一．代码及电路图

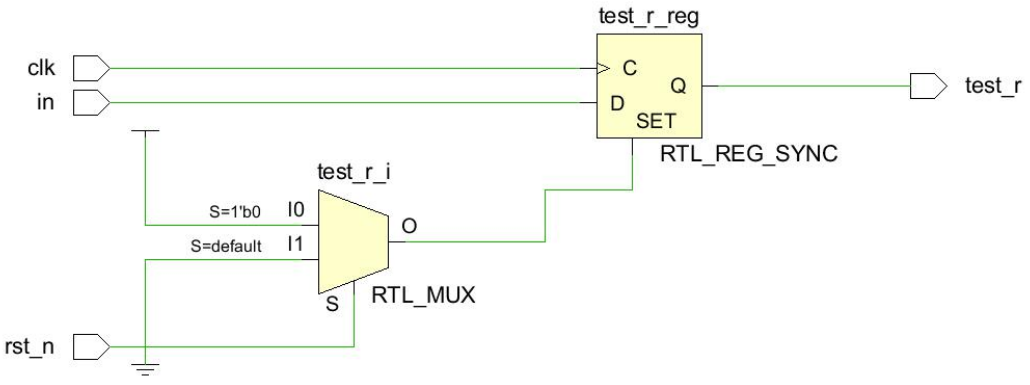
1. 同步清零的上升沿触发的 D 触发器

always 列表里只有 clk 信号而没有复位信号，说明为同步清零

代码：。

```
module hw1(clk,rst_n,test_r,in);
    input clk,rst_n,in;
    output test_r;
    reg test_r;
    always@(posedge clk)begin
        if(!rst_n) begin
            test_r<=1;
        end
        else begin
            test_r<=in;
        end
    end
endmodule
```

电路图：



2.同步清零的下降沿触发的 D 触发器

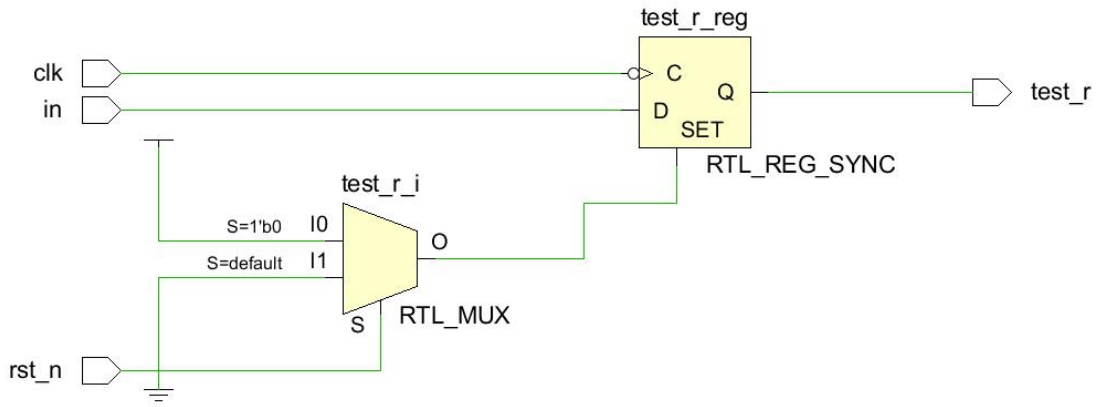
与（1）的区别在于这个为下降沿的 D 触发器，即电平信号由 1 变为 0 时触发器的值才可能改变

代码：

```
module hw2(clk,rst_n,test_r,in);
    input clk,rst_n,in;
    output test_r;
    reg test_r;
    always@(negedge clk)begin
        if(!rst_n) begin
            test_r<=1;
        end
        else begin
            test_r<=in;
        end
    end

endmodule
```

电路图：



3.异步清零的带使能上升沿触发的 D 触发器

异步清零触发器，always 列表除了 clk 信号，还有复位信号，即无论是否到了 clock 的一个周期，只要复位信号有效，触发器就会复位。在 always 块内，先判断复位信号，再判断使能信号，若有效则 sd_init_flag_r 赋值为 1，最后将 sd_init_flag_r 赋值给 test 输出。

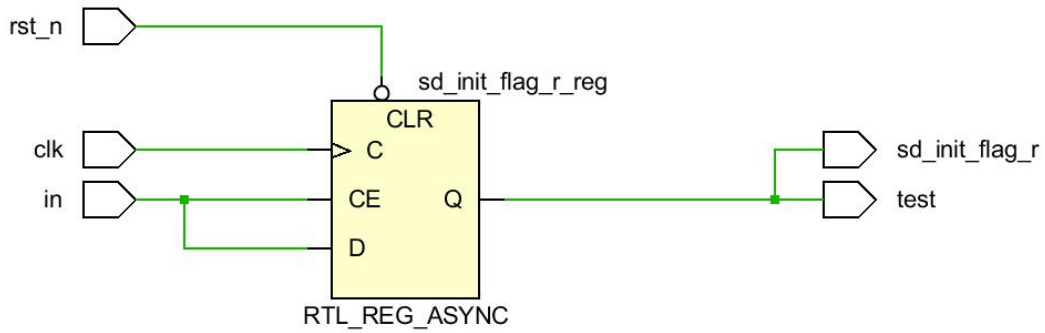
代码：

```
module hw3(clk,rst_n,test,in,sd_init_flag_r);
    input clk,rst_n,in;
    output test,sd_init_flag_r;
    reg sd_init_flag_r;
    wire test;
    always@(posedge clk or negedge rst_n)begin
        if(!rst_n)  begin
            sd_init_flag_r <= 1'b0;        //复位初始化
          end
          else begin
              if(in)
                  sd_init_flag_r <= 1;  //初始化完成
           end
        end

        assign    test = sd_init_flag_r;

endmodule
```

电路图：

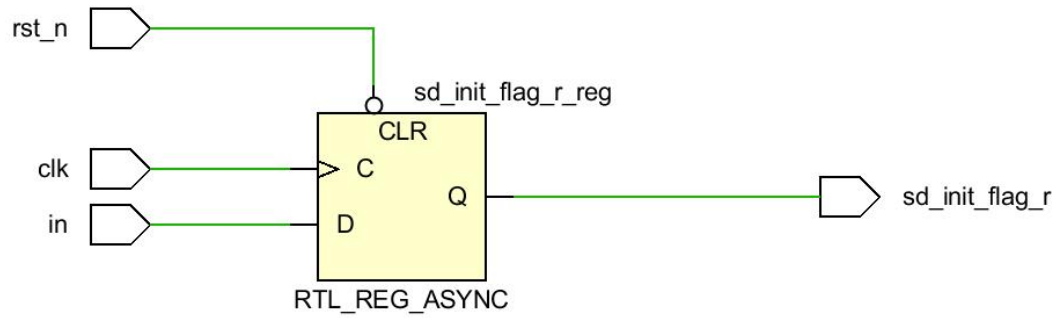


4.异步清零的带使能的上升沿触发的 D 触发器

与（3）的区别在于在使能信号无效时 sd_init_flag_r 被赋值为 0，同时没有将其赋值给 test 输出

```
module hw4(clk,rst_n,in,sd_init_flag_r);
    input clk,rst_n,in;
    output sd_init_flag_r;
    reg sd_init_flag_r;
    always@(posedge clk or negedge rst_n)begin
        if(!rst_n)  begin
            sd_init_flag_r <= 1'b0;         //复位初始化
          end
          else begin
               if(in)
                   sd_init_flag_r <= 1;  //初始化完成
                else
                   sd_init_flag_r <= 0;     //初始化完成
            end
        end

endmodule
```



体会：

触发器时边沿触发的存储器件，只有当时钟信号电平发生变化的那一刻其输出才变化。在 always 语句块内，首先判断是否清零，若清零信号有效则清零；然后判断 clk 信号是否有效，若有效则代表此刻为时钟沿信号改变的瞬间，则触发器可能发生改变。

上升沿触发&下降沿触发：上升沿即由低电平到高电平的瞬间作出响应，下降沿即高电平到低电平的瞬间做出响应。posedge 代表上升沿有效，negedge 代表下降沿有效，不加默认是上升沿有效。

同步&异步：同步时序电路的所有动作是和时钟同步的，而异步时序电路的设计则一般是不和时钟同步。同步复位即复位信号随系统时钟的边沿触发起作用，异步复位即复位信号不随系统时钟的边沿触发起作用，置数同理，rst_n 表示低电平复位。而同步复位和异步复位的区别就在于，前者的复何信号不能出现在 always 语句的敏感信号表中，

无论是同步复位还是异步。同步异步无非就是一个是否受系统时钟边沿触发，如果想要异步就直接加一个敏感信号就好了。