**实验一**

1. **简单的led灯控制电路的设计**

**激励文件**

`timescale 1ns / 1ps

module led\_tb;

// Inputs

reg CLK; // 时钟输入

// Outputs

wire [1:0] led\_out; // LED 输出

// Instantiate the LED module

led dut (

.CLK(CLK), // 连接时钟输入

.led\_out(led\_out) // 连接 LED 输出

);

// Clock generation

initial begin

CLK = 0; // 初始化时钟

forever #5 CLK = ~CLK; // 生成周期为 10 个时间单位的时钟信号

end

// Stimulus

initial begin

// 等待时钟稳定

#448;

// 开始仿真

$dumpfile("led\_tb.vcd"); // 将波形结果导出到 VCD 文件

$dumpvars(0, led\_tb); // 导出模块变量

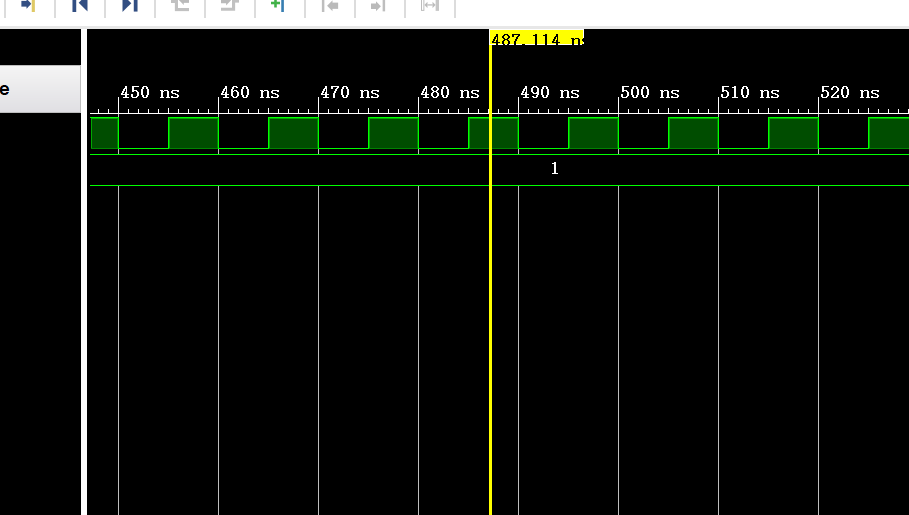
// 运行 1000 个时间单位的仿真

#448;

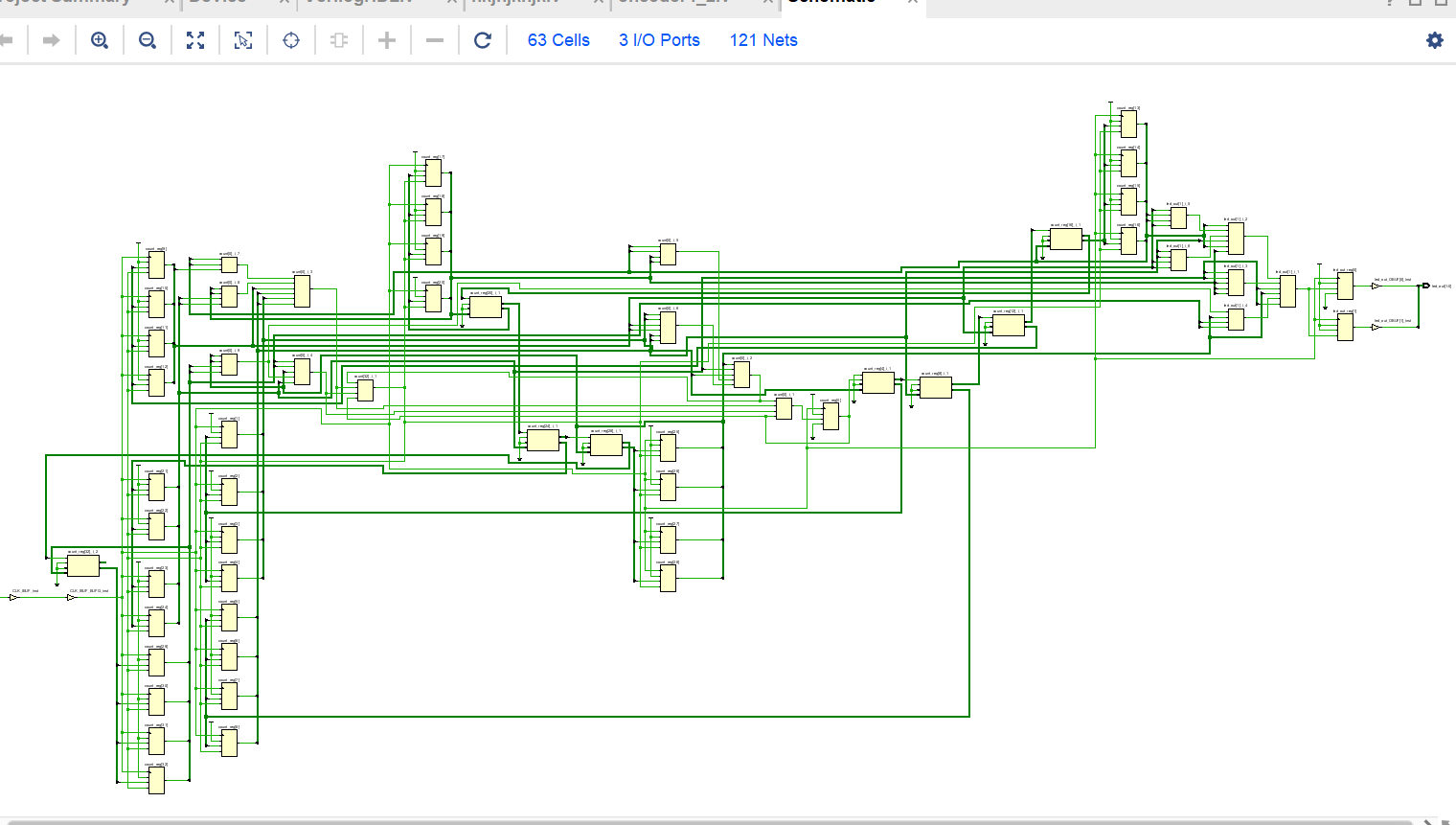
end

endmodule

**仿真结果**



**电路图**



**（二）一个简单编码器电路的设计**

**激励文件**

`timescale 1ns / 1ps

// 编码器4-2模块的测试台

module encoder4\_2\_tb;

// 输入

reg [3:0] d; // 输入数据

// 输出

wire [1:0] q; // 编码后的输出

// 实例化编码器4-2模块

encoder4\_2 dut (

.q(q), // 连接输出 q

.d(d) // 连接输入 d

);

// 刺激

initial begin

// 测试案例 1: d = 0111

d = 4'b0111;

#448; // 等待 448 个时间单位

// 测试案例 2: d = 1011

d = 4'b1011;

#448; // 等待 448 个时间单位

// 测试案例 3: d = 1101

d = 4'b1101;

#448; // 等待 448 个时间单位

// 测试案例 4: d = 1110

d = 4'b1110;

#448; // 等待 448 个时间单位

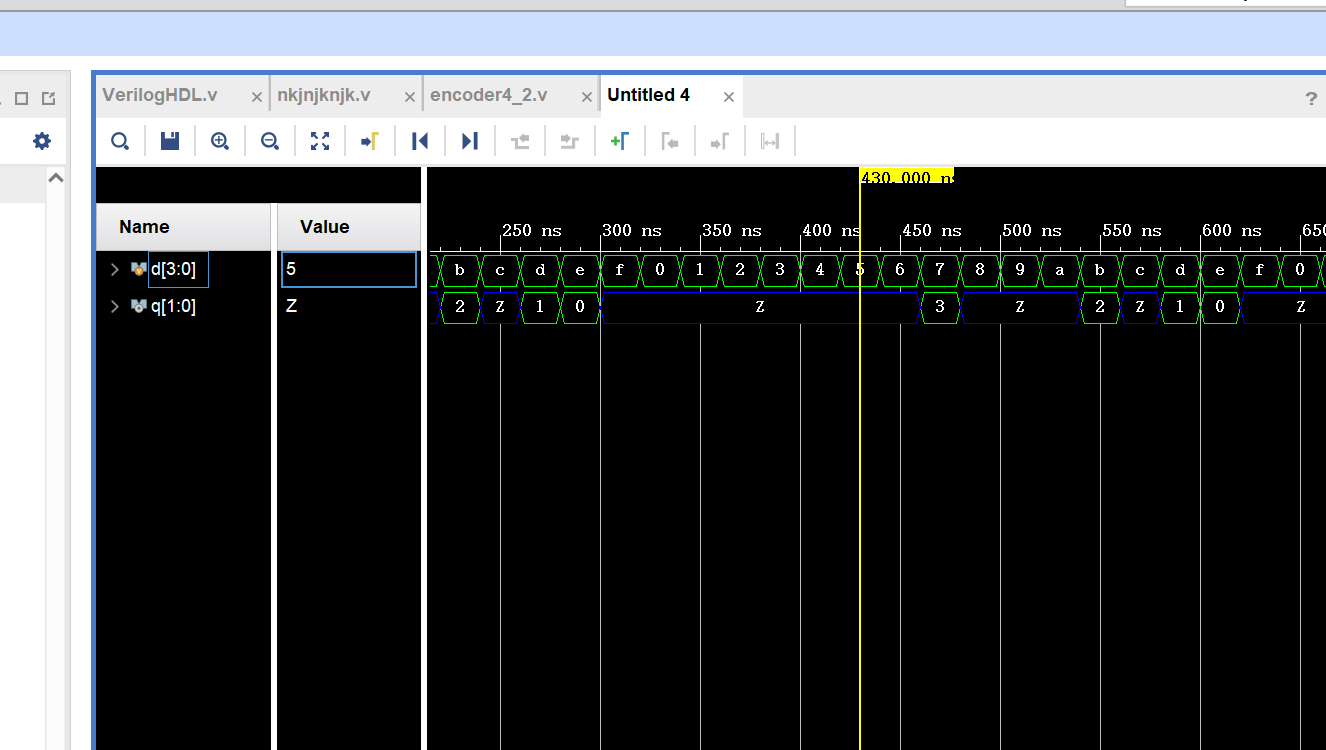
// 结束仿真

$finish;

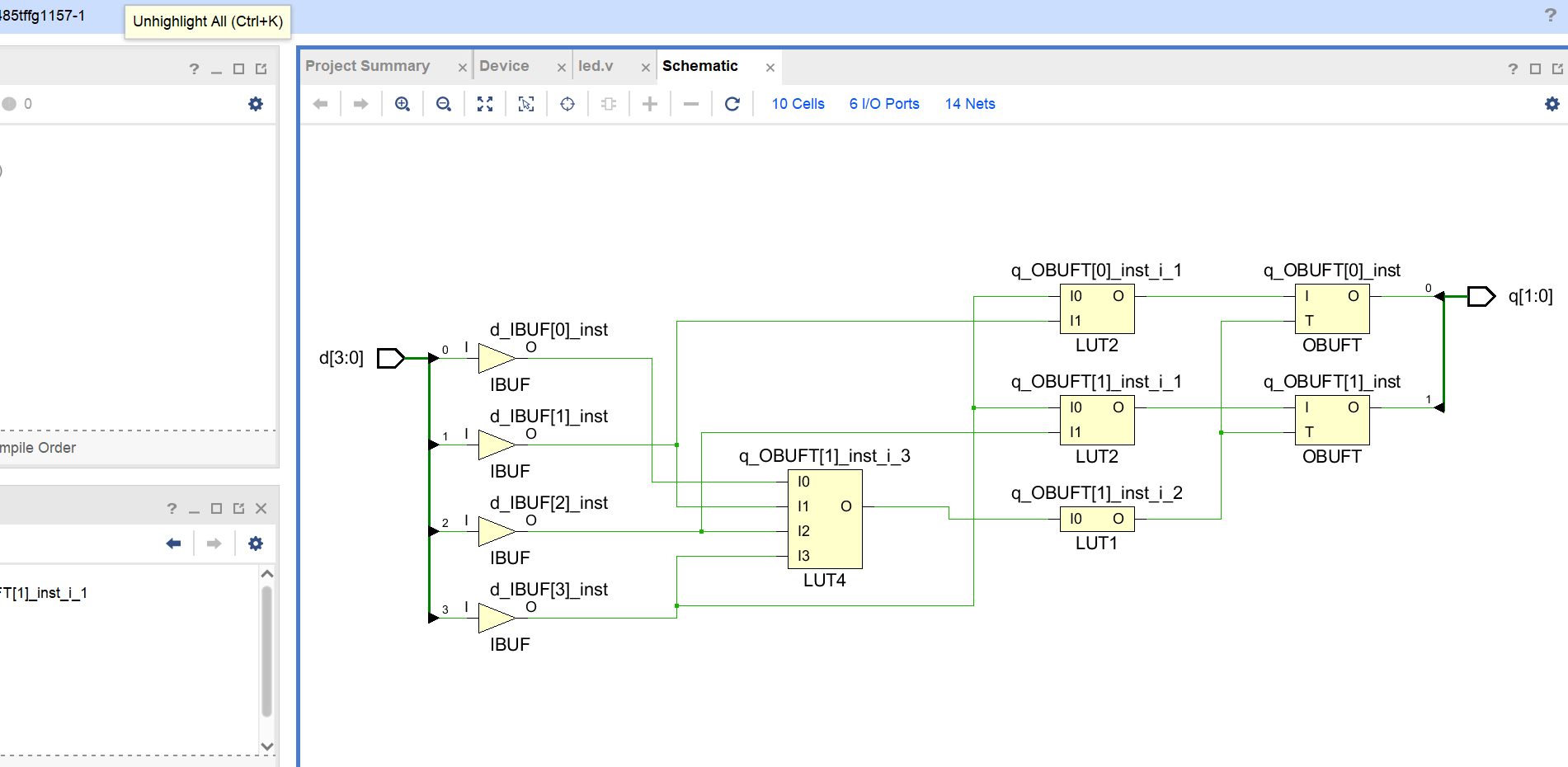
end

endmodule

**仿真结果**



**电路图**



**（三）一个简单比较器电路的设计**

**激励文件**

`timescale 1ns / 1ps

module comp\_tb;

// Inputs

reg CLK; // 时钟输入

reg RST; // 复位输入

reg [1:0] A; // 输入 A

reg [1:0] B; // 输入 B

// Outputs

wire AGTB; // 输出 AGTB

wire ALTB; // 输出 ALTB

wire AEQB; // 输出 AEQB

// 实例化比较器模块

comp dut (

.CLK(CLK), // 连接时钟输入

.RST(RST), // 连接复位输入

.A(A), // 连接输入 A

.B(B), // 连接输入 B

.AGTB(AGTB), // 连接输出 AGTB

.ALTB(ALTB), // 连接输出 ALTB

.AEQB(AEQB) // 连接输出 AEQB

);

// 仿真过程

initial begin

CLK = 0; // 初始化时钟

RST = 1; // 初始化复位

A = 2'b00; // 初始化输入 A

B = 2'b00; // 初始化输入 B

// 等待 100 个时间单位以稳定

#100;

// 开始仿真

$dumpfile("comp\_tb.vcd"); // 将波形结果导出到 VCD 文件

$dumpvars(0, comp\_tb); // 导出模块变量

// 输入 A 大于输入 B 的情况

A = 2'b10; B = 2'b01;

#7448;

// 输入 A 等于输入 B 的情况

A = 2'b01; B = 2'b01;

#7448;

// 输入 A 小于输入 B 的情况

A = 2'b01; B = 2'b10;

#7448;

end

// 时钟和输入信号翻转

always #448 CLK=~CLK;

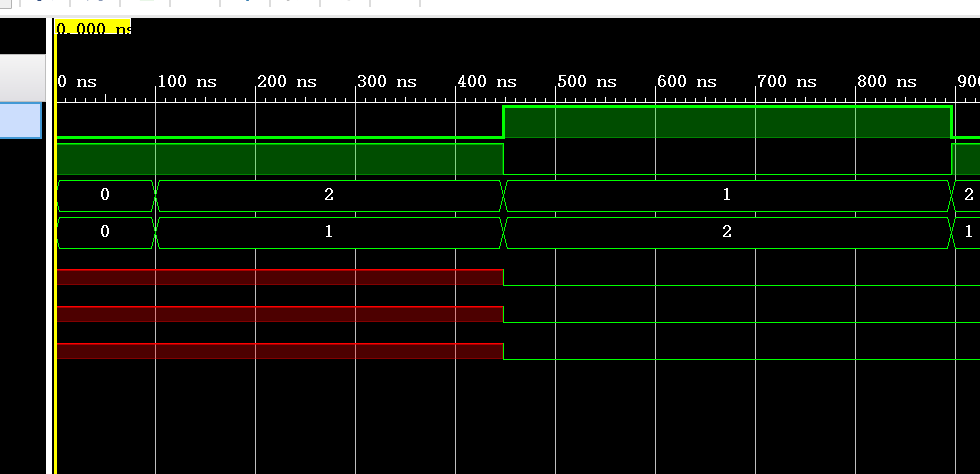
always #448 RST=~RST;

always #448 A=~A;

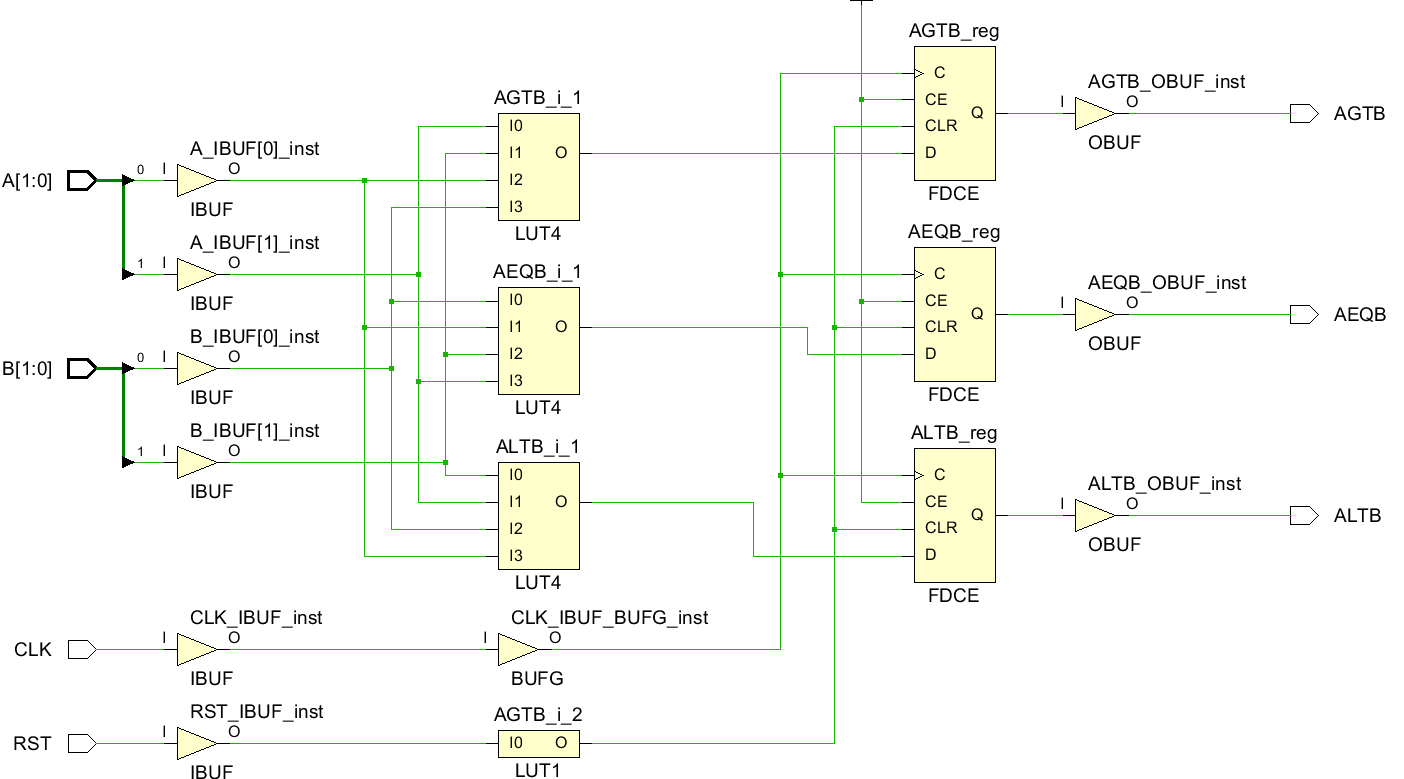
always #448 B=~B;

endmodule

**仿真结果**



**电路图**



**（四）一个简单全加器电路的设计**

**激励文件**

`timescale 1ns / 1ps

module full\_adder\_tb;

// Inputs

reg a; // 输入 a

reg b; // 输入 b

reg cin; // 输入 cin

// Outputs

wire sum; // 输出 sum

wire cout; // 输出 cout

// 实例化全加器模块

ful\_adder dut (

.a(a), // 连接输入 a

.b(b), // 连接输入 b

.cin(cin), // 连接输入 cin

.sum(sum), // 连接输出 sum

.cout(cout) // 连接输出 cout

);

// 仿真过程

initial begin

// 开始仿真

// 模拟输入信号

a = 1'b0; b = 1'b0; cin = 1'b0;

#10;

a = 1'b0; b = 1'b0; cin = 1'b1;

#10;

a = 1'b0; b = 1'b1; cin = 1'b0;

#10;

a = 1'b0; b = 1'b1; cin = 1'b1;

#10;

a = 1'b1; b = 1'b0; cin = 1'b0;

#10;

a = 1'b1; b = 1'b0; cin = 1'b1;

#10;

a = 1'b1; b = 1'b1; cin = 1'b0;

#10；

a = 1'b1; b = 1'b1; cin = 1'b1;

#10;

end

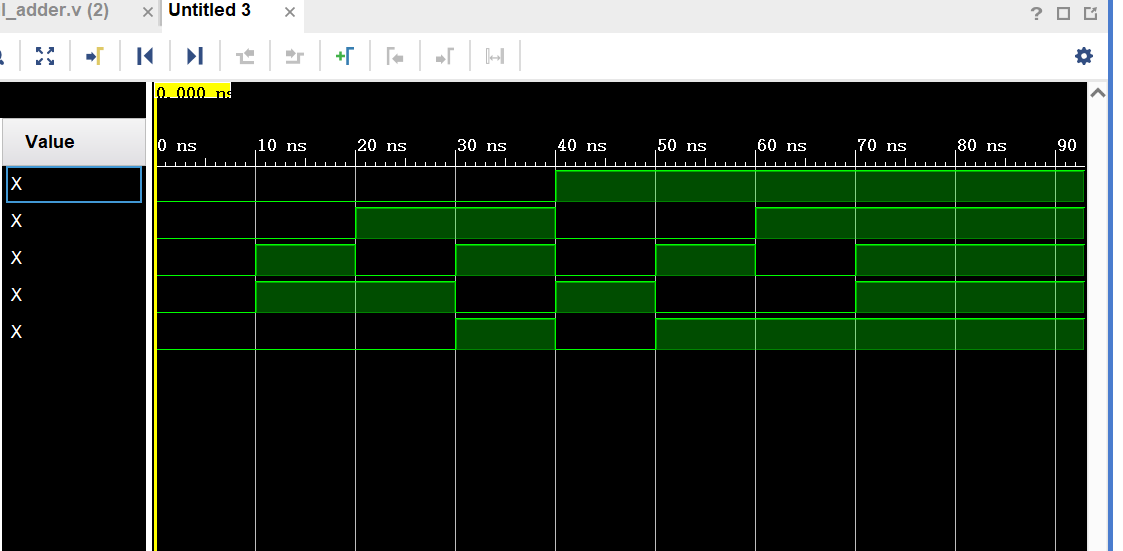
always#217448 a=~a;

always#217448 b=~b;

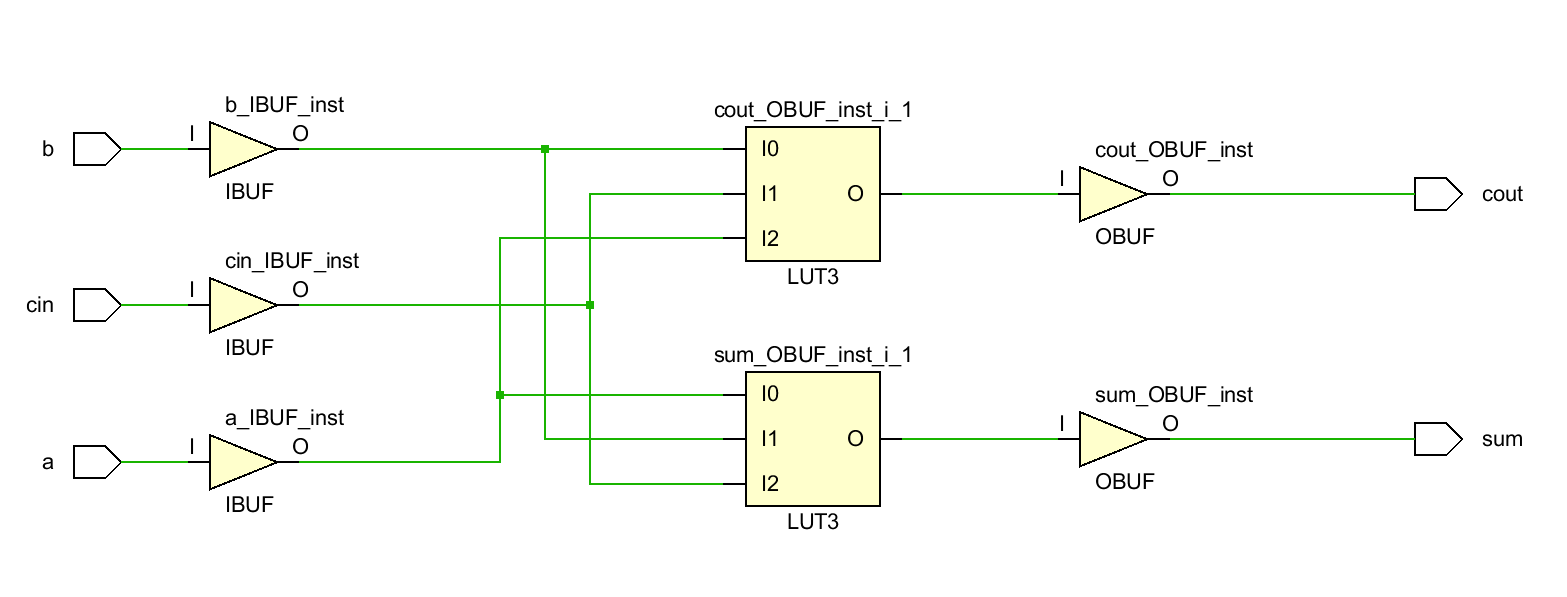
always#217448 cin=~cin;

endmodule

**仿真结果**



**电路图**



**实验二**

1. **D 触发器的实现**

**（1）基本的 D 触发器**

**激励文件**

`timescale 1ns / 1ps

module ddd;

// Inputs

reg clk; reg d;

// Outputs

wire q; wire qb;

// 实例化控制电路uut;

async\_rddf uut (

.clk(clk),

.d(d),

.q(q),

.qb(qb)

);

initial begin

// Initialize Inputs

clk = 0;

d = 0;

// Wait 100 ns

#100;

// Add stimulus here

// 模拟输入信号

clk = 0; d = 0;

#448;

clk = 0; d = 1;

#448;

clk = 1; d = 0;

#448;

clk = 1; d = 1;

#448;

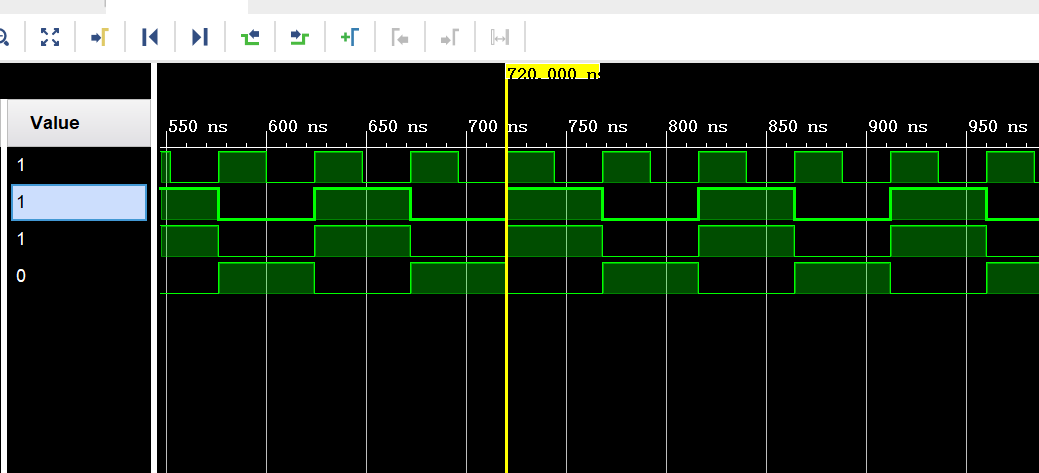
end

always #24 clk=~clk;

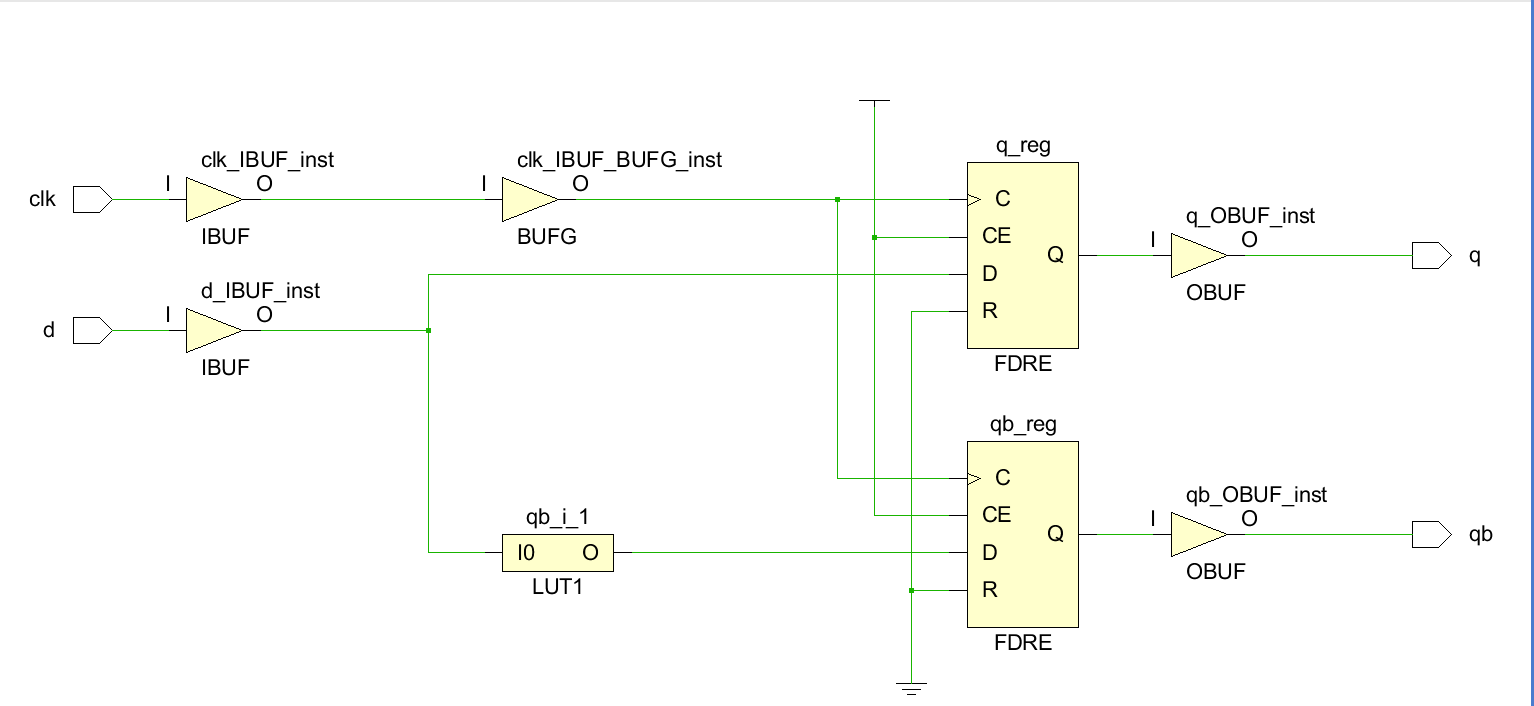
always #48 d=~d;

endmodule

**仿真结果**



**电路图**



**（2）同步复位的 D 触发器**

**激励文件**

`timescale 1ns / 1ps

module sync\_rddf\_tb;

// Inputs

reg clk; // 时钟输入

reg reset; // 复位输入

reg d; // 数据输入

// Outputs

wire q; // 输出 q

wire qb; // 输出 qb

// 实例化同步 D 触发器模块

sync\_rddf dut (

.clk(clk), // 连接时钟输入

.reset(reset),// 连接复位输入

.d(d), // 连接数据输入

.q(q), // 连接输出 q

.qb(qb) // 连接输出 qb

);

// 仿真过程

initial begin

// 开始仿真

#10;//准备

clk =0;

// 模拟输入信号

reset = 1; d = 0;

#24;//学号后两位的二分之一

reset = 0;

#48;//学号后两位

// 模拟数据输入变化

d = 1;

#24;//学号后两位的二分之一

d = 0;

#48;//学号后两位

d = 1;

#24;//学号后两位的二分之一

// 结束仿真

end

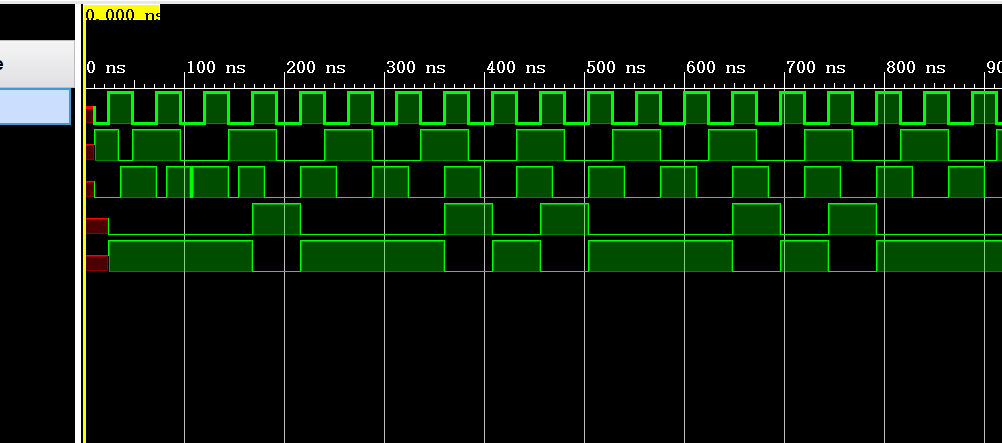
always #24 clk = ~clk;//学号后两位的二分之一

always #36 d = ~d;//学号后两位的三分之二

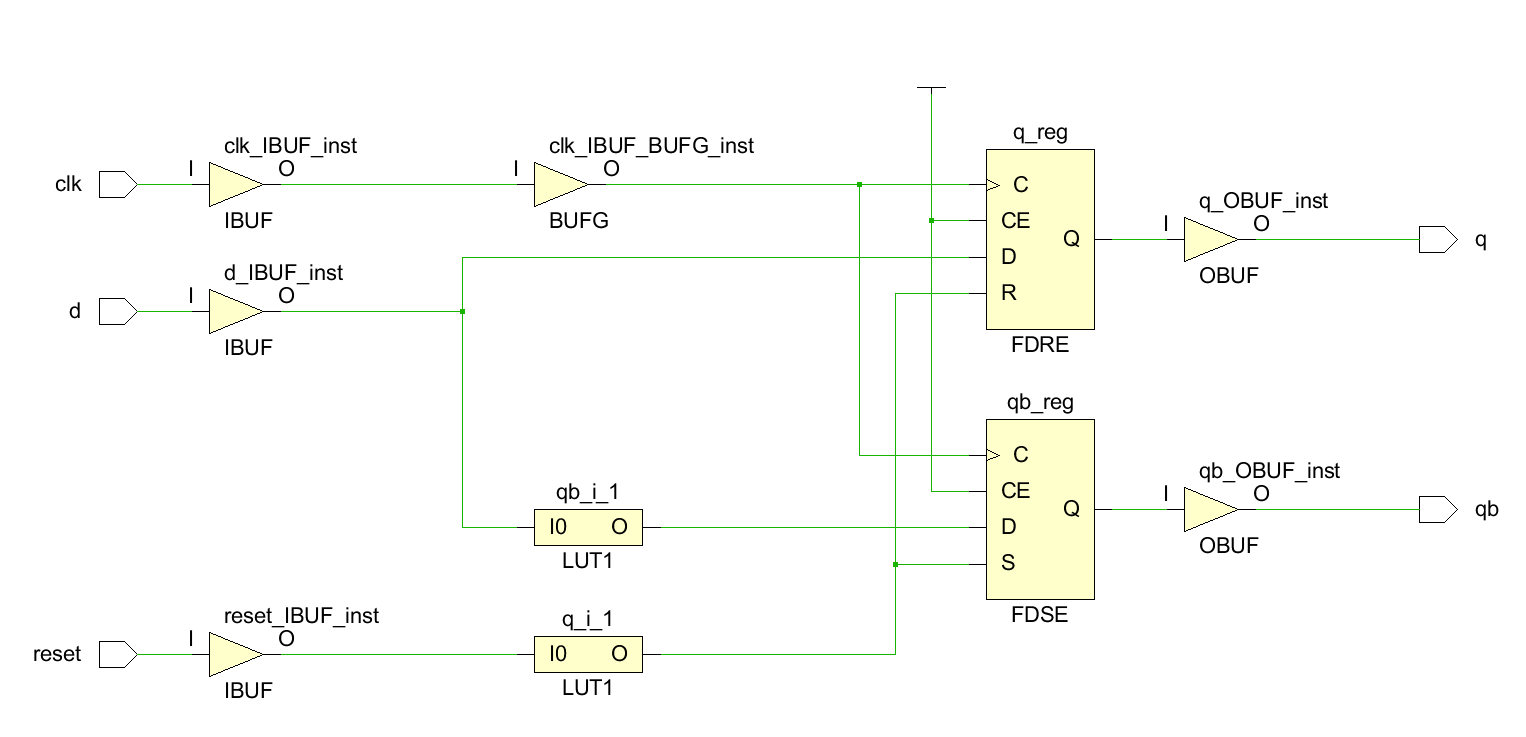
always #48 reset = ~reset;//学号后两位

endmodule

**仿真结果**



**电路图**



**（3）异步复位的 D 触发器**

**激励文件**

`timescale 1ns / 1ps

module async\_rddf\_tb;

// Inputs

reg clk; // 时钟输入

reg reset; // 复位输入

reg d; // 数据输入

// Outputs

wire q; // 输出 q

wire qb; // 输出 qb

// 实例化异步 D 触发器模块

async\_rddf dut (

.clk(clk), // 连接时钟输入

.reset(reset),// 连接复位输入

.d(d), // 连接数据输入

.q(q), // 连接输出 q

.qb(qb) // 连接输出 qb

);

// 仿真过程

initial begin

// 开始仿真

clk =0;

// 模拟输入信号

reset = 1; d = 0;

#48;

reset = 0;

#24;

// 模拟数据输入变化

d = 1;

#24;

d = 0;

#48;

d = 1;

#24;

end

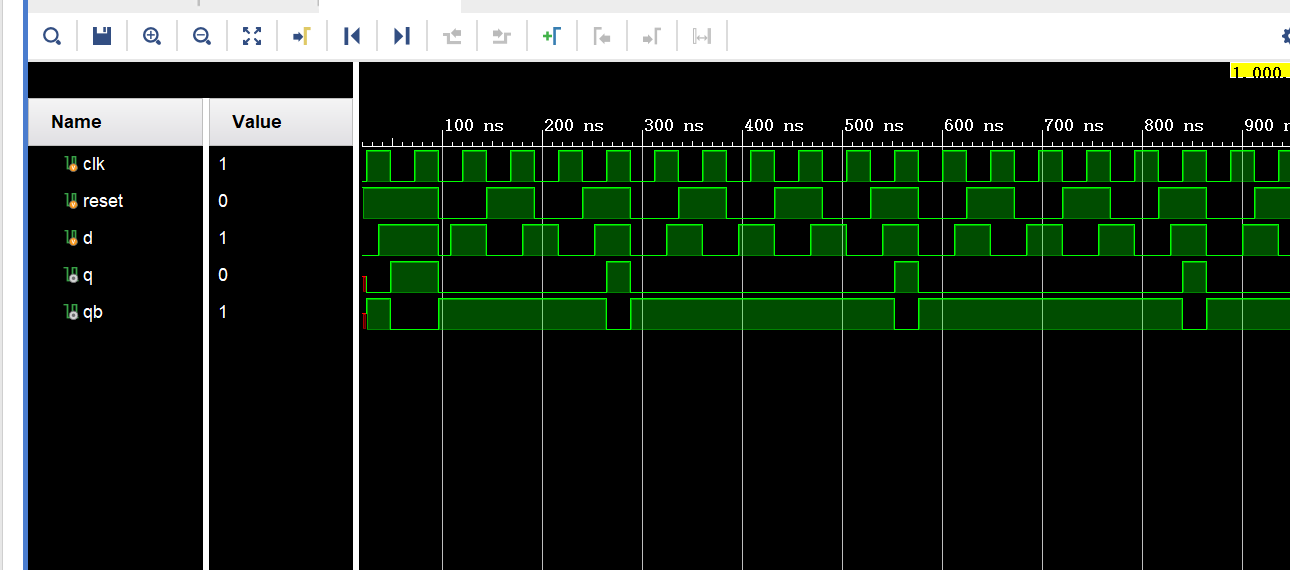
always #24 clk = ~clk;//学号后两位的二分之一

always #36 d = ~d;//学号后两位的三分之二

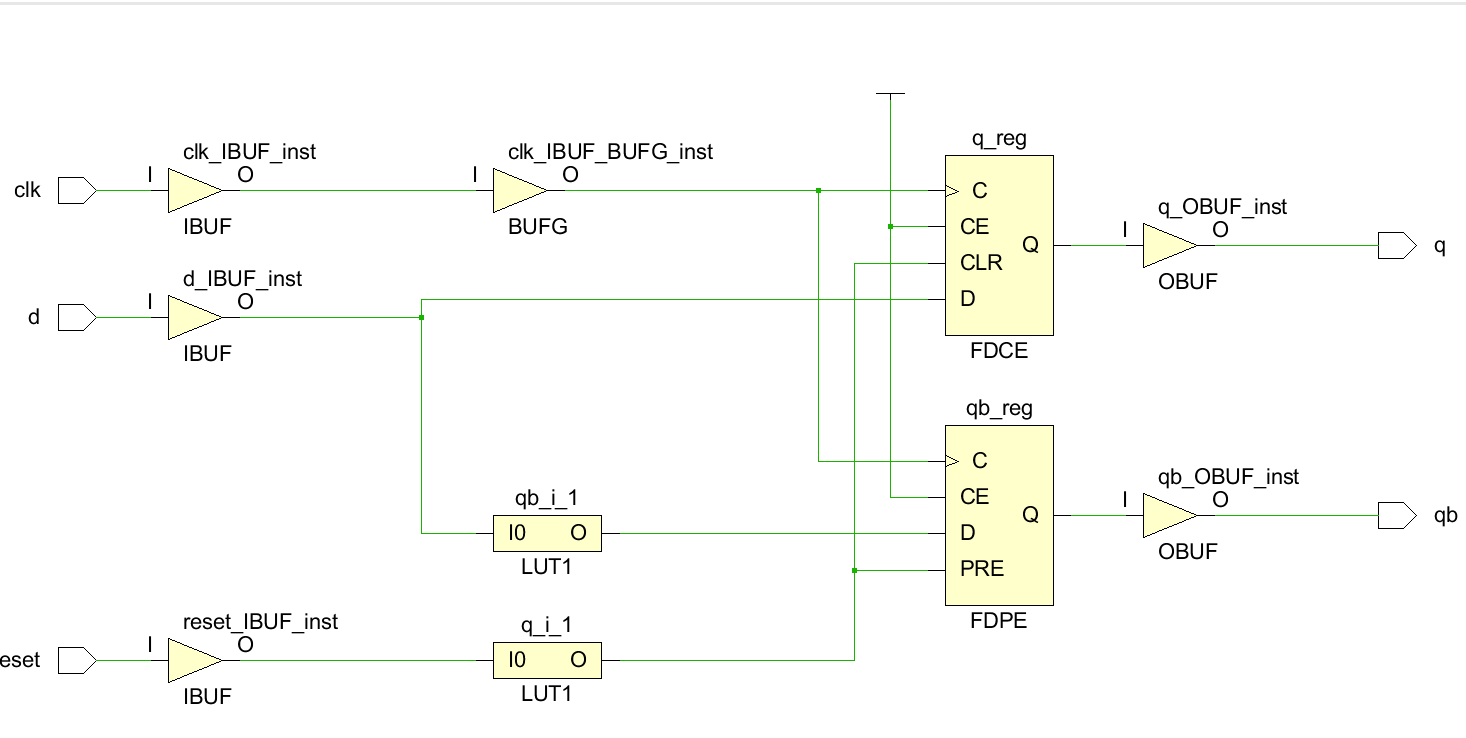
always #48 reset = ~reset;//学号后两位

endmodule

**仿真结果**



**电路图**



**（4）加法计数器**

**激励文件**

module addcounter (

input clk, // 时钟输入

output reg [3:0] Q // 输出 Q

);

reg [3:0] next\_Q; // 下一个 Q 的值

// 初始化 Q

initial begin

Q = 4'b0000; // 初始化为全零

end

// 加法计数逻辑

always @(posedge clk) begin

// 在时钟上升沿时更新 Q 的值

next\_Q = Q + 1; // Q 加一

end

// 更新 Q

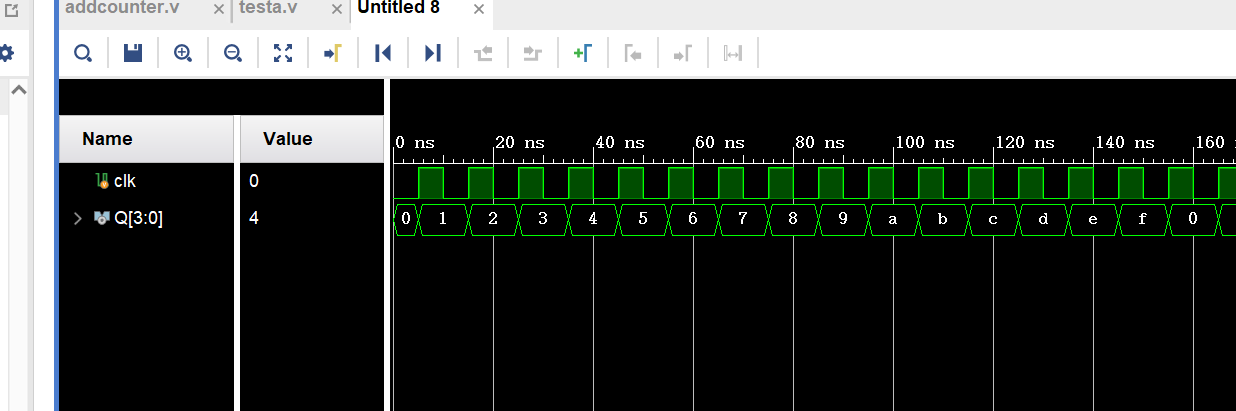
always @(posedge clk) begin

Q <= next\_Q;

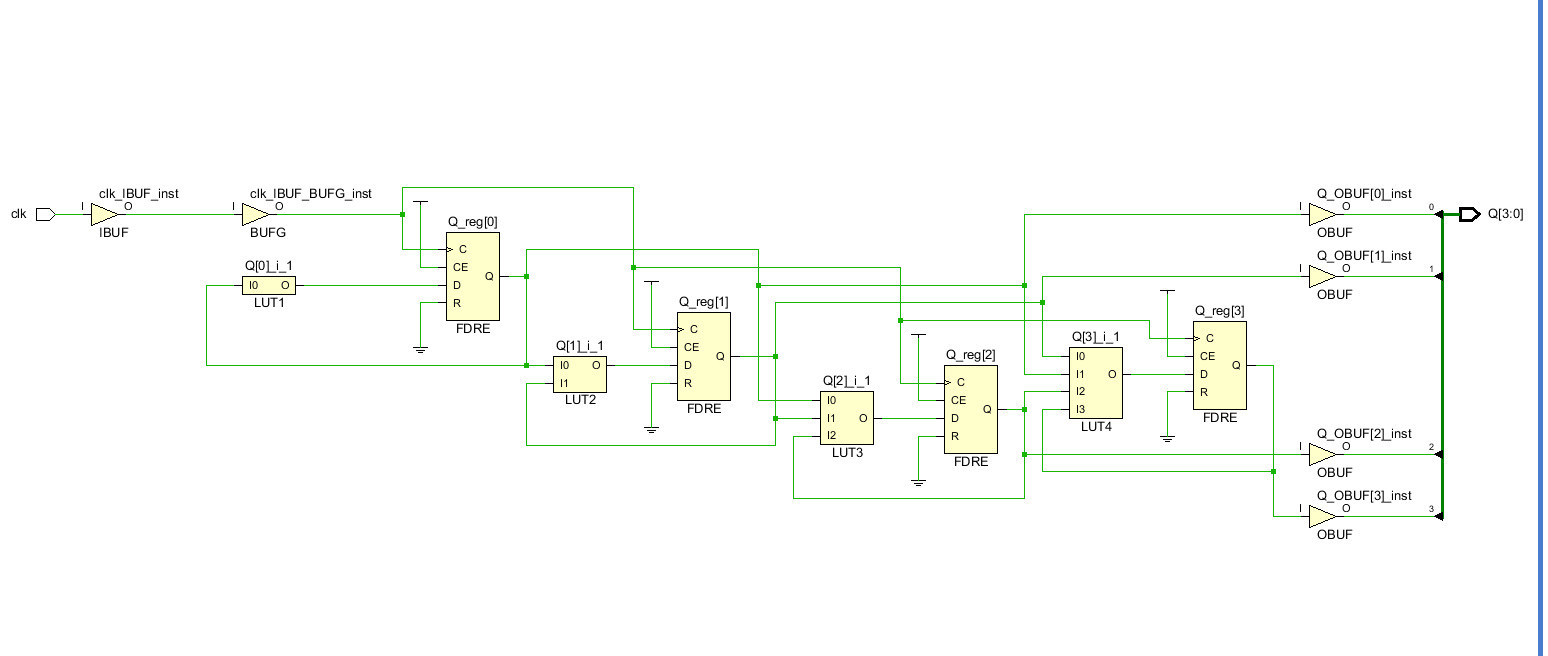
end

endmodule

**仿真结果**



**电路图**



**2.** **减法计数器**

**激励文件**

`timescale 1ns / 1ps

module subcounter\_tb;

reg clk; // 时钟输入

wire [3:0] Q; // 输出 Q

// 实例化待测试的 subcounter 模块

subcounter dut (

.clk(clk),

.Q(Q)

);

// 时钟生成

initial begin

clk = 0; // 初始化时钟

// 产生时钟脉冲，每 5 个时间单位切换一次

forever #5 clk = ~clk;

end

// 打印 Q 的值

initial begin

// 等待一段时间以观察输出

#10;

// 打印初始 Q 的值

$display("Initial Q: %b", Q);

// 每 5 个时间单位打印一次 Q 的值，持续一段时间

repeat (448) begin

#5;

$display("Q at time %0t: %b", $time, Q);

end

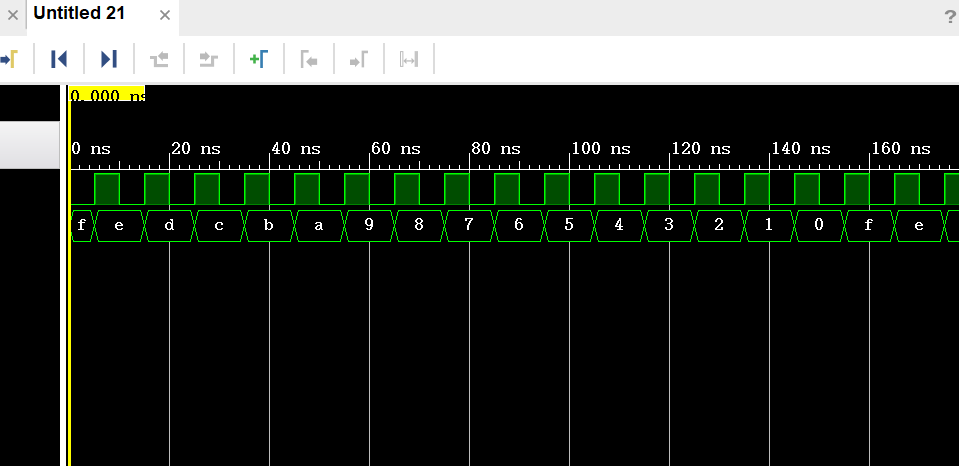
// 结束仿真

$finish;

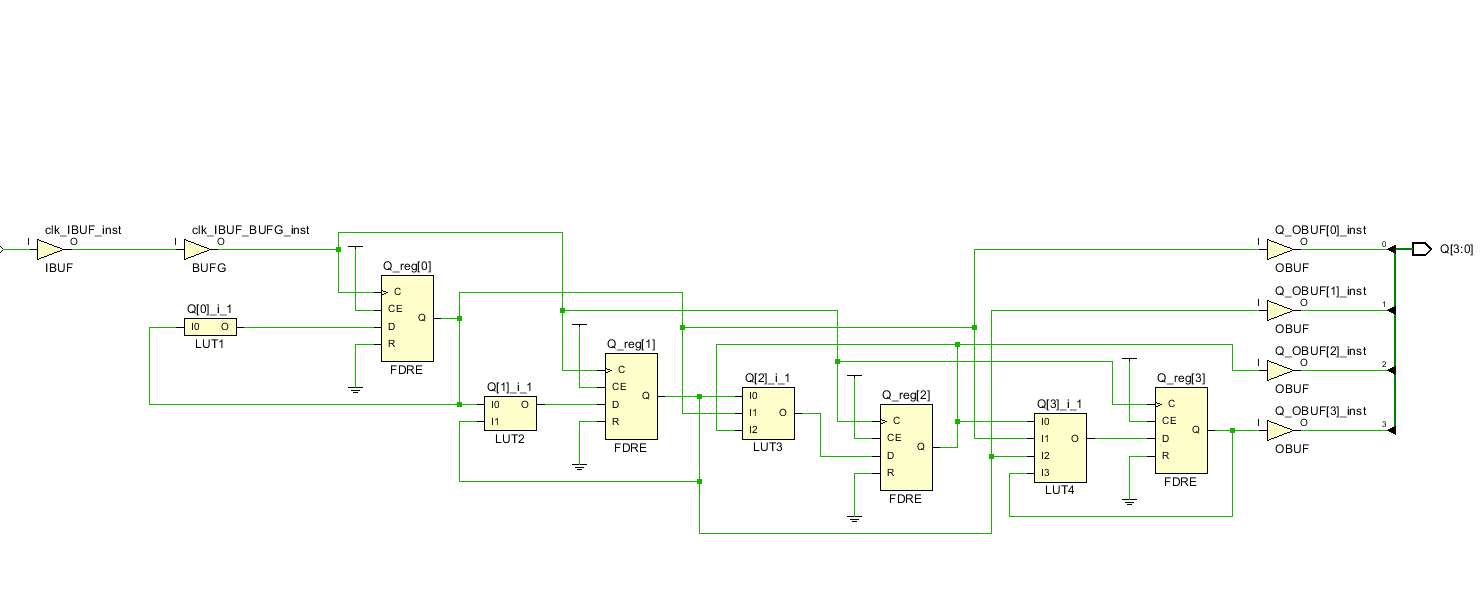
end

endmodule

**仿真结果**



**电路图**



**3.有限状态机(FSM) 的实现**

**激励文件**

`timescale 1ns / 1ps

module fsm\_all\_cases\_tb;

reg Clock = 0; // 时钟输入，初始化为0

reg Reset = 0; // 复位信号输入，初始化为0

reg A = 0; // 输入信号 A，初始化为0

wire F, G; // 输出信号 F 和 G

// 实例化待测试的 fsm 模块

fsm dut (

.Clock(Clock),

.Reset(Reset),

.A(A),

.F(F),

.G(G)

);

// 时钟生

always #1 Clock = ~Clock; // 产生时钟脉冲，每1个时间单位切换一次

always #10 Reset = ~Reset; // 产生时钟脉冲，每 10 个时间单位切换一次

always #2 A = ~A; // 产生时钟脉冲，每 20个时间单位切换一次

// 仿真测试序列

initial begin

#10; // 等待一段时间确保时钟和输入信号被正确初始化

// 第一个测试序列：Reset=0, A=0

Reset = 0; A = 0;

#448;//学号后三位

$display("Test sequence 1: Reset=0, A=0");

$display("Reset=%b, A=%b, F=%b, G=%b", Reset, A, F, G);

// 记录更多的时钟周期

#448;

// 在仿真波形图中添加观察点，以便查看更多的信号值

$monitor("Clock=%b, Reset=%b, A=%b, F=%b, G=%b", Clock, Reset, A, F, G);

// 第二个测试序列：Reset=0, A=1

Reset = 0; A = 1;

#448;

$display("Test sequence 2: Reset=0, A=1");

$display("Reset=%b, A=%b, F=%b, G=%b", Reset, A, F, G);

#448;

$monitor("Clock=%b, Reset=%b, A=%b, F=%b, G=%b", Clock, Reset, A, F, G);

// 其他测试序列类似...

// 第三个测试序列：Reset=1, A=0

Reset = 1; A = 0;

#448;

$display("Test sequence 3: Reset=0, A=0");

$display("Reset=%b, A=%b, F=%b, G=%b", Reset, A, F, G);

#448;

$monitor("Clock=%b, Reset=%b, A=%b, F=%b, G=%b", Clock, Reset, A, F, G);

// 第四个测试序列：Reset=1, A=1

Reset = 1; A = 1;

#448;

$display("Test sequence 4: Reset=1, A=1");

$display("Reset=%b, A=%b, F=%b, G=%b", Reset, A, F, G);

#448;

$monitor("Clock=%b, Reset=%b, A=%b, F=%b, G=%b", Clock, Reset, A, F, G);

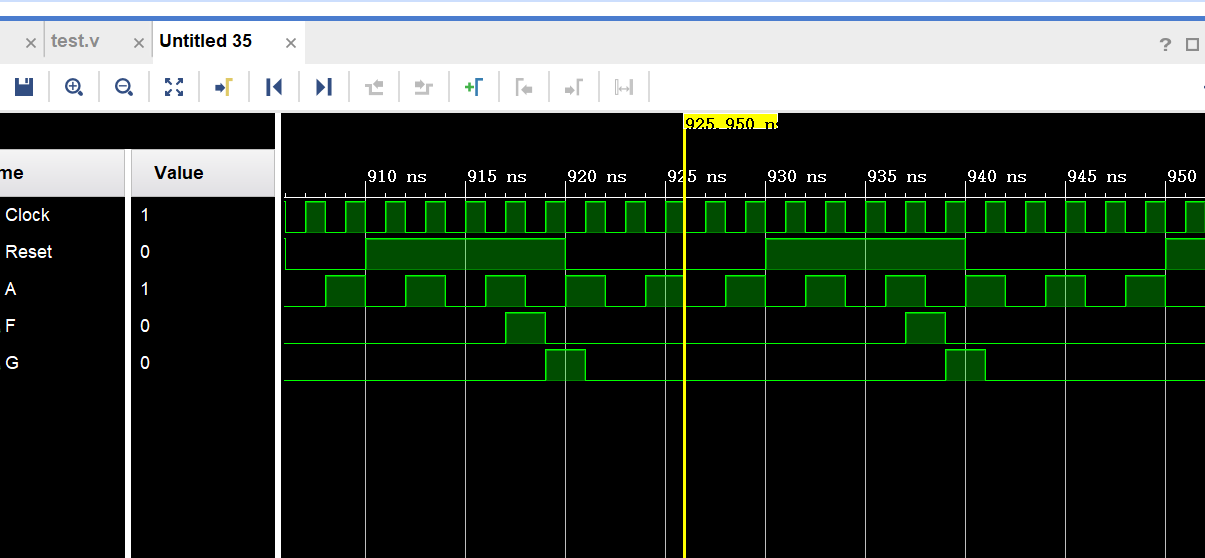
// 结束仿真

$finish;

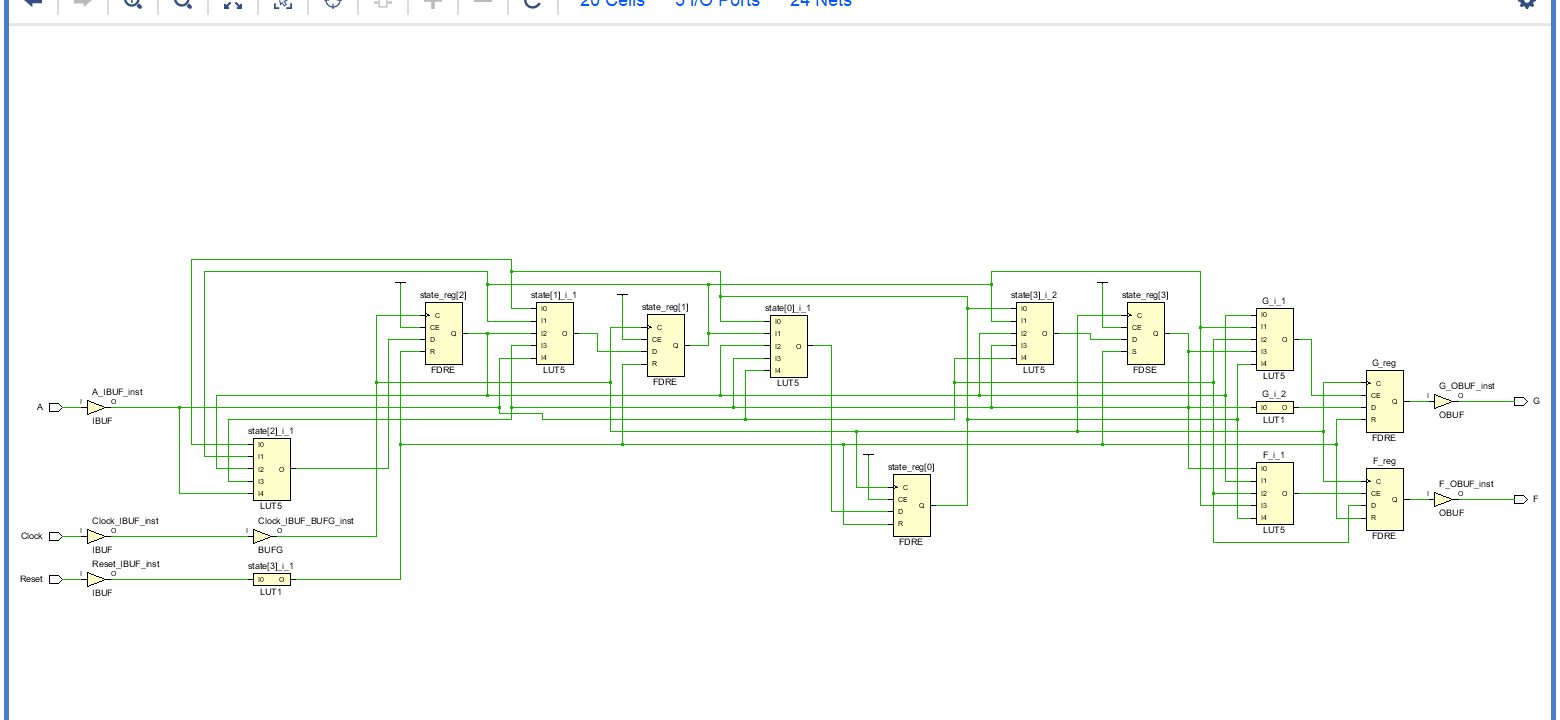
end

endmodule

**仿真结果**



**电路图**



**实验三 超前进位加法器的设计与仿真**

**激励文件**

`timescale 1ns / 1ps

module test\_ahead;

// Inputs

reg [3:0] A;

reg [3:0] B;

reg c\_in;

// Outputs

wire [3:0] F;

wire c\_out;

// Instantiate the Unit Under Test (UUT)

ahead\_adder uut (

.A(A),

.B(B),

.F(F),

.c\_in(c\_in),

.c\_out(c\_out)

);

initial begin

// Initialize Inputs

A = 0;

B = 0;

c\_in = 0;

// Wait 48 ns for global reset to finish 学号后两位

#48 A=4'b 1010;B=4'b 1001;c\_in=1;

#48 A=4'b 0010;B=4'b 1110;c\_in=0;

#48 A=4'b 0110;B=4'b 1101;c\_in=1;

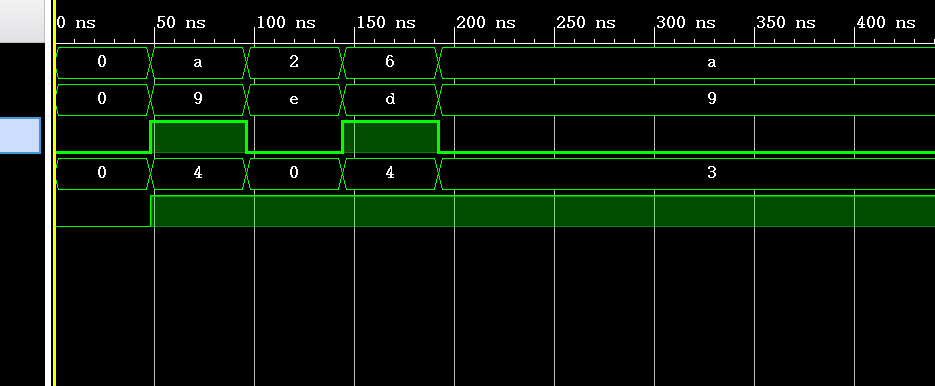
#48 A=4'b 1010;B=4'b 1001;c\_in=0;

// Add stimulus here

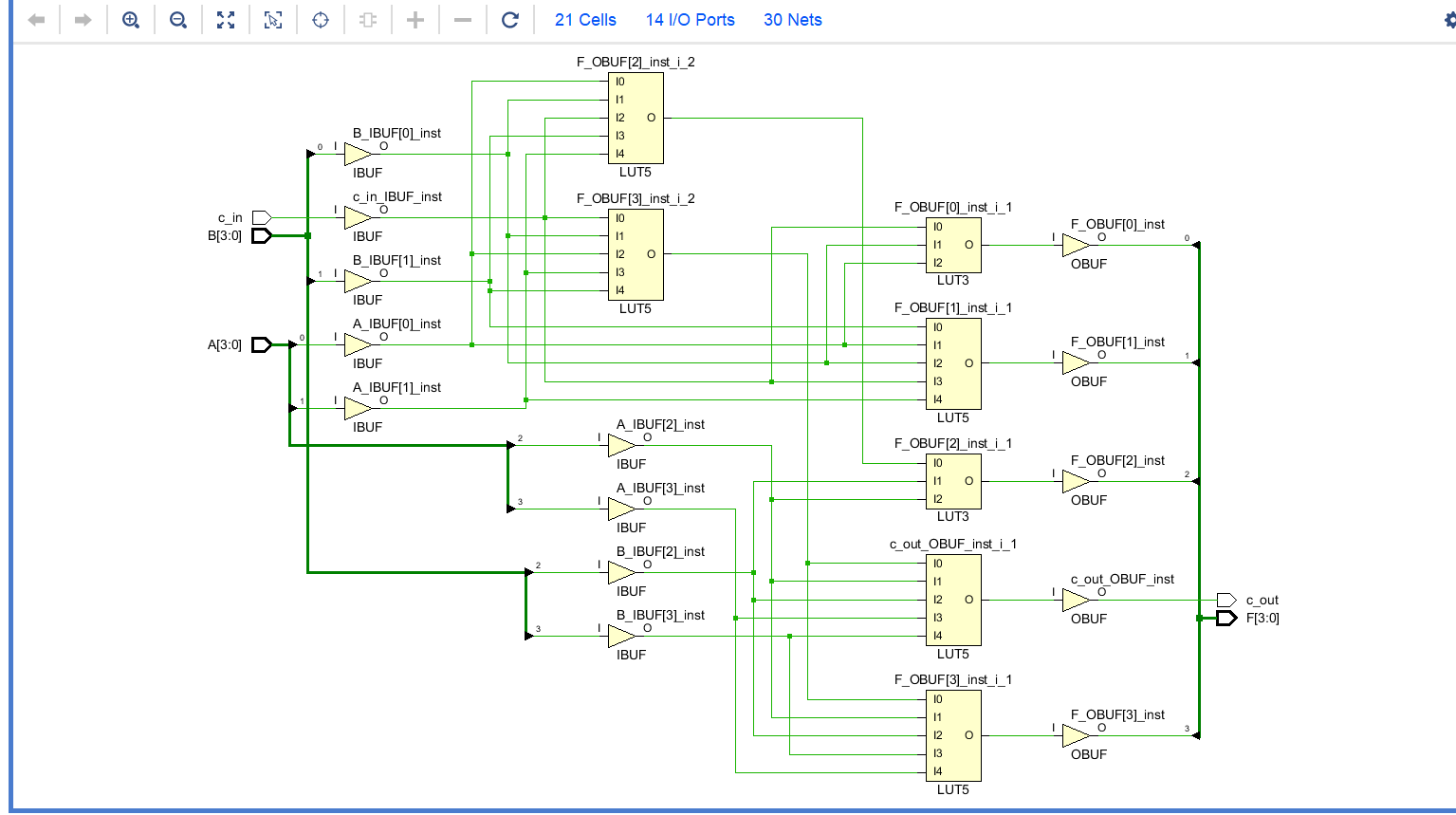
end

endmodule

**仿真结果**



**电路图**



**实验四 智力抢答器设计仿真与下载**

**激励文件**

`timescale 1ns / 1ps

//由于计数时间太长，很难看到计数，这里不再展示计数

module Smart\_responder\_tb;

reg clk,rst\_n;

reg [3:0]btn;

wire [3:0] an;

wire [7:0] seg\_code;

initial begin

clk = 1'b0;

rst\_n = 1'b0;

btn = 4'd0;

#10 rst\_n = 1'b1;

//

#48 rst\_n = 1'b0;//学号后两位

//有两个人按下按钮，但是时间不同

#10 btn = 4'd1;

#10 btn = 4'd3;

#10 btn = 4'd0;

#50 rst\_n = 1'b0;

#10 rst\_n = 1'b1;

//有两个人按下按钮，但是时间不同

#10 btn = 4'd2;

#10 btn = 4'd6;

#10 btn = 4'd0;

end

always #5 clk <= ~clk;

Smart\_responder Smart\_responder(

.clk( clk ),

.rst\_n( rst\_n ),

.btn( btn ),

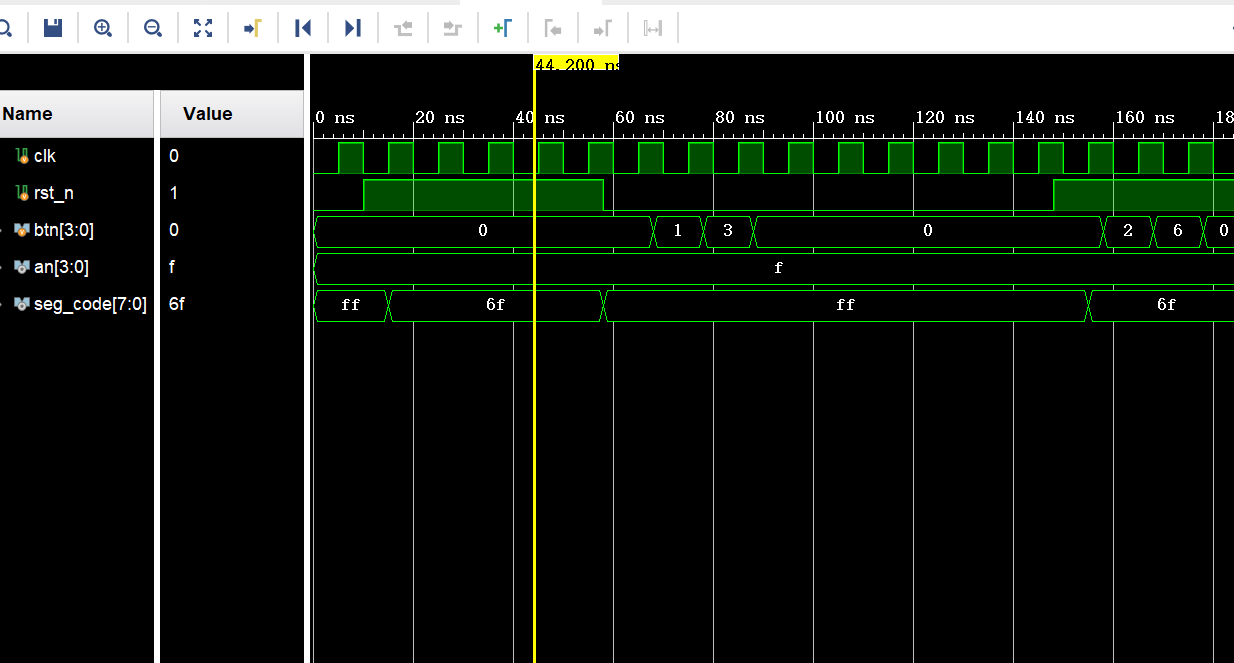
.an( an ),

.seg\_code( seg\_code )

);

endmodule

**仿真结果**



**电路图**

