計算機組織期末Project

第十二組

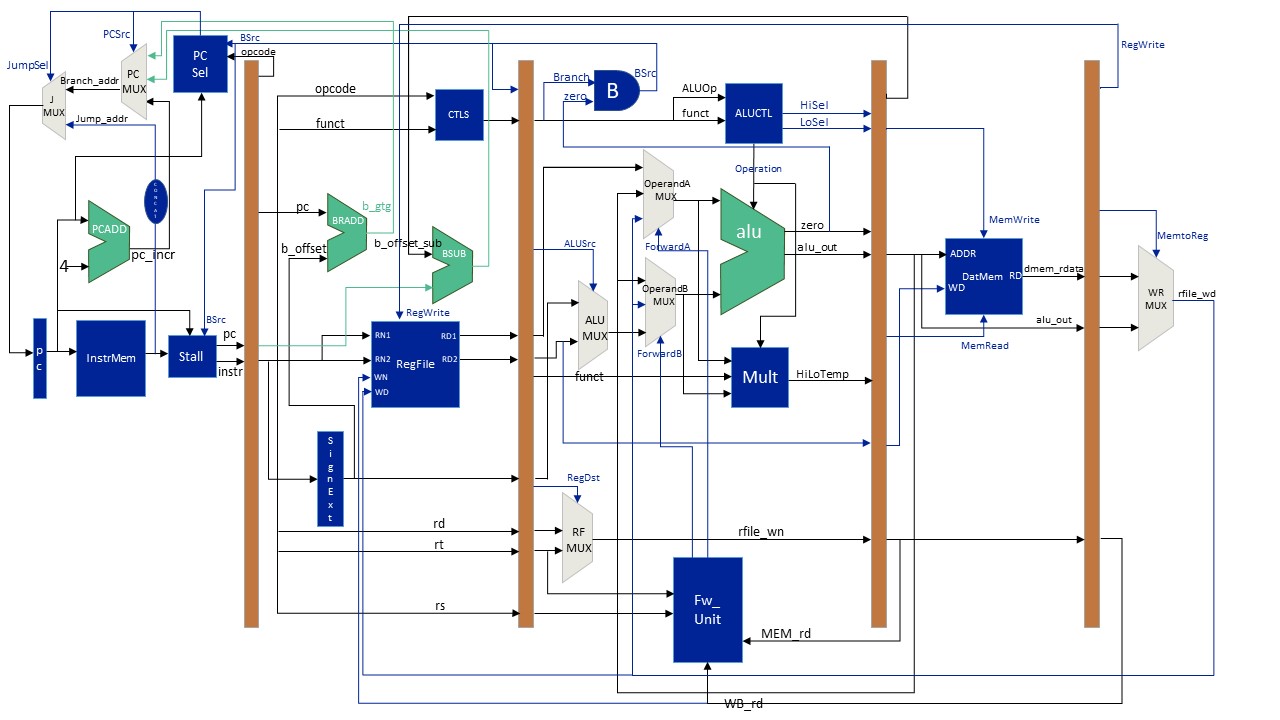
資訊二甲 10827101 林語潔

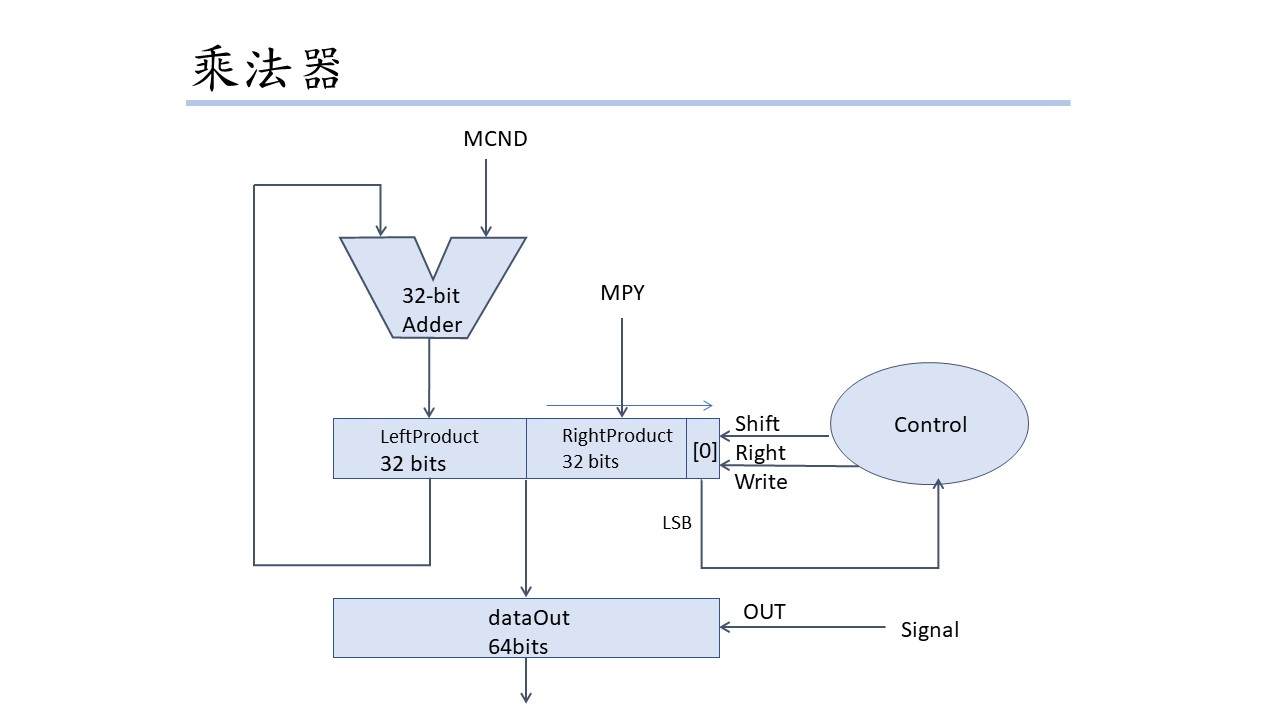
資訊二甲 10827119 陳宥心

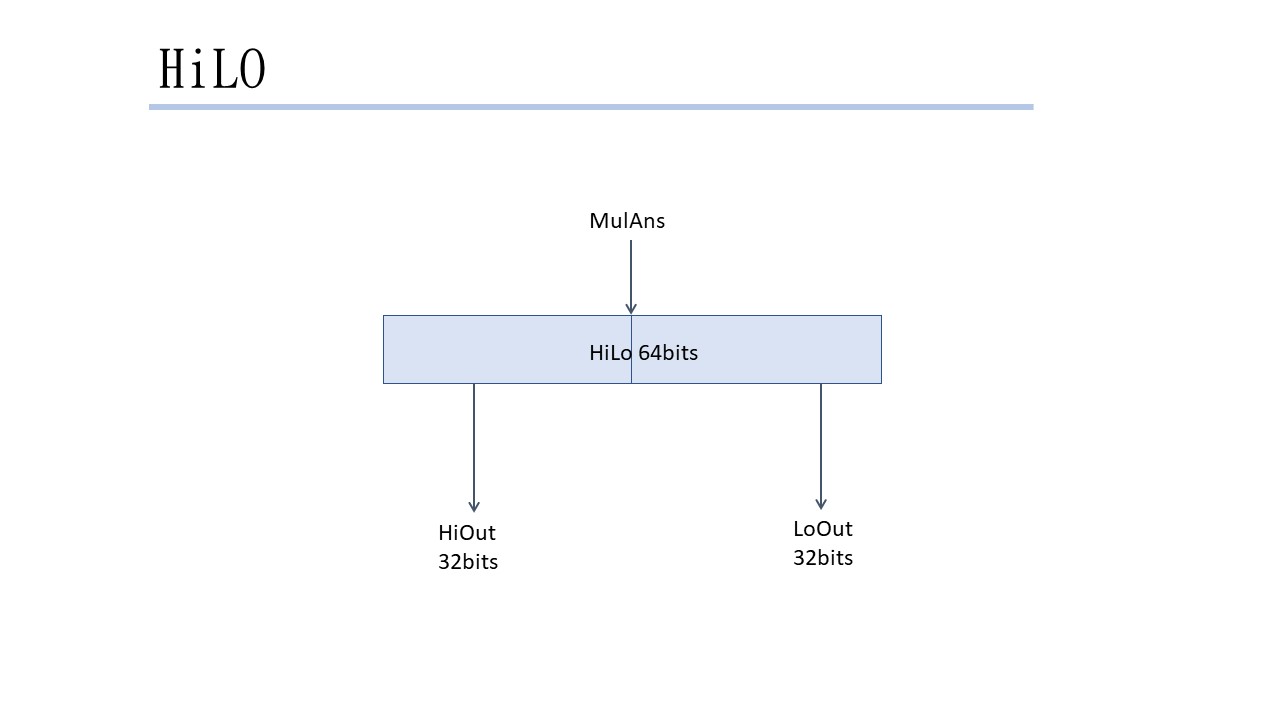
資訊二甲 10827137 李宜軒

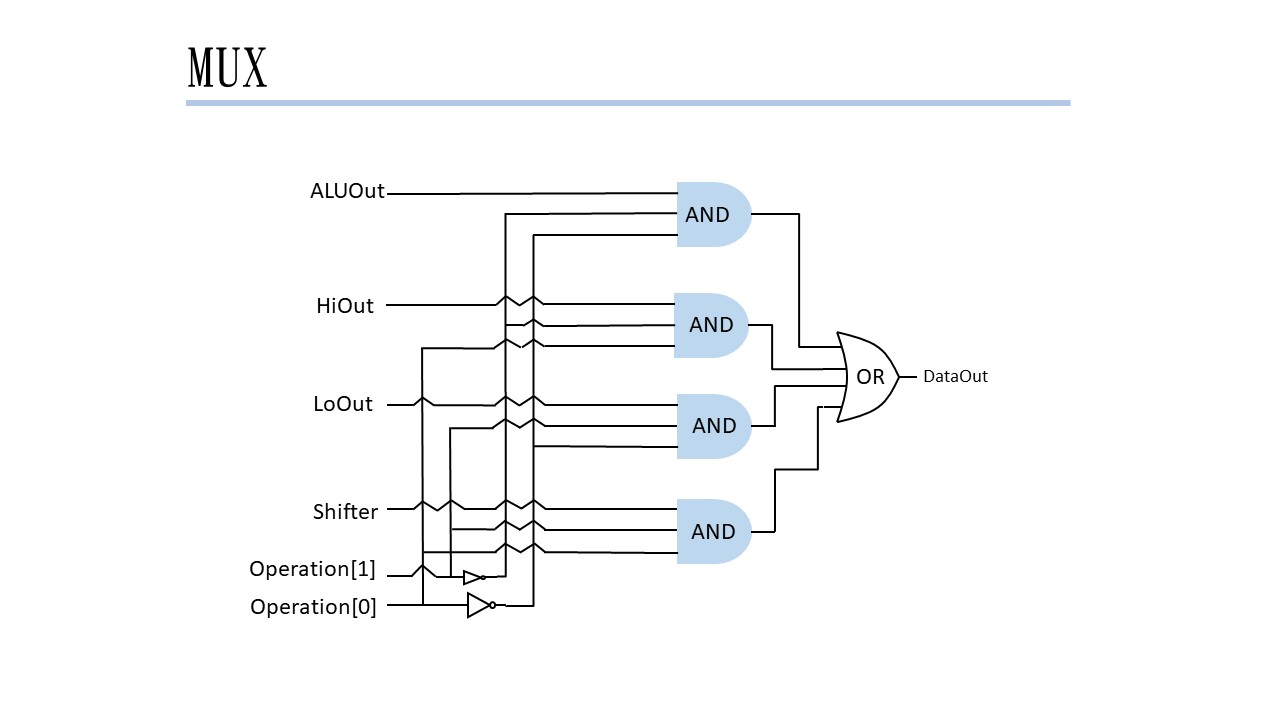
資訊二甲 10827145黃霈昕

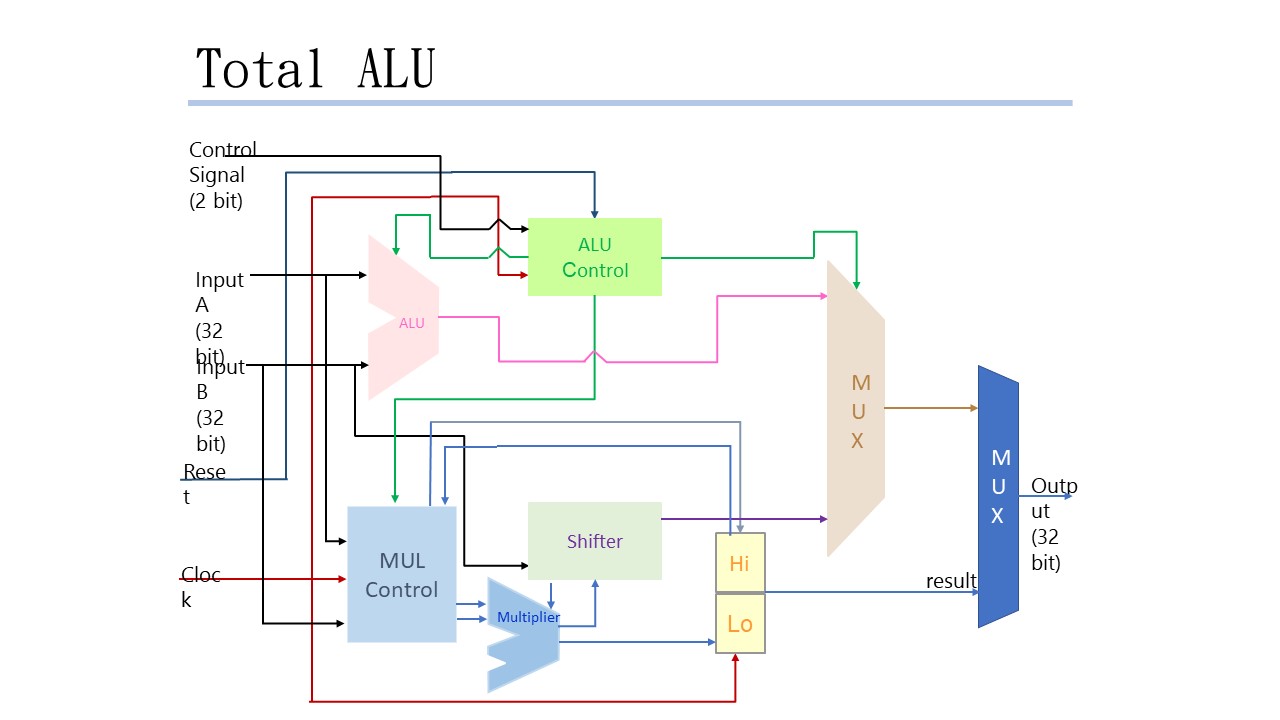
1. Datapath 與詳細架構圖

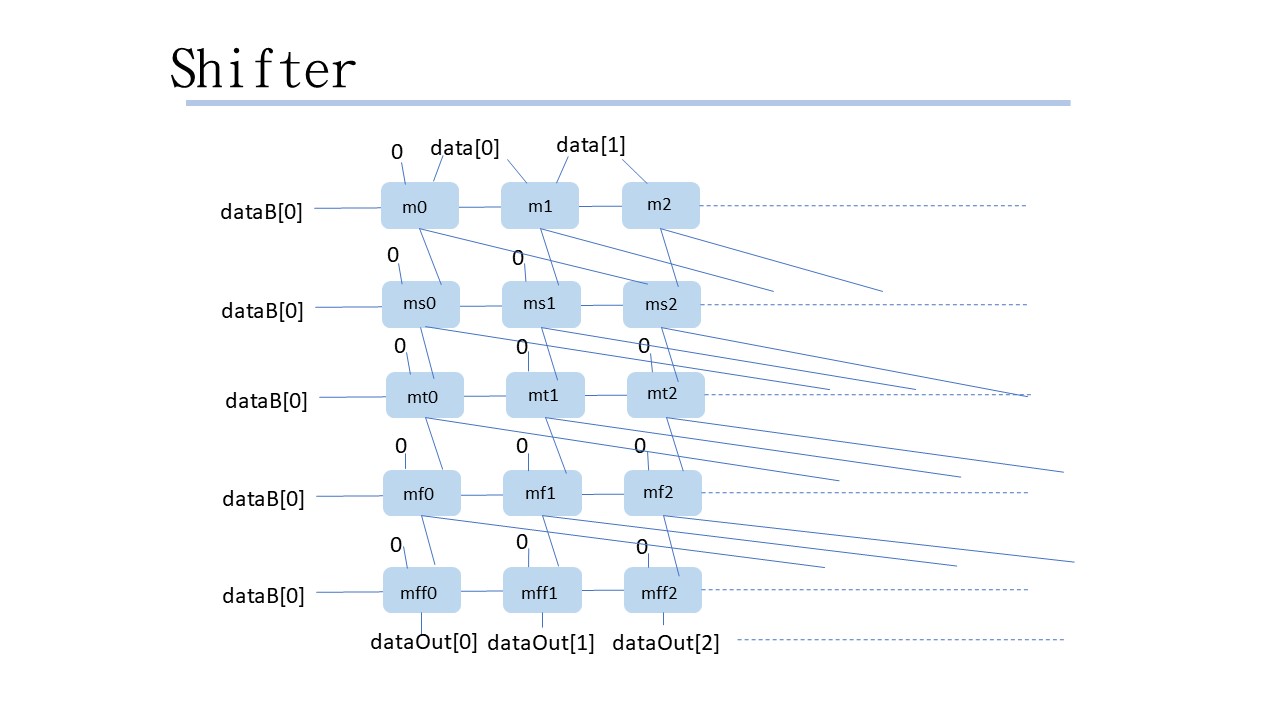


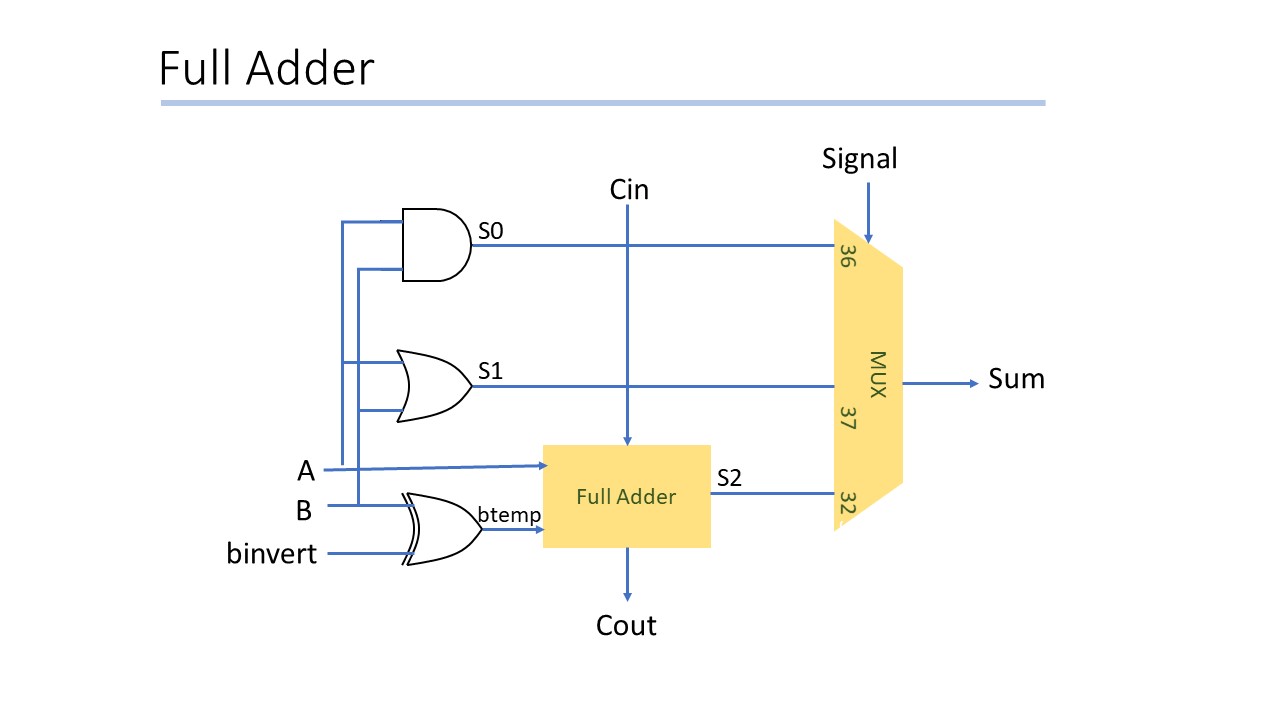


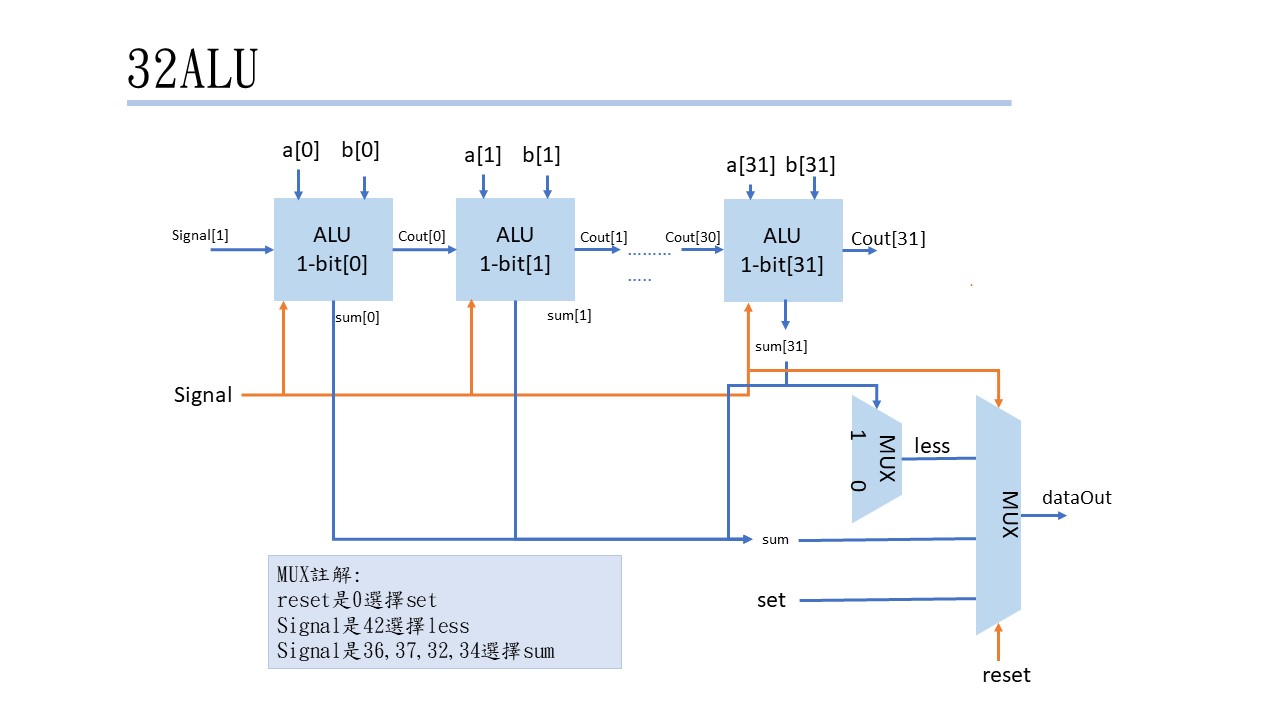












1. 設計重點說明

pipeline\_cpu : 實行5 stage pipeline cpu的模組

tb\_Pipeline\_cpu: 從檔案讀出測試資料並驗證所設計模組之正確性。

PCADD Pc = Pc + 4

PCSel 設定控制是否跳躍的訊號

PCMUX 選擇pc是否Branch

JMUX 選擇pc是否Jump

PC 設定pc\_try

InstrMem 根據MemRead和MemWrite來決定是否要讀檔或是寫檔

Stall 若是branch指令，則兩個訊號皆設為0，也就是空一個cycle

IFTOID : 備份2個訊號，若reset訊號為1，則全清為0

--------------------ID--------------------

RegFile 設定register file中RD1, RD2的訊號

SignExt 將16位元的有號數擴充成32位元

BRADD 若是遇到branch指令，計算出下一道指令位址

BSUB 若branch指令不會跳，計算出下一道指令位址

CTLS 根據opcode來判斷指令，並設定其指令的control signals

IDTOEX 備份20個訊號，若reset訊號為1，則全清為0

--------------------EX--------------------

ALUMUX 根據sel訊號，選擇出輸出訊號

ALUCTL 判斷指令，選擇出接到ALU的 sel訊號

OprandAMUX 選擇出接到ALU的input訊號

OprandBMUX 選擇出接到ALU的input訊號

total\_alu 裡面包含32 bits ALU和Shifter

ALU 1-bit ALU : 1-bit ALU裡總共會做四種運算，分別是AND，OR，ADD，SUB，前兩項能夠直接用邏輯閘實現，後兩項則透過五個邏輯閘合成一個一位元的全加器。其中加法和減法有兩個差異，一個是傳進全加器的第二個運算元要取NOT，這裡我們用控制訊號第二位元和原來的第二個運算元取XOR，舉例:如果是加法，控制訊號第二位元會是0，也就是說取完XOR的值不會改變，減法則相反。另一項則是減法的Carry in第0位元為1，這裡我們直接利用32位元的ALU把值傳進來。

四項運算都做完後，依照控制訊號傳回指定的結果。

32-bits ALU是由32個1-bit ALU所組成的，傳進的參數有兩個運

算元a、b，Carry in、Carry out、Sum以及控制訊號Signal。首先

第0位元(alu0)的Carry in為控制訊號的第二位元，因為減法的

Carry in為1，正好是減法的Signal[1]。從第1位元(alu1)開始的

Carry in就是前一位元的Carry out。整個32位元運算都結束後，

取Sum的最高位元，如果最高位元為1，代表結果是負的，就把

SLT設為1。最後再依照控制訊號傳回指定的結果。

Shifter 2^5= 32，因此可以得知32 bits的shifter需要五層，每層的一位

量分別為，第一層可移位2^0 = 1bit，第二層可移位2^1= 2 bits，

第三層可移位2^2= 4 bits，第四層可移位2^3 = 8 bits，第五層可

移位16 bits。

MUX2 to 1為基本的一個可以執行移位的位元組，利用32個

MUX2 to 1連接，可以得到一層32 bits，利用組合邏輯製作出32

bits的shifter。

B 判斷是否要branch

Mult 呼叫Multiplier和HiLo，並設定rst訊號Multiplier

傳入欲做運算之dataA、dataB兩數，一開始將32位元寬dataB

放入64位元暫存器prod中最低32位元，接著判斷prod第一位

元，若第一位元為1則將prod最高32位元與dataA相加存入

prod最高32位元再右移1位元，若第一位元為0則直接右移1

位元。32次後，在下一個clock敲起時(即第33個cycle)，將

prod的值存入暫存器中再將值傳入HiLo暫存器中。

HiLo 傳入乘法器之運算結果並存入HiLo暫存器，由最高32位元輸出

的值即為HiOut，最低32位元輸出的值即為LoOut。

RFMUX 根據sel訊號，選擇出WN

Fw\_Unit 輸入register number，設定ForwardA和ForwardB的值

EXTOMEM 備份17個訊號，若reset訊號為1，則全清為0

--------------------MEM--------------------

DatMem 根據MemRead和MemWrite來決定是否要讀出或是寫入記憶體

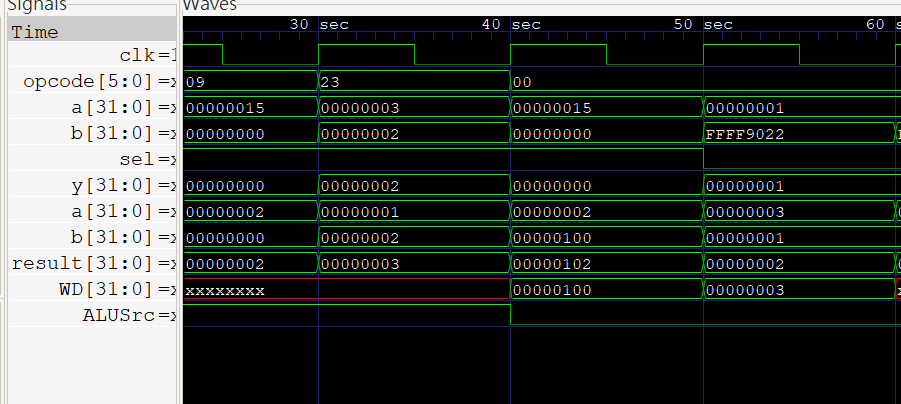
MEMTOWB 備份8個訊號，若reset訊號為1，則全清為0

--------------------WB--------------------

WRMUX 選擇出要WD的訊號

1. Icarus Verilog驗證結果與Waveform輸出圖形

* ADDIU $s2, $s0, 2 ( 1 + 2 = 3 )



ALSUSrc跳1

1. 跳wul4

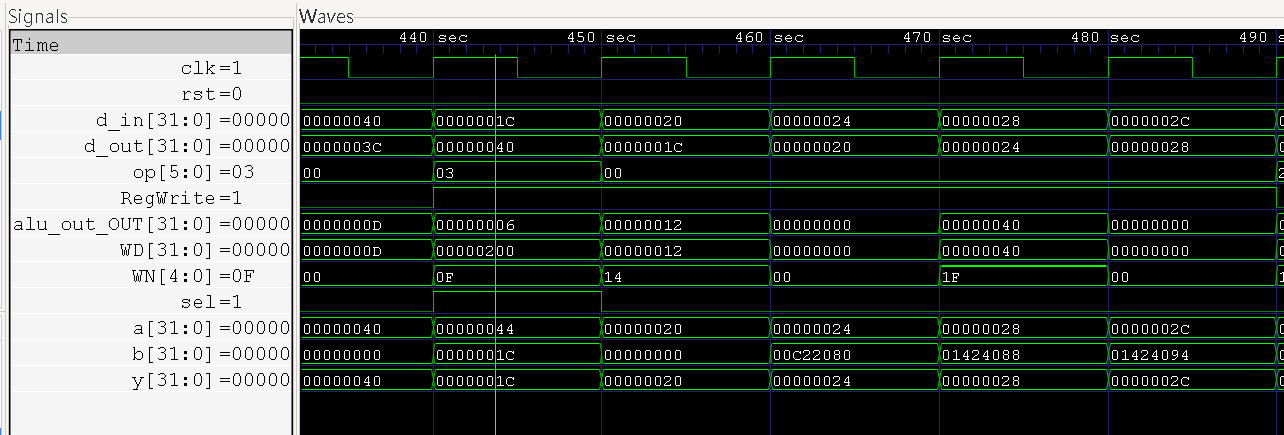
讀opcode

Offset當OprandB

送回回

opcode是 0x9，ALUSrc 設為1，拿offset當operandB，將ALU運算完的結果在WB階時寫回WD。

* jal 7



讀opcode

RegWrite=1

PC是1C

送回

Sel = 11 1

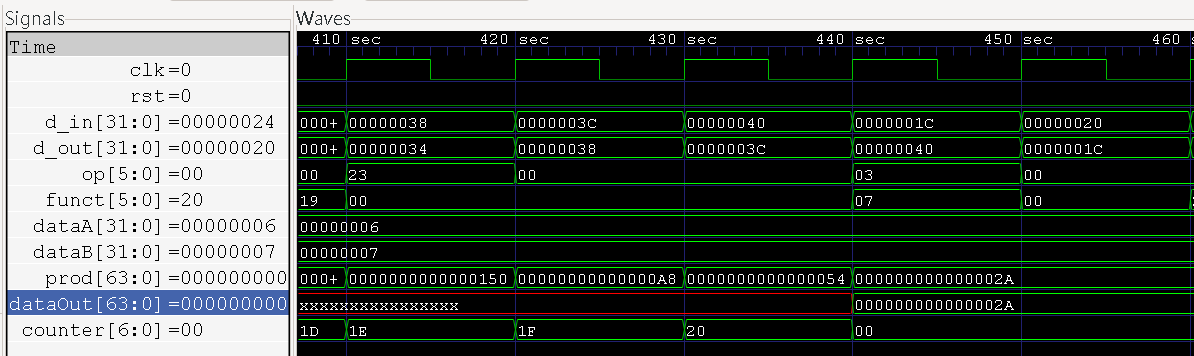
選擇1C

PC是0x 1C，opcode是0x03，代表執行JAL。RegWrite設為1，表示會寫入暫

存器，最後會將return address存入$ra。sel設為1並送訊號給JMUX，JMUX

會選擇jump\_addr(要jump到的PC)做為下一個PC。

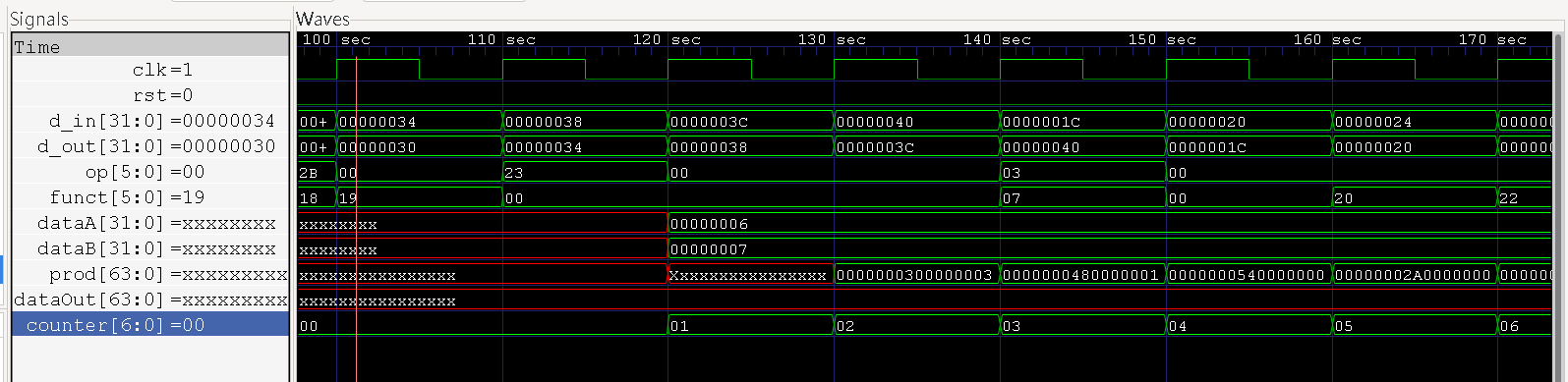
* MULTU $s5, $s6



Counter = 32

(HEX = 20)

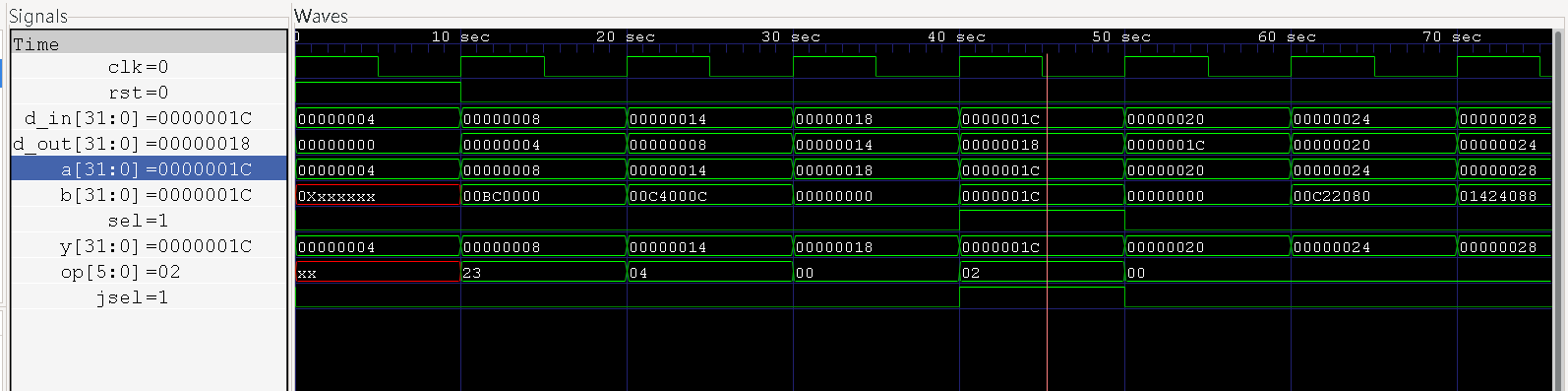
送出



讀funct

opcode是0x00， funct是 0x19，從counter為0開始為乘法器計算的第一個cycle，計算0x6 \* 0x7，在第33個cycle 將結果送入HiLo暫存器。

* j 7

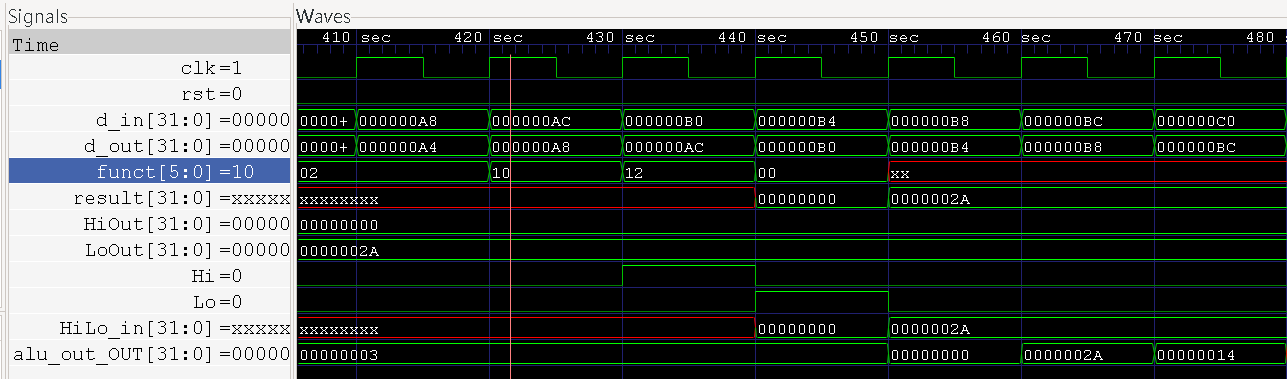


JMUX sel = 1

JMUX jsel = 1

opcode是0x2，PCSel會將 JMUX sel設為1並送訊號給JMUX，JMUX會選擇jump\_addr(要jump到的PC)做為下一個PC

* MFHI $s0



讀funct

Hi = 1

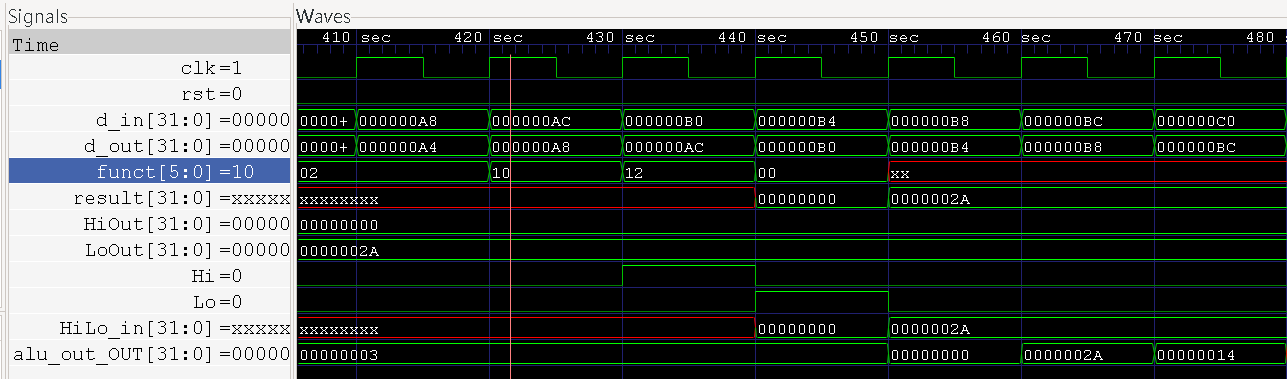
送回

funct輸入010000，因為wafeform上顯示的數字是16進位，顯示10。

因為MFHI指令是要將Hi放到$s0，在MultControl判斷出指令後，將Hi暫存

器的值送回給RegFile以存入暫存器。

* MFLO $s1



讀funct

Lo = 1

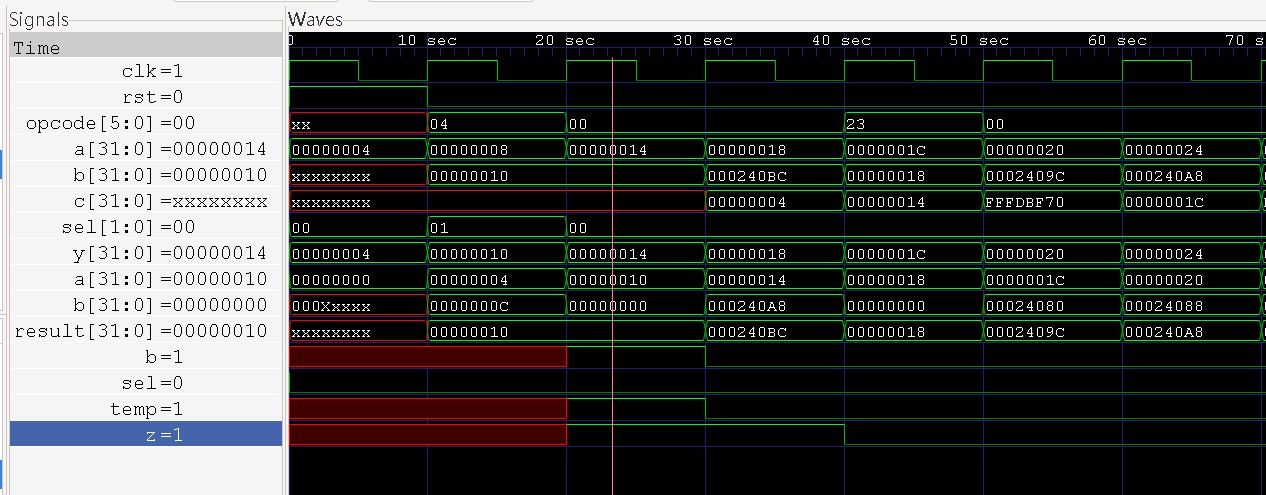
送回

funct輸入010010，因為wafeform上顯示的數字是16進位，顯示12。

因為MFHL指令是要將Lo放到$s1，在MultControl判斷出指令後，將Lo暫存

器的值送回給RegFile以存入暫存器。

* beq $s1, $s1, 3 ( result = PC + offset = 10 )



讀opcode

Zero = 0

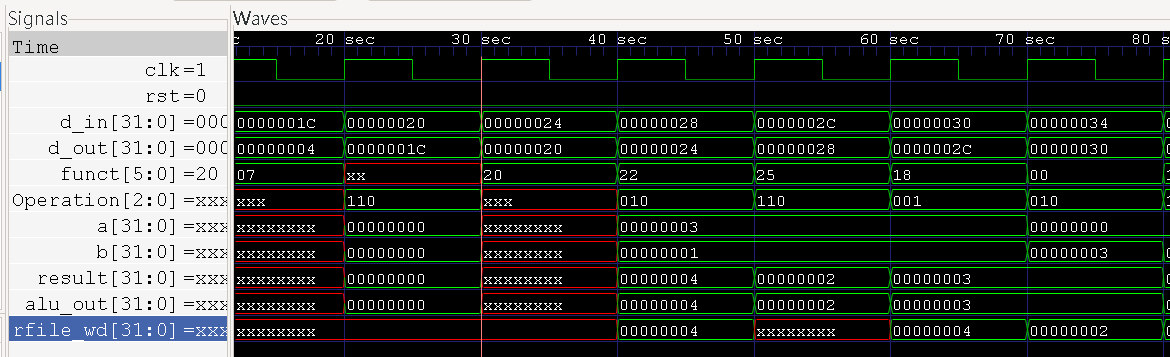
b = 1

sel = 0

不用跳回

opcode是0x04，由於我們預設遇到beq會branch，所以將第一個sel設為01選擇pc + 偏移量(要跳躍的位址)為下一個pc，又Zero and banch是1所以將第二個sel 設為0代表不用跳回beq指令的下一道。

* add $s2, $s0, $s2



讀funct

Operation = 010

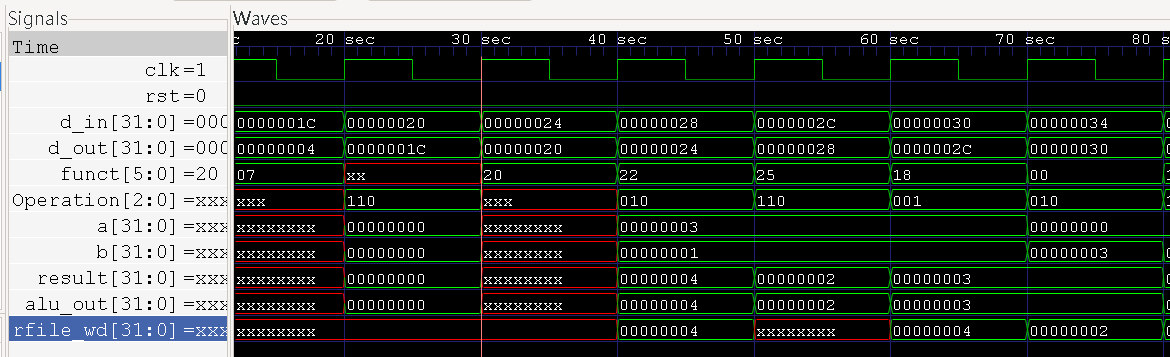
a = 3, b = 1

3 + 1 = 4

送回

opcode是0x0，funct是0x20，在EX階Operation會送訊號010給ALU做加法運算，0X3+0X1 = 0X4 ，再隔兩個cycle，WB階送回RegFile以存入暫存器。

* sub $s2, $s0, $s2



讀funct

Operation = 110

a = 3, b = 1

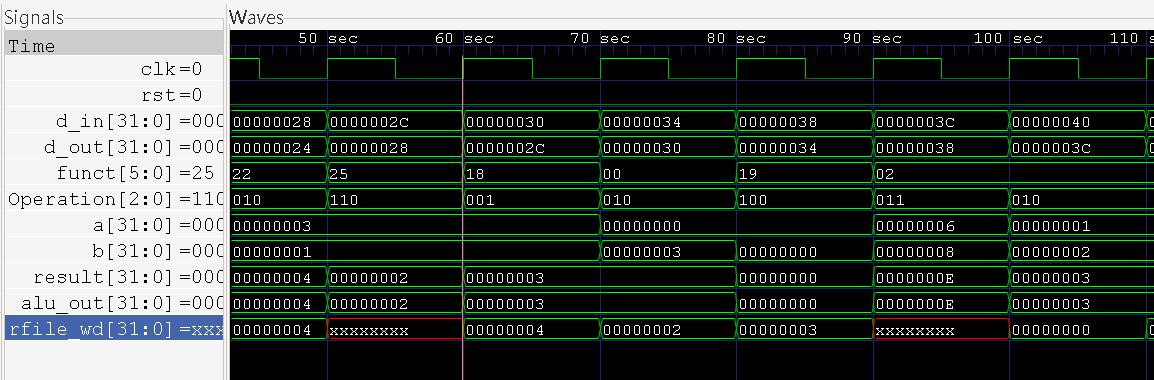
3 - 1 = 2

送回

opcode是0x0，funct是0x22，在EX階Operation會送訊號110給ALU做減

法運算，0X3-0X1 = 0X2 ，再隔兩個cycle，WB階送回RegFile以存入暫存器。

* or $s2, $s0, $s2



讀funct

Operation = 001

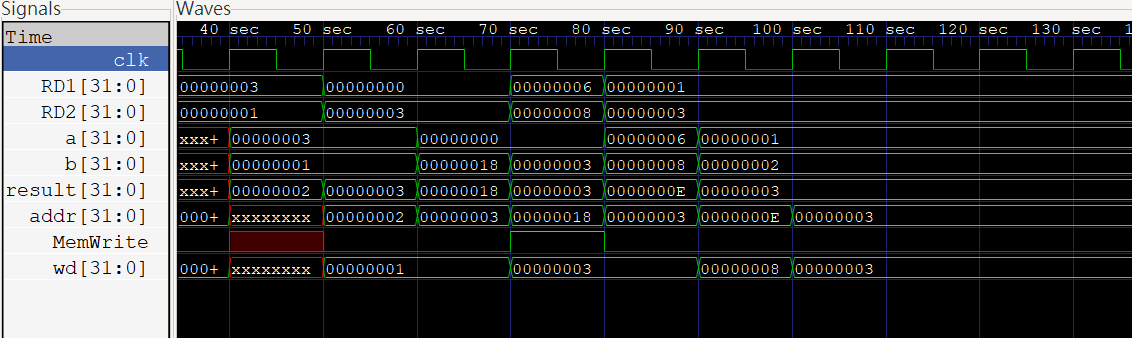
a = 3, b = 1

3 or 1 = 2

送回

opcode是0x0，funct是0x18，在EX階Operation會送訊號001給ALU做OR，0X3 OR 0X1 = 0X2 ，再隔兩個cycle，WB階送回RegFile以存入暫存器。

* sw $zero, $s2, 24



Rd1 = 0, Rd2 = 3

送回

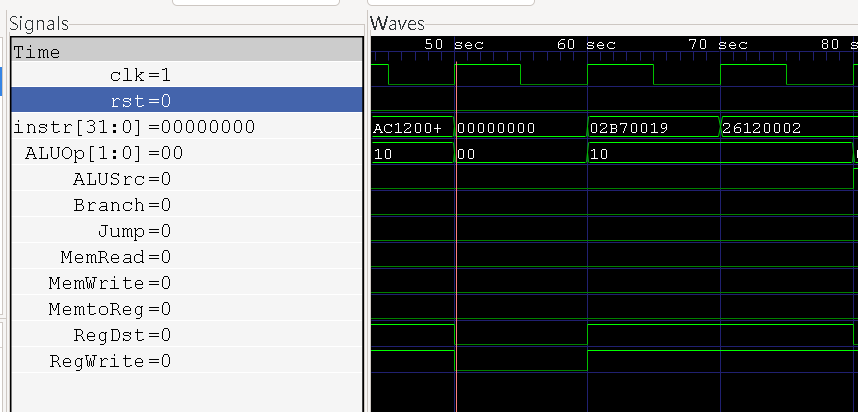
a = 0, b= 18, result = 18

MemWrite = 1

RD1是0x0，RD2是0x3，a是RD1的值，b是立即值擴充後的值，a+b後得

到要存入的暫存器，最後將RD1的值存入算好的暫存器($24也就是18的16

進制)

* NOP

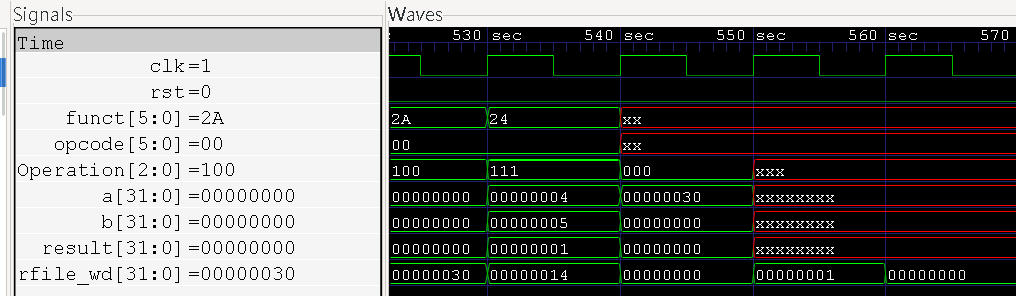
Instr = 0

control\_unit

全部歸零

NOP指令的instruction為32’d0，NOP代表什麼都不做，所以將control\_unit中所有控制訊號都設為0

* SLT $s2, $s3, $s4



讀funct

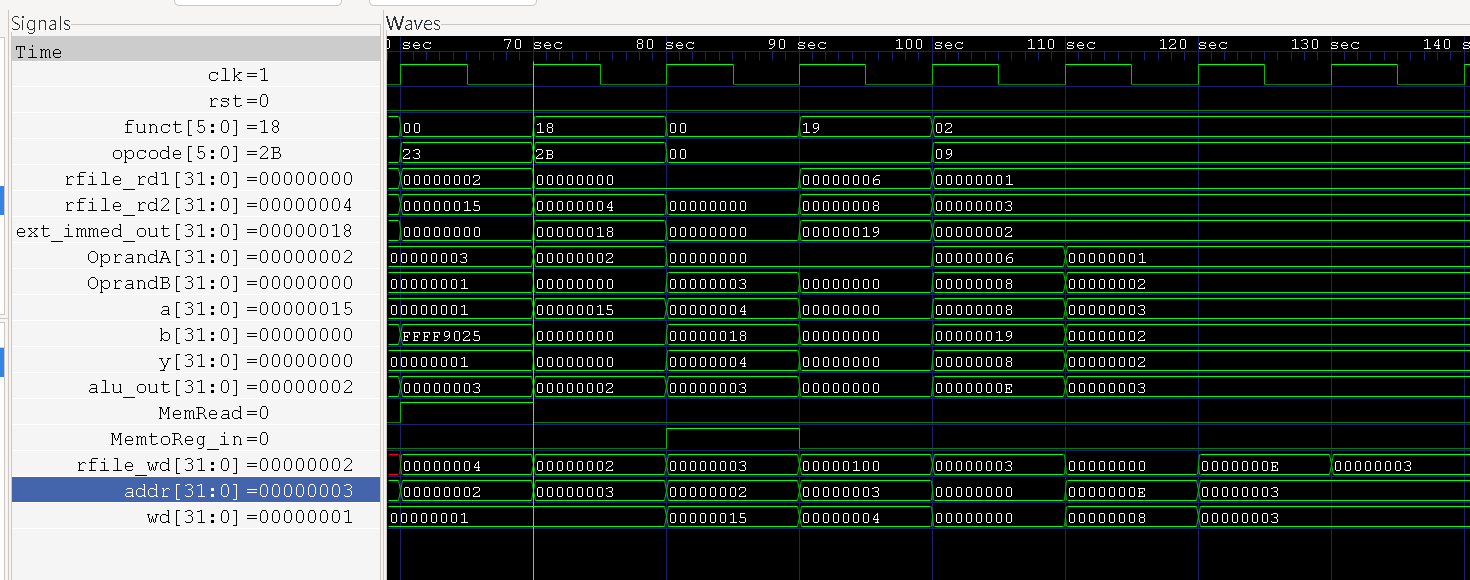
a = 4, b = 5

4 < 5

送回

Opcode是6’b0，funct是2A代表執行SLT，因為a<b所以result是1，最後將結果送回RegFile以存入暫存器。

* lw $s1, $t7, 0



讀opcode

MemRead = 1

送回

Rd1 = 2

Rd2 = 15

MemRead = 1

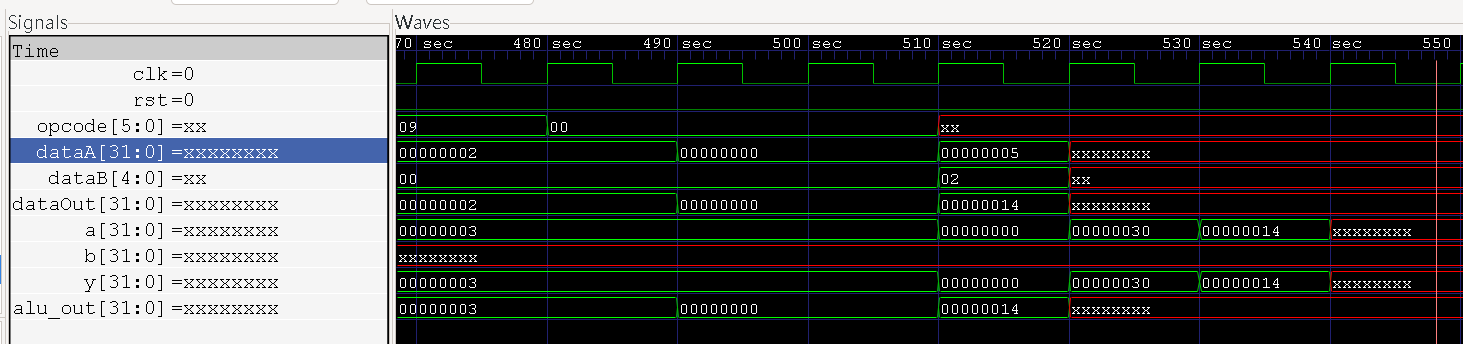
讀opcode

opcode是0x23，選offset(ext\_immed\_out)當operandB，將0x2+0x0 ，且

MemRead設為1，再用ALU運算完的結果定址data memory，並將指定位址的

內容寫回RegisterFile。

* SLL $s2, $s4, 2 ( 5 << 2 = 20 (Hex = 14 ))



讀opcode

dataA = 5

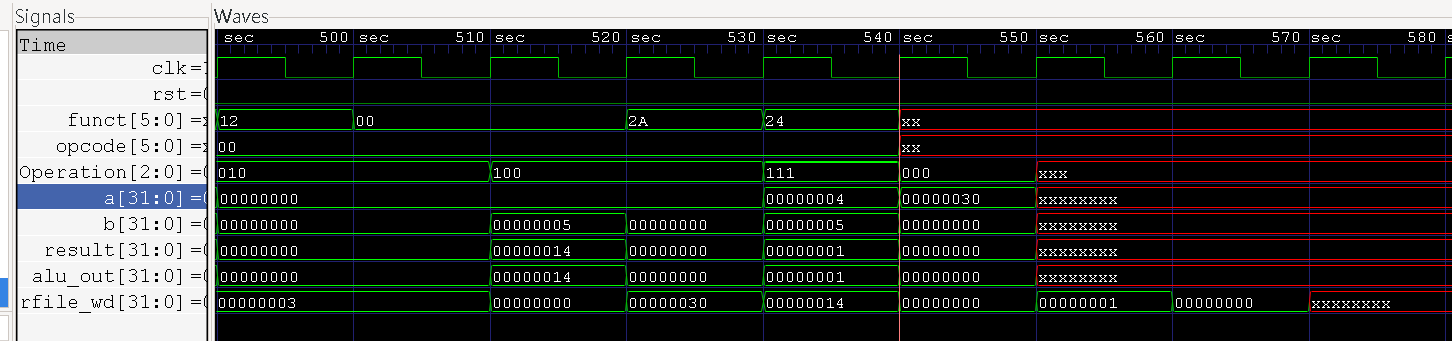
dataB = 2

送回

opcode是0x00，得dataA = 5，dataB = 2，5偏移2得到20，也就是十六

進位的14，經由TotalALU送回

* AND $s2, $s1, $s0 (110000 & 0 = 0)



讀funct

Operation = 000

a = 30, b = 0

30 & 0 = 0

送回

opcode是0x0，funct是0x24，在EX階Operation會送訊號000給ALU做

AND，0X30 AND 0X0 = 0X0 ，再隔兩個cycle，WB階送回RegFile存入暫存器。