实验名称：模拟延时 姓名： 学号：

装

订

线

浙江大学

本科实验报告

姓名：

学院：生物医学工程与仪器科学学院

系：生物医学工程系

专业：生物医学工程

学号：

指导教师：余锋

实验名称：模拟延时 姓名： 学号：

专业：生物医学工程
姓名：
学号：
日期：2025.4.15
地点：教 6－204

浙江大学 实验报告

课程名称：硬件描述语言 指导老师：余锋 实验名称：HDL 实验四

一、实验要求

二、实验代码与注释

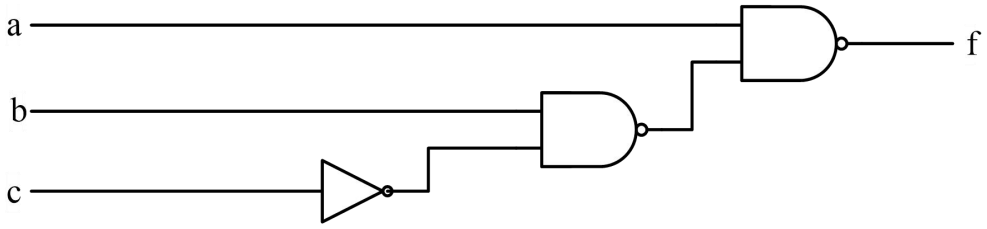
三、仿真结果与分析

四、讨论与心得

装 订 线

一、实验要求

实现如图所示的电路：



要求完成：

1、 circuit_logic.v

完成如图所示电路的逻辑功能

2、 circuit_timing.v

在 circuit_logic.v 的基础上，使用行为描述方式，加入器件延时和线路延时，模拟实现如下时序：

a）a 到 f 的总延时为 20ns（上升沿和下降沿）

b）b 到 f 的总延时为 20ns（上升沿和下降沿）

c）c 到 f 的总延时为 15ns（上升沿和下降沿）

d）器件延时 5ns，线路延时自由分配

实验名称：___模拟延时___ 姓名：__ 学号：

3、 tb_circuit.v

完成 circuit_logic.v 和 circuit_timing.v 模块的仿真验证，测试激励覆盖以下情况，每 40ns 改变一次输入

输入			输出
a	b	c	f
0	0	0	
0	0	1	
0	1	1	
1	1	1	
1	0	0	
1	z	0	
0	z	0	
0	z	x	
x	x	x	
z	z	z	

装 订 线

4、 分析讨论输入存在未定值/高阻态时的输出情况、延时对上述电路逻辑功能的影响等内容

二、实验代码与注释

1、circuit_logic.v

```
//circuit_logic.v
module circuit_logic(
    input               a_i,
    input               b_i,
    input               c_i,
    output              f_o
);
//add implementation code here
    wire w1, w2;

    // 门级原语
```

实验名称：___模拟延时___ 姓名：__ 学号：

```
    not G1 (w1, c_i);
    nand G2 (w2, b_i, w1);
    nand G3 (f_o, a_i, w2);
endmodule
```

circuit_logic.v 采用门级原语（not、nand）实现纯逻辑功能，不考虑任何延时，用于验证电路的逻辑正确性。

2、circuit_timing.v

```
//circuit_timing.v
module circuit_timing(
    input               a_i,
    input               b_i,
    input               c_i,
    output              f_o
);
//add implementation code here
    wire a_i_d, b_i_d, c_i_d;
    wire w1, w2;

    // 线路延时
    assign #15 a_i_d = a_i;
    assign #10 b_i_d = b_i;
    // 器件延时
    assign #5 w1 = ~c_i;
    assign #5 w2 = ~(b_i_d & w1);
    assign #5 f_o = ~(a_i_d & w2);
endmodule
```

circuit_timing.v 在逻辑模块基础上，通过 assign #delay 语句添加线路延时和器件延时，模拟实际电路的时序特性，满足实验要求的延时参数。

3、tb_circuit.v

```
//tb_circuit.v
module tb_circuit();
    reg         a;
    reg         b;
    reg         c;
    wire        f_logic;
    wire        f_timing;
    //add simulation code here
```

装 订 线

实验名称：___模拟延时___ 姓名：__ 学号：

装

订

线

```
    circuit_logic UUTlogic (
        .a_i(a),
        .b_i(b),
        .c_i(c),
        .f_o(f_logic)
    );
    circuit_timing UUTtiming (
        .a_i(a),
        .b_i(b),
        .c_i(c),
        .f_o(f_timing)
    );
    initial begin
        // Initialize inputs
        a = 0;
        b = 0;
        c = 0;
        // Wait for global reset to finish
        #10;

        // Apply test vectors every 40ns
        // Time  a b c   Expected f (logic)
        // ----  - - -   ------------------
        #40; a = 0; b = 0; c = 1; // Expected f = NAND(0, NAND(0, 0)) = 1
        #40; a = 0; b = 1; c = 1; // Expected f = NAND(0, NAND(1, 0)) = 1
        #40; a = 1; b = 1; c = 1; // Expected f = NAND(1, NAND(1, 0)) = 1
        #40; a = 1; b = 0; c = 0; // Expected f = NAND(1, NAND(0, 1)) = 1
        #40; a = 1; b = 1'bz; c = 0; // Expected f = NAND(1, NAND(z, 1)) =
NAND(1, z) = x
        #40; a = 0; b = 1'bz; c = 0; // Expected f = NAND(0, NAND(z, 1)) =
NAND(0, z) = 1
        #40; a = 0; b = 1'bz; c = 1'bx; // Expected f = NAND(0, NAND(z, x))
= NAND(0, x) = 1
        #40; a = 1'bx; b = 1'bx; c = 1'bx; // Expected f = NAND(x, NAND(x,
x)) = NAND(x, x) = x
        #40; a = 1'bz; b = 1'bz; c = 1'bz; // Expected f = NAND(z, NAND(z,
z)) = NAND(z, x) = x
        #40; // Add a final delay to observe the last output
        $finish;
    end
    initial begin
        // Monitor the outputs
        $monitor("Time: %0t | a: %b | b: %b | c: %b | f_logic: %b | f_timing: %b",
$time, a, b, c, f_logic, f_timing);
```

实验名称：___模拟延时___ 姓名：__ 学号：

```
        $dumpfile("tb_circuit.vcd");
        $dumpvars(0, tb_circuit);

    end

endmodule
```

a 路径：线路延时 15ns（总延时 = 15ns 线路 + 5ns 器件 = 20ns）。

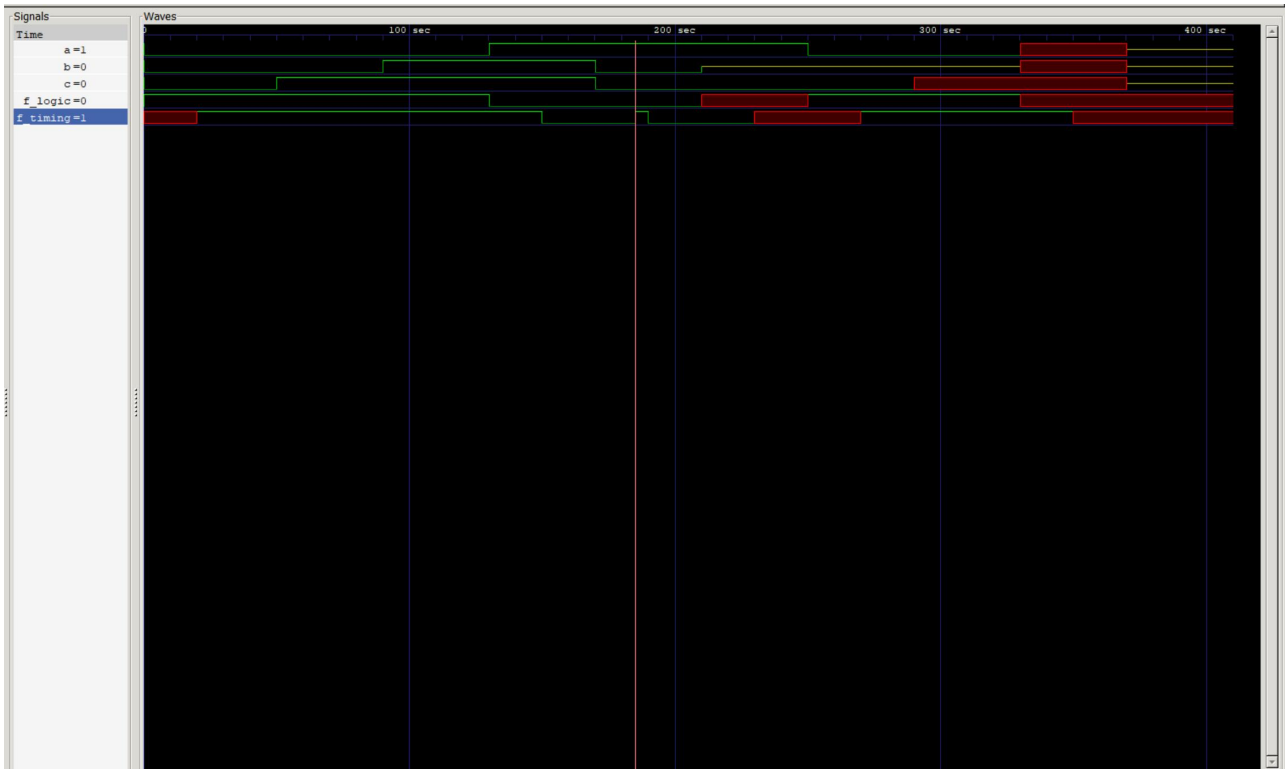
b 路径：线路延时 10ns（总延时 = 10ns 线路 + 5ns（第二级）+ 5ns（第三级）= 20ns）。

c 路径：无线路延时（总延时 = 5ns（第一级）+ 5ns（第二级）+ 5ns（第三级）= 15ns）。

三、仿真结果与分析

装 订 线

本次实验使用 Icarus Verilog (iverilog) 和 VVP 对 tb_circuit.v 测试平台进行了仿真，该平台同时实例化了无延时的 circuit_logic 和带延时的 circuit_timing 模块。



时间 /ns	a	b	c	f_logic	f_timing	分析

实验名称：___模拟延时___ 姓名：__ 学号：

0	0	0	0	1	x	初始状态，f_logic 立即输出 NAND(0, NAND(0, 1)) = 1，f_timing 输出未知 x。
20	0	0	0	1	1	f_timing 经过最长路径延时 (a/b->f, 20ns) 后稳定为 1。
50	0	0	1	1	1	输入 c 变化。逻辑 NAND(0, NAND(0, 0)) = 1。f_timing 稳定值不变。
90	0	1	1	1	1	输入 b 变化。逻辑 NAND(0, NAND(1, 0)) = 1。f_timing 稳定值不变。
130	1	1	1	**0**	1	输入 a 变化。逻辑 NAND(1, NAND(1, 0)) = 0。f_timing 尚未响应。
150	1	1	1	0	**0**	f_timing 经过 a->f 延时 (20ns) 后稳定为 0 (130ns + 20ns = 150ns)。
170	1	0	0	**0**	0	输入 b, c 变化。逻辑 NAND(1, NAND(0, 1)) = 0。f_timing 尚未响应。
185	1	0	0	0	**1**	! c 路径 (15ns) 比 b 路径 (20ns) 快，导致输出短暂变为 1。
190	1	0	0	0	**0**	f_timing 经过最长路径 b->f 延时 (20ns) 后稳定为 0 (170ns + 20ns = 190ns)。
210	1	z	0	**x**	0	输入 b 变为 'z'。f_logic 输出 x (NAND(1, x)). f_timing 尚未响应。
230	1	z	0	x	**x**	f_timing 经过 b->f 延时 (20ns) 后稳定为 x (210ns + 20ns = 230ns)。

装 订 线

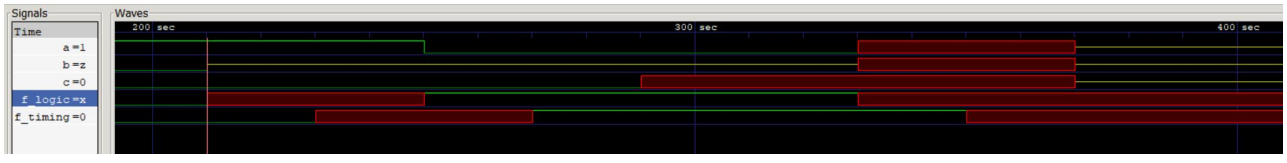
实验名称：___模拟延时___ 姓名：__ 学号：

250	0	z	0	**1**	x	输入 a 变为 '0'。f_logic 输出 1 (NAND(0, x)). f_timing 尚未响应。
270	0	z	0	1	**1**	f_timing 经过 a->f 延时 (20ns) 后稳定为 1 (250ns + 20ns = 270ns)。
290	0	z	x	1	1	输入 c 变为 'x'。逻辑 NAND(0, NAND(z, x)) = NAND(0, x) = 1。f_timing 稳定值不变。
330	x	x	x	**x**	1	输入 a, b, c 变为 'x'。逻辑 NAND(x, NAND(x, x)) = x。f_timing 尚未响应。
350	x	x	x	x	**x**	f_timing 经过最长路径延时 (20ns) 后稳定为 x (330ns + 20ns = 350ns)。
370	z	z	z	**x**	x	输入 a, b, c 变为 'z'。逻辑 NAND(z, NAND(z, z)) = NAND(z, x) = x。f_timing 稳定值不变 (已是 x)。

装 订 线

f_logic 的输出在每个测试向量施加后都与预期的逻辑功能 (NAND(a, NAND(b, ~c))) 一致，包括对 'x' 和 'z' 输入的处理。

（一）未定值与高阻态



1、Time=210ns (a=1, b=z, c=0)

观察到 f_logic 和 f_timing (在 230ns 后) 都变为 x。

f = NAND(a, NAND(b, ~c)) = NAND(1, NAND(z, ~0)) = NAND(1, NAND(z, 1))

根据 Verilog 规则，NAND(z, 1) 的结果是 x 或 z（通常传播为 x），而 NAND(1, x) 的结果是 x。因此最终输出为 x。

实验名称：___模拟延时___ 姓名：__ 学号：

2、Time=250ns (a=0, b=z, c=0)

观察到 f_logic 和 f_timing (在 270ns 后) 都变为 1。

f = NAND(0, NAND(z, ~0)) = NAND(0, NAND(z, 1)) = NAND(0, x)

因为与非门只要有一个输入为 0，输出就确定为 1，所以 NAND(0, x) 的结果是 1。

3、Time=290ns (a=0, b=z, c=x)

观察到 f_logic 和 f_timing 仍然为 1。

f = NAND(0, NAND(z, ~x)) = NAND(0, NAND(z, x)) = NAND(0, x)

同样，因为第一个输入是 0，输出确定为 1。

4、Time=330ns (a=x, b=x, c=x)

观察到 f_logic 在 330ns 时变为 x，f_timing 在 350ns 后也一同都变为 x。

f = NAND(x, NAND(x, ~x)) = NAND(x, NAND(x, x)) = NAND(x, x)

当所有输入都是未知时，输出也是未知的 x。

5、Time=370ns (a=z, b=z, c=z)

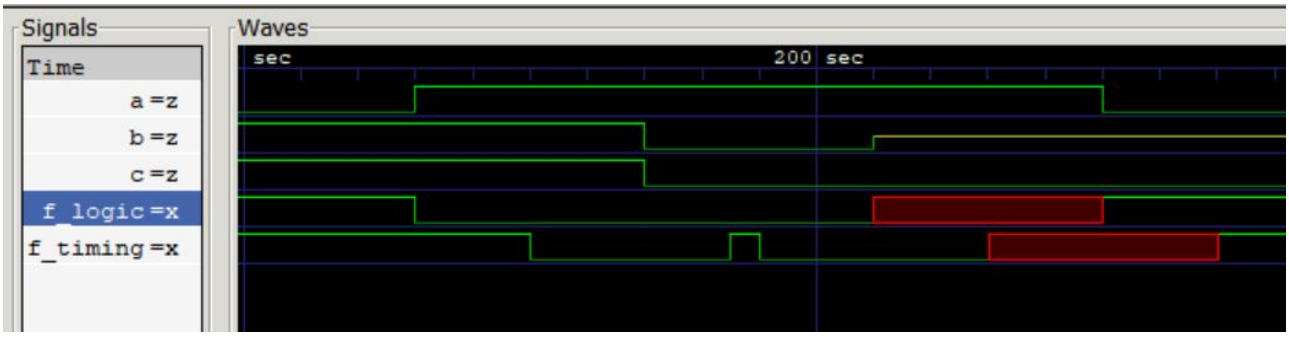
观察到 f_logic 和 f_timing 依然保持值 x。

f = NAND(z, NAND(z, ~z)) = NAND(z, NAND(z, x)) = NAND(z, x)

Verilog 中高阻态输入 z 在布尔运算中等效为 x，因此最终结果是 x。

（二）延时对电路逻辑功能的影响

对于所有确定的输入 (0/1)，一旦 f_timing 经过相应的延时稳定下来，其值与 f_logic 是一致的，表明延时不改变电路的最终稳态逻辑功能。



装 订 线

实验名称：模拟延时 姓名：__ 学号：

Time=170ns 时 (1,1,1 -> 1,0,0)， f_logic 直接从 0 变为 0(因为逻辑上 NAND(1,NAND(1,0))=0 和 NAND(1,NAND(0,1))=0);而 f_timing 在 Time=185ns 时短暂地变为 1，然后在 Time=190ns 才稳定到最终的 0 ——

其产生的原因是竞争冒险：输入 c 变化 (从 1 到 0) 影响输出的路径延时 (c->w1->w2->f_o 总共 15ns) 比输入 b 变化 (从 1 到 0) 影响输出的路径延时 (b->b_delayed->w2->f_o 总共 20ns) 要短，导致 c 的影响先到达输出端，使得输出短暂地变为 ~(1 & 0) = 1；之后 b 的影响到达，才使输出稳定为 ~(1 & 1) = 0。

虽然在目前 lab4 这个简单的组合逻辑实验中，这只是瞬态现象，但如果这个电路的输出 f_o 连接到时序逻辑（如触发器）的数据或时钟输入端，毛刺可能会被错误地采样，导致逻辑错误。同样，过长的延时可能导致时序电路的建立时间 (setup time) 或保持时间 (hold time) 违规。

装

订

线

四、讨论与心得

本次实验通过设计和仿真一个简单的组合逻辑电路及其带延时的版本，加深了我对 Verilog 硬件描述语言在逻辑建模和时序建模方面的理解。结果表明：确定值输入下，延时仅影响响应时间，不改变逻辑结果；高阻态 / 未定值输入会导致输出不确定，需在设计中避免；时序建模需精确分配线路与器件延时，以反映实际电路的信号传播规律。