浙江大学

本科实验报告

课程名称：计算机体系结构

姓　　名：吴同

学　　院：计算机科学与技术学院

系：计算机科学与技术系

专　　业：计算机科学与技术

学　　号：3170104848

指导教师：陈文智

2019 年　　10 月　　30 日

浙江大学实验报告

专业：计算机科学与技术
姓名：吴同
学号：3170104848
日期：2019 年 10 月 30 日
地点：曹西 301

课程名称：计算机体系结构　指导老师：陈文智　电子邮件：wutongcs@zju.edu.cn
实验名称：带旁路的五级流水线 CPU 设计　实验类型：综合型　同组同学：徐欣苑

一、 实验目的和要求

1. 实验目的

- 理解流水线 CPU 旁路模块的原理
- 掌握流水线旁路的实现方法
- 掌握流水线旁路的条件判断
- 掌握流水线旁路的程序验证

2. 实验要求

- 设计五级流水线的旁路模块
- 修改 CPU 控制模块，判断触发旁路的条件
- 用程序验证 CPU 并观察程序的执行

二、 实验内容和原理

1. 实验内容

- 修改上次实验的代码，将数据竞争的停顿改为旁路
- 完成 CPU 内核模块的仿真
- 在 FPGA 上完成 CPU 的硬件实现并测试

2. 实验原理

旁路解决数据竞争

如果在流水线中，后序指令要读前序指令所写的寄存器，则会出现数据竞争。通过停顿可以解决数据竞争，但会降低流水线的效率。为更好地解决数据竞争的问题，我们采用旁路的方法。

旁路的控制逻辑为，如果当前指令（ID）所读的寄存器是上一条指令（EXE）所写的寄存器，如果上一条指令不从内存读数据，则在该指令执行（下一节拍的 EXE）时，直接将 MEM 阶段的保存 ALU 输出的锁存器之值送入 ALU；如果上一条指令从内存读数据，则在该指令执行（下一节拍的 EXE）时，直接将 MEM 阶段从内存取出的数据送入 ALU。如果当前指令（ID）所读的寄存器是上两条指令（MEM）所写的寄存器，则在该指令执行（下一节拍的 EXE）时，直接将 WB 阶段的写回值送入 ALU。

加入旁路后的数据通路如图 1所示。

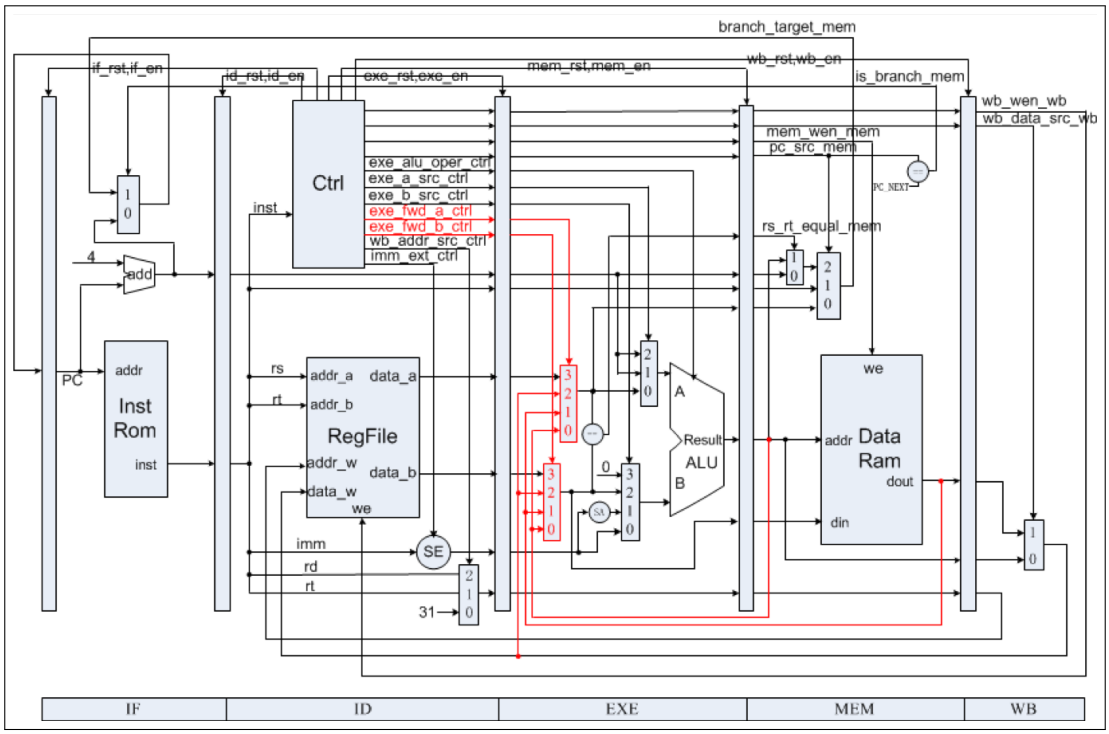

图 1: 加入旁路后的数据通路图

三、 实验过程和数据记录

对上一次实验的代码进行修改，删去 reg_stall 的信号，改用旁路实现。

controller.v

```
always @(*) begin
   exe_fwd_a_ctrl = 0;
   exe_fwd_b_ctrl = 0;
   if (rs_used && addr_rs != 0)
   begin
      if (regw_addr_exe == addr_rs && wb_wen_exe)
      begin
         if (mem_ren_exe)
            exe_fwd_a_ctrl = 2;
         else
            exe_fwd_a_ctrl = 1;
      end
      else if (regw_addr_mem == addr_rs && wb_wen_mem)
      begin
         exe_fwd_a_ctrl = 3;
      end
   end
   if (rt_used && addr_rt != 0)
   begin
      if (regw_addr_exe == addr_rt && wb_wen_exe)
      begin
         if (mem_ren_exe)
```

```
            exe_fwd_b_ctrl = 2;
         else
            exe_fwd_b_ctrl = 1;
      end
      else if (regw_addr_mem == addr_rt && wb_wen_mem)
      begin
         exe_fwd_b_ctrl = 3;
      end
   end
end
```

datapath.v

```
always @(*) begin
   data_rs_modified = data_rs_exe;
   data_rt_modified = data_rt_exe;
   case (fwd_a)
      0: data_rs_modified = data_rs_exe;
      1: data_rs_modified = alu_out_mem;
      2: data_rs_modified = mem_din;
      3: data_rs_modified = regw_data_wb;
   endcase
   case (fwd_b)
      0: data_rt_modified = data_rt_exe;
      1: data_rt_modified = alu_out_mem;
      2: data_rt_modified = mem_din;
      3: data_rt_modified = regw_data_wb;
   endcase
end
```

同时，修改两个代码文件的输入输出接口，并修改 mips_core.v。

对修改后的 CPU 核进行仿真。仿真测试中，相同的测试程序运行时间比上一次实验减少了 500ns，即 25 个时钟周期。仿真结果如下：

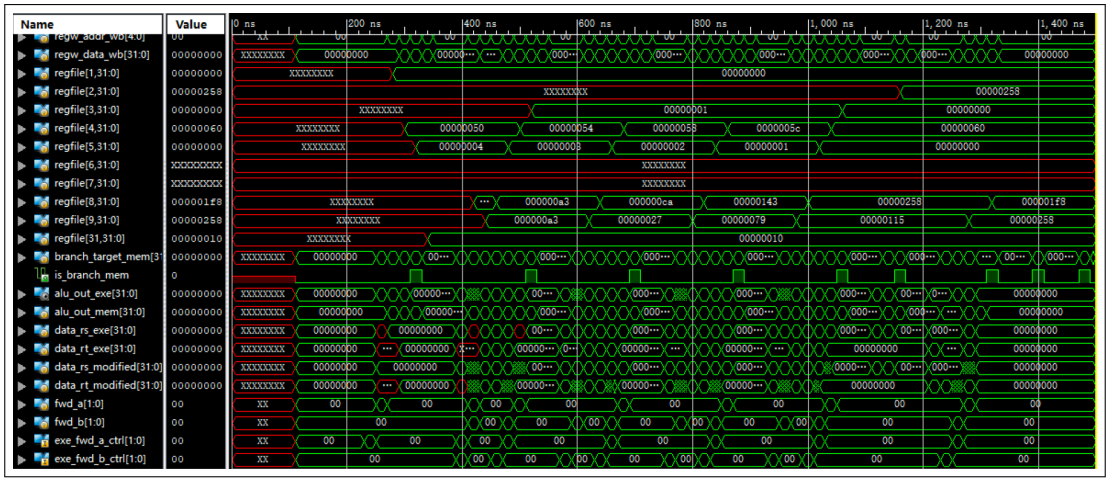

图 2: 仿真测试结果（$0 \sim 1500ns$）

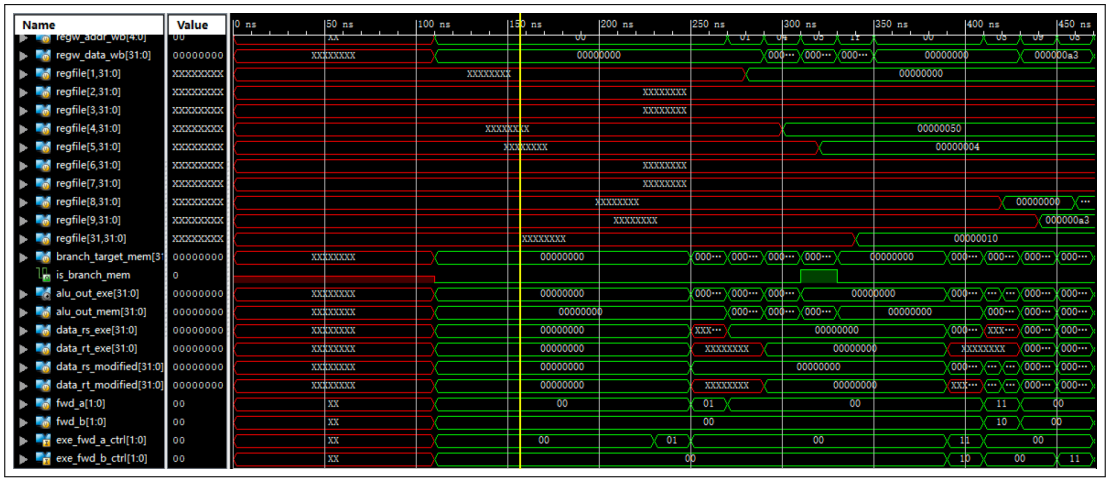

图 3: 仿真测试结果（0 ∼ 450ns）

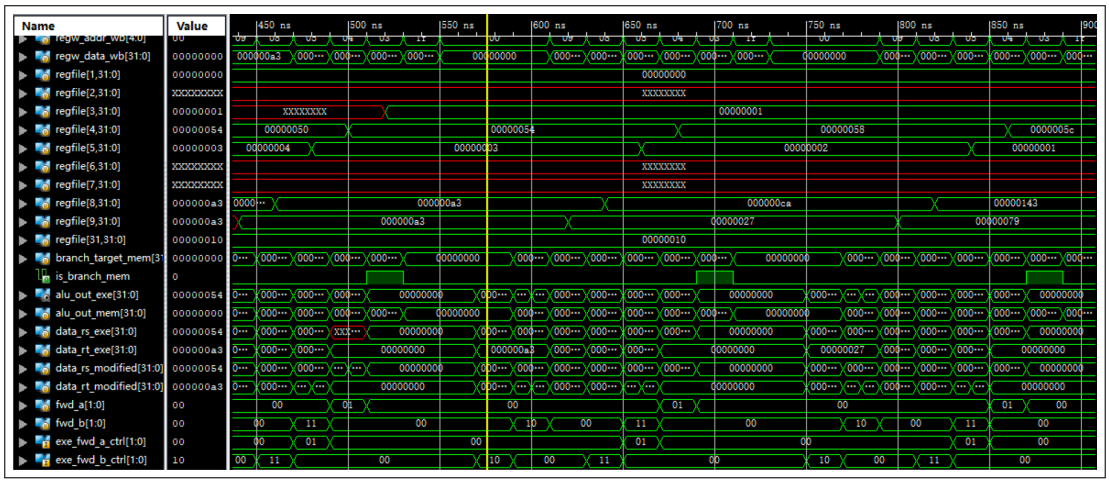

图 4: 仿真测试结果（450 ∼ 900ns）

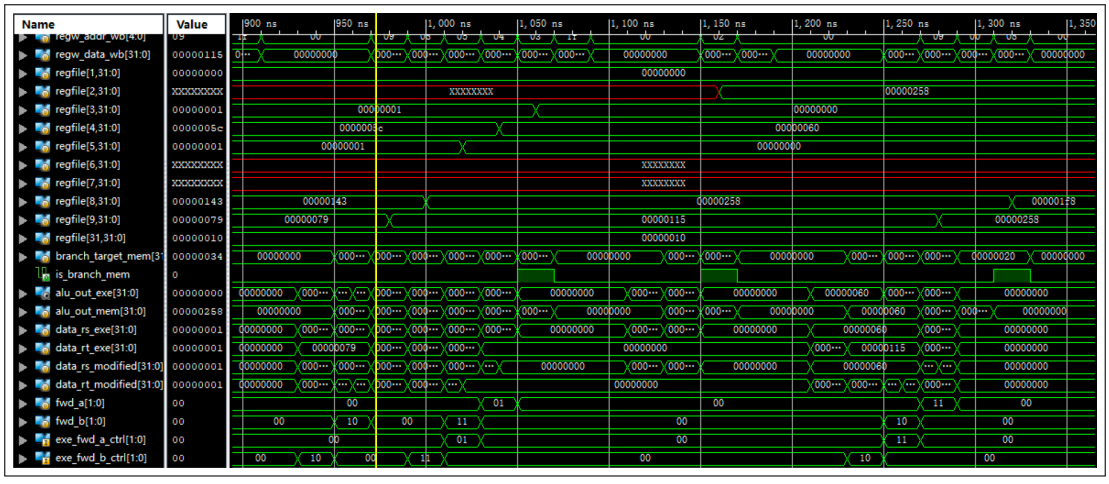

图 5: 仿真测试结果（900 ∼ 1350ns）

将修改后的代码进行综合，烧入 FPGA 板中，测试程序的执行结果与前两次实验相同。

四、 实验结果分析

在测试程序执行过程中，ALU 的操作数 A 使用了七次旁路，ALU 的操作数 B 使用了九次旁路。在图 6中，上一条 and 指令要写 1 号寄存器，本条指令的 rs 为 1，则 exe_fwd_a_ctrl 信号为 1。

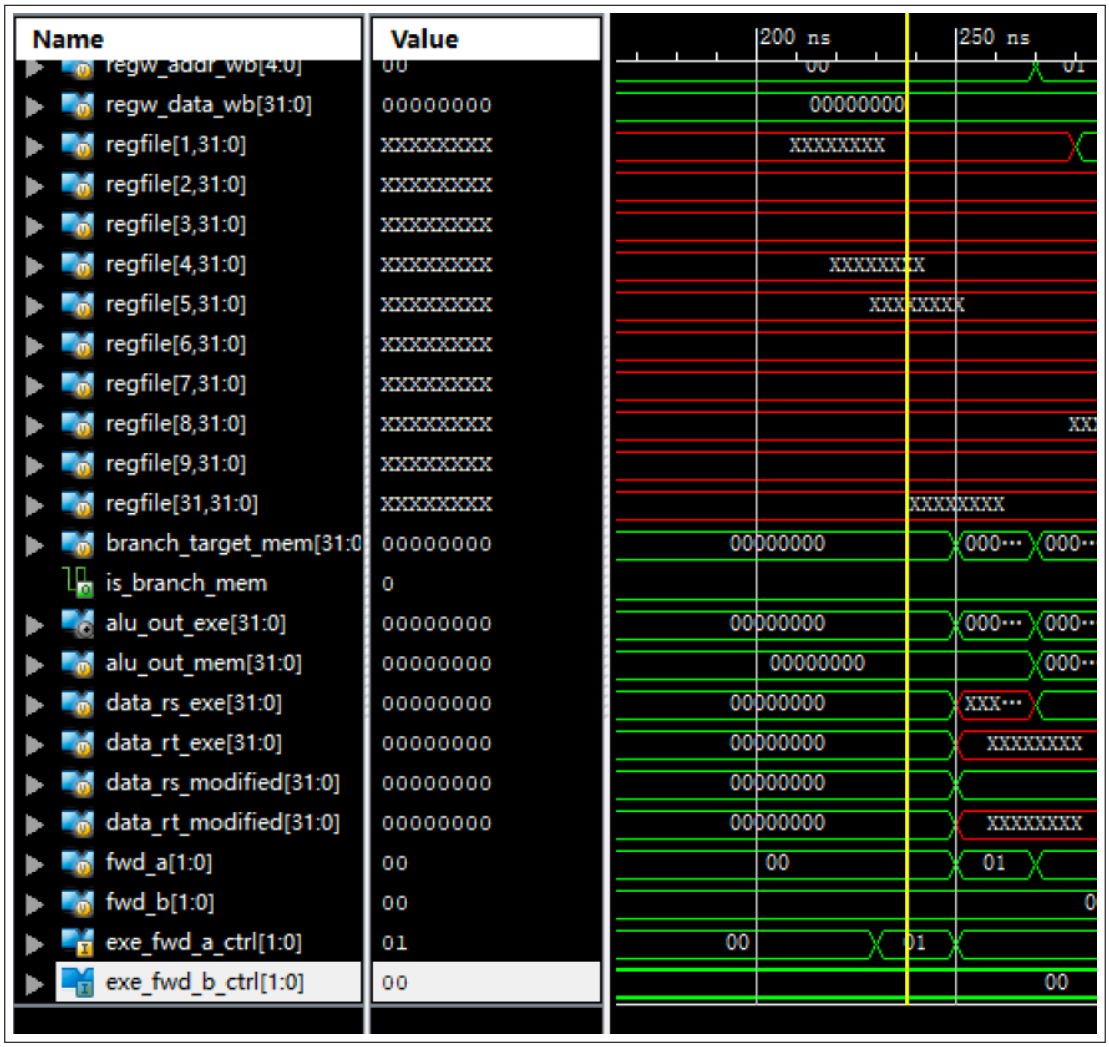

图 6: 旁路实例 1

在图 6中，上一条 ls 指令要写 9 号寄存器，再上一条 add 指令要写 8 号寄存器。本条指令的 rs 为 8，则 exe_fwd_a_ctrl 信号为 3；rt 为 9，则 exe_fwd_a_ctrl 信号为 2。

图 7: 旁路实例 2

五、 讨论与心得

本次实验在上一次做好的五级流水线 CPU 的基础上，去掉了数据竞争的停顿，用旁路进行解决。这次实验与前两次实验不同。前两次实验是在所给的代码框架上进行填空，这次实验要自己添加代码。在添加信号和修改数据通路的过程中，由于较长时间未写 Verilog HDL 代码，出现了很多的低级错误，比如不指定数据的位宽，EXE 阶段未增加旁路控制信号的锁存器。旁路的逻辑并不是十分复杂，实验过程的问题主要出在代码的写法上。根据仿真结果和警告信息，进行逐个的修改，最终将实验完成。