浙江大学

本科实验报告

课程名称：	数字逻辑电路设计
姓　　名：	蔡佳伟
学　　院：	计算机科学与技术学院
专　　业：	软件工程
邮　　箱：	3220104519@zju.edu.cn
QQ　　号：	3348536459
电　　话：	19550230334
指导教师：	洪奇军
报告日期：	2023 年 12 月 5 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 11：寄存器与寄存器传输设计

学生姓名：蔡佳伟 学号：3220104519 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

（一）设计 Top.v 和撰写约束文件

1、Top.v 文件依照给出的部分进行修改和增加，

Verilog 代码如下：

```
module Top(
    input clk,
    input [3:0] BTN_Y,
    input [15:0] SW,
    output BTN_X,
    output[3:0]AN,
    output[7:0] SEGMENT
);

    wire [31:0] my_clkdiv;
    wire [2:0] btn_out;
    reg  [11:0] num;
    wire [3:0] A1, A2, B1, B2, C1, C2;
    wire [3:0] mux_out;
    wire Co;
    wire [3:0] ALU_res;

    /* SW[1:0] to control if the counter for A or B is reversal */
    wire A_Ctrl = SW[0];
    wire B_Ctrl = SW[1];
    /* SW[3:2] to choose the mode of the ALU */
    wire [1:0] ALU_Ctrl = SW[3:2];
    /* SW[5:4] to choose from A B C and 0 */
    /* 00 for A; 01 for B; 10 for C; 11 for 0 */
    wire [1:0] Trans_select = SW[5:4];

    wire [3:0] reg_A_val = num[ 3: 0];
    wire [3:0] reg_B_val = num[ 7: 4];
    wire [3:0] reg_C_val = num[11: 8];

    assign BTN_X = 1'b0;
```

```
    clkdiv m0(.clk(clk), .rst(1'b0), .div_res(my_clkdiv));

    pbdebounce
m1(.clk(my_clkdiv[17]), .button(BTN_Y[0]), .pbreg(btn_out[0]));
    pbdebounce
m2(.clk(my_clkdiv[17]), .button(BTN_Y[1]), .pbreg(btn_out[1]));
    pbdebounce
m3(.clk(my_clkdiv[17]), .button(BTN_Y[2]), .pbreg(btn_out[2]));

    AddSub4b m4(.A(reg_A_val), .B(4'b0001), .Ctrl(A_Ctrl), .S(A1));
    AddSub4b m5(.A(reg_B_val), .B(4'b0001), .Ctrl(B_Ctrl), .S(B1));

    Mux4to1b4
m6(.D0(reg_A_val), .D1(reg_B_val), .D2(reg_C_val), .D3(4'b0000),
                               .S(Trans_select), .Y(mux_out));

    /* ALU module implemented in Lab8 */
    /* A/B    : operands */
    /* S       : select the operation on ALU  */
    /* C        : result of ALU */
    /* Co       : Carry bit */
    ALU
m7(.A(reg_A_val), .B(reg_B_val), .res(ALU_res), .Cout(Co), .op(ALU_Ct
rl)); // (Co) may be useless

    DisplayNumber m8(.clk(clk), .hexs({reg_A_val, reg_B_val, ALU_res,
reg_C_val}),
                         .LEs(4'b0000),  .points(4'b0000),  .rst(1'b0),
.AN(AN),
                         .SEGMENT(SEGMENT));

    /* Your code here */
    // SW[15]: 0 for ALU mode, 1 for Trans mode.
    assign A2 = (1'b0 == SW[15]) ? A1 : mux_out;
    assign B2 = (1'b0 == SW[15]) ? B1 : mux_out;
    assign C2 = (1'b0 == SW[15]) ? ALU_res : mux_out;

    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;
    /******************/

endmodule
```

2、撰写的约束文件 Verilog 代码如下：

```
# Filename: constraints_labB.xdc
## Constraints file for LabB

# 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 PACKAGE_PIN AA13 [get_ports {SW[2]}]
set_property PACKAGE_PIN AA12 [get_ports {SW[3]}]
```

```
set_property PACKAGE_PIN Y13 [get_ports {SW[4]}]
set_property PACKAGE_PIN Y12 [get_ports {SW[5]}]
set_property PACKAGE_PIN AD11 [get_ports {SW[6]}]
set_property PACKAGE_PIN AD10 [get_ports {SW[7]}]
set_property PACKAGE_PIN AE10 [get_ports {SW[8]}]
set_property PACKAGE_PIN AE12 [get_ports {SW[9]}]
set_property PACKAGE_PIN AF12 [get_ports {SW[10]}]
set_property PACKAGE_PIN AE8 [get_ports {SW[11]}]
set_property PACKAGE_PIN AF8 [get_ports {SW[12]}]
set_property PACKAGE_PIN AE13 [get_ports {SW[13]}]
set_property PACKAGE_PIN AF13 [get_ports {SW[14]}]
set_property PACKAGE_PIN AF10 [get_ports {SW[15]}]

set_property IOSTANDARD LVCMOS15 [get_ports {SW[0]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[1]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[2]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[3]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[4]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[5]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[6]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[7]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[8]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[9]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[10]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[11]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[12]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[13]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[14]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[15]}]

# Key as inputs
set_property PACKAGE_PIN V18 [get_ports {BTN_Y[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports {BTN_Y[0]}]
set_property PACKAGE_PIN V19 [get_ports {BTN_Y[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {BTN_Y[1]}]
set_property PACKAGE_PIN V14 [get_ports {BTN_Y[2]}]
set_property IOSTANDARD LVCMOS18 [get_ports {BTN_Y[2]}]


# 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]}]

# set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets BTN_Y[0]]
# set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets BTN_Y[1]]
# set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets BTN_Y[2]]
# set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets btn[3]]
set_property PACKAGE_PIN W16 [get_ports {BTN_X}]
set_property IOSTANDARD LVCMOS18 [get_ports {BTN_X}]

# # Main clock
# set_property PACKAGE_PIN AC18 [get_ports clk_p]
# set_property PACKAGE_PIN AD18 [get_ports clk_n]
# set_property IOSTANDARD LVCMOS18 [get_ports clk_p]
# set_property IOSTANDARD LVCMOS18 [get_ports clk_n]

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

# # FPGA RST
# set_property PACKAGE_PIN W13 [get_ports RSTN]
# set_property IOSTANDARD LVCMOS18 [get_ports RSTN]

# # 7SEG
# set_property PACKAGE_PIN M24 [get_ports seg_clk]
# set_property PACKAGE_PIN L24 [get_ports set_sout]
# set_property PACKAGE_PIN R18 [get_ports seg_pen]
# set_property PACKAGE_PIN M20 [get_ports seg_clrn]
# set_property IOSTANDARD LVCMOS33 [get_ports seg_clk]
# set_property IOSTANDARD LVCMOS33 [get_ports set_sout]
# set_property IOSTANDARD LVCMOS33 [get_ports seg_pen]
# set_property IOSTANDARD LVCMOS33 [get_ports seg_clrn]

# # Audio out
# set_property PACKAGE_PIN P26 [get_ports AUD_PWM]
# set_property PACKAGE_PIN M25 [get_ports AUD_SD]
# set_property IOSTANDARD LVCMOS33 [get_ports AUD_PWM]
# set_property IOSTANDARD LVCMOS33 [get_ports AUD_SD]

# # Key Array
# set_property PACKAGE_PIN V17 [get_ports BTN_X0]
# set_property IOSTANDARD LVCMOS18 [get_ports BTN_X0]
# set_property PACKAGE_PIN W18 [get_ports BTN_X1]
# set_property IOSTANDARD LVCMOS18 [get_ports BTN_X1]
# set_property PACKAGE_PIN W19 [get_ports BTN_X2]
# set_property IOSTANDARD LVCMOS18 [get_ports BTN_X2]
# set_property PACKAGE_PIN W15 [get_ports BTN_X3]
# set_property IOSTANDARD LVCMOS18 [get_ports BTN_X3]
# set_property PACKAGE_PIN W16 [get_ports BTN_X4]
# set_property IOSTANDARD LVCMOS18 [get_ports BTN_X4]
# set_property PACKAGE_PIN V18 [get_ports BTN_Y0]
# set_property IOSTANDARD LVCMOS18 [get_ports BTN_Y0]
# set_property PACKAGE_PIN V19 [get_ports BTN_Y1]
# set_property IOSTANDARD LVCMOS18 [get_ports BTN_Y1]
# set_property PACKAGE_PIN V14 [get_ports BTN_Y2]
# set_property IOSTANDARD LVCMOS18 [get_ports BTN_Y2]
# set_property PACKAGE_PIN W14 [get_ports BTN_Y3]
# set_property IOSTANDARD LVCMOS18 [get_ports BTN_Y3]

# # Arduino
# set_property PACKAGE_PIN AF25 [get_ports ard_rst]
```

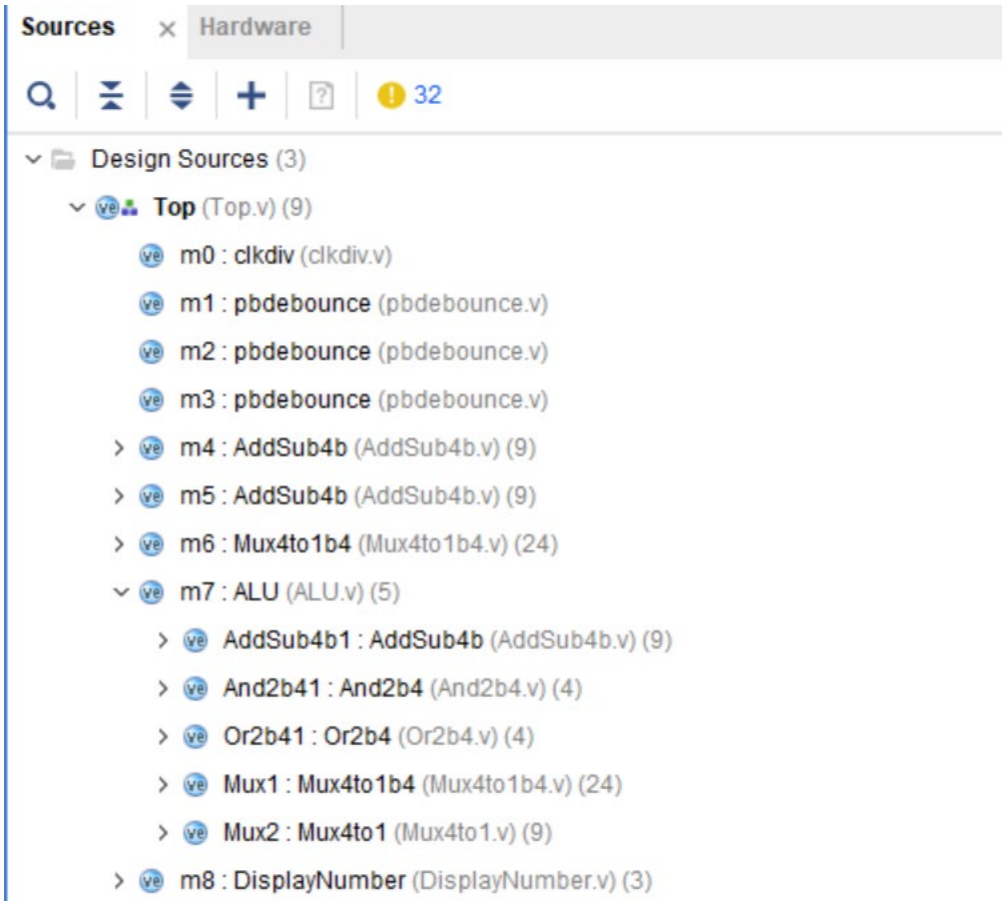
```
# set_property IOSTANDARD LVCMOS33 [get_ports ard_rst]
# set_property PACKAGE_PIN AF24 [get_ports {ard_led[0]}]
# set_property PACKAGE_PIN AF21 [get_ports {ard_led[1]}]
# set_property PACKAGE_PIN Y22 [get_ports {ard_led[2]}]
# set_property PACKAGE_PIN Y23 [get_ports {ard_led[3]}]
# set_property PACKAGE_PIN AA23 [get_ports {ard_led[4]}]
# set_property PACKAGE_PIN Y25 [get_ports {ard_led[5]}]
# set_property PACKAGE_PIN AB26 [get_ports {ard_led[6]}]
# set_property PACKAGE_PIN W23 [get_ports {ard_led[7]}]
# set_property IOSTANDARD LVCMOS33 [get_ports {ard_led[0]}]
# set_property IOSTANDARD LVCMOS33 [get_ports {ard_led[1]}]
# set_property IOSTANDARD LVCMOS33 [get_ports {ard_led[2]}]
# set property IOSTANDARD LVCMOS33 [get ports {ard led[3]}]
# set_property IOSTANDARD LVCMOS33 [get_ports {ard_led[4]}]
# set_property IOSTANDARD LVCMOS33 [get_ports {ard_led[5]}]
# set_property IOSTANDARD LVCMOS33 [get_ports {ard_led[6]}]
# set_property IOSTANDARD LVCMOS33 [get_ports {ard_led[7]}]
# set_property PACKAGE_PIN AD21 [get_ports {ard_an[0]}]
# set_property PACKAGE_PIN AC21 [get_ports {ard_an[1]}]
# set_property PACKAGE_PIN AB21 [get_ports {ard_an[2]}]
# set_property PACKAGE_PIN AC22 [get_ports {ard_an[3]}]
# set_property PACKAGE_PIN AB22 [get_ports {ard_seg[0]}]
# set_property PACKAGE_PIN AD24 [get_ports {ard_seg[1]}]
# set_property PACKAGE_PIN AD23 [get_ports {ard_seg[2]}]
# set_property PACKAGE_PIN Y21 [get_ports {ard_seg[3]}]
# set_property PACKAGE_PIN W20 [get_ports {ard_seg[4]}]
# set_property PACKAGE_PIN AC24 [get_ports {ard_seg[5]}]
# set_property PACKAGE_PIN AC23 [get_ports {ard_seg[6]}]
# set_property PACKAGE_PIN AA22 [get_ports {ard_seg[7]}]
# # set_property IOSTANDARD LVCMOS33 [get_ports {ard_dio[13]}]
# # set_property IOSTANDARD LVCMOS33 [get_ports {ard_dio[12]}]
# set_property IOSTANDARD LVCMOS33 [get_ports {ard_an[0]}]
# set_property IOSTANDARD LVCMOS33 [get_ports {ard_an[1]}]
# set_property IOSTANDARD LVCMOS33 [get_ports {ard_an[2]}]
# set_property IOSTANDARD LVCMOS33 [get_ports {ard_an[3]}]
# set_property IOSTANDARD LVCMOS33 [get_ports {ard_seg[0]}]
# set_property IOSTANDARD LVCMOS33 [get_ports {ard_seg[1]}]
# set_property IOSTANDARD LVCMOS33 [get_ports {ard_seg[2]}]
# set_property IOSTANDARD LVCMOS33 [get_ports {ard_seg[3]}]
# set_property IOSTANDARD LVCMOS33 [get_ports {ard_seg[4]}]
# set_property IOSTANDARD LVCMOS33 [get_ports {ard_seg[5]}]
# set_property IOSTANDARD LVCMOS33 [get_ports {ard_seg[6]}]
# set_property IOSTANDARD LVCMOS33 [get_ports {ard_seg[7]}]

# #16leds
# set_property PACKAGE_PIN N26 [get_ports LEDCLK]
# set_property PACKAGE_PIN N24 [get_ports LEDCLR]
# set_property PACKAGE_PIN M26 [get_ports LEDDT]
# set_property IOSTANDARD LVCMOS33 [get_ports LEDCLK]
# set_property IOSTANDARD LVCMOS33 [get_ports LEDCLR]
# set_property IOSTANDARD LVCMOS33 [get_ports LEDDT]

# #16dips
# set_property PACKAGE_PIN AA10 [get_ports {switch[0]}]
# set_property PACKAGE_PIN AB10 [get_ports {switch[1]}]
# set_property PACKAGE_PIN AA13 [get_ports {switch[2]}]
# set_property PACKAGE_PIN AA12 [get_ports {switch[3]}]
# set_property PACKAGE_PIN Y13 [get_ports {switch[4]}]
# set_property PACKAGE_PIN Y12 [get_ports {switch[5]}]
# set_property PACKAGE_PIN AD11 [get_ports {switch[6]}]
# set_property PACKAGE_PIN AD10 [get_ports {switch[7]}]
```

```
# set_property PACKAGE_PIN AE10 [get_ports {switch[8]}]
# set_property PACKAGE_PIN AE12 [get_ports {switch[9]}]
# set_property PACKAGE_PIN AF12 [get_ports {switch[10]}]
# set_property PACKAGE_PIN AE8 [get_ports {switch[11]}]
# set_property PACKAGE_PIN AF8 [get_ports {switch[12]}]
# set_property PACKAGE_PIN AE13 [get_ports {switch[13]}]
# set_property PACKAGE_PIN AF13 [get_ports {switch[14]}]
# set_property PACKAGE_PIN AF10 [get_ports {switch[15]}]
# set_property IOSTANDARD LVCMOS15 [get_ports {switch[0]}]
# set_property IOSTANDARD LVCMOS15 [get_ports {switch[1]}]
# set_property IOSTANDARD LVCMOS15 [get_ports {switch[2]}]
# set_property IOSTANDARD LVCMOS15 [get_ports {switch[3]}]
# set property IOSTANDARD LVCMOS15 [get ports {switch[4]}]
# set_property IOSTANDARD LVCMOS15 [get_ports {switch[5]}]
# set_property IOSTANDARD LVCMOS15 [get_ports {switch[6]}]
# set_property IOSTANDARD LVCMOS15 [get_ports {switch[7]}]
# set_property IOSTANDARD LVCMOS15 [get_ports {switch[8]}]
# set_property IOSTANDARD LVCMOS15 [get_ports {switch[9]}]
# set_property IOSTANDARD LVCMOS15 [get_ports {switch[10]}]
# set_property IOSTANDARD LVCMOS15 [get_ports {switch[11]}]
# set_property IOSTANDARD LVCMOS15 [get_ports {switch[12]}]
# set_property IOSTANDARD LVCMOS15 [get_ports {switch[13]}]
# set_property IOSTANDARD LVCMOS15 [get_ports {switch[14]}]
# set_property IOSTANDARD LVCMOS15 [get_ports {switch[15]}]

# # VGA
# set_property PACKAGE_PIN N21 [get_ports {vga_red[0]}]
# set_property PACKAGE_PIN N22 [get_ports {vga_red[1]}]
# set_property PACKAGE_PIN R21 [get_ports {vga_red[2]}]
# set_property PACKAGE_PIN P21 [get_ports {vga_red[3]}]
# set_property IOSTANDARD LVCMOS33 [get_ports {vga_red[0]}]
# set_property IOSTANDARD LVCMOS33 [get_ports {vga_red[1]}]
# set_property IOSTANDARD LVCMOS33 [get_ports {vga_red[2]}]
# set_property IOSTANDARD LVCMOS33 [get_ports {vga_red[3]}]
# set_property PACKAGE_PIN R22 [get_ports {vga_green[0]}]
# set_property PACKAGE_PIN R23 [get_ports {vga_green[1]}]
# set_property PACKAGE_PIN T24 [get_ports {vga_green[2]}]
# set_property PACKAGE_PIN T25 [get_ports {vga_green[3]}]
# set_property IOSTANDARD LVCMOS33 [get_ports {vga_green[0]}]
# set_property IOSTANDARD LVCMOS33 [get_ports {vga_green[1]}]
# set_property IOSTANDARD LVCMOS33 [get_ports {vga_green[2]}]
# set_property IOSTANDARD LVCMOS33 [get_ports {vga_green[3]}]
# set_property PACKAGE_PIN T20 [get_ports {vga_blue[0]}]
# set_property PACKAGE_PIN R20 [get_ports {vga_blue[1]}]
# set_property PACKAGE_PIN T22 [get_ports {vga_blue[2]}]
# set_property PACKAGE_PIN T23 [get_ports {vga_blue[3]}]
# set_property IOSTANDARD LVCMOS33 [get_ports {vga_blue[0]}]
# set_property IOSTANDARD LVCMOS33 [get_ports {vga_blue[1]}]
# set_property IOSTANDARD LVCMOS33 [get_ports {vga_blue[2]}]
# set_property IOSTANDARD LVCMOS33 [get_ports {vga_blue[3]}]
# set_property PACKAGE_PIN M22 [get_ports vga_hs]
# set_property PACKAGE_PIN [get_ports vga_vs]
# set_property IOSTANDARD LVCMOS33 [get_ports vga_hs]
# set_property IOSTANDARD LVCMOS33 [get_ports vga_vs]
```

（二）把文件导入 Vivado 进行验证

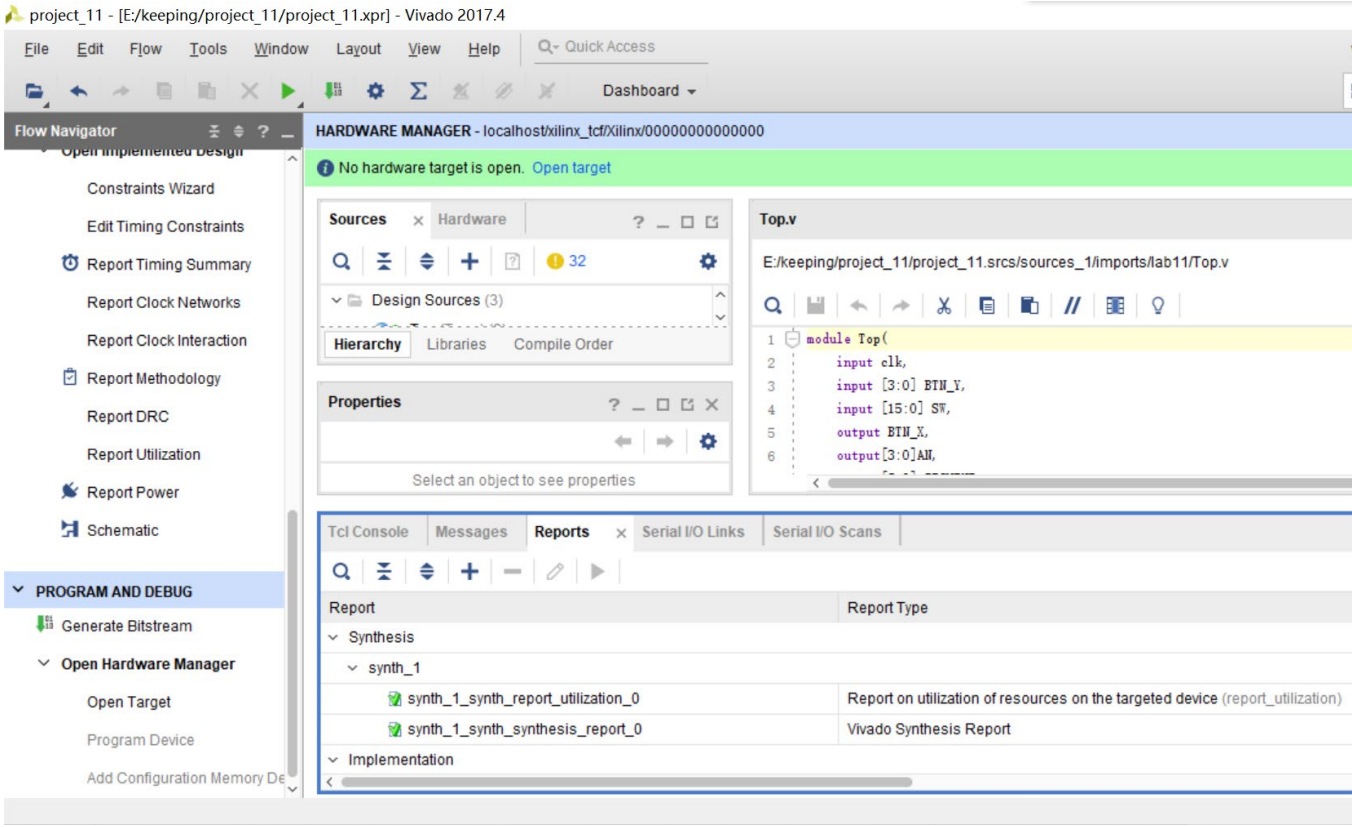
把 clkdiv.v, AddSub4b.v, Mux4to1b4, pbdebounce.v, DisplayNumber.v 和之前写这些 Verilog 代码用原理图实现的 Logisim 文件夹中的 circuits 和 gates 均导入 Vivado。

导入后文件属性如下：



生成 bitstream 并烧录

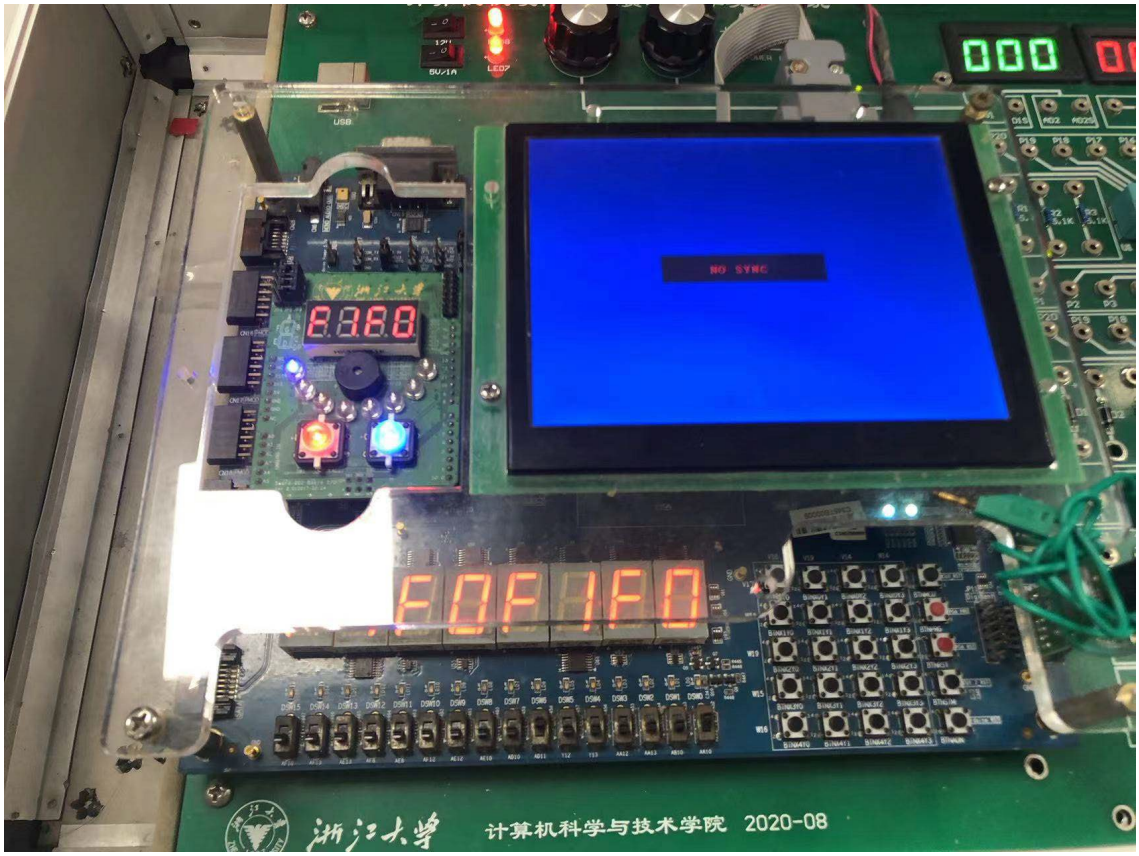
配置完成后，得到 bitstream。



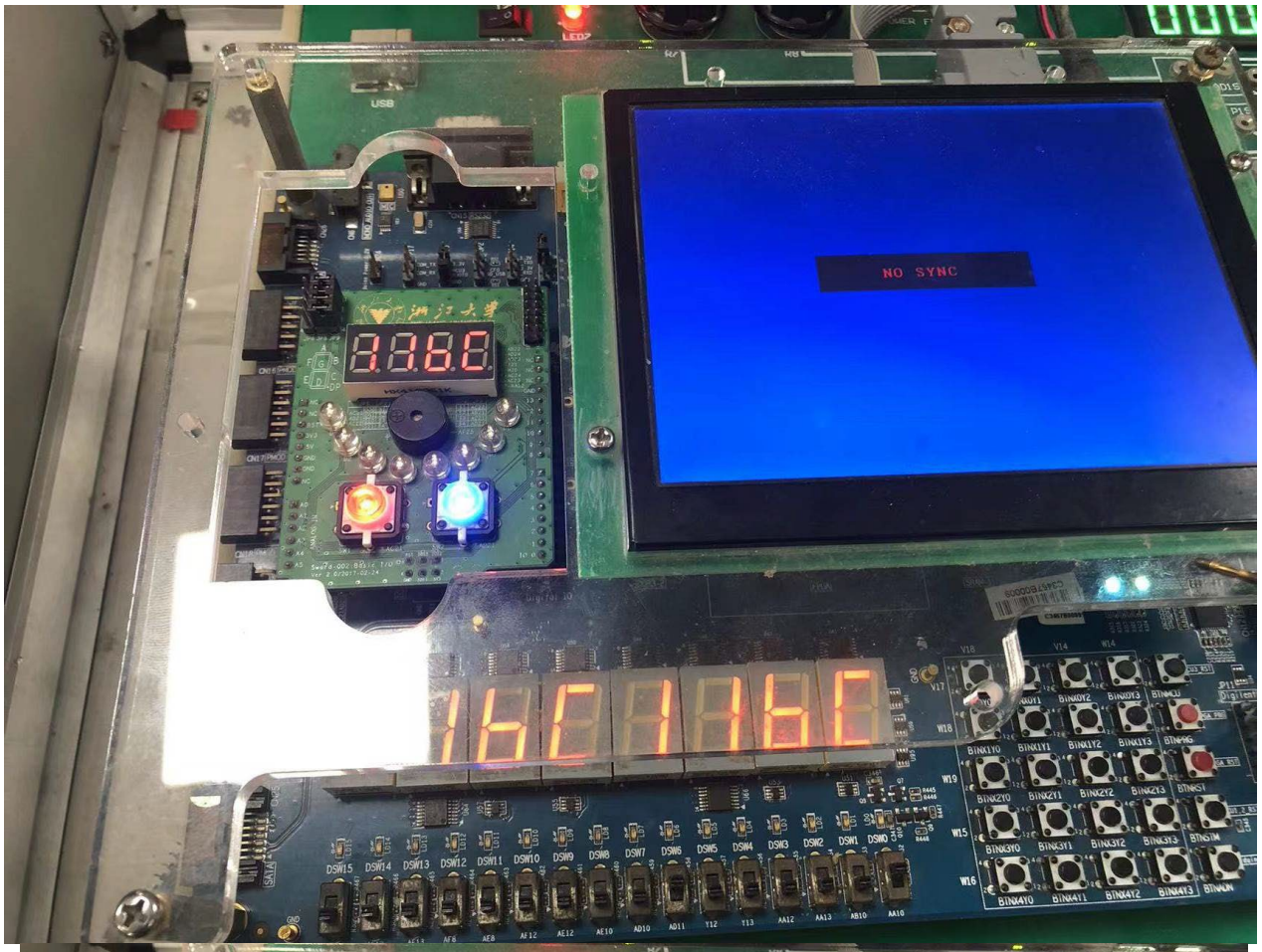
之后将下载器连接到电脑上。点击 PROGRAM AND DEBUG > Open Hardware Manager > Open Target > Auto Connect 进行识别和连接，成功连接后，点击 Program Device 选择 xc7k160t 设备，在下载程序界面选择我们刚刚生成的比特流文件，将其下载到板上。之后在板上实现相关操作。

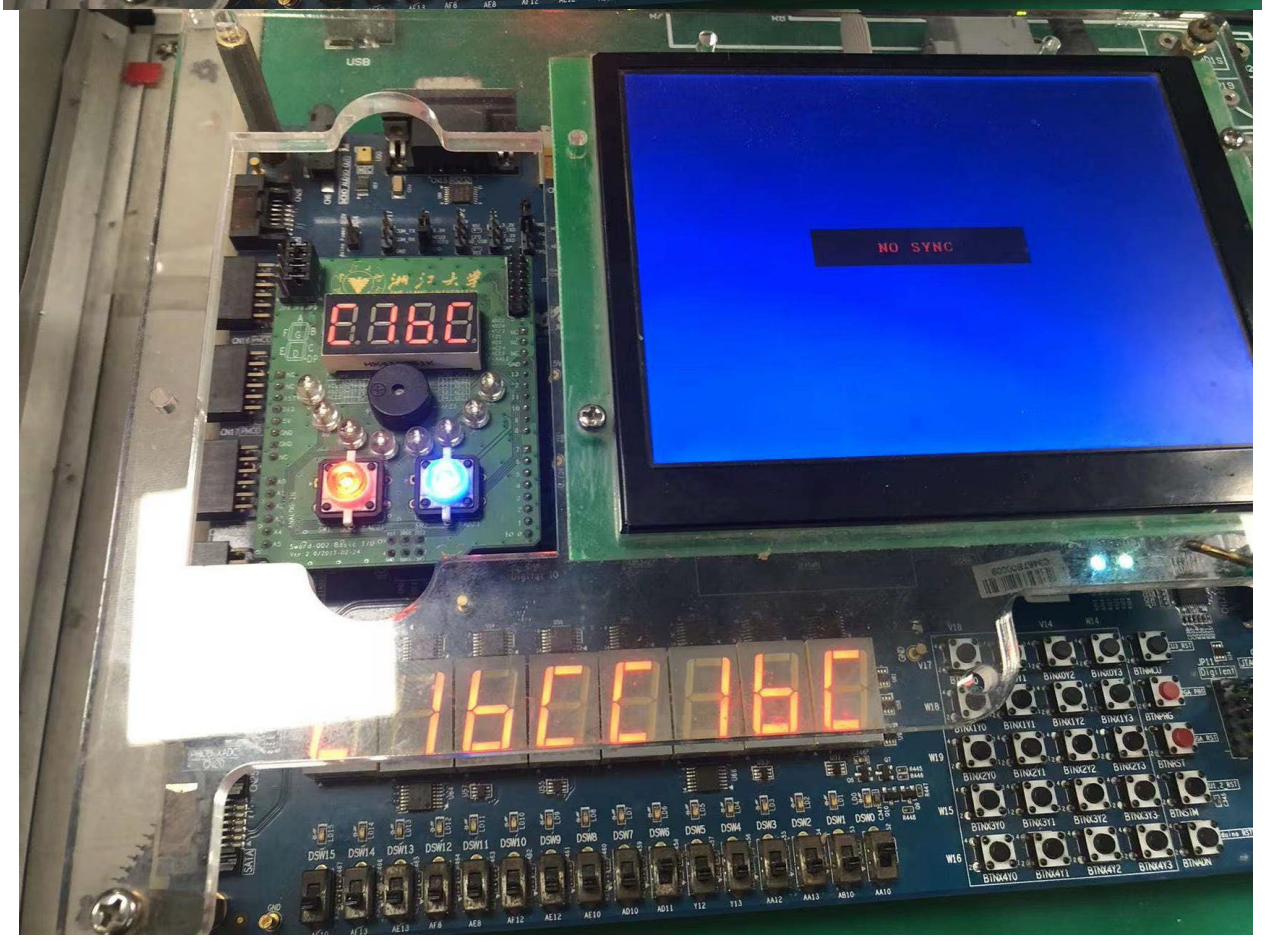
二、实验结果与分析

寄存器与寄存器传输上板实验结果



如图所示，第二位加第三位的值等于第四位，同时进位灯点亮，说明成功实现加法功能。


NO SYNC


NO SYNC

如图所示，按下 btn[2]对应的按钮，用加法结果的值更新了寄存器的值，说明成功实现该功能。

三、讨论、心得

这次实验依然是一个综合性很强的实验。在实验中，我进行了寄存器的设计，更深地理解了触发器的结果和寄存器的工作原理，也再次运用了加法器和 DisplayNumber 等常见的.v 文件来解决问题。总体来看，我在实验中遇到了很多问题，但都在求助同学后得到了解答。

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：	蔡佳伟
学　　院：	计算机科学与技术学院
专　　业：	软件工程
邮　　箱：	3220104519@zju.edu.cn
QQ　　号：	3348536459
电　　话：	19550230334
指导教师：	洪奇军
报告日期：	2023 年 12 月 12 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 12：计数器、定时器设计与应用

学生姓名：蔡佳伟 学号：3220104519 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、 实验目的

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、 实验内容和原理

2.1 实验内容

基于 ALU 的数据传输应用设计

2.2 实验原理

1、寄存器

寄存器是一组二进制的存储单元，它是由具有存储功能的触发器组合起来构成的。一个寄存器可以用于存储一列二进制值并且保持多个时钟周期，通常用于进行简单数据存储、移动和处理等操作。但是寄存器一个时钟周期只能执行一种操作，所以我们需要添加一个信号来控制寄存器执行的是加载（load）操作还是保存（store）操作。

对于 load 的控制信号，其中一种方法将 load 信号与 clock 信号一起经过一个或门组成一个控制信号，但是由于反相器存在着传输延迟，所以可能会出现时钟偏斜（clock skew）的问题，可能会影响到电路的时序。

所以更加可靠的一种方法是采用控制反馈的方法设计控制信号，这样能够保证时钟的连

续性，不会影响到电路的时序。

2、寄存器传输

寄存器传输主要设计到寄存器之间数据的传输和处理，寄存器传输涉及到的三个基本单元是寄存器组、操作和操作控制。基本操作有加载、计数、移位、加法、按位操作等。

基于寄存器传输可以设计计数器，使用寄存器传输输出当前寄存器内容，通过自增器后传输回寄存器的数据输入，然后通过加载操作进行加载，实现寄存器内值的自增。由于需要清零的操作，所以在进入数据输入之前还需要经过一个 2 选 1 多路复用器。

3、基于总线的多路复用输出

为了设计寄存器与寄存器之间的传输，我们可以使用多路选择器连接上数据总线，先将寄存器的内容传输到数据总线上，再传输到另一个寄存器中，实现寄存器之间的传输。这样可以做到节省硬件的开销，但是由于数据总线中一次只能存储一个数据，所以并不能实现多个寄存器之间的并行传输，只能够一个一个进行传输。

4、寄存器传输应用设计

在有了基于多路选择器总线的寄存器传输模块之后，我们就可以进行寄存器传输的应用设计。一个简单的应用就是在中间加上 ALU 模块，通过寄存器 A 和 B 经过 ALU 获得 C 的值，并且实现寄存器之间的数据传输。

三、 操作方法和实验步骤

一、实现 74LS161 芯片功能

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，上升沿触发

CRn：清零端，低电平有效，且为异步清零

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

1、 新建 project，工程名命名为 lab12

2、 导入 Verilog 代码 My74LS161.v 文件如下

```
module My74LS161(
    input CP,
    input CRn,
    input LDn,
    input [3:0] D,
    input CTT,
    input CTP,
    output [3:0] Q,
    output CO
```

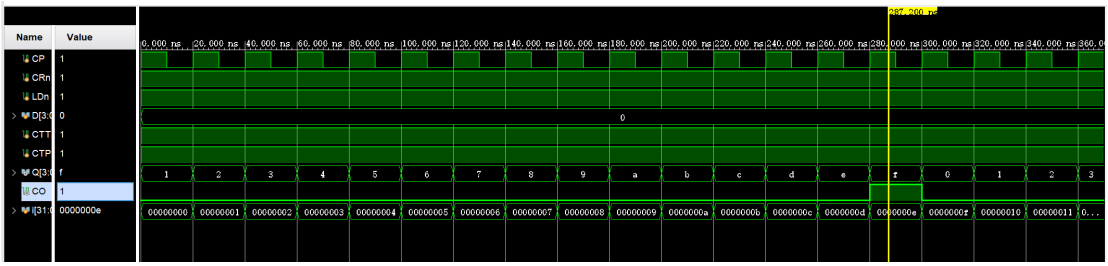
```
);

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

    always @(posedge CP or negedge CRn) begin
        if(!CRn) begin // reset
            Q_reg <= 4'b0000;
        end else if (!LDn) begin
            Q_reg <= D;
        end else if(CTT & CTP & LDn) begin //
                Q_reg <= Q_reg + 4'b0001;
        end
    end

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

endmodule
```

3、My74LS161_tb.v 文件如下：

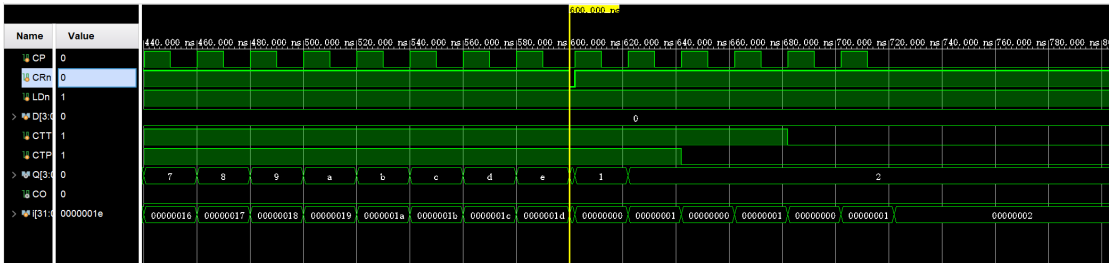
```
`timescale 1ns / 1ps

module My74LS161_tb();

// Inputs
reg CP;
reg CRn;
reg LDn;
reg [3:0] D;
reg CTT;
reg CTP;

// Output
wire [3:0] Q;
wire CO;

// Instantiate the UUT
My74LS161 My74LS161_inst (
.CP(CP),
.CRn(CRn),
.LDn(LDn),
.D(D),
.CTT(CTT),
.CTP(CTP),
```

```
.Q(Q),
.CO(CO)
);

integer i;
initial begin
    assign LDn = 1;
    assign D = 4'b0000;
    CRn = 1;
    for(i = 0; i < 30; i = i + 1)begin
        CTT = 1'b1;
        CTP = 1'b1;
        CP = 1; #10;
        CP = 0; #10;
    end
    CRn = 0;#2;
    CRn = 1;
    for(i = 0; i < 2; i = i + 1)begin
        CTT = 1'b1;
        CTP = 1'b1;
        CP = 1; #10;
        CP = 0; #10;
    end
    for(i = 0; i < 2; i = i + 1)begin
        CTT = 1'b1;
        CTP = 1'b0;
        CP = 1; #10;
        CP = 0; #10;
    end
    for(i = 0; i < 2; i = i + 1)begin
        CTT = 1'b0;
        CTP = 1'b0;
        CP = 1; #10;
        CP = 0; #10;
    end
end
endmodule
```

仿真结果：



在光标处实现进位功能，达到预期效果。



在光标处实现异步清零功能，达到预期效果。

二、74LS161 芯片应用

1、新建 project，工程名命名为 lab12_2

2、导入 top.v 文件、clk_1s.v 文件、clk_100ms.v 文件以及 My74LS161.v 文件。

top.v 文件如下

```
module top(
    input clk,
    input [1:0] SW,
    output [3:0] AN,
    output [7:0] SEGMENT
);

    wire clk_1s;
    wire clk_100ms;
    // Used in LabA
    clk_1s clk_div_1s (.clk(clk), .clk_1s(clk_1s));
    clk_100ms clk_div_100ms (.clk(clk), .clk_100ms(clk_100ms)); // Refer
to the code of clk_1s to complete this module

    wire clk_counter = (SW[0] == 1'b0) ? clk_100ms : clk_1s; // Connect
this clk_counter to CP-port of 74LS161

    wire [3:0] hour_tens;
    wire [3:0] hour_ones;
    wire [3:0] min_tens;
    wire [3:0] min_ones;

    wire [3:0] ht,ho,mt,mo;
```

```
    My74LS161 m1
(.CP(clk_counter),.CRn(1'b1),.LDn(~(min_ones[0]&min_ones[3])),.D(4'b0
),.CTT(1'b1),.CTP(1'b1),.Q(min_ones));
    My74LS161 m2
(.CP(clk_counter),.CRn(1'b1),.LDn(~(min_tens[2]&min_tens[0]&min_ones[
0]&min_ones[3])),.D(4'b0),.CTT(min_ones[0]&min_ones[3]),.CTP(1'b1),.Q
(min_tens));
    My74LS161 m3
(.CP(clk_counter),.CRn(1'b1),.LDn(~(hour_ones[0]&hour_ones[3])),.D(4'
b0),.CTT(min_tens[2]&min_tens[0]&min_ones[0]&min_ones[3]),.CTP(1'b1),
.Q(hour_ones));
    My74LS161 m4
(.CP(clk_counter),.CRn(1'b1),.LDn(~(hour_ones[0]&hour_ones[1]&hour_te
ns[1]&min_tens[2]&min_tens[0]&min_ones[0]&min_ones[3])),.D(4'b0),.CTT
(hour_ones[0]&hour_ones[3]),.CTP(1'b1),.Q(hour_tens));

    assign mo = (1'b1 == SW[1]) ? 4'b0 : min_ones;
    assign mt = (1'b1 == SW[1]) ? 4'b0 : min_tens;
    assign ho = (1'b1 == SW[1]) ? 4'b0011 : hour_ones;
    assign ht = (1'b1 == SW[1]) ? 4'b0010 : hour_tens;

    // Module written in Lab 7
    DisplayNumber display_inst(.clk(clk), .hexs({ht, ho, mt,
mo}), .points(4'b0100), .rst(1'b0), .LEs(4'b0000), .AN(AN), .SEGMENT(
SEGMENT));

Endmodule
```

clk_1s.v 文件如下：

```
`timescale 1ns / 1ps

module clk_1s(
    input clk,
    output reg clk_1s
);

    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_1s <= ~clk_1s;
        end
    end

endmodule
```

clk_100ms.v 文件如下

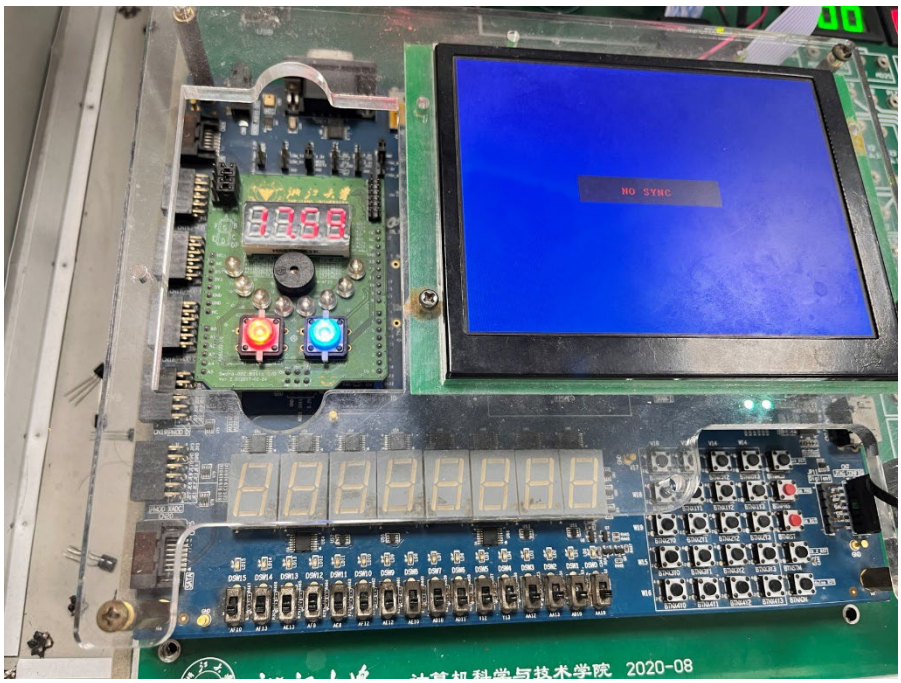
```
`timescale 1ns / 1ps

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<5_000_000)begin
            cnt <= cnt_next;
        end
        else begin
            cnt <= 0;
            clk_100ms <= ~clk_100ms;
        end
    end

endmodule
```

5、引脚约束代码 constraints_labB.xdc 如下：

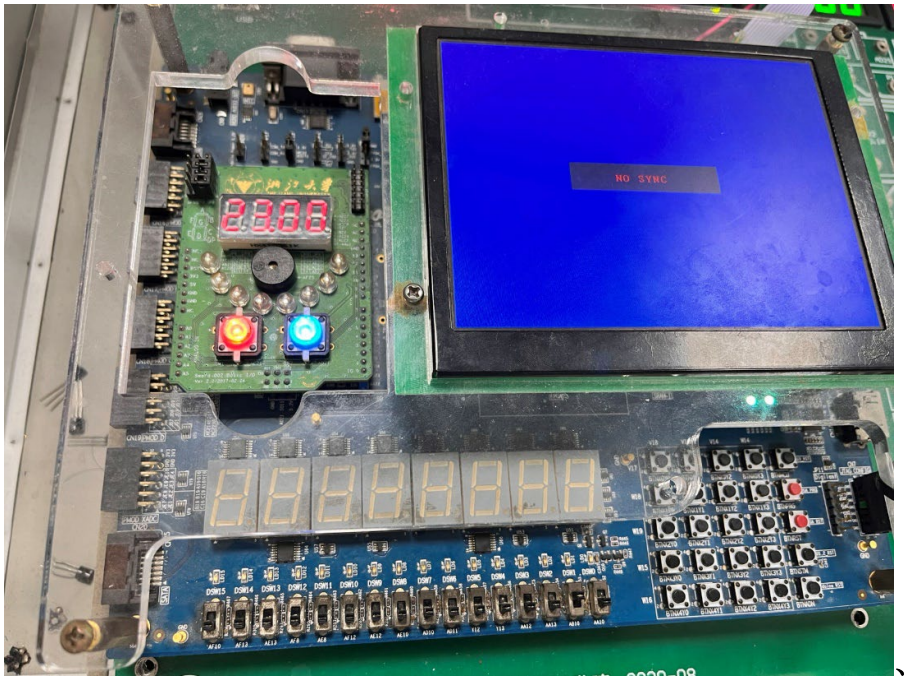
```
# Filename: constraints_labB.xdc
```

```
## Constraints file for LabB

# 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]}]
```

当 SW[0]，即最右边开关置为 1 时，时间流速为 100ms，时钟的示数迅速变化。



当 SW[0]，即最右边开关置为 1 时，时间流速为 100ms，时钟的示数变化较缓慢。

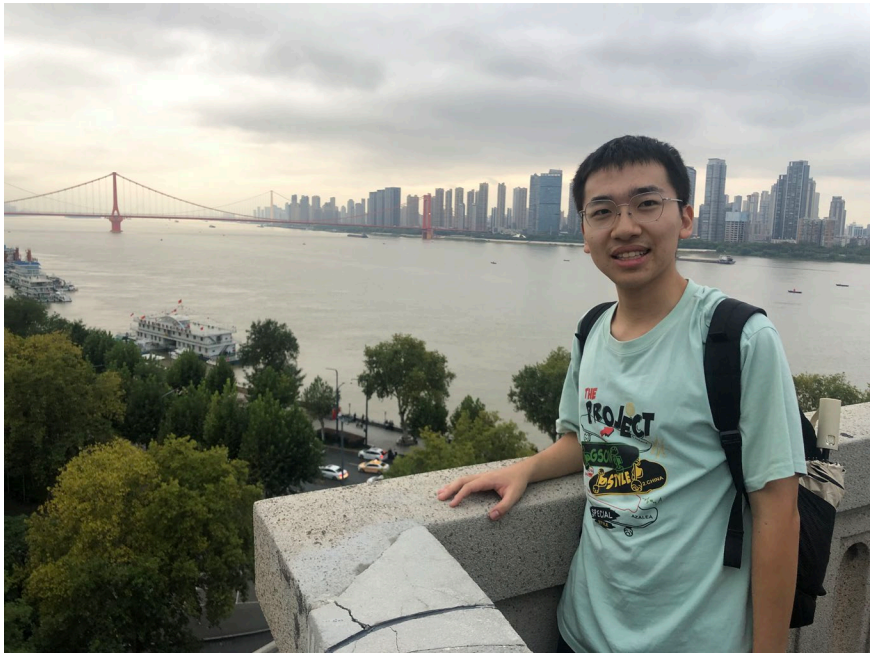


当 SW[1]置为 1 时，实现异步重置，时钟停 23:00，满足预期效果。

四、讨论、心得

在本次实验中，我逐渐感受到了实验的难度，也同时感受到实验的用处非常大和实验本身比较有趣。在我们小组开始制作大作业的时候，发现很多模块是可以直接运用的，之前也有很多设计的经验和积累的语法都对大作业有很大的帮助。

五、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：	蔡佳伟
学　　院：	计算机科学与技术学院
专　　业：	软件工程
邮　　箱：	3220104519@zju.edu.cn
QQ　　号：	3348536459
电　　话：	19550230334
指导教师：	洪奇军
报告日期：	2023 年 12 月 19 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 13：移位寄存器设计与应用

学生姓名：蔡佳伟 学号：3220104519 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

（一）移位寄存器的设计

1、移位寄存器(shift register)是一种在若干相同时钟脉冲下工作的以触发器级联为基础的器件，每个触发器的输出接在触发器链的下一级触发器的“数据”输入端，使得电路在每个时钟脉冲内依次向左或向右移动一个比特，并在输出端进行输出。

2、移位寄存器的实现

ShiftReg8b.v Verilog 代码如下：

```
module ShiftReg8b(
    input      clk,
    input      shiftn_loadp,
    input      shift_in,
    input [7:0] par_in,
    output[7:0] Q
);
    wire cons;
    wire [7:0] Qn;
    wire [7:0] D;

    genvar k; // Used in generate block
    generate
        for(k = 7; k < 8; k = k + 1) begin
            Inputgate
Inputgate_else(.Dat(shift_in), .par_in(par_in[k]), .SL(shiftn_loadp),
 .D(D[k]));
        end
    endgenerate
    genvar i; // Used in generate block
    generate
        for(i = 0; i < 7; i = i + 1) begin
            Inputgate             Inputgate_else(.Dat(Q[i             +
1]), .par_in(par_in[i]), .SL(shiftn_loadp), .D(D[i]));
        end
    endgenerate
    genvar j;
```

```
    generate
           for(j = 0; j < 8; j = j + 1) begin
           FD FD1(.clk(clk), .D(D[j]), .Q(Q[j]), .Qn(Qn[j]));
       end
     endgenerate
    endmodule;
```

本次还要用到 Lab10 中运用过的 FD.v Verilog 代码如下：

```
module FD(
    input clk,
    input D,
    output Q,
    output Qn
);

    reg Q_reg = 1'b0;
    always @(posedge clk) begin
       Q_reg <= D;
    end

    assign Q = Q_reg;
    assign Qn = ~Q_reg;

endmodule
```

其中用到 InputGate.v 代码如下：

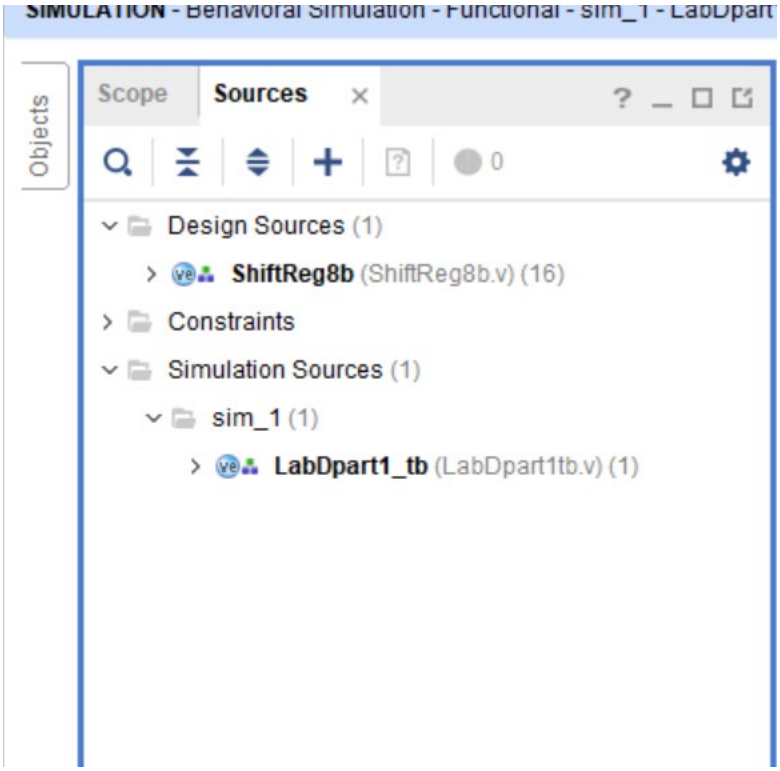
```
module Inputgate(
    input SL,
    input Dat,
    input par_in,
    output D
);

assign D = (~SL & Dat) | (SL & par_in);
endmodule
```

3、仿真文件代码

```
`timescale 1ns / 1ps

module LabDpart1_tb();

// Inputs
reg      clk;
reg      shiftn_loadp;
reg      shift_in;
reg [7:0] par_in;
// Output
wire [7:0] Q;

// Instantiate the UUT
ShiftReg8b ShiftReg8b_tb_inst(
    .clk(clk),
    .shiftn_loadp(shiftn_loadp),
    .shift_in(shift_in),
    .par_in(par_in),
    .Q(Q)
);

initial begin
    clk = 0;
    assign shift_in = 1;
```

```
    par_in = 8'b10101011;
    assign shift_in = 1;
    shiftn_loadp = 1;
    #20;
    shiftn_loadp = 0;
    #20;
    par_in = 8'b00000010;
    shiftn_loadp = 1;
    shift_in = 0;
    #20;
    shiftn_loadp = 0;
    #90;
end
always begin
    clk <= ~clk;
    #10;
end
endmodule
```

4、文件组织结构



如图所示，仿真结果在后文体现

（二）移位寄存器的应用

1、本次我完成的是 P2S 模块。以 7 位 P2S 模块为例，其工作过程简述如下：

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

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

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

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

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]}

最高位存在一个 0，因此一定有 finish=0

sclk = finish | clk，此时 sclk 值和 clk 完全相同，即输出的时间点和移位寄存器进行一次右移输出的时间点相同

ser_out 每一个时钟周期输出最低位，右移一位，高位补 1

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

finish 置 1，表示串行输出结束

sclk 置 1，不再存在“上升沿”

q=0，且 set 和 reset 信号均为 0，保持

等待下一个 start 信号，重新传输

2、P2S 模块的实现

P2S.v Verilog 代码如下：

```
module P2S
#(parameter BIT_WIDTH = 7)(
    input clk,
    input start,
    input[BIT_WIDTH-1:0] par_in,
    output sclk,
    output sclrn,
    output sout,
    output EN
);
    wire[BIT_WIDTH:0] Q;

    wire Qn;
    wire S;
    wire q;
    wire Qn;
    wire consone;
    wire finish;

    assign S = start & finish;
    assign R = ~finish;
```

```
    assign consone = 1'b1;

    SR_Latch SR_Latch_1(.S(S), .R(R), .Q(q), .Qn(Qn));

    ShiftReg8b
ShiftReg_1(.clk(clk), .shiftn_loadp(q), .shift_in(consone), .par_in({
1'b0, par_in[6:0]}), .Q(Q[7:0]));

    assign finish = Q[7] & Q[1] & Q[2] & Q[3] & Q[4] & Q[5] & Q[6];

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

endmodule
```

本次还要用到 SR_latch.v Verilog 代码如下：

```
module SR_Latch(
    input S,
    input R,
    output Q,
    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
```

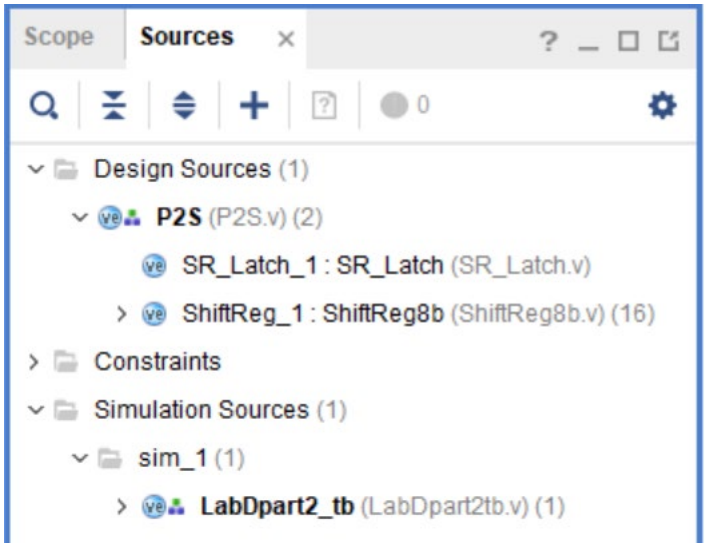
此外，还运用了上一部分的 FD.v InputGate.v 和 ShiftReg8b.v,代码不再赘述，

3、仿真文件代码

```
`timescale 1ns / 1ps

module LabDpart2_tb();

// Inputs
reg      clk;
reg      start;
reg [6:0] par_in;

// Output
wire sclk;
wire sclrn;
wire sout;
wire EN;

// Instantiate the UUT
P2S P2S_inst(
    .clk(clk),
    .start(start),
    .par_in(par_in),
    .sclk(sclk),
    .sclrn(sclrn),
```

```
    .sout(sout),
    .EN(EN)
);

initial begin
    start = 0;
    clk = 0;
    par_in = 7'b0101011;
    #250
    start = 1;
    #1;
    start = 0;
    #1;

    #125;
    #125;
    par_in = 7'b1010101;
    #250;
    start = 1;
    #1;
    start = 0;
    #1;
end
always begin
    clk <= ~clk;
    #5;
end
endmodule
```

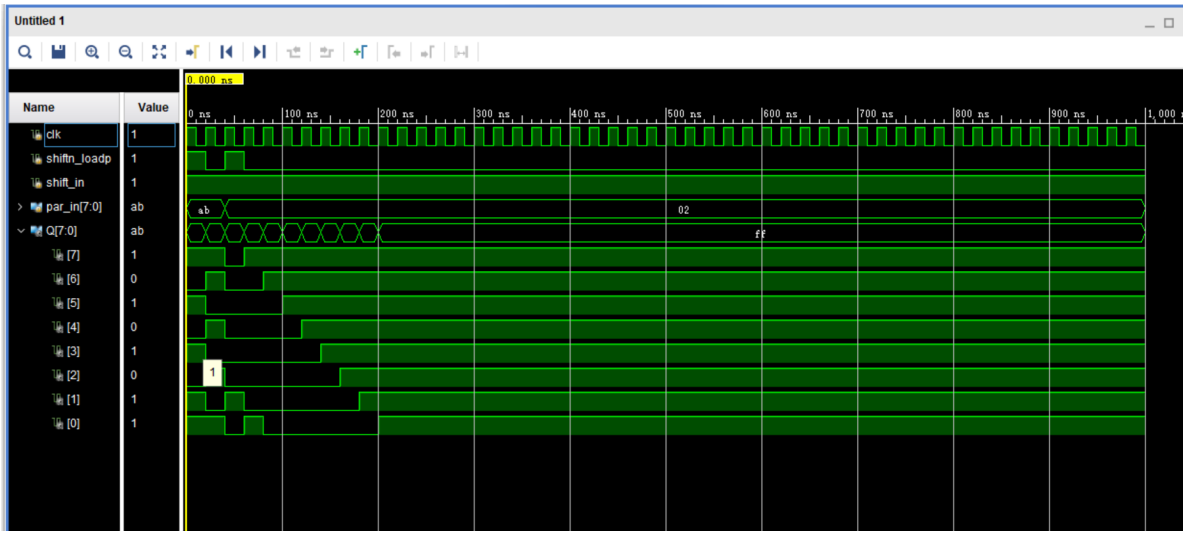
4、文件组织结构

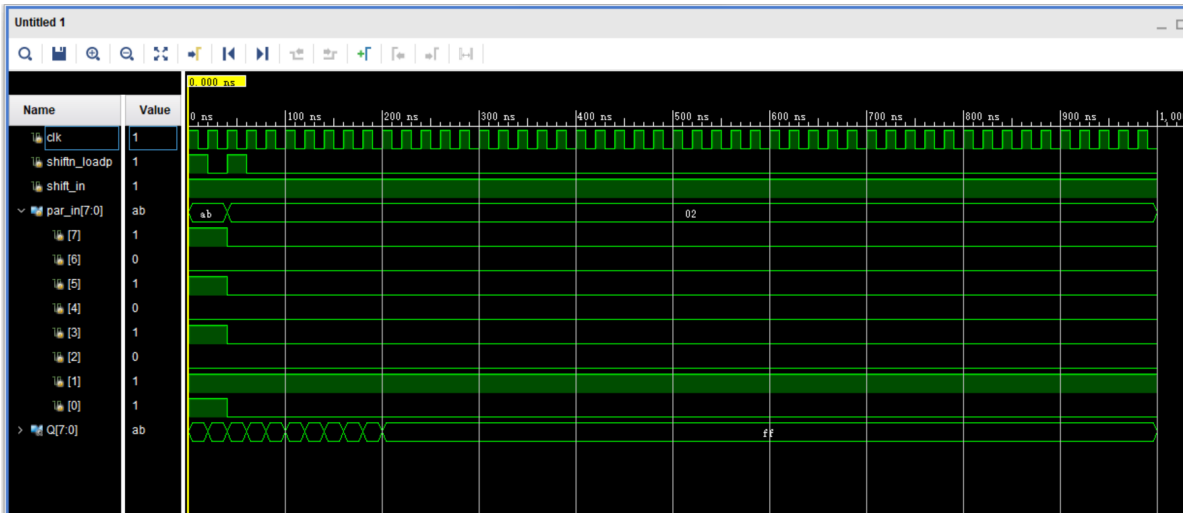


如图所示，仿真结果在后文体现

二、实验结果与分析

1、移位寄存器设计仿真结果



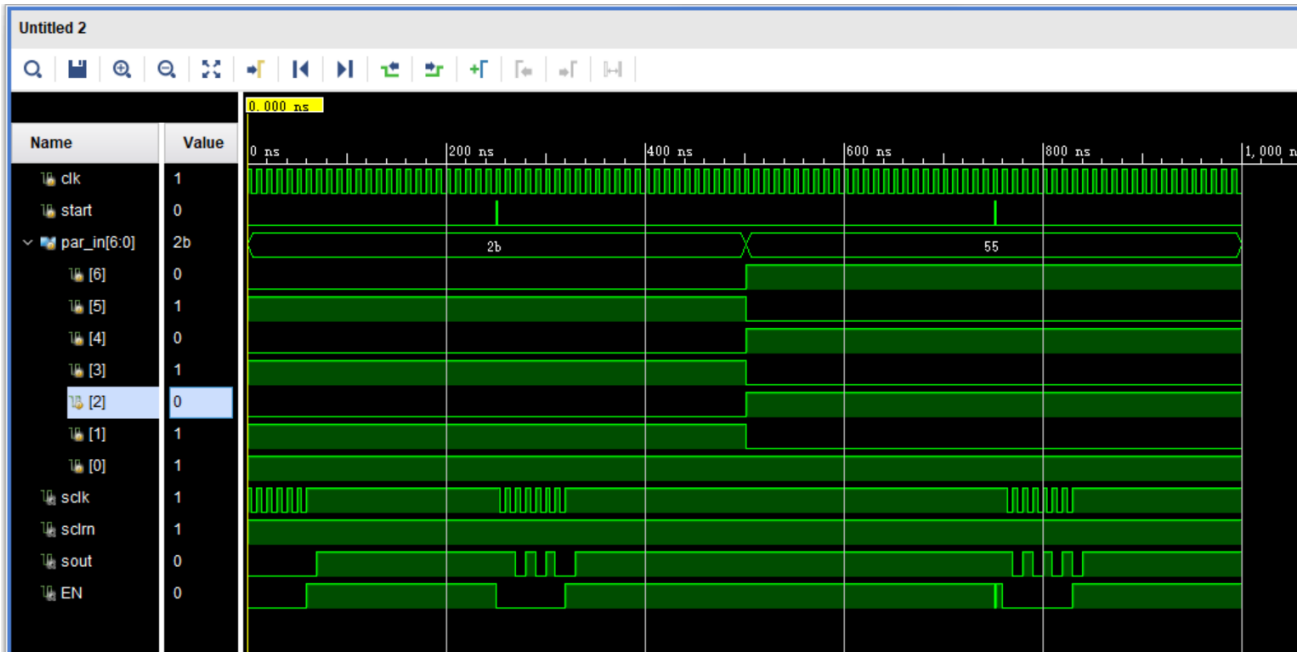


如图，串行/并行输入信号 Shift—/Load 为低位时进行串入串出，在每个时钟上升沿进行右移同时将串行输入的 shift_in 信号移入最左边的触发器中；串行/并行输入信号为高位时进行并行写入，将并行输入端口数据 par_in 写入触发器中。

在数据读入阶段，Q 和 part-in 数据相同，在移位阶段，可以看到 Q 的最低位轮流写入 shift-in 的信号 1，说明移位寄存器设计正确。

2、移位寄存器的应用

P2S 的仿真结果：



7 位 P2S 模块。

初始时 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

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

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。

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。

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]}，最高位存在一个 0，因此一定有 finish=0，sclk = finish | clk，此时 sclk 值和 clk 完全相同，即输出的时间点和移位寄存器进行一次右移输出的时间点相同，ser_out 每一个时钟周期输出最低位，右移一位，高位补 1，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

finish 置 1，表示串行输出结束，sclk 置 1，不再存在“上升沿”，q=0，且 set 和 reset 信号均为 0，保持。等待下一个 start 信号，重新传输。

三、讨论、心得

这是数字逻辑设计这门课的最后一次实验，实验过程中，我遇到了很多困难。比如不知道模块怎么补全，不理解时序电路的逻辑，不知道怎么用代码实现等，不过都在自己的网上学习和求助中得以解决。这学期的数逻实验结束了，我也学会了很多有用的知识，实验课对理论课的记忆和理解很有帮助，理论课也为实验课的操作打下基础，相辅相成地让我们对数字电路有了比较好的理解。很感谢老师一学期的付出!!

四、个人生活照片