单周期CPU

设计文档（P4）

贾博驿

初稿：2023年11月5日

编辑：2024年8月26日

设计实现指令

**共41条**

**实际指令39条**

R型指令：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 6 | 5 | 5 | 5 | 5 | 6 |
| opcode | rs | rt | rd | shamt | funct |

I型指令：

|  |  |  |  |
| --- | --- | --- | --- |
| 6 | 5 | 5 | 16 |
| opcode | rs | rt | imm16 |

J型指令：

|  |  |
| --- | --- |
| 6 | 26 |
| opcode | instr\_index |

注：以下的RTL除分支与跳转类指令均省略PC←PC+4。

**算数与逻辑类**

1. 算数类
2. R型

|  |  |  |
| --- | --- | --- |
| 名称 | funct | RTL |
| add | 100000 | GPR[rd] ← GPR[rs] + GPR[rt] |
| addu | 100001 | GPR[rd] ← GPR[rs] + GPR[rt] |
| sub | 100010 | GPR[rd] ← GPR[rs] - GPR[rt] |
| subu | 100011 | GPR[rd] ← GPR[rs] - GPR[rt] |

注：本次要求实现的add实际上为addu、sub实际上为subu。故本次实现中二者按照相同指令处理。

1. I型

|  |  |  |
| --- | --- | --- |
| 名称 | opcode | RTL |
| addiu | 001001 | GPR[rt] ← GPR[rs] + sign\_extend(immediate) |

1. 逻辑类
2. R型

|  |  |  |
| --- | --- | --- |
| 名称 | funct | RTL |
| and | 100100 | GPR[rd] ← GPR[rs] AND(bitwise) GPR[rt] |
| or | 100101 | GPR[rd] ← GPR[rs] OR(bitwise) GPR[rt] |
| xor | 100110 | GPR[rd] ← GPR[rs] XOR(bitwise) GPR[rt] |
| nor | 100111 | GPR[rd] ← GPR[rs] NOR(bitwise) GPR[rt] |
| sll | 000000 | GPR[rd] ← GPR[rt](31-shamt)..0 || 0shamt |
| srl | 000010 | GPR[rd] ← 0shamt || GPR[rt]31..shamt |
| sra | 000011 | GPR[rd] ← (GPR[rt]31)shamt || GPR[rt]31..shamt |
| sllv | 000100 | s ← GPR[rs]4..0 GPR[rd] ← GPR[rt](31-s)..0 || 0s |
| srlv | 000110 | s ← GPR[rs]4..0 GPR[rd] ← 0s || GPR[rt]31..s |
| srav | 000111 | s ← GPR[rs]4..0 GPR[rd] ← (GPR[rt]31)s || GPR[rt]31..s |

注：根据寄存器值移位指令只依照寄存器值的低五位移位，不受高位值影响。

1. I型

|  |  |  |
| --- | --- | --- |
| 名称 | opcode | RTL |
| andi | 001100 | GPR[rt] ← GPR[rs] AND(bitwise) zero\_extend(immediate) |
| ori | 001101 | GPR[rt] ← GPR[rs] OR(bitwise) zero\_extend(immediate) |
| xori | 001110 | GPR[rt] ← GPR[rs] XOR(bitwise) zero\_extend(immediate) |

**寄存器操作类**

1. R型

|  |  |  |
| --- | --- | --- |
| 名称 | funct | RTL |
| slt | 101010 | GPR[rd] ← 0GPRLEN-1 || GPR[rs] < GPR[rt] |
| sltu | 101011 | GPR[rd] ← 0GPRLEN-1 || (0 || GPR[rs]) < (0 || GPR[rt]) |

注：slt类指令成立时置1，**不成立时置0**。

1. I型

|  |  |  |
| --- | --- | --- |
| 名称 | opcode | RTL |
| slti | 001010 | GPR[rt] ← 0GPRLEN-1 || GPR[rs] < sign\_extend(immediate) |
| sltiu | 001011 | GPR[rt] ← 0GPRLEN-1 || (0 || GPR[rs]) < (0 || **sign\_extend**(immediate)) |
| lui | 001111 | GPR[rt] ← immediate || 016 |

注：sltiu对立即数的处理是先**有符号扩展**再进行**无符号比较**。

**分支与跳转类**

1. 分支类（I型）

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | opcode | rt | RTL |
| (all) |  |  | if condition then  PC ← PC + 4 + sign\_extend(offset || 02)  endif |
| bltz | 000001 | 00000 | condition ← GPR[rs] < 032 |
| bgez | 000001 | 00001 | condition ← GPR[rs] ≥ 032 |
| beq | 000100 | rt | condition ← (GPR[rs] = GPR[rt]) |
| bne | 000101 | rt | condition ← (GPR[rs] ≠ GPR[rt]) |
| blez | 000110 | 00000 | condition ← GPR[rs] ≤ 032 |
| bgtz | 000111 | 00000 | condition ← GPR[rs] > 032 |

1. 跳转类
2. R型

|  |  |  |
| --- | --- | --- |
| 名称 | funct | RTL |
| jr | 001000 | PC ← GPR[rs] |
| jalr | 001001 | **I:** GPR[rd] ← PC + 4 PC ← GPR[rs] |

1. J型

|  |  |  |
| --- | --- | --- |
| 名称 | opcode | RTL |
| j | 000010 | PC ← PC31..28 || instr\_index || 02 |
| jal | 000011 | GPR[31] ← PC + 4 PC ← PC31..28 || instr\_index || 02 |

**内存操作类（I型）**

|  |  |  |
| --- | --- | --- |
| 名称 | opcode | RTL |
| (all) |  | addr ← sign\_extend(offset) + GPR[base] |
| lb | 100000 | byte ← addr1..0  GPR[rt] ← sign\_extend(memory[addr]7+8\*byte..8\*byte) |
| lh | 100001 | half ← addr1  GPR[rt] ← sign\_extend(memory[addr]15+8\*half..8\*half) |
| lw | 100011 | GPR[rt] ← memory[addr] |
| lbu | 100100 | byte ← addr1..0  GPR[rt] ← zero\_extend(memory[addr]7+8\*byte..8\*byte) |
| lhu | 100101 | half ← addr1  GPR[rt] ← zero\_extend(memory[addr]15+8\*half..8\*half) |
| sb | 101000 | memory[addr]7..0 ← GPR[rt]7..0 |
| sh | 101001 | memory[addr]15..0 ← GPR[rt]15..0 |
| sw | 101011 | memory[addr]31..0 ← GPR[rt] |

注：

1.针对半字操作需要addr为2的非负整数倍，针对字节操作需要addr为4的非负整数倍，本次实现时实际只涉及针对字节操作，且保证地址是合法的。

2.针对字节和半字操作时只修改对应字节和半字的值，**同一字的其他部分不变**。

模块设计

**IFU**

1. 描述

取指令模块。内有PC子模块、NPC子模块和ROM。接收分支和跳转指令的控制信号和数据并实现跳转。输出指令和PC寄存器取值。

1. 接口

|  |  |  |
| --- | --- | --- |
| 名称 | 方向 | 描述 |
| RESET | I | 同步复位信号 |
| clk | I | 时钟信号 |
| NPCSel[1:0] | I | NPC子模块控制信号 |
| BranchComp | I | NPC子模块分支指令控制信号 |
| offset[15:0] | I | 分支指令跳转偏移 |
| instr\_index[25:0] | I | j、jal指令跳转地址 |
| instr\_register[31:0] | I | jr、jalr指令跳转寄存器值 |
| Instr[31:0] | O | 取出的当前指令 |
| PC[31:0] | O | 当前PC寄存器的值 |

1. 控制信号

BranchComp：接ALU的Comp。

NPCSel：接CTRL的NPCSel。

|  |  |  |
| --- | --- | --- |
| 指令 | 取值 | 描述 |
| 其他 | 00 | PC←PC+4 |
| 分支类 | 01 | 使用offset，结合BranchComp判断 |
| j、jal | 10 | 使用instr\_index |
| jr、jalr | 11 | 使用instr\_register |

**IFU\_PC**

1. 描述

程序计数器子模块。在复位信号有效时复位为32'h00003000。

1. 接口

|  |  |  |
| --- | --- | --- |
| 名称 | 方向 | 描述 |
| RESET | I | 同步复位信号 |
| clk | I | 时钟信号 |
| D[31:0] | O | 下一指令地址 |
| Q[31:0] | O | 当前指令地址 |

**IFU\_NPC**

1. 描述

下一条指令地址计算子模块。输入IFU接收分支跳转指令控制信号和相关数据，计算出下一条指令的地址。

1. 接口

|  |  |  |
| --- | --- | --- |
| 名称 | 方向 | 描述 |
| PC[31:0] | 1 | 当前指令地址 |
| NPCSel[1:0] | I | 同IFU |
| BranchComp | I | 同IFU |
| offset[15:0] | I | 同IFU |
| instr\_index[25:0] | I | 同IFU |
| instr\_register[31:0] | I | 同IFU |
| NPC[31:0] | O | 下一指令地址 |

**SPL**

1. 描述

指令分线器模块。按照三种指令的格式将一条指令分为不同的部分，供其他模块使用。

1. 接口

|  |  |  |
| --- | --- | --- |
| 名称 | 方向 | 描述 |
| Instr[31:0] | 1 | 当前指令 |
| opcode[5:0] | O | 指令的操作数 |
| rs[4:0] | O | R、I型指令的rs |
| rt[4:0] | O | R、I型指令的rt |
| rd[4:0] | O | R型指令的rd |
| shamt[4:0] | O | R型指令的shamt |
| funct[5:0] | O | R型指令的funct |
| imm16[15:0] | O | I 型指令的立即数 |
| instr\_index[25:0] | O | J型指令的跳转地址 |

**GRF**

1. 描述

寄存器堆模块，共有31个寄存器（0号寄存器直接接地）。通过A1、A2输入地址，可以取出指定寄存器存储的数据。当WE使能时，通过A3输入地址，WD输入数据，可以修改指定寄存器存储的数据。

1. 接口

|  |  |  |
| --- | --- | --- |
| 名称 | 方向 | 描述 |
| RESET | I | 同步复位信号 |
| clk | I | 时钟信号 |
| WE | I | 写入使能信号 |
| A1[4:0] | I | 读取的第一个寄存器的编号 |
| A2[4:0] | I | 读取的第二个寄存器的编号 |
| A3[4:0] | I | 写入的寄存器的编号 |
| WD[31:0] | I | 写入寄存器的数据 |
| RD1[31:0] | O | 读取的第一个寄存器的值 |
| RD2[31:0] | O | 读取的第二个寄存器的值 |

1. 控制信号

WE：接CTRL的GRFWE。

|  |  |  |
| --- | --- | --- |
| 指令 | 取值 | 描述 |
| 其他 | 0 | 不涉及寄存器写入的指令 |
| R型：除jr | 1 |  |
| I型：算数与逻辑、寄存器操作类 | 1 |  |
| I型：内存操作类读取类 | 1 |  |
| jal、jalr | 1 |  |

1. 关联MUX控制信号

GRFA3MUX：接CTRL的GRFA3MUX。

|  |  |  |
| --- | --- | --- |
| 指令 | 取值 | 描述 |
| 其他 | 00 | 固定0(0x00)，保护 |
| R型：忽略不使能的jr | 01 | 接入rd |
| I型：忽略不使能的指令 | 10 | 接入rt |
| jal | 11 | 固定31(0x1f) |

GRFWDMUX：接CTRL的GRFWDMUX。

|  |  |  |
| --- | --- | --- |
| 指令 | 取值 | 描述 |
| 其他 | 00 | 接入ALU |
| lui | 01 | 接入EXT |
| I型：内存操作读取类 | 10 | 接入DM |
| jalr、jal | 11 | 接入PC + 4 |

**EXT**

1. 描述

位扩展模块，将16位立即数无符号扩展、有符号扩展、右补16位0为32位。

1. 接口

|  |  |  |
| --- | --- | --- |
| 名称 | 方向 | 描述 |
| EXTSel[1:0] | I | 扩展方式控制信号 |
| imm16[15:0] | I | 16位立即数 |
| EXT[31:0] | O | 32位扩展结果 |

1. 控制信号

EXTSel：接CTRL的EXTSel。

|  |  |  |
| --- | --- | --- |
| 指令 | 取值 | 描述 |
| 其他 | 00 | 有符号拓展 |
| andi、ori、xori | 01 | 无符号拓展 |
| lui | 10 | 右补16位0 |

**ALU**

1. 描述

算数逻辑模块，完成算数运算（不考虑算数溢出的加、减）、逻辑运算（与、或、异或、或非）、移位运算（左移、逻辑右移、算数右移）、比较运算（两数相等、不等、有符号小于、无符号小于，一数有符号小于、小于等于、大于、大于等于零）。计算结果用于保存到寄存器、判断分支判断分支指令是否执行。

1. 接口

|  |  |  |
| --- | --- | --- |
| 名称 | 方向 | 描述 |
| OPSel[1:0] | I | OP输出控制信号 |
| FuncSel[2:0] | I | 计算方式控制信号 |
| CompSel[2:0] | I | 比较部分控制信号 |
| rs[31:0] | I | 第一个操作数 |
| rt[31:0] | I | 第二个操作数 |
| shamt[4:0] | I | 移位立即数 |
| OP[31:0] | O | 计算结果 |
| Comp | O | 比较结果 |

1. 控制信号

OPSel：接CTRL的OPSel。

|  |  |  |
| --- | --- | --- |
| 指令 | 取值 | 描述 |
| add、addu、sub、subu、addiu | 00 | 算数运算 |
| I型：内存操作类 | 00 | 算数运算 |
| and、or、xor、nor、andi、ori、xori | 01 | 逻辑运算 |
| sll、srl、sra、sllv、srlv、srav | 10 | 移位运算 |
| slt、sltu、slti、sltiu | 11 | 比较运算（无符号拓展Comp为32位） |

FuncSel：接CTRL的FuncSel，根据OPSel选择。

OPSel == 00时。

|  |  |  |
| --- | --- | --- |
| 指令 | 取值 | 描述 |
| add、addu、addiu | 000 | 不考虑算数溢出加法 |
| I型：内存操作类读取类 | 000 | 不考虑算数溢出加法 |
| sub、subu | 001 | 不考虑算数溢出减法 |

OPSel == 01时，根据指令是否为I型选择。R型接funct[2:0]、I型接opcode[2:0]。

|  |  |  |
| --- | --- | --- |
| 指令 | 取值 | 描述 |
| and(100100)、andi(001100) | 100 | 与 |
| or(100101)、ori(001101) | 101 | 或 |
| xor(100110)、xori(001110) | 110 | 异或 |
| nor(100111) | 111 | 或非 |

OPSel == 10时，接funct[2:0]。

|  |  |  |
| --- | --- | --- |
| 指令 | 取值 | 描述 |
| sll(000000) | 000 | 逻辑左移立即数 |
| srl(000010) | 010 | 逻辑右移立即数 |
| sra(000011) | 011 | 算数右移立即数 |
| sllv(000100) | 100 | 逻辑左移寄存器 |
| srlv(000110) | 110 | 逻辑右移寄存器 |
| srav(000111) | 111 | 算数右移寄存器 |

OPSel == 11时， FuncSel无效，保持默认000即可。

CompSel：接CTRL的CompSel。

|  |  |  |
| --- | --- | --- |
| 指令 | 取值 | 描述 |
| bltz(00000) | 000 | 有符号小于0 |
| bgez(00001) | 001 | 有符号大于等于0 |
| slt(101010)、slti(001010) | 010 | 有符号两数小于 |
| sltu(101011)、sltiu(001011) | 011 | 无符号两数小于 |
| beq(000100) | 100 | 两数相等 |
| bne(000101) | 101 | 两数不等 |
| blez(000110) | 110 | 有符号小于等于0 |
| bgtz(000111) | 111 | 有符号大于0 |

1. 关联MUX控制信号

ALUrtMUX：接CTRL的ALUrtMUX。

|  |  |  |
| --- | --- | --- |
| 指令 | 取值 | 描述 |
| 其他 | 0 | 接入GRF |
| I型：除分支与跳转类 | 1 | 接入EXT |

**DM**

1. 描述

内存模块。包括RAM和位处理两个部分。位处理部分用于处理针对半字和字节的读取和写入。

1. 接口

|  |  |  |
| --- | --- | --- |
| 名称 | 方向 | 描述 |
| RESET | I | 同步复位信号 |
| clk | I | 时钟信号 |
| WE | I | 写入使能信号 |
| DMSel[2:0] | I | 读取写入模式控制信号 |
| A[31:0] | I | 读取或写入的地址 |
| D\_write[31:0] | I | 写入的数据 |
| D\_read[31:0] | O | 读取的数据 |

1. 控制信号

WE：接CTRL的GRFWE。

|  |  |  |
| --- | --- | --- |
| 指令 | 取值 | 描述 |
| 其他 | 0 |  |
| sb、sh、sw | 1 |  |

DMSel：接CTRL的DMSel，接opcode[2:0]。

|  |  |  |
| --- | --- | --- |
| 指令 | 取值 | 描述 |
| lb(100000)、sb(101000) | 000 | 针对byte操作，**读取时**有符号拓展 |
| lh(100001)、sh(101001) | 001 | 针对halfword操作，**读取时**有符号拓展 |
| lw(100011)、sw(101011) | 011 | 针对word操作 |
| lbu(100100) | 100 | 针对byte操作，读取时无符号拓展 |
| lhu(100101) | 101 | 针对halfword操作，读取时无符号拓展 |

**CTRL**

1. 描述

控制模块。通过opcode、funct和rt判断指令的类型，输出各个模块的控制信号。

1. 接口

|  |  |  |
| --- | --- | --- |
| 名称 | 方向 | 描述 |
| opcode[5:0] | I | 指令的操作数 |
| funct[5:0] | I | R型指令funct |
| rt[4:0] | I | 分支指令rt |
| OPSel[1:0] | O | 接入ALU |
| EXTSel[1:0] | O | 接入EXT |
| DMSel[2:0] | O | 接入DM |
| NPCSel[1:0] | O | 接入IFU |
| FuncSel[2:0] | O | 接入ALU |
| CompSel[2:0] | O | 接入ALU |
| GRFWE | O | 接入GRF |
| DMWE | O | 接入DM |
| GRFA3MUX[1:0] | O | 接入GRF的GRFA3MUX |
| GRFWDMUX[1:0] | O | 接入GRF的GRFWDMUX |
| ALUrtMUX | O | 接入ALU的ALUrtMUX |