108-2 計算機組織 Final Project: ALU Design

108 學年度第 2 學期

指導老師：朱守禮　老師

資工二甲 第 30 組

10727114 周胤誠

10727129 高偉承

10727132 張任宏

10727170 黃煖軒

一、背景

這次 Project 使用 VerilogHDL 撰寫，以 Midterm Project 所設計之 ALU Design 為基礎， 參考課本 Chapter 4 與課程講義之 Pipelined Datapath，設計一個 Pipelined MIPS-Lite CPU。透過本次的分組作業讓我們能夠熟悉 VerilogHDL 的語法並能夠靈活運用。

二、方法

(1)設計重點說明:

利用老師的 Single Cycle CPU 加入 4 個 Pipeline Register 來實現本次 Final Project 的 PipelineCPU 並且實現下列 16 道 MIPS 指令

a) Integer Arithmetic: **add, sub, and, or, srl, slt, ori**

b) Integer Memory Access: **lw, sw**

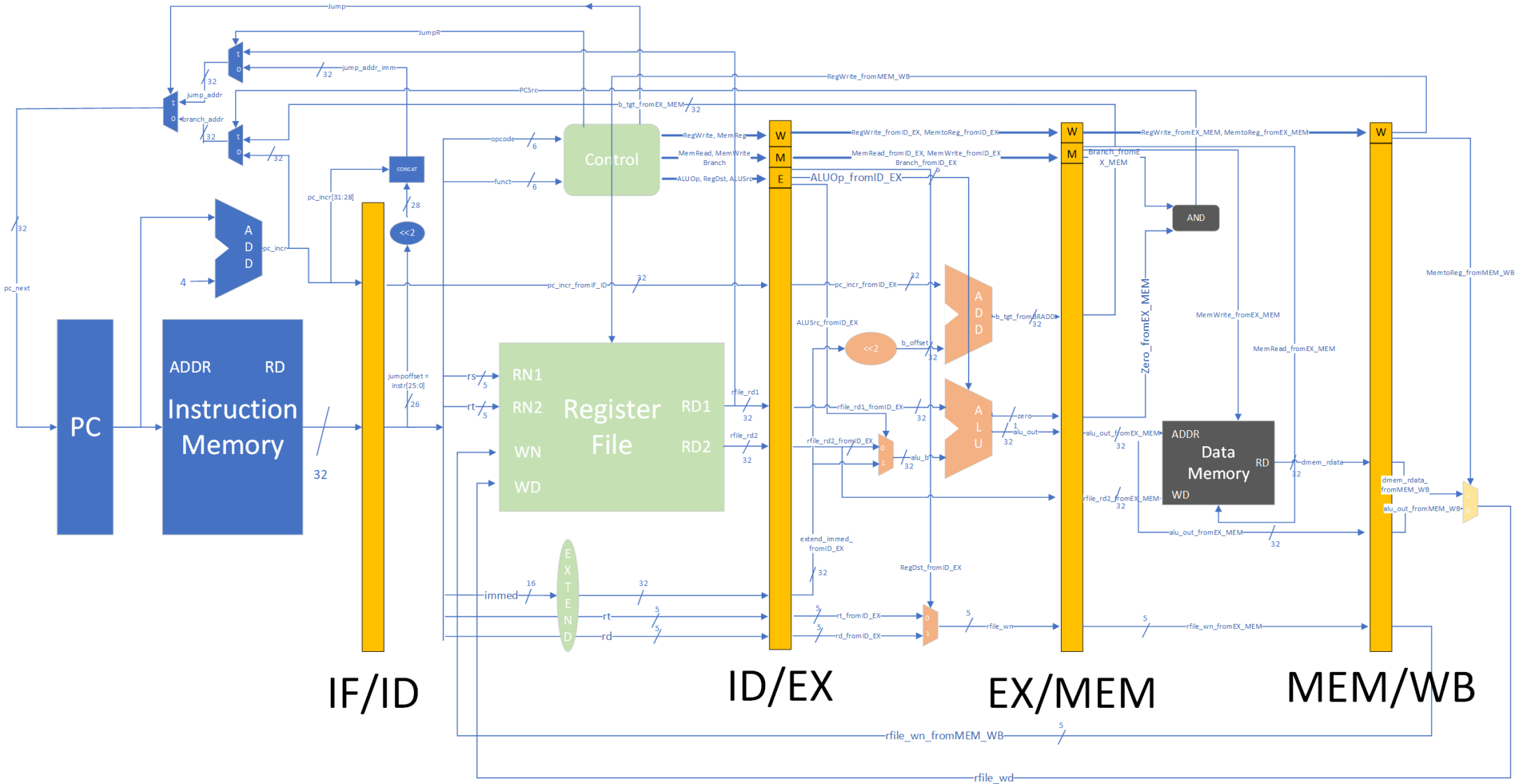
c) Integer Branch: **beq, j, jr**

d) Integer Multiply/Divide: **divu**

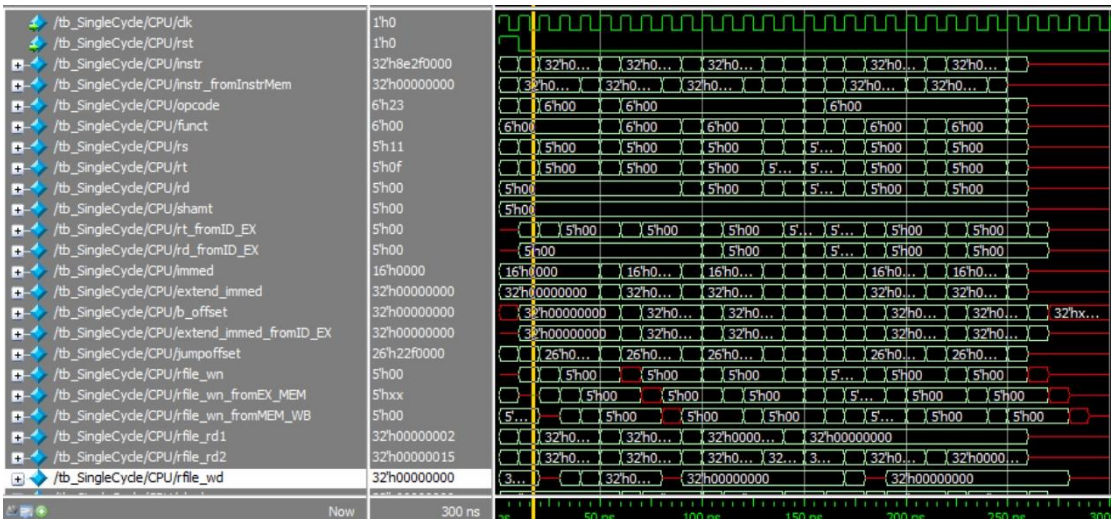
e) Other Instructions: **mfhi, mflo, nop**

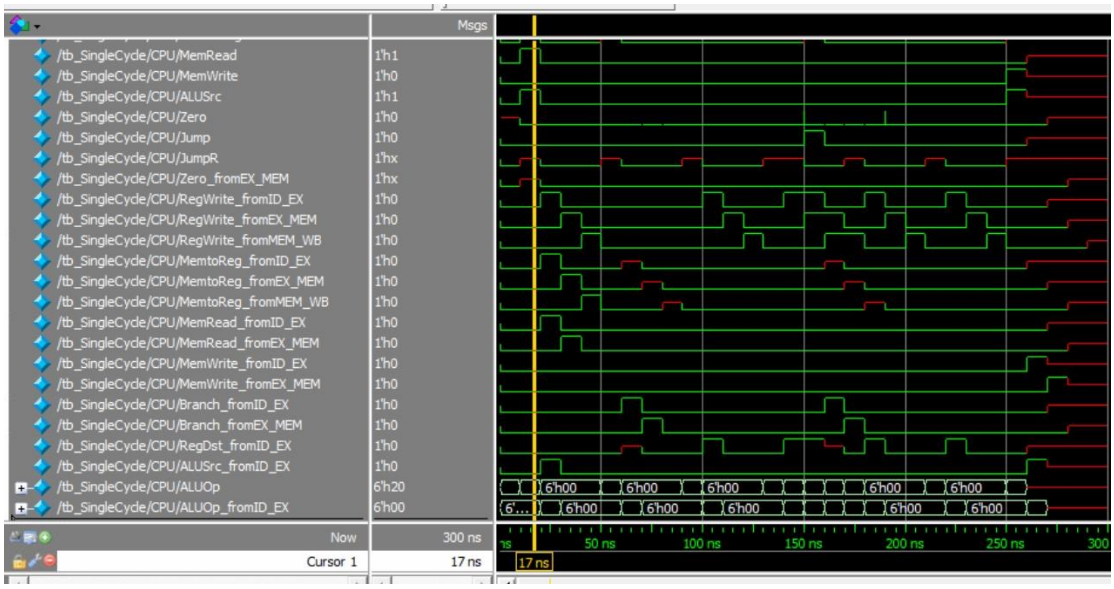
(2)Datapath 與詳細架構圖:

請見下一頁(較於詳細，易看)

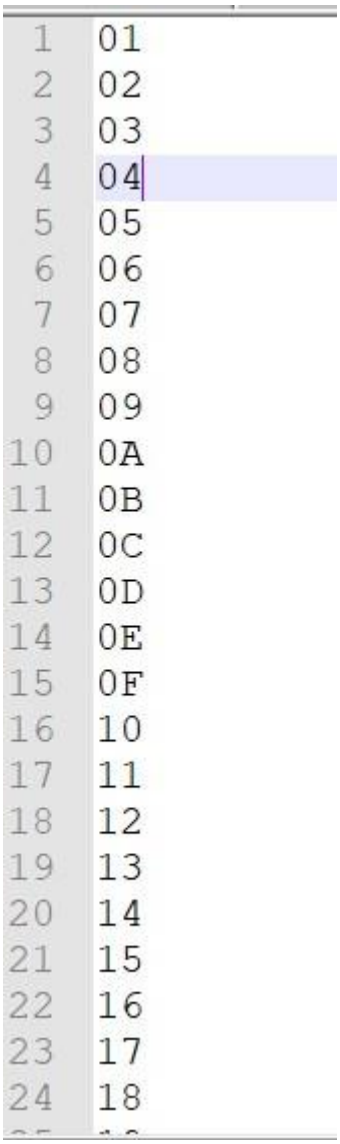

Jump
JumpR
jump_addr_imm
jump_addr
branch_addr
PCSrc
b_tgt_fromEX_MEM
RegWrite_fromMEM_WB
pc_next
PC
ADDR
RD
Instruction Memory
ADD
4
pc_incr
pc_incr[31:28]
CONCAT
<<2
jumpoffset = instr[25:0]
opcode
funct
Control
RegWrite, MemReg
MemRead, MemWrite
Branch
ALUOp, RegDst, ALUSrc
pc_incr_fromIF_ID
RN1
RN2
WN
WD
Register File
RD1
RD2
rs
rt
rfile_rd1
rfile_rd2
immed
EXTEND
rd
W
M
E
RegWrite_fromID_EX, MemtoReg_fromID_EX
MemRead_fromID_EX, MemWrite_fromID_EX
Branch_fromID_EX
ALUOp_fromID_EX
pc_incr_fromID_EX
ALUSrc_fromID_EX
b_tgt_fromBADD
b_offset
rfile_rd1_fromID_EX
rfile_rd2_fromID_EX
alu_b
ALU
zero
alu_out
extend_immed_fromID_EX
RegDst_fromID_EX
rt_fromID_EX
rd_fromID_EX
rfile_wn
RegWrite_fromEX_MEM, MemtoReg_fromEX_MEM
Branch_fromEX_MEM
AND
Zero_fromEX_MEM
MemWrite_fromEX_MEM
MemRead_fromEX_MEM
alu_out_fromEX_MEM
rfile_rd2_fromEX_MEM
Data Memory
dmem_rdata
rfile_wn_fromEX_MEM
alu_out_fromEX_MEM
MemtoReg_fromMEM_WB
dmem_rdata_fromMEM_WB
alu_out_fromMEM_WB
rfile_wn_fromMEM_WB
rfile_wd
IF/ID
ID/EX
EX/MEM
MEM/WB

三、結果





以上是我們用老師給的 instruction DATA 加上因為 hazard 所自行加上的 nop 指令的波型圖



這是我們的 data memory

下一頁開始是我們的程式執行結果。

```
#                    0, reading data: Mem[         x] =>           x
# 18446744073709551615, PC:         x
#                    0, reading data: Mem[         0] => 2385444864
#                    0, reg_file[ 0] =>          0 (Port 2)
#                    0, reg_file[ 0] =>          0 (Port 1)
#                    0, PC:          0
#                    0, wd:          0
#                    0, NOP
#
#                    1, reading data: Mem[         4] =>           0
#                    1, reg_file[15] =>         21 (Port 2)
#                    1, reg_file[17] =>          2 (Port 1)
#                    1, PC:          4
#                    1, LW
#
#                    2, reg_file[ 0] =>          0 (Port 2)
#                    2, reg_file[ 0] =>          0 (Port 1)
#                    2, PC:          8
#                    2, wd:          x
#                    2, NOP
#
#                    3, reading data: Mem[         0] =>   67305985
#                    3, PC:         12
#                    3, wd:          0
#                    3, NOP
#
#                    4, reading data: Mem[        16] =>  305201156
#                    5, reg_file[15] <=   67305985 (Write)
#                    4, PC:         16
#                    4, wd:   67305985
#                    4, NOP
#
#                    5, reading data: Mem[        20] =>           0
#                    5, reg_file[17] =>          2 (Port 1)
#                    5, reg_file[17] =>          2 (Port 2)
#                    5, PC:         20
#                    5, BEQ
#
#                    6, reg_file[ 0] =>          0 (Port 2)
#                    6, reg_file[ 0] =>          0 (Port 1)
#                    6, PC:         24
#                    6, wd:          0
#                    6, NOP
#
#                    7, PC:         28
#                    7, wd:          0
#                    7, NOP
#
#                    8, reading data: Mem[        32] =>   38834208
#                    8, PC:         32
#                    8, wd:          x
#                    8, NOP
```

```
#
#                    9, reading data: Mem[        36] =>          0
#                    9, reg_file[16] =>         1 (Port 2)
#                    9, reg_file[18] =>         2 (Port 1)
#                    9, PC:        36
#                    9, wd:         0
#                    9, ADD
#
#                   10, reg_file[ 0] =>         0 (Port 2)
#                   10, reg_file[ 0] =>         0 (Port 1)
run
#                   10, PC:        40
#                   10, wd:         0
#                   10, NOP
#
#                   11, PC:        44
#                   11, wd:         0
#                   11, NOP
#
#                   12, reading data: Mem[        48] =>   38834210
#                   13, reg_file[18] <=         0 (Write)
#                   12, PC:        48
#                   12, wd:         0
#                   12, NOP
#
#                   13, reading data: Mem[        52] =>   36735008
#                   13, reg_file[18] =>         0 (Port 1)
#                   13, reg_file[16] =>         1 (Port 2)
#                   13, PC:        52
#                   13, wd:         0
#                   13, SUB
#
#                   14, reading data: Mem[        56] =>  134217748
#                   14, reg_file[17] =>         2 (Port 1)
#                   14, PC:        56
#                   14, wd:         0
#                   14, ADD
#
#                   15, reading data: Mem[        60] =>          0
#                   15, reg_file[ 0] =>         0 (Port 2)
#                   15, reg_file[ 0] =>         0 (Port 1)
#                   15, PC:        60
#                   15, J
#
#                   16, reading data: Mem[        80] =>   38834210
#                   17, reg_file[18] <=         0 (Write)
#                   16, PC:        80
#                   16, wd:         0
#                   16, NOP
```

```
#                    17, reading data: Mem[        84] =>          0
#                    17, reg_file[16] =>          1 (Port 2)
#                    17, reg_file[18] =>          0 (Port 1)
#                    18, reg_file[17] <=          1 (Write)
#                    17, PC:         84
#                    17, wd:           1
#                    17, SUB
#
#                    18, reg_file[ 0] =>          0 (Port 1)
#                    18, reg_file[ 0] =>          0 (Port 2)
#                    18, PC:         88
#                    18, wd:           x
#                    18, NOP
#
#                    19, PC:         92
#                    19, wd:           0
#                    19, NOP
#
#                    20, reading data: Mem[        96] =>   38834213
VSIM 29> run
#                    21, reg_file[18] <=          0 (Write)
#                    20, PC:         96
#                    20, wd:           0
#                    20, NOP
#
#                    21, reading data: Mem[       100] =>          0
#                    21, reg_file[16] =>          1 (Port 2)
#                    21, reg_file[18] =>          0 (Port 1)
#                    21, PC:        100
#                    21, wd:           0
```

```
#                    22, reg_file[ 0] =>          0 (Port 2)
#                    22, reg_file[ 0] =>          0 (Port 1)
#                    22, PC:        104
#                    22, wd:           0
#                    22, NOP
#
#                    23, PC:        108
#                    23, wd:           0
#                    23, NOP
#
#                    24, reading data: Mem[       112] => 2886860824
#                    25, reg_file[18] <=          0 (Write)
#                    24, PC:        112
#                    24, wd:           0
#                    24, NOP
#
#                    25, reading data: Mem[       116] =>          x
#                    25, reg_file[18] =>          0 (Port 2)
#                    25, PC:        116
#                    25, SW
#
#                    26, reg_file[ x] =>          x (Port 2)
#                    26, reg_file[ x] =>          x (Port 1)
# control_single unimplemented opcode  x
#                    26, PC:        120
#                    28, writing data: Mem[         0] <=          0
#                    27, PC:          x
#                    28, PC:          x
#                    29, PC:          x
```

四、討論

1.寫 Pipeline 時，要確認每一道時序想像中與實際的差距，需要確認哪一到指令先做與否。

五、結論

這次的 Final Project 我們都覺得非常困難，一開始瞭解 pipeline 的原理和架構就覺得有點沒頭緒，概念性的東西似懂非懂，好像理解了，但實做的過程中又想不到要怎麼寫，實做過程基本上部分東西是拿期中 Project 的東西來改，幫我們的 CPU 加了一些功能，然後這次的東西其實老師已經幫我們寫好大部分的元件了，所以要作 Pipeline 必須要涉及到很多需要接線的狀況，不小心就會因為手殘造成的問題也是頻頻發生，debug 方面的感覺也跟期中不一樣，期中基本上就看出來的值是不是對的就可以找到問題，這次要對著 wave 一個個對比，確認訊號是否正確，再加上可能上課沒那麼認真聽講和信息傳達等一系列原因，以為 Hazard 是要解決的，實際上不用因為老師其實也沒講到。。。通過寫不用寫的 hazard，我們從錯誤中學到可以手動添加 NOP 來處理 hazard，總體來說還是收穫蠻大的，這次的 Final Project 的 PipelineCPU 是我們本學期計算機組織課程的集大成，也是最精采最精華的部分。

六、未來展望

我覺得經過這次的期末 project，我們對 Verilog 的使用終於走出了新手村，更加得心應手了，但因為這次可能進度有點趕的原因，沒有要求我們實做處理 hazard 的部分，雖然理論上知道該怎麼做，也確實陰差陽錯的實際寫了這塊的東西，但很遺憾寫出來的東西並不太行，我覺得只後有時間可以自己試著把這塊內容好好寫一遍，應該會有更多的收穫。

各隊員分工方式與負責項目：

計組紅 周胤誠：報告撰寫、Debug、切 Pipeline、接線幫手

計組黃 高偉承：報告撰寫、便當支援、切 Pipeline、接線師

計組綠 張任宏：報告撰寫、失去的 Hazard Unit 設計者、接線幫手

計組紫 黃煖軒：報告撰寫、防疫指導