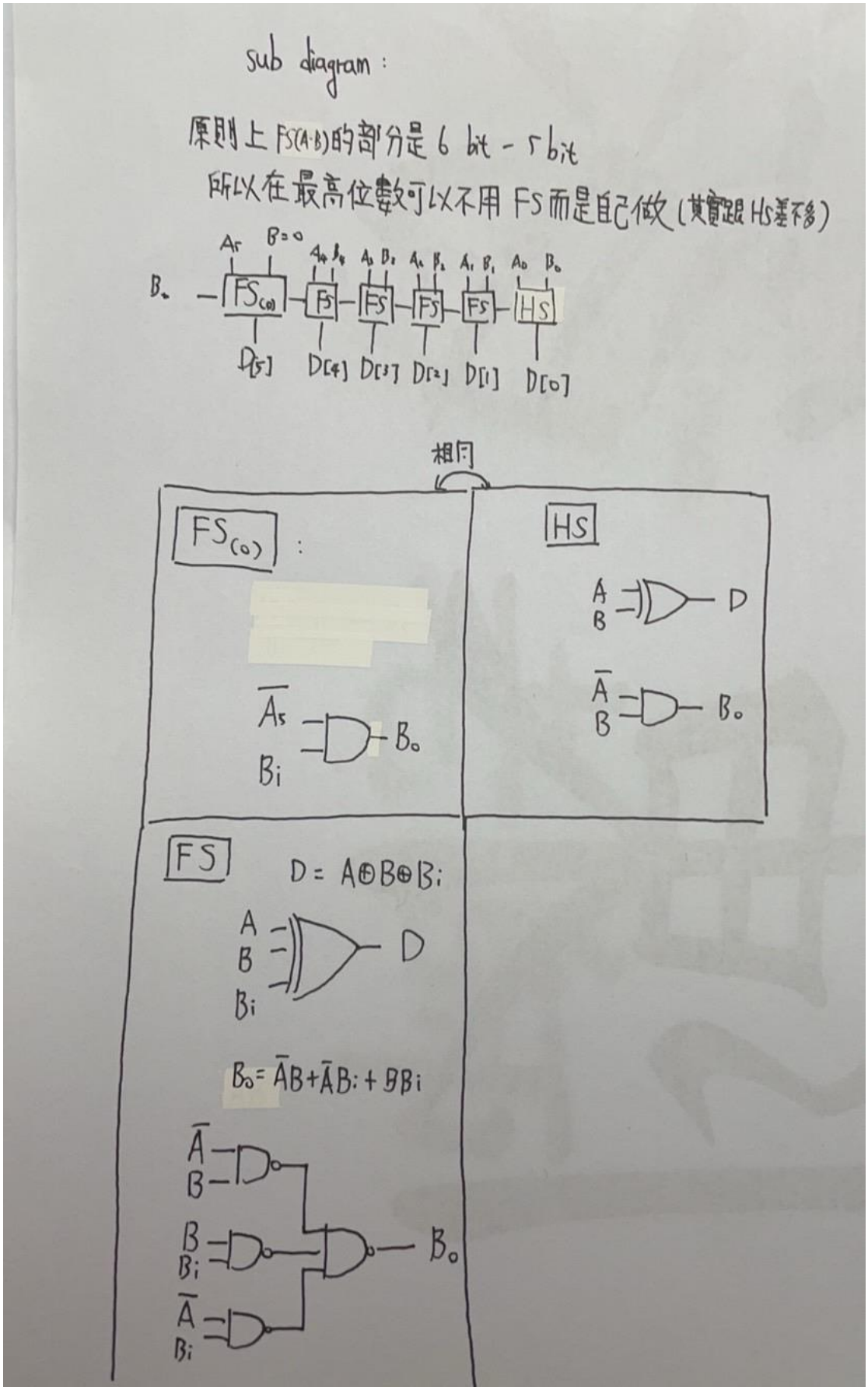
```
================================================
              Simulation passed
================================================

================================================
                   Summary
================================================
        Clock cycle:           3.527 ns
        Number of transistors: 4452
        Total excution cycle:  1006
        Correctness Score:     40.0
        Performance Score:     15796417.2
================================================
```

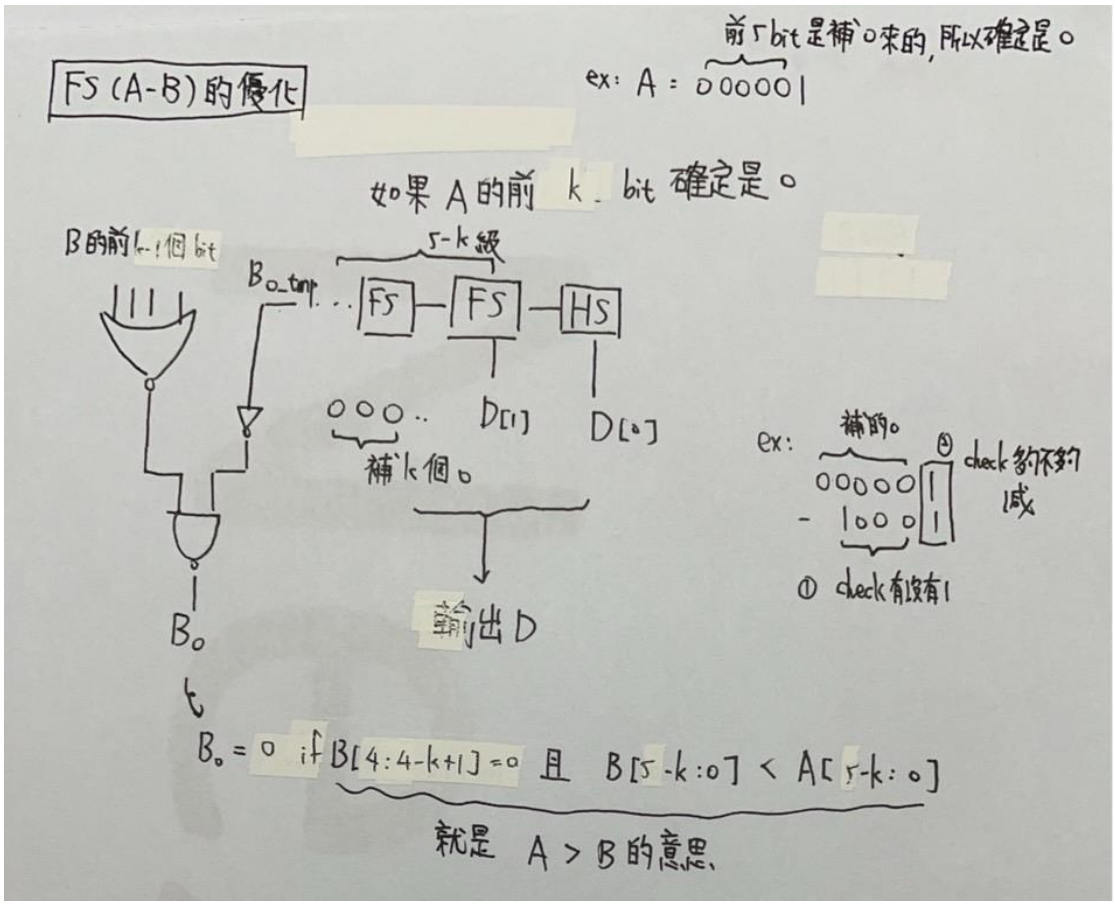
(b) **Circuit diagram (critical path 放在 top level)**

Top Level Design

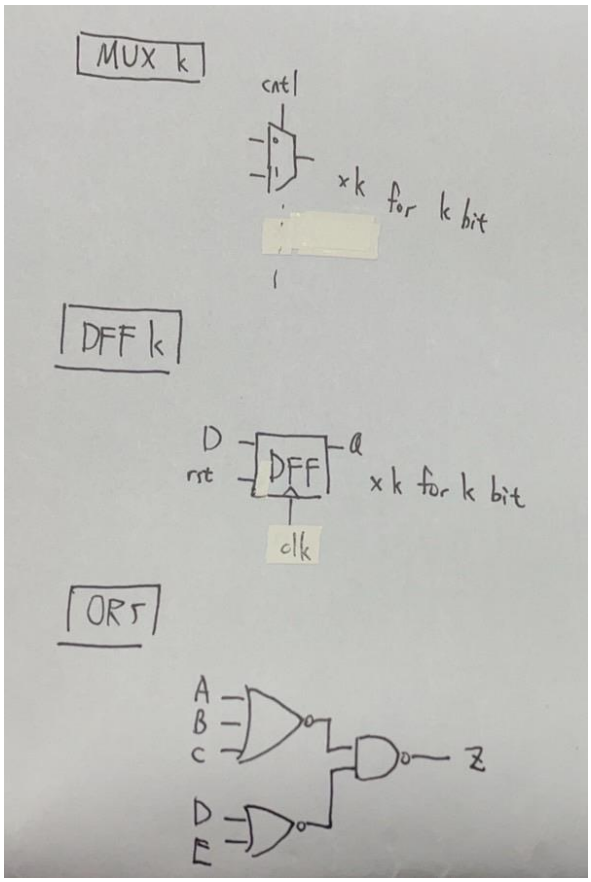


減法器


sub diagram :
原則上FS(A-B)的部分是6 bit - 5 bit
所以在最高位數可以不用FS而是自己做(其實跟HS差不多)
FS(0) :
HS
相同
FS
D = A⊕B⊕Bi
Bo = ĀB + ĀBi + BBi


前5bit是補0來的，所以確定是0
FS (A-B)的優化
ex: A = 00001
如果A的前 k bit 確定是0
B的前k-1個bit
5-k級
B0_tmp
FS
FS
HS
000.. D[1] D[0]
補k個0
ex:
補的0
00000
- 1000
check夠不夠減
① check有沒有1
輸出D
B0
B0 = 0 if B[4:4-k+1] = 0 且 B[5-k:0] < A[5-k:0]
就是 A > B 的意思

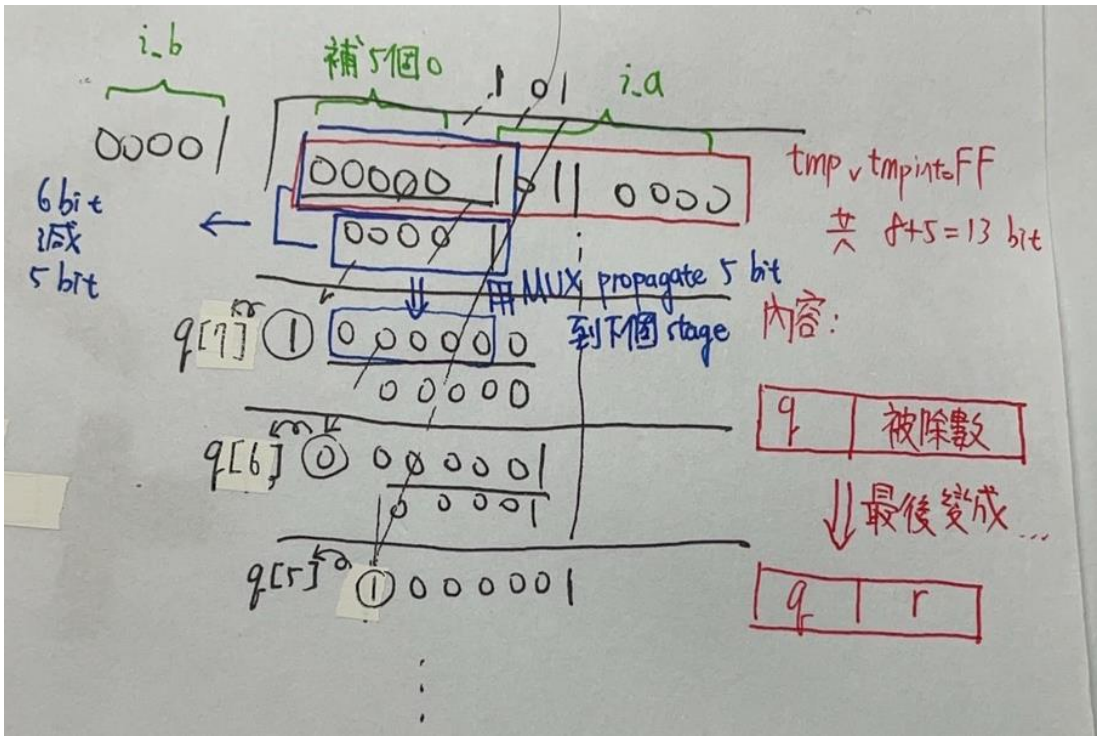
其他電路


MUX k
ctrl
xk for k bit
DFF k
D
rst
DFF
Q
xk for k bit
clk
OR5
A
B
C
D
E
Z

(c)

Introduce your design.

我用長除法去實做除法器，使用 pipeline 的方式，所以每一層就相當於長除法的一次相減，一共會有 8 層(但是最後經過我優化後剩下 7 個 stage)，示意圖大致如下:



主要是每個 stage 配合長除法的概念，做一個 6bit-5bit 的減法器，做完之後 6bit 的最高位可以拿掉，所以剛好那個位置就可以記除數，所以 DFF 的用量基本上是 {i_b, i_in_valid, q、r 和中間產物}，總共是 19 個 bits（q、r 和中間產物我在.v 檔中命名是 tmp 或是 tmpintoFF）。然後 MUX 是用來判斷有沒有要復原用的（如果 A-B，然後 A＜B，那就要復原），基本上做完除法後我們都只需要 propagate 五個 bit 到下個 stage，所以都用 MUX5。

在這之中我們可以做一些優化，主要是前面補 0 的部分是我們已經知道的，所以如果照常用 FF 去記它們或是用減法器去減會缺乏效率。主要在做法上多改了以下幾點：

1. FF 部分可以不用去記補 0 的那幾個 bit。
2. 減法器、MUX 的部分不須用正常方式去處理已知是 0 的部分，可以用更簡單的 circuit 達到相同效果。

做完之後可以發現 stage1, stage2 的 critical path 較短。

3. 把 stage1, stage2 縮成一個 stage。
4. o_out_valid 的部分不用 i_in_valid 判斷，可以直接抓除數做 OR(因為除數不可能是 0)來判斷現在 data 有沒有到了，所以可以把標準的 19 bit DFF 縮成 18 bits

How do you cut your pipelined or recursive design?

正如上面所說，我每次除法相減產生出一個商的 bit，這就是一個 pipeline stage

How do you improve your critical path and the number of transistors?

我最開始是做正常的長除法，即使補零的部分也用 FS 去相減、用 D Flip Flop 去存不必要的結果，導致 transistor 和 critical path 的部分有些許浪費。transistor 數是 5800 多，performance score 差不多是 22 或 23 開頭，我依序做了以下的改進：

1. Flip Flop 縮減：

我先將開頭補 0 的部分不用 flip flop 存，反正傳到下一個 stage 也只會是 0，這個可以縮 area，這樣 performance 可以進步到 20 出頭。

2. 減法器、MUX 的化簡：

一開始因為是處理被除數補 0 的部分，所以減法器不用那麼多個，例如說:

11001100/11001

被除數補 0 會變成：

0000011001100/11001

在除法器中，我們第一個 stage 會處理：

0000011001100 .

- 11001 .

我們只需要一個 HS 去處理最後一個 bit 的相減，因為被除數的前面都是我們補的 0，所以減號成立(也就是商數是 1)的情況就是在 (a)除數的前四個 bit 都是 0 且 (b)被除數和除數 HS 後 borrow out 是 0。

剩下也都是同樣的道理，就只是把減法器的部分變成 2 個 bit, 3 個 bit, 4 個 bit, …，然後前面除數 0 的判斷的部分 bit 數依序遞減。這個可以縮 area 還有 critical path 的長度，Performance score 差不多進步到 18 出頭

3. stage1, 2 的合併：

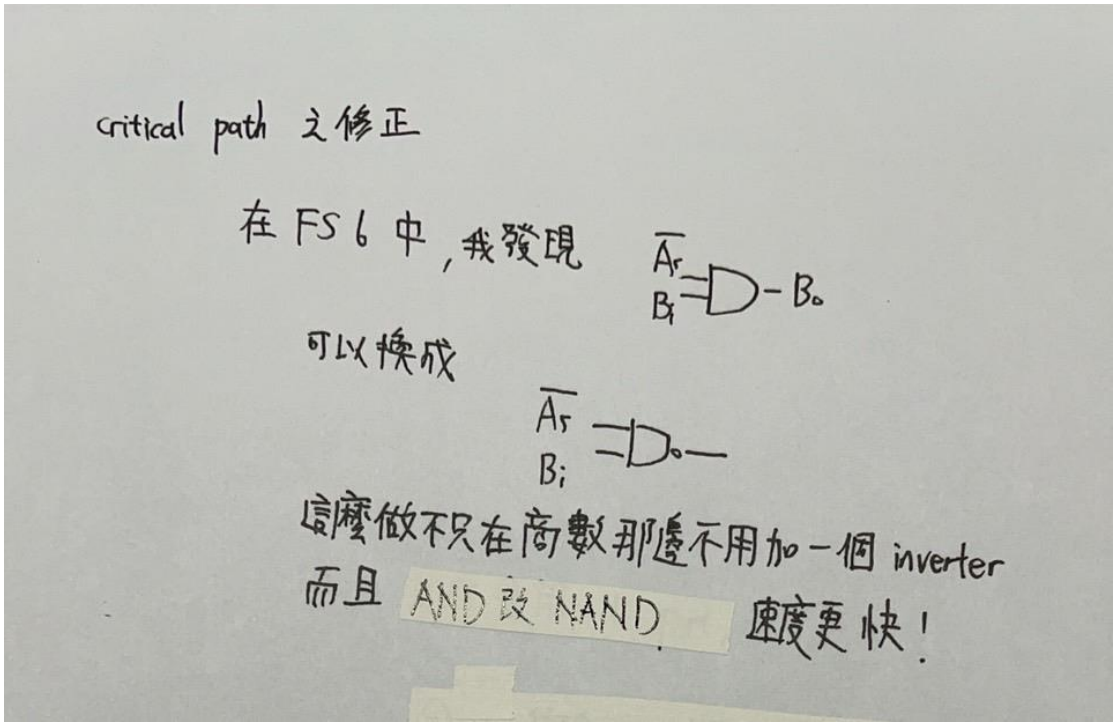
在做完第二步之後，因為電路的 critical path 都是在減法器的部分，所以 stage1, 2 的部分 critical path 大幅縮減，我就把這兩個合在一起做成一個大的 stage，也不會影響 cycle time，還省了一排 Flip Flop。Performance 可以進步到 16 開頭。

4. output_valid 不用 pipeline 傳：

我原本是把 in_valid 用 pipeline 傳到 out_valid，但是因為除法應該不會有除數是 0 的情況，所以我後來就不用 pipeline 傳 valid signal，而是用 OR5 gate 在最後一個 stage 偵測除數是不是非零，如果是非 0 就把 out_valid 弄成 1。這個只有在每個 pipeline 省一個 flip flop，所以進步不多，最後還是在 16 出頭。

5. FS 的輸出還有 MUX 位置的互換：(比較不重要)

我發現減法器的 output 直接輸出商數也可以少一些 gate 和 delay，所以對 critical path 的減法器做了一些微調。這讓我把 cycle time 又壓了一些，在最後 performance 成功到 15 開頭。



How do you trade-off between area and speed?

Area 的部分我一開始覺得可以先縮減 Flip flop 的 transistor 數，所以我打算把 pipeline 的 stage 兩兩合併，藉由犧牲 cycle time 的方式來減少中間的 Flip Flop 數目。但是這樣做經過我的實驗並沒有比較快，因為 cycle time 會變到快兩倍，但是減法器依然有佔一定比例的 transistor 數量，所以這個做法並沒有比較好，最後我還是選擇做正常的 pipeline 就好。

Compare with other architectures you have designed.

我只有試著做 pipeline 縮減的部分，結論就是還是維持原本的 pipeline 就好，但是 stage1 和 stage2 的 critical path 很短，所以可以把他們併在一起。

另外我覺得用 FS 的 Bo 去 control MUX 好像有點太長，所以我有試著做過直接用 comparator 判斷除數(5bit)被除數(6bit)的大小來控制 MUX，最後做出來發現面積耗太大了，cycle time 也沒有省多少，所以就不採用這個做法。