P4 单周期处理器（Verilog）设计文档

1. 数据通路设计

表1 数据通路模块定义

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| datapath.v | module datapath(  input Clk,  input Reset,  input [1:0] NPC\_ctrl,  input RegDst,  input RegDst\_2,  input [1:0]grf\_WDsrc,  input RegWr,  input EXT\_ZS,  input ALUsrc,  input [1:0] ALUOp,  input MemWr,  output [5:0] Funct,  output [5:0] Op,  output Eq  ); |

* 1. IFU：内部包括PC，IM和NPC共三个子模块。

表2 IFU端口定义

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| Clk | I | 全局时钟信号（由顶层统一接入时钟） |
| Reset | I | 复位有效时，将PC清零 |
| BrOffset[15:0] | I | Beq指令中的偏移量 |
| JrOffset[31:0] | I | Jr指令偏移量 |
| JalOffset[25:0] | I | Jal指令偏移量 |
| NPC\_ctrl[1:0] | I | 控制NPC模块输出下一条指令在IM中的地址 |
| PC4 | O | 输出PC+4的值 |
| ins[31:0] | O | 输出当前指令 |

表3 IFU功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能定义 | 功能描述 |
| 1 | 加载指令 | 初始化时通过加载raw文件将指令读取到IM中 |
| 2 | 读出指令 | 通过PC输出的地址输出IM中相应指令 |
| 3 | 计算NPC（下一条指令地址） | 用当前地址PC和相关信号计算NPC |

注：PC起始地址为0x00003000；

IM容量为1024字且ROM寻址方式为字地址故而取PC[11:2]

表4 IFU子模块接口及定义

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| pc.v | module pc(  input Clk, //时钟信号  input Reset, //复位信号  input [31:0] Addr, //输入的下一条指令地址  output reg [31:0] Out //输出当前指令地址  ); |
| npc.v | module npc(  input [31:0] PC, //当前指令地址  input [15:0] BrOffset,  input [31:0] JrOffset,  input [25:0] JalOffset,  output [31:0] NPC, //下一条指令地址  output [31:0] PC4,  input [1:0] NPC\_ctrl  ); |
| im.v | module im(  input [11:2] Addr, //当前指令地址  input Clk,  output [31:0] Out //当前指令  ); |

* 1. GRF（通用寄存器组）

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| grf.v | module grf(  input [4:0] A1,  input [4:0] A2,  input [4:0] A3,  input [31:0] WD,  output [31:0] RD1,  output [31:0] RD2,  input Clk,  input RegWr,  input Reset  ); |

表5 GRF模块定义

表6 GRF端口定义

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| A1[4:0] | I | 准备进行读取的第一个寄存器编号 |
| A2[4:0] | I | 准备进行读取的第二个寄存器编号 |
| A3[4:0] | I | 准备写入的寄存器编号 |
| WD[31:0] | I | 准备写入寄存器的32位数据 |
| RD1[31:0] | O | 读取的第一个寄存器中的数据 |
| RD2[31:0] | O | 读取的第二个寄存器中的数据 |
| RegWr | I | 寄存器写入使能 |
| Clk | I | 全局时钟信号 |
| Reset | I | 复位有效时，将所有寄存器清零 |

表7 GRF功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能定义 | 功能描述 |
| 1 | 读取数据 | 输出指定编号（A1,A2）寄存器的内容 |
| 2 | 写入数据 | 将来自于WD的数据写入指定编号（A3）寄存器 |
| 3 | 存储数据 | 将数据保存在寄存器当中即使断电也不会清零 |

注：寄存器总数为 32 个；

0 号寄存器的值保持为 0。

* 1. ALU（算术逻辑单元）:

表8 ALU模块接口定义

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| alu.v | module alu(  input [31:0] A,  input [31:0] B,  output [31:0] Out,  output Eq,  input [1:0] ALUOp  ); |

表9 ALU端口定义

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| A[31:0] | I | ALU进行运算的第一个操作数 |
| B[31:0] | I | ALU进行运算的第二个操作数 |
| Eq | O | 判断两个输入是否相等，相等则输出1 |
| ALUop[1:0] | I | ALU功能选择信号，决定输出哪一个运算结果 |
| Out[31:0] | O | 运算结果输出 |

表10 ALUop信号定义

|  |  |
| --- | --- |
| ALUop | Func |
| 00 | 加法 |
| 01 | 减法 |
| 10 | 比较大小，当且仅当A>B时输出1 |
| 11 | 按位或运算 |

表11 ALU功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能定义 | 功能描述 |
| 1 | 加 | 执行A+B |
| 2 | 减 | 执行A-B |
| 3 | OR | 按位计算A or B |
| 4 | 比较大小 | 当且仅当A > B 输出1 |
| 5 | 判断相等关系 | 当且仅当A = B 输出1 |

* 1. DM（数据存储器）

表12 DM模块接口定义

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| dm.v | module dm(  input Clk,  input MemWr,  input [31:0] WD,  input [31:0] Addr,  input Reset,  output [31:0] RD  ); |

表13 DM端口定义

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| WD[31:0] | I | 将写入内存的数据 |
| Addr[31:0] | I | 准备进行读写的内存地址 |
| MemWr | I | 内存写使能 |
| RD[31:0] | O | 输出指定地址中存放的数据 |
| Clk | I | 全局时钟信号 |
| Reset | I | 复位有效时，将Mem清零 |

表14 DM功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能定义 | 功能描述 |
| 1 | 读取 | 按照指定地址输出存储的数据 |
| 2 | 写入 | 将输入数据写入指定地址内存中 |

注：内存起始地址：0x00000000。

RAM大小共计1024字

* 1. EXT：

表15 EXT模块接口定义

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| ext.v | module ext(  input ExtOp,  input [15:0] In,  output [31:0] Out  ); |

表16 端口定义

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| In[15:0] | I | 输入16位带扩展数字 |
| ExtOp | I | 扩展功能选择 |
| Out[31:0] | O | 输出扩展结果 |

表17 ExtOp信号定义

|  |  |
| --- | --- |
| ExtOp | Func |
| 0 | 零扩展 |
| 1 | 符号扩展 |

表18 EXT功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能定义 | 功能描述 |
| 1 | 零扩展 | 将输入数据扩展为32位并用0填充高16位 |
| 2 | 符号扩展 | 将输入数据扩展为32位并用原数据符号位填充高16位 |

* 1. SH16（16位移位器）

表19 SH16模块接口定义

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| sl16.v | module sl16(  input [15:0] in,  output [31:0] out  ); |

表20 SH16端口定义

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| In[15:0] | I | 输入十六位待移位数据 |
| SH[31:0] | O | 移位结果 |

表21 SH16功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能定义 | 功能描述 |
| 1 | 16位左移 | 将输入数据零扩展为32位后左移指定位数 |

* 1. MUX（多路选择器）

表22 MUX模块接口定义

|  |  |  |
| --- | --- | --- |
| 文件 | 模块 | 模块接口定义 |
| mux.v | mux2\_5 | module mux2\_5( //5位2选1  input [4:0] d1, //输入数据  input [4:0] d2,  input src, //输出选择  output [4:0] out //输出  ); |
| mux4\_32 | module mux4\_32( //32位4选1  input [1:0] src,  input [31:0] d1,  input [31:0] d2,  input [31:0] d3,  input [31:0] d4,  output reg [31:0] out  ); |
| mux2\_32 | module mux2\_32( //32位2选1  input src,  input [31:0] d1,  input [31:0] d2,  output [31:0] out  ); |

1. 控制器设计

表23 Controler模块接口定义

|  |  |
| --- | --- |
| 文件 | 模块端口定义 |
| control.v | module control(  input [5:0] Funct,  input [5:0] Op,  input Eq,  output reg [1:0] NPC\_ctrl,  output RegDst,  output RegDst\_2,  output [1:0] grf\_WDsrc,  output RegWr,  output EXT\_ZS,  output ALUsrc,  output [1:0] ALUOp,  output MemWr  ); |

* 1. 为具有多个输入的模块添加选择信号确定输入来源，并根据指令做出真值表确定其他命令信号

表24 控制信号

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| FUNC | 100001 | 001000 | 100011 | N/A | | | | | |
| OP | 000000 | 000000 | 000000 | 001101 | 100011 | 101011 | 000100 | 001111 | 000011 |
| 控制  信号 | addu | jr | subu | ori | lw | sw | beq | lui | jal |
| RegDst | 0 | 0 | 0 | 1 | 1 | X | X | 1 | 0 |
| RegDst\_2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| grf\_WDsrc | 00 | X | 00 | 00 | 01 | X | X | 10 | 11 |
| RegWr | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| EXT\_ZS | X | X | X | 0 | 1 | 1 | 1 | X | X |
| ALUsrc | 0 | 0 | 0 | 1 | 1 | 1 | 0 | X | X |
| ALUOp | 00 | X | 01 | 11 | 00 | 00 | 10 | X | X |
| MemWr | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| NPC\_ctrl | 00 | 10 | 00 | 00 | 00 | 00 | 01 | 00 | 11 |

* 1. 具体参数

表25 控制器端口定义

|  |  |  |
| --- | --- | --- |
| 信号 | 方向 | 描述 |
| Funct[5:0] | I | 当前指令中的Func字段 |
| Op[5:0] | I | 当前指令中的操作码字段 |
| RegDst | O | 控制数据存储在GRF中的位置，0取rd字段作为地址，1取rt字段 |
| RegDst\_2 | O | 控制数据存储在GRF中的位置，0取RegDst对应输出，1取$31输出 |
| WDsrc[1:0] | O | GRF写入数据来源 |
| ALUsrc | O | ALU操作数来源，0为RD2，1为EXT |
| EXT\_ZS | O | 选择扩展器进行零扩展（0）或者符号扩展（1） |
| RegWr | O | 寄存器堆写使能 |
| MemWr | O | 内存写使能 |
| ALUop[1:0] | O | ALU功能选择信号 |
| NPC\_ctrl[1:0] | O | 控制下一条指令地址 |

表26 GRF写入数据来源控制信号

|  |  |
| --- | --- |
| WDsrc | Src |
| 00 | ALU计算结果 |
| 01 | Mem（内存） |
| 10 | SH（移位器） |
| 11 | PC+4 |

表27 ALU控制信号

|  |  |
| --- | --- |
| ALUop | Func |
| 00 | 加法 |
| 01 | 减法 |
| 10 | 比较大小，当且仅当A>B时输出1 |
| 11 | 按位或运算 |

表28 NPC控制信号

|  |  |
| --- | --- |
| NPC\_ctrl | Src |
| 00 | PC+4 |
| 01 | 输出beq对应地址 |
| 10 | Jr地址 |
| 11 | Jal地址 |

1. 测试程序

ori $t0, $t0, 0x1

addu $t1, $t0, $t0

ori $t2, $t2, 0x2

addu $t3, $t2, $t2

lui $t4, 0x2333

lui $t5, 0x2333

addu $t0, $t0, $t2

addu $t0, $t0, $t4

subu $t5, $t5, $t3

lui $t5, 0x2333

sw $t4, 0x4($t3)

lw $t5, 0x4($t3)

lw $t1, 0x4($t3)

jal jal\_0

subu $t1, $t1, $t5

beq $t1, $zero, beq\_1

jal\_0:

ori $t1, $t1, 0x3333

ori $t5, $t5, 0x3333

jr $ra

beq\_1:

nop

nop

beq $t1, $zero, beq\_1

测试程序期望结果是：

1. 此程序最后进入死循环
2. $ra = 0x00003038
3. $12 = 0x23330000
4. $13 = 0x23333333
5. $10 = 0x00000002
6. $11 = 0x00000004
7. $8 = 0x23330003