# 数字逻辑设计

# 实验5 序列检测器设计

马世禹

HITSZ 实验与创新实践教育中心
Education Center of Experiments and Innovations, HITSZ

# 实验目的

（1）掌握使用状态转换图描述状态机的方法；

（2）掌握Moore型和Mealy型状态机的设计方法和两者的区别；

（3）掌握使用Verilog语言实现状态机的方法。

# 实验内容

序列检测器是一种常见的时序逻辑电路，具有重要的工程应用价值。例如，序列检测可扩展成报文检测的应用。例如：发送端不断的发送报文，接收端不断的接收数据，并根据既定的通信协议对报文进行解析。

本实验要求同学们设计一个Moore型或Mealy型的状态机，用于检测一个8位的二进制数中，是否存在“10010”的子序列。例如，二进制数8’b1001_0110含有“10010”子序列，而8’b0011_0101则不含有“10010”子序列。

# 实验内容

**详细要求如下：**

A.输入时钟为100MHz，端口为Y18；

B.使用按键开关S1作为异步复位信号，且当S1为1时，序列检测模块将被复位；

C.按键开关S2作为序列检查启动信号；

D.10010为固定的检查序列；

E.待检测序列由拨码开关SW7-SW0输入，且检查顺序从SW7到SW0；

F. 检测结果输出到LED0上；

# 实验原理

## ➢ 状态机

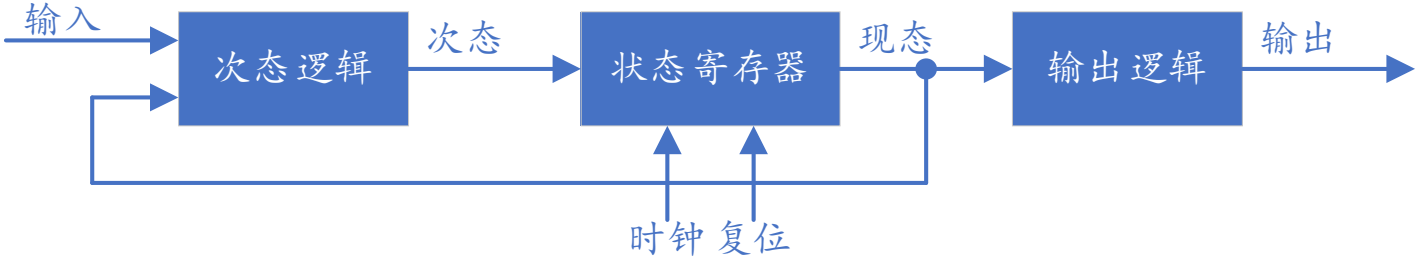
状态机的本质就是对具有逻辑顺序或时序规律事件的一种描述方法。状态机能够根据控制信号按照预先设定的状态进行状态转移，是协调相关信号动作、完成特定操作的控制中心。在CPU及其他控制电路当中，经常出现使用状态机实现控制逻辑的场景。

根据输出是否与当前输入有关，可将状态机分为2大类：Moore型状态机和Mealy型状态机。

# 实验原理

## ➢ Moore型状态机

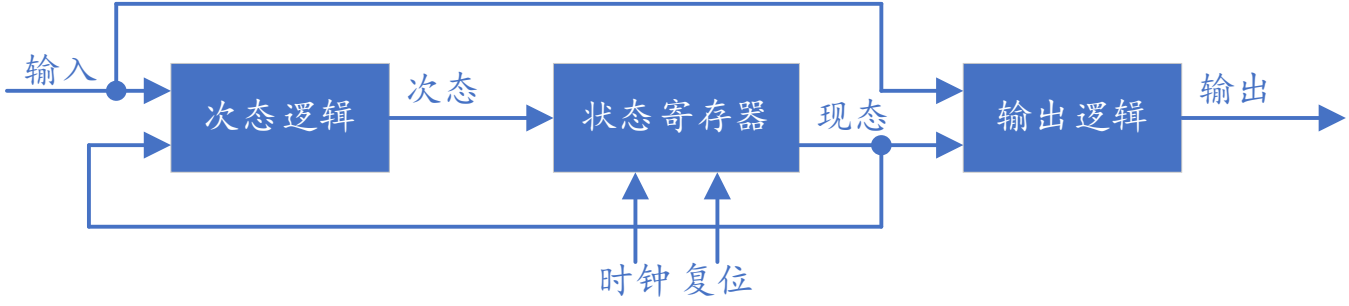
若状态机的输出只和现态有关而与当前输入无关，则称其为Moore型状态机，其原理如下图所示。



# 实验原理

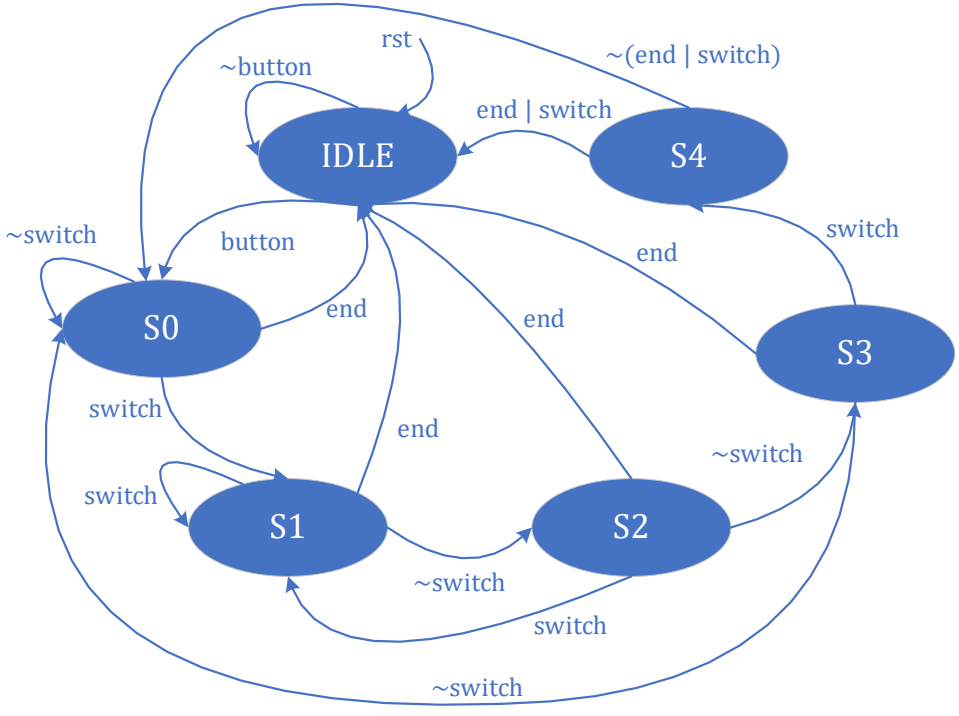
## ➢ Mealy型状态机

若状态机的输出不仅和现态有关而且和当前输入有关，则称其为Mealy型状态机，其原理如下图所示。



# 实验原理

## ➢ 状态机转换图




**状态说明**

IDLE：复位后的初始状态

S0~S4：检查序列状态

**信号说明**

rst：复位信号

button：检查启动信号

switch：依次输入的待检查序列

end：序列检查结束信号

# 实验原理

推荐：稳定性高

- 状态机描述

（1）**一段式**：整个状态机写到一个always模块里面，在该模块中既描述状态转移，又描述状态的输入和输出；

（2）**二段式**：用两个always模块来描述状态机，其中一个always模块采用同步时序描述状态转移；另一个模块采用组合逻辑判断状态转移条件，描述状态转移规律以及输出；

（3）**三段式**：在两个always模块描述方法基础上，使用三个always模块，一个always模块采用同步时序描述状态转移，一个always采用组合逻辑判断状态转移条件，描述状态转移规律，另一个always模块描述状态输出(可以用组合电路输出，也可以时序电路输出)。

# 实验原理

## ➢ 三段式描述方法示例模板

```
parameter IDLE = 3'b001;
parameter S1   = 3'b010;
parameter S2   = 3'b100;

reg [2:0] current_state;
reg [2:0] next_state   ;
reg [1:0] out          ;

//第1个always模块，描述次态寄存器迁移到现态寄存器
always @ (posedge clk or negedge rst_n) begin
    if (~rst_n) current_state <= IDLE;
    else        current_state <= next_state;
end
```

```
//第2个always模块，描述状态转移条件判断
always @ (*) begin
    case (current_state)
        IDLE : if (...) next_state = S1;
               else     next_state = IDLE;
        S1   : if (...) next_state = S2;
               else     next_state = S1;
        S2   : if (...) next_state = IDLE;
               else     next_state = S2;
        default : next_state = IDLE;
    endcase
end

//第3个always模块，描述次态寄存器输出
always @ (posedge clk or negedge rst_n) begin
    if (~rst_n) out <= 2'b00;
    else begin
        case (current_state)
            S1 : out <= 2'b01;
            S2 : out <= 2'b10;
            default : out <= 2'b00;
        endcase
    end
end
```

# 实验原理

## ➢ 接口定义

| Name | I/O | Width | Description |
|---|---|---|---|
| clk | input | 1 | 时钟信号（100MHz） |
| rst | input | 1 | 复位信号 |
| button | input | 1 | 检测序列启动信号 |
| switch | input | 8 | 输入的待检测序列 |
| led | output | 1 | LED0 显示信号 |

# 实验步骤

- 创建工程，工程名为sequence_detection；
- 编写并添加设计文件sequence_detection.v；
- 添加提供的仿真文件testbench.v，并完成仿真；
- 编写并添加约束文件，并综合实现，生成比特流；
- 将生成的比特流下载到开发板验证

# 验收要求

- 序列检测器仿真通过（1.5分）
- 序列检测器开发板实现（2分）
- 序列检测器仿真波形分析及RTL提交（0.5分）

# 提交要求

- 提交时间：详见作业提交网站
- 提交格式：学号_姓名.zip
- 注意：如有出现雷同，雷同者均不得分！

# 开始实验