Midterm Project

106 學年度第 2 學期

ALU Design

老師： 朱守禮 老師

學生： 10527130 陳少洋
10527132 林亞吟
10527135 張智欽
10527140 初 元

一、背景

本 Midterm Project 主要為設計一個 ALU、第三版除法器、Barrel Shifter、HiLo 暫存器以及 MUX 多工器，其目的主要為實現七項功能：AND、OR、ADD、SUB、SLT、SLL、DIVU。

本次 Project 我們使用 ModelSim 來 compiler 以及執行，透過模擬 waveform 來檢查是否與 Testbench 的計算結果相同。

本次 Project 須符合以下規範：

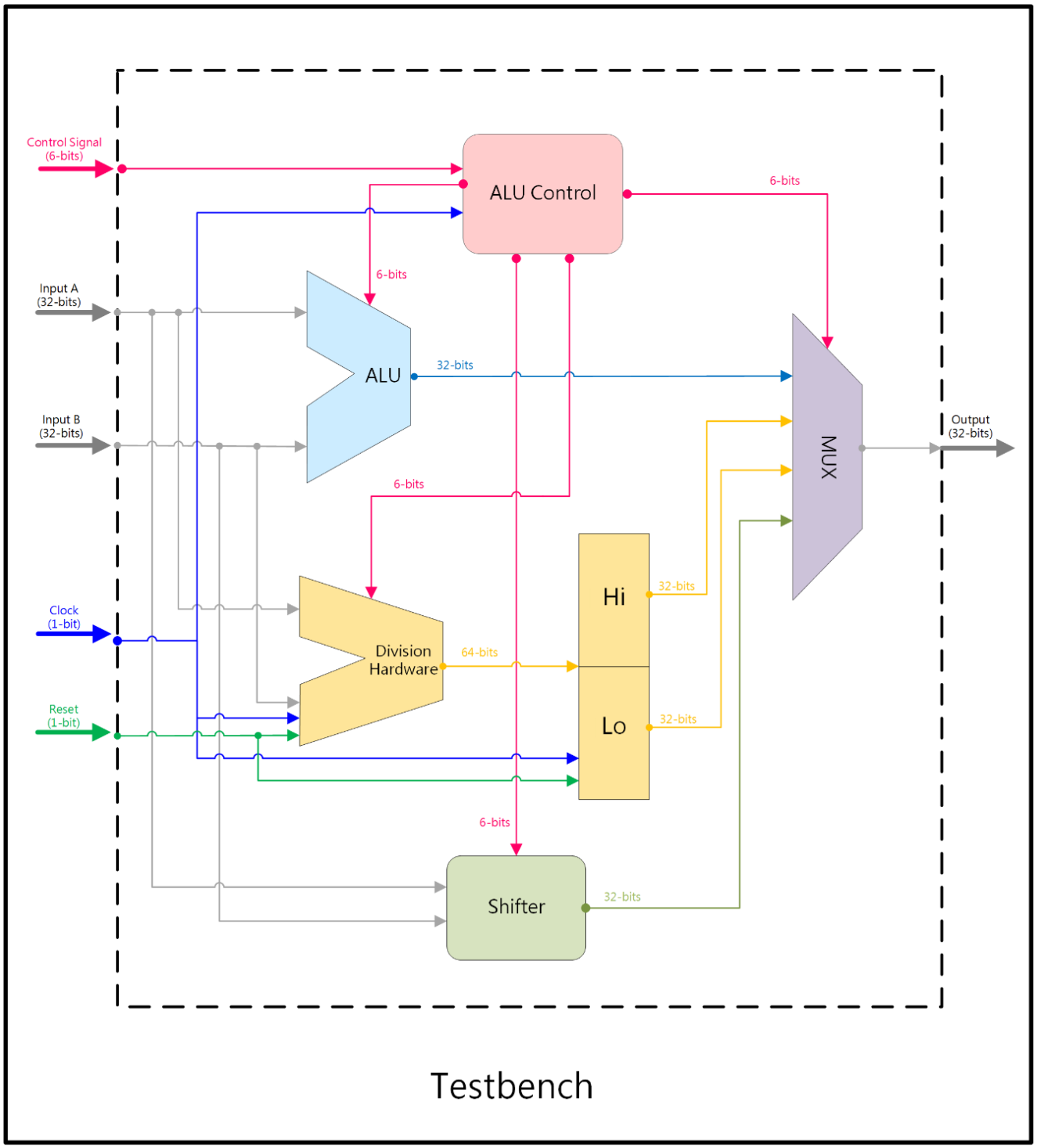
(1) 一個 Module 一個檔案，同時檔案名稱需與 Module 名稱相同。

(2) 須放置以 Visio、Word 或 PowerPoint 繪製的 Datapath 架構圖。

(3) Testbench 須依助教所提供之參考設計。

(4) 設計均以 Verilog 完成，且須通過 ModelSim 模擬執行。

(5) Verilog 設計只能包含規定的七項功能。


Control Signal
(6-bits)
ALU Control
6-bits
6-bits
Input A
(32-bits)
ALU
32-bits
Input B
(32-bits)
MUX
Output
(32-bits)
6-bits
Clock
(1-bit)
Division
Hardware
64-bits
Hi
32-bits
Lo
32-bits
Reset
(1-bit)
6-bits
Shifter
32-bits
Testbench

Datapath

二、方法

(一) ALU(組合邏輯)：

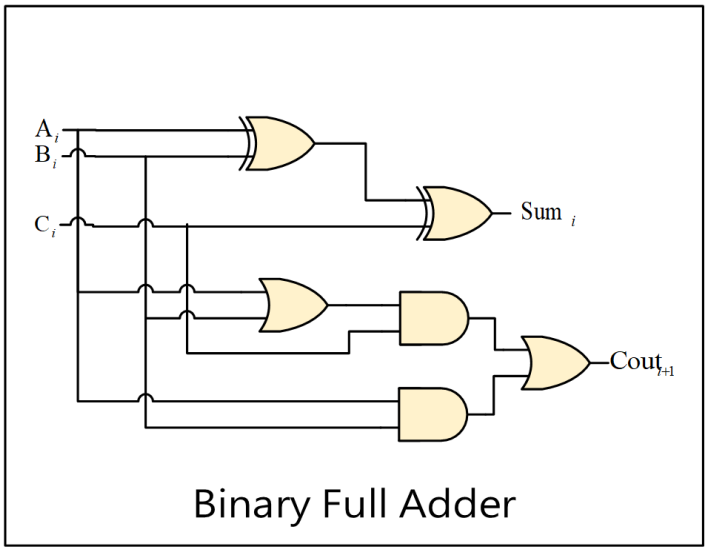
ALU 主要包含五項功能：32-bits AND、OR、ADD、SUB、SLT。

1. 設計重點

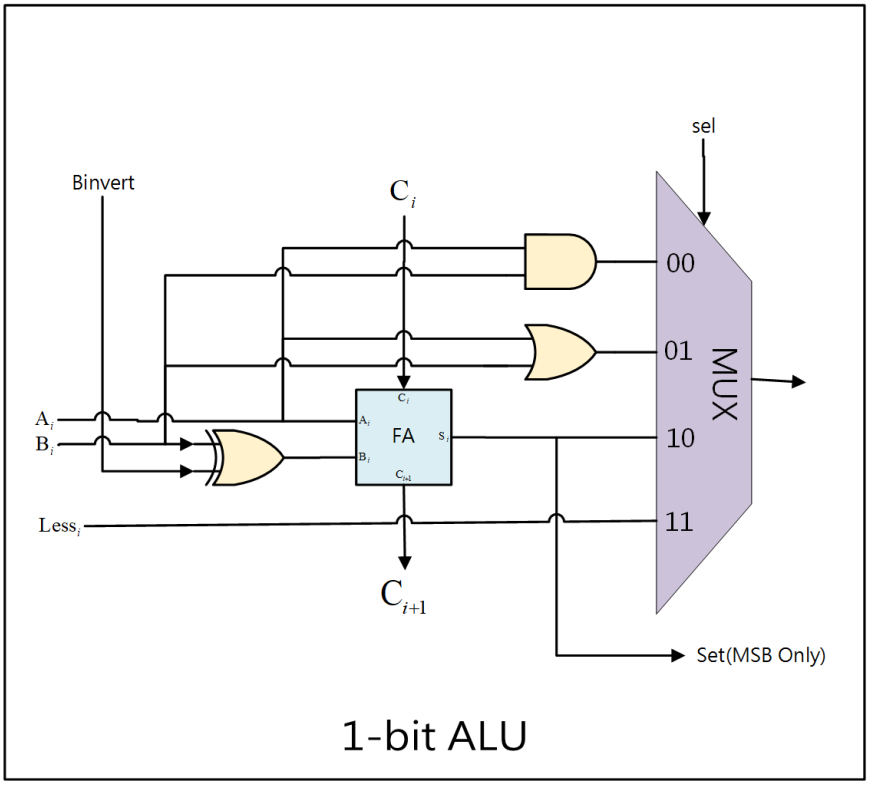
我們先設計 1 bit 的 ALU，instance 32 次，達到 Ripple Carry 的效果。

2. 說明

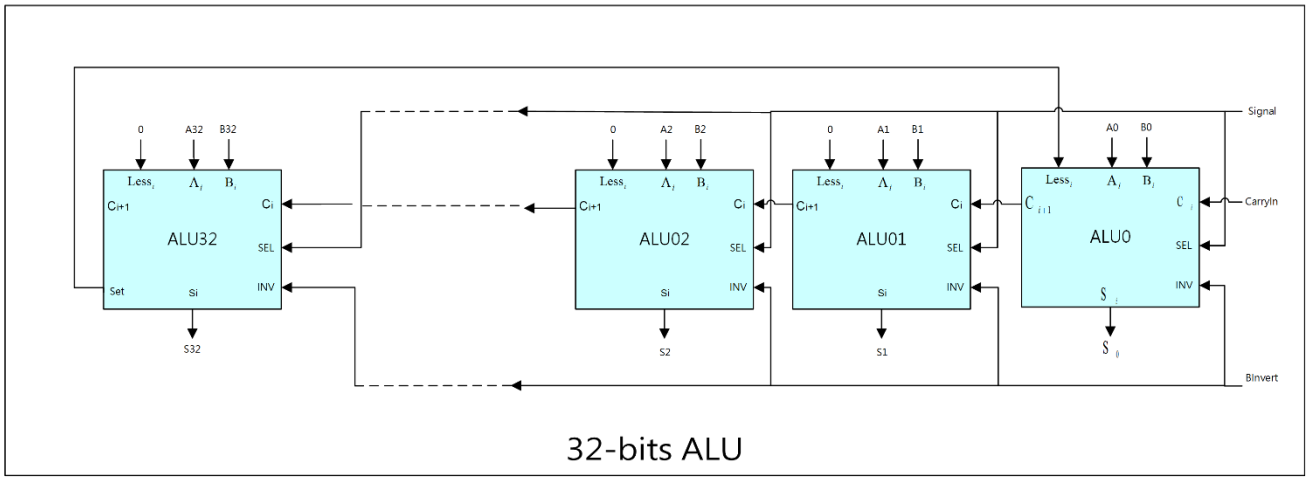
- ADD、SUB、SLT 不可直接使用”+”、”-“，必須以邏輯閘撰寫設計。
- AND : 000 / 36 / 100100
  OR : 001 / 37 / 100101
  ADD : 010 / 32 / 100000
  SUB : 110 / 34 / 100010
  SLT : 111 / 42 / 101010
- AND : 000 第一位元為 Bit Invert，二、三位元為 Selection，36 則為指令代碼，100100 則為 6-bits Signal，為 36 轉 binary 後的值，我們讓他 1 bit 1 bit 做 AND 直到 32bit 結束，OR 也是一樣的操作。
- ADD : 010 第一位元為 Bit Invert，二、三位元為 Selection，32 則為指令代碼，100000 則為 6-bits Signal，為 32 轉 binary 後的值，設計 ADD、SUB 時我們都是使用 Full Adder，差別在於 Binvert，從 AND、OR、ADD、SUB、SLT 的 Signal 可以發現，只有 SUB、SLT 的 Signal[1]為 1，因此我們利用 Signal[1]來做判斷，決定 Binvert 是 0 或 1。
- 使用 SUB、SLT 時，兩指令都會有相減的動作，因此我們設計了一個 BinvertCarry，用來當成第一 bit 的 ALU 的輸入，若指令為 SUB、SLT，BinvertCarry 就設定為 1，其餘指令為 0。
- SLT 會利用 SUB 判斷兩數的關係，若兩數相減大於等於 0，SUB 結果的最高位元為 0，反之，兩數相減小於 0 為 1。最後會將此最高位元設定成輸出值的最低位元。



Binary Full Adder



1-bit ALU



32-bits ALU

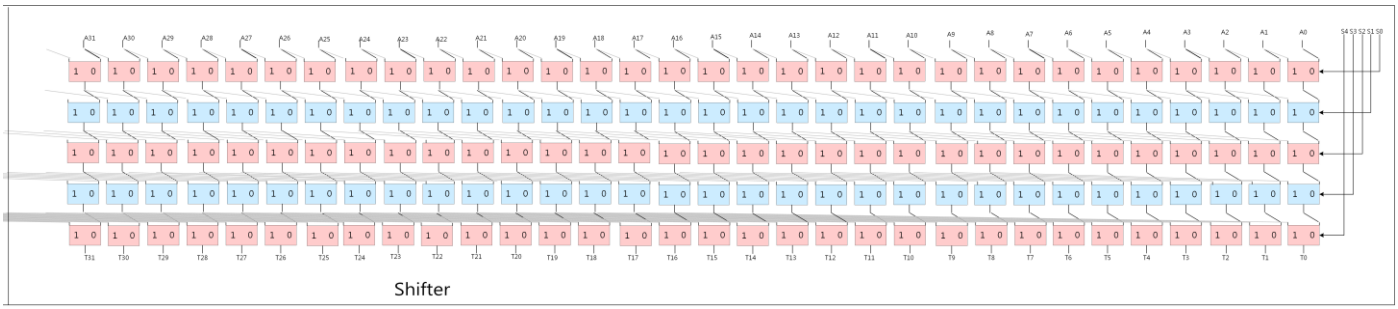
(二) Shifter(組合邏輯)：

1. 設計重點

Shifters 主要為實現 SLL，邏輯左移的運算，用 inShift 位移 in，將結果紀錄在 out。

2. 說明

- 用 5-bits 控制位移量，因為 2 的 5 次方最多可位移 31 bits，故以 5-bits 做為控制位移的 ，若要位移 32 bits 則全部補 0。
- 以一個 5bits 的線接收 inputB 的前 5bits 判斷位移量，從第 1 個 bits 判斷到第 5 個 bits 依序可位移 1、2、4、8、16



(三) DIVU：

1. 設計重點

DIVU 主要使用第三版除法器。第一版除法器與第二版除法器差別在於 ALU 從 64-bits 變 32-bits，第二版則跟第三版沒太大差別，只差在第三版除法器將商數暫存器合併到餘數暫存器。

2. 說明

我們依照第三版除法器的流程圖設計，先直接左移 1-bit，若 reset 是 1，則初始化讓 REM 暫存器 64-bits 為 0，我們還設計了一個 32-bits 的 resultSub，用來暫時存放 REM 左半邊與除數相減的結果(不是餘數)，利用 resultSub 的最高位元來判別相減結果，

resultSub[31] = 1 →相減結果是負數，代表該階段不可除，

resultSub[31] = 0 →相減結果正數或 0，代表該階段可除或整除，

LSB 因為與 resultSub 結果相反，所以我們~resultSub 來設定 LSB

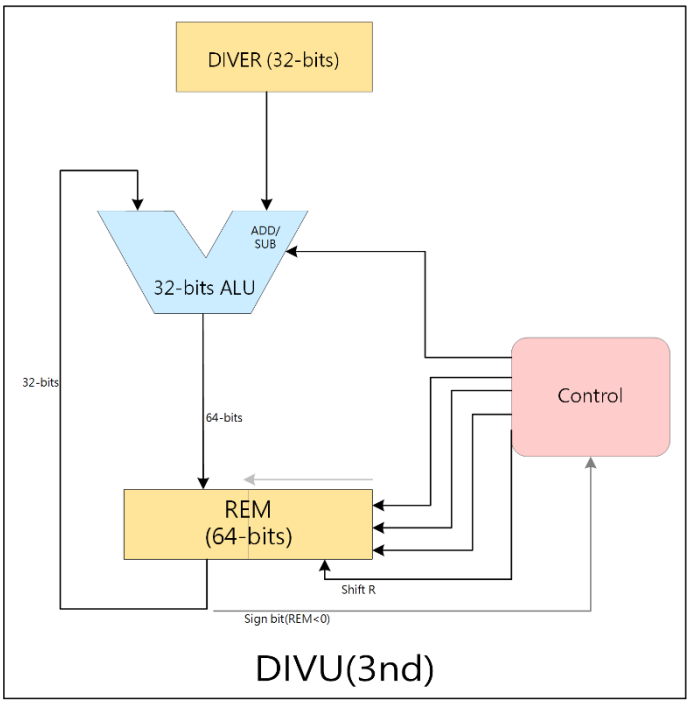
LSB = 0 → 不可除

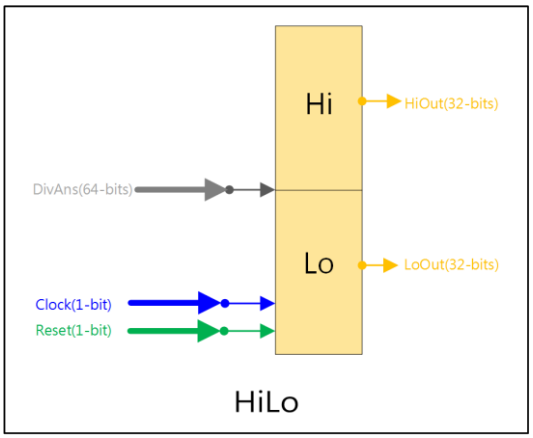
LSB = 1 → 可除

將 resultSub 設定給 REM 暫存器左半邊，再往左移 1 位元補 LSB(商數)。

執行此動作直到第 32 次完成(我們設定變數紀錄 count = 32)後將 REM 左半邊右移 1 位元，餘數才會是正確結果。

再將結果(64-bits)分別存到 32-bits 的 Hi(=REM 左半邊)、Lo(=REM 右半邊)。





(四) MUX(組合邏輯)：

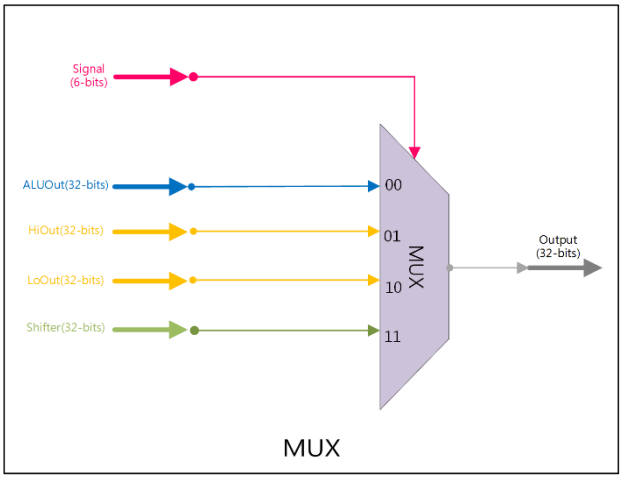
1. 設計重點

MUX 主要為整合所有結果，依據 ALU Control 給的訊號，決定輸出的結果。

2. 說明

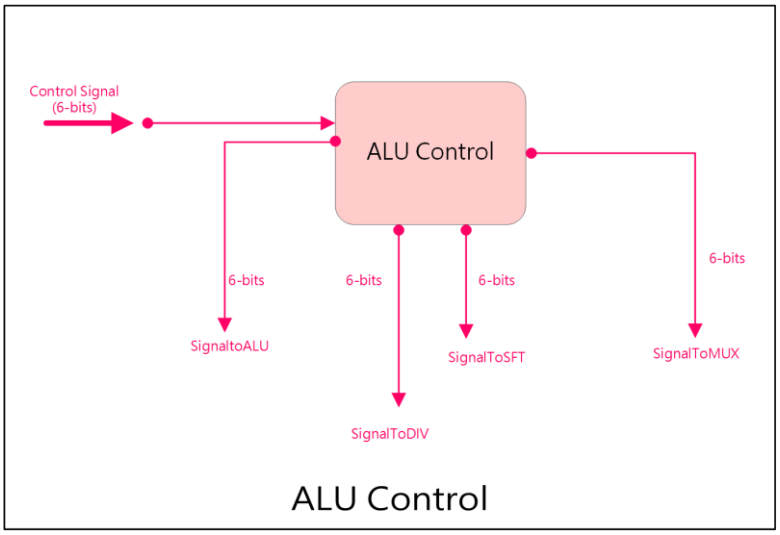
- 當傳入訊號是除法時要暫時讓 dataout 為 0，因為除法器並不會給予結果(結果將存至 HiLo 暫存器後，將訊號改為 Hi/Lo，傳進 MUX 傳出正確結果)，避免傳出上一道指令的計算結果。
- AND : 00 / 36 / 100100
  OR : 00 / 37 / 100101
  ADD : 00 / 32 / 100000
  SUB : 00 / 34 / 100010
  SLT : 00 / 42 / 101010
  MFHI : 01 / 16 / 010000
  MFLO : 10 / 18 / 010010
  SLL : 11 / 0 / 000000

  第一部分為 Selction，第二部分為其指令代碼，第三部分為 Signal，指令代碼轉 binary 後的值，我們利用 Signal 決定 Selection，再由 Selection 決定 dataout(輸出結果)。



(五) ALU_Control

ALU_Control 主要在控制 ALU、Shifter、MUX、DIVU，以輸入的 input 決定訊號，再傳至各元件，進行運算。



(六) TB_ALU

1. 設計重點

TB_ALU 主要目的為一個測試平台，從 Testbench 輸入，驗證各功能的運算結果是否正確。

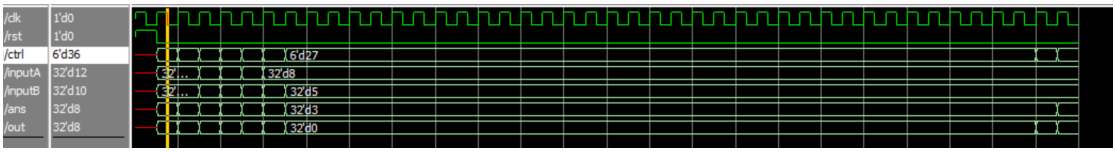
(七) TotalALU

將各項 module 建立並執行。

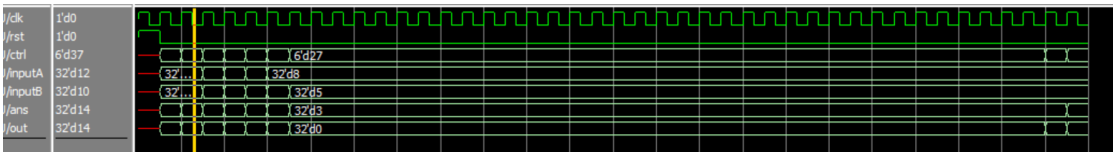
三、結果

第一個 clock 時，reset 初始化，等到第二個 clock 後才進行邏輯運算。

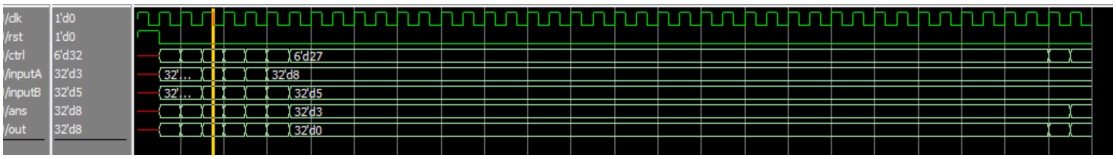
- 當 ctrl = 36 時，做 AND 運算，inputA = 12，inputB = 10，做完 AND 運算後得到 out = 8，而正確答案 ans 也等於 8，所以確定運算結果正確。



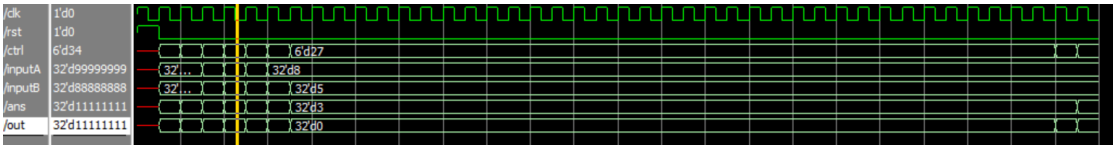
- 當 ctrl = 37 時，做 OR 運算，inputA = 12，inputB = 10，做完 OR 運算後得到 out = 14，而正確答案 ans 也等於 14，所以確定運算結果正確。



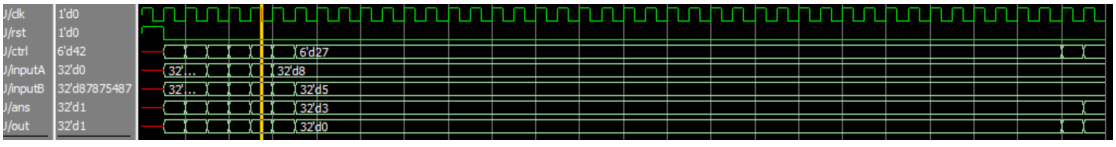
- 當 ctrl = 32 時，做 ADD 運算，inputA = 3，inputB = 5，做完 ADD 運算後得到 out = 8，而正確答案 ans 也等於 8，所以確定運算結果正確。



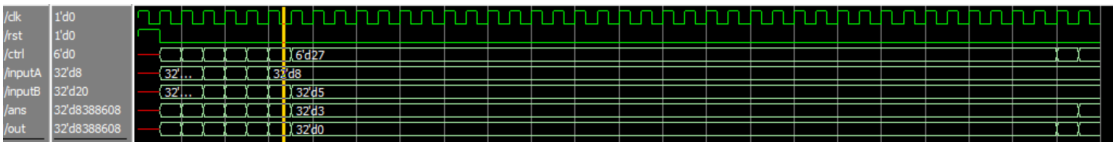
- 當 ctrl = 34 時，做 SUB 運算，inputA = 99999999，inputB = 88888888，做完 SUB 運算後得到 out = 11111111，而正確答案 ans 也等於 11111111，所以確定運算結果正確。



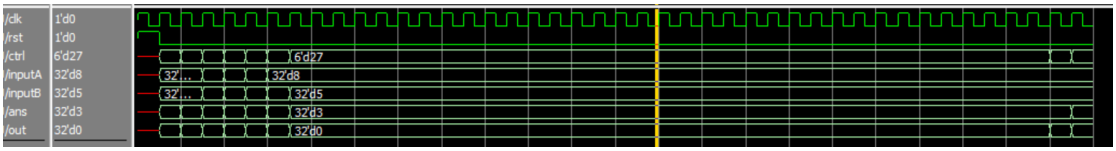
- 當 ctrl = 42 時，做 SLT 運算，inputA = 0，inputB = 87875487，做完 SLT 運算後因為 A<B，所以得到 out = 1，而正確答案 ans 也等於 1，所以確定運算結果正確。



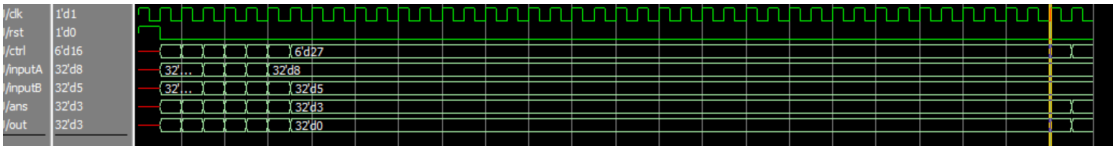
- 當 ctrl = 0 時，做 SLL 運算，inputA = 8，inputB = 20，做完 SLL 運算後得到 out = 8388608，而正確答案 ans 也等於 8388608，所以確定運算結果正確。



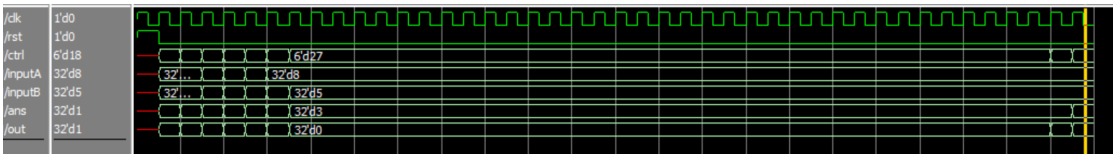
- 當 ctrl = 27 時，做除法運算，inputA = 8，inputB = 5，因為除法需要經過 32clk，才會得到最後運算結果(64-bits)，但避免 out 擷取上一道指令運算的結果，所以我們 out 暫時設定為 0，32 個 clk 後會再設定 ctrl。



- DIV 運算結束後會先將 ctrl 設定成 16，將 DIV 的結果前 32bit(餘數)設定給 Hi 暫存器，並將 out 設定為 3，而正確答案 ans 也等於 3，所以確定運算結果正確。



- Hi 結束後會先將 ctrl 設定成 18，將 DIV 的結果後 32bit(商)設定給 Lo 暫存器，並將 out 設定為 1，而正確答案 ans 也等於 1，所以確定運算結果正確。



四、討論

我們一開始討論的想法是照著講義上給的架構圖，像 ALU、Shifters、MUX、除法器則照著第三版除法器的流程圖，下去設計，接著利用老師上課所教的 verilog 內容，完成這次的 Midterm Project。

五、結論

我們的結論是在用 verilog 撰寫 ALU 時，遇到比較需要思考的問題像是構思要建立在 clock 時序邏輯上來實現循序邏輯，及 blocking 跟 non-blocking 的使用時機，還有要將思考方式從軟體的角度轉變成從硬體的角度等等，總而言

之，就是在過去寫 C 和 C++時所不會接觸到的，這些關於在硬體方面執行所要考慮到的問題，都是需要我們跳出原本對於程式的既定框架，去了解何謂電路的同步執行，也就是 non-blocking 的運作方式。而遇到比較小的問題像是對於 verilog 語法的不熟悉、、Shifters 忘記考慮到位移大於 32bits 時的狀況以及 Sub 的計算結果需要加一…，而隨著我們更加理解各項元件的功能、運作，就能一步步修正上面提到的小錯誤。

- 心得

  像上學期第一次學習組合語言一樣，接觸到一個全新的東西，多少有些不熟悉，儘管在電子實驗這門課已經操作過 verilog 以及 ModelSim，但在 coding 的難度上差距還是有點大，像電子實驗時我們寫過 Full Adder，而這次要寫的是一整個 ALU，難免會有些擔憂，不過上課時老師對於各項元件的講解都很仔細，讓我們理解當中的運作，怎麼實現其功能等等…，對我們 Midterm Project 的幫助很大，另外，除法器最讓我們頭痛，因為對於循序邏輯的概念十分陌生，因此我們花了很多時間跟心力在於理解上，而在老師的耐心教導，與組員們一起討論、和同學們互相交流過後，對於除法器就比較能駕輕就熟了> <。最後，因為第一次接觸計算機組織這門課，所以比較陌生而導致撰寫上有些小錯誤，但這些錯誤也讓我們收穫良多，幫助我們在之後的 final project 裡能避免重複失誤，而我們也在實作過後，更加熟悉這門課還有了解老師的上課內容。

  一個人走得快，但一群人走的遠，有一群組員們可以互相幫忙、讓彼此成長，是一件很棒的事。

六、未來展望

經過這次的 midterm project ，從一開始對這門科目完全不了解，到現在能夠設計出 ALU、Shifter…等各元件，感覺對計算機組織這門課又熟悉了點，希望未來在課堂上，能夠繼續學習更多計組相關的知識，也希望在未來能讓計算機組織成為我的一技之長。

七、分工

10527130 陳少洋：程式碼、Debug
10527140 初　元：部分程式碼、報告結果分析、Debug
10527132 林亞吟：撰寫報告分析、報告結果討論、Debug
10527135 張智欽：撰寫報告分析、報告結果討論、Debug