浙江大学

本科实验报告

课程名称：	数字逻辑设计
姓　　名：	王浩雄
学　　院：	竺可桢学院
系：	混合班
专　　业：	计算机科学与技术
学　　号：	3230106032
指导教师：	马德

2025 年 5 月 14 日

浙江大学实验报告

课程名称：数字逻辑设计 实验类型：

实验项目名称：移位寄存器设计与应用

学生姓名：王浩雄 专业：混合班 学号：3230106032

同组学生姓名：无 指导老师：马德

实验地点：紫金港东 4-509 实验日期：2025 年 5 月 14 日

一、 实验目的和要求

① 掌握支持并行输入的移位寄存器的工作原理；

② 掌握支持并行输入的移位寄存器的设计方法。

二、 实验内容

① 设计 8 位带并行输入的右移移位寄存器

② 设计主板 LED 灯驱动模块

③ 设计主板七段数码管驱动模块

三、 主要仪器设备

① 装有 Vivado 2024.2 Enterprise 的计算机 1 台

② SWORD 开发板 1 套

四、 操作方法与实验步骤

1、设计 8 位带并行输入的右移移位寄存器

① 根据设计要求，新建 Verilog 文件，以行为描述方式完成 8 位带并行输入的右移移位寄存器模块的设计。其中，S_L 为并行输入/串行输入选择位（1 表示并行输入，0 表示串行输入），s_in 为串行数据输入端口，p_in 为并行数据输入端口，Q 为并行数据输出端口。

```
module shiftreg_8bit(
  input wire clk, S_L, s_in,
  input wire [7:0] p_in,    // 1 表示并行输入，0 表示串行输入
  output wire [7:0] Q
  );

  wire [7:0] mux_out;
  wire [7:0] shift_in;

  // Invert S_L
  wire nS_L;
  INV inv0 (.I(S_L), .O(nS_L));

  // bit 7
  wire and0_out, and1_out;
  AND2 and0 (.I0(p_in[7]), .I1(S_L), .O(and0_out));
  AND2 and1 (.I0(s_in), .I1(nS_L), .O(and1_out));
  OR2  or0  (.I0(and0_out), .I1(and1_out), .O(mux_out[7]));
  FD   fd0  (.C(clk), .D(mux_out[7]), .Q(Q[7]));

  // bit 6
  wire and2_out, and3_out;
  AND2 and2 (.I0(p_in[6]), .I1(S_L), .O(and2_out));
  AND2 and3 (.I0(Q[7]), .I1(nS_L), .O(and3_out));
  OR2  or1  (.I0(and2_out), .I1(and3_out), .O(mux_out[6]));
  FD   fd1  (.C(clk), .D(mux_out[6]), .Q(Q[6]));

  // bit 5
  wire and4_out, and5_out;
  AND2 and4 (.I0(p_in[5]), .I1(S_L), .O(and4_out));
  AND2 and5 (.I0(Q[6]), .I1(nS_L), .O(and5_out));
  OR2  or2  (.I0(and4_out), .I1(and5_out), .O(mux_out[5]));
  FD   fd2  (.C(clk), .D(mux_out[5]), .Q(Q[5]));
```

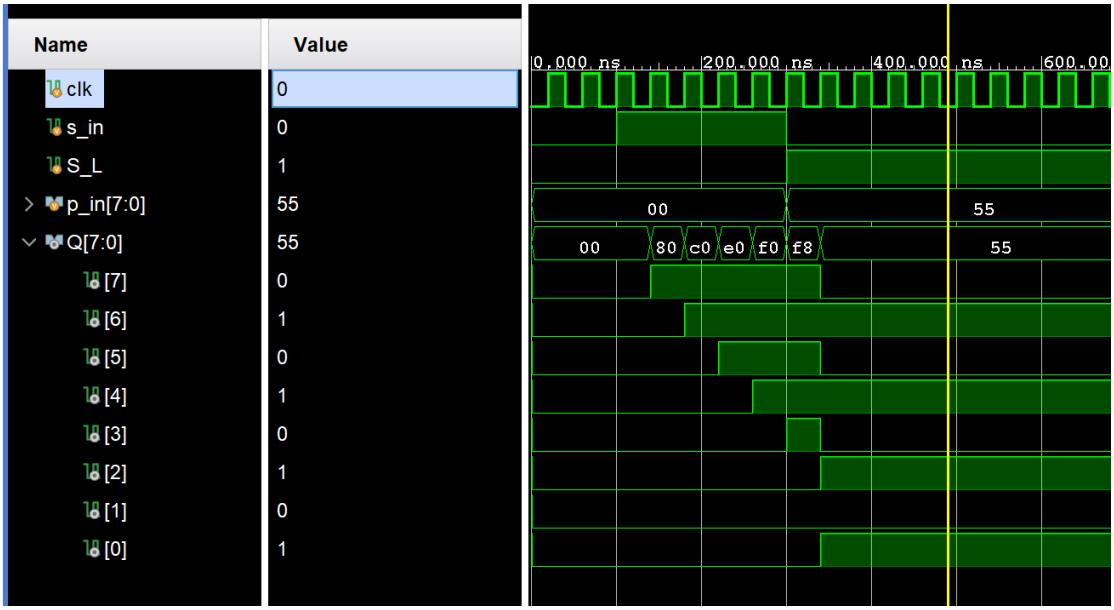
```
  // bit 4
  wire and0_out, and1_out;
  AND2 and6 (.I0(p_in[4]), .I1(S_L), .O(and6_out));
  AND2 and7 (.I0(Q[5]), .I1(nS_L), .O(and7_out));
  OR2  or3  (.I0(and6_out), .I1(and7_out), .O(mux_out[4]));
  FD   fd3  (.C(clk), .D(mux_out[4]), .Q(Q[4]));

  // bit 3
  wire and0_out, and1_out;
  AND2 and8  (.I0(p_in[3]), .I1(S_L), .O(and8_out));
  AND2 and9  (.I0(Q[4]), .I1(nS_L), .O(and9_out));
  OR2  or4   (.I0(and8_out), .I1(and9_out), .O(mux_out[3]));
  FD   fd4   (.C(clk), .D(mux_out[3]), .Q(Q[3]));

  // bit 2
  wire and0_out, and1_out;
  AND2 and10 (.I0(p_in[2]), .I1(S_L), .O(and10_out));
  AND2 and11 (.I0(Q[3]), .I1(nS_L), .O(and11_out));
  OR2  or5   (.I0(and10_out), .I1(and11_out), .O(mux_out[2]));
  FD   fd5   (.C(clk), .D(mux_out[2]), .Q(Q[2]));

  // bit 1
  wire and0_out, and1_out;
  AND2 and12 (.I0(p_in[1]), .I1(S_L), .O(and12_out));
  AND2 and13 (.I0(Q[2]), .I1(nS_L), .O(and13_out));
  OR2  or6   (.I0(and12_out), .I1(and13_out), .O(mux_out[1]));
  FD   fd6   (.C(clk), .D(mux_out[1]), .Q(Q[1]));

  // bit 0
  wire and0_out, and1_out;
  AND2 and14 (.I0(p_in[0]), .I1(S_L), .O(and14_out));
  AND2 and15 (.I0(Q[1]), .I1(nS_L), .O(and15_out));
  OR2  or7   (.I0(and14_out), .I1(and15_out), .O(mux_out[0]));
  FD   fd7   (.C(clk), .D(mux_out[0]), .Q(Q[0]));

endmodule
```

② 新建仿真激励文件如下：

```
module shiftreg_8b_tb();
    reg clk;
    reg s_in;
    reg S_L;
    reg [7:0] p_in;
```

```
    wire [7:0]Q;

    shiftreg_8bit UUT (
    .Q(Q),
    .clk(clk),
    .s_in(s_in),
    .S_L(S_L),
    .p_in(p_in)
    );

initial begin
    clk = 0;
    S_L = 0;
    s_in = 0;
    p_in = 0;
    #100;
    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
endmodule
```

③ 运行仿真，得到如下的仿真波形。该波形与预期相符，表明移位寄存器设计的正确性。



2、设计主板 LED 灯驱动模块

① 编写 LED 灯驱动代码如下：

```
module LED_driver (
  input wire clk,
  input wire reset,
  input wire start,
  input wire [15:0] num,
  output wire LED_DO,
  output wire LED_CLK,
  output wire LED_CLR,
  output wire LED_EN
);

  reg [15:0] shift_reg;
  reg [3:0] counter;
  reg shifting;

  assign LED_CLR = ~reset;
  assign LED_DO = ~shift_reg[15];            // 输出最高位的取反，因为 0 控
制 LED 灯亮
  assign LED_CLK = (shifting) ? ~clk : 1'b0;
  assign LED_EN = 1'b1;

  always @(posedge clk or posedge reset) begin
    if (reset) begin
      shift_reg <= 16'h0000;
      counter <= 4'd0;
```

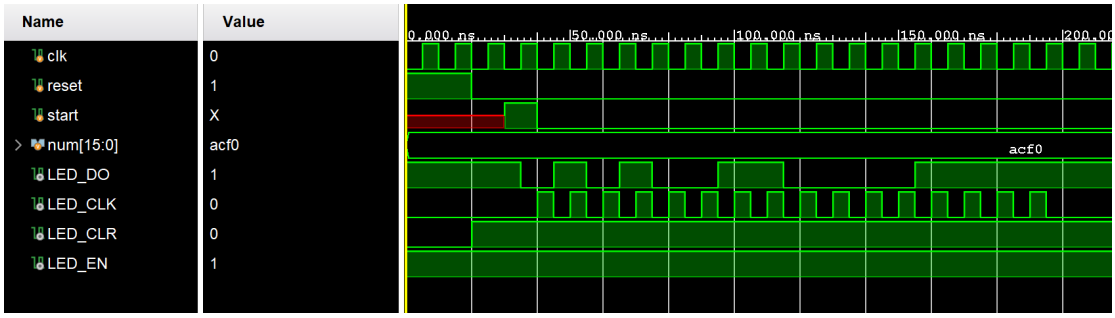
```
      shifting <= 0;
    end
    else begin
      if (start && !shifting) begin
        shift_reg <= num;
        counter <= 4'd0;
        shifting <= 1;
      end
      else if (shifting) begin
        shift_reg <= shift_reg << 1;     // 左移（高位先出）
        counter <= counter + 1;

        if (counter == 15) begin      // 最后一个周期，停止移位
          shifting <= 0;
        end
      end
    end
  end
endmodule
```

② 新建仿真激励文件如下：

```
module LED_driver_tb;

  // Input
  reg clk;
  reg reset;
  reg start;
  reg [15:0] num;

  // Output
  wire LED_DO;
  wire LED_CLK;
  wire LED_CLR;
  wire LED_EN;

  LED_driver uut (
    .clk(clk),
    .reset(reset),
    .start(start),
    .num(num),
    .LED_DO(LED_DO),
    .LED_CLK(LED_CLK),
    .LED_CLR(LED_CLR),
    .LED_EN(LED_EN)
```

```
  );

  // 提供时钟
  always #5 clk = ~clk;

  initial begin
    clk = 0;
    reset = 1;
    num = 16'b1010_1100_1111_0000; // 即将并行发送的测试数据

    #20;
    reset = 0;
    #10;
    start = 1;
    #10;
    start = 0;

    // 等待足够时间观察整个数据移位过程
    #500;
    $stop;
  end
endmodule
```

③ 运行仿真，得到如下的仿真波形。其中，LED_CLK 共计产生 16 个上升沿，每个上升沿处均有 1 位与之对应的 LED_DO 信号；16 个时钟周期过后，LED_CLK 不再产生新的边沿。该波形与预期相符，表明 LED 灯驱动代码设计的正确性。



3、设计主板七段数码管驱动模块

① 编写七段数码管驱动代码如下：

```
module seg_driver(
  input wire clk,
  input wire reset,
  input wire start,
```

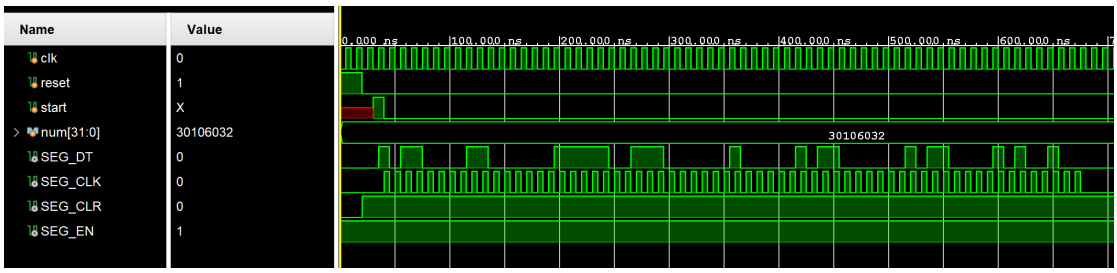
```
  input wire [31:0] num,
  output wire SEG_DT,
  output wire SEG_CLK,
  output wire SEG_CLR,
  output wire SEG_EN
    );

  reg [63:0] shift_reg;
  reg [5:0] counter;
  reg shifting;
  wire [63:0] cnum;

  MyMC14495
m1(.LE(1'b0),.D3(num[31]),.D2(num[30]),.D1(num[29]),.D0(num[28]),.point
(1'b0),.p(cnum[63]),.g(cnum[62]),.f(cnum[61]),.e(cnum[60]),.d(cnum[59])
,.c(cnum[58]),.b(cnum[57]),.a(cnum[56]));
  // 此处省略七个重复的 MyMC14495 模块

  assign SEG_CLR = ~reset;
  assign SEG_DT = shift_reg[63];
  assign SEG_CLK = (shifting) ? ~clk : 1'b0;
  assign SEG_EN = 1'b1;

  always @(posedge clk or posedge reset) begin
    if (reset) begin
      shift_reg <= 64'b0;
      counter <= 5'b0;
      shifting <= 0;
    end
    else begin
      if (start && !shifting) begin
        shift_reg <= cnum;
        counter <= 5'b0;
        shifting <= 1;
      end
      else if (shifting) begin
        shift_reg <= shift_reg << 1;     // 左移（高位先出）
        counter <= counter + 1;
        if (counter == 63) begin      // 最后一个周期
          shifting <= 0;
        end
      end
    end
  end
endmodule
```

② 新建仿真激励文件如下：

```
module seg_driver_tb;

  // Input
  reg clk;
  reg reset;
  reg start;
  reg [31:0] num;

  // Output
  wire SEG_DT;
  wire SEG_CLK;
  wire SEG_CLR;
  wire SEG_EN;

  seg_driver uut (
    .clk(clk),
    .reset(reset),
    .start(start),
    .num(num),
    .SEG_DT(SEG_DT),
    .SEG_CLK(SEG_CLK),
    .SEG_CLR(SEG_CLR),
    .SEG_EN(SEG_EN)
  );

  // 提供时钟
  always #5 clk = ~clk;

  initial begin
    clk = 0;
    reset = 1;
    num = 32'h30106032; // 即将并行发送的测试数据

    #20;
    reset = 0;
    #10;
    start = 1;
    #10;
    start = 0;

    // 等待足够时间观察整个数据移位过程
    #1000;
```

```
    $stop;
  end
endmodule
```

③ 运行仿真，得到如下的仿真波形。其中，SEG_CLK 共计产生 64 个上升沿，每个上升沿处均有 1 位与之对应的 SEG_DT 信号；64 个时钟周期过后，SEG_CLK 不再产生新的边沿。该波形与预期相符，表明七段数码管驱动代码设计的正确性。



4、完成顶层模块与下板验证

① 根据顶层模块的输入输出，完成如下的引脚约束文件。

```
set_property PACKAGE_PIN AC18 [get_ports clk]
set_property IOSTANDARD LVCMOS18 [get_ports clk]

set_property PACKAGE_PIN AB22      [get_ports {SEGMENT[0]}]
set_property IOSTANDARD  LVCMOS33  [get_ports {SEGMENT[0]}]

（省略若干行…）

set_property PACKAGE_PIN AD21      [get_ports {AN[0]}]
set_property IOSTANDARD  LVCMOS33  [get_ports {AN[0]}]

set_property PACKAGE_PIN AC21      [get_ports {AN[1]}]
set_property IOSTANDARD  LVCMOS33  [get_ports {AN[1]}]

set_property PACKAGE_PIN AB21      [get_ports {AN[2]}]
set_property IOSTANDARD  LVCMOS33  [get_ports {AN[2]}]

set_property PACKAGE_PIN AC22      [get_ports {AN[3]}]
set_property IOSTANDARD  LVCMOS33  [get_ports {AN[3]}]

set_property PACKAGE_PIN AA10      [get_ports {SW[0]}]
set_property IOSTANDARD  LVCMOS15  [get_ports {SW[0]}]

（省略若干行…）
```

```
set_property PACKAGE_PIN AF10      [get_ports {SW[15]}]
set_property IOSTANDARD  LVCMOS15  [get_ports {SW[15]}]

set_property PACKAGE_PIN N26      [get_ports {LED_CLK}]
set_property IOSTANDARD  LVCMOS33  [get_ports {LED_CLK}]

set_property PACKAGE_PIN N24      [get_ports {LED_CLR}]
set_property IOSTANDARD  LVCMOS33  [get_ports {LED_CLR}]

set_property PACKAGE_PIN M26      [get_ports {LED_DO}]
set_property IOSTANDARD  LVCMOS33  [get_ports {LED_DO}]

set_property PACKAGE_PIN P18      [get_ports {LED_EN}]
set_property IOSTANDARD  LVCMOS33  [get_ports {LED_EN}]

set_property PACKAGE_PIN M24      [get_ports {SEG_CLK}]
set_property IOSTANDARD  LVCMOS33  [get_ports {SEG_CLK}]

set_property PACKAGE_PIN M20      [get_ports {SEG_CLR}]
set_property IOSTANDARD  LVCMOS33  [get_ports {SEG_CLR}]

set_property PACKAGE_PIN L24      [get_ports {SEG_DT}]
set_property IOSTANDARD  LVCMOS33  [get_ports {SEG_DT}]

set_property PACKAGE_PIN R18      [get_ports {SEG_EN}]
set_property IOSTANDARD  LVCMOS33  [get_ports {SEG_EN}]

set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets BTN_IBUF[0]]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets BTN_IBUF[1]]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets BTN_IBUF[2]]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets BTN_IBUF[3]]

set_property PACKAGE_PIN V18       [get_ports {BTN[0]}]
set_property IOSTANDARD  LVCMOS18  [get_ports {BTN[0]}]

set_property PACKAGE_PIN V19       [get_ports {BTN[1]}]
set_property IOSTANDARD  LVCMOS18  [get_ports {BTN[1]}]

set_property PACKAGE_PIN V14       [get_ports {BTN[2]}]
set_property IOSTANDARD  LVCMOS18  [get_ports {BTN[2]}]

set_property PACKAGE_PIN W14       [get_ports {BTN[3]}]
set_property IOSTANDARD  LVCMOS18  [get_ports {BTN[3]}]
```

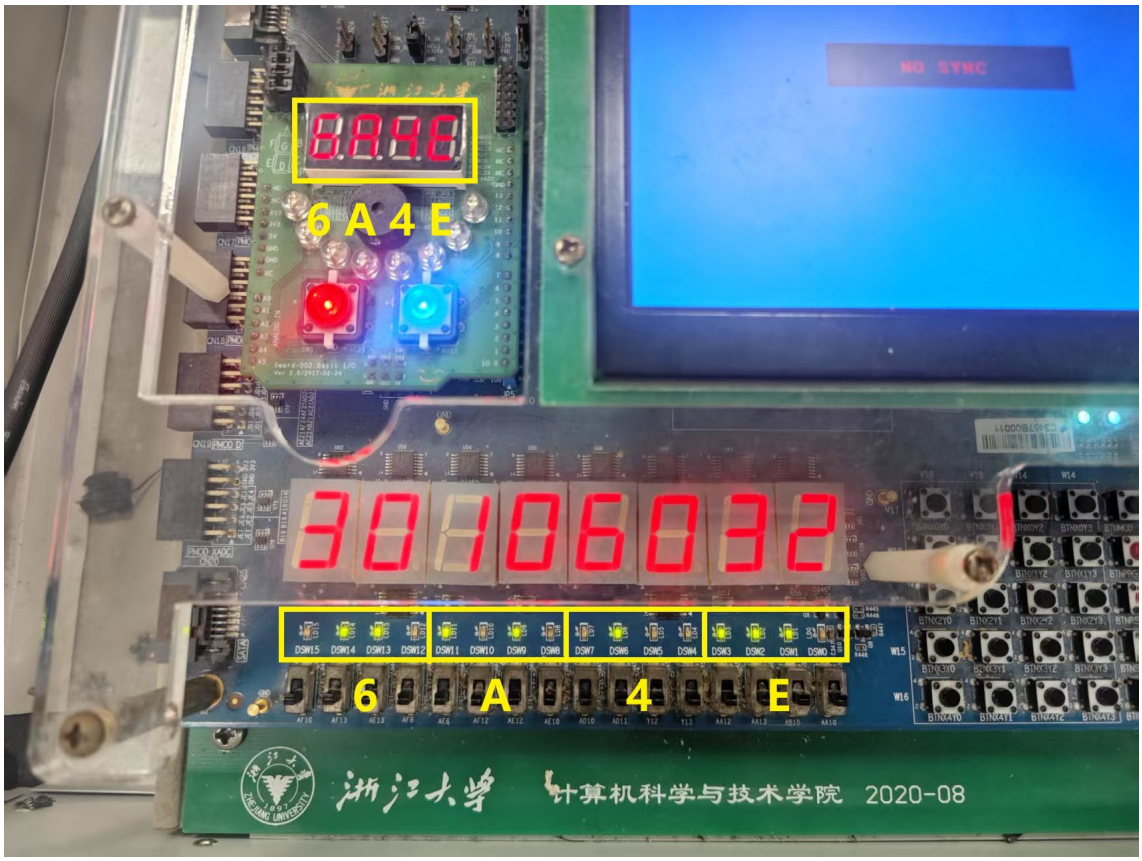
```
set_property PACKAGE_PIN W16       [get_ports BTNX4]
set_property IOSTANDARD  LVCMOS18  [get_ports BTNX4]
```

② 完成顶层模块 top.v

```
module top(
    input wire clk,
    input wire [15:0] SW,
    input wire [3:0] BTN,
    output wire [7:0] SEGMENT,
    output wire [3:0] AN,
    output wire BTNX4,
    output wire LED_DO, LED_CLK, LED_CLR, LED_EN,
    output wire SEG_DT, SEG_CLK, SEG_CLR, SEG_EN
    );

    wire [3:0] btn_out;
    wire [31:0] clk_div;
    wire [15:0] num_LED;
    wire [31:0] num_SEG;
    wire [15:0] SW_out;

    // 为驱动模块输入开始信号
    wire LED_driver_start =
SW[15]|btn_out[0]|btn_out[1]|btn_out[2]|btn_out[3];
    wire seg_driver_start =
SW[15]|SW_out[1]|SW_out[2]|SW_out[3]|SW_out[4]|SW_out[5]|SW_out[6]|SW_o
ut[7]|SW_out[8];

    // 按键消抖模块
    pbdebounce m1(clk_div[17], BTN[0], btn_out[0]);
    pbdebounce m2(clk_div[17], BTN[1], btn_out[1]);
    pbdebounce m3(clk_div[17], BTN[2], btn_out[2]);
    pbdebounce m4(clk_div[17], BTN[3], btn_out[3]);

    // SW 由断到通时，产生一个时钟周期的脉冲
    switch_edge_pulse p1(.clk(clk),.sw(SW[1]),.pulse(SW_out[1]));
    switch_edge_pulse p2(.clk(clk),.sw(SW[2]),.pulse(SW_out[2]));
    switch_edge_pulse p3(.clk(clk),.sw(SW[3]),.pulse(SW_out[3]));
    switch_edge_pulse p4(.clk(clk),.sw(SW[4]),.pulse(SW_out[4]));
    switch_edge_pulse p5(.clk(clk),.sw(SW[5]),.pulse(SW_out[5]));
    switch_edge_pulse p6(.clk(clk),.sw(SW[6]),.pulse(SW_out[6]));
    switch_edge_pulse p7(.clk(clk),.sw(SW[7]),.pulse(SW_out[7]));
    switch_edge_pulse p8(.clk(clk),.sw(SW[8]),.pulse(SW_out[8]));
```

```
    clkdiv u1(.clk(clk),.rst(SW[0]),.clkdiv(clk_div));
    CreateNum u2(.btn(btn_out),.num(num_LED));
    DispNum
u3(.clk(clk),.HEXS(num_LED),.LES(4'b0),.points(4'b0),.RST(SW[0]),.AN(AN
),.Segment(SEGMENT));
    LED_driver
u4(.clk(clk),.reset(SW[0]),.start(LED_driver_start),.num(num_LED),.LED_
DO(LED_DO),.LED_CLK(LED_CLK),.LED_CLR(LED_CLR),.LED_EN(LED_EN));

    CreateNum8Digit u5(.btn(SW_out[8:1]),.num(num_SEG));
    seg_driver
u6(.clk(clk),.reset(SW[0]),.start(seg_driver_start),.num(num_SEG),.SEG_
DT(SEG_DT),.SEG_CLR(SEG_CLR),.SEG_CLK(SEG_CLK),.SEG_EN(SEG_EN));

    assign BTNX4 = 1'b0;

endmodule
```

③ 下板，使用按钮调节小板显示的数字，拨动开关 SW[15]使数字 6A4E 按位显示在 LED 灯上；使用开关 SW[8:1]调节主板八位七段数码管显示的数字，其中 30106032 为本人学号的后 8 位。



五、 讨论、心得

在本次实验中，我尝试实现了 8 位移位寄存器模块，在此基础上编写了 LED 灯和主板数码管的驱动，并成功通过下板验证其正确性。本次实验使我对移位寄存器模块及其实际应用的理解和认识大为加深。此外，通过自己编写顶层模块，我对 Verilog 语言的熟悉程度进一步增强，对分层次、模块化设计的方法更为熟练。