组成原理实验课程第_二_次实报告

实验名称	迭代乘法器			班级	李涛班
学生姓名	田晋宇	学号	2212039	指导老师	董前琨
实验地点	实验楼 A306		实验时间		

一、实验目的

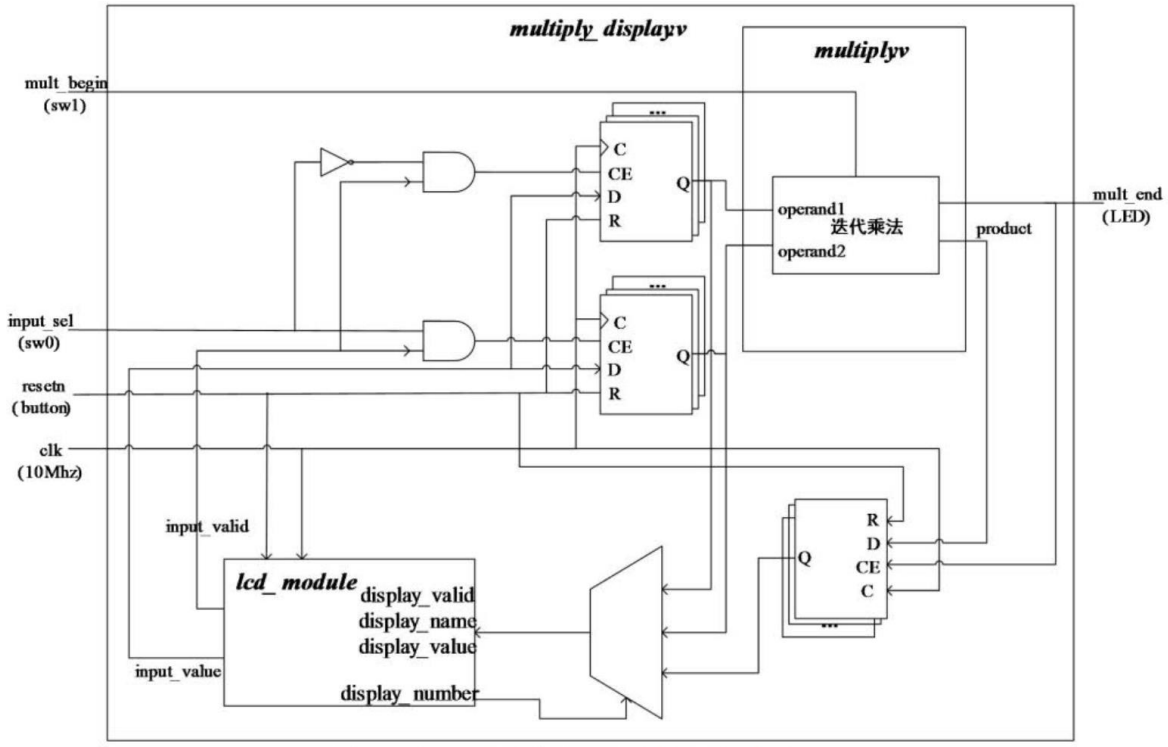
1. 理解定点乘法的不同实现算法的原理，掌握基本实现算法。
2. 熟悉并运用 verilog 语言进行电路设计。
3. 为后续设计 cpu 的实验打下基础。

二、实验内容说明

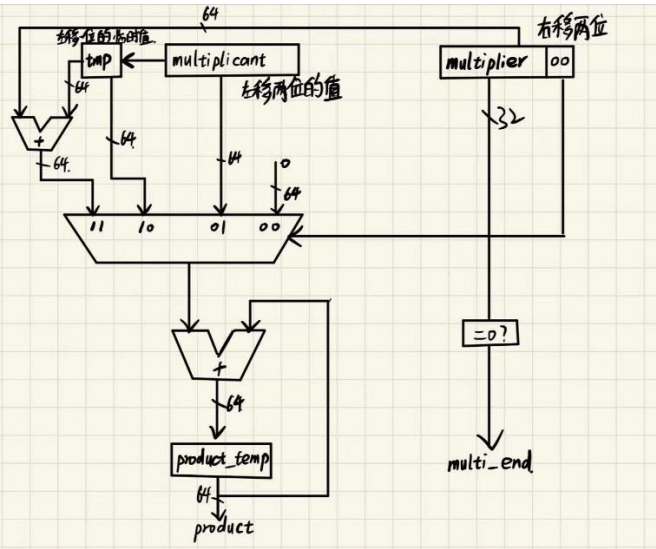
1. 请结合实验指导手册中的实验二（定点乘法器实验）完成性能改进，不在是原始的最长 32 个时钟周期完成乘法，注意以下几点：
2. 本次主要修改 multipler,v 模块，建议从两位乘开始进行，此外还有华莱士树等高级优化方式，鼓励大家尝试。
3. 实验报告中需要补充原理图，并对原理图进行解释说明。原理图参照图 3.2 进行修改，建议使用 visio 画图（别的画图软件也可，不会画图的可以手绘然后照片放报告里面）。
4. 实验报告中需要有仿真结果（波形截图），并针对图中的数据解释说明，还需要有实验箱上箱验证的照片，同样，针对照片中的数据也需要解释说明。
5. 实验报告模板参考百度云盘文件，注意提交截至时间为 4 月 12 日下午 18：00。

三、实验原理图

顶层模块的原理图与修改前相同：



乘法器修改后的实验原理图如下图：



四、实验步骤

本次实验是对迭代乘法器的改进，因此只涉及对乘法器模块的修改，顶层模块不需要进行修改。

原本的加法器中每一个时钟周期乘数移动一位，如果完成 32 位二进制数的乘法需要 32 个时钟周期，改进后的乘法器每次移动两位，使得完成乘法的过程只需要 16 个时钟周期，实现性能的优化。时序逻辑部分即对操作数的移位修改如下：

```
59      //加载被乘数，运算时每次左移两位
60      always @ (posedge clk)
61      begin
62          if (mult_valid)
63          begin    // 如果正在进行乘法，则被乘数每时钟左移两位
64              multiplicand <= {multiplicand[61:0],2'b0};
65          end
66          else if (mult_begin)
67          begin   // 乘法开始，加载被乘数，为乘数1的绝对值
68              multiplicand <= {32'd0,op1_absolute};
69          end
70      end
71
72      //加载乘数，运算时每次右移两位
73      always @ (posedge clk)
74      begin
75          if (mult_valid)
76          begin   // 如果正在进行乘法，则乘数每时钟右移两位
77              multiplier <= {2'b0,multiplier[31:2]};
78          end
79          else if (mult_begin)
80          begin   // 乘法开始，加载乘数，为乘数2的绝对值
81              multiplier <= op2_absolute;
82          end
83      end
```

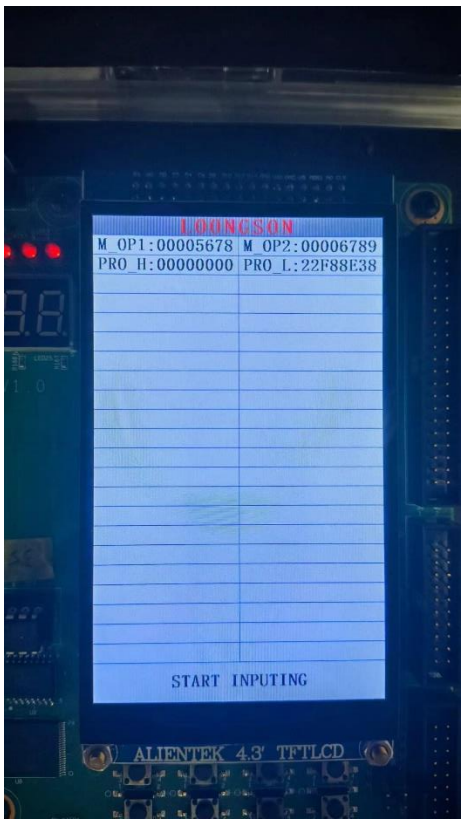
对与部分积的赋值共分为四种情况，分别为 multiplier 最低两位为 00,01,10,11 的时候，代码实现如下：

```
// 部分积：乘数不位为1，由被乘数左移得到；乘数不位为0，部分积为0
wire [63:0] partial_product;
assign partial_product = multiplier[1:0] == 2'b01 ? multiplicand:
                         multiplier[1:0] == 2'b10 ? {multiplicand[62:0],1'b0}:
                         multiplier[1:0] == 2'b11 ? multiplicand + {multiplicand[62:0],1'b0}:
                         64'd0;
```

五、实验结果分析

对修改后的代码进行仿真运行，获得如下波形图，通过计算验证结果正确。

在实验箱上分别输入操作数 1,2，最终所得结果与仿真结果相同。



六、总结感想

通过本次实验对 verilog 语言有了更好的掌握，尤其是对时序逻辑电路代码的理解和编写有了进一步的认识。深刻理解迭代乘法器的原理，以及对乘法器的优化方法，为以后 cpu 的设计与改进打下良好基础。