CH4 时序逻辑设计

一、引言

1. 时序逻辑电路

输出由*当前时刻输入*和*之前时刻的输入*共同决定。

时序逻辑电路有记忆。

状态：用来解释电路未来行为所需的信息。

锁存器和触发器：用于存储1比特状态的模块

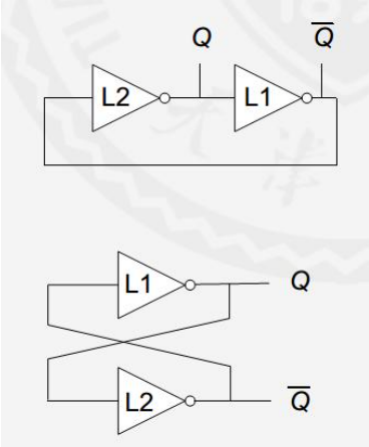
同步时序逻辑电路：一类由组合逻辑和一组表示电路状态的触发器所构成的电路。

2. 时序逻辑电路特征

- 按照一定的输入输出时序实现功能。
- 电路内部具有短期记忆
- 在输入和输出之间存在反馈回路

3. 双稳态电路

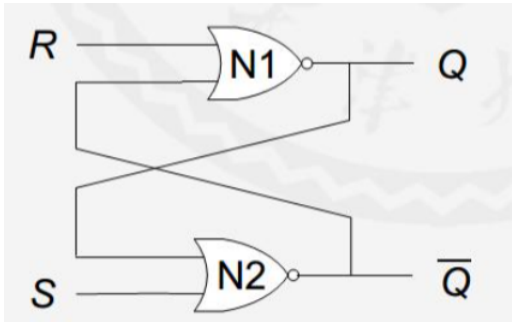
是其他存储模块的基础。



- 对称性
- 有两个输出Q和$\overline{Q}$
- 没有输入

4. SR锁存器

- 两个输入端（激励信号）
- 两个输出端（状态）



$SR = 10 \rightarrow Q = 1$ 置位

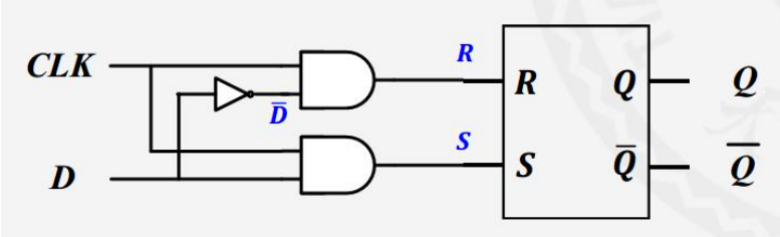
$SR = 01 \rightarrow Q = 0$ 复位

$SR = 00 \rightarrow Q^{n+1} = Q^n$ 保持

$SR = 11 \rightarrow Q^{n+1} = Q^n = 0$ 非稳态

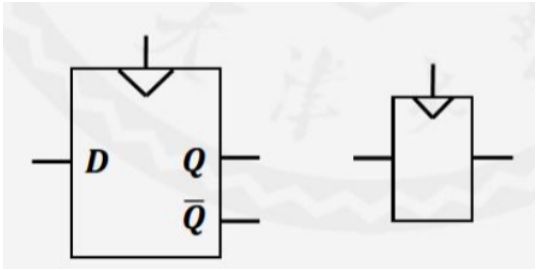
- SR锁存器是一个存储1比特状态的双稳态模块
  - SR分别表示置位（set）和复位（Reset）
- 通过控制SR信号的输入控制锁存器的状态

5. D锁存器



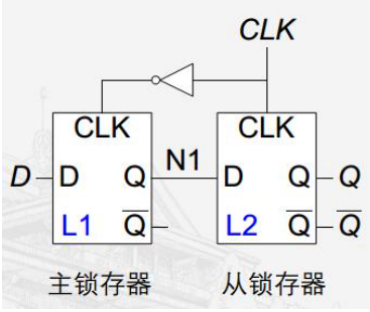
- 两个输入端：CLK，D
  - CLK：控制锁存器状态发生改变的时间
  - D：数据输入，控制下一个状态的值
- 功能
  - CLK = 1时，$Q^n = D$，Q跟随D变化，D锁存器是透明的
  - CLK = 0时，$Q^{n-1} = Q^n$，Q保持原来状态，D锁存器是不透明的
- 避免SR锁存器中$SR = 11$时的奇怪情况

6. D触发器



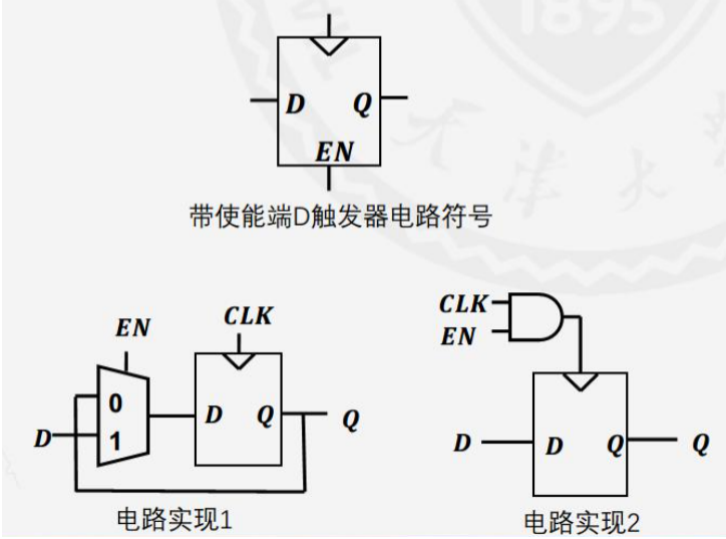
- 两个输入端：CLK, D
  - 在CLK的上升沿对D进行采样
    - CLK从0到1的瞬间，Q修改为D
    - 其他时刻，Q处于保持状态
  - Q的值只会在CLK的上升沿的时刻发生改变
- 这种工作模式称为边沿触发

主从式D触发器的实现



- 由两个顺序连接的D锁存器所构成（L1, L2）
- 由一组相反的时钟信号所控制
- 将~CLK传入L1（主锁存器），CLK传入L2（从锁存器）
- 0：N1跟随D，Q保持
- 1：Q跟随D，N1保持，一个周期后CLK复位。

带使能端的D触发器



- 三个输入端：CLK，D，EN
- EN = 1且CLK出于上升沿时Q=D
- 其余时候Q保持

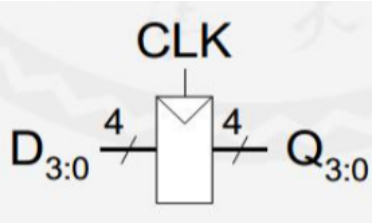
带复位功能的D触发器

- 三个输入端：CLK, D, Reset
- Reset=1：Q强制置0
- Reset=0：触发器正常工作

带置位功能的D触发器

与上面相反。

7. 寄存器



- 一个N位寄存器由一个共享的CLK输入和N个D触发器组成
- 寄存器的所有位同时被更新
- 大多数时序电路中的关键组件

二、同步逻辑设计

0. 竞争与冒险

和毛刺类似，由于不同路径信号突变导致输出发生跳变（瞬时脉冲）。

1. 几种电路

非稳态电路

- 特征
  - 非稳态电路
  - 输出结果周期性地翻转
  - 电路中具有回路，输出端反馈至输入端
- 环形振荡器

同步时序电路

- 在信号传播路径中插入寄存器以断开电路中环路
  - 使电路转变为组合逻辑电路和寄存器的集合
- 状态*同步于*时钟信号（仅在时钟沿到达时改变）
- 时钟足够慢，可以稳定信号，消除*冒险*。
- 一个时序电路包含一组有限的离散状态$\{S_0, S_1, \cdots, S_{k-1}\}$
- 同步时序电路有一个时钟输入
- 上升沿表示电路状态发生的时间
- 功能规范：描述当前状态和输入的各种组合所对应的下一个状态和输出
- 时序规范：建立时间，保持时间

2. 同步时序电路的组成规则

- 电路中的模块或是**寄存器**或是**组合逻辑电路**
- 模块中至少有一个寄存器
- 所有的寄存器都共用一个时钟信号
- 电路的每个环路中至少包含以一个寄存器

3. 常见的同步时序电路

- 流水线
- 有限状态机

D触发器是一个最简单的同步时序电路

> 包含一个输入D
>
> 一个时钟CLK
>
> 一个输出Q
>
> 两个状态{0, 1}

D触发器功能规范：下一个状态为D，输出Q为当前状态

***CLK带有三角形的是触发器，没有三角形的是锁存器。

4. 异步时序电路

- 非同步的时序电路称为异步时序电路
- 系统的时序不由时钟控制的寄存器所约束（比同步时序电路更通用）
- 同步时序电路更容易设计

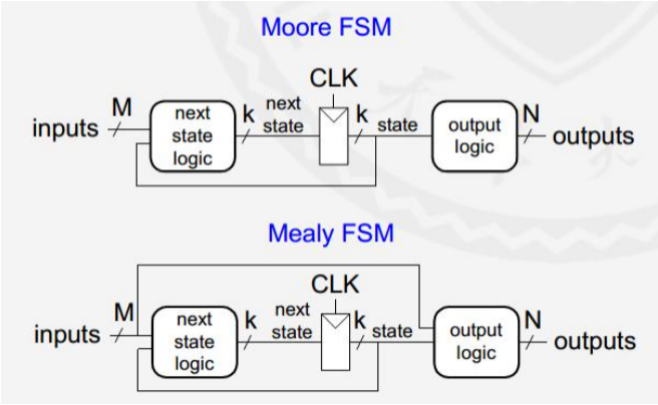
5. 有限状态机

组成：

- 状态寄存器
  - 存储当前时刻的状态
  - 状态在下一个有效时钟沿发生改变
- 组合逻辑
  - 下一个状态
  - 计算电路的输出

6. 两种不同的状态机

- 次态由现态和当前输入共同决定
  - Moore型：输出仅有当前时刻所决定
  - Mealy型：输出由当前时刻状态和输入共同决定



7. 状态转换图

- 圈圈表示状态
- 圆弧表示状态转换
- 圆弧上的项表示实现状态转换所需要的输入
- 状态转换发生在时钟有效边沿产生的时刻
- Moore型状态机：输出信息标在状态（圈圈）中
- Mealy型状态机：输出信息标在圆弧上（/右侧）

8. 状态机设计思路

状态图→状态转换表→次态方程→原理图

9. 状态编码

- 不同状态编码和输出编码会产生不同的电路
- 没有简单方法可以找到最优化编码
    - 二进制编码
    - 独热编码

10. 两种状态机的比较

Moore	Mealy
输出标记圆圈内	输出标记在圆弧上
输入后，等待状态变化后才能得到输出	输出可以直接响应输入
相同情况下需要使用更多的状态	相同情况下需要使用更少的状态

11. 状态机的分解

将复杂的状态机分解为简单的状态机的组合，一个状态机的输出是另一个状态机的输入。

12. 电路图导出状态机

> 1. 检查电路，标明输入输出和状态位
> 2. 写出次态方程和输出方程
> 3. 列出状态表和输出表
> 4. 删去不可达的状态，以简化状态表
> 5. 给每个有效状态编码指定状态名称
> 6. 用状态名称重写状态表和输出表
> 7. 画出状态转换图

三、时序逻辑中的时序问题

1. 时序问题

必须要输出处于稳态时采样才能得到较好效果。

如果采样时刻，输出未处于稳定的状态，则会产生**亚稳态**。

2. 输入时序

- 建立时间t_{setup} =在时钟有效边沿到来**前**输入信号所需要的稳定时间
- 保持时间t_{hold} =在时钟有效边沿到来**后**输入信号所保持的稳定时间
- 孔径时间t_a =在时钟边沿附近输入信号需要保持的稳定的总时间

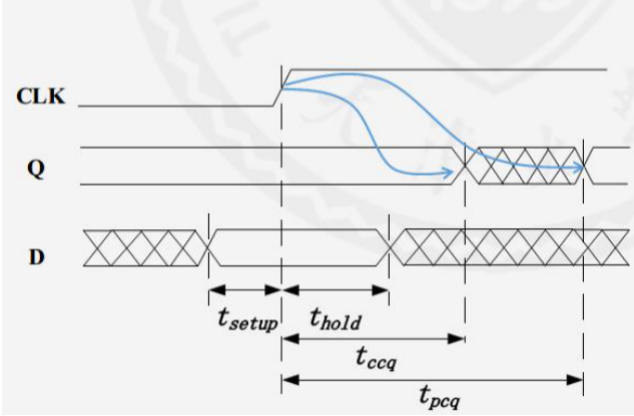
$t_a = t_{setup} + t_a$

3. 输出时序

- 传播延迟：t_{pcq} =时钟有效边沿到达后到Q最终稳定所需的最长时间
- 最小延迟：t_{ccq} =时钟有效比纳言到达后到Q开始改变所需的最短时间

4. 动态约束

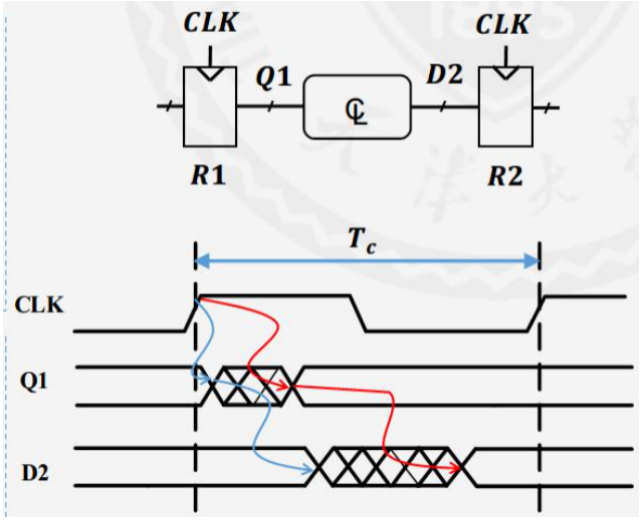
- 同步时序电路中，输入必须在时钟有效边沿附近的孔径时间内保持稳定
- 输入信号必须稳定
    - 在时钟有效边沿到达前，至少稳定t_{setup}
    - 在时钟有效边沿到达后，至少稳定t_{hold}



5. 系统时序

设时钟周期为T_c，$f_c = \frac{1}{T_c}$为时钟频率

- 提高时钟频率可以增加数字系统在单位时间内完成的工作量
- 无法无限制增加



t_{pd}为组合逻辑时延

$T_c \geq t_{pcq} + t_{setup} + t_{pd}$

输出稳定+输入前稳定+组合时延

6. 建立时间约束

$t_{pd} \leq T_c - (t_{setup} + t_{pcq})$

上式被称为**建立时间约束**或**最大延迟约束。**

T_c由研发总监和市场部提出。

$t_{setup} + t_{pcq}$为时序开销，由芯片生产工艺决定。

仅t_{pd}可以由设计人员控制

限制了组合逻辑的最大延迟。

7. 保持时间约束

保持时间约束由路径R_1至R_2的最短延迟所决定。

- 寄存器的最小延迟t_{ccq}
- 组合逻辑电路的最小延迟t_{cd}

寄存器R_2的输入信号必须在时钟上升沿后至少稳定t_{hold}时间

$t_{hold} \leq t_{ccq} + t_{cd}$

$t_{cd} \geq t_{hold} - t_{ccq}$

上式被称为**保持时间约束**或**最小延迟约束。**

- 一个可靠的触发器，保持时间比最小延迟小

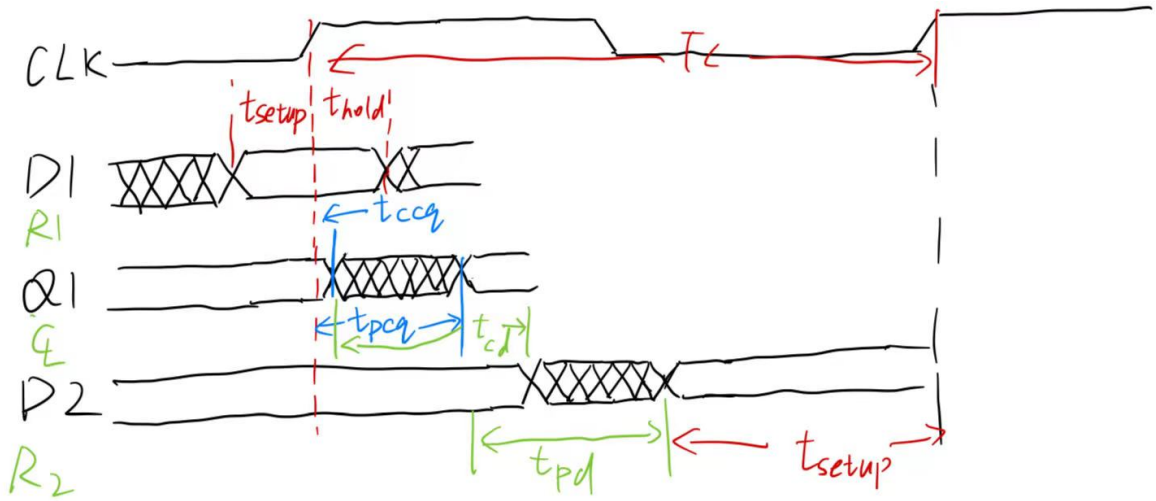
实际情况下，$t_{hold} = 0$，后续除特殊表明外，忽略保持时间约束。

***无法通过调整时钟周期解决。

8. 时序分析

几大指标：（D1表示R1输入，Q1表示R1输出，D2表示R2输入，Q2表示R2输出，Q1通过组合逻辑输出D2）

> t_{setup}：建立时间，从D1上一次结束变化到时钟上升沿（触发器的属性）
>
> t_{hold}：保持时间，从时钟上升沿到D1开始变化（触发器的属性）
>
> $t_a = t_{setup} + t_{hold}$：孔径时间：从D1上一次结束变化到D1开始变化
>
> t_{pcq}：传播延迟：从时钟上升沿到Q1结束变化
>
> t_{ccq}：最小延迟：从时钟上升沿到Q1开始变化
>
> t_{pd}：组合逻辑最大延迟：从Q1结束变化到D2结束变化
>
> t_{cd}：组合逻辑最小延迟：从Q1开始变化到Q1开始变化



$$t_{pcq} + t_{pd} + t_{setup} \leq T_c \Rightarrow T_{pd} \leq T_c - (t_{pcq} + t_{setup})$$

$$t_{ccq} + t_{cd} \geq t_{hold} \Rightarrow t_{cd} \geq t_{hold} - t_{ccq}$$

解题：使用组合逻辑的策略找到最短路径和关键路径，然后分别使用逻辑门最短延迟和最长延迟计算t_{pd}和t_{cd}

然后根据上述不等式求解结果。

9. 亚稳态

产生原因：孔径时间内采样信号不稳定。

即，没有保证建立时间和保持时间内信号不发生变化。

定义：当触发器对孔径时间内发生变化的输入进行采样时，输出可能时位于$0 \sim V_{DD}$之间进制区域内的任意一个电压值。

在最终，触发器确定输出到0或1其中一个稳态，但是，到达稳态的分辨时间是不确定的

绝对平衡：保持亚稳态

没有绝对平衡：偏向某一侧

稳态0↔亚稳态↔稳态1

每一个双稳态设备在两个稳态之间都存在着一个亚稳态。

分辨时间：设触发器在时钟周期内的随机事件发生改变，时钟有效沿出现后，触发器到达稳态所需的分辨时间为t_{res}，

- 孔径时间外改变：$t_{res} = t_{pcq}$
- 孔径时间内改变：$P(t_{res} > t) = \frac{T_0}{T_c} e^{-\frac{t}{\tau}}$（$T_c$为时钟周期，$T_o$和$\tau$由触发器属性决定）
- 等待时间足够长（超过t_{pcq}），触发器达到一个有效逻辑电平的概率将很高

四、时序逻辑模块

***不要在多于1个always语句块或者持续赋值语句中对用一个信号赋值

```
// vanilla
always_ff @(posedge sys_clk)q <= d;
// 复位功能寄存器
if(~sys_rst_n)q <= 0;
else q <= d;
// 带使能端寄存器
if(~sys_rst_n)q <= 0;
else if(en) q <= d;
else q <= q;
// 锁存器
always_latch if(clk)q <= d;
// 计数器
if(~sys_rst_n)q <= 0;
else q <= q + 1;
// 移位寄存器
if(~sys_rst_n)q <= 0;
else if(load)q <= d;
else q <= {q[N-2:0], sin};
assign sout = q[N-1];
// 有限状态机
always_ff @(posedge clk, negedge sys_rst_n) begin
    if(~sys_rst_n) cstate <= S0;
    else           cstate <= nstate;
end
always_comb begin // 次态逻辑
    // ...
end
assign q = (cstate == S0);  // 输出逻辑
```

五、存储器阵列

1. 概述

一种有效的存储大量数据的模块

每个N位地址（数据的索引）都可以读出或写入M位的数据（存储的内容）

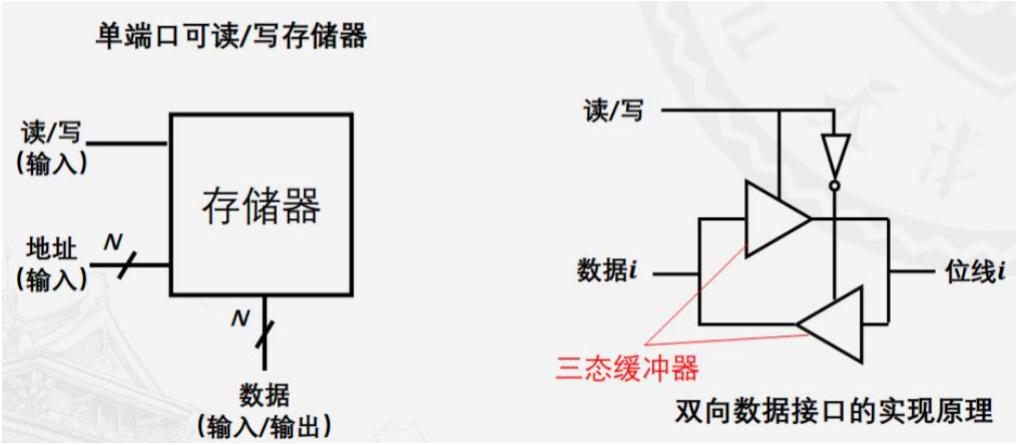
- 存储器由二维存储单元阵列构成
- 每个位单元存储1位数据
- 一个N位地址M位的数据：
  - 有2^N行和M列
  - 深度：行数
  - 宽度：列数
  - 总大小：深度×宽度
  - 字：每行数据都称为一个字

2. 存储器的组织

- 存储器阵列由位单元阵列组成
- 每个位单元存储1位数据
- 每一个位单元与一个字线（wordline）和一个位线（bitline）相连

字线： 类似于使能端，用于控制阵列中一行数据的读/写，对应着唯一的地址，同一时刻最多有一个字线为高电平

存储器端口：



3. 存储器的类型

- 随机访问存储器（RAM）：易失的
  - 动态随机访问存储器（DRAM）：计算机的主存
  - 静态随机访问存储器（SRAM）：CPU中的高速缓存
- 只读存储器（ROM ）：非易失的（也可随机访问，大多数现代ROM可读写）

4. 存储原理

DRAM：

- 数据存储在**1**个电容上
- 读操作后，存储的数据会被破坏
- 电容上存储的电荷也会慢慢泄露
- 内容需要频繁刷新（读，然后重写），所以被称为动态
- 由1个晶体管组成

SRAM：

- 数据存储在一个交叉耦合的反相器中
- 交叉耦合反相器具有很强的抗干扰能力
- 不需要刷新
- 由6个晶体管组成

5. 存储器的比较

触发器：成本3延迟1

SRAM：成本2延迟2

DRAM：成本1延迟3

ROM：读快，写慢

6. 存储器建模

```
logic [M-1:0] mem[2**N-1:0];     // M * 2^N 的数组
// RAM
always_ff @(posedge clk) if(we) mem[addr] <= din;
assign dout = mem[addr];
// ROM
always_comb case(adr) ... endcase
```

7. 寄存器文件

通常是一个小型多端口SRAM阵列。