流水线CPU设计文档(P5)

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功能的选择信号  0：Out = A + B  1：Out = A - B  2：Out = A & B  3：Out = A | B |
| 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. 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 |
| 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 |
| BeqA[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 | 通过比较Cmp1和Cmp2是否相等计算分支地址 |
| 2 | 计算跳转地址 | 通过位拼接操作计算跳转地址 |

8. HZD（冲突处理单元）

端口说明：

表15 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 |
| 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16 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17 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 |
| ALUsel[1:0] | O | ALU功能选择信号（详见ALU端口说明） |
| 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 |
| 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 |
| 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18 CTRL真值表（1）

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Funct | 100001 | 100011 |  | | | | | | |
| Op | 000000 | 000000 | 001101 | 100011 | 101011 | 000100 | 001111 | 000010 | 000011 |
|  | addu | subu | ori | lw | sw | beq | lui | j | jal |
| PC\_MUXsel[2:0] | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 2 | 2 |
| EXTsel[1:0] | X | X | 1 | 0 | 0 | 0 | 2 | X | X |
| GRF\_A3\_MUXsel[1:0] | 0 | 0 | 1 | 1 | X | X | 1 | X | 2 |
| ALUsel[1:0] | 0 | 1 | 3 | 0 | 0 | X | X | X | X |
| ALU\_B\_MUXsel | 0 | 0 | 1 | 1 | 1 | X | X | X | X |
| DM\_WE | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| GRF\_WD\_MUXsel[1:0] | 0 | 0 | 0 | 1 | X | X | 2 | X | 3 |
| GRF\_WE | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |

表18 CTRL真值表（2）

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Funct | 001000 | 000000 |  |  | |  | |  |  |  | |  |
| Op | 000000 | 000000 |  | |  | |  |  |  | |  |  |
|  | jr | nop |  | |  | |  |  |  | |  |  |
| PC\_MUXsel[2:0] | 3 | 0 |  | |  | |  |  |  | |  |  |
| EXTsel[1:0] | X | X |  | |  | |  |  |  | |  |  |
| GRF\_A3\_MUXsel[1:0] | X | X |  | |  | |  |  |  | |  |  |
| ALUsel[1:0] | X | X |  | |  | |  |  |  | |  |  |
| ALU\_B\_MUXsel | X | X |  | |  | |  |  |  | |  |  |
| DM\_WE | 0 | 0 |  | |  | |  |  |  | |  |  |
| GRF\_WD\_MUXsel[1:0] | X | X |  | |  | |  |  |  | |  |  |
| GRF\_WE | 0 | 0 |  | |  | |  |  |  | |  |  |

1. 数据通路（略去控制器及冲突处理单元）

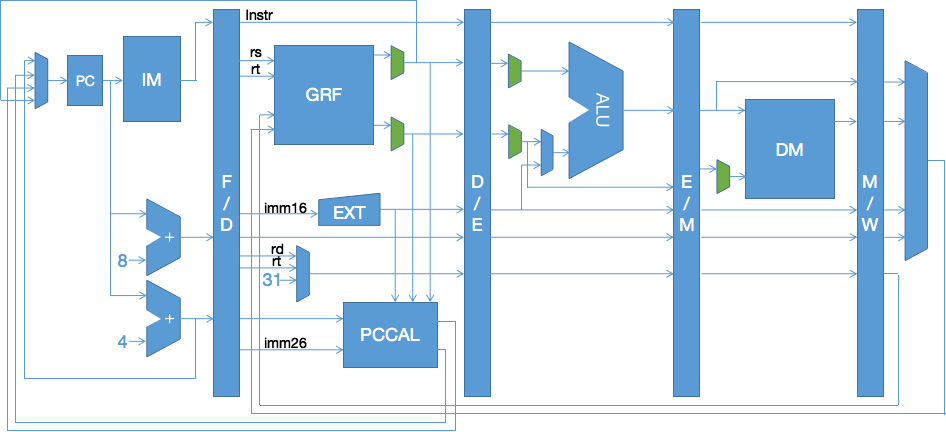


图1 数据通路

1. 测试程序

MIPS汇编代码：

ori $s2, $s2, 5

ori $s3, $s3, 1

ori $s4, $s4, 4

ori $t2, $t2, 0

ori $t3, $t3, 0

loop\_begin:

beq $t2, $s2, loop\_end

nop

sw $t2, 0($t3)

addu $t2, $t2, $s3

addu $t3, $t3, $s4

j loop\_begin

loop\_end:

lui $t4, 41341

lui $t5, 9524

jal label1

nop

beq $s0, $s1, label2

label1:

addu $s0, $t5, $t4

subu $t0, $zero, $t4

subu $t1, $zero, $t5

addu $t0, $t0, $t1

subu $s1, $zero, $t0

jr $ra

nop

label2:

lw $t6, 4($zero)

lw $t7, 8($zero)

nop

nop

lui $t8, 381

lui $t9, 7318

addu $s5, $t8, $t9

测试期望：

@00003000: $18 <= 00000005

@00003004: $19 <= 00000001

@00003008: $20 <= 00000004

@0000300c: $10 <= 00000000

@00003010: $11 <= 00000000

@0000301c: \*00000000 <= 00000000

@00003020: $10 <= 00000001

@00003024: $11 <= 00000004

@0000302c: $12 <= a17d0000

@0000301c: \*00000004 <= 00000001

@00003020: $10 <= 00000002

@00003024: $11 <= 00000008

@0000302c: $12 <= a17d0000

@0000301c: \*00000008 <= 00000002

@00003020: $10 <= 00000003

@00003024: $11 <= 0000000c

@0000302c: $12 <= a17d0000

@0000301c: \*0000000c <= 00000003

@00003020: $10 <= 00000004

@00003024: $11 <= 00000010

@0000302c: $12 <= a17d0000

@0000301c: \*00000010 <= 00000004

@00003020: $10 <= 00000005

@00003024: $11 <= 00000014

@0000302c: $12 <= a17d0000

@0000302c: $12 <= a17d0000

@00003030: $13 <= 25340000

@00003034: $31 <= 0000303c

@00003040: $16 <= c6b10000

@00003044: $ 8 <= 5e830000

@00003048: $ 9 <= dacc0000

@0000304c: $ 8 <= 394f0000

@00003050: $17 <= c6b10000

@00003040: $16 <= c6b10000

@0000305c: $14 <= 00000001

@00003060: $15 <= 00000002

@0000306c: $24 <= 017d0000

@00003070: $25 <= 1c960000

@00003074: $21 <= 1e130000

思考题

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 |