流水线CPU设计文档(P6)

17373252 丁禹衡

1. 模块规格
2. PC（程序计数器）

端口说明：

表1 PC端口说明

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号，将PC置为0x00003000  1：复位  0：无效 |
| En | I | 使能信号  1：可写入  0：不可写入 |
| nPC[31:0] | I | 下一条指令地址 |
| PC[31:0] | O | 当前指令地址 |

功能定义：

表2 PC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0x00003000 |
| 2 | 计数 | 当En有效且时钟上升沿到来时，PC更新为nPC的输入值 |

1. IM（指令存储器）

端口说明：

表3 IM端口说明

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| A[31:0] | I | 当前指令地址 |
| Instr[31:0] | O | A指定的当前指令 |

功能定义：

表4 IM功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 取指令 | 输出A指定的当前指令 |

3. GRF（通用寄存器文件）

端口说明：

表5 GRF端口说明

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号，将32个寄存器中的值全部清零  1：复位  0：无效 |
| WE | I | 写使能信号  1：可向GRF中写入数据  0：不能向GRF中写入数据 |
| A1[4:0] | I | 5位地址输入信号，指定32个寄存器中的一个，将其中存储的数据读出到RD1 |
| A2[4:0] | I | 5位地址输入信号，指定32个寄存器中的一个，将其中存储的数据读出到RD2 |
| A3[4:0] | I | 5位地址输入信号，指定32个寄存器中的一个，作为写入的目标寄存器 |
| WD[31:0] | I | 32位数据输入信号 |
| RD1[31:0] | O | 输出A1指定的寄存器中的32位数据 |
| RD2[31:0] | O | 输出A2指定的寄存器中的32位数据 |

功能定义：

表6 GRF功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | reset信号有效时，所有寄存器存储的数值清零 |
| 2 | 读数据 | 读出A1,A2地址对应寄存器中所存储的数据到RD1,RD2 |
| 3 | 写数据 | 当WE有效且时钟上升沿来临时，将WD写入A3所对应的寄存器中 |

4. ALU（算术逻辑单元）

端口说明：

表7 ALU端口说明

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| A[31:0] | I | 参与ALU计算的第一个值 |
| B[31:0] | I | 参与ALU计算的第二个值 |
| Op[2:0] | I | ALU功能选择信号  0：Out = A + B  1：Out = A - B  2：Out = A & B  3：Out = A | B  4：Out = A ^ B  5：Out = ~(A | B)  6：Out = B << A[4:0]  7：Out = B >> A[4:0]  8：Out = $signed(B) >>> A[4:0]  9：Out = ($signed(A) < $signed(B)) ? 1 : 0  10：Out = (A < B) ? 1 : 0 |
| Out[31:0] | O | ALU的计算结果 |

功能定义：

表8 ALU功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 加运算 | Out = A + B |
| 2 | 减运算 | Out = A - B |
| 3 | 与运算 | Out = A & B |
| 4 | 或运算 | Out = A | B |
| 5 | 异或运算 | Out = A ^ B |
| 6 | 或非运算 | Out = ~(A | B) |
| 7 | 逻辑左移 | Out = B << A[4:0] |
| 8 | 逻辑右移 | Out = B >> A[4:0] |
| 9 | 算术右移 | Out = $signed(B) >>> A[4:0] |
| 10 | 小于置1（有符号） | Out = ($signed(A) < $signed(B)) ? 1 : 0 |
| 11 | 小于置1（无符号） | Out = (A < B) ? 1 : 0 |

5. DM（数据存储器）

端口说明：

表9 DM端口说明

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号，将DM清零  1：复位  0：无效 |
| WE | I | 写使能信号  1：可向DM中写入数据  0：不能向DM中写入数据 |
| StoreType[1:0] | I | 存储指令类型信号  0：存储字  1：存储半字  2：存储字节 |
| LoadType[2:0] | I | 加载指令类型信号  0：加载字  1：加载半字  2：加载半字（无符号）  3：加载字节  4：加载字节（无符号） |
| A[31:0] | I | 32位地址输入信号，指定读出或写入数据的地址 |
| WD[31:0] | I | 32位数据输入信号 |
| RD[31:0] | O | 输出A指定的32位数据 |

功能定义：

表10 DM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，将DM中数据清零 |
| 2 | 读数据 | 读出A所指定的数据到RD |
| 3 | 写数据 | 当WE有效且时钟上升沿到来时，将输入数据WD写入A所指定的地址 |

6. EXT（位扩展单元）

端口说明：

表11 EXT端口说明

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| In[15:0] | I | 16位输入数据 |
| Op[1:0] | I | 位扩展方式选择信号  0：符号扩展  1：零扩展  2：加载至高位 |
| Out[31:0] | O | 扩展后的32位输出数据 |

功能定义：

表12 EXT功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 符号扩展 | 将16位输入数据进行符号扩展，输出32位数据 |
| 2 | 零扩展 | 将16位输入数据进行零扩展，输出32位数据 |
| 3 | 加载至高位 | 将16位输入数据加载至高16位，并将低16位置0 |

7. PCCAL（指令地址计算单元）

端口说明：

表13 PCCAL端口说明

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Base[31:0] | I | 分支指令的基地址（PC+4） |
| ImmB[31:0] | I | 分支指令的偏移量 |
| ImmJ[25:0] | I | 跳转指令的目的地址的中间26位 |
| Cmp1[31:0] | I | 分支指令的第一个比较数 |
| Cmp2[31:0] | I | 分支指令的第二个比较数 |
| CmpOp[2:0] | I | 分支指令类型选择信号  0：beq  1：bne  2：blez  3：bgtz  4：bltz  5：bgez |
| BranchA[31:0] | O | 分支指令计算出的下一条指令的地址 |
| JumpA[31:0] | O | 跳转指令计算出的下一条指令的地址 |

功能定义：

表14 PCCAL功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 计算分支地址 | 根据CmpOp比较Cmp1、Cmp2、0计算分支地址 |
| 2 | 计算跳转地址 | 通过位拼接操作计算跳转地址 |

1. MDU（乘除单元）

端口说明：

表15 MDU端口说明

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号，将MDU中的寄存器全部清零  1：复位  0：无效 |
| Start | I | 乘除指令启动信号  1：启动  0：无效 |
| RDsel | I | 读寄存器选择信号  0：读LO寄存器  1：读HI寄存器 |
| A[31:0] | I | 参与MDU计算的第一个值 |
| B[31:0] | I | 参与MDU计算的第二个值 |
| Op[2:0] | I | MDU功能选择信号  0：无操作  1：{HI, LO} = $signed(A) \* $signed(B)  2：{HI, LO} = A \* B  3：LO = $signed(A) / $signed(B)  HI = $signed(A) % $signed(B)  4：LO = A / B  HI = A % B  5：LO = A  6：HI = A |
| Out[31:0] | O | MDU的计算结果 |
| Busy | O | MDU繁忙信号  1：繁忙，不可执行乘除相关指令  0：空闲，可执行乘除相关指令 |

功能定义：

表16 MDU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，将MDU中的寄存器全部清零 |
| 2 | 有符号乘 | {HI, LO} = $signed(A) \* $signed(B) |
| 3 | 无符号乘 | {HI, LO} = A \* B |
| 4 | 有符号除 | LO = $signed(A) / $signed(B)  HI = $signed(A) % $signed(B) |
| 5 | 无符号除 | LO = A / B  HI = A % B |
| 6 | 写LO寄存器 | LO = A |
| 7 | 写HI寄存器 | HI = A |
| 8 | 读LO寄存器 | Out = LO |
| 9 | 读HI寄存器 | Out = HI |

1. HZD（冲突处理单元）

端口说明：

表17 HZD端口说明

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Instr\_D[31:0] | I | 位于D级的指令 |
| Instr\_E[31:0] | I | 位于E级的指令 |
| Instr\_M[31:0] | I | 位于M级的指令 |
| Instr\_W[31:0] | I | 位于W级的指令 |
| Start | I | 乘除指令启动信号 |
| Busy | I | MDU繁忙信号 |
| RS\_D\_MUXsel[2:0] | O | 位于D级的寄存器RS需求转发多选器的选择信号 |
| RT\_D\_MUXsel[2:0] | O | 位于D级的寄存器RT需求转发多选器的选择信号 |
| RS\_E\_MUXsel[2:0] | O | 位于E级的寄存器RS需求转发多选器的选择信号 |
| RT\_E\_MUXsel[2:0] | O | 位于E级的寄存器RT需求转发多选器的选择信号 |
| RT\_M\_MUXsel | O | 位于M级的寄存器RT需求转发多选器的选择信号 |
| Stall | O | 暂停信号 |

功能定义：

表18 HZD功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 转发 | 根据各流水级的寄存器供需关系检测冲突并通过转发处理 |
| 2 | 暂停 | 根据各流水级的寄存器供需关系检测冲突并通过暂停处理 |

二、控制器设计

端口说明：

表19 CTRL端口说明

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Instr[31:0] | I | 当前流水级的指令 |
| PC\_MUXsel[2:0] | O | PC的nPC端口数据源选择信号  0：来源为PC + 4  1：来源为PCCAL的BeqA端口输出  2：来源为PCCAL的JumpA端口输出  3：来源为RS\_D\_MUX转发多选器的输出 |
| EXTsel[1:0] | O | EXT扩展方式选择信号（详见EXT端口说明） |
| GRF\_A3\_MUXsel[1:0] | O | GRF的A3端口数据源选择信号  0：来源为指令的rd字段  1：来源为指令的rt字段  2：来源为常量31 |
| PCCALsel[2:0] | O | PCCAL中分支指令类型选择信号（详见PCCAL端口说明） |
| ALUsel[1:0] | O | ALU功能选择信号（详见ALU端口说明） |
| ALU\_A\_MUXsel | O | ALU的A端口数据源选择信号  0：来源为RS\_E\_MUX转发多选器的输出  1：来源为当前E级指令的[10:6]字段 |
| ALU\_B\_MUXsel | O | ALU的B端口数据源选择信号  0：来源为RT\_E\_MUX转发多选器的输出  1：来源为EXT的输出 |
| MDUsel[2:0] | O | MDU功能选择信号（详见MDU端口说明） |
| Start | O | 乘除指令启动信号 |
| MDU\_RDsel | O | MDU中读寄存器选择信号  0：读LO寄存器  1：读HI寄存器 |
| ALU\_MDU\_MUXsel | O | 运算结果选择信号  0：取ALU的运算结果  1：取MDU的运算结果 |
| DM\_WE | O | DM写使能信号  1：可向DM中写入数据  0：不能向DM中写入数据 |
| StoreType[1:0] | O | DM中存储指令类型信号（详见DM端口说明） |
| LoadType[2:0] | O | DM中加载指令类型信号（详见DM端口说明） |
| GRF\_WD\_MUXsel[1:0] | O | GRF的WD端口数据源选择信号  0：来源为W级流水线寄存器中的ALU输出数据  1：来源为W级流水线寄存器中的DM输出数据  2：来源为W级流水线寄存器中的EXT输出数据  3：来源为W级流水线寄存器中的PC+8值 |
| GRF\_WE | O | GRF写使能信号  1：可向GRF中写入数据  0：不能向GRF中写入数据 |

功能定义：

表20 CTRL真值表（1）

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Funct | 100000 | 100001 | 100010 | 100011 | 100100 | 100101 | 100110 | 100111 | 000100 |
| Op | 000000 | 000000 | 000000 | 000000 | 000000 | 000000 | 000000 | 000000 | 000000 |
| rt |  | | | | | | | | |
|  | add | addu | sub | subu | and | or | xor | nor | sllv |
| PC\_MUXsel[2:0] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| EXTsel[1:0] | X | X | X | X | X | X | X | X | X |
| GRF\_A3\_MUXsel[1:0] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCCALsel[2:0] | X | X | X | X | X | X | X | X | X |
| ALUsel[1:0] | 0 | 0 | 1 | 1 | 2 | 3 | 4 | 5 | 6 |
| ALU\_A\_MUXsel | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ALU\_B\_MUXsel | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| MDUsel[2:0] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Start | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| MDU\_RDsel | X | X | X | X | X | X | X | X | X |
| ALU\_MDU\_MUXsel | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| DM\_WE | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| StoreType[1:0] | X | X | X | X | X | X | X | X | X |
| LoadType[2:0] | X | X | X | X | X | X | X | X | X |
| GRF\_WD\_MUXsel[1:0] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| GRF\_WE | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

表20 CTRL真值表（2）

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Funct | 000110 | 000111 | 000000 | 000010 | 000011 |  | | | |
| Op | 000000 | 000000 | 000000 | 000000 | 000000 | 001000 | 001001 | 001100 | 001101 |
| rt |  | | | | | | | | |
|  | srlv | srav | sll | srl | sra | addi | addiu | andi | ori |
| PC\_MUXsel[2:0] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| EXTsel[1:0] | X | X | X | X | X | 0 | 0 | 1 | 1 |
| GRF\_A3\_MUXsel[1:0] | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| PCCALsel[2:0] | X | X | X | X | X | X | X | X | X |
| ALUsel[1:0] | 7 | 8 | 6 | 7 | 8 | 0 | 0 | 2 | 3 |
| ALU\_A\_MUXsel | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| ALU\_B\_MUXsel | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| MDUsel[2:0] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Start | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| MDU\_RDsel | X | X | X | X | X | X | X | X | X |
| ALU\_MDU\_MUXsel | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| DM\_WE | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| StoreType[1:0] | X | X | X | X | X | X | X | X | X |
| LoadType[2:0] | X | X | X | X | X | X | X | X | X |
| GRF\_WD\_MUXsel[1:0] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| GRF\_WE | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

表20 CTRL真值表（3）

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Funct |  | | | | 001000 | 001001 |  | | |
| Op | 001110 | 001111 | 000010 | 000011 | 000000 | 000000 | 000100 | 000101 | 000110 |
| rt |  | | | | | | | | |
|  | xori | lui | j | jal | jr | jalr | beq | bne | blez |
| PC\_MUXsel[2:0] | 0 | 0 | 2 | 2 | 3 | 3 | 1 | 1 | 1 |
| EXTsel[1:0] | 1 | 2 | X | X | X | X | 0 | 0 | 0 |
| GRF\_A3\_MUXsel[1:0] | 1 | 1 | X | 2 | X | 0 | X | X | X |
| PCCALsel[2:0] | X | X | X | X | X | X | 0 | 1 | 2 |
| ALUsel[1:0] | 4 | X | X | X | X | X | X | X | X |
| ALU\_A\_MUXsel | 0 | X | X | X | X | X | X | X | X |
| ALU\_B\_MUXsel | 1 | X | X | X | X | X | X | X | X |
| MDUsel[2:0] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Start | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| MDU\_RDsel | X | X | X | X | X | X | X | X | X |
| ALU\_MDU\_MUXsel | 0 | X | X | X | X | X | X | X | X |
| DM\_WE | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| StoreType[1:0] | X | X | X | X | X | X | X | X | X |
| LoadType[2:0] | X | X | X | X | X | X | X | X | X |
| GRF\_WD\_MUXsel[1:0] | 0 | 2 | X | 3 | X | 3 | X | X | X |
| GRF\_WE | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |

表20 CTRL真值表（4）

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Funct |  | | | 101010 | 101011 |  | | | |
| Op | 000111 | 000001 | 000001 | 000000 | 000000 | 001010 | 001011 | 101011 | 101001 |
| rt |  | 00000 | 00001 |  | | | | | |
|  | bgtz | bltz | bgez | slt | sltu | slti | sltiu | sw | sh |
| PC\_MUXsel[2:0] | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| EXTsel[1:0] | 0 | 0 | 0 | X | X | 0 | 0 | 0 | 0 |
| GRF\_A3\_MUXsel[1:0] | X | X | X | 0 | 0 | 1 | 1 | X | X |
| PCCALsel[2:0] | 3 | 4 | 5 | X | X | X | X | X | X |
| ALUsel[1:0] | X | X | X | 9 | 10 | 9 | 10 | 0 | 0 |
| ALU\_A\_MUXsel | X | X | X | 0 | 0 | 0 | 0 | 0 | 0 |
| ALU\_B\_MUXsel | X | X | X | 0 | 0 | 1 | 1 | 1 | 1 |
| MDUsel[2:0] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Start | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| MDU\_RDsel | X | X | X | X | X | X | X | X | X |
| ALU\_MDU\_MUXsel | X | X | X | 0 | 0 | 0 | 0 | 0 | 0 |
| DM\_WE | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| StoreType[1:0] | X | X | X | X | X | X | X | 0 | 1 |
| LoadType[2:0] | X | X | X | X | X | X | X | X | X |
| GRF\_WD\_MUXsel[1:0] | X | X | X | 0 | 0 | 0 | 0 | X | X |
| GRF\_WE | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |

表20 CTRL真值表（5）

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Funct |  | | | | | | 011000 | 011001 | 011010 |
| Op | 101000 | 100011 | 100001 | 100101 | 100000 | 100100 | 000000 | 000000 | 000000 |
| rt |  | | | | | | | | |
|  | sb | lw | lh | lhu | lb | lbu | mult | multu | div |
| PC\_MUXsel[2:0] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| EXTsel[1:0] | 0 | 0 | 0 | 0 | 0 | 0 | X | X | X |
| GRF\_A3\_MUXsel[1:0] | X | 1 | 1 | 1 | 1 | 1 | X | X | X |
| PCCALsel[2:0] | X | X | X | X | X | X | X | X | X |
| ALUsel[1:0] | 0 | 0 | 0 | 0 | 0 | 0 | X | X | X |
| ALU\_A\_MUXsel | 0 | 0 | 0 | 0 | 0 | 0 | X | X | X |
| ALU\_B\_MUXsel | 1 | 1 | 1 | 1 | 1 | 1 | X | X | X |
| MDUsel[2:0] | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 3 |
| Start | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| MDU\_RDsel | X | X | X | X | X | X | X | X | X |
| ALU\_MDU\_MUXsel | 0 | 0 | 0 | 0 | 0 | 0 | X | X | X |
| DM\_WE | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| StoreType[1:0] | 2 | X | X | X | X | X | X | X | X |
| LoadType[2:0] | X | 0 | 1 | 2 | 3 | 4 | X | X | X |
| GRF\_WD\_MUXsel[1:0] | X | 1 | 1 | 1 | 1 | 1 | X | X | X |
| GRF\_WE | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

表20 CTRL真值表（6）

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Funct | 011011 | 010011 | 010001 | 010010 | 010000 |  |  |  |  |
| Op | 000000 | 000000 | 000000 | 000000 | 000000 |  |  |  |  |
| rt |  | | | | |  |  |  |  |
|  | divu | mtlo | mthi | mflo | mfhi |  |  |  |  |
| PC\_MUXsel[2:0] | 0 | 0 | 0 | 0 | 0 |  |  |  |  |
| EXTsel[1:0] | X | X | X | X | X |  |  |  |  |
| GRF\_A3\_MUXsel[1:0] | X | X | X | 0 | 0 |  |  |  |  |
| PCCALsel[2:0] | X | X | X | X | X |  |  |  |  |
| ALUsel[1:0] | X | X | X | X | X |  |  |  |  |
| ALU\_A\_MUXsel | X | X | X | X | X |  |  |  |  |
| ALU\_B\_MUXsel | X | X | X | X | X |  |  |  |  |
| MDUsel[2:0] | 4 | 5 | 6 | 0 | 0 |  |  |  |  |
| Start | 1 | 0 | 0 | 0 | 0 |  |  |  |  |
| MDU\_RDsel | X | X | X | 0 | 1 |  |  |  |  |
| ALU\_MDU\_MUXsel | X | X | X | 1 | 1 |  |  |  |  |
| DM\_WE | 0 | 0 | 0 | 0 | 0 |  |  |  |  |
| StoreType[1:0] | X | X | X | X | X |  |  |  |  |
| LoadType[2:0] | X | X | X | X | X |  |  |  |  |
| GRF\_WD\_MUXsel[1:0] | X | X | X | 0 | 0 |  |  |  |  |
| GRF\_WE | 0 | 0 | 0 | 1 | 1 |  |  |  |  |

1. 数据通路（略去控制器及冲突处理单元）

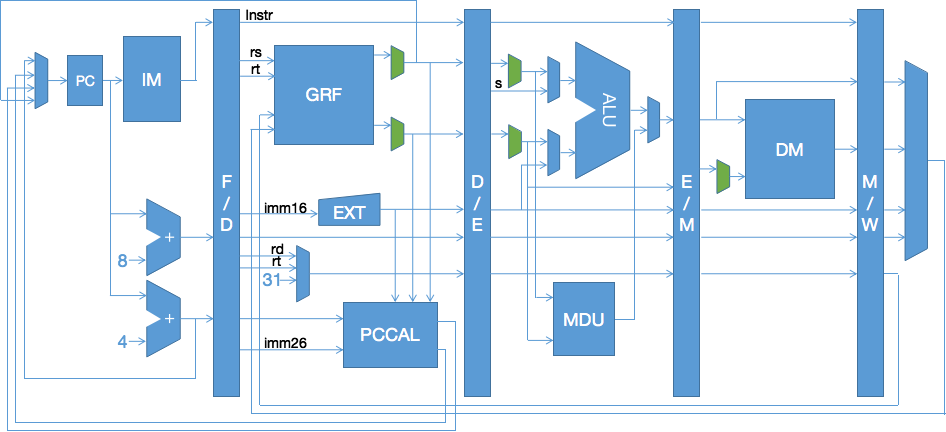


图1 数据通路

1. 测试程序及测试期望

（详见附件）

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？

单独的乘除法部件可以在ALU执行非乘除相关指令的同时执行乘除相关指令，如果整合进ALU，则需要更多的暂停，降低了CPU性能。同理，单独的HI、LO寄存器可以在非乘除相关指令使用通用寄存器时进行读写操作，不会受其影响。

1. 参照你对延迟槽的理解，试解释“乘除槽”。

乘除单元在执行mult、multu、div、divu指令时，如果其后的指令还是乘除相关指令，则需要暂停。而由于乘除单元和ALU可以并行工作，所以可以在mult、multu、div、divu指令之后放入若干无关指令，充分利用乘除单元的延迟，这就是乘除槽。

1. 为何上文文末提到的lb等指令使用的数据扩展模块应在 MEM/WB 之后，而不能在 DM 之后?

从DM中读数据往往是所有流水级中最慢的，而写回阶段是比较快的。如果把数据扩展模块放在DM后会增加关键路径长度，使得整个CPU的性能降低，所以应该放在MEM/WB之后。

1. 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）

C语言中字符串操作经常需要访问字节，此时如果按字访问则性能低。

1. 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？

规划者（Planner）型。所有端口及连线的命名统一规范；指令归类，对一类指令统一进行处理。

1. 你对流水线CPU设计风格有何见解？

不同的设计风格各有各的优缺点。个人认为在大多数情况下，Planner型中新添加的指令都可以被归到已有类别中，只有少数情况下需要进行较多的修改。此外，由于Planner型代码思路清晰，添加新指令其实也没有那么麻烦，且错误率低，不失为一种好的设计。

1. 在本实验中你遇到了哪些不同指令组合产生的冲突？你又是如何解决的？相应的测试样例是什么样的？请有条理的罗列出来。

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 测试类型 | 前序指令 | 测试序列 |
| 1 | R-M-RS | subu | subu $1, $2, $3  addu $4, $1, $2 |
| 2 | R-M-RT | subu | subu $1, $2, $3  addu $4, $2, $1 |
| 3 | R-W-RS | subu | subu $1, $2, $3  nop  addu $4, $1, $2 |
| 4 | R-M-RT | subu | subu $1, $2, $3  nop  addu $4, $2, $1 |
| 5 | I-M-RS | ori | ori $1, $2, 1000  addu $4, $1, $2 |
| 6 | I-M-RT | ori | ori $1, $2, 1000  addu $4, $2, $1 |
| 7 | I-W-RS | ori | ori $1, $2, 1000  nop  addu $4, $1, $2 |
| 8 | I-W-RT | ori | ori $1, $2, 1000  nop  addu $4, $2, $1 |
| 9 | LD-M-RS | lw | addu $1, $2, $3  lw $4, 0($1) |
| 10 | LD-W-RS | lw | addu $1, $2, $3  nop  lw $4, 0($1) |
| 11 | ST-M-RS | sw | addu $1, $2, $3  sw $4, 0($1) |
| 12 | ST-M-RT | sw | addu $1, $2, $3  sw $1, 0($4) |
| 13 | ST-W-RS | sw | addu $1, $2, $3  nop  sw $4, 0($1) |
| 14 | ST-W-RT | sw | addu $1, $2, $3  nop  sw $1, 0($4) |
| 15 | BQ-E-RS | beq | subu $1, $2, $3  beq $1, $4, label |
| 16 | BQ-E-RT | beq | subu $1, $2, $3  beq $4, $1, label |
| 17 | BQ-M-RS | beq | subu $1, $2, $3  nop  beq $1, $4, label |
| 18 | BQ-M-RT | beq | subu $1, $2, $3  nop  beq $4, $1, label |
| 19 | BQ-W-RS | beq | subu $1, $2, $3  nop  nop  beq $1, $4, label |
| 20 | BQ-W-RT | beq | subu $1, $2, $3  nop  nop  beq $4, $1, label |
| 21 | JR-E-RS | jr | subu $1, $2, $3  jr $1 |
| 23 | JR-M-RS | jr | subu $1, $2, $3  nop  jr $1 |
| 25 | JR-W-RS | jr | subu $1, $2, $3  nop  nop  jr $1 |