**实验4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流水线冲突产生的原因、设计阻塞五级流水线CPU，并在上次给出的无阻塞CPU设计代码的基础上，判断译码阶段的寄存器读操作和后续进行的寄存器写操作是否有冲突，并以此作为修改ds\_ready\_go置“1”的条件，以实现阻塞的五级流水CPU。

二、实验设计（40%）

（一）总体设计思路

**硬件结构设计图如下：**

**图片包含 屏幕截图

描述已自动生成**

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。在上次框架设计的基础上，该框架增加了四个接口连线，分别是将es\_to\_ms\_valid、es\_to\_ms\_bus、ms\_to\_ws\_valid、ms\_to\_ws\_bus连接到了ID模块，其值将影响流水线阻塞的判断。

（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ID模块会利用EXE、MEM、WB传回的数据，比较三个后续状态里是否有与ID模块里寄存器读地址相同的寄存器，判断是否有“写后读”的行为，以此判断是否需要在该阶段进行阻塞。后阶段写寄存器的地址通过新接入的es\_to\_ms\_bus、ms\_to\_ws\_bus两条总线传输到ID模块（WB阶段的写寄存器地址通过ws\_to\_rf\_bus传递给ID模块）。

根据分析发现，对于发生阻塞的原因可以根据读写地址分成两大部分:寄存器堆读地址1和写地址冲突、读地址2和写地址冲突。同时，根据MIPS指令的特点发现，addu、subu、slt、sltu、and、or、xor、nor、bne、beq可能与两个源操作数有关。其中beq和bne虽然没有对应寄存器写操作，但在译码阶段，跳转指令涉及的寄存器读数时，如果不加阻塞，就会被前面其他指令还未写回的状态所干扰，从而读出错误的值。addiu、jr只与源操作数1有关，sll、srl、sra、sw只与源操作数2有关。

根据上述分析，该设计设置了wire类型三个变量hazard，sr1\_hazard，sr2\_hazard。其中sr1\_hazard，sr2\_hazard是分别判断译码阶段的寄存器读地址与EXE、MEM、WB阶段的寄存器地址是否有冲突；而hazard变量则是结合MIPS不同指令产生源操作数冲突的不同，判断是否会出现写后读现象。若hazard置为1，则ds\_ready\_go置为0；

在设置完hazard阻塞控制变量后，另外一个需要解决的问题是CPU完成阻塞后，如何进行恢复。解决这个问题主要有一下两种思路：

第一种思路是判断EXE、MEM和WB三个阶段模块内的PC值是否相等。若相等，则说明可以在时钟的下一拍将ID阶段的阻塞信号恢复。然而这一种做法虽然可以通过原有的两个新接入的bus获得EXE和MEM两阶段的PC值，但对于WB阶段的PC值需要增加ws\_to\_rf\_bus的位宽，将WB阶段的PC值通过ws\_to\_rf\_bus一同传递过来。

第二种思路是判断EXE、MEM和WB阶段的valid信号，而这三个信号的值分别由es\_to\_ms\_valid, ms\_to\_ws\_valid和ws\_to\_rf中新增的ws\_valid信号来反映。当流水线发生阻塞后，上述信号会依次从1变为0，而此处在es\_to\_ms\_valid, ms\_to\_ws\_valid和ws\_valid信号变为0进行阻塞复位,wire类型变量hazard\_reset置为1。当hazard\_reset置为1时，hazard信号置为0，即停止阻塞流水。

在上述两种方案的选择中，本设计选择了第二种方案，因为其增加ws\_to\_rf\_bus的位宽更少。

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 |

1. 功能描述

将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传递给EXE模块,EXE模块在下一时钟周期接受。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。此外，该模块会利用EXE、MEM、WB传回的数据，比较三个后续状态里是否有ID模块里寄存器读地址相同的寄存器，判断是否有“写后读”的行为，以此判断是否要在该阶段进行阻塞。

（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 |

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 |

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。

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21日 09：00-09：30 阅读讲义

9月21日09：30-11：30 进行五级阻塞流水线 CPU设计，调试bug

9月16日19：30-21：00撰写实验报告

以下设计阻塞信号时的错误记录

（二）错误记录

1、错误1：仿真停止在某一时刻

（1）错误现象

运行仿真，发现仿真运行到某一时刻不能继续进行，PC值一直保持不变，如图3.1.1。

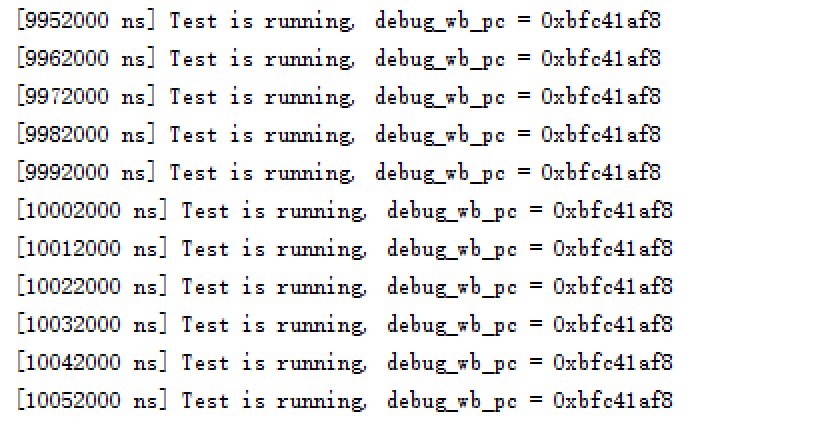


图3.1.1仿真停止

（2）分析定位过程

如图3.1.2，仿真时PC值一直保持不变，这说明流水线可能出现了一直阻塞的现象，对应查找hazard变量对应波形可以发现：hazard的值在发生阻塞后，恒为1。因此，此处需要设置一个hazard\_reset阻塞复位信号。

图片包含 物体, 室内

描述已自动生成

图3.1.2 发生阻塞后hazard值恒为1

（3）错误原因

hazard变量由于未设置阻塞复位信号，导致发生阻塞后，值恒为1。因此，此处需要设置一个hazard\_reset阻塞复位信号。

（4）修正效果

设置一个reg 类型的hazard\_reset阻塞复位信号，当时钟上升沿时，倘若(!es\_to\_ms\_valid&&!ms\_to\_bs\_valid&&!ws\_valid)为真，则hazard\_reset信号置为“1”，其他状态置为“0”。如此修改后，波形不再停止，但出现新的错误,如图3.1.3。

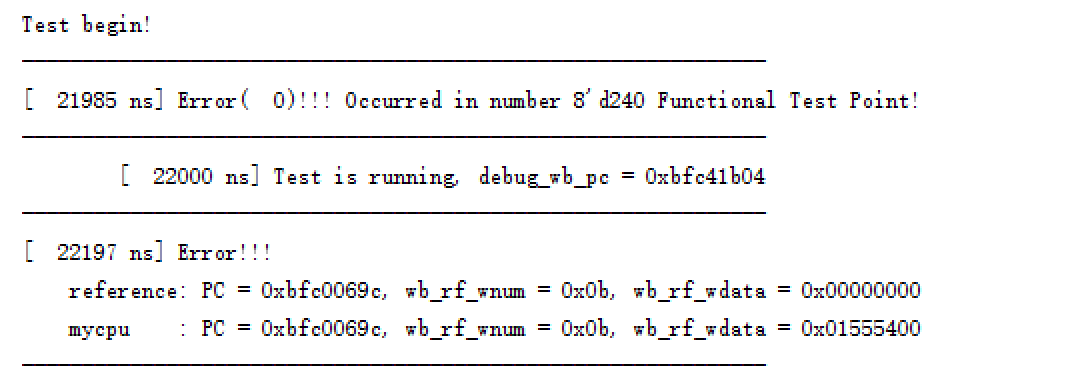


图3.1.3 波形不再停止，但出现新的错误

（5）归纳总结（可选）

在设置判断信号时，不仅要想其置为1的情况，也要考虑其置为0的情况。

2、错误2：数据错误

（1）错误现象

仿真时，trace反映wb\_rf\_wdata错误，如图3.1.3。

（2）分析定位过程

通过查看trace对比波形，如图3.2.1，发现在22,204,827ns时刻，实际数据和对比标准数据不同。通过查看此刻的valid波形和hazard\_reset波形发现，hazard\_reset在本应置为1的后一个时钟周期的上升沿才置为1，导致寄存器读数读了之后写入的数0x01555400而非0x00000000，如图3.2.2。

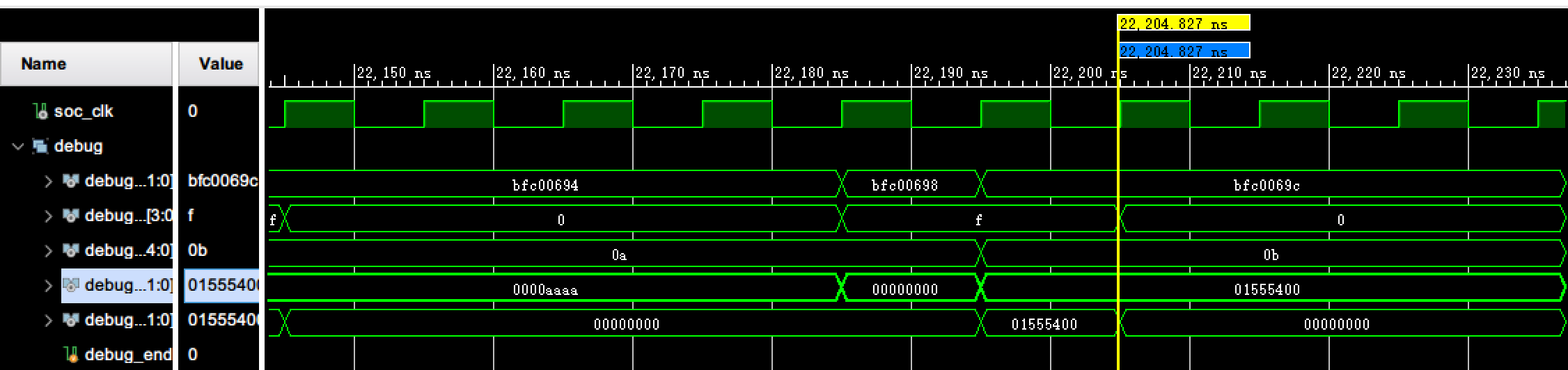


图3.2.1 debug模块波形对比

图片包含 绿色, 监视器, 屏幕, 电视

描述已自动生成

图3.2.2 debug模块波形对比

（3）错误原因

hazard\_reset在本应置为1的后一个时钟周期的上升沿才置为1，导致寄存器读数读了之后写入的数0x01555400而非0x00000000,因此，应该把hazard\_reset信号更改为wire类型的变量。

（4）修正效果

hazard\_reset信号更改为wire类型的变量后。通过assign hazard\_reset = (!es\_to\_ms\_valid&&!ms\_to\_bs\_valid&&!ws\_valid); 修改完毕后，该问题消失，全部测试点通过，如图3.2.3。

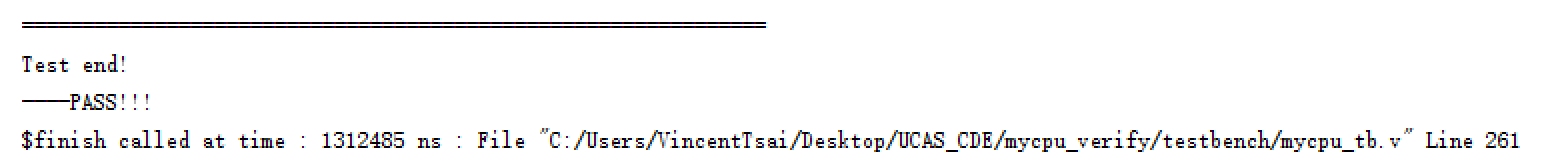


图3.2.3 所有测试点通过

（5）归纳总结（可选）

对于时序逻辑电路部分，要弄清处在时钟上升沿时不同变量之间的关系，要思考清楚变量到底应该采用同步逻辑还是异步逻辑。

四、实验总结（可选）

本次试验开始需要自己在已有代码的基础上设计一些信号传递方式和数据通路。这是第一次需要独立设计一个小功能实现方式的模式，相较于之前做过的实验，有很多自主性，同时，在无形中也增加了一些难度。自主设计意味着需要权衡利弊，例如本次试验关于阻塞信号复位的判断就想出了两种方案，再仔细思考过后，我发现两种方案看似不同，实际上有相同之处，而只增加一个ws\_valid信号的方案在数据传输带宽上要求更低，因此最终采用了这种方法。

这次实验虽然不要在老师写下的茫茫代码中去寻找几个bug，但自己设计的代码中隐藏的问题往往更加难以发现，需要我在熟悉整个电路时序和组合逻辑，以及各种数据之间的关联的情况下，才能找出这些隐蔽的问题。借助trace对比和代码不断地调试，最终上板效果如下，表明调试成功。

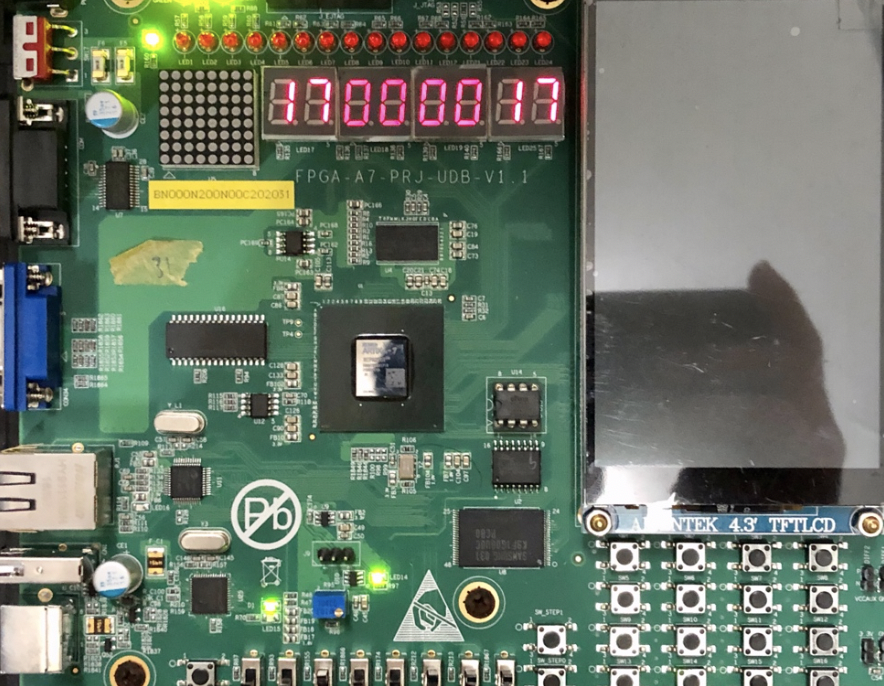


图4.1 上板成功运行