

浙江大学

本科实验报告

课程名称:	数字逻辑电路设计
姓 名:	姜雨童
学 院:	计算机科学与技术学院
专 业:	计算机科学与技术
邮 箱:	3220103450@zju.edu.cn
QQ 号:	1369218489
电 话:	19550103468
指导教师:	洪奇军
报告日期:	2023 年 12 月 11 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：_____ 实验十一：寄存器和寄存器传输设计 _____

学生姓名：_____ 姜雨童 _____ 学号：_____ 3220103450 _____ 同组学生姓名：_____ / _____

实验地点：_____ 紫金港东四 509 室 _____ 实验日期：_____ 2023 _____ 年 _____ 12 _____ 月 _____ 5 _____ 日

一、操作方法与实验步骤

实验目的：

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

实验任务：

基于 ALU 的数据传输应用设计
(Mode1: ALU 运算输出控制, Mode2: 数据传输控制)

工程实现：

新建 HDL 工程 **MyALUTrans**

添加如下模块 (均为以前实验使用过的模块或以前实验的产物):

- ALU 模块
- 4 位 4 选 1 模块
- 防抖动模块
- 显示模块

新建 Verilog 文件 **Top**, 并设置为顶层模块

```
module TOP(input wire clk,  
            input wire [2:0]BTN_Y,  
            // 012:ABC  
            input wire [15:0] SW,
```

```

// SW[15] = 0:Mode1, SW[4:3]:ALU, SW[2]:A+-, SW[1]:B+-
// SW[15] = 1:Mode2, SW[6:5]:SelectBus
output wire BTN_X,
output wire [3:0] AN,
// A B rst_alu C
output wire [7:0] SEGMENT,
output wire seg_clk,
output wire seg_clrn,
output wire seg_sout,
output wire SEG_PEN
);

reg [11:0] num; // ABC
wire [3:0] rst_alu;
wire [3:0] Bus;
wire [2:0] btn_out;
wire [3:0] A1;
wire [3:0] B1;
wire [3:0] A2;
wire [3:0] B2;
wire [3:0] C2;
wire Co;
wire [31:0] clk_div;

assign BTN_X = 1'b0;
clkdiv m0(clk,0,clk_div);
pbdebounce m1(clk_div[17],BTN_Y[0],btn_out[0]);
pbdebounce m2(clk_div[17],BTN_Y[1],btn_out[1]);
pbdebounce m3(clk_div[17],BTN_Y[2],btn_out[2]);

AddSub4b m4(.A(num[3:0]),.B(4'b0001),.Ctrl(SW[2]),.S(A1)); //A+-
AddSub4b m5(.A(num[7:4]),.B(4'b0001),.Ctrl(SW[1]),.S(B1)); //B+-
Mux4to1b4 m6(.I0(num[3:0]),.I1(num[7:4]),
               .I2(num[11:8]),.I3(4'b0),.s(SW[6:5]),.o(Bus)); // SelectBus

assign A2 = (SW[15]==1'b0)? A1:Bus;
assign B2 = (SW[15]==1'b0)? B1:Bus;
assign C2 = (SW[15]==1'b0)? rst_alu:Bus;

always @(posedge btn_out[0]) num[3:0] = A2;
always @(posedge btn_out[1]) num[7:4] = B2;
always @(posedge btn_out[2]) num[11:8] = C2;

ALU m7(.A(num[3:0]),.B(num[7:4]),.S(SW[4:3]),.O(rst_alu),.Co(Co));
disp_num m8(clk,{num[3:0],num[7:4],rst_alu,

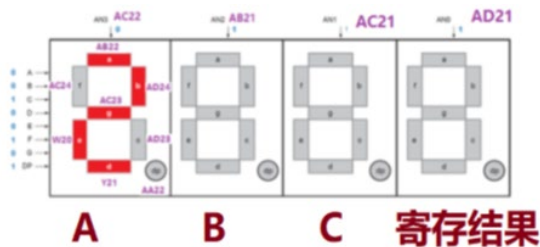
```

```

        num[11:8]},4'b0000,4'b0000,1'b0,AN,SEGMENT);
SSeg7_Dev m9(.clk(clk),.rst(1'b0),.Start(clk_div[20]),.flash(clk_div[25]),
        .Hexts({12'hFFF,Bus,num[3:0],num[7:4],rst_alu,num[11:8]}),.SW0(SW[0]),
        .point(8'h00),.LES(8'h00),.seg_clk(seg_clk),
        .seg_clrn(seg_clrn),.seg_sout(seg_sout),.SEG_PEN(SEG_PEN) );
endmodule

```

工程模式说明：



Mode1: ALU 运算输出控制（模式由 SW[15]决定）

SW[0]: SSeg7_Dev 模块中控制图形/文本显示（本工程使用文本显示）

SW[1]: 按下 btn[1]控制 B 自增/自减（零加一减）

SW[2]: 按下 btn[0]控制 A 自增/自减（零加一减）

SW[4:3]: 选择 ALU 模式（00:ADD; 01:Sub; 10:AND; 11:OR）

按下 btn[2]把 C 赋值给寄存器

（备注：我的工程中四位显示数据分别为 A，B，rst_alu，C，因此按下 btn[2]实现把 rst_alu 赋值给寄存器 C）

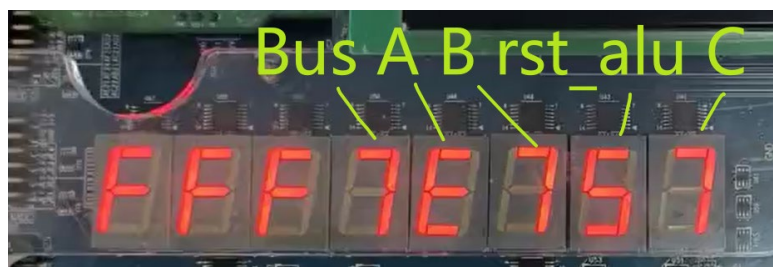
Mode2: 数据传输控制

SW[6:5]: SelectBus（总线，上板验证时，显示在 A 左边）上放的是什么数据

00: 总线数据选择 A; 01: 选择 B; 10: 选择 C

按 btn[0]将数据赋值给 A（btn[1]-B; btn[2]-C）

（备注：因为上述命名问题，这里做实验的时候被绕进去了。因此我的工程按下 btn[2]后不是将第三位 ALU 的计算结果赋值给总线，而是将寄存器 C 的值赋值给总线）



上板验证：

新建引脚约束文件：

```
NET"clk"LOC = AC18 | IOSTANDARD = LVCMOS18;
```

```
NET"SEGMENT[0]"LOC = AB22 | IOSTANDARD = LVCMOS33;
NET"SEGMENT[1]"LOC = AD24 | IOSTANDARD = LVCMOS33;
NET"SEGMENT[2]"LOC = AD23 | IOSTANDARD = LVCMOS33;
NET"SEGMENT[3]"LOC = Y21 | IOSTANDARD = LVCMOS33;
NET"SEGMENT[4]"LOC = W20 | IOSTANDARD = LVCMOS33;
NET"SEGMENT[5]"LOC = AC24 | IOSTANDARD = LVCMOS33;
NET"SEGMENT[6]"LOC = AC23 | IOSTANDARD = LVCMOS33;
NET"SEGMENT[7]"LOC = AA22 | IOSTANDARD = LVCMOS33;
NET"AN[0]"LOC = AD21 | IOSTANDARD = LVCMOS33;
NET"AN[1]"LOC = AC21 | IOSTANDARD = LVCMOS33;
NET"AN[2]"LOC = AB21 | IOSTANDARD = LVCMOS33;
NET"AN[3]"LOC = AC22 | IOSTANDARD = LVCMOS33;
```

#七段码串行移位接口

```
NET"seg_clk" LOC = M24 | IOSTANDARD = LVCMOS33;
NET"seg_clrn" LOC = M20 | IOSTANDARD = LVCMOS33;
NET"seg_sout" LOC = L24 | IOSTANDARD = LVCMOS33;
NET"SEG_PEN" LOC = R18 | IOSTANDARD = LVCMOS33;
```

btn 按钮接口

```
NET"BTN_Y[0]" LOC = V18 | IOSTANDARD = LVCMOS18;
NET"BTN_Y[0]"clock_dedicated_route = false;
NET"BTN_Y[1]" LOC = V19 | IOSTANDARD = LVCMOS18;
NET"BTN_Y[1]"clock_dedicated_route = false;
NET"BTN_Y[2]" LOC = V14 | IOSTANDARD = LVCMOS18;
NET"BTN_Y[2]"clock_dedicated_route = false;
NET"BTN_X" LOC = W16 | IOSTANDARD = LVCMOS18;
```

```
NET"SW[0]" LOC = AA10 | IOSTANDARD = LVCMOS15;
NET"SW[1]" LOC = AB10 | IOSTANDARD = LVCMOS15;
NET"SW[2]" LOC = AA13 | IOSTANDARD = LVCMOS15;
NET"SW[3]" LOC = AA12 | IOSTANDARD = LVCMOS15;
NET"SW[4]" LOC = Y13 | IOSTANDARD = LVCMOS15;
NET"SW[5]" LOC = Y12 | IOSTANDARD = LVCMOS15;
NET"SW[6]" LOC = AD11 | IOSTANDARD = LVCMOS15;
NET"SW[7]" LOC = AD10 | IOSTANDARD = LVCMOS15;
NET"SW[8]" LOC = AE10 | IOSTANDARD = LVCMOS15;
NET"SW[9]" LOC = AE12 | IOSTANDARD = LVCMOS15;
NET"SW[10]" LOC = AF12 | IOSTANDARD = LVCMOS15;
NET"SW[11]" LOC = AE8 | IOSTANDARD = LVCMOS15;
NET"SW[12]" LOC = AF8 | IOSTANDARD = LVCMOS15;
NET"SW[13]" LOC = AE13 | IOSTANDARD = LVCMOS15;
NET"SW[14]" LOC = AF13 | IOSTANDARD = LVCMOS15;
NET"SW[15]" LOC = AF10 | IOSTANDARD = LVCMOS15;
```

二、实验结果与分析

上板验证结果：

Model1:
SW[15]=0, ALU 运算输出控制

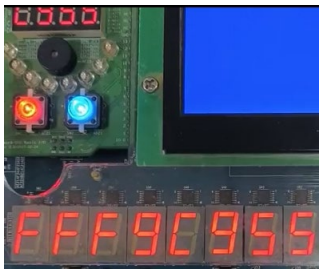
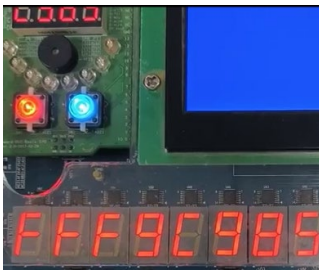
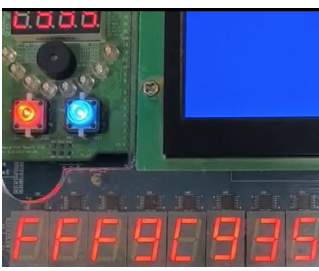
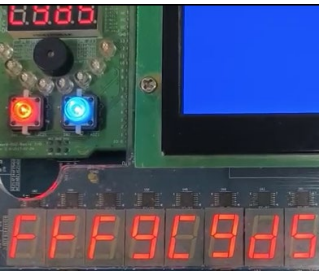
SW[2]: 按下 btn[0]控制 A 自增/自减（零加一减）

SW[2]	按下 btn[0]控制 A 自增		
0			
	按下 btn[0]控制 A 自减		
1			

SW[1]: 按下 btn[1]控制 B 自增/自减（零加一减）

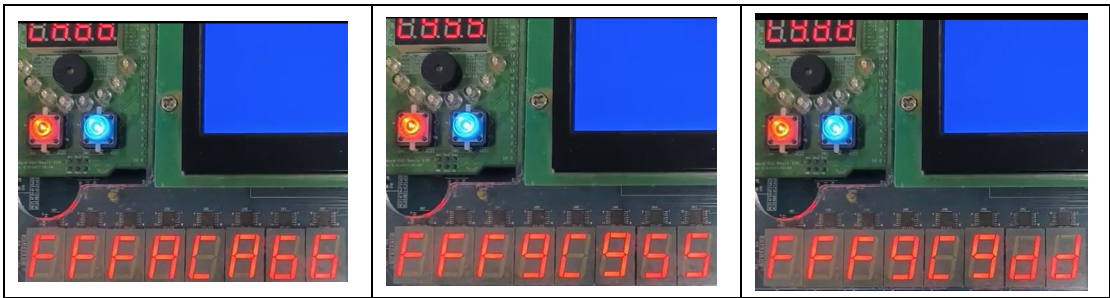
SW[1]	按下 btn[1]控制 B 自增		
0			
	按下 btn[1]控制 B 自减		
1			

SW[4:3]: 选择 ALU 模式 (00:ADD; 01:Sub; 10:AND; 11:OR)

SW[4:3]		SW[4:3]	
00 ADD		10 AND	
01 Sub		11 OR	

按下 btn[2]把 C 赋值给寄存器

(备注: 我的工程中四位显示数据分别为 A, B, rst_alu, C, 因此按下 btn[2]实现把 rst_alu 赋值给寄存器 C)

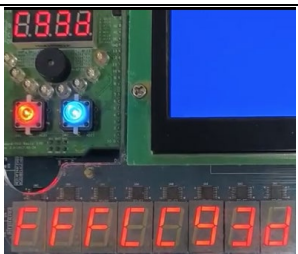
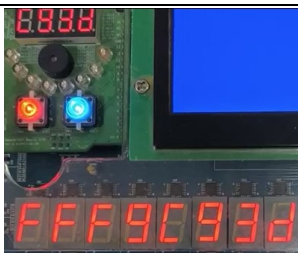
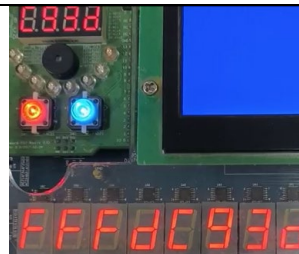


Mode2:

SW[15]=1, 数据传输控制

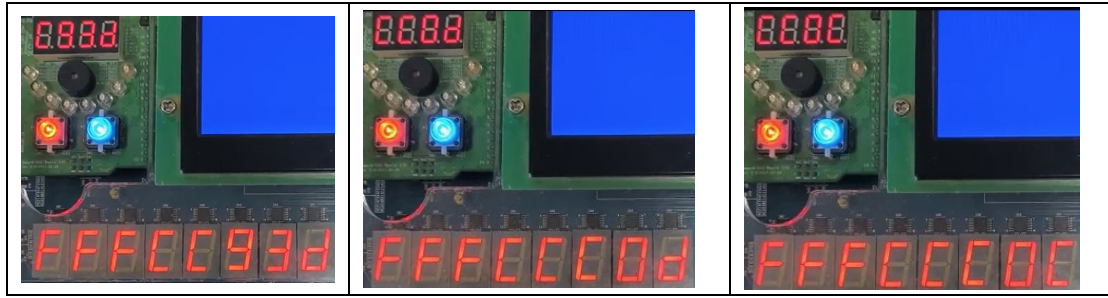
SW[6:5]: SelectBus (总线, 上板验证时, 显示在 A 左边) 上放的是什么数据

00: 总线数据选择 A; 01: 选择 B; 10: 选择寄存器 C

SW[6:5]	00	01	10
			

按 btn[0]将数据赋值给 A (btn[1]-B; btn[2]-C)

(备注: 因为上述命名问题, 这里做实验的时候被绕进去了。因此我的工程按下 btn[2]后不是将第三位 ALU 的计算结果赋值给总线, 而是将寄存器 C 的值赋值给总线)



三、讨论、心得

经过前几次实验的熟悉，本次实验进行地较为顺畅，没有遇到太多问题。但是在第一次上板的时候，发现 Mode1 模式下，无法将 `rst_alu` 的值赋给寄存器 C。排查问题后，发现是因为刚开始这一行划线处写的是 `C:Bus;`，也就是 Mode1 模式下，C2 为现在自身的值，Mode2 模式下，C2 为 Bus 上的值。因此，后续根据 `btn[2]` 按钮将 C2 赋值给 C 时，也就没有将 `rst_alu` 的值传递过去。修改后，程序模块可以正常执行。

```
assign BTN_X = 1'b0;
clkdiv m0(clk,0,clk_div);
pbdebounce m1(clk_div[17],BTN_Y[0],btn_out[0]);
pbdebounce m2(clk_div[17],BTN_Y[1],btn_out[1]);
pbdebounce m3(clk_div[17],BTN_Y[2],btn_out[2]);

AddSub4b m4(.A(num[3:0]),.B(4'b0001),.Ctrl(SW[2]),.S(A1)); //A+-
AddSub4b m5(.A(num[7:4]),.B(4'b0001),.Ctrl(SW[1]),.S(B1)); //B+-
Mux4to1b4 m6(.IO(num[3:0]),.I1(num[7:4]),.I2(num[11:8]),.I3(4'b0),.s(SW[6:5]),.o(Bus)); // SelectBus

assign A2 = (SW[15]==1'b0)? A1:Bus;
assign B2 = (SW[15]==1'b0)? B1:Bus;
assign C2 = (SW[15]==1'b0)? rst_alu:Bus;

always @(posedge btn_out[0]) num[3:0] = A2;
always @(posedge btn_out[1]) num[7:4] = B2;
always @(posedge btn_out[2]) num[11:8] = C2;
```

另外一个问题则是在写报告时仔细整理才发现的。（如上述备注所述，）原要求是，Mode2 模式下，赋值与被赋值的部分是 A、B、C（对应我程序中的 A、B、`rst_alu`）。但是我在编写寄存器传递模块时，被自己的命名绕进去了，选择了寄存器（对应我程序中的 C），因此实际程序的功能执行略有不同。

浙江大学

本科实验报告

课程名称:	数字逻辑电路设计
姓 名:	姜雨童
学 院:	计算机科学与技术学院
专 业:	计算机科学与技术
邮 箱:	3220103450@zju.edu.cn
QQ 号:	1369218489
电 话:	19550103468
指导教师:	洪奇军
报告日期:	2023 年 12 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： 综合

实验项目名称： 实验十二：计数器、定时器设计与应用

学生姓名： 姜雨童 学号： 3220103450 同组学生姓名： /

实验地点： 紫金港东四 509 室 实验日期： 2023 年 12 月 12 日

一、操作方法与实验步骤

实验目的：

掌握同步四位二进制计数器 74LS161 的工作原理和设计方法
掌握时钟/定时器的工作原理与设计方法

试验任务：

任务一：采用行为描述设计同步四位二进制计数器 74LS161
任务二：基于 74LS161 设计时钟应用

任务一：采用行为描述设计同步四位二进制计数器 74LS161

新建 HDL 工程 My74LS161

新建 Verilog 文件 My74LS161，根据 74LS161 功能表编写 Verilog 代码

输 入					输 出				
\overline{CR}	\overline{LD}	CT_P	CT_T	CP	$D_3D_2D_1D_0$	$Q_3Q_2Q_1Q_0$			
0	×	×	×	×	×	×	×	×	×
1	0	×	×	↑	$d_3d_2d_1d_0$	$d_3d_2d_1d_0$			
1	1	0	1	×	×	×	×	×	保持
1	1	×	0	×	×	×	×	×	保持
1	1	1	1	↑	×	×	×	×	计数

```

module My74LS161( input wire CR,
    input wire Load,
    input wire CTp,
    input wire CTt,
    input wire CP,
    input wire [3:0] D,
    output reg [3:0] Q,
    output wire Co
);
    always@(posedge CP or negedge CR) begin
        if(!CR) Q[3:0] <= 4'b0000;
        else if(!Load) Q[3:0] <= D[3:0];
        else if(CTp & CTt) Q[3:0] <= Q[3:0] + 1'b1;
    end
    assign Co = CTt & (&Q);

Endmodule

```

其中 CR 为异步清零，Load 为同步置位
输入激励代码进行波形仿真，代码如下：

```

module My74LS161_sim;

    // Inputs
    reg CR;
    reg Load;
    reg CTp;
    reg CTt;
    reg CP;
    reg [3:0] D;

    // Outputs
    wire [3:0] Q;
    wire Co;

    // Instantiate the Unit Under Test (UUT)
    My74LS161 uut (
        .CR(CR),
        .Load(Load),
        .CTp(CTp),
        .CTt(CTt),
        .CP(CP),
        .D(D),
        .Q(Q),
        .Co(Co)
    );

endmodule

```

```

initial begin
    // Initialize Inputs
    CR = 0;
    Load = 0;
    CTp = 0;
    CTt = 0;
    D = 0;

    // Wait 100 ns for global reset to finish
    #100;
    CR = 1;
    Load = 1;
    D = 4'b1100;
    CTt = 0;
    CTp = 0;

    #30; CR = 0;
    #20; CR = 1;
    #10; Load = 0;
    #30; CTt = 1; CTp = 1;
    #10; Load = 1;

    #510; CR = 0;
    #20; CR = 1;
    #500;

end

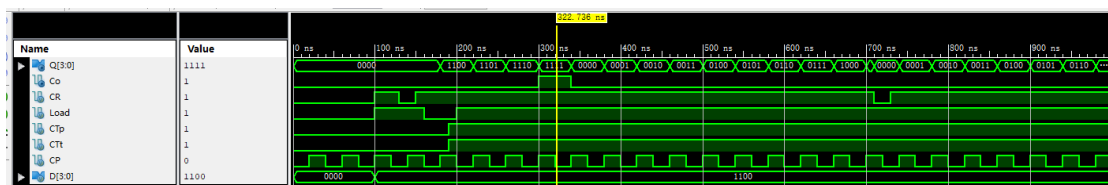
always begin
    CP = 0; #20;
    CP = 1; #20;

end

endmodule

```

仿真波形输出如图所示：



任务二：基于 74LS161 设计时钟应用

新建 HDL 工程 MyClock

用结构化描述设计计时器(使用 60 进制和 24 进制计数器,实现 24 小时内时间的实时显示):

- 调用 My74LS161 模块
- 调用分频模块,用 100ms 作为分的驱动时钟 (clk_100ms 和 clkdiv)
- 调用显示模块 (disp_num 和 SSeg7_Dev)

参考代码如下:

```
module Clock(input wire clk,
             output wire [3:0] AN,
             output wire [7:0] SEGMENT,
             output wire seg_clk,
             output wire seg_clrn,
             output wire seg_sout,
             output wire SEG_PEN
);
    wire [15:0] displaynumber;
    wire [31:0] div;
    wire clk_100ms;

    counter_100ms m0(clk,clk_100ms);
    clkdiv mm0(clk,1'b0,div[31:0]);

    // minites
    My74LS161 m1(.CR(1'b1),.Load(~(displaynumber[3]&displaynumber[0])),
                .CTt(1'b1),.CTp(1'b1),.CP(clk_100ms),
                .D(4'b0000),.Q(displaynumber[3:0]) );

    My74LS161 m2(.CR(1'b1),
    .Load(~(displaynumber[6]&displaynumber[4]&displaynumber[3]&displaynumber[0])),
    .CTt(displaynumber[3]&displaynumber[0]),.CTp(1'b1),.CP(clk_100ms),
    .D(4'b0000),.Q(displaynumber[7:4]) );

    // hours
    My74LS161 m3(.CR(1'b1),
    .Load(~((displaynumber[11]&displaynumber[8]&displaynumber[6]&
displaynumber[4]&displaynumber[3]&displaynumber[0])|
(displaynumber[13]&displaynumber[9]&displaynumber[8]&displaynumber[6]&
displaynumber[4]&displaynumber[3]&displaynumber[0]))),
    .CTt(displaynumber[6]&displaynumber[4]&displaynumber[3]&displaynumber[0]),
    .CTp(1'b1),.CP(clk_100ms),
    .D(4'b0000),.Q(displaynumber[11:8]) );

    My74LS161 m4(.CR(1'b1),
    .Load(~(displaynumber[13]&displaynumber[9]&displaynumber[8]&displaynumber[6]
&displaynumber[4]&displaynumber[3]&displaynumber[0])),
    .CTt(displaynumber[11]&displaynumber[8]&displaynumber[6]&
displaynumber[4]&displaynumber[3]&displaynumber[0]),
    .CTp(1'b1),.CP(clk_100ms),.D(4'b0000),.Q(displaynumber[15:12]) );
```

```

// display
disp_num m5(.clk(clk),.HEXS(displaynumber),.LES(4'b0000),
.points(4'b0100),.RST(1'b0),.AN(AN),.Segment(SEGMENT) );
SSeg7_Dev m6(.clk(clk),.rst(1'b0),.Start(div[20]),.SW0(1'b1),
.flash(1'b1),.Hexs({16'b0,displaynumber[15:0]}),
.point(8'b0000_0100),.LES(8'b1111_0000),.seg_clk(seg_clk),
.seg_clrn(seg_clrn),.seg_sout(seg_sout),.SEG_PEN(SEG_PEN) );

```

Endmodule

（备注：代码中加粗部分为实验中 debug 修改后的代码，具体分析会在报告第三部分的讨论心得中给出。加粗部分：时钟个位 m3 的 Load 部分和时钟十位 m4 的 CTt 部分）

上板验证，引脚约束文件如下：

```

NET"clk"LOC = AC18 | IOSTANDARD = LVCMOS18;

NET"SEGMENT[0]"LOC = AB22 | IOSTANDARD = LVCMOS33;
NET"SEGMENT[1]"LOC = AD24 | IOSTANDARD = LVCMOS33;
NET"SEGMENT[2]"LOC = AD23 | IOSTANDARD = LVCMOS33;
NET"SEGMENT[3]"LOC = Y21 | IOSTANDARD = LVCMOS33;
NET"SEGMENT[4]"LOC = W20 | IOSTANDARD = LVCMOS33;
NET"SEGMENT[5]"LOC = AC24 | IOSTANDARD = LVCMOS33;
NET"SEGMENT[6]"LOC = AC23 | IOSTANDARD = LVCMOS33;
NET"SEGMENT[7]"LOC = AA22 | IOSTANDARD = LVCMOS33;
NET"AN[0]"LOC = AD21 | IOSTANDARD = LVCMOS33;
NET"AN[1]"LOC = AC21 | IOSTANDARD = LVCMOS33;
NET"AN[2]"LOC = AB21 | IOSTANDARD = LVCMOS33;
NET"AN[3]"LOC = AC22 | IOSTANDARD = LVCMOS33;

#七段码串行移位接口
NET"seg_clk" LOC = M24 | IOSTANDARD = LVCMOS33;
NET"seg_clrn" LOC = M20 | IOSTANDARD = LVCMOS33;
NET"seg_sout" LOC = L24 | IOSTANDARD = LVCMOS33;
NET"SEG_PEN" LOC = R18 | IOSTANDARD = LVCMOS33;

```

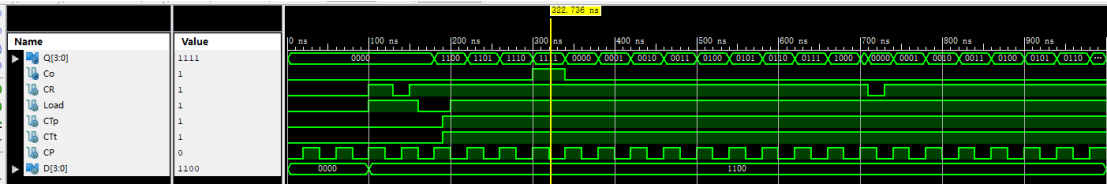
二、实验结果与分析

74LS161 仿真结果分析：

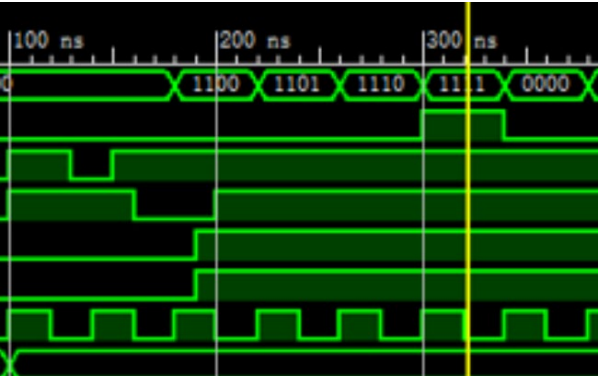
下图为仿真结果：从上到下分别为 Q[3:0]、Co、CR、Load、CTp、CTt、CP、D[3:0]

D 的取值：0000、1100

Q 的值：0000、1100、1101、1110、1111、0000、0001、0010、0011、0100、0101...

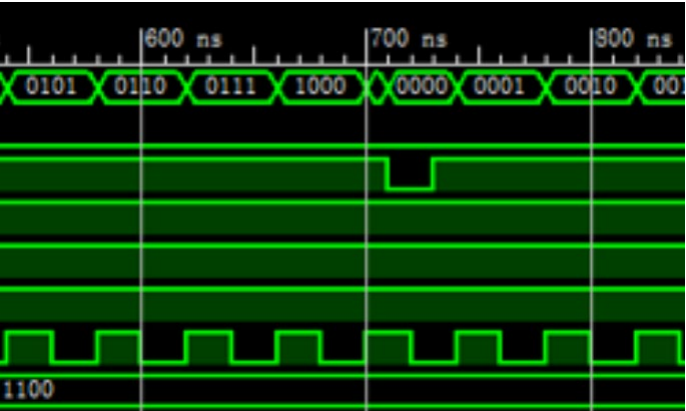


其中 CR 为异步清零，Load 为同步置位。下面根据局部大图进行分析。
观察图 1 不难发现：当 Load 为 0 时且 CP（时钟分频信号）处于上升沿时，Q 被置为 1100，之后 CTt 和 CTp 置 1，在下个 CP 上升沿开始计时功能。同时在计时器重置清零前，输出 Co



(图 1)

而图 2 验证了 CR 的异步清零功能：CR 置 0 时，计时器清零。



(图 2)

上板验证时钟：

表 1 分针 59 进位&时针 23 进位验证		

- 时钟个位出现 9 时，分钟计时的同时，时钟十位也在快速计时，直到时钟个位再次进位重置变成 0（具体行为呈现如表）：



原因在于编写 m4（时钟十位数字）的 CTt 时，只考虑了时钟个位数字为 9 的条件（displaynumber 的 11 位和 8 位），却没考虑进位时分钟应为 59（类似上一个问题）。

- 时钟过 23 进位后，直接出现 04 而不是 00：

原因是时钟个位只考虑了个位出现 9 进位时的情况，而没有考虑时钟为 23 时的进位情况。因此，时钟出现 23 时，十位重置为 0，个位则继续进位为 4。修改办法是在 m3 的 Load 部分加入 23:59 的判定条件。特别注意的是，Load 信号清零时会对 m3 进行重置，而我们传入的信号是 1 不是 0，要把全部内容用括号括起再进行取反，同时注意中间选择的逻辑运算符是“与”还是“或”。

浙江大学

本科实验报告

课程名称:	数字逻辑电路设计
姓 名:	姜雨童
学 院:	计算机科学与技术学院
专 业:	计算机科学与技术
邮 箱:	3220103450@zju.edu.cn
QQ 号:	1369218489
电 话:	19550103468
指导教师:	洪奇军
报告日期:	2023 年 12 月 22 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：_____ 实验十三：移位寄存器设计与应用 _____

学生姓名：_____ 姜雨童 _____ 学号：_____ 3220103450 _____ 同组学生姓名：_____ / _____

实验地点：_____ 紫金港东四 509 室 _____ 实验日期：_____ 2023 _____ 年 _____ 12 _____ 月 _____ 19 _____ 日

一、操作方法与实验步骤

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

实验任务：

设计 8 位带并行输入的右移移位寄存器
设计跑马灯应用

任务一：设计 8 位带并行输入的右移移位寄存器

新建 HDL 工程 **ShiftReg8b**

用结构化描述设计编写 Verilog 代码，代码如下：

```
module shift_reg(  
    input wire clk, S_L, s_in, p_in,  
    output wire q  
);  
    wire NS_L;  
    wire and1;  
    wire and2;  
    wire d;  
  
    INV INV1(.I(S_L),.O(NS_L) );
```

```

        AND2 AD1(.I0(s_in),.I1(NS_L),.O(and1) );
        AND2 AD2(.I0(p_in),.I1(S_L),.O(and2) );
        OR2 OR1(.I0(and1),.I1(and2),.O(d) );
        FD fd1(.D(d),.C(clk),.Q(q) );
    endmodule

module shift_reg8b(
    input wire clk, S_L, s_in,
    input wire [7:0] p_in,
    output wire [7:0] Q
);
    shift_reg s0(.clk(clk),.S_L(S_L),.s_in(s_in),.p_in(p_in[7]),.q(Q[7]));
    shift_reg s1(.clk(clk),.S_L(S_L),.s_in(Q[7]),.p_in(p_in[6]),.q(Q[6]));
    shift_reg s2(.clk(clk),.S_L(S_L),.s_in(Q[6]),.p_in(p_in[5]),.q(Q[5]));
    shift_reg s3(.clk(clk),.S_L(S_L),.s_in(Q[5]),.p_in(p_in[4]),.q(Q[4]));
    shift_reg s4(.clk(clk),.S_L(S_L),.s_in(Q[4]),.p_in(p_in[3]),.q(Q[3]));
    shift_reg s5(.clk(clk),.S_L(S_L),.s_in(Q[3]),.p_in(p_in[2]),.q(Q[2]));
    shift_reg s6(.clk(clk),.S_L(S_L),.s_in(Q[2]),.p_in(p_in[1]),.q(Q[1]));
    shift_reg s7(.clk(clk),.S_L(S_L),.s_in(Q[1]),.p_in(p_in[0]),.q(Q[0]));
endmodule

```

进行波形仿真，激励代码如下：

```

module shift_reg8b_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_reg8b 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;
    end
endmodule

```


新建 HDL 工程 MyMarquee

- 调用 ShiftReg8b 模块

- 调用 ShiftReg8b 模块
- 调用分频模块，用 1s 作为移位寄存器的驱动时钟
- 调用显示模块
- 调用 CreateNumber 模块

```

always@(posedge clk_1s)begin // unsure part
    if(SW[0]) btn_out[0] <= 1'b1;
    else btn_out[0] <= 1'b0;
    if(SW[1]) btn_out[1] <= 1'b1;
    else btn_out[1] <= 1'b0;
end
CreateNumber m0({2'b0,btn_out},num);

clk_1s m1(clk,clk_1s);
shift_reg8b m2(clk_1s,SW[2],s_in,num[7:0],LED);
always@(posedge clk_1s)begin
    if(SW[4]) s_in <= LED[0];
    else s_in <= SW[3];
end

disp_num m3(clk,{num[7:0],LED[7:0]},4'b0000,4'b0000,1'b0,AN,SEGMENT);

Endmodule

```

下载到 sword 板上进行验证，引脚约束文件如下：

```

NET"clk"LOC = AC18 | IOSTANDARD = LVCMOS18;

NET"SW[0]"LOC = AA10 | IOSTANDARD = LVCMOS15;
NET"SW[1]"LOC = AB10 | IOSTANDARD = LVCMOS15;
NET"SW[2]"LOC = AA13 | IOSTANDARD = LVCMOS15;
NET"SW[3]"LOC = AA12 | IOSTANDARD = LVCMOS15;
NET"SW[4]"LOC = Y13 | IOSTANDARD = LVCMOS15;

NET"SEGMENT[0]"LOC = AB22 | IOSTANDARD = LVCMOS33;
NET"SEGMENT[1]"LOC = AD24 | IOSTANDARD = LVCMOS33;
NET"SEGMENT[2]"LOC = AD23 | IOSTANDARD = LVCMOS33;
NET"SEGMENT[3]"LOC = Y21 | IOSTANDARD = LVCMOS33;
NET"SEGMENT[4]"LOC = W20 | IOSTANDARD = LVCMOS33;
NET"SEGMENT[5]"LOC = AC24 | IOSTANDARD = LVCMOS33;
NET"SEGMENT[6]"LOC = AC23 | IOSTANDARD = LVCMOS33;
NET"SEGMENT[7]"LOC = AA22 | IOSTANDARD = LVCMOS33;
NET"AN[0]"LOC = AD21 | IOSTANDARD = LVCMOS33;
NET"AN[1]"LOC = AC21 | IOSTANDARD = LVCMOS33;
NET"AN[2]"LOC = AB21 | IOSTANDARD = LVCMOS33;
NET"AN[3]"LOC = AC22 | IOSTANDARD = LVCMOS33;

NET"LED[0]"LOC=W23 | IOSTANDARD=LVCMOS33;
NET"LED[1]"LOC=AB26 | IOSTANDARD=LVCMOS33;
NET"LED[2]"LOC=Y25 | IOSTANDARD=LVCMOS33;

```

```

NET"LED[3]"LOC=AA23 | IOSTANDARD=LVCMOS33;
NET"LED[4]"LOC=Y23 | IOSTANDARD=LVCMOS33;
NET"LED[5]"LOC=Y22 | IOSTANDARD=LVCMOS33;
NET"LED[6]"LOC=AE21 | IOSTANDARD=LVCMOS33;
NET"LED[7]"LOC=AF24 | IOSTANDARD=LVCMOS33;

```

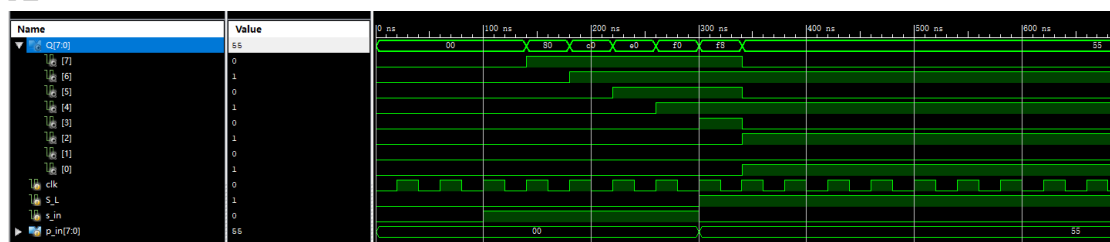
引脚说明：

- 用 sw[0]和 sw[1]作为 regA 和 regB 的按键自增控制输入
- sw[2]=1，并行输入，将{RegA,RegB}赋给移位寄存器
- sw[2]=0，串行/循环右移移位
- sw[4]作为移位寄存器的模式选择：
 - sw[4]=0，串行右移，串行输入值为 sw[3]
 - sw[4]=1，循环右移
- 8 位的移位寄存器的值用 LED 灯表示

二、实验结果与分析

8 位带并行输入的右移移位寄存器仿真结果：

从上到下依次为 Q[7:0]（输出）、clk（时钟信号）、S_L（并行输入命令）、s_in（串行输入）、p_in[7:0]（并行输入）



不难看出，刚开始是串行输入 0，故输出一直为 0；后续串行输入 1，故输出持续增大。300ns 时变更为并行输入，340ns 后，输出一直保持并行输入的 0x55。右移移位寄存器能够正常工作。

跑马灯上板验证：



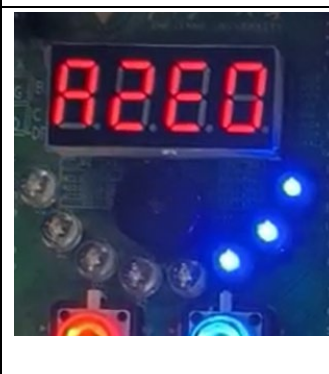
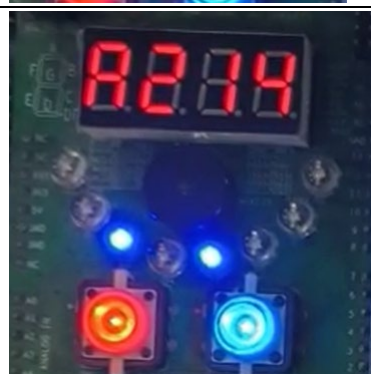
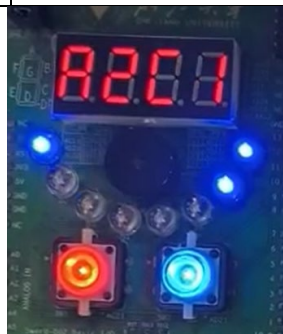
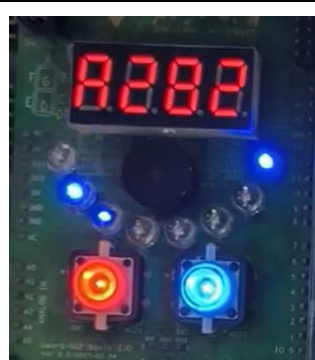
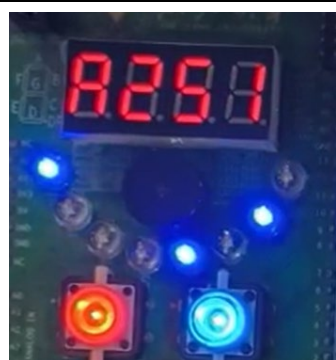
sw[2]=1, 并行输入, 将{RegA,RegB}赋给移位寄存器

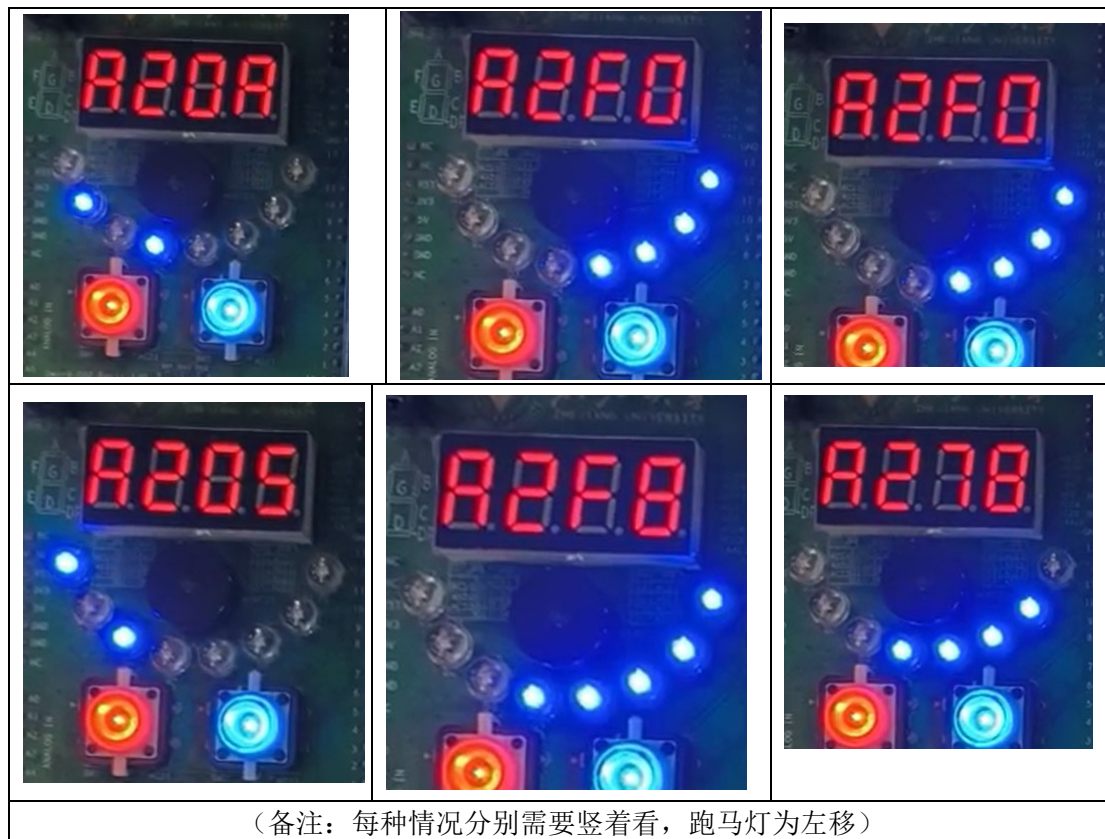


sw[2]=0, 串行/循环右移移位

sw[4]=0, 串行右移, 串行输入值为 sw[3]

sw[4]=1, 循环右移





三、讨论、心得

这个实验我没有做选做部分，而前面跑马灯的部分相对简单。唯一遇到的问题是刚开始时把移位寄存器的 0 当最低位，7 当最高位，因此制作出的是左移移位寄存器，当时的代码和仿真结果如下：

(上板时因为引脚约束文件，是左移寄存器)

```

module shift_reg(
    input wire clk, S_L, s_in, p_in,
    output wire q

);
    wire NS_L;
    wire and1;
    wire and2;
    wire d;

    INV INV1(.I(S_L),.O(NS_L) );
    AND2 AD1(.I0(s_in),.I1(NS_L),.O(and1) );
    AND2 AD2(.I0(p_in),.I1(S_L),.O(and2) );
    OR2 OR1(.I0(and1),.I1(and2),.O(d) );
    FD fd1(.D(d),.C(clk),.Q(q) );

endmodule

```



```

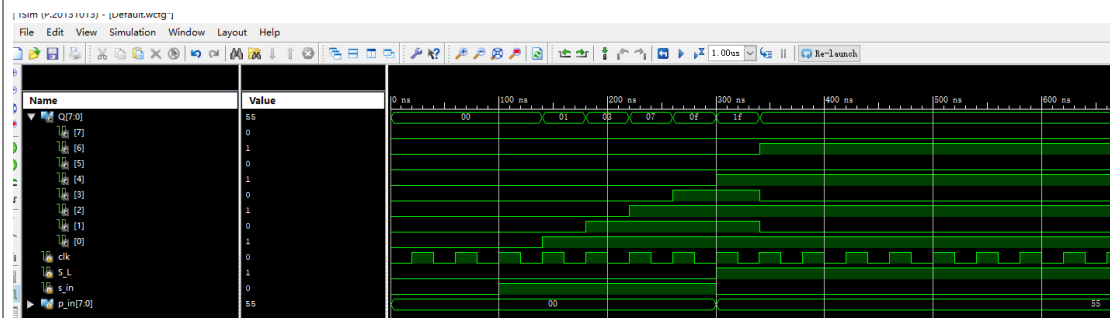
module shift_reg8b(
    input wire clk, S_L, s_in,
    input wire [7:0] p_in,
    output wire [7:0] Q

);

    shift_reg s0(.clk(clk),.S_L(S_L),.s_in(s_in),.p_in(p_in[0]),.q(Q[0]));
    shift_reg s1(.clk(clk),.S_L(S_L),.s_in(Q[0]),.p_in(p_in[1]),.q(Q[1]));
    shift_reg s2(.clk(clk),.S_L(S_L),.s_in(Q[1]),.p_in(p_in[2]),.q(Q[2]));
    shift_reg s3(.clk(clk),.S_L(S_L),.s_in(Q[2]),.p_in(p_in[3]),.q(Q[3]));
    shift_reg s4(.clk(clk),.S_L(S_L),.s_in(Q[3]),.p_in(p_in[4]),.q(Q[4]));
    shift_reg s5(.clk(clk),.S_L(S_L),.s_in(Q[4]),.p_in(p_in[5]),.q(Q[5]));
    shift_reg s6(.clk(clk),.S_L(S_L),.s_in(Q[5]),.p_in(p_in[6]),.q(Q[6]));
    shift_reg s7(.clk(clk),.S_L(S_L),.s_in(Q[6]),.p_in(p_in[7]),.q(Q[7]));

endmodule

```



四、个人生活照片

