計算機結構Computer Architecture HW1 report

B03901022 卓伯鴻

1.ALU

(1) RTL

我的設計概念是先做出每一個運算結果，再透過mux選擇的功能，使得ctrl可以控制output丟出正確的運算結果。這個設計因為進行運算結果時，在時間順序上是平行進行的，所以只會消耗掉部分的暫存記憶空間，而不會拖延到整體的運算時間。

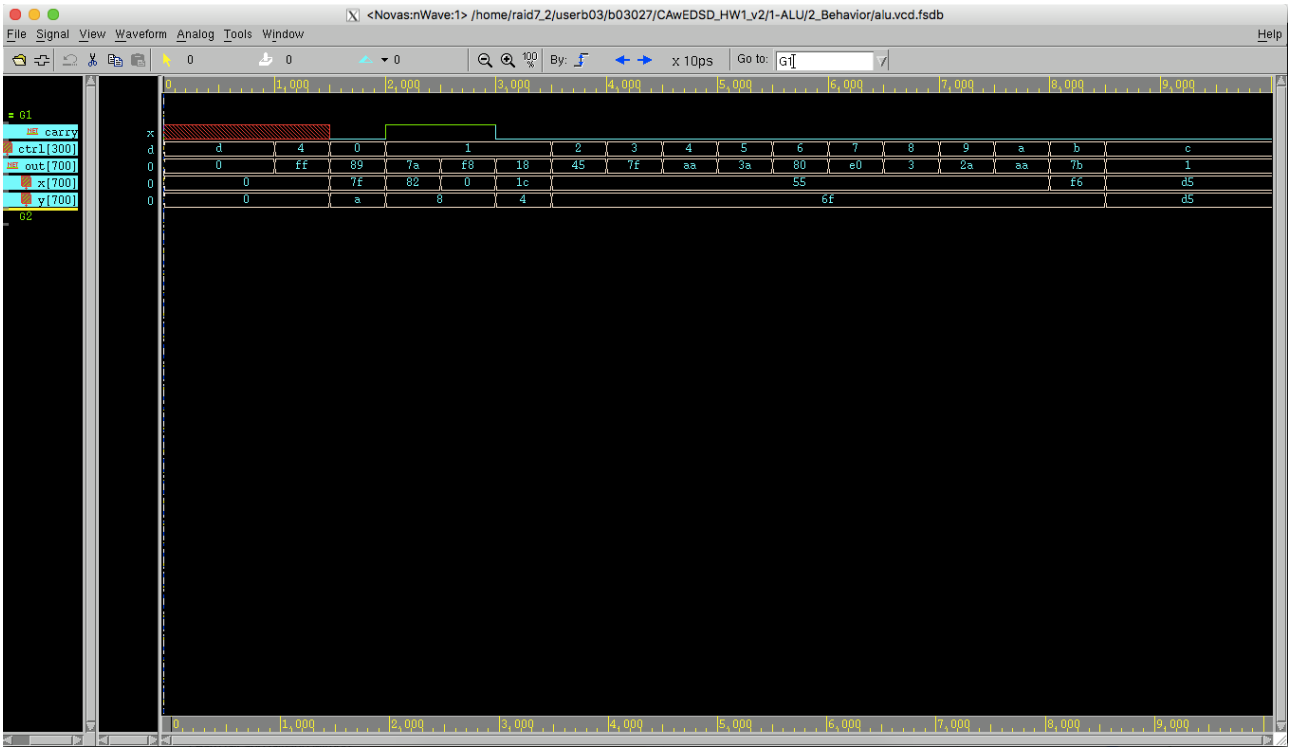
而在testbench中，我對每一種運算都給訂了一個固定的測資，並寫了if else敘述來判斷該運算結果out是否正確，如果正確則會在終端機display出“pass”的字樣，如果錯誤的話則會印出”fail”的字樣，並輸出標準答案及經由ALU所產生的錯誤輸出。

(2) Behavior Level

由於在行為描述的檔案中，可以直接使用if else敘述，所以我就使用了無條件可以隨時監控的always敘述，並在輸出端持續的輸出該種運算(ctrl控制的)運算結果，而此一結果會隨著ctrl跟input的更動而改變。

此處之test bench內容與RTL的test bench內容近乎一致，為不同處只有在model使用的不同。

(3)圖檔說明



2.Register File
(1)verilog
在register的write operation中，因為具有synchronous的性質，所以控制write的always敘述就限制只有在clock訊號的positive edge時才准許進入，而進入之後則檢驗Write_ENable訊號是否為1，若為1則繼續檢測register的編號是否不為$r0，若不為$r0才將busW端的值寫入register中。

在register的read operation中，具有asynchronous的性質，所以他的always敘述是無條件可執行的，而在read register的過程中，先使$r0中存的值為constant zero，在將RX和RY指定的register中存的值匯出到busX及busY中。

3.Simple Calculator
(1)verilog
在這個檔案中，因為大部分的計算都在前兩個檔案的module中做完了，所以我就直接引用前面兩個檔案，並把module的腳位接對之後，僅用一個無條件進入的always敘述來決定要使用register中存的數字繼續運算還是輸入的DataIn來進行運算。

在這個程式的debug中我遇到最大的問題就是，檔案之間的連結，因為有部分的測資是在前面的檔案中測不出bug的，所以在使他能compile過這件事琢磨了不少時間，再者，就是在前面檔案對於always@(expression)的使用，有一些bug就是因為測資可能會是空的，所以我一開始使用always@(測資改變時進入)就會產生一些don't care，而改成always@(*)的無條件進入之後，就修復bug了。