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

學生： 10527132 林亞吟

學生： 10527135 張智欽

學生： 10527140 初　元

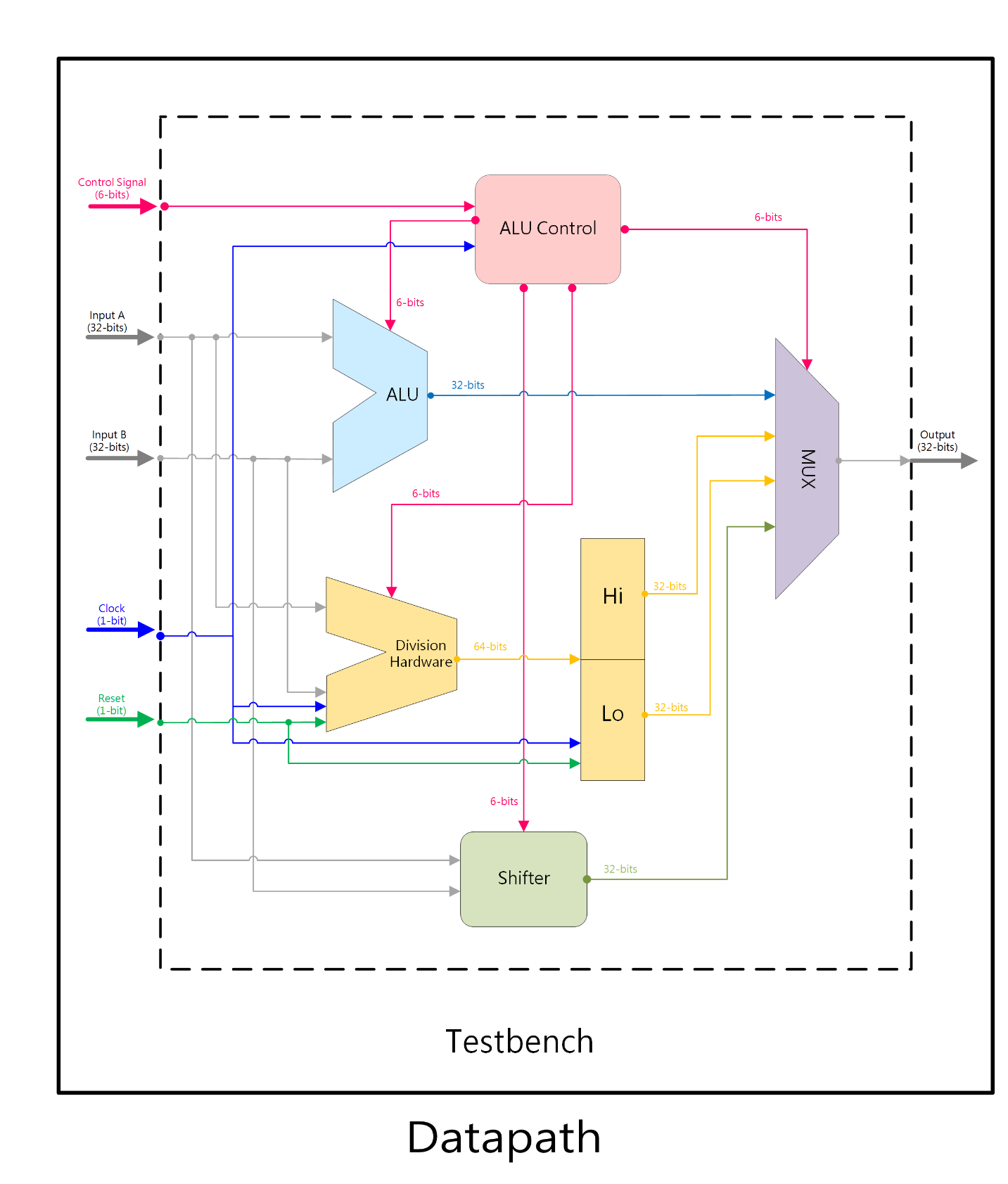
一、背景

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。



二、方法

1. **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。

1. **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

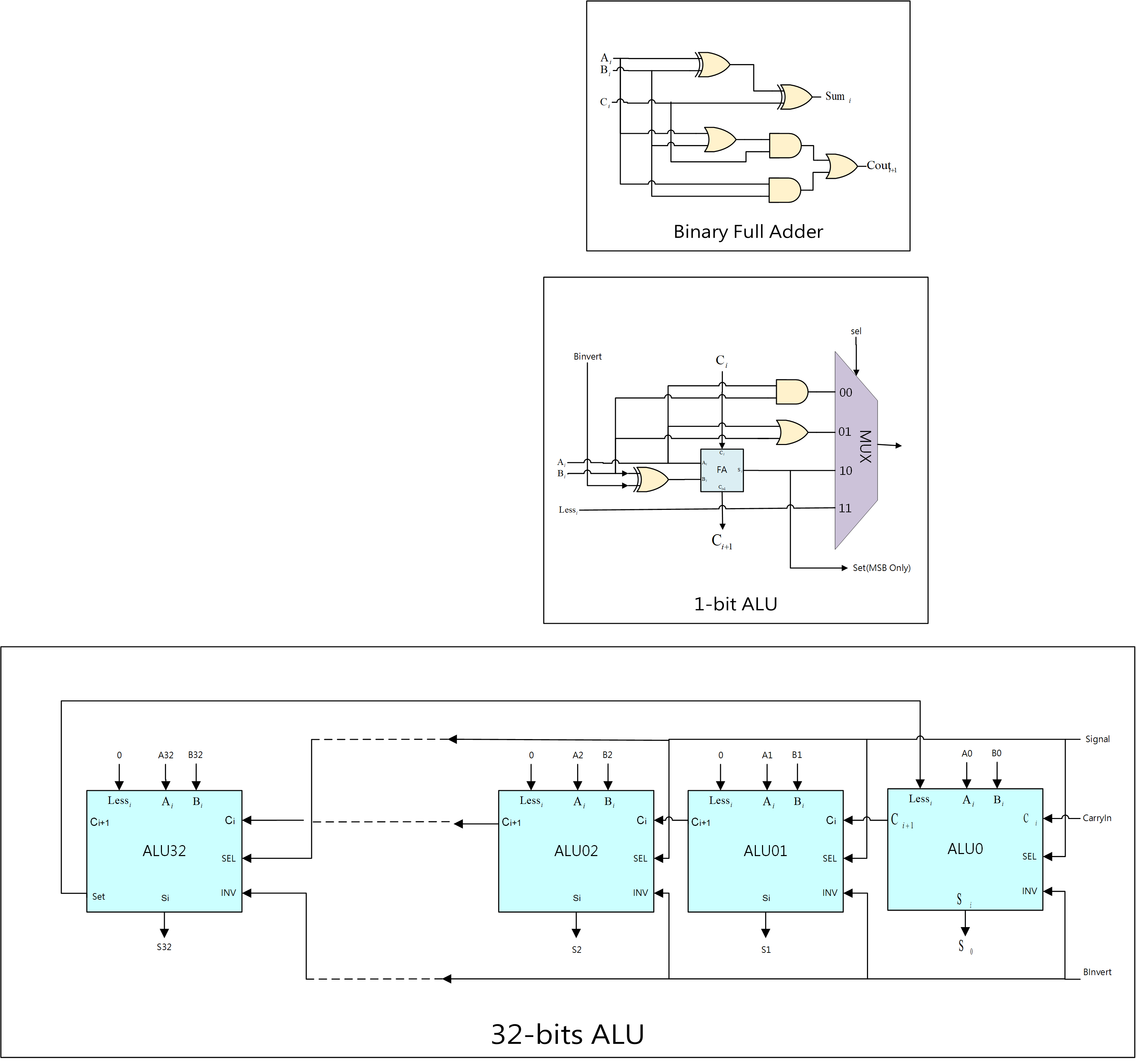
OR : 001 / 37 / 100101

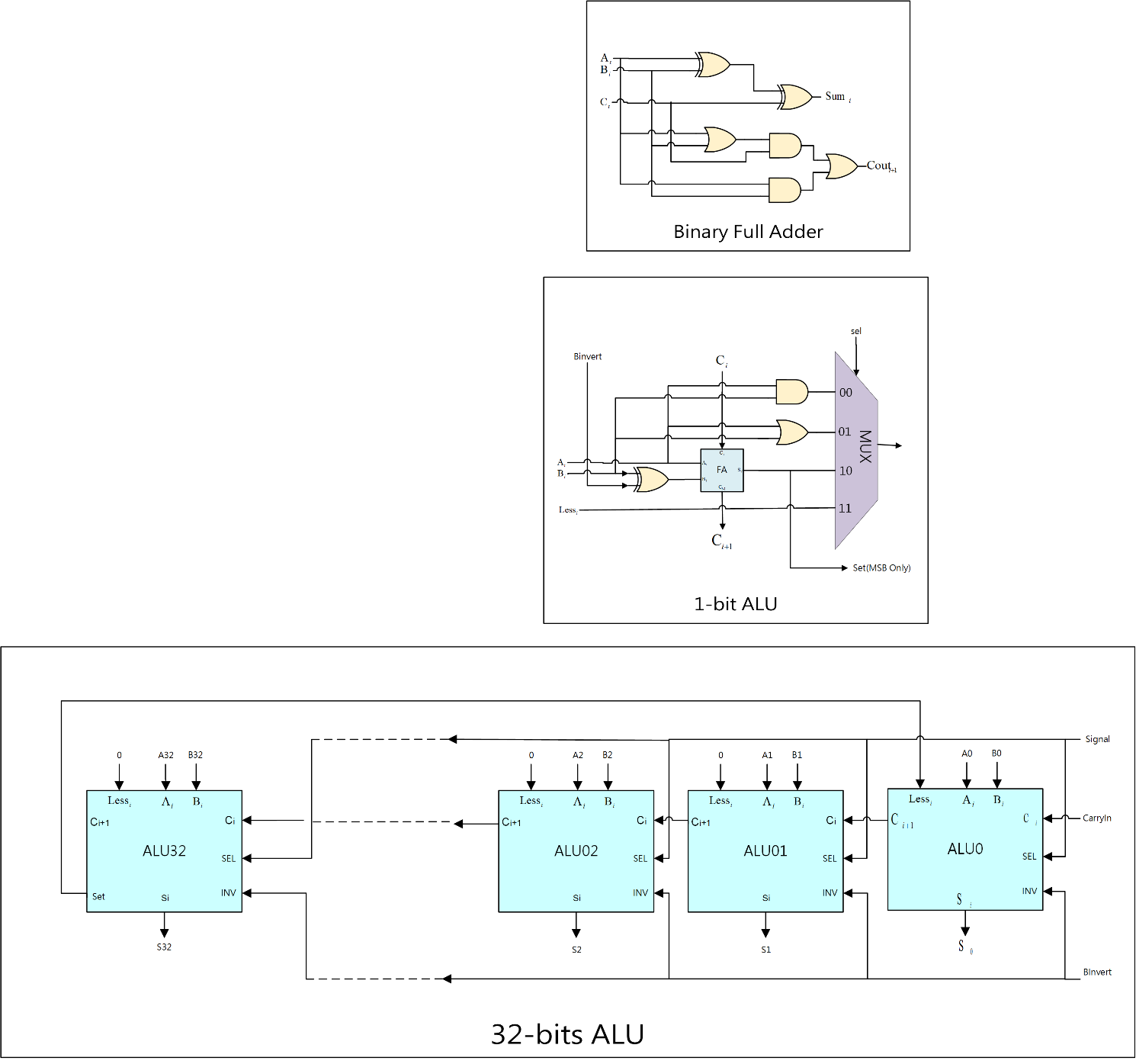
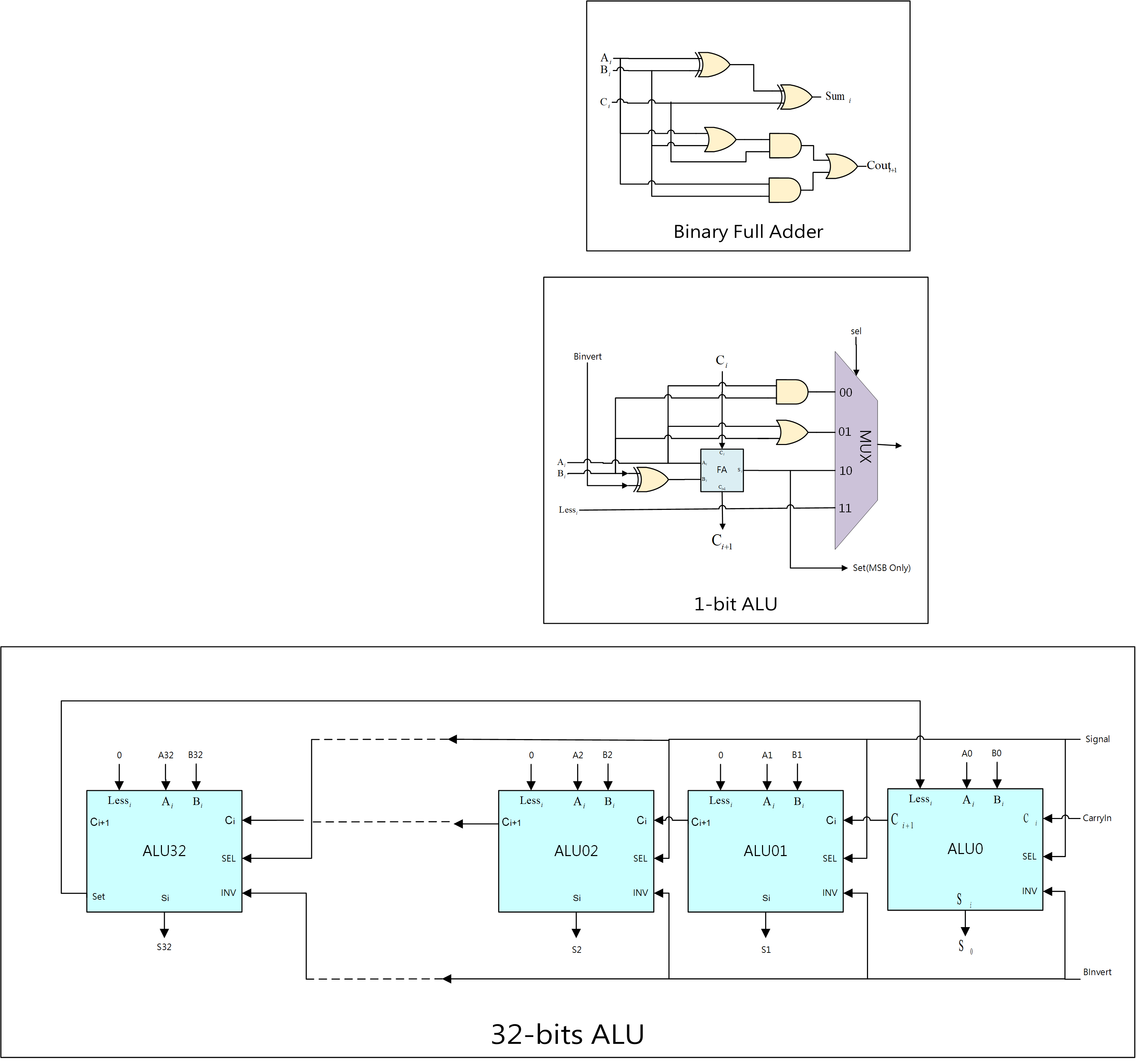
ADD : 010 / 32 / 100000

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。



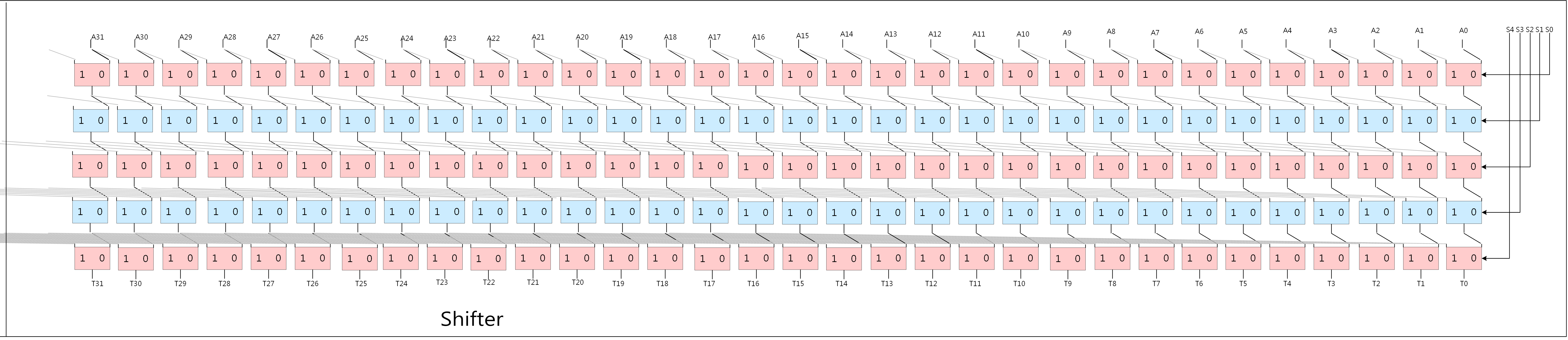


1. **Shifter(組合邏輯)：**
2. **設計重點**

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。

1. **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



1. **DIVU：**
2. **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。

1. **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，

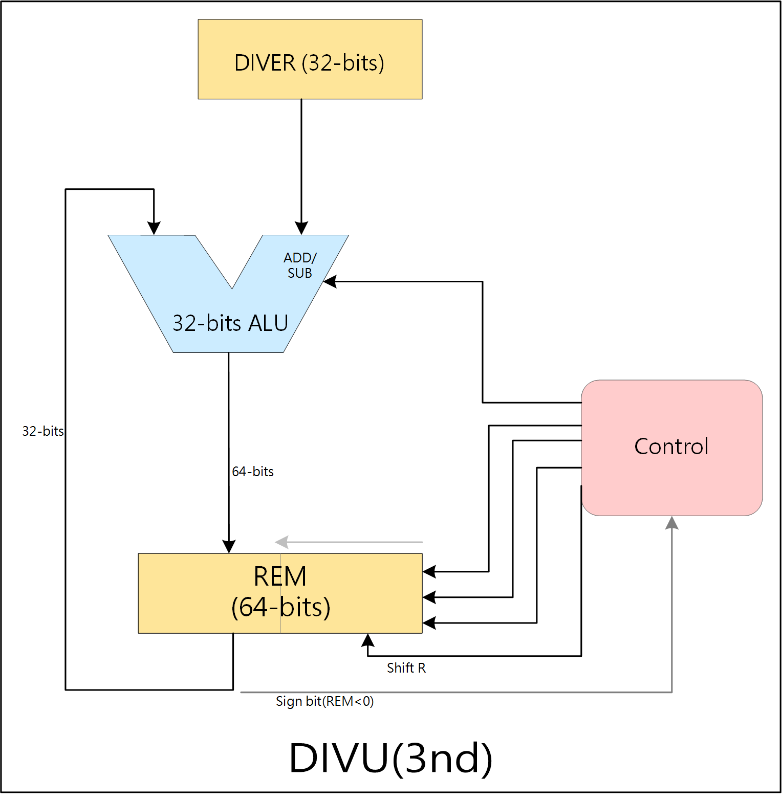
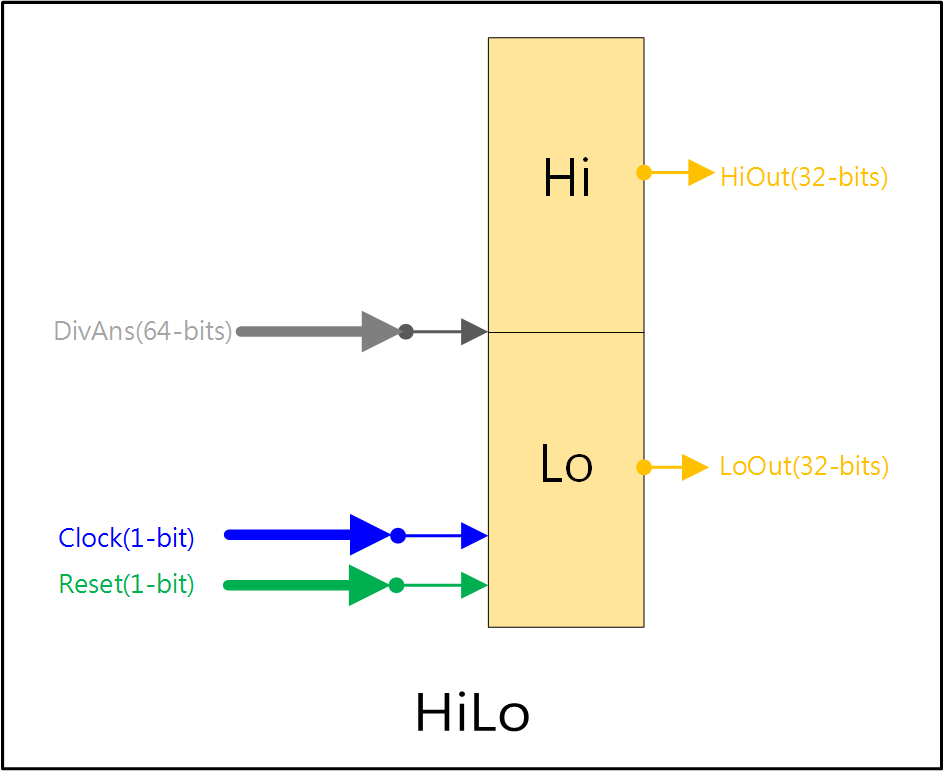
LSB因為與resultSub結果相反，所以我們~resultSub來設定LSB

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)。

1. **MUX(組合邏輯)：**
2. **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。

1. **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

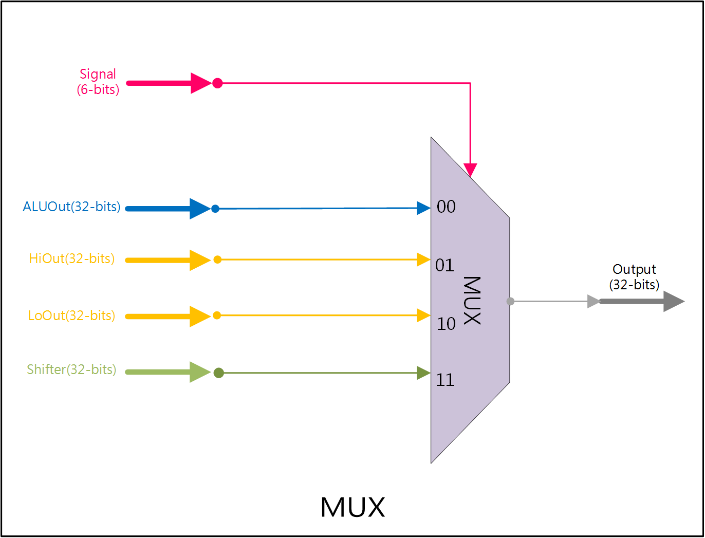
SUB : 00 / 34 / 100010

SLT : 00 / 42 / 101010

MFHI : 01 / 16 / 010000

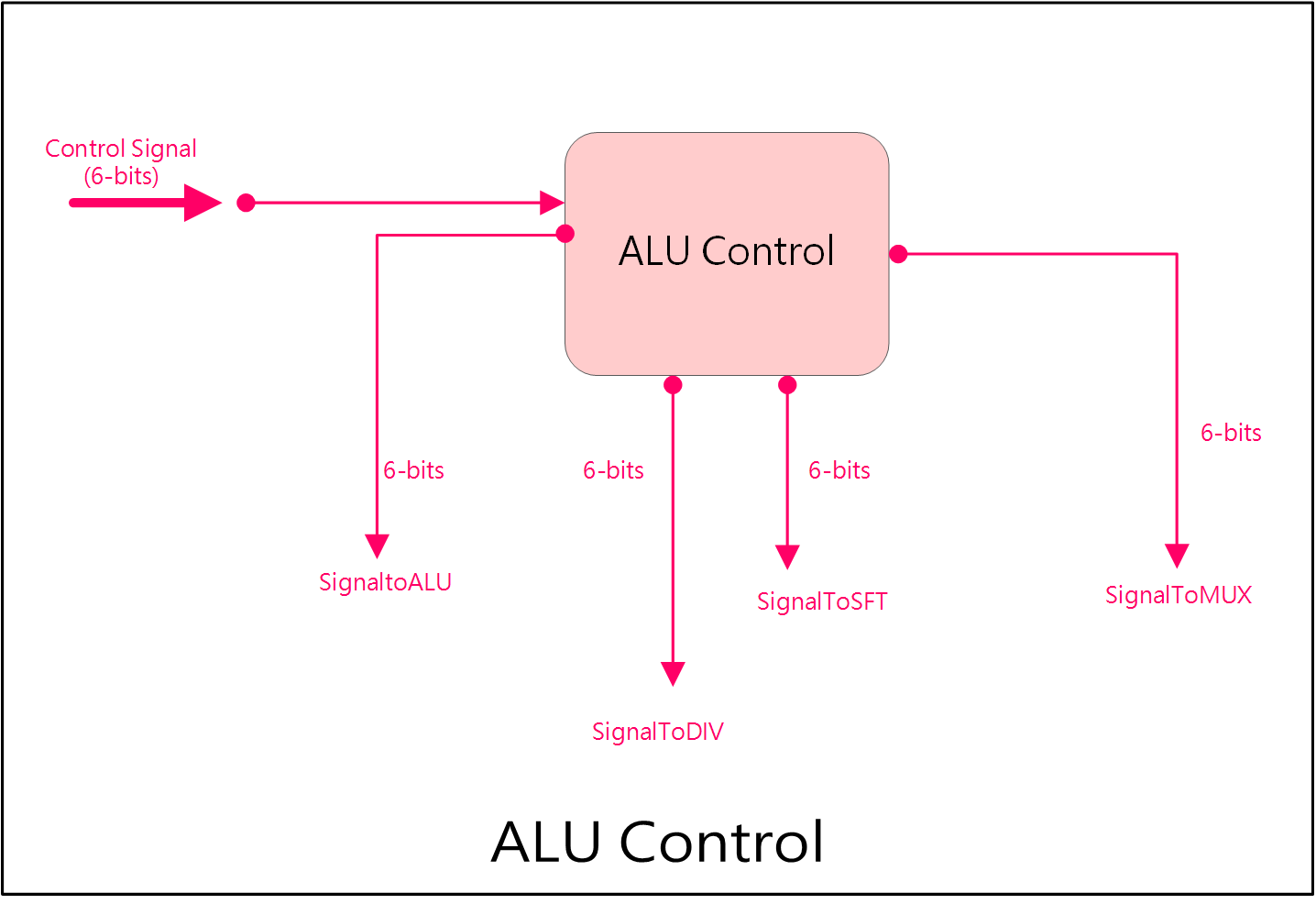
MFLO : 10 / 18 / 010010

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)。

1. **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。



1. **TB\_ALU**
2. **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。

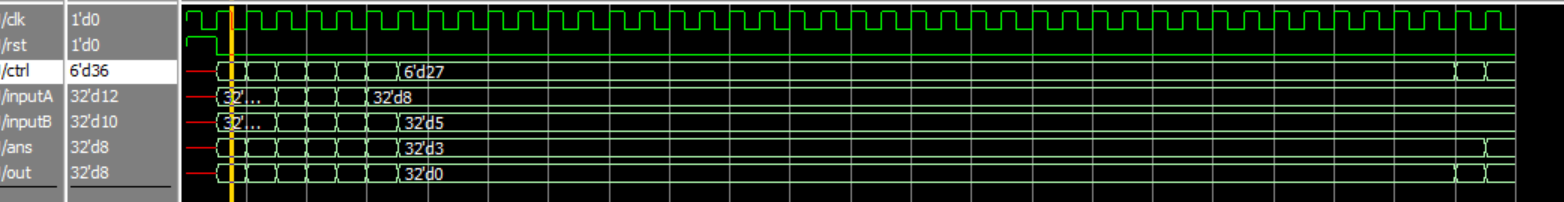
1. **TotalALU**

將各項module 建立並執行。

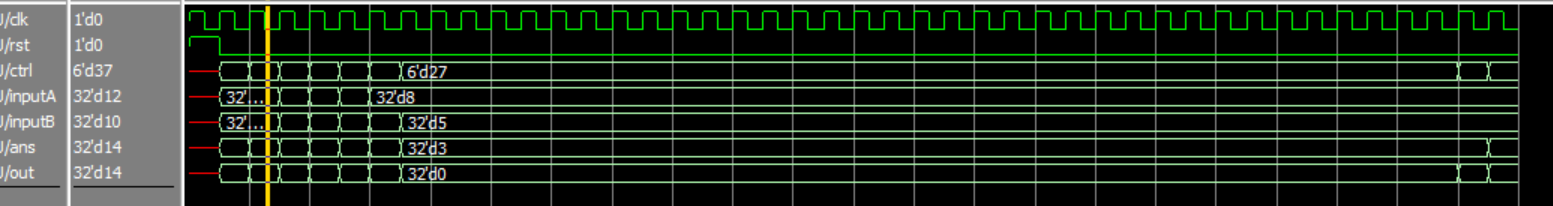
三、結果

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

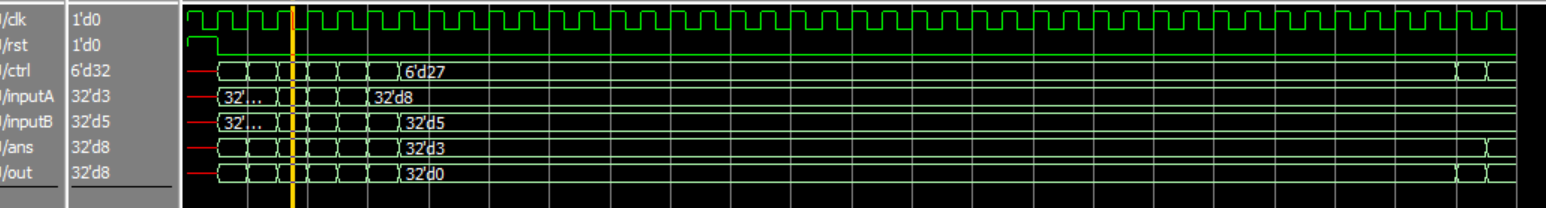
* 當ctrl = 36時，做AND運算，inputA = 12，inputB = 10，做完AND運算後得到out = 8，而正確答案ans也等於 8，所以確定運算結果正確。



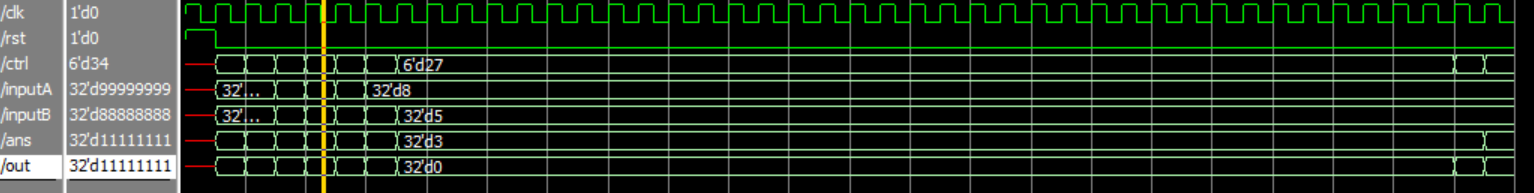
* 當ctrl = 37時，做OR運算，inputA = 12，inputB = 10，做完OR運算後得到out = 14，而正確答案ans也等於 14，所以確定運算結果正確。



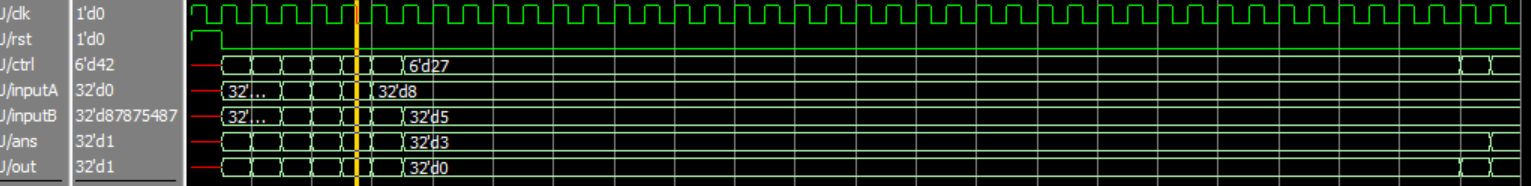
* 當ctrl = 32時，做ADD運算，inputA = 3，inputB = 5，做完ADD運算後得到out = 8，而正確答案ans也等於 8，所以確定運算結果正確。



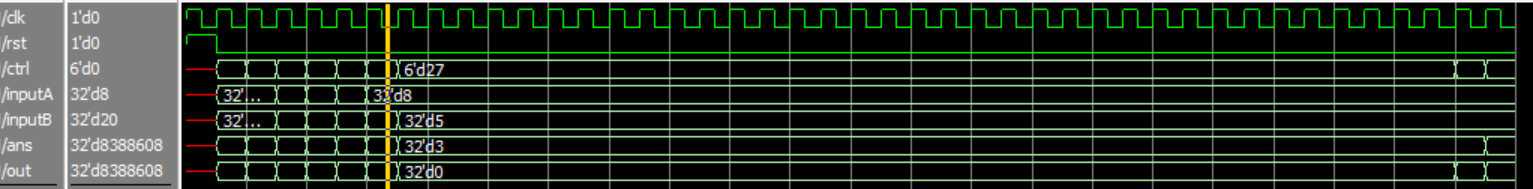
* 當ctrl = 34時，做SUB運算，inputA = 99999999，inputB = 88888888，做完SUB運算後得到out = 11111111，而正確答案ans也等於 11111111，所以確定運算結果正確。



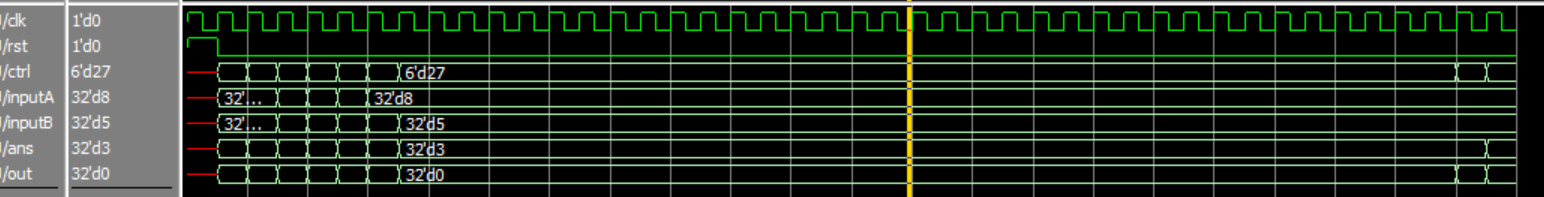
* 當ctrl = 42時，做SLT運算，inputA = 0，inputB = 87875487，做完SLT運算後因為A<B，所以得到out = 1，而正確答案ans也等於 1，所以確定運算結果正確。



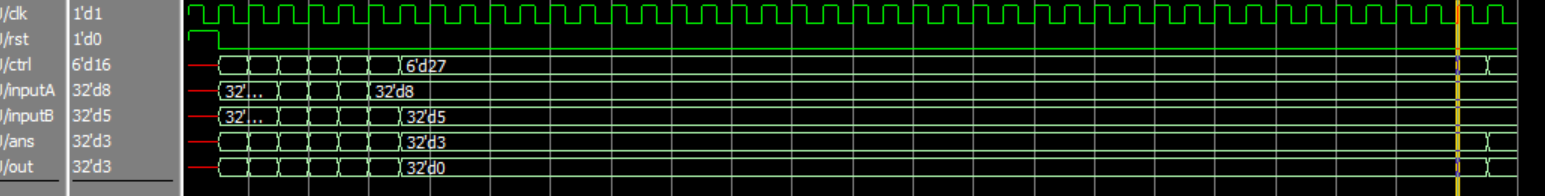
* 當ctrl = 0時，做SLL運算，inputA = 8，inputB = 20，做完SLL運算後得到out = 8388608，而正確答案ans也等於 8388608，所以確定運算結果正確。



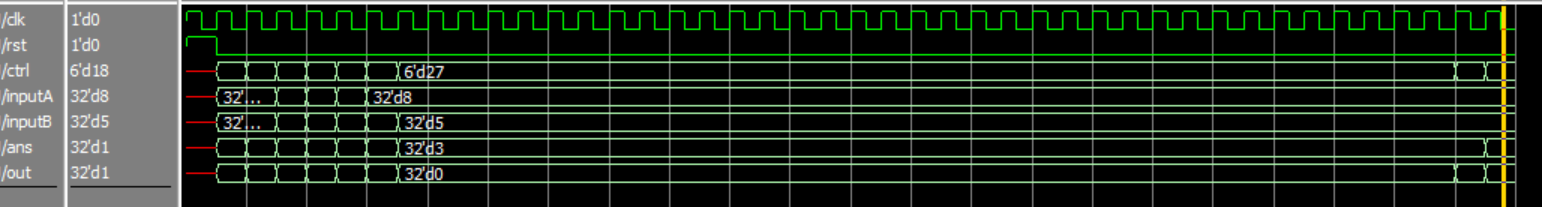
* 當ctrl = 27時，做除法運算，inputA = 8，inputB = 5，因為除法需要經過32clk，才會得到最後運算結果(64-bits)，但避免out擷取上一道指令運算的結果，所以我們out暫時設定為0，32個clk後會再設定ctrl。



* 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