数字电路

Digital Circuits and System

李文明

liwenming@ict.ac.cn

中国科学院计算技术研究所
INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

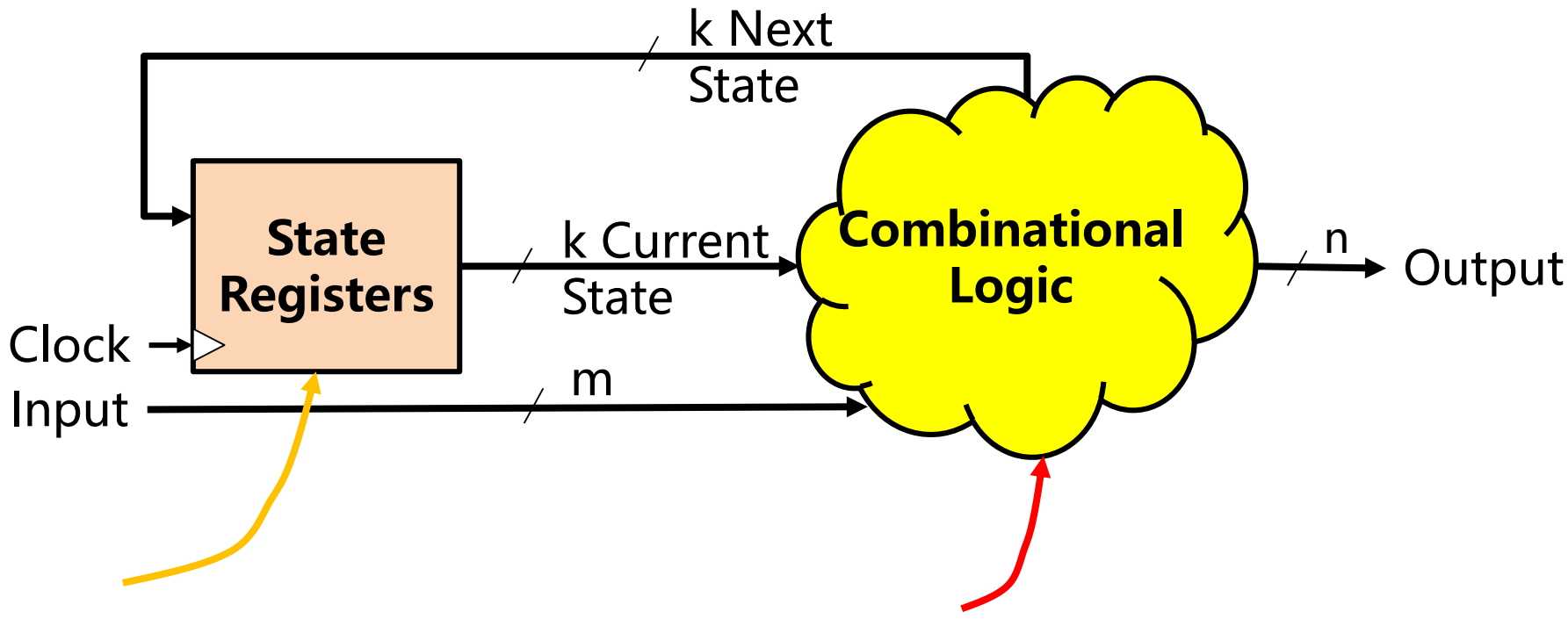
中国科学院大学
University of Chinese Academy of Sciences

有限状态机(FSM)

有限状态机

- 问题抽象
- 状态转换图
- 从状态图到电路实现
- 蚂蚁智能状态机设计
- 数字系统的多时钟域和亚稳态

一种新的数字系统



工作周期和状态是工程设计出来的

在特定的动态时序规范下工作

牢记 *k bits* 二进制，可以有 2^k 种组合状态

有向无环图

工作在静态组合逻辑约束下

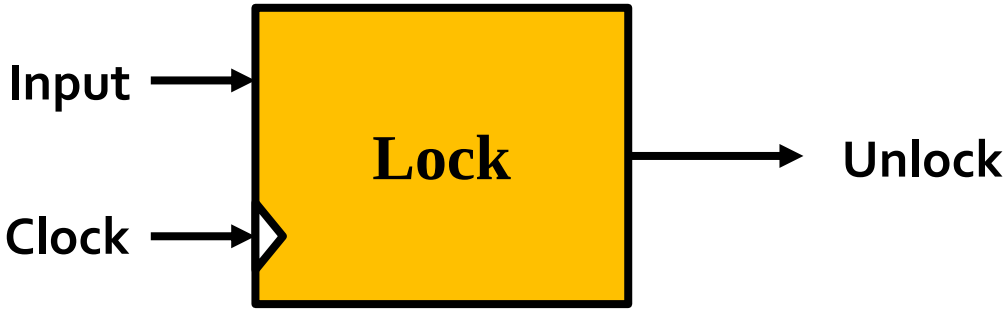
可以用2^{k+m}行真值表穷举出 n 个输出的取值

一个简单的时序逻辑电路

设计一个解锁 1bit 二进制组合序列的电路

电路规范（Specification）：

- 电路有 1 个输入（“0”或“1”）
- 电路有1个输出，解锁（“Unlock”）信号
- $Unlock = 1$，当且仅当：之前的4个输入组合序列为“*0110*”

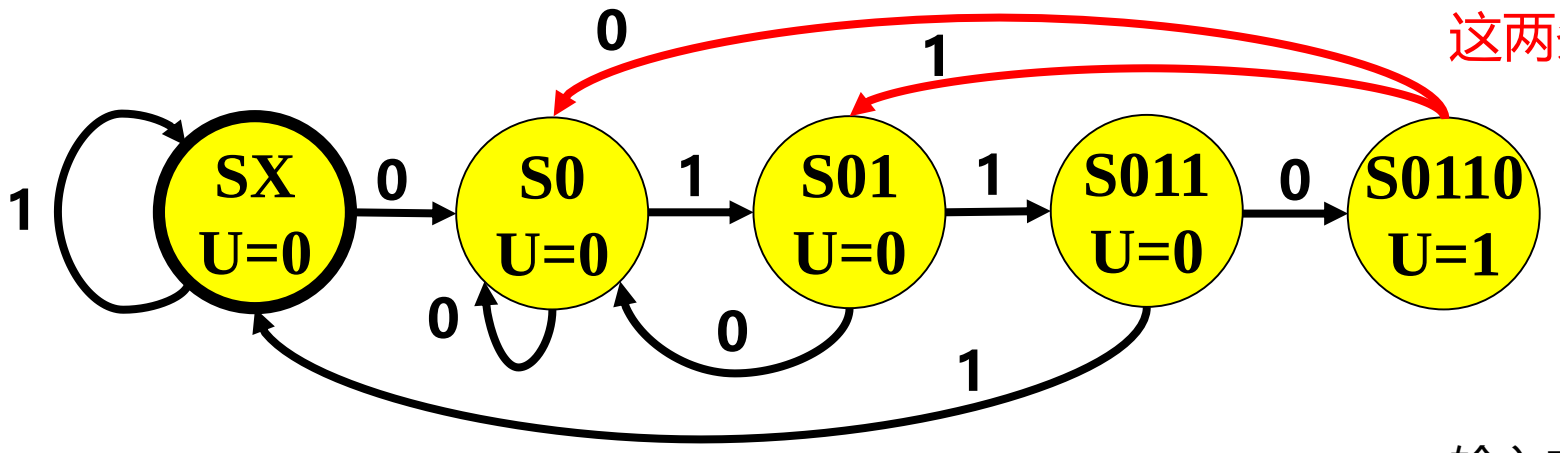


问题：这样的电路需要多少个状态?

问题抽象：有限状态机(Finite State Machine)

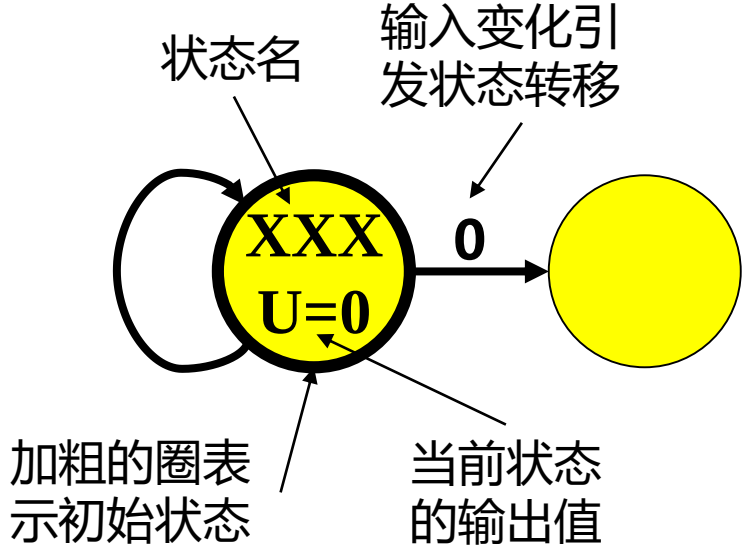
- FSM具有如下特征
- k 个状态：分别是 $S_1, S_2, \ldots, S_k$，其中某个状态是初始状态
- m 个输入，分别是 $I_1, I_2, \ldots, I_m$
- n 个输出，分别是 $O_1, O, \ldots, O_n$
- 状态转换规则：对于特定的输入 I 和当前状态 S，在下一个时钟周期的新状态为 $S'(I, S)$
- 输出规则：对于每个特定的状态 S，$Out(S)$ 或 $Out(S, I)$

状态转换图(State Transition Diagram)



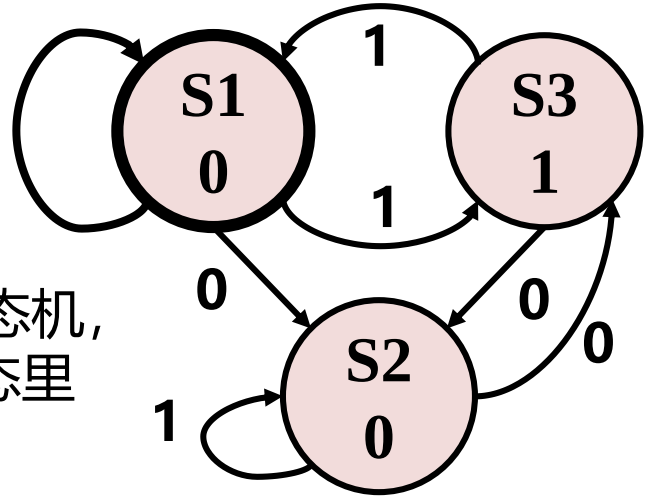
设计锁的状态转换图

- 初始状态设为 “SX”
- 对应输入序列，每一步都需要一个独立的状态
- 能够处理输入序列错误的情况

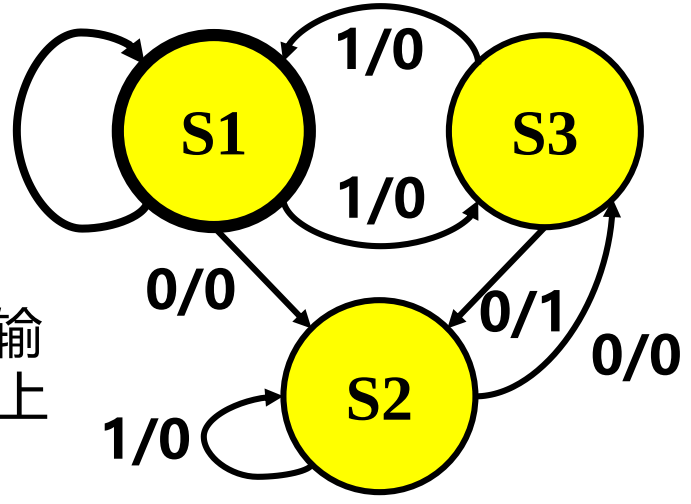


有效的状态图(Valid State Diagrams)

Moore状态机，输出在状态里



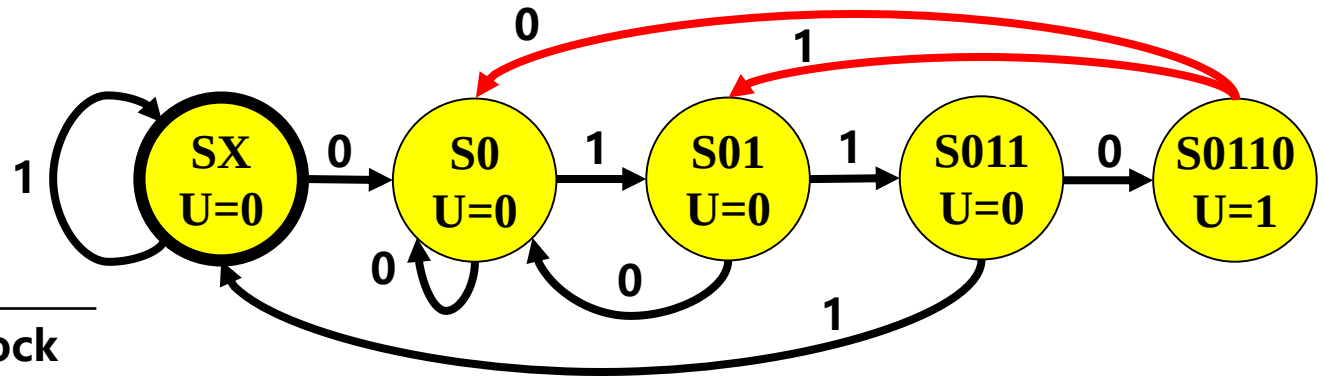
Mealy状态机，输出在状态转换线上



离开某个状态的弧线必须满足：

1. 互斥，对于给定的输入，只能引发一种可能的状态变化
2. 穷尽
   - 每个状态必须指定所有输入组合的可能性所引发的变化，
   - 如果什么都没变，意味着停留在原来的状态

状态转换图与真值表



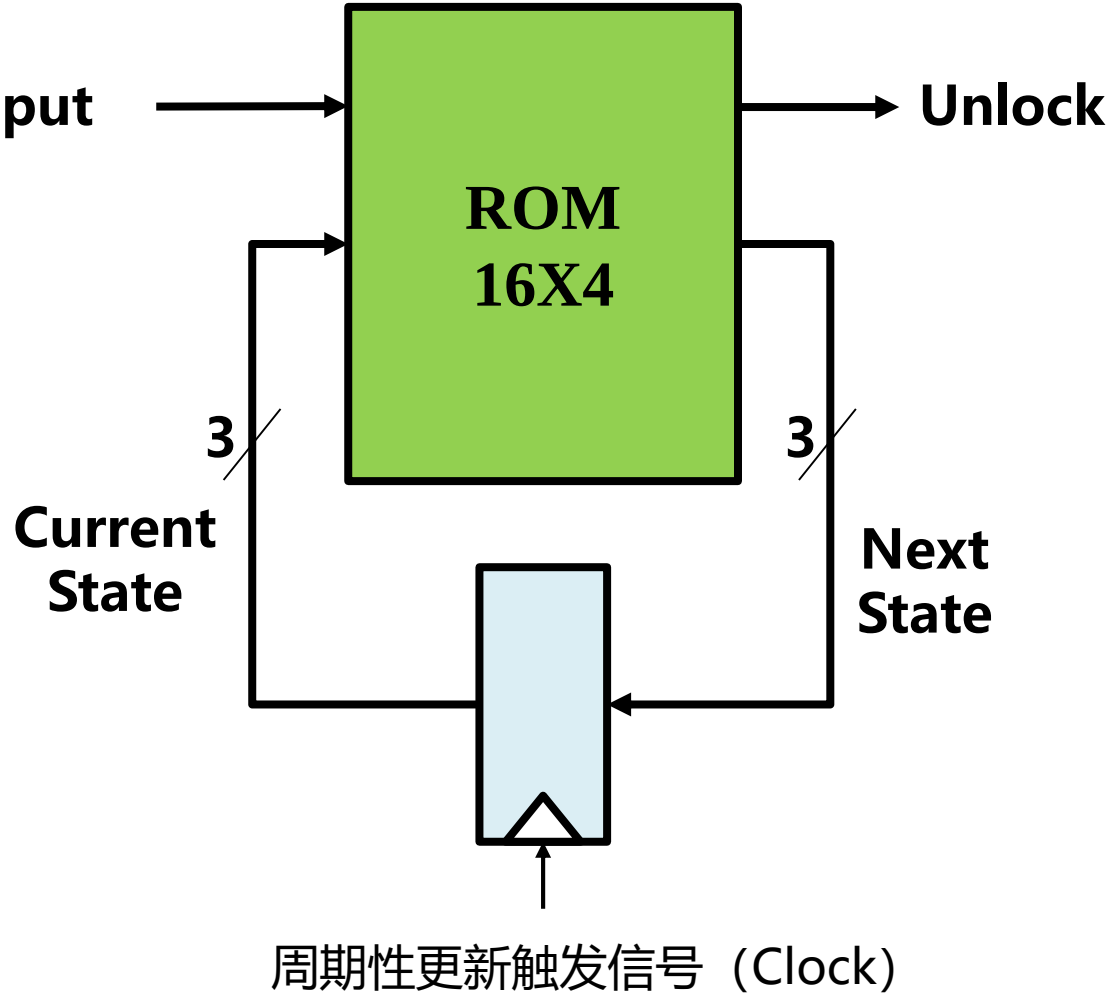
IN	Current State		Next State		Unlock
0	000	SX	S0	001	0
1	000	SX	SX	000	0
0	001	S0	S0	001	0
1	001	S0	S01	011	0
0	011	S01	S0	001	0
1	011	S01	S011	010	0
0	010	S011	S0110	100	0
1	010	S011	SX	000	0
0	100	S0110	S0	001	1
1	100	S0110	S01	011	1

- 所有的状态转换图都可以表示成真值表
- 每个状态用一组二进制编码来表示，这里有点儿小技巧
- 真值表可以用以前的组合逻辑化简方法来化简

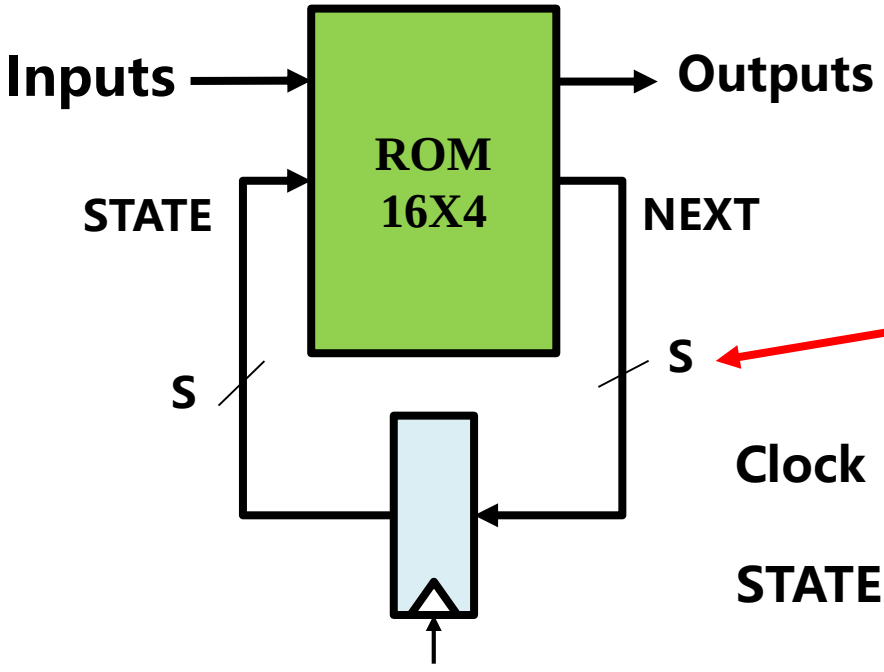
原则上，对每个状态分配的二进制编码可以随意，但如能够精心设计，能大大简化最终的电路实现

状态机的硬件实现

- 假设输入信号与时钟同步
- 4输入→2^4个地址单元
- 5个状态，需要 3bits 编码



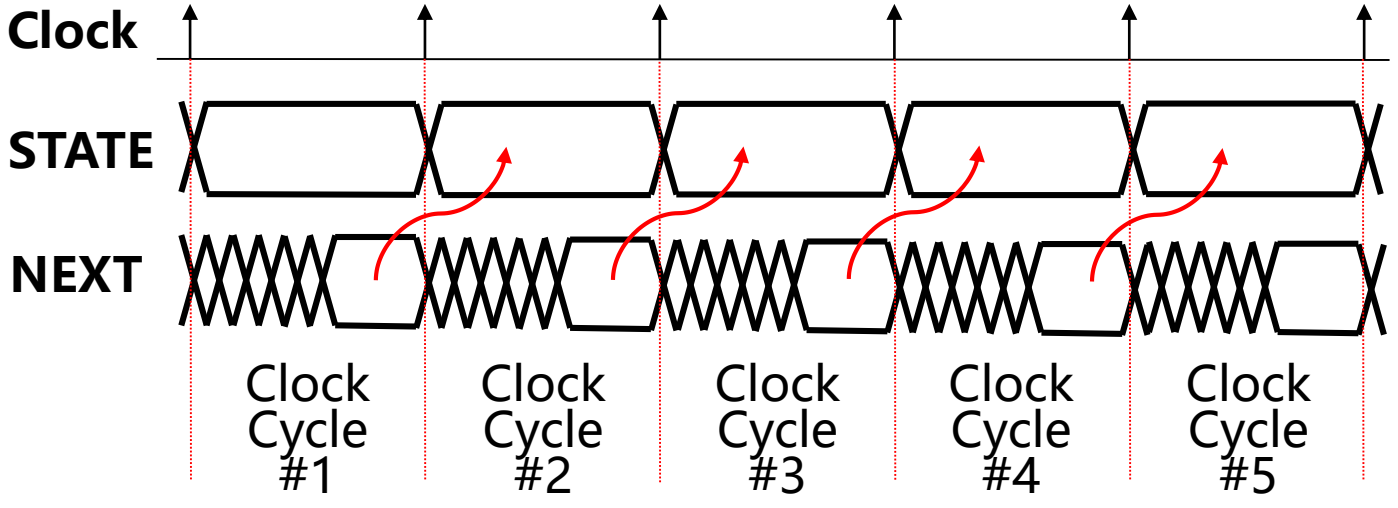
离散状态，离散时间



输出信号两种选择：

1. 输出仅依赖于“状态”（Moore）
2. 输出依赖于“状态+输入”（Mealy）

S State bits → 2^S 个可能的状态



状态机的几个基础问题

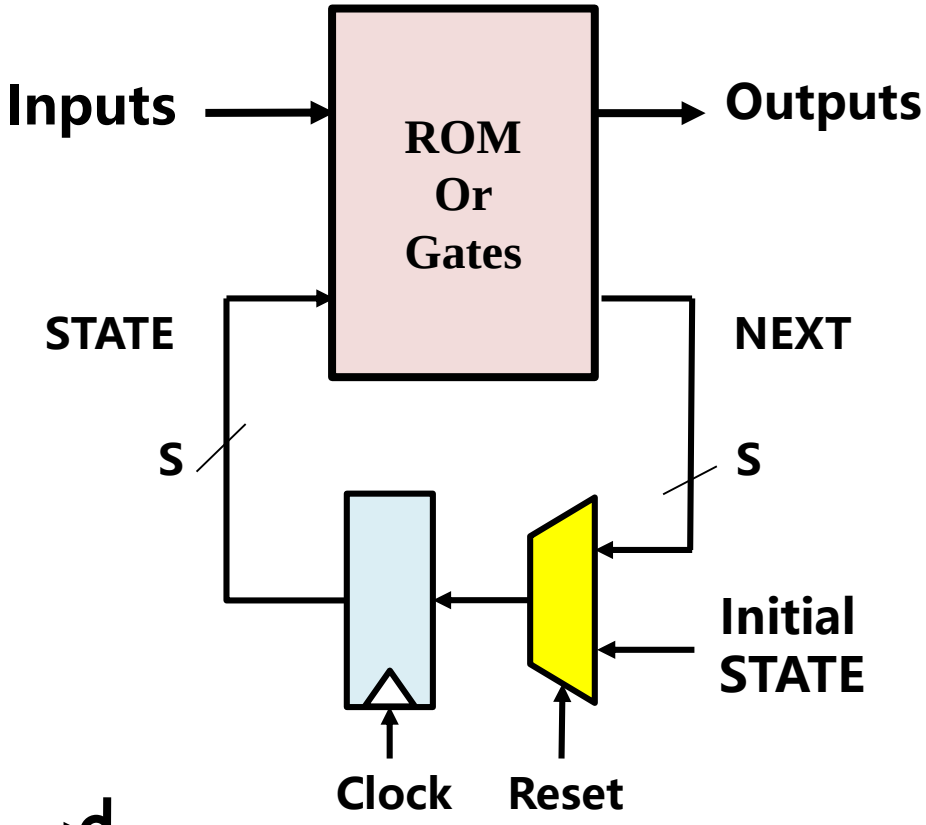
如何初始化？清除记忆的状态？

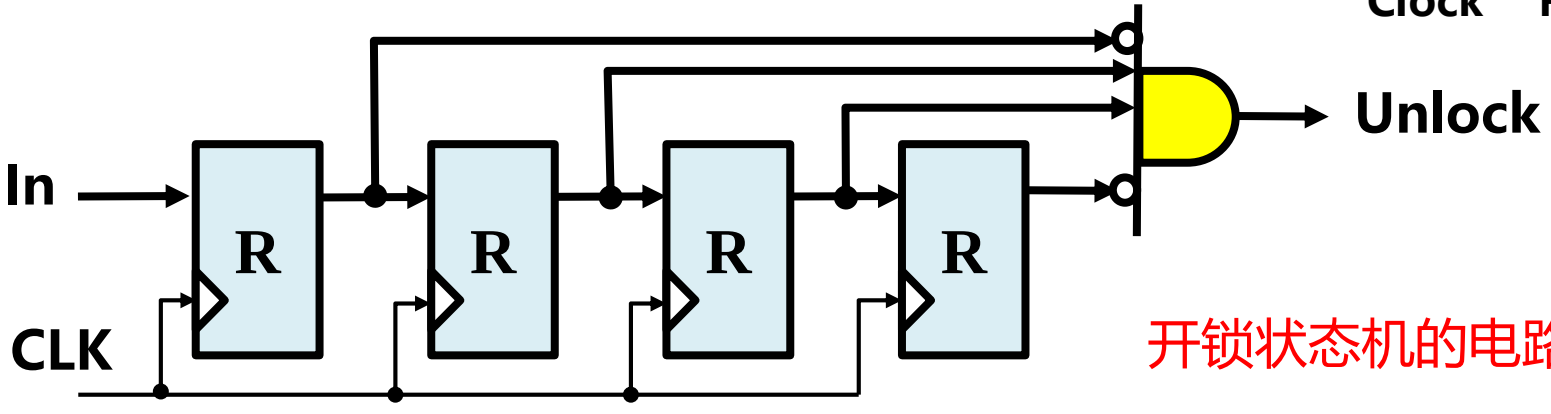
不用的状态也要编码？

- 浪费存储单元或逻辑门
- 不用的状态意味着什么？
- 状态机是否可以自恢复？

如何选择状态的编码？

如何使输入信号与状态更新同步？

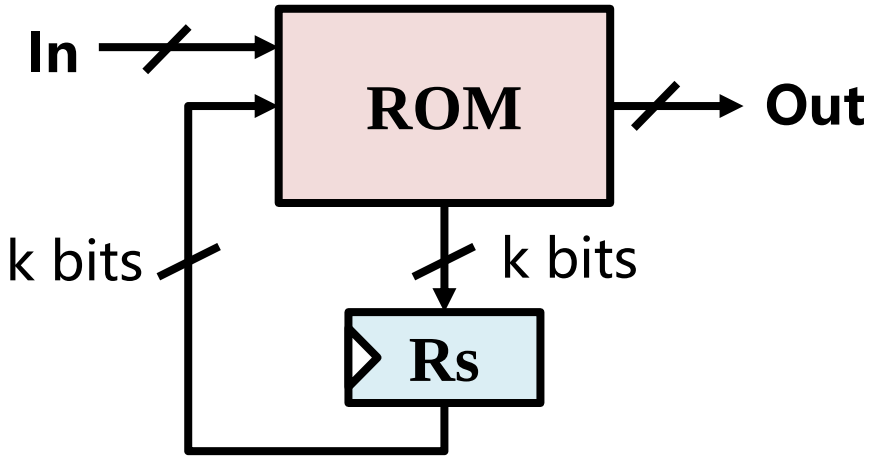




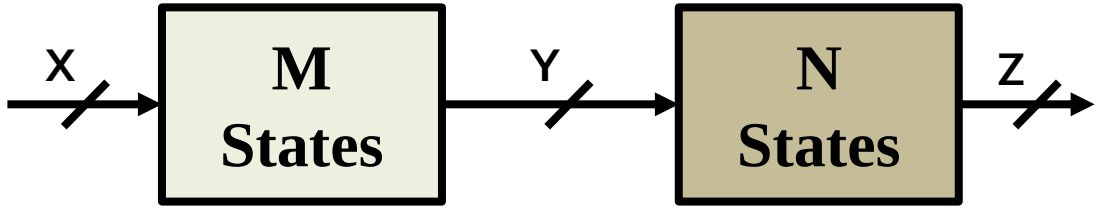
开锁状态机的电路实现

关于状态机的小测验

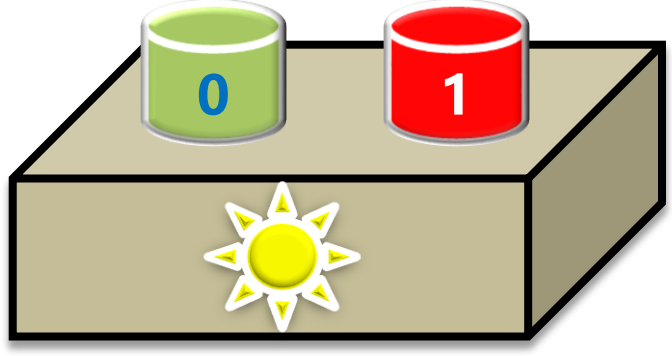
1. 如图所示状态机电路，其中包含了多少个状态?



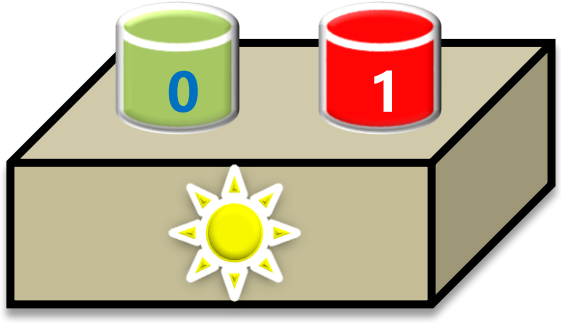
2. 如图所示状态机电路，其中包含了多少个状态?



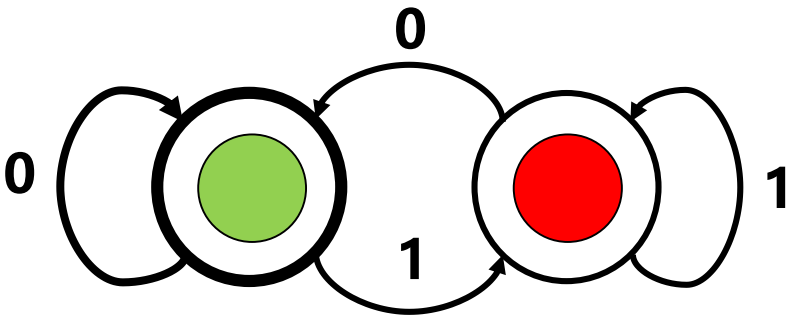
3. 如图所示状态机电路，是否可以找出其状态转换规则?



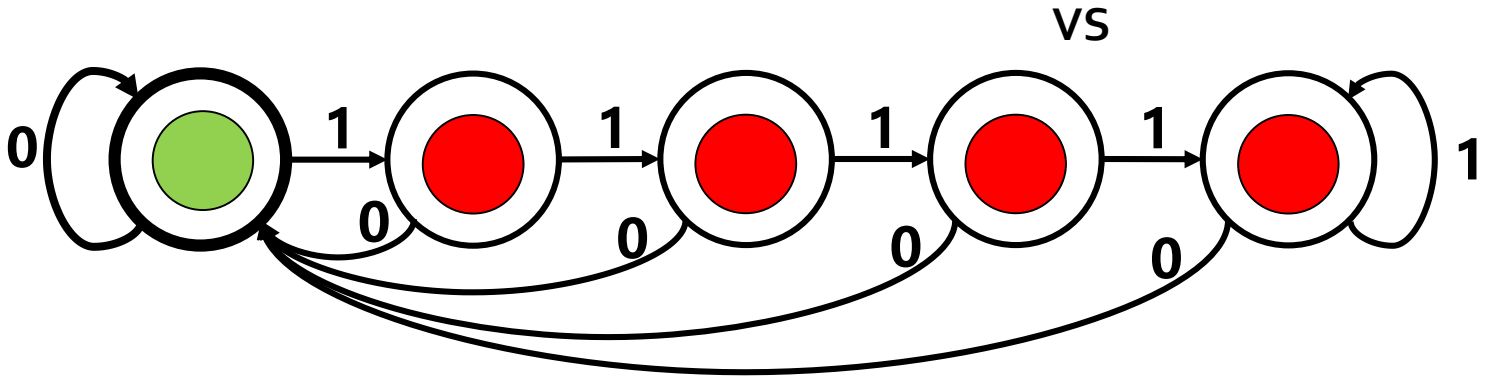
按键规则的状态转换图



按下“0”键，指示灯“灭”
按下“1”键，指示灯“亮”



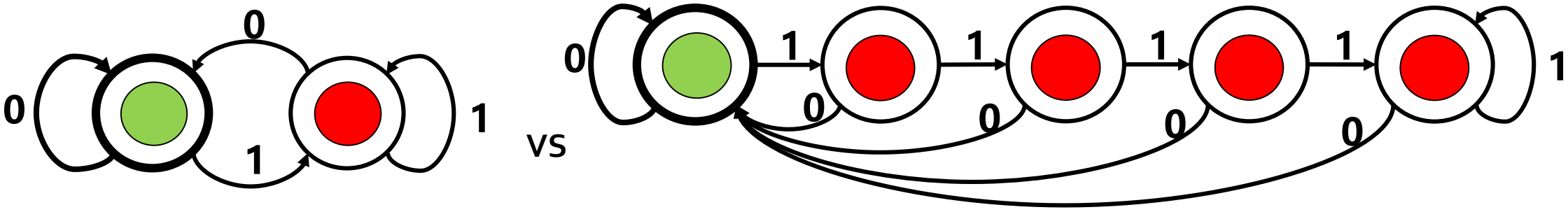
VS



- 如果没有明确的状态机描述规范，无法得到确定的状态转换图
- 如果知道状态数目的边界，则可以推断出其行为
  - 具有 K 个状态的状态机，其每个状态总可以在小于 K 的跳步到达

但状态机种的某些状态可能是**等价**的！

状态机等效(FSM Equivalence)



上述2个状态机是否有差异?

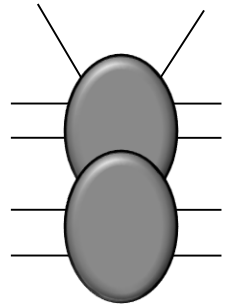
它们从表现的外部特征看不出任何差异，因此可以互换使用

状态机等价：当且仅当输入序列产生完全相同的输出序列

工程目标:

- 所设计的状态机具有最简单的等效状态
- 意味着最简单的电路实现，低成本，可能会高速度

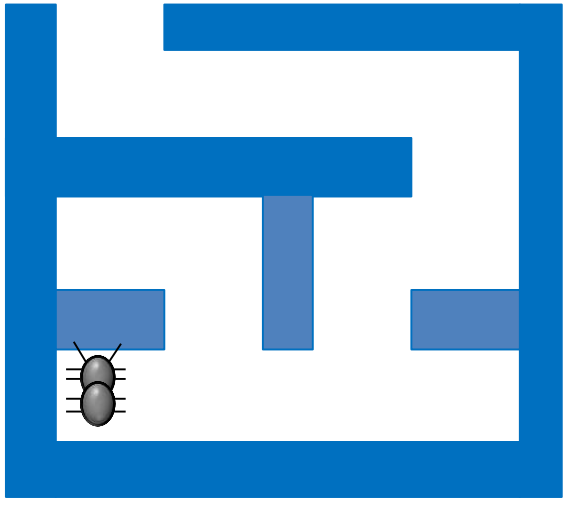
智能蚂蚁



- 传感器：Left 和 Right 两个触角，任何一个触角接触到物体，输入=1
- 动作：前进步 F，10 度转向（TL，TR）

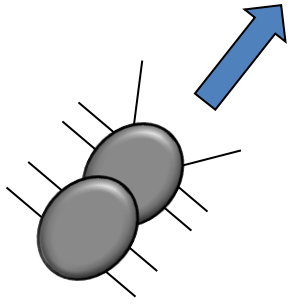
目标：设计蚂蚁的智能，使其能走出迷宫

策略：右侧触角始终接触墙壁

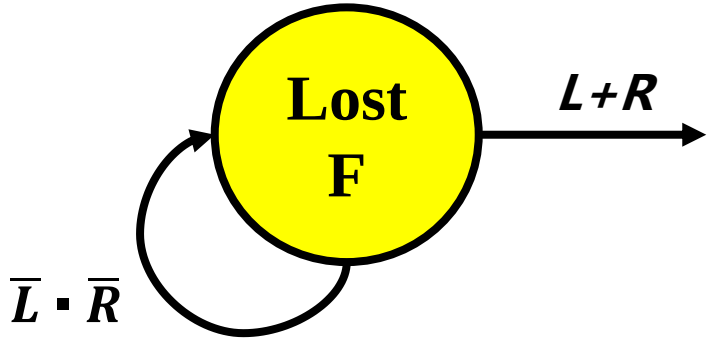


蚂蚁失去空间感(Lost Space)

状态：失去空间感

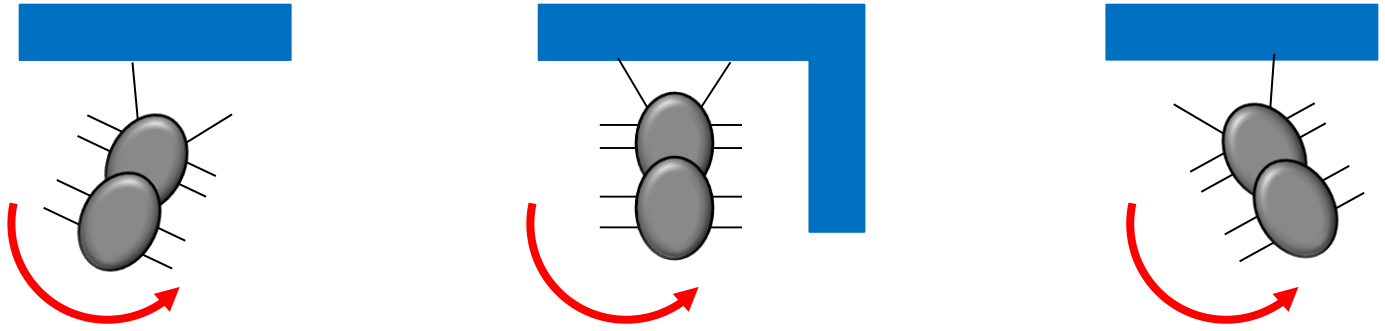


动作：一直向前走，直到接触到墙壁

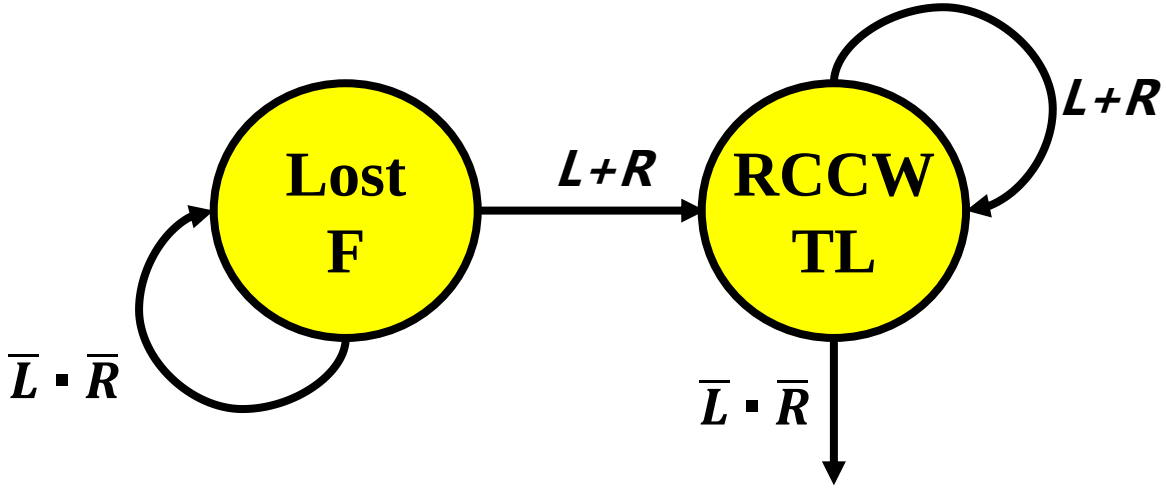


假设 *Lost* 是初始状态

触角碰壁(Contact Wall)

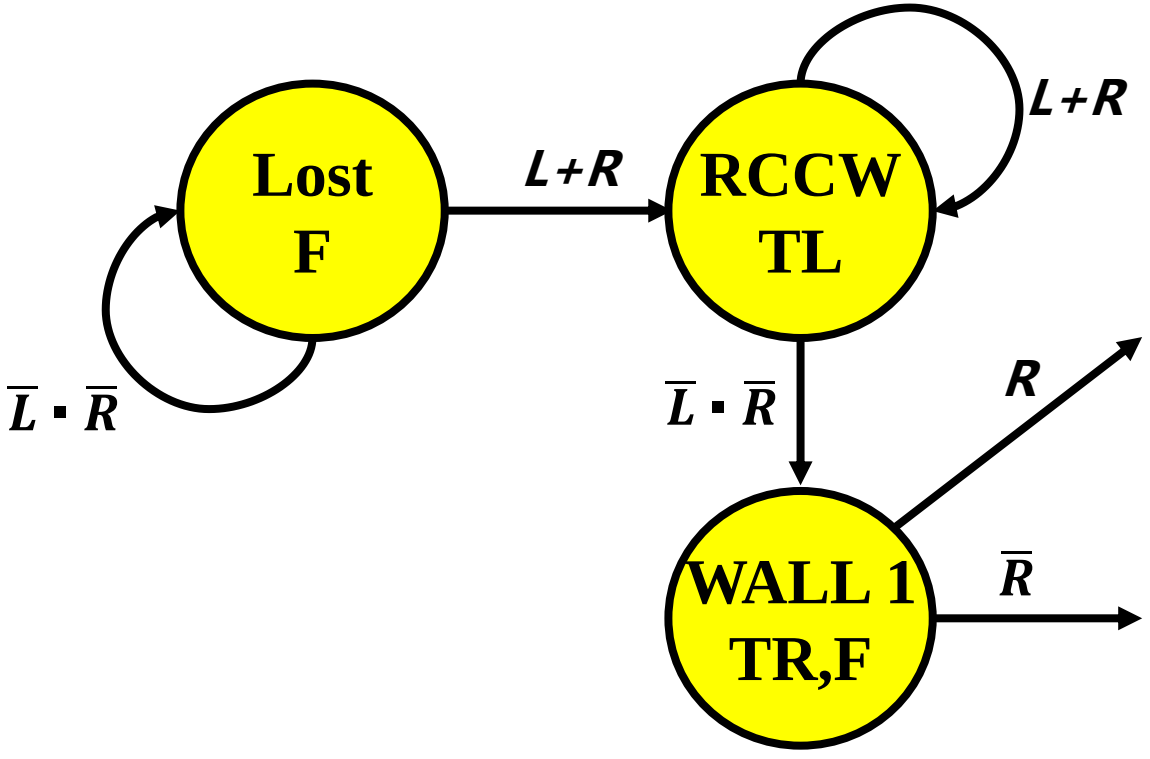


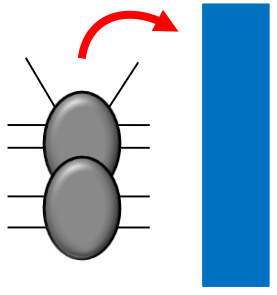
动作：向左转(TL，Counter Clock Wise, CCW)，直到不再触碰墙壁



向右转一点儿，找墙

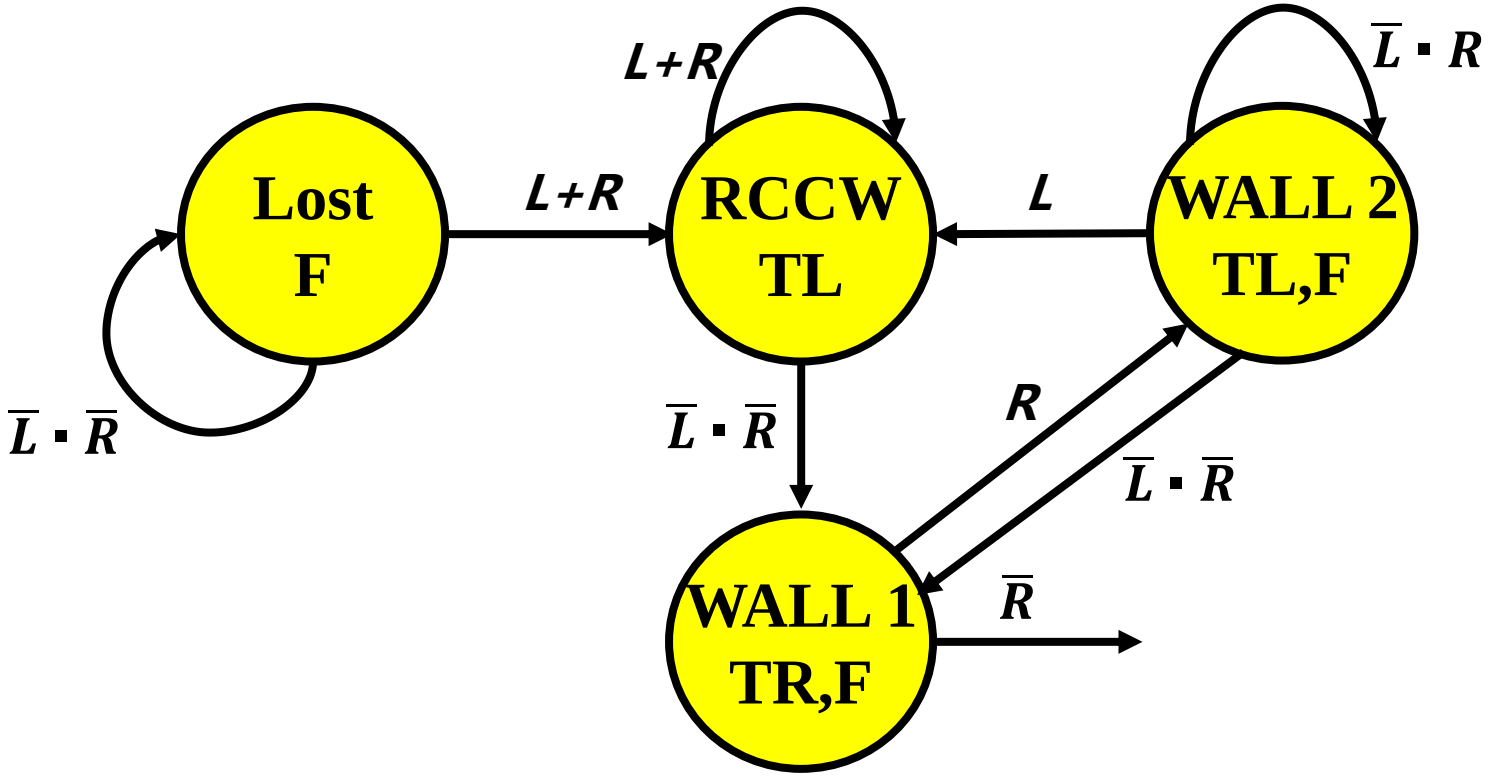
动作：向前走（F），并且向右转（TR）一点儿，找到墙

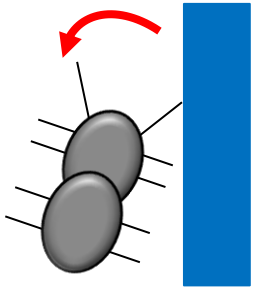




向左转一点儿，不再碰墙

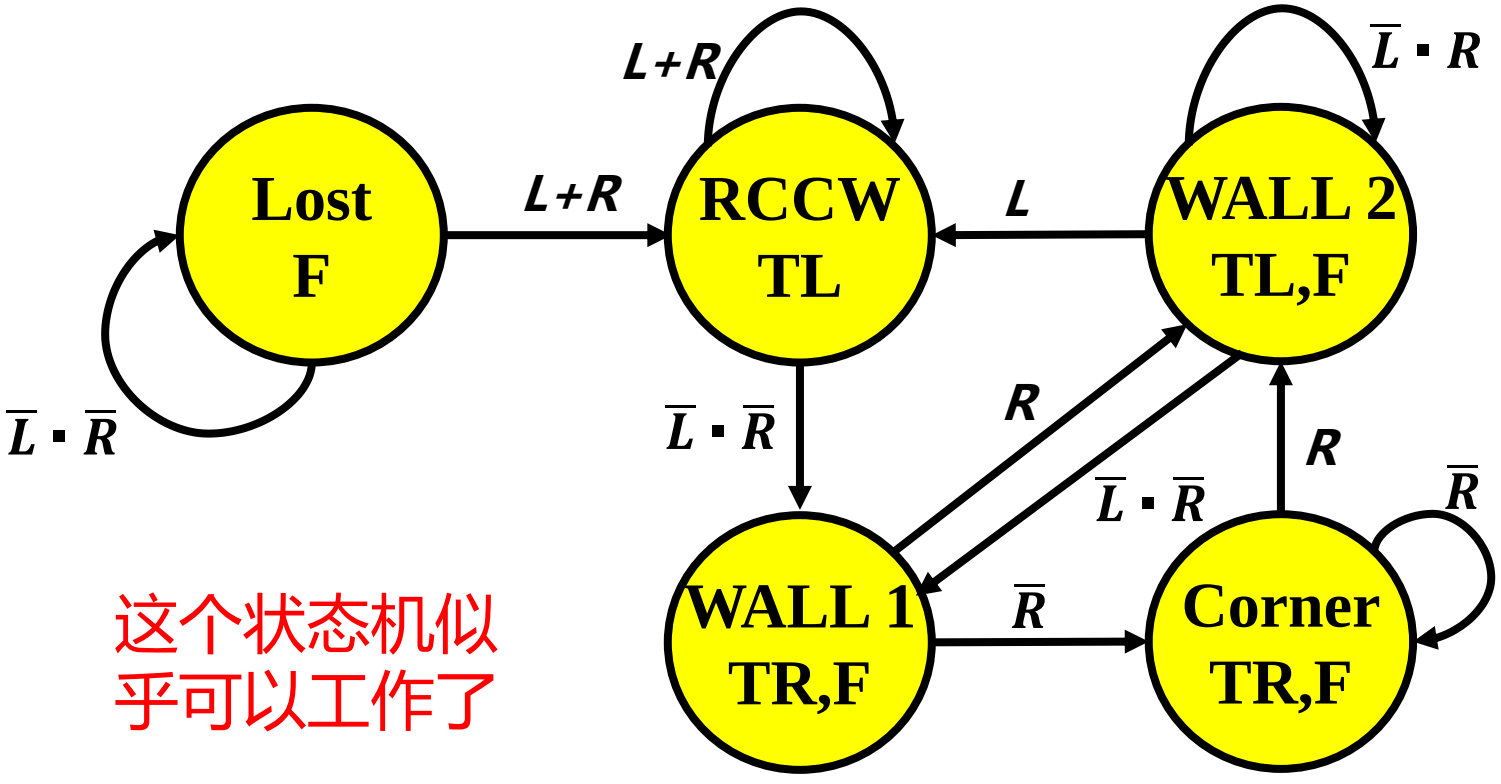
动作：向前一步（F），并向左转一点儿（TL），直到不再碰墙



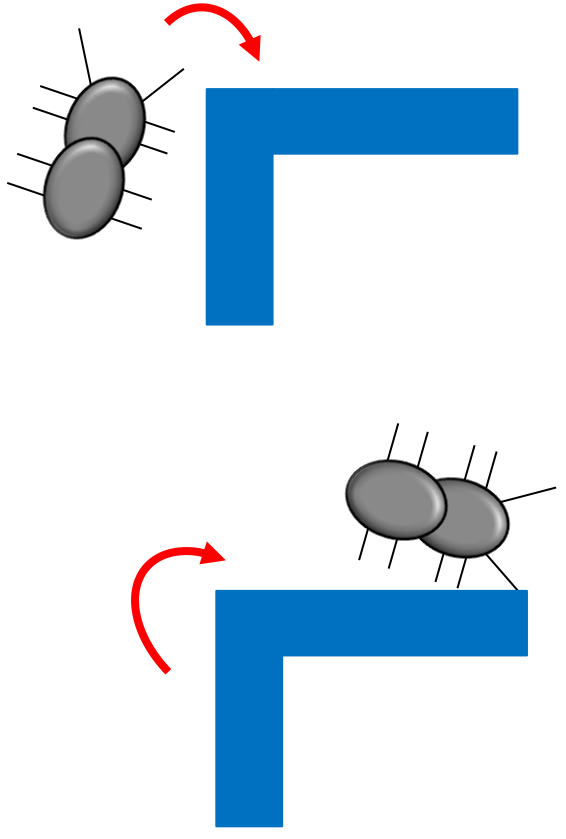


迷宫外墙角处理

动作：向前一步（F），并向右转（TR），直到碰到垂直的墙

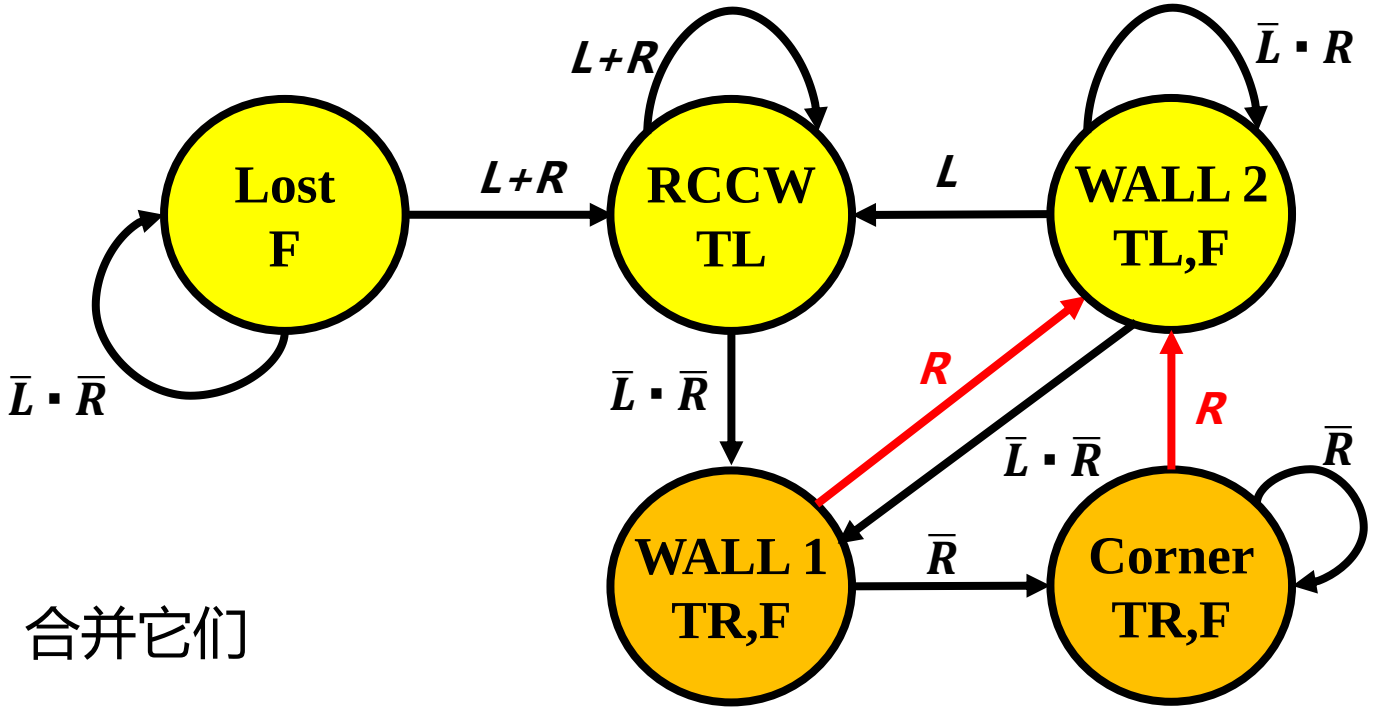


这个状态机似乎可以工作了



等价状态简化

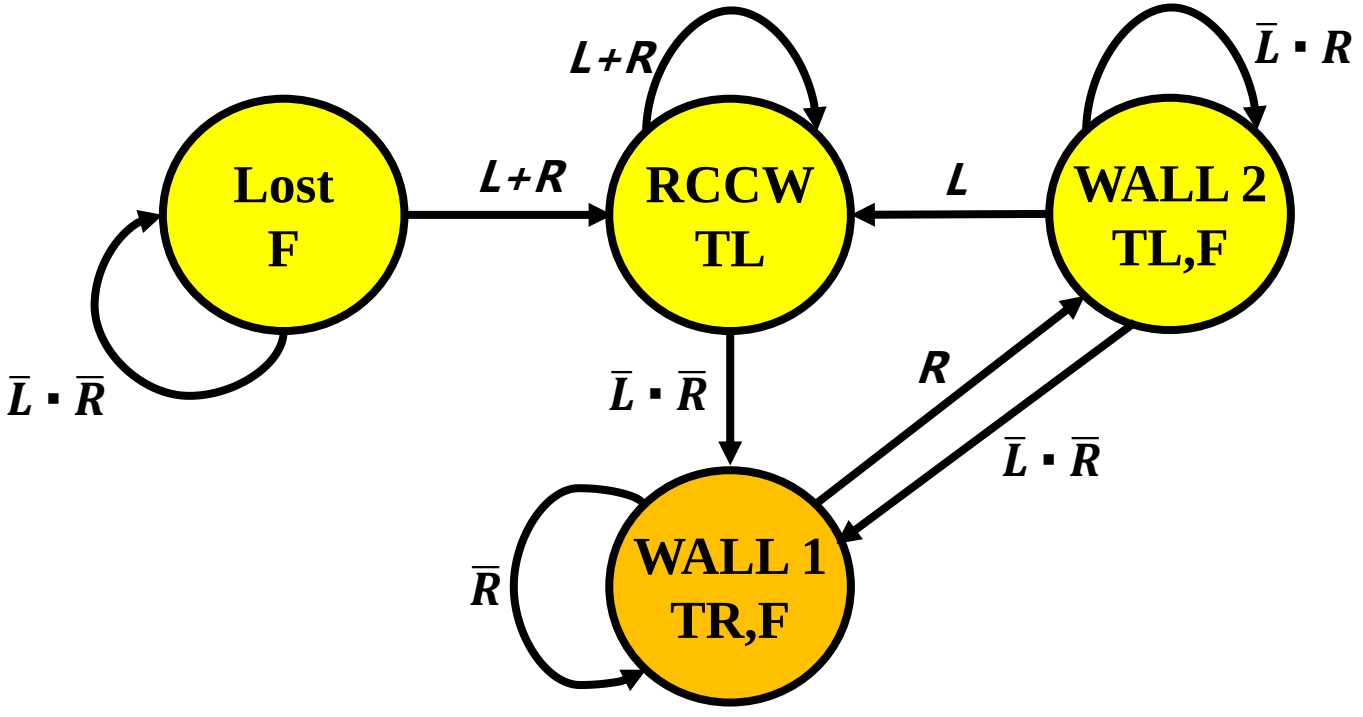
- 等价状态合并的前提：
  - 两个状态具有相同的输出，*并且*
  - 对于每组输入，都会使状态转换成相同的新状态



- 状态合并策略：
  - 找到一对等价状态，合并它们

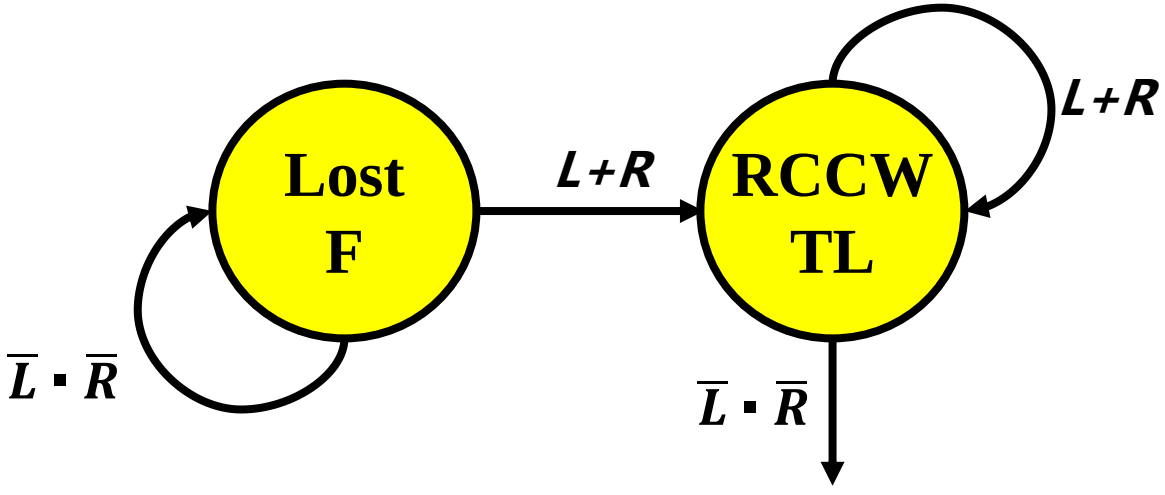
状态图进化

- 合并状态 **WALL1** 和 **Corner**，形成一个新状态 **WALL1**



此 4-State 状态机的行为与前面的 5-State 状态机完全相同，但只需1半ROM空间来实现！

建立状态转换表



S	L	R	S'	TR	TL	F
00	0	0	00	0	0	1
00	0	1	01	0	0	1
00	1	0	01	0	0	1
00	1	1	01	0	0	1
01	0	1	01	0	1	0
01	1	0	01	0	1	0
01	1	1	01	0	1	0

实现细节

S	L	R	S'	TR	TL	F
00	0	0	00	0	0	1
00	1	-	01	0	0	1
00	-	1	01	0	0	1
01	1	-	01	0	1	0
01	-	1	01	0	1	0
01	0	0	10	0	1	0
10	-	0	10	1	0	1
10	-	1	11	1	0	1
11	1	-	01	0	1	1
11	0	0	10	0	1	1
11	0	1	11	0	1	1

S'_1

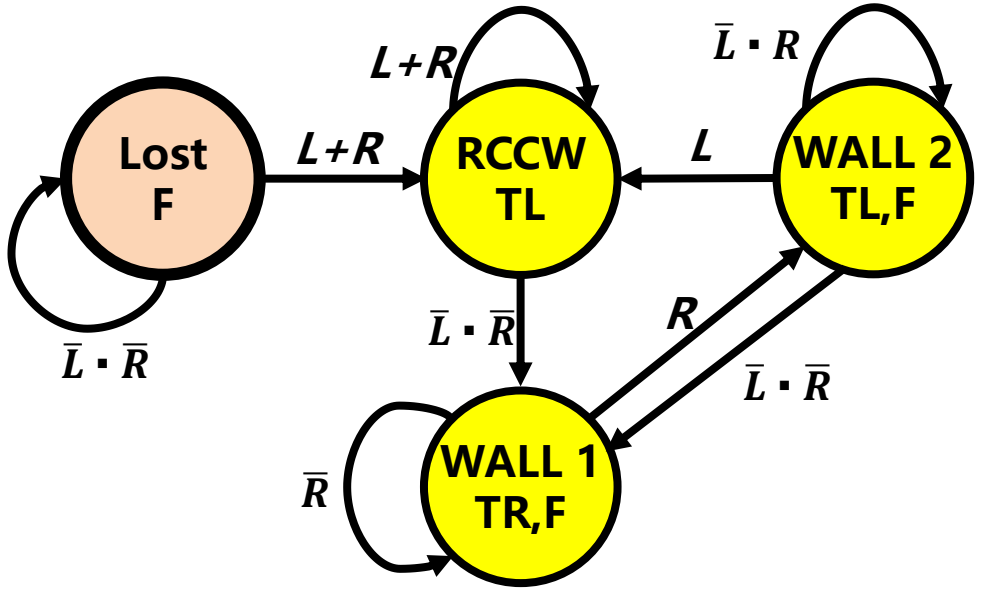
S_1S_0 / LR	S_1=0 00	S_1=0 01	S_1=1 11	S_1=1 10
L=0 00	0	1	1	1
L=0 01	0	0	1	1
L=1 11	0	0	0	1
L=1 10	0	0	0	1

$$S'_1 = S_1\overline{S_0} + \overline{L}S_1 + \overline{L}\overline{R}S_0$$

S'_0

S_1S_0 / LR	S_1=0 00	S_1=0 01	S_1=1 11	S_1=1 10
L=0 00	0	0	0	0
L=0 01	1	1	1	1
L=1 11	1	1	1	1
L=1 10	1	1	1	0

$$S'_0 = R + L\overline{S_1} + LS_0$$

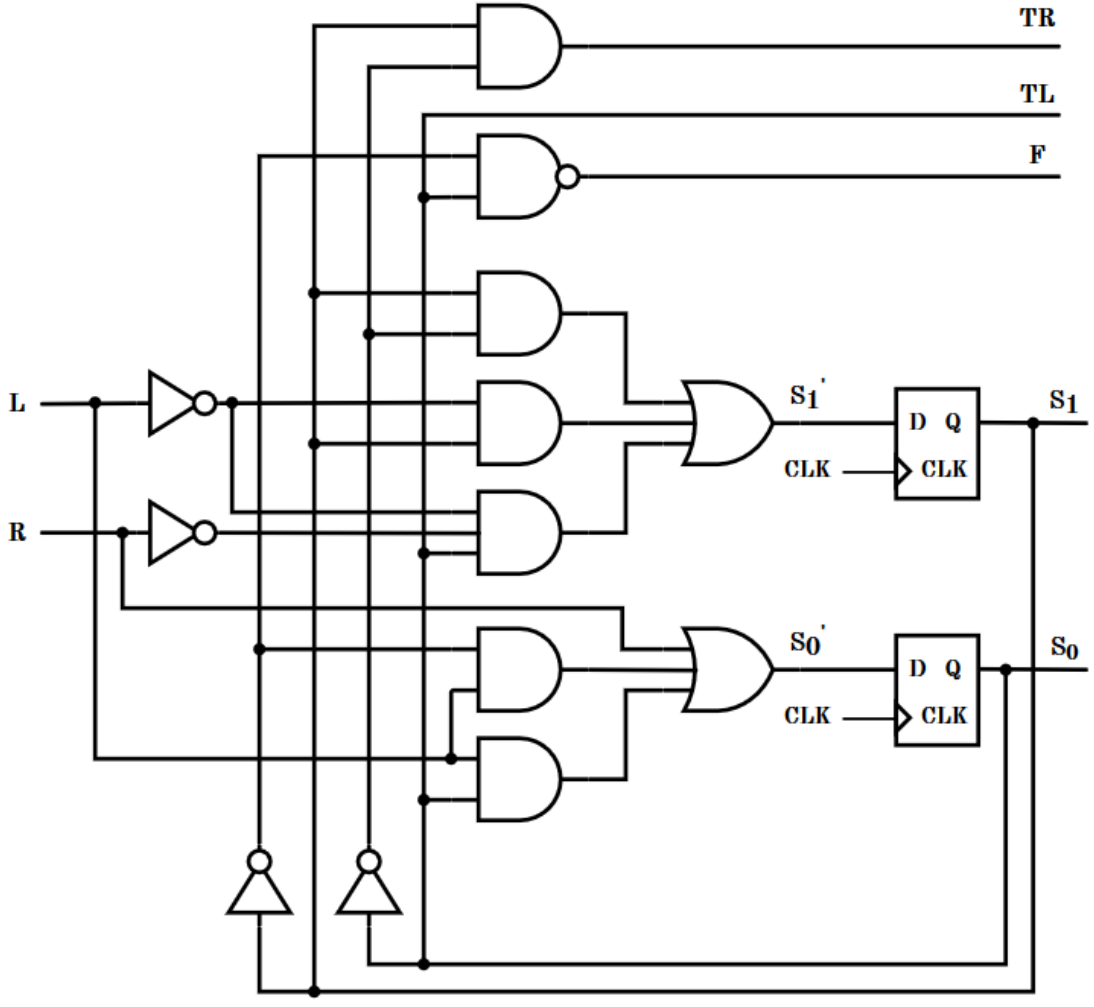


蚂蚁智能实现电路

驱动方程 $\begin{cases} S_1' = S_1\overline{S_0} + \overline{L}S_1 + \overline{L}\overline{R}S_0 \\ S_0' = R + L\overline{S_1} + LS_0 \end{cases}$

D触发器状态方程：$Q = D$

输出方程 $\begin{cases} TR = S_1\overline{S_0} \\ TL = S_0 \\ F = \overline{\overline{S_1} \cdot \overline{S_0}} \end{cases}$

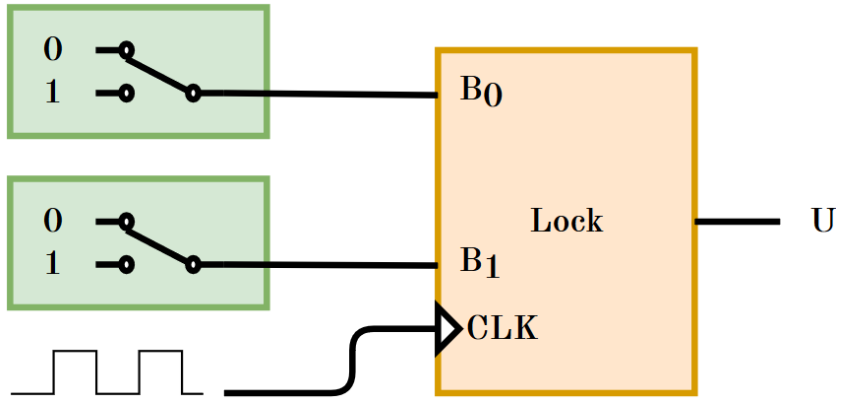


状态机将一直与我们为伴

- 不仅可以实现智慧蚂蚁
  - 如蜂群聚集、飞鸟聚集、学校开学等群体聚集活动，东可以用一些列简单状态机来描述
- 在物理学中
  - 细胞自动机(Cellular Automaton)，采用一系列有限状态机矩阵来描述流体力学问题，要比偏微分方程 (PDEs) 的数值解更精确
- 考虑
  - 如果我们采用RAM来代替ROM来实现FSM，并且允许状态机输出可以改写RAM内的数据?
- 未来的生活中，FSM将时钟与我们相伴

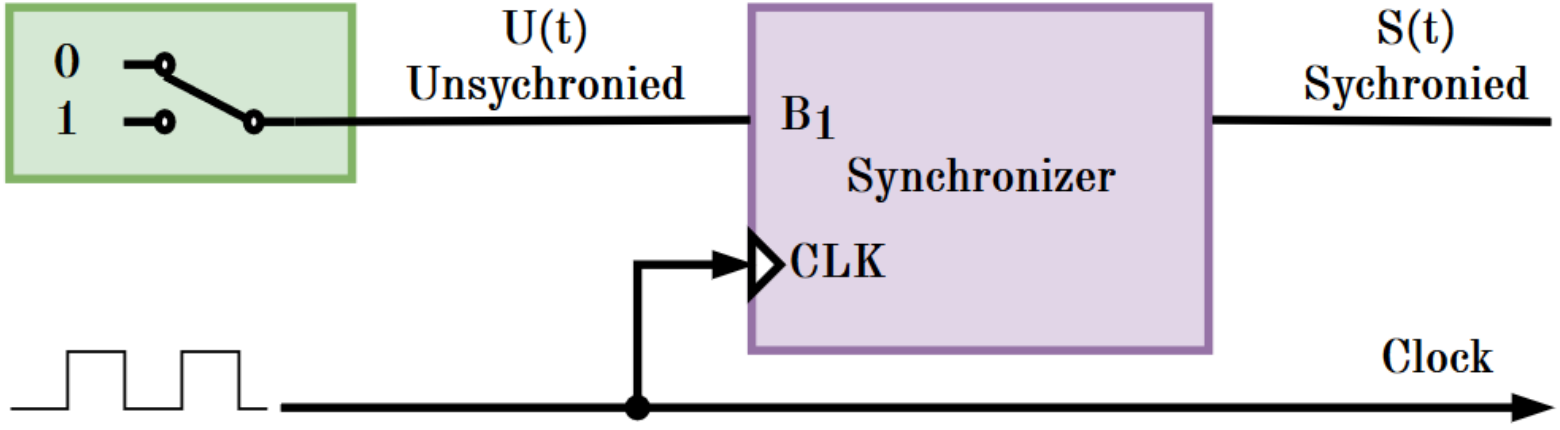
数字系统的多时钟域问题

The world does not run on our clock!



如图所示，如果两个按键输入的“0”和“1”电平是异步的

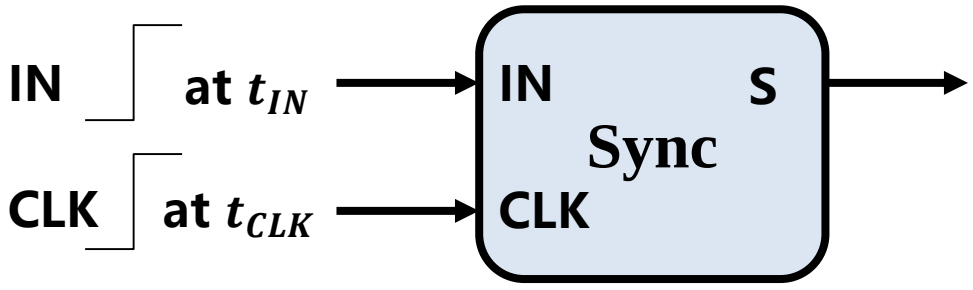
在包含异步输入的电路种，因为不能保证所有异步输入的建立和保持时间都能满足电路要求，所以在输入电路中要包含“同步器”



在一个Clock周期内，Clock上升沿后的一小段时间开始，一直有效

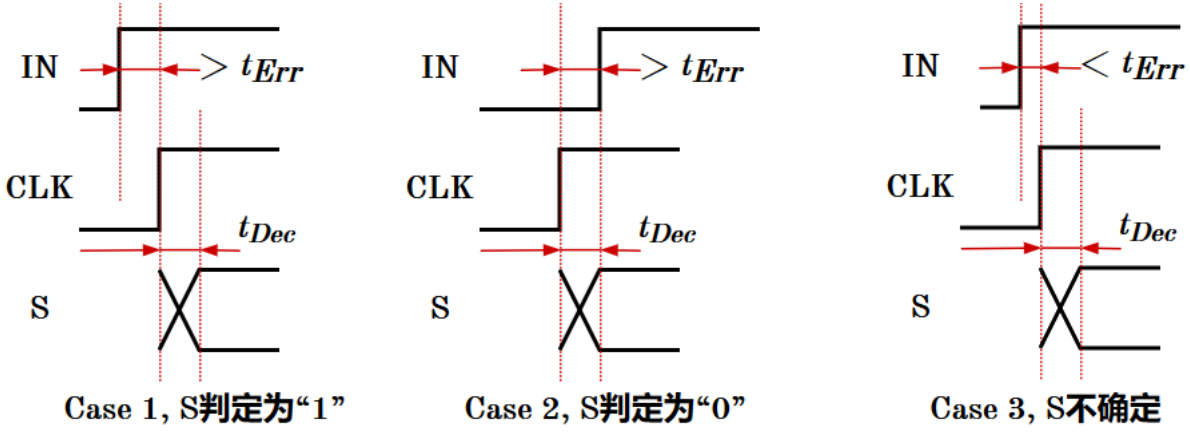
限定时间同步器

一个经典的未完全解决的问题



同步器的时序规范：

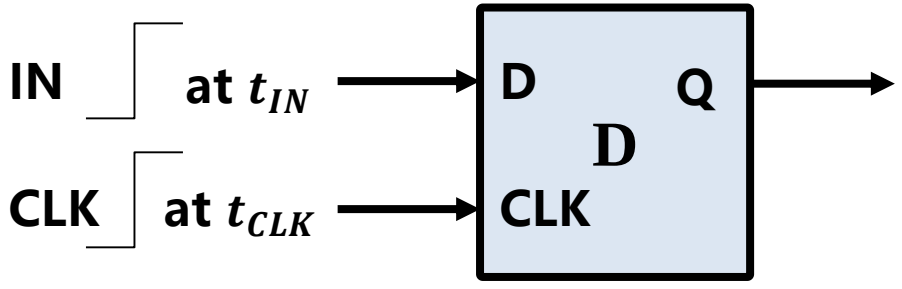
- 判定时间有限：t_{Dec}
- 允许的出错时间有限：t_{Err}
- 在时刻 $t_{CLK} + t_{Dec}$的同步输出S的值判定规则
  - **“1”，如果 $t_{IN} < t_{CLK} - t_{Err}$**
  - **“0”，如果 $t_{IN} > t_{CLK} + t_{Err}$**
  - **不确定(“0”或“1”)，其他情况**



即便采用100%可靠的部件，也不能完全解决同步问题，因为输出可能不确定

触发器实现同步器

使用D触发器实现同步器



使用触发器实现同步器的原因在于：在数字抽象时，我们总是假设触发器输出端Q的状态，总是“0”或“1”

但事实可能不是，假设锁存器的CLK和D几乎同时发生变化

触发器的判决时间是其传输延迟 t_{PD}

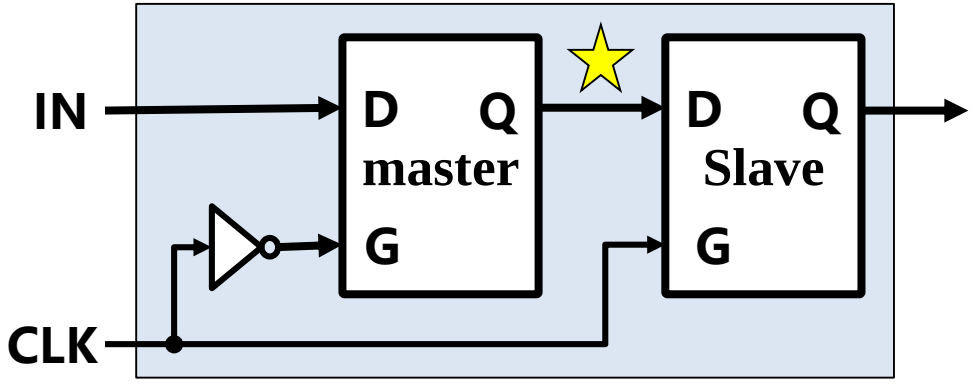
允许的出错时间 $t_{Err} = max\{t_{SETUP}, t_{HOLD}\}$

在t_{CLK}之后的t_{PD}时间，逻辑输出判定为：

$Q = 1$，当且仅当：$\boldsymbol{t_{IN} + t_{SETUP} < t_{CLK}}$

$Q = 0$，当且仅当：$\boldsymbol{t_{CLK} + t_{SETUP} < t_{IN}}$

$Q = 0$或1，其他情况

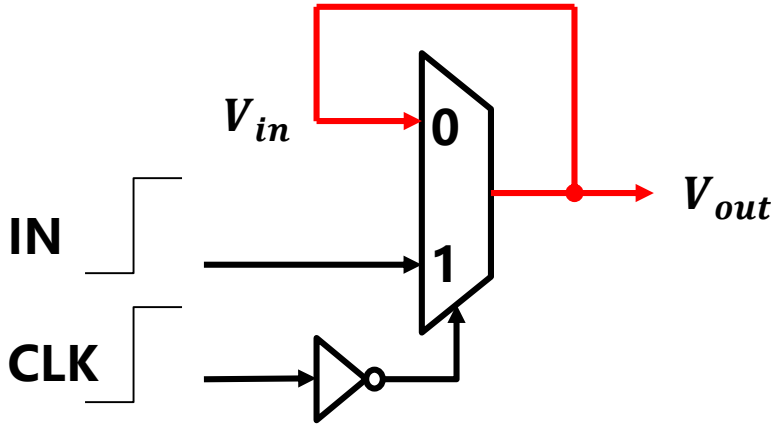


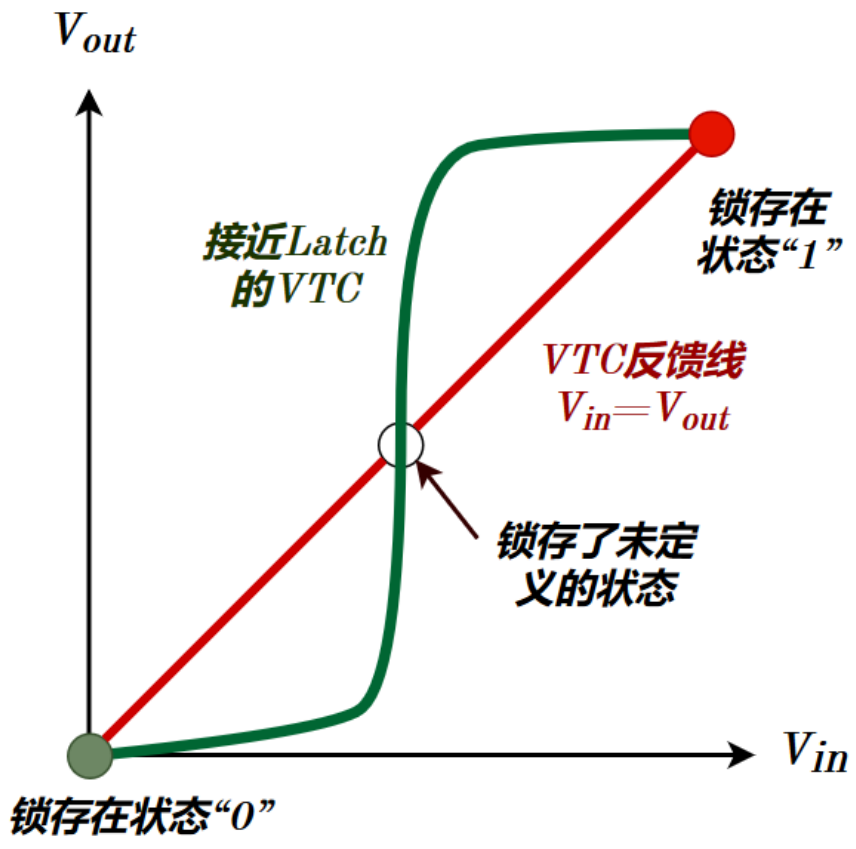
主从触发器结构

产生“亚稳态”的 原因

回顾锁存器(Latch)的输出结果受限于两个同时成立的假设

1. 输入、输出路径上的电压转移特性(VTC)
2. $V_{in} = V_{out}$

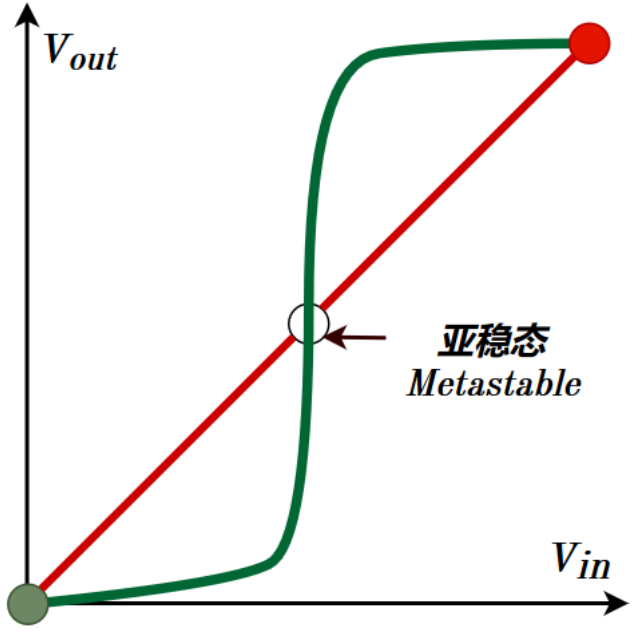




在锁存器的状态中，除了我们预想的“0”或“1”稳定状态外，还存在不稳定的平衡状态，称之为亚稳态(Metastable State)

亚稳态性质

- 亚稳态对应于一个无效的逻辑电平
- 是一个不稳定的平衡状态，小的波动会导致电路离开亚稳态
- 最终电路会回到稳定的“0”或“1”状态
- 亚稳态的持续时间取决于输出端Q的电压与亚稳态点的电压的接近程度，越接近，电路在亚稳态点的停留时间越长
- 任何双稳态电路都存在亚稳态现象

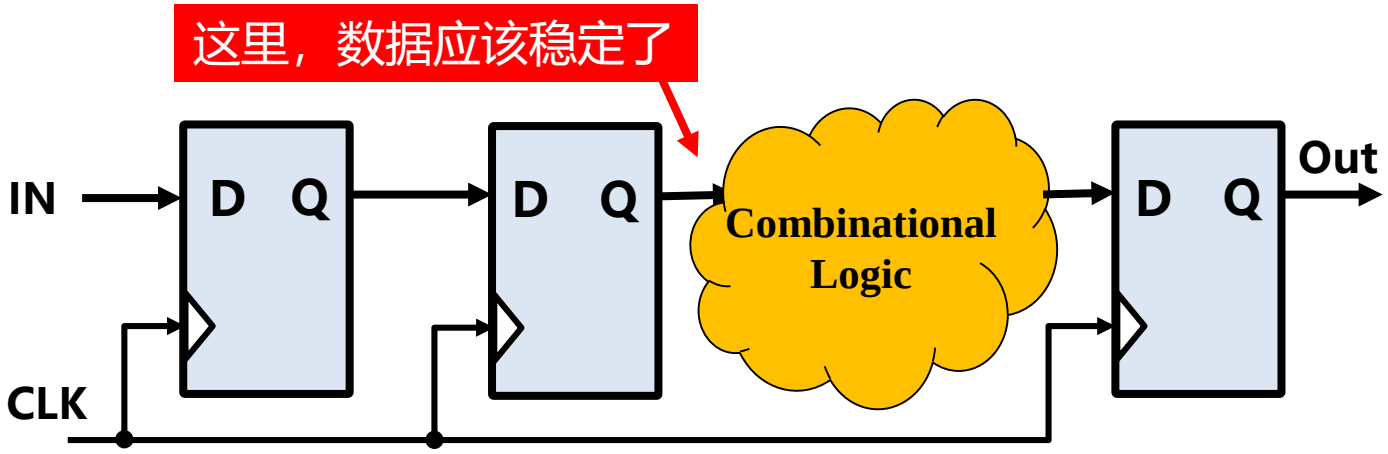


如果在t_0时刻电路处于亚稳态

1. $p_{(matastable\ at\ t_0+T)} > 0$, 经过时间间隔 ***T*** 电路仍处于亚稳态的概率大于0
2. 随着时间 ***T*** 的增加，电路仍处于亚稳态的概率指数下降

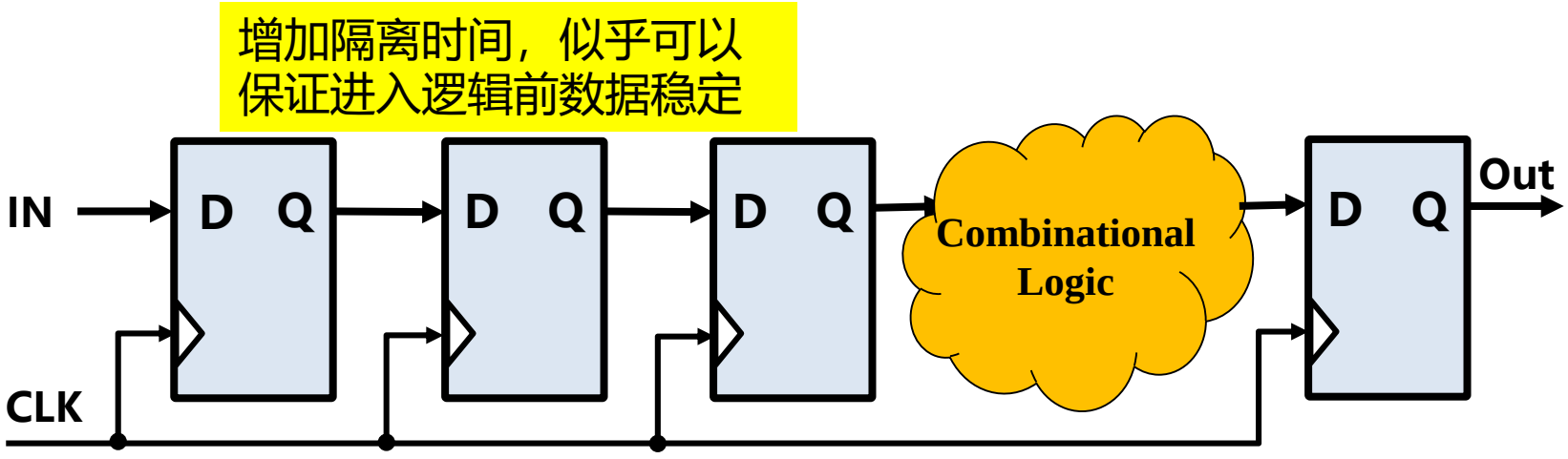
亚稳态的解决方法

在异步输入和逻辑实现之间，增加额外的寄存器是消除亚稳态的最保险的方法



对于高频工作的逻辑电路，可以考虑增加多余的寄存器

延时会增加可靠性



问题和建议?