组成原理实验课程第__一__次实报告

实验名称	数据运算：定点加法			班级	李涛老师
学生姓名	王昱	学号	2212046	指导老师	董前琨
实验地点	A306		实验时间	2024.3.21	

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. 做好预习：

1） 了解软硬件平台；

2） 掌握定点加法的工作原理；

3） 确定定点加法的输入输出端口设计；

4） 在课前画好设计框图或实验原理图；

5） 如果对 FPGA 板了解的话，可确定设计中与 FPGA 板上交互的接口，画出包含外围模块的整体设计框图，即补充完善图 2.1。

2. 实验实施：

1） 确认定点加法的设计框图的正确性；

2） 编写 verilog 代码；

3） 对该模块进行仿真，得出正确的波形，截图作为实验报告结果一项的材料；

4） 完成调用定点加法模块的外围模块的设计，并编写代码；

5） 对代码进行综合布局布线下载到实验箱里 FPGA 板上，进行上板验证。

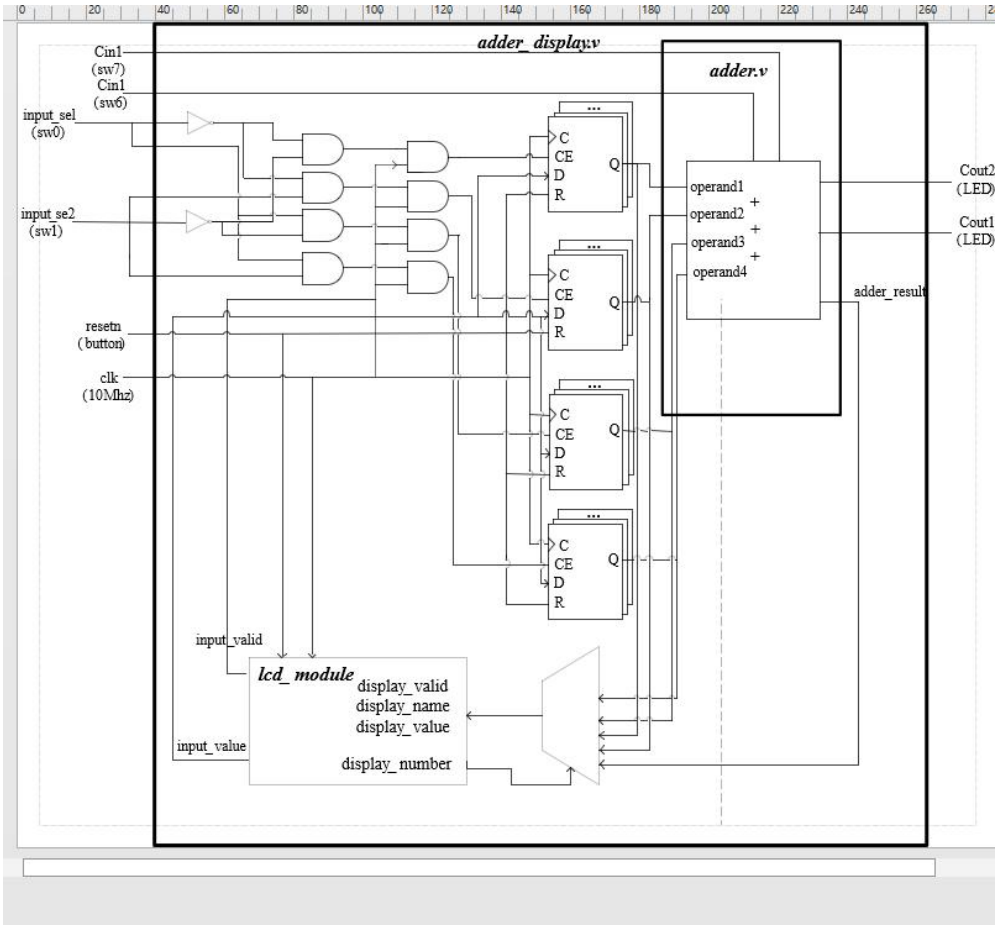
3. 实验检查：

1） 完成上板验证后，让指导老师或助教进行检查，进行现场演示，可对演示结果进行拍照作为实验报告结果一项的材料。

4. 实验报告的撰写：

1） 实验结束后，需按照规定的格式完成实验报告的撰写。

3、 实验原理图



实验原理图说明：

- 外围模块 adder_display 内部调用了 adder 和 lcd_module 模块。一共有五个输入，两个输出。由于从两个数相加改进为四个数相加，因此来自低位的进位可能有两位，故需要两个 cin 来确定有没有进位。同理也需要两个 cout 输出进位。
  - input_sel 是一个片选信号，选择输入的是第几个操作数。而修改之后因为需要有四种数值，所以增加一个拨码开关，修改逻辑门。当 00 输入 operand1，以此类推。
  - cin 和 input_sel 都是通过拨码开关输入，输出 cout 到 LED 灯上。
  - clk 是时钟信号，reset 是重置清零信号，负责清零数据。
- adder.v 模块是加法器，operand1,operand2,operand3,operand4，cin1 和 cin2 分别代表四个加数和来自低位的进位，执行相加操作后输出高位的进位 cin1,cin2 和相加的结果 adder_result，adder_result 输入到 lcd_module 模块中显示到屏幕。
- lcd_module 模块对应的是 LCD 触摸屏，既可以做输入也可以做输出，input_value 是输入的值，input_valid 是点击 ok 触发的脉冲，把输入的值存到寄存器中。display_number 选择把输入的加数和相加后的结果放到第几格。display_valid 判断是否显示，display_value 是选择显示多少位，display_name 是命名的名称。

以上就是修改后的实验原理图的简要说明。

4、 实验步骤

修改关键代码：

(1)adder.v：

```
module adder4(
    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 [1:0] cout
);
    assign {cout,result} = operand1 + operand2 + operand3 + operand4
+cin;

endmodule
```

说明：

为了实现四个 32 位数的加法，所以从两个加法器变成了四个，定义了四个 32 位的加数 oprend1-4。由于四个数可能产生两位进位，所以同时 cin 和 cout 的位宽也需要设置成两位。

(2)adder.display

```
module adder_display(
input clk,
input resetn,
input [1:0]input_sel,
input [1:0]sw_cin,
output [1:0]led_cout,
);
```

修改位宽，将 input_sel，sw_cin,led_cout 设置为两位

```
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 [1:0]          adder_cout;
    adder111 adder_module(
        .operand1(adder_operand1),
        .operand2(adder_operand2),
        .operand3(adder_operand3),
        .operand4(adder_operand4),
```

```
        .cin        (adder_cin        ),
        .result     (adder_result     ),
        .cout       (adder_cout       )
    );
```

添加两个操作数 **adder_operand3,4**

```
always @(posedge clk)
    begin
        if (!resetn)
        begin
            adder_operand1 <= 32'd0;
        end
        else if (input_valid && !input_sel)
        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==1)
        begin
            adder_operand2 <= input_value;
        end
    end
```

由于变成了四个加数，所以需要两位的 input_sel,sw_cin,led_cout，加法 **opreand** 的个数也需要随之变为四个。在判断条件中，当 **input_valid** 为 **1**，也就是输入 **ok** 的时候，同时 **input_sel** 分别为 **0**，**1**，**2**，**3**，拨码开关调整为 **0**，**1**，**2**，**3** 的时候分别输入 **operand1,2,3,4**。这样就可以分别输入四个加数，调用 **LCD** 触摸屏显示功能的代码。

```
always @(posedge clk)
    begin
        case(display_number)
            6'd1 :
            begin
                display_valid <= 1'b1;
                display_name    <= "ADD_3";
                display_value <= adder_operand3;
            end
```

为新增加的操作数分配显示区域，使其能在显示屏上正常显示，并且同时将 **result** 的显示区域向后顺延。也就是原理图中梯形的作用。

(3)testbench

```
module testbench;
    reg [31:0] operand1;
    reg [31:0] operand2;
    reg [31:0] operand3;
    reg [31:0] operand4;
    reg [1:0]cin;
    wire [31:0] result;
    wire [1:0]cout;
    adder4 uut (
        .operand1(operand1),
        .operand2(operand2),
        .operand3(operand3),
        .operand4(operand4),
        .cin(cin),
        .result(result),
        .cout(cout)
    );
```

要把两个加法器改为四个加法器，增加了两个操作数。注意 **reg** 类型才能存数。

```
initial begin
        operand1 = 0;
        operand2 = 0;
        operand3 = 0;
        operand4 = 0;
        cin = 0;
    end
    always #10 operand1 = $random;
    always #10 operand2 = $random;
    always #10 operand3 = $random;
    always #10 operand4 = $random;
    always #10 cin = {$random} % 4;
endmodule
```

这部分可以类比为 **C++**的类，需要对操作数进行初始化，之后进行随机化，由于当前的进位是两位，所以需要把模 **2** 修改为模 **4**，将其随机范围变为 **1-3** 才可以。

(4)adder.xdc(约束文件)

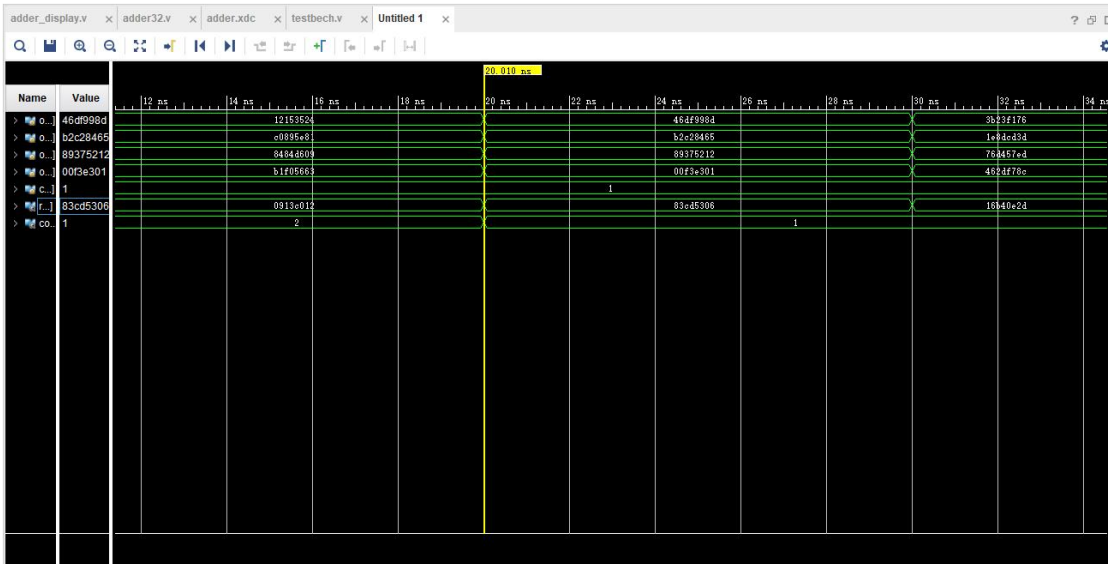
```
set_property PACKAGE_PIN AC19 [get_ports clk]
set_property PACKAGE_PIN H7   [get_ports {led_cout[0]}]
set_property PACKAGE_PIN D5   [get_ports {led_cout[1]}]
set_property PACKAGE_PIN Y3   [get_ports resetn]
set_property PACKAGE_PIN AC21 [get_ports {input_sel[0]}]
set_property PACKAGE_PIN AD24 [get_ports {input_sel[1]}]
set_property PACKAGE_PIN AC22 [get_ports {sw_cin[0]}]
set_property PACKAGE_PIN AC23 [get_ports {sw_cin[1]}]

set_property IOSTANDARD LVCMOS33 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports {led_cout[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led_cout[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports resetn]
set_property IOSTANDARD LVCMOS33 [get_ports {input_sel[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {input_sel[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw_cin[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw_cin[1]}]
```

说明部分：

添加了接口，由于 cin,cout,input_sel 均变成了两位，所以需要对其高位和低位分别添加引脚绑定。

5.实验结果分析

(1)仿真结果截图与分析：



解释说明：

在 20ns 的时候：

operand1=46df998d

operand2=b2c28465

operand3=89375212

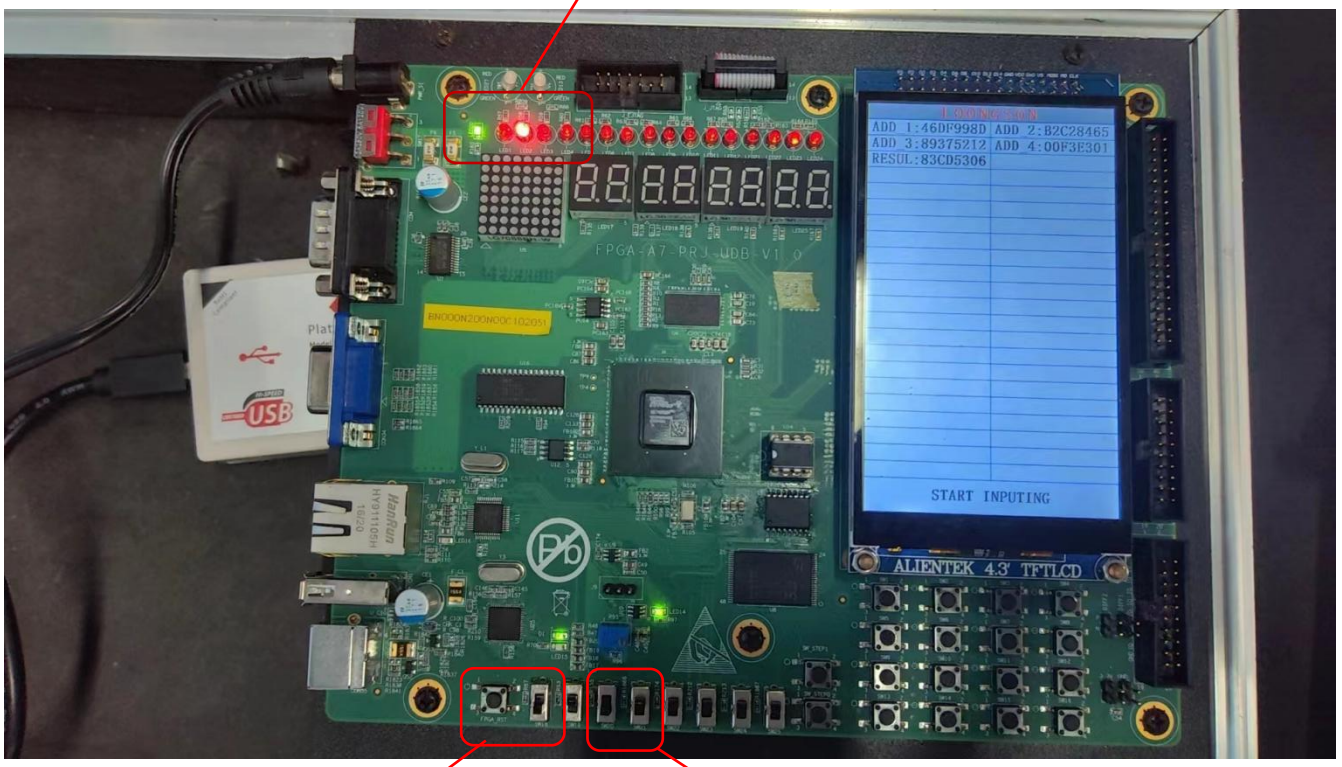
operand4=00f3e301

cin=01

用计算器计算得：183cd5306
由图可知 Result=83cd5306　cout=01
因此可知实验结果是正确的。

(2)实验箱结果验证：

这里表示向高位的进位，亮表示 0，不亮表示 1。从左到右是低位到高位。可以看到相加后进位为 01。

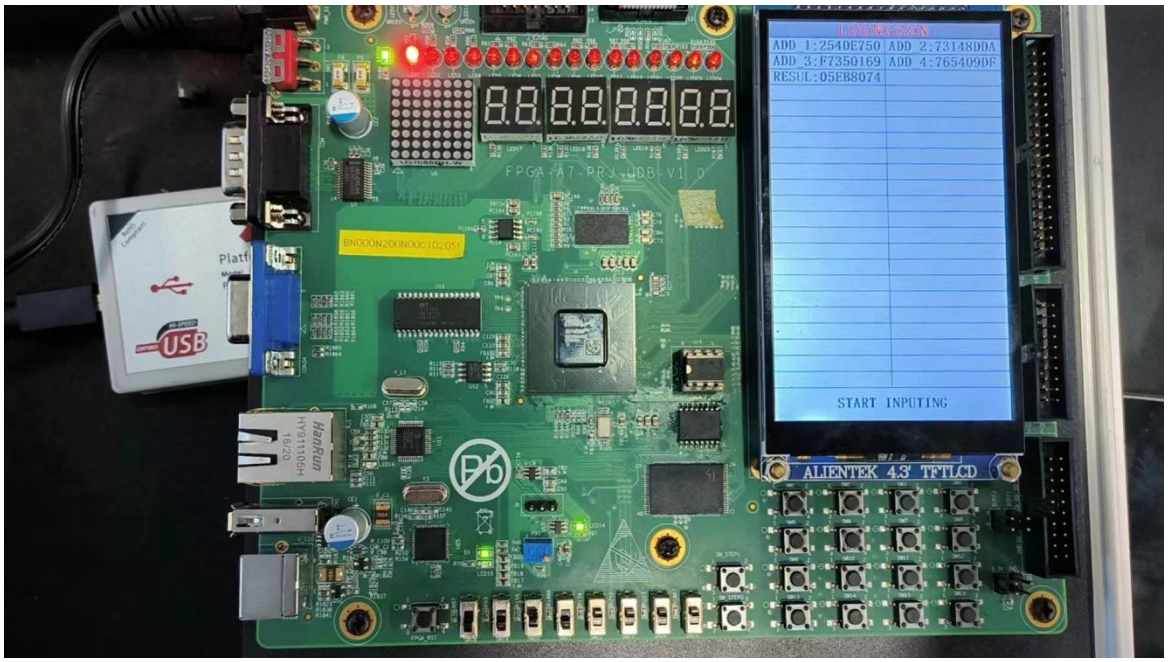


这里是拨码开关，调为 00，01，10，11 分别代表输入 operand 1-4。

这里是来自低位的进位的拨码开关，这里表示低位的进位是 01

由图可知，该实验箱运算结果与上面运算结果一致，因此是正确的。
通过调整拨码开关，可以控制输入哪一个加数，控制进位是多少，结果可以在 LED 屏和 LED 进位灯中观察得到。

再进行一次运算，验证可知该结果是正确的。



6.总结感想

（1）通过老师的一步步指导以及自己课后的学习，初步认识了 verilog 语言，了解了 vivado 软件和实验箱的基本使用方法与技巧，收获很多，为之后的实验打下了牢固的基础。

（2）深刻体会了用 vivado 如何实现一个加法器，自己也可以进行简单的修改。

（3）学习了 Visio 的基本用法，为以后作图打下了基础。