Verilog 练习 1 基本逻辑门和组合电路的设计

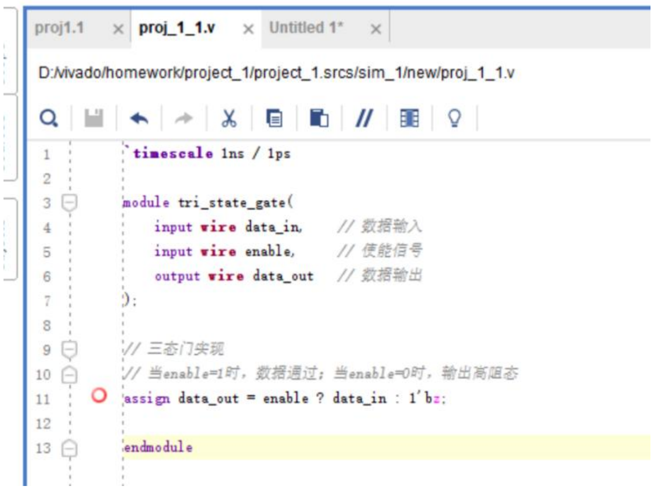
一．练习内容

1. 用 Verilog 完成三态门的设计，并用 Vivado 进行仿真。

2. 用 Verilog 完成 4 位加法器的设计，并使用 Vivado 进行仿真。

二．源程序及仿真结果截图

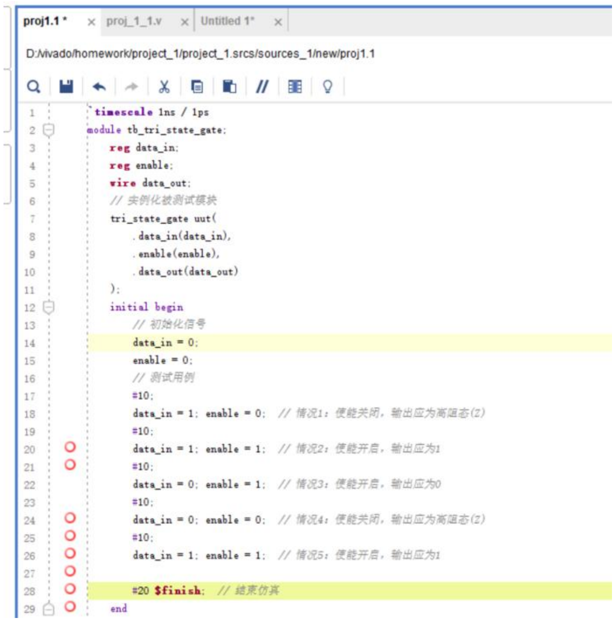
1.源程序：


proj1.1 × proj_1_1.v × Untitled 1* ×

D:/vivado/homework/project_1/project_1.srcs/sim_1/new/proj_1_1.v

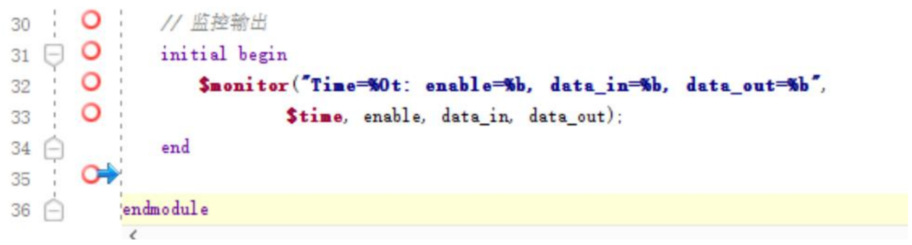
```verilog
`timescale 1ns / 1ps

module tri_state_gate(
    input wire data_in,     // 数据输入
    input wire enable,      // 使能信号
    output wire data_out    // 数据输出
);

// 三态门实现
// 当enable=1时，数据通过；当enable=0时，输出高阻态
assign data_out = enable ? data_in : 1'bz;

endmodule
```

测试代码：


proj1.1 * × proj_1_1.v × Untitled 1* ×

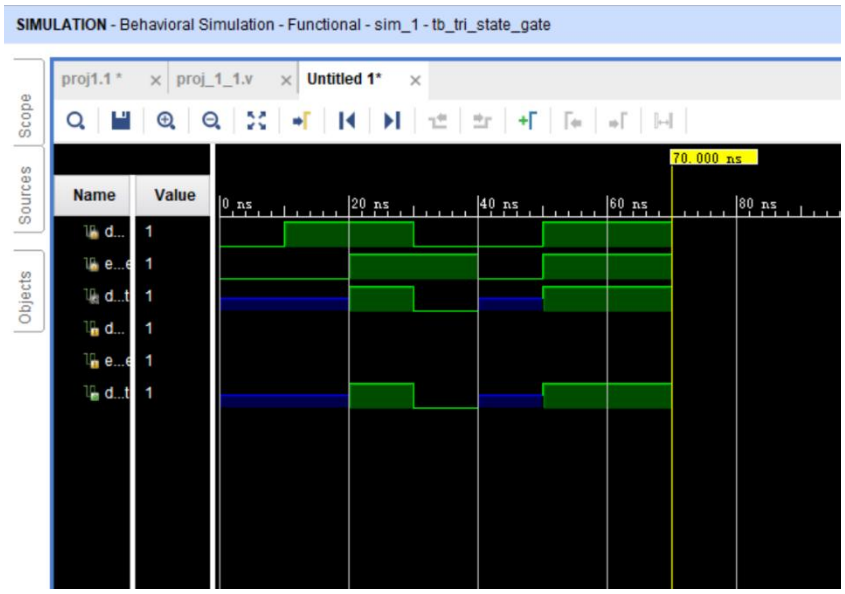
D:/vivado/homework/project_1/project_1.srcs/sources_1/new/proj1.1

```verilog
`timescale 1ns / 1ps
module tb_tri_state_gate;
    reg data_in;
    reg enable;
    wire data_out;
    // 实例化被测试模块
    tri_state_gate uut(
        .data_in(data_in),
        .enable(enable),
        .data_out(data_out)
    );
    initial begin
        // 初始化信号
        data_in = 0;
        enable = 0;
        // 测试用例
        #10;
        data_in = 1; enable = 0;  // 情况1: 使能关闭，输出应为高阻态(Z)
        #10;
        data_in = 1; enable = 1;  // 情况2: 使能开启，输出应为1
        #10;
        data_in = 0; enable = 1;  // 情况3: 使能开启，输出应为0
        #10;
        data_in = 0; enable = 0;  // 情况4: 使能关闭，输出应为高阻态(Z)
        #10;
        data_in = 1; enable = 1;  // 情况5: 使能开启，输出应为1

        #20 $finish;  // 结束仿真
    end
```



```
    // 监控输出
    initial begin
        $monitor("Time=%0t: enable=%b, data_in=%b, data_out=%b",
                 $time, enable, data_in, data_out);
    end

endmodule
```

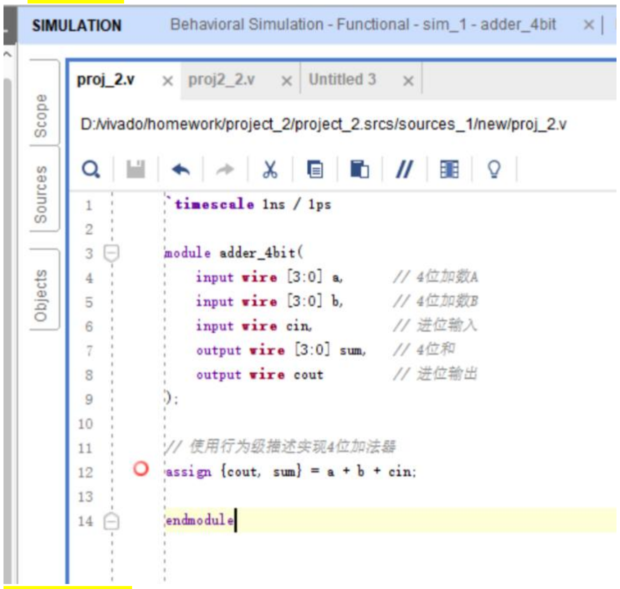
仿真结果：



SIMULATION - Behavioral Simulation - Functional - sim_1 - tb_tri_state_gate

```
Time=0: enable=0, data_in=0, data_out=z
Time=10000: enable=0, data_in=1, data_out=z
Time=20000: enable=1, data_in=1, data_out=1
Time=30000: enable=1, data_in=0, data_out=0
Time=40000: enable=0, data_in=0, data_out=z
Time=50000: enable=1, data_in=1, data_out=1
$finish called at time : 70 ns : File "D:/vivado/homework/project_1/project_1.srcs/sources_1/new/proj1.1" Line 35
```

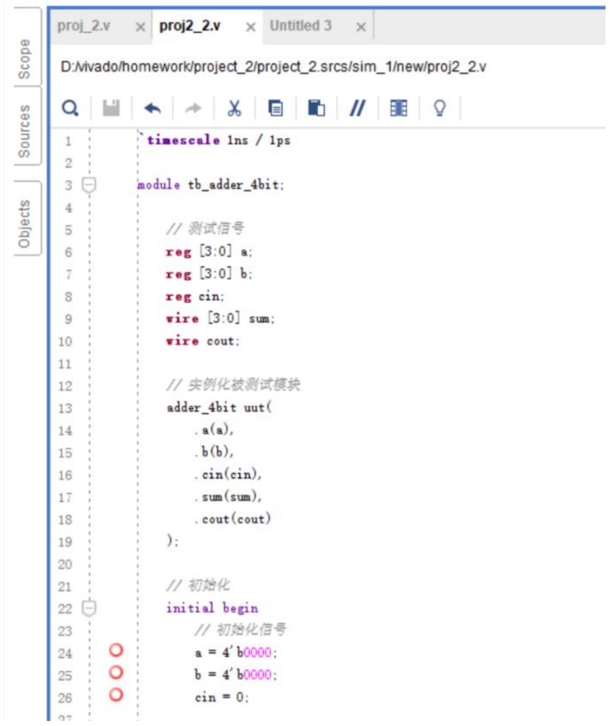
2.源程序



D:/vivado/homework/project_2/project_2.srcs/sources_1/new/proj_2.v

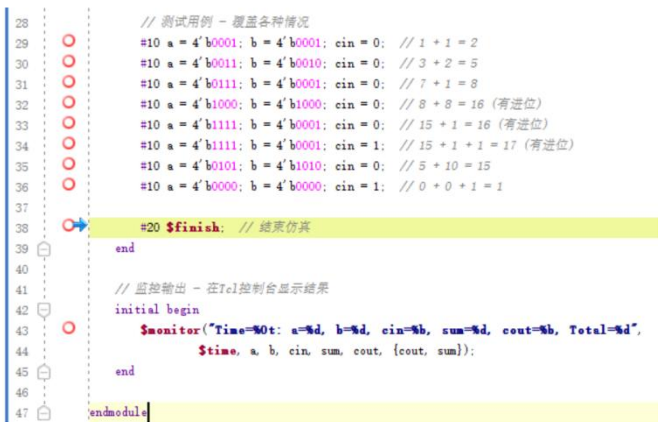
```verilog
`timescale 1ns / 1ps

module adder_4bit(
    input wire [3:0] a,      // 4位加数A
    input wire [3:0] b,      // 4位加数B
    input wire cin,          // 进位输入
    output wire [3:0] sum,   // 4位和
    output wire cout         // 进位输出
);

// 使用行为级描述实现4位加法器
assign {cout, sum} = a + b + cin;

endmodule
```

测试代码



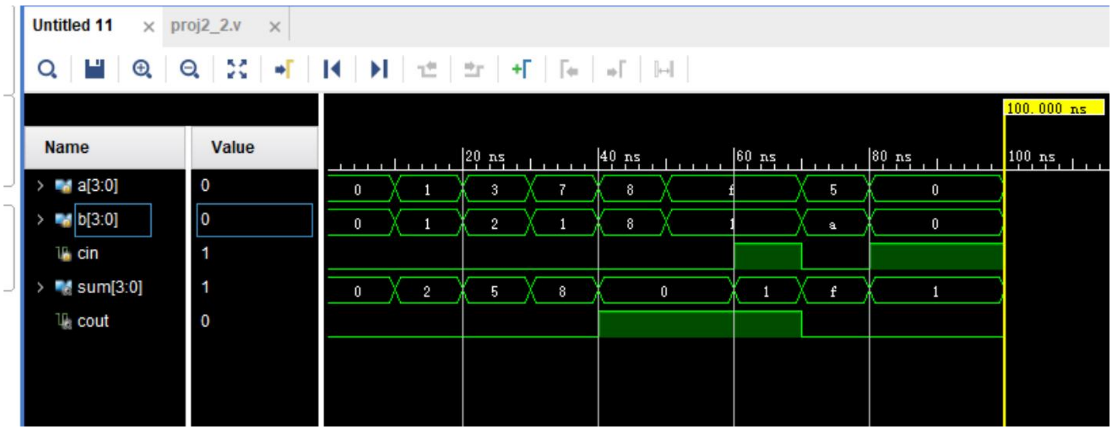
D:/vivado/homework/project_2/project_2.srcs/sim_1/new/proj2_2.v

```verilog
`timescale 1ns / 1ps

module tb_adder_4bit;

    // 测试信号
    reg [3:0] a;
    reg [3:0] b;
    reg cin;
    wire [3:0] sum;
    wire cout;

    // 实例化被测试模块
    adder_4bit uut(
        .a(a),
        .b(b),
        .cin(cin),
        .sum(sum),
        .cout(cout)
    );

    // 初始化
    initial begin
        // 初始化信号
        a = 4'b0000;
        b = 4'b0000;
        cin = 0;
```



```verilog
        // 测试用例 - 覆盖各种情况
        #10 a = 4'b0001; b = 4'b0001; cin = 0;  // 1 + 1 = 2
        #10 a = 4'b0011; b = 4'b0010; cin = 0;  // 3 + 2 = 5
        #10 a = 4'b0111; b = 4'b0001; cin = 0;  // 7 + 1 = 8
        #10 a = 4'b1000; b = 4'b1000; cin = 0;  // 8 + 8 = 16 (有进位)
        #10 a = 4'b1111; b = 4'b0001; cin = 0;  // 15 + 1 = 16 (有进位)
        #10 a = 4'b1111; b = 4'b0001; cin = 1;  // 15 + 1 + 1 = 17 (有进位)
        #10 a = 4'b0101; b = 4'b1010; cin = 0;  // 5 + 10 = 15
        #10 a = 4'b0000; b = 4'b0000; cin = 1;  // 0 + 0 + 1 = 1

        #20 $finish;  // 结束仿真
    end

    // 监控输出 - 在Tcl控制台显示结果
    initial begin
        $monitor("Time=%0t: a=%d, b=%d, cin=%b, sum=%d, cout=%b, Total=%d",
                 $time, a, b, cin, sum, cout, {cout, sum});
    end

endmodule
```

仿真结果



```
Time=0:  a= 0,  b= 0,  cin=0,  sum= 0,  cout=0,  Total= 0
Time=10000:  a= 1,  b= 1,  cin=0,  sum= 2,  cout=0,  Total= 2
Time=20000:  a= 3,  b= 2,  cin=0,  sum= 5,  cout=0,  Total= 5
Time=30000:  a= 7,  b= 1,  cin=0,  sum= 8,  cout=0,  Total= 8
Time=40000:  a= 8,  b= 8,  cin=0,  sum= 0,  cout=1,  Total=16
Time=50000:  a=15,  b= 1,  cin=0,  sum= 0,  cout=1,  Total=16
Time=60000:  a=15,  b= 1,  cin=1,  sum= 1,  cout=1,  Total=17
Time=70000:  a= 5,  b=10,  cin=0,  sum=15,  cout=0,  Total=15
Time=80000:  a= 0,  b= 0,  cin=1,  sum= 1,  cout=0,  Total= 1
$finish called at time : 100 ns : File "D:/vivado/homework/project_2/project_2.srcs/sim_1/new/proj2_2.v" Line 38
```

Verilog 练习 2 组合电路的设计

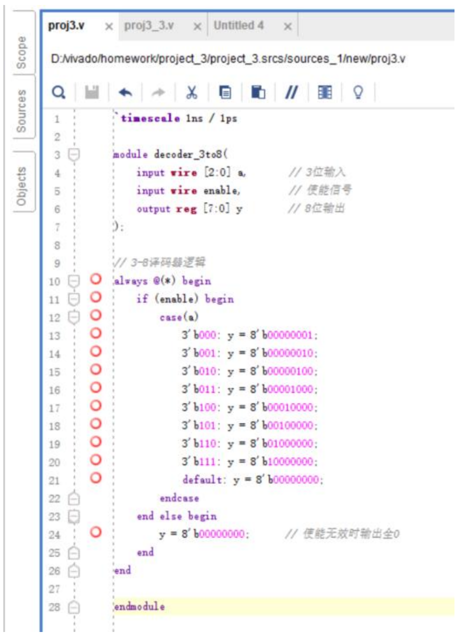
一．练习内容

1. 用 Verilog 完成 3-8 译码器的设计，并用 Vivado 进行仿真。

2. 用 Verilog 完成两个 4 位二进制数据比较器的设计，并使用 Vivado 进行仿真。

二．源程序及仿真结果截图

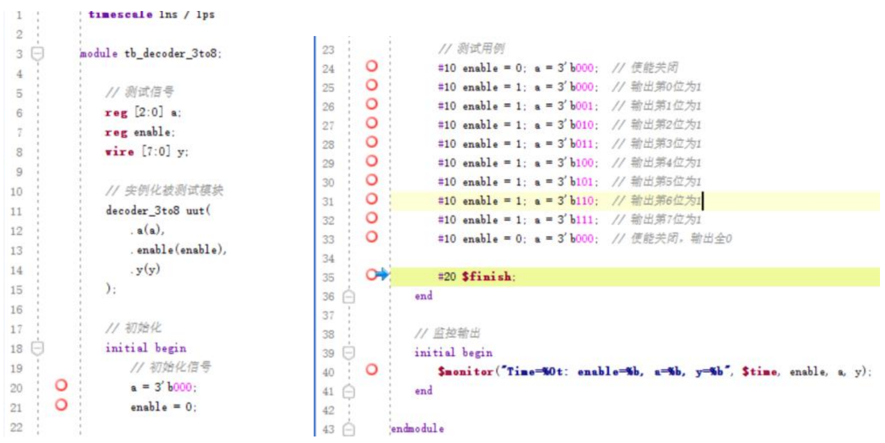
1.源程序



D:/vivado/homework/project_3/project_3.srcs/sources_1/new/proj3.v

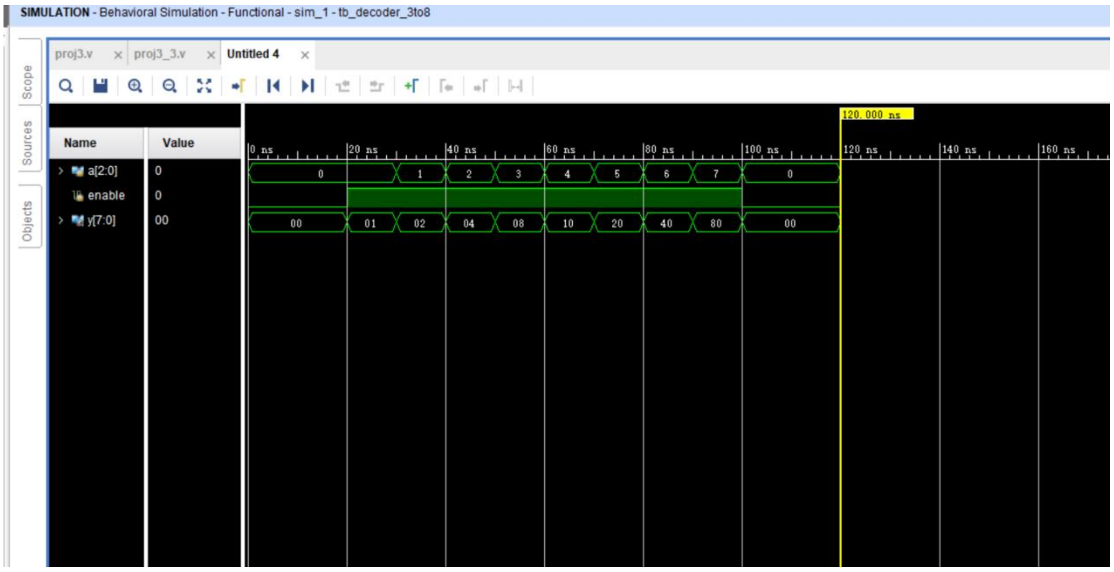
```verilog
`timescale 1ns / 1ps

module decoder_3to8(
    input wire [2:0] a,       // 3位输入
    input wire enable,        // 使能信号
    output reg [7:0] y        // 8位输出
);

// 3-8译码器逻辑
always @(*) begin
    if (enable) begin
        case(a)
            3'b000: y = 8'b00000001;
            3'b001: y = 8'b00000010;
            3'b010: y = 8'b00000100;
            3'b011: y = 8'b00001000;
            3'b100: y = 8'b00010000;
            3'b101: y = 8'b00100000;
            3'b110: y = 8'b01000000;
            3'b111: y = 8'b10000000;
            default: y = 8'b00000000;
        endcase
    end else begin
        y = 8'b00000000;       // 使能无效时输出全0
    end
end

endmodule
```

测试代码



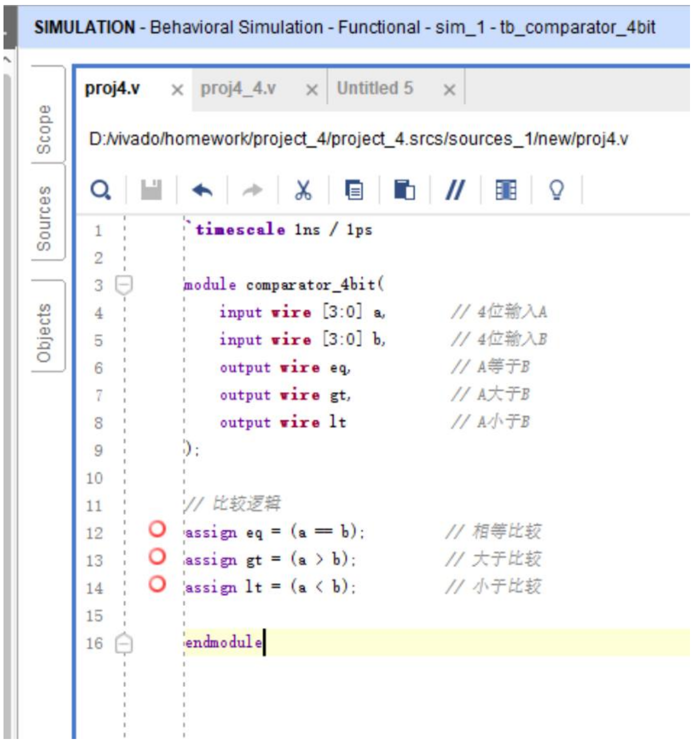
```verilog
`timescale 1ns / 1ps

module tb_decoder_3to8;

    // 测试信号
    reg [2:0] a;
    reg enable;
    wire [7:0] y;

    // 实例化被测试模块
    decoder_3to8 uut(
        .a(a),
        .enable(enable),
        .y(y)
    );

    // 初始化
    initial begin
        // 初始化信号
        a = 3'b000;
        enable = 0;

        // 测试用例
        #10 enable = 0; a = 3'b000;  // 使能关闭
        #10 enable = 1; a = 3'b000;  // 输出第0位为1
        #10 enable = 1; a = 3'b001;  // 输出第1位为1
        #10 enable = 1; a = 3'b010;  // 输出第2位为1
        #10 enable = 1; a = 3'b011;  // 输出第3位为1
        #10 enable = 1; a = 3'b100;  // 输出第4位为1
        #10 enable = 1; a = 3'b101;  // 输出第5位为1
        #10 enable = 1; a = 3'b110;  // 输出第6位为1
        #10 enable = 1; a = 3'b111;  // 输出第7位为1
        #10 enable = 0; a = 3'b000;  // 使能关闭，输出全0

        #20 $finish;
    end

    // 监控输出
    initial begin
        $monitor("Time=%0t: enable=%b, a=%b, y=%b", $time, enable, a, y);
    end

endmodule
```

仿真结果


SIMULATION - Behavioral Simulation - Functional - sim_1 - tb_decoder_3to8
proj3.v
proj3_3.v
Untitled 4
Name
Value
a[2:0]
enable
y[7:0]
120.000 ns

```
Time=0: enable=0, a=000, y=00000000
Time=20000: enable=1, a=000, y=00000001
Time=30000: enable=1, a=001, y=00000010
Time=40000: enable=1, a=010, y=00000100
Time=50000: enable=1, a=011, y=00001000
Time=60000: enable=1, a=100, y=00010000
Time=70000: enable=1, a=101, y=00100000
Time=80000: enable=1, a=110, y=01000000
Time=90000: enable=1, a=111, y=10000000
Time=100000: enable=0, a=000, y=00000000
$finish called at time : 120 ns : File "D:/vivado/homework/project_3/project_3.srcs/sim_1/new/proj3_3.v" Line 35
```

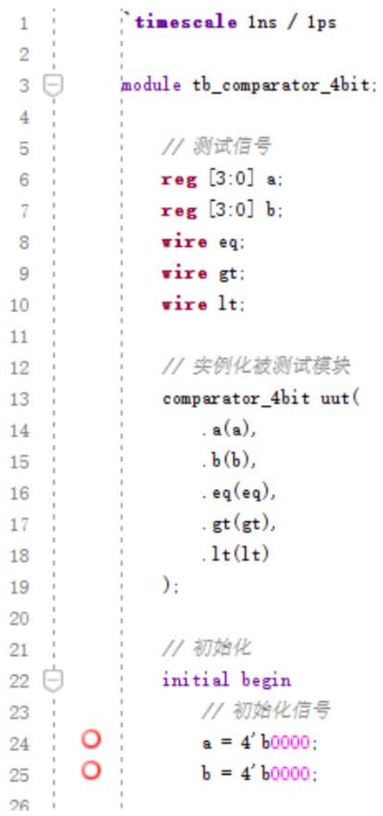
2.源代码

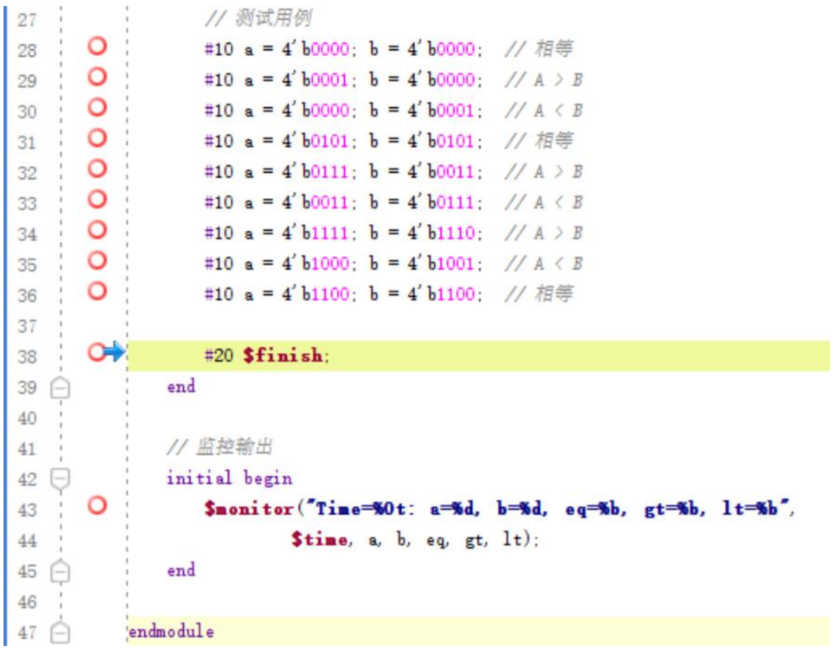

SIMULATION - Behavioral Simulation - Functional - sim_1 - tb_comparator_4bit

D:/vivado/homework/project_4/project_4.srcs/sources_1/new/proj4.v

```
`timescale 1ns / 1ps

module comparator_4bit(
    input wire [3:0] a,         // 4位输入A
    input wire [3:0] b,         // 4位输入B
    output wire eq,             // A等于B
    output wire gt,             // A大于B
    output wire lt              // A小于B
);

// 比较逻辑
assign eq = (a == b);          // 相等比较
assign gt = (a > b);           // 大于比较
assign lt = (a < b);           // 小于比较

endmodule
```

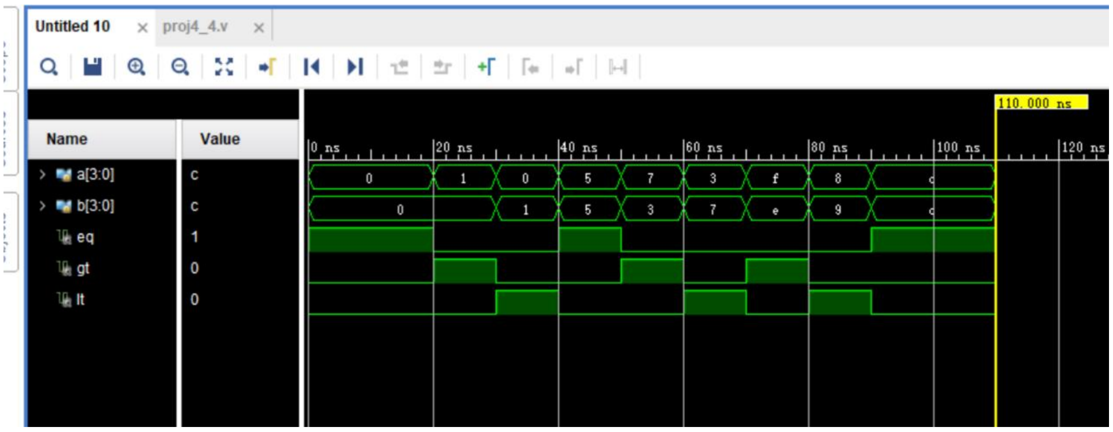
测试代码

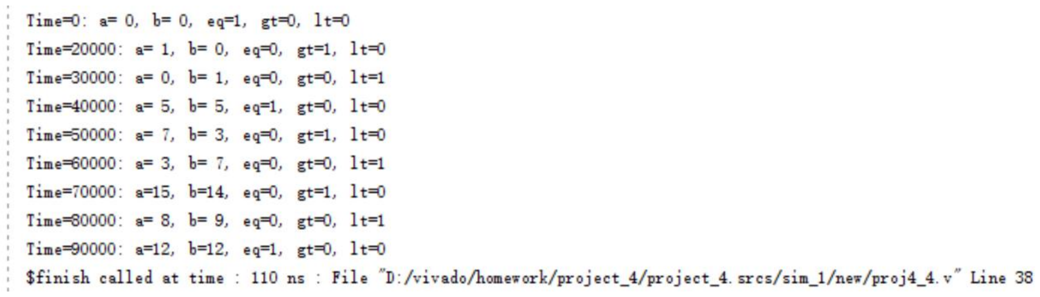

```
`timescale 1ns / 1ps

module tb_comparator_4bit;

    // 测试信号
    reg [3:0] a;
    reg [3:0] b;
    wire eq;
    wire gt;
    wire lt;

    // 实例化被测试模块
    comparator_4bit uut(
        .a(a),
        .b(b),
        .eq(eq),
        .gt(gt),
        .lt(lt)
    );

    // 初始化
    initial begin
        // 初始化信号
        a = 4'b0000;
        b = 4'b0000;
```



```verilog
        // 测试用例
        #10 a = 4'b0000; b = 4'b0000;  // 相等
        #10 a = 4'b0001; b = 4'b0000;  // A > B
        #10 a = 4'b0000; b = 4'b0001;  // A < B
        #10 a = 4'b0101; b = 4'b0101;  // 相等
        #10 a = 4'b0111; b = 4'b0011;  // A > B
        #10 a = 4'b0011; b = 4'b0111;  // A < B
        #10 a = 4'b1111; b = 4'b1110;  // A > B
        #10 a = 4'b1000; b = 4'b1001;  // A < B
        #10 a = 4'b1100; b = 4'b1100;  // 相等

        #20 $finish;
    end

    // 监控输出
    initial begin
        $monitor("Time=%0t: a=%d, b=%d, eq=%b, gt=%b, lt=%b",
                 $time, a, b, eq, gt, lt);
    end

endmodule
```

仿真结果





```
Time=0: a= 0, b= 0, eq=1, gt=0, lt=0
Time=20000: a= 1, b= 0, eq=0, gt=1, lt=0
Time=30000: a= 0, b= 1, eq=0, gt=0, lt=1
Time=40000: a= 5, b= 5, eq=1, gt=0, lt=0
Time=50000: a= 7, b= 3, eq=0, gt=1, lt=0
Time=60000: a= 3, b= 7, eq=0, gt=0, lt=1
Time=70000: a=15, b=14, eq=0, gt=1, lt=0
Time=80000: a= 8, b= 9, eq=0, gt=0, lt=1
Time=90000: a=12, b=12, eq=1, gt=0, lt=0
$finish called at time : 110 ns : File "D:/vivado/homework/project_4/project_4.srcs/sim_1/new/proj4_4.v" Line 38
```

Verilog 练习 3 触发器

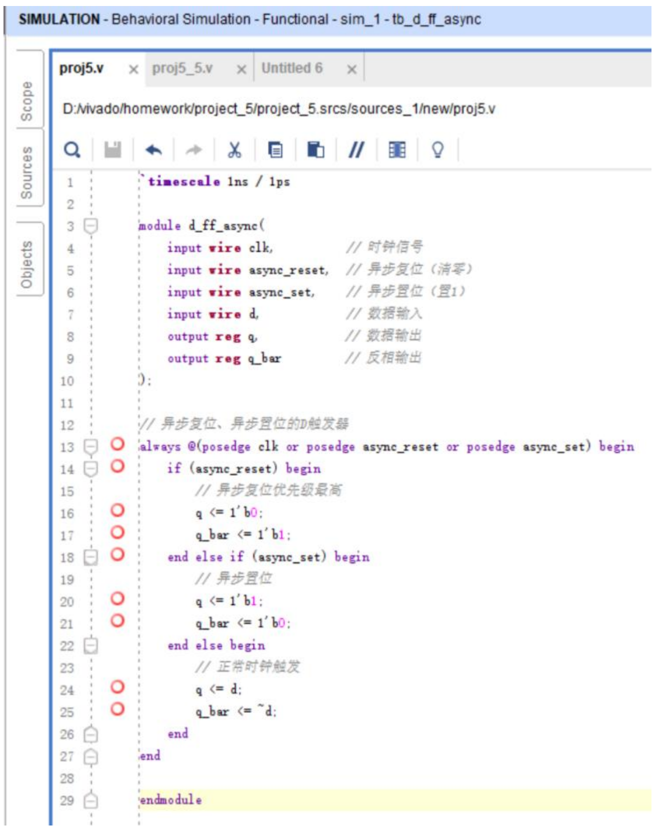
一．练习内容

1. 用 Verilog 完成具有异步清零和异步置 1 的 D 触发器的设计，并用 Vivado 进行仿真。

2. 用 Verilog 完成时钟下降沿触发的 JK 触发器的设计，使用 Vivado 进行仿真。

二．源程序及仿真结果截图

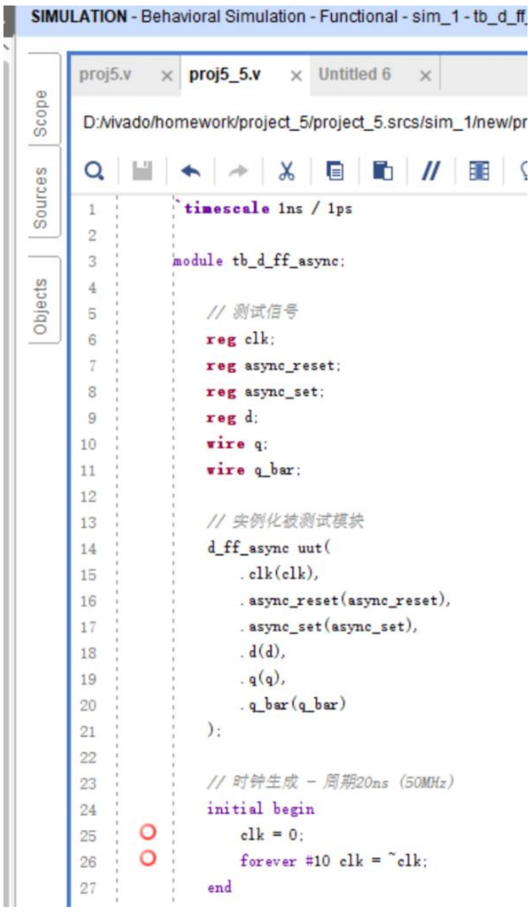
1.源程序


SIMULATION - Behavioral Simulation - Functional - sim_1 - tb_d_ff_async
proj5.v
proj5_5.v
Untitled 6
Scope
Sources
Objects

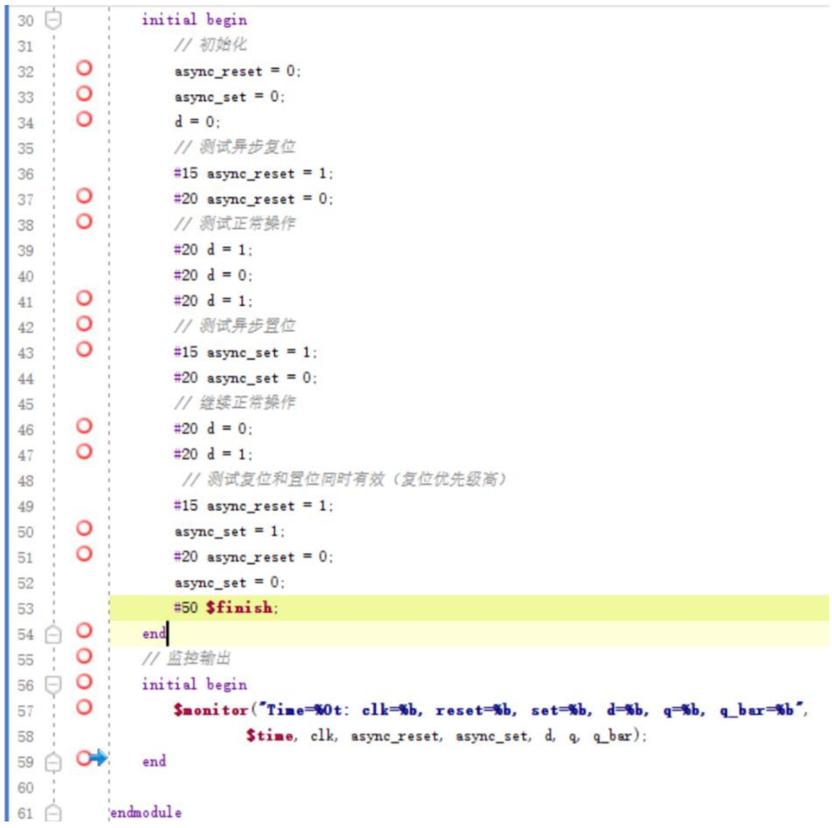
D:/vivado/homework/project_5/project_5.srcs/sources_1/new/proj5.v

```verilog
`timescale 1ns / 1ps

module d_ff_async(
    input wire clk,          // 时钟信号
    input wire async_reset,  // 异步复位（清零）
    input wire async_set,    // 异步置位（置1）
    input wire d,            // 数据输入
    output reg q,            // 数据输出
    output reg q_bar         // 反相输出
);

// 异步复位、异步置位的D触发器
always @(posedge clk or posedge async_reset or posedge async_set) begin
    if (async_reset) begin
        // 异步复位优先级最高
        q <= 1'b0;
        q_bar <= 1'b1;
    end else if (async_set) begin
        // 异步置位
        q <= 1'b1;
        q_bar <= 1'b0;
    end else begin
        // 正常时钟触发
        q <= d;
        q_bar <= ~d;
    end
end

endmodule
```

测试代码

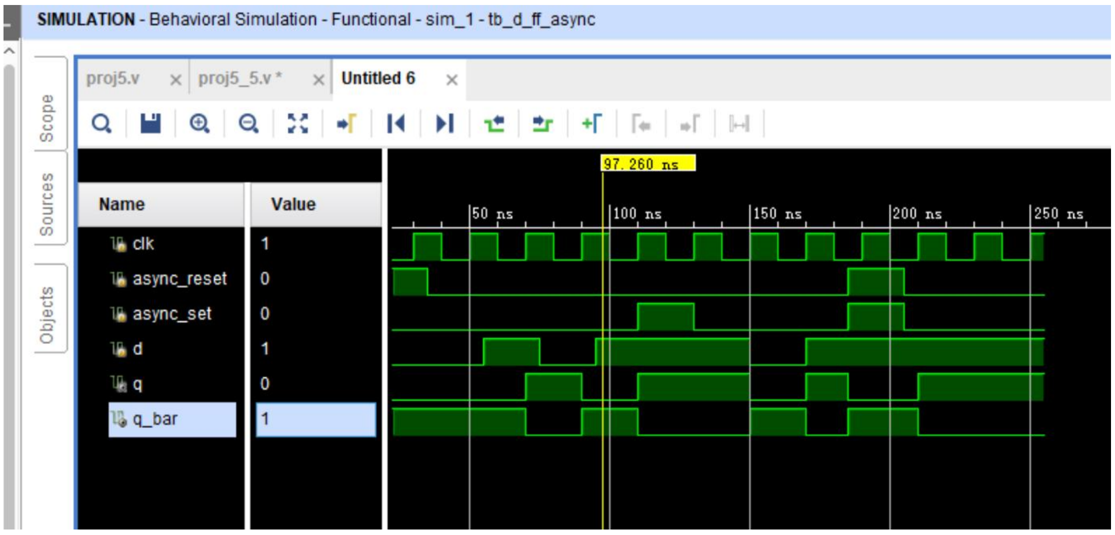

SIMULATION - Behavioral Simulation - Functional - sim_1 - tb_d_ff_
proj5.v
proj5_5.v
Untitled 6
Scope
Sources
Objects
D:/vivado/homework/project_5/project_5.srcs/sim_1/new/pr

```verilog
`timescale 1ns / 1ps

module tb_d_ff_async;

    // 测试信号
    reg clk;
    reg async_reset;
    reg async_set;
    reg d;
    wire q;
    wire q_bar;

    // 实例化被测试模块
    d_ff_async uut(
        .clk(clk),
        .async_reset(async_reset),
        .async_set(async_set),
        .d(d),
        .q(q),
        .q_bar(q_bar)
    );

    // 时钟生成 - 周期20ns (50MHz)
    initial begin
        clk = 0;
        forever #10 clk = ~clk;
    end
```

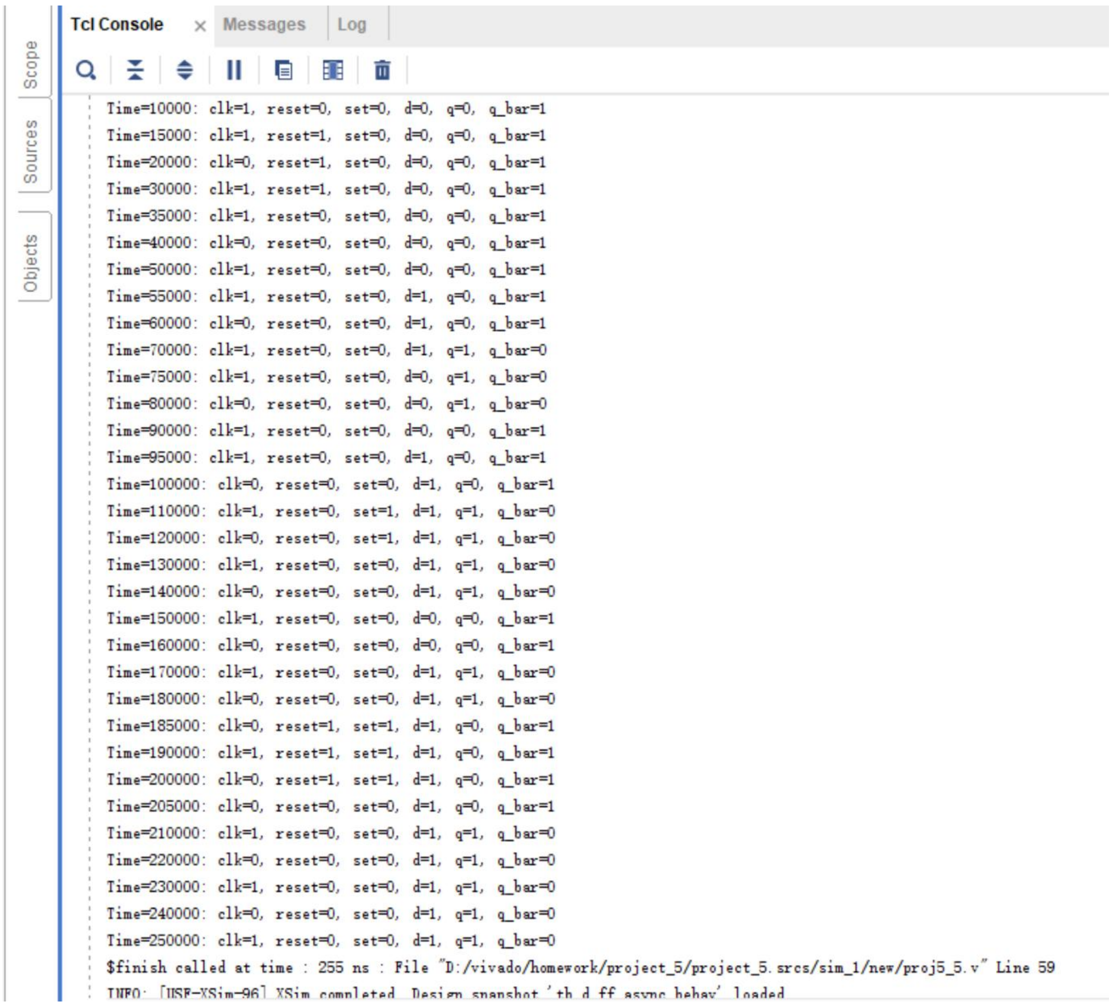


```verilog
    initial begin
        // 初始化
        async_reset = 0;
        async_set = 0;
        d = 0;
        // 测试异步复位
        #15 async_reset = 1;
        #20 async_reset = 0;
        // 测试正常操作
        #20 d = 1;
        #20 d = 0;
        #20 d = 1;
        // 测试异步置位
        #15 async_set = 1;
        #20 async_set = 0;
        // 继续正常操作
        #20 d = 0;
        #20 d = 1;
         // 测试复位和置位同时有效（复位优先级高）
        #15 async_reset = 1;
        async_set = 1;
        #20 async_reset = 0;
        async_set = 0;
        #50 $finish;
    end
    // 监控输出
    initial begin
        $monitor("Time=%0t: clk=%b, reset=%b, set=%b, d=%b, q=%b, q_bar=%b",
                 $time, clk, async_reset, async_set, d, q, q_bar);
    end

endmodule
```

仿真结果

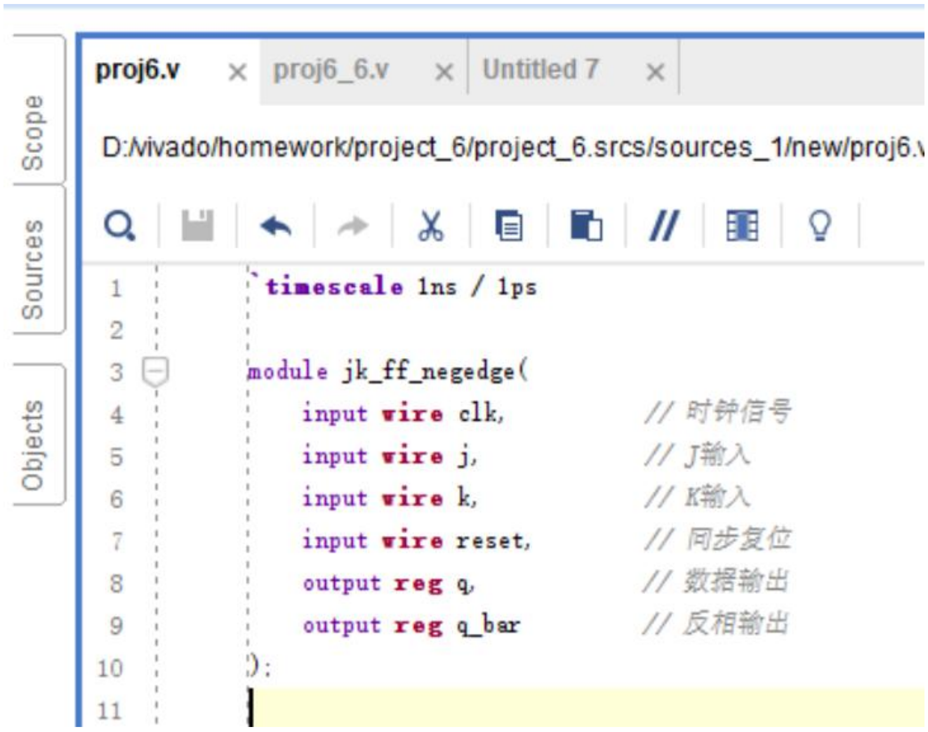


Name	Value
clk	1
async_reset	0
async_set	0
d	1
q	0
q_bar	1



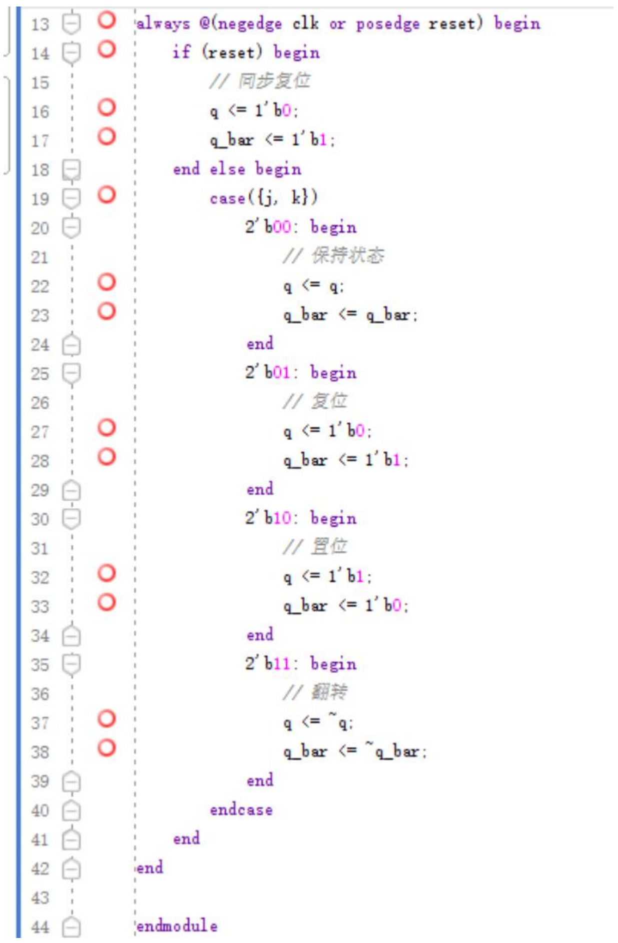
```
Time=10000: clk=1, reset=0, set=0, d=0, q=0, q_bar=1
Time=15000: clk=1, reset=1, set=0, d=0, q=0, q_bar=1
Time=20000: clk=0, reset=1, set=0, d=0, q=0, q_bar=1
Time=30000: clk=1, reset=1, set=0, d=0, q=0, q_bar=1
Time=35000: clk=1, reset=0, set=0, d=0, q=0, q_bar=1
Time=40000: clk=0, reset=0, set=0, d=0, q=0, q_bar=1
Time=50000: clk=1, reset=0, set=0, d=0, q=0, q_bar=1
Time=55000: clk=1, reset=0, set=0, d=1, q=0, q_bar=1
Time=60000: clk=0, reset=0, set=0, d=1, q=0, q_bar=1
Time=70000: clk=1, reset=0, set=0, d=1, q=1, q_bar=0
Time=75000: clk=1, reset=0, set=0, d=0, q=1, q_bar=0
Time=80000: clk=0, reset=0, set=0, d=0, q=1, q_bar=0
Time=90000: clk=1, reset=0, set=0, d=0, q=0, q_bar=1
Time=95000: clk=1, reset=0, set=0, d=1, q=0, q_bar=1
Time=100000: clk=0, reset=0, set=0, d=1, q=0, q_bar=1
Time=110000: clk=1, reset=0, set=1, d=1, q=1, q_bar=0
Time=120000: clk=0, reset=0, set=1, d=1, q=1, q_bar=0
Time=130000: clk=1, reset=0, set=0, d=1, q=1, q_bar=0
Time=140000: clk=0, reset=0, set=0, d=1, q=1, q_bar=0
Time=150000: clk=1, reset=0, set=0, d=0, q=0, q_bar=1
Time=160000: clk=0, reset=0, set=0, d=0, q=0, q_bar=1
Time=170000: clk=1, reset=0, set=0, d=1, q=1, q_bar=0
Time=180000: clk=0, reset=0, set=0, d=1, q=1, q_bar=0
Time=185000: clk=0, reset=1, set=1, d=1, q=0, q_bar=1
Time=190000: clk=1, reset=1, set=1, d=1, q=0, q_bar=1
Time=200000: clk=0, reset=1, set=1, d=1, q=0, q_bar=1
Time=205000: clk=0, reset=0, set=0, d=1, q=0, q_bar=1
Time=210000: clk=1, reset=0, set=0, d=1, q=1, q_bar=0
Time=220000: clk=0, reset=0, set=0, d=1, q=1, q_bar=0
Time=230000: clk=1, reset=0, set=0, d=1, q=1, q_bar=0
Time=240000: clk=0, reset=0, set=0, d=1, q=1, q_bar=0
Time=250000: clk=1, reset=0, set=0, d=1, q=1, q_bar=0
$finish called at time : 255 ns : File "D:/vivado/homework/project_5/project_5.srcs/sim_1/new/proj5_5.v" Line 59
INFO: [USF-XSim-96] XSim completed. Design snapshot 'tb_d_ff_async_behav' loaded.
```

2.源程序



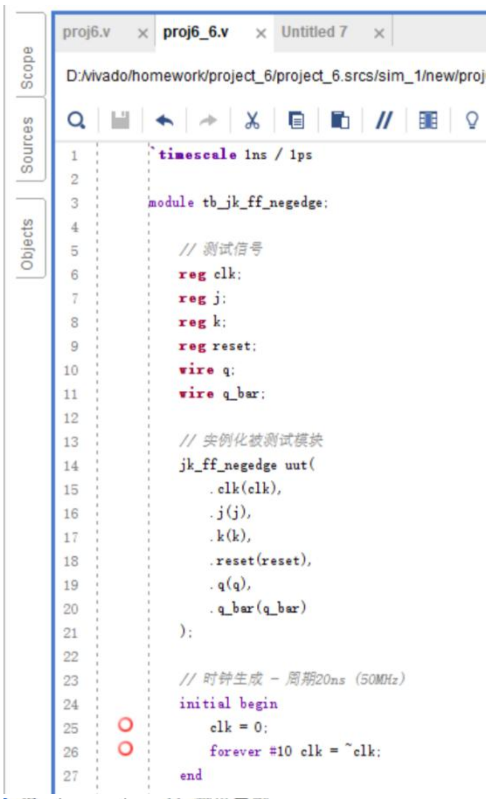
D:/vivado/homework/project_6/project_6.srcs/sources_1/new/proj6.v

```verilog
`timescale 1ns / 1ps

module jk_ff_negedge(
    input wire clk,          // 时钟信号
    input wire j,            // J输入
    input wire k,            // K输入
    input wire reset,        // 同步复位
    output reg q,            // 数据输出
    output reg q_bar         // 反相输出
);

```



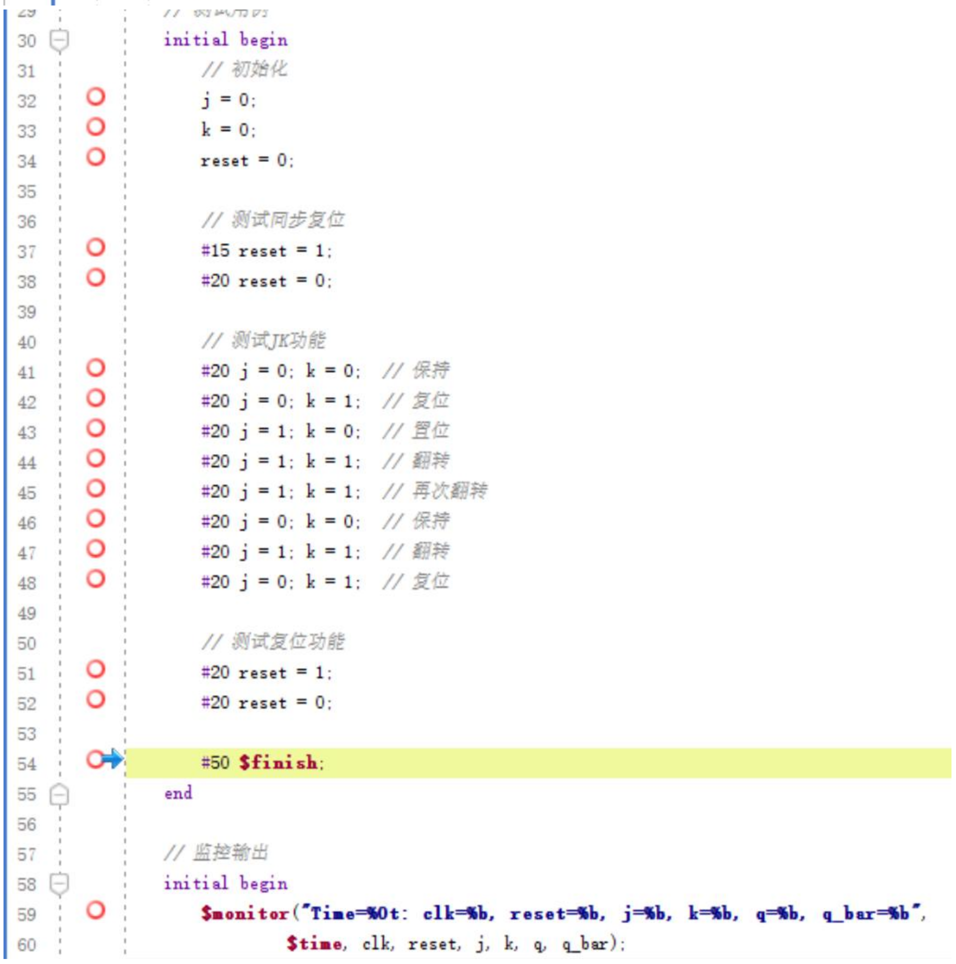
```verilog
always @(negedge clk or posedge reset) begin
    if (reset) begin
        // 同步复位
        q <= 1'b0;
        q_bar <= 1'b1;
    end else begin
        case({j, k})
            2'b00: begin
                // 保持状态
                q <= q;
                q_bar <= q_bar;
            end
            2'b01: begin
                // 复位
                q <= 1'b0;
                q_bar <= 1'b1;
            end
            2'b10: begin
                // 置位
                q <= 1'b1;
                q_bar <= 1'b0;
            end
            2'b11: begin
                // 翻转
                q <= ~q;
                q_bar <= ~q_bar;
            end
        endcase
    end
end

endmodule
```

测试代码



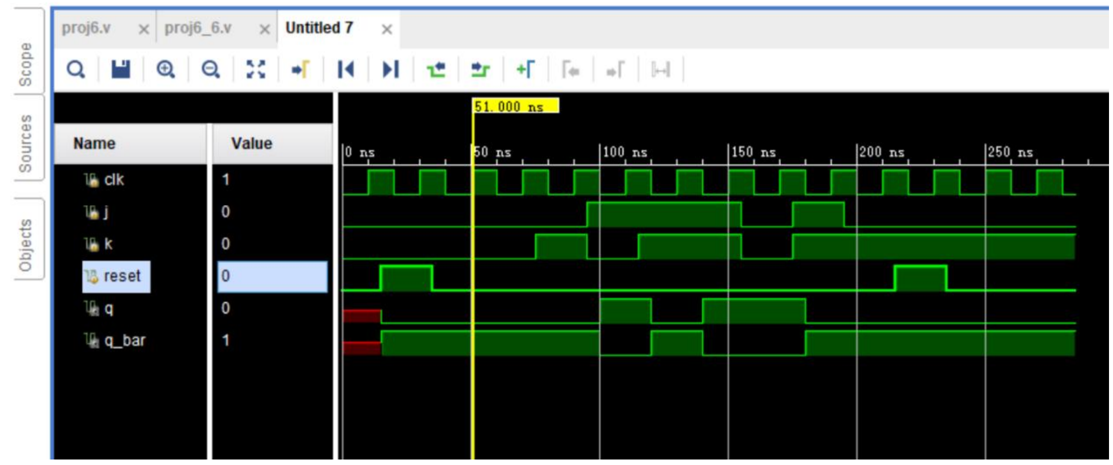
D:/vivado/homework/project_6/project_6.srcs/sim_1/new/proj

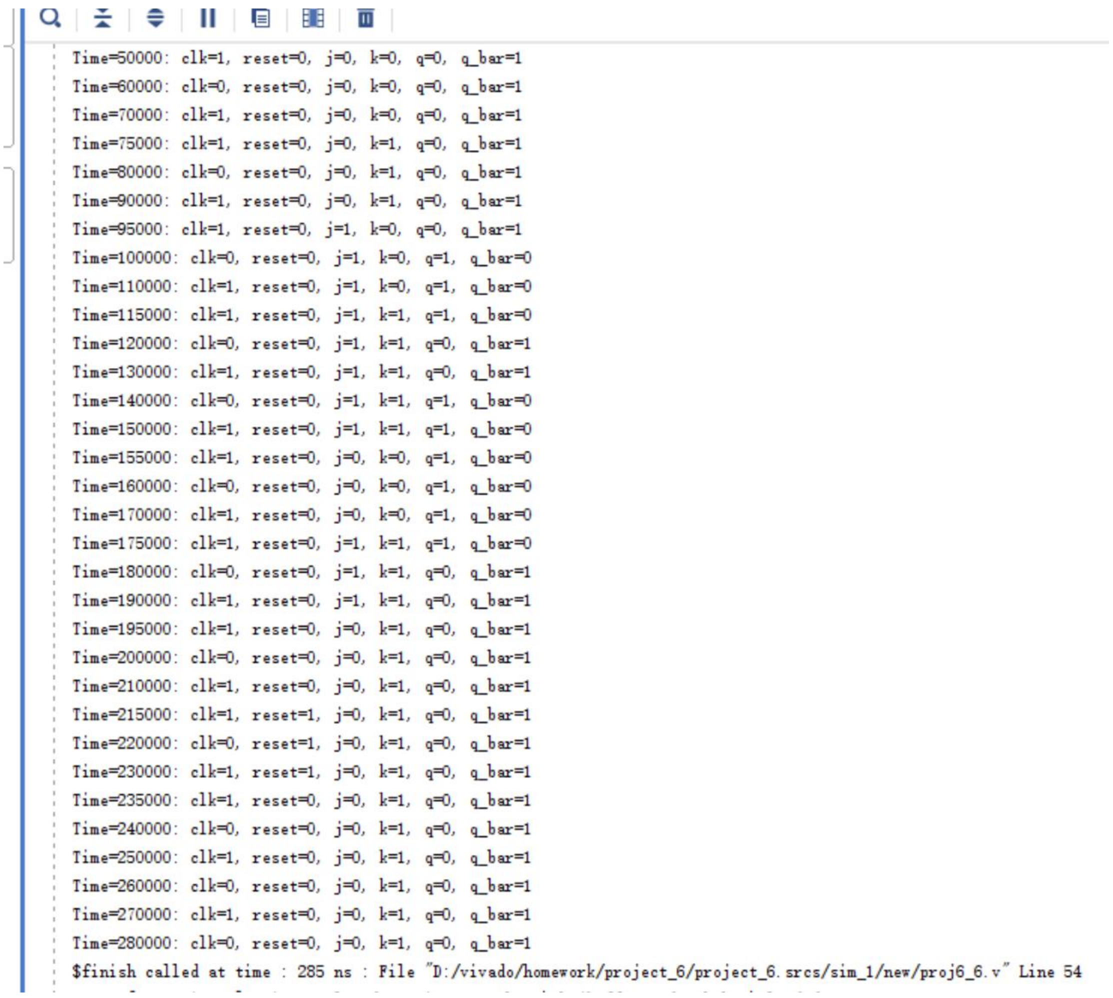
```verilog
`timescale 1ns / 1ps

module tb_jk_ff_negedge;

    // 测试信号
    reg clk;
    reg j;
    reg k;
    reg reset;
    wire q;
    wire q_bar;

    // 实例化被测试模块
    jk_ff_negedge uut(
        .clk(clk),
        .j(j),
        .k(k),
        .reset(reset),
        .q(q),
        .q_bar(q_bar)
    );

    // 时钟生成 - 周期20ns (50MHz)
    initial begin
        clk = 0;
        forever #10 clk = ~clk;
    end
```



```verilog
    initial begin
        // 初始化
        j = 0;
        k = 0;
        reset = 0;

        // 测试同步复位
        #15 reset = 1;
        #20 reset = 0;

        // 测试JK功能
        #20 j = 0; k = 0;  // 保持
        #20 j = 0; k = 1;  // 复位
        #20 j = 1; k = 0;  // 置位
        #20 j = 1; k = 1;  // 翻转
        #20 j = 1; k = 1;  // 再次翻转
        #20 j = 0; k = 0;  // 保持
        #20 j = 1; k = 1;  // 翻转
        #20 j = 0; k = 1;  // 复位

        // 测试复位功能
        #20 reset = 1;
        #20 reset = 0;

        #50 $finish;
    end

    // 监控输出
    initial begin
        $monitor("Time=%0t: clk=%b, reset=%b, j=%b, k=%b, q=%b, q_bar=%b",
                 $time, clk, reset, j, k, q, q_bar);
```

仿真结果





```
Time=50000: clk=1, reset=0, j=0, k=0, q=0, q_bar=1
Time=60000: clk=0, reset=0, j=0, k=0, q=0, q_bar=1
Time=70000: clk=1, reset=0, j=0, k=0, q=0, q_bar=1
Time=75000: clk=1, reset=0, j=0, k=1, q=0, q_bar=1
Time=80000: clk=0, reset=0, j=0, k=1, q=0, q_bar=1
Time=90000: clk=1, reset=0, j=0, k=1, q=0, q_bar=1
Time=95000: clk=1, reset=0, j=1, k=0, q=0, q_bar=1
Time=100000: clk=0, reset=0, j=1, k=0, q=1, q_bar=0
Time=110000: clk=1, reset=0, j=1, k=0, q=1, q_bar=0
Time=115000: clk=1, reset=0, j=1, k=1, q=1, q_bar=0
Time=120000: clk=0, reset=0, j=1, k=1, q=0, q_bar=1
Time=130000: clk=1, reset=0, j=1, k=1, q=0, q_bar=1
Time=140000: clk=0, reset=0, j=1, k=1, q=1, q_bar=0
Time=150000: clk=1, reset=0, j=1, k=1, q=1, q_bar=0
Time=155000: clk=1, reset=0, j=0, k=0, q=1, q_bar=0
Time=160000: clk=0, reset=0, j=0, k=0, q=1, q_bar=0
Time=170000: clk=1, reset=0, j=0, k=0, q=1, q_bar=0
Time=175000: clk=1, reset=0, j=1, k=1, q=1, q_bar=0
Time=180000: clk=0, reset=0, j=1, k=1, q=0, q_bar=1
Time=190000: clk=1, reset=0, j=1, k=1, q=0, q_bar=1
Time=195000: clk=1, reset=0, j=0, k=1, q=0, q_bar=1
Time=200000: clk=0, reset=0, j=0, k=1, q=0, q_bar=1
Time=210000: clk=1, reset=0, j=0, k=1, q=0, q_bar=1
Time=215000: clk=1, reset=1, j=0, k=1, q=0, q_bar=1
Time=220000: clk=0, reset=1, j=0, k=1, q=0, q_bar=1
Time=230000: clk=1, reset=1, j=0, k=1, q=0, q_bar=1
Time=235000: clk=1, reset=0, j=0, k=1, q=0, q_bar=1
Time=240000: clk=0, reset=0, j=0, k=1, q=0, q_bar=1
Time=250000: clk=1, reset=0, j=0, k=1, q=0, q_bar=1
Time=260000: clk=0, reset=0, j=0, k=1, q=0, q_bar=1
Time=270000: clk=1, reset=0, j=0, k=1, q=0, q_bar=1
Time=280000: clk=0, reset=0, j=0, k=1, q=0, q_bar=1
$finish called at time : 285 ns : File "D:/vivado/homework/project_6/project_6.srcs/sim_1/new/proj6_6.v" Line 54
```

Verilog 练习 4 计数器

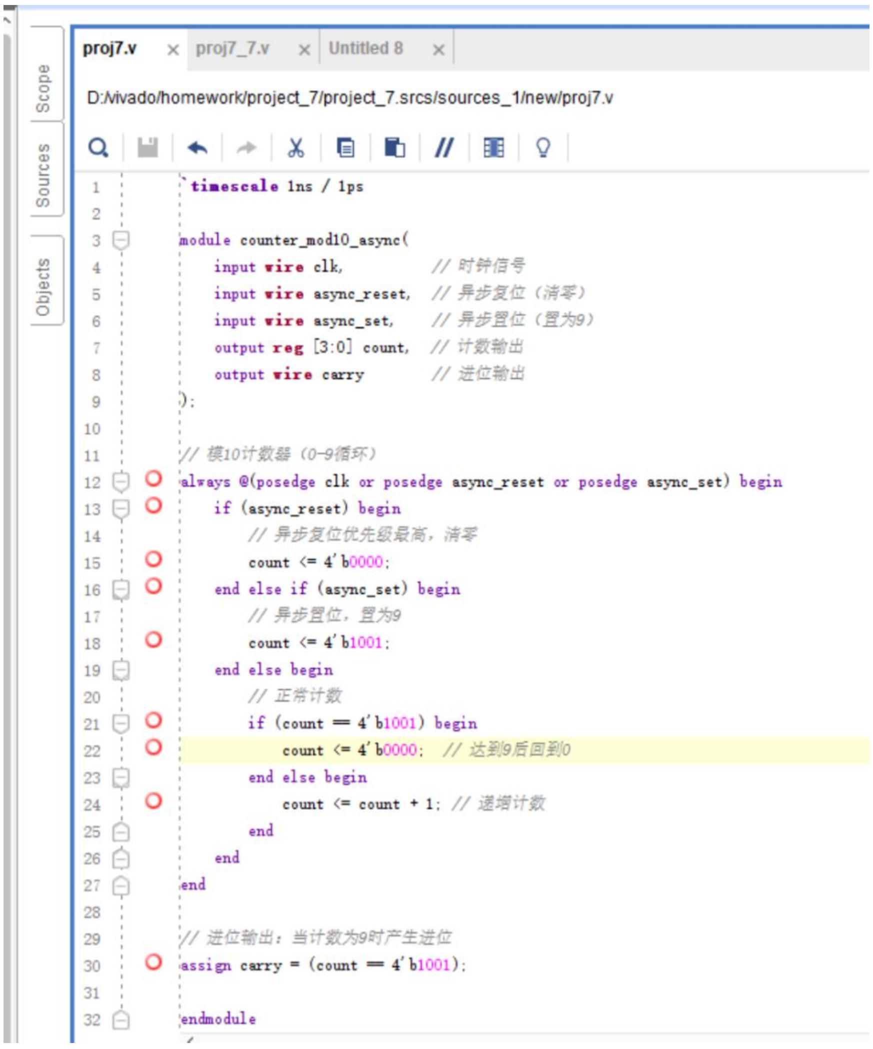
一．练习内容

1. 用 Verilog 完成具有异步清零和异步置 1 的模 10 计数器的设计，并用 Vivado 进行仿真。

2. 用 Verilog 完成具有同步清零和同步步置 1 的模 16 计数器的设计，使用 Vivado 进行仿真。

二．源程序及仿真结果截图

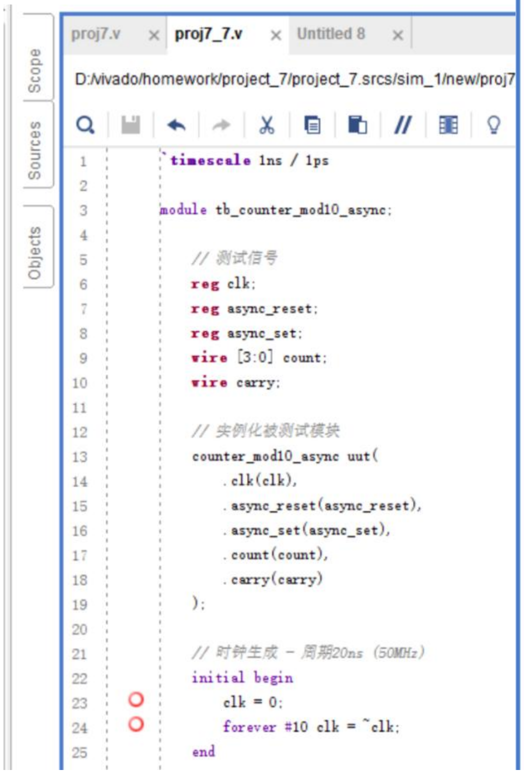
1.源程序



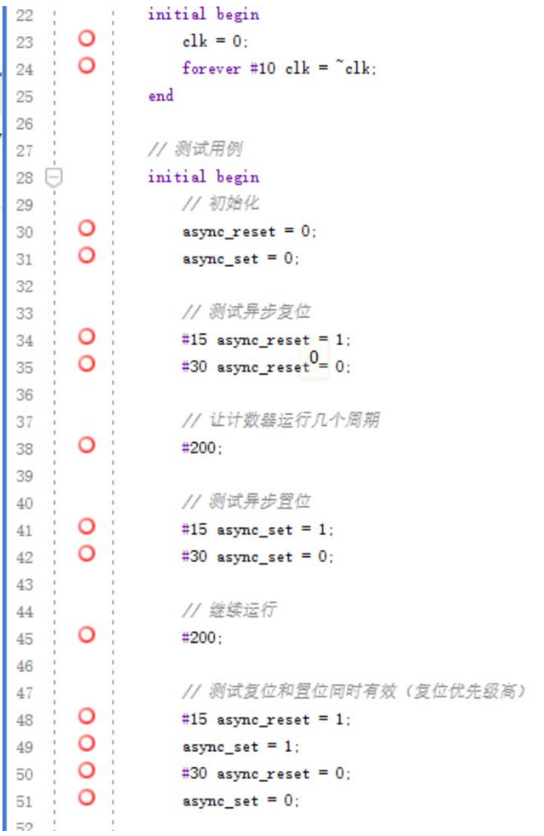
D:/vivado/homework/project_7/project_7.srcs/sources_1/new/proj7.v

```verilog
`timescale 1ns / 1ps

module counter_mod10_async(
    input wire clk,           // 时钟信号
    input wire async_reset,   // 异步复位（清零）
    input wire async_set,     // 异步置位（置为9）
    output reg [3:0] count,   // 计数输出
    output wire carry         // 进位输出
);

// 模10计数器（0-9循环）
always @(posedge clk or posedge async_reset or posedge async_set) begin
    if (async_reset) begin
        // 异步复位优先级最高，清零
        count <= 4'b0000;
    end else if (async_set) begin
        // 异步置位，置为9
        count <= 4'b1001;
    end else begin
        // 正常计数
        if (count == 4'b1001) begin
            count <= 4'b0000;  // 达到9后回到0
        end else begin
            count <= count + 1; // 递增计数
        end
    end
end

// 进位输出：当计数为9时产生进位
assign carry = (count == 4'b1001);

endmodule
```

测试代码

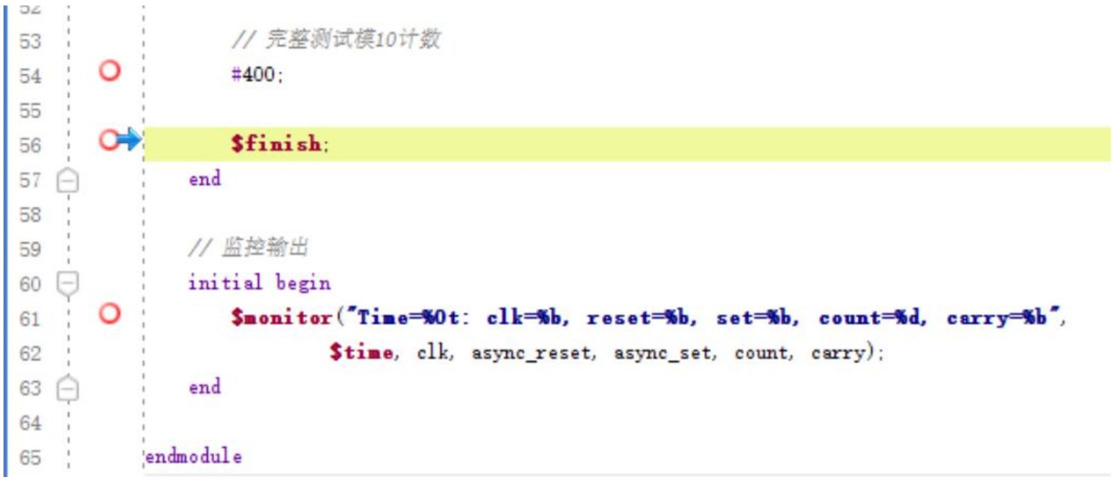


```verilog
`timescale 1ns / 1ps

module tb_counter_mod10_async;

    // 测试信号
    reg clk;
    reg async_reset;
    reg async_set;
    wire [3:0] count;
    wire carry;

    // 实例化被测试模块
    counter_mod10_async uut(
        .clk(clk),
        .async_reset(async_reset),
        .async_set(async_set),
        .count(count),
        .carry(carry)
    );

    // 时钟生成 - 周期20ns (50MHz)
    initial begin
        clk = 0;
        forever #10 clk = ~clk;
    end
```



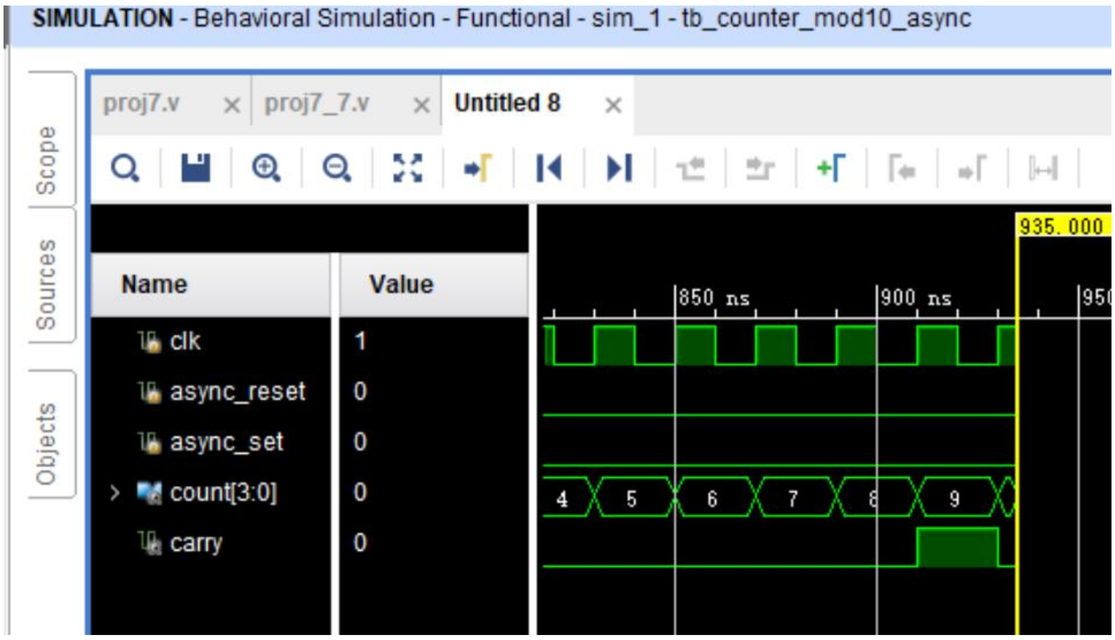
```verilog
    initial begin
        clk = 0;
        forever #10 clk = ~clk;
    end
```

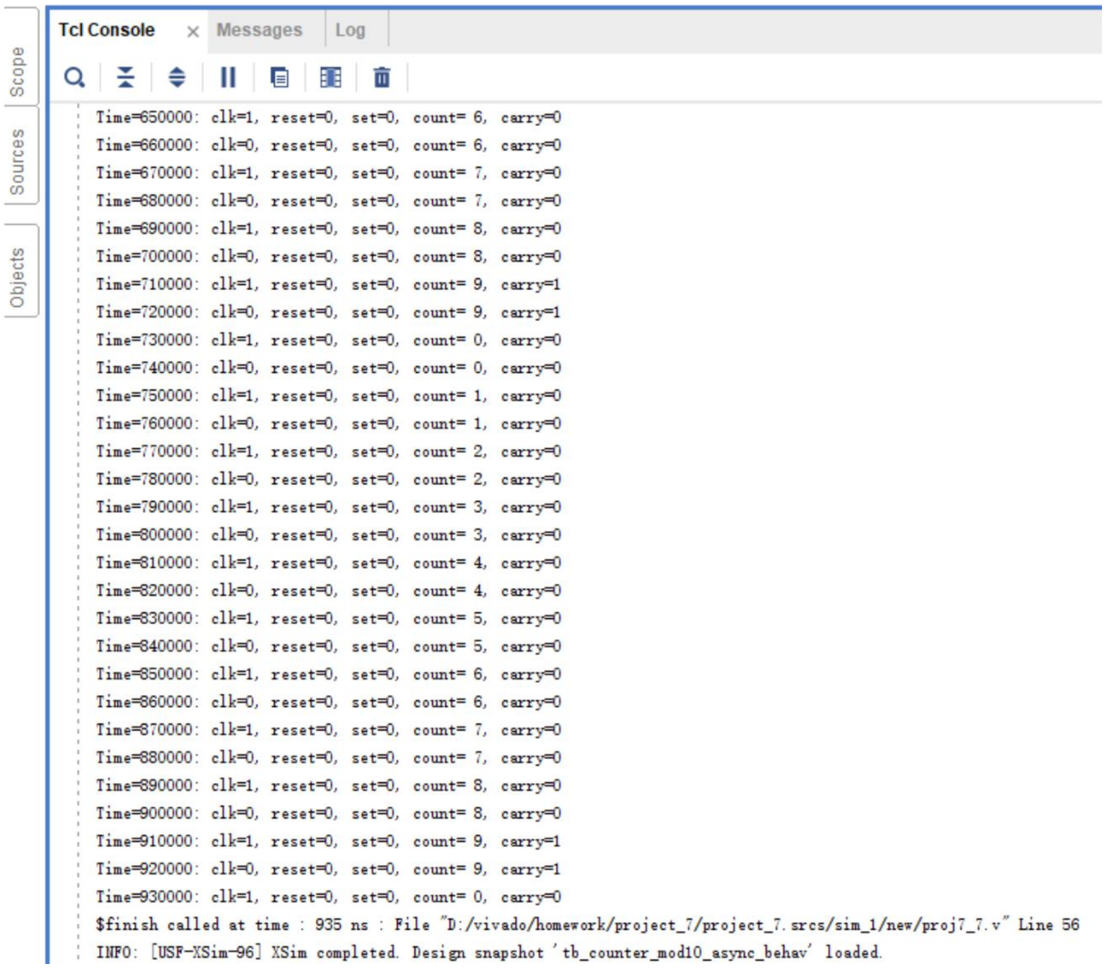
```verilog

    // 测试用例
    initial begin
        // 初始化
        async_reset = 0;
        async_set = 0;

        // 测试异步复位
        #15 async_reset = 1;
        #30 async_reset = 0;

        // 让计数器运行几个周期
        #200;

        // 测试异步置位
        #15 async_set = 1;
        #30 async_set = 0;

        // 继续运行
        #200;

        // 测试复位和置位同时有效（复位优先级高）
        #15 async_reset = 1;
        async_set = 1;
        #30 async_reset = 0;
        async_set = 0;
```



```verilog

        // 完整测试模10计数
        #400;

        $finish;
    end

    // 监控输出
    initial begin
        $monitor("Time=%0t: clk=%b, reset=%b, set=%b, count=%d, carry=%b",
                 $time, clk, async_reset, async_set, count, carry);
    end

endmodule
```

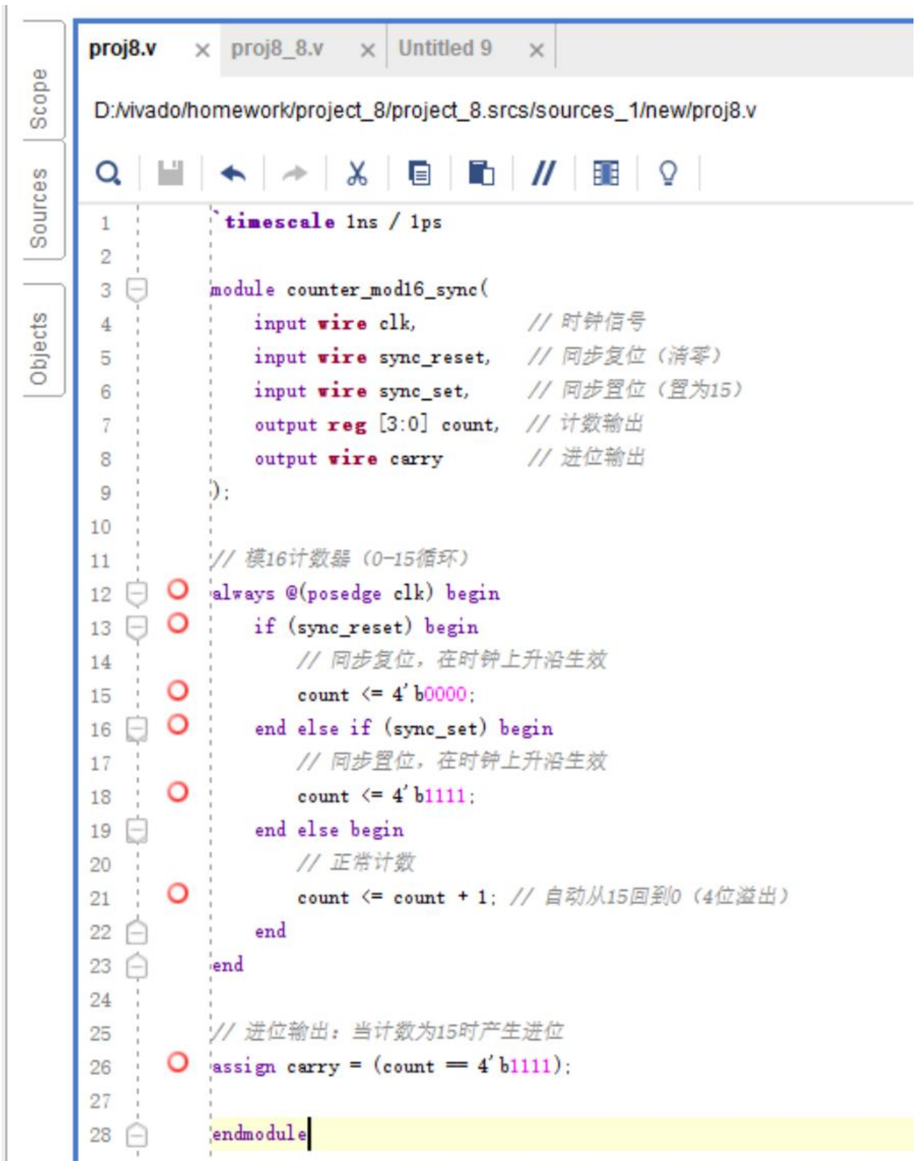
仿真结果





```
Time=650000: clk=1, reset=0, set=0, count= 6, carry=0
Time=660000: clk=0, reset=0, set=0, count= 6, carry=0
Time=670000: clk=1, reset=0, set=0, count= 7, carry=0
Time=680000: clk=0, reset=0, set=0, count= 7, carry=0
Time=690000: clk=1, reset=0, set=0, count= 8, carry=0
Time=700000: clk=0, reset=0, set=0, count= 8, carry=0
Time=710000: clk=1, reset=0, set=0, count= 9, carry=1
Time=720000: clk=0, reset=0, set=0, count= 9, carry=1
Time=730000: clk=1, reset=0, set=0, count= 0, carry=0
Time=740000: clk=0, reset=0, set=0, count= 0, carry=0
Time=750000: clk=1, reset=0, set=0, count= 1, carry=0
Time=760000: clk=0, reset=0, set=0, count= 1, carry=0
Time=770000: clk=1, reset=0, set=0, count= 2, carry=0
Time=780000: clk=0, reset=0, set=0, count= 2, carry=0
Time=790000: clk=1, reset=0, set=0, count= 3, carry=0
Time=800000: clk=0, reset=0, set=0, count= 3, carry=0
Time=810000: clk=1, reset=0, set=0, count= 4, carry=0
Time=820000: clk=0, reset=0, set=0, count= 4, carry=0
Time=830000: clk=1, reset=0, set=0, count= 5, carry=0
Time=840000: clk=0, reset=0, set=0, count= 5, carry=0
Time=850000: clk=1, reset=0, set=0, count= 6, carry=0
Time=860000: clk=0, reset=0, set=0, count= 6, carry=0
Time=870000: clk=1, reset=0, set=0, count= 7, carry=0
Time=880000: clk=0, reset=0, set=0, count= 7, carry=0
Time=890000: clk=1, reset=0, set=0, count= 8, carry=0
Time=900000: clk=0, reset=0, set=0, count= 8, carry=0
Time=910000: clk=1, reset=0, set=0, count= 9, carry=1
Time=920000: clk=0, reset=0, set=0, count= 9, carry=1
Time=930000: clk=1, reset=0, set=0, count= 0, carry=0
$finish called at time : 935 ns : File "D:/vivado/homework/project_7/project_7.srcs/sim_1/new/proj7_7.v" Line 56
INFO: [USF-XSim-96] XSim completed. Design snapshot 'tb_counter_mod10_async_behav' loaded.
```

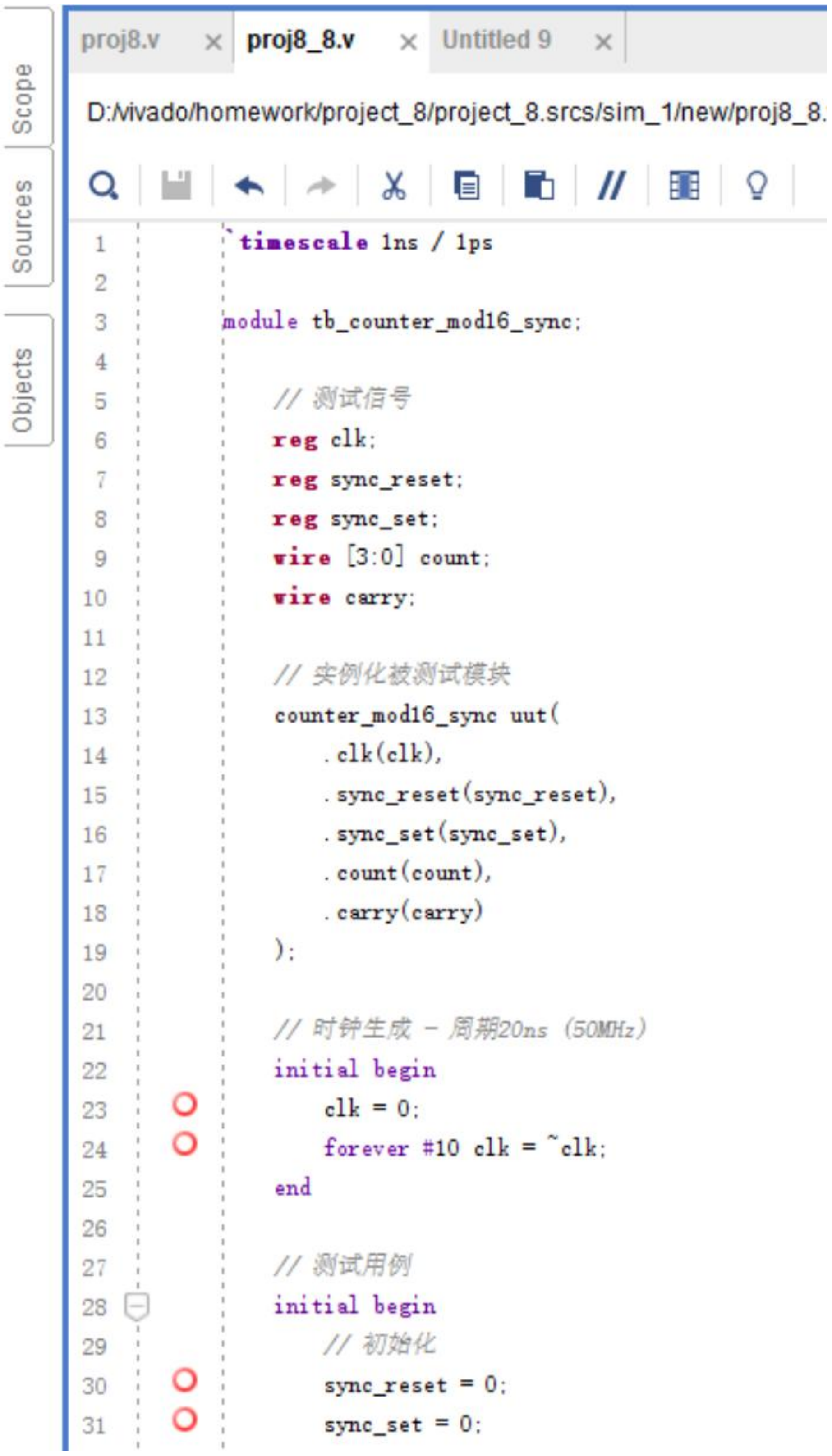
2.源程序



D:/vivado/homework/project_8/project_8.srcs/sources_1/new/proj8.v

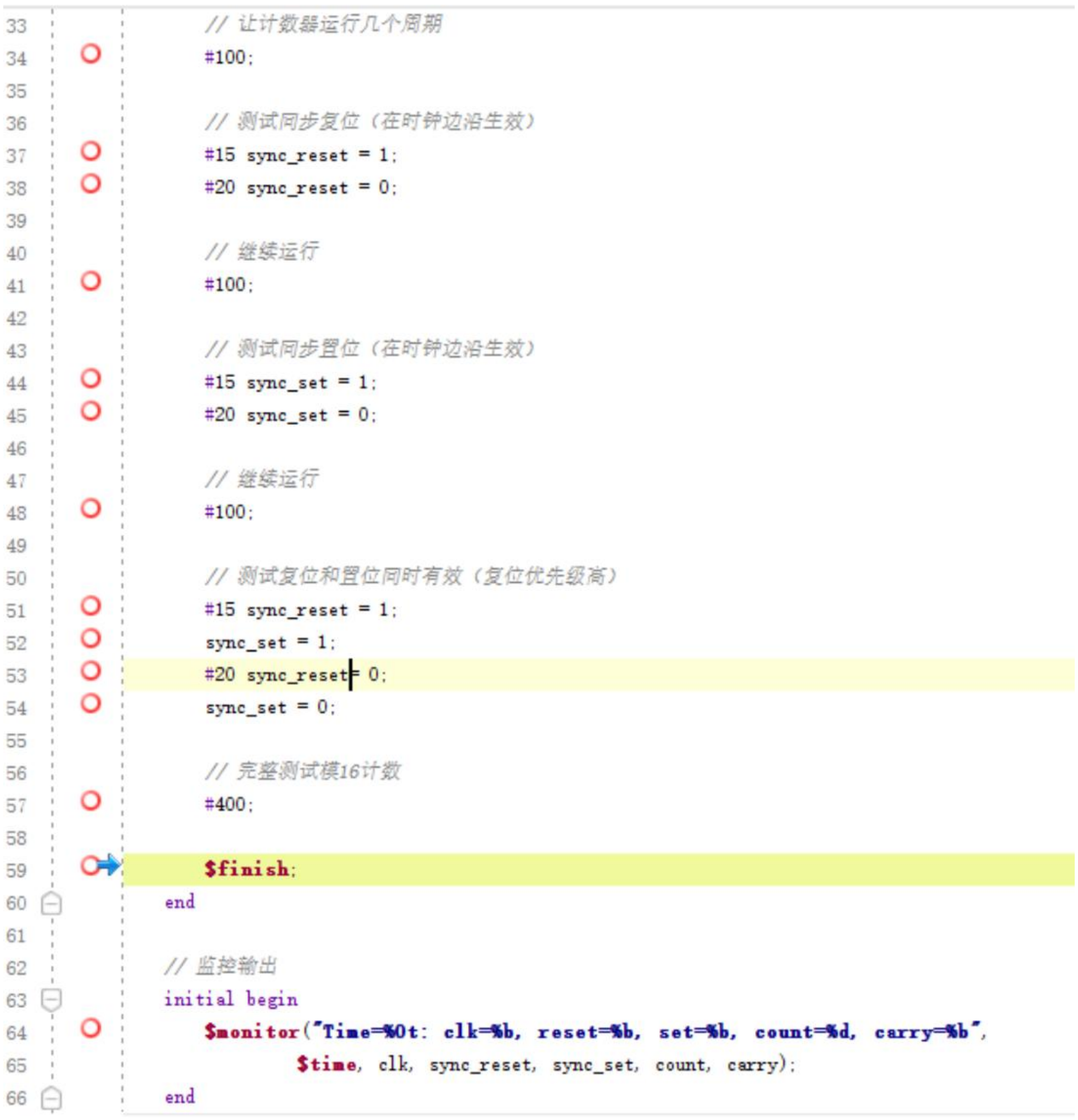
```
`timescale 1ns / 1ps

module counter_mod16_sync(
    input wire clk,           // 时钟信号
    input wire sync_reset,    // 同步复位（清零）
    input wire sync_set,      // 同步置位（置为15）
    output reg [3:0] count,   // 计数输出
    output wire carry         // 进位输出
);

// 模16计数器（0-15循环）
always @(posedge clk) begin
    if (sync_reset) begin
        // 同步复位，在时钟上升沿生效
        count <= 4'b0000;
    end else if (sync_set) begin
        // 同步置位，在时钟上升沿生效
        count <= 4'b1111;
    end else begin
        // 正常计数
        count <= count + 1; // 自动从15回到0（4位溢出）
    end
end

// 进位输出：当计数为15时产生进位
assign carry = (count == 4'b1111);

endmodule
```

测试代码



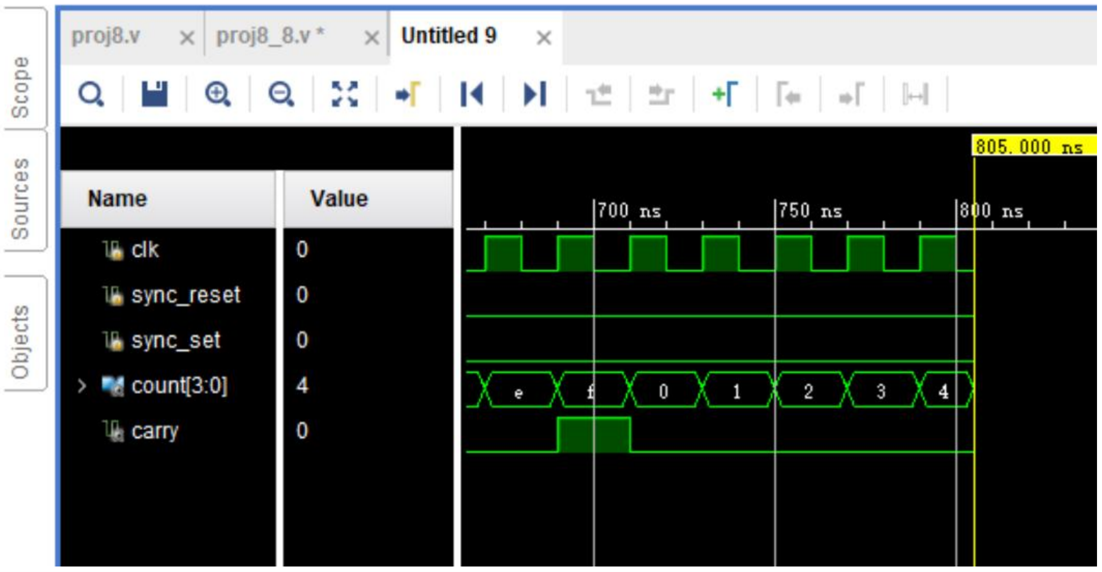
Scope
Sources
Objects
proj8.v
proj8_8.v
Untitled 9
D:/vivado/homework/project_8/project_8.srcs/sim_1/new/proj8_8.

```
`timescale 1ns / 1ps

module tb_counter_mod16_sync;

    // 测试信号
    reg clk;
    reg sync_reset;
    reg sync_set;
    wire [3:0] count;
    wire carry;

    // 实例化被测试模块
    counter_mod16_sync uut(
        .clk(clk),
        .sync_reset(sync_reset),
        .sync_set(sync_set),
        .count(count),
        .carry(carry)
    );

    // 时钟生成 - 周期20ns (50MHz)
    initial begin
        clk = 0;
        forever #10 clk = ~clk;
    end

    // 测试用例
    initial begin
        // 初始化
        sync_reset = 0;
        sync_set = 0;
```

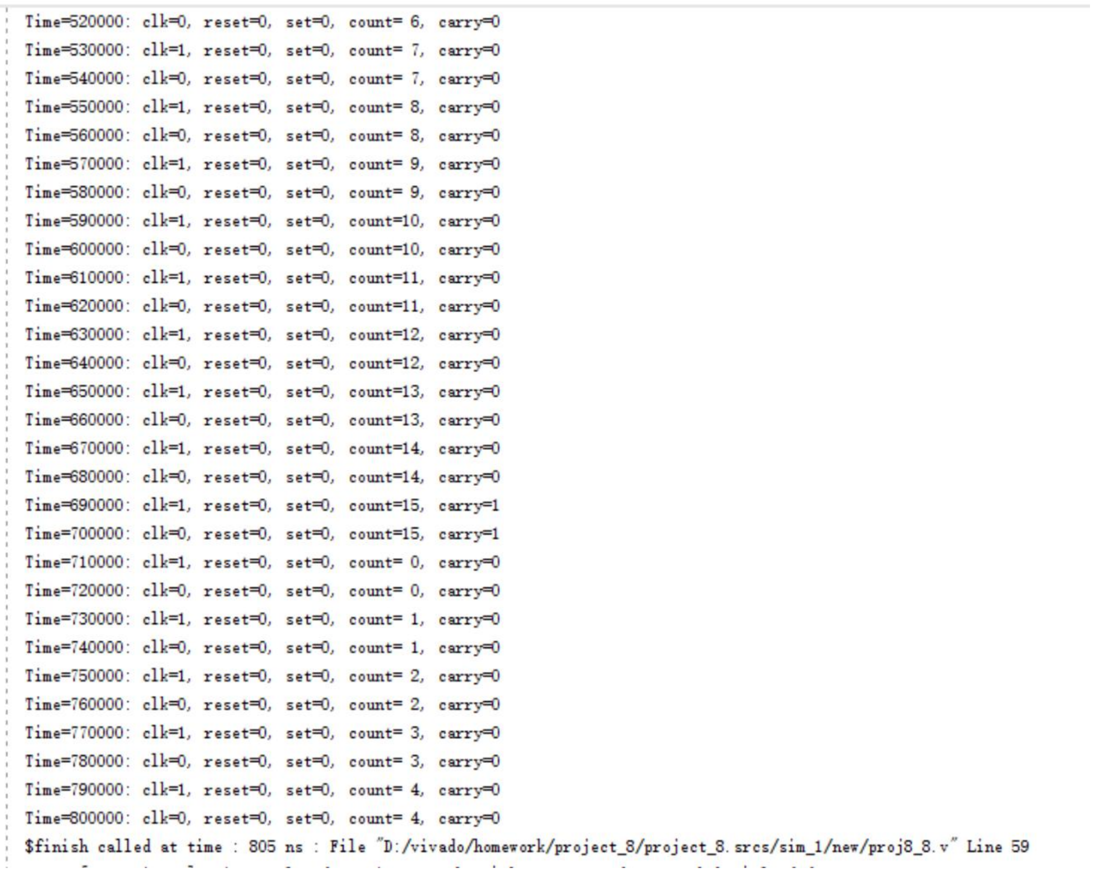


```
        // 让计数器运行几个周期
        #100;

        // 测试同步复位（在时钟边沿生效）
        #15 sync_reset = 1;
        #20 sync_reset = 0;

        // 继续运行
        #100;

        // 测试同步置位（在时钟边沿生效）
        #15 sync_set = 1;
        #20 sync_set = 0;

        // 继续运行
        #100;

        // 测试复位和置位同时有效（复位优先级高）
        #15 sync_reset = 1;
        sync_set = 1;
        #20 sync_reset = 0;
        sync_set = 0;

        // 完整测试模16计数
        #400;

        $finish;
    end

    // 监控输出
    initial begin
        $monitor("Time=%0t: clk=%b, reset=%b, set=%b, count=%d, carry=%b",
                 $time, clk, sync_reset, sync_set, count, carry);
    end
```

仿真结果



Name	Value
clk	0
sync_reset	0
sync_set	0
count[3:0]	4
carry	0



```
Time=520000: clk=0, reset=0, set=0, count= 6, carry=0
Time=530000: clk=1, reset=0, set=0, count= 7, carry=0
Time=540000: clk=0, reset=0, set=0, count= 7, carry=0
Time=550000: clk=1, reset=0, set=0, count= 8, carry=0
Time=560000: clk=0, reset=0, set=0, count= 8, carry=0
Time=570000: clk=1, reset=0, set=0, count= 9, carry=0
Time=580000: clk=0, reset=0, set=0, count= 9, carry=0
Time=590000: clk=1, reset=0, set=0, count=10, carry=0
Time=600000: clk=0, reset=0, set=0, count=10, carry=0
Time=610000: clk=1, reset=0, set=0, count=11, carry=0
Time=620000: clk=0, reset=0, set=0, count=11, carry=0
Time=630000: clk=1, reset=0, set=0, count=12, carry=0
Time=640000: clk=0, reset=0, set=0, count=12, carry=0
Time=650000: clk=1, reset=0, set=0, count=13, carry=0
Time=660000: clk=0, reset=0, set=0, count=13, carry=0
Time=670000: clk=1, reset=0, set=0, count=14, carry=0
Time=680000: clk=0, reset=0, set=0, count=14, carry=0
Time=690000: clk=1, reset=0, set=0, count=15, carry=1
Time=700000: clk=0, reset=0, set=0, count=15, carry=1
Time=710000: clk=1, reset=0, set=0, count= 0, carry=0
Time=720000: clk=0, reset=0, set=0, count= 0, carry=0
Time=730000: clk=1, reset=0, set=0, count= 1, carry=0
Time=740000: clk=0, reset=0, set=0, count= 1, carry=0
Time=750000: clk=1, reset=0, set=0, count= 2, carry=0
Time=760000: clk=0, reset=0, set=0, count= 2, carry=0
Time=770000: clk=1, reset=0, set=0, count= 3, carry=0
Time=780000: clk=0, reset=0, set=0, count= 3, carry=0
Time=790000: clk=1, reset=0, set=0, count= 4, carry=0
Time=800000: clk=0, reset=0, set=0, count= 4, carry=0
$finish called at time : 805 ns : File "D:/vivado/homework/project_8/project_8.srcs/sim_1/new/proj8_8.v" Line 59
```