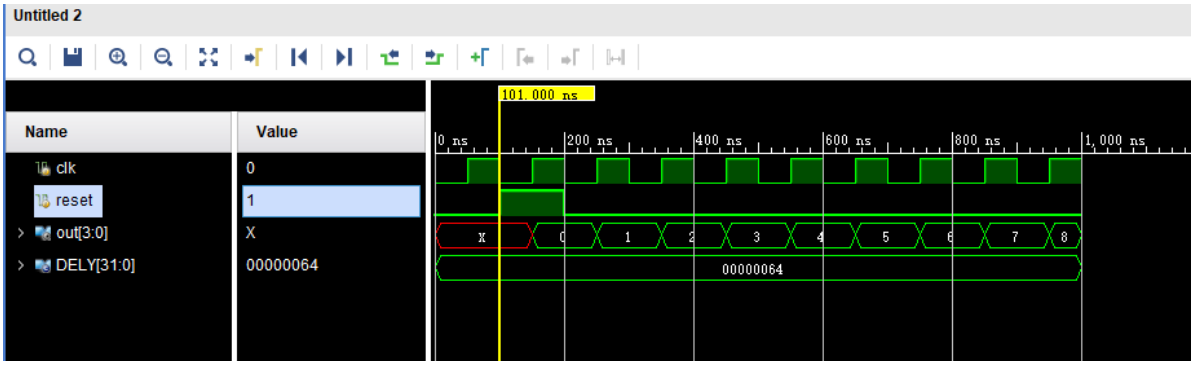
课后实验一：4位计数器

实验重难点

- 简单时序电路认识
- 基本verilog语法
  - always块
  - **非阻塞赋值**
  - if else使用
  - module的端口（input和output）
- testbench基本内容
  - 初始化（initial）
  - parameter的声明
  - 如何产生周期变化的clk信号
  - 模块调用
  - 使用monitor监控关键信息
  - 延迟的使用（#number）

实验内容

- 阅读4位计数器的模块代码和testbench代码
- 要求掌握计数器中包含的所有语法的含义
- 在vivado中新建工程count_4，将count_4模块和testbench加入后跑仿真（run simulation）
- 观察波形的内容，并能够描述清楚波形的每一个时钟周期信号变化的原因（即使波形发生变化的代码）



非阻塞赋值学习资料

- 课程网站参考书籍：
  - Verilog - A Guide to Digital Design and Synthesis

    7.2.1 blocking assignments

    7.2.2 nonblocking assignments
- QQ群分享：verilog数字系统设计教程-夏宇闻

  4.9.1 赋值语句

实验代码

count_4

```
`timescale 1ns / 1ps
module count_4(
    input reset,
    input clk,
    output reg [3:0] out//如果变量在always块中赋值，则在端口处声明reg类型
    );

    // reg类型表示需要被存储的类型数据
    // wire类型表示不需要存储的类型，通常理解为一根连线

/**********************************************************************************
*******************************/
    //posedge 表示上升沿，对应negedge为下降沿，
    //posedge和negedge分别对应上升沿触发和下降沿触发的功能模块
    //本例为上升沿触发的计数器，不必将verilog代码和逻辑电路图对应，学习行为级设计思考方式即可
    always @(posedge clk) begin// @（）中的内容表示敏感信号列表，当@（）中的信号发生变化或
者满足一定条件（clk到达上升沿或者下降沿时），always块中的语句才被执行
        if (reset) // 上升沿到达时，如果reset为1，表示需要置位，即置0
            out <= 0;
        else //上升沿到达时如果reset不为1，则计数，加1
            out <= out + 1; //注意赋值符号为<=，区别于=,<=称为非阻塞赋值，=称为阻塞赋值，
这里暂时不细讲。
    end
    // verilog不允许 out++ 的语法

/**********************************************************************************
*******************************/

endmodule
```

test_count_4

```
`timescale 1ns / 1ps
module test_count_4(

    );

reg clk;
reg reset;
wire [3:0] out;

//声明一个参数变量，用于初始化等，不可修改
parameter DELY = 100;

/*************************************************************************************/

initial begin
    //下面两行在仿真时同时执行
    clk = 1'b0;
    reset = 1'b0;
    //#number 表示在上面语句执行结束后number个时间单位后执行后面的语句
    #(DELY+1);
    reset = 1'b1;//该句在初始时间经过DELY+1时间后执行
    #DELY;
```

```
    reset = 1'b0;// 该句在上面 reset = 1'b1 执行后经过DELY个时间单位后执行，即在
                 // 初始时间经过DELY+1+DELY时间后执行
    #(DELY*20);
    $finish;// 在上面reset=1'b0执行后再经过20*DELY个时间单位后执行$finish，$finish表示仿
真结束，程序执行完毕退出
end

/*************************************************************************************/

always begin
    #(DELY/2);//每DELY/2个单位时间执行always块中的内容
    clk = ~clk;// clk信号取反
end

/*************************************************************************************/
/*实例化4位计数器模块
*输入：clk信号（时钟信号） reset置位信号
*输出：out输出计数信号
*/
count_4 count_4_inst_0(
    .clk(clk),
    .reset(reset),
    .out(out)
);

/*************************************************************************************/

initial begin// monitor用于监测信号变化，每次clk，reset，out发生变化时在控制台（Tcl
console）按照对应的格式输出结果
    $monitor($time,,,"clk=%d reset=%d out=%d ", clk, reset, out);
end

/*************************************************************************************/

// // not important
// // build for gtkwave
// initial begin
//     $dumpfile("test_count_4.vcd");
//     $dumpvars(0, test_count);
// end

endmodule
```

实验tips

- 没有验收和实验报告
- 自行学习了解，有问题直接在群里提问
- 没必要掌握某个语法例如always的所有用法，只需要掌握现有的例子中的用法即可
- 后续会有其他的课后实验，因此count_4是基础内容，务必掌握，**否则后续实验寸步难行**