中国科学技术大学
University of Science and Technology of China

数字电路实验

Lab5
时序逻辑电路

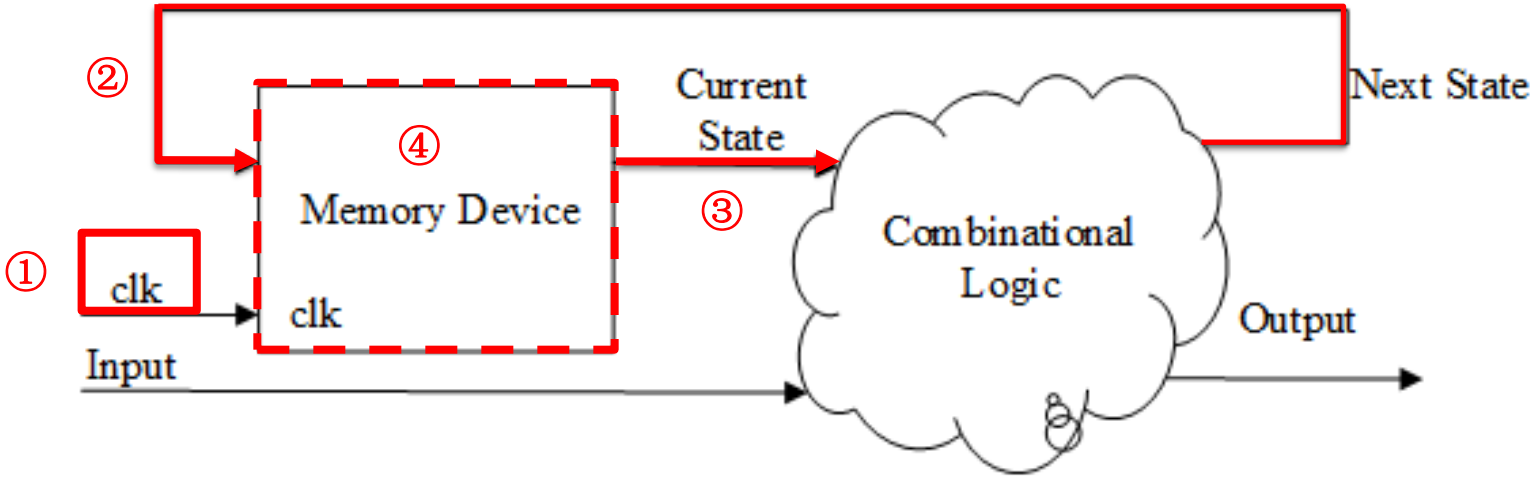
2024-10-31

实验介绍

- 在本次实验中，我们将要学习如何设计一个时序逻辑电路来解决实际问题。
- 我们将会学习一些基础的时序逻辑元件。
- 我们还会学习有限状态机的设计方法和利用有限状态机来设计时序逻辑电路

时序逻辑电路

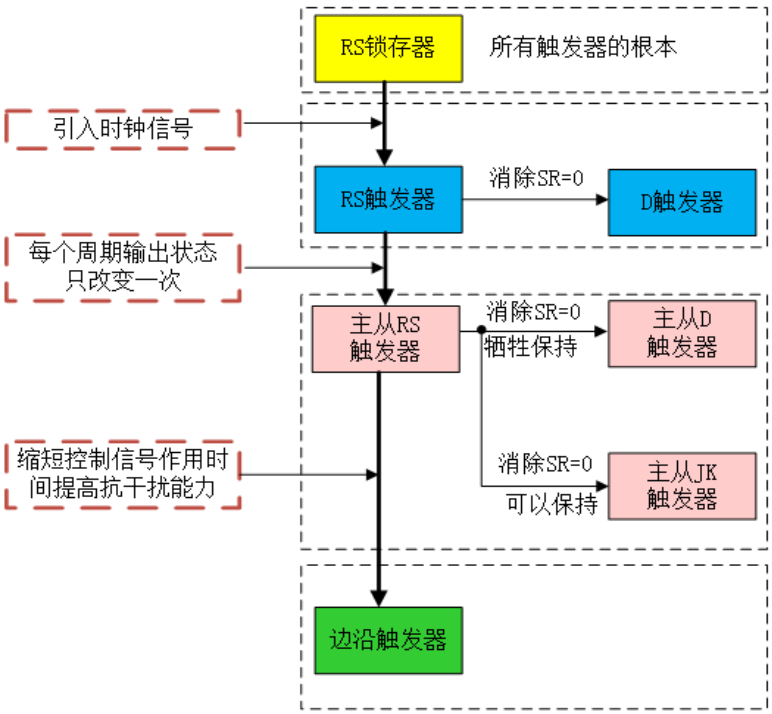
- 时序逻辑电路：时序逻辑电路是数字电路的一种，区别于组合逻辑电路，任意时刻的输出不仅取决于该时刻的输入，还与电路原来的状态有关。
- 输出受时钟信号的影响：时钟信号到达时才会计算和输出结果。



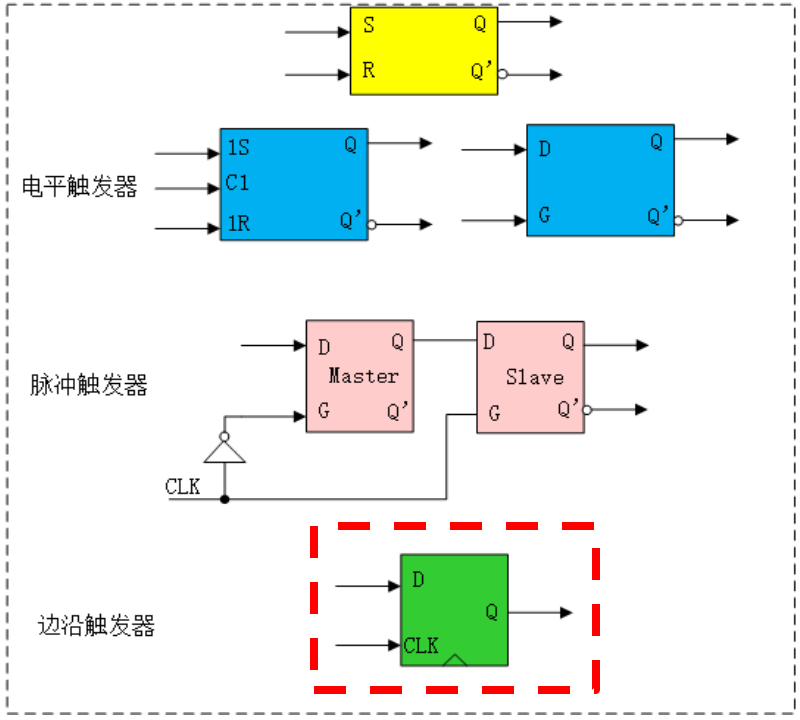
触发器

■ 如何用门电路构成理想存储器件?

触发器演变过程



触发器电路符号



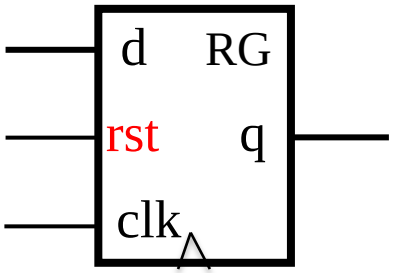
1. 引入时钟信号的锁存器叫做电平触发器;
2. 主从结构的触发器叫做脉冲触发器;
3. 边沿触发器本质是主从D触发器（脉冲触发器，利用了的D触发器没有保持的功能）。

寄存器

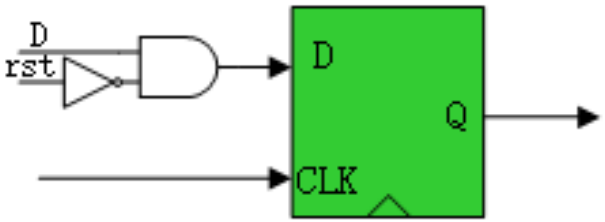
■ 寄存器本质上来说就是主从脉冲（边沿）D触发器

```
module  register
#( parameter WIDTH = 8
)(
    input  clk, rst,
    input  [WIDTH-1:0]  d,
    output reg  [WIDTH-1:0] q
);
always @(posedge clk)
    if (rst) q <= 0;
    else
        q <= d;
endmodule
```

高电平同步复位



- d, q：输入、输出数据
- clk, rst：时钟, 复位



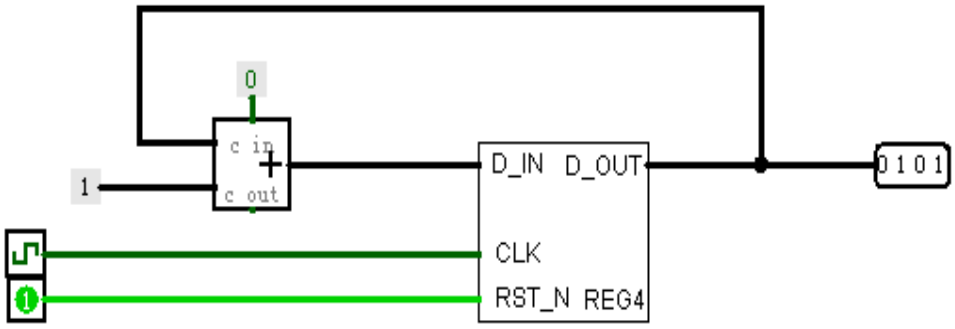
寄存器功能表

rst	clk	q	功能
1	↑	**0**	**复位**
0	↑	**d**	**置数**

简单时序逻辑电路

- 加法器为组合逻辑部件
- 寄存器为存储部件
- 两者一起构成了有实际作用的功能电路：计数器

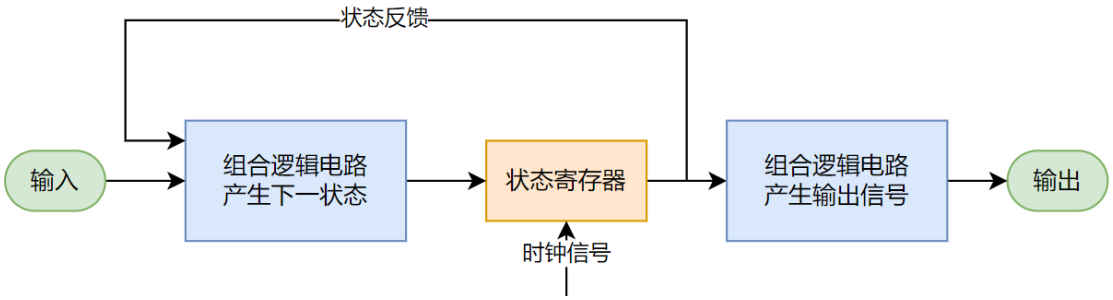
```
module CNT4(
input CLK,RST_N,
output reg [3:0] CNT);
always@(posedge CLK)
begin
        if(RST_N==0)
          CNT <= 4'b0;
        else
          CNT <= CNT + 4'b1;
end
endmodule
```



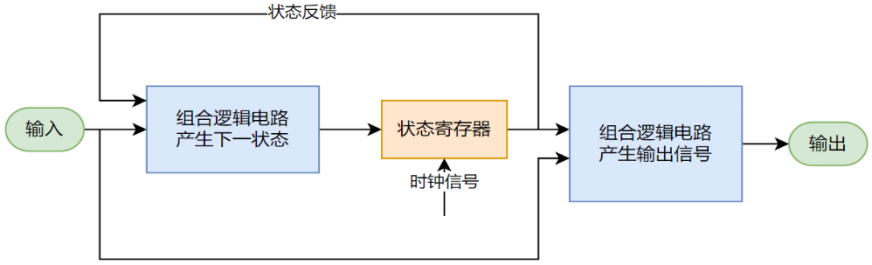
时序逻辑电路分类

- **同步时序电路与异步时序电路**
  - **同步时序电路：存储电路中所有触发器的时钟使用统一的clk，状态变化发生在同一时刻**
  - **异步时序电路：没有统一的clk,触发器状态的变化有先有后。**
- **Mealy型和Moore型**
  - **Moore型：Y = F（Q)**
  - **Mealy 型：Y = F(X,Q)**

Moore型



Mealy型



有限状态机

■ 设计FSM的基本步骤：

■ **（1）画出状态转移图**

把实际系统进行逻辑抽象，即实际问题转化为设计要求。首先确定电路输入输出引脚；然后根据实际需要列出所有的状态情况，并对状态顺序进行编号；最后根据状态转移条件画出状态转移图。

■ **（2）确定状态编码和编码方式**

编码方式的选择对所设计的电路复杂与否起着重要作用，要根据状态数目确定状态编码和编码方式。

■ **（3）给出状态方程和输出方程**

列写状态转移表，选定触发器类型，通过卡诺图化简给出状态方程和输出方程。（此步在FPGA编程中可省略）

■ **（4）编写Verilog代码**

按照步骤（1）~（2）编写具有可综合的Verilog代码。

有限状态机

Verilog 实现 FSM

- 通过分析有限状态机的结构图，我们可以发现其包含三个部分，所以可以依此编写 Verilog，接下来的代码框架也称**三段式**代码框架
- 第一部分

```
// ============================================================
// Part 1: 使用同步时序进行状态更新，即更新 current_state 的内容。
// ============================================================
always @(posedge clk) begin
    // 首先检测复位信号
    if (reset)
        current_state <= RESET_STATE;
    // 随后再进行内容更新
    else
        current_state <= next_state;
end
```

有限状态机

■ Verilog 实现 FSM

- **第二部分**

```
// ============================================================
// Part 2: 使用组合逻辑判断状态跳转逻辑，即根据 current_state 与
//         其他信号确定 next_state。
// ============================================================
// 一般使用 case + if 语句描述跳转逻辑
always @(*) begin
    // 先对 next_state 进行默认赋值，防止出现遗漏
    next_state = current_state;
    case (current_state)
        STATE_NAME_1: begin
            // ......
        end
        STATE_NAME_2: begin
            // ......
        end
        default: begin
            // ......
        end
    endcase
end
```

有限状态机

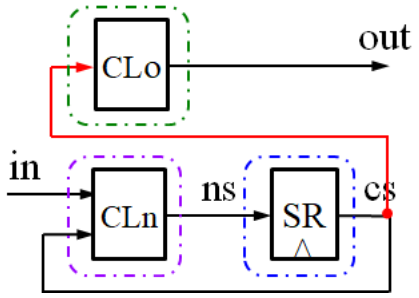
中国科学技术大学 University of Science and Technology of China

■ Verilog 实现 FSM

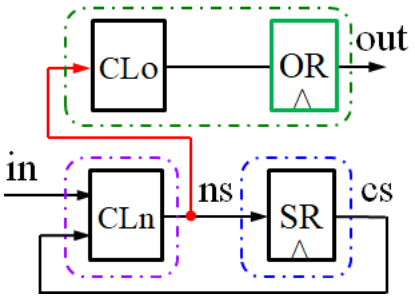
■ 第三部分

```
// ============================================================
// Part 3: 使用组合逻辑描述状态机的输出。这里是 mealy 型状态机
//         与 moore 型状态机区别的地方。
// ============================================================
// 可以直接使用 assign 进行简单逻辑的赋值
assign out1 = ......;
// 也可以用 case + if 语句进行复杂逻辑的描述
always @(*) begin
    case (current_state)
        STATE_NAME_1: begin
            // ......
        end
        STATE_NAME_2: begin
            // ......
        end
        default: begin
            // ......
        end
    endcase
end
endmodule
```

Moore型状态机



第三段为组合逻辑



第三段为时序逻辑

简单案例分析

■交通信号灯

- 要求设计一个交通信号灯，保证按绿灯黄灯红灯交替发光。
- 按照三段式代码框架，来进行代码的编写。

首先定义状态变量以及状态名称

```
reg [1:0] current_state, next_state;
localparam RED = 2'd0;
localparam YELLOW = 2'd1;
localparam GREEN = 2'd2;
```

接下来编写第一段：状态更新。假定 `reset`` 信号的效果是清除之前所有的输入，恢复初始状态。则按下 reset 后状态机应当跳转到 $GREEN$。

```
always @(posedge clk) begin
    if (reset)
        current_state <= GREEN;
    else
        current_state <= next_state;
end
```

简单案例分析

■ 交通信号灯

接下来编写第二段：状态转移。本案例中状态转换图是一个很简单的循环，根据状态转换图，我们可以编写如下的代码：

```
always @(*) begin
    next_state = current_state;
    case (current_state)
        GREEN:
            next_state = YELLOW;
        RED:
            next_state = GREEN;
        YELLOW:
            next_state = RED;
    endcase
end
```

简单案例分析

■ 交通信号灯

最后，我们编写第三段：输出。

```
assign green = current_state == GREEN;
assign yellow = current_state == YELLOW;
assign red = current_state == RED;
```

实验任务

- [必做] 了解时序逻辑电路的组成和特点；
- [必做] 学习基本的时序元件；
- [必做] 学习利用FSM来设计Verilog电路；
- [必做] 完成 Lab5 的实验练习题。

实验检查 DDL：11.7

报告提交 DDL：11.14

谢谢！