**实验3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无阻塞五级流水线CPU的设计思路，并调试给出的CPU设计代码，本次实验分为了两项子任务，其中：

子任务一:阅读代码并结合讲义，画出简单流水线 CPU 的结构设计框图和软件部分流程图。

子任务二: 结合仿真和上板测试，对给出的代码进行调试，找出其中的7处bug。（实际找出8处bug）

二、实验设计（40%）

（一）总体设计思路

**硬件结构设计图如下：**

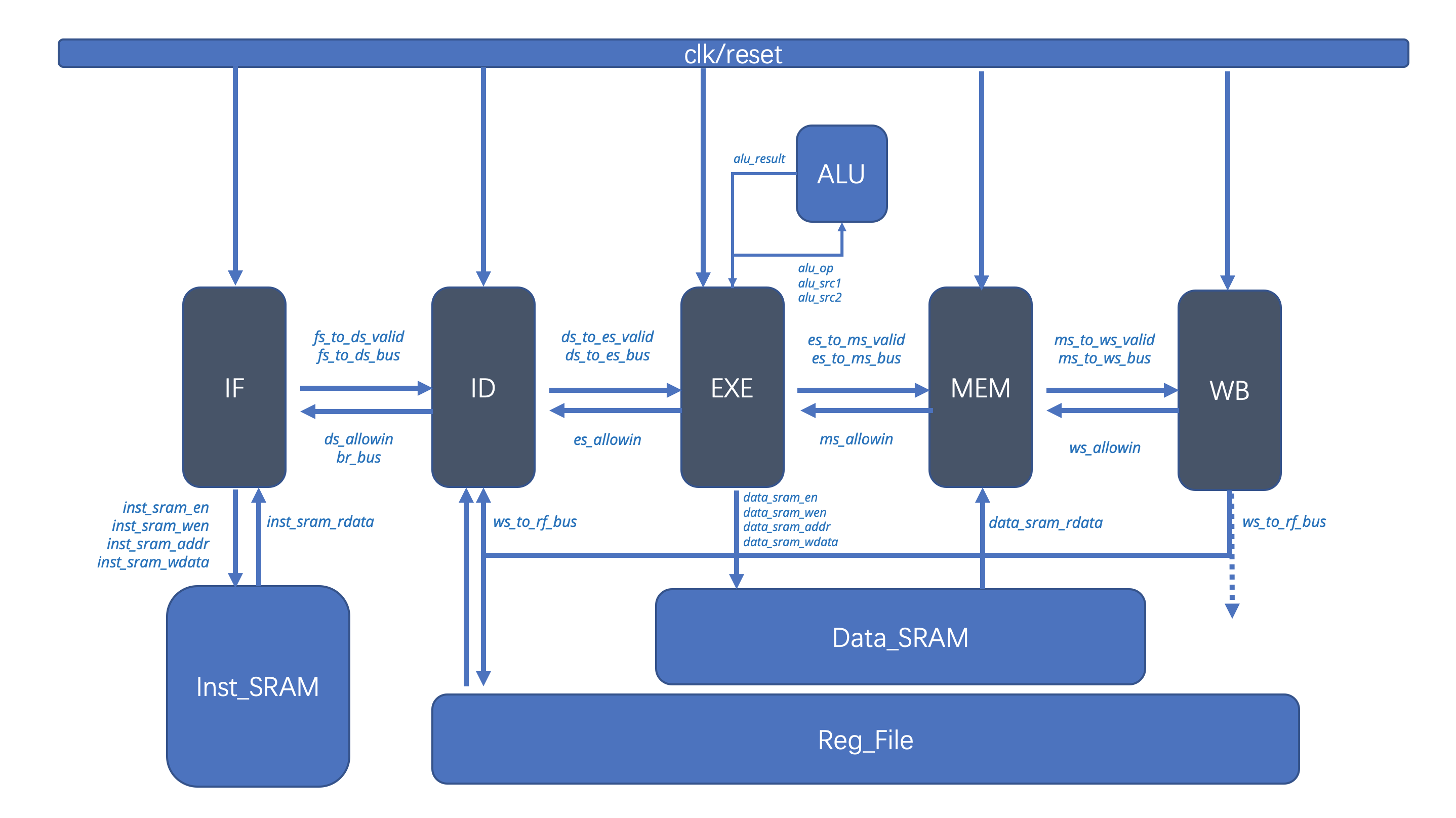
****

图1.1 硬件结构设计图

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。并且采用Tools模块进行译码。

（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，在下一周期再传递给ID模块 。

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bus，在下一周期再传递给ID模块。

（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，并处理PC是否需要跳转，将结果返还给IF模块。将译码后的数据和控制信号传通过数据总线在下一时钟周期递给EXE模块。另外，写回阶段的数据也通过该模块传递给寄存器堆。此模块需要调用寄存器堆模块和decode模块。

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 |

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。并处理PC是否需要跳转，将结果返还给IF模块。将译码后的数据和控制信号在下一时钟周期传递给EXE模块。另外，写回阶段的数据也通过该模块传递给寄存器堆。

（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递给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。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 |

1. 功能描述

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传通过总线在下一时间周期更新给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。此阶段在进行算数逻辑运算时，需要调用ALU模块。

（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。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递给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 |

1. 功能描述

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递给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。

1. 接口定义

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 | 38 | WB模块向寄存器堆模块传递数据 |
| 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14日 20：00-22：00 阅读讲义

9月15日10：00-22：00 进行试验撰写实验报告

9月16日19：00-23：00 进行试验撰写实验报告

以下错误记录 也就是记录 子任务二 的完成过程。

（二）错误记录

1、错误1：PC跳转错误

（1）错误现象

运行仿真，比对trace发现PC的值发生跳转错误，如图3.1。

（2）分析定位过程

如图3.1，PC出现错误，这说明跳转指令可能出现问题，对应查找PC的跳转相关指令，发现br\_bus只有32位。通过找br\_bus的定义和赋值语句发现，其定义语句中使用了宏定义，通过查看其宏定义，发现出现了“`define BR\_BUS\_WD 32”，这表明宏定义中的位宽出现了错误。

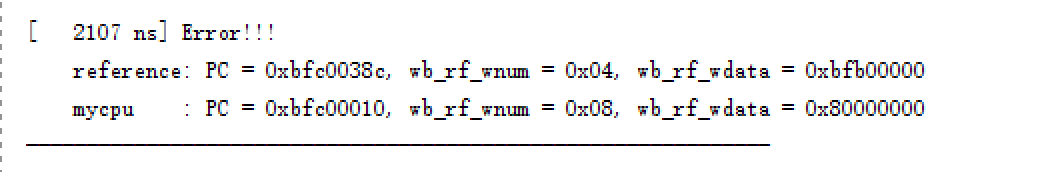


图3.1.1 IF模块里的BR\_BUS信号出现Z

（3）错误原因

br\_bus的宏定义语句出现了“`define BR\_BUS\_WD 32”，这表明宏定义中的位宽出现了错误。在实际使用过程中，br\_bus的最真实位宽应该是33，这导致到最终PC跳转错误。

（4）修正效果

如图3.2，通过将宏定义语句里的32改成33，PC跳转正常，此处error消失。

图片包含 绿色

描述已自动生成

图3.1.2 BR\_BUS\_WD 位宽改为33

（5）归纳总结（可选）

采用宏定义来定义位宽有便于修改的好处。但在今后的代码书写过程中，要养成写开发文档的习惯，最后根据br\_bus开发文档中的位宽，来对代码中的宏定义进行二次确认。

2、错误2：信号“X”

（1）错误现象

ID模块中，ds\_valid信号为X。

（2）分析定位过程

如图3.2，ds\_valid信号为X，通过找ds\_valid的定义和赋值语句发现，源代码并未给ds\_valid变量进行赋值。

图片包含 红色, 物体

描述已自动生成

图3.2 ID模块里的ds\_valid信号出现X

（3）错误原因

ds\_valid信号为X，其原因是源代码并未给ds\_valid变量进行赋值。

（4）修正效果

在always语句中，增加对ds\_valid的赋值语句：

if(reset)begin

ds\_valid<=0;

end

else if (ds\_allowin) begin

ds\_valid <= fs\_to\_ds\_valid;

end

修改完毕后，X信号消失，如图3.3。



图3.3 ID模块里的ds\_valid信号中X消失

（5）归纳总结（可选）

对于模式相近的代码块，可以采取复制粘贴框架的模式，以免一些重要组成部分忘记赋值。

3、错误3：信号“Z”

（1）错误现象

ID模块中，ds\_to\_es\_bus最高位信号为Z。

（2）分析定位过程

如图3.4，ds\_to\_es\_bus的123位(第124位)出现Z信号，通过找ds\_to\_es\_bus的定义和赋值语句发现，该bus组成部分中的load\_op并没有被赋值。

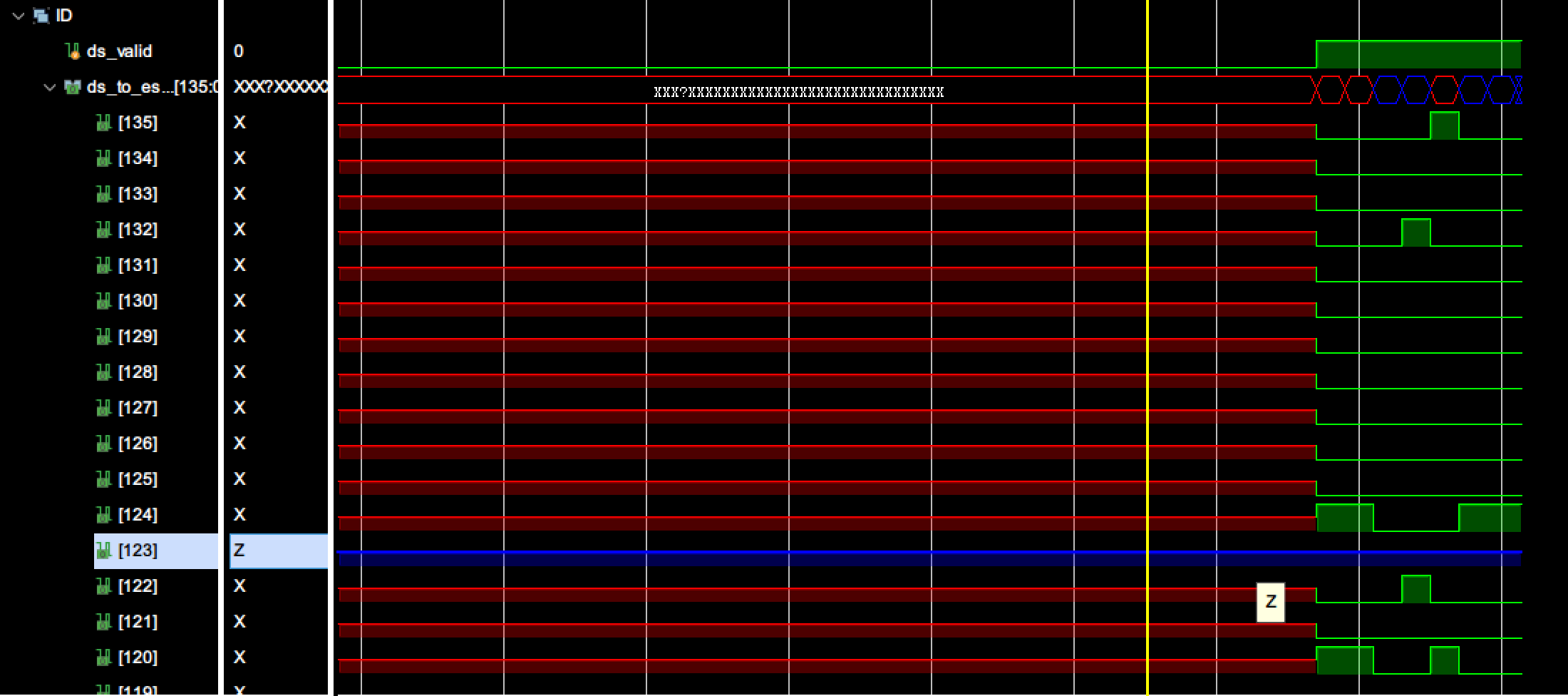


图3.4 ID模块里的ds\_to\_es\_bus信号出现Z

（3）错误原因

ds\_to\_es\_bus组成部分中，load\_op变量并没有被赋值，因此在ds\_to\_es\_bus中出现了“Z”信号。

（4）修正效果

通过将ds\_to\_es\_bus中的load\_op进行赋值，代码中增加“assign load\_op=inst\_lw;”语句，修改的的波形如图3.5, ID模块里的ds\_to\_es\_bus信号中Z消失。

图片包含 电子产品

描述已自动生成

图3.5 ID模块里的ds\_to\_es\_bus信号中Z消失

（5）归纳总结（可选）

变量忘记赋值可能容易发生，因此我们需要通过仿真查看波形来确定自己的代码中的变量是否都准确进行了赋值。

4、错误4：模块对应接口连接错误

（1）错误现象

在trace对比中wb\_rf\_wdata出错。

（2）分析定位过程

仿真进行get\_trace比对，发现下述error:

图片包含 物体

描述已自动生成

图3.6 get\_trace比对，发现wb\_rf\_wdata错误

这意味着写回寄存器堆的值出现错误，通过查找ALU模块，如图3.7，发现alu\_src1和alu\_src2的值恒为一样，通过查找ALU的模块调用，发现EXE模块与ALU模块接口连接错误。出现了“.alu\_src1 (es\_alu\_src2 )”。

图片包含 人员, 室内, 物体

描述已自动生成

图3.7 ALU模块中alu\_src1和alu\_src2的值恒为一样

（3）错误原因

EXE模块与ALU模块接口连接错误。出现了“.alu\_src1 (es\_alu\_src2 )”，导致alu\_src1和alu\_src2的值恒为一样，最终使得wb\_rf\_wdata的值错误。

（4）修正效果

通过将EXE模块和ALU模块接口进行更改，改为“.alu\_src1 (es\_alu\_src1 )”，修改的的波形如图3.8, ALU模块中alu\_src1和alu\_src2的值不恒为一样，且为正确传递的值，此处error消失。



图3.8 ID ALU模块中alu\_src1和alu\_src2的值恒为一样

（5）归纳总结（可选）

模块间接口的连接也应该根据设计文档进行检查，以免出现连接错误。

5、错误5：组合环

（1）错误现象

运行仿真，发现波形停止在某一时刻，如图3.9。

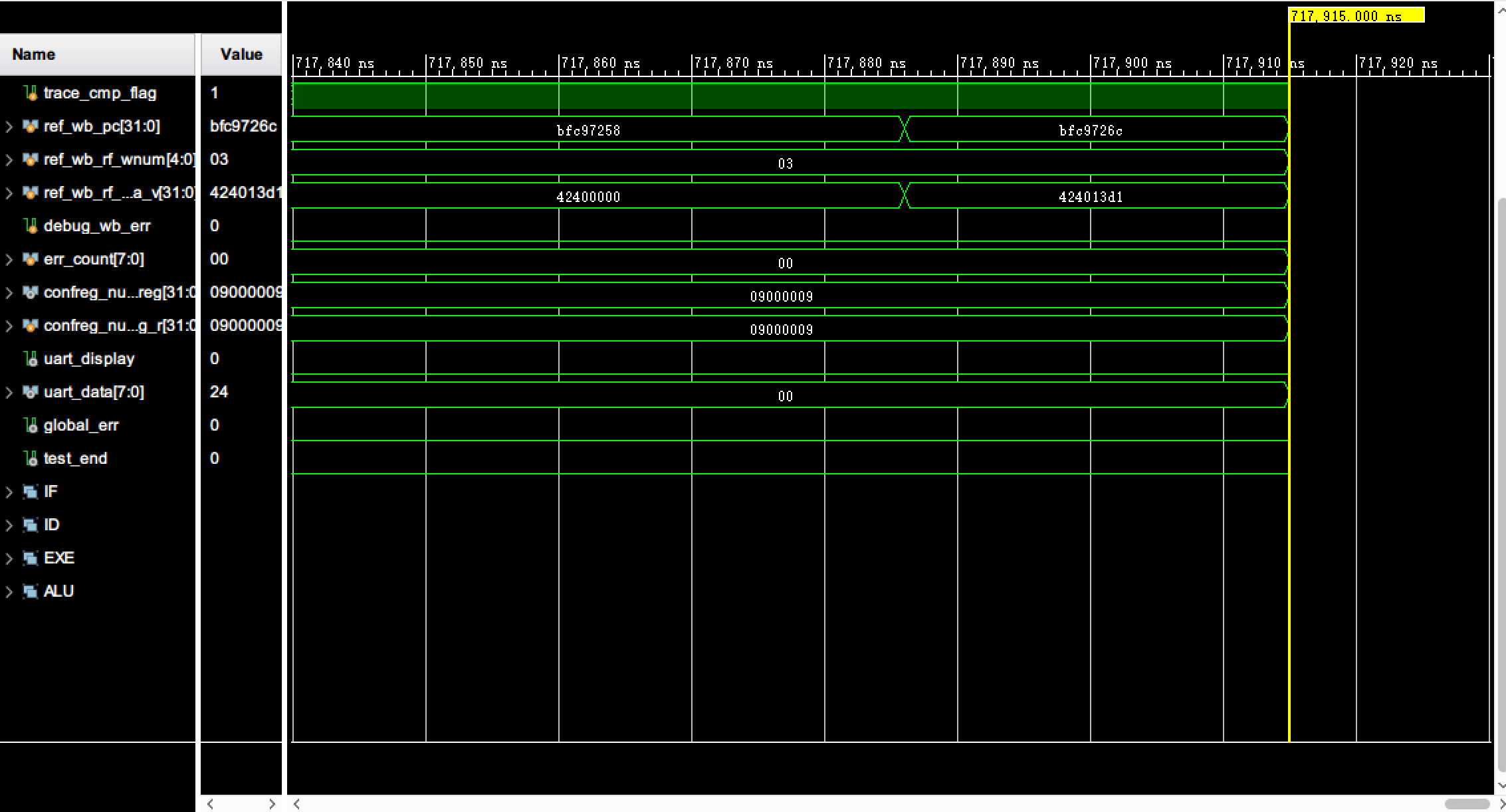


图3.9 仿真波形停止在某一时刻

（2）分析定位过程

经排查（vivado高亮指示）发现ALU模块的代码中出现了“assign or\_result = alu\_src1 | alu\_src2 | alu\_result;”而alu\_result中可以被or赋值，这出现了组合环。

（3）错误原因

ALU模块中出现了“assign or\_result = alu\_src1 | alu\_src2 | alu\_result;”而alu\_result中可以被or赋值，这形成了一个组合环。

（4）修正效果

通过分析代码逻辑，将原代码更改为“assign or\_result = alu\_src1 | alu\_src2 ;”，此error修复，进入到后续状态，但这时出现了新error，如图3.10。

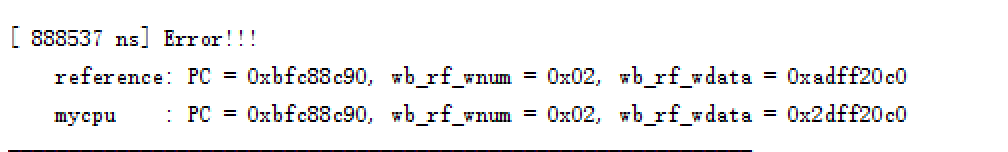


图3.10 修改为组合环后出现新error

6、错误6：ALU SRL移位错误

（1）错误现象

如图3.10，运行仿真，发现trace报错，wb\_rf\_wdata出现不匹配。

图3.9 仿真波形停止在某一时刻

（2）分析定位过程

通过查找对应PC的指令，如图3.11，发现这是一个srl指令，通过查找ALU中的srl发现，其srl的结果位宽只有31位，少了一位。

图片包含 文字

描述已自动生成

图3.11 对应PC的指令是srl

（3）错误原因

ALU模块中出现“assign sr\_result = sr64\_result[30:0];”，即输出srl指令的位宽只有31位，缺少了一位，导致最终wb\_rf\_wdata的值错误。

（4）修正效果

通过分析代码逻辑，将原代码更改为“assign sr\_result = sr64\_result[31:0];”此error修复，仿真运行显示全部 PASS，如图3.12。

图片包含 屏幕截图

描述已自动生成

图3.12 ALL PASS

（5）归纳总结（可选）

最后采用宏定义来定义位宽，同时要根据开发文档中的位宽来对代码中的宏定义进行二次确认。

7、错误7：越沿采样

（1）错误现象

仿真无错误现象，该错误对本次实验的结果显示无影响。

（2）分析定位过程

当上述问题被解决后，trace比对会全部通过，而此时还有1处bug未被发现。通过每个文件的一一查找，发现了此次实验的bug不止7处。第七处bug为always语句中采用了阻塞赋值。MEM模块中的always语句里出现了“es\_to\_ms\_bus\_r = es\_to\_ms\_bus;”

（3）错误原因

MEM模块中的always语句里出现了“es\_to\_ms\_bus\_r = es\_to\_ms\_bus;”。而always语句中应该采用非阻塞赋值。

（4）修正效果

通过分析代码逻辑，将原代码更改为“es\_to\_ms\_bus\_r <= es\_to\_ms\_bus;”，此错误修复。

（5）归纳总结（可选）

always里必须采用非阻塞赋值，这样的错误不应当出现。

8、错误8：信号“Z”

（1）错误现象

Tools模块里的decode出现错误，信号中出现“Z”，如图3.13。

图片包含 绿色

描述已自动生成

图3.13 decode的OUT信号最高位为’Z’

（2）分析定位过程

通过每个文件的一一查找，发现了此次实验的bug不止7处。第8处bug为for语句中的i的取值范围出现错误。导致上述信号最高位为Z，通过找tools模块，定位到64译码for循环的i只赋值到了第63位。

（3）错误原因

Tools模块中进行解码的for语句中的i的取值范围出现错误。代码中为“generate for (i=0; i<63; i=i+1)”，而上限应该是i=63。

（4）修正效果

通过分析代码逻辑，将原代码更改为“generate for (i=0; i<64; i=i+1)”，此错误修复。波形恢复正常，如图3.14。

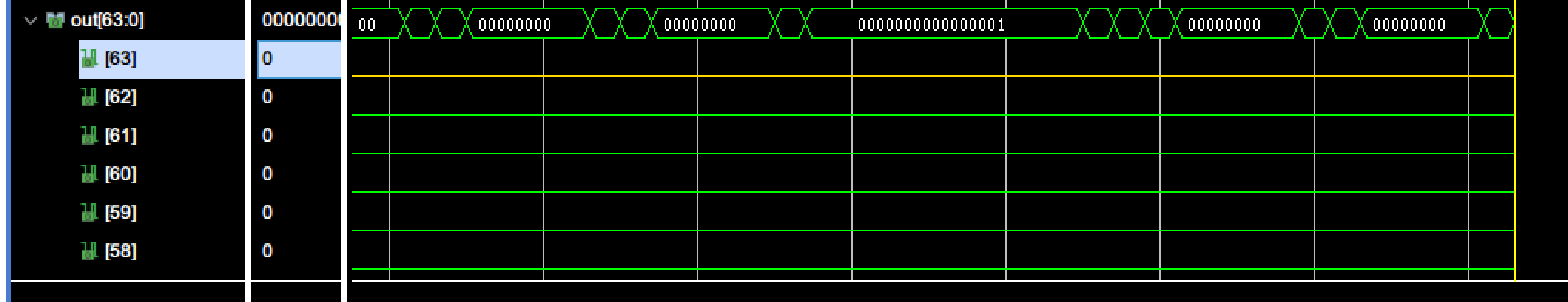


图3.14 decode的OUT信号最高位为恢复正常

（5）归纳总结（可选）

For循环的语句的i的范围的确容易出错，写之前应当三思。

四、实验总结（可选）

本次试验由于中间增加了中秋小长假，导致做实验的时间所有缩短，短时间内要阅读实验文档、完成实验、撰写实验报告是一件工作量不小的事。第一次在vivado软件内部运用trace进行比对，一开始有一些不适应，习惯在vivado里遇到问题就查波形。但实验做着做着就发现，比对工作和上个学期的组成原理实验的比对流程很类似，于是就迅速习惯了这种方式。但是实验中除了6处明显能通过波形和比对trace能找出的bug，另外两处bug都需要用细心一点去查看。实验指导里提到有7处bug，但最终找出了8处bug，不知道老师是刻意安排，还是也不小心笔误了。通过代码的调试，最终上板效果如下，表明调试成功。

图片包含 电子产品, 电路

描述已自动生成

图4.1 上板成功运行