组成原理课程第 1 次实验报告

实验名称：32 位减法器

学号：2312141 姓名：张德民 班次：李涛老师班

一、 实验目的

1. 熟悉 LS-CPU-EXB-002 实验箱和软件平台。
2. 掌握利用该实验箱各项功能开发组成原理和体系结构实验的方法。
3. 理解并掌握加法器的原理和设计。
4. 熟悉并运用 verilog 语言进行电路设计。
5. 为后续设计 cpu 的实验打下基础。

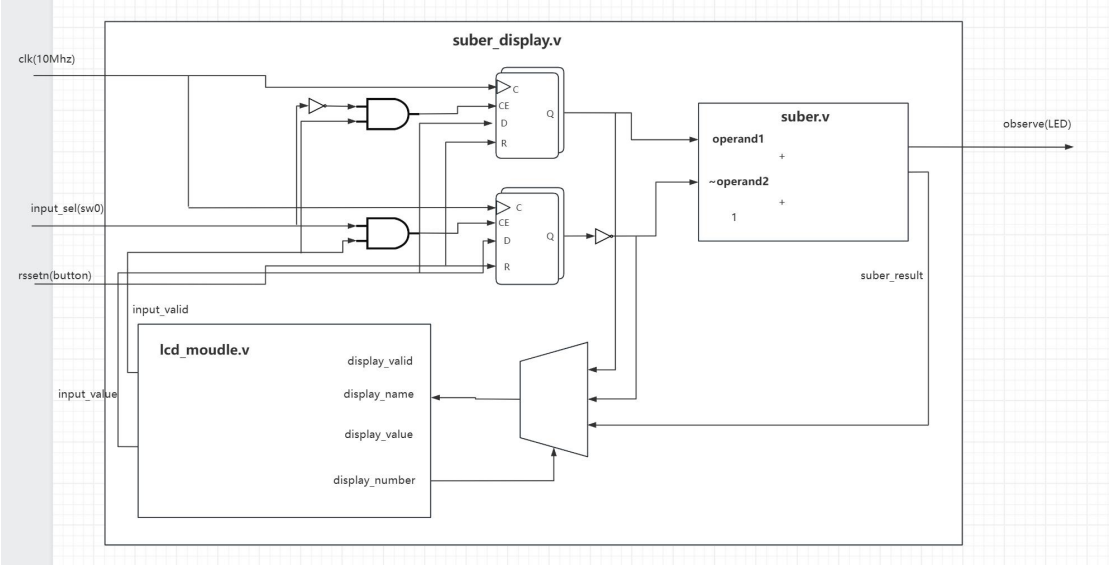
二、 实验内容说明

本次实验内容主要为根据第一次定点加法器实验进行学习改进，实现一个减法器，该减法器要求如下：

减法器拿到的 32 位输入都应该理解成数据的补码形式，要进行补码运算，最终得到差的补码进行输出。最后得到仿真结果。

在实验箱上进行测试，验证结果的正确性。

三、 实验原理图



四、 实验步骤

1. 模块 suber32 的实现

代码思路：

首先，我们需要实现减法器的代码，原代码是 32 位的加法，而减法不能简单的把加法改为减法，因此我们考虑在数字逻辑课程里学到的知识，使用补码重新把减法改为加法。并且原代码里有 cin 和 cout 分别代表上一位的进位和向下一位的进位，在减法中应该改为上一位的借位和向下一位的借位。不过考虑到在有符号计算中（也就是答案可能是负数），向下一位的借位似乎没有什么意义，因此我的减法器模块中删除了原来的 cin 和 cout，加入了一个检测位 observe，这个的作用就是检测最后结果是正数还是负数。

代码：

```
module suber32(
    input[31:0] op1,
    input[31:0] op2,
    output[31:0] result,
    output observe
    );
    wire [32:0] temp_result;

    assign temp_result=op1+~op2+1'b1;
    assign result=temp_result[31:0];
    assign observe=temp_result[31];
endmodule
```

修改部分：

1. 首先，我将 input 部分的 cin 与 ouput 部分的 cout 删除，因为在有符号整数减法中，这两个部分并没有什么意义。

```
input[31:0] op1,
input[31:0] op2,
```

2. 对于结果的计算是本代码的核心，我使用补码将减法修改为加法，即 op1-op2=op1+~op2+1,对 op2 按位取反并加一。

```
assign temp_result=op1+~op2+1'b1;
```

3. 然后我添加了检测为 observe，这一位等于结果 result 的最高位，如果结果为正，那么 observe 就是 0，如果为负数，那么就是 1。

```
assign observe=temp_result[31];
```

2. 模块 suber_display 的实现

代码思路：

这一部分的代码和原代码基本上是一致的，对于一些删除与修改的 input 和 output 量还有其名字进行了一些修改。

代码：

```
module suber_display(
    //时钟与复位信号
     input clk,
    input resetn,       //后缀"n"代表低电平有效

    //拨码开关，用于选择输入数和产生 cin
    input  input_sel,  //0: 输 入 为 被 减 数  1(add_operand1);1: 输 入 为 减 数 2(add_operand2)
    input sw_cin,

    //led 灯，用于显示 cout
    output led_cout,

    //触摸屏相关接口，不需要更改
    output lcd_rst,
    output lcd_cs,
    output lcd_rs,
    output lcd_wr,
    output lcd_rd,
    inout[15:0] lcd_data_io,
    output lcd_bl_ctr,
    inout ct_int,
    inout ct_sda,
    output ct_scl,
    output ct_rstn
    );

//-----{调用减法模块}begin
    reg   [31:0] suber_operand1;
    reg   [31:0] suber_operand2;
    wire [31:0] suber_result;
    wire observe;
    suber32 suberer_module(
        .op1(suber_operand1),
        .op2(suber_operand2),
        .result   (suber_result   ),
        .observe(observe)
    );
    assign suber_cin = sw_cin;
//-----{调用减法模块}end

//---------------------{调用触摸屏模块}begin--------------------//
//-----{实例化触摸屏}begin
//此小节不需要更改
```

```
    reg             display_valid;
    reg   [39:0] display_name;
    reg   [31:0] display_value;
    wire [5 :0] display_number;
    wire            input_valid;
    wire [31:0] input_value;

    lcd_module lcd_module(
        .clk                (clk              ),    //10Mhz
        .resetn             (resetn           ),

        //调用触摸屏的接口
        .display_valid   (display_valid ),
        .display_name    (display_name  ),
        .display_value   (display_value ),
        .display_number (display_number),
        .input_valid      (input_valid    ),
        .input_value      (input_value    ),

        //lcd 触摸屏相关接口，不需要更改
        .lcd_rst          (lcd_rst        ),
        .lcd_cs           (lcd_cs         ),
        .lcd_rs           (lcd_rs         ),
        .lcd_wr           (lcd_wr         ),
        .lcd_rd           (lcd_rd         ),
        .lcd_data_io      (lcd_data_io    ),
        .lcd_bl_ctr       (lcd_bl_ctr     ),
        .ct_int           (ct_int         ),
        .ct_sda           (ct_sda         ),
        .ct_scl           (ct_scl         ),
        .ct_rstn          (ct_rstn        )
    );
//-----{实例化触摸屏}end

//-----{从触摸屏获取输入}begin
//根据实际需要输入的数修改此小节，
//建议对每一个数的输入，编写单独一个 always 块
    //当 input_sel 为 0 时，表示输入数为被减数 1，即 operand1
    always @(posedge clk)
    begin
        if (!resetn)
        begin
            suber_operand1 <= 32'd0;
        end
```

```
            else if (input_valid && !input_sel)
            begin
                suber_operand1 <= input_value;
            end
        end

        //当 input_sel 为 1 时，表示输入数为减数 2，即 operand2
        always @(posedge clk)
        begin
            if (!resetn)
            begin
                suber_operand2 <= 32'd0;
            end
            else if (input_valid && input_sel)
            begin
                suber_operand2 <= input_value;
            end
        end
//-----{从触摸屏获取输入}end

//-----{输出到触摸屏显示}begin
//根据需要显示的数修改此小节，
//触摸屏上共有 44 块显示区域，可显示 44 组 32 位数据
//44 块显示区域从 1 开始编号，编号为 1~44，
        always @(posedge clk)
        begin
            case(display_number)
                6'd1 :
                begin
                    display_valid <= 1'b1;
                    display_name    <= "SUB_1";
                    display_value <= suber_operand1;
                end
                6'd2 :
                begin
                    display_valid <= 1'b1;
                    display_name    <= "SUB_2";
                    display_value <= suber_operand2;
                end
                6'd3 :
                begin
                    display_valid <= 1'b1;
                    display_name    <= "RESUL";
                    display_value <= suber_result;
```

```
                end
                 6'd4 :
                begin
                      display_valid <= 1'b1;
                      display_name    <= "OBSERVE";
                      display_value <= observe;
                end

                default :
                begin
                      display_valid <= 1'b0;
                      display_name    <= 40'd0;
                      display_value <= 32'd0;
                end
            endcase
        end
    //-----{输出到触摸屏显示}end
    //---------------------{调用触摸屏模块}end--------------------//
    endmodule
```

修改部分：

1. 首先我修改了一些名字，把原来的 add 改为 sub，改了参数的量，删除了原来的 cin 与 cout，添加了 observe

```
//------{调用减法模块}begin
    reg  [31:0] suber_operand1;
    reg  [31:0] suber_operand2;
    wire [31:0] suber_result;
    wire observe;
    suber32 suberer_module(
        .op1(suber_operand1),
        .op2(suber_operand2),
        .result   (suber_result   ),
        .observe(observe)
    );
```

2. 接下来要改的是触摸屏，我要添加一个新的名字为 OBSERVE 的模块，用来展示 observe 检测值，在第四块添加即可，格式与之前类似。还有把之前的 ADD_1 和 ADD_2 改为 SUB_1 和 SUB_2。

```
6'd4 :
begin
    display_valid <= 1'b1;
    display_name   <= "OBSERVE";
    display_value <= observe;
end
```

3.testbench 模块的实现

代码思路：

这一部分只需要修改一下调用的 suber32 模块的参数名以及参数量就可以。

代码：

```
module testbench();
reg[31:0] dataA,dataB;
wire[31:0] result;
wire observe;
suber32 uut(.op1(dataA),.op2(dataB),
            .result(result),
            .observe(observe));
initial begin
dataA=0;dataB=0;
end
always #3 dataA=$random;
always #5 dataB=$random;
endmodule
```

修改部分：

1. 修改调用 suber32 模块的时候传入的参数，删除 cin 和 cout，加入 observe 即可。

4.constraints 模块的实现

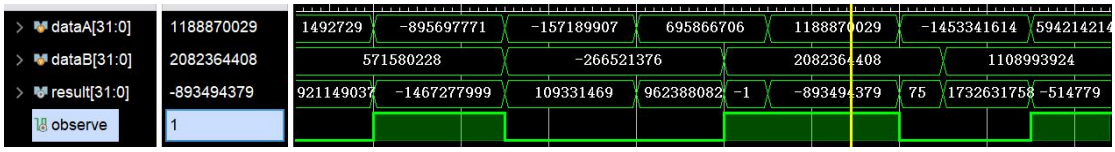
这一部分没有什么变化，使用之前的限制文件即可。

五、 实验结果分析

仿真结果分析：

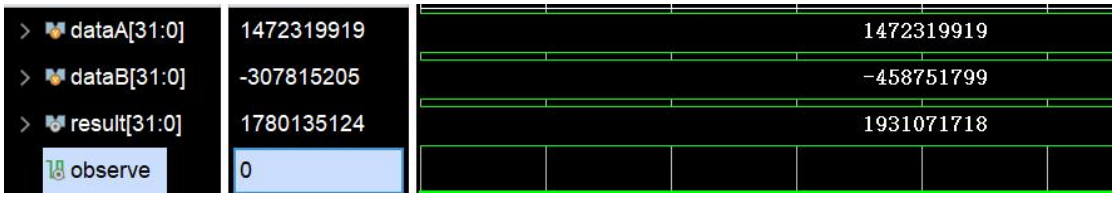
仿真结果我选取四种，即正减正，正减负，负减正，负减负。换为有符号十进制数，方便观察。

正减正:



这里被减数 11_8887_0029，减去 20_8236_4408，结果是负数，-8_9349_4379，检测位为 1。答案正确。

正减负：

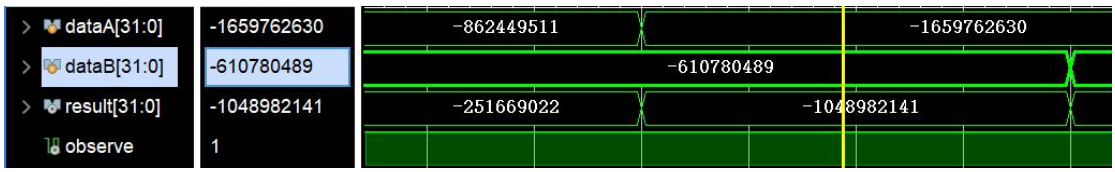


这里 14_7231_9919 减去-3_0781_5205，得到 17_8013_5124，检测位为 0，代表正数。答案正确。

负减正：

-10_9299_1875 减去 3_5362_3338，得到-13_3661_5213，检测位为 1，代表负数。正确。

负减负：

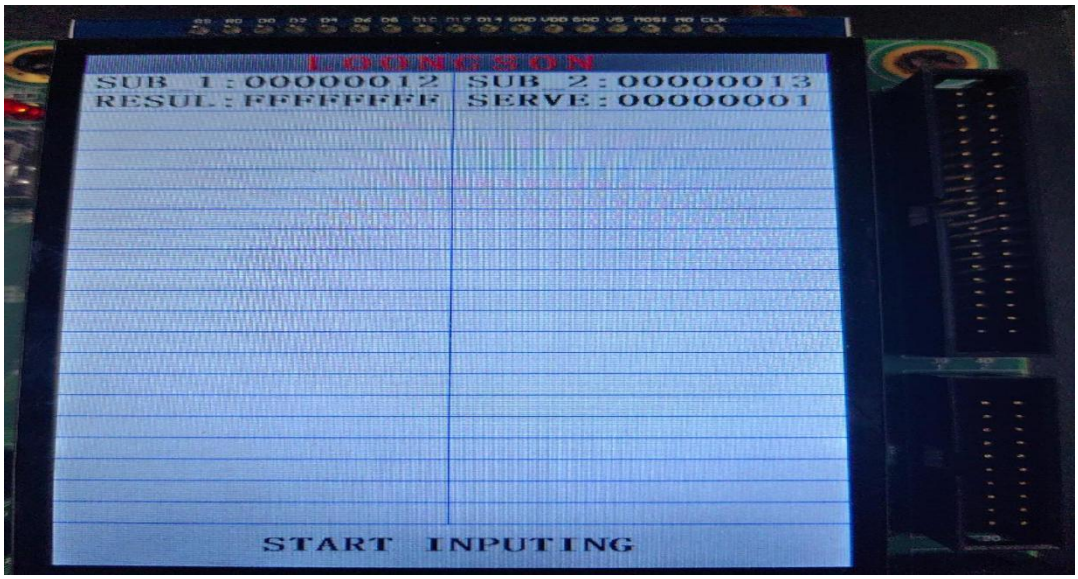


-16_5976_2630 减去-6_1078_0489，得到-10_4898_2141,检测位是 1，代表负数，正确。

实验箱验证结果：

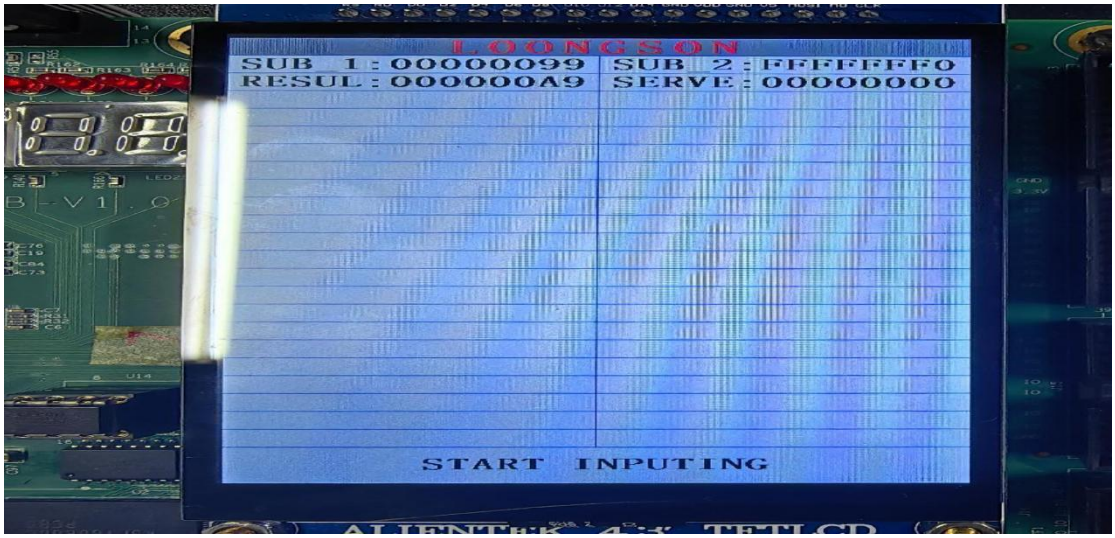
在 suber_display 模块中我给检测位命名为 OBSERVE,但是触摸屏上只有 SERVE，这应为触摸屏长度限制，导致的显示不全。

1



如上图，我输入了被减数 12h 和减数 13h，结果应该为-1，补码格式是 FFFFFFFF，检测位 1，结果正确。

2



如上图，我输入被减数 99h 和减数 FFFFFFF0h（-16），结果是 A9h，检测位是 0，结果正确。

六、 总结感想

这是计组第一次实验，这次实验里学习了 verilog 语言，并且学习使用 vivado 进行模拟仿真实验，还在实验箱上进行了测试。在改代码的过程里对 verilog 有了更深入的理解。并且在改写减法器的过程中，对于加法器的理解也更加深入了。