组成原理实验课程第三次实验报告

实验名称	寄存器堆实现实验			班级	张金
学生姓名	冯思程	学号	2112213	指导老师	董前琨
实验地点	实验楼 A306		实验时间	4 月 17 日 14：00	

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、 实验内容说明

准备活动： （这些内容形成的 verilog 代码可在 source_code 中得到验证和参考）

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.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 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。

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

按要求修改内容：

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。

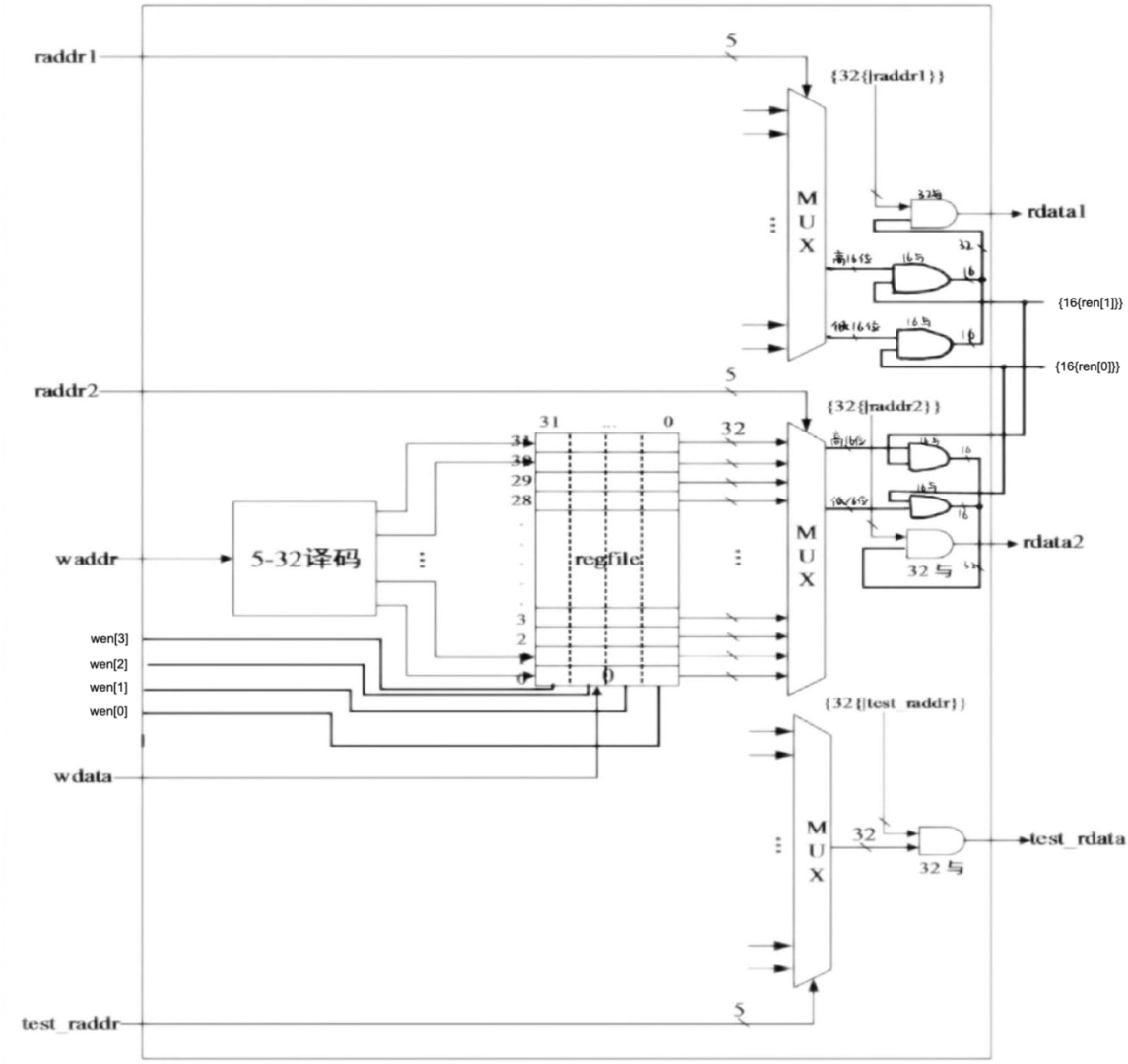
3.本次实验不用仿真波形，直接上实验箱验证即可，注意八个拨码开关应都用上，wen 用 4 个，ren 用 2 个，input_sel 用 2 个，上实验箱时请注意区分。

4.实验报告中的原理图为图 4.2 类似的示意图，只把 wen 和 ren 加入控制线路即可。不再是顶层模块图。

5.按实验报告模板要求完成实验报告，以附件形式提交。注意实验报告中要有介绍分析的内容，针对实验箱照片，要解释图中信息，是否验证成功。

3、 实验原理图

对本实验的原理图进行修改，修改后的原理图如下：



其中写使能信号 wen[3]到 wen[0]分别连接到寄存器堆的 4 个字节，对应按字节进行数据写入的操作。同时需要用 ren 来控制读取数据的高低 16 位所以读使能信号 ren[1]拓展为 16 位，和寄存器的高 16 位做与操作，ren[0]拓展为 16 位，和寄存器的低 16 位做与操作，二者拼接成 32 位的数，最终和拓展后的 raddr 做与操作(即若 raddr=0 则 rdata 必为 0) 实现了通过 ren 来控制分别读取数据的高低 16 位的功能。

4、 实验步骤

（1）对 regfile.v 的修改： （红色表示进行修改的地方）

1.对 input 输入定义根据实验的修改要求进行修改，其中 wen 改成 4 位按字节输入，同时要加入 ren 位控制读出数据的高 16 位和低 16 位，代码如下：

```
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
);
reg [31:0] rf[31:0];
```

2.输入的时候是按字节分成 4 个字节输入，修改改为通过 wen 的每个位的值来控制输入信号。

```
always @(posedge clk)
    begin
        if (wen)
        begin
            rf[waddr] <= wdata;
            //通过 wen 控制输入信号
            rf[waddr][7:0]      <= wen[0]?wdata[7:0]:8'd0;
            rf[waddr][15:8]     <= wen[1]?wdata[15:8]:8'd0;
            rf[waddr][23:16]    <= wen[2]?wdata[23:16]:8'd0;
            rf[waddr][31:24]    <= wen[3]?wdata[31:24]:8'd0;
        end
    End
```

3. 对读端口 1，2 进行修改，使其可以按照 ren 控制来进行读取数据，代码如下：

```
//读端口 1
    always @(*)
    begin
        case (raddr1)
            5'd1 : rdata1 <= {ren[1]==0?16'b0:rf[1][31:16],ren[0]==0?16'b0:rf[1][15:0]};
            5'd2 : rdata1 <= {ren[1]==0?16'b0:rf[2][31:16],ren[0]==0?16'b0:rf[2][15:0]};
            5'd3 : rdata1 <= {ren[1]==0?16'b0:rf[3][31:16],ren[0]==0?16'b0:rf[3][15:0]};
            5'd4 : rdata1 <= {ren[1]==0?16'b0:rf[4][31:16],ren[0]==0?16'b0:rf[4][15:0]};
            5'd5 : rdata1 <= {ren[1]==0?16'b0:rf[5][31:16],ren[0]==0?16'b0:rf[5][15:0]};
            5'd6 : rdata1 <= {ren[1]==0?16'b0:rf[6][31:16],ren[0]==0?16'b0:rf[6][15:0]};
            5'd7 : rdata1 <= {ren[1]==0?16'b0:rf[7][31:16],ren[0]==0?16'b0:rf[7][15:0]};
            5'd8 : rdata1 <= {ren[1]==0?16'b0:rf[8][31:16],ren[0]==0?16'b0:rf[8][15:0]};
            5'd9 : rdata1 <= {ren[1]==0?16'b0:rf[9][31:16],ren[0]==0?16'b0:rf[9][15:0]};
            5'd10: rdata1 <= {ren[1]==0?16'b0:rf[10][31:16],ren[0]==0?16'b0:rf[10][15:0]};
            5'd11: rdata1 <= {ren[1]==0?16'b0:rf[11][31:16],ren[0]==0?16'b0:rf[11][15:0]};
            5'd12: rdata1 <= {ren[1]==0?16'b0:rf[12][31:16],ren[0]==0?16'b0:rf[12][15:0]};
            5'd13: rdata1 <= {ren[1]==0?16'b0:rf[13][31:16],ren[0]==0?16'b0:rf[13][15:0]};
            5'd14: rdata1 <= {ren[1]==0?16'b0:rf[14][31:16],ren[0]==0?16'b0:rf[14][15:0]};
            5'd15: rdata1 <= {ren[1]==0?16'b0:rf[15][31:16],ren[0]==0?16'b0:rf[15][15:0]};
            5'd16: rdata1 <= {ren[1]==0?16'b0:rf[16][31:16],ren[0]==0?16'b0:rf[16][15:0]};
            5'd17: rdata1 <= {ren[1]==0?16'b0:rf[17][31:16],ren[0]==0?16'b0:rf[17][15:0]};
            5'd18: rdata1 <= {ren[1]==0?16'b0:rf[18][31:16],ren[0]==0?16'b0:rf[18][15:0]};
```

```
                5'd19: rdata1 <= {ren[1]==0?16'b0:rf[19][31:16],ren[0]==0?16'b0:rf[19][15:0]};
                5'd20: rdata1 <= {ren[1]==0?16'b0:rf[20][31:16],ren[0]==0?16'b0:rf[20][15:0]};
                5'd21: rdata1 <= {ren[1]==0?16'b0:rf[21][31:16],ren[0]==0?16'b0:rf[21][15:0]};
                5'd22: rdata1 <= {ren[1]==0?16'b0:rf[22][31:16],ren[0]==0?16'b0:rf[22][15:0]};
                5'd23: rdata1 <= {ren[1]==0?16'b0:rf[23][31:16],ren[0]==0?16'b0:rf[23][15:0]};
                5'd24: rdata1 <= {ren[1]==0?16'b0:rf[24][31:16],ren[0]==0?16'b0:rf[24][15:0]};
                5'd25: rdata1 <= {ren[1]==0?16'b0:rf[25][31:16],ren[0]==0?16'b0:rf[25][15:0]};
                5'd26: rdata1 <= {ren[1]==0?16'b0:rf[26][31:16],ren[0]==0?16'b0:rf[26][15:0]};
                5'd27: rdata1 <= {ren[1]==0?16'b0:rf[27][31:16],ren[0]==0?16'b0:rf[27][15:0]};
                5'd28: rdata1 <= {ren[1]==0?16'b0:rf[28][31:16],ren[0]==0?16'b0:rf[28][15:0]};
                5'd29: rdata1 <= {ren[1]==0?16'b0:rf[29][31:16],ren[0]==0?16'b0:rf[29][15:0]};
                5'd30: rdata1 <= {ren[1]==0?16'b0:rf[30][31:16],ren[0]==0?16'b0:rf[30][15:0]};
                5'd31: rdata1 <= {ren[1]==0?16'b0:rf[31][31:16],ren[0]==0?16'b0:rf[31][15:0]};
                    default : rdata1 <= 32'd0;
            endcase
        end
        //读端口 2
        always @(*)
        begin
            case (raddr2)
                5'd1 : rdata2 <= {ren[1]==0?16'b0:rf[1][31:16],ren[0]==0?16'b0:rf[1][15:0]};
                5'd2 : rdata2 <= {ren[1]==0?16'b0:rf[2][31:16],ren[0]==0?16'b0:rf[2][15:0]};
                5'd3 : rdata2 <= {ren[1]==0?16'b0:rf[3][31:16],ren[0]==0?16'b0:rf[3][15:0]};
                5'd4 : rdata2 <= {ren[1]==0?16'b0:rf[4][31:16],ren[0]==0?16'b0:rf[4][15:0]};
                5'd5 : rdata2 <= {ren[1]==0?16'b0:rf[5][31:16],ren[0]==0?16'b0:rf[5][15:0]};
                5'd6 : rdata2 <= {ren[1]==0?16'b0:rf[6][31:16],ren[0]==0?16'b0:rf[6][15:0]};
                5'd7 : rdata2 <= {ren[1]==0?16'b0:rf[7][31:16],ren[0]==0?16'b0:rf[7][15:0]};
                5'd8 : rdata2 <= {ren[1]==0?16'b0:rf[8][31:16],ren[0]==0?16'b0:rf[8][15:0]};
                5'd9 : rdata2 <= {ren[1]==0?16'b0:rf[9][31:16],ren[0]==0?16'b0:rf[9][15:0]};
                5'd10: rdata2 <= {ren[1]==0?16'b0:rf[10][31:16],ren[0]==0?16'b0:rf[10][15:0]};
                5'd11: rdata2 <= {ren[1]==0?16'b0:rf[11][31:16],ren[0]==0?16'b0:rf[11][15:0]};
                5'd12: rdata2 <= {ren[1]==0?16'b0:rf[12][31:16],ren[0]==0?16'b0:rf[12][15:0]};
                5'd13: rdata2 <= {ren[1]==0?16'b0:rf[13][31:16],ren[0]==0?16'b0:rf[13][15:0]};
                5'd14: rdata2 <= {ren[1]==0?16'b0:rf[14][31:16],ren[0]==0?16'b0:rf[14][15:0]};
                5'd15: rdata2 <= {ren[1]==0?16'b0:rf[15][31:16],ren[0]==0?16'b0:rf[15][15:0]};
                5'd16: rdata2 <= {ren[1]==0?16'b0:rf[16][31:16],ren[0]==0?16'b0:rf[16][15:0]};
                5'd17: rdata2 <= {ren[1]==0?16'b0:rf[17][31:16],ren[0]==0?16'b0:rf[17][15:0]};
                5'd18: rdata2 <= {ren[1]==0?16'b0:rf[18][31:16],ren[0]==0?16'b0:rf[18][15:0]};
                5'd19: rdata2 <= {ren[1]==0?16'b0:rf[19][31:16],ren[0]==0?16'b0:rf[19][15:0]};
                5'd20: rdata2 <= {ren[1]==0?16'b0:rf[20][31:16],ren[0]==0?16'b0:rf[20][15:0]};
                5'd21: rdata2 <= {ren[1]==0?16'b0:rf[21][31:16],ren[0]==0?16'b0:rf[21][15:0]};
                5'd22: rdata2 <= {ren[1]==0?16'b0:rf[22][31:16],ren[0]==0?16'b0:rf[22][15:0]};
                5'd23: rdata2 <= {ren[1]==0?16'b0:rf[23][31:16],ren[0]==0?16'b0:rf[23][15:0]};
                5'd24: rdata2 <= {ren[1]==0?16'b0:rf[24][31:16],ren[0]==0?16'b0:rf[24][15:0]};
```

```
                5'd25: rdata2 <= {ren[1]==0?16'b0:rf[25][31:16],ren[0]==0?16'b0:rf[25][15:0]};
                5'd26: rdata2 <= {ren[1]==0?16'b0:rf[26][31:16],ren[0]==0?16'b0:rf[26][15:0]};
                5'd27: rdata2 <= {ren[1]==0?16'b0:rf[27][31:16],ren[0]==0?16'b0:rf[27][15:0]};
                5'd28: rdata2 <= {ren[1]==0?16'b0:rf[28][31:16],ren[0]==0?16'b0:rf[28][15:0]};
                5'd29: rdata2 <= {ren[1]==0?16'b0:rf[29][31:16],ren[0]==0?16'b0:rf[29][15:0]};
                5'd30: rdata2 <= {ren[1]==0?16'b0:rf[30][31:16],ren[0]==0?16'b0:rf[30][15:0]};
                5'd31: rdata2 <= {ren[1]==0?16'b0:rf[31][31:16],ren[0]==0?16'b0:rf[31][15:0]};
                    default : rdata2 <= 32'd0;
            endcase
        end
```

（2）对 display 部分的修改：（红色表示进行修改的地方）

1.将 wen 修改为 4 位，同时添加 ren 两位控制读出数据的高 16 位和低 16 位，代码如下：

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

2.修改 LED 显示，wen 改成等于低位第一个字节。

```
//-----{LED 显示}begin
    assign led_wen      = wen[0];
    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.补充一个 ren 控制读取数据高低 16 位的端口

```
//-----{调用寄存器堆模块}begin
    //寄存器堆多增加一个读端口，用于在触摸屏上显示 32 个寄存器值
    wire [31:0] test_data;
    wire [4 :0] test_addr;

    reg   [4 :0] raddr1;
    reg   [4 :0] raddr2;
    reg   [4 :0] waddr;
    reg   [31:0] wdata;
    wire [31:0] rdata1;
    wire [31:0] rdata2;
    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
```

（3）约束文件进行修改：（红色表示进行修改的地方）

1.将实验中所需要用到的 8 个拨码开关重新定义，其中拨码开关的编号分别对应到实验箱正放拨码开关从左到右的 8 个拨码开关，代码如下：

```
#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 input_sel[1]]
set_property PACKAGE_PIN W6     [get_ports input_sel[0]]
set_property PACKAGE_PIN AA7 [get_ports ren[1]]
set_property PACKAGE_PIN Y6     [get_ports ren[0]]
```

2.对应上面管脚约束修改，修改设置引脚为电气标准为 LVCMOS33 语句，代码如下：

```
set_property IOSTANDARD LVCMOS33 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports resetn]
set_property IOSTANDARD LVCMOS33 [get_ports led_wen]
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[3]]
set_property IOSTANDARD LVCMOS33 [get_ports wen[2]]
set_property IOSTANDARD LVCMOS33 [get_ports wen[1]]
set_property IOSTANDARD LVCMOS33 [get_ports wen[0]]
set_property IOSTANDARD LVCMOS33 [get_ports ren[1]]
set_property IOSTANDARD LVCMOS33 [get_ports ren[0]]
set_property IOSTANDARD LVCMOS33 [get_ports input_sel[0]]
set_property IOSTANDARD LVCMOS33 [get_ports input_sel[1]]
```

（4）本实验进行仿真的意义不大而且仿真代码非常繁琐复杂，故不编写仿真文件来进行仿真，可以跳过仿真步骤直接将修改后的代码上箱验证。

（5）导入 lcd 屏模块，然后再进行上箱实验验证。lcd 屏模块在源码文件（source code）中给出，直接导入。然后分别依次跑综合、增强后确认无误，将电脑连接到实验箱后生成流文件到实验箱上，然后在实验箱上进行调试观察，验证是否完整的实现了目标功能。

5、 实验结果分析

上箱结果验证：

（1）对按钮进行说明：

最左侧的圆形按键是重置键，然后说明拨码开关：左起分别是

wen[3],wen[2],wen[1],wen[0],input_sel[1],input_sel[0],ren[1],ren[0]

（2）然后对功能进行说明：

wen[3]到 wen[0]是分别控制写入数据的 4 个字节，置 1 则是写入，置 0 则不写入。

input_sel[1]和 input_sel[0]是用于控制正在写入什么数据，00 是写入读数据 1 的地址，01 是写入读数据 2 的地址，10 是写入写入数据的地址，11 是写入写入数据的数据。

ren[1]和 ren[0]是用来控制读取数据的时候输出高低 16 位，11 的时候是读取 32 位，10 的时候读取高 16 位，01 时候读取低 16 位，00 的时候什么也不读取。

（3）对 LED 灯说明：(左起)

1：wen[0]

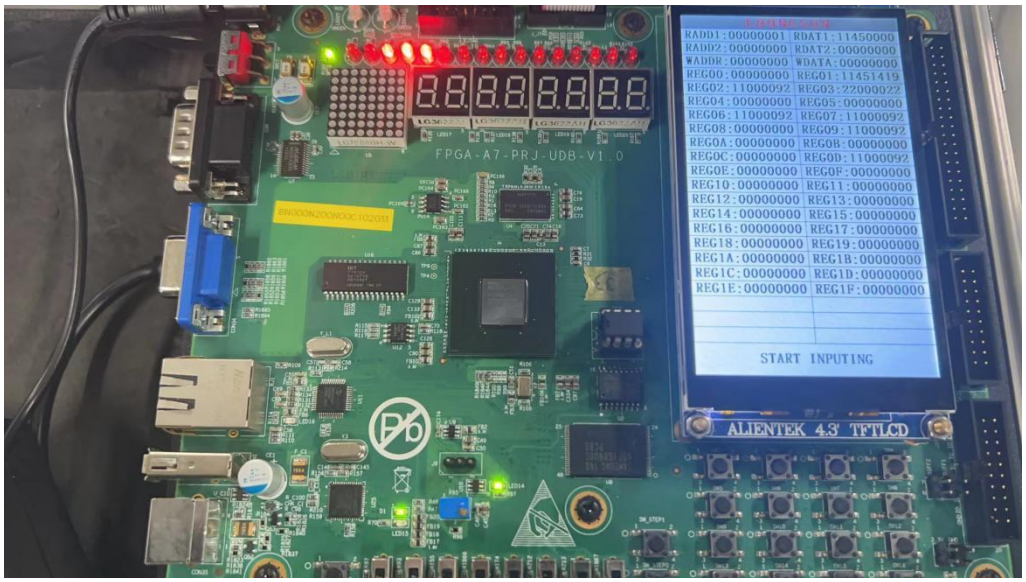
2：raddr1

3：raddr2

4：waddr

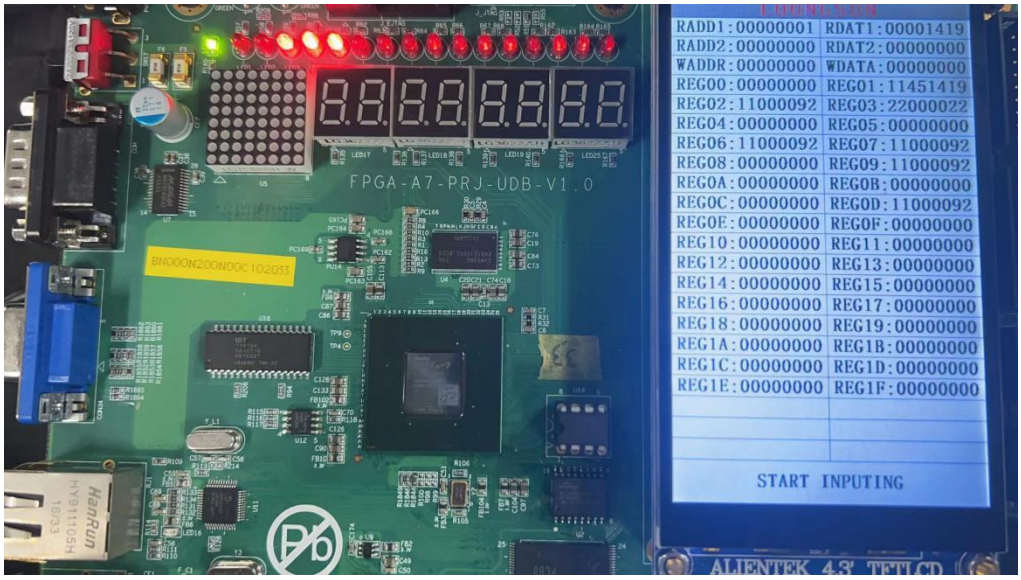
5：wdata

（4）然后是对功能进行验证：

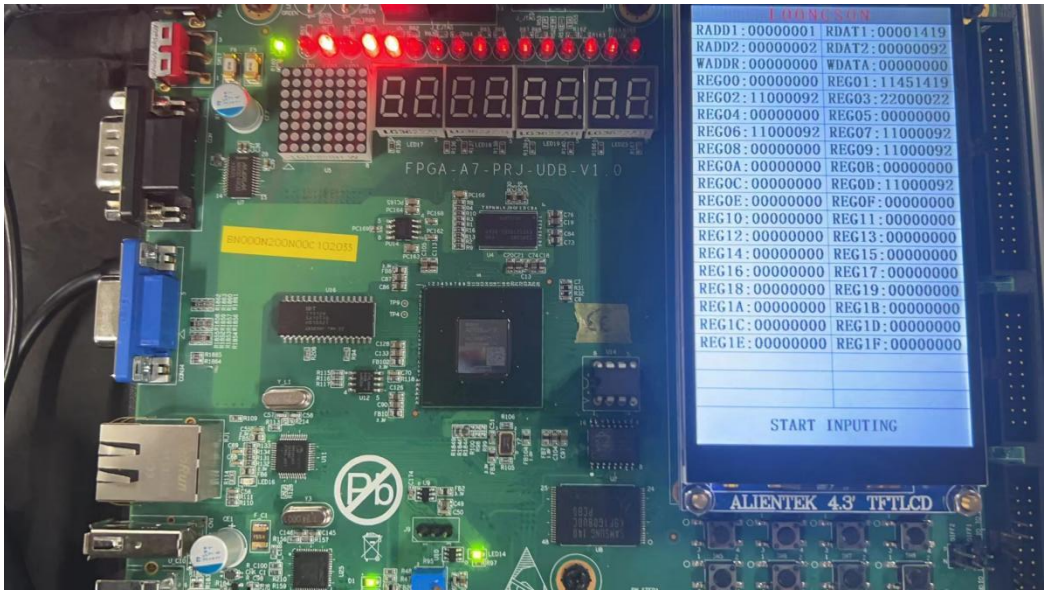


上图是验证读取功能：将 input_sel 设置成 00，写入地址 01，设置 ren 为 10，成功读取到 01 地址寄存器数据的高 16 位 1145。

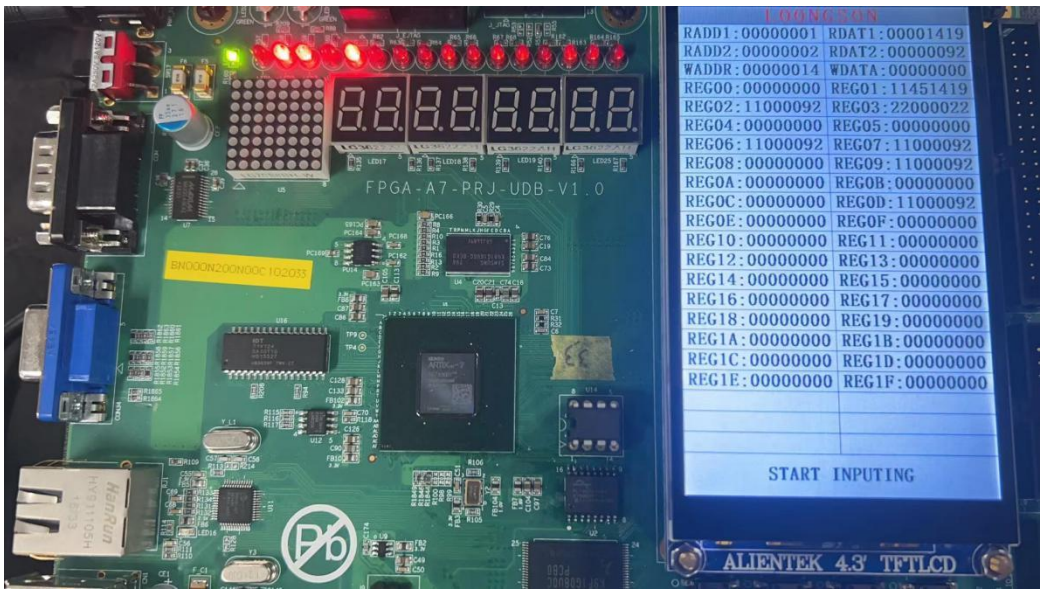
然后修改 ren 为 01，成功读取到 01 地址的寄存器数据的低 16 位 1419，如下图：



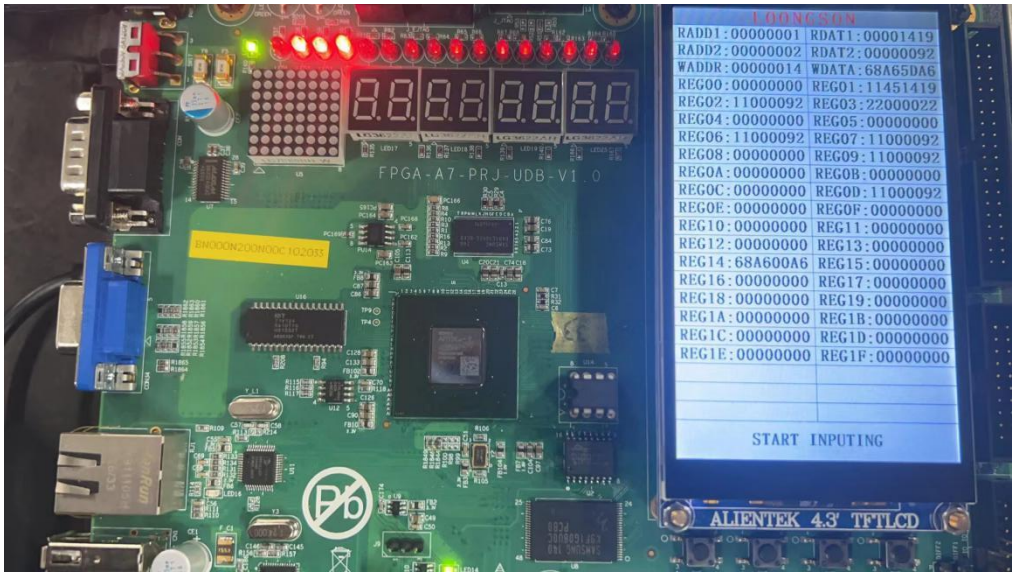
然后再修改 input_sel 为 01，写入地址 02，ren 设置不变，成功读取到地址 02 的数据的低 16 位 0092，如下图：



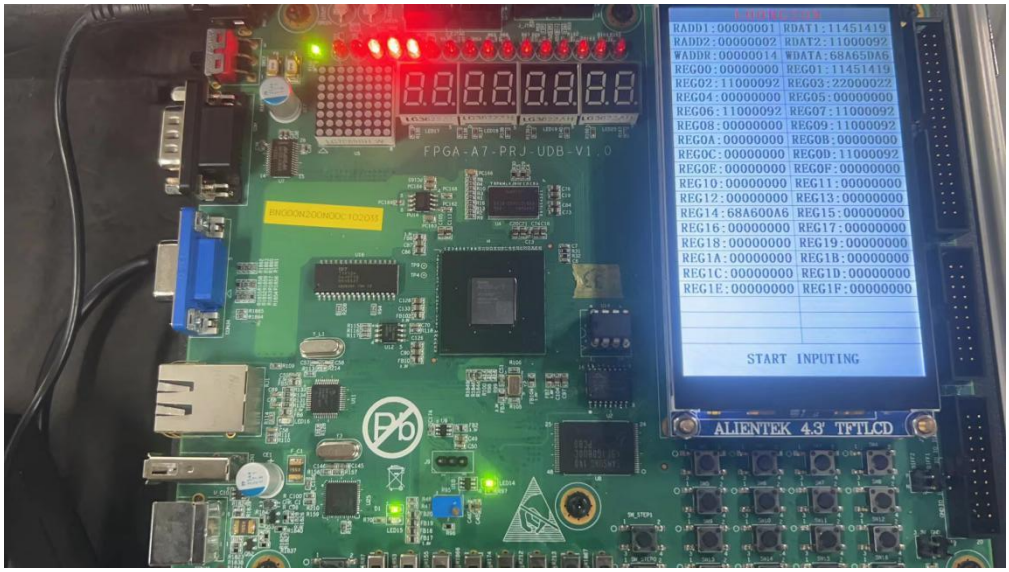
然后再更改 input_sel 为 10，写入 14，为写入数据的地址，如下图：



然后再更改 input_sel 为 11，写入数据 68A65DA6，设置 wen[3]到 wen[0]为 1101，发现成功只写入了第 4,3,1 字节，如下图：



补充：然后再更改 input_sel 为 00，写入 01 地址，然后设置 ren 为 11，可以成功读取到 32 位数据，如下图：



综上可以说明代码成功的实现了本次实验的要求。完成了实验任务。

6、 总结感想

在这个寄存器堆实验中，我学习了 MIPS 计算机中寄存器堆的设计和原理，掌握了 verilog 语言进行电路设计的基本方法和技巧。通过实验，根据修改要求，我还学习了如何更改输入端口的写入数据的方式，如何对 LCD 触摸屏模块修改显示寄存器堆的读写端口地址和数据，并最后成功完成了本次试验的修改的要求。

在实验中，我遇到了一些挑战，比如，在修改代码的过程，有一些小的语法错误，而且在修改 wen 写入数据方式时候写错代码，导致烧录到实验箱之后无法正确完成目标功能。但是，通过不断的尝试和调试，我最终成功地完成了实验，并验证了我的设计结果。

通过这个实验，我不仅学到了寄存器堆的设计和 verilog 语言的使用方法，还提高了自己解决问题的能力。同时，我对 vivado 的使用更加熟练，对 verilog 代码的编写能力也进一步提高。为今后的实验夯实基础。