



本科实验报告

课程名称: 数字逻辑电路设计

姓 名: NoughtQ

学 院: 计算机科学与技术学院

专 业: 计算机科学与技术

邮 箱:

QQ 号:

电 话:

指导教师: 洪奇军

报告日期: 2024 年 5 月 9 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: 加法器、加减法器和 ALU 基本原理与设计

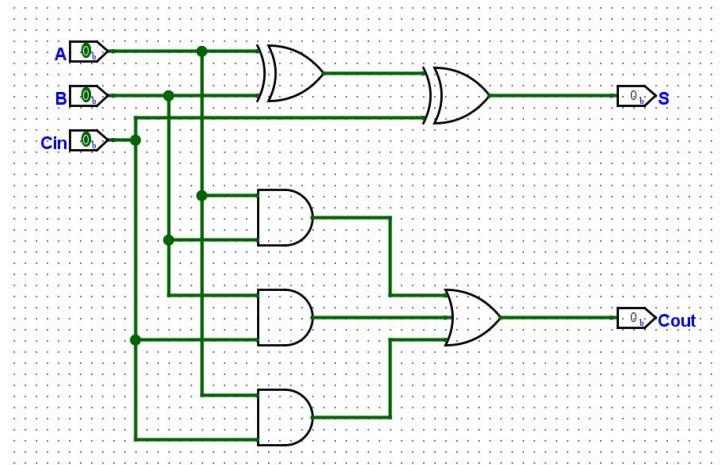
学生姓名: 钱梓洋 学号: 3230103502 同组学生姓名: 官欣

实验地点: 紫金港东四 511 室 实验日期: 2024 年 4 月 25 日

一、操作方法与实验步骤

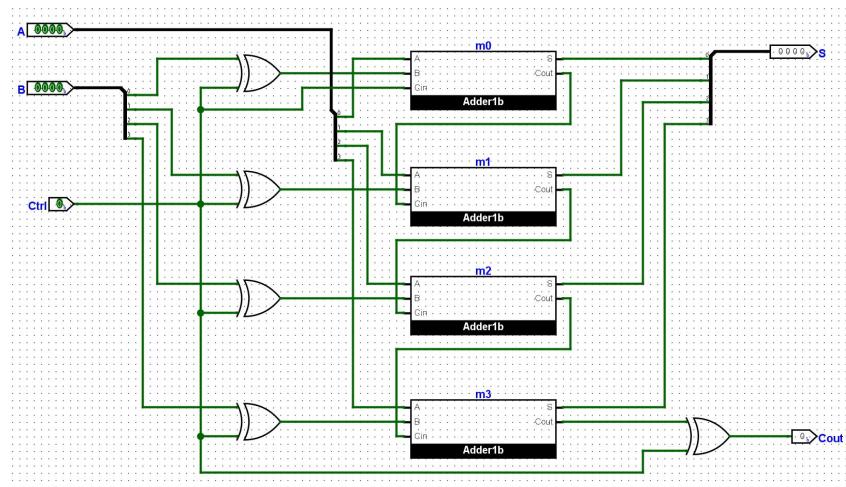
1. 原理图方式设计 4 位加减法器

(1) 在 Logisim 中新建项目 lab8, 然后绘制一位全加器的原理图 Adder1b, 如图所示:



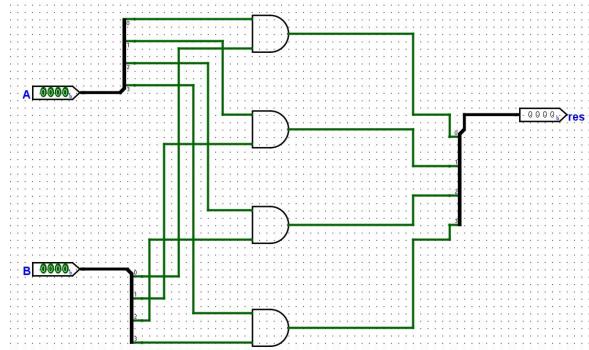
原理图 1

(2) 绘制四位加减法器的原理图 AddSub4b, 如图所示:

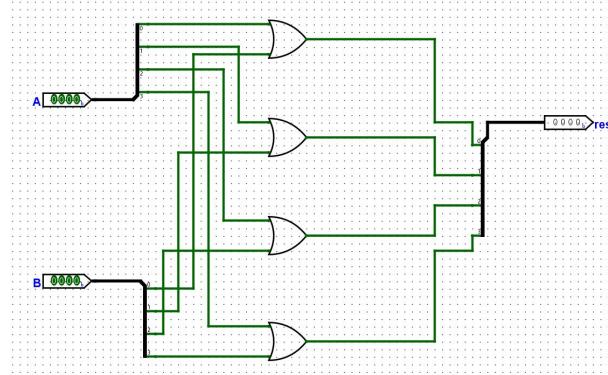


原理图 2

(3) 分别绘制四位输入按位与的原理图 And2b4 和四位输入按位或的原理图 Or2b4, 如图所示:

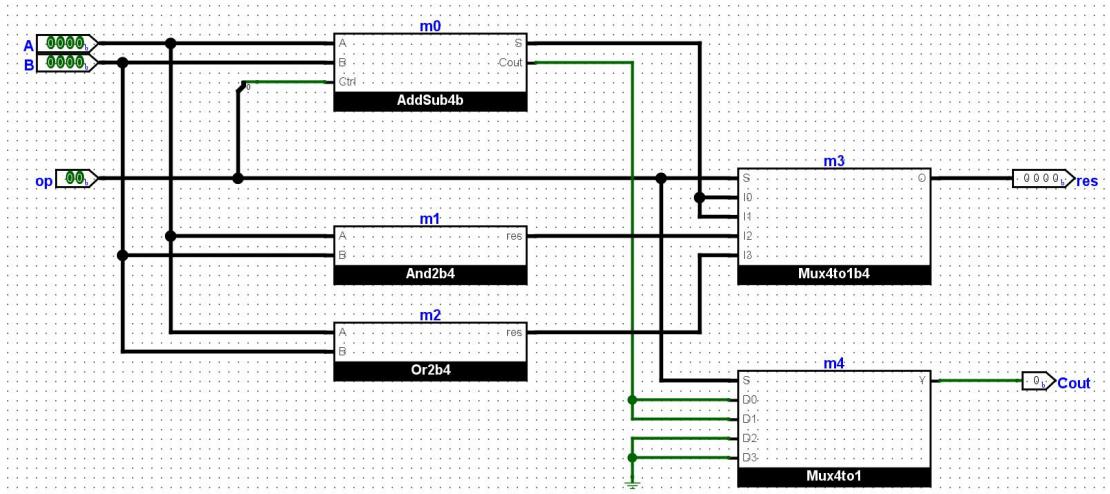


原理图 3(And2b4)

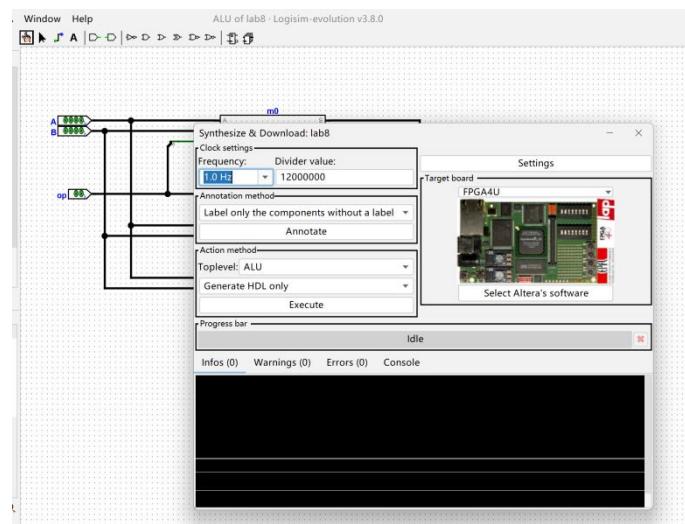


原理图 4(Or2b4)

(4) 利用 (2) (3) 两步中构建的模块, 以及 lab7 构建的 Mux4to1 和 Mux4to1b4 多路选择器模块, 绘制四位 ALU 的原理图 ALU, 如图所示。然后导出为 Verilog 代码。



原理图 5



(5) 在 Vivado 中新建工程 ALU，将上一步中导出的 Verilog 代码导入至工程中，然后创建仿真文件，仿真代码如下：

```
`timescale 1ns / 1ps
```

```
module alu_sim();
    // Inputs
    reg [3:0] A;
    reg [3:0] B;
    reg [1:0] op;
    // Outputs
    wire [3:0] res;
    wire Cout;

    ALU alu_inst(
        .A(A),
        .B(B),
        .op(op),

```

```

    .res(res),
    .Cout(Cout)
);
initial begin
    op = 2'b00; A = 4'b1100; B = 4'b0011; #50;
    A = 4'b1111; B = 4'b1111; #50;
    op = 2'b01; A = 4'b1100; B = 4'b11; #50;
    A = 4'b0011; B = 4'b1100; #50;
    op = 2'b10; A = 4'b0011; B = 4'b0110; #50;
    A = 4'b1110; B = 4'b0011; #50;
    op = 2'b11; A = 4'b0011; B = 4'b0110; #50;
    A = 4'b1110; B = 4'b0011; #50;
end

endmodule

```

(6) 点击左侧的“Run Synthesis”进行综合运行。成功运行完成后，再点击“Run Simulation”，生成仿真波形，观察波形图是否和预期结果一致。

2. 实现 4 位 ALU 及应用设计

(1) 去抖动模块：因为按键过程中会有机械原因导致的信号抖动，在上个实验中也可以观察到，一次按下按钮数字会跳动多次。因此，在 Vivado 中创建去抖动模块 pbdebounce.v，代码如下：

```

module pbdebounce(
    input wire clk,
    input wire button,
    output reg pbreg
);

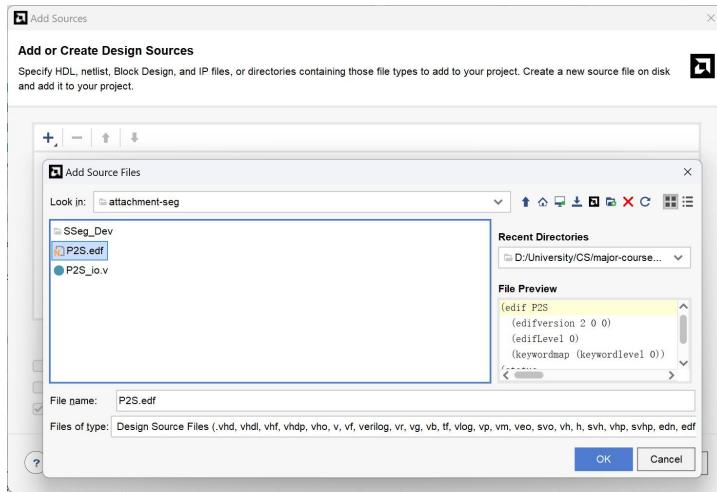
reg [7:0] pbshift;

always@(posedge clk) begin
    pbshift = pbshift<<1;
    pbshift[0] = button;
    if (pbshift==8'b0)
        pbreg=0;
    if (pbshift==8'hFF)
        pbreg=1;
end

endmodule

```

(2) 七段数码管静态显示：我们希望使用 SWORD 板上的 8 个七段数码管，它接收串行数据以减少管脚使用。因此，我们需要一个模块将 64 位的七段数码管数据转换为 1 位串行输出以及相应的同步信号，即并转串模块 P2S。本实验中只提供接口文件与用于综合的 EDIF 格式文件，目前只需要将 P2S_io.v 和 P2S.edf 添加进工程即可，需要注意将工程中 P2S.edf 的文件格式设置为 EDIF（右键文件选择 Set file type）。



(3) **CreateNumber** 模块：在 Vivado 中创建 CreateNumber 模块 CreateNumber.v，实现按键的自增或自减，代码如下：

```
/*
Module Name: CreateNumber
Description:
To change the value printed on Arduino using btns.
You will get a initial value printed as the para. INIT_HEXES defined.
After each pression on btn, a number will increase by 1.

This new module can handle i-1 when signal sw is 1
*/

module CreateNumber#(
parameter INIT_HEXES = 16'b1010_1011_1100_1101 // Init with "AbCd"
) (
input[3:0] btn,
input[3:0] sw,
output reg[15:0] num
);
wire[3:0] A, B, C, D;

initial num <= INIT_HEXES;

// D(the next num[3:0]) is always greater than current num[3:0] by 1
AddSub4b a0(.A(num[15:12]), .B(4'b0001), .Ctrl(sw[3]), .S(A));
```

```

AddSub4b a1(.A(num[11: 8]), .B(4'b0001), .Ctrl(sw[2]), .S(B));
AddSub4b a2(.A(num[ 7: 4]), .B(4'b0001), .Ctrl(sw[1]), .S(C));
AddSub4b a3(.A(num[ 3: 0]), .B(4'b0001), .Ctrl(sw[0]), .S(D));

// When pressing btn[0] num[3:0]++
always @(posedge btn[3]) num[15:12] <= A;
always @(posedge btn[2]) num[11: 8] <= B;
always @(posedge btn[1]) num[ 7: 4] <= C;
always @(posedge btn[0]) num[ 3: 0] <= D;

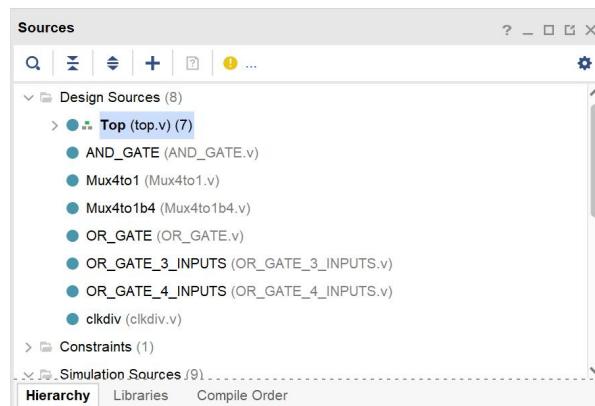
endmodule

```

(4) 设计顶层模块 top.v，它下面应当包括以下子模块：

- **clkdiv**: 时钟分频模块，直接使用 Lab7 提供的代码。
- **pbdebounce, CreateNumber**: 使用本实验实验指导提供的代码。
- **ALU**: 使用你在本实验中完成的模块。
- **DisplayNumber**: 使用你在 Lab7 中完成的模块，并引入其子模块。
- **Sseg_Dev**: 静态译码模块，使用附件中的模块。

如图所示：



顶层模块代码如下：

```

module Top(
    input wire clk,
    input wire [1:0] BTN,
    input wire [1:0] SW1,
    input wire [1:0] SW2,
    input wire [11:0] SW,
    output wire [3:0] AN,

```

```

        output wire [7:0] SEGMENT,
        output wire BTNX4,
        output wire seg_clk,
        output wire seg_clrn,
        output wire seg_sout,
        output wire SEG_PEN
    );
    wire [15:0] num;
    wire [1:0] btn_out;
    wire [3:0] res;
    wire Co;
    wire [31:0] clk_div;
    wire [15:0] disp_hexs;
    wire [15:0] disp_hexs_my;

    assign disp_hexs[15:12] = num[7:4]; // B
    assign disp_hexs[11:8] = num[3:0]; // A
    assign disp_hexs[7:4] = {3'b000, Co};
    assign disp_hexs[3:0] = res[3:0]; // C

    /* Code here */
    assign disp_hexs_my = (16'b0011010100000010); // Fill the last four
digits of your student id in ()
    assign BTNX4 = 1'b0;

    clkdiv m2(.clk(clk), .rst(0), .div_res(clk_div));
    pbdebounce
m0(.clk(clk_div[17]), .button(BTN[0]), .pbreg(btn_out[0]));
    pbdebounce
m1(.clk(clk_div[17]), .button(BTN[1]), .pbreg(btn_out[1]));

    CreateNumber m3(.btn({2'b0, btn_out}), .sw({2'b0, SW1}), .num(num)); // Attachment

    // The ALU module you wrote
    ALU m5(.A(num[3:0]),
            .B(num[7:4]), // fill sth.
in () .op(SW2[1:0]), // fill sth.
in () .res(res[3:0]), // fill sth.
in () .Cout(Co));

```

```

// Module you design in Lab7
DisplayNumber
m6(    .clk(clk), .hexs(disp_hexs), .LEs(4'b1111),           // fill sth. in
()                                .points(SW[3:0]), .rst(1'b0),
// fill sth. in ()
                                .AN(AN), .SEGMENT(SEGMENT));

// Attachment
SSeg_Dev m7(.clk(clk), .flash(clk_div[25]), .Hexts({disp_hexs_my,
disp_hexs}), .LES(SW[11:4]),
.Point({4'b0000,
SW[3:0]}), .rst(1'b0), .Start(clk_div[20]), .seg_clk(seg_clk),
.seg_clrn(seg_clrn), .SEG_PEN(SEG_PEN), .seg_sout(seg
_sout));

endmodule

```

(5) 创建引脚约束文件 constraints.v，代码如下所示。然后生成 bit 流文件，下板检验。

```

# Filename: constraints_lab8.xdc
## Constraints file for Lab8

# 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 {SW1[0]}]
set_property PACKAGE_PIN AE13 [get_ports {SW1[1]}]

```

```

set_property PACKAGE_PIN AF13 [get_ports {SW2[0]}]
set_property PACKAGE_PIN AF10 [get_ports {SW2[1]}]
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 {SW1[0]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW1[1]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW2[0]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW2[1]}]

# Key as inputs
set_property PACKAGE_PIN W16 [get_ports BTNX4]
set_property IOSTANDARD LVCMOS18 [get_ports BTNX4]
set_property PACKAGE_PIN V14 [get_ports {BTN[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {BTN[1]}]
set_property PACKAGE_PIN W14 [get_ports {BTN[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports {BTN[0]}]

set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets BTN*]

# 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 PACKAGE_PIN M24 [get_ports {seg_clk}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg_clk}]
set_property PACKAGE_PIN M20 [get_ports {seg_clrn}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg_clrn}]
set_property PACKAGE_PIN L24 [get_ports {seg_sout}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg_sout}]
set_property PACKAGE_PIN R18 [get_ports {SEG_PEN}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEG_PEN}]

# # 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 AE21 [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 PACKAGE_PIN P18 [get_ports LEDEN]
# set_property IOSTANDARD LVCMOS33 [get_ports LEDCLK]
# set_property IOSTANDARD LVCMOS33 [get_ports LEDCLR]
# set_property IOSTANDARD LVCMOS33 [get_ports LEDDT]
# set_property IOSTANDARD LVCMOS33 [get_ports LEDEN]

# #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]}]

```

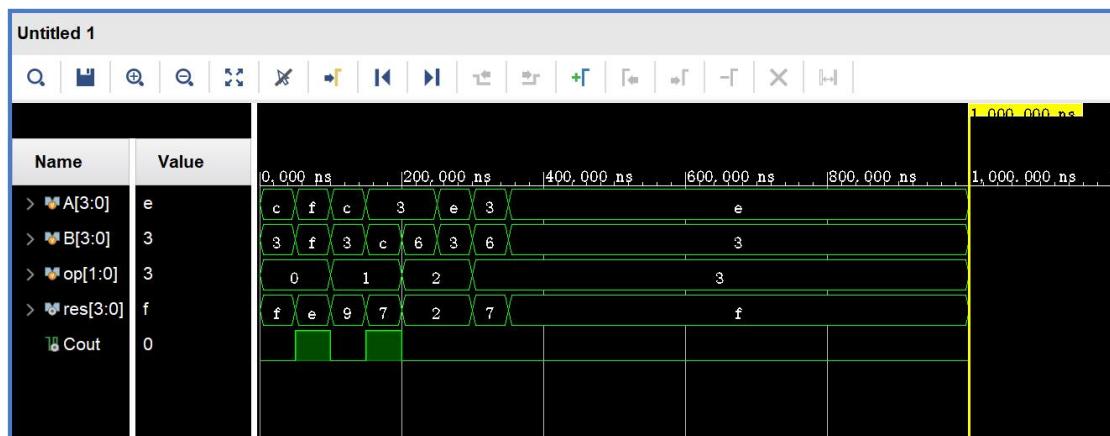
```

# 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 M21 [get_ports vga_vs]
# set_property IOSTANDARD LVCMOS33 [get_ports vga_hs]
# set_property IOSTANDARD LVCMOS33 [get_ports vga_vs]

```

二、实验结果与分析

1. 原理图方式设计 4 位加减法器



ALU 模块的仿真波形图

分析：

- (1) $op == 00$: 执行加法操作
 - $c + 3 == f$
 - $f + f == 1e$
- (2) $op == 01$: 执行减法操作
 - $c - 3 == 9$
 - $3 - c = 17$, 即 $00011 + 10100$ (c 的补码) $= 10111$
- (3) $op == 10$: 执行按位与操作
 - $0011 \& 0110 = 0010$
 - $1110 \& 0011 = 0010$
- (4) $op == 11$: 执行按位或操作
 - $0011 | 0110 = 0111$
 - $1110 | 0011 = 1111$

2. 实现 4 位 ALU 及应用设计

实验结果	说明
	<p>初始状态： · 左边四位显示我的学号 · 初始运算方式为加法： $7 + 2 = 9$</p>
	<p>运算方式为减法： $7 - 2 = 5$</p>
	<p>运算方式为按位与： $0111 \& 0010 = 0010$</p>

	<p>运算方式为按位或： $0111 \mid 0010 = 0111$</p>
	<p>数 A(从右往左第 3 位)和数 Res(从右往左第 1 位)显示小数点</p>
	<p>数 B 增加 1</p>

	数 A 增加 1
	数 A 减少 1
	数 B 减少 1



学号改变为同组同学的学号，

用大的数减小的数：

$$6 - 9 = 1d$$

即， $00110 + 10111 = 11101$

三、讨论、心得

这次实验有可能是这几次以来最顺利的一次——经过前几次的磨练后，我终于不再为某些愚蠢的错误而浪费很多时间。尽管如此，我认为这次实验还是比较复杂的，因为这次实验的顶层模块中包含了更多的子模块，而且有几个是上次实验设计过的（所以，如果上次实验没做好，这次实验也就无法完成了）；而且还要求我们完全独立设计仿真代码，并且要填补在 top.v 上挖的空，因此这对我们的 Verilog 掌握水平提出了更高的要求。基于前面的积累，我按部就班地完成每个步骤，还算比较顺利地实现了 ALU 的功能——除了因为 top.v 文件中的 point 引脚和我原来的多路选择器 Point 引脚的命名不一样，而需要我在十几处地方修改代码。

通过这次实验，我不仅对 Verilog 代码的掌握又“更上一层楼”，而且更熟悉 Vivado 的操作（包括顶层模块与个子模块之间的关系等等）。希望我能继续保持这个积极的势头，顺利完成之后的实验！

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: 锁存器与触发器基本原理

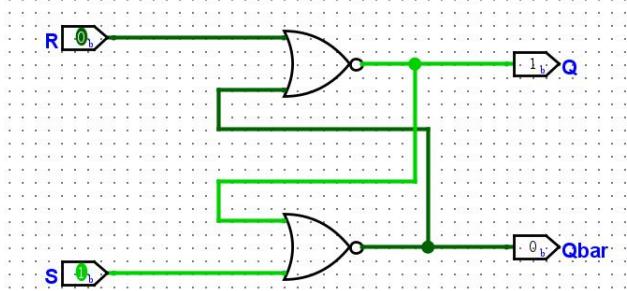
学生姓名: 钱梓洋 学号: 3230103502 同组学生姓名: 官欣

实验地点: 紫金港东四 511 室 实验日期: 2024 年 4 月 25 日

一、操作方法与实验步骤

1. 实现基本 SR 锁存器, 验证功能和存在的时序问题

(1) 在 Logisim 中新建项目 lab9, 然后绘制 SR 锁存器的原理图 SR_latch, 如图所示。然后导出为 Verilog 代码。



原理图 1

(2) 在 Vivado 中新建工程 Latches_and_Flip_flops。然后将刚刚导出的 Verilog 代码导入到该工程中。

(3) 接着创建仿真文件, 测试 SR 锁存器的各种输入组合。仿真代码如下:

```
`timescale 1ns / 1ps
module SR_latch_sim();
// Inputs
reg R;
reg S;
// Outputs
wire Q;
wire Qbar;
SR_latch SR1(
.R(R),
```

```

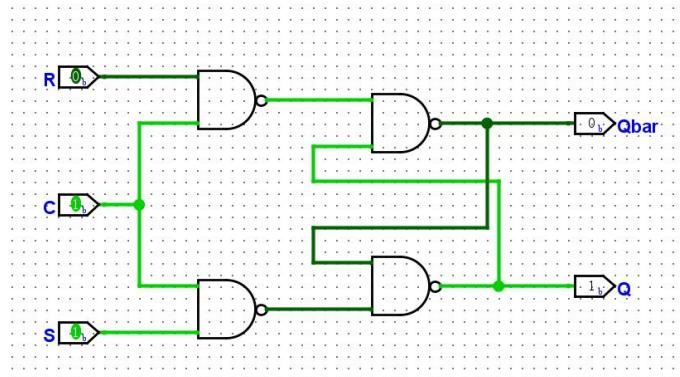
.S(S),
.Q(Q),
.Qbar(Qbar)
);
initial begin
    R = 1'b1; S=1'b0; #50;
    R = 1'b0; S=1'b0; #50;
    R = 1'b0; S=1'b1; #50;
    R = 1'b0; S=1'b0; #50;
    R = 1'b1; S=1'b0; #50;
    R = 1'b0; S=1'b0; #50;
    R = 1'b0; S=1'b1; #50;
    R = 1'b0; S=1'b0; #50;
end
endmodule

```

(4) 点击左侧“Run Simulation”，生成仿真波形，检查输出结果是否符合预期。

2. 实现门控 SR 锁存器，并验证功能和存在的时序问题

(1) 在 Logisim 中绘制门控 SR 锁存器的原理图 C_SR_latch，如图所示。然后导出为 Verilog 代码。



原理图 2

(2) 将刚刚导出的 Verilog 代码导入到 Vivado 工程中。然后创建仿真文件，测试门控 SR 锁存器的各种输入组合，并且在一个时钟周期中对门控 SR 锁存器的值进行多次修改，观察它的空翻现象。仿真代码如下：

```

`timescale 1ns / 1ps
module C_SR_latch_sim();
// Inputs
reg R;
reg S;
reg C;
// Outputs

```

```

wire Q;
wire Qbar;
C_SR_latch CSR1(
    .R(R),
    .S(S),
    .C(C),
    .Q(Q),
    .Qbar(Qbar)
);
initial begin
    C = 1'b1;
    R = 1'b1; S=1'b0; #30;
    R = 1'b0; S=1'b0; #30;
    R = 1'b0; S=1'b1; #30;
    R = 1'b0; S=1'b0; #30;
    R = 1'b1; S=1'b0; #30;
    R = 1'b0; S=1'b0; #30;
    R = 1'b0; S=1'b1; #30;
    R = 1'b0; S=1'b0; #30;

    C = 1'b0;
    R = 1'b1; S=1'b0; #30;
    R = 1'b0; S=1'b0; #30;
    R = 1'b0; S=1'b1; #30;
    R = 1'b0; S=1'b0; #30;
    R = 1'b1; S=1'b0; #30;
    R = 1'b0; S=1'b0; #30;
    R = 1'b0; S=1'b1; #30;
    R = 1'b0; S=1'b0; #30;

    C = 1'b1;
    R = 1'b1; S=1'b0; #30;
    R = 1'b0; S=1'b0; #30;
    R = 1'b0; S=1'b1; #30;
    R = 1'b0; S=1'b0; #30;
    R = 1'b1; S=1'b0; #30;
    R = 1'b0; S=1'b0; #30;
    R = 1'b0; S=1'b1; #30;
    R = 1'b0; S=1'b0; #30;

    C = 1'b0;
    R = 1'b1; S=1'b0; #30;
    R = 1'b0; S=1'b0; #30;
    R = 1'b0; S=1'b1; #30;

```

```

R = 1'b0; S=1'b0; #30;
R = 1'b1; S=1'b0; #30;
R = 1'b0; S=1'b0; #30;
R = 1'b0; S=1'b1; #30;
R = 1'b0; S=1'b0; #30;

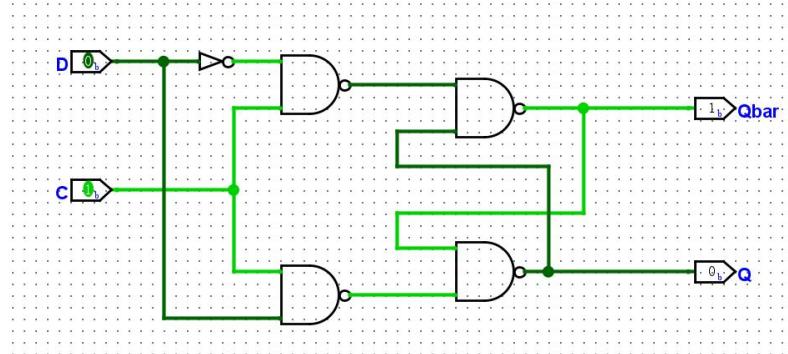
end
endmodule

```

(3) 点击左侧“Run Simulation”，生成仿真波形，检查输出结果是否符合预期。

3. 实现 D 锁存器，并验证功能和存在的时序问题

(1) 在 Logisim 中绘制门控 D 锁存器的原理图 C_D_latch，如图所示。然后导出为 Verilog 代码。



原理图 3

(2) 将刚刚导出的 Verilog 代码导入到 Vivado 工程中。然后创建仿真文件，测试门控 D 锁存器的各种输入组合，并且在一个时钟周期中对门控 D 锁存器的值进行多次修改，观察它的空翻现象。仿真代码如下：

```

`timescale 1ns / 1ps
module C_D_latch_sim();
// Inputs
reg D;
reg C;
// Outputs
wire Q;
wire Qbar;
// Instantiate the Unit Under Test (UUT)
C_D_latch uut(
.D(D),
.C(C),
.Q(Q),
.Qbar(Qbar)
);
initial begin
// Initialize Inputs

```

```

D = 0;
C = 0;
C=1;D=1; #50;
D=0; #50;
D=1;#50;
D=0;#50;
D=1;#50;
C=0;D=1; #50;
D=0;
end

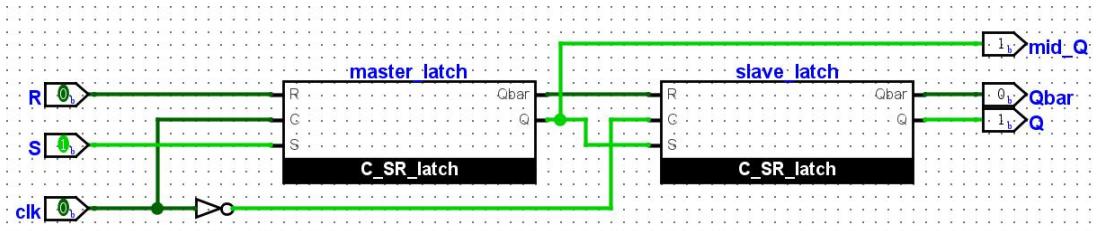
```

```
endmodule
```

(3) 点击左侧“Run Simulation”，生成仿真波形，检查输出结果是否符合预期。

4. 实现正边沿 SR 主从触发器，并验证功能和存在的时序问题

(1) 在 Logisim 中绘制正边沿 SR 主从触发器的原理图 MS_SR_flip_flop，如图所示。然后导出为 Verilog 代码。



原理图 4

(2) 将刚刚导出的 Verilog 代码导入到 Vivado 工程中。然后创建仿真文件，测试正边沿 SR 主从触发器的各种输入组合，并且在一个时钟周期中对正边沿 SR 主从触发器的值进行多次修改，观察是否出现一次性采样问题。仿真代码如下：

```
`timescale 1ns / 1ps
```

```

module MS_SR_flip_flop_sim();
// Inputs
reg S;
reg clk;
reg R;
// Outputs
wire mid_Q;
wire Q;
wire Qbar;
// Instantiate the Unit Under Test (UUT)
MS_SR_flip_flop uut(
    .S(S),
    .clk(clk),
    .R(R),

```

```
.mid_Q(mid_Q),
```

```
.Q(Q),
```

```
.Qbar(Qbar)
```

```
);
```

```
initial begin
```

```
// Initialize Inputs
```

```
S = 0;
```

```
clk = 0;
```

```
R = 0;
```

```
S=0;R=0;#25;S=1;R=0;#25;
```

```
S=0;R=1;#100;S=1;R=0;#100;
```

```
S=0;R=0;#100;S=0;R=1;#100;
```

```
S=1;R=1;#100;S=0;R=0;#100;
```

```
S=1;R=0;#10;
```

```
S=0;#5;R=1;#10;
```

```
S=0;R=0;#25;R=1;#5;
```

```
R=0;#5;S=0;R=0;#40;
```

```
S=1;#10;S=0;#50;
```

```
R=1;#10;R=0;
```

```
end
```

```
always begin
```

```
clk=0;#50;
```

```
clk=1;#50;
```

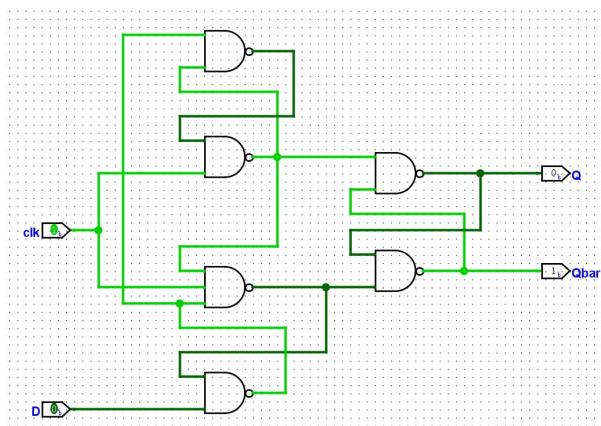
```
end
```

```
endmodule
```

(3) 点击左侧“Run Simulation”，生成仿真波形，检查输出结果是否符合预期。

5. 实现正边沿 D 触发器，并验证功能

(1) 在 Logisim 中绘制正边沿 D 触发器的原理图 ET_D_flip_flop，如图所示。然后导出为 Verilog 代码。



原理图 5

(2) 将刚刚导出的 Verilog 代码导入到 Vivado 工程中。然后创建仿真文件，测试正边沿 D 触发器的各种输入组合，并且在一个时钟周期中对正边沿 D 触发器的值进行多次修改，观察是否出现一次性采样问题。仿真代码如下：

```
'timescale 1ns / 1ps

module ET_D_flip_flop_sim();
// Inputs
reg D;
reg clk;
// Outputs
wire Q;
wire Qbar;

ET_D_flip_flop ET_D_flip_flop_inst(
.D(D),
.clk(clk),
.Q(Q),
.Qbar(Qbar)
);

initial begin
D=0;
clk=0; #50;
clk=1; #50;
D=1;
clk=0; #50;
clk=1; #50;
D=0;
clk=0; #50;
clk=1; #50;
D=1;
clk=0; #50;
clk=1; #50;
D=0;
clk=0; #50;
clk=1; #50;
D=1;
clk=0; #50;
clk=1; #50;
D=0;
clk=0; #50;
clk=1; #50;
D=1;
clk=0; #50;
clk=1; #50;
D=0;
clk=0; #50;
clk=1; #50;
D=1;
clk=0; #50;
```

```

clk=1; #50;
end
endmodule

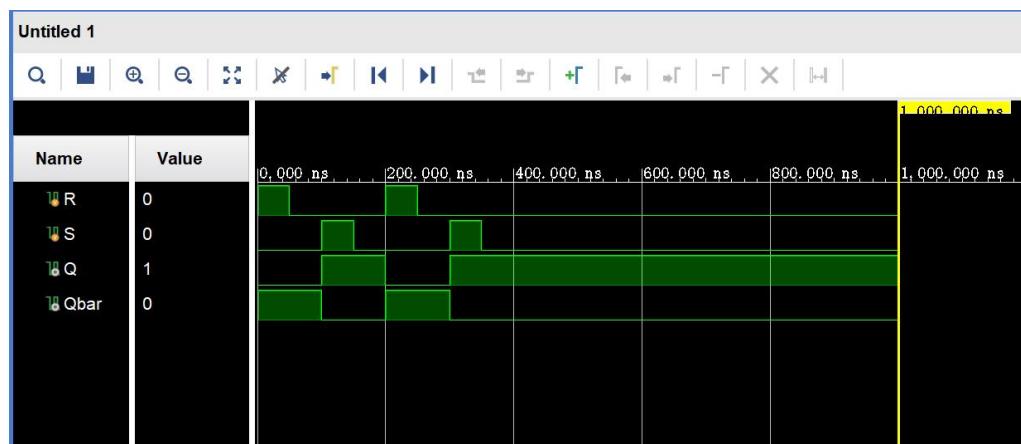
```

(3) 点击左侧“Run Simulation”，生成仿真波形，检查输出结果是否符合预期。

二、实验结果与分析

1. 实现基本 SR 锁存器，验证功能和存在的时序问题

<i>RS</i>	<i>QQn</i>	说明
0 0	<i>QQn</i>	保持
0 1	1 0	置1
1 0	0 1	置0
1 1	0 0	未定义

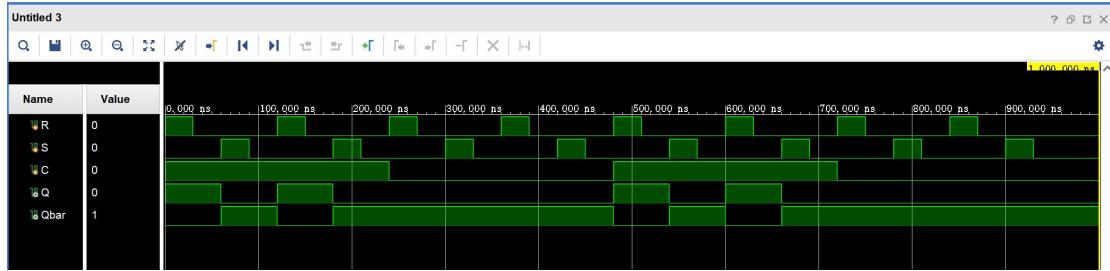


波形图 1

将仿真波形的结果与真值表进行比对，不难看出结果符合预期，因此 RS 锁存器的功能正确实现。

2. 实现门控 SR 锁存器，并验证功能和存在的时序问题

<i>C RS</i>	<i>QQn</i>	说明
0xx	<i>QQn</i>	保持
1 0 0	<i>QQn</i>	保持
1 0 1	1 0	置 1
1 1 0	0 1	置 0
1 1 1	1 1	未定义



波形图 2

将仿真波形的结果与真值表进行比对，不难看出结果符合预期；而且在一个时钟周期内，门控 RS 锁存器发生空翻现象（C=1 的时候，观察 Q 和 Qbar，发现它们变化了 3 次电平），因此门控 RS 锁存器的功能正确实现。

3. 实现 D 锁存器，并验证功能和存在的时序问题

C D	Q Qn	说明
0 ×	Q Qn	保持
1 0	0 1	置0
1 1	1 0	置1

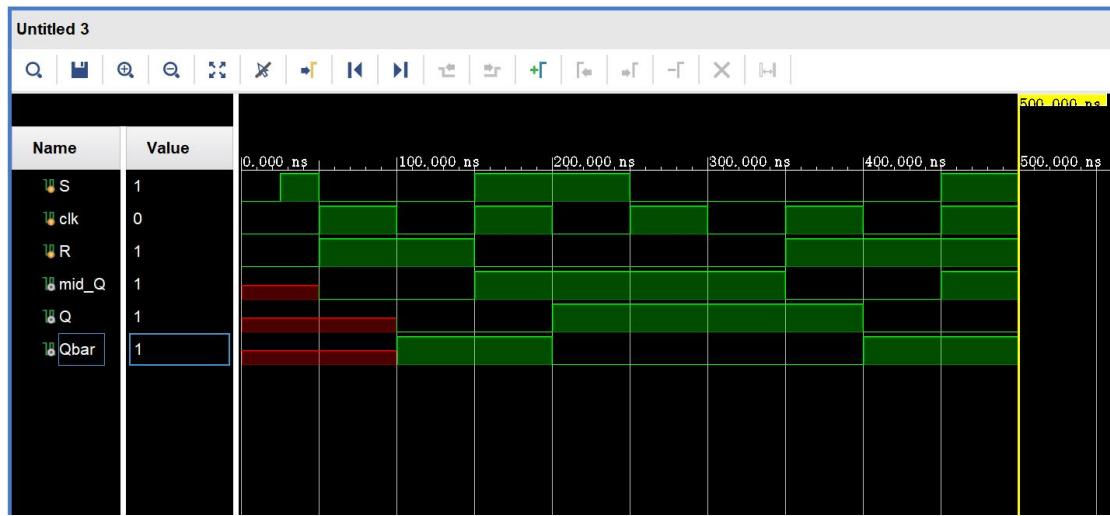


波形图 3

将仿真波形的结果与真值表进行比对，不难看出结果符合预期；而且在一个时钟周期内，门控 D 锁存器发生空翻现象（C=1 的时候，观察 Q 和 Qbar，发现它们变化了 4 次电平），因此门控 D 锁存器的功能正确实现。

4. 实现正边沿 SR 主从触发器，并验证功能和存在的时序问题

C R S	Q Qn	说明
0 × ×	Q Qn	保持
1 0 0	Q Qn	保持
1 0 1	1 0	置1
1 1 0	0 1	置0
1 1 1	1 1	未定义

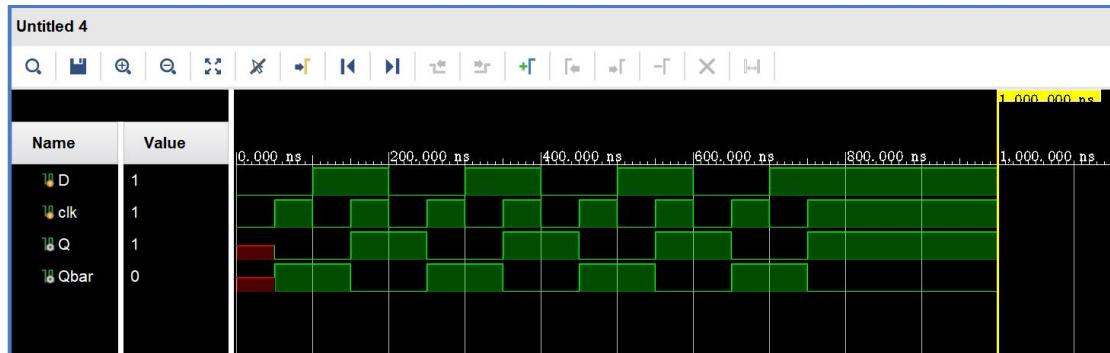


波形图 4

将仿真波形的结果与真值表进行比对，不难看出结果符合预期；而且在一个时钟周期内，SR 触发器的输入发生多次改变，但只有在上升沿和下降沿处输出发生改变，因此 SR 触发器的功能正确实现。

5. 实现正边沿 D 触发器，并验证功能

异步控制		上升沿触发			
R	S	C _P	D	Q	Q
0	1	×	×	0	1
1	0	×	×	1	0
1	1	↑	0	0	1
1	1	↑	1	1	0



波形图 5

将仿真波形的结果与真值表进行比对，不难看出结果符合预期；而且在一个时钟周期内，D 触发器的输入发生多次改变，但只有在上升沿和下降沿处输出发生改变，因此 D 触发器的功能正确实现。

三、讨论、心得

我原以为这个实验应该比较简单，因为整个实验就包含五个仿真实验，都不用上板操作，但没想到我还是在一些地方遇到了问题：在绘制正边沿 SR 主从触发器时，由于没有我并没有直接按照课件上给的图画，而是照着助教文档那个 D 主从触发器修改，导致原理图中非门的位置弄错了，从而在运行仿真时卡在某个时刻不动了。我比对了半天才发现这个问题。其次，由于这次实验要将五个模块的 Verilog 文件导入到一个 Vivado 工程中，在运行仿真时需要注意运行的是哪一个模块的仿真。为了安全起见，每当我要运行某个模块的仿真文件时，我都会将相应的文件“Set as Top”，这样就可能避免运行时的错误，同时也提醒自己现在在干什么。

这个实验主要暴露的问题是我还未熟悉锁存器和触发器的知识点，导致图都没画好。现在刚刚学到时序电路，我还没有适应过来，希望在今后我能巩固好这一块知识点，为将来学习和实验打下良好的基础。

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: 同步时序电路设计

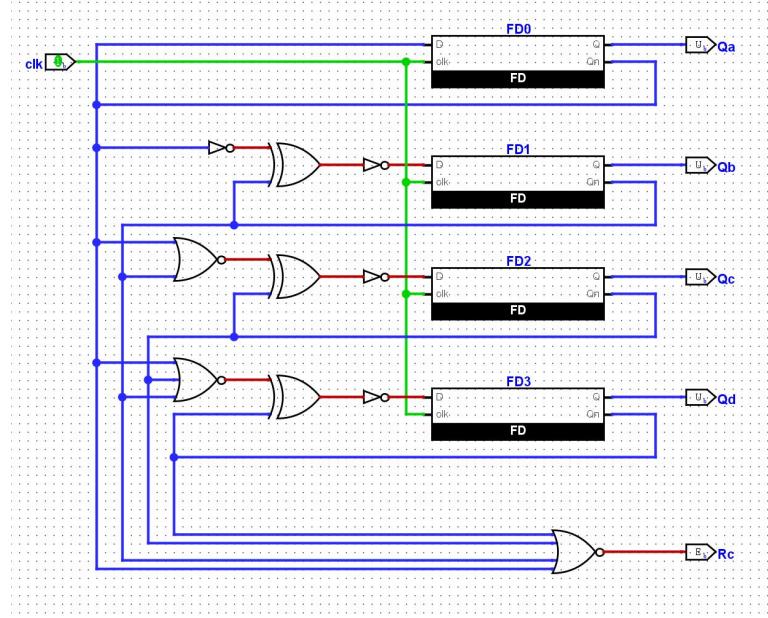
学生姓名: 钱梓洋 学号: 3230103502 同组学生姓名: 官欣

实验地点: 紫金港东四 511 室 实验日期: 2024 年 5 月 9 日

一、操作方法与实验步骤

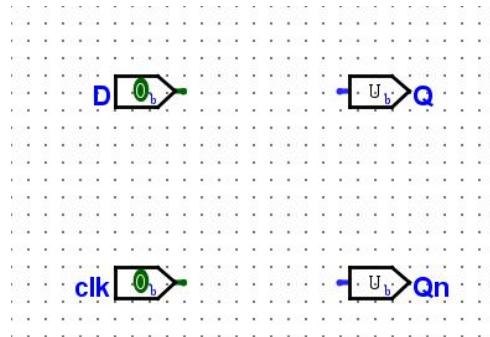
1. 原理图方式设计 4 位同步二进制计数器

(1) 在 Logisim 中新建项目 lab10, 然后绘制四位同步二进制计数器的原理图 Counter4b, 如图所示:



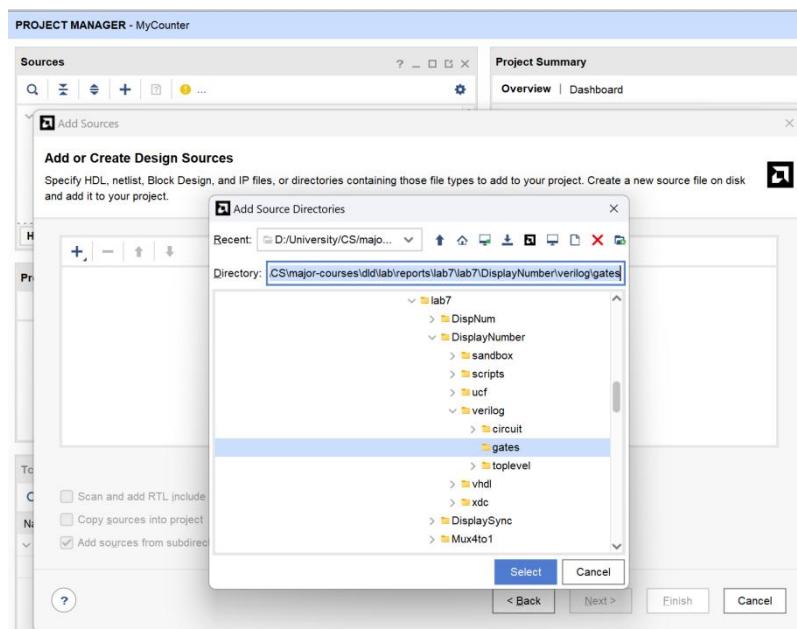
原理图 1(Counter4b)

其中 FD 模块表示 D 触发器, 但在 Lab9 中实现的 D 触发器没有重置或初始化的值, 因此在 Logisim 中画一个“空壳”即可, 之后会作修改。



原理图 2(FD)

(2) 在 Vivado 中新建工程 MyCounter, 然后将 Counter4b 的原理图导出为 Verilog 代码, 随后导入到 Vivado 工程中。接下来修改 FD.v 代码, 如下所示:



```
// FD.v
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
```

(3) 对 Counter4b 进行仿真， 仿真代码如下。仿真过程中，设定每 10 个单位时间翻转一次的时钟信号 clk，每次 clk 信号上升沿时，计数器的值应加 1，查看波形图是否符合预期。

```
// counter4b_sim.v
`timescale 1ns / 1ps
module counter4b_sim;
    // Inputs
    reg clk;
    // Outputs
    wire Qa;
    wire Qb;
    wire Qc;
    wire Qd;
    wire Rc;
    // Instantiate the Unit Under Test (UUT)
Counter4b uut (
    .Qa(Qa),
    .Qb(Qb),
    .Qc(Qc),
    .Qd(Qd),
    .Rc(Rc),
    .clk(clk)
);
    initial begin
        // Initialize Inputs
        clk = 0;
        #100;
    end
    always begin
        clk=clk^1;
        #10;
    end
endmodule
```

(4) 下板验证：

①添加子模块 clk_ls， 代码如下：

```
// clk_ls.v
`timescale 1ns / 1ps

module clk_ls(
    input clk,
    output reg clk_ls
```

```

);

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

```

②添加顶层模块 Top，代码如下(记得对 Top.v 进行“Set as Top”操作):

```

// Top.v
module Top(
    input wire clk,
    output wire LED,
    output wire [7:0] SEGMENT,
    output wire [3:0] AN
);

    wire Qa;
    wire Qb;
    wire Qc;
    wire Qd;
    wire [3:0] Hex;

    /* module clk_1s at submodules/clk_1s.v */
    clk_1s m0(.clk(clk), .clk_1s(clk_1s));

    /* You need to implement module Counter4b */
    Counter4b
    m1(.clk(clk_1s), .Qa(Qa), .Qb(Qb), .Qc(Qc), .Qd(Qd), .Rc(LED));

```

```

assign Hex = {Qd, Qc, Qb, Qa};

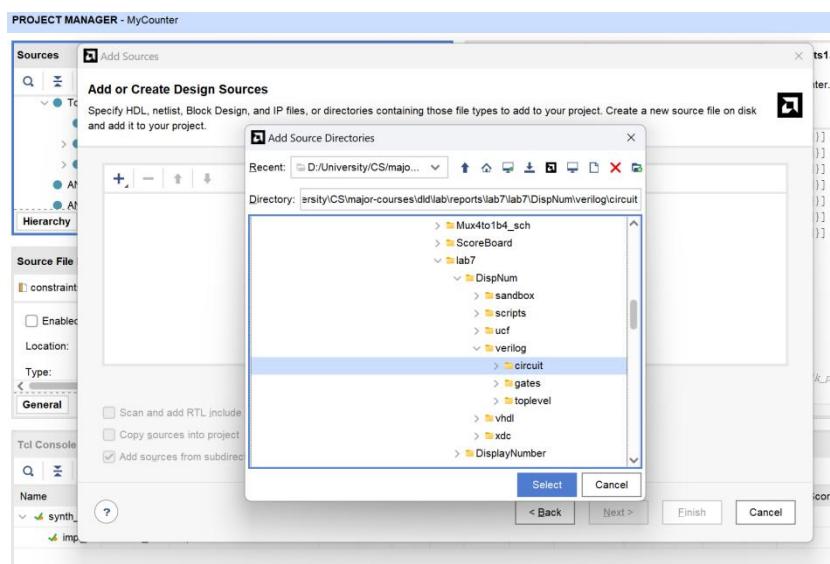
// Please replace module below with your module completed in Lab 6
// Pay attention to the correctness of the module name and port name
// NOTE: SEGMENT and Segement are different port names

// BTN[0]: LE, valid with value 0
// BTN[1]: point, light with value 1
// SW[7:4]: AN, light with value 1(AN[i] = ~SW[i+4])
// SW[3:0]: number to display
DispNum display(.BTN(2'b00), .SW({4'b0001,
Hex}), .SEGMENT(SEGMENT), .AN(AN));

endmodule

```

③将 Lab7 中的子模块 DispNum 的 Verilog 代码导入到工程中。



④添加引脚约束文件，代码如下：

```

# Filename: constraints_labA_part1.xdc
## Constraints file for LabA-part1

# 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"]

# LED
set_property PACKAGE_PIN AF24 [get_ports {LED}]

```

```

set_property IOSTANDARD LVCMOS33 [get_ports {LED}]

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]}]

```

⑤综合运行，烧录 bit 流文件，导入实验板进行验证，看结果是否符合预期。

2. 以 Verilog 行为描述方式设计 16 位可逆二进制同步计数器

(1) 在 Vivado 工程 MyCounter 中新建模块 RevCounter，代码如下：

(注意：与“四位可逆同步二进制计数器”相比，十六位可逆同步二进制计数器拓展位宽到 16 位，且添加了一个控制信号 s 用来选择“自增”或“自减”。)

```

// RevCounter.v
/** module RevCounter
 * input
 *   clk: A clock signal driven by module clk_1s.
 *   s: 0 for increment, 1 for decrement
 * output
 *   cnt: a 16-bits register
 *   Rc: rise when the counter reset(i.e. carry will be set), that is,
Rc becomes 1 when
*           increment(s=0 & cnt=F) or decrement(s=1, cnt=0)

```

```

/*
//! NOTE: DO NOT CHANGE THE MODULE NAME & PORT NAMES
module RevCounter(
    input wire clk,
    input wire rst,
    input wire s,
    output reg [15:0] cnt = 0,
    output wire Rc
);

    always @(posedge clk) begin
        if (rst == 1'b1) begin
            cnt <= 16'h0;
        end
        else begin
            if (s == 1'b0) begin
                cnt <= (cnt == 16'hFFFF) ? 16'h0 : cnt + 1;
            end
            else begin
                cnt <= (cnt == 16'h0) ? 16'hFFFF : cnt - 1;
            end
        end
    end
    assign Rc = (s == 1'b0) ? (cnt == 16'hFFFF) : (cnt == 16'h0);
endmodule

```

(2) 对 RevCounter 进行仿真，仿真代码如下（记得要禁用之前的仿真文件，并将该仿真文件进行“Set as Top”操作）。然后查看波形图是否符合预期。

```

// RevCounter_sim.v
`timescale 1ns / 1ps
module RevCounter_sim();
    // Inputs
    reg clk;
    reg rst;
    reg s;
    // Output
    wire Rc;
    wire [15:0] cnt;
    // Instantiate the UUT
    RevCounter RevCounter_inst (
        .clk(clk),
        .rst(rst),
        .s(s),
        .Rc(Rc),

```

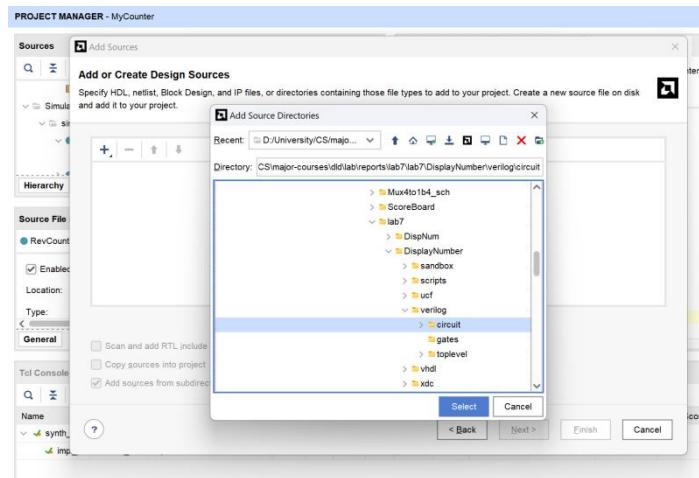
```

.cnt(cnt)
);
    integer i;
initial begin
    rst = 0;
    s = 0;
    for(i=0;i<4;i=i+1)begin
        clk=1#10;
        clk=0#10;
    end
    rst = 1;
    for(i=0;i<2;i=i+1)begin
        clk=1#10;
        clk=0#10;
    end
    rst = 0;
    s = 1;
    for(i=0;i<4;i=i+1)begin
        clk=1#10;
        clk=0#10;
    end
    rst = 1;
    for(i=0;i<2;i=i+1)begin
        clk=1#10;
        clk=0#10;
    end
    rst = 0;
end
endmodule

```

(3) 下板验证:

- ①导入 Lab7 中实现的 DisplayNumber 子模块。



②添加顶层模块 Top2，代码如下(记得对 Top2.v 进行“Set as Top”操作；以防万一，可以先禁用 Top 模块)：

```
// Top2.v
module Top2(
    input wire clk,
    input wire [1:0] SW,
    output wire LED,
    output wire [7:0] SEGMENT,
    output wire [3:0] AN
);

    wire[15:0] cnt;
    wire [3:0] Hex;
    wire clk_1s;

    /* module clk_100ms at submodules/clk_1s.v */
    clk_1s clk_div_1s (.clk(clk), .clk_1s(clk_1s));

    /* You need to implement module RevCounter */
    RevCounter
    counter(.clk(clk_1s), .rst(SW[1]), .s(SW[0]), .cnt(cnt), .Rc(LED));

    // Please replace module below with your module completed in Lab **7**
    // import submodules for module DisplayNumber from your prev. project
    DisplayNumber
    display(.clk(clk), .rst(1'b0), .hexs(cnt), .LEs(4'b0000), .points(4'b0000)
        , .AN(AN), .SEGMENT(SEGMENT));

endmodule
```

③添加引脚约束文件，代码如下：

```
# Filename: constraints_labA_part1.xdc
## Constraints file for LabA-part1

# 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"]

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]}]
```

```

# LED
set_property PACKAGE_PIN AF24 [get_ports {LED}]
set_property IOSTANDARD LVCMOS33 [get_ports {LED}]

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]}]

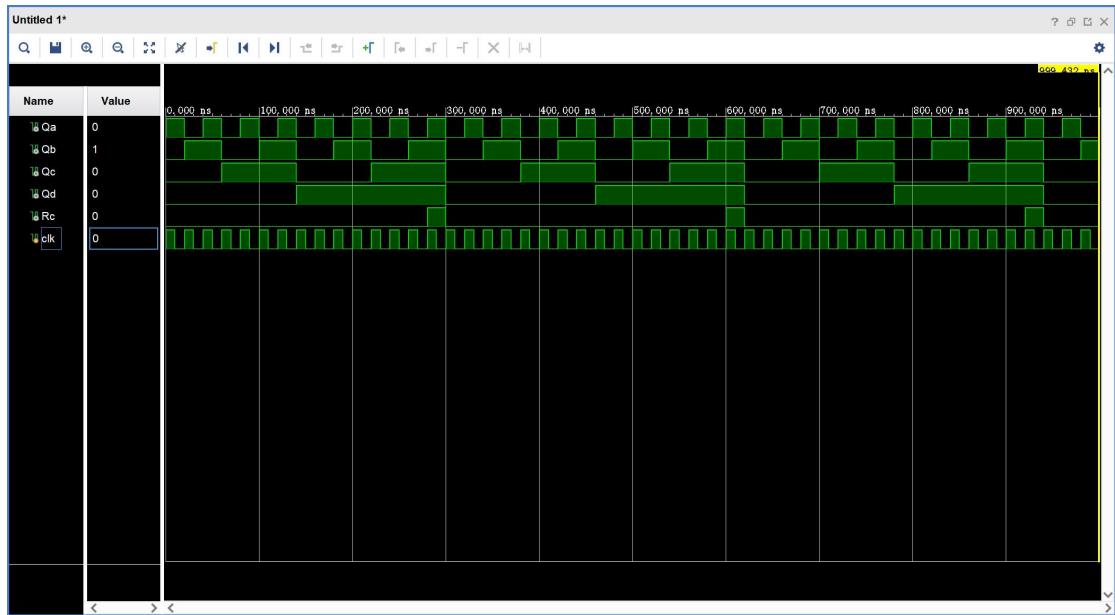
```

④综合运行，烧录 bit 流文件，导入实验板进行验证，看结果是否符合预期。

二、实验结果与分析

1. 实现基本 SR 锁存器，验证功能和存在的时序问题

(1) 4 位同步二进制计数器的仿真波形图



波形图 1

$$D_A = Q_A$$

$$D_B = \overline{Q}_A Q_B + Q_A \overline{Q}_B = \overline{Q}_A \oplus \overline{Q}_B$$

$$\begin{aligned} D_C &= \overline{Q}_A Q_C + \overline{Q}_B Q_C + Q_A Q_B \overline{Q}_C \\ &= (\overline{Q}_A + \overline{Q}_B) \oplus \overline{Q}_C \end{aligned}$$

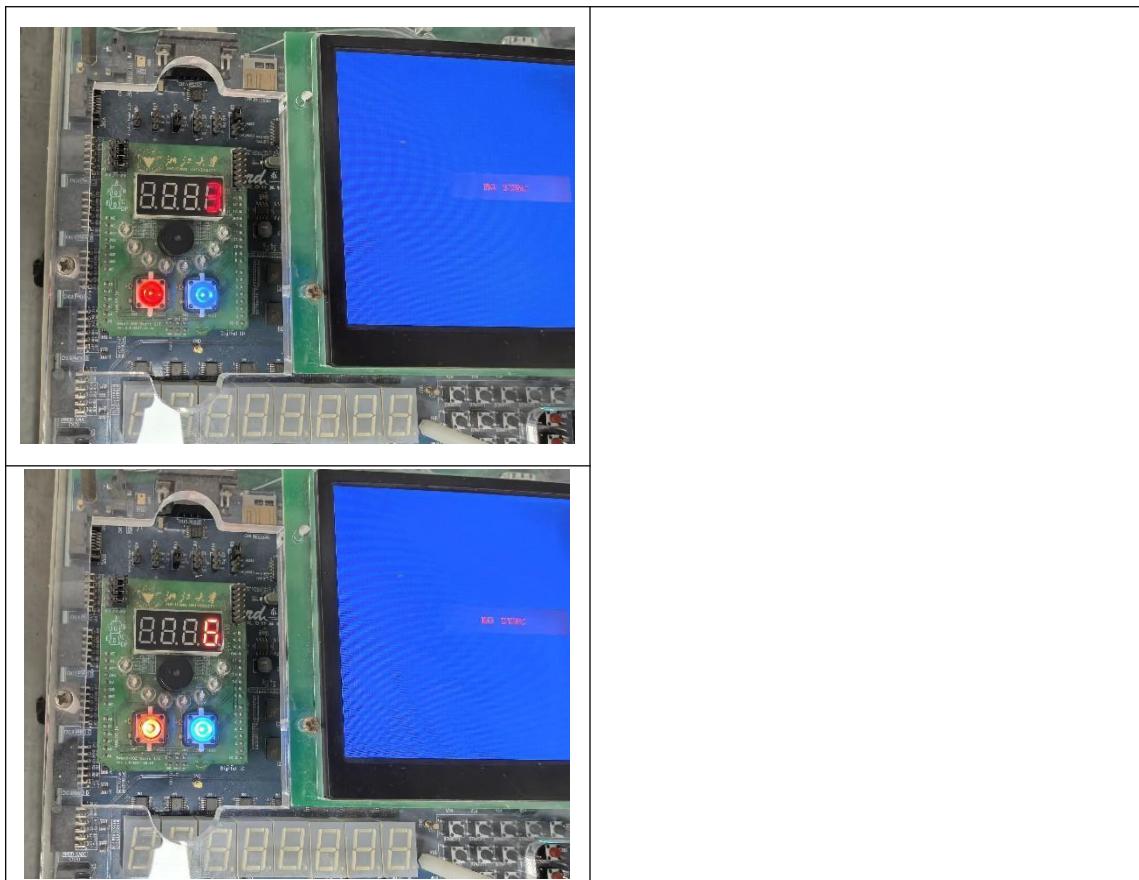
$$\begin{aligned} D_D &= \overline{Q}_A Q_D + \overline{Q}_B Q_D + \overline{Q}_C Q_D + Q_A Q_B Q_C \overline{Q}_D \\ &= (\overline{Q}_A + \overline{Q}_B + \overline{Q}_C) \oplus \overline{Q}_D \end{aligned}$$

$$R_C = \overline{Q}_A + \overline{Q}_B + \overline{Q}_C + \overline{Q}_D$$

将波形图的结果与对应的布尔函数对照，发现结果符合预期。

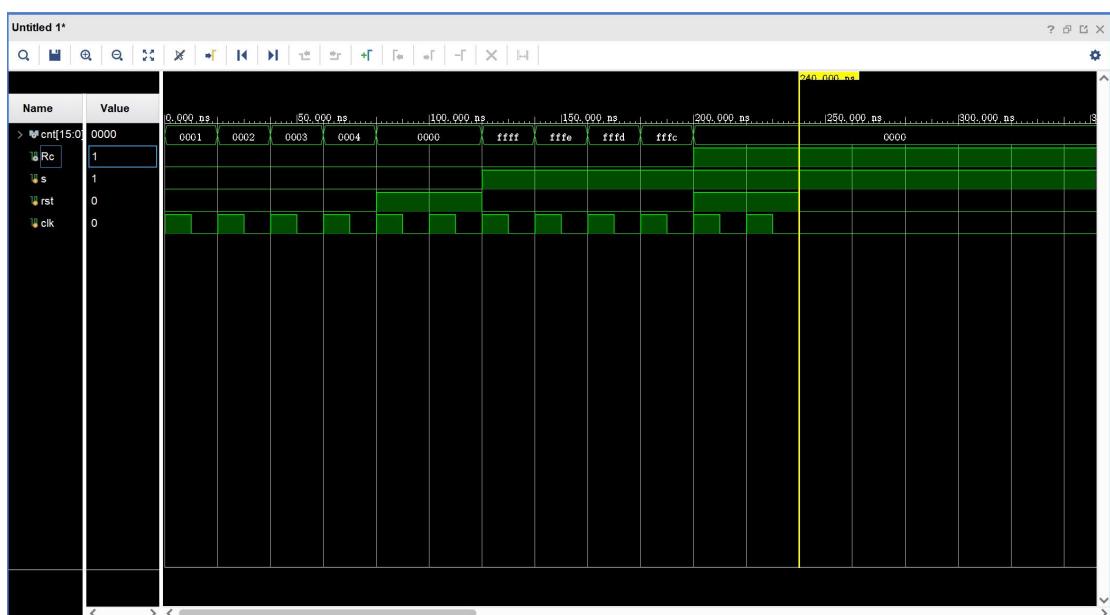
(2) 上板验证

图片	说明
	可以观察到 1 位七段数码管每隔 1s，数字就会增加 1；而且当 F + 1 时又回到了 0，说明功能正常



2. 以 Verilog 行为描述方式设计 16 位可逆二进制同步计数器

(1) 16 位可逆同步二进制计数器的仿真波形图

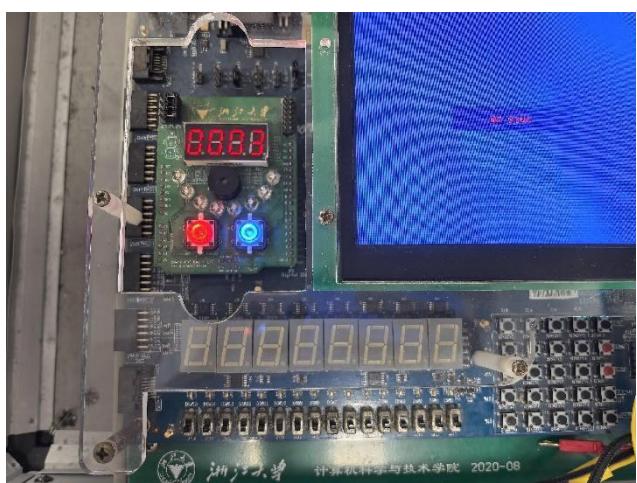


波形图 2

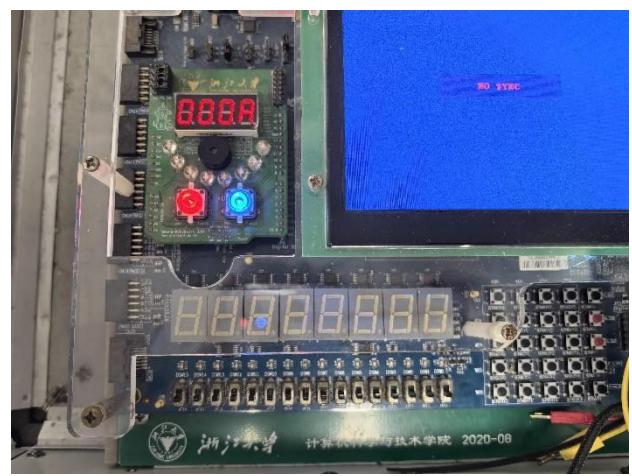
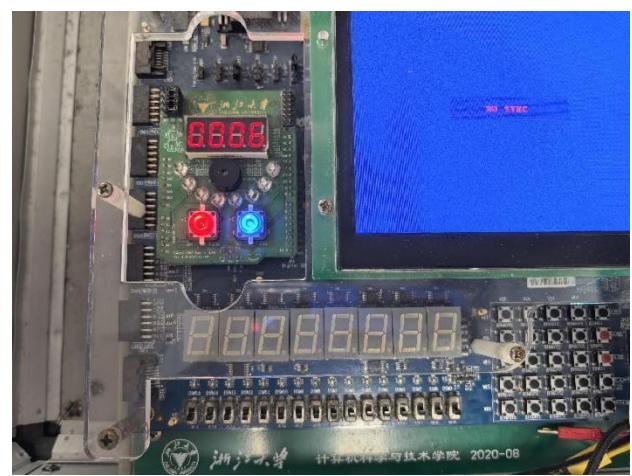
可以看出：S=0 时，数字不断减小；S=1 时，数字不断增大。因此仿真结果符合预期。

(2) 上板验证

图片	说明
	第 1 个开关往上拨，即 $s = 1$ 时，可以看到 4 位七段数码管正显示自减计数
	
	



第1个开关往下拨，即 $s=0$ 时，可以看到4位七段数码管正显示自增计数





拨动第 2 个开关，即同步重置信号 $rst = 1$ 时，重置到 0000

三、讨论、心得

这次实验总的来说还算比较容易，因为相比前面几次实验，除了要理解不同的知识点外，只要按照步骤按部就班地做，还是能够较为顺利地完成。个人认为，最大的挑战可能就是书写 Verilog 代码。可以看出，随着实验的深入，给我们自己写代码的机会越来越多。但是，目前的我还是不太熟悉 Verilog 语法，因此在书写 16 位可逆同步二进制计数器时常常要参考 Verilog 语法文档，虽然代码很短，但花的时间有些长。之后我们的大程主要用代码实现，而不是绘制原理图。因此趁现在我得抓紧时间快速且熟练地掌握 Verilog，打下扎实的基础。