中国科学技术大学计算机学院
《数字电路实验报告》

实验题目：简单组合逻辑电路

学生姓名：叶子昂

学生学号：PB20020586

完成时间：2021年11月10日

实验题目

简单时序电路

实验目的

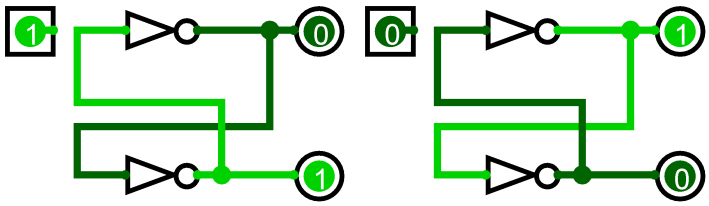
- 掌握时序电路相关器件的原理以及底层结构
- 能够用基本的逻辑门搭建割裂时序逻辑电路
- 能够使用Verilog HDL设计简单逻辑电路
- 进一步掌握logisim的图形化操作以及模拟功能
- 进一步了解掌握Verilog HDL的语法以及设计方法

实验环境

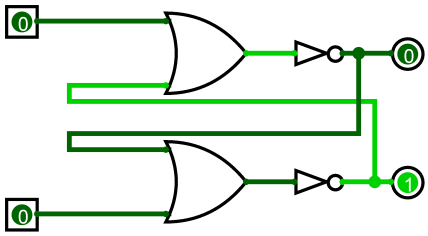
- 装有windows，能连校园网的笔记本电脑
- 本地的Logisim，和Vivado软件
- 配置好环境的VS Code用以编写verilog代码
- vlab.ustc.edu.cn综合平台

实验过程

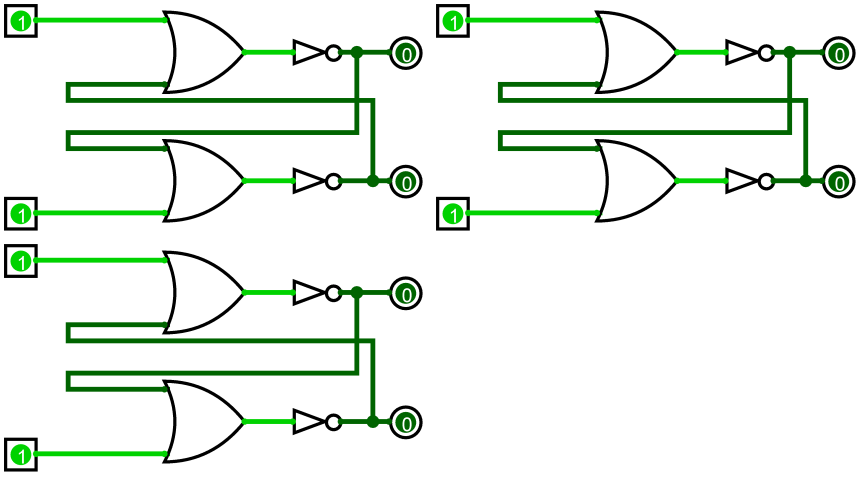
1. 搭建双稳态电路
    - Logisim按照顺序画出电路图



2. 搭建SR锁存器
    - Logisim画出逻辑图

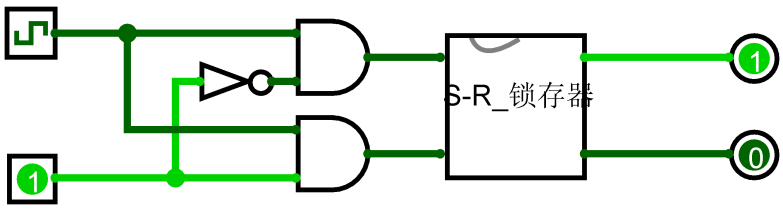


    - 改变输入观察输出

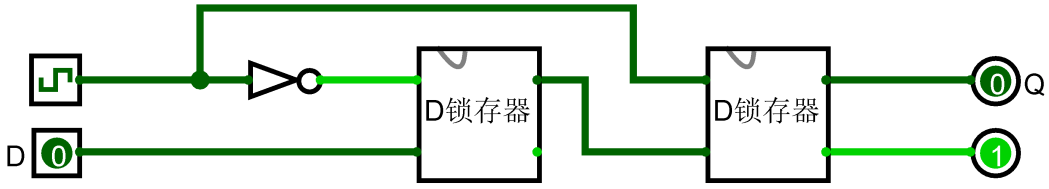


3. 搭建D锁存器

- 使用Logisim画出电路图



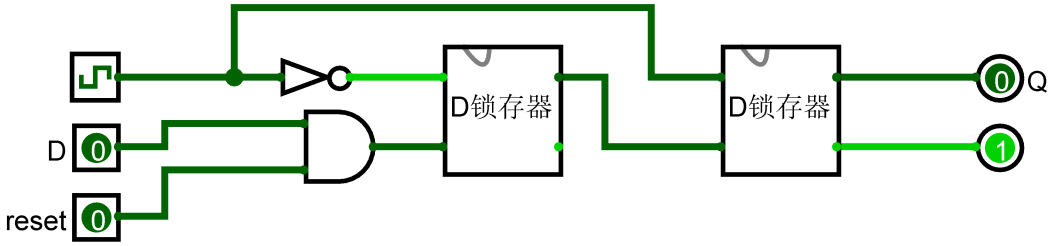
4. 搭建D触发器
   - 使用Logisim画出电路图



   - 根据电路写出Verilog代码

```
module d_ff (
    input clk,d,
    output reg q
);
    always@(posedge clk)
        q<=d;
endmodule
```

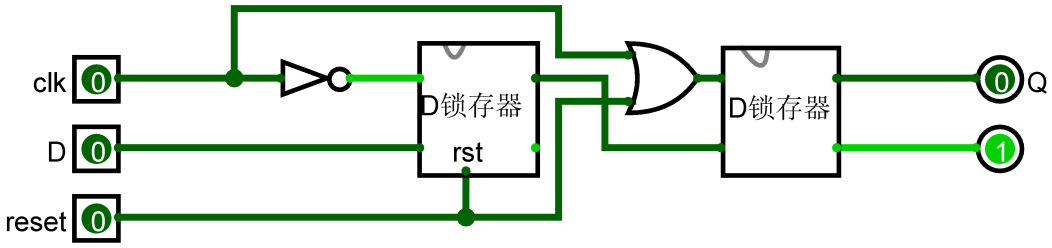
   - 在电路中添加同步复位功能



   - 根据电路写出Verilog代码

```
module d_ff_r (
    input clk,rst_n,d,
    output reg q
);
    always@(posedge clk)
    begin
        if(rst_n==0)
            q<=1'b0;
        else
            q<=d;
    end
endmodule
```

   - 在电路中添加异步复位功能

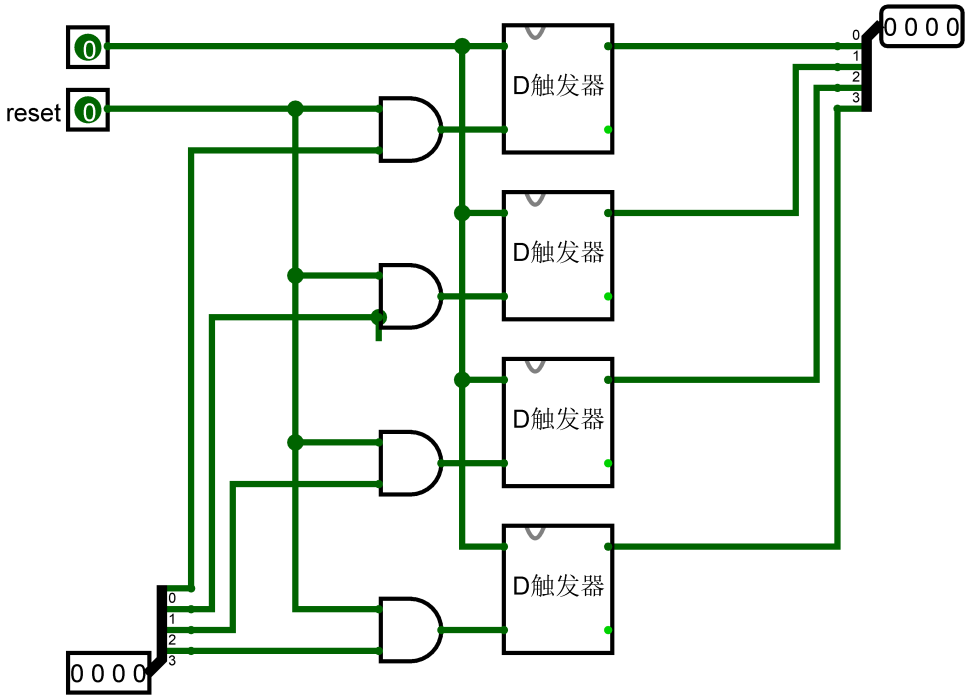


- 根据电路写出Verilog代码

```
module d_ff_s (
    input clk,rst_n,d,
    output reg q
);
    always@(posedge clk or negedge rst_n)
    begin
        if(rst_n==0)
            q<=1'b0;
        else
            q<=d;
    end
endmodule
```

5. 搭建寄存器

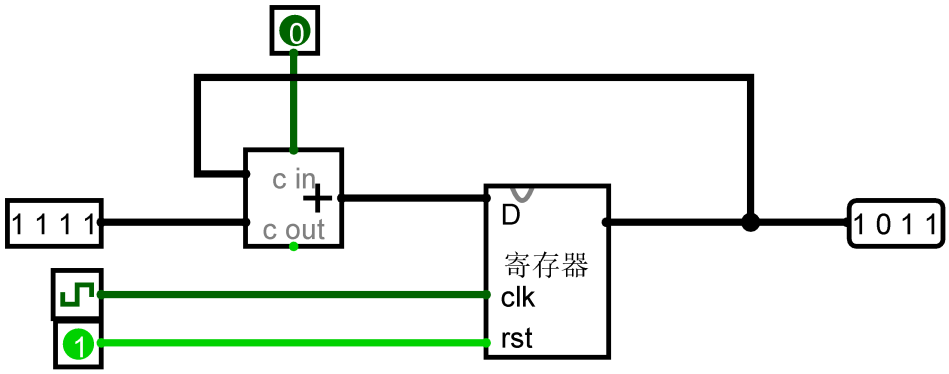
- 使用logisim绘制电路图



- 根据电路图写出Verilog代码

```
module REG4(
    input clk,rst_n,
    input [3:0] D_IN,
    output reg [3:0] D_OUT
);
    always@(posedge clk)
    begin
        if(rst_n==0)
            D_OUT<=4'b0;
        else
            D_OUT<=D_IN;
    end
endmodule
```

6. 搭建简单时序逻辑电路
    - 使用logisim绘制电路



    - 根据电路写出Verilog代码

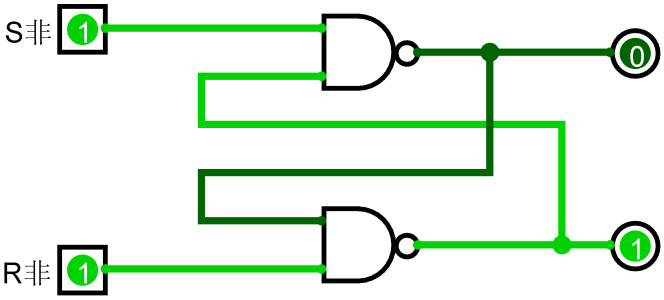
```
module RGE4 (
    input clk,rst_n,
    output reg [3:0] CNT
);
    always@(posedge clk)
    begin
        if(rst_n==0)
            CNT<=4'b0;
        else
            CNT<=CNT+4'b1;
    end
endmodule
```

实验练习

题目一

在 *Logisim* 中用与非门搭建 *SR* 锁存器，画出电路图，并分析其行为特性，列出电路在不同输入时的状态。

- 使用Logisim绘制出电路图

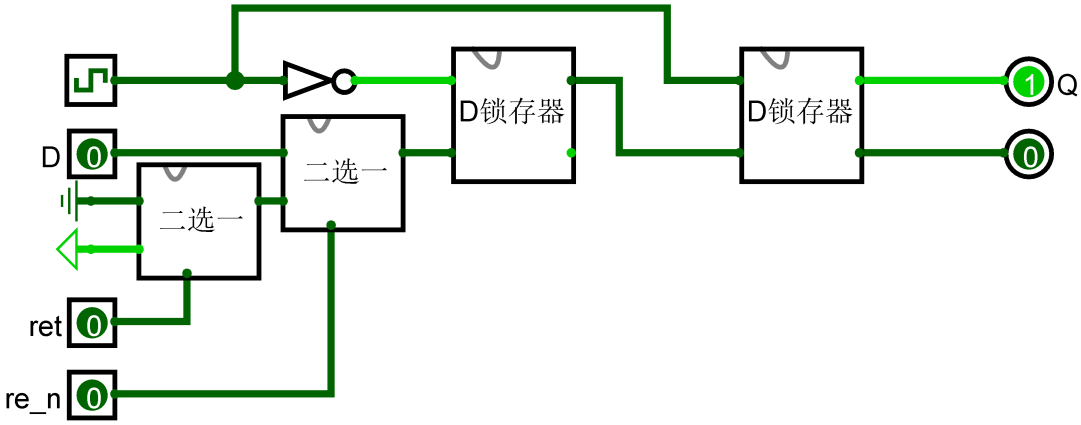


- 若S非为0则会使SR锁存器置为“1”态，R非为0则会使SR锁存器置为"0"态，S非和R非同为1时电路输出保持不变，S非R非同为0电路不稳定是不被允许的。

题目二

在 *Logisim* 中搭建一个支持同步置位功能的 *D* 触发器，画出其电路图，并编写对应的 *Verilog* 代码。

- 基于D触发器加入同步置位信号，在Logisim中绘制电路图



- 基于电路图写出Verilog代码

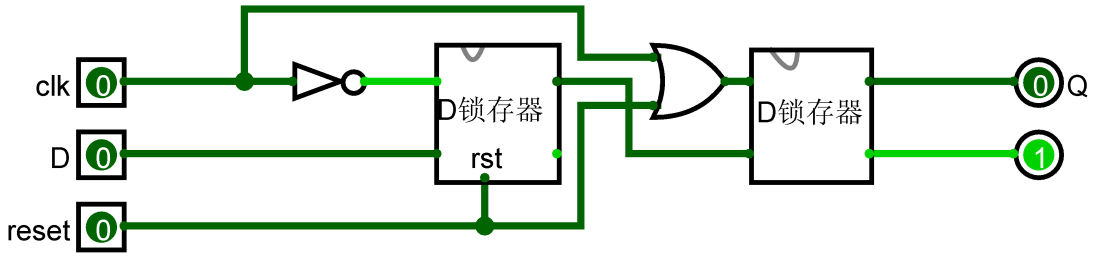
```
module top_module (
    input clk,D,ret,re_n,//ret_n控制是否置位，ret控制置位为0还是1
    output reg q
);
    wire temp1,temp2;
    select select1(.a(0),.b(1),.sel(ret),.cout(temp1));
    select select2(.a(D),.b(temp1),.sel(re_n),.cout(temp2));
    always@(posedge clk)
    begin
        q<=temp2;
    end
endmodule

module selsct (     //二选一模块
    input a,b,sel,
    output cout
);
    wire s,carry1,carry2;
    not(s,sel);
    and(carry1,s,a);
    and(carry2,sel,b);
    or(cout,carry1,carry2);
endmodule
```

题目三

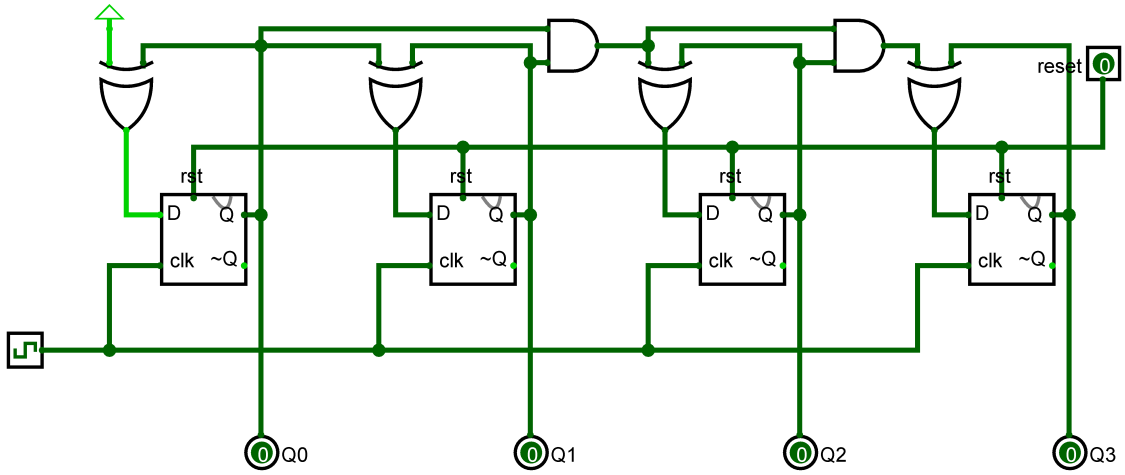
在 *Logisim* 中搭建一个带有异步复位功能的 *D* 触发器，画出其完整电路图，并进一步调用该触发器设计一个从 *0~15* 循环计数的*4bit* 计数器（可使用 *Logisim* 中的加法器模块，也可自行设计计数器），写出计数器的 *Verilog* 代码。

- 搭建异步复位D触发器（复位信号高电平有效）

  左侧D锁存器有一个异步复位输入端口rst。



- 调用该模块实现4bit计数器



- 根据电路写出Verilog代码

```
module top_module (
    input clk,reset,
    output Q0,Q1,Q2,Q3
);
  wire D0,D1,D2,D3;
  xor(D0,1,Q0);
  xor(D1,Q0,Q1);
  xor(D2,Q0&&Q1,Q2);
  xor(D3,Q0&&Q1&&Q2,Q3);
  noDchu FF0(.clk(clk),.D(D0),.rst(reset),.q(Q0));
  noDchu FF1(.clk(clk),.D(D1),.rst(reset),.q(Q1));
  noDchu FF2(.clk(clk),.D(D2),.rst(reset),.q(Q2));
  noDchu FF3(.clk(clk),.D(D3),.rst(reset),.q(Q3));
endmodule

module noDchu (//D触发器
    input clk,D,rst,
    output reg q
);
  always @(posedge clk or posedge rst)
  begin
      if(rst==1)
      q<=0;
      else
```

```
        q<=D;
    end
endmodule
```

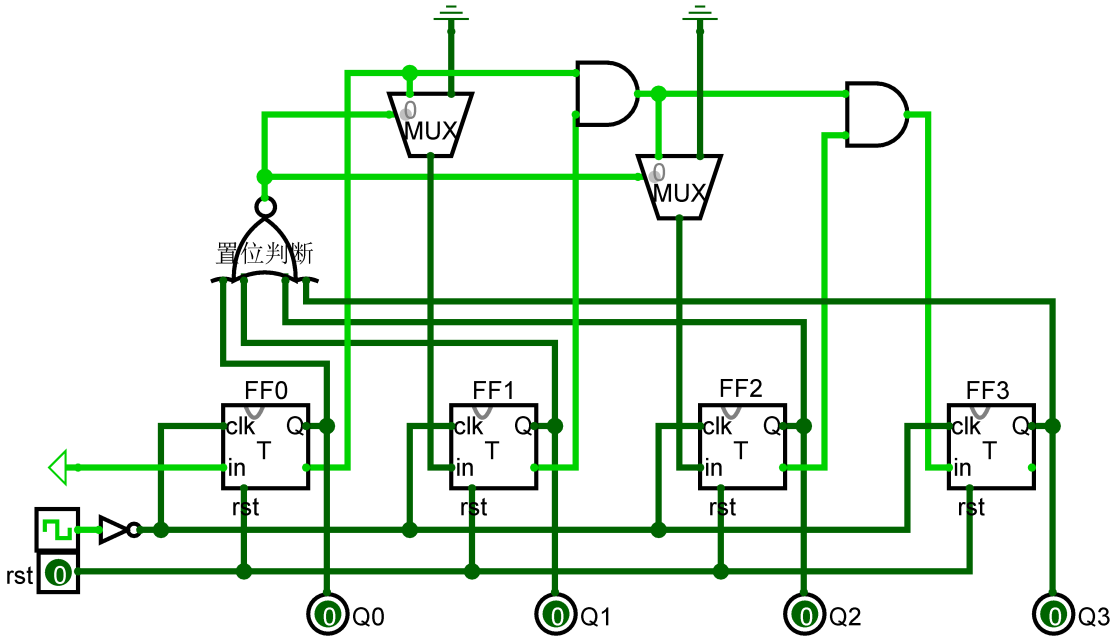
题目四

在 *Logisim* 中搭建一个 *9~0* 循环递减的计数器，复位值为 *9*，每个周期减一（可使用 *Logisim* 中的减法器模块，也可自行设计计数器）， 画出电路图，进行正确性测试，并写出其对应的 *Verilog* 代码。

- 计数器模为10，需用4个触发器，选择T触发器（输入信号为1时输出信号翻转，为0时保持），采用同步复位的方法在输出端均为0的下一个时钟下降沿将电路置为1001，logisim搭建电路图如下。

  计数原理（不考虑置位）：从低位开始翻转所有信号至第一个不为0的位数（该位也翻转）如：10**10**下一位10**01**，1**100**下一位1**011**，000**1**下一位000**0**

  置位原理：在输出为0000时通过置位判断门更改两个二选一选择器的输出，使得输入到FF1，FF2的信号为0，在下一时钟上升沿输出不翻转，而FF0,FF3依然反转为1，从而置位为1001即为9。



- 根据电路图分模块写出Verilog代码

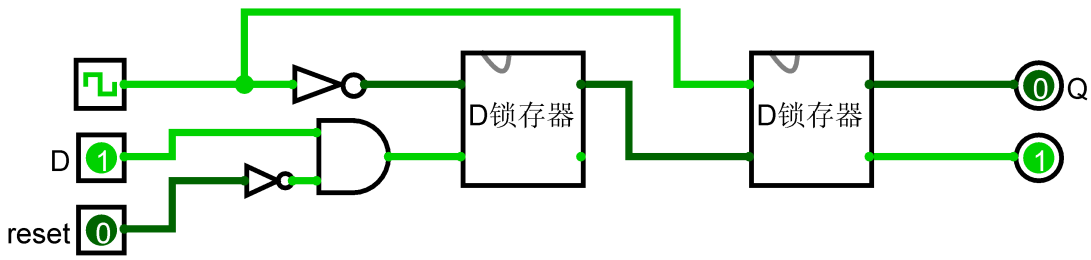
```
module top_module (
    input clk,rst //rst为异步清零信号
    output [4:0] Q
);
  wire [2:1] Qselect;
  wire [2:1] selout;
  wire clk_n,t1,t2,t3,sel;
  assign t1=~Q[0];
  assign clk_n=~clk;
  and(t2,~Q[0],~Q[1]);
  and (t3,~Q[0],~Q[1],~Q[2]);
  or(sel,Q[0],Q[1],Q[2]);
  T_ff FF0(.clk(clk_n),.t(1),.rst(rst),.cout(Q[0]));
  T_ff FF1(.clk(clk_n),.t(selout[1]),.rst(rst),.cout(Q[1]));
  T_ff FF2(.clk(clk_n),.t(selout[2]),.rst(rst),.cout(Q[2]));
  T_ff FF3(.clk(clk_n),.t(t3),.rst(rst),.cout(Q[3]));
  selsct sel1(.cout(selout[1]),.a(t1),.b(0),.sel(sel));
  selsct sel2(.cout(selout[2]),.a(t2),.b(0),.sel(sel));
```

```
endmodule

module selsct (
    input a,b,sel,
    output cout
);
    wire s,carry1,carry2;
    not(s,sel);
    and(carry1,s,a);
    and(carry2,sel,b);
    or(cout,carry1,carry2);
endmodule

module T_ff (
    input clk,t,rst
    output reg cout
);
    always @(negedge clk or posedge rst)
  begin
      if(rst==1'b0)
          cout==1'b0;
      else if(t==1'b1)
           cout<=~cout;
  end
endmodule
```

题目五

手册中给出的示例电路的复位信号都是低电平有效，如要使复位信号高电平有效，应如何实现？试用 *Logisim* 画出一个示例电路，并编写 *Verilog* 代码。

- 选择同步复位D触发器作为示例，使复位信号高电平有效。



- 根据电路编写Verilog代码

```
module D_rsth (
    input clk,D,rst,
    output reg q
);
    always @(posedge clk)
    begin
    if(rst==1)
        q=1'b0;
    else
        q=D;
    end
endmodule
```

总结与思考

1. 实验收获
    - 通过本次实验，我进一步熟练了Logisim的各项工具，能够利用Logisim设计绘制简单的时序逻辑电路。
    - 了解掌握新的Verilog语法和新关键字，能够理解阅读并设计简单的时序逻辑电路。
    - 在写Verilog代码的同时了解代码书写的规范性，提高代码整洁性与可读性，有利于后续纠错。
    - 通过实践加深了对时序逻辑基本元器件的原理和底层架构的理解。
2. 难易程度

    本次实验涉及较多时序逻辑相关内容较前两次实验稍难

3. 实验任务量

    本次实验既包含较为复杂电路图绘制以及Verilog代码实现，总体任务量偏多。

4. 对本次实验的建议

    希望在教授Verilog进一步知识的的同时能够介绍一些代码错误的排查方法以及verilog代码语句与实际电路的对应关系。