组成原理实验课程第_2_次实验报告

实验名称	数据运算：定点乘法			班级	李涛
学生姓名	胡博浩	学号	2212998	指导老师	董前琨
实验地点	津南实验楼 A308		实验时间	2024.4.11	

1、 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。

2、 实验内容说明

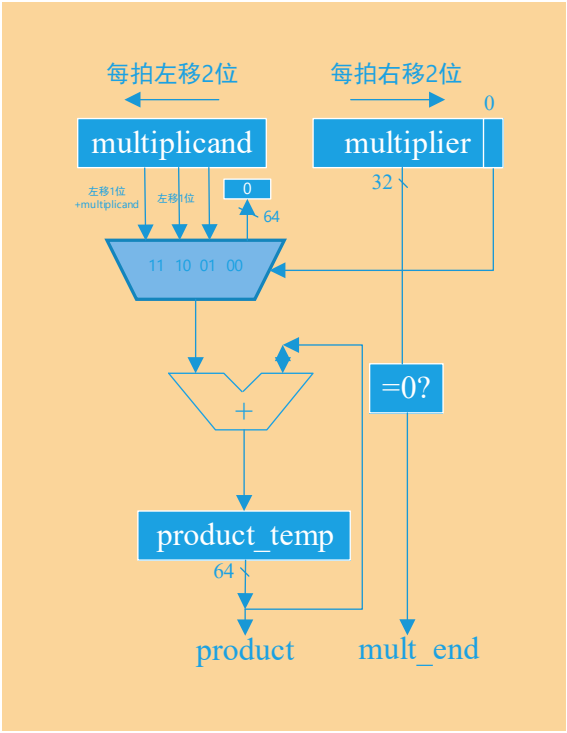
请结合实验指导手册中的实验二（定点乘法器实验）完成性能改进，不在是原始的最长 32 个时钟周期完成乘法，注意以下几点：

（1）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。

（2）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）。

（3）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。

实验原理图



该原理图描述了一个乘法器的改进版本。

（1）初始化：开始时，被乘数和乘数分别存储在 64 位和 32 位的寄存器中。被乘数初始时放在 64 位寄存器中的低 32 位。

（2）迭代过程：每个时钟周期，被乘数寄存器和乘数寄存器同时向左移动两位。乘数寄存器中的最后两位的值用作控制信号，控制多路选择器的输出。

（3）部分积计算：根据乘数寄存器的最后两位的值，选择相应的被乘数部分，并加到部分

积中。加法器根据这两位的值，选择正确数量的被乘数部分加到部分积中，以保持乘法计算的正确性。

（4）判断乘法结束：检查乘数是否为 0，若为 0 则表示乘法结束。

通过这些修改，乘法器的性能得到提升，每次计算的时间缩短到原来的一半，同时仍保持了计算结果的准确性。

3、 实验步骤

本次实验只修改了 multiply.v 文件。

（1）修改被乘数每个时钟周期移动的位数

被乘数每个时钟周期左移 2 位，每次取原被乘数的后 62 位和两个 0 位拼接后构成新的 64 位被乘数。

```
//加载被乘数，运算时每次左移2位
reg  [63:0] multiplicand;
always @ (posedge clk)
begin
    if (mult_valid)
    begin    // 如果正在进行乘法，则被乘数每时钟左移2位
        multiplicand <= {multiplicand[61:0],2'b00};
    end
    else if (mult_begin)
    begin   // 乘法开始，加载被乘数，为乘数1的绝对值
        multiplicand <= {32'd0,op1_absolute};
    end
end
```

（2）修改乘数每个时钟周期移动的位数

乘数每个时钟周期右移 2 位，每次取两个 0 位和原乘数的前 30 位拼接后构成新的 32 位乘数。

```
//加载乘数，运算时每次右移2位
reg  [31:0] multiplier;
always @ (posedge clk)
begin
    if (mult_valid)
    begin   // 如果正在进行乘法，则乘数每时钟右移一位
        multiplier <= {2'b00,multiplier[31:2]};
    end
    else if (mult_begin)
            begin   // 乘法开始，加载乘数，为乘数2的绝对值
        multiplier <= op2_absolute;
    end
end
```

（3）修改部分积的计算方法

根据乘数的最后两位进行判断：

a) 若为 00，部分积为 0

b) 若为 01，部分积为被乘数

c) 若为 10，部分积为被乘数的 2 倍

d) 若为 11，部分积为被乘数的三倍

所以对加法器与多路选择器的部分进行修改，语句类似与 c++中的条件表达式。

```
wire [63:0] partial_product;
assign partial_product = multiplier[1:0]==2'b11 ? multiplicand +multiplicand+multiplicand:
                         multiplier[1:0]==2'b10 ? multiplicand+multiplicand:
                         multiplier[1:0]==2'b01 ? multiplicand:64'd0;
```

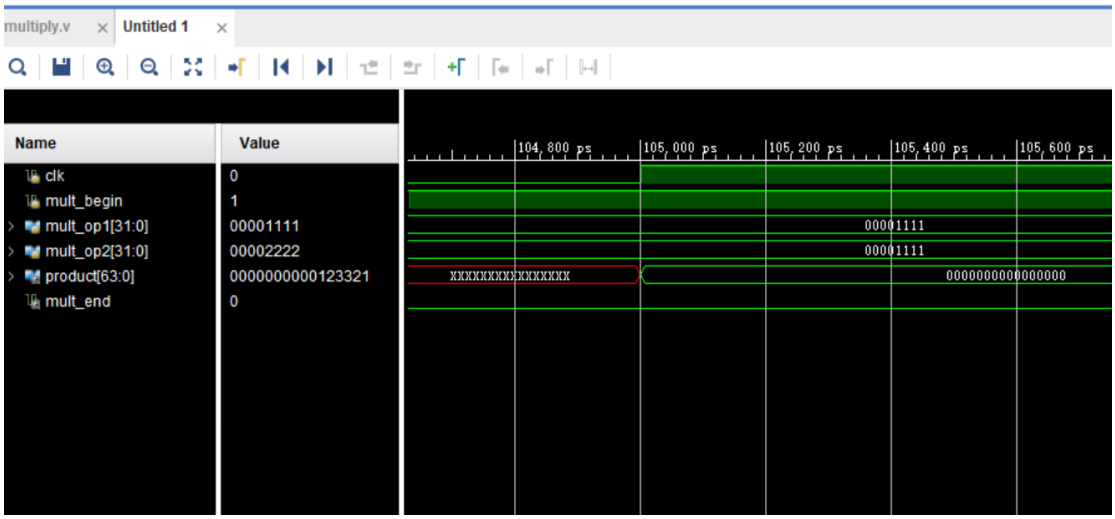
4、 实验结果分析

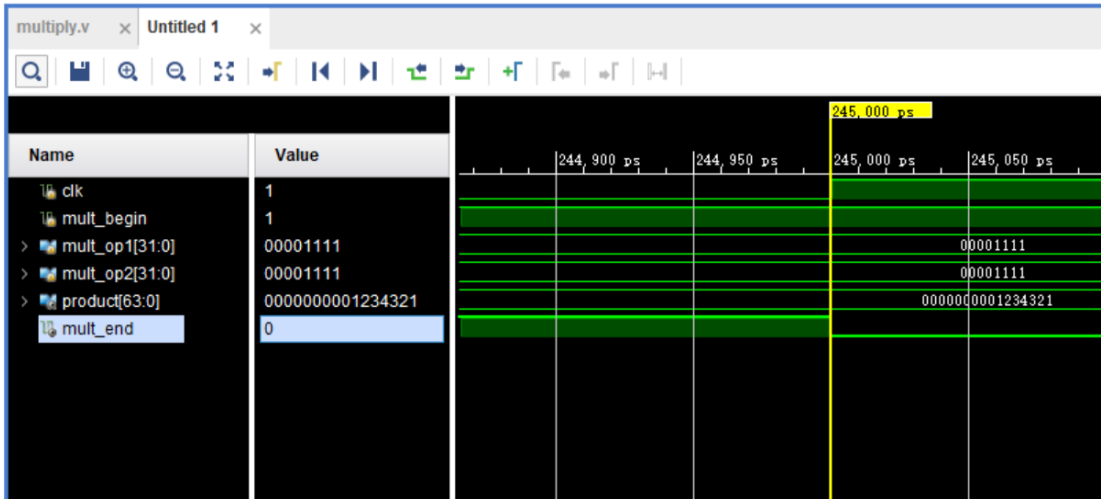
（1）仿真实验

a)改进前：

从 105000ps 开始乘法计算，直到 245000ps 时 mult_end=0 乘法结束。

一个周期 10ns，则改进前一次乘法操作需要（245-105）/10=14 周期。



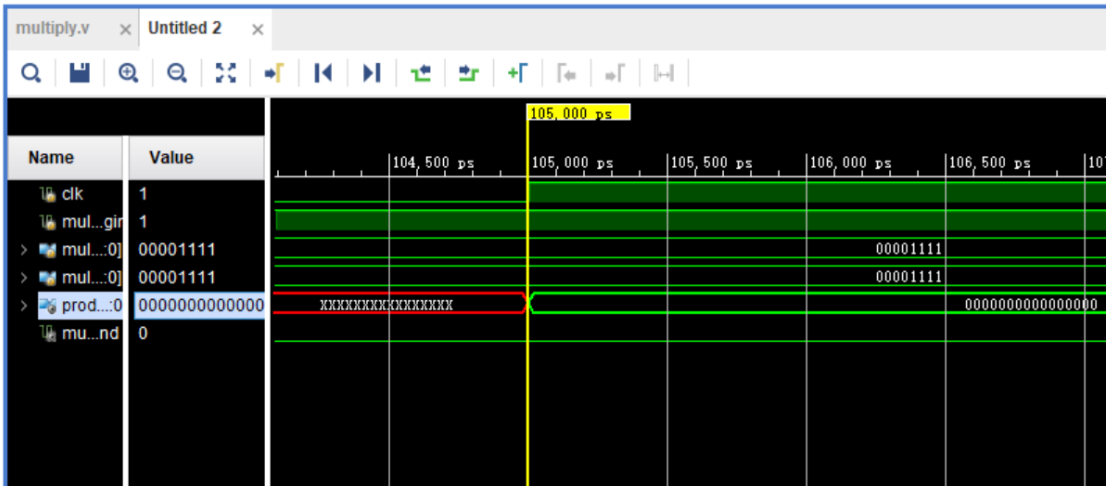


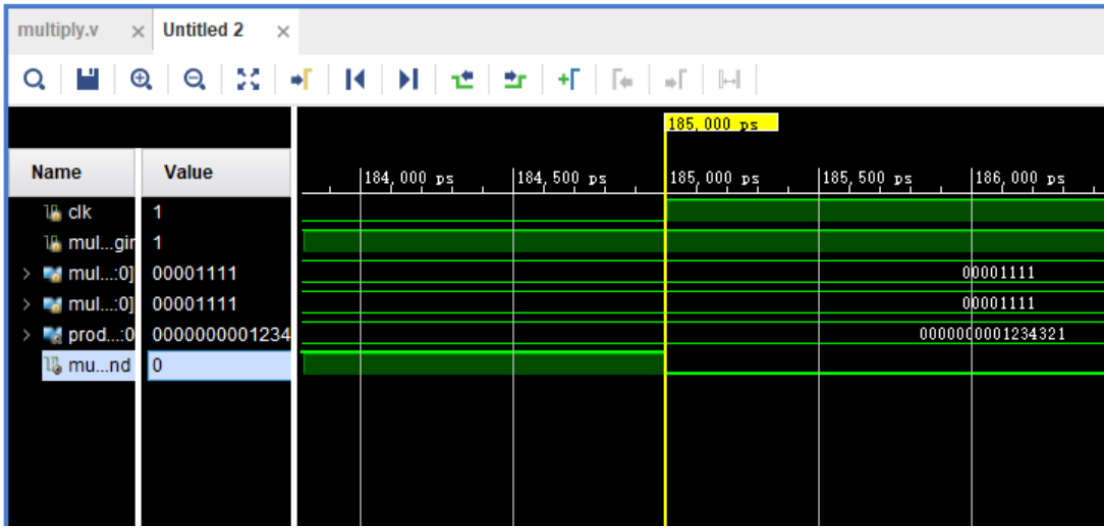
0x00001111*0x00001111=0x0000000001234321，运算正确

b）改进后

从 105000ps 开始乘法计算，直到 185000ps 时 mult_end=0 乘法结束。

一个周期 10ns，则改进前一次乘法操作需要（185-105）/10=8 周期。





0x00001111*0x00001111=0x0000000001234321，运算正确

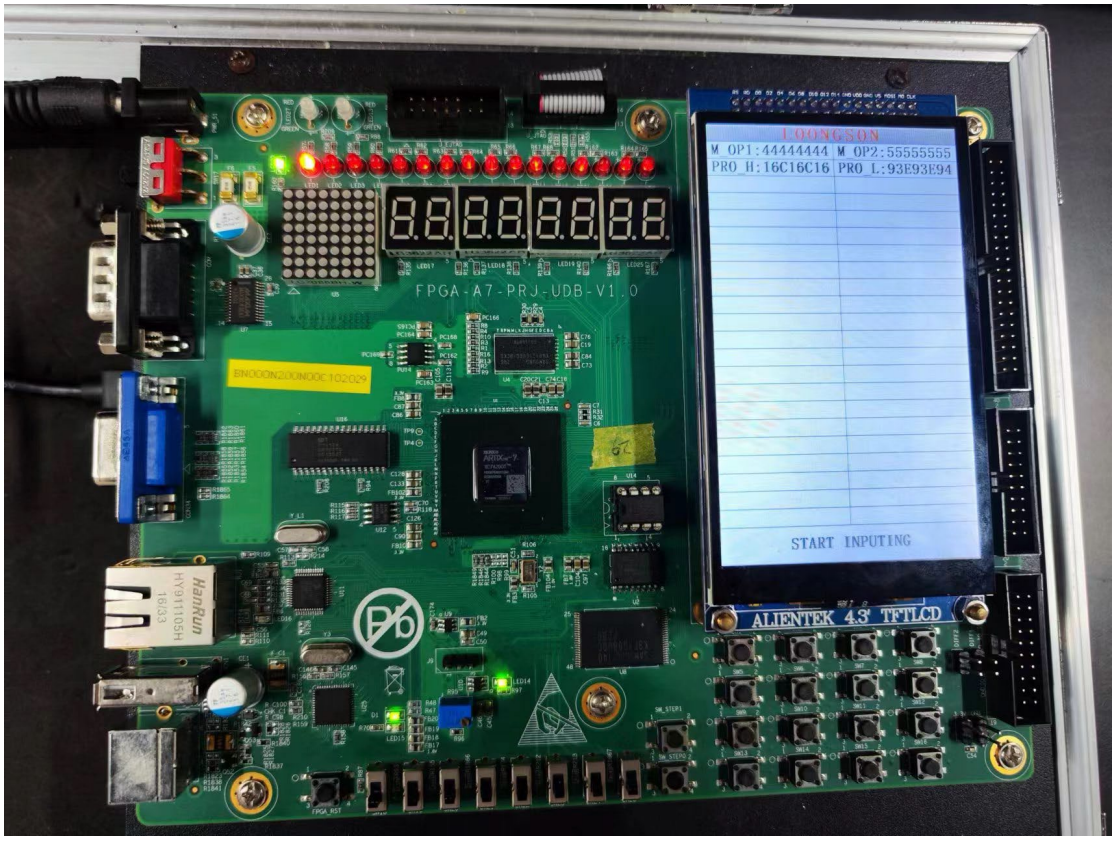
c）结果分析

改进后乘法操作的周期由 14 降为了 8，性能提升将近一倍；而且改进后乘法操作的运算结果正确。因此，实验成功。

（2）上箱实验

左起第一个 LED 灯亮起，表示运算结束

0x44444444*0x55555555=0x16C16C1693E93E94,运算结果正确



5、 总结感想

这次实验让我更加熟悉了实验流程和 Vivado 平台的使用方法。我学会了如何在实验中利用 Verilog 语言和实验箱进行硬件设计。

我还复习了 Visio 画图的基本方法，这为我将电路图呈现得更加清晰和准确提供了技能支持。

最重要的是，通过本次实验，我学会了实现一次性移多位来实现乘法器的优化，不仅巩固了我的理论知识，还让我对乘法器的工作原理有了更加深入的理解。乘法器作为计算机中的重要组成部分，其原理的理解对于我理解整个计算机系统的运作方式至关重要。

总的来说，这次实验让我受益匪浅，我将继续努力学习，不断提升自己在数字电路设计和硬件开发领域的能力。