P5 - Verilog流水（工程化方法）

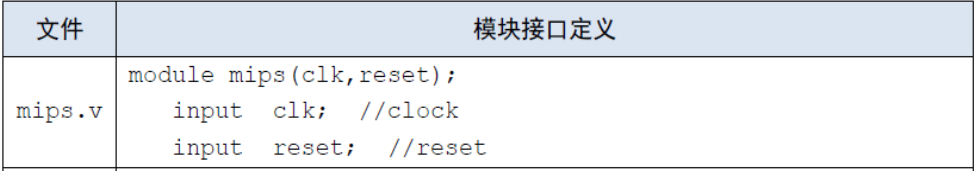
一．整体结构

1. 处理器应支持MIPS-lite2指令集。

MIPS-lite2={ addu, subu, ori, lw, sw, beq, lui, j, jal, jr, nop }

2. 处理器为流水线设计。

3. 顶层文件为mips.v，接口定义如下：



二．模块规格

1.pc.v

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| pc.v | module pc(  input clk,  input reset,  input en,  input[31:0] next\_pc,  output reg[31:0] pc  ); |

模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 功能描述 |
| Clk | I | 时钟信号 |
| Reset | I | 复位信号  1：复位  0：无效 |
| en | I | 使能信号 |
| next\_pc | I | 更新的PC（时钟上升沿更新） |
| Pc | I | PC |

功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | 当复位信号有效时，PC被设置为0x00003000 |
| 2 | 更新pc | 时钟上升沿时改变pc=next\_pc |

2.im.v

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| im.v | module im(  input [31:0] PC,  output[31:0] instruction  ); |

模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 功能描述 |
| PC[31:0] | I | 32位PC |
| Instruction[31:0] | O | 32位当前指令 |

功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 取指令 | 根据 PC 从 IM 中取出指令 |

3.ID.v

模块接口

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| ID.v | module ID(  input clk,  input reset,  input en,  input [31:0] Instr,  input [31:0] PC,  output reg[31:0] IR\_D,  output reg[31:0] PC\_D,  output reg[31:0] PC4\_D,  output reg[31:0] PC8\_D  ); |

功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | IF/ID流水线寄存器 | 保存PC,IR等信号的值 |

4.grf.v

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| grf.v | module grf(  input clk,  input reset,  input RegWrite,  input [4:0] RA1,  input [4:0] RA2,  input [4:0] WA,  input [31:0] WD,  input [31:0] PC,  output [31:0] RD1,  output [31:0] RD2  ); |

模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 功能描述 |
| WD[31:0] | I | 写入数据的输入 |
| RA1[4:0] | I | 读寄存器地址 1 |
| RA2[4:0] | I | 读寄存器地址 2 |
| WA[4:0] | I | 写寄存器地址 |
| Clk | I | 时钟信号 |
| Reset | I | 复位信号  1：复位  0：无效 |
| PC[31:0] | I | 当前PC |
| RegWrite | I | 是否可以写入控制信号(随时都可以读出)  1：可以写  0：不可以写 |
| RD1[31:0] | O | 32 位数据输出 1 |
| RD2[31:0] | O | 32 位数据输出 2 |

功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | 当复位信号有效时，所有寄存器被设置为 0x00000000 |
| 2 | 读寄存器 | 根据输入的寄存器地址读出32位数据 |
| 3 | 写寄存器 | 根据输入的地址，把输入的数据写进所选的寄存器 |

5.cmp.v

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| cmp.v | module cmp(  input [31:0] D1,  input [31:0] D2,  output Equal  ); |

模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 功能描述 |
| D1[31:0] | I | 输入1 |
| D2[31:0] | I | 输入2 |
| Equal | O | 判断两个输入是否相等 |

功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 比较器 | 比较两个输入是否相等 |

6. ext.v

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| ext.v | module ext(  input [15:0] in,  input [1:0] ExtOp,  output reg [31:0] out  ); |

模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 功能描述 |
| In[15:0] | I | 16 位数据输入 |
| Out[31:0] | O | 32 位数据输出 |
| ExtOp[1:0] | I | 扩展方式选择信号 |

功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 高位符号扩展 | 高16位补符号位 |
| 2 | 高位0扩展 | 高16位补0 |
| 3. | 低位0扩展 | 低16位补0 |

7. npc.v

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| npc.v | module npc(  input [31:0] PC4,  input [31:0] PC4D,  input [25:0] I26,  input [31:0] MFRSD,  input Zero,  input Branch,  input if\_j,//j或jal  input[1:0] PC\_sel,  output reg[31:0] next\_pc  ); |

模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 功能描述 |
| PC4 | I | PC+4的值(对应于无跳转 直接执行下一句) |
| PC4D | I | D级PC+4 |
| I26 | I | 26位立即数 |
| MFRSD | I | 转发PC的MUX结果（jr jalr需要转发） |
| Zero | I | 比较两个数是否相等的结果 |
| Branch | I | 判断是不是beq类指令 |
| If\_j | I | 判断是不是j/jal指令 |
| PC\_sel[1:0] | I | PC的选择信号 |
| Next\_pc | O | 更新的pc值 |

功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 更新PC | 更新PC |

8.controller.v （分布式译码 实例化4个）

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| controller.v | input [5:0] op,  input [5:0] func,  output reg[2:0] ALUCtrl,  output reg[1:0] RegDst,  output reg ALUASrc,  output reg ALUBSrc,  output reg RegWrite,  output reg MemRead,  output reg MemWrite,  output reg [1:0] MemtoReg,  output reg [1:0]ExtOp,  output reg Branch,  output reg if\_j,  output reg [1:0]PCsel  ); |

模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 功能描述 |
| Op[5:0] | I | 6位opcode段 |
| Func[5:0] | I | 6位func段 |
| ALUCtrl[2:0] | O | ALU控制信号 |
| RegDst[1:0] | O | 写地址控制 选择RT,RD |
| ALUASrc | O | ALU第一操作数选择控制 |
| ALUBSrc | O | ALU第二操作数选择控制 |
| RegWrite | O | GRF 写入控制 |
| MemRead | O | DM读信号 |
| MemWrite | O | DM写信号 |
| MemToReg[1:0] | O | GRF写入数据的选择信号 |
| ExtOp | O | 高位扩展方式选择信号 |
| Branch | O | 判断是否为beq指令的信号 是则为1 |
| If\_j | O | 判断是不是jal/j指令 是则为1 |
| PC\_sel[1:0] | O | PC选择信号 |

功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 产生控制信号 | 产生控制信号 |

9.EX.v

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| EX.v | module EX(  input clk,  input reset,  input en,  input [31:0] IR\_D,  input [31:0] PC\_D,  input [31:0] PC4\_D,  input [31:0] PC8\_D,  input [31:0] RF\_RD1,  input [31:0] RF\_RD2,  input [31:0] EXT,  output reg[31:0] IR\_E,  output reg[31:0] PC\_E,  output reg[31:0] PC4\_E,  output reg[31:0] PC8\_E,  output reg[31:0] RS\_E,  output reg[31:0] RT\_E,  output reg[31:0] EXT\_E  ); |

功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | ID/EX流水线寄存器 | 保存PC,IR等信号的值 |

10.alu.v

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| alu.v | module alu(  input [31:0] A,  input [31:0] B,  input [2:0] ALUCtrl,  output reg[31:0] Result  ); |

模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 功能描述 |
| A[31:0] | I | 32 位输入数据 1 |
| B[31:0] | I | 32 位输入数据 2 |
| ALUCtrl[2:0] | I | 控制信号  000：与  001：或  010：加  011：减  100：移位 |
| Result[31:0] | O | 32 位数据输出 |

功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 与 | A&B |
| 2 | 或 | A|B |
| 3 | 加 | A+B |
| 4 | 减 | A-B |
| 5 | 移位 | B<<A |

11.MEM.v

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| MEM.v | module MEM(  input clk,  input reset,  input en,  input [31:0] IR\_E,  input [31:0] PC\_E,  input [31:0] PC4\_E,  input [31:0] PC8\_E,  input [31:0] ALU,  input [31:0] RT\_E,  output reg[31:0] IR\_M,  output reg[31:0] PC\_M,  output reg[31:0] PC4\_M,  output reg[31:0] PC8\_M,  output reg[31:0] AO\_M,  output reg[31:0] RT\_M  ); |

功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | EX/MEM流水线寄存器 | 保存PC,IR等信号的值 |

12.dm.v

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| dm.v | module dm(  input clk,  input reset,  input MemWrite,  input MemRead,  input [31:0] MemAddr,  input [31:0] WD,  input [31:0] PC,  output [31:0] RD  ); |

模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 功能描述 |
| Clk | I | 时钟信号 |
| Reset | I | 复位信号  1：复位  0：无效 |
| MemWrite | I | 读写控制信号  1：写操作 |
| MemRead | I | 读写控制信号  1：读操作 |
| MemAddr[31:0] | I | 操作寄存器地址 |
| WD[31:0] | I | 输入（写入内存）的32位数据 |
| PC[31:0] | I | 当前PC |
| RD[31:0] | O | 32 位数据输出 |

功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | 当复位信号有效时，所有数据被设置为 0x00000000 |
| 2 | 读 | 根据输入的寄存器地址读出数据 |
| 3 | 写 | 根据输入的地址，把输入的数据写入 |

13.WB.v

模块接口

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| WB.v | module WB(  input clk,  input reset,  input en,  input [31:0] IR\_M,  input [31:0] PC\_M,  input [31:0] PC4\_M,  input [31:0] PC8\_M,  input [31:0] AO\_M,  input [31:0] DM,  output reg[31:0] IR\_W,  output reg[31:0] PC\_W,  output reg[31:0] PC4\_W,  output reg[31:0] PC8\_W,  output reg[31:0] AO\_W,  output reg[31:0] DR\_W  ); |

功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | MEM/WB流水线寄存器 | 保存PC,IR等信号的值 |

14.mux.v

模块接口

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| mux.v | module mux(  input [31:0] EXT\_E,  input [31:0] IR\_E,  input [31:0] IR\_W,  input [31:0] DR\_W,  input [31:0] AO\_W,  input [31:0] PC8\_W,  input [31:0] MFRSE,  input [31:0] MFRTE,  input ALUasel,  input ALUbsel,  input [1:0] RegDst,  input [1:0] MemtoReg,  output reg[31:0] ALU\_A,  output reg[31:0] ALU\_B,  output reg[4:0] MUX\_A3,  output reg[31:0] MUX\_WD  ); |

功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 多路选择器 | 各级多路选择器  ALU\_A,ALU\_B,MUX\_WD,MUX\_A3 |

15.forward\_mux.v

模块接口

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| Forward\_mux.v | module forward\_mux(  input [31:0] RS\_E,  input [31:0] RT\_E,  input [31:0] RT\_M,  input [31:0] WD,  input [31:0] AO\_M,  input [31:0] PC8\_E,  input [31:0] PC8\_M,  input [31:0] PC8\_W,  input [31:0] RF\_RD1,  input [31:0] RF\_RD2,  input [2:0] ForwardRSD,  input [2:0] ForwardRTD,  input [2:0] ForwardRSE,  input [2:0] ForwardRTE,  input [2:0] ForwardRTM,  output reg[31:0] MFRSD,  output reg[31:0] MFRTD,  output reg[31:0] MFRSE,  output reg[31:0] MFRTE,  output reg[31:0] MFRTM  ); |

功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 各级转发MUX | 转发信号的选择  MFRSD,MFRTD,MFRSE,MFRTE,MFRTM |

16. hazardUnit.v

模块接口

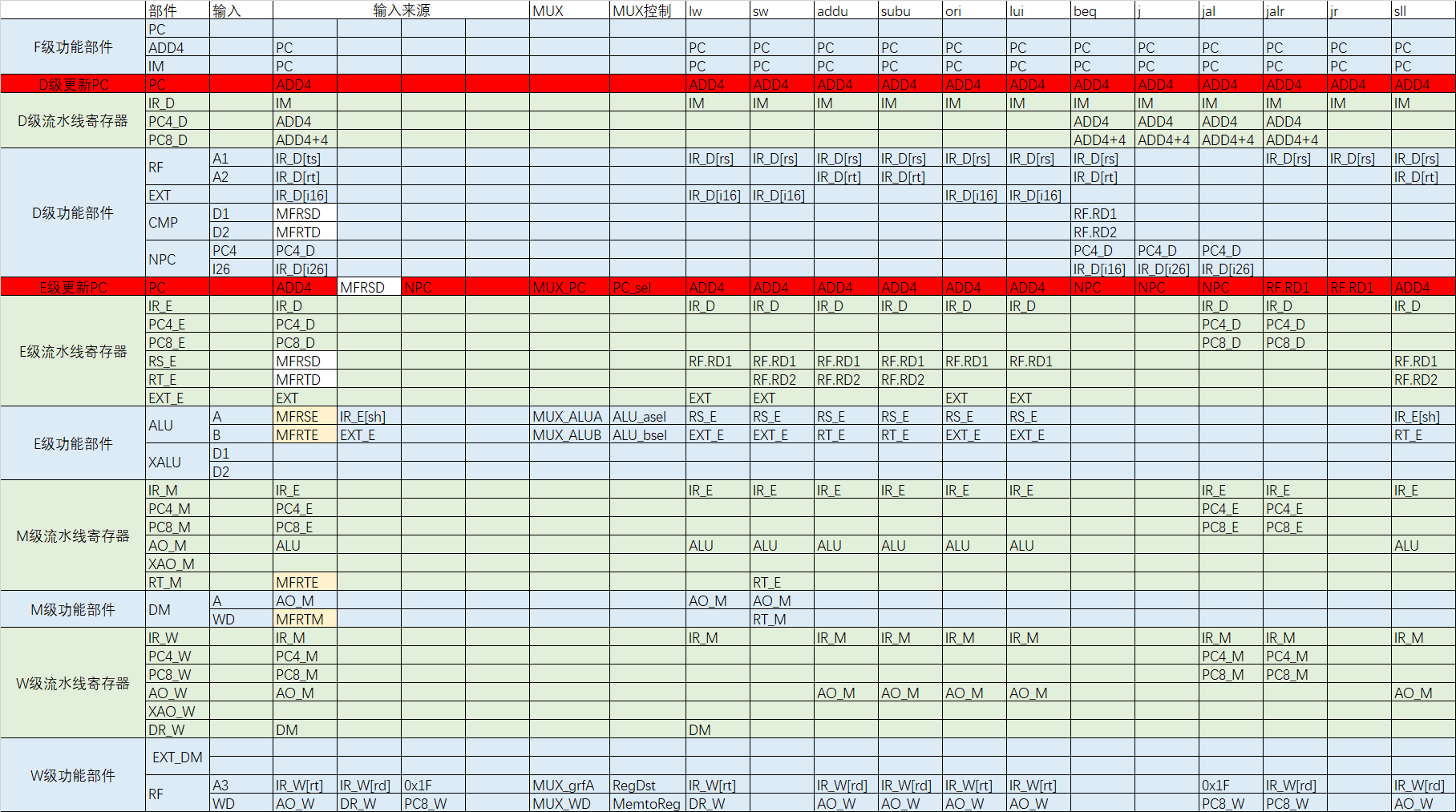
|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| hazardUnit.v | module hazardUnit(  input [31:0] IR\_D,  input [31:0] IR\_E,  input [31:0] IR\_M,  input [31:0] IR\_W,  output IR\_D\_en,  output IR\_E\_clr,  output PC\_en,  output [2:0]ForwardRSD,  output [2:0]ForwardRTD,  output [2:0]ForwardRSE,  output [2:0]ForwardRTE,  output [2:0]ForwardRTM  ); |

功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 冒险控制单元 | 产生转发和暂停的控制信号 |

三．控制器设计

数据通路如下



由此可见需要以下几个MUX多路选择器

1.GRF的WA端选择Rd,Rt需要一个MUX，控制信号RegDst[1:0]

2.GRF的WD输入端，有三种选择：RF.RD2，ALU的输出，lui指令直接对imm16后边补16位0，需要2选4MUX,选择信号MemToReg[1:0]

3.扩展方式的选择（符号扩展，0扩展）选择信号EXTOp[1:0]

4. ALU的A端两种选择，RF.RD1或IR\_E[sh]的输出，选择信号ALUASrc

5.ALU的B端两种选择，RF.RD2或EXT的输出，选择信号ALUBSrc

6.j/jal指令 跳转地址的选择 if\_j

7.PC的选择信号 PCsel[1:0]

8.beq类指令 跳转地址的选择 Branch

除了上述Branch, ALUASrc, ALUBSrc, EXTOp[1:0], MemToReg[1:0], RegDst[1:0] ,if\_j, PC\_sel[1:0] 还有三个读写控制信号，RegWrite是GRF写入信号，

MemRead, MemWrite是DM读写信号，ALUCtrl[2:0]是ALU控制信号，所以控制器Controller需要设计这12个控制信号。

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 功能描述 |
| Op[5:0] | I | 6位opcode段 |
| Func[5:0] | I | 6位func段 |
| ALUCtrl[2:0] | O | ALU控制信号 |
| RegDst[1:0] | O | 写地址控制 选择RT,RD |
| ALUASrc | O | ALU第一操作数选择控制 |
| ALUBSrc | O | ALU第二操作数选择控制 |
| RegWrite | O | GRF 写入控制 |
| MemRead | O | DM读信号 |
| MemWrite | O | DM写信号 |
| MemToReg[1:0] | O | GRF写入数据的选择信号 |
| ExtOp | O | 高位扩展方式选择信号 |
| Branch | O | 判断是否为beq指令的信号 是则为1 |
| If\_j | O | 判断是不是jal/j指令 是则为1 |
| PC\_sel[1:0] | O | PC选择信号 |

画出如下表格

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| name | lw | sw | beq | lui | ori | nop | jal | j | addu | subu | jr | sll |
| Op5 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Op4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Op3 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Op2 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Op1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| Op0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| Func5 |  |  |  |  |  |  |  |  | 1 | 1 | 0 | 0 |
| Func4 |  |  |  |  |  |  |  |  | 0 | 0 | 0 | 0 |
| Func3 |  |  |  |  |  |  |  |  | 0 | 0 | 1 | 0 |
| Func2 |  |  |  |  |  |  |  |  | 0 | 0 | 0 | 0 |
| Func1 |  |  |  |  |  |  |  |  | 0 | 1 | 0 | 0 |
| Func0 |  |  |  |  |  |  |  |  | 1 | 1 | 0 | 0 |
| RegDst[1:0] | 00 | 00 | 00 | 00 | 00 | xx | 10 | 00 | 01 | 01 | 01 | 01 |
| ALUASrc | 0 | 0 | 0 | 0 | 0 | x | 0 | 0 | 0 | 0 | 0 | 1 |
| ALUBSrc | 1 | 1 | 0 | 1 | 1 | x | 0 | 0 | 0 | 0 | 0 | 0 |
| RegWrite | 1 | 0 | 0 | 1 | 1 | x | 1 | 0 | 1 | 1 | 1 | 1 |
| MemRead | 1 | 0 | 0 | 0 | 0 | x | 0 | 0 | 0 | 0 | 0 | 0 |
| MemWrite | 0 | 1 | 0 | 0 | 0 | x | 0 | 0 | 0 | 0 | 0 | 0 |
| MemToReg[1:0] | 01 | 00 | 00 | 00 | 00 | xx | 10 | 00 | 00 | 00 | 00 | 00 |
| EXTOp[1:0] | 00 | 00 | 00 | 10 | 01 | xx | 00 | 00 | 00 | 00 | 00 | 00 |
| Branch | 0 | 0 | 1 | 0 | 0 | x | 0 | 0 | 0 | 0 | 0 | 0 |
| ALUCtrl[2:0] | 010 | 010 | 111 | 010 | 001 | xxx | 111 | 111 | 010 | 011 | 111 | 100 |
| If\_j | 0 | 0 | 0 | 0 | 0 | x | 1 | 1 | 0 | 0 | 0 | 0 |
| PC\_sel[1:0] | 00 | 00 | 10 | 00 | 00 | 00 | 10 | 10 | 00 | 00 | 01 | 00 |

分布式译码 实例化四级控制器（译码器）

controller my\_controllerD(.op(IR\_D[`op]),.func(IR\_D[`func]),.ExtOp(EXTop),.Branch(Branch),.if\_j(if\_j),.PCsel(PC\_sel));

controller my\_controllerE(.op(IR\_E[`op]),.func(IR\_E[`func]),.ALUCtrl(ALUCtrl),.ALUASrc(ALUASrc),.ALUBSrc(ALUBSrc));

controller my\_controllerM(.op(IR\_M[`op]),.func(IR\_M[`func]),.MemRead(MemRead),.MemWrite(MemWrite));

controller my\_controllerW(.op(IR\_W[`op]),.func(IR\_W[`func]),.RegDst(RegDst),.RegWrite(RegWrite),.MemtoReg(MemtoReg));

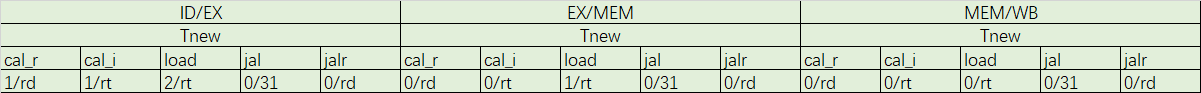
四．冒险处理单元设计

**需求时间——供给时间模型**。

Tuse



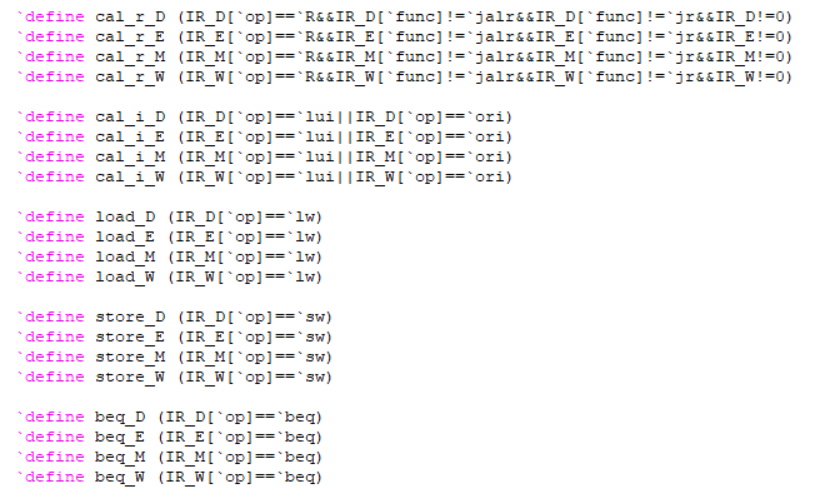
Tnew

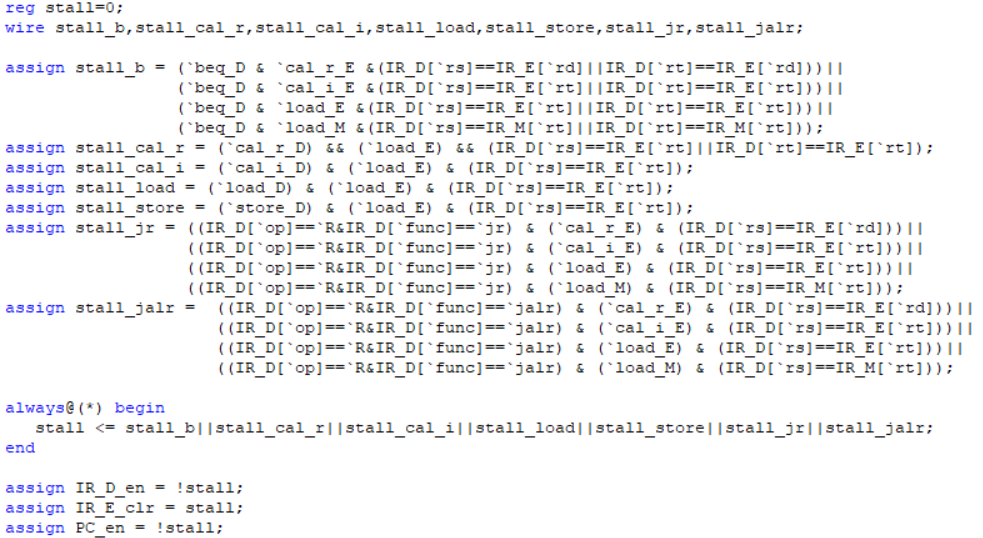


暂停

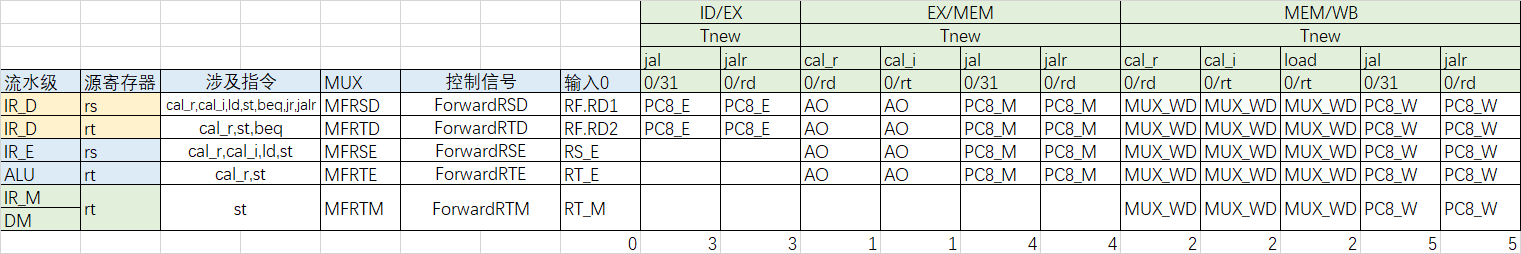
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| IF/ID当前指令 | | | ID/EX | | | EX/MEM |
| 指令类型 | 源寄存器 | Tuse | Tnew | | | Tnew |
| cal\_r | cal\_i | load | load |
| 1/rd | 1/rt | 2/rt | 1/rt |
| beq | rs/rt | 0 | 暂停 | 暂停 | 暂停 | 暂停 |
| cal\_r | rs/rt | 1 |  |  | 暂停 |  |
| cal\_i | rs | 1 |  |  | 暂停 |  |
| load | rs | 1 |  |  | 暂停 |  |
| store | rs | 1 |  |  | 暂停 |  |
| store | rt | 2 |  |  |  |  |
| jr | rs | 0 | 暂停 | 暂停 | 暂停 | 暂停 |
| jalr | rs | 0 | 暂停 | 暂停 | 暂停 | 暂停 |

由此可以写出各种控制信号的表达式如下

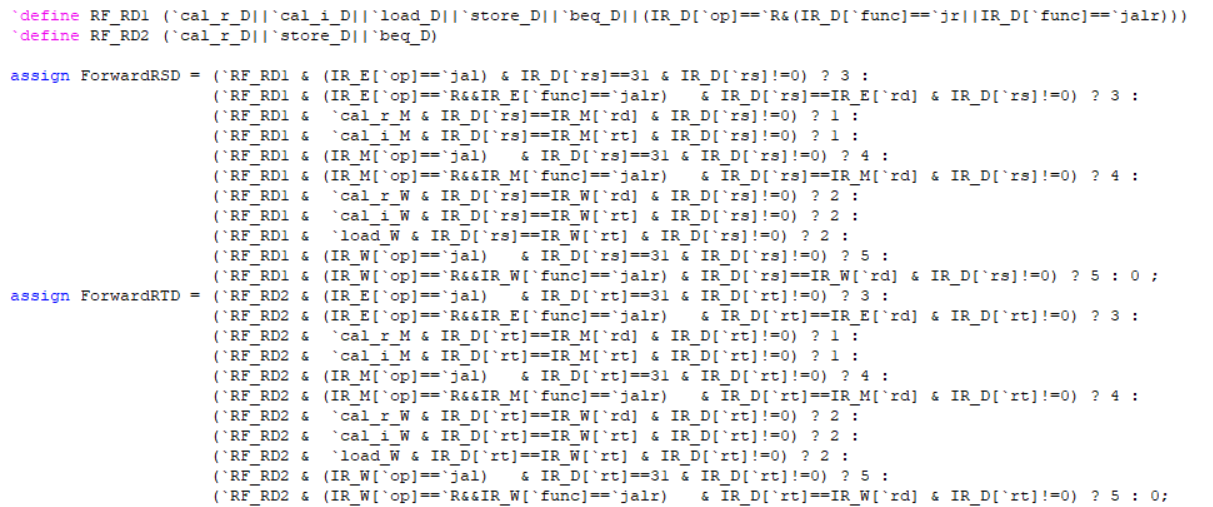


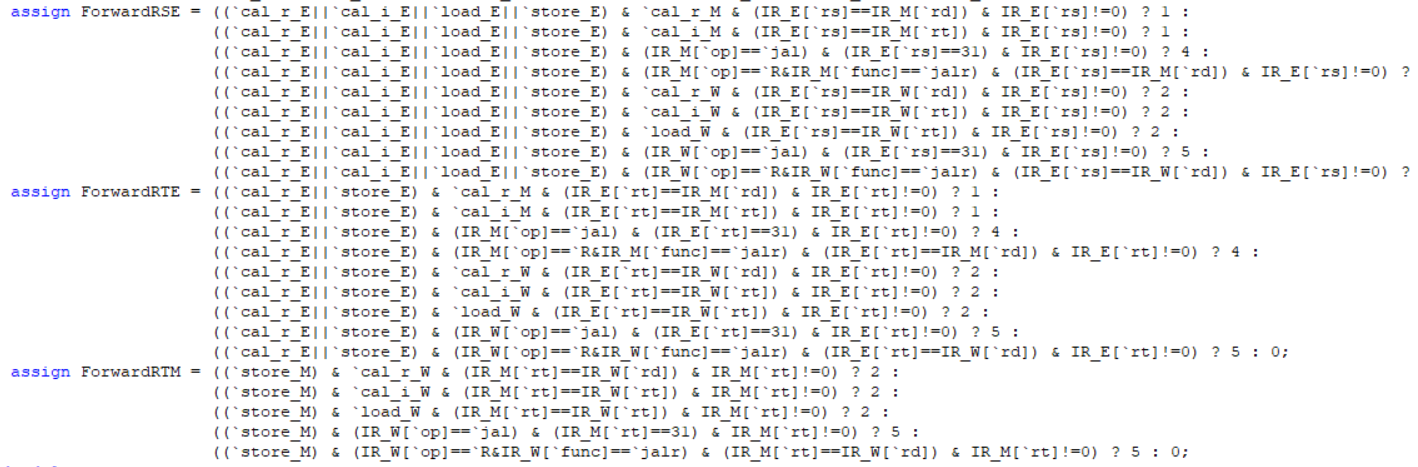


转发

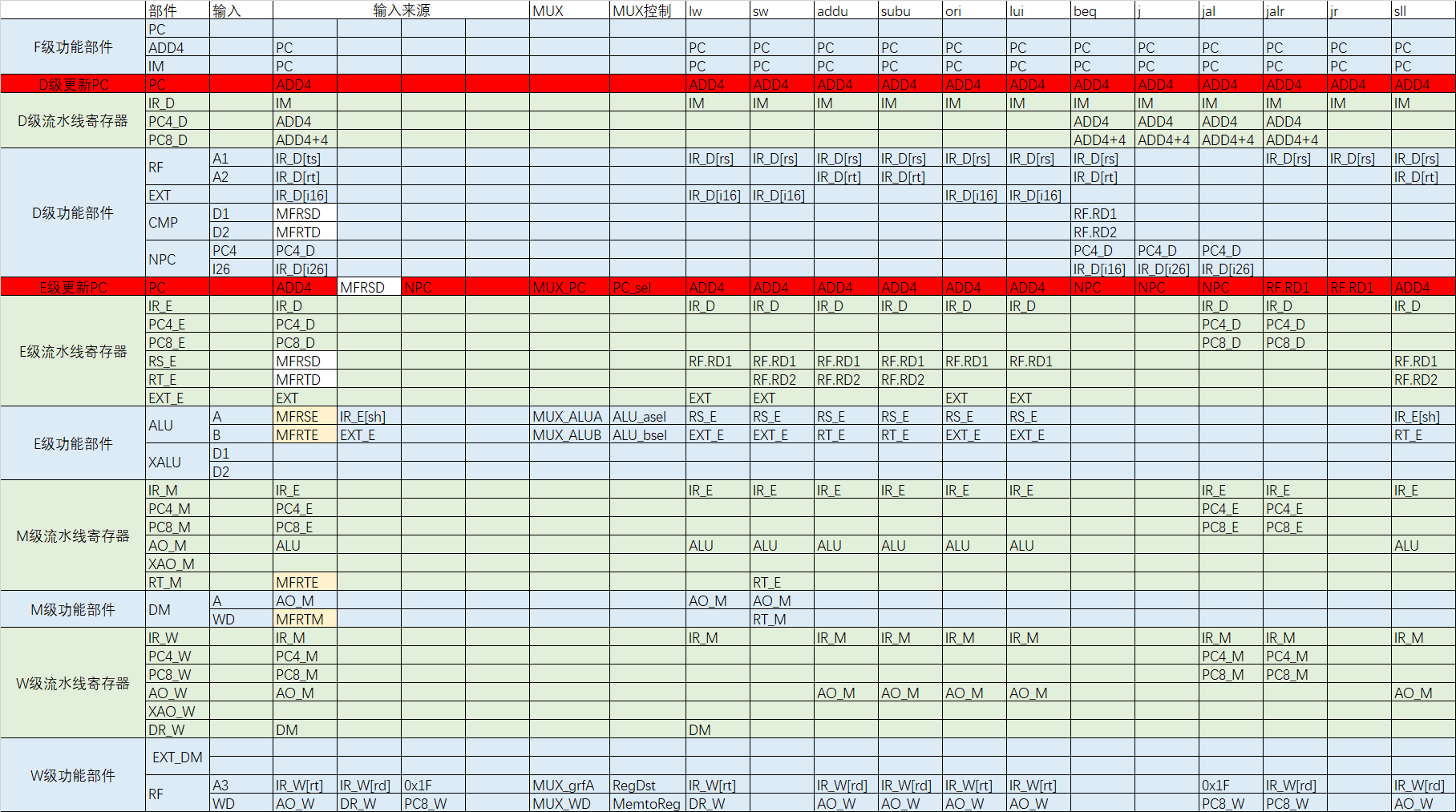


由此可以写出各种控制信号的表达式如下





更新后的数据通路 加入了转发MUX



Forward\_mux代码如下

always@(\*) begin

case(ForwardRSD)

3'b000 : MFRSD <= RF\_RD1;

3'b001 : MFRSD <= AO\_M;

3'b010 : MFRSD <= WD;

3'b011 : MFRSD <= PC8\_E;

3'b100 : MFRSD <= PC8\_M;

3'b101 : MFRSD <= PC8\_W;

default : MFRSD <= 0;

endcase

case(ForwardRTD)

0: MFRTD <= RF\_RD2;

1: MFRTD <= AO\_M;

2: MFRTD <= WD;

3: MFRTD <= PC8\_E;

4: MFRTD <= PC8\_M;

5: MFRTD <= PC8\_W;

default: MFRTD <= 0;

endcase

case(ForwardRSE)

0:MFRSE <= RS\_E;

1:MFRSE <= AO\_M;

2:MFRSE <= WD;

3:MFRSE <= 0;

4:MFRSE <= PC8\_M;

5:MFRSE <= PC8\_W;

default:MFRSE <= 0;

endcase

case(ForwardRTE)

0:MFRTE <= RT\_E;

1:MFRTE <= AO\_M;

2:MFRTE <= WD;

3:MFRTE <= 0;

4:MFRTE <= PC8\_M;

5:MFRTE <= PC8\_W;

default:MFRTE <= 0;

endcase

case(ForwardRTM)

0:MFRTM <= RT\_M;

1:MFRTM <= 0;

2:MFRTM <= WD;

3:MFRTM <= 0;

4:MFRTM <= 0;

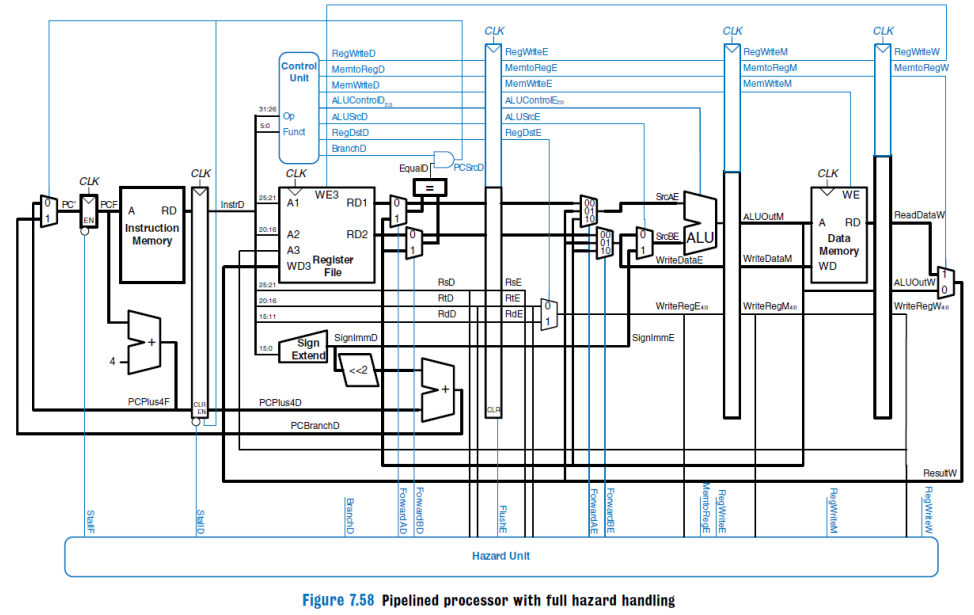
5:MFRTM <= PC8\_W;

default:MFRTM <= 0;

endcase

end

五．主程序，数据通路设计，tb



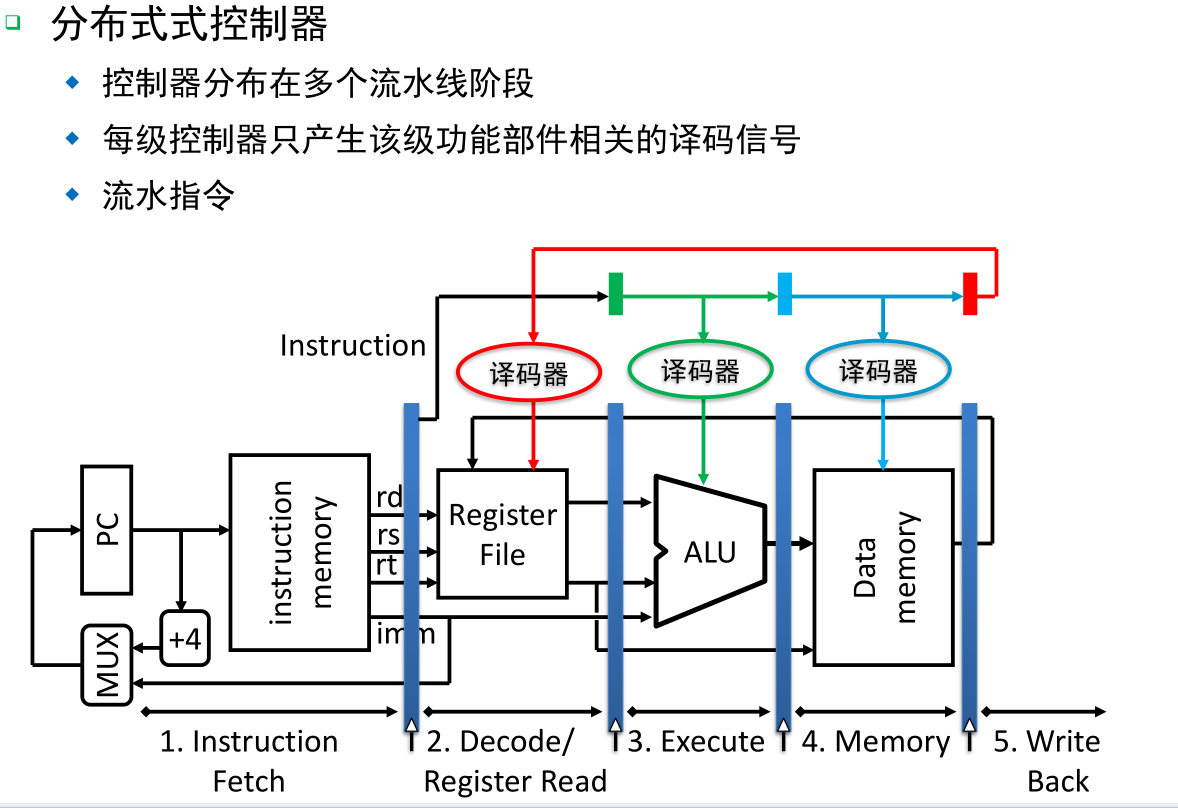
数据通路主要采用如上架构 区别是分布式译码

1. 流水线的设计以追求性能为第一目标，因此必须尽最大可能**支持转发**以解决数据冒险。这一点在本project的最终成绩中所占比重较大，课上测试时会通过测试程序所跑的**总周期数**进行判定，望大家慎重对待。

2. 对于 b 类和 j 类指令， 流水线设计必须**支持延迟槽**，因此设计需要注意使用 **PC+8**。

3. 为了解决数据冒险而设计的转发数据来源必须是**某级流水线寄存器**，**不允许**对功能部件的输出直接进行转发。

4.分布式译码



需要以下几个MUX多路选择器

1.GRF的WA端选择Rd,Rt需要一个MUX，控制信号RegDst[1:0]

2.GRF的WD输入端，有三种选择：RF.RD2，ALU的输出，lui指令直接对imm16后边补16位0，需要2选4MUX,选择信号MemToReg[1:0]

3. ALU的A端两种选择，RF.RD1或IR\_E[sh]的输出，选择信号ALUASrc

4.ALU的B端两种选择，RF.RD2或EXT的输出，选择信号ALUBSrc

1.mux.v

模块接口

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| mux.v | module mux(  input [31:0] EXT\_E,  input [31:0] IR\_E,  input [31:0] IR\_W,  input [31:0] DR\_W,  input [31:0] AO\_W,  input [31:0] PC8\_W,  input [31:0] MFRSE,  input [31:0] MFRTE,  input ALUasel,  input ALUbsel,  input [1:0] RegDst,  input [1:0] MemtoReg,  output reg[31:0] ALU\_A,  output reg[31:0] ALU\_B,  output reg[4:0] MUX\_A3,  output reg[31:0] MUX\_WD  ); |

always@(\*) begin

case(ALUasel)

1'b0: ALU\_A<=MFRSE;

1'b1: ALU\_A<={27'b0,IR\_E[10:6]};

endcase

case(ALUbsel)

1'b0: ALU\_B<=MFRTE;

1'b1: ALU\_B<=EXT\_E;

endcase

case(RegDst)

2'b00: MUX\_A3<=IR\_W[20:16];

2'b01: MUX\_A3<=IR\_W[15:11];

2'b10: MUX\_A3<=32'h1f;

2'b11: MUX\_A3<=0;

endcase

case(MemtoReg)

2'b00: MUX\_WD<=AO\_W;

2'b01: MUX\_WD<=DR\_W;

2'b10: MUX\_WD<=PC8\_W;

2'b11: MUX\_WD<=0;

endcase

end

2.mips.v

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| mips.v | module mips(  input clk,  input reset  ); |

pc my\_pc(clk,reset,PC\_en,next\_pc,PC);

im my\_im(PC,Instr);

ID my\_ID(clk,ID\_reset||reset,ID\_en,Instr,PC,IR\_D,PC\_D,PC4\_D,PC8\_D);

controller my\_controllerD(.op(IR\_D[`op]),.func(IR\_D[`func]),.ExtOp(EXTop),.Branch(Branch),.if\_j(if\_j),.PCsel(PC\_sel));

grf my\_grf(clk,reset,RegWrite,IR\_D[`rs],IR\_D[`rt],MUX\_A3,MUX\_WD,PC\_W,RF\_RD1,RF\_RD2);

cmp my\_cmp(MFRSD,MFRTD,Zero);

ext my\_ext(IR\_D[`imm16],EXTop,EXT\_out);

npc my\_npc(PC4,PC4\_D,IR\_D[`imm26],MFRSD,Zero,Branch,if\_j,PC\_sel,next\_pc);

EX my\_EX(clk,EX\_reset||reset,EX\_en,IR\_D,PC\_D,PC4\_D,PC8\_D,MFRSD,MFRTD,EXT\_out,IR\_E,PC\_E,PC4\_E,PC8\_E,RS\_E,RT\_E,EXT\_E);

controller my\_controllerE(.op(IR\_E[`op]),.func(IR\_E[`func]),.ALUCtrl(ALUCtrl),.ALUASrc(ALUASrc),.ALUBSrc(ALUBSrc));

alu my\_alu(ALU\_A,ALU\_B,ALUCtrl,ALU\_out);

MEM my\_MEM(clk,MEM\_reset||reset,MEM\_en,IR\_E,PC\_E,PC4\_E,PC8\_E,ALU\_out,MFRTE,IR\_M,PC\_M,PC4\_M,PC8\_M,AO\_M,RT\_M);

controller my\_controllerM(.op(IR\_M[`op]),.func(IR\_M[`func]),.MemRead(MemRead),.MemWrite(MemWrite));

dm my\_dm(clk,reset,MemWrite,MemRead,AO\_M,MFRTM,PC\_M,DM\_out);

WB my\_WB(clk,WB\_reset||reset,WB\_en,IR\_M,PC\_M,PC4\_M,PC8\_M,AO\_M,DM\_out,IR\_W,PC\_W,PC4\_W,PC8\_W,AO\_W,DR\_W);

controller my\_controllerW(.op(IR\_W[`op]),.func(IR\_W[`func]),.RegDst(RegDst),.RegWrite(RegWrite),.MemtoReg(MemtoReg));

mux my\_mux(EXT\_E,IR\_E,IR\_W,DR\_W,AO\_W,PC8\_W,MFRSE,MFRTE,ALUASrc,ALUBSrc,RegDst,MemtoReg,ALU\_A,ALU\_B,MUX\_A3,MUX\_WD);

forward\_mux my\_forward(RS\_E,RT\_E,RT\_M,MUX\_WD,AO\_M,PC8\_E,PC8\_M,PC8\_W,RF\_RD1,RF\_RD2,ForwardRSD,ForwardRTD,ForwardRSE,ForwardRTE,ForwardRTM,MFRSD,MFRTD,MFRSE,MFRTE,MFRTM);

hazardUnit my\_hazard(IR\_D,IR\_E,IR\_M,IR\_W,ID\_en,EX\_reset,PC\_en,ForwardRSD,ForwardRTD,ForwardRSE,ForwardRTE,ForwardRTM);

3.tb

module test;

// Inputs

reg clk;

reg reset;

// Instantiate the Unit Under Test (UUT)

mips uut (

.clk(clk),

.reset(reset)

);

initial begin

clk = 0;

reset = 1;

#12 reset = 0;

end

always #10 clk = ~clk;

endmodule

五．测试程序

(1)转发机制覆盖测试

**测试目录：**

一．D级rs

1.D级rs与E级jal

2.D级Rs与M级cal\_r

3.D级Rs与M级cal\_i

4.D级Rs与M级jal

5.D级Rs与W级cal\_r

6.D级Rs与W级cal\_i

7.D级Rs与W级load rt

8.D级Rs与W级jal

每一项又分为：cal\_r,cal\_i,ld,st,beq,jr,jalr

二．D级Rt

1.D级rt与E级jal

2.D级Rt与M级cal\_r

3.D级Rt与M级cal\_i

4.D级Rt与M级jal

5.D级Rt与W级cal\_r

6.D级Rt与W级cal\_i

7.D级Rt与W级load rt

8.D级Rt与W级jal

每一项又分为：cal\_r,st,beq

三．E级Rs

1.E级Rs与M级cal\_r

2.E级Rs与M级cal\_i

3.E级Rs与M级jal

4.E级Rs与W级cal\_r

5.E级Rs与W级cal\_i

6.E级Rs与W级load

7.E级Rs与W级jal

每一项又分为：cal\_r,cal\_i,ld,st

四．E级Rt

1.E级Rt与M级cal\_r

2.E级Rt与M级cal\_i

3.E级Rt与M级jal

4.E级Rt与W级cal\_r

5.E级Rt与W级cal\_i

6.E级Rt与W级load

7.E级Rt与W级jal

每一项又分为：cal\_r,st

五．M级Rt

1.M级Rt与W级cal\_r

2.M级Rt与W级cal\_i

3.M级Rt与W级load

4.M级Rt与W级jal

每一项又分为：st

一．D级rs

1.D级rs与E级jal

(1)Rs----cal\_r

ori $t0,11

jal eee

addu $t0,$ra,$0

eee:

110@00003000: $ 8 <= 0000000b

130@00003004: $31 <= 0000300c

150@00003008: $ 8 <= 0000300c

(2)Rs---cal\_i

ori $t0,11

jal eee

ori $t0,$ra,11

eee:

110@00003000: $ 8 <= 0000000b

130@00003004: $31 <= 0000300c

150@00003008: $ 8 <= 0000300f

2.D级Rs与M级cal\_r

(1)Rs----cal\_r

ori $t0,11

addu $t1,$t2,$t0

nop

addu $t2,$t1,$t0

110@00003000: $ 8 <= 0000000b

130@00003004: $ 9 <= 0000000b

170@0000300c: $10 <= 00000016

(2)Rs---cal\_i

ori $t0,11

addu $t1,$t2,$t0

nop

ori $t0,$t1,1

110@00003000: $ 8 <= 0000000b

130@00003004: $ 9 <= 0000000b

170@0000300c: $ 8 <= 0000000b

(3)Rs---load

addu $a0,$0,$0

nop

lw $t0,0($a0)

110@00003000: $ 4 <= 00000000

150@00003008: $ 8 <= 00000000

(4)Rs---store

ori $t0,111

addu $a0,$0,$0

nop

sw $t0,0($a0)

110@00003000: $ 8 <= 0000006f

130@00003004: $ 4 <= 00000000

150@0000300c: \*00000000 <= 0000006f

(5)Rs---beq

ori $t1,1

addu $t2,$t1,$0

nop

beq $t2,$t1,out

nop

out:

nop

110@00003000: $ 9 <= 00000001

130@00003004: $10 <= 00000001

(6)Rs---jr

ori $t1,0x00003014

addu $t2,$t1,$0

nop

jr $t2

nop

out:

nop

110@00003000: $ 9 <= 00003014

130@00003004: $10 <= 00003014

3.D级Rs与M级cal\_i

(1)Rs----cal\_r

ori $t0,11

ori $t1,$t2,0

nop

addu $t2,$t1,$t0

110@00003000: $ 8 <= 0000000b

130@00003004: $ 9 <= 00000000

170@0000300c: $10 <= 0000000b

(2)Rs---cal\_i

ori $t0,11

ori $t1,$t2,0

nop

ori $t0,$t1,1

110@00003000: $ 8 <= 0000000b

130@00003004: $ 9 <= 00000000

170@0000300c: $ 8 <= 00000001

190@00003010: $ 8 <= 0000000b

210@00003014: $ 9 <= 00000000

250@0000301c: $ 8 <= 00000001

(3)Rs---load

ori $a0,$0,0

nop

lw $t0,0($a0)

110@00003000: $ 4 <= 00000000

150@00003008: $ 8 <= 00000000

(4)Rs---store

ori $t0,111

ori $a0,$0,0

nop

sw $t0,0($a0)

110@00003000: $ 8 <= 0000006f

130@00003004: $ 4 <= 00000000

150@0000300c: \*00000000 <= 0000006f

(5)Rs---beq

ori $t1,1

ori $t2,$t1,0

nop

beq $t2,$t1,out

nop

out:

nop

110@00003000: $ 9 <= 00000001

130@00003004: $10 <= 00000001

(6)Rs---jr

ori $t1,0x00003014

ori $t2,$t1,0

nop

jr $t2

nop

out:

nop

110@00003000: $ 9 <= 00003014

130@00003004: $10 <= 00003014

4.D级Rs与M级jal

(1)Rs----cal\_r

ori $t0,11

jal eee

nop

addu $t0,$ra,$0

eee:

(2)Rs---cal\_i

ori $t0,11

jal eee

nop

ori $t0,$ra,11

eee:

(3)Rs---load

ori $a0,$0,0x00003000

jal eee

subu $ra,$ra,$a0

eee:

lw $t1,0($ra)

(4)Rs---store

ori $t1,1

ori $a0,$0,0x00003000

jal eee

subu $ra,$ra,$a0

eee:

sw $t1,0($ra)

5.D级Rs与W级cal\_r

(1)Rs----cal\_r

ori $t0,11

addu $t1,$t2,$t0

nop

nop

addu $t2,$t1,$t0

(2)Rs---cal\_i

ori $t0,11

addu $t1,$t2,$t0

nop

nop

ori $t0,$t1,1

(3)Rs---load

addu $a0,$0,$0

nop

nop

lw $t0,0($a0)

(4)Rs---store

ori $t0,111

addu $a0,$0,$0

nop

nop

sw $t0,0($a0)

(5)Rs---beq

ori $t1,1

addu $t2,$t1,$0

nop

nop

beq $t2,$t1,out

nop

out:

nop

(6)Rs---jr

ori $t1,0x00003014

addu $t2,$t1,$0

nop

nop

jr $t2

nop

out:

nop

6.D级Rs与W级cal\_i

(1)Rs----cal\_r

ori $t0,11

ori $t1,$t2,0

nop

nop

addu $t2,$t1,$t0

(2)Rs---cal\_i

ori $t0,11

ori $t1,$t2,0

nop

nop

ori $t0,$t1,1

(3)Rs---load

ori $a0,$0,0

nop

nop

lw $t0,0($a0)

(4)Rs---store

ori $t0,111

ori $a0,$0,0

nop

nop

sw $t0,0($a0)

(5)Rs---beq

ori $t1,1

ori $t2,$t1,0

nop

nop

beq $t2,$t1,out

nop

out:

nop

(6)Rs---jr

ori $t1,0x00003014

ori $t2,$t1,0

nop

nop

jr $t2

nop

out:

nop

7.D级Rs与W级load rt

(1)Rs----cal\_r

ori $t1,1

ori $t0,0x00000000

lw $t1,0($t0)

nop

nop

addu $t2,$t1,$t0

110@00003000: $ 9 <= 00000001

130@00003004: $ 8 <= 00000000

150@00003008: $ 9 <= 00000000

210@00003014: $10 <= 00000000

(2)Rs---cal\_i

ori $t1,1

ori $t0,0x00000000

lw $t1,0($t0)

nop

nop

ori $t0,$t1,1

110@00003000: $ 9 <= 00000001

130@00003004: $ 8 <= 00000000

150@00003008: $ 9 <= 00000000

210@00003014: $ 8 <= 00000001

(3)Rs---load

ori $t0,0x00000000

lw $t1,0($t0)

nop

nop

lw $t0,0($t1)

110@00003000: $ 8 <= 00000000

130@00003004: $ 9 <= 00000000

190@00003010: $ 8 <= 00000000

(4)Rs---store

ori $t0,0x00000000

lw $t1,0($t0)

nop

nop

sw $t0,0($t1)

110@00003000: $ 8 <= 00000000

130@00003004: $ 9 <= 00000000

170@00003010: \*00000000 <= 00000000

(5)Rs---beq

ori $t1,1

ori $t0,0x00000000

lw $t1,0($t0)

ori $t2,$t1,0

nop

beq $t1,$t2,out

nop

out:

nop

110@00003000: $ 9 <= 00000001

130@00003004: $ 8 <= 00000000

150@00003008: $ 9 <= 00000000

190@0000300c: $10 <= 00000000

(6)Rs---jr

ori $t0,0x00000000

ori $t2,$0,0x00003020

sw $t2,0($t0)

lw $t1,0($t0)

nop

nop

jr $t1

nop

out:

nop

110@00003000: $ 8 <= 00000000

130@00003004: $10 <= 00003020

130@00003008: \*00000000 <= 00003020

170@0000300c: $ 9 <= 00003020

8.D级Rs与W级jal

(1)Rs----cal\_r

ori $t0,11

jal eee

nop

nop

addu $t0,$ra,$0

eee:

(2)Rs---cal\_i

ori $t0,11

jal eee

nop

nop

ori $t0,$ra,11

eee:

(3)Rs---load

ori $a0,$0,0x00003000

jal eee

subu $ra,$ra,$a0

eee:

nop

lw $t1,0($ra)

(4)Rs---store

ori $t1,1

ori $a0,$0,0x00003000

jal eee

subu $ra,$ra,$a0

eee:

nop

sw $t1,0($ra)

二．D级Rt

1.D级rt与E级jal

(1)Rt----cal\_r

ori $t0,11

jal eee

addu $t0,$0,$ra

eee:

110@00003000: $ 8 <= 0000000b

130@00003004: $31 <= 0000300c

150@00003008: $ 8 <= 0000300c

(2)Rt---store

2.D级Rt与M级cal\_r

(1)Rt----cal\_r

ori $t0,11

addu $t1,$t2,$t0

nop

addu $t2,$t0,$t1

(2)Rt---store

ori $t0,111

addu $a0,$t0,$0

nop

sw $a0,0($0)

(2)Rt---beq

ori $t1,1

addu $t2,$t1,$0

nop

beq $t1,$t2,out

nop

out:

nop

3.D级Rt与M级cal\_i

(1)Rt----cal\_r

ori $t0,11

ori $t1,$t2,0

nop

addu $t2,$t0,$t1

(2)Rt---store

ori $a0,$0,111

nop

sw $a0,0($0)

(3)Rt---beq

ori $t1,1

ori $t2,$t1,0

nop

beq $t1,$t2,out

nop

out:

nop

4.D级Rt与M级jal

(1)Rt----cal\_r

ori $t0,11

jal eee

nop

addu $t0,$0,$ra

eee:

(2)Rt---store

ori $t1,1

ori $a0,$0,0x00003000

jal eee

subu $ra,$ra,$a0

eee:

sw $ra,0($0)

5.D级Rt与W级cal\_r

(1)Rt----cal\_r

ori $t0,11

addu $t1,$t2,$t0

nop

nop

addu $t2,$t0,$t1

(2)Rt---store

ori $t0,111

addu $a0,$0,$0

nop

nop

sw $a0,0($0)

(3)Rt---beq

ori $t1,1

addu $t2,$t1,$0

nop

nop

beq $t1,$t2,out

nop

out:

nop

6.D级Rt与W级cal\_i

(1)Rt----cal\_r

ori $t0,11

ori $t1,$t2,0

nop

nop

addu $t2,$t0,$t1

(4)Rt---store

ori $t0,111

ori $a0,$0,0

nop

nop

sw $a0,0($0)

(5)Rt---beq

ori $t1,1

ori $t2,$t1,0

nop

nop

beq $t1,$t2,out

nop

out:

nop

7.D级Rt与W级load rt

(1)Rt----cal\_r

ori $t1,1

ori $t0,0x00000000

lw $t1,0($t0)

nop

nop

addu $t2,$t0,$t1

(2)Rt---store

ori $t0,0x00000000

lw $t1,0($t0)

nop

nop

sw $t1,0($0)

(3)Rs---beq

ori $t1,1

ori $t0,0x00000000

lw $t1,0($t0)

ori $t2,$t1,0

nop

beq $t2,$t1,out

nop

out:

nop

8.D级Rt与W级jal

(1)Rt----cal\_r

ori $t0,11

jal eee

nop

nop

addu $t0,$0,$ra

eee:

(2)Rt---store

ori $t1,1

ori $a0,$0,0x00003000

jal eee

subu $ra,$ra,$a0

eee:

nop

sw $ra,0($0)

三．E级Rs

1.E级Rs与M级cal\_r

(1)Rs----cal\_r

ori $t2,$0,111

addu $t1,$t2,$t3

subu $t4,$t1,$0

110@00003000: $10 <= 0000006f

130@00003004: $ 9 <= 0000006f

150@00003008: $12 <= 0000006f

(2)Rs----cal\_i

ori $t2,$0,111

subu $t4,$t2,$0

ori $t2,$t4,111

(3)Rs----load

addu $t1,$t2,$t3

lw $t2,0($t1)

110@00003000: $ 9 <= 00000000

130@00003004: $10 <= 00000000

(4)Rs----store

addu $t1,$t2,$t3

sw $t2,0($t1)

2.E级Rs与M级cal\_i

(1)Rs----cal\_r

ori $t2,$0,111

ori $t1,$t2,0

subu $t4,$t1,$0

(2)Rs----cal\_i

ori $t2,$0,111

ori $t4,$t2,0

(3)Rs----load

ori $t1,$t2,$t3

lw $t2,0($t1)

(4)Rs----store

ori $t1,$t2,$t3

sw $t2,0($t1)

3.E级Rs与M级jal

(1)Rs----cal\_r

jal eee

subu $t1,$ra,$t2

eee:

(2)Rs----cal\_i

jal eee

lui $ra,111

eee:

(3)Rs----load

jal eee

lw $0,0($ra)

eee:

(4)Rs----store

jal eee

sw $0,0($ra)

eee:

4.E级Rs与W级cal\_r

(1)Rs----cal\_r

ori $t2,$0,111

addu $t1,$t2,$t3

nop

subu $t4,$t1,$0

(2)Rs----cal\_i

ori $t2,$0,111

subu $t4,$t2,$0

nop

ori $t2,$t4,111

(3)Rs----load

addu $t1,$t2,$t3

nop

lw $t2,0($t1)

(4)Rs----store

addu $t1,$t2,$t3

nop

sw $t2,0($t1)

5.E级Rs与W级cal\_i

(1)Rs----cal\_r

ori $t2,$0,111

ori $t1,$t2,0

nop

subu $t4,$t1,$0

(2)Rs----cal\_i

ori $t2,$0,111

nop

ori $t4,$t2,0

(3)Rs----load

ori $t1,$t2,$t3

nop

lw $t2,0($t1)

(4)Rs----store

ori $t1,$t2,$t3

nop

sw $t2,0($t1)

6.E级Rs与W级load

(1)Rs----cal\_r

ori $t1,1

ori $t0,0x00000000

lw $t1,0($t0)

nop

addu $t2,$t1,$t0

(2)Rs---cal\_i

ori $t1,1

ori $t0,0x00000000

lw $t1,0($t0)

nop

ori $t0,$t1,1

(3)Rs---load

ori $t0,0x00000000

lw $t1,0($t0)

nop

lw $t0,0($t1)

(4)Rs---store

ori $t0,0x00000000

lw $t1,0($t0)

nop

sw $t0,0($t1)

7.E级Rs与W级jal

(1)Rs----cal\_r

jal eee

nop

eee:

subu $t1,$ra,$t2

(2)Rs----cal\_i

jal eee

nop

eee:

lui $ra,111

(3)Rs----load

ori $t1,0x00003000

jal eee

eee:

subu $ra,$ra,$t1

lw $0,0($ra)

(4)Rs----store

ori $t1,0x00003000

jal eee

eee:

subu $ra,$ra,$t1

sw $0,0($ra)

四．E级Rt

1.E级Rt与M级cal\_r

(1)Rt----cal\_r

ori $t2,$0,111

addu $t1,$t2,$t3

subu $t4,$0,$t1

110@00003000: $10 <= 0000006f

130@00003004: $ 9 <= 0000006f

150@00003008: $12 <= ffffff91

170@0000300c: $10 <= 0000006f

190@00003010: $ 9 <= 0000006f

210@00003014: $12 <= ffffff91

(2)Rt----store

addu $t1,$t2,$t3

sw $t1,0($t2)

110@00003000: $ 9 <= 00000000

110@00003004: \*00000000 <= 00000000

2.E级Rt与M级cal\_i

(1)Rt----cal\_r

ori $t2,$0,111

ori $t1,$t2,0

subu $t4,$0,$t1

110@00003000: $10 <= 0000006f

130@00003004: $ 9 <= 0000006f

150@00003008: $12 <= ffffff91

(2)Rt----store

ori $t1,$t2,100

sw $t1,0($t1)

110@00003000: $ 9 <= 00000064

110@00003004: \*00000064 <= 00000064

3.E级Rt与M级jal

(1)Rt----cal\_r

jal eee

subu $t1,$t2,$ra

eee:

110@00003000: $31 <= 00003008

130@00003004: $ 9 <= ffffcff8

(2)Rt----store

jal eee

sw $ra,0($0)

eee:

110@00003000: $31 <= 00003008

110@00003004: \*00000000 <= 00003008

4.E级Rt与W级cal\_r

(1)Rt----cal\_r

ori $t2,$0,111

addu $t1,$t2,$t3

nop

subu $t4,$0,$t1

110@00003000: $10 <= 0000006f

130@00003004: $ 9 <= 0000006f

170@0000300c: $12 <= ffffff91

(2)Rt----store

addu $t1,$t2,$t3

nop

sw $t1,0($t2)

110@00003000: $ 9 <= 00000000

130@00003008: \*00000000 <= 00000000

5.E级Rt与W级cal\_i

(1)Rt----cal\_r

ori $t2,$0,111

ori $t1,$t2,0

nop

subu $t4,$0,$t1

110@00003000: $10 <= 0000006f

130@00003004: $ 9 <= 0000006f

170@0000300c: $12 <= ffffff91

(2)Rt----store

ori $t1,$t2,$t3

nop

sw $t1,0($t2)

110@00003000: $ 9 <= 00000064

130@00003008: \*00000000 <= 00000064

6.E级Rt与W级load

(1)Rt----cal\_r

ori $t1,1

ori $t0,0x00000000

lw $t1,0($t0)

nop

addu $t2,$t0,$t1

110@00003000: $ 9 <= 00000001

130@00003004: $ 8 <= 00000000

150@00003008: $ 9 <= 00000000

190@00003010: $10 <= 00000000

(2)Rt---store

ori $t0,0x00000000

lw $t1,0($t0)

nop

sw $t1,0($t0)

110@00003000: $ 8 <= 00000000

130@00003004: $ 9 <= 00000000

150@0000300c: \*00000000 <= 00000000

7.E级Rt与W级jal

(1)Rt----cal\_r

jal eee

nop

eee:

subu $t1,$t2,$ra

110@00003000: $31 <= 00003008

150@00003008: $ 9 <= ffffcff8

(2)Rt----store

ori $t1,0x00003000

jal eee

subu $ra,$ra,$t1

eee:

sw $ra,0($0)

110@00003000: $ 9 <= 00003000

130@00003004: $31 <= 0000300c

150@00003008: $31 <= 0000000c

150@0000300c: \*00000000 <= 0000000c

五．M级Rt

1.M级Rt与W级cal\_r

ori $t2,10

addu $t1,$t2,$t3

sw $t1,0($0)

110@00003000: $10 <= 0000000a

130@00003004: $ 9 <= 0000000a

130@00003008: \*00000000 <= 0000000a

2.M级Rt与W级cal\_i

ori $t2,10

ori $t1,$t2,10

sw $t1,0($0)

110@00003000: $10 <= 0000000a

130@00003004: $ 9 <= 0000000a

130@00003008: \*00000000 <= 0000000a

3.M级Rt与W级load

ori $t2,10

lw $t2,0($0)

sw $t2,0($0)

110@00003000: $10 <= 0000000a

130@00003004: $10 <= 00000000

130@00003008: \*00000000 <= 00000000

4.M级Rt与W级jal

ori $t2,10

jal eee

sw $ra,0($0)

eee:

110@00003000: $10 <= 0000000a

130@00003004: $31 <= 0000300c

130@00003008: \*00000000 <= 0000300c

(2)暂停机制覆盖测试

测试目录：

一．Beq\_rs/rt

(1)E级cal\_r\_rd

(2) E级cal\_i\_rt

(3) E级load\_rt

(4) M级load\_rt

二．Cal\_r\_rs/rt

E级load\_rt

三．Cal\_i\_rs

E级load\_rt

四．load\_rs

E级load\_rt

五．store\_rs

E级load\_rt

六．jr\_rs

(1)E级cal\_r\_rd

(2) E级cal\_i\_rt

(3) E级load\_rt

(4) M级load\_rt

一．Beq

(1) E段cal\_r

ori $s0,1

addu $s1,$s0,$0

beq $s0,$s1,eee

nop

eee:

nop

110@00003000: $16 <= 00000001

130@00003004: $17 <= 00000001

(2) E段cal\_i

ori $s0,1

ori $s1,$s0,2

beq $s1,$s0,eee

nop

eee:

nop

110@00003000: $16 <= 00000001

130@00003004: $17 <= 00000003

(3) E段load

ori $s0,$0,10

lw $s1,0($0)

beq $s1,$s0,eee

nop

eee:

nop

110@00003000: $16 <= 0000000a

130@00003004: $17 <= 00000000

(4) M段load

ori $s0,$0,10

lw $s1,0($0)

nop

beq $s1,$s0,eee

nop

eee:

addu $t0,$t0,$t0

110@00003000: $16 <= 0000000a

130@00003004: $17 <= 00000000

230@00003014: $ 8 <= 00000000

二．Cal\_r

E段load

lw $t2,0($0)

addu $t2,$t2,$t2

110@00003000: $10 <= 00000000

150@00003004: $10 <= 00000000

三．Cal\_i

E段load

lw $t2,0($0)

ori $t2,$t2,100

110@00003000: $10 <= 00000000

150@00003004: $10 <= 00000064

四．Load

E段load

lw $t2,0($0)

lw $t3,0($t2)

110@00003000: $10 <= 00000000

150@00003004: $11 <= 00000000

五．store

E段load

lw $t2,0($0)

sw $t3,0($t2)

110@00003000: $10 <= 00000000

130@00003004: \*00000000 <= 00000000

六．Jr

(1) E段cal\_r

ori $t3,$0,0x0000300c

addu $t2,$t2,$t3

jr $t2

nop

110@00003000: $11 <= 0000300c

130@00003004: $10 <= 0000300c

(2) E段cal\_i

ori $t3,$0,0x0000300c

ori $t2,$t3,0

jr $t2

nop

110@00003000: $11 <= 0000300c

130@00003004: $10 <= 0000300c

(3) E段load

ori $t3,$0,0x0000300c

sw $t3,0($0)

lw $t2,0($0)

jr $t2

nop

110@00003000: $11 <= 0000300c

110@00003004: \*00000000 <= 0000300c

150@00003008: $10 <= 0000300c

(4) M段load

ori $t3,$0,0x0000300c

sw $t3,0($0)

lw $t2,0($0)

nop

jr $t2

nop

110@00003000: $11 <= 0000300c

110@00003004: \*00000000 <= 0000300c

150@00003008: $10 <= 0000300c

综合测试

测试代码

ori $a0,$0,1999

ori $a1,$a0,111

lui $a2,12345

lui $a3,0xffff

lui $t0,0xffff

beq $a3,$t0,eee

addu $s7,$0,$a0

nop

ori $a3,$a3,0xffff

addu $s0,$a0,$a1

addu $s1,$a3,$a3

addu $s2,$a3,$s0

beq $s2,$s3,eee

subu $s0,$a0,$s2

subu $s1,$a3,$a3

eee:

subu $s2,$a3,$a0

subu $s3,$s2,$s1

ori $t0,$0,0x0000

sw $a0,0($t0)

nop

sw $a1,4($t0)

sw $s0,8($t0)

sw $s1,12($t0)

sw $s2,16($t0)

sw $s5,20($t0)

lw $t1,20($t0)

lw $t7,0($t0)

lw $t6,20($t0)

sw $t6,24($t0)

lw $t5,12($t0)

jal end

ori $t0,$t0,1

ori $t1,$t1,1

ori $t2,$t2,2

beq $t0,$t2,eee

lui $t3,1111

jal out

end:

addu $t0,$t0,$t7

jr $ra

out:

addu $t0,$t0,$t3

ori $t2,$t0,0

beq $t0,$t2,qqq

lui $v0,10

qqq:

lui $v0,11

j www

nop

www:

lui $ra,100

机器码

340407cf

3485006f

3c063039

3c07ffff

3c08ffff

10e80009

0004b821

00000000

34e7ffff

00858021

00e78821

00f09021

12530002

00928023

00e78823

00e49023

02519823

34080000

ad040000

00000000

ad050004

ad100008

ad11000c

ad120010

ad150014

8d090014

8d0f0000

8d0e0014

ad0e0018

8d0d000c

0c000c25

35080001

35290001

354a0002

110affec

3c0b0457

0c000c27

010f4021

03e00008

010b4021

350a0000

110a0001

3c02000a

3c02000b

08000c2e

00000000

3c1f0064

测试输出

110@00003000: $ 4 <= 000007cf

130@00003004: $ 5 <= 000007ef

150@00003008: $ 6 <= 30390000

170@0000300c: $ 7 <= ffff0000

190@00003010: $ 8 <= ffff0000

250@00003018: $23 <= 000007cf

270@0000303c: $18 <= fffef831

290@00003040: $19 <= fffef831

310@00003044: $ 8 <= 00000000

310@00003048: \*00000000 <= 000007cf

350@00003050: \*00000004 <= 000007ef

370@00003054: \*00000008 <= 00000000

390@00003058: \*0000000c <= 00000000

410@0000305c: \*00000010 <= fffef831

430@00003060: \*00000014 <= 00000000

470@00003064: $ 9 <= 00000000

490@00003068: $15 <= 000007cf

510@0000306c: $14 <= 00000000

510@00003070: \*00000018 <= 00000000

550@00003074: $13 <= 00000000

570@00003078: $31 <= 00003080

590@0000307c: $ 8 <= 00000001

610@00003094: $ 8 <= 000007d0

650@0000309c: $ 8 <= 000007d0

670@00003080: $ 9 <= 00000001

690@00003084: $10 <= 00000002

750@0000308c: $11 <= 04570000

770@00003090: $31 <= 00003098

790@00003094: $ 8 <= 00000f9f

810@0000309c: $ 8 <= 04570f9f

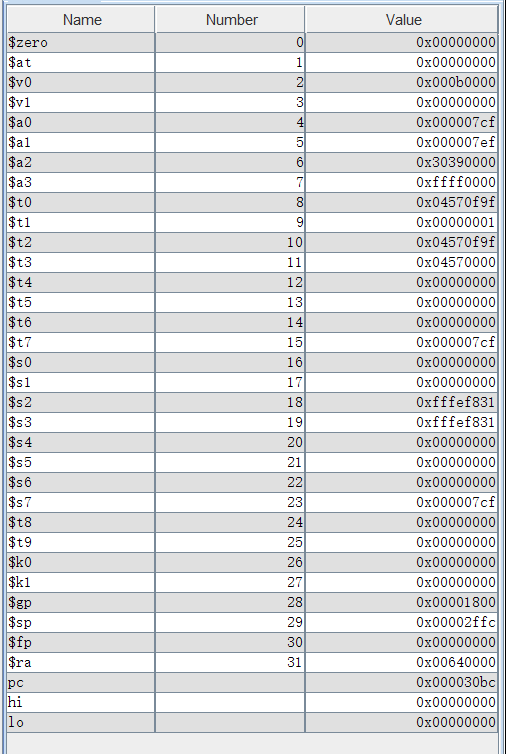
830@000030a0: $10 <= 04570f9f

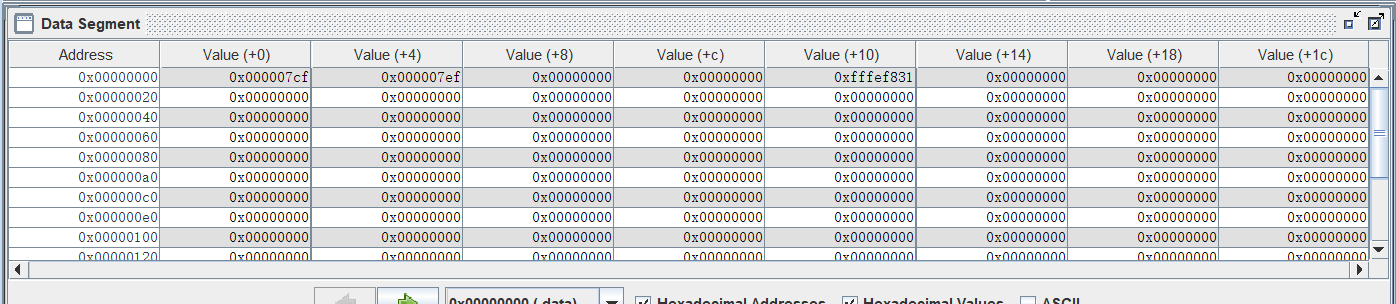
890@000030a8: $ 2 <= 000a0000

910@000030ac: $ 2 <= 000b0000

970@000030b8: $31 <= 00640000

MARS模拟



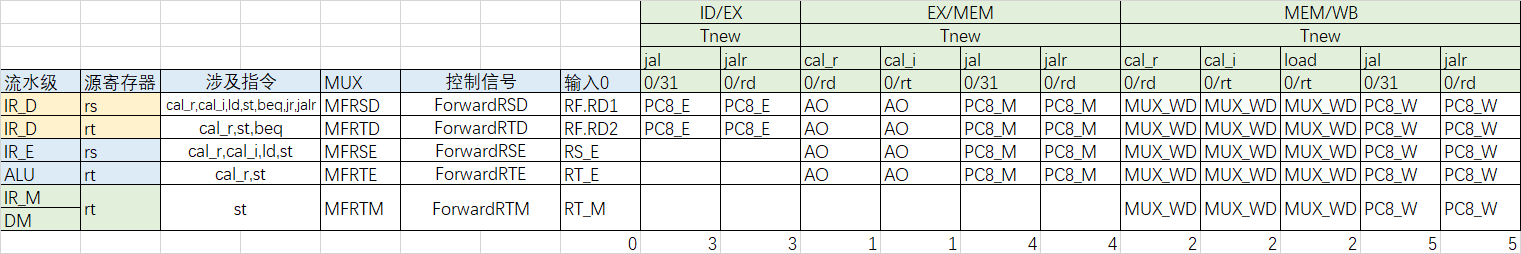


六．思考题

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**)

冲突（转发与暂停机制）已经在测试程序中覆盖测试，参见上一块内容。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| IF/ID当前指令 | | | ID/EX | | | EX/MEM |
| 指令类型 | 源寄存器 | Tuse | Tnew | | | Tnew |
| cal\_r | cal\_i | load | load |
| 1/rd | 1/rt | 2/rt | 1/rt |
| beq | rs/rt | 0 | 暂停 | 暂停 | 暂停 | 暂停 |
| cal\_r | rs/rt | 1 |  |  | 暂停 |  |
| cal\_i | rs | 1 |  |  | 暂停 |  |
| load | rs | 1 |  |  | 暂停 |  |
| store | rs | 1 |  |  | 暂停 |  |
| store | rt | 2 |  |  |  |  |
| jr | rs | 0 | 暂停 | 暂停 | 暂停 | 暂停 |
| jalr | rs | 0 | 暂停 | 暂停 | 暂停 | 暂停 |



解决冲突：暂停与转发