

# 浙江大学

## 本科实验报告

课程名称: 数字逻辑设计

姓 名: 颜晗

学 院: 计算机科学与技术学院

系:

专 业: 计算机科学与技术

学 号: 3200105515

指导教师: 蔡铭

2022 年 1 月 1 日

# 浙江大学实验报告

课程名称： 数字逻辑设计 实验类型： 综合

实验项目名称： 多路选择器设计及应用

学生姓名： 颜晗 专业： 计算机科学与技术 学号： 3200105515

同组学生姓名： \_\_\_\_\_ 指导老师： 蔡铭

实验地点： 东四 509 实验日期： 2021 年 12 月 27 日

## 一、实验目的和要求

掌握支持并行输入的移位寄存器的工作原理

掌握支持并行输入的移位寄存器的设计方法

## 二、实验内容和原理

### 实验内容：

任务 1：设计 8 位带并行输入的右移移位寄存器

任务 2：设计主板 LED 灯驱动模块

任务 3：设计主板七段数码管驱动模块

### 实验原理：

#### 1. 移位寄存器

移位寄存器的作用即执行移位操作；每来一个时钟脉冲，寄存器中的数据按顺序向左或向右移动一位。（采用触发器进行设计）移位方式分为左移、右移、循环移位。另外还包括数据输入输出方式：串行输入，串行输出；串行输入，并行输出；并行输入，串行输出。

移位的实质即将寄存器中一个 D 触发器的输出接到相邻（左、右）另一触发器的输入，如此经过一次时钟脉冲，寄存器的值便会转移一次。对于左(右)移位寄存器，在边缘处还需要外界提供一个额外的输入用来填补右(左)端的空缺，对于循环寄存器就不需要额外的输入了。

简单的左移移位寄存器代码如下：

```
module shift_reg(
```

```

input wire clk, s_in,
    output wire [7:0] s_out;

reg [7:0] temp;

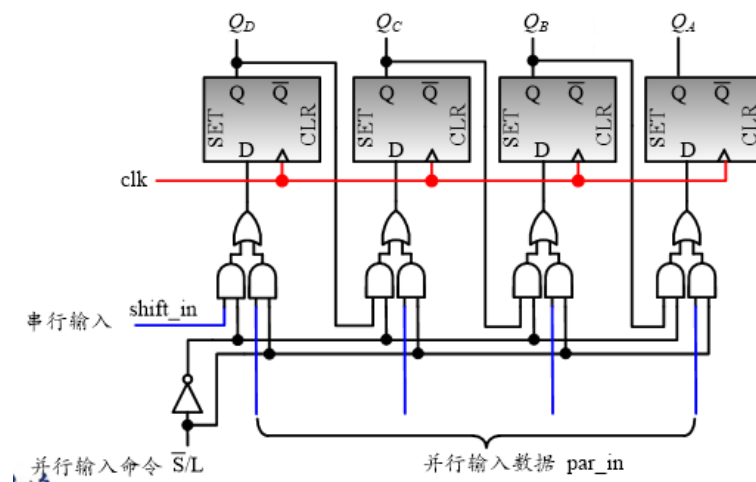
always @ (posedge clk)
    begin
        temp <= {temp[6:0],s_in};
    end

assign s_out = temp;
endmodule

```

## 2. 带并行输入的移位寄存器

寄存器还需要能够接受外界的数据，因此设计时还需加入多路复用器来选择寄存器是将移位操作的数据作为输入，还是接受外来并行输入数据。



通过并行输入命令作为选择信号，可使寄存器完成设置初值和移位功能。结构化描述代码如下：

```

module shift_regb8(
    input wire clk, S_L, s_in,
    input wire [7:0] p_in,
    output wire [7:0] Q
);
    wire inv_s1;
    wire s1,s2,s3,s4,s5,s6,s7,s8;
    wire p1,p2,p3,p4,p5,p6,p7,p8;
    wire d1,d2,d3,d4,d5,d6,d7,d8;

    INV inv(.I(S_L),.O(inv_s1));

    AND2 S8(s8,inv_s1,s_in),
        S7(s7,inv_s1,Q[7]),
        S6(s6,inv_s1,Q[6]),
        S5(s5,inv_s1,Q[5]),
        S4(s4,inv_s1,Q[4]),
        S3(s3,inv_s1,Q[3]),
        S2(s2,inv_s1,Q[2]),
        S1(s1,inv_s1,Q[1]);

    AND2 P1(p1,S_L,p_in[0]),
        P2(p2,S_L,p_in[1]),
        P3(p3,S_L,p_in[2]),
        P4(p4,S_L,p_in[3]),
        P5(p5,S_L,p_in[4]),
        P6(p6,S_L,p_in[5]),
        P7(p7,S_L,p_in[6]),
        P8(p8,S_L,p_in[7]);

    OR2 D8(d8,s8,p8),
        D7(d7,s7,p7),
        D6(d6,s6,p6),
        D5(d5,s5,p5),
        D4(d4,s4,p4),
        D3(d3,s3,p3),
        D2(d2,s2,p2),
        D1(d1,s1,p1);

    FD fd8(.C(clk),.D(d8),.Q(Q[7])),
        fd7(.C(clk),.D(d7),.Q(Q[6])),
        fd6(.C(clk),.D(d6),.Q(Q[5])),
        fd5(.C(clk),.D(d5),.Q(Q[4])),
        fd4(.C(clk),.D(d4),.Q(Q[3])),
        fd3(.C(clk),.D(d3),.Q(Q[2])),
        fd2(.C(clk),.D(d2),.Q(Q[1])),
        fd1(.C(clk),.D(d1),.Q(Q[0]));

endmodule

```

### 3. 并行—串行转换器

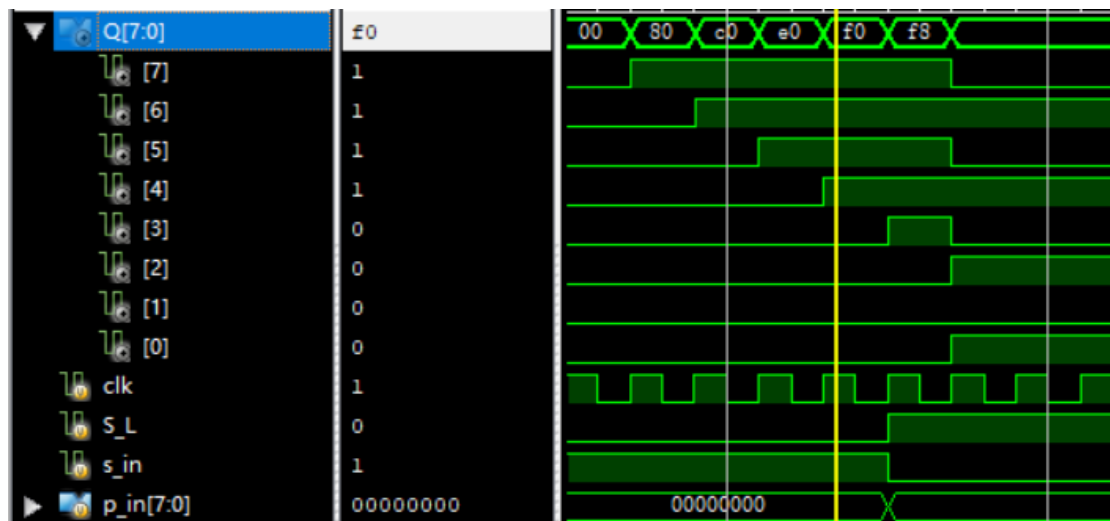
并行—串行转换器可以接受外来并行输入，并在时钟脉冲的指示下，将最低位串行输出，如下图为 7 位实例。在没有启动命令，即 `start` 为 0 时，不妨先假设寄存器输出结果相与为 0，那么易得到移位寄存器的并行输入命令为 0，即寄存器不断执行移位而不从外界加载数据。而对于右移寄存器，需每次从最左端输入一个 1 填补最高位移走后的空缺，经过几次移位后，寄存器输出的与结果变为 1，即电路达到稳定状态。

而启动命令有效时，移位寄存器的并行输入命令也生效，接受外来的 6 位数据和最高位 0 数据，由于最高位输入的 0 存在，寄存器与结果也为 0，导致下一周期寄存器的并行输入命令再次无效，寄存器不断移位，直至高位 0 移至最低位，与结果再次为 1，若启动命令未生效，寄存器就此稳定下来，虽然实际依旧在移位，但是寄存器的输出为全 1 不变。

注意，7 位的并行转换器只接受 6 位的并行输入，剩余的最高位留给 0 来控制转换完成的信号。



### 3. 仿真结果



根据仿真结果，并行输入命令无效时，实现移位，并行输入命令生效时，将输入数据加载进寄存器中。

#### 任务 2：设计主板 LED 灯驱动模块

1. 将所需模块添加进工程，修改任务一中的移位寄存器（LED 灯需要 17 位的移位寄存器，或者 8+9 位移位寄存器）。
2. 根据修改的移位寄存器，设计 LED 灯驱动。

```
module LED_DRV(  
    input wire clk,  
    input wire [15:0] LED,  
    input wire start,  
    output LEDDT,  
    output LEDCLK,  
    output LEDCLR,  
    output LEDEN  
);  
    assign LEDEN=1'b1;  
    assign LEDCLR=1'b1;  
    assign LEDDT=Q2[0];  
    wire [8:0] Q1;  
    wire [7:0] Q2;  
    nor s1(s_l,r_l,a_l),  
        r1(r_l,nfinish,s_l);  
    and a1(a_l,start,finish),  
        a2(finish,x1,x2),  
        and8_1(x1,Q1[8],Q1[7],Q1[6],Q1[5],Q1[4],Q1[3],Q1[2],Q1[1],Q1[0]),  
        and8_2(x2,Q2[7],Q2[6],Q2[5],Q2[4],Q2[3],Q2[2],Q2[1]);  
    or o1(LEDCLK,clk,finish);  
    not n1(nfinish,finish);  
    shift_regb9 srl(.clk(clk),.S_L(r_l),.s_in(1'b1),.p_in({1'b0,LED[15:8]}),.Q(Q1));  
    shift_regb8 sr2(.clk(clk),.S_L(r_l),.s_in(Q1[0]),.p_in(LED[7:0]),.Q(Q2));  
endmodule
```

3. 根据所需功能设计实现顶层模块

```

module Top(
    input wire clk,
    input wire rst,
    input BN,
    input wire [15:0] SW,
    input wire [3:0] BTN,
    output K_ROW,
    output [3:0] AN,
    output [7:0] SEGMENT,
    output LEDCLK, LEDCLR,
    output LEDEN,
    output LEDDT
);
    wire [31:0] clkdiv;
    wire [3:0] btn_out;
    wire [15:0] num;
    wire [15:0] renum;

    clkdiv f0(.clk(clk),.rst(rst),.clkdiv(clkdiv));

    pbdebounce p1(clkdiv[17], BTN[0], btn_out[0]);
    pbdebounce p2(clkdiv[17], BTN[1], btn_out[1]);
    pbdebounce p3(clkdiv[17], BTN[2], btn_out[2]);
    pbdebounce p4(clkdiv[17], BTN[3], btn_out[3]);

    createNumber f1(.btn(btn_out),
                    .SW(SW[3:0]),
                    .num(num));

    Disp_num f2(.clk(clk),
                .RSTN(rst),
                .num(num),
                .SEGMENT(SEGMENT),
                .AN(AN));

    re_num f4(num,renum);

    LED_DRV f3(.clk(clkdiv[23]),
                .LED(renum),
                .start(SW[15]),
                .LEDDT(LEDDT),
                .LEDCLK(LEDCLK),
                .LEDCLR(LEDCLR),
                .LEDEN(LEDEN));

    BUF f8(.I(BN), .O(K_ROW));
endmodule

```

4. 进行综合和写引脚约束文件等操作。

任务 3：设计主板七段数码管驱动模块

1. 修改任务一中的移位寄存器（65 位移位寄存器或  $8*7+9*1$  位移位寄存器）
2. 设计数码管的驱动，分为翻转和实际输入两部分。

```

module SEG_DRV(
    input clk,
    input start,
    input [31:0] num,
    input CR,
    output SEGDT,
    output SEGCLK,
    output SEGCLR
);
    wire [63:0] cnum;
    or ol(SEGCLK,finish,clk);
    assign SEGCLR=CR;

    //完成num的翻转
    segment7 y1(num[3:0],1'b0,1'b0,cnum[63:56]);
    segment7 y2(num[7:4],1'b0,1'b0,cnum[55:48]);
    segment7 y3(num[11:8],1'b0,1'b0,cnum[47:40]);
    segment7 y4(num[15:12],1'b0,1'b0,cnum[39:32]);
    segment7 y5(num[19:16],1'b0,1'b0,cnum[31:24]);
    segment7 y6(num[23:20],1'b0,1'b0,cnum[23:16]);
    segment7 y7(num[27:24],1'b0,1'b0,cnum[15:8]);
    segment7 y8(num[31:28],1'b0,1'b0,cnum[7:0]);

    p2s_b64 x1(clk,start,cnum[63:0],finish,SEGDT)
endmodule

module p2s_b64(
    input clk,
    input start,
    input [63:0] cnum,
    output finish,
    output SEGDT
);
    assign SEGDT=Q8[0];

    wire [8:0] Q1;
    wire [7:0] Q2;
    wire [7:0] Q3;
    wire [7:0] Q4;
    wire [7:0] Q5;
    wire [7:0] Q6;
    wire [7:0] Q7;
    wire [7:0] Q8;
    nor s1(s_l,r_l,a_l),
    rl(r_l,nfinish,s_l);

    and a1(a_l,start,finish),
    a2(finish,x1,x2,x3,x4,x5,x6,x7,x8),
    and9_1(x1,Q1[8],Q1[7],Q1[6],Q1[5],Q1[4],Q1[3],Q1[2],Q1[1],Q1[0]),
    and8_1(x2,Q2[7],Q2[6],Q2[5],Q2[4],Q2[3],Q2[2],Q2[1],Q2[0]),
    and8_2(x3,Q3[7],Q3[6],Q3[5],Q3[4],Q3[3],Q3[2],Q3[1],Q3[0]),
    and8_3(x4,Q4[7],Q4[6],Q4[5],Q4[4],Q4[3],Q4[2],Q4[1],Q4[0]),
    and8_4(x5,Q5[7],Q5[6],Q5[5],Q5[4],Q5[3],Q5[2],Q5[1],Q5[0]),
    and8_5(x6,Q6[7],Q6[6],Q6[5],Q6[4],Q6[3],Q6[2],Q6[1],Q6[0]),
    and8_6(x7,Q7[7],Q7[6],Q7[5],Q7[4],Q7[3],Q7[2],Q7[1],Q7[0]),
    and8_7(x8,Q8[7],Q8[6],Q8[5],Q8[4],Q8[3],Q8[2],Q8[1]);

    not n1(nfinish,finish);

    shift_regb9 left9(clk,r_l,1'b1,{1'b0,cnum[63:56]},Q1);
    shift_regb8 right8(clk,r_l,Q1[0],cnum[55:48],Q2);
    shift_regb8 right7(clk,r_l,Q2[0],cnum[47:40],Q3);
    shift_regb8 right6(clk,r_l,Q3[0],cnum[39:32],Q4);
    shift_regb8 right5(clk,r_l,Q4[0],cnum[31:24],Q5);
    shift_regb8 right4(clk,r_l,Q5[0],cnum[23:16],Q6);
    shift_regb8 right3(clk,r_l,Q6[0],cnum[15:8],Q7);
    shift_regb8 right2(clk,r_l,Q7[0],cnum[7:0],Q8);
endmodule

```



### 3. 设计顶层模块

```
module Top(
    input wire clk,
    input wire[15:0] SW,
    input wire [3:0] BTN,
    input  BN,
    output SEGDT,
    output SEGCLK,
    output SEGCLR,
    output SEGEN,
    output K_ROW,
    output [3:0] AN,
    output [7:0] SEGMENT
);
    assign SEGEN=1'b1;
    reg [31:0] num;
    wire [3:0] btn_out;
    wire [15:0] num_rr;
    reg [3:0] Ctrl;
    wire [31:0] clkdiv;
    assign S=SW[8];
    initial num <= 32'b0000_0001_0010_0011_0100_0101_0110_0111;

    pbdebounce p1(clkdiv[17], BTN[0], btn_out[0]);
    pbdebounce p2(clkdiv[17], BTN[1], btn_out[1]);
    pbdebounce p3(clkdiv[17], BTN[2], btn_out[2]);
    pbdebounce p4(clkdiv[17], BTN[3], btn_out[3]);

    clkdiv f0(.clk(clk), .rst(1'b0), .clkdiv(clkdiv));

    SEG_DRV S1(clkdiv[22], SW[15], num[31:0], 1'b1, SEGDT, SEGCLK, SEGCLR);

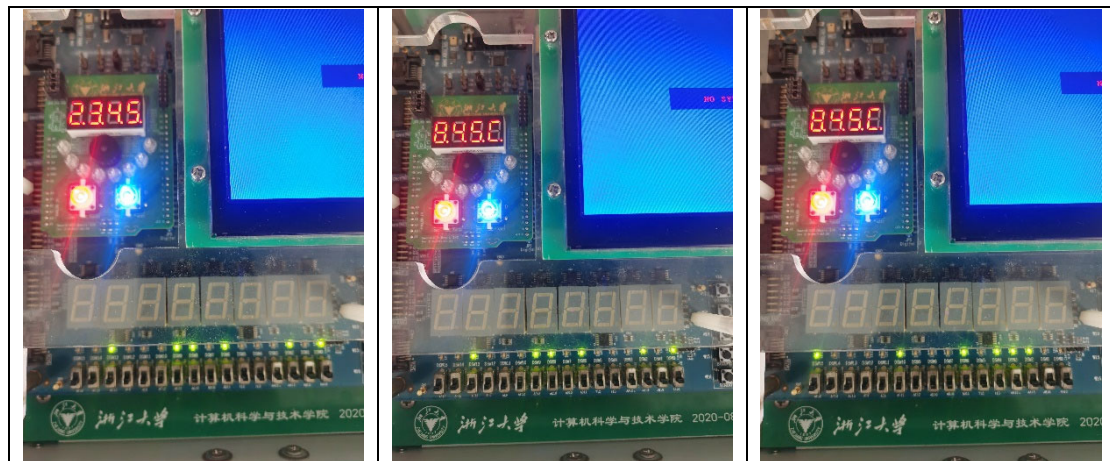
    BUF f8(.I(BN), .O(K_ROW));
    always@(*)begin
        if(SW[8]==0) begin
            num[15:0]<=num_rr;
            Ctrl<=SW[3:0];
        end
        else begin
            num[31:16]<=num_rr;
            Ctrl[3:0]<=SW[7:4];
        end
    end
    end
    createNumber C1(.btn(btn_out), .SW(Ctrl[3:0]), .num(num_rr));
    Disp_num D1(.clk(clk), .RSTN(1'b0), .num(num_rr), .SEGMENT(SEGMENT), .AN(AN));
endmodule
```

### 4. 进行综合和写引脚约束文件等操作。

## 四、实验结果分析

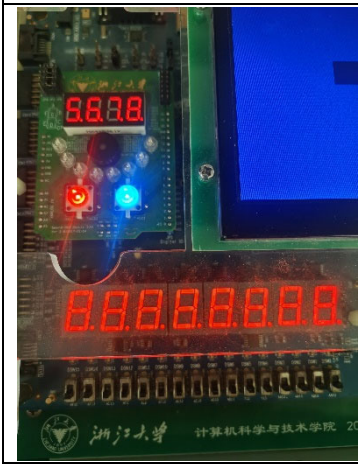
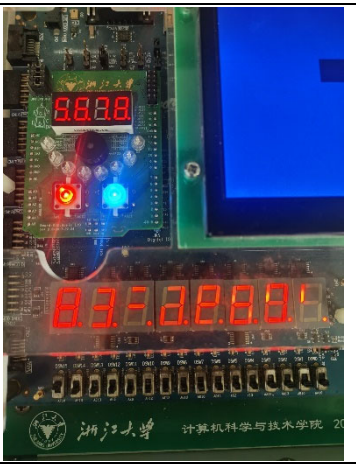
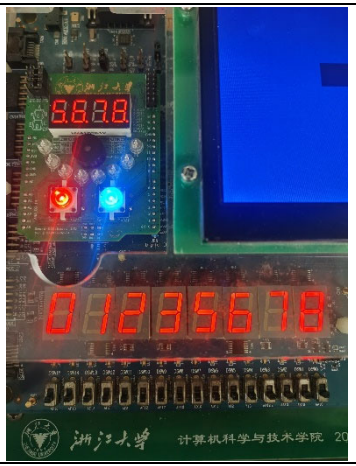
### 任务 2：设计主板 LED 灯驱动模块

四位一组，灯亮表示 1。从左至右分别与上面的 7 段数码管对应，可通过按钮改变数码管的值，再次启动 LED 灯输入并结束后便会和数码管对应起来。

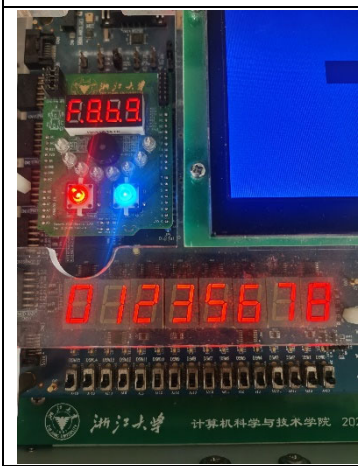
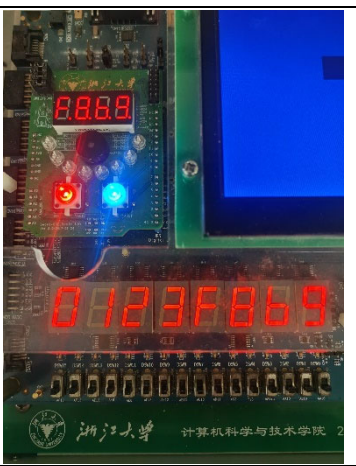
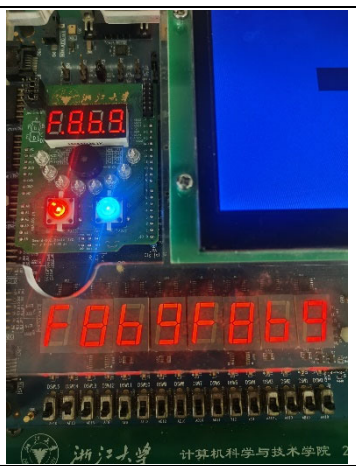




任务 3：设计主板七段数码管驱动模块

| 初始状态  | 输入过程  | 完成   |
|---|---|--|
|  |  |  |

可通过按钮对数据进行增减，另有开关控制对前四位进行增减还是后四位，拨动开关启动输入在结束后就可以看到变化。

| 改变源数字   | 后四位改变   | 前四位改变  |
|---|---|--|
|  |  |  |

五、实验总结与反思

本次实验我们实现了移位寄存器，并且用上了实验板上的两种显示工具。由于对于它们所使用的串行输入并行输出原理一直不太理解，本次实验花费了较长的时间理解 ppt，后经过同学的提点终于明悟。另外设计顶层模块时也由于对 Verilog 语法不熟悉，总是出现报错，经过不断查资料和尝试才最后成功。最后一个实验后，发现自己对 Verilog 和实验的各种硬件模块了解还是有限，为了接下来的课程着想，需要抽空多花时间弥补。