

计算机系统结构课程实验

总结报告

实验题目：简单的流水线CPU设计与性能分析

学号：2152809

姓名：曾崇然

指导教师：陆有军老师

日期：2023-11-26

1. 实验环境部署与硬件配置说明

1. 实验环境部署：windows11，使用vivado作为

开发工具，vivado自带的仿真工具进行仿真

2. 硬件配置说明：Xilinx FPGA器件：Nexys DDR开

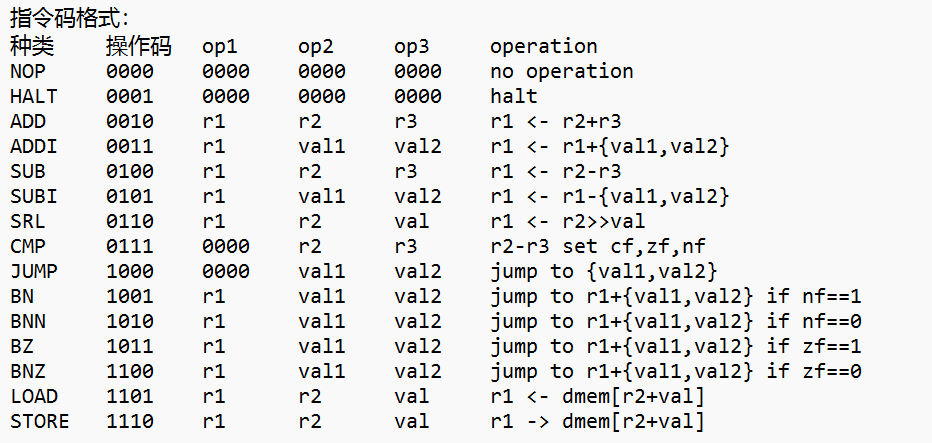
发板

1. 实验的总体结构
2. 指令选取：共选取常用指令15条，以完成指

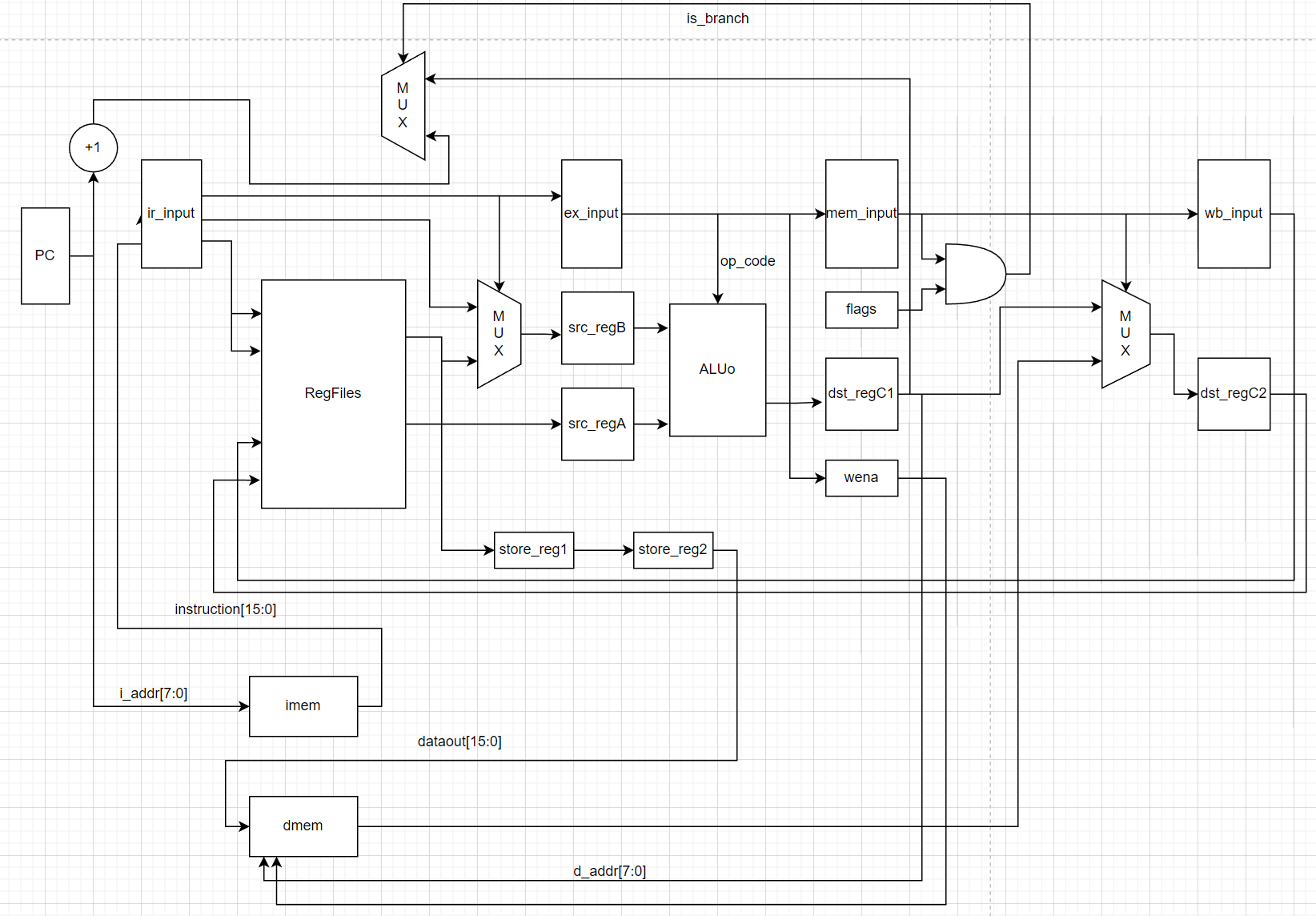
定的功能，依次为：NOP、HALT、ADD、ADDI、SUB、SUBI、SRL、CMP、JUMP、BN、BNN、BZ、BNZ、LOAD、STORE

1. 指令码设计，因为指令为15条，所以可用4

位表示操作码，每4位表示一个寄存器或者立即数，后8位可拼接起来表示一个立即数。具体设计如下，该CPU可执行15条指令，拥有16个通用寄存器



1. 静态流水线的总体结构



4、结构解释

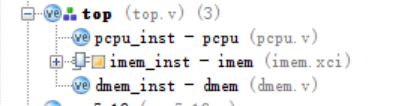
①总体采用哈佛结构，指令和数据分开存储，通过数据总线和指令总线和CPU进行数据交换

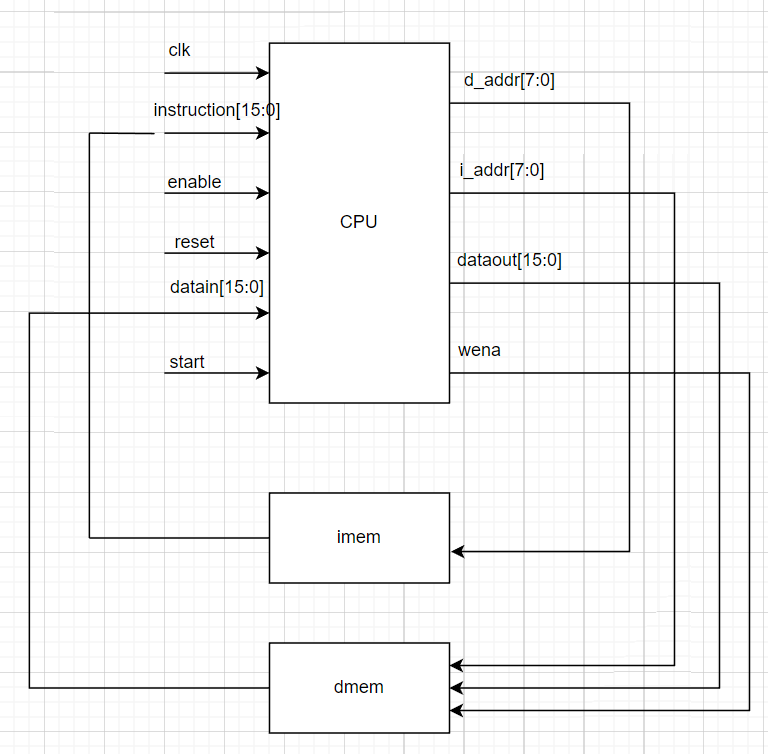
②一条指令的处理划分为IF、ID、EX、MEM、WB五个阶段，分别位取址、取值、计算、访存、写回，根据指令类型的不同在不同的阶段完成不同的操作。

③通过设置多个寄存器来实现指令的流水执行

1. 总体架构部件的解释说明
   1. 总体构架：主体分为三部分，CPU，imem（指

令寄存器），dmem（数据寄存器），imem和dmem通过指令总线和数据总线与CPU进行数据的交换，使用top.v文件将这三个部件进行连接





* 1. 静态流水线具体部件的解释说明

①PC寄存器：存放要读取的指令的地址，地址为

7位，即指令存储器的深度为256

②id\_input，ex\_input，mem\_input，wb\_input：存

放对应阶段流入的指令，每个阶段设置一个以保证指令流水执行

③RegFiles：通用寄存器堆，16个

④src\_regA，src\_regB：两个源操作数寄存器，在ID阶段根据指令类型的不同将这两个寄存器进行赋值

⑤store\_reg1，store\_reg2：用于STORE指令的两个寄存器，使用两个以保证两个STORE指令可以连续执行（流水）

⑥ALU：运算部件，在EX阶段进行计算，并置标志位，能进行加法，减法，移位等操作

⑦dst\_regC1，dst\_regC2：两个结果寄存器，分别在MEM和WB阶段将结果传出，设置两个是为了保证指令的流水执行

⑧wena：写信号寄存器

⑨flag：各种标志位寄存器，如nf，zf，cf

⑩imem，dmem：指令存储器和数据存储器，指令存储器位ROM，数据存储器为RAM，在使能信号和写入信号有效的上升沿写入，指令和数据的宽度均为16位，深度位256。

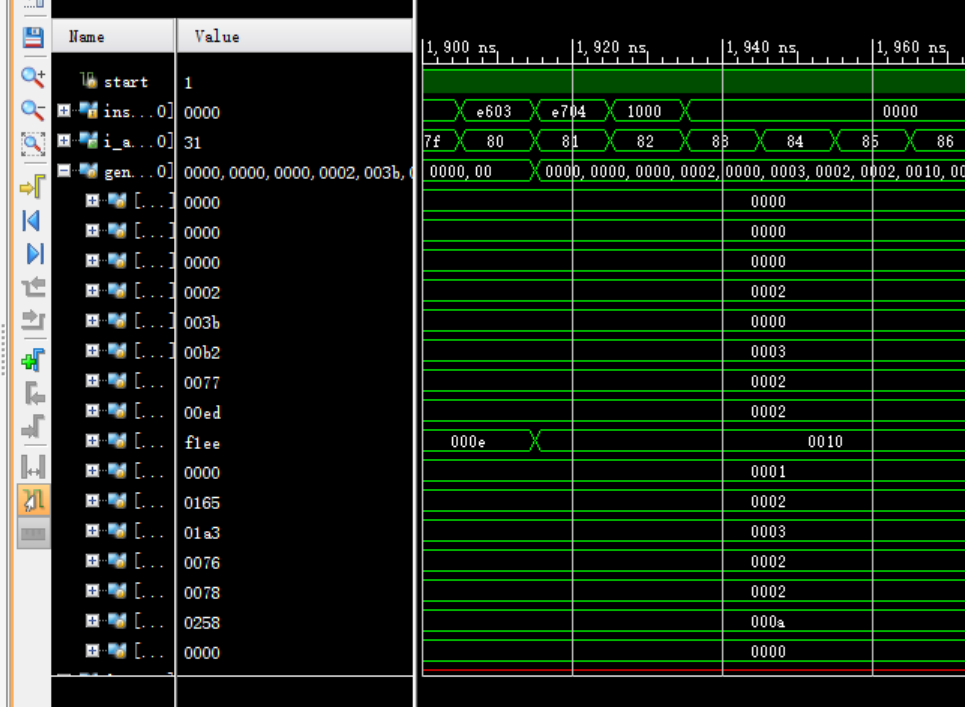
1. 实验仿真过程
   1. 编写用于仿真测试的C程序
   2. 将C程序转化为对应的指令序列，保证指令在

实现的指令范围内，同时使用NOP指令和调整指令顺序的方法解决冲突

* 1. 将指令序列转化为机器码，形成coe文件
  2. 使用该coe文件初始化imem
  3. 编写test\_bench文件进行测试仿真

1. 实验仿真的波形图及某时刻寄存器值的物理意义
   1. 静态流水线的波形图（未展示全部）：以下波形

包含start（开始信号），instruction（指令），i\_addr（指令地址），general\_reg（通用寄存器堆）



* 1. 寄存器值的物理意义

①cpu\_state、next\_cpu\_state：CPU当前和下一个

状态寄存器

②id\_input、ex\_input、mem\_input、wb\_input，存

储正在某个阶段执行的对应的指令

③src\_regA，src\_regB两个源操作数寄存器

④dst\_regC1，dst\_regC2两个目标寄存器，两个保证流水线不断

⑤store\_reg1，store\_reg2用于store指令的两个寄存器，两个保证流水线不断

⑥general\_reg[15:0]通用寄存器堆

⑦zf，nf，cf运算的标志位

⑧ALU\_result运算器的运算结果

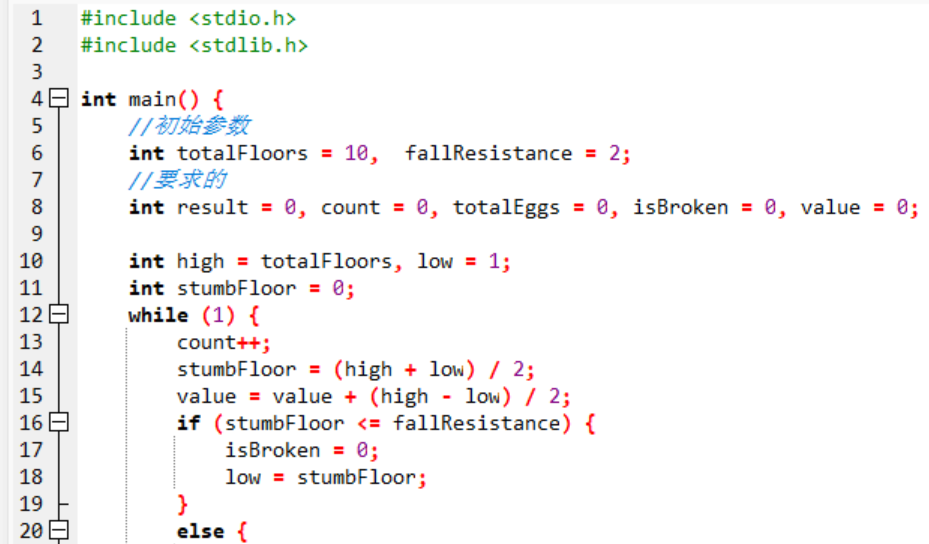
⑨cf\_buf存储溢出位

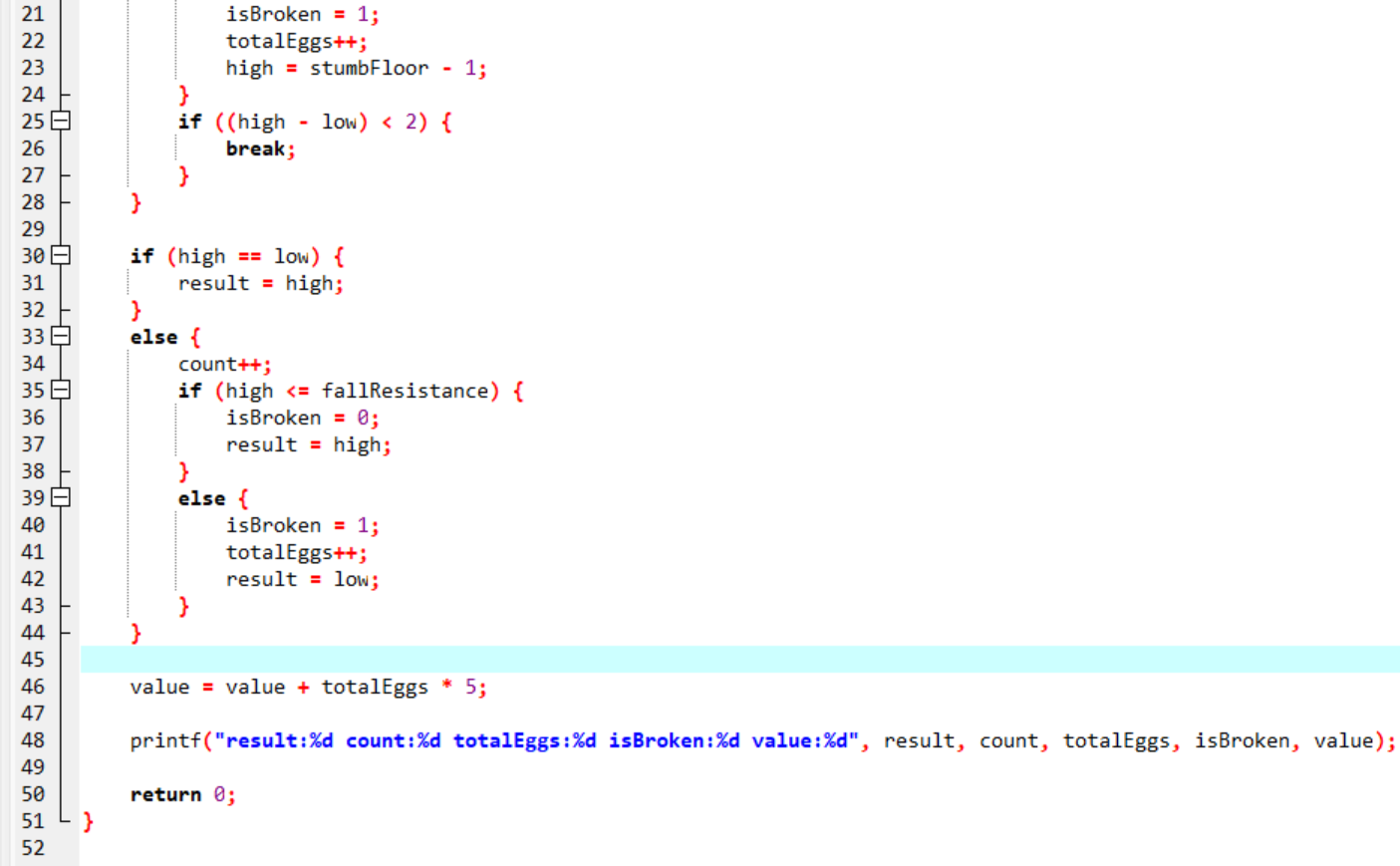
1. 流水线CPU实验性能验证模型

实验性能验证模型：比萨塔摔鸡蛋游戏。两个同学在可变换层数的比萨塔上摔鸡蛋，一个同学秘密设定同一批鸡蛋耐摔值；另一个同学在指定层高的比萨塔拿着鸡蛋往下摔，用最少的摔次数和摔破的鸡蛋数求出鸡蛋的耐摔值。假定在耐摔值的楼层及其下面楼层，鸡蛋摔不破，可以重复使用，否则鸡蛋摔破。要求模型的算法输出包括：摔的总次数、摔的总鸡蛋数、最后摔的鸡蛋是否摔破。请使用C语言设计该验证模型的算法，并把C语言汇编为RISC-V指令汇编程序，同时利用编译器生成RISC-V指令集可执行目标程序。

1、 编写C语言程序

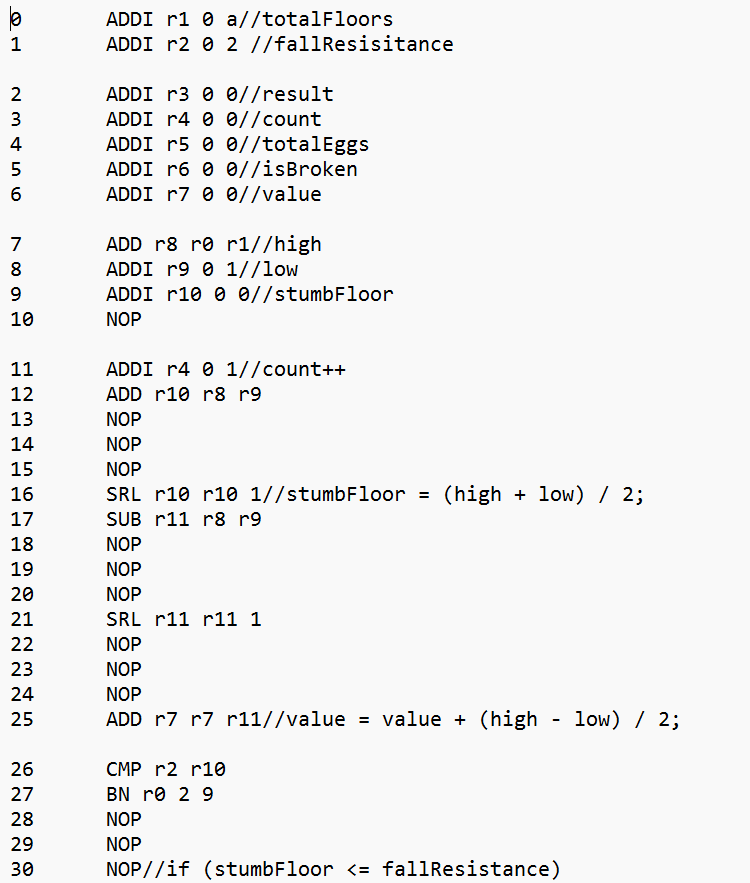
采用二分的方法进行耐摔值的查找，评估函数为爬的楼层数+5\*摔碎鸡蛋数





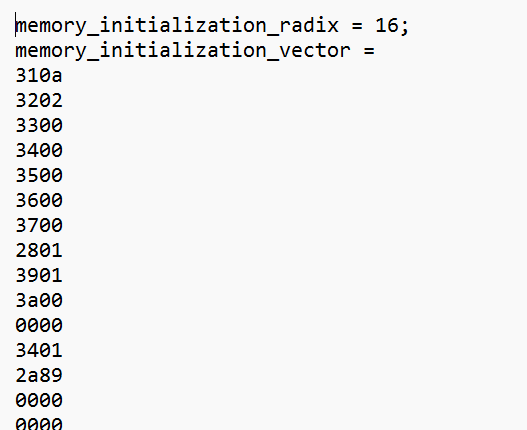
2、 编写对应的指令序列（汇编语言）

因为指令码的格式是自己确定的且只有15条，所以只能手动编译，手动调整指令顺序和添加NOP指令以解决冲突（部分截图如下）：



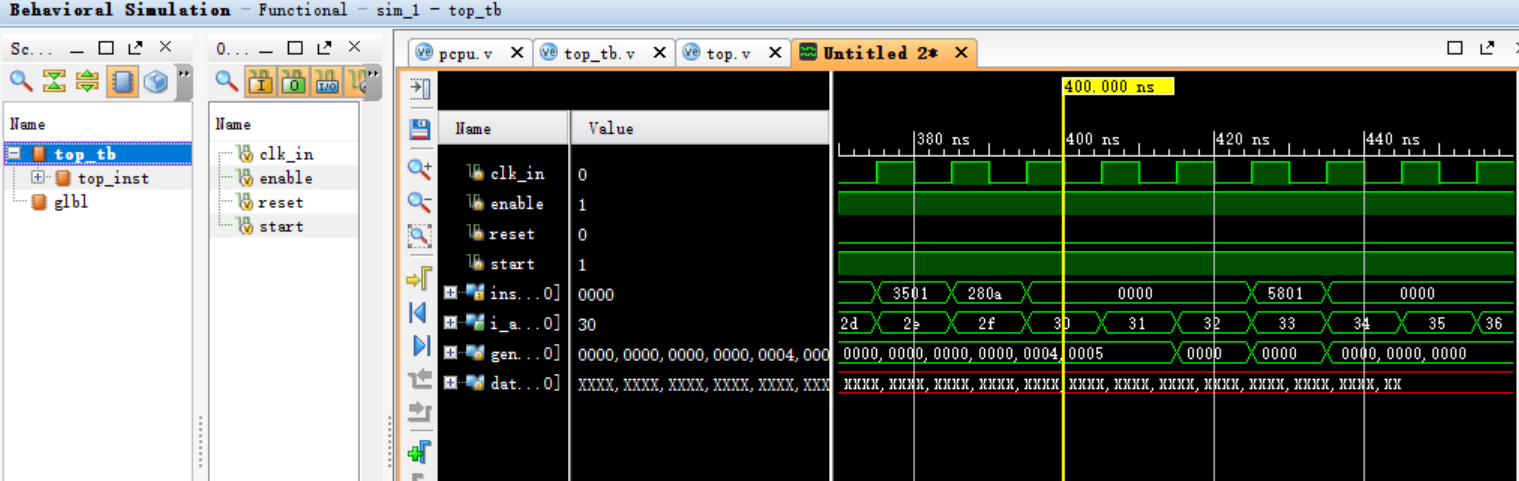
3、 将指令序列转化为机器码

编写了一个C++程序将指令转为机器码，部分截图如下：



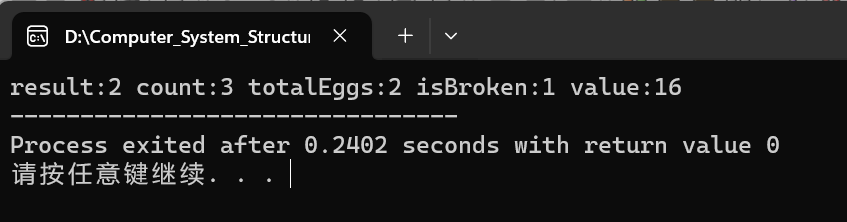
4、 将coe文件导入，并进行仿真

dmem初始值未知，所以为xxxxxxxx

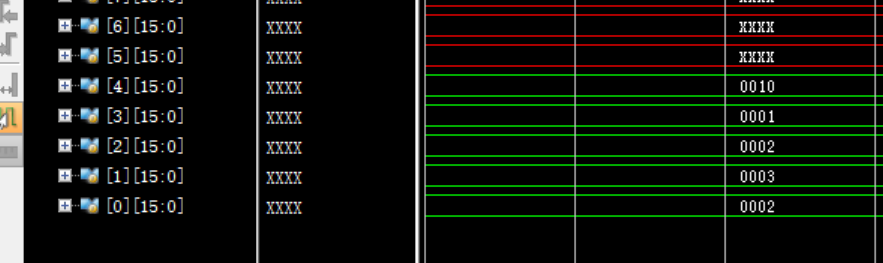


5、结果比较

C语言程序结果：



流水线CPU计算结果：

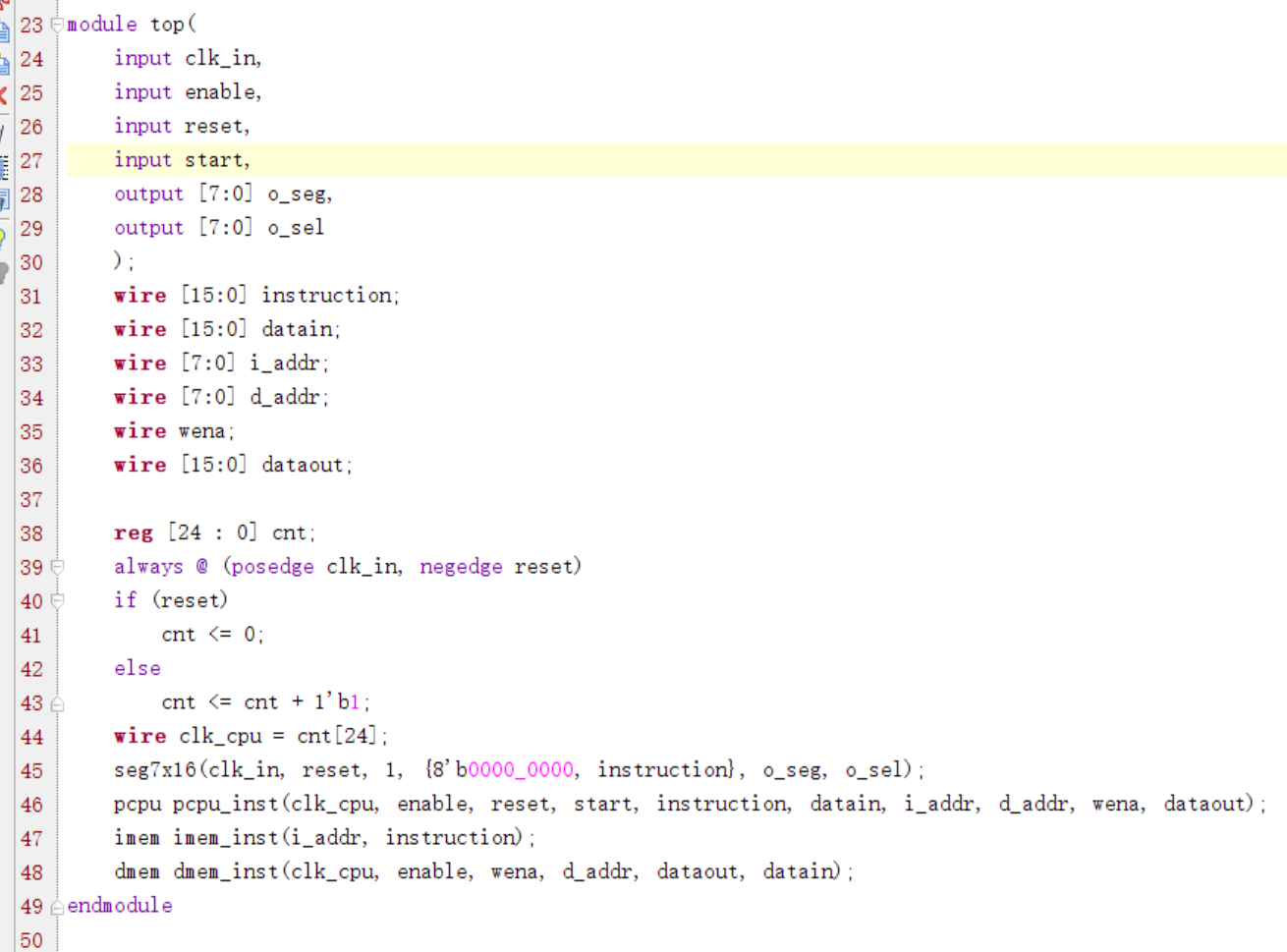


可以看出数据存储器中存储了对应值的计算结果，测试出的耐摔值为2，摔的次数为3，总共摔碎的鸡蛋为2，最后一个鸡蛋摔碎了，评估函数的值为10（16进制），与C语言程序计算的结果一致，CPU和汇编语言正确。

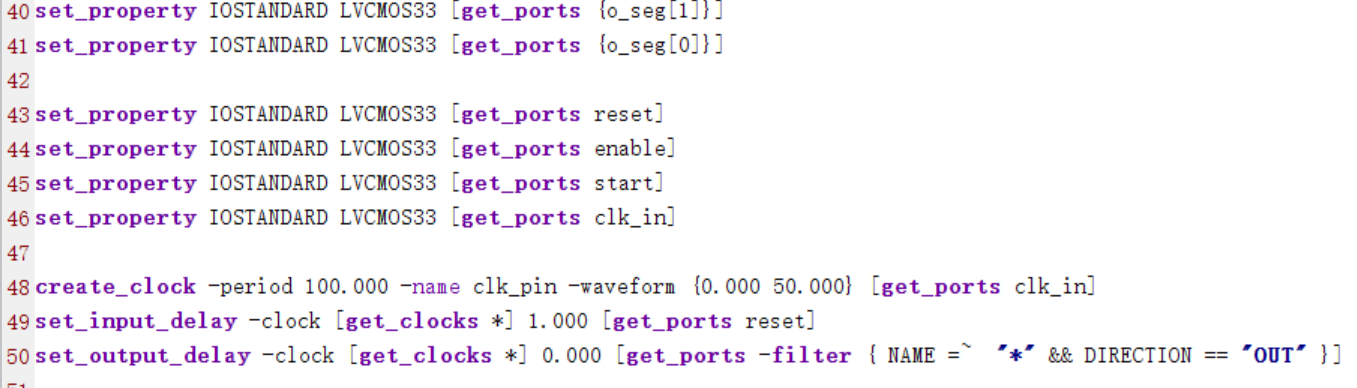
1. 实验验算程序下板测试过程与实现

1、重新配置顶层文件，加入分频和七段数码管以

方便观察实验现象，如下：



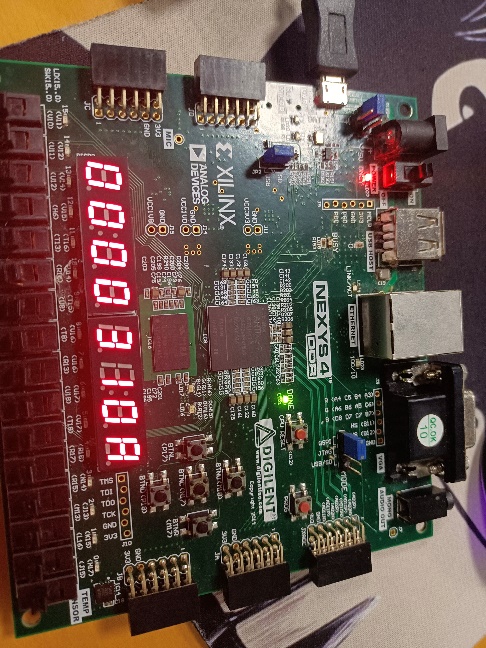
2、配置xdc文件，部分截图如下：



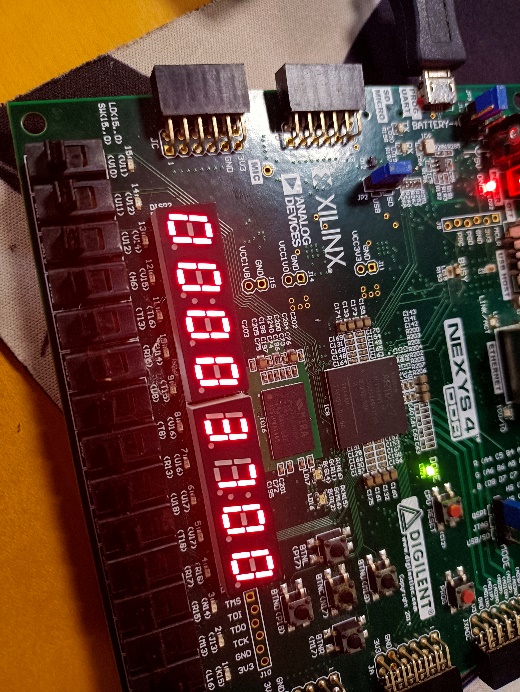
3、综合、布线、下板：

实验结果如下图：

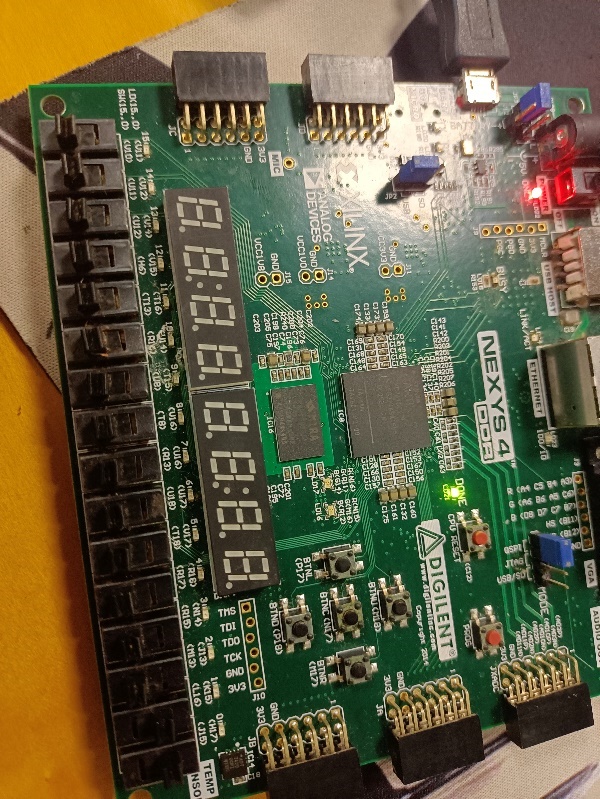
成功显示第一条指令：



开启enable和start能够正常流动：



关闭enable能停止流动，reset能复位



1. 流水线的性能指标定性分析（包括：吞吐率、加速比、效率及相关与冲突分析、CPU的运行时间及存储器空间的使用）

通过阅读时序报告可知延迟大概在90ns左右，因此时钟周期最短可设置在90ns附近。

* 1. 吞吐率：最大：1/t=11111111，实际：10893246

2、 加速比：n\*m/(n+m-1) ，n趋于无穷大时为5，

在测试程序中约为4.901

* 1. 效率：n\*m/(m\*(m+n-1))，n趋于无穷大时为1，

在测试程序中约为0.9804

* 1. 冲突分析

在遇到先写后读冲突时，可以使用NOP指令进

行缓冲或者调整指令执行顺序来解决冲突；在执行跳转指令时，由于跳转地址和跳转标志位的产生延迟的缘故会引发冲突，不能通过调整顺序解决，只能使用NOP进行缓冲。因此在程序执行的过程中需要大量的NOP指令，CPU空转，很大程度上影响了CPU的工作效率。

* 1. CPU运行时间：测试程序大约执行了几百条指

令，运行时间在20000ns左右

* 1. 存储器空间的使用：测试程序只在最后STORE

了5个值进数据存储器，其余计算均在寄存器，因此数据存储器空间消耗为5，指令存储器消耗空间为指令的数量为130。

1. 总结与体会

在本次实验中，我选择并设计了15条指令，并实

现了能够执行这15条指令的五段流水线CPU。然后编写了对应的摔鸡蛋的验证程序，将其转化为汇编和机器码置于CPU上进行运算，验证其正确性以及性能。

在这个过程中，我理解和掌握了指令设计的基本

方法和原理；掌握了简单的流水线CPU的设计和实现方法，明白其每个阶段的作用和意义；在编写汇编语言并转为机器码的过程中，我了解了通过NOP和调整指令顺序来解决流水线当中的冲突的方法。

这次实验使我对指令的设计，流水线CPU的设计

和性能评估，以及指令冲突的解决都有了更加深刻的认识。

1. 附件（所有程序）

cpu代码：cpu的代码置于代码文件夹中的cpu源

码文件夹中，其中包含.v文件，ip核，xdc配置文件，tb测试文件（使用tb需要注释掉分频和七段数码管相关内容，并改变对应的时钟），以及时序报告。

验证文件：验证相关文件置于代码文件夹中的验证

文件夹中，stumbEggs.c是验证的C程序，stumbEggsComplication.txt是对应的汇编语言，stumbEggsComplicationClear.txt是去掉注释和空行的汇编语言，stumbEggsMachine.coe是根据汇编语言转化的机器码，stumbEggsMachineLook.coe是每行加了序号的机器码，stumbEggsMachine.coe用于初始化imem。TranslateToMachine.cpp是一个能够将stumbEggsComplicationClear.txt汇编代码自动转化为stumbEggsMachine.coe机器语言代码初始化文件和stumbEggsMachineLook.coe文件的程序。