有限状态机

1.状态表

现态 (Current State)	输入 (Input)	次态 (Next State)	输出(Output)
s_0	x=1	s_2	0
s_0	y=1	s_3	1
s_1	x=1	s_2	1
s_1	y=1	s_3	0
s_2	x=1	s_1	0
s_2	y=1	s_3	1
s_3	y=1	s_2	0
s_3	x=1	s_1	1

2.Z的逻辑方程：

由于是四位的one-hot编码我们可以有以下定义：

$$Z = s_0 \cdot \overline{x} \cdot y + s_1 \cdot x \cdot \overline{y} + s_2 \cdot \overline{x} \cdot y + s_3 \cdot x \cdot \overline{y}$$

3.控制逻辑单元建模：

FSM.v

```
`timescale 1ns / 1ps  // 指定仿真的时间单位为纳秒和时间精度为皮秒

module FSM(
    output reg z,       // 定义输出变量z，使用reg类型，因为需要在always块中进行赋值操作。
    input x, y,         // 定义输入变量x和y，用于接收外部信号。
    input rst,          // 定义输入复位信号rst，用于重置状态机到初始状态。
    input clk           // 定义输入时钟信号clk，用于同步状态转换。
);

// 定义状态编码，使用4位二进制数表示四种状态。
parameter S0 = 4'b0001;
parameter S1 = 4'b0010;
parameter S2 = 4'b0100;
parameter S3 = 4'b1000;

reg [3:0] nextstate; // 下一个状态的变量
wire [3:0] state;    // 当前状态，由D触发器维持
reg en;              // 使能信号，控制状态寄存器是否更新
wire [1:0] inxy;     // 组合输入x和y

assign inxy = {x, y}; // 将输入x和y组合成一个二位的二进制数
```

```
22
23  // 状态寄存器的实例化，使用自定义的dff模块，控制状态的存储和更新。
24  dff dff0(.d(nextstate), .clk(clk), .rst(rst), .q(state), .en(en));
25
26  // 主要的逻辑控制块
27  always @(posedge clk or posedge rst) begin
28      if (rst) begin
29          nextstate = S0; // 如果复位，则状态设置为S0
30          en = 1;         // 使能状态寄存器更新
31          z = 0;          // 输出z重置为0
32      end else begin
33          en = 0;         // 默认不更新状态
34          if (inxy != 2'b00 && inxy != 2'b11) begin  // 当输入不是00或11时，处理状
    态转换
35              en = 1;     // 使能状态寄存器更新
36              case (state)  // 根据当前状态进行处理
37                  S0: begin
38                      if (inxy == 2'b10) begin
39                          nextstate = S2;
40                          z = 0;
41                      end else if (inxy == 2'b01) begin
42                          nextstate = S3;
43                          z = 1;
44                      end
45                  end
46                  S1: begin
47                      if (inxy == 2'b10) begin
48                          nextstate = S2;
49                          z = 1;
50                      end else if (inxy == 2'b01) begin
51                          nextstate = S3;
52                          z = 0;
53                      end
54                  end
55                  S2: begin
56                      if (inxy == 2'b10) begin
57                          nextstate = S1;
58                          z = 0;
59                      end else if (inxy == 2'b01) begin
60                          nextstate = S3;
61                          z = 1;
62                      end
63                  end
64                  S3: begin
65                      if (inxy == 2'b10) begin
66                          nextstate = S1;
67                          z = 1;
68                      end else if (inxy == 2'b01) begin
69                          nextstate = S2;
70                          z = 0;
71                      end
72                  end
73                  default: begin
74                      nextstate = S0; // 默认状态为S0
75                      en = 1;         // 总是使能，以确保从非法状态恢复
76                  end
```

```
            endcase
        end
    end
end

endmodule
```

dff.v

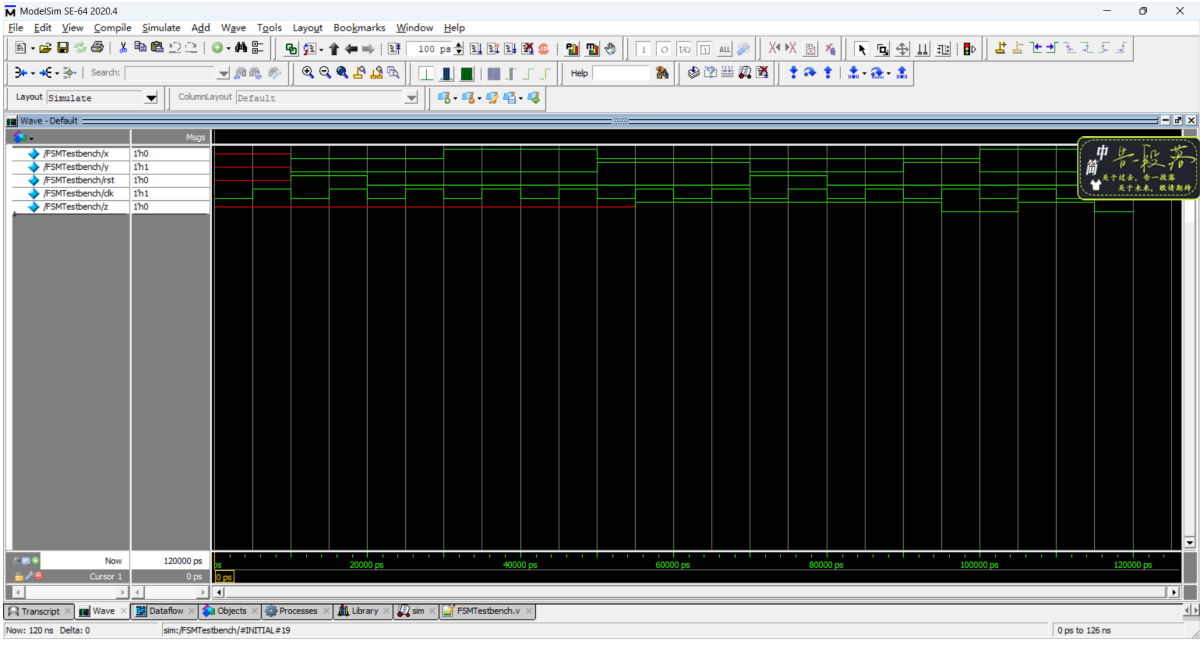
```
`timescale 1ns / 1ps  // 指定仿真的时间单位为1纳秒，时间精度为1皮秒。

// 定义模块dff，该模块用于实现一个4位宽的D触发器
module dff(
    output reg [3:0] q,  // 输出端口q，4位寄存器，存储触发器的当前状态
    input [3:0] d,       // 输入端口d，4位，用于设置触发器的目标状态
    input clk,           // 输入时钟信号clk，用于同步状态更新
    input rst,           // 输入复位信号rst，用于重置触发器状态到初始值
    input en             // 输入使能信号en，控制是否更新触发器的状态
);

// 描述触发器的行为，always块在时钟信号clk的上升沿触发
always @(posedge clk) begin
    if (rst)
        q <= 4'b0001;       // 如果复位信号激活，则将输出q重置为0001
    else if (en)
        q <= d;             // 如果使能信号激活，将输入d的值赋给输出q
end

endmodule
```

FSMTestbench.v

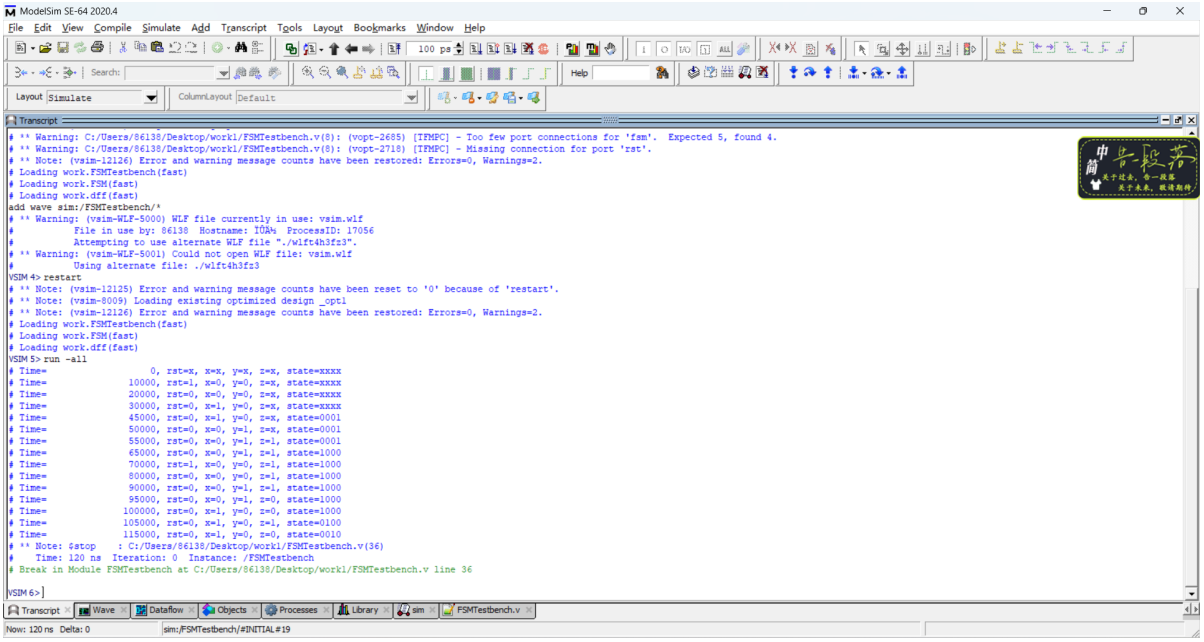
```
`timescale 1ns / 1ps  // 设置时间单位为1纳秒，时间精度为1皮秒，用于仿真环境。

module FSMTestbench;  // 定义模块FSMTestbench，这是一个测试平台模块。

reg x, y, rst, clk;  // 定义输入寄存器和时钟信号
wire z;  // 定义输出线

FSM fsm(  // 实例化有限状态机FSM，并连接端口
    .x(x),  // 输入端口x
    .y(y),  // 输入端口y
    .z(z),  // 输出端口z
    .clk(clk)  // 时钟信号
);

initial clk = 0;  // 初始化时钟信号为0
always #5 clk = ~clk;  // 每5个时间单位时钟信号取反一次

// 测试序列
initial begin
    #10 rst = 1; x = 0; y = 0;  // 10个时间单位后，设置复位信号为1，x和y为0
    #10 rst = 0;  // 10个时间单位后，取消复位信号

```

```
    // 不同组合输入并观察输出状态变化
    #10 x = 1; y = 0;  // 10个时间单位后，设置x为1，y为0
    #10 x = 1; y = 0;  // 10个时间单位后，再次设置x为1，y为0
    #10 x = 0; y = 1;  // 10个时间单位后，设置x为0，y为1
    #10 x = 0; y = 1;  // 10个时间单位后，再次设置x为0，y为1

    #10 rst = 1; x = 0; y = 0;  // 10个时间单位后，重新设置复位信号为1，x和y为0
    #10 rst = 0;  // 10个时间单位后，取消复位信号
    #10 x = 0; y = 1;  // 10个时间单位后，设置x为0，y为1
    #10 x = 1; y = 0;  // 10个时间单位后，设置x为1，y为0
    #10 x = 1; y = 0;  // 10个时间单位后，再次设置x为1，y为0
    #10 x = 0; y = 1;  // 10个时间单位后，再次设置x为0，y为1

    $stop;  // 停止仿真
end

// 监视输出和内部状态
initial begin
    $monitor("Time=%t, rst=%b, x=%b, y=%b, z=%b, state=%b", $time, rst, x, y,
z, fsm.state);
    // 使用$monitor跟踪和显示时间，复位信号，输入x，y，输出z以及状态机当前状态
end

endmodule
```

波形图



模拟输出


ModelSim SE-64 2020.4
File Edit View Compile Simulate Add Transcript Tools Layout Bookmarks Window Help
Transcript
** Warning: C:/Users/86138/Desktop/work1/FSMTestbench.v(8): (vopt-2685) [TFMPC] - Too few port connections for 'fsm'. Expected 5, found 4.
** Warning: C:/Users/86138/Desktop/work1/FSMTestbench.v(8): (vopt-2718) [TFMPC] - Missing connection for port 'rst'.
** Note: (vsim-12126) Error and warning message counts have been restored: Errors=0, Warnings=2.
Loading work.FSMTestbench(fast)
Loading work.FSM(fast)
Loading work.dff(fast)
add wave sim:/FSMTestbench/*
** Warning: (vsim-WLF-5000) WLF file currently in use: vsim.wlf
File in use by: 86138 Hostname: ÏÛÂ½ ProcessID: 17056
Attempting to use alternate WLF file "./wlft4h3fz3".
** Warning: (vsim-WLF-5001) Could not open WLF file: vsim.wlf
Using alternate file: ./wlft4h3fz3
VSIM 4> restart
** Note: (vsim-12125) Error and warning message counts have been reset to '0' because of 'restart'.
** Note: (vsim-8009) Loading existing optimized design _opt1
** Note: (vsim-12126) Error and warning message counts have been restored: Errors=0, Warnings=2.
Loading work.FSMTestbench(fast)
Loading work.FSM(fast)
Loading work.dff(fast)
VSIM 5> run -all
Time= 0, rst=x, x=x, y=x, z=x, state=xxxx
Time= 10000, rst=1, x=0, y=0, z=x, state=xxxx
Time= 20000, rst=0, x=0, y=0, z=x, state=xxxx
Time= 30000, rst=0, x=1, y=0, z=x, state=xxxx
Time= 45000, rst=0, x=1, y=0, z=x, state=0001
Time= 50000, rst=0, x=0, y=1, z=x, state=0001
Time= 55000, rst=0, x=0, y=1, z=1, state=0001
Time= 65000, rst=0, x=0, y=1, z=1, state=1000
Time= 70000, rst=1, x=0, y=0, z=1, state=1000
Time= 80000, rst=0, x=0, y=0, z=1, state=1000
Time= 90000, rst=0, x=0, y=1, z=1, state=1000
Time= 95000, rst=0, x=0, y=1, z=0, state=1000
Time= 100000, rst=0, x=1, y=0, z=0, state=1000
Time= 105000, rst=0, x=1, y=0, z=1, state=0100
Time= 115000, rst=0, x=1, y=0, z=0, state=0010
** Note: $stop : C:/Users/86138/Desktop/work1/FSMTestbench.v(36)
Time: 120 ns Iteration: 0 Instance: /FSMTestbench
Break in Module FSMTestbench at C:/Users/86138/Desktop/work1/FSMTestbench.v line 36
VSIM 6>
Transcript Wave Dataflow Objects Processes Library sim FSMTestbench.v
Now: 120 ns Delta: 0
sim:/FSMTestbench/#INITIAL#19