计算机组成原理P4 CPU设计实验报告

一、整体结构：

EXT符号扩展ALU算术逻辑单元DM数据存储器GRF通用寄存器IM取指令单元PC程序计数器control控制器nextPC指令地址计算 4个MUX和大佬写的测试显示指令的DASM组成

处理器为32位处理器

处理器应支持的指令集为：{[addu, subu,sll, ori,lui, lw, sw, beq, bne,j, jal, jr,slt](https://github.com/3BPM/CO-study/commit/b6b61da04f53bc76750f8febbe5fd0bc353e8e62) }。

为了支持p2全排列代码我自己加了3指令

二、模块规格：

（一）模块规格撰写

1．PC（程序计数器）

器件：32bit寄存器

模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号  1：有效  0：无效 |
| PCin[31:0] | I | nPC |
| Out\_PC[31:0] | O | 当前PC值 |

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，PC被设置为起始地址0x00000000 |
| 2 | 更新PC | 根据PCin在时钟上升沿时候更新为下一周期正确的PC |

2. 1．nextPC（程序计数器）

模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号  1：有效  0：无效 |
| PCin[31:0] | I | nPC |
| Jal | I |  |
| Jr | I |  |
| beq | I |  |
| Out\_PC[31:0] | O | 当前PC值 |

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，PC被设置为起始地址0x00000000 |
| 2 | 更新PC | 根据PCin在时钟上升沿时候更新为下一周期正确的PC |

2．IM（指令存储器）

器件：1024\*32bitROM 4KB大小

因为ROM中储存了1024个地址，且IM实际地址宽度仅为10位，

采用的是将PC[31：0]中[11:2]位作为ROM的地址，取指令。然后分了一下

模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| program counter address[31：0] | I | program counter address |
| Op[31:26] | O | 6bitOp |
| rs[25:21] | O | 5bitrs |
| Rt[20:16] | O | 5bit rt |
| Rd[15:11] | O | 5bit rd |
| S [10:6] | O | 5bit s slt时候使用 |
| Funct[5:0] | O | 6bit funct |
| Imm[15:0] | O | 16bit [immediate](javascript:;) [operand](javascript:;) or offset |
| Imm26[25:0] | O | Jal j时候使用 |
| Instr[31:0] | O | 32位MIPS指令 |

功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 取指令 | 根据PC从IM中取出指令 |

2.GRF（通用寄存器组）：内部包括32个寄存器

具有写使能的寄存器实现，寄存器总数为 32 个

**0 号寄存器**的值始终保持为 0。其他寄存器初始值均为 0

模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| WD | I | 读写控制信号  1：写操作  0：不可写 |
| Clk | I | 时钟信号 |
| Reset | I | 复位信号  1：有效  0：无效 |
| A1 | I | 读寄存器1的地址 |
| A2 | I | 读寄存器2的地址 |
| A3 | I | 写寄存器的地址 |
| RD1 | O | 32位输出1 |
| RD2 | O | 32位输出2 |

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，所有寄存器的值被设置为0x00000000 |
| 2 | 写寄存器 | 根据输入的写寄存器地址，把输入的数据写入写寄存器中 |
| 3 | 读寄存器 | 根据输入的读寄存器地址，将数据读出 |

3.ALU（算术逻辑单元）

提供32位加、减、或运算

可以不支持溢出

模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| A[31:0] | I | ALU32位输入数据A |
| B[31:0] | I | ALU32位输入数据B |
| ALUOp[2:0] | I | ALU功能选择信号  00:加法  01:减法  10:或运算  11：左移  100：slt比较 |
| Out | O | 32位数据输出 |
| Iseq | O | Beq的时候用到1代表相等 |

功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 或 | A|B |
| 2 | 减 | A-B |
| 3 | 加 | A+B |
| 4 | 左移 | Sll |
| 5 | 比较 | Slt |

4.DM（数据存储器）

使用RAM实现，容量为32bit\*32,采用双端口模式

起始地址：0x00000000

RAM 应使用双端口模式，即设置 RAM 的 Data Interface 属性为 Separate load and store ports

模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号  1：有效  0：无效 |
| Address[31:0] | I | 需访问的32位RAM地址 按照IM同理的方法连接MemAddr |
| WData[31:0] | I | 写入数据的输入 |
| RData | O | 读取数据的输出 |
| ReadEn | I | 可能有用还未开发 |
| WriteEn | I | 读写控制信号  1：可以进行写操作 |

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，所有数据被设置为0x00000000 |
| 2 | 写操作 | 根据输入的寄存器地址，把输入的数据写入 |
| 3 | 读操作 | 根据输入的寄存器地址，将其中的数据读出 |
| 4 | 32-5的转换 | 为了正确输出MemAddr |

1. EXT（位扩展器）：

模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Immin[15:0] | I | 16位imm数据输入 |
| Result[31:0] | O | 位扩展后的32位输出 |
| ExtOp[1:0] | I | 位扩展选择信号  00:高位补0（0扩展）  01：高位符号扩展  10：低位补0（0扩展左移16位） |

功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 高位补0 | 高16位补0 |
| 2 | 低位补0 | 低16位补0 |
| 3 | 符号扩展 | 若符号位为0，则高位补0  若符号位为1，则高位补1 |

6.控制器（Control）

模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| op[5:0] | I | 6位op |
| func[5:0] | I | 6位function |
| ALUOp[1:0] | O | 控制ALU进行相应运算 |
| RegDst[1:0] | O | //00means write GRF is 2th reg in Instr (rt)  //01means is 3th (rd)  //10means write GRF is 31 11111 ra  //这玩意控制一个MUX决定用那个地址GRF |
| ALUsrc | O | ALU的B操作数 0是grf中值 1是立即数 |
| Whattoreg[1:0] | O | //00means write data GRF is from ALU  //01:write GRF is DM  //10 means write data from PC+4  //这玩意控制一个MUX决定用那个数据写GRF |
| RegWrite | O | GRF中的writeEn |
| MemWrite | O | DM中的writeEn |
| PCsrc | O | nextPC选择信号 |
| Extop | O | 控制扩展方式 |
| ALUOp[2:0] | O | 控制ALU进行相应运算 |
| Need\_ALUC | O | 是R型指令的话需要ALUcontrol进一步判断 |
| jr | O | jr |
| beq | O | beq |
| jal | O | jal |
| bne | O | bne |

（二）思考题

1.根据你的理解，在下面给出的 DM 的输入示例中，地址信号 addr 位数为什么是 [11:2] 而不是 [9:0]？这个 addr 信号又是从哪里来的？

答案见DM和IM。

2. 思考 Verilog 语言设计控制器的译码方式，给出代码示例，并尝试对比各方式的优劣。

3.在相应的部件中，**reset 的优先级**比其他控制信号（不包括 clk 信号）都要**高**，且相应的设计都是**同步复位**。清零信号 reset 所驱动的部件具有什么共同特点？

是类似于状态机 有寄存器。不光是组合电路，组合电路不要清零。

4.C 语言是一种弱类型程序设计语言。C 语言中不对计算结果溢出进行处理，这意味着 C 语言要求程序员必须很清楚计算结果是否会导致溢出。因此，如果仅仅支持 C 语言，MIPS 指令的所有计算指令均可以忽略溢出。 请说明为什么在忽略溢出的前提下，addi 与 addiu 是等价的，add 与 addu 是等价的。提示：阅读《MIPS32® Architecture For Programmers Volume II: The MIPS32® Instruction Set》中相关指令的 Operation 部分 。

因为是补码啊 addiu无非是0x0-0xffff ffff ,addu无非是负数0x8000 0000-0xffff ffff 正数0x0-0x7fff ffff

5.根据自己的设计说明单周期处理器的优缺点。

缺点是必须跑完所有模块一圈才能运行下一条指令，许多部件在摸鱼没有最高效利用

优点是简单。