計算機組織 Final Project:

Pipelined CPU Design

111學年度第2學期

老師:朱守禮 老師

班級:資訊二甲

學生:

11027148 周品誼

11027152 黃婉禎

11027156 林芷安

11027164 趙怡儒

**一、背景**

使用 Verilog HDL 與 Modelsim Verilog 模擬器，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。

參考i-Learning上之Appendix A: Assemblers, Linkers, and the SPIM

Simulator，完成下列 16 道 MIPS 指令:

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

b) Integer Memory Access: lw, sw

c) Integer Branch: beq, j

d) Integer Multiply/Divide: multu, maddu

e) Other Instructions: mfhi, mflo, nop

設計要求:

(1) ALU: 需使用 Midterm Project 所設計之 ALU 完成 add, sub, and, or, srl, slt, addiu 指

令。其設計方法請詳見 Midterm Project。不接受其他設計方式。

(2) Datapath: 所有指令之執行，須遵守 5-Stage Pipelined CPU 執行指令之行為。

(3) multu: 32-bits 無號數乘法指令，需使用 Midterm Project 所設計之 Multiplier。其設計

方法請詳見 Midterm Project。不接受其他設計方式。

(4) maddu: 32-bits 無號數乘加指令，需使用 Midterm Project 所設計之 Multiplier。其設計方法請詳見 Midterm Project。不接受其他設計方式。

(5) Testbench: 為所設計之模組之測試平台，須以讀檔的方式，讀入測試資料。以驗證所設計之模組，功能正確性。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」。機測時將有不同的測試資料，以確認設計正確性與完整性。

(6) 本 Project 內所有 Verilog 程式，除了 Testbench 外，其餘部分皆【不接受迴圈形式的設計】；意即程式內不能有 for/while 等敘述。不能包含 Function/Task 敘述。不能包含always @(\*)敘述。

**二、方法**

設計重點說明:

(1) ALU: 將dataA, dataB, ALU Control singnal, dataOut, reset 傳入 ALU,如果reset為1就做歸零(temp1 = 32'b0,dataOut = temp1),接著用Signal設定sel以便MUX選擇做R-type、maddu、addiu中的哪一道指令，AND、OR、ADD、SUB、SLT、SRL、MULTU、MFHi、MFLo 。將期中project的乘法器及位移器HiLo暫存器等等集合成一大包並加以修改 ，在此ALU運用。當Signal == SUB 跟 SLT時設定c[0] = 1'b1(c[0]同時是cin0),同時也設定binvert為c[0](這樣與b做XOR才能達到減法的效果),特別注意的是set只在第32位元時需要設定(MSB:最高有效位元),且其回傳值就是第一位元的less的代入值,其他位元的less皆以1'b0代入1-bit ALU。而1-bit ALU中則進行and,or,add,sub,slt的計算，同時進行binvert與dataB的XOR，其中add.sub則用FA進行計算，計算結果由signal判斷後用MUX選擇輸出。

(2) Pipeline: 用IF/ID, ID/EX, EX/MEM, MEM/WB將整個datapath切成共五個階段。

write-back stage control line:

WB\_1、WB\_2、WB\_3、WB\_4分別代表WB在由pipeline分割的ID、EX、MEM、

WB層。位元[0]判斷Mem to Reg Mux、[1]判斷Reg Write

Memory access stage control line:

M\_1、M\_2、M\_3分別代表M在由pipeline分割的ID、EX、MEM層。位元[0]

判斷Mem Write、[1]判斷Mem Read、[2]判斷Branch、[3]判斷Jump Mux

Execution Calculation stage control line:

EX\_1、EX\_2分別代表EX在由pipeline分割的ID、EX層。位元[0]判斷ALU

Src Mux、[1]判斷ALU Op0、[2]判斷ALU Op1、[3]判斷Reg Dst Mux

IF/ID - ID/EX將從Instruction Memory中解碼後的指令傳遞至Control Single來判斷指令為何，也傳入Register file(rs, rt傳入RN1, RN2)﹑將立即值透過Extend來進行32位元擴充﹑將rd前26位元左移二並和pc+4結合成28位元。

ID/EX - EX/MEM，一多功器透過ALUSrc判斷要使用立即值還是register file中的rd2(r-type)，將結果和register file 中的 rd1一齊傳入ALU mux，並透過ALU control判斷指令。將pc+4結果和已擴充為32位元的立即值傳入加法器﹑還有一多功器透過RegDst判斷要使用rt或rd。

EX/MEM - MEM/WB將ALU mux執行結果傳入Data memory的Addr，並將

register file的rd2傳入Data memory的wd。透過MemRead MemWrite決定是否要讀取或將資料寫入記憶體。將ALU的zero和memory的branch判斷訊號傳入 一AND gate，並以此結果為PCSrc-另一多工器的判斷訊號，判斷是否要進行beq指令或pc+4，將此結果和在IF/ID - ID/EX階段將rd前26位元左移二並和 pc+4結合成28位元的值傳入另一多工器，並用jump訊號判斷要執行哪一到指令(可能為pc+4, beq和jump)。

WB-一多工器透過MemtoReg判斷要將rd或ALU mux結果傳回register

file的WD。將在EX/MEM - MEM/WB中的透過RegDst判斷要使用rt或rd的多功器結果傳回WN。把RegWrite訊號傳回register file。將在EX/MEM - MEM/WB階段判斷的指令(pc+4, beq或jump)傳回PC，接著就可以執行pc+4，並從PC重新讀取指令，再進入Instruction Memory解碼。

(3) maddu: 將乘法答案以及HiLo暫存器內容相加，前期操作與multu相同，在HiLo暫存器判斷指令為何，若為maddu則輸出答案為乘法器答案加HiLo暫存器原本的值。

(4) nop:如果在pc讀取指令後在instruction memory解碼發現該指令為32’b0且time非為0時，則須執行nop。會跑一次pipeline，但是什麼都不做。將占用一些時間，來讓前面還沒執行完的程式跑到下一階段，以便之後不會發生hazard。

(5) Mux多工器:Mem to Reg Mux判斷輸出為ALU out或Data Memery的RD、jump Mux判斷輸出是否為指令j、ALU Src Mux判斷進ALU的為RD2還是擴充的immediate、Reg Dst Mux判斷WN為rt或rd、PC Mux判斷輸出為PC+4還是branch指令結果

(6) ALU Control: 先前在control single先判斷了各個指令的ALUOP，現在則透過其判斷要執行何項指令。ALUOP = 2’b00則執行強制加法；ALUOP = 2’b01則執行強制減法；ALUOP = 2’b10則再透過Funct去判斷為ADD, SUB, AND, OR, SLT, SRL的哪項指令。

(7)hazard:先判斷指令為何，再去決定有關hazard的相關係數，其中hazard用來判斷是否發生hazard﹑check為延遲次數﹑count用來計數跑了多少次pipeline﹑en\_pc則用來判斷是否要輸入下一道指令。當輸入指令時，如果指令為乘法或乘加，則check設為30﹑en\_pc設為0；如果指令(32位元instr)為32’b0且count = 0，則check和count設為1﹑en\_pc設為0；如果指令為beq，則check設為2﹑en\_pc設為0；如果指令為jump，則check設為2﹑en\_pc設為0；而如果check = 0，將en\_pc設為1。

每當clock敲響一次，count++。如果rst為true，則將count, check﹑hazard設為0，且en\_pc為1。若rst為false，如果count = 1'b0且en\_pc = 1’b1，則hazard為0，否則hazard = 1。而當count = check時(表示可以執行下一到指令了)，將en\_pc設為1，count和check設為0。

(8) tb\_SingleCycle:參考老師給的範例程式碼，並設定cycle及加上缺少的指令。

**三、結果**

Modelsim 驗證結果與 Waveform 輸出圖形

一張含有 文字, 螢幕擷取畫面, 軟體 的圖片

自動產生的描述一張含有 文字, 螢幕擷取畫面, 軟體 的圖片

自動產生的描述

第0個不會有東西，在第一個cycle執行lw

lw:

// lw $s1, $t7, 0

reg\_file[17]的17是$s1, [15]是$t7,PC變為4

一張含有 文字, 軟體, 多媒體軟體, 電腦圖示 的圖片

自動產生的描述

beq:

// beq $s1, $s2, 3

beq會先判斷兩個暫存器是否equal，若equal則branch 3

會執行3個nop(跳過3條指令)再做跳過後的下一道指令

一張含有 文字, 螢幕擷取畫面, 軟體, 電腦圖示 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 軟體, 電腦圖示 的圖片

自動產生的描述

J:

// j 7

跳到第7道指令，所以經過nop會執行add

一張含有 文字, 螢幕擷取畫面, 軟體, 電腦圖示 的圖片

自動產生的描述

add:

 // add $s1, $s0, $s1, 0

reg\_file裡的值做add的動作後經過兩個cycle後結果(3)寫進reg\_file

一張含有 文字, 字型, 螢幕擷取畫面, 收據 的圖片

自動產生的描述

一張含有 文字, 字型, 螢幕擷取畫面, 設計 的圖片

自動產生的描述

Sub:

// sub $s2, $s0, $s2

Sub如果少一個暫存器的值會沿用上一個也就是1

所以sub出來的結果就是3-1 = 2 一樣經過2個cycle 才會以write訊號顯示

一張含有 文字, 字型, 螢幕擷取畫面, 行 的圖片

自動產生的描述

一張含有 文字, 字型, 收據, 白色 的圖片

自動產生的描述

maddu: 將乘法結果加上hilo

3\*2+3\*2= 12

一張含有 文字, 螢幕擷取畫面, 陳列, 軟體 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 軟體, 多媒體軟體 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 行, 字型 的圖片

自動產生的描述

一張含有 文字, 字型, 白色, 設計 的圖片

自動產生的描述

**四、datapath 與詳細架構圖**

一張含有 文字, 圖表, 方案, 圖解 的圖片

自動產生的描述

**五、結論**

**分工:**

ALU、接線:趙怡儒

datapath圖﹑報告:林芷安

pipeline、hazard、接線:周品誼

報告、接線:黃婉禎

**感想:**

11027148 周品誼:

經過這次的作業，我對Verilog這個語言熟悉許多，也學習了很多知識。雖然中間經過一些小插曲和挫折，像是找接錯線的東西或是訊號的接收輸出有沒有放錯東西，檢查這些需要有很大的耐心，但經過我們多次的作業討論，這份作業終於完成了。謝謝每一位組員的盡力付出我們才能夠順利完成這項作業。

11027152 黃婉禎 :

這次的作業真的非常讓人頭疼，光是理解要具體要做甚麼就花費了我好幾天的時間，從零開始想要做甚麼元件、每個元件要負責甚麼再轉乘verilog，這對我來說非常困難，而且還有更難的接線問題，非常考驗腦力還有眼力。這次真的還好有隊友們的幫助，不但解決和解答了我許多的問題，也讓我學到了許多，如果沒有他們的話我肯定是無法做出來的，真的非常感謝我的組員們。

11027156 林芷安:

這次的作業讓我覺得我好無能，我在想pipeline的時候想破頭都寫不太出來。還好最後其他組員都很凱瑞，我真的十分感謝他們。但最後做報告時寫各個module的作用時再看看我們的程式碼到底寫了什麼讓我也有學到這次得期末project的內容，希望我有機會再遇到pipeline時不要像現在一樣無能為力。再次感謝偉大的組員們!

11027164 趙怡儒:

這次的作業接觸了完全新的知識pipeline，在上課時聽到要把datapath切開讓指令可以分部執行，聽起來非常的有道理，想不到開始實作後光是接每個pipeline register的線就讓大家非常頭痛，接完線答案還不一定正確，Debug更是花了許多心力，幸好在每個組員的分工合作下完成了這次作業，真的是可喜可賀，謝謝大家的付出!

**六、未來展望**

經過這次Final Project，無疑大家都對接線和電路設計以及Pipeline以有更多的了解。希望經過這次的作業能讓我們在之後再次遇到電路設計時寫的更加順利，能夠活用現在所學的知識，更加熟練地寫出verilog及設計出電路架構。另外也期許未來能夠在相關領域，例如數位電路設計、嵌入式系統、硬體加速器、人工智慧硬體等等，透過更進一步的學習，培養出對硬體設計的興趣和能力，並將這些技能應用到實際的專案中。