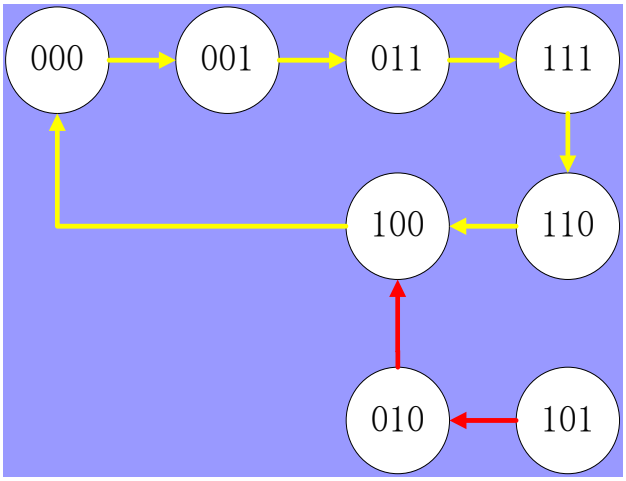
实验三：三位扭环计数器的设计与实现

【实验要求】：

1. 采用 Moore（摩尔型）电路，利用 D 触发器，设计并实现三位扭环计数器。

【实验目的】

1. 掌握时序逻辑电路的设计方法；
2. 熟悉 Vivado2014 集成开发环境和 Verilog 编程语言；
3. 实现如下图所示的三位扭环计数器。
4. 

【实验环境】

- FPGA 虚拟仿真平台。
- Vivado2014 集成开发环境。
- Verilog 编程语言。

【实验原理】

功能描述

计数功能：主要功能是计数，每次触发一次计数器，二进制数字会递增。计数器可以在每次触发时增加一个二进制单位，例如从 000 递增到 001，然后到 010，再到 011，以此类推。

显示功能：有三个二进制数字显示位，用于显示当前的计数数值。每个数字位都能够独立显示 0 或 1。当计数器计数达到 111（二进制）时，通常会回到 000 重新计数，形成循环显示。

复位功能：通常，计数器配备有复位按钮或功能，用于将计数器的值重置为初始状态，即 000。这样可以方便重新开始计数。

真值表

y2	y1	y0	D2	D1	D0
0	0	0	0	0	1
0	0	1	0	1	1
0	1	0	1	0	0
1	0	0	0	0	0
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	1	1	0

逻辑方程

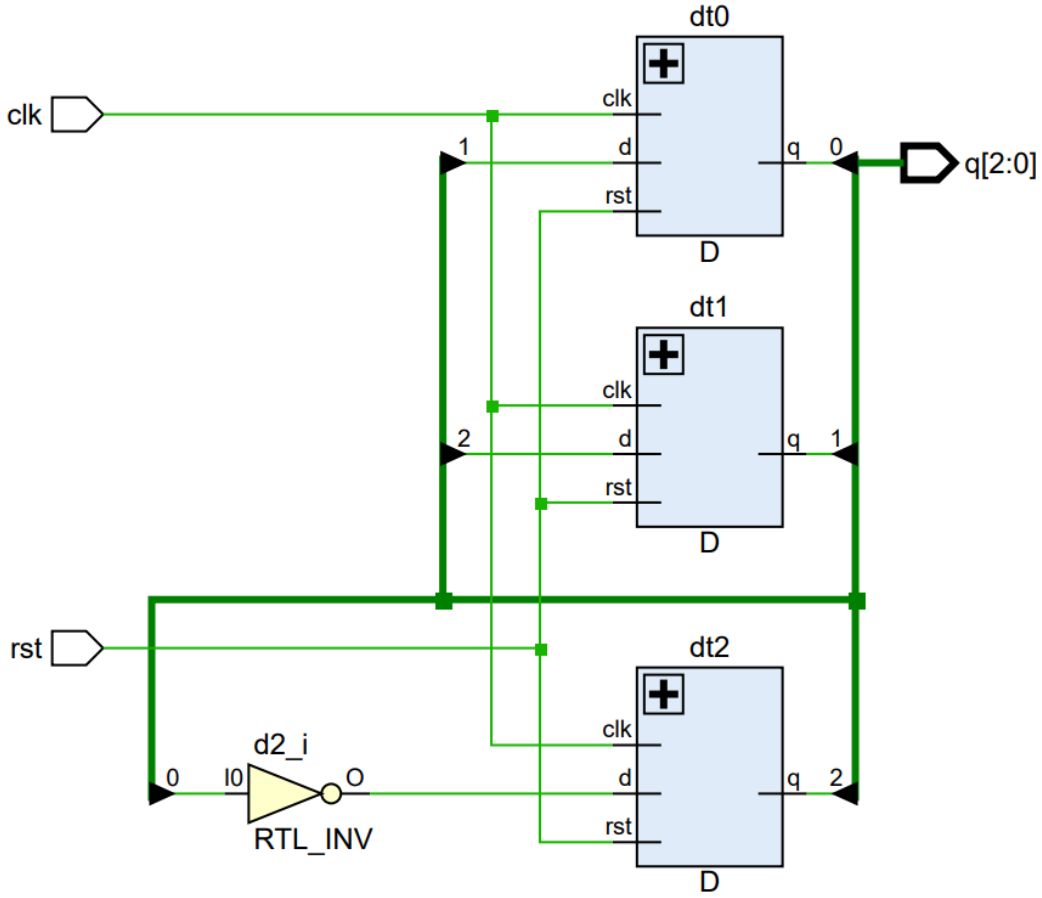
激励方程

$$D_2 = y_1$$
$$D_1 = y_0$$
$$D_0 = \bar{y}_2\bar{y}_1 + \bar{y}_2 y_0$$

次态方程

$$D_2^{n+1} = D_2$$
$$D_1^{n+1} = D_1$$
$$D_0^{n+1} = D_0$$

电路图



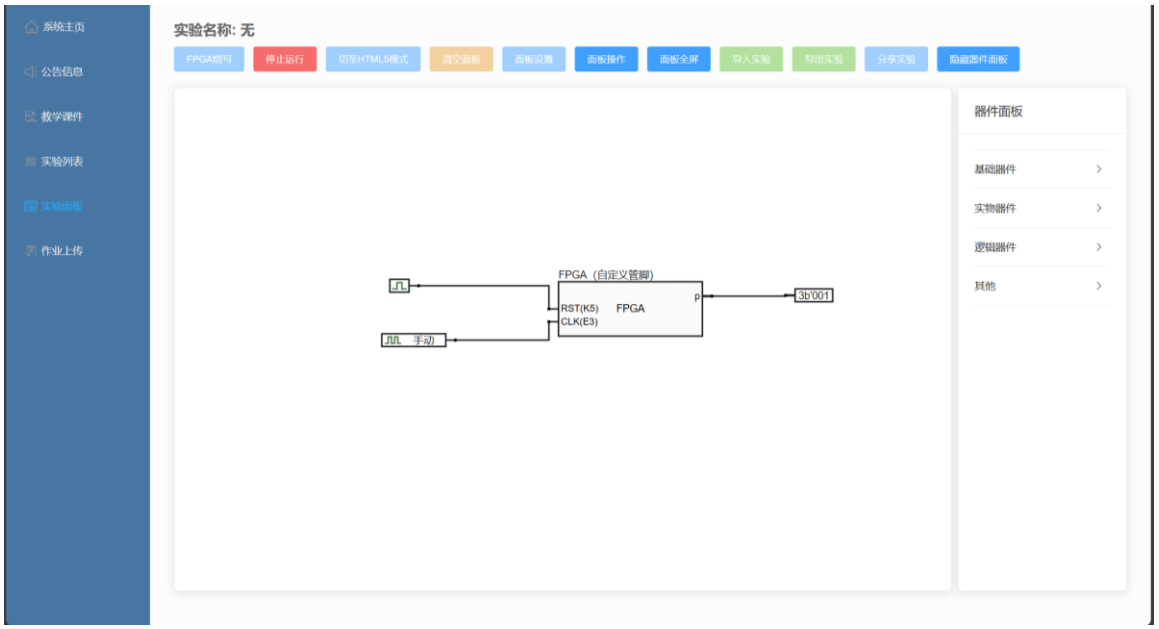
Verilog 代码实现

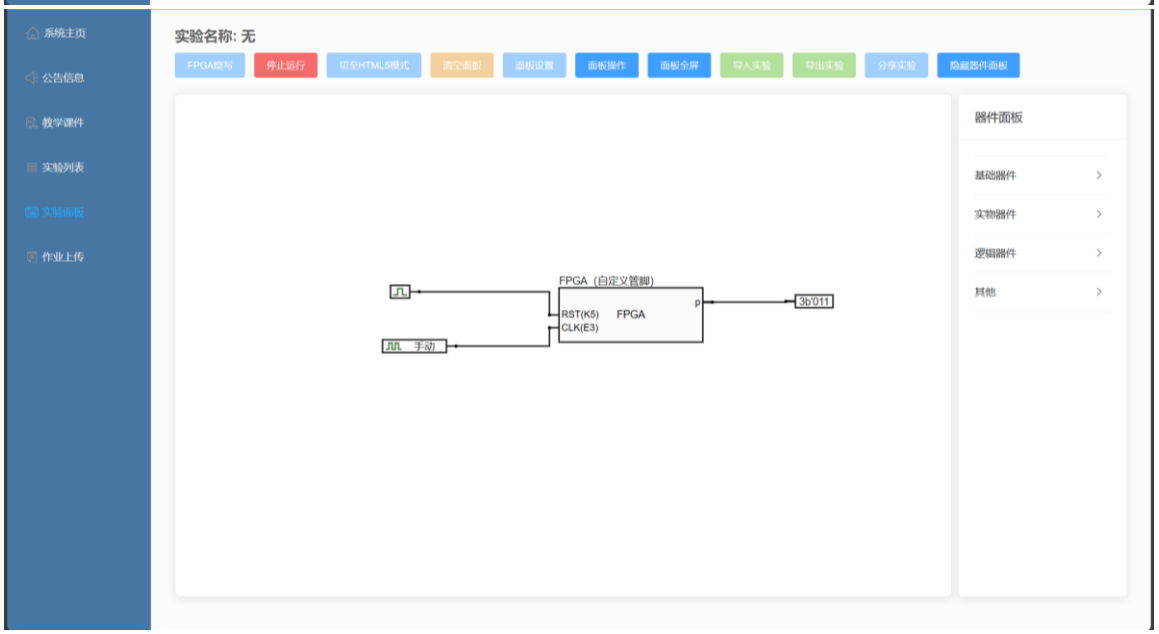
```
module counter(
input clk,
input rst,
output [2:0] q
);
wire d1, d0, d2;

assign d1 = q[2];
assign d0 = q[1];
assign d2 = ~q[0];
D dt1(clk,rst, d1, q[1]);
D dt0(clk,rst, d0,q[0]);
D dt2(clk,rst, d2,q[2]);
endmodule
```

```
module D(
    input clk,
    input rst,
    input d,
    output reg q
);
    always @(posedge clk,negedge rst)begin
    if(rst==1'b1)
    q<=1'b0;
    else
    q<=d;
    end
endmodule
```

实验结果与仿真结果





实验四：四位移位寄存器的设计与实现

【实验要求】：

1. 设计 4 位串行移位寄存器，使其能够实现串行移位功能，即串行输入信号 S1 输入，依次经过四个触发器串行输出，要求能够进行左移和右移。
2. 串行左移举例：串行输入信号 S1=1，现态 $Q_3^nQ_2^nQ_1^nQ_0^n$ 为 0000 时，当时钟上升沿到来时，次态（输出）$Q_3^nQ_2^nQ_1^nQ_0^n$ 为 0010；下一时钟上升沿到来时，次态（输出）$Q_3^nQ_2^nQ_1^nQ_0^n$ 为 0100；下一时钟上升沿到来时，次态（输出）$Q_3^nQ_2^nQ_1^nQ_0^n$ 为 1000。

【实验目的】

1. 掌握时序逻辑电路的设计方法；

2. 熟悉 Vivado2014 集成开发环境和 Verilog 编程语言；
3. 实现四位串行移位寄存器，包括左移和右移。

【实验环境】

- FPGA 虚拟仿真平台。
- Vivado2014 集成开发环境。
- Verilog 编程语言。

【实验原理】

功能描述

寄存器结构： 由四个触发器（Flip-Flop）组成，分别存储四个位（Q0 到 Q3）。

控制信号： 引入一个控制信号，比如说"Shift_Left"和"Shift_Right"。这个信号决定了在时钟上升沿到来时是左移还是右移。

左移操作： 如果控制信号为"Shift_Left"，那么在每个时钟上升沿到来时，数据从 Q0 传递到 Q1，Q1 传递到 Q2，Q2 传递到 Q3，同时 Q3 传递到一个辅助输出，比如说 Q3_out。这样就完成了左移一位的操作。

右移操作： 如果控制信号为"Shift_Right"，那么在每个时钟上升沿到来时，数据从 Q3 传递到 Q2，Q2 传递到 Q1，Q1 传递到 Q0，同时 Q0 传递到一个辅助输出，比如说 Q0_out。这样就完成了右移一位的操作。

真值表

mode	d	Q1	Q2	Q3	Q4	Q^n1	Q^n2	Q^n3	Q^n4
0	1	0	0	0	0	0	0	0	1
0	1	0	0	0	1	0	0	1	1

0	1	0	0	1	1	0	1	1	1
0	1	0	1	1	1	1	1	1	1
1	1	0	0	0	0	1	0	0	0
1	1	1	0	0	0	1	1	0	0
1	1	1	1	0	0	1	1	1	0
1	1	1	1	1	0	1	1	1	1

逻辑方程

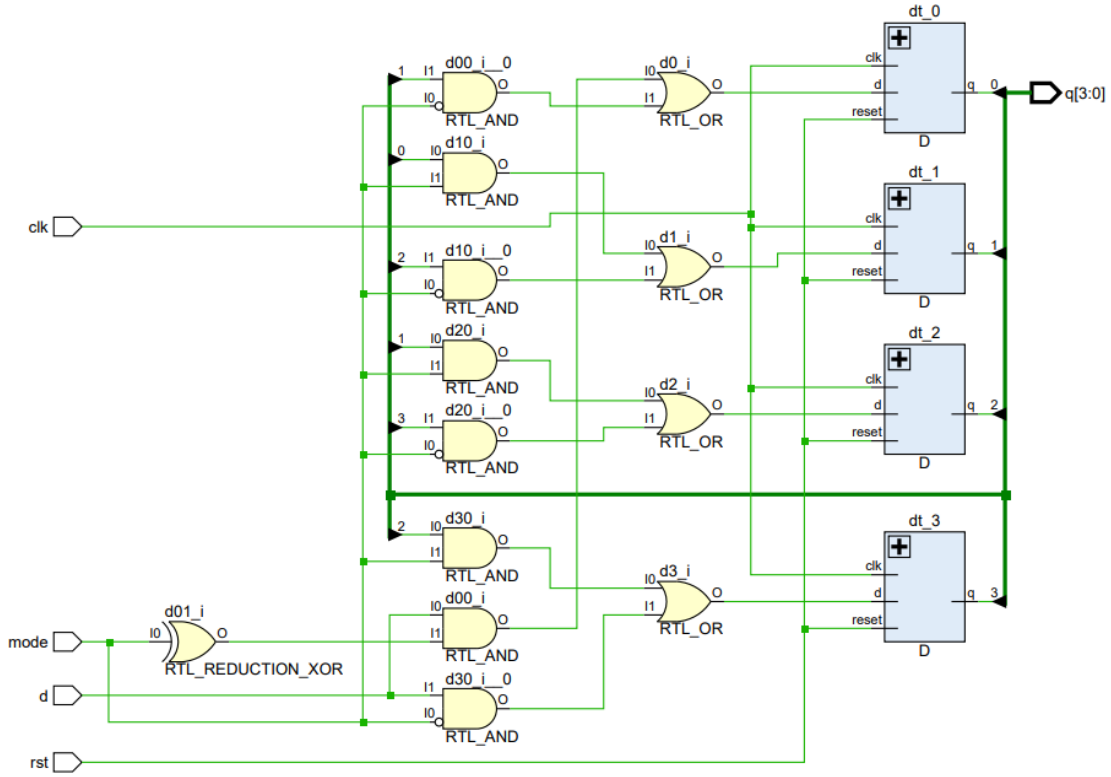
激励方程

$$
\begin{aligned}
D_0 &= D\bar{M} + \bar{M}y_1 \\
D_1 &= y_0 M + \bar{M}y_2 \\
D_2 &= \bar{y_1} M + \bar{M}y_3 \\
D_3 &= \bar{y_2} M + \bar{M}d
\end{aligned}
$$

次态方程

$$
\begin{aligned}
D_3^{n+1} &= D_3 \\
D_2^{n+1} &= D_2 \\
D_1^{n+1} &= D_1 \\
D_0^{n+1} &= D_0
\end{aligned}
$$

电路图

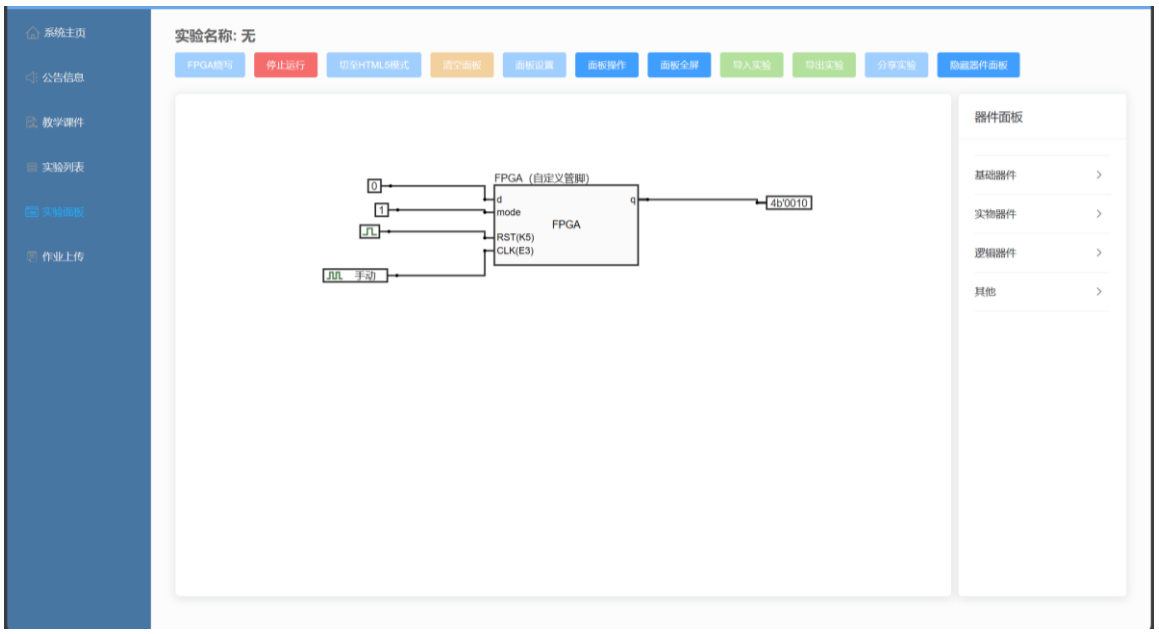


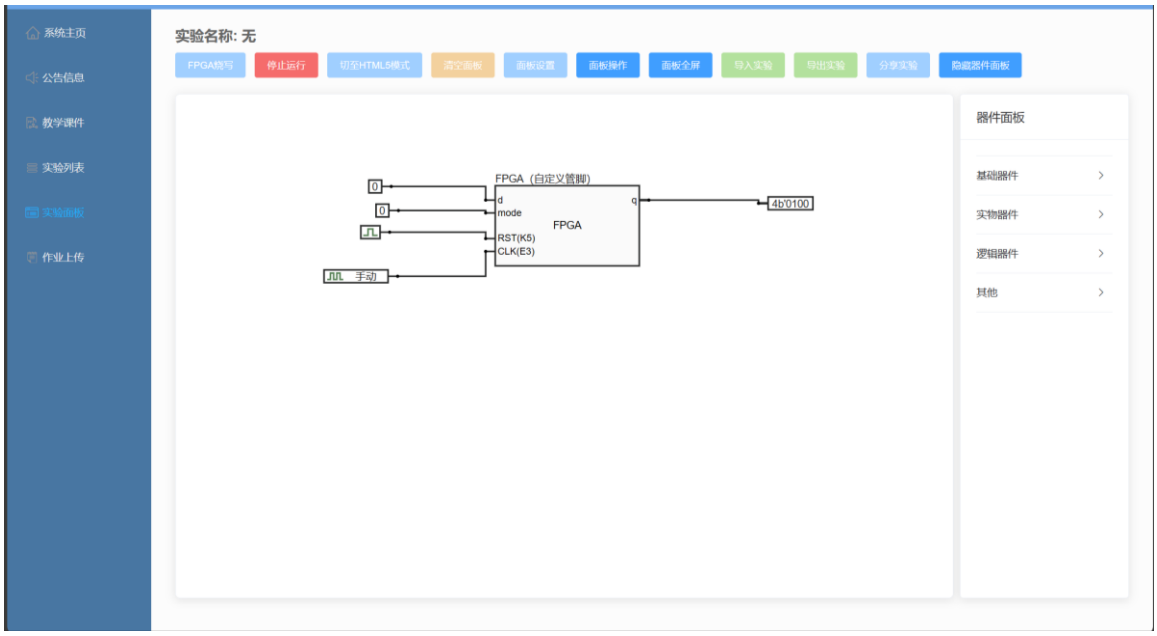
Verilog 代码实现

```
module lr_reg(
    input clk,
    input rst,
    input d,
    input mode,
    output [3:0] q
    );
    wire d0,  d1, d2, d3;
    assign d0 = (d & ^mode) | (~mode & q[1]);
    assign d1 = (q[0] & mode) | (~mode & q[2]);
    assign d2= (q[1] & mode) | (~mode & q[3]);
    assign d3 = (q[2] &  mode)| (~mode & d);
    D dt_0(clk,rst,d0,q[0]);
    D dt_1(clk,rst,d1,q[1]);
    D dt_2(clk,rst,d2,q[2]);
    D dt_3(clk,rst,d3,q[3]);
endmodule
```

```
module D(
    input clk,
    input reset,
    input d,
    output reg q
    );
    always @(posedge clk or posedge reset)
    begin
        if(reset) q<=0;
        else q<=d;
    end
endmodule
```

实验结果与仿真结果




系统主页
公告信息
教学课件
实验列表
实验面板
作业上传
实验名称: 无
FPGA烧写
停止运行
切至HTML5模式
清空面板
面板设置
面板操作
面板全屏
导入实验
导出实验
分享实验
隐藏器件面板
FPGA（自定义管脚）
0
0
d
mode
RST(K5)
CLK(E3)
FPGA
q
4'b0100
手动
器件面板
基础器件
实物器件
逻辑器件
其他