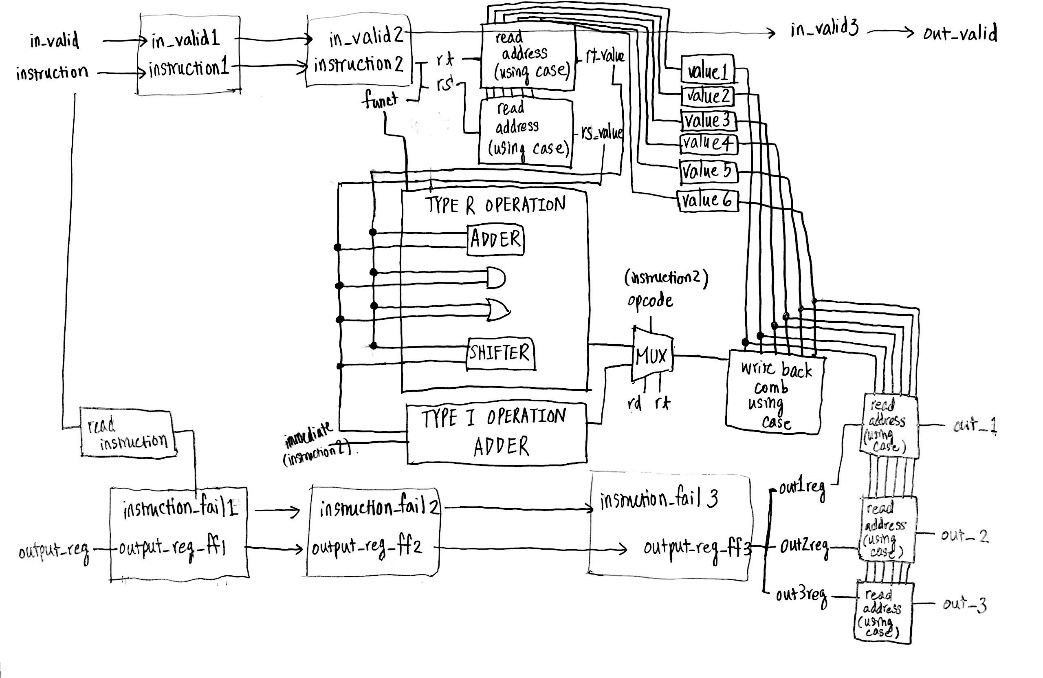
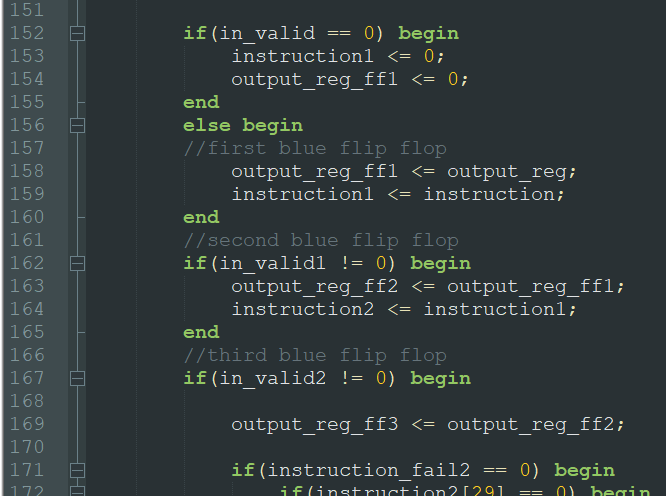
**Block Diagram**

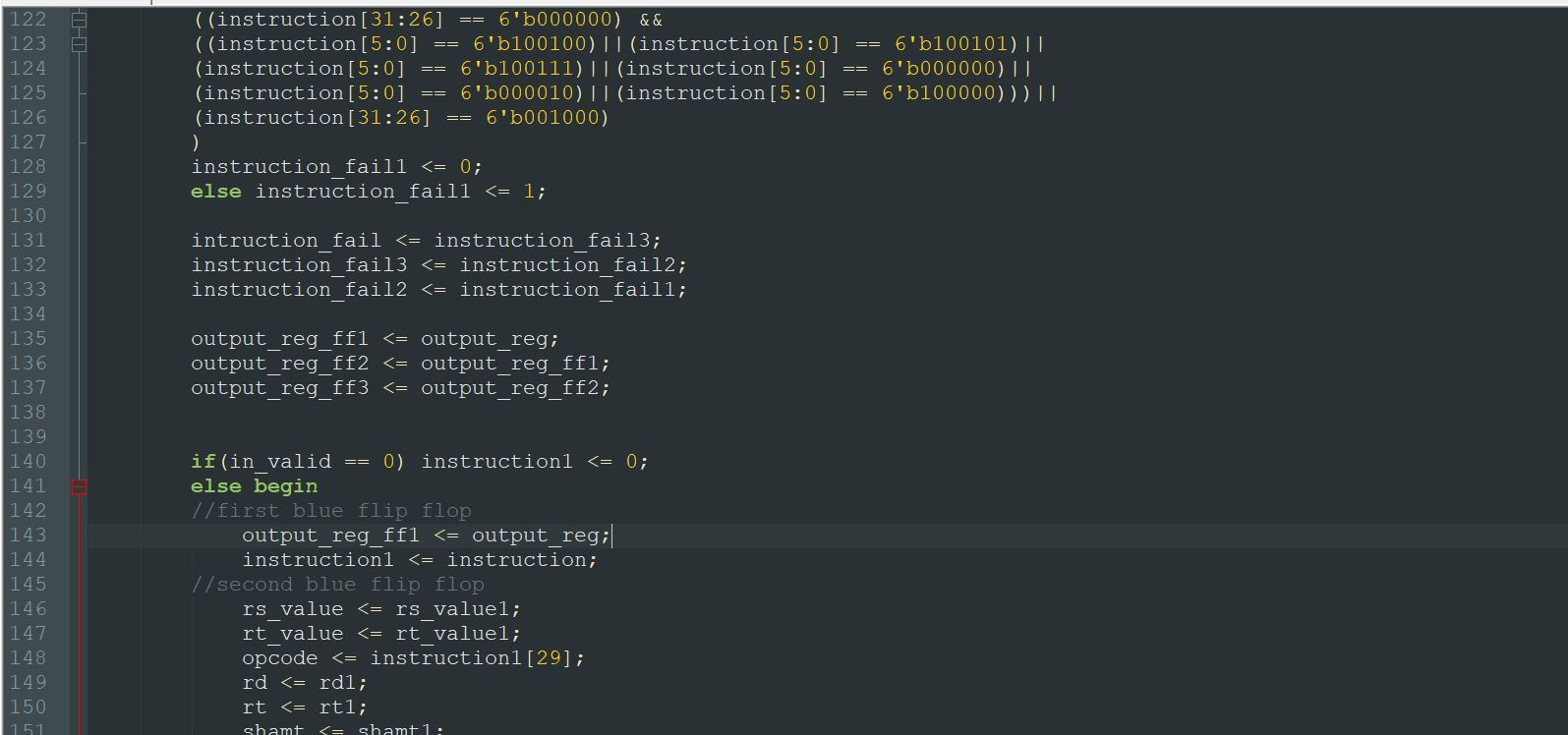


**設計過程錯誤和debug**

1. **判斷的in\_valid依據錯誤**可以由上方的block diagram看出，本次為了達成pipeline的效果，所以創了三個in\_valid的register做傳遞 (in\_valid1, in\_valid2, in\_valid3)，而同時instruction、output\_reg也要進來並且傳遞，而每一層instruction、output\_reg的flip flop要不要傳到下一個flip flop也必需看in\_valid是否為1，不是的話就要停止傳遞。但是每一層要判斷的in\_valid register是不一樣的，當初犯的錯誤就是把所有的傳遞都寫在in\_valid的判斷裡面，導致in\_valid一清零，中間傳遞到一半的值就停滯不出來了。所以應該要依據「前」一層級的in\_valid做判斷(因為in\_valid是用non\_blocking傳的)

如下圖:



1. **output\_reg的傳遞沒有寫入in\_valid的判斷條件中**  
   若沒有寫入in\_valid的if-else裡面，會在instruction給完後繼續讀取output\_reg的address，然後依然去取值，這樣輸出當然就會是錯的。所以必需將output\_reg的pipeline寫在in\_valid的判斷裏頭如下圖。

**程式、時間和面積的優化過程**

這次的修改主要分成兩部分:

A. 模組的廢除

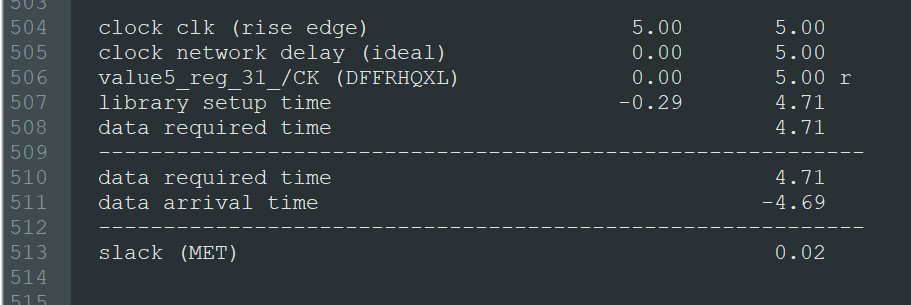
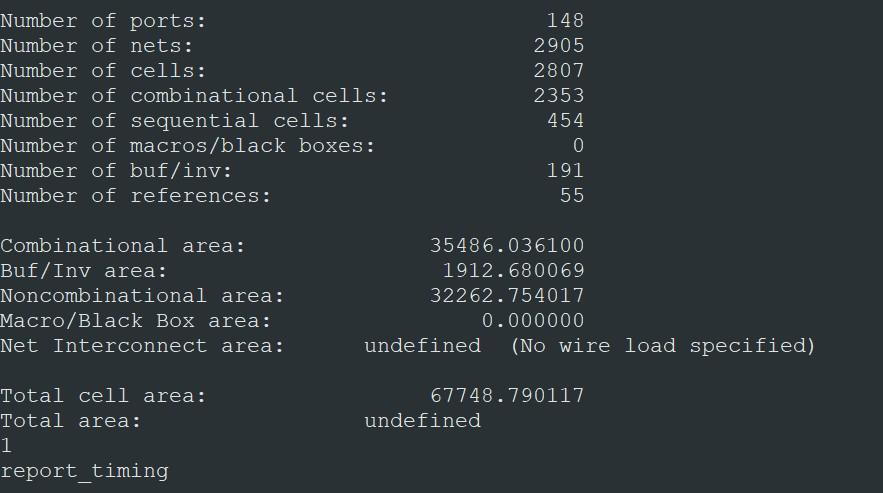
1. 在第二個flip flop才做decode的動作

在最初的設計中，我讀address這件事是另用一個模組來實現，原本的想法是因為很多動作，像是rs, rt, 讀取值、output\_reg為out1, out2, out3選取輸出，都是在做同一件事，所以用一個模組，寫一次case再接上六個變數就可以重複宣告，可以少寫很多行code。但是，code行數的多寡不見得代表面積和slack的增減!

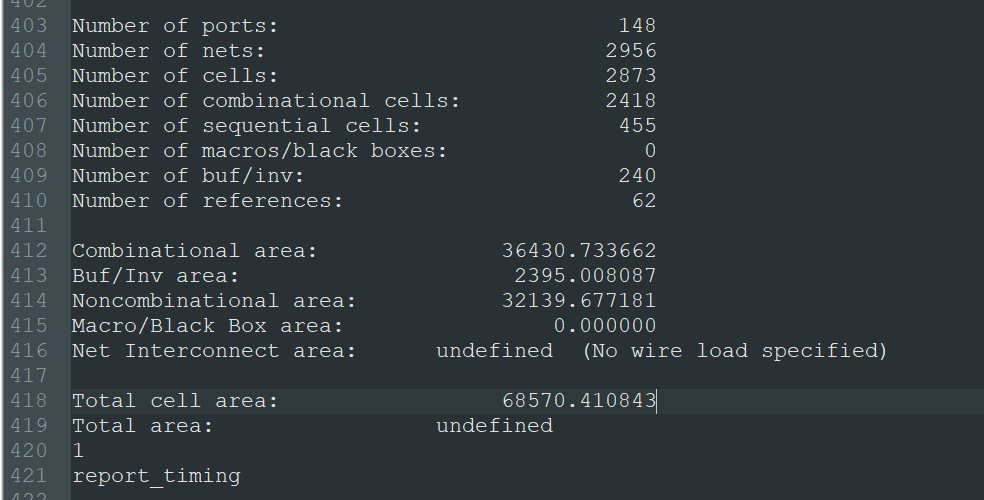
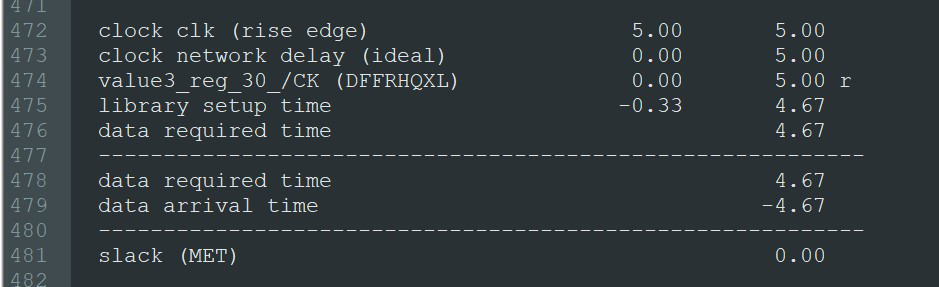
我依序嘗試了以下更改組合:

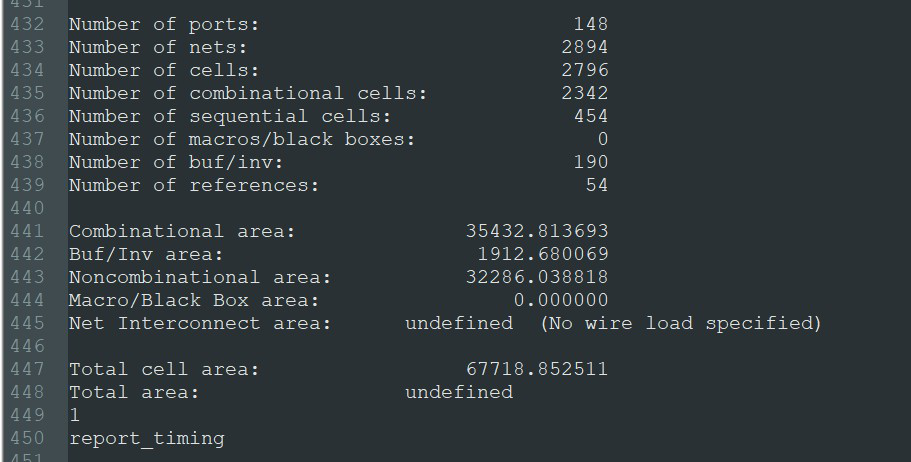
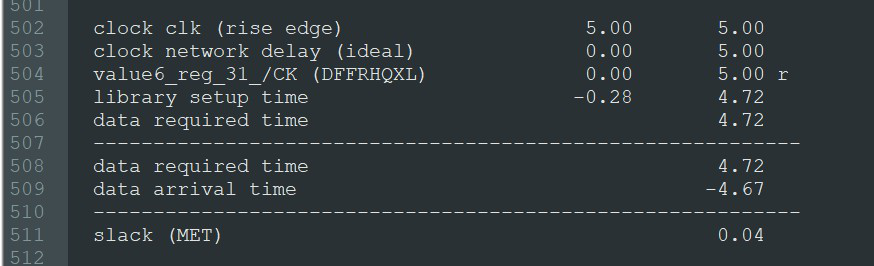
1. 一開始我在還保留模組的設計下先做B.，結果面積變大、slack變小；
2. 再來我把模組的寫法改成多個always\_comb寫在主模組裡面，也就是不另宣告模組，但是decode依然在第一個flip flop後做，雖然code的行數變多，但結果果真有好一些
3. 把模組廢除，寫多個always\_comb、decode到第二個flip flop後做，面積小最多，但slack比2. 小了些

我最終保留3. 的設計，原因是減掉幾千的面積感覺還是比少了零點零幾的slack來的優質，以下是各次跑synthesis的截圖:  
原始設計 (沒做A、沒做B)



1. 做B沒做A

1. 做A沒做B  
      
   
2. 做A做B (最終保留的設計)  
   