组成原理实验课程第__一__次实报告

实验名称	定点加法			班级	李涛老师
学生姓名	辛浩然	学号	2112514	指导老师	董前琨
实验地点	实验楼 A 区 304		实验时间	2023/3/21	

1、 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。

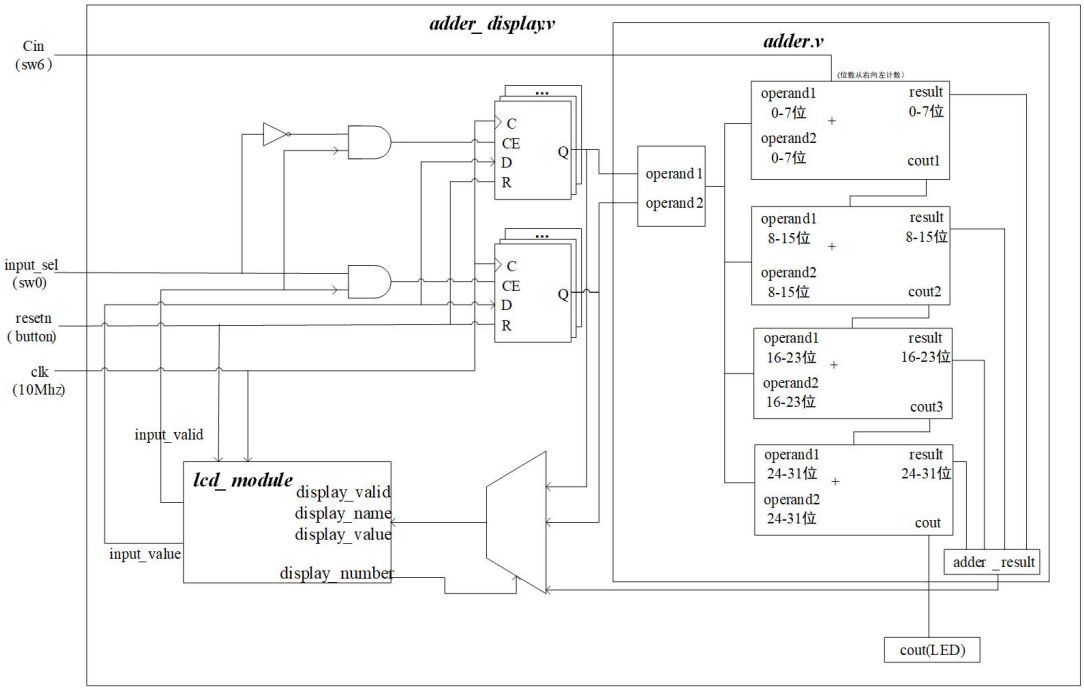
2、 实验内容说明

1.设计两个模块，8 位加法器和 32 位加法器，其中 32 位加法器通过调用 8 位加法器实现。

2.针对 32 位加法器进行仿真验证

3.针对 32 位加法器进行上实验箱验证

3、 实验原理图



4、 实验步骤

1.利用四个八位加法器实现 32 位加法

代码修改：

（1）8 位加法器模块

实现两个 8 位加数与低位进位的求和，得到 8 位结果和高位进位。

```
module adder8(
    input [7:0] ina,
```

```
    input [7:0] inb,
    input inc,
    output [7:0] sum,
    output outc
    );
    assign {outc,sum} = ina + inb + inc;
endmodule
```

（2）**32 位加法模块**，调用 8 位加法模块。实现思路：模块包含两个 32 位输入加数、32 位求和结果 res、低位进位、高位进位。对于输入的两个 32 位数，拆分成 8 位数。第一次调用 8 位加法模块，输入的两个 32 位数的后 8 位作为加数，输入的进位作为低位进位，运算和存到 res 的后八位，运算产生的进位作为下一次调用 8 位加法器的低位进位。依次进行调用，直到最后一次调用八位加法器，最后一次调用加和的结果存到 res 的前 8 位，最终的 32 位 res 就是两个 32 位数与低位进位加和的结果。产生的进位就是最终结果的进位。

```
module adder32(
    input [31:0] op1,
    input [31:0] op2,
    input cin,
    output [31:0] res,
    output cout
    );
    wire c1,c2,c3;
    adder8 adder8_1(
        .ina(op1[7:0]),
        .inb(op2[7:0]),
        .inc(cin),
        .sum(res[7:0]),
        .outc(c1)
    );
    adder8 adder8_2(
        .ina(op1[15:8]),
        .inb(op2[15:8]),
        .inc(c1),
        .sum(res[15:8]),
        .outc(c2)
    );
    adder8 adder8_3(
        .ina(op1[23:16]),
        .inb(op2[23:16]),
        .inc(c2),
        .sum(res[23:16]),
        .outc(c3)
    );
    adder8 adder8_4(
```

```
        .ina(op1[31:24]),
        .inb(op2[31:24]),
        .inc(c3),
        .sum(res[31:24]),
        .outc(cout)
    );
endmodule
```

（3）修改 **testbench 文件**

产生输入激励，即 2 个加数和 1 个低位进位信号，输出加法结果和向高位的进位信号。

```
module testbench();
    reg[31:0] op1,op2;
    reg op;
    wire[31:0] sum;
    wire flag;
    //接信号
    adder32 uut(op1,op2,op,sum,flag);
    initial //类似 c++里的构造函数 只执行一次
    begin
        op1 = 32'b0;op2 = 32'b0; op = 32'b0;
    end
    //always 语句对后面信号敏感，#3 延迟三个时间单位
    //$random 生成 32 位随机数
    always #3 op1 = $random;
    always #5 op2 = $random;
    always #7 op = $random % 2'b1_0;//取最低位
endmodule
```

2.修改 LCD 屏幕显示数据的位置

修改 display 文件输出到触摸屏显示部分

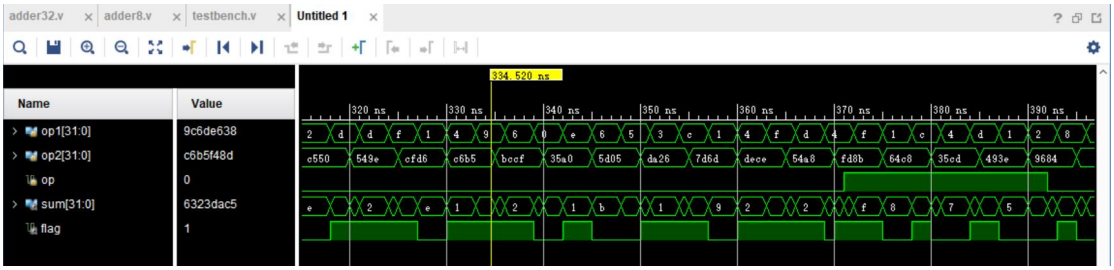
修改 case 语句中的 display_number. display_number 就是输出到外部说明当前需要显示的区域块为第几块，有效编号从 1~44，指示 44 块显示区域块。进行修改，第一块和第三块显示输入的加数，第五块显示输出的加和。当然，可以修改为 1-44 的任意数字，实现任意位置显示。

```
always @(posedge clk)
    begin
        case(display_number)
            6'd1 :  //修改
            begin
                display_valid <= 1'b1;
                display_name  <= "ADD_1";
                display_value <= adder_operand1;
            end
            6'd3 :  //修改
            begin
```

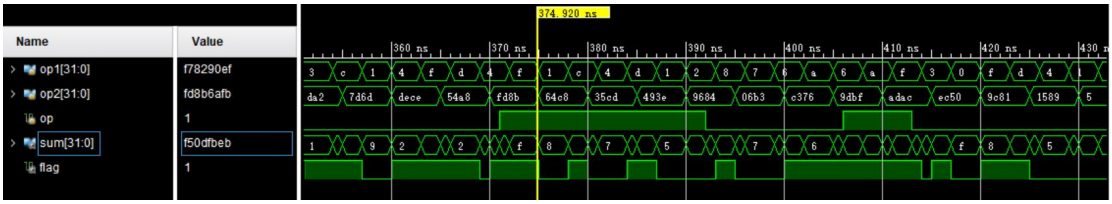
```
                display_valid <= 1'b1;
                display_name  <= "ADD_2";
                display_value <= adder_operand2;
            end
            6'd5 :  //修改
            begin
                display_valid <= 1'b1;
                display_name  <= "RESUL";
                display_value <= adder_result;
            end
            default :
            begin
                display_valid <= 1'b0;
                display_name  <= 40'd0;
                display_value <= 32'd0;
            end
        endcase
    end
```

5、 实验结果分析

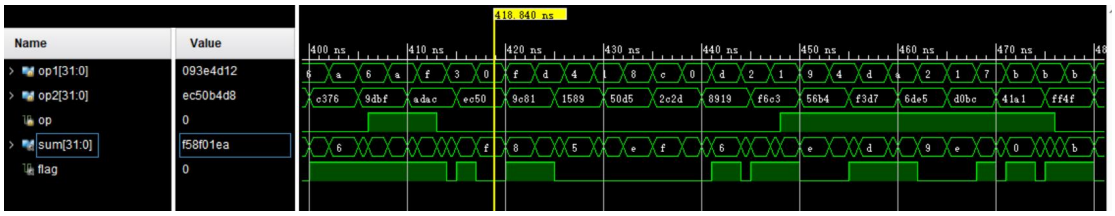
（1）仿真波形结果



c6b5f48d + 9c6de638 + 0 = 16323dac5，32 位结果为 6323dac5，高位进位为 1. 结果正确。



f78290ef + fd8b6afb + 1 = 1f50dfbeb，32 位结果为 f50dfbeb，高位进位为 1，结果正确。



093e4d12 + ec50b4d8 + 0 = f58f01ea，32 位结果为 f58f01ea，进位为 0，结果正确。

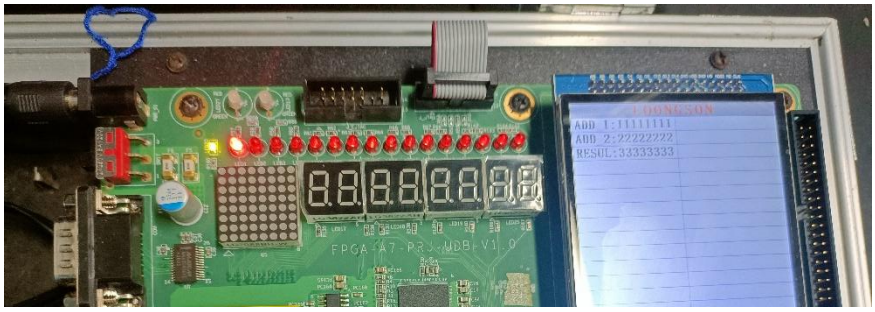
由上述仿真波形结果可验证 32 位加法器的正确性。

（2）实验箱结果

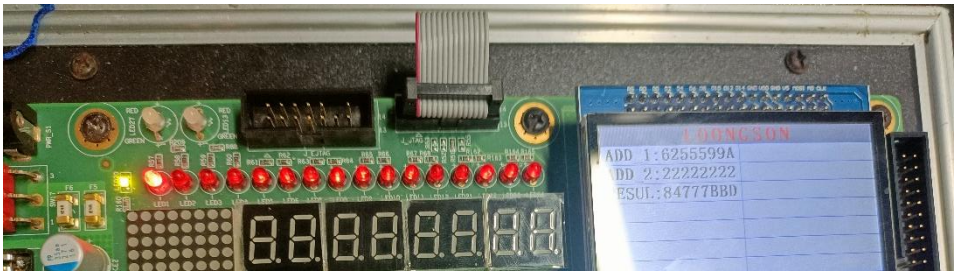
在 display 文件中，将显示位置修改为第 1、3、5 个位置，通过实验箱显示可以发现，显示位置修改正确。

在实验箱验证中，拨码开关最左侧的开关用来选择触摸屏输入的数据为加数 1 还是加数 2；左数第二个拨码开关选择输入进位为 0 还是 1.

最左侧的 led 灯为向高位的进位，进位为 0 亮，进位为 1 暗。



11111111 + 22222222 + 0 = 33333333，进位为 0（最左侧 led 灯亮），结果正确。



6255599A + 22222222 + 1 = 84777BBC，进位为 0（最左侧 led 灯亮），结果正确。

D1111111 + 92222222 + 0 = 163333333，进位为 1（最左侧 led 灯暗），结果正

确。

由上述实验箱结果可验证 32 位加法器的正确性。

6、 总结感想

1.第一次接触 Verilog 语言，对其有了基本的认知和了解，掌握了一些诸如模块、always 语句等语法。

2.熟悉了 LS-CPU-EXB-002 实验箱和软件平台。

3.掌握了利用实验箱各项功能开发组成原理和体系结构实验的方法。

4.理解并掌握了加法器的原理和设计。