VLSI System Design (Graduate Level)

Fall 2023

HOMEWORK I

REPORT

Must do self-checking before submission:

Compress all files described in the problem into one tar

All SystemVerilog files can be compiled under SoC Lab environment

All port declarations comply with I/O port specifications

Organize files according to File Hierarchy Requirement

No any waveform files in deliverables

Student name: \_蔡明翰\_

Student ID: \_N26120579\_

目錄

**一、Design Explanation**

**二、Simulation Results**

三、**Synthesize**

四、**Superlint**

**五、Others Work**

**六、心得和問題解決**

**Summary**

|  |  |
| --- | --- |
| Clk Period | 10ns |
| Area | 5869482.776079 um2 |
| Data Require Time | 10.77ns |
| Power | 123.9136mW |
| Superlint Coverage | 100% |
| Others Work | Design Comparison |
| Design Explanation | ˇ |
| RTL simulation | pass |
| SYN simulation | pass |
| Waveform | ˇ |
| Synthesize Results | ˇ |
| 問題說明和解決 | ˇ |
| Cells num | 35 |
| Net num | 612 |

**一、Design Explanation**

設計一符合RISC-V ISA之5-stage pipelined CPU，並支援Forwarding、stall、flush功能，以處理Data Hazard和Jump/Branch的發生。採用的是RV32I指令集並加上乘法和部分CSR指令，共支援45條instructions，內部有32個32bits Register供運算和暫存使用，以及2個64bits CSR Registers分別計算cycle數和指令數。透過兩個SRAM解決記憶體在同時讀取或寫入資料的危障，一個記憶體用來存放資指令，另一個用來存放資料。

一張含有 文字, 圖表, 方案, 平行 的圖片

自動產生的描述**整體架構:** : clk

1. IF stage

IF級功能為Instruction Fetch，向Instruction Memory送入Adder值並設定WEN為1111、DI為0(因為只需要讀取)，以取得instruction，並透過PC紀錄目前執行位址。最後將執行位址和指令輸出至暫存器，以送至下一級。

1. PC: 當clk正緣觸發時，更新輸出值，並分為以下三種情況。

|  |  |
| --- | --- |
| 一般情形 | 輸出值更新為上一個執行位址+4 |
| stall發生 | 需要暫停1個cycle，輸出值保持為上一個執行位址 |
| Jump/Branch發生 | 輸出值更新為Jump或Branch指令指定的跳轉位址 |

1. IM: 因為IM的讀取會延遲1個cycle，所以我選擇將mux決定後的指令位址不經過PC直接送入IM的Adder，以節省1個cycle，缺點就是此訊號可能會有不穩定的問題。另外是word adder的模式，故只需讀取pc[15:2]。

一張含有 文字, 圖表, 方案, 行 的圖片

自動產生的描述

1. ID stage

ID級的功能為Instruction Decode，負責進行指令的解碼，並送出控制訊號和暫存器資料至暫存器，供EXE級運算和判斷。

* 1. Decoder: 將指令的op,f3,f7,rd,rd2,rs2,immediate,csr\_op解碼出來。

一張含有 文字, 螢幕擷取畫面, 數字, 行 的圖片

自動產生的描述

* 1. Controller: 透過解碼的出來的op,f3,f7,csr\_op，產生控制訊號，同時減少後面EXE級的負擔。

|  |  |
| --- | --- |
| 控制訊號 | 功能 |
| JAL/JALR | 表示Jump指令，需要跳轉，決定jump位址運算的source。 |
| B\_type | 表示B-type指令，判斷是否要跳轉。 |
| Mem\_read | 表示讀取Data memory。 |
| Mem\_write | 表示寫入Data memory，判斷Data Hazard是否發生。 |
| Reg\_write | 決定資料是否寫入Register File。 |
| alu\_op | 決定ALU的運算方式 |
| alu\_rs1\_sel, alu\_rs2\_sel | 決定傳入ALU的source。 |
| DataWidth | Load/Store指令的資料存取寬度。 |

* 1. RegFile: 共32個32bits Registers，供EXE級運算和暫存資料使用。當clk正緣觸發時，一般情況會將舊有資料繼續暫存。當write back訊號拉起時，會將write back data寫入指定暫存器位址(0號暫存器永遠為0)。
  2. Forwarding Unit: 因為RegFile更新需要1個cycle，EXE級可能就會拿到尚未write back的資料，所以當WB\_rd == D\_rs1或D\_rs2且write back訊號拉起時，需要直接將資料write back data直接送到D\_rs1\_data或D\_rs2\_data，避免EXE級運算錯誤。
  3. Hazard Detection: 當load指令發生時，若下一個指令的來源暫存器和load指令的目標暫存器位址相同。Load指令至少要到MEM級結束，才能寫回資料，但下一指令也已在EXE級運算結束，所以需要拉起stall訊號，使前一指令暫停在IF REG，並清空ID REG。

一張含有 文字, 圖表, 方案, 平行 的圖片

自動產生的描述

1. EXE stage

EXE級負責的主要功能為ALU的運算以及跳轉指令的target address計算，並決定是否跳轉，並將運算完成的資料送至下一級。

* 1. ALU: 根據alu\_op的值進行各種運算，輸出大小為32bits的運算結果，和1bit的Branch用來判斷B\_type指令是否跳轉，這次新增乘法、read cycle和read instret指令。cycle暫存器統計經過cycle數，rst後，每當clk正緣觸發，輸出值會+1；instret 暫存器統計執行指令次數，rst後，每當clk正緣觸發，除了NOP發生時，輸出值保持不變，輸出值同樣+1。
  2. Forwarding Unit: 與前面ID stage所提data hazard發生原因相同，都是資料無法及時寫回RegFile，EXE forwarding可分為兩種情況。

[1] M\_rd == E\_rs1 or M\_rd == E\_rs2且M\_Reg\_write==1時，將M\_alu\_out的資料直接送入E\_rs1\_data或E\_rs2\_data。

[2] WB\_rd == E\_rs1 or WB\_rd == E\_rs2 且WB\_en==1時，將write back data資料直接送入E\_rs1\_data或E\_rs2\_data。

其中狀況[1]優先於[2]，因為狀況[1]是較新的指令，同樣的目標暫存器，會蓋過前面指令的值。

* 1. JB Unit: 根據E\_Jal、E\_Jalr、B\_type和alu\_branch的值決定是否跳轉，並根據指令計算跳轉的位址。同時發出跳轉訊號E\_jb清空IF REG和ID REG。

[1]Jal,B\_type: E\_jb\_pc=pc+imm; [2]Jalr: E\_jb\_pc=imm+E\_rs1\_data

一張含有 文字, 圖表, 方案, 工程製圖 的圖片

自動產生的描述

1. MEM stage

MEM級負責的主要功能為對欲存入記憶體的資料，進行調整，並決定存入的模式，即WEN的數值。並讀取data memory的資料輸出至暫存器，以送至下一級。

* 1. DM Controller: 在Mem\_write拉起時 (store指令)，透過data width和alu out最右2bits的數值決定data memory存入資料的模式；或只讀取資料。
  2. DM\_in Generator: 根據data width和alu out最右2bits對欲存入資料 (next\_rs2\_data)進行調整，之後送入DM 的data input。

因為DM的讀寫同樣會延遲1cycle，所以我決定將送入data memory的判斷訊號和資料，不經過EXE REG，直接送至DM Controller、DM\_in Generator和DM adder。

一張含有 文字, 圖表, 螢幕擷取畫面, 平行 的圖片

自動產生的描述

1. WB stage

WB級的主要功能是將資料寫回IF stage的RegFile，對data memory輸出的資料進行調整，並判斷寫回資料為DM讀取資料或ALU運算結果，同時回傳write back訊號WB\_en和WB\_rd。

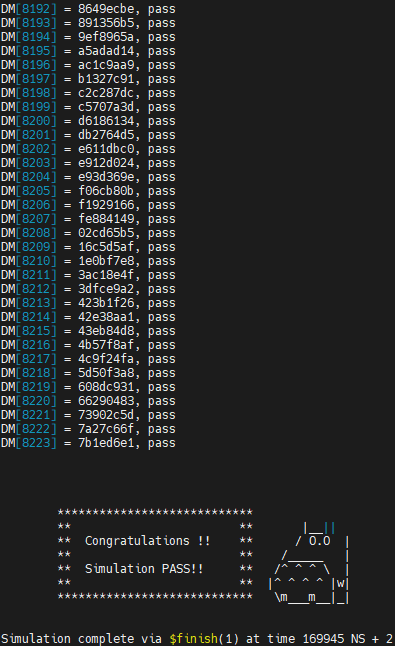
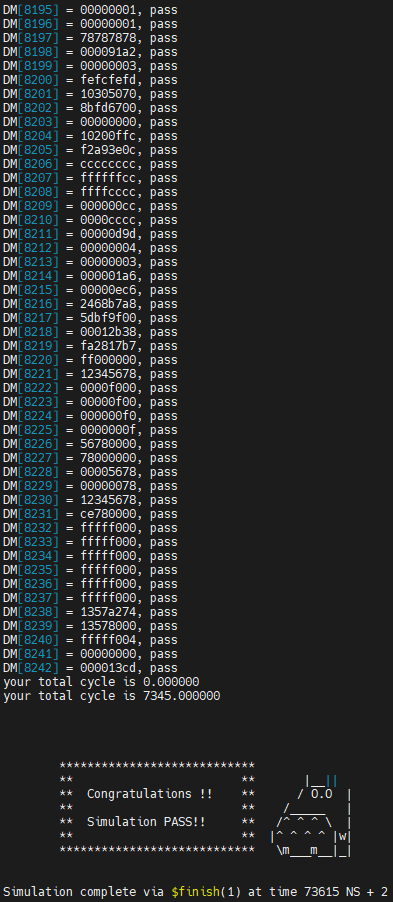
一張含有 文字, 圖表, 螢幕擷取畫面, 行 的圖片

自動產生的描述

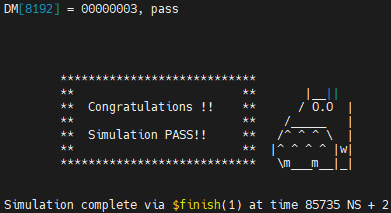
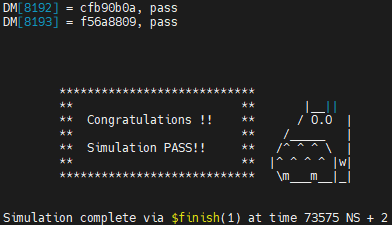
**二、Simulation Results**

**1.Pre-sim** (clk period: 10ns)

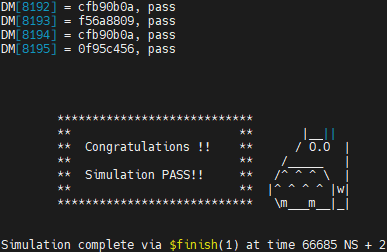
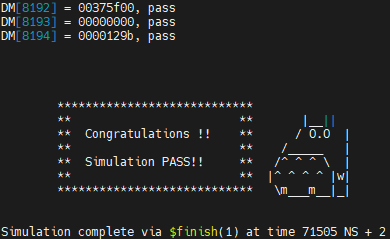
prog0 prog1



prog2 prod3

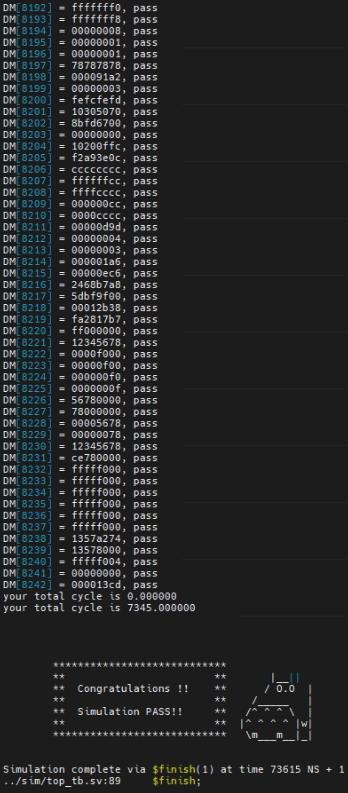


Prog4 prog5



**2.Post-sim** (clk period: 10ns)

Prog0: 7345ns

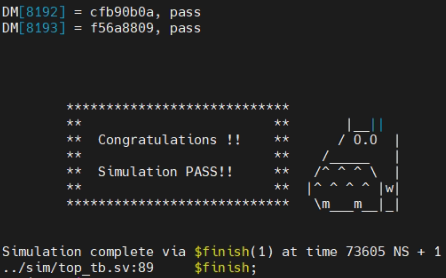


Prog1: 169945ns

一張含有 文字, 螢幕擷取畫面, 設計 的圖片

自動產生的描述

Prog2: 73605ns



Prog3: 85735ns

一張含有 文字, 螢幕擷取畫面, 字型, 圖表 的圖片

自動產生的描述

Prog4: 71505ns

一張含有 文字, 螢幕擷取畫面, 字型, 設計 的圖片

自動產生的描述

Prog5: 66695ns

一張含有 文字, 螢幕擷取畫面, 設計 的圖片

自動產生的描述

**3.Waveform**

以下無特別標示皆以16進位表示，為求訊號整齊，故使用Pre-sim波形說明。

(1)R\_type

一張含有 螢幕擷取畫面, 文字, 行, 數字 的圖片

自動產生的描述

Alu\_op=0，表示add指令，rd=rs1+rs2。

8000+0=8000、8000+4=8004、0+0=0、0+ffff\_ffff=ffff\_ffff，波形與預期相符，功能正常。

一張含有 螢幕擷取畫面, 行 的圖片

自動產生的描述

Alu\_op=1，表示sub指令，rd=rs1-rs2。

0-ffff\_ffff=1、1- ffff\_ffff=2、2-ffff\_ffff =3、3-ffff\_ffff =4、4-ffff\_ffff=5、-3-5=ffff\_fff8、-3-ffff\_fff8=-5，波形與預期相符，功能正常。



Alu\_op=2，表示sll指令。

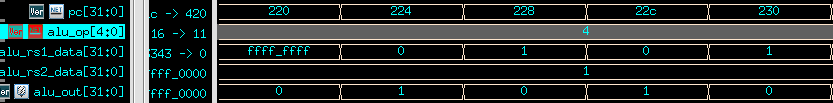
1<<1=2、2<<1=4、4<<1=8、8<<1=10、10<<1=20。在沒有overflow發生的情況下，每左移1bit當於\*2，波形與預期相符，功能正常。

一張含有 螢幕擷取畫面, 行 的圖片

自動產生的描述

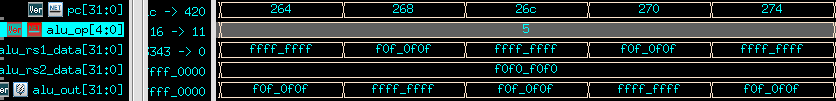
Alu\_op=3，表示slt指令。

True: ffff\_ffff<1、0<1；False: 1<1，波形與預期相符，功能正常。



Alu\_op=4，表示sltu指令。

True: 0<1；False: |ffff\_ffff|<1、1<1，波形與預期相符，功能正常。



Alu\_op=5，表示xor指令。

ffff\_ffff^f0f0\_f0f0=f0f\_0f0f、f0f\_0f0f^f0f0\_f0f0=ffff\_fff，波形與預期相符，功能正常。

一張含有 螢幕擷取畫面, 行 的圖片

自動產生的描述

Alu\_op=6，表示srl指令。

ffff\_ffff>>4=fff\_ffff、fff\_ffff>>4=ff\_ffff、f\_ffff>>4=ffff、ffff>>4= fff，邏輯右移會自動補0，波形與預期相符，功能正常。

一張含有 螢幕擷取畫面, 文字, 字型, 行 的圖片

自動產生的描述

Alu\_op=7，表示sra指令。

8765\_4321>>>4=f876\_432、f876\_5432>>>4=ff87\_6543、ff87\_6543 >>>4=fff8\_7654、fff8\_7654>>>4=ffff\_8765、7777\_7777>>>16=1dd、7777\_7777 >>>1d=3、7777\_7777>>>3=eee\_eeee、7777\_7777>>>e=1\_dddd、7777\_7777，算術右移會根據最左邊的bit補0c或補1，波形與預期相符，功能正常。

一張含有 螢幕擷取畫面, 文字, 字型, 行 的圖片

自動產生的描述

Alu\_op=13，表示mul指令，為signed\*signed。

1234\*f=1\_110c，後32bita為1\_110c；1\_110c\*f=f\_ffb4，後32bits為f\_ffb4。波形與預期相符，功能正常。

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

Alu\_op=15，表示mulhsu指令，為signed\*unsigned。

f0f0\_f0f0\*f0f0\_f0f0結果之前32bits為f1d3\_b596；f1d3\_b596\*f0f0\_f0f0之結果前32bits為f2a9\_2360。波形與預期相符，功能正常。

一張含有 螢幕擷取畫面, 文字, 字型, 行 的圖片

自動產生的描述

Alu\_op=16，表示mulhu指令，為unsigned\*unsigned。

1234\_5678\*f0f0\_f0f0結果之前32bits為1122\_3343；1122\_3343\*f0f0\_f0f0結果1020\_303e。波形與預期相符，功能正常。

(2)I\_type (alu\_rs2\_sel=1，alu\_rs2之input為imm)

一張含有 螢幕擷取畫面, 行, 多媒體軟體 的圖片

自動產生的描述

Alu\_op=3，表示slti指令。

True: ffff\_fc66<ffff\_fc94、ffff\_fc66<22b、ffff\_fc66; False: fff\_fc66<ffff\_f832、ffff\_fc66<ffff\_fc66，波形與預期相符，功能正常。

一張含有 螢幕擷取畫面, 行 的圖片

自動產生的描述

Alu\_op=8，表示ori指令。

1|1bc=1bd、1bd|6e5=7fd、7fd|ffff\_fedd=ffff\_fffd、ffff\_fffd|9c= ffff\_fffd、ffff\_fffd|502=ffff\_ffff，波形與預期相符，功能正常。

一張含有 螢幕擷取畫面, 行 的圖片

自動產生的描述

Alu\_op=9，表示andi指令。

abcd\_ef98&ffff\_feb3=abcd\_ee90、abcd\_ee90&ffff\_ff44=abcd\_ee00、abcd\_ee00 &209=200、200&ffff\_fe87=200、200&355=200，波形與預期相符，功能正常。

(3)CSR\_type

一張含有 螢幕擷取畫面, 文字, 行 的圖片

自動產生的描述

Alu\_op=17，表示rdinstreth指令。此時instret=13cc，前32bits值為0。

Alu\_op=18，表示rdinstret指令。此時instret=13cd，後32bits值為13cd。

波形與預期相符，且觀察到每當執行1個指令後，instret值會隨之+1，功能正常；當NOP出現時，instret值則維持不變。

一張含有 螢幕擷取畫面, 時鐘 的圖片

自動產生的描述

Alu\_op=19，表示rdcycleh指令。此時cycle=1cb0，前32bits值為0。

Alu\_op=1a，表示rdcycle，此時cycle=1cb1，後32bits值為0。

波形與預期相符，且觀察到每當經過1個clk後，cycle值會隨之+1，功能正常。

(4)B\_type

一張含有 螢幕擷取畫面, 行 的圖片

自動產生的描述

Alu\_op=a，表示beq指令。

False: ffff\_fff0=ff0，branch=0，波形與預期相符，功能正常。

一張含有 螢幕擷取畫面, 行, 多媒體軟體 的圖片

自動產生的描述

Alu\_op=b，表示bne指令。

False: ffff\_fff0!=ffff\_fff0，branch=0，波形與預期相符，功能正常。

(5)S\_type

一張含有 螢幕擷取畫面, 行 的圖片

自動產生的描述

Mem\_write=1，且data\_width=2，表示SW指令。

alu\_out=rs1+imm，DMin=next\_rs2\_data，波形與預期相符，功能正常。

一張含有 螢幕擷取畫面, 行 的圖片

自動產生的描述

Mem\_write=1，data\_width=0，表示SB指令，有號數補位，且根據alu\_out[1:0]調整位置。

DMin=next\_rs2\_data[7:0]，1234\_5678[7:0]=78，波形與預期相符，功能正常。

Mem\_write=1，data\_width=1，表示SH指令有號數補位，且根據alu\_out[1:0]調整位置。。

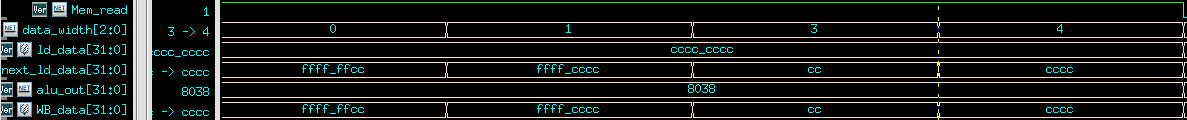
DMin=next\_rs2\_data[15:0]，1234\_5678[15:0]=5678，波形與預期相符，功能正常。

(6)L\_type

一張含有 螢幕擷取畫面, 行 的圖片

自動產生的描述

Mem\_read=1，data\_width=2，表示LW指令。next\_ld\_data=ld\_data，波形與預期相符，功能正常。



Mem\_read=1，data\_width=0，表示LB指令，並且有號數增空。next\_ld\_data=

ld\_data[7:0]，cccc\_cccc[7:0]=ffff\_ffcc，波形與預期相符，功能正常。

Mem\_read=1，data\_width=1，表示LH指令，並且有號數補位。next\_ld\_data= ld\_data[15:0]，cccc\_cccc[15:0]=ffff\_cccc，波形與預期相符，功能正常。

Mem\_read=1，data\_width=3，表示LBU指令，並且無號數補位。next\_ld\_data = ld\_data[7:0]，cccc\_cccc[7:0]=cc，波形與預期相符，功能正常。

Mem\_read=1，data\_width=4，表示LHU指令，並且無號數補位。next\_ld\_data = ld\_data[15:0]，cccc\_cccc[15:0]=cccc，波形與預期相符，功能正常。

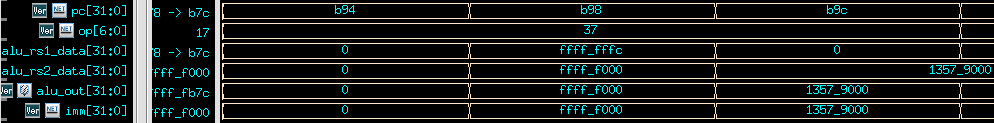
(7)U\_type

一張含有 螢幕擷取畫面, 行 的圖片

自動產生的描述

op=17，表示auipc指令。

alu\_out=pc+imm，b78=b78+0、ffff\_fb7c=b7c、b80+1357\_9000=1357\_9b80，波形與預期相符，功能正常。



op=37，表示lui指令。

alu\_out=imm，0=0、ffff\_0000=ffff\_0000、1357\_9000=1357\_9000，波形與預期相符，功能正常。

(8)Jal

一張含有 文字, 螢幕擷取畫面, 軟體, 多媒體 的圖片

自動產生的描述

Alu\_op=10，jb\_op(JB unit source)=pc，表示jal指令。

rd=pc+4=bb4+4=bb8，E\_jb\_pc=pc+imm=bb4+8=bbc，波形與預期相符，功能正常。

(9)Jalr

一張含有 文字, 螢幕擷取畫面, 軟體, 多媒體 的圖片

自動產生的描述

Alu\_op=10，jb\_op=next\_rs1\_data，表示jalr指令。

rd=pc+4=7e4+4=7e8，E\_jb\_pc=next\_rs1\_data+imm=7f8+4=7fc，波形與預期相符，功能正常。

**三、Synthesize**

**1.Synthesize Results**

|  |  |  |  |
| --- | --- | --- | --- |
| **Clock Period** | **Area(um2)** | **Power(mW)** | **Data Require Time** |
| 10ns | 5869482.776079 | 123.9136 | 10.77ns |

2.Timming

一張含有 螢幕擷取畫面, 文字 的圖片

自動產生的描述

3.Area

一張含有 文字, 螢幕擷取畫面 的圖片

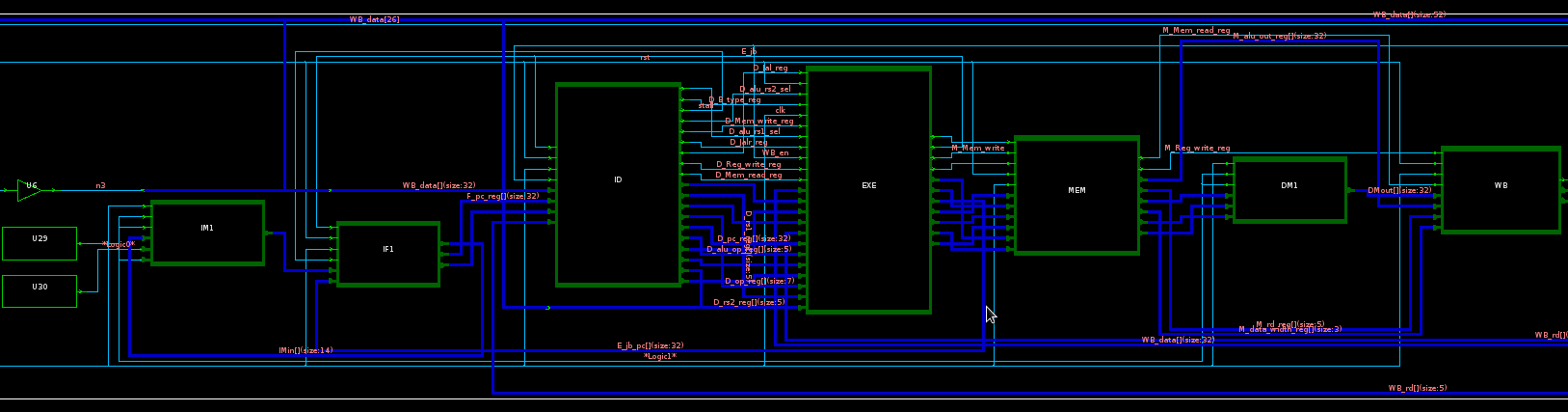
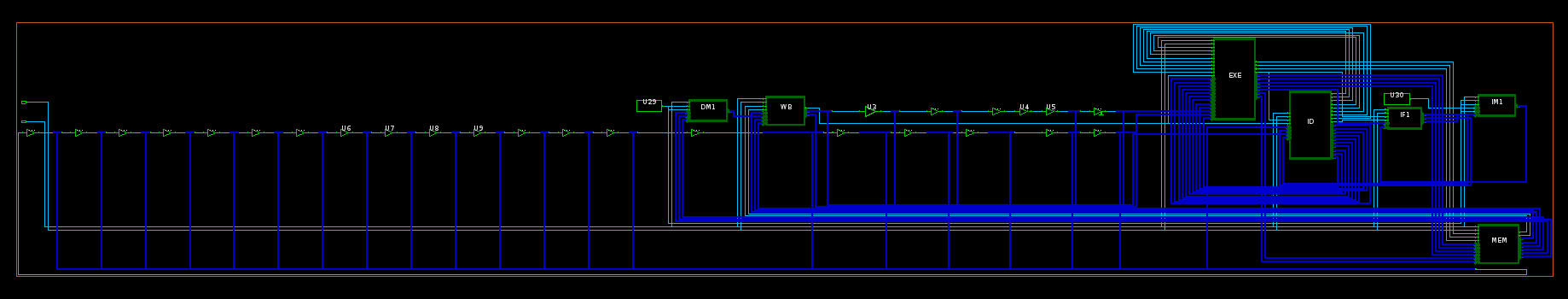
自動產生的描述

4.Power

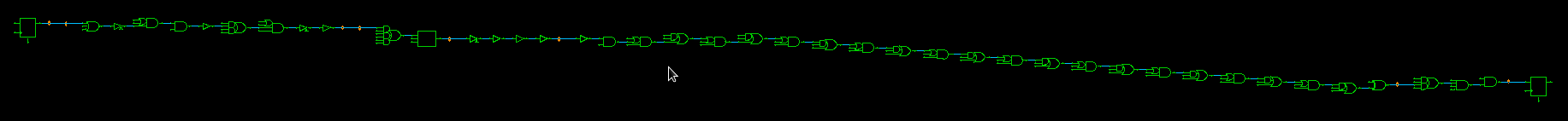
一張含有 文字, 螢幕擷取畫面, 軟體, 數字 的圖片

自動產生的描述

5.Overview



1. Critical path

一張含有 圖表, 文字, 方案, 行 的圖片

自動產生的描述

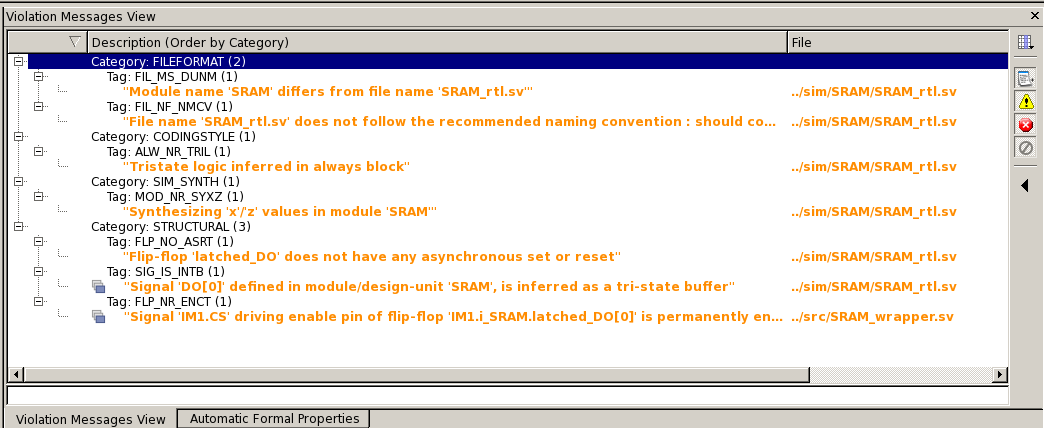
**四、Superlint**

1.Total line: 1371 lines

**一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述**

2.Superlint results

****

剩下的皆是跟SRAM檔案有關的warning。

3.Coverage

Code coverage = 100%。

**4.Modification**

在執行完Superlint後，針對結果，對我的Code作了以下調整。

[1]在register加入posedge rst訊號，完成非同步rst設計。

[2]將沒有用到的訊號接到0或default等不會改變電路功能的線，避免output no load和影響其他訊號。像乘法的指令有不會用到半部分的結果，直接將它跟0做and後接到其他線。

[3]調整命名方式，將大小寫不同但字母卻相同的name改變名稱，像不同stage有些用RS1和rs1，統一改成stage\_rs1；另外避免使用System Verilog的保留字，將IF改名為IF1。

**五、Others Work**

因為SRAM的讀寫需要1個cycle，為了補回這1個cycle，需要去除IM/DM前面或後面的Registers，但去掉這些Registers，可能就會增加Critical Path的長度，尤其是在DM(因為EXE級的ALU和forwardimg路徑極長)，為此我根據Registers的位置，設計了不同版本，比進行比較。(原本:reg-mem-reg)

|  |  |  |  |
| --- | --- | --- | --- |
| version | -IM-reg,-DM-reg | -IM-reg,reg-DM- | reg-IM-,reg-DM- |
| Data require time | 10.77ns | 11.87ns | 12.98ns |

**六、心得和問題解決**

過去雖然修過計算機組織，但從未想過要自己實現RISC-V 5stage CPU，在一開始的IF stage首先遇到的困難就是memory需要花費一個cycle的問題，原本的方法是將IM的output不經過reg直接拉線至ID stage，但後來不同versin的比較，決定改使用mem input不經過reg的做法。

在ID stage原本只有簡單解碼出op、f3、f7訊號，控制訊號大多是在EXE stage才進行計算，增加EXE級的負擔。為了減小critical path，決定將所有控制訊號先在ID stage運算完成。接著是forwarding的問題，後來發現只簡單將其他級的訊號拉近來判斷，但卻忽略了0號暫存器須一直保持為0，在debug時花了不少時間，解決的方法是在控制訊號，增加rd=0則不寫回RegFile的條件。

在EXE stage中，需要考慮ALU的控制訊號，因為指令很多，為了壓縮訊號，在設計時花了不少心思。forwading的優先順序，也要在設計時確認好。乘法需考慮unsigned\* unsigned、signed\*unsigned、unsigned\*unsigned三種，其中signed\*unsigned的狀況，system verilog無法直接合成。原本的方法是先判斷正負，後採二補數的方式，但會增加crtical path的長度。最後調整成在unsigned數的左方增加一位補0，就能直接進行可合成的有號數乘法。另外為了節省critical path的長度，進行了一些優化，包括控制訊號和架構。

有了前面的經驗，在後面2個stages遇到的問題就顯得較少，只有再考慮word adder的模式下，對寫入和讀取的資料和控制訊號WEN進行調整，花了一些時間思考，整體上順利很多。最後就是前面提過的superlint，改善自己的coding style，並繼續對面積和速度進行優化。

自己是第一次寫到這麼大的電路，雖然不需要複雜的演算法，但訊號非常多接線也較複雜。首要是能理解清楚RISC-V CPU的ISA，所以在寫之前，花了幾天在複習計組。並且能掌握自己的電路架構，常一邊coding一邊畫架構圖，debug和優化時才有明確方向。幸好過去有RTL coding的相關經驗，以及花費很長時間的努力，最後成功有將操作頻率剛好壓在10ns。這次獨力完成RISC-V CPU的經驗真的非常寶貴，不論是思考如何去完成每個元件的功能、不斷看waveform去debug的過程以及針對電路效能和codind style的優化，都是自己之後在數位電路領域前進的基石，希望在之後的作業能學習更多並順利完成。