

数字逻辑与计算机组成实验

- 实验名称: lab5 寄存器与存储器
- 院系: 计算机科学与技术系
- 学生姓名: 徐简
- 学号: 202220013
- 班级: 数字逻辑与计算机组成实验1班
- 邮箱: 161200063@mail.nju.edu.cn
- 实验时间: 2022 年 4 月 6 日

数字逻辑与计算机组成实验Lab5

202220013 徐简 161200063@smail.nju.edu.cn

一, 实验目的

- 复习寄存器与存储器的相关知识
- 设计一个寄存器与存储器RAM
- 学习Quartus中IP核的使用

二, 实验原理

寄存器组 (Register File) 与存储器 (Memory) 是数字系统中的记忆器件，用来存放程序和数据。从程序员的角度来看，CPU 的状态由其寄存器及存储器中的信息唯一确定。其中寄存器包括程序计数器 PC、通用寄存器，存储器指主存。

思考题：

会使用RAM模块。如果修改为非阻塞赋值，行为会发生变化。非阻塞赋值会等到写使能无效的时候，才能输出。而之前的做法，读数据不受写信号we的影响。

三, 实验环境/器材

- Quartus
- DE10-Standard开发平台
- FPGA开发板

四, 程序代码/流程图

总体，有总使能KEY和有选择不同存储器KEY，读写公用一个地址，写数据为两位，读结果实时展示在数码管上。

寄存器堆

首先完成一个数码管的设计，原理和之前几次实验一样。

```
module seg_7_out(in_4bit, out_seg);
    input [3:0] in_4bit;
    output reg [6:0] out_seg;

    always @ (in_4bit) begin
        case (in_4bit)
            0 : out_seg = 7'b1000000;
            1 : out_seg = 7'b1111001;
            2 : out_seg = 7'b0100100;
            3 : out_seg = 7'b0110000;
            4 : out_seg = 7'b0011001;
            5 : out_seg = 7'b0010010;
            6 : out_seg = 7'b0000010;
            7 : out_seg = 7'b1111000;
            8 : out_seg = 7'b0000000;
            9 : out_seg = 7'b0010000;
```

```

10 : out_seg = 7'b0001000;
11 : out_seg = 7'b0000011;
12 : out_seg = 7'b1000110;
13 : out_seg = 7'b0100001;
14 : out_seg = 7'b0000110;
15 : out_seg = 7'b0001110;
default: out_seg = 7'bx;
endcase
end
endmodule

```

下面介绍一下寄存器堆的设计。输入信号有时钟，选择信号，写使能信号，地址，以及写数据，输出为数码管。

因此设计的逻辑如下：

1. initial中初始化寄存器堆 reg [7:0] ram [15:0]
2. 读取时不需要时钟控制，即读地址有效后，直接输出数据。 assign dout_vec = ram[addr]
3. 写入时通过时钟上升沿进行控制。 if (en && we)

```

module ram1(clk, en, we, addr, din, dout_h, dout_l);
    input clk;
    input en, we;
    input [3:0] addr;
    input [1:0] din;
    output wire [6:0] dout_h, dout_l;

    reg [7:0] ram [15:0];
    wire [7:0] dout_vec;
    wire [6:0] dout_h_tmp, dout_l_tmp;

    initial begin
        $readmemh("D:/My_design/lab5/mem1.txt", ram, 0, 15);
    end
    assign dout_vec = ram[addr];
    seg_7_out s1(dout_vec[7:4], dout_h_tmp);
    seg_7_out s2(dout_vec[3:0], dout_l_tmp);
    assign dout_h = dout_h_tmp;
    assign dout_l = dout_l_tmp;
    always @ (posedge clk) begin
        if (en && we) ram[addr] <= {6'b0, din};
    end

endmodule

```

RAM

1. 按照手册中的步骤，利用IP核生成ram1port。
2. 在ram2模块中实例化ram1port，这里需要注意使能段的设计，由于有两个存储器，因此需要选择RAM和使能EN同时有效时，才能往ram2中写数据。

```
assign ram_en = en & we;
ram1port r1(
    .address(addr),
    .clock(clk),
    .data(din_full),
    .wren(ram_en),
    .q(dout_vec));
```

五, 实验步骤/过程

- 使用DE10软件建立工程，这样可以自动进行引脚分配
- 编写代码，并在项目中实例化
- 进行测试
- 上板验证

六, 测试方法

- 仿真：简单的测试了一下两个存储器的读写，主要的测试还是在上板部分。

```
initial begin
    KEY[1] = 1;
    KEY[0] = 0; // write
    SW[9:8] = 4'h0;
    SW[3:0] = 4'h1;
    #5;
    KEY[0] = 1; // read
    #5;

    KEY[1] = 0;
    KEY[0] = 0; // write
    SW[9:8] = 4'h0;
    SW[3:0] = 4'h1;
    #5;
    KEY[0] = 1; // read
    #5;
    $stop;
end
```

- 上板测试：写入fpga开发板，根据功能测试输入输出的情况

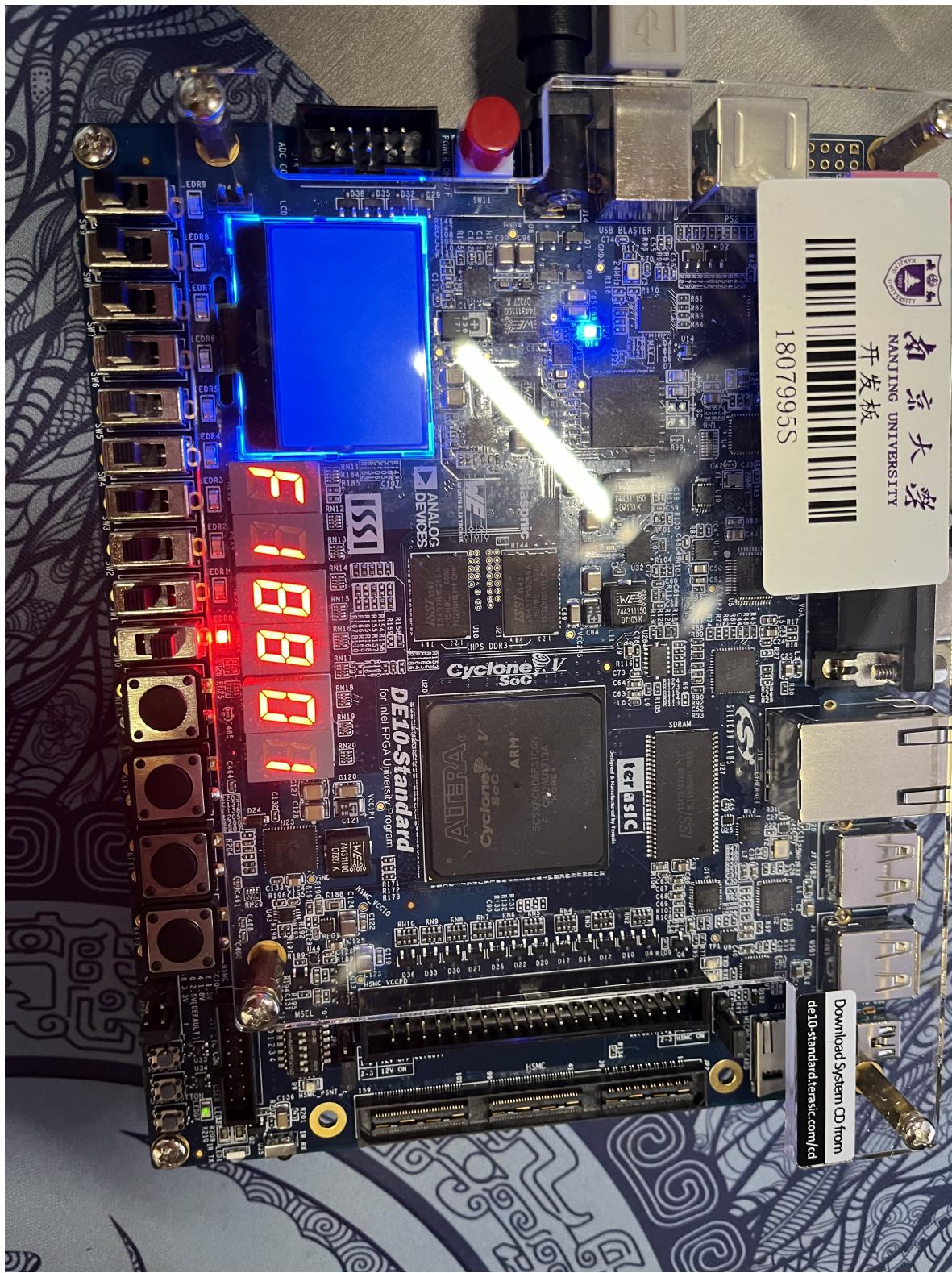
七, 实验结果

FPGA上板，进行了充分的测试（弥补了仿真orz

分别往两个存储器中写入3后。



读取1号位置存储的数据，可以发现符合未修改的初始化数据。



八、遇到的问题和解决办法

- 开发板接口有限，按照手册的建议，写入的时候只写入两位，
- 上板操作视频太大了

九、实验得到的启示

- 注意变量类型以及赋值语句的使用，会对实验结果产生很大的影响

十、意见和建议

- 希望有关于输出缓存的介绍