同济大学

计算机科学与技术系

计算机组成原理课程设计实验报告

学　　号：1853790

姓　　名：庄镇华

专　　业：计算机科学与技术

授课老师：陈永生

目录

一、 实验内容

1. 了解 32 位带符号、无符号乘法器的实现原理

2. 使用 $Verilog$ 实现 32 位无符号乘法器和带符号乘法器

二、 模块建模

2.1 功能描述

无符号乘法器功能为：将两个 32 位无符号数相乘，得到一个 64 位无符号数。将低 32 位存放在专用寄存器 LO 中，高 32 位存放在寄存器 HI 中。

带符号乘法器功能为：将两个 32 带符号数相乘，得到一个 64 位带符号数。只选取低 32 位存入指定寄存器。执行乘法指令过程中不产生异常。

2.2 实现原理

无符号乘法器：两个二进制数 a 和 b 相乘，可以认为是 a 和 b 的每一位相乘移位后的结果相加。关于 a 与 b 的每一位相乘产生的中间结果，如果 b 那位是 0，那么中间结果就是 0；如果是 1，那么中间结果就是在 a 前后补上相应位数的零通过字符拼接的方式表示。然后将这些中间乘积相加就是最后的结果。

带符号乘法器：可以在无符号乘法器的基础上，先将原始输入的数据进行处理，如果是负数就先转化为正数，接着按无符号乘法器的原理处理，最后判断结果的符号，如果原始输入数据同号，那么结果直接输出即可；如果原始输入数据异号，那么需要将结果转化为负数然后才能输出。

2.3 模块代码

```
`timescale 1ns / 1ps
//MULTU.v
module MULTU(
        input clk,
        input reset,
        input [31:0] a,
        input [31:0] b,
        output [63:0] z
);
```

```
    reg [63:0] temp;
    reg [63:0] stored0;
    reg [63:0] stored1;
    reg [63:0] stored2;
    reg [63:0] stored3;
    reg [63:0] stored4;
    reg [63:0] stored5;
    reg [63:0] stored6;
    reg [63:0] stored7;
    reg [63:0] stored8;
    reg [63:0] stored9;
    reg [63:0] stored10;
    reg [63:0] stored11;
    reg [63:0] stored12;
    reg [63:0] stored13;
    reg [63:0] stored14;
    reg [63:0] stored15;
    reg [63:0] stored16;
    reg [63:0] stored17;
    reg [63:0] stored18;
    reg [63:0] stored19;
    reg [63:0] stored20;
    reg [63:0] stored21;
    reg [63:0] stored22;
    reg [63:0] stored23;
    reg [63:0] stored24;
    reg [63:0] stored25;
    reg [63:0] stored26;
    reg [63:0] stored27;
    reg [63:0] stored28;
    reg [63:0] stored29;
    reg [63:0] stored30;
```

```
reg [63:0] stored31;
reg [63:0] add0_1;
reg [63:0] add2_3;
reg [63:0] add4_5;
reg [63:0] add6_7;
reg [63:0] add8_9;
reg [63:0] add10_11;
reg [63:0] add12_13;
reg [63:0] add14_15;
reg [63:0] add16_17;
reg [63:0] add18_19;
reg [63:0] add20_21;
reg [63:0] add22_23;
reg [63:0] add24_25;
reg [63:0] add26_27;
reg [63:0] add28_29;
reg [63:0] add30_31;
reg [63:0] add0t1_2t3;
reg [63:0] add4t5_6t7;
reg [63:0] add8t9_10t11;
reg [63:0] add12t13_14t15;
reg [63:0] add16t17_18t19;
reg [63:0] add20t21_22t23;
reg [63:0] add24t25_26t27;
reg [63:0] add28t29_30t31;
reg [63:0] add0t3_4t7;
reg [63:0] add8t11_12t15;
reg [63:0] add16t19_20t23;
reg [63:0] add24t27_28t31;
reg [63:0] add0t7_8t15;
reg [63:0] add16t23_24t31;
```

```
always @(posedge clk or posedge reset) begin
if (reset) begin
        temp <= 0;
        stored0 <= 0;
        stored1 <= 0;
        stored2 <= 0;
        stored3 <= 0;
        stored4 <= 0;
        stored5 <= 0;
        stored6 <= 0;
        stored7 <= 0;
        stored8 <= 0;
        stored9 <= 0;
        stored10 <= 0;
        stored11 <= 0;
        stored12 <= 0;
        stored13 <= 0;
        stored14 <= 0;
        stored15 <= 0;
        stored16 <= 0;
        stored17 <= 0;
        stored18 <= 0;
        stored19 <= 0;
        stored20 <= 0;
        stored21 <= 0;
        stored22 <= 0;
        stored23 <= 0;
        stored24 <= 0;
        stored25 <= 0;
        stored26 <= 0;
        stored27 <= 0;
        stored28 <= 0;
```

```
stored29 <= 0;
stored30 <= 0;
stored31 <= 0;
add0_1 <= 0;
add2_3 <= 0;
add4_5 <= 0;
add6_7 <= 0;
add8_9 <= 0;
add10_11 <= 0;
add12_13 <= 0;
add14_15 <= 0;
add16_17 <= 0;
add18_19 <= 0;
add20_21 <= 0;
add22_23 <= 0;
add24_25 <= 0;
add26_27 <= 0;
add28_29 <= 0;
add30_31 <= 0;

add0t1_2t3 <= 0;
add4t5_6t7 <= 0;
add8t9_10t11 <= 0;
add12t13_14t15 <= 0;
add16t17_18t19 <= 0;
add20t21_22t23 <= 0;
add24t25_26t27 <= 0;
add28t29_30t31 <= 0;

add0t3_4t7 <= 0;
add8t11_12t15 <= 0;
add16t19_20t23 <= 0;
```

```
            add24t27_28t31 <= 0;


            add0t7_8t15 <= 0;
            add16t23_24t31 <= 0;
        end else begin
            stored0 <= b[0] ? {32'b0 ,a } : 64'b0;
            stored1 <= b[1] ? {31'b0 ,a ,1'b0} : 64'b0;
            stored2 <= b[2] ? {30'b0 ,a ,2'b0} : 64'b0;
            stored3 <= b[3] ? {29'b0 ,a ,3'b0} : 64'b0;
            stored4 <= b[4] ? {28'b0 ,a ,4'b0} : 64'b0;
            stored5 <= b[5] ? {27'b0 ,a ,5'b0} : 64'b0;
            stored6 <= b[6] ? {26'b0 ,a ,6'b0} : 64'b0;
            stored7 <= b[7] ? {25'b0 ,a ,7'b0} : 64'b0;
            stored8 <= b[8] ? {24'b0 ,a ,8'b0} : 64'b0;
            stored9 <= b[9] ? {23'b0 ,a ,9'b0} : 64'b0;
            stored10 <= b[10] ? {22'b0 ,a ,10'b0} : 64'b0;
            stored11 <= b[11] ? {21'b0 ,a ,11'b0} : 64'b0;
            stored12 <= b[12] ? {20'b0 ,a ,12'b0} : 64'b0;
            stored13 <= b[13] ? {19'b0 ,a ,13'b0} : 64'b0;
            stored14 <= b[14] ? {18'b0 ,a ,14'b0} : 64'b0;
            stored15 <= b[15] ? {17'b0 ,a ,15'b0} : 64'b0;
            stored16 <= b[16] ? {16'b0 ,a ,16'b0} : 64'b0;
            stored17 <= b[17] ? {15'b0 ,a ,17'b0} : 64'b0;
            stored18 <= b[18] ? {14'b0 ,a ,18'b0} : 64'b0;
            stored19 <= b[19] ? {13'b0 ,a ,19'b0} : 64'b0;
            stored20 <= b[20] ? {12'b0 ,a ,20'b0} : 64'b0;
            stored21 <= b[21] ? {11'b0 ,a ,21'b0} : 64'b0;
            stored22 <= b[22] ? {10'b0 ,a ,22'b0} : 64'b0;
            stored23 <= b[23] ? {9'b0 ,a ,23'b0} : 64'b0;
            stored24 <= b[24] ? {8'b0 ,a ,24'b0} : 64'b0;
            stored25 <= b[25] ? {7'b0 ,a ,25'b0} : 64'b0;
            stored26 <= b[26] ? {6'b0 ,a ,26'b0} : 64'b0;
```

```
stored27 <= b[27] ? {5'b0 ,a ,27'b0} : 64'b0;
stored28 <= b[28] ? {4'b0 ,a ,28'b0} : 64'b0;
stored29 <= b[29] ? {3'b0 ,a ,29'b0} : 64'b0;
stored30 <= b[30] ? {2'b0 ,a ,30'b0} : 64'b0;
stored31 <= b[31] ? {1'b0 ,a ,31'b0} : 64'b0;

add0_1 <= stored0 + stored1;
add2_3 <= stored2 + stored3;
add4_5 <= stored4 + stored5;
add6_7 <= stored6 + stored7;
add8_9 <= stored8 + stored9;
add10_11 <= stored10 + stored11;
add12_13 <= stored12 + stored13;
add14_15 <= stored14 + stored15;
add16_17 <= stored16 + stored17;
add18_19 <= stored18 + stored19;
add20_21 <= stored20 +stored21;
add22_23 <= stored22 + stored23;
add24_25 <= stored24 + stored25;
add26_27 <= stored26 + stored27;
add28_29 <= stored28 + stored29;
add30_31 <= stored30 + stored31;

add0t1_2t3 <= add0_1 + add2_3;
add4t5_6t7 <= add4_5 + add6_7;
add8t9_10t11 <= add8_9 + add10_11;
add12t13_14t15 <= add12_13 + add14_15;
add16t17_18t19 <= add16_17 + add18_19;
add20t21_22t23 <= add20_21 + add22_23;
add24t25_26t27 <= add24_25 + add26_27;
add28t29_30t31 <= add28_29 + add30_31;
```

```
                add0t3_4t7 <= add0t1_2t3 + add4t5_6t7;
                add8t11_12t15 <= add8t9_10t11 + add12t13_14t15;
                add16t19_20t23 <= add16t17_18t19 + add20t21_22t23;
                add24t27_28t31 <= add24t25_26t27 + add28t29_30t31;

                add0t7_8t15 <= add0t3_4t7 + add8t11_12t15;
                add16t23_24t31 <= add16t19_20t23 + add24t27_28t31;

                temp <= add0t7_8t15 + add16t23_24t31;
        end
        end
        assign z = temp;
endmodule
```

```
`timescale 1ns / 1ps
//MUL.v
module MUL(
    input clk,
    input reset,
    input [31:0] a,
    input [31:0] b,
    output [63:0] z
    );
    reg [63:0] temp;
    reg [63:0] stored0;
    reg [63:0] stored1;
    reg [63:0] stored2;
    reg [63:0] stored3;
    reg [63:0] stored4;
    reg [63:0] stored5;
    reg [63:0] stored6;
    reg [63:0] stored7;
    reg [63:0] stored8;
```

```
    reg [63:0] stored9;
    reg [63:0] stored10;
    reg [63:0] stored11;
    reg [63:0] stored12;
    reg [63:0] stored13;
    reg [63:0] stored14;
    reg [63:0] stored15;
    reg [63:0] stored16;
    reg [63:0] stored17;
    reg [63:0] stored18;
    reg [63:0] stored19;
    reg [63:0] stored20;
    reg [63:0] stored21;
    reg [63:0] stored22;
    reg [63:0] stored23;
    reg [63:0] stored24;
    reg [63:0] stored25;
    reg [63:0] stored26;
    reg [63:0] stored27;
    reg [63:0] stored28;
    reg [63:0] stored29;
    reg [63:0] stored30;
    reg [63:0] stored31;
    reg [63:0] add0_1;
    reg [63:0] add2_3;
    reg [63:0] add4_5;
    reg [63:0] add6_7;
    reg [63:0] add8_9;
    reg [63:0] add10_11;
    reg [63:0] add12_13;
    reg [63:0] add14_15;
    reg [63:0] add16_17;
```

```
reg [63:0] add18_19;
reg [63:0] add20_21;
reg [63:0] add22_23;
reg [63:0] add24_25;
reg [63:0] add26_27;
reg [63:0] add28_29;
reg [63:0] add30_31;
reg [63:0] add0t1_2t3;
reg [63:0] add4t5_6t7;
reg [63:0] add8t9_10t11;
reg [63:0] add12t13_14t15;
reg [63:0] add16t17_18t19;
reg [63:0] add20t21_22t23;
reg [63:0] add24t25_26t27;
reg [63:0] add28t29_30t31;
reg [63:0] add0t3_4t7;
reg [63:0] add8t11_12t15;
reg [63:0] add16t19_20t23;
reg [63:0] add24t27_28t31;
reg [63:0] add0t7_8t15;
reg [63:0] add16t23_24t31;
reg [31:0] aa, bb;

always @(posedge clk or negedge reset)  begin
    if(reset)  begin
            stored0 <= 0;
            stored1 <= 0;
            stored2 <= 0;
            stored3 <= 0;
            stored4 <= 0;
            stored5 <= 0;
            stored6 <= 0;
```

```
stored7 <= 0;
stored8 <= 0;
stored9 <= 0;
stored10 <= 0;
stored11 <= 0;
stored12 <= 0;
stored13 <= 0;
stored14 <= 0;
stored15 <= 0;
stored16 <= 0;
stored17 <= 0;
stored18 <= 0;
stored19 <= 0;
stored20 <= 0;
stored21 <= 0;
stored22 <= 0;
stored23 <= 0;
stored24 <= 0;
stored25 <= 0;
stored26 <= 0;
stored27 <= 0;
stored28 <= 0;
stored29 <= 0;
stored30 <= 0;
stored31 <= 0;
add0_1 <= 0;
add2_3 <= 0;
add4_5 <= 0;
add6_7 <= 0;
add8_9 <= 0;
add10_11 <= 0;
add12_13 <= 0;
```

```
            add14_15 <= 0;
            add16_17 <= 0;
            add18_19 <= 0;
            add20_21 <= 0;
            add22_23 <= 0;
            add24_25 <= 0;
            add26_27 <= 0;
            add28_29 <= 0;
            add30_31 <= 0;
            add0t1_2t3 <= 0;
            add4t5_6t7 <= 0;
            add8t9_10t11 <= 0;
            add12t13_14t15 <= 0;
            add16t17_18t19 <= 0;
            add20t21_22t23 <= 0;
            add24t25_26t27 <= 0;
            add28t29_30t31 <= 0;
            add0t3_4t7 <= 0;
            add8t11_12t15 <= 0;
            add16t19_20t23 <= 0;
            add24t27_28t31 <= 0;
            add0t7_8t15 <= 0;
            add16t23_24t31 <= 0;
            aa <= 0;
            bb <= 0;
            temp <= 0;
        end else begin
            aa <= a[31] ? (~a + 1'b1) : a;
            bb <= b[31] ? (~b + 1'b1) : b;

            stored0 <= bb[0] ? {32'b0, aa} : 64'b0;
            stored1 <= bb[1] ? {31'b0, aa, 1'b0} : 64'b0;
```

```
stored2 <= bb[2] ? {30'b0, aa, 2'b0} : 64'b0;
stored3 <= bb[3] ? {29'b0, aa, 3'b0} : 64'b0;
stored4 <= bb[4] ? {28'b0, aa, 4'b0} : 64'b0;
stored5 <= bb[5] ? {27'b0, aa, 5'b0} : 64'b0;
stored6 <= bb[6] ? {26'b0, aa, 6'b0} : 64'b0;
stored7 <= bb[7] ? {25'b0, aa, 7'b0} : 64'b0;
stored8 <= bb[8] ? {24'b0, aa, 8'b0} : 64'b0;
stored9 <= bb[9] ? {23'b0, aa, 9'b0} : 64'b0;
stored10 <= bb[10] ? {22'b0, aa, 10'b0} : 64'b0;
stored11 <= bb[11] ? {21'b0, aa, 11'b0} : 64'b0;
stored12 <= bb[12] ? {20'b0, aa, 12'b0} : 64'b0;
stored13 <= bb[13] ? {19'b0, aa, 13'b0} : 64'b0;
stored14 <= bb[14] ? {18'b0, aa, 14'b0} : 64'b0;
stored15 <= bb[15] ? {17'b0, aa, 15'b0} : 64'b0;
stored16 <= bb[16] ? {16'b0, aa, 16'b0} : 64'b0;
stored17 <= bb[17] ? {15'b0, aa, 17'b0} : 64'b0;
stored18 <= bb[18] ? {14'b0, aa, 18'b0} : 64'b0;
stored19 <= bb[19] ? {13'b0, aa, 19'b0} : 64'b0;
stored20 <= bb[20] ? {12'b0, aa, 20'b0} : 64'b0;
stored21 <= bb[21] ? {11'b0, aa, 21'b0} : 64'b0;
stored22 <= bb[22] ? {10'b0, aa, 22'b0} : 64'b0;
stored23 <= bb[23] ? {9'b0, aa, 23'b0} : 64'b0;
stored24 <= bb[24] ? {8'b0, aa, 24'b0} : 64'b0;
stored25 <= bb[25] ? {7'b0, aa, 25'b0} : 64'b0;
stored26 <= bb[26] ? {6'b0, aa, 26'b0} : 64'b0;
stored27 <= bb[27] ? {5'b0, aa, 27'b0} : 64'b0;
stored28 <= bb[28] ? {4'b0, aa, 28'b0} : 64'b0;
stored29 <= bb[29] ? {3'b0, aa, 29'b0} : 64'b0;
stored30 <= bb[30] ? {2'b0, aa, 30'b0} : 64'b0;
stored31 <= bb[31] ? {1'b0, aa, 31'b0} : 64'b0;


add0_1 <= stored0 + stored1;
```

```
            add2_3 <= stored2 + stored3;
            add4_5 <= stored4 + stored5;
            add6_7 <= stored6 + stored7;
            add8_9 <= stored8 + stored9;
            add10_11 <= stored10 + stored11;
            add12_13 <= stored12 + stored13;
            add14_15 <= stored14 + stored15;
            add16_17 <= stored16 + stored17;
            add18_19 <= stored18 + stored19;
            add20_21 <= stored20 + stored21;
            add22_23 <= stored22 + stored23;
            add24_25 <= stored24 + stored25;
            add26_27 <= stored26 + stored27;
            add28_29 <= stored28 + stored29;
            add30_31 <= stored30 + stored31;

            add0t1_2t3 <= add0_1 + add2_3;
            add4t5_6t7 <= add4_5 + add6_7;
            add8t9_10t11 <= add8_9 + add10_11;
            add12t13_14t15 <= add12_13 + add14_15;
            add16t17_18t19 <= add16_17 + add18_19;
            add20t21_22t23 <= add20_21 + add22_23;
            add24t25_26t27 <= add24_25 + add26_27;
            add28t29_30t31 <= add28_29 + add30_31;
            add0t3_4t7 <= add0t1_2t3 + add4t5_6t7;
            add8t11_12t15 <= add8t9_10t11 + add12t13_14t15;
            add16t19_20t23 <= add16t17_18t19 + add20t21_22t23;
            add24t27_28t31 <= add24t25_26t27 + add28t29_30t31;
            add0t7_8t15 <= add0t3_4t7 + add8t11_12t15;
            add16t23_24t31 <= add16t19_20t23 + add24t27_28t31;
            temp <= add0t7_8t15 + add16t23_24t31;
        end
```

```
    end
    assign z = (a[31] == b[31]) ? temp : ~temp + 1'b1;
endmodule
```

三、 测试模块建模

3.1 测试思路

首先通过 8 组既含边界条件又含有正常条件的测试案例得到结果，然后与 *VIVADO* 自带的无符号、有符号乘法器所得结果进行对比，最后即可验证编写的程序的正确与否。

3.2 测试代码

```
`timescale 1ns / 1ps
module multu_tb;
        reg clk = 0;
        reg reset = 0;
        reg [31 : 0] a, b;
        wire [63 : 0] c;
        wire [63 : 0] rc;

        MULTU uut(.clk(clk), .reset(reset), .a(a), .b(b), .z(c));
    assign rc = a * b;

        always #5 clk = ~clk;
        initial begin
                reset <= 'b1;
                a <= 'b0;
                b <= 'b0;
                #5 reset <= 'b0;
                #30 a <= 'b0;
                b <= 'b11111111111111111111111111111111;
                #30 a <= 'b11111111111111111111111111111111;
                b <= 'b0;
```

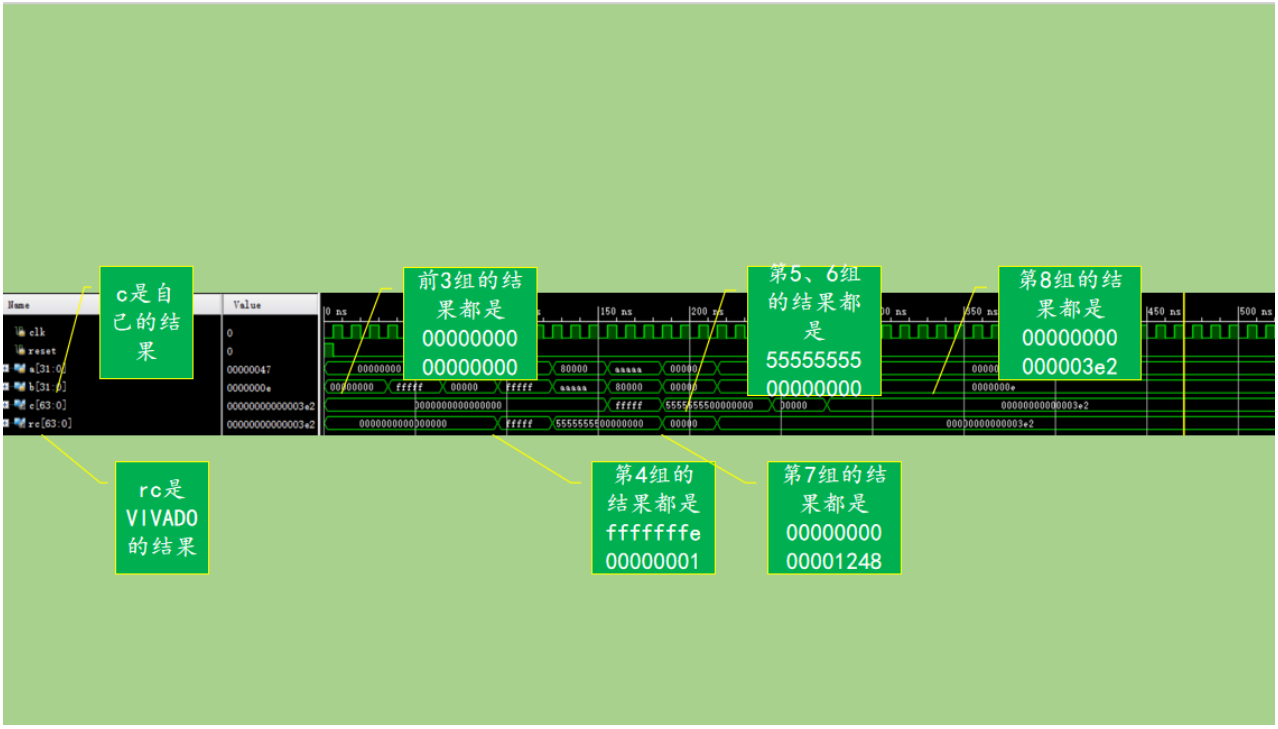
```
                #30 a <= 'b11111111111111111111111111111111;
                b <= 'b11111111111111111111111111111111;
                #30 a <= 'b10000000000000000000000000000000;
                b <= 'b10101010101010101010101010101010;
                #30 a <= 'b10101010101010101010101010101010;
                b <= 'b10000000000000000000000000000000;
                #30 a <= 'b101101;
                b <= 'b1101000;
                #30 a <= 'b1000111;
                b <= 'b1110;
        end
endmodule
```

```
`timescale 1ns / 1ps
module mul_tb;
        reg clk = 0;
        reg reset = 0;
        reg [31 : 0] a, b;
        wire [63 : 0] c;
        wire [63 : 0] rc;

        MUL uut(.clk(clk), .reset(reset), .a(a), .b(b), .z(c));
    assign rc = $signed(a) * $signed(b);

        always #5 clk = ~clk;
        initial begin
                reset <= 'b1;
                a <= 'b0;
                b <= 'b0;
                #5 reset <= 'b0;
                #30 a <= 'b0;
                b <= 'b11111111111111111111111111111111;
                #30 a <= 'b11111111111111111111111111111111;
```

```
            b <= 'b0;
            #30 a <= 'b11111111111111111111111111111111;
            b <= 'b11111111111111111111111111111111;
            #30 a <= 'b10000000000000000000000000000000;
            b <= 'b10101010101010101010101010101010;
            #30 a <= 'b10101010101010101010101010101010;
            b <= 'b10000000000000000000000000000000;
            #30 a <= 'b101101;
            b <= 'b1101000;
            #30 a <= 'b1000111;
            b <= 'b1110;
        end
endmodule
```

四、 实验结果

4.1 无符号乘法

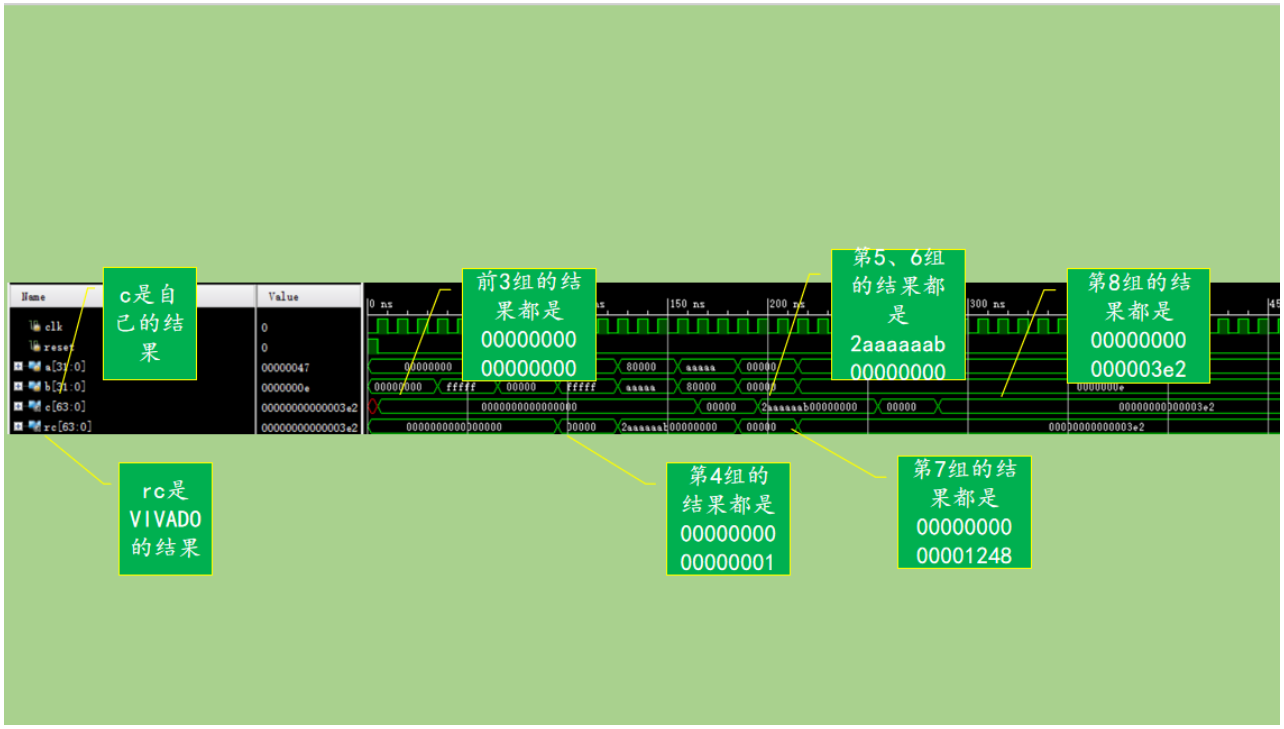


a、b 分别代表被乘数和乘数，c 代表自己编写的模块运算结果，rc 代表 $VIVADO$ 自带函数运算结果。

	a	b	c	rc
第 1 组	00000000	00000000	0000000000000000	0000000000000000
第 2 组	00000000	$ffffffff$	0000000000000000	0000000000000000
第 3 组	$ffffffff$	00000000	0000000000000000	0000000000000000
第 4 组	$ffffffff$	$ffffffff$	$fffffffe$00000001	$fffffffe$00000001
第 5 组	80000000	$aaaaaaaa$	5555555500000000	5555555500000000
第 6 组	$aaaaaaaa$	80000000	5555555500000000	5555555500000000
第 7 组	0000002d	00000068	0000000000001248	0000000000001248
第 8 组	00000047	0000000e	00000000000003e2	00000000000003e2

表 1: 无符号乘法

4.2 有符号乘法



a、b 分别代表被乘数和乘数，c 代表自己编写的模块运算结果，rc 代表 $VIVADO$ 自带函数运算结果。

	a	b	c	rc
第 1 组	00000000	00000000	0000000000000000	0000000000000000
第 2 组	00000000	$ffffffff$	0000000000000000	0000000000000000
第 3 组	$ffffffff$	00000000	0000000000000000	0000000000000000
第 4 组	$ffffffff$	$ffffffff$	0000000000000001	0000000000000001
第 5 组	80000000	$aaaaaaaa$	2$aaaaaab$00000000	2$aaaaaab$00000000
第 6 组	$aaaaaaaa$	80000000	2$aaaaaab$00000000	2$aaaaaab$00000000
第 7 组	0000002d	00000068	0000000000001248	0000000000001248
第 8 组	00000047	0000000e	00000000000003e2	00000000000003e2

表 2: 有符号乘法