浙江大学

本科实验报告

课程名称：	数字逻辑电路设计
姓　　名：	金祺书
学　　院：	计算机科学与技术学院
专　　业：	计算机科学与技术
指导教师：	洪奇军
报告日期：	2024 年 5 月 16 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：金祺书 学号：3230104248 同组学生姓名：蒋翼泽

实验地点：紫金港东四 509 室 实验日期：2024 年 5 月 16 日

一、操作方法与实验步骤

Vivado 新建工程，导入先前 lab 所制作的 clkdiv、pbdebounce、AddSub4b、Mux4to1b4、ALU、DisplayNumber 模块，顶层代码 Top 如下：

```
01  `timescale 1ns / 1ps
02
03  module Top(
04      input clk,
05      input [3:0] BTN_Y,
06      input [15:0] SW,
07      output BTN_X,
08      output[3:0]AN,
09      output[7:0] SEGMENT
10  );
11
12      wire [31:0] my_clkdiv;
13      wire [2:0] btn_out;
14      reg  [11:0] num;
15      wire [3:0] A1, A2, B1, B2, C1, C2; // C1 maybe useless
16      wire [3:0] mux_out;
17      wire Co;
18      wire [3:0] ALU_res;
19
20      /* SW[1:0] to control if the counter for A or B is reversal */
21      wire A_Ctrl = SW[0];
22      wire B_Ctrl = SW[1];
23      /* SW[3:2] to choose the mode of the ALU */
24      wire [1:0] ALU_Ctrl = SW[3:2];
25      /* SW[5:4] to choose from A B C and 0 */
26      /* 00 for A; 01 for B; 10 for C; 11 for 0 */
27      wire [1:0] Trans_select = SW[5:4];
28
```

```
29      wire [3:0] reg_A_val = num[ 3: 0];
30      wire [3:0] reg_B_val = num[ 7: 4];
31      wire [3:0] reg_C_val = num[11: 8];
32
33      assign BTN_X = 1'b0;
34
35      clkdiv m0(.clk(clk), .rst(1'b0), .div_res(my_clkdiv));
36
37      pbdebounce
m1(.clk(my_clkdiv[17]), .button(BTN_Y[0]), .pbreg(btn_out[0]));
38      pbdebounce
m2(.clk(my_clkdiv[17]), .button(BTN_Y[1]), .pbreg(btn_out[1]));
39      pbdebounce
m3(.clk(my_clkdiv[17]), .button(BTN_Y[2]), .pbreg(btn_out[2]));
40
41      AddSub4b
m4(.A(reg_A_val), .B(4'b0001), .Ctrl(A_Ctrl), .S(A1));
42      AddSub4b
m5(.A(reg_B_val), .B(4'b0001), .Ctrl(B_Ctrl), .S(B1));
43
44      Mux4to1b4
m6(.D0(reg_A_val), .D1(reg_B_val), .D2(reg_C_val), .D3(4'b0000),
45                                  .S(Trans_select), .Y(mux_out));
46
47      /* ALU module implemented in Lab8 */
48      /* A/B    : operands */
49      /* S       : select the operation on ALU  */
50      /* C        : result of ALU */
51      /* Co       : Carry bit */
52      ALU
m7(.A(reg_A_val), .B(reg_B_val), .res(ALU_res), .Cout(Co), .op(ALU_Ct
rl)); // (Co) may be useless
53
54      DisplayNumber m8(.clk(clk), .hexs({reg_A_val, reg_B_val,
ALU_res, reg_C_val}),
55
    .LEs(4'b0000), .points(4'b0000), .rst(1'b0), .AN(AN),
56                            .SEGMENT(SEGMENT));
57
58      /* Your code here */
59      // SW[15]: 0 for ALU mode, 1 for Trans mode.
60      assign A2 = (1'b0 == SW[15]) ? A1 : mux_out;
61      assign B2 = (1'b0 == SW[15]) ? B1 : mux_out;
62      assign C2 = (1'b0 == SW[15]) ? ALU_res : mux_out;
```

```
63
64      always@(posedge btn_out[0]) num[3:0] = A2;
65      always@(posedge btn_out[1]) num[7:4] = B2;
66      always@(posedge btn_out[2]) num[11:8] = C2;
67      /******************/
68
69  endmodule
```

主要实现的功能如下：

（1）SW[15] = 0 ALU 运算输出模式

SW[0] 控制 A 的增/减；SW[1] 控制 B 的增/减

SW[3:2] 控制 ALU 运算类型

按下 btn[0] 用 A 自增/自减的值更新 A

按下 btn[1] 用 B 自增/自减的值更新 B

按下 btn[2] 用 ALU 运算结果更新 C

（2）SW[15] = 1 数据传输控制模式

SW[5:4] 传输选择信号，00 选择 A，01 选择 B，10 选择 C，11 选择常数 0

btn[0], btn[1], btn[2] 分别为三个寄存器 A, B, C 的 load 信号。如在按下 btn[0] 时用当前总线上数据对 A 的值进行更新

其中代码段：

```
60      assign A2 = (1'b0 == SW[15]) ? A1 : mux_out;
61      assign B2 = (1'b0 == SW[15]) ? B1 : mux_out;
62      assign C2 = (1'b0 == SW[15]) ? ALU_res : mux_out;
```

在控制数据传输模式与运算输出模式，判断 SW[15]是否为 0，是则 A 保持原运算数 A1 不变，B 保持原运算数 B1 不变，C 保持运算结果 ALU_res 不变；否则将寄存器 mux_out 中的值赋值给相对应的变量。

代码段：

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

将需要显示的结果 A2，B2，C2 赋值给 num 以便显示。

通过约束文件：

```
01  # Filename: constraints_labB.xdc
02  ## Constraints file for LabB
03
04  # Main clock
```

```
05 set_property PACKAGE_PIN AC18 [get_ports clk]
06 set_property IOSTANDARD LVCMOS18 [get_ports clk]
07
08 create_clock -period 10.000 -name clk [get_ports "clk"]
09
10 # Switches as inputs
11 set_property PACKAGE_PIN AA10 [get_ports {SW[0]}]
12 set_property PACKAGE_PIN AB10 [get_ports {SW[1]}]
13 set_property PACKAGE_PIN AA13 [get_ports {SW[2]}]
14 set_property PACKAGE_PIN AA12 [get_ports {SW[3]}]
15 set_property PACKAGE_PIN Y13 [get_ports {SW[4]}]
16 set_property PACKAGE_PIN Y12 [get_ports {SW[5]}]
17 set_property PACKAGE_PIN AD11 [get_ports {SW[15]}]
18 set_property IOSTANDARD LVCMOS15 [get_ports {SW[0]}]
19 set_property IOSTANDARD LVCMOS15 [get_ports {SW[1]}]
20 set_property IOSTANDARD LVCMOS15 [get_ports {SW[2]}]
21 set_property IOSTANDARD LVCMOS15 [get_ports {SW[3]}]
22 set_property IOSTANDARD LVCMOS15 [get_ports {SW[4]}]
23 set_property IOSTANDARD LVCMOS15 [get_ports {SW[5]}]
24 set_property IOSTANDARD LVCMOS15 [get_ports {SW[15]}]
25
26 # Key as inputs
27 set_property PACKAGE_PIN W16 [get_ports BTN_X]
28 set_property IOSTANDARD LVCMOS18 [get_ports BTN_X]
29 set_property PACKAGE_PIN V18 [get_ports {BTN_Y[3]}]
30 set_property IOSTANDARD LVCMOS18 [get_ports {BTN_Y[3]}]
31 set_property PACKAGE_PIN V19 [get_ports {BTN_Y[2]}]
32 set_property IOSTANDARD LVCMOS18 [get_ports {BTN_Y[2]}]
33 set_property PACKAGE_PIN V14 [get_ports {BTN_Y[1]}]
34 set_property IOSTANDARD LVCMOS18 [get_ports {BTN_Y[1]}]
35 set_property PACKAGE_PIN W14 [get_ports {BTN_Y[0]}]
36 set_property IOSTANDARD LVCMOS18 [get_ports {BTN_Y[0]}]
37
38 set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets BTN*]
39
40 # Arduino-Segment & AN
41 set_property PACKAGE_PIN AD21 [get_ports {AN[0]}]
42 set_property PACKAGE_PIN AC21 [get_ports {AN[1]}]
43 set_property PACKAGE_PIN AB21 [get_ports {AN[2]}]
44 set_property PACKAGE_PIN AC22 [get_ports {AN[3]}]
45 set_property PACKAGE_PIN AB22 [get_ports {SEGMENT[0]}]
46 set_property PACKAGE_PIN AD24 [get_ports {SEGMENT[1]}]
47 set_property PACKAGE_PIN AD23 [get_ports {SEGMENT[2]}]
48 set_property PACKAGE_PIN Y21 [get_ports {SEGMENT[3]}]
```

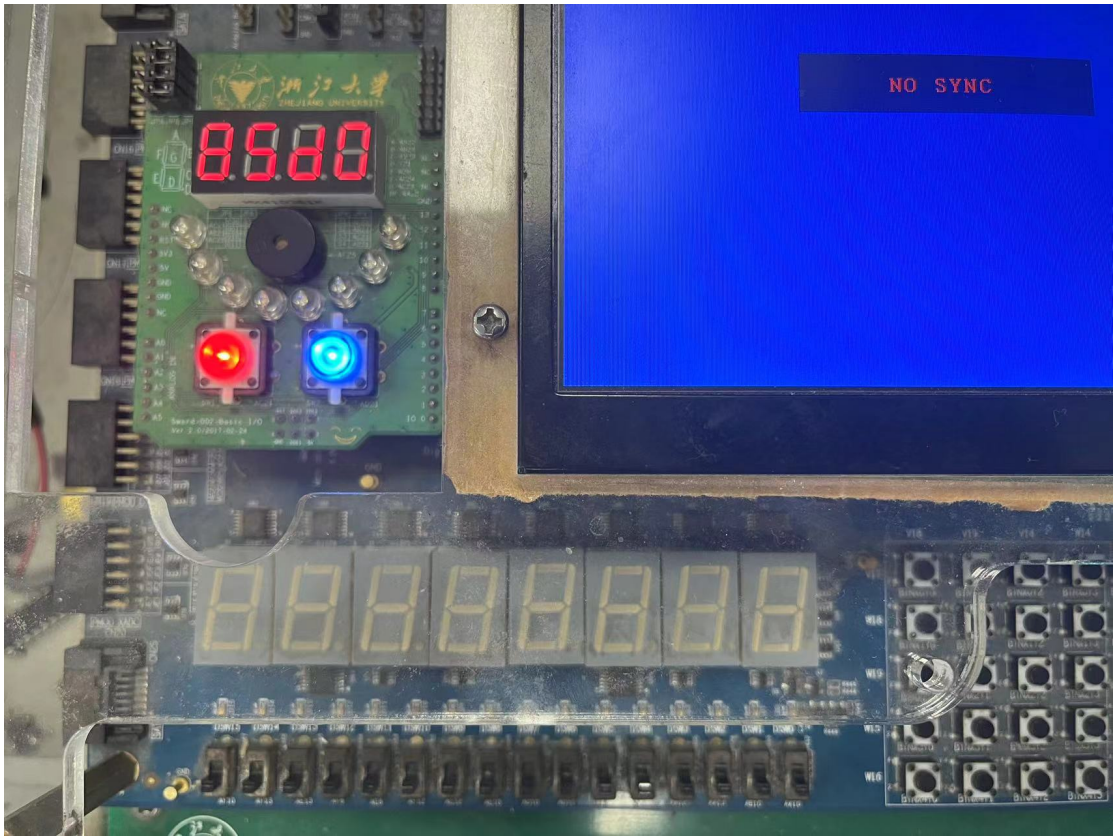
```
set_property PACKAGE_PIN W20 [get_ports {SEGMENT[4]}]
set_property PACKAGE_PIN AC24 [get_ports {SEGMENT[5]}]
set_property PACKAGE_PIN AC23 [get_ports {SEGMENT[6]}]
set_property PACKAGE_PIN AA22 [get_ports {SEGMENT[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[7]}]
```

生成比特流下板验证。

二、实验结果与分析

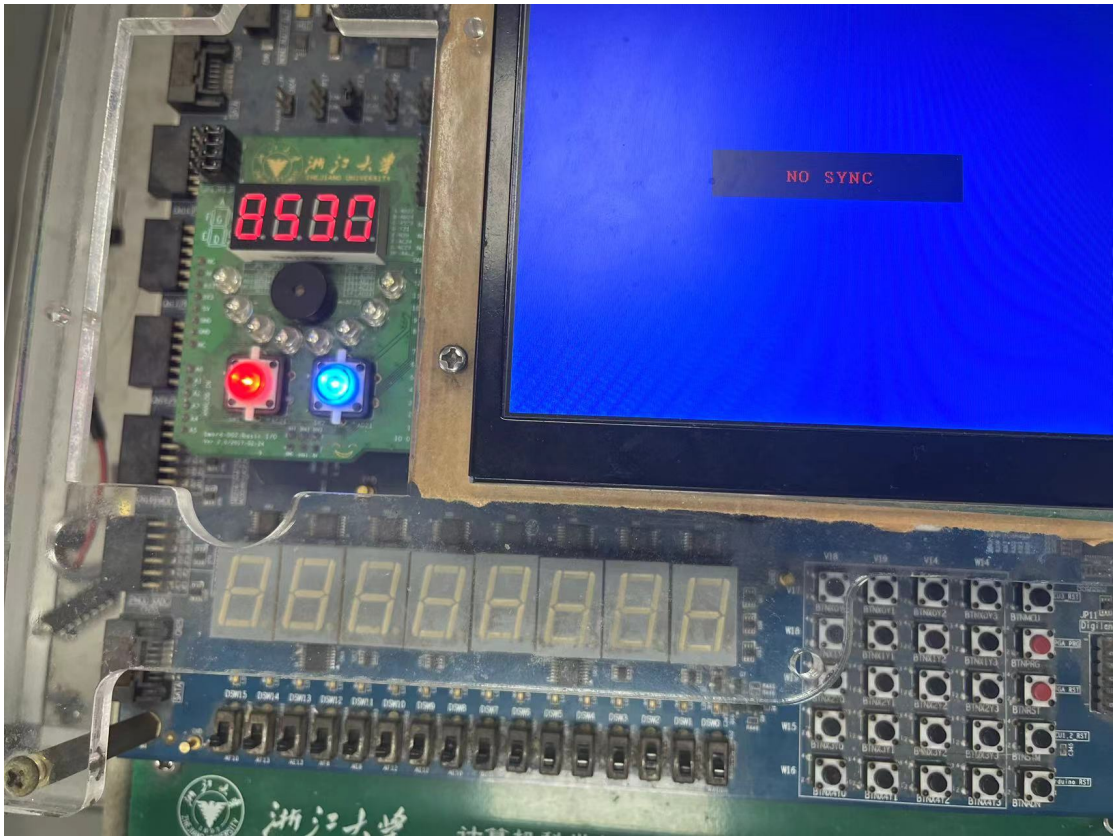
下板结果如下：

（1）初始情况（ALU运算模式，为加法）：

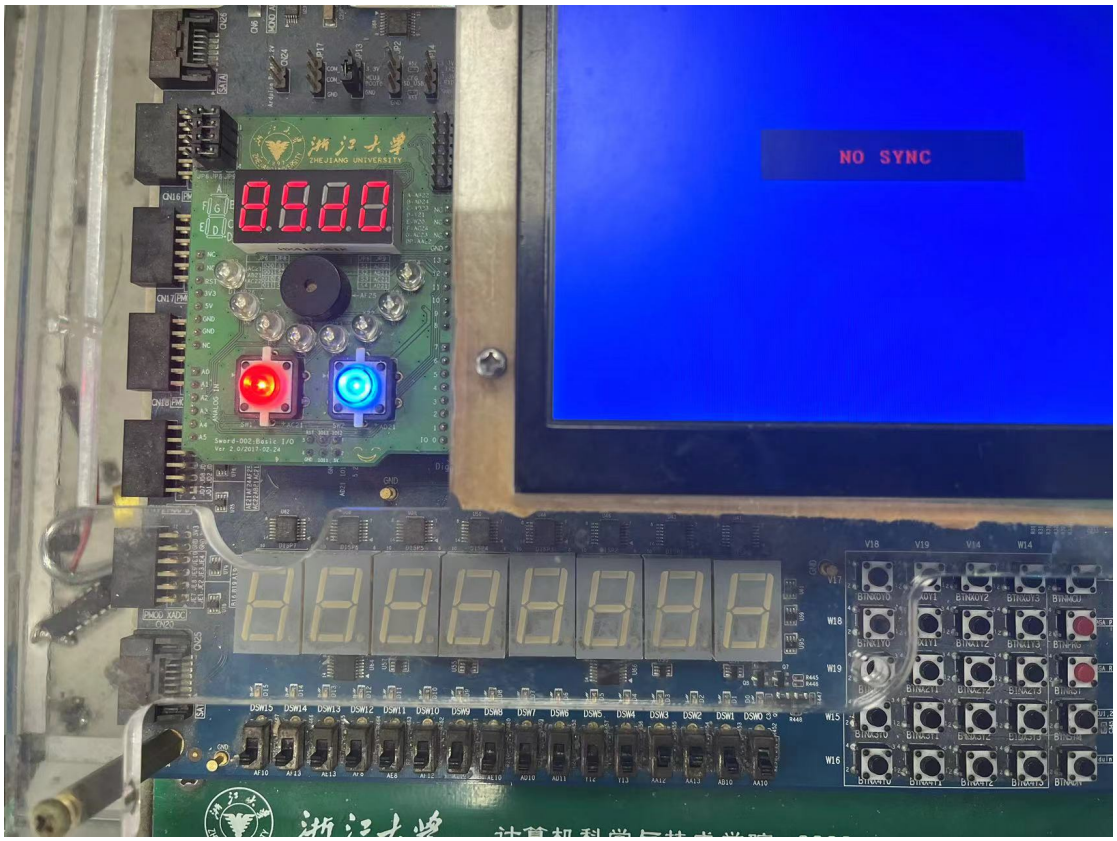


（2）改变 ALU 运算方式

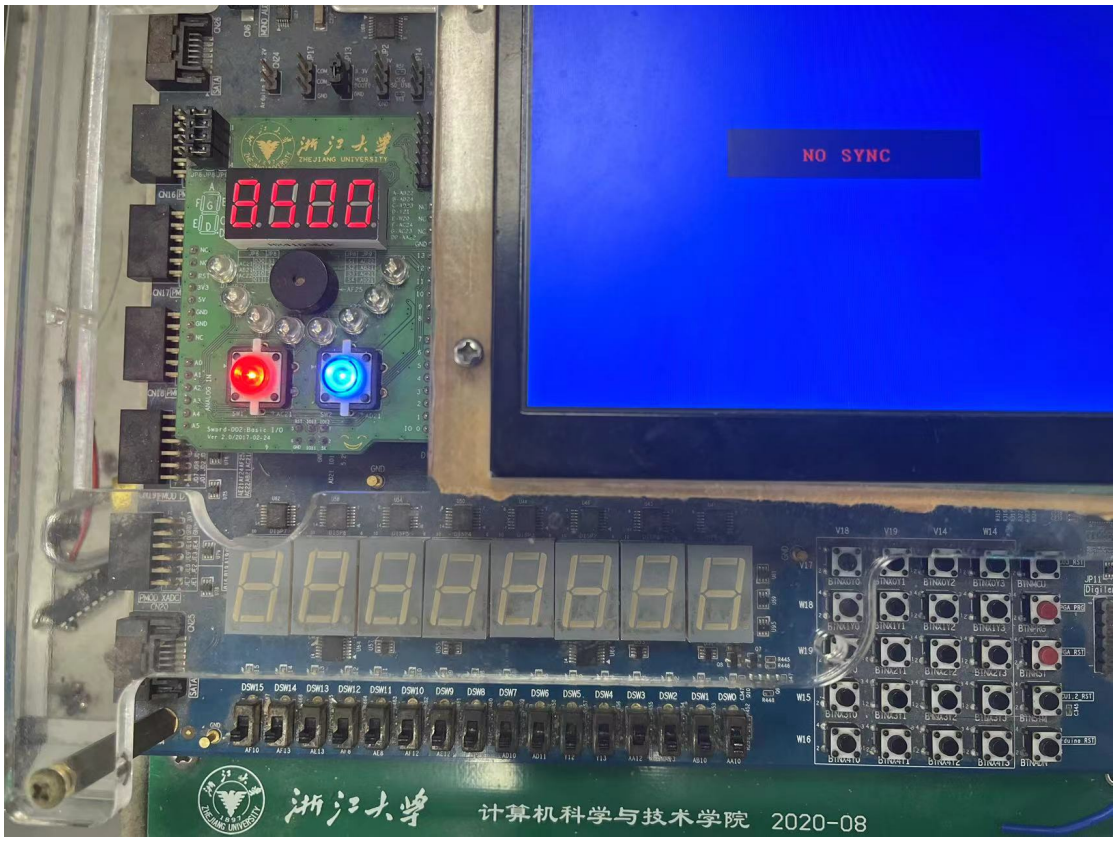
1）减法



2）或运算

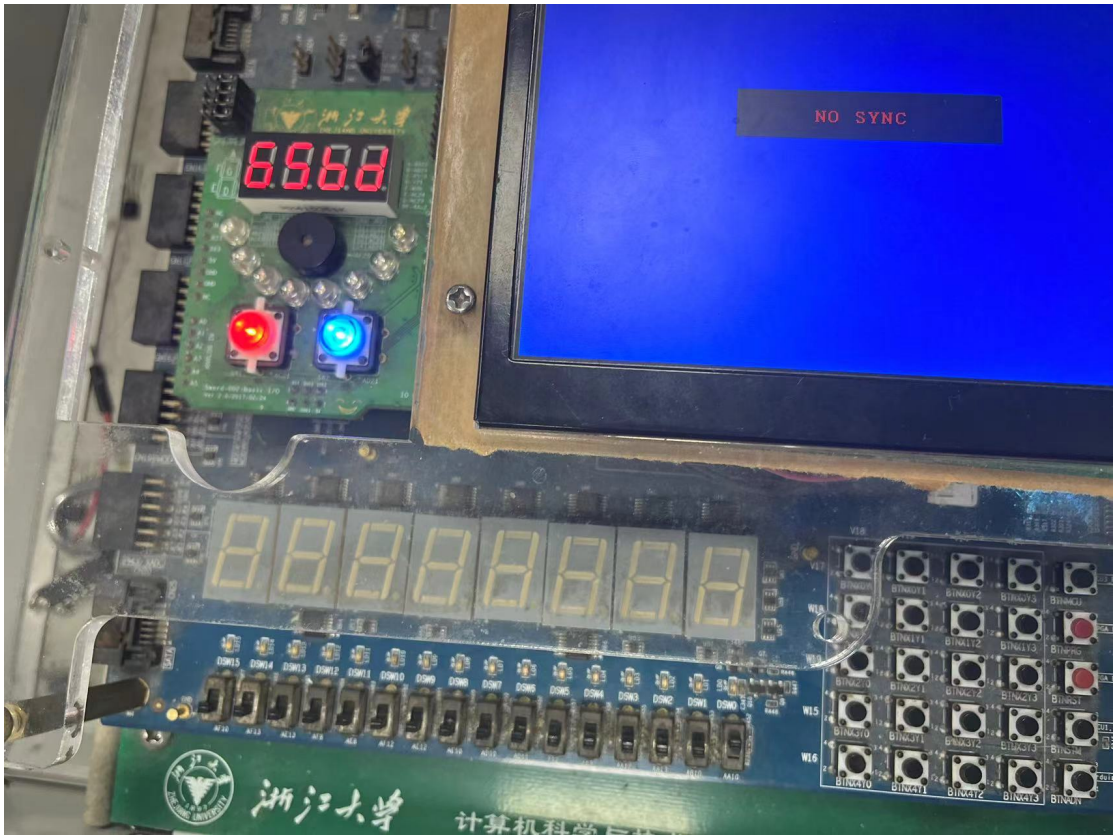


3）与运算

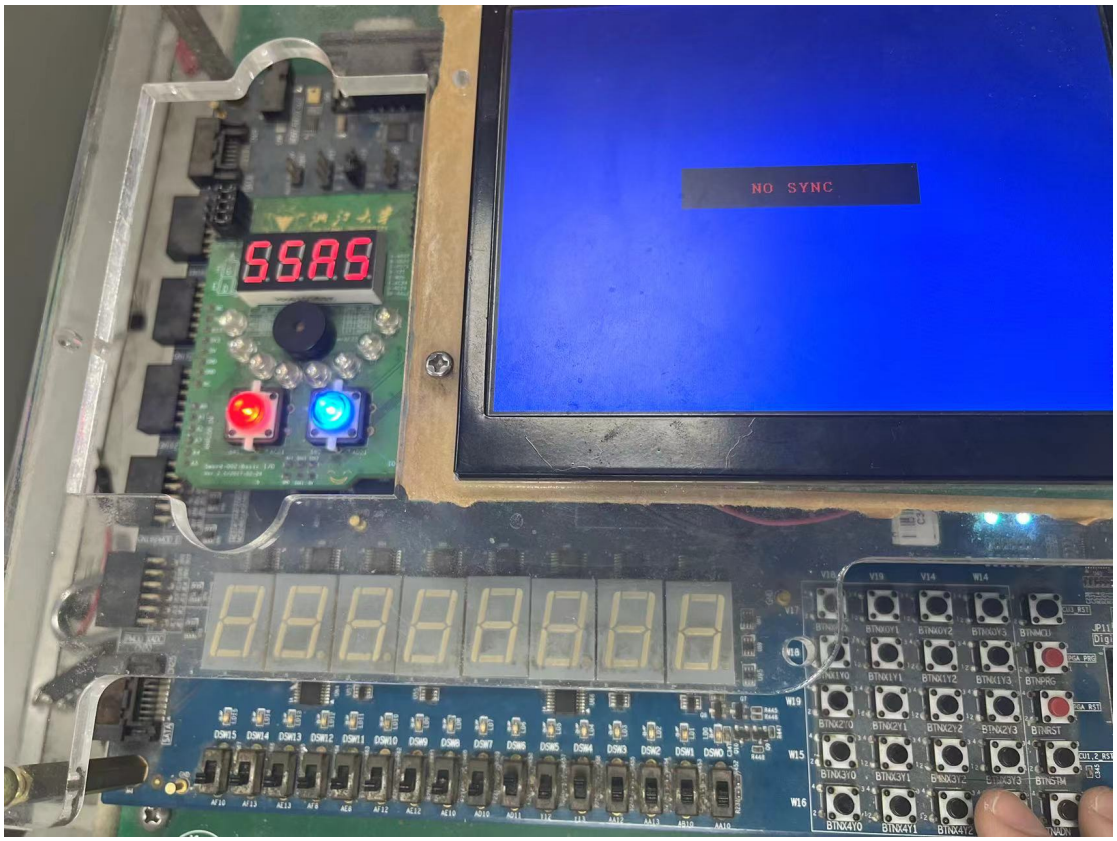


（3）切换数据传输控制模式

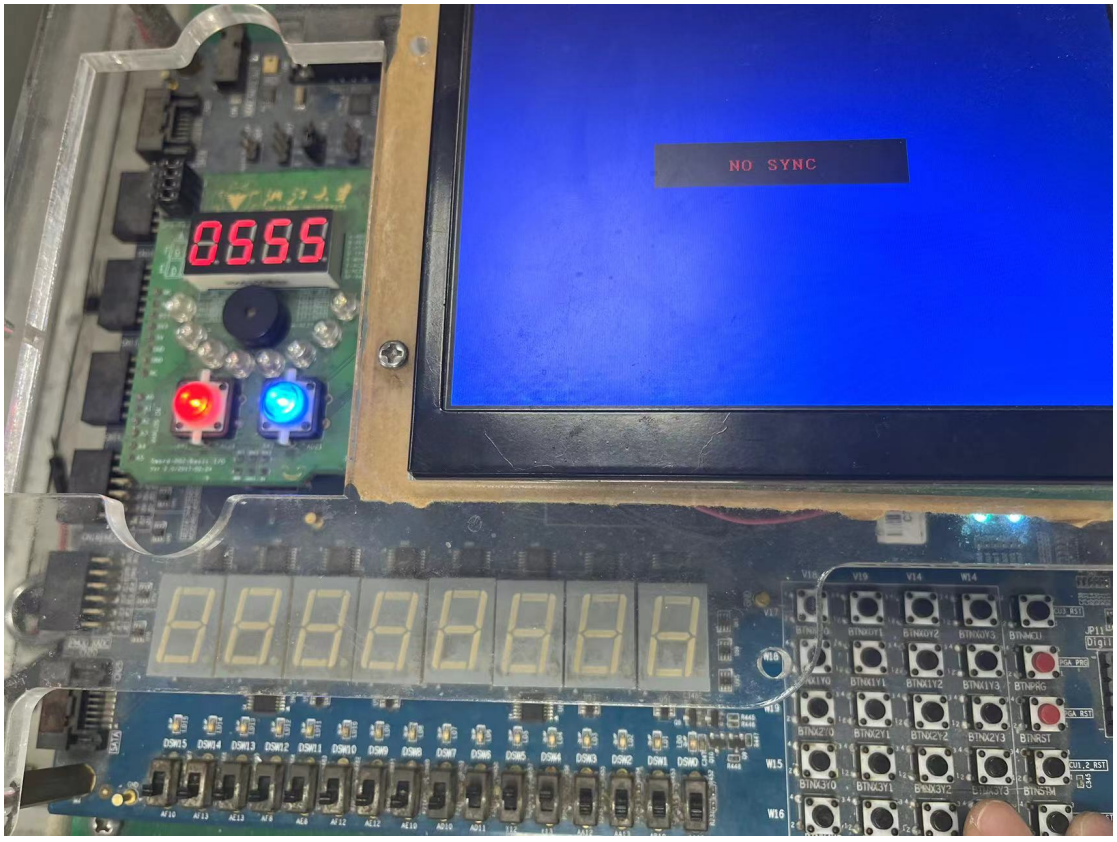
1）初始情况



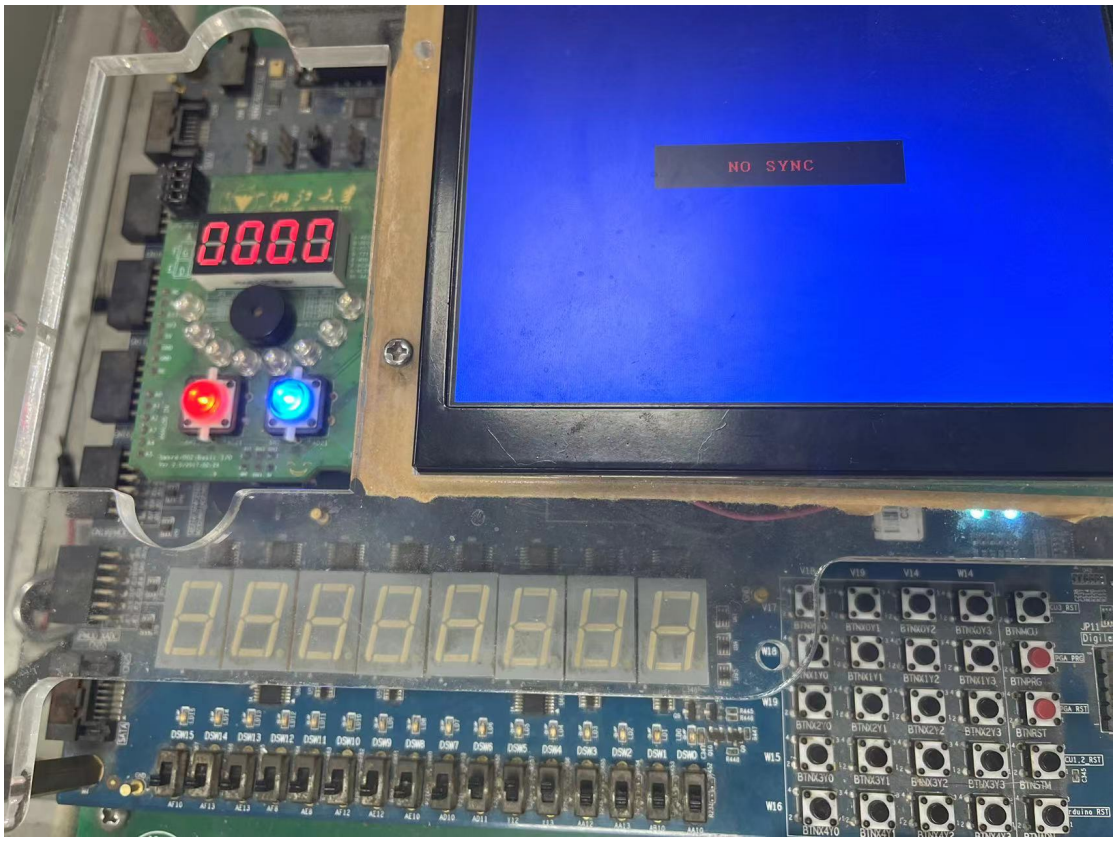
2）选择 B 代替 A



3）选择 0 代替 A



4）选择 0 代替 B



由上述功能介绍得下板结果符合预期。

三、讨论、心得

此次实验在 Top 代码部分犯了些小错误，其本质原因还是对整个代码原理理解不够透彻，应当继续提高 Verilog 阅读代码和写代码水平。

浙江大学

本科实验报告

课程名称：	数字逻辑电路设计
姓　　名：	金祺书
学　　院：	计算机科学与技术学院
专　　业：	计算机科学与技术
指导教师：	洪奇军
报告日期：	2024 年 5 月 23 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、定时器设计和应用

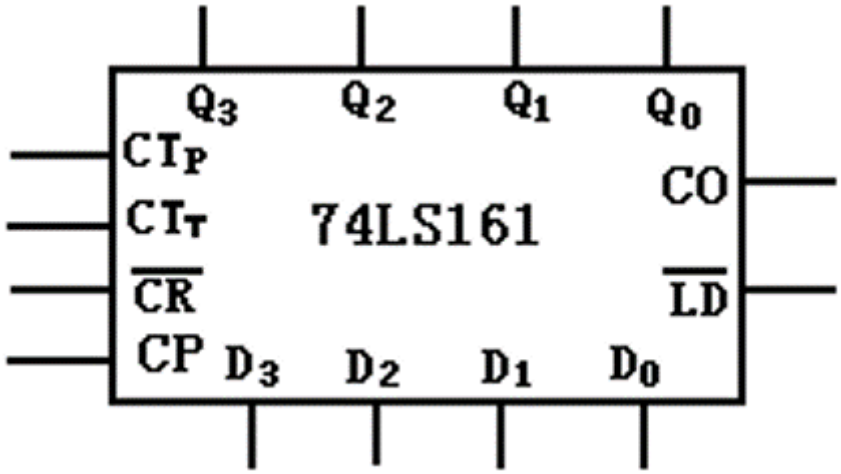
学生姓名：金祺书 学号：3230104248 同组学生姓名：蒋翼泽

实验地点：紫金港东四 509 室 实验日期：2024 年 5 月 23 日

一、操作方法与实验步骤

1、74LS161 芯片

74LS161 芯片具有同步四位二进制计数器功能，其引脚如下:

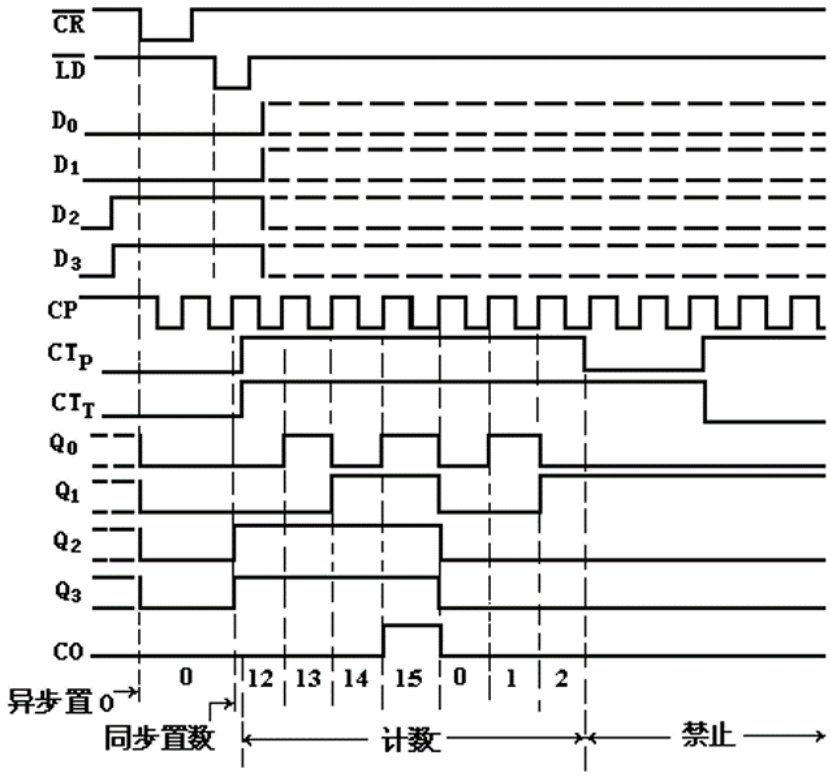


- CP:接入时钟信号，上升沿出发
- CRn:清零端，低电平有效，异步清零
- LDn:置数控制端，低电平有效
- D3~D0：置数数据端，当 LDn 有效时将数据写入
- CTT,CTP:使能端，两脚均为高电平时启用计数功能，任意一脚为低电平时计数器保持原状态
- Q3~Q0：数据输出端
- CO：进位输出端，当输出位均为 1 时置 1

其功能表如下：

输入						输出			
$\overline{CR}$	$\overline{LD}$	CT_P	CT_T	CP	$D_3\,D_2\,D_1\,D_0$	Q_3	Q_2	Q_1	Q_0
0	×	×	×	×	××××	0	0	0	0
1	0	×	×	↑	$d_3\,d_2\,d_1\,d_0$	d_3	d_2	d_1	d_0
1	1	0	1	×	××××	保	持		
1	1	×	0	×	××××	保	持		
1	1	1	1	↑	××××	计	数		

时序图：



2、实现 74LS161 功能

（1）My74LS161 模块代码如下：

```
01  `timescale 1ns / 1ps
02
03  module My74LS161(
04      input CP,
05      input CRn,
06      input LDn,
07      input [3:0] D,
08      input CTT,
09      input CTP,
10      output [3:0] Q,
11      output CO
```

```
);

    reg [3:0] Q_reg = 4'b0;

    always @(posedge CP or negedge CRn) begin
        if(!CRn) begin
            Q_reg = 4'b0000;
        end
        else begin
            if(!LDn) begin
                Q_reg = D;
            end else if (CTT & CTP) begin
                Q_reg = Q_reg + 1;
                if(Q_reg == 16)
                    Q_reg = 0;
            end
        end
    end

    assign Q = Q_reg;
    assign CO = (Q == 4'hF);

endmodule
```

（2）Vivado 新建工程，对该模块进行仿真激励，检验该模块功能是否正确，仿真激励代码如下：

```
module My74Ls161_tb();
//Inputs
    reg CP;
    reg CRn;
    reg CTP;
    reg CTT;
    reg LDn;
    reg [3:0] D;
//Outputs
    wire [3:0] Q;
    wire CO;
//Instantiate the UUT
    My74LS161 My74LS161_inst(
        .CP(CP),
        .CRn(CRn),
        .CTP(CTP),
        .CTT(CTT),
        .LDn(LDn),
```

```
19          .D(D),
20          .Q(Q),
21          .CO(CO)
22      );
23   //Initialize inputs
24      initial begin
25          CRn=1;
26          LDn=1;
27          CTP=1;
28          CTT=1;
29          D=4'b1001;#100;
30          CRn=0;#5;
31          CRn=1;#20;
32          LDn=0;#30;
33          LDn=1;#100;
34          CTP=0;#40;
35          CTT=0;#40;
36          CTT=1;#40;
37          CTP=1;#40;
38
39      end
40      always begin
41          CP = 1;#10;
42          CP = 0;#10;
43      end
```

3、74LS161 应用

实现一个格式为“小时：分钟”的时钟应用，使用 Arduino 板上的七段数码管进行输出。使用 SW[0] 选择时钟速度。Top 模块如下：

```
01  module top(
02      input clk,
03      input [1:0] SW,
04      output [3:0] AN,
05      output [7:0] SEGMENT
06  );
07
08      wire clk_10ms;
09      wire clk_100ms;
10      // clk_1s used in LabA
11      clk_10ms clk_div_10ms (.clk(clk), .clk_10ms(clk_10ms)); //
Refer to the code of clk_1s to complete these modules
12      clk_100ms clk_div_100ms (.clk(clk), .clk_100ms(clk_100ms));
13
14      wire clk_counter = (SW[0] == 1'b0) ? clk_10ms : clk_100ms; //
```

```
Connect this clk_counter to CP-port of 74LS161
15
16      wire [15:0] num;
17
18      // Your code here to get the correct HOUR and MINUTE
19      wire CO1;
20      assign RSTm0 = num[3] & num[0]; // Reset conditions for each
counter
21      assign RSTm1 = num[6] & num[4] & RSTm0;
22      assign RSTh0 = num[11] & num[8] & RSTm1;
23      assign RSTh1 = num[13] & num[9] & num[8] & RSTm0;
24
25     My74LS161    m0( .CRn(1'b1), .CTP(1'b1), .CTT(1'b1),
26                      .CP(clk_counter),
27                      .LDn(~RSTm0 ),
28                      .D(4'b0),
29                      .Q(num[3:0]) ),
30                  m1( .CRn(1'b1), .CTP(1'b1),
31                      .CTT( RSTm0 ),
32                      .CP(clk_counter),
33                      .LDn( ~RSTm1 ),
34                      .D(4'b0),
35                      .Q(num[7:4]) ),
36                  h0( .CRn(1'b1), .CTP(1'b1),
37                      .CTT( RSTm1 ),
38                      .CP(clk_counter),
39                      .LDn(~(RSTh0 | RSTh1) ),
40                      .D(4'b0),
41                      .Q(num[11:8]) ),
42                  h1( .CRn(1'b1), .CTP(1'b1),
43                      .CTT( RSTh0 ),
44                      .CP(clk_counter),
45                      .LDn(~RSTh1 ),
46                      .D(4'b0),
47                      .Q(num[15:12]) );
48      // Module written in Lab 7
49  DisplayNumber
display_inst(.clk(clk), .hexs(num), .points(4'b0100), .rst(1'b0), .LE
s(4'b0000), .AN(AN), .SEGMENT(SEGMENT));
50
51  endmodule
```

其中 clk_10ms 模块：

```
01  module clk_10ms(
02      input clk,
```

```
    output reg clk_10ms
    );
    reg [31:0] cnt;

    initial begin
        cnt = 32'b0;
    end

    wire[31:0] cnt_next;
    assign cnt_next = cnt + 1'b1;

    always @(posedge clk) begin
        if(cnt<5_000_000)begin
            cnt <= cnt_next;
        end
        else begin
            cnt <= 0;
            clk_10ms <= ~clk_10ms;
        end
    end
endmodule
```

clk_100ms 模块：

```
module clk_100ms(
    input clk,
    output reg clk_100ms
);

    reg [31:0] cnt;

    initial begin
        cnt = 32'b0;
    end

    wire[31:0] cnt_next;
    assign cnt_next = cnt + 1'b1;

    always @(posedge clk) begin
        if(cnt<50_000_000)begin
            cnt <= cnt_next;
        end
        else begin
            cnt <= 0;
            clk_100ms <= ~clk_100ms;
        end
```

```
    end

endmodule
```

通过约束文件：

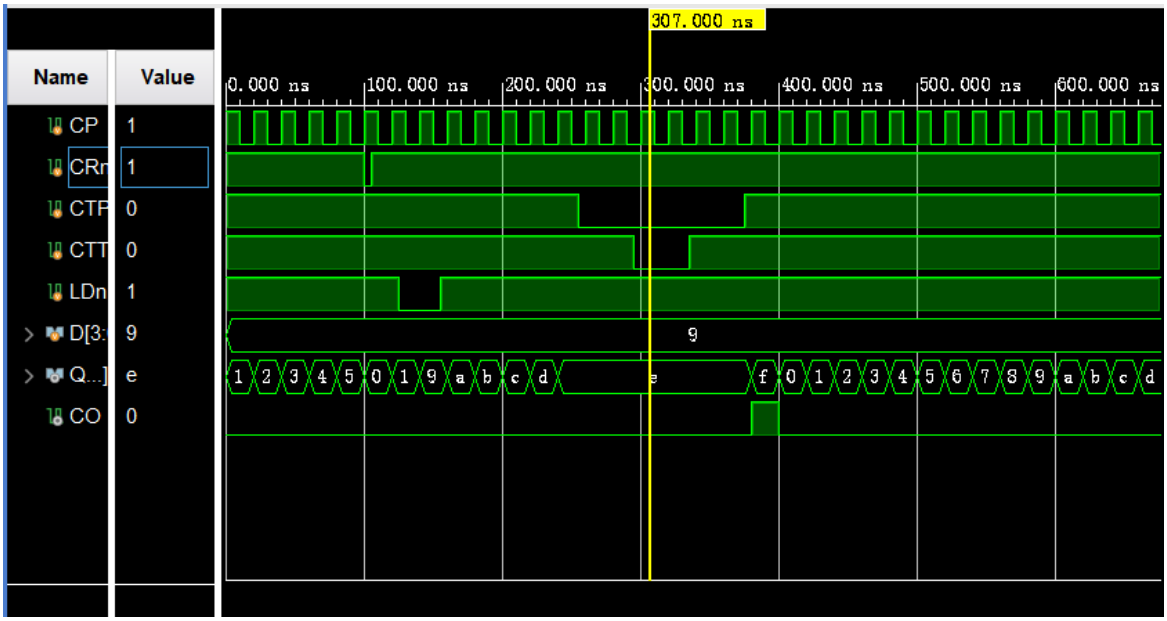
```
# Filename: constraints_labC.xdc
## Constraints file for LabC

# Main clock
set_property PACKAGE_PIN AC18 [get_ports clk]
set_property IOSTANDARD LVCMOS18 [get_ports clk]

create_clock -period 10.000 -name clk [get_ports "clk"]

# Switches as inputs
set_property PACKAGE_PIN AA10 [get_ports {SW[0]}]
set_property PACKAGE_PIN AB10 [get_ports {SW[1]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[0]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[1]}]

# Arduino-Segment & AN
set_property PACKAGE_PIN AD21 [get_ports {AN[0]}]
set_property PACKAGE_PIN AC21 [get_ports {AN[1]}]
set_property PACKAGE_PIN AB21 [get_ports {AN[2]}]
set_property PACKAGE_PIN AC22 [get_ports {AN[3]}]
set_property PACKAGE_PIN AB22 [get_ports {SEGMENT[0]}]
set_property PACKAGE_PIN AD24 [get_ports {SEGMENT[1]}]
set_property PACKAGE_PIN AD23 [get_ports {SEGMENT[2]}]
set_property PACKAGE_PIN Y21 [get_ports {SEGMENT[3]}]
set_property PACKAGE_PIN W20 [get_ports {SEGMENT[4]}]
set_property PACKAGE_PIN AC24 [get_ports {SEGMENT[5]}]
set_property PACKAGE_PIN AC23 [get_ports {SEGMENT[6]}]
set_property PACKAGE_PIN AA22 [get_ports {SEGMENT[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[7]}]
```

生成比特流下板验证。

二、实验结果与分析

1、实现 74LS161 功能

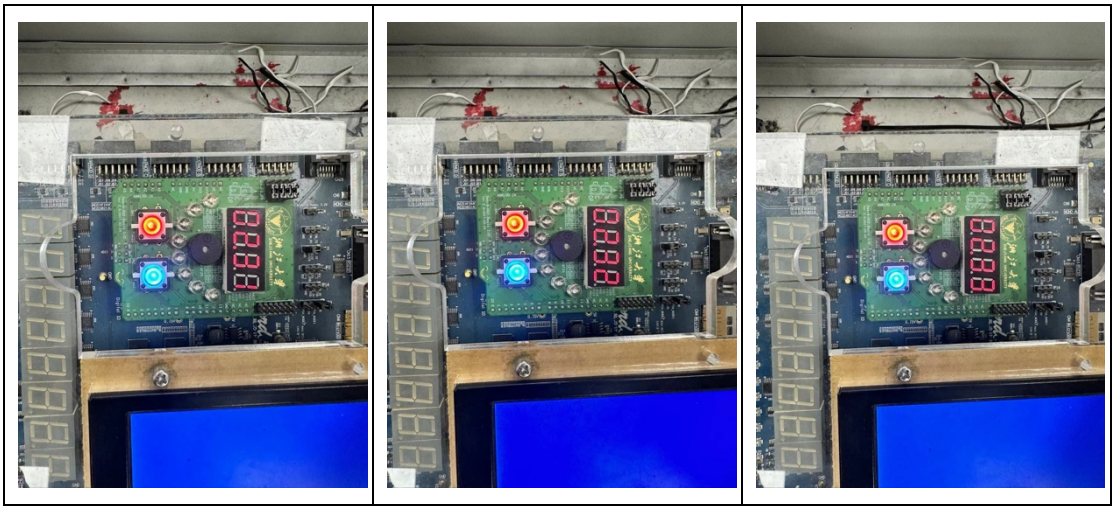
导入 Vivado 后的仿真界面截图如下：



当 CRn 置为 1 且 CTT\CTF 同时为 1 时，Q 自增，任意时刻 CRn 置为 0 后归零，LDn 置 0 后且在时钟上升沿时置数为 9，当 CTF，CTT 任意一个置为 0 时 Q 保持不变，Q 自增至 f 后恢复为 0，仿真结果符合预期。

2、74LS161 应用

下板结果如下：



当 SW[0]为 0 时以 100ms 的速度自增，当 SW[0]为 1 时以 10ms 的速度自增
第一位数字到9后进位，第二位数字到5且第一位数字到9后进位，第三位数字到9且第二位数字到5且第一位数字到9后进位，第四位数字到2且第三位数字到3且第二位数字到5且第一位数字到9时所有数字归为0，实现时钟的功能，符合预期。

三、讨论、心得

本次实验一开始较为顺利，但在 top 代码中如何实现进位卡住了较长时间，下板结果一直不如人意，说明对代码的熟悉度还是不够，应当再加强。

浙江大学

本科实验报告

课程名称：	数字逻辑电路设计
姓　　名：	金祺书
学　　院：	计算机科学与技术学院
专　　业：	计算机科学与技术
指导教师：	洪奇军
报告日期：	2024 年 5 月 30 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

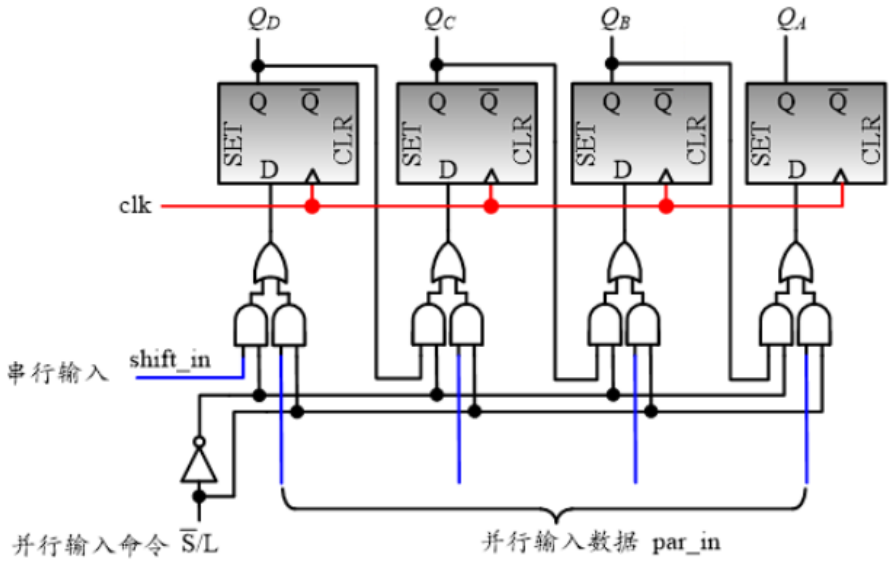
实验项目名称：移位寄存器设计与应用

学生姓名：金祺书　学号：3230104248　同组学生姓名：蒋翼泽

实验地点：紫金港东四 509 室　　实验日期：2024 年 5 月 30 日

一、操作方法与实验步骤

1、verilog 代码实现 8 位右移移位寄存器



clk：时钟信号，在时钟上升沿对存储内容进行修改
shiftn_loadp：控制信号，在低电平时进行移位操作，在高电平时进行并行数据读入
shift_in：移位时移入的数据
par_in：八位并行输入数据
Q：并行输出数据
（1）实现该模块代码如下：

```
01  `timescale 1ns / 1ps
02
03  module ShiftReg8b(
04      input       clk,
05      input       shiftn_loadp,
06      input       shift_in,
```

```
    input [7:0] par_in,
    output[7:0] Q
);
    reg [7:0] temp;
    initial temp=0;
    always @(posedge clk) begin
        if(shiftn_loadp)
            temp=par_in;
        else begin
            temp=temp>>1;
            temp[7]=shift_in;
        end
    end
    assign Q=temp;
endmodule
```

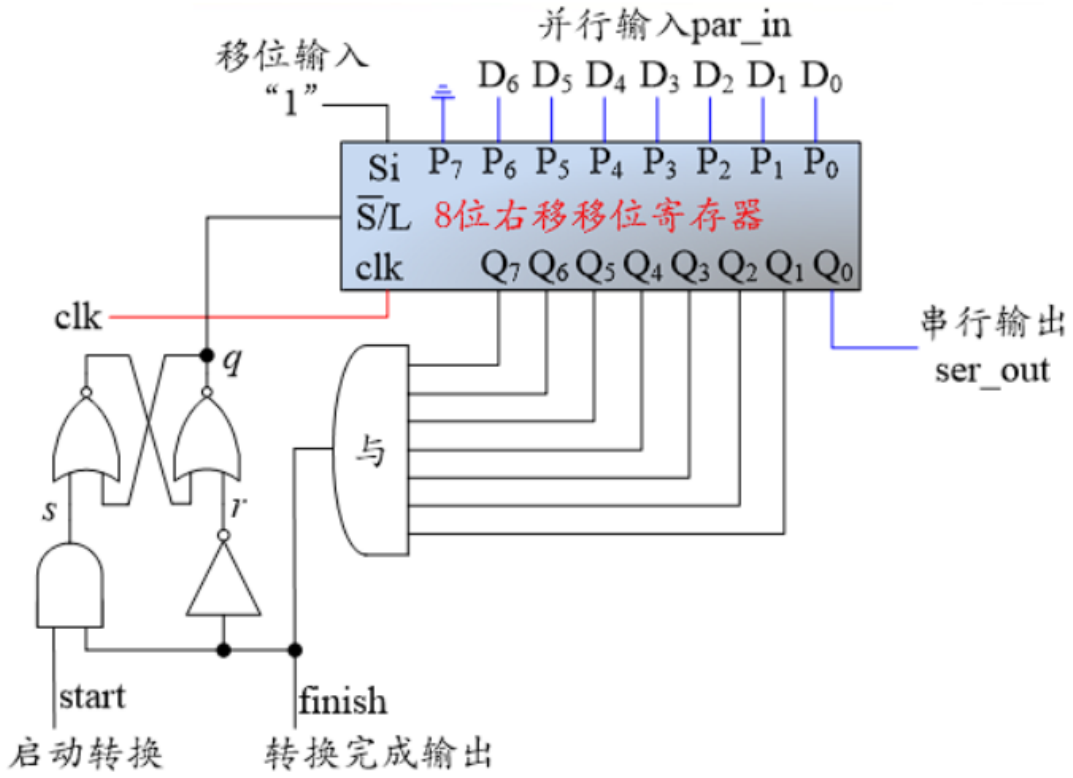
（2）Vivado新建工程，对该模块进行仿真激励，检验该模块功能是否正确，仿真激励代码如下：

```
`timescale 1ns / 1ps

module ShiftReg8b_tb();
//Inputs
    reg clk;
    reg shiftn_loadp;
    reg shift_in;
    reg [7:0] par_in;
//Outputs
    wire [7:0] Q;
//Instantiate the UUT
    ShiftReg8b ShiftReg8b_inst(
        .clk(clk),
        .shiftn_loadp(shiftn_loadp),
        .shift_in(shift_in),
        .par_in(par_in),
        .Q(Q)
    );
//Initialize Inputs
    initial begin
        clk=0;
        shiftn_loadp=0;
        shift_in=0;
        par_in=0;
        #100;

        shiftn_loadp=0;
```

```
28          shift_in=1;
29          par_in=0;
30          #200;
31
32          shiftn_loadp=1;
33          shift_in=0;
34          par_in=8'b0101_0101;
35          #500;
36      end
37      always begin
38          clk=0;#20;
39          clk=1;#20;
40      end
41  endmodule
```

2、verilog 代码实现 P2S 模块

并行数据转串行输出模块(P2S, Parallel to Serial Converter)的作用是将并行数据（比如 16 位 LED 亮灭的控制信号）转换成串行输出（同时需要管理串行通信的其他相关信号，如 sclk, sclrn）。原理图如下：



因为直接使用逻辑门可能会综合失败，这里我们模仿SR锁存器行为，代码如下：

```
01  module SR_Latch(
02      input S,
03      input R,
04      output Q,
05      output Qn
```

```
);

    reg Q_reg = 1'b0;

    always @(*) begin
        if(!S && R) Q_reg = 1'b0;
        else if(S && !R) Q_reg = 1'b1;
    end

    assign Q = Q_reg;
    assign Qn = ~Q_reg;

endmodule
```

将上述8位移位寄存器改造成任意位宽的移位寄存器，代码如下：

```
`timescale 1ns / 1ps

module ShiftReg
    #(parameter BIT_WIDTH = 8)(
    input       clk,
    input       shiftn_loadp,
    input       shift_in,
    input [BIT_WIDTH-1:0] par_in,
    output[BIT_WIDTH-1:0] Q,
    output flag
    );
    reg [BIT_WIDTH-1:0] temp;
    reg ans;
    initial temp=0;
    integer i;
    always @(posedge clk) begin
        if(shiftn_loadp)
            temp=par_in;
        else begin
            temp=temp>>1;
            temp[BIT_WIDTH-1]=shift_in;
        end
        ans=1;
        for(i=1;i<BIT_WIDTH;i=i+1) begin
            ans = ans & temp[i];
        end
    end
    assign Q=temp;
    assign flag = ans;
endmodule
```

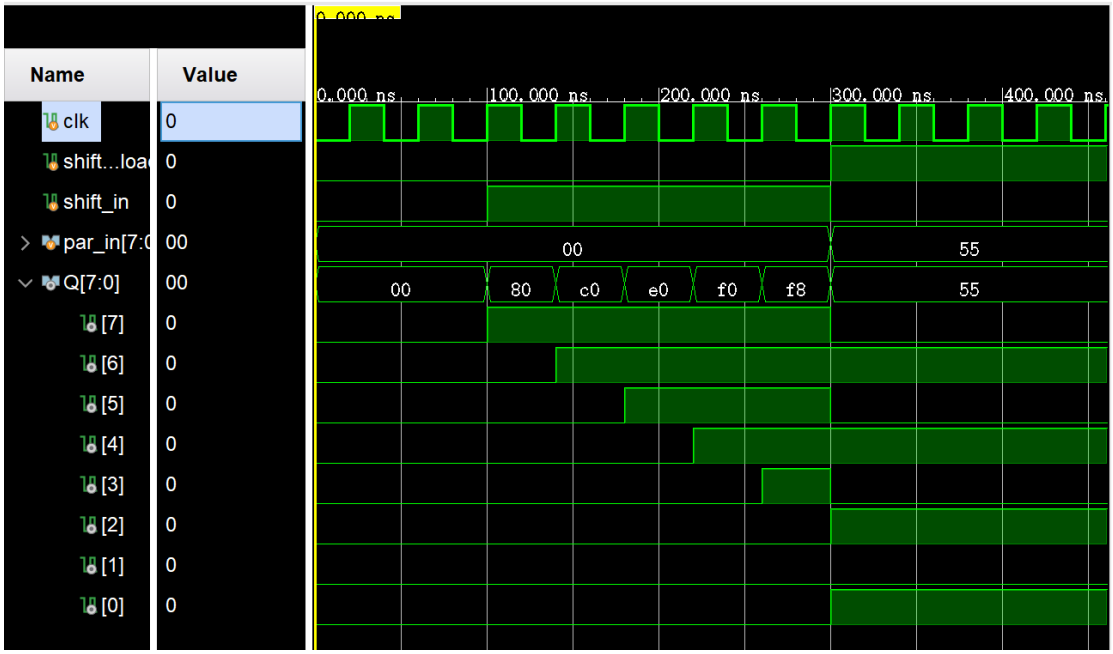
P2S 模块代码如下：

```
`timescale 1ns / 1ps

module P2S
#(parameter BIT_WIDTH = 8)(
    input clk,
    input start,
    input[BIT_WIDTH-1:0] par_in,
    output sclk,
    output sclrn,
    output sout,
    output EN,
    output q,
    output finish,
    output[BIT_WIDTH-1:0] Q,
    output flag
);

    wire qn;

    SR_Latch SR_Latch_inst(.S(start &
finish),.R(~finish),.Q(q),.Qn(qn));// Your code here

    ShiftReg
#(.BIT_WIDTH(8))ShiftReg_inst(.clk(clk),.shiftn_loadp(q),.shift_in(1'
b1),.par_in(par_in),.Q(Q),.flag(flag));// Your code here

    assign finish = flag;// Your code here

    assign EN = !start && finish;
    assign sclk = finish | ~clk;
    assign sclrn = 1'b1;
    assign sout = Q[0];// Your code here

endmodule
```

思考题：其中，`sclk = finish | ~clk;` 语句中 `~clk` 的作用是防止其与 clk 相互干扰，待移位稳定后再进行串行通信，如果与 clk 相同，会出现一边移位一边通信的情况，容易相互干扰产生问题。

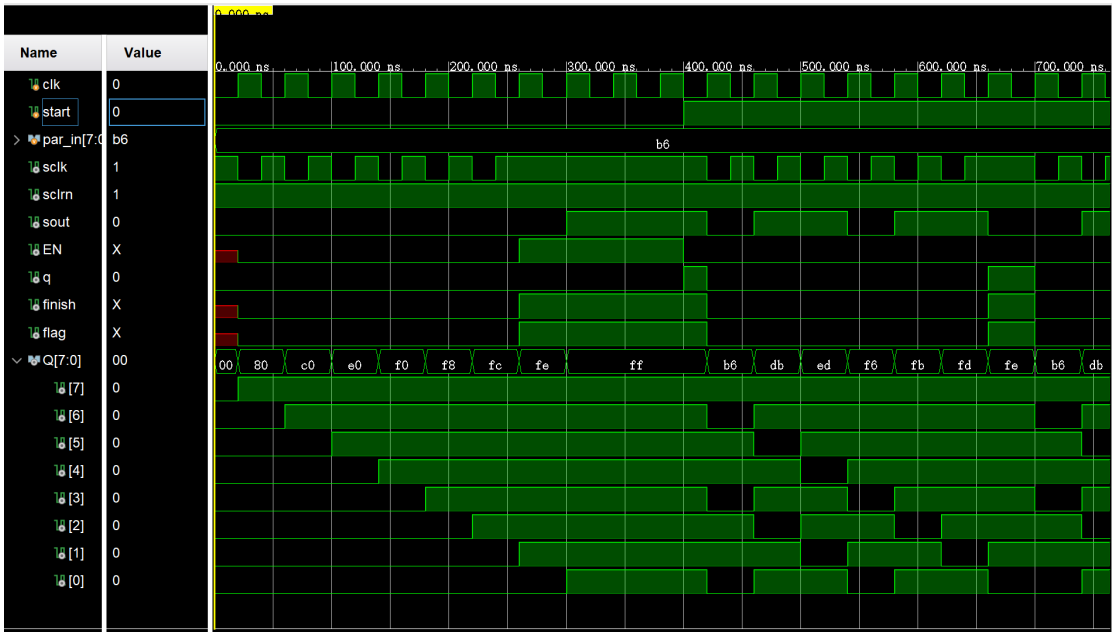
二、实验结果与分析

1、八位移位寄存器仿真结果



Shiftn_loadp 低电平时进行移位操作，移入 shift_in 的数据，高电平时进行并行数据读入，读入的数据位 par_in,所有都是在时钟上升沿时进行操作

2、P2S 模块仿真结果



（1）初始（start 置 0）：此时 S-R 锁存器的 set 信号一定为 0，根据锁存器当前存储信号 q 的值分类讨论:

1）q=0 表示进行串行输入，即每一个时钟周期移位并补 1，若干周期后 Q[7]~Q[1] 均为 1，此时 finish 信号置 1，reset 信号置 0，锁存器状态保持为 0

2）q=1 表示进行并行输入，此时并行输入 7 位脏值，但由于最高位接地一定为 0，finish 信号一定为 0，reset 信号置 1，锁存器状态改变 q=0，后经过一段时间后锁存器状态为 0，finish 为 1

3）初始状态开始一段时间以后，finish 信号一定为 1，表示并未进行串行输出，此时模块状态稳定，等待 start 信号

（2）开始传输（外界准备好并行输入的数据后，start 置 1）：在并行输入的 7 位数据准备好后，start 信号进行一次脉冲（0-1-0），（因为初始状态下 finish 置 1）在 start 置 1 时，S-R 锁存器进行一次 set，q=1，移位寄存器进行了并行输入 Q[7:0] = {1'b0, D[6:0]}
1）最高位存在一个 0，因此一定有 finish=0
a)sclk = finish | ~clk，此时 sclk 值和 clk 相反
b)ser_out 每一个时钟周期输出最低位，右移一位，高位补 1
（3）传输结束：传输过程中，高位始终补 1，当并行输入的 7 位全部输出后，当前的 Q 值为 8'b1111_1110
1）finish 置 1，表示串行输出结束
2）sclk 置 1，不再存在“上升沿”
3）q=0，且 set 和 reset 信号均为 0，保持
4）等待下一个 start 信号，重新传输
结合图可得仿真结果符合预期。

三、讨论、心得

本次实验在实现 P2S 模块的过程中还算顺利，顺利得出了正确的仿真结果，但是在后续实现 bonus 模块的过程中由于对整体模块的不熟悉导致代码难以撰写，上板操作难以实现，对代码的撰写能力还有待提升。