# 设计文档

## 模块规格

### mips.v

#### 端口说明

表 1‑1 端口说明

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号 |

### datapath.v

#### 端口说明

表 2‑1 端口说明

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| RegDst[1:0] | I | 待回写寄存器地址来源 |
| ALUSrc | I | ALU第二操作数来源 |
| DataSrc[1:0] | I | 待回写数据来源 |
| RegWrite | I | 寄存器写使能信号 |
| MemWrite | I | 数据存储器写使能信号 |
| NPC\_Sel[1:0] | I | 下一指令地址来源 |
| ExtOp[1:0] | I | 立即数扩展方式 |
| Instr[31:0] | O | 当前指令 |

#### 功能定义

表 2‑2 功能定义

|  |  |
| --- | --- |
| 功能 | 描述 |
| grf写入 | 若grf需要写入数据，则输出写入的位置及写入的值。 |
| dm写入 | 若dm需要写入数据，则输出写入的位置及写入的值。 |

### ifu.v

#### 端口说明

表 3‑1 端口说明

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| NPC\_Sel[1:0] | I | 下一指令来源 |
| Z | I | 运算结果为零 |
| Jr[31:0] | I | $ra |
| Clk | I | 时钟信号 |
| Reset | I | 复位信号 |
| Instr[31:0] | O | 指令内容 |
| PC[31:0] | O | 顺序执行下一指令地址 |

#### 功能定义

表 3‑2 功能定义

|  |  |
| --- | --- |
| 功能 | 描述 |
| 复位 | Reset信号有效时，pc同步复位于0x0000\_3000 |
| 存储指令 | 指令容量为32bit×1024字。 |
| 取指 | 输出当前指令，并计算下一指令的地址。 |

表 3‑3 功能定义

|  |  |  |
| --- | --- | --- |
| 功能 | NPC\_sel[1:0] | 描述 |
| Add\_4 | 00 | 顺序指令 |
| Jump | 01 | 跳转至{pc[31:26],Instr[25:0],2’b00} |
| Branch | 10 | 当Z为1时分支至pc+4+{Instr[15:0],2’b00} |
| Jr | 11 | 跳转寄存器 |

### grf.v

#### 端口说明

表 4‑1 端口说明

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| A1[4:0] | I | 第一操作数地址 |
| A2[4:0] | I | 第二操作数地址 |
| A3[4:0] | I | 第三操作数地址 |
| Wd[31:0] | I | 待写入数据 |
| We | I | 写使能信号 |
| Clk | I | 时钟信号 |
| Reset | I | 复位信号 |
| Rd1[31:0] | O | 第一操作数 |
| Rd2[31:0] | O | 第二操作数 |

#### 功能定义

表 4‑2 功能定义

|  |  |
| --- | --- |
| 功能 | 描述 |
| 复位 | Reset信号有效时，全部寄存器同步复位为0 |

### alu.v

#### 端口说明

表 5‑1 端口说明

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| A[31:0] | I | A |
| B[31:0] | I | B |
| ALUOp[2:0] | I | 运算类型 |
| C[31:0] | O | C |
| N | O | (C < 0) |
| Z | O | (C == 0) |
| V | O | 运算溢出 |
| Carry | O | 运算进位 |

#### 功能定义

表 5‑2 功能定义

|  |  |  |
| --- | --- | --- |
| 功能 | ALUOp[2:0] | 描述 |
| Nop | 000 | 0 |
| Add | 001 | A+B |
| Sub | 010 | A-B |
| And | 011 | A&B |
| Or | 100 | A|B |
| Xor | 101 | A^B |
| Le | 110 | A<B |
| B | 111 | B |

### dm.v

#### 端口说明

表 6‑1 端口说明

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| Addr[11:2] | I | 数据地址 |
| Din[31:0] | I | 待写入数据 |
| We | I | 写使能信号 |
| Clk | I | 时钟信号 |
| Reset | I | 复位信号 |
| Dout[31:0] | O | 数据 |

### ext.v

#### 端口说明

表 7‑1 端口说明

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| ExtOp[1:0] | I | 扩展功能 |
| Imm[15:0] | I | 待扩展立即数 |
| Ext[31:0] | O | 扩展结果 |

#### 功能定义u

表 7‑2 功能定义

|  |  |  |
| --- | --- | --- |
| 功能 | ExtOp[1:0] | 描述 |
| {16’b0, Imm} | 00 | 无符号扩展 |
| {{16{Imm[15]}}, Imm} | 01 | 有符号扩展 |
| {Imm, 16’b0} | 10 | 扩展至低位 |
| 32’b0 | 11 | 复位 |

### mux.v

#### RegDst

表 8‑1 RegDst功能定义

|  |  |  |
| --- | --- | --- |
| 信号 | RegDst[1:0] | 数据来源 |
| Instr[15:11] | 00 | R型指令 |
| Instr[20:16] | 01 | I型指令 |
| 31 | 10 | J型指令 |

#### ALUSrc

表 8‑2 ALUSrc功能定义

|  |  |  |
| --- | --- | --- |
| 信号 | ALUSrc | 数据来源 |
| Rd2 | 0 | 寄存器数据 |
| Ext | 1 | 已扩展立即数 |

#### DataSrc

表 8‑3 DataSrc功能定义

|  |  |  |
| --- | --- | --- |
| 信号 | DataSrc[1:0] | 数据来源 |
| C | 00 | ALU |
| Dout | 01 | DM |
| PC | 10 | IFU |

## ctrl.v

### 端口说明

表 1‑1 端口说明

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| OpCode[5:0] | I |  |
| Funct[5:0] | I |  |
| RegDst[1:0] | O | 待回写寄存器地址来源 |
| ALUSrc | O | ALU第二操作数来源 |
| DataSrc[1:0] | O | 待回写数据来源 |
| RegWrite | O | 寄存器写使能信号 |
| MemWrite | O | 数据存储器写使能信号 |
| NPC\_Sel[1:0] | O | 下一指令地址来源 |
| ExtOp[1:0] | O | 立即数扩展方式 |
| ALUOp[2:0] | O | ALU运算方式 |

### 功能定义

表 2‑1 R型指令功能定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Funct | 000000 | 001000 | 100001 | 100011 | 101010 |
| OpCode | 000000 | 000000 | 000000 | 000000 | 000000 |
|  | nop | jr | addu | subu | slt |
| RegDst[1:0] | xx | xx | 00 | 00 | 00 |
| ALUSrc | x | 0 | 0 | 0 | 0 |
| DataSrc[1:0] | xx | xx | 00 | 00 | 00 |
| RegWrite | 0 | 0 | 1 | 1 | 1 |
| MemWrite | 0 | 0 | 0 | 0 | 0 |
| NPC\_Sel[1:0] | 00 | 11 | 00 | 00 | 00 |
| ExtOp[1:0] | xx | xx | xx | xx | xx |
| ALUOp[2:0] | xxx | 100 | 001 | 010 | 110 |

表 2‑2 I型指令功能定义

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| OpCode | 000100 | 001001 | 001101 | 001111 | 100011 | 101011 |
|  | beq | addiu | ori | lui | lw | sw |
| RegDst[1:0] | 01 | 01 | 01 | 01 | 01 | 01 |
| ALUSrc | 0 | 1 | 1 | 1 | 1 | 1 |
| DataSrc[1:0] | xx | 00 | 00 | 00 | 01 | 01 |
| RegWrite | 0 | 1 | 1 | 1 | 1 | 0 |
| MemWrite | 0 | 0 | 0 | 0 | 0 | 1 |
| NPC\_Sel[1:0] | 10 | 00 | 00 | 00 | 00 | 00 |
| ExtOp[1:0] | xx | 01 | 00 | 10 | 01 | 01 |
| ALUOp[2:0] | 010 | 001 | 100 | 111 | 001 | 001 |

表 2‑3 J型指令功能定义

|  |  |  |
| --- | --- | --- |
| OpCode | 000010 | 000011 |
|  | j | jal |
| RegDst[1:0] | xx | 10 |
| ALUSrc | x | x |
| DataSrc[1:0] | xx | 10 |
| RegWrite | 0 | 1 |
| MemWrite | 0 | 0 |
| NPC\_Sel[1:0] | 01 | 01 |
| ExtOp[1:0] | xx | xx |
| ALUOp[2:0] | xxx | xxx |

## 测试程序

测试程序

.text

# addiu

addiu $t0, $zero, 10

# addu

addu $t1, $t0, $t0

# beq

beq $t0, $t0, if\_1\_else

addiu $t0, $zero, 1

if\_1\_else:

addiu $t0, $t0, 1

addiu $s0, $zero, 1

beq $t1, $s0, if\_2\_else

addiu $t0, $zero, 1

if\_2\_else:

addiu $t0, $t0, 1

# subu

subu $t0, $t0, 1

# slt

addiu $s0, $t0, 1

slt $t1, $t0, $s0

slt $t1, $s0, $t0

# ori

addiu $t1, $zero, 6

ori $t2, $t1, 1

# lui

lui $t0, 2

# jal, jr

jal test

addiu $t0, $zero, 1

# j

j jump

addiu $t0, $zero, 8

jump:

# sw

sw $t0, 0x00000000

# lw

lw $t0, 0x00000000

j end

test:

sw $ra, 0x0000004

jr $ra

end:

预期结果

@00003000: $ 8 <= 0000000a

@00003004: $ 9 <= 00000014

@00003010: $ 8 <= 0000000b

@00003014: $16 <= 00000001

@0000301c: $ 8 <= 00000001

@00003020: $ 8 <= 00000002

@00003024: $ 1 <= 00000000

@00003028: $ 1 <= 00000001

@0000302c: $ 8 <= 00000001

@00003030: $16 <= 00000002

@00003034: $ 9 <= 00000001

@00003038: $ 9 <= 00000000

@0000303c: $ 9 <= 00000006

@00003040: $10 <= 00000007

@00003044: $ 8 <= 00020000

@00003048: $31 <= 0000304c

@00003064: \*00000004 <= 0000304c

@0000304c: $ 8 <= 00000001

@00003058: \*00000000 <= 00000001

@0000305c: $ 8 <= 00000001

# 思考题

## 数据通路设计

地址信号来源于ALU运算结果。ALU结果是32位地址，但限于DM容量，且考虑到DM以字为单位存储，低两位不接入DM。因此接入[11:2]。

针对GRF、DM、PC。程序开始运行前，这些部件的初始值应该是0。如果不清零可能会用上次运算的中间值作为本次运算的运算数。

## 控制器设计

### if-else

代码示例：

if (OpCode == 6’b000000) begin

statements

end else if (OpCode == 6’b000001) begin

statements

end …

优缺点：

代码结构性强，但易读性不高。

### assign:

代码示例：

assign RegDst[0] = OpCode[5] | OpCode[4] | OpCode[3] | OpCode[2];

…

优缺点：

程序简洁，但可读性低，需要真值表优化。

### `define:

代码示例：

`define R 6’b000000

`define nop 6’b00000

…

case (OpCode)

`R: begin

case (Funct)

`nop: begin

statements

end

…

endcase

end

`beq: begin

statements

end

…

endcase

优缺点：

代码易读性强，但需要单独文件存储宏定义。

## 在线测试相关信息

addi中如果不考虑溢出则简化为

进一步化简得

即为addiu操作。

构架简单，控制器不需要时序部件。但所有指令都在一个周期内完成，有大量硬件空闲，执行效率低。

jal和jr用于函数调用，函数调用时需要将主函数的变量压栈，函数返回时，需要将其调出。