VLSI System Design (Graduate Level)

Fall 2020

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: \_\_\_\_\_\_\_\_\_\_\_

Student ID: \_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| INTRODUCTION FOR DESIGN OF THE CPU | |
| SPEC | |
| PIPELINE STAGE | 5 |
| INSTRUCTION | 33 |
| FULL GRAPH | |
|  | |

|  |  |  |
| --- | --- | --- |
| STAGE1 INSTRUCTION FETCH | | |
|  | | |
| MODULE | FUNCTION | SPECIAL\_DESIGN |
| DIDER4 | Get the Instruction Address by dividing  the pc\_counter result by4 | Using shift right 2 bits instead using the divider which can reduce hardware area. |
| PAUSE INSTRUCTION | If there is any load use data hazard happen,this module will enable to pause the instruction.  The instruction amd pc value will be rewrited tio the register in next clock. | NONE |
| rst\_contoller | If there is any branch happen this module will enable flush the register. The module will be more detailed discuss later. | NONE |

|  |  |  |
| --- | --- | --- |
| STAGE2 INSTRUCTION DECODE & STAGE5 WRITE BACK | | |
|  | | |
| MODULE | FUNCTION | SPECIAL\_DESIGN |
| DECODER | Decoding 33 bits instruction into  different elements which is used for the following module | NONE |
| SIGNED UNSIGNED EXTENDED\_  SELECT\_IMM | Make the Immediate data referring to the type of the instruction | NONE |
| REG | 32×32-bit. registers | Posedge clk write back  Negedge clk read data |
| CONTROL | Make the Control signal | NONE |
| rst\_contoller | If there is any branch or load use data hazard happen  this module will enable flush the register.  The module will be more detailed discuss later. | NONE |

|  |  |  |
| --- | --- | --- |
| STAGE3 ALU EXECUTION | | |
|  | | |
| MODULE | FUNCTION | SPECIAL\_DESIGN |
| ALU\_IN | Data hazard signal will be the input module will decide the input of the alu. | NONE |
| ALU\_RD | calculate for the data store in $rd | NONE |
| ALU\_ADDR | calculate for the Data memory’s Address which will be stored data next clock. | NONE |
| PC\_ADDR | calculate the jumping address and make the signal to enable the system jump or not. | NONE |
| rst\_contoller | If there is any branch happen this module will enable flush the register. The module will be more detailed discuss later. | NONE |

|  |  |  |
| --- | --- | --- |
| STAGE4 WRITE OR READ MEMORY | | |
|  | | |
| MODULE | FUNCTION | SPECIAL\_DESIGN |
| DIVIDER4 | Divide the address value by 4  Quotient will be the real address value of data memory  Reminder will be used for LB SB instruction to load and store data to the corresponding byte. | Using shift right 2 bits instead using the divider which can reduce hardware area. |
| LOW\_BYTE\_CONTROL | Control for LB SB instruction to load and store data to the corresponding byte. | NONE |
| WB\_CONTROL | Wb\_control signal will decide data from memrory and alu writing back to rd rigister | NONE |

|  |  |
| --- | --- |
| OTHER CONTROL MODULE | |
| MODULE | LOAD\_DATA\_HAZARD |
|  | FUNCTION | SPECIAL\_DESIGN |
| CONTROL FLOW | |
| **Detect if there is any load use data hazard**  pc\_stall\_stage1=  (id\_exe\_read\_mem==1'b1 && (if\_id\_rs1\_addr==id\_exe\_rd\_addr || if\_id\_rs2\_addr==id\_exe\_rd\_addr))?1'b1:1'b0;  **Detect if there is any jump signal happened at the same time and prevent pc\_stall signal flush the jump action.**  **Jump signal has the priority**  pc\_stall=pc\_jump\_confirm?1'b0:pc\_stall\_stage1;  instruction\_stall=pc\_stall; | |
| MODULE | FORWARDING UNIT |
|  |
| CONTROL FLOW | |
| **Detect if there is any data hazard between memory and register $rs1**  rs1\_mem\_hazard=(mem\_wb\_write\_reg==1'b1 && mem\_wb\_rd\_addr!=5'd0 && mem\_wb\_rd\_addr==rs1\_addr)?1'b1:1'b0;  **Detect if there is any data hazard between ALU and register $rs1**  rs1\_exe\_hazard=(exe\_mem\_write\_reg==1'b1 && exe\_mem\_rd\_addr!=5'd0 && exe\_mem\_rd\_addr==rs1\_addr)?1'b1:1'b0;  **Detect if there is any data hazard between memory and register $rs2**  rs2\_mem\_hazard=(mem\_wb\_write\_reg==1'b1 && mem\_wb\_rd\_addr!=5'd0 && mem\_wb\_rd\_addr==rs2\_addr)?1'b1:1'b0;  **Detect if there is any data hazard between ALU and register $rs2**  rs2\_exe\_hazard=(exe\_mem\_write\_reg==1'b1 && exe\_mem\_rd\_addr!=5'd0 && exe\_mem\_rd\_addr==rs2\_addr)?1'b1:1'b0; | |

|  |  |
| --- | --- |
| MODULE | rst\_controller  left side (for exe\_mem stage and if \_id stage)  right side (id\_exe stage) |
|  |
| CONTROL FLOW | |
| **(for exe\_mem stage and if \_id stage)Global rst is cpu’s rst ; jump and branch instruction will make [enable jump signal] active high; if brach conditiction is established then pc\_jump control will active high enable to jump address.**  rst\_data=global\_rst?1'b1:( enable\_jump ? ( pc\_jump\_control ? local\_rst:1'b0) :1'b0);  **(for id\_exe stage)1.Global rst is cpu’s rst ; 2.Modele will check the pc\_stall signal to decide whether pause the pipeline register or not.Next, jump and branch instruction will make [enable jump signal] active high; if branch condition is established then pc\_jump control will active high enable to jump address.**  rst\_data=global\_rst?1'b1: ( pc\_stall ? 1'b1: ( enable\_jump ? ( pc\_jump\_control ? local\_rst:1'b0) :1'b0) ); | |
| MODULE | PC\_CONTROL |
|  |
| CONTROL FLOW | |
| pc\_stall signal will enable rewrite the instruction address again  enable\_jump signal will enable to jump address.  if(pc\_stall==1'b1)  pc\_data=pc;  else  pc\_data=enable\_jump?(pc\_jump\_control?pc\_jump\_address:next\_pc):next\_pc; | |

|  |  |  |  |
| --- | --- | --- | --- |
| R-TYPE INSTRUCTION VERIFICATION | | | |
| INSTRUCTION  ADD | | Machine code | Assembly code |
| 0x006282b3 | add t0 t0 t1 |
| wave | | | |
|  | | | |
| Explanation | | | |
| ID stage | control module(cl ) produce alu\_rd\_signal(function of the alu);active write\_reg signal; and register(rigt)read the data by the address decode by decoder  Decoder decode opcode:33 | | |
| EXE stage | Alu\_in\_selector detect the hazard signal produced by forwarding unit and choose the data in memory or alu  Alu\_rd (ard)calculate the value that will store to the $rd address | | |
| MEM  stage | By pass the value calculate by Alu\_rd  Wb\_control control the value calculate by Alu\_rd to be the write back data. | | |
| WB  stage | Write\_reg signal enable to write back data in register | | |

|  |  |  |  |
| --- | --- | --- | --- |
| R-TYPE INSTRUCTION VERIFICATION | | | |
| INSTRUCTION  AND | | Machine code | Assembly code |
| 0x0062f2b3 | and t0 t0 t1 |
| wave | | | |
|  | | | |
| Explanation | | | |
| ID stage | control module(cl ) produce alu\_rd\_signal(function of the alu);active write\_reg signal; and register(rigt)read the data by the address decode by decoder  Decoder decode opcode:33,funt3:7,fun7:0 | | |
| EXE stage | Alu\_in\_selector detect the hazard signal produced by forwarding unit and choose the data in memory or alu  Alu\_rd\_control signal=9 execute and function  Alu\_rd (ard)calculate the value that will store to the $rd address | | |
| MEM  stage | By pass the value calculate by Alu\_rd  Wb\_control control the value calculate by Alu\_rd to be the write back data. | | |
| WB  stage | Write\_reg signal enable to write back data in register | | |

|  |  |  |  |
| --- | --- | --- | --- |
| I-TYPE INSTRUCTION VERIFICATION | | | |
| INSTRUCTION  LW | | Machine code | Assembly code |
| 0x0002a283 | Lw t0 0(t0) |
| wave | | | |
|  | | | |
| Explanation | | | |
| ID stage | Decoder decode opcode:3,funt3:2  Control module’output extended\_control control the imm data  Control module produce read mem signal | | |
| EXE stage | Alu\_addr calculate the address(alu\_addr\_out) that is going to load data.  Because thers is a load data hazard ,the pc\_stall signal active high to pause instruction. | | |
| MEM  stage | The address is divided by 4 to be the real data memory address  The OE signal is active high because it connects with read mem signal  Wb\_control control the value loaded from memory to be the write back data. | | |
| WB  stage | Write\_reg signal enable to write back data in register  The valus is stored in register in 5 stage | | |

|  |  |  |  |
| --- | --- | --- | --- |
| I-TYPE INSTRUCTION VERIFICATION | | | |
| INSTRUCTION  ADDI | | Machine code | Assembly code |
| 0xfff00293 | addi t0 t0 -1 |
| wave | | | |
|  | | | |
| Explanation | | | |
| ID stage | Decoder decode opcode:19funt3:0,alu\_control:10  Control module’output extended\_control control the imm data | | |
| EXE stage | Alu\_addr calculate the address(alu\_addr\_out) that is going to load data.  The calculation way is selected by alu\_control | | |
| MEM  stage | By pass the value calculate by Alu\_rd  Wb\_control control the value calculate by Alu\_rd to be the write back data. | | |
| WB  stage | Write\_reg signal enable to write back data in register  The valus is stored in register in 5 stage | | |

|  |  |  |  |
| --- | --- | --- | --- |
| S-TYPE INSTRUCTION VERIFICATION | | | |
| INSTRUCTION  SW | | Machine code | Assembly code |
| 0xfe542e23 | Sw t0,-4(s0) |
| wave | | | |
|  | | | |
| Explanation | | | |
| ID stage | Decoder decode opcode:35funt3:2  Control module’output extended\_control control the imm data  Control module activate the write mem signal | | |
| EXE stage | Alu\_rd (ard)calculate the value that will store to the $rd address  The calculation way is selected by alu\_control | | |
| MEM  stage | The address is divided by 4 to be the real data memory address  The 4 bits signal is active low because it connects with read mem signal  The data from register is stored in negedge cycle | | |
| WB  stage | NO ACTION | | |

|  |  |  |  |
| --- | --- | --- | --- |
| S-TYPE INSTRUCTION VERIFICATION | | | |
| INSTRUCTION  SB | | Machine code | Assembly code |
| 0xfe542e23 | Sw t0,-4(s0) |
| wave | | | |
|  | | | |
| Explanation | | | |
| ID stage | Decoder decode opcode:35funt3:0  Control module’output extended\_control control the imm data  Control module activate the write mem signal  Control module activate the low byte write signal | | |
| EXE stage | Alu\_rd (ard)calculate the value that will store to the $rd address  The calculation way is selected by alu\_control | | |
| MEM  stage | The address is divided by 4 to be the real data memory address  The 4 bits signal is 1110 because the low byte control signal only allow write in the low 8bits data.  The data from register is stored in negedge cycle | | |
| WB  stage | NO ACTION | | |