组成原理课程第一次实验报告

实验名称：4 个 32 位定点加法

学号：1611736 姓名：钟腾 班次：周日

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

1. 阅读 LS-CPU-EXB-002 实验箱相关文档，熟悉硬件平台,特别需要掌握利用显示屏观察特定信号的方法。学习软件平台和设计流程。

2. 熟悉计算机中加法器的原理。

3. 设计本次实验的方案，画出原理图，构建 4 个 32 位加法器。

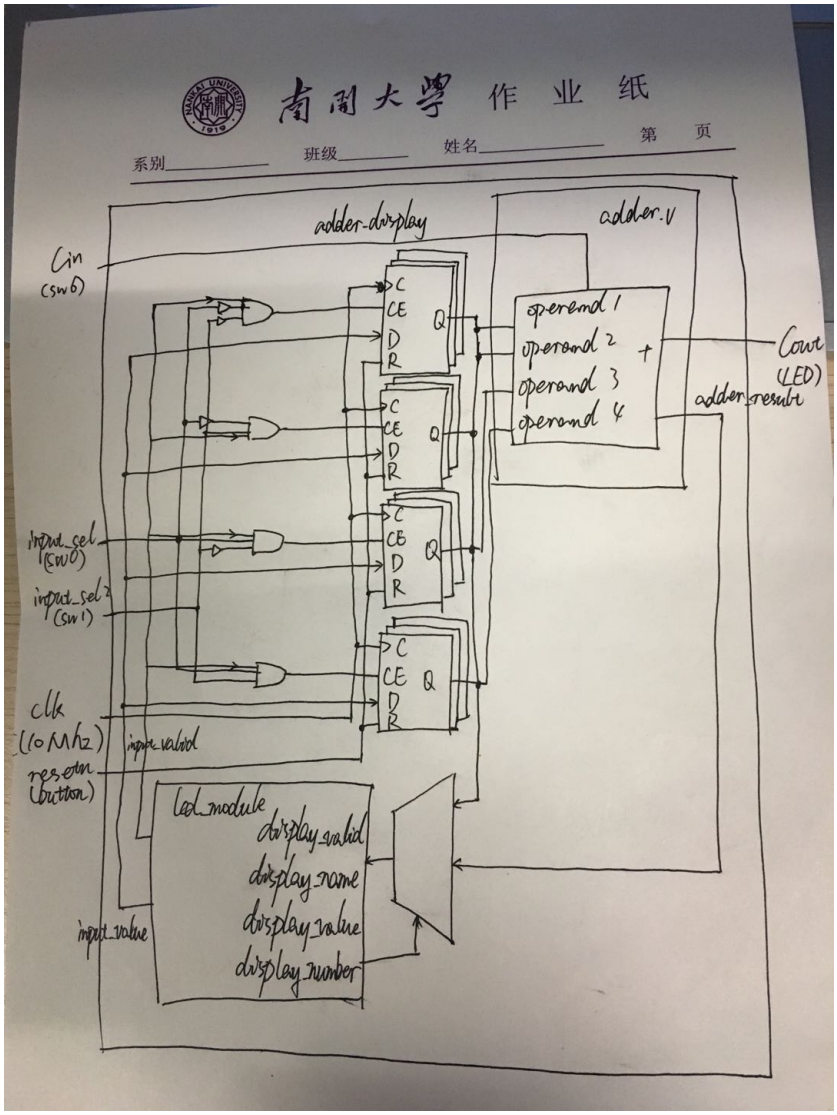
4. 根据设计的实验方案，使用 verilog 编写相应代码。

5. 对编写的代码进行仿真，得到正确的波形图。

6. 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。外围模块中需调用封装好的触摸屏模块，显示四个加数和加法结果，且需要利用触摸功能输入四个加数。

7. 将编写的代码进行综合布局布线，并下载到实验箱中的 FPGA 板上进行演示。

三、实验原理图



从图中可以看到，外围模块 adder_display 内部调用了 adder 和 lcd_module 模块。在外部除了时钟和复位信号外，还有 cin 和 input_sel、input_sel2 通过拨码开关输入，以及 cout 输出到 led 灯上。

四、实验步骤

1)新建工程

2)添加源文件，修改 adder.v 代码

```
module adder(
    input  [31:0] operand1,
    input  [31:0] operand2,
    input  [31:0] operand3,
    input  [31:0] operand4,
    input    [1:0]     cin,
    output [31:0] result,
    output        cout
    );
```

```
    assign {cout,result} = operand1 + operand2 + operand3 + operand4 + cin;

endmodule
```

代码有 4 个 32 位数的输入和 1 个进位输出，产生 1 个 32 位的加法和结果和 1 个向高位的进位。

3)添加展示外围模块

添加 adder_display.v 和 lcd_module.dcp，实现对主体代码的调用。

adder_display.v 修改部分：

```
input input_sel,
input input_sel2,
input sw_cin,
always @(posedge clk)
begin
    if (!resetn)
    begin
        adder_operand1 <= 32'd0;
    end
    else if (input_valid && !input_sel && input_sel2)
    begin
        adder_operand1 <= input_value;
    end
end

always @(posedge clk)
begin
    if (!resetn)
    begin
        adder_operand2 <= 32'd0;
    end
    else if (input_valid && input_sel && input_sel2)
    begin
        adder_operand2 <= input_value;
    end
end

always @(posedge clk)
begin
    if (!resetn)
    begin
        adder_operand3 <= 32'd0;
    end
    else if (input_valid && !input_sel && !input_sel2)
```

```
    begin
        adder_operand3 <= input_value;
    end
end

always @(posedge clk)
begin
    if (!resetn)
    begin
        adder_operand4 <= 32'd0;
    end
    else if (input_valid && input_sel && !input_sel2)
    begin
        adder_operand4 <= input_value;
    end
end
```

```
//-----{调用加法模块}begin
    reg  [31:0] adder_operand1;
    reg  [31:0] adder_operand2;
    reg  [31:0] adder_operand3;
    reg  [31:0] adder_operand4;
    wire  [1:0]     adder_cin;
    wire [31:0] adder_result  ;
    wire        adder_cout;
    adder adder_module(
        .operand1(adder_operand1),
        .operand2(adder_operand2),
        .operand3(adder_operand3),
        .operand4(adder_operand4),
        .cin     (adder_cin     ),
        .result  (adder_result  ),
        .cout    (adder_cout    )
    );
    assign adder_cin = sw_cin;
    assign led_cout  = adder_cout;
always @(posedge clk)
begin
    case(display_number)
        6'd1 :
        begin
            display_valid <= 1'b1;
            display_name  <= "ADD_1";
```

```
            display_value <= adder_operand1;
        end
        6'd2 :
        begin
            display_valid <= 1'b1;
            display_name  <= "ADD_2";
            display_value <= adder_operand2;
        end
        6'd3 :
        begin
            display_valid <= 1'b1;
            display_name  <= "ADD_3";
            display_value <= adder_operand3;
        end
        6'd4 :
        begin
            display_valid <= 1'b1;
            display_name  <= "ADD_4";
            display_value <= adder_operand4;
        end
        6'd5 :
        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
```

4)功能仿真

添加 testbench.v 文件，产生输入激励，读出功能模块的执行结果，与预期的结果进行比较，以验证功能模块的正确性。

```
`timescale 1ns / 1ps
module testbench;

    // Inputs
    reg [31:0] operand1;
```

```
    reg [31:0] operand2;
    reg [31:0] operand3;
    reg [31:0] operand4;
    reg [1:0] cin;

    // Outputs
    wire [31:0] result;
    wire cout;
    // Instantiate the Unit Under Test (UUT)
    adder uut (
        .operand1(operand1),
        .operand2(operand2),
        .operand3(operand3),
        .operand4(operand4),
        .cin(cin),
        .result(result),
        .cout(cout)
    );
    initial begin
        // Initialize Inputs
        operand1 = 0;
        operand2 = 0;
        operand3 = 0;
        operand4 = 0;
        cin = 0;
        // Wait 100 ns for global reset to finish
        #100;
        // Add stimulus here
    end
    always #10 operand1 = $random;
    always #10 operand2 = $random;
     always #10 operand3 = $random;
    always #10 operand4 = $random;
    always #10 cin = {$random} % 2;
endmodule
```

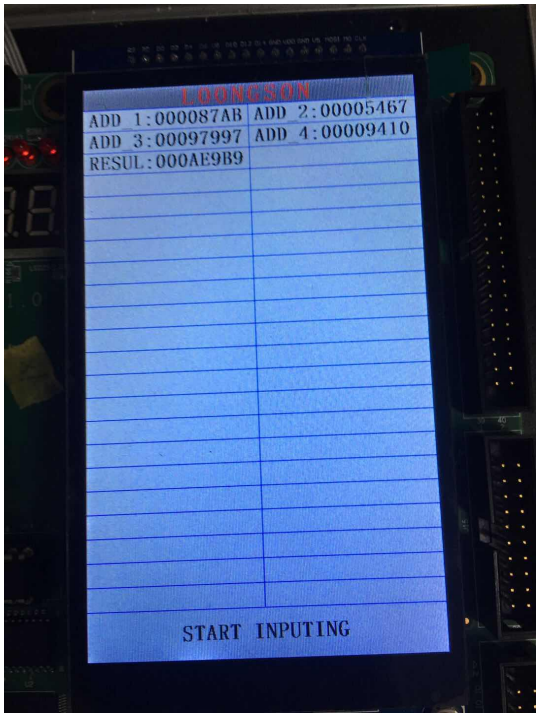
点击 Run Simulation，选择”Run Behavioral Simulation”弹出仿真界面

5）上板验证

添加约束文件 adder.xdc，设置 I/O Ports，连接 FPGA 板，完成上板验证。

五、实验结果分析

实验结果如图：



输入四个数，每个数相加再得到进位相加，得到最终结果。

验证：先验证个位，由于 B+7+7+0=25，减去一位 16 进位后为 9，与答案一致，其他数位验证也正确，因而实验顺利完成了。

六、总结感想

本次是计算机组成原理的第一次实验报告，由于不太清楚具体格式，因此在书写上存在一定的困难，尽量按照了自己理解的方式写。另外本次实验在原实验上要求上作了改变，最终也完成了实验，有不错的收获