Verilog流水线CPU设计文档

一、模块

1.IM

（1）介绍

取指令单元，内部包括指令存储器。

（2）端口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | 输入输出 | 位数 | 描述 |
| PC | I | 32 | 当前PC |
| Instr | O | 32 | 通过当前PC取出的指令 |

2.NPC

（1）介绍

计算下一条指令的逻辑。

（2）端口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | 输入输出 | 位数 | 描述 |
| PCBranch | I | 32 | 分支对应PC |
| PCJ | I | 32 | J、Jal指令对应PC |
| PCJr | I | 32 | Jr指令对应PC（GPR[rs]） |
| PC | I | 32 | 当前PC |
| J | I | 1 | 当前指令是否为J |
| Jal | I | 1 | 当前指令是否为Jal |
| B | I | 1 | 当前指令是否满足跳转条件 |
| NPC | O | 32 | 下一PC |

（3）功能定义

当满足跳转条件时，NPC=PCBranch；否则，当当前指令为J/Jal时，NPC=PCJ；否则，当当前指令为Jr时，NPC=PCJr；否则，NPC=PC+4。

3.PCregF

（1）介绍

PC寄存器。

（2）端口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | 输入输出 | 位数 | 描述 |
| clk | I | 1 | 时钟信号 |
| reset | I | 1 | 重置信号 |
| WE | I | 1 | 使能信号，当WE=1时PC可以写入，否则冻结 |
| NPC | I | 32 | 下一个PC |
| PC | O | 32 | 当前PC |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 清零 | 当reset信号有效时，将PC置为0x00003000 |
| 2 | 写入 | 当时钟信号位于上升沿且使能信号有效时，将NPC赋值到PC |

4.GRF

（1）介绍

通用寄存器组，也称为寄存器文件、寄存器堆。可以存取32位数据。

（2）端口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | 输入输出 | 位数 | 描述 |
| PC | I | 32 | 写入指令对应的PC |
| A1 | I | 5 | 指定32个寄存器中的一个，输出其中数据到R1 |
| A2 | I | 5 | 指定32个寄存器中的一个，输出其中数据到R2 |
| A3 | I | 5 | 指定32个寄存器中的一个，写入Data数据 |
| data | I | 32 | 输入数据 |
| WE | I | 1 | 写入使能信号 1: 可写入 0: 不可写入 |
| reset | I | 1 | 异步复位信号 1: 复位 0: 无效 |
| clk | I | 1 | 时钟信号 |
| R1 | O | 32 | A1指定寄存器中的数据 |
| R2 | O | 32 | A2指定寄存器中的数据 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | 当复位信号有效时，所有寄存器的数据清零 |
| 2 | 读数据 | 读出A1,A2指定寄存器中数据到R1,R2 |
| 3 | 写数据 | 当WE有效且时钟上升沿时，将data写入A3指定寄存器 |

5.CMP

（1）介绍

比较器，比较GRF[rs]和GRF[rt]的大小并输出。

（2）端口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | 输入输出 | 位数 | 描述 |
| data1 | I | 32 | 输入1 |
| data2 | I | 32 | 输入2 |
| equal | O | 1 | 两者是否相等 |
| greater | O | 1 | 输入1是否大于输入2 |
| less | O | 1 | 输入1是否小于输入2 |

6.Controller

（1）介绍

根据指令有关信息（opcode，func）判断指令类型，进而得到各个选择器、使能信号等的数据，决定各组件控制信号。

（2）端口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | 输入输出 | 位数 | 描述 |
| instr | I | 32 | 指令内容 |
| grfWriteAddr | O | 5 | 本条指令写入寄存器的下标（不存在则为0） |
| extOp | O | 2 | EXT的选择信号 |
| aluOp | O | 3 | ALU的选择信号 |
| memToReg | O | 2 | 存入寄存器的数据来源选择信号  0: from DM 1: from ALU 2: from PC+4 |
| aluB | O | 1 | ALU的B端口数据来源选择信号  0: from GRF[rt] 1:from EXT |
| aluA | O | 1 | ALU的A端口数据来源选择信号  0: from GRF[rs] |
| jal | O | 1 | 当前指令是否为jal |
| jr | O | 1 | 当前指令是否为jr |
| j | O | 1 | 当前指令是否为j |
| beq | O | 1 | 当前指令是否为beq |
| dmWE | O | 1 | DM的使能信号 |

7.EXT

（1）介绍

将16位立即数扩展为32位。

（2）端口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | 输入输出 | 位数 | 描述 |
| data | I | 16 | 待扩展的16位立即数 |
| sel | I | 2 | 扩展方式选择信号  00: 无符号扩展  01: 有符号扩展  10: 后面拼接两个0后符号扩展  11: 加载至高位 |
| out | O | 32 | 扩展后的数 |

（3）功能定义

表格 10 EXT功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 扩展 | 按照sel信号选择out为data做什么扩展得到的结果 |

8.ALU

（1）介绍

算术逻辑单元，提供32位加、减、与、或、异或运算，不检测溢出。

（2）端口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | 输入输出 | 位数 | 描述 |
| A | I | 32 | ALU的输入1 |
| B | I | 32 | ALU的输入2 |
| sel | I | 3 | 选择信号  0: A+B 1: A-B 2: A&B 3: A|B 4: A^B |
| out | O | 32 | 运算结果 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 运算 | 按照sel信号选择C为A和B做什么运算得到的结果 |
| 2 | 判零 | 判断C是否为0 |

9.DM

（1）介绍

存储数据。

（2）端口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | 输入输出 | 位数 | 描述 |
| address | I | 5 | 待操作地址 |
| data | I | 32 | 待输入数据 |
| WE | I | 1 | 写入使能信号1: 可写入 0: 不可写入 |
| clk | I | 1 | 时钟信号 |
| reset | I | 1 | 异步复位信号 1: 复位 0: 无效 |
| PC | I | 32 | 当前PC |
| out | O | 32 | 读出的数据 |
| sign | I | 1 | load是否有符号 |
| width | I | 3 | 输入位宽 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 写数据 | 当WE有效且时钟上升沿时，将data写入address地址 |
| 2 | 读数据 | 从address中读取数据，输出至out |
| 3 | 复位 | 当复位信号有效时，所有ROM的数据清零 |

8.IFID

（1）介绍

IF/ID之间的寄存器。

（2）端口定义

表格 5 ALU端口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | 输入输出 | 位数 | 描述 |
| clk | I | 1 | 时钟信号 |
| reset | I | 1 | 重置信号 |
| stall | I | 1 | 暂停信号 |
| instrIn | I | 32 | F中当前指令 |
| PCIn | I | 32 | F中当前PC |
| instrOut | O | 32 | 输出到D中指令 |
| PCOut | O | 32 | 输出到F中PC |

（3）功能定义

表格 6 ALU功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 传递 | 当时钟信号处于上升沿且暂停信号无效时，指令和PC输出赋值为输入 |
| 2 | 清零 | 当重置信号有效时，将输出清零 |

9.IDEX

（1）介绍

ID/EXE之间的寄存器。

（2）端口定义

表格 5 ALU端口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | 输入输出 | 位数 | 描述 |
| clk | I | 1 | 时钟信号 |
| flush | I | 1 | 重置信号 |
| instrIn | I | 32 | F中当前指令 |

另含有grfRs、grfRt、grfWriteAddr、memToReg、dmWE、aluB、aluA、aluOp、extimm、PC、instr的输入输出端口。

10.EXME

（1）介绍

EXE/MEM之间的寄存器。

（2）端口定义

表格 5 ALU端口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | 输入输出 | 位数 | 描述 |
| clk | I | 1 | 时钟信号 |
| reset | I | 1 | 重置信号 |

另含有grfWriteData、grfWriteAddr、memToReg、dmWE、extimm、PC、instr的输入输出端口。

11.MEWB

（1）介绍

MEM/WB之间的寄存器。

（2）端口定义

表格 5 ALU端口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | 输入输出 | 位数 | 描述 |
| clk | I | 1 | 时钟信号 |
| reset | I | 1 | 重置信号 |

另含有dmData、ALUOut、grfWriteData、PC、memToReg、instr的输入输出端口。

12.STALL

（1）介绍

根据当前D级指令判断是否需要暂停。

（2）端口定义

表格 5 ALU端口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | 输入输出 | 位数 | 描述 |
| instrD | I | 32 | ID中指令 |
| instrE | I | 32 | EXE中指令 |
| instrM | I | 32 | MEM中指令 |
| instrW | I | 32 | WB中指令 |
| stall | O | 1 | 当前状态是否需要暂停一周期 |
| mulBusy | I | 1 | 乘除块是否在运算 |

（3）功能定义

根据当前状态的指令分类，分别处理出每一种类型指令对应必须暂停的情况。

13.FORWARD

（1）介绍

处理转发对应的标记。

（2）端口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | 输入输出 | 位数 | 描述 |
| instrD | I | 32 | ID中指令 |
| instrE | I | 32 | EXE中指令 |
| instrM | I | 32 | MEM中指令 |
| instrW | I | 32 | WB中指令 |
| flagE | O | 1 | E中数据是否准备完毕 |
| flagM | O | 1 | M中数据是否准备完毕 |
| flagW | O | 1 | W中数据是否准备完毕 |
| addrE | O | 5 | E中数据要写入的寄存器 |
| addrM | O | 5 | M中数据要写入的寄存器 |
| addrW | O | 5 | W中数据要写入的寄存器 |
| dataE | O | 2 | E中数据的来源 0: from PCE+8 1: from extimmE |
| dataM | O | 2 | M中数据的来源  0: from ALUOutM 1: from PCE+8 2: from extimmE |

14.hilo

（1）介绍

乘除块。

（2）端口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | 输入输出 | 位数 | 描述 |
| clk | I | 1 | 时钟信号 |
| reset | I | 1 | 重置信号 |
| A | I | 32 | 输入1 |
| B | I | 32 | 输入2 |
| instr | I | 32 | E中指令 |
| out | O | 32 | 结果 |
| busy | O | 1 | 是否在运算 |
| start | O | 1 | 是否刚进来一个muldiv |

（3）功能定义

二、测试程序

自动生成。

三、思考题

1. 为什么需要有单独的乘除法部件而不是整合进ALU？为何需要有独立的HI、LO寄存器？

乘除法运算慢。在E级使用，不影响GRF。

2. 参照你对延迟槽的理解，试解释“乘除槽”。

当乘除法正在进行运算时，将与乘除有关的指令阻塞在D级，即进入槽内但被阻塞，和延迟槽类似。

3. 举例说明并分析何时按字节访问内存相对于按字访问内存性能上更有优势。（Hint： 考虑C语言中字符串的情况）

当字节访问多的时候更有优势（如字符串操作）

4. 如何概括你所设计的CPU的设计风格？为了对抗复杂性你采取了哪些抽象和规范手段？

命名规范（如instrW，readRsD），首单词首字母小写，后面首字母大写。流水线寄存器一律使用xxx，xxxOut命名，方便观看。

将指令放在interpreter中分类，这样可以在pause和forward模块中快速引用，改动方便。

5、 你对流水线CPU设计风格有何见解？

自己觉得哪种好改就写哪种得了。我觉得自己这种就挺好改的。

6. 在本实验中你遇到了哪些不同指令类型组合产生的冲突？你又是如何解决的？相应的测试样例是什么样的？

和P5几乎一样。列表，在stall里判断是否必须暂停，在forward中判断在哪里可以转发、转发的选择信号是什么。

测试样例主要是自动生成的。