

# Lab01-2

## 有限状态机

Ma De (马德)

[made@zju.edu.cn](mailto:made@zju.edu.cn)

2022

College of Computer Science, Zhejiang University

# Course Outline

---

- 一、实验目的
- 二、实验环境
- 三、实验目标及任务

# 实验目的

---

1. 复习有限状态机的基本概念
2. 掌握有限状态机的两种模型
3. 设计有限状态机解决实际问题

# 实验环境

---

## □ 实验设备

1. 计算机（Intel Core i5以上，4GB内存以上）系统
2. Sword2.0/Sword4.0开发板
3. Xilinx VIVADO2017.4及以上开发工具

## □ 材料

无

# 实验目标及任务

---

- **目标：** 熟悉有限状态机的基本原理，掌握moore和mealy两种类型状态机，设计并实现状态机解决实际问题
- **任务：** 设计有限状态机完成序列检测器并测试

---

## ■ 状态机原理介绍

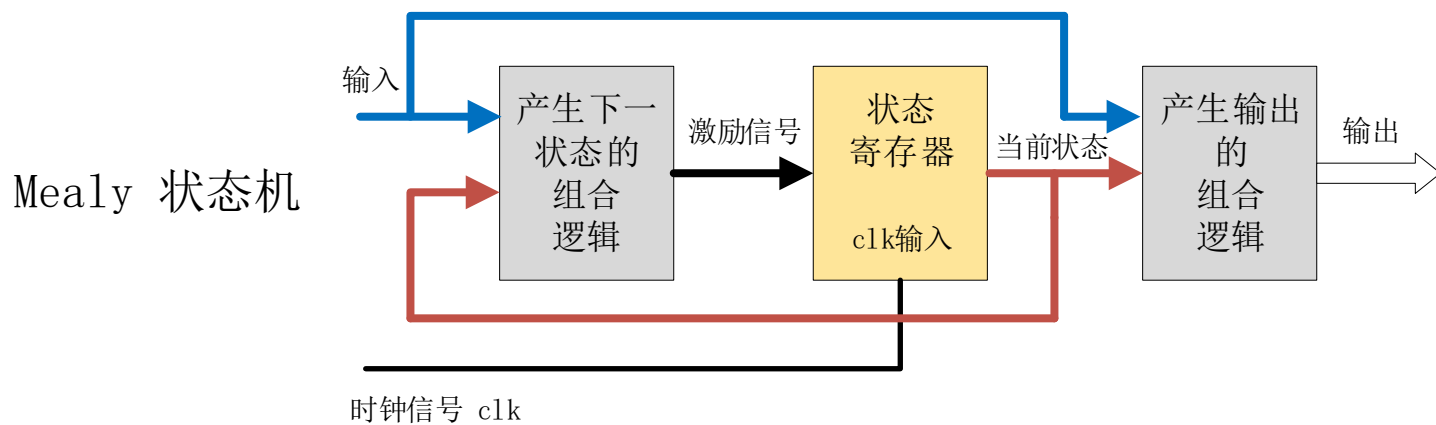
# 状态机基本概念

---

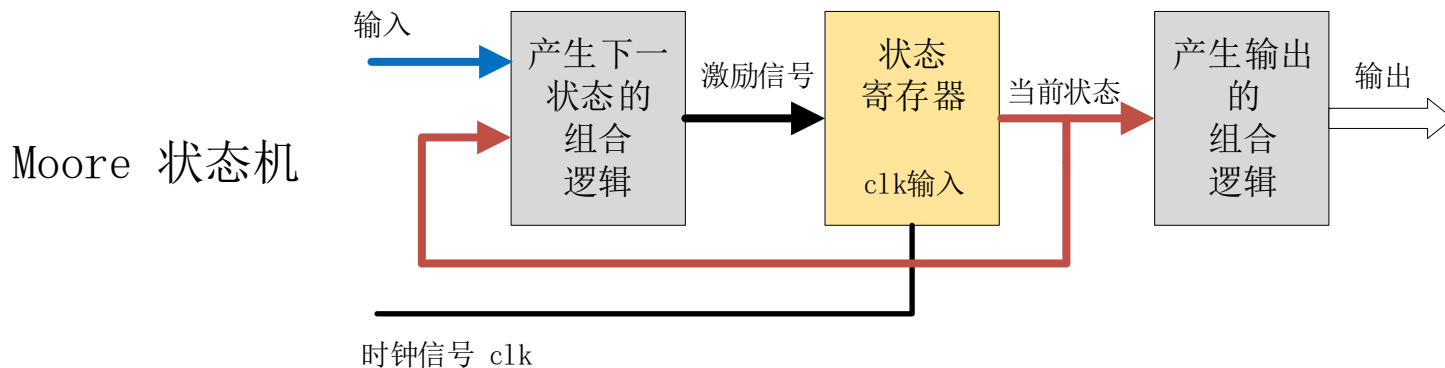
- **状态机 (State Machine)**
- 有限状态机 (Finite State Machine, 简称FSM) 在有限个状态之间按一定规律转换的时序电路。
- 有限状态机通常是由寄存器组和组合逻辑组成时序电路, 根据当前状态和输入信号可以控制下一个状态的跳转, 有限状态机在电路中通常是作为控制模块, 作为整个电路模块的核心而存在。

# 状态机基本概念

- 它主要包括两大类：**Mealy型状态机**和**Moore型状态机**。
- Mealy型状态机:其组合逻辑的输出不仅与当前状态有关，还与输入有关。

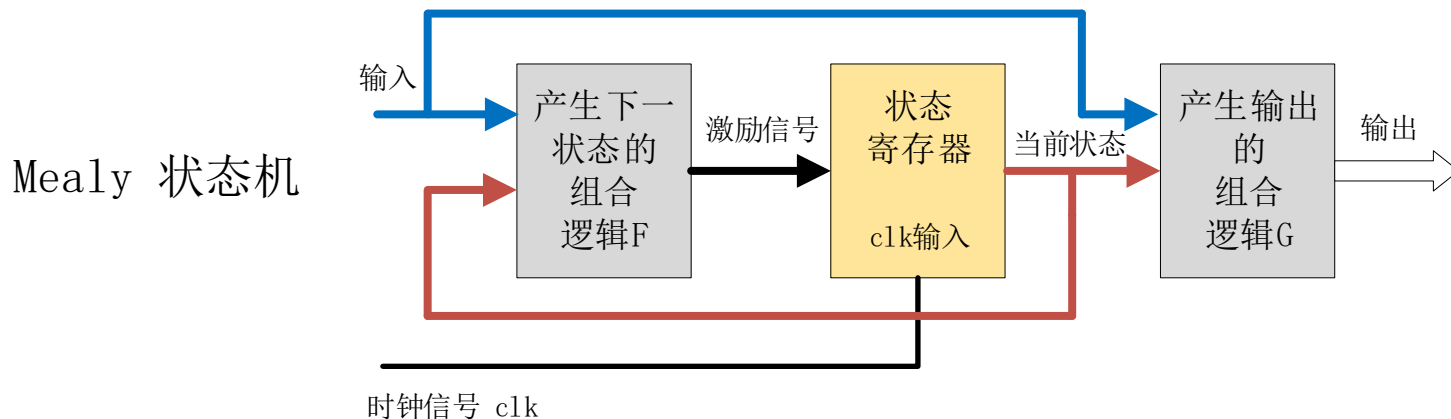


- Moore型状态机: 其组合逻辑的输出只与当前的状态有关。





# 状态机基本概念



- 状态寄存器由一组触发器组成，用来记忆状态机当前所处的状态，状态的改变只发生在时钟的跳变沿。
- 状态是否改变、如何改变，取决于组合逻辑F的输出，F是当前状态和输入信号的函数。
- 状态机的输出是由输出组合逻辑G提供的，G也是当前状态和输入信号的函数。

# 状态机设计方法

---

- 一段式描述（即状态跳转与输出信号都在同一个always块里面进行描述）
- 二段式描述（即将输出信号,与状态跳转分开描述，便于设计代码管理）
- 三段式描述（即将输出信号,与状态跳转分开描述，并且状态跳转用组合逻辑来控制）

# 状态机设计步骤

## ■ 系统架构和接口定义:

接口	接口定义
clk	系统时钟
rst_n	系统复位
X	序列输入
Y	检测输出

## ■ 状态定义和编码:

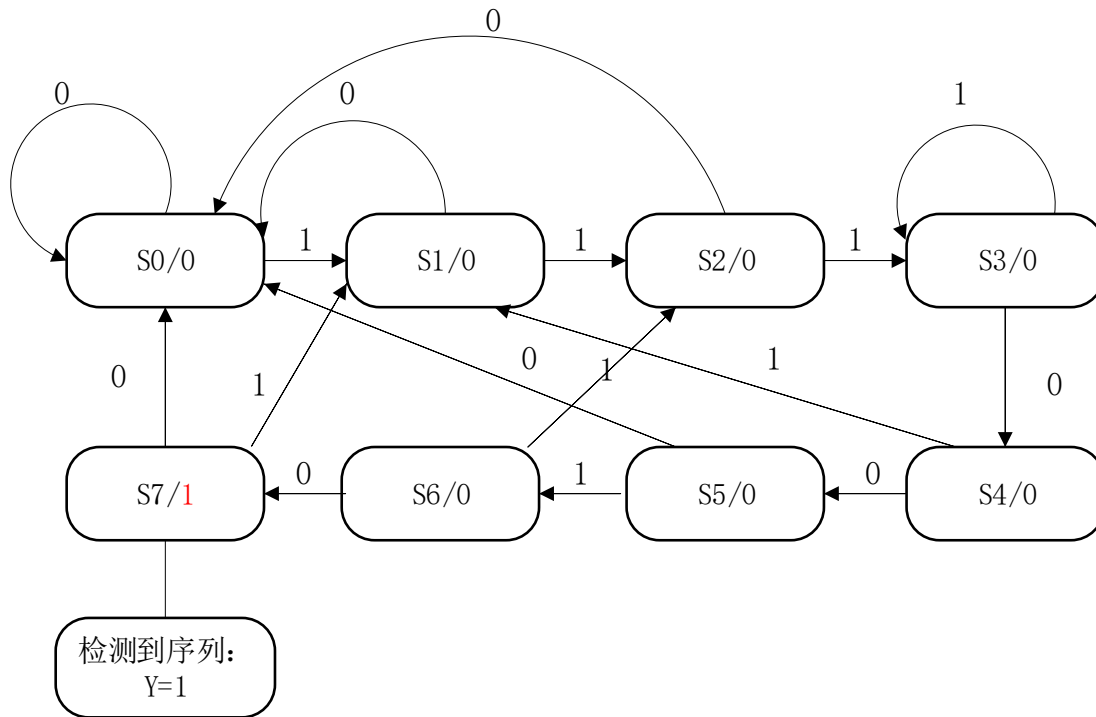
- 状态机的编码方式主要包括: 二进制码 (Binary), 格雷码 (gray), 独热码(one hot)
- 格雷码相对于二进制码而言, 在状态跳转的时候, 只有单比特翻转, 它的功耗相对较低。独热码相对于格雷码或者二进制码而言, 它增加了两个寄存器来表示状态, 但是它会更节省组合逻辑电路, 因为它在比较状态的时候, 只需要比较一个比特位, 那么其电路的速度和可靠性就会增加。

四个状态的编码方式表

	二进制	格雷码	独热码
S0	00	00	0001
S1	01	01	0010
S2	10	11	0100
S3	11	10	1000

# 状态机设计步骤

## ■ 状态转换图：

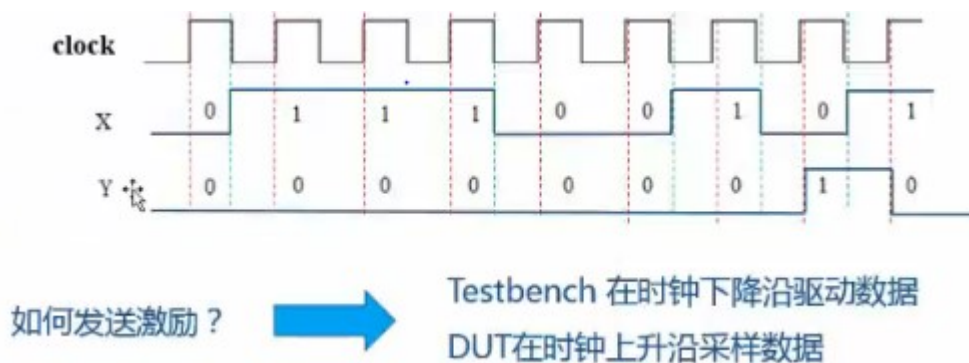


## ■ RTL实现：通过HDL语言将状态转换图进行描述实现设计

- 
- **任务：**设计状态机解决序列检测的问题（采用三段式）

# 状态机设计 -----序列检测器

- **设计要求：**用状态机设计序列检测器（1110010）
- **设计功能：**设计一个序列检测器，检测的序列为“1110010”；当输入信号X依次为“1110010”时，输出信号Y输出一个高电平，否则输出信号Y为低电平。
- **时序图：**序列检测器是一种同步时序电路，它用于搜索，检测输入的二进制代码串中是否出现指定的代码序列，1110010序列检测原理图如下：



# 状态机设计----三段式摩尔状态机

## □ 接口定义

```
module seq(  
    clk,  
    reset,  
    in,  
    out  
);  
  
input clk;  
input reset;  
input in;  
  
output out;
```

//系统时钟  
//系统复位  
//序列输入  
//检测结果输出

# 状态机设计---三段式摩尔状态机

## □ 状态定义和编码

```
//define state
parameter [2:0] S0 = 3'b000,
                S1 = 3'b001,
                S2 = 3'b010,
                S3 = 3'b011,
                S4 = 3'b100,
                S5 = 3'b101,
                S6 = 3'b110;
                S7 = 3'b111;

//internal variable
reg [2:0] curr_state;
reg [2:0] next_state;
wire out;
```



# 状态机设计----三段式摩尔状态机

## □ 第一段 ■ 状态跳转（时序逻辑）

```
//first segment:state transfer
always @(posedge clk or negedge reset)
begin
    if(!reset)
        curr_state <= S0;
    else
        curr_state <= next_state;
end
```

敏感列表：  
时钟信号以及复位信号边沿的组合

使用非阻塞赋值

# 状态机设计----三段式摩尔状态机

## □ 第二段

下个状态判断（组合逻辑）

```
//second segment:transfer condition
always @(curr_state or in) //@(*)
begin
    case(curr_state)
    S0: begin
        if(in == 0) next_state = S0;
        else       next_state = S1;
        end
    S1: begin
        if(in == 0) next_state = S0;
        else       next_state = S2;
        end
    S2: begin
        if(in == 0) next_state = S0;
        else       next_state = S3;
        end
    S3: begin
        if(in == 0) next_state = S4;
        else       next_state = S3;
        end
    S4: begin
        if(in == 0) next_state = S5;
        else       next_state = S1;
        end
    S5: begin
        if(in == 0) next_state = S0;
        else       next_state = S6;
        end
    S6: begin
        if(in == 0) next_state = S7;
        else       next_state = S2;
        end
    S7: begin
        if(in == 0) next_state = S0;
        else       next_state = S1;
        end
    default:
        next_state = S0;
    endcase
end
```

敏感信号表：  
所有的右边表达式  
中的变量以及if、  
case条件中的变量

使用阻塞赋值

If/else要配对以  
避免latch的产生

# 状态机设计---三段式摩尔状态机

---

## □ 第三段 结果输出（组合逻辑）

```
//three segment: state output  
//moore type fsm  
assign out = (curr_state == s7)?1:0;
```

# 状态机设计----三段式摩尔状态机

## □ 三段式

```
module seq(
    clk,
    reset,
    in,
    out,
);

input clk;
input reset;
input in;

output out;
//define state
parameter [2:0] S0 = 3'b000,
               S1 = 3'b001,
               S2 = 3'b010,
               S3 = 3'b011,
               S4 = 3'b100,
               S5 = 3'b101,
               S6 = 3'b110,
               S7 = 3'b111;

//internal variable
reg [2:0] curr_state;
reg [2:0] next_state;
wire out;

//first segment:state transfer
always @(posedge clk or negedge reset)
begin
    if(!reset)
        curr_state <= S0;
    else
        curr_state <= next_state;
end
```

```
//second segment:transfer condition
always @(curr_state or in) //@(*)
begin
    case(curr_state)
        S0: begin
            if(in == 0) next_state = S0;
            else        next_state = S1;
        end
        S1: begin
            if(in == 0) next_state = S0;
            else        next_state = S2;
        end
        S2: begin
            if(in == 0) next_state = S0;
            else        next_state = S3;
        end
        S3: begin
            if(in == 0) next_state = S4;
            else        next_state = S3;
        end
        S4: begin
            if(in == 0) next_state = S5;
            else        next_state = S1;
        end
        S5: begin
            if(in == 0) next_state = S0;
            else        next_state = S6;
        end
        S6: begin
            if(in == 0) next_state = S7;
            else        next_state = S2;
        end
        S7: begin
            if(in == 0) next_state = S0;
            else        next_state = S1;
        end
        default:
            next_state = S0;
    endcase
end
```

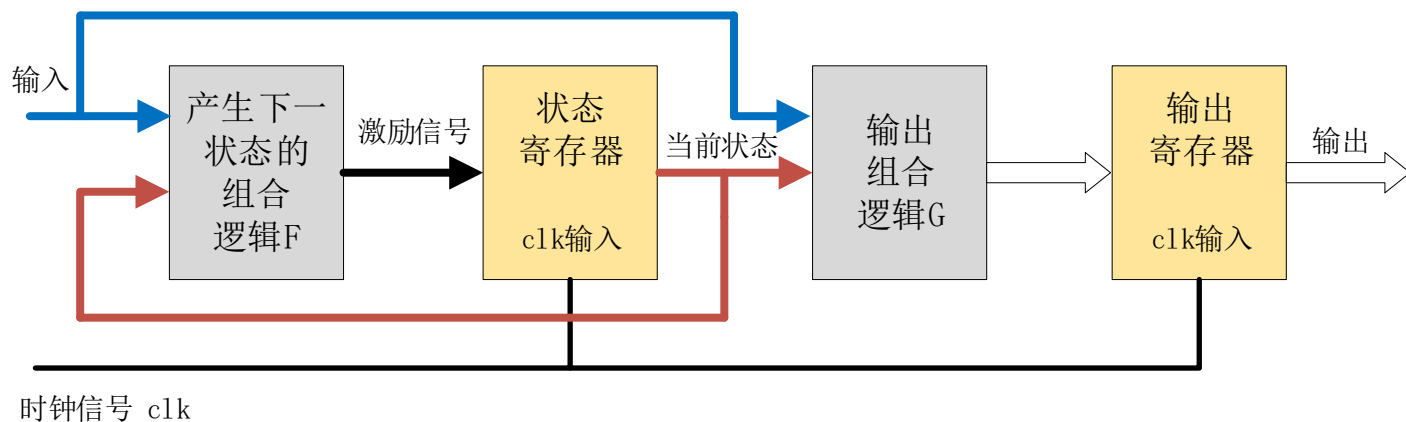
```
//three segment: state output
//moore type fsm
assign out = (curr_state == S7)?1:0;

endmodule
```

# 状态机设计---三段式状态机

□ 三段式可以在组合逻辑后再增加一级寄存器来实现时序逻辑输出：

- 1、可以有效地滤去组合逻辑输出的毛刺；
- 2、可以有效地进行时序计算与约束；
- 3、另外对于总线形式的输出信号来说，容易使总线数据对齐，从而减小总线数据间的偏移，减小接收端数据采样出错的频率。



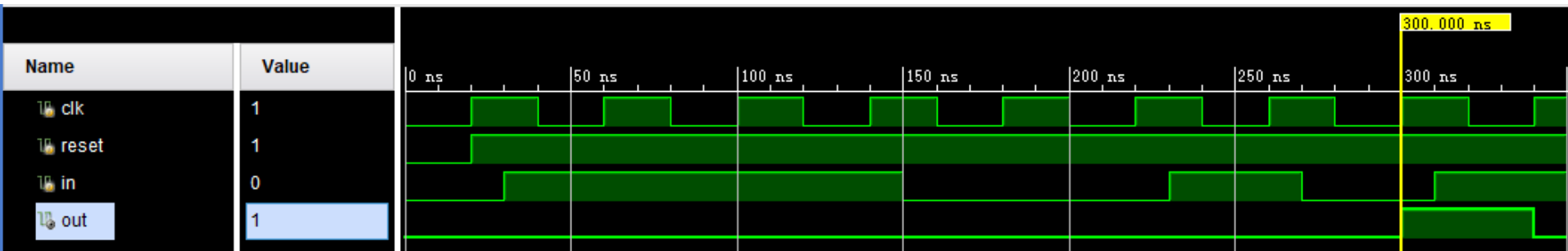
# 状态机设计----三段式状态机

## □ 测试文件

```
module tb_seq();  
    reg clk;                //011100101  
    reg reset;              initial begin  
    reg in;                  in = 0;  
    wire out;               #30 in = 1;  
                             #40 in = 1;  
                             #40 in = 1;  
always #20 clk = ~clk;     #40 in = 0;  
                             #40 in = 0;  
                             #40 in = 1;  
                             #40 in = 0;  
                             #40 in = 1;  
                             #40 $finish;  
initial begin              end  
    clk = 0;  
    reset = 0;  
    #20 reset = 1;  
end  
  
seq seq_u1(  
    .clk (clk ),  
    .reset(reset),  
    .in (in ),  
    .out (out )  
    );  
endmodule
```

# 状态机设计----三段式状态机

## □ 仿真波形



---

 **END**