**组成原理实验课程第 3次实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验名称 | 寄存器堆实现 | | | 班级 | 李涛 |
| 学生姓名 | 张丛 | 学号 | 2113662 | 指导老师 | 董前琨 |
| 实验地点 | 实验楼A区308 | | 实验时间 | 2023.04.18 | |

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。

1. **实验内容说明**

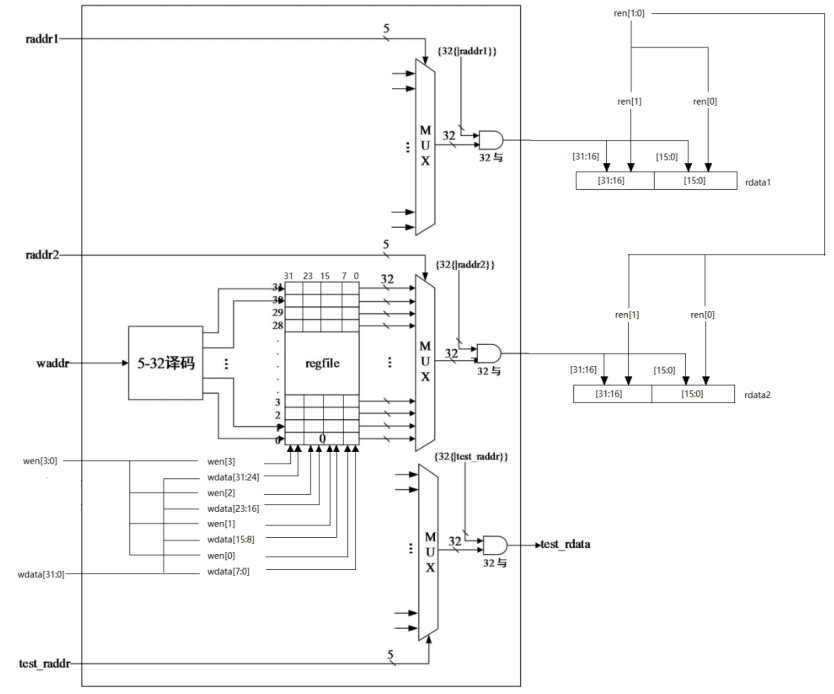
1. 将原有的寄存器堆的写操作进行改进，使用 4 位 wen 控制信号，对应写入 wdata 的四个字节，比如 wen 为“1011”时，写入第 4、2、1 三个字节。

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

1. **实验原理图**



1. **实验步骤**

1. 将 wen 设置为 4 位二进制输入，并添加 2 位二进制输入 ren：

input [3 :0] wen,//写入控制

input [1 :0] ren,//读取控制

2. 判断 wen[3:0]每一位上是否为 1，是 1 则输出更改字节，否则不更改

always @(posedge clk)

begin

if(wen[0])//控制写入数据的最高8位，次高8位，次低8位和最低8位

begin

rf[waddr][7:0]=wdata[7:0];

end

if(wen[1])

begin

rf[waddr][15:8]=wdata[15:8];

end

if(wen[2])

begin

rf[waddr][23:16]=wdata[23:16];

end

if(wen[3])

begin

rf[waddr][31:24]=wdata[31:24];

end

end

3. 通过判断 ren[1:0]的每一位上是否为 1，若为 1 则输出对应半字，否则不输出。

always @(\*)

begin

if(!raddr2)

begin

rdata2 <= 32'd0;

end

if(raddr2 && ren[0])

begin

rdata2[15:0] <=rf[raddr2][15:0];

end

if(raddr2 && ren[1])

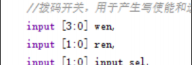
begin

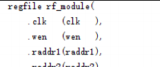
rdata2[31:16] <=rf[raddr2][31:16];

end

end

1. 更改display文件：





1. 修改约束文件：

首先我们要进行拨码开关的连接：

#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 readwhat[1]]

set\_property PACKAGE\_PIN Y6 [get\_ports readwhat[0]]

在电路板上便是从左到右的八个拨码开关前四个为wen信号，中间两个为input\_sel主要控制的是输入raddr1,rddr2,waddr还是wdata。

然后我们要将各个信号所在的引脚电气标准进行设置，统一设置为LVCMOS33，具体代码如下：

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]]

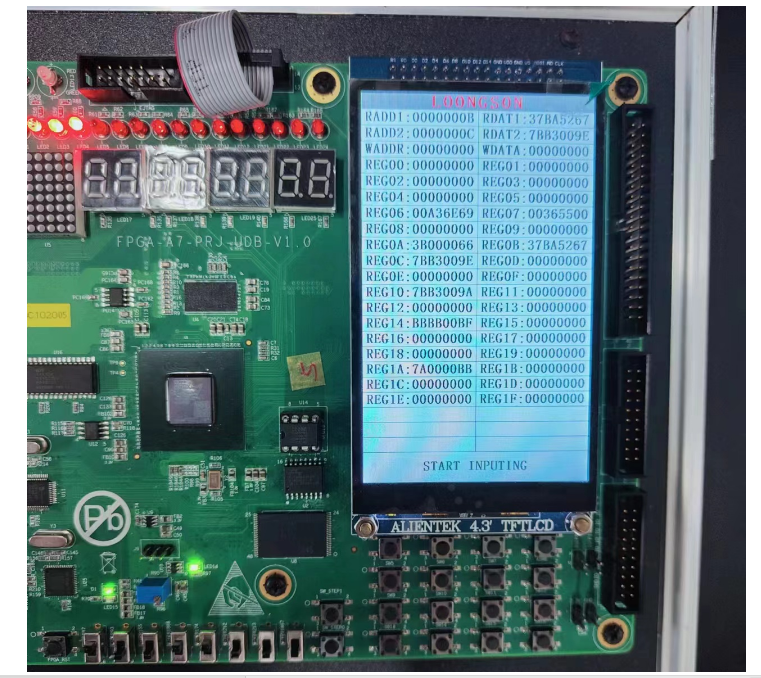
set\_property IOSTANDARD LVCMOS33 [get\_ports input\_sel[1]]

set\_property IOSTANDARD LVCMOS33 [get\_ports input\_sel[0]]

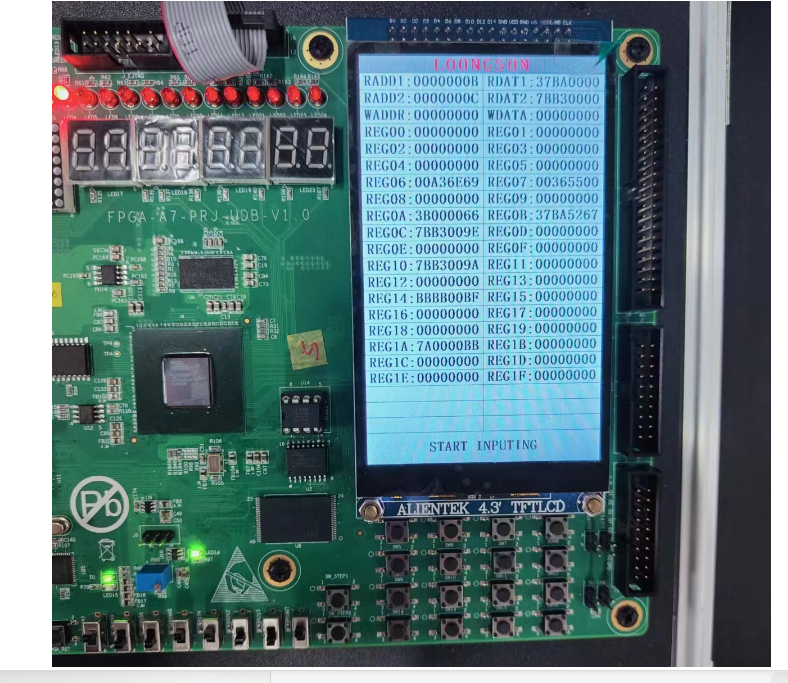
set\_property IOSTANDARD LVCMOS33 [get\_ports readwhat[1]]

set\_property IOSTANDARD LVCMOS33 [get\_ports readwhat[0]]

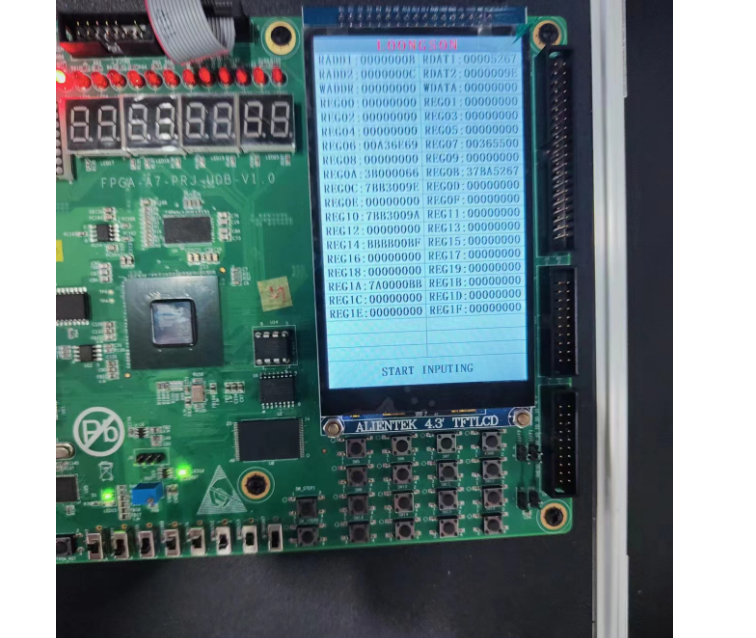
1. **实验结果分析**



如图，我们的第10号寄存器中的7A0000BB便是我们当时只写入第一个字节和第四个字节的结果，我们的读取第11号和第12号寄存器中的存取值也完全正确。



本图中我们将readwhat设置为10，即只读高16位，图中明显正确，第11号和第12号寄存器只读了高16位的值。



此图中readwhat设置01，即读入低16位，在读入第11号寄存器和第12号寄存器时我们观察实验板可以发现确实只读入了低16位。

1. **总结感想**

学习了寄存器堆的原理和设计方法，了解了MIPS指令结构和源操作数、目的操作数的概念，实验了寄存器堆的读写机制。

初步掌握了.xdc的编写方法。