國 立 台 灣 科 技 大 學

計 算 機 組 織

指 導 教 授：陳雅淑 教 授

**------------------------------------------------------------------------**

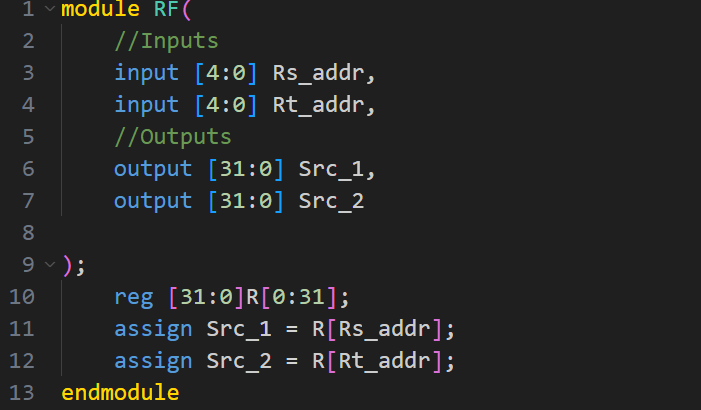
計算機組織作業一報告

班 級 ： 四電機三甲

學 生 ： 呂佳祐

學 號 ： B11107055

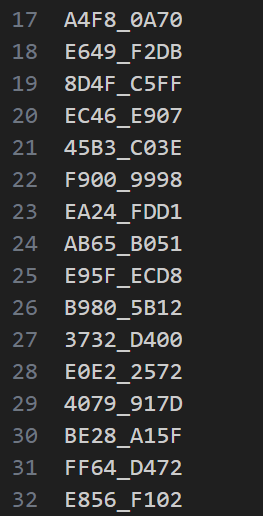
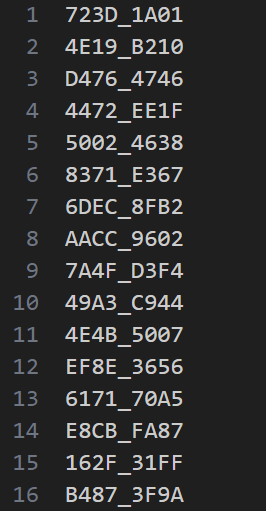
1. 32-bits Read Only Register File
2. 主程式



RF.v程式碼

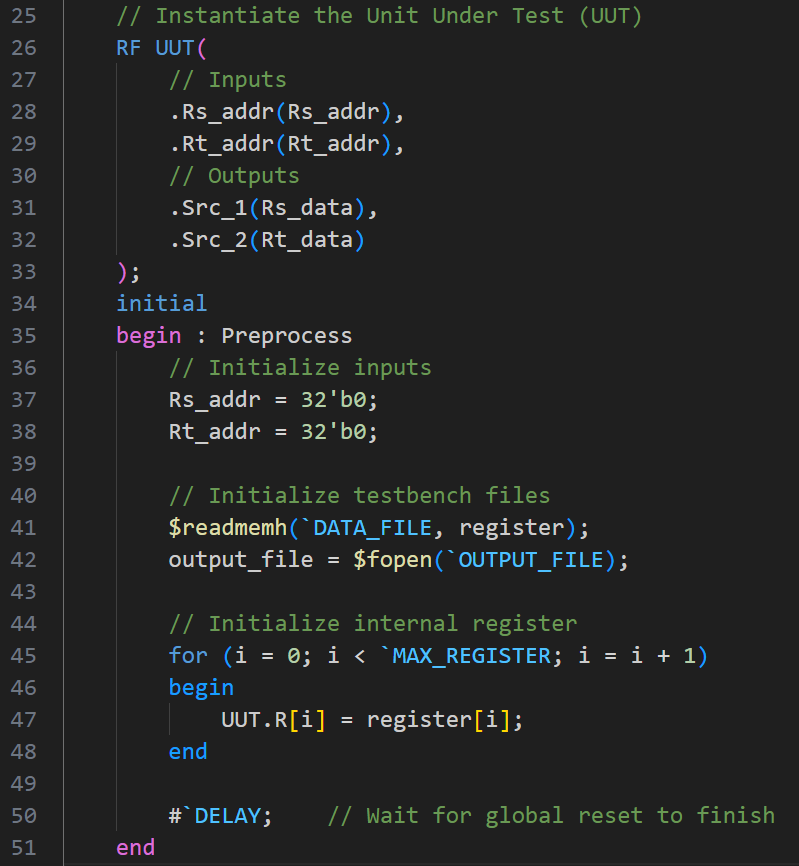
RF唯讀暫存器寬度大小為32bit、深度為32bit。將輸入的Rs\_addr、Rt\_addr作為索引，透過陣列表示法，讀取對應暫存器內的值。因為Register File內有32(25)個暫存器，所以Address大小宣告為5個bit。而各暫存器內大小為32bit，所以Src\_1、Src\_2大小宣告為32個bit。

1. Testbench and Simulation

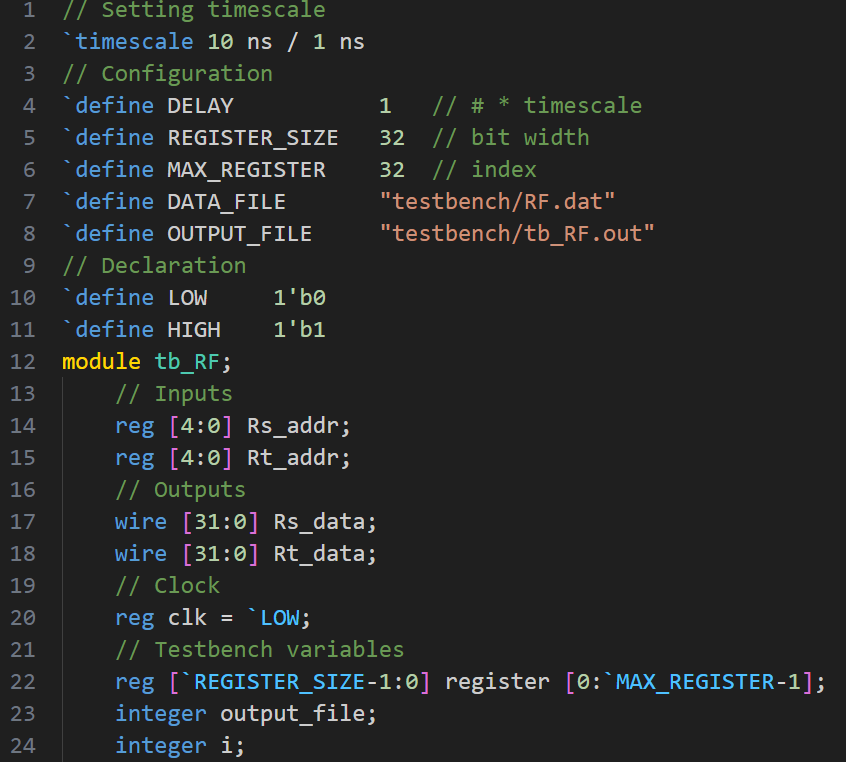


RF.dat儲存Register File內各暫存器內資料

設定時間單位10ns、時間精度1ns。



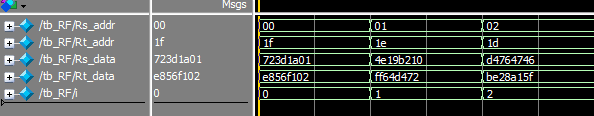
tb\_RF.v連接RF.v module與預處理

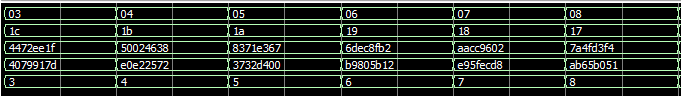


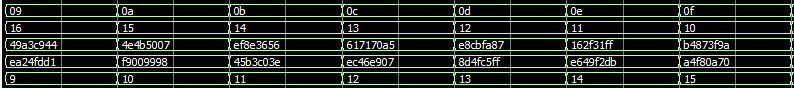
tb\_RF.v初始化設定

與主程式module連接。將RF.dat內資料讀取並存入register變數內，再將資料轉移至Register File裡，之後從Register File讀到的就是RF.dat內的資料。

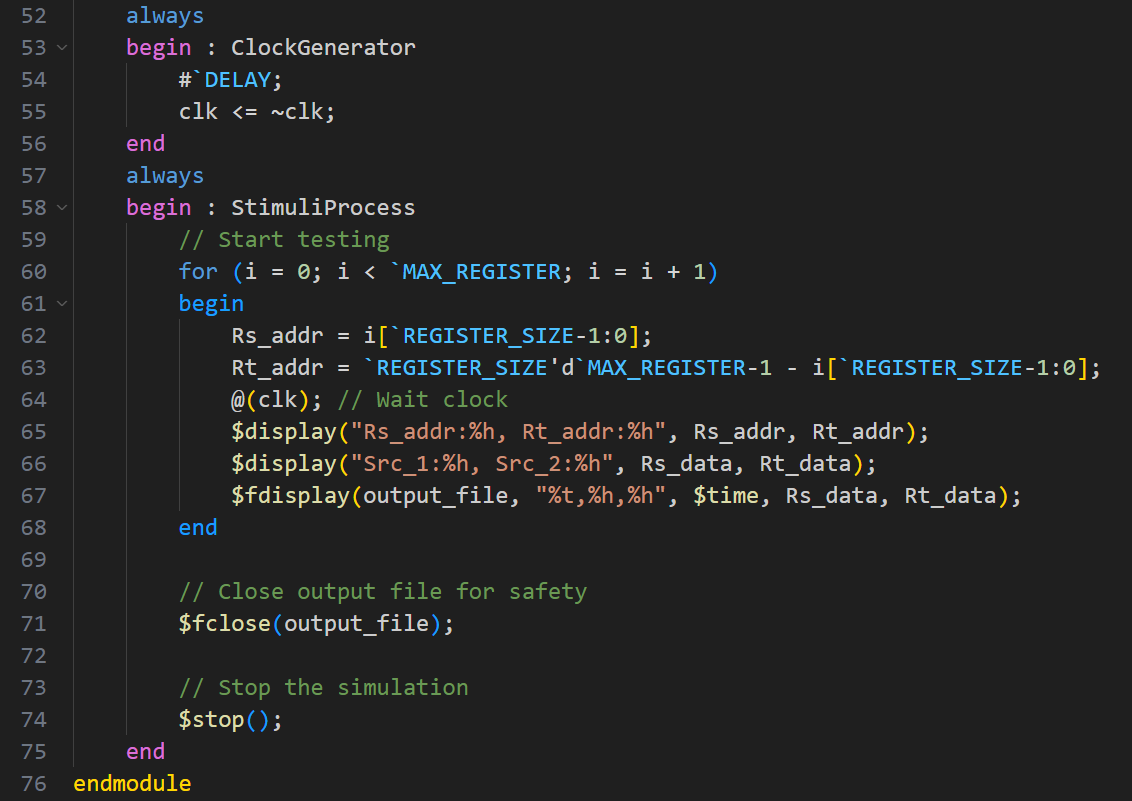
設定clock時序，週期為兩個`DELAY的時間(20ns)。Source Register是從頭開始讀取，而Target Register從最後一個暫存器開始讀。變數i資料型態為integer，大小32個bit；而Address大小為5個bit，所以要用陣列表示法限定i的前5個bit賦值到Rs\_addr/Rt\_addr。最後將讀取結果輸出到終端並記錄到tb\_RF.out。







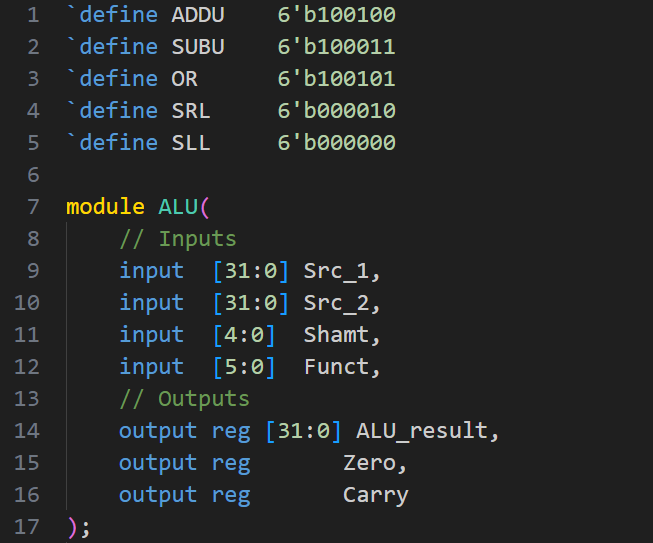
tb\_RF.v模擬



tb\_RF.v讀取暫存器資料

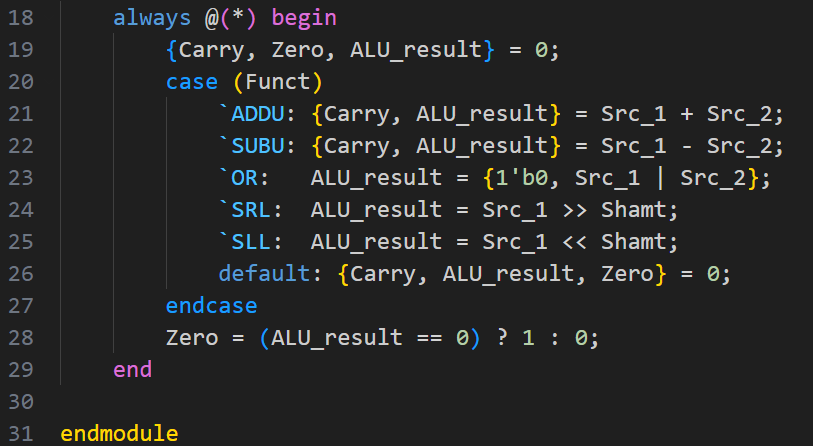
模擬後總共會有32次讀取的資料，因為有兩個暫存器，一個從頭開始讀，一個從尾，所以這邊截圖到第16次讀取的資料就可以把所以暫存器的值都讀完。對比一下各位址的值，可以發現是與RF.dat吻合的。

1. 32-bits Arithmetic Logic Unit
2. 主程式



ALU.v初始化設定

定義各Function Code對應的功能常數(ADDU、SUBU、…)，宣告此module要用到的input及output。

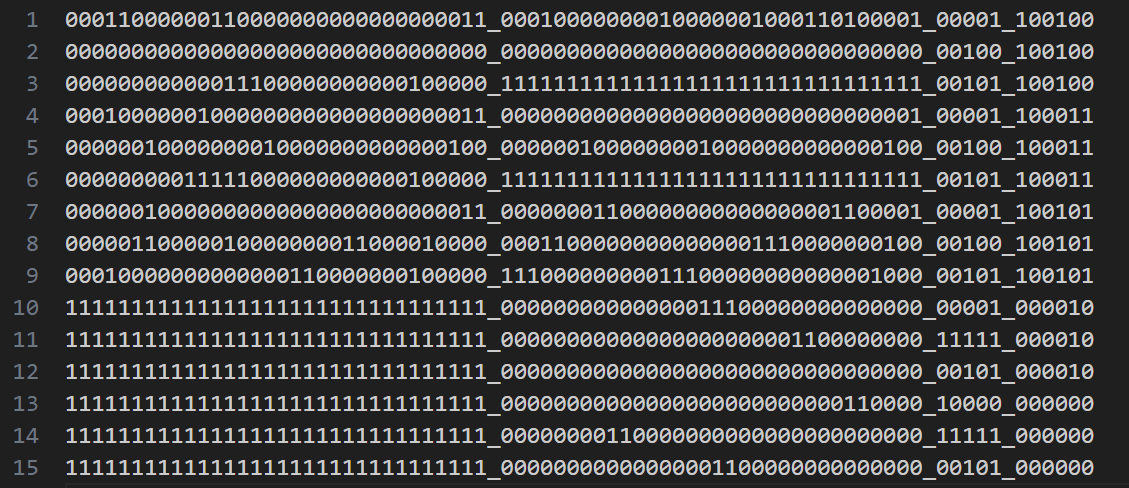


ALU.v主要功能

定義此ALU為由其中一個輸入變化而改變輸出的組合邏輯電路。

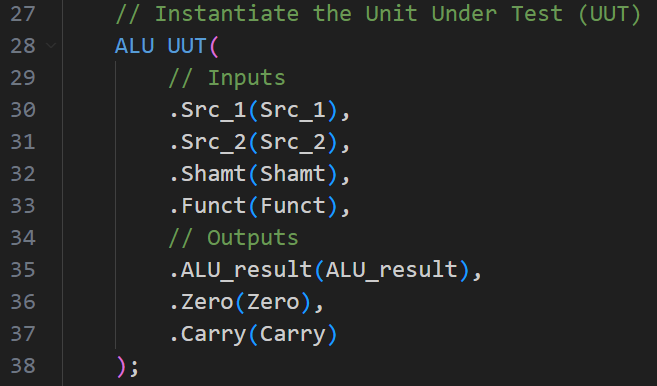
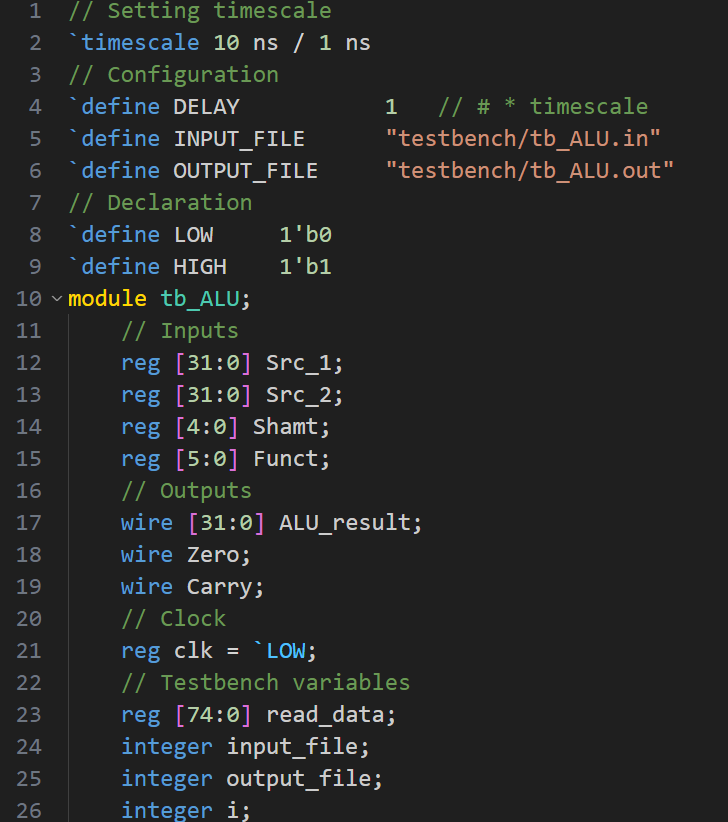
利用Switch-Case判斷Function Code對應的功能做運算，最後再用三元運算子判斷ALU\_result是否全為0；若是，則Zero旗標為1。

1. Testbench and Simulation



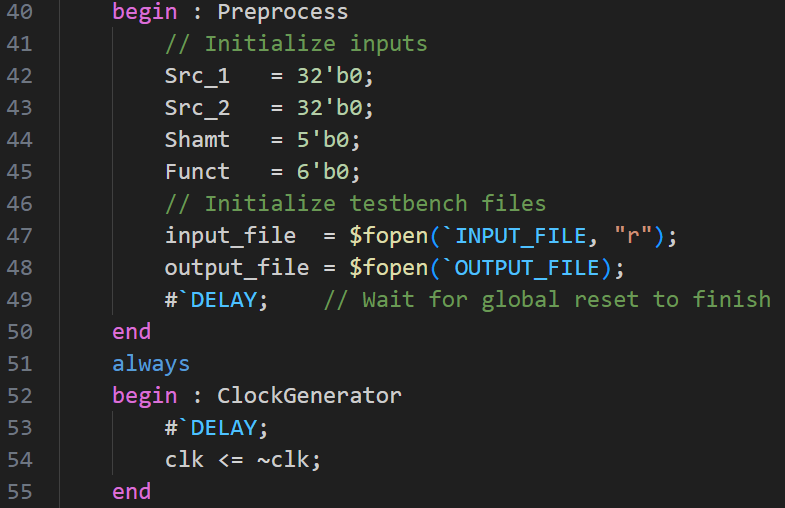
tb\_ALU.in所有測試向量

測試向量將5個功能都測試到，每個功能都有三組資料。



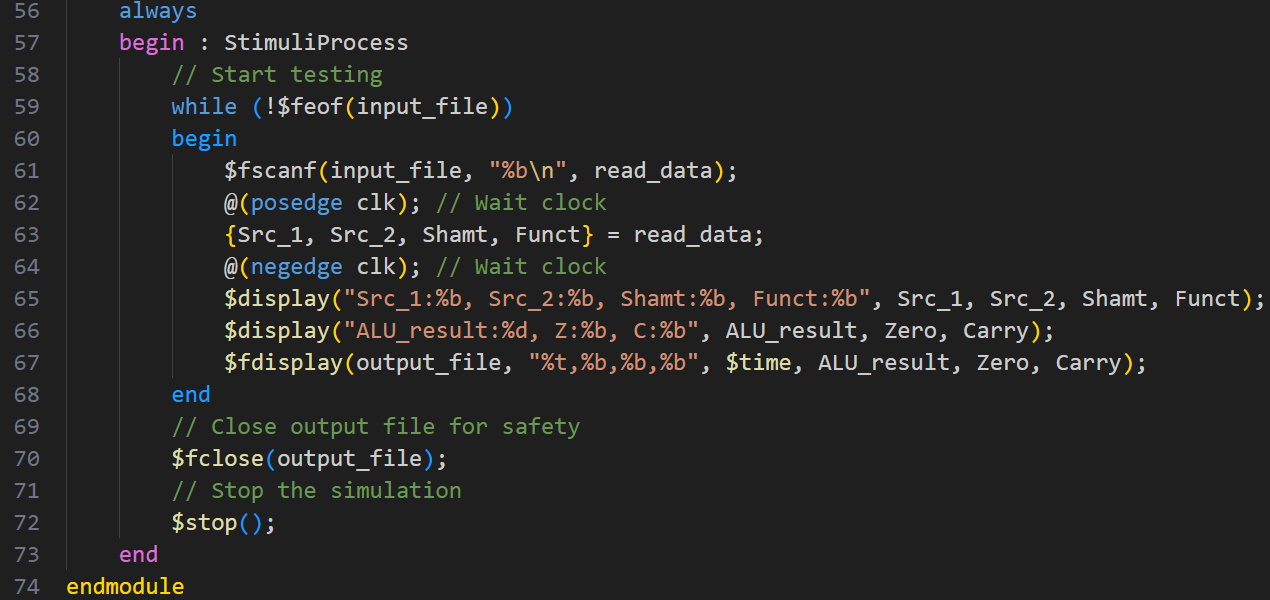
tb\_ALU.v初始化設定

設定時間單位10ns、時間精度1ns。宣告各暫存器、變數、連接線，並與ALU Module連接。



tb\_ALU.v預處理

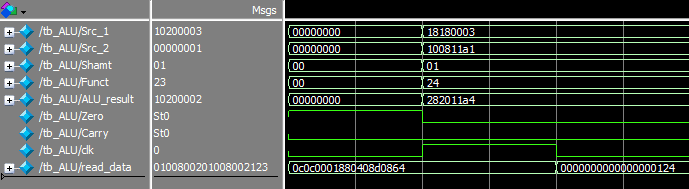
初始化各變數、讀取tb\_ALU.in內測試向量、設定clock。

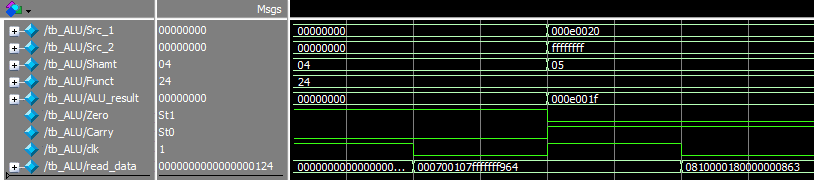


tb\_ALU.v測試

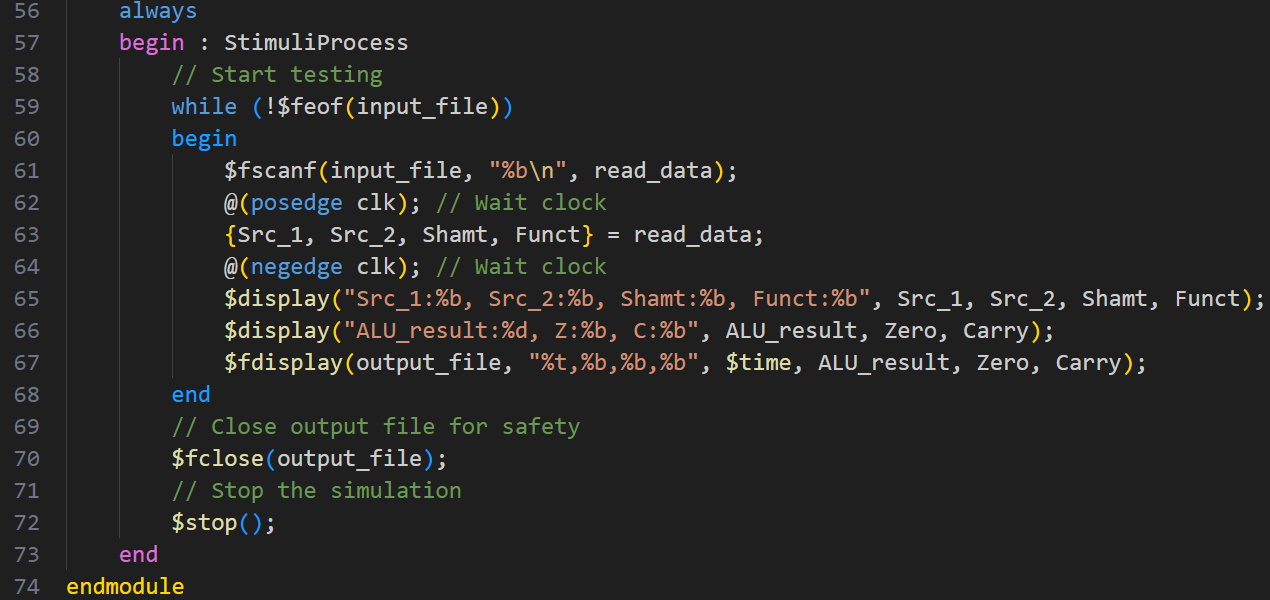
將各行測試向量賦值進read\_data變數，並利用連結運算子將read\_data拆分進Src\_1、Src\_2、Shamt、Funct。最後輸出結果至終端與tb\_ALU.out。

I. 加法測試





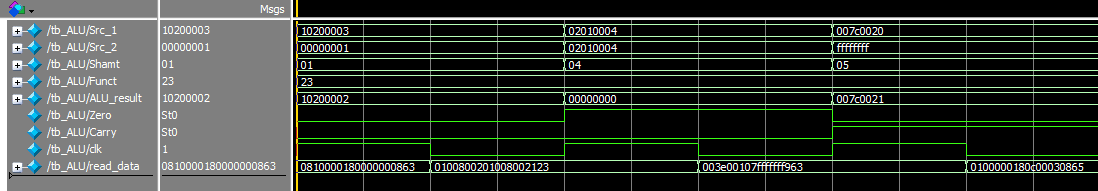
tb\_ALU.v模擬加法部分



tb\_ALU.v測試

三組資料測試後可發現加法後的結果正確、Zero與Carry旗標正常運作、Shamt不影響加法功能。由此可判斷加法功能正常執行。

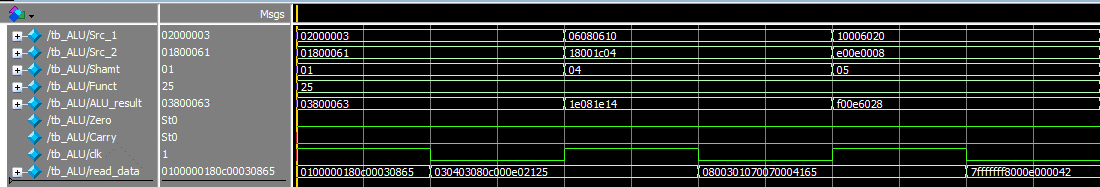
II. 減法測試



tb\_ALU.v模擬減法部分

三組資料測試發現減法結果正確、Zero與Carry(借位)旗標正常運作、Shamt不影響減法功能。由此可判斷減法功能正常執行。

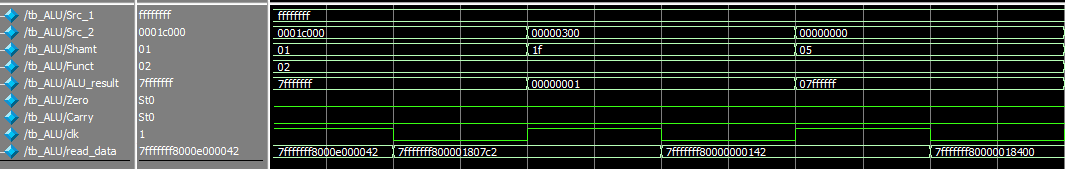
III. OR測試



tb\_ALU.v模擬OR部分

三組資料測試發現或閘結果正確、Zero與Carry正常運作、Shamt不影響或閘功能。由此可判斷或閘功能正常執行。

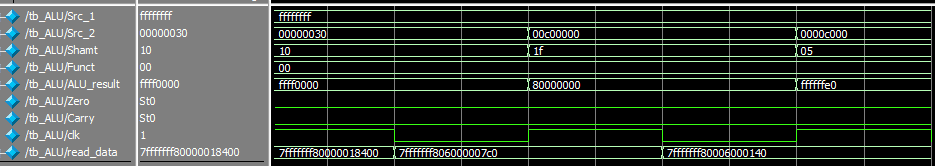
IV. 右移測試



tb\_ALU.v模擬右移部分

三組資料測試發現右移正常、Zero與Carry正常運作、右移位元數確實依照Shamt決定。由此可判斷右移功能正常執行。

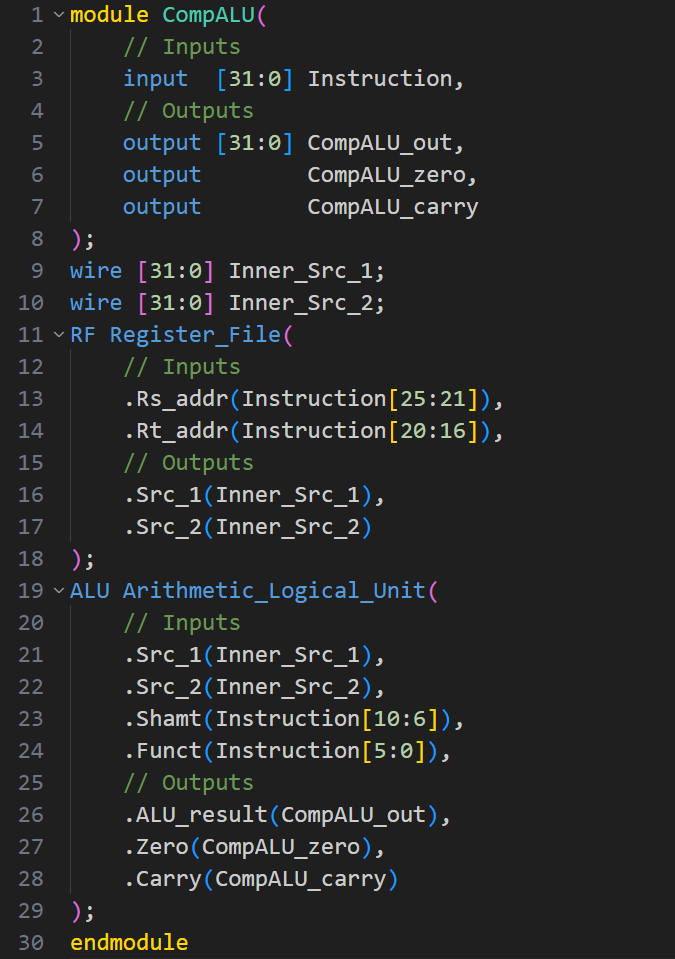
V. 左移測試



tb\_ALU.v模擬左移部分

三組資料測試發現左移正常、Zero與Carry正常運作、左移位元數確實依照Shamt決定。由此可判斷左移功能正常執行。

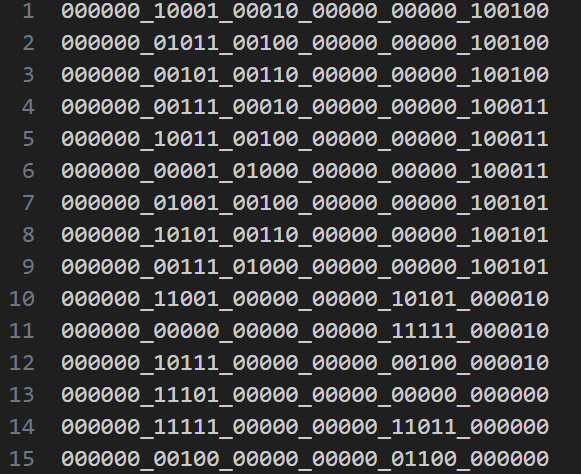
1. 32-bits Complete ALU
2. 主程式



CompALU.v程式碼

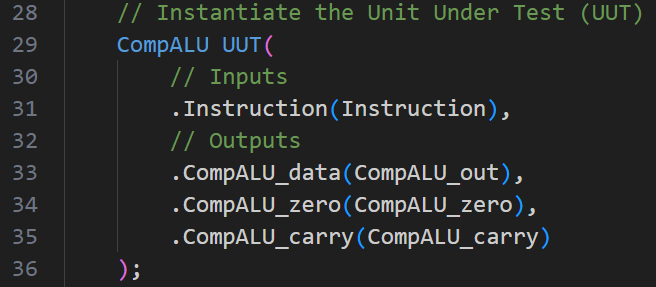
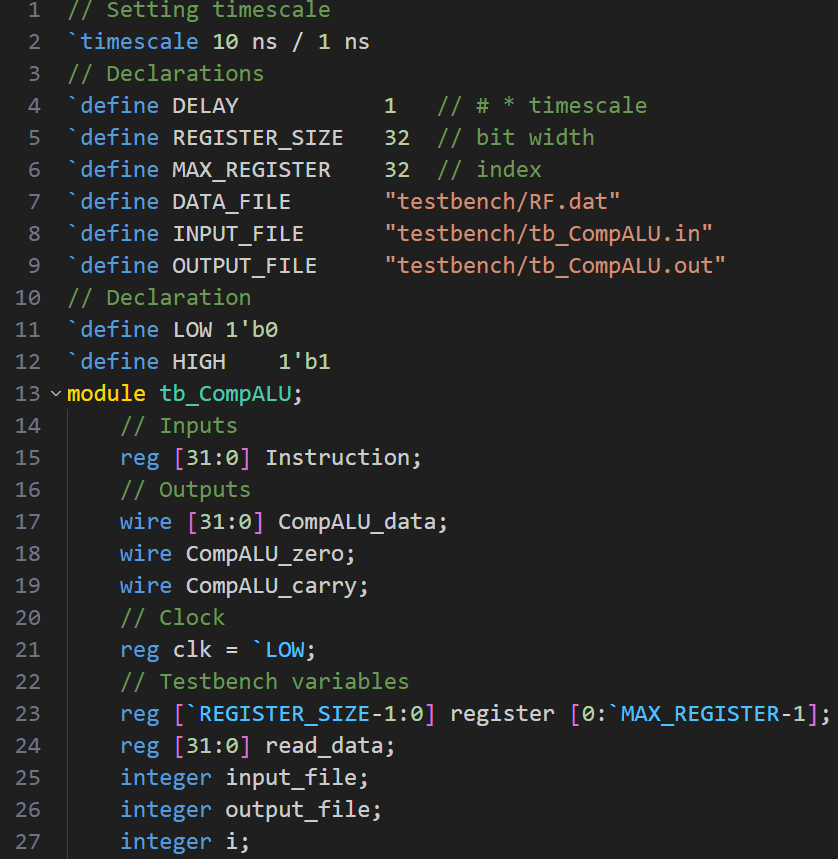
這程式把Register File與ALU結合起來，輸入32-bits的Instruction，利用陣列表示法將Instruction分離出Source Register、Target Register、Shamt、Function Code，並各自傳入RF與ALU module。就像是在CompALU這黑盒子內將RF與ALU連接起來。

1. Testbench and Simulation



tb\_CompALU.in測試向量

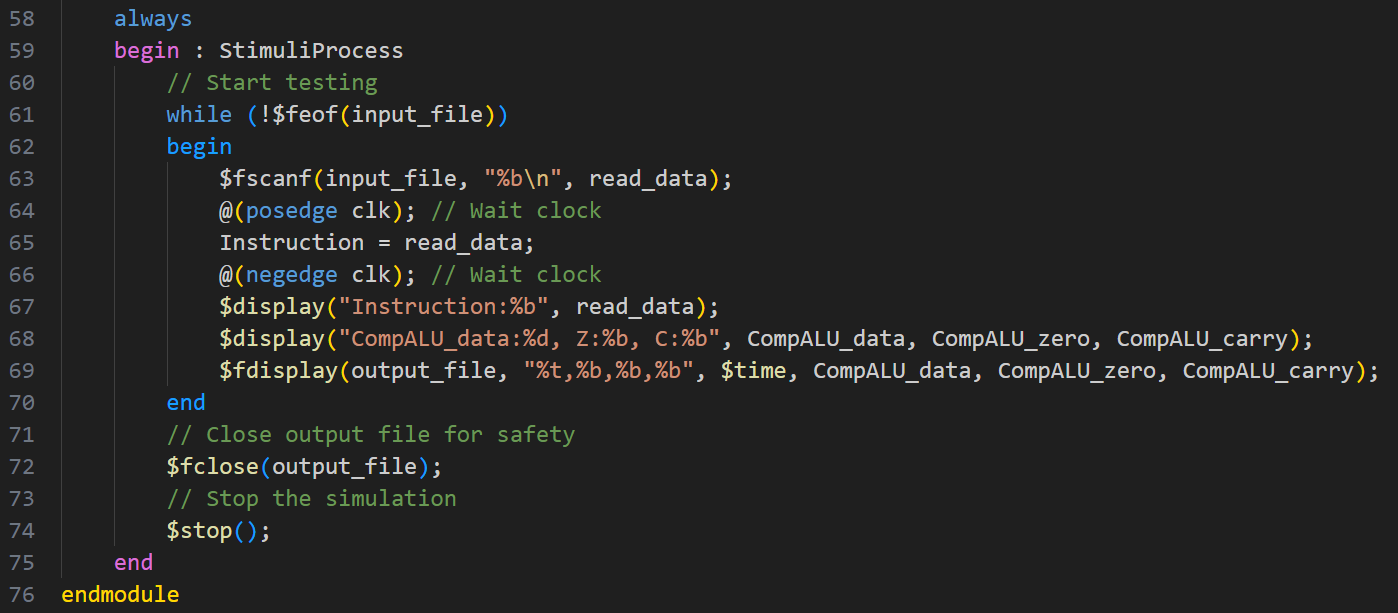
每個功能都有三組測試資料，總共十五組測試向量。



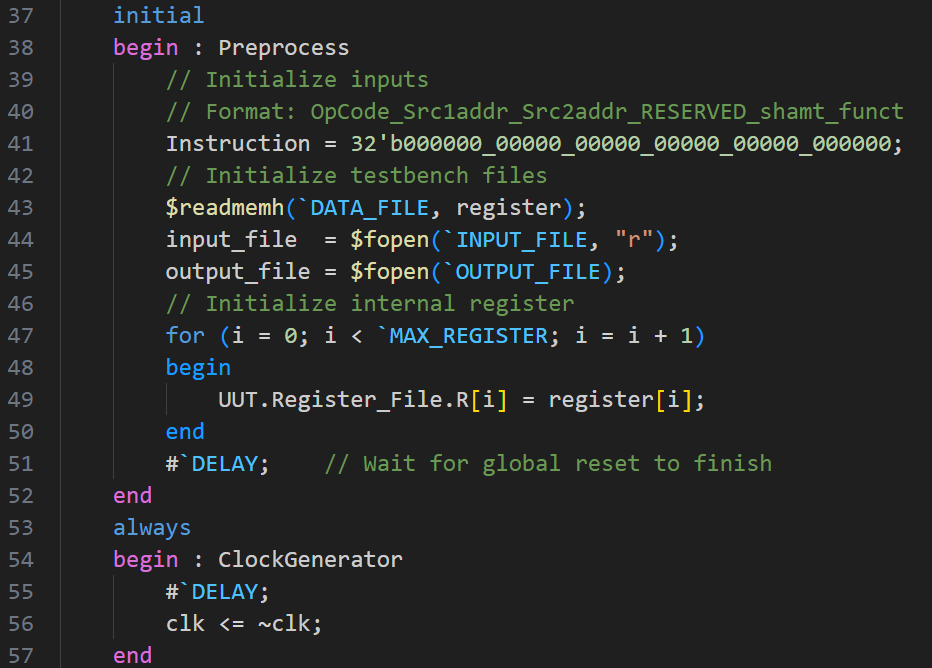
tb\_CompALU.v初始化設定

設定時間單位10ns、時間精度1ns。宣告各變數並連接至CompALU module。

將RF.dat記憶體資料一一匯入Redister File個暫存器內，並打開tb\_CompALU.in，將檔案位址傳進input\_file中，方便之後使用。ClockGenerator產生週期為20ns的clock。



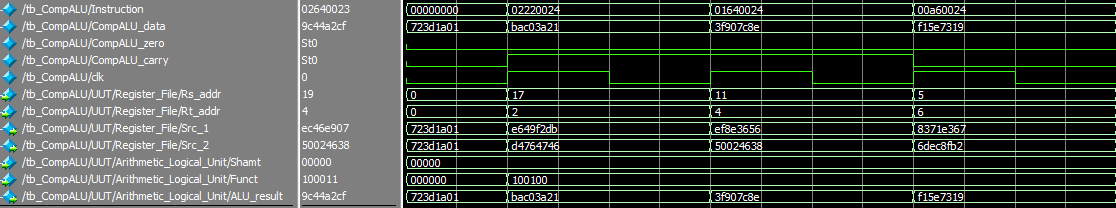
tb\_CompALU.v開始傳入資料



tb\_CompALU.v預處理

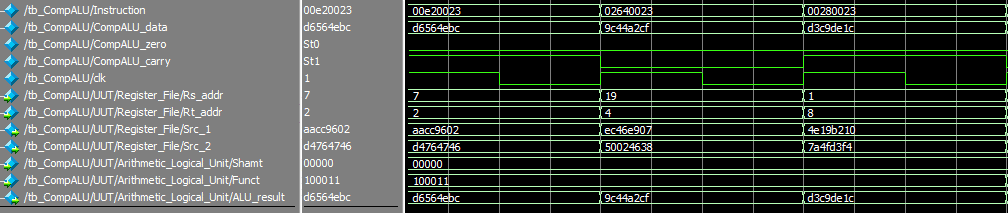
將tb\_CompALU.in內的測試向量，每隔一個clock週期，在上緣觸發時傳入CompALU中。最後將結果輸出至終端與tb\_CompALU.out檔案裡。

1. 加法測試



tb\_CompALU.v模擬加法測試

根據address從RF取的值與RF.dat對比後相符、加法結果正確、Zero與Carry旗標正確。以上確認加法功能正常。

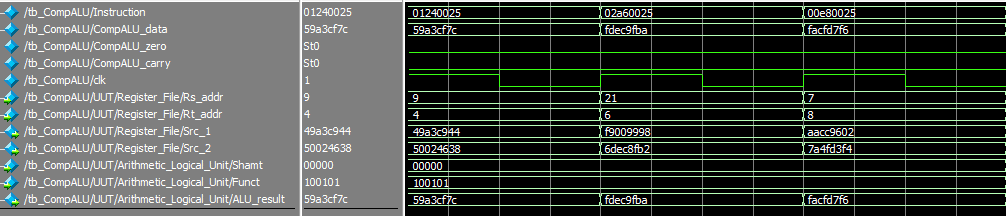


tb\_CompALU.v模擬減法測試

1. 減法測試

根據address從RF取的值與RF.dat對比後相符、減法結果正確、Zero與Carry旗標正確。以上確認減法功能正常。

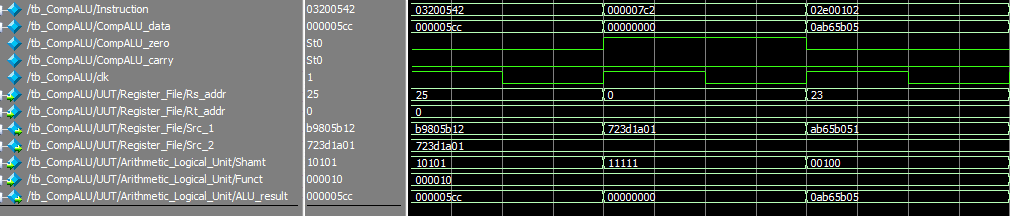
1. OR測試



tb\_CompALU.v模擬OR測試

或閘運算結果正確、Zero與Carry旗標正常反應。以上確認減法功能正常。

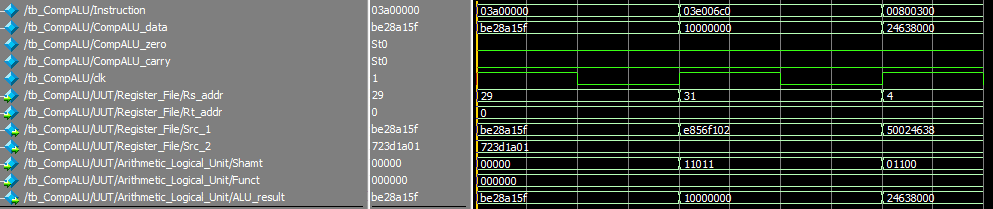
1. 右移測試



tb\_CompALU.v模擬右移測試

右移數量有確實依照shamt決定，結果正確、旗標正常運作，以上確定右移功能正常。

1. 左移測試



tb\_CompALU.v模擬左移測試

左移數量與shamt相符，輸出結果正確、旗標正常運作，以上確定左移功能正常。

1. Conclusion and Insight

一個CPU裡包含許多功能，我們可以把這些功能先分開實作，而每個獨立的實作就像是個黑盒子。確定個別實作沒問題後，我們只要把各個黑盒子相對應的接腳連接上就行，最後再組成一個更大的黑盒子。如此這樣模組化、階層化的去看待電路，會使得開發及維護更輕鬆。

在設計測試向量時，除了要有正常的數據外，還需添加邊緣條件，確保電路的每個輸出端都正常運作、電路可以正常處理近乎全部的輸入。像是在此專案，為了要測試Zero旗標，可能就要故意將兩個指定暫存器的值設相同，或兩個Address都指向同個暫存器。或是測試在不需要用到Shamt時的場景，例如加法時，給Shamt一個值，看會不會對加法功能有影響，若有，表示電路不如預期。