数电实验EXP4

刘永鹏 191220070 计科 693901492@qq.com 2020.9.12

1. 实验目的

1. 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**，步骤如下：
    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。
    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。

2. 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。**

2. 实验原理

1. **阻塞赋值与非阻塞赋值：**前者使用"="，在执行完这一步赋值前，**不允许**后续代码执行，即程序阻塞。而后者使用**"<="**，在一个always模块中，所有<=语句在模块结束时**同步执行。**
2. 上述的赋值方法，在逻辑电路中没有差别，而在**时序电路**中可能会导致不同，通常表现为两种不同赋值方法**结果相差一个时钟周期。**
3. D触发器在时钟正边沿触发，遵循激励方程：

```
D_VAL = D_DATA
```

4. **同步清零和异步清零**：区别在于：同步清零端在生效后需等待下一个时钟上升沿才能真正清零，而异步是即刻清零。

3. 实验环境/器材

1. Quartus: 这次没有使用systembuilder, 而是自己设置引脚。
2. 使用开关代替时钟（开发板的时钟频率过高）。
3. 对于4-1，给出其RTL视图，采用**EXP4_1顶层文件**，调用**block.v和unblock.v**，分别查看阻塞赋值和非阻塞赋值的效果。

4. 程序代码

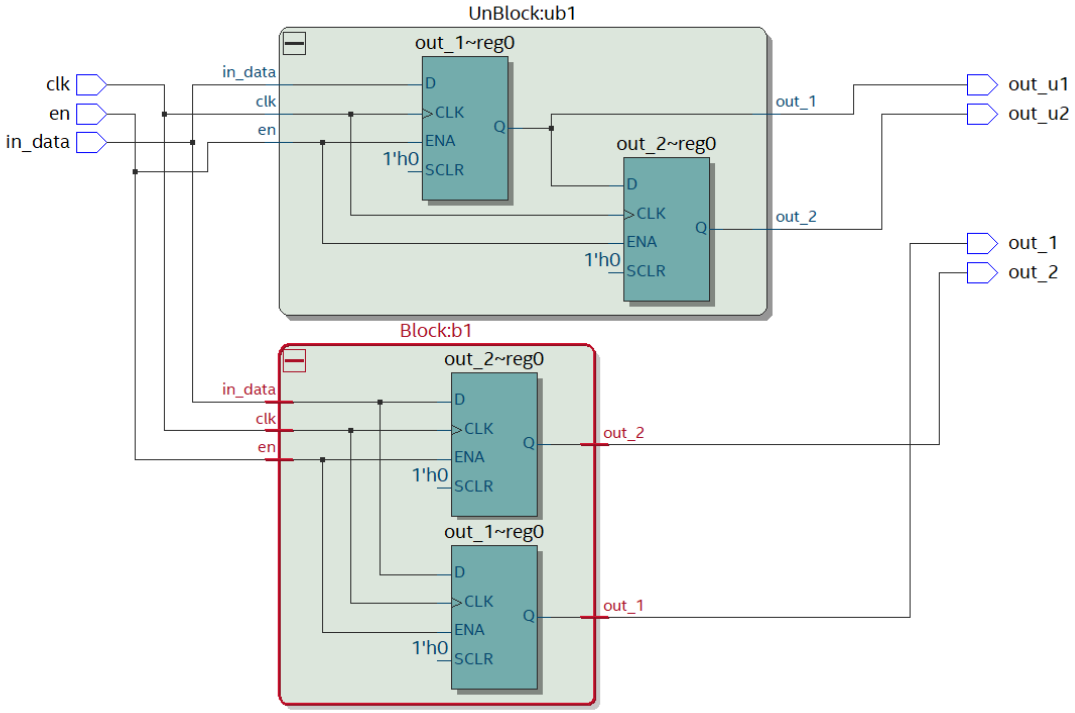
实验4_1

```
//顶层实体
module EXP4_1(
    input clk,
    input in_data,
    input en,
    output out_u1,
    output out_u2,
    output out_1,
    output out_2
);
Block b1(.clk(clk), .in_data(in_data), .en(en), .out_1(out_1),
.out_2(out_2));
UnBlock ub1(.clk(clk), .in_data(in_data), .en(en), .out_1(out_u1),
.out_2(out_u2));
endmodule
```

```
//非阻塞赋值模块
module UnBlock(
    input clk,
    input in_data,
    input en,
    output reg out_1,
    output reg out_2
);
always @ (posedge clk )

if(en)
    begin
        out_1 <= in_data;
        out_2 <= out_1;
    end
else
    begin
        out_1 <= out_1;
        out_2 <= out_2;
    end

endmodule

//阻塞赋值模块
module Block(
    input clk,
    input in_data,
    input en,
    output reg out_1,
    output reg out_2
);
always @ (posedge clk )
if(en)
    begin
        out_1 = in_data;
        out_2 = out_1;
    end
else
    begin
        out_1 = out_1;
```

```
        out_2 = out_2;
    end
endmodule
```

RTL视图



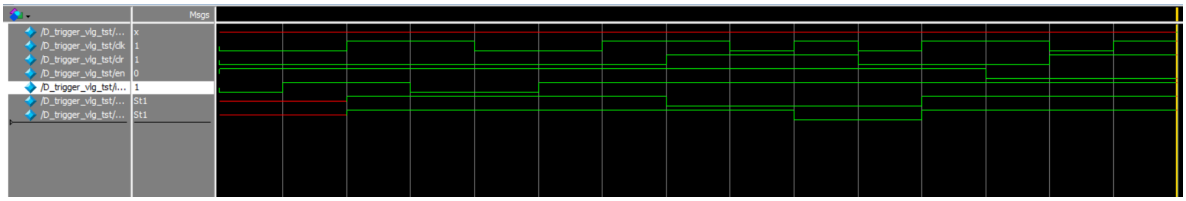
实验4_2

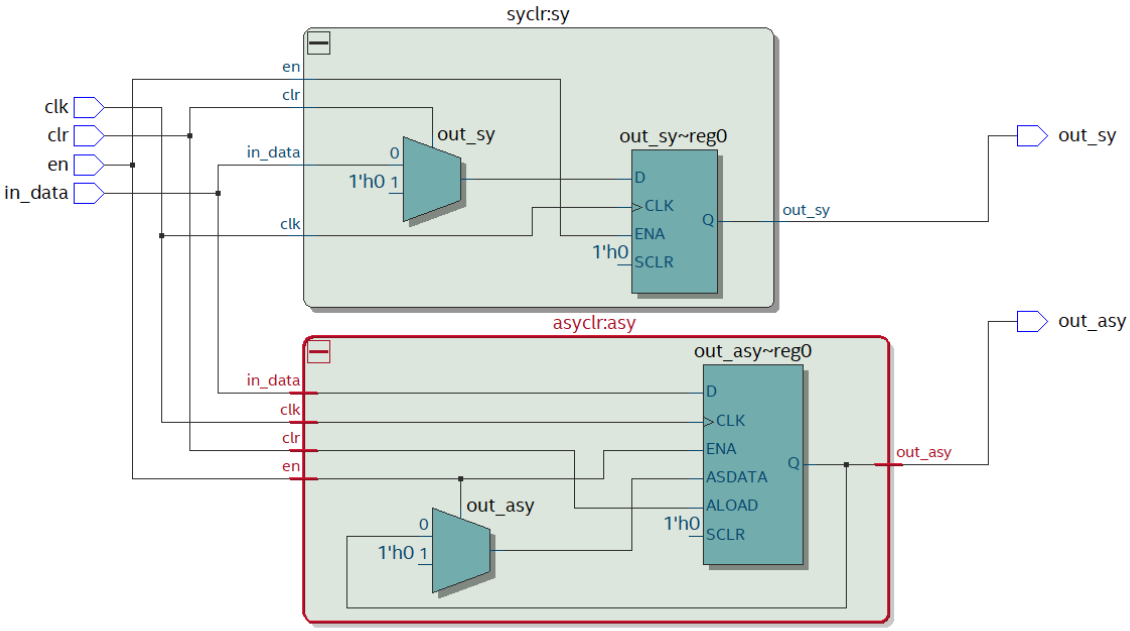
```
//顶层实体
module D_trigger(
    input en,
    input clk,//SW[3]
    input clr,
    input in_data,
    output out_sy,
    output out_asy //清零：SW[2],使能，SW[1]，数据SW[0]
);

asyclr asy(.en(en), .clk(clk), .clr(clr), .in_data(in_data),
.out_asy(out_asy));
syclr sy(.en(en), .clk(clk), .clr(clr), .in_data(in_data),
.out_sy(out_sy));


endmodule
```

```
//异步清零
module asyclr(
    input en,
    input clk,
    input clr,
    input in_data,
    output reg out_asy
);
```

```
always @ (posedge clk or posedge clr)
begin
    if(clr)
        if(en)
            out_asy <= 0;
        else out_asy <= out_asy;
    else
    begin
        if(en)
            out_asy <= in_data;
        else out_asy <= out_asy;
    end
end

endmodule
//同步清零
module syclr(
    input en,
    input clk,
    input clr,
    input in_data,
    output reg out_sy
);
always @ (posedge clk)
begin
    if(en)
        if(clr)
            out_sy <= 0;
        else out_sy <= in_data;
    else out_sy <= out_sy;
end

endmodule
```

```
//testbench
begin
// code executes for every event on sensitivity list
// insert code here --> begin
clk = 0; en = 1; clr = 0; in_data = 0; #10;
in_data = 1; #10;
clk = 1; in_data = 1; #10;
in_data = 0; #10;
clk = 0; #10;
in_data = 1; #10;
clk = 1; #10;
clr = 1; #10;
clk = 0; in_data = 1; #10;
clk = 1; #10;
clk = 0; clr = 0; #10;
clk = 1; in_data = 1; #10;
en = 0;
clk = 1; in_data = 1; #10;
clk = 0; in_data = 1; clr = 1; #10;
clk = 1; in_data = 1; #10;
@eachvec;
// --> end
end
```





5. 实验结果

结果符合预期，已验收，

6. 问题与解决方案/启示

1. 这期作业确实没什么问题，花了一个小时就写完了。
2. 大概要掌握多模块设计的技巧，这在后面的大项目中很有必要！

###