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：刘晨

学　　院：计算机科学与技术学院

专　　业：图灵班

学　　号：3190104666

指导教师：董亚波

2020 年 12 月 24 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： 寄存器和寄存器传输设计

学生姓名： 刘晨 专业： 图灵 1901 学号： 3190104666

同组学生姓名： 指导老师： 董亚波

实验地点： 东 4-509 实验日期：2020 年 12 月 24 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

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

原理：

1. 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。

在这里必须采用主从触发器或边沿触发器，不能采用锁存器。

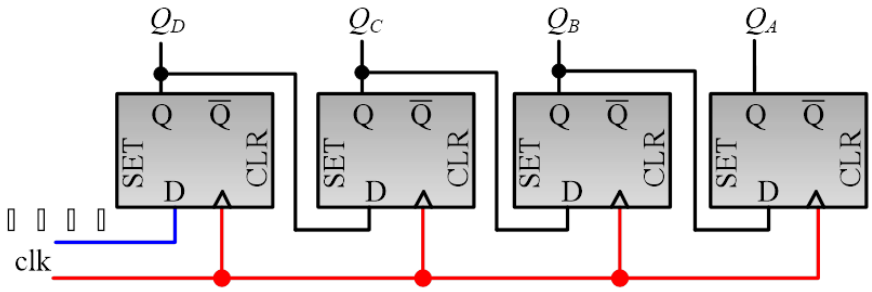
数据移动方式：左移、右移、循环移位

数据输入输出方式：

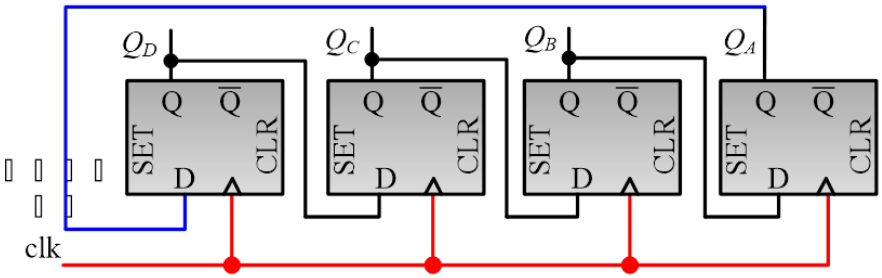
串行输入，串行输出；串行输入，并行输出；并行输入，串行输出

2. 串行输入右移移位寄存器 原理图如下：

使用 D 触发器构成串行输入的右移移位寄存器：



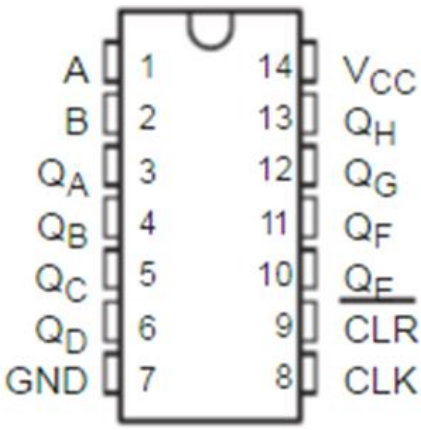
3. 循环右移移位寄存器 原理图如下：



4. 实验板上使用的芯片：74LV164A

芯片是 8 位串行右移移位寄存器，可以实现串-并转换

芯片示意图如下：

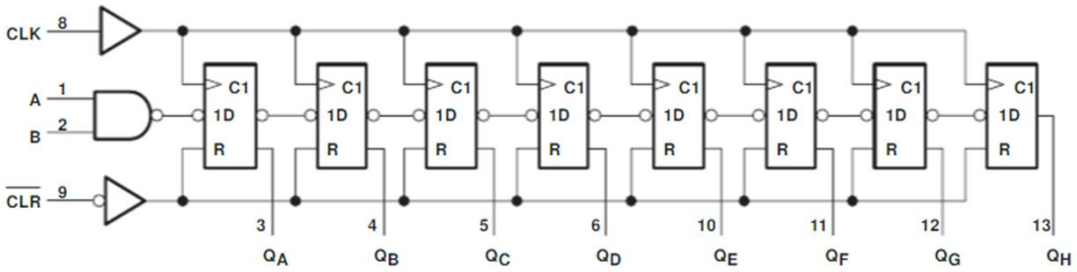


FUNCTION TABLE

INPUTS				OUTPUTS		
$\overline{CLR}$	CLK	A	B	Q_A	$Q_B \ldots Q_H$	
L	X	X	X	L	L	L
H	L	X	X	Q_{A0}	Q_{B0}	Q_{H0}
H	↑	H	H	H	Q_{An}	Q_{Gn}
H	↑	L	X	L	Q_{An}	Q_{Gn}
H	↑	X	L	L	Q_{An}	Q_{Gn}

Q_{A0}, Q_{B0}, Q_{H0} = the level of Q_A, Q_B, or Q_H, respectively, before the indicated steady-state input conditions were established.

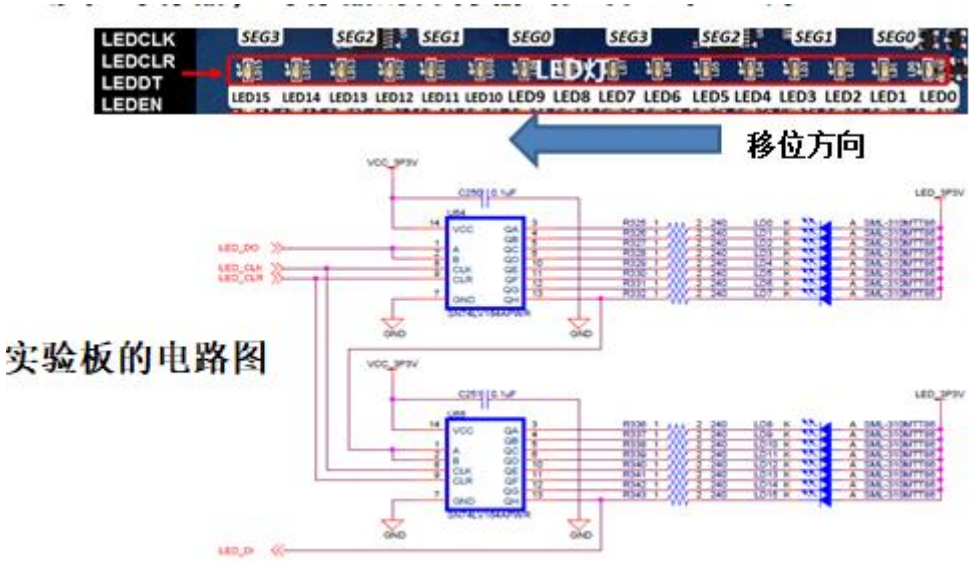
Q_{An}, Q_{Gn} = the level of Q_A or Q_G before the most recent ↑ transition of the clock; indicates a 1-bit shift.



5. 接口说明：实验板 16 位 LED 灯

实验板上，采用 2 个 74LV164A 构成 16 位串行输入并行输出移位寄存器，寄存

器的并行输出控制 16 个 LED 灯。实验板电路如下：



6. 引脚约束：实验板 16 位 LED 灯

```
#主板 LED 输出引脚约束
NET "LED_CLK"        LOC = N26    | IOSTANDARD = LVCMOS33 ;
NET "LED_CLR"        LOC = N24    | IOSTANDARD = LVCMOS33 ;
NET "LED_DO"         LOC = M26    | IOSTANDARD = LVCMOS33 ;
NET "LED_EN"         LOC = P18     | IOSTANDARD = LVCMOS33 ;
```

说明：

LED_CLK：16 位 LED 灯的时钟

LED_CLR：清零，使所有 LED 亮

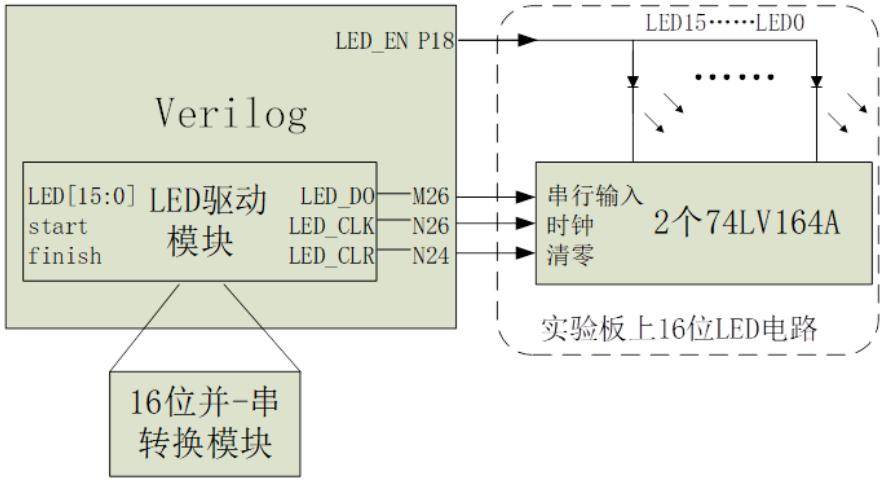
LED_DO：16 位 LED 数据串行输入，输入 0 使 LED 亮

LED_EN：控制 LED 电源，1 为使能 LED 模块

16 位串行输入顺序是 LED15，LED14，……，LED1，LED0

7. 设计对于实验板 LED 灯的模块

采用 Verilog 语言设计 LED 驱动模块，模块里实例化 16 位并-串转换模块，将需要显示的 16 位二进制数串行送给实验板上的 16 位 LED 灯模块，最终实现对 LED 的控制。

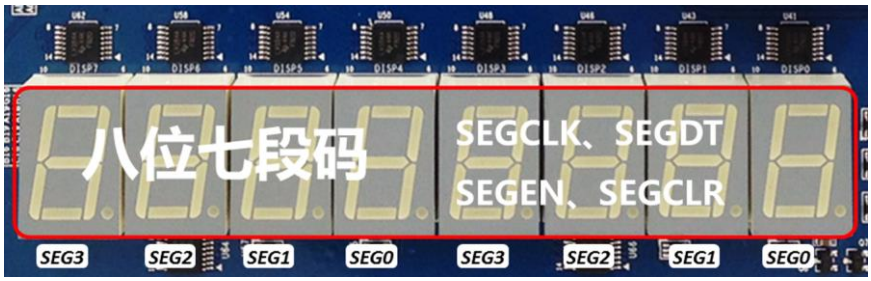


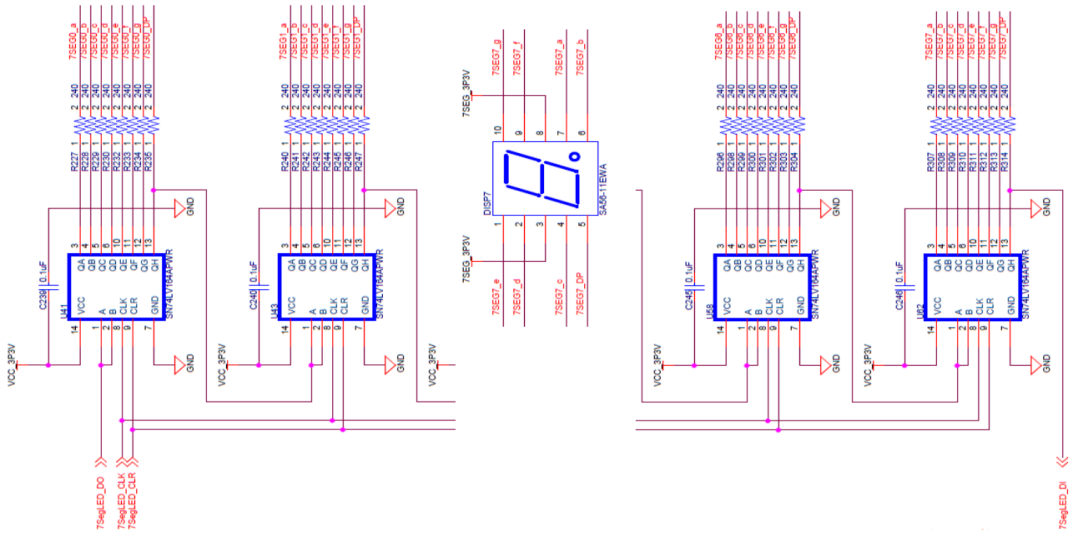
8. 主板七段数码管的接口说明

实验板上，8 个 74LS164A 的并行输出控制 8 个 7 段数码管的段码进行数据串行输入，通过并串转换电路输出：

P_Data[63:0]=SEGMENT[63:0]

实验板上的示意图：





其中使用的引脚约束：

```
#七段码移位输出引脚约束
NET "SEGCLK"     LOC = M24    | IOSTANDARD = LVCMOS33 ;
```

```
NET "SEGCLR"     LOC = M20   | IOSTANDARD = LVCMOS33 ;
NET "SEGDT"      LOC = L24    | IOSTANDARD = LVCMOS33 ;
NET "SEGEN"      LOC = R18    | IOSTANDARD = LVCMOS33 ;
```

说明：

SEGCLK：64 位串-并转换模块的时钟

SEGCLR：清零，所有段亮

SEGDT：数据串行输入，输入 0 亮

SEGEN：控制数码管电源，1 为使能

64 位串行输入顺序是 SEG7_DP，SEG7_g，SEG7_f，……，SEG0_b，SEG0_a

具体设计方式如下：

8 位数码管显示采用静态显示，替代动态扫描方式

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。

每个 7 段数码管的段码输入 a-g、dp 分别直接接在串-并转换模块的不同引脚上，64 位串-并转换模块接收 FPGA 通过数据串行输入引脚发送的 64 位段码数据，并行输出以后直接控制 8 个数码管的段码引脚。

FPGA 的 M24、M20、L24、R18 这 4 个引脚分别控制串-并转换模块的时钟、清零、数据串行输入和使能。

采用 Verilog 语言设计七段数码管驱动模块，模块里设计 64 位并-串转换模块，将需要显示的 8 位数码管的 64 位段码串行送给 64 位串-并转换模块，实现对数码管的控制。

三、实验过程和数据记录

任务 1：设计 8 位带并行输入的右移移位寄存器

新建工程，工程名称用 ShfitReg8b。

Top Level Source Type 用 HDL。

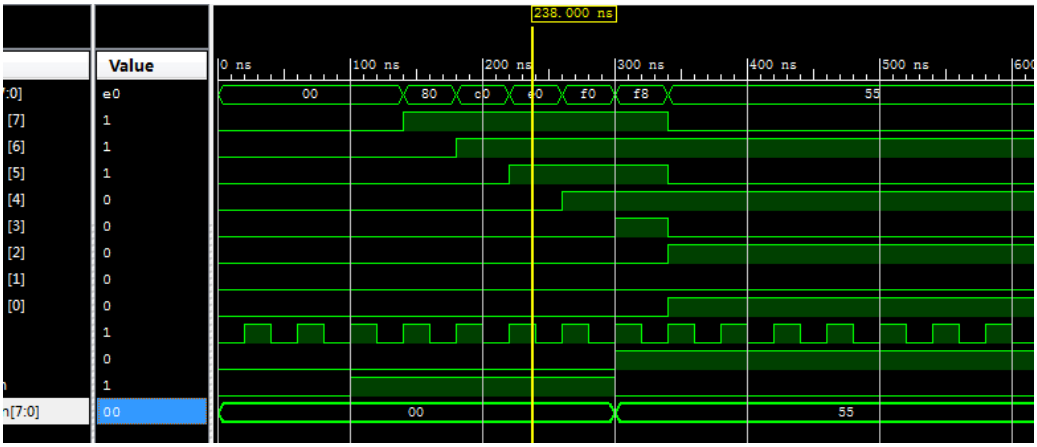
主要描述代码如下：

```
module shiftreg8b(
    input wire clk, S_L, shift_in,
    input wire [7:0] p_in,
    output wire [7:0] Q
    );

    wire a,b,c,d,e,f,g,h;
    wire a0,a1,b0,b1,c0,c1,d0,d1,e0,e1,f0,f1,g0,g1,h0,h1;
    wire S_bar;

FD m0(
    .Q(Q[0]),
    .C(clk),
    .D(a)
);

FD m1(
    .Q(Q[1]),
    .C(clk),
    .D(b)
);
```

仿真代码如下：

```
initial begin
        // Initialize Inputs
        clk = 0;
        L = 0;
        s_in = 0;
        p_in = 0;

        #100;

        // Add stimulus here
        S_L = 0;
        s_in = 1;
        p_in =0;

        #200;
        S_L = 1;
        s_in = 0;
        p_in = 8'b0101_0101;
        #500;
    end

    always begin
        clk = 0; #20;
        clk = 1; #20;
    end
```

仿真结果如下：



任务 2：设计主板 LED 灯驱动模块

新建工程，工程名称用 LEDP2S。

Top Level Source Type 用 HDL。

用结构化描述设计，调用 CreatNumber 模块，用小实验板的 4 位七段数码管设置 16 位 LED 灯的初值

利用 ShfitReg8b 模块，设计 LED 灯驱动模块 LED_DRV

主要模块代码如下：

Top 模块：

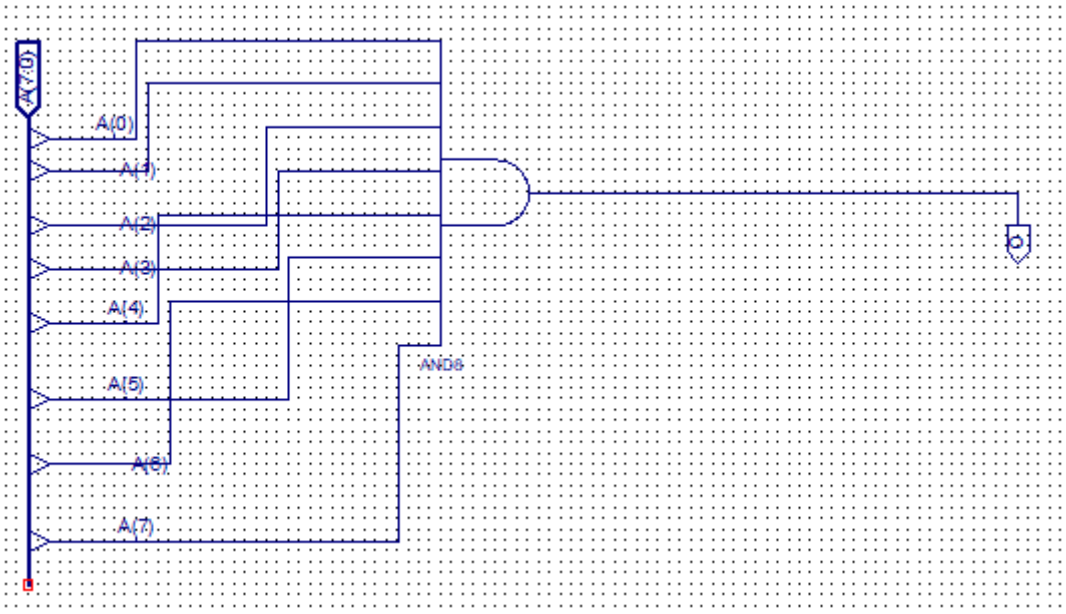
```
module top(
    input wire clk,
    input wire [3:0]BTN,
    input wire [15:0]SW,
    output wire [3:0]AN,
    output wire [7:0]SEGMENT,
    output wire BTNX4,
    output wire LED_EN,
    output wire LED_DO,
    output wire LED_CLK,
    output wire LED_CLR
    );
    wire [15:0] num;
    wire [3:0] btn_out;
    wire [31:0] clk_div;
    pbdebounce p0(clk_div[17], BTN[0], btn_out[0]);
    pbdebounce p1(clk_div[17], BTN[1], btn_out[1]);
    pbdebounce p2(clk_div[17], BTN[2], btn_out[2]);
    pbdebounce p3(clk_div[17], BTN[3], btn_out[3]);
```

```
    clkdiv c0(.clk(clk),.rst(1'b0),.clkdiv(clk_div));
    CreateNumber m3(.btn(btn_out),.num(num));
    DispNum m6(.clk(clk), .HEXS(num), .LES(4'b0),.points(4'b0),.RST(1'b0),
.AN(AN), .segment(SEGMENT));
    LED_DRV(.LED(num),.start(SW[15]),.clk(clk),.LED_CLK(LED_CLK),.ser_out(L
ED_DO));
    assign BTNX4 = 1'b0;
    assign LED_EN = 1'b1;
    assign LED_CLR = SW[14];
 endmodule
```

LED_DRV 模块：

```
module LED_DRV(
    input wire [15:0]LED,
    input wire start,
    input wire clk,
    output wire LED_CLK,
    output wire ser_out
    );
    wire [16:0]Q;
    wire q,r,s,q1,d0,d1,d2;
    wire finish,f0,f1,finish1;
    AND2 n1(.I0(1'b1),.I1(q1),.O(d0));
    AND2 n2(.I0(1'b0),.I1(q),.O(d1));
    OR2 n3(.I0(d0),.I1(d1),.O(d2));
    INV i1(.I(q),.O(q1));
    FD fd1(.D(d2),.C(clk),.Q(Q[16]));
    shiftreg8b s1(.shift_in(Q[16]),.S_L(q),.clk(clk),.p_in({LED[0],LED[1],L
ED[2],LED[3],LED[4],LED[5],LED[6],LED[7]}),.Q(Q[15:8]));shiftreg8b s2(.shift
_in(Q[8]),.S_L(q),.clk(clk),.p_in({LED[8],LED[9],LED[10],LED[11],LED[12],LED
[13],LED[14],LED[15]}),.Q(Q[7:0]));
    AND8 band1(.A(Q[16:9]),.O(f0));
    AND8 band2(.A(Q[8:1]),.O(f1));
    AND2 n4(.I0(f0),.I1(f1),.O(finish));
    INV i2(.I(finish),.O(finish1));
    AND2 n5(.I0(start),.I1(finish),.O(s));
    NOR2 n6(.I0(s),.I1(q),.O(r));
    NOR2 n7(.I0(finish1),.I1(r),.O(q));
    INV i3(.I(Q[0]),.O(ser_out));
    assign LED_CLK=(clk||finish);
    endmodule
```

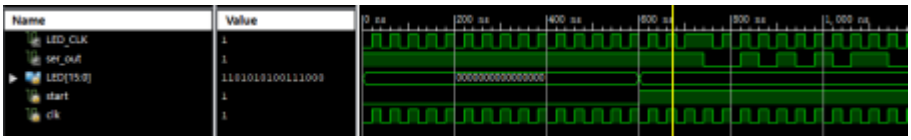
ADND8 模块使用原理图：



对于驱动模块 LED_DRV 进行仿真，代码如下：

```
module sim;
 // Inputs
 reg [15:0] LED;
 reg start;
 reg clk;
 // Outputs
 wire LED_CLK;
 wire ser_out;
 LED_DRV uut (
 .LED(LED),
 .start(start),
 .clk(clk),
 .LED_CLK(LED_CLK),
 .ser_out(ser_out)
 );
 initial begin
 LED=0;
 start = 0;
 #600;
 LED=16'b1101010100111000;
 start=1;
 #600;
 end
 always begin
 clk=0;#20;
 clk=1;#20;
 end
endmodule
```

得到的仿真结果如下：



下载验证

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

UCF 文件内容如下所示：

```
NET "SEGMENT[0]"LOC = AB22 | IOSTANDARD = LVCMOS33;
 NET "SEGMENT[1]" LOC = AD24 | IOSTANDARD = LVCMOS33;
 NET "SEGMENT[2]" LOC = AD23 | IOSTANDARD = LVCMOS33;
 NET "SEGMENT[3]" LOC = Y21 | IOSTANDARD = LVCMOS33;
 NET "SEGMENT[4]" LOC = W20 | IOSTANDARD = LVCMOS33;
 NET "SEGMENT[5]" LOC = AC24 | IOSTANDARD = LVCMOS33;
 NET "SEGMENT[6]" LOC = AC23 | IOSTANDARD = LVCMOS33;
 NET "SEGMENT[7]" LOC = AA22 | IOSTANDARD = LVCMOS33;

 NET "AN[0]" LOC = AD21 | IOSTANDARD = LVCMOS33;
 NET "AN[1]" LOC = AC21 | IOSTANDARD = LVCMOS33;
 NET "AN[2]" LOC = AB21 | IOSTANDARD = LVCMOS33;
 NET "AN[3]" LOC = AC22 | IOSTANDARD = LVCMOS33;

 NET "BTN[0]" LOC = W14 | IOSTANDARD = LVCMOS18;
 NET "BTN[0]" CLOCK_DEDICATED_ROUTE = FALSE;
 NET "BTN[1]" LOC = V14 | IOSTANDARD = LVCMOS18;
 NET "BTN[1]" CLOCK_DEDICATED_ROUTE = FALSE;
 NET "BTN[2]" LOC = V19 | IOSTANDARD = LVCMOS18;
 NET "BTN[2]" CLOCK_DEDICATED_ROUTE = FALSE;
 NET "BTN[3]" LOC = V18 | IOSTANDARD = LVCMOS18;
 NET "BTN[3]" CLOCK_DEDICATED_ROUTE = FALSE;
 NET "BTNX4" LOC = W16 | IOSTANDARD = LVCMOS18;

 NET "LED_CLK" LOC = N26 | IOSTANDARD = LVCMOS33 ;
 NET "LED_CLR" LOC = N24 | IOSTANDARD = LVCMOS33 ;
 NET "LED_DO" LOC = M26 | IOSTANDARD = LVCMOS33 ;
 NET "LED_EN" LOC = P18 | IOSTANDARD = LVCMOS33 ;

 NET "SW[14]" LOC = AF13 | IOSTANDARD = LVCMOS15;
```

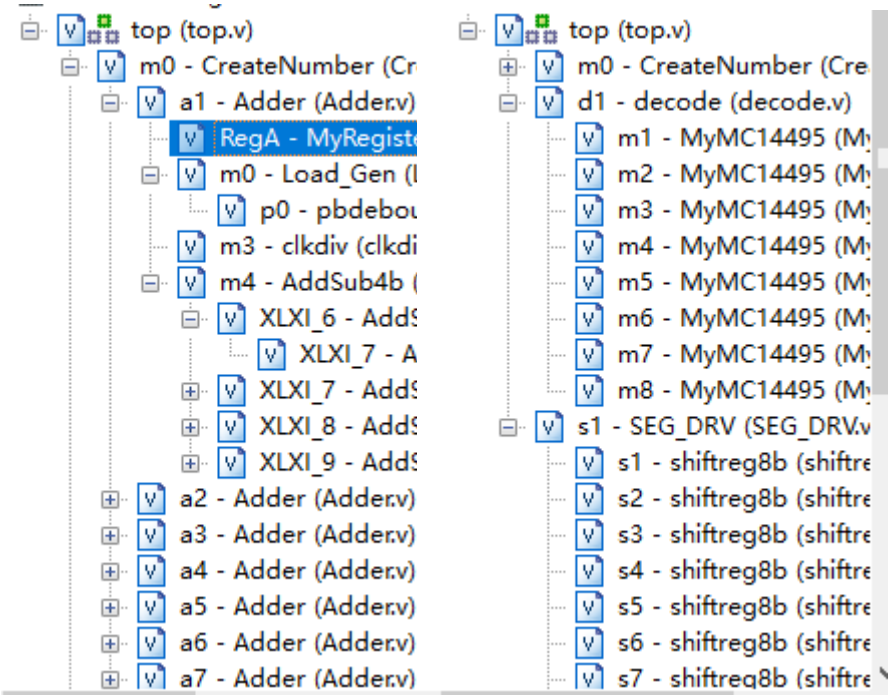
```
NET "SW[15]" LOC = AF10 | IOSTANDARD = LVCMOS15;

NET "clk" LOC = AC18 | IOSTANDARD = LVCMOS18;
```

任务 3：设计主板七段数码管驱动模块

新建工程，工程名称用 SEGP2S。Top Level Source Type 用 HDL

用结构化描述设计，利用 ShfitReg8b 模块，设计主板七段数码管驱动模块 SEG_DRV，调用 MyMC14495 模块进行段码译码

主要设计结构如下：



（左图为将 m0 内容打开，右图为将 m0 关闭）

主要模块代码如下：

1. top. v

```
module top(
    input wire [14:0]SW,
    input wire clk,
    output wire SEG_CLK,
    output wire SEG_CLR,
    output wire SEG_DT,
    output wire SEG_EN
    );
    wire [31:0]num;
    CreateNumber m0(.btn(SW[7:0]),.clk(clk),.num(num));
    wire [63:0]LED;
    decode d1(.num(num),.LED(LED));
```

```
    wire start,linshi1,linshi2;
    // use the sentencese below to describe or7
    OR5  o1 (.I0(SW[0]),
                  .I1(SW[1]),
                  .I2(SW[2]),
                  .I3(SW[3]),
                  .I4(SW[4]),
                  //.I5(SW[5]),
                  //.I6(SW[6]),
                       .O(linshi1));
    OR2 o2(.I0(SW[5]),
           .I1(SW[6]),
             .O(linshi2));
    OR2 o3(.I0(linshi1),
           .I1(linshi2),
             .O(start));
    SEG_DRV s1(.LED(LED),.start(start),.clk(clk),.SEG_CLK(SEG_CLK),.ser_out(
SEG_DT));
    assign SEG_CLR=SW[14];
    assign SEG_EN=1'b1;
    endmodule
```

CreateNumber 模块：

```
module CreateNumber(
    input wire [7:0] btn,
    input wire clk,
    output wire [31:0] num
    );
    wire [3:0] A1,B1,C1,D1,E1,F1,G1,H1;

    Adder a1(.SW(btn[0]),.clk(clk),.A(num[3:0]));
    Adder a2(.SW(btn[1]),.clk(clk),.A(num[7:4]));
    Adder a3(.SW(btn[2]),.clk(clk),.A(num[11:8]));
    Adder a4(.SW(btn[3]),.clk(clk),.A(num[15:12]));
    Adder a5(.SW(btn[4]),.clk(clk),.A(num[19:16]));
    Adder a6(.SW(btn[5]),.clk(clk),.A(num[23:20]));
    Adder a7(.SW(btn[6]),.clk(clk),.A(num[27:24]));
    Adder a8(.SW(btn[7]),.clk(clk),.A(num[31:28]));
    endmodule
```

Adder 模块：

```
module Adder(
    input wire SW,
    input wire clk,
    output wire [3:0]A
```

```
    );
    wire [3:0] Load;
    wire [3:0] A1,A_IN;
    wire [31:0] clk_div;
    MyRegister4b RegA(.clk(clk), .Input(A_IN), .Load(Load), .Output(A));
    Load_Gen m0(.clk(clk), .clk_1ms(clk_div[17]), .
    btn_in(SW),.Load_out(Load));
    clkdiv m3(clk, 1'b0, clk_div);
    AddSub4b m4(.A(A), .B(4'b0001), .Ctrl(1'b1), .S(A1));
    assign A_IN=A1;
   endmodule
```

decode 模块：

```
module decode(
    input wire [31:0]num,
    output wire [63:0]LED
    );
    MyMC14495 m1(.LE(1'b0),.point(1'b0),.D0(num[0]),.D1(
   num[1]),.D2(num[2]),.D3(num[3]),.a(LED[0]),.b(LED[1]
   ),.c(LED[2]),.d(LED[3]),.e(LED[4]),.f(LED[5]),.g(LED
   [6]),.p(LED[7]));
    MyMC14495 m2(.LE(1'b0),.point(1'b0),.D0(num[4]),.D1(
   num[5]),.D2(num[6]),.D3(num[7]),.a(LED[8]),.b(LED[9]
   ),.c(LED[10]),.d(LED[11]),.e(LED[12]),.f(LED[13]),.g
   (LED[14]),.p(LED[15]));
    MyMC14495 m3(.LE(1'b0),.point(1'b0),.D0(num[8]),.D1(num[9]),.D2(num[10])
,.D3(num[11]),.a(LED[16]),.b(LED
   [17]),.c(LED[18]),.d(LED[19]),.e(LED[20]),.f(LED[21]
   ),.g(LED[22]),.p(LED[23]));
    MyMC14495 m4(.LE(1'b0),.point(1'b0),.D0(num[12]),.D1(num[13]),.D2(num[14
]),.D3(num[15]),.a(LED[24]),.b(LED[25]),.c(LED[26]),.d(LED[27]),.e(LED[28]),
.f(LED[29]),.g(LED[30]),.p(LED[31]));
    MyMC14495 m5(.LE(1'b0),.point(1'b0),.D0(num[16]),.D1(num[17]),.D2(num[18
]),.D3(num[19]),.a(LED[32]),.b(
   LED[33]),.c(LED[34]),.d(LED[35]),.e(LED[36]),.f(LED[37]),.g(LED[38]),.p(L
ED[39]));
    MyMC14495 m6(.LE(1'b0),.point(1'b0),.D0(num[20]),.D1(num[21]),.D2(num[22
]),.D3(num[23]),.a(LED[40]),.b(
   LED[41]),.c(LED[42]),.d(LED[43]),.e(LED[44]),.f(LED[
   45]),.g(LED[46]),.p(LED[47]));
    MyMC14495 m7(.LE(1'b0),.point(1'b0),.D0(num[24]),.D1(num[25]),.D2(num[26
]),.D3(num[27]),.a(LED[48]),.b(
   LED[49]),.c(LED[50]),.d(LED[51]),.e(LED[52]),.f(LED[
   53]),.g(LED[54]),.p(LED[55]));
```

```
    MyMC14495 m8(.LE(1'b0),.point(1'b0),.D0(num[28]),.D1(num[29]),.D2(num[30
]),.D3(num[31]),.a(LED[56]),.b(
   LED[57]),.c(LED[58]),.d(LED[59]),.e(LED[60]),.f(LED[
   61]),.g(LED[62]),.p(LED[63]));
   endmodule
```

Pbdebounce 模块和 clkdiv 模块已经在前述实验中说明过。

运用 shiftreg8b 设计的 SEG_DRV：

```
module SEG_DRV(
    input wire [63:0]LED,
    input wire start,
    input wire clk,
    output wire SEG_CLK,
    output wire ser_out
    );
    wire [64:0]Q;
    wire q,r,s,q1,d0,d1,d2;
    wire finish1,finish;
    wire [7:0]f;
    AND2 n1(.I0(1'b1),.I1(q1),.O(d0));
    AND2 n2(.I0(1'b0),.I1(q),.O(d1));
    OR2 n3(.I0(d0),.I1(d1),.O(d2));
    INV i1(.I(q),.O(q1));
    FD fd1(.D(d2),.C(clk),.Q(Q[64]));
    shiftreg8b s1(.shift_in(Q[64]),.S_L(q),.clk(clk),.p_in({LED[0],LED[1],LE
D[2],LED[3],LED[4],LED[5],LED[6
   ],LED[7]}),.Q(Q[63:56]));
    shiftreg8b s2(.shift_in(Q[56]),.S_L(q),.clk(clk),.p_in({LED[8],LED[9],LE
D[10],LED[11],LED[12],LED[13],LED[14],LED[15]}),.Q(Q[55:48]));
    shiftreg8b s3(.shift_in(Q[48]),.S_L(q),.clk(clk),.p_in({LED[16],LED[17],
LED[18],LED[19],LED[20],LED[21]
   ,LED[22],LED[23]}),.Q(Q[47:40]));
    shiftreg8b s4(.shift_in(Q[40]),.S_L(q),.clk(clk),.p_in({LED[24],LED[25],
LED[26],LED[27],LED[28],LED[29]
   ,LED[30],LED[31]}),.Q(Q[39:32]));
    shiftreg8b s5(.shift_in(Q[32]),.S_L(q),.clk(clk),.p_in({LED[32],LED[33],
LED[34],LED[35],LED[36],LED[37]
   ,LED[38],LED[39]}),.Q(Q[31:24]));
    shiftreg8b s6(.shift_in(Q[24]),.S_L(q),.clk(clk),.p_in({LED[40],LED[41],
LED[42],LED[43],LED[44],LED[45]
   ,LED[46],LED[47]}),.Q(Q[23:16]));
    shiftreg8b s7(.shift_in(Q[16]),.S_L(q),.clk(clk),.p_in({LED[48],LED[49],
LED[50],LED[51],LED[52],LED[53]
   ,LED[54],LED[55]}),.Q(Q[15:8]));
```

```
    shiftreg8b s8(.shift_in(Q[8]),.S_L(q),.clk(clk),.p_in({LED[56],LED[57],L
ED[58],LED[59],LED[60],LED[61],
   LED[62],LED[63]}),.Q(Q[7:0]));

    AND8 band1(.A(Q[64:57]),.O(f[0]));
    AND8 band2(.A(Q[56:49]),.O(f[1]));
    AND8 band3(.A(Q[48:41]),.O(f[2]));
    AND8 band4(.A(Q[40:33]),.O(f[3]));
    AND8 band5(.A(Q[32:25]),.O(f[4]));
    AND8 band6(.A(Q[24:17]),.O(f[5]));
    AND8 band7(.A(Q[16:9]),.O(f[6]));
    AND8 band8(.A(Q[8:1]),.O(f[7]));
    AND8 band9(.A(f),.O(finish));

    INV i2(.I(finish),.O(finish1));
    AND2 n5(.I0(start),.I1(finish),.O(s));
    NOR2 n6(.I0(s),.I1(q),.O(r));
    NOR2 n7(.I0(finish1),.I1(r),.O(q));
    assign ser_out=Q[0];
    assign SEG_CLK=(clk||finish);
   endmodule
```

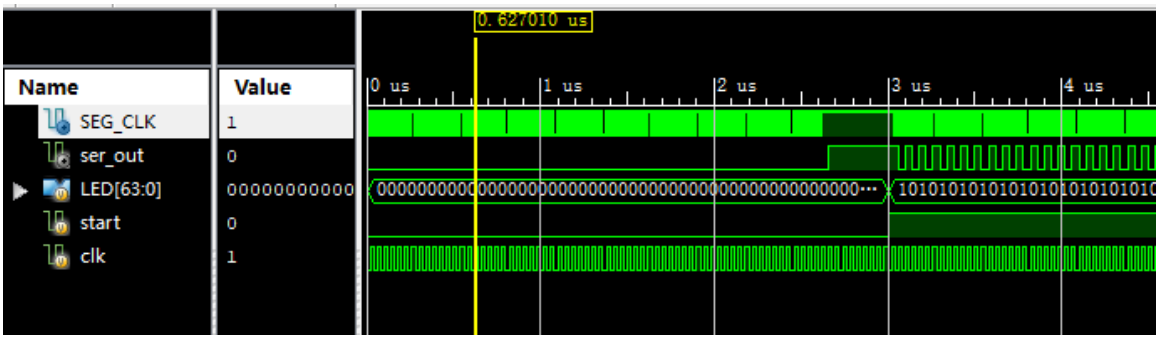
所使用的 AND8 已在前述章节中描述。

进行仿真，仿真代码如下：

```
module SEG_DRV_sim;
 // Inputs
 reg [63:0] LED;
 reg start;
 reg clk;
 // Outputs
 wire SEG_CLK;
 wire ser_out;
 // Instantiate the Unit Under Test (UUT)
 SEG_DRV uut (
 .LED(LED),
 .start(start),
 .clk(clk),
 .SEG_CLK(SEG_CLK),
 .ser_out(ser_out)
 );
 initial begin
 // Initialize Inputs
 LED = 64'b0000000000000000000000000000000000000000000000000000000000000000;
```

```
 start = 0;
 #3000;
 // Add stimulus
 LED = 64'b1010101010101010101010101010101010101010101010101111000010101010101010;
 start=1;
 end

 always begin
 clk=0;#20;
 clk=1;#20;
end

endmodule
```

结果如图所示：



在开发板上下载验证，用 8 个 SW 开关拨动来修改每个数码管的数值。

UCF 引脚约束文件如下：

```
NET "clk" LOC = AC18 | IOSTANDARD = LVCMOS18;

 NET "SW[0]" LOC = AA10 | IOSTANDARD = LVCMOS15;
 NET "SW[1]" LOC = AB10 | IOSTANDARD = LVCMOS15;
 NET "SW[2]" LOC = AA13 | IOSTANDARD = LVCMOS15;
 NET "SW[3]" LOC = AA12 | IOSTANDARD = LVCMOS15;
 NET "SW[4]" LOC = Y13 | IOSTANDARD = LVCMOS15;
 NET "SW[5]" LOC = Y12 | IOSTANDARD = LVCMOS15;
 NET "SW[6]" LOC = AD11 | IOSTANDARD = LVCMOS15;
 NET "SW[7]" LOC = AD10 | IOSTANDARD = LVCMOS15;
 NET "SW[14]" LOC = AF13 | IOSTANDARD = LVCMOS15;

 NET "SEG_CLK" LOC = M24 | IOSTANDARD = LVCMOS33 ;
 NET "SEG_CLR" LOC = M20 | IOSTANDARD = LVCMOS33 ;
 NET "SEG_DT" LOC = L24 | IOSTANDARD = LVCMOS33 ;
 NET "SEG_EN" LOC = R18 | IOSTANDARD = LVCMOS33 ;
```

四、实验结果分析

总体和预期实验结果内容一致。

硬件描述代码和原理图所要实现的功能一致。

仿真结果符合预期。

Verliog 代码内容实现了预期功能。

在开发板上的实验结果验证了应有的功能。

五、讨论与心得

在这次实验中，我们学会了移位寄存器设计与应用，了解了移位寄存器设计与应用的方法。我在实验中深刻体会到了实验当中出现的问题和解决的办法，以及协作的重要性。在这次实验当中，我在实验当中遇到了编写 Verilog 代码和 top 模块这几个问题，所幸通过老师和同学的帮助，这些问题顺利解决了，在开发板上面的实验结果也验证了这个结果。