**组成原理实验课程第 2 次实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验名称 | 定点乘法 | | | 班级 | 李涛老师 |
| 学生姓名 | 孙蕗 | 学号 | 2112060 | 指导老师 | 董前琨 |
| 实验地点 | A308 | | 实验时间 | 2023.4.4 | |

1. **实验目的**
2. 理解定点乘法的不同实现算法的原理，掌握基本实现算法。
3. 熟悉并运用verilog语言进行电路设计。
4. 为后续设计cpu的实验打下基础。
5. **实验设备**
6. 装有Xilinx Vivado的计算机一台。
7. LS-CPU-EXB-002教学系统实验箱一套。
8. **实验要求**
9. 将原有的迭代乘法改进成两位乘法，即每个时钟周期移位移两位，从而提高乘法效率。（其他形式的优化也建议尝试）
10. 将改进后的乘法器进行仿真验证
11. 将改进后的乘法器进行上实验箱验证，上箱验证时调整数据不在前4格显示
12. 实验报告中的原理图为迭代乘法的算法图，不再是顶层模块图
13. 使用三目运算符实现
14. **实验内容说明**
15. 学习并理解计算机中定点乘法器的多种实现算法的原理，重点掌握迭代乘法的实现算法。
16. 自行设计本次实验的方案，画出结构框图，详细标出输入输出端口，本次实验的乘法器建议采用迭代的方式实现，如果能力有余的，也可以采用其他效率更高的算法实现。本次实验要求实现的乘法为有符号乘法，因此需要注意计算机存储的有符号数都是补码的形式，设计方案传递进来的数也需是补码。
17. 根据设计的实验方案，使用verilog编写相应代码。
18. 对编写的代码进行仿真，得到正确的波形图。
19. 将以上设计作为一个单独的模块，设计一个外围模块去调用该模块，见图3.1。外围模块中需调用封装好的LCD触摸屏模块，显示两个乘数和乘法结果，且需要利用触摸功能输入两个乘数。
20. 将编写的代码进行综合布局布线，并下载到实验箱中的FPGA板子上进行演示
21. **实验原理图**



参与运算的为两个乘数的绝对值，乘法结果也是绝对值，需要单独判断符号位后校正乘积乘数每次右移两位，被乘数每次左移两位。

判断乘数的最低两位，如果乘数最低两位是 00，部分积为 0；如果乘数最低两位是 01，部分积为被乘数的值；如果乘数最低两位是 10，部分积为被乘数的值的2倍；如果乘数最低两位是 11，部分积为被乘数的值的3倍。乘积不停地累加部分积即可就得到结果。当乘数所有位全为 0，标志计算完成。

定点乘法参考设计的顶层模块图：



1. **实验步骤**
2. **multiply.v修改（使用case判断）**
3. multiply.v修改，实现被乘数每次左移两位

|  |  |
| --- | --- |
| 42 | //加载被乘数，运算时每次左移两位 |
| 43 | reg [63:0] multiplicand; |
| 44 | always @ (posedge clk) |
| 45 | begin |
| 46 | if (mult\_valid) |
| 47 | begin // 如果正在进行乘法，则被乘数每时钟左移两位 |
| 48 | multiplicand <= {multiplicand[61:0],2'b0}; |
| 49 | end |
| 50 | else if (mult\_begin) |
| 51 | begin // 乘法开始，加载被乘数，为乘数1的绝对值 |
| 52 | multiplicand <= {32'd0,op1\_absolute}; |
| 53 | end |
| 54 | end |

1. multiply.v修改，乘数每次右移两位

|  |  |
| --- | --- |
| 56 | //加载乘数，运算时每次右移两位 |
| 57 | reg [31:0] multiplier; |
| 58 | always @ (posedge clk) |
| 59 | begin |
| 60 | if (mult\_valid) |
| 61 | begin // 如果正在进行乘法，则乘数每时钟右移两位 |
| 62 | multiplier <= {2'b0,multiplier[31:2]}; |
| 63 | end |
| 64 | else if (mult\_begin) |
| 65 | begin // 乘法开始，加载乘数，为乘数2的绝对值 |
| 66 | multiplier <= op2\_absolute; |
| 67 | end |
| 68 | end |

|  |  |
| --- | --- |
| 70 | // 部分积：乘数末位为1，由被乘数左移得到；乘数末位为0，部分积为0; |
| 71 | reg [63:0] partial\_product; |
| 72 | always@(multiplier[1:0]) |
| 73 | begin |
| 74 | case(multiplier[1:0]) // 根据乘数最低两位判断 |
| 75 | 2'd00: // 乘数最低两位为 00，部分积为 0 |
| 76 | partial\_product <= 64'd0; |
| 77 | 2'd01: // 乘数最低两位为 01，部分积为被乘数的值 |
| 78 | partial\_product <= multiplicand; |
| 79 | 2'd11: // 乘数最低两位为 11，部分积为被乘数的值的三倍 |
| 80 | partial\_product <= multiplicand + multiplicand +multiplicand; |
| 81 | 2'd10: // 乘数最低两位为 10，部分积为被乘数的值的二倍 |
| 82 | partial\_product <= multiplicand + multiplicand; |
| 83 | endcase |
| 84 | end |

1. **multiply.v修改（使用三目运算符）**
2. multiply.v修改，实现被乘数每次左移两位

|  |  |
| --- | --- |
| 42 | //加载被乘数，运算时每次左移两位 |
| 43 | reg [63:0] multiplicand; |
| 44 | always @ (posedge clk) |
| 45 | begin |
| 46 | if (mult\_valid) |
| 47 | begin // 如果正在进行乘法，则被乘数每时钟左移两位 |
| 48 | multiplicand <= {multiplicand[61:0],2'b00}; |
| 49 | end |
| 50 | else if (mult\_begin) |
| 51 | begin // 乘法开始，加载被乘数，为乘数1的绝对值 |
| 52 | multiplicand <= {32'd0,op1\_absolute}; |
| 53 | end |
| 54 | end |

1. multiply.v修改，乘数每次右移两位

|  |  |
| --- | --- |
| 56 | //加载乘数，运算时每次右移两位 |
| 57 | reg [31:0] multiplier; |
| 58 | always @ (posedge clk) |
| 59 | begin |
| 60 | if (mult\_valid) |
| 61 | begin // 如果正在进行乘法，则乘数每时钟右移两位 |
| 62 | multiplier <= {2'b00,multiplier[31:2]}; |
| 63 | end |
| 64 | else if (mult\_begin) |
| 65 | begin // 乘法开始，加载乘数，为乘数2的绝对值 |
| 66 | multiplier <= op2\_absolute; |
| 67 | end |
| 68 | end |

|  |  |
| --- | --- |
| 70 | // 部分积：乘数末位为1，由被乘数左移得到；乘数末位为0，部分积为0; |
| 71 | wire[63:0] partial\_product;//64位数，保存部分积 |
| 72 | assign partial\_product = multiplier[1] ? (multiplier[0]?multiplicand+multiplicand+  multiplicand:multiplicand+multiplicand):(multiplier[0]?multiplicand:64'd0);//乘数最低两位是 00，部分积为 0；如果乘数最低两位是 01，部分积为被乘数的值；如果乘数最低两位是 10，部分积为被乘数的值的2倍；如果乘数最低两位是 11，部分积为被乘数的值的3倍。乘积不停地累加部分积即可就得到结果。 |

1. **multiply\_display.v**

修改multiply\_display.v，改变实验箱上输出的格子位置。

|  |  |
| --- | --- |
| 129 | //-----{输出到触摸屏显示}begin |
| 130 | //根据需要显示的数修改此小节， |
| 131 | //触摸屏上共有44块显示区域，可显示44组32位数据 |
| 132 | //44块显示区域从1开始编号，编号为1~44， |
| 133 | always @(posedge clk) |
| 134 | begin |
| 135 | case(display\_number) |
| 136 | 6'd5 ://第5个格子显示 |
| 137 | begin |
| 138 | display\_valid <= 1'b1; |
| 139 | display\_name <= "M\_OP1"; |
| 140 | display\_value <= mult\_op1; |
| 141 | end |
| 142 | 6'd6 ://第6个格子显示 |
| 143 | begin |
| 144 | display\_valid <= 1'b1; |
| 145 | display\_name <= "M\_OP2"; |
| 146 | display\_value <= mult\_op2; |
| 147 | end |
| 148 | 6'd7 ://第7个格子显示 |
| 149 | begin |
| 150 | display\_valid <= 1'b1; |
| 151 | display\_name <= "PRO\_H"; |
| 152 | display\_value <= product\_r[63:32]; |
| 153 | end |
| 154 | 6'd8 ://第8个格子显示 |
| 155 | begin |
| 156 | display\_valid <= 1'b1; |
| 157 | display\_name <= "PRO\_L"; |
| 158 | display\_value <= product\_r[31: 0]; |
| 159 | end |
| 160 | default : |
| 161 | begin |
| 162 | display\_valid <= 1'b0; |
| 163 | display\_name <= 48'd0; |
| 164 | display\_value <= 32'd0; |
| 165 | end |
| 166 | endcase |
| 167 | end |

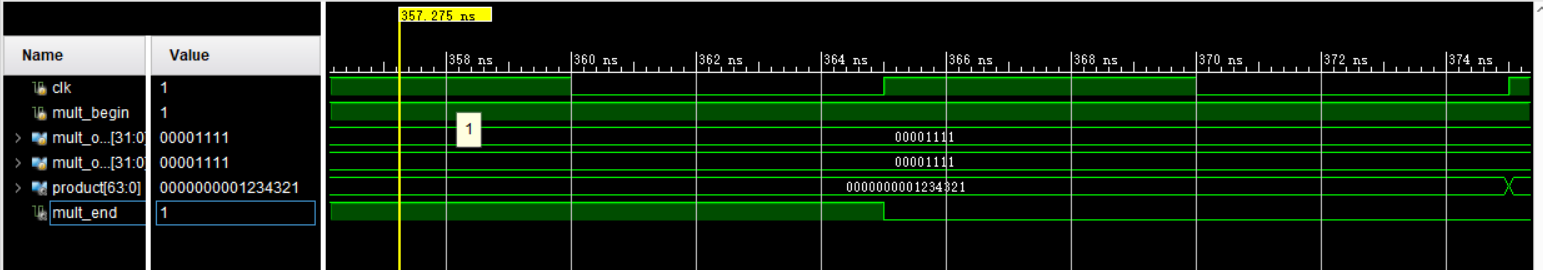
1. **实验结果分析**
2. 仿真波形分析

multiplicand=00001111

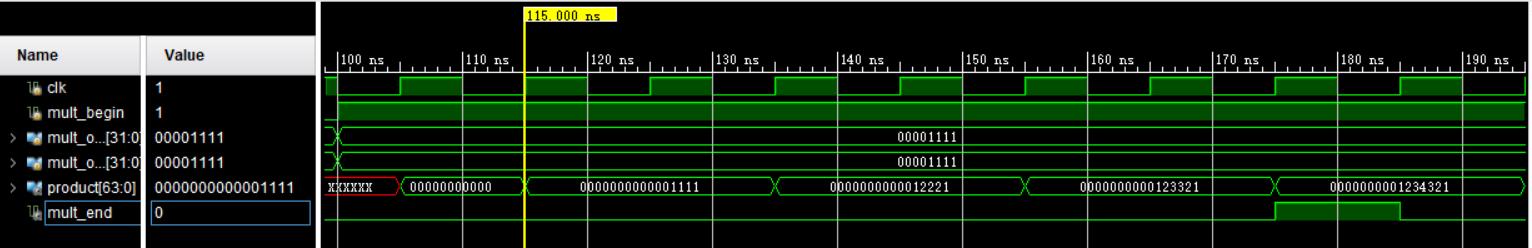
multiplier=00001111

00001111×00001111=0000 0000 0123 4321

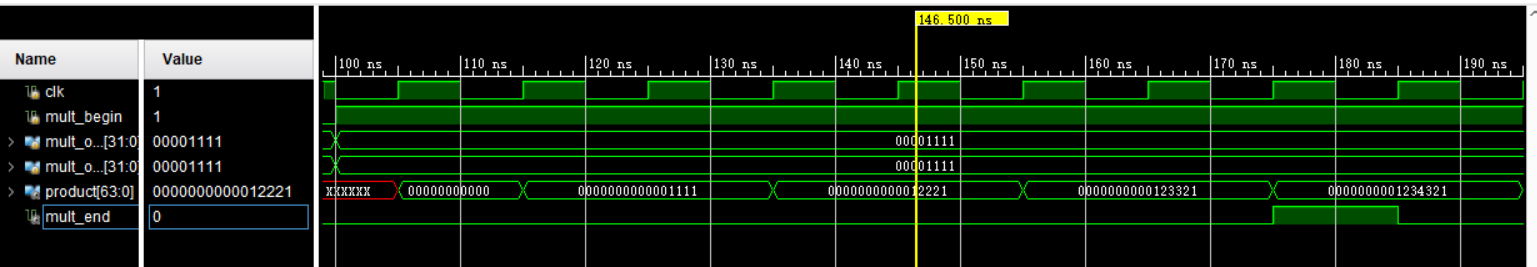
1. 仿真波形最终结果



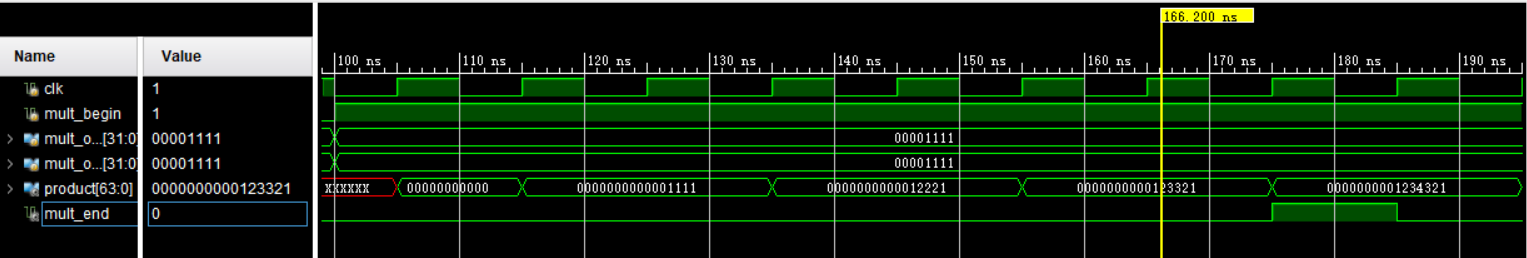
1. 过程分析
2. 第一个脉冲，乘数multiplier后两位为01，部分积partial\_product应为0000 0000 0000 1111，即被乘数的值1111，此时的product也为被乘数的值0000 0000 0000 1111。



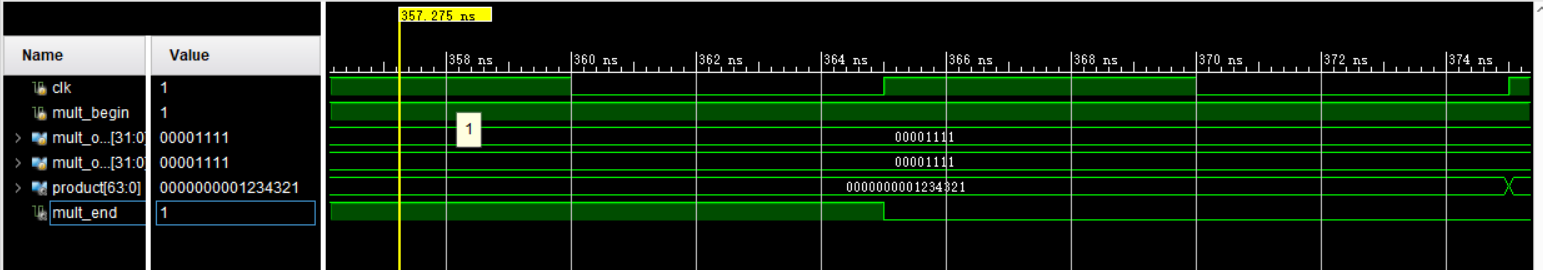
1. 第二个脉冲，乘数multiplier向右移动两位，被乘数multiplicand向左移动两位，乘数multiplier最后两位为00，部分积partial\_product为0，此时乘积product 不变。
2. 第三个脉冲，乘数multiplier向右移动两位，被乘数multiplicand向左移动两位，乘数multiplier最后两位为01，部分积partial\_product为被乘数的值0000 0000 0001 1110，乘积product应加上partial\_product,变成0000 0000 0001 2221。



1. 第四个脉冲，乘数multiplier向右移动两位，被乘数multiplicand向左移动两位，乘数最后两位为00，partial\_product=0，product不变。
2. 乘数multiplier向右移动两位，被乘数multiplicand向左移动两位，乘数最后两位为01，部分积partial\_product为被乘数的值0000 0000 0011 1100，乘积product应加上partial\_product,变成0000 0000 0012 3321。

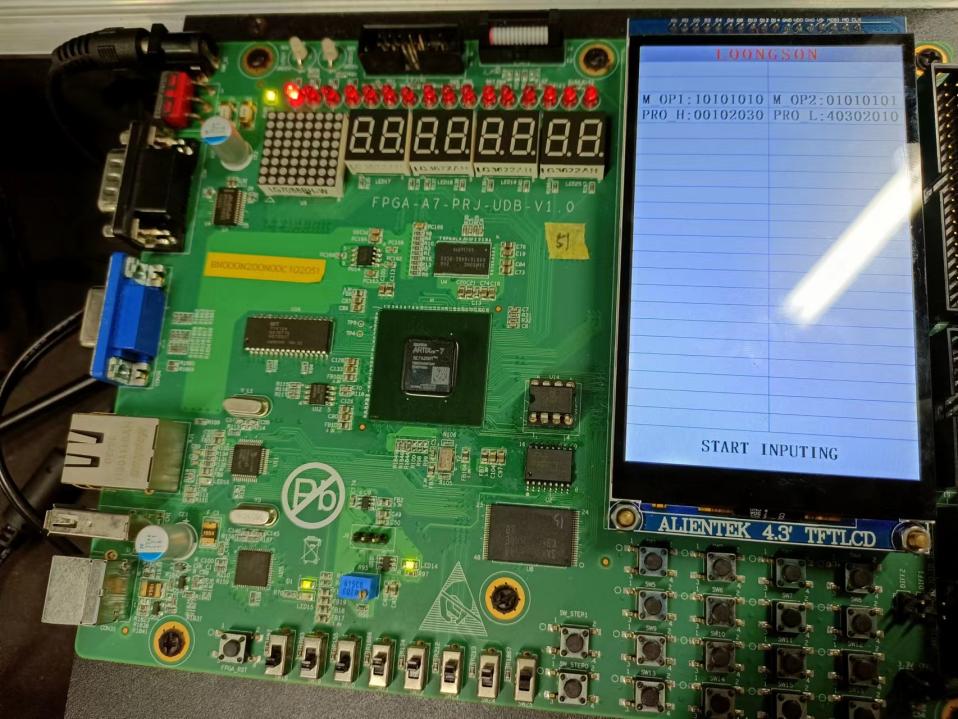


1. 乘数multiplier向右移动两位，被乘数multiplicand向左移动两位，乘数最后两位为01，部分积partial\_product为被乘数的值0000 0000 0111 1000，乘积product应加上partial\_product,变成0000 0000 0123 4321。



1. 实验箱

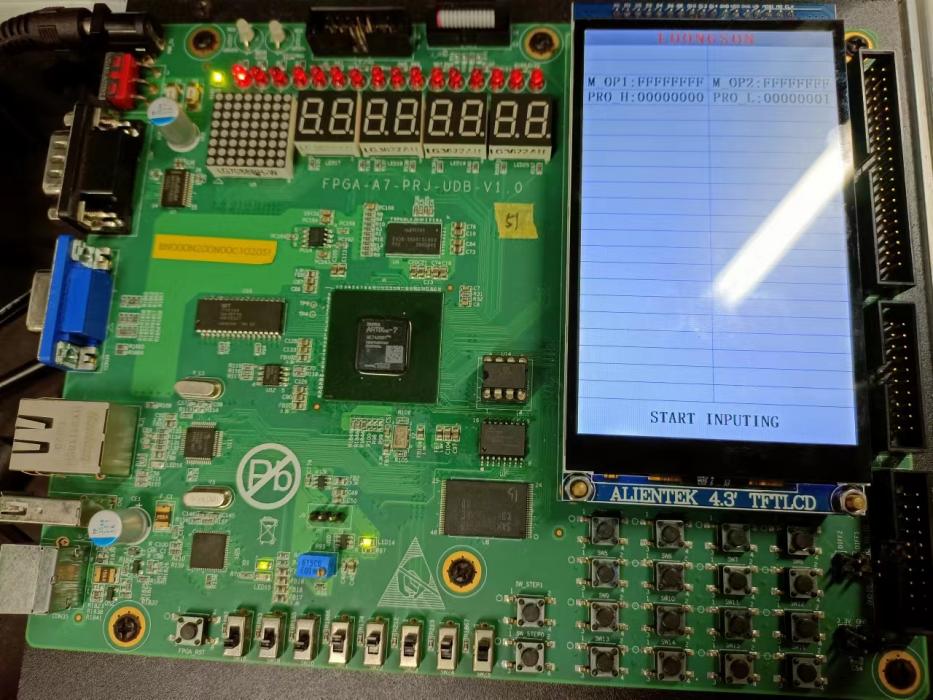
M\_OP1显示在第5格，M\_OP2显示在第6格，PRO\_H显示在第7格，PRO\_L显示在第8格。



被乘数为10101010，为正数

乘数为01010101，为正数

乘积为0010 2030 4030 2010，结果正确



被乘数为FFFF FFFF，是负数，绝对值为0000 0001

乘数位FFFF FFFF，是负数，绝对值为0000 0001

乘积为正数，为0000 0000 0000 0001，结果正确

1. **总结感想**

通过这个实验，对Verilog的应用能力有了更深入的了解和提升。通过将每个时钟周期的移位改为移两位，可以提高计算的精度和效率。同时，使用三目运算符可以简化代码的编写和理解，使代码更加清晰。在修改移位时，需要注意运算的溢出问题；在使用三目运算符时，需要注意不同运算符的优先级和使用方式。