Verilog搭建流水线CPU设计报告

1. 数据通路设计

1.PC

(1)接口定义

|  |  |  |
| --- | --- | --- |
| 信号名称 | 方向 | 描述 |
| reset | I | PC复位至0x00003000 |
| NPC[31:0] | I | 下一个PC值（下一个时钟上升沿的时候要写入PC寄存器的值） |
| PC[31:0] | O | PC输出 |

（2）PC功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 复位 | 当reset有效时，PC寄存器被赋值为0x00003000 |
| 2 | 输出指令地址 | 在时钟上升沿的时候更新PC |

2.ADD4模块

(1)接口定义

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| PC[31：0] | I | 当前PC值 |
| PC4[31：0] | O | =PC+4 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 输出PC+4 | 纯组合逻辑，永远输出PC4=PC+4 |

3. IM模块

（1）接口定义

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk,reset | I | 时钟，复位信号 |
| PC[31:0] | I | 要读取的指令的地址 |
| instr[31:0] | O | 从IM中取出来的32位指令 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 取指令 | 根据输入的PC值输出相应位置的指令 |

4、RF

GRF由32个寄存器构成（其中0号寄存器恒为0）。首先计算RegA、RegB、RegD信号的值，再进行读或写操作：读操作时，将编号为RegA、RegB的两个寄存器的值读出到busA、busB。写操作时，在RegWr信号为1时，将dataWr写入RegD编号对应的寄存器中。

表格 1 GRF端口说明

|  |  |  |
| --- | --- | --- |
| 端口名 | 方向 | 说明 |
| IR\_D[31:0] | I | D级指令，用来提取rs,rt |
| R3[4:0] | I | 待写入寄存器的编号 |
| RFIn[31:0] | I | 待写入寄存器的数据 |
| RegWr | I | 寄存器写使能信号  0：不写入 1：写入 |
| clk | I | 时钟信号 |
| reset | I | 复位信号 |
| PC4[31:0] | I | PC+4值，$display时-4后使用 |
| RFOut1[31:0] | O | 读出数据1($rs的值) |
| RFOut2[31:0] | O | 读出数据2($rt的值) |

表格 2 GRF功能定义

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 功能名称 | 功能描述 | |
| 1 | 读寄存器 | 将编号为R1的寄存器中的数据输出到RFOUt1端口；  将编号为R2的寄存器中的数据输出到RFOut2端口 |
| 2 | 写寄存器 | RegWr=1时，在时钟上升沿将RFIn写入到编号为R3的寄存器中；写入寄存器时进行输出操作 |

5、EXT（扩展器）

EXT根据ExtOp对16位立即数imm16进行各类扩展。

表格 3 EXT端口说明

|  |  |  |
| --- | --- | --- |
| 端口名 | 方向 | 说明 |
| IR\_D[31:0] | I | D级指令，用来提取16位立即数imm16 |
| ExtOp[1:0] | I | 进行何种扩展的选择信号。  00：无符号扩展  01：有符号扩展  10：加载到高16位，低16位补0  11：（未定义） |
| ExtOut[31:0] | O | 扩展后的数据。 |

表格 4 EXT功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 无符号扩展 | ExtOp=00时，对imm16进行无符号扩展并输出到ExtOut。 |
| 2 | 有符号扩展 | ExtOp=01时，对imm16进行有符号扩展并输出到ExtOut。 |
| 3 | 后补16位0 | ExtOp=10时，将imm16加载到高16位，在低16位补0，并输出到ExtOut。 |

6.CMP模块

（1）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名称 | 方向 | 描述 |
| CMPIn1[31：0] | I | 第一个操作数 |
| CMPIn2[31：0] | I | 第二个操作数 |
| BrType[2：0] | I | Br指令类型，比较方式选择 |
| BrTrue | O | 比较结果，用于是否做br跳转的判断 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述(C为BrType;A,B为CMPIn1,CMPIn2) |
| 1 | ==(BrType=000) | C=(A==B) |
| 2 | !=(BrType=001) | C=(A!=B) |
| 3 | A>B(BrType=010) | C=(A>B) |
| 4 | A<=B(BrType=011) | C=(A<=B) |
| 5 | A<B(BrType=100) | C=(A<B) |
| 6 | A>=0(BrType=101) | C=(A>=0) |

7.NPC模块

(1)接口定义(计算下一个PC的值，纯组合逻辑)

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| PC4[31：0] | I | 当前PC值 |
| IR\_D[31：0] | I | D级指令，用于提取imm16和imm26 |
| BrTrue | I | 从CMP块得到的是否满足br指令跳转条件的信号 |
| jPC[31:0] | O | j指令的下一个PC |
| brPC[31:0] | O | br指令的下一个PC |

（2）NPC功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 计算brPC | BrTrue=0时，brPC=PC4+4  BrTrue=1时，brPC=PC4+sign\_extend(offset||00) |
| 2 | 计算jPC | jrPC= PC4[31:28] || imm26 || 00 |

8、ALU（算术逻辑单元）

ALU由何种算数逻辑组成。根据ALUctr对ALUin1和ALUin2进行加、减、或、相等比较等操作并输出。

表格 5 ALU端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 端口名 | 方向 | | 说明 |
| ALUIn1[31:0] | I | 第一个待操作数。 | |
| ALUIn2[31:0] | I | 第二个待操作数。 | |
| ALUOp[2:0] | I | 进行何种运算的选择信号。  00：ALUIn1+ALUIn2  01：ALUIn1-ALUIn2  10：ALUIn1 | ALUIn2  11：ALUIn2 << ALUIn1[4:0] | |
| ALUOut[31:0] | O | 运算后的结果。 | |

表格 6 ALU功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 加（无溢出） | ALUOut=ALUin1 + ALUin2 |
| 2 | 减（无溢出） | ALUOut=ALUin1 - ALUin2 |
| 3 | 或 | ALUOut=ALUin1 | ALUin2 |
| 4 | 移位 | ALUOut=ALUIn2 << ALUIn1[4:0] |

9、DM（数据存储器）

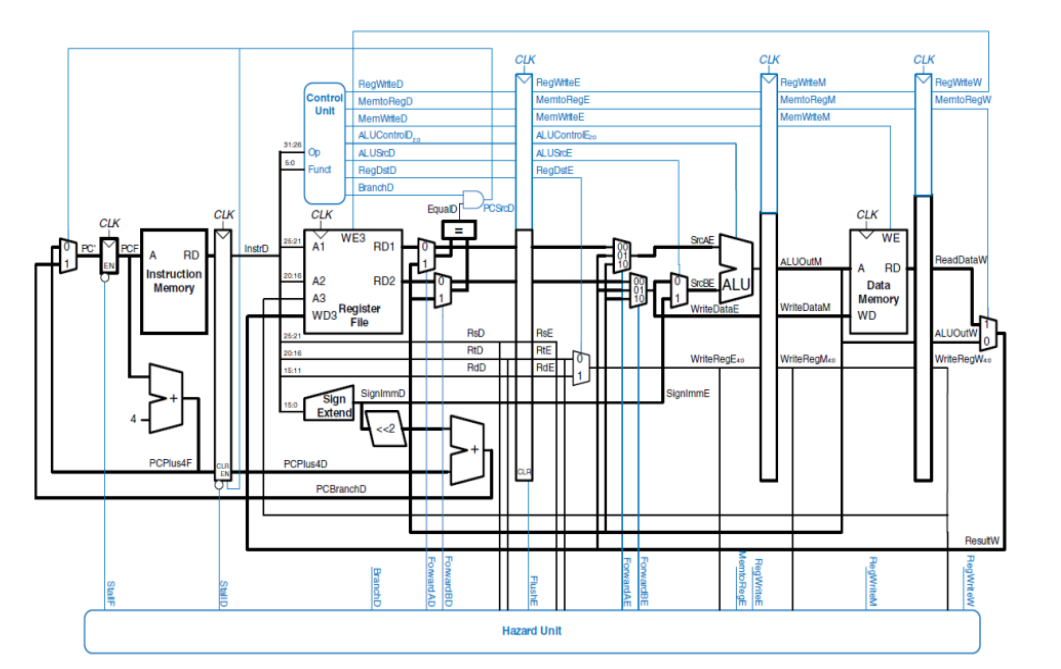
表格 7 DM端口说明

|  |  |  |
| --- | --- | --- |
| 端口名 | 方向 | 说明 |
| DMAddr[31:0] | I | DM中的读出/写入地址，即ALU的输出端ALUOut |
| DMIn[31:0] | I | 待写入DM的数据,即GRF的输出端RFOut2 |
| MemWr | I | 将DMdata写入DM的写使能信号。  0：不写入 1：写入 |
| clk | I | 时钟信号 |
| reset | I | 复位信号 |
| PC4[31:0] | I | PC4值，$display时-4后使用 |
| DMOut[31:0] | O | DM输出数据 |

表格 8 DM功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 读出 | DMOut=DM中DMAddr地址中的数据。 |
| 2 | 写入 | MemWr=1时，将DMIn写入DM的DMaddr地址中。 |

10、电路图总览



1. Controller（控制器）设计

1、基本描述

Controller根据指令中的opcode段和funct段，先利用与门确定该指令类型，再利用或门确定各控制信号。

|  |  |  |
| --- | --- | --- |
| 端口名 | 方向 | 说明 |
| instr[31:0] | I | 指令 |
| ALUSrc1 | O | 控制信号  说明见Excel |
| ALUSrc2 | O |
| MemWr | O |
| RegWr | O |
| ExtOp[1:0] | O |
| ALUOp[2:0] | O |
| NPCsel[1:0] | O |
| RegDst[1:0] | O |
| MemtoReg[1:0] | O |
| BrType | O |
| calr | O | 指令类型  说明见Excel |
| cali | O |
| br | O |
| load | O |
| store | O |
| jal | O |
| jr | O |
| jalr | O |

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 译码 | 将instr根据上表所述进行译码 |

2、控制信号真值表（见Excel）

3、控制信号含义（见Excel）

1. 测试CPU
2. 指令功能测试(instruction function)

期待输出：

$ 8 <= ffff0000

$ 9 <= 00000001

$10 <= 00000064

$ 8 <= ffffffff

$11 <= 00000000

$12 <= fffffffe

$13 <= 00000065

$14 <= fffffffe

$15 <= 00000000

$16 <= 00000063

\*00000000 <= 00000000

\*00000004 <= fffffffe

$17 <= 00000000

$18 <= fffffffe

$ 8 <= ffff0000

$ 9 <= 00000001

$10 <= 00000064

$11 <= 00000001

$ 8 <= ffffffff

$ 8 <= 00000000

$10 <= 00000065

$10 <= 00000066

$ 8 <= 00000001

$31 <= 000030ac

$12 <= 00000032

$13 <= 00000033

$12 <= 00000033

$10 <= 00000067

实际输出：

$ 8 <= ffff0000

$ 9 <= 00000001

$10 <= 00000064

$ 8 <= ffffffff

$11 <= 00000000

$12 <= fffffffe

$13 <= 00000065

$14 <= fffffffe

$15 <= 00000000

$16 <= 00000063

\*00000000 <= 00000000

\*00000004 <= fffffffe

$17 <= 00000000

$18 <= fffffffe

$ 8 <= ffff0000

$ 9 <= 00000001

$10 <= 00000064

$11 <= 00000001

$ 8 <= ffffffff

$ 8 <= 00000000

$10 <= 00000065

$10 <= 00000066

$ 8 <= 00000001

$31 <= 000030ac

$12 <= 00000032

$13 <= 00000033

$12 <= 00000033

$10 <= 00000067

1. 暂停测试(stall)

期待输出：

$ 1 <= 00001234

$ 2 <= 00001234

\*00000000 <= 00001234

$ 1 <= 00005678

$ 2 <= 00005678

\*00000004 <= 00005678

$ 1 <= 00001234

$ 2 <= 00001234

\*00000008 <= 00001234

$ 1 <= 00005678

$ 2 <= 00005678

\*0000000c <= 00005678

$ 1 <= 00000000

$ 1 <= 00001234

$ 2 <= 00001234

$ 0 <= 00005678

$ 1 <= 00640000

$ 1 <= 00005678

$ 2 <= 00005678

$ 1 <= 00000008

\*00000008 <= 00000008

$ 2 <= 00000008

$ 1 <= 00000008

$ 1 <= 00000010

\*00000010 <= 00000010

$ 2 <= 00000010

\*00000010 <= 00000010

$31 <= 000030a4

$ 1 <= 00000014

$ 1 <= 000030b8

\*00000008 <= 000030b8

$ 2 <= 000030b8

$31 <= 000030c4

$ 1 <= 0000000c

$ 1 <= 000030d0

$31 <= 000030dc

$ 1 <= 00000010

$ 1 <= 000030ec

$ 2 <= 000030ec

$31 <= 000030f8

$ 1 <= 00000018

$ 1 <= 00003110

\*00000008 <= 00003110

$ 2 <= 00003110

$31 <= 0000311c

$ 1 <= 0000000c

$ 1 <= 00003128

$ 2 <= 0000312c

$ 1 <= 00000010

$ 2 <= 0000313c

$ 1 <= 0000313c

$ 2 <= 00003140

$ 1 <= 00000014

$ 2 <= 00003154

\*00000008 <= 00003154

$ 1 <= 00003154

$ 2 <= 00003158

$ 1 <= 00000018

$ 2 <= 00003170

\*00000008 <= 00003170

$ 1 <= 00003170

$ 2 <= 00003174

$ 1 <= 00001234

$ 0 <= 00001234

\*00000000 <= 00003174

$ 1 <= 00005678

$ 2 <= 00000000

$ 0 <= 00005678

\*00000004 <= 00000000

$ 1 <= 00000000

$ 0 <= 00003174

\*00000008 <= 00000000

$ 1 <= 00000000

$ 0 <= 00000000

\*0000000c <= 00000000

$ 1 <= 00000000

$ 0 <= 00003174

$ 2 <= 00000000

$ 0 <= 00000000

$ 1 <= 00640000

$ 0 <= 00000000

$ 2 <= 00000000

$ 0 <= 00000008

\*00000008 <= 00000000

$ 0 <= 00000000

$ 0 <= 00003174

$ 0 <= 00640000

\*00000010 <= 00000000

$ 0 <= 00000000

\*00000000 <= 00000000

实际输出：

$ 1 <= 00001234

$ 2 <= 00001234

\*00000000 <= 00001234

$ 1 <= 00005678

$ 2 <= 00005678

\*00000004 <= 00005678

$ 1 <= 00001234

$ 2 <= 00001234

\*00000008 <= 00001234

$ 1 <= 00005678

$ 2 <= 00005678

\*0000000c <= 00005678

$ 1 <= 00000000

$ 1 <= 00001234

$ 2 <= 00001234

$ 0 <= 00005678

$ 1 <= 00640000

$ 1 <= 00005678

$ 2 <= 00005678

$ 1 <= 00000008

\*00000008 <= 00000008

$ 2 <= 00000008

$ 1 <= 00000008

$ 1 <= 00000010

\*00000010 <= 00000010

$ 2 <= 00000010

\*00000010 <= 00000010

$31 <= 000030a4

$ 1 <= 00000014

$ 1 <= 000030b8

\*00000008 <= 000030b8

$ 2 <= 000030b8

$31 <= 000030c4

$ 1 <= 0000000c

$ 1 <= 000030d0

$31 <= 000030dc

$ 1 <= 00000010

$ 1 <= 000030ec

$ 2 <= 000030ec

$31 <= 000030f8

$ 1 <= 00000018

$ 1 <= 00003110

\*00000008 <= 00003110

$ 2 <= 00003110

$31 <= 0000311c

$ 1 <= 0000000c

$ 1 <= 00003128

$ 2 <= 0000312c

$ 1 <= 00000010

$ 2 <= 0000313c

$ 1 <= 0000313c

$ 2 <= 00003140

$ 1 <= 00000014

$ 2 <= 00003154

\*00000008 <= 00003154

$ 1 <= 00003154

$ 2 <= 00003158

$ 1 <= 00000018

$ 2 <= 00003170

\*00000008 <= 00003170

$ 1 <= 00003170

$ 2 <= 00003174

$ 1 <= 00001234

$ 0 <= 00001234

\*00000000 <= 00003174

$ 1 <= 00005678

$ 2 <= 00000000

$ 0 <= 00005678

\*00000004 <= 00000000

$ 1 <= 00000000

$ 0 <= 00003174

\*00000008 <= 00000000

$ 1 <= 00000000

\*0000000c <= 00000000

$ 1 <= 00000000

$ 0 <= 00003174

$ 2 <= 00000000

$ 1 <= 00640000

$ 2 <= 00000000

$ 0 <= 00000008

\*00000008 <= 00000000

$ 0 <= 00003174

$ 0 <= 00640000

\*00000010 <= 00000000

\*00000000 <= 00000000

1. 转发测试1-5(forwarding)

期待输出：

$ 4 <= 00000004

$ 5 <= 00000005

$ 6 <= 00000006

\*00000000 <= 00000005

\*00000004 <= 00000004

\*00000008 <= 00000006

$ 5 <= 00000005

$ 5 <= 0000000a

$ 5 <= 00000005

$ 7 <= 00000001

$ 5 <= 0000000a

$ 5 <= 00000005

$ 7 <= 00000003

$ 8 <= 00000003

$ 5 <= 0000000a

$ 5 <= 00000005

$ 5 <= 0000000a

$ 5 <= 00000014

$ 5 <= 00000028

$10 <= 00000004

$10 <= ffffffdc

$10 <= 00000004

$11 <= 0000000b

$10 <= 00000000

$10 <= 00000004

$11 <= 0000000f

$11 <= 0000000f

$10 <= 00000000

$10 <= 00000004

$11 <= 00000000

$11 <= 00000000

$11 <= 00000000

$12 <= 00000005

$12 <= 00000005

$12 <= 00000004

$13 <= 0000000d

$12 <= 00000004

$12 <= 00000006

$13 <= 0000000d

$13 <= 0000000f

$12 <= 00000006

$12 <= 00000006

$12 <= 00000006

$12 <= 00000006

$12 <= 00000006

$14 <= 00000005

\*00000000 <= 00000005

$14 <= 00000004

$15 <= 00000005

\*00000004 <= 00000004

$14 <= 00000006

$15 <= 00000007

$15 <= 00000007

\*00000008 <= 00000006

$16 <= 00000004

$17 <= 00000005

$18 <= 00000006

$19 <= 00000004

$16 <= 00000004

$16 <= 00000004

$16 <= 00000004

$16 <= 00000004

$31 <= 00003100

$25 <= 00000019

$31 <= 000030fc

\*0000000c <= 000030fc

$31 <= 000030fc

$25 <= 00000019

$26 <= 0000001a

$27 <= 0000001b

$31 <= 00003134

$25 <= 00000019

$31 <= 00003130

\*0000000c <= 00003130

$31 <= 00003130

$25 <= 00000019

$26 <= 0000001a

$27 <= 0000001b

$31 <= 00003168

$25 <= 00000019

$31 <= 00003164

\*0000000c <= 00003164

$31 <= 00003164

$25 <= 00000019

$26 <= 0000001a

$27 <= 0000001b

$20 <= 00000005

$21 <= 00000004

$23 <= 00000017

$24 <= 00000018

$25 <= 00000019

$20 <= 00000005

$21 <= 00000004

$23 <= 00000017

$24 <= 00000018

$25 <= 00000019

实际输出：

$ 4 <= 00000004

$ 5 <= 00000005

$ 6 <= 00000006

\*00000000 <= 00000005

\*00000004 <= 00000004

\*00000008 <= 00000006

$ 5 <= 00000005

$ 5 <= 0000000a

$ 5 <= 00000005

$ 7 <= 00000001

$ 5 <= 0000000a

$ 5 <= 00000005

$ 7 <= 00000003

$ 8 <= 00000003

$ 5 <= 0000000a

$ 5 <= 00000005

$ 5 <= 0000000a

$ 5 <= 00000014

$ 5 <= 00000028

$10 <= 00000004

$10 <= ffffffdc

$10 <= 00000004

$11 <= 0000000b

$10 <= 00000000

$10 <= 00000004

$11 <= 0000000f

$11 <= 0000000f

$10 <= 00000000

$10 <= 00000004

$11 <= 00000000

$11 <= 00000000

$11 <= 00000000

$12 <= 00000005

$12 <= 00000005

$12 <= 00000004

$13 <= 0000000d

$12 <= 00000004

$12 <= 00000006

$13 <= 0000000d

$13 <= 0000000f

$12 <= 00000006

$12 <= 00000006

$12 <= 00000006

$12 <= 00000006

$12 <= 00000006

$14 <= 00000005

\*00000000 <= 00000005

$14 <= 00000004

$15 <= 00000005

\*00000004 <= 00000004

$14 <= 00000006

$15 <= 00000007

$15 <= 00000007

\*00000008 <= 00000006

$16 <= 00000004

$17 <= 00000005

$18 <= 00000006

$19 <= 00000004

$16 <= 00000004

$16 <= 00000004

$16 <= 00000004

$16 <= 00000004

$31 <= 00003100

$25 <= 00000019

$31 <= 000030fc

\*0000000c <= 000030fc

$31 <= 000030fc

$25 <= 00000019

$26 <= 0000001a

$27 <= 0000001b

$31 <= 00003134

$25 <= 00000019

$31 <= 00003130

\*0000000c <= 00003130

$31 <= 00003130

$25 <= 00000019

$26 <= 0000001a

$27 <= 0000001b

$31 <= 00003168

$25 <= 00000019

$31 <= 00003164

\*0000000c <= 00003164

$31 <= 00003164

$25 <= 00000019

$26 <= 0000001a

$27 <= 0000001b

$20 <= 00000005

$21 <= 00000004

$23 <= 00000017

$24 <= 00000018

$25 <= 00000019

$20 <= 00000005

$21 <= 00000004

$23 <= 00000017

$24 <= 00000018

$25 <= 00000019

四、思考题

1、在本实验中你遇到了哪些不同指令组合产生的冲突？你又是如何解决的？相应的测试样例是什么样的？请有条理的罗列出来。(**非常重要**)

暂停

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 类型 | 测试类型 | 前序指令 | 冲突位置 | 冲突寄存器 | 测试序列 |
| R型 | LD-E-RS | lw | D | RS | lw $4,0($5)  addu $4,$4,$5 |
|  | LD-E-RT | lw | D | RT | lw $4,0($5)  addu $4,$5,$4 |
| I型 | LD-E-RS | lw | D | RS | lw $4,0($5)  ori $5,$4,0xffff |
| LD型 | LD-E-RS | lw | D | RS | lw $4,0($5)  lw $3,4($4) |
| ST型 | LD-E-RS | lw | D | RS | lw $4,0($5)  sw $3,4($4) |
| JR | R-E-RS | addu | D | RS | addu $4,$4,$5  jr $4 |
|  | I-E-RS | ori | D | RS | ori $4,0xffff  jr $4 |
|  | LD-E-RS | lw | D | RS | lw $4,0($5)  jr $4 |
|  | LD-M-RS | lw | D | RS | lw $4,0($5)  nop  jr $4 |
| B型 | R-E-RS | addu | D | RS | addu $4,$4,$5  beq $4,$5,loop |
|  | I-E-RS | ori | D | RS | ori $4,0xffff  beq $4,$5,loop |
|  | LD-E-RS | lw | D | RS | lw $4,0($5)  beq $4,$5,loop |
|  | LD-M-RS | lw | D | RS | lw $4,0($5)  nop  beq $4,$5,loop |
|  | R-E-RT | addu | D | RT | addu $4,$4,$5  beq $5,$4,loop |
|  | I-E-RT | ori | D | RT | ori $4,0xffff  beq $5,$4,loop |
|  | LD-E-RT | lw | D | RT | lw $4,0($5)  beq $5,$4,loop |
|  | LD-M-RT | lw | D | RT | lw $4,0($5)  nop  beq $5,$4,loop |

转发

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 类型 | 测试类型 | 前序指令 | 冲突位置 | 冲突寄存器 | 测试序列 |
| R型  (以addu为例) | R-W-RS | addu | D | RS | addu $4,$4,$5  nop  nop  addu $4,$4,$5 |
|  | I-W-RS | ori | D | RS | ori $4,$5,0xffff  nop  nop  addu $4,$4,$5 |
|  | LD-W-RS | lw | D | RS | lw $4,0($5)  nop  nop  addu $4,$4,$5 |
|  | JAL-W-RS | jal | D | RS | jal loop  nop  nop  addu $1,$31,$1 |
|  | R-W-RT | addu | D | RT | addu $4,$4,$5  nop  nop  addu $4, $5,$4 |
|  | I-W-RT | ori | D | RT | ori $4,$5,0xffff  nop  nop  addu $4,$5,$4 |
|  | LD-W-RT | lw | D | RT | lw $4,0($5)  nop  nop  addu $4,$5,$4 |
|  | JAL-W-RT | jal | D | RT | jal loop  nop  nop  addu $1,$1,$31 |
|  | R-W-RS | addu | E | RS | addu $4,$4,$5  nop  addu $4,$4,$5  nop |
|  | I-W-RS | ori | E | RS | ori $4,$5,0xffff  nop  addu $4,$4,$5  nop |
|  | LD-W-RS | lw | E | RS | lw $4,0($5)  nop  addu $4,$4,$5  nop |
|  | JAL-W-RS | jal | E | RS | jal loop  nop  addu $1,$31,$1  nop |
|  | R-M-RS | addu | E | RS | addu $4,$4,$5  addu $4,$4,$5  nop |
|  | I-M-RS | ori | E | RS | ori $4,$5,0xffff  addu $4,$4,$5  nop |
|  | JAL-M-RS | jal | E | RS | jal loop  addu $1,$31,$1  nop |
|  | R-W-RT | addu | E | RT | addu $4,$4,$5  nop  addu $4, $5,$4  nop |
|  | I-W-RT | ori | E | RT | ori $4,$5,0xffff  nop  addu $4,$5,$4  nop |
|  | LD-W-RT | lw | E | RT | lw $4,0($5)  nop  addu $4,$5,$4  nop |
|  | JAL-W-RT | jal | E | RT | jal loop  nop  addu $1,$1,$31  nop |
|  | R-M-RT | addu | E | RT | addu $4,$4,$5  addu $4,$5,$4  nop |
|  | I-M-RT | ori | E | RT | ori $4,$5,0xffff  addu $4,$5,$4  nop |
|  | JAL-M-RT | jal | E | RT | jal loop  addu $1,$1,$31  nop |
| I型  (以ori为例) | R-W-RS | addu | D | RS | addu $4,$4,$5  nop  nop  ori $4,$4,0xffff |
|  | I-W-RS | ori | D | RS | ori $4,$5,0xffff  nop  nop  ori $4,$4,0x0000 |
|  | LD-W-RS | lw | D | RS | lw $4,0($5)  nop  nop  ori $4,$4,0xffff |
|  | JAL-W-RS | jal | D | RS | jal loop  nop  nop  ori $1,$31,0xffff |
|  | R-W-RS | addu | E | RS | addu $4,$4,$5  nop  ori $4,$4,0xffff  nop |
|  | I-W-RS | ori | E | RS | ori $4,$5,0xffff  nop  ori $4,$4,0x0f0f  nop |
|  | LD-W-RS | lw | E | RS | lw $4,0($5)  nop  ori $4,$4,0xffff  nop |
|  | JAL-W-RS | jal | E | RS | jal loop  nop  ori $1,$31,0xffff  nop |
|  | R-M-RS | addu | E | RS | addu $4,$4,$5  ori $4,$4,0xffff  nop |
|  | I-M-RS | ori | E | RS | ori $4,$5,0xffff  ori $4,$4,0xf0f0  nop |
|  | JAL-M-RS | jal | E | RS | jal loop  ori $1,$31,0xfff0  nop |
| LD型 | R-W-RS | addu | D | RS | addu $4,$4,$5  nop  nop  lw $5,0($4) |
|  | I-W-RS | ori | D | RS | ori $4,$5,0xffff  nop  nop  lw $5,0($4) |
|  | LD-W-RS | lw | D | RS | lw $4,0($5)  nop  nop  lw $5,0($4) |
|  | JAL-W-RS | jal | D | RS | jal loop  nop  nop  lw $5,0($31) |
|  | R-W-RS | addu | E | RS | addu $4,$4,$5  nop  lw $5,0($4)  nop |
|  | I-W-RS | ori | E | RS | ori $4,$5,0xffff  nop  lw $5,0($4)  nop |
|  | LD-W-RS | lw | E | RS | lw $4,0($5)  nop  lw $5,0($4)  nop |
|  | JAL-W-RS | jal | E | RS | jal loop  nop  lw $5,0($31)  nop |
|  | R-M-RS | addu | E | RS | addu $4,$4,$5  lw $5,0($4)  nop |
|  | I-M-RS | ori | E | RS | ori $4,$5,0xffff  lw $5,0($4)  nop |
|  | JAL-M-RS | jal | E | RS | jal loop  lw $5,0($31)  nop |
| ST型 | R-W-RS | addu | D | RS | addu $4,$4,$5  nop  nop  sw $5,0($4) |
|  | I-W-RS | ori | D | RS | ori $4,$5,0xffff  nop  nop  sw $5,0($4) |
|  | LD-W-RS | lw | D | RS | lw $4,0($5)  nop  nop  sw $5,0($4) |
|  | JAL-W-RS | jal | D | RS | jal loop  nop  nop  sw $5,0($31) |
|  | R-W-RT | addu | D | RT | addu $4,$4,$5  nop  nop  sw $4,0($5) |
|  | I-W-RT | ori | D | RT | ori $4,$5,0xffff  nop  nop  sw $4,0($5) |
|  | LD-W-RT | lw | D | RT | lw $4,0($5)  nop  nop  sw $4,0($6) |
|  | JAL-W-RT | jal | D | RT | jal loop  nop  nop  sw $31,0($5) |
|  | R-W-RS | addu | E | RS | addu $4,$4,$5  nop  sw $5,0($4)  nop |
|  | I-W-RS | ori | E | RS | ori $4,$5,0xffff  nop  sw $5,0($4)  nop |
|  | LD-W-RS | lw | E | RS | lw $4,0($5)  nop  sw $5,0($4)  nop |
|  | JAL-W-RS | jal | E | RS | jal loop  nop  sw $5,0($31)  nop |
|  | R-W-RT | addu | E | RT | addu $4,$4,$5  nop  sw $4,0($5)  nop |
|  | I-W-RT | ori | E | RT | ori $4,$5,0xffff  nop  sw $4,0($5)  nop |
|  | LD-W-RT | lw | E | RT | lw $4,0($5)  nop  sw $4,0($6)  nop |
|  | JAL-W-RT | jal | E | RT | jal loop  nop  sw $31,0($5)  nop |
|  | R-W-RT | addu | M | RT | addu $4,$4,$5  sw $4,0($5)  nop  nop |
|  | I-W-RT | ori | M | RT | ori $4,$5,0xffff  sw $4,0($5)  nop  nop |
|  | LD-W-RT | lw | M | RT | lw $4,0($5)  sw $4,0($6)  nop  nop |
|  | JAL-W-RT | jal | M | RT | jal loop  sw $31,0($5)  nop  nop |
|  | R-M-RS | addu | E | RS | addu $4,$4,$5  sw $5,0($4)  nop |
|  | I-M-RS | ori | E | RS | ori $4,$4,$5  sw $5,0($4)  nop |
|  | JAL-M-RS | jal | E | RS | jal loop  sw $5,0($31)  nop |
| JR | R-W-RS | addu | D | RS | addu $4,$4,$5  nop  nop  jr $4 |
|  | I-W-RS | ori | D | RS | ori $4,$5,0xffff  nop  nop  jr $4 |
|  | LD-W-RS | lw | D | RS | lw $4,0($5)  nop  nop  jr $4 |
|  | JAL-W-RS | jal | D | RS | jal loop  nop  nop  jr $31 |
|  | R-M-RS | addu | D | RS | addu $4,$4,$5  nop  jr $4 |
|  | I-M-RS | ori | D | RS | ori $4,$5,0xffff  nop  jr $4 |
|  | JAL-M-RS | jal | D | RS | jal loop  nop  jr $31 |
| 行为未定义 | JAL-E-RS | jal | D | RS | jal loop  jr $31 |
| B型 | R-W-RS | addu | D | RS | addu $4,$4,$5  nop  nop  beq $4,$3,loop |
|  | I-W-RS | ori | D | RS | ori $4,$5,0xffff  nop  nop  beq $4,$3,loop |
|  | LD-W-RS | lw | D | RS | lw $4,0($5)  nop  nop  beq $4,$3,loop |
|  | JAL-W-RS | jal | D | RS | jal loop  nop  nop  beq $31,$3,loop |
|  | R-W-RT | addu | D | RT | addu $4,$4,$5  nop  nop  beq $3,$4,loop |
|  | I-W-RT | ori | D | RT | ori $4,$5,0xffff  nop  nop  beq $3,$4,loop |
|  | LD-W-RT | lw | D | RT | lw $4,0($5)  nop  nop  beq $3,$4,loop |
|  | JAL-W-RT | jal | D | RT | jal loop  nop  nop  beq $3,$31,loop |
|  | R-M-RS | addu | D | RS | addu $4,$4,$5  nop  beq $4,$3,loop |
|  | I-M-RS | ori | D | RS | ori $4,$5,0xffff  nop  beq $4,$3,loop |
|  | JAL-M-RS | jal | D | RS | jal loop  nop  beq $31,$3,loop |
|  | R-M-RT | addu | D | RT | addu $4,$4,$5  nop  beq $3,$4,loop |
|  | I-M-RT | ori | D | RT | ori $4,$5,0xffff  nop  beq $3,$4,loop |
|  | JAL-M-RT | jal | D | RT | jal loop  nop  beq $3,$31,loop |
| 行为未定义 | JAL-E-RS | jal | D | RS | jal loop  beq $31,$3,loop |
| JAL-E-RT | jal | D | RT | jal loop  beq $3,$31,loop |

附：比对程序（C）

#include<stdio.h>

#include<string.h>

char a[100];

char b[100];

int main(){

FILE \*fp1=fopen("mips.txt","r");

FILE \*fp2=fopen("verilog.txt","r");

//freopen("res.txt","w+",stdout);

while((fscanf(fp1,"%s",a))!=EOF){

fscanf(fp2,"%s",b);

if(strcmp(a,b)==0)printf("right:r:%s w:%s\n",b,a);

else {printf("wrong:r:%s w:%s\n",b,a);break;}

}

return 0;

}