# Project5 Verilog开发流水线CPU

## 一、CPU设计方案综述

### （一）总体设计概述

- 处理器为32位处理器，5级流水线设计
- 支持的指令集位`{addu,subu,ori,lw,sw,beq,liu,j,jal,jr,nop}`
- `addu、subu`可以不支持溢出
- 需要考虑延迟槽
- 支持同步复位

### （二）关键模块定义

#### 1.IFU

- 介绍：介绍：内部包括PC（程序计数器）、IM（指令存储器）及相关逻辑，PC用寄存器实现，具有同步复位功能，复位值为起始地址，IM容量为16KB
- 端口定义：

| 端口 | 方向 | 位宽 | 功能描述 |
| --- | --- | --- | --- |
| clk | I | 1 | 时钟控制信号 |
| reset | I | 1 | 同步复位信号 |
| npc | I | 1 | 下一个PC |
| WE | I | 1 | IFU使能信号 |
| PC | O | 32 | 当前指令地址 |
| Instr | O | 32 | 当前指令 |

- 功能定义

| 序号 | 功能名称 | 功能描述 |
| --- | --- | --- |
| 1 | 同步复位 | 在时钟上升沿到来时，如果`reset`有效，则进行复位，复位PC位0x0000_3000 |
| 2 | 取指令 | PC从IFU取出相应的指令 |
| 3 | 更新PC值 | 在时钟上升沿到来时，如果WE有效，则将PC更新为npc |
| 4 | 暂停 | 当WE=0时，代表暂停操作 |

## 2.GRF

- 介绍：用具有写使能的寄存器实现，寄存器总数为32个，具有同步复位功能，其中0号寄存器始终为0
- 端口定义：

| 端口 | 方向 | 位宽 | 功能描述 |
| --- | --- | --- | --- |
| clk | I | 1 | 时钟信号 |
| reset | I | 1 | 复位信号 |
| A1 | I | 5 | 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 | I | 5 | 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 | I | 5 | 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 |
| WE | I | 1 | 写使能信号 |
| WD | I | 32 | 32位写入数据 |
| PC | I | 32 | 当前指令地址 |
| RD1 | O | 32 | A1所对应的寄存器的值 |
| RD2 | O | 32 | A2所对应的寄存器的值 |

- 功能定义：

| 序号 | 功能名称 | 功能定义 |
| --- | --- | --- |
| 1 | 同步复位 | 在时钟上升沿到来时，如果reset有效时，则进行复位，将所有寄存器中存储的数值清零 |
| 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 |

## 3.DM

- 介绍：容量为4KB，具有同步复位功能
- 端口定义：

| 端口 | 方向 | 位宽 | 功能描述 |
| --- | --- | --- | --- |
| clk | I | 1 | 时钟信号 |
| reset | I | 1 | 同步复位信号 |
| WE | I | 1 | 写使能信号 |
| Addr | I | 32 | 写入地址 |
| WD | I | 32 | 写入的数据 |
| PC | I | 32 | 当前指令地址 |
| RD | O | 32 | 读出的数据 |

- 功能定义：

| 序号 | 功能名称 | 功能描述 |
| --- | --- | --- |
| 1 | 复位 | 在时钟上升沿到来时，若reset有效，则进行复位 |
| 2 | 写入数据 | 当WE = 1时，将WD写入Addr地址中 |
| 3 | 读数据 | RD始终为Addr地址的值 |

### 4.ALU

- 介绍：提供32位加、减、或运算以及大小比较功能，不检测溢出。
- 端口定义：

| 端口 | 方向 | 位宽 | 功能描述 |
| --- | --- | --- | --- |
| SrcA | I | 32 | 操作数1 |
| SrcB | I | 32 | 操作数2 |
| ALUOP | I | 3 | ALU运算控制信号 |
| Result | O | 32 | 运算结果 |

- 功能定义：

| 序号 | 功能名称 | 功能描述 |
| --- | --- | --- |
| 1 | 或运算 | 当ALUOP = 1，进行或运算 |
| 2 | 加法运算 | 当ALUOP = 2，进行加法运算 |
| 3 | 减法运算 | 当ALUOP = 3，进行减法运算 |
| 4 | 置高位操作 | 当ALUOP = 4，将B右移16位 |

## 5.Controller

- 介绍：产生控制信号
- 端口定义：

| 端口 | 方向 | 位宽 | 功能描述 |
| --- | --- | --- | --- |
| I | I | 32 | 当前指令 |
| addu | O | 1 | addu信号 |
| subu | O | 1 | subu信号 |
| ori | O | 1 | ori信号 |
| lw | O | 1 | lw信号 |
| sw | O | 1 | sw信号 |
| beq | O | 1 | beq信号 |
| lui | O | 1 | lui信号 |
| jal | O | 1 | jal信号 |
| jr | O | 1 | jr信号 |
| j | O | 1 | j信号 |
| rs | O | 5 | 当前指令的rs |
| rt | O | 5 | 当前指令的rt |
| rd | O | 5 | 当前指令的rd，当I=jal时，rd=31 |
| R | O | 1 | 除jalr、jr外的R型运算指令 |
| load | O | 1 | 需要从内存中取值的信号 |
| store | O | 1 | 需要存入内存的信号 |
| type_i | O | 1 | I型运算指令 |
| j_l | O | 1 | 跳转并链接信号的值 |
| j_ra | O | 1 | 需要用到rs寄存器的跳转指令 |
| j_2reg | O | 1 | 需要用到双寄存器的跳转指令 |

- 功能定义：

| 序号 | 功能名称 | 功能定义 |
| --- | --- | --- |
| 1 | 产生控制信号 | 产生控制信号 |

## 6.Ext

- 介绍：支持16bit到32bit的零拓展以及符号拓展
- 端口定义：

| 端口 | 方向 | 位宽 | 功能描述 |
| --- | --- | --- | --- |
| imm_16 | I | 16 | 16位待拓展的立即数 |
| ExtOp | I | 1 | 拓展控制信号 |
| Result | O | 32 | 拓展后的结果 |

- 功能定义：

| 序号 | 功能名称 | 功能描述 |
| --- | --- | --- |
| 1 | 零拓展 | 当ExtOp = 0时，进行零拓展 |
| 2 | 符号拓展 | 当ExtOp = 1时，进行符号拓展 |

## 7.IF_ID

- 介绍；I级和D级间的流水线寄存器，同时产生D级的控制信号
- 端口定义：

| 端口 | 方向 | 位宽 | 功能描述 |
| --- | --- | --- | --- |
| reset | I | 1 | 复位信号 |
| clk | I | 1 | 时钟控制信号 |
| F_I | I | 32 | IFU指令 |
| F_PC | I | 32 | IFU的PC |
| D_I | O | 32 | D级指令 |
| D_PC | O | 32 | D级PC |
| D_Branch | O | 4 | 跳转信号 |

- 功能定义：

| 序号 | 功能名称 | 功能描述 |
| --- | --- | --- |
| 1 | 流水F_I | 在时钟上升沿到来时，将F_I赋给D_I |
| 2 | 流水F_PC | 在时钟上升沿到来时，将F_PC赋给D_PC |
| 3 | 生成D级控制信号 | 生成D级控制信号 |

## 8.ID_EX

- 介绍：D级和E级间的流水线寄存器，同时产生E级的控制信号
- 端口定义：

| 端口 | 方向 | 位宽 | 功能描述 |
| --- | --- | --- | --- |
| clk | I | 1 | 时钟控制信号 |
| reset | I | 1 | 复位信号 |
| D_I | I | 32 | D级指令 |
| D_PC | I | 32 | D级PC |
| D_RD1 | I | 32 | D级RD1 |
| D_RD2 | I | 32 | D级RD2 |
| E_I | O | 32 | E级指令 |
| E_PC | O | 32 | E级PC |
| E_RD1 | O | 32 | E级RD1 |
| E_RD2 | O | 32 | E级RD2 |
| ExtOp | O | 1 | 拓展信号 |
| E_s | O | 1 | SrcB选择信号 |
| E_ALUOP | O | 1 | ALU控制信号 |

- 功能定义：

| 序号 | 功能名称 | 功能描述 |
| --- | --- | --- |
| 1 | 流水D_I | 在时钟上升沿到来时，将D_I赋给E_I |
| 2 | 流水D_PC | 在时钟上升沿到来时，将D_PC赋给E_PC |
| 3 | 流水D_RD1 | 在时钟上升沿到来时，将D_RD1赋给E_RD1 |
| 4 | 流水D_RD2 | 在时钟上升沿到来时，将D_RD2赋给E_RD2 |
| 5 | 产生E级控制信号 | 产生E级ExtOp、E_s、E_ALUOP信号 |

## 9.EX_MEM

- 介绍：E级和M级间的流水线寄存器，同时产生M级的控制信号
- 端口定义：

| 端口 | 方向 | 位宽 | 功能描述 |
| --- | --- | --- | --- |
| clk | I | 1 | 时钟控制信号 |
| reset | I | 1 | 复位信号 |
| E_I | I | 32 | E级指令 |
| E_PC | I | 32 | E级PC |
| E_AO | I | 32 | E级AO |
| E_RD2 | I | 32 | E级RD2 |
| M_WE | O | 1 | M级写入信号 |
| M_I | O | 32 | M级指令 |
| M_PC | O | 32 | M级PC |
| M_AO | O | 32 | M级AO |
| M_RD2 | O | 32 | M级RD2 |

- 功能定义：

| 序号 | 功能名称 | 功能描述 |
| --- | --- | --- |
| 1 | 流水E_I | 在时钟上升沿到来时，将E_I赋给M_I |
| 2 | 流水E_PC | 在时钟上升沿到来时，将E_PC赋给M_PC |
| 3 | 流水E_AO | 在时钟上升沿到来时，将E_AO赋给M_AO |
| 4 | 流水E_RD2 | 在时钟上升沿到来时，将E_RD2赋给M_RD2 |
| 5 | 产生M级控制信号 | 产生M级控制信号 |

### 10.MEM_WB

- 介绍：M级和W级间的流水线寄存器，同时产生W级的控制信号
- 端口定义：

| 端口 | 方向 | 位宽 | 功能描述 |
| --- | --- | --- | --- |
| reset | I | 1 | 复位信号 |
| clk | I | 1 | 时钟控制信号 |
| M_I | I | 32 | M级指令 |
| M_PC | I | 32 | M级PC |
| M_RD | I | 32 | M级RD |
| M_AO | I | 32 | M级AO |
| WE | O | 1 | GRF写入信号 |
| ra | O | 1 | ra写入信号 |
| RegDst | O | 1 | 写入rd信号 |
| GRF_PC | O | 32 | GRF的PC |
| GRF_I | O | 32 | GRF的I |
| W_RD | O | 32 | W级RD |
| W_AO | O | 32 | W级AO |
| WD_S | O | 3 | GRF写入数据选择信号 |

- 功能定义：

| 序号 | 功能名称 | 功能描述 |
| --- | --- | --- |
| 1 | 流水M_I | 在时钟上升沿到来时，将M_I赋给GRF_I |
| 2 | 流水M_PC | 在时钟上升沿到来时，将M_PC赋给GRF_PC |
| 3 | 流水M_RD | 在时钟上升沿到来时，将M_RD赋给W_RD |
| 4 | 流水M_AO | 在时钟上升沿到来时，将M_AO赋给W_AO |
| 5 | 产生W级控制信号 | 产生W级控制信号 |

### 11.SFU

- 介绍：暂停 (Stall) 和 转发 (Forward) 的控制单元，产生两者的控制信号。
- 端口定义：

| 端口 | 方向 | 位宽 | 功能描述 |
| --- | --- | --- | --- |
| d_I | I | 32 | D级指令 |
| ex_I | I | 32 | E级指令 |
| mem_I | I | 32 | M级指令 |
| wb_I | I | 32 | W级指令 |
| Stall | O | 1 | 暂停信号 |
| RD1_s | O | 4 | RD1选择信号<br>0：GRF[rs]<br>1：E级的PC + 8<br>2：M级的PC + 8<br>3：M级的AO |
| RD2_s | O | 4 | RD2选择信号<br>0：GRF[rs]<br>1：E级的PC + 8<br>2：M级的PC + 8<br>3：M级的AO |
| SrcA_s | O | 4 | SrcA选择信号<br>0：ID_EX寄存器的RD1<br>2：M级的PC + 8<br>3：M级的AO<br>4：W级的WD |
| SrcB_s | O | 4 | SrcB选择信号<br>0：ID_EX寄存器的RD2<br>2：M级的PC + 8<br>3：M级的AO<br>4：W级的WD |
| M_Data_s | O | 4 | M_Data选择信号<br>0：EX_MEM的AO<br>4：W级的WD |

- 功能定义：

| 序号 | 功能名称 | 功能描述 |
| --- | --- | --- |
| 1 | D级RD1、RD2的选择信号 | 当D级比较器的两个输入与之前指令产生数据冲突时，进行转发 |
| 2 | E级SrcA、SrcB的选择信号 | 当E级ALU的两个来源于之前指令产生数据冲突时，进行转发 |
| 3 | M级Data的选择信号 | 当M级Data于之前指令产生数据冲突时，进行转发 |
| 4 | 暂停 | 当发生控制冲突时，IFU和IF_ID将暂停，同时清空ID_EX |

## （三）重要机制实现方法

### 利用Tuse-Tnew法解决数据冒险

- 先分析每个指令的Tuse和Tnew，然后根据Tuse和Tnew产生暂停以及转发信号
- 如果Tuse<Tnew，则优先考虑转发，其次考虑暂停
- 暂停
  - `beq`需要用到`rs,rt`寄存器，所以对于任何**需要写入寄存器**的指令，都需要考虑暂停和转发，其中如果和`lw`发生数据冲突，需要暂停两个周期
  - `jr`需要用到`rs`寄存器，所以对于任何**需要写入寄存器**的指令，都要考虑暂停和转发，与`beq`的情况类似
  - 除跳转指令外**任何需要用到寄存器进行计算**的指令与`lw`指令之如果发生了数据冲突，需要暂停
- 转发
  - D级与W级的转发已通过GRF内部转发实现，故无需考虑
  - D级比较器的两个输入的转发来源
    - E级的PC + 8
    - M级的PC + 8
    - M级的AO
  - SrcA、SrcB的转发来源
    - M级的PC + 8
    - M级的AO
    - W级的WD
  - M_Data的转发来源
    - W级的WD

### 测试样例

```
ori $1, $0, 1
ori $2, $1, 2
ori $3, $2, 3
ori $4, $3, 4
ori $5, $3, 5
ori $6, $3, 6
ori $7, $5, 7
ori $8, $6, 8
ori $9, $7, 9
ori $10, $9, 10
ori $11, $10, 11
ori $12, $10, 12
ori $13, $12, 13
ori $14, $13, 14
ori $15, $13, 15
ori $16, $14, 16
ori $17, $15, 17
ori $18, $17, 18
ori $19, $18, 19
ori $20, $19, 20
ori $21, $20, 21
ori $22, $15, 22
```

```
ori $23, $22, 23
ori $24, $2, 24
ori $25, $6, 25
ori $26, $23, 26
ori $27, $25, 27
ori $28, $27, 28
ori $29, $28, 29
ori $30, $20, 30
ori $31, $30, 31
addu $1, $2, $3
subu $9 $1, $2
addu $10, $9, $1
subu $6, $9, $10
sw $6, 20($0)
ori $8, $0, 20
lw $8, 20($0)
lw $10,0($8)
sw $1, 0($0)
sw $2, 4($0)
sw $3, 12($0)
sw $4, 16($0)
ori $1, $0, 200
sw $1, 200($0)
ori $10, $0, 200
lw $10, 0($0)
lw $20, 0($1)
jal loop
nop
ori $5,$0, 200
sw $7, 400($0)
j end
sw $8, 200($0)
loop:
jalr $5, $31
lw $7, 0($0)
end:
ori $5, $5, 200
sw  $31, 200($0)
ori $8, $0, 200
lw $7,0($8)
beq $8, $7, final
nop
ori $25, $8, 200
final:
sw $8, 0($3)
```

# 二、思考题

## 流水线冒险

1. 在采用本节所述的控制冒险处理方式下，PC 的值应当如何被更新？请从数据通路和控制信号两方面进行说明。
    - 将IFU的npc输入端和D级的一个MUX输出端相连，该MUX的输出为正常执行命令的下一个PC地址以及各种跳转指令对应的下一个PC地址

- IF_ID流水线寄存器直接译码产生D_Branch信号，配合前移至D级的比较器一起产生控制信号，用来控制产生npc的地址

2. 对于 jal 等需要将指令地址写入寄存器的指令，为什么需要回写 PC+8？
   - PC + 4 的地址是延迟槽，延迟槽中的指令一定执行，所以跳回时应该跳到延迟槽下一条指令，即PC + 8

## 数据冒险的分析

1. 为什么所有的供给者都是存储了上一级传来的各种数据的**流水级寄存器**，而不是由 ALU 或者 DM 等部件来提供数据?
   - 如果从非流水线寄存器部件转发，那么某一级的总延迟就会增加，从而根据木桶效应，时钟周期就会增加，总效率就会增加，而转发的目的是为了降低延迟，所以不能由ALU、DM转发

## AT法处理流水线数据冒险

1. “转发（旁路）机制的构造”中的 Thinking 1-4；
   - **Thinking 1**：如果不采用已经转发过的数据，而采用上一级中的原始数据，会出现怎样的问题？试列举指令序列说明这个问题。
     - 运算过程中会用到上一级还为保存的数值，从而出错，例如：

       ```
       ori  $1, $0, 100
       addu $2, $1, $1
       sw   $2, 0($0)
       ```

       这样`addu`就会把`0`存进`$2`，接着`sw`就会把`0`存进DM中。
   - **Thinking 2**：我们为什么要对 GPR 采用内部转发机制？如果不采用内部转发机制，我们要怎样才能解决这种情况下的转发需求呢?
     - GPR采用内部转发机制相当于MEM_WB流水线寄存器的值可以实时反馈到GPR的输出端，这样的话，就不用再考虑W级和D级之间的转发。
     - 如果不采用内部转发机制，需要考虑从W级到D级的数据通路。
   - **Thinking 3**：为什么 0 号寄存器需要特殊处理?
     - 因为指令可以对0号寄存器赋值，只是不会造成实际作用，但是转发过程中如果不特判就默认0号寄存器的值被修改了，从而造成错误。
   - **Thinking 4**：什么是“最新产生的数据”?
     - 按照指令的执行顺序，越后执行的指令更改的寄存器的值越新，按照D、E、M、W的顺序，越靠前所转发的信息越新，因此优先级更高
2. 在 AT 方法讨论转发条件的时候，只提到了“供给者需求者的A相同，且不为 0”，但在 CPU 写入 GRF 的时候，是有一个 we 信号来控制是否要写入的。为何在 AT 方法中不需要特判 we 呢？为了**用且仅用** A 和 T 完成转发，在翻译出 A 的时候，要结合 we 做什么操作呢?
   - AT法是说：只要当前位点的读取寄存器地址和某转发输入来源的写入寄存器地址相等且不为0。因为已经说是要写入的，WE必然为1，不需特判。
   - 不用做任何操作。

## 在线测试相关说明

- 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?

如果你是手动构造的样例，请说明构造策略，说明你的测试程序如何保证**覆盖**了所有需要测试的情况；如果你是**完全随机**生成的测试样例，请思考完全随机的测试程序有何不足之处；如果你在生成测试样例时采用了**特殊的策略，**比如构造连续数据冒险序列，请你描述一下你使用的策略如何**结合了随机性**达到强测的效果。

此思考题请同学们结合自己测试 CPU 使用的具体手段，按照自己的实际情况进行回答。

- 冲突方面的话，结构冲突因为采用了DM和IM分离分策略所以不存在，那么主要就是看控制冒险和数据冒险，分别通过比较前移+延迟槽和暂停转发解决。
- 构造策略：使用`ori`指令将每个寄存器赋值，然后将这些都存入DM，最后构造数据相关的运算指令，中间插入跳转指令，根据这样的规则生成了大量的测试样例，针对每种转发、暂停情况，都进行了相应的测试。