**计算机组成原理**

**课程设计报告**

**学 号\_\_\_\_\_\_\_\_21071003\_\_\_\_\_\_\_\_\_\_\_\_**

**姓 名\_\_\_\_\_\_\_\_高立扬\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**指导教师\_\_\_\_\_\_\_\_\_高明霞\_\_\_\_\_\_\_\_\_\_\_\_\_**

**提交日期\_\_\_\_\_\_\_2023.7\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**成绩评价表**

|  |  |  |
| --- | --- | --- |
| **报告内容** | **报告结构** | **报告最终成绩** |
| **□丰富正确**  **□基本正确**  **□有一些问题**  **□问题很大** | **□完全符合要求**  **□基本符合要求**  **□有比较多的缺陷**  **□完全不符合要求** |  |
| **报告与Project功能一致性** | **报告图表** | **总体评价** |
| **□完全一致**  **□基本一致**  **□基本不一致** | **□符合规范**  **□基本符合规范**  **□有一些错误**  **□完全不正确** |  |

**教师签字:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

目录

[Project1 VerilogHDL完成单周期处理器开发 4](#_Toc20308)

[一． 总体数据通路结构设计图 4](#_Toc11609)

[二． 模块定义 5](#_Toc20684)

[2.1 PC模块 5](#_Toc10462)

[2.2 IM模块 5](#_Toc10393)

[2.3 PC计算模块（以下简称NPC模块） 6](#_Toc25602)

[2.4 GPR模块 7](#_Toc19112)

[2.5 EXT模块 8](#_Toc12804)

[2.6 DM模块 9](#_Toc14391)

[2.7 ALU模块 10](#_Toc1594)

[2.8 M1模块 10](#_Toc31246)

[2.9 M2模块 11](#_Toc17903)

[2.10 M3模块 12](#_Toc26980)

[2.11 Controller模块 13](#_Toc12846)

[三． MIPS-Lite1指令集+LH指令 机器指令描述 14](#_Toc9472)

[四． 测试程序 15](#_Toc28900)

[4.1 MIPS-LITE1指令集测试程序 15](#_Toc15453)

[4.2 新增LH指令测试程序 16](#_Toc14327)

[五． 测试结果 17](#_Toc10445)

[5.1 MARS中测试结果 17](#_Toc29260)

[5.2 modelsim验证结果 18](#_Toc11516)

[5.3 新增指令测试 18](#_Toc13780)

[六． Project1完成后的心得体会 19](#_Toc13632)

[Project2 VerilogHDL 完成多周期处理器开发 19](#_Toc23314)

[一． 总体数据通路结构设计图 19](#_Toc26509)

[二． 模块定义 20](#_Toc18871)

[2.1 PC模块 20](#_Toc18455)

[2.2 IM模块 21](#_Toc27469)

[2.3 NPC模块 22](#_Toc16350)

[2.4 GPR模块 23](#_Toc31748)

[2.5 EXT模块 24](#_Toc31926)

[2.6 DM模块 25](#_Toc29471)

[2.7 ALU模块 26](#_Toc7829)

[2.8 M1模块 26](#_Toc18325)

[2.9 M2模块 27](#_Toc1053)

[2.10 M3模块 28](#_Toc15068)

[2.11 Controller模块 29](#_Toc22726)

[2.12 ALUOUT模块 30](#_Toc25345)

[2.13 AR模块 & BR模块 31](#_Toc22808)

[2.14 DR模块 31](#_Toc13935)

[2.15 IR模块 32](#_Toc9669)

[三．MIPS-Lite2指令集+JALR指令 机器指令描述 32](#_Toc12062)

[四．测试程序 34](#_Toc1205)

[4.1 MIPS-LITE2指令集测试程序 34](#_Toc2042)

[4.2 新增JALR指令测试程序 34](#_Toc10041)

[五．测试结果 35](#_Toc28356)

[5.1 MARS中测试结果 35](#_Toc1787)

[5.2 modelsim验证结果 36](#_Toc5565)

[5.3 新增指令测试 37](#_Toc14194)

[六．Project2完成后的心得体会 37](#_Toc11220)

[Project3 VerilogHDL完成MIPS微系统开发(支持设备与中断) 38](#_Toc14297)

[一．总体数据通路结构设计图 38](#_Toc14137)

[二．模块定义 39](#_Toc29181)

[2.1 PC模块 39](#_Toc28025)

[2.2 IM模块 39](#_Toc13405)

[2.3 NPC模块 40](#_Toc15765)

[2.4 GPR模块 41](#_Toc10869)

[2.5 EXT模块 43](#_Toc9134)

[2.6 DM模块 43](#_Toc11303)

[2.7 ALU模块 44](#_Toc23360)

[2.8 M1模块 45](#_Toc17819)

[2.9 M2模块 46](#_Toc27649)

[2.10 M3模块 47](#_Toc8988)

[2.11 Controller模块 47](#_Toc11992)

[2.12 ALUOUT模块 50](#_Toc10409)

[2.13 AR模块 & BR模块 50](#_Toc5014)

[2.14 DR模块 51](#_Toc31583)

[2.15 IR模块 51](#_Toc30027)

[2.16 M4模块 52](#_Toc13028)

[2.17 TIMER模块 53](#_Toc31342)

[2.18 OUTPUTDEV模块 54](#_Toc23421)

[2.19 BRIDGE模块 54](#_Toc10044)

[2.20 CP0模块 56](#_Toc30065)

[三．MIPS-Lite3指令集 机器指令描述 57](#_Toc15337)

[四．测试程序 58](#_Toc17639)

[4.1 MIPS-LITE3指令集测试程序 58](#_Toc1448)

[五．测试结果 59](#_Toc7529)

[5.1波形图（重点观察倒计时和中断返回） 59](#_Toc16522)

[5.2寄存器结果 60](#_Toc28326)

[六．Project3完成后的心得体会 61](#_Toc22985)

# Project1 VerilogHDL完成单周期处理器开发

## 总体数据通路结构设计图

Project1单周期处理器的总体数据通路结构设计图如图1.所示，本处理器参考了Project1任务书所给的图，支持MIPS-Lite1指令集{MIPS-Lite，addi，addiu， slt，jal，jr}(MIPS-Lite指令集：addu，subu，ori，lw，sw，beq，lui，j)。本处理器的取指模块没有采用大作业中的IFU，而是效仿多周期处理器，将IFU拆解，设计了PC计算模块、PC模块和IM模块，这么做是为了方便做project2和3的时候进行模块迭代，避免大幅度修改增加BUG出现几率，节省DEBUG时间开销。图中M1-3为MUX选择模块，原本可以分别内置在GPR和ALU内，但是为了方便调试，单独拆成了三个模块，直观，且体现了模块化编程的好处，减少了GPR和ALU代码复杂度，更注重于其功能特点。

本project自主编写；在线下检查时，添加的指令为“LH”

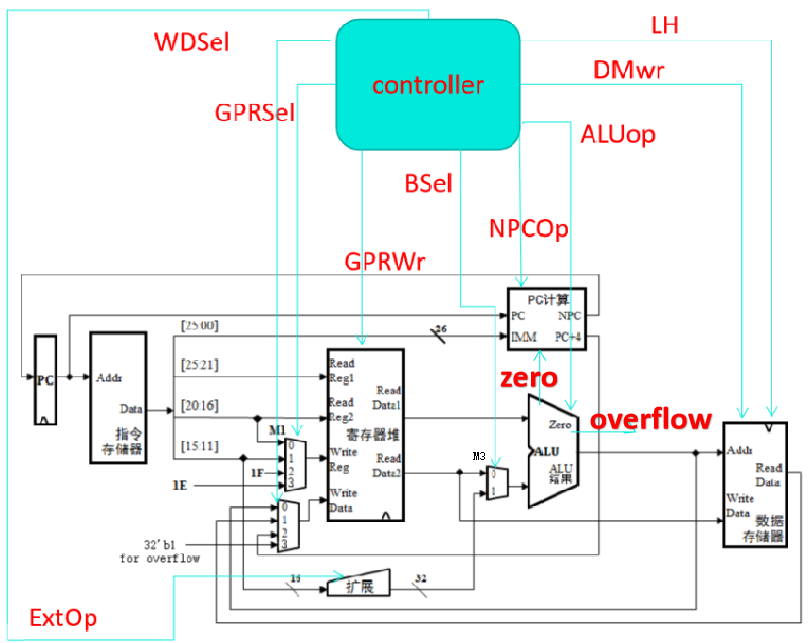


图1.project1 总体数据通路图

## 模块定义

### 2.1 PC模块

#### 2.1.1 基本描述

PC模块是时序性模块，其功能为接收来自于PC计算模块输出的NPC信号，并在时钟信号clk上升沿时向外传输NPC输入的值

#### 2.1.2 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| pcin[31:0] | I | 来源于PC计算模块输入 |
| pcout[31:0] | O | 本模块输出值，和PC计算模块输入值的数值相等 |

#### 2.1.3 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 取地址 | 当clk信号上升沿时，本模块将PC计算模块计算好并输入进来的PC，输出出去（输出到IM模块） |

### 2.2 IM模块

#### 2.2.1 基本描述

IM模块是非时序的。其功能为读取事先写好的十六进制代码程序，存入内部的1kb指令寄存器，指令起始地址本应为0000\_3000h，但由于1kb的寄存器地址最大到32位地址的低10位，因此指令的起始地址在书写代码时可以等效为0000\_0000h。依据输入的PC地址信号，进行取指。小端序存储指令

#### 2.2.2 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| addr[9:0] | I | 来源于PC模块的输出，也就是地址，由于IM模块存储指令的寄存器仅1KB，因此只需要PC的低10位 |
| dout[31:0] | O | 本模块输出值，是地址所对应的32位指令 |

#### 2.2.3 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 取指令 | 根据输入的地址的低10位，在本模块内部的指令寄存器里取得指令，并输出 |

### 2.3 PC计算模块（以下简称NPC模块）

#### 2.3.1 基本描述

NPC模块是非时序性模块。本模块是推进PC地址更新的核心，它会根据当前指令的性质，判断并分析下一条指令的地址，传送给PC模块。对于MIPS-Lite1指令集，NPC会根据beq，j和jal指令，jr指令和此外的指令，进行四种地址分析策略，从而计算出下一条指令的地址。特别地，对于jal指令，NPC模块还会输出PC+4以存入31号寄存器；对于jr指令，NPC模块需要输入GPR输出的读取到rs寄存器的内容，便于跳转。

复位时，本模块锁定PC地址为0000\_3000h

#### 2.3.2 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| imm[25:0] | I | 由IM输出的32位指令分线出低26位而来，用于计算j和jal指令跳转的地址，或计算beq跳转的地址 |
| pcin[31:0] | I | 来源于PC模块的输出，也就是接收目前PC地址，便于计算下一条地址 |
| rd1[31:0] | I | 来源于GPR模块的输出，其内容是rs寄存器内部存储的32位数据 |
| npcop[1:0] | I | 来源于Controller的输出，代表着本模块进行哪一种地址计算逻辑：  00：地址+4推进  01：beq指令跳转  10：j/jal指令跳转  11：jr指令跳转 |
| zero | I | 来源于ALU的输出，用于判断beq指令条件，决定是否跳转，如果为1才会进行beq指令跳转，否则正常pc+4推进 |
| rst | I | 异步复位信号，为1时会将PC地址重置为0000\_3000h，直至rst信号重新变为0 |
| nextpc | O | 计算好的32位地址，输出给PC模块 |
| pc\_4 | O | 输入的pcin，在此基础上+4，专用于jal指令 |

#### 2.3.3 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 异步复位 | 复位信号为1时，pcout输出锁定为0000\_3000h |
| 2 | 判断并计算 | 本模块会先计算出j/jal指令和beq指令跳转后的地址，  beq跳转：pc + 4 + imm<<2  j跳转：{pc+4的高四位, imm26, 00}  然后根据npcop进行如下操作：  00：输出pc+4  01：如果zero=1输出beq跳转地址；否则pc+4  10：j/jal跳转地址  11：输出rd1信号内容 |
| 3 | 地址传递 | 计算好的新地址会传输给PC模块 |

### 2.4 GPR模块

#### 2.4.1 基本描述

本模块是时序性模块。但比较特殊的是，其数据输出并不受到时间信号的影响，仅存储数据时需要等待clk上升沿。本模块是CPU的寄存器堆，共有32个32位的寄存器，各司其职。其中0号寄存器恒为0，无法被修改；31号寄存器一般用来存储jal类指令所需要存储的地址，便于jr指令读取。

#### 2.4.2 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号 |
| regwrite | I | 来源于Controller模块，是寄存器的写入权限，1代表可写 |
| m1out[4:0] | I | 来源于M1模块，是待写入数据的寄存器的编码 |
| m2out[31:0] | I | 来源于M2模块，是待写入m1out指定寄存器的数据 |
| rs[4:0] | I | 来源于IM模块的输出dout[25:21] |
| rt[4:0] | I | 来源于IM模块的输出dout[20:16] |
| busa[31：0] | O | 读出rs对应的寄存器的数值 |
| busb[31:0] | O | 读出rt对应的寄存器的数值 |

#### 2.4.3 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 异步复位 | 复位信号为1时，32个寄存器全置0 |
| 2 | 写入数据 | clk为上升沿的时候，m2out的内容存入m1out解码后对应的寄存器 |
| 3 | 读取数据 | 读取IM模块dout输出对应dout[25:21]和[20:16]解码后对应的寄存器的数值并输出 |

### 2.5 EXT模块

#### 2.5.1基本描述

本模块是非时序模块。本模块的功能是根据信号，对输入的16位数进行32位扩展并输出至目标模块M3

#### 2.5.2 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| imm16[15:0] | I | 来源于IM模块的dout的低16位，代表着I型指令对应的立即数 |
| extop[1:0] | I | 来源于Controller控制本模块扩展立即数的逻辑  00：无符号拓展  01：符号拓展  10：高位拓展（专用于lui指令） |
| out[31:0] | O | 拓展完的数值进行输出 |

#### 2.5.3 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 扩展并输出 | 对立即数进行相应的拓展并输出 |

### 2.6 DM模块

#### 2.6.1 基本描述

本模块是时序模块。本模块是专门存储数据的数据寄存器，容量为1kb。当clk为上升沿的时候，允许写入数据。数据地址从0000\_0000h开始。小端序存储数据

#### 2.6.2 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| addr[9:0] | I | 来源于ALU的输出，代表了本模块将要读写的地址 |
| din[31:0] | I | 来源于GPR的busb输出，代表了本模块将要写入的值 |
| we | I | 来源于Controller的输出，1为DM可写 |
| clk | I | 时钟信号 |
| lh | I | 来源于Controller的输出，1代表当前指令为lh指令 | I |
| dout[31:0] | O | 根据当前指令是否为lh，决定本模块的输出  lh=0 根据addr输出32位数据  lh=1 根据addr输出16位数据符号扩展后的32位数据 |

#### **2.6.3 功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 写入数据 | 写使能有效的时候，当clk为上升沿，根据地址进行数据写入 |
| 2 | 读取数据 | 根据地址和lh指令判断，进行数据读取并输出 |

### **2.7 ALU模块**

#### 2.7.1基本描述

本模块为非时序模块。本模块为运算模块，对输入的两个数，根据控制信号进行相应运算并进行输出。同时在特殊情况下会进行两数相等的判定，或两数相加溢出的判定，并输出对应的信号。

#### **2.7.2 模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| A[31:0] | I | 运算数A |
| B[31:0] | I | 来源于M3模块的输出，运算数B |
| aluctr[2:0] | I | 运算逻辑  000：加法  001：减法  010：按位或运算  011：(A和B符号拓展) A < B ? 1 : 0，针对于slt指令  100：针对于addi指令的加法，会追加判断是否溢出 |
| zero | O | 针对于beq指令的信号，A=B时输出1，否则0 |
| overflow | O | 针对于addi指令的信号，A+B溢出时为1，否则0 |
| out[31:0] | O | 运算结果输出，为A(运算逻辑)B |

#### **2.7.3 功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 计算 | 根据aluctr进行相应逻辑的运算并输出结果 |
| 2 | 判断 | 针对于beq、slt、addi指令分别做出相应的判断，如上文所示 |

### **2.8 M1模块**

#### 2.8.1基本描述

本模块为非时序模块，用于GPR待写入寄存器编码的选择工作。

#### **2.8.2模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| gprsel[1:0] | I | 来源于Controller模块的输出，用于选择GPR待写入寄存器的5位编码，控制着m1out信号的输出数值  00：rt  01：rd  10：1F  11：1E |
| rt[4:0] | I | 来源于IM模块的输出，为dout[20:16] |
| rd[4:0] | I | 来源于IM模块的输出，为dout[15:10] |
| m1out[4:0] | O | 输出数值被gprsel控制 |

#### **2.8.3 功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 多路选择 | 选择GPR待写入寄存器编码 |

### **2.9 M2模块**

#### 2.9.1 基本描述

本模块为非时序模块，用于GPR待写入数据的选择工作。

#### **2.9.2模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| wdsel[1:0] | I | 来源于Controller模块的输出，用于选择GPR待写入寄数据，控制着m2out输出信号的值  00：aluo  01：dmout  10：pc\_4  11：0000\_0001h（专用于addi指令overflow的情况） |
| aluo[31:0] | I | 来源于IM模块的输出，为dout[20:16] |
| dmout[31:0] | I | 来源于IM模块的输出，为dout[15:10] |
| pc\_4[31:0] | I | 来源于NPC模块的输出，为PC+4 |
| m2out[31:0] | O | 输出数值被wdsel控制 |

#### **2.9.3 功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 多路选择 | 选择GPR待写入数据 |

### **2.10 M3模块**

#### 2.10.1 基本描述

本模块为非时序模块，用于ALU运算数B的选择工作

#### **2.10.2模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| bsel | I | 来源于Controller模块的输出，用于ALU运算数B的选择工作，控制着m3out输出信号的值  0：bo  1：imm32 |
| bo[31:0] | I | 来源于GPR模块的输出，为busb |
| imm32[31:0] | I | 来源于EXT模块的输出，为EXT模块的out |
| m3out[31:0] | O | 输出数值被bsel控制 |

#### **2.10.3 功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 多路选择 | ALU运算数B的选择 |

### **2.11 Controller模块**

#### 2.11.1 基本描述

本模块为非时序模块。本模块是整个CPU中的重中之重，犹如人之大脑一般重要。其功能为输入opcode和funct，从而对指令进行译码，输出控制信号，控制几乎CPU中每一个部件，没有了本模块，CPU中的各个模块只能是一盘散沙，可见其重要性所在

本模块中有一个没有用到的信号，名为j，由于controller内部信号书写问题，删除此信号怕出现bug，后续没有对j信号进行删除，予以保留（不影响模块正确执行所有规定指令），在此特殊说明

#### **2.11.2 模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| opcode[5:0] | I | 来源于IM模块的输出，dout的高六位 |
| funct[5:0] | I | 来源于IM模块的输出，dout的低六位 |
| overflow | I | 来源于ALU模块的输出，是对addi指令运算结果是否溢出的判断信号，1代表溢出 |
| j | O | 废弃信号，怕删除出bug，予以保留 |
| aluop[2:0] | O | 输出到ALU模块，控制运算逻辑 |
| gprsel[1:0] | O | 输出到M1模块，选择待写入数据的寄存器的五位编码 |
| gprwr | O | 输出到GPR模块，控制GPR的写入权限 |
| extop[1:0] | O | 输出到EXT模块，控制其拓展逻辑 |
| wdsel[1:0] | O | 输出到M2模块，选择写入GPR目标寄存器的数值 |
| npcop[1:0] | O | 输出到NPC模块，选择PC地址运算逻辑 |
| dmwr | O | 输出到DM模块，控制DM的写入权限 |
| bsel | O | 输出到M3模块，选择输入ALU作为运算数B的数值 |
| lh | O | 输出到DM模块，选择dmout输出的数值 |

#### **2.11.3 功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 译码产生控制信号 | 根据opcode和funct进行译码，获得上文所示的各种重要控制信号 |

## MIPS-Lite1指令集+LH指令 机器指令描述

表1.MIPS-LITE1指令+LH指令

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 助记符 | opcode | funct | 指令功能 | |
| 1 | addu | 000000 | 100001 | 两个来自寄存器的数无符号加法 | GPR[rd] ← GPR[rs] + GPR[rt] |
| 2 | subu | 000000 | 100011 | 两个来自寄存器的数无符号减法 | GPR[rd] ← GPR[rs] - GPR[rt] |
| 3 | ori | 001101 | - | 两个来自寄存器的数逻辑或运算 | GPR[rt] ← GPR[rs] or immediate |
| 4 | lw | 100011 | - | 读取DM中一个字到指定寄存器中 | GPR[rt] ← memory[GPR[base] + offset] |
| 5 | sw | 101011 | - | 指定寄存器中的字内容写到DM中 | memory[GPR[base] + offset] ← GPR[rt] |
| 6 | beq | 000100 | - | 两数相等则跳转 | if GPR[rs] = GPR[rt] then branch |
| 7 | lui | 001111 | - | 16位立即数高位拓展 | GPR[rt] ← immediate || 016 |
| 8 | j | 000010 | - | 无条件跳转 | PC ← PCGPRLEN-1..28 || instr\_index || 02 |
| 9 | addi | 001000 | - | 加立即数（符号拓展，有溢出检验） | temp ← (GPR[rs]31||GPR[rs]31..0) + sign\_extend(immediate)  if temp32 ≠ temp31 then  SignalException(IntegerOverflow)  else  GPR[rt] ← temp  endif |
| 10 | addiu | 001001 | - | 加立即数（符号拓展，无溢出检验） | GPR[rt] ← GPR[rs] + immediate |
| 11 | slt | 000000 | 101010 | 小于则置1 | GPR[rd] ← (GPR[rs] < GPR[rt]) |
| 12 | jal | 000011 | - | 跳转并链接（PC+4存到$31） | GPR[31]← PC + 8  PC ← PCGPRLEN-1..28 || instr\_index || 02 |
| 13 | jr | 000000 | 001000 | 跳转到寄存器存储的地址 | PC ← GPR[rs] |
| 14 | lh | 100001 | - | 读取DM中半个字（符号拓展存到GPR寄存器） | GPR[rt] ← memory[GPR[base] + offset] |

## **测试程序**

### 4.1 MIPS-LITE1指令集测试程序

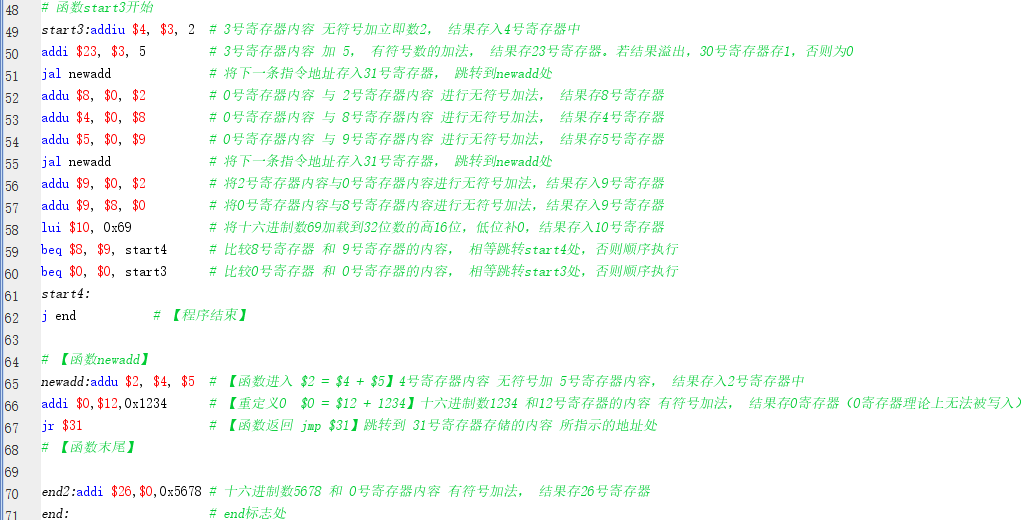
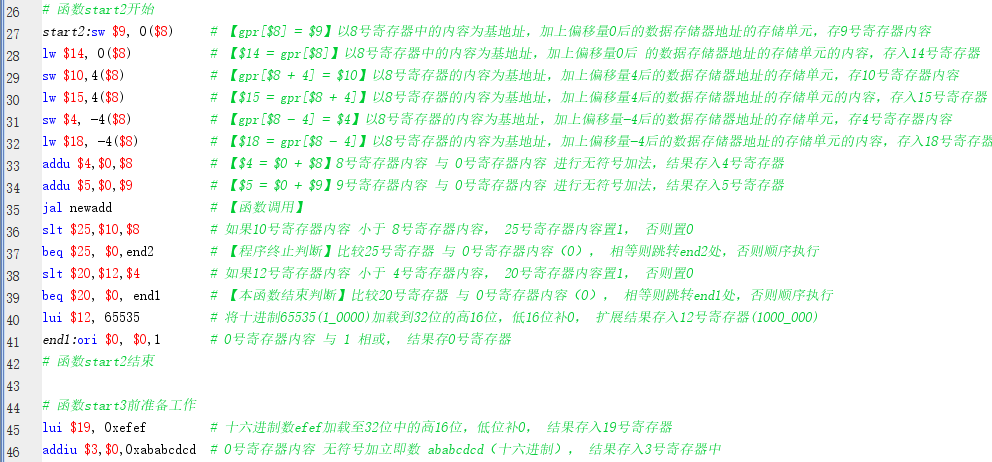
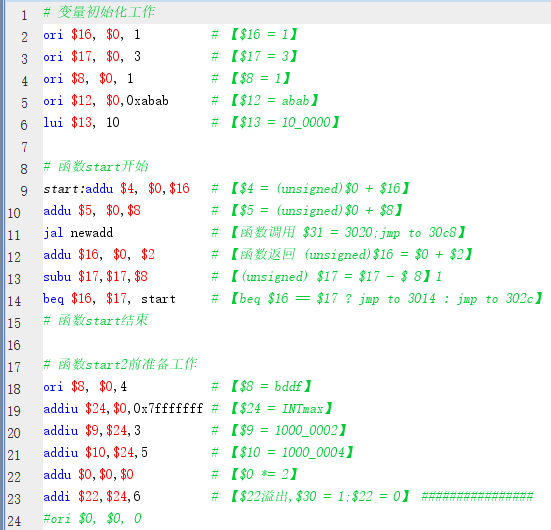


图2-4.针对于MIPS-LITE1指令集的测试程序

### **4.2 新增LH指令测试程序**



图5.新增指令测试程序

说明：在4.1所示程序（对应提交文件中的p1-test2.asm）的第64行新增lh $22,8($0)指令，其余指令不变。对于读取半个字的两种情况（高16位或低16位），设计了两套测试程序，第二套程序即“lh $22,6($0)”，offset为8对应读取DM第三个字的低半字，offset为6对应读取DM第二个字的高半字。两个测试程序分别对应提交文件中的p1-test-lh.asm和p1-test-lh2.asm。后续结果测试时，只需关注22号寄存器存储的数值是否正确即可

## **测试结果**

### 5.1 MARS中测试结果

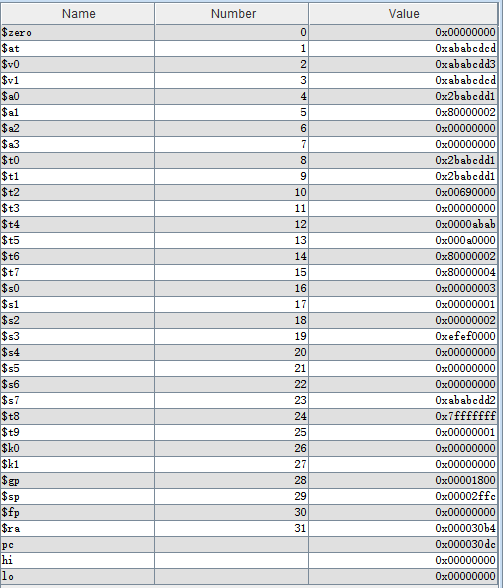


图6.MARS中GPR最终结果



图7.MARS中DM最终结果

特殊说明：addi指令运算结果若溢出，在MARS中会报错导致程序无法跑完，因此临时替换addi语句为“ori $0, $0, 0”可以在不影响所有GPR和DM数据的情况下跑通程序，图6.的结果按MIPS-LITE1的要求，$30最终存储的值应当为0000\_0001，这将在下文着重验证

$28和$29分别为global pointer和stack pointer，在MARS运行时会有数值，而对于MIPS-LITE1简单指令集，不会涉及到这两个寄存器的写入访问，因此在modelsim波形仿真时，这两个寄存器的最终结果应当为0

### **5.2 modelsim验证结果**

#### 5.2.1 波形图（重点检验溢出）

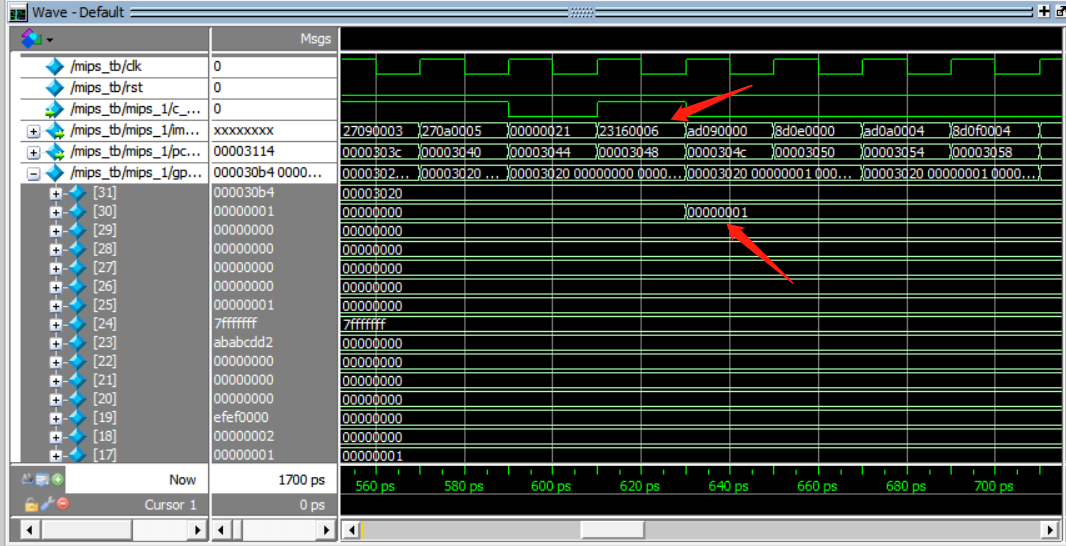


图8.overflow的判断和$30存储0000\_0001h数据

如图8.所示，23160006正是指令“addi $22,$24,6”，由图可知，执行本指令的时候24号寄存器存储的是32位二进制数正数最大值，+6肯定会发生溢出，而后续的波形表示，不仅判断出了指令的溢出，而且还将0000\_0001h存入了30号寄存器

#### **5.2.2 DM和GPR最终结果**



图9.modelsim中GPR最终结果



图10.modelsim中DM最终结果

结果正确，说明CPU设计合理，已经能正确执行MIPS-LITE1指定的所有指令

### **5.3 新增指令测试**

#### 5.3.1 MARS中测试结果



图11-12.MARS中22号寄存器存储结果，分别对应offset=8和6

#### **5.3.2 modelsim中测试结果**

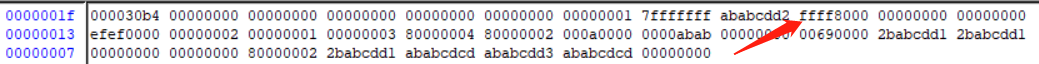


图13-14.modelsim中22号寄存器的存储结果

结果正确，说明针对新指令的设计合理，能正确执行

## **Project1完成后的心得体会**

这是自数字逻辑实验以来，又一次使用Verilog语言。因为有大作业的基础，project1的编写并不难，再加上有学习通中的教程，我很快就写完了所有模块的代码，学会了用testbench，但是调试耗费了我很多的时间。经过数小时的调试，我将常见错误归纳为：Controller控制信号判断和编写有误、npc指令传递错误、顶层连线错误（包括信号连错和位宽错误）、testbench中信号书写有误。在调试的时候，我掌握了很多诀窍和方法，使得我能快速定位到错误所在，在后续两个project中节省了不少的时间。

起初，我本来设计好了IFU模块，但是想到后续两个project是多周期，又看到project1的例图中的模块也和多周期的模块极度相似，我就尝试将IFU拆开进行编写，并自主设计了M1-3模块，体验到了模块化对顶层连线、代码理解和调试带来的好处。进行设计的时候，我还手画了顶层设计图，进一步加深了对单周期数据通路的理解。

在检查作业时，通过老师的提示，我发现了针对addi指令进行$30置1时，只需要将信号传入GPR，在存储的时候进行if else判断即可完成任务，并不需要单独在M1和M2模块占用一个多路选择信号，还要在controller里判断多个控制信号的逻辑，十分占用资源。不过我在编写GPR的时候，因为配合M1和M2的模块化设计，导致了不好按照上述思路单独修改GPR，况且通过溢出信号输入GPR的方法来决定存储加法结果还是溢出后的1，本质上也是一个二路选择器，所以我对原来的代码予以保留，没有进行修改，但是我已经理解了上述的思路，并且在project2检查的时候正确复述了出来。

本次添加指令环节，我的任务是添加LH指令，一开始因为紧张，我一时不知道应该从哪个模块开始入手修改，后来我开始对Controller进行分析，并画图辅助理解，快速完成了模块修改。接下来就要进行测试程序编写了，编写完成后导出指令并进行验证，我用时23分钟及时完成了任务，并在课后及时归纳了添加指令的分析步骤，便于后续project2检查节省时间。

# Project2 VerilogHDL 完成多周期处理器开发

## 总体数据通路结构设计图

project2的总体数据通路结构设计如图15.。本处理器参考了Project2任务书所给的图，支持MIPS-Lite2指令集{MIPS-Lite1，lb，sb}。本处理器的模块完美继承了project1中的模块，使得在进行多周期模块修改时，不仅方便且节省时间，而且尽最大可能减少了代码修改行数，进而避免了修改代码导致不必要的BUG出现，节省了DEBUG时间开销。在继承p1中的模块后，任务只剩下了修改个别模块（只需添加几行代码），新增图中的时序性寄存器模块，调整顶层连接，最后调试即可。状态机参考的同样是PPT当中的10个状态设计，将在下文Controller模块着重介绍。

本project自主编写；在线下检查时，添加的指令为“JALR”

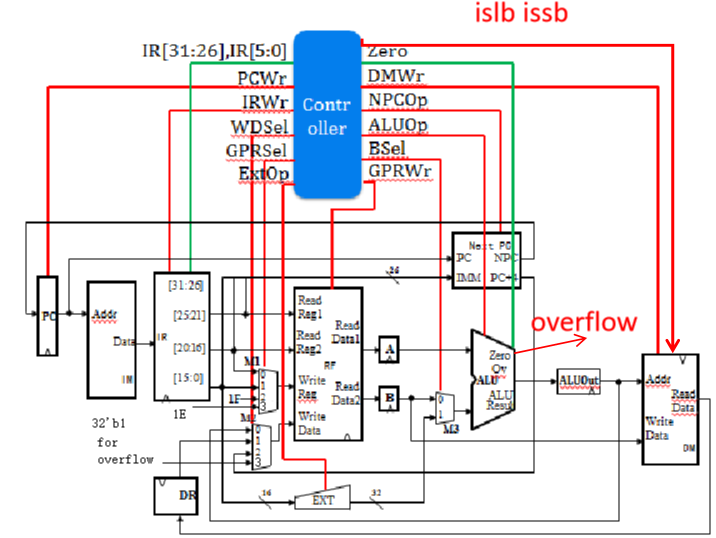


图15.project2总体数据通路结构设计图

## 模块定义

### 2.1 PC模块

#### 2.1.1 基本描述

PC模块是时序性模块，其功能为接收来自于PC计算模块输出的NPC信号，并在时钟信号clk上升沿时向外传输NPC输入的值

#### 2.1.2 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| pcin[31:0] | I | 来源于PC计算模块输入 |
| pcwr | I | PC模块写入权限，1为可写 |
| pcout[31:0] | O | 本模块输出值，和PC计算模块输入值的数值相等 |

#### 2.1.3 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 取地址 | 当clk信号上升沿时，如果PC可写，本模块将NPC模块计算好并输入进来的PC，输出出去（输出到IM模块） |

### 2.2 IM模块

#### 2.2.1 基本描述

IM模块是非时序的。其功能为读取事先写好的十六进制代码程序，存入内部的1kb指令寄存器，指令起始地址本应为0000\_3000h，但由于1kb的寄存器地址最大到32位地址的低10位，因此指令的起始地址在书写代码时可以等效为0000\_0000h。依据输入的PC地址信号，进行取指。小端序存储指令

#### 2.2.2 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| addr[9:0] | I | 来源于PC模块的输出，也就是地址，由于IM模块存储指令的寄存器仅1KB，因此只需要PC的低10位 |
| dout[31:0] | O | 本模块输出值，是地址所对应的32位指令 |

#### 2.2.3 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 取指令 | 根据输入的地址的低10位，在本模块内部的指令寄存器里取得指令，并输出 |

### 2.3 NPC模块

#### 2.3.1 基本描述

NPC模块是非时序性模块。本模块是推进PC地址更新的核心，它会根据当前指令的性质，判断并分析下一条指令的地址，传送给PC模块。对于MIPS-Lite2指令集，NPC会根据beq，j和jal指令，jr和jalr指令和此外的指令，进行四种地址分析策略，从而计算出下一条指令的地址。特别地，对于jal指令，NPC模块还会输出PC+4以存入31号寄存器；对于jr和jalr指令，NPC模块需要输入GPR输出的读取到rs寄存器的内容，便于跳转。

值得一提的是，本CPU为多周期，按照10个状态设计状态机，因此第n周期读取到的指令将会在第n+1个周期进行译码和计算等工作，也就是延后一周期，但是NPC对PC的推进并没有延后一周期。这导致了在进行jal和jalr指令时，如果PC\_4信号还按照PC+4来计算的话，$31存入的地址实际上为PC+8，因为指令的执行实际上比PC的推进滞后。解决这种错误的办法是pc\_4传递pc而不是pc+4

复位时，本模块锁定PC地址为0000\_3000h

#### 2.3.2 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| imm[25:0] | I | 由IM输出的32位指令分线出低26位而来，用于计算j和jal指令跳转的地址，或计算beq跳转的地址 |
| pcin[31:0] | I | 来源于PC模块的输出，也就是接收目前PC地址，便于计算下一条地址 |
| rd1[31:0] | I | 来源于GPR模块的输出，其内容是rs寄存器内部存储的32位数据 |
| npcop[1:0] | I | 来源于Controller的输出，代表着本模块进行哪一种地址计算逻辑：  00：传输PC  01：beq指令跳转  10：j/jal指令跳转  11：jr和jalr指令跳转 |
| zero | I | 来源于ALU的输出，用于判断beq指令条件，决定是否跳转，如果为1才会进行beq指令跳转，否则正常pc+4推进 |
| rst | I | 异步复位信号，为1时会将PC地址重置为0000\_3000h，直至rst信号重新变为0 |
| nextpc | O | 计算好的32位地址，输出给PC模块 |
| pc\_4 | O | 输入的pcin，在此基础上+4，专用于jal指令 |

#### 2.3.3 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 异步复位 | 复位信号为1时，pcout输出锁定为0000\_3000h |
| 2 | 判断并计算 | 本模块会先计算出j/jal指令和beq指令跳转后的地址，  beq跳转：pc + 4 + imm<<2  j跳转：{pc+4的高四位, imm26, 00}  然后根据npcop进行如下操作：  00：输出pc  01：如果zero=1输出beq跳转地址；否则pc+4  10：j/jal跳转地址  11：输出rd1信号内容 |
| 3 | 地址传递 | 计算好的新地址会传输给PC模块 |

### 2.4 GPR模块

#### 2.4.1 基本描述

本模块是时序性模块。但比较特殊的是，其数据输出并不受到时间信号的影响，仅存储数据时需要等待clk上升沿。本模块是CPU的寄存器堆，共有32个32位的寄存器，各司其职。其中0号寄存器恒为0，无法被修改；31号寄存器一般用来存储jal类指令所需要存储的地址，便于jr指令读取。

#### 2.4.2 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号 |
| regwrite | I | 来源于Controller模块，是寄存器的写入权限，1代表可写 |
| m1out[4:0] | I | 来源于M1模块，是待写入数据的寄存器的编码 |
| m2out[31:0] | I | 来源于M2模块，是待写入m1out指定寄存器的数据 |
| rs[4:0] | I | 来源于IR模块的输出irout[25:21] |
| rt[4:0] | I | 来源于IR模块的输出irout[20:16] |
| busa[31：0] | O | 读出rs对应的寄存器的数值 |
| busb[31:0] | O | 读出rt对应的寄存器的数值 |

#### 2.4.3 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 异步复位 | 复位信号为1时，32个寄存器全置0 |
| 2 | 写入数据 | clk为上升沿的时候，m2out的内容存入m1out解码后对应的寄存器 |
| 3 | 读取数据 | 读取IR模块irout输出对应irout[25:21]和[20:16]解码后对应的寄存器的数值并输出 |

### 2.5 EXT模块

#### 2.5.1基本描述

本模块是非时序模块。本模块的功能是根据信号，对输入的16位数进行32位扩展并输出至目标模块M3

#### 2.5.2 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| imm16[15:0] | I | 来源于IR模块的irout的低16位，代表着I型指令对应的立即数 |
| extop[1:0] | I | 来源于Controller控制本模块扩展立即数的逻辑  00：无符号拓展  01：符号拓展  10：高位拓展（专用于lui指令） |
| out[31:0] | O | 拓展完的数值进行输出 |

#### 2.5.3 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 扩展并输出 | 对立即数进行相应的拓展并输出 |

### 2.6 DM模块

#### 2.6.1 基本描述

本模块是时序模块。本模块是专门存储数据的数据寄存器，容量为1kb。当clk为上升沿的时候，允许写入数据。数据地址从0000\_0000h开始。小端序存储数据

根据islb和issb信号会进行字节读写，由于project3没有再新增针对于CPU内部的指令，而是新增了三个和外设交互的指令，因此在sb存储环节没有单独设计新的多路选择模块，而是选择集成在了模块代码内部

#### 2.6.2 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| addr[9:0] | I | 来源于ALU的输出，代表了本模块将要读写的地址 |
| din[31:0] | I | 来源于GPR的busb输出，代表了本模块将要写入的值 |
| we | I | 来源于Controller的输出，1为DM可写 |
| clk | I | 时钟信号 |
| islb | I | 来源于Controller的输出，1代表当前指令为lb指令 |
| issb | I | 来源于Controller的输出，1代表当前指令为sb指令 |
| dout[31:0] | O | 输出数据 |

#### **2.6.3 功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 写入数据 | 写使能有效的时候，当clk为上升沿，根据sb指令判断，再根据地址进行数据写入 |
| 2 | 读取数据 | 根据地址和lb指令判断，进行数据读取并输出 |

### **2.7 ALU模块**

#### 2.7.1基本描述

本模块为非时序模块。本模块为运算模块，对输入的两个数，根据控制信号进行相应运算并进行输出。同时在特殊情况下会进行两数相等的判定，或两数相加溢出的判定，并输出对应的信号。

#### **2.7.2 模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| A[31:0] | I | 运算数A |
| B[31:0] | I | 来源于M3模块的输出，运算数B |
| aluctr[2:0] | I | 运算逻辑  000：加法  001：减法  010：按位或运算  011：(A和B符号拓展) A < B ? 1 : 0，针对于slt指令  100：针对于addi指令的加法，会追加判断是否溢出 |
| zero | O | 针对于beq指令的信号，A=B时输出1，否则0 |
| overflow | O | 针对于addi指令的信号，A+B溢出时为1，否则0 |
| out[31:0] | O | 运算结果输出，为A(运算逻辑)B |

#### **2.7.3 功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 计算 | 根据aluctr进行相应逻辑的运算并输出结果 |
| 2 | 判断 | 针对于beq、slt、addi指令分别做出相应的判断，如上文所示 |

### **2.8 M1模块**

#### 2.8.1基本描述

本模块为非时序模块，用于GPR待写入寄存器编码的选择工作。

#### **2.8.2模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| gprsel[1:0] | I | 来源于Controller模块的输出，用于选择GPR待写入寄存器的5位编码，控制着m1out信号的输出数值  00：rt  01：isjalr==1 ？ $31 ： rd  10：1F  11：1E |
| isjalr | I | 来源于Controller的输出，1代表执行的是本条指令 |
| rt[4:0] | I | 来源于IM模块的输出，为dout[20:16] |
| rd[4:0] | I | 来源于IM模块的输出，为dout[15:10] |
| m1out[4:0] | O | 输出数值被gprsel控制 |

#### **2.8.3 功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 多路选择 | 选择GPR待写入寄存器编码 |

### **2.9 M2模块**

#### 2.9.1 基本描述

本模块为非时序模块，用于GPR待写入数据的选择工作。

#### **2.9.2模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| wdsel[1:0] | I | 来源于Controller模块的输出，用于选择GPR待写入寄数据，控制着m2out输出信号的值  00：aluo  01：dmout  10：pc\_4  11：0000\_0001h（专用于addi指令overflow的情况） |
| aluo[31:0] | I | 来源于IM模块的输出，为dout[20:16] |
| dmout[31:0] | I | 来源于IM模块的输出，为dout[15:10] |
| pc\_4[31:0] | I | 来源于NPC模块的输出，为PC+4 |
| m2out[31:0] | O | 输出数值被wdsel控制 |

#### **2.9.3 功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 多路选择 | 选择GPR待写入数据 |

### **2.10 M3模块**

#### 2.10.1 基本描述

本模块为非时序模块，用于ALU运算数B的选择工作

#### **2.10.2模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| bsel | I | 来源于Controller模块的输出，用于ALU运算数B的选择工作，控制着m3out输出信号的值  0：bo  1：imm32 |
| bo[31:0] | I | 来源于GPR模块的输出，为busb |
| imm32[31:0] | I | 来源于EXT模块的输出，为EXT模块的out |
| m3out[31:0] | O | 输出数值被bsel控制 |

#### **2.10.3 功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 多路选择 | ALU运算数B的选择 |

### **2.11 Controller模块**

#### 2.11.1 基本描述

本模块为非时序模块。本模块是整个CPU中的重中之重，犹如人之大脑一般重要。其功能为输入opcode和funct，从而对指令进行译码，输出控制信号，控制几乎CPU中每一个部件，没有了本模块，CPU中的各个模块只能是一盘散沙，可见其重要性所在

本模块针对于多周期CPU新增了不少控制信号，并且删除了project1中未删除的j信号；同时加入了本project的重中之重：状态机。本CPU采用10个状态的状态机如图16.

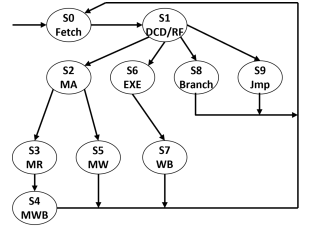


图16.状态图

S0为取指，S1为译码，是MIPS-LITE2指令集中每一条指令必经之路。

S2为针对于DM读写相关指令的入口，也就是lw,lb,sw,sb；其中lw和lb会进入 S3状态先对DM进行读取，之后进入S4状态进行GPR回写；sw和sb会进入S5状态访存DM

S6为针对于ALU运算相关指令的入口，也就是addu,subu,ori,addi,addiu,lui,slt，运算后会进入S7状态进行GPR回写

S8为针对于分支相关指令的入口，也就是beq指令

S9为针对于j型指令的入口，也就是j,jal,jr,jalr指令，本状态跳转和回写同时进行

由图可知，S4,S5,S7,S8,S9执行完毕后，重回S0状态取指，完成一个周期

rst复位会让状态机的状态锁定到S0

#### **2.11.2 模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| opcode[5:0] | I | 来源于IM模块的输出，dout的高六位 |
| funct[5:0] | I | 来源于IM模块的输出，dout的低六位 |
| overflow | I | 来源于ALU模块的输出，是对addi指令运算结果是否溢出的判断信号，1代表溢出 |
| clk | I | 时钟信号 |
| rst | I | 复位信号 |
| zero | I | 来源于ALU模块的输出，代表了两数是否相等 |
| aluop[2:0] | O | 输出到ALU模块，控制运算逻辑 |
| gprsel[1:0] | O | 输出到M1模块，选择待写入数据的寄存器的五位编码 |
| gprwr | O | 输出到GPR模块，控制GPR的写入权限 |
| extop[1:0] | O | 输出到EXT模块，控制其拓展逻辑 |
| wdsel[1:0] | O | 输出到M2模块，选择写入GPR目标寄存器的数值 |
| npcop[1:0] | O | 输出到NPC模块，选择PC地址运算逻辑 |
| dmwr | O | 输出到DM模块，控制DM的写入权限 |
| bsel | O | 输出到M3模块，选择输入ALU作为运算数B的数值 |
| pcwr | O | 输出到PC模块，决定PC是否写入来自NPC的输入 |
| irwr | O | 输出到IR模块，决定IR是否写入来自IM的输入 |
| islb | O | 输出到DM模块，决定DM的输出 |
| issb | O | 输出到DM模块，决定DM的写入 |
| isjalr | O | 输出到M1模块，参与M1的多路选择 |

#### **2.11.3 功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 译码产生控制信号 | 根据opcode和funct以及zero和overflow进行译码，获得上文所示的各种重要控制信号 |
| 2 | 状态机推进 | 推进状态机，完成多周期 |
| 3 | 复位 | 重置状态机 |

### **2.12 ALUOUT模块**

#### 2.12.1 基本描述

本模块为时序性模块。本模块接收ALU的输出结果，当时钟上升沿时，输出相应结果

#### **2.12.2 模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| aluoutin[31:0] | I | 来源于ALU模块的输出，是两数相运算的结果 |
| aluoutout[31:0] | O | 输出aluoutin |

#### **2.12.3 功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 传递数据 | clk为上升沿的时候，传递ALU的输出 |

### **2.13 AR模块 & BR模块**

#### 2.13.1基本描述

AR和BR模块均为时序性模块，类似于ALUOUT模块，也是传递信息用的，传递的是GPR读取到的busa和busb。由于二者性质完全相同，仅仅模块名和信号名为了区分才不同，因此下面只介绍AR，BR同理

#### **2.13.2模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| arin[31:0] | I | 来源于GPR模块的输出，也就是busa |
| arout[31:0] | O | 输出arin |

#### **2.13.3 功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 传递数据 | clk为上升沿的时候，传递GPR的输出 |

### **2.14 DR模块**

#### 2.14.1基本描述

DR模块为时序性模块，用于传递DM的输出

#### **2.14.2模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| drin[31:0] | I | 来源于DM模块的输出，也就是dmout |
| drout[31:0] | O | 输出dmout |

#### **2.14.3 功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 传递数据 | clk为上升沿的时候，传递DM的输出 |

### **2.15 IR模块**

#### 2.15.1基本描述

IR模块为时序性模块，用于传递IM的输出

#### **2.15.2模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| irin[31:0] | I | 来源于IM模块的输出，也就是dout |
| irwr | I | 写入权限 |
| irout[31:0] | O | 输出dout |

#### **2.15.3 功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 传递数据 | clk为上升沿的时候，且irwr为1时，传递DM的输出 |

## 三．MIPS-Lite2指令集+JALR指令 机器指令描述

表2.MIPS-LITE2指令+JALR指令

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 助记符 | opcode | funct | 指令功能 | |
| 1 | addu | 000000 | 100001 | 两个来自寄存器的数无符号加法 | GPR[rd] ← GPR[rs] + GPR[rt] |
| 2 | subu | 000000 | 100011 | 两个来自寄存器的数无符号减法 | GPR[rd] ← GPR[rs] - GPR[rt] |
| 3 | ori | 001101 | - | 两个来自寄存器的数逻辑或运算 | GPR[rt] ← GPR[rs] or immediate |
| 4 | lw | 100011 | - | 读取DM中一个字到指定寄存器中 | GPR[rt] ← memory[GPR[base] + offset] |
| 5 | sw | 101011 | - | 指定寄存器中的字内容写到DM中 | memory[GPR[base] + offset] ← GPR[rt] |
| 6 | beq | 000100 | - | 两数相等则跳转 | if GPR[rs] = GPR[rt] then branch |
| 7 | lui | 001111 | - | 16位立即数高位拓展 | GPR[rt] ← immediate || 016 |
| 8 | j | 000010 | - | 无条件跳转 | PC ← PCGPRLEN-1..28 || instr\_index || 02 |
| 9 | addi | 001000 | - | 加立即数（符号拓展，有溢出检验） | temp ← (GPR[rs]31||GPR[rs]31..0) + sign\_extend(immediate)  if temp32 ≠ temp31 then  SignalException(IntegerOverflow)  else  GPR[rt] ← temp  endif |
| 10 | addiu | 001001 | - | 加立即数（符号拓展，无溢出检验） | GPR[rt] ← GPR[rs] + immediate |
| 11 | slt | 000000 | 101010 | 小于则置1 | GPR[rd] ← (GPR[rs] < GPR[rt]) |
| 12 | jal | 000011 | - | 跳转并链接（PC+4存到$31） | GPR[31]← PC + 8  PC ← PCGPRLEN-1..28 || instr\_index || 02 |
| 13 | jr | 000000 | 001000 | 跳转到寄存器存储的地址 | PC ← GPR[rs] |
| 14 | lb | 100000 | - | 读取DM中的字节 | GPR[rt] ← memory[GPR[base] + offset] |
| 15 | sb | 101000 | - | 向DM中存储字节 | memory[GPR[base] + offset] ← GPR[rt] |
| 16 | jalr | 000000 | 001001 | 跳转到rs中存储的地址，并将PC+4存储到rd中，如果没有指定rd，则默认为31号寄存器 | GPR[rd] ← return\_addr, PC ← GPR[rs] |

## 四．测试程序

### 4.1 MIPS-LITE2指令集测试程序

本测试程序和MIPS-LITE1的测试程序相比，仅仅在19-25行有改变（当然也不含lh指令），其余代码不变，因此下面只展示改变的代码。由于lb和sb和lw，sw相通，因此没有做注释

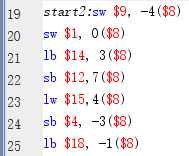


图17.相对于MIPS-LITE1中测试程序改变的全部代码

### **4.2 新增JALR指令测试程序**



图18-19.新增指令

本此测试同样分了两种情况，来测试jalr指令存储pc+4的默认情况和指定存储到哪个寄存器的情况，在后续的验证中也需要特别注意相应寄存器的存储

## **五．测试结果**

### 5.1 MARS中测试结果

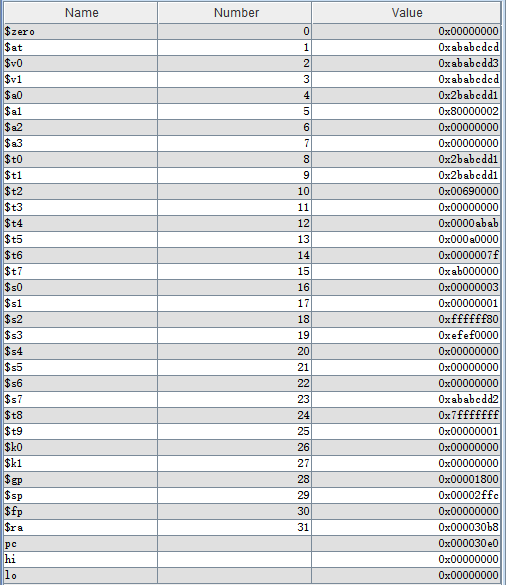


图19.MARS中GPR最终结果



图20.MARS中DM最终结果

特殊说明：addi指令运算结果若溢出，在MARS中会报错导致程序无法跑完，因此临时替换addi语句为“ori $0, $0, 0”可以在不影响所有GPR和DM数据的情况下跑通程序，图6.的结果按MIPS-LITE1的要求，$30最终存储的值应当为0000\_0001，这将在下文着重验证

$28和$29分别为global pointer和stack pointer，在MARS运行时会有数值，而对于MIPS-LITE1简单指令集，不会涉及到这两个寄存器的写入访问，因此在modelsim波形仿真时，这两个寄存器的最终结果应当为0

### **5.2 modelsim验证结果**

#### 5.2.1 波形图（重点检验溢出）

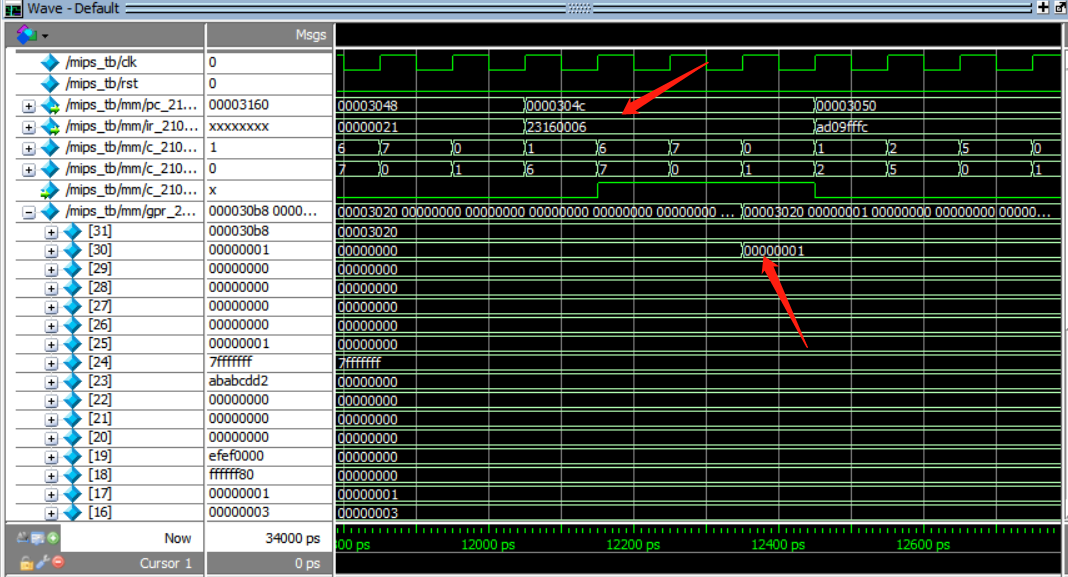


图21.overflow的判断和$30存储0000\_0001h数据

如图21.所示，23160006正是指令“addi $22,$24,6”，由图可知，执行本指令的时候24号寄存器存储的是32位二进制数正数最大值，+6肯定会发生溢出，而后续的波形表示，不仅判断出了指令的溢出，而且还将0000\_0001h存入了30号寄存器

#### **5.2.2 DM和GPR最终结果**



图22.modelsim中GPR最终结果

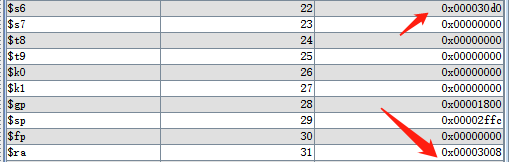


图23.modelsim中DM最终结果

结果正确，说明CPU设计合理，已经能正确执行MIPS-LITE1指定的所有指令

### **5.3 新增指令测试**

#### 5.3.1 MARS中测试结果



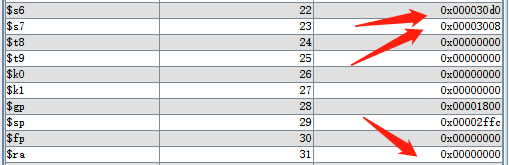
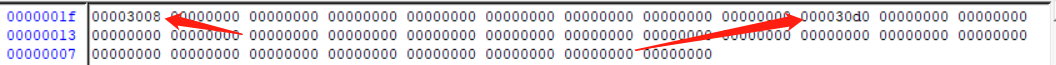


图24-25.MARS中22号寄存器存储结果，分别对应不指定rd和指定rd

#### **5.3.2 modelsim中测试结果**



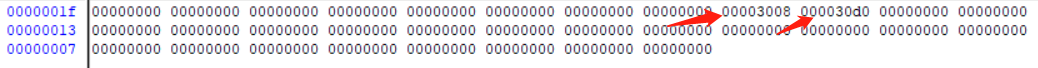


图26-27.modelsim中22号和23号寄存器的存储结果，其中23号寄存器为指定的寄存器

结果正确，说明针对新指令的设计合理，能正确执行

## **六．Project2完成后的心得体会**

由于project1编写时，本着可以迭代到project2的理念，进行了模块化编程，所以在进行project2编写的时候费时极少，基本耗时都花在了状态机的理解和书写上。多周期对于单周期来说，重中之重就在于controller模块的修改，不仅要添加很多信号，在信号翻译时还需要注意信号是否是时序性，还要兼顾状态机的书写，考虑进各种转移情况。但是程序写好后，调试却用了很久很久。

由于对状态机的理解尚停留在书本的范畴，本次实操完毕在验证时，我对着延迟执行的指令疑惑了很久，再加上jal指令PC+4变为PC+8的问题，我一直以为是我代码设计和书写出现了问题，于是我开始画图辅助理解，尝试修改testbench，都没有任何效果。本来萌生过修改pc\_4信号从pc+4变为pc，但是当时的我认为这是面向结果编程，并不会解决实质性问题，所以没有修改。直到第二天和同学交流才发现，原来我的想法是正确的，只不过是对状态机的理解还不够透彻，没有真正清除每一次clk推进，哪些模块都干了些什么事情，经过快速修改后，果然解决了问题。这也同样说明了理论需要与实践结合，不能将自己的水平停留在书本上，一定要亲自动手，收获更宝贵的财富。

在检查时，我的任务是添加jalr指令。吸取了第一次检查的教训，我有条不紊地进行修改和调试，20分钟不到就完成了任务，然而没有考虑到jalr还有不指定rd的默认情况，随后我又进行了修改，差点超时。这警示我不能浮躁，尽管准备充分，也应当处处小心谨慎。

# Project3 VerilogHDL完成MIPS微系统开发(支持设备与中断)

## 一．总体数据通路结构设计图

project3的总体数据通路结构设计如图28.。本处理器参考了计算机组成原理PPT5-5所给的图，支持MIPS-Lite3指令集{MIPS-Lite2，ERET、MFC0、MTC0}。本处理器的模块继承了project2中的模块，因此只需要针对三条指令和CP0模块与CPU外设的模块编写，和原来模块的修改即可。

本project编写时参考了外部资料，半自主编写

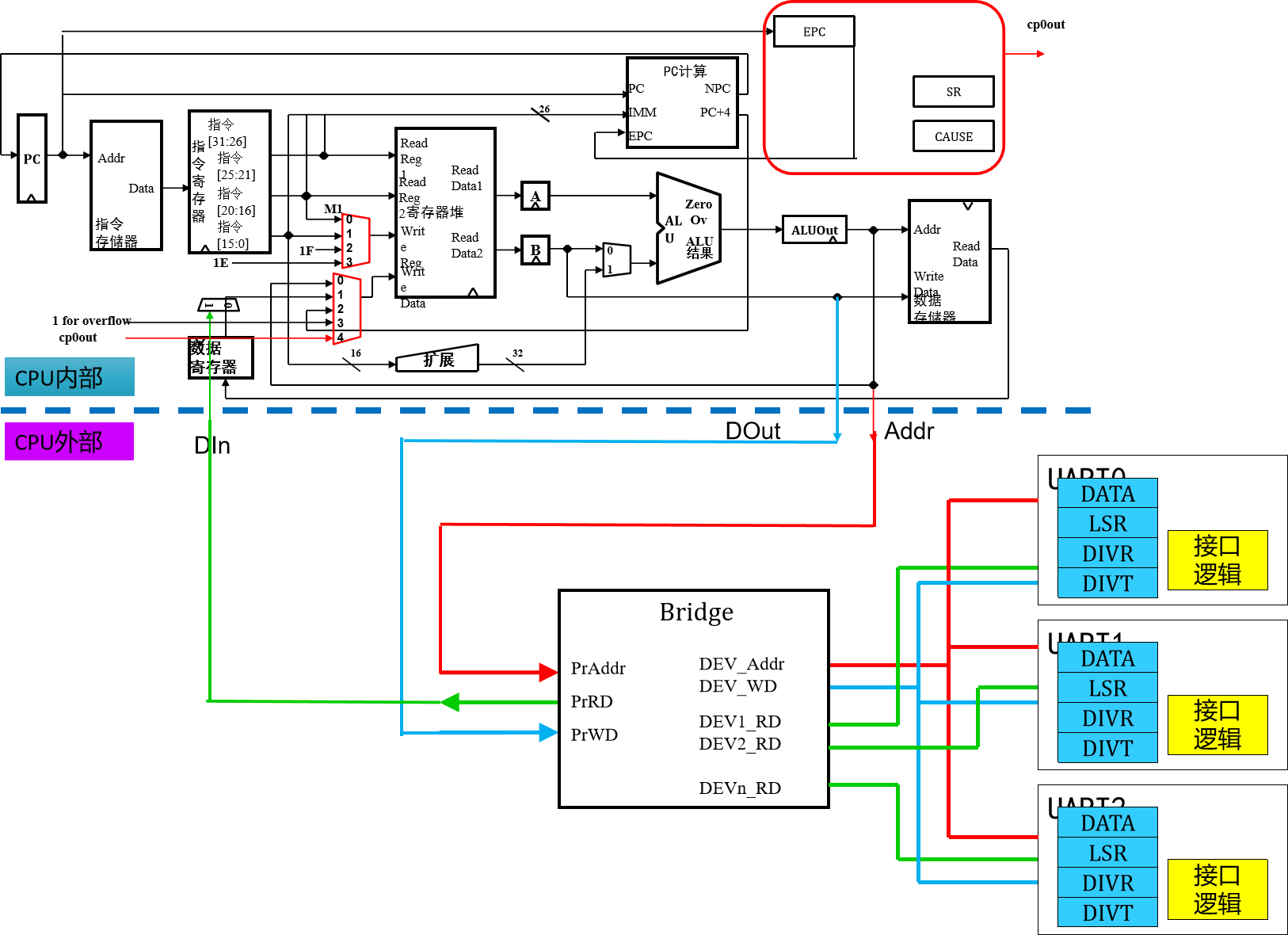


图28.project3总体数据通路结构设计图

## 二．模块定义

### 2.1 PC模块

#### 2.1.1 基本描述

PC模块是时序性模块，其功能为接收来自于PC计算模块输出的NPC信号，并在时钟信号clk上升沿时向外传输NPC输入的值

#### 2.1.2 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| pcin[31:0] | I | 来源于PC计算模块输入 |
| pcwr | I | PC模块写入权限，1为可写 |
| pcout[31:0] | O | 本模块输出值，和PC计算模块输入值的数值相等 |

#### 2.1.3 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 取地址 | 当clk信号上升沿时，如果PC可写，本模块将NPC模块计算好并输入进来的PC，输出出去（输出到IM模块） |

### 2.2 IM模块

#### 2.2.1 基本描述

IM模块是非时序的。其功能为读取事先写好的十六进制代码程序，存入内部的8kb指令寄存器，依据输入的PC地址信号，进行取指。小端序存储指令。将中断处理运行的指令存储在0000\_4180h

#### 2.2.2 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| addr[12:0] | I | 来源于PC模块的输出，也就是地址，由于IM模块存储指令的寄存器仅8KB，因此只需要PC的低13位 |
| dout[31:0] | O | 本模块输出值，是地址所对应的32位指令 |

#### 2.2.3 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 取指令 | 根据输入的地址，在本模块内部的指令寄存器里取得指令，并输出 |

### 2.3 NPC模块

#### 2.3.1 基本描述

NPC模块是非时序性模块。本模块是推进PC地址更新的核心，它会根据当前指令的性质，判断并分析下一条指令的地址，传送给PC模块。对于MIPS-Lite2指令集，NPC会根据beq，j和jal指令，jr和jalr指令和此外的指令，进行四种地址分析策略，从而计算出下一条指令的地址。特别地，对于jal指令，NPC模块还会输出PC+4以存入31号寄存器；对于jr和jalr指令，NPC模块需要输入GPR输出的读取到rs寄存器的内容，便于跳转。

而对于MIPS-LITE3指令集，NPC模块新加入了EPC模块输入的接口，并根据IntReq和ERET指令情况决定是否更新地址为EPC存储的内容。

复位时，本模块锁定PC地址为0000\_3000h

#### 2.3.2 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| imm[25:0] | I | 由IM输出的32位指令分线出低26位而来，用于计算j和jal指令跳转的地址，或计算beq跳转的地址 |
| pcin[31:0] | I | 来源于PC模块的输出，也就是接收目前PC地址，便于计算下一条地址 |
| rd1[31:0] | I | 来源于GPR模块的输出，其内容是rs寄存器内部存储的32位数据 |
| npcop[1:0] | I | 来源于Controller的输出，代表着本模块进行哪一种地址计算逻辑：  00：传输PC  01：beq指令跳转  10：j/jal指令跳转  11：jr和jalr指令跳转 |
| zero | I | 来源于ALU的输出，用于判断beq指令条件，决定是否跳转，如果为1才会进行beq指令跳转，否则正常pc+4推进 |
| rst | I | 异步复位信号，为1时会将PC地址重置为0000\_3000h，直至rst信号重新变为0 |
| epc[31:0] | I | 来源于CP0中EPC寄存器的输出，存储中断地址 |
| ERET | I | 来源于Controller模块的输出，判断当前执行指令是否为ERET |
| IntReq | I | 来源于Controller模块的输出，代表着当前有中断请求且为S10状态时，PC传入异常处理地址0000\_4180h |
| nextpc | O | 计算好的32位地址，输出给PC模块 |
| pc\_4 | O | 输入的pcin，在此基础上+4，专用于jal指令 |

#### 2.3.3 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 异步复位 | 复位信号为1时，pcout输出锁定为0000\_3000h |
| 2 | 判断并计算 | 本模块会先计算出j/jal指令和beq指令跳转后的地址，  beq跳转：pc + 4 + imm<<2  j跳转：{pc+4的高四位, imm26, 00}  然后根据npcop进行如下操作：  00：输出pc  01：如果zero=1输出beq跳转地址；否则pc+4  10：j/jal跳转地址  11：输出rd1信号内容 |
| 3 | 地址传递 | 计算好的新地址会传输给PC模块 |
| 4 | 支持中断和返回 | 通过ERET信号和IntReq信号，决定PC是否存入EPC存的值，或PC是否跳转到0000\_4180h |

### 2.4 GPR模块

#### 2.4.1 基本描述

本模块是时序性模块。但比较特殊的是，其数据输出并不受到时间信号的影响，仅存储数据时需要等待clk上升沿。本模块是CPU的寄存器堆，共有32个32位的寄存器，各司其职。其中0号寄存器恒为0，无法被修改；31号寄存器一般用来存储jal类指令所需要存储的地址，便于jr指令读取。

#### 2.4.2 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号 |
| regwrite | I | 来源于Controller模块，是寄存器的写入权限，1代表可写 |
| m1out[4:0] | I | 来源于M1模块，是待写入数据的寄存器的编码 |
| m2out[31:0] | I | 来源于M2模块，是待写入m1out指定寄存器的数据 |
| rs[4:0] | I | 来源于IR模块的输出irout[25:21] |
| rt[4:0] | I | 来源于IR模块的输出irout[20:16] |
| busa[31：0] | O | 读出rs对应的寄存器的数值 |
| busb[31:0] | O | 读出rt对应的寄存器的数值 |

#### 2.4.3 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 异步复位 | 复位信号为1时，32个寄存器全置0 |
| 2 | 写入数据 | clk为上升沿的时候，m2out的内容存入m1out解码后对应的寄存器 |
| 3 | 读取数据 | 读取IR模块irout输出对应irout[25:21]和[20:16]解码后对应的寄存器的数值并输出 |

### 2.5 EXT模块

#### 2.5.1基本描述

本模块是非时序模块。本模块的功能是根据信号，对输入的16位数进行32位扩展并输出至目标模块M3

#### 2.5.2 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| imm16[15:0] | I | 来源于IR模块的irout的低16位，代表着I型指令对应的立即数 |
| extop[1:0] | I | 来源于Controller控制本模块扩展立即数的逻辑  00：无符号拓展  01：符号拓展  10：高位拓展（专用于lui指令） |
| out[31:0] | O | 拓展完的数值进行输出 |

#### 2.5.3 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 扩展并输出 | 对立即数进行相应的拓展并输出 |

### 2.6 DM模块

#### 2.6.1 基本描述

本模块是时序模块。本模块是专门存储数据的数据寄存器，容量为12kb。当clk为上升沿的时候，允许写入数据。数据地址从0000\_0000h开始。小端序存储数据

根据islb和issb信号会进行字节读写

#### 2.6.2 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| addr[13:0] | I | 来源于ALU的输出，代表了本模块将要读写的地址 |
| din[31:0] | I | 来源于GPR的busb输出，代表了本模块将要写入的值 |
| we | I | 来源于Controller的输出，1为DM可写 |
| clk | I | 时钟信号 |
| islb | I | 来源于Controller的输出，1代表当前指令为lb指令 |
| issb | I | 来源于Controller的输出，1代表当前指令为sb指令 |
| dout[31:0] | O | 输出数据 |

#### **2.6.3 功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 写入数据 | 写使能有效的时候，当clk为上升沿，根据sb指令判断，再根据地址进行数据写入 |
| 2 | 读取数据 | 根据地址和lb指令判断，进行数据读取并输出 |

### **2.7 ALU模块**

#### 2.7.1基本描述

本模块为非时序模块。本模块为运算模块，对输入的两个数，根据控制信号进行相应运算并进行输出。同时在特殊情况下会进行两数相等的判定，或两数相加溢出的判定，并输出对应的信号。

#### **2.7.2 模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| A[31:0] | I | 运算数A |
| B[31:0] | I | 来源于M3模块的输出，运算数B |
| aluctr[2:0] | I | 运算逻辑  000：加法  001：减法  010：按位或运算  011：(A和B符号拓展) A < B ? 1 : 0，针对于slt指令  100：针对于addi指令的加法，会追加判断是否溢出 |
| zero | O | 针对于beq指令的信号，A=B时输出1，否则0 |
| overflow | O | 针对于addi指令的信号，A+B溢出时为1，否则0 |
| out[31:0] | O | 运算结果输出，为A(运算逻辑)B |

#### **2.7.3 功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 计算 | 根据aluctr进行相应逻辑的运算并输出结果 |
| 2 | 判断 | 针对于beq、slt、addi指令分别做出相应的判断，如上文所示 |

### **2.8 M1模块**

#### 2.8.1基本描述

本模块为非时序模块，用于GPR待写入寄存器编码的选择工作。

#### **2.8.2模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| gprsel[1:0] | I | 来源于Controller模块的输出，用于选择GPR待写入寄存器的5位编码，控制着m1out信号的输出数值  00：rt  01：isjalr==1 ？ $31 ： rd  10：1F  11：1E |
| isjalr | I | 来源于Controller的输出，1代表执行的是本条指令 |
| rt[4:0] | I | 来源于IM模块的输出，为dout[20:16] |
| rd[4:0] | I | 来源于IM模块的输出，为dout[15:10] |
| m1out[4:0] | O | 输出数值被gprsel控制 |

#### **2.8.3 功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 多路选择 | 选择GPR待写入寄存器编码 |

### **2.9 M2模块**

#### 2.9.1 基本描述

本模块为非时序模块，用于GPR待写入数据的选择工作。

#### **2.9.2模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| wdsel[2:0] | I | 来源于Controller模块的输出，用于选择GPR待写入寄数据，控制着m2out输出信号的值  000：aluo  001：dmout  010：pc\_4  011：0000\_0001h（专用于addi指令overflow的情况）  100：cp0out |
| aluo[31:0] | I | 来源于IM模块的输出，为dout[20:16] |
| dmout[31:0] | I | 来源于IM模块的输出，为dout[15:10] |
| pc\_4[31:0] | I | 来源于NPC模块的输出，为PC+4 |
| cp0out[31:0] | I | 来源于CP0模块的输出，即DOut信号 |
| m2out[31:0] | O | 输出数值被wdsel控制 |

#### **2.9.3 功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 多路选择 | 选择GPR待写入数据 |

### **2.10 M3模块**

#### 2.10.1 基本描述

本模块为非时序模块，用于ALU运算数B的选择工作

#### **2.10.2模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| bsel | I | 来源于Controller模块的输出，用于ALU运算数B的选择工作，控制着m3out输出信号的值  0：bo  1：imm32 |
| bo[31:0] | I | 来源于GPR模块的输出，为busb |
| imm32[31:0] | I | 来源于EXT模块的输出，为EXT模块的out |
| m3out[31:0] | O | 输出数值被bsel控制 |

#### **2.10.3 功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 多路选择 | ALU运算数B的选择 |

### **2.11 Controller模块**

#### 2.11.1 基本描述

本模块为非时序模块。本模块是整个CPU中的重中之重，犹如人之大脑一般重要。其功能为输入opcode和funct，从而对指令进行译码，输出控制信号，控制几乎CPU中每一个部件，没有了本模块，CPU中的各个模块只能是一盘散沙，可见其重要性所在。在project2的基础上，针对MIPS-LITE3的要求，添加了中断状态S10

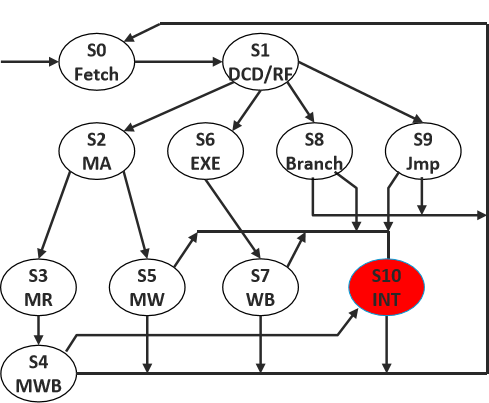


图29.状态图

S0为取指，S1为译码，是MIPS-LITE2指令集中每一条指令必经之路。

S2为针对于DM读写相关指令的入口，也就是lw,lb,sw,sb；其中lw和lb会进入 S3状态先对DM进行读取，之后进入S4状态进行GPR回写；sw和sb会进入S5状态访存DM

S6为针对于ALU运算相关指令的入口，也就是addu,subu,ori,addi,addiu,lui,slt，运算后会进入S7状态进行GPR回写

S8为针对于分支相关指令的入口，也就是beq指令

S9为针对于j型指令的入口，也就是j,jal,jr,jalr指令，本状态跳转和回写同时进行

由图可知，S4,S5,S7,S8,S9执行完毕后，重回S0状态取指，完成一个周期

S4,S5,S7,S8,S9若收到中断请求，均会转移到S10状态，进行中断处理

rst复位会让状态机的状态锁定到S0

#### **2.11.2 模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| opcode[5:0] | I | 来源于IM模块的输出，dout的高六位 |
| funct[5:0] | I | 来源于IM模块的输出，dout的低六位 |
| overflow | I | 来源于ALU模块的输出，是对addi指令运算结果是否溢出的判断信号，1代表溢出 |
| clk | I | 时钟信号 |
| rst | I | 复位信号 |
| zero | I | 来源于ALU模块的输出，代表了两数是否相等 |
| MF | I | 来源于IR模块的输出，irout[25:21] |
| IntReq | I | 来源于CP0模块的输出，1代表中断请求 |
| aluop[2:0] | O | 输出到ALU模块，控制运算逻辑 |
| gprsel[1:0] | O | 输出到M1模块，选择待写入数据的寄存器的五位编码 |
| gprwr | O | 输出到GPR模块，控制GPR的写入权限 |
| extop[1:0] | O | 输出到EXT模块，控制其拓展逻辑 |
| wdsel[2:0] | O | 输出到M2模块，选择写入GPR目标寄存器的数值 |
| npcop[1:0] | O | 输出到NPC模块，选择PC地址运算逻辑 |
| dmwr | O | 输出到DM模块，控制DM的写入权限 |
| bsel | O | 输出到M3模块，选择输入ALU作为运算数B的数值 |
| pcwr | O | 输出到PC模块，决定PC是否写入来自NPC的输入 |
| irwr | O | 输出到IR模块，决定IR是否写入来自IM的输入 |
| islb | O | 输出到DM模块，决定DM的输出 |
| issb | O | 输出到DM模块，决定DM的写入 |
| isjalr | O | 输出到M1模块，参与M1的多路选择 |
| cp0\_wen | O | 输出到CP0模块，决定了CP0内部寄存器的写入权限 |
| bridge\_wen | O | 输出到BRIDGE模块，决定了CPU内外部访存的开关 |
| EXLSet | O | 输出到CP0模块，参与对SR寄存器置位 |
| EXLClr | O | 输出到CP0模块，参与对SR寄存器的重置 |
| IntPc | O | 输出到NPC模块，决定NPC是否读取EPC |

#### **2.11.3 功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 译码产生控制信号 | 根据opcode和funct、zero和overflow以及IntReq和MF进行译码，获得上文所示的各种重要控制信号 |
| 2 | 状态机推进 | 推进状态机，完成多周期 |
| 3 | 复位 | 重置状态机 |

### **2.12 ALUOUT模块**

#### 2.12.1 基本描述

本模块为时序性模块。本模块接收ALU的输出结果，当时钟上升沿时，输出相应结果

#### **2.12.2 模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| aluoutin[31:0] | I | 来源于ALU模块的输出，是两数相运算的结果 |
| aluoutout[31:0] | O | 输出aluoutin |

#### **2.12.3 功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 传递数据 | clk为上升沿的时候，传递ALU的输出 |

### **2.13 AR模块 & BR模块**

#### 2.13.1基本描述

AR和BR模块均为时序性模块，类似于ALUOUT模块，也是传递信息用的，传递的是GPR读取到的busa和busb。由于二者性质完全相同，仅仅模块名和信号名为了区分才不同，因此下面只介绍AR，BR同理

#### **2.13.2模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| arin[31:0] | I | 来源于GPR模块的输出，也就是busa |
| arout[31:0] | O | 输出arin |

#### **2.13.3 功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 传递数据 | clk为上升沿的时候，传递GPR的输出 |

### **2.14 DR模块**

#### 2.14.1基本描述

DR模块为时序性模块，用于传递DM的输出

#### **2.14.2模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| drin[31:0] | I | 来源于DM模块的输出，也就是dmout |
| drout[31:0] | O | 输出dmout |

#### **2.14.3 功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 传递数据 | clk为上升沿的时候，传递DM的输出 |

### **2.15 IR模块**

#### 2.15.1基本描述

IR模块为时序性模块，用于传递IM的输出

#### **2.15.2模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| irin[31:0] | I | 来源于IM模块的输出，也就是dout |
| irwr | I | 写入权限 |
| irout[31:0] | O | 输出dout |

#### **2.15.3 功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 传递数据 | clk为上升沿的时候，且irwr为1时，传递DM的输出 |

### 2.16 M4模块

#### 2.16.1基本描述

M4模块为非时序模块，用于选择M2模块001接口的输入

#### **2.16.2模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| drout[31:0] | I | 来源于DR模块的输出 |
| prrd[31:0] | I | 来源于BRIDGE模块的输出 |
| addr[31:0] | I | 来源于ALUOUT模块的输出，用于判断当前读取数据来源于CPU外部还是内部  本着节省引脚资源，只用判断addr的[15:8]==8’h7f即可  0：drout  1：prrd |
| m4out[31:0] | O | 模块输出 |

#### **2.16.3 功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 多路选择 | 选择M2模块001接口的输入 |

### 2.17 TIMER模块

#### 2.17.1基本描述

本模块为时序性模块。本模块是外部设备，内部包含了三个32位小设备CTRL、PRESET和COUNT，地址是0000\_7F00 ~ 0000\_7F0B。具有计时功能，并且当程序进入死循环时可以发送中断请求，能进行计时设置和置位。

当计数器值为0时，根据计时器不同的模式执行不同的操作。当为模式0时，则一直保持0不变，直到PRESET再次被外部写入后，PRESET值再次被加载至COUNT，COUNT重新启动倒计数，如果此时CTRL[3]为1，那么将会发送中算请求IRQ；当为模式1时，则加载PRESET的值到COUNT，开始新一轮的倒计时。

值得一提的是，TIMER作为外部设备，其clk频率和CPU内部频率并不相同，因此TIMER和CPU交互的时候还需要设置分频器，但是我在阅读设计要求时没有注意到这个细节，在进行TIMER设计时虽然有过疑惑为什么clk信号要单独写成CLK\_I，但是没重视，没有进行分频器的编写。

#### **2.17.2模块接口**

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| CLK\_I | I | TIMER设备自己的时钟信号，为1秒 |
| RST\_I[3:2] | I | TIMER设备专属复位信号 |
| WE\_I | I | TIMER设备的写入权限 |
| ADD\_I[3:2] | I | TIMER内部三设备选址 |
| DAT\_I[31:0] | I | 待写入TIMER的数据 |
| DAT\_O[31:0] | O | TIMER读出的数据 |
| IRQ | O | 中断请求 |

#### **2.17.3 功能定义**

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | 重置内部三个设备的值 |
| 2 | 计数 | 倒计时计数 |
| 3 | 产生中断 | 发出中断请求 |

### 2.18 OUTPUTDEV模块

#### 2.18.1基本描述

本模块为时序模块。本模块是外部设备中的输出设备。包含了两个设备，也就是两个32位的寄存器，规定其地址为0000\_7F10h——0000\_7F18h。基于地址特点，我们仅需要地址位的[3:2]即可进行硬件选择

#### 2.18.2 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| en | I | 写使能 |
| din[31;0] | I | 待写入的数据 |
| addr[3:2] | I | 硬件地址 |
| dout[31:0] | O | 读取数据 |

#### 2.18.3 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 存储数据 | 存储数据 |
| 2 | 读取数据 | 读取数据 |

### 2.19 BRIDGE模块

#### 2.19.1基本描述

本模块为非时序模块。本模块是链接CPU内外的桥梁，正如其名。它具有一套地址，一套写数据，和n套读数据（n=设备个数）。有了本模块，CPU内外部得以获得联系，是中断和异常的必须模块。

#### 2.19.2 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| praddr[31:0] | I | 来源于ALUOUT的输出，是外部设备读取地址 |
| prwd[31:0] | I | 来源于BR的输出，是待写入外部设备的数据 |
| dev0\_rd[31:0] | I | 来源于从外部设备1读取到的数据 |
| dev1\_rd[31:0] | I | 来源于从外部设备2读取到的数据 |
| dev2\_rd[31:0] | I | 来源于从外部设备3读取到的数据 |
| wen | I | 来源于Controller的输出，代表了CPU内外部链接是否开启，即外部设备写入使能 |
| irq | I | 来源于TIMER的输出，代表了中断请求 |
| prrd[31:0] | O | 读取到的硬件信息内容 |
| dev\_wd[31:0] | O | 待写入数据输出 |
| dev\_addr[31:0] | O | 写入地址输出 |
| dev0\_we | O | 外部设备1的写使能 |
| dev2\_we | O | 外部设备3的写使能 |
| hwint[5:0] | O | 外部中断信号 |

#### 2.19.3 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 输出地址 | 直接输出prraddr[X:2],X由N个设备中地址空间需求最大者决定 |
| 2 | 地址匹配 | 设备地址译码 |
| 3 | CPU读数据 | 所有设备的数据输出汇聚至CPU的数据输入  MUX的控制由PrAddr中某些位译码决定 |
| 4 | CPU写数据 | CPU写数据：连接至所有设备的输入  直通输出，不需要再转换 |

### 2.20 CP0模块

#### 2.20.1基本描述

本模块为时序模块。本模块是CPU中的协处理器0，内部含有32个32位的寄存器，本project只关注其中四个寄存器，它们的编号和名字依次是12：SR 13：CAUSE 14：EPC 15：PRID。

SR寄存器只读，允许某硬件的中断请求，控制着全局使能，并且可以对已经中断的设备进行标记防止再次进入中断；CAUSE寄存器记录硬件中断是否有效；EPC寄存器保存PC，并进行指令读取；PRID可以存入开发商编写的只读的、独特的、个性的32位标识码。

正如其名，CP0协助着CPU，主要掌管异常和中断操作

#### 2.20.2 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| PC[31:0] | I | 来源于PC模块的输出，保存中断PC至EPC |
| DIn[31:0] | I | 来源于BR寄存器，为CP0待写入数据 |
| HWInt[5:0] | I | 来源于BRIDGE模块输出，是外部中断信号 |
| Sel[4:0] | I | 来源于IR的输出，是irout[15:11]也就是rd |
| Wen | I | 来源于Controller的输出，是CP0的写使能 |
| EXLSet | I | 来源于Controller的输出，参与SR寄存器置位 |
| EXLClr | I | 来源于Controller的输出，参与SR寄存器复位 |
| clk | I | 时钟信号 |
| rst | I | 复位信号 |
| IntReq | O | 中断请求信号 |
| epc[31:0] | O | epc内容输出 |
| DOut[31:0] | O | CP0的寄存器[Sel]输出 |

#### 2.20.3 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | 四个寄存器复位 |
| 2 | 记录PC | 记录中断产生的地址 |
| 3 | 写数据 | 向CP0的寄存器组写入数据 |
| 4 | 产生中断 | 根据中断请求和中断有效，产生中断 |
| 5 | 记录异常 | HWInt不断被CAUSE寄存器锁存 |

## 三．MIPS-Lite3指令集 机器指令描述

表3.MIPS-LITE3指令

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 助记符 | opcode | funct | 指令功能 | |
| 1 | addu | 000000 | 100001 | 两个来自寄存器的数无符号加法 | GPR[rd] ← GPR[rs] + GPR[rt] |
| 2 | subu | 000000 | 100011 | 两个来自寄存器的数无符号减法 | GPR[rd] ← GPR[rs] - GPR[rt] |
| 3 | ori | 001101 | - | 两个来自寄存器的数逻辑或运算 | GPR[rt] ← GPR[rs] or immediate |
| 4 | lw | 100011 | - | 读取DM中一个字到指定寄存器中 | GPR[rt] ← memory[GPR[base] + offset] |
| 5 | sw | 101011 | - | 指定寄存器中的字内容写到DM中 | memory[GPR[base] + offset] ← GPR[rt] |
| 6 | beq | 000100 | - | 两数相等则跳转 | if GPR[rs] = GPR[rt] then branch |
| 7 | lui | 001111 | - | 16位立即数高位拓展 | GPR[rt] ← immediate || 016 |
| 8 | j | 000010 | - | 无条件跳转 | PC ← PCGPRLEN-1..28 || instr\_index || 02 |
| 9 | addi | 001000 | - | 加立即数（符号拓展，有溢出检验） | temp ← (GPR[rs]31||GPR[rs]31..0) + sign\_extend(immediate)  if temp32 ≠ temp31 then  SignalException(IntegerOverflow)  else  GPR[rt] ← temp  endif |
| 10 | addiu | 001001 | - | 加立即数（符号拓展，无溢出检验） | GPR[rt] ← GPR[rs] + immediate |
| 11 | slt | 000000 | 101010 | 小于则置1 | GPR[rd] ← (GPR[rs] < GPR[rt]) |
| 12 | jal | 000011 | - | 跳转并链接（PC+4存到$31） | GPR[31]← PC + 8  PC ← PCGPRLEN-1..28 || instr\_index || 02 |
| 13 | jr | 000000 | 001000 | 跳转到寄存器存储的地址 | PC ← GPR[rs] |
| 14 | lb | 100000 | - | 读取DM中的字节 | GPR[rt] ← memory[GPR[base] + offset] |
| 15 | sb | 101000 | - | 向DM中存储字节 | memory[GPR[base] + offset] ← GPR[rt] |
| 16 | jalr | 000000 | 001001 | 跳转到rs中存储的地址，并将PC+4存储到rd中，如果没有指定rd，则默认为31号寄存器 | GPR[rd] ← return\_addr, PC ← GPR[rs] |
| 17 | eret | 010000 | 011000 | 中断返回 | PC ← EPC |
| 18 | mtc0 | 010000 | irout[25:21]=00100 | CP0寄存器写入 | CPR[0, rd] ← GPR[rt] |
| 19 | mfc0 | 010000 | irout[25:21]=00000 | 读取CP0寄存器 | GPR[rt] ← CPR[0,rd] |

## 四．测试程序

### 4.1 MIPS-LITE3指令集测试程序



图30.主程序

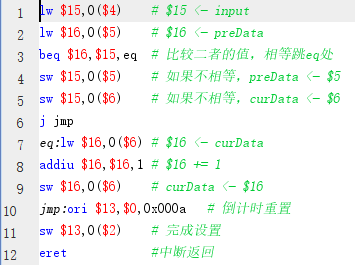


图31.中断子程序

①主程序通过读取 32 位输入设备内容并显示在32位输出设备上。

②主程序将定时器初始化为模式 0，并加载正确的计数初值至预置计数初值寄存器以产生 1s 的计数周期。

③主程序启动定时器计数后进入死循环。

④中断子程序不断读取新的输入设备内容，一旦发现与之前的 32位输入值不同，则更新32位输出设备显示为当前新值；否则将输出设备显示内容加1。然后重置初值寄存器从而再次启动定时器计数，实现

新一轮秒计数。

由于MARS在对外设和中断程序执行调试很繁琐，因此下面只展示modelsim验证波形

## 五．测试结果

### 5.1波形图（重点观察倒计时和中断返回）

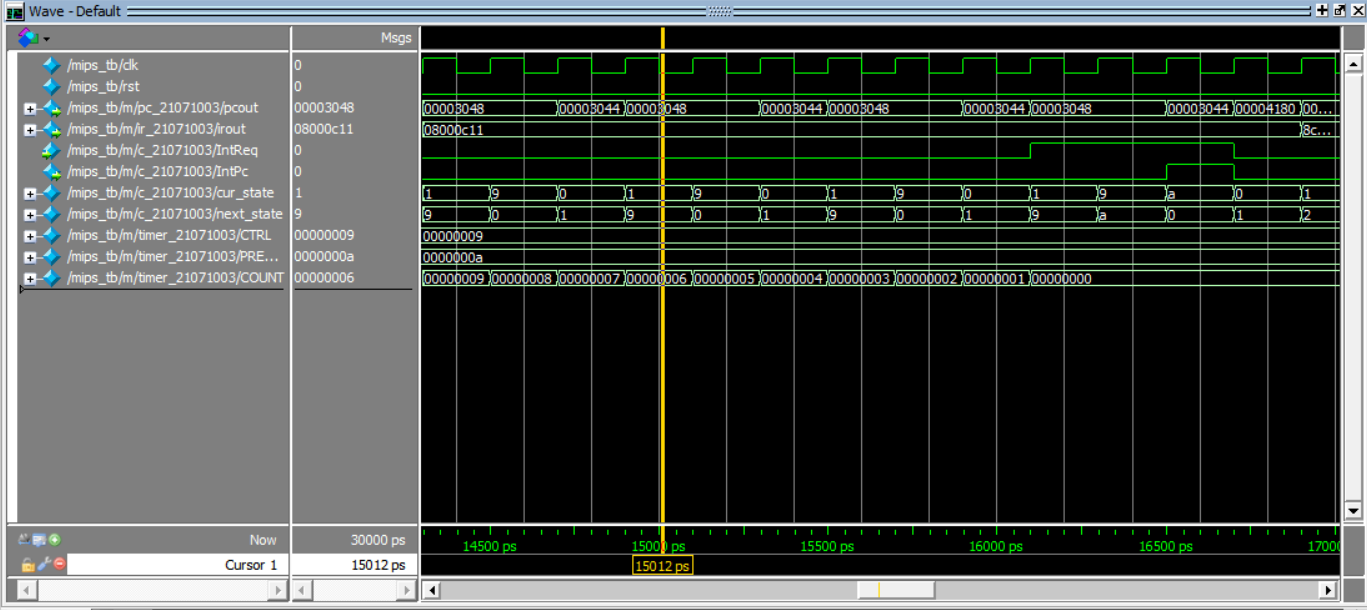


图32.倒计时

由图32.可见，在死循环反复跳转时，倒计时模块进行倒计时功能，当倒计时到0时，中断请求发起，进入预先在0000\_4180h放入的中断指令，也就是图31.的指令开始执行。

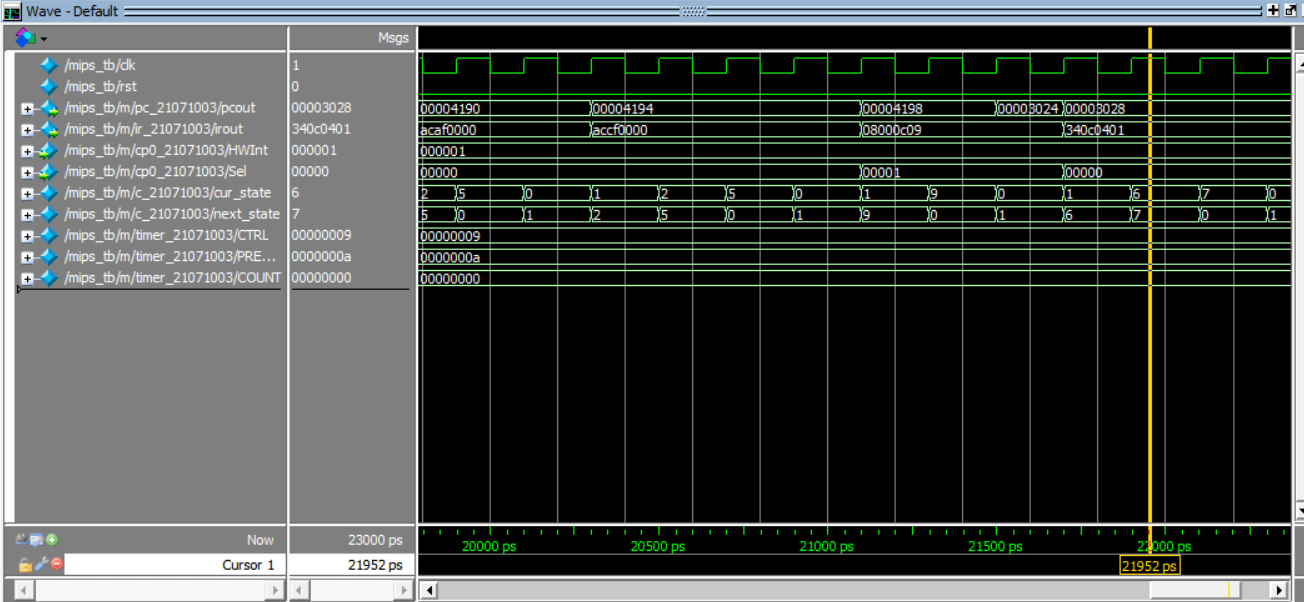


图33.中断返回

由图33.可见，中断返回，回到了中断产生的地址，由于中断产生的地址对应指令为死循环，因此后续的波形将循环进行中断和中断后指令处理，从而对输出设备的数值进行循环更新。

### 5.2寄存器结果

#### 5.2.1 GPR

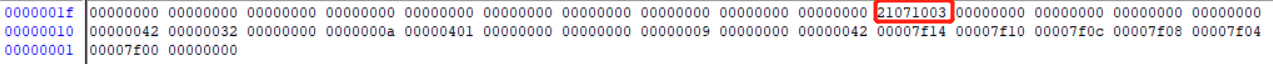


图34.GPR结果

由图34.可见，mfc0指令正确执行，读取到了PRID

#### 5.2.2 CP0寄存器



图35.CP0寄存器结果1

由图35.，图为中断跳转前的寄存器，可以看出SR寄存器此时已经放置好初值，CAUSE寄存器已经锁存好HWInt，而EPC因为未跳转，还未存入地址，PRID固定为设置好的21071003

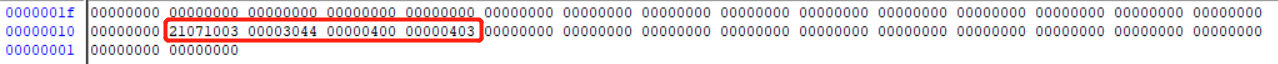


图36.CP0寄存器结果2

由图36.，图为中断跳转后的寄存器，可以看出SR寄存器中exl已经置1，CAUSE寄存器仍在锁存，而EPC已经存入了中断产生的地址，便于后续的返回操作

经过如上验证，证明CPU设计合理，已经基本完成了规定的任务

## 六．**Project3完成后的心得体会**

project3给我的感觉就是非常难，在周二调试完p2后，我马不停蹄赶到了图书馆去重新学习中断相关知识，然而废了很久时间，仅仅是能自主画出数据通路图，大致理解了CPU内外部沟通的逻辑，但是当上手写代码的时候，发现各个控制信号什么时候该是什么还是不理解。

由于检查时间紧张，我选择了半自主编写代码，和同学进行了探讨和交流，互相借鉴思路，并且借助了一些外部资料，来辅助自己理解各信号间的关系，最后做了很多代码注释，理解了p3的全部知识内容。在检查时我也坦白了project3并非全自主完成，对此拿不到A档满分我心服口服，更何况我没有注意到COUNTER周期要求设置为1s的细节，没有设计对应的分频器。

我认为本project非自主完整，大部分原因是我课堂上没有认真学习中断知识，认为考试不考就不需要花费时间去复习巩固，然而等临近检查才开始学习为时已晚，本次扣分已经给了我经验教训。同时我和同学认为p3的任务指导书有些不具体，不足够我们理解project3的全部内容，已经在检查当日汇报。我们还发现任务书中提到的“《异常中断及协处理器》”资料找不到。

从入学开始，我就深知计算机组成原理对于计科专业的重要性，它是考研必考且占比很大的科目，它是计科人必备的知识，它是理解硬件最底层最基础的原理，通过一个学期的学习，我见识到了它的难度和强度，也通过课设亲自体会感受到了CPU设计的神奇之处。本学期，我萌生了以后向硬件、芯片研究的想法，在结束了计组和课设的学习后，我将会承载着实践中获得的知识和经验，在暑假进一步学习中科院“一生一芯”计划，不断努力！

最后，请允许我在课设报告的结尾，感谢老师一学期以来的教学，感谢学习过程中老师、同学们给予的帮助和鼓励！为我的计组和课设画上了完美的句号！