|  |
| --- |
| **MIPS多周期处理器设计** |
| **实验报告** |
| |  |  | | --- | --- | | **班号：** | **10011301** | | **学号：** | **2013302488** | | **姓名：** | **张锦阳** | | **电话：** | **18829235904** | |

西北工业大学计算机学院

2016/1/3

**版权声明**

任何收存和保管本文档各种版本的单位和个人，未经本文档作者和本单位同意，不得将本文档转借他人，亦不得随意复制、抄录、拍照或以任何方式传播。否则，引起有碍作者著作权之问题，将可能承担法律责任。

摘 要

此文档是设计基于mips指令的多周期CPU的描述，使用有限状态机描述各个指令所经过的状态，将指令分成不同类型，进入状态机后逐次执行。

多周期是在单周期的基础之上改变而来的，单周期处理器的每一条指令的执行时间相同，但是实际上，指令可以进一步分成不同的几个状态，不同指令执行过程中经过不同状态，所以不同指令执行的时间就不同，lw会经过5个状态，j只有三个状态，如果采用单周期，就会有些指令的某些状态空等待，所以用多周期来消除空等待的时间。

目 录

1 概述 3

1.1 简介 3

2 系统设计 4

2.1 概述 4

2.2 顶层接口 4

2.3 接口时序 5

3 模块设计 6

3.1 概述 6

3.2 ALU模块 6

3.2.1 功能说明 6

3.2.2 接口说明 6

3.2.3 内部结构 6

3.3 Control模块 7

3.3.1 功能说明 7

3.3.2 接口说明 7

3.3.3 内部接口 7

3.4 8

4 修改记录 10

参考文献 11

# 概述

## 简介

MIPS多周期处理器是在单周期处理器结构之上改变而来的，利用有限状态机来实现不同指令的执行。多周期是介于单周期和流水线之间的一种状态，它可以消除单周期某些指令的空等待状态，使执行效率提高，但是不如流水线，可以实现并行处理。

MIPS多周期

# 系统设计

## 概述

此处描述MIPS多周期处理器的系统设计，给出结构框图，例如：

图 1 MIPS多周期处理器结构图

## 顶层接口

此处描述顶层输入、输出接口，格式如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 方向 | 位宽 | 备注 |
| 时钟、复位 | | | |
|  | I | 1 |  |
| 内部接口 | | | |
|  |  |  |  |

## 接口时序

# 模块设计

## 概述

本章节对处理器各模块进行详细设计，有些简单的计算模块可以不做详细说明，重点说明控制状态机的设计。

## PC模块

### 功能说明

主要完成复位操作和接受NPC计算的下一个地址。

### 接口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 方向 | 位宽 | 备注 |
| 时钟、复位 | | | |
| Clk | I | 1 | 时钟 |
| RST | I | 1 | 复位 |
| 内部接口 | | | |
| PCWr | I | 1 | 控制PC模块是否可写  0：寄存器不写  1：寄存器写 |
| NPC | I | 32 | NPC计算的下一个PC |
| PC | O | 32 | 输出 |

### 内部结构

## NPC模块

### 功能说明

计算下一个指令的地址。

### 接口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 方向 | 位宽 | 备注 |
| 内部接口 | | | |
| npc\_op | I | 2 | 控制pc的计算方式，是顺序还是跳转  00：order，顺序跳转  01：branch，条件跳转  10：jump，跳转 |
| NPC | O | 32 | NPC计算的下一个PC的输出 |
| PC | I | 30 | 上一个PC |
| imm | I | 26 | 指令的后26位，用于计算跳转的地址 |

### 内部结构

## IM模块

### 功能说明

指令存储器，根据PC所给的地址取指令。

### 接口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 方向 | 位宽 | 备注 |
| 内部接口 | | | |
| addr | I | 10 | PC给的地址，pc[11:2]  给出32位地址的十位，因为指令存储器我们只给了1024大小，并且是按字节存取的，所以低两位去掉 |
| do | O | 32 | addr对应的指令 |

## IR模块

### 功能说明

传递IM得来的指令，但是有时钟，复位和控制是否可写的信号。

### 接口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 方向 | 位宽 | 备注 |
| 时钟、复位 | | | |
| clk | I | 1 | 时钟 |
| rst | I | 1 | 复位 |
| 内部接口 | | | |
| irwr | I | 1 | 控制IR模块是否可写  0：寄存器不写  1：寄存器写 |
| im\_inst | I | 32 | 从IM传递而来的指令 |
| ir\_inst | O | 32 | 从IR流出的指令 |

## RF模块

### 功能说明

RF模块包含指令存储器，对译码得来的rs，rt，rd进行相应的写入或者读取操作。

### 接口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 方向 | 位宽 | 备注 |
| 时钟、控制 | | | |
| Clk | I | 1 | 时钟 |
| RFWr | I | 1 | 控制可写  0：寄存器不写  1：寄存器写 |
| 内部接口 | | | |
| A1 | I | 6 | rs对应的寄存器号 |
| A2 | I | 6 | rt对应的寄存器号 |
| A3 | I | 6 | rd对应的寄存器号 |
| WD | I | 32 | 要写入寄存器堆的数据  总共有五个选择，所以接一个五选一选择器  分别是来自ALU计算后的结果，DM中读出的结果，PC+4的结果，hi寄存器，lo寄存器 |
| RD1 | I | 32 | A1对应的数据 |
| RD2 | I | 32 | A2对应的数据 |

## EXE模块

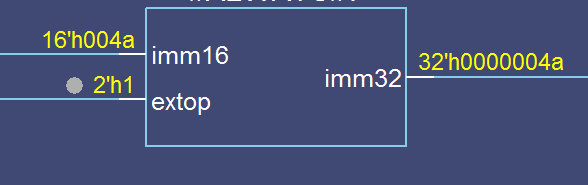
### 功能说明

对得到的16位立即数进行32位扩展。

### 接口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 方向 | 位宽 | 备注 |
| 内部接口 | | | |
| extop | I | 2 | 扩展类型，可以分为三种  00：0扩展  01：符号扩展  10：高位扩展 |
| imm16 | I | 16 | 16位的待扩展立即数 |
| Imm32 | O | 32 | 32位的扩展立即数 |

### 内部结构



## ALU模块

### 功能说明

给出两个源操作数和操作类型，得到输出目的操作数。

ALU可以直接进行的计算：add, addu, sub, subu, or, xor, and, nor, sllv(逻辑可变右移), sll(逻辑右移), srlv(逻辑可变左移), srl(逻辑左移), srav(算术右移), sra(算术可变右移), slt(小于有符号数置一), sltu(小于无符号数置一)。

通过zero的状态，可以确定是否跳转，可以进行beq,bgez,bgtz,blez,bltz,bne是否跳转的判断。

### 接口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 方向 | 位宽 | 备注 |
| 内部接口 | | | |
| aluop | I | 2 | 操作码 |
| A | I | 32 | 输入操作数 |
| B | I | 32 | 输入操作数 |
| C | O | 32 | 输出操作数 |
| zero | O | 7 | 用于条件跳转的标志位  zero[0]=0表示A==B成立，否则为1  zero[1]=0表示A!=B成立，否则为1  zero[2]=0表示A>0成立，否则为1  zero[3]=0表示A>=0成立，否则为1  zero[4]=0表示A<0成立，否则为1  zero[5]=0表示A<=0成立，否则为1  zero[6]=0表示是bgez或者bltz指令，否则为1 |
| shamt | I | 5 | 逻辑移位操作时的移位个数 |
| rt | I | 5 | 判断bgez和bltz的操作类型 |
| exception | O | 1 | 是否有溢出存在 |

## hi\_lo模块

### 功能说明

hi\_lo模块是为了完成mult, multu, div, divu, mthi,mtlo 的指令操作的，对于乘法，lo存高32位，hi存低32位，对于除法，hi存余数，lo存商。

### 接口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 方向 | 位宽 | 备注 |
| 内部接口 | | | |
| A | I | 32 | 第一个操作数 |
| B | I | 32 | 第二个操作数 |
| hi | O | 32 | 表示hi寄存器的输入输出 |
| lo | O | 32 | 表示lo寄存器的输入输出 |
| funct | I | 6 | 确定运算类型 |

## PC\_Sel模块

### 功能说明

PC\_Sel是针对jr和jarl设计的，跳转并链接，跳转至寄存器，指令的源操作地址是从寄存器里来的，所以需要在RF[rs]和原始PC之间选择。

### 接口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 方向 | 位宽 | 备注 |
| 内部接口 | | | |
| PC\_ | I | 30 | 从PC模块过来的地址 |
| GPRA | I | 32 | rs寄存器里存的地址 |
| PC | O | 32 | 选择计算后给NPC模块的地址 |
| op | I | 6 | op和funct用来确定jr和jalr |
| funct | I | 6 |

## DM模块

### 功能说明

DM模块是根据地址对内存(假设数据存储器是内存)进行读写操作。其中sb,sh,sl操作是根据addr的低两位来判断具体该存入哪个地址。

### 接口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 方向 | 位宽 | 备注 |
| 时钟，控制 | | | |
| Clk | I | 1 | 时钟 |
| DMWr | I | 1 | 可写控制  0：模块不写  1：模块可写 |
| 内部接口 | | | |
| addr | I | 12 | 从PC模块过来的地址  如果是sw：要写入的就是in32  如果是sh：addr[1]==1’b0,表示存入到对应地址的低16位  addr[1]==1’b1,表示存入到对应地址的高16位  如果是sb：  addr[1:0]==1’b00,表示存入到对应地址的[7:0]位  addr[1:0]==1’b01,表示存入到对应地址的[15:8]位  addr[1:0]==1’b10,表示存入到对应地址的[23:16]位  addr[1:0]==1’b11,表示存入到对应地址的[31:24]位 |
| In32 | I | 32 | 要写入的32位数 |
| In16 | I | 16 | 要写入的一个字 |
| In8 | I | 8 | 要写入的一个字节 |
| op | I | 6 | 判读操作类型，主要是sb,sh,sw |
| DO | O | 32 | Addr地址对应读出的数据 |

## SW模块

### 功能说明

对要写入DM的数据进行处理，取出它的低8位，16位和32位。

### 接口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 方向 | 位宽 | 备注 |
| 内部接口 | | | |
| in\_ | I | 32 | 输入的32位数据 |
| op | I | 6 | sb:输入的低八位  sh:输入的低十六位  sw:输入的全部 |
| wr | I | 1 | 是否可写的控制 |
| out8 | O | 8 | 输入的低八位 |
| out16 | O | 16 | 输入的低十六位 |
| out32 | O | 32 | 输入的全部 |

## LW模块

### 功能说明

LW模块是对lb, lbu, lh, lhu, lw指令的处理操作，从数据存数器中取到数据，按一定的格式给寄存器，lb和lbu取8位，具体哪八位有addr的后两位来判断，不同的是lb进行符号扩展，lbu进行0扩展; lh和lhu同理。

### 接口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 方向 | 位宽 | 备注 |
| 内部接口 | | | |
| op | I | 6 | 指令的选择，包含lw,lhu,lh,lbu,lb |
| addr | I | 2 | ALU算出的地址的低两位  lw:out=in\_  lhu: addr[1]==1’b0, out={16'b0,in\_[15:0]};  addr[1]==1’b1, out={16'b0,in\_[31:16]};  lh: addr[1]==1’b0, out={{16{in\_[15]}},in\_[15:0]};  addr[1]==1’b1, out={{16{in\_[31]}},in\_[31:16]};  lbu: addr[1:0]==2'b0, out={24'b0,in\_[7:0]};  addr[1:0]==2'b0, out={24'b0,in\_[15:8]};  addr[1:0]==2'b0, out={24'b0,in\_[23:16]};  addr[1:0]==2'b0, out={24'b0,in\_[31:24]};  lb: addr[1:0]==2'b0, out={ 24{in\_[7]}},in\_[7:0]};  addr[1:0]==2'b01, out={{24{in\_[15]}},in\_[15:8]};  addr[1:0]==2'b10, out={{24{in\_[23]}},in\_[23:16]};  addr[1:0]==2'b11, out={{24{in\_[31]}},in\_[31:24]}; |
| in\_ | I | 32 | 输入数据 |
| out | O | 32 | 输出数据 |

## 多选器模块

### jy\_mux4\_gpr\_sel

三选一，说明GPR写入的寄存器号的选择

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 方向 | 位宽 | 备注 |
| 内部接口 | | | |
| D0 | I | 6 | rd |
| D1 | I | 6 | rt |
| D2 | I | 6 | 5’b11111,代表31号寄存器 |
| s | I | 2 | gprsel |
| y | O | 6 | A3 |

### jy\_mux2\_B\_sel

二选一，说明对ALU第二个操作数的选择，是从RF[rd],还是从立即数。

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 方向 | 位宽 | 备注 |
| 内部接口 | | | |
| D0 | I | 32 | rd2 |
| D1 | I | 32 | imm32 |
| s | I | 1 | bsel |
| y | O | 32 | B |

### jy\_mux8\_gpr\_data

五选一，说明对寄存器堆写入数据的选择。

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 方向 | 位宽 | 备注 |
| 内部接口 | | | |
| D0 | I | 32 | aluout |
| D1 | I | 32 | dm\_out |
| D2 | I | 32 | pc+4 |
| D3 |  | 32 | hi |
| D4 |  | 32 | lo |
| s | I | 3 | wdsel |
| y | O | 32 | wd |

## FLOPR模块

**说明：**异步触发器，一共有5个，分别是对ALU的操作数A,B,C, hi\_lo模块hi，lo和DM输出的数据的异步触发。可以实例化为ALUOut、DR等无写使能信号的寄存器。

### FLOPR #(32) jy\_A(.clk(clk),.rst(rst),.d(d1),.p(rd1));

### FLOPR #(32) jy\_B(.clk(clk),.rst(rst),.d(d2),.p(rd2));

### FLOPR #(32) jy\_C(.clk(clk),.rst(rst),.d(aluout),.p(raluout));

### FLOPR #(32) jy\_hi(.clk(clk),.rst(rst),.d(hi),.p(rhi));

### FLOPR #(32) jy\_lo(.clk(clk),.rst(rst),.d(lo),.p(rlo));

### FLOPR #(32) jy\_DR(.clk(clk),.rst(rst),.d(dm\_out1),.p(rdm\_out));

## CTRL模块

### 功能说明

根据op和funct，给出控制信号。

Ctrl模块根据如图所示的状态机使指令运转，具体的可以将指令分成r型，i型，branch型，jump型，加载(lw类)，存储(sw类)。

状态机如下：

**有限状态机**

所有指令都可以归为上状态的一路，具体的：

R：addu，subu，sub，add，and，or，xor，nor，mult，multu，div，divu，sll，sllv，srl，srlv，sra，srav，mfhi，mthi，mflo，mtlo，slt，sltu，jalr，jr

I：addiu，addi，ori，slti，sltiu，andi，xori，lui

SW：sw，sh，sb

LW：lw，lh，lhu，lb，lbu

Branch：beq，bgez，bgtz，blez，bltz，bne

Jump：jal，j

### 接口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 方向 | 位宽 | 备注 |
| 时钟，控制 | | | |
| Clk | I | 1 | 时钟 |
| reset | I | 1 | 复位 |
| 内部接口 | | | |
| op | I | 6 | 当前指令类型 |
| funct | I | 6 | 确定为r型指令后额当前指令类型 |
| zero | I | 7 | Branch的标志选择 |
| PCWr | O | 1 | PC模块是否可写的控制 |
| IRWr | O | 1 | IR模块是否可写的控制 |
| RFWr | O | 1 | RF模块是否可写的控制 |
| DMWr | O | 1 | DM模块是否可写的控制 |
| NPCOp | O | 2 | NPC模块对下一个PC计算类型的选择 |
| ALUOp | O | 6 | ALU模块要计算的运算符的类型 |
| EXEOp | O | 2 | EXT模块对16位立即数扩展类型的选择 |
| GPRSel | O | 2 | RF模块对要写入的寄存器号的选择 |
| WDSel | O | 3 | RF模块对要写入寄存器的数据的选择 |
| BSel | O | 1 | ALU模块对第二个操作数B的选择 |

### 内部结构

Ctrl输入为op，funct，zero，输出控制信号。

MIPS指令集的绝大部分，都可以由op和funct确定，

# 修改记录

此处按照时间依次记录发现的问题以及修改的结果，格式不限，主要用来记录调试过程。

记录格式可以参考以下格式：

2015-11-12 18:09:09

问题：ALU输出值与正确值比较不正确（可以放置Modelsim截图或相关代码）。

解决：将Testbench中比较结果的时间推后半个周期。

# 参考文献

此处放置参考文献。