浙江大学

本科实验报告

课程名称:	数字逻辑电路设计
姓　　名:	陈诺
学　　院:	计算机科学与技术学院
专　　业:	计算机
邮　　箱:	chennuo731@zju.edu.cn
QQ　　号:	2528064826
电　　话:	15358508348
指导教师:	洪奇军
报告日期:	2022 年 12 月 27 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：　陈诺　学号：　3210102020　同组学生姓名：

实验地点：　线上　实验日期：　2022　年　12　月　14　日

一、操作方法与实验步骤

1.1 采用寄存器传输原理设计计数器

1. 新建工程文件，命名为 MyRegCounter，Top Level Source Type 为 HDL
2. 新建类型为 verilog 的源文件，命名为 MyRegister4b，用 verilog 代码设计，具体如下所示：

```
module MyRegister4b(
   input wire clk,
   input [3:0] IN,
   input wire Load,
   output reg [3:0] OUT
   );
   always @ (posedge clk) begin
    if (Load) OUT <= IN;
   end
endmodule
```

3. 新建类型为 verilog 的源文件，命名为 Load_Gen，用 verilog 代码设计，具体如下所示：

```
module Load_Gen(
   input wire clk,
   input wire clk_1ms,
   input wire btn_in,
   output reg Load_out
   );
   initial Load_out = 0;
   wire btn_out;
   reg old_btn;
   pbdebounce p0(clk_1ms, btn_in, btn_out);//
   always@(posedge clk) begin
```

```
    if ((old_btn == 1'b0) && (btn_out == 1'b1)) //btn 出现上升沿
     Load_out <= 1'b1;
    else
     Load_out <= 1'b0;
  end
  always@(posedge clk) begin //保存上一个周期 btn 的状态
   old_btn <= btn_out;
  end
endmodule
```

4. 新建类型为 verilog module 的源文件，命名为 top，并右键 Set as Top Module
5. 用 verilog 代码设计，具体如下所示：

```
module top(
    input wire clk,
    input wire [15:0] SW,
  output wire [15:0] num
    );
    wire [3:0] Load_A, Co;
    wire [3:0] A, A_IN, A1;
    wire [31:0] clk_div;

  MyRegister4b RegA(.clk(clk), .IN(A_IN), .Load(Load_A), .OUT(A));
  Load_Gen m0(.clk(clk), .btn_in(SW[2]),.Load_out(Load_A)); //寄存器 A 的 Load 信号
    clkdiv_Number m3(clk, 1'b0, clk_div);
    myAddSub4b m4(.A(A), .B(4'b0001), .Ctrl(SW[0]), .S(A1), .Co(Co)); //自增/自减逻辑
    assign A_IN = (SW[15] == 1'b0)? A1: 4'b0000; //2 选 1 多路复用器，复位寄存器初值
    assign num = {A, A1, A_IN, 4'b0000};

endmodule
```

6. 对 top 进行仿真，激励代码如下：

```
always begin
  #10;
  clk = ~clk;
 end
 initial begin
  // Initialize Inputs
      SW = 0;
  clk = 0;
```

```
9.
10.   // Wait 100 ns for global reset to finish
11.   #100;
12.
13.   // Add stimulus here
14.   SW[15] = 1;
15.   SW[2] = 1;#50;
16.   SW[2] = 0;#50;
17.
18.   SW[15] = 0;
19.   SW[0] = 0;
20.   SW[2] = 1;#50;
21.   SW[2] = 0;#50;
22.
23.   SW[2] = 1;#50;
24.   SW[2] = 0;#50;
25.
26.   SW[0] = 1;
27.   SW[2] = 1;#50;
28.   SW[2] = 0;#50;
29.
30.   SW[2] = 1;#50;
31.   SW[2] = 0;#50;
32.
33.   SW[2] = 1;#50;
34.   SW[2] = 0;#50;
35. end
```

1.2 基于多路选择器总线的寄存器传输

1. 新建工程文件，命名为 RegDataPathTrans，Top Level Source Type 为 HDL
2. 新建类型为 verilog 的源文件，命名为 top，并右键 Set as Top Module
3. 用 verilog 代码设计，具体如下所示：

```
1. module top(
2.   input clk,
3.     input [15:0] SW
4.     );
5.     wire [31:0] clk_div;
6.     wire [3:0] A_IN, B_IN, C_IN;
7.     wire [3:0] Load_A, Load_B, Load_C;
8.     wire [3:0] A_OUT, B_OUT, C_OUT;
9.     wire [3:0] I1, I0_A, I1, I0_B, I0_C;
10.    wire [3:0] C;
11.    wire Co;
```

```
12.
13. Load_Gen m0(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(SW[2]),.Lo
    ad_out(Load_A));
14. Load_Gen m1(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(SW[3]),.Lo
    ad_out(Load_B));
15. Load_Gen m2(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(SW[4]),.Lo
    ad_out(Load_C));
16. clkdiv_Number m3(clk, 1'b0, clk_div);
17.
18. myAddSub4b m4(.A(A_OUT), .B(4'b0001), .Ctrl(SW[0]), .S(I0_A));
19. myAddSub4b m5(.A(B_OUT), .B(4'b0001), .Ctrl(SW[1]), .S(I0_B));
20.
21. assign A_IN = (SW[15] == 1'b0)? I0_A: I1;
22.   assign B_IN = (SW[15] == 1'b0)? I0_B: I1;
23.   assign C_IN = (SW[15] == 1'b0)? 1'b0: I1;
24.
25. Mux4to1b4 m9(A_OUT, B_OUT, C_OUT, 4'b0000, SW[8:7], I1);
26.
27. MyRegister4b RegA( clk, A_IN, Load_A, A_OUT );
28. MyRegister4b RegB( clk, B_IN, Load_B, B_OUT );
29. MyRegister4b RegC( clk, C_IN, Load_C, C_OUT );
30.
31.endmodule
```

1.3 基于 ALU 的数据传输应用设计

1. 新建工程文件，命名为 MyALUTrans，Top Level Source Type 为 HDL
2. 新建类型为 verilog 的源文件，命名为 top，并右键 Set as Top Module
3. 用 verilog 代码设计，具体如下所示：

```
1. module top(
2.     input clk,
3.     input [15:0] SW,
4.   output [15:0] num
5.     );
6.   wire [3:0] A_IN, B_IN, C_IN;
7.   wire Load_A, Load_B, Load_C;
8.   wire [3:0] A_OUT, B_OUT, C_OUT;
9.   wire [3:0] I1, I0_A, I0_B, I0_C;
10. wire Co;
11.
12.  Load_Gen m0(.clk(clk), .btn_in(SW[2]),.Load_out(Load_A));
13.  Load_Gen m1(.clk(clk), .btn_in(SW[3]),.Load_out(Load_B));
14.  Load_Gen m2(.clk(clk), .btn_in(SW[4]),.Load_out(Load_C));
15.
```

```
  myAddSub4b m4(.A(A_OUT), .B(4'b0001), .Ctrl(SW[0]), .S(I0_A), .Co(Co));
  myAddSub4b m5(.A(B_OUT), .B(4'b0001), .Ctrl(SW[1]), .S(I0_B), .Co(Co));

  //Mux2to1b m6( I0_A, I1, SW[15], A_IN);
  //Mux2to1b m7( I0_B, I1, SW[15], B_IN);
  //Mux2to1b m8( I0_C, I1, SW[15], C_IN);
  Mux4to1b4 m9(A_OUT, B_OUT, C_OUT, 4'b0000, SW[8:7], I1);

  MyRegister4b RegA( clk, A_IN, Load_A, A_OUT );
  MyRegister4b RegB( clk, B_IN, Load_B, B_OUT );
  MyRegister4b RegC( clk, C_IN, Load_C, C_OUT );

  myALU m10( A_OUT, B_OUT, SW[6:5], I0_C, Co);

  assign A_IN = (SW[15] == 1'b0)? I0_A: I1;
  assign B_IN = (SW[15] == 1'b0)? I0_B: I1;
  assign C_IN = (SW[15] == 1'b0)? I0_C: I1;
  assign num = {A_OUT, B_OUT, C_OUT, I1};
endmodule
```

4. 对 top 进行仿真，激励代码如下：

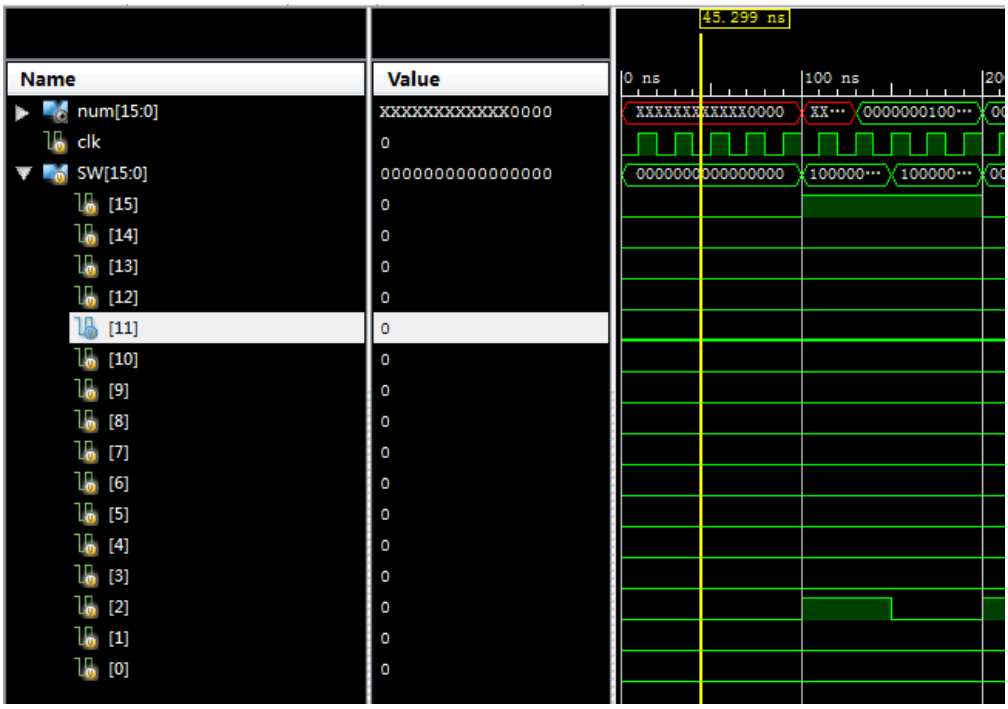
```
always begin
   #10;
   clk = ~clk;
  end

 initial begin
   // Initialize Inputs
   clk = 0;
   SW = 0;

   #100;
   // Wait 100 ns for global reset to finish

   // Add stimulus here
   SW[15] = 1;
   SW[7] = 1;
   SW[8] = 1;
   SW[2] = 1; #50;
   SW[2] = 0; #50;

   SW[3] = 1; #50;
```

```
22.   SW[3] = 0; #50;
23.
24.   SW[15] = 0;
25.   SW[2] = 1; #50;
26.   SW[2] = 0; #50;
27.   SW[2] = 1; #50;
28.   SW[2] = 0; #50;
29.   SW[2] = 1; #50;
30.   SW[2] = 0; #50;
31.
32.   SW[1] = 1;
33.   SW[3] = 1; #50;
34.   SW[3] = 0; #50;
35.   SW[3] = 1; #50;
36.   SW[3] = 0; #50;
37.
38.   SW[6] = 0;
39.   SW[5] = 1;
40.   SW[4] = 1; #50;
41.   SW[4] = 0; #50;
42.
43.   SW[15] = 1;
44.   SW[7] = 0;
45.   SW[8] = 1;
46.   SW[2] = 1; #20
47.   SW[2] = 0; #20;
48.
49.   SW[7] = 1;
50.   SW[8] = 0;
51.   SW[4] = 1; #20
52.   SW[4] = 0; #20;
53. end
```

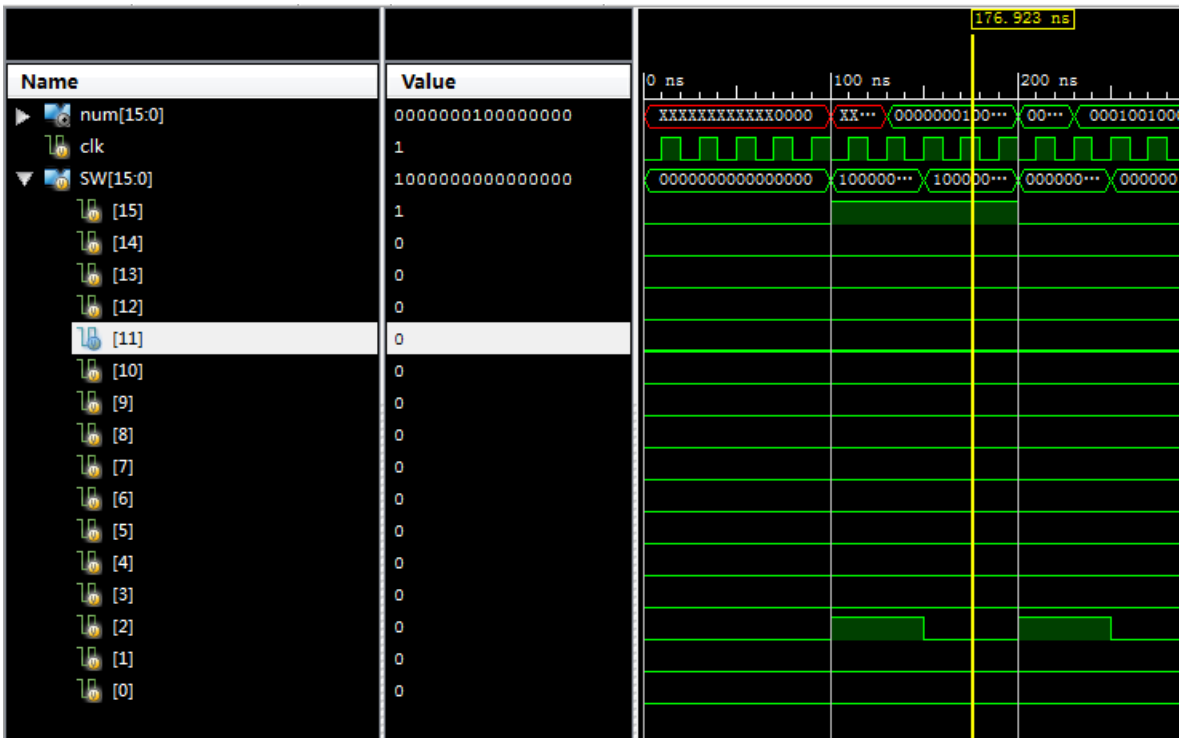
二、实验结果与分析

2.1 采用寄存器传输原理设计计数器

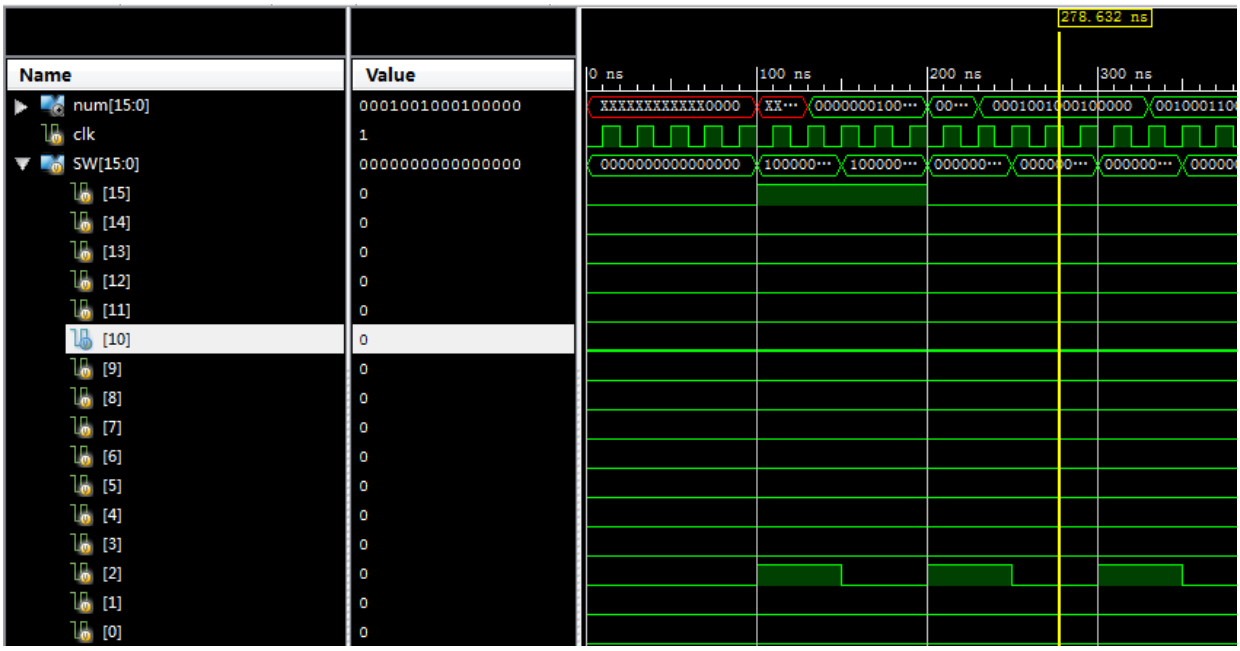
仿真波形如下:
初始状态



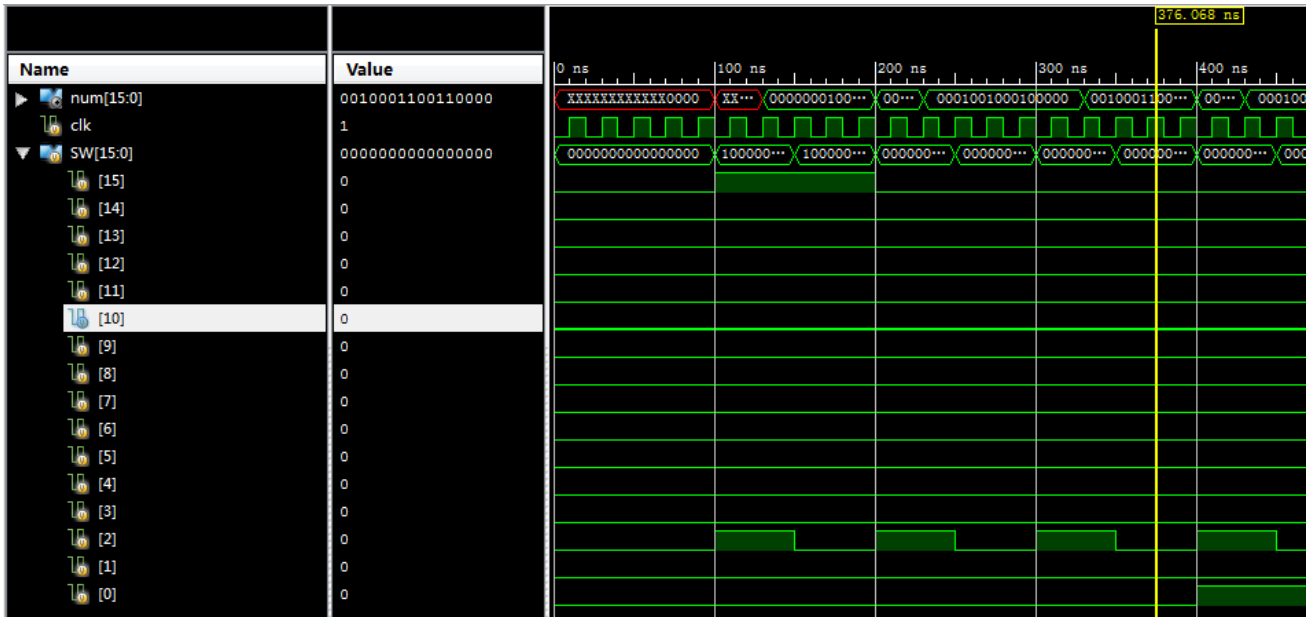
SW[15]拨到 1，SW[2]上下一次，寄存器 A 清零



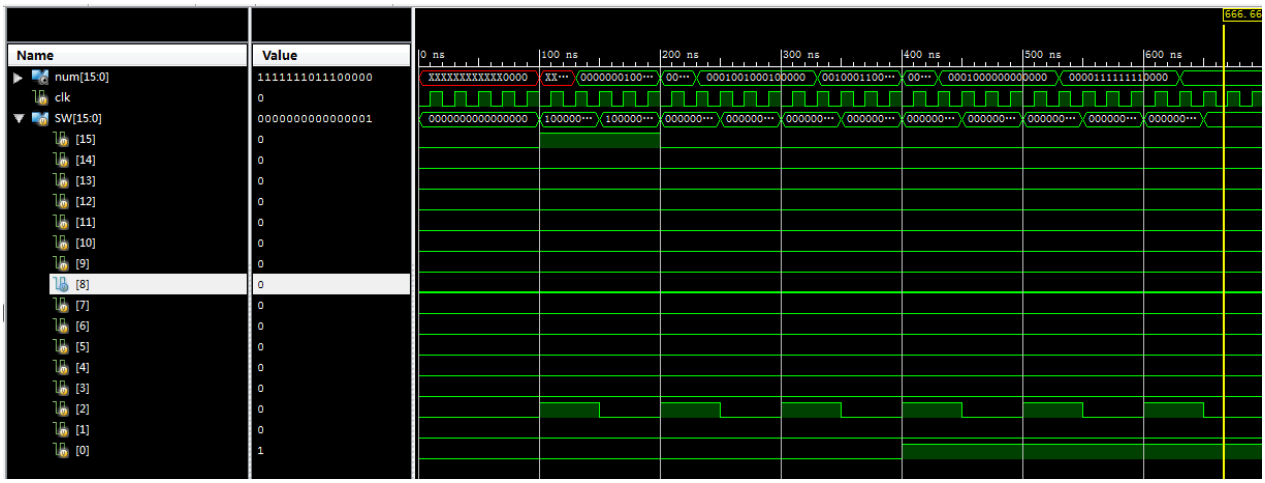
SW[15]拨到 0，SW[0]拨到 0，SW[2]上下拨动 1 次，寄存器 A 加 1



SW[2]再拨动 1 次，A 寄存器再加 1



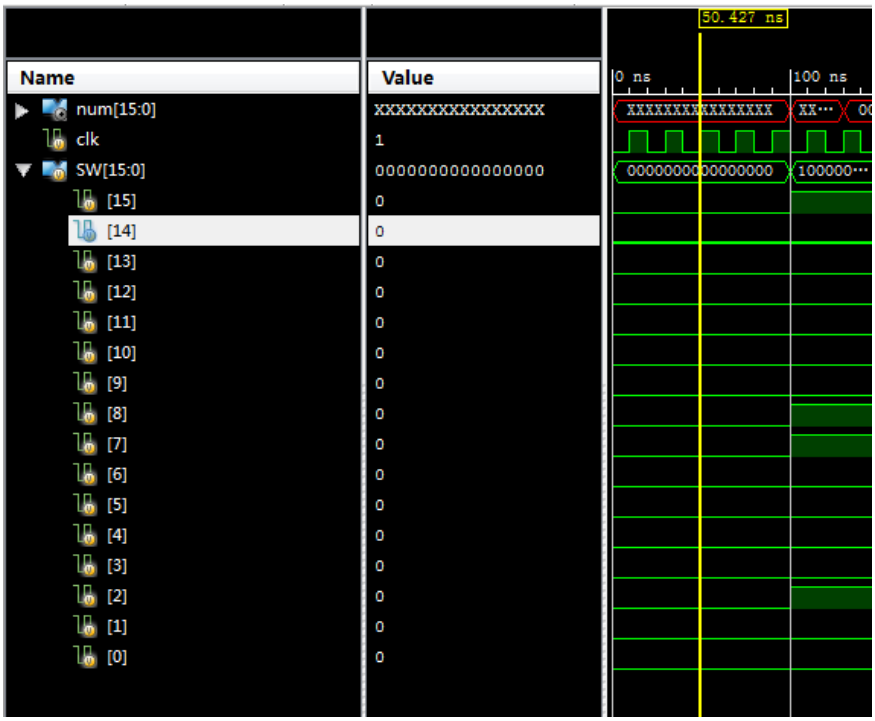
SW[0]拨到 1，SW[2]拨动 3 次，A 寄存器累积减 3



观察仿真波形里 num 总线输出结果，A、A1、A_IN 的变化过程符合预期

2.2 基于 ALU 的数据传输应用设计

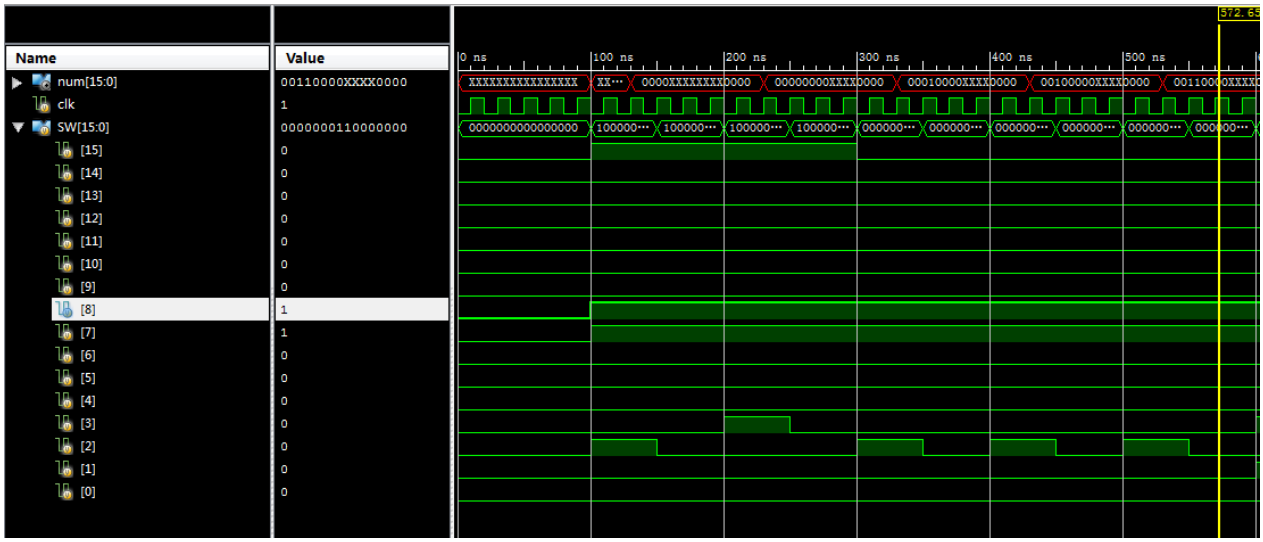
仿真波形如下:
初始状态



SW[15]置为 1，此时由于 SW 均初始置为 0，4 选 1 多路复用器对应输出 out 为 0，上下拨动一次 SW[2]，寄存器 A 读入相应值 I1，寄存器 A 初始化为 0，总输出由 XXX0 变为 0XX0

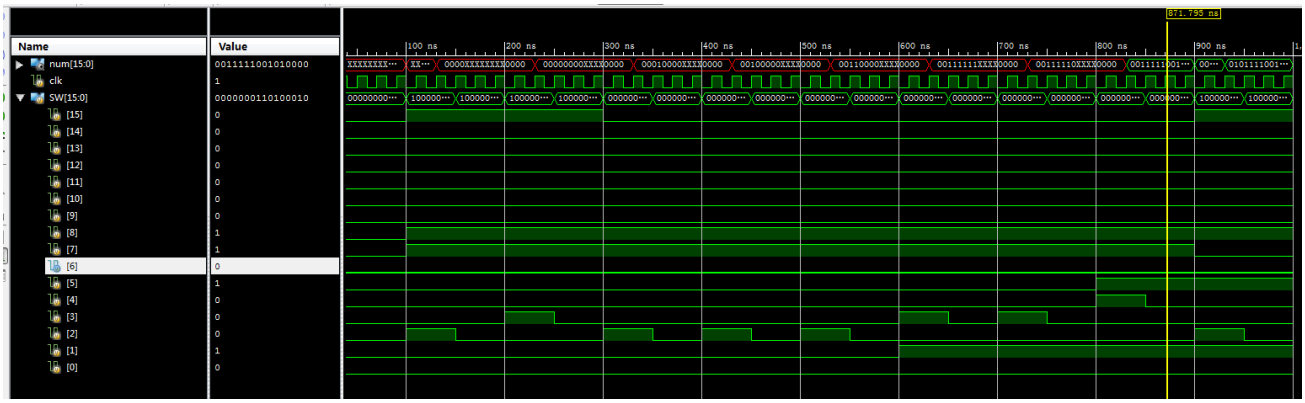
类似上面，上下拨动一次 SW[3]，寄存器 B 初始化为 0，输出由 0XX0 变为 00X0

SW[15]置为 0，SW[0]置为 0，寄存器 A 准备读入 I0（自增器的输出值），即 A 自加，上下拨动 SW[2]三次，A 加 3，输出结果为 30X0

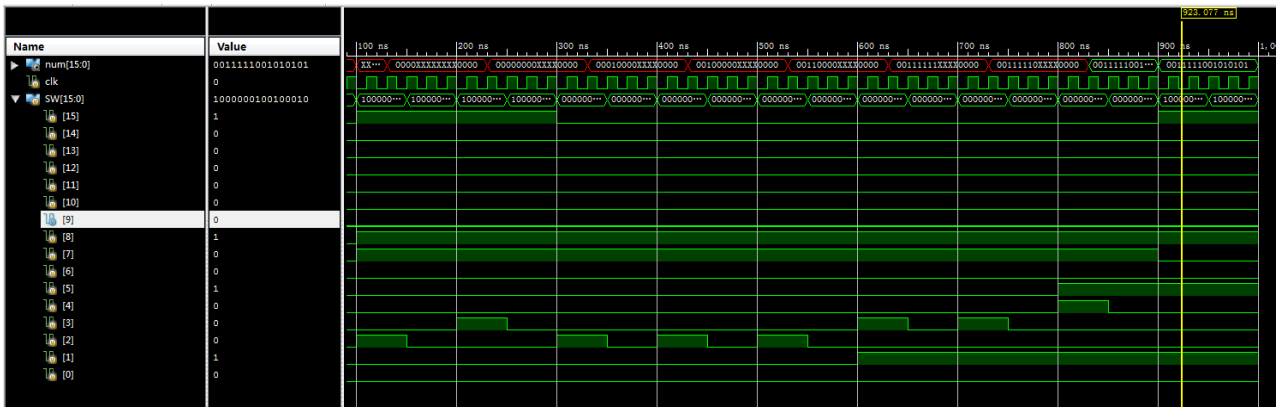


SW[15]保持为 0 不变，SW[1]置为 1，寄存器 B 准备读入 I0（自减器的输出值），即 B 自减，上下拨动 SW[3]两次，B 减 2，输出结果为 3eX0

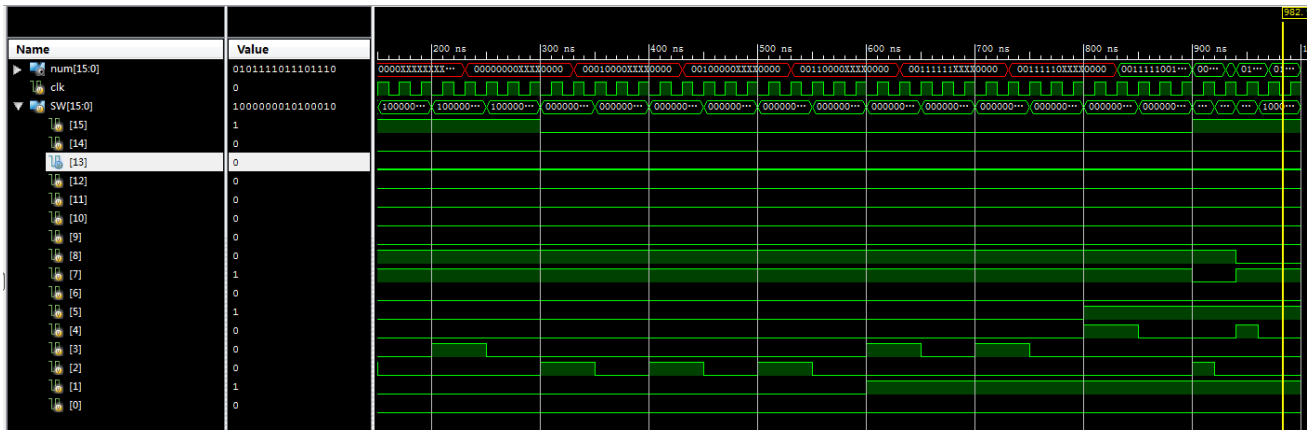
SW[15]保持为 0 不变，将 SW[6:5]置为 01，寄存器 C 准备读入 A-B 的结果，上下波动一次 SW[4]，C=A-B，输出结果为 3e50



SW[15]置为 1，三个寄存器都准备读入四选一多路复用器的输出结果，将 SW[8:7]置为 11，输出 out 等于 C，上下拨动一次 SW[2]，out 被传输到寄存器 A 中，输出结果为 5e55



将 SW[8:7]置为 10，输出 out 等于 B，上下拨动一次 SW[4]，out 被传输到寄存器 C 中，输出结果为 5eee



三、讨论、心得

本次实验主要是之前做的几次小实验的综合，ALU 部分有些遗忘，又回去复习了一下当时的课件。难点主要在于自己设计仿真代码，开始时由于对 verilog 语言掌握不够熟练，在 initial 里面写了 always 导致一直报错 syntax error，有改了一段时间才改正。通过本次实验我加深了课上对寄存器理论知识的理解，进一步学习和熟练了 verilog 的语法。

四、个人生活照片

浙江大学

本科实验报告

课程名称:	数字逻辑电路设计
姓　　名:	陈诺
学　　院:	计算机科学与技术学院
专　　业:	计算机
邮　　箱:	chennuo731@zju.edu.cn
QQ　　号:	2528064826

电　　话:	15358508348
指导教师:	洪奇军
报告日期:	2022 年 12 月 28 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: 陈诺 学号: 3210102020 同组学生姓名:

实验地点: 线上 实验日期: 2022 年 12 月 21 日

一、操作方法与实验步骤

1.1 设计 8 位带并行输入的右移移位寄存器

1. 新建工程 ShiftReg8b ，
2. 进行结构化描述设计，代码如下:

```
1. module ShiftReg8b(
2.  input wire clk, S_L, s_in,
3.  input wire [7:0] p_in,
4.  output wire [7:0] Q
```

```
    );
  wire [7:0] D;
  wire [7:0] Q_out;
  wire [15:0] OR_in;
  wire S_L_INV;
  assign Q = Q_out;
  FD m7(.C(clk),.D(D[7]),.Q(Q_out[7]));
  FD m6(.C(clk),.D(D[6]),.Q(Q_out[6]));
  FD m5(.C(clk),.D(D[5]),.Q(Q_out[5]));
  FD m4(.C(clk),.D(D[4]),.Q(Q_out[4]));
  FD m3(.C(clk),.D(D[3]),.Q(Q_out[3]));
  FD m2(.C(clk),.D(D[2]),.Q(Q_out[2]));
  FD m1(.C(clk),.D(D[1]),.Q(Q_out[1]));
  FD m0(.C(clk),.D(D[0]),.Q(Q_out[0]));
  OR2 n7(.I0(OR_in[15]),.I1(OR_in[14]),.O(D[7]));
  OR2 n6(.I0(OR_in[13]),.I1(OR_in[12]),.O(D[6]));
  OR2 n5(.I0(OR_in[11]),.I1(OR_in[10]),.O(D[5]));
  OR2 n4(.I0(OR_in[9]),.I1(OR_in[8]),.O(D[4]));
  OR2 n3(.I0(OR_in[7]),.I1(OR_in[6]),.O(D[3]));
  OR2 n2(.I0(OR_in[5]),.I1(OR_in[4]),.O(D[2]));
  OR2 n1(.I0(OR_in[3]),.I1(OR_in[2]),.O(D[1]));
  OR2 n0(.I0(OR_in[1]),.I1(OR_in[0]),.O(D[0]));
  AND2 H1(.I0(s_in),.I1(S_L_INV),.O(OR_in[15]));
  AND2 H2(.I0(p_in[7]),.I1(S_L),.O(OR_in[14]));
  AND2 G1(.I0(Q_out[7]),.I1(S_L_INV),.O(OR_in[13]));
  AND2 G2(.I0(p_in[6]),.I1(S_L),.O(OR_in[12]));
  AND2 F1(.I0(Q_out[6]),.I1(S_L_INV),.O(OR_in[11]));
  AND2 F2(.I0(p_in[5]),.I1(S_L),.O(OR_in[10]));
  AND2 E1(.I0(Q_out[5]),.I1(S_L_INV),.O(OR_in[9]));
  AND2 E2(.I0(p_in[4]),.I1(S_L),.O(OR_in[8]));
  AND2 D1(.I0(Q_out[4]),.I1(S_L_INV),.O(OR_in[7]));
  AND2 D2(.I0(p_in[3]),.I1(S_L),.O(OR_in[6]));
  AND2 C1(.I0(Q_out[3]),.I1(S_L_INV),.O(OR_in[5]));
  AND2 C2(.I0(p_in[2]),.I1(S_L),.O(OR_in[4]));
  AND2 B1(.I0(Q_out[2]),.I1(S_L_INV),.O(OR_in[3]));
  AND2 B2(.I0(p_in[1]),.I1(S_L),.O(OR_in[2]));
  AND2 A1(.I0(Q_out[1]),.I1(S_L_INV),.O(OR_in[1]));
  AND2 A2(.I0(p_in[0]),.I1(S_L),.O(OR_in[0]));
  INV S_L_(.I(S_L),.O(S_L_INV));

endmodule
```

3. 波形仿真，激励代码如下：

```
initial begin
```

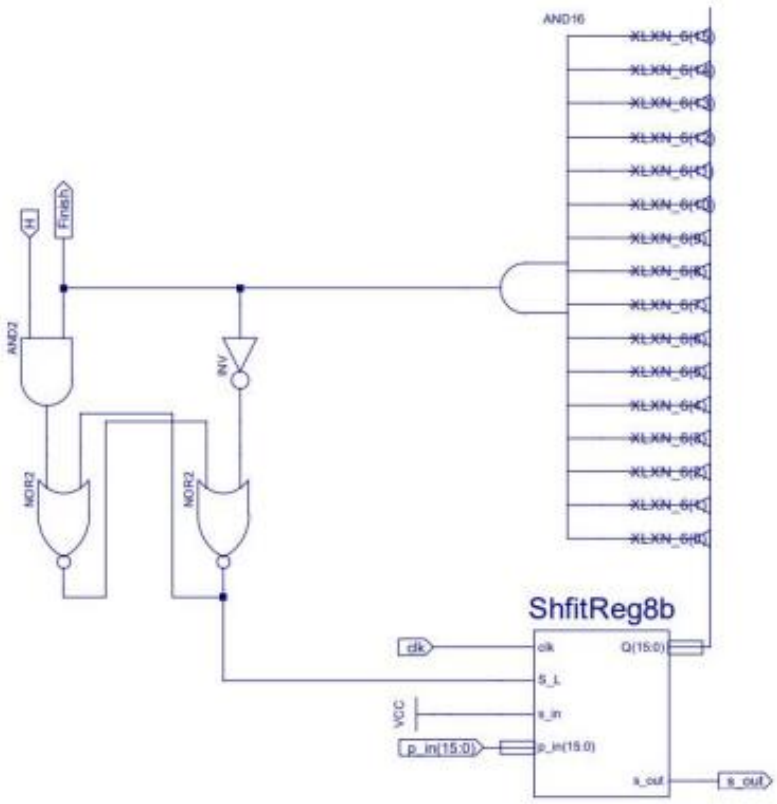
```
    clk = 0;
    S_L = 0;
    s_in = 0;
    p_in = 0; #100;
    S_L = 0;
    s_in = 1;
    p_in =0; #200;
    S_L = 1;
    s_in = 0;
    p_in = 8'b0101_0101;#500;
end
always begin
    clk = 0; #20;
    clk = 1; #20;
end
```

1.2 设计主板 LED 灯驱动模块

1. 新建工程 LEDP2S ，Top Level Source Type 用 HDL，修改 1.1 中的八位右移寄存器为 16 位右移寄存器，接口如下：

```
module ShfitReg16b(
input wire clk, S_L, s_in,
input wire [15:0] p_in,
output wire [15:0] Q,
output wire s_out
);
```

2. 根据 16 位右移寄存器，设计 LED 灯驱动模块：



原理图如上

```
1. module LEDDRV(
2.   input wire clk,
3. input wire H,
4. input wire [15:0]p_in,
5. output wire Finish,
6. output wire s_out
7.     );
8.   wire nor1;
9. wire nor2;
10.wire and1;
11.wire inv1;
12.wire finish;
13.wire [15:0]Q;
14.assign finish = Finish;
15.AND2 a2(.I0(H), .I1(Finish), .O(and1));
16.INV i1(.I(finish), .O(inv1));
17.NOR2 n1(.I0(and1), .I1(nor2), .O(nor1));
18.NOR2 n2(.I0(inv1), .I1(nor1), .O(nor2));
19.ShiftReg16b myreg(.clk(clk), .S_L(nor2), .s_in(1'b1), .p_in(p_in)
   , .Q(Q), .s_out(s_out));
20.assign Finish = &Q;
```

```
endmodule
```

3. 进行仿真，激励代码如下：

```
initial begin
   clk = 0;
   H=0;
   p_in = 0; #100;
   H=1;

   p_in = 16'b0101_0101_0101_0101;#500;
  end
  always begin
   clk = 0; #20;
   clk = 1; #20;
  end
```

4. 设计顶层模块，代码如下：

```
module top(
  input wire clk,
  input wire S_CLR,
  input wire S,
  input wire [15:0] num,
  output wire LED_CLK,
  output wire LED_CLR,
  output wire LED_EN,
  output wire LED_DO
     );

 wire Finish;
 wire LED_DO_0;

 LEDDRV drv(.clk(clk), .H(S), .p_in(num), .Finish(finish), .s_out
   (LED_DO_0));

 assign LED_EN = 1'b1;
 assign LED_CLK = clk || finish;
 assign LED_CLR = S_CLR;
 assign LED_DO = ~LED_DO_0;

endmodule
```

1.3 设计主板七段数码管驱动模块

1. 新建工程 SEGP2S ，Top Level Source Type 用 HDL，修改 1.1 中的八位右移寄存器为 32 位右移寄存器，接口如下：

```
1. module ShfitReg32b(
2. input wire clk, S_L, s_in,
3. input wire [31:0] p_in,
4. output wire [31:0] Q,
5. output wire s_out
6. );
```

2. 根据 32 位右移寄存器，设计数码管驱动模块，代码如下：

```
1. module SEGDRV(
2. input wire clk,
3. input wire H,
4. input wire [31:0]p_in,
5. output wire Finish,
6. output wire s_out
7.     );
8.
9. wire nor1;
10.wire nor2;
11.wire and1;
12.wire inv1;
13.wire finish;
14.wire [31:0]Q;
15.assign finish = Finish;
16.AND2 a2(.I0(H), .I1(Finish), .O(and1));
17.INV i1(.I(finish), .O(inv1));
18.NOR2 n1(.I0(and1), .I1(nor2), .O(nor1));
19.NOR2 n2(.I0(inv1), .I1(nor1), .O(nor2));
20.ShiftReg32b myreg(.clk(clk), .S_L(nor2), .s_in(1'b1), .p_in(p_in)
   , .Q(Q), .s_out(s_out));
21.assign Finish = &Q;
22.endmodule
```

3.进行仿真，激励代码如下：

```
1. initial begin
2.   // Initialize Inputs
3.   clk = 0;
4.   H = 0;
5.   p_in = 0;
6.
7.   // Wait 100 ns for global reset to finish
8.   #100;
9.
```

```
10.  // Add stimulus here
11.  H=1;
12.  p_in = 32'b0101_0101_0101_0101_0101_0101_0101_0101;#500;
13. end
14. always begin
15.  clk = 0; #20;
16.  clk = 1; #20;
17. end
```

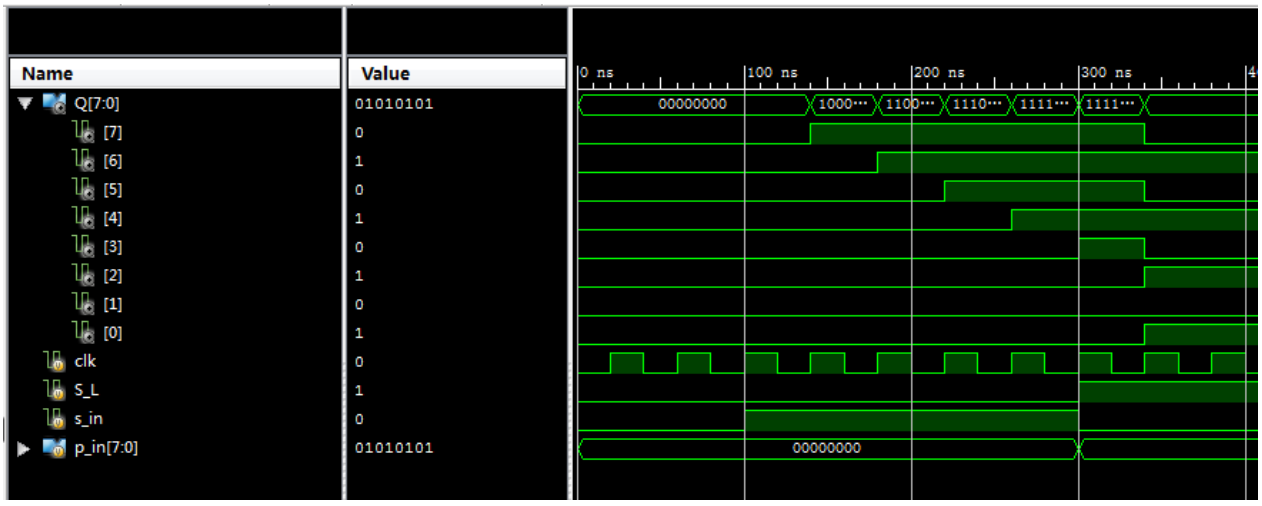
4.设计顶层模块，代码如下：

```
1. module top(
2. input wire clk,
3. input wire S,
4. input wire S_CLR,
5. input wire [7:0] sw,
6. output wire SEGCLK,
7. output wire SEGCLR,
8. output wire SEGDT,
9. output wire SEGEN
10.    );
11.wire [63:0] Seg;
12.wire [31:0] num;
13.wire finish;
14.
15.MyMC14495 Seg1(.D0(num[0]), .D1(num[1]), .D2(num[2]), .D3(num[3])
   , .LE(1'b0), .point(1'b0), .a(Seg[0]), .b(Seg[1]), .c(Seg[2]), .d
   (Seg[3]), .e(Seg[4]), .f(Seg[5]), .g(Seg[6]), .p(Seg[7] ));
16.MyMC14495 Seg2(.D0(num[4]), .D1(num[5]), .D2(num[6]), .D3(num[7])
   , .LE(1'b0), .point(1'b0), .a(Seg[8]),.b(Seg[9]), .c(Seg[10]), .d
   (Seg[11]), .e(Seg[12]), .f(Seg[13]), .g(Seg[14]), .p(Seg[15]));
17.MyMC14495 Seg3(.D0(num[8]), .D1(num[9]), .D2(num[10]), .D3(num[11
   ]), .LE(1'b0), .point(1'b0), .a(Seg[16]), .b(Seg[17]), .c(Seg[18]
   ), .d(Seg[19]), .e(Seg[20]), .f(Seg[21]), .g(Seg[22]), .p(Seg[23]
   ) );
18.MyMC14495 Seg4(.D0(num[12]), .D1(num[13]), .D2(num[14]), .D3(num[
   15]), .LE(1'b0), .point(1'b0), .a(Seg[24]), .b(Seg[25]), .c(Seg[2
   6]), .d(Seg[27]), .e(Seg[28]), .f(Seg[29]), .g(Seg[30]), .p(Seg[3
   1]));
19.MyMC14495 Seg5(.D0(num[16]), .D1(num[17]), .D2(num[18]), .D3(num[
   19]), .LE(1'b0), .point(1'b0), .a(Seg[32]), .b(Seg[33]), .c(Seg[3
   4]), .d(Seg[35]), .e(Seg[36]), .f(Seg[37]), .g(Seg[38]), .p(Seg[3
   9]));
20.MyMC14495 Seg6(.D0(num[20]), .D1(num[21]), .D2(num[22]), .D3(num[
   23]), .LE(1'b0), .point(1'b0), .a(Seg[40]), .b(Seg[41]), .c(Seg[4
```

```
   2]), .d(Seg[43]), .e(Seg[44]), .f(Seg[45]), .g(Seg[46]), .p(Seg[4
   7]));
21.MyMC14495 Seg7(.D0(num[24]), .D1(num[25]), .D2(num[26]), .D3(num[
   27]), .LE(1'b0), .point(1'b0), .a(Seg[48]), .b(Seg[49]), .c(Seg[5
   0]), .d(Seg[51]), .e(Seg[52]), .f(Seg[53]), .g(Seg[54]), .p(Seg[5
   5]));
22.MyMC14495 Seg8(.D0(num[28]), .D1(num[29]), .D2(num[30]), .D3(num[
   31]), .LE(1'b0), .point(1'b0), .a(Seg[56]), .b(Seg[57]), .c(Seg[5
   8]), .d(Seg[59]), .e(Seg[60]), .f(Seg[61]), .g(Seg[62]), .p(Seg[6
   3]));
23.SEGDRV drv(.clk(clk), .H(S), .p_in(Seg), .Finish(finish), .s_out(
   SEGDT));
24.assign SEGEN = 1'b1;
25.assign SEGCLK = clk | finish;
26.assign SEGCLR = S_CLR;
27.
28.endmodule
```

二、实验结果与分析

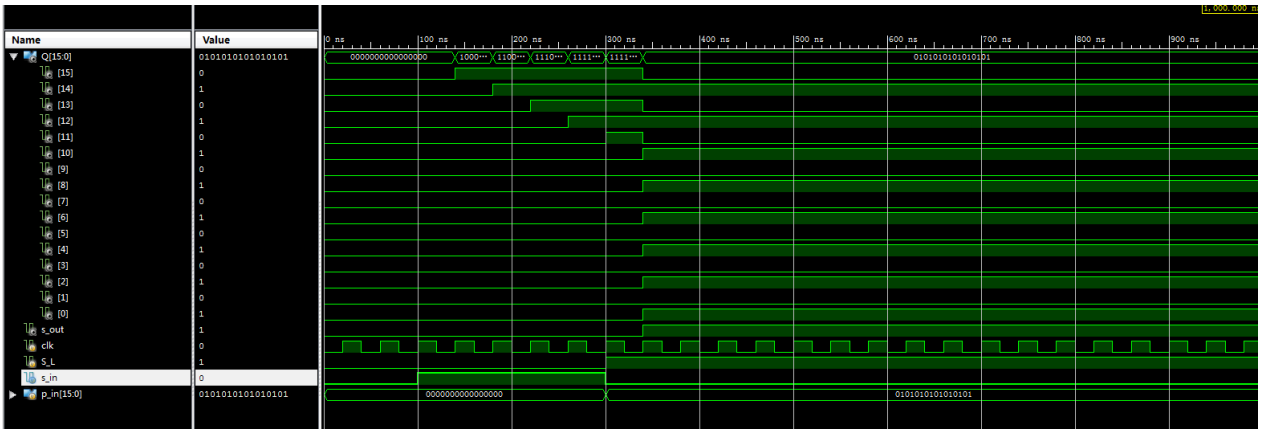
2.1 设计 8 位带并行输入的右移移位寄存器



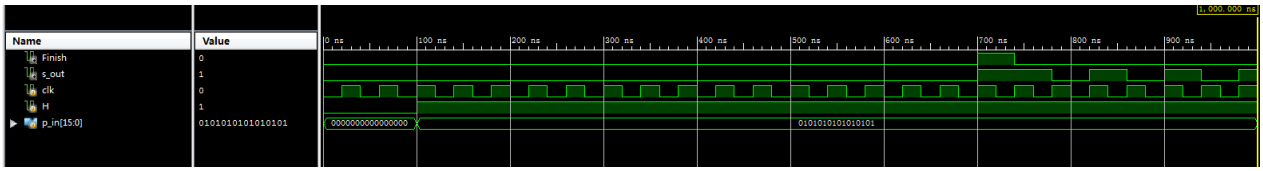
由图可见，寄存器高位 0 逐渐拓展至低位，寄存器功能得以实现

2.2 设计主板 LED 灯驱动模块

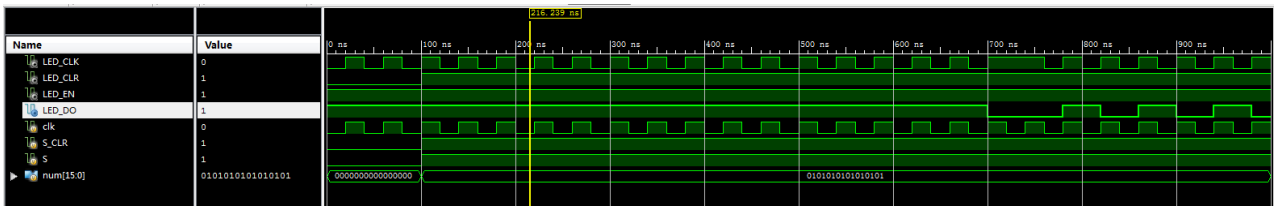
16 位移位寄存器仿真：



LED 驱动器仿真：

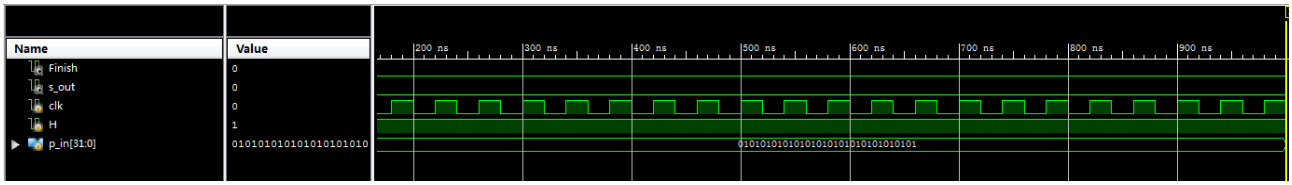


top 仿真：



2.3 设计主板七段数码管驱动模块

数码管驱动器仿真：



三、讨论、心得

本次实验完成得不算很顺利，开始没有搞明白实验目的和原理，后来通过询问求助后大致理顺了本次实验的要求。由于线上无法下到板上验证，许多环节都省略了，就进行了一些基础的仿真。这次实验也是没有用以前的画图，尝试使用 verilog 代码编写模块，有了一定进步。

四、个人生活照片

浙江大学

本科实验报告

课程名称:	数字逻辑电路设计
姓　　名:	陈诺
学　　院:	计算机科学与技术学院
专　　业:	计算机
邮　　箱:	chennuo731@zju.edu.cn
QQ　　号:	2528064826

电　　话:	15358508348
指导教师:	洪奇军
报告日期:	2022 年 12 月 29 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: 计数器、定时器设计与应用

学生姓名: 陈诺 学号: 3210102020 同组学生姓名:

实验地点: 线上 实验日期: 2022 年 12 月 27 日

一、操作方法与实验步骤

1.1 采用行为描述设计同步四位二进制计数器 74LS161

1. 创建工程 My74LS161，Top Level Source Type 为 HDL
2. 行为描述设计实现 My74LS161 模块，代码如下:

```
1. module My74LS161(
2. CP, CR, CTP, CTT, Ld, D, Co, Q
3.     );
4. input CP, CR, CTP, CTT, Ld;
```

```
input [3:0]D;
output reg Co;
output reg [3:0]Q;
initial Co <= 1'b0;
always @ (posedge CP or negedge CR) begin
 if (~CR) Q <= 4'b0;
 else if(~Ld) Q <= D;
 else if(CTP&CTT) begin
  Co <= &(Q[3:0]);
  Q <= Q+1;
 end
 else Q <= Q;
end

endmodule
```

3. 进行波形仿真，激励代码如下：

```
initial begin
   CR = 0;
   D = 0;
   CTP = 0;
   CTT = 0;
   Ld = 0;
   #100;
   CR = 1;
   Ld = 1;
   D = 4'b1100;
   CTT = 0;
   CTP = 0;
   #30 CR = 0;
   #20 CR = 1;

   #10 Ld = 0;
   #30 CTT = 1;
   CTP = 1;
   #10 Ld = 1;
   #510;
   CR = 0;
   #20 CR = 1;
   #500;
 end
 always begin
   #20 CP = 1;
   #20 CP = 0;
```

```
28. end
```

1.2 基于 74LS161 设计时钟应用

1. 新建工程 MyClock , Top Level Source Type 为 HDL
2. 时分秒计数器实现如下:

m1-m6 分别是秒、分、时计数的低位到高位，cr 信号将计数器清零，ld 信号将指定值（23:59:58）输入到寄存器里

```
1. module Test(
2. clk, ld, cr, Q
3.     );
4. input clk, ld, cr;
5. output wire [23:0]Q;
6. My74LS161 m1(.CP(clk), .D(4'b1000), .CR((~ld) ||(~((Q[3]&&Q[1]) ||cr))), .CTP(1'b1), .CTT(1'b1), .Ld(ld), .Q(Q[3:0]));
7. My74LS161 m2(.CP(clk), .D(4'b0101), .CR((~ld) ||(~((Q[6]&&Q[5]) ||cr))), .CTP(Q[3]&&Q[0]), .CTT(1'b1), .Ld(ld), .Q(Q[7:4]));
8. My74LS161 m3(.CP(clk), .D(4'b1001), .CR((~ld) ||(~((Q[11]&&Q[9]) ||cr))), .CTP(Q[6]&&Q[4]&&Q[3]&&Q[0]), .CTT(1'b1), .Ld(ld), .Q(Q[11:8]));
9. My74LS161 m4(.CP(clk), .D(4'b0101), .CR((~ld) ||(~((Q[14]&&Q[13]) ||cr))), .CTP(Q[11]&&Q[8]&&Q[6]&&Q[4]&&Q[3]&&Q[0]), .CTT(1'b1), .Ld(ld), .Q(Q[15:12]));
10.My74LS161 m5(.CP(clk), .D(4'b0011), .CR((~ld) || (~((Q[19]&&Q[17]) ||(Q[21]&&Q[18]) ||cr))), .CTP(Q[14]&&Q[12]&&Q[11]&&Q[8]&&Q[6]&&Q[4]&&Q[3]&&Q[0]), .CTT(1'b1), .Ld(ld),  .Q(Q[19:16]));
11.My74LS161 m6(.CP(clk), .D(4'b0010), .CR((~ld) || (~((Q[21]&&Q[18]) ||cr))), .CTP(Q[19]&&Q[16]&&Q[14]&&Q[12]&&Q[11]&&Q[8]&&Q[6]&&Q[4]&&Q[3]&&Q[0]), .CTT(1'b1), .Ld(ld), .Q(Q[23:20]));
12.
13.endmodule
```

3．进行仿真，激励代码如下:

```
1. initial begin
2.   // Initialize Inputs
3.   clk = 0;
4.   ld = 0;
5.   cr = 0;
6.   #100;
7.   #50 cr = 1;
8.   #50 cr = 0;
9.   #100 ld = 1;
10.
```

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

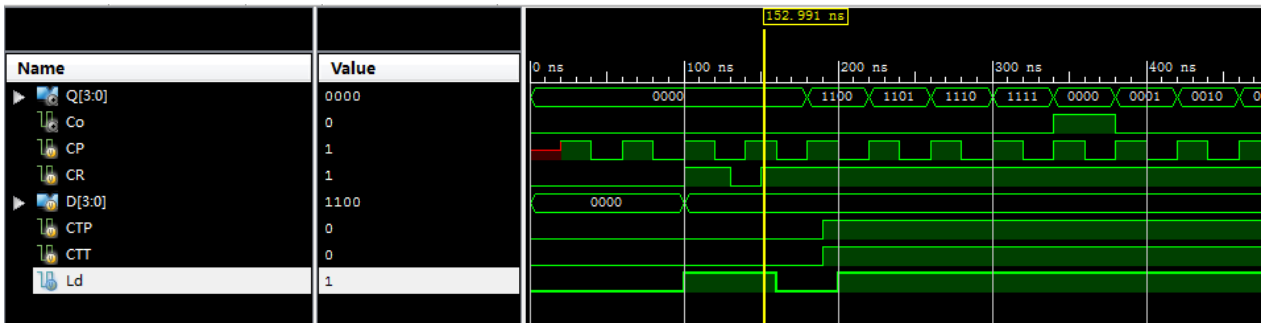
4. top 模块实现如下：

```
module top(
input wire clk,
input wire ld,
input wire cr,
output wire SEGCLK,
output wire SEGCLR,
output wire SEGDT,
output wire SEGEN
    );
wire [23:0] Q;
wire clk_100ms;
wire [31:0]clk_div;
wire start;
clkdiv_Number c(clk,1'b0,clk_div);
Load_Gen l0 (clk,clk_div[17],start);
My74LS161 m1(.CP(clk), .D(4'b1000), .CR((~ld) ||(~((Q[3]&&Q[1]) ||cr))), .CTP(1'b1), .CTT(1'b1), .Ld(ld), .Q(Q[3:0]));
My74LS161 m2(.CP(clk), .D(4'b0101), .CR((~ld) ||(~((Q[6]&&Q[5]) ||cr))), .CTP(Q[3]&&Q[0]), .CTT(1'b1), .Ld(ld), .Q(Q[7:4]));
My74LS161 m3(.CP(clk), .D(4'b1001), .CR((~ld) ||(~((Q[11]&&Q[9]) ||cr))), .CTP(Q[6]&&Q[4]&&Q[3]&&Q[0]), .CTT(1'b1), .Ld(ld), .Q(Q[11:8]));
My74LS161 m4(.CP(clk), .D(4'b0101), .CR((~ld) ||(~((Q[14]&&Q[13]) ||cr))), .CTP(Q[11]&&Q[8]&&Q[6]&&Q[4]&&Q[3]&&Q[0]), .CTT(1'b1), .Ld(ld), .Q(Q[15:12]));
My74LS161 m5(.CP(clk), .D(4'b0011), .CR((~ld) || (~((Q[19]&&Q[17]) ||(Q[21]&&Q[18]) ||cr))), .CTP(Q[14]&&Q[12]&&Q[11]&&Q[8]&&Q[6]&&Q[4]&&Q[3]&&Q[0]), .CTT(1'b1), .Ld(ld),  .Q(Q[19:16]));
My74LS161 m6(.CP(clk), .D(4'b0010), .CR((~ld) || (~((Q[21]&&Q[18]) ||cr))), .CTP(Q[19]&&Q[16]&&Q[14]&&Q[12]&&Q[11]&&Q[8]&&Q[6]&&Q[4]&&Q[3]&&Q[0]), .CTT(1'b1), .Ld(ld), .Q(Q[23:20]));
SegSig Sig(.clk(clk), .S(start), .S_CLR(1'b0), .num(Q),.SEGCLK(SEGCLK), .SEGCLR(SEGCLR), .SEGDT(SEGDT), .SEGEN(SEGEN));

endmodule
```

二、实验结果与分析

2.1 采用行为描述设计同步四位二进制计数器 74LS161

仿真结果如下：

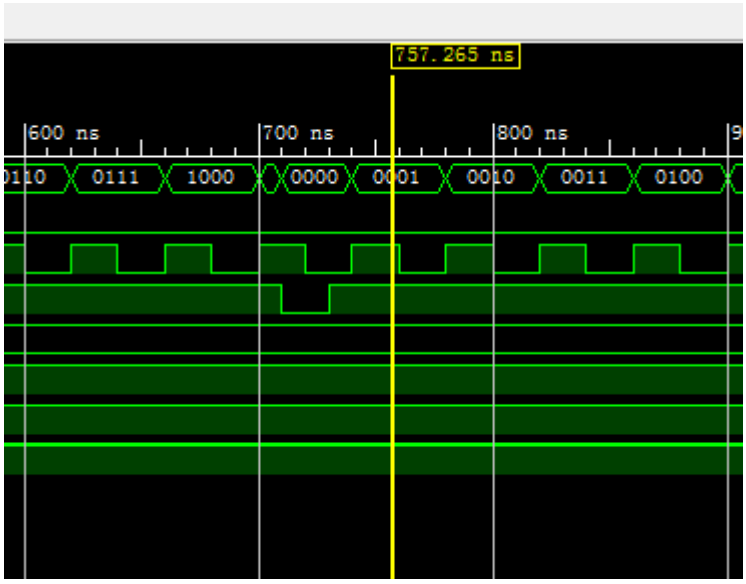


如图，180ns 处，ld 信号功能得到验证

260-360ns 处，保持功能得到验证

360ns 处，自增功能得到验证

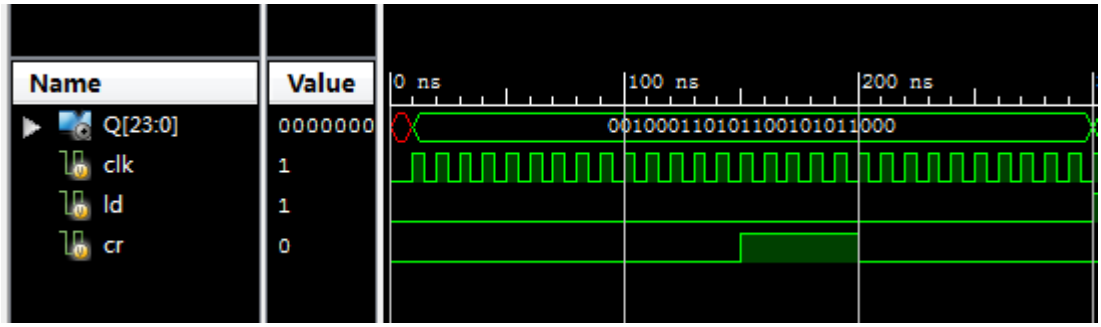
以上功能受时钟同步信号控制



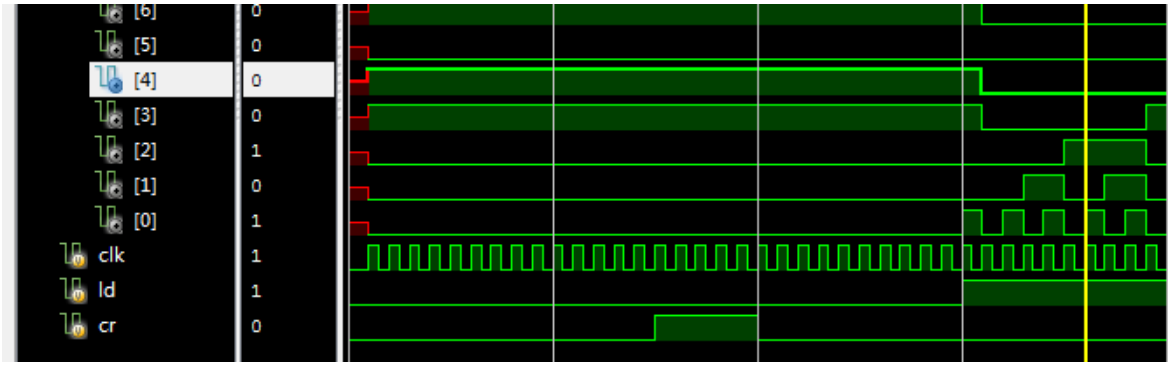
720ns 处，异步清零功能得到验证

2.2 基于 74LS161 设计时钟应用

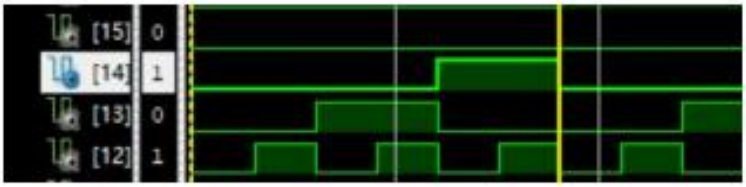
仿真结果如下：



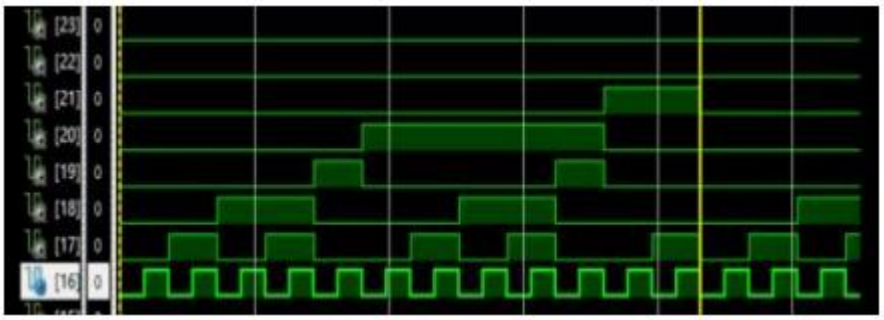
上图验证了计时器的初始化



如图，从黄线部分开始，验证了秒计时个位十进制计数器的正确性（见[3:0]）



如图，两条黄线之间验证了分计时十位六进制计数器的正确性（见[15:12]）



如图，两条黄线之间验证了小时计时 24 进制计数器的正确性（见[23:16]）

三、讨论、心得

本次实验使我对层级设计有了新的理解。最开始时钟计数是分两部分设计的：60 进制计数器和 24 进制计数器，但最后由于分立设计会导致计数器产生的进位信号与时钟存在一定冲突，所以将整个时钟计数器进行整合。整合后一些冗余的逻辑结构可以省区，提高设计的效率，有时也可以避免一些信号冲突。

四、个人生活照片