课后作业

always块语句中的阻塞和非阻塞赋值是Verilog学习的一大难点，每年都有很多同学都不遵守规范， 混用、用错，导致出现了各种奇怪的错误，花了大量时间debug。希望通过实验的作业，让大家了解非阻塞和阻塞赋值在仿真和综合异同。

本次作业对比时序逻辑always块中，无依赖的两行赋值语句，用阻塞和非阻塞赋值的异同。请大家分别对以下两份RTL代码在vivado进行RTL分析、综合、仿真，下板不做要求，并将RTL分析图、综合后电路图、仿真波形截图写到实验报告，并分别指出RTL分析图、综合后电路图、仿真波形上是相同还是不同。

使用阻塞赋值：

```
module ex1_block(
    input      clk,
    input      rst,
    input      A,
    input      B,
    output reg C,
    output reg D
);

    always @ (posedge clk or posedge rst) begin
        if(rst) begin
            C = 0;
            D = 0;
        end else begin
            C = A & B;
            D = A | B;
        end
    end

endmodule
```

用到的仿真代码：

```
`timescale 1ns/1ps
module ex1tb;
    reg clk=0;
    reg rst;
    reg A;
    reg B;
    wire C;
    wire D;

    ex1_block u_ex1_block (
        .clk(clk),
        .rst(rst),
        .A(A),
        .B(B),
        .C(C),
        .D(D)
    );

    always #5  clk = ! clk ;

    // Use monitor to track changes in signals A, B, C, and D
    initial begin
        $monitor("Time: %0t | A = %b, B = %b | C = %b, D = %b", $time, A, B, C, D);
    end

    initial begin
        rst = 1; A = 0; B = 0;
        # 10 rst = 0;
        #10 A = 0; B = 0;         // Test case 1
        #10 A = 0; B = 1;         // Test case 2
        #10 A = 1; B = 0;         // Test case 3
        #10 A = 1; B = 1;         // Test case 4
        #10;
        $finish;
    end
endmodule
```

改为非阻塞赋值：仿真代码中的模块例化请自行修改为ex1_nonblock。

```
module ex1_nonblock(
    input      clk,
    input      rst,
    input      A,
    input      B,
    output reg C,
    output reg D
);

    always @ (posedge clk or posedge rst) begin
        if(rst) begin
            C <= 0;
            D <= 0;
        end else begin
            C <= A & B;
            D <= A | B;
        end
    end

endmodule
```