# 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 z,           // 定义输出变量z，不使用reg类型，直接通过D触发器控制
    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 z_internal;      // 通过D触发器控制的内部z输出
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

```

```
// 状态寄存器和输出z的D触发器实例化
dff state_reg(.d(nextstate), .clk(clk), .rst(rst), .q(state), .en(1'b1));
dff z_reg(.d(z_internal), .clk(clk), .rst(rst), .q(z), .en(1'b1));

// 组合逻辑块
always @* begin
    nextstate = state; // 默认保持当前状态
    z_internal = z;    // 默认保持当前输出
    case (state)  // 根据当前状态进行处理
        S0: begin
            if (inxy == 2'b10) begin
                nextstate = S2;
                z_internal = 0;
            end else if (inxy == 2'b01) begin
                nextstate = S3;
                z_internal = 1;
            end
        end
        S1: begin
            if (inxy == 2'b10) begin
                nextstate = S2;
                z_internal = 1;
            end else if (inxy == 2'b01) begin
                nextstate = S3;
                z_internal = 0;
            end
        end
        S2: begin
            if (inxy == 2'b10) begin
                nextstate = S1;
                z_internal = 0;
            end else if (inxy == 2'b01) begin
                nextstate = S3;
                z_internal = 1;
            end
        end
        S3: begin
            if (inxy == 2'b10) begin
                nextstate = S1;
                z_internal = 1;
            end else if (inxy == 2'b01) begin
                nextstate = S2;
                z_internal = 0;
            end
        end
        default: begin
            nextstate = S0; // 默认状态为S0
            z_internal = 0;
        end
    endcase
end

endmodule
```

## dff.v

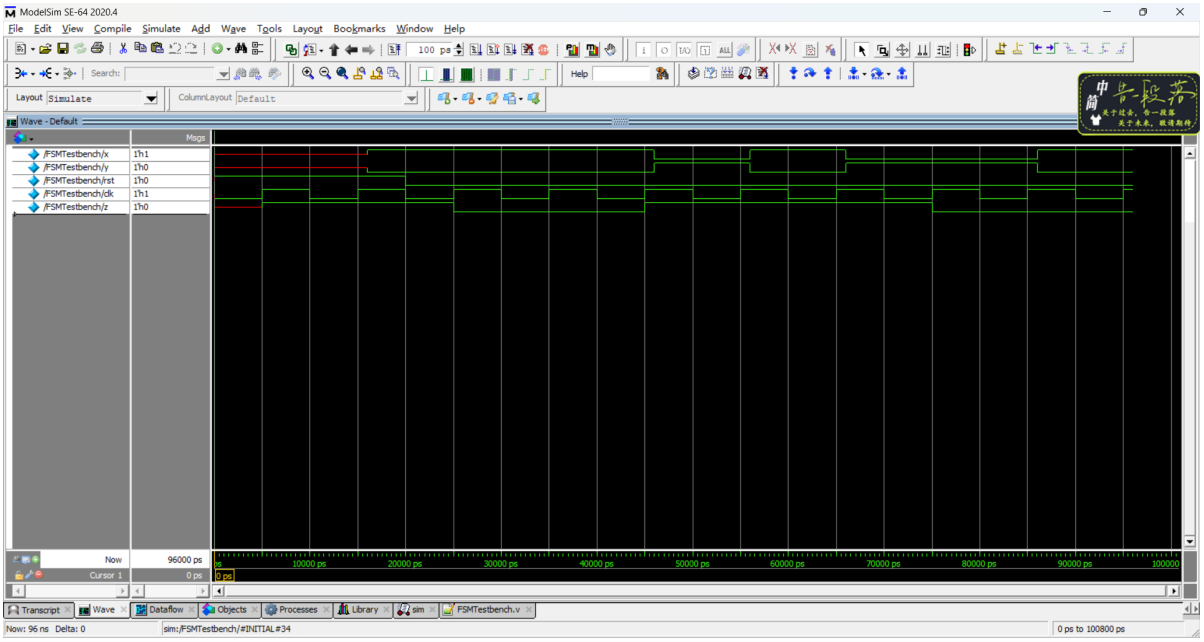
```
`timescale 1ns / 1ps  // 指定仿真的时间单位为1纳秒，时间精度为1皮秒。

// 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

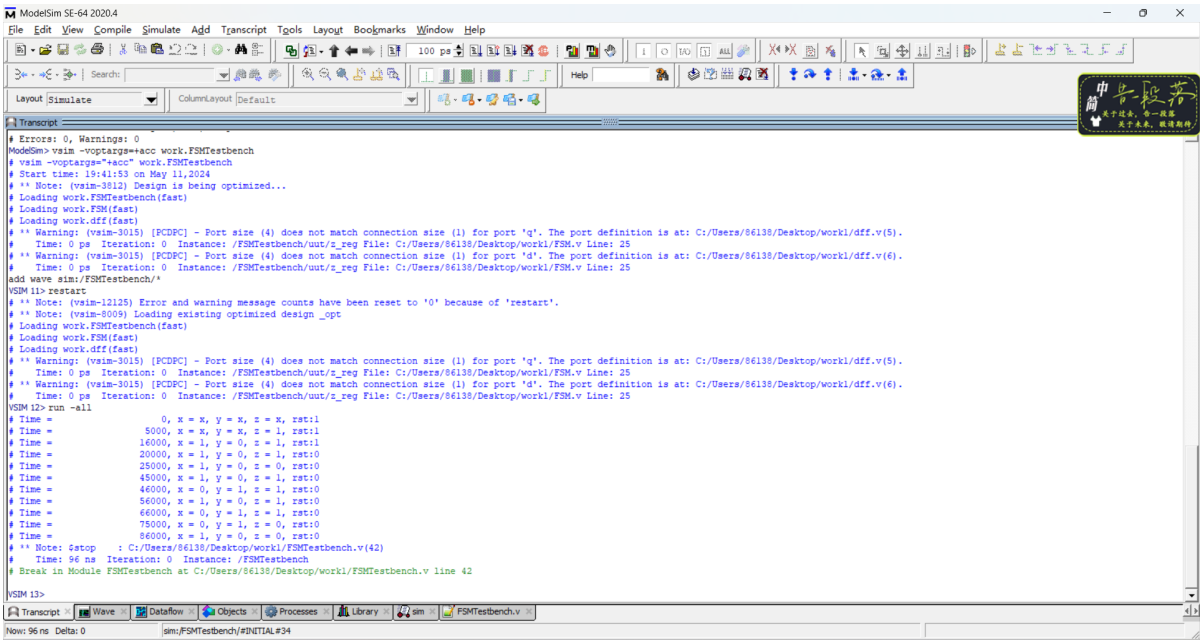
```
`timescale 1ns / 1ps  // 设置仿真的时间单位为1纳秒，时间精度为1皮秒

module FSMTestbench;
  reg x, y, rst, clk;
  wire z;

  // 实例化状态机模块
  FSM uut (
    .z(z),
    .x(x),
    .y(y),
    .rst(rst),
    .clk(clk)
  );

  // 初始化输入并启动仿真
  initial #2000 $finish;  // 在2000个仿真时间单位后结束仿真

  // 时钟信号生成
  initial begin
    clk = 0;                // 初始化时钟为低
    forever #5 clk = ~clk; // 每5个时间单位翻转时钟信号
  end

  // 复位信号控制
  initial begin
    rst = 1;    // 初始置复位为高
    #20 rst = 0;  // 20个时间单位后释放复位
    #150 rst = 1; // 150个时间单位后再次复位
```

```
    #10 rst = 0;  // 10个时间单位后释放复位
  end

  // 输入信号序列
  initial begin
    #16  {x,y} = 2'b10; // 切换到状态S2
    #30  {x,y} = 2'b01; // 切换到状态S3
    #10  {x,y} = 2'b10; // 切换到状态S1
    #10  {x,y} = 2'b01; // 切换到状态S3
    #10  {x,y} = 2'b01; // 切换到状态S2
    #10  {x,y} = 2'b10; // 切换到状态S1
    #10  {x,y} = 2'b10; // 维持在状态S1
    $stop;
  end

  // 监视信号变化
  initial begin
    $monitor("Time = %t, x = %b, y = %b, z = %b, rst:%b", $time, x, y, z,
rst);
  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
# Errors: 0, Warnings: 0
ModelSim> vsim -voptargs=+acc work.FSMTestbench
# vsim -voptargs="+acc" work.FSMTestbench
# Start time: 19:41:53 on May 11,2024
# ** Note: (vsim-3812) Design is being optimized...
# Loading work.FSMTestbench(fast)
# Loading work.FSM(fast)
# Loading work.dff(fast)
# ** Warning: (vsim-3015) [PCDPC] - Port size (4) does not match connection size (1) for port 'q'. The port definition is at: C:/Users/86138/Desktop/work1/dff.v(5).
# Time: 0 ps Iteration: 0 Instance: /FSMTestbench/uut/z_reg File: C:/Users/86138/Desktop/work1/FSM.v Line: 25
# ** Warning: (vsim-3015) [PCDPC] - Port size (4) does not match connection size (1) for port 'd'. The port definition is at: C:/Users/86138/Desktop/work1/dff.v(6).
# Time: 0 ps Iteration: 0 Instance: /FSMTestbench/uut/z_reg File: C:/Users/86138/Desktop/work1/FSM.v Line: 25
add wave sim:/FSMTestbench/*
VSIM 11> restart
# ** Note: (vsim-12125) Error and warning message counts have been reset to '0' because of 'restart'.
# ** Note: (vsim-8009) Loading existing optimized design _opt
# Loading work.FSMTestbench(fast)
# Loading work.FSM(fast)
# Loading work.dff(fast)
# ** Warning: (vsim-3015) [PCDPC] - Port size (4) does not match connection size (1) for port 'q'. The port definition is at: C:/Users/86138/Desktop/work1/dff.v(5).
# Time: 0 ps Iteration: 0 Instance: /FSMTestbench/uut/z_reg File: C:/Users/86138/Desktop/work1/FSM.v Line: 25
# ** Warning: (vsim-3015) [PCDPC] - Port size (4) does not match connection size (1) for port 'd'. The port definition is at: C:/Users/86138/Desktop/work1/dff.v(6).
# Time: 0 ps Iteration: 0 Instance: /FSMTestbench/uut/z_reg File: C:/Users/86138/Desktop/work1/FSM.v Line: 25
VSIM 12> run -all
# Time = 0, x = x, y = x, z = x, rst:1
# Time = 5000, x = x, y = x, z = 1, rst:1
# Time = 16000, x = 1, y = 0, z = 1, rst:1
# Time = 20000, x = 1, y = 0, z = 1, rst:0
# Time = 25000, x = 1, y = 0, z = 0, rst:0
# Time = 45000, x = 1, y = 0, z = 1, rst:0
# Time = 46000, x = 0, y = 1, z = 1, rst:0
# Time = 56000, x = 1, y = 0, z = 1, rst:0
# Time = 66000, x = 0, y = 1, z = 1, rst:0
# Time = 75000, x = 0, y = 1, z = 0, rst:0
# Time = 86000, x = 1, y = 0, z = 0, rst:0
# ** Note: $stop : C:/Users/86138/Desktop/work1/FSMTestbench.v(42)
# Time: 96 ns Iteration: 0 Instance: /FSMTestbench
# Break in Module FSMTestbench at C:/Users/86138/Desktop/work1/FSMTestbench.v line 42
VSIM 13>
Transcript Wave Dataflow Objects Processes Library sim FSMTestbench.v
Now: 96 ns Delta: 0
sim:/FSMTestbench/#INITIAL#34