# Verilog流水线处理器

20373864 谭立德

## 一、CPU设计方案综述

### （一）总体设计概述

本CPU为verilog实现的流水线MIPS - CPU，支持的指令集包含{ addu, subu, ori, lw, sw, beq, lui, j, jal, jr, nop }。为了实现这些功能，CPU主要包含了IM、GRF、DM、ALU、PC、CU ，这些模块按照自顶向下的顶层设计逐级展开。

### （二）关键模块定义

#### 1. GRF （通用寄存器组，也称为寄存器文件、寄存器堆）

GRF端口定义：

表0 GRF端口表

|  |  |  |
| --- | --- | --- |
| 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：不可向GEF中写入数据 |
| A1 | 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至RD1 |
| A2 | 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至RD2 |
| A3 | 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 |
| WD3 | I | 32位数据输入信号 |
| RD1 | O | 输出指定的寄存器中的32位数据 |
| RD2 | O | 输出指定的寄存器中的32位数据 |

GRF模块功能定义：

表1 GRF功能表

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 复位 | Reset信号有效时，所有寄存器储存的数值清零 |
| 2 | 读数据 | 读出A1，A2地址对应寄存器中所储存的数据到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 |

1. DM （数据存储器）：

DM端口定义：

表2 DM端口表

|  |  |  |
| --- | --- | --- |
| 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DM中写入数据  0：不可向DM中写入数据 |
| A | I | 5位地址输入信号，指定中储存器上的地址，将其中存储的数据读出至RD1 |
| WD | I | 32位数据输入信号 |
| RD | O | 输出储存器指定地址上的32位数据 |

DM模块功能定义：

表3 DM功能表

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 复位 | Reset信号有效时，储存器储存的所有数值清零 |
| 2 | 读数据 | 读出A地址对应储存器中所储存的数据到RD |
| 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 |

#### ALU （算术逻辑运算单元）：

ALU端口定义：

表4 ALU端口表

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| SrcA | I | 32位运算数输入信号 |
| SrcB | I | 32位运算数输入信号 |
| ALU Control | I | 3位逻辑运算选择信号，选择进行哪种逻辑运算 |
| Zero | O | 输出比较两运算数比较的1位输出 |
| ALU Result | O | 输出对两运算数进行指定逻辑运算后的32位结果 |

ALU模块功能定义：

表5 ALU功能表

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 计算 | 根据控制信号进行对应的逻辑计算并输出 |
| 2 | 比较 | 判断两个输入是否相等 |

#### IM （指令存储器）：

IM端口定义：

表6 IM端口表

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| PC | I | 5位输入地址信号 |
| Instr | 0 | 输出地址所储存32位指令 |

IM模块功能定义：

表7 IM功能表

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 描述 |
| 1 | 读指令 | 根据输入输出对应32位指令 |

#### Control Unit （指令译码器）：

Control Unit端口定义：

表8 Control Unit端口表

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Opcode[5:0] | I | 指令操作码 |
| Funct[5:0] | I | 指令功能码 |
| Jump | O | 跳转信号 |
| ToHigh16 | O | 高位置位信号 |
| ExtOp | O | 位扩展方式 |
| MemtoReg | O | 读内存信号 |
| MemWrite | O | 内存写使能信号 |
| Branch | O | 分支信号 |
| ALUCtrl[2:0] | O | ALU控制信号 |
| ALUSrc | O | ALU操作数2的来源0：寄存器1：立即数 |
| RegDst | O | 寄存器写地址选择0：Instr[20:16] 1：Instr[15:11] |
| RegWrite | O | 寄存器写使能信号 |
| DMop[1:0] | O | 存储、读取方式控制信号 |

### （三）重要机制实现方法

#### 1. J类型指令

根据输入判断和ALU模块协同工作算出跳转地址后跳转。

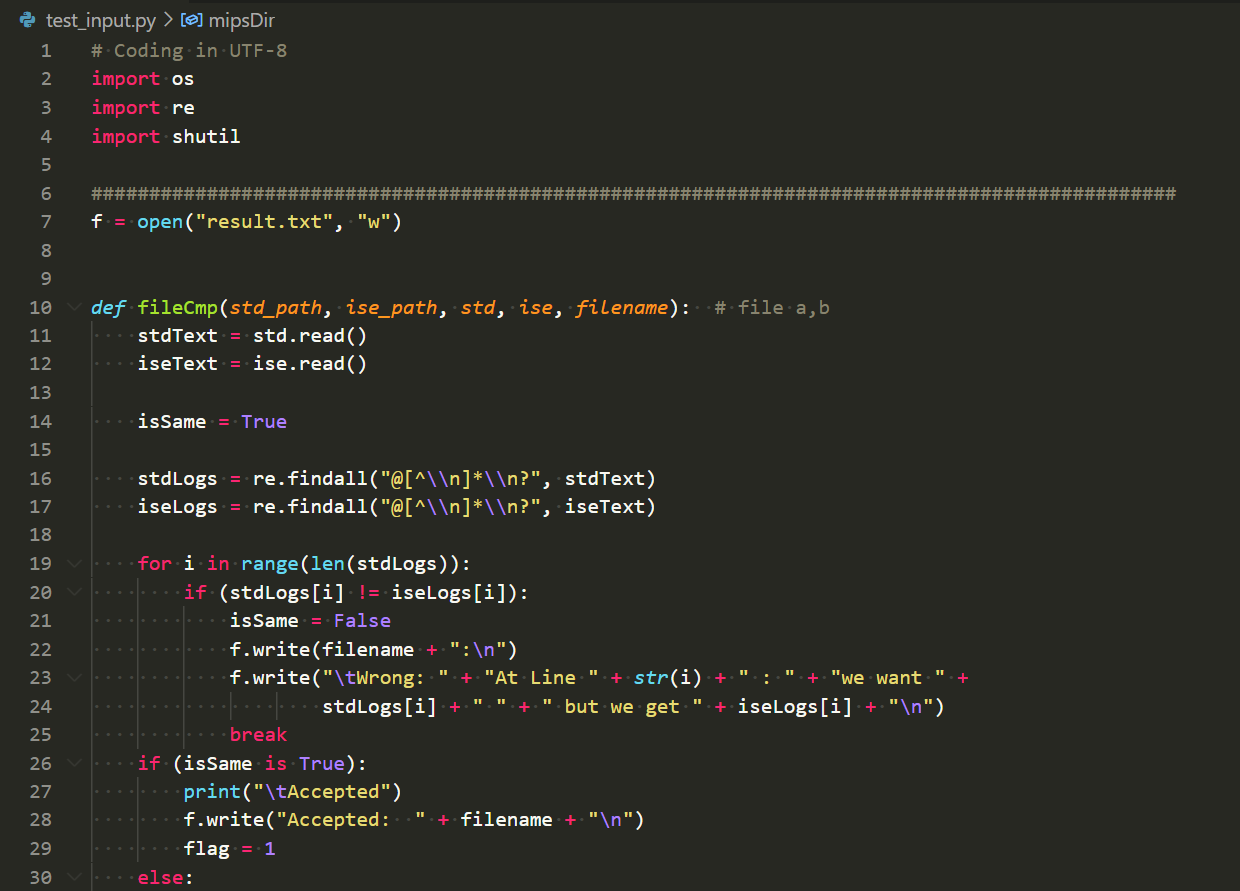
#### 2. R类型指令

根据输入判断和ALU模块协同工作算出结果后存储回寄存器堆中以实现指令R类型指令。

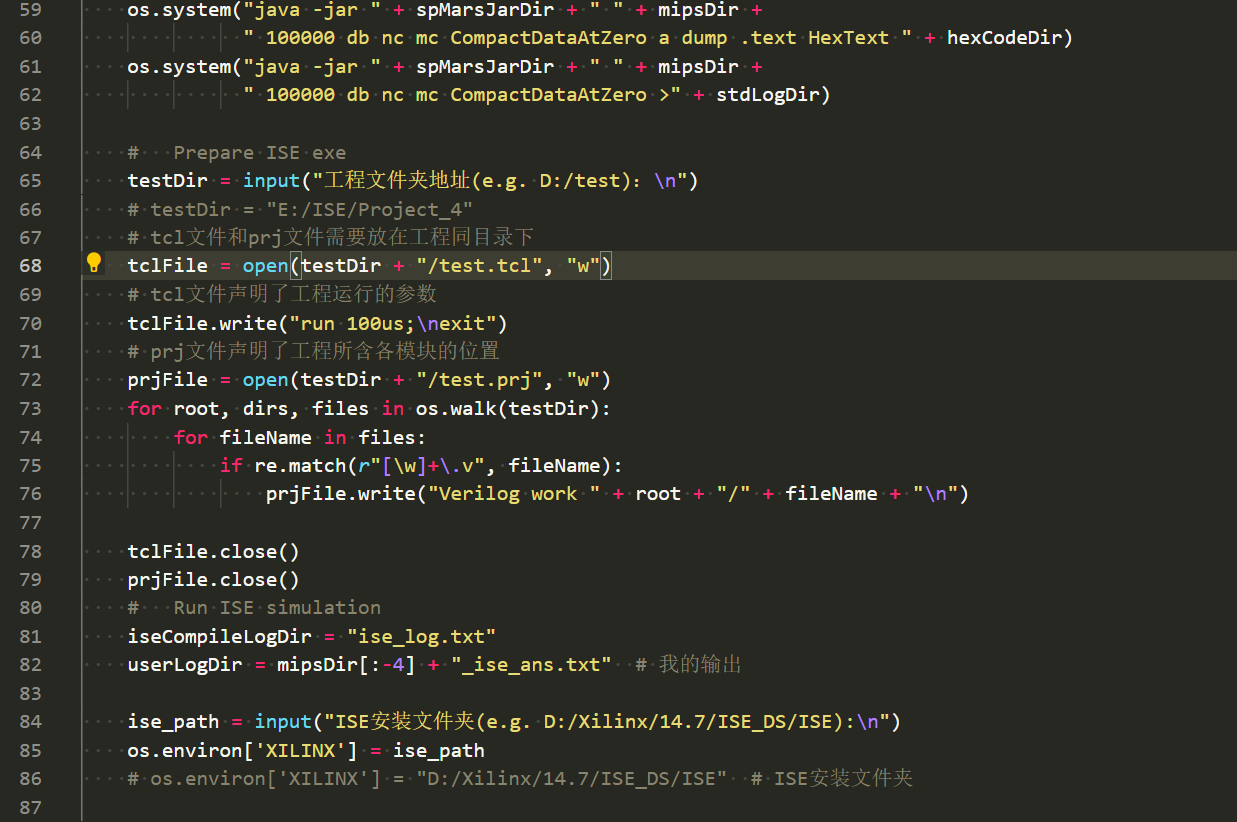
#### 3. I类型指令

根据输入判断和ALU模块和DM模块协同工作支持I类型指令。

## 测试方案









## 三、思考题

### （一）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。

解：

默认更新为PC+4，在跳转指令时更新为NPC的输出，在jr指令时更新为对应寄存器的转发后的值。

### 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 ？

解：

本流水线CPU考虑延迟槽，无论是否跳转跳转指令的下一条指令都应该被执行，故回写指令应为下下条指令的地址，即为PC+8。

### （二）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？

解：

ALU或者 DM 等部件来提供数据会导致当前所需时间周期的增长，从而整个流水线的时钟周期会随之增长，大大降低流水线的流水效率，得不偿失。

### （三）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；（什么是“最新产生的数据？）

解：

最新产生的数据即指在流水线通路中最靠近D级寄存器的寄存器中所存储的数据。

### 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？

解：

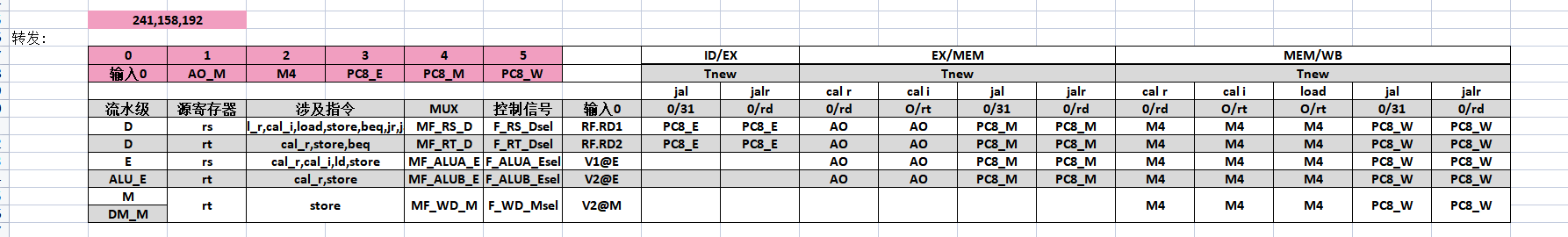
A\_T法在获得A和T后暂停和转发只由A和T控制，与是否使能无关。翻译A时，需要把we加入判断条件，若we为零，则不能产生对应信号。

### （四）在线测试相关说明

### 1.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。）

解：

不同类型组合产生的冲突：





解决方法：构造转发和暂停机制，在遇到冲突冒险时进行对应转发和冒险操作。

尚未完成数据构造。

附1：

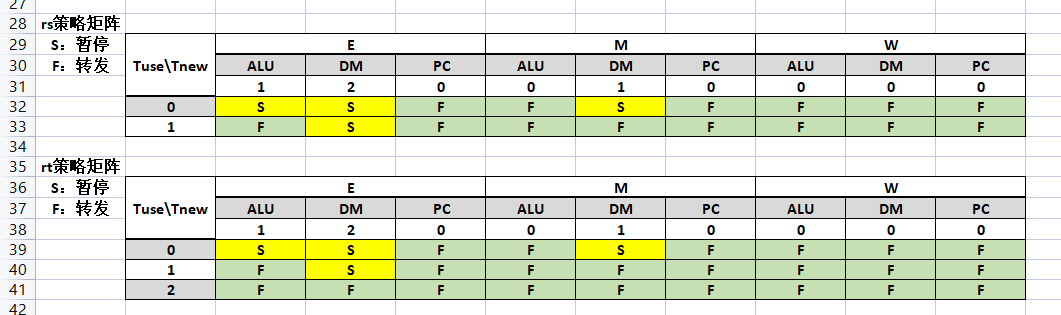
数据通路

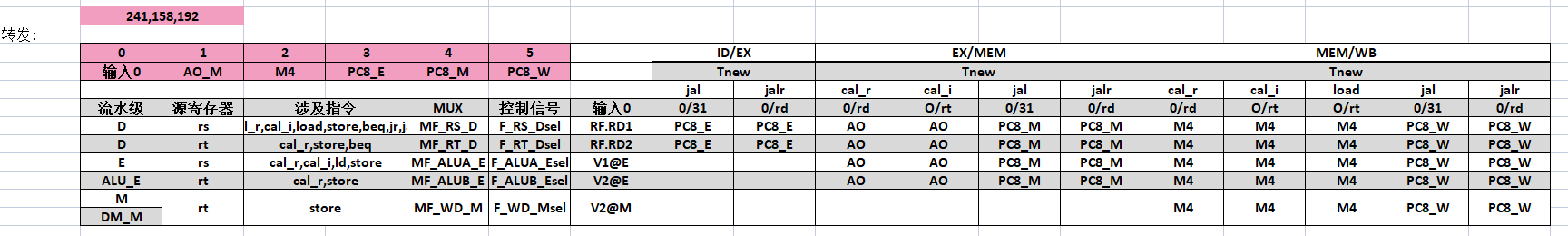


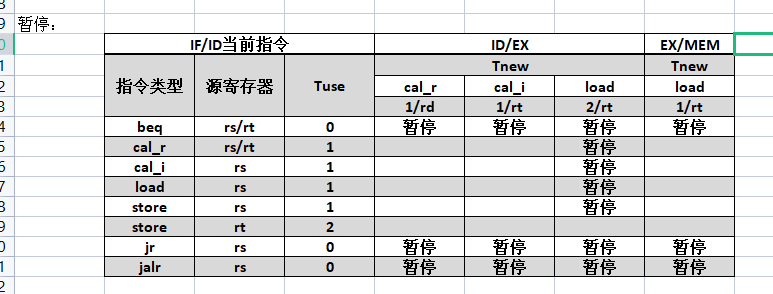
附2：

冲突策略矩阵



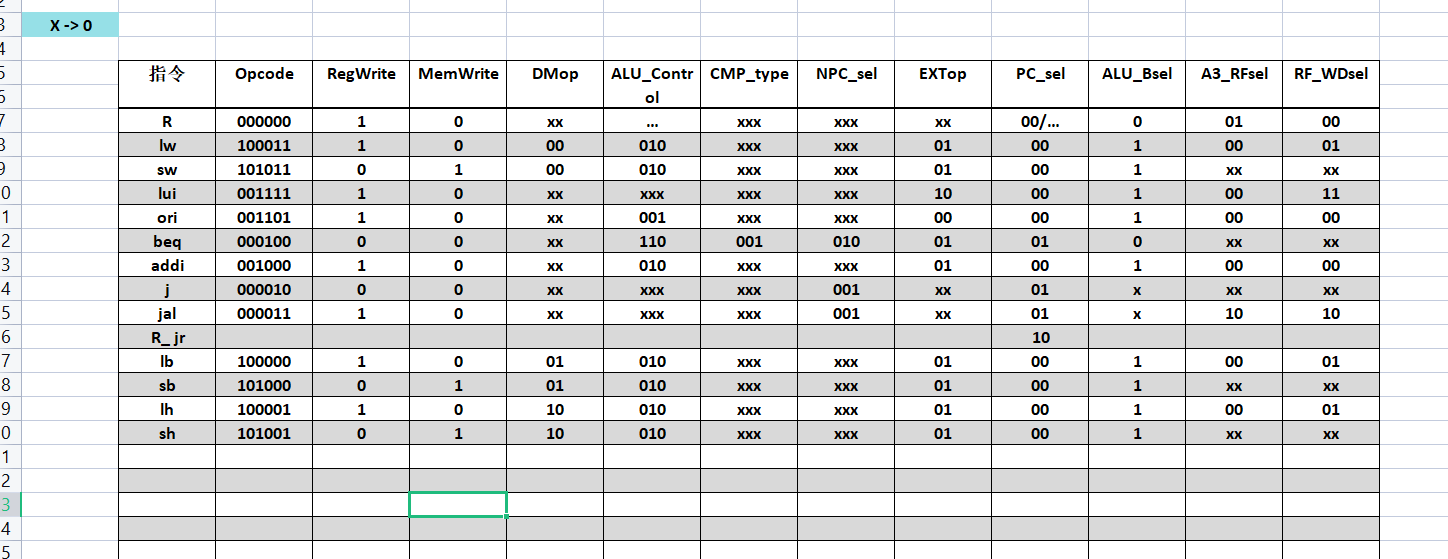


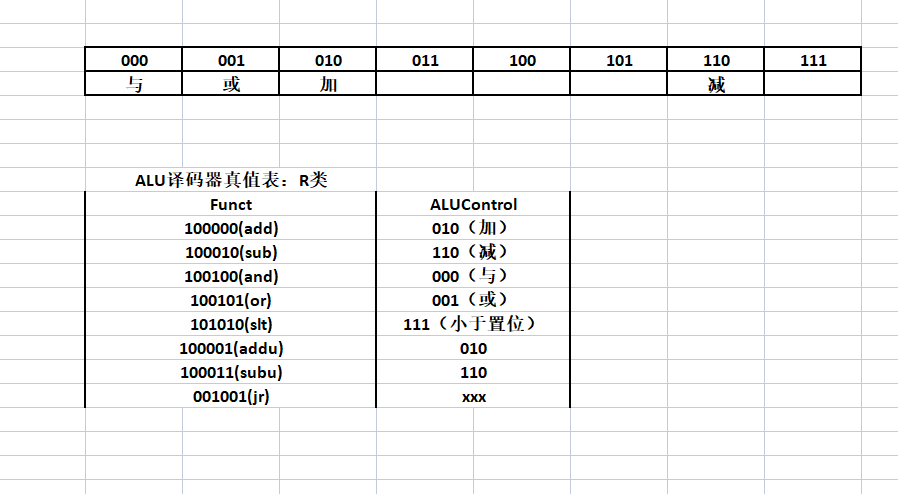




附3：

控制器





附4：

多路选择器控制信号

