

浙江大学



计算机逻辑设计基础

Lab 13

Author: 苏煜程

Student ID: 3220105481

Date: 2023 年 12 月 28 日

浙江大学实验报告

课程名称：计算机逻辑设计基础

实验名称：移位寄存器设计与应用

学生姓名：苏煜程 专业：人工智能（图灵班） 学号：3220105481

指导老师：董亚波 实验地点：东 4-509 实验日期：2023 年 12 月 13 日

1 实验目的

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

2 实验任务

1. 设计 8 位带并行输入的右移移位寄存器
2. 设计主板 16 位 LED 灯驱动模块
3. 设计主板 8 位数码管驱动模块

3 实验原理

3.1 移位寄存器

1. 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
 - 不能采用锁存器
2. 数据移动方式：左移、右移、循环移位
3. 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

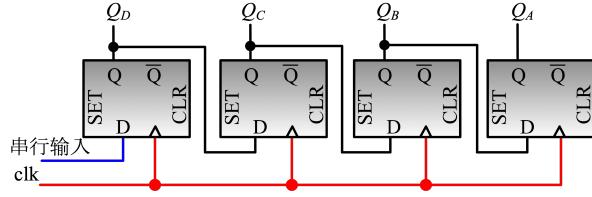


图 1: 使用 D 触发器构成串行输入的右移移位寄存器

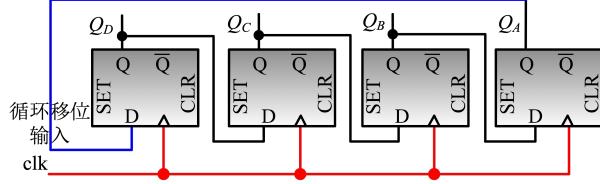


图 2: 循环右移移位寄存器

3.2 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

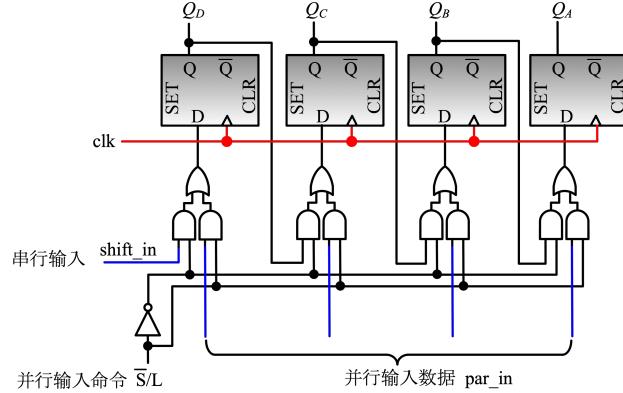


图 3: 带并行输入的移位寄存器

3.3 接口说明: 实验板 16 位 LED 灯

实验板上，采用 2 个 8 位移位寄存器 74LV164A 构成 16 位串行左移移位寄存器，寄存器的并行输出控制 16 个 LED 灯 LED0 - LED15。

- LED_CLK: 16 位 LED 灯模块的时钟，上升沿触发移位
- LED_CLR: 清零，使所有 LED 亮，低电平有效
- LED_D0: 串行移位数据输入，0 使 LED 亮

- LED_EN: LED 模块总控开关, 1 为使能

用 LED_CLK 和 LED_D0 按顺序 LED15, LED14, ……, LED1, LED0 串行移入 16 位数据。

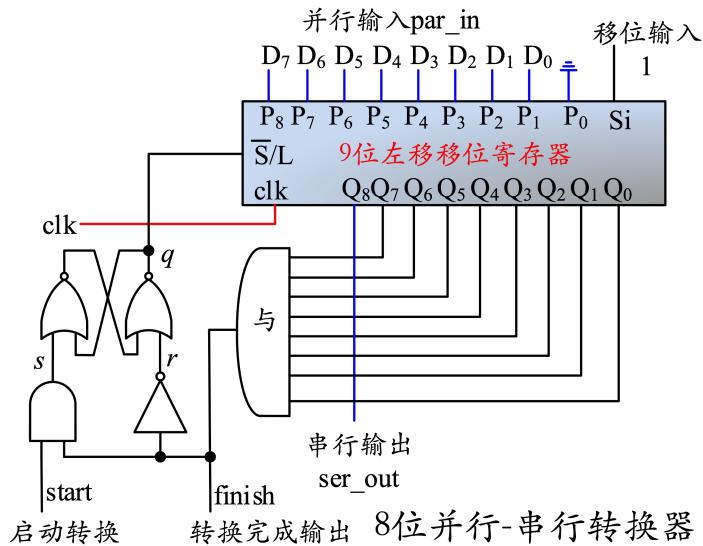
3.3.1 并行-串行转换

16 位数据移位完成后要停止时钟, 避免把之前的数据移出 16 位 LED 灯模块。

采用门控时钟方式给 LED_CLK 提供时钟信号:

```
assign LED_CLK = clk | finish;
```

其中 `finish` 为 1 时标志转换结束。为了在移位过程中产生结束标志, 需要使用 9 位串行移位寄存器移动 8 位数据:



`start` 启动信号拉高以后, 加载并行输入 D0-D7, 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。

1. 没有启动命令时: 寄存器处于左移状态, 将 1 左移移入 `ser_out`。
2. 有启动命令 (`start` 高电平) 时: SL 锁存器输入 $S = 1, R = 0$, 输出 $Q = 1$ 到移位寄存器, 启用并行输入, 将 $D[7:0]$, 0 移入寄存器 $Q[8:0]$ 。
3. 并行输入后, `finish` 变为 0, SL 锁存器输入 $S = 0, R = 1$, 输出 $Q = 0$ 到移位寄存器, 禁用并行输入, 同时 `LED_CLK = clk`, 开始移位。
4. 当 D0 被移出时, `finish` 变为 1, SL 锁存器输入 $S = 0, R = 0$ 保持输出 $Q = 0$, 同时 `LED_CLK` 为 1, 移位停止。

3.4 接口说明：主板七段数码管

实验板上，8个74LS164A的并行输出控制8个7段数码管的段码。

- SEGCLK: 8位七段数码管模块的时钟，上升沿触发移位
- SEGCLR: 清零，所有段亮，低电平有效
- SEGDT: 串行移位数据输入，0亮
- SEGEN: 8位七段数码管模块总控开关，1为使能

用SEGCLR和SEGDT按顺序SEG7_DP, SEG7_g, SEG7_f, ……, SEGO_b, SEGO_a串行移入64位数据。

数码管为共阳接法，段码为0时对应段亮。

4 实验内容与步骤

4.1 任务1：设计8位带并行输入的右移移位寄存器

1. 新建工程，工程名称用ShiftReg8b，Top Level Source Type用HDL。
2. 用结构化描述设计。

```
1 `timescale 1ns / 1ps
2 module shift_reg(
3     input wire clk, S_L, s_in,
4     input wire [7:0] p_in,
5     output wire [7:0] Q
6 );
7
8     FD m0(.C(clk),
9         .D((Q[1] & !S_L) | (p_in[0] & S_L)),
10        .Q(Q[0]));
11
12    FD m1(.C(clk),
13        .D((Q[2] & !S_L) | (p_in[1] & S_L)),
14        .Q(Q[1]));
15
16    FD m2(.C(clk),
17        .D((Q[3] & !S_L) | (p_in[2] & S_L)),
18        .Q(Q[2]));
19
20    FD m3(.C(clk),
21        .D((Q[4] & !S_L) | (p_in[3] & S_L)),
22        .Q(Q[3]));
23
```

```

24     FD m4(.C(clk),
25         .D((Q[5] & !S_L) | (p_in[4] & S_L)),
26         .Q(Q[4]));
27
28     FD m5(.C(clk),
29         .D((Q[6] & !S_L) | (p_in[5] & S_L)),
30         .Q(Q[5]));
31
32     FD m6(.C(clk),
33         .D((Q[7] & !S_L) | (p_in[6] & S_L)),
34         .Q(Q[6]));
35
36     FD m7(.C(clk),
37         .D((s_in & !S_L) | (p_in[7] & S_L)),
38         .Q(Q[7]));
39
endmodule

```

3. 波形仿真。

仿真代码：

```

1 initial begin
2     clk = 0;
3     S_L = 0;
4     s_in = 0;
5     p_in = 0;
6
7     #100;
8
9     S_L = 0;
10    s_in = 1;
11    p_in = 0;
12    #100;
13    s_in = 0;
14    #100;
15    S_L = 1;
16    s_in = 0;
17    p_in = 8'b1010_1010;
18    #200;
19
20    clk = 0;
21    S_L = 0;
22    s_in = 0;
23    p_in = 0;
24
end
25
26 always begin

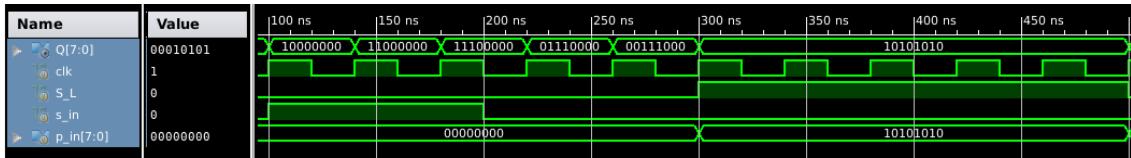
```

```

27     clk = 0; #20;
28     clk = 1; #20;
29 end

```

仿真波形：



仿真激励代码中，先将 `SL` 置为 0 启用串行输入，将串行输入 `s_in` 置为 1，可见输出 `Q` 每时钟上升沿右移入一位 1。随后将串行输入 `s_in` 置为 0，可见输出 `Q` 每时钟上升沿右移入一位 0。最后将并行输入 `p_in` 置为 `8'b1010_1010`，然后将 `SL` 置为 1 启用并行输入，可见输出 `Q` 被 `p_in` 对应位初始化。

4.2 任务 2：设计主板 16 位 LED 驱动模块

4.2.1 工程设计

- 新建工程，工程名称用 `LEDP2S`，Top Level Source Type 用 HDL。

- 用行为描述设计。

- 简化实验 12 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。

```

1 RegCounter4b reg0(.clk(clk),
2             .btn(BTN[0]),
3             .CLR(SW[14]),
4             .num(num[3:0]));
5 RegCounter4b reg1(.clk(clk),
6             .btn(BTN[1]),
7             .CLR(SW[14]),
8             .num(num[7:4]));
9 RegCounter4b reg2(.clk(clk),
10            .btn(BTN[2]),
11            .CLR(SW[14]),
12            .num(num[11:8]));
13 RegCounter4b reg3(.clk(clk),
14            .btn(BTN[3]),
15            .CLR(SW[14]),
16            .num(num[15:12]));
17 DispNum disp0(.clk(clk),
18             .HEXS(num),
19             .LES(4'b0),
20             .point(4'b0),

```

```

21      .RST(1'b0),
22      .AN(AN),
23      .Segment(Segment));

```

- 改造 ShiftReg8b 模块为左移寄存器 SLReg8b。为了在移位过程中产生结束标志，这里改造为 SLReg9b。

```

1 module SLReg9b(
2     input wire clk, S_L, s_in,
3     input wire [8:0] p_in,
4     output wire [8:0] Q // Q[8] is overflow bit
5 );
6     FD #( .INIT(1'b1) ) m0(.C(clk),
7         .D((s_in & !S_L) | (p_in[0] & S_L)),
8         .Q(Q[0]));
9     FD #( .INIT(1'b1) ) m1(.C(clk),
10        .D((Q[0] & !S_L) | (p_in[1] & S_L)),
11        .Q(Q[1]));
12    FD #( .INIT(1'b1) ) m2(.C(clk),
13        .D((Q[1] & !S_L) | (p_in[2] & S_L)),
14        .Q(Q[2]));
15    FD #( .INIT(1'b1) ) m3(.C(clk),
16        .D((Q[2] & !S_L) | (p_in[3] & S_L)),
17        .Q(Q[3]));
18    FD #( .INIT(1'b1) ) m4(.C(clk),
19        .D((Q[3] & !S_L) | (p_in[4] & S_L)),
20        .Q(Q[4]));
21    FD #( .INIT(1'b1) ) m5(.C(clk),
22        .D((Q[4] & !S_L) | (p_in[5] & S_L)),
23        .Q(Q[5]));
24    FD #( .INIT(1'b1) ) m6(.C(clk),
25        .D((Q[5] & !S_L) | (p_in[6] & S_L)),
26        .Q(Q[6]));
27    FD #( .INIT(1'b1) ) m7(.C(clk),
28        .D((Q[6] & !S_L) | (p_in[7] & S_L)),
29        .Q(Q[7]));
30    FD #( .INIT(1'b1) ) m8(.C(clk),
31        .D((Q[7] & !S_L) | (p_in[8] & S_L)),
32        .Q(Q[8]));
33 endmodule

```

为了使得 `finish` 的初值正确，这里将 FD 模块的 INIT 参数置为 1。

- 利用 2 个 SLReg8b 模块和 1 个触发器，设计 16 位 LED 驱动模块 LED_DRV。

```

1 `timescale 1ns / 1ps
2 module LED_DRV(
3     input wire clk,
4     input wire [15:0] SW,

```

```

5   input wire [3:0] BTN,
6   output wire LED_CLK,
7   output wire LED_CLR,
8   output wire LED_DO,
9   output wire LED_EN,
10  output wire [3:0] AN,
11  output wire [7:0] Segment,
12  output wire BTNX4
13 );
14 wire [15:0] num;
15 wire [31:0] clk_div;
16 wire [17:0] Q;
17 wire finish, start, SL;
18 wire ser_out;
19 assign ser_out = Q[16];
20 clkdiv m0(.clk(clk),
21             .rst(1'b0),
22             .clkdiv(clk_div));
23 RegCounter4b reg0(.clk(clk),
24                     .btn(BTN[0]),
25                     .CLR(SW[14]),
26                     .num(num[3:0]));
27 RegCounter4b reg1(.clk(clk),
28                     .btn(BTN[1]),
29                     .CLR(SW[14]),
30                     .num(num[7:4]));
31 RegCounter4b reg2(.clk(clk),
32                     .btn(BTN[2]),
33                     .CLR(SW[14]),
34                     .num(num[11:8]));
35 RegCounter4b reg3(.clk(clk),
36                     .btn(BTN[3]),
37                     .CLR(SW[14]),
38                     .num(num[15:12]));
39 DispNum disp0(.clk(clk),
40                 .HEXS(num),
41                 .LES(4'b0),
42                 .point(4'b0),
43                 .RST(1'b0),
44                 .AN(AN),
45                 .Segment(Segment));
46 assign LED_CLK = clk | finish;
47 assign LED_CLR = 1'b1;
48 assign LED_DO = ~ser_out;
49 assign LED_EN = SW[15];
50 SR_latch sr0(.S(start & finish),
51               .R(~finish),

```

```

52          .Q(SL));
53      assign finish = Q[0] & Q[1] & Q[2] & Q[3] & Q[4] & Q[5] & Q[6] & Q[7] & Q[8] &
54          Q[9] & Q[10] & Q[11] & Q[12] & Q[13] & Q[14] & Q[15];
55      Load_Gen load0(.clk(clk),
56                      .clk_1ms(clk_div[17]),
57                      .btn_in(SW[15]),
58                      .Load_out(start));
59      SLReg9b s10(.clk(clk),
60                      .S_L(SL),
61                      .s_in(1'b1),
62                      .p_in({num[7:0], 1'b0}),
63                      .Q(Q[8:0]));
64      SLReg9b s11(.clk(clk),
65                      .S_L(SL),
66                      .s_in(Q[8]),
67                      .p_in({1'b0, num[15:8]}),
68                      .Q(Q[17:9]));
69      assign BTNX4 = 1'b0;
70  endmodule

```

4.2.2 仿真

设计激励代码，对驱动模块进行仿真。

仿真之前，参考实验 12 修改代码，去掉去抖、扫描显示等存在大延迟的模块，去掉 AN、SEGMENT 输出端口。并在顶层模块中添加输出 num_disp，用于显示总线 num 的值。

仿真激励代码：

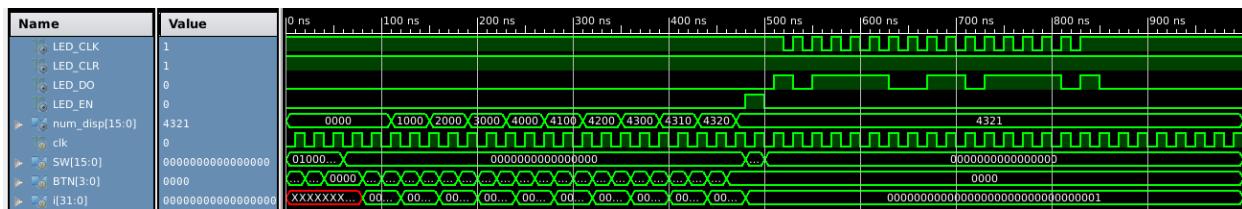
```

1 integer i;
2 initial begin
3     SW = 0;
4     BTN = 0;
5
6     // Initialize Registers
7     SW[14] = 1'b1; #20;
8     BTN = 4'b1111; #20;
9     BTN = 4'b0000; #20;
10
11    // Set Registers
12    SW[14] = 1'b0; #20;
13    for (i = 0; i < 4; i = i + 1) begin
14        BTN[3] = 1'b1; #20;
15        BTN[3] = 1'b0; #20;
16    end
17    for (i = 0; i < 3; i = i + 1) begin
18        BTN[2] = 1'b1; #20;
19        BTN[2] = 1'b0; #20;

```

```
20
21     for (i = 0; i < 2; i = i + 1) begin
22         BTN[1] = 1'b1; #20;
23         BTN[1] = 1'b0; #20;
24     end
25     for (i = 0; i < 1; i = i + 1) begin
26         BTN[0] = 1'b1; #20;
27         BTN[0] = 1'b0; #20;
28     end
29
30     SW[15] = 1'b1; #20;
31     SW[15] = 1'b0;
32 end
33
34 always begin
35     clk = 0; #10;
36     clk = 1; #10;
37 end
```

仿真波形：



仿真激励代码中，先将 SW[14] 置为 1，按下四个按钮使得 num 的值初始化为 0。随后将 SW[14] 置为 0，初始化寄存器为 4321h。随后拨动 SW[15] 启用串行输入，可见输出 LED_D0 和 LED_CLK 能正确地将 num 的值移入 16 位 LED 灯模块。

4.2.3 上板验证

- 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。

使用 `BTN[3:0]` 作为自增按键，设置 4 位七段数码管的初值：

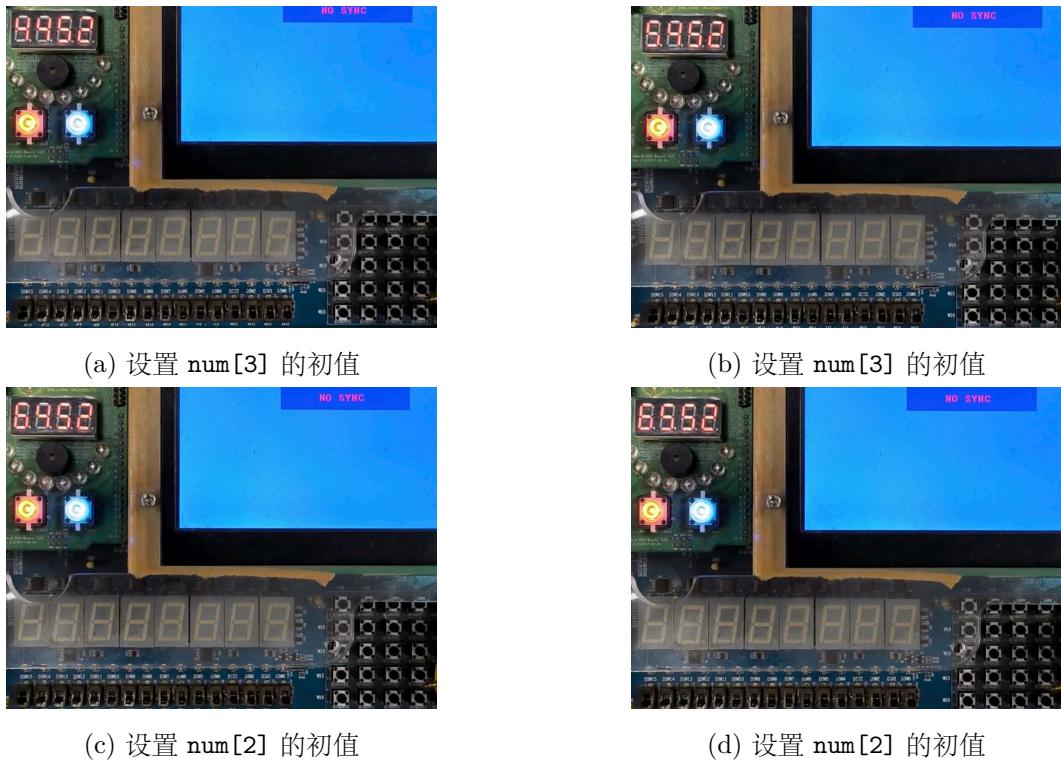


图 4: 设置 4 个寄存器的初值

使用 SW[15] 控制将 4 位七段数码管的数据输出到 LED 灯:



图 5: 使用 SW[15] 控制输出

可见 LED 灯显示的二进制数为 0110011101010011，与数码管显示的十六进制数 6753 相同。

4.3 任务 3: 设计主板 8 位数码管驱动模块

4.3.1 工程设计

1. 新建工程，工程名称用 SEGP2S，Top Level Source Type 用 HDL。
2. 用行为描述设计。

- 利用实验 12 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]。
- 编写 SegmentDecoder 模块进行段码译码。

```

1  module SegmentDecoder(
2      input wire [3:0] hex,
3      output reg [7:0] Segment
4  );
5      always @(*) begin
6          case (hex)
7              4'b0000: Segment <= 8'b11000000; // 0
8              4'b0001: Segment <= 8'b11111001; // 1
9              4'b0010: Segment <= 8'b10100100; // 2
10             4'b0011: Segment <= 8'b10110000; // 3
11             4'b0100: Segment <= 8'b10011001; // 4
12             4'b0101: Segment <= 8'b10010010; // 5
13             4'b0110: Segment <= 8'b10000010; // 6
14             4'b0111: Segment <= 8'b11111000; // 7
15             4'b1000: Segment <= 8'b10000000; // 8
16             4'b1001: Segment <= 8'b10010000; // 9
17             4'b1010: Segment <= 8'b10001000; // A
18             4'b1011: Segment <= 8'b10000011; // B
19             4'b1100: Segment <= 8'b11000110; // C
20             4'b1101: Segment <= 8'b10100001; // D
21             4'b1110: Segment <= 8'b10000110; // E
22             4'b1111: Segment <= 8'b10001110; // F
23             default: Segment <= 8'b00000000; // X
24         endcase
25     end
26 endmodule

```

- 利用 7 个 SLReg8b 模块、1 个 SLReg9b 模块和 1 个触发器，设计主板 8 位数码管驱动模块 SEGP2S。

```

1 `timescale 1ns / 1ps
2 module SEGP2S(
3     input wire clk,
4     input wire [15:0] SW,
5     output wire SEG_CLK,
6     output wire SEG_CLR,
7     output wire SEG_EN,
8     output wire SEG_DT
9 );
10    wire [31:0] num;
11    wire [64:0] Segment;
12    wire [63:0] num_disp;
13    wire [31:0] clk_div;
14    wire start, finish, SL;
15    wire ser_out;

```

```

16   assign ser_out = Segment[64];
17   clkdiv m0(.clk(clk),
18             .rst(1'b0),
19             .clkdiv(clk_div));
20   RegCounter4b reg0(.clk(clk), .btn(SW[0]), .CLR(SW[14]), .num(num[3:0]));
21   RegCounter4b reg1(.clk(clk), .btn(SW[1]), .CLR(SW[14]), .num(num[7:4]));
22   RegCounter4b reg2(.clk(clk), .btn(SW[2]), .CLR(SW[14]), .num(num[11:8]));
23   RegCounter4b reg3(.clk(clk), .btn(SW[3]), .CLR(SW[14]), .num(num[15:12]));
24   RegCounter4b reg4(.clk(clk), .btn(SW[4]), .CLR(SW[14]), .num(num[19:16]));
25   RegCounter4b reg5(.clk(clk), .btn(SW[5]), .CLR(SW[14]), .num(num[23:20]));
26   RegCounter4b reg6(.clk(clk), .btn(SW[6]), .CLR(SW[14]), .num(num[27:24]));
27   RegCounter4b reg7(.clk(clk), .btn(SW[7]), .CLR(SW[14]), .num(num[31:28]));
28   SegmentDecoder dec0(.hex(num[3:0]), .Segment(num_disp[7:0]));
29   SegmentDecoder dec1(.hex(num[7:4]), .Segment(num_disp[15:8]));
30   SegmentDecoder dec2(.hex(num[11:8]), .Segment(num_disp[23:16]));
31   SegmentDecoder dec3(.hex(num[15:12]), .Segment(num_disp[31:24]));
32   SegmentDecoder dec4(.hex(num[19:16]), .Segment(num_disp[39:32]));
33   SegmentDecoder dec5(.hex(num[23:20]), .Segment(num_disp[47:40]));
34   SegmentDecoder dec6(.hex(num[27:24]), .Segment(num_disp[55:48]));
35   SegmentDecoder dec7(.hex(num[31:28]), .Segment(num_disp[63:56]));
36   assign SEG_CLK = clk | finish;
37   assign SEG_CLR = 1'b1;
38   assign SEG_DT = ser_out;
39   assign SEG_EN = 1'b1;
40   assign finish = (Segment[63:0] == 64'b1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111);
41   SR_latch sr0(.S(start & finish), .R(~finish), .Q(SL));
42   Load_Gen load0(.clk(clk), .clk_1ms(clk_div[17]), .btn_in(SW[15]),
43                 .Load_out(start));
44   SLReg9b s10(.clk(clk), .S_L(SL), .s_in(1'b1), .p_in({num_disp[7:0], 1'b0}),
45                 .Q(Segment[8:0]));
46   SLReg8b s11(.clk(clk), .S_L(SL), .s_in(Segment[8]), .p_in(num_disp[15:8]),
47                 .Q(Segment[16:9]));
48   SLReg8b s12(.clk(clk), .S_L(SL), .s_in(Segment[16]), .p_in(num_disp[23:16]),
49                 .Q(Segment[24:17]));
50   SLReg8b s13(.clk(clk), .S_L(SL), .s_in(Segment[24]), .p_in(num_disp[31:24]),
51                 .Q(Segment[32:25]));
52   SLReg8b s14(.clk(clk), .S_L(SL), .s_in(Segment[32]), .p_in(num_disp[39:32]),
53                 .Q(Segment[40:33]));
54   SLReg8b s15(.clk(clk), .S_L(SL), .s_in(Segment[40]), .p_in(num_disp[47:40]),
55                 .Q(Segment[48:41]));
56   SLReg8b s16(.clk(clk), .S_L(SL), .s_in(Segment[48]), .p_in(num_disp[55:48]),
57                 .Q(Segment[56:49]));
58   SLReg8b s17(.clk(clk), .S_L(SL), .s_in(Segment[56]), .p_in(num_disp[63:56]),
59                 .Q(Segment[64:57]));
60
61 endmodule

```

4.3.2 仿真

设计激励代码，对驱动模块进行仿真。

仿真之前，在顶层模块中添加输出 num_reg 用于显示寄存器的值。

仿真激励代码：

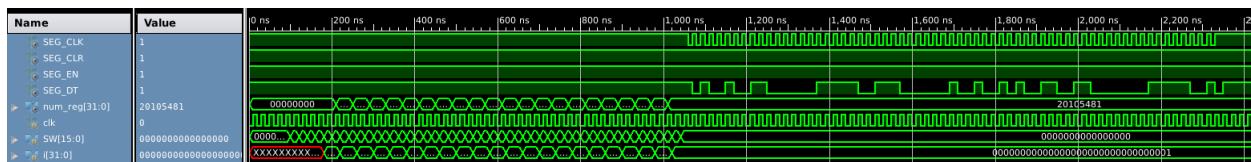
```
1 integer i;
2 initial begin
3     SW = 0;
4     clk = 0;
5     #100;
6
7     // Initialize Registers
8     SW[14] = 1'b1; #20;
9     SW[7:0] = 8'b11111111; #20;
10    SW[7:0] = 8'b00000000; #20;
11
12    // Set Registers
13    SW[14] = 1'b0; #20;
14    // 20105481
15    for (i = 0; i < 2; i = i + 1) begin
16        SW[7] = 1'b1; #20;
17        SW[7] = 1'b0; #20;
18    end
19    for (i = 0; i < 1; i = i + 1) begin
20        SW[5] = 1'b1; #20;
21        SW[5] = 1'b0; #20;
22    end
23    for (i = 0; i < 5; i = i + 1) begin
24        SW[3] = 1'b1; #20;
25        SW[3] = 1'b0; #20;
26    end
27    for (i = 0; i < 4; i = i + 1) begin
28        SW[2] = 1'b1; #20;
29        SW[2] = 1'b0; #20;
30    end
31    for (i = 0; i < 8; i = i + 1) begin
32        SW[1] = 1'b1; #20;
33        SW[1] = 1'b0; #20;
34    end
35    for (i = 0; i < 1; i = i + 1) begin
36        SW[0] = 1'b1; #20;
37        SW[0] = 1'b0; #20;
38    end
39
40    SW[15] = 1'b1; #20;
41    SW[15] = 1'b0;
```

```

42 end
43
44 always begin
45   clk = 0; #10;
46   clk = 1; #10;
47 end

```

仿真波形：



仿真激励代码中，先将寄存器初始化为 0。随后将寄存器初值设为学号后 8 位 20105481。拨动 SW[15] 启用串行输入，可见输出 SEG_DT 和 SEG_CLK 能正确地将寄存器的值移入 8 位数码管驱动模块。

4.3.3 上板验证

可以拨动 8 个开关 SW[7:0] 来修改每个数码管的数值。



(a) 主板七段数码管显示



(b) 将主板七段数码管设成显示学号后 8 位



(c) 将 SW[14] 设为 1 把主板七段数码管每一位清零

可见上板结果符合预期。

5 实验结果分析

实验结果在前一节已经分析，符合实验要求。

6 讨论与心得

本次实验工作量较大，代码较长，比较考验细心程度。没有遇到仿真结果和上板结果不一致的情况，总体还算顺利。