浙江大學

本科实验报告

课程名称：	数字逻辑设计
姓　　名：	
学　　院：	竺可桢学院
专　　业：	混合班
指导教师：	董亚波
报告日期：	2025 年 5 月 15 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：　　移位寄存器设计与应用

学生姓名：__ 学号：___同组学生姓名：　无

实验地点：　紫金港东四 509 室　实验日期：　2025　年　5　月　14　日

一、 实验目的

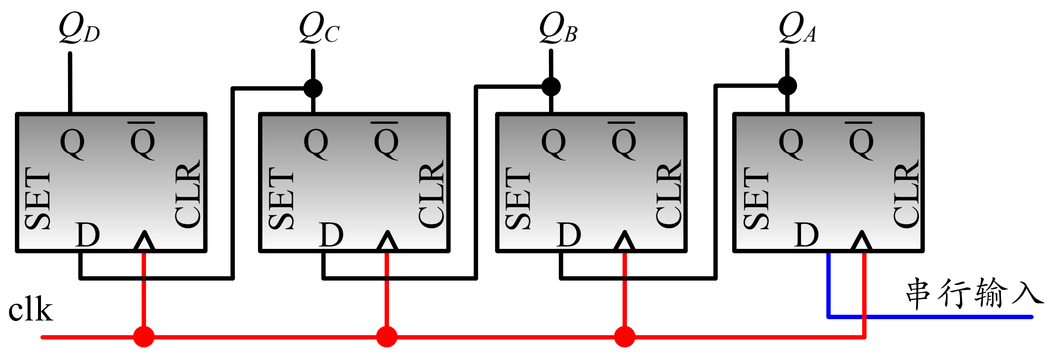
- ➢ 掌握支持并行输入的移位寄存器的工作原理
- ➢ 掌握支持并行输入的移位寄存器的设计方法

二、操作方法与实验步骤

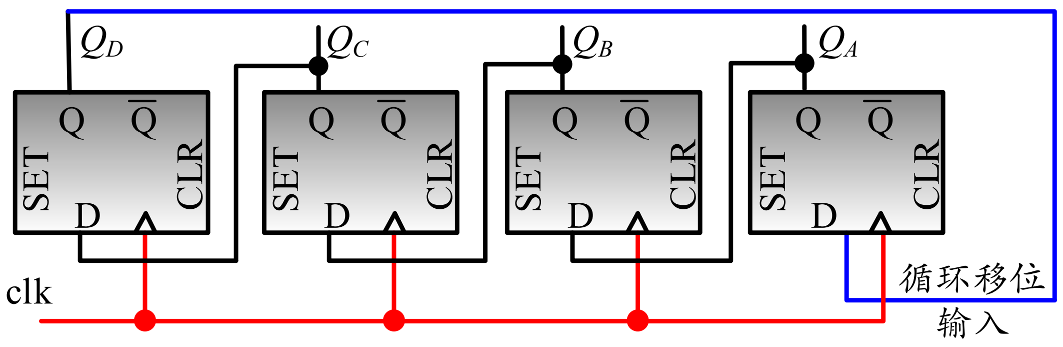
1. 实验原理

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，而不能采用锁存器）
- ➢ 数据移动方式：左移、右移、循环移位
- ➢ 数据输入输出方式有：串行输入，串行输出；串行输入，并行输出；并行输入，串行输出。
- ➢ 使用 D 触发器构成串行输入的左移移位寄存器如图所示：

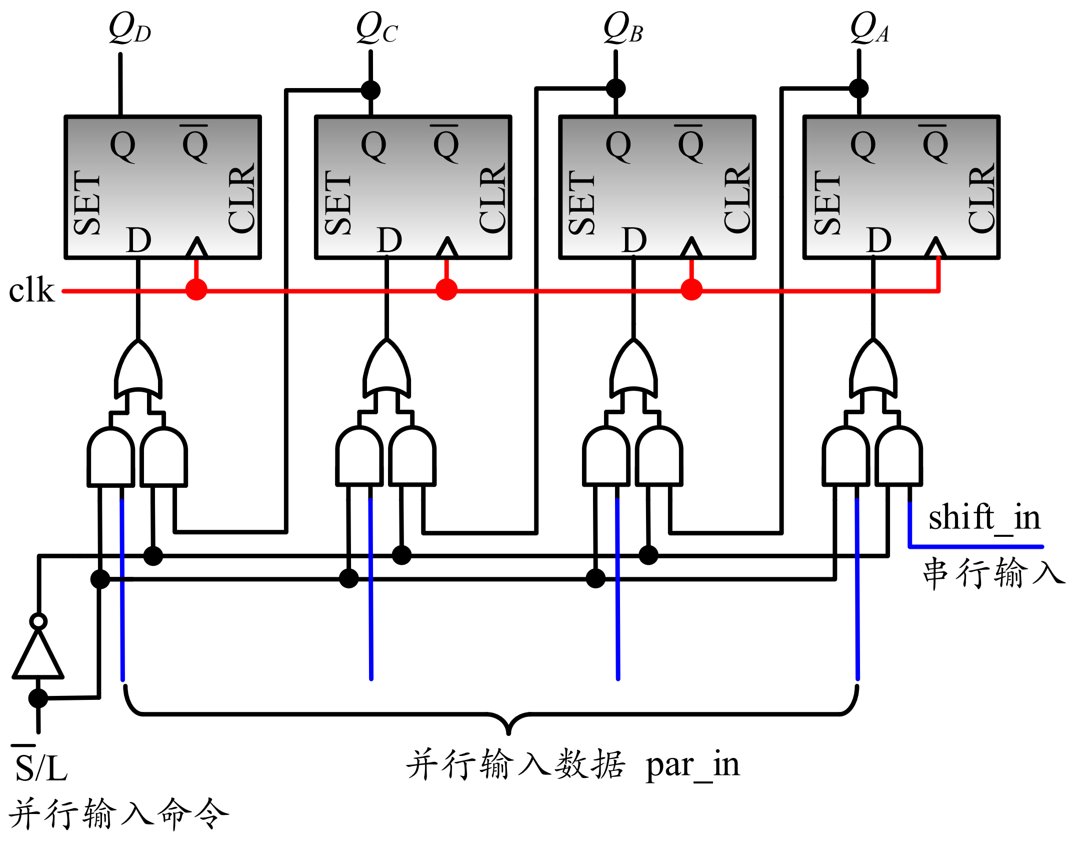


- ➢ 循环左移移位寄存器如图所示：



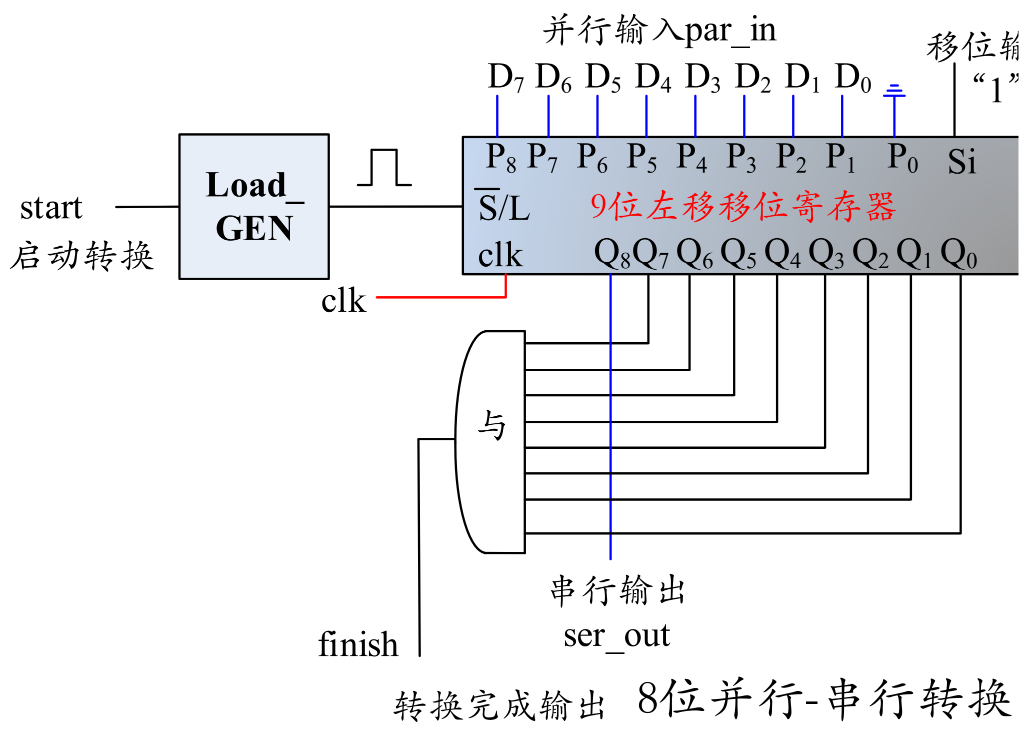
带并行输入的移位寄存器

- 数据输入方式：串行输入、并行输入
- 带并行输入的左移移位寄存器如图所示：



并行-串行转换器

- 设计的关键：串行数据输入完毕后需要停止时钟
- 采用门控时钟方式给 LED_CLK/SEG_CLK 提供时钟
- 8 位并行-串行转换器的原理：start 启动信号拉高以后，产生一个时钟周期宽度的并行加载信号，加载并行输入 D0-D7，然后启动左移串行输出，等 D0 输出后自动停止移位操作；finish 输出为 0 表示当前正在进行左移，为 1 表示移位停止。其原理图如下：



8位并行-串行转换

2. 设计 8 位带并行输入的左移移位寄存器

实验步骤如下：

- 在 vivado 中新建工程 shift_reg8b
- 新建源文件，设计 shift_reg8b.v 文件
- 设计 shift_reg8b.v 文件的仿真波形代码，进行行为仿真

3. 设计主板 16 位 LED 驱动模块

实验步骤如下：

- 在 Digital 中绘制 shfit_reg8b、 shfit_reg9b 模块原理图
- 在 vivado 中新建工程 LEDP2S
- 简化实验 11 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
- 利用 1 个 shfit_reg8b 模块和 1 个 shfit_reg9b 模块，设计 16 位 LED 驱动模块 LED_DRV.v
- 自行设计激励代码，对驱动模块进行仿真
- 在开发板上对其功能进行进一步验证

4. 设计主板 8 位数码管驱动模块

实验步骤如下：

- 在 vivado 中新建工程 SEGP2S
- 利用实验 11 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 7 个 shfit_reg8b 模块和 1 个 shfit_reg9b 模块，设计主板 8 位数码管

驱动模块 SEG_DRV.v

- 自行设计激励代码，对驱动模块进行仿真
- 在开发板上对其功能进行进一步验证

三、实验结果与分析

1. 设计 8 位带并行输入的左移移位寄存器

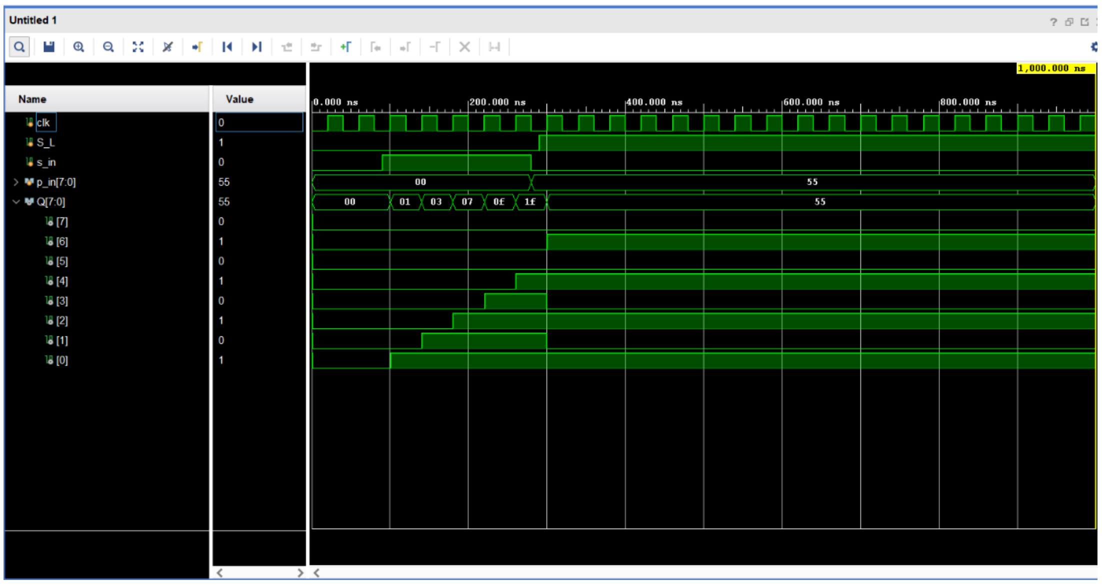
按要求创建工程，需要新建文件 shift_reg8b.v，代码如下（非 digital 导出）：

```
`timescale 1ns / 1ps
module shift_reg8b(
    input wire clk, S_L, s_in,
    input wire [7:0] p_in,
    output wire [7:0] Q);
    FD FD0(.C(clk), .D(D0), .Q(Q[0])),
       FD1(.C(clk), .D(D1), .Q(Q[1])),
       FD2(.C(clk), .D(D2), .Q(Q[2])),
       FD3(.C(clk), .D(D3), .Q(Q[3])),
       FD4(.C(clk), .D(D4), .Q(Q[4])),
       FD5(.C(clk), .D(D5), .Q(Q[5])),
       FD6(.C(clk), .D(D6), .Q(Q[6])),
       FD7(.C(clk), .D(D7), .Q(Q[7]));
    defparam FD0.INIT = 1'b0,
             FD1.INIT = 1'b0,
             FD2.INIT = 1'b0,
             FD3.INIT = 1'b0,
             FD4.INIT = 1'b0,
             FD5.INIT = 1'b0,
             FD6.INIT = 1'b0,
             FD7.INIT = 1'b0;
     OR2 O0(.I0(A0), .I1(B0), .O(D0)),
         O1(.I0(A1), .I1(B1), .O(D1)),
         O2(.I0(A2), .I1(B2), .O(D2)),
         O3(.I0(A3), .I1(B3), .O(D3)),
         O4(.I0(A4), .I1(B4), .O(D4)),
         O5(.I0(A5), .I1(B5), .O(D5)),
         O6(.I0(A6), .I1(B6), .O(D6)),
         O7(.I0(A7), .I1(B7), .O(D7));
     AND2 AA0(.I0(S_L), .I1(p_in[0]), .O(A0)),
          AA1(.I0(S_L), .I1(p_in[1]), .O(A1)),
          AA2(.I0(S_L), .I1(p_in[2]), .O(A2)),
          AA3(.I0(S_L), .I1(p_in[3]), .O(A3)),
          AA4(.I0(S_L), .I1(p_in[4]), .O(A4)),
          AA5(.I0(S_L), .I1(p_in[5]), .O(A5)),
          AA6(.I0(S_L), .I1(p_in[6]), .O(A6)),
          AA7(.I0(S_L), .I1(p_in[7]), .O(A7));
     AND2 BA0(.I0(Sbar), .I1(s_in), .O(B0)),
          BA1(.I0(Sbar), .I1(Q[0]), .O(B1)),
          BA2(.I0(Sbar), .I1(Q[1]), .O(B2)),
          BA3(.I0(Sbar), .I1(Q[2]), .O(B3)),
          BA4(.I0(Sbar), .I1(Q[3]), .O(B4)),
          BA5(.I0(Sbar), .I1(Q[4]), .O(B5)),
          BA6(.I0(Sbar), .I1(Q[5]), .O(B6)),
          BA7(.I0(Sbar), .I1(Q[6]), .O(B7));
     INV SI(.I(S_L), .O(Sbar));
 endmodule
```

对 shift_reg8b.v 进行仿真，仿真代码如下：

```
`timescale 1ns / 1ps
module shift_reg8b_testbench();
    reg clk, S_L, s_in;
    reg [7:0] p_in;
    wire [7:0] Q;
    shift_reg8b
uut(.clk(clk), .S_L(S_L), .s_in(s_in), .p_in(p_in), .Q(Q));
    initial begin
        p_in=0;
        S_L=0;
        s_in=0;
        #90
        s_in=1;
        #190
        s_in=0;
        p_in=8'h55;
        #10
        S_L=1;
    end
    always begin
        clk=0; #20;
        clk=1; #20;
    end
endmodule
```

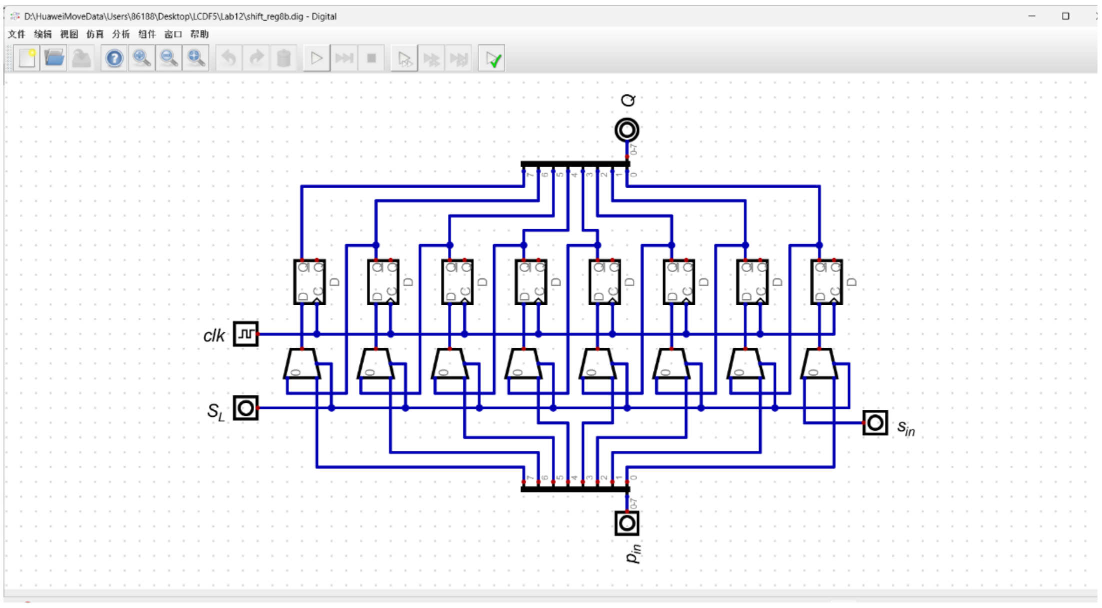
仿真波形如图所示：

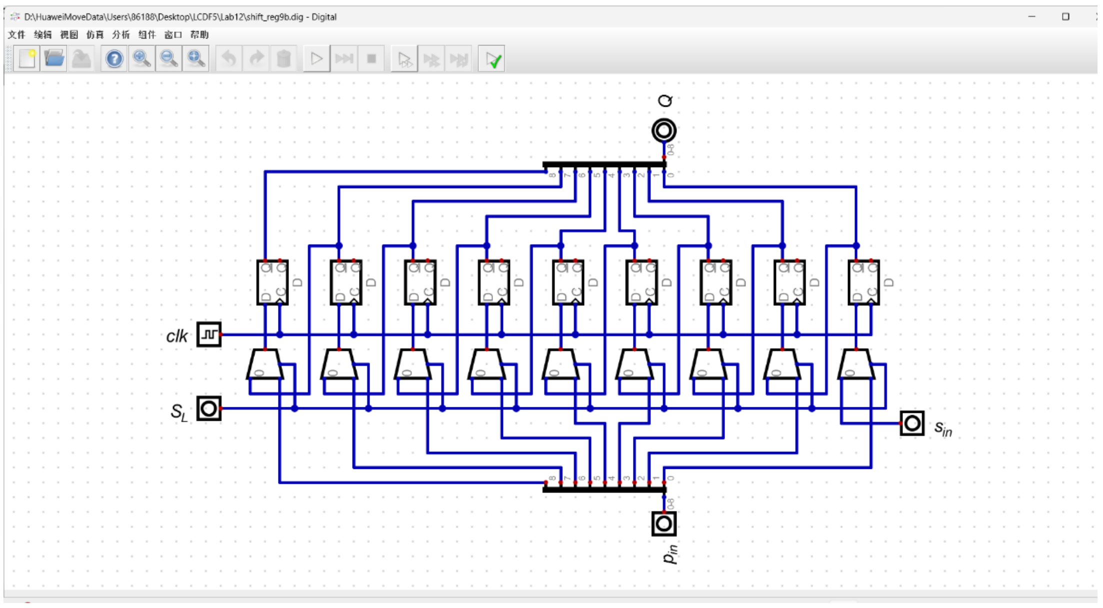


该仿真波形完成了五个 1 的串行输入以及 16 进制下的 55 的并行输入，与仿真波形示例一致，说明代码设计无误。

2. 设计主板 16 位 LED 驱动模块

在 digital 中绘制 shift_reg8b 及 shift_reg9b 的电路，如图所示：

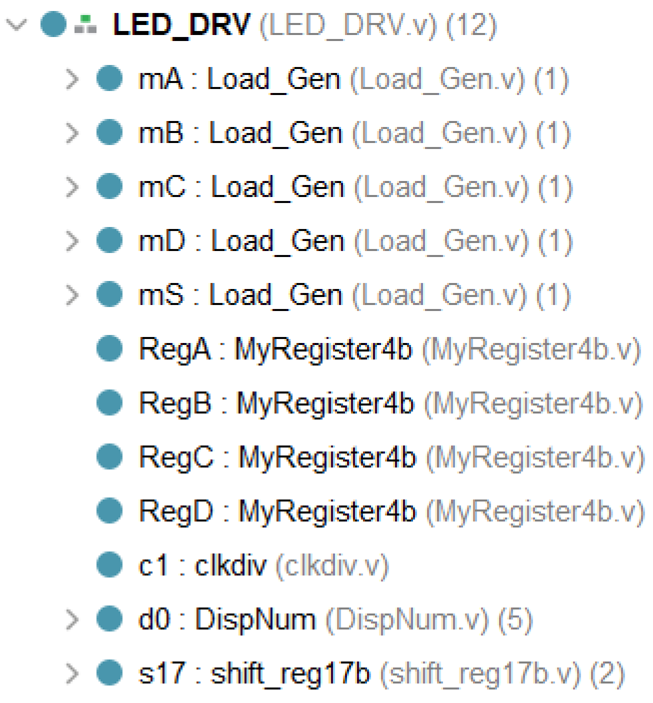

D:\HuaweiMoveData\Users\86188\Desktop\LCDF5\Lab12\shift_reg8b.dig - Digital
文件 编辑 视图 仿真 分析 组件 窗口 帮助
Q
clk
S_L
S_{in}
p_{in}


D:\HuaweiMoveData\Users\86188\Desktop\LCDF5\Lab12\shift_reg9b.dig - Digital
文件 编辑 视图 仿真 分析 组件 窗口 帮助
Q
clk
S_L
s_{in}
p_{in}

由于 verilog 代码过长，故不进行展示，只需将这两个电路分别导出为 verilog 代码即可。

按要求创建工程，需要添加之前的文件 clkdiv.v, DispNum.v, Load_Gen.v, MyRegister4b.v, pbdebounce.v, shift_reg8b.v, shift_reg9b.v（不再重复展示），并新建 shift_reg17b.v（用于封装两个寄存器）, LED_DRV.v。

文件结构如图所示：



```
LED_DRV (LED_DRV.v) (12)
    mA : Load_Gen (Load_Gen.v) (1)
    mB : Load_Gen (Load_Gen.v) (1)
    mC : Load_Gen (Load_Gen.v) (1)
    mD : Load_Gen (Load_Gen.v) (1)
    mS : Load_Gen (Load_Gen.v) (1)
    RegA : MyRegister4b (MyRegister4b.v)
    RegB : MyRegister4b (MyRegister4b.v)
    RegC : MyRegister4b (MyRegister4b.v)
    RegD : MyRegister4b (MyRegister4b.v)
    c1 : clkdiv (clkdiv.v)
    d0 : DispNum (DispNum.v) (5)
    s17 : shift_reg17b (shift_reg17b.v) (2)
```

shift_reg17b.v 代码如下：

```
`timescale 1ns / 1ps
module shift_reg17b(
    input S_L,
    input clk,
    input s_in,
    input [16:0] p_in,
    output [16:0] Q
);
    shift_reg9b
lower(.S_L(S_L), .clk(clk), .s_in(s_in), .p_in(p_in[8:0]), .Q(Q[8:0]));
    shift_reg8b
higher(.S_L(S_L), .clk(clk), .s_in(Q[8]), .p_in(p_in[16:9]), .Q(Q[16:9]
));
endmodule
```

LED_DRV.v 代码如下：

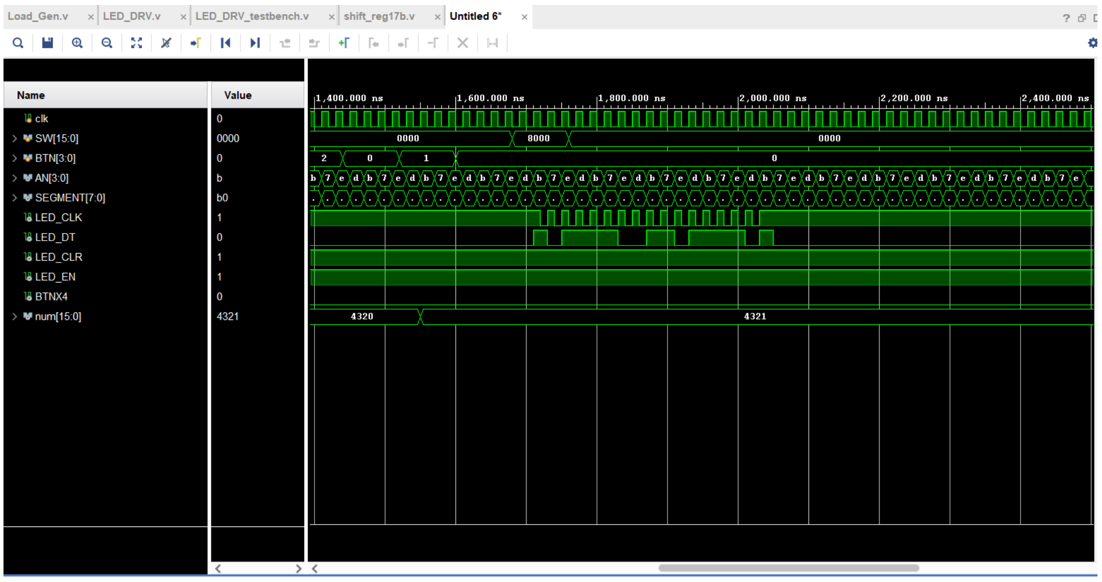
```
`timescale 1ns / 1ps
module LED_DRV(
    input wire clk,
    input wire [15:0] SW,
    input wire [3:0] BTN,
    output wire [3:0] AN,
    output wire [7:0] SEGMENT,
    output wire LED_CLK, LED_DT, LED_CLR, LED_EN,
    output wire BTNX4
);
    wire [15:0] num;
    wire [31:0] clk_div;
    wire [16:0] Q;
    wire finish;
    wire Load_A, Load_B, Load_C, Load_D, Load_S;
    wire [3:0] A, B, C, D, A_IN, B_IN, C_IN, D_IN;
    assign num={A,B,C,D};
    assign A_IN = A + 1;
    assign B_IN = B + 1;
    assign C_IN = C + 1;
    assign D_IN = D + 1;
    Load_Gen
mA(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(BTN[3]),.Load_out(Load_A))
;
    Load_Gen
mB(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(BTN[2]),.Load_out(Load_B))
;
    Load_Gen
mC(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(BTN[1]),.Load_out(Load_C))
;
    Load_Gen
mD(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(BTN[0]),.Load_out(Load_D))
;
```

```
    Load_Gen
mS(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(SW[15]),.Load_out(Load_S))
;
    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));
    clkdiv c1(.clk(clk), .rst(1'b0), .clk_div(clk_div));
    DispNum
d0(.scan(clk_div[18:17]), .HEXS(num), .LES(4'b0), .point(4'b0), .AN(AN)
, .SEGMENT(SEGMENT));
    shift_reg17b
s17(.S_L(Load_S), .clk(clk), .s_in(1), .p_in({num,1'b0}), .Q(Q));
    assign BTNX4 = 1'b0;
    assign LED_EN = 1'b1;
    assign LED_CLR = 1'b1;
    assign LED_DT = ~Q[16];
    assign finish = Q[15] & Q[14] & Q[13] & Q[12]
                 & Q[11] & Q[10] & Q[9] & Q[8]
                 & Q[7] & Q[6] & Q[5] & Q[4]
                 & Q[3] & Q[2] & Q[1] & Q[0];
    assign LED_CLK = clk | finish;
endmodule
```

对 LED_DRV.v 进行仿真，仿真代码如下：

```
`timescale 1ns / 1ps
module LED_DRV_testbench();
    reg clk;
    reg [15:0] SW;
    reg [3:0] BTN;
    wire [3:0] AN;
    wire [7:0] SEGMENT;
    wire LED_CLK, LED_DT, LED_CLR, LED_EN;
    wire BTNX4;
    LED_DRV
uut(.clk(clk),.SW(SW),.BTN(BTN),.AN(AN),.SEGMENT(SEGMENT),.LED_CLK(LED_
CLK),.LED_DT(LED_DT),.LED_CLR(LED_CLR),.LED_EN(LED_EN),.BTNX4(BTNX4));
    initial begin
        SW=0;
        BTN=0;
        #80;
        BTN[3]=1;
        #80;
        BTN[3]=0;
        #80;
        BTN[3]=1;
        #80;
        BTN[3]=0;
        #80;
        BTN[3]=1;
        #80;
        BTN[3]=0;
        #80;
        BTN[3]=1;
        #80;
        BTN[3]=0;

        #80;
        BTN[2]=1;
        #80;
        BTN[2]=0;
        #80;
        BTN[2]=1;
        #80;
        BTN[2]=0;
        #80;
        BTN[2]=1;
        #80;
        BTN[2]=0;

        #80;
        BTN[1]=1;
        #80;
        BTN[1]=0;
        #80;
        BTN[1]=1;
        #80;
        BTN[1]=0;

        #80;
        BTN[0]=1;
        #80;
        BTN[0]=0;

        #80
        SW[15]=1;
        #80
        SW[15]=0;
    end
    always begin
        clk=0;#10;
        clk=1;#10;
    end
endmodule
```

仿真波形如图所示（只截取最后一段）：

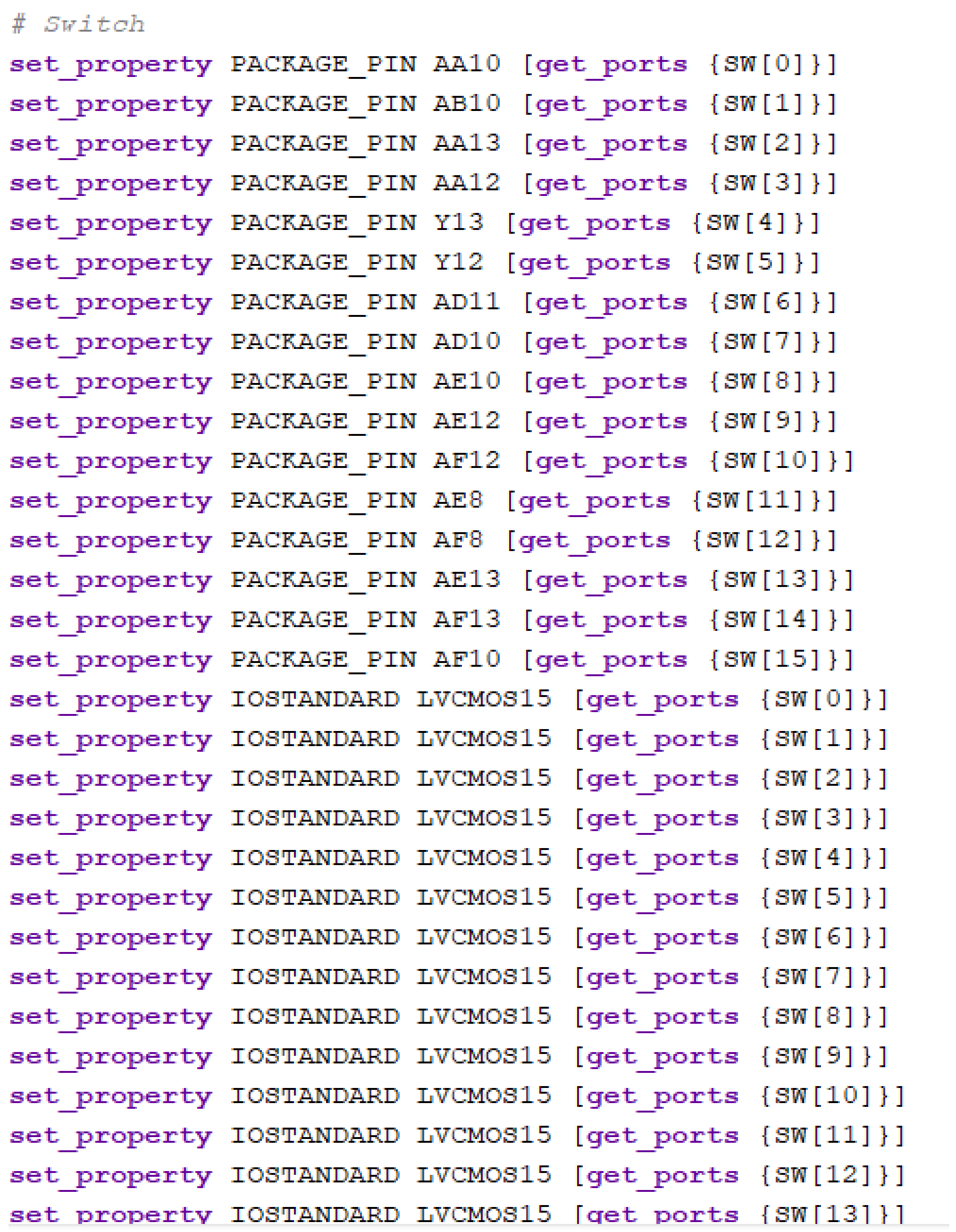


Name	Value
clk	0
SW[15:0]	0000
BTN[3:0]	0
AN[3:0]	b
SEGMENT[7:0]	b0
LED_CLK	1
LED_DT	0
LED_CLR	1
LED_EN	1
BTNX4	0
num[15:0]	4321

可以看到，这一段波形中，LED_DT 低电平部分即为 16 位二进制数的 1，进行换算发现与 4321h 一致，且在数据传输完成后，LED_CLK 成功置 1，说明代码无误。

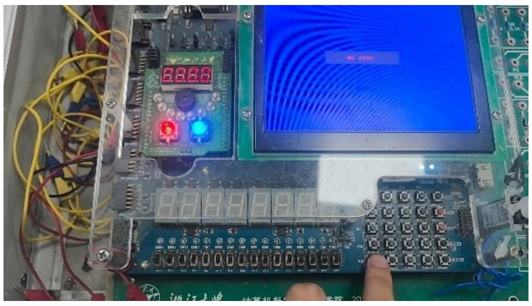
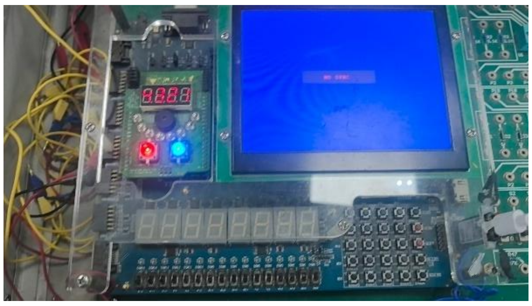
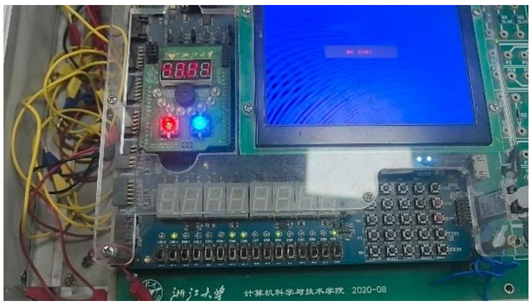
再将该工程进行上板验证，其约束文件如下：

```
create_clock -name clk100MHZ -period 10.0 [get_ports clk]
set_property PACKAGE_PIN AC18 [get_ports clk]
set_property IOSTANDARD LVCMOS18 [get_ports clk]

# Button
set_property PACKAGE_PIN W14 [get_ports BTN[0]]
set_property PACKAGE_PIN V14 [get_ports BTN[1]]
set_property PACKAGE_PIN V19 [get_ports BTN[2]]
set_property PACKAGE_PIN V18 [get_ports BTN[3]]
set_property PACKAGE_PIN W16 [get_ports BTNX4]
set_property IOSTANDARD LVCMOS18 [get_ports BTN[0]]
set_property IOSTANDARD LVCMOS18 [get_ports BTN[1]]
set_property IOSTANDARD LVCMOS18 [get_ports BTN[2]]
set_property IOSTANDARD LVCMOS18 [get_ports BTN[3]]
set_property IOSTANDARD LVCMOS18 [get_ports BTNX4]
```



```
# Switch
set_property PACKAGE_PIN AA10 [get_ports {SW[0]}]
set_property PACKAGE_PIN AB10 [get_ports {SW[1]}]
set_property PACKAGE_PIN AA13 [get_ports {SW[2]}]
set_property PACKAGE_PIN AA12 [get_ports {SW[3]}]
set_property PACKAGE_PIN Y13 [get_ports {SW[4]}]
set_property PACKAGE_PIN Y12 [get_ports {SW[5]}]
set_property PACKAGE_PIN AD11 [get_ports {SW[6]}]
set_property PACKAGE_PIN AD10 [get_ports {SW[7]}]
set_property PACKAGE_PIN AE10 [get_ports {SW[8]}]
set_property PACKAGE_PIN AE12 [get_ports {SW[9]}]
set_property PACKAGE_PIN AF12 [get_ports {SW[10]}]
set_property PACKAGE_PIN AE8 [get_ports {SW[11]}]
set_property PACKAGE_PIN AF8 [get_ports {SW[12]}]
set_property PACKAGE_PIN AE13 [get_ports {SW[13]}]
set_property PACKAGE_PIN AF13 [get_ports {SW[14]}]
set_property PACKAGE_PIN AF10 [get_ports {SW[15]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[0]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[1]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[2]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[3]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[4]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[5]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[6]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[7]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[8]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[9]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[10]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[11]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[12]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[13]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[14]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[15]}]

# 7-Segment LED
set_property PACKAGE_PIN AB22 [get_ports {SEGMENT[0]}]
set_property PACKAGE_PIN AD24 [get_ports {SEGMENT[1]}]
set_property PACKAGE_PIN AD23 [get_ports {SEGMENT[2]}]
set_property PACKAGE_PIN Y21 [get_ports {SEGMENT[3]}]
set_property PACKAGE_PIN W20 [get_ports {SEGMENT[4]}]
set_property PACKAGE_PIN AC24 [get_ports {SEGMENT[5]}]
set_property PACKAGE_PIN AC23 [get_ports {SEGMENT[6]}]
set_property PACKAGE_PIN AA22 [get_ports {SEGMENT[7]}]
set_property PACKAGE_PIN AD21 [get_ports {AN[0]}]
set_property PACKAGE_PIN AC21 [get_ports {AN[1]}]
set_property PACKAGE_PIN AB21 [get_ports {AN[2]}]
set_property PACKAGE_PIN AC22 [get_ports {AN[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[3]}]

# LED Serial
set_property PACKAGE_PIN N26 [get_ports LED_CLK]
set_property PACKAGE_PIN N24 [get_ports LED_CLR]
set_property PACKAGE_PIN M26 [get_ports LED_DT]
set_property PACKAGE_PIN P18 [get_ports LED_EN]
set_property IOSTANDARD LVCMOS33 [get_ports LED_CLK]
set_property IOSTANDARD LVCMOS33 [get_ports LED_CLR]
set_property IOSTANDARD LVCMOS33 [get_ports LED_DT]
set_property IOSTANDARD LVCMOS33 [get_ports LED_EN]
```

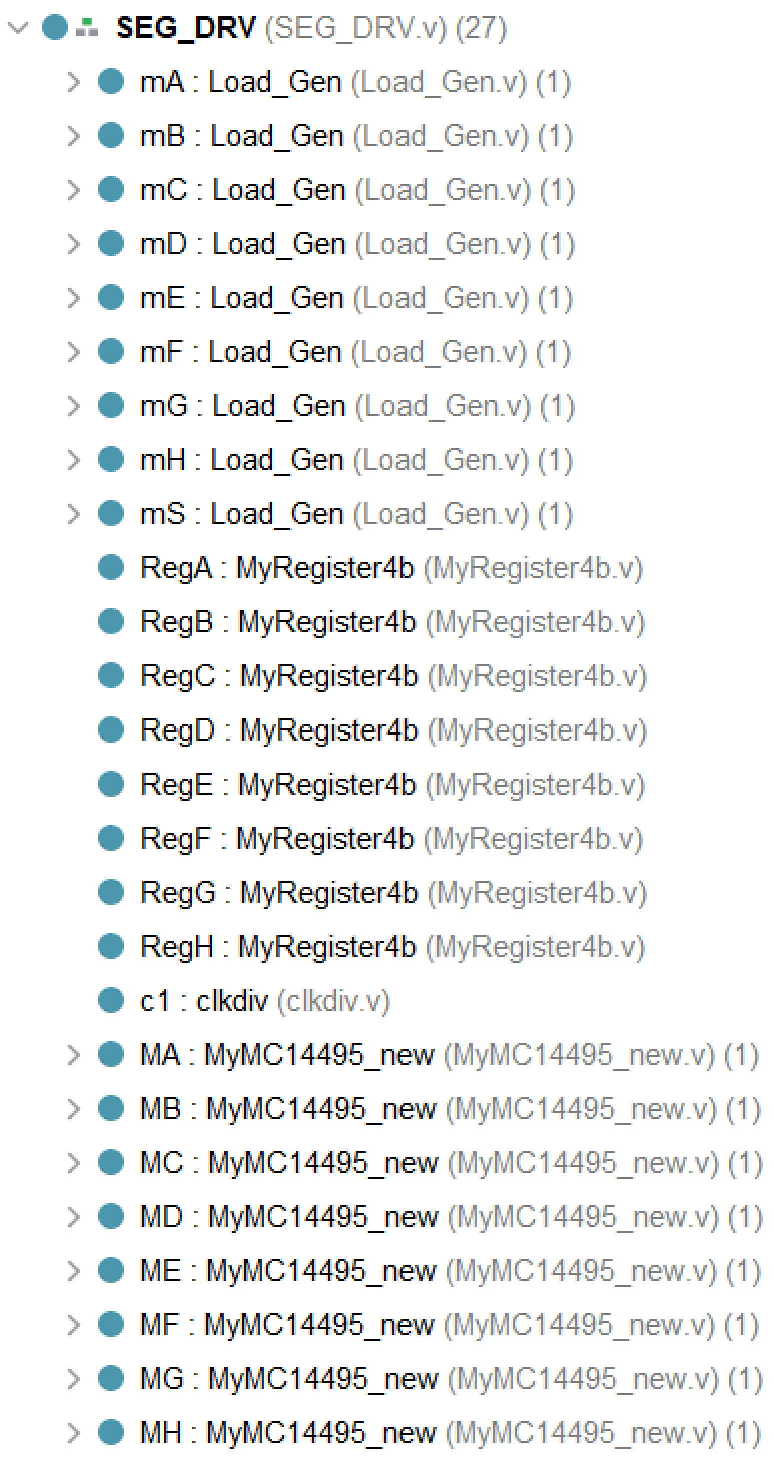
上板结果如图所示：

由于重复操作较多，故只展示了下载完成、设好数字以及最终显示这三张图片。可以看到，将 LED 灯灭对应 0，亮对应 1 以后，其 16 位二进制数值与 4321h 一致。

3. 设计主板 8 位数码管驱动模块

按要求创建工程，需要添加之前的文件 clkdiv.v, MyMC14495.v, Load_Gen.v, MyRegister4b.v, pbdebounce.v, shift_reg8b.v, shift_reg9b.v（不再重复展示），并新建 MyMC14495_new.v（用于封装数码管显示）, shift_reg65b.v（用于封装八个寄存器）, SEG_DRV.v。
文件结构如图所示：



SEG_DRV (SEG_DRV.v) (27)
mA : Load_Gen (Load_Gen.v) (1)
mB : Load_Gen (Load_Gen.v) (1)
mC : Load_Gen (Load_Gen.v) (1)
mD : Load_Gen (Load_Gen.v) (1)
mE : Load_Gen (Load_Gen.v) (1)
mF : Load_Gen (Load_Gen.v) (1)
mG : Load_Gen (Load_Gen.v) (1)
mH : Load_Gen (Load_Gen.v) (1)
mS : Load_Gen (Load_Gen.v) (1)
RegA : MyRegister4b (MyRegister4b.v)
RegB : MyRegister4b (MyRegister4b.v)
RegC : MyRegister4b (MyRegister4b.v)
RegD : MyRegister4b (MyRegister4b.v)
RegE : MyRegister4b (MyRegister4b.v)
RegF : MyRegister4b (MyRegister4b.v)
RegG : MyRegister4b (MyRegister4b.v)
RegH : MyRegister4b (MyRegister4b.v)
c1 : clkdiv (clkdiv.v)
MA : MyMC14495_new (MyMC14495_new.v) (1)
MB : MyMC14495_new (MyMC14495_new.v) (1)
MC : MyMC14495_new (MyMC14495_new.v) (1)
MD : MyMC14495_new (MyMC14495_new.v) (1)
ME : MyMC14495_new (MyMC14495_new.v) (1)
MF : MyMC14495_new (MyMC14495_new.v) (1)
MG : MyMC14495_new (MyMC14495_new.v) (1)
MH : MyMC14495_new (MyMC14495_new.v) (1)

MyMC14495_new.v 代码如下：

```
`timescale 1ns / 1ps
module MyMC14495_new(
    input [3:0] D,
    input point,
    input LE,
    output [7:0] s
);
```

```
    MyMC14495
old(.D3(D[3]), .D2(D[2]), .D1(D[1]), .D0(D[0]), .point(point), .LE(LE),
.p(s[7]), .g(s[6]), .f(s[5]), .e(s[4]), .d(s[3]), .c(s[2]), .b(s[1]), .
a(s[0]));
endmodule
```

shift_reg65b.v 代码如下：

```
`timescale 1ns / 1ps
module shift_reg65b(
    input S_L,
    input clk,
    input s_in,
    input [64:0] p_in,
    output [64:0] Q
);
    shift_reg9b
r0(.S_L(S_L), .clk(clk), .s_in(s_in), .p_in(p_in[8:0]), .Q(Q[8:0]));
    shift_reg8b
r1(.S_L(S_L), .clk(clk), .s_in(Q[8]), .p_in(p_in[16:9]), .Q(Q[16:9]));
    shift_reg8b
r2(.S_L(S_L), .clk(clk), .s_in(Q[16]), .p_in(p_in[24:17]), .Q(Q[24:17])
);
    shift_reg8b
r3(.S_L(S_L), .clk(clk), .s_in(Q[24]), .p_in(p_in[32:25]), .Q(Q[32:25])
);
    shift_reg8b
r4(.S_L(S_L), .clk(clk), .s_in(Q[32]), .p_in(p_in[40:33]), .Q(Q[40:33])
);
    shift_reg8b
r5(.S_L(S_L), .clk(clk), .s_in(Q[40]), .p_in(p_in[48:41]), .Q(Q[48:41])
);
    shift_reg8b
r6(.S_L(S_L), .clk(clk), .s_in(Q[48]), .p_in(p_in[56:49]), .Q(Q[56:49])
);
    shift_reg8b
r7(.S_L(S_L), .clk(clk), .s_in(Q[56]), .p_in(p_in[64:57]), .Q(Q[64:57])
);
endmodule
```

SEG_DRV.v 代码如下：

```
`timescale 1ns / 1ps
module SEG_DRV(
    input wire clk,
    input wire [15:0] SW,
    output wire SEG_CLK, SEG_DT, SEG_CLR, SEG_EN
);
```

```
    wire [31:0] num;
    wire [63:0] seg;
    wire [31:0] clk_div;
    wire [64:0] Q;
    wire finish;
    wire Load_A, Load_B, Load_C, Load_D, Load_E, Load_F, Load_G, Load_H,
Load_S;
    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;
    assign num={A,B,C,D,E,F,G,H};
    assign A_IN = A + 1;
    assign B_IN = B + 1;
    assign C_IN = C + 1;
    assign D_IN = D + 1;
    assign E_IN = E + 1;
    assign F_IN = F + 1;
    assign G_IN = G + 1;
    assign H_IN = H + 1;
    Load_Gen
mA(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(SW[7]),.Load_out(Load_A));
    Load_Gen
mB(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(SW[6]),.Load_out(Load_B));
    Load_Gen
mC(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(SW[5]),.Load_out(Load_C));
    Load_Gen
mD(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(SW[4]),.Load_out(Load_D));
    Load_Gen
mE(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(SW[3]),.Load_out(Load_E));
    Load_Gen
mF(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(SW[2]),.Load_out(Load_F));
    Load_Gen
mG(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(SW[1]),.Load_out(Load_G));
    Load_Gen
mH(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(SW[0]),.Load_out(Load_H));
    Load_Gen
mS(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(SW[15]),.Load_out(Load_S))
;
//    assign Load_S = Load_A | Load_B | Load_C | Load_D | Load_E |
Load_F | Load_G | Load_H;
    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));
    clkdiv c1(.clk(clk), .rst(1'b0), .clk_div(clk_div));
    MyMC14495_new MA(.D(A), .point(1'b0), .LE(1'b0), .s(seg[63:56]));
    MyMC14495_new MB(.D(B), .point(1'b0), .LE(1'b0), .s(seg[55:48]));
    MyMC14495_new MC(.D(C), .point(1'b0), .LE(1'b0), .s(seg[47:40]));
    MyMC14495_new MD(.D(D), .point(1'b0), .LE(1'b0), .s(seg[39:32]));
    MyMC14495_new ME(.D(E), .point(1'b0), .LE(1'b0), .s(seg[31:24]));
    MyMC14495_new MF(.D(F), .point(1'b0), .LE(1'b0), .s(seg[23:16]));
    MyMC14495_new MG(.D(G), .point(1'b0), .LE(1'b0), .s(seg[15:8]));
    MyMC14495_new MH(.D(H), .point(1'b0), .LE(1'b0), .s(seg[7:0]));
    shift_reg65b
s65(.S_L(Load_S), .clk(clk), .s_in(1), .p_in({seg,1'b0}), .Q(Q));
    assign SEG_EN = 1'b1;
    assign SEG_CLR = 1'b1;
    assign SEG_DT = Q[64];
    assign finish = (~Q[63:0]) == 0;
    assign SEG_CLK = clk | finish;
endmodule
```

对 SEG_DRV.v 进行仿真，仿真代码如下：

```
`timescale 1ns / 1ps
module SEG_DRV_testbench();
    reg clk;
    reg [15:0] SW;
    wire SEG_CLK, SEG_DT, SEG_CLR, SEG_EN;
    SEG_DRV
uut(.clk(clk),.SW(SW),.SEG_CLK(SEG_CLK),.SEG_DT(SEG_DT),.SEG_CLR(SEG_CL
R),.SEG_EN(SEG_EN));
    integer i;
    initial begin
        SW=0;
        for(i=1;i<=X;i=i+1) begin
            #80
            SW[7]=1;
            #80
            SW[7]=0;
            end
        #80
        SW[5]=1;
        #80
        SW[5]=0;
        for(i=1;i<=X;i=i+1) begin
            #80
            SW[2]=1;
            #80
            SW[2]=0;
            end
        for(i=1;i<=X;i=i+1) begin
            #80
            SW[1]=1;
            #80
            SW[1]=0;
            end
        for(i=1;i<=X;i=i+1) begin
            #80
            SW[0]=1;
            #80
            SW[0]=0;
            end
        #80
        SW[15]=1;
        #80
        SW[15]=0;
    end
    always begin
        clk=0;#10;
        clk=1;#10;
    end
endmodule
```

仿真波形如图所示（只截取最后一段）：

（含有本人学号，删去）

由于数码管对应的二进制串较抽象，在仿真波形中可以判断的只有 SEG_CLK 停止计时成功了，至于数码管能否正常显示，需要上板验证。

将其下载到板上进行验证，约束文件如下：

```
create_clock -name clk100MHZ -period 10.0 [get_ports clk]
set_property PACKAGE_PIN AC18 [get_ports clk]
set_property IOSTANDARD LVCMOS18 [get_ports clk]
```

```
# Switch
set_property PACKAGE_PIN AA10 [get_ports {SW[0]}]
set_property PACKAGE_PIN AB10 [get_ports {SW[1]}]
set_property PACKAGE_PIN AA13 [get_ports {SW[2]}]
set_property PACKAGE_PIN AA12 [get_ports {SW[3]}]
set_property PACKAGE_PIN Y13 [get_ports {SW[4]}]
set_property PACKAGE_PIN Y12 [get_ports {SW[5]}]
set_property PACKAGE_PIN AD11 [get_ports {SW[6]}]
set_property PACKAGE_PIN AD10 [get_ports {SW[7]}]
set_property PACKAGE_PIN AE10 [get_ports {SW[8]}]
set_property PACKAGE_PIN AE12 [get_ports {SW[9]}]
set_property PACKAGE_PIN AF12 [get_ports {SW[10]}]
set_property PACKAGE_PIN AE8 [get_ports {SW[11]}]
set_property PACKAGE_PIN AF8 [get_ports {SW[12]}]
set_property PACKAGE_PIN AE13 [get_ports {SW[13]}]
set_property PACKAGE_PIN AF13 [get_ports {SW[14]}]
set_property PACKAGE_PIN AF10 [get_ports {SW[15]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[0]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[1]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[2]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[3]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[4]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[5]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[6]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[7]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[8]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[9]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[10]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[11]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[12]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[13]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[14]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[15]}]

# 7-Segment Serial
set_property PACKAGE_PIN M24 [get_ports SEG_CLK]
set_property PACKAGE_PIN M20 [get_ports SEG_CLR]
set_property PACKAGE_PIN L24 [get_ports SEG_DT]
set_property PACKAGE_PIN R18 [get_ports SEG_EN]
set_property IOSTANDARD LVCMOS33 [get_ports SEG_CLK]
set_property IOSTANDARD LVCMOS33 [get_ports SEG_CLR]
set_property IOSTANDARD LVCMOS33 [get_ports SEG_DT]
set_property IOSTANDARD LVCMOS33 [get_ports SEG_EN]
```

上板结果如图所示：

（含有本人学号，删去）　　（含有本人学号，删去）

（含有本人学号，删去）　　（含有本人学号，删去）

由于拉下开关的操作并不能及时反馈吗，这里只在视频中截取了几个关键状态，分别为下载完成、第一位设置完成、二三位设置完成以及全部设置完成。可以看到，该数码管已经可以正常显示本人学号的后 8 位，说明整个工程无误。

四、讨论、心得

这一次实验中，我学会了移位寄存器的设计及应用，整体而言，这一次实验任务难度较大，需要不少时间才能完成。但在实验通过验收后，我对移位寄存器的原理以及开发板上的 LED、8 位数码管的使用也有了更多的了解。