

# 实验03 简单时序逻辑电路

## 【实验题目】

使用 Logisim 设计简单时序电路，并学习用 Verilog 语言描述简单时序逻辑电路。

## 【实验目的】

- 掌握时序逻辑相关器件的原理及底层结构
- 能够用基本逻辑门搭建各类时序逻辑器件
- 能够使用 Verilog HDL 设计简单逻辑电路

## 【实验环境】

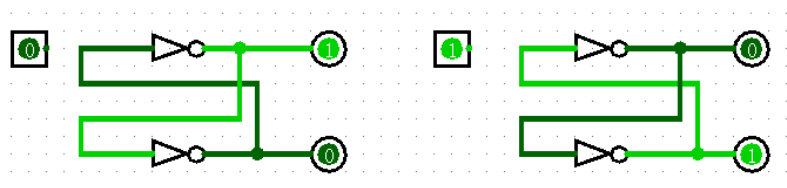
- <[www.vlab.ustc.edu.cn](http://www.vlab.ustc.edu.cn)>
- Logisim
- <[www.verilogoj.ustc.edu.cn](http://www.verilogoj.ustc.edu.cn)>

## 【实验过程】

### Step 1: 搭建双稳态电路

双稳态电路是一种最简单的时序逻辑电路，没有输入信号，状态一旦确定之后也无法改变，没有实际使用价值，但却是所有时序逻辑电路的基础。

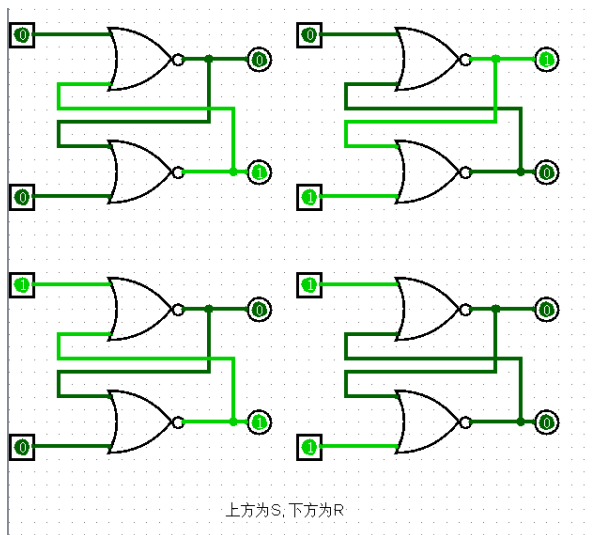
在 Logisim 中搭建双稳电路，注意应先将两条交叉耦合线断开一条，等输入信号将其状态初始到确定状态后再将耦合线连上。否则电路将处于一种不确定状态。



### Step 2: 搭建 SR 锁存器

双稳态电路没有输入信号，所以无法进行操作，对其进行修改，将两个非门用或非门代替。两个输入信号分别命名为 S 和 R，输出信号命名为 Q 和 /Q，其中 /Q 是 Q 取反的意思，S 信号负责对 Q 置位 (Set)，R 信号负责对 Q 信号置位 (Reset)。当 SR 信号都无效 (为0) 时，电路将保持之前的状态，即处于锁存状态，因此这种电路称为 SR 锁存器。SR 信号都有效 (为 1) 时，Q 和 /Q 信号都为零，虽然也是一种确定状态，但不符合 /Q 为 Q 取反的定义，将其看成是一种未定义状态，在实际使用过程中应避免这种状态的出现。

搭建 S-R 锁存器，并尝试改变输入端数据输入。

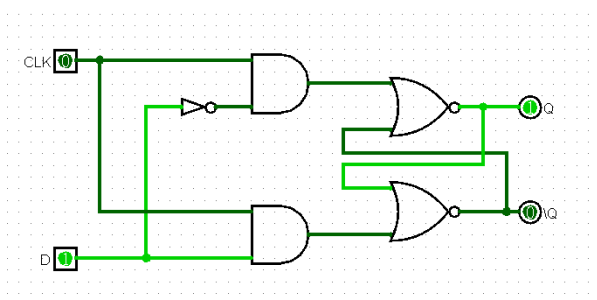


### Step 3: 搭建 D 锁存器

不希望上述 SR 锁存器输入都是1的状态出现，为此在 SR 锁存器前面添加两个与门和一个非门，便构成了 D 锁存器。

分析 D 锁存器电路可以发现，当 CLK 信号为高电平时，Q 信号将随着 D 端输入信号的变化而变化，称之为“跟随”状态。当 CLK 信号为低电平时，Q 信号将保持之前的值，不会收到 D 信号变化的影响，称之为“锁存”状态。D 锁存器是一种电平敏感的时序逻辑器件。

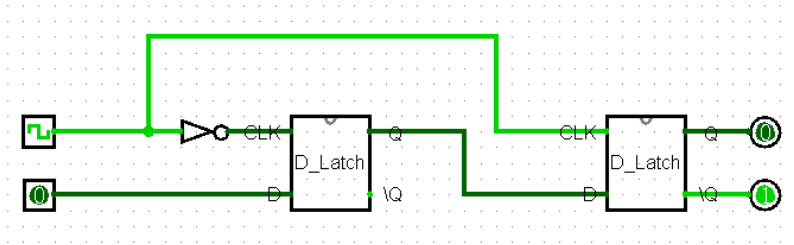
搭建 D 锁存器。



分析 D 锁存器电路可以发现，当 CLK 信号为高电平时，Q 信号将随着 D 端输入信号的变化而变化，称之为“跟随”状态。当 CLK 信号为低电平时，Q 信号将保持之前的值，不会收到 D 信号变化的影响，称之为“锁存”状态。D 锁存器是一种电平敏感的时序逻辑器件。

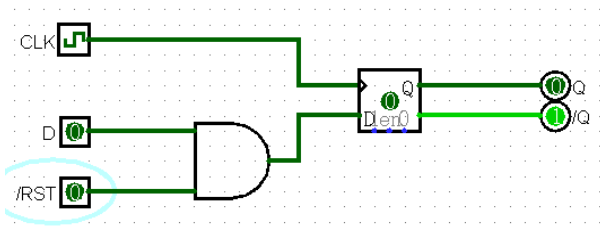
### Step 4: 搭建 D 触发器

D 锁存器在信号的传输过程中起到了类似于开关的作用，当开关（CLK 信号）打开的时候，信号能够传输过去，当开关（CLK 信号）关闭时信号无法通过。如果将两个 D 锁存器串起来，其控制信号有效值始终相反，就构成了 D 触发器，CLK 信号为低电平时，D 信号通过了 D1，当 CLK 信号由低电平变为高电平时，D1 关闭，D2 打开，信号到达 Q 端。



把 CLK 端口换成可自动变化的时钟信号后，只有在 CLK 信号由低电平变为高电平的瞬间，D 信号才会传播到 Q 端，其余时刻 Q 端的值都保持不变。

为触发器添加复位信号，可以看出，当复位信号有效（低电平有效）时，输出信号 Q 始终为零。



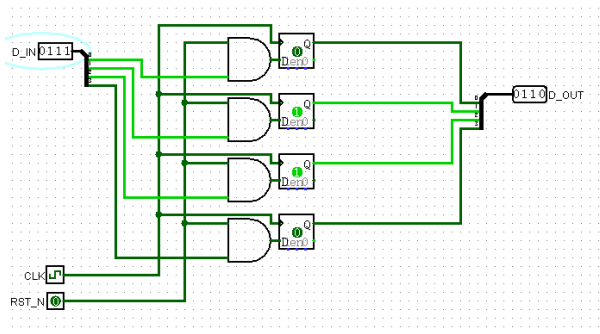
这种触发器的复位信号只有在时钟信号的上升沿才起作用，在非上升沿时刻，复位信号不起作用。这种复位方式称为**同步复位**。

与此同步复位相对应的，还有一种**异步复位**方式，即不论时钟和 D 信号如何，一旦复位信号有效，输出端 Q 立即变为确定的复位值（一般为低电平）。

异步复位与同步复位最大的区别在于，复位信号与时钟信号同时出现在了 `always` 语句的敏感变量列表中，在没有时钟上升沿的情况下，复位信号也能够起作用。因为复位操作不再完全与时钟信号的上升沿同步，因此称为异步复位。

## Step 5: 搭建寄存器

寄存器本质上来讲就是 D 触发器，用 4 个 D 触发器构成了一个能够存储 4bit 数据的寄存器，带有低电平有效的同步复位信号。



在 Logisim 中使用仿真功能对其仿真，总结其行为特征：

1. 只有在 CLK 从低电平变到高电平的瞬间（上升沿），输入信号才会传到输出端。
2. 当复位信号（低电平）有效时，输出信号在时钟信号上升沿时清0。
3. 其他时候，输出信号保持原态不变，不受输入信号影响。

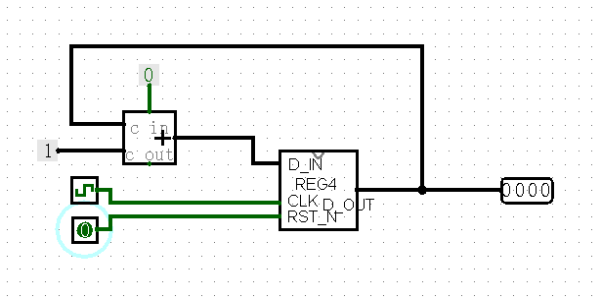
如果有需求使其复位值不为 0000 , 比如为 0011 , Verilog 代码应该改为:

```
module reg4(  
    inout CLK, RST_N,  
    input [3:0] D_IN,  
    output reg [3:0] q);  
    always @ (posedge CLK)  
    begin  
        if (RST_N == 0)  
            D_OUT <= 4'b0011;  
        else  
            D_OUT <= D_IN;  
    end  
end  
endmodule
```

电路图应改为:

### Step 6: 搭建简单时序逻辑电路

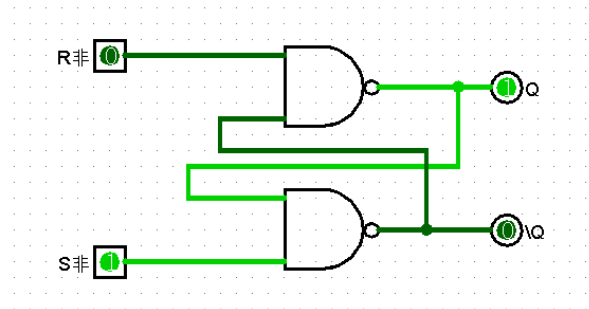
按照要求利用 4bit 寄存器搭建一个 4bit 计数器, 在 0~15 之间循环计数, 复位输出0.



### 【实验练习】

题目 1. 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。

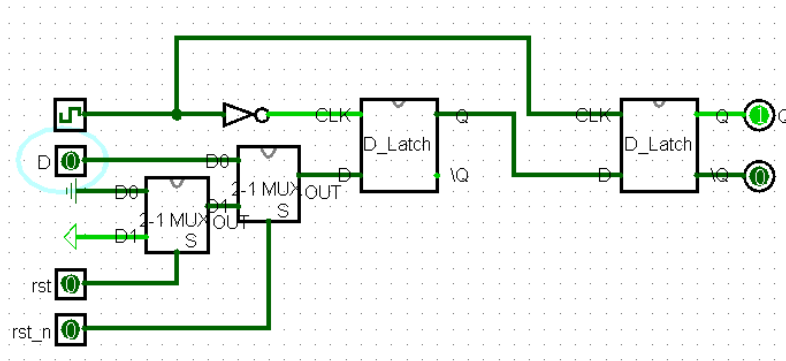
搭建 SR 锁存器。



行为特征：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0电路不稳定是不被允许的。

题目 2. 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。

同步置位功能的 D 触发器：



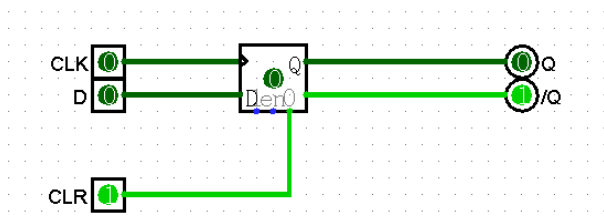
写出其 Verilog 代码：

```
module t2(  
    input clk, D, rst, rst_n, // rst_n 控制是否置位, rst 控制置位为0还是1  
    output reg q  
);  
    wire temp1, temp2;  
    select select1 (.d0(0), .d1(1), .s(rst), .out(temp1));  
    select select2 (.d0(D), .d1(temp1), .s(rst_n), .out(temp2));  
    always @ (posedge clk)  
        begin  
            q <= temp2;  
        end  
endmodule  
  
module select ( //二选一选择器模块  
    input d0, d1, s,  
    output out  
);  
    wire carry0, carry1, carry2;  
    not (carry0, s);  
    and (carry1, d0, carry0);  
    and (carry2, s, d1);  
    or (out, carry1, carry2);  
endmodule
```

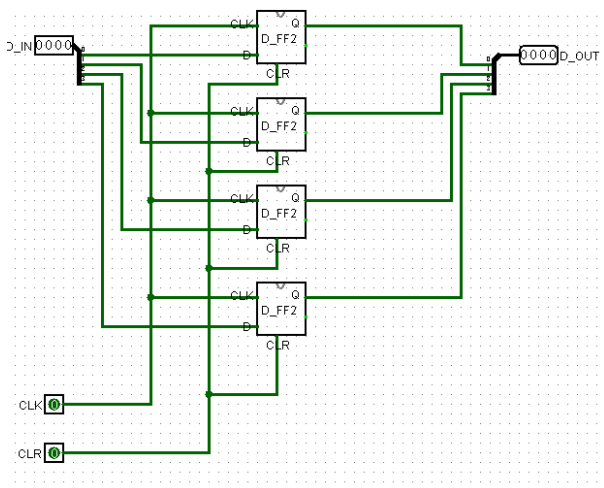
题目 3. 在 Logisim 中搭建一个带有异步复位功能的 D 触发器，画出其完整电路图，并进一步调用该触发器设计一个从 0~15 循环计数的 4bit 计数器（可使用 Logisim 中的加法器模块，也可自行设计计数器），写出计数器的 Verilog 代码。

^755340

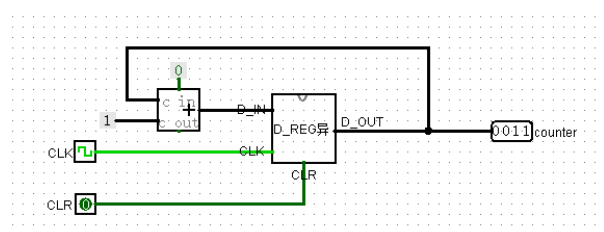
首先搭建异步复位的 D 触发器，复位信号高电平有效，一旦有效，即使输出变为低电平，不受输入和时钟信号的影响。



然后用此触发器搭建能够存储 4bit 数据，高电平有效的异步复位 D 寄存器。



最后再用这个寄存器结合已有的加法器搭建出所求计数器。



计数器的 Verilog 代码为：

```
module REG4( //用异步复位的触发器搭建的寄存器搭建的计数器
    input CLK, CLR,
    output reg [3:0] counter
);
always @ (posedge CLK or posedge CLR)
//因为异步复位寄存器的CLR是高电平有效，所以用or posedge CLR
begin
    if (CLR == 1)
```

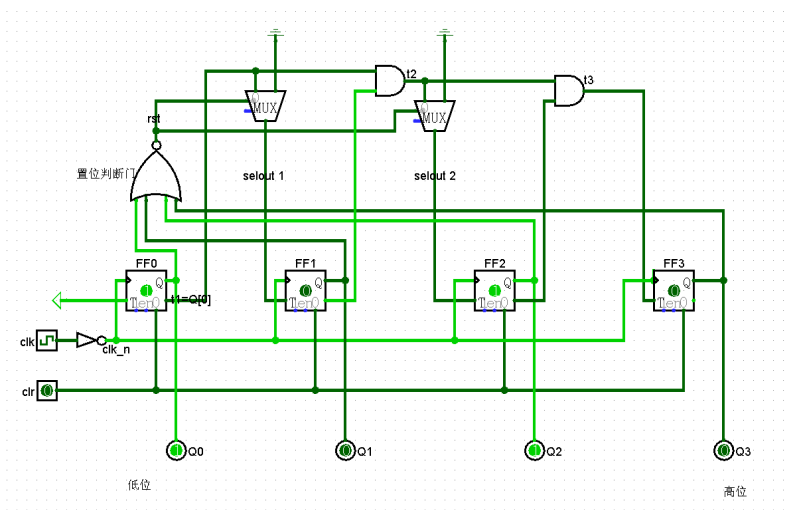
```

        counter <= 4'b0;
    else
        counter <= counter + 4'b1;
end
endmodule

```

题目 4. 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），用高电平有效的异步复位进行清零，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直至第一个不为0的位数（该位也翻转）如：1010下一位1001，1100下一位1011，0001下一位0000。
- 置位原理：在输出为0000时（周期结束或者是按了复位清零）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。

Verilog 代码:

```
module top_module (
    input clk,clr,
    output reg [4:0] Q
);
    wire [2:1] Qselect;
    wire [2:1] selout;
    wire clk_n,t1,t2,t3,rst;
```

```

    assign t1=~Q[0];
    and(t2,~Q[0],~Q[1]);
    assign clk_n=~clk;
    and (t3,~Q[0],~Q[1],~Q[2]);
    T_ff FF0(.clk(clk_n),.t(1),.clr(clr),.cout(Q[0]));
    T_ff FF1(.clk(clk_n),.t(selout[1]),.clr(clr),.cout(Q[1]));
    T_ff FF2(.clk(clk_n),.t(selout[2]),.clr(clr),.cout(Q[2]));
    T_ff FF3(.clk(clk_n),.t(t3),.clr(clr),.cout(Q[3]));
    selsct sel1(.cout(selout[1]),.a(t1),.b(0),.sel(rst));
    selsct sel2(.cout(selout[2]),.a(t2),.b(0),.sel(rst));
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

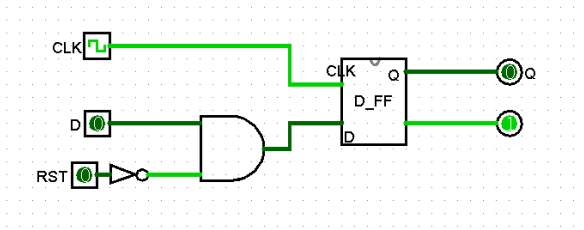
module T_ff (
    input clk,t,rst
    output reg cout
);
    always @(negedge clk or posedge clr)
begin
    if(clr==1'b1)
        out==1'b0;
    else if(t==1'b1)
        cout<=~cout;
end
endmodule

```

题目 5.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。





修改 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

- 进一步熟悉了 Logisim 中的各种工具使用，学会使用仿真工具对时钟信号的输入进行操作。
- 了解了关于时序电路的 Verilog 代码的关键字和语法，能用其设计时序逻辑电路。
- 强化了理论课中学习的时序电路知识，对锁存器、触发器、寄存器的印象得到加深。

### 2. 难易程度

(对于我来说) 难度比较高，尤其是实验题目里面的一些设计。

### 3. 实验任务量

任务量很大，要画很多复杂电路，写很多代码，又做了十几个小时...

### 4. 对本次实验的建议

- 减少一点任务，期中周真的顶不住。
- 多介绍一些代码排错的方法。