計算機組織

112學年度第2學期

Final Project

Pipelined ALU Design

老師： 朱守禮 老師

學生： 11127132羅海綺

11127137黃乙家

11127138林雨臻

一、背景

為了使學生充分了解，該如何使用Verilog HDL與Modelsim模擬器，以Midterm Project 所設計之 ALU Design及計算機組織課程講義：

[1] Chapter 4:The Processor

[2] Pipelined Datapath

為基礎，設計一個 Pipelined MIPS-Lite CPU。

二、方法

1.功能說明：本Project包含下列 16 道 MIPS 指令。

(a) Integer Arithmetic: add, sub, and, or, sll, slt, andi

(b) Integer Memory Access: lw, sw

(c) Integer Branch: beq, j, jr

(d) Integer Multiply/Divide: multu

(e) Other Instructions: mfhi, mflo, no

2.設計要求：Datapath 與詳細架構圖(並以 PowerPoint 或 Word 設計繪製)。

(1) ALU:

(a) 設計重點說明：

* 需使用Midterm Project所設計之ALU完成add, sub, and, or, sll, slt, andi指令。

(b) Datapath與詳細架構圖：請參考附錄(CO\_ALU.pptx)。

(2) Datapath:

(a) 設計重點說明：

* 所有指令之執行，須遵守 5-Stage Pipelined CPU 執行指令之行為。

(b) Datapath與詳細架構圖：請參考附錄(CO.pptx)。

(3) multu:

(a) 設計重點說明：

* 32-bits 無號數乘法指令。
* 需使用 Midterm Project 所設計之 Multiplier。

(b) Datapath與詳細架構圖：請參考附錄(CO.pptx)。

(5) Testbench:

(a) 設計重點說明：

* 須以讀檔的方式，讀入測試資料。
* Single Cycle 版本之 MIPS 處理器設計與Testbench Verilog 範例，請詳見 i-Learning 上之「Single Cycle CPU and Testbench」
* 本機測時將有不同的測試資料，以確認設計正確性與完整性。

(b) Datapath與詳細架構圖：請參考附錄(CO.pptx)。

程式架構：

mips\_pipeline:

IF - Instruction Fetch:

PC

PCADD (PC+4)

memory (InstrMem)

2-1 MUX\*3

ID - Instruction Decode:

ControlUnit

sign\_extend or unsign\_extend

RegFile

BEQ (從MEM移到ID)

2-1 MUX\*1

EX - Execute / Address Calculation:

ALU

ALUControl

Multiplier

HiLo

2-1 MUX\*2

3-1 MUX\*1

MEM - Memory Access (read write):

memory (DatMem)

WB - Write Back (result into register file):

2-1 MUX\*1

三、結果

(1) mips\_pipeline:

(a) 宣告線路:

 (i) instruction bus

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述(ii) break out important fields from instruction

一張含有 文字, 螢幕擷取畫面, 字型, 行 的圖片

自動產生的描述(iii) datapath signals

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述(iv) control signals

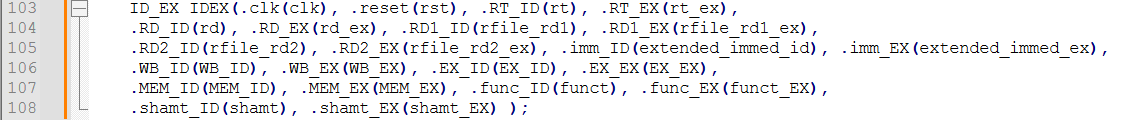
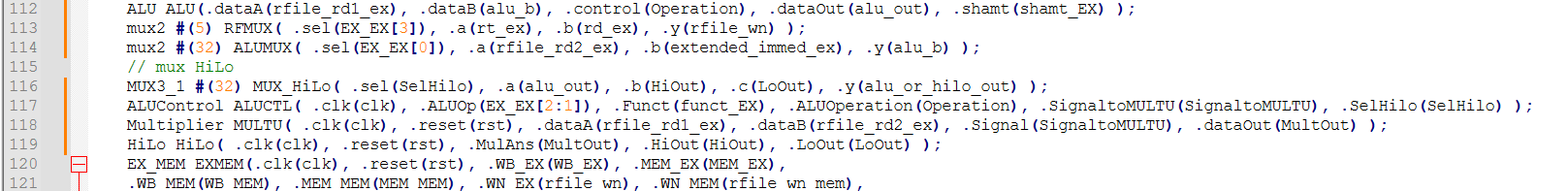
一張含有 文字, 字型, 螢幕擷取畫面 的圖片

自動產生的描述(v) 分割線路

(b) Pipelining stages:

(i) IF - Instruction Fetch一張含有 文字, 字型, 螢幕擷取畫面 的圖片

自動產生的描述

(ii) ID - Instruction Decode一張含有 文字, 字型, 數字, 螢幕擷取畫面 的圖片

自動產生的描述

(iii) EX - Execute / Address Calculation

(iv) MEM - Memory Access (read write)

(v) WB - Write Back (result into register file)

(2) IF - Instruction Fetch:

(a) PC:

當時脈clk posedge時，若非reset就輸出算好的下道指令位置。

(b) memory:

MemRead或者mem\_array有變化時，判斷MemRead是否為1，若為1至指定記憶體位置讀取資料。

當時脈posedge且MemWrite為1時，將資料寫入指定記憶體位置。

(3) IF\_ID:

若非reset，傳遞instruction以及nextPC。

(4) ID - Instruction Decode:

(a) ControlUnit:

判斷OpCode，設定control signals。

(b) reg\_file:

當RN1或file\_array有變化時，判斷RN1是否為0，若為0直接賦值RD1=zero，否則RD1 = file\_array[RN1]。

RN2同理。

當時脈posedge時判斷RegWrite訊號是否為1且WN是否不等於0，若皆成立，寫入資料WN暫存器中。

(c) unsign\_extend:

無號數擴充直接補16’b0。

(d) sign\_extend:

有號數擴充將第15個bit補16次。

(e) BEQ:

我們將BEQ指令從MEM移到ID階，提早判斷並決定是否跳躍，減少nop數量從3個至1個。

實現方法：先判斷opcode是否為BEQ，再比較兩個基底暫存器的內容是否相等，若成立則將zero設為0。

(5) ID\_EX:

若非reset，傳遞訊號至下階段。

(6) EX - Execute / Address Calculation:

(a) ALUControl:

用funct決定ALU訊號。

(b) ALU:

ALU 32位元做AND、OR、ADD、SUB、SLT、SLL、將乘法器64位元的運算結果分割成Hi、Lo、移位器左移運算。

(c) Multiplier:

執行期中project的乘法運算。

(d) HiLo:

若非reset，將HiLo設為乘法運算的結果。

(7) EX\_MEM:

若非reset，傳遞訊號至下階段。

(8) WB - Write Back (result into register file):

用一個2-1MUX判斷是否要寫回reg\_file。

四、討論

(1) Modelsim 驗證結果：

讀入 input.txt 後，根據每個 opcode 算出答案，

第一筆為 AND 運算，

第二筆為 OR 運算，

第三筆為 ADD 運算，

第四筆為 SUB 運算，

第五筆為 SLT 運算，，計算的

取 sum 的最高位 ，因此

第六筆為 SLL 運算，

第七筆為 MUL 運算，使用第二代乘法器，

第一輪：確認乘數第 0 位的值，目前為 1，因此將乘數放進乘積的左半邊，並將乘數與 product 右移一位，

第二輪與第一輪相同，第三輪發現乘數第 0 位是 0，因此乘數與乘積直接右移即可，以此類推直到第 32 輪，最後 ，Hi 暫存器為 ，Lo 暫存器為

(2) Waveform 輸出圖形：

1. 結論

完整波型圖：

各組員分工方式與負責項目：

1. 分工方式：

組員討論寫法後，一人寫一支程式，其他人透過加註解及撰寫書面報告

的方式理解程式，為求所有人都理解程式內容。

1. 負責項目：
2. 程式碼：黃乙家、林雨臻、羅海綺
3. 書面報告：黃乙家、林雨臻、羅海綺

心得感想：

我們認為本次的計組期中專案太晚公告題目內容，因為儘管期中考週結束了，卻仍然有許多科目的project也才剛要開始，同時間我們組員中還有人必須準備多益檢定考試、程式考試，以及資結作業等等。

因為還不是非常熟悉Verilog的語法，因此在將程式轉換成電路的過程中，我們理解了一段時間。最後我們是由一個對於 Verilog 最熟悉的組員寫最複雜的module，其他人在那個module完成時，負責加上註解與理解 module 內容，最後再整理到書面報告上，也因為這種分工模式，負責製作報告與加註解的組員們可以在一有疑惑時就立馬發問，直到理解。

此次專案中特別的一點是，除了完成程式需要達成的事情之外還加入了一些限制規範，希望我們能去思考如何善用語法，以另一種 model 撰寫而不影響到電路本身的運作，這使我們可以更加理解那些語法的實際作用。雖然專案題目說的這些需求有可能增加程式的複雜度，但這同時可以幫助我們以後在編寫硬體描述語言，想設計自己的電路的時候熟悉基本的概念。

對於 Debugging，我們認為應該要先讓大家學會如何以波形圖來抓出錯誤，可以的話再介紹 modelSim 更多使用方式，因為在業界上有許多 IC 設計公司還是以 ModelSim 作為 Verilog 模擬平台，這樣才能夠幫助大家儘早熟悉編寫 Verilog。