浙江大学

本科实验报告

课程名称:	计算机逻辑设计基础
姓　　名:	张晋恺
学　　院:	竺可桢学院
系:	所在系
专　　业:	计算机科学与技术
学　　号:	3230102400
指导教师:	董亚波

2024 年 5 月 28 日

浙江大学实验报告

课程名称: 计算机逻辑设计基础 实验类型: 综合

实验项目名称: 移位寄存器设计与应用

学生姓名: 张晋恺 专业: 计算机科学与技术 学号: 3230102400

同组学生姓名: 杨吉祥 指导老师: 董亚波

实验地点: 东 4-511 实验日期: 2024 年 5 月 29 日

一、实验目的

- 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

实验任务

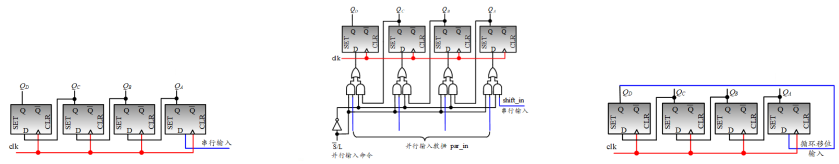
- 任务一：设计 8 位带并行输入的左移移位寄存器
- 任务二：设计主板 16 位 LED 灯驱动模块
- 任务三：设计主板 8 位数码管驱动模块

实验原理

移位寄存器

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

- 必须采用主从触发器或边沿触发器
- 不能采用锁存器
- 数据移动方式有左移、右移，循环移位等
- 数据输入输出方式有串行输入输出、并行输入输出等



(a) 串行输入寄存器　(b) 并，串行输入寄存器　(c) 循环移位寄存器

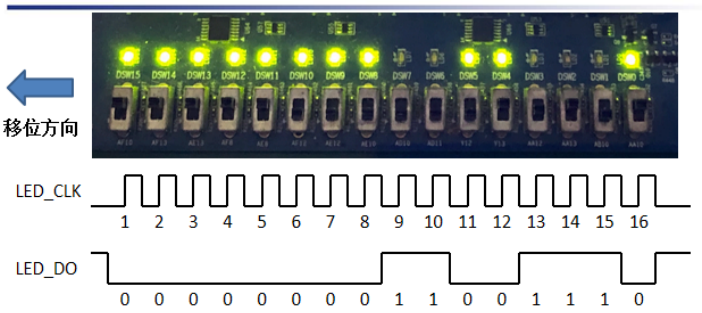
图 1: D 触发器构成的左移移位寄存器

实验板 16 位 LED 灯接口说明

16 为 LED 灯：　实验板上，采用 2 个 8 位移位寄存器 74LV164A 构成16 位串行左移移位寄存器，寄存器的并行输出控制 16 个 LED 灯 LED0-LED15

示意图

- LED_CLK:16 位 LED 灯模块的时钟，上升沿触发移位
- LED_CLR: 清零，使所有 LED 亮，低电平有效
- LED_DO: 串行移位数据输入，0 使 LED 亮
- LED_EN：LED 模块总控开关，1 为使能
- 用 LED_CLK 和 LED_DO 按顺序 LED15，LED14，……，LED1，LED0 串行移入 16 位数据



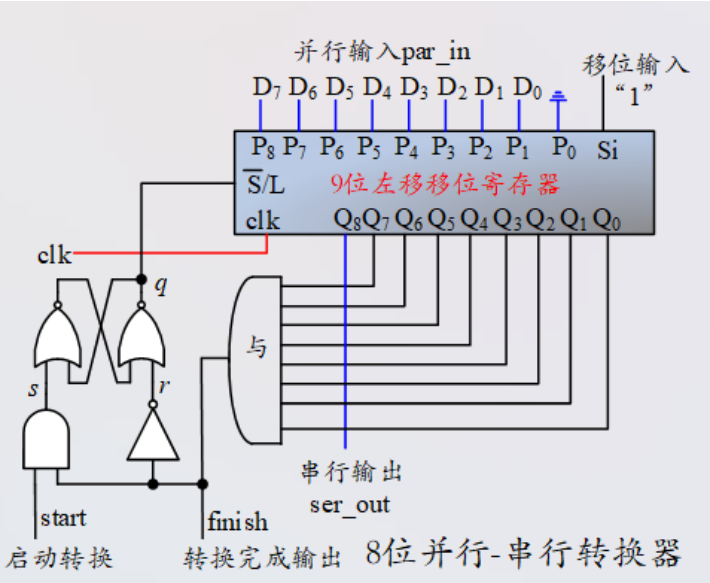
使用例子

并行串行转换器

如图是一个并行串行转换器。

start 启动信号拉高以后，加载并行输入 D0-D7，启动左移串行输出，等 D0 输出后自动停止移位操作

start 启动信号拉高以后，加载并行输入 D0-D7，启动左移串行输出，等 D0 输出后自动停止移位操作



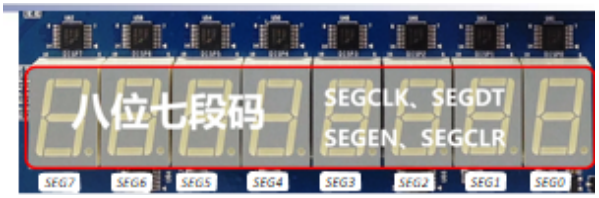
并行串行转换器

主板七段数码管接口说明

实验板上，8 个 74LS164A 的并行输出控制 8 个 7 段数码管的段码

- SEGCLK:8 位七段数码管模块的时钟，上升沿触发移位
- SEGCLR: 清零，使所有数码管显示 0，低电平有效
- SEGDT: 串行移位数据输入，0 使数码管段亮
- SEGEN：数码管模块总控开关，1 为使能

主板上 8 位数码管显示采用的是静态显示，不是动态扫描方式
实验板上用 8 个 74LV164A 构成 64 位串-并转换模块，并行输出控制 8 个 7 段数码管通过 SEGCLR 和 SEGDT 串行接收 8 个数码管 *8 段码，共计 64 位数据，移位先后顺序为 SEG7_DP，SEG7_g，SEG7_f，……，SEG0_b，SEG0_a
数码管共阳接法，段码为 0 时对应段亮



8 位七段数码管

三、实验步骤和结果记录

任务一：设计 8 位带并行输入的左移移位寄存器

建立工程文件

- 新建工程文件，命名为 shfit_reg8b
- 根据四位带并行输入的左移移位寄存器原理图添加源文件 shfit_reg8b.v, 代码如下：

```
module shfit_reg8b (
    input wire clk, S_L, s_in,
    input wire [7:0] p_in,
    output wire [7:0] Q
);
        FD m0(.C(clk),.D((!S_L)?s_in:p_in[0]),.Q(Q[0]));
        FD m1(.C(clk),.D((!S_L)?Q[0]:p_in[1]),.Q(Q[1]));
        FD m2(.C(clk),.D((!S_L)?Q[1]:p_in[2]),.Q(Q[2]));
        FD m3(.C(clk),.D((!S_L)?Q[2]:p_in[3]),.Q(Q[3]));
        FD m4(.C(clk),.D((!S_L)?Q[3]:p_in[4]),.Q(Q[4]));
        FD m5(.C(clk),.D((!S_L)?Q[4]:p_in[5]),.Q(Q[5]));
        FD m6(.C(clk),.D((!S_L)?Q[5]:p_in[6]),.Q(Q[6]));
        FD m7(.C(clk),.D((!S_L)?Q[6]:p_in[7]),.Q(Q[7]));
endmodule
```

工程文件仿真

添加仿真文件 shfit_tb.v, 代码如下：

```
    module shift_tb();
reg clk;
reg [7:0] p_in;
reg S_L;
reg s_in;
wire [7:0] Q;

shfit_reg8b uut
(
```

```
        .clk(clk),
        .S_L(S_L),
        .s_in(s_in),
        .p_in(p_in),
        .Q(Q)
    );

    initial begin
        clk = 0;
        forever begin
         #20; clk = ~clk;
        end
    end

    initial begin
        S_L=1;//选择并行输入
        p_in=0;//初始化为 0
        #100;
        S_L=0;//选择串行输入
        s_in=1;
        p_in=8'hFF;
        #400;
        S_L=1;//选择并行输入
    end

endmodule
```

运行仿真文件，查看仿真波形如下

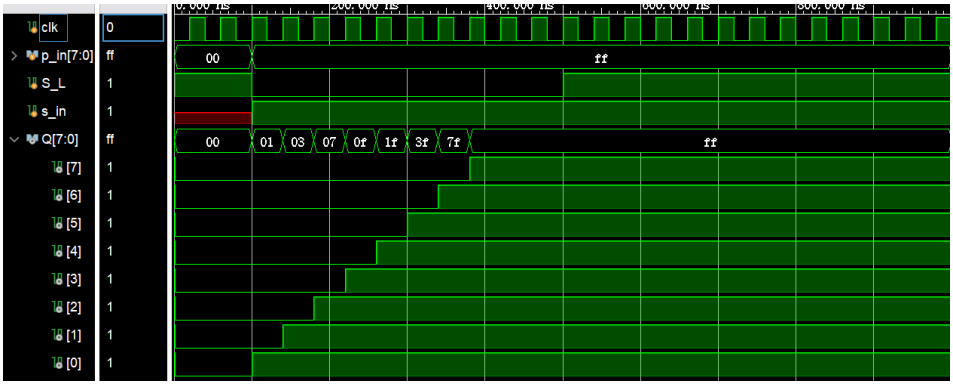


图 2: 仿真波形

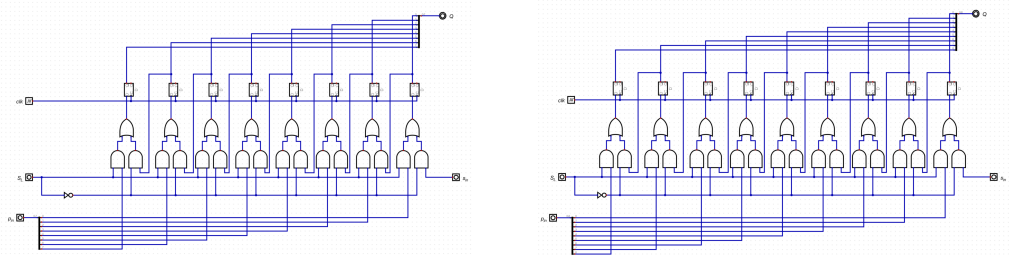
波形解释：由波形图可知，一开始 SL=1，选择并行输入，输入为 0，输出为 0；之后 SL=0，选择串行输入，输入为 s_in 等于 1，随着时钟信号的变化，s_in 不断

串行左移输入，可以看到输出 Q 的值也在不断变化。最后 SL=1，选择并行输入，输入为 ff，输出为 ff。

任务二：设计主板 16 位 LED 灯驱动模块

Digital 画图理解

在 Digital 中绘制 shift_reg8b、shift_reg9b 模块原理图：



(a) shfit_reg8b　　(b) shfit_reg9b

图 3: Digital 原理图

绘制 P2S8b 模块原理图，通过仿真理解并串转换的启停原理

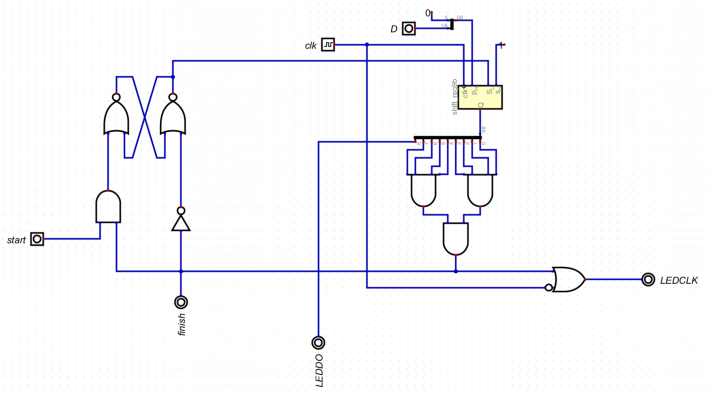


图 4: P2S8b 原理图

通过 Digital 中实时仿真功能，可以看到 LED_DO 不断左移输出寄存器中的值，直到输出完毕。

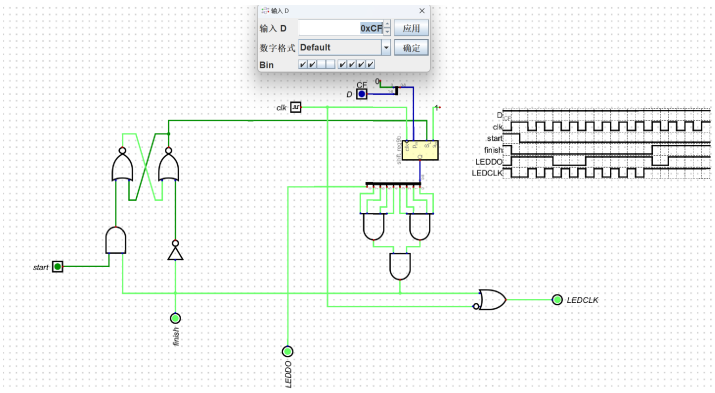


图 5: P2S8b 仿真波形

Vivado **实现**

要求： 简化实验 12 任务一的电路，设计 4 个可设自增的 4 位寄存器，汇总成总线 num[15:0]，显示在小实验板的 4 位七段数码管上

新建工程 LEDP2S，从以前的工程中导入需要用到的文件，新建源文件 LED_DRV.v，代码以及注释解释如下：

```
`timescale 1ns / 1ps
module LED_DRV(
    input wire clk,//时钟信号
    input wire [15:0] SW,
    input wire [3:0] BTN,//基本控制信号
    output wire BTNX4,//按钮使能

    output wire [7:0] SEGMENT,
    output wire [3:0] AN,//数码管动态扫描显示信号

    output LED_CLK,//LED 时钟信号
    output LED_CLR,
    output LED_DO,
    output LED_EN

    );
    assign BTNX4 = 1'b0;//按钮使能信号

    wire [3:0] Load_A,Load_B,Load_C,Load_D;
    wire [3:0] A, B, C, D, A_IN, B_IN, C_IN, D_IN,A1,B1,C1,D1;
    wire [31:0] clk_div;
    wire [15:0] num;

    assign num = {A, B, C, D};
```

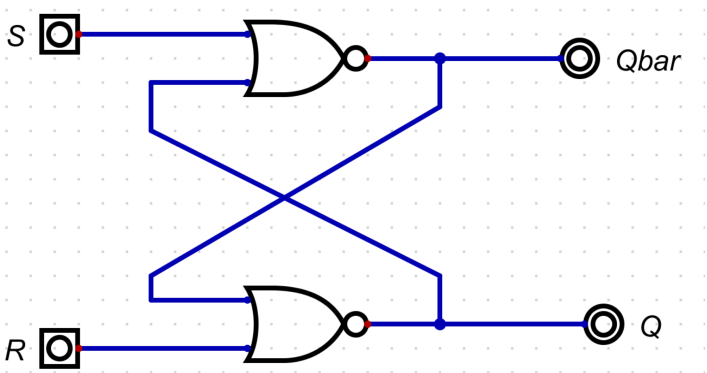
```
25
26      clkdiv clock1(clk, 1'b0, clk_div);
27
28      MyRegister4b RegA(.clk(clk), .IN(A_IN), .Load(Load_A), .OUT(A));
29      MyRegister4b RegB(.clk(clk), .IN(B_IN), .Load(Load_B), .OUT(B));
30      MyRegister4b RegC(.clk(clk), .IN(C_IN), .Load(Load_C), .OUT(C));
31      MyRegister4b RegD(.clk(clk), .IN(D_IN), .Load(Load_D),
        ↪  .OUT(D));//寄存器模块
32
33      assign A_IN = (SW[7] == 1'b0)? A1: 4'b0000;
34      assign B_IN = (SW[6] == 1'b0)? B1: 4'b0000;
35      assign C_IN = (SW[5] == 1'b0)? C1: 4'b0000;
36      assign D_IN = (SW[4] == 1'b0)? D1: 4'b0000;//2选1多路复用器,
        ↪  复位寄存器初值
37
38      AddSub4b m4(.A(A), .B(4'b0001), .Ctrl(SW[3]), .S(A1));
39      AddSub4b m5(.A(B), .B(4'b0001), .Ctrl(SW[2]), .S(B1));
40      AddSub4b m6(.A(C), .B(4'b0001), .Ctrl(SW[1]), .S(C1));
41      AddSub4b m7(.A(D), .B(4'b0001), .Ctrl(SW[0]), .S(D1));//自增/自减逻辑
42
43      Load_Gen m0(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(BTN[3]),
        ↪  .Load_out(Load_A));
44      Load_Gen m1(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(BTN[2]),
        ↪  .Load_out(Load_B));
45      Load_Gen m2(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(BTN[1]),
        ↪  .Load_out(Load_C));
46      Load_Gen m3(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(BTN[0]),
        ↪  .Load_out(Load_D));//寄存器Load信号
47
48      DispNum m8(.scan(clk_div[18:17]), .HEXS(num), .LES(4'b0),
        ↪  .point(4'b0), .AN(AN), .SEGMENT(SEGMENT));//数码管显示模块
49
50      wire [16:0] tmp;
51      wire finish,start,S_L;
52      assign start = SW[15];//开始移位信号
53      assign LED_CLK = ~clk | finish;//移位结束后将时钟停止，避免继续移位
54      assign LED_CLR = SW[14];//清零信号，使所有 LED 灯亮，低电平有效
55      assign LED_DO = ~tmp[16];//LED 灯输出信号，取反左移输出
56      assign LED_EN = SW[13];//使能信号
57
58      assign finish = tmp[15]&tmp[14]&tmp[13]&tmp[12]&tmp[11]&tmp[10]&tmp[
        ↪  9]&tmp[8]&tmp[7]&tmp[6]&tmp[5]&tmp[4]&tmp[3]&tmp[2]&tmp[1]&tmp[0
        ↪  ];//移位结束信号
59
60      SR_LATCH m9(.S(start & finish ),.R(~finish),.Q(S_L));//移位信号锁存器
```

```
61
62      //构造 17 位右移移位寄存器
63      FD Dfiflop(.C(clk),.D((!S_L)?1'b1:1'b0),.Q(tmp[0]));//锁存器模块
64      shfit_reg8b m10(.clk(clk),.S_L(S_L),.s_in(tmp[0]),.p_in(num[7:0]),.Q
        ↪  (tmp[8:1]));
65      shfit_reg8b m11(.clk(clk),.S_L(S_L),.s_in(tmp[8]),.p_in(num[15:8]),.
        ↪  Q(tmp[16:9]));//移位寄存器模块
66  endmodule
```

代码解释：我们用两个 8 位移位寄存器和一个 D 触发器的串联构成了 17 位左移移位寄存器，仿照 Digital 中 P2S8b 的原理图完成了并行数据转换为串行数据的功能。

我们使用 SW[15] 来控制 start 信号，SW[14] 来控制 clear 信号，SW[13] 来控制 enable 信号，BTN[3:0] 来依次控制四个寄存器的 load 的信号，并且对应关系与四段数码管的显示一致。需要注意的是,16 个 LED 是低电平使能的，所以 LED_DO 要取反输出。SR_LATCH 模块是我们自己设计的 SR 锁存器，其电路图和 Verilog 代码如下。

```
1   module SR_LATCH (
2       input S,
3       input R,
4       output Qbar,
5       output Q
6     );
7       wire Q_temp;
8       wire Qbar_temp;
9       assign Qbar_temp = ~ (S | Q_temp);
10      assign Q_temp = ~ (Qbar_temp | R);
11      assign Qbar = Qbar_temp;
12      assign Q = Q_temp;
13    endmodule
```



仿真验证要求

- 参考实验 12 修改代码，减少去抖、扫描显示等模块中的延时等待时间
- 将 Top 模块中的 num 总线显示出来，以 16 进制显示
- 操作 BTNX4Y0 到 BTNX4Y4，将 4 个寄存器初值设为 4321h，拨动 SW[15] 启动移位，观察 LED_CLK 和 LED_DO 的输出

```
        module LEDP2S(

    );
    reg clk;
    reg [15:0] SW;
    reg [3:0] BTN;
    wire BTNX4;
    wire [7:0] SEGMENT;
    wire [3:0] AN;
    wire LED_CLK;
    wire LED_CLR;
    wire LED_DO;
    wire LED_EN;

    LED_DRV m0(
        .clk(clk),
        .SW(SW),
        .BTN(BTN),
        .BTNX4(BTNX4),
        .SEGMENT(SEGMENT),
        .AN(AN),
        .LED_CLK(LED_CLK),
        .LED_CLR(LED_CLR),
        .LED_DO(LED_DO),
        .LED_EN(LED_EN)
        );
    initial begin
        SW=16'b0;//开关初始化
        BTN=4'b0;//按钮初始化
        #20;
        SW[7:4]=4'b1111;
        #20;
        BTN=4'b1111;
        #20;
        BTN=4'b0000;//四个寄存器初始化为 0
        #20;
```

```
        SW[7:4]=4'b0000;
        #20;
        BTN=4'b1111;#20;BTN=4'b0000;//四个寄存器加到一
        #20;
        BTN=4'b1110;#20;BTN=4'b0000;//前三个寄存器加到二
        #20;
        BTN=4'b1100;#20;BTN=4'b0000;//前两个寄存器加到三
        #20;
        BTN=4'b1000;#20;BTN=4'b0000;//第一个寄存器加到四
        #20;
        #200;
        SW[15]=1;//开始移位
        #50;
        SW[15]=0;//停止移位
    end
    initial begin
        clk=0;
        forever #10 clk=~clk;
    end
endmodule
```

运行仿真文件，查看仿真波形如下

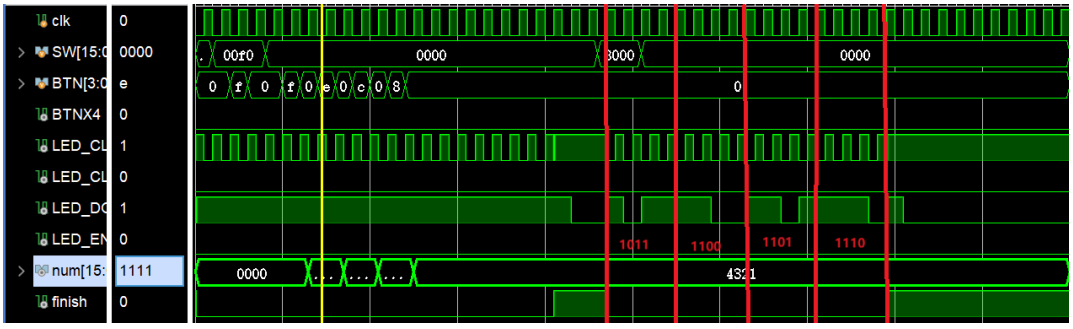


图 6: 仿真波形

仿真波形解释：由波形图可知，我们将四个寄存器初始化为 4321h 后，需要等待一段时间，等到寄存器里面旧的值全部输出，寄存器被清空之后，再用 SW[15], 提供一个脉冲，信号，使得 4321h 可以作为并行信号输入到寄存器当中，然后随着时钟移位输出，如在波形图显示的从左到右依次为,(1011),(1100),(1101),(1110), 分别对应 4，3，2，1 的二进制反码，符合我们的预期,LED_CLK 信号也在 finish 等于 1 的情况下停止了时钟。故仿真波形正确

下载验证

验证要求

- 用 BTNX4Y0 到 BTNX4Y4 作为自增按键，设置 4 位七段数码管的初值
- 用 SW[15] 控制将 4 位七段数码管的数据输出到 LED 灯
- LED 灯显示应清晰稳定，没有残影
- LED 灯的高低位顺序要与数码管显示顺序匹配，0 为暗，1 为亮

结果如下

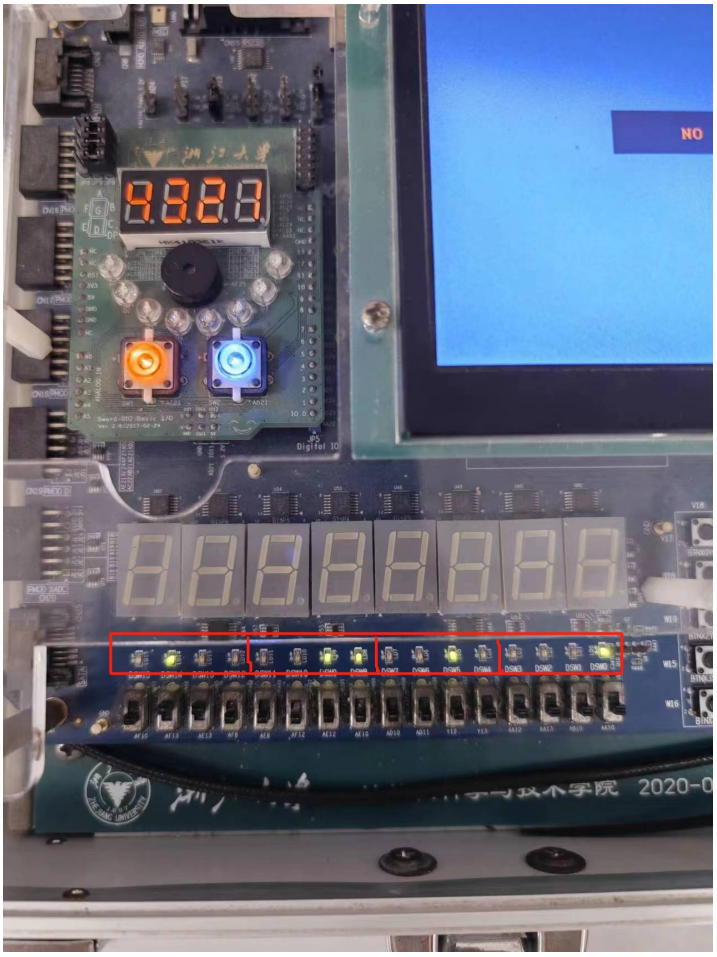


图 7: 实验结果

如图从左到右的亮暗顺序依次表示 4321

任务三：设计主板 8 位数码管驱动模块

建立工程

- 利用实验 12 任务一的电路，设计 8 个可自增的 4 位寄存器，接入总线 num[31:0]
- 调用 8 个 MyMC14495 模块进行段码译码
- 利用 8 个 shfit_reg8b 模块和 1 个触发器，设计主板 8 位数码管驱动模块 SEG_-DRV

主要代码如下

```
`timescale 1ns / 1ps

module SEG_DRV(
     input wire clk,//时钟信号
     input wire [15:0] SW,

     output wire SEGCLK,//数码管时钟信号
     output wire SEGCLR,
     output wire SEGDT,
     output wire SEGEN
    );

    wire [31:0] clk_div;
    wire [3:0] Load_A,Load_B,Load_C,Load_D,Load_E,Load_F,Load_G,Load_H;
    wire [3:0] A, B, C, D, E, F, G, H, A_IN, B_IN, C_IN, D_IN, E_IN,
     ↪  F_IN, G_IN, H_IN, A1, B1, C1, D1, E1, F1, G1, H1;
    wire [31:0] reg_num;
    wire [63:0] disp_num;// 并行数据 Dispnum

    clkdiv clock1(clk, 1'b0, clk_div);

    MyRegister4b RegA(.clk(clk), .IN(A_IN), .Load(Load_A), .OUT(A));
    MyRegister4b RegB(.clk(clk), .IN(B_IN), .Load(Load_B), .OUT(B));
    MyRegister4b RegC(.clk(clk), .IN(C_IN), .Load(Load_C), .OUT(C));
    MyRegister4b RegD(.clk(clk), .IN(D_IN), .Load(Load_D), .OUT(D));
    MyRegister4b RegE(.clk(clk), .IN(E_IN), .Load(Load_E), .OUT(E));
    MyRegister4b RegF(.clk(clk), .IN(F_IN), .Load(Load_F), .OUT(F));
    MyRegister4b RegG(.clk(clk), .IN(G_IN), .Load(Load_G), .OUT(G));
    MyRegister4b RegH(.clk(clk), .IN(H_IN), .Load(Load_H),
     ↪  .OUT(H));//寄存器模块

    assign A_IN = (SW[13] == 1'b0)? A1: 4'b0000;
```

```
31      assign B_IN = (SW[13] == 1'b0)? B1: 4'b0000;
32      assign C_IN = (SW[13] == 1'b0)? C1: 4'b0000;
33      assign D_IN = (SW[13] == 1'b0)? D1: 4'b0000;
34      assign E_IN = (SW[13] == 1'b0)? E1: 4'b0000;
35      assign F_IN = (SW[13] == 1'b0)? F1: 4'b0000;
36      assign G_IN = (SW[13] == 1'b0)? G1: 4'b0000;
37      assign H_IN = (SW[13] == 1'b0)? H1: 4'b0000;//2选1多路复用器,
        ↪  复位寄存器初值
38
39      AddSub4b Aadd(.A(A), .B(4'b0001), .Ctrl(1'b0), .S(A1));
40      AddSub4b Badd(.A(B), .B(4'b0001), .Ctrl(1'b0), .S(B1));
41      AddSub4b Cadd(.A(C), .B(4'b0001), .Ctrl(1'b0), .S(C1));
42      AddSub4b Dadd(.A(D), .B(4'b0001), .Ctrl(1'b0), .S(D1));
43      AddSub4b Eadd(.A(E), .B(4'b0001), .Ctrl(1'b0), .S(E1));
44      AddSub4b Fadd(.A(F), .B(4'b0001), .Ctrl(1'b0), .S(F1));
45      AddSub4b Gadd(.A(G), .B(4'b0001), .Ctrl(1'b0), .S(G1));
46      AddSub4b Hadd(.A(H), .B(4'b0001), .Ctrl(1'b0), .S(H1));//自增逻辑
47
48      Load_Gen m0(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(SW[7]),
        ↪  .Load_out(Load_A));
49      Load_Gen m1(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(SW[6]),
        ↪  .Load_out(Load_B));
50      Load_Gen m2(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(SW[5]),
        ↪  .Load_out(Load_C));
51      Load_Gen m3(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(SW[4]),
        ↪  .Load_out(Load_D));
52      Load_Gen m4(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(SW[3]),
        ↪  .Load_out(Load_E));
53      Load_Gen m5(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(SW[2]),
        ↪  .Load_out(Load_F));
54      Load_Gen m6(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(SW[1]),
        ↪  .Load_out(Load_G));
55      Load_Gen m7(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(SW[0]),
        ↪  .Load_out(Load_H));//寄存器Load信号
56
57      assign reg_num = {A, B, C, D, E, F, G, H};
58
59      //将寄存器数据段码译码传递给数码管显示模块
60      Disp_Decorder m8(.D(reg_num[3:0]), .point(1'b0), .LE(1'b0),
        ↪  .SEGMENT(disp_num[7:0]));
61      Disp_Decorder m9(.D(reg_num[7:4]), .point(1'b0), .LE(1'b0),
        ↪  .SEGMENT(disp_num[15:8]));
62      Disp_Decorder m10(.D(reg_num[11:8]), .point(1'b0), .LE(1'b0),
        ↪  .SEGMENT(disp_num[23:16]));
63      Disp_Decorder m11(.D(reg_num[15:12]), .point(1'b0), .LE(1'b0),
        ↪  .SEGMENT(disp_num[31:24]));
```

```
64      Disp_Decorder m12(.D(reg_num[19:16]), .point(1'b0), .LE(1'b0),
        ↪  .SEGMENT(disp_num[39:32]));
65      Disp_Decorder m13(.D(reg_num[23:20]), .point(1'b0), .LE(1'b0),
        ↪  .SEGMENT(disp_num[47:40]));
66      Disp_Decorder m14(.D(reg_num[27:24]), .point(1'b0), .LE(1'b0),
        ↪  .SEGMENT(disp_num[55:48]));
67      Disp_Decorder m15(.D(reg_num[31:28]), .point(1'b0), .LE(1'b0),
        ↪  .SEGMENT(disp_num[63:56]));//数码管显示模块
68
69
70      wire [64:0] tmp;
71      wire finish,start,S_L;
72      assign start = SW[15];//开始移位信号
73      assign SEGCLK = ~clk | finish;//移位结束后将时钟停止，避免继续移位
74      assign SEGCLR = SW[14];//清零信号，使所有 LED 灯亮，低电平有效
75      assign SEGDT = tmp[64];//LED 灯输出信号，取反左移输出
76      assign SEGEN = 1'b1;//使能信号
77
78      AND_64bit m16(.in(tmp[63:0]),.out(finish));//移位结束信号
79
80      SR_LATCH m17(.S(start & finish ),.R(~finish),.Q(S_L));//移位信号锁存器
81
82      //构造 65 位左移移位寄存器
83      FD Dfiflop(.C(clk),.D((!S_L)?1'b1:1'b0),.Q(tmp[0]));//锁存器模块
84      shfit_reg8b m18(.clk(clk),.S_L(S_L),.s_in(tmp[0]),.p_in(disp_num[7:0
        ↪  ]),.Q(tmp[8:1]));
85      shfit_reg8b m19(.clk(clk),.S_L(S_L),.s_in(tmp[8]),.p_in(disp_num[15:
        ↪  8]),.Q(tmp[16:9]));
86      shfit_reg8b m20(.clk(clk),.S_L(S_L),.s_in(tmp[16]),.p_in(disp_num[23
        ↪  :16]),.Q(tmp[24:17]));
87      shfit_reg8b m21(.clk(clk),.S_L(S_L),.s_in(tmp[24]),.p_in(disp_num[31
        ↪  :24]),.Q(tmp[32:25]));
88      shfit_reg8b m22(.clk(clk),.S_L(S_L),.s_in(tmp[32]),.p_in(disp_num[39
        ↪  :32]),.Q(tmp[40:33]));
89      shfit_reg8b m23(.clk(clk),.S_L(S_L),.s_in(tmp[40]),.p_in(disp_num[47
        ↪  :40]),.Q(tmp[48:41]));
90      shfit_reg8b m24(.clk(clk),.S_L(S_L),.s_in(tmp[48]),.p_in(disp_num[55
        ↪  :48]),.Q(tmp[56:49]));
91      shfit_reg8b m25(.clk(clk),.S_L(S_L),.s_in(tmp[56]),.p_in(disp_num[63
        ↪  :56]),.Q(tmp[64:57]));//移位寄存器模块
92      //
93
94  endmodule
95
```

代码解释：

- 首先，我们设置了 8 个可自增的寄存器 A-H，然后把他们的值汇聚成总线 *reg_num*,
- 接下来，我们需要把 *reg_num*, 通过译码模块 *Disp_Decorder* 转换成实验板上 8 个七段数码管共 64 位的并行数据 *disp_num*
- 然后，与任务二类似，我们可以利用八个八位左移移位寄存器，把 64 位并行数据，转换为，数码管的串行输出，数码管的低电平有效已经在译码器中被考虑到，因此输出 SEGDT 的时候不需要再用到非门
- 代码中用到了自己设计的模块 AND_64bit, 作用是把 64 位输入做逻辑且输出
- 代码中用到了自己设计的模块 Disp_Decorder, 主要把 MyMC14495 的输入输出汇成总线，方便操作，代码如下

```
module Disp_Decorder(
    input wire [3:0] D,
    input wire point,
    input wire LE,
    output wire [7:0] SEGMENT
    );
    MyMC14495 m0(
        .D3(D[3]),
        .D2(D[2]),
        .D1(D[1]),
        .D0(D[0]),
        .point(point),
        .LE(LE),
        .g(SEGMENT[6]),
        .f(SEGMENT[5]),
        .e(SEGMENT[4]),
        .d(SEGMENT[3]),
        .c(SEGMENT[2]),
        .b(SEGMENT[1]),
        .a(SEGMENT[0]),
        .p(SEGMENT[7])
        );

endmodule
```

仿真验证

仿真验证要求：　将 Top 模块中的 num 总线显示出来，以 16 进制显示操作 SW[7:0]，将 8 个寄存器初值设为学号后 8 位，拨动 SW[15] 启动移位，观察 SEGCLK 和 SEGDT 的输出

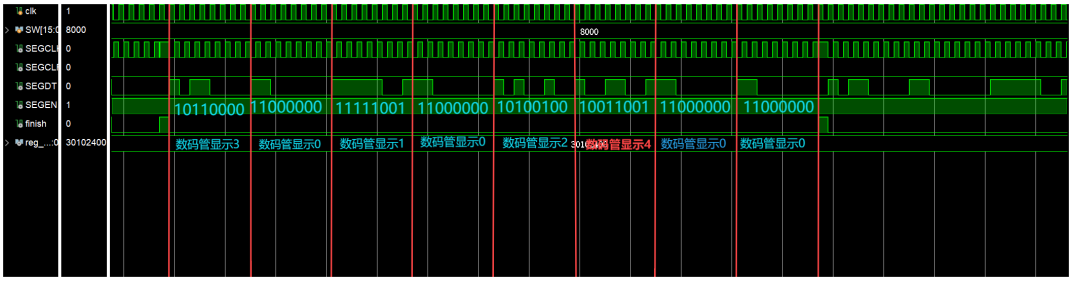
```
        module SEG_DRV_tb(

    );
    reg clk;
    reg [15:0] SW;
    wire SEGCLK;
    wire SEGCLR;
    wire SEGDT;
    wire SEGEN;

    SEG_DRV m0(
        .clk(clk),
        .SW(SW),
        .SEGCLK(SEGCLK),
        .SEGCLR(SEGCLR),
        .SEGDT(SEGDT),
        .SEGEN(SEGEN)
        );
    initial begin
        clk=0;
        forever begin
        #5; clk=~clk;
        end
    end
    initial begin
        SW=16'b0;//开关初始化
        #20;//学号后八位为 30102400
        SW[7:0]=8'b10101100;
        #20;
        SW[7:0]=8'b00000000;//此时为 10101100
        #20;
        SW[7:0]=8'b10001100;
        #20;
        SW[7:0]=8'b00000000;//此时为 20102200
        #20;
        SW[7:0]=8'b10000100;
```

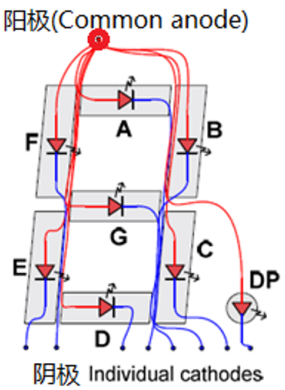
```
        #20;
        SW[7:0]=8'b00000000;//此时为 30102300
        #20;
        SW[7:0]=8'b00000100;
        #20;
        SW[7:0]=8'b00000000;//此时为 30102400//初始化为学号后八位
        #20;
        #500;//等待寄存器被排空
        SW[15]=1;//使能将 30102400 输入到串并转换器
    end
endmodule
```

仿真代码如下 仿真代码基本作用已经注释好，具体仿真结果图如下

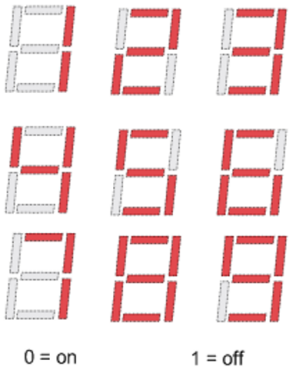


仿真结果

仿真波形解释：如图，在设置八个寄存器的值为学号后八位后，我们等待寄存器被排空，然后给予 start 脉冲，使得 30102400 可以作为并行信号输入到寄存器当中，然后随着时钟移位输出，具体显示结果已在波形图中画出，从高位到低位依次对应着，七段数码管的小数点位 DP，G,F,E,D,C,B,A.



X	a	b	c	d	e	f	g
0	0	0	0	0	0	0	1
1	1	0	0	1	1	1	1
2	0	0	1	0	0	1	0
3	0	0	0	0	1	1	0
4	1	0	0	1	1	0	0
5	0	1	0	0	1	0	0
6	0	1	0	0	0	0	0
7	0	0	0	1	1	1	1
8	0	0	0	0	0	0	0
9	0	0	0	0	1	0	0
A	0	0	0	1	0	0	0
B	1	1	0	0	0	0	0
C	0	1	1	0	0	0	1
D	1	0	0	0	0	1	0
E	0	1	1	0	0	0	0
F	0	1	1	1	0	0	0



数码管对应关系图

下载验证

要求 ：可以拨动 8 个开关 SW[7:0] 来修改每个数码管的数值
将主板七段数码管设成显示学号后 8 位
数码管显示应清晰稳定，没有残影

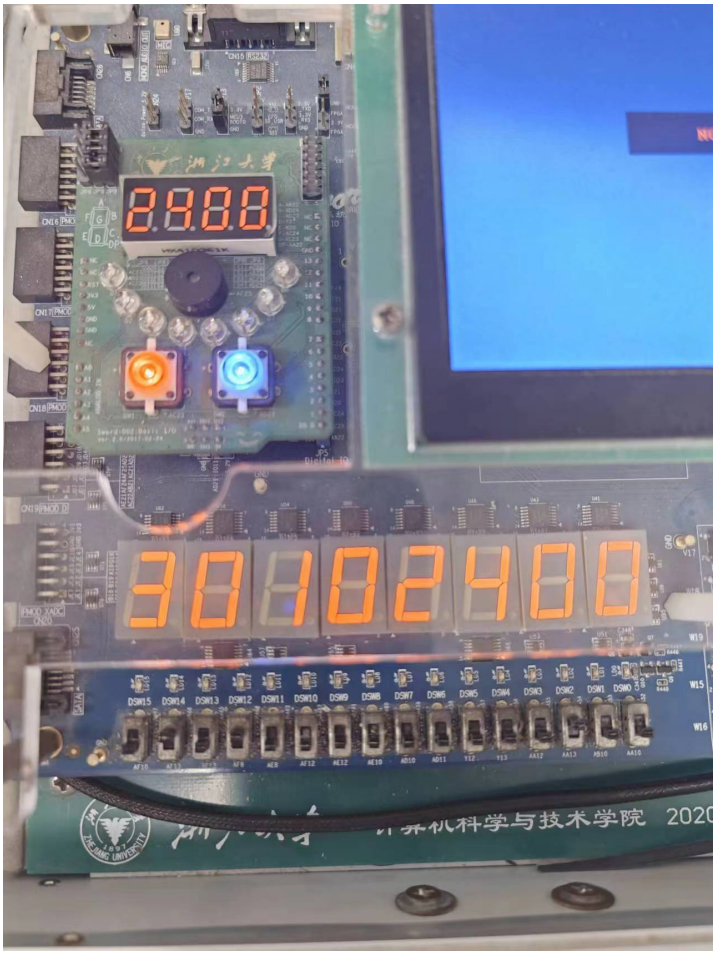


图 8: 实验结果

四、实验结果分析

相关实验结果以及波形分析已在实验部分进行了详细的介绍，这里不再赘述。

五、讨论与心得

这次实验可以说是这么多次以来做的最难的一次实验了，需要自主完成部分很多，在预习的过程中也是被硬控两个晚上，后面自己好好从头画了一遍移位寄存器，以及对于自己设计的每一个模块都进行了仿真测试，感觉龙场悟道！框框一顿写把任务都做完了！感觉十分锻炼自己的 debug 能力，也让自己对于 verilog 的代码有了更深的理解。

然后提前去实验室把上板的内容试了一下，全都正确无误，成就感满满！验收的时候也帮附近的同学解答了一些他们的错误，自己的能力有了一定的提升。最后验收也是十分顺利。