浙江大学

本科实验报告

课程名称：	数字逻辑电路设计
姓　　名：	陈杰伟
学　　院：	地球科学学院
专　　业：	地理信息科学
邮　　箱：	chenjiewei@zju.edu.cn
QQ　　号：	1455044088
电　　话：	13688406524
指导教师：	洪奇军
报告日期：	2022 年　12 月　　26 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：寄存器和寄存器传输设计

学生姓名：陈杰伟 学号：3200101205 同组学生姓名：无

实验地点：紫金港东四 509 室 实验日期：2022 年 12 月 1 日

一、实验目的和要求

1.掌握寄存器传输电路的工作原理
2.掌握寄存器传输电路的设计方法
3.掌握 ALU 和寄存器传输电路的综合应用

二、操作方法与实验步骤

2.1 基于 ALU 的数据传输应用设计

1、新建工程，工程名称用 MyALUTrans。

2、新建 HDL 源文件，命名为 Load_Gen

3、编写 Load_Gen 模块如下，实现按键的缓冲输出正确 Load 信号

```
module Load_Gen(
    input wire clk,
    input wire btn,
    output reg Load
    );

    initial Load = 0;

    reg old_btn;

    always@(posedge clk) begin
        if ((old_btn == 1'b0) && (btn == 1'b1)) //btn出现上升沿
            Load <= 1'b1;
        else
            Load <= 1'b0;
```

```
    end

    always@(posedge clk) begin      //保存上一个周期 btn 的状态
        old_btn <= btn;
    end

endmodule
```

4、新建 HDL 源文件，命名为 Regiser4b

5、编写四位寄存器逻辑如下

```
module Register4b(
    input wire clk,
    input wire [3:0] In,
    input wire Load,
    output wire [3:0] Out
    );

    wire [3:0] In_control;

    assign In_control[0] = (In[0] & Load) | (Out[0] & ~Load);
    assign In_control[1] = (In[1] & Load) | (Out[1] & ~Load);
    assign In_control[2] = (In[2] & Load) | (Out[2] & ~Load);
    assign In_control[3] = (In[3] & Load) | (Out[3] & ~Load);

    FD d1(.D(In_control[0]), .C(clk), .Q(Out[0]));
    FD d2(.D(In_control[1]), .C(clk), .Q(Out[1]));
    FD d3(.D(In_control[2]), .C(clk), .Q(Out[2]));
    FD d4(.D(In_control[3]), .C(clk), .Q(Out[3]));

endmodule
```

6、新建 HDL 源文件，命名为 top

7、编写 top 模块如下

```
module top(
    input wire [15:0] SW,
    input wire clk,
    output wire [15:0] num
    );
    wire [31:0] clk_div;
    wire [3:0] A_IN, B_IN, C_IN;
    wire Load_A, Load_B, Load_C;
    wire [3:0] A_OUT, B_OUT, C_OUT;
    wire [3:0] I1, I0_A, I0_B, I0_C;
    wire Co;
```

```
12.
13.     Load_Gen m0(.clk(clk), .btn(SW[2]), .Load(Load_A));
14.     Load_Gen m1(.clk(clk), .btn(SW[3]), .Load(Load_B));
15.     Load_Gen m2(.clk(clk), .btn(SW[4]), .Load(Load_C));
16.
17.     AddSub4b m4(.A(A_OUT), .B(4'b0001), .CTRL(SW[0]), .S(I0_A));
18.     AddSub4b m5(.A(B_OUT), .B(4'b0001), .CTRL(SW[1]), .S(I0_B));
19.     ALU m10(A_OUT, B_OUT, SW[6:5], I0_C, Co);
20.
21.     assign A_IN = (SW[15]) ? I1 : I0_A;
22.     assign B_IN = (SW[15]) ? I1 : I0_B;
23.     assign C_IN = (SW[15]) ? I1 : I0_C;
24.     Mux4to1b4 m9(4'b0000, A_OUT, B_OUT, C_OUT, SW[8:7], I1);
25.
26.     Register4b RegA(.clk(clk), .In(A_IN), .Load(Load_A), .Out(A_OUT));
27.     Register4b RegB(.clk(clk), .In(B_IN), .Load(Load_B), .Out(B_OUT));
28.     Register4b RegC(.clk(clk), .In(C_IN), .Load(Load_C), .Out(C_OUT));
29.
30.     assign num = {A_OUT, B_OUT, C_OUT, I1};
31.
32. endmodule
```

8、新建 top 模块仿真波形文件，命名为 top_sim

9、编写仿真波形文件如下

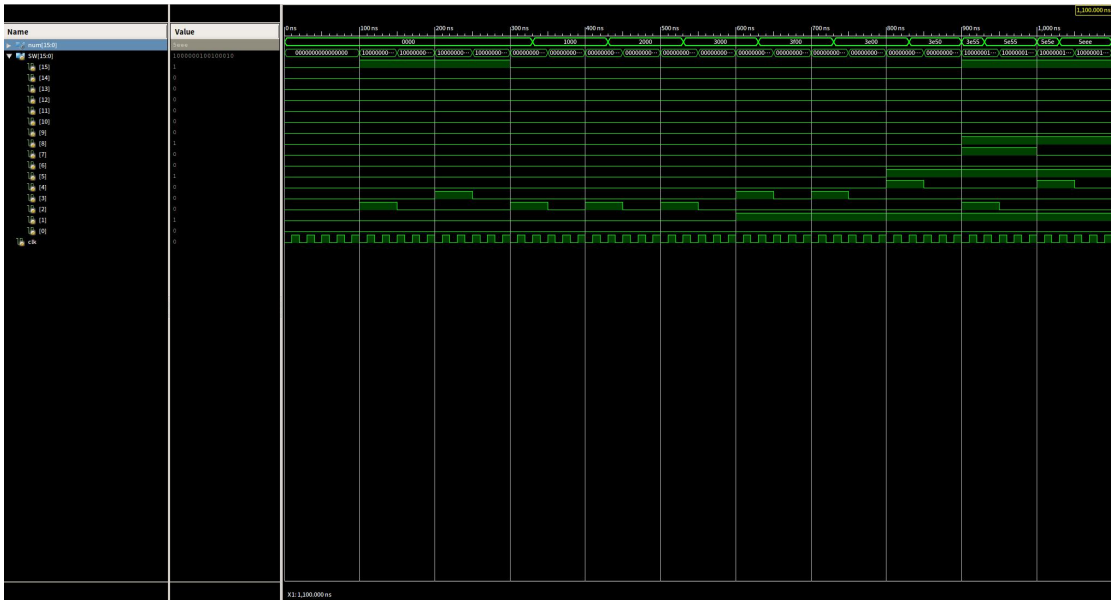
```
1.  module sim_top;
2.   // Inputs
3.      reg [15:0] SW;
4.      reg clk;
5.      // Outputs
6.      wire [15:0] num;
7.      // Instantiate the Unit Under Test (UUT)
8.      top uut (
9.          .SW(SW),
10.         .clk(clk),
11.         .num(num)
12.     );
13.
14.     always #10 clk = ~clk;
15.
16.     initial begin
17.         clk = 0;
18.         SW = 0; #100;
19.         SW[15] = 1;
20.         SW[2] = 1; #50;
```

```
        SW[2] = 0; #50;
        SW[3] = 1; #50;
        SW[3] = 0; #50;
        SW[15] = 0;
        SW[2] = 1; #50;
        SW[2] = 0; #50;
        SW[2] = 1; #50;
        SW[2] = 0; #50;
        SW[2] = 1; #50;
        SW[2] = 0; #50;
        // A = 3
        SW[1] = 1;
        SW[3] = 1; #50;
        SW[3] = 0; #50;
        SW[3] = 1; #50;
        SW[3] = 0; #50;
        // B = -2
        SW[6:5] = 2'b01;
        SW[4] = 1; #50;
        SW[4] = 0; #50;
        // C = 3 - (-2) = 5
        SW[15] = 1;
        SW[8:7] = 2'b11;
        SW[2] = 1; #50;
        SW[2] = 0; #50;
        // C to A
        SW[8:7] = 2'b10;
        SW[4] = 1; #50;
        SW[4] = 0; #50;
        // B to C
        #100;
    end

endmodule
```

10、仿真验证 top 模块

三、实验结果与分析

对 top 模块的仿真波形如下



观察仿真波形
在整个仿真过程中，sw[4:2]正确处理了寄存器 ABC 的赋值，当开关被置一时 Load_Gen 模块传出相应的 Load 信号使寄存器赋予相应的值
sw[1:0]正确控制了寄存器值随时钟信号的加减
sw[6:5]正确控制 ALU 的计算方式
sw[8:7]正确控制数据选择器的最终输出（A，B，C 和 ALU 输出值）
ALU 正确进行了加减运算
仿真波形验证了 Load_Gen 和 Register4b 模块功能正确
ALU 寄存器传输计算设计正确

四、讨论、心得

本实验设计了四位寄存器，利用四位寄存器设计了 ALU 的传输应用
我理解了寄存器的工作原理及其功能，可以熟练运用寄存器做电路设计
本实验完全运用 VerilogHDL 设计，使我初步掌握了使用 HDL 语言进行模块化电路设计和行为描述法电路设计的方法。

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：计数器、定时器设计与应用

学生姓名：陈杰伟 学号：3200101205 同组学生姓名：无

实验地点：紫金港东四 509 室 实验日期：2022 年 12 月 8 日

一、实验目的和要求

1.掌握同步四位二进制计数器 74LS161 的工作原理和设计方法

2.掌握时钟/定时器的工作原理与设计方法

二、操作方法与实验步骤

2.1 设计同步四位二进制计数器 74LS161

1、新建工程，工程名为 Clock

2、新建 HDL 源文件，文件名为 my74LS161

3、使用行为描述方法设计模块如下

```
module My74LS161(
    input wire C_R,
    input wire C_P,
    input wire L_D,
    input wire CT_P,
    input wire CT_T,
    input wire [3:0] D,
    output wire [3:0] Q,
    output wire C_O
    );

    reg [3:0] new_q;
    reg [3:0] old_q;
    reg rst;
    reg temp_co;

    wire [3:0] reset_ctrl;
```

```
    wire [3:0] load_ctrl;
    wire [3:0] count_ctrl;
    wire [3:0] keep_ctrl;
    wire change_ctrl;

    assign reset_ctrl = {4{C_R}};
    assign load_ctrl = {4{C_R & ~L_D}};
    assign change_ctrl = C_R & L_D;
    assign keep_ctrl = {4{change_ctrl & ((~CT_P & CT_T) | ~CT_T)}};
    assign count_ctrl = {4{change_ctrl & CT_P & CT_T}};

    always @(posedge C_P or negedge C_R)  begin
        if(C_R == 1'b0)begin
            new_q <= 4'b0;
            old_q <= 4'b0;
            temp_co <= 1'b0;
        end
        else if(Q<4'b1111)begin
            new_q <= ((Q + 4'b0001 )& count_ctrl) | (D & load_ctrl);
            temp_co <= 1'b0;
            old_q <= Q & reset_ctrl;
        end
        else begin
            new_q <= (4'b0000 & count_ctrl) | (D & load_ctrl);
            temp_co <= count_ctrl[0] & 1'b1 & reset_ctrl[0];
            old_q <= Q & reset_ctrl;
        end
    end

    assign Q = ((old_q & keep_ctrl)| (new_q & ~keep_ctrl)) & reset_ctrl;

    assign C_O = temp_co;

endmodule
```

4、为 my74LS161 编写波形仿真文件如下

```
module My74LS161_Sim;

// Inputs
reg C_R;
reg C_P;
reg L_D;
reg CT_P;
reg CT_T;
reg [3:0] D;
```

```
// Outputs
wire [3:0] Q;
wire C_O;
// Instantiate the Unit Under Test (UUT)
My74LS161 uut (
.C_R(C_R),
.C_P(C_P),
.L_D(L_D),
.CT_P(CT_P),
.CT_T(CT_T),
.D(D),
.Q(Q),
.C_O(C_O)
);
initial begin
C_R = 0;
C_P = 0;
L_D = 0;
CT_P = 0;
CT_T = 0;
D = 0; #100;
C_R = 1;
L_D = 1;
D = 4'b1100;
CT_T = 0;
CT_P = 0; #30;
C_R = 0; #20;
C_R = 1; #10;
L_D = 0; #30;
CT_T = 1;
CT_P = 1; #10;
L_D = 1; #510;
C_R = 0; #20;
C_R = 1; #500;
end
always #20 C_P = ~C_P;

endmodule
endmodule
```

5、波形仿真验证 my74LS161

6、新建 HDL 文件，命名为 TOP

7、编写 TOP 文件如下

```
module Top(
input wire clk,
```

```
input wire rst,
output wire [23:0] num
);

    assign num0 = num[3] & num[0];
    assign num1 = num[6] & num[4] & num0;
    assign num2 = num[11] & num[8] & num1;
    assign num3 = num[14] & num[12] & num2;
    assign num4 = (num[19] & num[16] & num3) | num5;
    assign num5 = num[21] & num[17] & num[16] & num3;
    My74LS161 m0(.C_R(rst), .C_P(clk), .L_D(~num0), .CT_P(1'b1), .CT_T(1'b1), .D(4'b0000), .Q(num[3:0]));
    My74LS161 m1(.C_R(rst), .C_P(clk), .L_D(~num1), .CT_P(1'b1), .CT_T(num0), .D(4'b0000), .Q(num[7:4]));
    My74LS161 m2(.C_R(rst), .C_P(clk), .L_D(~num2), .CT_P(1'b1), .CT_T(num1), .D(4'b0000), .Q(num[11:8]));
    My74LS161 m3(.C_R(rst), .C_P(clk), .L_D(~num3), .CT_P(1'b1), .CT_T(num2), .D(4'b0000), .Q(num[15:12]));
    My74LS161 m4(.C_R(rst), .C_P(clk), .L_D(~num4), .CT_P(1'b1), .CT_T(num3), .D(4'b0000), .Q(num[19:16]));
    My74LS161 m5(.C_R(rst), .C_P(clk), .L_D(~num5), .CT_P(1'b1), .CT_T(num4), .D(4'b0000), .Q(num[23:20]));

endmodule
```

8、为 TOP 文件编写波形仿真文件如下

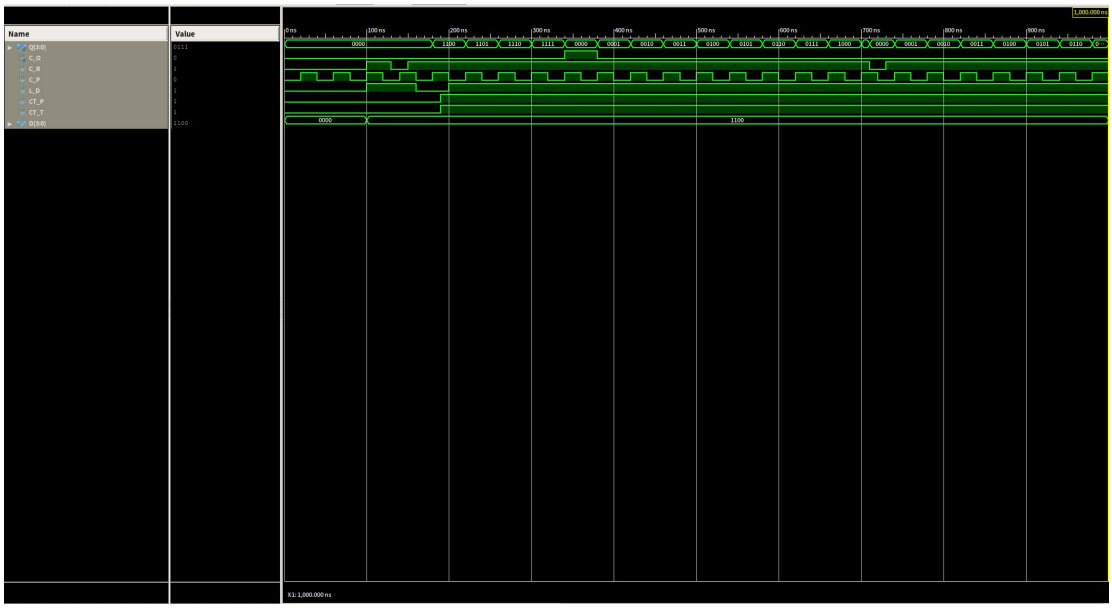
```
module Top_Sim;

// Inputs
reg clk;
reg rst;
// Outputs
wire [23:0] num;
// Instantiate the Unit Under Test (UUT)
Top uut (
.clk(clk),
.rst(rst),
.num(num)
);
always #10 clk = ~clk;
initial begin
clk = 0;
rst = 1; #10; rst = 0; #10; rst = 1;
#864010;
end
```

```
20. endmodule
```

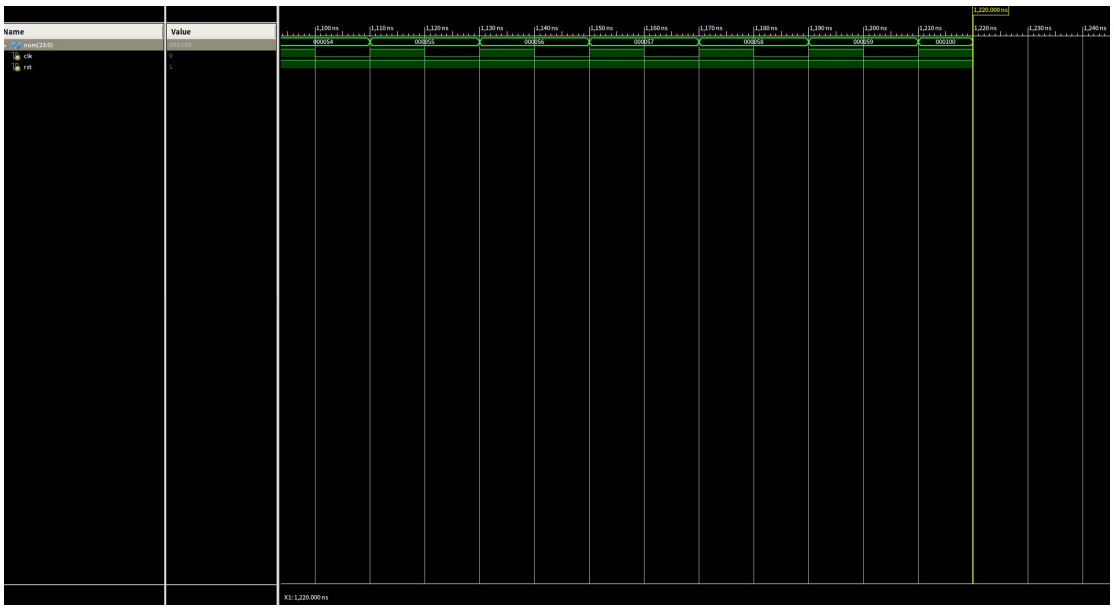
9、波形验证 top 模块的功能

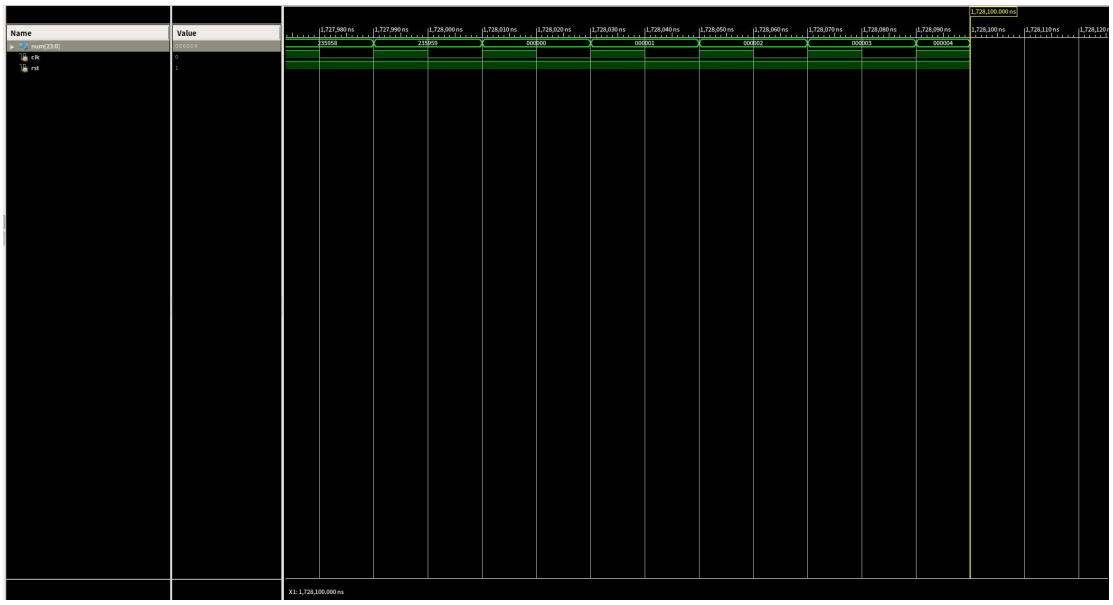
三、实验结果与分析

对 my74LS161 的波形仿真如下



分析波形仿真图
可以看到该模块能够正确进行按时钟计数和 rst 功能
也能够被赋予 D[3:0]的值
波形仿真结果符合真值表
对 TOP 模块仿真结果如下





分析波形仿真图

top 模块能够正确行使 24 小时时钟功能，每一个时钟周期计时一秒钟，在整分钟、整小时和整天能够正确进行进位和清零

top 模块设计正确，同时也验证了 my74LS161 模块设计正确

四、讨论、心得

本次实验设计了 my74LS161 二进制计数模块和一个 24 小时时钟，进一步加深了我对时钟和时序电路的理解

本次实验利用行为描述的方法设计时序电路，进一步增强了我的时序电路设计能力

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：陈杰伟 学号：3200101205 同组学生姓名：无

实验地点：紫金港东四 509 室 实验日期：2022 年 12 月 20 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、操作方法与实验步骤

2.1 设计 8 位带并行输入的右移移位寄存器

1、新建工程，命名为 ShfitReg8b
2、新建 HDL 文件，命名为 shift_reg
3、模块化设计八位移位寄存器如下

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

    wire [7:0] d_in;

    assign d_in[0] = (Q[1] & ~S_L) | (p_in[0] & S_L);
    assign d_in[1] = (Q[2] & ~S_L) | (p_in[1] & S_L);
    assign d_in[2] = (Q[3] & ~S_L) | (p_in[2] & S_L);
    assign d_in[3] = (Q[4] & ~S_L) | (p_in[3] & S_L);
    assign d_in[4] = (Q[5] & ~S_L) | (p_in[4] & S_L);
    assign d_in[5] = (Q[6] & ~S_L) | (p_in[5] & S_L);
    assign d_in[6] = (Q[7] & ~S_L) | (p_in[6] & S_L);
    assign d_in[7] = (s_in & ~S_L) | (p_in[7] & S_L);

    FD d0(.D(d_in[0]), .C(clk), .Q(Q[0]));
```

```
    FD d1(.D(d_in[1]), .C(clk), .Q(Q[1]));
    FD d2(.D(d_in[2]), .C(clk), .Q(Q[2]));
    FD d3(.D(d_in[3]), .C(clk), .Q(Q[3]));
    FD d4(.D(d_in[4]), .C(clk), .Q(Q[4]));
    FD d5(.D(d_in[5]), .C(clk), .Q(Q[5]));
    FD d6(.D(d_in[6]), .C(clk), .Q(Q[6]));
    FD d7(.D(d_in[7]), .C(clk), .Q(Q[7]));

endmodule
```

4、为 shift_reg 设计仿真文件如下

```
module reg_sim;

    // Inputs
    reg clk;
    reg S_L;
    reg s_in;
    reg [7:0] p_in;

    // Outputs
    wire [7:0] Q;

    // Instantiate the Unit Under Test (UUT)
    shift_reg uut (
        .clk(clk),
        .S_L(S_L),
        .s_in(s_in),
        .p_in(p_in),
        .Q(Q)
    );

    initial begin
        // Initialize Inputs
        clk = 0;
        S_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


endmodule
```

5、对 shift_reg 进行波形仿真验证功能
6、新建文件命名为 shift_recursive_reg
7、设计可以循环移位的八位移位寄存器如下

```
module shift_recursive_reg(
    input wire clk, S_L, s_in, rc,
    input wire [7:0] p_in,
    output wire [7:0] Q
    );

    wire [7:0] d_in;

    assign d_in[0] = (Q[1] & ~S_L) | (p_in[0] & S_L);
    assign d_in[1] = (Q[2] & ~S_L) | (p_in[1] & S_L);
    assign d_in[2] = (Q[3] & ~S_L) | (p_in[2] & S_L);
    assign d_in[3] = (Q[4] & ~S_L) | (p_in[3] & S_L);
    assign d_in[4] = (Q[5] & ~S_L) | (p_in[4] & S_L);
    assign d_in[5] = (Q[6] & ~S_L) | (p_in[5] & S_L);
    assign d_in[6] = (Q[7] & ~S_L) | (p_in[6] & S_L);
    assign d_in[7] = ((Q[0] & rc | s_in & ~rc) & ~S_L) | (p_in[7] & S_L);

    FD d0(.D(d_in[0]), .C(clk), .Q(Q[0]));
    FD d1(.D(d_in[1]), .C(clk), .Q(Q[1]));
    FD d2(.D(d_in[2]), .C(clk), .Q(Q[2]));
    FD d3(.D(d_in[3]), .C(clk), .Q(Q[3]));
    FD d4(.D(d_in[4]), .C(clk), .Q(Q[4]));
    FD d5(.D(d_in[5]), .C(clk), .Q(Q[5]));
    FD d6(.D(d_in[6]), .C(clk), .Q(Q[6]));
    FD d7(.D(d_in[7]), .C(clk), .Q(Q[7]));

endmodule
```

8、新建 HDL 文件命名为 TOP

9、设计 top 跑马灯模块如下

```
module top(
    input wire clk,
    input wire [4:0] sw,
    output wire [7:0] LED
    );

    wire [7:0] num1;
    wire [3:0] regA;
    wire [3:0] regB;

    CreateNumber num(.btn({2'b0,sw[0],sw[1]}),.num({num1,regA,regB}));

     shift_recursive_reg sr(.clk(clk),.S_L(sw[2]),.s_in(sw[3]),.p_in({regA,regB}),.Q(LED),.rc(sw[4]));

endmodule
```

10、为 top 编写仿真文件如下

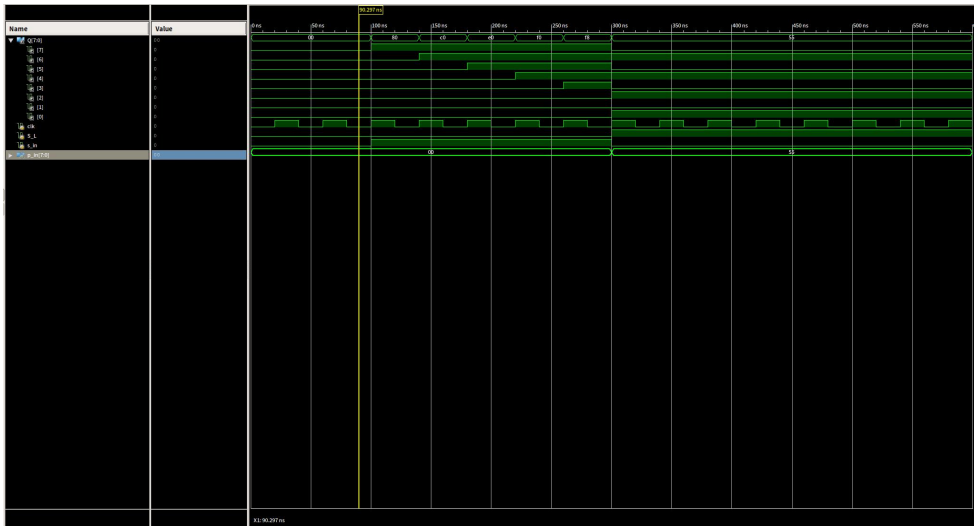
```
module top_sim;

    // Inputs
    reg clk;
    reg [4:0] sw;

    // Outputs
    wire [7:0] LED;

    // Instantiate the Unit Under Test (UUT)
    top uut (
        .clk(clk),
        .sw(sw),
        .LED(LED)
    );

    initial begin
        sw[2] = 1;
        #20;
        sw[0] = 1;
        sw[1] = 1;
        sw[2] = 0;
        sw[3] = 1;
```

```
        sw[4] = 0;
        #10;
        sw[0] = 0;
        sw[1] = 0;
        #10;
        sw[1] = 1;
        #10;
        sw[1] = 0;
        #200;
        sw[3] = 0;
        #80;
        sw[4] = 1;
        #200;
        sw[2] = 1;

    end

    always begin
        clk = 0; #10;
        clk = 1; #10;
    end

endmodule
```

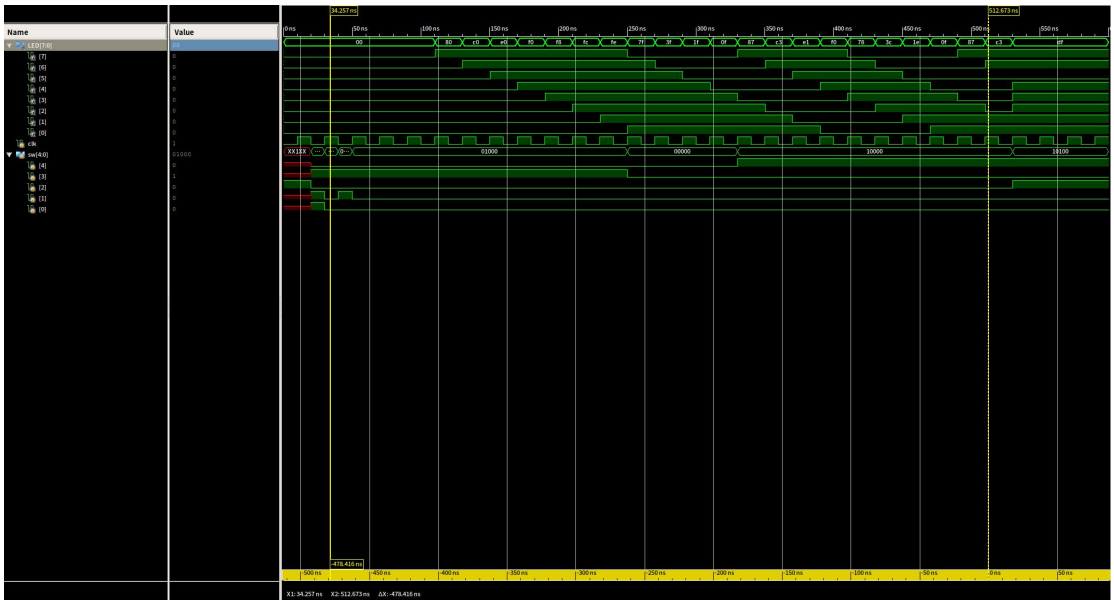
11、仿真验证跑马灯功能实现

三、实验结果与分析

对 shift_reg 波形仿真结果如下



分析波形仿真结果
移位寄存器能够正确按照信号进行移位串行赋值和按照 p_in 进行并行赋值
模块设计正确
对 TOP 模块仿真结果如下



分析仿真
sw[1:0]控制 regA 和 regB 的值，在前若干个周期中分别被赋值为 1，2
sw[2]控制并行输入，最开始为寄存器赋值为 0，最后为寄存器赋值为 1，2
sw[3]控制串行赋值的值
sw[4]控制循环赋值和按照串行信号赋值
可以看到 top 模块能正确进行循环赋值和串行赋值，以及并行读入寄存器值
具体体现为 LED 信号，即循环点亮的跑马灯
模块设计正确

四、讨论、心得

本实验设计了八位移位寄存器和跑马灯模块，进一步强化了我设计时序电路的能力，使我理解了移位寄存器的设计和功能
本实验使用模块化设计的方式设计电路，强化了我的电路理解和 Verilog 设计能力

五、个人生活照