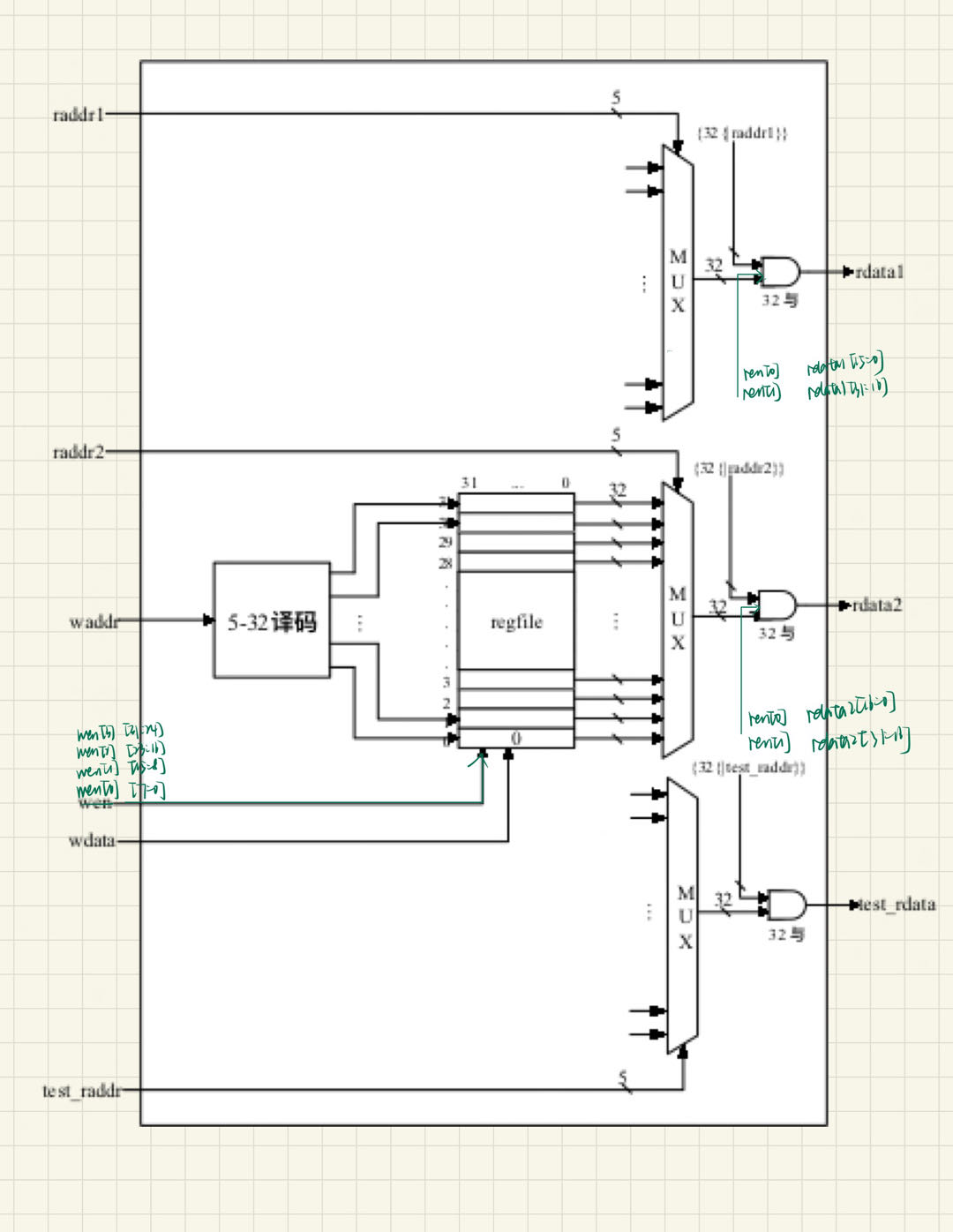
**组成原理实验课程第 3 次实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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**
2. 熟悉并掌握MIPS计算机中寄存器堆的原理和设计方法。
3. 初步了解MIPS指令结构和源操作数/目的操作数的概念。
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。
6. **实验设备**
7. 装有Xilinx Vivado的计算机一台。
8. LS-CPU-EXB-002教学系统实验箱一套。
9. **实验要求**
10. 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。
11. 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。
12. 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。
13. **实验内容说明**
14. 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。
15. 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。
16. 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。
17. 根据设计的实验方案，使用verilog编写相应代码。
18. 对编写的代码进行仿真，得到正确的波形图。
19. 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。
20. **实验原理图**



实验顶层模块框图：



1. **实验步骤**
2. **regfile.v修改**
3. 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 |
|  | ); |

1. 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。

1. 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。

1. **regfile\_display.v修改**
2. 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 |

1. 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 |

1. 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 |

1. **修改regfile.xdc文件**
2. 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] |

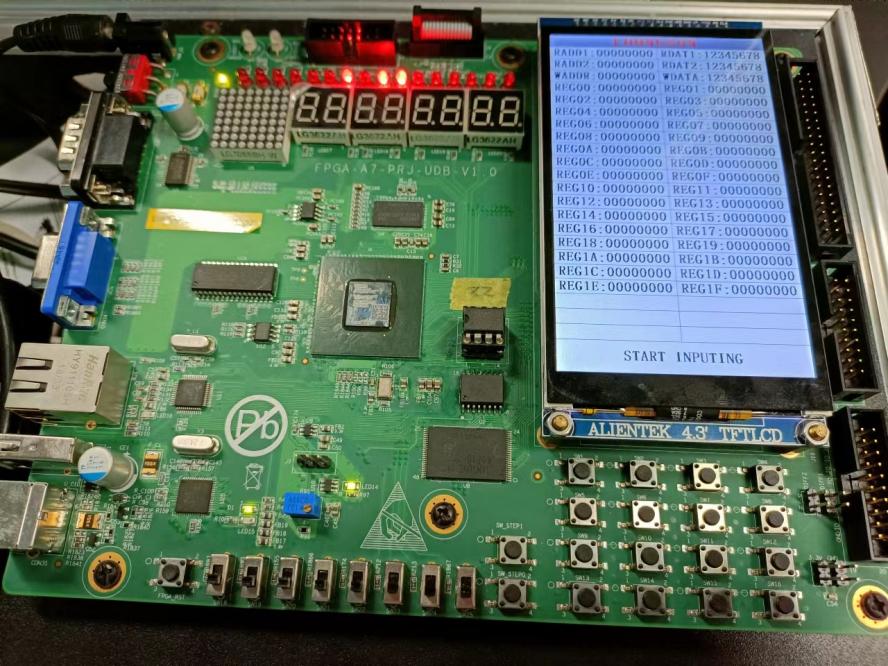
1. 拨码开关连接，用于输入

|  |  |
| --- | --- |
|  | #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]] |

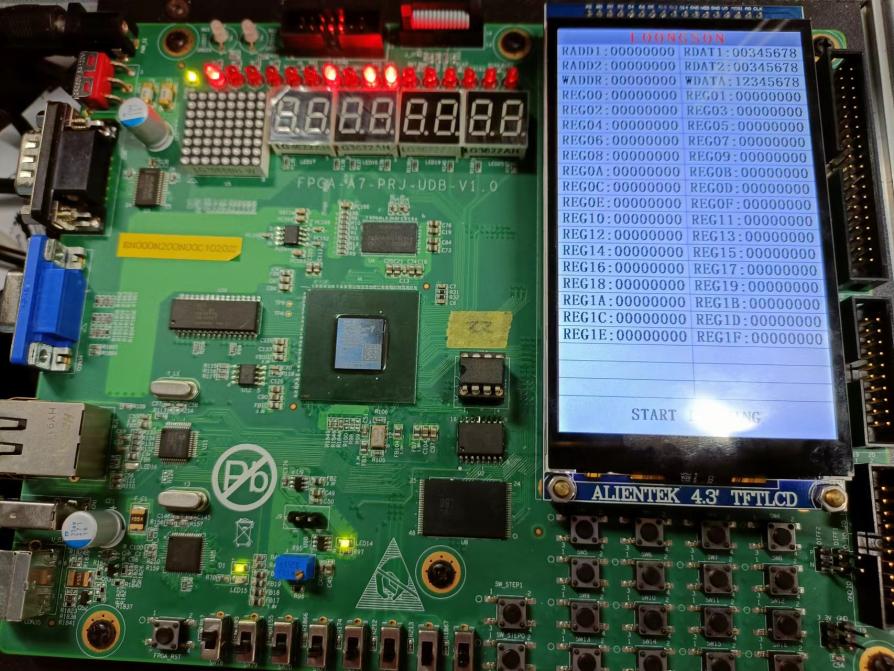
led灯从左到右依次为wen[3]，wen[2]，wen[1]，wen[0]，ren[1]，ren[0]，是否输入写地址，是否输入写数据，是否输入读地址1，是否输入读地址2。

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]。

1. **实验结果分析**
2. 输入12345678

****

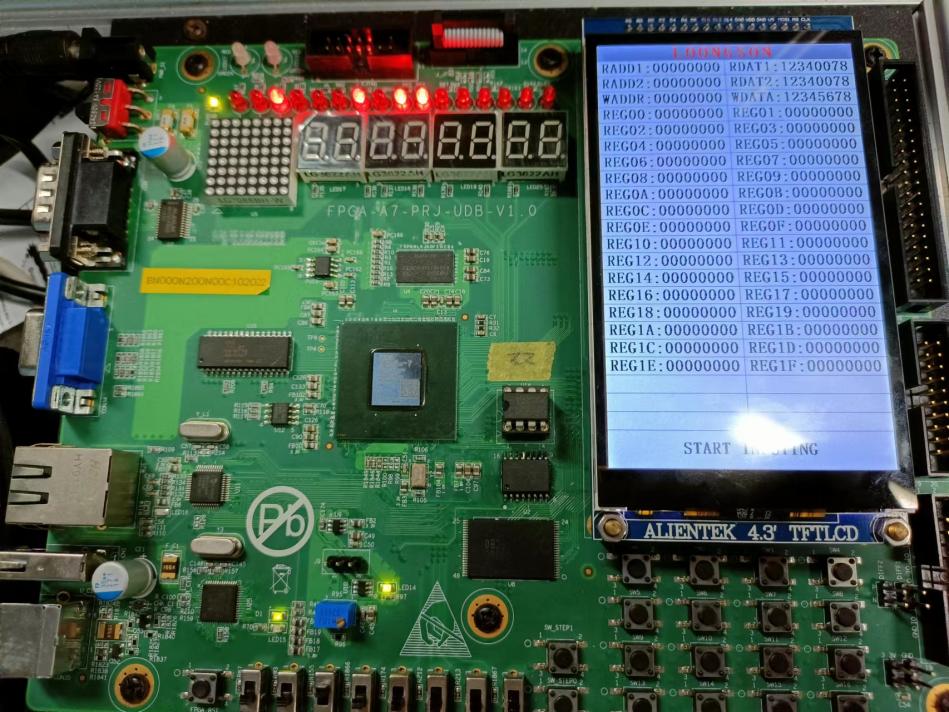
1. 拨码开关1，12位00



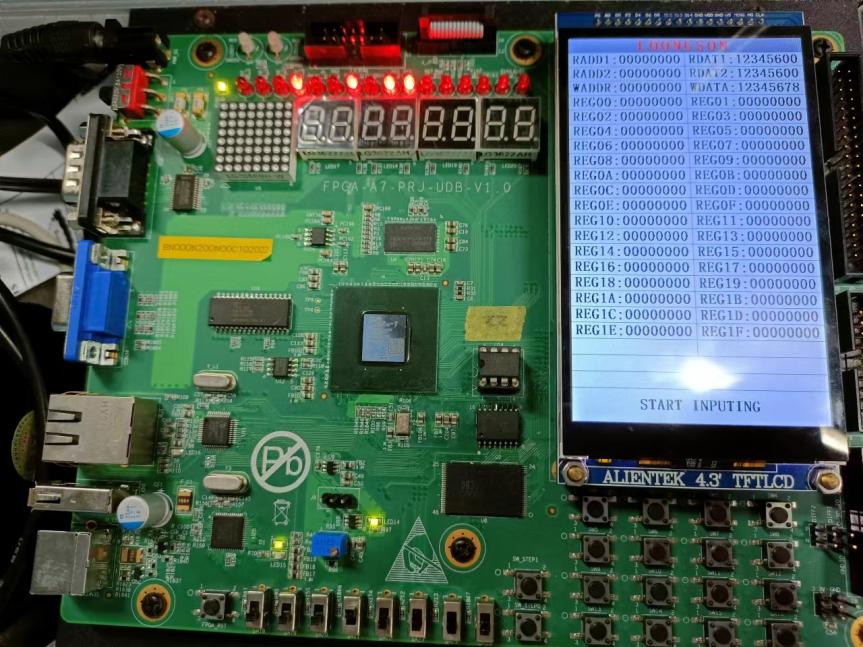
1. 拨码开关2，34位00



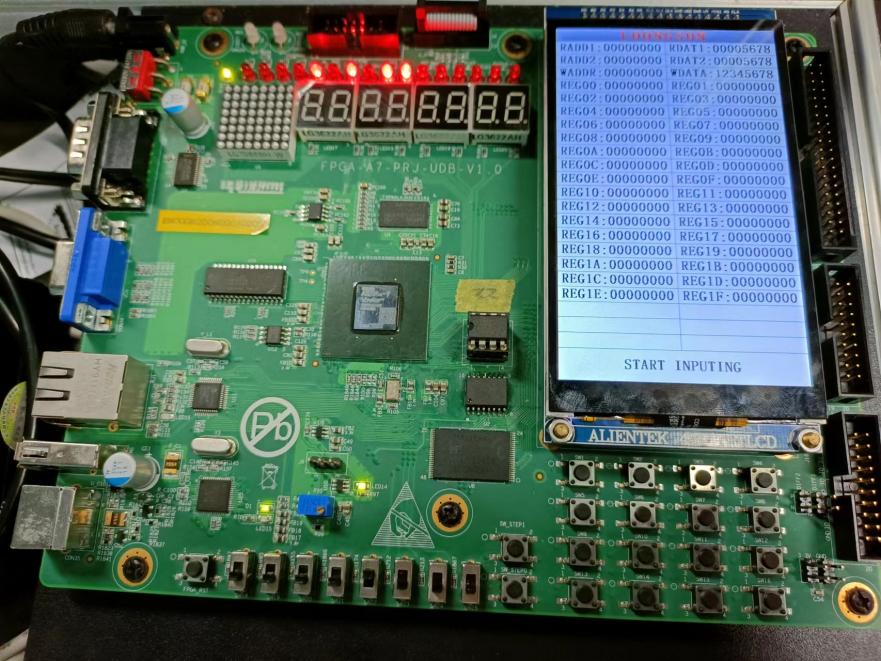
1. 拨码开关3，56位00



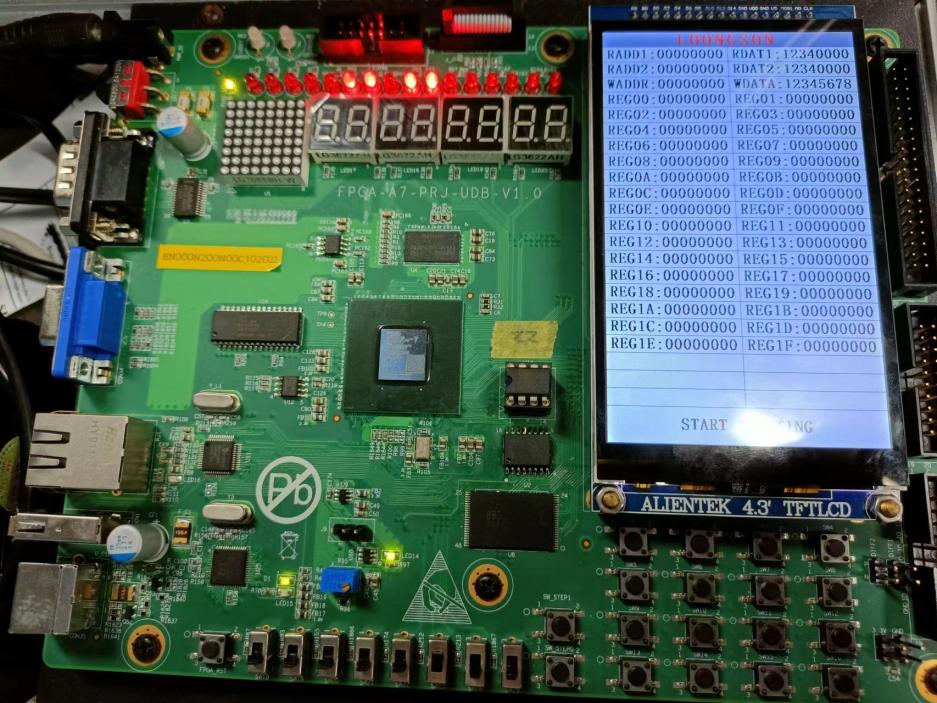
1. 拨码开关4，78位00



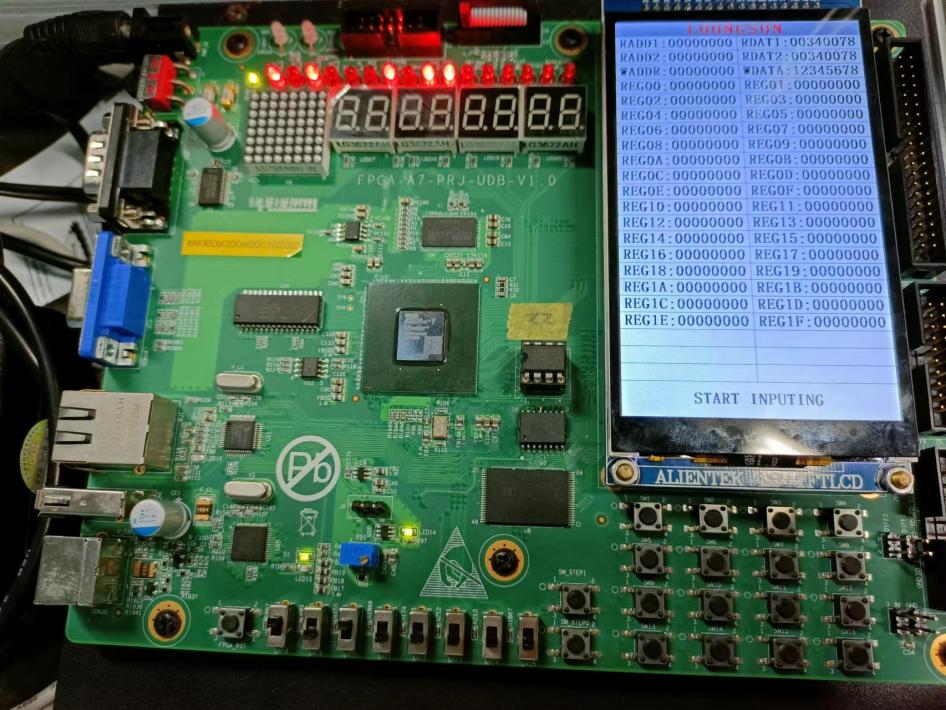
1. 拨码开关5，1234位0000



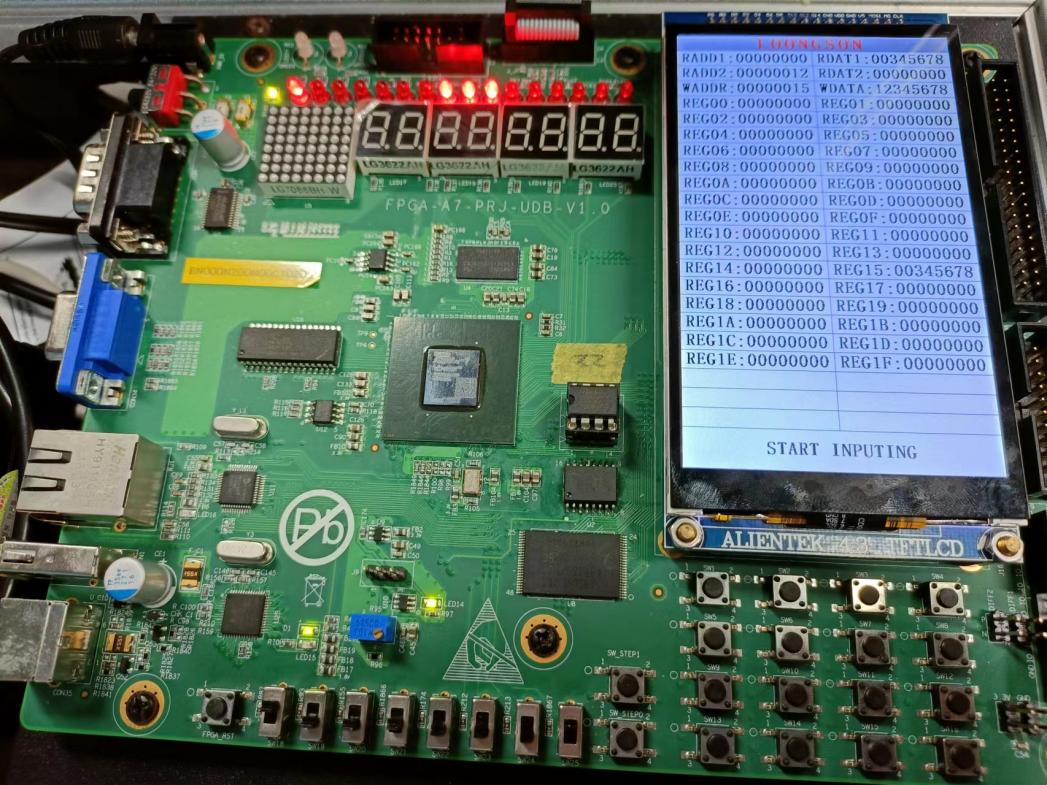
1. 拨码开关6,5678位0000



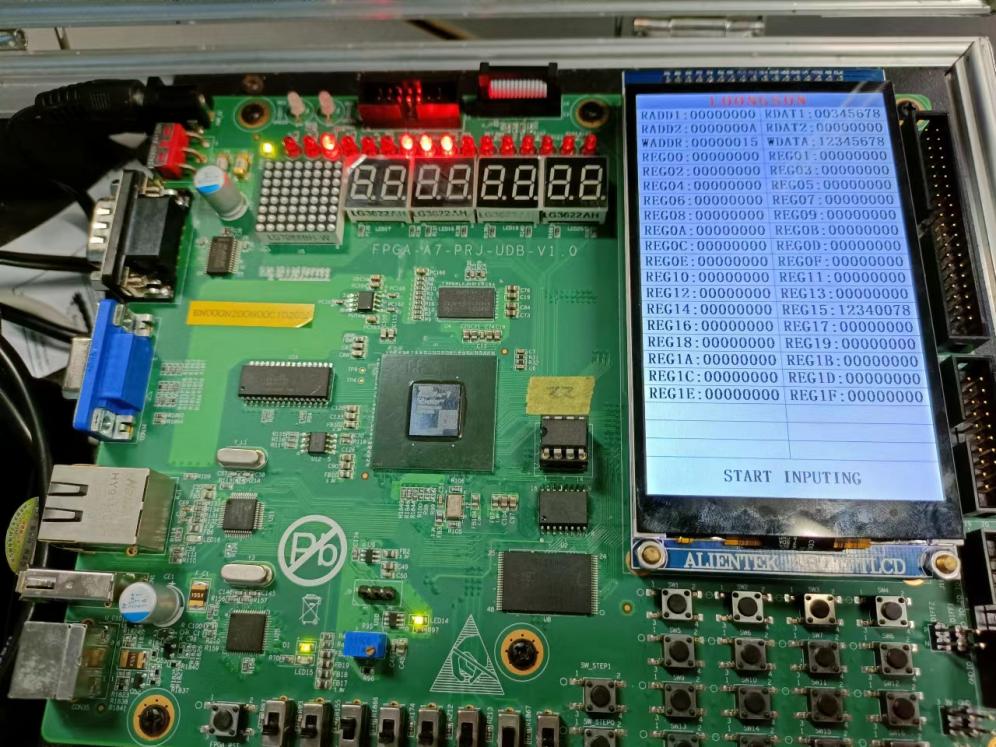
1. 拨码开关1、3,1256位0000



1. 拨码开关1、8，reg15写进00345678



1. 再拨动拨码开关3，reg15 12340078



1. **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。