Lab 3 报告

学号 2020K8009929048
姓名 尹群杰
箱子号 50

一、实验任务（10%）

Lab3 主要任务为调试已有的 20 条指令单周期 CPU 的代码，修改代码使其能正常运行，同时理解 CPU 的数据通路，检验方式为通过运行 Vivado 仿真并且上板运行观察验证。

二、实验设计（40%）

（一）总体设计思路

本次实验为单周期 CPU，因此不需要其他的寄存器或模块来实现状态机等，因此只需完成基本的数据通路的连接即可。

首先是 PC 的更新部分，如图 1 所示

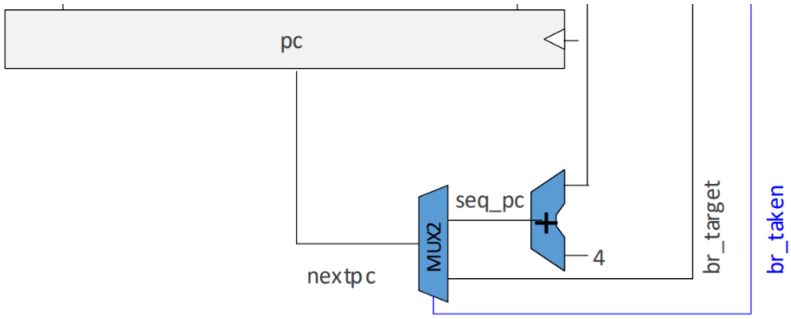

图 1：PC 更新结构图

PC 的更新通过 nextPC 进行，主要输入值为 PC+4 或者转移指令的目的地址，因此需要一个选择器来实现，相关代码如下：

```
assign seq_pc    = fs_pc + 3'h4;
assign nextpc    = br_taken ? br_target : seq_pc;

always @(posedge clk) begin
    if (!resetn) begin
        pc <= 32'h1c000000;
    end
    else begin
        pc <= nextpc;
    end
end
```

图 2：PC 更新相关代码

在取指完成后，从 inst_ram 中会取得指令，此后经历译码单元完成译码过程以及部分立即数和地址产生，译码单元将在之后阐述，该部分相关代码如下：

```
assign op_31_26  = inst[31:26];
assign op_25_22  = inst[25:22];
assign op_21_20  = inst[21:20];
assign op_19_15  = inst[19:15];

assign rd   = inst[ 4: 0];
assign rj   = inst[ 9: 5];
assign rk   = inst[14:10];

assign i12  = inst[21:10];
assign i20  = inst[24: 5];
assign i16  = inst[25:10];
assign i26  = {inst[ 9: 0], inst[25:10]};

decoder_6_64 u_dec0(.in(op_31_26 ), .out(op_31_26_d ));
decoder_4_16 u_dec1(.in(op_25_22 ), .out(op_25_22_d ));
decoder_2_4  u_dec2(.in(op_21_20 ), .out(op_21_20_d ));
decoder_5_32 u_dec3(.in(op_19_15 ), .out(op_19_15_d ));
```

图 3：译码部分相关代码

对于寄存器读取地址的选择如下图所示：

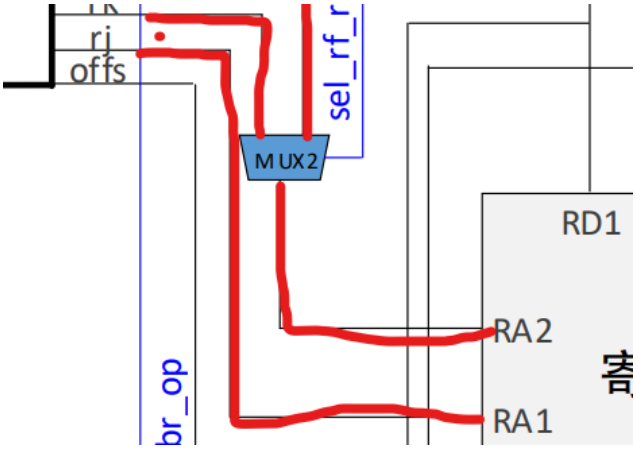

图 4：寄存器读取地址结构图

由于第二个操作数的地址可能为 rd 或 rk 从而需要一个选择器来完成该步骤相关代码如下：

```
assign rf_raddr1 = rj;
assign rf_raddr2 = src_reg_is_rd ? rd :rk;
```

图 5：寄存器读取地址代码

对于 ALU 操作数的选择，如图 6 所示：

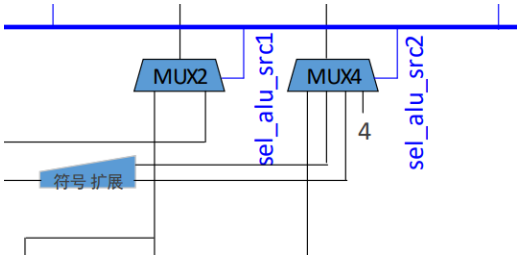

图 6：ALU 操作数结构图

因为 ALU 的第一个操作数，存在有 PC 和 rj 地址读取到的数据两种可能，因此需要选择器控制该部分，对于

第二个操作数则存在有 PC+4 时需要的 4 或其他立即数的情况，读取到的 rk 或 rd 的值，因此需要一个选择器完成，同时立即数的选择和寄存器数据的选择也可由选择器完成，因此相关代码如下：

```
assign alu_src1 = src1_is_pc  ? pc[31:0] : rj_value;
assign alu_src2 = src2_is_imm ? imm : rkd_value;
```

图 6：ALU 操作数相关代码

对于寄存器写的部分，写地址的选择包括部分转移指令要求的 1 号寄存器，因此需要一个选择器来完成，对于，而写入数据包括 ALU 的结果和从 data_ram 中读取到的数据两种情况，因此也需要一个选择器来完成。

（二）重要模块 1 设计：ALU 模块

1、工作原理

ALU 用于进行数据的相关处理，包括运算，移位两部分内容。

2、接口定义

输入信号为 alu_op, alu_src1, alu_src2，输出信号为 alu_result。

3、功能描述

内部设计根据外部输入的 alu_op，译码得到各类运算的相关控制信号，再通过相关控制信号选择各类运算的输出结果。

（三）重要模块 2 设计：译码器模块

1. 工作原理

译码器包含四类，分别为 2-4 译码器、4-16 译码器、5-32 译码器和 6-64 译码器，主要通过输入信号的值拉高相应输出信号对应位置的值

2. 接口定义

输入接口为 in，输出接口为 out。

3. 功能描述

通过把 in 所对应的值相同位置的 out 信号的部分拉高来实现。

三、实验过程（50%）

（一）实验流水账

2022 年 8 月 31 日 17:47，找到 bug1，编译出错部分。

2022 年 8 月 31 日 18:10，找到 bug2，PC 部分。

2022 年 8 月 31 日 21:01，找到 bug3，ALU 接口部分。

2022 年 8 月 31 日 21:18，找到 bug4，写入寄存器的数据部分。

2022 年 8 月 31 日 21:30，找到 bug5，PC 更新部分。

2022 年 8 月 31 日 21:55，找到 bug6，bl 指令的写寄存器信号控制部分。

2022 年 8 月 31 日 22:46，找到 bug7，移位运算的操作数部分。

2022 年 8 月 31 日 22:54，找到 bug8，or 运算出错

2022 年 8 月 31 日 23:18，找到 bug9，sra 运算结果位数出错。

2022 年 8 月 31 日 23:55，上板实验，通过。

（二）错误记录

1、错误 1：信号未命名

（1）错误现象

编译出错，error 信息显示'fs_pc', 'ds_pc', 'ds_valid', 'rfrom_mem'未声明。

（2）分析定位过程

error 信息显示 fs_pc', 'ds_pc', 'ds_valid', 'rfrom_mem'未声明。

（3）错误原因

未声明信号

（4）修正效果

删除相关无用信号'ds_valid', 'rfrom_mem'，声明'fs_pc', 'ds_pc'并赋值。

2、错误 2：wire 型变量未赋值

（1）错误现象

debug_wb_rf_we 信号显示为“Z”。

（2）分析定位过程

找到 debug_wb_rf_we 信号相关赋值部分

（3）错误原因

发现代码中，赋值时未使用 debug_wb_rf_we 信号，而使用了 debug_wb_rf_wen 信号。

（4）修正效果

debug_wb_rf_wen 更改为 debug_wb_rf_we。

3、错误 3：加法结果出错 1

（1）错误现象

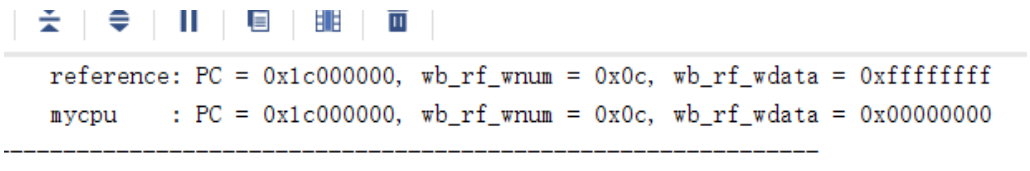

图 7：加法结果出错 1—现象 1

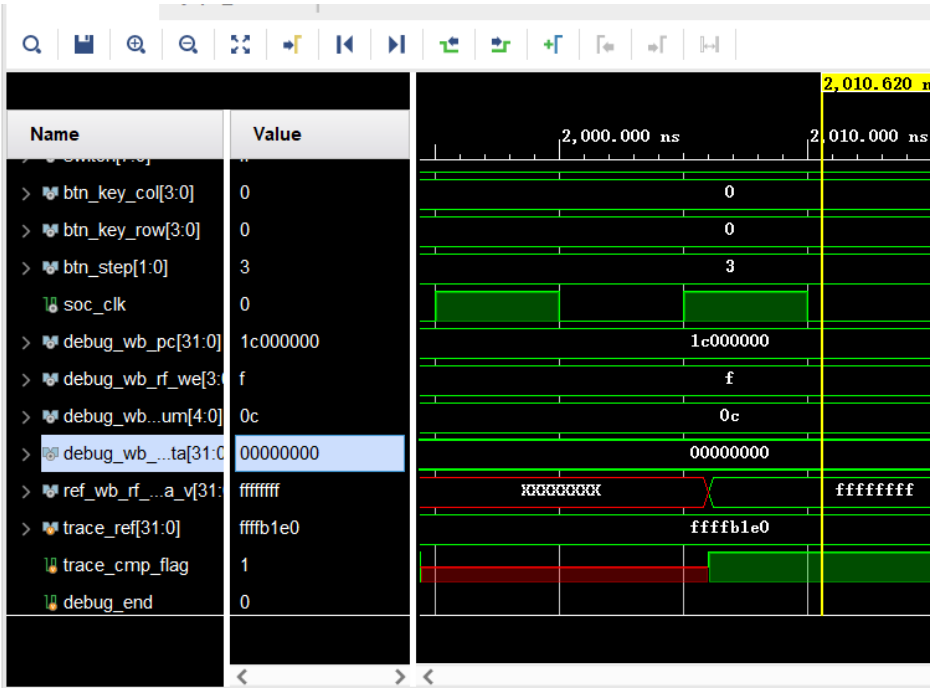

图 8：加法结果出错 1—现象 2

（2）分析定位过程

找到 alu 相关部分，发现 alu 内部加法无错误，从而在 CPU 设计中找错误，此时发现加法两端分别为 0x00000000 和 0xffffffff，加发结果发现仍然为 0x00000000，从而判断为操作时读取时出现的问题，从而发现 alu 接口处出现错误。

（3）错误原因

ALU 接口名出错，把 alu_src1 的接口修改正确。

4、错误 4：加法结果出错 2

（1）错误现象

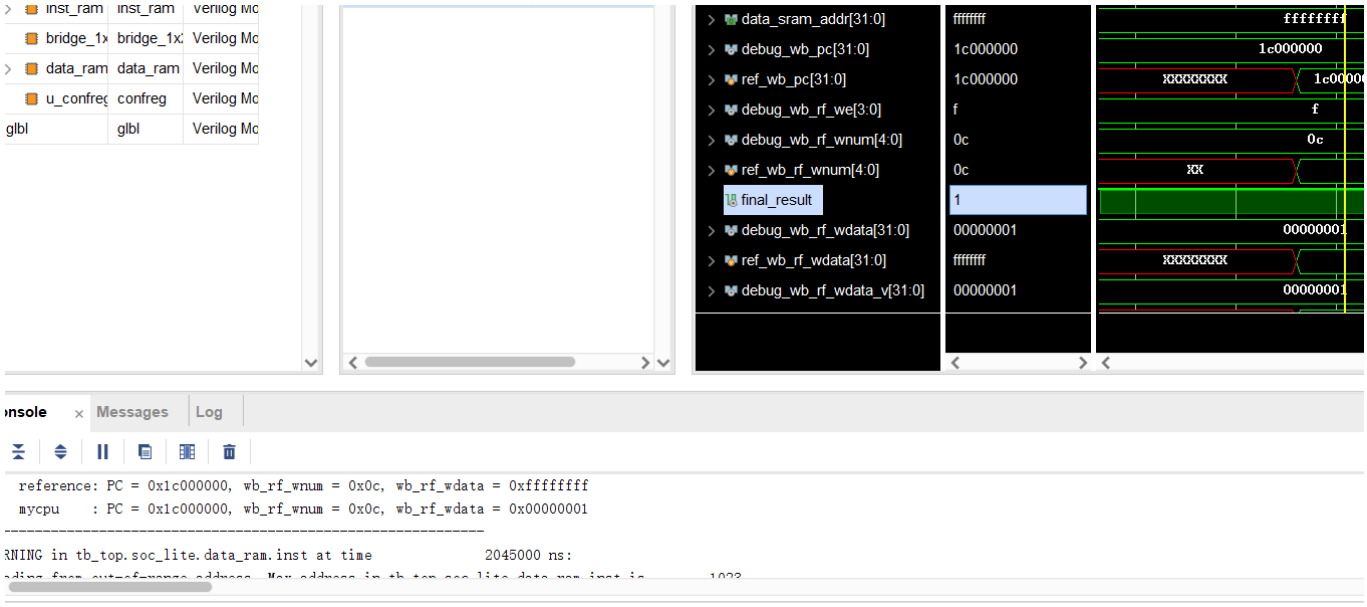

图 9：加法结果出错 2—现象

（2）分析定位过程

在确认错误 3 修改正确的基础之上，寻找发现，仅保留了最低位的 1，从而找到 final_result 仅有 1 位

（3）错误原因

声明 final_result 信号为 32 位

（4）修正效果

加法运算结果正确。

5、错误 5：PC 更新慢一拍

（1）错误现象

PC 更新慢一拍。

（2）分析定位过程

找到 PC 更新的部分，发现 PC 更新由 reset 信号控制，而 reset 信号将在 resetn 信号更新的下一拍写入，因此将会产生错误

（3）错误原因

PC 更新由 reset 信号控制，reset 信号将在 resetn 信号更新的下一拍

（4）修正效果

修改 PC 更新的控制信号 reset 变为!resetn，PC 更新结果正确。

6、错误 6：bl 指令寄存器写信号未拉高

（1）错误现象

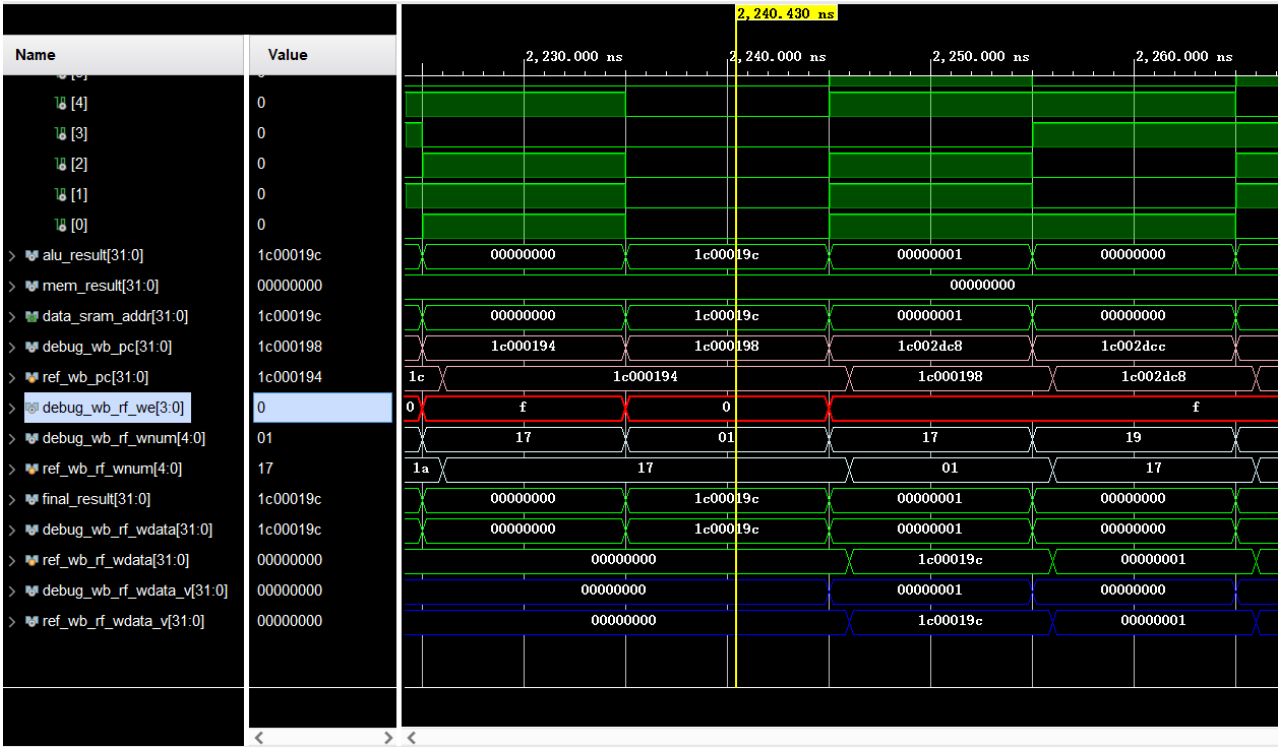

图 10：bl 指令寄存器写信号未拉高一现象

（2）分析定位过程

观察图 10 波形发现，reference 与 CPU 信号比较时（相同颜色的信号），reference 下一拍的结果与图中黄线这一拍的结果相同，发现 reference 的比较慢了一拍，阅读 mycpu_tb 文件时发现，寄存器写信号未拉高是不会对 reference 进行比较，从而发现寄存器写的控制信号有误。

（3）错误原因

Bl 指令时未拉高寄存写信号。

（4）修正效果

修改寄存写信号，使其在 Bl 指令时拉高。

7、错误 7：移位运算出错

（1）错误现象

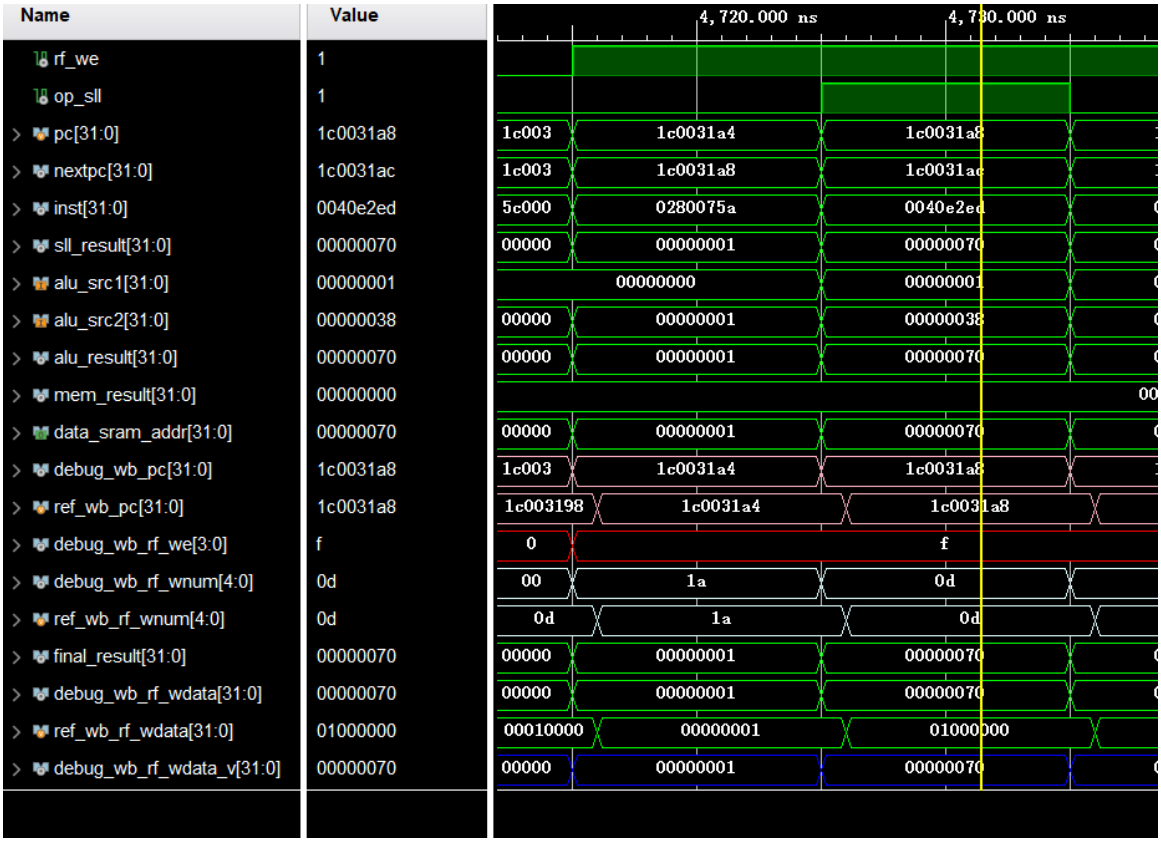

图 11：移位运算出错—现象

（2）分析定位过程

查阅手册发现该条指令为移位指令，从而转移至 alu.v 文件中检查，发现移位时选择的移位信号和移位位数位置相反。

（3）错误原因

移位信号和移位位数位置相反。

（4）修正效果

修改移位信号和移位位数的位置。

8、错误 8：OR 运算出错

（1）错误现象

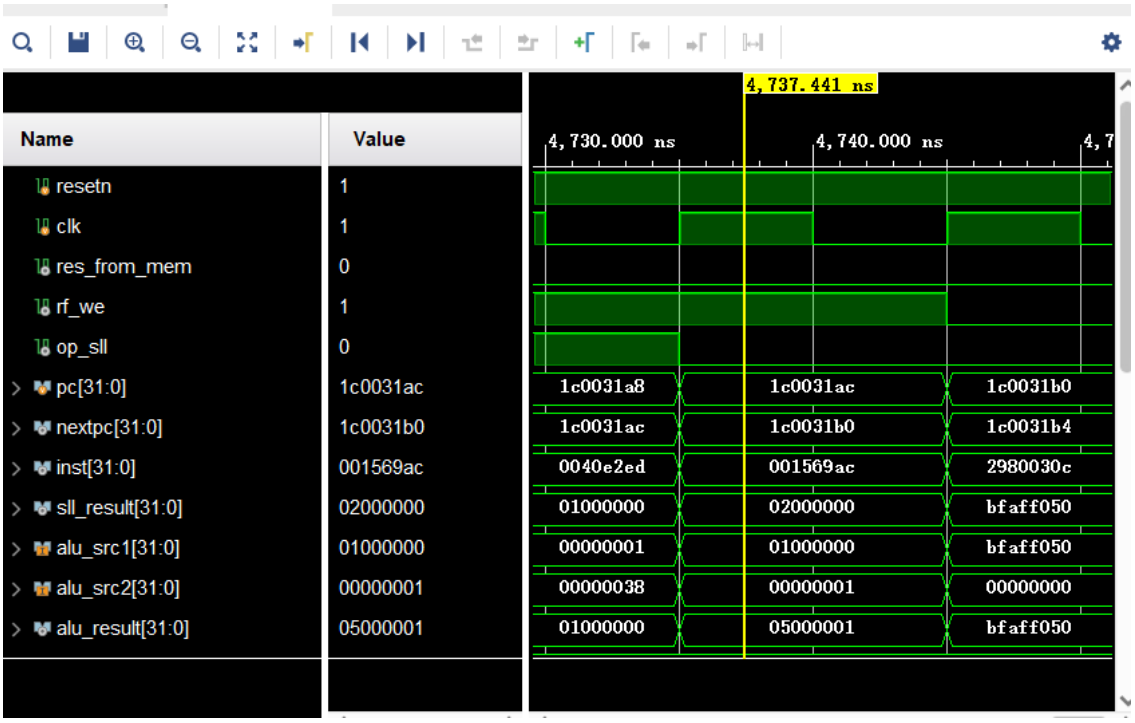

图 12：OR 运算出错—现象

（2）分析定位过程

找到 alu.v 文件中 or 运算的部分，发现 or 运算出错

（3）错误原因

or 运算出错。

（4）修正效果

修改 or 运算。

9、错误 9：sr 运算出错

（1）错误现象

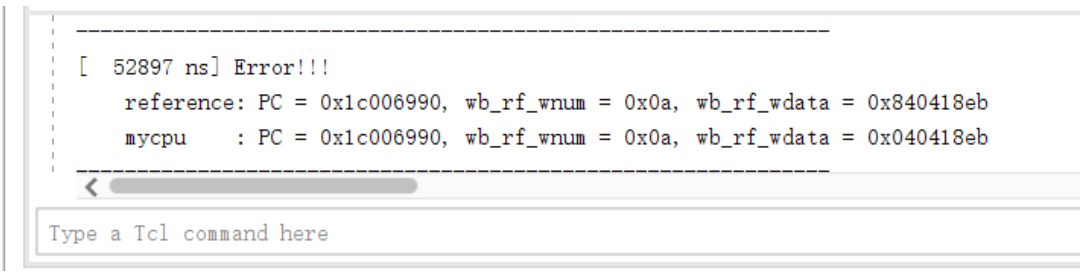

图 13：sr 运算出错—现象

（2）分析定位过程

找到 alu.v 文件中 sr 运算部分，发现 sr 运算本身无误，检查结果时，发现应当去 64 位中低 32 结果的 sr_result 变为了 31 位。

（3）错误原因

sr_result 变为了 31 位。

（4）修正效果

修改 sr_result 为 32 位。

四、实验总结

本次实验为单周期的实验，总体难度不大。