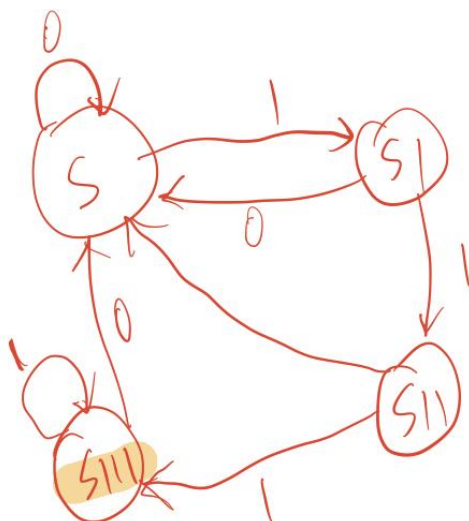


一. 序列检测电路：111（可重叠检测，需要在时钟上升沿探测到信号才算）

1. 设计思路：



2. 功能模块代码：

```
1 //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）
2 module Seq_Detec_111(
3     //input
4     clk,
5     rst, //同步复位
6     x,
7     //output
8     out,
9     state
10 );
11
12 input clk;
13 input rst;
14 input x;
15
16 output reg out;
17 output [1:0] state;
18
19 parameter S = 0, S1 = 1, S11 = 2, S111 = 3; //4个状态编码
20
21 reg [1:0] current_sate;
22 reg [1:0] next_state;
23
24 always @(posedge clk) //同步复位
25 begin
26     if(rst) current_sate = S;
27     else current_sate = next_state;
28 end
```

```

30     always @(posedge clk) //只在时钟上升沿探测信号
31     begin
32         case(current_sate)
33             S:      if(x) begin next_state = S1;   out = 0; end
34                     else begin next_state = S;     out = 0; end
35             S1:     if(x) begin next_state = S11;  out = 0; end
36                     else begin next_state = S;     out = 0; end
37             S11:    if(x) begin next_state = S111; out = 0; end
38                     else begin next_state = S;     out = 0; end
39             S111:   if(x) begin next_state = S111; out = 1; end
40                     else begin next_state = S;     out = 1; end
41         endcase
42     end
43
44     assign state = current_sate;
45
46 endmodule

```

3. 测试模块代码:

```

1  //序列检测电路111测试程序（可重叠）
2  //检测内容: 0110 1110 1011 1011 1011 1101 0010
3  `timescale 1ns/1ns
4  module Seq_Detec_111_tb;
5      reg clk;
6      reg rst;
7      reg x;
8
9      wire [1:0] state;
10     wire out;
11
12     Seq_Detec_111 Seq_Detec_111_test(
13         //input
14         .clk(clk),
15         .rst(rst), //同步复位
16         .x(x),
17         //output
18         .out(out),
19         .state(state)
20     );
21
22     always #5 clk = ~clk;
23
24     initial begin
25         rst = 1; x = 0; clk = 0;
26         #6  rst = 0; x = 1;
27         #5  rst = 0; x = 1;
28         #10 rst = 0; x = 0;
29         #10 rst = 0; x = 1;
30         #10 rst = 0; x = 1;
31         #10 rst = 0; x = 1;
32         #10 rst = 0; x = 0;
33         #10 rst = 0; x = 1;
34         #10 rst = 0; x = 0;

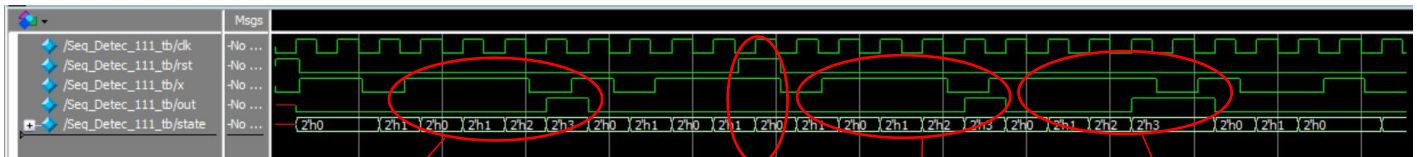
```

```

35      #10 rst = 0; x = 1;
36      #10 rst = 0; x = 1;
37      #10 rst = 1; x = 1;
38      #10 rst = 0; x = 0;
39      #10 rst = 0; x = 1;
40      #10 rst = 0; x = 1;
41      #10 rst = 0; x = 1;
42      #10 rst = 0; x = 0;
43      #10 rst = 0; x = 1;
44      #10 rst = 0; x = 1;
45      #10 rst = 0; x = 1;
46      #10 rst = 0; x = 1;
47      #10 rst = 0; x = 0;
48      #10 rst = 0; x = 1;
49      #10 rst = 0; x = 0;
50      #10 rst = 0; x = 0;
51      #10 rst = 0; x = 1;
52      #10 rst = 0; x = 0;
53      #10 $finish;
54      end
55
56      initial begin
57          $monitor($time,, " out = %d ", out);
58          $dumpfile("Seq_Detec_111.vcd");
59          $dumpvars;
60      end
61
62  endmodule

```

4. ModelSim 仿真结果及分析:



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

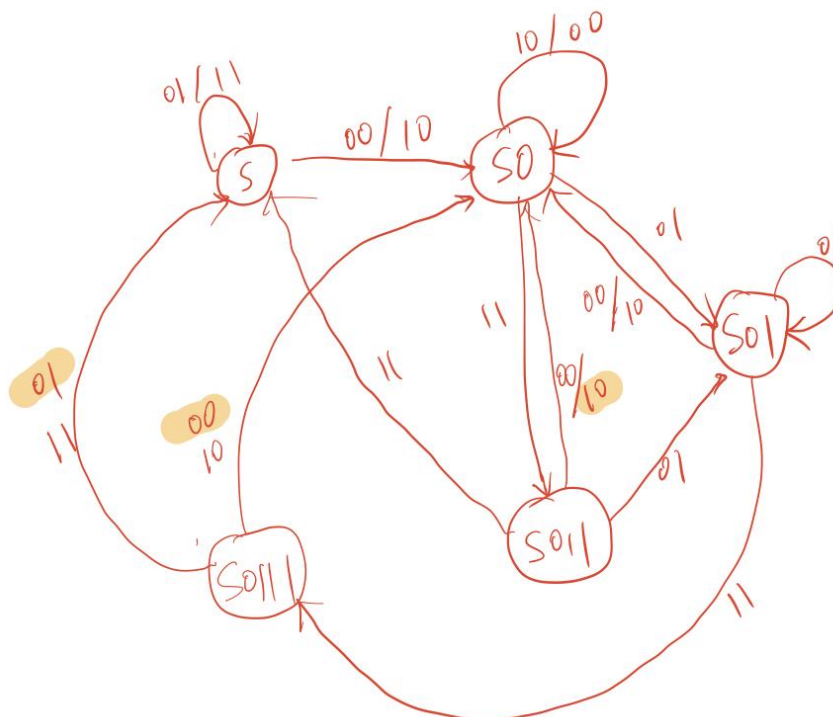
```

#          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

```
1 //序列检测电路（不重叠检测）
2 //检测内容：01110
3 module Seq_Detec(
4     //Input
5     clk,
6     clr,
7     A,
8     B,
9     //Output
10    Z,
11    state
12 );
13 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
14
15 input clk;
16 input clr; //低电平有效的异步复位信号
17 input A;
18 input B;
19
20 output reg Z;
21 output [2:0] state;
22
23 reg [2:0] current_state;
24 reg [2:0] next_state;
25 reg flag; //状态指示
26
27 //二段式状态机描述
28 always @(posedge clk or negedge clr) //异步复位
29 begin
30     if(!clr) current_state = S;
31     else current_state = next_state;
32 end
```



```

34 always @(*)
35 begin
36     case(current_state)
37         S: if (A==0&&B==0) begin next_state = S0; Z = 0; flag = 0; end
38             else if (A==1&&B==0) begin next_state = S0; Z = 0; flag = 0; end
39             else if (A==0&&B==1) begin next_state = S01; Z = 0; flag = 0; end
40             else begin next_state = S; Z = 0; flag = 0; end
41
42         S0: if (A==0&&B==0) begin next_state = S0; Z = 0||flag; flag = 0; end
43             else if (A==1&&B==0) begin next_state = S; Z = 0||flag; flag = 0; end
44             else if (A==0&&B==1) begin next_state = S01; Z = 0||flag; flag = 0; end
45             else begin next_state = S011; Z = 0||flag; flag = 0; end
46
47         S01: if (A==0&&B==0) begin next_state = S0; Z = 0; flag = 0; end
48             else if (A==1&&B==0) begin next_state = S0; Z = 0; flag = 0; end
49             else if (A==0&&B==1) begin next_state = S; Z = 0; flag = 0; end
50             else begin next_state = S0111; Z = 0; flag = 0; end
51
52         S011: if (A==0&&B==0) begin next_state = S0; Z = 0; flag = 0; end
53             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）
54             else if (A==0&&B==1) begin next_state = S01; Z = 0; flag = 0; end
55             else begin next_state = S; Z = 0; flag = 0; end
56
57         S0111: if (A==0&&B==0) begin next_state = S1; Z = 0; flag = 1; end
58             else if (A==1&&B==0) begin next_state = S0; Z = 0; flag = 0; end
59             else if (A==0&&B==1) begin next_state = S1; Z = 1; flag = 0; end
60             else begin next_state = S; Z = 0; flag = 0; end
61
62         S1: if (A==0&&B==0) begin next_state = S0; Z = 1; flag = 0; end
63             else if (A==1&&B==0) begin next_state = S0; Z = 1; flag = 0; end
64             else if (A==0&&B==1) begin next_state = S01; Z = 1; flag = 0; end
65             else begin next_state = S; Z = 1; flag = 0; end
66     endcase
67 end
68
69 assign state = current_state;
70
71 endmodule

```

2. 测试模块代码:

输入序列 110100111011011100011100100, 有三个不重叠的 01110, 中间的一个 01110 中有 clr 信号产生。

```

1 //序列检测电路测试程序（不重叠检测）
2 //检测内容: 110100111011011100011100100
3 `timescale 1ns/1ns
4 module Seq_Detec_tb;
5     reg clk;
6     reg clr; //低电平有效
7
8     reg A;
9     reg B;
10
11     wire Z;
12     wire [2:0] state;
13
14     Seq_Detec Seq_Detec_test(
15         //Input
16         .clk(clk),
17         .clr(clr),
18         .A(A),
19         .B(B),
20         //Output
21         .Z(Z),
22         .state(state)
23     );
24
25     always #5 clk = ~clk;

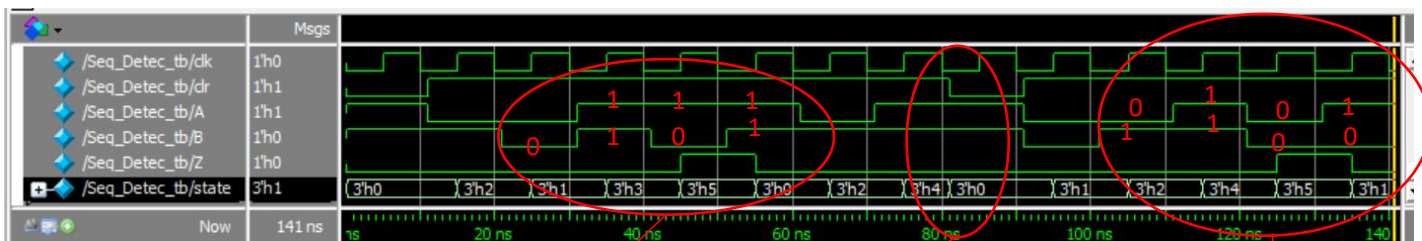
```

```

27 initial begin
28     A = 1; B = 1; clr = 0; clk = 0;
29     #11 A = 0; B = 1; clr = 1;
30     #10 A = 0; B = 0; clr = 1;
31     #10 A = 1; B = 1; clr = 1;
32     #10 A = 1; B = 0; clr = 1;
33     #10 A = 1; B = 1; clr = 1;
34     #10 A = 0; B = 1; clr = 1;
35     #10 A = 1; B = 1; clr = 1;
36     #10 A = 1; B = 1; clr = 0;
37     #10 A = 0; B = 0; clr = 1;
38     #10 A = 0; B = 1; clr = 1;
39     #10 A = 1; B = 1; clr = 1;
40     #10 A = 0; B = 0; clr = 1;
41     #10 A = 1; B = 0; clr = 1;
42     #10 $finish;
43 end
44
45 initial begin
46     $monitor($time, "Z = %d", Z);
47     $dumpfile("Seq_Detec.vcd");
48     $dumpvars;
49 end
50
51 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

```