**实验5报告**

2017K8009908018

蔡润泽

箱子号：45

一、实验任务（10%）

本次实验任务是了解前递式五级流水线CPU的设计思路，并在实验4 CPU 代码的基础上，加入数据前递通路来减少阻塞。通过运行 func\_lab4，成功通过仿真和上板验证，并且仿真运行时间较 lab4 的结果有下降。

二、实验设计（40%）

（一）总体设计思路

采用前递方式（指令RAM和数据RAM为同步RAM）的硬件设计图如下：

**图片包含 地图

描述已自动生成**

图1.1 硬件结构设计图

如图1.2的流水示意图，在代码设计中，主要有7个模块，包括五级流水、ALU以及寄存器堆。该设计使用了两个IP，Inst\_RAM和Data\_RAM（采用同步RAM）。并且采用Tools模块进行译码。

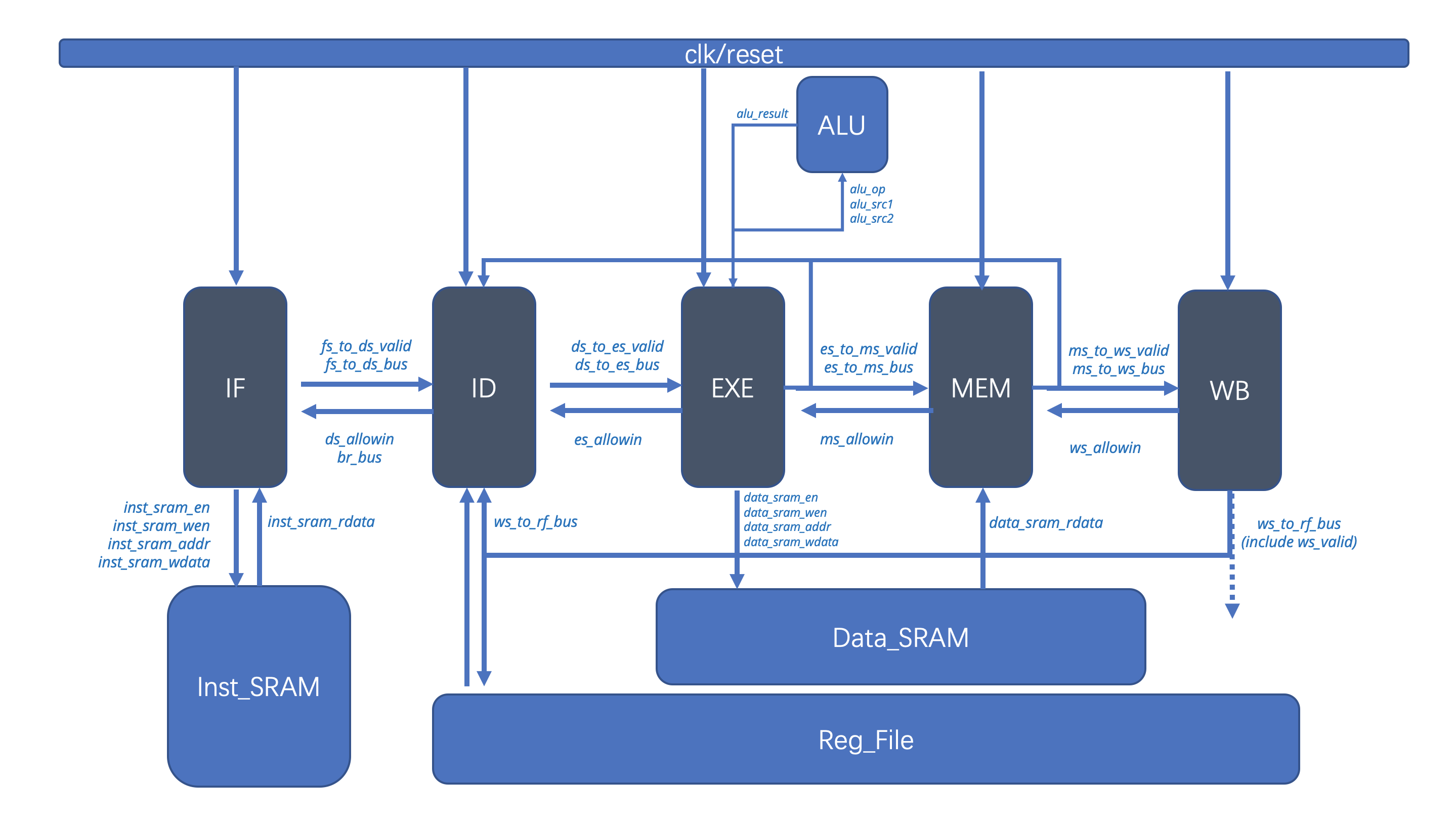


图1.2 流水示意图

（二）重要模块1设计：算数逻辑单元（ALU）模块

1. 工作原理

将CPU中的运算处理进行模块化，方便外界调用。同时模块化的ALU设计便于在其中增加新的运算功能，提高代码的扩展性。

1. 接口定义

input [11:0] alu\_op, //输入运算符

input [31:0] alu\_src1, //输入数据1

input [31:0] alu\_src2, //输入数据2

output [31:0] alu\_result //输出结果

1. 功能描述

采用12位的独热码对ALU进行控制，根据独热码进行12项不同的算数逻辑运算操作，并将结果传回给exe阶段。

（三）重要模块2设计：寄存器堆（Reg\_File）模块

1. 工作原理

将32个32位宽的寄存器堆模块化，以实现两读一写，同步读异步写的操作。

1. 接口定义

input clk,

// READ PORT 1

input [ 4:0] raddr1,

output [31:0] rdata1,

// READ PORT 2

input [ 4:0] raddr2,

output [31:0] rdata2,

// WRITE PORT

input we, //write enable, HIGH valid

input [ 4:0] waddr,

input [31:0] wdata

1. 功能描述

当写使能信号为1时，在写回阶段对寄存器堆进行写入。同时，对于两个读端口信号，进行异步读取，将输出结果传递给ID阶段。

（四）重要模块3设计：取指阶段（IF\_stage）模块

1. 工作原理

将取指操作模块化，IF从inst\_ram中读出指令，并且在该周期模块之内处理PC的值，并且将指令传递给bus传递给ID模块，ID模块在下一周期再接收 。

1. 接口定义

表2.1 IF\_stage接口定义

| **名称** | **方向** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| clk | IN | 1 | 时钟信号 |
| reset | IN | 1 | 复位信号 |
| ds\_allowin | IN | 1 | ID模块允许接受IF传值 |
| br\_bus | IN | 33 | 输入是否跳转和branch的target |
| fs\_to\_ds\_valid | OUT | 1 | IF模块可以向ID模块传值 |
| fs\_to\_ds\_bus | OUT | 64 | IF模块向ID模块传递数据（指令码和地址） |
| inst\_sram\_en | OUT | 1 | Inst\_sram读使能 |
| inst\_sram\_wen | OUT | 4 | Inst\_sram写使能，此处恒为0 |
| inst\_sram\_addr | OUT | 32 | Inst\_sram目标地址 |
| inst\_sram\_wdata | OUT | 32 | Inst\_sram写数据 |
| inst\_sram\_rdata | IN | 32 | Inst\_sram读数据 |

1. 功能描述

PC在收到reset信号时设为偏移量32'hbfbffffc，并且在该周期模块之内处理PC的值，PC值的变化根据br\_bus取出来决定是否跳转还是加4。IF模块当inst\_sram\_en读使能信号为1时，将处理后的next\_pc作为地址传递给inst\_sram, 并从inst\_ram中读出指令。IF模块将取出的指令和地址传递给ID模块，ID模块在下一周期再接收 。

（五）重要模块4设计：译码阶段（ID\_stage）模块

1. 工作原理

将从IF模块获取的指令进行译码，获得指令格式类型、ALU操作类型、是否需要加载、写回内存和参与运算数据的值、跳转的目标PC。判断PC是否需要跳转，并将结果返还给IF模块。将译码后的数据和控制信号在传递给EXE模块,EXE模块在下一时钟周期接受。另外，写回阶段的数据也通过该模块传递给寄存器堆。

相较于之前的无阻塞五级流水设计，新的CPU数据通路增加旁路设计，来让前面的指令直接把已经生成出来的结果直接转给后面的指令。在本设计中，采用了“流水级组合逻辑的结果传递到译码级寄存器读出处”的方案。并通过后续阶段的valid信号和gr\_we信号来控制ID模块中，rs\_value和rt\_value的值。

另外对于ready\_go信号，当译码级的指令和处在执行级的LW指令相关时，需要设置成“0”

1. 接口定义

表2.2 ID\_stage接口定义

| **名称** | **方向** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| clk | IN | 1 | 时钟信号 |
| reset | IN | 1 | 复位信号 |
| es\_allowin | IN | 1 | EXE模块允许接受ID传值 |
| ds\_allowin | OUT | 1 | 允许IF模块向ID模块传递数据 |
| fs\_to\_ds\_valid | IN | 1 | IF模块可以向ID模块传值 |
| fs\_to\_ds\_bus | IN | 64 | IF模块向ID模块传递数据（指令码及地址） |
| ds\_to\_es\_valid | OUT | 1 | 允许ID模块向ES模块传递数据 |
| ds\_to\_es\_bus | OUT | 136 | ID模块向EXE模块传递数据 |
| br\_bus | OUT | 33 | 输出是否跳转和branch的target给IF模块 |
| ws\_to\_rf\_bus | IN | 38 | WB模块向ID模块传递的需要写回REG FILE的信息 |
| es\_to\_ms\_bus | IN | 71 | EXE模块向MEM模块传递数据 |
| ms\_to\_ws\_bus | IN | 70 | MEM模块向WB模块传递数据 |
| es\_to\_ms\_valid | IN | 1 | EXE模块可以向MEM模块传值 |
| ms\_to\_ws\_valid | IN | 1 | MEM模块可以向WB模块传值 |
| out\_es\_valid | IN | 1 | 接收es\_valid是否为1 |
| out\_ms\_valid | IN | 1 | 接收ms\_valid是否为1 |

1. 功能描述

将从IF模块获取的指令进行译码，获得指令格式类型、ALU操作类型、是否需要加载、写回内存和参与运算数据的值、跳转的目标PC。判断PC是否需要跳转，并将结果返还给IF模块。将译码后的数据和控制信号在传递给EXE模块,EXE模块在下一时钟周期接受。另外，写回阶段的数据也通过该模块传递给寄存器堆。并且采用前递的方式来减少CPU的阻塞，缩短运行时间。

（六）重要模块5设计：执行阶段（EXE\_stage）模块

1. 工作原理

将从ID模块获取的指令相应的执行。将执行后的ALU结果和前阶段传递的通用寄存器写使能、写地址控制信号、PC传通过总线传递给MEM模块,MEM模块在下一周期接收新值。另外，load指令的发出读信号处理也在EXE阶段完成，EXE模块将数据传递给MEM模块，在下一周期WB阶段进行写回。输出数据RAM的写信号和数据。

此阶段在进行算数逻辑运算时，需要调用ALU模块。

1. 接口定义

表2.3 EXE\_stage接口定义

| **名称** | **方向** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| clk | IN | 1 | 时钟信号 |
| reset | IN | 1 | 复位信号 |
| ms\_allowin | IN | 1 | MEM模块允许接受EXE传值 |
| es\_allowin | OUT | 1 | EXE模块允许接受ID传值 |
| ds\_to\_es\_valid | IN | 1 | ID模块可以向EXE模块传值 |
| ds\_to\_es\_bus | IN | 136 | ID模块向EXE模块传递数据 |
| es\_to\_ms\_valid | OUT | 1 | EXE模块可以向MEM模块传值 |
| es\_to\_ms\_bus | OUT | 71 | EXE模块向MEM模块传递数据 |
| data\_sram\_en | OUT | 1 | data\_sram读使能 |
| data\_sram\_wen | OUT | 4 | data\_sram写使能 |
| data\_sram\_addr | OUT | 32 | data\_sram目标地址 |
| data\_sram\_wdata | OUT | 32 | data\_sram写数据 |
| out\_es\_valid | OUT | 1 | 将es\_valid的值传递给ID模块 |

1. 功能描述

将从ID模块获取的指令相应的执行。将执行后和前阶段传递的数据控制信号传通过数据总线在下一时钟周期更新给MEM模块。另外，load指令的发出读信号处理也在EXE阶段完成，EXE模块将数据传递给MEM模块，在下一周期进行写回。输出数据RAM的写信号和数据。

（六）重要模块6设计：访存阶段（MEM\_stage）模块

1. 工作原理

将从EXE模块获取的访存指令相应的执行。根据es\_to\_ms\_bs中的是否数据来自数据RAM信号确定是否有访存取出的数据，并将相应指令的最终结果和前阶段传递的通用寄存器写使能、写地址控制信号、PC传通过总线在下一时钟周期更新给WB模块。

1. 接口定义

表2.4 MEM\_stage接口定义

| **名称** | **方向** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| clk | IN | 1 | 时钟信号 |
| reset | IN | 1 | 复位信号 |
| ws\_allowin | IN | 1 | WB模块允许接受MEM传值 |
| ms\_allowin | OUT | 1 | MEM模块允许接受EXE传值 |
| es\_to\_ms\_valid | IN | 1 | EXE模块可以向MEM模块传值 |
| es\_to\_ms\_bus | IN | 71 | EXE模块向MEM模块传递数据 |
| ms\_to\_ws\_valid | OUT | 1 | MEM模块可以向EXE模块传值 |
| ms\_to\_ws\_bus | OUT | 70 | MEM模块向WB模块传递数据 |
| data\_sram\_rdata | OUT | 32 | data\_sram读出的数据 |
| out\_ms\_valid | OUT | 1 | 将ms\_valid的值传递给ID模块 |

1. 功能描述

将从EXE模块获取的访存指令相应的执行。确定是否有访存指令，并将相应指令的数据和前阶段传递的数据控制信号传通过总线在下一时钟周期更新给WB模块。

（七）重要模块7设计：访存阶段（WB\_stage）模块

1. 工作原理

将从MEM模块获取的写回指令相应的执行。确定是否有写回指令，并进行相应的操作。同时，该模块将PC、寄存器堆写使能、地址、和写回结果传递给debug模块，用于调试CPU的正确性。

表2.5 WB\_stage接口定义

| **名称** | **方向** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| clk | IN | 1 | 时钟信号 |
| reset | IN | 1 | 复位信号 |
| ws\_allowin | OUT | 1 | WB模块允许接受MEM传值 |
| ms\_to\_ws\_valid | IN | 1 | MEM模块可以向WB模块传值 |
| ms\_to\_ws\_bus | IN | 70 | MEM模块向WB模块传递数据 |
| ws\_to\_rf\_bus | OUT | 40 | WB模块向寄存器堆模块（通过ID模块）传递数据（包括ws\_gr\_we、ws\_valid等） |
| debug\_wb\_pc | OUT | 32 | debug显示PC |
| Debug\_wb\_rf\_wen | OUT | 4 | debug显示寄存器堆写使能 |
| Debug\_wb\_rf\_wnum | OUT | 5 | debug显示寄存器堆写地址 |
| Debug\_wb\_rf\_wdata | OUT | 32 | debug显示寄存器堆写数据 |

1. 功能描述

将从MEM模块获取的写回指令相应的执行。确定是否有写回指令，并进行相应的操作。同时，该模块将PC、寄存器堆写使能、地址、和写回结果传递给debug模块，用于调试CPU的正确性。

三、实验过程（50%）

（一）实验流水账

9月28日 20：00-20：30 阅读讲义

9月28日20：30-22：00 设计CPU数据通路，编写程序

10月5日09：00-12：00 撰写实验报告

（二）错误记录

1、错误1：值为X

（1）错误现象

运行仿真，比对trace发现wb\_rf\_wdata的值为“X”，如图3.1。

图片包含 屏幕截图

描述已自动生成

图3.1 wb\_rf\_wdata的值为“X”

（2）分析定位过程

如图3.1，wb\_rf\_wdata的值出现错误，为了查找bug方便，我增设了rs\_choice和rt\_choice变量来查看rs\_value和rt\_value采用了四选一中的哪一种赋值。通过查看波形发现，rs的读数不正常，即前部分为“X”而后部分存在数字。如图3.2，查看对应时刻rs\_choice的选择发现，rs\_value选择了第三种赋值。如图3.3，结合源代码发现，此处对于rs\_value传递的赋值出现了错误。

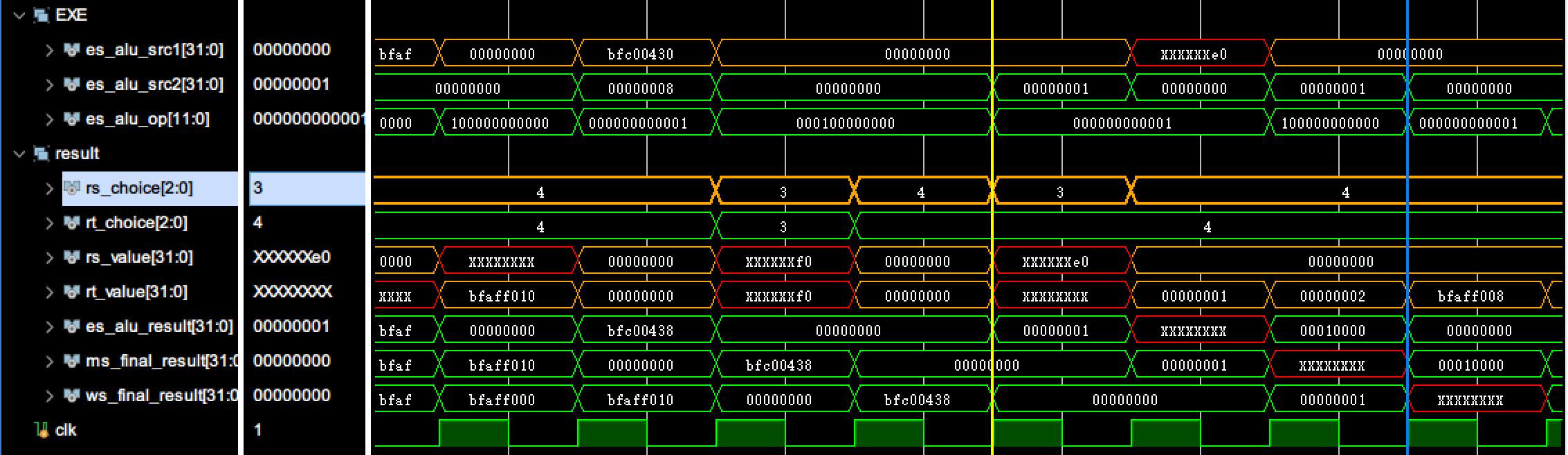


图3.2 rs\_choice的选择为3

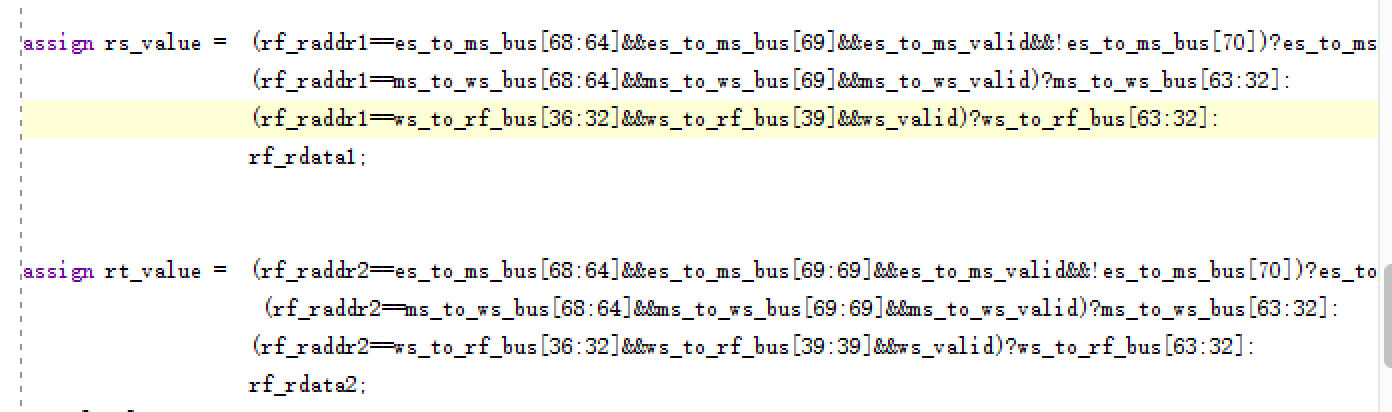


图3.3 rs\_value第三种选择出现错误

（3）错误原因

rs\_value和rt\_value的第三种选择赋值出现了错误，应该为ws\_to\_rf\_bus[31:0]。

（4）修正效果

将rs\_value和rt\_value的第三种选择赋值语句改成：

“ (rf\_raddr1==ws\_to\_rf\_bus[36:32]&&ws\_to\_rf\_bus[39]&&ws\_valid)?ws\_to\_rf\_bus[31:0]:”和

“(rf\_raddr2==ws\_to\_rf\_bus[36:32]&&ws\_to\_rf\_bus[39]&&ws\_valid)?ws\_to\_rf\_bus[31:0]:”。

如图3.4，所以测试通过。

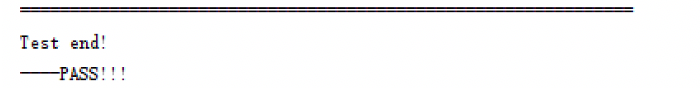


图3.4 所有测试通过

（二）对比分析

如图3.5和图3.6，分别采用阻塞和前递方式的CPU五级流水设计，在运行相同测试程序的情况下，运行时长分别文1，310，945ns和1，153，185ns。采用前递的流水线控制相较于阻塞，时间缩短了12.03%。可以看出采用前递的运行效率增加了。

图片包含 文字

描述已自动生成

图3.5 阻塞设计运行时长

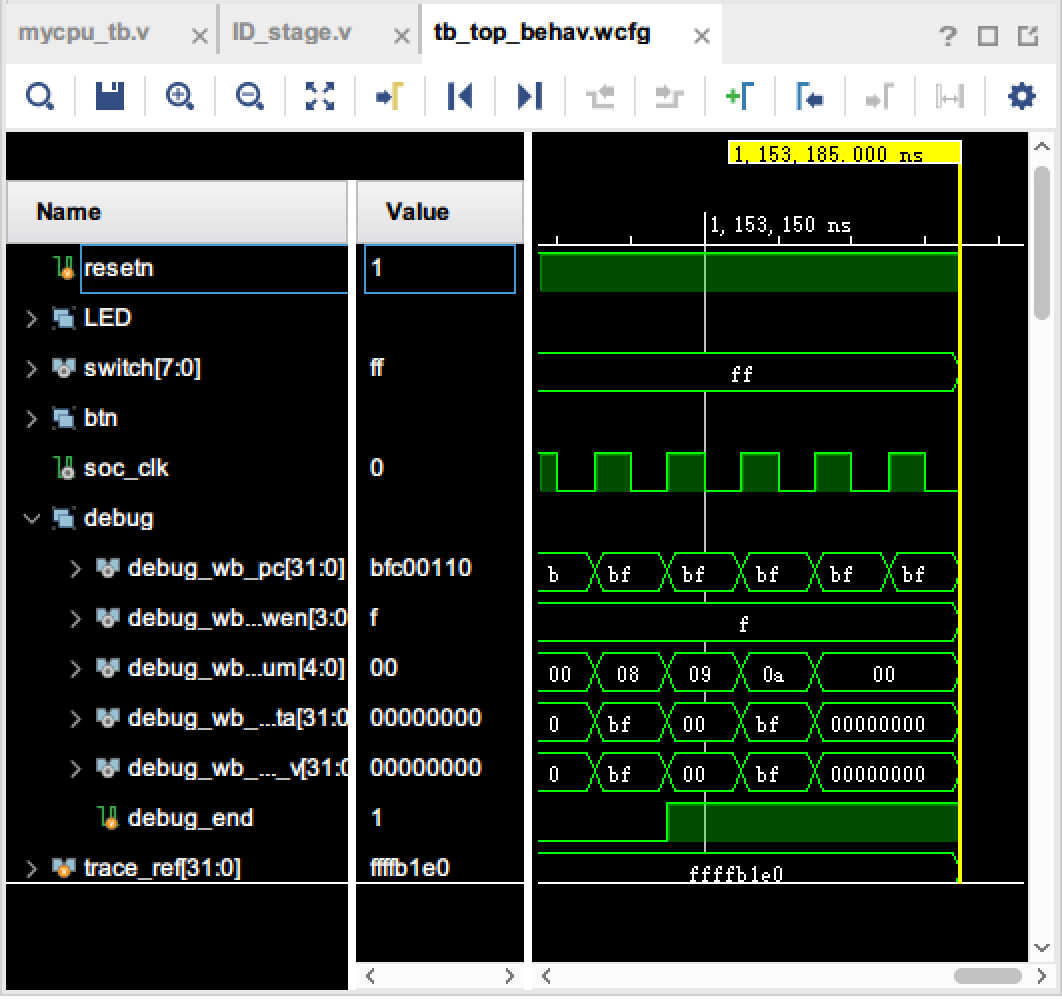


图3.6 前递设计运行时长

四、实验总结（可选）

本次实验是简单流水线CPU的最后一个阶段。通过三个阶段CPU的学习了解和动手实验，我对于五级流水线CPU的原理和设计有了更为深入的了解。此外本次实验中，我寻找和调试了老师设置的bug和自己写出的bug，通过多途径的bug查找定位，我相信自己能在今后的实验中，主动去规避一些错误写法。并且在遇到bug时，能在尽可能短的时间内将错误找到。