**计算机组成原理**

**课内大作业报告**

**学 号\_\_\_\_\_\_\_\_\_\_20020203 \_\_\_\_\_\_\_\_**

**姓 名\_\_\_\_\_\_\_\_\_\_ 王思哲 \_\_\_\_\_\_\_\_\_**

**指导教师\_\_\_\_\_\_\_\_\_\_\_魏坚华\_\_\_\_\_\_\_\_\_\_\_**

**提交日期\_\_\_\_\_\_\_ 2022年5月9日\_\_\_\_\_\_**

**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

[一、 总体数据通路结构设计图 4](#_Toc103017715)

[二、 模块定义 5](#_Toc103017716)

[2.1 IFU模块 5](#_Toc103017717)

[2.1.1 设计总览&模块封装 5](#_Toc103017718)

[2.1.2 基本描述 5](#_Toc103017719)

[2.1.3 模块接口 6](#_Toc103017720)

[2.1.4 功能描述 6](#_Toc103017721)

[2.2 ALU模块 7](#_Toc103017722)

[2.2.1 设计总览&模块封装 7](#_Toc103017723)

[2.2.2 基本描述 7](#_Toc103017724)

[2.2.3 模块接口 8](#_Toc103017725)

[2.2.4 功能描述 8](#_Toc103017726)

[2.3 GPR模块 9](#_Toc103017727)

[2.3.1 设计总览&模块封装 9](#_Toc103017728)

[2.3.2 基本描述 9](#_Toc103017729)

[2.3.3 模块接口 10](#_Toc103017730)

[2.3.4 功能描述 10](#_Toc103017731)

[2.4 并行加法器模块 11](#_Toc103017732)

[2.4.1 设计总览&模块封装 11](#_Toc103017733)

[2.4.2 基本描述 12](#_Toc103017734)

[2.4.3 模块接口 12](#_Toc103017735)

[2.4.4 功能描述 13](#_Toc103017736)

[2.5 Controller模块 13](#_Toc103017737)

[2.5.1 设计总览&模块封装 13](#_Toc103017738)

[2.5.2 基本描述 13](#_Toc103017739)

[2.5.3 模块接口 14](#_Toc103017740)

[2.5.4 功能描述 15](#_Toc103017741)

[2.6 EXT模块 15](#_Toc103017742)

[2.6.1 设计总览&模块封装 15](#_Toc103017743)

[2.6.2 基本描述 16](#_Toc103017744)

[2.6.3 模块接口 16](#_Toc103017745)

[2.6.4 功能描述 16](#_Toc103017746)

[2.7 DM模块 17](#_Toc103017747)

[2.7.1 设计总览&模块封装 17](#_Toc103017748)

[2.7.2 基本描述 17](#_Toc103017749)

[2.7.3 模块接口 18](#_Toc103017750)

[2.7.4 功能描述 18](#_Toc103017751)

[三、 指令描述 19](#_Toc103017752)

[3.1 addu无符号加法 19](#_Toc103017753)

[3.2 subu无符号减法 19](#_Toc103017754)

[3.3 ori或立即数 20](#_Toc103017755)

[3.4 lw加载字 20](#_Toc103017756)

[3.5 sw存储字 21](#_Toc103017757)

[3.6 beq相等时转移 21](#_Toc103017758)

[3.7 lui立即数加载到高位 22](#_Toc103017759)

[3.8 j跳转 22](#_Toc103017760)

[四、 测试程序 22](#_Toc103017761)

[五、 测试结果 23](#_Toc103017763)

[5.1 Mars中仿真结果 23](#_Toc103017764)

[5.1.1 寄存器数据 23](#_Toc103017765)

[5.1.2 数据存储器数据 23](#_Toc103017767)

[5.2 logisim中仿真结果 24](#_Toc103017769)

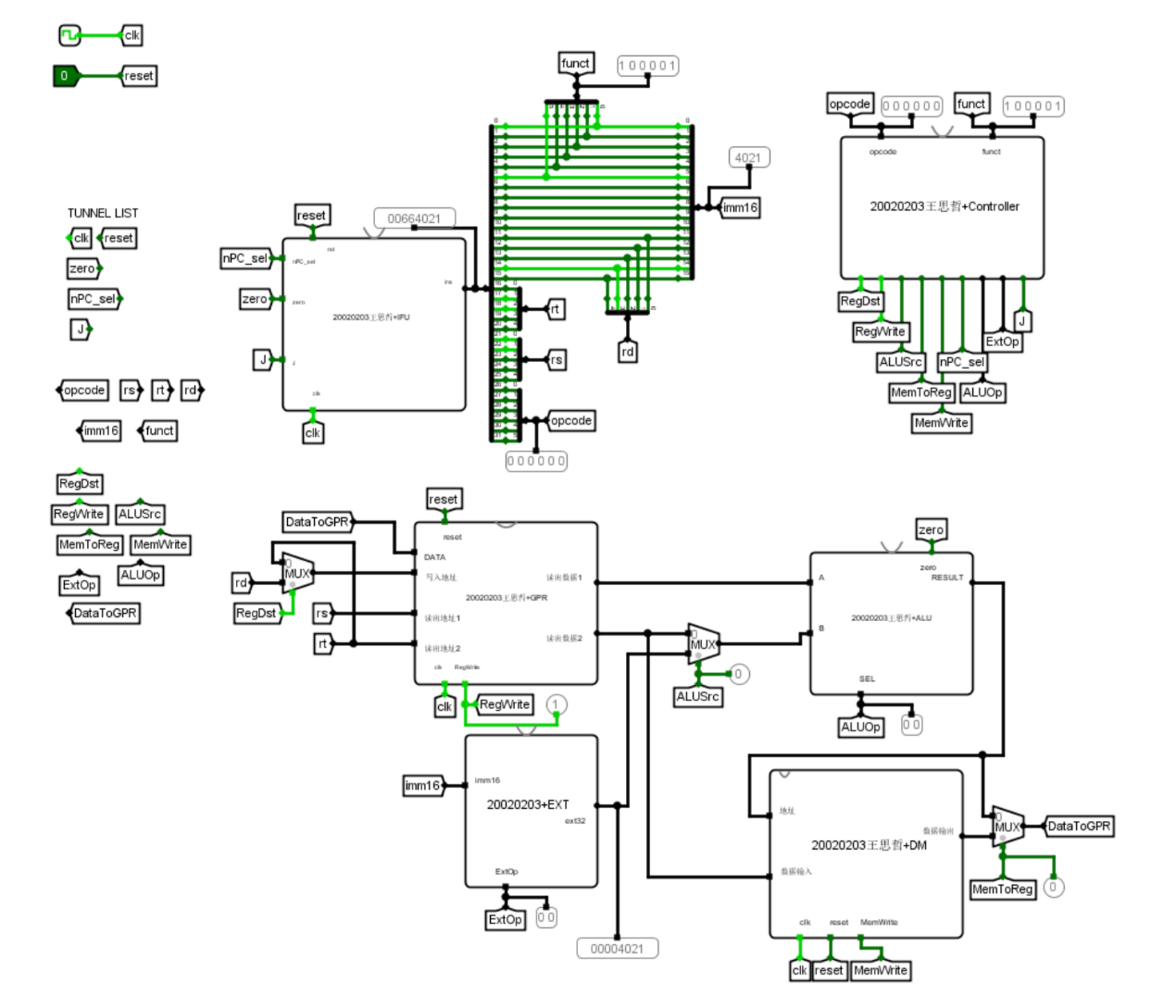
[5.2.1 寄存器数据 24](#_Toc103017770)

[5.2.2 数据存储器数据 24](#_Toc103017772)

[5.3 结论 25](#_Toc103017774)

[六、 总结与心得体会 25](#_Toc103017775)

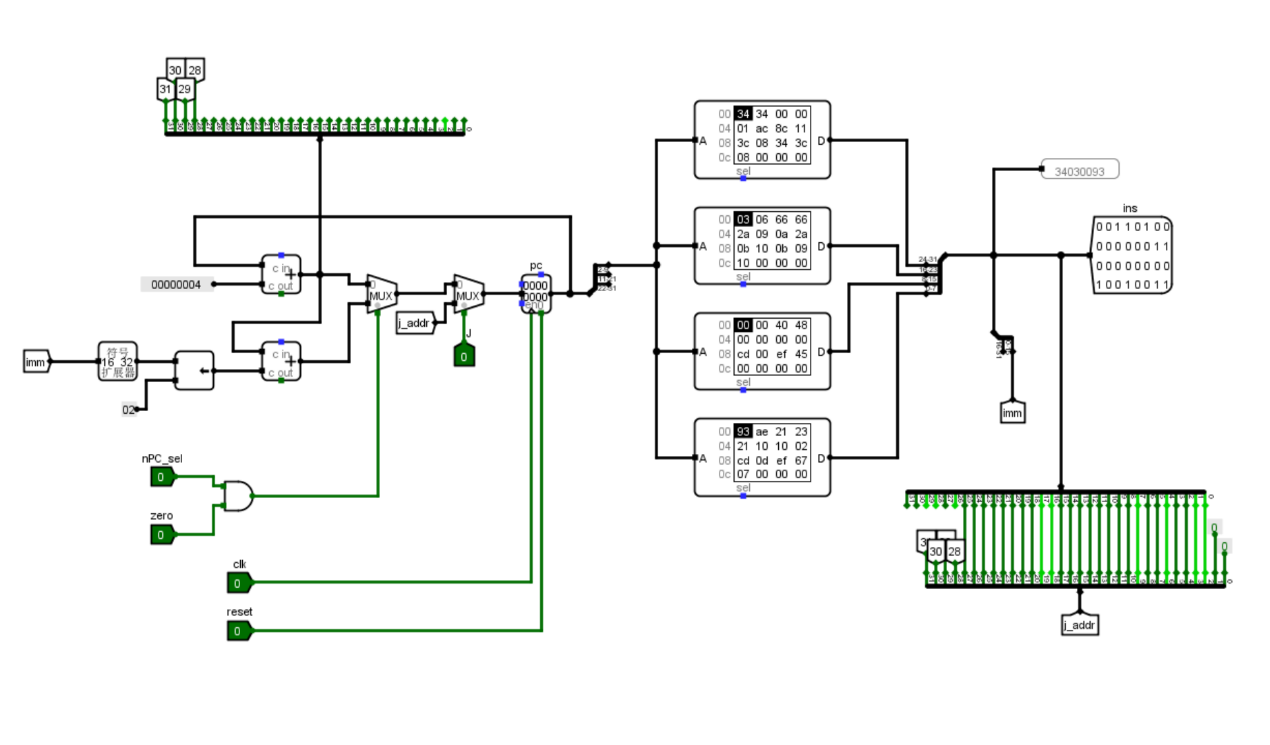
1. **总体数据通路结构设计图**

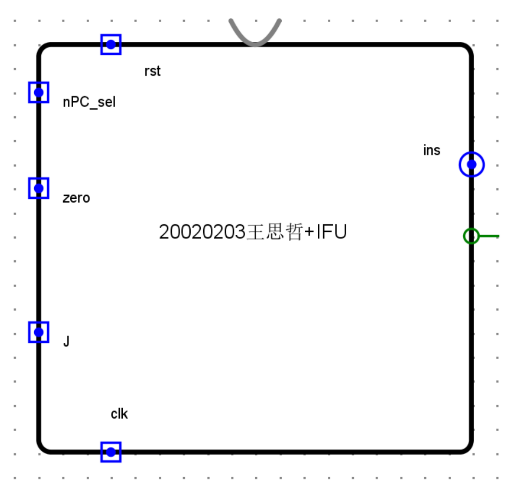


1. **模块定义**

## 2.1 IFU模块

### 2.1.1 设计总览&模块封装





### 2.1.2 基本描述

IFU在数据通路中充当的是取指部件的角色，主要完成取指令的功能。

IFU内部结构关键部件有PC、指令存储器。其中，PC使用寄存器实现，位宽为32位。IM由4个ROM组成，其输入地址为8位，数据位宽也为8位。

IFU在每个时钟沿到来时进行更新，分为两种情况：当为一般情况时，PC=PC+4；当为lui或j等跳转指令时，PC会根据跳转地址进行更新。

### 2.1.3 模块接口

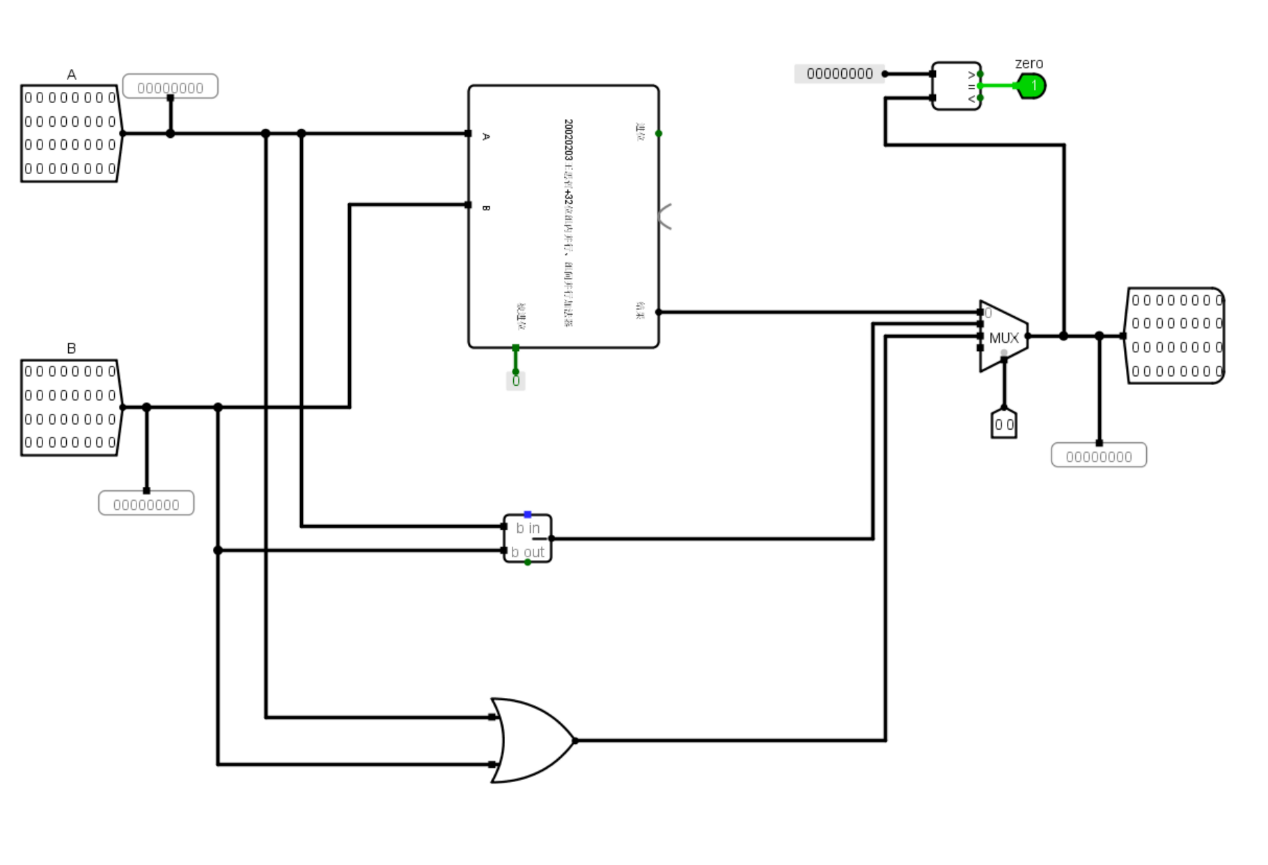
|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| nPC\_sel | I | 当前指令是否为beq指令。  1：指令为beq指令  0：指令非beq指令 |
| zero | I | ALU计算结果是否为0  1：是  0：否 |
| clk | I | 时钟信号 |
| reset | I | 复位信号，使PC为0。  1：复位  0：无效 |
| J | I | 当前指令是否为J指令  1:指令为J指令  0:指令非J指令 |
| Ins[31:0] | O | PC指针指向指令存储器对应地址的指令。 |

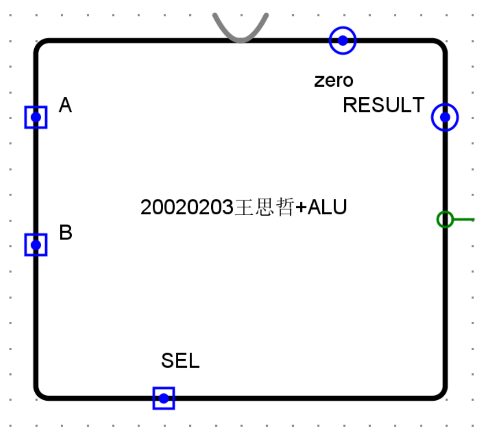
### 2.1.4 功能描述

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | 当复位信号有效时，PC置0。 |
| 2 | 取指令 | 根据PC从IM中取出指令。 |
| 3 | 计算下一条指令地址 | 如果当前指令不是beq指令，则PC🡨PC+4  如果当前指令是beq指令，并且zero为0，则PC🡨PC+4  如果当前指令是beq指令，并且zero为1，则PC🡨PC+4+(sign\_ext(ins[15:0])<<2)  如果当前是J指令，则PC[31:28]由PC+4高四位提供，PC[27:2]由ins低26位提供，PC[1:0]补0. |

## 2.2 ALU模块

### 2.2.1 设计总览&模块封装





### 2.2.2 基本描述

根据选择信号指示的运算类型，将两个输入的数据进行对应的运算后，输出结果。同时，如果结果为0，则将zero信号置1.

### 2.2.3 模块接口

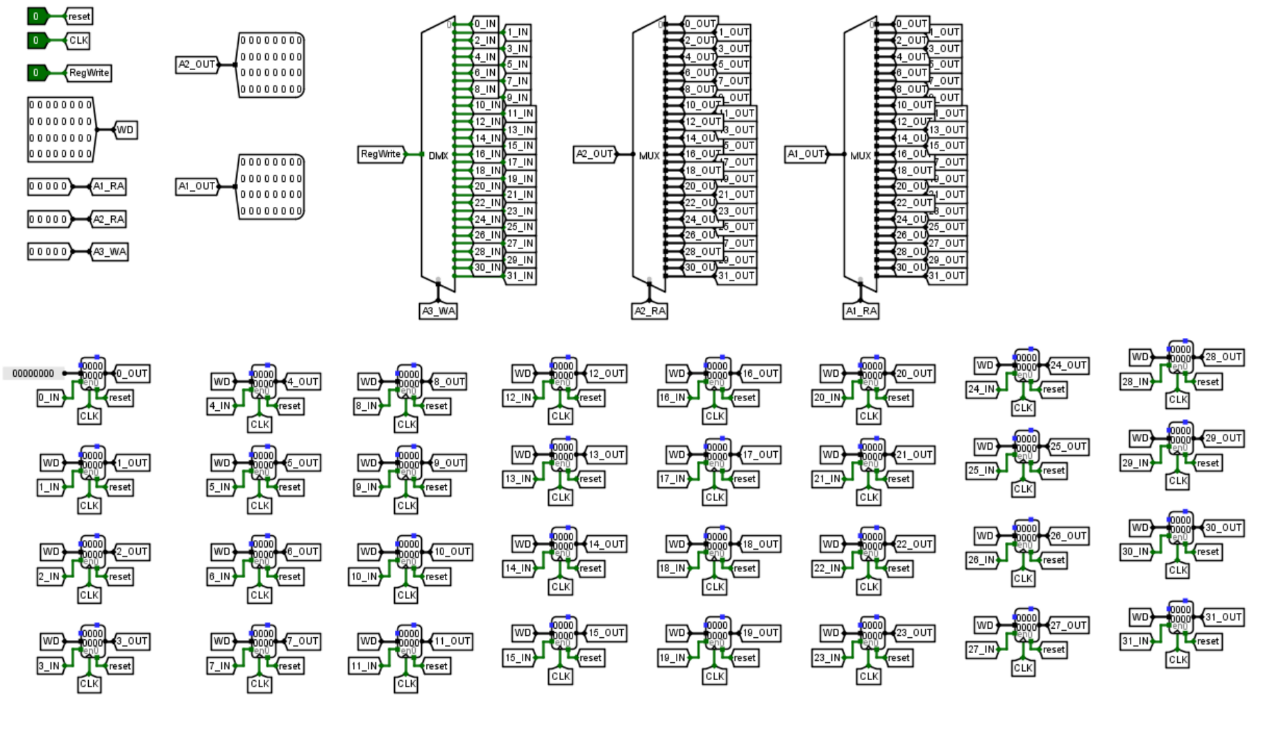
|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| A [31:0] | I | 操作数1 |
| B [31:0] | I | 操作数2 |
| SEL[1:0] | I | 运算类型  00：加法  01：减法  10：逻辑或 |
| zero | O | 判断运算结果是否为0  1:为0  0:非0 |
| RESULT[] | O | 运算结果输出 |

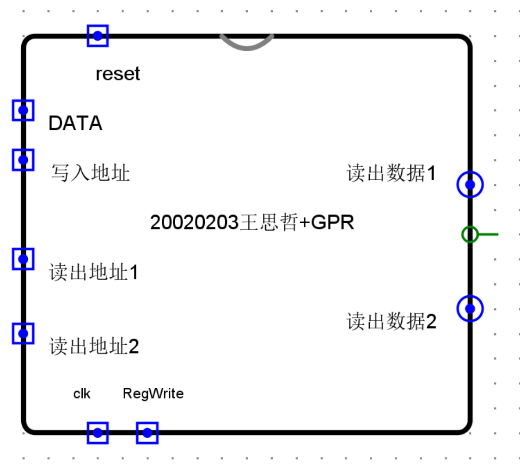
### 2.2.4 功能描述

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 加法运算 | 计算操作数1+操作数2 |
| 2 | 减法运算 | 计算操作数1-操作数2 |
| 3 | 逻辑或运算 | 计算操作数1 || 操作数2 |
| 4 | 结果0判断 | 判断运算结果是否为0 |

## 2.3 GPR模块

### 2.3.1 设计总览&模块封装





### 2.3.2 基本描述

包括32个具有写使能的32位寄存器。写使能有效时可以根据数据写入地址以及输入的数据写数据，任何时刻都可以根据读数据地址读数据。可以异步置0.

### 2.3.3 模块接口

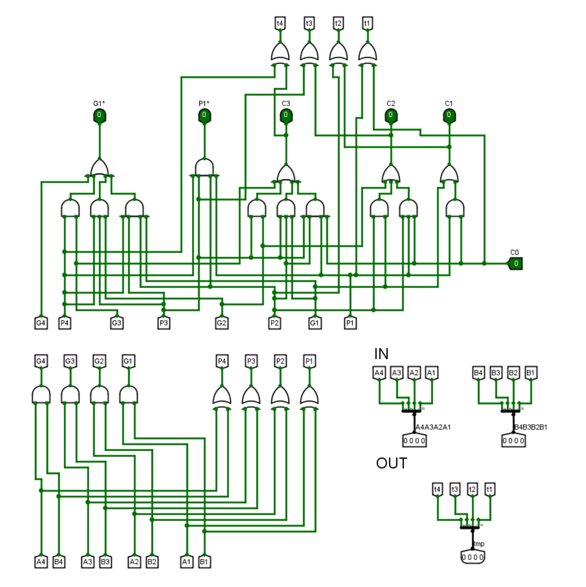
|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| reset | I | 寄存器数据是否置0  1：置0  0：无效 |
| clk | I | 时钟信号 |
| RegWrite | I | 写使能信号  1：写有效  0：写无效 |
| DATA[31:0] | I | 数据输入 |
| 写入地址[4:0] | I | 数据应该写入的寄存器的地址 |
| 读出地址1[4:0] | I | 应该读数据的寄存器的地址 |
| 读出地址2[4:0] | I | 应该读数据的寄存器的地址 |
| 读出数据1[31:0] | O | 数据输出 |
| 读出数据2[31:0] | O | 数据输出 |

### 2.3.4 功能描述

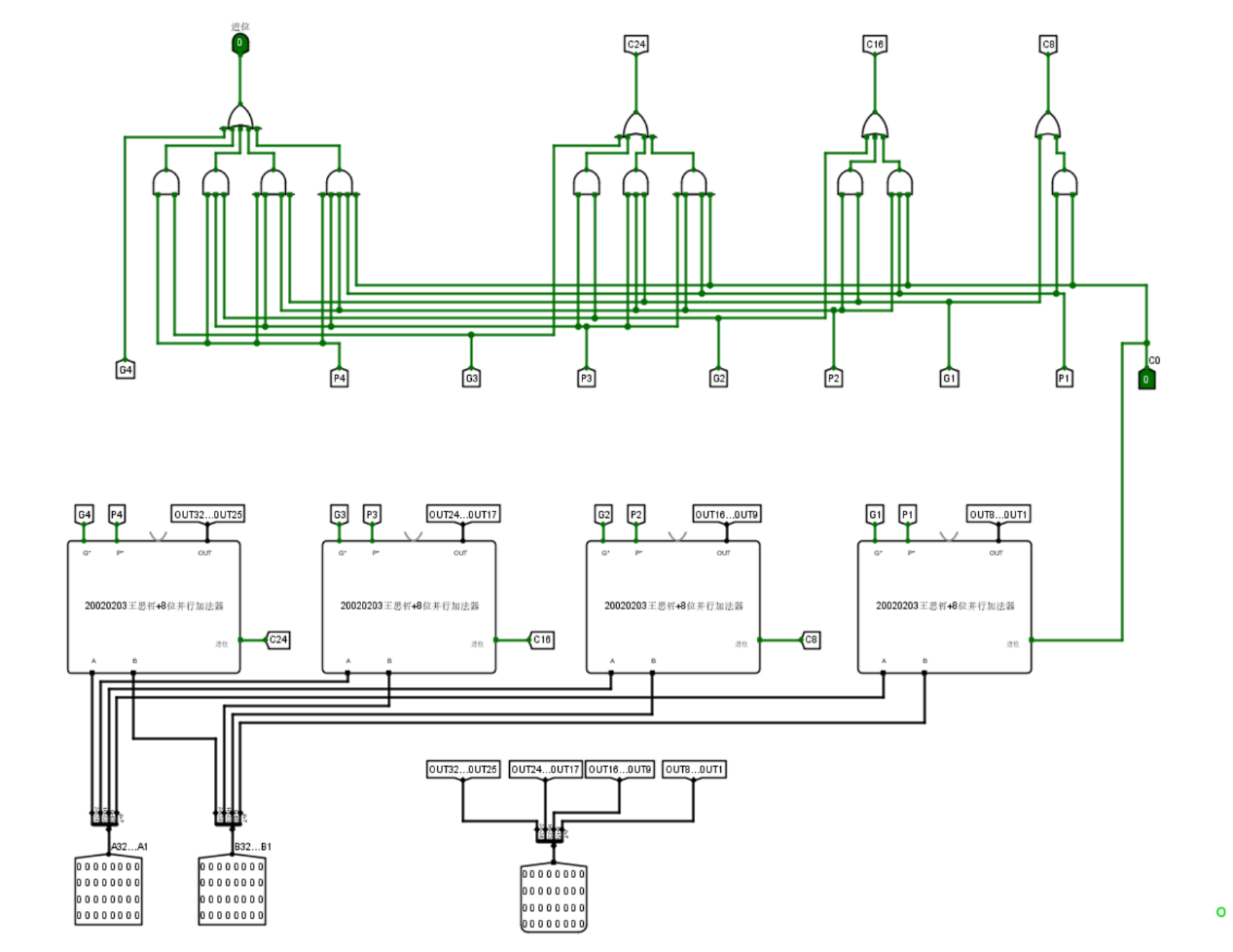
|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | 当复位信号有效时，32个寄存器内容均置0. |
| 2 | 写数据 | 根据写入地址，将数据写入寄存器中。 |
| 3 | 读数据 | 根据读出地址，将数据读出并输出。 |

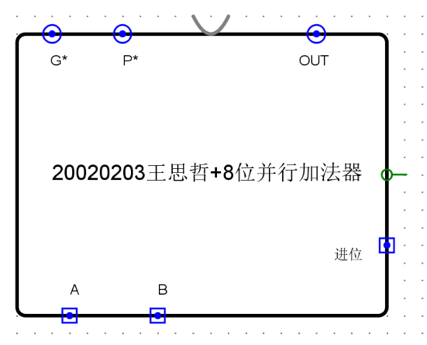
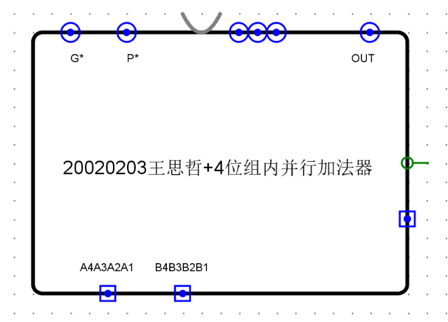
## 2.4 并行加法器模块

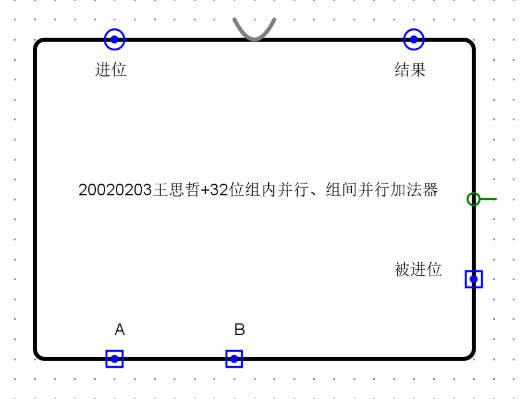
### 2.4.1 设计总览&模块封装

图示

描述已自动生成







### 2.4.2 基本描述

32位组内组间全并行加法器实现两个操作数的加法运算，支持溢出功能。

32位并行加法器是由4个8位加法器组间并行组成，其中8位由两个4位加法器构成，4位加法器实现组内并行的加法运算。

设计参考：计算机组成原理与汇编语言P43-P45内容

### 2.4.3 模块接口

只给出32位组内组间全并行加法器模块接口，8位&4位加法器略。

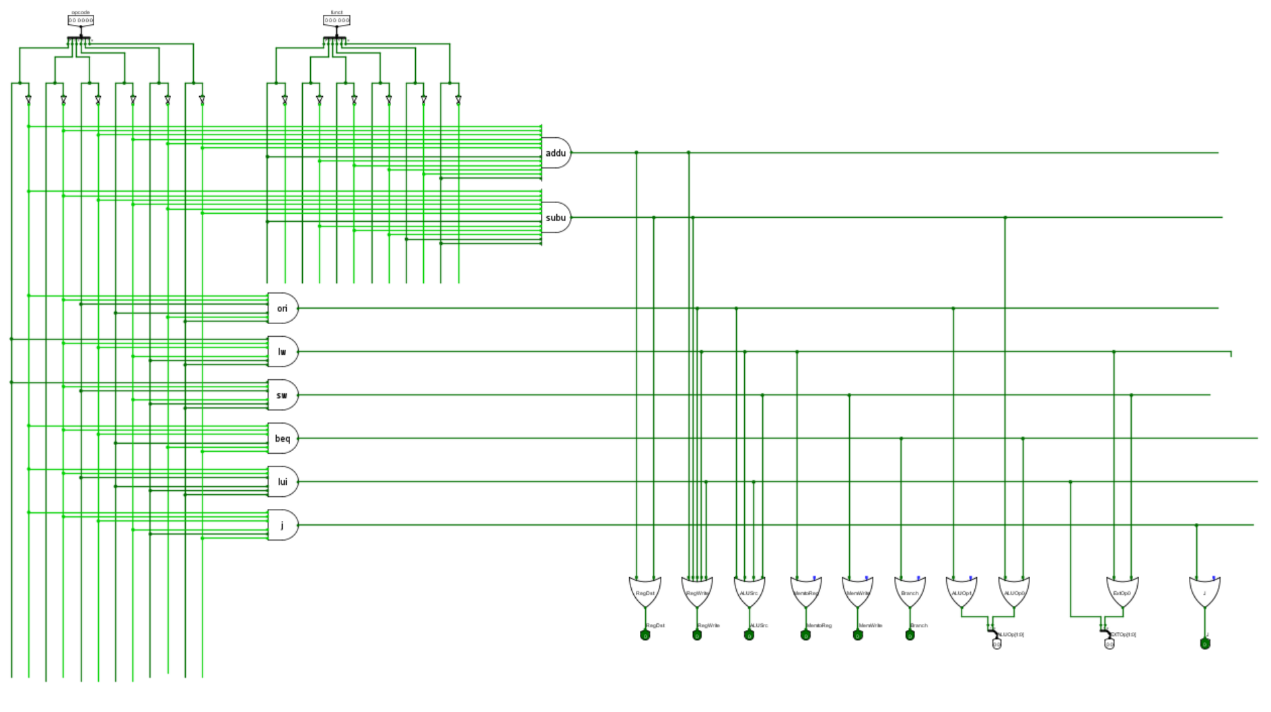
|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| A[31:0] | I | 操作数1 |
| B[31:0] | I | 操作数2 |
| 被进位 | I | 一般置常数0 |
| 进位 | O | 向上进位，溢出判断。  1：有溢出  0：无溢出 |
| 结果[31:0] | P | 运算结果：操作数1+操作数2 |

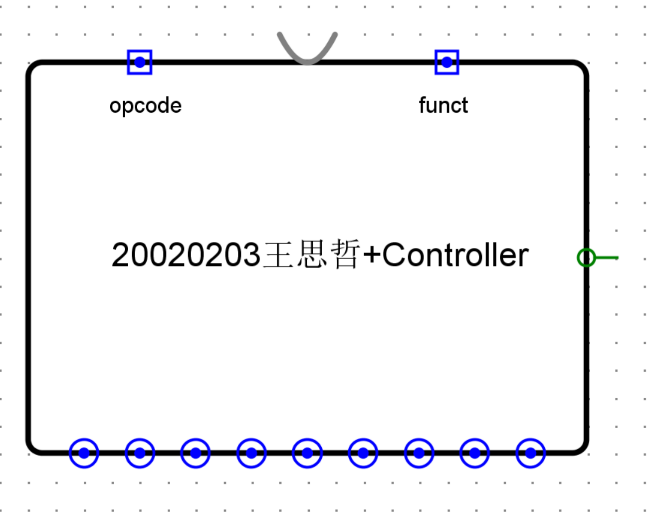
### 2.4.4 功能描述

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 加法运算 | 将两个输入的操作数进行加法运算。 |

## 2.5 Controller模块

### 2.5.1 设计总览&模块封装





### 2.5.2 基本描述

根据opcode和funct信号产生对应的控制信号。

### 2.5.3 模块接口

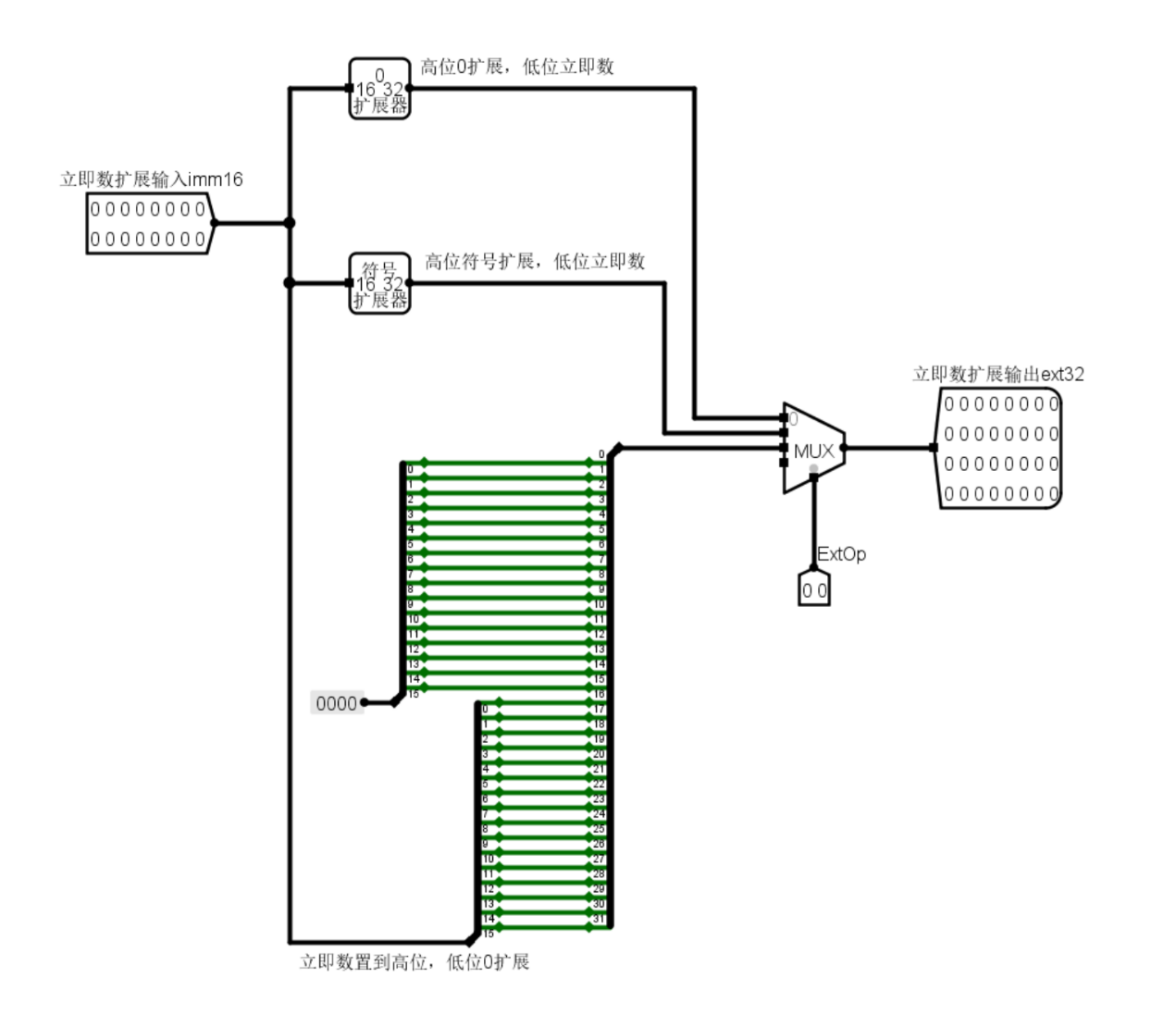
|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| opcode[5:0] | I | 操作码 |
| funct[5:0] | I | 功能码 |
| RegDst | O | 判断是否使用rd作为写入寄存器的地址  1：是  0：否 |
| ALUSrc | O | 判断是否使用扩展后的立即数作为ALU的第二操作数  1：是  0：否 |
| MemToReg | O | 判断是否使用DM中读出的数据输入寄存器  1：是  0：否 |
| MemWrite | O | 判断是否写入寄存器  1：是  0：否 |
| RegWrite | O | 判断是否写入寄存器  1：是  0：否 |
| nPC\_sel | O | 判断是否为beq指令  1：是  0：否 |
| ALUOp[1:0] | O | ALU的操作类型  00：加法  01：减法  10：逻辑或 |
| ExtOp[1:0] | O | 立即数拓展的类型  00：0拓展  01：符号拓展  10：将16位立即数扩展至高16位，低位补0 |
| J | O | 判断是否为j指令  1：是  0：否 |

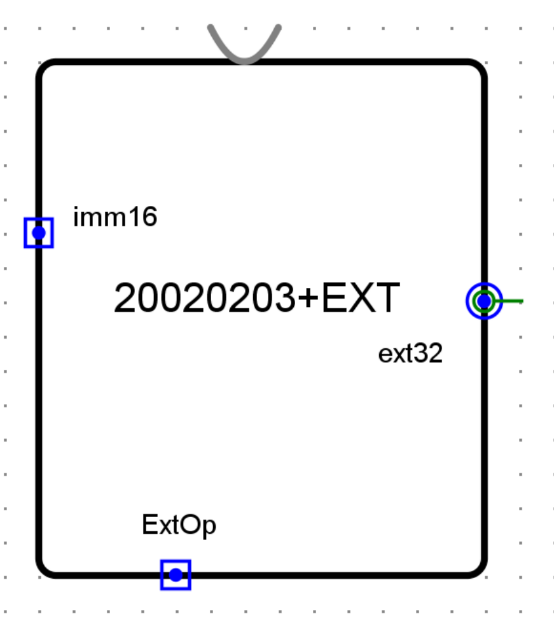
### 2.5.4 功能描述

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 产生控制信号 | 根据opcode & funct信息，产生对应的控制信号。 |

## 2.6 EXT模块

### 2.6.1 设计总览&模块封装





### 2.6.2 基本描述

根据选择信号ExtOp[1:0]指示的类型，将16位立即数拓展到32位。类型包括0拓展，符号拓展以及立即数拓展至高位、低位补0三种。

### 2.6.3 模块接口

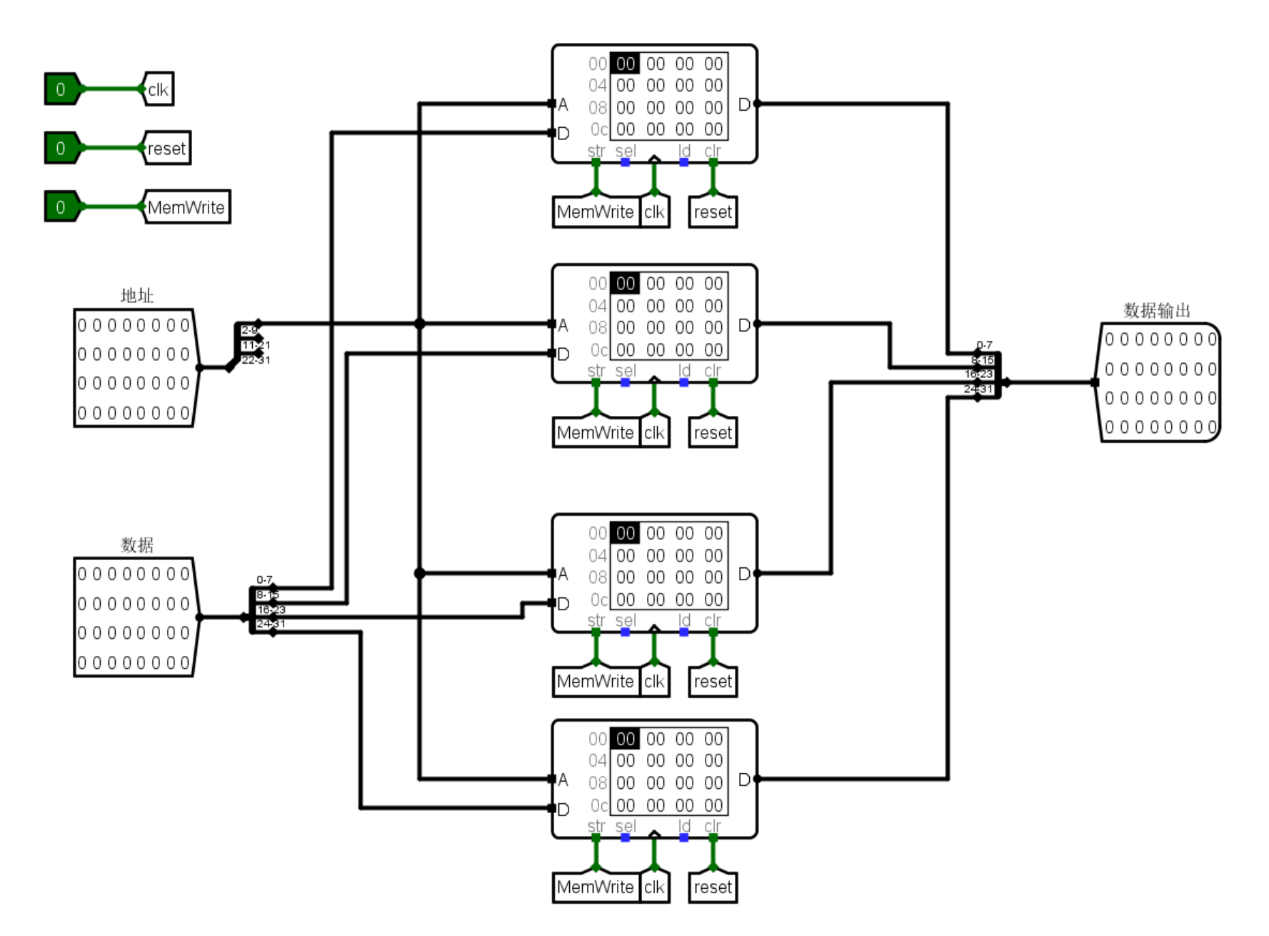
|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| imm16[15:0] | I | 16位立即数 |
| ExtOp[1:0] | I | 选择立即数拓展的类型  00：0拓展  01：符号拓展  10：立即数拓展至高位，低位补0 |
| ext32 | O | 拓展结果 |

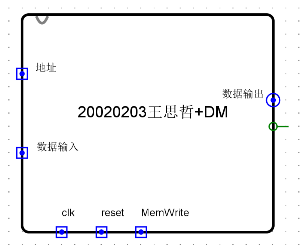
### 2.6.4 功能描述

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 0拓展 | 高16位0拓展 |
| 2 | 符号拓展 | 高16位符号拓展 |
| 3 | lui指令拓展 | 高16位置为原立即数，低16位补0 |

## 2.7 DM模块

### 2.7.1 设计总览&模块封装





### 2.7.2 基本描述

4个存储体并行的小端序存储方式存储，容量1KB，RAM实现。输入地址32位，输入数据32位。取输入地址2-9位作为RAM存储地址的位置。

### 2.7.3 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号，存储器中数据置0。  1：置0  0：无效 |
| MemWrite | I | 是否写入存储器中  1：是  0：否 |
| 地址[7:0] | I | 写入的存储器单元地址 |
| 数据输入[31:0] | I | 需要写入的数据 |
| 数据输出[31:0] | O | 地址对应的存储器中读出的数据 |

### 2.7.4 功能描述

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | 当复位信号有效时，存储器中数据置0。 |
| 2 | 写入数据 | 根据地址将数据写入对应的存储器单元 |
| 3 | 读取数据 | 根据地址从存储器单元中读出数据。 |

1. **指令描述**

## 3.1 addu无符号加法

表格

描述已自动生成

## 3.2 subu无符号减法

表格

描述已自动生成

## 3.3 ori或立即数

表格

描述已自动生成

## 3.4 lw加载字

表格

描述已自动生成

表格

描述已自动生成

## 3.5 sw存储字

表格

描述已自动生成

## 3.6 beq相等时转移

表格

描述已自动生成

## 3.7 lui立即数加载到高位

表格

描述已自动生成

## 3.8 j跳转

表格

描述已自动生成

1. **测试程序**

文本

描述已自动生成

1. **测试结果**

将测试程序test-m.asm分别在Mars和logisim中仿真，观察logisim中的仿真结果是否与Mars中相同，从而验证电路设计的正确性。

## 5.1 Mars中仿真结果

### 5.1.1 寄存器数据

表格

描述已自动生成

### 5.1.2 数据存储器数据



## 5.2 logisim中仿真结果

### 5.2.1 寄存器数据

图示

描述已自动生成

### 5.2.2 数据存储器数据

图示

描述已自动生成

## 5.3 结论

测试文件在Mars中与在logisim中仿真结果相同，即电路设计正确。

1. **总结与心得体会**

通过这次开发单周期处理器，我了解了一个处理器是如何工作的，也对于MIPS指令系统每一条指令的构成、作用有了实际的体会。

开发一个处理器，首先需要明确的是需要什么样的硬件基础以及什么样的数据通路，然后再根据数据通路设置控制信号，来完成指定的功能。控制信号的生成依赖于当前指令，即根据当前指令的操作码或功能码来产生对应的指令，控制硬件设备的工作，控制数据传送的路径，最终实现指定的功能。

一开始听到自己要开发一个cpu的时候我感觉非常的不可思议，十分的困难。但听过老师的讲解与自己的思考过后，我从每一个基础的原件开始搭建：从加法器开始、到IM结束。每完成一个部件的搭建，我都对其有了更为透彻的理解和更深的记忆。最后，我使用了3天时间完成了单周期处理器的搭建与测试工作，搭建很成功，这给予了我很大的成就感，也让我对以后的学习更加的有了自信。

总结自己的工作，我认为搭建过程的难点有以下几点：

一是存储器数据位宽、地址位宽、地址来源的选择。

二是跳转指令需要跳转到的地址的计算，即PC的更新问题

三是比较容易忽略的点，即$0寄存器需要指定为常数0，其内容不应该被允许改变，这也正是我初次搭建时忽略、仿真时改正的错误。