组成原理课程第二次实验报告

实验名称：　定点乘法

学号：　2310764　姓名：　王亦辉　班次：　计科一班

一、实验目的

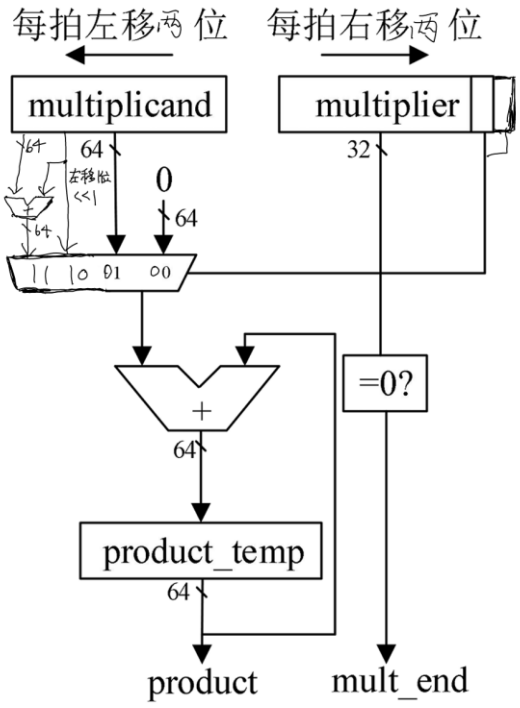
1. 理解定点乘法的不同实现算法的原理，掌握基本实现算法。
2. 熟悉并运用 verilog 语言进行电路设计。
3. 为后续设计 cpu 的实验打下基础

二、实验内容说明

修改 multiply.v 文件，让乘法器每次移动两位而不是一位，来进行部分乘积的计算。

三、实验原理图

从两路选择器改成四路选择器，使用 multiplier 的低 2 位进行选择；然后根据相应乘的数，添上 10 和 11 被选择的数 (multiplicand * 2 和 multiplicand *3)。



四、实验步骤

修改部分乘积的 assign 语句，因为从二路选择器换成四路选择器了。

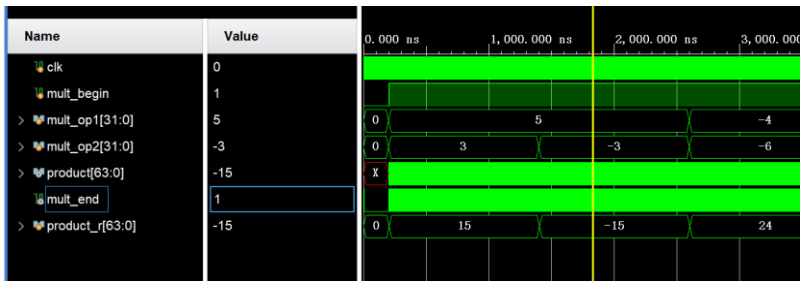
```
assign partial_product =
    (multiplier[1:0] == 2'b00) ? 64'd0 :
    (multiplier[1:0] == 2'b01) ? multiplicand :
    (multiplier[1:0] == 2'b10) ? (multiplicand << 1) :
    ((multiplicand << 1) + multiplicand);
```

更改被乘数和乘数的移位逻辑，改成每次移两位。

```
//加载被乘数，运算时每次左移两位
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两位
        multiplicand <= {multiplicand[61:0],2'b0};
    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

//加载乘数，运算时每次右移两位
always @ (posedge clk)
begin
    if (mult_valid)
    begin   // 如果正在进行乘法，则乘数每时钟右移两位
        multiplier <= {2'b0,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
```

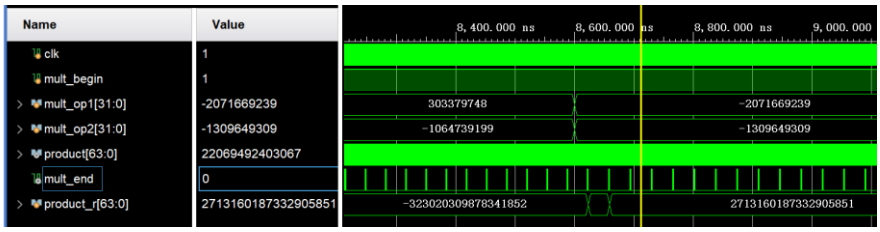
五、实验结果分析

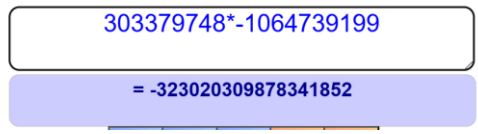


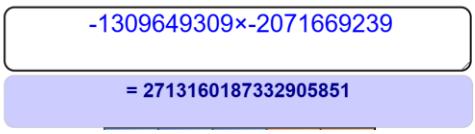
对于三种情况：正数*正数， 正数*负数，负数*负数 进行验证。

3*5 = 15；-3*5 = -15；(-4)*(-6) = 24；

结果正确





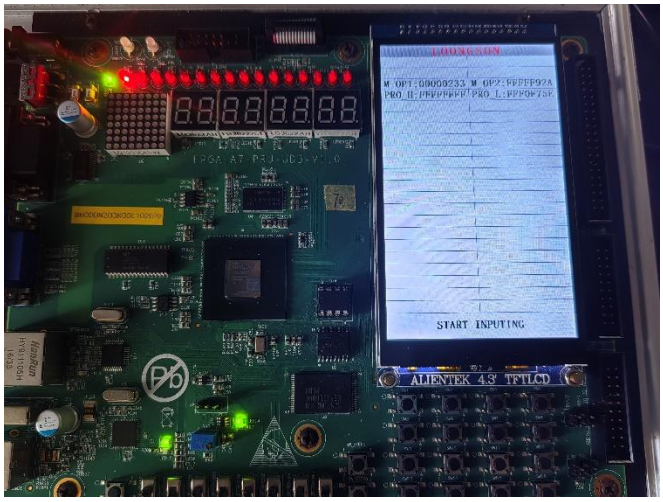


对较大的数进行乘法，通过计算器验证可知，结果也正确。

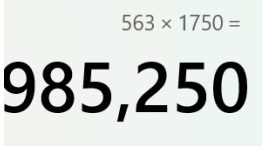
上箱验证：(等号左边是 16 进制，等号右边是 10 进制)
第一组：

输入：FFFFFFFF * FFFFFF63 = -1 * (-157)
输出：9D = 157
正确。

第二组：



输入：00000233 * FFFFF92A = 563 * (-1750)
输出：FFFFFFFF_FFF0F75E = - 985250



正确。

第三组：

输入：000000AB * 00000023 = 171 * 35

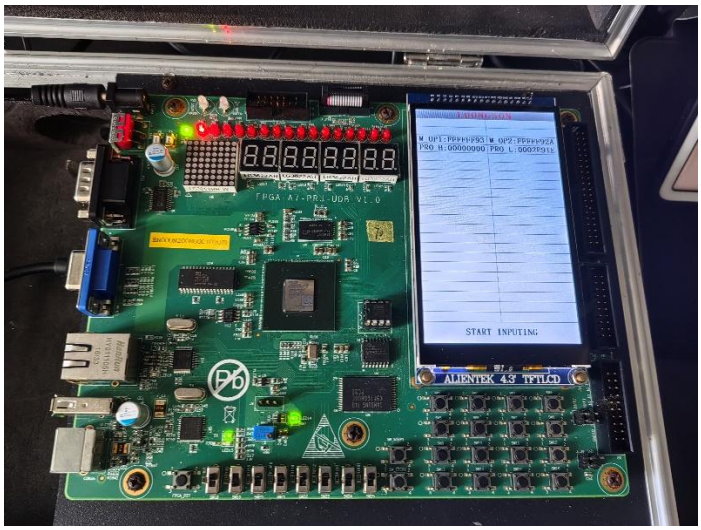
输出：1761= 5985

171 × 35 =

5,985

正确。

第四组：



输入：FFFFFF93 * FFFFF92A = (-109) * (-1750)

输出：2E91E = 190750

109 × 1750 =

190,750

正确。

六、总结感想

乘法可以用迭代的加法来实现，如果 1 位变成 2 位会更快，那么为什么不一步移动 32 位？有什么取舍？