

数字逻辑与处理器基础知识与方法

T^TT

2024 年 10 月 14 日

目录

1	布尔代数	2	2.2.2	组合逻辑电路的设计过程	5
1.1	数的编码与表示	2	2.2.3	组合逻辑电路的评价指标	6
2	逻辑计算	3	2.2.4	组合逻辑电路的设计实例	6
2.1	从电路到逻辑门	3	2.3	时序逻辑	9
2.2	组合逻辑	3	2.3.1	时序逻辑电路的基本概念	9
2.2.1	组合逻辑电路的分析方法	3	2.3.2	锁存器和寄存器	9
			2.3.3	时序逻辑电路分析	14
			2.3.4	时序逻辑电路设计	14

1 布尔代数

1.1 数的编码与表示

定义 1.1.1. 二进制

二进制是基数为 2，只有两个数码 0 和 1 的数制。二进制数中，每一个数码称为一个二进制位 (**bit**)，权值最小的二进制位称为**最低位 (LSB)**，权值最大的二进制位称为**最高位 (MSB)**。

所有的 4-bit 二进制数如表 1.1 所示。

表 1.1: 4-bit 二进制数

BIN				DEC	HEX	BIN				DEC	HEX
0	0	0	0	0	0	1	0	0	0	8	8
0	0	0	1	1	1	1	0	0	1	9	9
0	0	1	0	2	2	1	0	1	0	10	A
0	0	1	1	3	3	1	0	1	1	11	B
0	1	0	0	4	4	1	1	0	0	12	C
0	1	0	1	5	5	1	1	0	1	13	D
0	1	1	0	6	6	1	1	1	0	14	E
0	1	1	1	7	7	1	1	1	1	15	F

二进制数的**左移**运算和**右移**运算分别是将二进制数的所有位向左或向右移动一位，移动后的空位补 0。左移一位相当于乘 2，右移一位相当于除 2。

定义 1.1.2. BCD 码

BCD (binary-coded decimal) 码是二进制编码的一种，用 4 位二进制数表示一个十进制数的一位。**8421 BCD 码**的编码规则是：用二进制数的 0-9 的编码表示十进制数的 0-9，不使用二进制数的 10-15 的编码。

由于 8421 BCD 码是**有权码**，其加减法运算可以直接使用二进制数和十进制数的加减法运算规则。

例题 1.1.1. (1) $34_{10} + 45_{10} = 0011\ 0100_{BCD} + 0100\ 0101_{BCD} = 0111\ 1001_{BCD} = 79_{10}$ 。

(2) $14_{10} + 9_{10} = 0001\ 0100_{BCD} + 0000\ 1001_{BCD} = 0001\ 1101_{BCD} \xrightarrow{\text{进位}} 0010\ 0011_{BCD} = 23_{10}$ 。

8 个二进制位称为一个**字节**。

2 逻辑计算

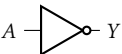

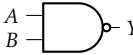
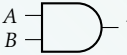




2.1 从电路到逻辑门

定义 2.1.1. 逻辑门

逻辑门是一种能够实现逻辑运算的电路，其输入和输出均为逻辑值。逻辑门的输入和输出均为二进制数，输入的二进制数称为**输入变量**，输出的二进制数称为**输出变量**。

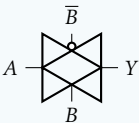
常用的逻辑门如表 2.2 所示。

表 2.2: 常用逻辑门

逻辑门	符号	记号	运算	逻辑门	符号	记号	运算
非门		NOT	$Y = A' ^1$	缓冲器		BUF	$Y = A$
与非门		NAND	$Y = \overline{A \cdot B}$	与门		AND	$Y = A \cdot B$
或非门		NOR	$Y = \overline{A + B}$	或门		OR	$Y = A + B$
异或非门		XNOR	$Y = \overline{A \oplus B}$	异或门		XOR	$Y = A \oplus B$

元件 2.1. 传输门

记号



特性

传输门是一种多输入单输出的逻辑门，其输出为

$$Y = \begin{cases} A, & B = 1 \\ \text{undefined}, & B = 0 \end{cases}$$

2.2 组合逻辑

定义 2.2.1. 组合逻辑

组合逻辑是一种逻辑电路，其输出仅取决于当前的输入及延时，与电路的历史状态无关。组合逻辑电路中没有反馈回路。

2.2.1 组合逻辑电路的分析方法

分析组合逻辑电路，即是从给定的设计电路（晶体管或逻辑门电路）中，找出输入与输出之间的关系，用真值表、布尔表达式等形式表示。

¹为区别单个的非逻辑和其他多目运算中的非，这里约定单独的非逻辑用 A' 表示，夺目运算附带的非逻辑用 \overline{A} 表示。

例题 2.2.1. 分析如图 2.1a 所示的组合逻辑电路。

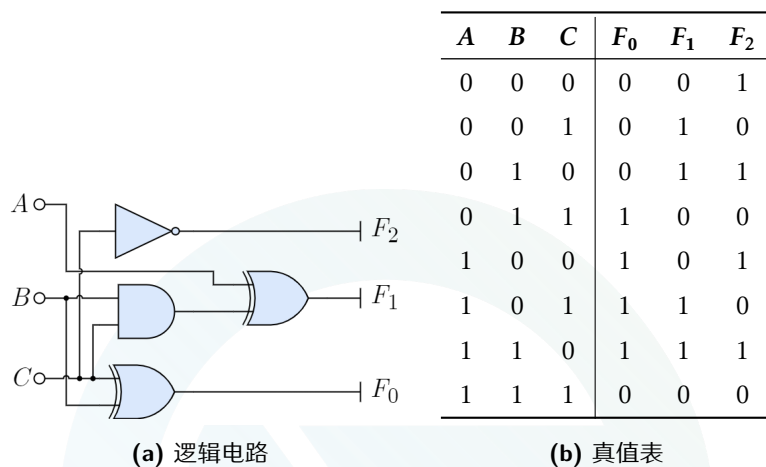


图 2.1: 组合逻辑电路实例 1

解. 根据逻辑门的运算规则, 可得到

$$\begin{cases} F_0 = A \oplus (B \cdot C) \\ F_1 = B \oplus C \\ F_2 = \bar{C} \end{cases}$$

因此, 该组合逻辑电路输出的真值表如表 2.1b 所示。可以看出, 这个电路所实现的功能为 $(F_0F_1F_2)_2 = (ABC)_2 + 1$, 这是一个 3-bit 二进制自增电路。

⑤

例题 2.2.2. 分析如图 2.2a 所示的组合逻辑电路。

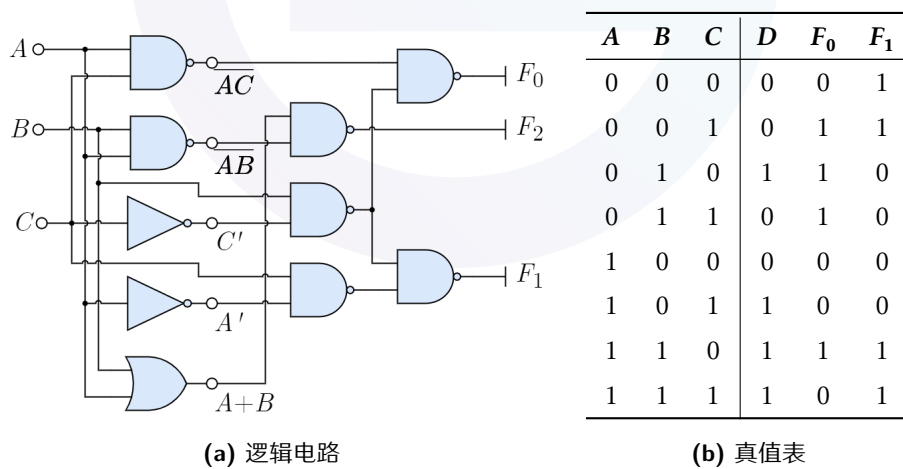


图 2.2: 组合逻辑电路实例 2

解. 根据逻辑门的运算规则, 可得到

$$\begin{cases} F_0 = \overline{AC} \cdot \overline{Bc} \\ F_1 = \overline{BC} \cdot \overline{CA} \\ F_2 = (A + B) \cdot \overline{AB} \end{cases}$$

因此, 该组合逻辑电路输出的真值表如表 2.2b 所示, 这是一个 Gray 码递增电路。

⑤

2.2.2 组合逻辑电路的设计过程

设计一个组合逻辑电路, 需要将算法转化为二元逻辑的计算, 化简后用逻辑电路结构实现。

例题 2.2.3. 设计一个 2-bit 比较器。

解. 2-bit 比较器的设计需求为:

- 功能: 比较两个 2-bit 二进制数的大小;
- 输入: 两个 2-bit 二进制数 A_1A_0 和 B_1B_0 ;
- 输出: 三个逻辑值 LT 、 EQ 和 GT , 分别表示 $A < B$ 、 $A = B$ 和 $A > B$ 。

用卡诺图表示出 LT 、 EQ 和 GT 的逻辑表达式如表 2.3 所示, 即知其两级与或表达式为

$$LT = A_1'B_1 + A_1'A_0B_0 + A_0'B_1B_0$$

$$EQ = A_1'A_0B_1'B_0' + A_1'A_0B_1'B_0 + A_1A_0'B_1B_0' + A_1A_0B_1B_0$$

$$GT = A_1B_1' + A_1B_1'B_0' + A_1A_0B_0'$$

其中 EQ 可以更简单地表示为

$$EQ = \overline{A_1 \oplus B_1} \cdot \overline{A_0 \oplus B_0}$$

或者利用另外两个输出的逻辑表达式, 即

$$EQ = \overline{LT + GT}$$

将 LT 、 EQ 和 GT 的逻辑表达式转化为逻辑电路, 即可得到 2-bit 比较器的设计。

⑤

表 2.3: 2-bit 比较器设计的卡诺图

(a) LT					(b) EQ					(c) GT				
$A_1A_0 \setminus B_1B_0$	00	01	11	10	$A_1A_0 \setminus B_1B_0$	00	01	11	10	$A_1A_0 \setminus B_1B_0$	00	01	11	10
00		1	1	1	00	1				00				
01			1	1	01		1			01	1			
11					11			1		11	1	1		1
10			1		10				1	10	1	1		

如上 4-bit 输入的逻辑运算已经比较复杂。对于更加复杂的逻辑运算, 在处理中需要采取更多的技巧以简化运算, 如:

- 将输入变量分组, 写成更简单的逻辑表达式的多级运算:

$$f(A, B, C, \dots) = F(g_1(A, B, \dots), g_2(C, \dots), \dots)$$

- 将输入变量分离, 写成更简单的逻辑表达式的分支计算:

$$f(A, B, C, \dots) = A \cdot g_1(B, C, \dots) + \bar{A} \cdot g_2(B, C, \dots)$$

- 从结构化表达式中找出重复的部分加以复用, 简化逻辑电路。

同时, 还需要照应到实际的功耗、性能、面积等要求。

2.2.3 组合逻辑电路的评价指标

评价逻辑电路的主要指标包括:

- 稳态因素:
 - 逻辑电平: 逻辑电路的输入和输出电平高低;
 - 噪声容限: 逻辑电路抵抗噪声的能力;
 - 静态功耗: 逻辑电路在稳态工作时的功耗, 主要与电路的 V_{CC} 有关;
 - 面积: 逻辑电路的物理尺寸;
 - 扇出系数: 逻辑门的输出能够驱动的输入数量。
- 动态因素:
 - 传输延迟和时钟频率: 逻辑电路的输入到输出的延迟时间;
 - 时序容限: 逻辑电路的输入信号的时序要求;
 - 动态功耗: 逻辑电路在工作时的功耗, 主要与电路的切换频率有关;
 - 噪声: 逻辑电路在工作时产生的噪声。

2.2.4 组合逻辑电路的设计实例

A) 编码器 (Encoder) 和译码器 (Decoder) 用 m 个二进制位对 $n \leq 2^m$ 个输入信号进行编码, 得到 m 位二进制代码的电路, 称为 $2^m - m$ 线编码器。

例题 2.2.4. 设计一个 4-2 线编码器用作抢答器, 其中每个抢答按钮按下时对应输入信号为 1, 其余输入信号为 0。

解. 4 个输入信号的抢答器的真值表如表 2.3a 所示^a, 容易得到其逻辑表达式为

$$Y_0 = A_3 + A_1$$

$$Y_1 = A_3 + A_2$$

该编码器的设计如图 2.3b 所示。

但是, 这个编码器不能满足抢答器的优先要求。为了实现优先级, 可以将输入信号的优先级从高到低排列, 然后将优先级高的输入信号的输出信号设为 1, 优先级低的输入信号的输出信号设为 0。具体的电路设计略。 ⑤

^a 由于编码器的输入信号类型数最多为输出信号所能表示的最大数值, 其必定要用无关项的形式归总一些设计之外的情况。考虑到输出信号的唯一性, 表中无关项的分布不是对称的。

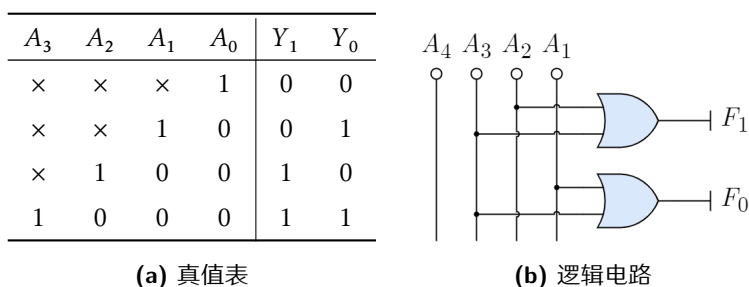


图 2.3: 4-2 线编码器设计

译码是编码的逆过程, 即将编码后的信号转换为原始信号。用 n 个二进制位对 $m \leq \log_2 n$ 个输入信号进行译码, 得到 n 位二进制代码的电路, 称为 $\log_2 n - n$ 线译码器。

B) 多路选择器 (Multiplexer, MUX) 用 n 个控制信号对 2^n 个输入信号进行选择, 得到一个输出信号的电路, 称为 $2^n : 1$ 多路选择器。

例题 2.2.5. 设计一个 4:1 多路选择器。

解. 4:1 多路选择器的真值表如表 2.4a 所示, 容易得到其逻辑表达式为

$$Q = A_1' A_0' D_0 + A_1' A_0 D_1 + A_1 A_0' D_2 + A_1 A_0 D_3$$

如图 2.4b, 该多路选择器的每一个输入信号都与一组控制信号相与, 各组中当且仅当输入控制信号全都对应时, 输出信号为对应的输入信号。

⑤

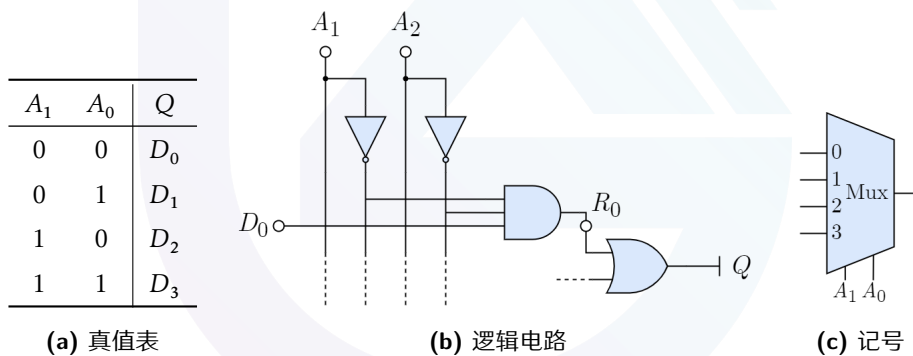


图 2.4: 4:1 多路选择器设计

实例 1 移位器

采用多路选择器可以实现移位运算。如图 2.5 所示, 一个 4-bit 右移位器可以通过 8 个 2:1 多路选择器实现。其中, S_0 控制右移 1 位, S_1 控制右移 2 位, 当 $S_1 S_0_{(2)} = 11_{(2)}$ 时即右移 3 位。换句话说, 控制信号形成的二进制数表示右移的位数。右移后, 高位补入 MSB 的值, 通常对正数补 0, 对负数补 1。

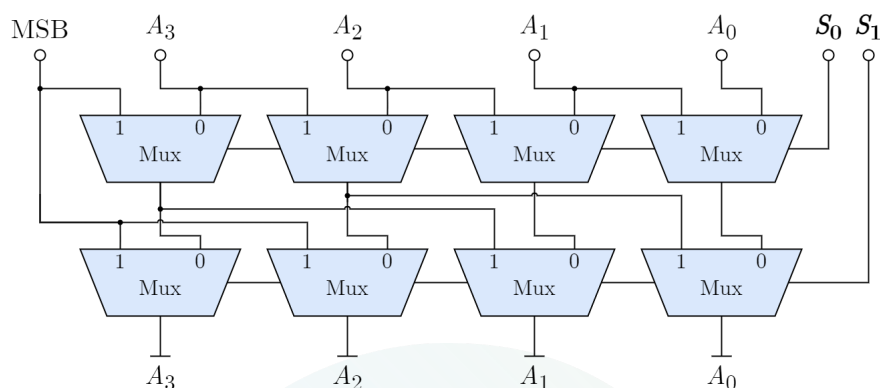


图 2.5: 4-bit 右移位器

C) 加法器 用于实现二进制数的加法运算。

例题 2.2.6. 设计一个 1-bit 全加器电路。

解. 1-bit 全加器有 3 个输入信号 (A 、 B 和进位信号 C_{in})、2 个输出信号 (本位的和 S 、进位输出信号 C_{out}), 其真值表如表 2.6a 所示。根据真值表, 可以得到 1-bit 全加器的逻辑表达式为

$$S = A \oplus B \oplus C_{in}$$

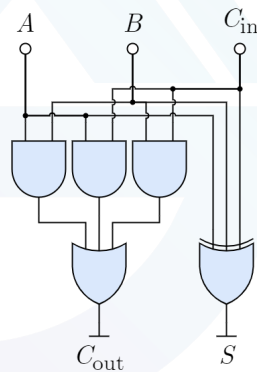
$$C_{out} = AB + BC_{in} + AC_{in}$$

该 1-bit 全加器的设计如图 2.6b 所示。

⑤

A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

(a) 真值表



(b) 逻辑电路

将 1-bit 全加器串联, 可得到一种 4-bit 全加器的设计, 如图 2.6 所示, 这称为**行波进位加法器 (ripple carry adder)**。

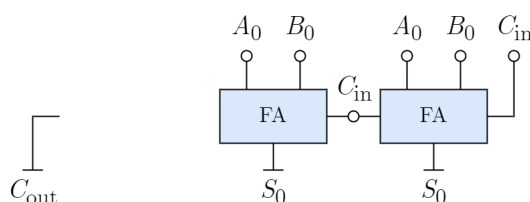


图 2.6: 4-bit 行波进位加法器

2.3 时序逻辑

2.3.1 时序逻辑电路的基本概念

A) **触发与节拍信号** 用时钟信号控制逻辑电路的运行，使其在时钟信号的**节拍**下按照一定的顺序依次触发响应进行逻辑运算，这种电路称为**时序逻辑电路**。

定义 2.3.1. 时钟信号发生器是一种能够产生连续的方波信号的电路。

用奇数个反相器可以串联成环形振荡器，若串联的反相器个数为 n ，每个反相器的延迟时间为 t_{pd} ，则环形振荡器的周期为 $T = 2n \cdot t_{pd}$ 。因此，环形振荡器是一种简单的时钟信号发生器。

基于时钟信号，在边缘（如上升沿）采样输入信号、更新状态，可以实现时序逻辑。

定义 2.3.2. 使信号事件与时钟节拍一致的操作称为**同步**，所有同步于同一个时钟信号的信号集合称为**时钟域**。

B) **状态与有限状态机** 一个系统的**状态 (state)** 应包含所有需要的信息而可以有冗余信息。依据输入信号和当前状态，系统的输出信号和下一个状态是确定的。

定义 2.3.3. 有限状态机

有限状态机 (finite state machine, FSM) 是一种能够根据输入信号和当前状态，确定输出信号和下一个状态的系统。

描述一个 FSM 需要确定 6 大要素：

- 状态集合 State, FSM 的所有可能状态；
- 初始状态 S_0 , FSM 启动及复位时的状态；
- 输入信号 Input；
- 输出信号 Output；
- 输出函数 f , 描述 FSM 的输出信号与输入信号和当前状态的关系；
- 状态转移函数 g , 描述 FSM 的下一个状态与输入信号和当前状态的关系。

FSM 有两种类型：**Moore FSM** 和 **Mealy FSM**。Moore FSM 的输出只与当前状态有关，而 Mealy FSM 的输出与当前状态和输入信号有关。两种 FSM 的状态转移函数 g 都可以用**状态转换表 (state transition table)** 表示，如表 2.4 所示；或者用**状态转换图 (state transition diagram)** 表示，如图 2.7 所示。

2.3.2 锁存器和寄存器

A) 锁存器

表 2.4: 一般的状态转换表

(a) Moore FSM

State	Input				Output
	I_1	I_2	\dots	I_n	
S_1	S_a	S_b	\dots	S_c	O_1
S_2	S_d	S_e	\dots	S_f	O_2
\vdots	\vdots	\vdots	\ddots	\vdots	\vdots
S_m	S_x	S_y	\dots	S_z	O_m

(b) Mealy FSM

State	Input			
	I_1	I_2	\dots	I_n
S_1	S_a, O_a	S_b, O_b	\dots	S_c, O_c
S_2	S_d, O_d	S_e, O_e	\dots	S_f, O_f
\vdots	\vdots	\vdots	\ddots	\vdots
S_m	S_x, O_x	S_y, O_y	\dots	S_z, O_z

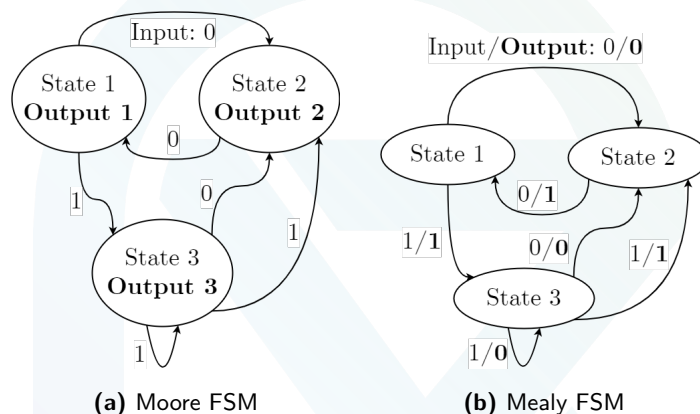


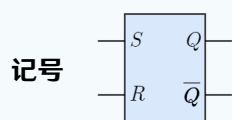
图 2.7: 一般的状态转换图

定义 2.3.4. 锁存器

锁存器 (latch) 是一个存储 1-bit 状态的电路单元, 其输出信号可以保持, 直到特定的输入信号触发重置或写入。

依据具体实现方式, 锁存器可以分为 SR 锁存器、D 锁存器等类型。

元件 2.2. SR 锁存器



特性 下一个输出信号 Q^+ 与输入信号 R 、 S 和当前输出信号 Q 的关系为 $Q^+ = S + R'Q$, 其中 $SR = 0$ 。

用门电路实现 SR 锁存器, 如图 2.8a 所示。

注 2.1. 要求 $SR = 0$ 的原因

当 $S = R = 1$ 时, SR 锁存器的输出保持 $Q = Q' = 0$, 这不符合输出的逻辑要求; 并且, 此后若 S 和 R 同时变为 0, 输出信号将不稳定, 可能导致电路失效。因此, SR 锁存器的真值表如表 2.8b 所示。

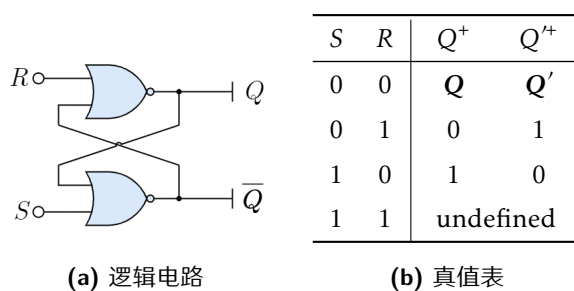


图 2.8: SR 锁存器

可以给 SR 锁存器加上门控使能信号, 如图 2.9 所示, 仅当使能信号 $C = 1$ 时, SR 锁存器才能改变状态。在门控信号上加上时钟信号 CLK, 就可以实现对输入的定时采样。其 Q^+ 与 Q 的关系可写为

$$Q^+ = C \cdot (S + R'Q) + C'Q$$

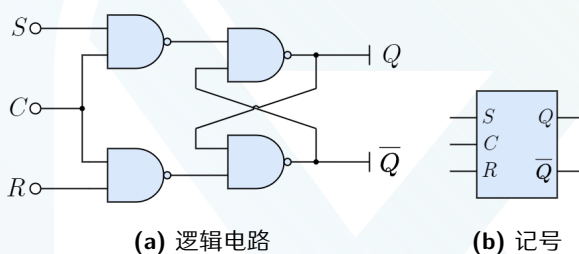
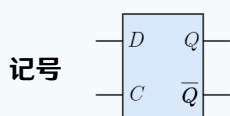


图 2.9: 门控 SR 锁存器

元件 2.3. D 锁存器



特性 $C = 1$ 时, 下一个输出信号 Q^+ 与输入信号 D 的关系为 $Q^+ = D$, 即输出跟踪输入; $C = 0$ 时, 输出信号保持。

用门电路实现 D 锁存器, 如图 2.10a 所示。此外, 也可以用传输门和非门实现 D 锁存器, 如图 2.10b 所示。

注 2.2. 门控 SR 锁存器和 D 锁存器的时序差异

门控 SR 锁存器和 D 锁存器都是在信号 C 的下降沿时锁定, 但后者的锁定输出只与下降沿的输入有关, 而前者的锁定输出还受到 S 与 R 都为低电平前的输入的影响。

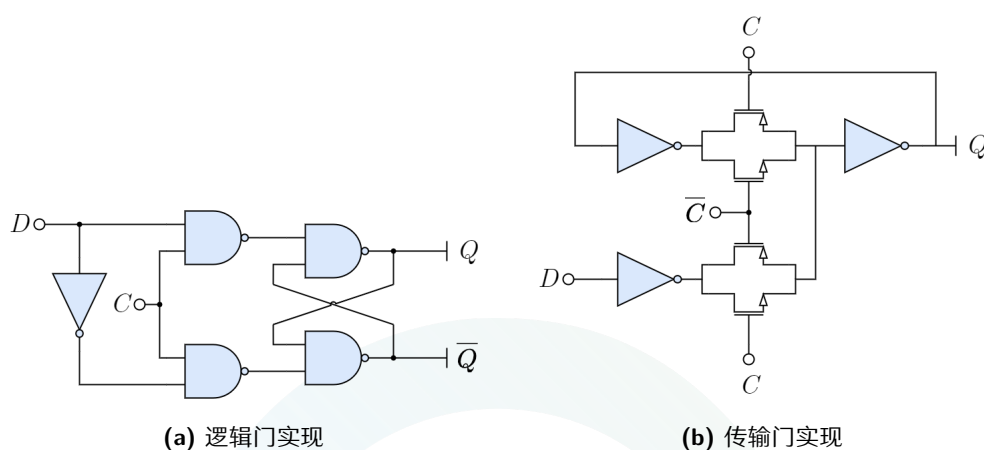


图 2.10: D 锁存器

注 2.3. 锁存器的时序参数

锁存器的性能受到以下几个时序参数的影响：

- **最小脉冲宽度**：能够稳定锁存的最短输入脉冲宽度，记作 $t_{w(\min)}$ ；
- **延迟时间**：输入信号发生变化到输出信号发生变化的时间，记作 t_{pd} ，分为 **上升延迟时间** t_{pLH} 和 **下降延迟时间** t_{pHL} ；
- **建立时间、保持时间**：锁存操作开始、结束之前、之后输入信号需要保持的时间，分别记作 t_{su} 和 t_h 。如果信号不满足建立时间和保持时间约束，输出结果会呈现不确定的亚稳态。

各个参数的具体定义如图 2.11 所示。

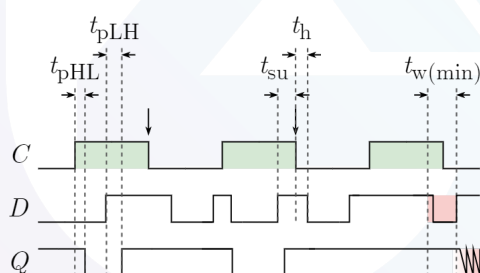


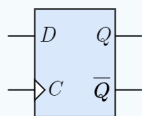
图 2.11: 锁存器的时序参数

B) 触发器**定义 2.3.5. 触发器**

触发器 (flip-flop) 是一种只在触发沿把输入信号写入输出信号的电路。

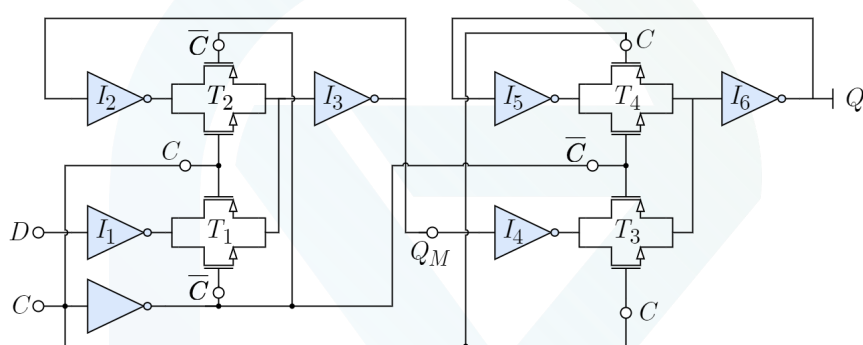
元件 2.4. D 触发器(DFF)

记号

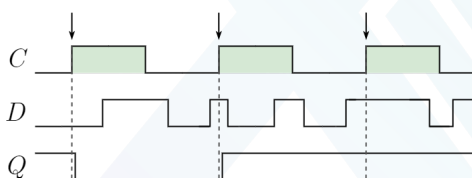


特性 在时钟信号 C 的上升沿, 下一个输出信号 Q^+ 与输入信号 D 的关系为 $Q^+ = D$; 其他时间, 输出信号保持。

DFF 的一种常见的实现方式是两个 D 锁存器级联, 如图 2.12a 所示, 其中左侧的 D 锁存器用于存储输入信号, 称为**主锁存器**; 右侧的 D 锁存器用于控制输出信号, 称为**从锁存器**。



(a) 逻辑电路



(b) 时序图

图 2.12: D 触发器

在一个时钟周期中, 时钟信号 C 为低电平时, 主锁存器的输出信号 Q_M 随输入信号 D 变化, 从锁存器的输出信号 Q 保持不变; 时钟信号 C 为高电平时, 主锁存器的输出信号 Q_M 保持不变, 从锁存器的输出信号 Q 将 Q_M 的值输出。因此, 在 C 的上升沿, DFF 表现为将输入信号 D 写入输出信号 Q 锁存一个周期。DFF 的时序图如图 2.12b 所示。

注 2.4. DFF 的时序参数

DFF 的时序参数与锁存器类似, 包括:

- **建立时间 t_{su}** : 输入信号在时钟信号上升沿前需要传输到 T_2 两端以避免竞争, 这一过程所需要保持的时间;
- **保持时间 t_h** : 输入信号在时钟信号上升沿后需要维持到 T_1 完全关闭以避免破坏写入的值, 这一过程所需要保持的时间;
 D 和 C 传播到 T_1 的时间一般相等, 因此通常原生满足保持时间约束, 即 $t_h = 0$ 。
- **输出响应时间 t_{c-q}** : 输入信号在时钟信号上升沿后到输出信号发生变化的时间。

当两个 DFF 级联时:

- 第一个 DFF 的输入信号 D_1 在开始采样 t_{c-q} 时间后才会传输到其输出 Q_1 , 第二个 DFF 的输入信号 D_2 在开始采样前 t_{su} 时间内需要保持不变, 因此留给中间的组合逻辑电路的时间为 $t_{cyc} - t_{c-q} - t_{su}$, 逻辑计算必须在这段时间内完成, 这称为**建立时间约束**;
- 第一个 DFF 的输出信号 Q_1 在 t_{c-q} 时间后才会更新, 经过组合逻辑电路后才传输到 D_2 , 此前 D_2 因前一次采样而需要保持不变 t_h 时间, 因此组合逻辑电路的延迟时间不能小于 $t_h - t_{c-q}$, 这称为**保持时间约束**。

2.3.3 时序逻辑电路分析

在如图 2.13 所示的一般时序逻辑电路中, 有以下几个要素需要分析:

- **输出方程**: 根据输入信号 x 和当前状态 s 计算输出信号 $y = f(x, s)$;
- **激励方程**: 根据输入信号 x 和当前状态 s 计算触发器的输入信号 $e = g(x, s)$;
- **状态转移方程 (次态方程)**: 根据触发输入 e 和当前状态 s 计算下一个状态 $s^+ = h(e, s)$;

可以通过状态转移图、状态转移表、时序图和电路行为描述等方式进行分析。

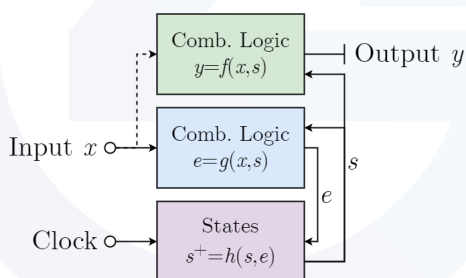


图 2.13: 一般时序逻辑电路

2.3.4 时序逻辑电路设计

例题 2.3.1. 设计一个训练累计器, 输入每天的跑步距离 $Today$, 根据给定的目标跑步距离 $Goal$ 计算剩余距离 $Left$ 并输出。

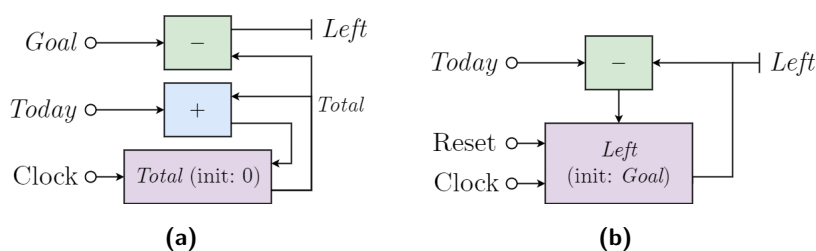


图 2.14: 训练累计器的 FSM 设计

表 2.5: 训练累计器的状态转移表

<i>Left</i>	<i>Today</i>					Output
	0	1	...	t	...	
1000	1000	999	...	$1000 - t$...	1000
999	999	998	...	$999 - t$...	999
\vdots	\vdots	\vdots	\ddots	\vdots	\ddots	\vdots
t	t	$t - 1$...	0	...	t
$t - 1$	$t - 1$	$t - 2$...	0	...	$t - 1$
\vdots	\vdots	\vdots	\ddots	\vdots	\ddots	\vdots
0	0	0	...	0	...	0

解. 训练累计器可以抽象成如图 2.14 所示的 FSM, 其中图 2.14b 显然更优。根据 FSM 设计, 可以得到状态转移表如表 2.5 所示。

⑤

注 2.5. 时序逻辑电路设计方法

时序逻辑电路的设计包括以下几个步骤:

1. 抽象出一个有限状态机, 定义初始状态、输入信号、输出信号、输出函数和状态转移函数;
2. 状态化简、合并;
3. 状态分配;
4. 确定激励方程和输出方程;
5. 画出逻辑电路图。

A) 寄存器

定义 2.3.6. 寄存器

寄存器 (register) 是一个能够存储多个 bit 状态的电路单元, 其输出信号可以保持, 直到特定的输入信号触发重置或写入。仅有存储功能的寄存器称为**存储寄存器**, 还带有移位功能的寄存器称为**移位寄存器**。

右移、左移移位寄存器可简单地通过串联多个 DFF 实现, 如图 2.15 所示。

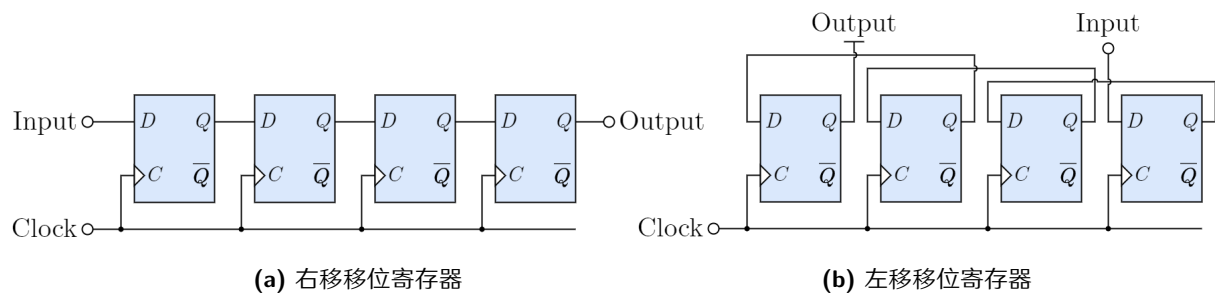


图 2.15: 移位寄存器

B) 计数器