CPU-流水线工程化设计文档

目录

[一、 模块设计 1](#_Toc11025)

[1.IFU 1](#_Toc10941)

[2.GRF 2](#_Toc3135)

[3.ALU 3](#_Toc27475)

[4.DM 3](#_Toc5053)

[5.EXT 4](#_Toc6211)

[6.顶层模块（mips） 5](#_Toc25938)

[二、 控制器设计 8](#_Toc12908)

[三、冒险处理 10](#_Toc8636)

[1.数据通路表格 10](#_Toc13731)

[2.冒险表格 14](#_Toc25434)

[3.转发表格 16](#_Toc12578)

[四、测试代码 20](#_Toc18799)

[CPU-text-1: 20](#_Toc10317)

[CPU-text-2: 21](#_Toc20051)

[五、思考题 24](#_Toc9877)

1. 模块设计

**1.IFU**

IFU内部包括 PC、IM(指令存储器)、Adder、选择器和beq跳转值作为输入。PC 用寄存器实现，具有复位功能且有时钟信号。起始地址：0x00003000。IM用ROM 实现。IM的实际地址宽度10位与PC中2-11位相接，ROM输出Instr为指令编码，选择器的选择信号为beq跳转信号，即当指令为beq且置零端为真时选择信号为1。添加了J,Jal,Jr指令，分别有jump(1bit),jr(1bit),ra(32bit),index(26)输出端口，还外输出PC+4的值。IFU里调用了一个ROM块，ROM采用readmemh函数读取指令码。

表 1-IFU端口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 定义 | | 描述 |
| 1 | input | CLK | 时钟信号，下一个时钟上升沿来到PC寄存器存入当前PC |
| 2 | RESET | 复位信号，当复位信号有效时，PC被设置为0x00003000 |
| 3 | beq | 32位，指令中立即数imm移位且符号扩展后的值 |
| 4 | Zero | 多路选择器选择信号，当为beq指令且ALU置零端为真时为1 |
| 5 | output | Instr | 输出对应PC的指令码 |
| 新增 | 定义 | | 描述 |
| 1 | input | jump | 1bit，为1时跳转到PC[31...28] || instr\_index(指令码后26位) || 00 |
| 2 | ra | 32位$ra的值($31=PC+4) |
| 3 | jr | 1bit,为1时跳转至$ra($31)保存的地址 |
| 4 | index | instr\_index(指令码后26位) |
| 5 | output | PC\_ | PC+4的值，32bit |

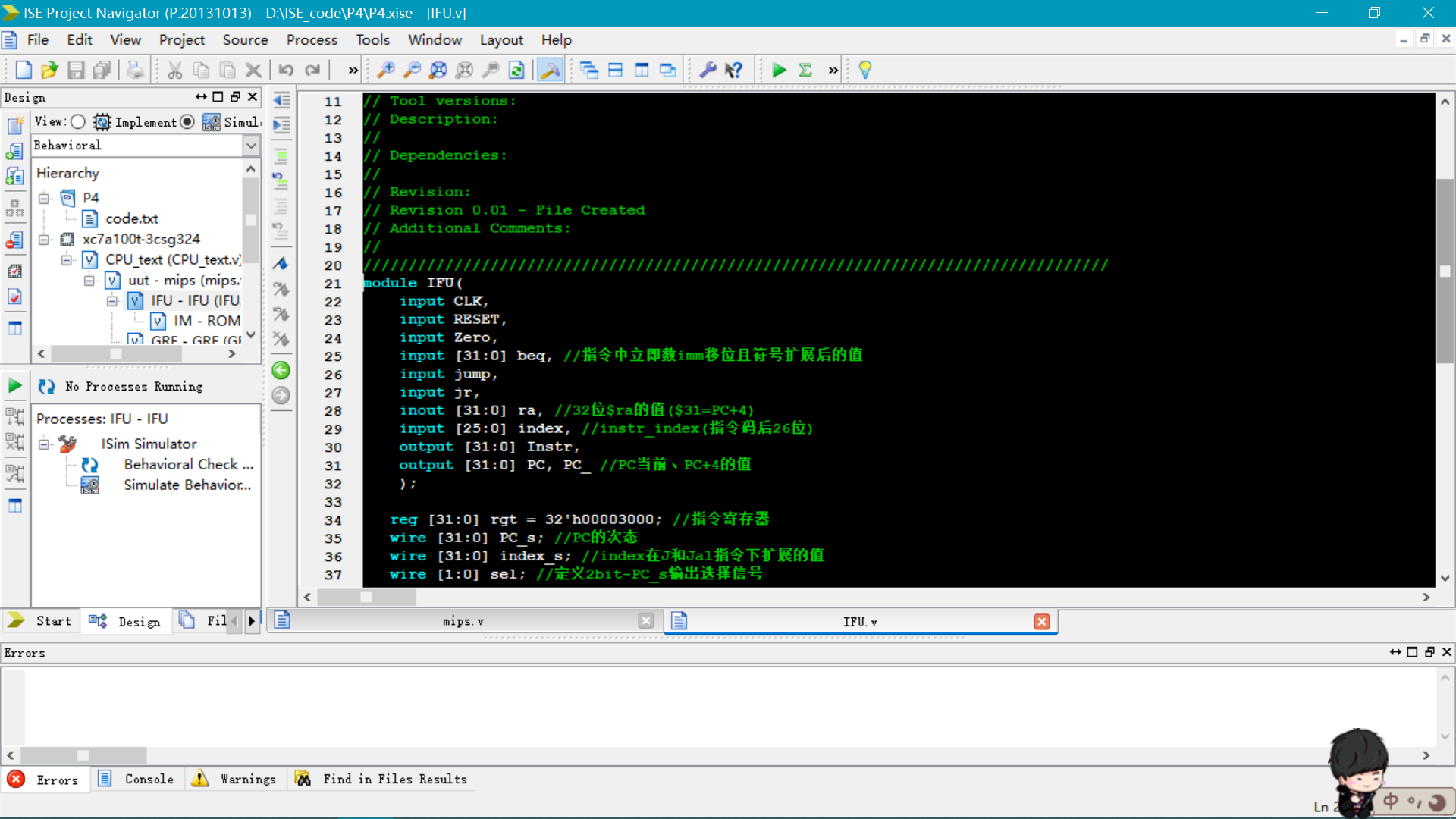


图 1-IFU模块图

**2.GRF**

用具有写使能和复位信号的寄存器实现，寄存器总数为 32 个，第一个寄存器值始终为0x00000000，其他寄存器默认为0x00000000。输入端有三个地址，一个数据输入，两个数据输出。增加一个输入端口PC，用于接收来自IFU的PC的值。

表 2-GRF端口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 定义 | | 描述 |
| 1 | input | CLK | 时钟信号，下一个时钟上升沿来临且写入信号有效写入数据 |
| 2 | RESET | 复位信号，当复位信号有效时，寄存器被清零 |
| 3 | WE | 写入寄存器使能信号 |
| 4 | A1 | Rs寄存器地址5bit |
| 5 | A2 | Rt寄存器地址5bit |
| 6 | WA | Rt或Rd寄存器地址5bit |
| 7 | WD | 需要写入寄存器的值 |
| 8 | PC | 接收来自IFU的当前PC值 |
| 9 | output | RD1 | 输出对应Rs寄存器里的值 |
| 10 | RD2 | 输出对应Rt寄存器里的值 |

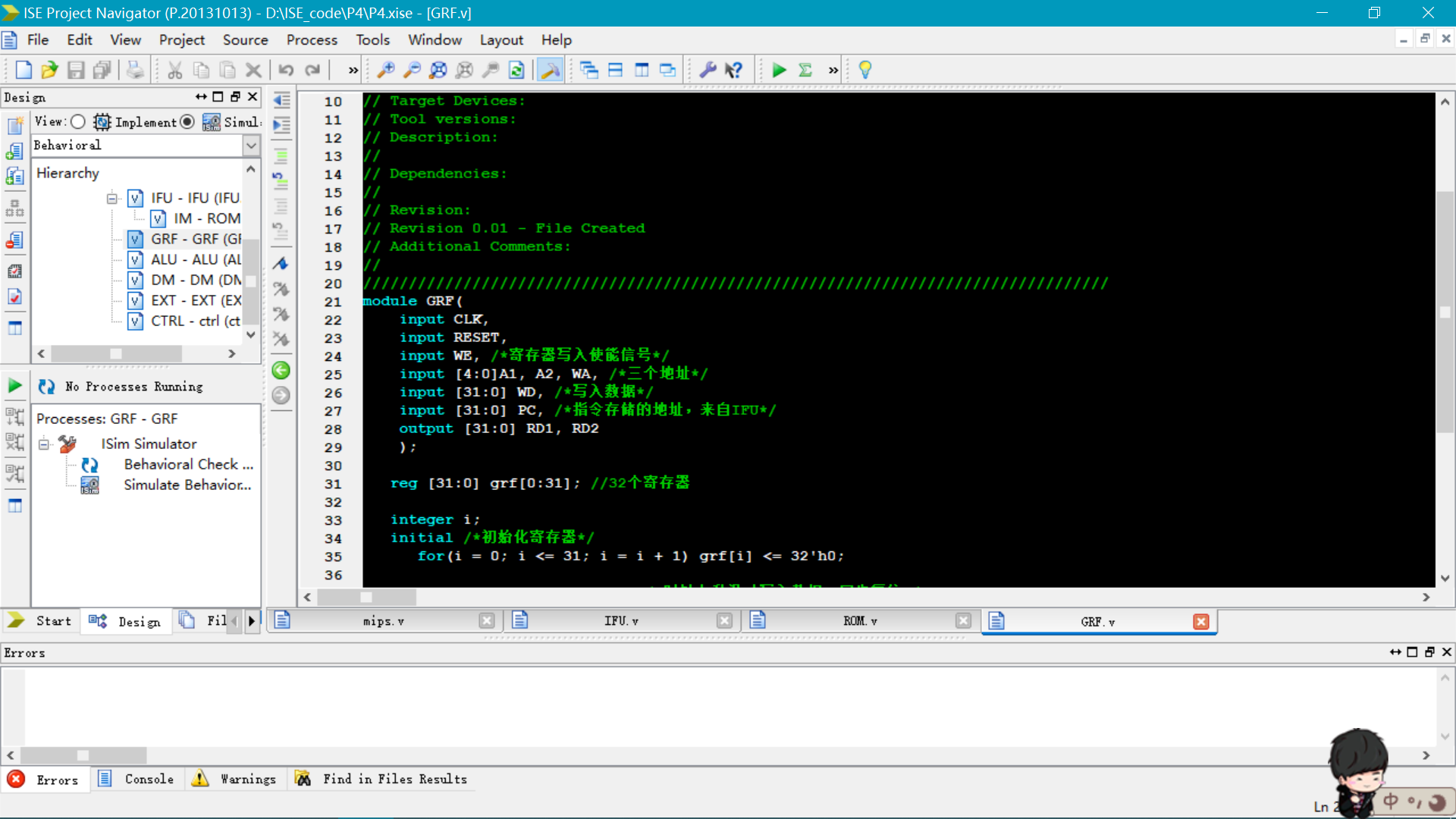


图 2-RGF模块图

**3.ALU**

算术逻辑单元，32位无符号加、无符号减、立即数或运算及大小比较功能。在这里大小比较采用无符号数相减之后的结果与32’h0比较，相等则输出1。

表 3-ALU端口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 定义 | | 描述 |
| 1 | input | OP | ALU运算控制信号，0001无符号加、0010无符号减、0100立即数或、1000立即数加载至高位 |
| 2 | A | 第一个数，32位 |
| 3 | B | 第二个数，32位 |
| 4 | output | Result | 运算结果，32位 |
| 5 | Zero | A-B=0时Zero=1，置零端 |

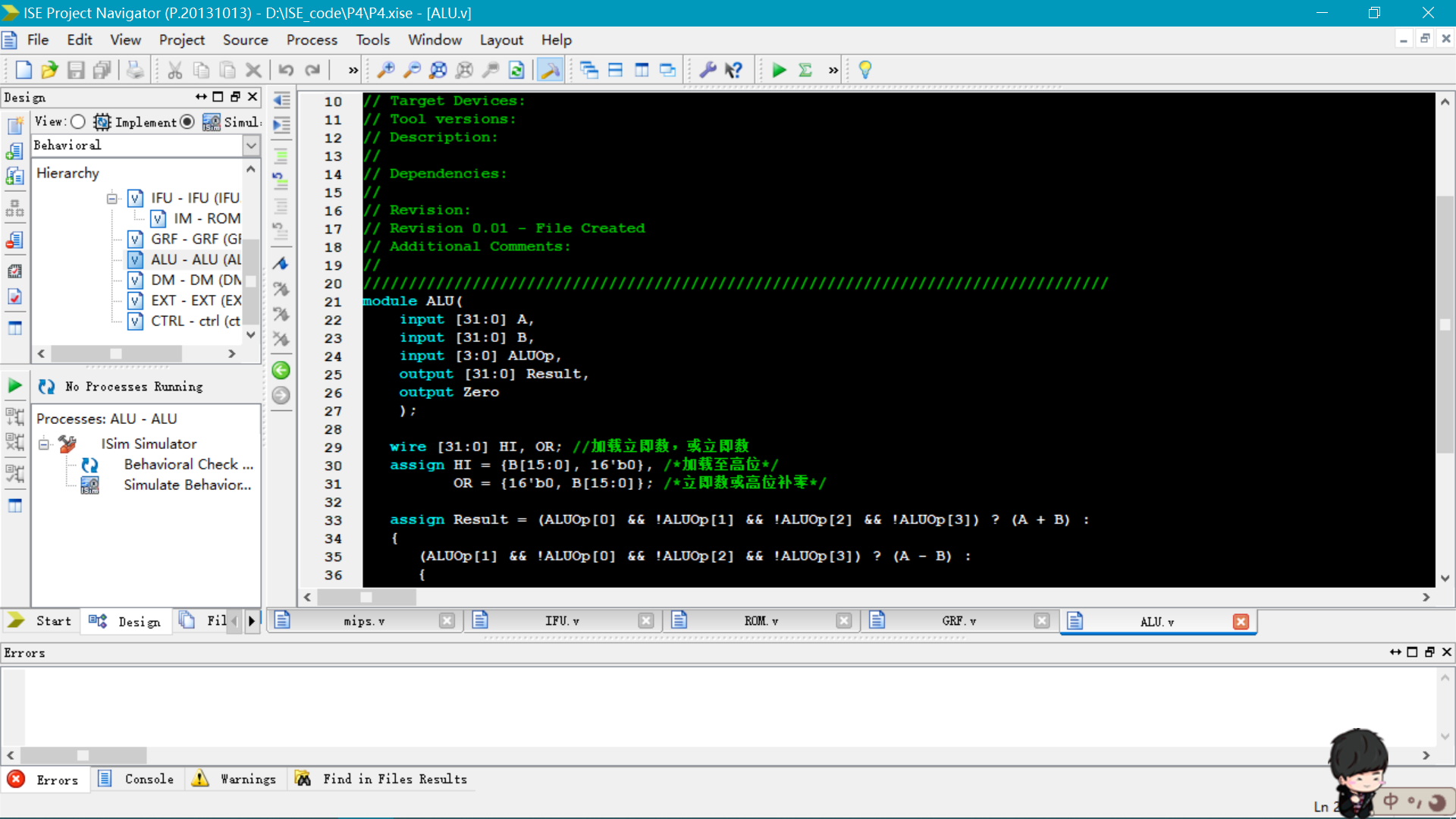


图 3-ALU模块图

**4.DM**

数据存储器，使用RAM实现，容量为32bit \* 1024。起始地址：0x00000000。输入端口有10位地址，写入数据32位等，输出对应的32位数据。增加一个输入端口PC，用于接收来自IFU的PC的值。

表 4-DM端口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 定义 | | 描述 |
| 1 | input | clk | 时钟信号，当时钟上升沿来到且写入信号有效则写入数据 |
| 2 | clr | 复位信号 |
| 3 | A | 10位存储器地址 |
| 4 | WD | 写入的数据，32位 |
| 5 | SD | SW指令存数信号 |
| 6 | LD | LW指令取数信号 |
| 7 | PC | 接收来自IFU的当前PC值 |
| 8 | output | RD | 读取数据，32位 |

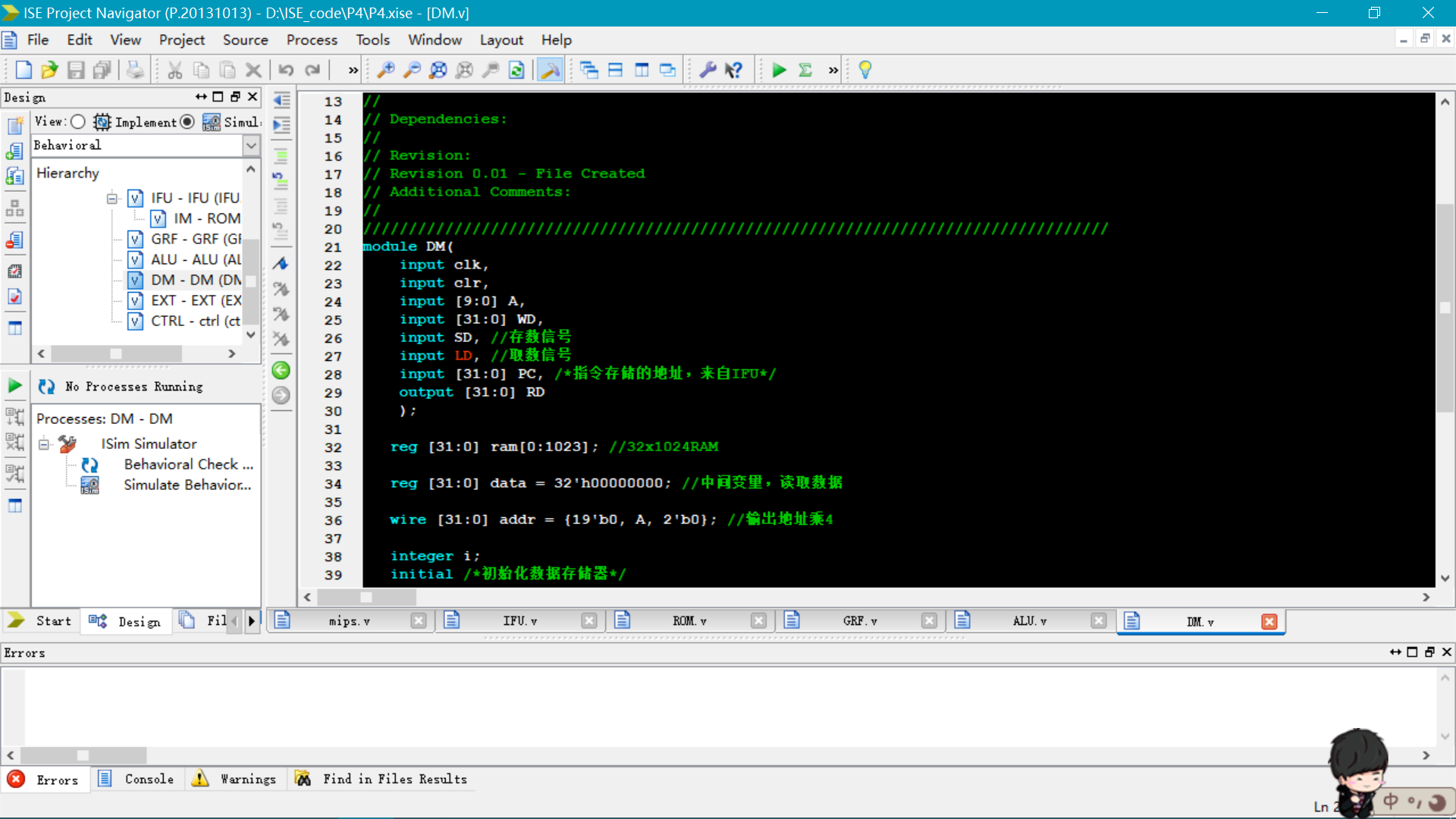


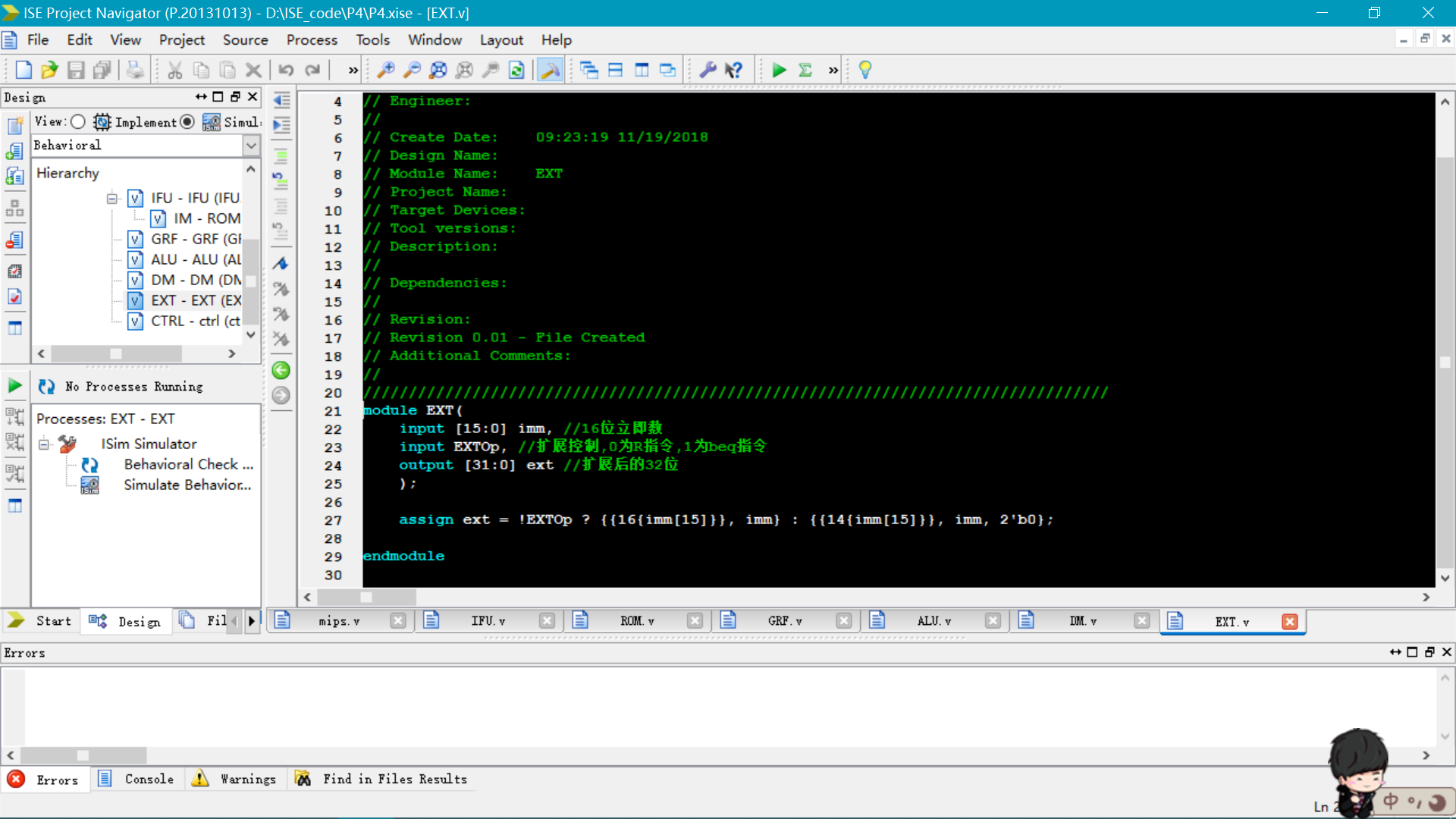
图 4-DM模块图

**5.EXT**

扩展器，该扩展器包括了跳转指令的立即数扩展。

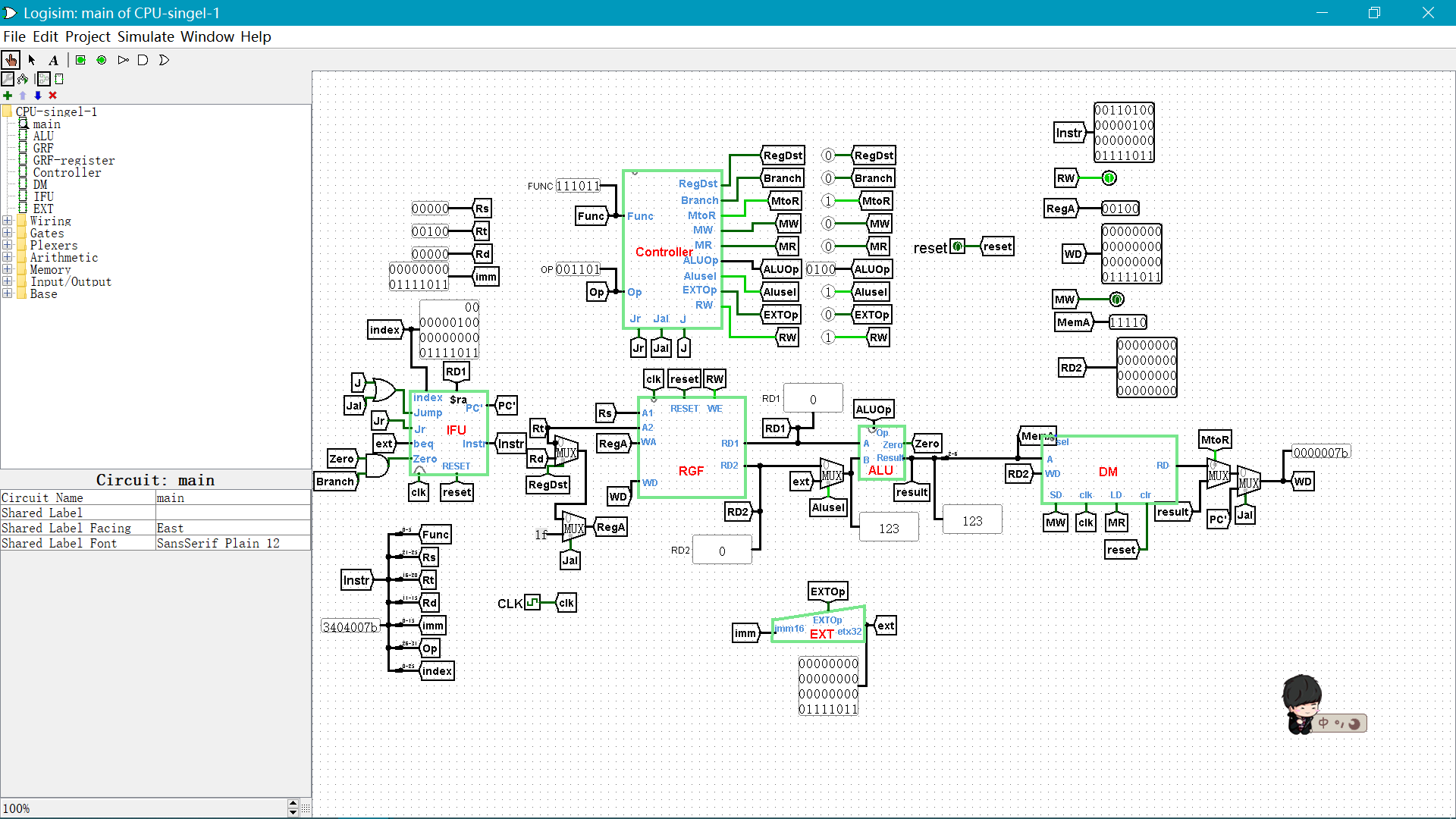
表 5-EXT端口定义

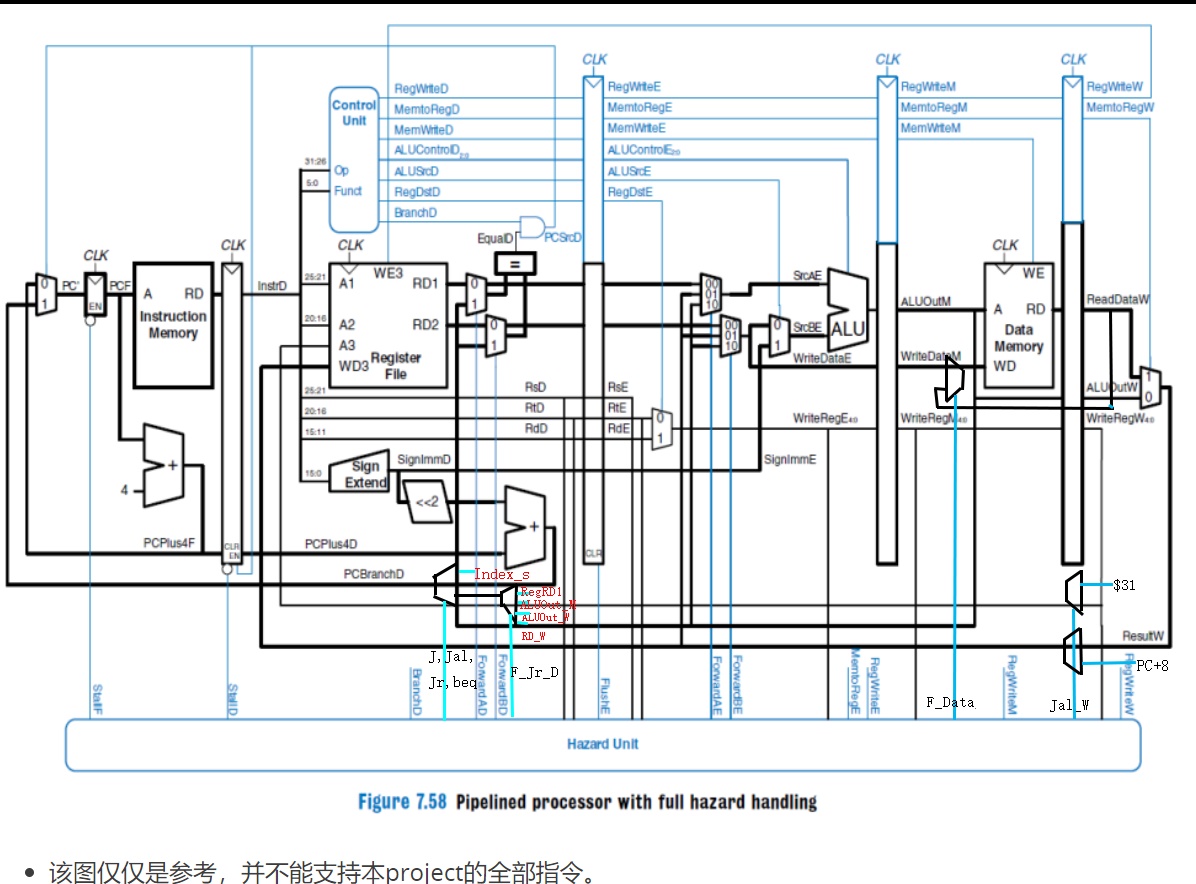
|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 定义 | | 描述 |
| 1 | input | EXTOp | 扩展类型信号，为0代表符号扩展，为1代表beq跳转(sign(imm||00))左移两位后符号扩展 |
| 2 | imm16 | 16位立即数或offset |
| 3 | output | ext32 | 扩展后的32位数 |

图 5-EXT模块图

**6.顶层模块（mips）**

这里是几个模块连接而成的单周期CPU顶层块，顶层模块也是用来调用各个模块，定义wire型变量将各模块的信号相连。利用assign对变量进行链接。





f\_Data\_M

1. 控制器设计

此控制器ALUOp采用独热编码，0001无符号加、0010无符号减、0100立即数或、1000立即数加载至高位；在DM（数据存储器）模块增加了MW,MR作为写入使能信号、读取信号；EXTOp信号用来选择立即数扩展的类型，扩展器选择信号，0选择符号扩展，1选择beq跳转。之后添加了J，Jal, Jr输出信号，代表相应的跳转信号。

表 6-控制器端口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 定义 | | 描述 |
| 1 | input | Func | 指令码后6位 |
| 2 | Op | 指令码前6位操作码 |
| 3 | output | RegDst | Rt与Rd选择信号，0选择Rt，1选择Rd |
| 4 | Branch | 若是分支指令（当前指beq）为1 |
| 5 | MtoR | 写入寄存器数据的选择信号，0选择存储器读取的数据，1选择ALU运算的result |
| 6 | MW | 存储器写入信号 |
| 7 | MR | 存储器读取信号 |
| 8 | ALUOp | ALU控制信号，0001无符号加、0010无符号减、0100立即数或、1000立即数加载至高位 |
| 9 | Alusel | ALU数据B端口的选择信号，0选择RD2，1选择扩展数 |
| 10 | EXTOp | 扩展器选择信号，0选择符号扩展，1选择beq跳转 |
| 11 | RW | 寄存器文件写入信号 |
| 新增 | 定义 | | 描述 |
| 12 | output | J | 若是跳转指令J则为1 |
| 13 | Jal | 若是跳转指令Jal则为1 |
| 14 | Jr | 若是跳转指令Jr则为1 |
| 15 | output | LB | 若是加载半字指令则为1 |

表 7-控制器功能真值表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Func | 100001 | 100011 | x | x | x | x | x |
| Op | 000000 | | 001101 | 100011 | 101011 | 000100 | 001111 |
|  | Addu | Subu | Ori | Lw | Sw | Beq | Lui |
| RegDst | 1 | 1 | 0 | 0 | x | x | 0 |
| Branch | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| MtoR | 0 | 0 | 0 | 1 | x | x | 0 |
| MW | x | x | x | 0 | 1 | x | x |
| MR | x | x | x | 1 | 0 | x | x |
| ALUOp | 0001 | 0010 | 0100 | 0001 | 0001 | 0010 | 1000 |
| Alusel | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| EXTOp | x | x | 0 | 0 | 0 | 1 | 0 |
| RW | 1 | 1 | 1 | 1 | x | x | 1 |
| Func | x | x | 001000 | x |  |  |  |
| Op | 000010 | 000011 | 000000 | 100000 |  |  |  |
| 新增 | J | Jal | Jr | LB |  |  |  |
| RegDst | x | x | x | x |  |  |  |
| Branch | x | x | x | x |  |  |  |
| MtoR | x | x | x | 1 |  |  |  |
| MW | x | x | x | x |  |  |  |
| MR | x | x | x | 1 |  |  |  |
| ALUOp | x | x | x | 0001 |  |  |  |
| Alusel | x | x | x | 1 |  |  |  |
| EXTOp | x | x | x | 0 |  |  |  |
| RW | x | 1 | x | 1 |  |  |  |
| J | 1 | 0 | 0 | x |  |  |  |
| Jal | 0 | 1 | 0 | x |  |  |  |
| Jr | 0 | 0 | 1 | x |  |  |  |

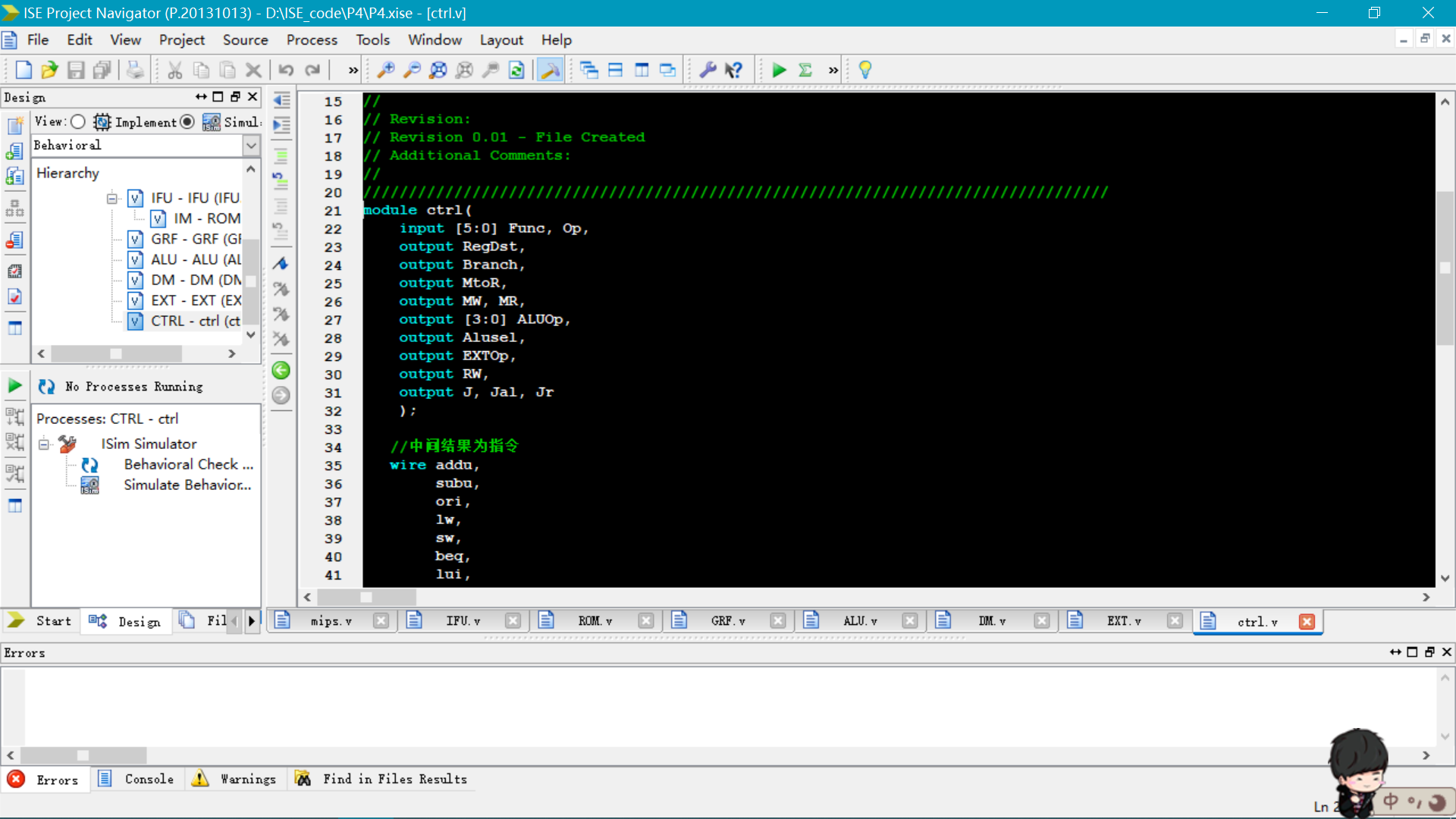
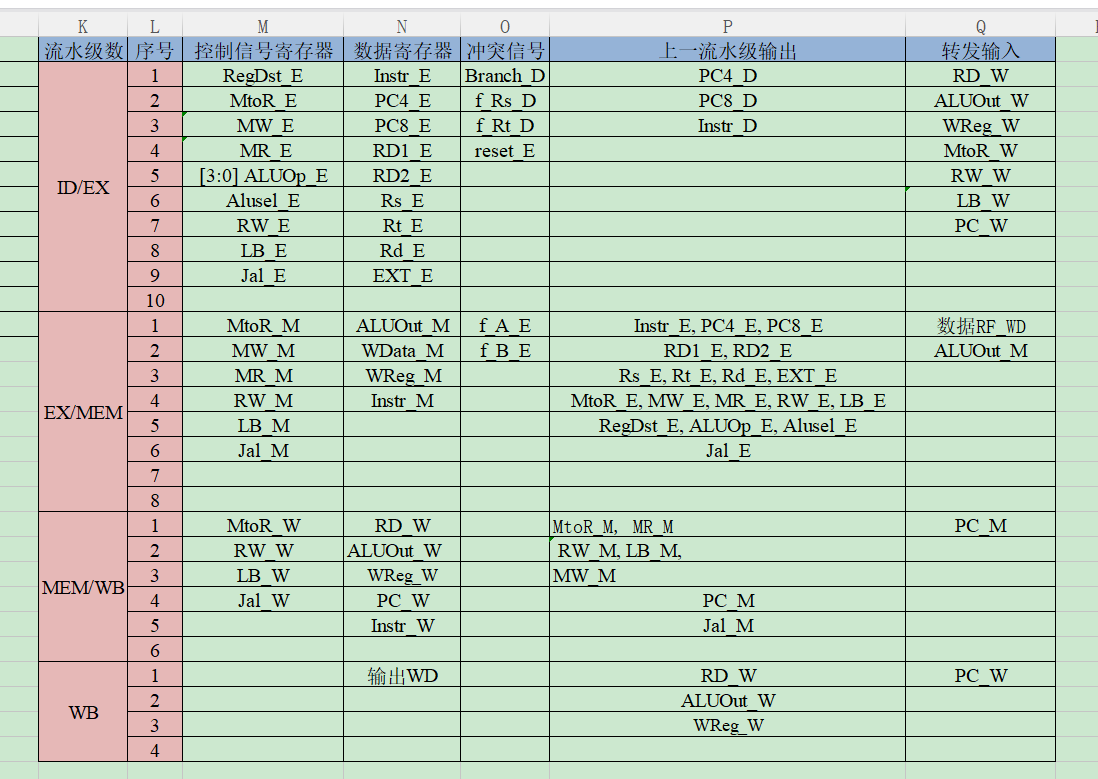


图 7-Controller模块图

三、冒险处理

**1.数据通路表格**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 流水级数 | 部件 | 输入端 | 备注 | 输入来源 | | | | MUX | MUX信号 | **Add**u | **Sub**u | **Ori** | **Lw** | **Sw** | **Beq** | **Lui** | **J** | **Jal** | **Jr** | **LB** |  |
| IF | PC |  | **PC**选择,beq,j,jal,jr | add4 | NPC |  |  | mux\_PC | **PCsrc\_D** | add4 | add4 | add4 | add4 | add4 | NPC | add4 | NPC | NPC | NPC | add4 |  |
| add4 |  |  | PC |  |  |  |  |  | PC | PC | PC | PC | PC | PC | PC | PC | PC | PC | PC |  |
| IM |  |  | PC |  |  |  |  |  | PC | PC | PC | PC | PC | PC | PC | PC | PC | PC | PC |  |
| ID更新PC | PC\_D |  | 当前指令PC值 | PC |  |  |  |  |  | PC | PC | PC | PC | PC | PC | PC | PC | PC | PC | PC |  |
| F/D寄存器 | Instr\_D |  | 下一级指令 | IM |  |  |  |  |  | IM | IM | IM | IM | IM | IM | IM | IM | IM | IM | IM |  |
| PC4\_D |  |  | PC\_D+4 |  |  |  |  |  | add4 | add4 | add4 | add4 | add4 | add4 | add4 | add4 | add4 | add4 | add4 |  |
| PC8\_D |  |  | PC4\_D+4 |  |  |  |  |  | add4+4 | add4+4 | add4+4 | add4+4 | add4+4 | PC | add4+4 | PC | PC | PC | add4+4 |  |
| ID | RegFile | A1 |  | Instr\_D[Rs] |  |  |  |  |  | Instr\_D[Rs] | Instr\_D[Rs] | Instr\_D[Rs] | Instr\_D[Rs] | Instr\_D[Rs] | Instr\_D[Rs] | Instr\_D[Rs] |  |  | Instr\_D[Rs] | Instr\_D[Rs] |  |
| A2 |  | Instr\_D[Rt] |  |  |  |  |  | Instr\_D[Rt] | Instr\_D[Rt] | Instr\_D[Rt] | Instr\_D[Rt] | Instr\_D[Rt] | Instr\_D[Rt] |  |  |  |  | Instr\_D[Rt] |  |
| CMP比较器 | RD1 | 转发进行选择 | RegRD1 | ALUout\_M |  |  | MFRsD | f\_Rs\_D | RegRD1 | RegRD1 | RegRD1 | RegRD1 | RegRD1 | mux | RegRD1 | RegRD1 | RegRD1 | RegRD1 | RegRD1 |  |
| RD2 | 转发进行选择 | RegRD2 | ALUout\_M |  |  | MFRtD | f\_Rt\_D | RegRD2 | RegRD2 | RegRD2 | RegRD2 | RegRD2 | mux | RegRD2 | RegRD2 | RegRD2 | RegRD2 | RegRD2 |  |
| EXT |  | 符号扩展 | Instr\_D\_16 |  |  |  |  |  |  |  | Instr\_D\_16 | Instr\_D\_16 | Instr\_D\_16 | Instr\_D\_16 | Instr\_D\_16 | Instr\_D\_26 | Instr\_D\_26 |  | Instr\_D\_16 |  |
| NPC选址PC | PC4 |  | PC4\_D |  |  |  |  |  | PC4\_D | PC4\_D | PC4\_D | PC4\_D | PC4\_D | PC4\_D | PC4\_D | PC4\_D | PC4\_D | PC4\_D | PC4\_D |  |
| J\_jr | jr | RegRD1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | jr |  |  |
| Index\_26 | j||jal | Instr\_D\_26 |  |  |  |  |  |  |  |  |  |  | EXT<<2+4 |  | Instr\_D\_26 | Instr\_D\_26 |  |  |  |
| & | beq,EXT<<2+4 | CMP为真 | Branch\_D |  |  |  |  |  |  |  |  |  | beq |  |  |  |  |  |  |
| EX更新PC | PC\_E |  | 当前指令PC值 | PC\_D |  |  |  |  |  | add4 | add4 | add4 | add4 | add4 | NPC | add4 | NPC | NPC | D1 | add4 |  |
| D/E寄存器 | Instr\_E |  |  | Instr\_D |  |  |  |  |  | Instr\_D | Instr\_D | Instr\_D | Instr\_D | Instr\_D | Instr\_D | Instr\_D | Instr\_D | Instr\_D | Instr\_D | Instr\_D |  |
| PC4\_E |  |  | PC4\_D |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| PC8\_E |  |  | PC8\_D |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| RD1\_E |  |  | RegRD1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| RD2\_E |  |  | RegRD2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Rs\_E |  |  | Instr\_E[Rs] |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Rt\_E |  |  | Instr\_E[Rt] |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Rd\_E |  |  | Instr\_E[Rd] |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Jal\_E |  | 存储jal指令 | Jal来自控制器 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| EXT\_E |  |  | EXT |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| EX | ALU | A |  | RD1\_E | RF\_WD | ALUout\_M |  | mux\_A | f\_A\_E |  |  |  |  |  |  |  |  |  |  |  |  |
| B |  | RD2\_E | RF\_WD | ALUout\_M | EXT\_E | WD\_E/mux\_B | f\_B\_E/Alusel\_E |  |  | EXT\_E | EXT\_E | EXT\_E |  | EXT\_E |  |  |  | EXT\_E |  |
| XALU | D1 |  | Rs\_E |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| D2 |  | Rt\_E |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| WReg\_E |  |  | Rt\_E | Rd\_E |  |  | mux\_WReg | RegDst\_E | Rd\_E | Rd\_E | Rt\_E | Rt\_E | Rt\_E |  | Rt\_E |  |  |  | Rt\_E |  |
|  | PC\_M |  |  | PC\_E |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| E/M寄存器 | Instr\_M |  |  | Instr\_E |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| PC4\_M |  |  | PC4\_E |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| PC8\_M |  |  | PC8\_E |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ALUout\_M |  |  | ALU |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| XALUout\_M |  |  | XALU |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| WData\_M |  |  | WD\_E |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| WReg\_M |  |  | WReg\_E |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Jal\_M |  |  | Jal\_E |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| M | DM | A |  | ALUout\_M |  |  |  |  |  |  |  |  | ALUout\_M | ALUout\_M |  |  |  |  |  | ALUout\_M |  |
| WD |  | WData\_M |  |  |  |  |  |  |  |  | WD\_M | WD\_M |  |  |  |  |  | WD\_M |  |
|  | PC\_W |  |  | PC4\_W-4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| M/W寄存器 | Instr\_W |  |  | Instr\_M |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| PC4\_W |  |  | PC4\_M |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| PC8\_W |  |  | PC8\_M |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ALUout\_W |  |  | ALUout\_M |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| XALUout\_W |  |  | XALUout\_M |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| RD\_W |  |  | DM |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| WReg\_W |  |  | WReg\_M |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Jal\_W |  |  | Jal\_M |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| WB | EXT\_DM取数lb | A |  | WReg\_W |  |  |  |  |  |  |  |  |  |  |  | WReg\_W |  |  |  |  |  |
| Din |  | RD\_W[b+7:b] |  |  |  |  |  |  |  |  |  |  |  | RD\_W[b+7:b] |  |  |  |  |  |
| RF | A3 |  | WReg\_W |  |  |  |  |  | WReg\_W | WReg\_W | WReg\_W | WReg\_W |  |  | WReg\_W |  |  |  | WReg\_W |  |
| WD | 输出WD | RD\_W | ALUout\_W |  |  |  | MtoR\_W | ALUout\_W | ALUout\_W | ALUout\_W |  | RD\_W |  | ALUout\_W |  |  |  |  |  |
| 流水级数 | 部件 | 输入端 |  | 输入来源 | | | | MUX | MUX信号 | **Add**u | **Sub**u | **Ori** | **Lw** | **Sw** | **Beq** | **Lui** | **J** | **Jal** | **Jr** | **LB** |  |

****

**2.冒险表格**



暂停信号：注意零号寄存器不需要转发

**stall\_b** = beq\_D && (

(cal\_r\_E && (Instr\_E\_Rd == Instr\_D\_Rs || Instr\_E\_Rd == Instr\_D\_Rt)) ||

(cal\_i\_E && (Instr\_E\_Rt == Instr\_D\_Rs || Instr\_E\_Rt == Instr\_D\_Rt)) ||

(load\_E && (Instr\_E\_Rt == Instr\_D\_Rt || Instr\_E\_Rt == Instr\_D\_Rs)) ||

(Instr\_M == load && (Instr\_M\_Rt == Instr\_D\_Rt || Instr\_M\_Rt == Instr\_D\_Rs)) ).

**stall\_cal\_r** = Instr\_E == load && cal\_r\_D && (Instr\_E\_Rt == Instr\_D\_Rs || Instr\_E\_Rt == Instr\_D\_Rt).

**stall\_cal\_i =** Instr\_E == load && cal\_i\_D && (Instr\_E\_Rt == Instr\_D\_Rs || Instr\_E\_Rt == Instr\_D\_Rt).

**stall\_load =** Instr\_E == load && load\_D && Instr\_E\_Rt == Instr\_D\_Rs.

**stall\_store =** Instr\_E == load && store\_D && Instr\_E\_Rt == Instr\_D\_Rs.

**stall\_jr =** jr\_D &&( (Instr\_E == load && Instr\_E\_Rt == Instr\_D\_Rs)

|| (Instr\_M == load && Instr\_M\_Rt == Instr\_D\_Rs)).

stall\_F = **stall\_b + stall\_cal\_r + stall\_cal\_i + stall\_load + stall\_store + stall\_jr.**

stall\_D = **stall\_b + stall\_cal\_r + stall\_cal\_i + stall\_load + stall\_store + stall\_jr.**

控制信号：

Stall\_F.冻结PC；

Reset\_E信号有效，让ID/EX流水级寄存器清零，等于插入了一个NOP指令；

Clk信号上来时且Stall\_D有效,冻结IF/ID流水级寄存器，不让它的值改变。

**3.转发表格**

当非暂停且有冲突的时候，进行转发，表格与暂停表格类似。

转发控制信号：

GRF内部转发：

//寄存器文件内部转发

f\_RD1\_D = (cal\_r\_D || cal\_i\_D || load\_D || store\_D) && cal\_r\_W && Rs\_D == Rd\_W && Rd\_W != 0 ? 2'b01 :

(cal\_r\_D || cal\_i\_D || load\_D || store\_D) && cal\_i\_W && Rs\_D == Rt\_W && Rt\_W != 0 ? 2'b01 :

(cal\_r\_D || cal\_i\_D || load\_D || store\_D) && load\_W && Rs\_D == Rt\_W && Rt\_W != 0 ? 2'b01 :

(cal\_r\_D || cal\_i\_D || load\_D || store\_D) && jal\_W && Rs\_D == 5'h1f ? 2'b10 : 2'b00,

f\_RD2\_D = (cal\_r\_D || store\_D) && cal\_r\_W && Rt\_D == Rd\_W && Rd\_W != 0 ? 2'b01 :

(cal\_r\_D || store\_D) && cal\_i\_W && Rt\_D == Rt\_W && Rt\_W != 0 ? 2'b01 :

(cal\_r\_D || store\_D) && load\_W && Rt\_D == Rt\_W && Rt\_W != 0 ? 2'b01 :

(cal\_r\_D || store\_D) && jal\_W && Rt\_D == 5'h1f ? 2'b10: 2'b00,

**//beq转发**

**f\_Rs\_D = beq\_D && jal\_E && Rs\_D == 5'h1f ? 3'b100 :**

**beq\_D && jal\_M && Rs\_D == 5'h1f ? 3'b101 :**

**beq\_D && cal\_r\_M && Rs\_D == Rd\_M && Rd\_M != 0 ? 3'b001 :**

**beq\_D && cal\_i\_M && Rs\_D == Rt\_M && Rt\_M != 0 ? 3'b001 :**

**beq\_D && cal\_r\_W && Rs\_D == Rd\_W && Rd\_W != 0 ? 3'b010 :**

**beq\_D && cal\_i\_W && Rs\_D == Rt\_W && Rt\_W != 0 ? 3'b010 :**

**beq\_D && load\_W && Rs\_D == Rt\_W && Rt\_W != 0 ? 3'b011 :**

**beq\_D && jal\_W && Rs\_D == 5'h1f ? 3'b110 : 3'b000,**

**f\_Rt\_D = beq\_D && jal\_E && Rt\_D == 5'h1f ? 3'b100 :**

**beq\_D && jal\_M && Rt\_D == 5'h1f ? 3'b101 :**

**beq\_D && cal\_r\_M && Rt\_D == Rd\_M && Rd\_M != 0 ? 3'b001 :**

**beq\_D && cal\_i\_M && Rt\_D == Rt\_M && Rt\_M != 0 ? 3'b001 :**

**beq\_D && cal\_r\_W && Rt\_D == Rd\_W && Rd\_W != 0 ? 3'b010 :**

**beq\_D && cal\_i\_W && Rt\_D == Rt\_W && Rt\_W != 0 ? 3'b010 :**

**beq\_D && load\_W && Rt\_D == Rt\_W && Rt\_W != 0 ? 3'b011 :**

**beq\_D && jal\_W && Rt\_D == 5'h1f ? 3'b110 : 3'b000,**

**//jr转发**

**f\_jr\_D = jr\_D && jal\_E && Rs\_D == 5'h1f ? 3'b100 :**

**jr\_D && cal\_r\_M && Rs\_D == Rd\_M && Rd\_M != 0 ? 3'b001 :**

**jr\_D && cal\_i\_M && Rs\_D == Rt\_M && Rt\_M != 0 ? 3'b001 :**

**jr\_D && jal\_M && Rs\_D == 5'h1f ? 3'b101 :**

**jr\_D && cal\_r\_W && Rs\_D == Rd\_W && Rd\_W != 0 ? 3'b010 :**

**jr\_D && cal\_i\_W && Rs\_D == Rt\_W && Rt\_W != 0 ? 3'b010 :**

**jr\_D && load\_W && Rs\_D == Rt\_W && Rt\_W != 0 ? 3'b011 :**

**jr\_D && jal\_W && Rs\_D == 5'h1f ? 3'b110 : 3'b000,**

**//E级ALU转发**

**f\_A\_E = (cal\_r\_E || cal\_i\_E || load\_E || store\_E) && cal\_r\_M && Rs\_E == Rd\_M && Rd\_M != 0 ? 3'b010 :**

**(cal\_r\_E || cal\_i\_E || load\_E || store\_E) && cal\_i\_M && Rs\_E == Rt\_M && Rt\_M != 0 ? 3'b010 :**

**(cal\_r\_E || cal\_i\_E || load\_E || store\_E) && jal\_M && Rs\_E == 5'h1f ? 3'b100 :**

**(cal\_r\_E || cal\_i\_E || load\_E || store\_E) && cal\_r\_W && Rs\_E == Rd\_W && Rd\_W != 0 ? 3'b001 :**

**(cal\_r\_E || cal\_i\_E || load\_E || store\_E) && cal\_i\_W && Rs\_E == Rt\_W && Rt\_W != 0 ? 3'b001 :**

**(cal\_r\_E || cal\_i\_E || load\_E || store\_E) && load\_W && Rs\_E == Rt\_W && Rt\_W != 0 ? 3'b011 :**

**(cal\_r\_E || cal\_i\_E || load\_E || store\_E) && jal\_W && Rs\_E == 5'h1f ? 3'b101 : 3'b000,**

**f\_B\_E = (cal\_r\_E || store\_E) && cal\_r\_M && Rt\_E == Rd\_M && Rd\_M != 0 ? 3'b010 :**

**(cal\_r\_E || store\_E) && cal\_i\_M && Rt\_E == Rt\_M && Rt\_M != 0 ? 3'b010 :**

**(cal\_r\_E || store\_E) && jal\_M && Rt\_E == 5'h1f ? 3'b100 :**

**(cal\_r\_E || store\_E) && cal\_r\_W && Rt\_E == Rd\_W && Rd\_W != 0 ? 3'b001 :**

**(cal\_r\_E || store\_E) && cal\_i\_W && Rt\_E == Rt\_W && Rt\_W != 0 ? 3'b001 :**

**(cal\_r\_E || store\_E) && load\_W && Rt\_E == Rt\_W && Rt\_W != 0 ? 3'b011 :**

**(cal\_r\_E || store\_E) && jal\_W && Rt\_E == 5'h1f ? 3'b101 : 3'b000,**

**//数据存储器数据转发**

**f\_Data\_M = store\_M && load\_W && Rt\_M == Rt\_W && Rt\_W !=0 ? 2'b01 :**

**store\_M && jal\_W && Rt\_M == 5'h1f ? 2'b10 : 2'b00;**

**f\_jal\_D** = Jal\_W ;需要持续到WB级，不能内部转发，在WB级存寄存器文件

**总结几个指令的特征：**

**条件跳转并链接指令：blez只跳转, bltzl（likely可能跳转条件无效）跳转或无效, bltzal（link链接）跳转并保存$31, bltzalr跳转并链接, bne只跳转, blezal同bltzal**

在描述上，跳转条件满足，则跳转到标签所在位置，即PC+target，无标签的即PC+8；likely可能跳转，也有可能条件无效；link链接即需要保存指定寄存器的值。

1. 控制器功能

可以类比beq指令新增branch\_控制信号，在GRF模块里也即IF/ID流水级引入该控制信号。如果需要保存寄存器值，则需要添加RW信号。

1. 转发与冒险

考虑到跳转条件的寄存器冒险，转发可以和beq类似。

1. 单独新增操作

保存寄存器的值，此时需要在GRF中用WD写入数据,WA写入地址，以及处理相应的冒险，冒险与Jal类似（保存31号寄存器）。

**J型指令：jal, jalr, jr**

1. 控制器功能

单独添加控制信号。

1. 转发与冒险

对于jalr，还需要保存rd寄存器，PC跳转到RS寄存器的值，RS冒险，与R类型处理相同。

1. 单独新增操作

保存rd寄存器，WA需要增加信号。

**存储取数指令：sw, sb, lb, lh**

1. 控制器功能

注意存取的字节。控制器类比lw,sw。

1. 转发与冒险

冒险类似lw, sw.

1. 单独新增操作

存取的字节操作问题。

四、测试代码

预期测试结果在下侧

CPU-text-1:

# lui, ori, addu, subu, sw, lw, j, jal, jr

.text

lui $2, 0x1234

ori $1, $0, 1234

ori $1, $1, 0xfff

ori $1, $1, 1123

ori $1, $1, 1281

addu $1, $1, $2

addu $1, $1, $2

addu $1, $1, $2

addu $1, $1, $1

nop

nop

nop

lui $1, 0

sw $3, 0($1)

lw $1, 0($1)

sw $1, 0($0)

lw $1, 4($0)

sw $3, 0($1)

lui $2, 0x0000

jal loop

ori $3, $0, 256

addu $3, $3, $2

j end

lui $2, 0

sw $3, 0($2)

loop:

ori $2, $2, 512

subu $3, $2, $0

sw $3, 4($2)

addu $3, $3, $3

sw $3, 4($3)

lw $4, 0($3)

addu $4, $3, $4

jr $ra

nop

end:

@00003000: $ 2 <= 12340000

@00003004: $ 1 <= 000004d2

@00003008: $ 1 <= 00000fff

@0000300c: $ 1 <= 00000fff

@00003010: $ 1 <= 00000fff

@00003014: $ 1 <= 12340fff

@00003018: $ 1 <= 24680fff

@0000301c: $ 1 <= 369c0fff

@00003020: $ 1 <= 6d381ffe

@00003030: $ 1 <= 00000000

@00003034: \*00000000 <= 00000000

@00003038: $ 1 <= 00000000

@0000303c: \*00000000 <= 00000000

@00003040: $ 1 <= 00000000

@00003044: \*00000000 <= 00000000

@00003048: $ 2 <= 00000000

@0000304c: $31 <= 00003054

@00003050: $ 3 <= 00000100

@00003064: $ 2 <= 00000200

@00003068: $ 3 <= 00000200

@0000306c: \*00000204 <= 00000200

@00003070: $ 3 <= 00000400

@00003074: \*00000404 <= 00000400

@00003078: $ 4 <= 00000000

@0000307c: $ 4 <= 00000400

@00003054: $ 3 <= 00000600

@0000305c: $ 2 <= 00000000

CPU-text-2:

# j,jal,jr，beq， blez, bltzal, clz, lb, jalr

.text

lui $1, 0

lui $2, 0xffff

ori $3, $2, 1234

jal loop

nop

loop:

jal end

nop

beq $ra, $0, o

nop

lui $3, 0

addu $3, $3, 0x303c

jalr $1, $3

nop

subu $3, $3, $2

subu $3, $3, $2

bltzal $3, a

nop

end:

o:

ori $3, $0, 6

lui $4, 0

lui $2, 0x0000

addu $2, $3, $2

ori $2, $2, 0x3016

addu $2, $2, $3

jr $2

nop

jal a

nop

sw $ra, 0($4)

a:

addu $31, $31, $31

ori $31, $31, 2

addu $2, $31, $3

blez $2, final

nop

addu $31, $31, $2

final:

lui $1, 0xffff

lui $2, 0x0fff

ori $3, $2, 1234

# bltzal

bltzal $1, Bltzal

nop

Bltzal:

addu $1, $1, $2

bltzal $1, Bltzal2

nop

addu $2, $ra, $2

Bltzal2:

subu $31, $31, $1

# blez

lui $1, 0

lui $2, 0x0fff

ori $3, $2, 1234

blez $1, Blez

nop

Blez:

blez $3, Blez2

nop

Blez2:

addu $3, $3, $3

# lb, sb

lui $1, 0

ori $1, 0x0004

sw $1, 4($1)

lui $2, 0x000f

ori $3, $2, 1234

# sb $3, 5($1)

lb $2, 5($1)

addu $2, $3, $2

# sb $2, 3($1)

lb $1, 3($1)

@00003000: $ 1 <= 00000000

@00003004: $ 2 <= ffff0000

@00003008: $ 3 <= ffff04d2

@0000300c: $31 <= 00003014

@00003014: $31 <= 0000301c

@0000304c: $ 3 <= 00000006

@00003050: $ 4 <= 00000000

@00003054: $ 2 <= 00000000

@00003058: $ 2 <= 00000006

@0000305c: $ 2 <= 00003016

@00003060: $ 2 <= 0000301c

@00003024: $ 3 <= 00000000

@00003028: $ 1 <= 00000000

@0000302c: $ 1 <= 0000303c

@00003030: $ 3 <= 0000303c

@00003034: $ 1 <= 0000303c

@0000303c: $ 3 <= 00000020

@00003040: $ 3 <= ffffd004

@00003044: $31 <= 0000304c

@00003078: $31 <= 00006098

@0000307c: $31 <= 0000609a

@00003080: $ 2 <= 0000309e

@0000308c: $31 <= 00009138

@00003090: $ 1 <= ffff0000

@00003094: $ 2 <= 0fff0000

@00003098: $ 3 <= 0fff04d2

@0000309c: $31 <= 000030a4

@000030a4: $ 1 <= 0ffe0000

@000030b0: $ 2 <= 0fff30a4

@000030b4: $31 <= f00230a4

@000030b8: $ 1 <= 00000000

@000030bc: $ 2 <= 0fff0000

@000030c0: $ 3 <= 0fff04d2

@000030d4: $ 3 <= 1ffe09a4

@000030d8: $ 1 <= 00000000

@000030dc: $ 1 <= 00000004

@000030e0: \*00000008 <= 00000004

@000030e4: $ 2 <= 000f0000

@000030e8: $ 3 <= 000f04d2

@000030ec: $ 2 <= 00000000

@000030f0: $ 2 <= 000f04d2

@000030f4: $ 1 <= 00000000

五、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)答：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 冲突序号 | 冲突类型 | 冲突寄存器 | 前序指令 | 后序指令 |
| 1 | LAW  LAR  WAR  WAL | Rs\Rt | load | Store |
| 2 | R |
| 3 | I |
| 4 | Rs | load | load |
| 5 | Rs\Rt | R | R\I |
| 6 | I |
| 7 | Rs | R | load |
| 8 | Rs | I | load |
| 9 | Rs | R | store |
| 10 | Rs | I | store |
| 11 | JAW | Rs\Rt | R\I\load | Beq\jr |
| 12 | Rs | R\I\load | jr |
| 13 | RAJal | 31 | Jal | R/J/beq/load/store |

分别作相应的转发与暂停，详见转发表格和暂停表格及其信号表达式。