

**本科实验报告**

|  |  |
| --- | --- |
| 课程名称： | 计算机组成 |
| 姓 名： | 应周骏 |
| 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： | 3200103894 |
| 指导教师： | 马德 |

2022 年 5月 14 日

**浙江大学实验报告**

课程名称： 计算机组成 实验类型： 综合

实验项目名称： CPU设计

学生姓名： 应周骏 专业：计算机科学与技术 学号：3200103894

同组学生姓名： 无 指导老师： 马德

实验地点： 东4-509实验日期：2022 年 3 月 28 日

**实验0：**

**一、实验目的和要求**

1. 复习寄存器传输控制技术；

2. 掌握CPU的核心组成：数据通路与控制器；

3. 设计数据通路的功能部件；

4. 进一步了解计算机系统的基本结构；

5. 熟练掌握IP核的使用方法；

**二、实验内容和原理**

**目标：**

熟悉二进制原码补码的概念，了解二进制加减乘除的原理，掌握浮点加法的操作实现 。

**内容：**

1. 设计实现乘法器；

2. 设计实现除法器；

3. 设计实现浮点加法器；

**原理：**

**1. 设计实现乘法器**

因为是两个N位数相乘，所以结果应该是四个数加和得到的。先判断y的最低位是0还是1，如果是1，则需要把x加到部分积上，若为0，则需要把0加到部分积上（实际上加0的这个过程计算机并不执行，因为加0对部分积没有任何影响），x左移一位，之后再让y右移一位，若y为0，则循环结束，否则继续此循环过程，一共循环N次。流程图和具体架构如下。

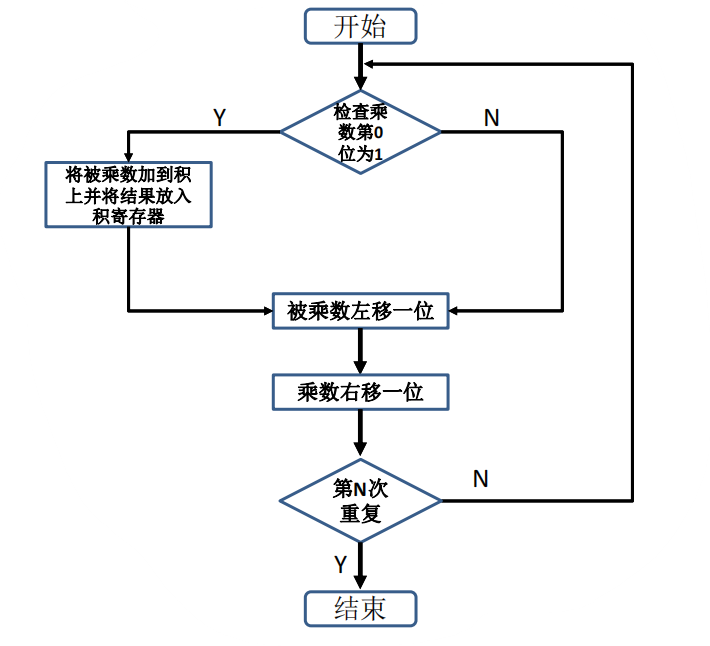


图 2.1.1 N位乘法流程图

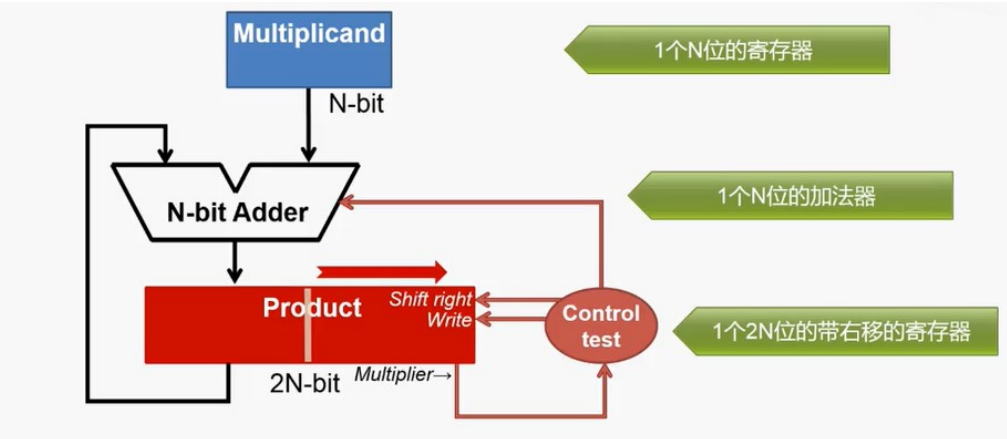


图2.1.2 N位乘法器设计架构

**2. 除法器的实现**

被除数x为1001010，除数y为1000，下面的除法过程是手工运算的一个步骤，而计算机在做除法时就是模拟手工运算的执行过程，具体计算机流程图如下。

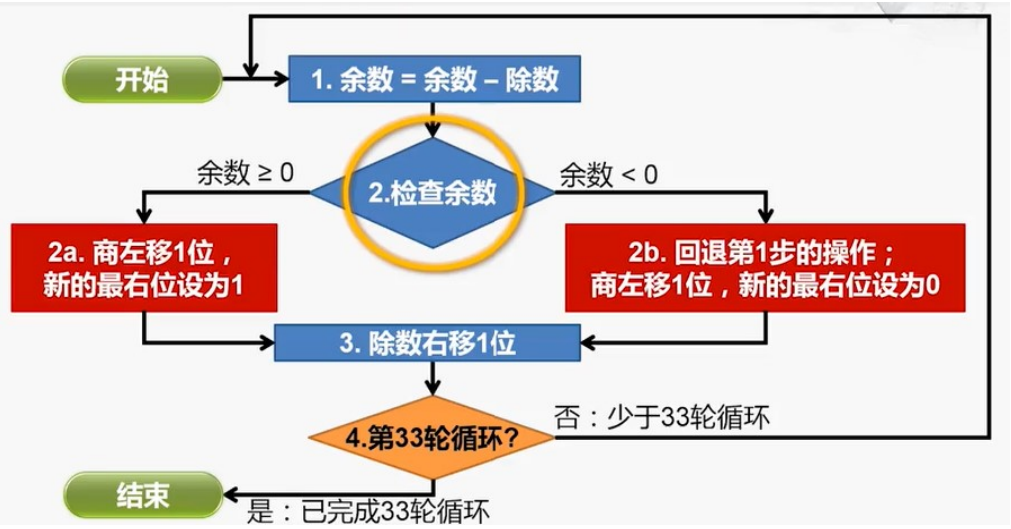


图2.2.1 32位除法流程图

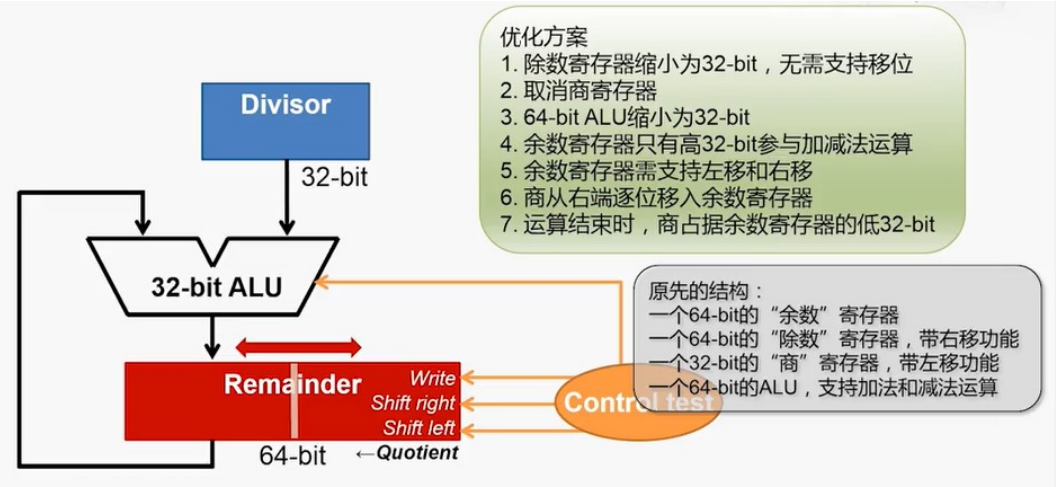


图2.2.2 32位除法流程图

**3. 浮点数加法**

完成浮点数加减运算的操作过程大体分为：

1. 0操作数的检查；

2. 比较阶码大小并完成对阶；

3. 尾数进行加或减运算；

4. 结果规格化；

5. 舍入处理；

6. 溢出处理；

在舍入处理中，本次实验我采用的是0舍1入。

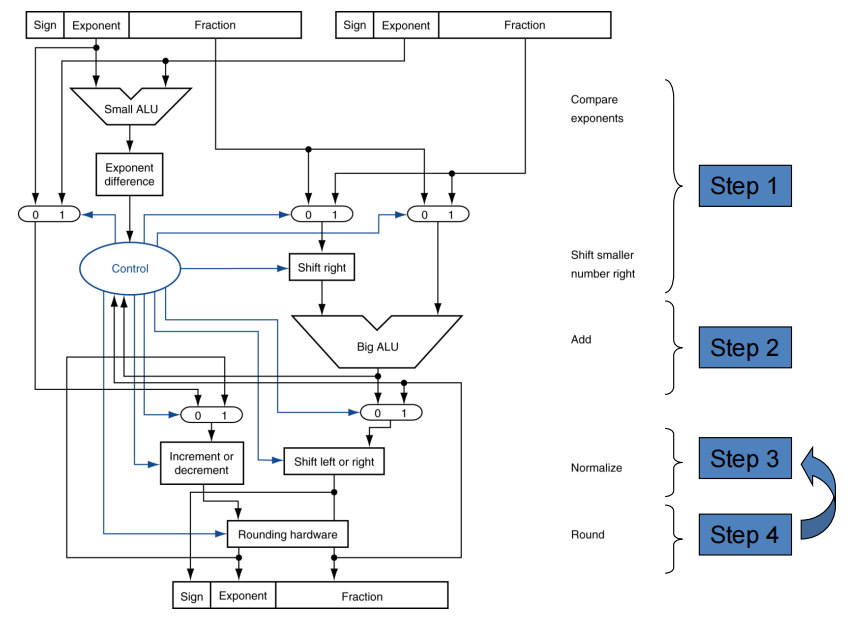


图2.3.1 浮点数加减法硬件实现

**三、实验过程和数据记录**

**实验一：乘法器设计**

**1. 工程文件建立**

新建工程文件，命名为“OxExp03\_mul32”。

**2. 乘法模块设计**

新建Verilog文件“mul32. v”，输入以下代码。

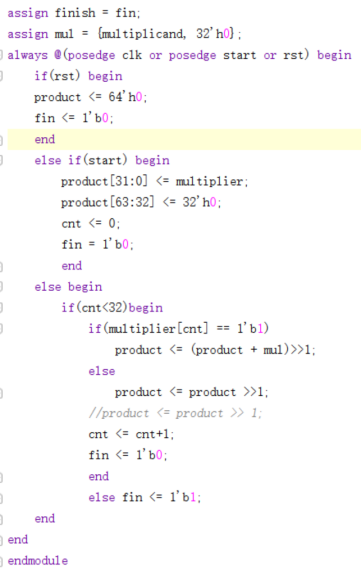
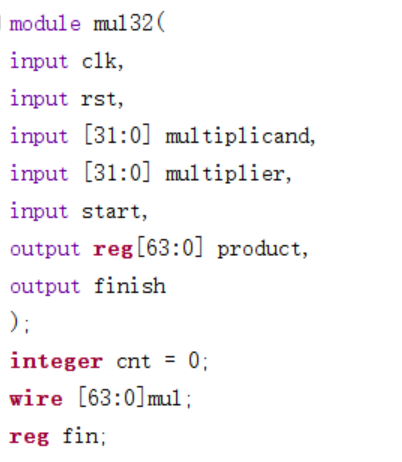


图3.1.1 乘法器结构化描述（详见附件）

**3. 仿真**

对乘法器进行仿真，输入仿真代码如下。

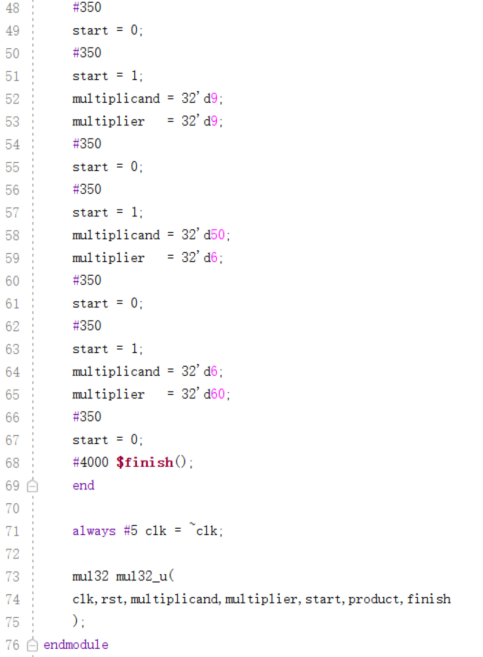
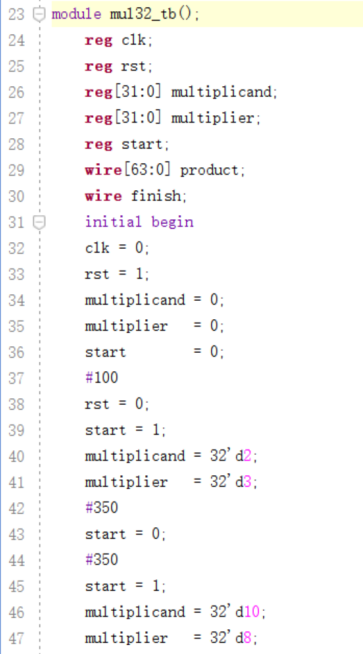
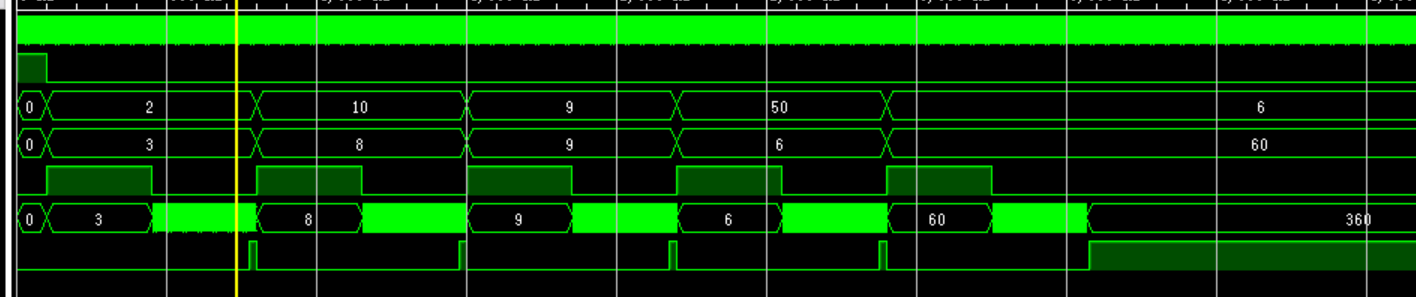
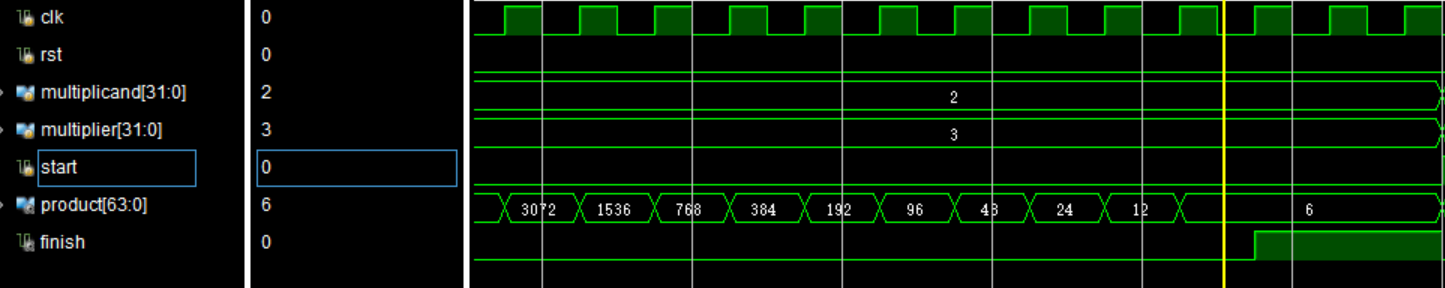
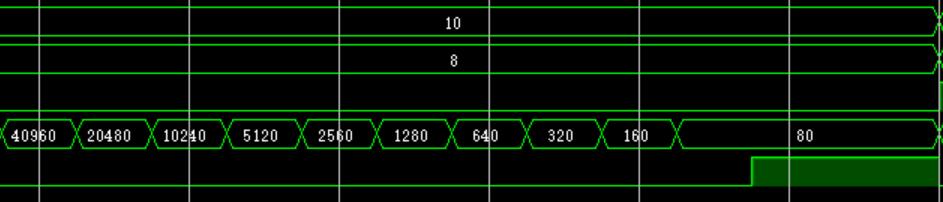


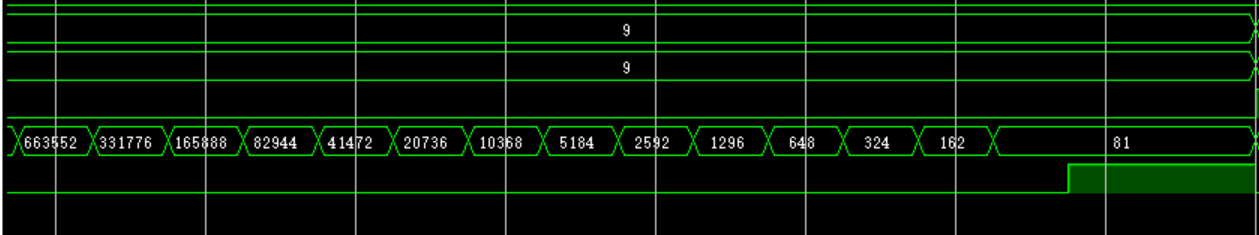
图3.1.2 乘法器仿真代码

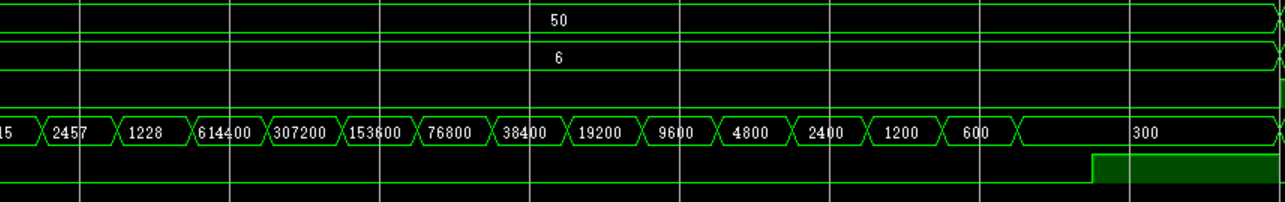
得到仿真结果图如下：











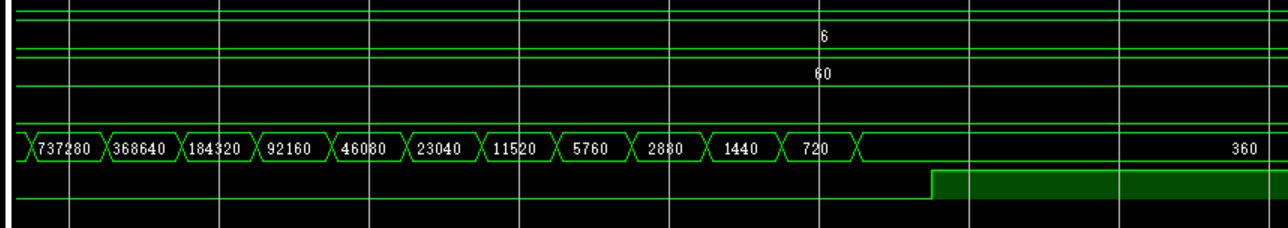


图3.1.3 乘法器仿真结果图

**实验二：除法器设计**

**1.工程文件建立**

新建工程文件“OxExp03\_div32”。

**2. 除法器模块**

新建Verilog文件“div32.v”，输入如下代码。

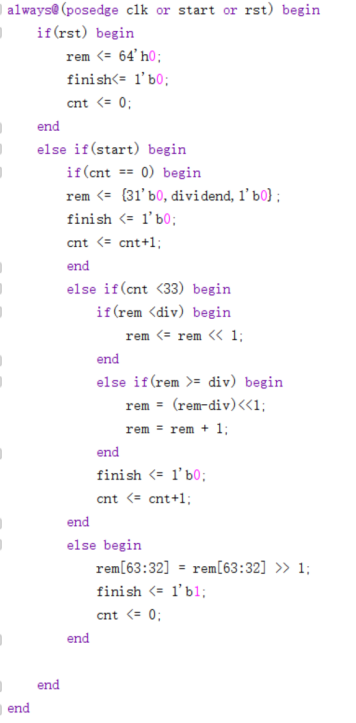
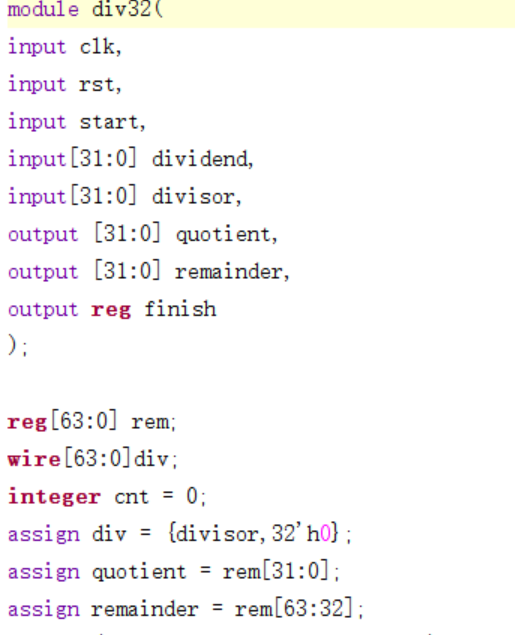


图3.2.1 除法器代码

对该模块进行仿真，输入如下仿真代码。

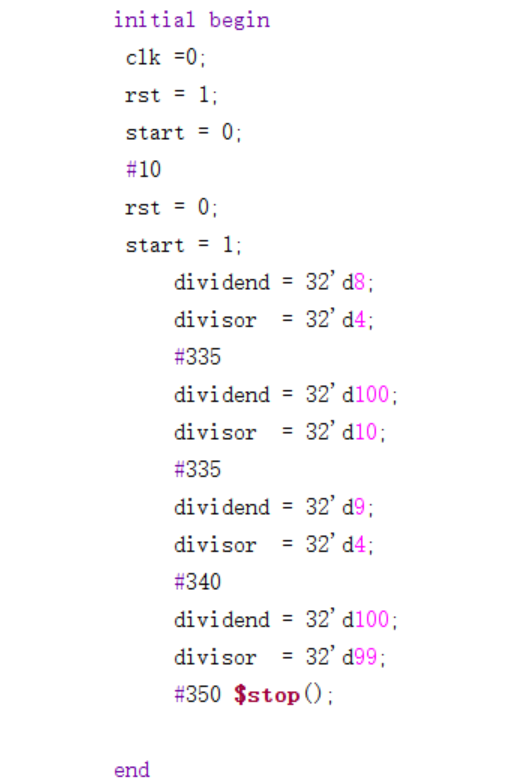
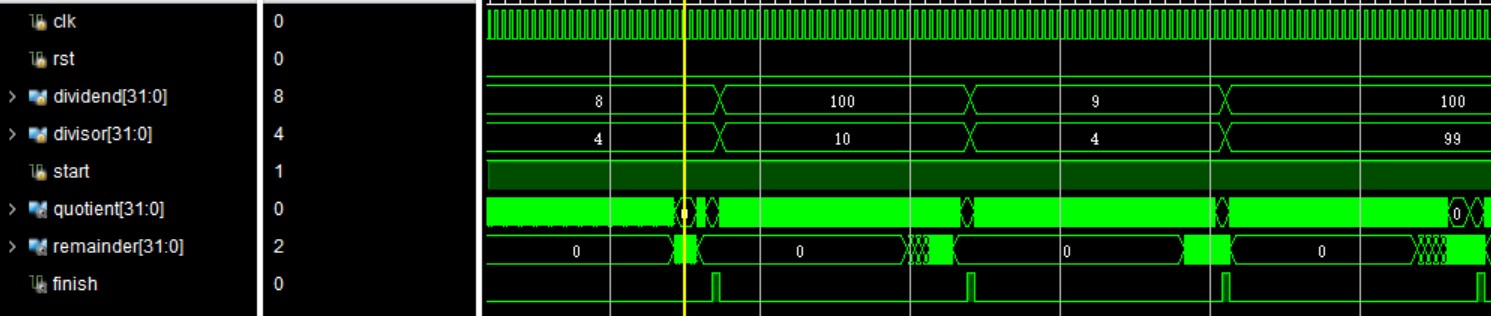
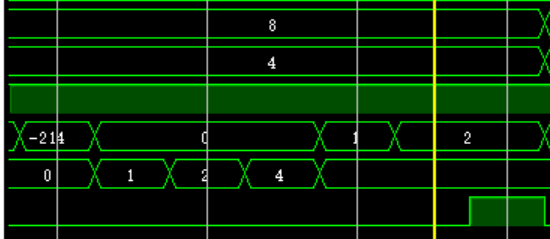
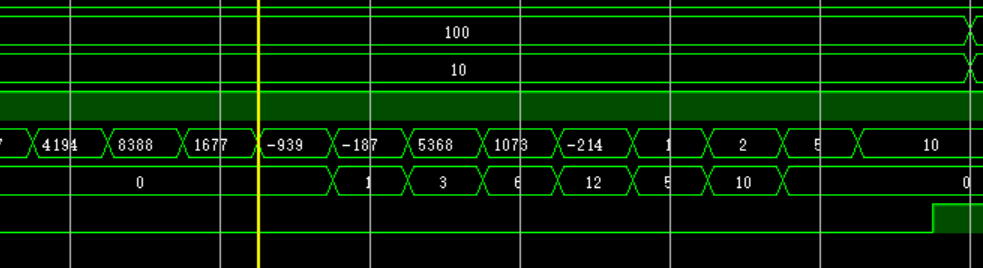


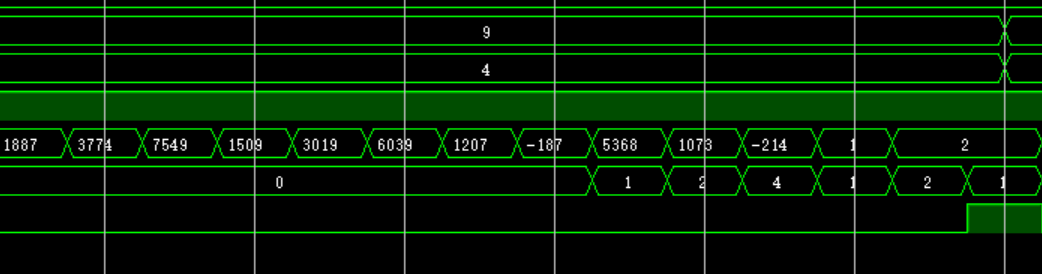
图3.2.2 除法器仿真代码

得到仿真结果如下。









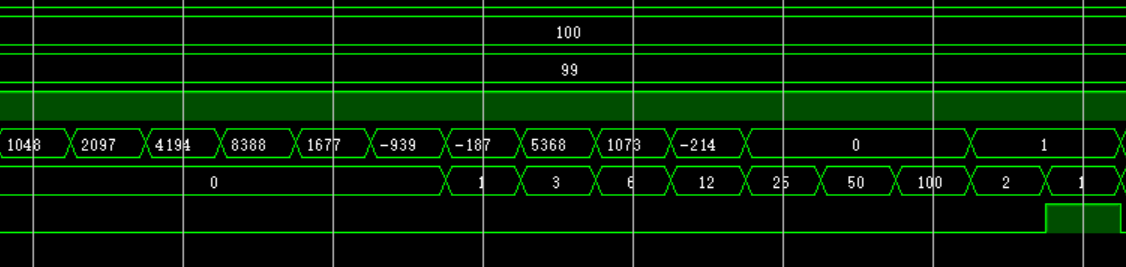


图3.2.3 除法器仿真结果图

**实验三：浮点数加减法**

**1. 工程文件建立**

新建工程文件，命名“OxExp03\_floatadd”。

**2. 状态机设计**

新建Verilog文件“float\_add32.v”，输入代码(由于过长，附在附件中，此处截取部分，后面代码分析进一步展开）。

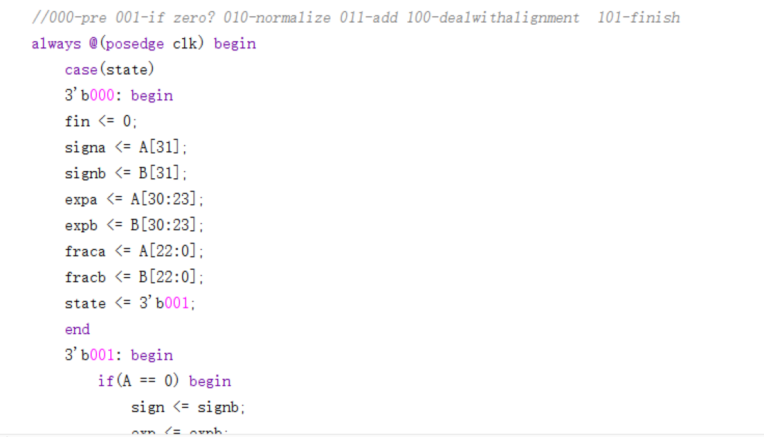


图3.3.1 浮点数加法器代码

对该原理图进行仿真，输入如下仿真代码。

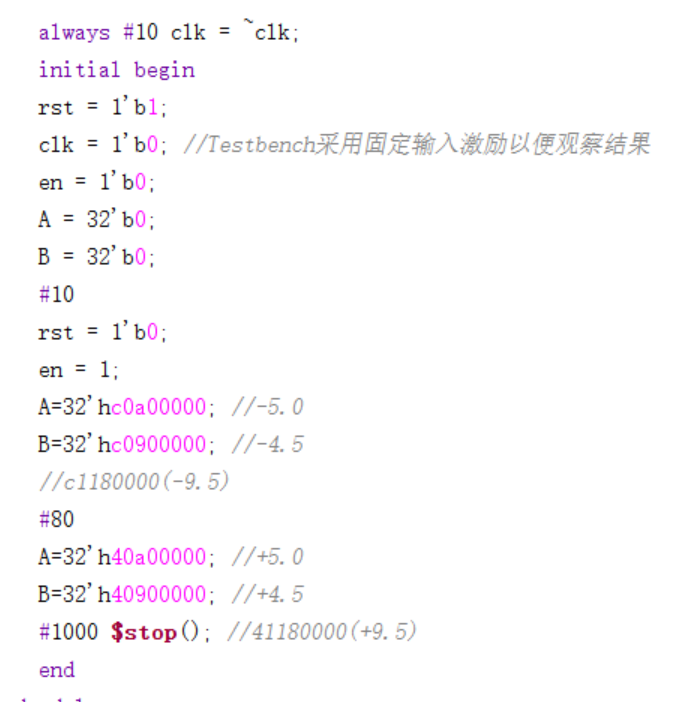


图3.3.2 浮点数加法仿真代码

得到仿真结果。

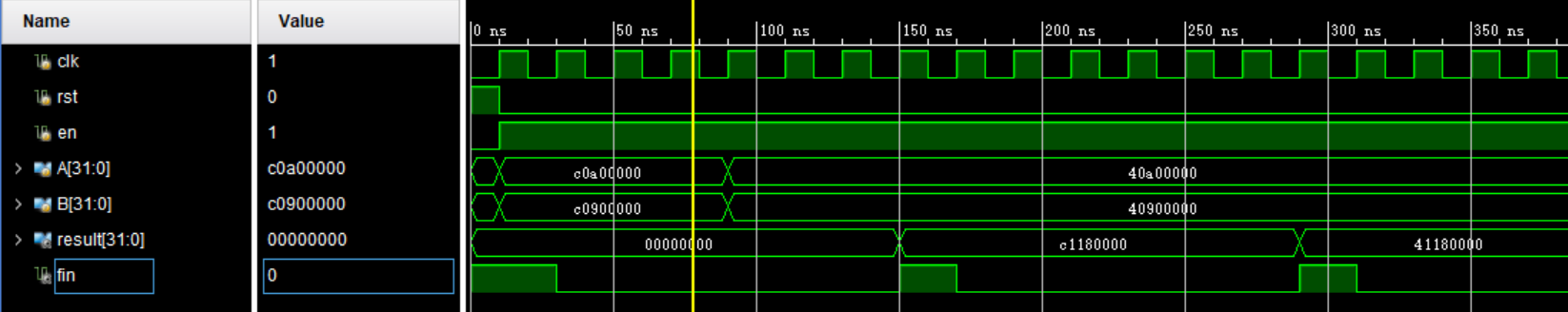


图3.3.3 浮点数加法仿真结果图

**四、实验结果分析**

**实验一：乘法器设计**

**1. 乘法器代码分析**

通过下述代码，对复位信号和开始信号做出响应，为后续算数实现做好准备。

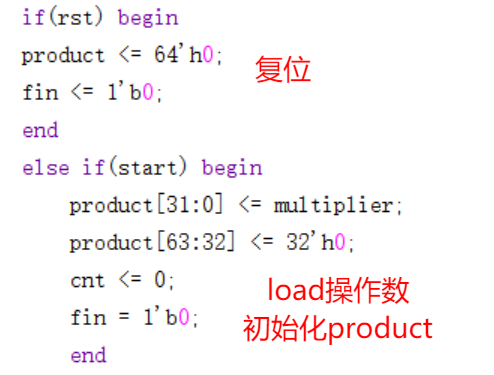
****

图4.1.1 乘法器初始部分

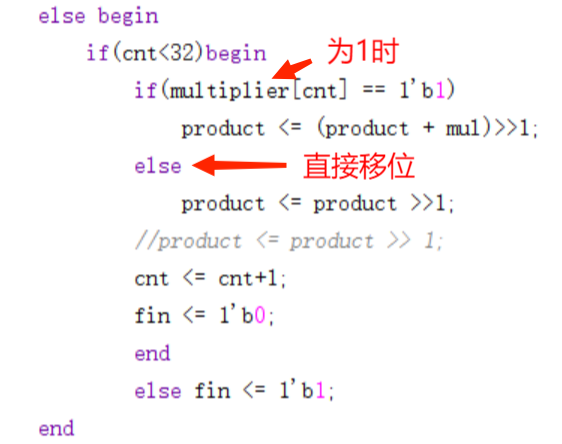


图4.1.2 乘法算术操作模拟

上述代码模拟实现了乘法操作，具体过程与乘法的流程对应。

**2.乘法器仿真结果分析**

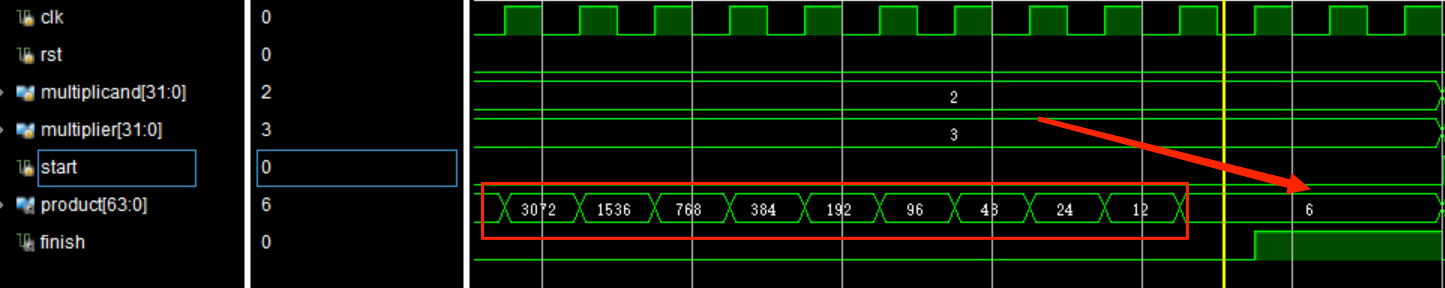


图4.1.3 乘法器仿真结果分析

在仿真时，能够显示乘法器工作流程，并且结果也和预期相同，此处仅列出一组，其余不再赘述。

**实验二：除法器设计**

**1. 代码分析**

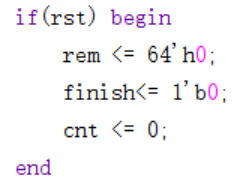


图4.2.1 除法器复位

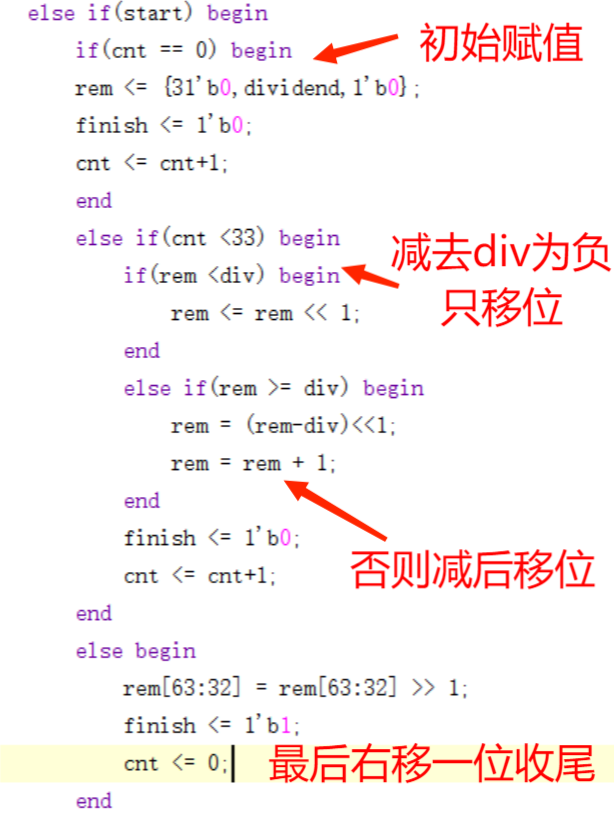


图4.2.2 除法器操作流程

按照除法器流程图，编写如上代码，可以实现预期功能。

**2. 仿真结果分析**

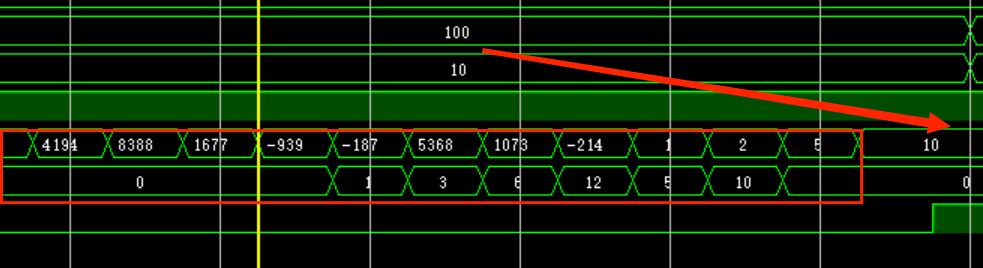


图4.2.3 除尽

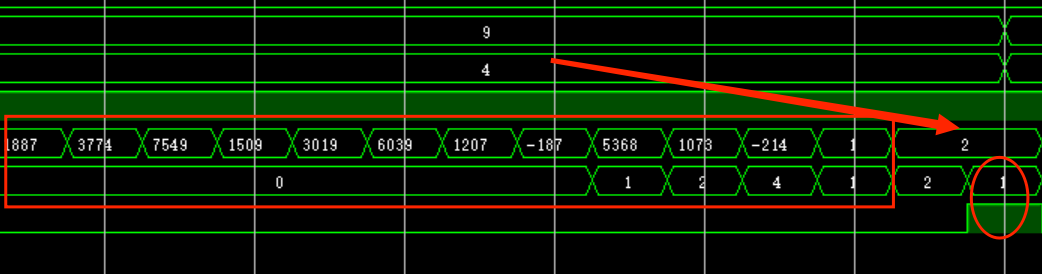


图4.2.4 除不尽有余数

可以显示除法器运行细节，且能够正确输出除尽和除不尽有余数情况的结果，符合实验预期。

**实验三：浮点数加法器设计**

**1. 浮点数加法状态机及代码分析**

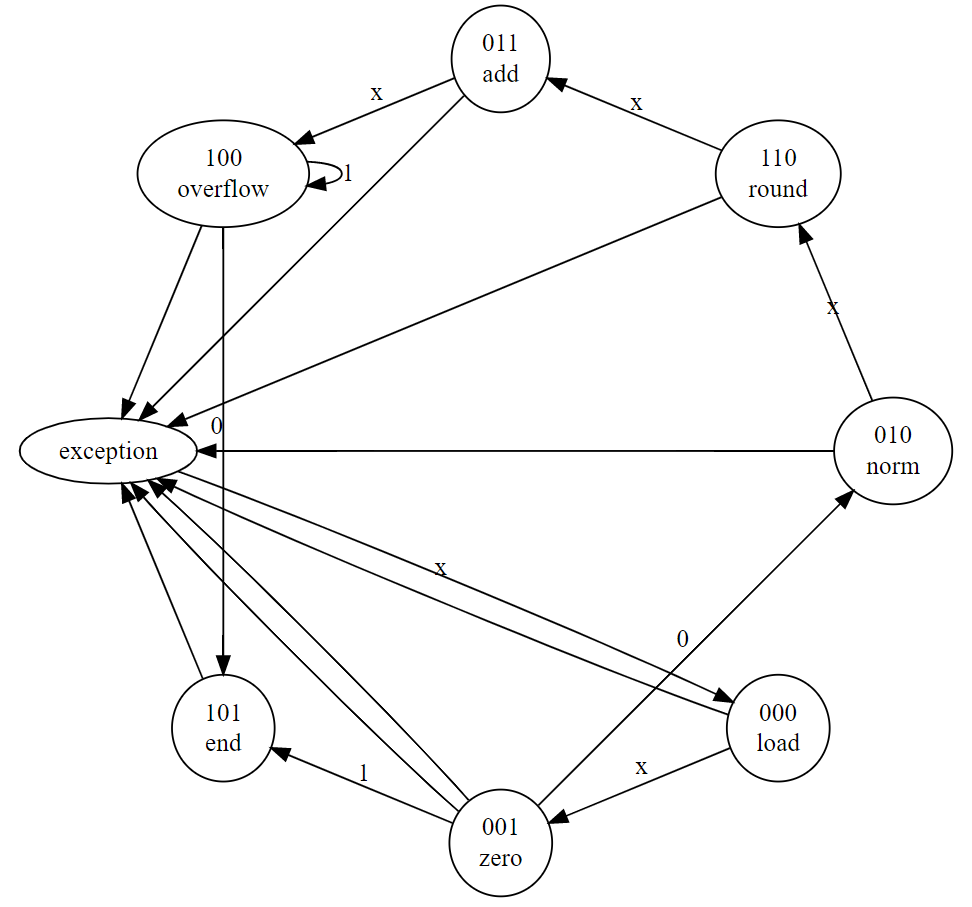
****

图4.3.1 浮点数加法器状态机

依据设计的状态机，编写相应的verilog代码。状态1编码为000，功能为将操作数进行分解，暂时存储在寄存器中。

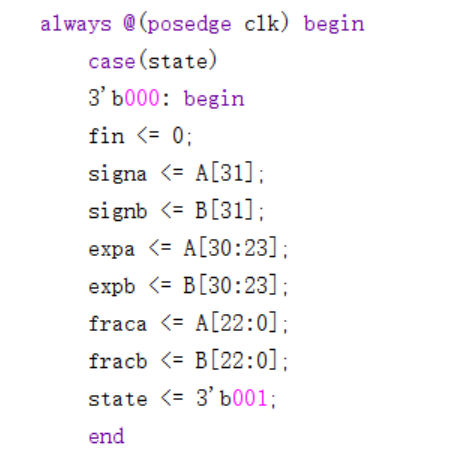


图4.3.2 状态1加载和分解操作数

状态2编码为001，用于判断操作数中是否有0，如果有0则直接转到结束，节省时间和资源。

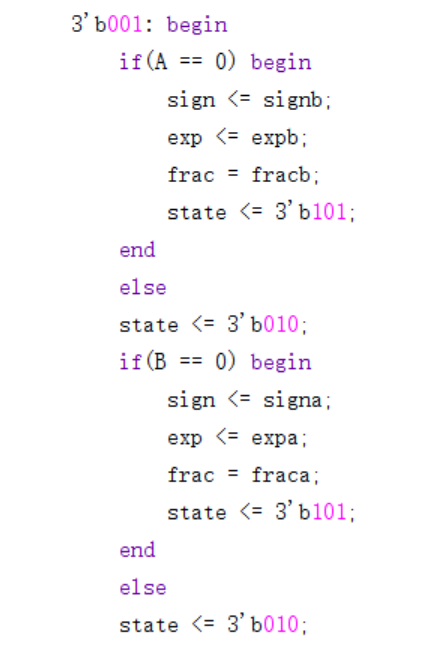


图4.3.3 状态2 0操作数判定

状态3编码为010，用于判断两个操作数是同号还是异号，并且比较指数大小，确定对齐位数和结果的符号。

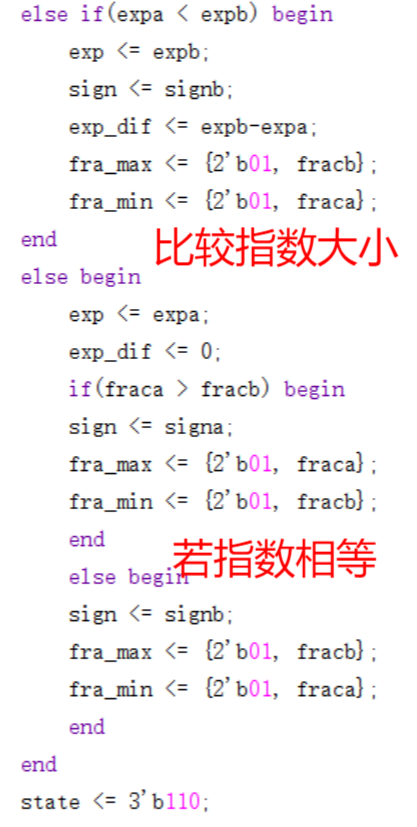
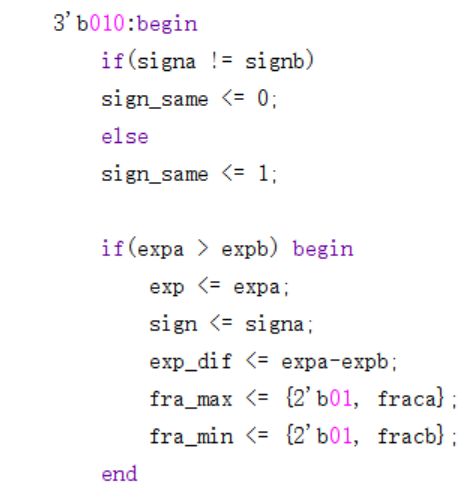


图4.3.4 状态3 同号判断和对齐

状态4编码110，暂时存储舍去部分的最高位，并进行对齐操作。

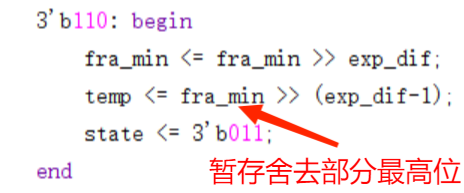


图4.3.5 状态4 舍入位暂存

状态5编码011，用于判断是否需要舍入，并完成浮点数加减。

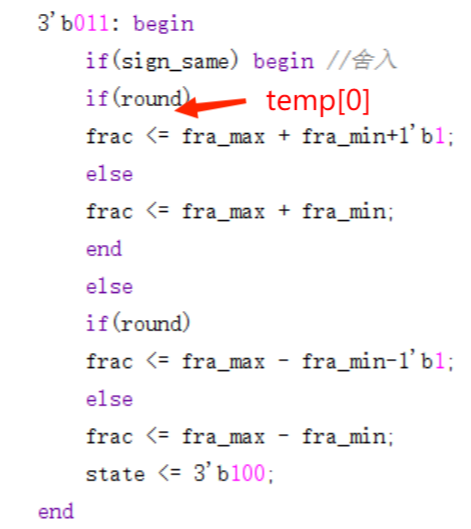


图4.3.6 状态5 舍入确认和加法

状态6编码100，用于溢出判断，若上溢，则对指数进行加一调整；下溢，则持续该状态直到移位到符合要求的指数。

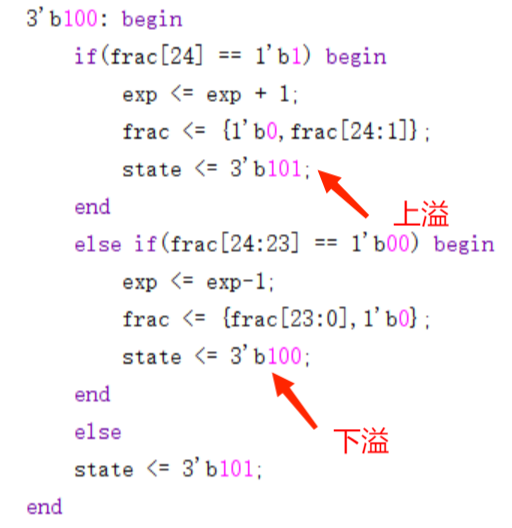


图4.3.7 状态6 溢出判断

状态7编码101，结束后返回状态1，并设置fin信号。

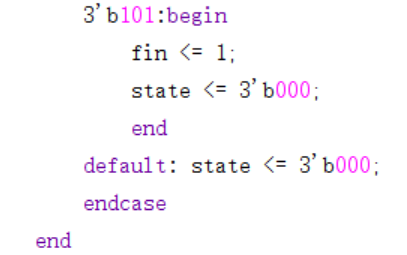


图4.3.8 状态7 结束判断

下面一段代码对fin信号进行响应，设置结果。

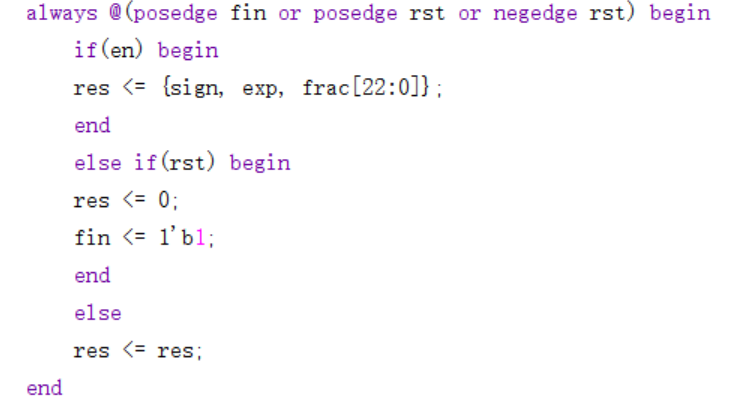


图4.2.9 结果存储

**2. 浮点数加法仿真分析**

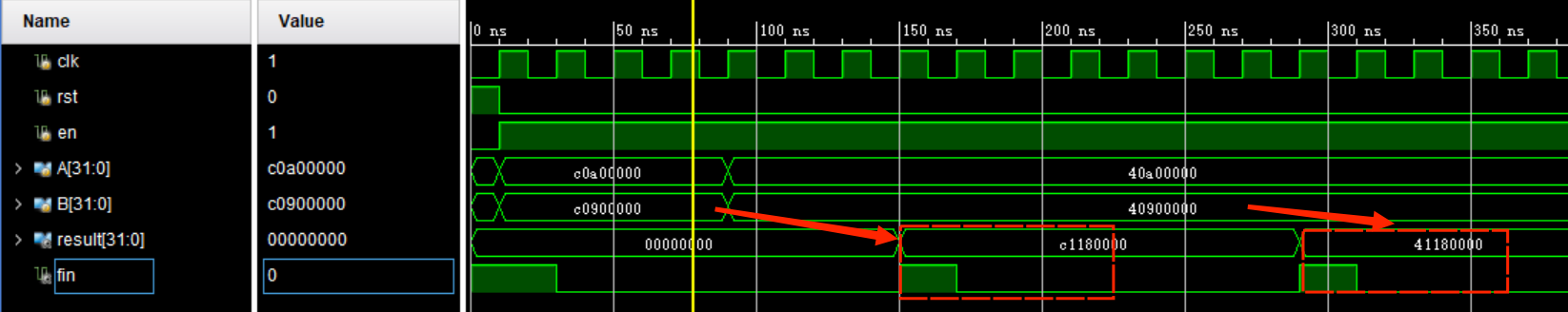


图4.3.2 浮点数加法仿真结果分析

对于该段仿真，我们可以看到结果产生和输入改变之间存在一定的时延，因为浮点数的状态机产生结果需要多个周期，同时可以看到结果符合预期。

**五、讨论与心得**

1. 通过本次实验，我对基本算数操作（乘法、除法、浮点数加减）的硬件实现有了更全面的了解，能够更清晰地理解操作的各个流程。同时，从最初较多资源浪费的方法到后续优化方案的演进中，我也感受到设计者的巧妙思路。

2. 本次实验中，由于浮点数是选做实验，本来想放弃，但是最后还是坚持下来完成了该项实验，并通过这个实验，对较复杂状态机的编写进行了巩固和提高，有助于Verilog代码能力的进一步突破。

3. 本次实验整体相对顺利，实现了课堂内容和实验的交互以及互相促进。