# 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：

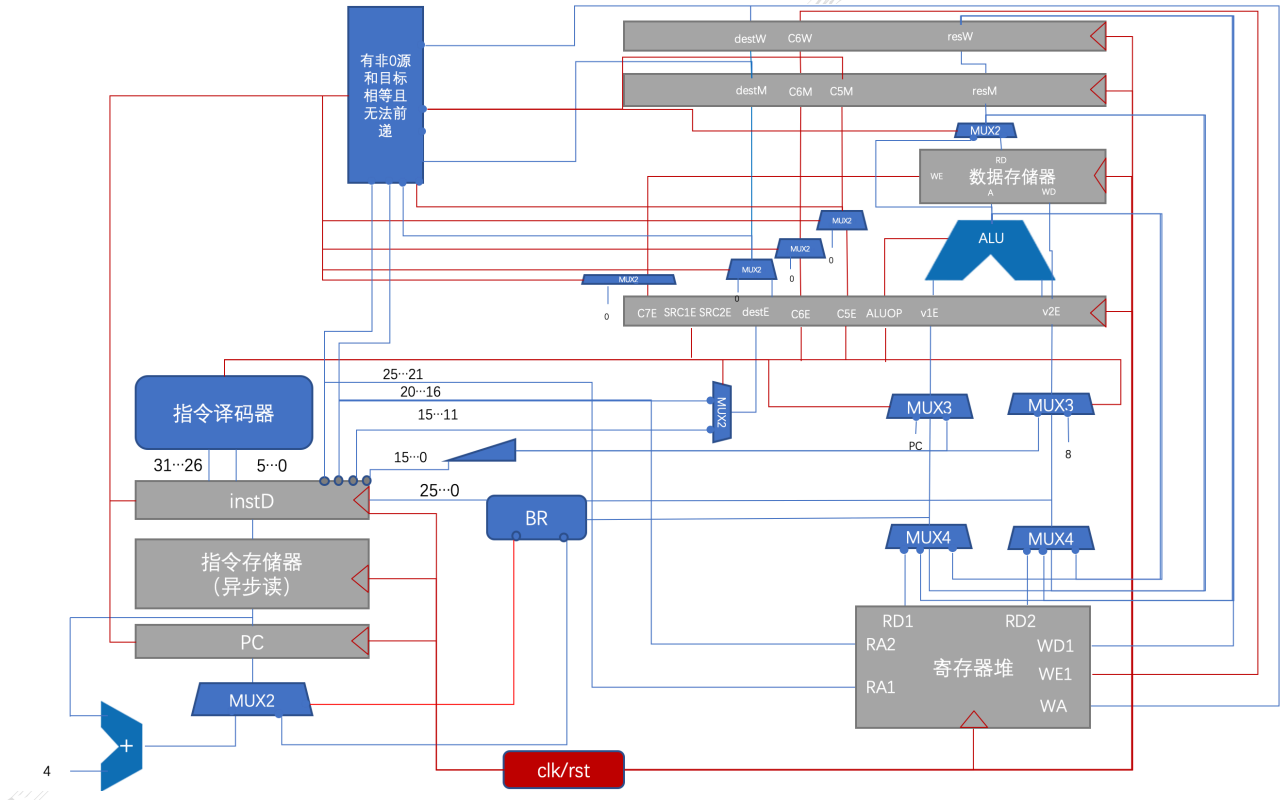


图 1.1 硬件结构设计图

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。

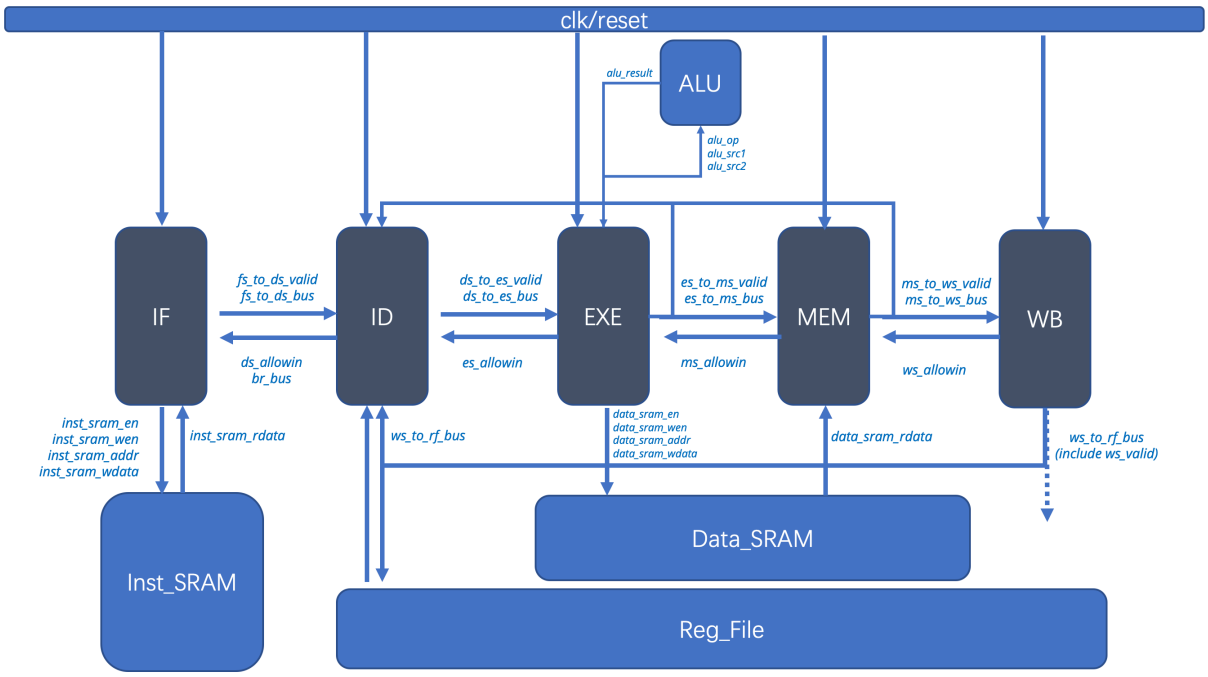


图 1.2 流水示意图

## （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。

### 2、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
```

### 3、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。

### 2、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
```

### 3、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 。

### 2、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 |

### 3、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”

### 2、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 |

### 3、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。

### 2、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 |

### 3、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。

### 2、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 |

| 名称 | 方向 | 位宽 | 功能描述 |
|---|---|---|---|
| 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 |

### 3、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 |

### 2、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。

```
[   2557 ns] Error!!!
    reference: PC = 0xbfc408b4, wb_rf_wnum = 0x12, wb_rf_wdata = 0x00000000
    mycpu    : PC = 0xbfc408b4, wb_rf_wnum = 0x12, wb_rf_wdata = 0xxxxxxxxx
```

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。

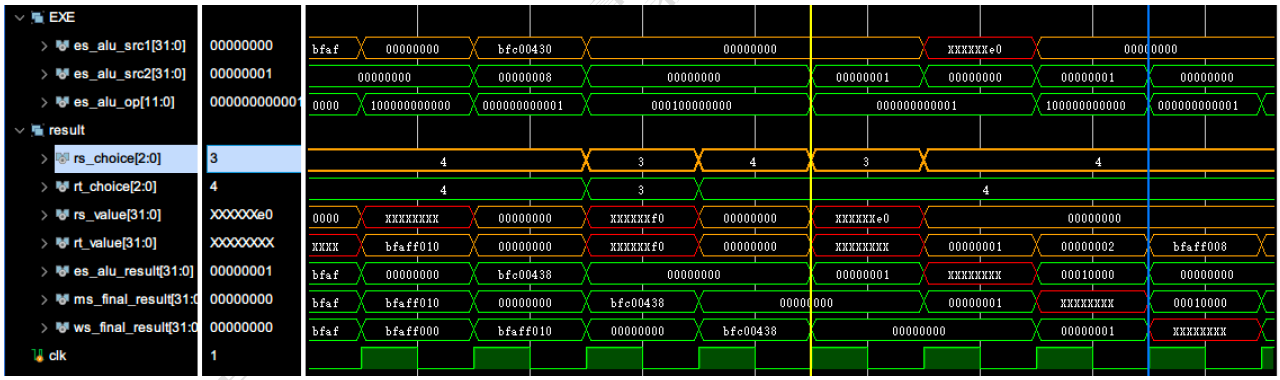


图 3.2 rs_choice 的选择为 3

```
assign rs_value =  (rf_raddr1==es_to_ms_bus[68:64]&&es_to_ms_bus[69]&&es_to_ms_valid&&!es_to_ms_bus[70])?es_to_ms
                   (rf_raddr1==ms_to_ws_bus[68:64]&&ms_to_ws_bus[69]&&ms_to_ws_valid)?ms_to_ws_bus[63:32]:
                   (rf_raddr1==ws_to_rf_bus[36:32]&&ws_to_rf_bus[39]&&ws_valid)?ws_to_rf_bus[63:32]:
                   rf_rdata1;


assign rt_value =  (rf_raddr2==es_to_ms_bus[68:64]&&es_to_ms_bus[69:69]&&es_to_ms_valid&&!es_to_ms_bus[70])?es_to
                    (rf_raddr2==ms_to_ws_bus[68:64]&&ms_to_ws_bus[69:69]&&ms_to_ws_valid)?ms_to_ws_bus[63:32]:
                   (rf_raddr2==ws_to_rf_bus[36:32]&&ws_to_rf_bus[39:39]&&ws_valid)?ws_to_rf_bus[63:32]:
                   rf_rdata2;
```

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。

```
Test end!
----PASS!!!
```

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。

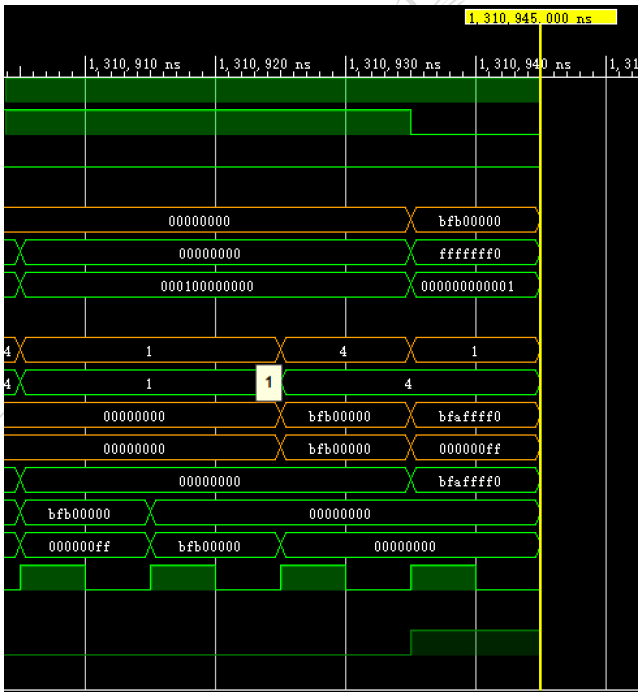


图 3.5 阻塞设计运行时长

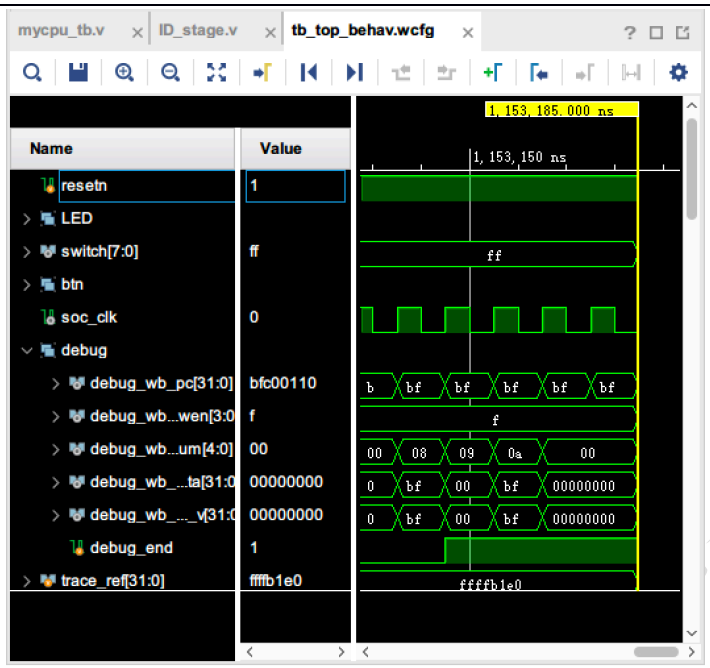


图 3.6 前递设计运行时长

## 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。