5 STAGE PIPELINE CPU – VERILOG實作

108學年度第2學期

老師：朱守禮　老師

第20組

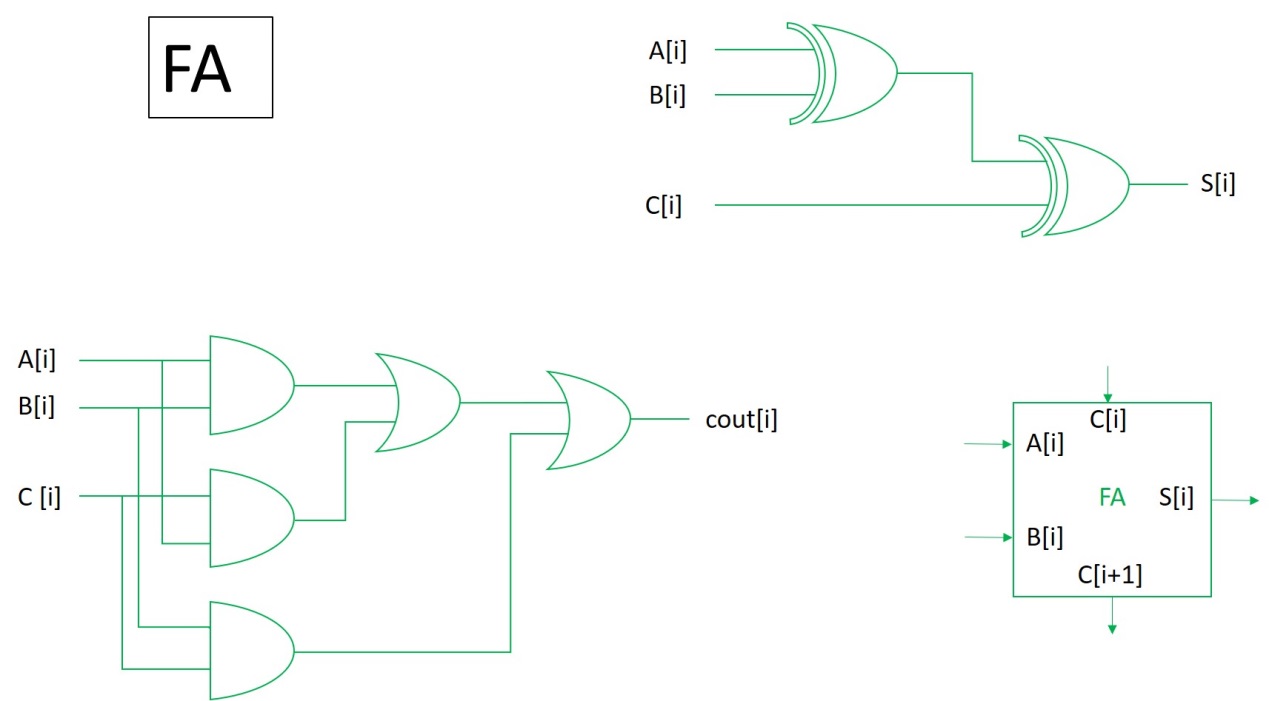
10727219許志仲 10727224李敏嘉

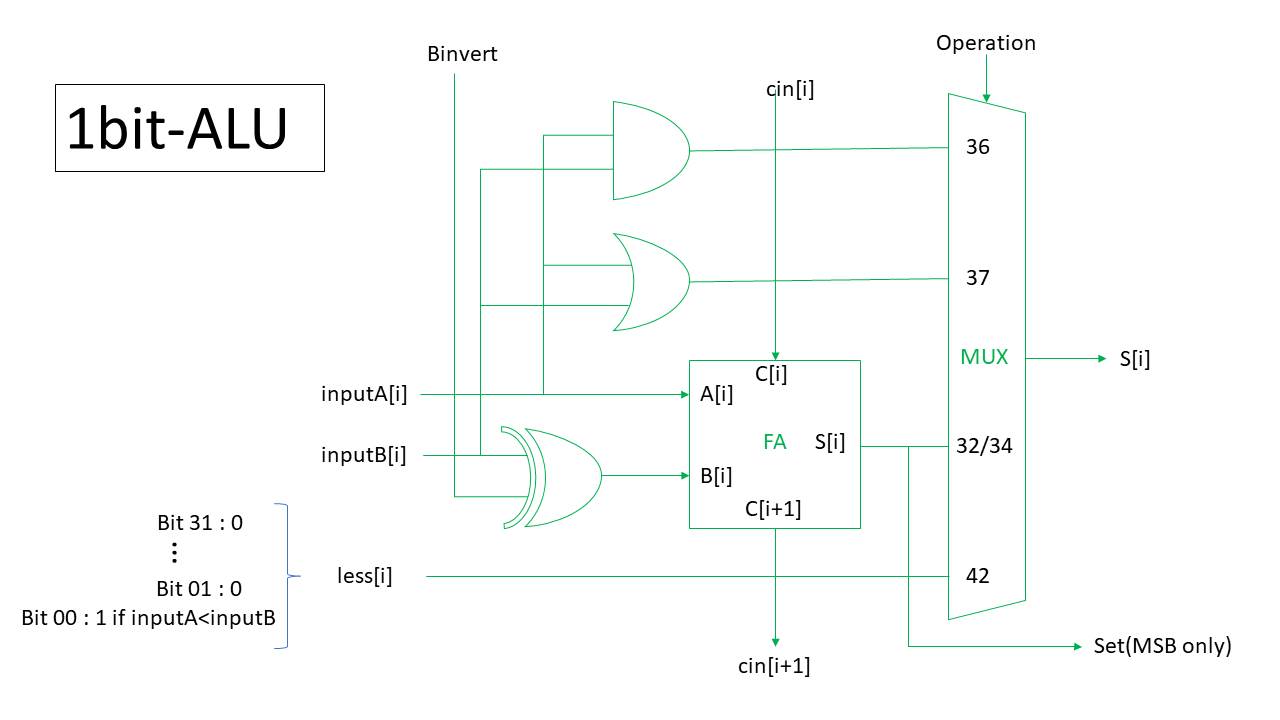
10727229 黃雅萱 10727252黃于九

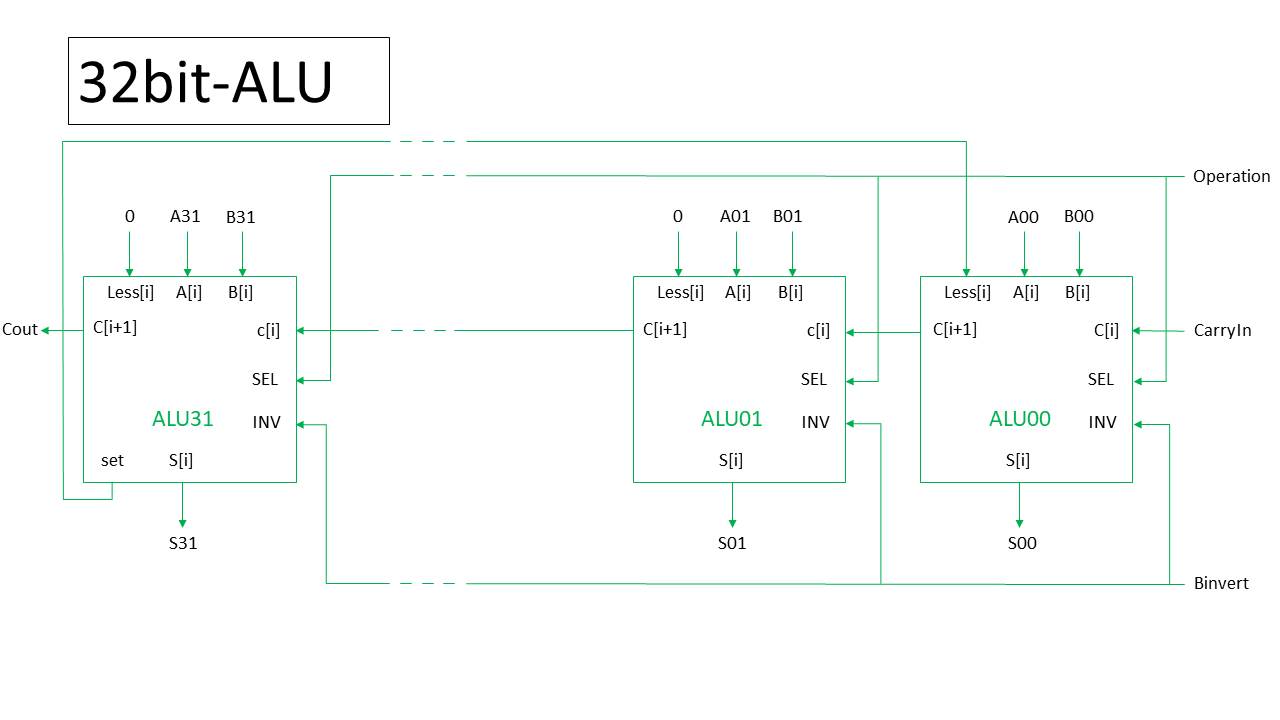
**背景**

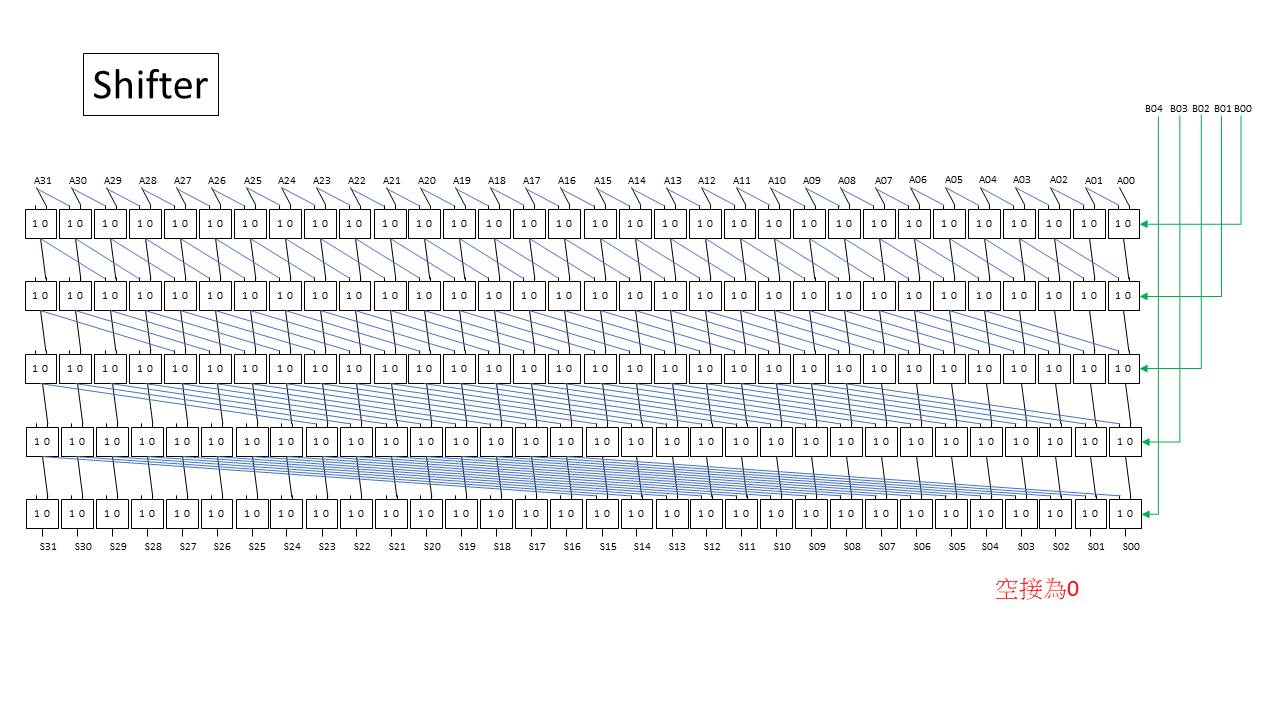
現代電腦升級快速，了解cpu構造與其運作成了資工系學生必備知識，因此利用這次的final project實作出一顆5 stage pipeline CPU，除了能了解實際創作之困難，更能對流程有更深度的理解。

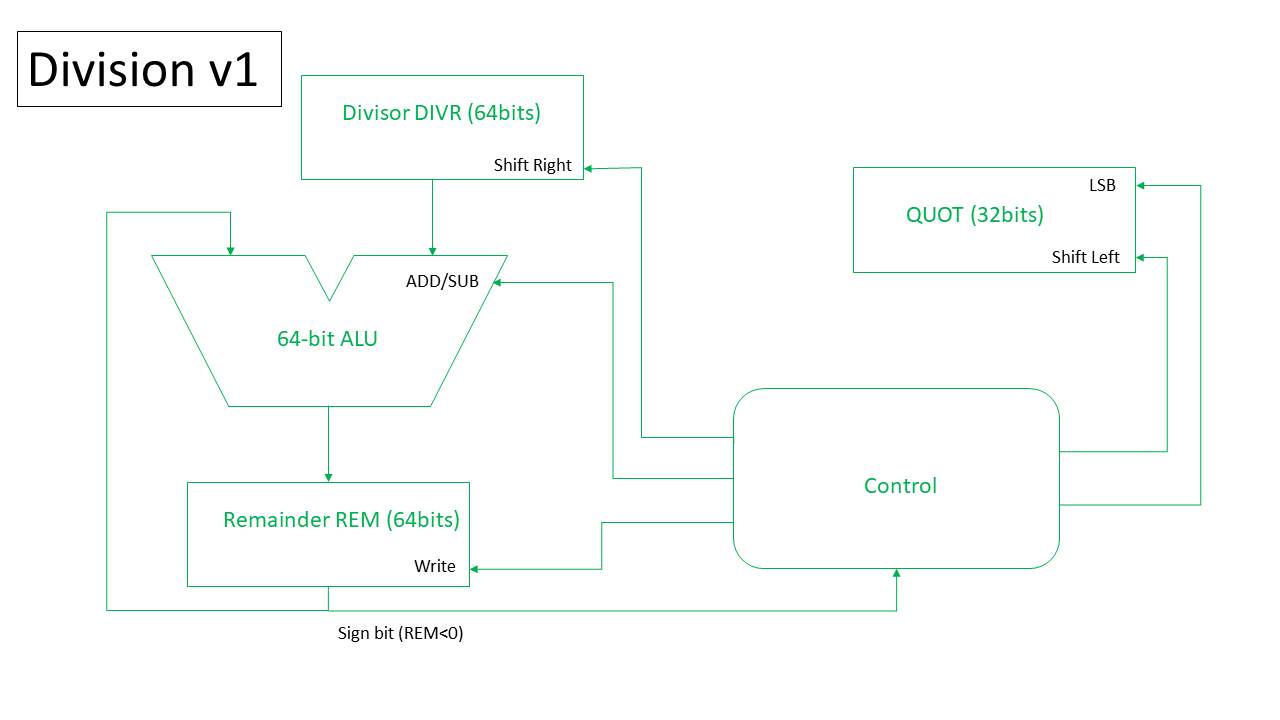
**Datapath與詳細架構圖**

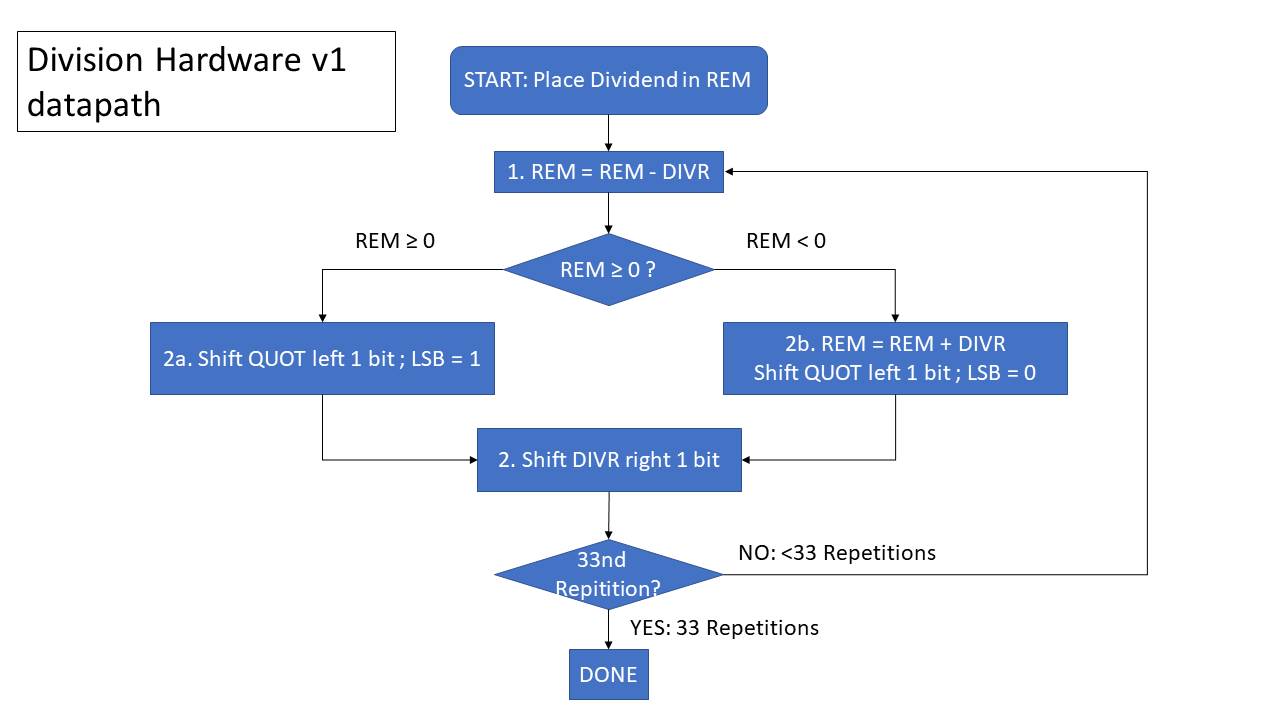
****

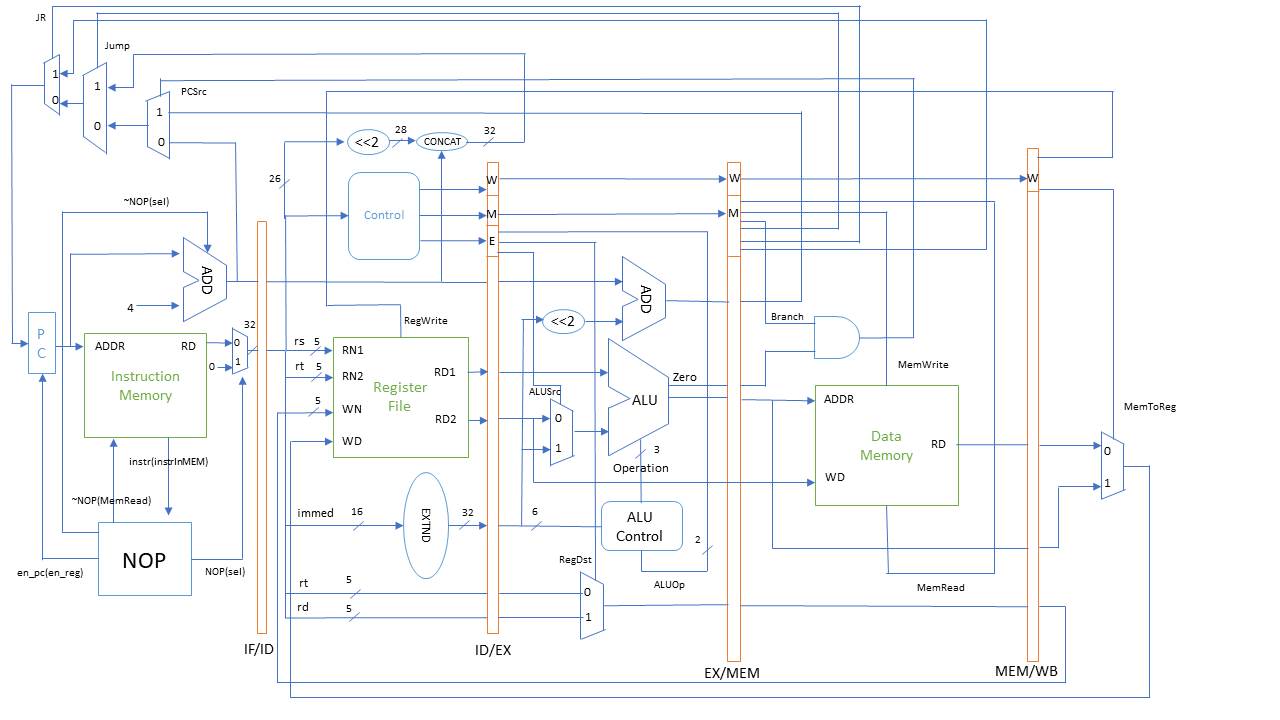












**設計重點說明**

**系統設計:**

我們將我們的系統設計成在第一筆指令執行之前先跑一次nop，方便清理訊號，並且在最後停止執行時，一遇到x(沒有指令了)就停止pc繼續上加的動作，維持在第一個沒指令的pc，以方便我們查看停止位置。

1. **ALU:**

ALU中包含32-bitsAND, OR, ADD, SUB, SLT, SRL這些功能，在串接32bitsALU時，我們還特別將最高位元分開成另一個Module，這樣就可以將相減結果設定到最低位元的less中，其餘的less皆設為0；也利用shifter同時做移位，並於最後依照功能選擇輸出；若總結果為0時則將zero訊號設為1。

1. **Division Hardware**

設計除法器的時候，我們特別考慮到除以0的情況，若除以零則輸出X，其餘則是正常的以加減法和位移來完成。除法器中我們有個count負責數33次，count從33往下數，當count = 0 時會算出正確答案，所以就不再繼續除下去。

一開始先把被除數放到餘數暫存器中(擴充成64bits，被除數放置低32bits中)，除數放到除數暫存器中(也是擴充成64bits，除數放在高32bits中)，進入除法後，將餘數暫存器的值與除數暫存器的值相減，判斷最高位元(符號位元): 若為1則為負數，所以將餘數暫存器加回成原本的值，並將quotient右移後，最低位元設為0，若最高位元為0則將quotient右移後，最低位元設成1。最後再將除數暫存器左移，count減一，重複執行到count = 0為止。

1. **Shifters**

我們的設計是將位移2^0 module的output給位移2^1的module當input，再由位移2^1的output當成位移2^2 module的input…以此類推，每個module中都有個ctrl負責判斷要不要位移，若ctrl = 0代表不位移，照原本的input輸出，若ctrl = 1，代表要位移，必須將位移結果輸出。

1. **HiLo**暫存器

除法計算完之後，將結果放至32bits的Hi暫存器及32bits的Lo暫存器中，Hi暫存器放餘數，Lo暫存器放商數。此module與clock訊號同步，為循序邏輯。

1. **Mux**

多工器，依照選擇來控制訊號，用以處理訊號衝突。

1. **ALU Control**

根據輸入的6 bits訊號，來決定要做什麼運算，選擇有AND、OR、ADD、SUB、SLT、SRL。

1. **Testbench**

以讀檔方式讀入測試資料及正確答案，並於transcript輸出執行時間與動作，方便看查資料正確性。

1. **Control single**

對確定欲執行之指令做訊號判別，依序傳送給下個階段。

1. **Reg\_file**

讀取與存放暫存器中的值，受write訊號控制，隨時可讀，clk正緣敲起時可寫。

1. **InstrMEM**

放入pc值便可取出位於pc之指令，受read和write(永遠為0)兩個控制訊號控制，隨時可讀，不可寫。

1. **Add32**

用於計算beq欲跳轉之指令地址，與pc=pc+4之計算，為32位元加法。

1. **DataMEM**

放入地址便可取出位於該地之之資料，也可放入地址以後寫入，受read和write兩個控制訊號控制，隨時可讀，clk正緣敲起時可寫。

1. **Reg\_pc**

暫存pc值，在clk正緣敲起時才送出指令，維持時序使系統保持穩定。

1. **Reg\_IFID/IDEX/EXMEM/MEMWB**

將資料值與控制訊號在下一個clk敲起時傳送給下一階段，控制訊號會逐漸消耗。

1. **Nop**

查看是否需要等待執行，並輸出對於控制指令之多工器訊號、reg\_pc、instrMem之讀取訊號。

1. **Sign extend**

有號數擴充，將16bits轉為32bits，用以處理立即數。

**Modelsim驗證結果與Waveform輸出圖形，並加以說明。**

clk=1 : pc=0 指令(LW),clk=6寫回暫存器

clk=2 : pc=4 指令(BEQ),接上3個nop等待運算,clk=5得出結果,clk=6移至目標pc

clk=3,4,5為nop,在clk=4,5,6時解碼得到

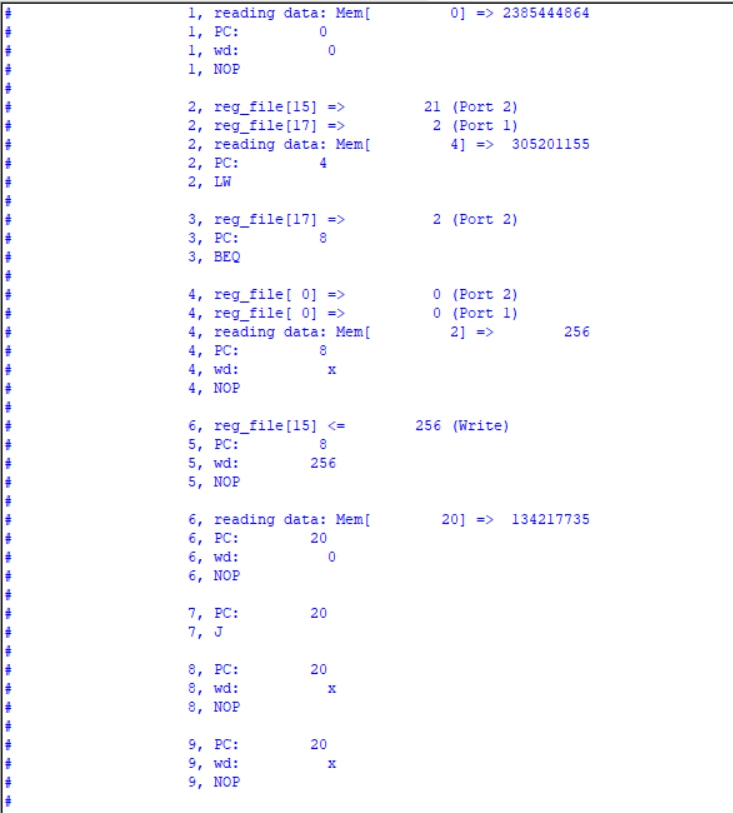
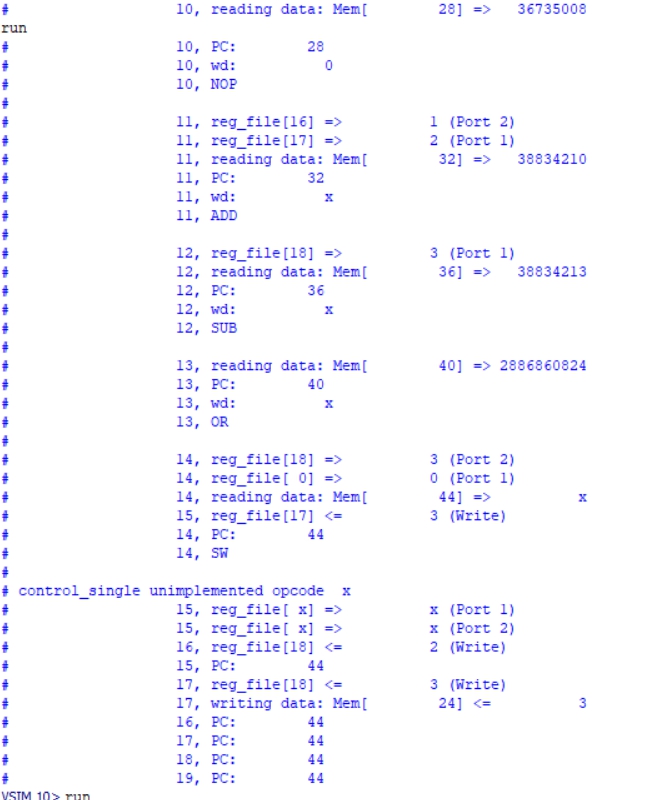
clk=6 : pc=20 指令(J),與BEQ作相同處理(nop\*3),clk=10跳至結果

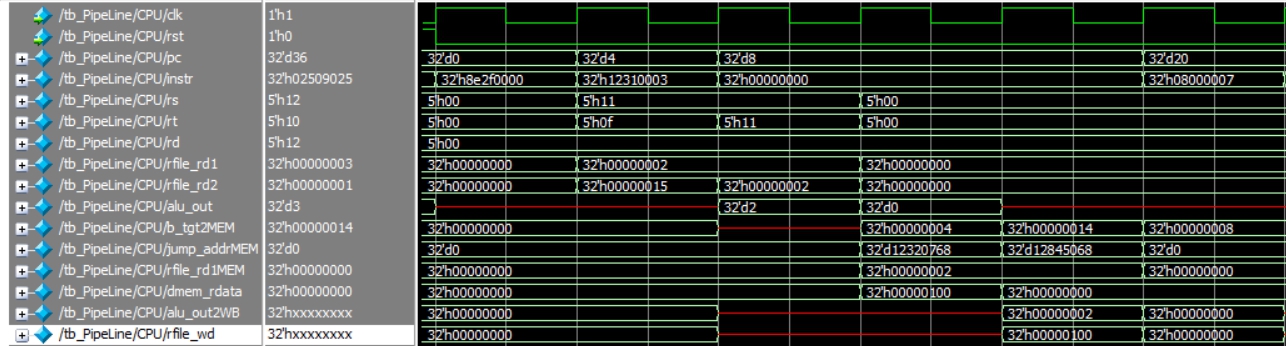
clk=10 : pc=28 指令(ADD),clk=15寫回暫存器

clk=11 : pc=32 指令(SUB),clk=16寫回暫存器

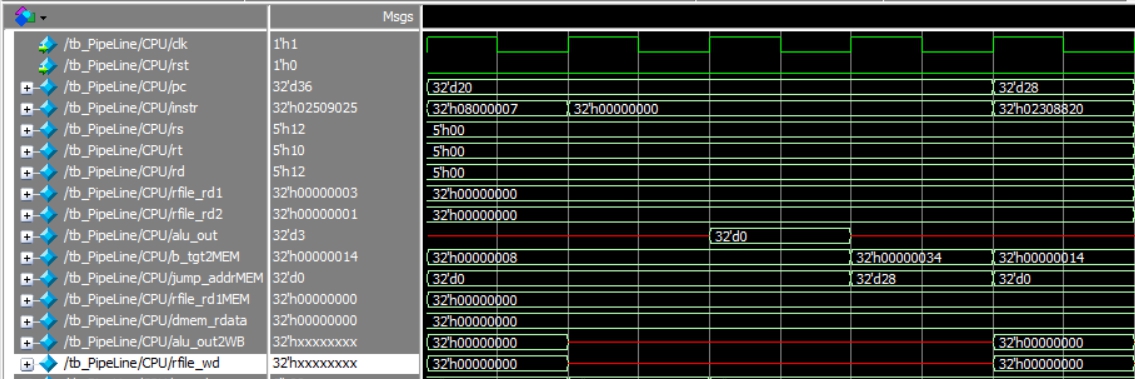
clk=12 : pc=36 指令(OR),clk=17寫回暫存器

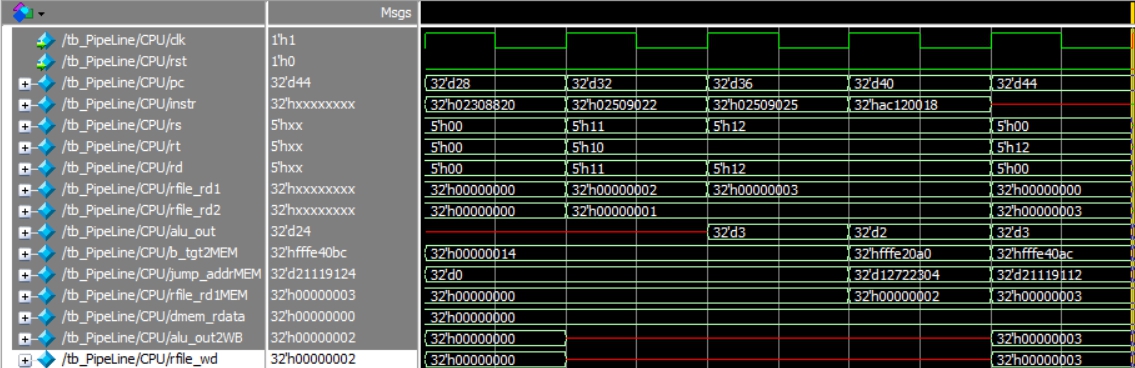
clk=13 : pc=40 指令(SW),clk=17寫進記憶體

****

**lw**

**beq**

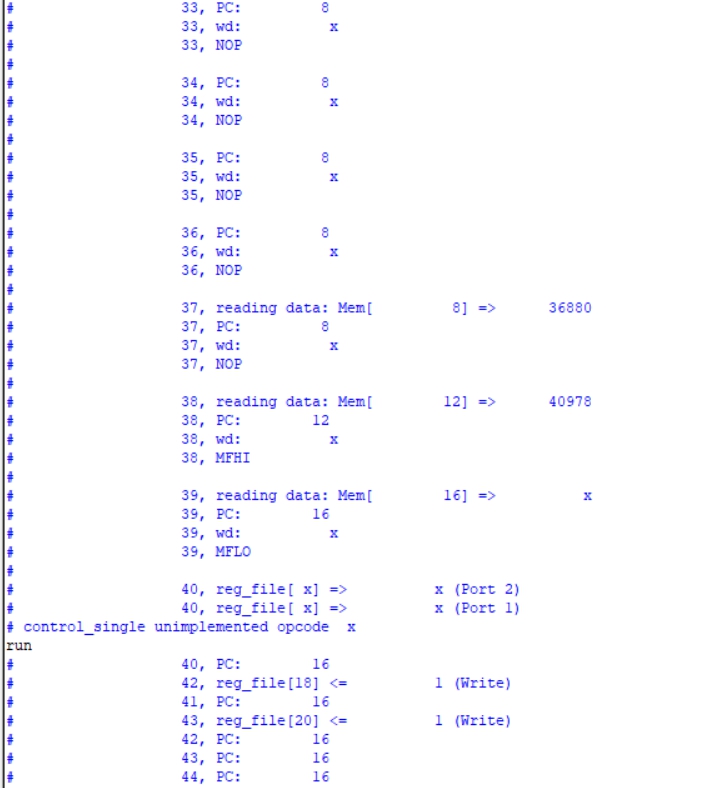
**j**

**R-type**

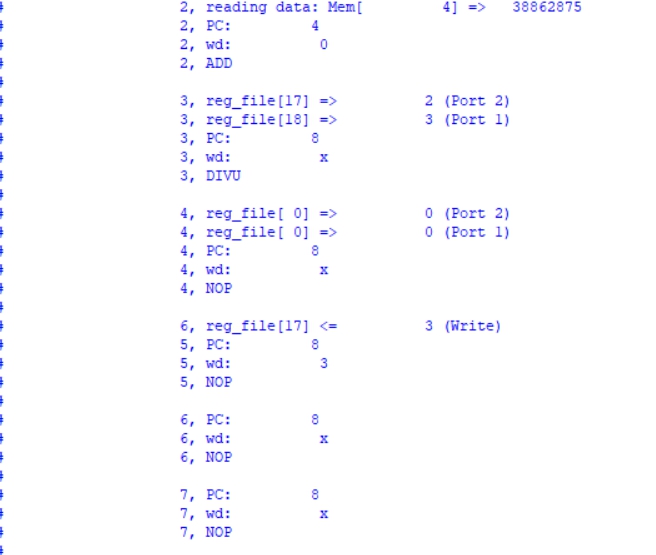
**sw**

****

除法的結果由hilo暫存器接收

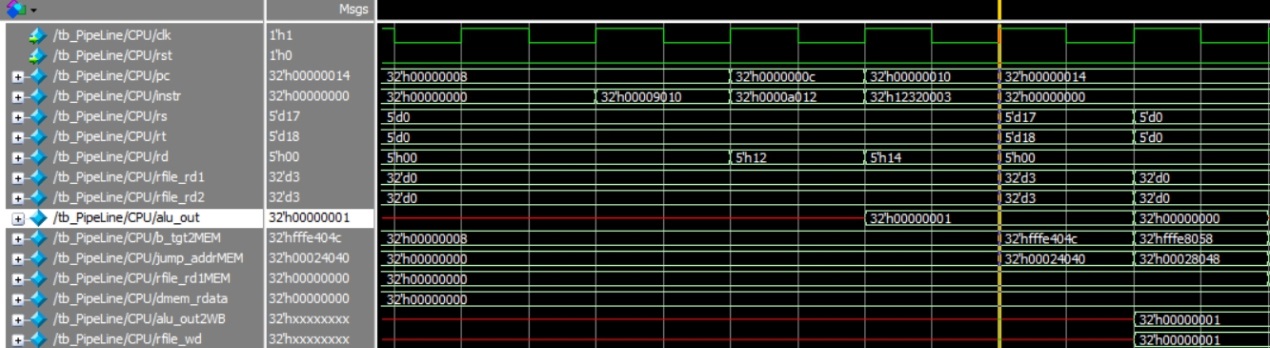
mfhi將餘數寫回指定暫存器

mflo將商數寫回指定暫存器

****

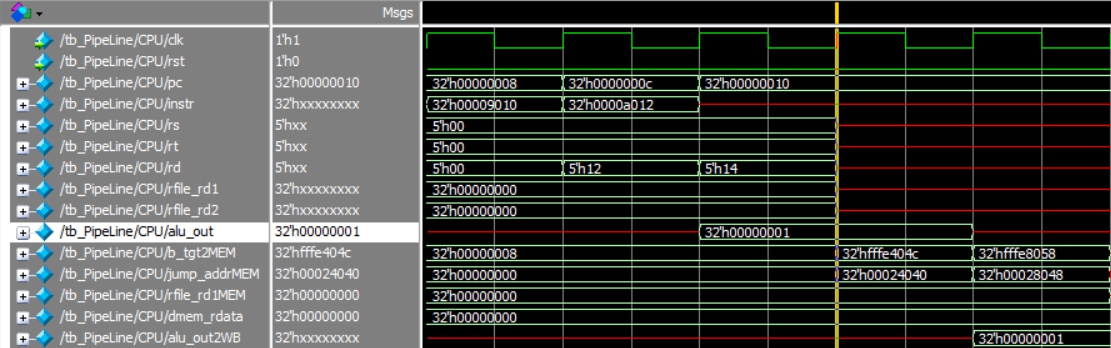
**Divu**

****

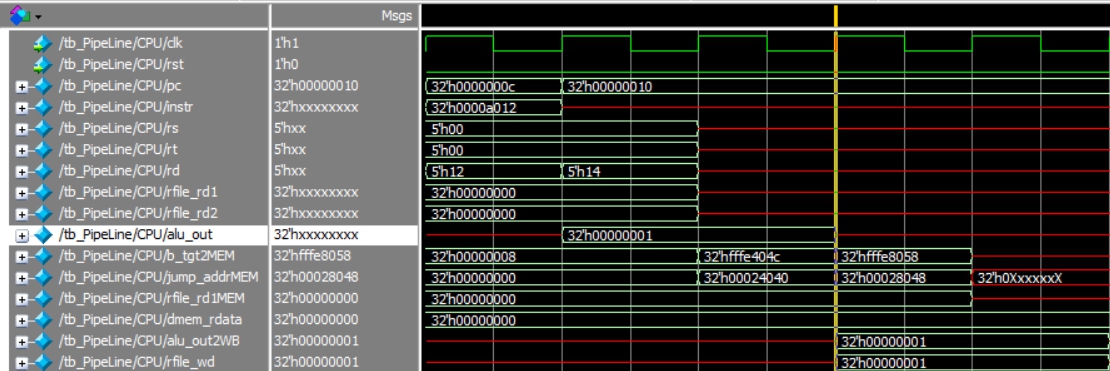
****

**Mfhi**

**Mfhi**

****

**Mflo**

****

**心得感想**

此次final project是延續midterm的功能來實作一顆5 stage pipeline CPU，聽起來少了很多部分需要實作，但如何接線卻成了一大難題，以下我們按照我們的實作流程來仔細探討。

STEP1切pipeline

首先我們使用了老師的Single Cycle版本將其切成5 Stage Pipeline，在這方面上我們並沒有遇到太大的困難，但我們也發現，Pipeline為了在每一階段都能執行不同指令，所使用的接線會比Single Cycle更多，若是不清楚架構圖，則會被混淆。

STEP2將ALU替換成我們的期中版本，完成ADD、SUB、OR、AND、SLT指令

替換上我們也執行得十分順利，但也必須注意去生成SLT的alu指令，否則便會遇上無法執行的窘境，而因為測試檔案中並沒有SLT的指令，因此我們在自己生成指令上有了更深一步的認識，期中前學的將指令轉為2進制就成了我們final的基底，我們也了解到了指令在電腦識別上是以Little Endian的方式，實在讓我們學了一課。

STEP3確認SW、LW的正確性

基本上只要ALU沒有問題的話，這兩道指令都不會出錯，值得注意的是我們要透過波形圖和Transcript所印出的資料來觀察這兩道指令的執行，更能增進我們對pipeline的理解。

STEP4添加SRL、ORI兩道指令

若是SRL要在ALU中達成，應該如何實現呢?我們在ALU中同時執行右移指令，並用多工器處理訊號衝突的特質，選擇輸出，如此便能達成我們所期望的成果；ORI是一道很特別的指令，需要讓暫存器中的值和立即值做OR運算，因此我們在control中獨立給予他的控制訊號們，便能安全達成。

STEP5添加BEQ、J、JR三道指令

這三道指令都有差不多的性質，就是跳躍，但在我們接線完成以後，發現會遇到control hazard的問題，因此在共同討論以後，我們選擇使用NOP指令來等control訊號計算完成。

STEP6討論NOP之實作方式並添加硬體，完成NOP指令

但NOP訊號該如何實作呢?這也是我們在這次project中卡最久的問題，經過多番討論以後，我們額外製作了一顆硬體，讓他判別我們取出的訊號是否需要解決control hazard，並在InstrMem的輸出方添加一個多工器，讓其可以自動轉換為NOP指令，便可達成自動之需求，但我們卻因為這個硬體多方碰壁，才發現我們太習慣將硬體想成軟體，而忘了硬體的特性，透過組員的互相提醒與討論，才讓我們逐漸步入正軌，真的是非常感謝所有組員大力配合燒腦動作。

STEP7添加DIVU及MFHI、MFLO三道除法器相關指令

我們將DIVU指令放在ALU中，以保持DATAPATH的穩定性，就能使用現有的接線來完成這些部份了，也能防止過多的訊號產生混亂，這點實在非常具有挑戰性，因為我們原本是將除法器外接的，直到發現訊號太多不好管控，所以這部份讓我們印象十分深刻。

STEP8加入各種測試資料以確認系統正確性

測試資料是檢測系統運作最佳的確認方法，因此我們在測資上十分花心思，翻遍所有資料，創造出許多情形下的指令(如:branch與否的兩個情形)，每次的檢測都讓我們對於datapath有更深的印象，架構圖也因此近乎烙印在我們的腦海中，日有所思、夜有所夢，連作夢都在寫計組，實在讓我們獲益良多與印象深刻。

透過這次的專題，以及組員們互相的討論，我們對彼此產生了更高的情誼，也了解到四個臭皮匠總會勝過一個諸葛亮的，這點讓我們十分慶幸；雖然以後走的方向不見得是硬體，但硬體是各界都需要的一個工具角色，有著不可或缺的魅力，希望我們能夠抱著此次有如雲霄飛車的經歷，繼續往資訊方邁進。

**未來展望**

有了此次經驗，相信我們對於pipeline CPU有了更深的情感，盼望我們能活用這次的經驗與相關知識，發展出更具前瞻性與效率更高的CPU。

**分工**

|  |  |  |
| --- | --- | --- |
| 程式碼 | 全部分皆為共同討論實作 | |
| 說明文件 | 架構圖與datapath | 黃于九、黃雅萱 |
| 設計重點說明 | 李敏嘉 |
| Waveform模擬圖與說明 | 許志仲 |
| 心得感想與檢討 | 共同討論修正 |