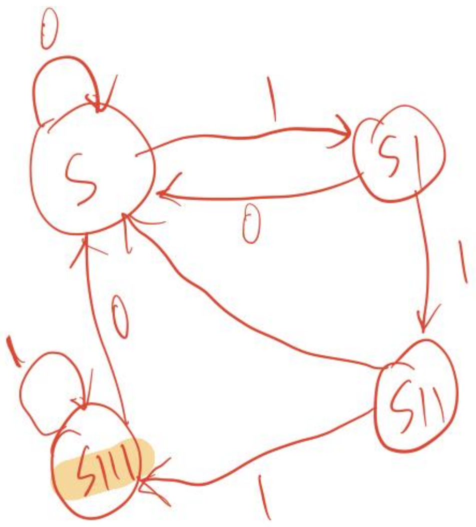
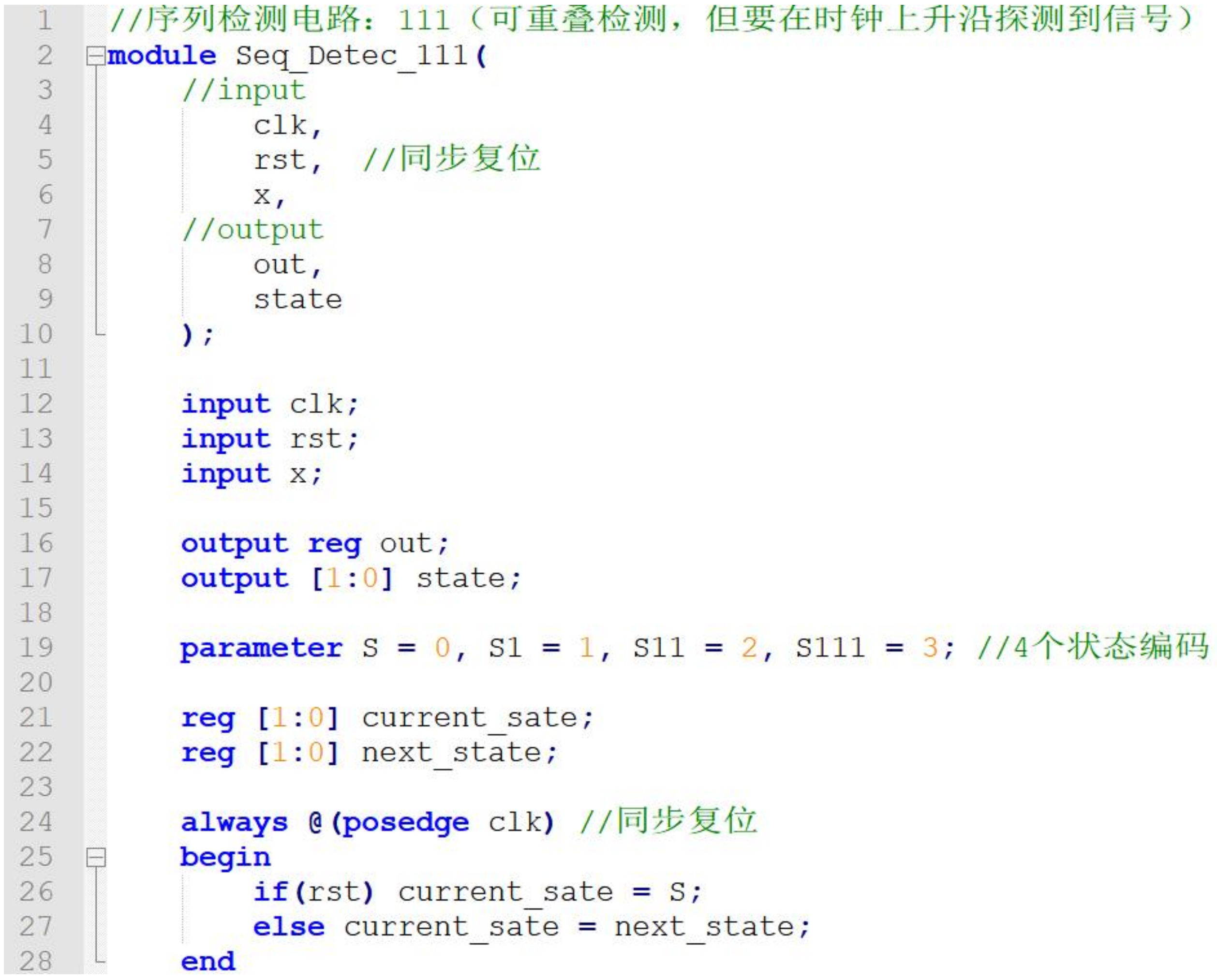
一．序列检测电路：111（可重叠检测，需要在时钟上升沿探测到信号才算）

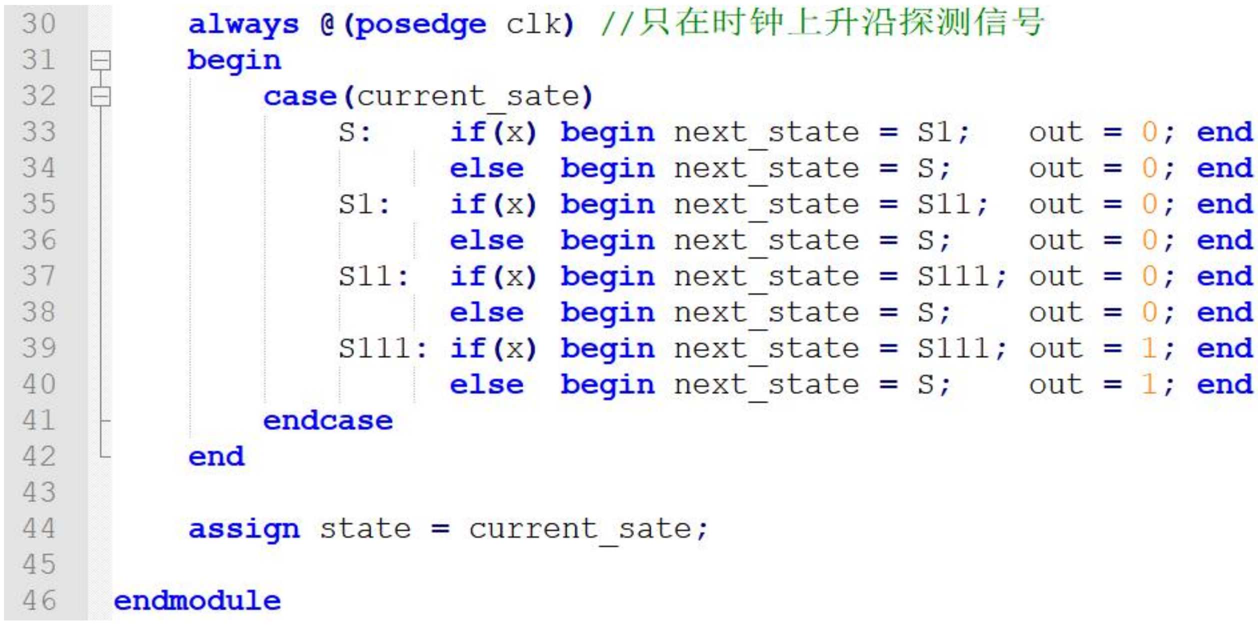
1. 设计思路：



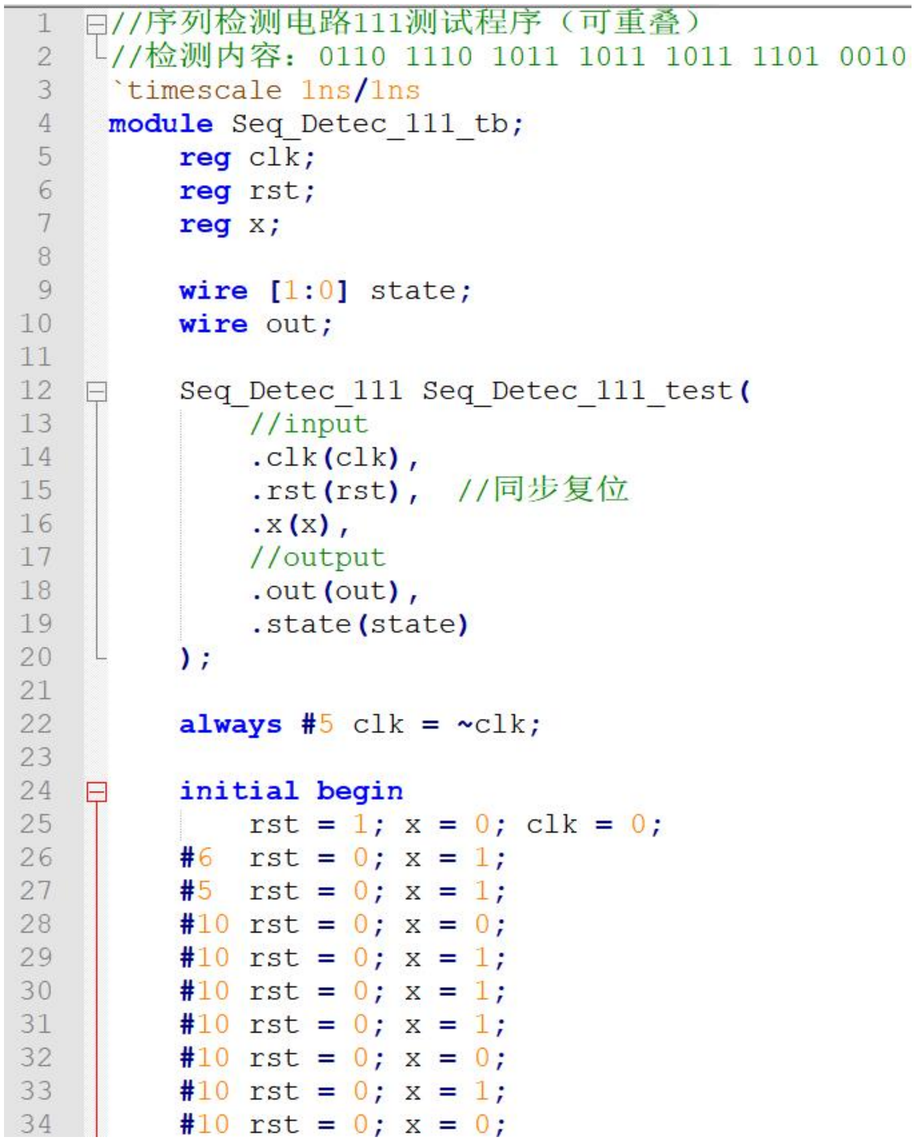
2. 功能模块代码：

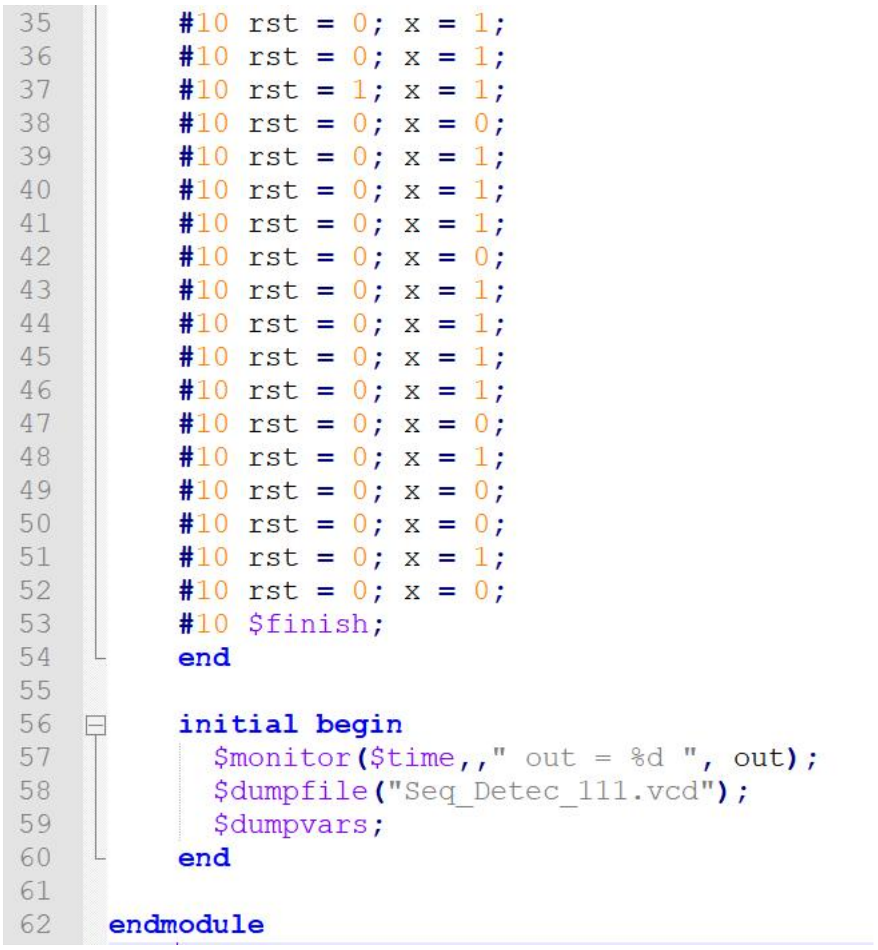


```
//序列检测电路：111（可重叠检测，但要在时钟上升沿探测到信号）
module Seq_Detec_111(
    //input
        clk,
        rst,  //同步复位
        x,
    //output
        out,
        state
    );

    input clk;
    input rst;
    input x;

    output reg out;
    output [1:0] state;

    parameter S = 0, S1 = 1, S11 = 2, S111 = 3; //4个状态编码

    reg [1:0] current_sate;
    reg [1:0] next_state;

    always @(posedge clk) //同步复位
    begin
        if(rst) current_sate = S;
        else current_sate = next_state;
    end
```


```
    always @(posedge clk) //只在时钟上升沿探测信号
    begin
        case(current_sate)
            S:    if(x) begin next_state = S1;   out = 0; end
                  else  begin next_state = S;    out = 0; end
            S1:   if(x) begin next_state = S11;  out = 0; end
                  else  begin next_state = S;    out = 0; end
            S11:  if(x) begin next_state = S111; out = 0; end
                  else  begin next_state = S;    out = 0; end
            S111: if(x) begin next_state = S111; out = 1; end
                  else  begin next_state = S;    out = 1; end
        endcase
    end

    assign state = current_sate;

endmodule
```

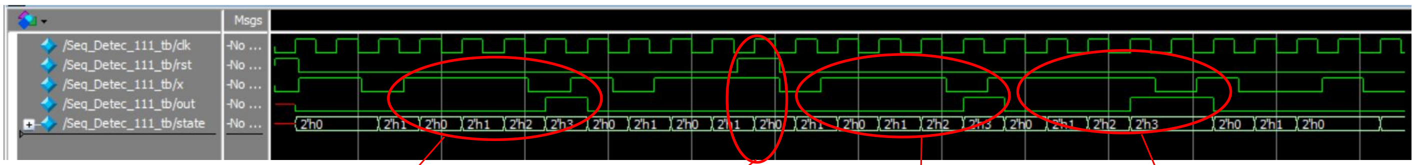
3. 测试模块代码：


```
//序列检测电路111测试程序（可重叠）
//检测内容：0110 1110 1011 1011 1011 1101 0010
`timescale 1ns/1ns
module Seq_Detec_111_tb;
    reg clk;
    reg rst;
    reg x;

    wire [1:0] state;
    wire out;

    Seq_Detec_111 Seq_Detec_111_test(
        //input
        .clk(clk),
        .rst(rst),  //同步复位
        .x(x),
        //output
        .out(out),
        .state(state)
    );

    always #5 clk = ~clk;

    initial begin
        rst = 1; x = 0; clk = 0;
    #6  rst = 0; x = 1;
    #5  rst = 0; x = 1;
    #10 rst = 0; x = 0;
    #10 rst = 0; x = 1;
    #10 rst = 0; x = 1;
    #10 rst = 0; x = 1;
    #10 rst = 0; x = 0;
    #10 rst = 0; x = 1;
    #10 rst = 0; x = 0;
```



```
        #10 rst = 0; x = 1;
        #10 rst = 0; x = 1;
        #10 rst = 1; x = 1;
        #10 rst = 0; x = 0;
        #10 rst = 0; x = 1;
        #10 rst = 0; x = 1;
        #10 rst = 0; x = 1;
        #10 rst = 0; x = 0;
        #10 rst = 0; x = 1;
        #10 rst = 0; x = 1;
        #10 rst = 0; x = 1;
        #10 rst = 0; x = 1;
        #10 rst = 0; x = 0;
        #10 rst = 0; x = 1;
        #10 rst = 0; x = 0;
        #10 rst = 0; x = 0;
        #10 rst = 0; x = 1;
        #10 rst = 0; x = 0;
        #10 $finish;
    end

    initial begin
        $monitor($time,," out = %d ", out);
        $dumpfile("Seq_Detec_111.vcd");
        $dumpvars;
    end

endmodule
```

4. ModelSim 仿真结果及分析：



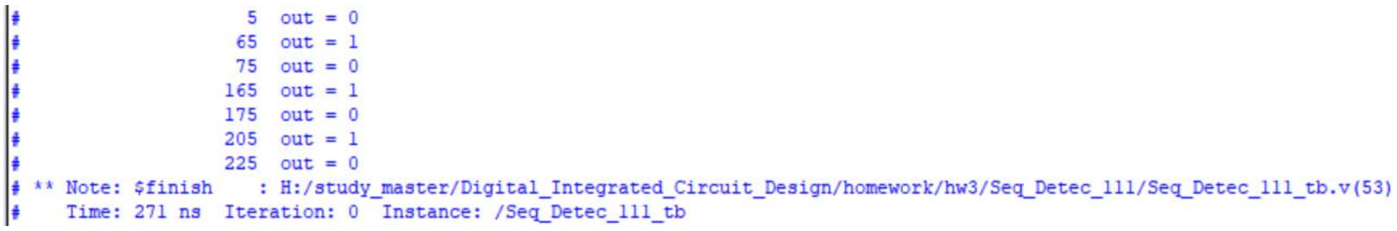
Msgs
/Seq_Detec_111_tb/clk
/Seq_Detec_111_tb/rst
/Seq_Detec_111_tb/x
/Seq_Detec_111_tb/out
/Seq_Detec_111_tb/state
2'h0 2'h1 2'h0 2'h1 2'h2 2'h3 2'h0 2'h1 2'h0 2'h1 2'h0 2'h1 2'h0 2'h1 2'h2 2'h3 2'h0 2'h1 2'h2 2'h3 2'h0 2'h1 2'h0

连续输入三个 1 之后，在下一个时钟上升沿输出为 1

复位信号为 1 时，在时钟上升沿来临后，状态变为初始状态，没有输出 1

连续输入三个 1 之后，在下一个时钟上升沿输出为 1

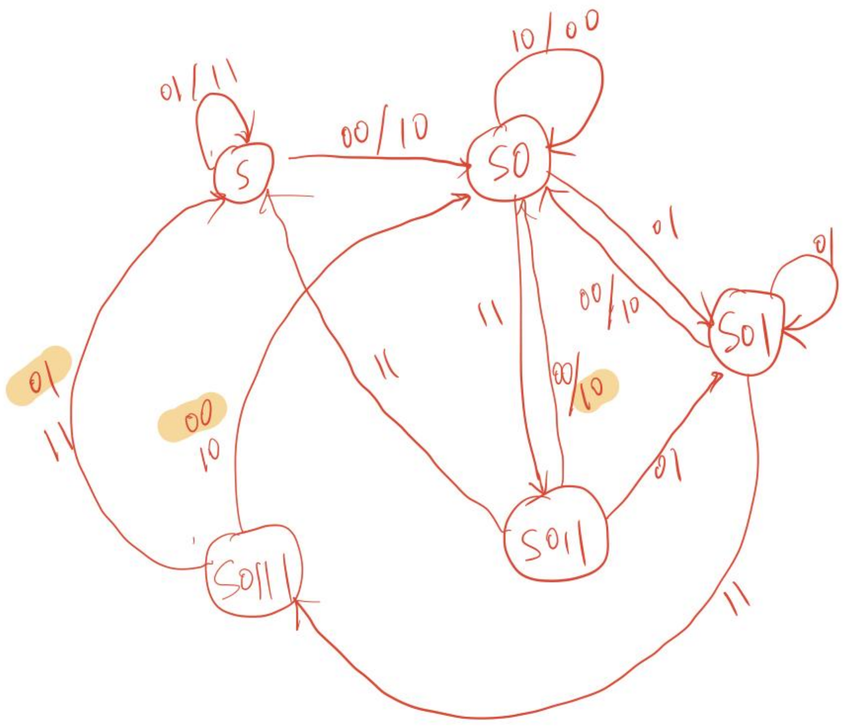
连续输入四个 1 之后，在下一个时钟上升沿连续两个时钟输出 1



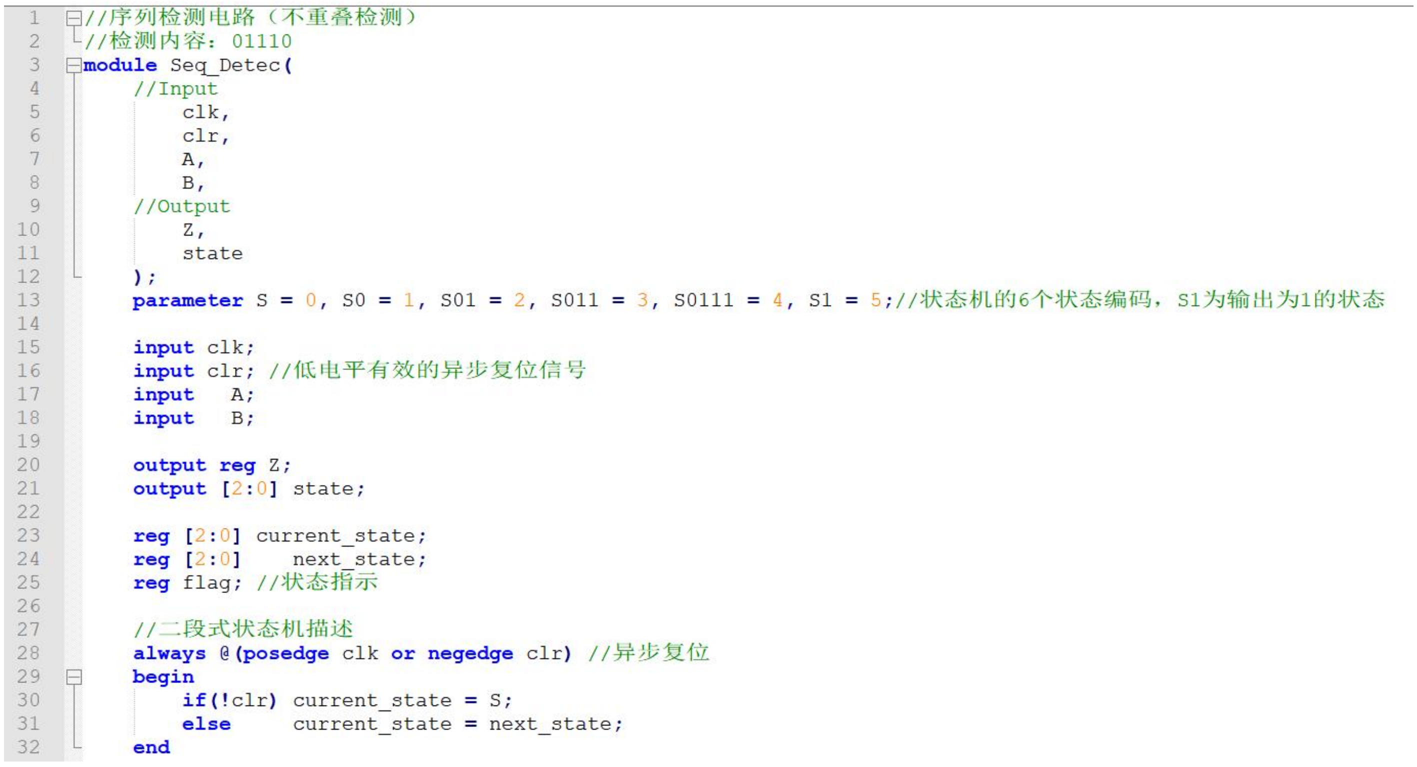
```
#                    5  out = 0
#                   65  out = 1
#                   75  out = 0
#                  165  out = 1
#                  175  out = 0
#                  205  out = 1
#                  225  out = 0
# ** Note: $finish    : H:/study_master/Digital_Integrated_Circuit_Design/homework/hw3/Seq_Detec_111/Seq_Detec_111_tb.v(53)
#    Time: 271 ns  Iteration: 0  Instance: /Seq_Detec_111_tb
```

二．序列检测电路：01110（不重叠检测）

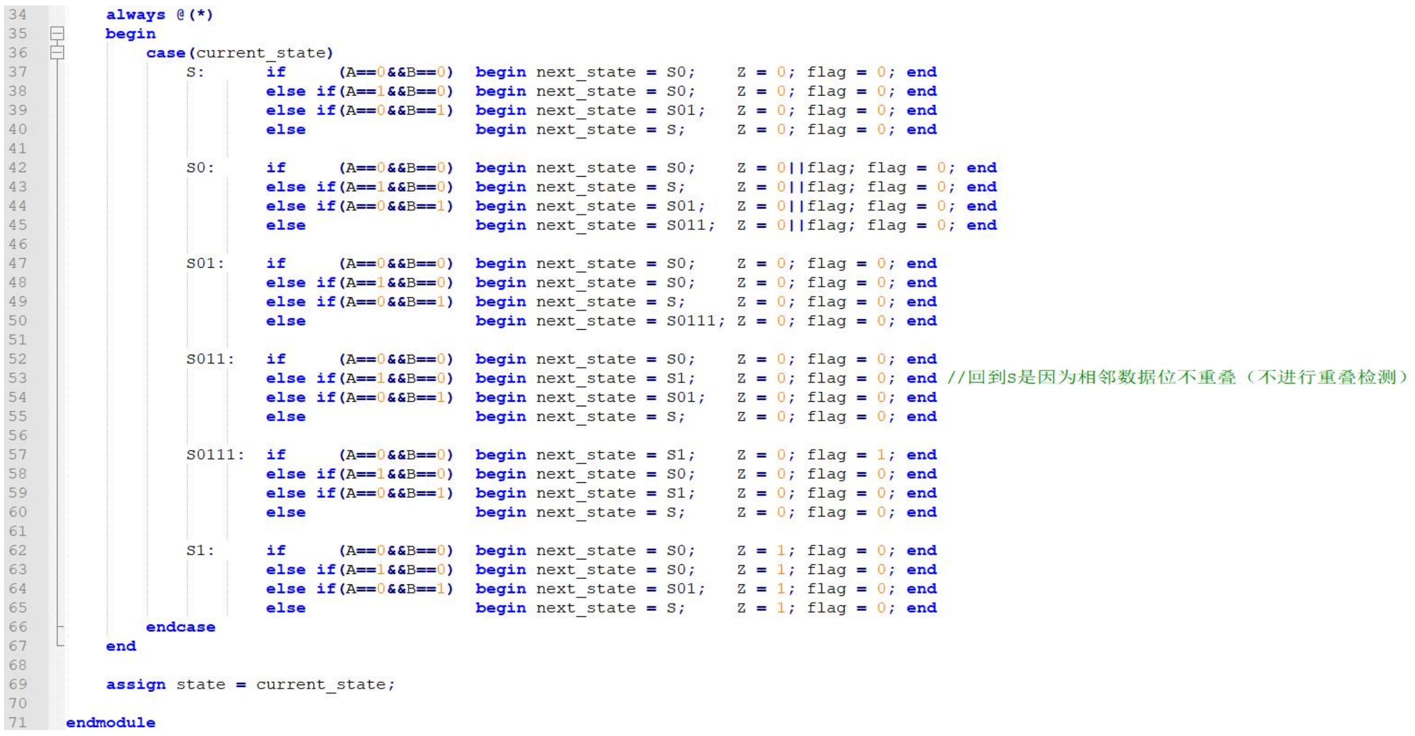
1. 设计思路：



2. 功能模块代码：二段式状态机，6 个状态



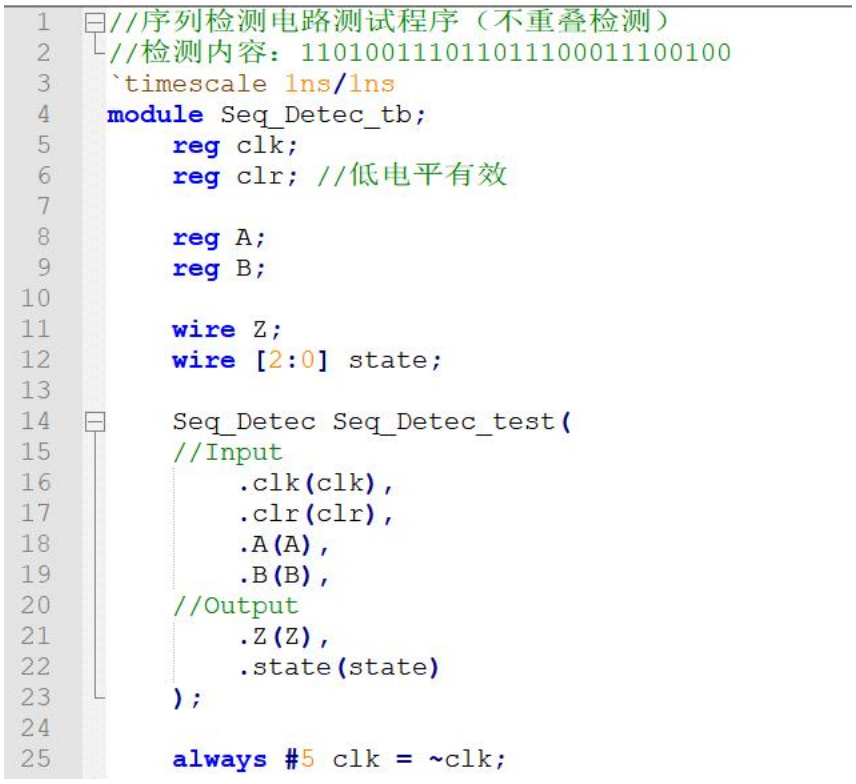
```
//序列检测电路（不重叠检测）
//检测内容：01110
module Seq_Detec(
    //Input
        clk,
        clr,
        A,
        B,
    //Output
        Z,
        state
    );
    parameter S = 0, S0 = 1, S01 = 2, S011 = 3, S0111 = 4, S1 = 5;//状态机的6个状态编码，S1为输出为1的状态

    input clk;
    input clr; //低电平有效的异步复位信号
    input   A;
    input   B;

    output reg Z;
    output [2:0] state;

    reg [2:0] current_state;
    reg [2:0]    next_state;
    reg flag; //状态指示

    //二段式状态机描述
    always @(posedge clk or negedge clr) //异步复位
    begin
        if(!clr) current_state = S;
        else     current_state = next_state;
    end
```



```verilog
    always @(*)
    begin
        case(current_state)
            S:      if      (A==0&&B==0)  begin next_state = S0;    Z = 0; flag = 0; end
                    else if(A==1&&B==0)   begin next_state = S0;    Z = 0; flag = 0; end
                    else if(A==0&&B==1)   begin next_state = S01;   Z = 0; flag = 0; end
                    else                  begin next_state = S;     Z = 0; flag = 0; end

            S0:     if      (A==0&&B==0)  begin next_state = S0;    Z = 0||flag; flag = 0; end
                    else if(A==1&&B==0)   begin next_state = S;     Z = 0||flag; flag = 0; end
                    else if(A==0&&B==1)   begin next_state = S01;   Z = 0||flag; flag = 0; end
                    else                  begin next_state = S011;  Z = 0||flag; flag = 0; end

            S01:    if      (A==0&&B==0)  begin next_state = S0;    Z = 0; flag = 0; end
                    else if(A==1&&B==0)   begin next_state = S0;    Z = 0; flag = 0; end
                    else if(A==0&&B==1)   begin next_state = S;     Z = 0; flag = 0; end
                    else                  begin next_state = S0111; Z = 0; flag = 0; end

            S011:   if      (A==0&&B==0)  begin next_state = S0;    Z = 0; flag = 0; end
                    else if(A==1&&B==0)   begin next_state = S1;    Z = 0; flag = 0; end //回到s是因为相邻数据位不重叠（不进行重叠检测）
                    else if(A==0&&B==1)   begin next_state = S01;   Z = 0; flag = 0; end
                    else                  begin next_state = S;     Z = 0; flag = 0; end

            S0111:  if      (A==0&&B==0)  begin next_state = S1;    Z = 0; flag = 1; end
                    else if(A==1&&B==0)   begin next_state = S0;    Z = 0; flag = 0; end
                    else if(A==0&&B==1)   begin next_state = S1;    Z = 0; flag = 0; end
                    else                  begin next_state = S;     Z = 0; flag = 0; end

            S1:     if      (A==0&&B==0)  begin next_state = S0;    Z = 1; flag = 0; end
                    else if(A==1&&B==0)   begin next_state = S0;    Z = 1; flag = 0; end
                    else if(A==0&&B==1)   begin next_state = S01;   Z = 1; flag = 0; end
                    else                  begin next_state = S;     Z = 1; flag = 0; end
        endcase
    end

    assign state = current_state;

endmodule
```

2. 测试模块代码：

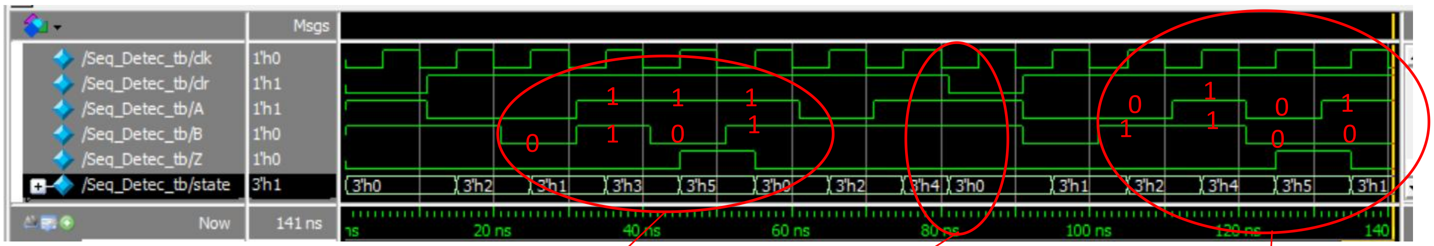
输入序列 1101001110110111000111001 00，有三个不重叠的 01110，中间的一个 01110 中有 clr 信号产生。



```verilog
//序列检测电路测试程序（不重叠检测）
//检测内容：1101001110110111000111001 00
`timescale 1ns/1ns
module Seq_Detec_tb;
    reg clk;
    reg clr; //低电平有效

    reg A;
    reg B;

    wire Z;
    wire [2:0] state;

    Seq_Detec Seq_Detec_test(
    //Input
        .clk(clk),
        .clr(clr),
        .A(A),
        .B(B),
    //Output
        .Z(Z),
        .state(state)
    );

    always #5 clk = ~clk;
```

```
initial begin
    A = 1; B = 1; clr = 0; clk = 0;
#11  A = 0; B = 1; clr = 1;
#10 A = 0; B = 0; clr = 1;
#10 A = 1; B = 1; clr = 1;
#10 A = 1; B = 0; clr = 1;
#10 A = 1; B = 1; clr = 1;
#10 A = 0; B = 1; clr = 1;
#10 A = 1; B = 1; clr = 1;
#10 A = 1; B = 1; clr = 0;
#10 A = 0; B = 0; clr = 1;
#10 A = 0; B = 1; clr = 1;
#10 A = 1; B = 1; clr = 1;
#10 A = 0; B = 0; clr = 1;
#10 A = 1; B = 0; clr = 1;
#10 $finish;
end

initial begin
    $monitor($time,,"Z = %d", Z);
    $dumpfile("Seq_Detec.vcd");
    $dumpvars;
end

endmodule
```

4. ModelSim 仿真结果及分析：



输入序列：01110，在下一个时钟上升沿到来后，状态变为 S1，并输出高电平。之后的时钟上升沿之前输入是 11，则下一个状态转为初始状态 S。

在时钟周期内将 clr 置为低电平，状态立即转为初始状态 S，而没有等待时钟上升沿。

输入序列：01110，在下一个时钟上升沿到来后，状态变为 S1，并输出高电平。之后的时钟上升沿之前输入是 10，则下一个状态转为初始状态 S0。

```
#                    0 Z = 0
#                   45 Z = 1
#                   55 Z = 0
#                  125 Z = 1
#                  135 Z = 0
# ** Note: $finish    : H:/study_master/Digital_Integrated_Circuit_Design/homework/hw3/Seq_Detec/Seq_Detec_tb.v(42)
#    Time: 141 ns  Iteration: 0  Instance: /Seq_Detec_tb
```