实验名称：实验四 触发器和锁存器

姓名：张涵之

学号：191220154

班级：周一 5-6

邮箱：191220154@smail.nju.edu.cn

实验时间：2020/10/9

4.3.1 分析阻塞和非阻塞 RTL 视图和仿真结果

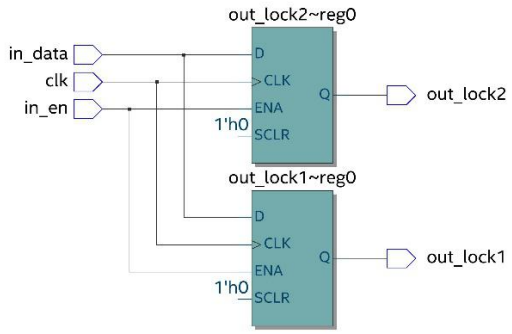
请你建立两个工程，分别研究阻塞赋值和非阻塞赋值的 RTL 级视图和仿真结果：

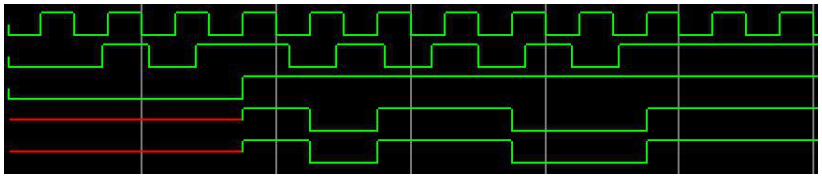
1. 新建工程，用阻塞赋值语句设计两个触发器；保存 Verilog 语言文件。

```
module exp_4_2_1(clk,in_data,in_en,out_lock1,out_lock2);
   input clk;
   input in_data;
   input in_en;
   output reg out_lock1;
   output reg out_lock2;

always @ (posedge clk )
   if(in_en)
      begin
         out_lock1 = in_data;
         out_lock2 = out_lock1;
      end
   else
      begin
         out_lock1 = out_lock1;
         out_lock2 = out_lock2;
      end
endmodule
```

2. 在 Tools 栏，点击 Netlist Viewers 栏下的 RTL Viewer 查看生成的 RTL Schematic，看看在用阻塞赋值语句生成两个触发器的实际电路原理。

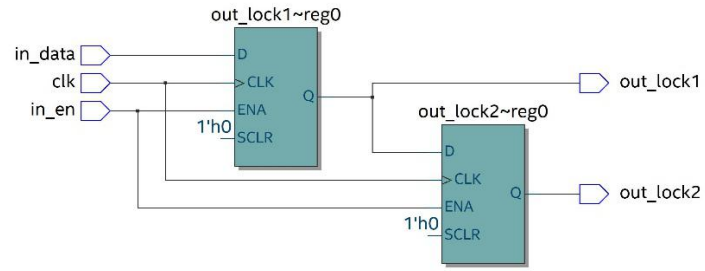


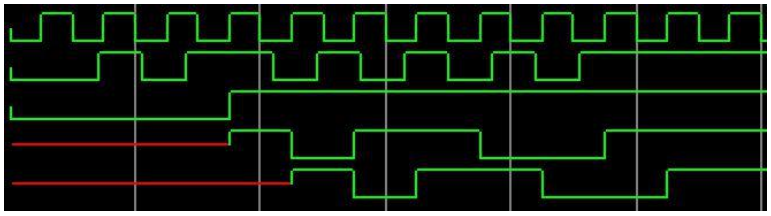


3. 新建另一个工程，用非阻塞赋值语句实现两个触发器，重复上述步骤，比较两种触发器实现方式在硬件电路实现上的异同。

```
module exp_4_2_2(clk,in_data,in_en,out_lock1,out_lock2);
   input clk;
   input in_data;
   input in_en;
   output reg out_lock1;
   output reg out_lock2;

always @ (posedge clk )
   if(in_en)
      begin
         out_lock1 <= in_data;
         out_lock2 <= out_lock1;
      end
   else
      begin
         out_lock1 <= out_lock1;
         out_lock2 <= out_lock2;
      end
endmodule
```





用阻塞赋值语句来设计两个触发器，程序综合出了两个并列的触发器。

用非阻塞赋值语句来设计两个触发器，程序综合出了两个级联的触发器。

4.3.2 设计一个同步清零和一个异步清零的 D 触发器

查阅资料，分析同步清零和异步清零的不同，并请在一个工程中设计两个触发器，一个是带有异步清零端的 D 触发器，而另一个是带有同步清零端的 D 触发器。

实验目的：设计一个带有异步清零端的 D 触发器和一个带有同步清零端的 D 触发器。

实验原理：异步清零只要清零信号有效，就立即进行清零操作，同步清零即使清零信号有效也要等到时钟信号有效沿，才进行清零操作。

在代码中表现为 always @ (posedge clk or posedge clr)和 always @ (posedge clk)，即前者在时钟信号和清零信号的上升沿都执行代码，后者只在时钟信号上升沿执行代码。

以 button KEY0 为时钟，按下一次一个信号上升沿，SW0，SW1，LEDR0 分别为异步清零的清零端、数据端和输出，SW2，SW3，LEDR1 分别为同步清零的清零端和数据端。

实验环境/器材：实验箱一个，笔记本电脑一台。

程序代码或流程图：

```
module myasynchro(clk,clr,d,q);
   input clk;
   input clr;
   input d;
   output reg q;

always @ (posedge clk or posedge clr)
   if (clr==1) q <= 0;
   else q <= d;
endmodule
```

```
module mysynchro(clk,clr,d,q);
   input clk;
   input clr;
   input d;
   output reg q;

always @ (posedge clk)
   if (clr==1) q <= 0;
   else q <= d;
endmodule
```

```
module trigger(clk,clr1,clr2,d1,d2,q1,q2);
   input clk;
   input clr1,clr2;
   input d1,d2;
   output q1;
   output q2;
   myasynchro m1(clk,clr1,d1,q1);
   mysynchro m2(clk,clr2,d2,q2);
endmodule
```

实验步骤/过程：

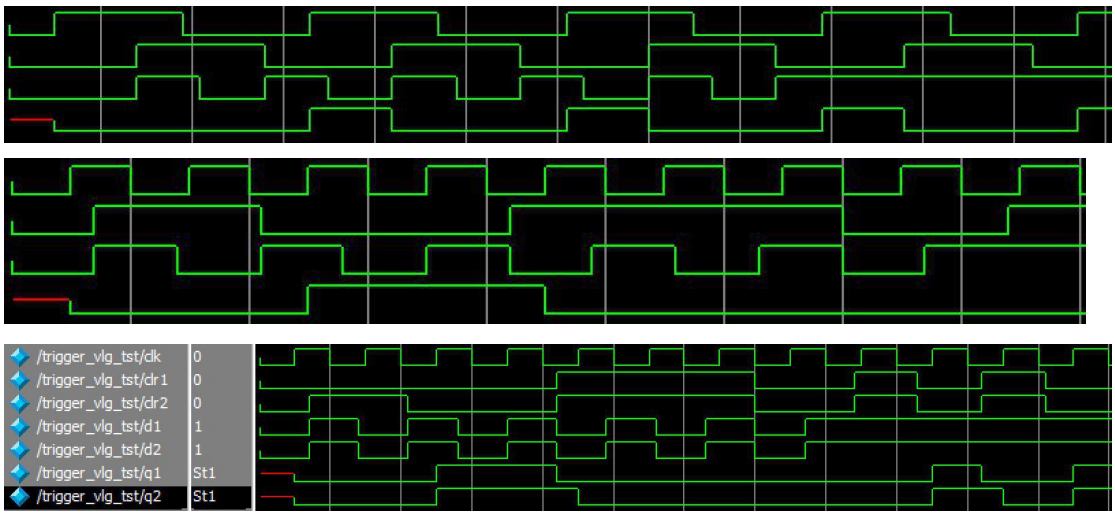
先把两个模块分别设为顶层实体，写单独的测试文件进行仿真模拟。

最后把两个模块合并，写新的测试文件进行仿真，在实验箱上进行操作。

测试方法：

```
initial
begin
// code that executes only once
// insert code here --> begin
   clk = 0; d1 = 0; d2 = 0; clr1 = 0; clr2 = 0; #7;
            d1 = 1; d2 = 1; clr2 = 1; clr2 = 1; #7;
            d1 = 0; d2 = 0;                     #7;
            d1 = 1; d2 = 1; clr1 = 0; clr2 = 0; #7;
            d1 = 0; d2 = 0;                     #7;
            d1 = 1; d2 = 1;                     #7;
            d1 = 0; d2 = 0; clr1 = 1; clr2 = 1; #7;
            d1 = 1; d2 = 1;                     #7;
            d1 = 0; d2 = 0;                     #7;
            d1 = 1; d2 = 1;                     #7;
            d1 = 0; d2 = 0; clr1 = 0; clr2 = 0; #7;
            d1 = 1; d2 = 1;                     #7;
                              clr1 = 1; clr2 = 1; #9;
                              clr1 = 0; clr2 = 0; #9;
                              clr1 = 1; clr2 = 1; #9;
                              clr1 = 0; clr2 = 0; #9;
// --> end
//$display("Running testbench");
end
always
// optional sensitivity list
// @(event1 or event2 or .... eventn)
begin
// code executes for every event on sensitivity list
// insert code here --> begin
   #5 clk = ~clk;
//@eachvec;
// --> end
end
endmodule
```

实验结果：



通过观察对比，异步与同步的清零行为符合预期。经接入实验箱检验，显示也符合预期。

实验中遇到的问题及解决办法：不知道如何写出上升沿恰好错开的测试代码。
解决方法：将信号变化周期设为互质数，多写几个循环，便于观察。

实验得到的启示：无。
意见和建议：无。