组成原理实验课程第_3_次实验报告

实验名称	定点乘法			班级	李涛老师
学生姓名	孙蕗	学号	2112060	指导老师	董前琨
实验地点	A308		实验时间	2023.4.4	

1、 实验目的

（1） 熟悉并掌握 MIPS 计算机中寄存器堆的原理和设计方法。

（2） 初步了解 MIPS 指令结构和源操作数/目的操作数的概念。

（3） 熟悉并运用 verilog 语言进行电路设计。

（4） 为后续设计 cpu 的实验打下基础。

2、 实验设备

（1） 装有 Xilinx Vivado 的计算机一台。

（2） LS-CPU-EXB-002 教学系统实验箱一套。

3、 实验要求

（1） 将原有的寄存器堆的写操作进行改进，使用 4 位 wen 控制信号，对应写入 wdata 的四个字节，比如 wen 为“1011”时，写入第 4、2、1 三个字节。

（2） 将原有的寄存器堆的读操作进行改进，使用 2 位 ren 控制信号，控制读出数据的高 16 位和低 16 位.注意寄存器堆的两个读端口同时控制。

（3） 本次实验不用仿真波形，直接上实验箱验证即可，注意八个拨码开关应都用上，wen 用 4 个，ren 用 2 个，input_sel 用 2 个，上实验箱时请注意区分。

4. 实验内容说明

（1） 学习 MIPS 计算机中寄存器堆的设计及原理，如：有多少个寄存器，有无特殊设置的寄存器，mips 指令如何去索引寄存器的等。

（2） 自行设计本次实验的方案，画出结构框图，详细标出输入输出端口，本次实验建议设计为异步读同步写的寄存器堆，即读寄存器不需要时钟控制，但写寄存器需时钟控制。

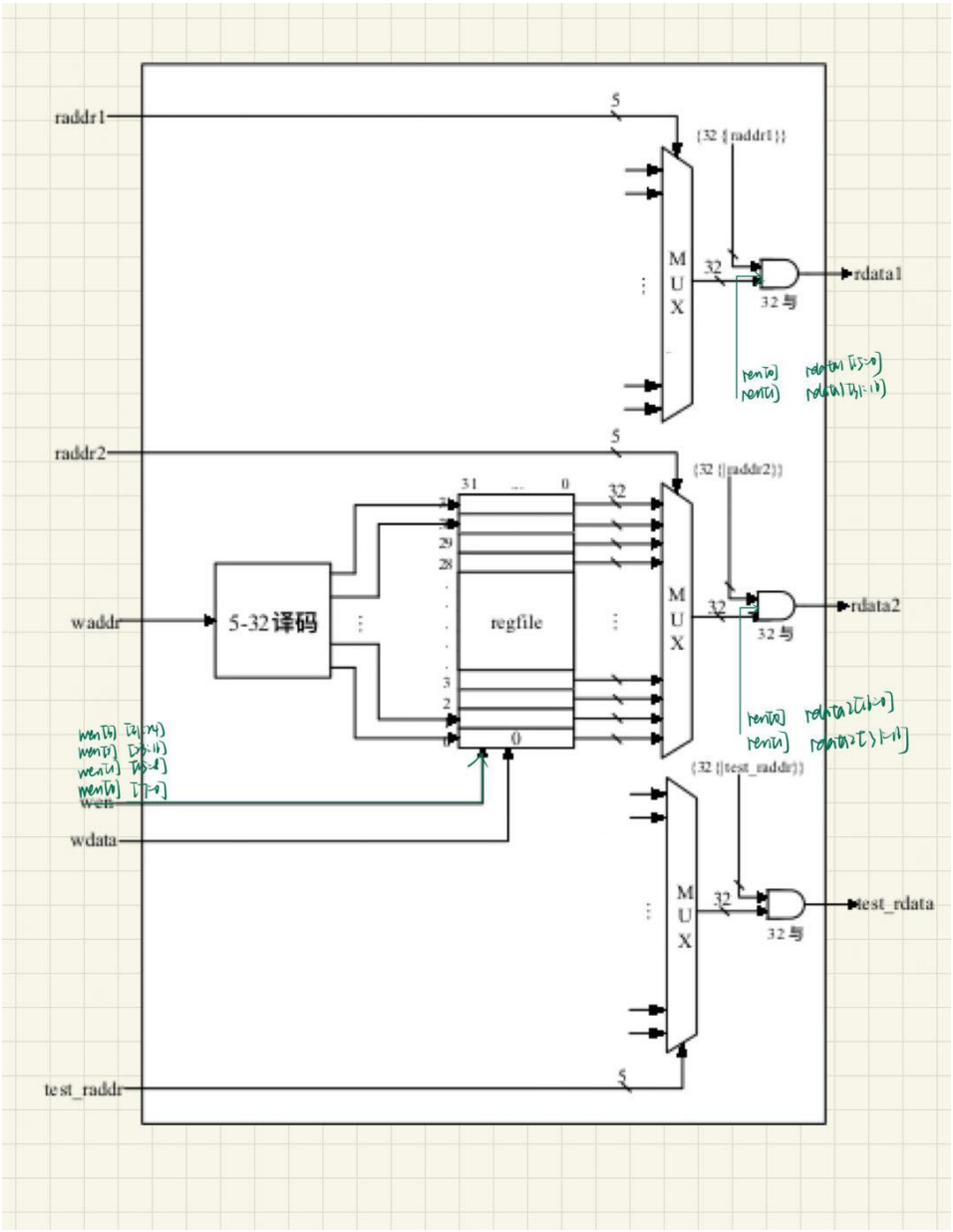
（3） 本次实验建议寄存器堆设计为 1 个写端口和 2 个读端口，后续 CPU 实验用到的寄存器堆需要 1 个写端口和 2 个读端口。

（4） 根据设计的实验方案，使用 verilog 编写相应代码。

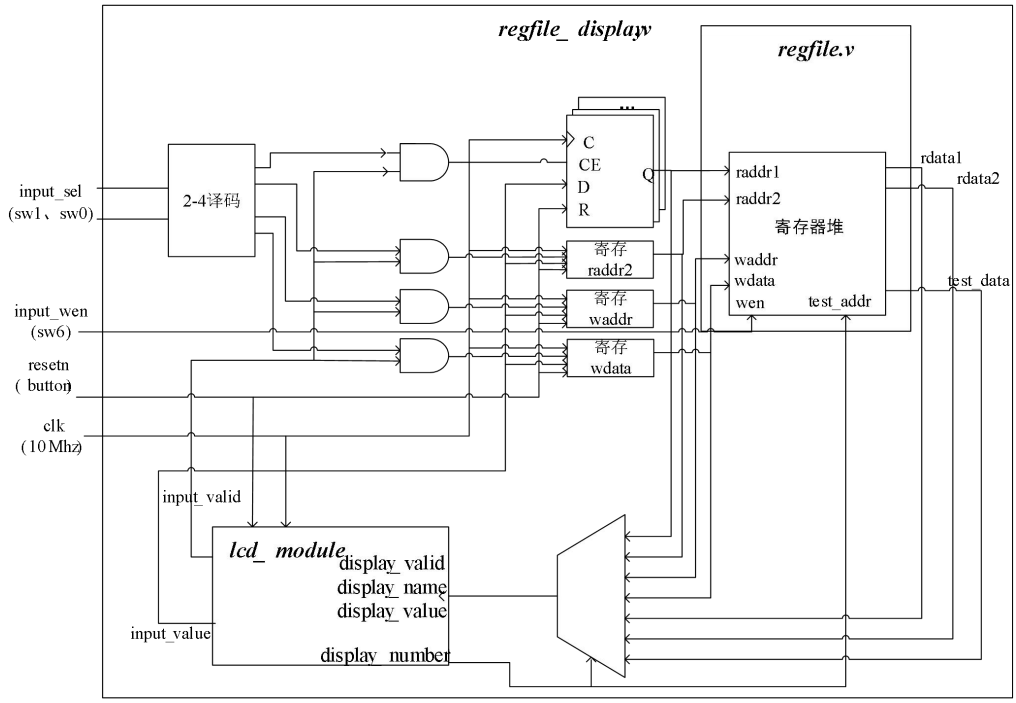
（5） 对编写的代码进行仿真，得到正确的波形图。

（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，见图 4.1。外围模块中需调用封装好的 LCD 触摸屏模块，显示寄存器堆的读写端口地址和数据，最好能扫描出所有寄存器的值显示在 LCD 触摸屏上，并且需要利用触摸功能输入寄存器堆的读写地址和写数据。

5. 实验原理图


raddr1
5
{32 {raddr1}}
M
U
X
32
32 与
rdata1
raddr2
5
{32 {raddr2}}
31
0
32
29
28
3
2
1
0
5-32译码
regfile
waddr
M
U
X
32
32 与
rdata2
wen
wdata
{32 {test_raddr}}
M
U
X
32
32 与
test_rdata
test_raddr
5

实验顶层模块框图：



4、 实验步骤

A. regfile.v 修改

（1） regfile.v 修改，增加 2 位 ren 控制信号和 4 位 wen 控制信号

```
module regfile(
    input                clk,
    input        [3 :0] wen,
    input        [1 :0] ren,
    input        [4 :0] raddr1,
    input        [4 :0] raddr2,
    input        [4 :0] waddr,
    input        [31:0] wdata,
    output reg [31:0] rdata1,
    output reg [31:0] rdata2,
    input        [4 :0] test_addr,
    output reg [31:0] test_data
    );
```

（2） regfile.v 修改，wen 写操作

```
 always @(posedge clk)
    begin
        if (wen[0])
            rf[waddr][7:0] <= wdata[7:0];// 写使能控制字节 0 写入
        else
            rf[waddr][7:0] <= 8'd0;
        if (wen[1])
            rf[waddr][15:8] <= wdata[15:8];// 写使能控制字节 1 写入
        else
```

```
                rf[waddr][15:8] <= 8'd0;
            if (wen[2])
                rf[waddr][23:16] <= wdata[23:16];// 写使能控制字节 2 写入
            else
                rf[waddr][23:16] <= 8'd0;
            if (wen[3])
                rf[waddr][31:24] <= wdata[31:24];// 写使能控制字节 3 写入
            else
                rf[waddr][31:24] <= 8'd0;
        end
```

判断 wen[i]的值，如果为 1 的话，将第 i 字节的值写入寄存器。如果 wen[0]是 1，写入寄存器的最低字节；如果 wen[1]是 1，写入寄存器的次低字节；如果 wen[2]是 1，写入寄存器的次高字节；如果 wen[3]是 1，写入寄存器的最高字节。

（3） regfile.v 修改，ren 读操作

```
    always @(*)
        begin
            if (ren[0])
                rdata1[15:0] <= rf[raddr1][15:0];//控制低 16 位读取
            else
                rdata1[15:0] <= 16'd0;
            if (ren[1])
                rdata1[31:16] <= rf[raddr1][31:16];//控制高 16 位读取
            else
                rdata1[31:16] <= 16'd0;
        end
        //读端口 2
        always @(*)
        begin
            if (ren[0])
                rdata2[15:0] <= rf[raddr2][15:0];//控制低 16 位读取
            else
                rdata2[15:0] <= 16'd0;
            if (ren[1])
                rdata2[31:16] <= rf[raddr2][31:16];//控制高 16 位读取
            else
                rdata2[31:16] <= 16'd0;
        end
```

判断 ren[i]的值，如果 ren[0]为 1，读寄存器后 16 位内容；如果 ren[1]为 1，读寄存器前 16 位内容。

B. regfile_display.v 修改

（1） 修改 regfile_display.v 文件，增加 wen[3:0]，ren[1:0]的拨码开关和 led 显示

```
module regfile_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，用于产生写使能、读使能和选择输入数
    input [3:0] wen,
    input [1:0] ren,
    input [1:0] input_sel,

    //led 灯，用于指示写使能、读使能信号，和正在输入什么数据
    output led_wen0,
        output led_wen1,
        output led_wen2,
        output led_wen3,
        output led_ren0,
        output led_ren1,
        output led_waddr,       //指示输入写地址
        output led_wdata,       //指示输入写数据
        output led_raddr1,      //指示输入读地址 1
        output led_raddr2,      //指示输入读地址 2
```

（2） 修改 regfile_display.v 文件，增加 wen 和 ren 的 led 显示

```
    assign led_wen0 = (wen[0]==1);
    assign led_wen1 = (wen[1]==1);
    assign led_wen2 = (wen[2]==1);
    assign led_wen3 = (wen[3]==1);
    assign led_ren0 = (ren[0]==1);
    assign led_ren1 = (ren[1]==1);
    assign led_raddr1 = (input_sel==2'd0);
    assign led_raddr2 = (input_sel==2'd1);
    assign led_waddr    = (input_sel==2'd2);
    assign led_wdata    = (input_sel==2'd3);
//-----{LED 显示}end
```

（3） 修改 regfile_display.v 文件，修改 regfile 模块

```
regfile rf_module(
        .clk    (clk    ),
        .wen    (wen    ),
        .ren    (ren),
        .raddr1(raddr1),
        .raddr2(raddr2),
        .waddr (waddr ),
        .wdata (wdata ),
        .rdata1(rdata1),
        .rdata2(rdata2),
        .test_addr(test_addr),
```

```
        .test_data(test_data)
    );
//-----{调用寄存器堆模块}end
```

C. 修改 regfile.xdc 文件

（1） led 灯连接，用于输出

```
#led 灯连接，用于输出
set_property PACKAGE_PIN H7 [get_ports led_wen3]
set_property PACKAGE_PIN D5 [get_ports led_wen2]
set_property PACKAGE_PIN A3 [get_ports led_wen1]
set_property PACKAGE_PIN A5 [get_ports led_wen0]
set_property PACKAGE_PIN A4 [get_ports led_ren1]
set_property PACKAGE_PIN F7 [get_ports led_ren0]
set_property PACKAGE_PIN D5 [get_ports led_waddr]
set_property PACKAGE_PIN A3 [get_ports led_wdata]
set_property PACKAGE_PIN A5 [get_ports led_raddr1]
set_property PACKAGE_PIN A4 [get_ports led_raddr2]
```

（2） 拨码开关连接，用于输入

```
#拨码开关连接，用于输入，依次为 sw0,sw1,sw7
set_property PACKAGE_PIN AC21 [get_ports wen[3]]
set_property PACKAGE_PIN AD24 [get_ports wen[2]]
set_property PACKAGE_PIN AC22 [get_ports wen[1]]
set_property PACKAGE_PIN AC23 [get_ports wen[0]]
set_property PACKAGE_PIN AB6 [get_ports ren[1]]
set_property PACKAGE_PIN W6 [get_ports ren[0]]
set_property PACKAGE_PIN AA7 [get_ports input_sel[1]]
set_property PACKAGE_PIN Y6 [get_ports input_sel[0]]

set_property IOSTANDARD LVCMOS33 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports resetn]
set_property IOSTANDARD LVCMOS33 [get_ports led_wen3]
set_property IOSTANDARD LVCMOS33 [get_ports led_wen2]
set_property IOSTANDARD LVCMOS33 [get_ports led_wen1]
set_property IOSTANDARD LVCMOS33 [get_ports led_wen0]
set_property IOSTANDARD LVCMOS33 [get_ports led_ren1]
set_property IOSTANDARD LVCMOS33 [get_ports led_ren0]
set_property IOSTANDARD LVCMOS33 [get_ports led_raddr1]
set_property IOSTANDARD LVCMOS33 [get_ports led_raddr2]
set_property IOSTANDARD LVCMOS33 [get_ports led_waddr]
set_property IOSTANDARD LVCMOS33 [get_ports led_wdata]
set_property IOSTANDARD LVCMOS33 [get_ports wen]
set_property IOSTANDARD LVCMOS33 [get_ports input_sel[1]]
set_property IOSTANDARD LVCMOS33 [get_ports input_sel[0]]
```

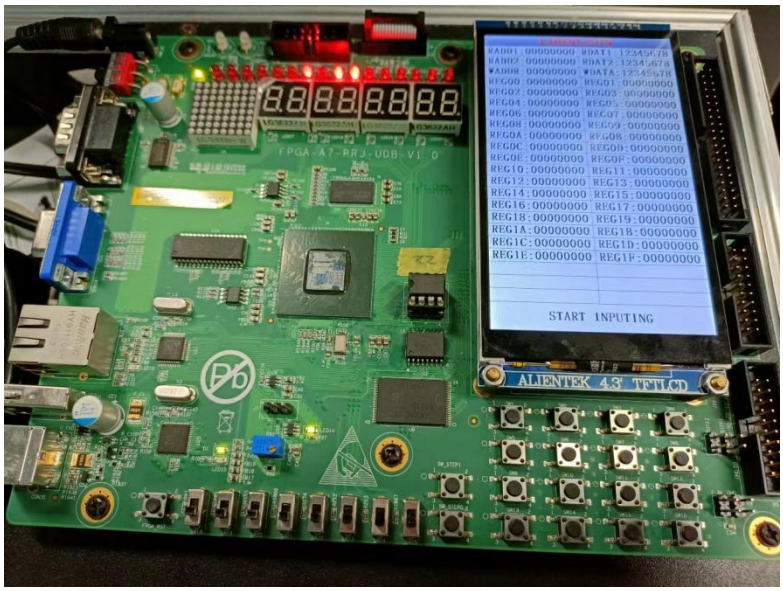
led 灯从左到右依次为 wen[3]，wen[2]，wen[1]，wen[0]，ren[1]，ren[0]，是否输入写地址，

是否输入写数据，是否输入读地址 1，是否输入读地址 2。

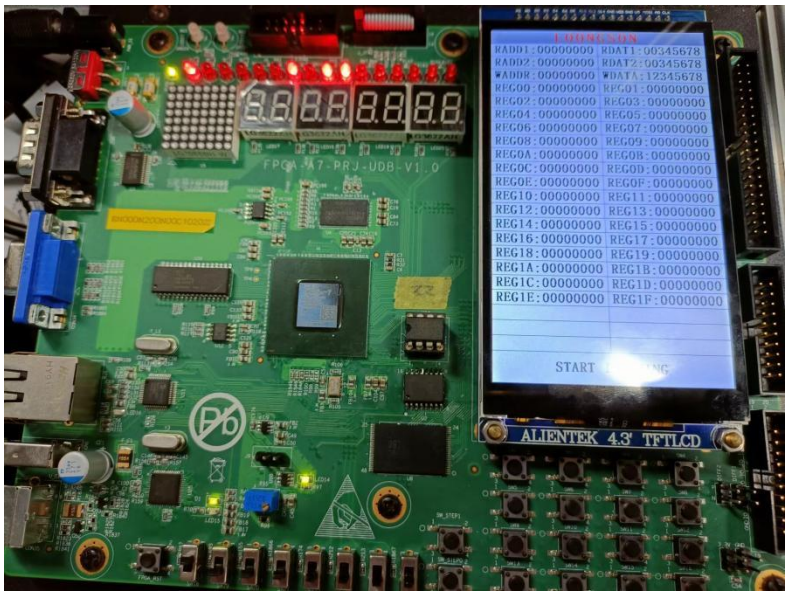
拨码开关从左到右依次为 wen[3]，wen[2]，wen[1]，wen[0]，ren[1]，ren[0]，input_sel[1]，input_sel[0]。

5、 实验结果分析

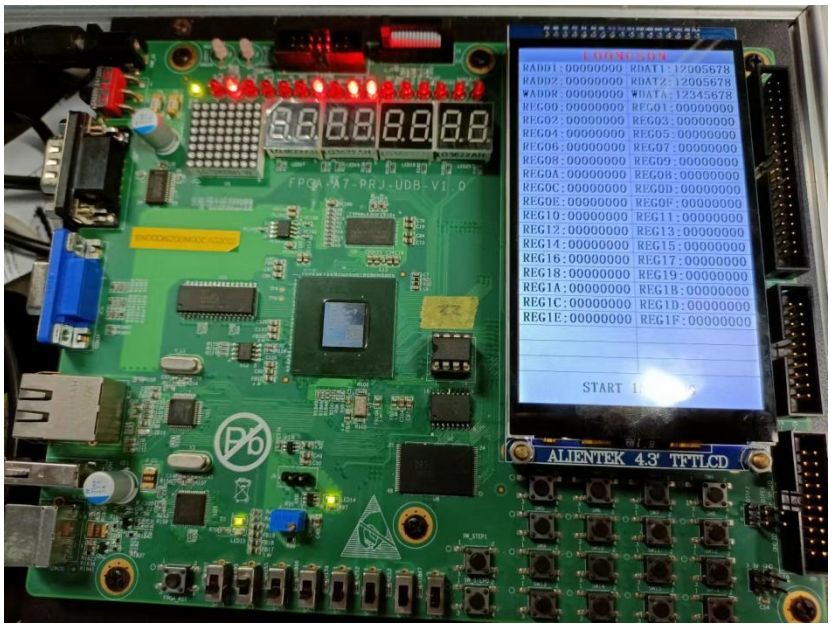
（1） 输入 12345678



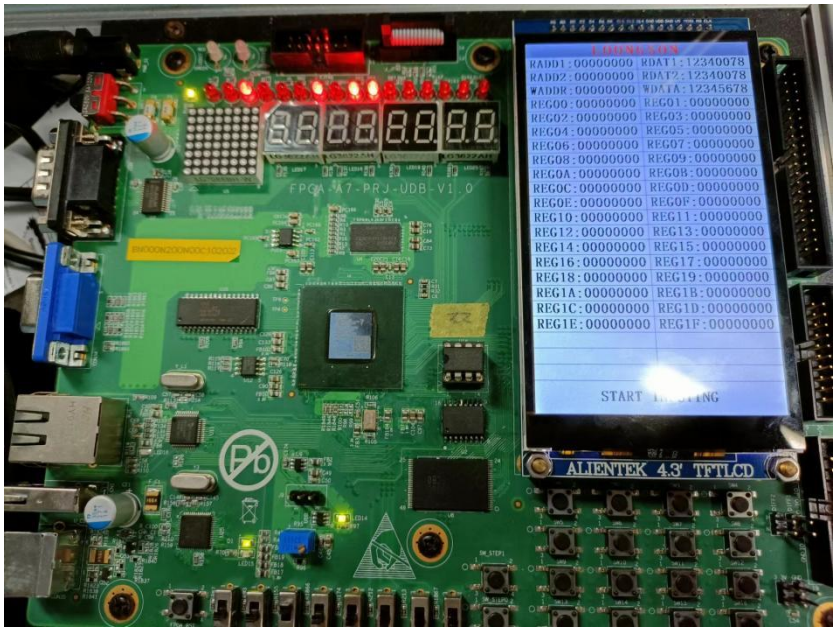
（2） 拨码开关 1



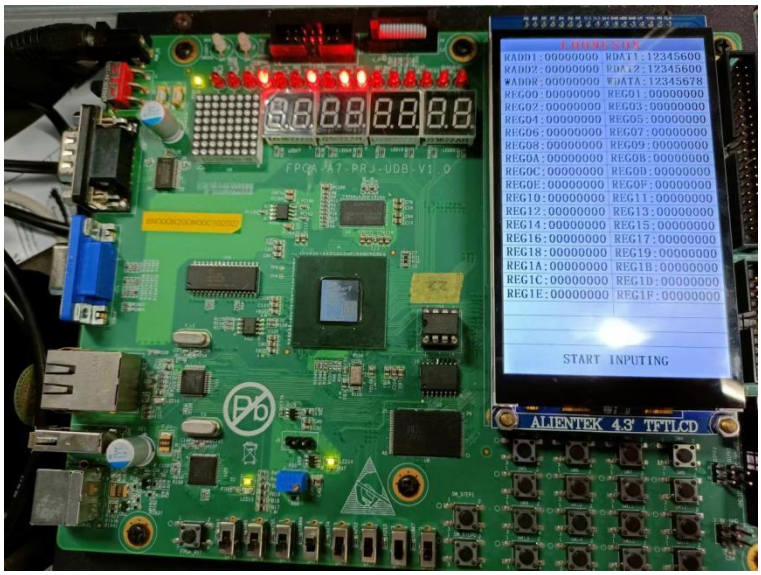
（3） 拨码开关 2



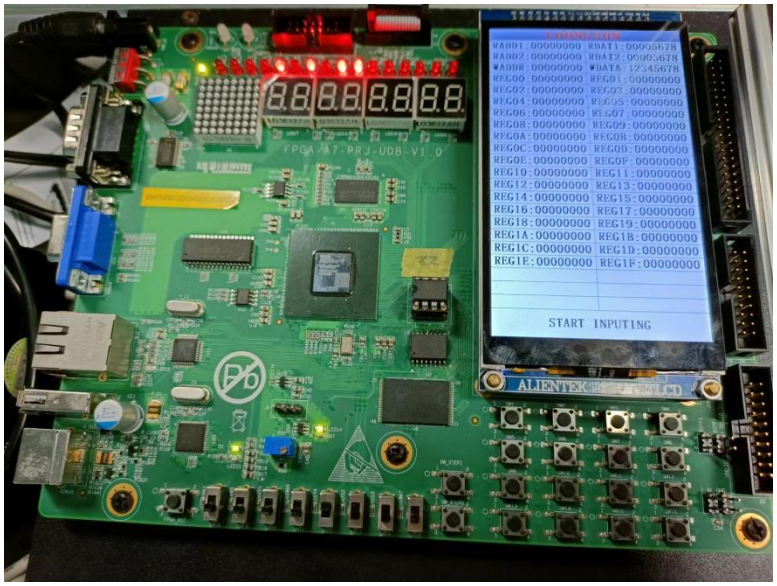
（4） 拨码开关 3



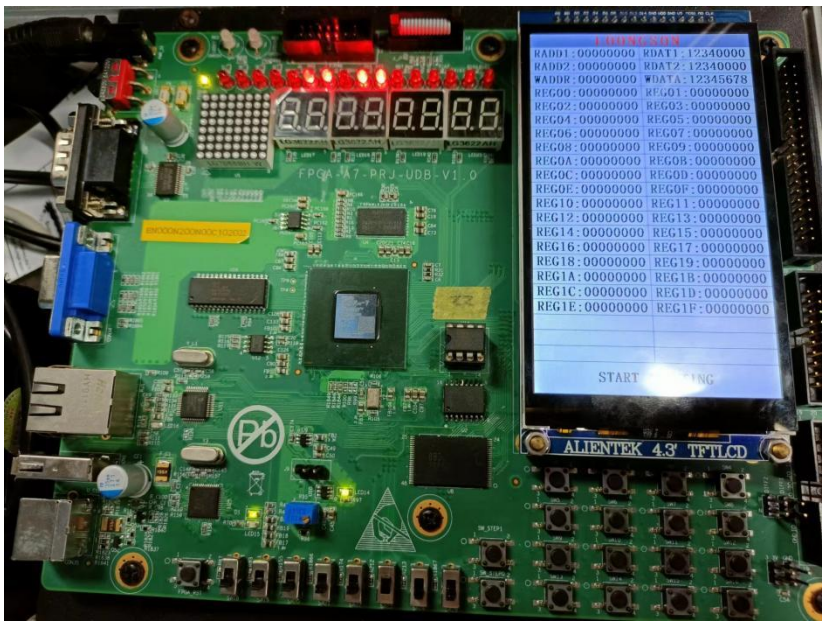
（5） 拨码开关 4



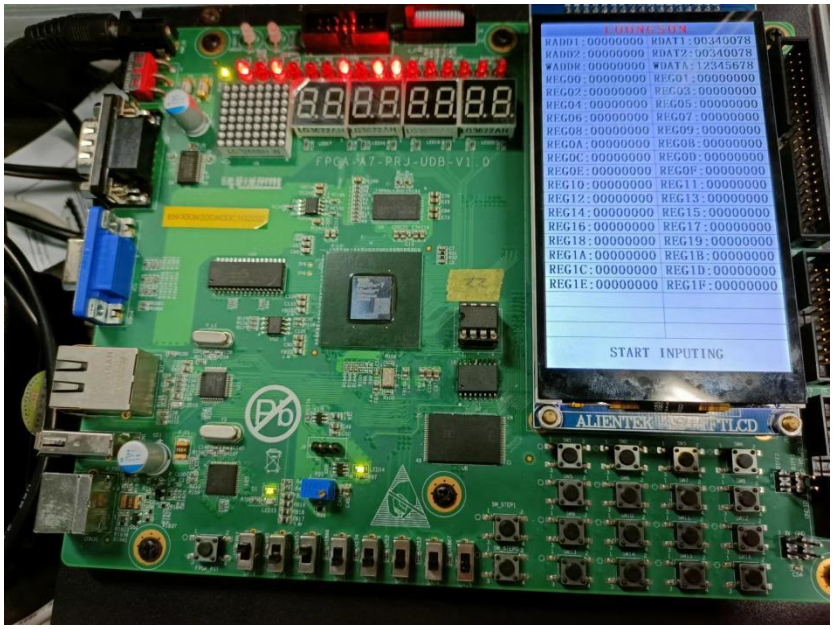
（6） 拨码开关 5



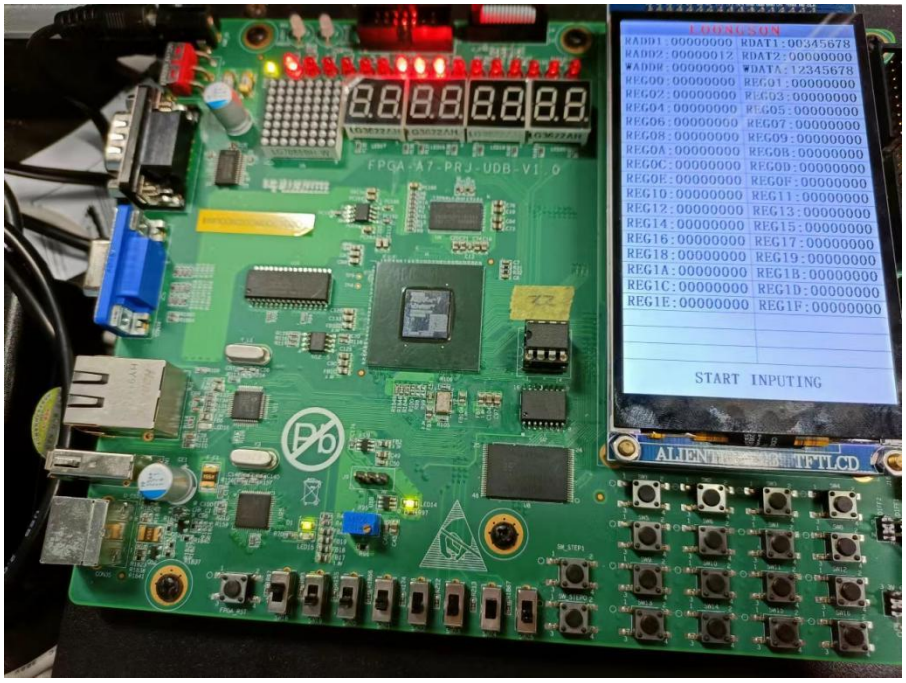
（7） 拨码开关 6



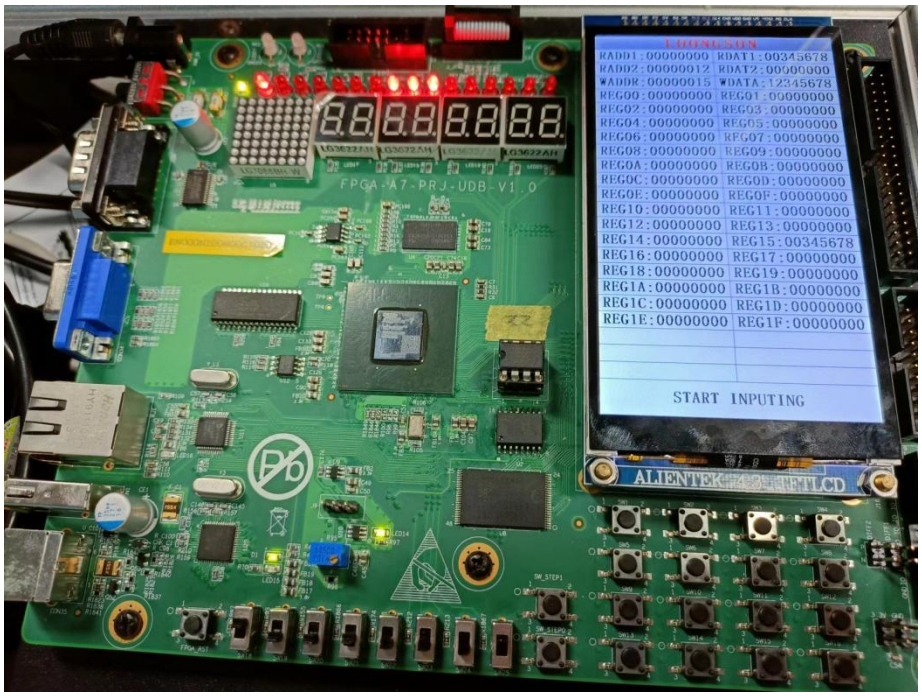
（7） 拨码开关 1、3



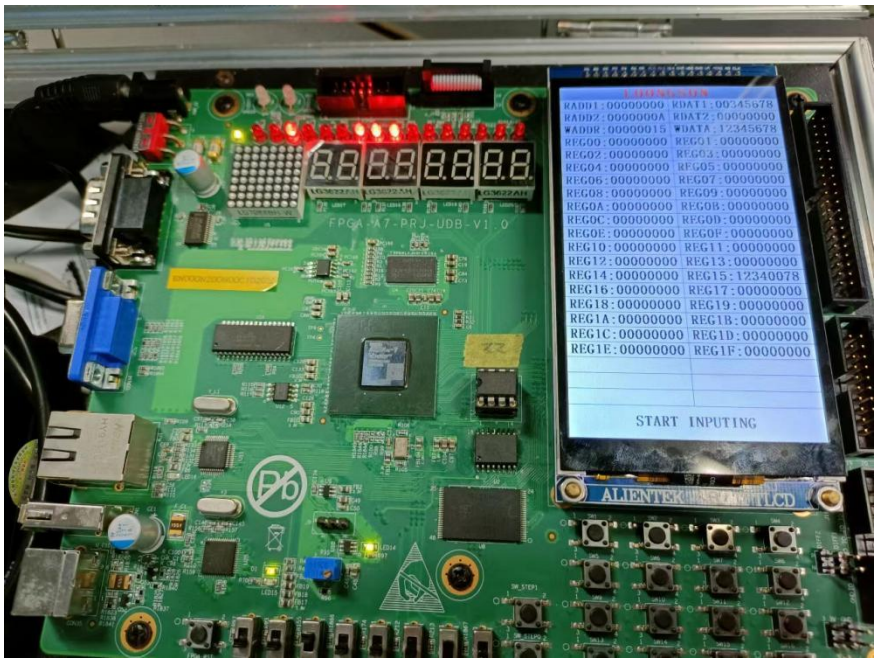
（8） 拨码开关 1、7



（8） 拨码开关 1、8



（9） 再拨动拨码开关 3



6、 总结感想

在原有的寄存器堆实现的基础上，实现了对写操作和读操作的改进。通过 wen 控制信号的四位，实现对 wdata 的四个字节的选择性写入；通过 ren 控制信号的两位，实现对 rdata1 和 rdata2 的高 16 位和低 16 位的选择性输出。同时，注意到寄存器堆的两个读端口需要同时控制。