

目 录

第 0 章 绪论	(1)
0.1 数字技术发展史	(1)
0.2 脉冲信号与数字信号	(2)
0.3 数制与数制转换	(3)
0.4 算术运算与逻辑运算	(7)
第 1 章 逻辑函数	(8)
1.1 逻辑函数	(8)
1.1.1 基本逻辑运算	(8)
1.1.2 逻辑函数的基本定理	(10)
1.1.3 逻辑函数的基本运算规则	(11)
1.2 逻辑函数的标准型	(12)
1.2.1 逻辑函数的两种标准形式	(12)
1.2.2 将逻辑函数变换为标准型	(14)
1.3 几种常用的复合逻辑及其逻辑门	(16)
1.3.1 3 种基本逻辑门	(16)
1.3.2 常用的复合逻辑及其逻辑门	(17)
*1.3.3 正逻辑与负逻辑	(18)
1.4 逻辑函数的简化	(19)
1.4.1 逻辑代数简化法	(19)
1.4.2 卡诺图简化法	(21)
*1.5 卡诺图的其他应用	(28)
*1.6 多输出函数的简化	(30)
小结	(33)
思考题和习题	(34)
第 2 章 集成逻辑门	(37)
2.1 数字集成电路概述	(37)
2.2 晶体二极管和三极管的开关特性	(38)
2.2.1 晶体二极管的开关特性	(38)

2.2.2	晶体三极管的开关特性	(40)
2.3	TTL 逻辑门	(43)
2.3.1	浅饱和型 TTL 与非门	(43)
2.3.2	TTL 与非门的性能指标	(46)
2.3.3	其他系列 TTL 与非门	(49)
2.3.4	集电极开路与非门(OC 门)	(51)
2.3.5	三态 TTL 与非门(TSL)	(54)
2.4	射极耦合逻辑门(ECL)	(55)
2.4.1	ECL 反相器	(56)
2.4.2	ECL 或/或非门	(56)
2.5	MOS 逻辑门	(57)
2.5.1	MOS 管	(58)
2.5.2	MOS 管的静态特性	(59)
2.5.3	NMOS 逻辑门	(62)
2.6	CMOS 逻辑门	(64)
2.6.1	CMOS 反相器	(64)
2.6.2	CMOS 逻辑门	(65)
*2.6.3	其他系列 CMOS 门	(67)
2.6.4	CMOS 逻辑门的性能指标	(69)
2.6.5	CMOS 门使用中的几个问题	(71)
*2.7	集成逻辑门使用中的几个问题	(72)
	小结	(74)
	思考题和习题	(75)

第 3 章 组合逻辑电路 (78)

3.1	组合逻辑电路的设计	(78)
3.1.1	提供原、反变量输入条件的组合电路设计	(79)
3.1.2	提供原变量输入条件的组合电路设计	(84)
*3.1.3	多端输出组合电路的设计	(84)
3.2	组合逻辑电路的分析	(87)
3.3	编码与编码器	(88)
3.3.1	编码	(88)
3.3.2	编码器	(91)
3.4	译码与译码器	(95)
3.4.1	译码器的设计	(95)

3.4.2	中规模集成通用译码器	(97)
3.4.3	码制转换译码器	(103)
3.4.4	中规模集成数字显示译码器	(107)
3.5	二进制运算电路	(111)
3.5.1	半加器	(111)
3.5.2	全加器	(112)
3.5.3	中规模集成 4 位加法器	(113)
3.5.4	数值比较器	(115)
3.6	数据选择器	(119)
3.6.1	数据选择器的类型及主要性能	(119)
3.6.2	用数据选择器设计组合逻辑电路	(123)
*3.7	数据分配器	(126)
*3.8	奇偶产生器/检验器	(128)
3.8.1	奇偶产生器/检验器的设计	(128)
3.8.2	中规模集成奇偶产生器/检验器	(129)
3.9	组合逻辑电路中的冒险现象	(130)
3.9.1	组合电路中的竞争与冒险	(130)
3.9.2	逻辑冒险的检查和消除	(132)
3.9.3	功能冒险的消除	(135)
	小结	(135)
	思考题和习题	(136)
第 4 章	集成触发器	(140)
4.1	RS 触发器	(140)
4.1.1	基本 RS 触发器	(140)
4.1.2	时钟 RS 触发器	(142)
4.1.3	主从 RS 触发器	(145)
4.2	主从 JK 触发器	(146)
4.3	维持阻塞型 D 触发器	(149)
4.4	T 和 T' 触发器	(151)
4.4.1	T 触发器	(151)
4.4.2	T' 触发器	(152)
4.5	边沿型触发器	(152)
4.5.1	TTL 边沿型 JK 触发器	(153)
*4.5.2	CMOS 边沿型 D 触发器	(154)

*4.5.3	CMOS 边沿型 JK 触发器	(155)
4.6	集成触发器使用中的几个问题	(156)
4.6.1	集成触发器的选用	(156)
4.6.2	激励表的应用	(157)
4.6.3	触发器类型转换	(157)
4.6.4	如何画工作波形	(158)
小结	(160)
思考题和习题	(160)
第 5 章	时序逻辑电路	(164)
5.1	时序逻辑电路概述	(164)
5.2	同步时序电路的设计	(166)
5.2.1	同步时序电路的设计方法	(166)
5.2.2	拟定原始状态表或状态图	(166)
5.2.3	状态简化	(169)
5.2.4	状态分配	(173)
5.2.5	确定激励函数和输出函数	(173)
5.2.6	设计举例	(175)
5.3	同步时序电路的分析	(177)
5.4	寄存器和锁存器	(181)
5.4.1	寄存器	(181)
5.4.2	锁存器	(184)
5.5	移位寄存器	(185)
5.5.1	移位寄存器的设计	(185)
5.5.2	中规模集成移位寄存器	(187)
*5.5.3	移位寄存器的应用	(191)
5.6	计数器	(193)
5.6.1	计数器的功能和分类	(193)
5.6.2	同步计数器	(193)
5.6.3	异步计数器	(204)
5.6.4	中规模集成计数器	(208)
5.6.5	采用中规模集成计数器设计任意进制计数器	(215)
5.7	序列信号发生器	(218)
小结	(222)
思考题和习题	(222)

第 6 章 硬件描述语言	(228)
6.1 硬件描述语言概述	(228)
6.1.1 现代 EDA 技术简介	(228)
6.1.2 硬件描述语言及其开发软件简介	(230)
6.2 ABEL-HDL 的基本语法	(232)
6.2.1 ABEL-HDL 的特点	(232)
6.2.2 ABEL-HDL 的语法规则	(233)
6.2.3 ABEL-HDL 中对基本逻辑器件的描述	(242)
6.3 ABEL-HDL 模块源文件结构	(244)
6.3.1 ABEL-HDL 模块基本结构框架	(244)
6.3.2 ABEL-HDL 模块结构	(245)
*6.3.3 ABEL-HDL 中的层次设计方法	(254)
6.4 ISP Synario 软件及其使用方法	(257)
6.4.1 ISP Synario 软件的特点及安装	(257)
6.4.2 ISP Synario 设计起点	(259)
6.4.3 ISP Synario 原理图设计	(260)
6.4.4 ISP Synario ABEL-HDL 设计	(263)
6.4.5 设计结果的测试与仿真	(264)
6.5 利用 ABEL-HDL 源文件描述真值表和逻辑函数	(266)
*6.6 利用 ABEL-HDL 源文件描述组合逻辑电路	(269)
6.7 利用 ABEL-HDL 源文件描述时序逻辑电路	(272)
小结	(276)
思考题和习题	(276)
第 7 章 半导体存储器	(279)
7.1 随机存储器(RAM)	(279)
7.1.1 RAM 的基本结构	(280)
7.1.2 RAM 的典型产品介绍	(284)
7.1.3 RAM 的容量扩展	(286)
7.2 只读存储器(ROM)	(287)
7.2.1 只读存储器(ROM)	(288)
7.2.2 可擦可编程只读存储器(EPROM)	(290)
7.2.3 电可擦可编程只读存储器(E ² PRO)	(292)
*7.2.4 快闪只读存储器 (U 盘)	(294)
*7.2.5 ROM 的应用	(295)

7.3 双端口随机存储器 (DPRAM)	(297)
7.3.1 电路结构	(297)
7.3.2 工作原理	(298)
小结	(299)
思考题和习题	(299)
第 8 章 可编程逻辑器件	(301)
8.1 可编程逻辑器件的结构和分类	(301)
8.1.1 PLD 器件表示方法	(301)
8.1.2 PLD 的基本结构	(303)
8.1.3 PLD 的分类	(304)
8.1.4 可编程逻辑器件的优点	(305)
8.2 可编程通用阵列逻辑(GAL)	(306)
8.2.1 GAL 器件的电路结构	(307)
8.2.2 输出逻辑宏单元(OLMC)的组态结构	(312)
*8.2.3 编程应用中的几个技术问题	(314)
*8.2.4 用 GAL 设计逻辑电路	(316)
8.3 可编程逻辑阵列(PLA)	(318)
8.3.1 PLA 的工作原理	(318)
8.3.2 用 PLA 设计逻辑电路	(319)
8.4 复杂可编程逻辑器件 (CPLD)	(321)
8.4.1 MAX 7000S 系列 CPLD 器件	(321)
8.4.2 ispLSI 1032EA 器件	(328)
8.5 现场可编程门阵列(FPGA)	(332)
8.5.1 FPGA 的电路结构	(332)
8.5.2 FPGA 的连接方法	(340)
8.5.3 FPGA 的编程和加载方法	(343)
*8.6 用 CPLD、FPGA 设计数字系统	(347)
8.6.1 大规模可编程逻辑器件的设计流程	(347)
8.6.2 用 CPLD、FPGA 设计数字跑表	(348)
小结	(356)
思考题和习题	(358)
第 9 章 脉冲单元电路	(359)
9.1 概述	(359)

9.1.1 脉冲信号与脉冲电路	(359)
9.1.2 脉冲信号的主要参数	(359)
*9.1.3 集成运算放大器的传输特性	(360)
9.2 施密特触发器	(361)
9.2.1 集成门构成的施密特触发器	(361)
9.2.2 集成运放构成的施密特触发器	(364)
9.3 单稳态触发器	(366)
9.3.1 集成运放构成的单稳态触发器	(366)
9.3.2 单片集成单稳态触发器	(367)
9.3.3 单稳态触发器的应用	(371)
9.4 多谐振荡器	(372)
9.4.1 集成运放构成的多谐振荡器	(373)
9.4.2 施密特触发器构成的多谐振荡器	(374)
9.4.3 单稳态触发器构成的多谐振荡器	(375)
*9.4.4 石英晶体振荡器	(376)
*9.5 555 定时器及其应用	(376)
9.5.1 555 定时器的电路结构与功能	(376)
9.5.2 用 555 定时器构成施密特触发器	(378)
9.5.3 用 555 定时器构成单稳态触发器	(379)
9.5.4 用 555 定时器构成自激多谐振荡器	(380)
小结	(382)
思考题和习题	(383)
第 10 章 模数及数模转换技术	(386)
10.1 模数及数模转换技术概述	(386)
10.2 数模转换器	(387)
10.2.1 数模转换器的基本原理	(387)
10.2.2 电压型数模转换器	(388)
*10.2.3 电流型数模转换器	(391)
10.2.4 中规模集成数模转换器	(393)
10.2.5 数模转换器的性能指标	(398)
10.3 模数转换器	(400)
10.3.1 模数转换的基本原理	(400)
10.3.2 直接转换型模数转换器	(402)
*10.3.3 间接转换型模数转换器	(407)

10.3.4 模数转换器的性能指标	(408)
*10.4 集成模数转换器	(409)
小结	(414)
思考题和习题	(415)
附录一 常用逻辑单元图形符号对照表	(416)
附录二 本书中的文字符号和图形符号及其说明	(418)
附录三 汉英名词、缩写词对照表	(420)
参考文献	(424)

第 0 章 绪 论

内容提要 本章简介了数字技术发展历史, 脉冲信号与数字信号; 重点讨论了数制转换方法, 包括二进制数与十进制数、二进制数与八(十六)进制数以及任意进制数与任意进制数之间的相互转换方法。

0.1 数字技术发展史

数字技术是研究数字电路和它在各种学科领域应用的一门科学, 它的发展大致可分为 5 个阶段。

数字技术早在 19 世纪末就开始获得工程应用, 电报通信就是一个简单的二值数字系统。20 世纪 30 年代, 电话逐渐普及, 在众多电话机中依靠人工方法来选取所需要的对象已经不可能了, 因此, 拨号式自动电话交换系统应运而生。该系统在数字技术中首次引进了信息存储的新功能。特别需要指出的是在该系统的研究中, 英国数学家乔治·布尔(George Boole)早在 1847 年创立了布尔代数理论获得工程应用, 并在随后的实践中丰富和发展, 逐渐形成了近代开关理论: 继电-触点网络理论(Relay-Contact Network Theory), 在这个理论基础上建立了一套对数字逻辑电路的分析方法和设计方法, 为以后数字技术的发展奠定了理论基础。

进入 20 世纪 40 年代, 许多军事科学的研究迫切需要进行快速的大量计算, 例如, 火箭的飞行轨迹和自动控制等, 这就要求计算工作自动化, 因此, 1946 年世界上出现了第一台以电子管为基本元件的电子计算机(名为 ENIAC)。但是, 由于电子管在性能指标上存在许多缺陷, 因此, 在晶体管出现以前, 采用电子管为基本器件的一些数字设备只是在自动电话交换系统、数字通信和专用计算机等少数学科领域获得应用, 这是数字技术发展的初期阶段。

从 20 世纪 60 年代开始, 就在数字技术中广泛采用晶体管代替电子管作为基本器件。由于晶体管具有体积小、功耗低、工作速度高和工作寿命长等优点, 使数字设备缩小了体积、降低了功耗、提高了工作速度和可靠性, 因而, 为数字技术的推广应用创造了条件。在计算机、数字通信、测量仪表和自动控制等学科领域中都开始应用数字技术。这可称为数字技术发展的第二阶段。

20 世纪 60 年代末至 70 年代中期是数字技术发展的第三阶段。这一时期在数字技术中广泛采用集成电路作为基本器件。集成电路可以把成千上万的晶体管、电阻、

电容等元件以及它们的连线都制作在一个面积很小的芯片上,它的应用使数字设备的体积缩小、功耗降低和可靠性大幅度提高,特别是集成电路的价格随着生产工艺技术的进步而愈来愈低廉。因此,数字技术开始进入国民经济的各行各业中,在数字雷达、卫星电视、自动控制、遥控、遥测、医学等学科领域都获得应用。

20 世纪 70 年代中期至 80 年代中期,由于微电子学的发展和集成电路生产工艺的进步,集成电路在集成度和工作速度等性能指标上取得突破性进展。大规模和超大规模集成电路的生产技术已经非常成熟。一块芯片上可以集成几百万、甚至上千万个元件。因而,出现了将一台计算机都集成在一块芯片上的微型计算机。它的出现标志着数字技术发展进入了第四阶段,这是数字技术全面迅猛发展的一个阶段,不仅在计算机、通信、雷达、卫星电视、测量仪表、宇航、医学及生物工程等学科领域获得普遍应用,而且遍及人们日常生活中的各个方面,如交通自动控制、程控电话全电子交换系统、可视电话、家庭炊具自动控制等。

从 20 世纪 80 年代中期开始,超大规模的专用集成电路 ASIC^①的制作技术已趋成熟。厂商可以代客户将他们设计的十分庞大的数字系统制作在一块芯片上,客户便得到了所需要的系统级芯片;与此同时,涌现了各种用户可编程逻辑器件,例如,可编程阵列逻辑 PAL,可编程通用阵列逻辑 GAL,现场可编程门阵列 FPGA 等等。它们有配套的系列产品,用户只要将自己设计的数字系统通过编制一定的程序(通常称为软件),再将程序输入到这些可编程逻辑器件中便可得到所需的芯片。这些专用的和通用的系统级芯片的应用不但进一步提高了设备的性能,而且将数字系统的设计、安装和调试融为一体,并且都是在计算机上来完成的,这就彻底更新了数字设备传统的研制方法,大大缩短了设备的研制周期和降低了设备成本,使之成为当今数字技术发展的主要方向。

通过对数字技术发展史的简要回顾,我们已经可以得出结论:在人类迈向信息社会的进程中,数字技术起到了关键性的作用。

0.2 脉冲信号与数字信号

在自然界中存在着许多物理量,其中有一些物理量,如温度、湿度、压力、速度等,它们在时间上和数值上都具有连续变化的特点,在一定范围内可以取任意实数值,通常称这种连续变化的物理量为模拟量。表示模拟量的电信号称为模拟信号。模拟信号分为两类:一是正弦信号,二是脉冲信号。广义的来说,凡不具有连续变化形状的信号,都可以通称为脉冲信号。如方波、矩形波、尖脉冲、锯齿波、钟形

^① 参见“附录三 汉英名词、缩写词对照表”,(下同)

脉冲以及梯形波等等。完成脉冲信号产生、传输、变换和处理的电路称为脉冲电路。还有一类物理量在时间和数值上是离散的，它们的大小以及每次的增减变化都是某个最小单位的整数倍。例如产量，若最小单位为吨，则产量是指在一些离散时刻完成产品有多少吨，显然它只能以吨为单位增加或减少。这一类物理量称为数字量。表示数字量的电信号称为数字信号，完成数字信号产生、传输和处理的电路就称为数字电路。

在数字电路中最常采用的只有 0、1 两种数值所组成的数字信号，这种二值数字信号又称为二进制信号。这类信号中的数值 1 或 0 可以用电平的高或低来表示；也可以用脉冲的有或无来表示，如图 0.2.1 所示。

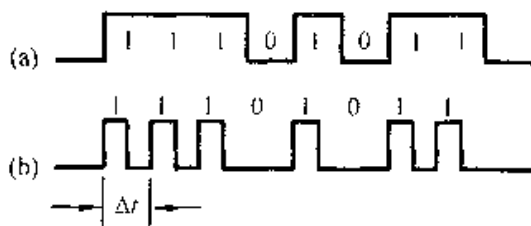


图 0.2.1 数字信号

图中，每个 1 和 0 的持续时间都为 Δt ，称为一位(1bit)，或者称为一拍。图(b)所示数字信号波形是以有脉冲代表 1、无脉冲代表 0，称为脉冲型数字信号或归 0 型数字信号；图(a)是以高电平代表 1、低电平代表 0，称为电平型数字信号或不归 0 型数字信号。

0.3 数制与数制转换

按进位规则进行计数称为进位计数制，简称数制。在日常生活中最广泛使用的是十进制，其他还有十二进制(一打)，六十进制(时钟)等；在数字电路和计算机中广泛使用的是二进制，八进制和十六进制。因此，经常需要在两种不同数制之间进行转换，这就是数制转换。

1. 数的表示方法

数的表示方法有两种：位置记数法和按位权展开法。例如：一见十进制数 345，大家马上便读出“三百四十五”，这就是位置记数法，一个数码在十进制数中处在不同数位时，它所代表的数值不同，3 在百位上，4 在十位上，5 在个位上。不同数位(位置)上的数码所具有的十进制的数值称为位权值，所以，100、10、1 称为十进制数的位权值。显然，十进制数的各个数位的位权值是 10 的幂。345 按位权展开，即可表示为

$$(345)_{10} = 3 \times 10^2 + 4 \times 10^1 + 5 \times 10^0$$

式中, 下角标 10 表示十进制。

对于一个有 n 位整数、 m 位小数的十进制数可以表示为

$$\begin{aligned}(N)_{10} &= a_{n-1} \cdots a_0 a_{-1} \cdots a_{-m} \\ &= a_{n-1} \times 10^{n-1} + \cdots + a_0 \times 10^0 + a_{-1} \times 10^{-1} + \cdots + a_{-m} \times 10^{-m} \\ &= \sum_{i=-m}^{n-1} a_i \times 10^i\end{aligned}\quad (0.1)$$

式中, a_i 表示 i 位数码, 它可以是十进制数码 0~9 中的任意一个; 10^i 表示 i 位的位权值; n 和 m 为正整数。

在一种数制中采用的数码个数称为基数, 所以, 十进制数的基数为 10。

在二进制数中, 只有 0 和 1 两个数码, 所以, 基数是 2, 各个数位的位权值是 2 的幂。任意一个二进制数按位权展开可表示为

$$\begin{aligned}(N)_2 &= a_{n-1} \cdots a_0 a_{-1} \cdots a_{-m} \\ &= a_{n-1} \times 2^{n-1} + \cdots + a_0 \times 2^0 + a_{-1} \times 2^{-1} + \cdots + a_{-m} \times 2^{-m} \\ &= \sum_{i=-m}^{n-1} a_i \times 2^i\end{aligned}\quad (0.2)$$

式中, a_i 表示数码 0 或 1; 2^i 表示 i 位数码的位权值; n 和 m 为正整数。

在八进制中, 有 8 个数码 (0~7), 所以基数是 8, 各个数位的位权值是 8 的幂; 在十六进制中, 有 16 个数码 (0~9, A~F), 所以基数是 16, 各个数位的位权值是 16 的幂。

上述表示方法可以推广到任意进制 R , 在 R 进制中, 有 R 个数码, 基数为 R , 其各位数码的位权值是 R 的幂。因此, 按位权展开法可以将任意一个 R 进制数表示为

$$\begin{aligned}(N)_R &= a_{n-1} \cdots a_0 a_{-1} \cdots a_{-m} \\ &= a_{n-1} \times R^{n-1} + \cdots + a_0 \times R^0 + a_{-1} \times R^{-1} + \cdots + a_{-m} \times R^{-m} \\ &= \sum_{i=-m}^{n-1} a_i \times R^i\end{aligned}\quad (0.3)$$

式中, a_i 表示 R 个数码中的任意一个; R^i 表示 i 位数码的位权值; n 和 m 为正整数。

2. 数制转换

(1) 二进制数转换为十进制数

法则 将被转换的二进制数按权相加。

例 0.1.1 将二进制数 $(11011.101)_2$ 转换为十进制数。

解 $(11011.101)_2 = 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1}$
 $+ 1 \times 2^{-2} = (27.625)_{10}$

上述法则可以推广应用到任意进制数至十进制数的转换。

例 0.1.2 将八进制数 $(345.67)_8$ 转换为十进制数。

解 $(345.67)_8 = 3 \times 8^2 + 4 \times 8^1 + 5 \times 8^0 + 6 \times 8^{-1} + 7 \times 8^{-2} = (229.859)_{10}$

(2) 十进制数转换为二进制数

将十进制数转换为二进制数, 需将十进制数的整数部分和小数部分分别进行转换, 然后将它们合起来。设整数部分为 N , 小数部分为 M 。

整数转换法则 首先采用基数除法, 即用基数 2 去除 N , 得到商 Q_1 和余数 b_0 ; 然后继续用 2 去除 Q_1 , 得到商 Q_2 和余数 b_1 。如此继续下去直至商为 0, 所得到的余数 $b_i \cdots b_1 b_0$ 便是相应的二进制数。

例 0.1.3 将 $(49)_{10}$ 转换为二进制数。

解 将 $(49)_{10}$ 连除以 2, 直到商为 0, 所得余数即为相应的二进制数。

2	49	余数	(a_i)	
2	24	1	a_0	—— 最低有效位 (LSB)
2	12	0	a_1	
2	6	0	a_2	
2	3	0	a_3	
2	1	1	a_4	
	0	1	a_5	—— 最高有效位 (MSB)

所以, $(49)_{10} = (110001)_2$ 。

小数转换法则 采用基数乘法, 即将 M 逐次乘以 2, 取乘积的整数部分作为二进制小数的各有关位, 而乘积的小数部分继续乘以 2, 直至最后乘积为 0 或达到满意精度为止。

例 0.1.4 将 $(0.706)_{10}$ 转换为二进制数, 要求其误差不大于 2^{-10} 。

解 由于要求转换误差不大于 2^{-10} , 而数码 a_{-10} 的权位值为 2^{-10} , 所以小数部分应有 10 位。

$0.706 \times 2 = 1.412$	$a_{-1} = 1$
$0.412 \times 2 = 0.824$	$a_{-2} = 0$
$0.824 \times 2 = 1.648$	$a_{-3} = 1$
$0.648 \times 2 = 1.296$	$a_{-4} = 1$
$0.296 \times 2 = 0.592$	$a_{-5} = 0$
$0.592 \times 2 = 1.184$	$a_{-6} = 1$
$0.184 \times 2 = 0.368$	$a_{-7} = 0$
$0.368 \times 2 = 0.736$	$a_{-8} = 0$
$0.736 \times 2 = 1.472$	$a_{-9} = 1$

$$0.472 \times 2 = 0.944 \quad a_{-10} = 0$$

因此, $(0.706)_{10} = (0.1011010010)_2$ 。

例 0.1.5 将 $(0.39)_{10}$ 转换为二进制数, 要求精度为 1%。

解 将 $(0.39)_{10}$ 转换为二进制数的精度取决于二进制数小数部分的位数 m , 即 m 应依据精度要求确定, 令 $2^{-m} \leq 1\%$, 则可得

$$m \log_{10} 2 \geq \log_{10} 100$$

因此, $m \geq 6.6$ 。取 $m=7$, 则

$$\begin{array}{ll} 0.39 \times 2 = 0.78 & a_{-1} = 0 \\ 0.78 \times 2 = 1.56 & a_{-2} = 1 \\ 0.56 \times 2 = 1.12 & a_{-3} = 1 \\ 0.12 \times 2 = 0.24 & a_{-4} = 0 \\ 0.24 \times 2 = 0.48 & a_{-5} = 0 \\ 0.48 \times 2 = 0.96 & a_{-6} = 0 \\ 0.96 \times 2 = 1.92 & a_{-7} = 1 \end{array}$$

因此, $(0.39)_{10} = (0.0110001)_2$ 。

剩余误差 $e < 2^{-7} \approx 0.7\%$ 。显然在允许的精度之内。

上述方法可以推广到十进制数转换为任意进制数尺, 此时只需将基数改为 R 。

(3) 二进制数与八(十六)进制数的转换

二进制数转换为八(十六)进制数的方法 由于 3(4) 位二进制数码可以构成 1 位八(十六)进制数, 并且有唯一的对应关系, 所以, 它们之间的转换十分简便。将二进制数的整数由小数点向左每 3(4) 位分为一组, 最后不足 3(4) 位的补 0; 小数则由小数点向右每 3(4) 位分为一组, 不足 3(4) 位的补 0。然后把每一组 3(4) 位二进制数用相应的八(十六)进制数代替即可转换成八(十六)进制数。

例 0.1.6 将 $(01100110.0111011)_2$ 转换成八进制数。

$$\begin{array}{ccccccc} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ \hline & \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} \\ & 1 & 4 & 6 & 3 & 5 & 4 & & & & & & & & & & & \end{array}$$

因此, $(01100110.0111011)_2 = (146.354)_8$ 。

八(十六)进制数转换成二进制数的方法 无论是整数或是小数, 只需把八(十六)进制数的每一位数码用相应 3(4) 位二进制数代替即可。

例 0.1.7 将 $(5A.BC)_{16}$ 转换为二进制数。

$$\text{解} \quad (5A.BC)_{16} = (01011010.10111100)_2$$

(4) (任意) R 进制数与(任意) S 进制数之间的转换

将 R 进制数先转换为十进制数, 然后再将该十进制数转换为 S 进制数, 反之亦

然。

例如，将八进制数转换为十二进制数，可先将八进制数转换为十进制数，然后再将这个十进制数转换为十二进制数，其转换方法已如前述，读者可举例练习。

0.4 算术运算与逻辑运算

当用两个二进制数码表示两个数值时，它们之间可以进行数值运算。这种运算称为算术运算。二进制数算术运算同十进制数算术运算的规则基本相同，唯一的区别在于相邻两位之间的关系是“逢2进1”（加法），“借1当2”（减法）。

例如，两个4位二进制数的算术运算有

加法运算

$$\begin{array}{r} 1\ 0\ 0\ 1 \\ +\ 0\ 1\ 0\ 1 \\ \hline 1\ 1\ 1\ 0 \end{array}$$

减法运算

$$\begin{array}{r} 1\ 0\ 0\ 1 \\ -\ 0\ 1\ 0\ 1 \\ \hline 0\ 1\ 0\ 0 \end{array}$$

乘法运算

$$\begin{array}{r} 1\ 0\ 0\ 1 \\ \times\ 0\ 1\ 0\ 1 \\ \hline 1\ 0\ 0\ 1 \\ 0\ 0\ 0\ 0 \\ 1\ 0\ 0\ 1 \\ 0\ 0\ 0\ 0 \\ \hline 0\ 1\ 0\ 1\ 1\ 0\ 1 \end{array}$$

除法运算

$$\begin{array}{r} 1\ 1\ 1\dots \\ 0\ 1\ 0\ 1 \overline{) 1\ 0\ 0\ 1} \\ \underline{0\ 1\ 0\ 1} \\ 1\ 0\ 0\ 0 \\ \underline{0\ 1\ 0\ 1} \\ 0\ 1\ 1\ 0 \\ \underline{0\ 1\ 0\ 1} \\ 0\ 0\ 1 \end{array}$$

在数字电路中，1位二进制数码0和1不仅可以表示数值大小，而且可以表示两种对立的逻辑状态。例如，可用0和1分别表示一个事件的真和伪、有和无、好和坏，或者表示电位的低和高、脉冲的无和有、电路的断和通、电灯的熄灭和发亮等。这里，0和1是代表两种逻辑状态的符号，它们的含义完全由人们事先约定。当二进制数码0和1表示逻辑状态时，它们之间可以按照某种因果关系进行运算，这种运算称为逻辑运算，它同算术运算有本质上的差别。在下一章中，将重点介绍逻辑运算的各种规律。

第1章 逻辑函数

内容提要 本章重点讨论逻辑函数的基本概念、主要定律和常用的运算规则，在此基础上介绍了几种逻辑函数的简化方法。其中，卡诺图简化法是重点内容，它是分析和设计逻辑电路的基础。本章最后还讨论了卡诺图的其他应用。

数字电路是处理数字信号的电路，它主要用于对数字信号进行存储、变换和运算。数字电路的基本单元是开关器件，或者是作开关应用的各种电子器件。开关器件只有“接通”和“断开”两种状态，可采用逻辑变量(二值变量)来描述。1847年英国数学家乔治·布尔(George Boole)首先提出了描述客观事物逻辑关系的数学方法——布尔代数。1938年，克劳德·香农(Claude E. Shannon)将布尔代数用于设计电话继电器开关电路，因此，又称为开关代数。随着数字技术的发展，布尔代数已成为分析和设计数字逻辑电路的数学基础，故又称为逻辑代数，它是研究二值数字逻辑电路的一种数学工具。

1.1 逻辑函数

逻辑函数是研究有某种规律的变量之间的因果关系的。逻辑函数中的变量只有两种可能的取值，即为0或1。我们把这种二值变量称为逻辑变量(布尔变量或开关变量)或简称变量。对于 n 个逻辑变量(A_1, A_2, \dots, A_n)的逻辑函数，当 n 个逻辑变量取任意一组确定值之后，逻辑函数 $F(A_1, A_2, \dots, A_n)$ 的值也就被唯一地确定了，显然也只有0或1两种可能的取值。例如，一个两变量的逻辑函数 $F(A, B)$ ，假设对于变量 A, B 的4种可能取值00, 01, 10, 11，其 $F(1, 1)=1$ ；那么，逻辑函数与变量的所有可能取值的一一对应关系可以用列表方式表示，如表1.1.1所示。这种表称为真值表，它在研究数字逻辑电路中获得广泛应用。

1.1.1 基本逻辑运算

在二值逻辑函数中，最基本的逻辑运算有与逻辑、或逻辑、非逻辑等3种运算。

1. 与逻辑运算

与逻辑运算简称与运算，又称为逻辑乘，通常用符号“ \cdot ”表示，称之为与。

两个变量的与运算的逻辑表达式为

$$F=A \cdot B \quad (1.1.1)$$

n 个变量与运算的逻辑表达式为

$$F=A_1 \cdot A_2 \cdot \cdots \cdot A_n \quad (1.1.2)$$

式(1.1.1)所表示的逻辑关系也可以用真值表来表示,如表 1.1.1 所示。由表可见:只有在 A 、 B 取值均为 1 时,逻辑函数 F 才为 1;其他取值都使 F 为 0。

2. 或逻辑运算

或逻辑运算简称或运算,又称为逻辑加,通常用符号“+”表示,称之为或。两个变量或运算的逻辑表达式为

$$F=A+B \quad (1.1.3)$$

n 个变量或运算的逻辑表达式为

$$F=A_1+A_2+\cdots+A_n \quad (1.1.4)$$

式(1.1.3)所表示的或运算也可以用真值表 1.1.2 来表示。由表可见:只要有一个变量取值为 1,逻辑函数 F 的值就为 1。

3. 非逻辑运算

非逻辑运算简称非运算,又称为反相运算。非运算的逻辑表达式为

$$F=\bar{A} \quad (1.1.5)$$

式中, \bar{A} 读作“ A 非”,通常在变量字符上方画“—”表示非,真值表如表 1.1.3 所示。

表 1.1.1 与逻辑真值表

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

表 1.1.2 或逻辑真值表

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

表 1.1.3 非逻辑真值表

A	F
0	1
1	0

以上定义的 3 种基本逻辑运算统称为逻辑运算;3 种运算符号“ \cdot ”、“+”、“—”统称为逻辑运算符;式(1.1.1)~式(1.1.5)统称为逻辑函数表达式或逻辑表达式,简称逻辑式,它是描述逻辑函数与逻辑变量之间关系的表达式。逻辑式和真值表都是描述逻辑函数的重要工具。

实际应用中会出现一些复杂的逻辑式,它们都是由一些基本逻辑运算符组合而成的。当 3 种运算符混合运算时,其运算次序是非 \rightarrow 与 \rightarrow 或,其中与运算符号“ \cdot ”可省略;若有括号则先进行括号内运算;若括号与非号下的变量一致,则括号可以省略。例如,

$$F=\overline{(A+B)} \cdot C=\overline{A+B} \cdot C=\overline{A} \cdot \overline{B} \cdot C$$

1.1.2 逻辑函数的基本定理

在介绍逻辑函数的基本定理之前,有必要先介绍逻辑函数相等的概念。设有两个逻辑函数

$$F_1 = f_1(A_1, A_2, \dots, A_n)$$

$$F_2 = f_2(A_1, A_2, \dots, A_n)$$

如果对于 $A_1 \sim A_n$ 的任何一组取值使 F_1 和 F_2 具有相同的值,则称这两个逻辑函数相等,即 $F_1 = F_2$ 。例如:

$F_1 = A + B$, $F_2 = A + \overline{A}B$ 。为了判断 F_1 与 F_2 是否相等,可

先列出它们的真值表,如表 1.1.4 所示。由表可见:对

应于变量的任何一组取值, F_1 与 F_2 均有相同的值,所以

有 $F_1 = F_2$ 。既然 $F_1 = F_2$,故可把 F_1 和 F_2 认为是同一个逻辑函数的不同表达式。相等的逻辑函数一定有相同的真值表,反之亦然。

表 1.1.4 相等函数真值表

A	B	F_1	F_2
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

1. 关于变量与常量关系的定理

$$\left. \begin{aligned} A \cdot 0 &= 0 \\ A + 0 &= A \end{aligned} \right\} \quad (1.1.6)$$

$$\left. \begin{aligned} A \cdot 1 &= A \\ A + 1 &= 1 \end{aligned} \right\} \quad (1.1.7)$$

2. 交换律、结合律、分配律

$$\left. \begin{aligned} \text{交换律} \quad A \cdot B &= B \cdot A \\ A + B &= B + A \end{aligned} \right\} \quad (1.1.8)$$

$$\left. \begin{aligned} \text{结合律} \quad A \cdot (B \cdot C) &= (A \cdot B) \cdot C \\ A + (B + C) &= (A + B) + C \end{aligned} \right\} \quad (1.1.9)$$

$$\left. \begin{aligned} \text{分配律} \quad A \cdot (B + C) &= AB + AC \\ A + BC &= (A + B)(A + C) \end{aligned} \right\} \quad (1.1.10)$$

3. 逻辑函数独有的基本定理

交换律、结合律、分配律同普通数学中的公式类似,下面几条是逻辑函数独有的基本定理。

$$\left. \begin{aligned} \text{互补律} \quad A \cdot \overline{A} &= 0 \\ A + \overline{A} &= 1 \end{aligned} \right\} \quad (1.1.11)$$

$$\left. \begin{aligned} \text{重叠律} \quad A \cdot A &= A \\ A + A &= A \end{aligned} \right\} \quad (1.1.12)$$

$$\overline{\overline{A}} = A \quad (1.1.13)$$

$$\left. \begin{array}{l} \text{吸收律} \\ A + AB = A \\ A \cdot (A + B) = A \end{array} \right\} \quad (1.1.14)$$

$$\left. \begin{array}{l} \text{摩根定律} \\ \overline{A \cdot B} = \overline{A} + \overline{B} \\ \overline{A + B} = \overline{A} \cdot \overline{B} \end{array} \right\} \quad (1.1.15)$$

以上定理的正确性可以借用真值表来验证,读者自己证明。

1.1.3 逻辑函数的基本运算规则

1. 代入规则

对于任何一个含有变量 A 的等式,如果将所有出现 A 的地方都用函数 G 替换,则等式仍然成立。

因为任何一个逻辑函数都同逻辑变量一样,只有两种可能取值,所以代入规则是正确的。该规则的运用可以扩展定理的应用范围。

例 1.1.1 已知 $A+BC=(A+B)(A+C)$, 设 $G=C+D$ 。试证明将所有出现变量 B 的位置代之以 G , 则等式仍然成立。

证明 因为

$$A+BC=A+(C+D)C=A+C$$

$$(A+B)(A+C)=(A+C+D)(A+C)=A+C$$

所以

$$A+BC=(A+B)(A+C)$$

2. 反演规则

若已知逻辑函数 F , 则只要将 F 表达式中的所有“ \cdot ”换为“ $+$ ”, “ $+$ ”换为“ \cdot ”, 常量 0 换为 1 , 常量 1 换为 0 , 所有原变量换为反变量, 反变量换为原变量, 即得其反函数 \bar{F} 。反演规则实际上是摩根定理的推广应用, 它的意义在于运用反演规则可以方便地求出反函数 \bar{F} (补函数)。

例 1.1.2 已知 $F=\overline{AB} + CD$, 试求 \bar{F} 。

解 由反演规则有

$$\bar{F} = \overline{\overline{AB} + CD} = \overline{\overline{AB}} \cdot (\overline{C+D})$$

应用反演规则时应注意两点: 其一, 对跨越两个或两个以上变量的非号应保持不变; 其二, 不得改变原式的运算顺序。

依据逻辑函数的二值性质, 如果两个逻辑函数相等, 则它们的反函数必然相等, 反之亦然。

3. 对偶规则

在介绍对偶规则之前有必要先定义对偶式。设 F 为一个逻辑函数式, 如果将 F 中所有的“ $+$ ”换为“ \cdot ”, “ \cdot ”换为“ $+$ ”, 1 换为 0 , 0 换为 1 , 而变量保持不变, 则所得新的逻辑式就称为 F 的对偶式, 记为 F' (或 F^*)。例如,

$$\begin{array}{ll} F=A(B+\bar{C}) & F'=A+B\cdot\bar{C} \\ F=(A+\bar{B})(A+C\cdot 1) & F'=A\cdot\bar{B}+A\cdot(C+0) \\ \overline{\overline{F}}=A+BC & \overline{\overline{F'}}=\overline{A\cdot B\cdot\bar{C}} \end{array}$$

如果两个逻辑函数 F 和 G 相等, 则它们的对偶式 F' 与 G' 也必然相等。反之亦然, 这就是对偶规则。逻辑函数基本定理的两组表达式就是互为对偶式, 也证明了该规则的正确性。

除以上 3 条基本运算规则之外, 还有一条在简化函数中运用十分广泛的添加项规则: 如果两个乘积项中分别包含有一个变量及其反变量, 则用它们的系数(剩余变量)组成一个新乘积项, 该乘积项加入原函数后不会改变函数之值, 反之亦然。下面举例证明。

例 1.1.3 求证 $AB+\bar{A}C=AB+\bar{A}C+BC$ 。

证明 $AB+\bar{A}C+BC=AB+\bar{A}C+(A+\bar{A})BC$
 $=AB+\bar{A}C+ABC+\bar{A}BC=AB+\bar{A}C$

例 1.1.4 求证 $AB+\bar{A}C+BCDE=AB+\bar{A}C$ 。

证明 $AB+\bar{A}C+BCDE=AB+\bar{A}C+BC+BCDE$
 $=AB+\bar{A}C+BC=AB+\bar{A}C$

1.2 逻辑函数的标准型

一个逻辑函数可以有多种等效的表达式, 但其标准形式是唯一的。逻辑函数有两种标准型, 即标准与或式及标准或与式。

1.2.1 逻辑函数的两种标准形式

1. 最小项及标准与或式

设有 n 个变量为 $A_1\sim A_n$, P 是由这 n 个变量组成的与项。若与项 P 中的每一个变量都以 A_i 或 \bar{A}_i 的形式出现一次且仅一次, 则称 P 是最小项。例如, A_1 、 A_2 、 A_3 3 个变量构成的最小项有 $\bar{A}_1\bar{A}_2\bar{A}_3$, $\bar{A}_1\bar{A}_2A_3$, $\bar{A}_1A_2\bar{A}_3$, $A_1\bar{A}_2\bar{A}_3$, $A_1\bar{A}_2A_3$, $A_1A_2\bar{A}_3$, $A_1A_2A_3$ 等 8 个。

n 个变量构成的最小项共有 2^n 个, 为了叙述和书写方便, 通常用 m_i 表示最小项。下角标 i 按下面规则确定: 将变量 $A_1\sim A_n$ 按顺序排列, 与项中文字以原变量出现时记为 1, 以反变量形式出现记为 0。它们按序排列成一个二进制数, 其相应十进制数即为 i 之值。

例如, 三变量的最小项表示为 $\bar{A}_1\bar{A}_2\bar{A}_3 = m_0, \dots, A_1A_2A_3 = m_7$; 四变量的最小项表示为 $\bar{A}_1\bar{A}_2\bar{A}_3\bar{A}_4 = m_0, \dots, A_1A_2A_3A_4 = m_{15}$ 。为了区分 m_i 是多少变量的最小项, 通常在 m_i 上增加一个上角标来注明变量数。例如, m_0^3 和 m_0^4 分别表示最小项为 $\bar{A}_1\bar{A}_2\bar{A}_3$ 和 $\bar{A}_1\bar{A}_2\bar{A}_3\bar{A}_4$ 。在变量数已明确的条件下, 上角标可省略。

由最小项的逻辑和所构成的逻辑函数式称为逻辑函数的标准与或式, 又称为最小项之和标准型。例如, 三变量的逻辑函数

$$F = \bar{A}\bar{B}\bar{C} + ABC$$

显然, F 便是标准与或式。

2. 最大项及标准或与式

设有 n 个逻辑变量为 $A_1 \sim A_n$, 假如 Q 是由这 n 个变量构成的或项(和项), 若 Q 中的每一个变量都以 A_i 或 \bar{A}_i 的形式出现一次且仅一次, 则称 Q 是 n 变量的最大项。

n 个变量的最大项也有 2^n 个。例如, A_1, A_2, A_3 三变量构成的最大项有 $A_1+A_2+A_3, A_1+A_2+\bar{A}_3, \dots, \bar{A}_1+\bar{A}_2+\bar{A}_3$ 等 8 个。

最大项通常用 M_i 表示, 但是, 下角标 i 的确定规则与最小项不同。它是把最大项中以原变量出现记为 0, 以反变量出现记为 1, 它们按序排列成一个二进制数, 其相应十进制数即为 i 之值。

例如, $A_1+A_2+A_3=M_0, \dots, \bar{A}_1+\bar{A}_2+\bar{A}_3=M_7$ 。

由最大项的逻辑乘所构成的逻辑函数式称为逻辑函数的标准或与式, 又称为最大项之积标准型。例如, 三变量的逻辑函数

$$F = (A + \bar{B} + C)(\bar{A} + B + C)$$

显然, F 是逻辑函数的标准或与式。

3. 最小项与最大项的性质

(1) 最小项性质

性质 1 对于任意一个最小项, 仅有一组变量取值, 使它的值为 1, 而变量的其他组取值, 皆使它的值为 0; 最小项不同, 使其值为 1 的变量取值也不同。

例如, 两个变量 A_1, A_2 组成的最小项为 $\bar{A}_1\bar{A}_2, \bar{A}_1A_2, A_1\bar{A}_2, A_1A_2$ 等 4 种:

使 $\bar{A}_1\bar{A}_2 = 1$ 时变量的取值为 00; 使 $\bar{A}_1A_2 = 1$ 时变量的取值为 01;

使 $A_1\bar{A}_2 = 1$ 时变量的取值为 10; 使 $A_1A_2 = 1$ 时变量的取值为 11。

性质 2 n 个变量组成的全体最小项之逻辑和为 1, 即有

$$\sum_{i=0}^{2^n-1} m_i = 1$$

式中, \sum 为连或符号。因为对于 n 个变量的任意一组取值总对应着一个最小项为 1, 故全体最小项之逻辑和恒为 1。

(2) 最大项性质

性质 1 对于任意一个最大项, 变量只有一组取值使它的值为 0, 而变量的其

他组取值使它的值皆为 1；并且最大项不同，使其为 0 的变量取值也不同。

仍以两变量为例说明。 A_1A_2 两变量共有 4 个最大项：

使 $A_1+A_2=0$ 时变量的取值为 00；使 $A_1+\bar{A}_2=0$ 时变量的取值为 01；

使 $\bar{A}_1+A_2=0$ 时变量的取值为 10；使 $\bar{A}_1+\bar{A}_2=0$ 时变量的取值为 11。

性质 2 n 个变量全体最大项之逻辑乘恒为 0，即有

$$\prod_{i=0}^{2^n-1} M_i = 0$$

式中， \prod 为连与符号。因为对于 n 个变量的任一组取值总对应着一个最大项为 0，故全体最大项之逻辑乘恒为 0。

(3) 最小项与最大项的关系

设逻辑变量为 A, B, C ，则有 $m_5 = A\bar{B}C$ 。利用摩根定理对 m_5 求补，即有

$$\overline{m_5} = \overline{A\bar{B}C} = \bar{A} + B + \bar{C} = M_5$$

同理可得

$$\overline{M_5} = \overline{\bar{A} + B + \bar{C}} = A\bar{B}C = m_5$$

由此可见， m_5 与 M_5 是互补的，这个性质对于 n 个变量中任意一对最小项 m_i 及最大项 M_i 都是适用的，即有

$$\overline{m_i} = M_i \quad \text{和} \quad \overline{M_i} = m_i$$

1.2.2 将逻辑函数变换为标准型

有时需要将任意逻辑函数变换为标准型，实现这种变换的方法很多，如利用逻辑运算或者借助真值表和卡诺图等工具来完成这种变换。下面介绍两种常用的变换方法。

1. 利用真值表将函数变换为标准型

利用真值表能方便地将逻辑函数变换为两种标准型。下面结合例子来讨论这种变换方法。

例 1.2.1 将逻辑函数 $F = A\bar{C} + \bar{B}C + \bar{A}BC$ 变换为两种标准型。

解 首先作出函数的真值表，如表 1.2.1 所示。由表可得

$$F(A,B,C) = \bar{A}\bar{B}\bar{C} + \bar{A}BC + A\bar{B}C + ABC = m_0 + m_3 + m_4 + m_6$$

或简写成

$$F(A,B,C) = \sum m(0,3,4,6) \quad (1.2.1)$$

又由真值表可写出

表 1.2.1 例 1.2.1 真值表

A	B	C	F	\bar{F}
0	0	0	1	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	1	0
1	0	1	0	1
1	1	0	1	0
1	1	1	0	1

$$\bar{F}(A,B,C)=\bar{A}\bar{B}C+\bar{A}B\bar{C}+A\bar{B}C+ABC=m_1+m_2+m_5+m_7=\sum m(1,2,5,7) \quad (1.2.2)$$

所以 $F(A,B,C)=\bar{\bar{F}}(A,B,C)=\overline{m_1+m_2+m_5+m_7}=\bar{m}_1 \cdot \bar{m}_2 \cdot \bar{m}_5 \cdot \bar{m}_7=M_1 \cdot M_2 \cdot M_5 \cdot M_7$

或简写成
$$F(A,B,C)=\prod M(1,2,5,7) \quad (1.2.3)$$

由本例可总结出利用真值表求函数两种标准型的方法如下。

标准与或式 由真值表确定 F 为 1 的项作为函数的最小项。若输入变量为 1，则取其变量本身；若输入变量为 0，则取其反变量。它们构成最小项，然后取这些最小项之和。

标准或与式 由真值表确定 F 为 0 的项作为函数的最大项。若输入变量为 1，则取其反变量；若输入变量为 0，则取其变量本身。它们构成最大项，然后取这些最大项之积。

2. 利用公式 $A+\bar{A}=1$ 及 $A \cdot \bar{A}=0$ 将函数变换为标准型

包含 n 个变量的任何一个逻辑函数，都可以通过公式 $A+\bar{A}=1$ ($A \cdot \bar{A}=0$) 将非标准与或式(或与式)中的每一个与项(或项)所缺的变量逐步补齐，变换为标准与或式(或与式)。

例 1.2.2 将 $F=\bar{A}\bar{B}(A+C)$ 变换成两种标准型。

解 (1) 标准与或式

$$\begin{aligned} F &= \bar{A}\bar{B}(A+C) = (\bar{A}+B)(A+C) = \bar{A}C + AB + BC = \bar{A}C + AB \\ &= \bar{A}(B+\bar{B})C + AB(C+\bar{C}) = \bar{A}BC + \bar{A}\bar{B}C + ABC + AB\bar{C} \\ &= m_1 + m_3 + m_6 + m_7 = \sum m(1,3,6,7) \end{aligned}$$

(2) 标准或与式

$$\begin{aligned} F &= (\bar{A}+B)(A+C) = (\bar{A}+B+C \cdot \bar{C})(A+B \cdot \bar{B}+C) \\ &= (\bar{A}+B+C)(\bar{A}+B+\bar{C})(A+B+C)(A+\bar{B}+C) \\ &= M_0 \cdot M_2 \cdot M_4 \cdot M_5 = \prod M(0,2,4,5) \end{aligned}$$

3. 两种逻辑函数标准型之间的关系

如果已知逻辑函数的标准与或式，则由式 (1.2.1)~(1.2.3) 很容易总结出将它改写成标准或与式的规则以及写出它的反函数的标准或与式。

(a) 在将一个 n 变量的标准与或式改写成标准或与式时，其最大项的编号（即 i 之值）必定都不是最小项的编号，而且最小项与最大项的总个数为 2^n 。

例如，对于 $F=\sum m^4(0,1,3,4,7,9)$ ，则有 $F=\prod M^4(2,5,6,8,10 \sim 15)$ 。

(b) 由 m 个最小项构成 n 变量函数的标准与或式，其反函数可以用 m 个最大项构成的标准或与式表示，这 m 个最大项的编号同 m 个最小项的编号全部一致。

例如，对于 $F=\sum m^4(0,1,3,4,7,9)$ ，则有 $\bar{F}=\prod M^4(0,1,3,4,7,9)$ 。

1.3 几种常用的复合逻辑及其逻辑门

逻辑函数的基本运算只有与逻辑、或逻辑、非逻辑等3种运算。但是，有时仅用它们进行运算不太方便，因此，要由这3种基本运算导出几种常用的复合逻辑运算：与非逻辑、或非逻辑、与或非逻辑、或与非逻辑、异或逻辑、同或逻辑、禁止逻辑等运算。实现逻辑运算的集成电路称为集成逻辑门，简称逻辑门。

1.3.1 3种基本逻辑门

1. 与门

完成与运算的电路称为与逻辑门，简称为与门。它有两个或两个以上的输入端和一个输出端。图1.3.1为逻辑符号，逻辑式为

$$F=A \cdot B \cdot C$$

真值表如表1.3.1所示。

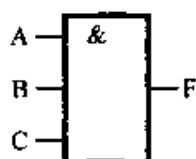


图 1.3.1 与门逻辑符号

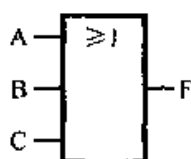


图 1.3.2 或门逻辑符号

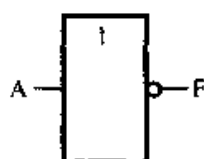


图 1.3.3 非门逻辑符号

2. 或门

完成或运算的电路称为或逻辑门，简称或门。它有两个或两个以上的输入端和一个输出端。图1.3.2为逻辑符号，逻辑式为

$$F=A+B+C$$

真值表如表1.3.2所示。

3. 非门

完成非运算的电路称为非逻辑门，又称为反相器。图1.3.3为逻辑符号，图中，输出端的小圆圈表示非。逻辑式为

$$F=\bar{A}$$

表 1.3.1 与门真值表

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

表 1.3.2 或门真值表

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

1.3.2 常用的复合逻辑及其逻辑门

1. 与非逻辑

用与及非逻辑合成与非逻辑。实现这种逻辑关系的电路称为与非门。其逻辑符号和真值表分别示于图 1.3.4 和表 1.3.3, 逻辑式为 $F = \overline{ABC}$ 。

由真值表可见: 只有当输入全为 1 时, 输出才为 0, 否则输出就为 1。

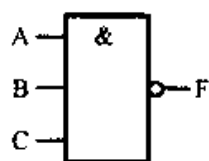


图 1.3.4 与非门逻辑符号

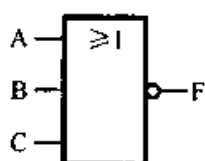


图 1.3.5 或非门逻辑符号

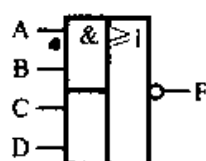


图 1.3.6 与或非门逻辑符号

2. 或非逻辑

用或及非逻辑合成或非逻辑。实现这种逻辑关系的电路称为或非门，逻辑符号和真值表分别示于图 1.3.5 和表 1.3.4, 逻辑式为 $F = \overline{A+B+C}$ 。

由真值表可见: 只有当输入全为 0 时, 输出才为 1; 否则输出便为 0。

3. 与或非逻辑

用与、或及非 3 种运算合成与或非逻辑运算。实现这种逻辑关系的电路称为与或非门。逻辑符号如图 1.3.6 所示, 逻辑式为 $F = \overline{AB+CD}$ 。

4. 异或逻辑及同或逻辑

异或逻辑及同或逻辑都是两变量的逻辑函数。异或逻辑的表达式为

$$F = A\bar{B} + \bar{A}B = A \oplus B$$

同或逻辑的表达式为

$$F = AB + \bar{A}\bar{B} = A \odot B$$

式中, \oplus 符号——异或运算; \odot 符号——同或运算。它们的真值表示于表 1.3.5, 逻辑符号分别如图 1.3.7 和图 1.3.8 所示。分别称为异或门和同或门。由真值表可见: 对于异或逻辑而言, 只有当输入相异时输出才为 1; 而对于同或逻辑而言, 只有当输入相同时输出才为 1。这就是异或(同或)逻辑的含义。

表 1.3.3 与非门真值表

A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

表 1.3.4 或非门真值表

A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

表 1.3.5 异或、同或逻辑真值表

A	B	$A \oplus B$	$A \odot B$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

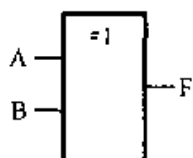


图 1.3.7 异或门逻辑符号

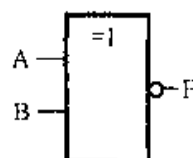


图 1.3.8 同或门逻辑符号

从真值表和逻辑式可以证明异或门和同或门具有如下重要特性。

(a) 异或逻辑及同或逻辑互为反函数：

$$\overline{A \oplus B} = A \odot B \quad \overline{A \odot B} = A \oplus B$$

(b) 异或逻辑及同或逻辑互为对偶式：

设

$$F = A \oplus B = A\bar{B} + \bar{A}B$$

$$G = A \odot B = AB + \bar{A}\bar{B}$$

则

$$F' = (A + \bar{B})(\bar{A} + B) = AB + A\bar{B} = A \odot B = G$$

$$G' = (A + B)(\bar{A} + \bar{B}) = A\bar{B} + \bar{A}B = A \oplus B = F$$

因此，F 与 G 互为对偶式， \oplus 与 \odot 两种运算符互为对偶符号。

(c) 异或运算的性质如下：

交换律

$$A \oplus B = B \oplus A$$

若 $A \oplus B = C$ ，则 $A \oplus C = B$ ；若 $B \oplus C = A$ ，则 $A \oplus B \oplus C = 0$ 。

结合律

$$A \oplus (B \oplus C) = (A \oplus B) \oplus C$$

分配律

$$A(B \oplus C) = AB \oplus AC$$

证明

$$A(B \oplus C) = A(B\bar{C} + \bar{B}C) = AB\bar{C} + A\bar{B}C$$

$$= AB(\bar{C} + \bar{A}) + AC(\bar{B} + \bar{A})$$

$$= AB\bar{A}\bar{C} + \bar{A}\bar{B}AC = AB \oplus AC$$

$$A \oplus \bar{A} = 1$$

$$A \oplus A = 0$$

当 n 个变量 A 异或时，若 n 为偶数，则有

$$A \oplus \cdots \oplus A = 0$$

推论 当 n 个变量作异或运算时，若有偶数个变量取值为 1，则函数值为 0；若有奇数个变量取值为 1，则函数值为 1。

*1.3.3 正逻辑与负逻辑

在逻辑函数中，输入变量和输出函数都具有二值性，即只有逻辑 1 和逻辑 0 两种状态。若是用电路来实现逻辑关系，则可用高电平表示逻辑 1，以低电平表示逻辑 0。

辑 0，在这种规定之下的逻辑关系称为正逻辑。

事实上，还有另一种规定：用低电平表示逻辑 1，用高电平表示逻辑 0。在这种规定之下的逻辑关系称为负逻辑。对于同一个逻辑电路，由于采用的逻辑制度不同，故它所实现的逻辑关系也就不同。例如，图 1.3.9 所示的是由二极管构成的门电路，按照正逻辑规定它是与门；而按照负逻辑规定它是或门。由此可见：正、负逻辑的逻辑式互为对偶关系。若给出一个电路的正逻辑的逻辑式，则该逻辑式的对偶式便是这个电路的负逻辑的逻辑式，反之亦然。

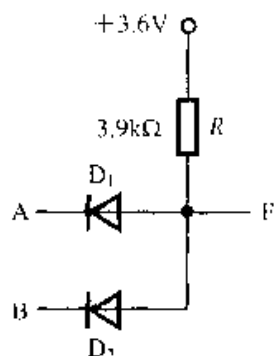


图 1.3.9 二极管门电路

在本书中，除特殊情况注明负逻辑外，一律采用正逻辑。

1.4 逻辑函数的简化

一个逻辑函数可以有多种不同的逻辑表达式，它们在繁简程度上有所差异，但是，它们所表示的逻辑功能是完全等效的。将较繁的逻辑表达式变换成与之等效的最简逻辑表达式就称为逻辑函数的简化。

为什么要对逻辑函数进行简化呢？因为逻辑函数与逻辑电路之间存在着对应关系。逻辑函数简化意味着可以用较少的逻辑元件及较少的输入端来实现同样的逻辑功能。这对于提高电路可靠性和降低成本都是有利的。

逻辑函数简化的方法较多，本节主要介绍逻辑代数简化法和卡诺图简化法。

1.4.1 逻辑代数简化法

逻辑代数简化法是运用逻辑代数的一些定理、公式和运算规则对逻辑函数进行等式变换来求得最简结果。

1. 与或式的简化

(1) 与或式

逻辑变量的与运算称为与项(乘积项)，与项的逻辑加(和)即为与或式。

例如， ABC ， $\overline{A}C$ 均为与项，所以 $ABC + \overline{A}C$ 为与或式。最小项也是与项中的一种，故标准与或式是与或式中的一种。在所有等效的与或式中，若某个与或式所含与项最少，且每个与项所含变量数量少，则这个与或式便称为最简与或式。

(2) 简化方法

合并项法 利用公式 $AB + \overline{A}B = B$ 消去一个变量。

例 1.4.1 简化 $F = A(BC + \overline{B}\overline{C}) + A(\overline{B}C + B\overline{C})$ 。

解 $F = A[(BC + \overline{B}\overline{C}) + (\overline{B}C + B\overline{C})] = A[(\overline{B} \oplus \overline{C}) + (B \oplus C)] = A$

吸收法 利用公式 $A + AB = A$ 消去一项。

例 1.4.2 简化 $F = A\overline{B} + A\overline{B}CD(E + G)$ 。

解 令 $W = A\overline{B}$

则 $F = W + WCD(E + G) = W = A\overline{B}$

消除法 利用公式 $A + \overline{A}B = A + B$ 消去多余的变量。

例 1.4.3 简化 $F = AB + \overline{A}C + \overline{B}C$ 。

解 $F = AB + \overline{A}C + \overline{B}C = AB + (\overline{A} + \overline{B})C = AB + \overline{A}BC + AB + C$

配项法 先适当增加一些项，然后再进行简化。

例 1.4.4 简化 $F = A\overline{B} + AC + ADE + \overline{C}D$ 。

解
$$\begin{aligned} F &= A\overline{B} + AC + ADE + \overline{C}D = A\overline{B} + AC + ADE(C + \overline{C}) + \overline{C}D \\ &= A\overline{B} + AC + ACDE + AC\overline{D}E + \overline{C}D \\ &= A\overline{B} + AC + \overline{C}D \end{aligned}$$

一般，对于较复杂的逻辑函数式则需综合运用上述方法进行简化。

例 1.4.5 简化 $F = AB + A\overline{C} + \overline{B}C + \overline{B}C + \overline{B}D + \overline{B}D + ADE(G + H)$ 。

解
$$\begin{aligned} F &= A(B + \overline{C}) + \overline{B}C + \overline{B}C + \overline{B}D + \overline{B}D + ADE(G + H) \\ &= A\overline{B}C + \overline{B}C + \overline{B}C + \overline{B}D + \overline{B}D + ADE(G + H) \\ &= A + \overline{B}C + \overline{B}C + \overline{B}D + \overline{B}D = A + \overline{B}C + \overline{B}C + \overline{B}D + \overline{B}D + \overline{C}D \\ &= A + \overline{B}C + \overline{B}D + \overline{C}D \end{aligned}$$

2. 或与式的简化

(1) 或与式

逻辑变量的或运算称为或项(和项)，或项的逻辑乘即构成或与式。若等效的或与式中，某个或与式所含或项最少，且每个或项中所含变量数量最少，则称它为最简或与式。

(2) 简化方法

直接简化法 直接利用逻辑代数的公式进行简化。

例 1.4.6 简化 $F = A(A + B)(\overline{A} + D)(\overline{B} + \overline{C})(A + C + E + H)$ 。

解
$$\begin{aligned} F &= A(A + B)(\overline{A} + D)(\overline{B} + \overline{C})(A + C + E + H) \\ &= A(\overline{A} + D)(\overline{B} + \overline{C}) = AD(\overline{B} + \overline{C}) \end{aligned}$$

两次对偶简化法 利用对偶规则将或与式转换为它的对偶与或式，对它进行简化并得到最简与或式；然后再求最简与或式的对偶式，即得到原函数的最简或与式。

用两次对偶简化法来求解例 1.4.6，先求 F 的对偶式并进行简化：

$$\begin{aligned} F' &= A + AB + \bar{A}D + \bar{B}\bar{C} + ACEH \\ &= A + \bar{A}D + \bar{B}\bar{C} = A + D + \bar{B}\bar{C} \end{aligned}$$

再求 F' 的对偶式:

$$F=(F')' = AD(\bar{B} + \bar{C})$$

运用代数简化法时,要求熟练掌握逻辑函数的公式和运算规则,而且还要有一定的简化技巧;对于较复杂的逻辑式往往难于判断简化结果是否就是最简结果。卡诺图简化法能克服这些缺点,直观地得出最简结果。

1.4.2 卡诺图简化法

任意一个逻辑函数都可以用真值表正确地、唯一地表达出来,真值表使逻辑函数成为描述逻辑功能的重要工具。但是,真值表直接作为运算工具就显得不方便。因此,有人便想将真值表改换一种画法,使其保留真值表的特性,又便于作逻辑运算。首先提出这种作图方法的是卡诺(Karnaugh)和范奇(Veitch),故称为卡诺图,或称为真值图。

1. 卡诺图

卡诺图是真值表的一种图形表示方法。把真值表中的变量分成两组分别排在行和列中,就构成二维图表。

图 1.4.1 是真值表 1.4.1 的卡诺图。变量 A 的取值 0、1 排在左列;变量 B 的 0、1 取值排在最上行。图中与变量 A 、 B 取值所对应的小方格表示一个最小项,相应最小项可以用变量的组合来表示,或者用 m_i 来表示,也可简化成下用角标 i 表示。如图 1.4.1(a)和(b)所示,我们称它为卡诺框。在每个小方格中填入相应函数值,便将真值表转换为卡诺图了,如图 1.4.1(c)所示。

用类似的方法可以画出三变量和四变量的卡诺框,如图 1.4.2(a)和(b)所示。假如给出它们的真值表,将表中每个最小项的函数值填入相应的小方格,便构成三变量或四变量的卡诺图。

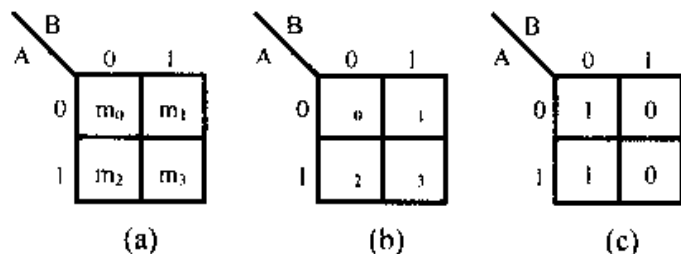


图 1.4.1 卡诺图标记法

(a) 二变量卡诺框之一 (b) 二变量卡诺框之二

(c) 表 1.4.1 的卡诺图

表 1.4.1 图 1.4.1
真值表

A	B	F
0	0	1
0	1	0
1	0	1
1	1	0

为了便于简化函数, 变量取值的排列很重要, 其规则是变量取值按循环码^①次序排列, 即要求相邻小方格的最小项只有一个变量不同。例如, 四变量卡诺图中, AB 及 CD 的排列次序应为 00, 01, 11, 10; 而不能排列成 00, 01, 10, 11。

这里有必要说明一下相邻的概念。若两个最小项中只有一个变量取值不同, 则称它们为相邻最小项。例如, ABC 与 $AB\bar{C}$ 便是相邻最小项。在卡诺图中与这两个相邻最小项对应的小方格称为相邻小方格。请读者注意: 相邻小方格不仅是指几何位置上相邻的小方格, 而且与纵、横轴对称的小方格也是相邻小方格。如图 1.4.2(b) 中 m_0 与 m_2, m_2 与 m_{10} 等都是相邻小方格。任一小方格的相邻小方格的数目都等于函数的变量数, 如图 1.4.2(b) 所示四变量卡诺图中, m_0 的相邻小方格是 m_1, m_2, m_4, m_8 。

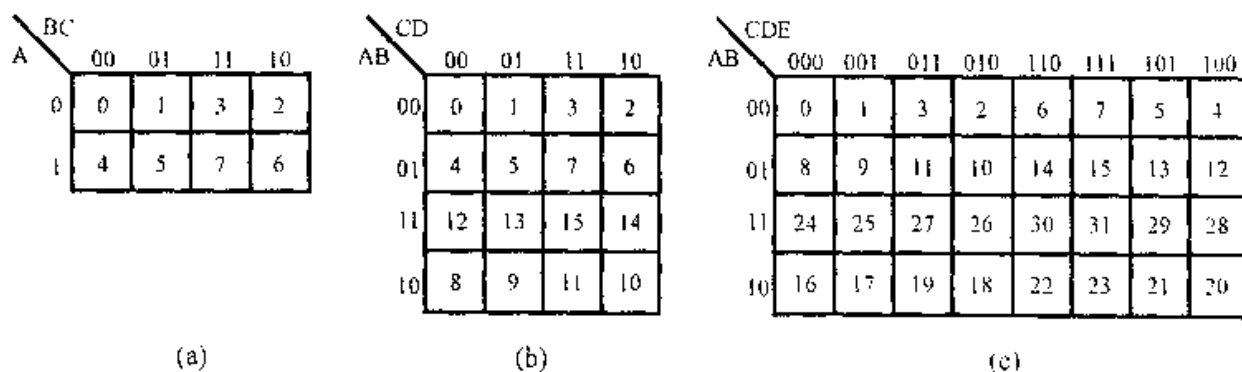


图 1.4.2 多变量卡诺图

(a) 三变量卡诺图 (b) 四变量卡诺图 (c) 五变量卡诺图

变量数超过 4 个以上的卡诺图, 其变量取值的排列可以采用折叠法, 将 n 个变量的卡诺图扩展为 $n+1$ 个变量的卡诺图。例如, 将图 1.4.2(b) 所示四变量卡诺图向右折叠即成五变量卡诺图。图中以折线为轴, 轴线左边变量 C 取值为 0, 右边变量 C 取值为 1; 右边 DE 的取值是左边 DE 取值的镜像, 如图 1.4.2(c) 所示。用类似的方法可将五变量卡诺图折叠成六变量卡诺图。

2. 已知逻辑函数, 如何作卡诺图

对于一个给定的逻辑函数, 一般有 3 种方法可作出它的卡诺图, 即真值表法、标准型法和观察法。

(1) 真值表法

先作出已知函数的真值表, 然后将表中每行函数值填入卡诺框中相应的小方格。

(2) 标准型法

将函数变换为标准与或式, 然后在与函数所含有的最小项相应的小方格填 1, 其余小方格填 0。在实际应用中, 为了简化作图, 通常只填写一种逻辑值。例如,

① 循环码的特性参阅 3.3.1。

填写逻辑 1(或逻辑 0), 而另一种逻辑值 0(或逻辑 1)可以不填写, 如图 1.4.3 所示。

例 1.4.7 作出 $F = AB\bar{C}D + B\bar{C}\bar{D} + BD$ 的卡诺图。

解 将 F 展开为最小项之和标准型

$$\begin{aligned} F &= AB\bar{C}D + B\bar{C}\bar{D} + BD \\ &= AB\bar{C}D + AB\bar{C}\bar{D} + \bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}BCD + AB\bar{C}D + ABCD \\ &= \sum m(4, 5, 7, 12, 13, 15) \end{aligned}$$

将式中最小项在图中对应的小方格填 1, 其他小方格填 0(可不填), 如图 1.4.3 所示。

(3) 观察法

采用观察法就不需要前两种方法那样的严格而繁琐的步骤, 而是审视逻辑函数, 确定每个与项(或项)应该在哪些对应小方格填 1(0), 便可画出卡诺图。例如, 用观察法来画例 1.4.7 的卡诺图。与项 $AB\bar{C}D$ 在 m_{13} 填 1, 与项 $B\bar{C}\bar{D}$ 在 m_4 和 m_{12} 填 1, 与项 BD 在 m_5, m_7, m_{13}, m_{15} 填 1。所得结果与图 1.4.3 一致。

例 1.4.8 作出 $F = ABCD + B$ 的卡诺图。

解 审视逻辑函数可以发现: 当 $B=1$ 时, $F=1$; 当 $B=0$ 时, $F=0$ 。因此, 可很迅速地画出如图 1.4.4 所示的卡诺图。

AB \ CD				
	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

图 1.4.3 例 1.4.7 的卡诺图

AB \ CD				
	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

图 1.4.4 例 1.4.8 的卡诺图

3. 利用卡诺图简化逻辑函数

(1) 合并最小项的规则

在卡诺图 1.4.5(a)中, 最小项 m_1 和 m_5 都填有 1, 若求其逻辑和, 则有

$$F = \bar{A}\bar{B}C + A\bar{B}C = \bar{B}C$$

式中, m_1 和 m_5 是相邻最小项。由此可见: 若将相邻的两个 1 格(填 1 的最小项)合并为一个与项, 则可消去一个变量。相邻 1 格合并的方法可以推广到相邻单元。例如, 在图 1.4.5(b)中, m_0 与 m_4 合并为一项, 并用圈①(虚线圈)表示; m_2 与 m_6 合并为一项, 用圈②(虚线圈)表示。圈①相应的与项为 $\bar{B}\bar{C}$, 圈②相应的与项为 $B\bar{C}$ 。所以, 圈①与圈②是相邻单元, 它们又可合并为圈③(实线圈), 相应与项便为 \bar{C} 。这样合并的结果为 $F = \bar{C}$, 显然逻辑函数得以简化。相邻单元的概念可以继续推广运用。如图 1.4.6 中, 可以把 8 个 1 格圈在一起, 其结果为 $F = B$ 。读者可以证明: 如

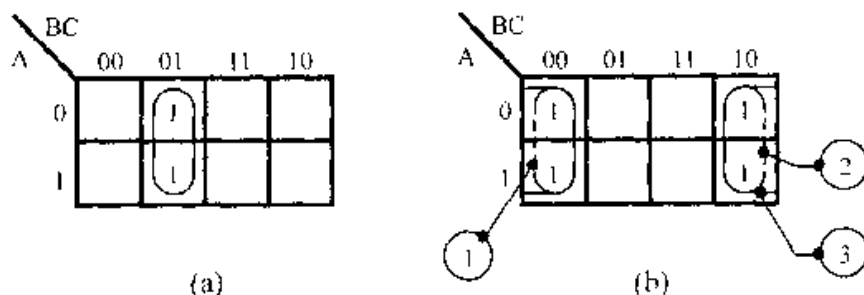


图 1.4.5 相邻项合并举例之一

(a) $F = \bar{B}C$ (b) $F = \bar{C}$

果卡诺图上所有小方格的逻辑值为 1, 则 $F=1$ 。

综上所述, 可以得出以下合并最小项的规则。

规则 1 将逻辑值为 1 的相邻最小项圈起来, 圈内最小项(1 格)的个数 $M=2^i$ ($i=0, 1, 2, \dots$), 即必须是 2 的幂; 2^i 个 1 格圈在一起可以消去 i 个变量, 所以, 合并后的与项中只包含 $(n-i)$ 个变量。由此可见, 相邻 1 格愈多, 即合并圈愈大, 可以消去的变量数愈多, 合并后的与项愈简单。

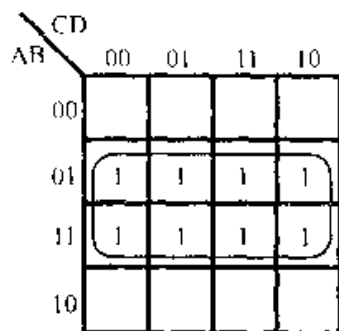


图 1.4.6 相邻项合并举例之二

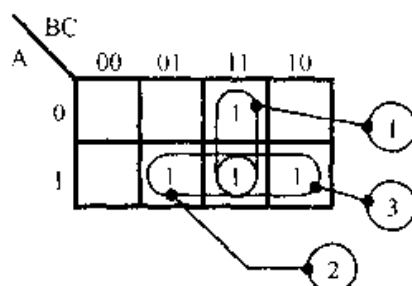


图 1.4.7 合并圈重叠举例

下面分析图 1.4.7 所示的卡诺图, 依据规则 1, m_3 与 m_7 , m_5 与 m_7 , m_6 与 m_7 均为相邻最小项, 它们分别合并为圈①, 圈②和圈③, 简化结果为

$$F = BC + AC + AB$$

显然 3 个合并圈中都包含有 m_7 , 由此得出

规则 2 为了使函数简化得到最佳结果, 合并圈之间允许部分重叠, 某些 1 块甚至可以多次重叠(如 m_7)。

最后来分析图 1.4.8, 依据规则 1 和 2, 可以画出 3 个合并圈。但是, 圈③所包含的最小项 m_3 和 m_7 已分别被圈①和圈②包含了, 即圈③中每个 1 格都不只圈过一次, 因此, 圈③就是多余的, 其相应的与项称为多余项(或冗余项), 在逻辑简化中不允许出现多余项, 否则, 简化结果不是最简的。由此得出

规则 3 若一个合并圈包含的 1 格均被其他合并圈分别包含了, 则这个合并圈就是多余的, 必须消除。

(2) 利用卡诺图将函数简化为或与式

在卡诺图上的1格表示函数F为1, 补函数 \bar{F} 为0; 而0格表示函数F为0, 补函数 \bar{F} 为1。所以, 可以圈0求得补函数 \bar{F} 的与或式。例如, 在图1.4.9中圈0求得

$$\bar{F} = \bar{C}\bar{D} + \bar{A}\bar{B}$$

$$F = (C + D) \cdot (A + B)$$

因此, 在卡诺图上圈0可以直接得到F的或与表达式

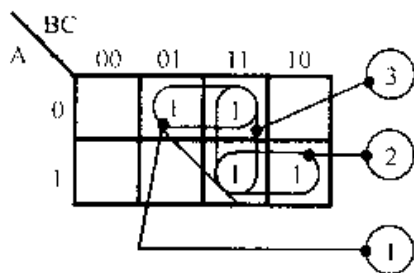


图 1.4.8 多余项举例

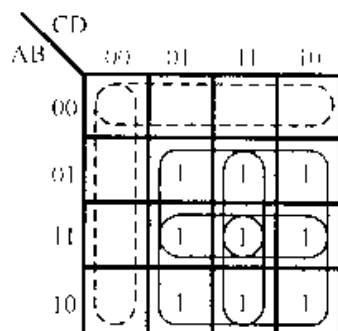


图 1.4.9 例 1.4.9 卡诺图化简

现在可以总结出卡诺图简化逻辑函数的基本规律:

(a) 在卡诺图上圈1得到F的与或式。每一个合并圈就是一个与项。它的构成方法是变量取值为0则用反变量, 变量取值为1则用原变量, 所有与项求和便得到与或式。

(b) 在卡诺图上圈0得到F的或与式。每一个合并圈就是一个或项。它的构成方法是变量取值为0则用原变量, 变量取值为1则用反变量, 所有或项相与便得到或与式。

(c) 为了获得最佳的简化结果, 在覆盖全部1(或0)的条件下, 合并圈最少; 而每个合并圈尽可能大, 为了构成更大的合并圈, 圈与圈之间允许部分重叠。

(d) 在卡诺图上是采用圈1格或是圈0格, 主要应依据图上0、1分布情况, 以获得最简单的逻辑函数表达式(与或式或者或与式)为原则。

(3) 卡诺图的应用举例

在掌握了合并最小项的规则和运用卡诺图的基本规律之后, 就不难对逻辑函数进行卡诺图简化。在简化过程中最关键的是寻找相邻1格(0格)和相邻单元; 此外, 还要避免出现多余项, 否则, 就得不到最佳简化结果。

例 1.4.9 逻辑函数F的卡诺图示于图1.4.9, 求F的最简与或式及或与式。

解 圈1格得到最简与或式, 如实线圈所示。最简与或式为

$$F = AD + AC + BD + BC$$

圈0格得到最简或与式, 如虚线圈所示。最简或与式为

$$F = (C + D)(A + B)$$

例 1.4.10 逻辑函数 F 的卡诺图示于图 1.4.10, 求 F 的最简与或式。

解 依据合并最小项规则(1), m_0, m_2, m_8 和 m_{10} 为相邻 1 格, 它们可以合并为圈①, 而圈②和圈③是由规则 2 画出的。所以, 最简与或式为

$$F = \bar{A}\bar{B} + \bar{C}D + \bar{B}\bar{D}$$

例 1.4.11 逻辑函数 F 的卡诺图如图 1.4.11 所示, 求 F 的最简与或式及最简或与式。

解 按图中实线圈合并 1 格, 得到最简与或式为

$$F = AD + AB\bar{C} + \bar{A}\bar{B}\bar{D}$$

按图中虚线圈合并 0 格, 得到最简或与式为

$$F = (A + \bar{D})(B + D)(\bar{A} + \bar{C} + D)$$

例 1.4.12 简化逻辑函数 $F = \sum m(1, 3, 6, 7, 10, 11, 13, 15)$ 。

解 首先作四变量卡诺图, 如图 1.4.12 所示; 然后, 圈 1 格得到最简与或式为

$$F = BC + AC + \bar{A}\bar{B}D$$

本例中, 如果把相邻 1 格 m_3, m_7, m_{15}, m_{11} 圈起来(如虚线圈), 则会产生多余项 CD 。因为依据规则 3, 这个圈中的 1 格已被其他圈分别包含了, 所以应予消除。

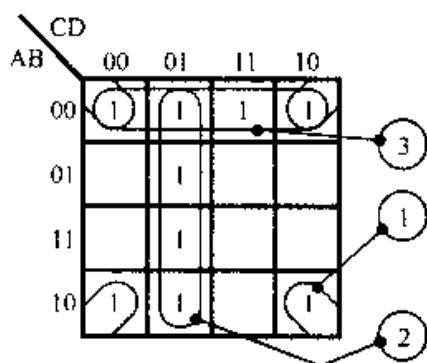


图 1.4.10 例 1.4.10
卡诺图化简

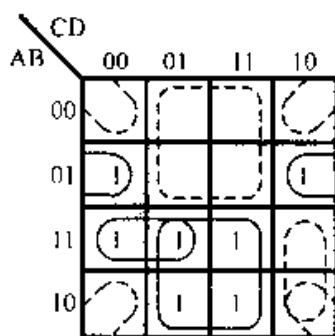


图 1.4.11 例 1.4.11
卡诺图化简

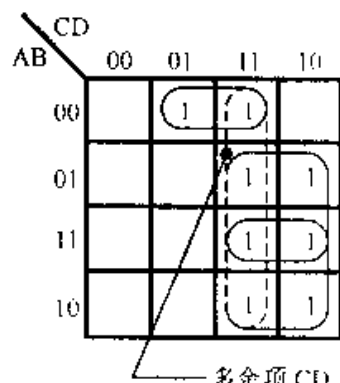


图 1.4.12 例 1.4.12
卡诺图化简

例 1.4.13 简化逻辑函数 $F = \prod M(2, 3, 4, 6, 11, 12, 14)$, 写出最简或与式。

解 在前面讨论卡诺图的构成和卡诺框时, 已指出每一个小方格都对应着一个最小项, 同样, 它也对应着一个最大项。例如, 小方格 9, 对应最小项为 $AB\bar{C}D = M_9$; 如果用最大项表示, 则依据变量为 0 者取原变量, 变量为 1 者取反变量的规则, 可以写出小方格 9 的最大项为 $\bar{A} + B + C + \bar{D} = M_9$ 。所以, 同一个小方格所对应的最大项同最小项的编号是一样的。在填写卡诺图时, 函数所包含的所有最大项应填 0, 因为函数是标准或与式, 所包含的最大项有一个为 0, 则函数为 0。例如, M_2 包含在函数 F 之中, 所以, 在 $ABCD$ 为 0010 时, M_2 填 0, 由此画出 F 的卡诺图如图 1.4.13 所示。依据图中所画合并圈可写出最简或与式为

$$F = (\bar{B} + D)(B + \bar{C} + \bar{D})(A + B + \bar{C})$$

也可以合并 M_2 和 M_6 来构成圈③, 显然 F 的表达式有些差异, 但它们是等效的, 因为两种合并圈都只能消去一个变量。

还有另一种解法: 利用两种标准型的关系可以写出

$$\bar{F} = \sum m(2, 3, 4, 6, 11, 12, 14)$$

这里 \bar{F} 是读者熟悉的标准与或式。先画出 \bar{F} 的卡诺图, 如图 1.4.13(b) 所示。由图中合并圈可得

$$\bar{F} = B\bar{D} + \bar{A}\bar{B}C + \bar{B}CD$$

因此

$$F = (\bar{B} + D)(A + B + \bar{C})(B + \bar{C} + \bar{D})$$

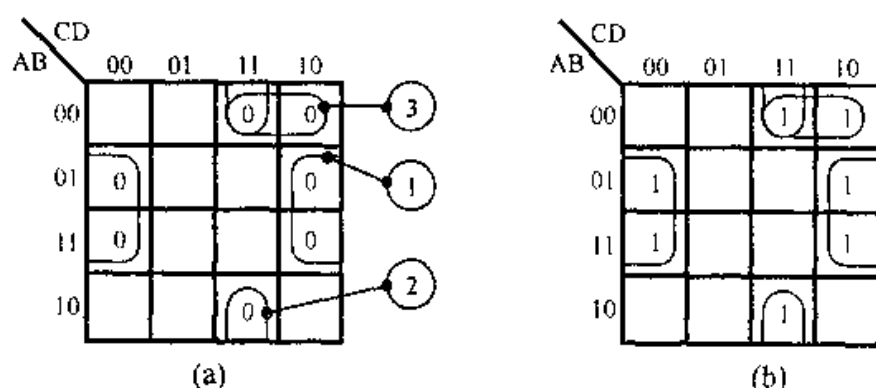


图 1.4.13 例 1.4.13 卡诺图化简

(a) F 卡诺图 (b) \bar{F} 卡诺图

例 1.4.14 逻辑函数 F 的卡诺图如图 1.4.14 所示, 求 F 的最简与或式。

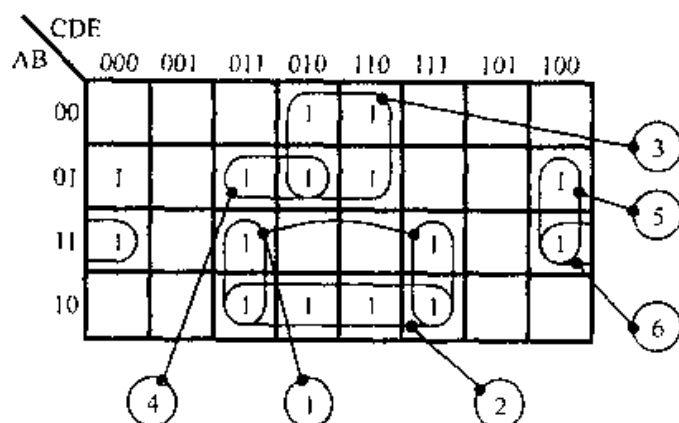


图 1.4.14 例 1.4.14 卡诺图化简

解 这是五变量的卡诺图, 相邻 1 格或相邻单元不易辨认。例如, 用弧线连接的两个圈表示它们为相邻单元, 可以继续合并为圈①, 所以, 在使用五变量以上的卡诺图时, 必须仔细识别相邻最小项和相邻单元(注意: m_{24} 与 m_{28} 是相邻 1 格)。

本例依据图中合并圈得到最简与或式为

$$F = ADE + A\bar{B}D + \bar{A}D\bar{E} + \bar{A}B\bar{C}D + BC\bar{D}\bar{E} + AB\bar{D}\bar{E}$$

(4) 含有无关最小项的逻辑函数简化

在有些实际逻辑事件中,对 n 个变量的逻辑函数并不是 2^n 个最小项都有确定的函数值(0 或 1),而是其中的一部分有确定值,另一部分没有确定值。这种函数称为含有无关最小项(又称为任意项或随意态)的逻辑函数。

出现无关最小项的情况有两类:其一,输入变量的某些组合对输出函数无影响,不必关心与它们相对应的函数值是 0 或是 1;其二,输入变量的某些组合不会在逻辑问题中出现,因而也没有必要关心其相应的函数值是 0 或是 1。例如,用 A、B、C 3 个变量的组合来表示电路不操作或加法、乘法和除法操作,而且电路每次只能执行一种操作。假如选择 A、B、C 的取值为 000(不操作)、100(加法)、010(乘法)、001(除法),而 011, 101, 110 和 111 等组合是不允许出现的,即 A、B、C 的取值是有约束条件的。这种约束关系可以用逻辑式来描述,即

$$\bar{A}BC + A\bar{B}C + AB\bar{C} + ABC = 0$$

或写为

$$\sum \phi(3, 5, 6, 7) = 0$$

显然,把这些恒为 0 值的最小项加入逻辑函数的与或式之中不会改变逻辑函数的最后运算结果。所以,这些无关最小项的函数值可以取值为 0,也可取值为 1。合理地使用它们可以使函数进一步简化。

例 1.4.15 简化逻辑函数

$$F = \sum m(0, 1, 5, 8, 9, 14, 15) + \sum \phi(2, 4, 11, 13)$$

解 先作四变量卡诺图,如图 1.4.15 所示。

图中, m_2 、 m_4 、 m_{11} 、 m_{13} 为无关最小项,故填 ϕ ,

它们的取值可以随意确定。所以,在圈 1 格时,

为了使合并圈更大,可令某些 ϕ 格值为 1,如图①;

同理,在圈 0 格时,可以令某些 ϕ 格值为 0,如虚线圈④、③和②。利用某些 ϕ 格可以随意取值这一特点构成更大合并圈,因而使函数进一步简化。简化结果为

$$F = \bar{A}\bar{C} + \bar{B}\bar{C} + ABC$$

$$F = (A + \bar{C})(B + \bar{C})(\bar{A} + \bar{B} + C)$$

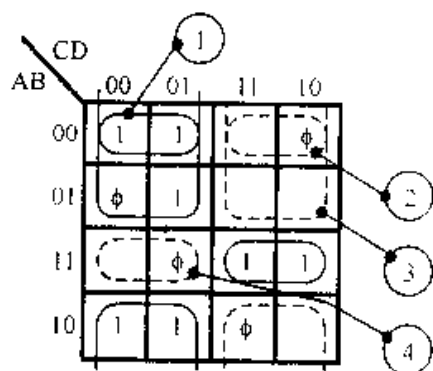


图 1.4.15 例 1.4.15 卡诺图化简

*1.5 卡诺图的其他应用

卡诺图是数字电路中广泛使用的重要工具,它除了进行函数简化之外,在其他许多场合也经常使用。本节将介绍卡诺图其他几个应用的例子。

1. 判明函数关系和进行函数的运算

(1) 判明函数关系

借助卡诺图可以很容易判明两个函数是否相等或者是否互补。设两个函数为 F 和 G ，如果它们的卡诺图完全相同，则 $F=G$ ，否则 $F \neq G$ ；如果 F 卡诺图中的 1 格 (0 格) 与 G 卡诺图中的 0 格 (1 格) 是一一对应关系，则 F 与 G 互补，即 $F = \bar{G}$ 或者 $G = \bar{F}$ 。因此，将 F 卡诺图上填 1 格变为填 0 格；填 0 格变为填 1 格，则得到的卡诺图便是补函数 \bar{F} 的卡诺图。

(2) 函数运算

(a) 或运算。若已知函数 F_1 及 F_2 ，求它们的逻辑和。可以先作出 F_1 及 F_2 的卡诺图，将它们对应小方格中的值求逻辑和，则得到一张新卡诺图，它便是 $F_1 + F_2$ 的卡诺图，如图 1.5.1 所示。

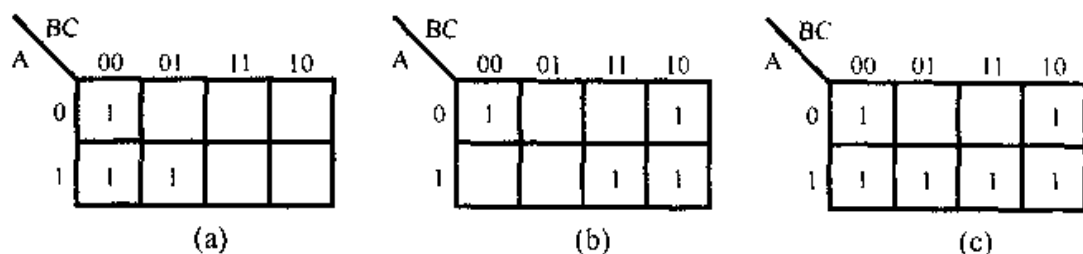


图 1.5.1 卡诺图或运算

(a) F_1 (b) F_2 (c) $F_1 + F_2$

(b) 与运算。若已知 F_1 和 F_2 的卡诺图，则将这两张卡诺图的对应小方格的逻辑值相乘，所得到的新卡诺图便是 $F_1 \cdot F_2$ 的卡诺图，又称为乘积卡诺图，如图 1.5.2 所示。

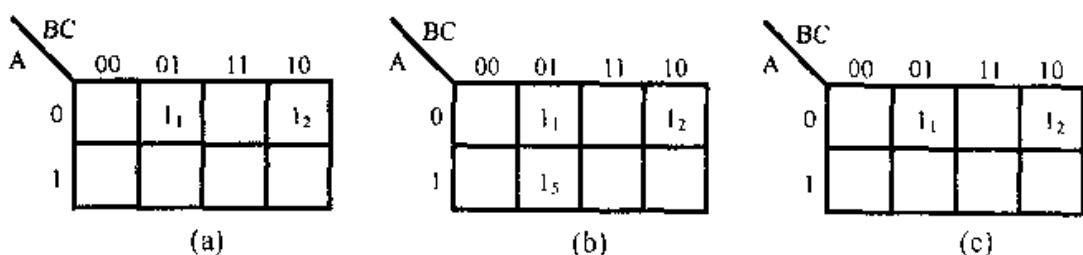


图 1.5.2 乘积卡诺图

(a) F_1 (b) F_2 (c) $F_1 \cdot F_2$

乘积卡诺图的正确性很容易由逻辑运算加以证明，即

$$\begin{aligned}
 F_1 \cdot F_2 &= (m_1 + m_2)(m_1 + m_2 + m_5) = m_1 m_1 + m_1 m_2 + m_1 m_5 + m_2 m_1 + m_2 m_2 + m_2 m_5 \\
 &= m_1 + m_2 = \sum m(1, 2)
 \end{aligned}$$

2. 逻辑表达式类型转换

逻辑表达式有多种类型，如常用的有与或式、与非式、或与式、或非式等。不

同类型的逻辑表达式需用不同类型的门器件来实现。因此,有时候需要对逻辑表达式进行类型转换。这种转换借助于卡诺图就能方便地完成,下面举例说明。

(1) 将与或式转换为或与式

该类型的转换在函数简化中已多次使用,其实就是由卡诺图求出最简与或式及最简或与式。例如,对 $F = \bar{A}\bar{B}D + A\bar{D} + B\bar{D} + A\bar{B}C$, 若要将函数表达式转换为或与式,则可用观察法作出 F 的卡诺图,如图 1.5.3 所示;然后圈 0 块便可得到最简或与式,即

$$F = (\bar{B} + \bar{D})(\bar{A} + C + \bar{D})(A + B + D)$$

(2) 将与或式转换为与或非表达式

设 $F = \bar{A}C + \bar{A}\bar{D} + \bar{B}C + \bar{B}\bar{D}$, 为了得到 F 的与或非表达式,先作出 F 的卡诺图,如图 1.5.4 所示。在卡诺图上圈 0 格可以得到补函数 \bar{F} 的与或式,即

$$\bar{F} = AB + \bar{C}D$$

因此

$$F = \overline{AB + \bar{C}D}$$

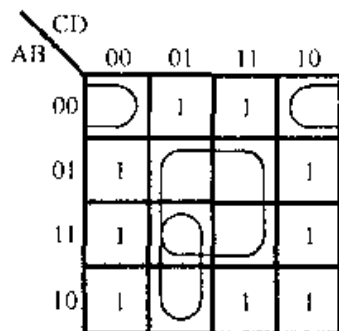


图 1.5.3 简化为或与式

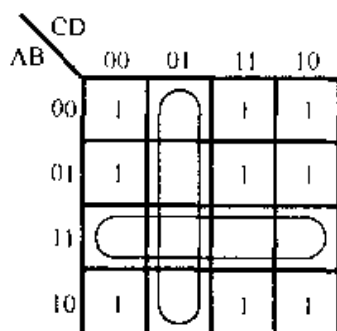


图 1.5.4 简化为与或非表达式

(3) 将与或式转换为或非表达式

先将与或式转换为或与表达式,然后应用摩根定理即可方便地转换为或非表达式。例如, $F = \bar{A}\bar{B}D + A\bar{D} + B\bar{D} + A\bar{B}C$, 先按照与或式转换为或与式的方法(参见图 1.5.3),求得

$$F = (\bar{B} + \bar{D})(\bar{A} + C + \bar{D})(A + B + D)$$

然后应用摩根定理,即求得

$$F = \overline{(\bar{B} + \bar{D})(\bar{A} + C + \bar{D})(A + B + D)} = \overline{\bar{B} + \bar{D}} + \overline{\bar{A} + C + \bar{D}} + \overline{A + B + D}$$

*1.6 多输出函数的简化

在实际应用中,有许多逻辑电路都是具有相同的输入变量而具有多个输出端,描述这类逻辑电路就要采用多输出函数。在对多输出函数简化时,如果仅将某个函数

简化为最佳形式,则往往整体函数并不是最佳形式,因此,逻辑电路也不是最佳的形式。例如,

$$F_1 = A\bar{B} + C$$

$$F_2 = A\bar{B}\bar{C} + BC$$

显然, F_1 和 F_2 均为最简形式,但是,实现它的整体电路并不是最佳的。若使 $F_1 = A\bar{B} + C = A\bar{B}\bar{C} + C$, 则 F_1 和 F_2 之间可公用乘积项 $A\bar{B}\bar{C}$, 使整体电路更佳。由此可见,在多输出函数简化时,单个函数并不要求最简形式,而是尽可能使公用的乘积项最多。

多输出函数的简化需要采用系统简化法^[3,6],即 Q-M 法。此法具有较严格的算法,并且适用于简化具有较多输入变量和多个输出的函数,但是,它的计算过程繁琐,需列出大量表格,若借助计算机设计则较方便。在手算的条件下,即使用 Q-M 法简化四输入变量和三输出的多输出函数,其简化过程也非常繁琐。因此,笔者在教学实践中,探讨借助于乘积卡诺图来简化多输出函数和设计多端输出组合电路,并获得最佳或次佳的良好效果^[5,24]。下面结合例子来说明乘积卡诺图法的设计步骤和应注意的问题。

例 1.6.1 简化下列函数:

$$\left. \begin{aligned} F_1 &= (A, B, C, D) = \sum(5, 7, 8, 9, 10, 11, 13) \\ F_2 &= (A, B, C, D) = \sum m(1, 7, 11, 15) \\ F_3 &= (A, B, C, D) = \sum m(1, 6, 7, 8, 9, 10, 11) \end{aligned} \right\} \quad (1.6.1)$$

解 (1) 画出单个函数和所有函数乘积的卡诺图,如图 1.6.1 所示。

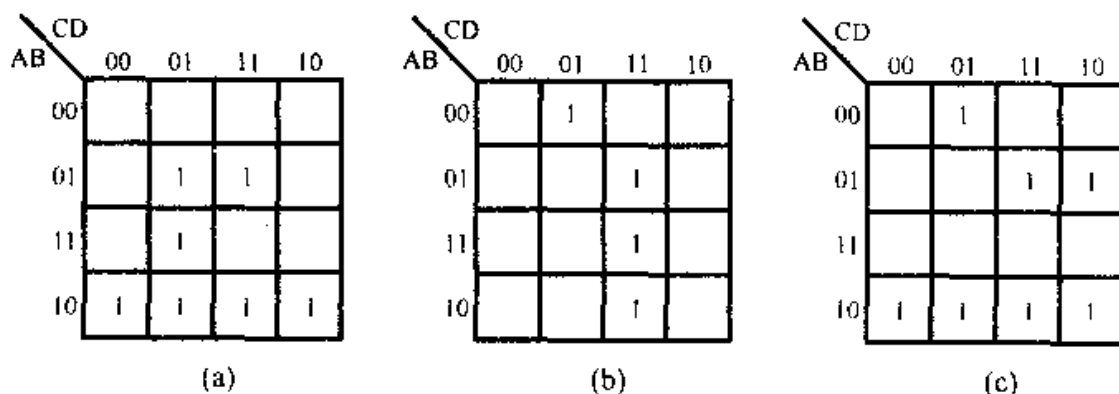


图 1.6.1 例 1.6.1 卡诺图

(a) F_1 (b) F_2 (c) F_3

函数乘积卡诺图是由单个函数卡诺图相乘得到的,如 $F_1 \cdot F_2$ 卡诺图就是由 F_1 和 F_2 卡诺图相乘得到的,由于在 F_1 和 F_2 卡诺图上 m_7 均为 1,所以, $F_1 \cdot F_2$ 卡诺图上 m_7 为 1,如此类推,可完成所有乘积卡诺图,见图 1.6.2。

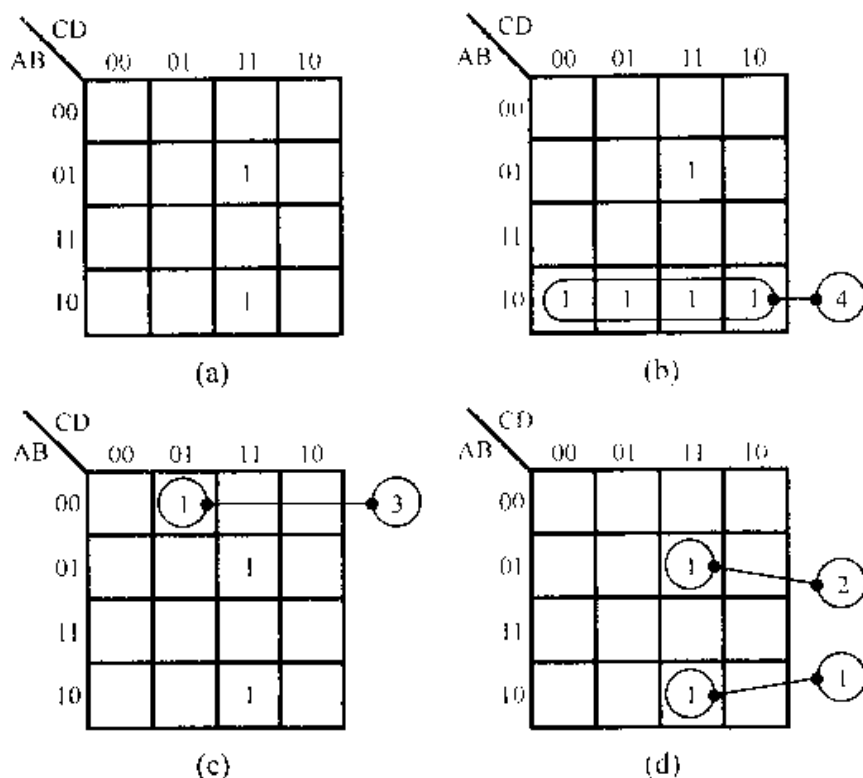


图 1.6.2 例 1.6.1 乘积卡诺图

(a) $F_1 \cdot F_2$ (b) $F_2 \cdot F_3$ (c) $F_2 \cdot F_3$ (d) $F_1 \cdot F_2 \cdot F_3$

(2) 在函数乘积卡诺图上求公有质蕴涵项^①。

在函数乘积卡诺图上确定两个或多个函数所公有的质蕴涵项。首先从最高阶乘积卡诺图开始，然后逐渐向较低阶卡诺图进行。所以，首先考虑 $F_1 \cdot F_2 \cdot F_3$ 卡诺图，圈出 3 个函数公有的质蕴涵项为圈①和②，见图 1.6.2(d)；然后再考虑两个函数的乘积卡诺图，这时，在 3 个函数乘积卡诺图上已圈过的质蕴涵项不必再圈了，因为它们已被 3 个函数所公有的质蕴涵项所覆盖。由此可画出两个函数所公有的质蕴涵项，如图 1.6.2(a)~(c)所示。

(3) 确定每个函数的最佳质蕴涵项组。

在各单个函数卡诺图上直接确定每个函数的最佳质蕴涵项组，其要求是将卡诺图上的 1 全部覆盖，为使整体电路最佳，应遵循下列优化准则。

(a) 优先选用多个及两个函数公有的质蕴涵项，剩余 1 再由单个函数产生新的且为它独有的质蕴涵项来覆盖。

例如，在 F_1 卡诺图上优先选用圈②（3 个函数的公有项）和圈④（两个函数的公有项），余下的 1 再产生新的且为 F_1 所独有的质蕴涵项圈⑤来覆盖，如图 1.6.3(a)所示。

(b) 函数的简化次序也会对简化结果产生影响。例如，首先简化 F_2 ，在它的卡

① 质蕴涵项：在与或表达式中，构成最简逻辑函数的与项称为质蕴涵项。

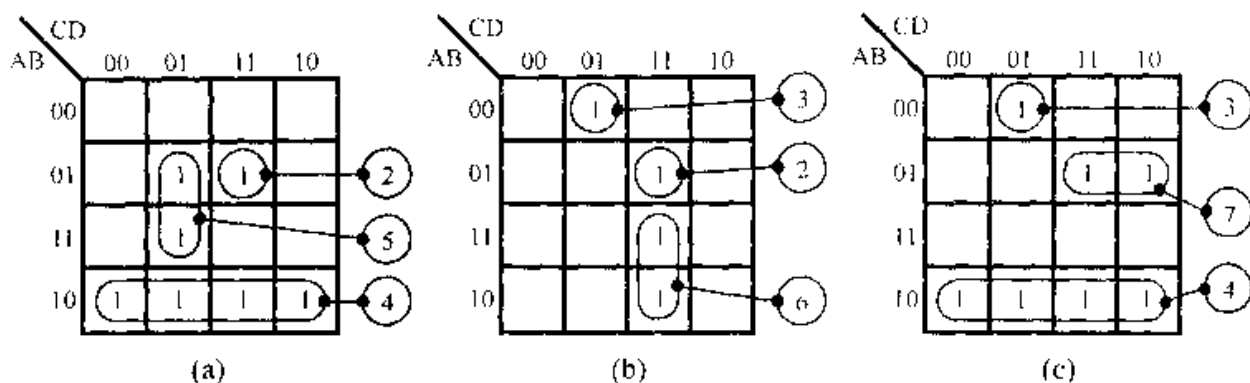


图 1.6.3 例 1.6.1 优化卡诺图

(a) F_1 (b) F_2 (c) F_3

诺图上既可选用圈②,也可选用圈①,这对 F_2 来说是等效的。如选用了圈①,然后再简化 F_1 ,则此时圈②必须选用,而圈①已含在圈④中而不必选用,这样得到的整体函数的简化结果不是最佳。因此,排列简化的次序应该是将容易确定公有质蕴涵项者放在前,有多种等效方案者放在后。

(c) 次序在前的函数上已选用的公有质蕴涵项,在其他函数卡诺图上应优先选用。

例如,在 F_2 卡诺图上,优先选用圈③(两个函数的公有项),圈①和②都是 3 个函数公有的质蕴涵项,但是,由于在 F_1 卡诺图上已选用圈②,故这里应选用圈②,余下的 1 再产生 F_2 独有的质蕴涵项圈⑥来覆盖。此处若选用圈①而舍弃圈②,对 F_2 来说是等效的,但对整体电路来说将会多一个质蕴涵项,同样确定 F_3 卡诺图上的圈③、④和⑦,分别示于图 1.6.3(b)和(c)。

应用上述优化规则便可从函数优化卡诺图 1.6.3 上直接写出各函数的最佳质蕴涵项组的表达式

$$\left. \begin{aligned} F_1 &= A\bar{B} + B\bar{C}D + \bar{A}BCD \\ F_2 &= \bar{A}BCD + ACD + \bar{A}\bar{B}\bar{C}D \\ F_3 &= A\bar{B} + \bar{A}BC + \bar{A}\bar{B}\bar{C}D \end{aligned} \right\} \quad (1.6.2)$$

在式 1.6.2 中, $A\bar{B}$ 是 F_1 和 F_3 的公有项, $\bar{A}BCD$ 是 F_1 和 F_2 的公有项, $\bar{A}\bar{B}\bar{C}D$ 是 F_2 和 F_3 的公有项,其余各项是单个函数所独有的质蕴涵项,如用电路实现式(1.6.2),则可获得整体电路最佳的结果。

小 结

逻辑函数是本课程的基础理论,本章从逻辑函数的基本概念出发,阐述了逻辑函数的基本运算、基本定理及运算规则。运用这些基本定理和运算规则来简化逻辑

函数是本章的重点内容。

卡诺图是设计和分析逻辑电路的重要工具。本章介绍了卡诺图的构成、引出相邻最小项、相邻单元等基本概念,旨在帮助读者正确地运用卡诺图进行逻辑函数简化,合理应用任意项有利于逻辑函数的进一步简化。卡诺图法简化逻辑函数具有直观和简便等优点,应用十分广泛。所以,这是本章讨论的另一个重点内容,必须熟练掌握。

逻辑函数有 4 种表示方法:真值表、逻辑式、卡诺图和逻辑图。可以用它们来描述同一个逻辑函数,所以,它们之间可以相互转换。

用乘积卡诺图法简化多输出逻辑函数,是作者在教学实践中探讨的一种新方法。它不仅为多端输出组合电路的设计提供了一个直观和简便的工程设计方法,同时也可提高读者灵活运用卡诺图的能力。

思考题和习题

1.1 解释下列名词:

最小项、最大项、正逻辑、负逻辑、逻辑函数标准型。

1.2 试述下列定理或规则,说明其应用:

摩根定律、添加项规则、对偶规则、反演规则。

1.3 用代数法简化逻辑函数应遵循什么准则?卡诺图法简化逻辑函数应遵循什么准则?

1.4 将下列各数按权展开:

(1) 10101 (2) 0.10101 (3) 1010.101

1.5 将下列十进制数转换为二进制数:

(1) 163 (2) 0.525 (3) 41.41

1.6 将 $(123)_8$ 转换为二进制数及十进制数。

1.7 将 $(8.705)_{10}$ 转换为六进制数,要求 $\pm(0.3)^3$ 的精度。

1.8 已知有 4 个运动员参加拳击比赛,举行拳击比赛的条件有:

- (1) 只有在有他人在旁的条件下, A 才可与任何人比赛;
- (2) B 只与 C 比赛,并且是在无其他运动员在场的情况下;
- (3) C 可与任何人比赛,但 D 在场则拒绝参加比赛;
- (4) D 宣布不与任何人比赛。

试求出举行一次拳击比赛的逻辑表达式,并用逻辑语言解释之。

1.9 假定:

- (1) A 从来不说话;
- (2) B 只有在 A 在场时才说话;
- (3) C 在任何情况下、甚至一个人也说话;

(4) D只有在A在场时才说话。

试求房间里没有人说话的逻辑表达式。

1.10 指出A, B, C取哪些值时, F为1:

(1) $F(A, B, C) = AB + \bar{A}C$

(2) $F(A, B, C) = \overline{A + BC(A + B)}$

1.11 用真值表验证下列等式是否成立:

(1) $\overline{A + B} = \bar{A} \cdot \bar{B}$

(2) $A\bar{B} + \bar{A}B = (\bar{A} + B)(A + \bar{B})$

1.12 用代数法证明:

$$\overline{A + C + D \cdot (A + \bar{C})(\bar{A} + B)(\bar{B} + C)} = A\bar{C} + \bar{A}BD + \bar{B}CD$$

1.13 用代数法简化逻辑函数:

(1) $F = A(\bar{B} + C + D)(B + \bar{D})$

(2) $F = \bar{A}\bar{B} + (AB + A\bar{B} + \bar{A}B)C$

(3) $F = A + A\bar{B}\bar{C} + A\bar{C}D + (\bar{C} + \bar{D})E$

(4) $F = A\bar{B}(C + D) + B\bar{C} + \bar{A}\bar{B} + \bar{A}C + BC + \bar{B}\bar{C}\bar{D}$

(5) $F = (A + B)(A + C)(A + \bar{C})$

(6) $F = \overline{(A + B\bar{C})(\bar{A} + \bar{D}E)}$

(7) $F = A(A + \bar{B} + \bar{C})(\bar{A} + C + D)(E + \bar{C}\bar{D})$

1.14 用代数法简化逻辑函数:

(1) $F(A, B, C) = \sum m(2, 3, 6, 7)$

(2) $F(A, B, C, D, E) = \prod M(0, 4, 8, 12, 16, 20, 24, 28)$

1.15 将下列函数变化为标准或与式:

(1) $F(A, B, C) = \sum m(1, 3, 7)$

(2) $F(A, B, C, D) = \sum m(0, 2, 6, 12, 13, 14)$

1.16 将下列函数变化为标准与或式:

(1) $F(A, B, C) = \prod M(0, 3, 6, 7)$

(2) $F(A, B, C, D) = \prod M(0, 1, 2, 3, 4, 6, 12)$

1.17 将下列函数展开为标准与或式:

(1) $F(A, B, C, D) = AB + \bar{A}\bar{B} + C\bar{D}$

(2) $F(A, B, C) = (A + B)(\bar{B} + C)$

1.18 将下列函数表示成标准或与式:

(1) $F(A, B, C) = A \oplus B + \bar{A}C$

(2) $F(A, B, C, D) = (A + \bar{B} + C)(A + \bar{B})(A + \bar{C} + \bar{D})(B + \bar{C} + \bar{D})$

1.19 写出下列函数的补函数表达式:

$$(1) F = (AB + \overline{A}\overline{B})(C + D)(E + \overline{C}\overline{D})$$

$$(2) F = A + B + \overline{C} + \overline{D} + \overline{E}$$

1.20 写出下列函数的对偶式:

$$(1) F = AB + CD + \overline{A}C$$

$$(2) F = A(\overline{B}C + B\overline{C}) + A\overline{C}$$

$$(3) F = (\overline{A} + B)(B + \overline{A}C)$$

1.21 证明下列等式成立(方法不限定):

$$(1) A \oplus \overline{B} = \overline{A} \oplus B = \overline{A \oplus B}$$

$$(2) \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC = A \oplus B \oplus C$$

$$(3) AB + BC + AC = (A+B)(B+C)(A+C)$$

$$(4) A\overline{B} + B\overline{C} + \overline{A}C = \overline{A}B + \overline{B}C + A\overline{C}$$

1.22 用卡诺图法简化下列逻辑函数, 写出最简与或式及最简或与式:

$$(1) F(A,B,C,D) = \sum m(3,4,5,7,9,13,14,15)$$

$$(2) F(A,B,C,D) = \sum m(2,3,12,13,14,15)$$

$$(3) F(A,B,C,D) = \sum m(0,1,4,5,6,7,9,10,13,14,15)$$

$$(4) F(A,B,C,D) = \sum m(0,2,4,6,9,11,13,15,16,17,20,21,25,27,29,31)$$

1.23 用卡诺图法简化逻辑函数, 写出最简或与式及与或式:

$$(1) F(A,B,C,D) = \prod M(1,3,5,7,13,15)$$

$$(2) F(A,B,C,D,E) = \prod M(0,1,2,3,4,6,8,10,12,13,14)$$

1.24 利用卡诺图简化下列逻辑函数:

$$(1) F(A,B,C,D) = \sum m(3,5,6,9,12,13,14,15) + \sum \phi(0, 1, 7)$$

$$(2) F(A,B,C,D) = \sum m(1,3,4,7,11,13,14) + \sum \phi(2,5,12,15)$$

$$(3) F(A,B,C,D,E) = \sum m(0,1,2,6,17,18,20,21,22,24,27,30,31) + \sum \phi(4,5,10,16,26,8)$$

$$(4) F(A,B,C,D,E) = \overline{A}\overline{B}C + A\overline{B}D + \overline{A}\overline{B}\overline{C}\overline{D} + \sum \phi$$

式中, $\sum \phi = ABC + A\overline{B}\overline{D}$

$$(5) F(A,B,C,D) = \overline{A}\overline{B}\overline{C} + ABC + \overline{A}\overline{B}\overline{C}\overline{D} + \sum \phi$$

式中, $\sum \phi = A \oplus B$

1.25 简化下列多输出函数:

$$(1) F_1 = A\overline{B} + B\overline{C} + \overline{A}C; F_2 = \overline{A}B + \overline{B}C$$

$$(2) F_1 = A\overline{C} + \overline{B}C + \overline{A}C; F_2 = A\overline{B} + B\overline{C} + \overline{A}B$$

$$(3) F_1(A,B,C,D) = \sum m(1,3,4,5,6,7,15)$$

$$F_2(A,B,C,D) = \sum m(1,3,10,14,15)$$

第2章 集成逻辑门

内容提要 在脉冲与数字电路中,二极管、晶体三极管和场效应管通常是作为开关元件使用的,所以,本章先简要介绍它们的开关特性。在此基础上,介绍 TTL 逻辑门和 MOS 逻辑门,重点讨论它们的典型电路结构、工作原理和性能指标。对射极耦合逻辑门(ECL)只进行一般介绍。最后介绍使用集成逻辑门时应注意的问题。

2.1 数字集成电路概述

集成电路是将晶体管、电阻、电容等元件和内部电路连线一起做在一块半导体基片上,然后封装在一个外壳内构成的一个电路单元,通常又称为集成电路组件。

集成电路种类繁多,可从不同角度分类。

集成电路依据处理的信号对象不同,可分为数字集成电路和模拟集成电路两大类。

数字集成电路依据内部有源器件的不同,可分为双极型晶体管集成电路(即 TTL 型集成电路)和绝缘栅场效应管集成电路两大类(MOS 型电路)。

数字集成电路依据集成度,可分为4类:小规模集成(SSI),单片内包含元件少于100个;中规模集成(MSI),单片内包含元件为100~1000个;大规模集成(LSI),单片内包含元件为1000~100000个;超大规模集成(VLSI),单片内包含元件在100000个以上。

数字集成电路依据芯片的设计方法,可分为3类:(a)标准逻辑器件,如采用小、中、大规模集成的各种通用产品;(b)可编程逻辑器件,如 PROM、EPROM、GAL、FPGA、CPLD 等;(c)专用集成芯片,它可分为半定制集成电路(如门阵列和标准单元)和全定制集成电路(生产商专门为某用户设计和制作的芯片)两类。

本书主要介绍数字集成电路中的标准逻辑器件和可编程逻辑器件,对于脉冲单元电路和数模与模数转换器等模拟集成电路也作适当介绍。数字集成电路的基本逻辑单元是集成逻辑门,所以本章将重点介绍。

2.2 晶体二极管和三极管的开关特性

2.2.1 晶体二极管的开关特性

1. 稳态开关特性

电路处于相对稳定状态下,晶体二极管所呈现的开关特性称为稳态开关特性。图 2.2.1 所示的是硅二极管的伏安特性曲线。

当外加电压 $v_D=0$ 时, $i_D=0$; 当外加正电压达到某一定值之后, i_D 才明显上升。将 i_D 开始明显上升时所对应的电压称为二极管的导通电压 V_T (阈值电压)。在电路分析中, 当 $v_D \geq V_T$ 时, 二极管完全导通, i_D 随 v_D 的增加而按指数规律上升; 反之, 若二极管完全导通, 则二极管两端的电压降近似等于导通电压 V_T 。硅二极管 V_T 为 $0.7V$, 锗二极管为 $0.3V$ 。

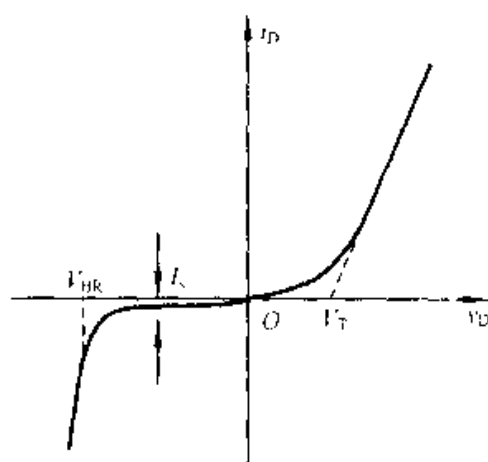


图 2.2.1 硅二极管的伏安特性曲线

在 $0 \leq v_D < V_T$ 范围内有很微弱的电流, 二极管不能完全导通, 可以认为它处于截止状态。当外加反向电压时, PN 结反偏, 在一定范围内都有反向漏电流 I_S (反向饱和电流)。对于硅二极管, I_S 小于 $1 \mu A$; 对于锗二极管, I_S 可以从几十 μA 到几百 μA 。当外加反向偏压超过反向击穿电压 V_{BR} 时, I_S 将急剧增加, 二极管被击穿。

需要指出: 二极管的 V_T 和 I_S 都与温度有关。当温度升高时, V_T 下降, 温度每升高 $1^\circ C$, V_T 约下降 $2.4mV$ 。当温度升高时, I_S 增加, 对于锗管, 每增加 $12^\circ C$, I_S 约增加一倍; 对于硅管, 温度每增加 $8^\circ C$, I_S 约增加一倍。由于锗管的 I_S 很大, 所以温度对锗管的影响比对硅管的影响更大。

2. 瞬态开关特性

电路处于瞬变状态下, 晶体二极管所呈现的开关特性称为瞬态开关特性 (简称瞬态特性), 即二极管由导通到截止, 或者由截止到导通的瞬态特性。

在输入信号频率不高时, 二极管的导通与截止状态之间的转换可以认为是瞬间完成的。但是, 在输入信号频率很高时, 二极管的开关惰性便不可忽略了, 因为完成两种状态的转换都需要一定的时间。二极管由截止转向导通所需时间称为正向恢复时间 (开通时间) t_{on} ; 二极管由导通转向截止所需时间称为反向恢复时间 t_R (关断时

间)。二者之和称为二极管的开关时间,它的长短便决定了二极管的最高工作频率。由于 $t_{on} \ll t_R$, 所以下面仅讨论反向恢复时间。

(1) 反向恢复时间

图 2.2.2(a)是一个简单的二极管开关电路,输入信号 v_i 在 V_F 和 $V_R(V_R < 0V)$ 之间跳变。当 $v_i = V_F$ 时,二极管正向导通, $i_D = I_F = (V_F - V_D)/R \approx V_F/R$ 。在 t_1 时刻, v_i 由 V_F 跳变到 V_R , 如果二极管是一个理想开关,它应该立刻截止, i_D 应从 I_F 变为反向漏电流 I_s 。但实际上并非如此,当外加电压突然反向时,会出现一个很大的反向电流 $I_R \approx V_R/R$, 此时二极管仍然导通,其两端仍然有很小的正向压降。在经过一段时间 t_s 之后,这个反向电流才逐渐减小,再经过时间 t_f 之后,才趋于反向漏电流 I_s 。这一反向恢复过程的 i_D 和 v_D 波形如图 2.2.2(b)所示。 t_s 称为存储时间, t_f 称为下降时间,反向恢复时间 $t_R = t_s + t_f$ 。

(2) 存储电荷效应

二极管具有反向恢复过程的原因是它具有存储电荷效应。

在正向电压作用下, P 区中的空穴向 N 区扩散, N 区的电子向 P 区扩散, 不仅使空间电荷区变窄, 而且在对方区域中都有相当数量的电荷存储, 靠近 PN 结处浓度大, 离 PN 结越远浓度越小。图 2.2.3 是存储电荷分布的示意图。这些扩散到对方区域并积累起来的少数载流子是非平衡载流子, 这种非平衡少数载流子的积累现象称为存储电荷效应。

在外加电压由 V_F 跳变到 V_R 时, 存储电荷不会立即消失, 它们的存在使 PN 结仍然维持正向偏置, 呈现低阻特性, PN 结的结电阻与 R 的阻值相比可以忽略, 所以反向电流很大, 其值由 V_R 和 R 决定, 并在 t_s 时间内保持不变。但是, 在外加反向电压作用下, P 区所储存的电子被拉回 N 区, 形成电子漂移电流; N 区所储存的空穴也被拉回 P 区, 形成空穴漂移电流, 这两个漂移电流就构成了很大的反向电流 I_R 。反向

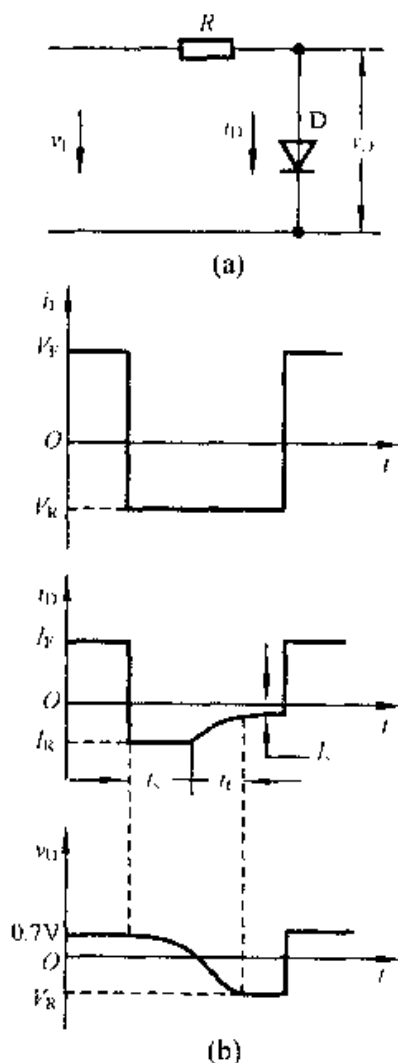


图 2.2.2 二极管的瞬态特性

(a) 电路图 (b) 波形图

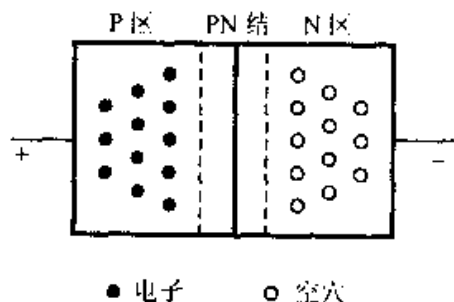


图 2.2.3 存储电荷分布示意图

电流一方面使存储电荷逐渐消失, 另一方面使存储电荷不断与多数载流子复合。因此, 存储电荷愈来愈少, 最终存储电荷消散, 存储时间 t_s 结束。

存储电荷消散之后, PN 结上电压反向, 反向电压对 PN 结充电, 空间电荷区逐渐变宽, 二极管内阻增加, 反向电流逐渐减少到稳态时的反向饱和电流, 下降时间 t_f 结束(参见图 2.2.2)。

反向恢复时间与二极管本身结构和电路参数有关。PN 结面积越大, 管内存储电荷越多, 反向恢复时间越长; 正向电流越大(V_F 大)则存储电荷越多, 反向恢复时间越长; 反向电流越大(V_R 大)则存储电荷消散快, 反向恢复时间越短。

2.2.2 晶体三极管的开关特性

1. 稳态开关特性

图 2.2.4(a)所示的是一个 NPN 管共发射极开关电路, 图(b)所示的是其输出特性。

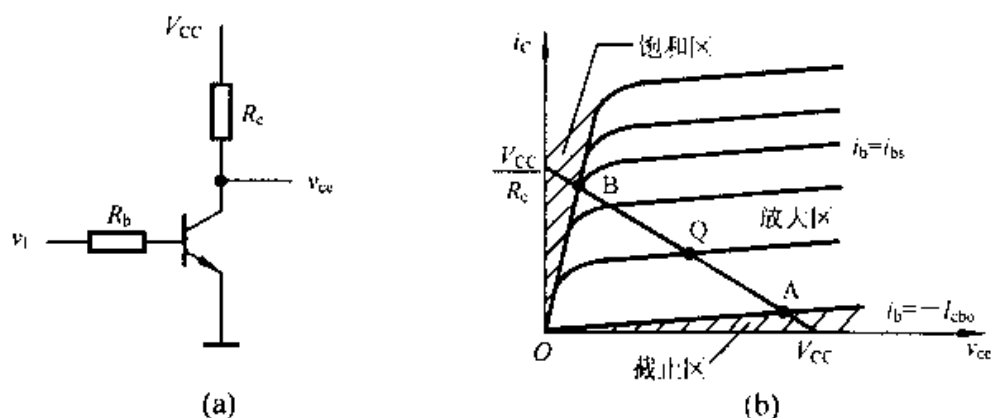


图 2.2.4 三极管开关电路及稳态输出特性

(a) 电路图 (b) 输出特性

R_b 为基极电阻, 通过它来控制基极电流大小; R_c 为集电极电阻, 通过它来限制集电极电流大小; 基极 b 可以加控制信号来实现开关的闭合状态; 集电极 c 及发射极 e 形成开关的两个端点: c、e 两端电压即作为开关电路的输出电压 v_{ce} 。

当输入电压 v_1 为高电平时, 三极管饱和导通(由 R_b 、 R_c 阻值保证), 输出低电平; 当 v_1 为低电平时, 三极管截止, 输出高电平。这就是理想的三极管稳态开关特性。

实际的三极管开关特性与工作状态有关, 由图 2.2.4(b)所示特性曲线可见: 为了使三极管工作在开关“接通”状态, 就应该使之工作于饱和区; 为了使它工作在开关“断开”状态, 应该使之工作于截止区。在饱和型开关电路中, 稳态时, 三极管只工作在饱和或截止这两种状态, 只有在三极管的开关状态发生转换过程中, 才经过放大状态。所以, 应该重点讨论饱和区和截止区的工作情况。

截止区 当输入信号 $v_1 < 0$ 时, $v_{be} < 0$ 和 $v_{bc} < 0$ 。这时, 电路工作点位于 A 点,

由于三极管的两个 PN 结均处于反向偏置, 故载流子的扩散运动无法进行, 只有极少数因热激发而产生的少数载流子流过两个 PN 结, 形成反向电流 I_{cbo} 和 I_{cbo} 。一般情况下, 可忽略 I_{cbo} , 因此有

$$i_b = -I_{cbo} \quad \text{和} \quad v_{be} \leq 0$$

实际上, 在 $0 < v_{be} \leq V_T$ 时, i_b 近似为 0, 三极管已截止。因此, 常用 $i_b = 0$ 或 $v_{be} \leq V_T$ 作为导通与截止的分界条件。式中, V_T 为发射结门限电压, 对于硅开关三极管 $V_T \approx 0.6\text{V}$, 对于锗开关三极管, $V_T \approx 0.2\text{V}$ 。

放大区 随着输入信号 v_i 增大, v_{be} 将增大。当 $v_i > V_T$ 时, 三极管处于放大状态, 工作点 Q 落在输出特性曲线的平坦部分。三极管处于放大状态时的特点是

$$i_c = \beta_0 i_b \\ v_{be} > 0 \quad \text{和} \quad v_{bc} < 0$$

饱和区 当输入信号继续增大时, i_b 增大, i_c 也增大, 所以, v_{ce} 下降。当 $v_{ce} = v_{be}$ 时, 工作点落到 B 点位置, 晶体管失去放大能力, 处于饱和状态。饱和状态的特点是发射结和集电结都处于正向偏置, 且有

$$v_{be} > v_{ce}$$

三极管饱和时的压降称为饱和压降。发射结饱和压降记为 V_{bes} ; 集电结饱和压降记为 V_{ces} 。对于硅开关三极管, $V_{bes} \approx 0.7\text{V}$ 和 $V_{ces} \approx 0.3\text{V}$; 对于锗开关三极管, $V_{bes} \approx 0.3\text{V}$ 和 $V_{ces} \approx 0.1\text{V}$ 。

当晶体管进入饱和状态之后, 集电极电流 i_c 不再随 i_b 的增加而增大, 其值近似由电源电压 V_{CC} 和电阻 R_c 决定。集电极饱和电流记为 I_{cs} , 则有

$$I_{cs} = \frac{V_{CC} - V_{ces}}{R_c} \approx \frac{V_{CC}}{R_c}$$

饱和基极电流由基极回路确定, 三极管刚进入饱和时的基流称为临界饱和基流 I_{bs} , 即

$$I_{bs} = \frac{I_{cs}}{\beta_0} \approx \frac{V_{CC}}{\beta_0 R_c}$$

由此得到三极管工作于饱和导通状态的条件为 $i_b \geq I_{bs}$, 且 i_b 愈大, 三极管饱和愈深。

2. 瞬态开关特性

在开关电路中, 三极管由基极的控制信号 v_i 控制工作状态, 它一会儿工作于截止状态, 一会儿又工作于饱和状态。但是, 工作状态之间的转换都不是在瞬间完成, 而需要一个过渡过程, 下面讨论在这个过渡过程中各电流及电压与时间的关系, 即研究瞬态开关特性, 如图 2.2.5 所示。

延迟时间 从输入电压正跳变开始时刻到三极管集电极电流 i_c 上升到 $0.1I_{cs}$ 所需要的时间, 记为 t_d 。

当 $v_1 = E_2$ 时, 发射结为反向偏置, 势垒区宽; 当 v_1 发生正跳变之后, 势垒区由宽变窄, 由于发射结由反向偏置变为正向偏置都需要一定的时间, 所以虽有正向基极电流, 但不能立即产生集电极电流 i_c 。随着基极电流注入, 进入基区电子逐渐增多, 才开始出现 i_c 。

t_d 的长短除了与管子本身结构(如发射结面积等)有关之外, 还与电路工作条件有关。三极管截止愈深($|E_2|$ 愈大), t_d 愈长; 正向驱动电流愈大(E_1 愈大), t_d 愈短。

上升时间 发射区向基区注入电子使基区电子浓度逐渐加大, 因而 i_c 逐渐增加。将 i_c 从 $0.1I_{cs}$ 增加到 $0.9I_{cs}$ 所需时间称为上升时间, 记为 t_r 。

t_r 的长短同样取决于三极管本身结构和电路工作条件。例如, 正向驱动电流愈大(E_1 愈大), 则 t_r 愈短。

存储时间 从 v_1 负跳变开始时刻到三极管集电极电流 i_c 下降到 $0.9I_{cs}$ 所需要的时间, 记为 t_s 。

三极管进入饱和状态之后, 集电结吸取电子能力降低, 多余电子在基区构成多余存储电荷。当 v_1 负跳变之后, 多余存储电荷消散需要一定时间, 即构成存储时间 t_s 。决定 t_s 长短的因素和上面讨论的相同。例如, 基极反向电流愈大, 存储电荷消散越快, t_s 也愈短; 饱和愈浅, 存储电荷少些, t_s 也愈短。

下降时间 集电极电流 i_c 从 $0.9I_{cs}$ 下降至 $0.1I_{cs}$ 所需要的时间, 记为 t_f 。

多余存储电荷消散后, 随着外加反向电压的作用, 势垒区不断加宽, 由发射区向基区注入电子减少, 电子浓度下降, 所以, i_c 不断下降, 当 i_c 下降至 $0.1I_{cs}$ 时, 下降过程基本结束。

在定义开关参数时, 将延迟时间 t_d 和上升时间 t_r 之和定义为开启时间 t_{on} ; 把存储时间 t_s 与下降时间 t_f 之和定义为关闭时间 t_{off} 。

由以上分析可以知道, 饱和型开关电路的工作速度决定于三极管本身结构和电路工作条件。为了提高开关电路的工作速度, 应该合理地选用三极管(如选用高速开关管); 同时在电路设计时采取改善电路工作条件的措施, 例如, 一种简单而有效的方法是在基极电阻 R_b 上并联一个加速电容, 这在有关电子线路课程中已介绍了。

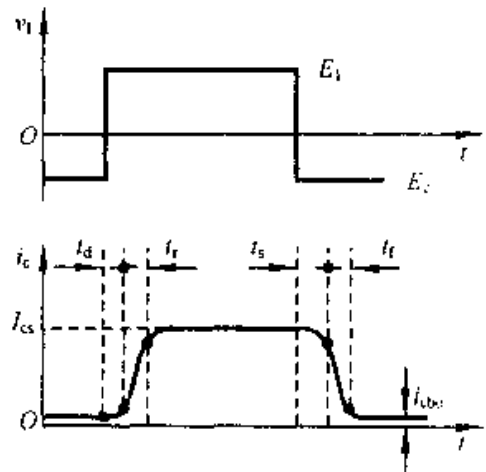


图 2.2.5 三极管瞬态开关特性

2.3 TTL 逻辑门

晶体管集成逻辑门是目前广泛应用的逻辑器件。显然，它内部的有源器件是双极型晶体管，通常称它为 TTL 逻辑门。

TTL 逻辑器件分为 54 系列和 74 系列两大类，这两类器件具有完全相同的电路结构、逻辑功能和电气性能参数。所不同的是 54 系列比 74 系列的工作环境温度范围更宽，电源允许的工作范围更大。74 系列的工作环境温度规定为 $0^{\circ}\text{C} \sim 70^{\circ}\text{C}$ ，电源电压工作范围为 $5\text{V} \pm 5\%$ ；而 54 系列的工作环境温度规定为 $-55^{\circ}\text{C} \sim +125^{\circ}\text{C}$ ，电源电压工作范围为 $5\text{V} \pm 10\%$ 。

依据工作速度和功耗，这两大类器件又可以分为 4 个系列：

(a) 标准通用系列。国产型号为 CT54/74 系列，与国际上 SN54/74 系列相当，沿用的部标型号是 T1000 系列。

(b) 高速系列。国产型号为 CT54H/74H 系列，与国际上 SN54H/74H 系列相当，沿用的部标型号是 T2000 系列。

(c) 肖特基系列。国产型号为 CT54S/74S 系列，与国际上 SN54S/74S 系列相当，沿用的部标型号是 T3000 系列。

(d) 低功耗肖特基系列。国产型号为 CT54LS/74LS 系列，与国际上 SN54LS/74LS 系列相当，沿用的部标型号为 T4000 系列。

任何逻辑运算都可用与非门实现，所以，与非门是一种最基本的、应用最广泛的逻辑门。本节以 TTL 与非门为例来介绍 TTL 逻辑门的工作原理和性能指标。

2.3.1 浅饱和型 TTL 与非门

不同系列的与非门电路结构有些差异，因为不同系列产品在工作速度和功耗指标上有不同的要求，但是，这些差异并不影响对电路实现逻辑功能的分析。所以，我们以高速系列的浅饱和型 TTL 与非门为例来介绍与非门的电路结构和性能指标，电路如图 2.3.1 所示。

电路由一个多发射极晶体管 T_1 和其他 5 只晶体管 ($T_2 \sim T_6$) 组成。电路结构可以分为 4 个部分：由多发射极晶体管 T_1 和

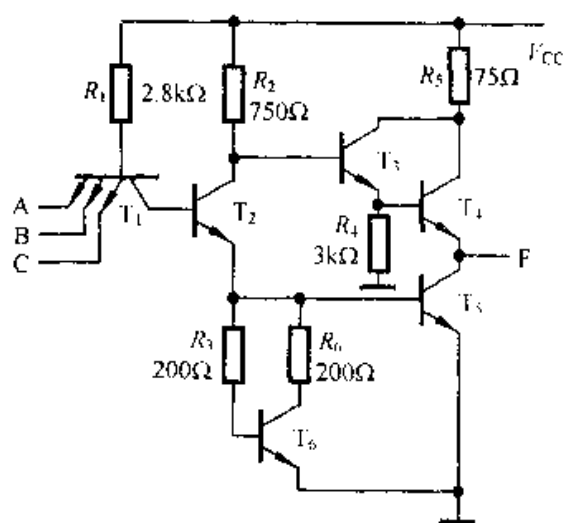


图 2.3.1 TTL 与非门电路

电阻 R_1 构成输入级, 输入信号通过 T_1 的多发射极实现与逻辑; T_2 和 R_2 、 R_3 构成中间级, 完成与门输出信号的倒相放大功能, 从 T_2 的集电极和发射极同时输出两个相位相反的信号, 分别控制 T_3 和 T_5 的工作状态; T_3 、 T_4 、 T_5 和 R_4 、 R_5 组成推挽式输出级(又称为图腾柱输出电路); T_6 和 R_3 、 R_6 构成有源泄放网络, 用以改善与非门性能指标。

1. 输出高电平的工作状态

设输入 A 为低电平, B 和 C 为高电平, 如图 2.3.2(a)所示。因为 A 为 0.3V, 所以, T_1 管的 b_1e_1 结导通, T_1 管基极电位 V_{b1} 为 1V, 因此, 另外两个发射结 b_1e_2 和 b_1e_3 均为反偏截止。 T_1 的集电极通过 T_2 的集电极和 R_2 与电源 V_{CC} 连接, 所以, T_2 的集电结为反偏截止, T_1 的 I_{C1} 等于 T_2 的 I_{cho} , 因此, $I_{c1} \ll \beta I_{b1}$, T_1 管处于深饱和, $V_{ces1} \approx 0.1V$, 所以, T_2 管基极电位为 0.4V, T_2 管为截止状态, 射极电流为 0A, 因此, T_5 和 T_6 均为截止状态。由于 T_2 截止, $V_{c2} \approx V_{CC}$, 它驱动 T_3 和 T_4 导通, 因此, 输出电压

$$v_O = V_{CC} - (I_{b3}R_2 + V_{be3} + V_{be4})$$

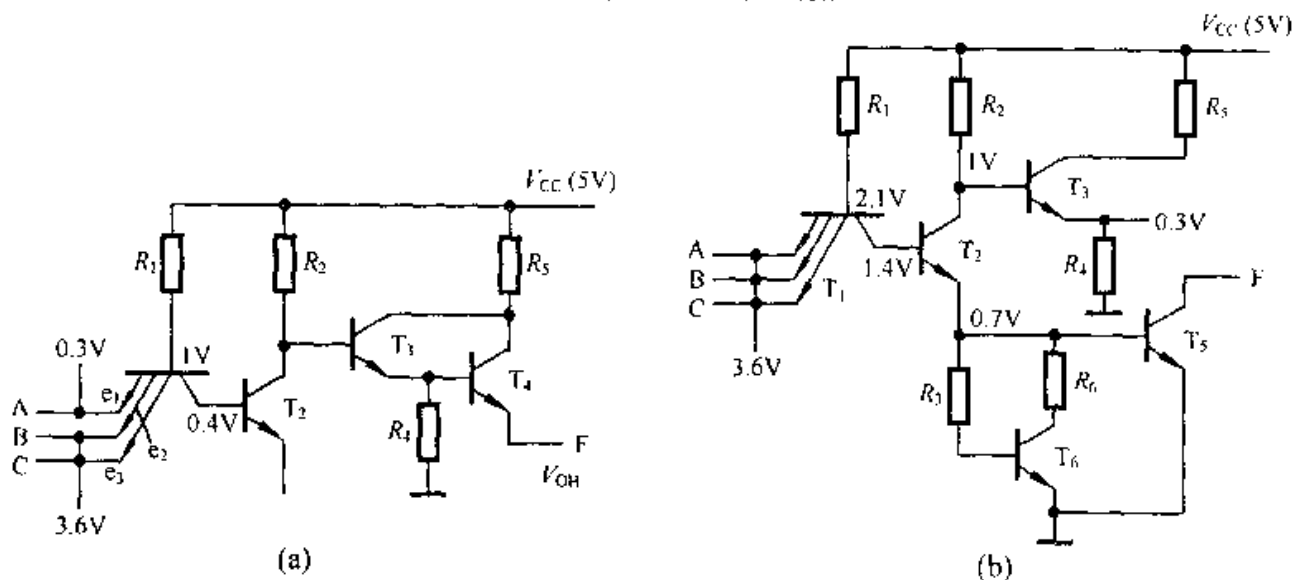


图 2.3.2 TTL 与非门的两种工作状态

(a) 输出为高电平时的等效电路 (b) 输出为低电平时的等效电路

因为 T_3 为射随器, I_{b3} 很小, 所以忽略 $I_{b3}R_2$ 电压降, 则输出高电平

$$v_O = V_{OH} = V_{CC} - (V_{be3} + V_{be4}) = 5V - 2 \times 0.7V = 3.6V$$

通常将输出高电平记为 V_{OH} 。

2. 输出低电平的工作状态

当输入端 A、B、C 全部为高电平(见图 2.3.2(b))时, T_1 集电结, T_2 和 T_5 的发射结导通, 因此, T_1 基极电位被钳定在 2.1V。这时, T_1 集电极电位为 1.4V, T_1 的 3 个发射结为反偏、而集电结为正偏, 所以, T_1 为“倒置”(即反向应用)状态, $\beta_R \ll 1$, 则有

$$I_{c1} = (1 + \beta_2)I_{b1} \approx I_{b1} \quad \text{和} \quad I_{b1} = \frac{V_{CC} - V_{b1}}{R}$$

I_{c1} 就是 T_2 的基极电流, 适当选择 R_2 、 R_3 和 β_2 , 即可使 T_2 处于饱和状态, 则有

$$V_{c2} = V_{be5} + V_{ces2} = 0.7\text{V} + 0.3\text{V} = 1\text{V}$$

T_3 发射极通过 R_4 接地, 基极电位大于开启电压, 所以, T_3 处于导通状态。 T_4 基极电位为 0.3V , 所以, 它为截止状态。

T_5 基极从 T_2 射极获得足够驱动电流, T_5 为浅饱和状态。这是由于有源泄放网络(T_6 和 R_3 、 R_6)的分流作用, 能够防止 T_5 进入深饱和状态。合理地选择 R_1 、 R_2 和有源泄放网络的设计参数, 可保证 T_5 处于浅饱和状态。输出 v_O 为低电平, 记为 V_{OL} 。

$$v_O = V_{OL} = V_{ces5} = 0.3\text{V}$$

电路中各晶体管的工作状态如表 2.3.1 所示。

表 2.3.1 晶体管工作状态表

管号 工作状态	T_1	T_2	T_3	T_4	T_5	T_6
输出高电平	深饱和	截止	浅饱和	放大	截止	截止
输出低电平	倒置工作	饱和	导通	截止	浅饱和	饱和

3. 推拉式输出结构

T_3 、 T_4 和 T_5 管构成推拉式输出电路。当输出为高电平时, T_5 截止, T_3 和 T_4 管组成的射极跟随器工作, 构成有源上拉电路, 其输出阻抗很低, 有较强驱动能力, 可给出 5mA 输出电流; 当输出为低电平时, T_4 截止, 而 T_5 饱和, 构成有源下拉电路, 其输出阻抗小于 100Ω , 可从输出端灌入 14mA 电流, 也有较强的驱动能力。

4. 有源泄放网络

浅饱和型 TTL 与非门具有工作速度快、抗干扰能力强等优点, 所以是目前使用较为广泛的 TTL 门器件。这主要是在设计上增加了有源泄放网络, 改善了多项性能指标的缘故。在早期的中、低速与非门电路中, 没有有源泄放网络, 只有一个泄放电阻(如图 2.3.1 中的 R_3 接地, R_6 和 T_6 去掉), 被称为基本 TTL 与非门, 其工作原理与浅饱和型 TTL 完全相同, 但是, 性能就比较差了。有源泄放网络可以提高工作速度, 改善电压传输特性(提高抗干扰能力)和温度特性^[1,3]。例如, 它可以缩短 T_5 管的开启时间 t_{on} 和关闭时间 t_{off} , 因而使工作速度大大提高。因为在输出由高电平至低电平(T_5 管由截止状态转换为饱和状态)的转换过程中, 由于 T_5 管基极直接与 T_2 管的发射极连接, 而 T_6 管的基极却通过 R_3 与它连接, 所以, T_5 管将会先导通, 因而 T_2 管的射极电流全部作为 T_5 管的驱动电流, 从而缩短了 T_5 管的开启时间 t_{on} ; 另一方面, 在 T_5 快速进入饱和之后, T_2 管的射极电流被 T_6 管分流, 合理选择 R_3 、 R_6 的阻值可控制分流大小, 使 T_5 管保持在浅饱和状态, 因此, 减少了 T_5 管基区的存

储电荷。在 T_3 管由饱和状态转换为截止状态过程中, 基区的存储电荷可以通过有源泄放网络迅速消散, 因而缩短了关闭时间 t_{off} 。显然, t_{on} 和 t_{off} 的缩短都有利于工作速度提高。

2.3.2 TTL 与非门的性能指标

若要合理地选用集成逻辑门, 就必须了解它的特性。下面以 TTL 与非门为例来介绍集成逻辑门的主要性能指标。

1. 高、低电平的标称值

TTL 逻辑门输出高电平的允许范围为 $2.4\sim 5V$, 其标称值为 $3.6V$ 。输出低电平的允许范围为 $0\sim 0.7V$, 其标称值为 $0.3V$ 。在 $0.7\sim 2.4V$ 之间的电平值, 在 TTL 电路中是不允许出现的, 若电平值处于这个范围, 就会造成逻辑混乱。

2. 电压传输特性及抗干扰容限

TTL 与非门的输出电压与输入电压之间的关系可以用电压传输特性来描述, 它表示输入由低电平向高电平变化时, 输出电平发生的相应变化, 如图 2.3.3 所示。

关门电平 V_{off} 保持电路输出端为高电平状态所允许的输入低电平的最大值。典型值为 $V_{off}=1.3V$ 。

开门电平 V_{on} 保持电路输出端为低电平状态所允许的输入高电平的最小值。典型值为 $V_{on}=1.5V$ 。

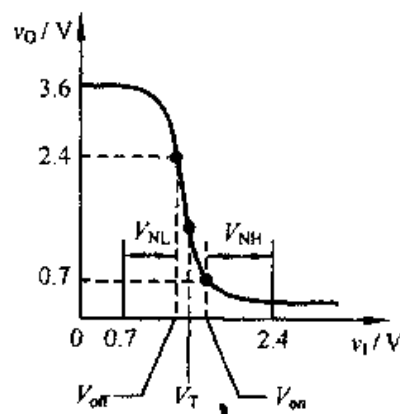


图 2.3.3 电压传输特性

在 V_{off} 至 V_{on} 这一段很窄的转折区内, 输出状态将发生急剧变化。通常将转折区的中点所对应的输入电压作为门电路的阈值电压 V_T , 一般 $V_T \approx 1.4V$ 。

在数字系统中, 门电路的输入端接前级门的输出端, 门电路的输出端接后级门的输入端。在使用环境下, 连线上往往存在干扰, 所以, 在门电路的电平上叠加有噪声。分析图 2.3.3 所示的电压传输曲线可知: 在考虑叠加噪声影响之后, 只要输入低电平不超过 V_{off} 或输入高电平不低于 V_{on} , 输出端能保持正常输出电平, 对后级门的正常工作就不会产生影响。门电路的这种能力称为抗干扰容限(或噪声容限)。

输入高电平时的抗干扰容限 V_{NH} 为

$$V_{NH} = V_{OH \min} - V_{on} = 2.4V - 1.5V = 0.9V$$

式中, $V_{OH \min}$ 为输出(指前级)高电平的最小值($2.4V$)。

输入低电平时的抗干扰容限 V_{NL} 为

$$V_{NL} = V_{off} - V_{OL \max} = 1.3V - 0.7V = 0.6V$$

式中, $V_{OL\max}$ 为输出(指前级)低电平的最大值(0.7V)。

3. 扇出系数 N_O 和扇入系数 N_I

扇出系数是一个门电路能够驱动同类型门电路的个数。门电路的驱动能力在输出高电平或低电平时是不同的。现以驱动同类型与非门的情况下来讨论。

(1) 输出低电平的工作状态

图 2.3.4(a)所示为输出低电平的工作状态, 图中左边为驱动门, 右边为负载门。由于驱动门输出为低电平, 所以, 负载门由电源 V_{CC} 通过 R_1 和 T_1 的发射结向驱动门的集电极灌入电流 I_{IL} , 这时负载电流是从外电路(负载门)流进驱动门, 所以称为灌电流负载。显然, 当负载门个数增加时, 总的灌电流便增加, 因而引起输出低电平的升高。所以负载门的数目取决于输出低电平时输出管 T_3 允许灌入的最大电流 $I_{OL\max}$, 即由下式确定扇出系数:

$$N_{OL} = \frac{I_{OL\max}(\text{驱动门})}{I_{IL}(\text{负载门})}$$

(2) 输出高电平的工作状态

图 2.3.4(b)所示的是驱动门输出高电平的工作状态。这时负载门的 T_1 工作在倒置状态, 接在负载门输出端上的每一个输入端都相当于反向偏置的二极管, 故有反向漏电流 I_{IH} , 这个电流是从驱动门拉出而流至负载门, 即负载电流是从与非门流向负载门, 所以称为拉电流负载。显然, 当负载门的个数增多时, 必将引起输出高电平降低, 但不能低于标准高电平的最低值, 所以负载门的数目取决于输出高电平时与非门的输出电流 I_{OH} , 即由下式确定扇出系数:

$$N_{OH} = \frac{I_{OH}(\text{驱动门})}{I_{IH}(\text{负载门})}$$

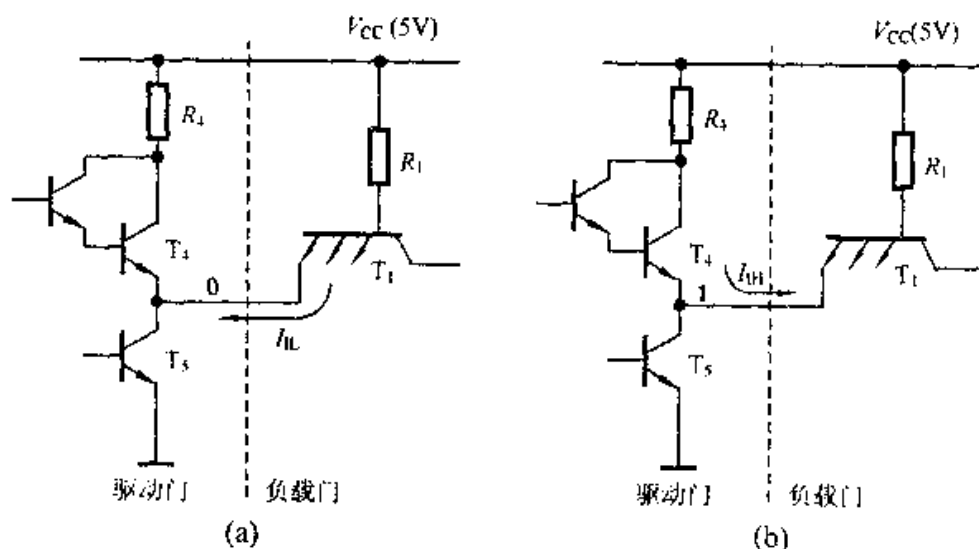


图 2.3.4 与非门的带负载能力

(a) 灌电流负载 (b) 拉电流负载

一般情况下, $N_{OL} \neq N_{OH}$ 。所以扇出系数应选用其中最小值。

扇入系数是门电路的输入端数, 它是由工厂制造时确定, 一般 $N_i \leq 5$ 。

4. 平均传输延时 t_{pd}

t_{pd} 是表示门电路工作速度的重要指标, 是指信号通过一级门电路的延迟时间。如图 2.3.5 所示, v_i 为输入信号, v_o 为反相输出信号。输出信号下降到 $V_m/2$ 相对于输入信号上升到 $V_m/2$ 之间的延迟时间称为导通延迟时间 t_{PHL} ; 输出信号上升到 $V_m/2$ 相对于输入信号下降到 $V_m/2$ 之间的延时称为截止延时 t_{PLH} 。一般情况下 $t_{PLH} > t_{PHL}$, 所以, 取二者平均值作为门电路的速度指标, 即平均传输延时 t_{pd} 为

$$t_{pd} = \frac{1}{2} (t_{PLH} + t_{PHL})$$

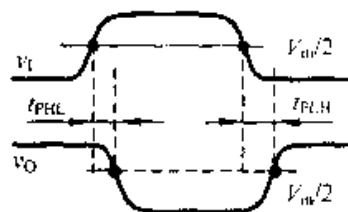


图 2.3.5 传输延时

TTL 门的平均传输延时为 3~40ns。

5. 平均功耗 \bar{P}

门电路的功耗是指在空载条件下工作时所消耗的功率。门电路输出低电平时的空载功耗称为空载导通功耗 P_L ; 门电路输出高电平时的空载功耗称为空载截止功耗 P_H 。因此, 平均功耗 \bar{P} 为

$$\bar{P} = \frac{1}{2} (P_L + P_H)$$

在实际估算电源容量及考虑门电路的散热要求时, 有两个问题必须注意: 其一、门电路在接负载后有负载电流加入, 平均功耗 \bar{P} 会有所增加; 其二、工作频率升高, 平均功耗 \bar{P} 将会增加。因为在输出由低电平向高电平转换的过程中, 有一阶段是 T_4 和 T_5 同时导通, 这时有一个很大的电流(几十毫安)通过 T_4 和 T_5 , 使总电流出现峰值, 瞬时功耗增加, 平均功耗也就增加。随着工作频率升高, 两种工作状态的转换次数增多, 峰值电流发生的次数增多, 平均功耗便会增加。

6. 速度-功耗积 M

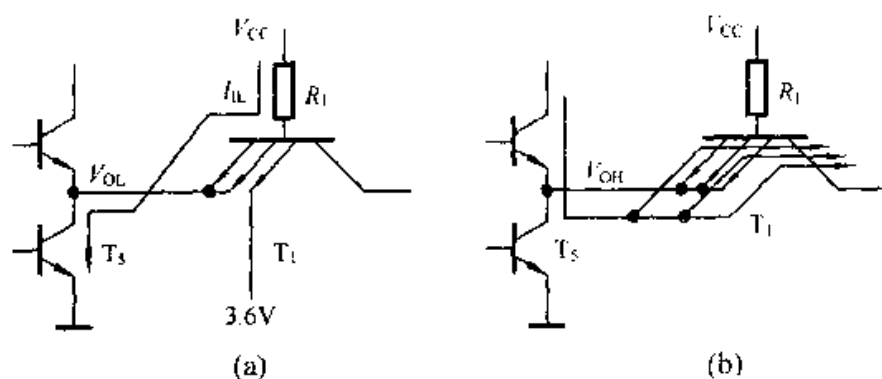
门电路的平均传输延时与空载导通功耗的乘积称为速度-功耗积, 即

$$M = t_{pd} P_L$$

速度-功耗积是衡量门电路性能的一个指标, 其值愈小愈好, 一般 TTL 门的速度-功耗积为几十皮焦耳($1\text{pJ} = 10^{-12}\text{J}$)。

7. 输入短路电流 I_{IL} 、输入漏电流 I_{IH}

输入短路电流 I_{IL} 是指与非门的一个或多个输入端接低电平, 而其他输入端接高电平(或开路)时流向低电平端的电流, 如图 2.3.6(a)所示。 I_{IL} 约为 1.4mA, 这个电流与接至低电平的输入端数无关。

图 2.3.6 I_{IL} 和 I_{IH} 示意图(a) 输入短路电流 I_{IL} (b) 反向漏电流

输入漏电流 I_{IH} 如图 2.3.6(b)所示。在图(b)中, 3 个输入端并接高电平, 输入管 T_1 是“倒置”状态(参见图 2.3.2(b)), 此时集电结导通, 发射结反偏。对每一个输入端来说, 它都有一个输入漏电流 I_{IH} (反向漏电流), 所以, 总的输入漏电流与接入端数有关, 每一端的反向漏电流为 $\beta_{RE} I_{BI}$ (β_{RE} 约为 0.05)。

2.3.3 其他系列 TTL 与非门

1. 肖特基 TTL 与非门(S 系列)

为了提高门电路的工作速度, 要求三极管不要工作在深饱和状态, 因为饱和愈深, 存储时间 t_s 愈长, 关闭时间 t_{off} 就愈长(参见图 2.2.2)。所以, 在工作于饱和状态的三极管的集电结上并接一只肖特基二极管 D_k , 如图 2.3.7(a)所示。

肖特基二极管又称为金属-半导体二极管, 它是借助于金属铝和 N 型硅的接触势垒而产生整流作用的。这种二极管的正向压降约为 0.3V, 这比硅二极管的正向压降小许多; 而且开关速度比一般开关二极管高一万倍。图 2.3.7(b)所示的是带有肖特基二极管的三极管符号, 图 2.3.8 所示的是肖特基 TTL 与非门电路图。

硅三极管饱和时, 集电结为正偏, 且饱和愈深, 正偏愈大。一般三极管深饱和时 V_{ces} 为 0.1V 左右, 集电结正偏压 V_{bc} 约为 0.6V, 这一电压足以使 D_k 导通, 将基极电流分流一部分, 使三极管处于浅饱和状态, 其 V_{bc} 被限制在 0.3V 左右。当三极管处于未饱和状态时, $V_{bc} < 0V$, D_k 截止, 对电路不会有影响。

图 2.3.8 所示电路中仅 T_4 没有加肖特基二极管, 这是因为无论与非门工作在哪一种状态, T_4 管都不会处于饱和状态。

肖特基 TTL 与非门的工作速度比普通与非门提高 3~4 倍, 平均传输延时约为 3ns。

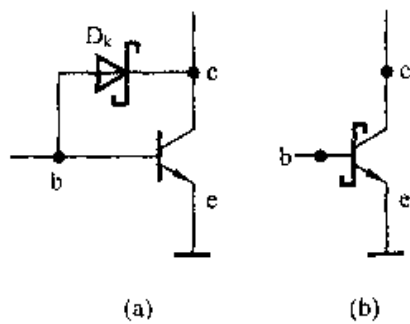


图 2.3.7 抗饱和晶体管

(a) 电路图

(b) 抗饱和晶体管符号

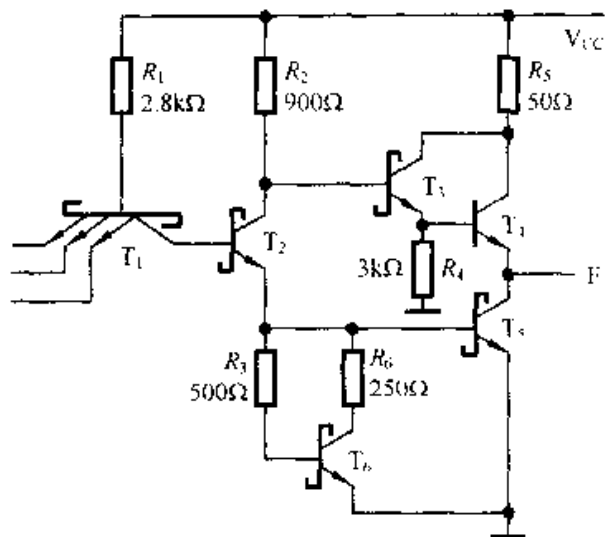


图 2.3.8 肖特基 TTL 与非门电路

2. 低功耗肖特基 TTL 与非门(LS 系列)

低功耗肖特基 TTL 与非门电路如图 2.3.9 所示。它与肖特基 TTL 与非门比较, $R_1 \sim R_5$ 的电阻值较大, 同时 R_4 由接地改接至输出端, 因而在 T_3 导通时, 减小了 R_4 上的功耗, 使 LS 系列的功耗仅为标准通用系列(54/74)的 1/5, 约为 2mW; 为了提高速度, 将多发射极三极管改为用肖特基二极管构成与门, 因为它无电荷存储效应。此外, 增加了 D_5 、 D_6 。当 T_4 由截止向饱和导通转换时, D_6 通过 T_1 集电极既加速了负载电容的放电, 又加大了对 T_4 基极的驱动电流, 同时 D_5 也通过 T_1 集电极为 T_3 的基极提供了泄放回路, 使 T_3 的截止加快。由于采用了上述措施, 使得 LS 系列在功耗大大降低的条件下, 工作速度仍可达到 54/74 系列水平。

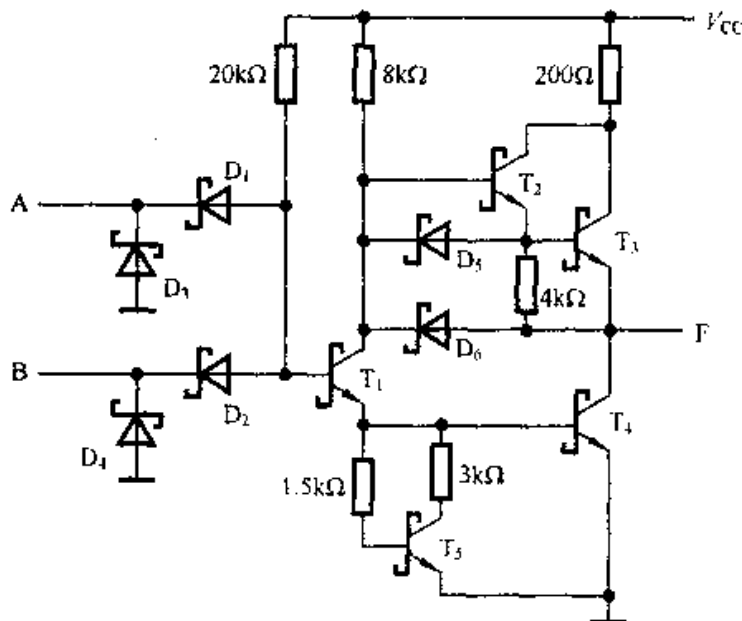


图 2.3.9 低功耗肖特基 TTL 与非门电路

表 2.3.2 列出了 4 种系列的主要性能指标。

表 2.3.2 TTL 系列器件主要性能比较表

性能指标	CT54/74	CT54H/74H	CT54S/74S	CT54LS/74LS
平均传输延迟/ns/门	10	6	3	9
平均功耗/mW/门	10	22	19	2
最高工作频率/MHz	35	50	125	45

2.3.4 集电极开路与非门(OC 门)

两个 TTL 门输出端直接相连, 如图 2.3.10(a)所示。当 F_1 和 F_2 中至少有一个为 0 时, F 为 0; 只有 F_1 和 F_2 均为 1 时, F 才为 1。所以 $F=F_1 \cdot F_2$, 即两个逻辑门输出端相连, 可以实现两个输出信号的相与功能, 称为线与。在用门电路设计各种逻辑电路时, 如果能将输出端直接并接, 以得到线与功能, 有时能使电路简化。

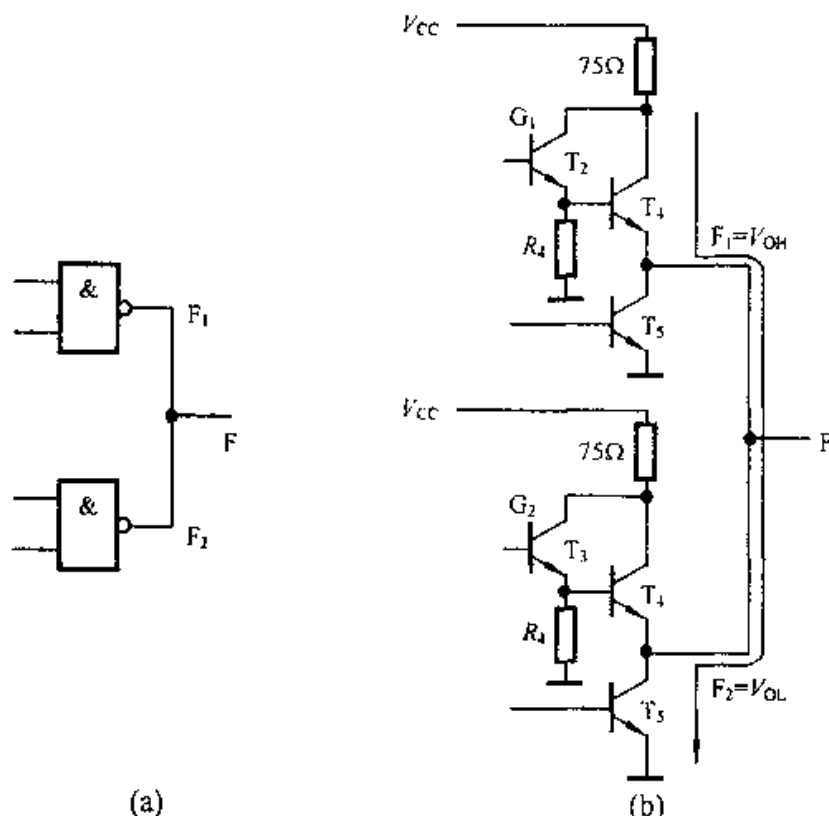


图 2.3.10 两个 TTL 与非门输出并联使用的情况

(a) 逻辑图 (b) 电路图

前面介绍的推拉式输出结构的 TTL 门是不能将两门输出端直接并接的, 在图 2.3.10(b)所示连接中, 若 F_1 输出高电平, F_2 输出低电平, 则会在 V_{CC} 与地之间形成

低阻通路(如箭头线所示),因而将会有有一个很大负载电流流过 G_2 门的 T_4 和 G_1 门的 T_3 , 这个电流远远超过了正常工作电流,会造成逻辑混乱,甚至损坏门电路。

为了使 TTL 门能实现线与功能,把输出级改为集电极开路结构,这种集电极开路的门电路,简称 OC 门。

1. OC 门的电路结构

图 2.3.11(a)是 OC 门的电路图。它与普通 TTL 门的不同之处在于取消了 T_3 和 T_4 构成的射随器,因此,集电极是开路的,没有集电极负载。使用时需要在开路的集电极与电源之间外接负载电阻 R_L ,电路才能实现与非功能,即输出 $F = \overline{ABC}$ 。图 2.3.11(b)为逻辑符号。

若干个 OC 门输出并联使用,只需要一个公用负载电阻 R_L ,如图 2.3.12 所示。当 OC_1 门输出低电平,其他 OC 门输出高电平时,全部负载电流通过 OC_1 门 T_3 ,故只要合理选择 R_L ,就可以使 OC_1 门 T_3 管处于饱和,并联输出 F 为低电平;若图 2.3.12 中所示的所有 OC 门输出均为低电平,则每个 OC 门 T_3 管均导通,通过每个 OC 门 T_3 管的电流只为负载电流的 $1/3$,所以饱和更深,更容易保证输出低电平。

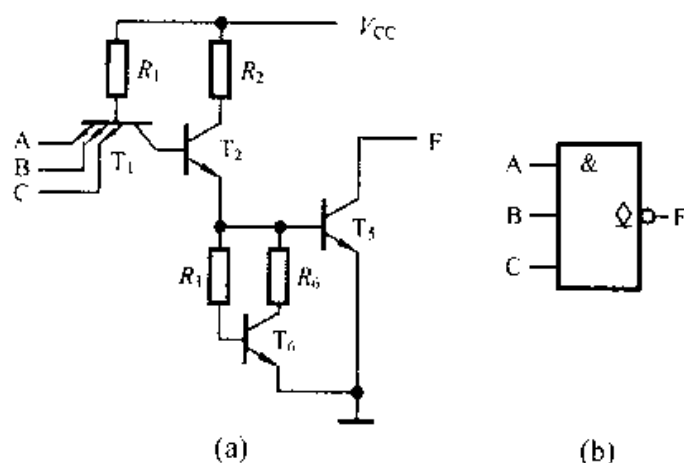


图 2.3.11 集电极开路与非门

(a) 电路图 (b) 逻辑符号

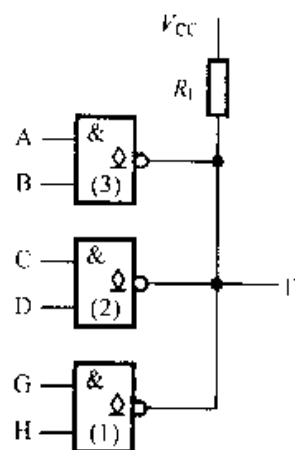


图 2.3.12 OC 门输出并联

当所有 OC 门输出高电平时,它们的 T_3 均截止,通过 R_L 的负载电流为集电极反向漏电流之和,其值很小,可以忽略不计,所以,输出高电平近似等于 V_{CC} 。由此可见:OC 门能方便地实现线与功能,即

$$F = F_1 \cdot F_2 \cdot F_3$$

2. 负载电阻 R_L 的计算

设有 n 个 OC 门并联,其输出端接 m 个 TTL 与非门,每个 TTL 与非门有 k 个输入端并联使用,如图 2.3.13 所示。

(1) OC 门输出为高电平

OC 门输出为高电平时,流过 R_L 的负载电流 I_{RL} 由下式确定:

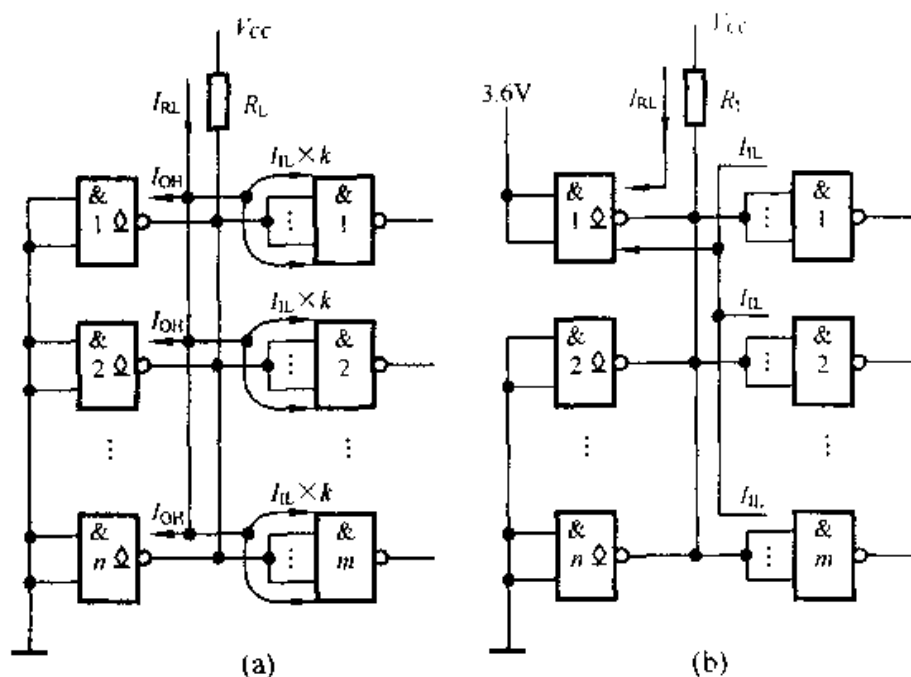


图 2.3.13 OC 门外接电阻的计算

(a) OC 门输出为高电平 (b) OC 门输出为低电平

$$I_{RL} = nI_{OH} + mkI_{IH}$$

式中, I_{OH} 为 OC 门输出管截止时的反向漏电流; I_{IH} 为与非门一个输入端的反向漏电流。

因为 $V_{OH} = V_{CC} - I_{RL}R_L$, 所以 R_L 增大会引起输出高电平下降。因此, 为了保证 OC 门输出高电平不低于 $V_{OH\min}$, 有

$$R_{L\max} = \frac{V_{CC} - V_{OH\min}}{nI_{OH} + mkI_{IH}} \quad (2.3.1)$$

(2) OC 门输出为低电平

假设只有一个 OC 门输出为低电平, 其余 OC 门输出均为高电平(这是最不利的情况), 则有

$$I_{OL} = I_{RL} + mI_{IL}$$

式中, I_{OL} 为 OC 门输出管饱和导通时的灌电流; I_{IL} 为 TTL 门输入管的输入短路电流。

因为 $V_{OL} = V_{CC} - I_{RL}R_L$, 所以 R_L 减小会引起输出低电平升高。为了保证 OC 门输出低电平不高于 $V_{OL\max}$, 则有

$$R_{L\min} = \frac{V_{CC} - V_{OL\max}}{I_{OL\max} - mI_{IL}} \quad (2.3.2)$$

因此, R_L 选定范围为

$$R_{L\min} \leq R_L \leq R_{L\max}$$

由式(2.3.1)和式(2.3.2)可以看出 TTL 门的输入短路电流 I_{IL} 与接入端数无关; 而 TTL 门的反向漏电流却与接入端数有关。

3. OC 门的应用

(1) 实现与或非逻辑

OC 门可方便地实现输入变量的与或非逻辑。例如, $F = \overline{AB + CD + GH}$, 只需 3 个 OC 门, 如图 2.3.12 所示。如果用普通 TTL 与非门来实现, 则有

$$F = \overline{\overline{ABCDGH}}$$

不仅会使电路结构复杂, 使用器件增多, 而且信号的延时也增加了。

(2) 电平转换

OC 门可以作为接口电路, 很方便地实现逻辑电平转换。例如, 实现 0.3~3.6V 逻辑电平至 0.3~12V 逻辑电平的转换, 只需将负载电阻接至 12V 电源便可, 如图 2.3.14(a)所示。

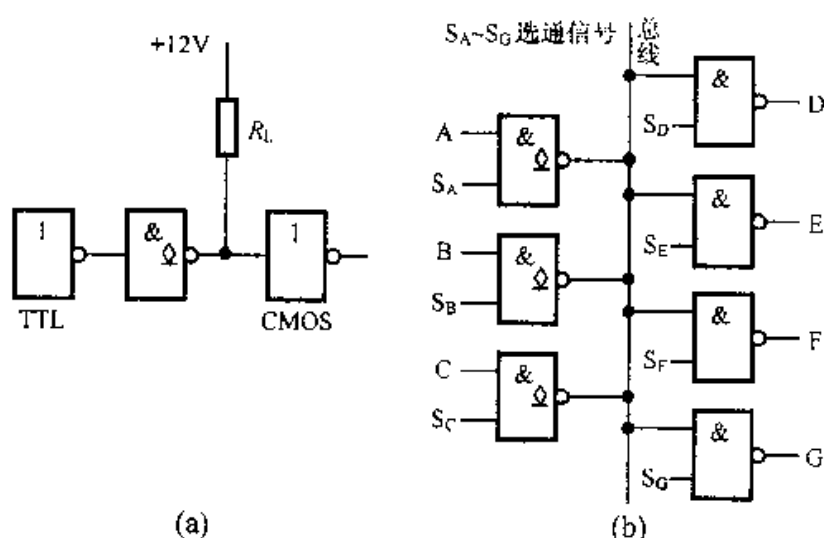


图 2.3.14 OC 门的应用

(a) 电平转换 (b) 数据采集

(3) 实现数据采集

在计算机或其他数字系统中, 经常使用一种母线(总线)结构, 来实现多种信号的接收和传送。为了使接收进来的信号传送到相应通道而不发生混乱, 广泛采用 OC 门来实现。如图 2.3.14(b)所示, 每个 OC 门有两个输入端, 一个作为数据端, 另一个作为控制端。可利用不同时刻到来的选通脉冲(正脉冲)来实现对不同通道的数据采集, 并汇总在母线上。同理, 选通脉冲可以将母线上的信号传送到不同的接收点, 这时只需用普通与非门即可。

2.3.5 三态 TTL 与非门(TSL)

三态 TTL 与非门又称为三态门(电路), 简称 TSL(Three State Logic)门。它的特点是输出端除了高、低电平之外, 还有第三种状态, 即高阻抗状态。

图 2.3.15 是三态门的典型电路, 它与 TTL 与非门电路的主要差别是输入级多了

一个使能端 E 和一个二极管 D。逻辑符号如图 2.3.16(a)所示。

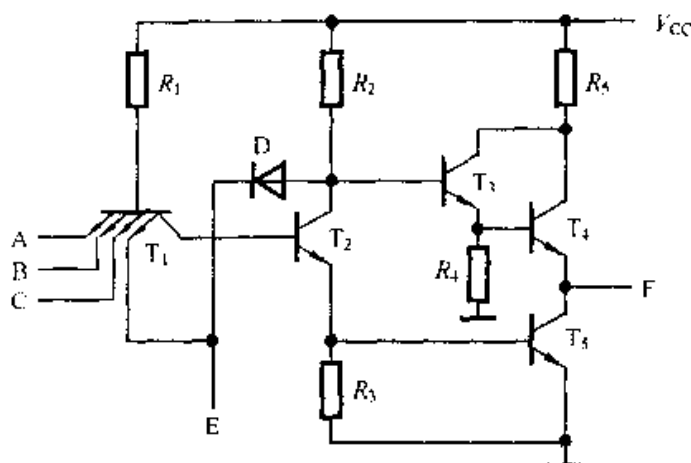


图 2.3.15 三态与非门

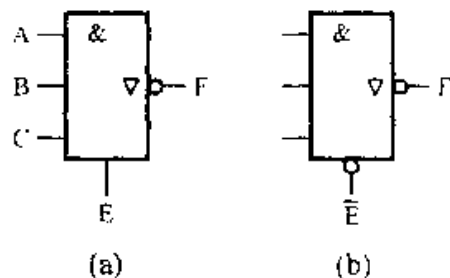


图 2.3.16 三态与非门逻辑符号

(a) E 高电平使能 (b) \bar{E} 低电平使能

当 $E=1$ 时，与非门能正常工作，称为工作状态，即 $F=\overline{ABC}$ ，故有时也称为使能状态。

当 $E=0$ 时，输出管 T_5 截止，同时二极管 D 导通将 V_{c2} 钳位在 1V 左右，使 T_4 截止。由于 T_4 和 T_5 都截止，故输出端呈现高阻抗，即高阻状态，有时也称为禁止状态。

三态门中也有控制信号 \bar{E} 为低电平时处于工作状态，而 \bar{E} 为高电平时处于禁止状态的产品，逻辑符号如图 2.3.16(b)所示。图中的小圆圈表示 \bar{E} 为低电平时使能。

利用三态门具有高阻状态这一特性，可以把多个三态门的输出端并联构成线与电路，如图 2.3.17 所示。但是，图中两个使能信号不能同时使能，否则，也会产生普通 TTL 门输出端并联使用的后果。当 \bar{E}_1 有效时 ($\bar{E}_1=0$ 和 $\bar{E}_2=1$)，三态门(2) 为高阻状态，三态门(1) 有信号输出，反之亦然。

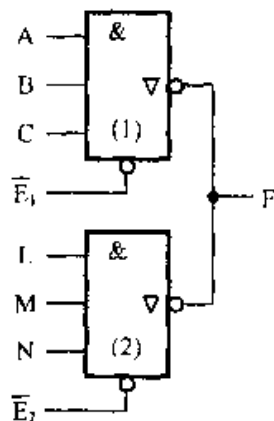


图 2.3.17 三态门输出并联

三态门与 OC 门相比，它在使用时不需外接负载和电源，在正常工作时输出阻抗低，故带负载能力比 OC 电路强，工作速度比 OC 电路高。因此，在中、大规模集成电路及母线结构中获得广泛应用。

2.4 射极耦合逻辑门(ECL)

由于 TTL 门中三极管工作于饱和或截止状态，所以，开通时间 t_{on} 和关闭时间

t_{off} 受到工作状态的限制, 工作速度难以进一步提高。最高速 TTL 门(S 系列及 AS 系列), 平均传输延时也在 2~3ns 之间。ECL 逻辑门采用非饱和型开关, 三极管工作在放大或截止两种状态, 因而, 其平均传输延时缩短到 1ns 以下。这是目前工作速度最高的一类集成电路。

2.4.1 ECL 反相器

ECL 门的基本电路是 ECL 反相器, 如图 2.4.1 所示。图中, V_R 为基准电压(-1.2V), 输入 v_i 加于 T_2 基极, 两管发射极接公用电阻 R_e , 其集电极接有相同电阻 R_c 。 v_i 输入低电平为 -1.6V, 输入高电平为 -0.8V。

1. 输入低电平

当 v_i 为低电平(-1.6V)时, T_1 导通, T_2 截止, v_e 为 -1.9V, 所以

$$I_c = \frac{v_e - V_{EE}}{R_e} = \frac{-1.9V - (-5V)}{600\Omega} \approx 5.2mA$$

$$v_{O1} = 0V - R_c I_{c1} = 0V - 0.14 \times 5.2V \approx -0.73V$$

$$v_{O2} = 0V$$

由于 R_c 较小, 所以 v_{ce1} 为 1.17V, 显然 T_1 工作于放大区。

2. 输入高电平

当 v_i 为高电平(-0.8V)时, T_2 导通, T_1 截止, $v_e = -1.5V$, 所以

$$I_c = \frac{v_e - V_{EE}}{R_e} = \frac{-1.5V - (-5V)}{600\Omega} \approx 5.8mA$$

$$v_{O1} = 0V$$

$$v_{O2} = 0V - R_c I_{c2} = 0V - 0.14 \times 5.8V \approx -0.8V$$

显然, $v_{ce2} = 0.7V$, 所以 T_2 是工作于放大区。

由上述分析可知, T_1 和 T_2 起电流开关作用, I_c 近似不变。两管中只有一管导通且工作在放大状态, 另一管则工作在截止状态, 所以, 两管都不会进入饱和区。因此, 基区不会有多余存储电荷, 存储时间 t_s 为 0s, 工作速度提高; 此外, 逻辑摆幅很小, 只有 0.8V, 所以, 分布电容的充放电时间缩短, 也有利于工作速度的提高。目前, 在这种反相器基础上构成的 ECL 门的平均传输延时可小于 1ns。

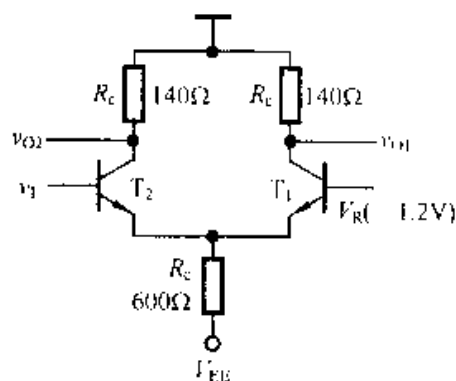


图 2.4.1 ECL 反相器

2.4.2 ECL 或/或非门

图 2.4.2(a)所示的是在 ECL 反相器基础上构成的 ECL 或/或非门, 逻辑符号示

于图 2.4.2(b)。

$T_2 \sim T_4$ 管并联构成或门：只要其中有一个为高电平输入，就会使 T_1 截止，输出电压 v_{O1} 为高电位，输出电压 v_{O2} 为低电位；只有在所有输入电压均为低电平时， T_1 才导通， T_2 截止。电压 v_{O1} 为低电平， v_{O2} 为高电平，所以

$$\begin{aligned} v_{O1} &= A+B+C \\ v_{O2} &= \overline{A+B+C} \end{aligned}$$

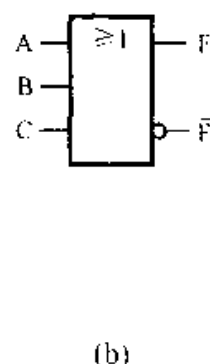
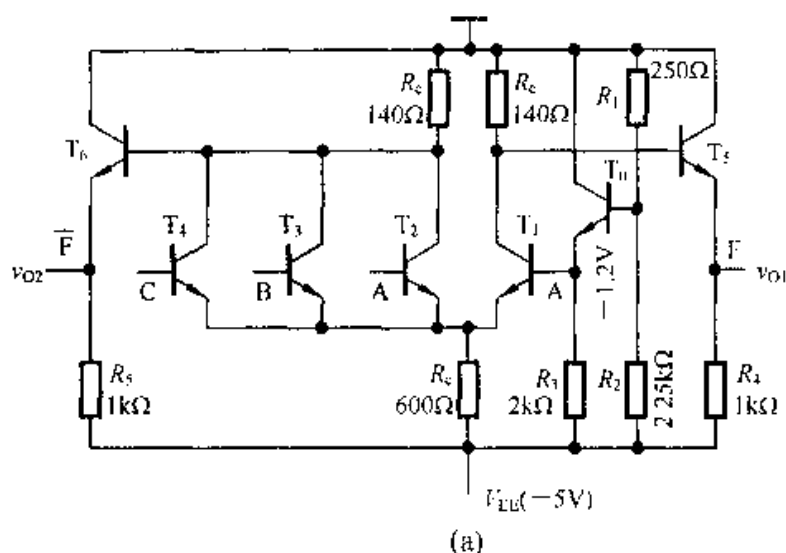


图 2.4.2 ECL 门

(a) 电路图 (b) 逻辑符号

T_0 管构成一个高稳定度的基准源， T_0 管为射极跟随器，输出阻抗低，使 T_1 基极的基准电压稳定。 T_5 和 T_6 管起电平转移作用，由于它们的发射结电压降 $0.7 \sim 0.8V$ ，所以，使输出 v_{O1} 和 v_{O2} 的高、低电平分别为 $-0.8V$ 和 $-1.6V$ ，便于各门之间的逻辑电平匹配。此外，由于 T_5 和 T_6 管的电流放大作用，增强了带负载能力， N_O 可多达 50 以上。

ECL 门的主要优点是工作速度高，负载能力强，电路设置有互补输出端，使用方便；其缺点是功耗大 $25 \sim 40mW$ ；逻辑摆幅小，因而抗干扰能力差 ($0.3V$ 左右)；高、低电平不能与 TTL 兼容。

2.5 MOS 逻辑门

TTL 和 ECL 逻辑门都是以双极型晶体管为基本元件构成的集成电路。本节将讨论以金属-氧化物-半导体场效应管(简称 MOS 管)为基本元件的集成逻辑门，简称 MOS 逻辑门。

2.5.1 MOS 管

MOS 管依据结构不同分为 PMOS(又称 P 沟道)管和 NMOS(N 沟道)管。每一类又依据其特性分为增强型和耗尽型两种。PMOS 管是早期产品,工作速度较低、电源电压高且为负电压,由它构成的逻辑门同 TTL 型逻辑门不能兼容。而 NMOS 管工作速度较高,电源电压为正电压,由它构成的逻辑门与 TTL 型逻辑门可以兼容,所以应用范围广泛。此外,由一对 PMOS 和 NMOS 管可以构成功耗特别小的 CMOS 管,由于其性能优越,应用范围也十分广泛,下面以 N 沟道增强型 MOS 管为例来介绍 MOS 管的结构和特点,然后介绍各类 MOS 管之间的差异。

1. NMOS 管

(1) N 沟道增强型 MOS 管

N 沟道增强型 MOS 管的结构如图 2.5.1 所示,它是以 P 型硅片作衬底、利用扩散方法在其上形成两个掺杂浓度很高的 N 区,分别称为源极 S 和漏极 D。在硅表面上有 SiO_2 绝缘层,在它之上再蒸发一层金属铝,引出线称为栅极 G。所以称为金属-氧化物-半导体场效应管,又称为绝缘栅场效管。

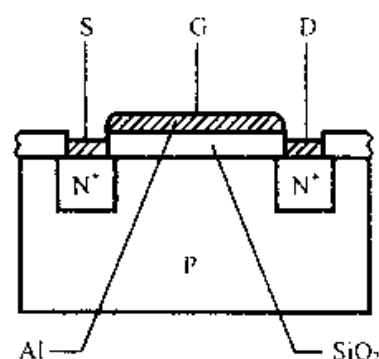


图 2.5.1 N 沟道增强型 MOS 管

在栅极未加正电压时,在源、漏极之间有两个方向相反的 PN 结,即使在源、漏极之间加电压 v_{DS} ,但由于总会有一个 PN 结是反向偏置,所以管子不会有电流,即它是截止状态, $i_{DS}=0A$ 。

当在栅极上加正电压 v_{GS} 时,在衬底表面将会感应产生电子(空穴被排斥)。但在 v_{GS} 较小时,感应的电子浓度小,在 S 与 D 极间还不能形成导电沟道。当 v_{GS} 增加到一定值时就会感应出足够电子,在衬底表面形成 N 型层(反型层),将源、漏极连接起来,即形成导电沟道(N 沟道)。此时,在源、漏极之间加电压 v_{DS} 便会产生电流 i_D ,即管子为导通状态。开始形成导电沟道的栅极电压称为开启电压 V_{TN} ,一般 $V_{TN}=1\sim 2V$ 。

(2) N 沟道耗尽型 MOS 管

这类 MOS 管在结构上与增强型类似,但在制造过程中,在 S 与 D 极间的衬底表面上已形成了 N 型沟道,因而栅压 v_{GS} 为 0V 时仍有导电沟道存在,在 S 与 D 极之间加电压便有 i_D 。

在栅极上加负压, N 型沟道变浅,当栅压负到一定值时, MOS 管就不能导通了,这时的栅压称为夹断电压 V_{IN} 。

2. PMOS 管

(1) P 沟道增强型 MOS 管

用N型硅片作衬底,在其上扩散两个P区,并在栅极上加负电压,则可形成P沟道增强型MOS管。一般它的开启电压 V_{TP} 为 $-2.5 \sim -1\text{V}$ 。在P型沟道中,载流子是空穴,由于空穴的迁移率约为电子迁移率的一半,所以,PMOS工作速度较NMOS管低。

(2) P沟道耗尽型MOS管

这类MOS管在制造过程中,在栅压为0V时也有P沟道形成,当栅极加正电压并达到一定数值时,P沟道被耗尽,该MOS管不能导通。这类MOS管制造工艺复杂,在集成电路中很少采用。

2.5.2 MOS管的静态特性

以N沟道增强型MOS管为例来讨论MOS管的主要特性。对于其他类型MOS管也有类似特性,仅具体参数的数值有差异。

1. 输出特性曲线

N沟道增强型MOS管的输出特性曲线如图2.5.2所示。输出特性曲线是表示在一定的 v_{GS} 下, i_{DS} 与 v_{DS} 之间的关系。

非饱和区 v_{DS} 值较小,在满足 $v_{DS} < v_{GS} - V_{TN}$ 时, i_{DS} 与 v_{DS} 呈近似线性关系。当 v_{GS} 增大,则形成的导电沟道愈宽,相应的等效电阻愈小,所以 i_{DS} 愈大;当 v_{GS} 一定时,沟道电阻也为定值,所以 v_{DS} 愈大,使电子流动加快, i_{DS} 便呈线性增加。非饱和区又称为可调电阻区。

饱和区 当 v_{DS} 增加到 $v_{DS} \leq v_{GS} - V_{TN}$ 之后,在离漏极 δ 处,沟道不能形成,即沟道在此处“夹断”了,如图2.5.3所示。 δ 范围形成空间电荷区,其中,电子是在电压作用下吸入漏极,形成 i_{DS} 。例如,当 $v_{GS}=5\text{V}$, $v_{DS}=5\text{V}$, $V_{TN}=2\text{V}$ 时,由于 $v_{GS} > V_{TN}$,所以在沟道中有 i_{DS} 流过,沿着沟道会产生电压降,氧化层两边的电压就不相等。

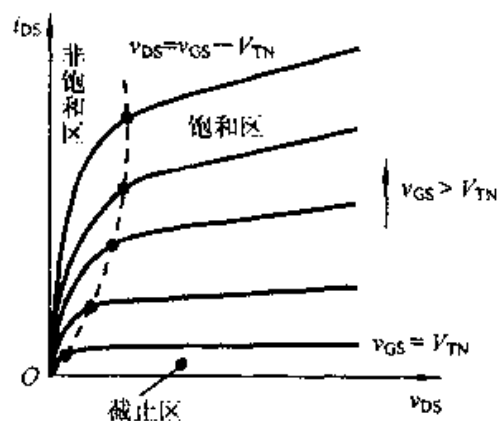


图 2.5.2 N沟道增强型MOS管特性曲线

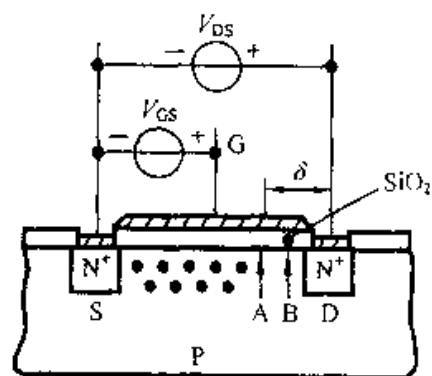


图 2.5.3 δ 示意图

上层电压均为 5V，下层电压离源极 S 愈远则愈大。设 A 点处电压 V_{AS} 为 3V，B 点处电压为 4V，则在 A、B 处的氧化层两边电压差分别为

$$V_{GA}=v_{GS}-V_{AS}=5V-3V=2V$$

$$V_{GB}=v_{GS}-V_{BS}=5V-4V=1V$$

所以，在 A 处沟道刚好形成，而在 B 处沟道不能形成，在 A 处以后的沟道(即 δ 范围内)均被“夹断”。由于 V_{AS} 是一定的， i_{DS} 基本上不变，这时，不管 v_{DS} 如何增加，只会使 δ 空间电荷区的电压降增加，对 i_{DS} 的影响很小；所以，在饱和区， i_{DS} 与 v_{DS} 近似无关。对应不同的 v_{GS} ，使电流趋于饱和的 v_{DS} 也不同。在输出特性曲线上，把满足 $v_{DS}=v_{GS}-V_{TN}$ 的临界点连接起来便构成非饱和区和饱和区的分界线，如图 2.5.2 中虚线所示，称之为临界线。

截止区 当 $v_{GS}<V_{TN}$ 时，没有导电沟道， $i_{DS}\approx 0$ 。

依据以上特性分析，MOS 管作为开关应用时，在开关信号作用下，交替工作在截止与导通状态。

2. 转移特性曲线

转移特性是指在漏源电压 v_{DS} 一定时， i_D 与栅极控制电压 v_{GS} 之间的关系，如图 2.5.4 所示。当 $v_{GS}<V_{TN}$ 时， $i_D=0A$ ；在 $v_{GS}>V_{TN}$ 后，在 v_{DS} 作用下才形成 i_{DS} 。

i_{DS} 与 v_{GS} 之间的关系可用跨导 g_m 来描述，其定义为

$$g_m=\left.\frac{\partial i_{DS}}{\partial v_{GS}}\right|_{v_{DS}=\text{常数}}$$

g_m 表达了 v_{GS} 对 i_{DS} 的控制能力。

3. 输入电阻和输入电容

在 MOS 电路中，输入信号通常是从栅源两极加入，因此，输入电阻实际上就是 SiO_2 绝缘层的绝缘电阻，一般在 $10^{10}\sim 10^{15}\Omega$ 范围，这样高的输入电阻使它具有许多特点。首先，它的输入电流很小，当一个 MOS 管驱动后面的 MOS 电路时，吸取电流很小，故静态负载能力很强。此外，由于输入电阻高，使栅极泄漏电流很小，这样，可以使信息在栅极电容上暂时储存一定时间，为构成动态 MOS 电路(参见 7.1 节)创造了条件。

输入电容是栅、源之间存在的寄生电容，一般数值约为 0.1 pF 至几 pF。

表 2.5.1 列出了 4 种 MOS 管的主要特性和逻辑符号。

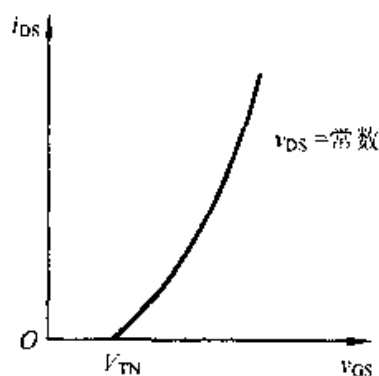
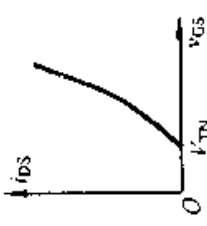
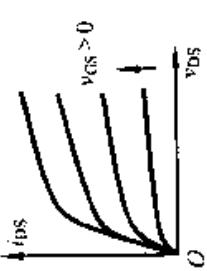
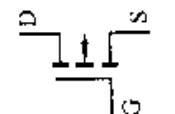
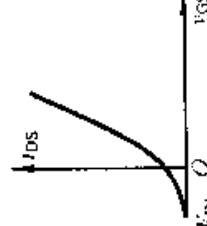
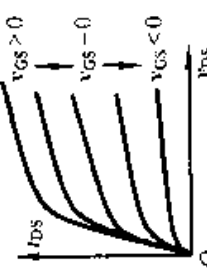
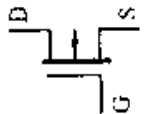
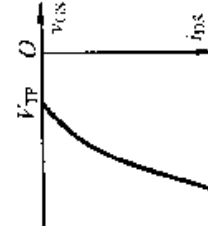
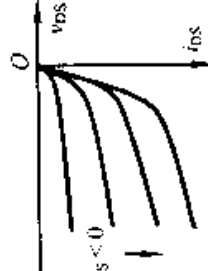
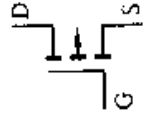
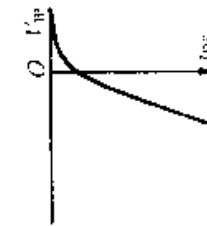
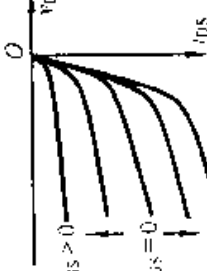
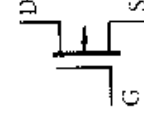


图 2.5.4 N 沟道增强型 MOS 管转移特性

表 2.5.1 MOS 管的特性和逻辑符号表

结构类型	栅源电压	漏源电压	阈值电压/V	转移特性曲线	输出特性曲线	逻辑符号
N 沟道 增强型 基片材料: P 型 漏源材料: N 型 导电沟道: 电子	正	正	开启电压 $V_{TN} > 0$			
N 沟道 耗尽型 基片材料: P 型 漏源材料: N 型 导电沟道: 电子	零、正、负	正	夹断电压 $V_{NK} < 0$			
P 沟道 增强型 基片材料: N 型 漏源材料: P 型 导电沟道: 空穴	负	负	开启电压 $V_{TP} < 0$			
P 沟道 耗尽型 基片材料: N 型 漏源材料: P 型 导电沟道: 空穴	零、正、负	负	夹断电压 $V_{KP} > 0$			

2.5.3 NMOS 逻辑门

1. NMOS 反相器

NMOS 反相器是构成 NMOS 门的基本单元。依据结构和负载特性可分为两类：增强型负载反相器(简称 E/E MOS)^①，耗尽型负载反相器(简称 E/D MOS)^②。

(1) 增强型负载反相器 E/E NMOS

E/E NMOS 反相器由负载管 T_L 和输入管 T_O 构成，两管均采用 N 沟道增强型 MOS 管，如图 2.5.5(a)所示。图中， T_L 管的 G、D 都与 V_{DD} 连接，因此， T_L 管总是满足 $v_{DSL} > v_{GSL} - V_{TL}$ 。式中， V_{TL} 为 T_L 管的开启电压。所以 T_L 管总是工作在饱和导通状态。

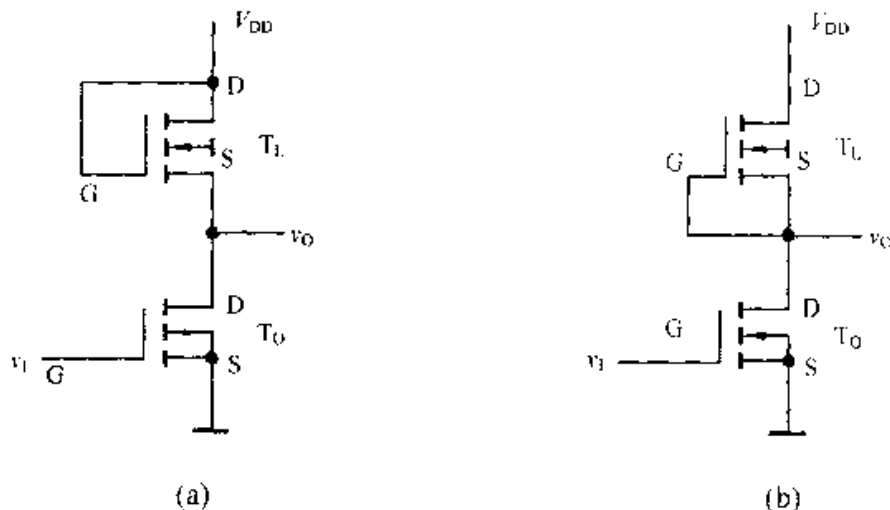


图 2.5.5 NMOS 反相器

(a) 增强型负载反相器 (b) 耗尽型负载反相器

当 $v_i=0$ 时， T_O 截止， T_L 饱和导通，这时流过 T_L 管只有很小的泄漏电流。输出高电平为

$$v_O = V_{OH} = V_{DD} - V_{TL}$$

当 $v_i=1$ 时， T_L 和 T_O 都导通，并且两管有同一电流流过，输出低电平。为了保证 $V_{OL} \leq 0.3V$ ，要求 T_L 的导通电阻 r_{DSL} 远大于 T_O 的导通电阻 r_{DSO} 。导通电阻与沟道的长宽比(L/W)成正比，所以要求

$$\frac{L_L}{W_L} \gg \frac{L_O}{W_O}$$

式中， L_L 和 W_L 为 T_L 的导电沟道的长和宽； L_O 和 W_O 为 T_O 的导电沟道的长和宽。

① E/E MOS 是英文 Enhancement/Enhancement Metal Oxide Semiconductor 的缩写。

② E/D MOS 是英文 Enhancement/Depletion Metal Oxide Semiconductor 的缩写。

这类电路称为有比型电路。

(2) 耗尽型负载反相器 E/D NMOS

E/D NMOS 电路如图 2.5.5(b)所示。输入管 T_O 为增强型，负载管 T_L 为耗尽型。在分析工作过程中应注意耗尽型 NMOS 管在 $v_{GSL}=0V$ 时仍有电流导通； T_L 和 T_O 的沟道尺寸(L 和 W)也应保持一定比例，以保证 $V_{OL} \leq 0.3V$ 。

当 $V_I=0$ 时， T_O 截止， T_L 饱和导通。但是，由于 T_O 管截止仅有很小的泄漏电流，故输出高电平

$$v_O = v_{OH} \approx V_{DD}$$

当 $V_I=1$ 时， T_L 和 T_O 都导通，并且两管有同一电流流过，输出的低电平由两管的导通电阻分压确定。所以，为了保证输出低电平 $V_{OL} \leq 0.3V$ ，也要求 $r_{DSL} >> r_{DSO}$ 。即 T_L 和 T_O 的沟道尺寸 (L 和 W) 也应保持一定比例，显然它也是有比型电路。

2. NMOS 逻辑门

E/E NMOS 和 E/D NMOS 逻辑门的电路结构和分析方法都是类似的，故只讨论 E/E NMOS 逻辑门。E/E NMOS 与非门、或非门及与或非门的电路分别如图 2.5.6(a)，(b)和(c)所示。

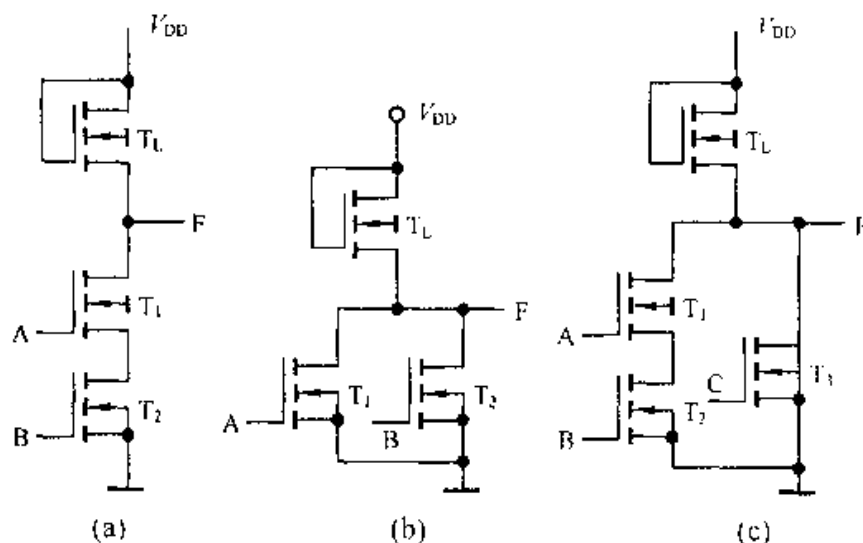


图 2.5.6 E/E MOS 逻辑门

(a) 与非门 (b) 或非门 (c) 与或非门

在图 2.5.6(a)所示电路中， T_L 为负载管， T_1 和 T_2 是两个串联的输入管(驱动管)。只要输入 A 、 B 中有一个为 0 电平，相应的输入管就截止，由 T_1 和 T_2 构成的串联通路断开，即输出 F 为 1 电平；只有输入 A 、 B 均为 1 电平时， T_1 和 T_2 才均导通，输出 F 为 0 电平。显然电路实现 $F = \overline{AB}$ 。

在图 2.5.6(b)所示电路中， T_1 和 T_2 是两个并联输入管。只要输入 A 、 B 之中有一个为 1 电平，相应输入管就导通，则输出 F 为 0 电平；只有输入 A 、 B 均为 0 电平，输出 F 才为 1 电平。显然，电路实现 $F = \overline{A+B}$ 。

图 2.5.6(c)所示电路实现 $F = \overline{AB + C}$ ，它不过是输入管串联及并联的综合应用，读者不难分析它的工作过程。

读者不难发现图 2.5.6(a)所示与非门电路，在变量数增加时，由于串联的输入管个数增加，在输出低电平时会导致低电平升高，使抗干扰能力下降，甚至不能工作。图 2.5.6(c)所示与或非门电路也存在类似问题。其改进措施将在 2.6 节中再讨论。

2.6 CMOS 逻辑门

CMOS 逻辑器件是继 TTL 器件之后所开发的第二种应用广泛的数字逻辑器件，由于 CMOS 工艺不断完善，它的工作速度逐渐接近 TTL 器件，而在功耗、抗干扰能力、带负载能力等指标上都远优于 TTL 器件，所以超大规模集成器件，如大容量的存储器、可编程逻辑器件 PLD、现场可编程门阵列 FPGA 等几乎都采用 CMOS 工艺制造。

国产的 CMOS 器件有：CC 4000 系列，同国际上的 CD 4000 系列及 MC 14000 系列相当；高速 54HC/74HC 系列，同国际上的 MC54HC/74HC 系列相当，它的平均传输延时可达到 $6 \sim 10\text{ns/门}$ ，其工作速度已与 TTL 器件的 74LS 系列相当。此外，还有与 TTL 器件兼容的高速 74HCT 系列和 74BCT 系列（BiCMOS）。

下面首先介绍 4000 系列的反相器和逻辑门，然后再介绍其他系列的 CMOS 逻辑门。

2.6.1 CMOS 反相器

CMOS 反相器电路如图 2.6.1 所示。它由一对 PMOS 管 T_{LP} 和 NMOS 管 T_{ON} 构成(均为增强型)。PMOS 管作为负载管，NMOS 管作为输入管。两管的栅极连接起来作为输入端；漏极连接起来作为输出端。 T_{LP} 管源极接电源 $V_{DD} = +5\text{V}$ ； T_{ON} 管源极接地。设 $V_{TN} = 2\text{V}$, $V_{TP} = -2.5\text{V}$ ； $V_{IL} = 0\text{V}$, $V_{IH} = 5\text{V}$ 。

当 v_i 为 0 电平时， T_{ON} 截止， T_{LP} 导通。 T_{LP} 管导通电阻 R_{OL} 为几百欧， T_{ON} 管截止电阻 R_{OFF} 为兆欧量级，所以输出 1 电平，即 $V_{OH} \approx V_{DD}$ 。

当 v_i 为 1 电平时， T_{ON} 导通，而 T_{LP} 截止。同上述分析一样，输出为 0 电平，即 $V_{OL} \approx 0\text{V}$ ，显然电路实现反相器功能。

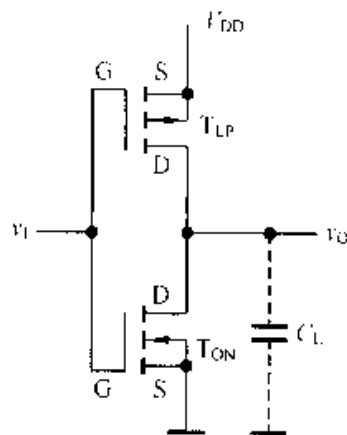


图 2.6.1 CMOS 反相器

在静态工作时, T_{LP} 和 T_{ON} 不会同时导通, 故是无比型电路, 在制造时可以使两管导电沟道完全一致。为了保证电路正常工作, 电源电压 V_{DD} 应大于 T_{LP} 和 T_{ON} 管的开启电压的绝对值之和, 即 $V_{DD} > |V_{TP}| + V_{TN}$ 。

2.6.2 CMOS 逻辑门

1. CMOS 与非门

图 2.6.2 所示的电路是二输入 CMOS 与非门电路, 每个输入端都连接到配对的 NMOS 管和 PMOS 管的栅极。其中, T_1 和 T_2 为两个串联的 NMOS 管, T_3 和 T_4 为两个并联的 PMOS 管。

当 A、B 两个输入信号中有一个为 0 时, 与该端相连的 NMOS 管截止, PMOS 管导通, 输出端 F 为 1; 只有当 A、B 均为 1 时, NMOS 管 T_1 和 T_2 均导通, PMOS 管 T_3 和 T_4 均截止, 输出 F 为 0。因此, 该电路具有与非功能:

$$F = \overline{AB}$$

2. CMOS 或非门

图 2.6.3 所示电路是 CMOS 或非门电路。两个 NMOS 管 T_1 和 T_2 并联, 两个 PMOS 管 T_3 和 T_4 串联。只要 A、B 两个输入信号中有一个为 1, 则与该端相连的 NMOS 管导通, PMOS 管截止, 输出 F 为 0; 只有 A、B 均为 0, 两个 NMOS 管 T_1 和 T_2 均不导通, 而两个 PMOS 管均导通, 输出 F 才为 1。该电路具有或非功能:

$$F = \overline{A + B}$$

上述电路虽然简单, 但存在一些缺点, 现以图 2.6.2 所示与非门电路为例来分析。

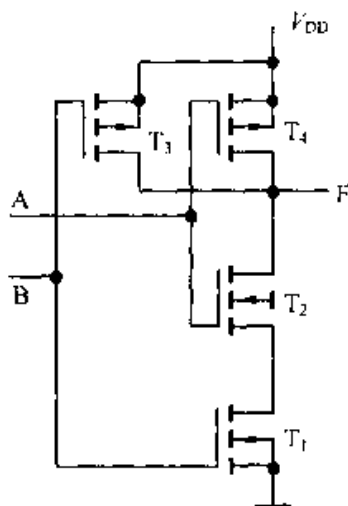


图 2.6.2 CMOS 与非门电路

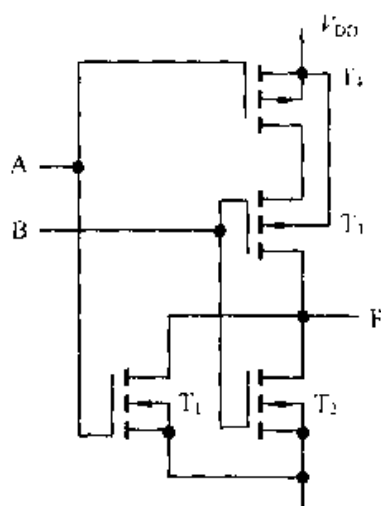


图 2.6.3 CMOS 或非门电路

由于与非门的输入管是串联的, 所以, 输入端数增多时会使输出低电平抬高, 因而使抗干扰能力下降。此外, 它的输出电阻 R_O 会因输入状态不同而有很大变化。

设每个管导通电阻均为 R_{ON} , 则

当 $A=0, B=0$ 时, T_3 和 T_4 导通, 所以 $R_O=R_{ON3}/R_{ON4}=R_{ON}/2$;

当 $A=1, B=1$ 时, T_1 和 T_2 导通, 所以 $R_O=R_{ON1}+R_{ON2}=2R_{ON}$;

当 $A=0, B=1$ 时, T_4 导通, 所以 $R_O=R_{ON4}=R_{ON}$;

当 $A=1, B=0$ 时, T_3 导通, 所以 $R_O=R_{ON3}=R_{ON}$ 。

可见, 由于输入状态不同会使输出电阻相差 4 倍。

为了克服上述缺点, 可以在每个输入端、输出端加一级反相器 (称为缓冲级) 构成实用的带缓冲级的与非门, 电路如图 2.6.4 所示。显然它是在或非门基础上构成 (因为 $F=\overline{AB}=\overline{\overline{A+B}}$)。

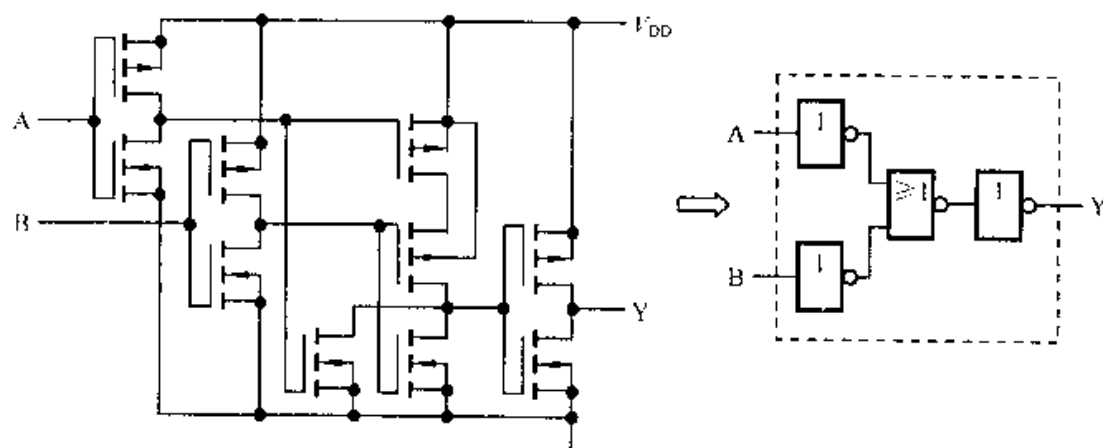


图 2.6.4 带缓冲级的 CMOS 与非门电路

3. CMOS 三态门

CMOS 三态门电路是在 CMOS 反相器的基础上加控制电路, 其电路结构有多种形式。其中一种较简单的电路如图 2.6.5 所示。它是在反相器的基础上增加一对 NMOS 管 T'_{ON} 和 PMOS 管 T'_{LP} 。当控制信号 $\overline{E}=1$ 时, T'_{ON} 和 T'_{LP} 均截止, 故输出呈现高阻抗; 当 $\overline{E}=0$ 时, T'_{ON} 和 T'_{LP} 同时导通, 反相器正常工作。

4. CMOS 传输门

将一对 NMOS 管和 PMOS 管并联可构成传输门电路, 如图 2.6.6(a)所示。图中, 两个 MOS 管均为增强型, NMOS 管 T_N 衬底接地, PMOS 管 T_P 衬底接 V_{DD} , 它们的栅极分别由一对互补电压控制。逻辑符号如图 2.6.6(b)所示。

当 C 端为 V_{DD} , \overline{C} 为 0V 时, 两管同时导通, 传输门为接通状态; 反之, 两管同时截止, 传输门为断开状态。由于 MOS 结构的对称性, CMOS 传输门的输入和输出端可以互换, 因此, 它是一个双向开关。在接通时, 导通电阻只有数百欧; 断开时, 只有很微小的漏电流, 两端电阻大于 $10^9\Omega$ 。所以是一种较为理想的开关器件, 应用领域十分广阔。例如, 在模拟电路中广泛应用的模拟开关, 它的用途是传输连续变化的模拟电压信号, 其基本电路就是由 CMOS 传输门和一个 CMOS 反相

器构成的,如图 2.6.7 所示。同 CMOS 传输门一样,它也是双向开关,输入和输出端可以互换。

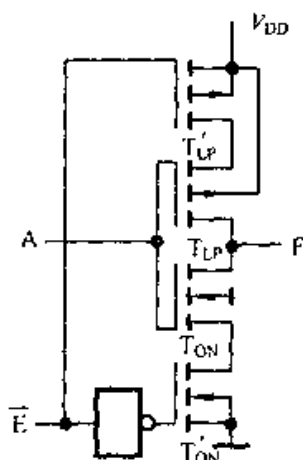


图 2.6.5 CMOS 三态门电路

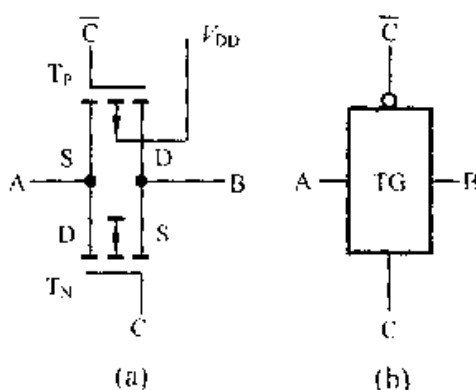


图 2.6.6 CMOS 传输门

(a) 电路图 (b) 逻辑符号

5. 漏极开路的 CMOS 门(OD 门)

CMOS 门电路的输出端不能并联使用,其原因与 TTL 门电路输出端不能并联使用相同。因此,CMOS 门的输出电路结构也可以做成漏极开路的形式,简称它为 OD 门。

图 2.6.8 所示的是 CC 40107 双 2 输入与非缓冲/驱动器的逻辑图。与非门和反相器均为 CMOS 电路,使用电源 V_{DD1} ; 输出电路是一只漏极开路的 NMOS 管,使用电源 V_{DD2} , 并且外接负载电阻 R_L 。

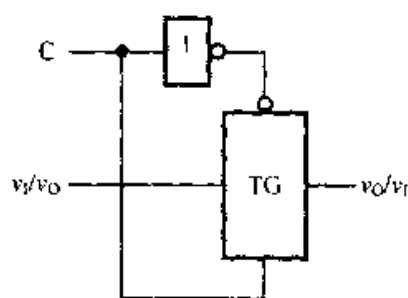


图 2.6.7 CMOS 双向模拟开关

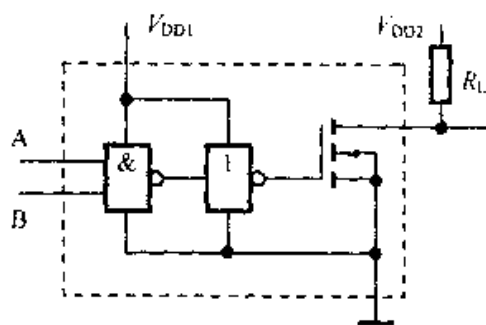


图 2.6.8 漏极开路输出门

电路的工作原理和它的应用同 OC 门类似,请读者自己分析。

*2.6.3 其他系列 CMOS 门

1. 高速 CMOS 门(54HC/74HC 系列)

早期生产的 4000 系列器件的工作速度较低,所以,应用范围受到一定限制。高速 CMOS 门主要是从制造工艺上进行改进,以便提高工作速度。

CMOS 门的工作速度不高的主要原因是 MOS 管中存在许多寄生电容。为了减小这些寄生电容,在制造工艺上采用短沟道(即尽可能地缩短沟道长度,缩小整个 MOS 管的尺寸)和硅栅自对准技术^[6]。这样制造的高速 CMOS 门,其平均传输延时小于 10ns,只是 CC4000 系列的 1/10。但其工作速度已与 TTL 门的 74LS 系列相当了。

74HCT 系列也是高速 CMOS 器件,但是,这类器件输出的高、低电平与 TTL 电路兼容。这为 74HCT 系列器件替代 74LS 系列器件提供了方便。

2. BiCMOS 门

BiCMOS 是双极型 CMOS (Bipolar CMOS) 的简称。这类门电路的特点是利用双极型器件速度快和 CMOS 器件功耗低的优点,将两种类型器件混合应用。电路中的逻辑部分采用 CMOS 器件,而输出级则采用双极型器件。

BiCMOS 反相器电路如图 2.6.9 所示。其中 T_1 和 T_4 为 PMOS 管、 T_2 和 T_3 为 NMOS 管; T_5 和 T_6 为双极型晶体管。

T_1 和 T_3 构成 CMOS 输入级反相器; T_5 和 T_6 构成推拉式输出级; 而 T_2 和 T_4 的作用分别是为 T_5 和 T_6 提供基区存储电荷的泄放网络。

当 v_i 为低电平时, T_1 、 T_4 和 T_5 导通, T_2 、 T_3 和 T_6 截止, 输出为高电平 V_{OH} 。这时由于 T_4 导通, 它为 T_6 的基区存储电荷提供了快速释放的通道, 缩短了 T_6 由导通转换到截止的时间, 这种作用与 TTL 与非门中的有源泄放网络类似 (参见图 2.3.1), 有利于提高开关速度。

当 v_i 为高电平时, T_2 、 T_3 和 T_6 导通, T_1 、 T_4 和 T_5 截止, 输出为低电平 V_{OL} 。这时 T_2 导通为 T_5 的基区存储电荷提供了快速释放通道, 缩短了 T_5 由导通转换到截止的时间, 提高了开关速度。

由于 T_5 和 T_6 的导通电阻都很小, 所以负载电容 C_L 的充、放电时间很短, 因而有效地减小了电路的传输延迟时间。

由上述分析可知, BiCMOS 反相器的平均传输延时将比其他系列 CMOS 的小得多, 目前 BiCMOS 的平均传输延迟时间已可达到 1ns。

图 2.6.10 所示电路是 BiCMOS 与非门的逻辑电路图。若 A、B 之中有一个为低电平, 则 T_8 导通、 T_9 截止, 输出高电平; 只有 A、B 同时为高电平, 才使 T_8 截止、 T_9 导通, 输出低电平。

图 2.6.11 所示电路是 BiCMOS 或非门的逻辑电路图。若 A、B 之中有一个为高电平, 则 T_1 和 T_4 的通路被切断, 并且 T_3 或 T_6 导通, 输出低电平, 这时 T_2 或 T_5

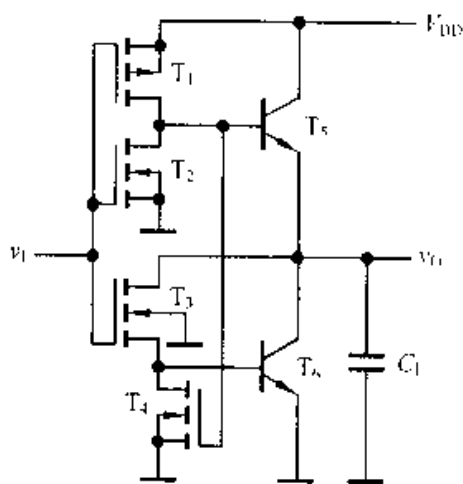


图 2.6.9 BiCMOS 反相器电路

导通, 为 T_8 提供存储电荷的释放通道; 只有 A、B 同时为低电平, T_1 和 T_4 均导通, 所以 T_8 导通, 而 T_3 和 T_6 均截止, 因此 T_9 截止, 输出高电平, 这时 T_7 导通, 为 T_9 提供存储电荷的释放通道。

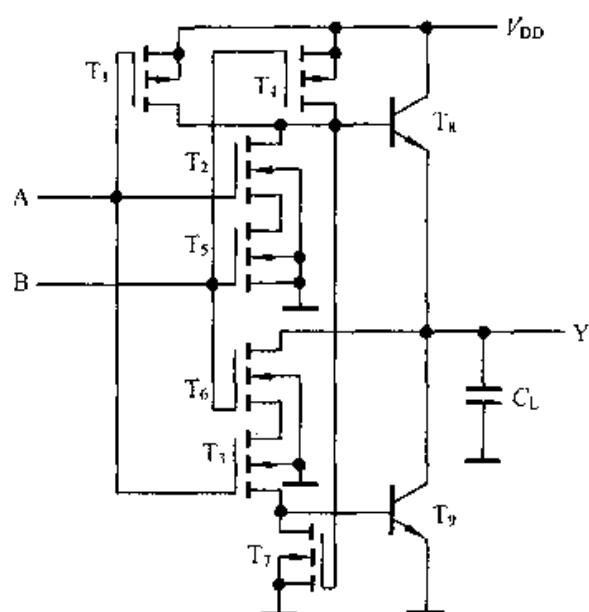


图 2.6.10 BiCMOS 与非门电路

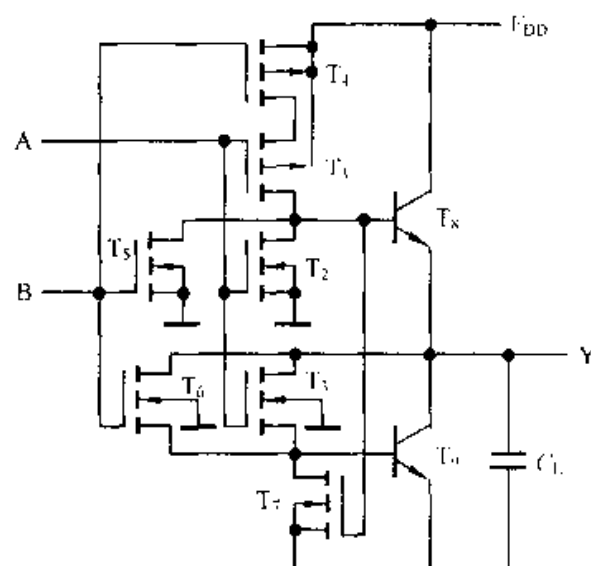


图 2.6.11 BiCMOS 或非门电路

2.6.4 CMOS 逻辑门的性能指标

1. 工作速度

从对 CMOS 反相器的工作原理分析(见图 2.6.1)可知: 在输出低电平时, T_{ON} 为导通状态, 为负载电容 C_L 提供了一个低阻放电回路, 使输出快速放电至低电平; 当输入由高电平跳变为低电平时, T_{LP} 为导通状态, 为 C_L 提供了一个低阻充电回路, 使输出快速充电至高电平(充电电流大)。所以工作速度高于其他类型的 MOS 门, 其平均传输延迟时间为几十 ns 至 100ns; 高速 CMOS 门电路, 由于采用了短沟道结构和硅栅自对准技术, 平均传输延迟时间可小于 10ns; BiCMOS 门电路, 由于利用了双极型晶体管速度快的优点和采用了有源泄放网络等措施, 使平均传输延迟时间可达到 1ns。

2. 电压传输特性及抗干扰容限

图 2.6.12 所示曲线是 CMOS 反相器的典型传输特性曲线。图中 $V_{DD}=10V$, $V_{TN}=|V_{TP}|=2V$ 。

当 $0 \leq v_I < V_{TN}$ 时, 即反相器工作于 AB 段, 由于 $v_I < V_{TN}$, T_2 截止(参见图 2.6.1), 而 $|v_{GS1}| > |V_{TP}|$, 所以 T_1 导通, 并且导通电阻很小。输出高电平。即 $v_O = V_{OH} \approx V_{DD}$ 。当 $V_{TN} \leq v_I < V_{DD} - |V_{TP}|$ 时, 即反相器工作于 BC 段。由于 $v_{GS2} > V_{TN}$, $v_{GS1} > |V_{TP}|$,

所以 T_1 和 T_2 同时导通。若 T_1 和 T_2 的参数完全对称, 则在 $v_i = V_{DD}/2$ 时两管的导通电阻相等, 故有 $v_o = V_{DD}/2$ 。这时工作在特性曲线转换区的中点, 两管处于转换状态, 所以, CMOS 反相器的阈值电压近似等于 $V_{DD}/2$ 。

当 $V_{DD} - |V_{TP}| \leq v_i \leq V_{DD}$ 时, 即反相器工作于 CD 段, 因为 $|v_{GS1}| < |V_{TP}|$, $v_{GS2} > V_{TN}$, 因而 T_1 截止, T_2 导通。所以 $v_o = V_{OL} \approx 0$ 。

从特性曲线上可以看出: CMOS 反相器的转换区非常狭窄, 由于阈值电压等于 $V_{DD}/2$, 输入低电平的抗干扰容限近似等于输入高电平的抗干扰容限, 即

$$V_{NL} = V_{NH} \approx V_{DD}/2$$

如果使用的电源电压升高, 则抗干扰能力增加, 可见其抗干扰能力远强于 TTL 电路。此外, 由于阈值电压随 V_{DD} 变化而变化, 所以允许 V_{DD} 在较宽范围内选用。一般 V_{DD} 允许范围为 $3 \sim 18V$ 。例如, 同 TTL 电路兼容的高速 CMOS 74HCT 系列, 采用 $5V$ 电源电压, 则输入低或高电平的抗干扰容限大约都是 $2.5V$ 。

3. 电源功耗

电源功耗包含静态功耗 P_S 和动态功耗 P_D 两部分。

静态功耗主要是输入保护电路中的保护二极管 (参见图 2.6.16) 和 CMOS 反相器中寄生二极管的反相漏电流所产生的。此外, 在静态下, 因为 T_{LP} 或 T_{ON} 总会有一个截止, 截止时的漏电流也会产生功耗, 不过这个漏电流极小, 所以, 由它产生的静态功耗可以忽略不计。静态功耗通常是以指定电源电压下的漏电流来表示, 例如, 规定国产 CC4000 系列 CMOS 反相器在常温 ($+25^\circ C$) 下的静态电源电流不超过 $1\mu A$ 。

动态功耗也包含两部分: 一部分是由于 T_{LP} 和 T_{ON} 在短时间内同时导通所产生的瞬时导通功耗 P_T ; 另一部分是负载电容 C_L 充、放电所消耗的功耗 P_C 。所以, 动态功耗 $P_D = P_T + P_C$ 。 P_T 和 P_C 主要取决于工作频率, 随工作频率升高而增加。

由以上分析可知: CMOS 门的静态功耗非常低 (微瓦级), 动态功耗随工作频率升高而增加, 在工作频率较高时, 静态功耗可忽略不计, 而动态功耗可达到毫瓦级, 其平均功耗最低。

4. 负载能力

在 MOS 电路中, 由于采用了绝缘栅场效应管, 其栅极输入阻抗很高 ($10^{10} \sim 10^{12} \Omega$), 所以, CMOS 门在带同类门作负载的情况下, 如果输出为高电平, 则驱动门为负载门提供的拉电流几乎可以忽略不计, 因此扇出数很大; 在输出低电平时, 负载门

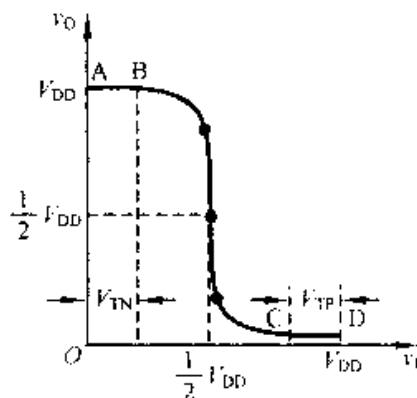


图 2.6.12 CMOS 反相器的电压传输特性曲线

的输入管处于截止状态，几乎不会向前级驱动门灌入电流，因此扇出数也很大。但是当负载门个数太多时，总的负载电容增加，会使电路的工作速度下降很多，以致不能满足速度指标。因此，扇出数主要取决于负载电容的大小和平均传输延时的要求，CMOS 的扇出数为 50。

由于 CMOS 门具有速度高、平均功耗低、抗干扰能力强和负载能力强等优点，所以，获得广泛应用。

2.6.5 CMOS 门使用中的几个问题

1. 输入保护电路和过流保护

因为 CMOS 的输入阻抗，栅极与衬底之间存在着以 SiO_2 为介质的输入电容，所以，在它上面极易产生很高的感应电压，致使绝缘层永久性击穿而损坏器件。

在目前生产的 CMOS 门中都已采用了各种形式的输入保护电路。

2. 输入电路的静电防护

由于 CMOS 输入保护电路所能承受的静电电压和脉冲功率都有一定限度，因此，在储存、运输、组装和调试过程中应注意下列问题：

(a) 闲置端不允许悬空，以免拾取脉冲干扰。

(b) 在储存和运输过程中采用金属屏蔽层进行包装，并且用套管或导线将所有输入脚短接。

(c) 在组装和调试时，所有仪器、工作台、和电烙铁必须可靠接地。焊接 CMOS 管时，最好先拔掉电源，利用余热进行快速焊接。

3. CMOS 门锁定效应的防护

由于在 CMOS 反相器的生产工艺中，将同一片 N 型衬底上同时制作 N 沟道和 P 沟道两种类型的 MOS 管，因此，在制造过程中就形成许多 NPN 型和 PNP 型寄生三极管。在一定的条件下，这些寄生三极管可能产生正反馈而导致电流无限增大，因而造成器件损坏。这种现象称为锁定效应。虽然生产商一直在研究从电路本身设计和制造工艺上克服锁定效应，但目前还不能保证所有的产品都不会发生锁定效应。为了防止锁定效应，可以采取以下防护措施。

(a) 在 CMOS 门的输入和输出端设置钳位二极管，以保证 v_i 和 v_o 工作在容许范围，如图 2.6.13 所示。

(b) 在电源输入端加去耦电路，如图 2.6.14 所示。去耦电路可以防止 CMOS 门的电源输入端 V_{DD} 出现瞬时高压。

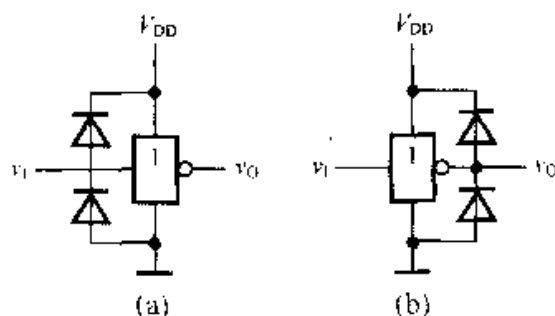


图 2.6.13 CMOS 电路的钳位保护电路
(a) 输入端的钳位电路 (b) 输出端的钳位电路

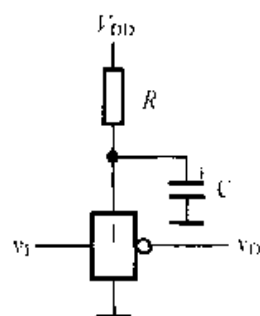


图 2.6.14 在 CMOS 电路电源上加去耦保护电路

(c) 当一个系统由几个电源分别供电时, 各电源的开、关顺序必须合理。启动时先接通 CMOS 电路的电源, 然后再接通输入信号和负载电路的电源; 关机时应先切断信号源和负载电路的电源, 然后再切断 CMOS 电路的电源。

*2.7 集成逻辑门使用中的几个问题

1. 多余输入端的处理

在实际应用中, 经常碰到逻辑门的输入端没有全部使用的情况, 对于未被使用的输入端(又称闲置端)要依据门电路的类型进行合理处理。

一种最简单的方法是将闲置端同使用的输入端并联使用, 这对于各种类型的门器件都可采用。但是, 这种并联使用的方法对某些类型的门器件会造成不良后果。例如, ECL 或非门, 并接一个输入端就会增加一个负载, 使实际有用的扇出数减少; 对 MOS 门而言, 每并接一个闲置端都会使负载电容增大(一般为几皮法), 这会使工作速度降低和功耗增加。

有的人将闲置端悬空或剪掉, 这对 TTL 门来说, 由于输入端是多发射极管的发射极, 输入阻抗较低, 悬空造成的影响不明显, 多数情况下可以正常工作; 但是, 对 MOS 门而言, 由于输入阻抗很高, 所以悬空输入端拾取的干扰会造成电路不能稳定工作。显然, ECL 门抗噪声能力差, 也不可悬空。

在实际应用中, 依据逻辑门的功能将闲置端接固定电平是经常采用的有效方法。例如, 将与门、与非门的闲置端接 1 电平, 而将或门, 或非门的闲置端接 0 电平。

2. 逻辑电平匹配

在一个较大的数字系统中, 为了使性能/价格比最佳, 往往会采用不同类型的器件, 所以, 这类器件之间就存在逻辑电平配合的问题。

TTL 门的高电平最小值为 2.4V, 低电平的最大值为 0.7V; ECL 门高电平为

-0.8V, 低电平为-1.6V。因此, 它们之间不能直接相连, 必须经过电平转换电路。CMOS 门在电源电压为 5V 时, 阈值电压为 2.5V; 而高、低电平分别为 5V 和 0V。因此, CMOS 门可以直接驱动 TTL 门; 但 TTL 门输出高电平最小值 2.4V 低于 CMOS 门的阈值电压 2.5V, 两者逻辑电平不匹配。最简单有效的方法是在连接处到电源(5V)之间接入一个几千欧的电阻, 将连接处的电平“上拉”到 5V, 如图 2.7.1(a)所示。图中, R_c 称为上拉电阻。

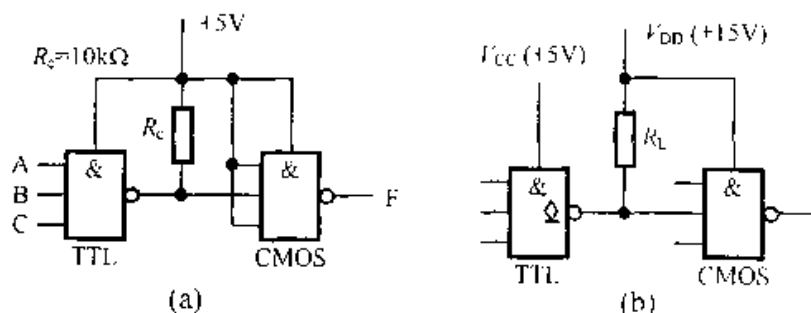


图 2.7.1 电平匹配
(a) 采用上拉电阻 (b) 采用 OC 门

在 TTL 门输出高电平时, TTL 门输出级 T_5 截止, 只有很小的漏电流; CMOS 门输入为高电平时的输入电流 I_{IH} 也很小 (若带 n 个 CMOS 门, 则为 nI_{IH})。因此, 它们流过 R_c 上产生的压降可以忽略, 输出高电平将被提升到 5V。

在 CMOS 门的电源电压较高时, 它所要求的输入高电平将超过 TTL 电路输出端能够承受的电压。例如, CMOS 门在使用 15V 电源电压时, 要求输入高电平大于 11V, 因此, TTL 门输出高电平必须大于 11V。这时, 可以采用 OC 门进行电平转换, 如图 2.7.1(b)所示。OC 门输出管的耐压可达 30V, 将输出端高电平提升到 15V 不会超过它的耐压范围。

生产商已生产了各种类型的电平转换电路供设计者选用, 读者可查阅器件手册。

3. 负载能力匹配

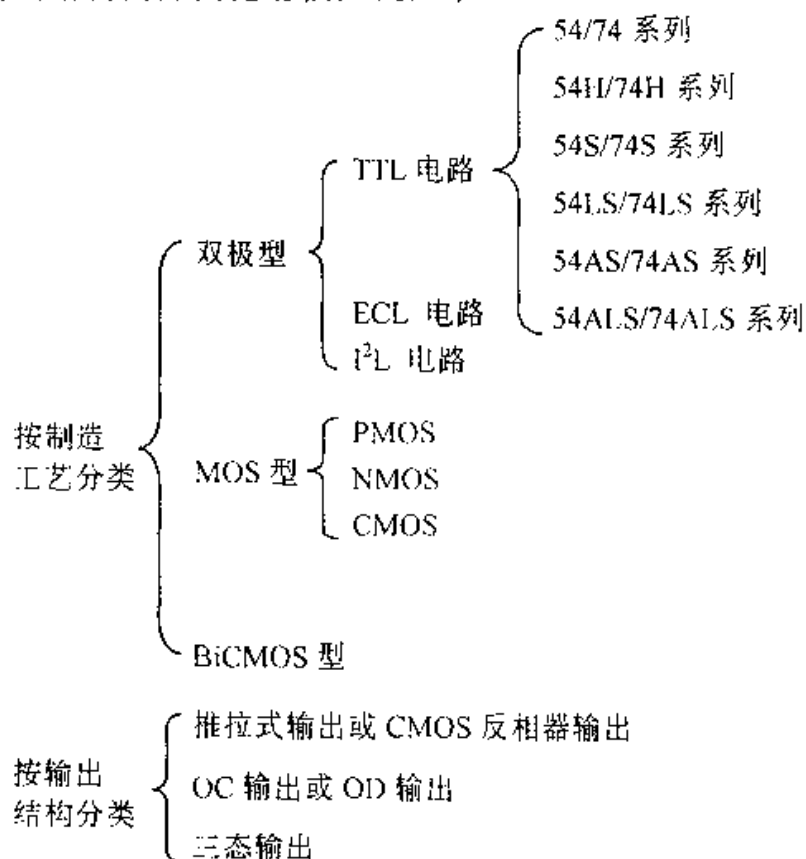
采用不同类型器件还必须考虑驱动能力是否匹配。在满足电平匹配条件下, TTL 门可以驱动 MOS 门; 但是, MOS 门驱动电流较小, 一般不能直接驱动 TTL 门。因此, 采用专门的接口器件是最有效的方法。例如, 在 CMOS 门输出端接入 CMOS 缓冲器, 其输出再驱动 TTL 门。还必须指出: 同类型但不同系列的门, 它们相互驱动负载的能力也不同, 有时差异甚大。例如, LS 系列 TTL 门只可驱动 4 个 S 系列 TTL 门; 而 S 系列 TTL 门可以驱动 50 个 LS 系列的 TTL 门^[3], 在不同类型或不同系列器件配合使用时就必须注意这个问题。

小 结

本章重点介绍了 TTL 逻辑门和 CMOS 逻辑门。对常用的 ECL 逻辑门、NMOS 逻辑门以及近年来新出现的 BiCMOS 逻辑门也作了适当的介绍。

在学习这些逻辑门时应把重点放在它们的外部特性上，即逻辑功能和电气特性。电气特性主要包括电压传输特性，输入及输出特性、负载能力及抗干扰能力，为此应重点了解电路结构的输入和输出两部分。它们可以帮助读者加深对外部特性的理解和运用。

逻辑门种类繁多，每种类型又分为许多系列。本章所介绍的逻辑门可以从制造工艺和输出结构两方面把它们归类如下：



为了对不同类型的各种系列器件有一个定性的比较，表 2.1 中列出了各种系列门电路的主要参数。顺便指出：由于不同厂家生产的同一类产品的性能指标相差很大，故表中参数仅供定性比较时参考，详细的参数及测试条件可查阅文献[27~30]。

TTL 门具有速度高、逻辑电平摆幅大、抗干扰能力和负载能力较强等优点，并且有中速、高速、超高速和低功耗等系列产品，目前仍然是应用最广泛的一类集成电路。

ECL 门具有最高工作速度的优点, 并且逻辑电平摆幅小, 无尖峰电流, 是一种低噪声器件, 主要应用于大型计算机和实时数据处理系统。

MOS 门中, CMOS 门在工作速度、功耗、负载能力以及电源利用系数等方面都优于其他 MOS 门, 尤其是 74HC CMOS 系列出现以后, 其平均传输延迟时间已可与 S-TTL 门媲美, 而速度功耗积这一综合指标明显优于其他各类型的器件, 所以, CMOS 门的应用范围将越来越广阔。

表 2.1 各种系列逻辑门的性能比较表

参数		电源电压/V	延迟时间 t_{pd}/ns	静态功耗 P/mW	速度-功耗积 M/pJ	逻辑摆幅/V
系列						
TTL	74	+5	10	10	100	3.3
	74H	+5	6	22	132	3.3
	74S	+5	3	20	60	3.3
	74LS	+5	9	2	18	3.3
ECL	CE10K	-5.2	2	25	50	0.8
	CE100K	-4.5	0.75	40	30	0.8
CMOS	4000	+5	75	0.002	0.15	5
	74HC	+5	10	1.5×10^{-3}	15×10^{-1}	5
	74HCT	+5	13	1×10^{-1}	13×10^{-1}	5
NMOS		+5	500	1	500	5
Bi CMOS		+5	2.9	$3 \times 10^{-4} \sim 7.5$	$8 \times 10^{-4} \sim 22$	5

思考题和习题

- 2.1 解释下列名词: 开门电平、关门电平、阈值电平、抗干扰容限。
- 2.2 试述 TTL 与非门、集电极开路与非门(OC 门)、三态与非门和 CMOS 与非门的特点。
- 2.3 TTL 门、MOS 门和 ECL 门的多余输入端应如何处理? 试举例说明之。
- 2.4 某电路如图 P2.1 所示。问调节电位器阻值对与非门输出是否产生影响, 为什么?
- 2.5 在下列 5 种不同情况下, 如果用内阻为 $20k\Omega/V$ 的万用表去测量 TTL 与非门的一个悬空输入端, 问测量的电压值为多少?
 - (1) 其他输入端均悬空;
 - (2) 其他输入端均接 1 电平;
 - (3) 其他输入端中有一个接地;
 - (4) 其他输入端均接地;
 - (5) 其他输入端接 0.3V。
- 2.6 试画出用 OC 门实现异或逻辑功能的逻辑图。

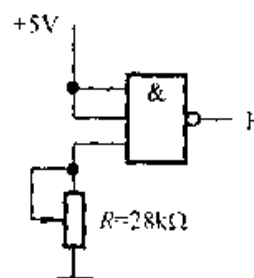


图 P2.1

2.7 电路及输入波形分别如图 P2.2(a)~(c)和(d)所示, 试画出 F_1 、 F_2 、 F_3 的输出波形。

2.8 某电路如图 P2.3 所示。图中, G_1 为三态门, G_2 为 TTL 门。 \bar{E} 控制端为低电平选通, 试分别求出 \bar{E} 为低电平及高电平两种情况下, 开关 K 在闭合及断开两种状态下的电压表的测量值为多少?

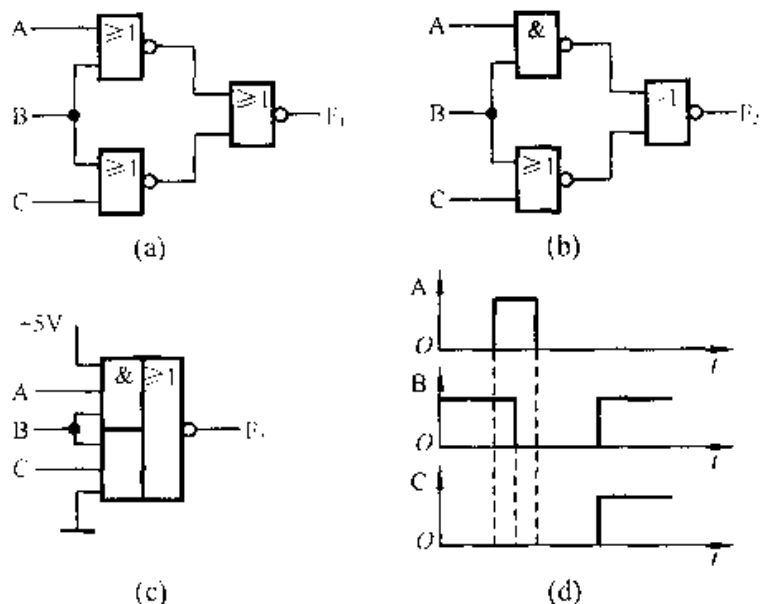


图 P2.2

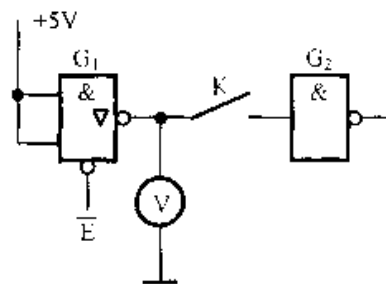


图 P2.3

2.9 图 P2.4 所示电路中各 MOS 管均为增强型 NMOS 管, 试求各管工作状态和各输出端的电压。

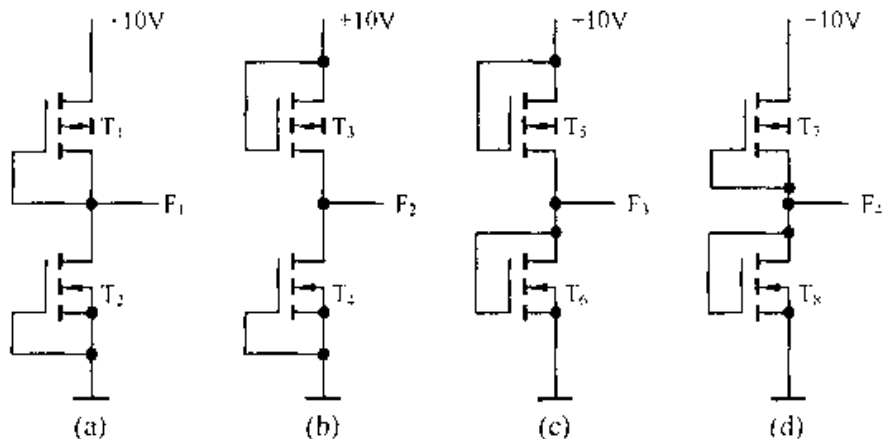


图 P2.4

2.10 简化图 P2.5 所示电路, 并画出用 NMOS 门构成简化后的逻辑图。

2.11 求图 P2.6 所示电路的输出函数 F 和 G 。

2.12 用 NMOS 门实现函数 $F = \overline{A(B+C)D + A + D}$ 。

2.13 用 NMOS 门实现下列逻辑函数(条件是仅提供原变量):

(1) $F_1 = AC + B$

(2) $F_2 = \overline{A + B \cdot C + D}$

2.14 试画出用 5 个 NMOS 管实现异或逻辑功能的逻辑图。

2.15 试用 CMOS 门实现逻辑函数 $F = \overline{AB} + C$ 。

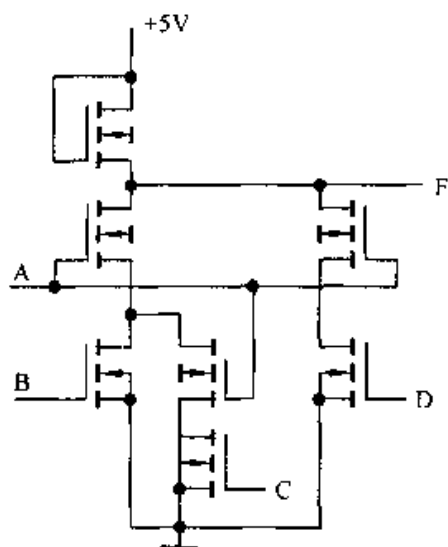


图 P2.5

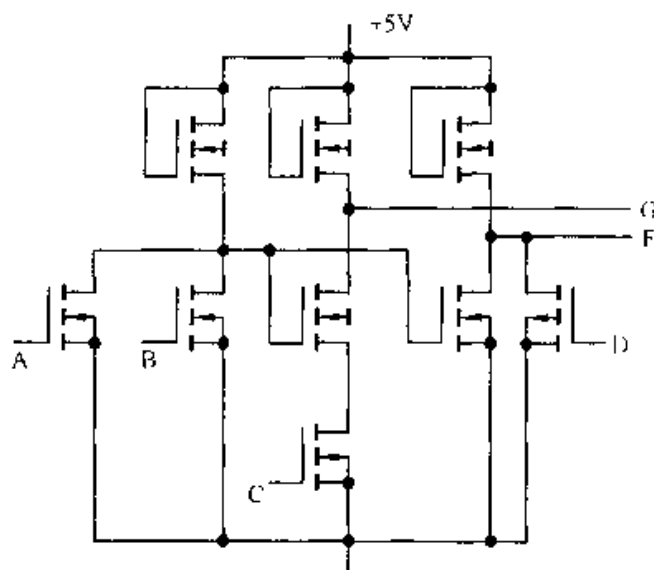


图 P2.6

2.16 设仅提供原变量和 3 只 NMOS 管, 请设计实现同或逻辑功能的逻辑图。

2.17 求图 P2.7(a)所示电路的输出函数 F 和 G。输入波形如图 P2.7(b)所示。

2.18 图 P2.8 所示电路中, 用两个接为线与形式的 OC 门, 它们的输出驱动 3 个 TTL 与非门, 接入的输入端分别为 1 个、2 个、3 个。设 OC 门输出低电平时允许灌入的最大电流为 14mA, 输出高电平时输出管(截止)的反向漏电流为 250 μ A; TTL 与非门的输入短路电流为 1.6mA, 一个输入端的反向漏电流为 50 μ A。试求外接负载电阻的取值范围。

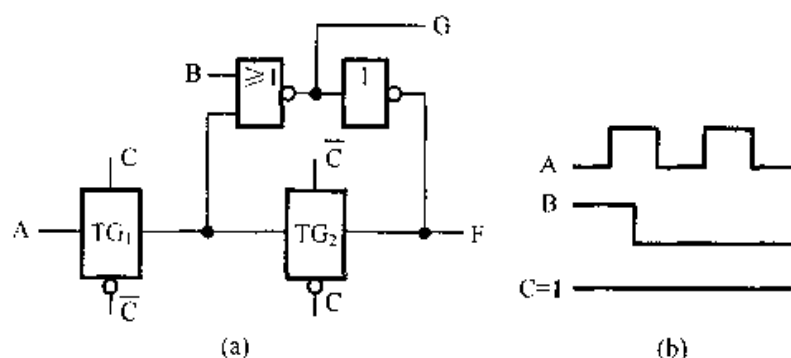


图 P2.7

(a) 电路图 (b) 波形图

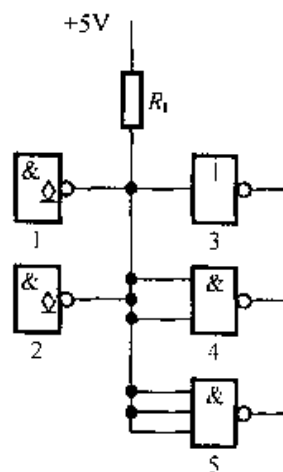


图 P2.8

2.19 试用 CMOS 门实现下面逻辑函数:

(1) $F = ABC$

(2) $F = A + B + C$

(3) $F = AB + C$

(4) $F = AB + CD$

第3章 组合逻辑电路

内容提要 本章讨论组合逻辑电路的设计方法和分析方法。对于通用性强、使用灵活的中规模组合逻辑部件，如数据选择器、4位全加器等，将重点讨论它们在逻辑设计中的应用。最后讨论组合逻辑电路中的冒险及其消除方法。

组合逻辑电路(简称组合电路)是由各种集成逻辑门按一定要求连接并实现某种逻辑功能的电路。这类电路的特点是，在任意指定时刻的稳态输出仅取决于该时刻的输入变量的状态，而与以前各时刻的输入状态无关。组合电路框图如图 3.0.1 所示，图中， X_1, \dots, X_n 是二值逻辑变量，输出信号 F_1, \dots, F_m 是二值逻辑函数。逻辑函数表达式为

$$\left. \begin{aligned} F_1 &= f_1(X_1, \dots, X_n) \\ &\vdots \\ F_m &= f_m(X_1, \dots, X_n) \end{aligned} \right\} \quad (3.0.1)$$

在电路结构上信号流向是单向性的，没有从输出端反馈到输入端的反馈回路；电路一般由逻辑门构成，不含有记忆元件；输出与输入之间存在一定的延迟时间。

在数字系统中，常用的组合逻辑部件有编码器、译码器、数据选择器、数据分配器、4位全加器、4位数值比较器和算术运算电路等。由于这些组合逻辑部件经常使用，所以我国已生产了配套的中规模组合逻辑部件，并且可以提供类型齐全(TTL 电路、CMOS 电路、ECL 电路)的芯片供用户选用。因此，直接使用中规模组合部件进行逻辑电路的设计是一种有效的方法，这是本章讨论的一个重点内容。

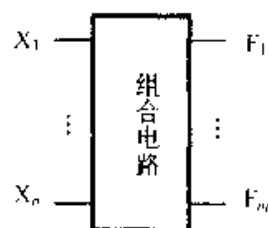


图 3.0.1 组合电路框图

3.1 组合逻辑电路的设计

组合电路的设计就是依据逻辑功能的要求，设计能实现该功能的简单而又可靠的最佳电路。设计组合电路的一般步骤为：

- 依据设计要求列出真值表；
- 写出最简逻辑函数表达式；
- 依据提供的器件类型，进行函数表达式的变换；

(d) 画逻辑图。

需要指出：采用上述设计方法(简称四步法)虽然可实现任何组合电路的设计，但在较多输入变量的情况下就非常繁琐，以致设计工作量非常大；对许多组合电路的设计可以不按照传统的四步法设计，而是利用某些逻辑函数的特点和逻辑门的特性，采用非常灵活的设计方法才能迅速而正确地设计出满足功能要求的最佳电路。由于这种方法不像四步法那样有规可循，所以称之为灵活设计法^[1,5]。

在组合电路设计中还应考虑下述几个问题：

(a) 提供输入变量的方式，一种是提供原、反变量，另一种是只提供原变量；

(b) 组合电路的信号传输时间要求，如允许传输时间较长，有些电路通过增加“级数”可以减少逻辑门的数量；

(c) 对多输出组合电路的设计需要应用多输出函数简化方法。

本节将结合这几个问题讨论组合电路的设计方法。

3.1.1 提供原、反变量输入条件的组合电路设计

在提供原、反变量输入条件下，依据四步法设计组合电路的最重要步骤是，将用文字描述的设计要求变换为真值表，由此便可简化函数表达式而获得最简的函数表达式，这类表达式只有与或式及或与式两种形式，进行简单的表达式变换，即两次求反便可得到与非-与非表达式及或非-或非表达式。

例 3.1.1 设计交通灯工作状态的检测电路。设交通灯由红、黄、绿 3 盏灯组成，正常工作状态下，任何时刻只有且仅有一盏灯点亮。其他情况都属于不正常工作状态，检测电路发出故障报警信号。

题意分析 在许多情况下，设计要求是用文字描述的一个具有一定因果关系的逻辑问题。所以，需要通过逻辑抽象的方法用一个逻辑函数来描述其因果关系。首先确定输入变量和输出函数。本例事件的结果用函数 F 表示。并规定 F 为 0，工作状态正常； F 为 1，检测电路发出故障警报信号。引起事件的因素定为输入变量，本例引起事件的因素是 3 盏灯，分别用变量 A 、 B 、 C 表示红、黄、绿 3 盏交通灯。并规定灯亮时为 1，不亮时为 0。这种以 0 和 1 两种状态分别代表输入变量和输出函数的两种状态称为逻辑状态赋值。需要指出：这里 0 和 1 的含意应由设计者人为规定（也完全可以反过来定义）。依据因果关系便可列真值表。

解 1 (1) 列真值表：设红、黄、绿 3 盏灯的状态分别用 A 、 B 、 C 表示，并规定灯亮为 1，不亮为 0；电路输出故障报警信号为 F ，并规定正常工作状态下为 0，故障状态下为 1。依据题意可列真值表 3.1.1。

(2) 写出最简逻辑函数表达式：由真值表作卡诺图如图 3.1.1 所示。

表 3.1.1 例 3.1.1 真值表

A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

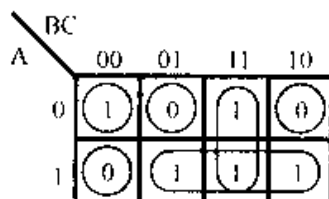


图 3.1.1 例 3.1.1 的卡诺图

由此可得

$$F = \overline{A}\overline{B}\overline{C} + BC + AC + AB \quad (3.1.1)$$

(3) 变换函数表达式:题意未要求门电路种类,故可以自行选择。本例选择与非门设计电路,则将函数 F 变为与非-与非表达式

$$F = \overline{\overline{A}\overline{B}\overline{C} + BC + AC + AB} = \overline{\overline{A}\overline{B}\overline{C}} \cdot \overline{BC} \cdot \overline{AC} \cdot \overline{AB}$$

(4) 画逻辑电路图,如图 3.1.2 所示。

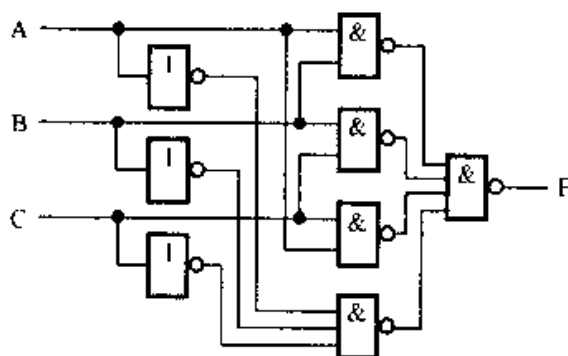


图 3.1.2 例 3.1.1 的逻辑图之一

如果题意要求用或非门实现,则可在卡诺图上圈 0 得到或与式,则有

$$\begin{aligned} F &= (A+B+\overline{C}) \cdot (A+\overline{B}+C) \cdot (\overline{A}+B+C) \\ &= \overline{\overline{A+B+\overline{C}} \cdot \overline{A+\overline{B}+C} \cdot \overline{\overline{A}+B+C}} \\ &= \overline{\overline{A+B+\overline{C}} + \overline{A+\overline{B}+C} + \overline{\overline{A}+B+C}} \end{aligned} \quad (3.1.2)$$

由上式可画出用或非门组成的逻辑电路图(请读者画出)。

如果题意要求用与或非门实现,则可在卡诺图上圈 0 得到 \overline{F} 的与或式,即有

$$F = \overline{\overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + A\overline{B}\overline{C}}$$

所以

$$F = \overline{\overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + A\overline{B}\overline{C}}$$

由上式画出用与或非门组成的逻辑电路图,如图 3.1.3 所示。

解 2 本例用灵活设计法来设计将会简便些。

假设输入变量与输出函数的确定和逻辑状态赋值同解1。

因为3盏灯都不亮或者3盏灯中有任何两盏灯同时亮就表示电路发生故障,所以,检测电路应发出报警信号。由此可直接写出下式:

$$F = \overline{A}\overline{B}\overline{C} + BC + AC + AB \quad (3.1.3)$$

因为F为1表示电路发生故障,所以 \overline{F} 为1则表示电路工作状态正常,它发生在3盏灯中任意一盏灯亮,其他两盏必须不亮时。由此可得

$$\overline{F} = A\overline{B}\overline{C} + \overline{A}B\overline{C} + \overline{A}\overline{B}C \quad (3.1.4)$$

显然,利用逻辑运算的特点及逻辑关系的分析直接写出的式(3.1.3)和式(3.1.4)分别与式(3.1.1)和式(3.1.2)完全相同,余下步骤同解1。

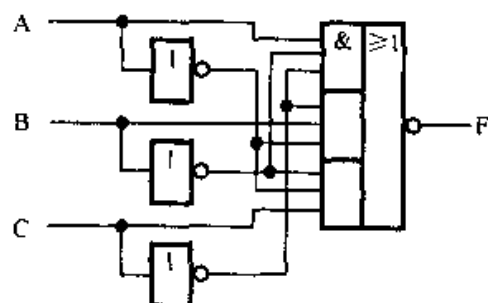


图 3.1.3 例 3.1.1 的逻辑图之二

表 3.1.2 例 3.1.2 的真值表

A	B	C	F_1	F_2
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

例 3.1.2 3个工厂由甲和乙两个变电站供电。若一个工厂用电,则由甲站供电;若两个工厂用电,则由乙站供电;若3个工厂用电,则由甲、乙两站同时供电。试设计一个供电控制电路。

解1 设3个工厂分别用A、B、C表示,并规定用电者为1,不用电者为0;甲和乙两变电站分别用 F_1 和 F_2 表示,并规定供电者为1,不供电者为0。

(1) 列真值表:如表3.1.2所示。

(2) 写出最简函数表达式:由真值表画出 F_1 和 F_2 的卡诺图,如图3.1.4(a)和(b)所示。

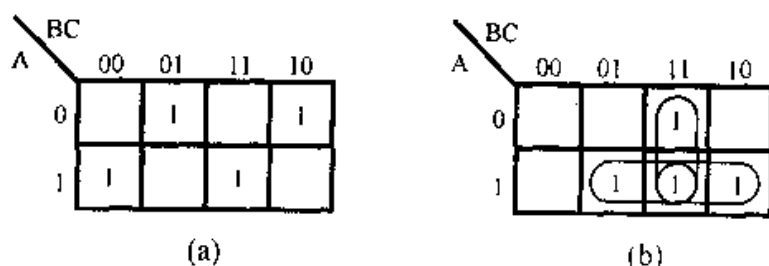


图 3.1.4 例 3.1.2 的卡诺图

由图可得

$$F_1 = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC = A \oplus B \oplus C \quad (3.1.5)$$

$$F_2 = AB + BC + AC \quad (3.1.6)$$

(3) 变换函数表达式:

$$F_2 = AB + BC + AC = \overline{\overline{AB} \cdot \overline{BC} \cdot \overline{AC}} \quad (3.1.7)$$

(4) 画逻辑电路图: 按式 (3.1.5) 和式 (3.1.7) 可画出由与非门和异或门组成的供电控制电路, 如图 3.1.5 所示。

解 2 用灵活设计法设计。因为甲站供电的条件是 3 个工厂中任一工厂用电或者 3 个工厂都用电, 即 A、B、C 中有一个为 1 或者 3 个为 1 则 F_1 为 1, 所以得

$$F_1 = A \oplus B \oplus C$$

因为乙站供电的条件是 3 个工厂中任意两个工厂用电, 即 A、B、C 中任意两个为 1 则 F_2 为 1。所以得到

$$F_2 = AB + BC + AC$$

显然两种设计方法写出的逻辑函数表达式完全相同, 所以逻辑电路图也完全一样。

例 3.1.3 设两个两位二进制数为 $A = A_1A_0$, $B = B_1B_0$ 。试用与非门设计一个比较这两个二进制数大小的逻辑电路。要求 $A > B$ 时, 输出为 1; $A \leq B$ 时, 输出为 0。

解 (1) 列真值表: 如表 3.1.3 所示。

(2) 写出最简逻辑函数表达式: 由真值表作卡诺图如图 3.1.6(a)所示, 由此可得

$$F = A_1\overline{B}_1 + A_1A_0\overline{B}_0 + A_0\overline{B}_1\overline{B}_0$$

(3) 变换函数表达式: 将函数 F 变为与非-与非表达式, 即

$$F = \overline{\overline{A_1\overline{B}_1} \cdot \overline{A_1A_0\overline{B}_0} \cdot \overline{A_0\overline{B}_1\overline{B}_0}}$$

(4) 画逻辑电路图: 如图 3.1.6(b)所示。

例 3.1.4 用或非门设计组合电路, 实现函数 $F = \sum m(0, 2, 3, 5, 8, 9, 10, 11)$ 。

解 (1) 写出最简函数表达式: 作 F 的卡诺图, 圈 0 (见图 3.1.7(a)) 得最简或与式

$$F = (\overline{A} + \overline{B})(\overline{B} + D)(\overline{B} + \overline{C})(A + B + C + \overline{D}) \quad (3.1.8)$$

(2) 变换函数表达式:

$$F = \overline{\overline{\overline{A} + \overline{B}} \cdot \overline{\overline{B} + D} \cdot \overline{\overline{B} + \overline{C}} \cdot \overline{A + B + C + \overline{D}}} \quad (3.1.9)$$

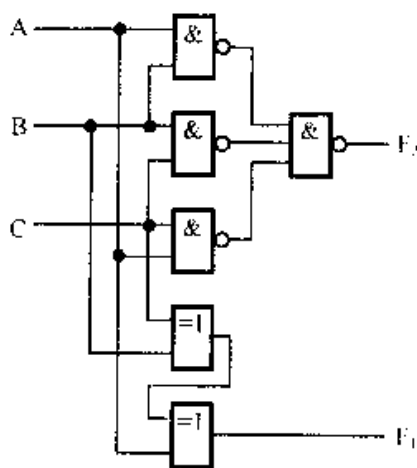


图 3.1.5 例 3.1.2 的逻辑电路图

表 3.1.3 例 3.1.3 真值表

A_1	A_0	B_1	B_0	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

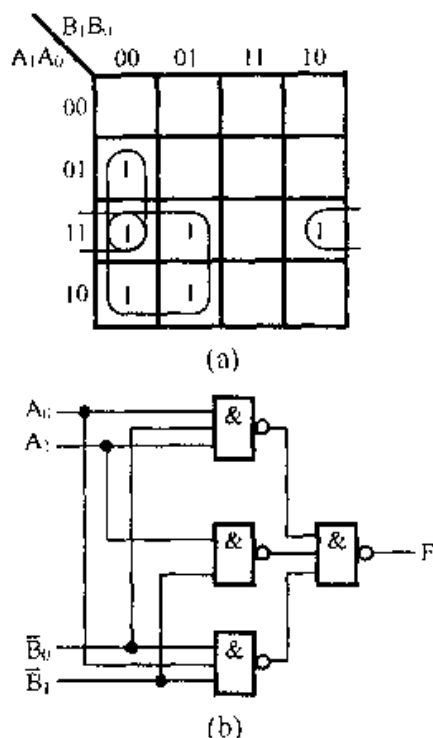


图 3.1.6 与非结构的组合电路

(a) 卡诺图 (b) 逻辑电路图

或者利用分配律将式(3.1.8)中含有公因子 \bar{B} 的 3 个或项合并, 即

$$\begin{aligned}
 F &= (\bar{B} + ACD) \cdot (A + B + C + \bar{D}) \\
 &= (\bar{B} + A + C + \bar{D})(A + B + C + \bar{D}) \\
 &= \bar{B} + A + C + \bar{D} + A + B + C + \bar{D}
 \end{aligned} \quad (3.1.10)$$

(3) 画逻辑电路图: 由式(3.1.9)画逻辑电路图, 如图 3.1.7(b)所示, 电路为两级或非门结构; 由式(3.1.10)画逻辑电路图, 如图 3.1.7(c)所示, 电路为三级或非门结构。它比图 3.1.7(b)电路省器件, 连线也简化了, 但信号传输时间较长。

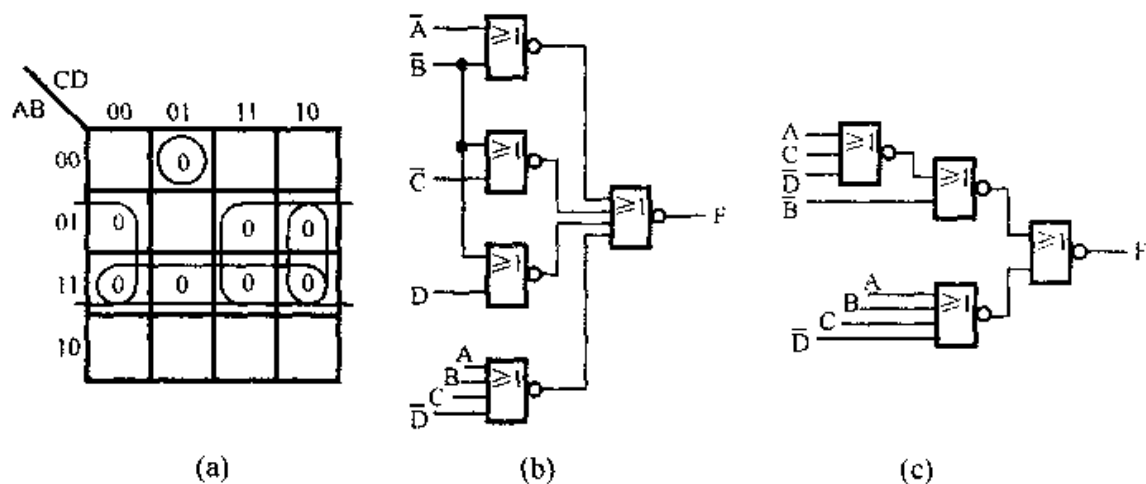


图 3.1.7 或非结构的组合电路

(a) 卡诺图 (b) 两级或非门结构的逻辑电路图 (c) 三级或非门结构的逻辑电路图

3.1.2 提供原变量输入条件的组合电路设计

在组合电路的输入端不提供反变量的条件下,若采用非门将原变量取反来提供反变量,则会使用较多器件,所以,要讨论无反变量输入条件下的设计方法。在无反变量输入条件下设计组合电路的关键是寻找无反变量输入条件下的最简函数表达式。可以利用一些公式对逻辑函数进行变换,使之不含有或少含有反变量,通常采用并项法或代替因子法。并项法是利用公式 $A\bar{B} + A\bar{C} = A\bar{B}\bar{C}$ 使函数简化;代替因子法是利用公式 $A\bar{B} + \bar{A}B = A\bar{A}B + B\bar{A}B$ 使函数简化。对复杂函数则综合应用这两种方法。

例 3.1.5 设输入无反变量,试设计组合电路实现函数 $F = A\bar{B} + \bar{A}BC + A\bar{C}$ 。

解

$$\begin{aligned} F &= A\bar{B} + \bar{A}BC + A\bar{C} = A(\bar{B} + \bar{C}) + \bar{A}BC \\ &= A\bar{B}\bar{C} + \bar{A}BC = A\bar{A}BC + BC\bar{A}BC \\ &= \overline{\overline{A}ABC} \cdot \overline{\overline{BC}ABC} \end{aligned} \quad (3.1.11)$$

逻辑电路如图 3.1.8 所示。

例 3.1.6 设输入无反变量,试设计组合电路实现函数 $F = A\bar{B}\bar{C} + B\bar{C}\bar{D} + A\bar{C}\bar{D} + \bar{B}\bar{C}\bar{D}$ 。

解

$$\begin{aligned} F &= A\bar{B}\bar{C} + B\bar{C}\bar{D} + A\bar{C}\bar{D} + \bar{B}\bar{C}\bar{D} \\ &= A\bar{C}(\bar{B} + \bar{D}) + BC(\bar{B} + \bar{D}) + CD(\bar{B} + \bar{D}) \\ &= A\bar{C}\bar{B}\bar{D} + BC\bar{B}\bar{D} + CDB\bar{D} \\ &= \overline{\overline{A}C\bar{B}\bar{D}} \cdot \overline{\overline{BC}B\bar{D}} \cdot \overline{\overline{CDB}D} \end{aligned} \quad (3.1.12)$$

逻辑电路如图 3.1.9 所示。在输入无反变量的条件下,需要加反相器提供所需要的反变量。

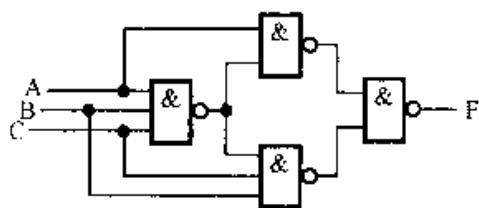


图 3.1.8 例 3.1.5 逻辑电路图

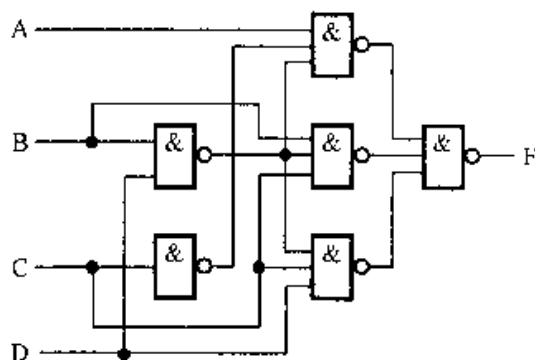


图 3.1.9 例 3.1.6 逻辑电路图

*3.1.3 多端输出组合电路的设计

一个具有相同输入变量而具有多个输出函数的组合电路称为多端输出组合电路。实用的许多组合电路都是属于这类电路,如编码器、代码转换电路、显示译码

器等。多端输出组合电路设计的关键是对多输出函数的简化。关于多输出函数的简化问题已在第1章中讨论了。

例 3.1.7 用与非门设计一个组合逻辑电路, 实现下列多输出函数:

$$F_1 = \sum m(2, 3, 5, 7, 8, 9, 10, 11, 13, 15)$$

$$F_2 = \sum m(2, 3, 5, 6, 7, 10, 11, 14, 15)$$

$$F_3 = \sum m(6, 7, 8, 9, 13, 14, 15)$$

解 用乘积卡诺图法简化多输出函数, 以便求出一组最佳的函数。

(1) 画出单个函数的卡诺图, 如图 3.1.10(a)~(c)所示。

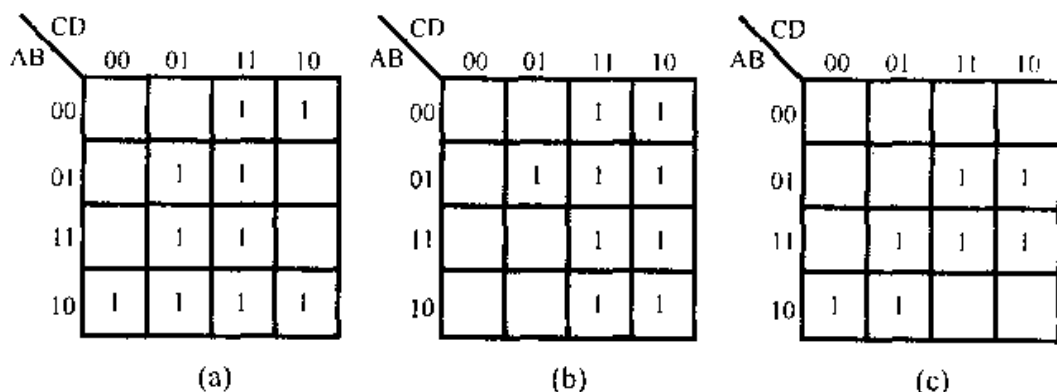


图 3.1.10 例 3.1.7 卡诺图

(a) F_1 (b) F_2 (c) F_3

(2) 画出 3 个和所有两个函数乘积卡诺图, 如图 3.1.11(a)~(d)所示。

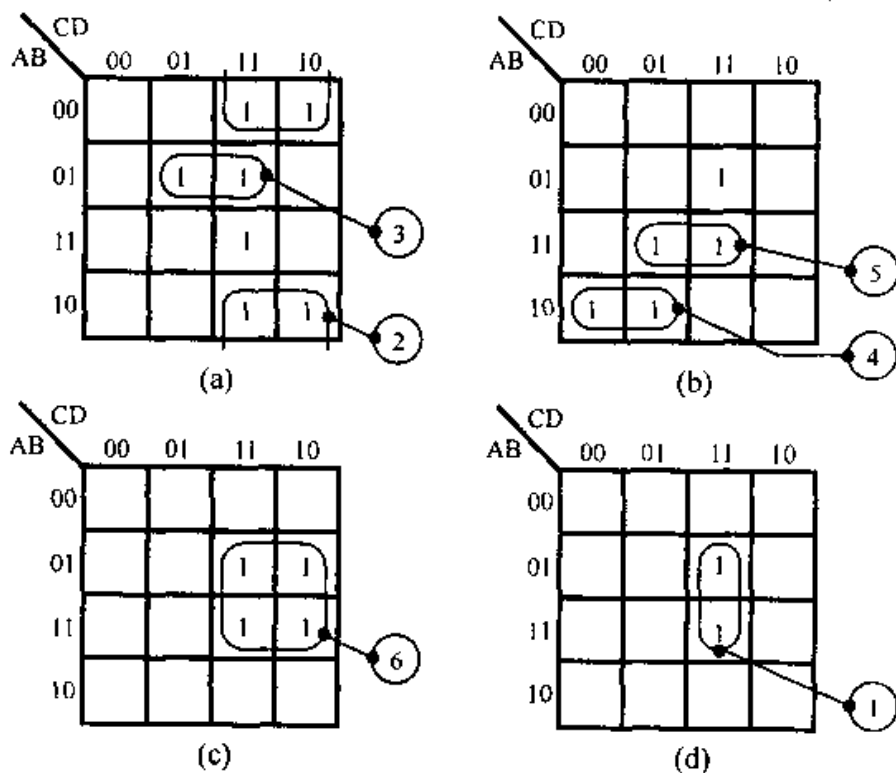


图 3.1.11 例 3.1.7 乘积卡诺图

(a) $F_1 \cdot F_2$ (b) $F_1 \cdot F_3$ (c) $F_2 \cdot F_3$ (d) $F_1 \cdot F_2 \cdot F_3$

(3) 在乘积卡诺图上求公有乘积项 (质蕴涵项)。

首先从最高阶乘积卡诺图开始, 在 $F_1 \cdot F_2 \cdot F_3$ 的卡诺图中圈出 3 个函数所公有的乘积项为圈①; 然后再考虑两个函数的乘积卡诺图, 在 $F_1 \cdot F_2$ 卡诺图中圈出 F_1 和 F_2 所公有的乘积项为圈②和圈③ (注意: 已被高阶卡诺图覆盖了的 1 应作随意态处理)。同理, 在 $F_1 \cdot F_3$ 卡诺图中得到圈④和圈⑤; 在 $F_2 \cdot F_3$ 卡诺图中得到圈⑥, 如图 3.1.11(a)~(d)所示。

(4) 在单个函数的卡诺图上确定每个函数的最佳乘积项组。

在 F_1 卡诺图上, 优先选用圈③和圈⑤。由于它们已包含了圈①中所有 1, 故圈①不能选用。此外, 还应选用圈②和圈④, 如图 3.1.12(a)所示。

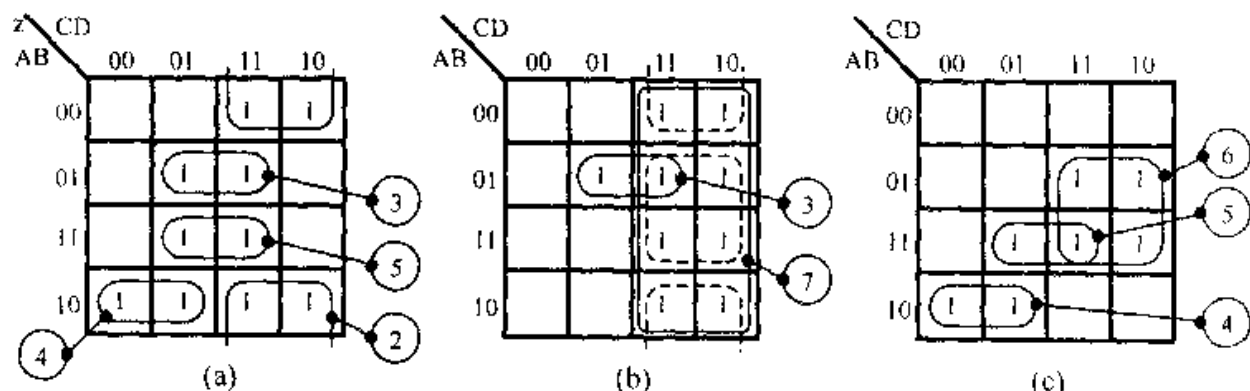


图 3.1.12 例 3.1.7 优化卡诺图

(a) F_1 (b) F_2 (c) F_3

在 F_2 卡诺图上优先选用圈③, 此外, 还可选用圈②和圈⑥ (如图 3.1.12(b)中虚线圈所示)。但是, 它们可以合并为只含有一个变量的合并圈⑦, 虽然圈⑦为 F_2 所独有, 但由于圈⑦只含有一个变量, 故无需集成门。

在 F_3 卡诺图上选用圈④、圈⑤和圈⑥, 如图 3.1.12(c)所示。

由图 3.1.12(a)~(c)可写出最佳简化结果为

$$\left. \begin{aligned} F_1 &= \bar{A}BD + ABD + \bar{B}C + A\bar{B}\bar{C} \\ F_2 &= C + \bar{A}BD \\ F_3 &= A\bar{B}\bar{C} + ABC + BC \end{aligned} \right\} \quad (3.1.13)$$

由式(3.1.12)可画出逻辑电路图, 如图 3.1.13 所示。

通过以上讨论可以发现: 设计组合电路可以采用不同类型的逻辑门实现, 也可以采用不同的电路结构(两级, 三级或多级逻辑门电路)来实现, 它们各具特色。应依据电路的使用条件和提供的设计条件来选择最佳方案。

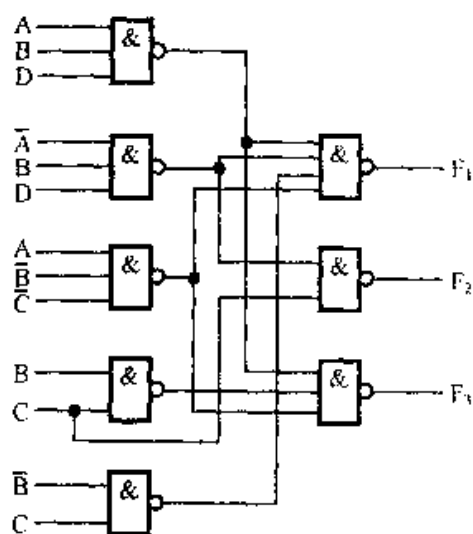


图 3.1.13 例 3.1.7 逻辑图

3.2 组合逻辑电路的分析

为了研究给定的组合电路的逻辑功能,就需要对该电路进行分析,找出电路输出与输入之间的逻辑关系,分析的步骤为

- 写出电路的逻辑函数表达式;
- 简化逻辑函数,求出最简逻辑函数或列真值表;
- 描述电路的逻辑功能。

在实际进行组合电路分析时,可视电路的繁简情况,省略某些步骤。对于较简单的电路可以直接由电路写出逻辑函数表达式,并由此描述电路功能。本节重点讨论从组合电路求逻辑函数表达式的方法以及如何从逻辑函数表达式或真值表来描述电路的逻辑功能。

例 3.2.1 分析图 3.2.1 所示组合电路。

解 从输入级开始,逐级写出各逻辑门的输出函数

$$F = \overline{AB} \cdot A \cdot \overline{BC} \cdot \overline{AB} = \overline{AB} \cdot \overline{A} \cdot \overline{BC} + AB = (A + \overline{B})\overline{A}(\overline{B} + \overline{C}) + AB = \overline{AB} + AB = A \odot B$$

上式表明该电路完成同或运算功能。

例 3.2.2 分析图 3.2.2 所示组合电路。

解 从输入级开始写出各个逻辑门的输出函数,逐级向输出端推演,直到写出

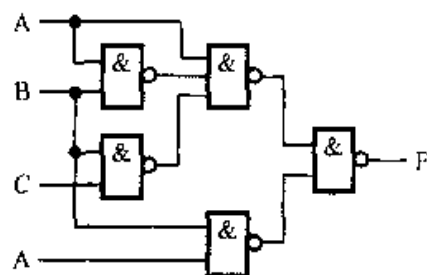


图 3.2.1 例 3.2.1 逻辑电路图

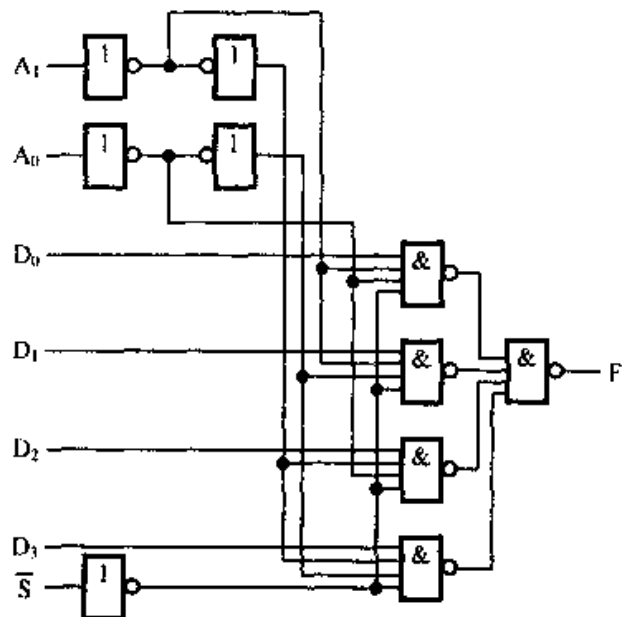


图 3.2.2 例 3.2.2 逻辑电路图

表 3.2.1 例 3.2.2 真值表

输 入							输出
\bar{S}	A_1	A_0	D_0	D_1	D_2	D_3	F
1	×	×	×	×	×	×	0
0	0	0	D_0	×	×	×	D_0
0	0	1	×	D_1	×	×	D_1
0	1	0	×	×	D_2	×	D_2
0	1	1	×	×	×	D_3	D_3

电路的输出函数, 然后用代数法进行化简, 即

$$F = \overline{A_1} \overline{A_0} D_0 \overline{S} + \overline{A_1} A_0 D_1 \overline{S} + A_1 \overline{A_0} D_2 \overline{S} + A_1 A_0 D_3 \overline{S}$$

$$= \overline{S} (\overline{A_1} \overline{A_0} D_0 + \overline{A_1} A_0 D_1 + A_1 \overline{A_0} D_2 + A_1 A_0 D_3)$$

由上式作真值表(见上页表 3.2.1)。由表可见: 当 $\overline{S}=1$ 时, $F=0$, 此时与输入信号 A_0 和 A_1 的状态无关, 即电路为禁止状态; 当 $\overline{S}=0$ 时, 输出 F 由 $A_0 A_1$ 的取值选择一个输入数据输出, 即实现数据选择器功能。由此可知该电路是一个带有使能端的 4 选 1 数据选择器(参见 3.6 节)。

3.3 编码与编码器

用数码来作为某一特定信息的代号称为代码或编码。在生活中多采用十进制数码 0~9 作为代码的符号, 例如, 248 列车; 95 中学; 8199 部队等。在数字技术中最常采用的是用二进制数码 0 和 1 作为代码的符号, 称为二进制码。必须指出: 二进制码不一定表示二进制数, 它的含义由人们预先约定。

在数字技术中, 编码就是用二进制数码来表示一个特定的信息(十进制数、文字、符号和指令等)。实现这一功能的逻辑电路称为编码器。

3.3.1 编码

1. 二-十进制编码

用一组二进制码来表示一个给定的十进制数称为二-十进制编码, 简称 BCD 码。4 位二进制码可以表示 $2^4=16$ 种不同状态, 用它来表示十进制的 0~9 时, 其中, 有 6 种状态是不可以用的, 称为禁用码组。从 16 种状态中选出 10 种来表示 0~9 这 10 个数的方法有

$$N_{\text{BCD}} = A_{16}^{10} = 16!/(16-10)! \approx 2.9 \times 10^{10}$$

种。尽管 BCD 的编码方法有如此之多, 但绝大多数使用起来很困难, 实际上经常使用的只是少数几种。

表 3.3.1 中列出常用的几种 BCD 码。下面简要介绍它们的编码方法及其特点。

(1) 有权码

有权码的特点是在表示 0~9 这 10 个数码的 4 位二进制码中的各位数码都有固定的位权值。例如, 8421 BCD 码的位权值由高至低为 8、4、2、1。对于有权码, 只需将数码 1 的位权值相加, 便可求得所代表的十进制数。例如,

$$[0\ 1\ 1\ 0]_{8421\text{BCD}} = 4+2=(6)_{10} \quad [0\ 1\ 1\ 0]_{5121\text{BCD}} = 1+2=(3)_{10}$$

表 3.3.1 常用 BCD 代码

十进制数	有权码			无权码		
	8421 码	2421 码	5121 码	余 3 码	余 3 格雷码	右移码
0	0000	0000	0000	0011	0010	00000
1	0001	0001	0001	0100	0110	10000
2	0010	0010	0010	0101	0111	11000
3	0011	0011	0011	0110	0101	11100
4	0100	0100	0111	0111	0100	11110
5	0101	1011	1000	1000	1100	11111
6	0110	1100	1100	1001	1101	01111
7	0111	1101	1101	1010	1111	00111
8	1000	1110	1110	1011	1110	00011
9	1001	1111	1111	1100	1010	00001

在数字技术中用得最多的还是 8421BCD 码, 它的位权值是按 2 的幂增加的, 这与二进制数的位权值完全一致, 其构成与使用都比较简便, 有时又称它为自然权码, 也可简称为 BCD 码或 8421 码。

2421 码和 5121 码的特点是具有自补特性, 即十进制数 0 与 9, 1 与 8、……4 与 5 等的码组互为反码(按位取反)。例如, 4 的 2421 码为 0100, 5 的 2421 码必为 1011。这种特点使它们在数字仪表中获得广泛应用。

(2) 无权码

这种代码的各位二进制数码都无固定的位权值, 所以, 不能按位权值展开求得十进制数。但是, 它们或者由于构成简易, 或者由于具有某些特点, 因而获得应用。

余 3 码 余 3 码可以由 8421 码得到, 即某十进制数加 3 所对应的 8421 码便是该十进制数的余 3 码。例如,

$$[4]_{\text{余 3 码}} = [7]_{8421 \text{ 码}} = 0111$$

余 3 码也具有自补特性, 例如, 4 和 5 的余 3 码便互为反码。

余 3 格雷码(余 3 循环码) 余 3 格雷码可由格雷码(参见表 3.3.2)得到, 即某十进制数加 3 所对应的格雷码便是该数的余 3 格雷码。例如,

$$[4]_{\text{余 3 Gray}} = [7]_{\text{Gray}} = 0100$$

右移码 右移码由 5 位码元组成。首先令十进制数 0 的右移码为 00000, 其他十进制数的右移码构成规律是, 右边 4 位由前一码组右移一位, 而最左一位由前一码组最右一位取反而成。例如, 由 2 的右移码为 11000, 则可得 3 的右移码 11100。

余 3 格雷码和右移码的共同特点是两个相邻码组之间仅有一位码元不同。这种特性称为单位间距特性。具有这种特性的编码称为单位间距码。从编码表上还可看出从数 9 的码组回到数 0 的码组, 仍然只有一位码元不同, 构成一个“循环”, 所以也称为循环码。由于它们具有单位间距特性而在逻辑电路的设计中获得广泛应

用。例如，在计数器和角度编码盘中，数的出现是按顺序进行的。这样，在一个数转换至相邻数的过程中，就不会出现其他许用码组。假如使用其他 BCD 码，由于相邻码组之间有两个或两个以上的码元不同，就可能出现其他许用码组。例如，使用 8421BCD 码，从 3(0011)转换到 4(0100)时，有 3 个码元不同，但实际电路不可能使 3 个码元在同一瞬间改变，总有先后之分，若低位先改变，那么，在改变过程中就会出现 0010 和 0000 码组，即产生瞬间状态模糊，尽管时间是短暂的，但在某些场合下是不允许的，它可能导致逻辑差错或产生不应有的噪声(例如，在译码器中就产生译码噪声)，就应在电路上采取措施防止逻辑差错或者使噪声尽可能削弱。但是，使用上述两种码就能从根本上杜绝产生瞬间状态模糊的可能性，这就是它们获得广泛应用的原因。

2. 二进制编码

用 n 位二进制数码来表示 m 个特定信息(特定对象)称为二进制编码， n 与 m 之间必须满足条件： $2^n \geq m$ 。显然，二进制编码的特定对象不受限制，若特定对象增加，可以用增加二进制数码位数的方法解决。

在数字电路中最常用的二进制编码是格雷码和奇偶检验码。

(1) 格雷码(Gray)

格雷码是一种无权码，它具有多种形式。各种格雷码的共同特点是具有单位间距特性，其编码表列于表 3.3.2。

表 3.3.2 常用二进制代码

N	二进制码	Gray 码	奇检验 8421 码	偶检验 8421 码
0	0000	0000	00001	00000
1	0001	0001	00010	00011
2	0010	0011	00100	00101
3	0011	0010	00111	00110
4	0100	0110	01000	01001
5	0101	0111	01011	01010
6	0110	0101	01101	01100
7	0111	0100	01110	01111
8	1000	1100	10000	10001
9	1001	1101	10011	10010
10	1010	1111		
11	1011	1110		
12	1100	1010		
13	1101	1011		
14	1110	1001		
15	1111	1000		
⋮	⋮	⋮		

格雷码可以由二进制码转换得到,即已知一组二进制码便可求出一组对应的格雷码,反之亦然。由二进制码转换为格雷码的规则是格雷码的第 n 位等于二进制码的第 n 位与 $n+1$ 位的异或(模 2 和),即

$$G_n = B_n \oplus B_{n+1}$$

例如,求 6 的格雷码。首先在二进制码最高位前补一个 0,然后由下式求得

$$\begin{array}{ccccccc} (6)_{10} & = & 0 & 0 & 1 & 1 & 0 & \text{二进制码} \\ & & & \vee & \vee & \vee & \vee & \\ & & \oplus & \oplus & \oplus & \oplus & & \\ & & 0 & 1 & 0 & 1 & & \text{格雷码} \end{array}$$

格雷码还有一个特点,即十进制数 2^n-1 (n 为正整数)的格雷码都是在 n 位为 1,其余位为 0。例如, $n=4$, $2^4-1=15$,其格雷码为 1000。因此,十进制数 0 与 (2^n-1) 所对应的格雷码之间只有一位码元不同。若在 2^n-1 范围内编码,它就是典型格雷码,也是一种循环码。

(2) 奇偶检验码

在数字设备中,由于存在噪声和干扰,故在代码的产生、变换和传输过程中可能会发生差错。为了减少差错,人们对编码方法进行了深入研究,提出了许多种不易出错的代码,如右移码、余 3 循环码、格雷码等;同时还提出了一些能够发现差错的代码以及既能发现差错、又能确定差错位置并予以纠正的代码。在编码技术中将具有上述这些特性的代码统称为可靠性编码。

奇偶检验码是常用的具有检测一位差错能力的代码。它由两部分构成:信息码和奇偶检验位(简称为奇偶位或检验位)。信息码就是被传的信息,可以是位数不定的二进制码组;奇偶检验位又常称为监督码元,仅有一位,记为 P 。监督码元 P 的取值取决于数字系统是采用奇检验或偶检验。若规定采用奇检验,则 P 的取值应使奇偶检验码中为 1 的码元是奇数;若规定采用偶检验,则 P 的取值应使奇偶检验码中为 1 的码元是偶数,(但信息码中的全 0 码除外,它的偶检验码仍为全 0)。收到奇偶检验码之后要进行检验,即检验码组中为 1 码元的奇偶性是否符合原先规定,因而具有检测一位差错(奇数差错)的功能,但是,这类码不能检测码组中发生“双错”的情况。由于数字系统中发生一位码差错的概率比两位码同时差错的概率大得多,所以,在数字设备和计算机中获得广泛应用。前面介绍的各种代码都可以构成奇偶检验码。8421BCD 码的奇偶检验码列于表 3.3.2 中。

3.3.2 编码器

1. 编码器的设计

图 3.3.1 是编码器的框图,它有 m 个输入端, n 个输出端, m 与 n 之间满足条件 $2^n \geq m$ 。因此,编码器是一个多输入、多输出的组合电路,可以用组合电路的设

计方法来设计。二-十进制编码器和二进制编码器的设计过程是类似的。现以最常用的 8421 码编码器为例作些简要说明。设计步骤如下：

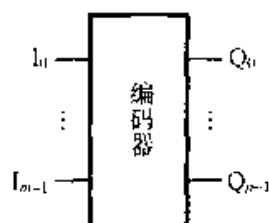


图 3.3.1 编码器框图

表 3.3.3 8421BCD 码编码器真值表

I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	Q_3	Q_2	Q_1	Q_0
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	1

(a) 列出十进制数与 8421 码的编码表。由表 3.3.3 可见，10 个输入信号中只有一个为 1，其余均为 0，这称为输入信号高电平有效；反之，则称为输入信号低电平有效。无论哪一种情况，编码器只能对一个输入信号进行编码，所以，输入信号中不允许出现两个或多个同时有效的情况，即输入信号具有互相排斥的特性。因此，可以用简化的编码表来代替，如表 3.3.4 所示。

(b) 写出逻辑函数表达式。分析编码表可知：当 I_8 或者 I_9 为 1 时，输出 Q_3 为 1，即 $Q_3 = I_8 + I_9$ 。同理可求 Q_2 、 Q_1 和 Q_0 ，即

$$Q_3 = I_8 + I_9 = \overline{I_8} \cdot \overline{I_9}$$

$$Q_2 = I_4 + I_5 + I_6 + I_7 = \overline{I_4} \cdot \overline{I_5} \cdot \overline{I_6} \cdot \overline{I_7}$$

$$Q_1 = I_2 + I_3 + I_6 + I_7 = \overline{I_2} \cdot \overline{I_3} \cdot \overline{I_6} \cdot \overline{I_7}$$

$$Q_0 = I_1 + I_3 + I_5 + I_7 + I_9 = \overline{I_1} \cdot \overline{I_3} \cdot \overline{I_5} \cdot \overline{I_7} \cdot \overline{I_9}$$

(c) 画逻辑图，如图 3.3.2 所示。

表 3.3.4 表 3.3.3 的简化表

N	Q_3	Q_2	Q_1	Q_0
I_0	0	0	0	0
I_1	0	0	0	1
I_2	0	0	1	0
I_3	0	0	1	1
I_4	0	1	0	0
I_5	0	1	0	1
I_6	0	1	1	0
I_7	0	1	1	1
I_8	1	0	0	0
I_9	1	0	0	1

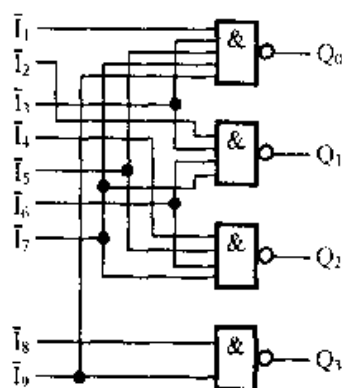


图 3.3.2 8421 码编码器

需要指出：图中没有 I_0 输入端，即对十进制数 0 的编码具有隐含性。当输入信号 $I_1 \sim I_9$ 均为 0 时，电路的输出 0000 就是数 0 的 8421 码。

2. 中规模集成优先编码器

在许多实际应用中，编码器的输入端有可能同时收到几个信号，这时要求按预先规定的优先次序输出。例如，微机中的中断控制就有优先级别，在这种情况下要求只对优先级别高的输入信号进行编码，完成这种功能的编码器称为优先编码器。

我国已生产了优先编码器的系列产品。8-3 线优先编码器有 74LS148(TTL 型)、CE10165(ECL 型)、CC4532(CMOS 型)；10-4 线优先编码器有 74LS147、CC40147 等(本书中对各类型器件仅只列出一种或少数几种典型产品型号,系列产品的型号与主要性能可查阅文献[28]或其他集成电路手册)。

74LS148 型 8-3 线优先编码器逻辑图示于图 3.3.3(a)，图(b)是逻辑框图。

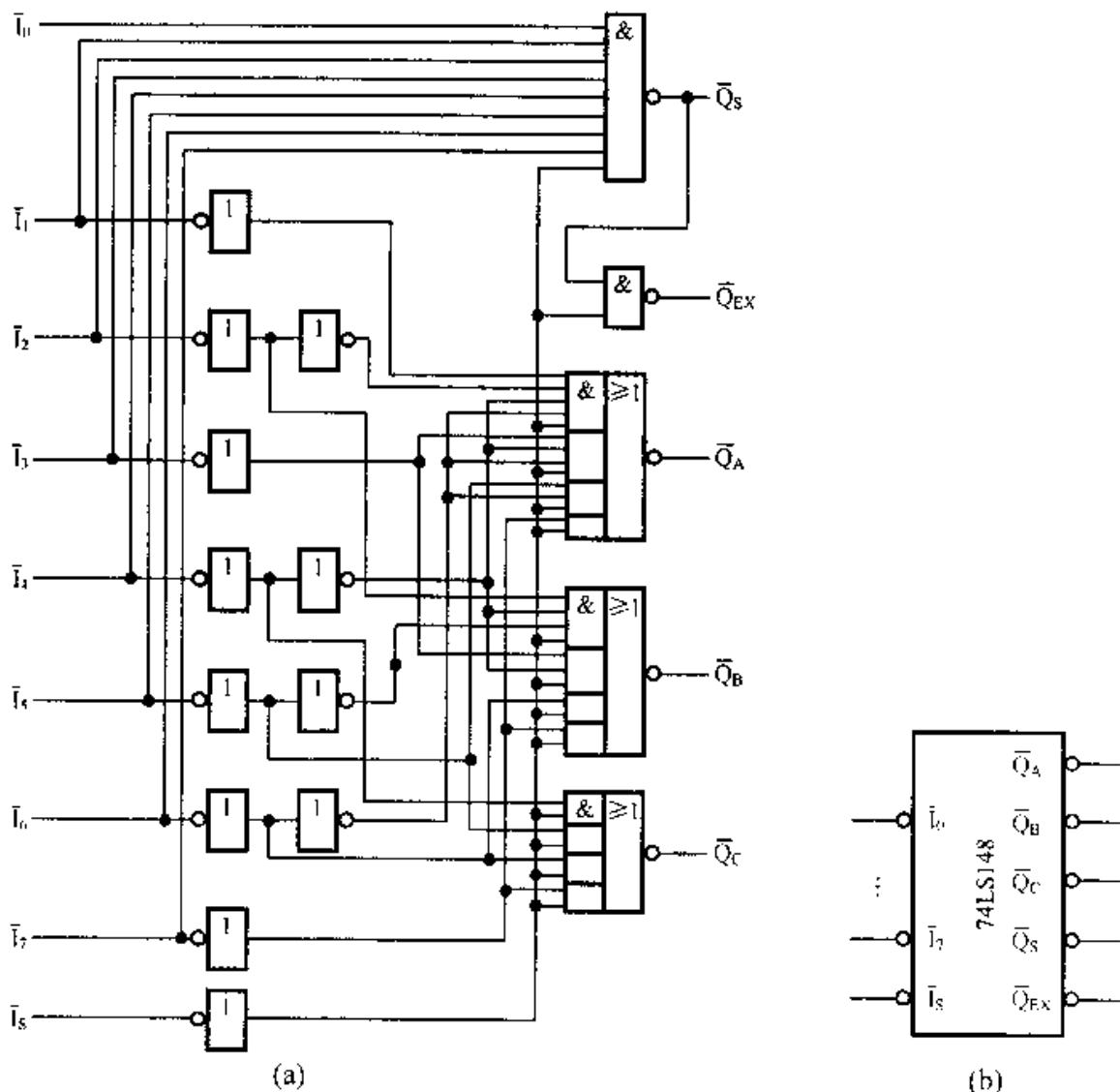


图 3.3.3 74LS148 型优先编码器

(a) 逻辑图 (b) 逻辑框图

$\bar{I}_0 \sim \bar{I}_7$ 是输入端, $\bar{Q}_C \bar{Q}_B \bar{Q}_A$ 为输出端, \bar{I}_S 为使能输入端。为了扩展组件功能还增设有输出使能(允许输出)端 \bar{Q}_S 和群编码选择端 \bar{Q}_{EX} 。图 3.3.3(b)所示逻辑框图中输入及输出线上的小圆圈表示低电平有效, 而 \bar{Q}_C 、 \bar{Q}_B 、 \bar{Q}_A 输出线上小圆圈表示编码输出 $\bar{Q}_C \bar{Q}_B \bar{Q}_A$ 是相应输入信号的反码输出。

由图 3.3.3(a)逻辑图可写出

$$\begin{aligned}\bar{Q}_C &= (\bar{I}_4 + \bar{I}_5 + \bar{I}_6 + \bar{I}_7) \bar{I}_S \\ \bar{Q}_B &= (\bar{I}_2 \bar{I}_4 \bar{I}_5 + \bar{I}_3 \bar{I}_4 \bar{I}_5 + \bar{I}_6 + \bar{I}_7) \bar{I}_S \\ \bar{Q}_A &= (\bar{I}_1 \bar{I}_2 \bar{I}_4 \bar{I}_6 + \bar{I}_3 \bar{I}_4 \bar{I}_6 + \bar{I}_5 \bar{I}_6 + \bar{I}_7) \bar{I}_S \\ \bar{Q}_S &= \bar{I}_0 \bar{I}_1 \bar{I}_2 \bar{I}_3 \bar{I}_4 \bar{I}_5 \bar{I}_6 \bar{I}_7 \cdot \bar{I}_S \\ \bar{Q}_{EX} &= \bar{Q}_S \cdot \bar{I}_S\end{aligned}$$

由上式可作出功能表, 如表 3.3.5 所示。由表可见:

(a) 当 $\bar{I}_S=1$ 时, 组件为禁止状态, 输出 \bar{Q}_C 、 \bar{Q}_B 、 \bar{Q}_A 和 \bar{Q}_S 、 \bar{Q}_{EX} 均为 1。

(b) 当 $\bar{I}_S=0$ 时, 组件为工作状态。若输入均无信号, 即 $\bar{I}_1 \sim \bar{I}_7$ 均为 1 电平, 则无信号输出, 即 \bar{Q}_C 、 \bar{Q}_B 、 \bar{Q}_A 均为 1, 同时输出 $\bar{Q}_{EX}=1$ 和 $\bar{Q}_S=0$, 表示本级组件无输出, 即不在本级电路工作范围, 而是受 \bar{Q}_S 控制的低一级电路工作, 如表 3.3.5 中第 2 行所示; 若输入有信号, 则 $\bar{Q}_C \bar{Q}_B \bar{Q}_A$ 输出相应的二进制码, 其优先次序为 $\bar{I}_7 \sim \bar{I}_0$, 并且是反码输出。此时, $\bar{Q}_{EX}=0$ 和 $\bar{Q}_S=1$, 表示本组件有输出, 而受 \bar{Q}_S 控制的低位组件停止工作。如表 3.3.5 中第 3 行所示, 只要 \bar{I}_7 有信号($\bar{I}_7=0$), 则无论其他输入端有无信号, 其输出为 7 的反码 000。其余各行依此类推。

表 3.3.5 8-3 线优先编码器功能表

输 入									输 出				
\bar{I}_S	\bar{I}_0	\bar{I}_1	\bar{I}_2	\bar{I}_3	\bar{I}_4	\bar{I}_5	\bar{I}_6	\bar{I}_7	\bar{Q}_C	\bar{Q}_B	\bar{Q}_A	\bar{Q}_{EX}	\bar{Q}_S
1	Φ	Φ	Φ	Φ	Φ	Φ	Φ	Φ	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	Φ	Φ	Φ	Φ	Φ	Φ	Φ	0	0	0	0	0	1
0	Φ	Φ	Φ	Φ	Φ	Φ	0	1	0	0	1	0	1
0	Φ	Φ	Φ	Φ	Φ	0	1	1	0	1	0	0	1
0	Φ	Φ	Φ	Φ	0	1	1	1	0	1	1	0	1
0	Φ	Φ	Φ	0	1	1	1	1	1	0	0	0	1
0	Φ	Φ	0	1	1	1	1	1	1	0	1	0	1
0	Φ	0	1	1	1	1	1	1	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1

图 3.3.4 是用两片 74LS148 扩展为 16-4 线优先编码器的连接图。高位片(组件(1))的使能端 \bar{I}_S 接选通信号 \bar{S} , 低位片(组件(2))的使能端 \bar{I}_S 由高位片的输出使能端 \bar{Q}_S 控

制。当 $\bar{S}=0$ 时, 高位片被选通, 若它的输入端都无信号(输入端 8~15 均为 1), 则有 $\bar{Q}_{EX}=1$, 表示现在不是本级工作范围; 同时有 $\bar{Q}_S=0$, 它为低位片提供了选通信号, 低位片处于工作状态, 所以, $\bar{Q}_C \sim \bar{Q}_A$ 的输出状态由 $\bar{I}_0 \sim \bar{I}_7$ 中数值最大的输入信号来确定。例如, 若输入 \bar{I}_5 是数值最大的输入信号, 则输出为 $\bar{Q}_C \bar{Q}_B \bar{Q}_A = 010$, 此时高位片 $\bar{Q}_{EX}=1$, 即 $\bar{Q}_D=1$ 。所以, 输出为 $\bar{Q}_D \bar{Q}_C \bar{Q}_B \bar{Q}_A = 1010$ (5 的反码)。

当 $\bar{S}=0$ 时, 若高位片有输入信号, 则有 $\bar{Q}_{EX}=0$ 和 $\bar{Q}_S=1$ 。这表示本级组件工作, 并且禁止低级组件工作, 故低位片输出为 $\bar{Q}_C \bar{Q}_B \bar{Q}_A = 111$ 。

此时, 整个组件的输出由高位片的输入 $\bar{I}_8 \sim \bar{I}_{15}$ 中数值最大的输入信号确定。例如, 输入 \bar{I}_{11} 是数值最大的输入信号, 由于输入 \bar{I}_{11} 接在高位片 \bar{I}_3 上, 所以, $\bar{Q}_C \bar{Q}_B \bar{Q}_A = 100$, 此时, 高位片 $\bar{Q}_{EX}=0$, 输出为 $\bar{Q}_D \bar{Q}_C \bar{Q}_B \bar{Q}_A = 0100$ (11 的反码)。

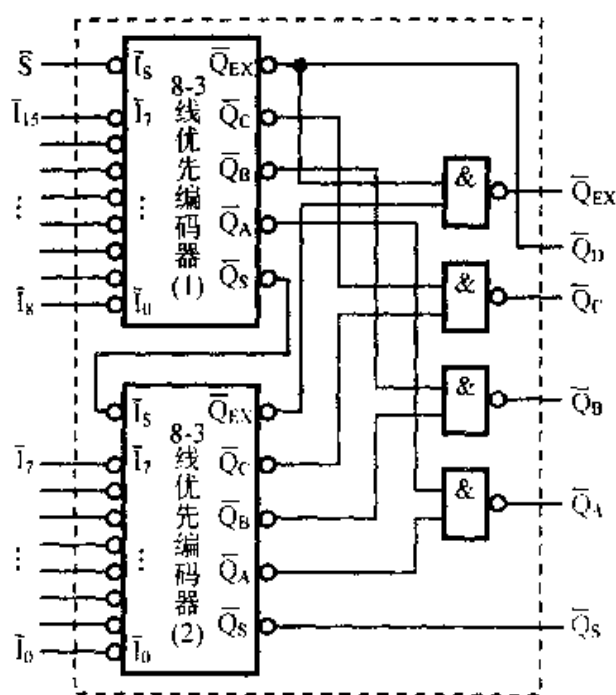


图 3.3.4 8-3 线扩展为 16-4 线优先编码器

3.4 译码与译码器

把代码的特定含义翻译出来的过程称为译码, 显然译码是编码的逆过程。译码可以将二进制代码翻译成十进制数、字符或其他进制的代码; 也可将十进制数或其他进制代码译成所需进制的信号码。实现译码功能的逻辑电路称为译码器, 它在数字技术中有着广泛的应用。例如, 各种数字显示译码器、规则码发生器和分配器都是在译码器的基础上构成的。在计算机技术中也广泛使用译码器, 例如, 存储器的地址译码器, 控制器中的指令译码器等等。

译码器依据用途可分为通用译码器, 码制转换译码器, 数字显示译码器等。

3.4.1 译码器的设计

图 3.4.1 是译码器的框图。它有 n 个输入端, m 个输出端, n 与 m 之间满足条件 $2^n \geq m$ 。所以, 译码器是一个多输入、多输出的组合电路, 任何类型的译码器都

可以用组合电路的设计方法来设计。现以最常用的将 8421 码译成十进制数的二-十进制译码器为例来作些简要说明, 设计步骤如下。

(a) 列出 8421 码与十进制数的真值表, 如表 3.4.1 所示, 此表又称为功能表或译码表。因为输入一组 8421 码, 只有相应的输出端为 1, 其余 9 个输出端均为 0, 所以, 表 3.4.1 是一种简化的功能表, 表中有 6 种状态(1010~1111)不用, 因而可以作随意态处理。

(b) 写出最简逻辑函数表达式。由表 3.4.1 可得 $Q_0 \sim Q_9$ 的卡诺图, 如图 3.4.2 所示, 由图求出 $Q_0 \sim Q_9$ 的最简逻辑函数表达式(又称为译码方程)如下:

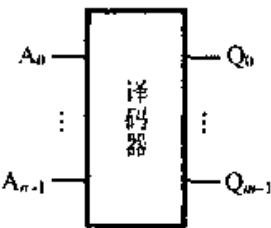


图 3.4.1 译码器框图

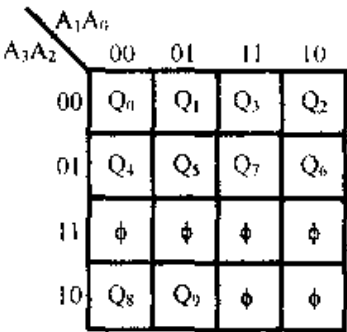


图 3.4.2 $Q_0 \sim Q_9$ 的卡诺图

表 3.4.1 4-10 线译码器真值表

8421 码				十进制数
A_3	A_2	A_1	A_0	N
0	0	0	0	Q_0
0	0	0	1	Q_1
0	0	1	0	Q_2
0	0	1	1	Q_3
0	1	0	0	Q_4
0	1	0	1	Q_5
0	1	1	0	Q_6
0	1	1	1	Q_7
1	0	0	0	Q_8
1	0	0	1	Q_9
1	0	1	0	ϕ
		\vdots		
1	1	1	1	

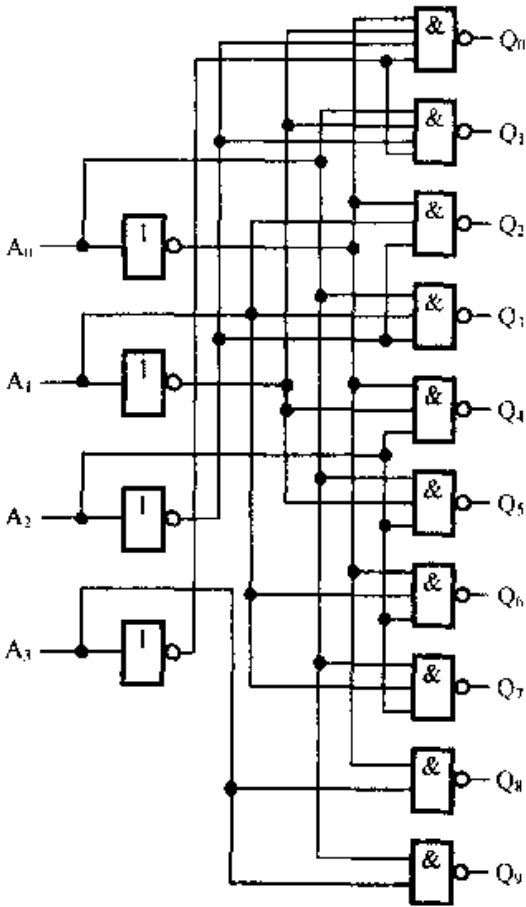


图 3.4.3 二-十进制译码器

$$\begin{aligned}
 Q_0 &= \bar{A}_3 \bar{A}_2 \bar{A}_1 \bar{A}_0 & Q_1 &= \bar{A}_3 \bar{A}_2 \bar{A}_1 A_0 \\
 Q_2 &= \bar{A}_2 A_1 \bar{A}_0 & Q_3 &= \bar{A}_2 A_1 A_0 \\
 Q_4 &= \bar{A}_2 \bar{A}_1 \bar{A}_0 & Q_5 &= A_2 \bar{A}_1 \bar{A}_0 \\
 Q_6 &= A_2 A_1 \bar{A}_0 & Q_7 &= A_2 A_1 A_0 \\
 Q_8 &= A_3 \bar{A}_0 & Q_9 &= A_3 A_0
 \end{aligned}$$

(c) 画出逻辑图, 见上页图 3.4.3.

3.4.2 中规模集成通用译码器

通用译码器有二进制译码器和二十进制译码器两类产品。

二进制译码器产品有双 2-4 线译码器(74LS139, CE10172, CC4555 等); 3-8 线译码器(74LS138, CE10161, CC74HC138 等); 4-16 线译码器(74154, CC4515, CC74HC154 等)。

二十进制译码器通常称为 4-10 线译码器, 其产品有 74LS145, CC74HC42 等。

1. 二进制译码器

(1) 3-8 线译码器

3-8 线译码器 74LS138 逻辑电路如图 3.4.4(a)所示, 图(b)是逻辑框图。

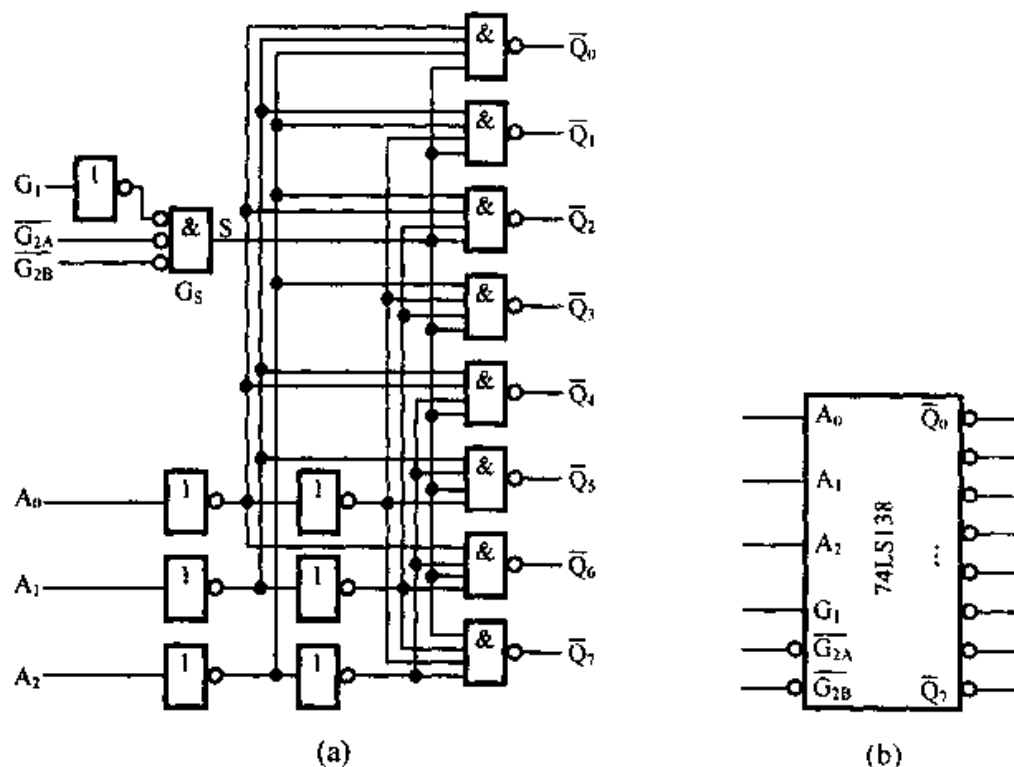


图 3.4.4 74LS138 型 3-8 线译码器

(a) 逻辑电路图 (b) 逻辑框图

其功能表如表 3.4.2 所示。该组件设置有 3 个使能输入端 G_1 、 \overline{G}_{2A} 和 \overline{G}_{2B} ，只有在 G_1 为 1， \overline{G}_{2A} 和 \overline{G}_{2B} 均为 0 时组件才工作；它们的其它取值均使组件为禁止状态。图 3.4.4(b) 中 G_1 表示高电平有效，而使能端 \overline{G}_{2A} 和 \overline{G}_{2B} 上的小圆圈表示低电平有效。输出端 $\overline{Q}_0 \sim \overline{Q}_7$ 上的小圆圈表示对应译码地址输入 $A_2 \sim A_0$ 的每一组代码输入，都能译成在相应输出端输出低电平，称它为译码输出是低电平有效。

表 3.4.2 3-8 线译码器功能表

输 入						输 出							
G_1	\overline{G}_{2A}	\overline{G}_{2B}	A_2	A_1	A_0	\overline{Q}_0	\overline{Q}_1	\overline{Q}_2	\overline{Q}_3	\overline{Q}_4	\overline{Q}_5	\overline{Q}_6	\overline{Q}_7
$\times^{①}$	1	\times	\times	\times	\times	1	1	1	1	1	1	1	1
\times	\times	1	\times	\times	\times	1	1	1	1	1	1	1	1
0	\times	\times	\times	\times	\times	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	0	0	1	1	1	1	0	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0	1	1
1	0	0	1	1	0	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0

从功能表或逻辑图都可写出译码方程：

$$\left. \begin{aligned} \overline{Q}_0 &= G_1 \cdot \overline{G}_{2A} \cdot \overline{G}_{2B} \cdot \overline{A}_2 \overline{A}_1 \overline{A}_0 \\ &\vdots \\ \overline{Q}_7 &= G_1 \cdot \overline{G}_{2A} \cdot \overline{G}_{2B} \cdot A_2 A_1 A_0 \end{aligned} \right\} \quad (3.4.1)$$

在组件工作状态下，它可以产生三变量的全部最小项的反函数，即 $\overline{Q}_0 = \overline{m}_0, \dots, \overline{Q}_7 = \overline{m}_7$ 。

(2) 4-16 线译码器

4-16 线译码器 74LS154 逻辑电路如图 3.4.5(a) 所示，图(b)是逻辑框图。

其功能表如表 3.4.3 所示。该组件设置有两个使能输入端 \overline{S}_A 和 \overline{S}_B ，只有当它们均为 0 时组件才工作；它们的其它取值均使组件为禁止状态。图(b)中使能端 \overline{S}_A 和 \overline{S}_B 上的小圆圈表示两个使能信号均是低电平有效；输出端 $\overline{Q}_0 \sim \overline{Q}_{15}$ 上的小圆圈表示对应译码地址输入 $A_3 \sim A_0$ 的每一组代码输入，都能译成在相应输出端输出低电平，有

① 手册中一般多用 \times 表示随意态。

时简称它为译码输出是低电平有效。

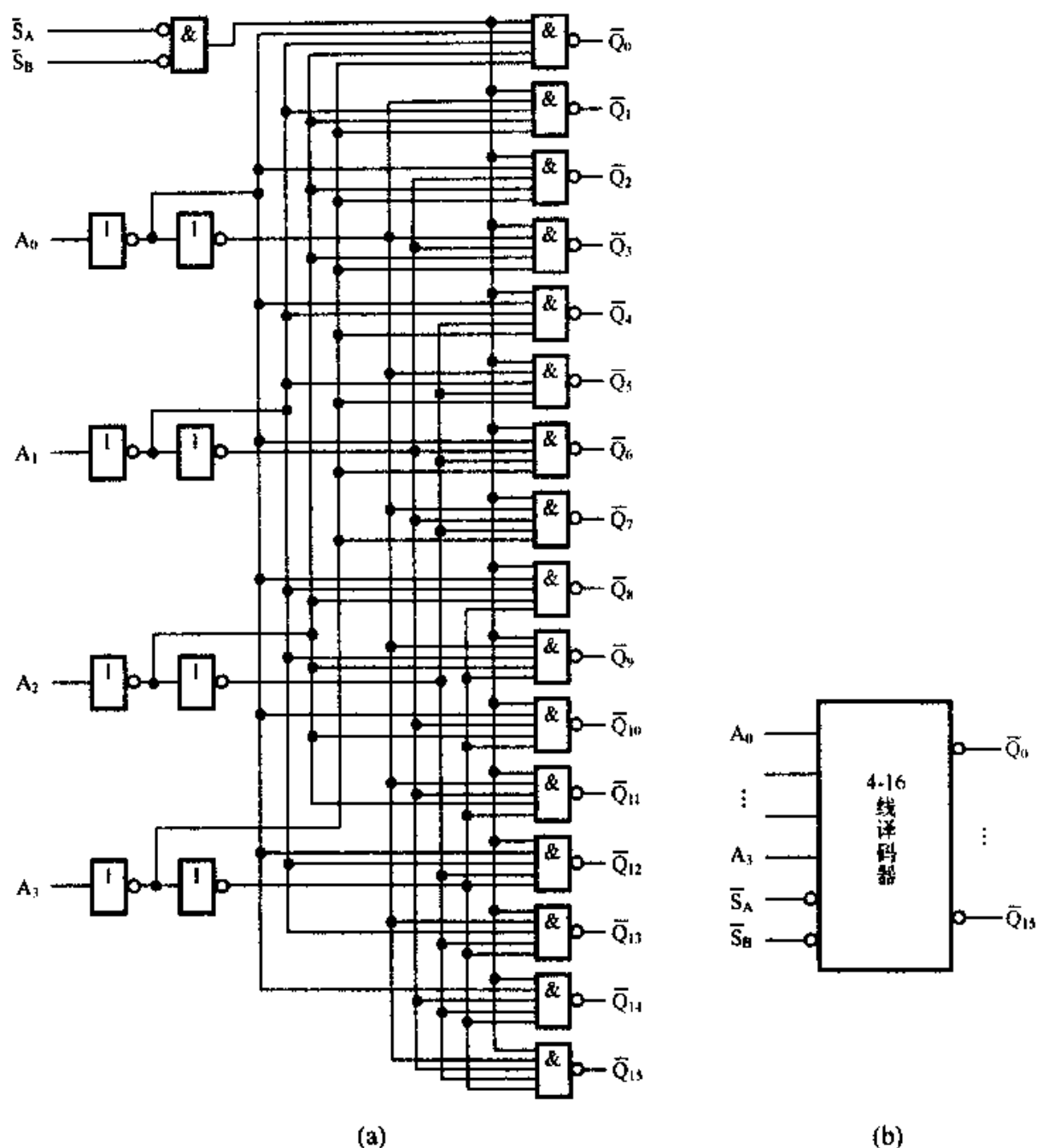


图 3.4.5 74LS154 型 4-16 线译码器

(a) 逻辑电路图 (b) 逻辑框图

4-16 线译码器是通用性很强的器件，它可以很方便地构成各种 BCD 码 4-10 线译码器，如表 3.4.4 所示。

例如，用该组件构成余 3 码 4-10 线译码器，则可将输出 \bar{Q}_3 作为 \bar{Q}'_0 ，…… \bar{Q}_9 作为 \bar{Q}'_9 ，而 $\bar{Q}_0, \bar{Q}_1, \bar{Q}_2, \bar{Q}_{13}, \bar{Q}_{14}, \bar{Q}_{15}$ 等 6 种输出不用。这样 $\bar{Q}'_0 \sim \bar{Q}'_9$ 便构成余 3 码 4-10 线译码器。

表 3.4.3 4-16 线译码器功能表

输 入					输 出																
\bar{S}_A	\bar{S}_B	A_3	A_2	A_1	A_0	\bar{Q}_0	\bar{Q}_1	\bar{Q}_2	\bar{Q}_3	\bar{Q}_4	\bar{Q}_5	\bar{Q}_6	\bar{Q}_7	\bar{Q}_8	\bar{Q}_9	\bar{Q}_{10}	\bar{Q}_{11}	\bar{Q}_{12}	\bar{Q}_{13}	\bar{Q}_{14}	\bar{Q}_{15}
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	0	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
0	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
0	0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	1																				
1	0	×	×	×	×	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1																				

表 3.4.4 4-16 线译码器构成 BCD 码 4-10 线译码器

输 入					输 出					
A_3	A_2	A_1	A_0		十进制数	8421 码	2421 码	4221 码	余 3 码	余 3 循环码
0	0	0	0		0	0	0	0		
0	0	0	1		1	1	1	1		
0	0	1	0		2	2	2	2		0
0	0	1	1		3	3	3	3	0	
0	1	0	0		4	4	4		1	4
0	1	0	1		5	5	5		2	3
0	1	1	0		6	6	6	4	3	1
0	1	1	1		7	7	7	5	4	2
1	0	0	0		8	8			5	
1	0	0	1		9	9			6	
1	0	1	0		10				7	9
1	0	1	1		11				8	
1	1	0	0		12			6	9	5
1	1	0	1		13			7		6
1	1	1	0		14		8	8		8
1	1	1	1		15		9	9		7

灵活地运用使能端可以扩展组件功能。例如,用两片 74LS154 便可方便地构成 5-32 线译码器,如图 3.4.6 所示。

2. 二-十进制译码器

4-10 线译码器 74LS145 的逻辑框图示于图 3.4.7。其功能表(又称译码表)示于表 3.4.5。由表可见:对应译码地址输入 $A_3 \sim A_0$ 的每一组代码输入,都在相应输出端输出低电平,即输出译码信号。如果有未被使用的码组(1010~1111)输入,则输出端均为 1 电平,即无信号输出。

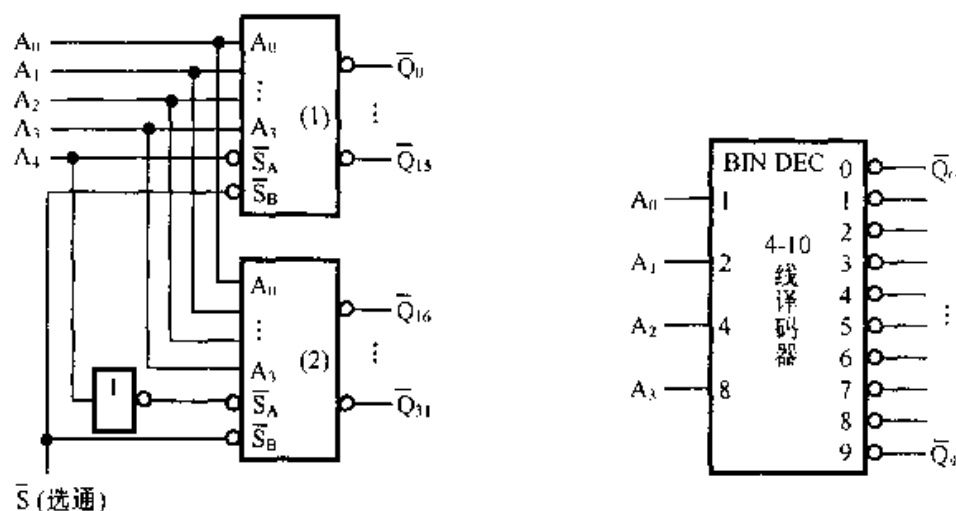


图 3.4.6 4-16 线扩展为 5-32 线译码器

图 3.4.7 4-10 线译码器逻辑框图

表 3.4.5 4-10 线译码器功能表

N	输 入				输 出									
	A_3	A_2	A_1	A_0	\bar{Q}_0	\bar{Q}_1	\bar{Q}_2	\bar{Q}_3	\bar{Q}_4	\bar{Q}_5	\bar{Q}_6	\bar{Q}_7	\bar{Q}_8	\bar{Q}_9
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1	1	1
2	0	0	1	0	1	1	0	1	1	1	1	1	1	1
3	0	0	1	1	1	1	1	0	1	1	1	1	1	1
4	0	1	0	0	1	1	1	1	0	1	1	1	1	1
5	0	1	0	1	1	1	1	1	1	0	1	1	1	1
6	0	1	1	0	1	1	1	1	1	1	0	1	1	1
7	0	1	1	1	1	1	1	1	1	1	1	0	1	1
8	1	0	0	0	1	1	1	1	1	1	1	1	0	1
9	1	0	0	1	1	1	1	1	1	1	1	1	1	0
10	1	0	1	0	全 1									
11	1	0	1	1										
12	1	1	0	0	全 1									
13	1	1	0	1										
14	1	1	1	0	全 1									
15	1	1	1	1										

该组件可方便地构成 3-8 线译码器,只需将选通信号 S 加至 A_3 , 输出端 \bar{Q}_8 、 \bar{Q}_9 ,

不用, 连接图示于图 3.4.8.

3. 用译码器设计组合逻辑电路

用中规模组件设计组合逻辑电路是一种有效的方法。在本章所介绍的常用中规模组件中有许多就可以用来设计组合逻辑电路。因此, 在介绍这些组件之后, 将结合实例来说明用中规模组件设计组合逻辑电路的方法。

例 3.4.1 设计一个多输出组合逻辑电路实现下列函数:

$$\left. \begin{aligned} F_1 &= A\bar{C} + \bar{A}BC + A\bar{B}C \\ F_2 &= BC + \bar{A}\bar{B}C \\ F_3 &= \bar{A}B + \bar{A}\bar{B}C \\ F_4 &= \bar{A}B\bar{C} + \bar{B}\bar{C} + ABC \end{aligned} \right\} \quad (3.4.2)$$

解 将式(3.4.2)化为标准与或式, 得到

$$\begin{aligned} F_1 &= A\bar{B}\bar{C} + A\bar{B}C + \bar{A}BC + A\bar{B}C = m_3 + m_4 + m_5 + m_6 \\ F_2 &= ABC + \bar{A}BC + \bar{A}\bar{B}C = m_1 + m_3 + m_7 \\ F_3 &= \bar{A}BC + \bar{A}B\bar{C} + \bar{A}\bar{B}C = m_2 + m_3 + m_5 \\ F_4 &= \bar{A}B\bar{C} + \bar{A}B\bar{C} + \bar{A}\bar{B}\bar{C} + ABC = m_0 + m_2 + m_4 + m_7 \end{aligned}$$

利用摩根定律对上式进行变换, 得到

$$\begin{aligned} F_1 &= \overline{\bar{m}_3 \cdot \bar{m}_4 \cdot \bar{m}_5 \cdot \bar{m}_6} \\ F_2 &= \overline{\bar{m}_1 \cdot \bar{m}_3 \cdot \bar{m}_7} \\ F_3 &= \overline{\bar{m}_2 \cdot \bar{m}_3 \cdot \bar{m}_5} \\ F_4 &= \overline{\bar{m}_0 \cdot \bar{m}_2 \cdot \bar{m}_4 \cdot \bar{m}_7} \end{aligned}$$

74LS138(3-8 线译码器)可以在 8 个输出端输出 3 个变量的反函数, 即 $\bar{m}_0 \sim \bar{m}_7$, 所以利用 3-8 线译码器和 4 个与非门便可构成实现式 (3.4.2) 的组合逻辑电路, 如图 3.4.9 所示。

顺便指出: 若译码器选用的译码输出是高电平有效的组件, 即 $\bar{Q}_0 = m_0, \dots, \bar{Q}_7 = m_7$, 则只需将图 3.4.9 中的与非门改换为或门就可以了。

由本例可以看出: n 位二进制译码器的输出提供了 n 变量的全部最小项

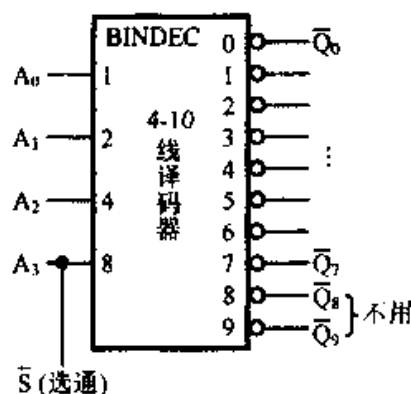


图 3.4.8 4-10 线构成 3-8 线译码器

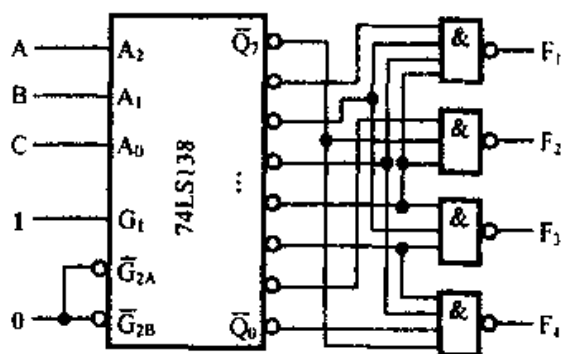


图 3.4.9 例 3.3.3 的电路

$\bar{m}_0 \sim \bar{m}_{n-1}$, 或者全部最小项的反函数 $\bar{m}_0 \sim \bar{m}_{n-1}$, 所以, 只要利用合适的附加门(输出门)将它们适当地组合起来便可构成任何形式的 n 变量的组合逻辑电路。

3.4.3 码制转换译码器

如前所述, 在数字电路中常采用多种形式的编码, 因而经常需把一种形式的编码转换成另一种形式的编码, 称为码制转换或码组转换, 这种转换可以用组合电路来实现, 也可以用时序电路或者大规模集成电路(ROM)来实现。

实现码制转换功能的逻辑电路称为码制转换译码器。现通过例子来讨论用组合电路实现码制转换译码器的设计方法, 先设计 4 位二进制码至格雷码的码制转换译码器。

任何码制转换译码器都可以用传统的“四步法”设计, 但是, 对于某些设计任务采用“灵活设计法”, 即利用逻辑函数的某些特性或者逻辑运算的某些特点来设计, 可以获得事半功倍的效果, 本例就属于这种情况。

先列出二进制码与格雷码的编码表, 如表 3.4.6 所示。直接分析编码表即可写出

$$\left. \begin{aligned} G_3 &= B_3 \\ G_2 &= B_3 \oplus B_2 \\ G_1 &= B_2 \oplus B_1 \\ G_0 &= B_1 \oplus B_0 \end{aligned} \right\} \quad (3.4.3)$$

逻辑图如图 3.4.10(a)所示。

表 3.4.6 二进制码与格雷码的编码表

N	二进制码				格雷码			
	B ₃	B ₂	B ₁	B ₀	G ₃	G ₂	G ₁	G ₀
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

由式(3.4.3)可以写出每一位格雷码的一般表达式为

$$G_i = B_{i+1} \oplus B_i \quad (3.4.4)$$

式中, $i=0 \sim (n-1)$; 且令 $B_n=0$ 。

此式表明第 i 位的格雷码是二进制码第 i 位与第 $i+1$ 位的模二和。由此可构成 n 位二进制码至 n 位格雷码的转换电路, 如图 3.4.10(b)所示。

格雷码至二进制码的转换电路设计, 可以利用异或运算的交换律, 由式(3.4.3)可得

$$\left. \begin{aligned} B_3 &= G_3 \\ B_2 &= B_3 \oplus G_2 = G_3 \oplus G_2 \\ B_1 &= B_2 \oplus G_1 = G_3 \oplus G_2 \oplus G_1 \\ B_0 &= B_1 \oplus G_0 = G_3 \oplus G_2 \oplus G_1 \oplus G_0 \end{aligned} \right\} \quad (3.4.5)$$

由上式作出逻辑图示于图 3.4.10(c)。

顺便指出: 选用四 2 输入异或门 74LS86 能很方便地构成图 3.4.10 所示电路。

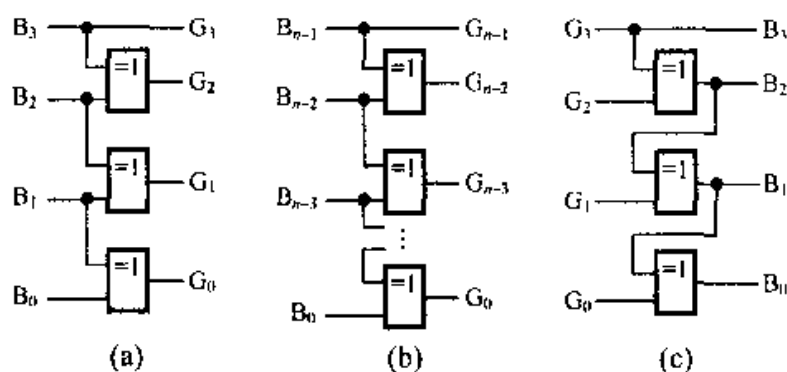


图 3.4.10 码制转换译码器

(a) 4 位二进制码至格雷码的转换连接图 (b) n 位二进制码至格雷码的转换连接图

(c) 4 位格雷码至二进制码的转换连接图

例 3.4.2 试用门电路设计一个可控码组变换器。其电路框图如图 3.4.11(a)所示, 要求当 $K=1$ 时, 将输入 4 位二进制码变换成 4 位格雷码输出; 当 $K=0$ 时, 将输入 4 位格雷码变换成 4 位二进制码输出。其编码表如表 3.4.6 所示。

题意分析 本例要求由控制信号 K 来实现 4 位二进制码与 4 位格雷码之间的可逆变换, 所以是具有 5 个输入变量和 4 个输出函数的组合逻辑电路。若采用传统“四步法”设计, 仅真值表就多达 32 行, 且过程十分繁琐。现在采用灵活设计法来设计就简便得多。用灵活设计法设计该电路的关键是仔细分析真值表, 找出输出函数与输入变量的特殊关系, 直接求得函数表达式。

解 1 分析真值表可以直接写出下列方程:

$$\left. \begin{aligned} G_3 &= B_3 \\ G_2 &= B_3 \oplus B_2 \\ G_1 &= B_2 \oplus B_1 \\ G_0 &= B_1 \oplus B_0 \end{aligned} \right\} \quad (3.4.6)$$

$$\left. \begin{aligned} B_3 &= G_3 \\ B_2 &= G_3 \oplus G_2 \\ B_1 &= G_3 \oplus G_2 \oplus G_1 \\ B_0 &= G_3 \oplus G_2 \oplus G_1 \oplus G_0 \end{aligned} \right\} \quad (3.4.7)$$

上列方程是这样得到的。因为在将二进制码变换为格雷码时， $B_3B_2B_1B_0$ 是输入变量， $G_3G_2G_1G_0$ 是输出函数， G_3 这一列同 B_3 那一列完全相同，所以有 $G_3=B_3$ ； G_2 这一列的函数值等于相应行的 B_3 与 B_2 异或的结果，所以有 $G_2=B_3 \oplus B_2$ ；同理可得 G_1 和 G_0 。在将格雷码变换为二进制码时， $G_3G_2G_1G_0$ 是输入变量，而 $B_3B_2B_1B_0$ 是输出函数。例如，求 B_0 ，表中 B_0 这一列的函数值都等于相应行的 4 个变量 $G_0 \sim G_3$ 的异或，所以得到

$$B_0 = G_3 \oplus G_2 \oplus G_1 \oplus G_0$$

同样分析可得 B_3 、 B_2 、 B_1 等的表达式。

当 $K=1$ 时，令 $X_3X_2X_1X_0=B_3B_2B_1B_0$

$$Y_3Y_2Y_1Y_0=G_3G_2G_1G_0$$

则由式 (3.4.6) 得到

$$\left. \begin{aligned} Y_3 &= X_3 \\ Y_2 &= X_3 \oplus X_2 \\ Y_1 &= X_2 \oplus X_1 \\ Y_0 &= X_1 \oplus X_0 \end{aligned} \right\} \quad (3.4.8)$$

当 $K=0$ 时，令 $X_3X_2X_1X_0=G_3G_2G_1G_0$ ， $Y_3Y_2Y_1Y_0=B_3B_2B_1B_0$ ，则由式 (3.4.7) 得到

$$\left. \begin{aligned} Y_3 &= X_3 \\ Y_2 &= X_3 \oplus X_2 \\ Y_1 &= X_3 \oplus X_2 \oplus X_1 \\ Y_0 &= X_3 \oplus X_2 \oplus X_1 \oplus X_0 \end{aligned} \right\} \quad (3.4.9)$$

将式(3.4.8)和式(3.4.9)合并为一个公式，则有

$$\left. \begin{aligned} Y_3 &= X_3 \\ Y_2 &= X_3 \oplus X_2 \\ Y_1 &= K(X_2 \oplus X_1) + \bar{K}(X_3 \oplus X_2 \oplus X_1) \\ Y_0 &= K(X_1 \oplus X_0) + \bar{K}(X_3 \oplus X_2 \oplus X_1 \oplus X_0) \end{aligned} \right\} \quad (3.4.10)$$

由式 (3.4.10) 画逻辑电路图, 如图 3.4.11(b) 所示。

解 2 步骤与解 1 相同, 不同之处在于实现式 (3.4.9) 的电路有差异。因为 $Y_2 = X_3 \oplus X_2$, 所以式 (3.4.9) 中 Y_1 可写成

$$Y_1 = K(X_2 \oplus X_1) + \bar{K}(Y_2 \oplus X_1)$$

又因为在 $K=0$ 时, $Y_1 = Y_2 \oplus X_1$, 所以

$$\begin{aligned} Y_0 &= K(X_1 \oplus X_0) + \bar{K}(X_3 \oplus X_2 \oplus X_1 \oplus X_0) \\ &= K(X_1 \oplus X_0) + \bar{K}(Y_1 \oplus X_0) \end{aligned}$$

在 Y_1 的表达式中, 当 K 为 1 时, $Y_1 = X_2 \oplus X_1$; 当 K 为 0 时, $Y_1 = Y_2 \oplus X_1$ 。其中, 输入变量 X_1 是共有的, 所以利用与或门的选通特性, 利用 K 和 \bar{K} 作选通信号, 就可以实现 Y_1 的表达式, Y_0 的表达式也可以用类似方法实现。其逻辑电路图如图 3.4.11(c) 所示。

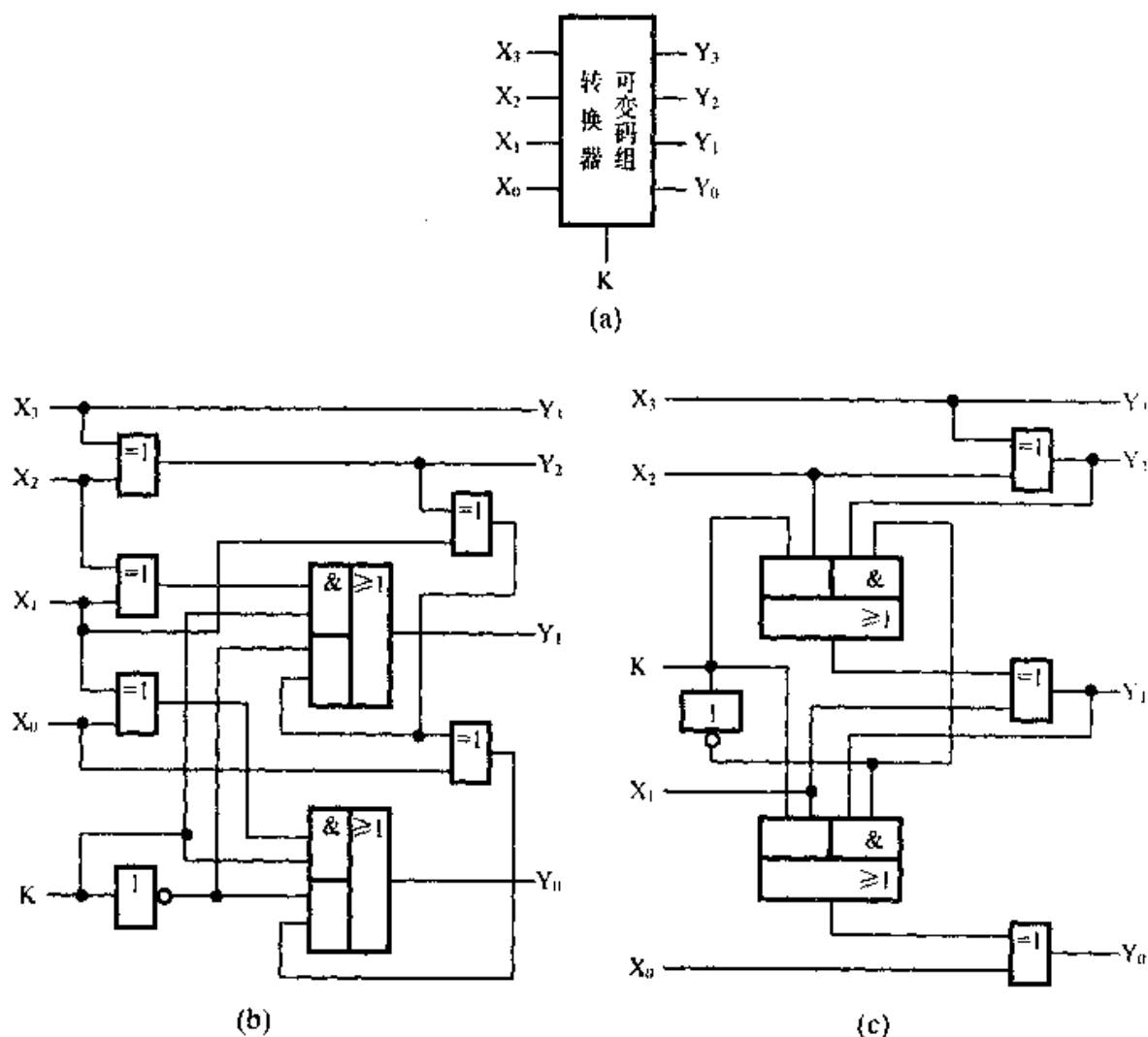


图 3.4.11 可控码组变换器

(a) 电路框图 (b) 逻辑电路图之一 (c) 逻辑电路图之二

3.4.4 中规模集成数字显示译码器

在数字技术中,经常需要把测量或运算的结果用十进制数码显示出来。实现这种功能的逻辑电路称为数字显示电路。

数字显示电路主要包含数字显示器和显示译码器两部分。

1. 数字显示器

数字显示器的种类很多,常见的有辉光数码管、荧光数码管、半导体数码管和液晶显示器等。在数字技术中广泛采用的是半导体数码管和液晶显示器两种,因为它们都是低电压电源供电。

(1) 半导体数码管

半导体数码管是一种7段字符显示器,它由7个发光二极管(LED)构成7个字段。共阴字形管是将7个LED的阴极连在一起,当阳极为高电平时,这个二极管发亮。数字的显示就是由发亮字段组合而成的,如图3.4.12所示。

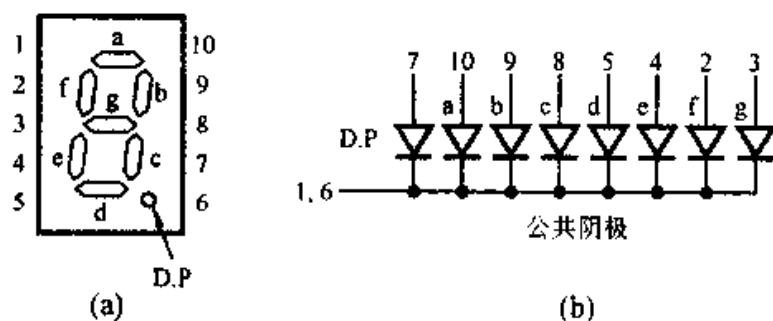


图 3.4.12 半导体数码管 BS201A

(a) 外形图 (b) 等效电路

共阳字形管是将7个LED的阳极连在一起并接入正电源,当某个阴极为低电平时,这个字段发亮,由发亮字段便可组成数字显示。

半导体数码管不仅具有工作电压低、寿命长、可靠性高等优点,而且响应速度快(小于 $0.1\mu\text{s}$),亮度也比较强。其缺点是工作电流较大,每段的工作电流约 10mA 。

(2) 液晶显示器

液晶显示器(简称LCD)的结构如图3.4.13所示。在两块玻璃片内表面用光刻的方法刻出电极:一块刻7个字段电极,另一块刻公共电极。两块玻璃板间距 $10\mu\text{m}$ 并在其中注入液晶材料,利用液晶材料的光电效应便制成了液晶显示器。7个字段和公共极

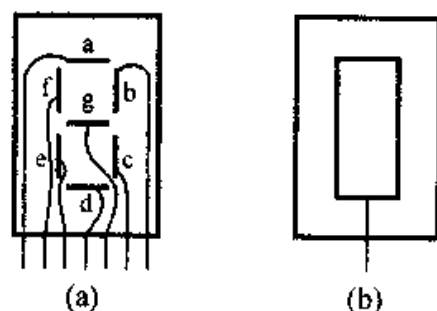


图 3.4.13 液晶显示器结构图

各有一根引出线,若某字段与公共极之间加有控制电压,则该字段便显示出来,并由显示字段组合而成数字显示,所以每个字段都需加一个驱动器。

驱动器采用交变电压驱动,对交变电压的控制可以用异或门实现,如图 3.4.14(a)所示。 v_i 是外加的对称方波电压,A是a字段的控制信号,当 $A=0$ 时,LCD 两端电压 $v_L=0$,A 字段不发亮;当 $A=1$ 时,LCD 两端电压 v_L 为幅度等于 v_i 两倍的对称方波,a 字段发亮。各点工作电压波形示于图 3.4.14(b)。

液晶显示器的优点是工作电压低、功耗极小,每平方厘米的功耗在 $1\mu\text{W}$ 以下。因此,在电子手表及小型便携式仪表中获得广泛应用。它的缺点是亮度差,响应速度较低 ($10\sim 200\text{ms}$)。

2. 中规模集成显示译码器

十进制数一般都以 BCD 码形式出现,因此,就需要使用显示译码器将 BCD 码译成 7 段字符显示器所需要的 7 个驱动信号。

驱动上述两类 LED 7 段数码管有相应的两类 4-7 线译码器。驱动共阴型的典型产品有 74LS48、74LS248、CC14513、CC14547 等;驱动共阳型的典型产品有 74LS47、74LS247 等。这些产品一般都带有驱动器,可以直接驱动 LED 7 段数码管。

图 3.4.15(a)是 BCD-7 段译码器(4-7 线译码器)74LS47 的逻辑电路图,图(b)是逻辑框图。该组件直接驱动共阳型 LED 7 段数码管(如 SB204)便可构成数字显示,即输出 $\overline{Q_3}\sim\overline{Q_6}$ 分别驱动数码管的 $\overline{a}\sim\overline{g}$ 段。

$A_3\sim A_0$ 是 BCD 码输入端; $\overline{Q_3}\sim\overline{Q_6}$ 是输出端,低电平有效(相应字段亮)。 \overline{LT} 是灯测试信号输入端; \overline{RBI} 为灭零输入端。 $\overline{BI}/\overline{RBO}$ 为熄灭输入/灭零输出端。以上 3 个信号均为低电平有效。

组件的功能表如表 3.4.7 所示。下面主要讨论几个辅助控制端的功能和使用方法。

\overline{LT} 为灯测试信号输入端,用来检查数码管的 7 个字段是否正常。此时,应令 $\overline{LT}=0$ 和 $\overline{BI}/\overline{RBO}=1$ (熄灭输入无效),若数码管的 7 个字段都发亮,显示 H ,则表明数码管正常。

\overline{RBI} 为灭零输入端,用来熄灭不需要显示的 0。因为在多位数字显示系统中,

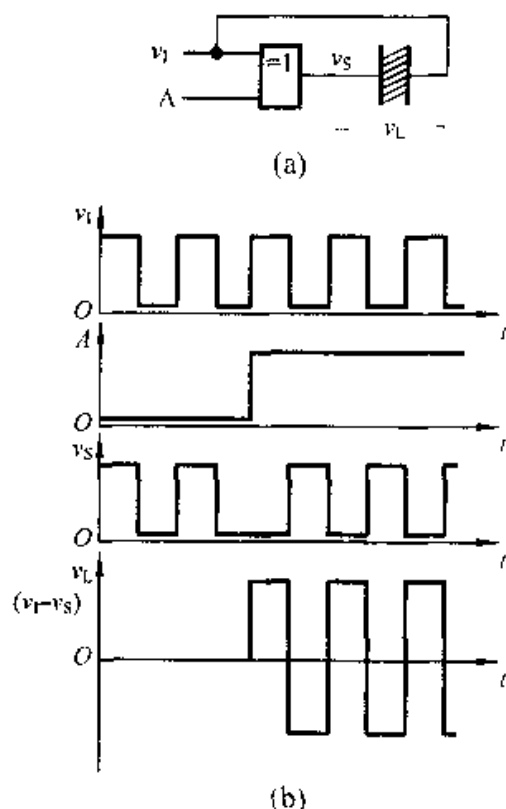


图 3.4.14 用异或门驱动液晶显示器

(a) 电路 (b) 电压波形

可以将有效数字前、后多余的 0 熄灭, 这样既便于读数, 又可减少功耗。此时, 应令 $\overline{LT}=1$, $\overline{BI}/\overline{RBO}=1$, $\overline{RBI}=0$, 则当 $A_3A_2A_1A_0=0000$ 时, 数码管不会显示 0。

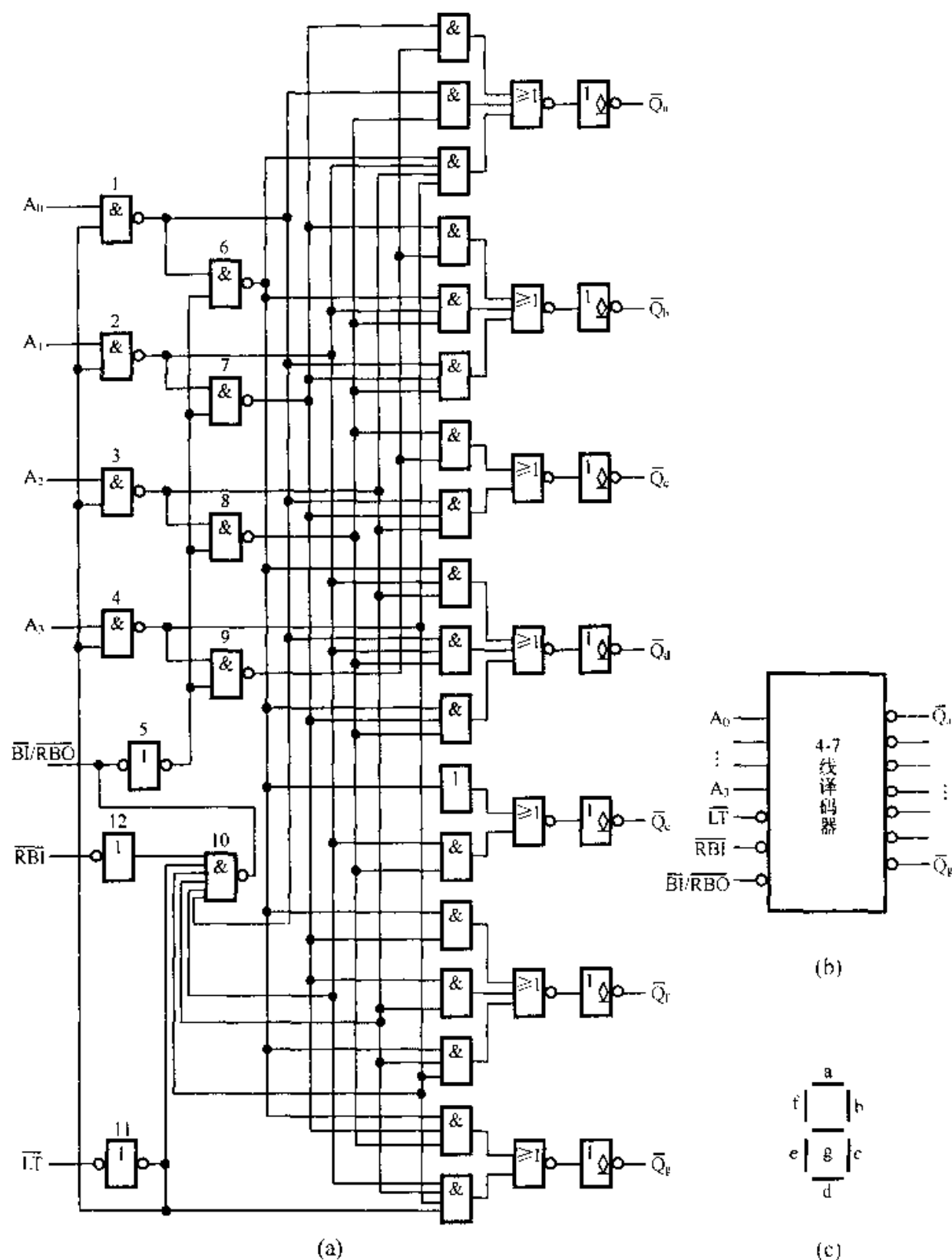


图 3.4.15 74LS47 型 BCD-7 段译码器

(a) 逻辑电路图 (b) 逻辑框图 (c) 7 段字形

表 3.4.7 74LS47 功能表

输 入						$\overline{\text{BI}}/\overline{\text{RBO}}$	输 出							显示字符
$\overline{\text{LT}}$	$\overline{\text{RBI}}$	A_3	A_2	A_1	A_0		$\overline{Q_a}$	$\overline{Q_b}$	$\overline{Q_c}$	$\overline{Q_d}$	$\overline{Q_e}$	$\overline{Q_f}$	$\overline{Q_g}$	
0	x	x	x	x	x	1	0	0	0	0	0	0	0	测试
1	0	0	0	0	0	0	1	1	1	1	1	1	1	灭零
x	x	x	x	x	x	0	1	1	1	1	1	1	1	熄灭
1	x	0	0	0	0	1	0	0	0	0	0	0	1	0
1	x	0	0	0	1	1	1	0	0	1	1	1	1	1
1	x	0	0	1	0	1	0	0	1	0	0	1	0	2
1	x	0	0	1	1	1	0	0	0	0	1	1	0	3
1	x	0	1	0	0	1	1	0	0	1	1	0	0	4
1	x	0	1	0	1	1	0	1	0	0	0	0	0	5
1	x	0	1	1	0	1	1	1	0	0	0	0	0	6
1	x	0	1	1	1	1	0	0	0	1	1	1	1	7
1	x	1	0	0	0	1	0	0	0	0	0	0	0	8
1	x	1	0	0	1	1	0	0	0	1	1	0	0	9

$\overline{\text{RI}}/\overline{\text{RBO}}$ 为熄灭输入/灭零输出端，它有两个功能：其一，在 $\overline{\text{RI}}/\overline{\text{RBO}}$ 端加入低电平，则不论其他输入状态如何，数码管熄灭；其二，若令 $\overline{\text{LT}}=1$ 、 $\overline{\text{RBI}}=0$ ， $A_3A_2A_1A_0=0000$ ，则有 $\overline{\text{BI}}/\overline{\text{RBO}}=0$ ，即 $\overline{\text{BI}}/\overline{\text{RBO}}$ 端输出低电平，此低电平可以加至另一片同类组件的灭零输入端而实现两片同时灭零功能。

如果在组件输入端 $A_3\sim A_0$ 出现非法码组 1010~1111，则显示管会显示出一些特殊符号，借助于它们可以判定输入 BCD 码是否发生错误。

利用该组件的灭零输入端 $\overline{\text{RBI}}$ 和灭零输出端 $\overline{\text{RBO}}$ 的功能，将它们配合使用便可实现多位数码显示系统的灭零控制。例如，一个整数部分为 5 位，小数部分为 3 位的 8 位数码显示系统的灭零控制的连接方法示于图 3.4.16。在整数部分把高位的灭零输出端 $\overline{\text{RBO}}$ 与低位的灭零输入端 $\overline{\text{RBI}}$ 相连；在小数部分把低位的灭零输出端 $\overline{\text{RBO}}$ 与高位的灭零输入端 $\overline{\text{RBI}}$ 相连。这样，就可以把无效的零熄灭掉，因为整数部

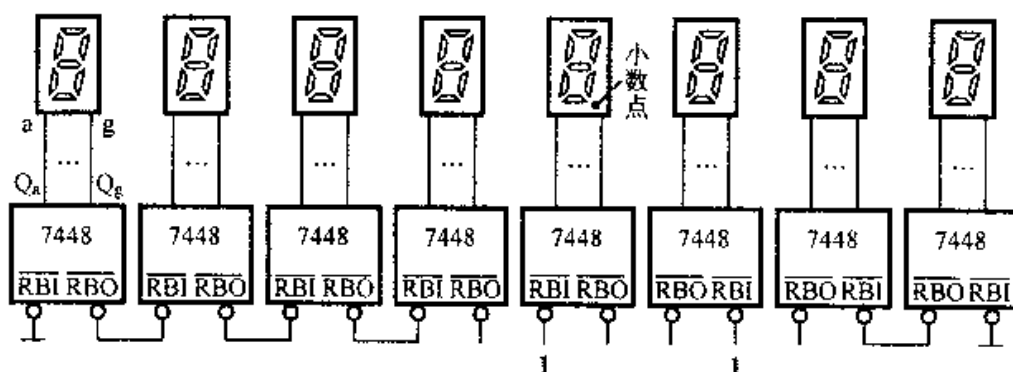


图 3.4.16 有灭零控制的 8 位数码显示系统

分只有在高位是零而且被熄灭了的条件下,低位才有灭零输入信号,若低位也是零,则同时被熄灭。同理,小数部分只有在低位是零而且被熄灭的条件下,高位才有灭零输入信号。例如,输入数码(BCD 码输入)为 00068.900,则该系统实际显示为 68.9;若输入数码为 01068.909,则该系统实际显示为 1068.909。熄灭无效零的工作过程,读者可参照图 3.4.16 进行分析(7448 的输出 $Q_5 \sim Q_8$ 是高电平有效)。

3.5 二进制运算电路

完成二进制数的加、减、乘和除法等算术运算及二进制数的累加或数值比较等功能的逻辑电路统称为二进制运算电路,简称为运算电路或运算器。本节以加法器和数值比较器为例来讨论这类电路的设计方法。

3.5.1 半加器

半加器是完成两个 1 位二进制数相加求和及进位的逻辑电路,它不考虑低位来的进位。采用灵活设计法可以直接写出函数表达式。因为只有在这两个二进制数不同时,和数 F' 为 1;只有在它们均为 1 时才产生向高位进位,即 $C'_0=1$,所以

$$\left. \begin{aligned} F' &= A \oplus B \\ C'_0 &= AB \end{aligned} \right\} \quad (3.5.1)$$

用两级与非门设计半加器的逻辑电路如图 3.5.1(a)所示;和数 F' 为 A 与 B 的异或函数(模二和),所以用异或门设计比较方便,如图 3.5.1(b)所示。

如果无反变量输入,又指定采用与非门实现,则可用并项-代替因子法对 F' 表达式进行变换,即

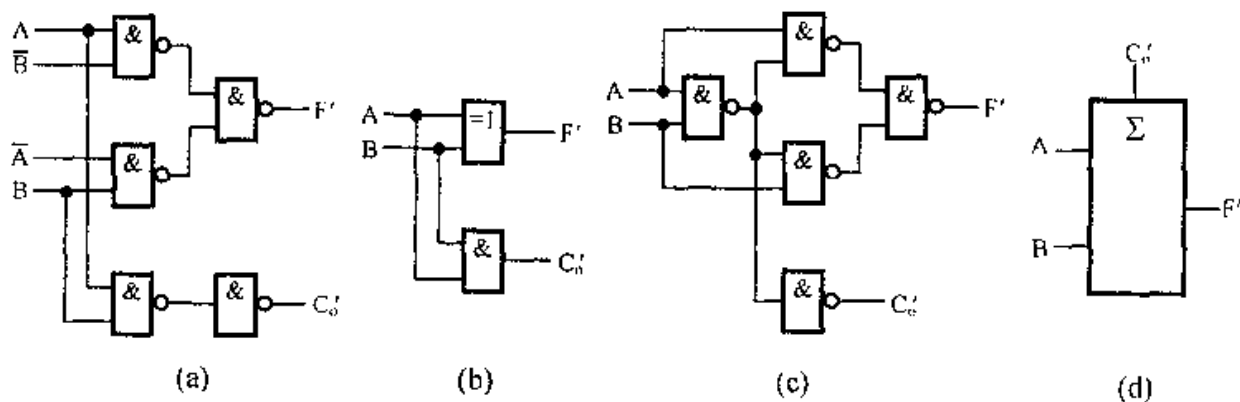


图 3.5.1 半加器逻辑电路图和逻辑框图

- (a) 两级与非结构的半加器逻辑电路图 (b) 异或门构成的半加器逻辑电路图
(c) 3 级与非结构的半加器逻辑电路图 (d) 逻辑符号

$$F' = A\bar{B} + \bar{A}B = A\bar{A}B + \bar{A}B\bar{B} = \overline{AAB \cdot BAB}$$

用 3 级与非门实现半加器的逻辑电路图, 如图 3.5.1(c)所示。图 3.5.1(d)是半加器的逻辑符号。

3.5.2 全加器

全加器是指在两个二进制数相加时, 要考虑低一位来的进位数 C_i , 所以, 它是 3 个 1 位二进制数进行相加“求和”及“进位”的逻辑电路。

用灵活设计法设计(请读者用四步法设计, 并比较设计结果)全加器。因为 3 个二进制数相加, 若其中取值为 1 者是奇数, 则其“和数” F 必为 1; 若其中任意两个取值为 1, 则“进位” C_o 为 1。由此可得

$$\left. \begin{aligned} F &= A \oplus B \oplus C_i \\ C_o &= AB + AC_i + BC_i \end{aligned} \right\} \quad (3.5.2)$$

若输入提供原、反变量, 则用与非门实现的两级逻辑门全加器电路如图 3.5.2(a)所示。

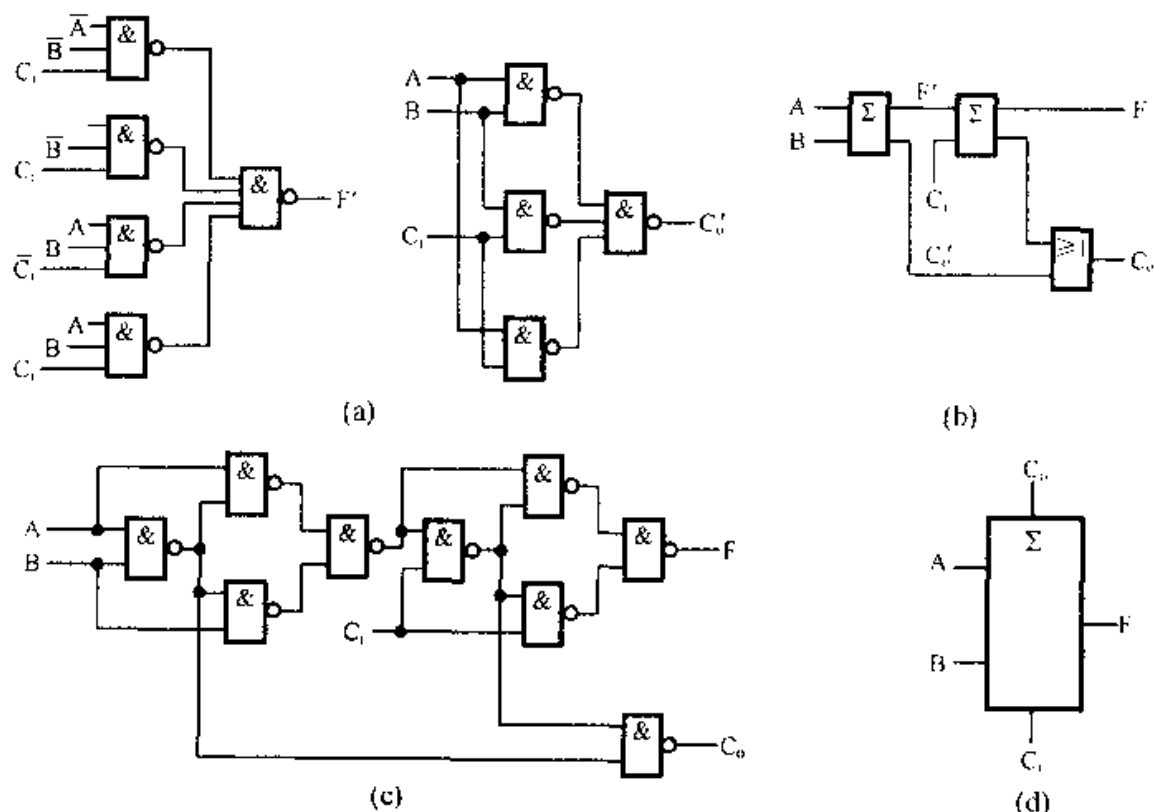


图 3.5.2 全加器逻辑电路图和逻辑符号

- (a) 两级与非结构的全加器逻辑电路图 (b) 半加器构成的全加器逻辑电路图
(c) 6 级与非结构的全加器逻辑电路图 (d) 逻辑符号

若用半加器设计电路, 则对式(3.5.2)进行变换, 即

$$F = A \oplus B \oplus C_i = F' \oplus C_i$$

$$\begin{aligned} C_o &= AB + AC_i + BC_i = AB + AC_i(B + \bar{B}) + BC_i(A + \bar{A}) \\ &= AB + (A\bar{B} + \bar{A}B)C_i = C_o' + F'C_i \end{aligned} \quad (3.5.3)$$

用半加器构成全加器的逻辑电路如图 3.5.2(b)所示。

若输入仅提供原变量，并且指定采用与非门实现，则可用图 3.5.1(c)所示电路代替图 3.5.2(b)中的半加器，即可得 6 级与非门构成的全加器，如图 3.5.2(c)所示。全加器逻辑符号示于图 3.5.2(d)。

3.5.3 中规模集成 4 位加法器

完成两个并行 4 位二进制数相加求和及其进位的逻辑电路称为 4 位加法器。其电路结构有串行进位和超前(快速)进位两类。

串行进位 4 位加法器由 4 个全加器(1 位)通过进位信号串接而成，所以，这类电路是由低位至高位依次逐位相加的，完成一次相加的操作时间长，即操作速度慢。

超前进位 4 位加法器是同时产生各位的进位信号，所以，各位可以同时进行相加运算，即操作速度快。

1. 4 位超前进位全加器

该组件的各位全加器电路结构类似，仅各位的进位信号方程有差异，由图 3.5.3(a)可写出各位进位方程。为此，令

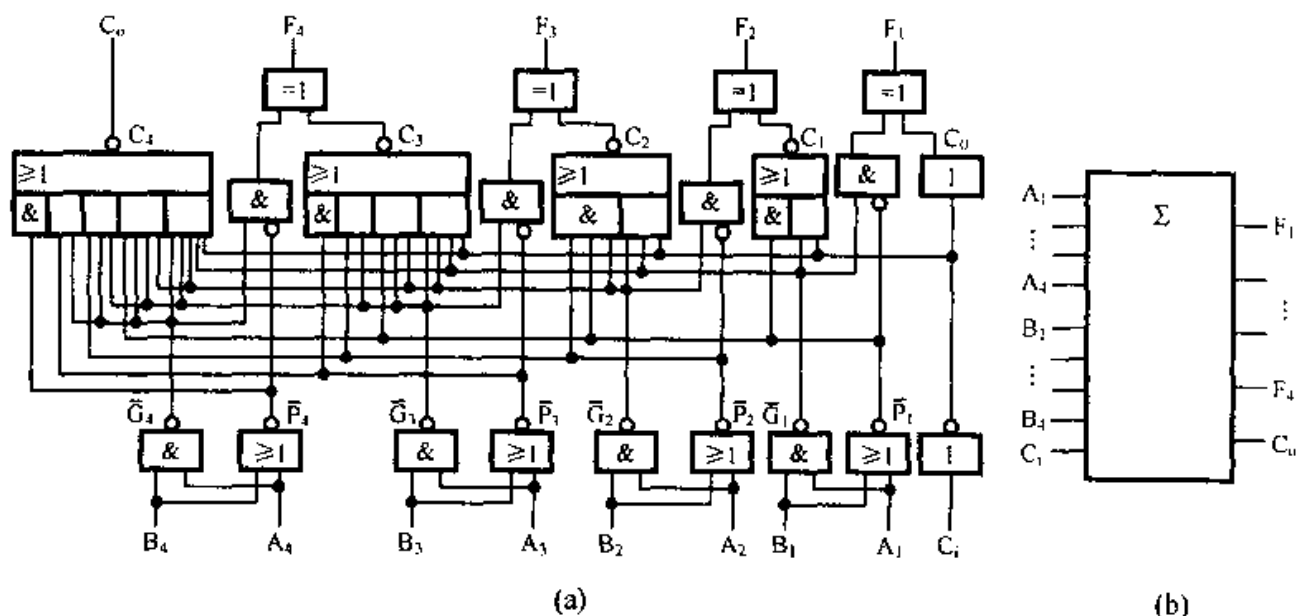


图 3.5.3 4 位超前进位全加器

(a) 逻辑电路图 (b) 逻辑框图

进位产生项 $G_i = A_i B_i$
 进位传输项 $P_i = A_i + B_i$
 则

$$G_i P_i = A_i B_i (A_i + B_i) = G_i$$

由此可写出各位进位方程为

$$\left. \begin{aligned} C_0 &= \overline{C_i} = C_i \\ C_1 &= \overline{A_1 B_1 C_0} + \overline{A_1 + B_1} = \overline{G_1 C_0} + \overline{P_1} = G_1 + P_1 C_0 \\ C_2 &= \overline{G_2 \overline{G_1 C_0} + G_2 \overline{P_1} + P_2} = \overline{G_2 C_0} + \overline{P_2} = G_2 + P_2 C_1 \\ \text{同理, } C_3 &= G_3 + P_3 C_2 \\ C_4 &= G_4 + P_4 C_3 \end{aligned} \right\} \quad (3.5.4a)$$

各位求和方程为

$$\left. \begin{aligned} F_1 &= \overline{P_1} \overline{G_1} \oplus \overline{C_0} = P_1 \overline{G_1} \oplus C_0 = A \oplus B \oplus C_0 \\ F_2 &= A_2 \oplus B_2 \oplus C_1 \\ F_3 &= A_3 \oplus B_3 \oplus C_2 \\ F_4 &= A_4 \oplus B_4 \oplus C_3 \end{aligned} \right\} \quad (3.5.4b)$$

在图 3.5.3 和图 3.5.4 所示全加器中, C_i 为低位单元来的进位信号, 也就是本单元最低位全加器的进位信号 C_0 ; C_4 为本单元进位输出信号, 以提供多片级联应用的串行进位信号。

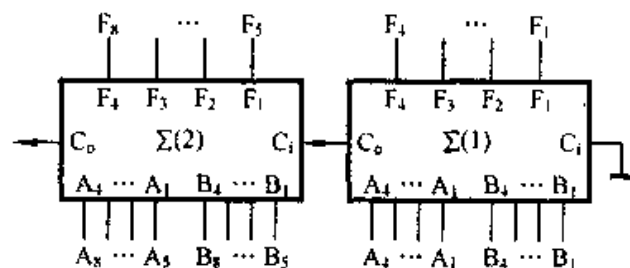


图 3.5.4 8 位并串型进位全加器连接图

图 3.5.4 是用两片 4 位超前进位全加器构成 8 位并串型进位全加器, 并串型进位表明组件内是超前进位, 而组件之间是串行进位。

2. 用全加器设计组合逻辑电路

4 位全加器也是通用性很强的 MSI 组件, 除了可进行二进制数的加法运算之外, 还在 BCD 码的原码、反码及补码变换、码制转换、二进制加减运算电路等领域有着广泛用途。下面仅举几例说明它在这些领域中的应用。

(1) 带符号二进制数的求补电路

设 $D_7 \sim D_0$ 为带符号二进制数, D_7 为符号位。若 D_7 为 0, 则二进制数为正数,

反之,则为负数。用两片4位全加器及异或门构成带符号二进制数的求补电路如图3.5.5所示。

当 $D_7=0$ 时,送入 A_n 端的是 D_n 的原码,且有 $C_i=0$,所以 $F_n=A_n$ 。

当 $D_7=1$ 时,表示二进制数为负数。送入 A_n 端的是 \bar{D}_n ,且有 $C_i=1$,所以全加器实现 $A(7$ 位二进制数)的反码加 1 功能。

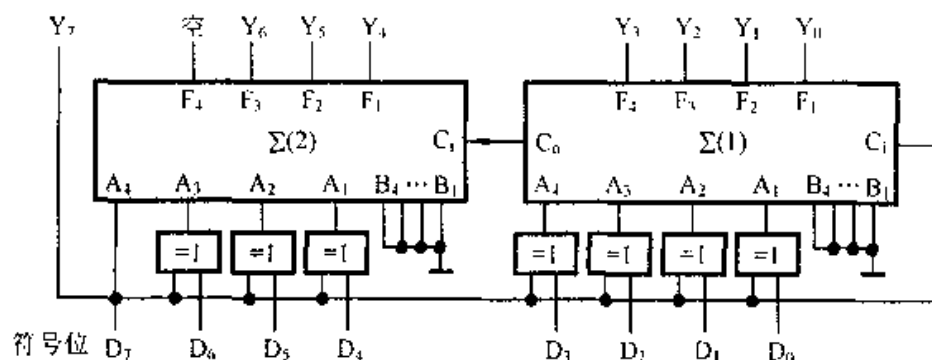


图 3.5.5 4 位全加器构成带符号二进制数的求补电路

(2) 码制转换电路

用 4 位全加器能方便地实现各种 BCD 码之间,或者 BCD 码与二进制码之间的转换。图 3.5.6 是 8421BCD 码至余 3 码的转换电路。因为某数的余 3 码等于该数的 8421BCD 码加 $(3)_{10}$, 所以将 8421BCD 码从 $A_1 \sim A_4$ 输入, 令 $B_4 B_3 B_2 B_1 = 0011$, $C_i = 0$, 则输出 $F_4 F_3 F_2 F_1 = A_4 A_3 A_2 A_1 + 0011$, 即为余 3 码输出。图 3.5.7 是余 3 码至 8421BCD 码的转换电路。因为某数的余 3 码减 $(3)_{10}$ 即等于该数的 8421BCD 码。利用全加器将减法运算变为补码加法运算, 即 $(0011)_2 = 1101$, 所以, 输出 $F_4 F_3 F_2 F_1 = A_4 A_3 A_2 A_1 + 1101$, 因此, 将余 3 码从 $A_1 \sim A_4$ 输入, 且令 $C_i = 0$, 在全加器的输出端 $F_4 \sim F_1$ 便得到 8421BCD 码。

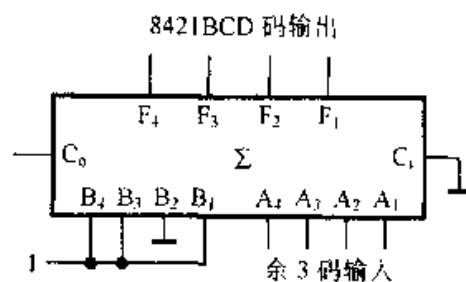
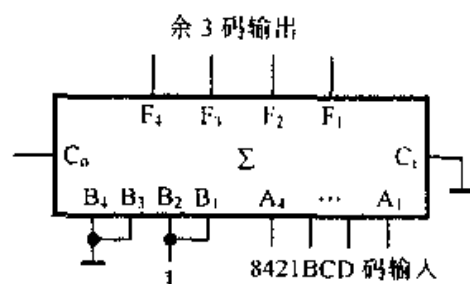


图 3.5.6 8421BCD 码至余 3 码的转换连接图 图 3.5.7 余 3 码至 8421BCD 码的转换连接图

3.5.4 数值比较器

1. 数值比较器的设计

数值比较器具有判别两个二进制数大小的逻辑功能。例如, 对两个二进制数 A

和 B 进行比较, 可以给出 3 种比较结果: $A > B$; $A = B$; $A < B$ 。两个二进制数 A 和 B 可以由 1 位或多位二进制数码组成。由于 1 位数值比较器是多位数值比较器的基本单元, 所以, 先用组合电路设计方法求得 1 位数值比较器的方程为

$$\left. \begin{aligned} F_{A>B} &= \overline{A}B = \overline{AAB} \\ F_{A=B} &= A \odot B = \overline{AAB + BAB} \\ F_{A<B} &= \overline{AB} = \overline{BAB} \end{aligned} \right\} \quad (3.5.5)$$

式(3.5.5)是利用逻辑运算的某些特点直接写出的, 因为 $A > B$ 即表示 A 为 1、 B 为 0; $A = B$ 表示 A 和 B 的逻辑值相同($AB=00$ 或 $AB=11$); $A < B$ 表示 A 为 0、 B 为 1。1 位数值比较器逻辑电路如图 3.5.8 所示。

在 1 位数值比较器的基础上可以构成 4 位数值比较器, 由于输入变量增多, 用传统的组合电路设计方法设计就非常繁琐, 但如果采用灵活设计法来设计就很简便。

当用 4 位数值比较器对两个 4 位二进制数进行比较判别时, 应该首先进行最高位比较, 若能判别最高位逻辑值有大小之分, 则组件便应有确定的输出, 即 $F_{A>B}$ 为 1 或者 $F_{A<B}$ 为 1。如果最高位逻辑值相等, 则再进行次高位比较。如此类推, 从高到低逐位依次进行, 直到组件给出确定的输出为止。按照这种设计思想便可以设计中规模集成 4 位数值比较器了。

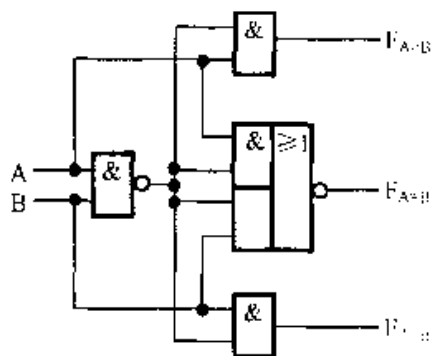


图 3.5.8 一位数值比较器逻辑电路

2. 中规模集成 4 位数值比较器

图 3.5.9(a)是 74LS85 的逻辑电路图, 图(b)为逻辑符号。图中, 设置有 3 个级联输入端 $G(A > B)$ 、 $E(A = B)$ 、 $S(A < B)$ 。只有在被比较的 4 位数码都相等时, 级联输入端的数据才能直接传送到相应输出端, 在组件级联应用中, 灵活地选择级联输入端的连接方式或者逻辑电平, 可以扩展数值比较器的位数, 一般称之为扩展字长。

由图 3.5.9(a)可写出

$$\left. \begin{aligned} F_{A>B} &= \overline{A}_3B_3 + \overline{A}_2B_2P_3 + \overline{A}_1B_1P_3P_2 + \overline{A}_0B_0P_3P_2P_1 + P_3P_2P_1P_0S + P_3P_2P_1P_0E \\ F_{A=B} &= P_3P_2P_1P_0E \\ F_{A<B} &= A_3\overline{B}_3 + A_2\overline{B}_2P_3 + A_1\overline{B}_1P_3P_2 + A_0\overline{B}_0P_3P_2P_1 + P_3P_2P_1P_0G + P_3P_2P_1P_0E \end{aligned} \right\} \quad (3.5.6)$$

若 $A_3=1$, $B_3=0$, 则由于 $P_3=A_3 \odot B_3=0$, 所以

$$F_{A>B}=1 \quad F_{A=B}=0 \quad F_{A<B}=0$$

若 $A_3=0$, $B_3=1$, 则有

$$F_{A>B}=0 \quad F_{A=B}=0 \quad F_{A<B}=1$$

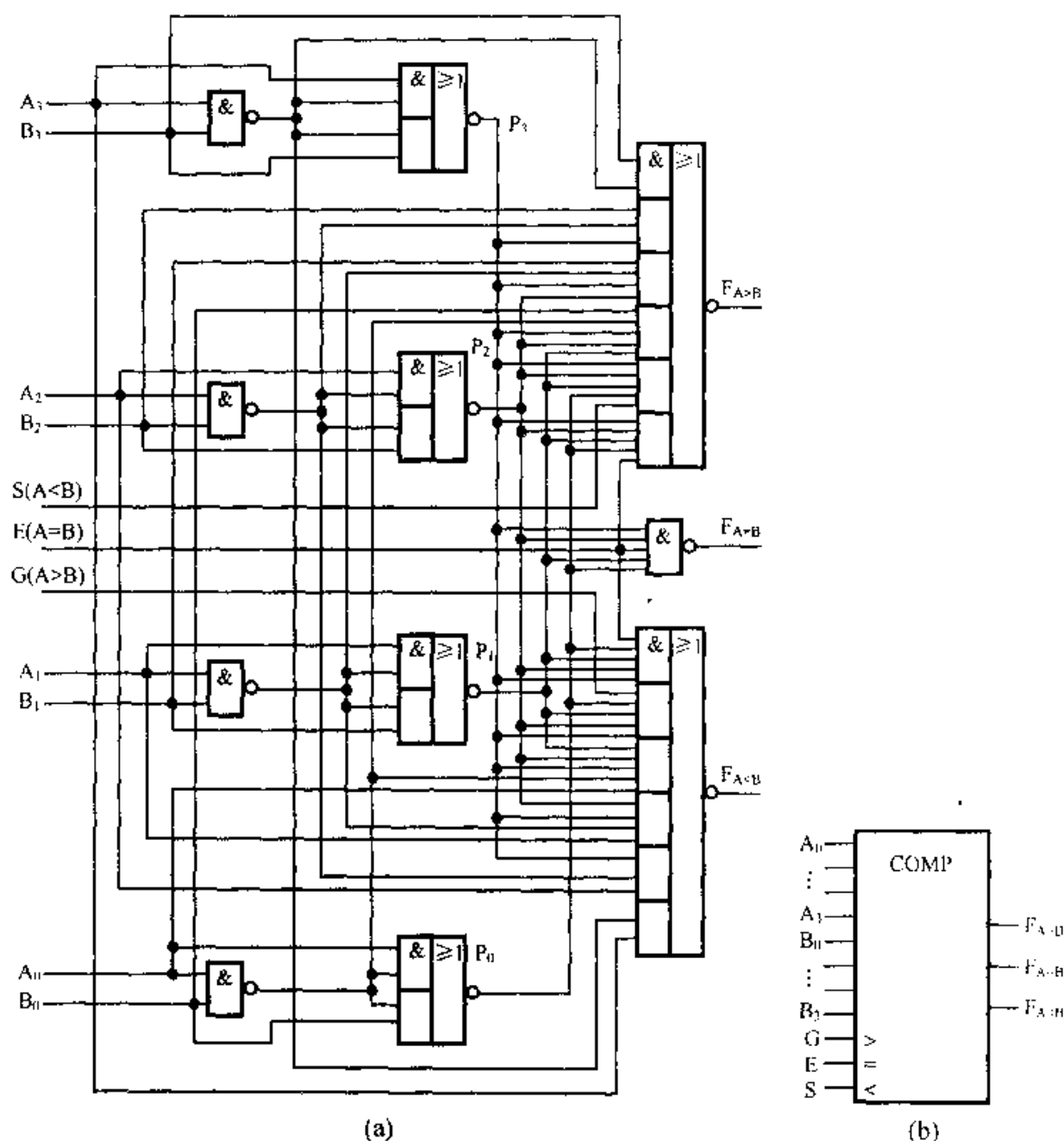


图 3.5.9 4 位数值比较器

(a) 逻辑电路图 (b) 逻辑框图

可见, 若最高位 $A_3 \neq B_3$, 则组件有确定输出。若 $A_3 = B_3$, 则 $P_3 = 1$, 并且 $\bar{A}_3 B_3 = A_3 \bar{B}_3 = 0$ 。所以, 用类似方法进行次高位 A_2 与 B_2 的比较。如果 A 和 B 相等, 即 $A_3 = B_3$, $A_2 = B_2$, $A_1 = B_1$, $A_0 = B_0$, 则有 $P_3 = P_2 = P_1 = P_0 = 1$, 则式(3.5.6)成为

$$\left. \begin{aligned} F_{A>B} &= \overline{P_3 P_2 P_1 P_0} S + P_3 P_2 P_1 P_0 E \\ F_{A=B} &= E \\ F_{A<B} &= \overline{P_3 P_2 P_1 P_0} G + P_3 P_2 P_1 P_0 E \end{aligned} \right\} \quad (3.5.7)$$

由上式可知: 在 $A=B$ 时, 若级联输入端 $E=1$, 则有 $F_{A=B}=1$, $F_{A>B}=F_{A<B}=0$, 此

时级联输入端 S 和 G 的数据对输出没有影响；在 $A=B$ 时，若 $E=0$ ，则有 $F_{A=B}=0$ ， $F_{A>B}=\overline{S}$ ， $F_{A<B}=\overline{G}$ 。在 $A=B$ 时，3 个级联输入端有 8 种输入状态，输出端都有确定的逻辑值。

4 位数值比较器的功能表如表 3.5.1 所示。由表可见：若最高位 $A_3 \neq B_3$ ，则组件有确定的输出；若 $A_3=B_3$ ，则继续进行次高位的比较，如此类推。表中前 8 行表示本组件的 4 位数码比较可以得出确定的结果($A>B$ 或 $A<B$)，所以，级联输入信号不起作用；表中后 3 行表示本组件比较的 4 位数码完全相等的条件下，级联输入端的数据同输出逻辑值的关系。

表 3.5.1 4 位数值比较器功能表

输 入						级联输入			输 出					
A_3	B_3	A_2	B_2	A_1	B_1	A_0	B_0	$G(>)$	$S(<)$	$E(=)$	$F_{A>B}$	$F_{A<B}$	$F_{A=B}$	
$A_3>B_3$		\times	\times	\times	\times	\times	\times	\times	\times	\times	1	0	0	
$A_3<B_3$		\times	\times	\times	\times	\times	\times	\times	\times	\times	0	1	0	
$A_3=B_3$		$A_2>B_2$		\times	\times	\times	\times	\times	\times	\times	1	0	0	
		$A_2<B_2$		\times	\times	\times	\times	\times	\times	\times	0	1	0	
		$A_2=B_2$		$A_1>B_1$		\times	\times	\times	\times	\times	\times	1	0	0
				$A_1<B_1$		\times	\times	\times	\times	\times	\times	0	1	0
				$A_1=B_1$		$A_0>B_0$		\times	\times	\times	\times	1	0	0
						$A_0<B_0$		\times	\times	\times	\times	0	1	0
						$A_0=B_0$		1	0	0	1	0	0	
								0	1	0	0	1	0	
								0	0	1	0	0	1	
								\times	\times	1	0	0	1	
								1	1	0	0	0	0	
								0	0	0	1	1	0	

必须指出：目前生产的 4 位数值比较器产品中，不同类型或不同系列的产品采用的电路结构不同，因此，级联输入端的用法也不完全一样，应依据功能表正确地使用级联输入端。

将几片 4 位数值比较器级联可以扩展字长，可采用串联方式，也可以采用并联方式进行级联。图 3.5.10 是用串联方式构成的 12 位数值比较器的连接图。低位组件的比较输出直接送到高一位的组件级联输入端，而最低位组件(1)的级联输入端接相应的逻辑电平，以便在 12 位数码都是相等的条件下，获得 12 位数值比较器的正确的比较结果。为此，组件(1)的级联输入端 $G(A>B)$ 和 $S(A<B)$ 可接 0 电平； $E(A=B)$ 端应接 1 电平。

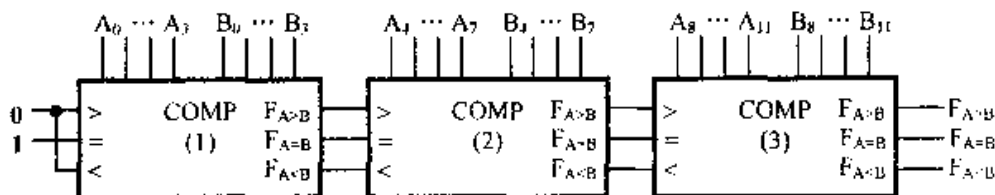


图 3.5.10 串联式 12 位数值比较器

图 3.5.11 是并联式 12 位数值比较器的连接图。它由 4 片 4 位数值比较器构成，其中，组件(1)~(3)构成输入级，对 12 位数码进行比较，比较结果送到由组件(4)构成的输出级进行比较，组件(4)此时改接为 2 位数值比较器使用，并将 12 位数码比较的最后结果输出。必须注意，各组件级联输入端的连接方式应该正确处理，否则最后的判别结果会发生差错。组件(1)和(2)的级联输入端可以接相同的逻辑电平(0 或 1 电平)；组件(3)的 G 和 S 端必须接 0 电平，而 E 端必须接 1 电平。

从以上两例可以发现级联输入端的设置使组件的通用性增强了，但是，它们的连接方式应依据应用情况正确处理。

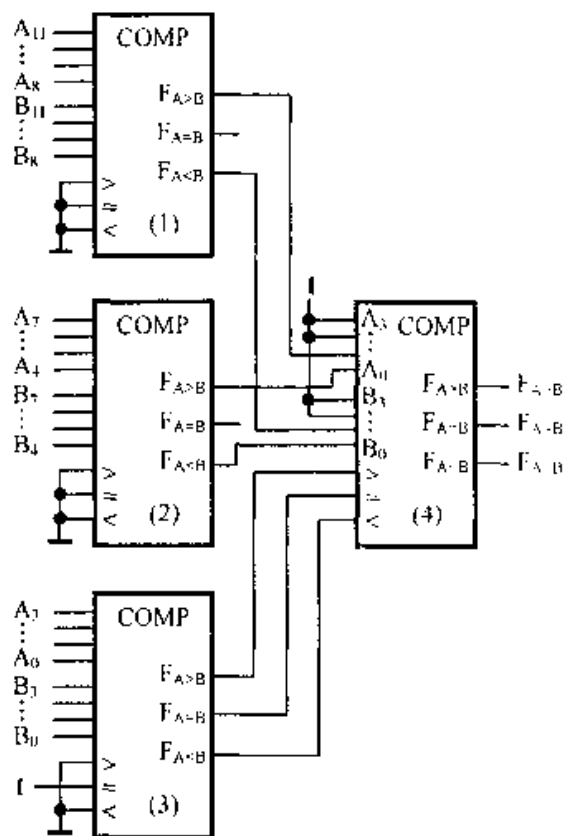


图 3.5.11 并联式 12 位数值比较器

3.6 数据选择器

数据选择器(多路选择器、多路器、多路开关)的功能是从多路输入数据中选择一路数据输出。它可以用来完成并行码输入、串行码输出的转换；也可作 n 线至 1 线的转换器；还可用它设计各种逻辑电路以及实现逻辑函数。它是应用非常广泛的一种中规模集成组件。

3.6.1 数据选择器的类型及主要性能

4 选 1 数据选择器框图示于图 3.6.1。图中， $D_0 \sim D_3$ 为数据输入端，又称为输入通

道；A、B 为地址输入端，又称为选择输入端，A 和 B 的每一组代码称为地址，每一个地址选择一路对应的数据输出；W 为数据输出端。

4 选 1 数据选择器的逻辑表达式为

$$W = \bar{A}\bar{B}D_0 + \bar{A}BD_1 + A\bar{B}D_2 + ABD_3 \quad (3.6.1)$$

中规模数据选择器都设置有一个或多个使能端，以便扩展功能。同时，还具有多种输出方式和结构，如原码、反码或原/反码互补输出；集电极开路(OC)输出及三态(3S)输出结构等。各种类型组件中又有品种繁多的系列产品供用户选用。表 3.6.1 列出各类型组件的典型产品型号及主要性能。

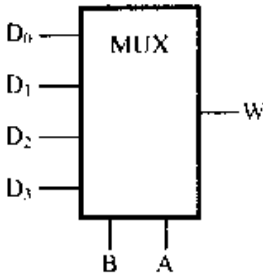


图 3.6.1 4 选 1 数据选择器框图

表 3.6.1 数据选择器典型产品型号及主要性能

名 称	型 号	类 型	输 出	延迟时间/ns	功耗/mW (电源电流/电压)
四 2 选 1 数据选择器	74LS157	TTL	原码	5.8	49
	74LS158	TTL	反码	7.12	24
	74LS257	TTL	原码(3S)	12.2	60
	74LS258	TTL	反码(3S)	12.2	60
	CE10158	ECL	原码	2.5~3.2	197
	CE10159	ECL	反码	2.5~3.2	197
	CC74HC257	CMOS	原码	12	(80μA/5V)
双 4 选 1 数据选择器	74LS153	TTL	原码	14.17	35
	74LS353	TTL	反码	12.21	43
	CE10174	ECL	原码	3.5	305
	CC14529	CMOS	原码	210	(5μA/5V)
	CC74HC253	CMOS	原码(3S)	24	(5μA/5V)
8 选 1 数据选择器	74LS151	TTL	原/反码	11	30
	74LS251	TTL	原/反码(3S)	21	35
	CE10164	ECL	原码	3	310
	CC4512	CMOS	原码	200	(5μA/5V)
	CC74HC251	CMOS	原码(3S)	27	(80μA/5V)
16 选 1 数据选择器	74150	TTL	反码	11	200

1. 双 4 选 1 数据选择器

双 4 选 1 数据选择器 74LS153 的逻辑电路如图 3.6.2(a)所示, 图(b)是逻辑框图。该组件含有两个相同的 4 选 1 数据选择器, 共用一对地址输入线 A_1, A_0 , 各自设置有使能输入端 $\bar{1S}$ 或 $2\bar{S}$ 。由图 3.6.2(a)电路可以写出

$$W = (\bar{A}_1 \bar{A}_0 D_0 + \bar{A}_1 A_0 D_1 + A_1 \bar{A}_0 D_2 + A_1 A_0 D_3) \bar{1S} \quad (3.6.2)$$

式中, 未标系数, 是两个 4 选 1 数据选择器的通用表达式。由此作出功能表 3.6.2。由表可见: 当 $\bar{1S}=1$ 时, 组件为禁止状态, W 为 0 (与其他输入信号无关); 当 $\bar{1S}=0$ 时, 由地址码 A_1 和 A_0 的一组代码选择相应输入信号输出。

74LS253 的电路与图 3.6.2(a)类似, 不同之处在于输出采用三态结构, 所以, 在禁止状态($\bar{1S}=1$)时, 其输出端 W 为高阻抗。

表 3.6.2 双 4 选 1 数据选择器功能表

选通	选择输入		数据输入				输出
$\bar{1S}$	A_1	A_0	D_0	D_1	D_2	D_3	W
1	×	×	×	×	×	×	0
0	0	0	0	×	×	×	0
0	0	0	1	×	×	×	1
0	0	1	×	0	×	×	0
0	0	1	×	1	×	×	1
0	1	0	×	×	0	×	0
0	1	0	×	×	1	×	1
0	1	1	×	×	×	0	0
0	1	1	×	×	×	1	1

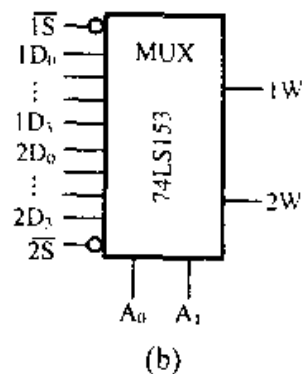
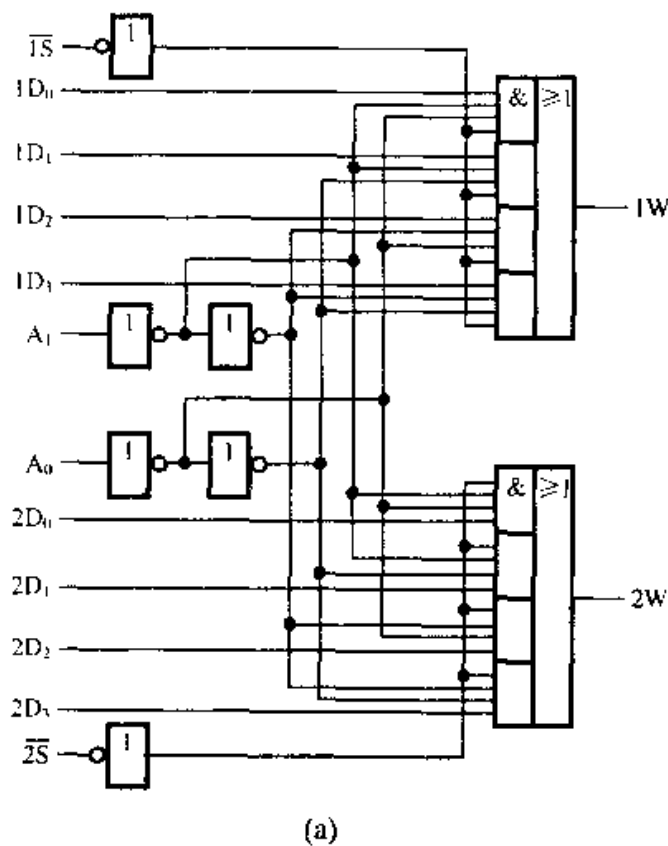


图 3.6.2 双 4 选 1 数据选择器

(a) 逻辑电路图 (b) 逻辑框图

74LS353 的电路与图 3.6.2(a)也类似, 不同之处在于输出为反码, 即函数表达式为

$$W = (\bar{A}_1 \bar{A}_0 D_0 + \bar{A}_1 A_0 \bar{D}_1 + A_1 \bar{A}_0 \bar{D}_2 + A_1 A_0 \bar{D}_3) \bar{S} \quad (3.6.3)$$

2. 8 选 1 数据选择器

8 选 1 数据选择器 74LS151 的逻辑电路图如图 3.6.3(a)所示, 图(b)是逻辑框图。该组件设置了两个互补的输出端 W 和 \bar{W} , 分别输出原码和反码。功能表示于表 3.6.3。由图 3.6.3(a)可以写出

$$\begin{aligned} W = & \bar{A}_2 \bar{A}_1 \bar{A}_0 D_0 + \bar{A}_2 \bar{A}_1 A_0 D_1 + \bar{A}_2 A_1 \bar{A}_0 D_2 + \bar{A}_2 A_1 A_0 D_3 \\ & + A_2 \bar{A}_1 \bar{A}_0 D_4 + A_2 \bar{A}_1 A_0 D_5 + A_2 A_1 \bar{A}_0 D_6 + A_2 A_1 A_0 D_7 \end{aligned} \quad (3.6.4)$$

74LS251 电路与图 3.6.3(a)所示电路类似, 但它为三态输出结构。

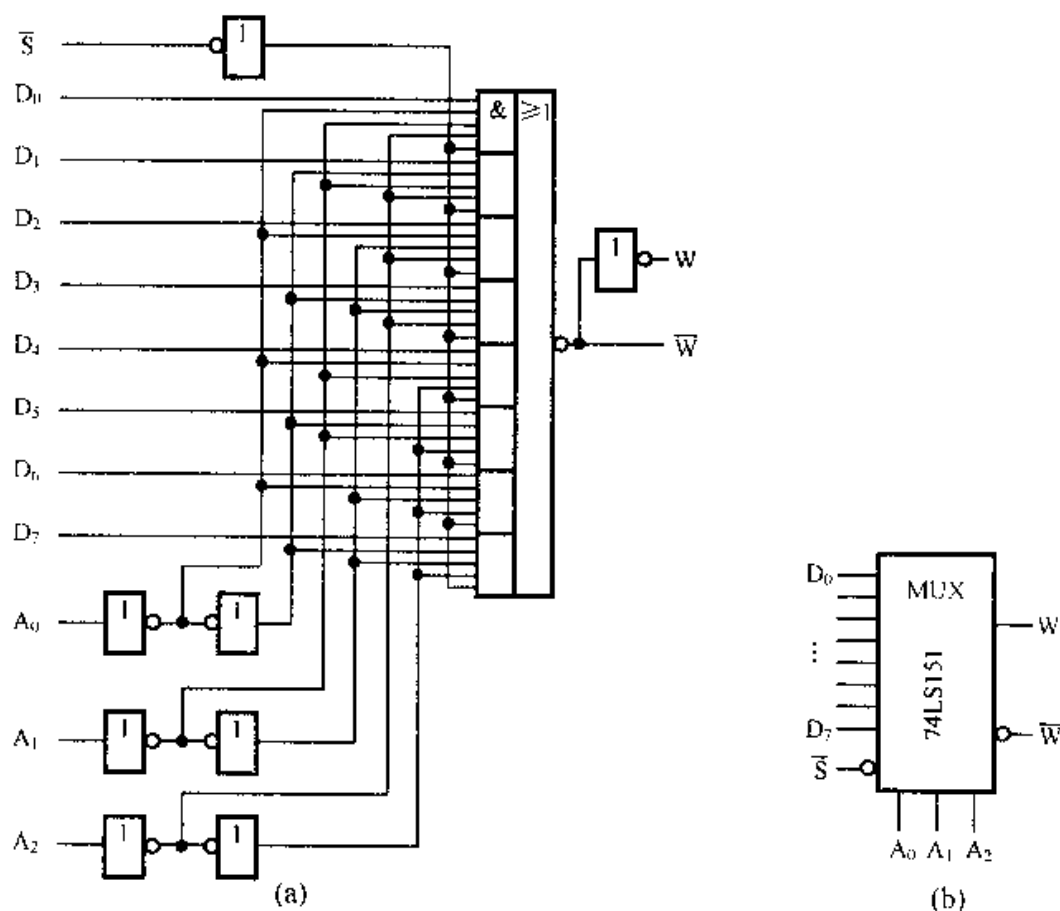


图 3.6.3 8 选 1 数据选择器

(a) 逻辑电路图 (b) 逻辑框图

欲使可供选择的输入信号增加, 可以选用输入通道数目较多的组件, 也可以采用组合的方法来扩展输入端。图 3.6.4(a)所示的是利用一片双 4 选 1 组件构成的 8 选 1 数据选择器; 图(b)所示的是利用两片双 4 选 1 数据选择器构成的 16 选 1 数据选择器, 读者不难分析它们的工作原理。

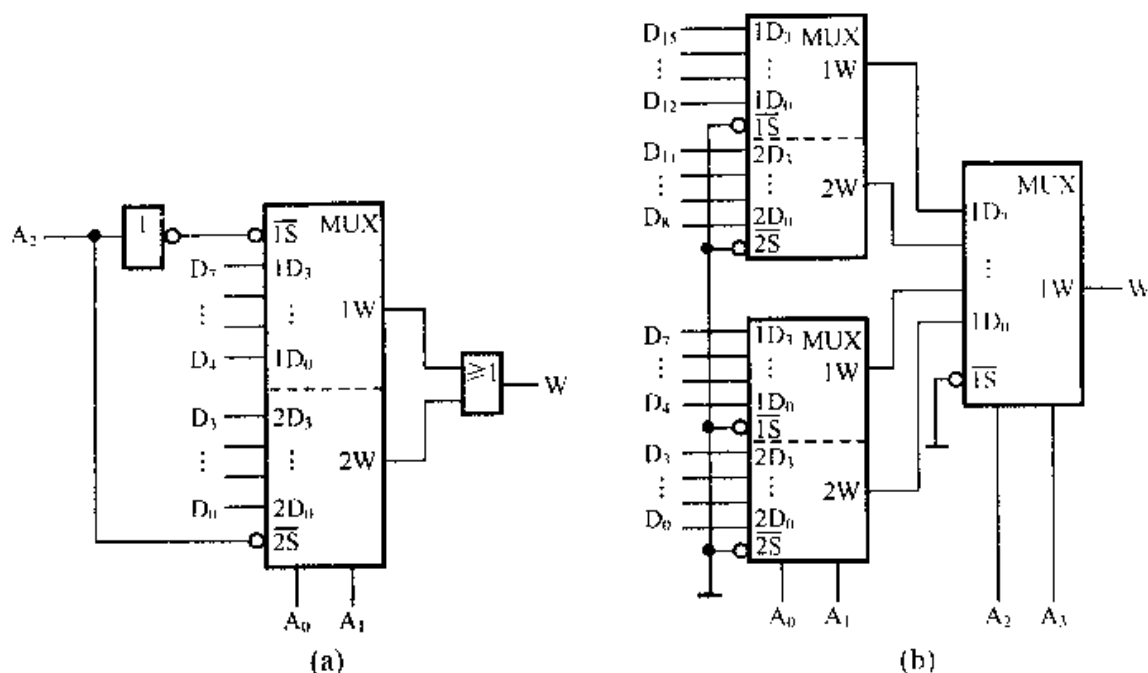


图 3.6.4 数据选择器输入通道的扩展

(a) 双 4 选 1 扩展为 8 选 1 数据选择器 (b) 双 4 选 1 扩展为 16 选 1 数据选择器

3.6.2 用数据选择器设计组合逻辑电路

1. 函数发生器

数据选择器能方便而有效地实现逻辑函数，即构成函数发生器。4 选 1 数据选择器的函数表达式如式(3.6.1)所示，它表示对应 A 和 B 的每一组代码就有相应的输出 $D_i (i=0\sim3)$ 。假如有一个 n 变量的函数，则可以把其中两个变量提出来接到地址输入端，而把剩下的 $(n-2)$ 个变量组成的“剩余函数”接到数据选择器的相应输入端，这样就可以用 4 选 1 数据选择器来实现 n 变量逻辑函数。

例 3.6.1 用 4 选 1 数据选择器实现三变量逻辑函数

$$F = \overline{A}\overline{B}C + \overline{A}BC + A\overline{B}\overline{C} + ABC$$

解 从三变量逻辑函数中提出其中任意两个变量作为地址输入。本例中提出 A 和 B 作为地址输入，那么剩余变量只有一个变量 C。一个变量 C 所能构成的剩余函数有 4 种情况： C 、 \overline{C} 、 $C + \overline{C} = 1$ 、 $C \cdot \overline{C} = 0$ 。所以，现在的问题是确定每个数据输入端应接哪一种剩余函数，为此，可以利用卡诺图来解决这个问题。

首先将函数 F 填在卡诺图上，如图 3.6.5(a)所示；然后指定 A 和 B 作为地址输入，剩余变量为 C，各数据输入端的剩余函数分别为 D_0 、 D_1 、 D_2 和 D_3 。由数据选择器的构成原理可知： D_0 只与 $AB=00$ 的小方块有关，…… D_3 只与 $AB=11$ 的小方块有关。之所以三变量卡诺图变成了 4 个双格的单变量卡诺图，这是因为对每个双格来说，A 和 B 的取值均是确定的。因此，剩余函数 $D_0\sim D_3$ 可以直接从 4 个单变量卡

诺图求出来。例如, 当 $AB=00$ 时, $D_0=C$; 当 $AB=01$ 时, $D_1=1$; 同理可求出 $D_2=0$ 和 $D_3=C$ 。实现函数的逻辑框图如图 3.6.5(b)所示。

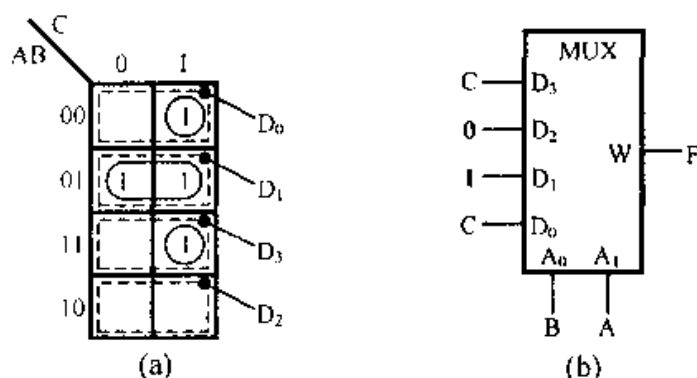


图 3.6.5 4 选 1 数据选择器实现三变量逻辑函数

(a) 三变量卡诺图分解为一个变量卡诺图 (b) 逻辑框图

例 3.6.2 用 4 选 1 数据选择器实现四变量逻辑函数

$$F = \sum m(0,1,5,6,7,9,10,14,15)$$

解 设选用 A 和 B 作为地址输入, 由于 A 和 B 已从函数中提出来另行组合为 4 种不同状态, 所以, 四变量卡诺图就分成为 4 个二变量卡诺图, 如图 3.6.6(a)所示。在每个二变量卡诺图上求出相应剩余函数, 例如, 在 $AB=00$ 的二变量卡诺图上求 D_0 。此时可以在该二变量卡诺图上进行函数化简(注意: 不能跨越此二变量卡诺图范围), 由此可得

$$D_0 = \bar{C} \quad D_1 = C + D \quad D_2 = \bar{C}D + C\bar{D} = C \oplus D \quad D_3 = C$$

实现四变量逻辑函数的逻辑图如图 3.6.6(c)所示。

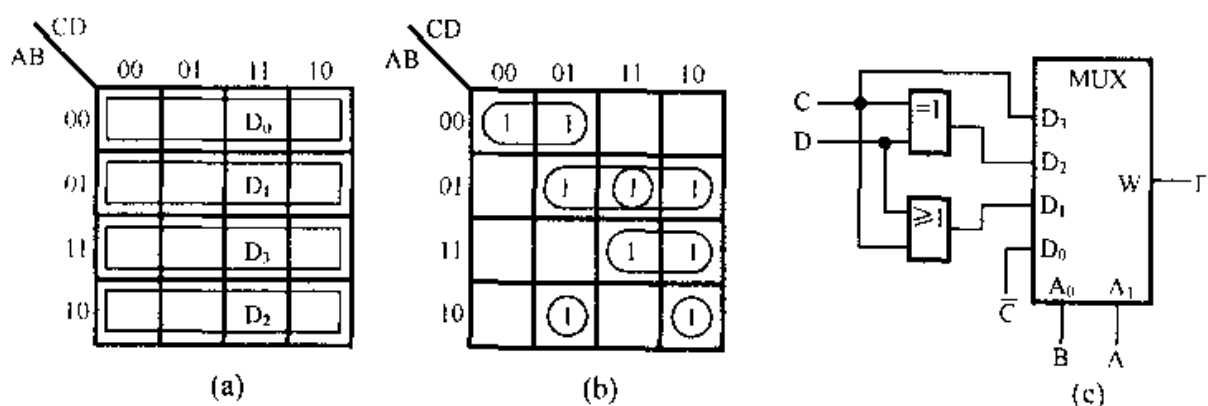


图 3.6.6 4 选 1 数据选择器实现四变量逻辑函数

(a) 四变量卡诺图分解为二变量卡诺图 (b) 二变量卡诺图化简剩余函数 (c) 逻辑框图

顺便指出: 从四变量中提出任意二变量作为地址输入的方案共有 6 种, 不同方案的设计结果会有所差异。究竟选择哪一种方案才可以获得最佳电路呢? 这个问题

需要依靠熟练运用卡诺图来解决,在作出函数卡诺图之后应仔细推敲选用哪一种地址输入方案有利于剩余函数的化简,从而获得最佳设计结果。

例 3.6.3 用 8 选 1 数据选择器实现例 3.1.1 的交通灯工作状态的检测电路。

解 由表 3.1.1 可以直接写出

$$F = \bar{A}\bar{B}C + \bar{A}BC + A\bar{B}C + ABC = m_1 + m_3 + m_5 + m_7$$

用 8 选 1 数据选择器 74LS151 来设计,令 $\bar{S}=0$,用 ABC 作为地址输入。在输出函数 F 包含的最小项的相应输入通道接 1,即 D_1 、 D_3 、 D_5 、 D_6 和 D_7 接 1,其余接 0,电路如图 3.6.7 所示。

例 3.6.4 用 8 选 1 数据选择器实现例 3.6.2 中的逻辑函数

$$F = \sum m(0,1,5,6,7,9,10,14,15)$$

解

$$\begin{aligned} F &= \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}CD + \bar{A}BC\bar{D} + \bar{A}BCD \\ &\quad + A\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D} + ABCD \\ &= \bar{A}\bar{B}\bar{C} \cdot 1 + \bar{A}\bar{B}\bar{C} \cdot D + \bar{A}\bar{B}C \cdot 1 + \bar{A}\bar{B}C \cdot D + \bar{A}BC \cdot \bar{D} + ABC \cdot 1 \end{aligned}$$

将上式与式(3.6.4)对照可得: $D_0=1$, $D_2=D$, $D_3=1$, $D_4=D$, $D_5=\bar{D}$, $D_7=1$ 。

将上列数据接入相应输入通道,其余接 0,如图 3.6.8 所示。

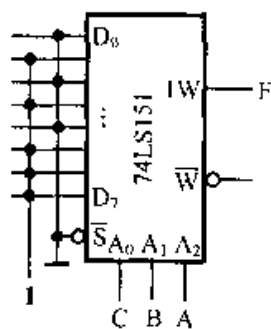


图 3.6.7 例 3.6.3 逻辑框图

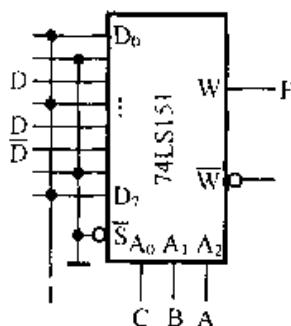


图 3.6.8 例 3.6.4 逻辑框图

从上面例子可以看出:

(a) 用具有 n 个地址输入端(2^n 个输入通道)的数据选择器实现 n 变量的逻辑函数,只需按函数中所包含的最小项,在相应输入通道上接 1,其余通道接 0。

(b) 用具有 n 个地址输入端的数据选择器实现 $n+1$ 个变量的逻辑函数,无需增加附加门,只需在输入通道上依据函数要求接入剩余变量 X 的剩余函数 X 、 \bar{X} 、0 和 1。

(c) 用具有 n 个地址输入端的数据选择器实现 $m(m > n+1)$ 个变量的逻辑函数,一般需要增加小规模的门电路配合使用。

2. 运算电路

用该组件能方便地构成全加器或者全减器,它们都是组合电路,所以,设计过程实际上就是用它实现逻辑函数。如设计全减器,设被减数为 A ,减数为 B ,低位向本位的借位为 C_i ;差为 D , C_o 是本位向高位的借位。

因为 A 、 B 、 C_i 之中有奇数个 1,则差 D 必然为 1;当 A 为 0 时, B 为 1 或者

C_i 为 1, 则 C_o 为 1, 或者 B 、 C_i 为 1, 则 C_o 为 1。由此可得

$$\left. \begin{aligned} D &= A \oplus B \oplus C_i = \bar{A}\bar{B}C_i + \bar{A}B\bar{C}_i + A\bar{B}\bar{C}_i + ABC_i \\ C_o &= \bar{A}(B + C_i) + BC_i = \bar{A}B + \bar{A}C_i + BC_i = \bar{A}B + \bar{A}\bar{B}C_i + ABC_i \end{aligned} \right\} \quad (3.6.5)$$

用 A 、 B 作为地址输入, 由上式便可作出用双 4 选 1 数据选择器构成全减器的连接图, 如图 3.6.9 所示。

3. 实现并行数据到串行数据的变换

图 3.6.10 所示的是实现并-串变换的逻辑框图。电路由一个 8 选 1 数据选择器和一个 3 位二进制计数器构成。计数器 (参阅 5.6 节) 的作用是累计时钟脉冲的数目。随着时钟脉冲 CP 的作用, 它的输出状态 $Q_2Q_1Q_0$ 将从 $000 \rightarrow 001 \rightarrow \dots \rightarrow 111 \rightarrow 000 \rightarrow \dots$ 依次循环变化, 所以输出端 W 便得到一串随时钟节拍变化的串行数据 01001101 。而且是循环长度为 8 位的串行数据。

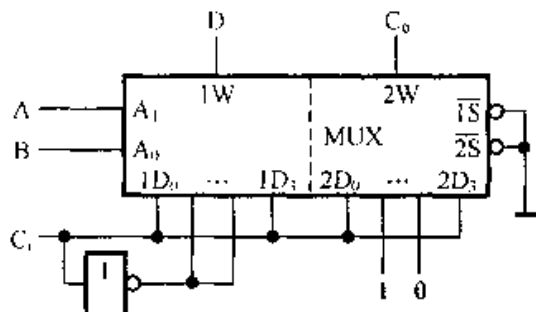


图 3.6.9 双 4 选 1 数据选择器构成全减器

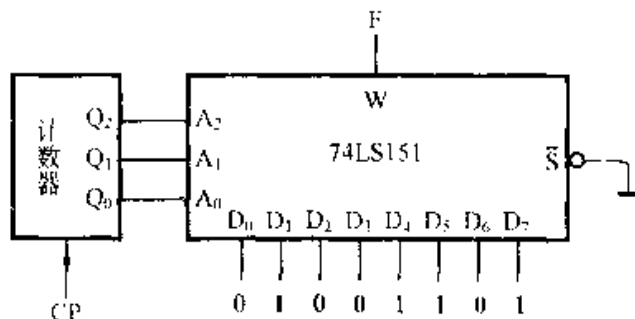


图 3.6.10 实现并-串变换的逻辑框图

4. 脉冲序列产生器

用数据选择器与计数器配合, 可以灵活地构成任意循环长度和任意的脉冲序列。逻辑框图如图 3.6.11 所示。

4 位计数器提供 16 选 1 数据选择器的地址输入, 而输入通道依据要求的脉冲序列接固定电平, 这样便可产生循环长度为 16 的脉冲序列, 每个码元宽度等于计数脉冲 CP 的周期。

假如模数改变为 N , 则构成 N 位脉冲序列; 改变输入通道的逻辑电平值便可实现 N 位的任意脉冲序列。

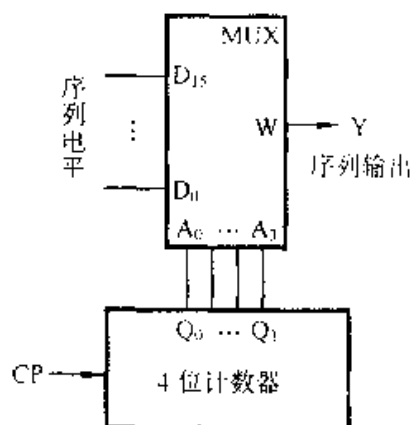


图 3.6.11 脉冲序列产生器逻辑框图

*3.7 数据分配器

数据分配器(多路分配器)的逻辑功能是, 将一路输入数据 D 依据地址代码分配

到相应输出端(又称输出通道)。4 输出数据分配器(简称 4 路分配器)的框图如图 3.7.1 所示。

D 为输入数据; A 和 B 为地址输入; $Y_0 \sim Y_3$ 为输出通道。对应 A 和 B 的一组代码便将数据 D 分配到相应输出通道。一般情况, 若有 n 个地址输入则有 2^n 个输出通道。

数据分配器用途十分广泛, 例如, 用它与计数器配合构成多相脉冲序列; 它与数据选择器配合就可构成多路通信中的常用部件, 即无触点双刀多位开关等。尽管应用广泛, 但是, 很少生产这类专用产品, 一般都是在设计译码器组件时已考虑数据分配器兼容问题。对于一般译码器组件, 只要正确地运用使能端或地址输入端, 就可以作为数据分配器使用。

在设计译码器组件时已考虑兼容的产品有: 双 2-4 线译码器/分配器 74LS155 等; 地址锁存 3-8 译码器/分配器 74LS137。

双 2-4 线译码器/分配器 74LS155 的逻辑框图如图 3.7.2 所示。

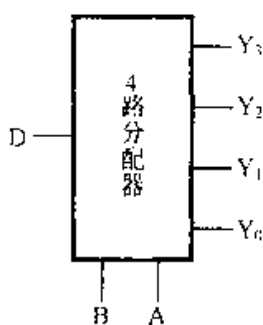


图 3.7.1 4 路数据分配器框图

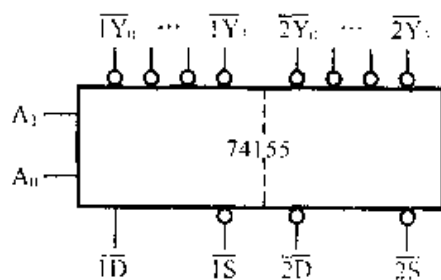


图 3.7.2 双 2-4 线译码器/分配器

该组件包含两个独立的 2-4 线译码器/分配器。当作译码器使用时, A_0 、 A_1 为输入信号(公用), $1D$ 、 $\overline{1S}(\overline{2D}, \overline{2S})$ 作为两个使能端(注意: $\overline{1S}$ 、 $\overline{2S}$ 、 $\overline{2D}$ 均为低电平有效, 而 $1D$ 为高电平有效); 当作分配器使用时, $\overline{1S}(\overline{2S})$ 作为使能端, $1D(\overline{2D})$ 作为数据输入端, A_0 和 A_1 则作为公有的地址输入。

其功能示于表 3.7.1(a)和(b)。

用类似上述方法, 可将设置有使能端的译码器改接为数据分配器使用。图 3.7.3(a)所示的是由 4-16 线译码器改接为 16 路分配器使用, 此时将使能端 \overline{S}_A 保留, 而将另一使能端 \overline{S}_B 作为数据输入端 D , $A_3 \sim A_0$ 作为地址输入。由地址代码将数据 D 分配到相应输出通道(参见图 3.4.5)。图 3.7.3(b)是由 4-10 线译码器改接为 8 路分配器, 此时组件无使能端(参见图 3.4.7), 所以, 用 A_3 作数据输入端 D , $A_2 \sim A_0$ 作为地址输入; $\overline{Y}_0 \sim \overline{Y}_7$ 作为输出通道(\overline{Y}_8 、 \overline{Y}_9 不用)。当 $D=0$ 时, 地址选中的输出为 0; 当 $D=1$ 时, 地址选中的输出为 1。

表 3.7.1(a) 2-4 线译码器/分配器功能表之一

输 入				输 出			
选择		选通	数据	$\overline{1Y_0}$	$\overline{1Y_1}$	$\overline{1Y_2}$	$\overline{1Y_3}$
A_1	A_0	$\overline{1S}$	$1D$				
×	×	1	×	1	1	1	1
0	0	0	1	0	1	1	1
0	1	0	1	1	0	1	1
1	0	0	1	1	1	0	1
1	1	0	1	1	1	1	0
×	×	×	0	1	1	1	1

表 3.7.1(b) 2-4 线译码器/分配器功能表之二

输 入				输 出			
选择		选通	数据	$\overline{2Y_0}$	$\overline{2Y_1}$	$\overline{2Y_2}$	$\overline{2Y_3}$
A_1	A_0	$\overline{2S}$	$\overline{2D}$				
×	×	1	×	1	1	1	1
0	0	0	0	0	1	1	1
0	1	0	0	1	0	1	1
1	0	0	0	1	1	0	1
1	1	0	0	1	1	1	0
×	×	×	1	1	1	1	1

也可以采用多级结构来扩展多路分配器输出通道。图 3.7.4 是用 9 个 8 路分配器构成 64 路分配器的示意图。

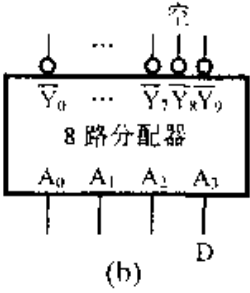
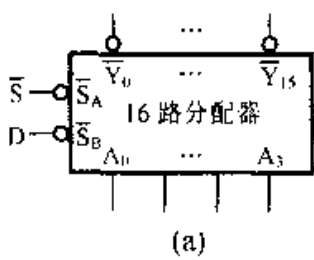


图 3.7.3 用译码器构成数据分配器
(a) 4-16 线译码器构成 16 路分配器
(b) 4-10 线译码器构成 8 路分配器

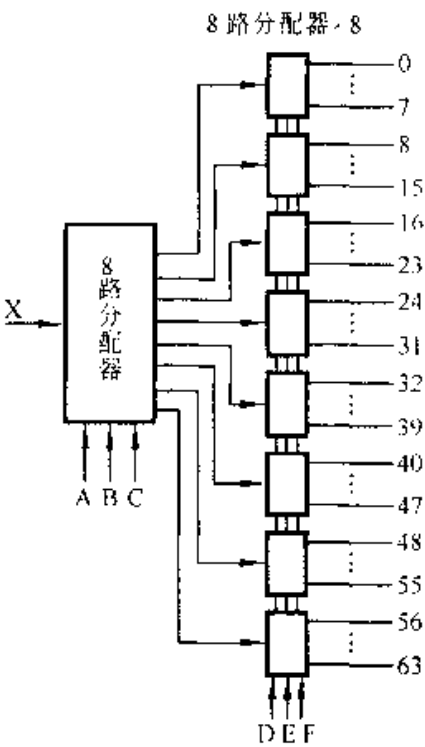


图 3.7.4 64 输出多路分配器

*3.8 奇偶产生器/检验器

3.8.1 奇偶产生器/检验器的设计

具有产生奇偶位和进行奇偶检测功能的逻辑电路称为奇偶产生器/检验器。现以

8421 码的偶检验码为例来讨论这类电路的设计方法。

设 8421 码为 $A_3A_2A_1A_0$ ，奇偶位为 P 。

由异或运算特点可知：当信息码中的 1 码元为奇数时，信息码所有码元异或结果为 1，故异或结果可以作为奇偶位 P ，即有

$$P = A_3 \oplus A_2 \oplus A_1 \oplus A_0$$

由上式构成奇偶位发生器逻辑图如图 3.8.1(a)所示。

在接收端对收到的码组 $A_3A_2A_1A_0P$ 进行检测，由异或运算可以得到

$$F = A_3 \oplus A_2 \oplus A_1 \oplus A_0 \oplus P = 0$$

所以，当 F 为 0 时判定码组正确，当 F 为 1 时判定码组差错。

其逻辑电路图如图 3.8.1(b)所示。

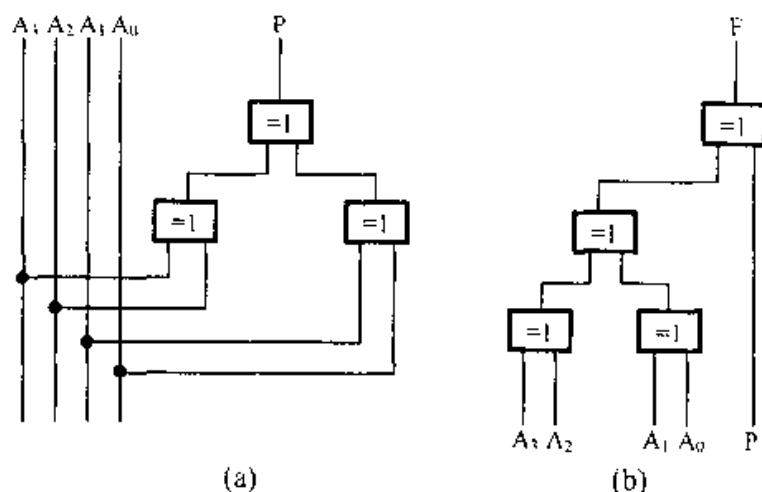


图 3.8.1 奇偶产生器/检验器

(a) 奇偶产生器 (b) 奇偶检验器

3.8.2 中规模集成奇偶产生器/检验器

国产中规模组件有 9 位奇偶产生器/检验器，型号为 74LS280；8 位奇偶产生器/检验器，型号为 74LS180。图 3.8.2(a)所示的是 74LS280 逻辑框图，功能表示于表 3.8.1。

$A \sim I$ 为 9 个数码输入端，两个互补输出端 F_E 和 F_{OD} 。该组件作奇偶位(监督码元)产生器使用时，输入 $A \sim H$ 接 8 位待传数码， I 接 1 电平。若采用偶检验则 F_E 为监督码元；若采用奇检验则 F_{OD} 为监督码元，即

$$F_{OD} = A \oplus B \oplus C \oplus D \oplus E \oplus F \oplus G \oplus H \oplus I$$

$$F_E = \bar{F}_{OD}$$

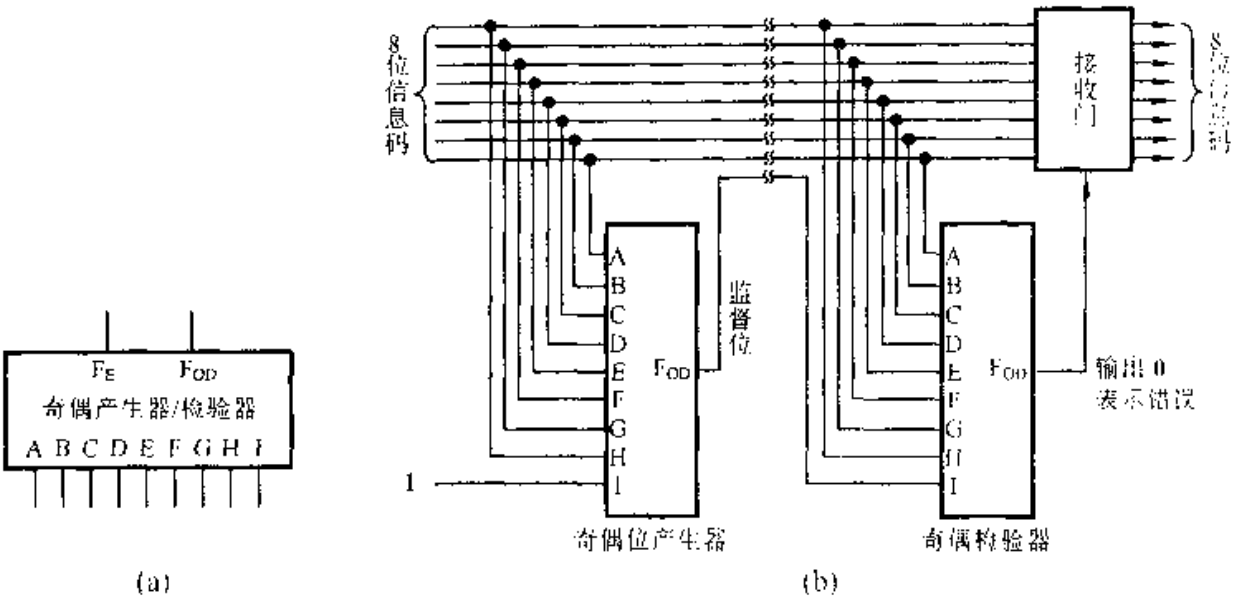


图 3.8.2 9 位奇偶产生器/检验器
(a) 逻辑框图 (b) 奇检验的 8 位数据传输系统

该组件作检验器使用时, 将所有 9 位输入数码加入 A~I。若采用偶检验时 F_E 为 1, 则判正确; 若采用奇检验时 F_{OD} 为 1 则判正确。否则就判为产生差错。图 3.8.2(b)所示的是用两片 9 位奇偶产生器/检验器构成检验 8 位数据传输系统的连接图, 约定传输系统采用奇检验。

表 3.8.1 9 位奇偶产生器/检验器功能表

输 入		输 出	
A~I 中 1 的个数		F_E	F_{OD}
偶 数		1	0
奇 数		0	1

3.9 组合逻辑电路中的冒险现象

3.9.1 组合电路中的竞争与冒险

在前面讨论组合电路时, 都没有考虑门电路的延迟时间, 即仅讨论电路的输出与输入之间的稳态关系。实际上输入信号的变化总是存在一个过渡过程, 而门电路也都存在延迟时间, 它们对组合电路的输出都会产生影响。在本节的讨论中, 假定门电路的延迟时间相同, 令一个门的延迟时间为 t_{pd} 。

图 3.9.1(a)所示的是一个简单的组合电路, 若不考虑门的延迟时间, 则有 $F=A \cdot \overline{A}=0$, 所以, 在稳态时的输出恒等于 0; 若考虑门的延迟时间, 则信号 A 经非门延迟 t_{pd} 才得到信号 \overline{A} , 因此, 信号 A 与 \overline{A} 到达与门的输入端便有先后之差, 由此导致了正尖脉冲的错误输出, 如图 3.9.1(b)所示。习惯上说电路产生了毛刺。

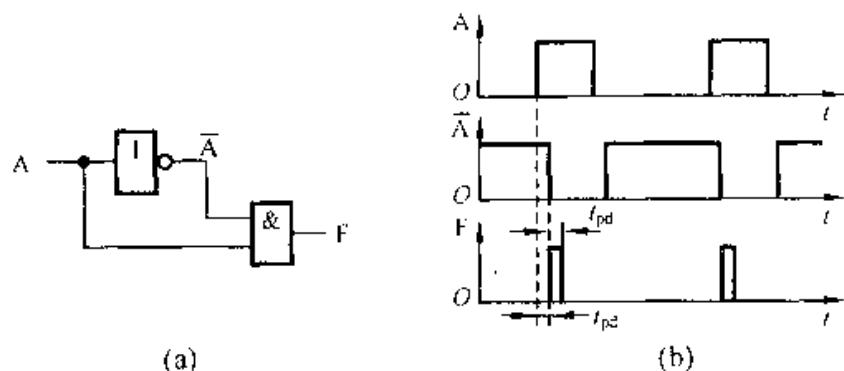
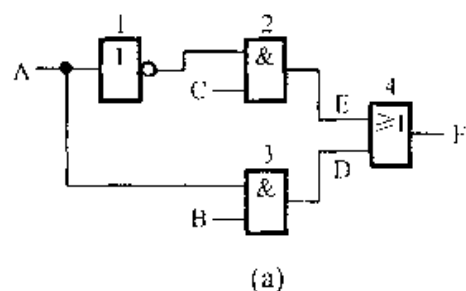


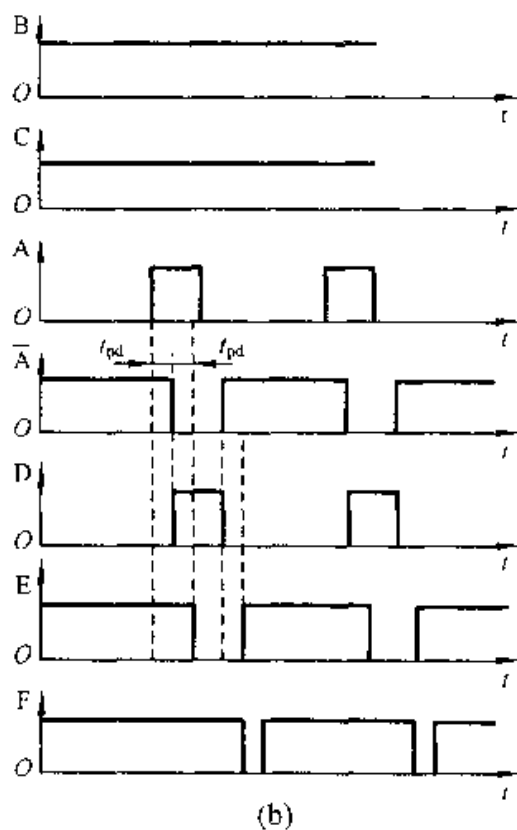
图 3.9.1 逻辑冒险举例之一

(a) 电路图 (b) 波形图

下面来讨论图 3.9.2(a)所示电路, 由图可得 $F=AB+\bar{A}C$ 。当 $B=C=1$ 时, 则有 $F=A+\bar{A}$, 所以在稳态条件下 F 应恒等于 1。现在来分析考虑门电路延迟时间所出现的情况。信号 A 经非门 1 延迟 t_{pd} 产生 \bar{A} , 再经与门 2 延迟 t_{pd} 到达或门 4 的输入端 E , 总计延迟时间为 $2t_{pd}$; 信号 A 经与门 3 延迟 t_{pd} 便到达或门 4 的输入端 D 。由于或门 4 的两个输入端上的信号 A 与 \bar{A} 有先后之差, 所以, 在输出端产生负尖脉冲, 如图 3.9.2(b)所示。



(a)



(b)

图 3.9.2 逻辑冒险举例之二

(a) 电路图 (b) 波形图

由上面例子的分析, 可以发现在考虑门电路延迟时间的情况下, 产生毛刺的原因是由于一个信号经过不同途径到达同一门的输入端时有“时差”现象, 这种时差现象称为竞争。由于竞争而产生了毛刺, 一个组合电路是否存在竞争可以通过最简逻辑函数表达式来判断: 如果在一个输入变量发生变化的条件下, 若其他输入变量的某种取值使函数为 $F=A \cdot \bar{A}$ 或者 $F=A+\bar{A}$, 则可以判定电路存在竞争。前者称为 $A \cdot \bar{A}$ 型竞争, 后者称为 $A+\bar{A}$ 型竞争。

组合电路中竞争是很普遍的现象, 但并不是存在竞争的电路就一定会产生冒险。例如, 图 3.9.3 所示电路, 在 $B=1$ 时, 与门 2 的输出为 $F_2=A+\bar{A}$; 在 $A=1$ 时, 与门 3 的输出 $F_3=B+\bar{B}$ 。与门 2 和 3 均存在 $A+\bar{A}$ 型竞争, 但是该电路的输出为

$$F = A\bar{B} + \bar{A}B$$

当 $B=1$ 时, $F=\bar{A}$; 当 $B=0$ 时, $F=A$ 。同理, 当 $A=1$ 时, $F=\bar{B}$; 当 $A=0$ 时, $F=B$ 。由上面分析可知电路输出没有出现 $A+\bar{A}$ 型或者 $A \cdot \bar{A}$ 型竞争, 所以, 电路输出也不会产生冒险。

定义不产生错误输出的竞争为非临界竞争; 产生错误输出的竞争为临界竞争, 或称为冒险。显然, 由于冒险现象(即毛刺)将使电路工作可靠性降低, 所以, 在设计组合电路时, 需要研究临界竞争, 即研究冒险产生的条件, 研究发现及消除它们的方法。

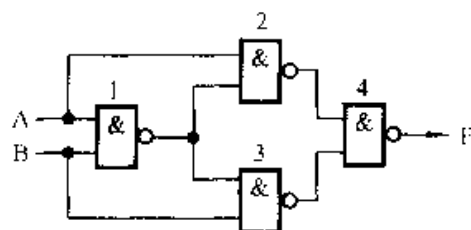


图 3.9.3 竞争未产生冒险举例

前面的分析都是在一个输入变量发生变化的条件下, 电路在过渡过程中产生冒险, 一般称为逻辑冒险; 在两个或多个输入变量同时发生变化的条件下, 由于它们发生变化的“时差”现象, 电路在过渡过程中也可能导致冒险, 这类冒险称为功能冒险。

3.9.2 逻辑冒险的检查和消除

逻辑冒险可分为静态和动态两类。静态逻辑冒险(静态逻辑冒险简称为逻辑冒险或冒险), 是指电路在某个输入变量发生变化的前后, 其输出的稳态值是相同的, 在输入信号发生变化时产生了毛刺; 动态逻辑冒险是指电路的稳态输出值在输入信号变化前后是不同的, 但在输入信号发生变化时产生了毛刺。动态逻辑冒险是在静态逻辑冒险的基础上发展起来的, 假如消除了静态逻辑冒险, 动态逻辑冒险也就可以防止了, 所以, 应该重点研究静态逻辑冒险的发现和消除方法。

借助于卡诺图可以发现电路是否存在静态逻辑冒险, 现以图 3.9.2 所示电路来说明检查方法。由图示电路写出函数表达式为 $F=AB+\bar{A}C$, 将函数 F 填在卡诺图上, 与项 AB 和 $\bar{A}C$ 分别对应合并圈①和②, 如图 3.9.4 所示。

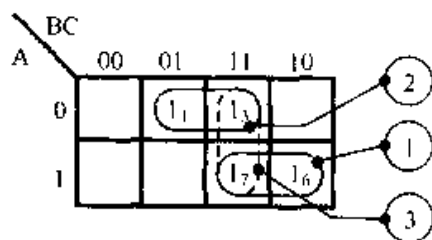


图 3.9.4 图 3.9.2 电路卡诺图

在 $B=C=1$ 条件下, 合并圈①代表 A , 而合并圈②代表 \bar{A} , 所以有 $F=A+\bar{A}$, 检查结果是电路存在静态逻辑冒险。在卡诺图上的表现形式是两个合并圈之间存在着相邻最小项 m_3 和 m_7 , 而且没有公共的合并圈覆盖它们, 那么这两个合并圈之逻辑和就会发生冒险现象。

检查电路存在冒险之后, 怎样才能消除它呢? 仔细分析卡诺图就会发现: 当

$B=C=1$ 时, A 的变化就是从合并圈①跨入合并圈②, 或反之。假如增加一个合并圈③, 那么, A 的变化就在合并圈③之内, 而合并圈③所对应的与项显然与 A 无关, 这时输出函数应为

$$F=AB+\bar{A}C+BC$$

在 $B=C=1$ 条件下, $F=1$, 所以消除了冒险。由此得出结论: 在有相邻项且无公共合并圈的合并圈之间增加一个新合并圈, 使彼此有公共合并圈将相邻项覆盖, 就可以消除冒险。

还可以用布尔代数对上述结论进行验证。因为

$$F=AB+\bar{A}C=AB+\bar{A}C+BC$$

式中, BC 是添加项, 故其函数值不变。显然, 这相当于函数简化的反过程, 函数变复杂了。实际设计时可以暂不考虑冒险是否存在, 用组合电路设计方法设计最简电路, 然后再借助卡诺图判断是否存在冒险, 若存在冒险就采用增加添加项的方法修改设计, 改用较为复杂的电路来达到消除冒险之目的。但是, 从布尔代数的基本定律可以知道一个函数可以增加许多添加项而不改变其值。为了避免增加不必要的添加项就需要准确地找出能消除冒险的添加项, 用上述卡诺图检查和消除便可达到这个要求。合并圈③正是我们所寻找的添加项。图 3.9.5 是无冒险现象的逻辑图。

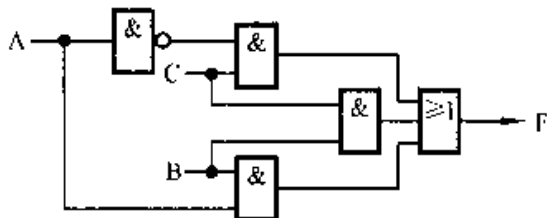


图 3.9.5 增加与门消除逻辑冒险

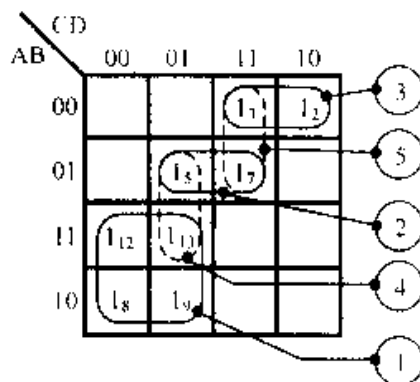


图 3.9.6 例 3.9.1 卡诺图

例 3.9.1 设计无冒险的组合电路, 实现逻辑函数

$$F=AB\bar{C}D+\bar{A}\bar{B}C+\bar{A}BD+\bar{A}\bar{B}\bar{C}+AB\bar{C}+\bar{A}BCD+\bar{A}\bar{B}CD+\bar{A}\bar{B}C\bar{D}$$

解 用观察法直接作出 F 卡诺图并对 F 进行化简, 如图 3.9.6 所示。由图可得最简函数表达式为

$$F=A\bar{C}+\bar{A}BD+\bar{A}\bar{B}C \quad (3.9.1)$$

分析卡诺图可以判定电路存在冒险, 因为合并圈①与②之间有相邻项(m_{13} 与 m_{15})且无公共合并圈; 合并②与③之间有相邻项(m_7 与 m_3)且无公共合并圈。所以, 需要增加合并圈④及⑤, 使这些相邻项都有公共合并圈覆盖, 从而消除冒险。由此可写出无冒险的最简函数表达式为

$$F = A\bar{C} + \bar{A}BD + \bar{A}\bar{B}C + B\bar{C}D + \bar{A}CD \quad (3.9.2)$$

其中, $B\bar{C}D$ 和 $\bar{A}CD$ 就是为消除冒险而增加的添加项, 显然由式(3.9.2)画出的逻辑图比按式(3.9.1)画出的逻辑图要复杂些, 读者可以通过作图予以验证。

例 3.9.2 检查图 3.9.7 所示电路有无逻辑冒险, 若有则予以消除。

解 由图 3.9.7 可写出

$$F = (\bar{A} + B) \cdot (\bar{B} + C) \quad (3.9.3)$$

当 $A=1, C=0$ 时, 上式将成为

$$F = B \cdot \bar{B}$$

故电路存在竞争。但是否导致逻辑冒险, 还需借助卡诺图检查。式(3.9.3)为或与式, 由第 1 章讨论已经知道在卡诺图上圈 0 可以直接写出或与式, 所以, 用观察法将式(3.9.3)中或项的合并圈在卡诺图上画出来, 如图 3.9.8(a)所示。检查卡诺图便发现 M_4

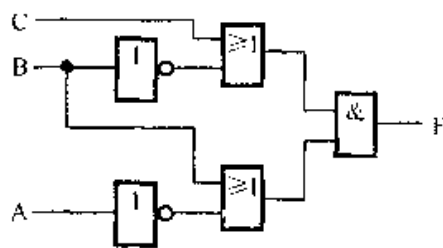
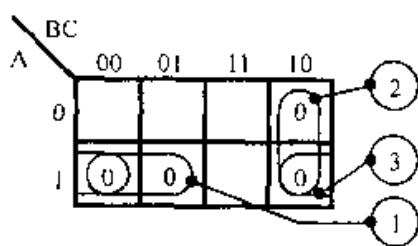
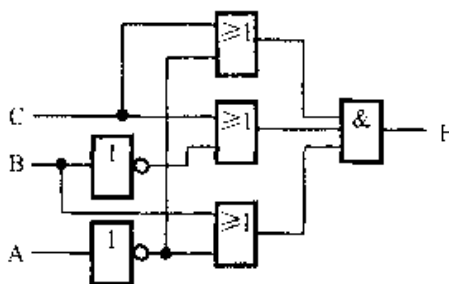


图 3.9.7 例 3.9.2 的逻辑电路图

与 M_6 是相邻最大项, 但无公共覆盖圈, 所以可判定该电路存在逻辑冒险, 增加圈③, 即增加添加项 $\bar{A} + C$ 便可消除逻辑冒险, 其逻辑电路如图 3.9.8(b)所示。



(a)



(b)

图 3.9.8 例 3.9.2 的冒险检查与消除

(a) 卡诺图 (b) 无冒险的逻辑电路图

顺便指出: 本例中不能对式(3.9.3)进行逻辑运算及简化, 否则就将电路简化了, 达不到检查指定电路的目的。

通过以上讨论可以得出:

(a) 若要检查给定电路是否存在逻辑冒险, 可先依据电路写出函数表达式, 若为与或式 (或与式), 则将每个与项 (或项) 的合并圈在卡诺图上画出来; 然后借助卡诺图法检查便可判断。必须指出的是, 不能对电路的函数表达式进行简化, 否则就是对原电路进行简化, 就达不到检查给定电路的目的。

(b) 若要设计无冒险的组合逻辑电路, 则可按照组合电路设计方法, 求出最简函数表达式, 然后利用卡诺图对最简函数表达式进行检查。若存在冒险, 则进一步

利用卡诺图找出必需的添加项,便可得到无冒险的最简函数表达式。

3.9.3 功能冒险的消除

功能冒险是由于两个或多个输入信号同时发生变化,但这些信号之间存在“时差”。因此,可能导致电路输出在过渡过程中产生错误。功能冒险不能采取修改逻辑设计的办法加以消除,通常是采用外加选通脉冲对输入信号进行“取样”的方法来解决。因为冒险现象仅仅发生在输入信号变化转换的瞬间,在它们转换完成之后的稳态情况下是不会有冒险的,所以,可以错开输入信号发生转换的瞬间,使电路稳定之后再对输入信号的逻辑值“取样”,这样就能正确反映稳定时的输出值。此法对选通脉冲的宽度和产生的时间有一定要求:一般选通脉冲的宽度应比较窄,并要求错开输入信号发生转换的瞬间,使之能正确反映组合电路稳定时的输出值。常用的选通脉冲的极性和所加位置如图 3.9.9 所示。需要指出:在增加选通脉冲之后,组合电路的输出已不是电位信号而是脉冲信号了。

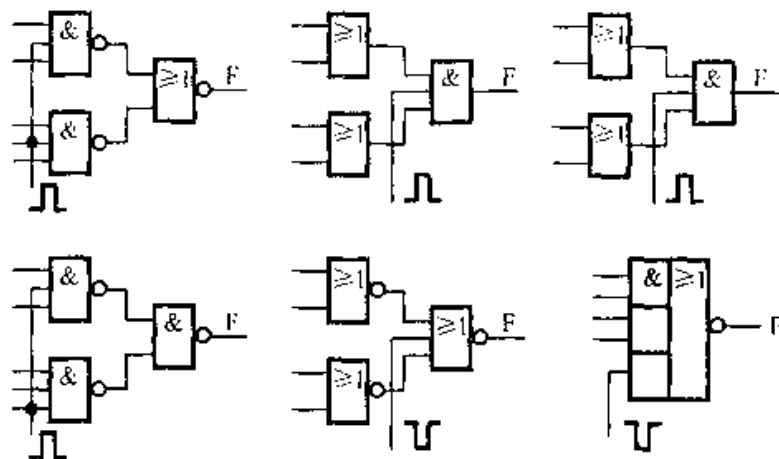


图 3.9.9 采用取样方法消除冒险

小 结

组合电路的特点是任何时刻的稳态输出仅与该时刻的输入状态有关,在电路结构上不含有存储元件,一般由各种逻辑门构成。

组合电路设计的传统方法是“四步法”。若提供原、反变量,则可用两级门结构实现;若仅提供原变量,则需用3级门结构实现。在逻辑电路设计中还有一种灵活设计法,它不局限于传统的步骤,而是利用逻辑函数或逻辑运算中的某些特点来设计电路,往往收到事半功倍的效果。这种方法有利于读者开拓设计思路以及培养

灵活运用逻辑函数和逻辑运算的能力。

组合电路分析是设计的逆过程,关键是写出电路的最简逻辑函数表达式或列出其值表,由此便可描述电路的逻辑功能。

目前,我国中规模集成电路已生产了配套的系列产品,其中一些组合逻辑部件是经常使用的器件,应用它们的最重要问题是熟悉各种控制端的功能、正确的使用方法和功能扩展方法。

组合电路中的冒险现象是实际应用中经常遇到的一个重要问题,在调试数字系统时必须引起足够的重视。如出现逻辑冒险,则可采用添加项的设计方法予以消除;若是出现功能冒险,则采用选通脉冲的“取样”方法予以消除。

思考题和习题

3.1 下列组件各实现什么逻辑功能?它们还可以扩展哪些应用?

译码器、编码器、多路数据选择器、多路数据分配器、4位全加器、4位比较器。

3.2 试述余3码、余3循环码、右移码、格雷码的构成、特点以及它们的应用。

3.3 解释下列名词:

竞争、逻辑冒险、功能冒险、静态逻辑冒险、动态逻辑冒险。

3.4 如何消除组合电路的逻辑冒险及功能冒险?

3.5 写出图 P3.1 中各电路输出函数 F 的表达式。若要求图中(a)、(c)、(f)等电路输出高电平,则它们的输入变量应取何值?

3.6 用与门及异或门实现逻辑函数 $F = \overline{A}BC\overline{D} + \overline{A}BCD + A\overline{B}C\overline{D} + A\overline{B}CD$ 。

3.7 用3个半加器实现下列4个逻辑函数:

$$(1) F_1 = A \oplus B \oplus C$$

$$(2) F_2 = \overline{A}BC + A\overline{B}C$$

$$(3) F_3 = \overline{A}B\overline{C} + (\overline{A} + \overline{B})C$$

$$(4) F_4 = ABC$$

3.8 设 A 、 B 、 C 、 D 为4个二进制数,试设计一个判断电路实现下列功能:

它们中间没有1;它们中间有两个1;它们中间有奇数个1。

3.9 设计一个能接收5位二进制数的组合电路,当它们能被6及7整除时便发出信号。

3.10 设计一个能接收3位二进制数的组合电路,要求输出的二进制数等于输入二进制数的平方。

3.11 设计一个右移码至8421BCD码的码制转换电路。

3.12 用与非门和10只开关设计一个供体育场使用的数字显示电路(只设计1位十进制数)。

3.13 设计一个7段数字显示译码器,要求在出现非法码组时显示E。

3.14 7个开关按下述要求控制一盏灯:当开关1、3、5、7闭合,而开关2断开;或开关2、4、6闭合,而开关3断开;或7个开关都闭合,则此灯点亮。试写出逻辑表达式,画出逻辑图。

3.15 设计一个两个2位二进制数相乘的乘法电路(器件不限)。

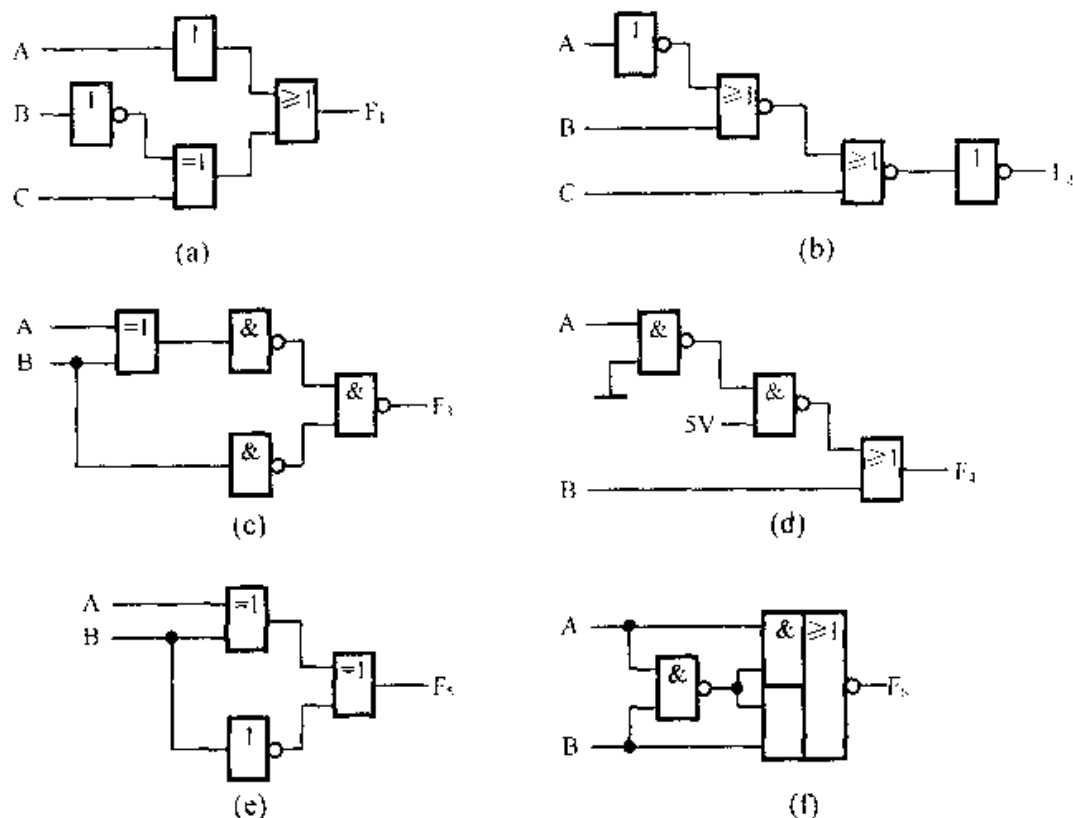


图 P3.1

3.16 一个组合电路设置有两个控制信号 C_1 和 C_2 ，要求：

- (1) $C_2C_1=00$ 时， $F=A \oplus B$
- (2) $C_2C_1=01$ 时， $F=\overline{AB}$
- (3) $C_2C_1=10$ 时， $F=\overline{A+B}$
- (4) $C_2C_1=11$ 时， $F=A \cdot B$

试用与非门设计实现上述要求的组合电路。

3.17 设计一个选通网络，其操作要求如下：当控制信号 $C=1$ 时， n 个异或门完成二进制码至格雷码的转换；当 $C=0$ 时，这些异或门完成格雷码至二进制码的转换。

3.18 已知输入提供原、反变量，但所有输入变量不能同时为 0，且 A 、 B 不能同时为 1，用与非门实现函数。

$$F = \overline{A}B\overline{C} + \overline{B}C\overline{D} + \overline{A}C\overline{D} + \overline{A}B\overline{C}D + \overline{A}B\overline{C}\overline{D}$$

判断所设计的电路有无逻辑冒险，若有则予以消除。

3.19 分析图 P3.2 所示电路，写出逻辑函数表达式。

3.20 分析图 P3.3 所示组合电路，写出逻辑函数表达式。

3.21 分析图 P3.4 所示组合电路。

3.22 用 4 选 1 多路器实现下列函数：

- (1) $F(A, B, C) = \sum m(2, 3, 5, 7)$
- (2) $F(A, B, C, D) = \sum m(0, 1, 2, 3, 8, 9, 10, 11)$

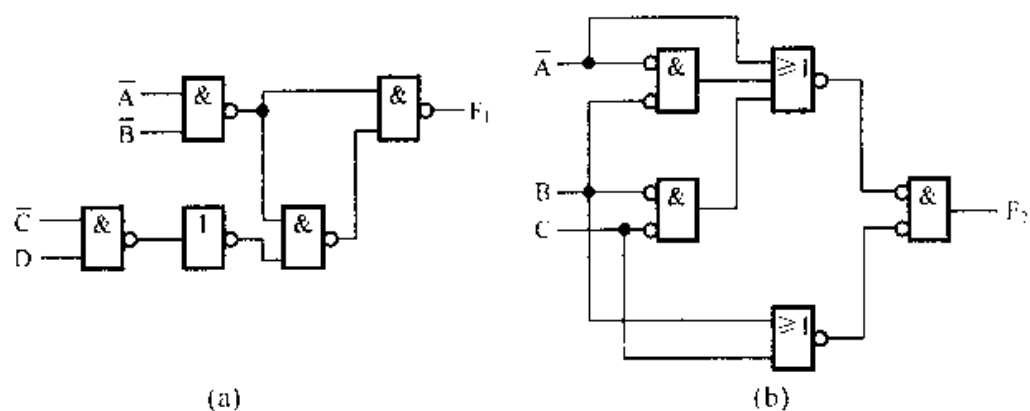


图 P3.2

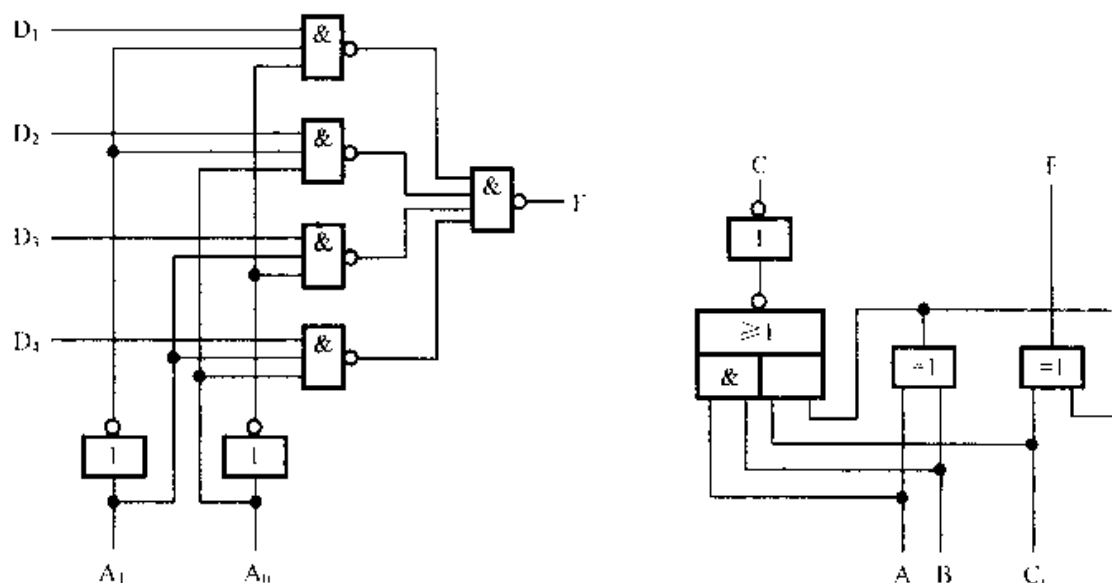


图 P3.3

图 P3.4

3.23 用 4 选 1 多路器实现全加器及全减器。

3.24 写出图 P3.5 所示电路的逻辑函数表达式。

3.25 检查图 P3.6 中所示各电路是否存在逻辑冒险?若有, 请予以消除。

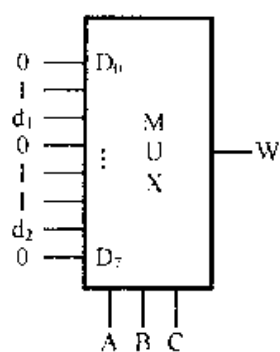


图 P3.5

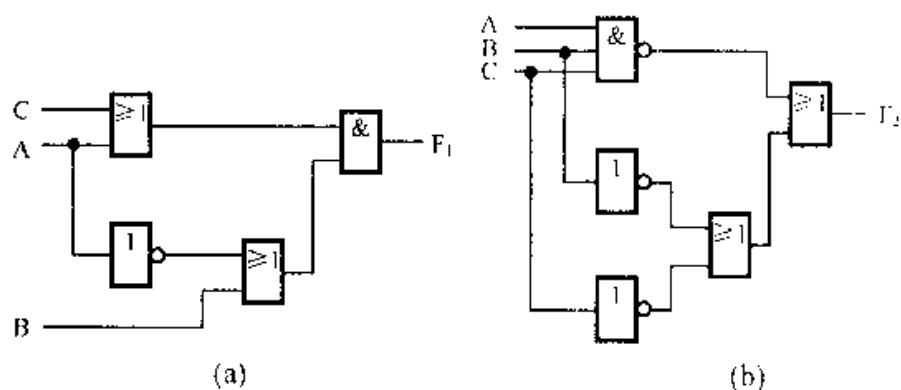


图 P3.6

- 3.26 设输入提供原、反变量，且 C、D 不能同时为 1，试用与非门设计无逻辑冒险的组合电路，实现函数

$$F = \overline{A}CD + A\overline{C}\overline{D} + A\overline{B}\overline{C}D$$

- 3.27 用 4 位全加器和门电路设计下列码组转换器：

- (1) 将格雷码转换为余 3 码；
- (2) 将余 3 码转换为格雷码。

- 3.28 用 4 位比较器设计一个 9 位二进制数是大干、等于或小于 400 的逻辑电路。

- 3.29 用 4-16 线译码器实现下列函数：

- (1) $F_1 = \sum m(0, 3, 6, 10, 15)$
- (2) $F_2 = \overline{A}\overline{B}C + ACD + A\overline{B}CD$
- (3) $F_3 = A\overline{B}C + \overline{A}BCD + \overline{A}\overline{B}CD$

- 3.30 用两片 8 选 1 多路器和门电路扩展为 16 选 1 多路器。

第4章 集成触发器

内容提要 本章重点介绍 TTL 型触发器, 对 CMOS 型和边沿型触发器也作了适当介绍, 最后介绍使用触发器时应注意的问题和描绘工作波形的方法。

在各种数字系统中, 为了存储数字信息, 实现更复杂的逻辑功能, 常常需要具有记忆功能的单元电路。这类电路在某一时刻的输出状态不仅取决于当时的输入信号, 而且和电路以前的状态有关, 在输入信号去除后, 能保留这个信号对电路造成的影响。触发器就是具有这种记忆能力的基本单元电路, 采用它可以使逻辑电路的功能大为扩展。

触发器的种类繁多, 因而可以从不同角度来分类:

依据集成工艺分类, 可分为 TTL 型、MOS 型和 ECL 型;

依据逻辑功能分类, 可分为 RS、JK、D 和 T 型等;

依据电路结构分类, 可分为主从结构、维持阻塞结构;

依据触发方式分类, 可分为电位触发、脉冲触发和边沿型触发等。

4.1 RS 触发器

4.1.1 基本 RS 触发器

基本 RS 触发器是最简单的触发器, 它是将两个与非门的输入与输出交叉连接构成, 如图 4.1.1(a)所示。触发器的两个输入端分别是 \bar{R} 和 \bar{S} , \bar{R} 称为置 0 或复位(Reset)输入端, \bar{S} 称为置 1 或置位(Set)输入端。触发器有两个输出端 Q 和 \bar{Q} , 在正常工作时, 它们总是处于互补的状态。

Q 称为原码输出端 (简称为 1 端), \bar{Q} 为反码输出端 (简称为 0 端), 通常用 Q 端状态来表示触发器的状态。图 4.1.1(b)是基本 RS 触发器的逻辑符号。

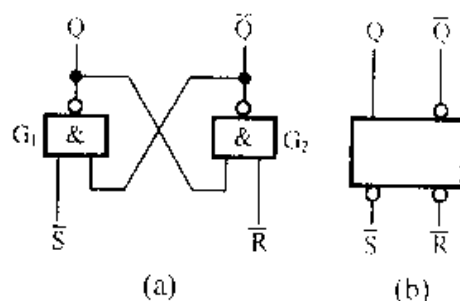


图 4.1.1 基本 RS 触发器
(a) 逻辑电路 (b) 逻辑符号

触发器的逻辑功能可用逻辑功能真值表表示。表 4.1.1 是基本 RS 触发器的逻辑功能真值表。分析如下。

(a) $\bar{S}=1, \bar{R}=0$, 触发器置 0: 如图 4.1.1(a)所示, 由于 $\bar{R}=0$, 必将使门 2 的输出 $\bar{Q}=1$, 而门 1 的两个输入皆为 1, 所以 $Q=0$, 触发器的输出为 0 状态。

(b) $\bar{S}=0, \bar{R}=1$, 触发器置 1: 分析方法同上。 $Q=1$, 触发器的输出为 1 状态。

(c) $\bar{S}=1, \bar{R}=1$, 触发器保持原状态输出: 如原状态 $Q=0$, 由于门 1 两个输入皆为 1, 则触发器保持 $Q=0$ 的原状态输出; 如原状态 $Q=1$, 由于门 2 的两个输入都为 1, $\bar{Q}=0$, 则触发器保持 $Q=1$ 的原状态输出。

(d) $\bar{S}=0, \bar{R}=0$, 禁用的输入状态: 在该输入作用期间, Q 和 \bar{Q} 端均为 1, 但是, 当输入信号同时撤销时(即 $\bar{R}=\bar{S}=1$, 相当于同时加宽度相等的负脉冲于两个输入端), 触发器的状态是不确定的, 因为门 1 和门 2 的延迟时间的差异无法事先确定。输出波形如图 4.1.2 所示, 图中虚线和“?”号表示状态不确定。

顺便指出: 图 4.1.1(b)中, 逻辑符号的输入端上的小圆圈及符号 $\bar{R}、\bar{S}$ (字母上带有非号“-”)表示触发器是低电平触发。

用两个或非门的输入与输出端交叉连接构成基本 RS 触发器的逻辑图示于图 4.1.3(a)、逻辑符号示于图 (b)。逻辑符号的输入端上没有小圆圈, 符号为 $R、S$ (字母上不带非号“-”)表示触发器是高电平触发, 即触发信号是高电平有效。

表 4.1.2 是或非门构成的基本 RS 触发器功能表。

表 4.1.1 基本 RS 触发器真值表

\bar{S}	\bar{R}	Q
1	0	0
0	1	1
1	1	不变
0	0	不确定

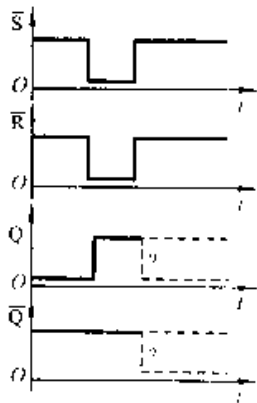


图 4.1.2 禁用状态工作波形

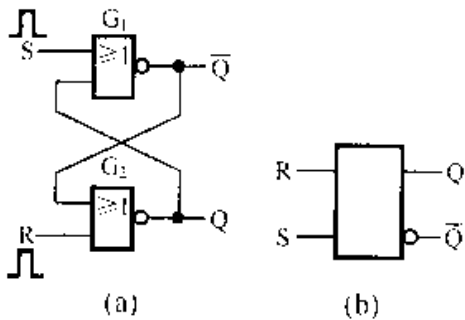


图 4.1.3 两或非门组成的基本 RS 触发器
(a) 逻辑图 (b) 逻辑符号

表 4.1.2 或非门构成的基本 RS 触发器真值表

S	R	Q
1	0	1
0	1	0
0	0	不变
1	1	不确定

4.1.2 时钟 RS 触发器

基本 RS 触发器是直接受输入信号控制,由输入状态便确定了触发器的新状态。但在实际应用中更广泛的触发器,是要求它按一定时间节拍把输入状态反映到输出端,这就需要增加一个触发控制端,只有当触发信号出现时,触发器的输出状态才由输入状态确定,这个触发信号是按一定时间间隔重复出现的脉冲,通常称为时钟脉冲(CP),所以,称它为时钟 RS 触发器,或者称为同步 RS 触发器,因为触发器的动作是与时钟脉冲同步,通常简称为 RS 触发器。

1. 电路结构及逻辑功能

时钟 RS 触发器的逻辑电路图示于图 4.1.4(a),图(b)是逻辑符号。其电路由基本 RS 触发器和由门 3、门 4 构成的控制电路两部分组成。

为了分析方便,定义时钟脉冲出现之前触发器的状态为原状态(现态),记为 $Q_n^{(n)}$;时钟脉冲作用后触发器的状态为新状态(次态),记为 $Q_{n+1}^{(n)}$ (或 Q^{n+1})。此外,规定在时钟脉冲持续期间,其输入状态不变化,即输入信号的持续时间比时钟脉冲长。

由图 4.1.4(a)可见,在 $CP=0$ 时,门 3 和门 4 被封锁,此时门 3 和门 4 输出均为 1(与输入状态无关),所以,触发器的状态维持原状态不变。当时钟脉冲出现时, $CP=1$,将门 3 和门 4 打开, R、S 端的输入信号倒相传送到基本 RS 触发器的输入端,因而确定了触发器的输出状态。由此可见,时钟 RS 触发器是由正脉冲使门 3 或门 4 产生负脉冲将触发器置 1 或置 0 的,显然,在时钟脉冲持续期间,其工作过程与基本 RS 触发器类似,由此便可列出它的真值表,如表 4.1.2 所示。表中, $R=S=1$ 是禁止使用状态,因为在这种输入状态下,当时钟脉冲出现时,门 3 和门 4 输出均为 0,使门 1 和门 2 输出均为 1,即 $Q^{n+1} = \overline{Q^{n+1}} = 1$,但在时钟脉冲消失之后,触发器的状态就不确定。

2. 逻辑功能的表示方法

描述触发器的逻辑功能可采用以下几种方法。

(1) 真值表(见表 4.1.3)

(2) 状态转换真值表

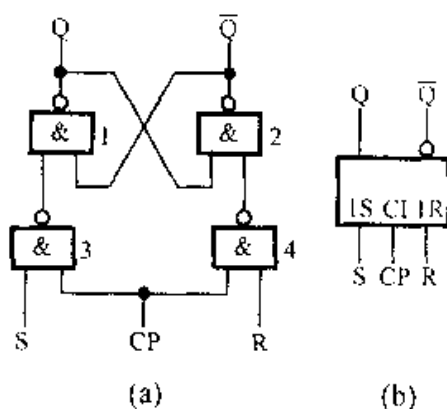


图 4.1.4 时钟 RS 触发器

(a) 逻辑电路 (b) 逻辑符号

① 为了书写方便,本书规定下角标 n 可以省略,如果下角标 n 不省略时,可写为上角标,如 Q_n 和 Q_{n+1} 可写为 Q^n 和 Q^{n+1} ;下角标 $n+1$ 也可以写为上角标,如 Q_{n+1} 可写为 Q^{n+1} , Q_{2n+1} 可写为 Q_2^{n+1} 。

触发器的新状态与原状态和输入信号之间的转换关系可用表格来描述,称之为状态转换真值表。它很容易由真值表得到,如表 4.1.4 所示。表中, $R=S=1$ 为禁用状态,故 Q^{n+1} 作 ϕ 处理。

表 4.1.3 时钟 RS 触发器真值表

S	R	Q^{n+1}
0	0	Q
0	1	0
1	0	1
1	1	不允许

表 4.1.4 时钟 RS 触发器状态转换真值表

Q	S	R	Q^{n+1}	Q	S	R	Q^{n+1}
0	0	0	0	1	0	0	1
0	0	1	0	1	0	1	0
0	1	0	1	1	1	0	1
0	1	1	ϕ	1	1	1	ϕ

(3) 特征方程

描述触发器的逻辑功能的逻辑函数表达式称为特征方程。由表 4.1.4 可得 Q^{n+1} 的卡诺图,如图 4.1.5 所示。特征方程为

$$\left. \begin{aligned} Q^{n+1} &= S + \bar{R}Q \\ SR &= 0 \end{aligned} \right\} \quad (4.1.1)$$

其中, $SR=0$ 称为约束条件,因为 S 和 R 均为 1,则 Q^{n+1} 不确定,为了获得确定的 Q^{n+1} ,则输入信号 S 和 R 应满足 $SR=0$ 。

(4) 激励表

激励表用来描述触发器由现态转换到确定的次态时,对输入信号的要求。激励表可由状态转换表导出,如表 4.1.5 所示。

	SR	00	01	11	10
Q	0	0	0	ϕ	1
	1	1	0	ϕ	1

图 4.1.5 时钟 RS 触发器 Q^{n+1} 的卡诺图

表 4.1.5 RS 触发器激励表

Q	Q^{n+1}	S	R
0	0	0	ϕ
0	1	1	0
1	0	0	1
1	1	ϕ	0

(5) 状态图

触发器的逻辑功能也可以采用图形的方式来描述,称之为状态转换图,简称状态图。RS 触发器的状态图示于图 4.1.6,图中,两个圆圈分别表示两个稳定状态,用带箭头的直线或弧线表示状态转换方向,而在其上标注转换条件。

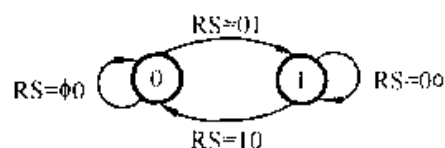


图 4.1.6 时钟 RS 触发器的状态图

RS 触发器的逻辑功能可作如下描述:

(a) 当触发器为 0(1)状态时,若 RS 为

01(10)，则触发器转换至 1(0)状态^①。

(b) 当触发器为 0(1)状态时，若 RS 为 $\phi 0(0\phi)$ ，则触发器维持在 0(1)状态。

(6) 时序图

时序图又称为工作波形图或时间波形图，它是以波形的形式来描述逻辑功能的。RS 触发器的工作波形如图 4.1.7 所示，图中，假设触发器的起始状态(初态)为 0。

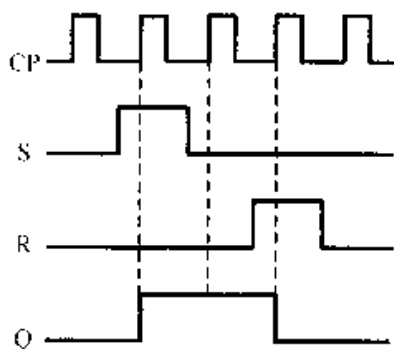


图 4.1.7 时钟 RS 触发器的工作波形

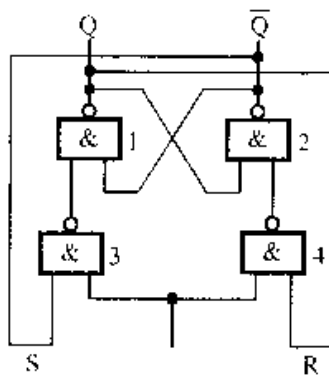


图 4.1.8 计数触发器逻辑电路

3. RS 触发器的空翻现象

RS 触发器在 CP 为 0 电平时，状态维持不变，而在 CP 为 1 电平期间，输入信号的变化对触发器的状态都会产生控制，故称之为电位触发方式，由此会导致空翻现象。

下面用 RS 触发器构成计数触发器为例来说明空翻现象。

计数触发器就是每来一个时钟脉冲，触发器的状态改变一次，所以，状态改变的次数就反映了时钟脉冲数目。计数触发器是构成计数电路的基本单元。用时钟 RS 触发器构成计数触发器的逻辑电路如图 4.1.8 所示。只需将 Q 与 R， \bar{Q} 与 S 分别连接便可，但是，这种电路在时钟脉冲较宽时，输出状态就不能严格地按时钟节拍动作，而是在一个时钟脉冲下可能引起触发器状态产生两次或多次翻转，造成节拍混乱，这就是空翻现象。

例如，设触发器原状态为 0 态，则有 $S=\bar{Q}=1$ 和 $R=Q=0$ 。当 CP 脉冲由 0 变为 1 时，门 3 输出低电平使触发器状态 Q 由 0 变为 1， \bar{Q} 则由 1 变为 0，这时，如果 CP 脉冲较宽，在状态翻转之后 CP 脉冲仍处在高电平，则有 $S=\bar{Q}=0$ 和 $R=Q=1$ ，因而门 4 输出低电平又使触发器状态翻转，Q 由 1 变为 0，如此翻来覆去形成空翻。由此可见，产生空翻的原因是在 CP 脉冲作用期间，控制电路的门 3 和门 4 总是开启的，开始控制电路进行正确导引，而当触发器状态改变之后又进行错误导引，使触发器产生再次翻转。

为了克服空翻现象就要对控制电路进行改进，目前常用的有主从结构、维持阻塞结构和边沿型触发结构。

^①为了简便，本书采用两句类似的描述合并为一句的描述方式，它们的差异用括号表示。

刻,通常称为下跳沿(下降沿)触发,这是所有采用主从结构的触发器所共有的特性,如下面要介绍的主从 JK 触发器也具有这个特性,在图 4.1.9(b)所表示的逻辑符号中,“ \downarrow ”表示延迟输出,即输出状态的变化发生在 CP 脉冲的下跳沿。

主从 RS 触发器还设置有异步置 1 端 \bar{S}_D 和异步置 0 端 \bar{R}_D , 通过它们可以直接使触发器置 1(一般称置位)或置 0(一般称复位), 它们都是低电平有效。所以, 图 4.1.9(b)表示的逻辑符号中, \bar{S}_D 和 \bar{R}_D 端都加有小圆圈。

4.2 主从 JK 触发器

1. 电路结构及逻辑功能

主从 RS 触发器克服了空翻现象,但是, R、S 端仍然存在约束条件,如果在 R、S 端同时加 1, 就会出现 CP 脉冲作用之后新状态不确定的情况, 这个缺点使主从 RS 触发器不能获得广泛应用。为了构成无约束条件的触发器, 可以利用 Q 和 \bar{Q} 端不可能同时为 1 的特点, 将 Q 端与 R 端之中的一个输入端相连, 如将 \bar{Q} 端与 S 端之中的一个输入端相连, 那么, 在 R、S 端就不可能同时出现 1, 因而克服了约束条件这个缺点。按这种要求改接后的逻辑电路示于图 4.2.1(a)。为了与主从 RS 触发器区别, 将 R 端改称为 K 端, S 端改称为 J 端, 便构成主从 JK 触发器, 图(b)是逻辑符号。逻辑功能分析如下。

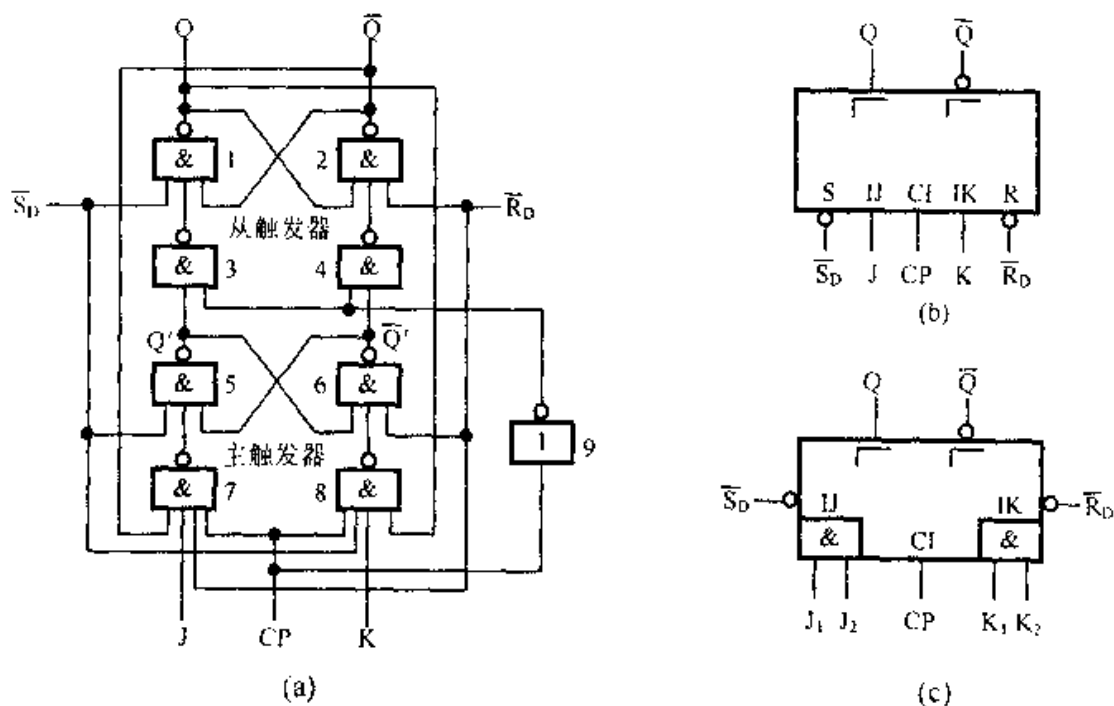


图 4.2.1 主从 JK 触发器

(a) 逻辑电路图 (b) 逻辑符号之一 (c) 逻辑符号之二

当 $J=K=1$ 时, 这种情况相当于主从 RS 触发器构成计数触发器, 所以, 在 CP 脉冲作用后状态要改变, 即有 $Q^{n+1}=\bar{Q}$ 。

当 $J=K=0$ 时, 由于主触发器被封锁, CP 脉冲作用后触发器状态保持不变, 即有 $Q^{n+1}=Q$ 。

当 $J=1$ 和 $K=0$ 时, 设原状态 $Q=0$, 则 $S=J\bar{Q}=1$, $R=KQ=0$ 。在 $CP=1$ 时, 主触发器状态 $Q'=1$, 此时从触发器被封锁, 其状态 Q 保持不变; 当 CP 由 1 变为 0 时, 主触发器被封锁, 它的状态保持不变, 此时从触发器打开, 主触发器的状态便传送到从触发器中去, 即 Q 由 0 变为 1。

设原状态 $Q=1$, 则 $S=J\bar{Q}=0$, $R=KQ=0$, 这种情况就相当于主从 RS 触发器输入端 R 和 S 均为 0, 所以, 在 CP 脉冲作用之后触发器状态不变, 即保持 1 态。由此可见, 无论触发器原状态如何, 在 CP 脉冲作用后触发器总是置 1, 所以有 $Q^{n+1}=1$ 。

当 $J=0$ 和 $K=1$ 时, 这种情况的分析方法与 $J=1$ 和 $K=0$ 时相同, 其结果是 $Q^{n+1}=0$ 。

综上所述, 可得主从 JK 触发器的真值表, 如表 4.2.1 所示。

表 4.2.1 主从 JK 触发器的真值表

J	K	Q^{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	\bar{Q}_n

在集成的产品中, 设置有异步置 1 端 \bar{S}_D 、异步清 0 端 \bar{R}_D ; 设置有多个 J 端和 K 端(2 个~5 个), 它们是相与关系, 即 $J=J_1 \cdot J_2 \cdot J_3$, $K=K_1 \cdot K_2 \cdot K_3$, 如图 4.2.1(c) 所示。在图(b)和图(c)所表示的逻辑符号中, “ Γ ”表示延时输出。

2. 状态转换表、特征方程、激励表、状态图和时序图

由表 4.2.1 所示的真值表可列出 JK 触发器的状态转换真值表如表 4.2.2 所示。由表可知, 输入端 J 和 K 允许任意取值, 其新状态都有确定的状态, 即输入端 J 和 K 的取值不存在约束条件, 这一特点使它比主从 RS 触发器的应用范围广得多。

主从 JK 触发器的特征方程可以由主从 RS 触发器的特征方程直接导出, 将 $R=KQ$ 和 $S=J\bar{Q}$ 代入时钟 RS 触发器的特征方程式(4.1.1)中, 即得到 JK 触发器的特征方程为

$$Q^{n+1}=S+\bar{R}Q=J\bar{Q}+\bar{K}Q=J\bar{Q}+\bar{K}Q \quad (4.2.1)$$

若将 $R=KQ$ 和 $S=J\bar{Q}$ 代入时钟 RS 触发器的约束条件方程, 则有

$$SR=J\bar{Q}KQ=0$$

表 4.2.2 JK 触发器的状态转换真值表

Q	J	K	Q^{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

上式表明无论 J 和 K 的取值如何, 方程都成立, 即主从 JK 触发器的输入信号不存在约束条件。激励表可由状态转换表 4.2.2 导出, 如表 4.2.3 所示。由表 4.2.3 可得状态图, 如图 4.2.2 所示。

表 4.2.3 JK 触发器激励表

Q	Q^{n+1}	J	K
0	0	0	ϕ
0	1	1	ϕ
1	0	ϕ	1
1	1	ϕ	0

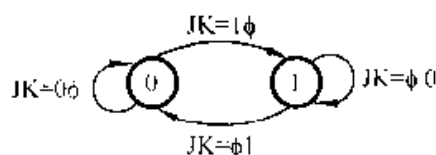


图 4.2.2 JK 触发器状态图

图 4.2.3 为工作波形图, 图中表明, 在 CP 脉冲下跳时刻, 由 J 、 K 端的取值确定了新状态。

3. 主从 JK 触发器的一次翻转现象

使用主从 JK 触发器, 要求其 JK 输入端的信号在 CP 脉冲作用期间保持恒定, 以免产生一次翻转现象, 造成永久性的逻辑错误。一次翻转和上述的空翻是两种不同的现象。所谓一次翻转现象是指, 在 CP 脉冲作用期间输入信号变化所发生的翻转。现举例说明之。

设触发器状态 $Q=0$, 主触发器状态 $Q'=0$; $J=0$, $K=1$ 。在正常工作情况下, 在 CP 脉冲作用后, Q 应维持 0 状态。现在的情况是在 CP 脉冲作用期间, 在 J 端发生一个正脉冲干扰。在 J 端信号由 0 变为 1 时, 主触发器 Q' 为 1 状态; 当 J 端信号由 1 变为 0 时, Q' 不能返回到 0 状态, 而是维持在 1 状态, 因为 $Q=0$ (见图 4.2.1), 门 8 关闭、门 7 开启, J 端上的正脉冲经门 7 倒相传输, 使门 5 和门 6 组成的基本 RS 触发器置 1。当这个正脉冲结束时, 基本 RS 触发器的输入为 11 (即门 7 和门 8 输出均为 1), 所以它的状态维持 1 状态, 即 $Q'=1$ 。在 CP 脉冲下跳沿时, 主触发器的状态传送到从触发器,

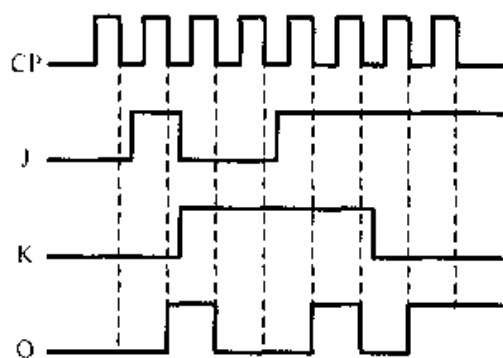


图 4.2.3 JK 触发器的工作波形

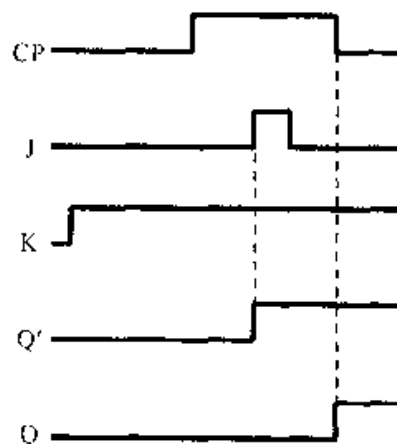


图 4.2.4 JK 触发器的一次翻转

所以, $Q=1$, 即在 J 端的正脉冲干扰引起了主从 JK 触发器的一次翻转, 导致了永久性的逻辑错误, 工作波形如图 4.2.4 所示。为避免之, 则要求在 CP 脉冲作用期间输入信号保持恒定, 这就使主从 JK 触发器的抗干扰能力下降, 所以, 这类触发器适合于窄脉冲触发的条件下工作。

4.3 维持阻塞型 D 触发器

1. 电路结构及逻辑功能

图 4.3.1(a)为维持阻塞 D 触发器 (简称维阻 D 触发器) 逻辑电路图, 图(b)、(c)是逻辑符号。

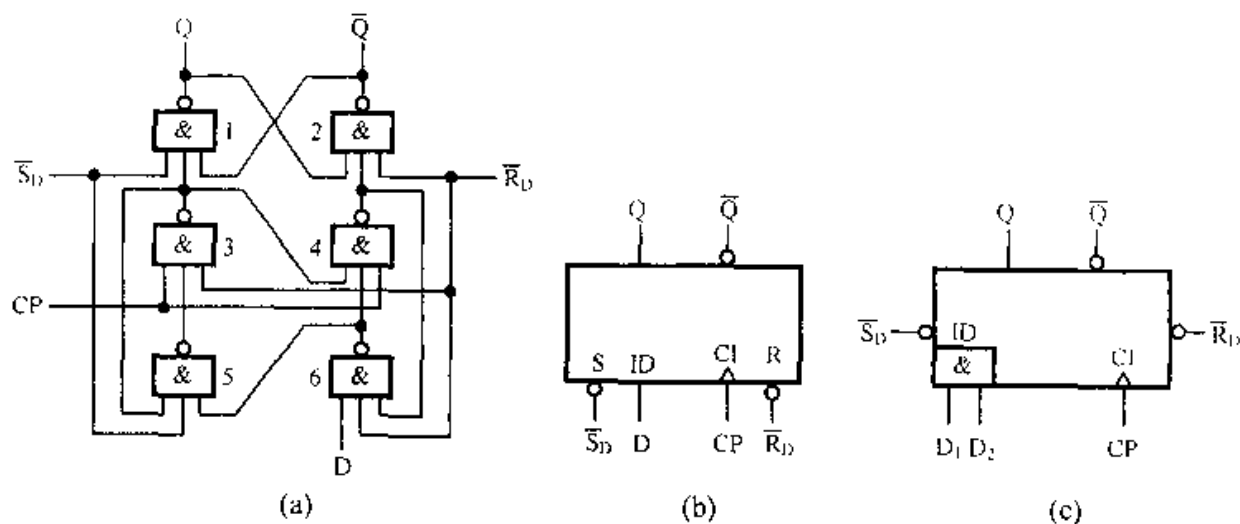


图 4.3.1 维持阻塞 D 触发器

(a) 逻辑电路图 (b) 逻辑符号之一 (c) 逻辑符号之二

电路由 6 个与非门构成, 其中, 门 1、门 2 构成基本 RS 触发器, 门 3~门 6 构成维持阻塞触发方式的导引电路, D 为信号输入端, CP 为时钟脉冲输入端, \bar{R}_D 和 \bar{S}_D 为异步置 0 端和异步置 1 端。

逻辑功能分析如下。

(a) 在 $CP=0$ 时, 门 3 和门 4 被封锁, 它们的输出均为 1, 基本 RS 触发器保持状态不变, 换言之, 若无 CP 脉冲作用, D 触发器保持状态不变。

(b) 设 $D=0$: 在 $CP=0$ 时, 门 3 和门 4 被封锁, 它们的输出 Q_3 和 Q_4 均为 1, 又由于 $D=0$, 所以 $Q_6=1$, $Q_5=0$, 对于 CP 脉冲而言, 门 3 被封锁, 门 4 是开启的。

在 $CP=1$, 即 CP 脉冲出现时, 经门 4 倒相在 Q_4 端输出置 0 负脉冲使基本 RS 触发器置 0, 即 $Q=0$ 。由于 Q_4 端输出的负脉冲与 CP 脉冲具有相同宽度, 它一方面使基本 RS 触发器置 0, 同时又通过反馈线加在门 6 的一个输入端, 使在 $CP=1$ 期间 Q_6 保

持 1, 而门 5 的两个输入均为 1(\bar{S}_D 、 \bar{R}_D 平时都是 1 电平, 这里暂不讨论), 所以 $Q_5=0$ 。这样, 就保证了在整个 CP 脉冲高电平期间, 门 4 总是开启, 而门 3 总是关闭, 即 Q_4 端输出的负脉冲既维持了置 0 负脉冲的存在, 又阻塞了产生置 1 负脉冲的可能。因此在 CP=1 期间, 即使 D 端的值发生变化也不会对触发器的状态产生影响。当 CP 脉冲由 1 变到 0 之后, 触发器便维持 0 态, 通常把 Q_4 至门 6 输入端的连线称为置 0 维持线, 而把 Q_6 至门 5 输入端的连线称为置 1 阻塞线。

(c) 设 $D=1$: 当 CP=0 时, $Q_6=0$, $Q_5=1$, 所以对 CP 脉冲而言, 门 3 是开启的, 而门 4 被封锁。CP 脉冲出现之后, 经门 3 倒相在 Q_3 端输出置 1 负脉冲, 使基本 RS 触发器置 1, 同样的分析可知, 在 CP=1 期间, Q_3 端输出的负脉冲既维持 Q_3 为 1, 保证门 3 总是开启, 又通过 Q_3 至门 4 输入端的反馈线阻塞了 Q_4 端产生置 0 负脉冲的可能。在 CP 脉冲由 1 变为 0 之后, 触发器便维持 1 态。通常把 Q_3 至门 5 输入端的连线称为置 1 维持线, 而将 Q_3 至门 4 输入端的连线称为置 0 阻塞线。

综上所述, 该类触发器由于采用了维持阻塞结构, 在 CP 脉冲上跳沿到达之后, 经几个门的延迟便完成了状态转换, 在 CP=1 期间允许 D 端状态改变, 对触发器状态不会发生影响。所以, 每来一个 CP 脉冲, 触发器不可能产生两次或多次翻转, 因而克服了空翻。此外, 这种触发器是在 CP 脉冲上跳沿翻转, 新状态取决于 CP 脉冲上跳时刻 D 的状态, 但是, D 的状态只需维持极短暂时间(见 4.5 节), 一旦状态翻转之后允许 D 的状态改变, 所以, 它是属于边沿型触发器。在图 4.3.1(b)所示的逻辑符号中, CP 输入端处加有动态符号“ \wedge ”。维持阻塞 D 触发器的抗干扰能力比主从 JK 触发器强。图 4.3.1(c)所示的是具有多输入端的维持阻塞 D 触发器逻辑符号。

D 触发器的真值表和状态转换真值表分别示于表 4.3.1 和表 4.3.2, 其特征方程为

$$Q^{n+1}=D \tag{4.3.1}$$

由于它的新状态就是前一时刻输入状态, 故又称为数据触发器或延迟触发器。

表 4.3.1 D 触发器真值表

D	Q^{n+1}
0	0
1	1

表 4.3.2 D 触发器状态转换真值表

Q	D	Q^{n+1}
0	0	0
0	1	1
1	0	0
1	1	1

由表 4.3.2 可得状态图, 如图 4.3.2 所示。工作波形图示于图 4.3.3, 图中表明: 在 CP 脉冲上跳时刻, 由 D 端的状态确定了新状态。

2. 异步置 1 端和置 0 端

实际的触发器产品都设置了异步置 1 端 \bar{S}_D 和异步置 0 端 \bar{R}_D , 它们都是低电平

有效(负脉冲触发), 无论触发器状态如何以及有无 CP 脉冲出现, 只要在 \bar{R}_D (\bar{S}_D) 端加负脉冲则触发器置 0(1)。

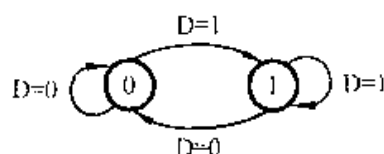


图 4.3.2 D 触发器的状态转换图

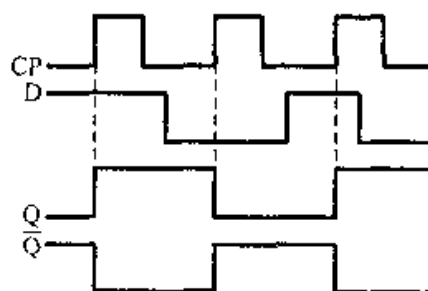


图 4.3.3 D 触发器的工作波形

为了满足在 $CP=0$ 及 $CP=1$ 时都能置 0 或置 1, 置 0(1)负脉冲仅加在门 2(门 1)是不行的。例如, $CP=1$ 、 $D=1$, 此时触发器状态 $Q=1$, 并且 $Q_3=0$, 如在 \bar{R}_D 端加置 0 负脉冲, 就会得到 $Q=\bar{Q}=1$, 当置 0 负脉冲消失之后触发器仍维持 1 态, 故未达到异步置 0 之目的。所以, 要将置 0 负脉冲同时加在门 2、门 3 和门 6, 才能得到 $Q_3=1$ 和 $Q_6=1$, 从而保证在 $CP=1$ 时也可实现异步置 0。同时, 把置 1 负脉冲加在门 1 和门 4 才能保证可靠地实现异步置 1。

4.4 T 和 T' 触发器

4.4.1 T 触发器

T 触发器是逻辑设计中使用较多的一种触发器, 但是, 一般不生产这样的产品, 因为它可以由主从 JK 触发器或维持阻塞 D 触发器转换得到(参见 4.6 节), 相应的逻辑符号分别如图 4.4.1(a)和(b)所示, T 为信号输入端。

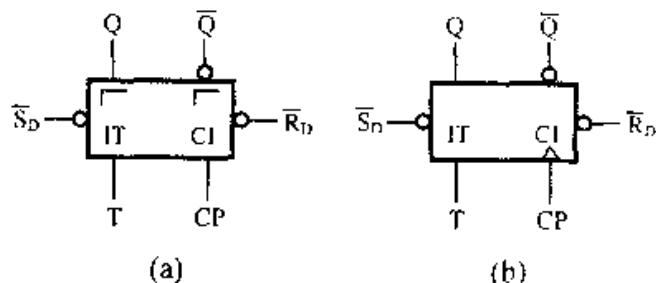


图 4.4.1 T 触发器

(a) 下降沿触发 (b) 上升沿触发

当 $T=0$ 时, 在 CP 作用后, 其状态保持不变即 $Q^{n+1}=Q$ 。当 $T=1$ 时, 在 CP 作用

后, 其状态改变, 即 $Q^{n+1} = \bar{Q}$ 。由此可以直接写出特征方程

$$Q^{n+1} = T\bar{Q} + \bar{T}Q = T \oplus Q \quad (4.4.1)$$

T 触发器的真值表和状态转换真值表分别示于表 4.4.1 和表 4.4.2。

由于 T 触发器在 $T=0$ 时保持状态不变, 在 $T=1$ 时构成计数(翻转)触发器, 所以, T 触发器又称为受控计数(翻转)触发器。

表 4.4.1 T 触发器真值表

T	Q^{n+1}
0	Q^n
1	\bar{Q}^n

表 4.4.2 T 触发器状态转换真值表

Q	T	Q^{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

4.4.2 T' 触发器

T' 触发器的逻辑功能是, 每来一个 CP 脉冲, 触发器状态改变一次, 所以, T' 触发器也就是计数触发器, 其特征方程为 $Q^{n+1} = \bar{Q}$ 。

由 JK 触发器、D 触发器、T 触发器可方便地构成 T' 触发器, 如图 4.4.2 所示。

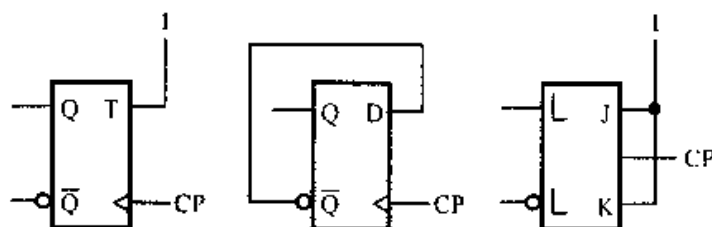


图 4.4.2 T' 触发器

4.5 边沿型触发器

主从 JK 触发器在 $CP=1$ 期间要求 J、K 端输入状态保持不变, 否则, 可能引起错误翻转(即所谓“一次翻转”)。维持阻塞 D 触发器的新状态仅与 CP 脉冲上跳沿附近的 D 端状态有关, 所以, D 触发器属于边沿型触发器, 在 $CP=0$ 或 $CP=1$ 期间 D 端状态的变化不会对触发器的状态产生影响, 故抗干扰能力比主从 JK 触发器强些。近年来又发展了一种比 D 触发器对输入信号要求更低的边沿型触发器。依据触发方式不同, 可将其分为正边沿或负边沿触发器。

4.5.1 TTL 边沿型 JK 触发器

图 4.5.1(a)为负边沿型 JK 触发器的逻辑电路图；图(b)是逻辑符号，在 CP 输入端处加有动态符号“^”和小圆圈，表明它是负边沿触发方式。

该触发器由门 1 和门 2 构成输入信号的接收门；门 3~门 5、门 6~门 8 分别构成两个与或非门。其工作过程分析如下。

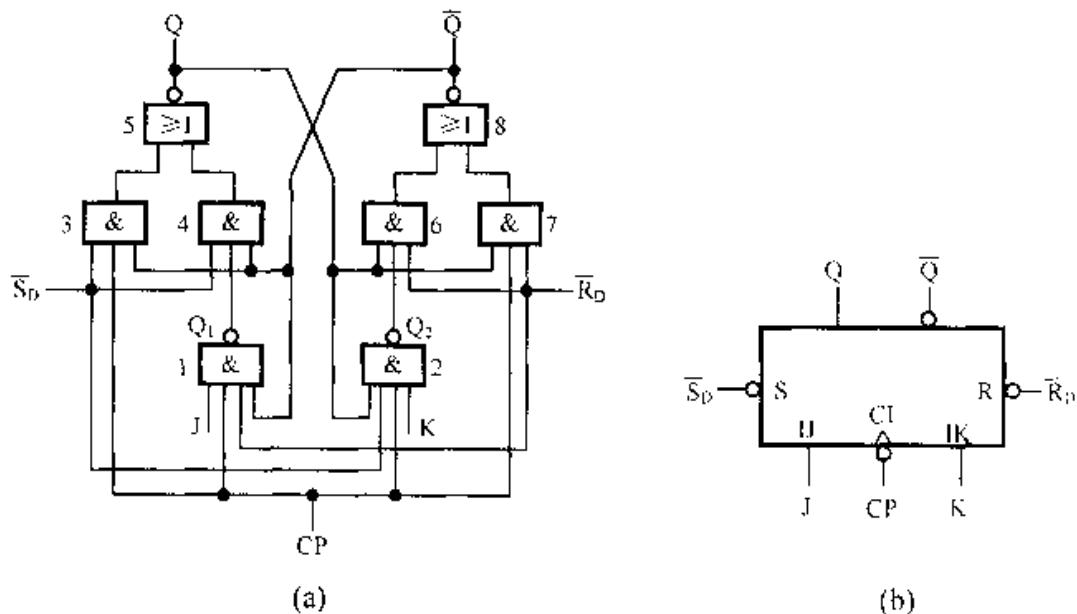


图 4.5.1 负边沿型 JK 触发器

(a) 逻辑电路图 (b) 逻辑符号

(1) CP=0 时，触发器状态不变

CP 为 0 时，门 1 和门 2 关闭，所以 $Q_1=Q_2=1$ ，此时门 3 和门 7 也被关闭，两个与或非门构成基本 RS 触发器。 Q_1 端相当于 S， Q_2 端相当于 R。由于 $R=S=1$ ，所以触发器状态不变。

(2) CP 由 0 变为 1 时，触发器状态不变，为接收输入信号作准备

设触发器 $Q=0$ ， $\bar{R}_D=\bar{S}_D=1$ ， $J=1$ ， $K=0$ 。当 CP 由 0 变 1 时，有两个过程同时进行。一方面是门 1 和门 2 开启，但由于 $Q=0$ ， Q_2 仍为 1；而 $\bar{Q}=1$ ， $J=1$ ，故 Q_1 由 1 变为 0，经与门 4 延迟到达或非门 5 输入端。另一方面门 3 也开启，即 CP 由 0 变为 1，经与门 3 延迟到达或非门 5 的另一个输入端，但是，前者延迟时间长（与非门 1 及与门 4 延迟的总和），后者延迟时间短（与门 3 延迟），所以，在门 4 输出由 1 变 0 之前，门 3 输出早已由 0 变为 1，这样保证了 Q 维持 0 不变。同理分析当 $Q=1$ 时，状态也维持不变。

当 CP=1 时，门 1 和门 2 的输出分别为

$$\left. \begin{aligned} Q_1 &= \overline{JQ} \\ Q_2 &= \overline{KQ} \end{aligned} \right\} \quad (4.5.1)$$

由于 Q 和 \overline{Q} 中总有一个为 0，故 Q_1 和 Q_2 可能同时为 1，但不会同时为 0。此时，依据 J 、 K 的数据和原状态 Q 由式(4.5.1)确定接收门 1 和门 2 的输出，为触发器的新状态准备了条件。

(3) CP 由 1 变 0，触发器翻转

设 CP 脉冲下跳沿到来前瞬间 $J=1$ ， $K=0$ 。当 CP 下跳沿到来时，一方面将门 3 和门 7 关闭，两个与或非门又构成基本 RS 触发器，另一方面门 1 和门 2 也被封锁， Q_1 、 Q_2 都要变为 1。但是，由于门 1、门 2 的延迟时间比门 3、门 7 的长些，就使门 3 的输出在由 1 变 0 后的一段短暂时间内， Q_1 和 Q_2 仍保持下跳沿到来之前的状态，并由此决定触发器的新状态。此时， Q_1 相当于基本 RS 触发器的 S 端， Q_2 相当于 R 端，由式(4.5.1)确定 $S=Q_1=\overline{JQ}=0$ ； $R=Q_2=\overline{KQ}=1$ ，所以，触发器新状态为 $Q^{n+1}=1$ 。同理，如果在 CP 下跳沿到来前瞬间的输入条件(Q, J, K)使接收门的状态为 $Q_1=Q_2=1$ ，则 $Q^{n+1}=Q$ ；若 $Q_1=1$ 、 $Q_2=0$ ，则 $Q^{n+1}=0$ 。

*4.5.2 CMOS 边沿型 D 触发器

CMOS 触发器具有功耗低、集成度高、抗干扰能力强和成本低廉等优点。其最高工作频率略低于 TTL 型触发器，典型值为 10~50MHz。这类触发器一般都是采用边沿型触发方式和主从电路结构。

CMOS 边沿型 D 触发器的逻辑电路示于图 4.5.2(a)，图(b)是逻辑符号。在图(b)中，CP 输入端处未加小圆圈，表明它是正边沿型 D 触发器。在 S_D 与 R_D 输入端处未加小圆圈，而且字符上没有非号“—”，表明异步复位及异步置位均是高电平有效。

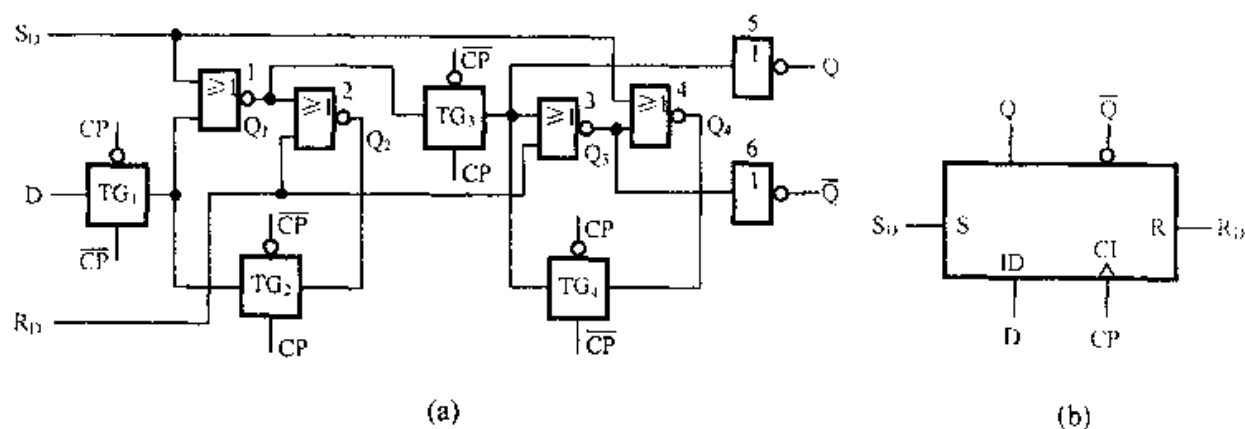


图 4.5.2 CMOS D 触发器

(a) 逻辑电路图 (b) 逻辑符号

电路由主、从触发器构成。主触发器由或非门 1、2 和传输门 TG_2 构成；从触发器由门 3、4 和传输门 TG_4 构成。 TG_1 和 TG_3 分别是主触发器和从触发器的输入控制门。图中还画出了异步置 0(1)端 $R_D(S_D)$ 的连接方法，应注意它们是高电平有效。

当 $CP=0$ 、 $\overline{CP}=1$ 时， TG_1 导通、 TG_2 截止。D 端的数据送入主触发器中， $Q_2=D$ 。但这时由于 TG_2 截止而未构成反馈连接，不能自行保持，所以， Q_2 跟随 D 端的状态变化。同时，由于 TG_3 截止、 TG_4 导通，所以，从触发器状态维持不变，而且它与主触发器之间的联系被 TG_3 切断，保证了 D 端的状态变化不会影响从触发器的状态。

当 CP 脉冲上升沿到达时， $CP=1$ ， $\overline{CP}=0$ ，所以 TG_1 截止、 TG_2 导通。由门 1、门 2 和 TG_2 构成主触发器，又因为门 1 输入电容的存储效应，门 1 输入端的电压不会立刻消失，所以，在 CP 脉冲上升沿到来之前的状态被保存下来。同时，由于 TG_3 导通、 TG_4 截止，主触发器的状态便传送到输出端，即

$$Q^{n+1} = \overline{Q_1^{n+1}} = \overline{D} = D$$

在 $CP=1$ 期间， $Q^{n+1}=D$ ，实现 D 触发器功能。当 CP 脉冲再由 1 变为 0 时，传输门 TG_2 和 TG_3 又断开，而 TG_1 和 TG_4 又导通。 TG_2 断开为或非门 1 和 2 接收新数据提供了条件，而 TG_1 导通使新数据输入到或非门 1 和 2；与此同时， TG_3 断开使主、从触发器通道切断，防止新数据可能使触发器状态发生变化， TG_4 导通则使从触发器恢复记忆，触发器维持翻转后的状态，并且在 $CP=0$ 期间不变。

综上所述，这种触发器是按主从方式工作，在 $CP=0$ 时接收数据，在 CP 脉冲上跳沿状态翻转，在 $CP=1$ 期间状态维持不变。由于触发器只接收上跳沿到达时的 D 端数据，因此，防止了在 $CP=1$ 期间因 D 端数据变化而出现的错误，这种触发器属于正边沿触发型触发器。

R_D 和 S_D 是异步置 0 端和异步置 1 端，平时都处于低电平，只要在 R_D 或 S_D 端加高电平(即正脉冲触发)，不论此时 D 和 CP 端状态如何，触发器将被置 0 或置 1。例如， $R_D=0$ 、 $S_D=1$ 。若 $CP=0$ ，则 CMOS 传输门 TG_1 、 TG_4 导通， $S_D=1$ 经或非门 4、传输门 TG_4 、缓冲门 5 送到 Q 端； $R_D=0$ 经过或非门 3、缓冲门 6 送到 \overline{Q} 端，实现异步置 1。此时， TG_3 是断开的，D 端的变化不会影响触发器的状态。同样，若 $CP=1$ ，则 TG_2 、 TG_3 导通， $S_D=1$ 经或非门 1、 TG_3 、缓冲门 5 送到 Q 端； $R_D=0$ 经或非门 3、缓冲门 6 送到 \overline{Q} 端，实现异步置 1，此时， TG_1 是断开的，排除了 D 端变化对置 1 的影响。

*4.5.3 CMOS 边沿型 JK 触发器

在图 4.5.2(a)所示的 CMOS 边沿型 D 触发器的输入端上增加一个组合电路，使 J、K 输入端信号经组合电路送到 D 输入端，便构成图 4.5.3(a)所示的 CMOS 边沿型 JK

触发器,图(b)为逻辑符号。

由图 4.5.3(a)可写出

$$D = (J + Q)\overline{KQ} = J\overline{Q} + \overline{K}Q$$

所以

$$Q^{n+1} = D = J\overline{Q} + \overline{K}Q$$

上式为 JK 触发器的特征方程,故该触发器实现了 JK 触发器的功能。

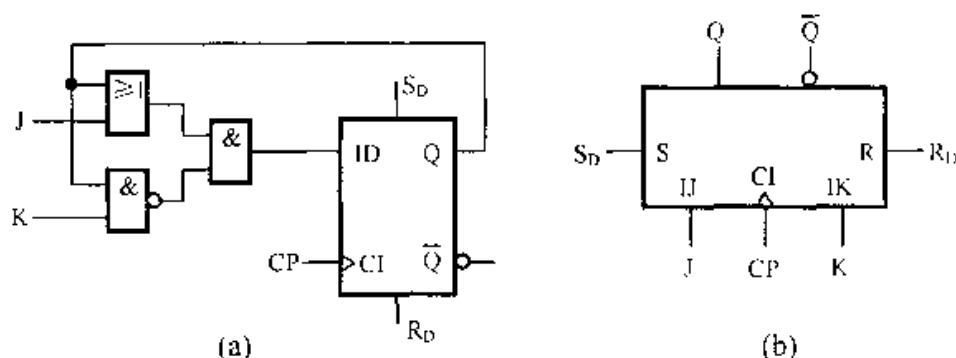


图 4.5.3 CMOS 边沿型 JK 触发器

(a) 逻辑框图 (b) 逻辑符号

4.6 集成触发器使用中的几个问题

4.6.1 集成触发器的选用

触发器种类繁多且各具特色,所以,必须依据实际要求正确地选用。基本 RS 触发器是异步触发器(无时钟信号),属于电位触发方式,它是构成各类触发器的基本电路形式。时钟 RS 触发器也是电位触发方式,电路结构简单,但是,输入信号有约束条件和空翻现象,所以,一般只能作数码寄存器和移位寄存器,它的重要性在于可以在它的基础上构成实用性强的 JK 触发器和 D 触发器;JK 触发器属于脉冲触发方式,一般采用主从结构,它有一次翻转现象,故抗干扰能力较弱;D 触发器采用维持阻塞结构,属于边沿型触发方式;边沿型 JK 触发器和边沿型 D 触发器具有更强的抗干扰能力。

CMOS 触发器的最大优点是功耗低和抗干扰能力强,但工作速度较 TTL 型低些;ECL 型触发器最大优点是工作速度高,其最高工作频率一般可达数百 MHz 以上,但功耗较大和抗干扰能力差。

在工作条件较差的场合,则应选用维持阻塞 D 触发器;在工作条件更为恶劣的场合,则应选用边沿型触发器;如果在功耗指标要求苛刻的场合,则应选用 CMOS

触发器：若要求工作频率达到几百 MHz,则必须选用 ECL 型触发器。

4.6.2 激励表的应用

在时序逻辑电路设计中，必须运用各类触发器的激励表，如表 4.6.1 所示。由表可见：对于 RS 和 JK 触发器而言，在完成某些转换时，只需其中一个输入信号有确定值，而另一个输入信号可为任意态 ϕ 。在运用激励表时必须注意这个特点，因它将有利于所设计的电路结构简化。

表 4.6.1 常用触发器激励表

Q	Q^{n+1}	S	R	J	K	D	T
0	0	0	ϕ	0	ϕ	0	0
0	1	1	0	1	ϕ	1	1
1	0	0	1	ϕ	1	0	1
1	1	ϕ	0	ϕ	0	1	0

4.6.3 触发器类型转换

在触发器实际应用中存在触发器之间相互转换的情况，例如，T 或 T' 触发器在逻辑电路设计中是比较常用的，但实际上没有现成产品，这就需要由 D 触发器或 JK 触发器转换。同时，通过触发器相互转换的讨论，可以使我们加深对各类触发器的真值表、激励表、逻辑功能及特征方程等内容的理解和运用。有些转换很简单，由逻辑功能的分析便可画出转换图，例如，JK \rightarrow T，D \rightarrow T'等。但有些转换需要增加少量门电路，不能直接由逻辑功能的分析便画出转换图，所以，需要介绍一种通用的转换方法。图 4.5.3(a)电路就是将 CMOS 边沿型 D 触发器转换为 CMOS 边沿型 JK 触发器的例子。

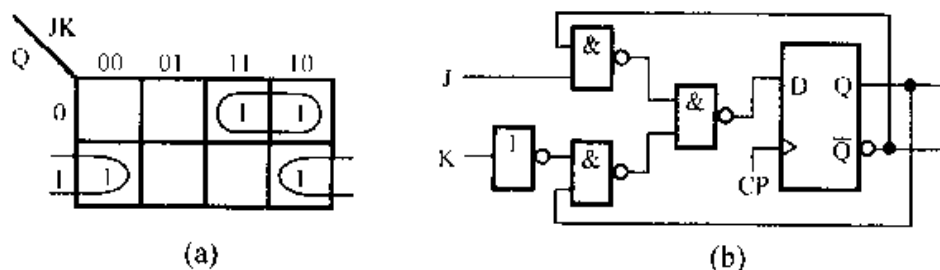


图 4.6.1 例 4.6.1 的卡诺图和逻辑电路图

(a) 卡诺图 (b) 逻辑电路图

例 4.6.1 试将 D 触发器转换为 JK 触发器。

解 (1) 列出 D 型和 JK 型的激励表, 见表 4.6.2。

(2) 写出 $D=f(Q,J,K)$, 为此作出 D 的卡诺图, 见图 4.6.1(a)。由图可得

$$D = J\bar{Q} + \bar{K}Q = \overline{JQ \cdot KQ}$$

(3) 画转换逻辑电路图, 见图 4.6.1(b)。

表 4.6.2 D 和 JK 触发器激励表

Q	Q^{n+1}	D	J	K
0	0	0	0	ϕ
0	1	1	1	ϕ
1	0	0	ϕ	1
1	1	1	ϕ	0

4.6.4 如何画工作波形

在实际调试逻辑电路时, 经常碰到给定各种输入波形, 要求画出触发器的工作波形的情况, 这时必须注意以下几点:

(a) 若提供了 \bar{R}_D 或 \bar{S}_D 信号, 则必须考虑 \bar{R}_D 或 \bar{S}_D 信号的异步清 0 或置 1 功能。

(b) 不同类型的触发器的翻转时刻不同, 主从 JK 型在 CP 脉冲下跳沿时状态翻转, 维阻 D 型在 CP 脉冲上跳沿时状态翻转, T 型则由转换的触发器确定。CMOS 主从 D 型和 CMOS 主从 JK 型都是在 CP 脉冲上跳沿时状态翻转。

(c) 确定触发器输入端的状态, 并注意它和 CP 脉冲的关系。在不要求考虑触发器中门电路的延迟时间的条件下, 如时钟脉冲与输入值的跳变发生在同一时刻, 则输入值应取跳变前的状态。

例 4.6.2 电路和输入信号如图 4.6.2(a)和(b)所示, 试对应输入波形画出 Q_1 和 Q_2 端输出波形。

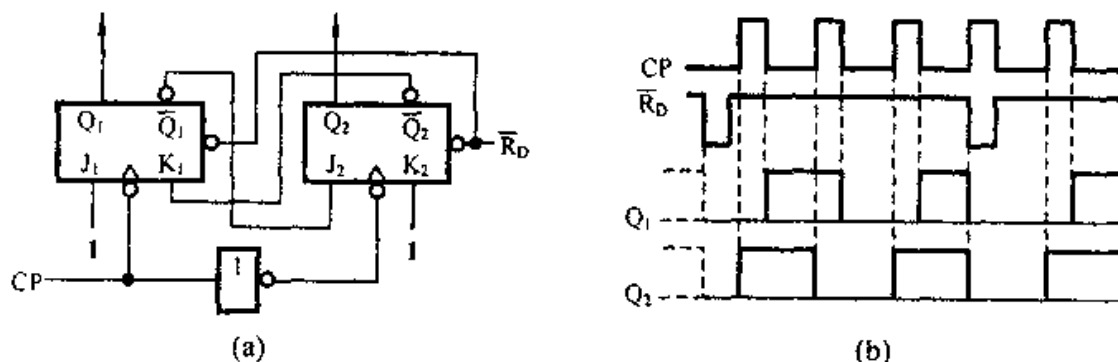


图 4.6.2 例 4.6.2 的逻辑电路图和波形图

(a) 逻辑电路图 (b) 波形图

解 由电路写出 $J_1=1$, $K_1=\bar{Q}_2$; $J_2=\bar{Q}_1$, $K_2=1$ 。输入信号 \bar{R}_D 的异步置 0 功能使两个触发器无论起始状态如何, 在 \bar{R}_D 端第 1 个负脉冲出现时都被置 0。触发器(1)在 CP 脉冲下跳沿翻转, 而触发器(2)在 CP 脉冲上跳沿翻转, 此外应注意 \bar{R}_D 端的第 2 个负脉冲使两个触发器被异步置 0, 并且第 4 个 CP 脉冲下跳沿与第 2 个异步置 0

负脉冲的跳变发生在同一时刻, 此时应取跳变前的值, 即 $\bar{R}_D = 0$ 。考虑以上因素画出 Q_1 和 Q_2 端的输出波形如图 4.6.2(b) 所示。

例 4.6.3 设主从 JK 触发器的起始状态为 0, 各输入端的波形如图 4.6.3 所示, 试画出其输出端 Q 的波形。

解 本例有异步复位信号 \bar{R}_D 和异步置位信号 \bar{S}_D , 所以, 要考虑它们的复位和置位功能。主从 JK 触发器是在 CP 脉冲下跳沿发生状态转换, CP 脉冲下跳沿有与输入端状态的变化发生在同一时刻的情况, 这时, 输入端状态应选用跳变前一时刻的状态。考虑以上因素, Q 端波形如图 4.6.3 所示。由于第一个 CP 下跳沿同 \bar{R}_D 信号跳变发生在同一时刻, 所以 \bar{R}_D 应取 0, Q 仍维持 0 状态; 第二个 CP 下跳沿同 \bar{S}_D 信号跳变发生在同一时刻, 所以 \bar{S}_D 应取 0, Q 仍被置 1; 第三个 CP 下跳沿时, $J=K=1$, 所以 Q 端状态由 1 变为 0。请读者注意: 第五个 CP 下跳沿同 \bar{S}_D 跳变发生在同一时刻, \bar{S}_D 应取 1, 而且 $J=K=1$, 所以 Q 本应由 1 变为 0, 但是, 随后 \bar{S}_D 为 0 电平使 Q 又置 1, 所以 Q 端维持 1 状态不变。

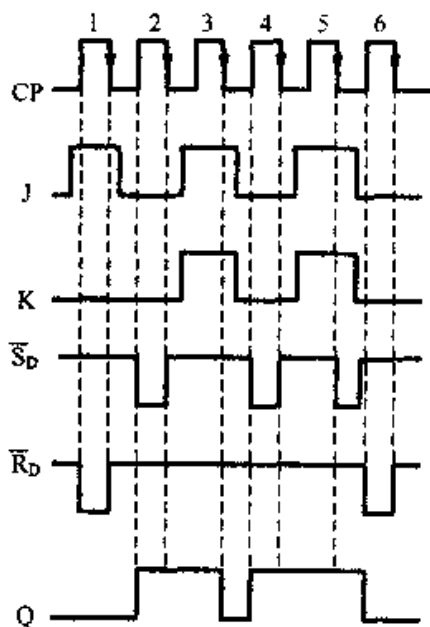


图 4.6.3 例 4.6.3 的波形图

例 4.6.4 电路和输入波形如图 4.6.4(a) 和 (b) 所示, 试画出 Q_1 和 Q_2 端输出波形。设起始状态为 $Q_2Q_1=11$ 。

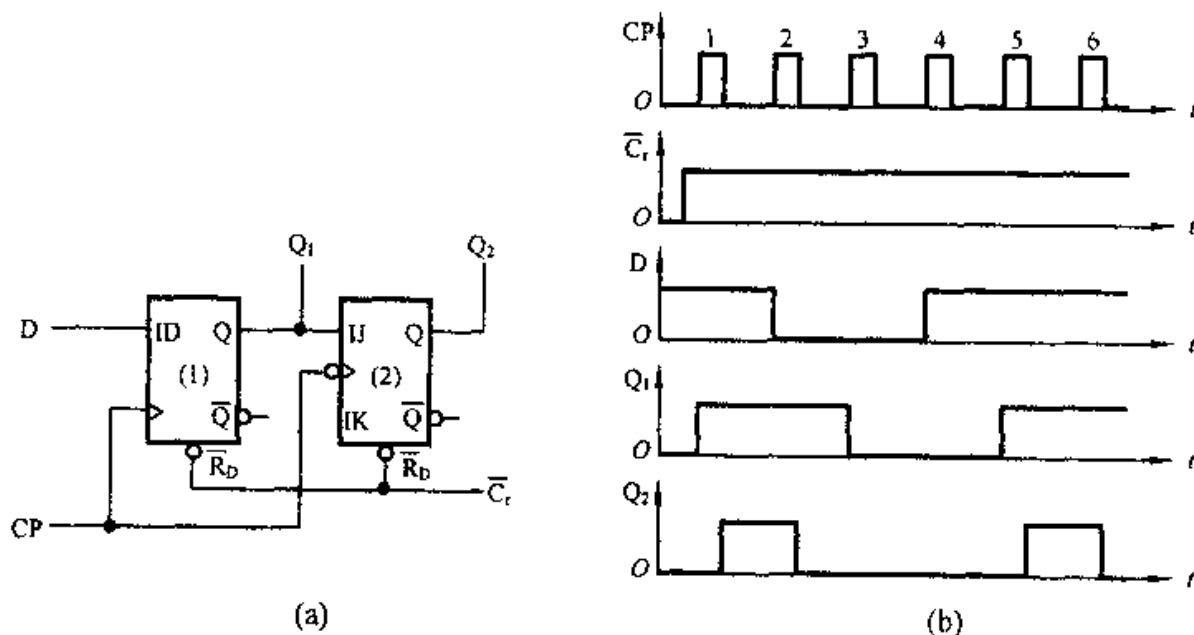


图 4.6.4 例 4.6.4 的波形图

解 因为 $D_1=D$; $J_2=Q_1$ 和 $K_2=1$; \bar{C}_r 信号对两个触发器都有复位功能; D 触发

器是上跳沿触发,而 JK 触发器是下跳沿触发。考虑以上因素, Q_2 和 Q_1 端输出波形如图 4.6.4(b)所示。

小 结

触发器是时序电路中最常用的存储单元,它有两个稳定状态,在一定的输入条件下,两个稳定状态之间可以相互转换。

触发器种类繁多,各具特色。依据逻辑功能可分为基本 RS 触发器、时钟 RS 触发器、JK 触发器、D 触发器、T 和 T'触发器等。讨论它们的真值表、状态转换表、特征方程和激励表是本章的重点内容。

目前市场上供应的产品多为 D 或 JK 触发器,而 T 和 T'触发器在时序电路设计中也有广泛用途,所以,要掌握由 D 或 JK 触发器转换为 T 或 T'触发器的方法。

为了改善触发器的某些性能,新产品不断出现,例如,边沿型触发器、CMOS 触发器和 ECL 触发器等,它们具有独特的优点,所以,应依据实际情况正确地选用。

画出工作波形是时序电路调试中的一个重要方法,如何正确地画出工作波形也是本章重点讨论的内容。

思考题和习题

- 4.1 试述 RS 触发器、D 触发器、JK 触发器、T 和 T'触发器的逻辑功能。
- 4.2 触发器的状态转换表和激励表如何构成,试举例说明之。
- 4.3 试举例说明触发器的状态图的读图方法。
- 4.4 试举例说明触发器的“不定”状态,哪些类型的触发器存在“不定”状态?
- 4.5 在画由触发器和集成门构成的电路工作波形时,应注意些什么?
- 4.6 采用主从或维持阻塞结构的触发器为什么能克服空翻?
- 4.7 边沿触发器和 CMOS 触发器各具有什么独特的优点和缺点?
- 4.8 一个由与非门组成的时钟 RS 触发器,当 R 和 S 端加有如图 P4.1 所示波形时,分别画出 Q 和 \bar{Q} 的波形(不计 t_{pd})。
- 4.9 设主从 JK 触发器的原状态为 1,当 J、K 和 CP 端分别加入如图 P4.2 所示的波形时,试画出输出端 Q 之波形。
- 4.10 设主从 JK 触发器的原状态为 0,各输入波形如图 P4.3 所示,试画出 Q 端之波形。
- 4.11 试画出图 P4.4 所示电路中 A、B、 Q_1 、 \bar{Q}_1 、C、D、 Q_2 及 \bar{Q}_2 各点的波形(设起始状态 $Q_1=0$ 、 $Q_2=0$)。

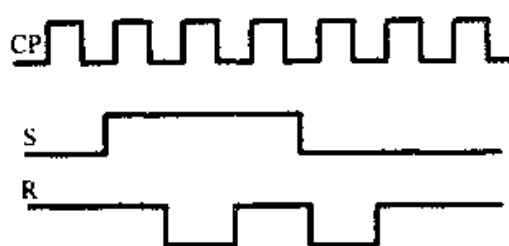


图 P4.1

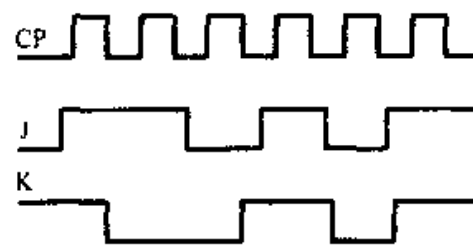


图 P4.2

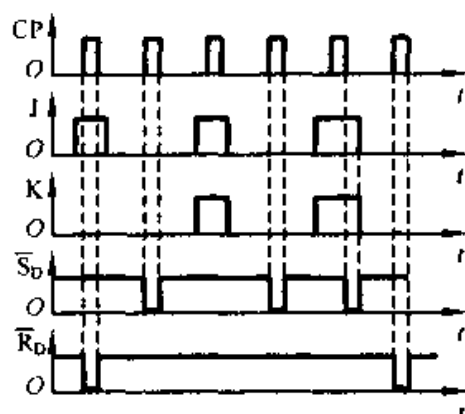


图 P4.3

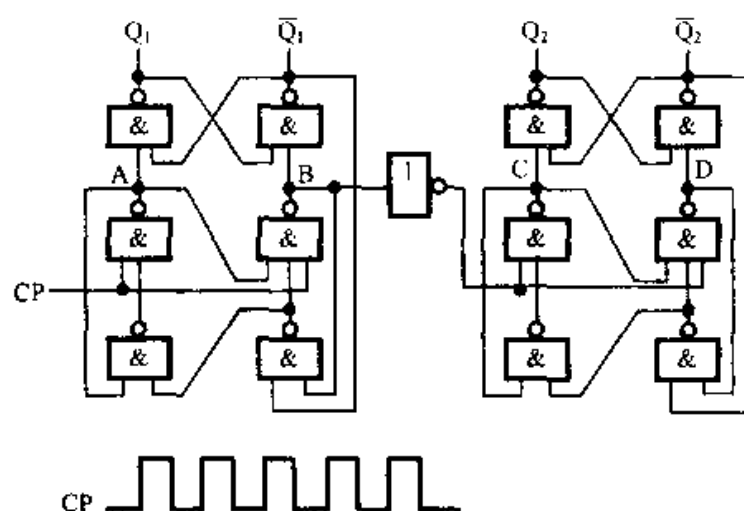


图 P4.4

- 4.12 根据图 P4.5 中所示的电路及相应的 CP、 \bar{R}_D 和 D 的波形，画出 Q_1 、 Q_2 之波形。
- 4.13 采用触发器的激励表，试由无空翻现象的 RS 触发器和门电路来构成 T 触发器。
- 4.14 试画出图 P4.6 所示电路中 B、C 端的波形(输入端 A 及 CP 之波形如图所示，设起始状态 $Q_1=0$ 、 $Q_2=0$)。

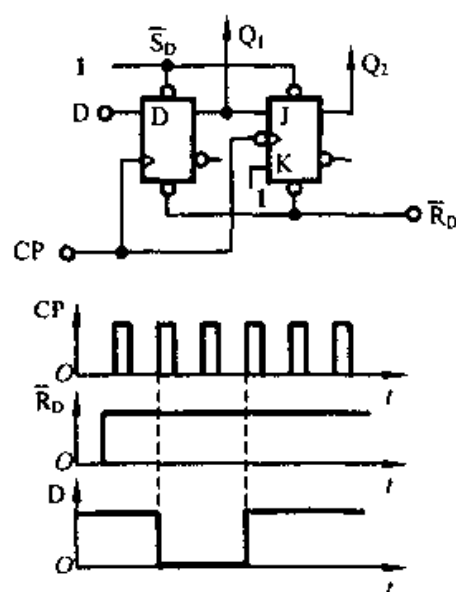


图 P4.5

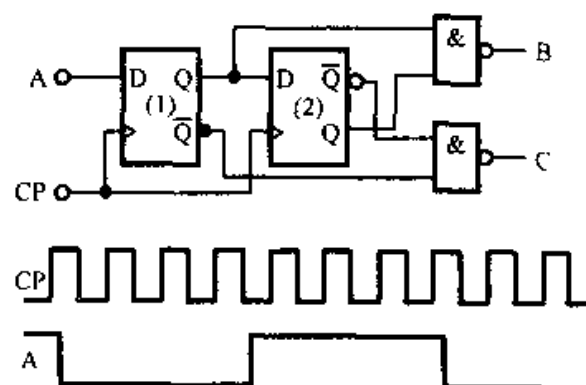


图 P4.6

4.15 电路和输入波形如图 P4.7 所示, 试画出 Q_1 、 Q_2 和 F 点波形(设初态 $Q_A Q_B = 11$)。

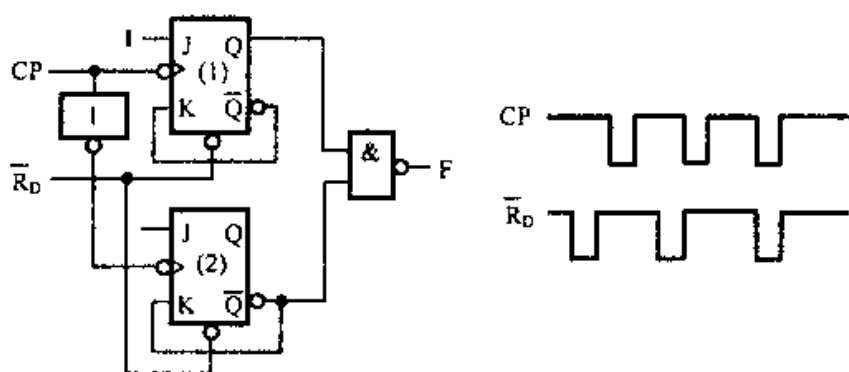


图 P4.7

4.16 电路和输入波形如图 P4.8 所示, 试画出 Q_1 、 Q_2 和 F 点波形。

4.17 电路和输入波形如图 P4.9 所示, 设初态为 $Q_1 Q_2 = 11$, 试画出 Q_1 和 Q_2 点波形。

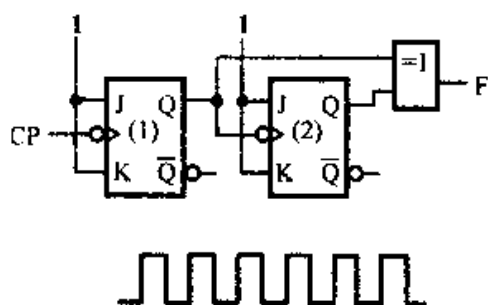


图 P4.8

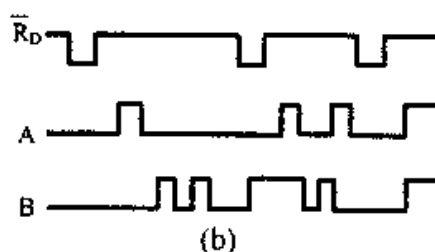
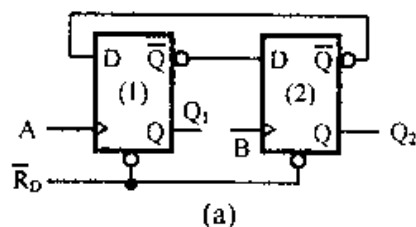


图 P4.9

4.18 在图 P4.10(a)所示触发器的 CP 端, 输入图(b)所示信号, 试画出各触发器之 Q 端在初态为 0 及 1 两种条件下的波形。

4.19 试分析图 P4.11 所示触发器的功能, 写出其特征方程。

4.20 试分析图 P4.12 所示锁存触发器的逻辑功能, 列出其真值表。

4.21 试将 JK 触发器转换为 T 触发器。

4.22 试将 JK 触发器转换为 D 触发器。

4.23 将图 P4.13 所示波形分别加到下列 3 个触发器, 试分别画出 Q 端波形(设初态均为 0)。

(1) 正边沿 JK 触发器;

(2) 负边沿 JK 触发器。

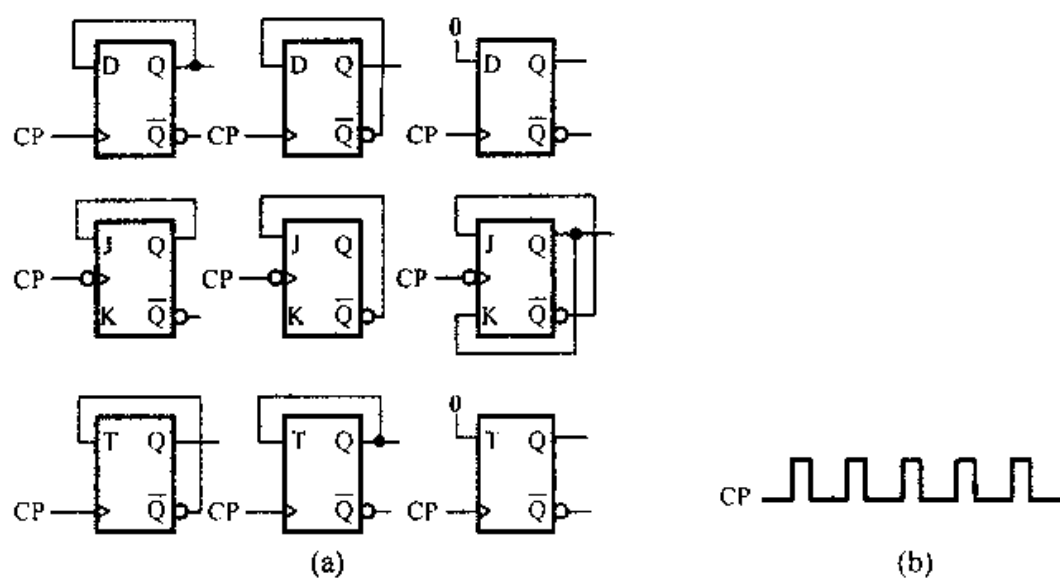


图 P4.10

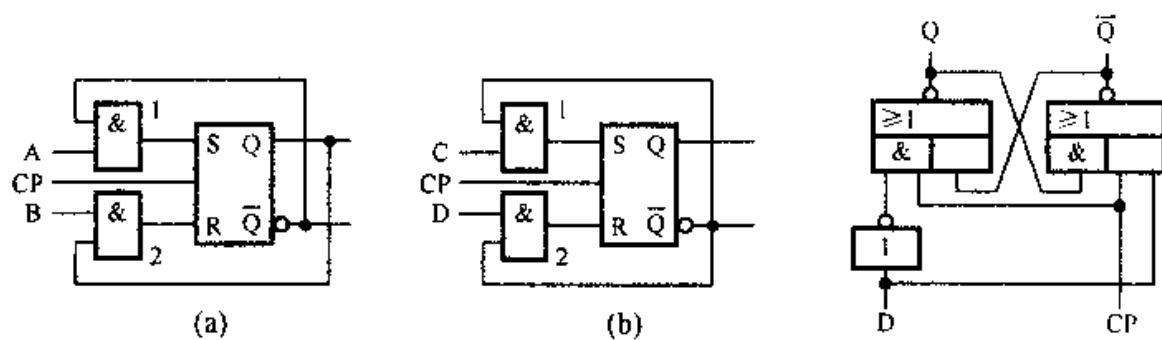


图 P4.11

图 P4.12

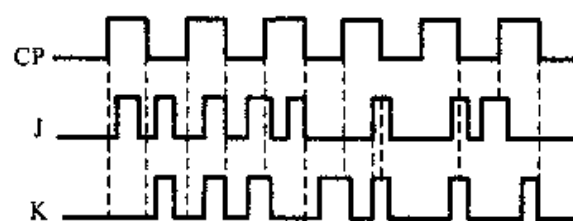


图 P4.13

第 5 章 时序逻辑电路

内容提要 本章重点讨论时序逻辑电路的分析方法和设计方法。对常用的时序逻辑部件及其中规模集成组件——寄存器、移位寄存器、计数器和序列信号发生器等逐一进行讨论，重点介绍它们的电路结构、工作原理、使用方法和功能扩展方法，同时也介绍它们在逻辑电路设计中的应用。

5.1 时序逻辑电路概述

所谓时序逻辑电路(简称时序电路)是指其稳定输出不仅与该时刻的输入状态有关，而且还与过去的输入状态有关的逻辑电路。因此，在时序电路中，除了有反映现在输入状态的组合电路之外，还应包含能记忆过去状态的存储电路。

时序电路框图如图 5.1.1 所示，它由组合电路和存储电路两部分组成。组合电路的输出包括外输出和内输出，外输出 Z_1, \dots, Z_m 作为整个时序电路的输出；内输出 W_1, \dots, W_k 作为存储电路的输入。组合电路的输入也包括两部分，外输入 X_1, \dots, X_n 是整个时序电路的输入，而内输入 Y_1, \dots, Y_k 是存储电路的输出。

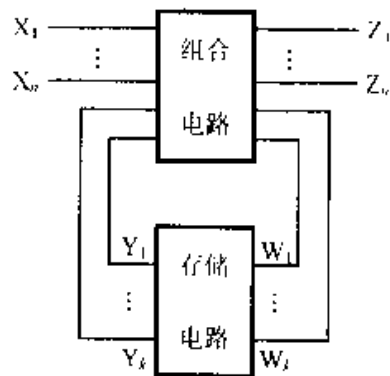


图 5.1.1 时序电路框图

由图 5.1.1 可写出逻辑函数式

$$Z_i = f_i(X_1, \dots, X_n, Y_1, \dots, Y_k) \quad i=1, \dots, m \quad (5.1.1a)$$

$$W_j = g_j(X_1, \dots, X_n, Y_1, \dots, Y_k) \quad j=1, \dots, k \quad (5.1.1b)$$

称(5.1.1a)式为输出函数；称(5.1.1b)式为控制函数或激励函数。

图 5.1.1 示出了时序电路在结构上的两个特点：其一，在一般情况下，电路包含组合电路和存储电路两部分；其二，组合电路至少有一个输出反馈到存储电路的输入端，而存储电路的输出至少有一个是组合电路的输入，同其他外输入共同决定电路的输出。

存储电路一般由触发器构成，其他记忆元件很少采用。

在时序电路中，“状态”是一个很重要的概念，时序电路“状态”分为内部状态和外部状态，内部状态是指存储电路的输出状态 Y_1, \dots, Y_k ；而外部状态是指时

序电路的外输出 Z_1, \dots, Z_m 。以后所讨论的时序电路状态都是指内部状态即存储电路状态。

时序电路分为同步时序电路和异步时序电路两大类。在同步时序电路中,有一个统一的时钟同步脉冲,只有在时钟脉冲的特定时刻存储电路状态才变化,并且时钟脉冲间隔不能太短,只有在前一时钟脉冲所引起的电路响应完全结束之后,也就是电路进入新的稳定状态之后,下一个时钟脉冲才能到来,否则,电路状态就会发生混乱。在异步时序电路中,或者没有时钟脉冲(例如,由两个与非门构成的基本 RS 触发器),或者虽有时钟脉冲,但是存储电路状态的变化与时钟脉冲是异步的。下面将要讨论的异步计数器就属于后一种情况。

在实际应用的时序电路中,大多数是属于同步时序电路,所以,本章将系统地讨论这类电路的分析方法和设计方法;仅以异步计数器为例来介绍异步时序电路的工作特性。

图 5.1.2(a)和(b)分别是由 JK 触发器和 D 触发器构成存储电路的同步时序电路框图。图(a)所示电路的输出函数和激励函数为

$$Z_i = f_i(X_1, \dots, X_n, Y_1, \dots, Y_k) \quad i=1, \dots, m \quad (5.1.2a)$$

$$\left. \begin{aligned} J_j &= g_j(X_1, \dots, X_n, Y_1, \dots, Y_k) \\ K_j &= g_j(X_1, \dots, X_n, Y_1, \dots, Y_k) \end{aligned} \right\} \quad j=1, \dots, k \quad (5.1.2b)$$

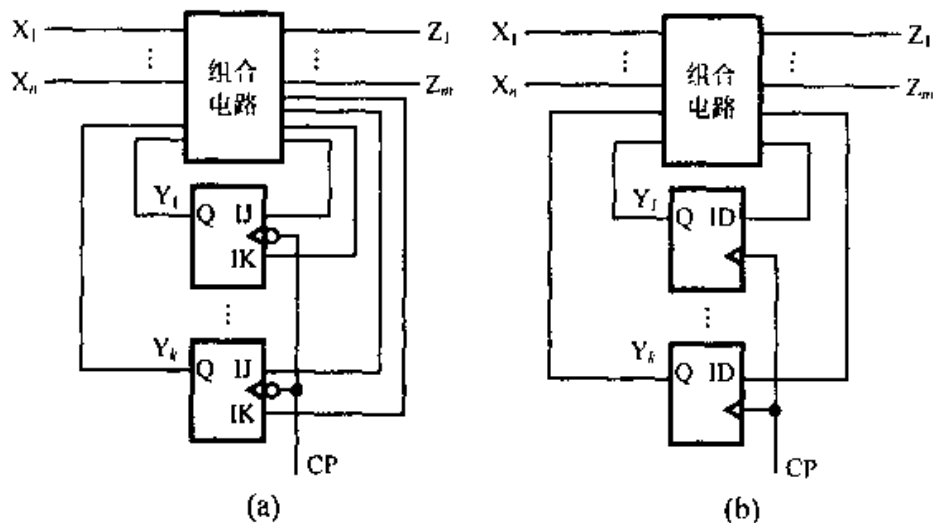


图 5.1.2 同步时序电路框图

(a) 由 JK 触发器构成的存储电路

(b) 由 D 触发器构成的存储电路

同样,对于图(b)所示电路有

$$Z_i = f_i(X_1, \dots, X_n, Y_1, \dots, Y_k) \quad i=1, \dots, m \quad (5.1.3a)$$

$$D_j = g_j(X_1, \dots, X_n, Y_1, \dots, Y_k) \quad j=1, \dots, k \quad (5.1.3b)$$

同步时序电路可分为 mealy 型和 moore 型两类。mealy 型电路的输出状态不仅

与电路状态有关,同时还与外输入有关,其输出函数 Z 可用式(5.1.1a)表示。moore 型电路的输出状态仅与电路状态有关而与外输入无关(或者没有外输入),它的输出函数可用下式表示:

$$Z_i = f_i(Y_1, \dots, Y_k) \quad i=1, \dots, m$$

5.2 同步时序电路的设计

5.2.1 同步时序电路的设计方法

同步时序电路的设计,就是根据给定的逻辑功能,设计其逻辑电路。设计步骤如下。

(a) 拟定原始状态表或状态图:把待设计电路的一般文字描述变成电路输入、输出和状态关系的说明,在此基础上,拟定出原始状态表或状态图。

(b) 状态简化:原始状态表中可能有多余的状态,可用状态简化的方法将其消去,以得到最小化状态表。

(c) 状态分配:根据最小化状态表中的状态数目 r ,确定构成存储电路的触发器数目 k 。 k 个触发器可以具有 2^k 个不同状态,即 2^k 个二进制代码,所以,可以对表中每一个状态指定一个二进制代码表示。

(d) 确定激励函数和输出函数。

(e) 画逻辑电路图。

5.2.2 拟定原始状态表或状态图

根据用文字描述的设计要求构成原始状态表,确定电路有几个状态、状态之间的转换关系和电路输出等几个问题。一般采用直接构成法,即根据设计要求直接画出状态图,再由状态图得到状态表。

必须指出,在构成原始状态表的过程中,注意力不应放在追求最少状态数目上,而应确保状态图的正确性,因为即使有多余状态也无妨,在状态简化步骤中可以消去。

例 5.2.1 拟定 01 序列检测器原始状态表。

解 根据设计要求,电路具有一个输入 X 和一个输出 Z ,输入 X 为一串随机信号,当其中出现 01 序列时,检测器输出为 1,在其他输入序列情况下,检测器输出都为 0。例如,输入 X 为序列 01010011 时,相应输出 Z 序列 01010010。

采用直接构成法,即先根据设计要求直接拟定原始状态图,然后得到相应的状态表。

假定电路收到 1 时处于 A 状态,收到 0 时处于 B 状态。构成原始状态图过程如下:

当电路处于 A 状态时,若输入为 1(0)则电路停留在(转换到)A(B)状态,并且输出为 0;

当电路处于 B 状态时,若输入为 0(1)则电路停留在(转换到)B(A)状态,并且输出为 0(1)。

以上 4 种情况包括了该电路工作的所有情况,所以,电路只包含 A 和 B 两个状态就够了。假如将电路预置在 A 状态,那么,该电路就可以正确地完成 01 序列检测器的逻辑功能。根据以上分析画出状态图,如图 5.2.1 所示,由图可作出状态表(见表 5.2.1)。

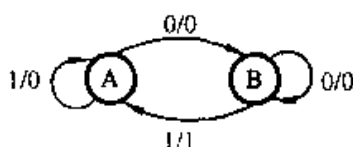


图 5.2.1 01 序列检测器的状态图

表 5.2.1 01 序列检测器的原始状态表

Q	Q^{n+1}/Z	
	X	
	0	1
A	B/0	A/0
B	B/0	A/1

例 5.2.2 拟定下面两个具有一个输入、一个输出的同步时序电路的原始状态图和状态表。

(1) 电路连续不停地工作,凡是遇到连续的 4 位输入为 1001 时,输出便为 1,其他情况下输出为 0。

(2) 每输入 4 位码之后,电路自动恢复到初态,在这 4 位输入码中,当且仅当其 1001 时,输出为 1,否则,输出为 0。

解 (1) 设时序电路输入为 X,输出为 Z,电路收到 0 时为 S_0 ,并定为起始状态。

当电路为 S_0 时,若 $X=0$,则电路停留在 S_0 ,且 $Z=0$;当 X 为 1,则电路转换到 S_1 。

当电路为 S_1 时,若 $X=1$,则认为收到的第二个 1 是待测码组的第一个 1,故电路停留在 S_1 ,且 $Z=0$;若 $X=0$,则电路转换到 S_2 。

当电路为 S_2 时,若 $X=0$,则电路转换到 S_3 ,且 $Z=0$;若 $X=1$,则认为这个 1 是待测码组的第一位 1,故电路转换到 S_1 ,且 $Z=0$ 。

当电路为 S_3 时,若 $X=0$,则可确定不是待测码组,电路回到起始状态 S_0 ,且

$Z=0$; 若 $X=1$, 则电路检测到待测码组, 故 $Z=1$, 并且由于电路是不停地工作, 凡是连续出现 1001 就应输出 1, 所以, 最后收到的 1 又认为是待测码组的第一位 1, 电路应转换到 S_1 。

综合上述情况画出状态图, 如图 5.2.2 所示, 由图作出状态表, 如表 5.2.2 所示。

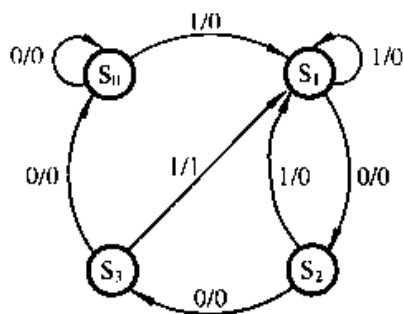


图 5.2.2 例 5.2.2(1)的状态图

表 5.2.2 例 5.2.2(1)状态表

Y	Y_{n+1}/Z	
	X	
	0	1
S_0	$S_0/0$	$S_1/0$
S_1	$S_2/0$	$S_1/0$
S_2	$S_3/0$	$S_1/0$
S_3	$S_0/0$	$S_1/1$

(2) 设电路起始状态为 S_0 , 由于每收到 4 位码后电路恢复到起始状态, 所以, 要假设收到待测码组的 1、2、3 位码的状态分别为 S_1 、 S_2 、 S_3 ; 同时还要设收到不属于待测码组的 1、2、3 位码的状态为 S_4 、 S_5 、 S_6 。

当电路为 S_0 时, 若 $X=1$, 则电路转换到 S_1 , 且 $Z=0$; 若 $X=0$, 则电路转换到 S_4 , 且 $Z=0$ 。

当电路为 S_1 时, 若 $X=0$, 则电路转换到 S_2 , 且 $Z=0$; 若 $X=1$, 则认为电路已收到不属于待测码组的两位码, 故电路转换到 S_5 , 且 $Z=0$ 。

当电路为 S_2 时, 若 $X=0$, 则电路转换到 S_3 , 且 $Z=0$; 若 $X=1$, 则认为电路已收到不属于待测码组的 3 位码, 故电路转换到 S_6 , 且 $Z=0$ 。

当电路为 S_3 时, 则无论是收到 0 或是 1, 电路已经收到 4 位码了, 故电路应回到起始状态 S_0 。若 $X=0$, 则不属于待测码组, $Z=0$; 若 $X=1$, 则检测到待测码组, $Z=1$ 。

当电路为 S_4 时, 则无论是收到 0 或是 1, 电路都转换到 S_5 , 且 $Z=0$ 。

当电路为 S_5 时, 则无论是收到 0 或是 1, 电路都转换到 S_6 , 且 $Z=0$ 。

当电路为 S_6 时, 则无论是收到 0 或是 1, 电路已收到 4 位码了, 但是不属于待测码组, 所以, 电路回到起始状态, 且 $Z=0$ 。

综合上述分析拟定出状态图, 如图 5.2.3 所示, 由图作出状态表, 如表 5.2.3 所示。

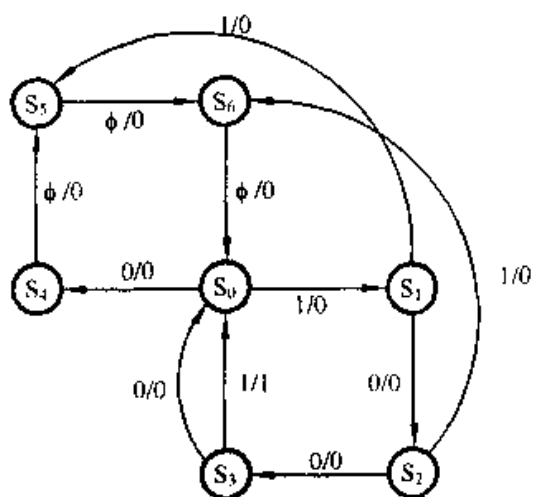


图 5.2.3 例 5.2.2(2)的状态图

表 5.2.3 例 5.2.2(2)状态表

Y	Y_{n+1}/Z	
	X	
	0	1
S_0	$S_4/0$	$S_1/0$
S_1	$S_2/0$	$S_3/0$
S_2	$S_3/0$	$S_0/0$
S_3	$S_0/1$	$S_6/1$
S_4	$S_5/0$	$S_5/0$
S_5	$S_6/0$	$S_6/0$
S_6	$S_6/0$	$S_6/0$

5.2.3 状态简化

1. 等价状态和最小化状态表

如前所述, 原始状态表可能包含有多余状态, 换言之, 其中可能有等价(等效)状态, 因为状态数目的多少直接影响电路的复杂程度, 所以要进行状态简化, 使所设计的状态数目最少, 即求得最小化状态表。在介绍状态简化方法之前, 先来说明等价状态概念。

在原始状态表中, 两个(或两个以上)状态在相同输入条件下, 它们的输出相同, 并且次态等价, 则这两个(或两个以上)状态称为等价状态。如两个状态等价, 对于任意输入序列均产生相同的输出序列。判断次态等价有 3 种情况:

(a) 次态相同就是次态等价。例如, 表 5.2.4 中的状态 0 和 7。

(b) 次态交错也算次态等价。例如, 表 5.2.5 中的状态 B 和 C, 在 $X=1$ 时次态相同, 在 $X=0$ 时次态交错, 所以状态 B 和 C 的状态等价。

(c) 次态循环也算次态等价。这种情况在后面再讨论。

关于等价状态还有几个概念需要说明, 就是等价关系的传递性, 等价类和最大等价类。

等价关系的传递性 假若 A 与 B 等价, 又有 A 与 C 等价, 则 B 与 C 也等价。记为

表 5.2.4 原始状态表

Y	Y_{n+1}/Z	
	X	
	0	1
0	0/0	1/0
1	0/0	2/0
2	0/0	3/0
3	4/0	3/0
4	5/0	1/0
5	0/0	6/0
6	7/1	2/0
7	0/0	1/0

$$(A, B), (A, C) \rightarrow (B, C)$$

所谓等价类是指若干等价状态的集合，显然其中所有状态之间都是等价的。例如(A, B, C)为一等价类，则显然有(A, B), (A, C), (B, C)。反之，若有(A, B), (A, C), (B, C)，则有(A, B, C)。记为

$$(A, B), (A, C), (B, C) \rightarrow (A, B, C)$$

所谓最大等价类是指包含了原始状态表中全部等价状态的等价类。但需指出，原始状态表中不与其他状态等价的单个状态也是一个最大等价类。

例 5.2.3 简化表 5.2.5 所示的原始状态表。

解 由表可以看出，状态 B 和 C 在 X 为 0 和 1 时，电路的输出都相同，并且次态等价，所以 B 和 C 等价(或称等价对)；状态 D 和 E 也是等价状态。因此，得到等价类(B, C) 和(D, E)。

在原始状态表中，既然状态 B 和 C，状态 D 和 E 分别是等价类，就可分别消去其中一个多余状态，例如，消去 C 和 E，并且把表中的 C 和 E 分别用 B 和 D 代替，得到最小化状态表，如表 5.2.6 所示。或者可以对简化后的状态重新命名，例如：A→A'，(B, C)→B'，(D, E)→C'，则可得到最小化状态表如表 5.2.7 所示。

表 5.2.5 例 5.2.3 原始状态表

Y	Y _{n+1} /Z	
	X	
	0	1
A	C/1	B/0
B	C/1	E/0
C	B/1	E/0
D	D/1	B/1
E	D/1	B/1

表 5.2.6 例 5.2.4 最小化状态表之一

Y	Y _{n+1} /Z	
	X	
	0	1
A	B/1	B/0
B	B/1	D/0
D	D/1	B/1

表 5.2.7 例 5.2.4 最小化状态表之二

Y	Y _{n+1} /Z	
	X	
	0	1
A'	B'/1	B'/0
B'	B'/1	C'/0
C'	C'/1	B'/1

2. 隐含表法

用直观比较法进行状态简化只适用于比较简单的原始状态表，对于较复杂的原始状态表就不适用了，这时一般采用系统比较方法即隐含表法。下面通过两例来介绍隐含表法及其应用。

例 5.2.4 简化表 5.2.8 所示的原始状态表。

解 隐含表法一般按下列步骤进行。

(1) 画隐含表表格：隐含表表格是一个直角三角形网格，两直角边网格数相等，且等于原始状态表中状态数目减 1，本例状态数目为 5，故两直角边网格数各为 4；

表 5.2.8 例 5.2.4 原始状态表

Y	Y _{n+1} /Z	
	X	
	0	1
A	B/0	B/1
B	D/0	A/1
C	D/1	B/0
D	B/0	A/1
E	C/1	B/0

表中, 状态次序为 A, B, C, D, E, 隐含表网格纵向从上至下标注时按状态顺序“缺头”, 横向从左至右标注时“少尾”, 见表 5.2.9(a)。

(2) 顺序比较: 将隐含表中所有状态逐一进行比较, 例如, A 逐一与 B、C、D、E 比较, B 逐一与 C、D、E 比较等等。并将比较结果采用 3 种方法标明在相应方格上。

(a) 输出不完全相同的状态对, 标注“×”以示不等价, 例如, A 与 C, B 与 E 等。

(b) 输出完全相同, 次态相同或交错者, 标注“√”以示等价, 称为等价对, 例如, B 与 D。

(c) 输出完全相同, 次态不相同又不交错者, 标注次态, 以便进一步比较。例如, A 与 B 对应方格中标注 BD, 比较结果见表 5.2.9(b)。

(3) 关联比较: 检查隐含表中所填状态对, 观察这些状态对是否等价。有时要进行多次跟踪追寻才可确定一对状态是否等价。例如, 表 5.2.9(b)中, 欲使 AB 等价则需 BD 等价, 而 BD 是等价的, 故 AB 等价, 在填写 BD 的小方格上重新打上“√”记号; 同理, AB 等价使 AD 也等价, 在填写 AB 的小方格上也打上“√”记号。又例如, 要使 CE 等价则要求 CD 等价, 但已知 CD 不等价, 故确定 CE 也不等价, 在填写 CD 的小方格上打上“×”记号。关联比较的结果见表 5.2.9(c)。

表 5.2.9(a) 隐含表

B				
C				
D				
E				
	A	B	C	D

表 5.2.9(b) 顺序比较表

B	BD			
C	×	×		
D	AB	√	×	
E	×	×	CD	×
	A	B	C	D

表 5.2.9(c) 关联比较表

B	BD√			
C	×	×		
D	AB√	√	×	
E	×	×	CD×	×
	A	B	C	D

(4) 列最大等价类: 在关联比较之后可以确定哪些状态是“等价对”, 再由“等价对”构成“等价类”。由表 5.2.9(c)可确定“等价对”为(A、B), (B、D), (A、D), 所以“等价类”为(A、B、D), 最后列出最大等价类。如前所述, 原始状态表中不与其他状态等价的一个单独状态也算一个最大等价类, 所以本例原始状态表中的最大等价类共有(A、B、D), (C), (E)3 个。

(5) 求最小化状态表: 每个最大等价类保留一个状态, 就可构成最小化状态表。本例保留 A, C, E, 则可得最小化状态表, 如表 5.2.10(a)所示。

假若对本例 3 个最大等价类重新命名, 即(A、B、D)→A', (C)→B', (E)→C', 则可得最小化状态表, 如表 5.2.10(b)所示。

例 5.2.5 简化表 5.2.11 所示的原始状态表。

解 (1) 画隐含表表格: 原始状态表中有 8 个状态, 所以纵、横向都是 7 格。

(2) 顺序比较: 将隐含表中所有状态逐一比较, 比较结果见表 5.2.12(a)。

(3) 关联比较: 先来看 GH 是否等价, 欲使 GH 等价则要 CH 和 BC 等价, 但是,

表上已确定 CH、BG 都不等价，故 GH 不等价，在相应方格上打“×”记号。

表 5.2.10(a) 最小化
状态表一

Y	Y_{n+1}/Z	
	X	
	0	1
A	A/0	A/1
C	A/1	A/0
E	C/1	A/0

表 5.2.10(b) 最小化
状态表二

Y	Y_{n+1}/Z	
	X	
	0	1
A'	A'/0	A'/1
B'	A'/1	A'/0
C'	B'/1	A'/0

表 5.2.11 例 5.2.5 原始
状态表

Y	Y_{n+1}/Z	
	X	
	0	1
A	E/0	D/0
B	A/1	F/0
C	C/0	A/1
D	B/0	A/0
E	D/1	C/0
F	C/0	D/1
G	H/1	G/1
H	C/1	B/1

再来看 AD 是否等价，AD 取决于 BE，而 BE 又取决于 AD 和 CF，并且 CF 又取决于 AD，所以，这些状态构成循环，如图 5.2.4 所示。由于这些状态对在输入相同的条件下，输出完全相同；另一组次态是相同或交错。因此，这些状态对都是等价对，比较结果见表 5.2.12(a)。

(4) 列最大等价类：由表 5.2.12(a)可找出等价对为(A、D)，(B、E)，(C、F)，这也就是等价类，所以表 5.2.11 的最大等价类为(A、D)，(B、E)，(C、F)，(G)，(H)。

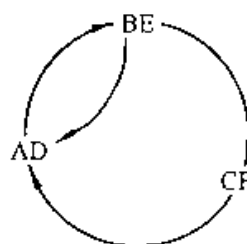


图 5.2.4 状态循环示意图

表 5.2.12(a) 隐含表

B	×						
C	×	×					
D	BE✓	×	×				
E	×	AD CF✓	×	×			
F	×	×	AD ✓	×	×		
G	×	×	×	×	×	×	
H	×	×	×	×	×	×	CH BG×
	A	B	C	D	E	F	G

表 5.2.12(b) 最小化状态表

Y	Y_{n+1}/Z	
	X	
	0	1
A'	B'/0	A'/0
B'	A'/1	C'/0
C'	C'/0	A'/1
D'	E'/1	D'/1
E'	C'/1	B'/1

(5) 列最小化状态表：重新对最大等价类命名，即(A、D)→A'，(B、E)→B'，(C、F)→C'，(G)→D'，(H)→E'，可列出只有 5 个状态的最小化状态表，如表 5.2.12(b)所示。

5.2.4 状态分配

原始状态表经过状态简化之后得到最小化状态表,其中,每一个状态通常用英文字母(如 A、B、C 等)或十进制数字(如 0、1、2 等)表示。状态分配就是把状态表中每个用英文字母或十进制数字表示的状态赋予一组二进制码。用二进制码表示状态所构成的状态表(图)称为二进制状态表(图)。

假如最小化状态表中状态数目为 r ,则电路需要用 k 个状态变量(即 k 个触发器),并且满足关系 $2^k \geq r > 2^{k-1}$ 。例如,状态数目 $r=5$,则最少需用 3 个触发器,即 $k=3$,也就是 3 位二进制码进行状态分配(状态编码)。状态分配方案数 N 可用下式表示:

$$N = \frac{2^k!}{(2^k - r)!} \quad (5.2.1)$$

例如, $r=5, k=3$,即用 3 位二进制码的 8 种不同组合表示 5 种状态的可能分配方案共有

$$N = \frac{2^3!}{(2^3 - 5)!} = 6720 \text{ (种)}$$

其中,有一些方案对电路设计的结果是等效的,可以合并为一组,则可分为

$$N_{\text{组}} = \frac{(2^k - 1)!}{(2^k - r)!k!} = \frac{(2^3 - 1)!}{(2^3 - 5)!3!} = 140^{\text{①}} \quad (5.2.2)$$

状态分配方案不同,其电路结构的优劣会有一定的差异,然而,要对 140 组分配方案作出评价与选择是很困难的。在实际工程设计中,一般是采用相邻状态分配法^[3,5]以寻求比较理想的状态分配方案,但其结果并非最佳(或次佳)方案。

状态分配应遵循哪些规则才能简化电路,各国学者对此进行了长期的研究,并有了初步的结果。但是,由于这个问题的解决难度很大,因此,至今仍然没有一个完善的、能为设计者普遍接受的好方法。基于上述情况,特别是考虑到目前集成电路价格已降得较低的事实,我们在状态分配时只进行状态的二进制码的分配,而不去考虑它们对简化电路的优劣的影响。

5.2.5 确定激励函数和输出函数

确定了状态分配方案并列出二进制状态表之后,就可以选择常用的各类触发器来设计电路。目前,触发器的主要产品是 D 触发器和 JK 触发器,可以从它们

^①此式推导见第一版 § 8.3。

之中选择一种来设计电路, 求出它们的激励函数和输出函数, 便可画出逻辑电路图。

例 5.2.6 最小化二进制状态表如表 5.2.13 所示, 试设计该同步时序电路。

解 本例采用 D 和 JK 两种触发器设计出两种电路方案, 以便读者比较。

由表 5.2.13 可列出状态转换真值表和激励表, 如表 5.2.14 所示。表中, 包含有两种触发器的激励表。

表 5.2.13 例 5.2.6 最小化状态表

Q_2Q_1	$Q_2^{n+1}Q_1^{n+1}/Z$	
	X	
	0	1
00	00/0	10/0
01	00/0	01/1
10	00/0	01/0

表 5.2.14 状态转换真值表和激励表

X	Q_2	Q_1	Q_2^{n+1}	Q_1^{n+1}	Z	D_2	D_1	J_2	K_2	J_1	K_1
0	0	0	0	0	0	0	0	0	ϕ	0	ϕ
0	0	1	0	0	0	0	0	0	ϕ	ϕ	1
0	1	0	0	0	0	0	0	ϕ	1	0	ϕ
0	1	1	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
1	0	0	1	0	0	1	0	1	ϕ	0	ϕ
1	0	1	0	1	1	0	1	0	ϕ	ϕ	0
1	1	0	0	1	0	0	1	ϕ	1	1	ϕ
1	1	1	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ

由状态转换真值表, 可分别画出输出函数 Z 以及触发器的控制输入函数 D_2 、 D_1 、 J_2 、 J_1 、 K_2 和 K_1 的卡诺图, 如图 5.2.5 所示。

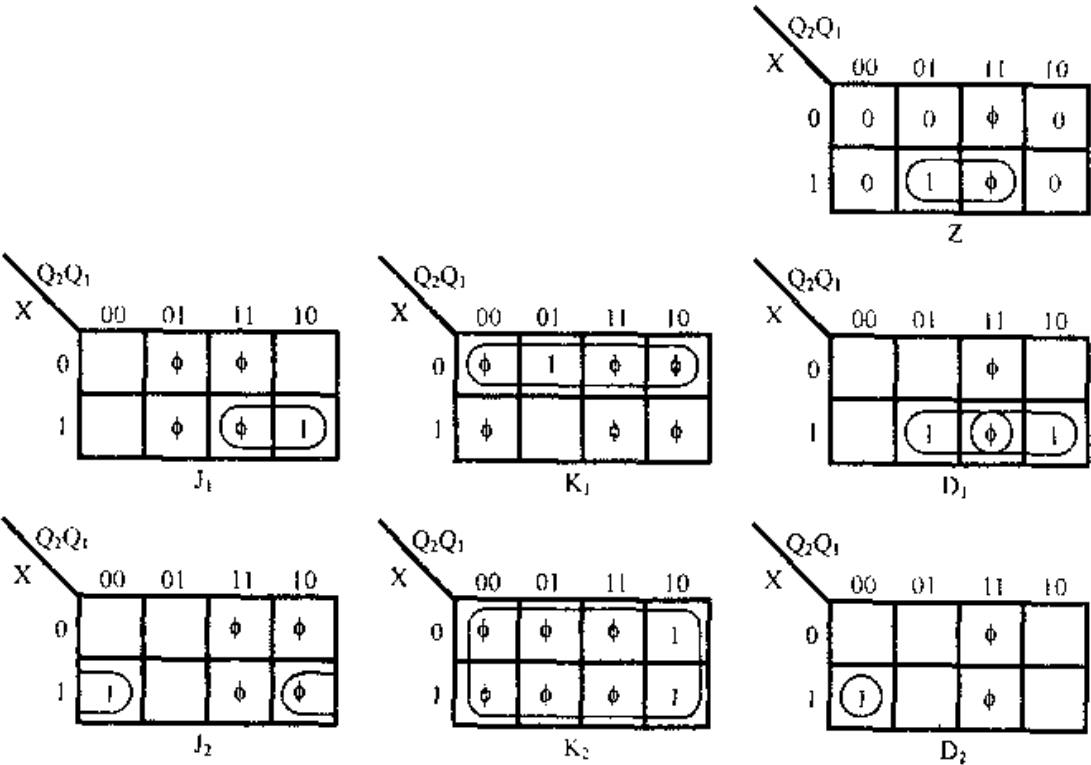


图 5.2.5 例 5.2.6 的卡诺图

由此写出其最简表达式

$$Z = XQ_1 \quad (5.2.3)$$

$$\left. \begin{aligned} D_2 &= X\bar{Q}_2\bar{Q}_1 \\ D_1 &= XQ_2 + XQ_1 \end{aligned} \right\} \quad (5.2.4)$$

$$\left. \begin{aligned} J_2 &= X\bar{Q}_1, & K_2 &= 1 \\ J_1 &= XQ_2, & K_1 &= X \end{aligned} \right\} \quad (5.2.5)$$

由式(5.2.3)和式(5.2.5)可画出由 JK 触发器构成的逻辑电路图(请读者画出由 D 触发器构成的逻辑电路图), 如图 5.2.6 所示。

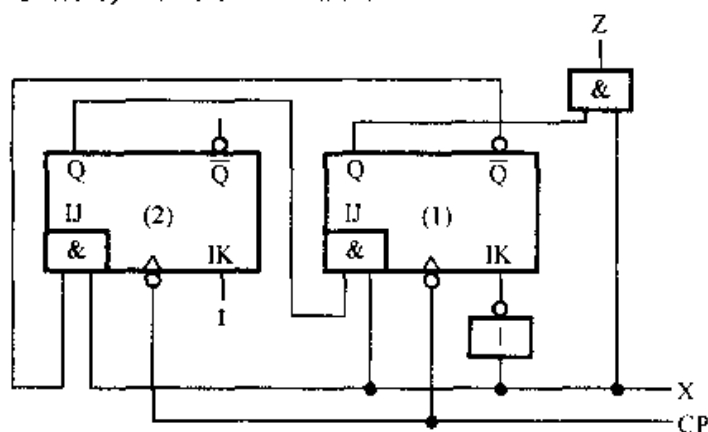


图 5.2.6 例 5.2.6 的逻辑电路图

5.2.6 设计举例

下面举例说明同步时序电路设计的全过程。

例 5.2.7 设计一个 111 序列信号检测器。

解 该检测器的功能是, 仅当输入信号连接 3 个 1 或 3 个以上 1 时, 输出才为 1, 而在其他输入序列情况下检测器输出为 0。

(1) 拟定原始状态表: 该电路有一个输入端 X, 输入为一串随机信号, 有一个输出端 Z。

该电路需记忆的信息有: 已收到 0 的输入, 用 S_0 状态表示; 已收到一个 1 的输入, 用 S_1 状态表示; 已连续收到两个 1 的输入, 用 S_2 状态表示; 已连续收到 3 个 1 的输入, 用 S_3 状态表示。设 S_0 为电路的起始状态。

在 S_0 状态时, 若 $X=1$, 由 S_0 状态转换至 S_1 状态, 且 $Z=0$ 。在 S_1 状态时, 若 $X=1$, 由 S_1 状态转换至 S_2 状态, 且 $Z=0$ 。在 S_2 状态时, 若 $X=1$, 由 S_2 状态转换至 S_3 状态, 由于已连续收到 3 个 1, 故 $Z=1$ 。此后, 若输入连续为 1, 则电路停留在 S_3 状态且 Z 总为 1。

在 S_0 状态时, 若 $X=0$, 则电路停留在 S_0 状态, 且 $Z=0$ 。在 S_1 , S_2 和 S_3 状态时,

若 $X=0$, 则它们都转换到 S_0 , 且 $Z=0$ 。

图 5.2.7 是该电路的原始状态图, 由图得状态表, 如表 5.2.15 所示。

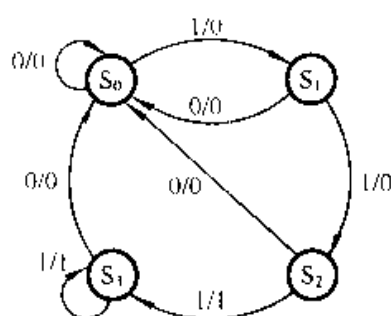


图 5.2.7 111 序列信号检测器的状态图

表 5.2.15 111 序列检测器的状态表

Y	Y^{n+1}/Z	
	X	
	0	1
S_0	$S_0/0$	$S_1/0$
S_1	$S_0/0$	$S_2/0$
S_2	$S_0/0$	$S_3/1$
S_3	$S_0/0$	$S_3/1$

(2) 状态简化: 分析表 5.2.15 可以发现, S_2 和 S_3 状态在 $X=0$ 时次态都是 S_0 且输出相同, 在 $X=1$ 时次态都是 S_3 且输出相同, 因此, S_2 和 S_3 状态是等价状态, 可以消去其中的一个状态, 如消去 S_3 状态, 这样, 表 5.2.15 就简化为 3 个状态的最小化状态表, 如表 5.2.16 所示。

(3) 状态分配: 因为电路状态数目 $r=3$, 所以, 触发器数目 $k=2$, 即用两位二进制码对表 5.2.16 中的状态进行分配。指定 $S_0=00, S_1=01, S_2=11$, 则由表 5.2.16 得二进制状态表, 如表 5.2.17 所示。

(4) 确定输出函数及控制输入函数: 选用 D 触发器, 由表 5.2.17 和 D 触发器的激励表, 可列出电路状态转换真值表和激励表, 如表 5.2.18 所示。

表 5.2.16 最小化状态表

Y	Y^{n+1}/Z	
	X	
	0	1
S_0	$S_0/0$	$S_1/0$
S_1	$S_0/0$	$S_2/0$
S_2	$S_0/0$	$S_2/1$

表 5.2.17 二进制状态表

Q_2Q_1	$Q_2^{n+1}Q_1^{n+1}/Z$	
	X	
	0	1
00	00/0	01/0
01	00/0	11/0
11	00/0	11/1

表 5.2.18 状态转换真值表和激励表

X	Q_2	Q_1	Q_2^{n+1}	Q_1^{n+1}	Z	D_2	D_1
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	1	0	ϕ	ϕ	ϕ	ϕ	0
0	1	1	0	0	0	0	0
1	0	0	0	1	0	0	1
1	0	1	1	1	0	1	1
1	1	0	ϕ	ϕ	ϕ	ϕ	0
1	1	1	1	1	1	1	1

由表 5.2.18 可得 Z 、 D_2 和 D_1 的卡诺图, 见图 5.2.8。

由图 5.2.8 可分别写出它们的最简表达式

$$\left. \begin{aligned} D_2 &= XQ_1 \\ D_1 &= X \\ Z &= XQ_2 \end{aligned} \right\} \quad (5.2.6)$$

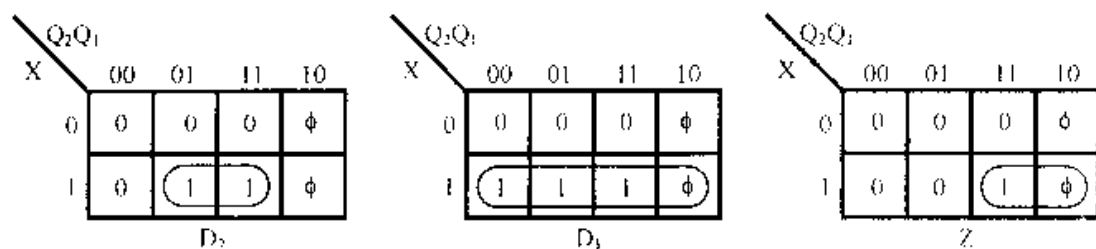


图 5.2.8 111 序列信号检测器卡诺图

由式(5.2.6)可画出该检测器的逻辑电路,如图 5.2.9 所示。

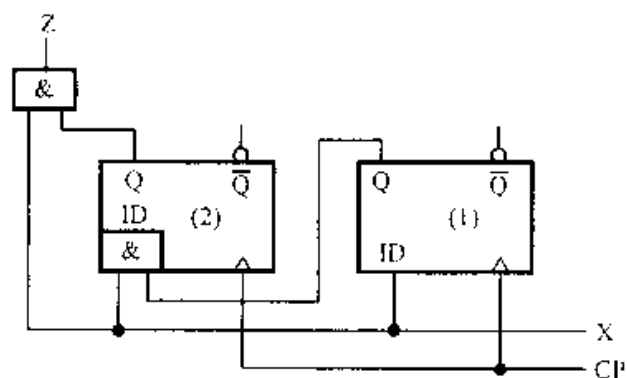


图 5.2.9 111 序列信号检测器的逻辑电路图

顺便指出:序列检测器必须预置在起始状态,以免电路刚投入运行时可能出现的逻辑错误。本例中,应预置在 S_0 状态,而 $S_0=00$,即令 $Q_2=Q_1=0$ 。

5.3 同步时序电路的分析

时序电路的分析就是根据电路图分析它的逻辑功能,以便得到该电路工作特性的详尽说明。其目的在于弄清该电路的用途,同时也可以帮助我们掌握同步时序电路的设计方法。同步时序电路的一般分析步骤为

- 分析电路的组成;
- 确定输出方程和激励方程;
- 写出存储电路的次态方程;
- 列出状态转换真值表和状态表;
- 画出状态图;
- 电路功能描述。

下面举例说明同步时序电路的分析方法。

例 5.3.1 分析图 5.3.1 所示同步时序电路。

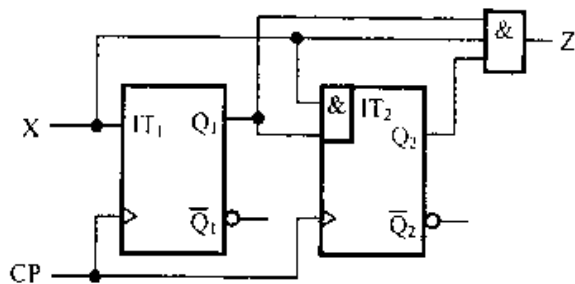


图 5.3.1 例 5.3.1 的逻辑电路图

解 (1) 分析电路的组成: 组合电路是一个与门, 存储电路是两级 T 触发器。

(2) 确定输出方程和激励方程:

$$\left. \begin{aligned} T_1 &= X \\ T_2 &= XQ_1 \end{aligned} \right\} \quad (5.3.1)$$

$$Z = XQ_2Q_1 \quad (5.3.2)$$

式(5.3.2)表明电路的输出与外输入 X 有关, 故属于 Mealy 型电路。

(3) 写出存储电路的次态方程:

$$\left. \begin{aligned} Q_1^{n+1} &= X \oplus Q_1 \\ Q_2^{n+1} &= XQ_1 \oplus Q_2 \end{aligned} \right\} \quad (5.3.3)$$

(4) 列出状态转换真值表和状态表: 该电路有一个外输入 X, 两个状态变量 Q_1 和 Q_2 , 所以, 有 8 种不同的组合, 可以求出每一种组合时的即刻输出 Z 和次态 Q_1^{n+1} , Q_2^{n+1} , 列成状态转换真值表, 见表 5.3.1。表中, Z 由式(5.3.2)确定, Q_1^{n+1} 和 Q_2^{n+1} 由式(5.3.3)确定。为了使表直观和完善, 还增加了 T_1 和 T_2 两列。

由表 5.3.1 可以列出更简单的状态转换表, 如表 5.3.2 所示。

表 5.3.1 例 5.3.1 的状态转换真值表

X	Q_2	Q_1	T_2	T_1	Q_2^{n+1}	Q_1^{n+1}	Z
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	0	0	1	1	0
1	0	0	0	1	0	1	0
1	0	1	1	1	1	0	0
1	1	0	0	1	1	1	0
1	1	1	1	1	0	0	1

表 5.3.2 例 5.3.1 的状态表

Q_2	Q_1	$Q_2^{n+1}Q_1^{n+1}/Z$	
		X	
		0	1
0	0	00/0	01/0
0	1	01/0	10/0
1	0	10/0	11/0
1	1	11/0	00/1

(5) 画状态图: 状态表虽然可以描述电路的状态转换和逻辑功能, 但是不直观, 所以, 通常采用状态图来更直观地描述逻辑功能。假设电路现态 Q_2Q_1 为 00, 01, 10 和 11, 分别用 S_0 , S_1 , S_2 和 S_3 表示, 由表 5.3.2 可以画出状态图 5.3.2。

(6) 电路功能描述: 从状态图可以看出, 当 $X=0$ 时, 状态维持不变; 当 $X=1$ 时, 状态发生转换。假如, X 固定取 1 值, 并预置电路在 S_0 状态, 那么, 随着 CP 的作用, 状态将按照 $S_0 \rightarrow S_1 \rightarrow S_2 \rightarrow S_3$ 循环转换, 并且每 4 个 CP 脉冲作用后输出一个进位脉冲, 故知此电路逻辑功能是可控模 4 加法计数器(参见 5.6 节)。

例 5.3.2 分析图 5.3.3 所示同步时序电路。

解 (1) 分析电路的组成(略)。

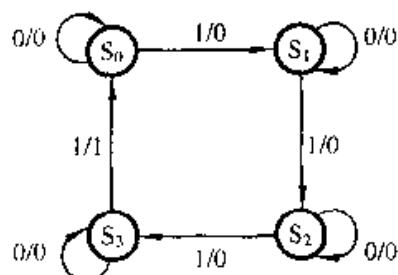


图 5.3.2 例 5.3.1 的状态图

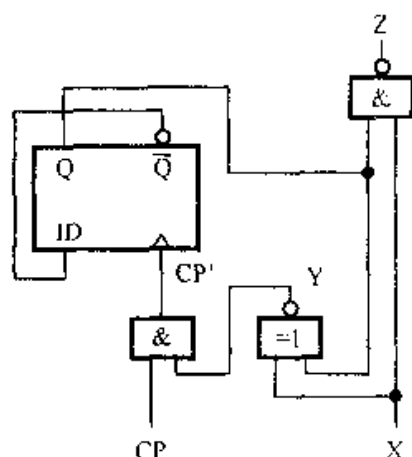


图 5.3.3 例 5.3.2 的逻辑电路图

(2) 确定输出方程和激励方程:

$$D = \bar{Q} \quad \text{和} \quad Z = XQ$$

(3) 写出存储电路的次态方程:

$$Q^{n+1} = \bar{Q} \quad (5.3.4)$$

请注意本例中 D 触发器并不是直接由 CP 信号触发, 而是由 CP' 信号触发, $CP' = CP \cdot Y$, 式中, $Y = X \odot Q$, 所以, 当 $Y=0$ 时, 与门关闭, $CP'=0$, D 触发器得不到触发信号, 状态不变; 当 $Y=1$ 时, 与门开启, $CP'=CP$, 即 CP 信号通过与门加至触发器, 新状态由式 (5.3.4) 确定。所以, 可以写出

当 $Y=0$ 时, $Q^{n+1}=Q$; 当 $Y=1$ 时, $Q^{n+1}=\bar{Q}$ 。

(4) 列出状态转换真值表和状态表: 状态转换真值表如表 5.3.3 所示, 为了使表直观和完善, 表中增加了 Y 列。

由表 5.3.3 可导出状态表 5.3.4。

(5) 画状态图: 由表 5.3.4 画出状态图, 见图 5.3.4。

表 5.3.3 例 5.3.2 的状态
转换真值表

X	Q	Y	Q^{n+1}	Z
0	0	1	1	1
0	1	0	1	1
1	0	0	0	1
1	1	1	0	0

表 5.3.4 例 5.3.2 的
状态表

Q	Q^{n+1}/Z	
	X	
	0	1
0	1/1	0/1
1	1/1	0/0

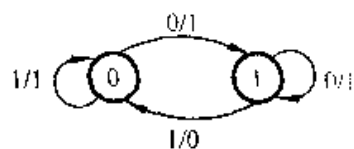


图 5.3.4 例 5.3.2 的状态图

(6) 电路功能描述: 图 5.3.4 表明, 凡 X 为 0, 电路进入 1 状态; 凡 X 为 1, 则电路进入 0 状态。并且只有在由 1 状态转换到 0 状态时电路输出为 0, 其他情况下输出均为 1。假若预置电路初始状态为 0 状态, 当输入序列是先为 0 后为 1 时, 电路将由 0 状态转换到 1 状态, 然后又由 1 状态回到 0 状态, 并且输出为 0。在其他

输入序列情况下输出均为 1。故该电路是 01 序列检测器。

例 5.3.3 分析图 5.3.5 所示同步时序电路。

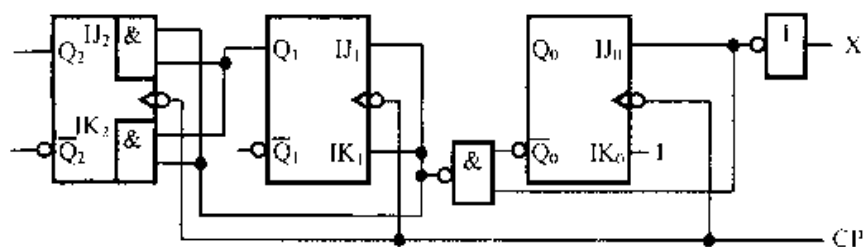


图 5.3.5 例 5.3.3 的逻辑电路图

解 (1) 分析电路的组成(略)。

(2) 写出激励方程:

$$\begin{aligned} J_0 &= \bar{X} & K_0 &= 1 \\ J_1 &= K_1 = \bar{X} \bar{Q}_0 \\ J_2 &= K_2 = \bar{X} \bar{Q}_0 \cdot Q_1 \end{aligned}$$

存储电路的状态就是时序电路的输出。

(3) 写出存储电路的次态方程:

$$\begin{aligned} Q_0^{n+1} &= J_0 \bar{Q}_0 + \bar{K}_0 Q_0 = \bar{X} \bar{Q}_0 \\ Q_1^{n+1} &= \bar{X} \bar{Q}_0 \cdot \bar{Q}_1 + \bar{X} \bar{Q}_0 \cdot Q_1 = \bar{X} \bar{Q}_0 \oplus Q_1 \\ Q_2^{n+1} &= \bar{X} \bar{Q}_0 \cdot Q_1 \oplus Q_2 \end{aligned}$$

(4) 列出状态转换真值表和状态表: 状态转换真值表, 如表 5.3.5 所示, 表中增

表 5.3.5 例 5.3.3 的状态转换真值表

X	Q_2	Q_1	Q_0	$\bar{X} \bar{Q}_0$	Q_2^{n+1}	Q_1^{n+1}	Q_0^{n+1}
0	0	0	0	0	0	0	1
0	0	0	1	1	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	1	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	1	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	1	0
1	0	0	1	1	0	1	0
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	0
1	1	0	0	1	1	1	0
1	1	0	1	1	1	1	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0

加了 $\overline{XQ_0}$ 一列, 以便使表更完善和直观。

由表 5.3.5 可导出状态表, 如表 5.3.6 所示。

(5) 画状态图: 由表 5.3.6 画出状态图, 如图 5.3.6(a)和(b)所示。

(6) 电路功能描述: 由状态图 5.3.6(a)可知, 在 $X=0$ 时, 电路完成二进制计数器($N=8$)功能; 由图(b)可知在 $X=1$ 时, 电路完成模 4 计数器功能, 并且具有自启动功能(参见 5.6 节)。因而电路是一个可控计数器, 分别完成模 8 或模 4 计数功能。

表 5.3.6 例 5.3.3 的状态表

Y	Y^{n+1}	
	X	
	0	1
S_0	S_1	S_2
S_1	S_2	S_2
S_2	S_3	S_4
S_3	S_4	S_4
S_4	S_5	S_6
S_5	S_6	S_0
S_6	S_7	S_0
S_7	S_0	S_0

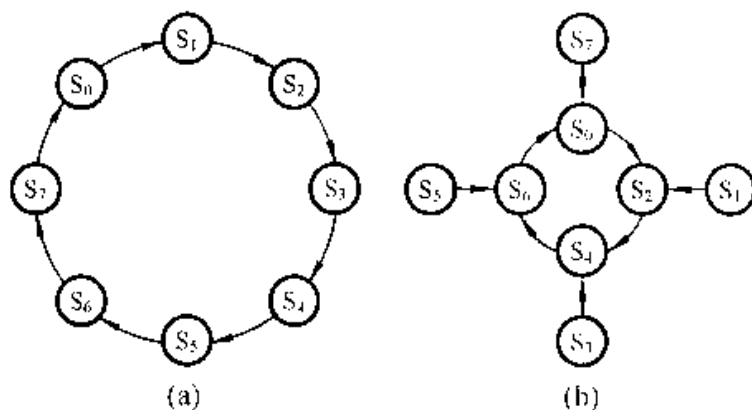


图 5.3.6 例 5.3.3 的状态图

(a) $X=0$ 状态图 (b) $X=1$ 状态图

在系统地介绍了同步时序电路的设计方法和分析方法之后, 则可以运用这些方法来设计和分析一些常用的中规模集成时序逻辑部件(简称时序组件), 如寄存器与锁存器、移位寄存器、计数器、序列发生器等。

5.4 寄存器和锁存器

5.4.1 寄存器

1. 寄存器的设计

在计算机和某些数字电路中, 常常需要将一些数码暂时存放起来, 这种存放数码的逻辑部件称为寄存器。触发器具有两个稳定状态, 可以寄存二进制码, 每一位二进制码需用一个触发器, 各类触发器都可以构成寄存器。寄存器一般应具有以下 4 种逻辑功能。

(a) 清除数码: 为了将寄存器数码清除, 可以将构成寄存器的所有触发器异步置 0 端连接起来, 作为置 0 信号输入端, 加入清 0 负脉冲就能使所有触发器处于 0

状态。

(b) 接收数码：寄存器只有在接收指令(常称写入脉冲，一般为正脉冲)作用下，才能将外输入数码送到各触发器输入端，即寄存器接收数码。

(c) 寄存数码：寄存器只有在寄存指令(一般是触发器的时钟脉冲)作用下，才能将各触发器输入端收到的数码寄存起来。数码寄存之后，只要不出现清 0 或其他指令，寄存数码就不会变，即各级触发器状态不变。

(d) 输出数码：寄存器只有在收到读出指令(一般为正脉冲)之后，才通过读出电路输出数码。

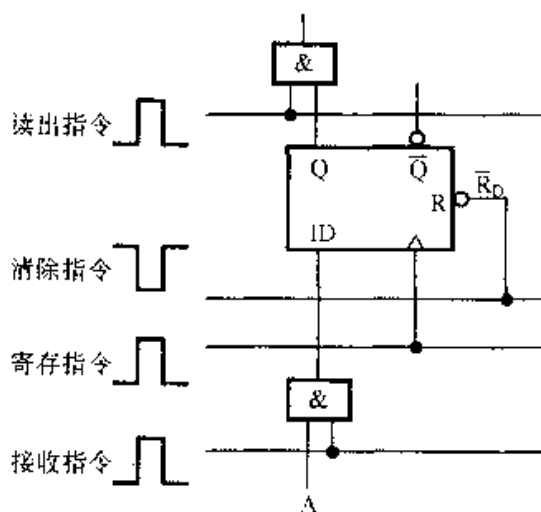


图 5.4.1 D 寄存器中 4 种指令的功能

图 5.4.1 是 D 寄存器中 4 种指令分别实现上述 4 种逻辑功能的示意图。

2. 中规模集成寄存器

中规模集成寄存器有 4 位、6 位、8 位 3 种，一般都具有清除、接收、寄存和输出等 4 种功能。也有一些器件为了适应设计者的实际需要和简化电路，只具有清除或禁止功能。图 5.4.2(a)是 74LS175 型 4 位 D 寄存器的逻辑电路图(其中一位)，该电路仅具有清除端，设置了互补输出端。图 5.4.2(b)是逻辑框图。

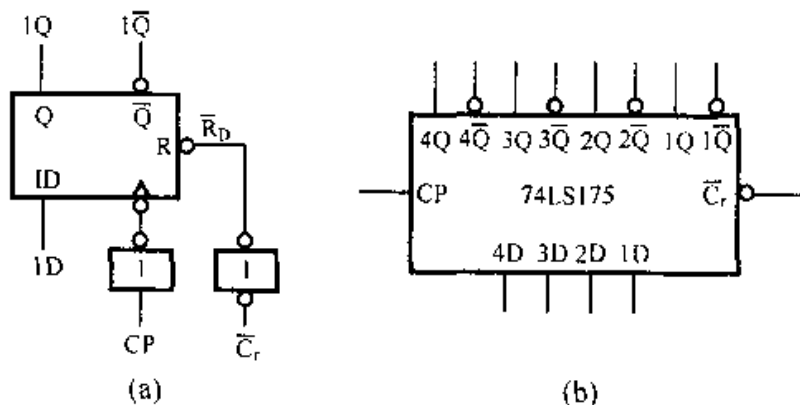


图 5.4.2 74LS175 型 4D 寄存器

(a) 逻辑电路图 (b) 逻辑框图

图 5.4.3(a)和(b)是 74LS377 型 8 位 D 寄存器的逻辑电路图(其中一位)和逻辑框图。该电路具有使能端 \overline{G} ，低电平有效。当 $\overline{G}=1$ 时，寄存器状态不变；当 $\overline{G}=0$ 时，寄存器接收数据(即使能信号相当于接收指令)，在时钟脉冲作用下完成数据寄存功能。

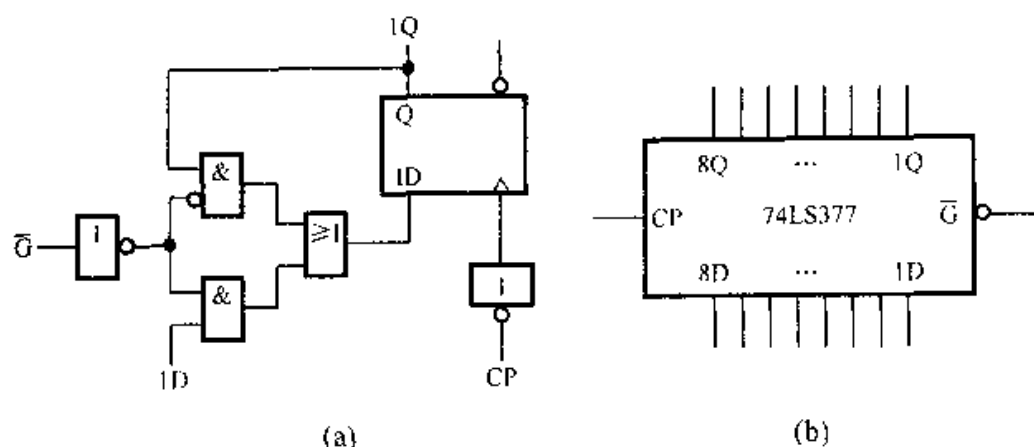


图 5.4.3 74LS377 型 8 位 D 寄存器

(a) 逻辑电路图 (b) 逻辑框图

图 5.4.4(a)和(b)是 CC4076 型 4 位 D 寄存器的逻辑电路图和逻辑框图。它由 4

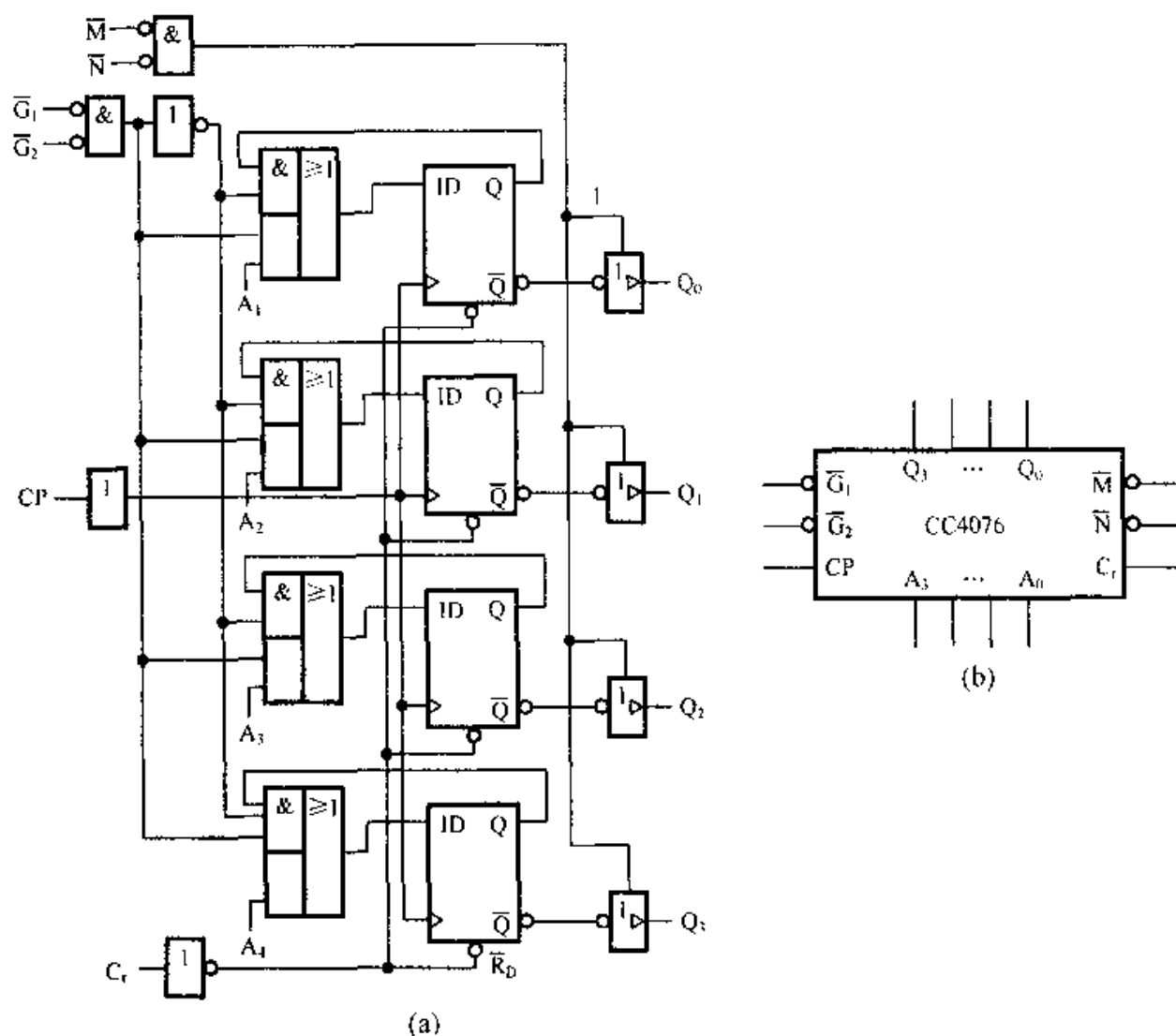


图 5.4.4 CC4076 型 4 位 D 寄存器

(a) 逻辑电路图 (b) 逻辑框图

级 D 触发器和一些门电路构成, 组件设有两个数据输入控制端 \overline{G}_1 和 \overline{G}_2 , 只要其中有一个为高电平, 则禁止数据传送, 只有 \overline{G}_1 和 \overline{G}_2 都为低电平时, 各个数据通道被打开, 各位数据送到相应触发器的 D 端, 在时钟脉冲(寄存指令)作用下完成数据寄存操作。

该组件采用三态门输出结构, 设有两个数据输出控制端 \overline{M} 和 \overline{N} , 它们是与逻辑关系, 即有 $I = \overline{M} \cdot \overline{N}$, 只要其中有一个为高电平, 则 $I=0$, 此时寄存器处于禁止状态; 只有当 \overline{M} 和 \overline{N} 都为低电平时, $I=1$, 此时寄存器数据输出, 功能表如表 5.4.1 所示。为了扩展寄存器的位数, 只需将 m 块组件的时钟端 CP、清除端 C_i 、数据输入控制端 \overline{G}_1 和 \overline{G}_2 、数据输出控制端 \overline{M} 和 \overline{N} 等分别并联起来, 而数据输入和输出端保持独立, 就可以扩展成为 $4m$ 位寄存器。

表 5.4.1 CC4076 功能表

C_i	CP	\overline{G}_1	\overline{G}_2	数据	输出
1	ϕ	ϕ	ϕ	ϕ	0
0	0	ϕ	ϕ	ϕ	Q
0	↑	1	ϕ	ϕ	Q
0	↑	ϕ	1	ϕ	Q
0	↑	0	0	1	1
0	↑	0	0	0	0
0	1	ϕ	ϕ	ϕ	Q
0	·	ϕ	ϕ	ϕ	Q

5.4.2 锁存器

锁存器是具有“透明”特性的一种寄存器, 所谓“透明”特性是指在使能状态时, 输出随输入信号变化(即输出端相当于直接同输入端连接); 当使能信号结束时, 其跳变前那一时刻的输入数据被锁存。

1 位锁存器的电路如图 5.4.5 所示。当使能信号 $\overline{G}=0$ 时^①, 与门 1 打开, 与门 2 关断, 所以 $Q=D$, 即输出 Q 随输入数据变化; 当 $\overline{G}=1$ 时, 与门 1 关断, 与门 2 打开, 所以输出 Q 保持锁存状态不变。

常用的中规模集成锁存器有双 2 位锁存器、双 4 位锁存器、8 位锁存器和 8 位可寻址锁存器等。图 5.4.6 是 8 位锁存器 74LS373 的逻辑框图, G 为使能端, 高电平有效; \overline{OE} 为输出控制端, 低电平有效。若令 $\overline{OE}=0$, 则当 $G=1$ 时, 输出端状态随输入端数据变化(即透明特性); 当 $G=0$ 时, 输出端状态不变。换言之, 在使能信号 G 由 1 跳变到 0 时(使能信号失效那一时刻)完成锁存。

图 5.4.7 是 8 位可寻址锁存器 74LS259 的逻辑框图。 \overline{G} 为使能端, D 为数据输入端, \overline{C}_i 为清除端, C 、 B 、 A 为地址输入端。当 $\overline{G}=0$ 时, 数据输入端同 C 、 B 、 A 地址所确定的输出端保持透明; 当 \overline{G} 由 0 上跳到 1 时刻(使能信号失效), 由 C 、 B 、 A 地址确定将输入数据锁存至相应输出端。其锁存选择表示于表 5.4.2。

① 在锁存器中, 使能信号有效时, 锁存器的输入同输出保持透明特性; 反之, 锁存器状态不变, 所以使能信号失效时刻便实现锁存。

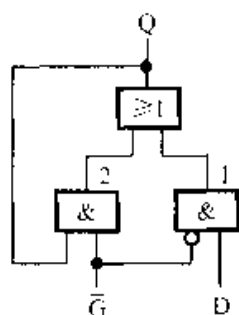


图 5.4.5 1 位锁存器

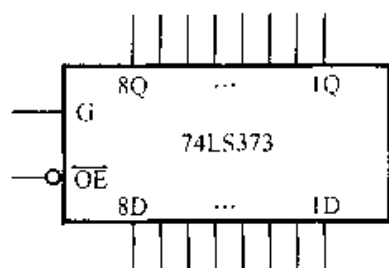


图 5.4.6 8 位锁存器逻辑框图

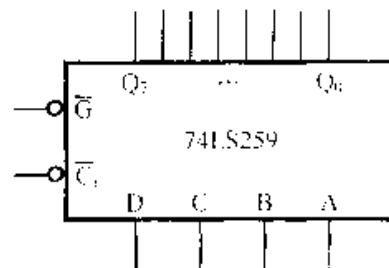


图 5.4.7 8 位可寻址锁存器逻辑框图

表 5.4.2 锁存选择表

C	B	A	被寻址输出端	C	B	A	被寻址输出端
0	0	0	0	1	0	0	4
0	0	1	1	1	0	1	5
0	1	0	2	1	1	0	6
0	1	1	3	1	1	1	7

5.5 移位寄存器

5.5.1 移位寄存器的设计

具有移位逻辑功能的寄存器称为移位寄存器。为了使寄存数码移位，需要加移位指令信号，在它的作用下使数码向左或向右依次顺移一位。

1. 单向移位寄存器

4 级 D 触发器构成的右移寄存器的逻辑电路如图 5.5.1(a)所示。由图可见：移位指令信号就是各级触发器的 CP 信号。要求在 CP 作用下，每次将数码右移一位，由此可以得出各级触发器的连接关系。例如，对 B 级触发器，由 D 触发器特征方程有 $Q_B^{n+1} = D_B$ ，又因为 $Q_B^{n+1} = Q_A$ （此式常称为应用方程），故得 $D_B = Q_A$ 。因此，只要将前一级输出接到下一级 D 输入端就可构成右移寄存器。

用同样方法可以设计出由 JK 触发器构成的右移寄存器。例如，对 B 级触发器，由 JK 触发器特征方程有 $Q_B^{n+1} = J_B \bar{Q}_B + \bar{K}_B Q_B$ ，又有数码右移的应用方程 $Q_B^{n+1} = Q_A = Q_A Q_B + Q_A \bar{Q}_B$ ，故得 $J_B = Q_A$ 和 $K_B = \bar{Q}_A$ 。

上式表明，只要将前一级的 Q 端与 \bar{Q} 端分别与下一级 J 端和 K 端连起来就可构成右移寄存器。图中，第一级(A)触发器输入是这样确定的：为了使数码 D 能在移位指令到来时寄存在 A 触发器中，应令 $Q_A^{n+1} = D$ ；又由 JK 触发器特征方程有 $Q_A^{n+1} = J_A \bar{Q}_A + \bar{K}_A Q_A = D$ 。若令 $J_A = D$ 和 $K_A = \bar{D}$ ，则上式成立，故数码输至 A 触

发器的 J 端, 倒相后输至 K 端, 见图 5.5.1(b)。用类似的方法可以构成左移寄存器。

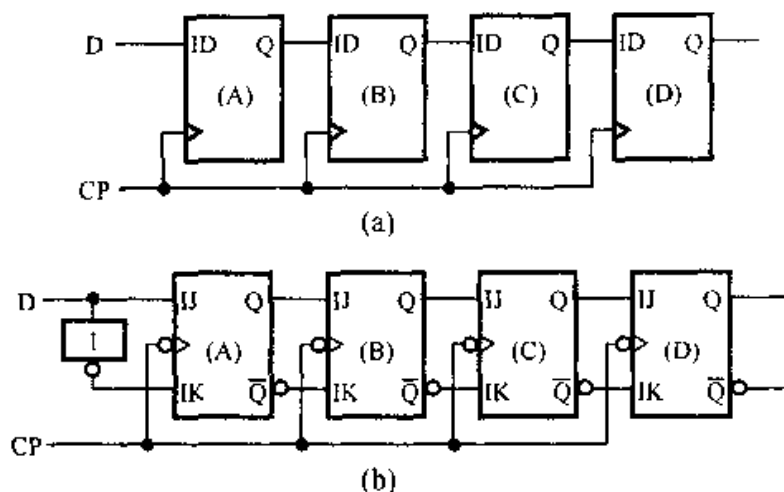


图 5.5.1 单向移位寄存器

(a) 4 位 D 型移位寄存器 (b) 4 位 JK 型移位寄存器

2. 双向移位寄存器

具有既能左移又能右移逻辑功能的寄存器称为双向移位寄存器, 图 5.5.2 是由 D 触发器构成的双向移位寄存器。

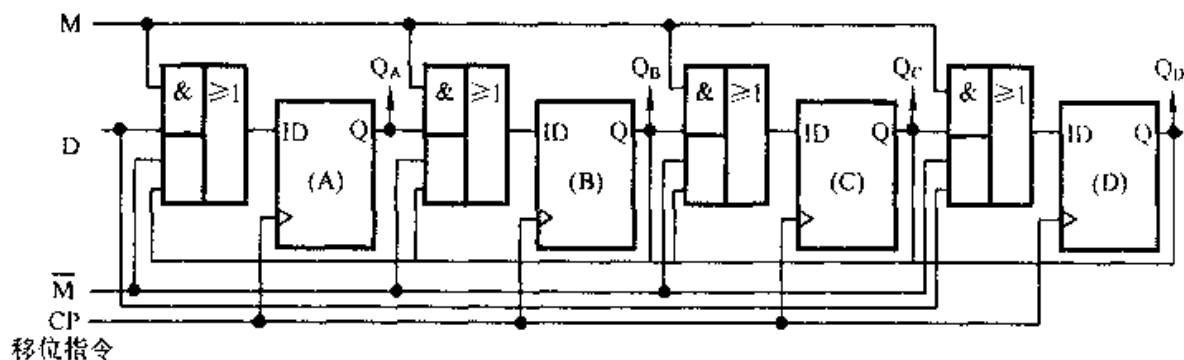


图 5.5.2 双向移位寄存器

为了实现双向移位逻辑功能, 采用移位方向控制指令 M, 当 $M=1$ 时实现右移, 当 $M=0$ 时实现左移, 下面讨论它的设计方法。

对于 B 和 C 触发器的控制输入信号 D_B 和 D_C 的设计方法是相似的。例如, 对于 B 触发器, 其应用方程为 $Q_B^{n+1} = MQ_A + \overline{M}Q_C$, 其特征方程为 $Q_B^{n+1} = D_B$, 由此两式得到

$$D_B = MQ_A + \overline{M}Q_C \quad (5.5.1a)$$

同理得到 C 触发器的控制输入信号为

$$D_C = MQ_B + \overline{M}Q_D \quad (5.5.1b)$$

对于 A 触发器, 其应用方程为 $Q_A^{n+1} = MD + \overline{M}Q_B$, 其特征方程为 $Q_A^{n+1} = D_A$, 由此两式得到

$$D_A = MD + \overline{M} Q_B \quad (5.5.1c)$$

对于 D 触发器，其应用方程为 $Q_D^{n+1} = MQ_C + \overline{M}D$ ，故得

$$D_D = MQ_C + \overline{M}D \quad (5.5.1d)$$

由式(5.5.1a)~式(5.5.1d)画出逻辑电路图，如图 5.5.2 所示。

上述设计方法同样适用于其他类型触发器构成的双向移位寄存器。

5.5.2 中规模集成移位寄存器

移位寄存器依据数据写入和读出方式分为 4 种工作模式：串入-串出；串入-并出；并入-串出；并入-并出。一般通用性较强的中规模集成移位寄存器都具有这 4 种工作模式。

1. 单向移位寄存器

图 5.5.3(a)是 4 位移位寄存器 74LS195 的逻辑电路图，图(b)是逻辑框图。该组

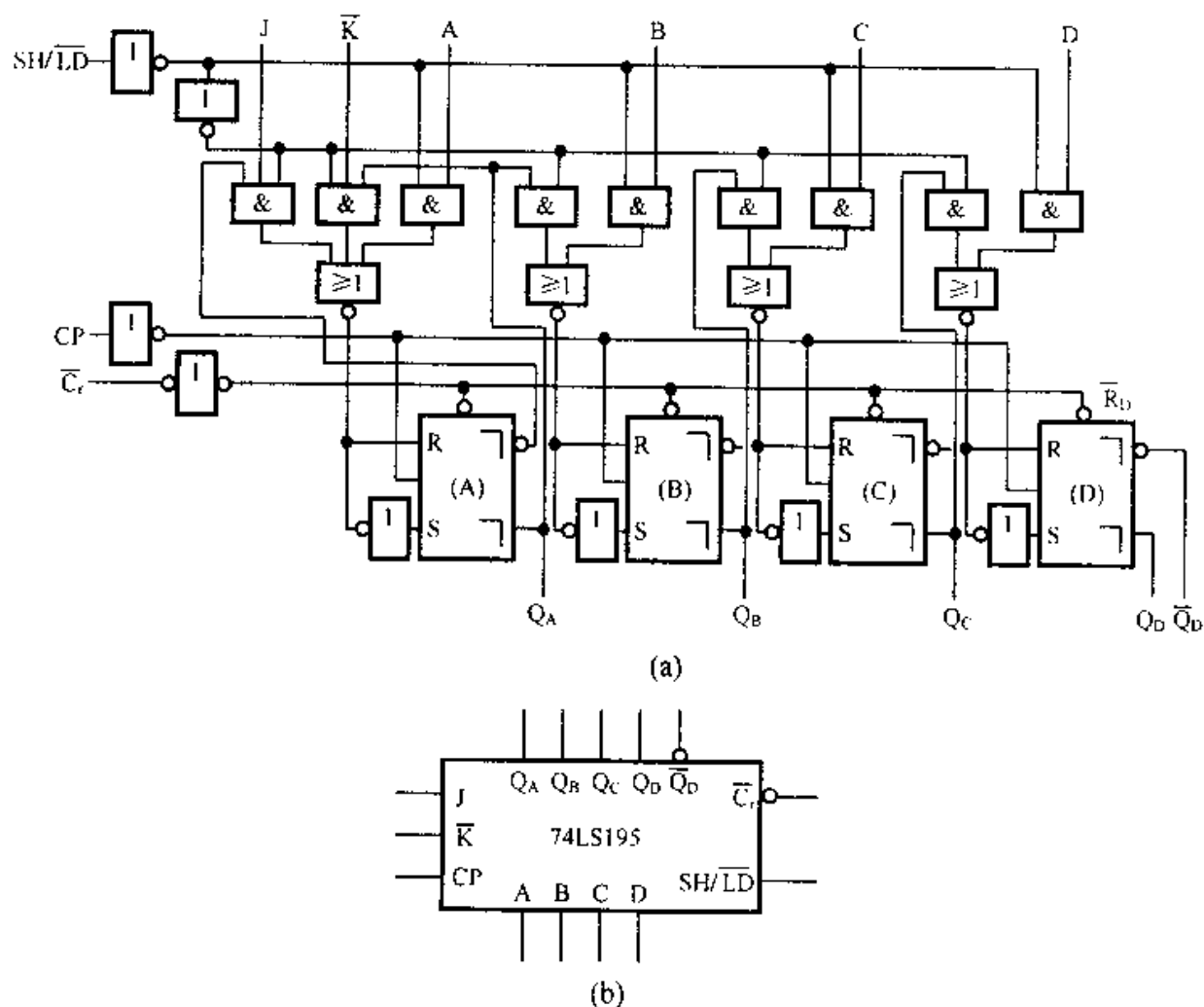


图 5.5.3 74LS195 型 4 位移位寄存器

(a) 逻辑电路图 (b) 逻辑框图

件具有串入和并入的功能,同时也具有串出和并出功能,是一种通用性较强的单向移位寄存器。

该组件设置有清除端 $\overline{C_r}$, 低电平有效; J 、 \overline{K} 是串行数据输入端; SH/\overline{LD} 是移位/置数端, 当 $SH/\overline{LD}=0$ 时完成并行输入数据, 当 $SH/\overline{LD}=1$ 时实现移位功能。在串行输入条件下($SH/\overline{LD}=1$), 输入值 J 、 \overline{K} 的取值不同, 移位寄存器的最低位(Q_A)寄存的数据不同。例如, $J\overline{K}=01$, 则 $Q_A^{n+1}=Q_A$; $J\overline{K}=10$, 则 $Q_A^{n+1}=\overline{Q}_A$ 等, 如功能表 5.5.1 所示。若只要求实现数据右移, 只需将 J 和 \overline{K} 端并接使用便可。

当 $SH/\overline{LD}=0$ 时, 各级 RS 触发器的输入状态为

$$\begin{cases} S_A = A \\ R_A = \overline{A} \end{cases} \quad \begin{cases} S_B = B \\ R_B = \overline{B} \end{cases} \quad \begin{cases} S_C = C \\ R_C = \overline{C} \end{cases} \quad \begin{cases} S_D = D \\ R_D = \overline{D} \end{cases}$$

所以在 CP 脉冲作用后实现同步置数功能。

当 $SH/\overline{LD}=1$ 时, 各级 RS 触发器的输入状态为

$$\begin{cases} S_A = J\overline{Q}_A + \overline{K}Q_A \\ R_A = \overline{S}_A \end{cases} \quad \begin{cases} S_B = Q_A \\ R_B = \overline{Q}_A \end{cases} \quad \begin{cases} S_C = Q_B \\ R_C = \overline{Q}_B \end{cases} \quad \begin{cases} S_D = Q_C \\ R_D = \overline{Q}_C \end{cases}$$

所以在 CP 脉冲作用后实现右移功能, 即第一级(最低位)接收新数据, 其余各级的次态是低一级的原状态。

表 5.5.1(a) 74LS195 功能表之一

$\overline{C_r}$	SH/\overline{LD}	CP	功能
0	ϕ	ϕ	置 0
1	0	\uparrow	并行输入
1	1	\uparrow	右移

表 5.5.1(b) 74LS195 功能表之二

J	\overline{K}	Q_A^{n+1}
0	0	0
0	1	Q_A
1	0	\overline{Q}_A
1	1	1

2. 双向移位寄存器

图 5.5.4(a)是 74LS198 型 8 位双向移位寄存器的逻辑电路图, 图(b)为逻辑框图。

该组件由 8 级负边沿型 D 触发器和一些门控电路构成。组件设置有清除端 $\overline{C_r}$, 低电平有效。 D_{SR} 为右移串行输入端, D_{SL} 为左移串行输入端, 模式控制输入端 M_A 和 M_B 。 M_A 和 M_B 有 4 种不同组合分别实现 4 种工作模式。

(a) 当 $M_A M_B=00$ 时, 由于时钟通路被封锁, 所有触发器均得不到时钟信号, 故移位寄存器状态不变, 即保持操作。

(b) 当 $M_A M_B=10$ 时, 实现右移操作。为了说明工作过程, 可以写出任意一级触发器的激励方程。例如, 对于最低位触发器可写出

$$D_0 = \overline{M}_B D_{SR} + \overline{\overline{M}_A + \overline{M}_B} \cdot d_0 + \overline{M}_A Q_1 \quad (5.5.2)$$

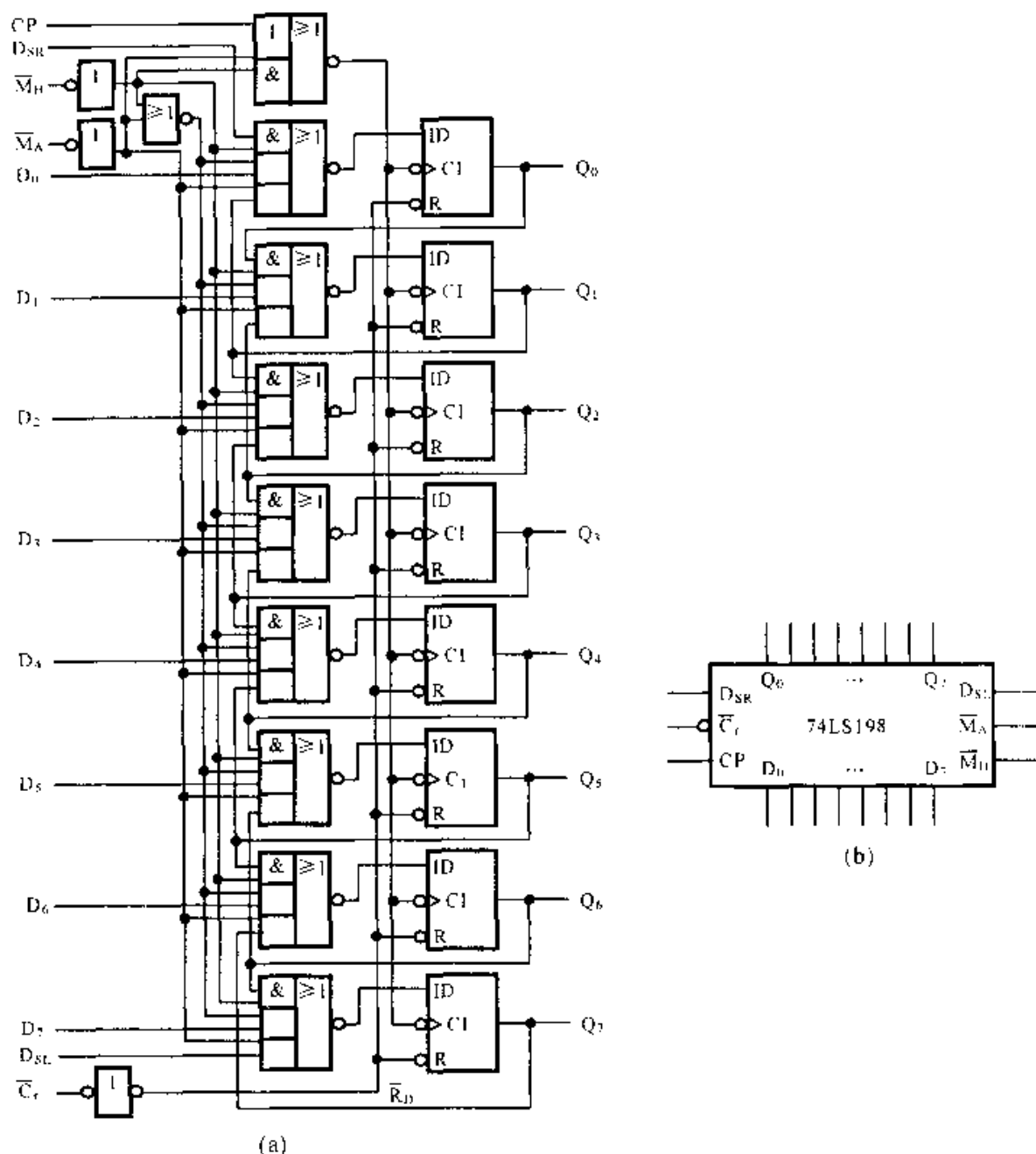


图 5.5.4 74LS198 型 8 位双向移位寄存器

(a) 逻辑电路图 (b) 逻辑框图

因为 $M_A M_B = 10$, 所以, 式(5.5.2)简化为 $D_0 = D_{SR}$, 则有

$$Q_0^{n+1} = D_0 = D_{SR}$$

(c) 当 $M_A M_B = 01$ 时, 实现左移操作。此时, 式(5.5.2)简化为 $D_0 = Q_1$, 所以,

$$Q_0^{n+1} = D_0 = Q_1$$

(d) 当 $M_A M_B = 11$ 时, 实现并行置数操作。此时, 式(5.5.2)简化为 $D_0 = d_0$, 所以

$$Q_0^{n+1} = d_0$$

由于该组件具有清除、保持、置数、右移和左移等功能, 还可以实现串入-串出、串入-并出, 并入-串出和并入-并出等 4 种不同的输入和输出方式。所以它是一种通

用性很强的多功能时序组件，功能表示于表 5.5.2。

表 5.5.2 74LS198 功能表

功能	输 入							输 出	
	\overline{C}_1	CP	M_A	M_B	D_{SR}	D_{SL}	$d_0 \cdots d_7$	$Q_0 \cdots Q_7$	Q_7
清除	0	ϕ	ϕ	ϕ	ϕ	ϕ	0	0 \cdots 0	0
保持	1	0	ϕ	ϕ	ϕ	ϕ	ϕ	$Q_0 \cdots Q_7$	Q_7
	1	0	0	0	0	ϕ	ϕ	$Q_0 \cdots Q_7$	Q_7
置数	1	\uparrow	1	1	ϕ	0	$d_0 \cdots d_7$	$d_0 \cdots d_7$	d_7
右移	1	1	1	0	1	ϕ	ϕ	$1Q_{7n} \cdots Q_{6n}$	Q_{7n}
	1	1	1	0	0	0	0	$0Q_{7n} \cdots Q_{6n}$	Q_{7n}
左移	1	\uparrow	0	1	0	1	0	$Q_{6n} \cdots Q_{1n}$	Q_{1n}
	1	1	0	1	0	0	ϕ	$Q_{6n} \cdots Q_{1n}$	Q_{1n}

3. 可变长度移位寄存器

可变长度移位寄存器的优点是在一定的移位范围内，可以任意选择移位长度，即移位寄存器的位数。图 5.5.5 是美国产品 MC14557B 1~64 位可变长度移位寄存器的逻辑框图，功能表示于表 5.5.3。

该组件是 CMOS 器件。该组件设置了两个数据输入端 A 和 B，A/B 是输入通道选择端。当 A/B=0 时，数据从 B 通道输入；当 A/B=1 时，数据从 A 通道输入。 \overline{C}_1 为清除端，高电平有效； \overline{CE} 为使能端，低电平有效； L_1 、 L_2 、 L_4 、 L_8 、 L_{16} 和 L_{32} 是预置数据输入端，预置数据确定了移位长度，如表 5.5.4 所示。

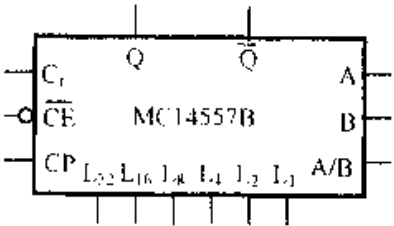


图 5.5.5 可变长度移位寄存器逻辑框图

表 5.5.3 MC14557B 功能表

C_1	\overline{CE}	A/B	CP	Q
1	ϕ	ϕ	ϕ	0
0	1	ϕ	ϕ	不变
0	0	0	\uparrow	B
0	0	1	\uparrow	A

表 5.5.4 移位长度真值表

L_{32}	L_{16}	L_8	L_4	L_2	L_1	移位长度
0	0	0	0	0	0	1 位
0	0	0	0	0	1	2 位
0	0	0	0	1	0	3 位
0	0	0	0	1	1	4 位
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
1	0	0	0	0	0	33 位
1	0	0	0	0	1	34 位
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
1	1	1	1	1	0	63 位
1	1	1	1	1	1	64 位

*5.5.3 移位寄存器的应用

1. 实现数码串-并和并-串转换

在数字系统中，线路上信息的传送通常是串行传送，而信息处理系统一般是并行数据。所以，这二者之间需要进行串-并变换，或者并-串变换，移位寄存器（简称移存器）可以方便地实现这种功能^[27, 28]。

在目前生产移存器的产品中，有一些具有专用功能，如 74LS164 是具有串入-并出功能的 8 位移存器；74LS166 是具有并入-串出（互补输出）功能的 8 位移存器；74LS166 则是具有串/并输入-串出功能的 8 位移存器。这些产品电路结构简化了，成本较低、使用方便；但其通用性降低了。依据逻辑电路要完成的功能，恰当选用这些芯片可以获得良好效果。

实现 8 位并-串转换可以选择 74LS166 8 位移存器，逻辑框图示于图 5.5.6(a)。其中， D_{SR} 为右移输入端， Q_7 为串行输出端， \overline{C}_r 为异步清除端， SH/\overline{LD} 为移位/置数端， $D_0 \sim D_7$ 是并行数据输入端， CP_A 和 CP_B 是时钟输入端；它们是“或”的关系。功能表见表 5.5.5。

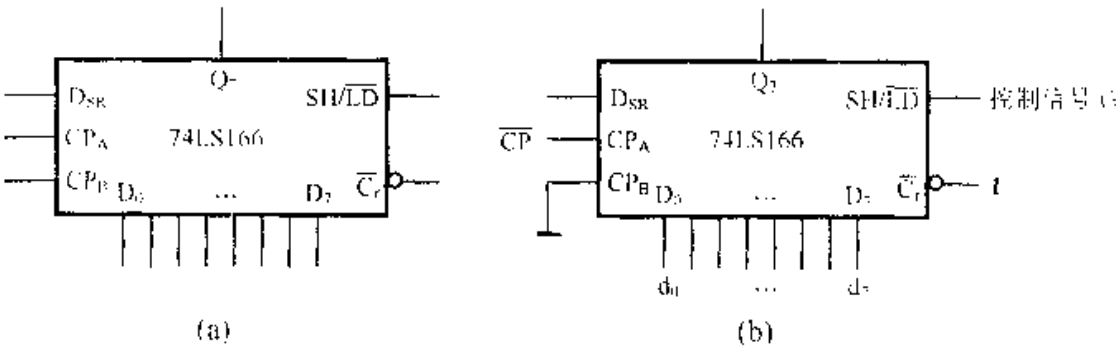


图 5.5.6 74LS166 构成 8 位并-串转换电路

(a) 逻辑框图 (b) 连接图

表 5.5.5 74LS166 功能表

输 入						输 出
\overline{C}_r	SH/\overline{LD}	CP_B	CP_A	D_{SR}	$D_0 \sim D_7$	Q_7
0	ϕ	ϕ	ϕ	ϕ	ϕ	0
1	ϕ	0	0	ϕ	ϕ	不变
1	0	0	↑	ϕ	$d_0 \sim d_7$	d_7
1	1	0	↑	0 ^①	ϕ	Q_{in}
1	1	0	↑	1 ^①	ϕ	Q_{in}
1	ϕ	1	↑	ϕ	ϕ	不变

① 0 或 1 移入内部状态 Q_{in} 。

8 位并-串转换电路的连接方式示于图 5.5.6(b)。其中, CP_B 接 0, 时钟脉冲从 CP_A 加入; 8 位并行数据 $d_0 \sim d_7$ 加至 $D_0 \sim D_7$ 端; 控制信号 G 加至 SH/\overline{LD} 端; 串出数据由 Q_7 端输出。

控制信号 G 与 CP 信号的时序波形图如图 5.5.7 所示。在 SH/\overline{LD} 端出现在第 1 个负脉冲之后到来的第 1 个 \overline{CP} 信号的上跳沿时, 移存器进行并行置数操作, Q_7 端输出 d_7 。当第 2 个 \overline{CP} 信号上跳沿出现时, SH/\overline{LD} 端为高电平 (第 1 个负脉冲已消失), 所以, 移存器进行右移操作, Q_7 端输出 d_6 。随后的 \overline{CP} 信号出现均使移存器进行右移操作, Q_7 端逐位输出 d_5, d_4, \dots 直到第 8 个 \overline{CP} 信号出现时, Q_7 端输出 d_0 。至此, 第 1 组 8 位并-串转换已经完成了。当第 9 个 \overline{CP} 信号上跳沿出现时, SH/\overline{LD} 端为低电平, 所以, 移存器又进行 8 位并行置数操作, 第 2 组的 8 位数据 $D'_0 \sim D'_7$ 被置入移存器, Q_7 端输出 D'_7 , 这样便开始第 2 组 8 位并-串转换过程。

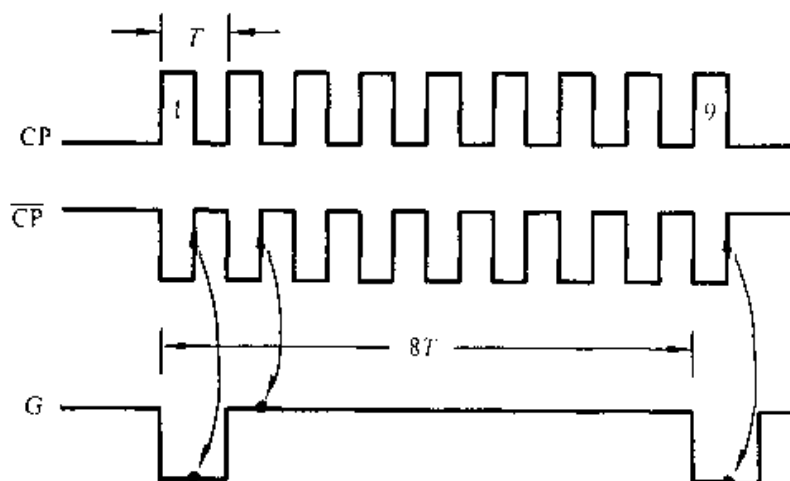


图 5.5.7 图 5.5.6(b)电路工作波形图

2. 构成数字延迟线

当移存器工作在串入-串出工作模式时, 输入信号经过逐级移位寄存后才到达输出端。输入信号移位一级需要的时间为 T_{CP} , T_{CP} 为时钟脉冲周期, 所以, 输入信号经过 n 级 (位) 移存器之后的延迟时间

$$t_d = nT_{CP}$$

在已知要求信号延迟时间 t_d 的条件下, 只需确定时钟脉冲周期 T_{CP} , 便可确定需要移位的级数 n 。如果每一级输出都作为整个数字延迟线的中间抽头, 使用就更方便、灵活了。

3. 构成脉冲序列发生器、计数器

在移存器的基础上不增加任何电路便可以构成环形或扭环形计数器; 或者增加少量门电路便可构成脉冲序列发生器。其设计方法将在下面两节介绍。

5.6 计 数 器

5.6.1 计数器的功能和分类

计数器是最常用的一种时序逻辑部件，它的基本功能是统计输入脉冲的个数。计数器所能统计的脉冲个数的最大值称为模（用 N 表示）。

计数器的种类繁多，可以从不同的角度来分类：

根据工作方式分类，可分为同步和异步两类。同步计数器的所有触发器公用一个时钟脉冲源，而它也就是被计数的输入脉冲。异步计数器只有部分触发器的 CP 信号是计数脉冲，而另一部分触发器的 CP 信号是其他触发器的输出信号，因而各级触发器的状态更新不是同时发生的。

根据进位制分类，可分为二进制、十进制和任意进制等。它们各自按其计数进位规律工作。

从逻辑功能分类，可分为加法计数器、减法计数器和可逆计数器等。加法计数器的状态变化与数的依次累加相对应；减法计数器的状态变化与数的依次递减相对应，见表 5.6.1；可逆计数器由控制信号实现相应状态的累加或递减，它可实现加法或减法计数。

表 5.6.1 计数顺序表

计 数 脉 冲	加法计数	十 进 制 数	减法计数	十 进 制 数
	二进制码		二进制码	
	$Q_2Q_1Q_0$		$Q_2Q_1Q_0$	
0	000	0	111	7
1	001	1	110	6
2	010	2	101	5
3	011	3	100	4
4	100	4	011	3
5	101	5	010	2
6	110	6	001	1
7	111	7	000	0
8	000	0	111	7

5.6.2 同步计数器

同步二进制、十进制和任意进制计数器都可以按照同步时序电路的设计方法进行设计。但是，由于计数器的某些特定情况，有些步骤可以省略或简化。

1. 同步二进制计数器

(1) 同步二进制加法计数器

例 5.6.1 设计模 8 同步二进制加法计数器。

解 (1) 画状态图：因为 $N=8$ ，故用 3 级触发器构成电路的 8 种状态 (0~7)，初态置状态 0，当达到状态 7 时，计数器满量，输出为 1，其他情况下计数器不满量，输出均为 0。状态图如图 5.6.1(a)，它属于 moore 型时序电路(没有外输入)

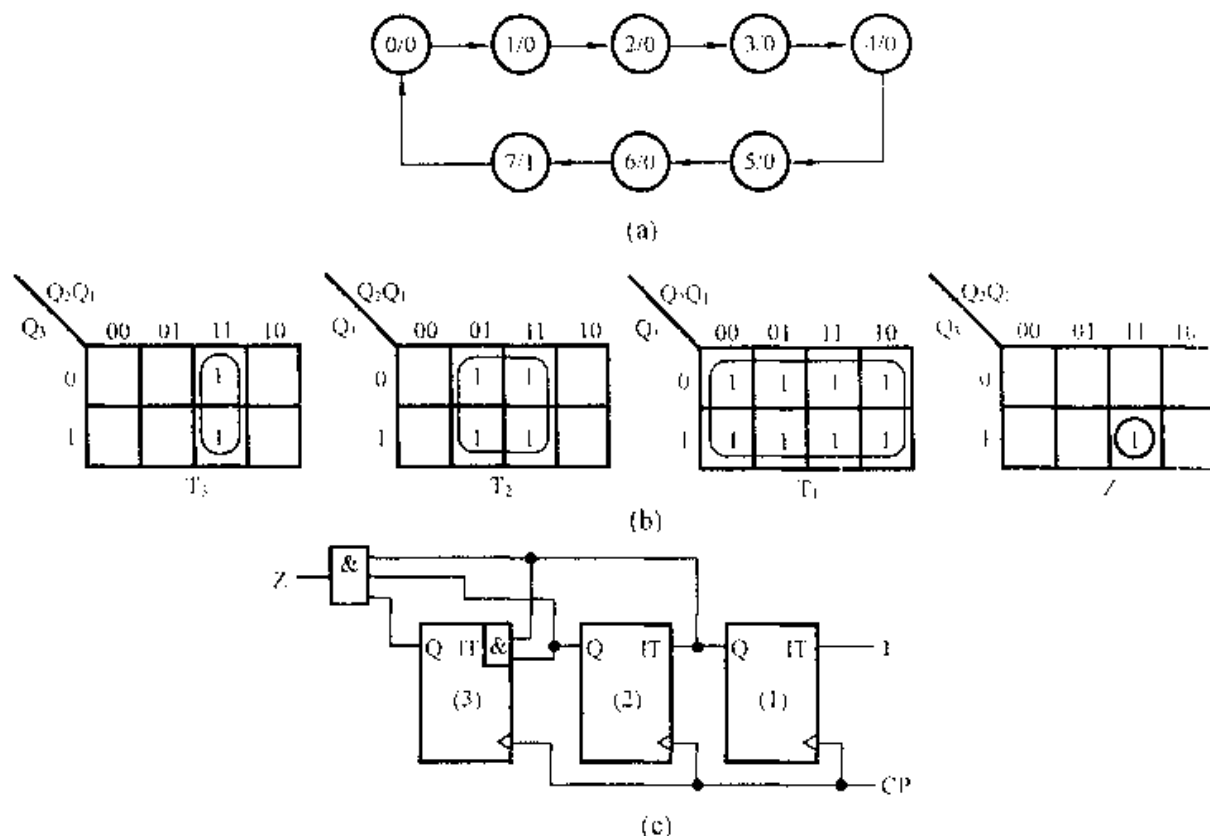


图 5.6.1 模 8 同步二进制加法计数器

(a) 状态图 (b) T_3 、 T_2 、 T_1 、 Z 的卡诺图 (c) 逻辑电路图

(2) 列状态转换真值表和激励表：

令 000=状态 0，…，111=状态 7，由图 5.6.1(a)可列状态转换真值表，当选用 T 触发器时，同时可以列出激励表，如表 5.6.2 所示。

(3) 求激励函数和输出函数：由表 5.6.2 画出 T_3 、 T_2 、 T_1 和 Z 的卡诺图，如图 5.6.1(b)所示。由此可得

$$T_3=Q_2Q_1, T_2=Q_1, T_1=1$$

$$Z=Q_3Q_2Q_1$$

表 5.6.2 模 8 计数器状态转换真值表和激励表

Q_3	Q_2	Q_1	Q_3^{n+1}	Q_2^{n+1}	Q_1^{n+1}	Z	T_3	T_2	T_1
0	0	0	0	0	1	0	0	0	1
0	0	1	0	1	0	0	0	1	1
0	1	0	0	1	1	0	0	0	1
0	1	1	1	0	0	0	1	1	1
1	0	0	1	0	1	0	0	0	1
1	0	1	1	1	0	0	0	1	1
1	1	0	1	1	1	0	0	0	1
1	1	1	0	0	0	1	1	1	1

(4) 画逻辑电路图, 如图 5.6.1 (c) 所示。

上述设计方法同样适用于减法计数器; 并可以推广到模为 2^k 同步加法 (减法) 计数器。

(2) 同步二进制可逆计数器

例 5.6.2 设计模 8 同步二进制可逆计数器。

解 (1) 画状态图: 为了构成可逆计数器, 需加控制信号 M , 当 $M=1$ 时计数器按加法计数, 当达到满量状态 7 时, 输出 1 电平, 第 8 个计数脉冲到来时电路返回到起始状态, 并且输出变为 0 电平, 这时输出一个进位脉冲; 当 $M=0$ 时, 计数器按减法计数, 从全 1 状态开始减 1 计数, 当达到全 0 状态时, 输出高电平, 第 8 个计数脉冲到来时电路返回到起始状态 7, 并且输出变为低电平, 这时输出一个借位脉冲, 状态图如图 5.6.2 (a) 所示。显然它是属于 moore 型电路。

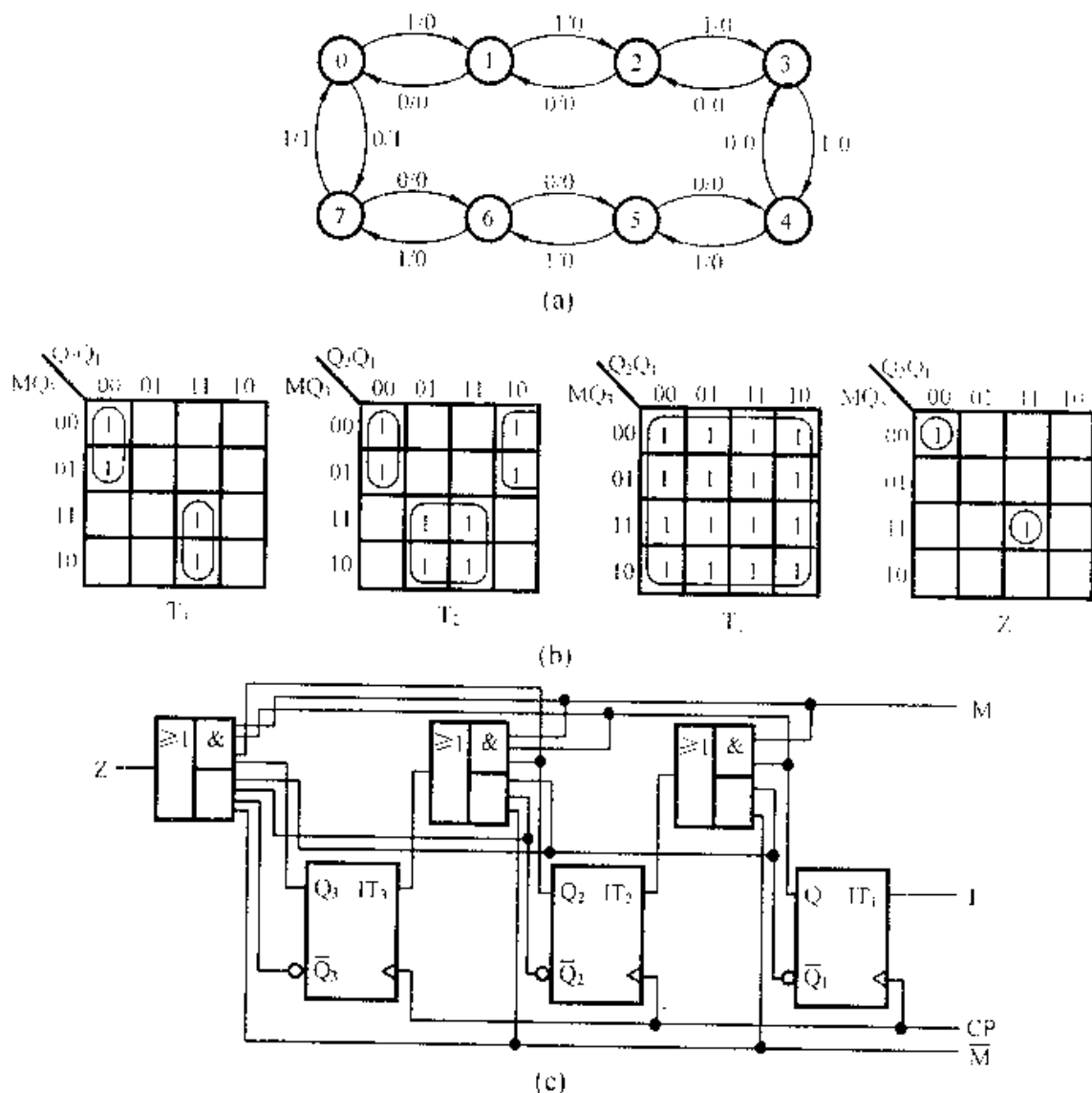


图 5.6.2 模 8 同步二进制可逆计数器

(a) 状态图 (b) 卡诺图 T_1 (c) 逻辑电路图

(2) 列状态转换真值表和激励表：令状态分配为 000 =状态 0, ..., 111 =状态 7。由图 5.6.2(a)可列出状态转换真值表，若选用 T 触发器，则同时可列出激励表，如表 5.6.3 所示。

(3) 求激励函数和输出函数：由表 5.6.3 画出 T_3 、 T_2 、 T_1 和 Z 的卡诺图，见图 5.6.2 (b)，由此可得

$$T_3 = \bar{M}\bar{Q}_2\bar{Q}_1 + MQ_2Q_1$$

$$T_2 = \bar{M}\bar{Q}_1 + MQ_1$$

$$T_1 = 1$$

$$Z = \bar{M}\bar{Q}_2\bar{Q}_1\bar{Q}_0 + MQ_2Q_1Q_0$$

表 5.6.3 例 5.6.2 的状态转换真值表和激励表

M	Q_3	Q_2	Q_1	Q_3^{n+1}	Q_2^{n+1}	Q_1^{n+1}	T_3	T_2	T_1	Z
1	0	0	0	0	0	1	0	0	1	0
1	0	0	1	0	1	0	0	1	1	0
1	0	1	0	0	1	1	0	0	1	0
1	0	1	1	1	0	0	1	1	1	0
1	1	0	0	1	0	1	0	0	1	0
1	1	0	1	1	1	0	0	1	1	0
1	1	1	0	1	1	1	0	0	1	0
1	1	1	1	0	0	0	1	1	1	1
0	1	1	1	1	1	0	0	0	1	0
0	1	1	0	1	0	1	0	1	1	0
0	1	0	1	1	0	0	0	0	1	0
0	1	0	0	0	1	1	1	1	1	0
0	0	1	1	0	1	0	0	0	1	0
0	0	1	0	0	0	1	0	1	1	0
0	0	0	1	0	0	0	0	0	1	0
0	0	0	0	1	1	1	1	1	1	1

(4) 画逻辑电路图：如图 5.6.2(c)所示。

上述设计方法可推广到模为 2^k 的二进制可逆计数器。

2. 同步十进制计数器

同步十进制计数器的设计方法与同步二进制计数器类似，不同之处在于十进制计数器要表示 10 种状态，故需要 4 级触发器构成。而 4 级触发器可以构成 16 种状态，故有 6 种状态没有使用。实用中有许多不同的 BCD 码，这些编码都可以用于

计数器。例如, 8421BCD 码十进制计数器, 2421 码十进制计数器, 格雷码计数器等。不过, 使用最广泛的还是 8421BCD 码十进制计数器, 所以, 下面以它为例来说明同步十进制计数器的设计方法。通常将 8421BCD 码十进制计数器简称为同步 BCD 码计数器。

例 5.6.3 设计 8421BCD 码计数器。

解 (1) 拟定状态图, 见图 5.6.3(a)。

(2) 列状态转换真值表和激励表: 本例指定采用 JK 触发器。如表 5.6.4 所示, 其中, 状态 10~15 是未使用状态, 故作 ϕ 处理。

表 5.6.4 例 5.6.3 的状态转换真值表和激励表

N	Q_4	Q_3	Q_2	Q_1	Q_4^{n+1}	Q_3^{n+1}	Q_2^{n+1}	Q_1^{n+1}	J_4	K_4	J_3	K_3	J_2	K_2	J_1	K_1	Z
0	0	0	0	0	0	0	0	1	0	ϕ	0	ϕ	0	ϕ	1	ϕ	0
1	0	0	0	1	0	0	1	0	0	ϕ	0	ϕ	1	ϕ	0	1	0
2	0	0	1	0	0	0	1	1	0	ϕ	0	ϕ	ϕ	0	1	0	0
3	0	0	1	1	0	1	0	0	0	ϕ	1	ϕ	ϕ	1	ϕ	1	0
4	0	1	0	0	0	1	0	1	0	ϕ	ϕ	0	0	ϕ	1	0	0
5	0	1	0	1	0	1	1	0	0	ϕ	ϕ	0	1	ϕ	0	1	0
6	0	1	1	0	0	1	1	1	0	ϕ	ϕ	0	ϕ	0	1	ϕ	0
7	0	1	1	1	1	0	0	0	1	ϕ	ϕ	1	ϕ	1	ϕ	1	0
8	1	0	0	0	1	0	0	1	ϕ	0	0	ϕ	0	0	1	ϕ	0
9	1	0	0	1	0	0	0	0	ϕ	1	0	ϕ	0	ϕ	ϕ	1	1
10	0	0	1	0	ϕ				ϕ								ϕ
\vdots	\vdots																
15																	

(3) 求激励函数和输出函数: 由表 5.6.4 画出 J_2K_2 至 J_4K_4 和 Z 的卡诺图。见图 5.6.3(b)。由此可得

$$J_4=Q_3Q_2Q_1$$

$$K_4=Q_1$$

$$J_3=K_3=Q_2Q_1$$

$$J_2=\overline{Q_4}Q_1$$

$$K_2=Q_1$$

$$J_1=K_1=1 \quad (\text{由表求得})$$

$$Z=Q_4Q_1$$

(4) 画逻辑图, 见图 5.6.3(c)。

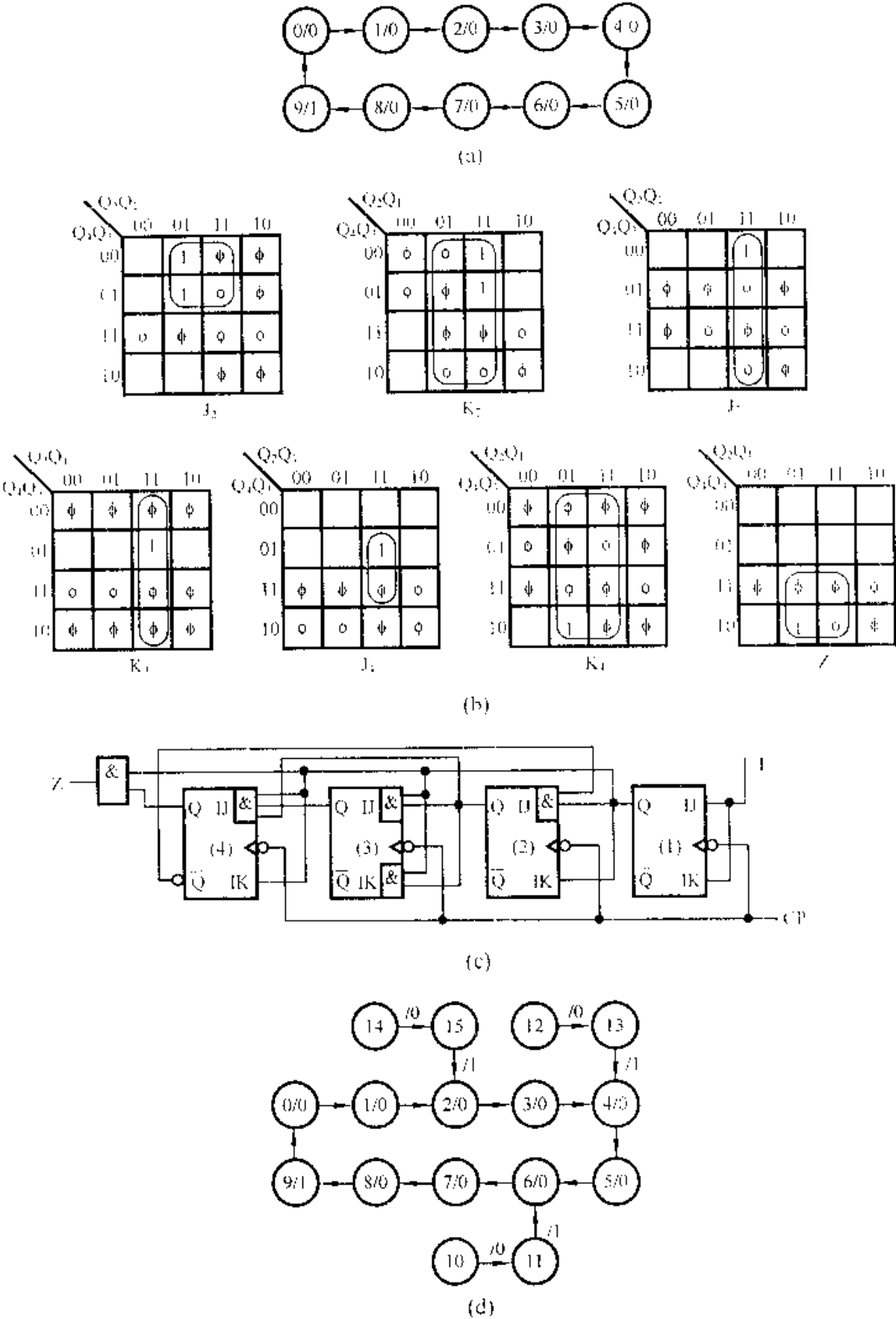


图 5.6.3 同步 BCD 码计数器
(a) 状态图 (b) 卡诺图 (c) 逻辑电路图 (d) 完整的状态图

(5) 进行状态检查和画出完整的状态图: 在正常工作时, 计数器只使用了状态 0~9, 而状态 10~15 是不使用的, 一般不会出现, 但是, 在受到干扰或电源波动等特殊情况下, 计数器可能进入不使用状态, 那么, 在时钟脉冲作用下, 它的次态转到何处? 能否经过几个时钟脉冲作用之后, 重新进入某个使用状态, 从而回到工作循环? 为此应对不使用状态逐一检查。例如, 在特殊情况下计数器进入状态 12, 此时 $Q_4Q_3Q_2Q_1=1100$, 所以有 $J_4K_4=00, J_3K_3=00, J_2K_2=00, J_1K_1=11$ 。在 CP 脉冲作用下状态 12 将转换到状态 13, 而它也是未使用状态, 故还应继续检查, 此时, $Q_4Q_3Q_2Q_1=1101$, 所以, 有 $J_4K_4=01, J_3K_3=00, J_2K_2=01, J_1K_1=11$ 。在 CP 脉冲作用下状态 13 将转换到状态 4(0100), 即进入工作循环了。用同样方法对其他未使用状态逐一检查, 便可画出完整状态图, 如图 5.6.3(d)所示。

全部状态都能进入工作循环而无孤立状态的时序电路称为自行启动电路; 假如有某个或某些状态不能进入工作循环, 即存在孤立状态, 这样的时序电路称为非自行启动电路, 在设计中应该采取措施使之变成自行启动电路。关于这个问题将在下面详细讨论。

3. 移位计数器

在同步计数器中, 有一类是在移存器的基础上构成计数器, 所以称为移位计数器, 或称为移存器型计数器。最简单的移位计数器有环形计数器和扭环形计数器两种。

(1) 环形计数器

用 D 或 JK 触发器构成的移存器都可以构成环形计数器, 见图 5.6.4(a)和(b)。现以图(a)为例来说明环形计数器的设计方法。

假如将各级触发器预置在 0001 状态, 则在计数脉冲作用下, 环形计数器将按表 5.6.5 前 4 行工作, 其状态图为图 5.6.4(c)。4 位环形计数器有 16 个状态, 只有其中的 4 个状态是属于工作状态, 其他都是非工作状态。状态图有 6 个循环, 只有 1 个是工作循环, 其他 5 个是非工作循环。假如偶然原因使电路进入非工作循环, 则电路就不能回到工作循环中去(通常称之为堵塞现象), 所以, 这种环形计数器是非自行启动的。为了使电路正常工作, 必须采取措施消除堵塞现象, 使它变成自行启动电路。

消除堵塞的办法是切断非工作循环, 并使每个非工作循环中的任何状态都能转回到工作循环中的某一个状态。实现的方案很多, 为了使附加电路简单, 可按下述方法设计。

(a) 附加电路仅仅改变第一级触发器状态, 即只改变计数器状态 $Q_4Q_3Q_2Q_1$ 中的 Q_1 , 换言之, 附加电路输出仅加在第一级触发器输入端。

(b) 计数器具有 5 个非工作循环, 通常是先切断一个(或两个)最简单的非工作循环, 设计出附加电路, 然后进行状态检查, 如果已不存在非工作循环了, 那么,

设计便告完成。假如还有非工作循环，则再切断尚存的另一个非工作循环，重新设计附加电路，然后再进行状态检查。一般只需反复 1~2 次便可消除堵塞，使它变为自行启动电路。

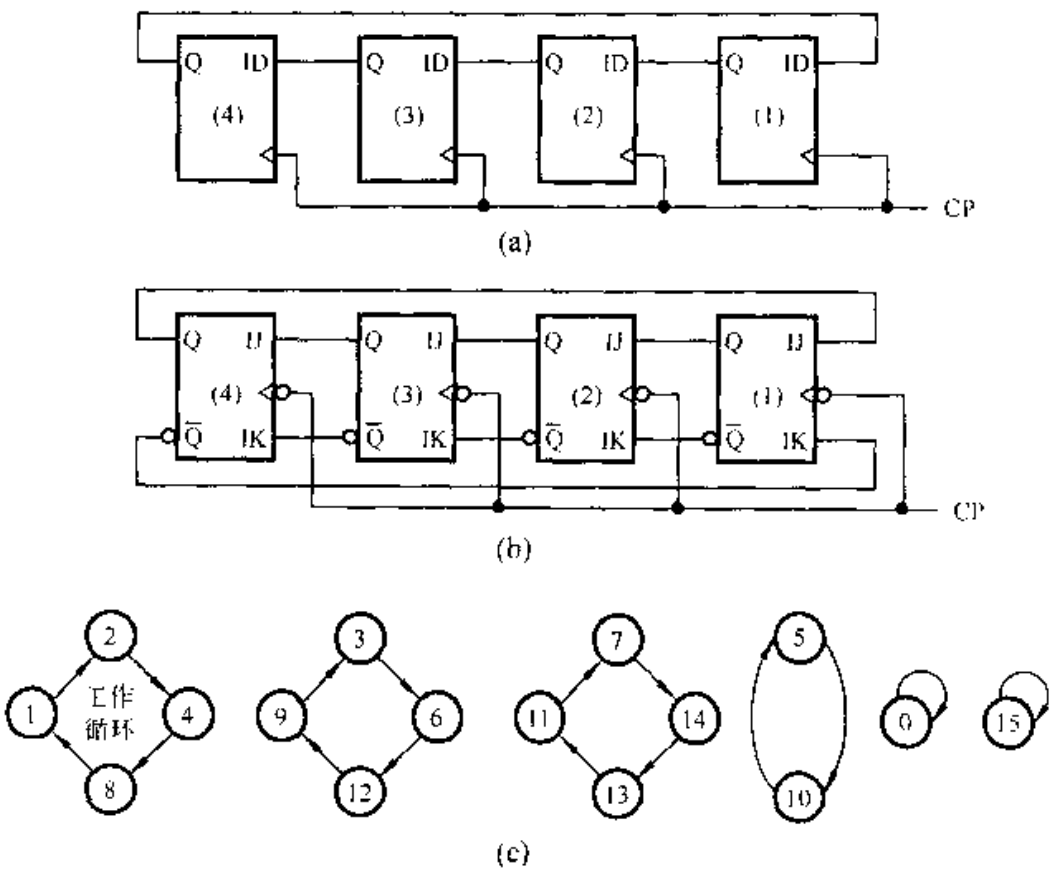


图 5.6.4 环形计数器

(a) D 型环形计数器 (b) JK 型环形计数器 (c) 状态图

表 5.6.5 环形计数器的状态表和激励表

态序	Q_4	Q_3	Q_2	Q_1	Q_4^{n+1}	Q_3^{n+1}	Q_2^{n+1}	Q_1^{n+1}	D_1
1	0	0	0	1	0	0	1	0	0
2	0	0	1	0	0	1	0	0	0
4	0	1	0	0	1	0	0	0	0
8	1	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	1	1
15	1	1	1	1	1	1	1	0	0
3	0	0	1	1	ϕ				ϕ
5	0	1	0	1					
7	1	0	1	1					
14	1	1	1	0					

本例最简单的非工作循环就是状态“0”和“15”的循环,可首先切断它。设计一个附加电路使状态 0000 转到 0001,使 1111 转到 1110(显然,这里都满足附加电路只改变第一级触发器状态的原则)。状态 0001 已是工作循环中的一个状态,状态 1110 虽然不是工作状态,但是,由于附加电路的作用,非工作循环 $11 \rightarrow 7 \rightarrow 14 \rightarrow 13 \rightarrow 11$ 将被切断,状态 14 还要转换到其他中间状态,有可能经过几个中间状态之后进入工作循环,具体情况尚待检查。根据上述功能要求可列出状态转换真值表 5.6.5。表中,1、2、4、8 是工作状态,0 和 15 是为切断非工作循环所加的,其余状态作随意态处理。由于已规定附加电路只改变第一级触发器输入,其他三级触发器输入与原电路相同,故只需求出第一级触发器的激励函数。例如,画出 D_1 卡诺图 5.6.5(a)、由图可得 $D_1 = \overline{Q_3}\overline{Q_2}\overline{Q_1}$ 。自启动环形计数器逻辑电路图见图 5.6.5(b)。

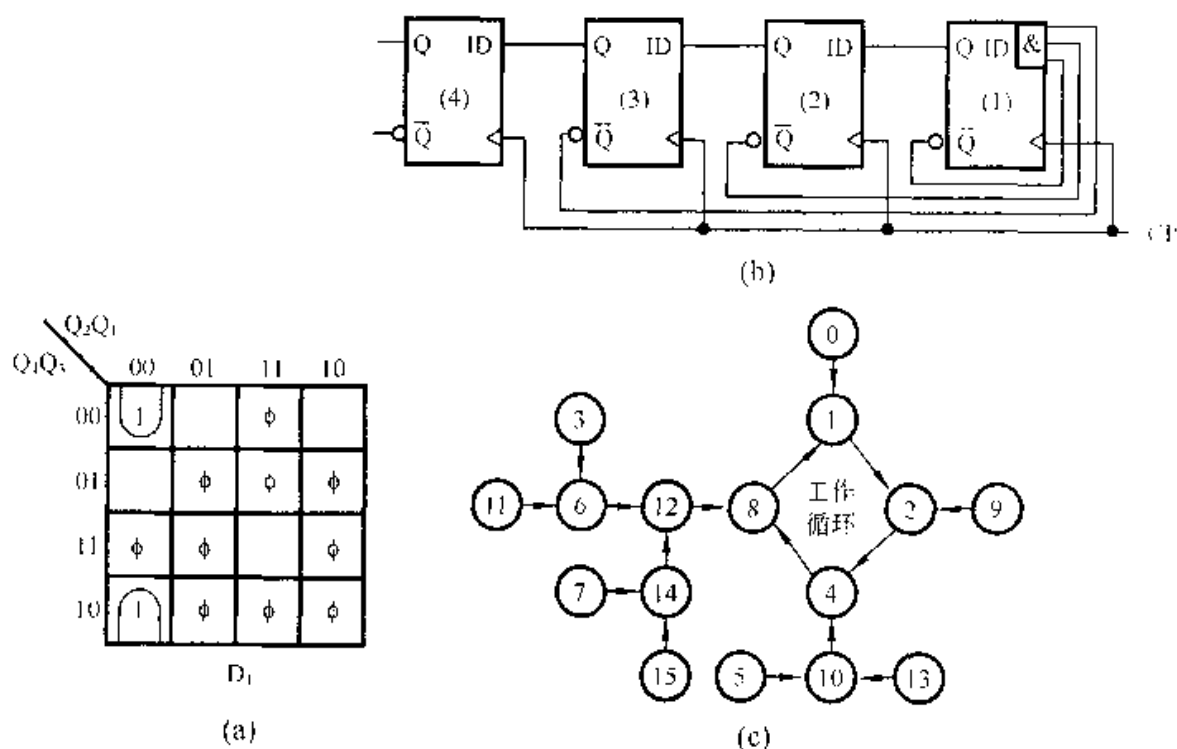


图 5.6.5 自启动环形计数器

(a) D_1 的卡诺图 (b) 逻辑电路图 (c) 完整的状态图

附加电路完成之后,要进行状态检查以便确定是否还存在非工作循环,有必要重新设计附加电路,最后画出完整的状态图。例如,对图 5.6.5(b)所示电路,检查状态 11,此时,状态为 $Q_4Q_3Q_2Q_1=1011$,所以有 $D_1=\overline{Q_3}\overline{Q_2}\overline{Q_1}=0$,在下一个计数脉冲作用后,状态 11 转到状态 6(0110),但是,状态 6 也不是工作状态,应继续检查,此时 $D_1=0$,在下一个计数脉冲作用后状态 6 转换到状态 12(1100),显然还需要进一步检查,此时 $Q_4Q_3Q_2Q_1=1100$,所以有 $D_1=0$,在下一个计数脉冲作用后状态 12 转换到状态 8(1000),从而进入工作循环。用同样方法检查其他非工作状态,检

查结果不存在非工作循环(即无孤立状态),设计便告完成,最后画出完整状态图,如图 5.6.5(c)所示。

(2) 扭环形计数器

扭环形计数器电路与环形计数器很类似,其区别仅在于第一级触发器输入信号不同。D 触发器构成的扭环形计数器有 $D_1 = \bar{Q}_4$; JK 触发器构成的扭环形计数器有 $J_1 = \bar{Q}_4$ 和 $K_1 = Q_4$ 。逻辑电路图和状态图,见图 5.6.6(a)、(b)和(c)。

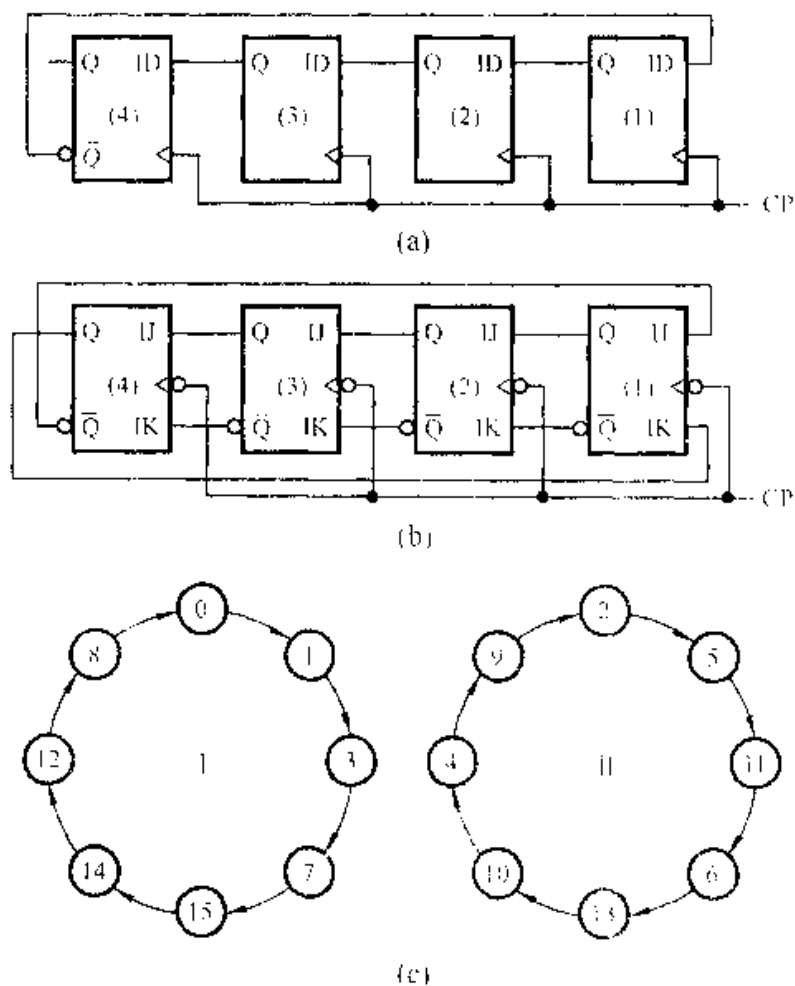


图 5.6.6 扭环形计数器

(a) 扭环形计数器之一 (b) 扭环形计数器之二 (c) 状态图

假如状态预置在 0 状态,则图 5.6.6(c)中 I 为工作循环,II 为非工作循环。可见这种扭环形计数器也存在堵塞现象。仍可用上述方法消除堵塞,这里应考虑既要切断非工作循环,又只能改变第一级触发器输入。可供选择的方案有几个。例如,在非工作循环(II)中的 $11 \rightarrow 6$, $6 \rightarrow 13$, $4 \rightarrow 9$, $9 \rightarrow 2$ 这几处切断均符合条件。假如在 $9 \rightarrow 2$ 处切断,那么,在增加附加电路后,状态 9 应转换到工作状态 3(0011),从而进入工作循环。根据上述功能要求可列状态转换真值表(见表 5.6.6),前 8 行是工作状态,第 9 行是为消除堵塞而增加的,其余状态作随意态处理。由于只

改变了第一级触发器的输入,故只需填第一级的激励值,其他各级电路不变。例如,对于图 5.6.6(b)电路,可以只画出 J_1 、 K_1 卡诺图,如图 5.6.7(a)所示。由图可得 $J_1 = \overline{Q_4}$ 、 $K_1 = Q_4 Q_3$ 。自启动扭环形计数器逻辑电路如图 5.6.7(b)所示,在设计完成之后应该进行状态检查,画出完整状态图,如图 5.6.7(c)所示。

表 5.6.6 扭环形计数器的状态转换真值表和激励表

态序	Q_4	Q_3	Q_2	Q_1	Q_4^{n+1}	Q_3^{n+1}	Q_2^{n+1}	Q_1^{n+1}	D	J_1	K_1
0	0	0	0	0	0	0	0	1	1	1	ϕ
1	0	0	0	1	0	0	1	1	1	0	0
3	0	0	1	1	0	1	1	1	1	0	0
7	0	1	1	1	1	1	1	1	1	ϕ	0
15	1	1	1	1	1	1	1	0	0	0	1
14	1	1	1	0	1	1	0	0	0	0	ϕ
12	1	1	0	0	1	0	0	0	0	0	ϕ
8	1	0	0	0	0	0	0	0	0	0	ϕ
9	1	0	0	1	0	0	1	1	1	ϕ	0

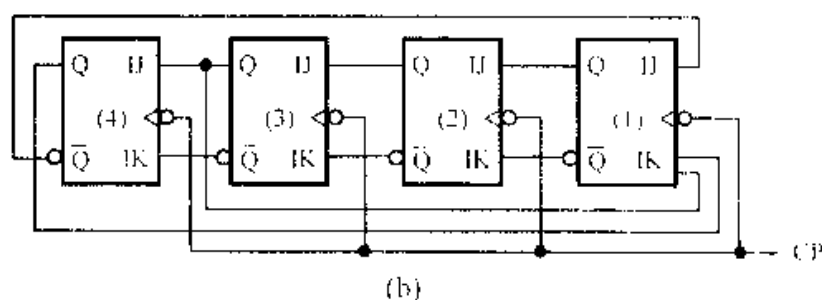
Q_2Q_1		Q_4Q_3			
		00	01	11	10
Q_4Q_3	00	1	0	0	0
	01	0	0	0	0
	11		0	0	
	10		0	0	0

 J_1

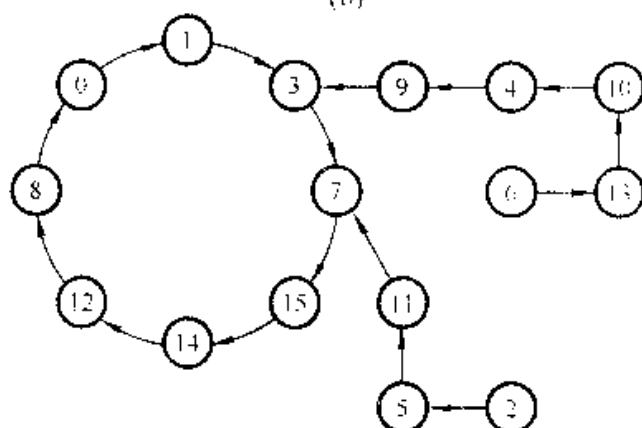
$Q_4 Q_3$ \ $Q_2 Q_1$		$Q_2 Q_1$			
		00	01	11	10
$Q_4 Q_3$	00	0			0
	01	0	ϕ		0
	11	0	ϕ	1	ϕ
	10	0		ϕ	0

 K_1

(a)



(b)



(c)

图 5.6.7 自启动扭环形计数器

(a) J_1 、 K_1 卡诺图 (b) 逻辑电路图 (c) 完整的状态图

5.6.3 异步计数器

在异步计数器中, 各级触发器的时钟信号并不是都来自计数脉冲, 其中一部分来自其他触发器的输出端 Q 或 \bar{Q} , 所以, 合理选择各级触发器的时钟信号是一个重要问题, 需采用时钟覆盖法^[4]来解决, 但是, 其设计过程较繁琐, 并且透明性较差。所以, 实际工程中广泛采用串联进位的电路结构, 即计数脉冲加在第一级触发器的时钟输入端, 其他各级的时钟输入则来自它相邻低位触发器的 Q 或 \bar{Q} 端, 通常称之为异步串行计数器或者简称为异步计数器。

设计这类异步计数器的方法具有较强的规律性, 主要需解决 3 个问题: 每一级触发器如何构成为翻转(计数)触发器? 级间如何连接? 如何满足模数要求?

1. 异步二进制计数器

(1) 异步二进制加法计数器

为了构成异步计数器, 每一级触发器必须为计数触发器, 即 T 触发器。各类触发器构成 T 触发器的连接图如图 4.4.2 所示。

图 5.6.8(a) 是模 8 异步加法计数器(即 3 位异步二进制计数器)的逻辑电路图, 图 (b) 是工作波形, 计数脉冲加在第一级, 而其他各级的时钟输入来自低位触发器的 \bar{Q} 端。由于 D 触发器是上升沿触发, 所以, 计数脉冲的个数与计数器状态之间的转换关系符合表 5.6.1 中的加法计数器, 即实现了计数器的状态随计数脉冲的输入而累加。

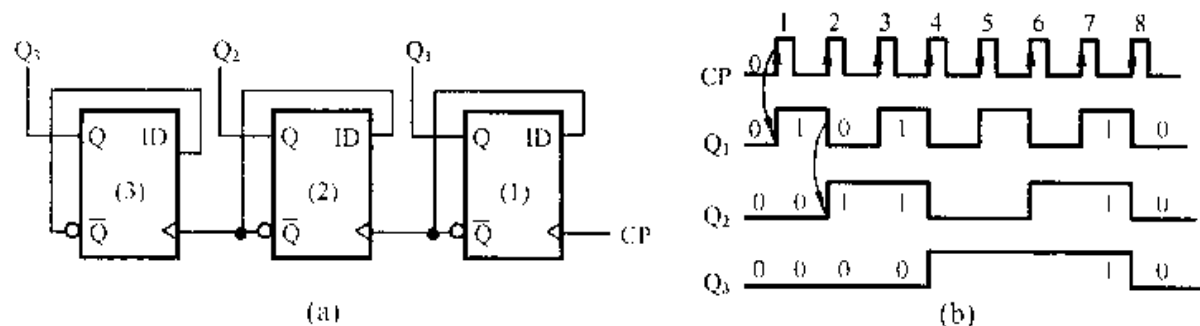


图 5.6.8 D 型模 8 异步加法计数器

(a) 逻辑电路图 (b) 波形图

图 5.6.9 示出由 JK 触发器构成的异步二进制加法计数器, 由于 JK 触发器(指主-从式)的状态翻转是发生在时钟脉冲下降沿时刻, 故每级 CP 端应与前级 Q 端相连, 画工作波形时必须注意这一点(请读者画出它的波形图)。

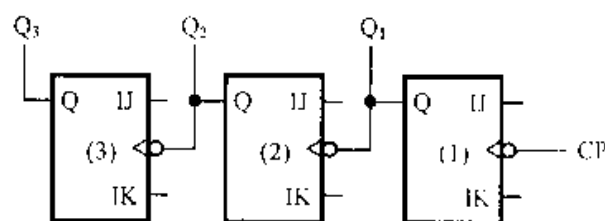


图 5.6.9 JK 型模 8 异步加法计数器

(2) 异步二进制减法计数器

图 5.6.10(a)和(b)分别示出由 D 和 JK 触发器构成的模 8 异步减法计数器。图(c)是图(b)所示电路的工作波形。只要注意到它们的初态应置 **111** 状态, 并且注意到它们的状态翻转是发生在时钟脉冲的不同时刻, 那么就不难确定每一级 CP 端与前一级的连接规律和画出工作波形, 读者可以自己练习。

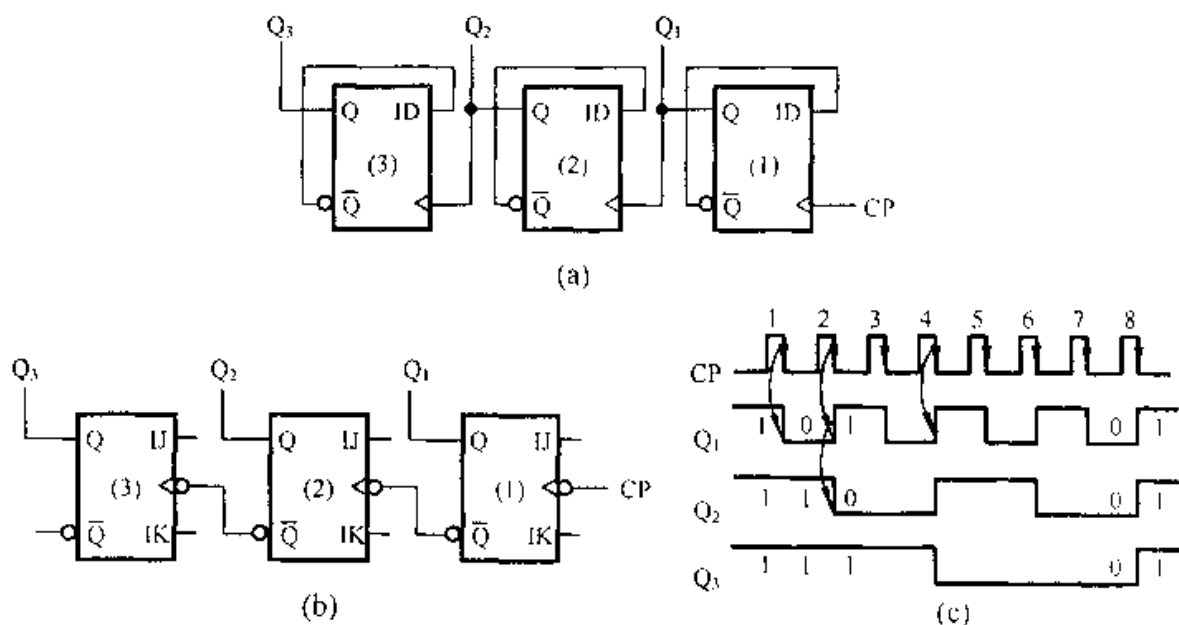


图 5.6.10 模 8 异步减法计数器

(a) D 型模 8 异步减法计数器 (b) JK 型模 8 异步减法计数器 (c) 图(b)所示电路的波形图

2. 异步十进制计数器

以异步十进制加法计数器为例来说明它的设计方法。

异步十进制加法计数器, 其模 $N=10$, 计数范围是状态 $0 \sim 9$, 为了表示 10 种状态需用 4 级触发器。

令状态分配为 **0000**=状态 0, \dots , **1001**=状态 9。

异步十进制加法计数器的设计思想是将 4 级触发器按异步二进制加法计数器连接, 以便实现累加。在第 9 个计数脉冲作用后, 计数器状态为 **1001**, 而在第 10 个计数脉冲作用下, 计数器回到状态 **0000**, 然后开始下一周期循环。所以, 设计的关键是怎样使计数器状态从状态 **1001** 变回到 **0000**。

定义 $N-1$ 的二进制代码中出现连续 1 的最前位(即最低位)称为首 1 位, 单个 1 也算首 1 位; 出现连续 0 的最前位称为首 0 位, 单个 0 也算首 0 位。对于本例 $N=10$, 得到 $N-1$ 的二进制代码表示为 **1001**, 即 $Q_4Q_3Q_2Q_1=1001$, 因此, 首 1 位是 Q_4 和 Q_1 , 首 0 位是 Q_2 。

为了实现十进制计数器的逻辑功能, 要求在第 10 个计数脉冲作用下, Q_4 由 1 变为 0, Q_1 也由 1 变为 0, 而 Q_3 和 Q_2 维持 0 不变, 假如选用 JK 触发器来构成电路, 各级之间可按异步二进制加法计数器连接。为了使状态 **1001** 回到 **0000**, 可以加一

个控制信号 C_A ，在状态 $0 \sim 8$ 时，令 $C_A=0$ ，计数器按二进制加法计数，在状态 9 时，令 $C_A=1$ ，由 C_A 信号控制首 1 位触发器能够加进计数脉冲。由于第一级触发器 CP 端直接和计数脉冲接通，故实际上只需由 C_A 控制第 4 级触发器(A_4)，使它能加进第 10 个计数脉冲。同时，为了维持 Q_3 和 Q_2 为 0，需另加一信号 C_B 去控制首 0 位触发器 A_2 的 J 端，在状态 $0 \sim 8$ 时，令 $C_B=1$ ，计数器按二进制加法计数，在状态 9 时，令 $C_B=0$ ，使 A_2 在串行进位脉冲作用下的新状态维持在 0 状态。由于 A_2 状态不变， A_3 得不到触发，故也维持 0 状态。

依据上述分析可以写出

$$\left. \begin{aligned} CP_{A4} &= \bar{C}_A Q_3 + C_A CP \\ J_2 &= C_B \cdot 1 + \bar{C}_B \cdot 0 = C_B \cdot 1 = C_B \end{aligned} \right\} \quad (5.6.1)$$

依据对控制信号 C_A 和 C_B 的要求, 可列出 C_A 和 C_B 的真值表 (见表 5.6.7), 由真值表画出 C_A 卡诺图 (见图 5.6.11 (a))。由图可得

$$\left. \begin{aligned} C_A &= Q_4 Q_1 \\ C_B &= \bar{C}_A = \overline{Q_4 Q_1} \end{aligned} \right\} \quad (5.6.2)$$

由式 (5.6.1) 和式 (5.6.2) 画逻辑电路图, 如图 5.6.11 (b) 所示。

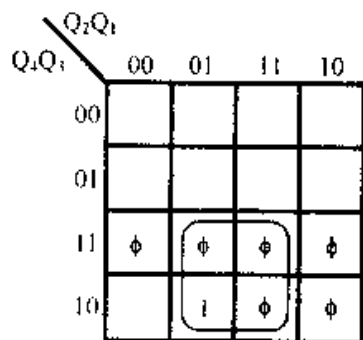
实际上, 在熟悉了上述设计方法之后, 不必列真值表就可直接得到 C_A 的逻辑式。对于本例, 在状态 0~8 时, 令 $C_A=0$; 在状态 9 时, 令 $C_A=1$; 在状态 10~15 时, C_A 作随意态处理。故有

$$C_A = m_9 + \sum \phi(10, 11, 12, 13, 14, 15)$$

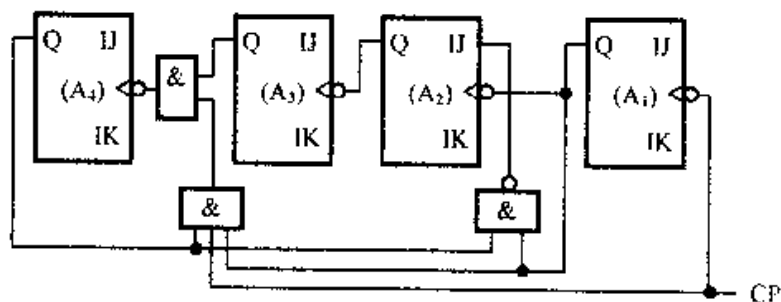
由上式画出的 C_A 卡诺图就是图 5.6.11(a)。

表 5.6.7 C_A 的真值表

序号	Q_1	Q_2	Q_3	Q_4	C_A
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	ϕ
\vdots		\vdots			
15	1	1	1	1	



(a)



(b)

图 5.6.11 异步十进制计数器

3. 异步任意进制计数器

异步十进制计数器设计方法可以推广到异步任意进制计数器,下面以 D 触发器构成模 13 异步加法计数器为例来小结一下异步计数器的设计方法,具体设计步骤如下:

(a) 确定触发器级数: 由公式 $2^k \geq N > 2^{k-1}$ 确定, 所以 $k=4$ 。

(b) 状态分配: 令 0000=状态 0, ..., 1100=状态 12。

(c) 确定首 1 位和首 0 位: 用二进制码表示 $N-1$ 的状态, 即 $Q_4Q_3Q_2Q_1$ 为 1100, 故首 1 位为 Q_3 , 首 0 位为 Q_1 。

(d) 加控制信号 C_A 和 C_B : 在状态 0~11 时按二进制计数器连接, 而在状态 12 时, 触发器 A_3 能够通过 C_A 加进第 13 个计数脉冲, 使新状态变为 0, 而 C_B 加至触发器 A_1 的 D 端, A_1 能够通过 C_B 维持新状态仍然为 0。

现在的问题是如何使第 13 个计数脉冲通过 C_A 加到 A_3 的 CP_{A_3} 上。在 0~11 状态时, $C_A=0$, 计数器保持二进制连接, 所以 CP_{A_3} 应该与 \bar{Q}_2 端相连接; 而在状态 12 时, $C_A=1$, CP_{A_3} 端应与计数脉冲输入端 CP 连接, 所以有

$$CP_{A_3} = C_A \cdot CP + \bar{C}_A \cdot \bar{Q}_2$$

用同样的方法可以确定首 0 位触发器 A_1 的输入端 D_1 的信号。因为在 0~11 状态时, $C_B=1$, 保持 A_1 为计数触发器, 即 $D_1=\bar{Q}_1$; 在状态 12 时, $C_B=0$, 为了使新状态维持 0 态不变, 故有 $D_1=0$, 所以得到 $D_1=C_B \cdot \bar{Q}_1$ 。

(e) 求 C_A 和 C_B : 因为状态 0~11 时, $C_A=0$; 状态 12 时, $C_A=1$; 在状态 13~15 时, C_A 作随意态处理, 所以有

$$C_A = m_{12} + \sum \phi(13, 14, 15)$$

由上述画 C_A 卡诺图 (见图 5.6.12 (a)), 由图写出

$$C_A = Q_4 Q_3$$

所以 $C_B = \bar{C}_A = \overline{Q_4 Q_3}$ 。

(f) 画逻辑电路图: 见图 5.6.12(b)。

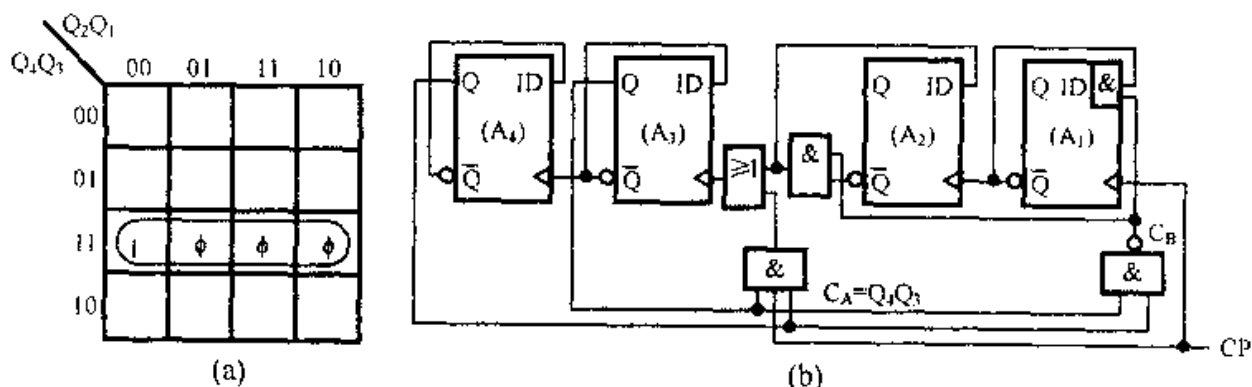


图 5.6.12 异步模 13 加法计数器

(a) C_A 卡诺图 (b) 逻辑电路图

5.6.4 中规模集成计数器

国产中规模集成计数器的产品种类很多。异步计数器的主要产品有 2-5-10 进制计数器、2-8-16 进制计数器、双十进制计数器和双 4 位二进制计数器等。同步计数器的主要产品有十进制计数器、十进制可逆计数器、4 位二进制计数器和 4 位二进制可逆计数器等。异步计数器具有电路结构简单和使用方便等优点，但是，工作速度较低；同步计数器的工作速度高，所以应用领域更广阔，其电路结构分为同步预置和异步预置两种。

1. 4 位二进制同步计数器

4 位二进制同步计数器是一种加法计数器。图 5.6.13(a)是 74LS161 型 4 位二进制同步计数器的逻辑电路图，图(b)是逻辑框图。

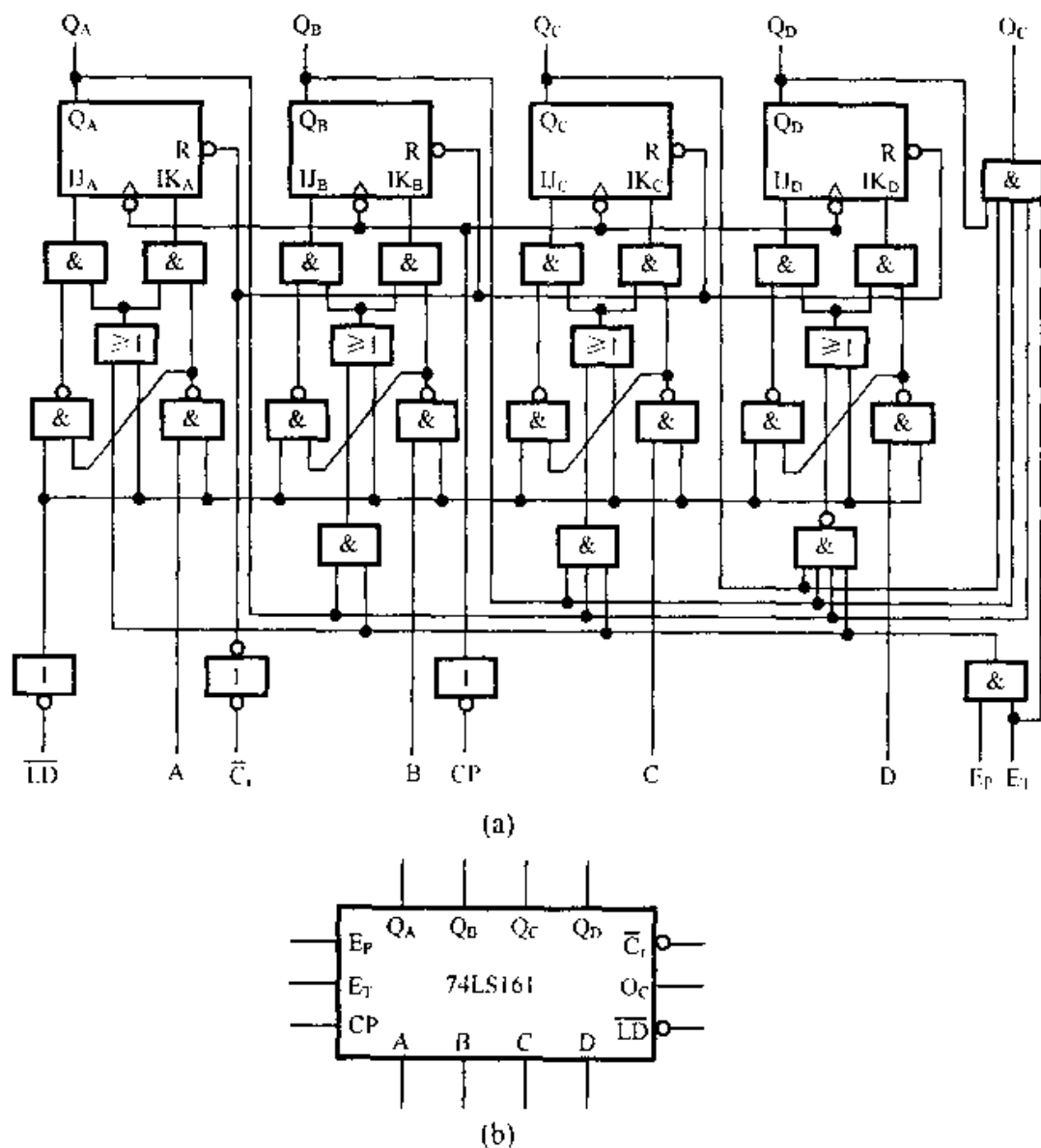


图 5.6.13 74LS161 型 4 位同步二进制计数器
(a) 逻辑电路图 (b) 逻辑框图

\overline{C}_r 是清除端, \overline{LD} 是置数(预置)端, 它们都是低电平有效; E_p 和 E_T 是使能端, 高电平有效; O_c 是进位输出(正脉冲); $A \sim D$ 是数据输入端。

CP 脉冲通过一个非门加到各级 JK 触发器的 CP 端, 所以, 计数器是在 CP 脉冲的上跳沿动作。只要 E_p 和 E_T 信号中有一个(或两个)为低电平, 则组件不能选通, 即计数器保持状态不变。数据预置的方法是采用同步预置的电路结构, 即采用 JK 预置结构, 当 $\overline{LD}=0$ 时, 将数据输入通道的所有与门打开, 输入数据便加到相应触发器的 J、K 端。例如, 对最低位触发器, 则有 $J=A$, $K=\overline{A}$, 各级触发器的 J、K 端都得到相应的数据, 为计数器的状态更新作好准备, 在 $\overline{LD}=0$ 之后再出现 CP 脉冲作用时, 计数器便预置在输入数据状态, 由于它与 CP 脉冲是同步的, 所以称为同步预置(或 JK 预置)。该组件的功能表如表 5.6.8 所示。工作波形如图 5.6.14 所示。

表 5.6.8 74LS161 的功能表

输 入								输 出			
\overline{C}_r	\overline{LD}	E_p	E_T	A	B	C	D	Q_A	Q_B	Q_C	Q_D
0	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	0	0	0	0
1	0	ϕ	ϕ	A	B	C	D	A	B	C	D
1	1	0	ϕ	ϕ	ϕ	ϕ	ϕ	保 持			
1	1	ϕ	0	ϕ	ϕ	ϕ	ϕ	保 持			
1	1	1	1	ϕ	ϕ	ϕ	ϕ	计 数			

注: Q_D 为高位。

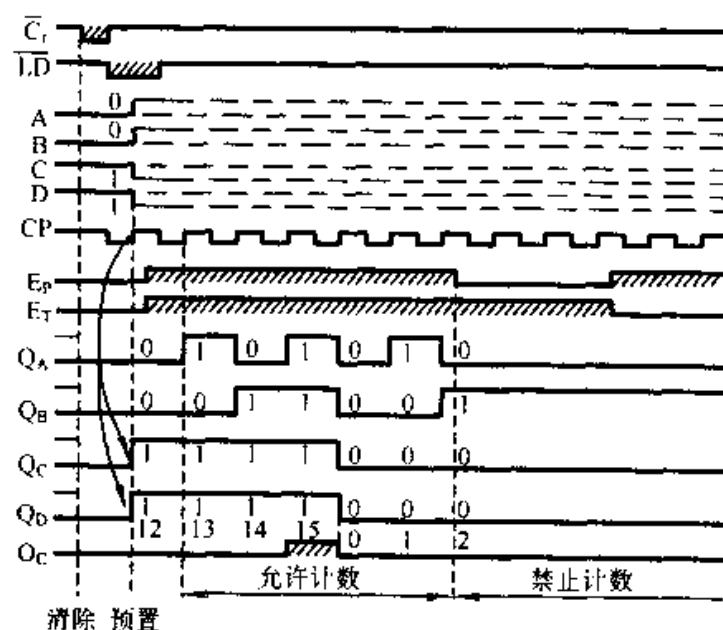


图 5.6.14 74LS161 的波形图

同步十进制加法计数器 74LS160 也采用了与 74LS161 类似的电路结构,所以它们的功能表及逻辑框图也相同,其差别仅在于 74LS160 是十进制而 74LS161 是十六进制; 74LS163 是 4 位二进制同步计数器,它也是一种加法计数器,它与 74LS161 的差别在于清零方式不同,前者采用同步清零方式,而后者采用异步清零方式(关于清零方式下面还将进一步讨论)。它们的逻辑框图完全相同。

2.4 位二进制同步加/减计数器

(1) 双时钟加/减计数器

双时钟 4 位二进制同步加/减计数器 74LS193 的逻辑电路示于图 5.6.15(a),图(b)是逻辑框图。

该组件具有双时钟 CP_+ 和 CP_- 。当作加法计数时,令 $CP_-=1$,计数脉冲从 CP_+ 加入;当减法计数时,令 $CP_+=1$,计数脉冲从 CP_- 加入。计数器的状态更新在计数脉冲的上升沿动作。每级触发器具有两个置 0(复位)端 \overline{R}_{D1} 和 \overline{R}_{D2} ,一个置 1 端 \overline{S}_D ,各位触发器都可以利用清除端 C_r 的正脉冲“清 0”。该组件还具有预置能力,当预置控制端 \overline{LD} 为低电平时,倒相输出的高电平将各位数码通道打开,各位数码的原码和反码分别传送到相应触发器的 \overline{R}_{D1} 端和 \overline{S}_D 端,实现 4 位数码并行预置。例如,在 $C_r=0$ 和 $\overline{LD}=0$ 时, A 触发器的 \overline{R}_{D1} 端得到 A,而在 \overline{S}_D 端得到 \overline{A} ,所以, A 触发器被预置在 A 状态。在预置脉冲结束之后,即可进行加法计数或减法计数。

显然可见,预置与 CP_+ (或 CP_-)无关,所以是异步预置结构(或称 \overline{R}_D 、 \overline{S}_D 预置结构)。

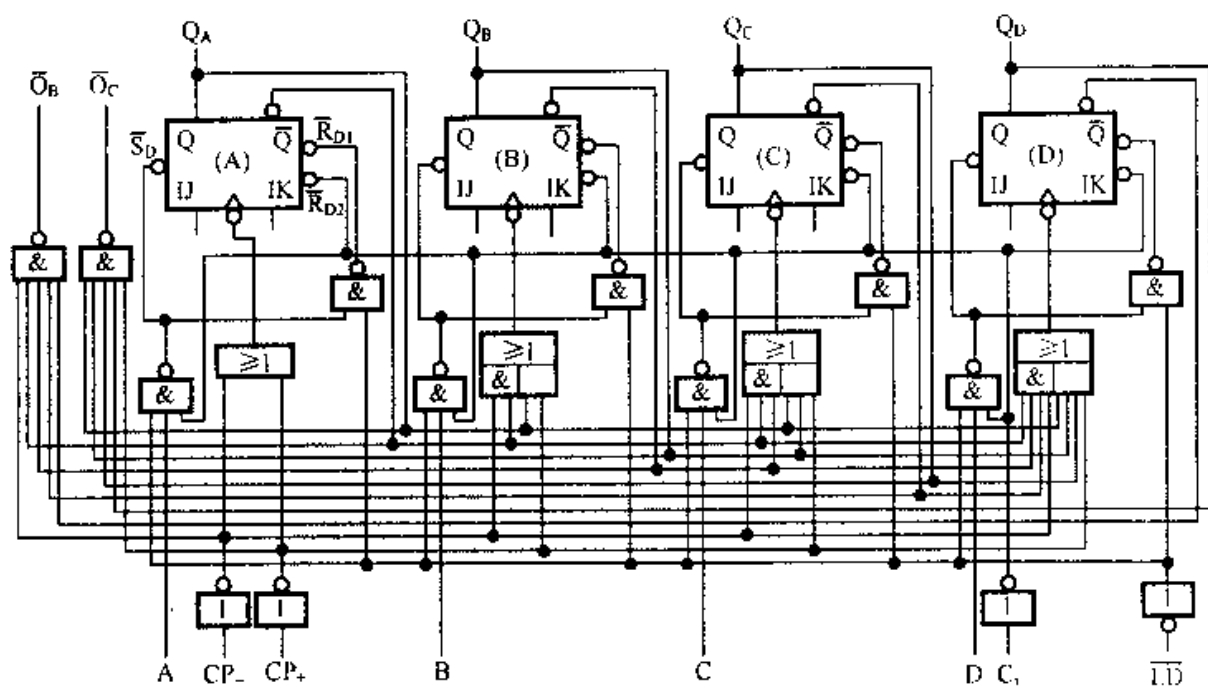
加法计数过程如下:由图 5.6.15(a)先写出各位触发器 CP 信号的逻辑函数式

$$\begin{aligned} CP_A &= \overline{CP}_+ + \overline{CP}_- \\ CP_B &= Q_A \overline{CP}_+ + \overline{Q}_A \overline{CP}_- \\ CP_C &= Q_A Q_B \overline{CP}_+ + \overline{Q}_A \overline{Q}_B \overline{CP}_- \\ CP_D &= Q_A Q_B Q_C \overline{CP}_+ + \overline{Q}_A \overline{Q}_B \overline{Q}_C \overline{CP}_- \end{aligned}$$

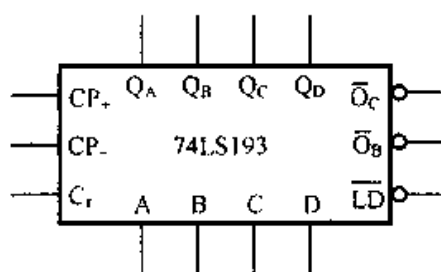
当加法计数时,应令 $CP_-=1$,上式简化为

$$\left. \begin{aligned} CP_A &= \overline{CP}_+ \\ CP_B &= Q_A \overline{CP}_+ \\ CP_C &= Q_A Q_B \overline{CP}_+ \\ CP_D &= Q_A Q_B Q_C \overline{CP}_+ \end{aligned} \right\} \quad (5.6.3)$$

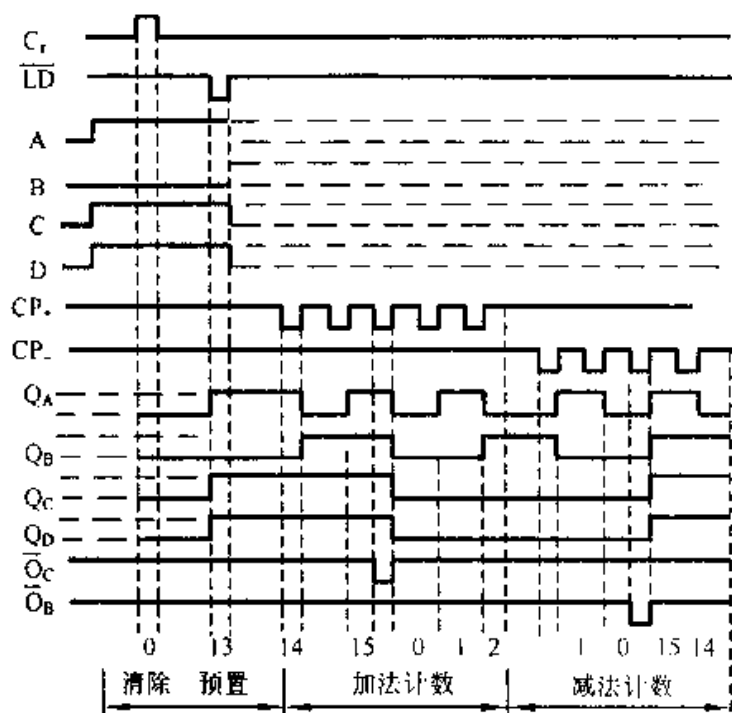
此式表明:只有在前面(低位)各级状态为 1 时,才能在 \overline{CP}_- 作用下使本级状态翻转,否则得不到触发信号,状态保持不变。但是,最低位(第一级)除外,它直接由 \overline{CP}_+ 触发,每个计数脉冲都可以使它的状态翻转。各位触发器按式(5.6.3)确定状态翻转与否,就实现了加法计数的要求。



(a)



(b)



(c)

图 5.6.15 74LS193 型 4 位同步可逆计数器

(a) 逻辑电路图 (b) 逻辑框图 (c) 波形图

当作减法计数时, 应令 $CP_+ = 1$, 分析方法同上, 各位触发器(最低位除外)只有在前面各级状态为 0 时, 在 $\overline{CP_-}$ 作用下状态翻转, 否则状态不变。这就实现了减法计数的要求。

当 CP_+ 和 CP_- 都为高电平时, 计数器处于禁止状态, 即保持状态不变。

由于该组件具有双时钟，故无必要设置方式控制信号。

该组件还设有进位输出端 \overline{Q}_C 和借位输出端 \overline{Q}_B ，其逻辑函数式为

$$\overline{Q}_C = \overline{Q_A Q_B Q_C Q_D CP_+}$$

$$\overline{Q}_B = \overline{Q_A Q_B Q_C Q_D CP_-}$$

上式表明：在做加法计数时，计数器全 1 状态输出一个负进位脉冲；在做减法计数时，全 0 状态输出一个负借位脉冲，显然 \overline{Q}_C 或 \overline{Q}_B 都与 CP 同步工作。

图 5.6.15(c) 是该组件的工作波形图，图中显示了“清 0”、预置、加法计数和减法计数 4 种工作方式，其功能如表 5.6.9 所示。

表 5.6.9 74LS193 的功能表

输 入								输 出			
C_r	\overline{LD}	CP_-	CP_+	A	B	C	D	Q_A	Q_B	Q_C	Q_D
1	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	0	0	0	0
0	0	ϕ	ϕ	A	B	C	D	A	B	C	D
0	1	1	1	ϕ	ϕ	ϕ	ϕ	加法计数			
0	1	1	1	ϕ	ϕ	ϕ	ϕ	减法计数			

(a) 当清除脉冲作用在 C_r 端时，对各位触发器同时“清 0”，因为它们的 \overline{R}_{D1} 端得到负脉冲。

(b) 当预置脉冲作用在 \overline{LD} 端时，各位数码的通道被打开，各位数码的原码和反码分别送入相应触发器的 \overline{R}_{D1} 端和 \overline{S}_D 端，完成预置操作。图中预置的数码为 DCBA=1101，即预置在状态 13。

(c) 令 $CP_- = 1$ ，计数脉冲加在 CP_+ 端，此时进行加法计数，从 13 开始，经过 14、15(进位输出)，0，1 计到 2 为止(因 CP_+ 端变为高电平)。

(d) 令 $CP_+ = 1$ ，计数脉冲加在 CP_- 端，此时进行减法计数，从 2 开始(因 $CP_- = CP_+ = 1$ 时为保持状态)，经过 1，0(借位输出)，15，计到 14 为止。

(2) 单时钟加/减计数器

单时钟 4 位二进制同步加/减计数器 74LS191 的逻辑电路与 74LS193 的电路结构类似，其差别主要有以下两点。一是组件只设置一个时钟输入端，所以必须再设置工作模式控制信号 M。当 $M=0$ 时，进行加法操作；当 $M=1$ 时，进行减法操作。二是组件只设置了一个进位/借位输出端 O_C/O_B 。加法操作： O_C/O_B 在计数器全 1 状态输出一个正进位脉冲，其宽度等于 CP 脉冲的周期。减法操作： O_C/O_B 在计数器全 0 状态输出一个正借位脉冲，其宽度等于 CP 脉冲的周期。为了扩展功能，构成多片级联应用，组件还增设了串行输出端 \overline{O}_{CR} ，在 $O_C/O_B=1$ 时， \overline{O}_{CR} 端输出一个负

脉冲, 其宽度等于 CP 脉冲的低电平维持宽度。O_C/O_B、 \overline{O}_{CR} 与 CP 脉冲之间的时序关系如图 5.6.16 (b) 所示。

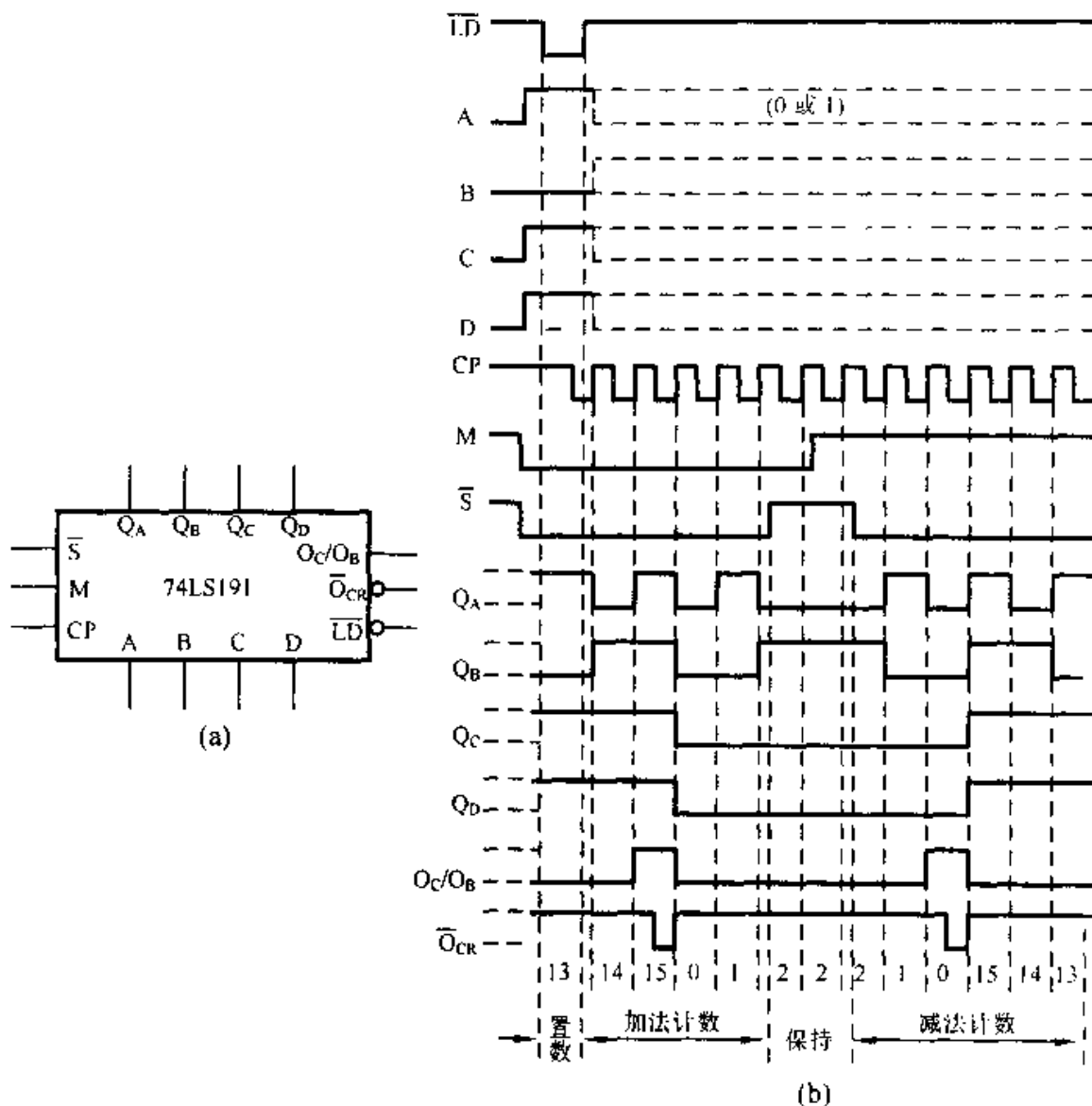


图 5.6.16 74LS191 型单时钟可逆计数器





(a) 逻辑框图 (b) 波形图

图 5.6.16 (a) 是 74LS191 逻辑框图, 其中, S 为使能端, 低电平有效; M 为模式控制端; O_C/O_B 为进位/借位输出端; \overline{O}_{CR} 为串行输出端; CP 为时钟输入端。其他输入、输出端的定义与 74LS193 相同。功能表如表 5.6.10 所示。

双时钟十进制同步加/减计数器 74LS192 与 74LS193 的电路结构基本相同, 所以它们的功能表和逻辑框图也相同, 其差别在于它们的模数不同。

单时钟十进制同步加/减计数器 74LS190 与 74LS191 的电路结构基本相同, 所以它们的功能表和逻辑符号也相同, 其差别也是它们的模数不同。

表 5.6.10 74LS191 功能表

输 入								输 出					
CP	\bar{S}	M	\overline{LD}	A	B	C	D	O_C/O_B	$\overline{O_{CH}}$	Q_A	Q_B	Q_C	Q_D
ϕ	1	ϕ	1			ϕ		0	1	保持			
ϕ	ϕ	ϕ	0	a	b	c	d	0	1	a	b	c	d
\uparrow	0	0	1			ϕ		0	1	加法计数			
\uparrow	0	1	1			ϕ		0	1	减法计数			
\uparrow	0	0	1			ϕ				1	1	1	1
\uparrow	0	1	1			ϕ				0	0	0	0

此外, 同步计数器的清零方式有两种: 即异步清零 (如 74LS161、74LS193、74LS192) 和同步清零 (如 74LS162、74LS163)。读者应注意两种清零方式的区别: 在异步清零的计数器中, 只要清除信号有效, 计数器立即被置成全 0 状态, 与 CP 信号无关; 在同步清零的计数器中, 清除信号有效之后, 只有在 CP 信号作用下才能将计数器置成全 0 状态。

在介绍 74LS161 和 74LS193 的电路中, 已讨论了异步清零的电路结构, 下面将介绍同步清零的电路结构, 如图 5.6.17 所示。它是 74LS163 电路中 A 级触发器同步清零的电路结构, 其他 3 级电路结构类似。由图可见: 若 $\overline{C_r}=0$, 则 G_1 输出为 0, G_2 输出为 1。所以, 无论 \overline{LD} 、A 的状态如何, G_3 输出为 0, 而 G_4 输出为 1, 即 $J=0$ 和 $K=1$ 。在随后出现的 CP 脉冲触发下, A 触发器被清 0, 由于清 0 操作与 CP 是同步的, 故称为同步清零方式。

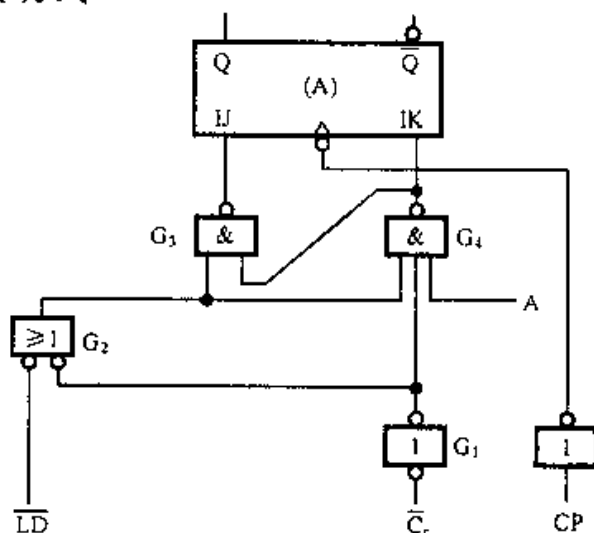


图 5.6.17 同步清零的电路结构

TTL 型同步计数器的主要产品及其特性和表 5.6.11 所示。CMOS 和 ECL 器件也有同类型产品, 读者可查阅集成电路手册。

表 5.6.11 TTL 型同步计数器

型 号	计数模式	清零方式	预置方式
74LS161	4 位二进制加法	异步 (低电平)	同步
74LS163	4 位二进制加法	同步 (┐)	同步
74LS191	单时钟 4 位二进制可逆	无	异步
74LS193	双时钟 4 位二进制可逆	异步 (高电平)	异步
74LS160	十进制加法	异步 (低电平)	同步
74LS162	十进制加法	同步 (┐)	同步
74LS192	双时钟十进制可逆	异步 (高电平)	异步
74LS190	单时钟十进制可逆	无	异步

5.6.5 采用中规模集成计数器设计任意进制计数器

尽管中规模集成计数器种类和品种都很多,但仍不能满足实际工程设计的需要。例如,在逻辑电路设计中经常使用模 6、模 12、模 24、模 60、模 2048 加法或减法计数器,但是,不可能都有其对应的产品。所以,必须采用中规模集成计数器来构成任意进制计数器。任意进制计数器的设计有两种方法,即反馈清零法和反馈置数法。反馈清零法适用于设置有清零端的集成计数器,但这种方法需增加附加控制电路,电路较复杂,而且可靠性较差;反馈置数法具有电路简单,而且可靠性高之优点。所以,下面重点介绍这种方法。

反馈置数法适用于设置有预置控制端的集成计数器。它是利用计数器的进位信号 $\overline{O_C}$ 或借位信号 $\overline{O_B}$ (74LS193 型计数器) 反馈至预置控制端 \overline{LD} , 使计数器进入预置状态, 然后开始计数的。由于集成计数器有同步预置和异步预置两种方式, 所以, 它们的预置数据 (预置状态) 是不同的, 这一点必须重视。下面结合例子来说明两种预置方式的设计方法。

(1) $M > N$ 时的设计方法

例 5.6.4 设计模 12 同步加法计数器。

解 1 选用具有异步预置功能的 4 位二进制可逆计数器 74LS193 ($M=16$), 电路连接如图 5.6.18 (a) 所示。计数器的工作状态为状态 3 至状态 14, 当计数器进入到 1111 状态时, 立即输出进位负脉冲, 即 $\overline{O_C}$ 端由 1 变为 0, 预置状态 0011 马上进入计数器, 因而 1111 状态仅存在很短时间, 所以它不是工作状态。

这是一种固定的结构, 只要改变预置数据 D_n , 就可以实现不同模数的计数器。对于采用异步预置方式的计数器, 其预置数据为

$$D_n = 15 - N$$

解 2 选用具有同步预置功能的 4 位二进制加法计数器 74LS161, 电路连接如图 5.6.18 (b) 所示。计数器的工作状态为状态 4 至状态 15。当计数器进入到 1111 状态时, O_C 端输出正进位脉冲, 经非门反相后在 \overline{LD} 端得到负脉冲。但是, 由于采用同步预置方式, 必须等到下一个 CP 脉冲出现时才能使预置数据 0100 进入计数器, 所以, 1111 是工作状态。预置数据为

$$D_5 = 16 - N$$

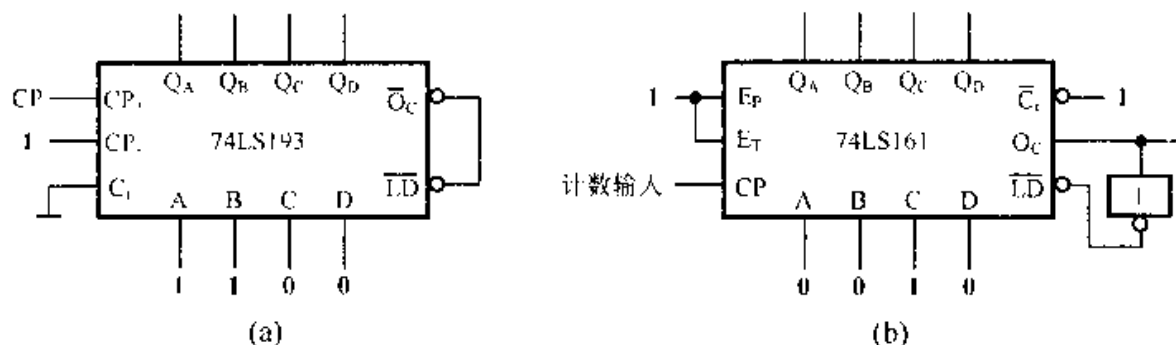


图 5.6.18 模 12 计数器的连接图

(a) 连接图之一 (b) 连接图之二

例 5.6.5 设计模 12 同步减法计数器。

解 选用 4 位二进制同步可逆计数器 74LS193, 电路如图 5.6.19 所示。

CP₊接 1, 计数脉冲从 CP 加入以实现减法操作。借位输出 \overline{O}_B 与 \overline{LD} 连接, 预置数据为 1100。计数器的工作状态为状态 12 至状态 1。

当计数器减到全 0 状态时, \overline{O}_B 端由 1 变为 0,

因此, 立即使预置数据进入计数器, 所以, 状态 0 只是瞬间出现, 不是工作状态。

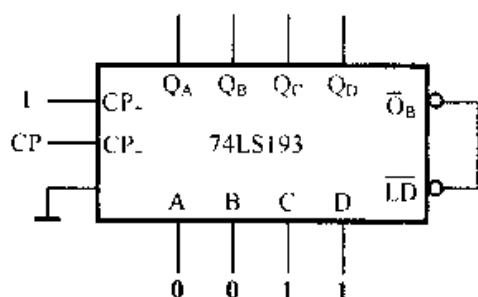


图 5.6.19 例 5.6.7 连接图

(2) $M < N$ 时的设计方法

例 5.6.6 设计模 462 同步加法计数器。

解 选用具有同步预置功能的 4 位二进制加法计数器 74LS161。图 5.6.20 是构成模 462 加法计数器的 3 种不同连接方法的连接图。

图 5.6.20 (a) 所示电路是用 3 块组件构成全并行进位结构, 单元 1、2、3 分别完成模 6、模 7、模 11 计数, 为此, 各单元输入数据分别为 DCBA=1010 (单元 1)、DCBA=1001 (单元 2)、DCBA=0101 (单元 3), 各单元均另加非门使进位输出反传送到各自预置端 \overline{LD} 。为了实现各单元同步计数, 各单元的 E_P 和 E_T 端接法不同。单元 1 的 E_P 和 E_T 端都接高电平, 即令其处于计数运行状态; 单元 1 的 O_C 端接到以后各级的 E_P 端; 单元 2 的 O_C 端接到单元 3 的使能端 E_T ; 单元 2 的 E_T 端接高电平。如此连接就保证了只有在低位单元进位端都为高电平时, 相邻单元才能计数。

图 5.6.20 (b) 所示电路是用 3 块组件构成串-并行进位结构, 即单元内部是并

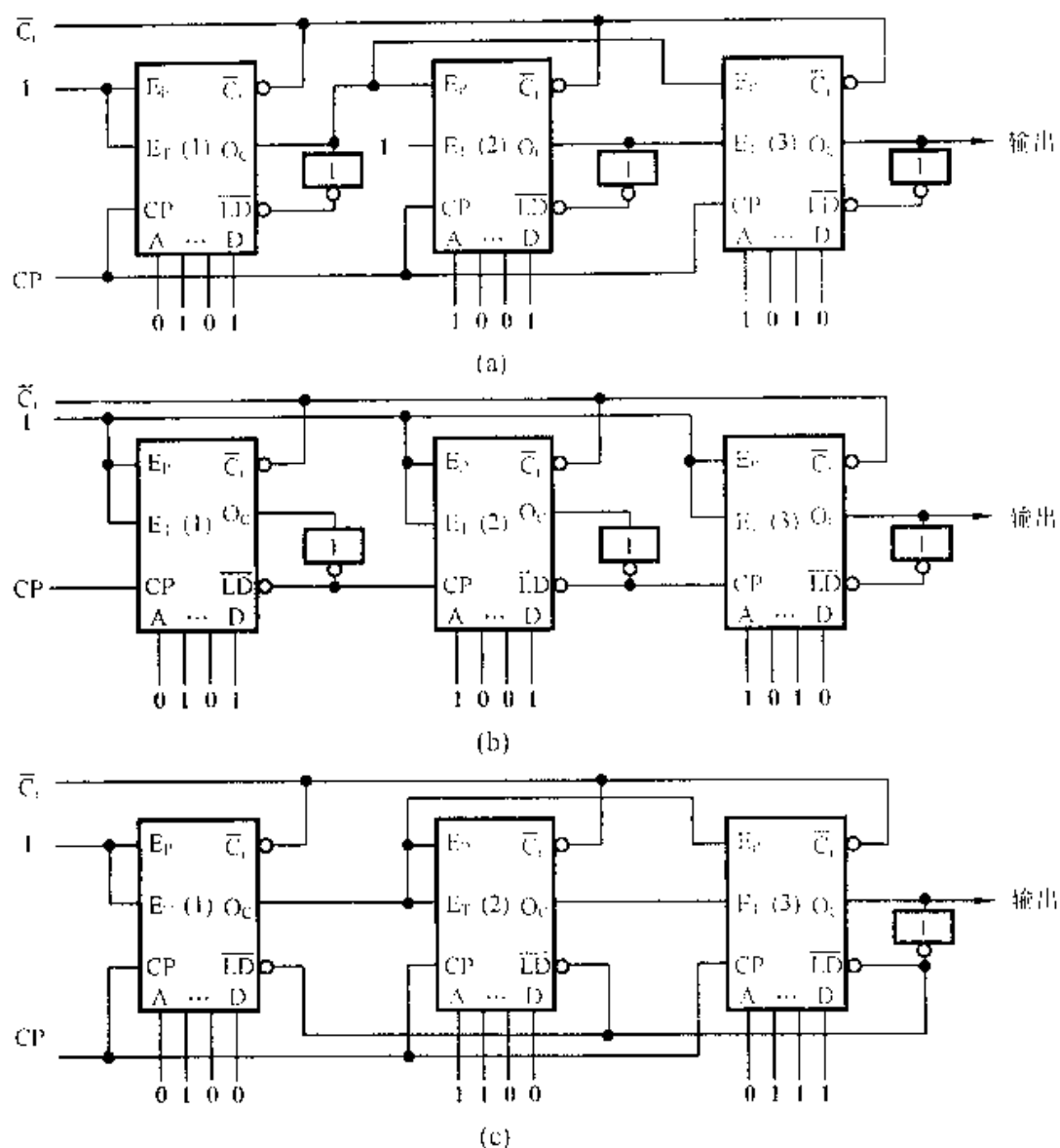


图 5.6.20 模 462 计数器的连接图

(a) 连接图之一 (b) 连接图之二 (c) 连接图之三

行进位，单元之间是串行进位，计数脉冲从单元 1 加入，单元 2 和 3 的计数脉冲由前一级进位信号提供，各单元分别接成模 6、模 7、模 11 计数器。所有单元的使能端 E_P 和 E_T 都并接到公用线上，并令它接高电平；所有单元的清除端都接到公用线 \overline{C}_i 上，利用 \overline{C}_i 上的负脉冲同时“清 0”。单元 3 的进位输出就是整个计数器的进位输出信号。用这种连接方法构成的加法计数器的工作速度较慢。

图 5.6.20(c)所示电路是用 3 块组件构成全并行进位结构，令起始状态为 $M=4096-462=3634$ ，即选择工作状态为 3634~4095（共 462 个状态），由于

$$3634=2^{11}+2^{10}+2^9+2^5+2^4+2^1 \quad (5.6.3)$$

上式表示预置端“权”为 2^{11} 、 2^{10} 、 2^9 、 2^5 、 2^4 、 2^1 等端接高电平，其余各端接低电平，即各端输入数据为 DCBA=0010(单元 1)，DCBA=0011(单元 2)，DCBA=1110

(单元 3)。增加一个非门将单元 3 的进位输出反传送到所有单元的预置端 \overline{LD} 。当计数满量后 (即全 1 状态), 各单元 \overline{LD} 端得到低电平, 在下一个 CP 脉冲作用后, 计数器回到起始状态 3634, 实现模 462 计数功能。

5.7 序列信号发生器

序列信号发生器的功能是产生一组或多组二进制序列信号。例如, 在通信设备中经常需要产生一组规则码用来调机或检修设备; 又如通信系统中的同步就需要产生一组特定的二进制序列信号来表示一组信息的开始或终止。对于一个复杂的通信系统, 在一个工作周期内要完成各种功能, 其控制部分经常要产生一系列的序列信号, 它也是二进制序列信号。产生二进制序列信号的逻辑电路称为序列信号发生器。序列信号发生器在雷达、通信、遥控与遥测、测量以及无线电仪表等领域有广泛应用。

序列信号发生器通常可以在移存器或计数器的基础上构成, 前者通常只产生一组序列信号, 后者可以产生一组或多组序列信号, 下面分别讨论它们的设计方法。

1. 移存器型序列信号发生器

移存器型序列信号发生器的框图如图 5.7.1 所示。反馈电路的作用是检测移存器的现态, 产生 0 或 1 的输出, 输至移存器以便得到相应的次态, 使电路输出给定的序列信号。在序列信号的每个循环周期中所含有的码元位数称为循环长度 M (或序列长度), 也称为序列周期。

移存器型序列信号发生器的设计步骤: 确定移存器的级数; 列出状态转换真值表; 设计反馈电路。

例 5.7.1 设计产生序列信号 00011101 的移存器型序列信号发生器。

解 (1) 确定移存器的级数 n : 移存器的级数按 $2^n \geq M$ 来确定。对于本例, $M=8$, 所以 $n \geq 3$ 。首先应考虑选用 $n=3$, 并检查是否满足设计要求。检查方法是按输出序列次序每次取 3 位得到 8 种状态, 如图 5.7.2 所示。第一次取左 3 位为起始状态;

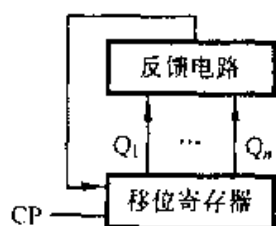


图 5.7.1 移存器型序列信号发生器框图

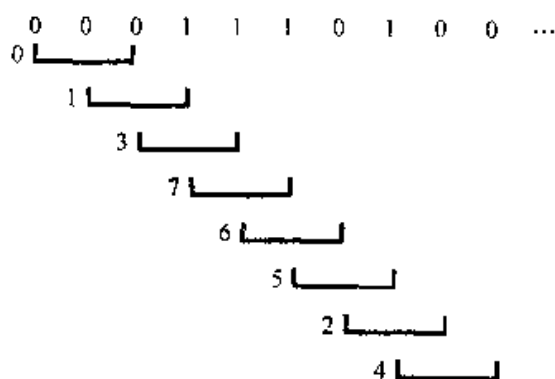


图 5.7.2 例 5.7.1 的工作状态

第二次取左起第二位开始的3位；如此继续下去，直到从序列的最末位开始的3位取完为止。由此可画出状态图 5.7.3。由于图中每一个状态都是独立的，即状态图中没有重复状态，所以 $n=3$ 满足设计要求。

(2) 列状态转换真值表：由图 5.7.3 列出状态转换真值表，如表 5.7.1 所示。为了确定反馈函数表达式，在表中填上第一级触发器的激励值，如选用 D 触发器来实现电路，其激励值列入表 5.7.1 中最右一列。

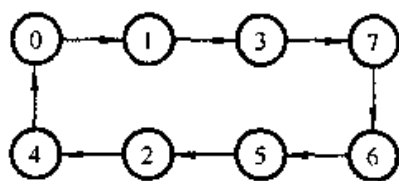


图 5.7.3 例 5.7.1 的状态图

表 5.7.1 状态转换真值表和激励表

Q_1	Q_2	Q_3	Q_3^{n+1}	Q_2^{n+1}	Q_1^{n+1}	D_1
0	0	0	0	0	1	1
0	0	1	0	1	1	1
0	1	1	1	1	1	1
1	1	1	1	1	0	0
1	1	0	1	0	1	1
1	0	1	0	1	0	0
0	1	0	1	0	0	0
1	0	0	0	0	0	0

(3) 确定 D_1 的控制方程：由表 5.7.1 画出 D_1 的卡诺图 5.7.4(a)，由此可得

$$D_1 = \bar{Q}_3 \bar{Q}_2 + \bar{Q}_3 Q_1 + Q_3 Q_2 \bar{Q}_1$$

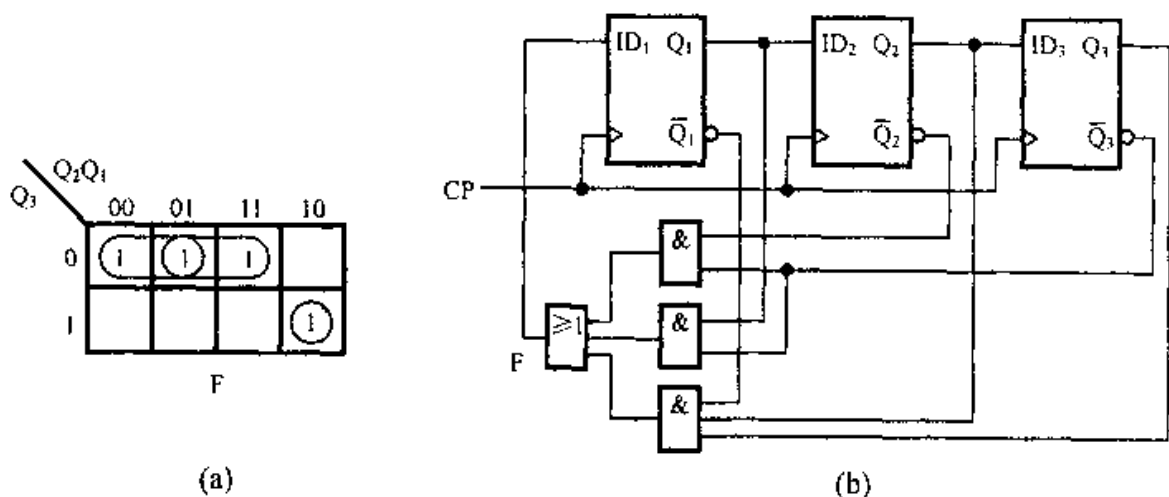


图 5.7.4 移位寄存器型序列信号发生器

(a) D_1 的卡诺图 (b) 逻辑图

(4) 画逻辑图：先由 D 触发器构成 3 级移存器，然后由 D_1 的控制方程画反馈电路，逻辑图示于图 5.7.4(b)。序列信号可以从任一触发器的 Q 端输出。

例 5.7.2 用移存器型序列信号发生器产生循环序列 11110000，试确定移存器的级数。

解 依据 $2^n \geq 3$, 首先选取 $n=3$, 按照依次序每次取 3 位构成电路状态的方法, 可得状态图, 见图 5.7.5(a), 但是, 图中 000 和 111 均为重复状态, 它们各自都有两种次态。例如, 状态 111 的次态为 110 和 111, 这意味着移寄存器在同一个状态下, 要求产生两个不同的反馈信号, 显然反馈电路不能满足这种要求, 所以, $n=3$ 不能满足设计要求。再选取 $n=4$, 则按照依次序每次取 4 位的方法构成状态图, 见图 5.7.5(b)。由图可见: 增加一级后, 消除了重复状态, 故可依据状态图 5.7.5(b) 来设计电路(余下步骤同例 5.7.1), 所以, 移寄存器的级数选取 4 级最佳。

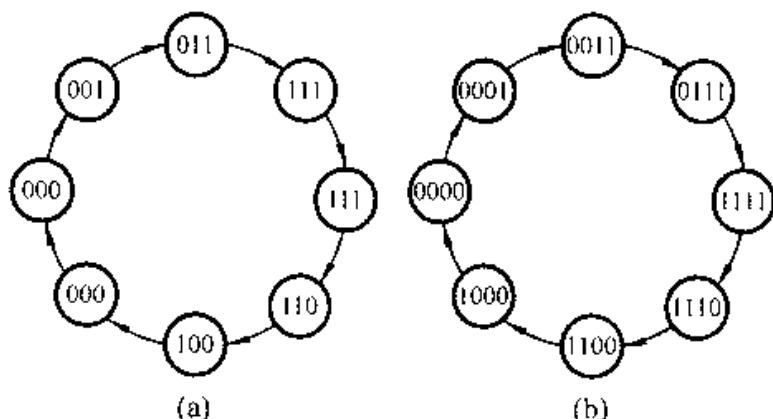


图 5.7.5 例 5.7.2 的状态图

(a) $n=3$ (b) $n=4$

顺便指出: 4 级移寄存器可构成 16 种电路状态, 所以, 对未使用状态要进行状态检查。

2. 计数器型序列信号发生器

由计数器构成的序列信号发生器框图如图 5.7.6(a)所示, 它是由周期为 M 的计数器和组合电路两部分组成。组合电路的输出序列可以是周期为 M 的一组, 也可以是周期为 M 的多组序列信号。前者对应的是一个输出端的组合电路, 后者对应的是多个输出端的组合电路。

例 5.7.3 设计产生序列信号为 00100011101 的计数器型序列发生器。

解 因为序列长度 $M=11$, 所以, 先用 74LS161 构成模 11 加法计数器, 然后依据序列信号列出组合电路的真值表, 见表 5.7.2, 由表画出 F 卡诺图, 见图 5.7.6(b)。由图可得

$$F = Q_B Q_A + Q_D Q_C \bar{Q}_B$$

由上式画逻辑图, 如图 5.7.6(c)所示。

用同样的方法可以设计组合电路输出两组或多组循环长度相同的序列信号, 但是, 这类电路存在组合电路的功能冒险, 并且无法采取设计措施消除, 因此, 必须采用“取样”措施消除功能冒险, 一般都是利用时钟脉冲作为“取样”脉冲, 如图 5.7.6(c)中虚线所示, 这时, 必须注意计数器的状态更新应是发生在上跳沿, 所以, CP 脉冲经

非门反相后作为取样脉冲。电路输出的序列信号是脉冲信号，若需要电位信号，则可增加一级 D 触发器，如图 5.7.6(d)所示。从 Q 端输出的序列 F' 是电位信号。

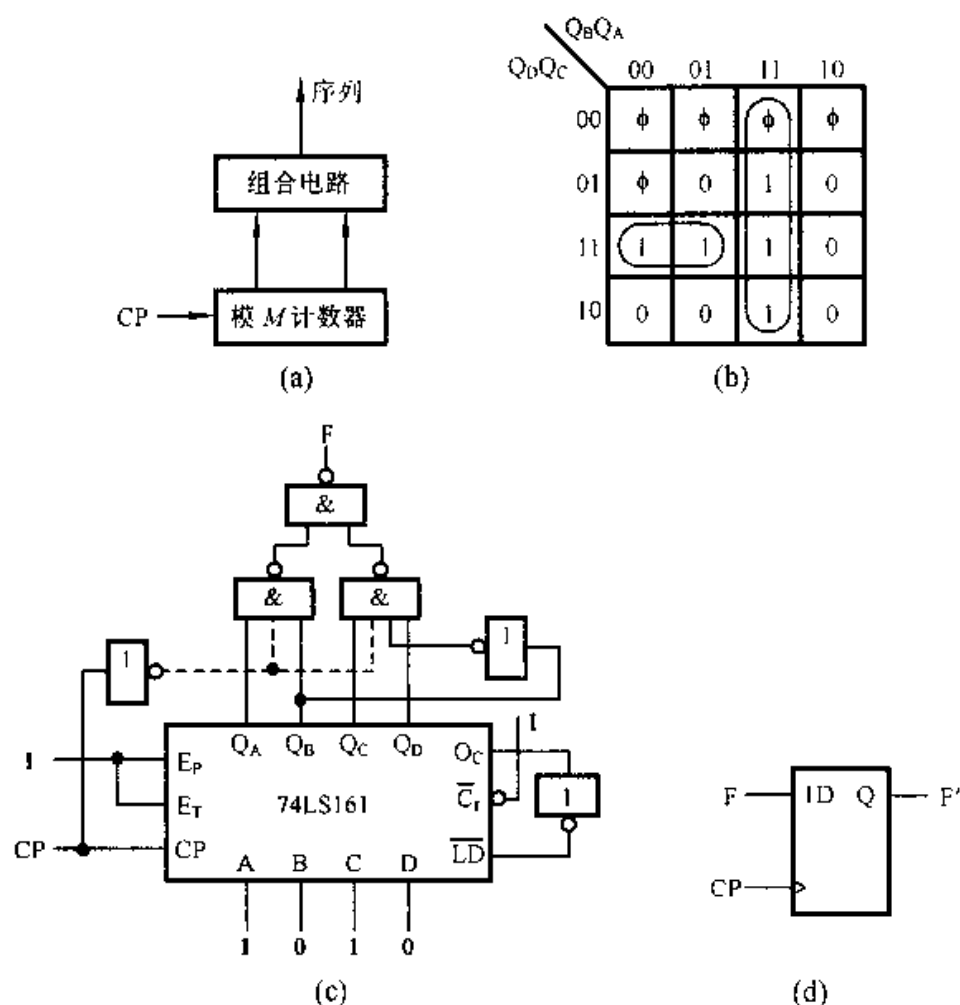


图 5.7.6 计数器型序列发生器

(a) 框图 (b) 卡诺图 (c) 逻辑电路图 (d) 电位信号输出图

表 5.7.2 例 5.7.3 真值表

Q_D	Q_C	Q_B	Q_A	F
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

小 结

时序逻辑电路与组合逻辑电路在功能上的不同点是,时序逻辑电路在任一时刻的输出,不仅取决于该时刻的输入,而且还依赖于过去的输入。因此,在其电路结构上包含有组合逻辑电路和存储电路两部分,并且从组合逻辑电路输出经存储电路回到组合逻辑电路的回路间,至少存在着一条反馈支路。

时序电路分为同步时序电路和异步时序电路两大类。在同步时序电路中,有一个统一的时钟脉冲,使所有的触发器同步工作;而在异步时序电路中,存储电路状态的改变是与时钟异步的。同步时序电路的应用比异步时序电路更为广泛。

同步时序电路分析,就是对给定的电路,确定它的逻辑功能。通常,是用状态转换表或状态转换图来表示其逻辑功能。

同步时序电路设计,就是对给定的逻辑功能,画出其相应的电路。其设计过程一般分以下几步:原始状态表或状态图的拟定,状态简化,状态分配,确定控制输入和画逻辑电路图。需要指出,其中原始状态表或状态图的拟定,是整个设计工作的关键,难度也较大。

本章介绍了常用的同步和异步时序逻辑部件的设计方法,还较详细地介绍了常用的 MSI 时序逻辑部件的设计与使用方法。其中,各种控制端及选通端的逻辑功能、电平要求、功能扩展方法是本章重点内容。

在最常用的计数器、移位寄存器等通用性强的时序组件中,状态数的预置方式有异步预置和同步预置两类。在异步预置方式中,只要预置信号有效,则组件立刻进入预置状态,为此,只要预置数据是加至触发器的 \overline{R}_D 、 \overline{S}_D 端便可完成预置功能(参见图 5.6.15);在同步预置方式中,则要在预置信号有效条件下,在随后出现的时钟脉冲协助下才能完成预置功能,为此,预置数据是加至触发器的 J、K 端,等待随后出现时钟脉冲作用后才能完成预置功能(参见图 5.6.13 和图 5.5.3)。

这些组件的清零方式也有两种方式,即异步清零和同步清零。在异步清零方式中,只要是清除信号有效,组件即被清零;在同步清零方式中,在清除信号有效之后,由随后出现的时钟脉冲作用才完成清零功能(参见图 5.6.17)。

思考题和习题

- 5.1 组合电路与时序电路有何区别?试举例说明。
- 5.2 为什么要对逻辑电路进行分析?怎样对时序电路进行分析?
- 5.3 试举例说明时序电路的设计方法。

- 5.4 试举例说明异步串行计数器的设计方法。请用灵活设计法设计模 12 异步串行计数器(指定采用 JK 触发器)。
- 5.5 在使用 MSI 组件时应注意什么问题?在进行功能扩展时又应该注意什么问题?
- 5.6 同步预置型的同步二进制计数器与异步预置型的同步二进制计数器在电路结构、预置数据方式、预置数据等方面有何不同?
- 5.7 MSI 移位寄存器在数字电路或系统中有哪些应用,你能举出几例来吗?
- 5.8 状态转换真值表如表 P5.1 所示,试作出状态转换表和状态转换图。
- 5.9 已知同步时序电路状态转换表如表 P5.2 所示,试作出它的状态图。

表 P5.1

即刻输入	原状态	即刻控制输入	新状态	即刻输出
X	Q	D	Q_{n+1}	Z
0	0	0	0	0
0	1	1	1	0
1	0	1	1	0
1	1	0	0	1

表 P5.2

Y	Y_{n+1}		Z
	X=0	X=1	
A	C	B	0
B	C	D	0
C	B	B	0
D	D	A	1

- 5.10 分析图 P5.1 所示时序电路,作出其状态转换表和状态转换图。
- 5.11 由状态图 P5.2 作出状态表,并用 D 触发器实现。

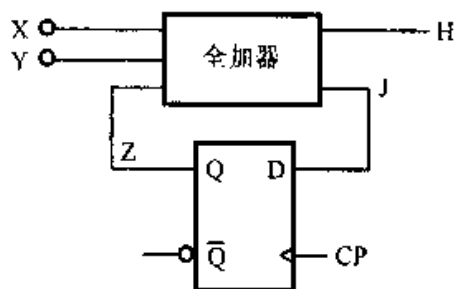


图 P5.1

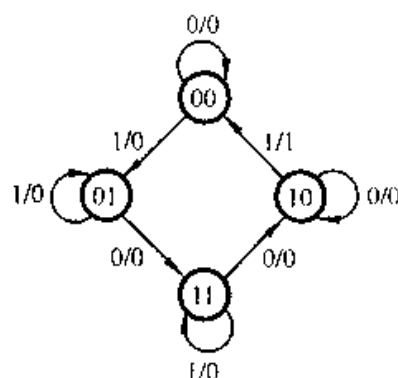


图 P5.2

- 5.12 下面两个具有一个输入 X, 一个输出 Z 的同步时序电路, 请拟定原始状态图。
- (1) 电路连续不停地工作, 凡是遇到连续 4 位输入为 1101 时, 输出 Z 便为 1, 其他情况下, 输出 Z 为 0。
 - (2) 每输入 4 位码之后, 电路自动恢复到初态, 在这 4 位输入码中, 当且仅当为 1101 时, 输出 Z 为 1, 否则 Z 为 0。
- 5.13 拟定 101 序列检测器的状态图。
- (1) 101 序列可以重叠, 如输入序列 010101101, 输出序列 000101001。

(2) 101 序列不可重叠, 如输入序列 010101101, 输出序列 000100001。

5.14 设计一个同步时序电路, 只有在连续两个或两个以上时钟作用期间两个输入信号 X_1 和 X_2 都一致时, 输出信号才为 1, 其余情况下输出为 0。

5.15 简化原始状态表 P5.3(a)和(b), 列出最小化状态表。

5.16 用 74LS193 分别构成模 13 的加法计数器和模 9 的减法计数器。

5.17 用 74LS161 分别构成模 13 的加法计数器和模 539 的加法计数器。

5.18 用两只 JK 触发器和门器件组成时序电路, 各触发器的控制输入函数和输出函数分别为

$$\begin{cases} J_1 = Q_2 X \\ K_1 = Q_2 X \end{cases} \quad \begin{cases} J_2 = Q_1 X \\ K_2 = Q_1 X \end{cases} \quad Z = \bar{Q}_1 \oplus X$$

5.19 用 JK 触发器设计一个时序电路, 使其满足下列状态方程:

$$\begin{aligned} Q_1^{n+1} &= XQ_1Q_2 + Y\bar{Q}_1Q_3 + XY \\ Q_2^{n+1} &= XQ_1Q_3 + \bar{Y}Q_2\bar{Q}_3 \\ Q_3^{n+1} &= \bar{X}Q_2 + YQ_1\bar{Q}_2 \end{aligned}$$

5.20 设计一个同步时序电路, 它有两个输入 X_1 和 X_2 , 一个输出 Z , 只有当 X_1 输入 3 个 1(或 3 个以上的 1)后, X_2 再输入一个 1 时, Z 才为 1, 而在同一时刻两个输入不能同时为 1。(提示: 这里 X_1 输入 3 个 1 可以不连续, 只要其间没有 X_2 为 1 插入即可, 一旦 Z 为 1 则电路回到原始状态)。

5.21 在一数据传送系统中, 信号发送结束时, 采用连续发 5 个 1 作为结束信号, 当接收端检测装置接收到 5 个连续 1 时, 要求输出 $Z=1$ 作为回答, 其他情况 $Z=0$ 。试设计这个回答信号发生器。

5.22 设计一个计数器型序列发生器, 要求产生两组序列信号分别为 101100110111 和 111000111000。输出序列信号应无“毛刺”, 并要求是电位信号。

5.23 设计移存型序列信号发生器, 要求产生序列信号为 11110000。

5.24 试用 3 级 D 触发器设计移存型序列信号发生器, 要求产生序列为 00011101。

5.25 采用 4 级 D 触发器设计移存型序列发生器, 其产生序列与预置初态满足表 P5.4。

表 P5.3(a)

Y	X	
	0	1
A	B/0	A/1
B	C/0	A/C
C	C/0	B/0
D	F/0	D/1
E	C/0	D/0

表 P5.3(b)

Y	X	
	0	1
A	A/0	E/1
B	E/1	C/0
C	A/1	D/1
D	F/0	G/1
E	B/1	C/0
F	F/0	E/1
G	A/1	D/1

表 P5.4

Q_3	Q_2	Q_1	Q_0	输出序列
0	1	1	0	011, 011, ...
1	1	1	1	111100, ...
1	0	0	0	100010, ...
0	0	0	0	全 0 序列

5.26 设计一个可控同步计数器, 要求能实现维持原状态; 八进制计数器; 十进制计数器等 3 种不同操作。

5.27 设计一个可控同步计数器, 要求

- (1) 在 $M=0$ 时, 在时钟作用下按加 1 顺序($0 \rightarrow 1 \rightarrow 2 \rightarrow \dots$)计数, 并实现模 5 计数器;
- (2) 在 $M=1$ 时, 在时钟作用下按加 2 顺序($0 \rightarrow 2 \rightarrow 4 \rightarrow 1 \dots$)计数, 也实现模 5 计数器。

5.28 图 P5.3 为一个由移存器构成的脉冲序列产生器的逻辑电路图。

- (1) 设初态为 $Q_C Q_B Q_A = 001$, 画出状态转换图;
- (2) 此电路具有自启动功能否? 若否, 如何使它变成自启动电路?

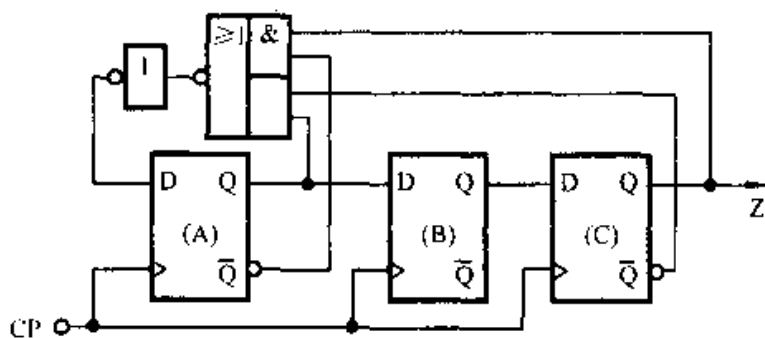


图 P5.3

5.29 试用 74LS195 或 74LS194 构成 16 位右移计数器。

5.30 设计一个可控同步计数器, M_1 和 M_2 为控制信号。要求:

- (1) $M_1 M_2 = 00$ 时, 维持操作;
- (2) $M_1 M_2 = 01$ 时, 模 2 计数;
- (3) $M_1 M_2 = 10$ 时, 模 4 计数;
- (4) $M_1 M_2 = 11$ 时, 模 8 计数。

5.31 设计模 5 同步计数器, 要求在时钟信号为方波时, 输出也是方波。

5.32 已知某一时序电路的状态方程为

$$\begin{aligned} Q_4^{n+1} &= (Q_2 \oplus Q_1)X + (Q_2 \odot Q_1)\bar{X} \\ Q_3^{n+1} &= Q_4 \\ Q_2^{n+1} &= Q_3 \\ Q_1^{n+1} &= Q_2 \end{aligned}$$

设起始状态为 $Q_4 Q_3 Q_2 Q_1 = 0001$, 请列出:

- (1) 当 $X=0$ 时, Q_1 的输出序列;
- (2) 当 $X=1$ 时, Q_3 的输出序列。

5.33 输入为串行电位码, 当输入连续 4 位码之中 1 的个数大于或等于 3 时, 输出为 1, 试设计该时序电路(器件任选)。

5.34 某数字通信系统中使用的帧同步码为 00010011(左位在前), 试设计一个帧同步码的检测电路。当收到同步码时, 电路输出为 1; 反之输出为 0(器件任选)。

5.35 设提供两片 MSI 十进制计数器(见图 P5.4), 试构成 25 进制计数器。 O_C 是低位十进制计数器的进位输出端。问反馈控制 X、Y、Z 应分别接何处。

5.36 试用 MSI 四位移存器 74LS195(见图 5.5.3 和表 5.5.1)组件构成一个扭环计数器, 画出逻辑电路图, 并列出现态表。

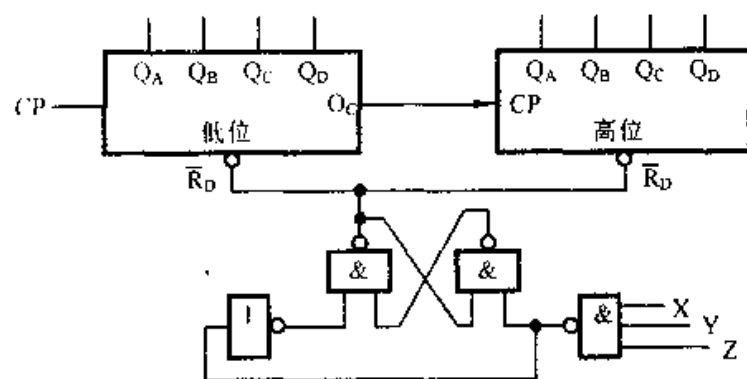


图 P5.4

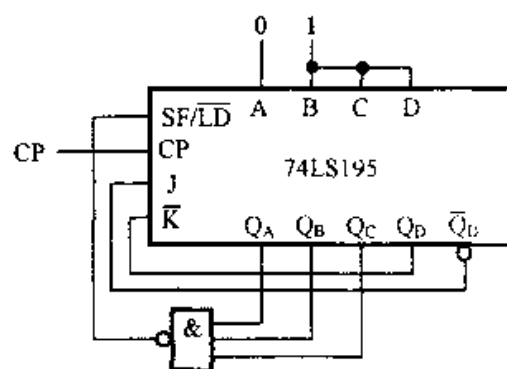


图 P5.5

5.37 试分析图 P5.5 所示电路的逻辑功能，画出状态图。设电路初态为 $Q_D Q_C Q_B Q_A = 1110$ 。

5.38 试分析图 P5.6(a)和(b)所示电路的逻辑功能，画出状态图。设初态为 0 状态。

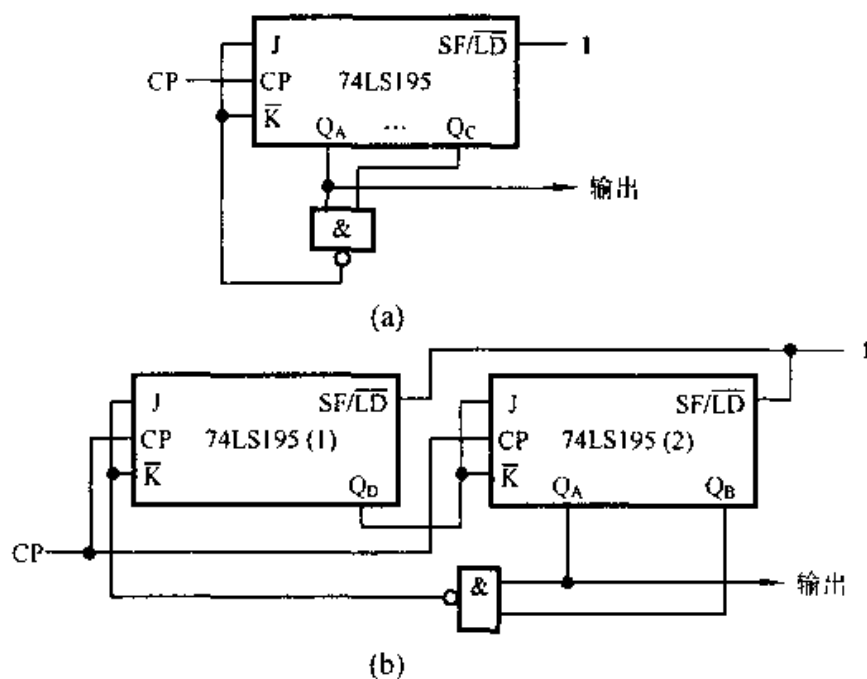


图 P5.6

5.39 试分析图 P5.7(a)和(b)所示同步时序电路。

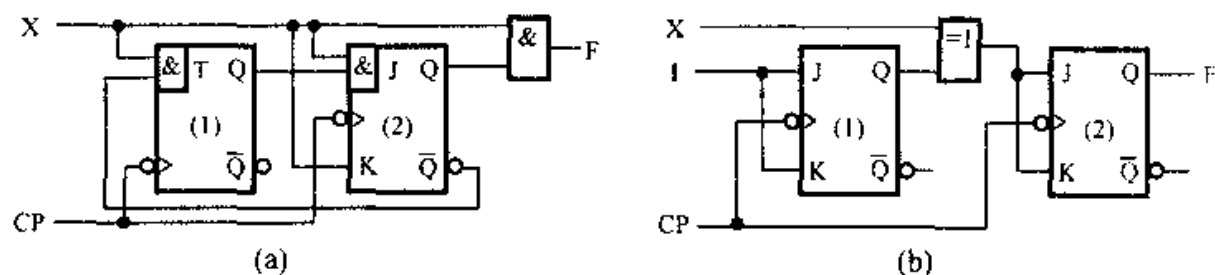


图 P5.7

5.40 试分析图 P5.8 所示同步时序电路。

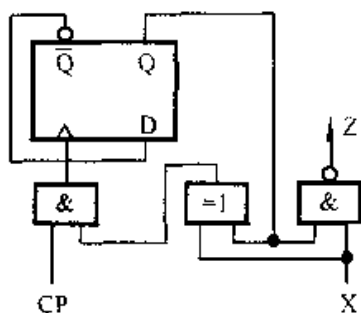


图 P5.8

- 5.41 设有两个由 D 触发器构成的 3 位寄存器 A 和 B，它们除受 CP 脉冲控制外，还有两个控制信号 C_2 和 C_1 ；3 个外输入 $[X] = [X_3X_2X_1]$ 。要求电路实现表 P5.5 所示功能，试画出逻辑电路图。

表 P5.5

C_2	C_1	操作模式
0	0	不动作
0	1	A 的内容输入 B；X 值输入 A
1	0	X 值输入 A 中；B 内容不变
1	1	A 的内容输入 B；A 全清 0

- 5.42 图 P5.9 为产生宽度等于 14 个 CP 周期的脉冲产生器，试选择合适的 MSI 计数器并设计控制电路，并画出电路的工作波形。

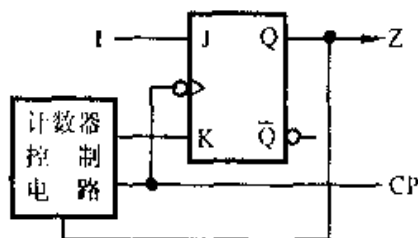


图 P5.9

- 5.43 设计一个脉冲分配器，要求具有 4 路输出(见图 P5.10(a))，且满足图 P5.10(b)所示输入与输出关系。

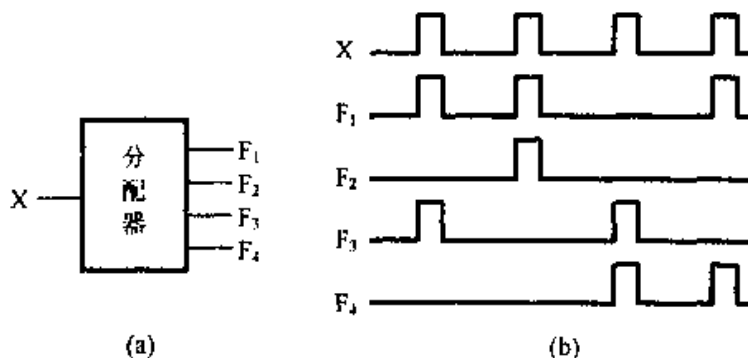


图 P5.10

第 6 章 硬件描述语言

内容提要 本章以 ABEL-HDL 语言为主线, 介绍硬件描述语言的特点、语法规则、程序结构, 简述在 ISP-Synario 软件平台上的开发过程, 最后举例说明硬件描述语言在组合逻辑和时序逻辑电路分析和设计中的应用。

硬件描述语言是现代 EDA 技术的重要组成部分。硬件设计者以计算机为工具, 在 EDA 软件平台上, 用硬件描述语言完成设计。尽管设计目标是硬件系统, 但借助于 EDA 开发平台, 就可以使整个设计 (包括仿真和修改) 过程如同软件设计一样方便快捷。因此, 掌握 EDA 技术已成为对硬件工程师的基本要求。

6.1 硬件描述语言概述

6.1.1 现代 EDA 技术简介

EDA 是 Electronics Design Automation (电子设计自动化) 的缩写, EDA 设计平台是伴随着集成电路和计算机技术的飞速发展而产生的一种高级、快速、有效的电子设计自动化工具, 是一个集数据库、图形学、拓扑、逻辑、计算数学、优化理论等多学科最新成果研制而成的计算机辅助设计通用软件包。

传统的电子系统设计, 不论是模拟系统、数字系统还是混合系统设计, 基本思路都是所谓“自底向上 (Bottom-up)”设计, 即由若干器件构成电路板并调试单板, 再由各单板连成系统, 最后系统联调, 如图 6.1.1(a) 所示。这是一种积木式的设计思路。对数字系统来说, “积木”即是各种中大规模功能模块器件 (运算部件、编码器、多路器、计数器、存储器等), 以及小规模门器件等。这样设计出的电子系统, 所用电路元件的种类多、数量多, 板上连线多, 体积大、功耗大, 因而系统的后期调试困难, 设计隐患难以发现, 可靠性也难以保证。

现代电路系统设计则是“自顶向下 (Top-down)”的。即是由系统级设计开始, 在顶层进行系统功能方框图的划分, 再对各功能方框进行功能验证和仿真, 如图 6.1.1(b) 所示。这种仿真可以是系统级的 (纯算法性质), 以研究系统的功能行为为目的; 也可以是功能级的, 以研究系统各部分方框的相互作用和对系统总性能的影响; 还可以是电路级的, 研究具体的电路 (门级逻辑或电路元器件) 实现形式对系统总体

性能的贡献与限制；最后仿真到物理级上，直接研究印制电路板或专用可编程集成电路的工程实现性能。这种设计思想，能使所设计的电子系统由概念到具体实现逐步深入，可以在设计的早期阶段发现各设计层次的错误、缺陷和限制，并评估其对系统功能的影响。显而易见，这种以“概念驱动工程”的设计方法，预见性强，起点高，效率高，系统的兼容性和可靠性得到了充分保证。

“自顶向下”设计理念的实际应用，是以 EDA 技术和可编程器件的高速发展并且相互促进与结合为基础的。以数字系统来看，传统的逻辑器件功能单一，输入/输出引脚固定，更不用说重新定义其芯片功能了，这就决定了其设计只能是部件级的实现。现代的 CPLD/FPGA 等各类复杂可编程器件，单片集成度高，内部可编程单元功能强，互连灵活可变，大部分引脚的功能和特性均可由用户定义，因此，一个器件可以设计为具有指定功能的专用芯片，或者专用系统，或者片上系统 SOP (System On Chip)。今天，还可以达到在系统板上直接重新编程和配置（即在系统可编程 ISP）。上述功能的实现，正是靠 EDA 工具达到的

EDA 技术伴随着计算机技术、集成电路技术、电子系统设计方法的发展，经历了以下 3 个发展阶段。

第一阶段——计算机辅助设计 CAD(Computer Assist Design)，即早期的电路 CAD。这一阶段的特点是只有一些单独的工具软件，受当时计算机工作平台的制约，第一代 EDA 工具能支持的设计工作有限且性能比较差。例如，当时国内广泛使用的 Tango CAD 软件等。后期发展为成套的软件工具包。通过这些软件包完成产品开发的设计、分析、生产、测试等各项工作。

第二阶段——计算机辅助工程设计 CAED(Computer Assist Engineering Design)，是具有一定的计算机仿真和自动综合布局、布线能力的 EDA 工具。这一阶段的特点是软件平台中各类设计工具已经齐全，如各种元件库、层次设计和原理图输入、编译连接、仿真模拟、版图自动化等，并采用了统一的数据管理。但是，大部分 EDA 工具仍然从原理图出发，以具体器件为中心，设计思想的实现受到限制，一般仅能作规模不大的部件或电子系统仿真，不能适应复杂电子系统设计的要求。

第三阶段——电子系统设计自动化 EDA(Electronic-System Design Automation)，产生了真正意义上的功能强大的 EDA 设计平台。这一阶段的特点是系统级的高层描述

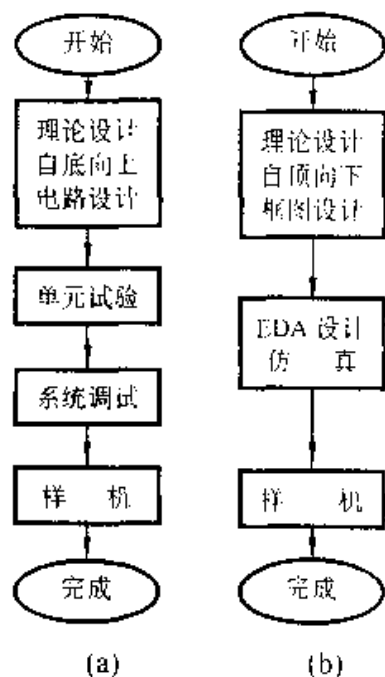


图 6.1.1 传统与现代电路系统设计流程

(a) “自底向上”设计法框图

(b) “自顶向下”设计法框图

与硬件的高级语言描述相结合,全面实现“自顶向下”的现代电子系统设计思想。硬件描述语言的问世,突破了原理图描述的限制。高层综合方法可把系统的行为描述转换到底层硬件实现,使设计人员的注意力集中于系统行为设计和算法设计上。以系统级设计为核心的 EDA 设计平台能够完成包括系统行为级描述与结构级综合、系统仿真与测试验证、系统划分与指标分配、系统决策与文件生成等一整套电子系统设计过程。同时可编程器件的规模和性能的极大提高,尤其是在系统可编程器件的出现,也为实现上述的 EDA 思想提供了技术保证。运行于 PC 机上的 EDA 平台大大普及了 EDA 的概念,例如,在系统功能级上进行仿真和算法分析的 MATLAB,在电路仿真和 PCB 设计中应用的软件 PSPICE、OrCAD、PROTEL,作为电路测试与实验的软件 EWB 等,加上各种 PLD 器件的开发平台,基本涵盖了从算法设计、系统性能分析、电路(逻辑)设计和仿真直到下载实现的全部设计层次。造就了更多的熟悉 EDA 设计思想的新一代硬件设计者,进一步扩大了 EDA 的应用范围。伴随第三代 EDA 技术的出现,硬件设计者终于开始从使用硬件转向设计硬件。这一革命性的变化使设计师们把精力集中于创造性的方案与概念构思上,极大地提高了设计效率,缩短了产品的研制周期。

综上所述,现代 EDA 技术的基本特征是:

- (a) 采用“自顶向下”的系统级设计方法;
- (b) 广泛采用高级语言即硬件描述语言;
- (c) 具有从系统级、功能级、电路级直至物理级的设计验证和仿真能力;
- (d) 具有高性能的综合优化能力;
- (e) 具有设计的并行性和共享性;
- (f) 具有标准化的开放平台。

6.1.2 硬件描述语言及其开发软件简介

硬件描述语言 HDL(Hardware Description Language)是一种用形式化方法来描述数字电路和设计数字逻辑关系的语言。采用 HDL 的设计包括三大步骤:设计描述→仿真验证→优化综合。具体来讲,HDL 通常都包括逻辑方程方式、真值表方式和状态图描述方式,它可以使设计者利用这种语言来描述自己的逻辑设计思想;然后通过 EDA 工具进行仿真并修改设计;最后自动优化综合,生成门级电路,并用 ASIC 或可编程逻辑器件实现其功能。目前,大部分的 ASIC 和可编程逻辑器件是采用 HDL 方法完成设计开发的。最主要的硬件描述语言有 VHDL, Verilog HDL, ABEL-HDL 和 AHDL。

1. VHDL

VHDL 的英文全名是 Very High Speed Integrated Circuit HDL(超高速集成电路硬

件描述语言)。它是 20 世纪 70 年代末和 80 年代初,由美国国防部为其超高速集成电路 VHSIC 计划开发的硬件描述语言,其早期目的是为了在各个承担国防部订货的集成电路厂商之间建立一个统一的设计数据和文档交换格式。1987 年,IEEE (The Institute of Electrical and Electronics Engineers 电气和电子工程师协会)公布了 VHDL 的标准(IEEE STD1076 / 1987)。

VHDL 是一种全方位的硬件描述语言,包括从系统到电路的所有设计层次,主要用于描述数字系统的结构、行为、功能和接口。VHDL 的语言形式和描述风格上分类似于一般的计算机高级语言。其特点如下。

(a) 具有很强的行为(功能)描述能力。它的语言描述能力极强,覆盖了逻辑设计的诸多领域和层次,并支持众多的硬件模型,从而成为系统设计领域最佳的硬件描述语言。强大的行为描述能力是避开具体的器件结构,从逻辑功能上描述和设计大规模系统的重要保证。

(b) 具有层次结构性。VHDL 的程序结构特点是将设计实体(一个元件、电路模块或一个系统)分成外部(即端口)和内部。外部描述系统输入/输出接口和有关参数;内部描述系统内部的结构和行为状态,并且可形成功能模块,以备其他设计调用。这种分开描述有助于层次化的设计。

(c) 是一种并行语言,其编程思想与传统顺序执行的计算机语言(如 C, Pascal)有很大的区别。

(d) 对设计的描述具有相对独立性,设计者不必考虑最终设计实现的目标器件是什么。

(e) 可重复利用他人的 IP 模块(具有知识产权的功能单元块)。这是 VHDL 的特色。

(f) 它是一种标准语言,具有良好的可读性,移植性很强。其设计描述可以被不同的工具所支持。经过 EDA 工具综合处理,最终生成付诸生产的电路描述或版图参数描述的工艺文件。

目前,在可编程逻辑器件设计输入中,广泛使用 VHDL,并且规定每个 PLD 厂家开发的设计系统都要支持 VHDL。但 VHDL 设计的最终实现取决于针对目标器件的编程,工具不同导致综合的质量也不一样。

2. Verilog HDL

Verilog HDL 是 1983 年由 GDA(Gateway Design Automation)公司的 Philip R.Moorby 创建的。IEEE 于 1995 年制定了 Verilog HDL 的 IEEE 标准:IEEE1364--1995。从此,Verilog HDL 就成了一种极具竞争力的用于数字电路与系统设计的硬件描述语言。

Verilog HDL 是一种简洁清晰、功能强大、便于学习、容易掌握的硬件描述语言,只要有 C 语言的编程基础,在了解了 Verilog HDL 的基本语法、建模方式等以

后,辅之以上机操作,就能很快掌握这一新的设计技术。

近年来,关于 VHDL 语言和 Verilog HDL 语言在 EDA 界一直争论不休。这两种语言各有所长,市场占有率也相差不多。Verilog HDL 是专门为 ASIC 设计而开发的,通常适用于寄存器传输级(RTL)和门电路级的描述,是一种相对较低级的描述语言。而 VHDL 语言通常适于行为(功能)级和寄存器传输级(RTL)的描述,是一种高级描述语言,更适合于描述系统功能,但几乎不能直接控制门电路的生成。大多数 EDA 软件都支持这两种硬件描述语言。

3. ABEL-HDL 和 AHDL

这两种硬件描述语言是由相应的 EDA 开发软件所限定使用的语言,还没有成为 IEEE 标准。它们适合于寄存器传输级(RTL)和门电路级的描述。它们受支持的程度远远不如 VHDL 和 Verilog HDL。ABEL-HDL 和 AHDL 从可编程逻辑器件的设计中发展而来。

ABEL 是由 Data I/O 公司开发的,虽然有不少 EDA 软件(如 ispEXPERT, Synario, Foundation)支持,但提供 ABEL-HDL 综合器的 EDA 公司仅 Data I/O 一家。AHDL 则完全集成于 Altera 公司的 MAX+PLUS II 的软件开发系统中,只能在该开发软件中进行编译和调试。两种语言的共同特点是语法简单、格式简洁、使用灵活,虽然作为复杂程度很高的系统级设计并不完全合适,但作为一般规模的系统设计和实现,其功能并不弱。自顶向下的现代电子系统设计思想,集设计、仿真、下载、调试于一体的开发环境,层次设计、混合输入等各种常规方式等,在其开发工具中都得到了充分体现,因此,十分适合于初学者入门。

6.2 ABEL-HDL 的基本语法

6.2.1 ABEL-HDL 的特点

ABEL-HDL 的特点:

- (a) 适用于各种型号的 PLD 器件的通用语法;
- (b) 结构化的高级设计语言;
- (c) 支持各种逻辑描述形式——逻辑方程式、真值表、状态图;
- (d) 支持仿真与测试向量;
- (e) 省时的宏定义及指示字;

(f) ABEL 源文件是一个 ASCII 格式的文本文件,并可通过标准格式化转换到其他设计环境;

(g) 语言规定简洁明了,极易入门。

ABEL-HDL 设计环境包括两部分,一部分是符合 ABEL 语法规定的高级语言源程序,用来描述逻辑设计;另一部分是语言处理程序,用于编译 ABEL 源程序,提供测试仿真结果,同时也可将逻辑描述适配为指定可编程逻辑器件所需要的二进制代码编程文件,最后在下载硬件的支持下完成下载编程。ABEL-HDL 的处理程序提供了如下强大功能:

- (a) 语法检查;
- (b) 检验所选器件能否实现设计要求;
- (c) 逻辑简化;
- (d) 设计模拟;
- (e) 自动生成各类文件。

本章以 Synario 版本的语言处理程序所支持的 ABEL-HDL 为例,简要介绍该语言的具体语法规则及其源程序的结构,更详细完整的材料可参考 ABEL-HDL 手册。

6.2.2 ABEL-HDL 的语法规则

1. 一般语法规则

ABEL-HDL 是以 ASCII 字符为基本符号集构造的源程序。字符集由 ASCII 字符组成,可以使用 101 键盘中的大部分字符。这些字符可用于标识符、字符串和注释中。

ABEL-HDL 源文件中的每一行必须符合以下语法规则和限制条件:

- (a) 一行最多为 150 个字符长。
- (b) 每行可用换行符(0A),纵向列表符(0B)或换页符(0C)来结束。在大多数情形下,用户通过按回车键来结束一行。
- (c) 关键字、标识符和数之间最少要用一个空格隔开。
- (d) 在关键字、运算符或标识符中不可嵌入空格或句点。
- (e) 关键字可以用大写、小写或大小写字母混合的方式键入。关键字中字母大小写等同。
- (f) 标识符可以是大小写或大小写字母混用,但它对大小写是敏感的。

2. 标识符与关键字

标识符是用户提供的名字或标号,用于标识器件(devices)、器件引脚(pins)、内部节点(nodes)、各种功能块(functional blocks)、集合(sets)、输入信号(input)、输出信号(output)、常量(constants)、宏(macros)以及哑变量(dummy arguments)等。

标识符规则如下:

(a) 标识符长不得超过 31 个字符。

(b) 标识符必须以字母或下划线 “_” 开头。

(c) 标识符除首字母或下划线以外, 可以是大小写字母、数字、波浪线 “~” 和下划线 “_”。

(d) 标识符中不得出现空格, 建议用下划线或大写字母来区分一个标识符中的各个字词以构成短句。

(e) 除了保留的标识符(关键字)外, 标识符与字母的大小写有关。

(f) 除合法的点扩展名外, 标识符中不能用英语句号 “.”。

下面是合法的标识符的例子:

HELLO

hello

_K5input

This_is_a_long_identifier

AnotherLongIdentifier

以下则是不合法的标识符:

7_ 和 \$4 (没有以字母或下划线开头)

HELLO (包含了一个句点)

b6 kj (包含了一个空格)

为了更好地使用标识符, 除不违反以上的规定外, 还应注意以下几点:

(a) 标识符应做到“见名知意”, 符合它表达的函数功能或逻辑意义, 以增加可读性。如: And_In1, And_In2, 表示二输入与门的两个输入信号。加法器上的进位输入引脚可用 Carry_In 标识符命名。

(b) 标识符应简单明了, 易于区分, 避免使用大量相似的标识符。

(c) 一个标识符如用几个单词组成, 则各单词之间宜用下划线或大小写字符加以区分, 以方便阅读。例如: THIS_IS_A_COUNTER 或 ThisIsACounter 就比 THISISACOUNTER 或 Thisisacounter 更容易阅读和理解。

关键字是 ABEL_HDL 语言系统所保留的标识符。因此, 用户命名标识符时不能使用关键字, 否则, 处理程序编译时会给出一个错误标志。表 6.2.1 中列出关键字及其说明。

3. 常量、数值与字符串

ABEL-HDL 中有各种常量。常量可以是非数值的特殊常量值。这些特殊常量值可以是大小写字母 (见表 6.2.2)。

表 6.2.1 ABEL-HDL 中的关键字

关键字	说 明	关键字	说 明
async_reset	异步复位状态	case...endcase	条件选择
declarations	定义段	device	器件定义
end	结束	equations	逻辑方程
functional...block	功能模块定义	fuses	熔丝状态定义
goto	无条件转移	interface	功能模块接口定义
istype	属性定义	trace	跟踪选择
library	库引用	macro	宏定义
module	模块定义	node	节点定义
options	控制选项	pin	引脚定义
property	特征定义	state	状态描述
state_diagram	状态图	state_register	状态寄存器说明
sync_reset	同步复位状态	test_vectors	测试向量
title	标题	with...endwith	转移方程语句
truth-table	真值表	when...then...else	条件转移(只在方程中使用)
if...then...else	条件转移(只在状态图使用)		

表 6.2.2 特殊常量值

常 量	说 明
H	逻辑高电平
L	逻辑低电平
.C.	时钟输入(低电平—高电平—低电平转换)
.K.	时钟输入(高电平—低电平—高电平转换)
.D.	时钟下降沿(高电平—低电平转换)
.U.	时钟上升沿(低电平—高电平转换)
.F.	悬浮的输入/输出信号
.X.	任意值
.Z.	高阻态
.P.	寄存器预加载

常量也可以是数值。ABEL-HDL 中使用了数值操作数，宽度为 128 位。这意味着它支持的数值范围是 $0 \sim 2^{128} - 1$ 。数值有 5 种格式，前 4 种是基值（见表 6.2.3）。基值符号可以是大写或者是小写字母。

表 6.2.3 不同基值的数的表示方法

基值名	基 值	符 号	例
二进制	2	^B	^B1011
八进制	8	^O	^O76
十进制	10	^D(缺省)	^D45
十六进制	16	^H	^H123

“^”是一个键盘字符，不是控制键“Ctrl”。通常缺省的基值是 10。若一个数没有前导基值符号，则默认它是一个十进制数；可以使用@RADIX 命令改变缺省默认的数制基值。

第 5 种格式是用字符串指定一个数值。字符串代表的数值用串中字母的 ASCII 码值表示。例如，字符“a”的 ASCII 码值是十进制 97 或者十六进制 61。字符“b”的 ASCII 码值是十进制 98 或者十六进制 62。对字母字符序列，首先将其转换成相应的二进制代码，然后连接起来形成一个数。例如：字符序列 abc 表示的数是 ^H616263 或 ^D6382179。

在实际编写源文件时用户可直接使用常量。有时为书写方便，也可将一常量赋予标识符，并在整个模块中用此标识符代表该常量，如将 H, L 定义为 1, 0。将 C, 和 X, 定义为 C 和 X, 等等。

4. 运算符和表达式

ABEL-HDL 的运算符分为 4 种类型：逻辑运算符、算术运算符、关系运算符和赋值运算符。赋值运算符稍后介绍，先讨论前 3 种运算符及其运算规则。

(1) 逻辑运算符

逻辑代数中的各种逻辑运算，如与、或、非、异或等，在 ABEL-HDL 的语句中有专门的符号，这些符号称为逻辑运算符。表 6.2.4 列出了标准逻辑运算符。逻辑运算符按位(bit)进行运算，若其施行于两个多位运算数上时，则两个运算数逐位对齐运算。

表 6.2.4 标准逻辑运算符

运算符	定 义	举 例	逻辑代数中的含义
!	非(按位求反)	! A	\bar{Q}
&	与	A & B	AB
#	或	A # B	A + B
\$	异或	A \$ B	A ⊕ B
! \$	异或非(同或)	A ! \$ B	A ⊙ B

(2) 算术运算符

算术运算符指定表达式中各项之间的算术关系。移位运算符也包括在这一类中，表 6.2.5 列出了算术运算符。

在不同场合，减号(-)具有不同的含义。当减号只和一个操作数一起使用时，它指定求补；当减号处于两个操作数中间时，它指定将第二个操作数的补码加到第一个操作数上，即用第一个操作数减去第二个操作数。

除号(/)用于无符号整数除法，其结果是正整数。使用模运算符(%)得到除运算后的余数。移位运算执行无符号逻辑移位，右移时，左边高位用零填充，所以，右移一位相当于被 2 除。左移时，右边低位用零填充，所以，左移一位相当于乘 2。

表 6.2.5 算术运算符

运算符	例	说 明
-	-A	求补(负)
-	A-B	减
+	A+B	加
下述运算符不支持集合运算		
*	A*B	乘
/	A/B	无符号整除
%	A%B	模(即整除后的余数)
<<	A<<B	A 左移 B 位(bit)
>>	A>>B	A 右移 B 位(bit)

(3) 关系运算符

关系运算符对表达式中的两项进行比较。关系运算符形成的表达式产生布尔真或假的值。表 6.2.6 列出了关系运算符。

表 6.2.6 关系运算符

运算符	说 明	运算符	说 明
==	等于	<=	小于或者等于
!=	不等于	>	大于
<	小于	>=	大于或者等于

所有关系运算符都是无符号运算。考察关系表达式 $!0 > 4$ ，由于 ABEL-HDL 中数值是 128 位，因此，!0 的值是 128 位(bit)，每位都为 1，所以， $!0 > 4$ 为真。

(4) 表达式和运算符的优先级

表达式是标识符和运算符的组合。表达式中可以使用任一个逻辑运算符、算术运算符或关系运算符，并根据运算符规定对标识符运算，运算后产生一个结果。当多个运算符混合运算时，则按优先级从高到低的次序进行运算。在 ABEL-HDL 中最高优先级是 1，最低优先级是 4，如表 6.2.7 所示。

优先级相同的运算符从左到右展开运算。使用括号可以改变运算的次序。

可以利用 @Alternate 替换运算符命令，选用另一运算符集，如表 6.2.8 所示。使用替换的运算符集后就不能再使用 ABEL-HDL 的加、乘和除运算，因为此时它代表逻辑或、与及非运算符。替换的运算符集在使用命令 @Standard 之前或者本模块结束之前一直是有效的。

5. 逻辑等式和赋值运算符

ABEL-HDL 中用 4 个赋值运算符 =、?=、:=、?:= 来构成逻辑等式。下面介绍它们的区别。

表 6.2.7 运算符优先级

优先级	运算符	描 述	优先级	运算符	描 述
1	~	负 (求补)	3	#	或
1	!	非	3	\$	异或
2	&	与	3	!\$	异或非 (同或)
2	<<	左移	4	==	相等
2	>>	右移	4	!=	不相等
2	*	乘	4	<	小于
2	/	除	4	<=	小于等于
2	%	模	4	>	大于
3	+	加	4	>=	大于等于
3	-	减			

表 6.2.8 替换的运算符集

ABEL-HDL 运算符	替换的运算符集	说 明
!	/	NOT(非)
&	*	AND(与)
#	+	OR(或)
\$	+	XOR(异或)
!\$	*	NOT XOR(同或)

(1) 赋值的意义

逻辑等式是把等号右边表达式的值赋给左边的输出信号(或信号集合)。如果等式左边使用非运算符(!),则表示在将等号右边的表达式的值赋给这个信号之前先做求非运算。如: $!A=B\#C$ 即为 $\overline{A} = B+C$ 或 $A = \overline{B+C}$ 。

(2) 立即赋值和时钟赋值

在 4 个赋值运算符中, = 和 ?= 是立即型,表示将右边表达式的值立即赋值给左边的信号,适用于组合逻辑赋值。:= 和 ?:= 则是延迟型,表示在下一个时钟脉冲到来时才将右边表达式的值赋给左边的信号,它适用于寄存器(即触发器)赋值。

(3) 确切赋值和随意赋值

在 4 个赋值运算符中, = 和 := 用于完全确定的赋值, ?= 和 ?:= 用于随意条件的赋值。例如,某逻辑函数 $F(A,B), F(11)=0, F(01)=1, F(10)=1$, 而 $F(00)$ 没有指定。如果用逻辑等式表达,可写为

@DCSET

$F = A \& !B \# !\&B;$

$F? = !A \& !B; \quad //指定随意条件$

指定随意条件对编译时的优化有利,但必须同时使用 @DCSET 命令,或者用 ISTYPE 指定信号的“DC”属性。

(4) 多次赋值

ABEL-HDL 允许对同一标识符多次赋值,称之为多重赋值。注意同一标识符在多次赋值后,并不等于其最后一次的赋值结果。当某个标识符出现在多个逻辑等式左边时,赋值给这个标识符的多个表达式首先“或”在一起,然后赋给标识符。如果对逻辑等式左边的标识符求反,则在所有表达式“或”在一起后再求反。这和其他语言的规定不同,请看下表 6.2.9 中的例子。

6. 集合

集合是信号及常量的汇集,集合允许把一组信号用一个名字来引用,简化了 ABEL-HDL 语言的逻辑描述和测试向量。

(1) 集合表示

由逗号或范围运算符(..)分开的常量及信号清单并用方括号将其括起来,可用来表示一个集合。例如, MultOut 及 Select 集合定义如下:

```
MultOut = [B0, B1, B2, B3, B4, B5, B6, B7];
```

```
Select = [S2, S1, S0];
```

以上集合也能用[..]表示,例如,

```
Multout = [B0..B7];
```

```
Select = [S2..S0];
```

(2) 集合索引号

集合索引号使设计者能指定集合中的具体元素,例如,

```
declarations          //集合说明部
```

```
    set1=[f15..f0];
```

```
    set2=[q3..q0];
```

```
equations             //逻辑等式部
```

```
    set2:=set1[7..4];
```

```
    out1:=set1[4]==1;
```

集合索引号的数值指的是集合中元素的位置,当它为 0 时,指集合的最低位置的元素(最右边)。因此, set2:=set1[7..4]是 set1 中的 f8 到 f5 这 4 个元素赋给 set2,即索引号 7 对应 f8,索引号 6 对应 f7,以此类推。而 out1:=set1[4]==1 将集合 set1 第 4 号元素 f5 的值赋给 out1。

(3) 集合运算

适用于集合的运算符如下:

表 6.2.9 多重赋值运算

逻辑等式	等价逻辑等式
A=B; A=C;	A=B#C;
A=B; A=C&D;	A=B#(C&D);
A=!B; A=!C;	A=!B#!C;
!A=B; !A=C;	A=! (B#C);
!A=B; A=!C;	A=!C#!B;
!A=B; !A=C; A=!D; A=!E;	A=!D#!E#!(B#C);

= := ! & # \$!\$ - + == != < <= > >=

它们对集合中的各个对应元素进行运算。集合运算符符合布尔代数运算规则，并有其特点。

集合运算基本规则如下：

- (a) 对两个或多个集合进行运算时，集合中的元素个数必须相同。
- (b) 使用逻辑运算符进行运算，实际是两个集合中的对应元素进行运算。如：
[a,b,c]&[d,e,f]等价于[a&d,b&e,c&f]。
- (c) 比较两个集合是否相等，必须是对应元素全部相等才算相等。
- (d) 在比较两个集合大小或进行算术运算时，集合中的元素相当于组成一个二进制数，第 0 个元素是最低位。
- (e) 若一个集合与单个信号进行逻辑运算，则集合中的每个元素都与此信号进行对应运算。
- (f) 用单个信号对集合赋值，就是用此信号对每个元素赋值。
- (g) 在将集合和数值比较以及用数值给集合赋值时，如果数值宽度不够，则在数值左边添 0；如果宽度超过，则将左边超过的位删除。

试对比如下例子：

[a,b]+[c,d,e]	是错误表达式
[a,b,c]&[d,e,f]	等价于 [a&d,b&e,c&f]
[a,b,c]&sig	等价于 [a&sig,b&sig,c&sig]
[a,b,c]=sig	等价于 {sig, sig, sig}
sigset=[1,1,0]&[0,1,1];	等价于 sigset=6&3;
[a, b]^B101011;	等价于 [a, b]^B111;
[d, c]^B01;	等价于 [d, c]^B1;

再如，[A1, A2, A3]=1 的结果是

$$A1 = 0 \quad A2 = 0 \quad A3 = 1$$

而[A1, A2, A3]=7（或^B111）的结果则是

$$A1 = 1 \quad A2 = 1 \quad A3 = 1$$

7. 变量和变量置换

变量用于宏、模块和命令中。在 ABEL-HDL 中，有两种类型的变量：实变量和哑变量。它们定义如下：

哑变量 一个标识符，指出在宏、模块或命令中实变量被代人的地方。

实变量 实变量置换宏、模块或命令中的哑变量。一个实变量是一个文本，它包括标识符、数字、字符串、运算符、集合或任何一个 ABEL-HDL 中的其他元素。

哑变量在宏说明、宏体、模块以及命令中被指定。哑变量以一个“?”做前缀，

它放在实变量代人的地方。

例如,宏定义

```
OR_EM MACRO(a, b, c){?a#?b#?c}
```

定义了一个名为 OR_EM 的宏,该宏是 3 个变量的逻辑或。这些变量在宏定义中用哑变量 a, b 和 c 代表。在宏体(用花括号括起来的部分)中,哑变量前面的“?”指出了实变量代人的位置。

当用实变量 x, y, 2 & 1 调用宏 OR_EM 时,由 D = OR_EM(x,y,2 & 1); 则产生了如下逻辑等式:

$$D = x \# y \# 2 \& 1;$$

真实变量必须严格按它出现的形式替代,其中的空格也不能忽略。哑变量与真实变量的关系总结如下:

- (a) 哑变量的位置是保留给真实变量的。
- (b) 前面带问号的哑变量需用真实变量替代。
- (c) 变量替代在检查源文件前完成。
- (d) 真实变量中空格有意义。

在检查源文件的语法和逻辑之前,哑变量被实变量置换。如果实变量包含了不支持的语法和逻辑,则编译程序在置换之后将会检查并报告出错误。

8. 注释

注释是使源文件易于读懂的一段文字。ABEL-HDL 规定了两种注释方法:

- 以双引号 (“) 开始,以另一双引号 (”) 结束或该行回车结束;
- 以双斜杠 (//) 开始,以该行回车结束。

语言处理程序对双引号中的注释内容不进行处理。

利用注释双引号对程序中无用的语句进行屏蔽,使该语句在编译时无效。这是调试程序时常用的方法。如要使语句

```
IC4 device'P16V8R'
```

在编译时无效,可用如下方法:

```
//IC4 device'P16V8R'
```

9. DIRECTIVES 指示字

指示字也是 ABEL-HDL 处理程序中规定的一组关键字。在源程序中指示字均以符号@打头。指示字在源文件中的作用是指示处理程序如何对源文件的内容进行处理。根据不同指示字,处理程序可以有条件地对源文件的某些部分进行不同的操作,例如,可以在源文件中加入另一源文件的内容;可以按需要在处理的某一步输出必要的显示信息;可以自动重复某些过程等等。

使用指示字可以简化逻辑描述和文件调试。表 6.2.10 中列出了 ABEL-HDL 中

的常用指示字

表 6.2.10 常用指示字

指示字	功 能	指示字	功 能
@Alternate	替代	@Include	引用
@Standard	标准	@Irp	无限重复
@Const	常量	@Irpc	字符无限重复
@Message	信息显示	@Ifiden	如果相等
@Ifniden	如果不相等	@Page	分页
@Expr	表达式	@Radix	基数
@Exit	退出	@Repeat	重复
@If	条件	@Ifb	若为空
@Ifnb	若不为空	@Ifdef	如果定义过
@Ifndef	如果未定义过	@Dcset	不完全定义逻辑

在前面的若干例子中，已经用过的指示字有

@Alternate @Standard @Radix

指示字也是关键字，所以其字母可以大写、小写或大小写混用。

6.2.3 ABEL-HDL 中对基本逻辑器件的描述

数字逻辑电路中有两种基本逻辑器件,即门和触发器,在 ABEL-HDL 中,触发器都统称为寄存器。

对于逻辑门电路可直接用逻辑表达式表示。例如,

$$\text{Out1} = A \& B \& !C$$

表示一个与门。

对于寄存器的描述则比较复杂,需要时钟方程、复位方程、状态方程等一组方程才能完整地描述。为了简洁,ABEL-HDL 对一个(或一组)寄存器只定义一个标识符,而将该寄存器的时钟信号,复位信号等等都在该寄存器的标识符后面加点扩展名(点后缀)来表示。因此,使用点扩展名时的语法规则是“寄存器标识符.扩展名”。

例如,一个 4 位计数器的时钟方程、复位方程如下:

$$[Q3..Q0].\text{clk} = \text{CLK};$$

$$[Q3..Q0].\text{cr} = !\text{CR};$$

也可用集合 count=[Q3,Q2,Q1,Q0]表示为

$$\text{count}.\text{clk} = \text{CLK};$$

$$\text{count}.\text{cr} = !\text{CR};$$

完整的点扩展名定义可以帮助我们更精确地描述一个电路的行为。点扩展名中有些是与器件结构无关的,有些则是某些器件特有的。图 6.2.1 标示了一个 JK 触发器及其部分扩展名。ABEL-HDL 支持的点扩展名如表 6.2.11 所示。

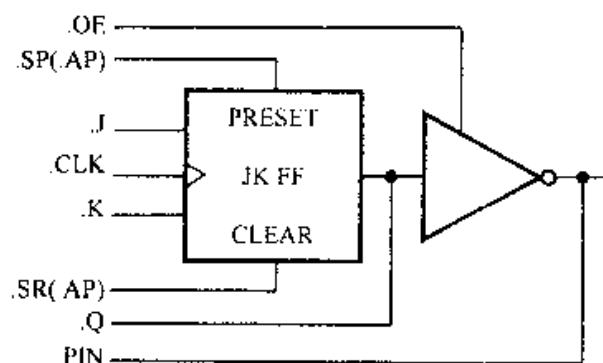


图 6.2.1 寄存器标识扩展名示意图

表 6.2.11 ABEL-HDL 支持的点扩展名

点扩展名	说 明	点扩展名	说 明
· ACLR	异步复位	· J	JK 触发器的 J 端
· ASET	异步置位	· K	JK 触发器的 K 端
· CLK	边沿触发器的时钟	· LD	锁存器 D 端
· CLR	同步复位	· LE	锁存器锁存允许端
· COM	不经寄存器的组合反馈	· LH	锁存器锁存允许 (H) 端
· FB	从寄存器输出反馈	· PR	寄存器预置位
· OE	输出允许	· Q	寄存器反馈
· PIN	从输出引脚反馈	· R	SR 触发器 R 端
· SET	同步置位	· RE	寄存器复位
· AP	异步寄存器预置位	· S	SR 触发器 S 端
· AR	异步寄存器复位	· SP	同步寄存器预置位
· CE	门控时钟触发器的时钟允许	· SR	同步寄存器复位
· D	D 型触发器 D 端	· T	T 型触发器 T 端
· FC	触发器模式控制		

点扩展名也是关键字,所以其字母可以大写、小写或大小写混用。

对寄存器而言, := 和 ?:= 赋值运算符只能用于写引脚到引脚(与器件结构无关)的寄存器逻辑等式,而在使用点扩展名赋值时使用 = 和 ?= 赋值运算符。比较下面的两个逻辑等式:

q0:=a&b; //逻辑等式 1

q0.d=a&b; //逻辑等式 2

逻辑等式 1 指出,当时钟到来时,将 a&b 的值赋给 q0,它没有指出寄存器的类型,只指出寄存器输出端的行为;等式 2 则明确指出将 a&b 的值立即赋给寄存

器的 D 端。如果两个等式中描述的寄存器都是 D 型寄存器，那么，两等式的行为是一致的；但等式 1 适用范围广，而且对其他各种不同类型的寄存器也适用。

6.3 ABEL-HDL 模块源文件结构

6.3.1 ABEL-HDL 模块基本结构框架

先看一个语言实例，描述一个 2 位计数器的 ABEL-HDL 模块源程序如下：

```
MODULE counter                                //标题段模块名
TITLE 'Two Bits counter Block';               //标题
DECLARATIONS                                  //定义段声明
    clock          PIN;                       //InPut
    q0, q1          PIN    istype 'reg';      //OutPut
EQUATIONS                                      //逻辑描述段
    q0.clk=clock;
    q1.clk=clock;
    q0:=!q0;
    q1:=(q1&q0#q1&q1$q0!q0);
END      counter                             //结束段
```

这个名为 counter 的两位计数器只有一个输入引脚 clock，是计数器的时钟。它有两个输出引脚，q0 是 clock 的二分频，q1 是 q0 的二分频。它虽然简单，却是一个完整的设计模块。经过编译、连接、器件适配、下载后，能在可编程器件中实现两位计数器的功能。对照该例，可以看出 ABEL-HDL 模块的基本结构有 5 段，如本例中的标题段、定义说明段、逻辑描述段、结束段，设计仿真期间往往对每个设计模块还要加上测试向量段进行测试。

源文件的基本框架如下：

```
//标题段
module  模块名[(哑变量名[, 哑变量名]...)]
[flag 标志语句]...
[title 标题语句]...
//定义段
[declarations]
device 器件名
```



```

        信号 pin
        信号 node
        常量 constant、属性、宏等
//逻辑描述段
    [equations 方程]...
    [truth_table 真值表]...
    [state_diagram 状态图]...
//测试向量段
    [test_vectors]...
//结束段
end [模块名]

```

下面具体讨论 ABEL-HDL 源文件的上述基本结构。

6.3.2 ABEL-HDL 模块结构

1. 标题段

(1) 模块语句

关键字 MODULE 或 module

格式 module 模块名

此语句是必须的。模块语句定义一个模块的开始，而且必须有一个 END 语句与之相配合。模块名是用户自定义的模块名称。模块语句相当于原理图文件中的元件符号。在 Synario System 中，模块名应按标识符的规则命名。

(2) 标题语句

关键字 TITLE 或 title

格式 title'字符串'

此语句为可选语句，主要说明模块的内容、用途、作者、设计时间和地点等，单引号中为说明内容，ABEL-HDL 在编译时不处理此语句。

在 Synario ABEL 语言处理系统中，若增加一个 ABEL-HDL 源文件，则会弹出文件信息对话框，要求输入模块名和标题等信息。

(3) 接口语句

关键字 INTERFACE 或 interface

格式有两个，在较高层模块中声明所用到的较底层模块时的语法是

模块名 interface (输入信号 / 集->输出信号 / 集:>双向信号 / 集)

在较低层模块中的语法是

interface (输入信号 / 集->输出信号 / 集:>双向信号 / 集)

接口语句也为可选语句,其作用是在当前源文件中指定一个低层模块并定义它的端口信号和默认值,即声明某一模块的使用接口。接口语句主要用于层次设计中,通常和 FUNCTIONAL_BLOCK 语句配合使用以声明该低层模块的每个具体例化。具体应用方法在后面层次设计中举例说明。

2. 定义段

定义段用以声明设计中使用的信号名字(引脚或内部节点)及属性,定义常量、宏和状态,说明低层模块和电原理图,以及说明一个器件(可选)。每个模块至少有一个定义段,定义段一般跟在标题语句后面。

(1) 说明关键字 DECLARATIONS 或 declarations

declarations 显示说明了定义段的开始,关键字 declarations 允许在源文件的任何一部分中进行说明,但若在模块中定义段紧接在标题段后,位于逻辑描述段之前,也可不加这个关键字。

(2) 器件定义

关键字 DEVICE 或 device

格式 器件标识符 device 实际器件

器件定义语句把一个器件标识符与特定的实际可编程逻辑器件联系在一起。实际器件为所代表的器件的工业型号,用字符串表示。在 Synario System 中使用的实际器件是在菜单下选择的,故器件定义语句可以省略。每个模块中最多只能有一个器件说明。

例: UI device 'MACH211';

(3) 信号引脚定义

关键字 PIN 或 pin

格式 [!]信号[, !]信号 ... pin [引脚号, 引脚号...] [istype '属性'];

格式中,方括号中的内容可选,下面的格式语句中均如此表示。

(4) 信号节点定义

关键字 NODE 或 node

格式 [!]信号[, !]信号 ... node [节点号, 节点号...] [istype '属性'];

引脚(pin)代表模块对外的信号,节点(node)代表模块内部的信号。

引脚和节点说明用来说明设计中使用的信号,并可根据需要把这些信号和模块的引脚号或节点号联系在一起。在一个信号前加!表示该信号为低有效。使用带有关键字 istype 的引脚和节点说明能够给信号分配属性。在逻辑等式中使用点扩展名也能精确地描述信号。在信号引脚定义中的引脚号规定了一个信号必须从器件的特定引脚输入/输出,在不十分必要时不要如此分配引脚号。

(5) 属性说明

关键字 ISTYPE 或 istype

格式 信号[, 信号, ...] istype '属性';

ISTYPE 语句定义信号属性。它可以在节点或引脚说明语句中使用, 也可以单独使用。单独使用时须放在引脚或节点说明语句之后。属性规定了器件结构或者对信号的限制。输入信号不指定属性, 输出信号有属性, 缺省的属性是'COM'。某些属性与器件结构有关, 另一些则与器件结构无关。由于 ABEL-HDL 是与器件无关的通用语言, 若不规定信号属性或者其他信息(如点扩展名), 则在将设计结果从一种器件移植到另一种器件时, 可能工作情况不一致。指定信号的属性并不表示设计必须总是指定一个特定器件。不过, 在针对某个特定器件进行设计时, 使用属性可以更精确地描述电路的行为, 排除模棱两可的情况, 也更能充分利用该器件的资源。

ABEL-HDL 中使用的信号属性如表 6.3.1 所示。

表 6.3.1 信号属性

属 性	说 明	属 性	说 明
buffer	目标器件中寄存器输出到引脚输出之间无反相器	pos	未指定的逻辑是 1
collapse	优化时此信号可能去掉	retain	不对输出进行最小化优化, 保留冗余乘积项
com	组合输出	reg	寄存器
dc	未指定的逻辑为任意态	reg_d	D 型触发器
invert	目标器件中寄存器输出到引脚输出之间有反相器	reg_jk	JK 型触发器
keep	优化时不能去掉此信号	reg_sr	RS 型触发器
neg	未指定的逻辑是 0	reg_t	T 型寄存器
		xor	目标器件中的 XOR 门

若干说明:

(a) 某信号如果既不指定“keep”也不指定“collapse”属性, 则在优化和适配器件时会根据需要决定保留或去掉该信号。

(b) 属性中 neg, pos, dc 是互斥的, 即一个信号只能具有这 3 个属性之一。

(c) 属性 retain 仅控制 ABEL-HDL 编译程序优化过程。如要保留冗余项, 则必须指定该属性。

下面是信号属性的若干例子:

a,b,c pin; //a, b, c 是 3 个引脚

a,b,c pin 3,8,10; //a, b, c 是 3 个引脚, 且 a 引脚号是 3, b 引脚号是 8, c 引脚号是 10

x,y,z node; //x, y, z 是 3 个内部节点

o1,o2,o3 pin istype 'reg_d,buffer';

//o1,o2,o3 输出引脚是 D 型寄存器信号, 寄存器具有缓存功能

(6) 常量定义

关键字 =

格式 标识符[, 标识符, ...]=表达式[, 表达式, ...];

常量是在模块中保持固定不变的值的标识符。它用 = 指定。常量说明须在说明段中, 或者在指示字@CONST 命令之后。

下面是正确的常量说明的例子:

ABC=3*17; //ABC 赋值为 51

Y='Bc'; //Y=^H4263

X-, X.; //X 是随意值

C=.C.; //C 是时钟信号输入(低电平—高电平—低电平转换)

Addr={1,0,1}; //addr 是 3 个元素的集合

A,b,c = 5,[1,0],6; //说明了 3 个常量

(7) 宏定义

关键字 MACRO 或 macro

格式 宏标识符 macro [哑变量, [哑变量], ...]{块};

宏定义语句定义了一个宏。在源文件中使用宏可避免重复写一些功能块代码。

(8) 库说明

关键字 LIBRARY 或 library

格式 library '库名'

library 语句将指定的 ABEL-HDL 库文件插入到当前文件中。

3. 逻辑描述段

逻辑描述段描述设计电路的逻辑功能, 实际上就是对所设计的逻辑电路进行描述。逻辑描述段是设计的主体, 也是设计工作中最繁琐的部分。描述逻辑的方法有 5 种: 逻辑方程、真值表、状态图、熔丝图、XOR 因子。完成了前面的定义段后, 用户就能以这其中的一个或多个表达方式描述电路的逻辑功能。本章仅介绍前 3 种方法。每种表达方式编写的源程序(称为块)都有相应的关键字作为块首。

(1) 逻辑方程

关键字 EQUATIONS 或 equations

语法 equations

信号名 = 表达式;

信号名 ?= 表达式;

信号名 := 表达式;

信号名 ?:= 表达式;

when-then-else 语句:

equations 语句指明逻辑方程块的开始,即以一组逻辑等式来描述器件逻辑功能。在一个具体的描述中,上面的5种逻辑式不一定都存在,但在 equations 关键字下至少有一个逻辑等式。这其中信号名、表达式和赋值运算符等语言要素已在前面讨论。这里介绍 when—then—else 语句。

该语句的一般格式是:

```
when 条件(式) then    逻辑等式;
      [else          逻辑等式; ]
```

表示当条件表达式为真时,执行 then 后的逻辑等式;条件表达式为假时,执行 else 后的逻辑等式;若无 else 语句,则什么也不做,执行下一条语句。when—then—else 语句只用于逻辑方程块中。

例 6.3.1 下面的逻辑方程块描述了一个一位十进制加法计数器。

```
EQUATIONS
Counter.clk = clk;
Counter.ar = cd;
WHEN (Counter = 9) THEN Counter:=0;
ELSE Counter:=(Counter.fb+1);
```

(2) 真值表

关键字 TRUTH_TABLE 或 truth_table

语法 (a) truth_table ([输入]→[输出])

```
[输入]→[输出];
      ⋮
```

(b) truth_table ([输入]:>[寄存器输出])

```
[输入]:>[寄存器输出];
      ⋮
```

(c) truth_table ([输入]:>[寄存器输出]→[输出])

```
[输入]:>[寄存器输出]→[输出];
      ⋮
```

真值表分为表头和表格两部分,表头加圆括号(),注意此处方括号[]是格式所要求,而不是可选记号。→和:>表示从输入到输出。组合型的函数用→,寄存器型的函数用:>。因此,语法(a)是一个组合型真值表输出,语法(b)是寄存器型输出。

表头确定了输入和输出的信号名,表头行不用分号。圆括号内描述的格式规定真值表的格式。表格列出输入组合及其产生的输出。可能的输入组合中的全部或部分在表中列出。表格中的所有值必须是常量,表格的每行(列出的每对输入和输出)

需要用·一个分号结束。

例 6.3.2 下面的真值表块描述了一个 4 位十进制加法计数器（定义为集合 $\text{count}=[Q3,Q2,Q1,Q0]$ ）及 7 段译码器的逻辑关系。

```
truth_table ([count]:>[count]) -> [a,b,c,d,e,f,g])
    [0]  >  [1] -> [1,1,1,1,1,0];
    [1]  >  [2] -> [0,1,1,0,0,0,0];
    [2]  >  [3] -> [1,1,0,1,1,0,1];
    [3]  >  [4] -> [1,1,1,1,0,0,1];
    [4]  >  [5] -> [0,1,1,0,0,1,1];
    [5]  >  [6] -> [1,0,1,1,0,1,1];
    [6]  >  [7] -> [0,0,1,1,1,1,1];
    [7]  >  [8] -> [1,1,1,0,0,0,0];
    [8]  >  [9] -> [1,1,1,1,1,1,1];
    [9]  >  [0] -> [1,1,1,0,0,1,1];
```

本例将表格与表头直接在书写格式上对应起来。例中，前面的:>是寄存器真值关系，即每个时钟脉冲到来时的对应关系，从中可以清楚看出，这是一个十进制加法计数器，而 count 的输出是 7 段显示译码输入，比如第一行对应 7 段中 g 段不亮即显示 0。

(3) 状态图

状态图是描述逻辑关系的一种重要方法。状态图是状态机的变迁表示。状态机是一种数字部件，它划分为若干稳定的“状态”。各状态下输出一些信号(有的状态可能没有输出)，从而完成某些工作。在时钟控制下，状态机按照预定的顺序在状态之间进行转移。使用 STATE_DIAGRAM 定义状态机。

关键字 STATE_DIAGRAM 或 state_diagram

格式 state_diagram 状态标识符 [->状态输出]

state 状态表达式: 逻辑方程

[逻辑方程

⋮

逻辑方程]

状态转移语句

state ...

⋮

上面的格式中，语句“state 状态表达式:”给出状态机的某一状态，状态机中有几个状态，就有几个 state 语句。冒号后的逻辑方程是该状态下的输出。状态转

移语句表示状态机从该状态向另一状态的转移。ABEL-HDL 中定义了 GOTO、CASE 和 IF—THEN—ELSE 3 种状态转移语句，需根据具体的状态图情况来选择。

(a) 状态图形式和 3 种转移语句选择。以图 6.3.1 为例来说明。

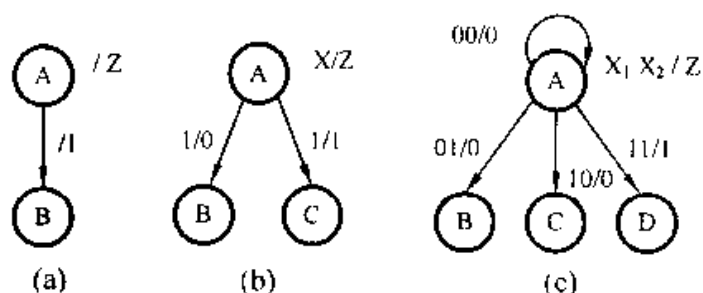


图 6.3.1 状态转移形式

(a) 无条件转移 (b) 简单分支转移 (c) 多分支转移

图 6.3.1(a)表示在状态 A 时，下一状态必为 B，故是无条件转移，应用 goto 语句。

```

格式 state A:  ...
                :
                goto B;

```

图 6.3.1(b)表示在状态 A 时，下一状态根据条件 X 或为 B，或为 C，故是简单分支转移，应用 if—then—else 语句。

```

格式 state A:  ...
                :
                if X==1 then C else B;

```

图 6.3.1(c)表示在状态 A 时，下一状态有 3 个以上分支的情况。此时的转移条件不止一个，并且条件之间是互斥的。这种多分支转移宜用 case 语句实现。

```

格式 state A:  ...
                :
                case !X1&!X2: A;
                case !X1& X2: B;
                case  X1&!X2: C;
                case  X1& X2: D;
                endcase

```

(b) 状态图中的输出信号表示时序电路有 Moore 型和 Mealy 型两种基本形式。在 ABEL-HDL 中描述它们的输出有以下区别。

对于 Moore 型，因其输出仅仅取决于当前状态，即每个状态有一个确定的输出，

所以，只要在每个转移语句前写出输出逻辑表达式即可。

对于 Mealy 型, 其输出不仅取决于当前状态, 还与该时刻输入信号有关。因此, 其输出是伴随着状态变迁条件而不同的。为了描述这种情况, 要用 with 语句, 将输出信息加在 goto 语句或 if—then—else 语句或 case 语句中的每一个状态表达式后面, 仍以图 6.3.1 为例进行说明。

图(a) 格式为 state A: goto B with Z=1;
 endwith;

图(b) 格式为 state A: if X==1 then C with Z=1; endwith;
 else B with Z=0; endwith;

图(c) 格式为

```
state A:
    case    !X1&!X2:    A with Z=0; endwith;
    case    !X1& X2:    B with Z=0; endwith;
    case     X1&!X2:    C with Z=0; endwith;
    case     X1& X2:    D with Z=1; endwith;
endcase
```

(4) 逻辑块

逻辑块并不是逻辑描述段中必需的语句，但在较为复杂的逻辑描述中得到广泛的应用。在很多场合需要用到块，以达到表述更复杂的函数、改进可读性等目的。

块是用花括号“{ }”括起来的文本段。块使用在逻辑等式、状态图、宏和命令中，块中的文本可以是一行，也可以是多行。块还能够嵌套。

(a) 逻辑等式中的块。在一个 when-then 的表达式中,若要确定多于一个的输出是不方便的。

如下面的表达式:

```

when (Mode==S_data)      then Out_data:=S_in;
  else when (Mode==T_Data) then Out_data:=T_in;
when (Mode==S_data)      then S_valid:=1;
  else when (Mode==T_Data) then T_valid:=1;

```

使用块能够将上面表达式简化如下：

```

when (Mode=S_data) then {Out_data:=S_in; S_valid:=1;}
  else when (Mode=T_Data) then {Out_data:=T_in; T_valid:=1;}

```

(b) 状态图输出方程中的块。块提供一种较好的书写状态图输出逻辑等式的方法。例如, 下面两个等价的状态输出语句, 前者不使用块, 后者使用了块, 后者的可读性更好些, 在输出方程较多时此优点尤为明显。

```

if (Hold) then State1    with  Q1:=Q1.fb; Q2:=Q2.fb; ...;
                        endwhile

```



```
else State2;
```

```
if (Hhold) then State1 with { Q1:=Q1.fb; Q1:=Q1.fb; ...; }
```

```
else State2;
```

(c) 状态图转换中使用块。状态图描述中,在嵌套的 if—then—else 语句中使用块能够简化复杂的转换逻辑描述,看下面的例子。

不使用块

```
IF( HOLD & !Reset) THEN State1;
```

```
IF( HOLD & Error) THEN State2;
```

```
IF (!HOLD) THEN State3;
```

使用块

```
IF (HOLD) THEN { IF (!Reset) THEN State1;
```

```
IF (Error) THEN State2; }
```

```
ELSE State3;
```

4. 测试向量段

测试向量段用来检查逻辑段描述的电路设计是否能实现预期的设计功能。该段是以关键字 TEST_VECTORS 引导的测试向量。

关键字 TEST_VECTORS 或 test_vectors

格式 test_vectors [注释]

([输入]~>[输出])

[输入值]~>[输出值];

⋮

可以看到,测试向量的格式类似于真值表,也有一个用圆括号括出的表头和下面的各行向量表 (“[输入值]~>[输出值];” 语句)。表头和向量表的左边是输入,右边是输出,输入或输出可以是一个或一组信号(一组信号必须另用方括号括起来,信号之间用逗号分开),中间由符号~>连接。向量表的输入值和输出值必须与表头匹配。测试向量通过将输出作为输入的函数,规定了希望完成的操作,以检查逻辑段描述的电路设计是否能实现预期的功能。向量表可以写多行,每行必须以分号结束。

ABEL-HDL 将根据向量表,逐行逐句地对编译建立的逻辑模型进行测试仿真。而每行向量表表示了一个给定的输入状态及其对应的输出状态,所以,语句中的信号值必须为常量(包括特殊常量),输出值可以取确定值,也可取任意值(.x.)。若取确定值,则逻辑模拟时将模拟结果与其比较并给出比较结果。若取任意值,则根据输入向量得出输出值,并给出结果。后者常用在波形图显示方式中。

对于输入信号不发生变化,而需重复 N 次测试的场合,如计数器在 N 个时钟下的状态变化,可以应用指示字@repeat N {[输入值]~>[输出值];} 来简化书写。

测试向量段是可选内容，也可以单独作为一个文件来编写，一个单独的测试向量文件，必须包含除逻辑描述段以外的其他各段。

例 6.3.3 测试例 6.3.1 描述的一位十进制加法计数器。

```
test_vectors      //测试一位十进制加法计数器
    ([clk,cd] => [Counter])
//测试表头中，时钟信号 clk 和异步清零信号 cd 为输入，计数器状态为输出
    [x..1] => [0,0,0,0];           //异步清零 cd 有效
    @repeat 12 {[c..0] => [x...x...x...x].;} //重复 12 次时钟
end
```

5. 结束段

关键字 END 或 end

格式 END [模块名]

结束段指示一个模块结束。当一个源文件中有多个模块时，END 后的模块名是必须的。

*6.3.3 ABEL-HDL 中的层次设计方法

1. ABEL-HDL 语言中的层次设计思想

现代电子设计广泛采用结构化、层次化的设计方法，分层次、按模块地进行设计描述。ABEL-HDL 语言处理程序完全支持层次设计，一个 ABEL-HDL 设计的基本单位是模块。一个设计可以只有一个模块，也可以有若干个模块。若是多个模块，则模块间按照层次结构进行组织：有一个顶层模块，用于设计的总体描述或作为顶层系统框图；较低层的模块可分别描述顶层模块中的各子功能块；更低层的模块则对各个子功能块进行细化，这样逐层下去。ABEL-HDL 的层次数不受限制。

另一方面，在实际支持 ABEL-HDL 的 EDA 软件中，通常不仅仅支持各种 HDL 语言及其 HDL 语言间的混合描述，也同时支持原理图输入描述或 HDL 语言和原理图的混合描述，即用各种不同 HDL 语言或原理图设计的模块组成一个完整的设计。因此，实际的系统设计层次也可以是混合的，即有些模块是原理图描述或其他非 ABEL 的 HDL 语言描述。

本章不讨论不同 HDL 语言间混合设计的实现，而对 ABEL-HDL 描述与原理图描述的混合层次设计在本章末用例子加以说明。

2. ABEL-HDL 语言中的层次设计实现方法

应用层次设计的重要环节是正确的层次接口说明，层次说明语句即是层次设计的接口语句。

当一个设计中的各模块组织为若干层次时，高层模块将低层模块作为它的子功

能块使用。使用一个低层模块的方法是

- (a) 低层模块标题段用 `interface` 语句说明其输入/输出信号,它是可选的。
- (b) 高层模块在定义段用 `interface` 语句对将使用的低层模块予以说明。
- (c) 同时,高层模块在定义段用 `functional_block` 语句对低层模块予以例化。
- (d) 高层模块中在逻辑描述段中引用低层模块信号。

高层模块中对同一个低层模块能够予以多次例化,例化是此低层模块的一个具体实现。例如,假定低层模块是一个4位计数器,一个高层模块中使用5个4位计数器,则需要5个 `function_block` 语句对该低层的4位计数器模块进行5次例化,得到5个具体实现的4位计数器,并能互相区别。

3. 接口说明

为充分理解和更好地掌握层次设计方法,需要仔细讨论 ABEL 的层次说明语句。在较低层模块中的语法是

`module` 低层模块名

`interface` (输入信号/集->输出信号/集:>双向信号/集);

低层模块中使用 `interface` 语句为此模块分配缺省端口表和输入值,供高层模块例化时使用。低层模块中的 `interface` 语句在标题段中,紧跟在 `module` 语句之后。低层模块中的 `interface` 语句是可选的。

在较高层模块中声明所用到的较底层模块时的语法是

`module` 高层模块名

低层模块名 `interface` (输入信号/集->输出信号/集:>双向信号/集);

例化名 `functional_block` 低层模块名;

在较高层模块中的 `interface` 语句说明了当前模块中要使用的低层模块,它是必须的语句,并且与功能块说明 `functional_block` 联合使用。在高层模块中,必须按低层模块接口说明的顺序和分组来说明信号和集合。`->` 和 `:>` 定界符用于指出各端口信号的方向。

在高层模块中使用 `functional_block` 说明了以例化名代表的低层模块具体实现后,即可在逻辑描述段引用。引用时,采用的形式是例化名.端口信号名,如

例化名.端口信号名=信号名;

至此,整个设计在层次说明和例化引用上完全打通。

注意:接口说明语句不能包含点扩展名。对于信号的点扩展名逻辑等式而言,它只能在该信号被定义的模块中使用。在该模块之外,绝对不能引用这些信号的点扩展名。如果确实需要指定点扩展名(例如,用于解决反馈的模糊性),则必须在低层模块中引进一个中间信号,提供给高层模块作为连接使用。

下面举一个完整的层次设计例子具体说明以上方法。

例 6.3.4 设计描述:有两个低层模块,一个是有3个输入端和1个输出端的与门

3AND, 另一个是有两个输入端和 1 个输出端的或门 _2OR。高层模块是一个与或非门 A_O_INV, 它例化两个 3 输入与门, 与门的输出由例化或门或起来后求反, 即实现由两个三输入端与门构成的与或非门。

解 共应有 3 个模块, 低层模块 _3AND 和低层模块 _2OR。由一个高层模块 A_O_INV 来例化调用。

(1) 高层模块源程序:

```
module A_O_INV
title '2_3AND_OR_INVERT GATE';
declarations                                // 定义段
    _3AND interface (in_and1, in_and2, in_and3 => out_and);
                                                //与门接口说明
        AND1 functional_block _3AND;          //与门例化 1, 例化名 AND1
        AND2 functional_block _3AND;          //与门例化 2, 例化名 AND2

    _2OR interface (in_or1, in_or2 => out_or); //或门接口说明
        OR1 functional_block _2OR;           //或门例化, 例化名 OR1
in1, in2, in3, in4, in5, in6    pin;
out                              pin;          //信号定义
equations                        //逻辑描述段
    AND1.in_and1 = in1; AND1.in_and2 = in2; AND1.in_and3 = in3;
                                                //第一个例化与门 AND1 的 3 个输入信号
    AND2.in_and1 = in4; AND2.in_and2 = in5; AND2.in_and3 = in6;
                                                //第二个例化与门 AND2 的 3 个输入信号
    OR1.in_or1 = AND1.out_and;    OR1.in_or2 = AND2.out_and;
                                                //例化或门 OR1 的两个输入信号分别是两个例化与门输出
out = !OR1.out_or;                //最后输出的是例化或门 OR1 的输出取非
end A_O_INV
```

(2) 低层模块 _3AND 源程序:

```
module _3AND
interface (in_and1, in_and2, in_and3 => out_and); //与门接口说明
title '3 input AND gate';
declarations                                //说明段

    in_and1, in_and2, in_and3    pin;
    out_and                      pin    istype'com';
equations                        //逻辑描述段
```

```

        out_and = in_and1&in_and2&in_and3;
    end _3AND
(3) 低层模块_2OR 源程序:
    module _2OR                                //省略了 interface 语句
    title '2 input OR gate';
    declarations                                //说明段
        in_or1, in_or2        pin;
        out_or                pin istype'com';
    equations                                    //逻辑描述段
        out_or = in_or1#in_or2;
    end _2OR

```

6.4 ISP Synario 软件及其使用方法

PLD 产品受众多开发工具软件的支持, 这些开发工具软件一般是第三方公司的 EDA 软件或同某个器件公司联合推出的工具软件, ISP Synario 系统就是这样一个套装软件, 包括 Synario 软件和 Lattice 公司的 PDS + Fitter 适配器软件。Synario 是美国 Data I/O 公司开发的运行于 PC 机 Windows 环境的优秀通用电子设计工具软件。

6.4.1 ISP Synario 软件的特点及安装

1. ISP Synario 软件的特点

ISP Synario 软件提供了设计输入、设计实现、设计仿真及器件编程的整个过程, 也包括其相应各步骤所需要的可执行文件和相关的库文件。

(1) 设计输入

ISP Synario 软件的设计输入、设计实现和设计仿真都是在软件集成环境下进行的。该开发软件把整个设计视为一个“项目”或“工程”(Project), 用“项目导航器”(Project Navigator) 对一个项目进行统一管理, 包括用户输入的各种设计描述文件、仿真文件、开发文档等, 也包括 Synario 开发环境所生成的各种中间文件, 详见表 6.4.1。用户输入设计描述文件称为“源”(Source)。一个“项目”包含多个源时, 只有一个源作为顶层文件, 其他输入文件作为与顶层文件有关的底层文件。Synario 软件支持多种输入方式: 教学版本的 Synario 软件只有原理图输入和 ABEL-HDL 文本输入; 专业版的 Synario 软件还支持 VHDL 和 Verilog HDL 硬件描

述语言的文本输入和编译调试。

表 6.4.1 ISP Synario system 部分文件后缀

扩展名	文件来源	文件类型	扩展名	文件来源	文件类型
SYN	源文件	设计项目管理文件	PPN	源文件	引脚锁定描述文件
ABL	源文件	ABEL-HDL 源文件	PAR	源文件	适配器控制参数文件
ABV	源文件	测试向量描述文件	FXP	中间文件	逻辑布局结果文件
SCH	源文件	电路原理图文件	LST	中间文件	ABEL 源文件的列表文件
SYM	中间文件	电路符号文件	LOG	中间文件	运行流程记录文件
EQ0	中间文件	逻辑描述文件	SIM	中间文件	模拟用网表文件
EQ1	中间文件	简化逻辑文件	JHD	中间文件	层次化关系连接表文件
EQ2	中间文件	带层次连接关系的逻辑描述文件	JED	结果文件	熔丝图文件、JEDEC 文件
EQ3	中间文件	经优化的逻辑描述文件	REP	结果文件	GAL 器件设计编译报告文件
EQ4	中间文件	经反复优化的逻辑描述文件	RPT	结果文件	ispLSI 器件设计编译报告文件
TMV	中间文件	经编译的测试向量文件	XRF	结果文件	信号和节点简缩名称文件
TT2	中间文件	Synario 逻辑网表输出文件、适配器输入文件	ERR	结果文件	错误报告文件

(2) 设计实现

主要包括设计检验、布局、布线以及生成目标文件即 JEDEC 文件(熔丝图)。编译时完成设计检验, 开发软件将编译的结果报告给设计者, 以便设计者修改设计。布局、布线工作是由软件系统自动完成的, 能以最优的方式对逻辑元件、输入/输出单元布局, 并准确地实现元件之间的互联。设计实现的最后一步就是要指定编程的器件, 并生成目标文件(熔丝图)。每个输入文件都可以单独进行检查、优化和编译等过程。

(3) 设计仿真

包括逻辑仿真和定时仿真。在仿真前需要设计者预先建立好测试向量及测试序列, 可通过报告文件形式或波形观察器检查仿真结果。

(4) 下载编程

它是将开发软件生成的 JEDEC 文件装入(Download 下载)到指定的器件中。可以使用 ispCODE 和 ISP 菊花链下载软件 IDCD, 通过编程电缆下载。

2. ISPSynario 软件的计算机配置要求

运行 ISPSynario 软件的计算机配置要求较低, 并有专门的教学版本既可供实验室大批安装, 也可由学生自由使用, 十分适合教学目的。

- (a) 486 以上 IBM 兼容的 PC 机;
- (b) 80MB RAM, 20MB 硬盘空间;
- (c) 一个 3.5 英寸的软驱, 一个光驱;
- (d) 一个并行打印口;
- (e) Windows 操作系统, 对于 98 或 XP 有不同的安装版本。

3. 软件安装

(a) 首先将含有 ISP Synario 的安装光盘放入光驱中, 进入根目录, 双击文件 Setup.exe, 安装 Synario (根据程序安装界面的要求进行即可)。

(b) 进入安装光盘 pDSPLUS 目录, 双击文件 Setup.exe, 进入 pDS+ Synario 的适配软件 Fitter 安装。

(c) 进入安装光盘 ispDaisy 目录, 双击文件 Setup.exe 即可安装菊花链下载应用程序。


如果不对实际可编程器件进行编程实验, 则可以不进行(b)、(c)步骤。

6.4.2 ISP Synario 设计起点

1. 启动 ISP Synario 软件

双击 ISP Synario 软件的图标或文件名(Synplsi.exe)启动后, 进入主界面, 如图 6.4.1 所示。该界面分为左右两个窗口: 左边窗口(Sources in Project)将显示项目的源文件; 右边窗口(Processes for Current Source)将显示项目和源文件的处理过程(包括检查、优化和编译)。

2. 建立项目文件(.syn)

打开菜单项 File\New Project, 或单击工具栏中图标(新建项目文件)按钮, 如图 6.4.2 所示, 出现对话框 Create New Project。在文件名栏内, 填入项目文件名(myname.syn), 然后单击“OK”按钮。完成后的主界面顶端一行将显示该项目文件名。

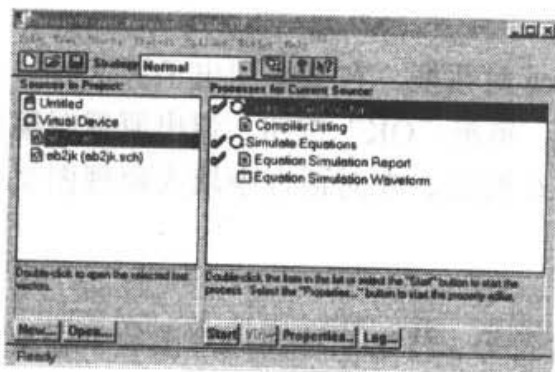


图 6.4.1 ISP Synari 的启动画面

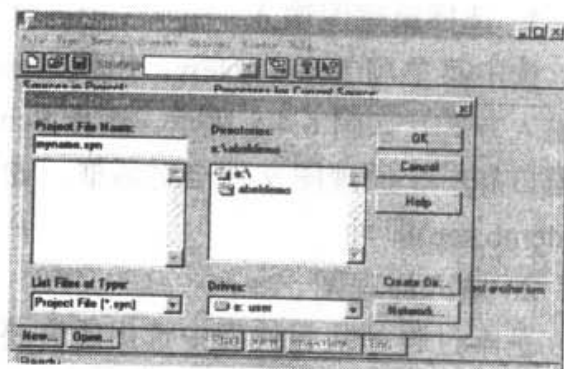


图 6.4.2 新建项目文件窗口

3. 选择器件

在主界面左边窗口中, 双击 Virtual Device 项, 参见图 6.4.3, 出现 Choose Device 对话框, 如图 6.4.4 所示, 在 Device Kit 窗口中选择 ISP Synario Device list 项, 则在下方 Device 窗口中出现器件目录, 按动滚动条选中所希望的器件, 然后单击“OK”按钮即可。

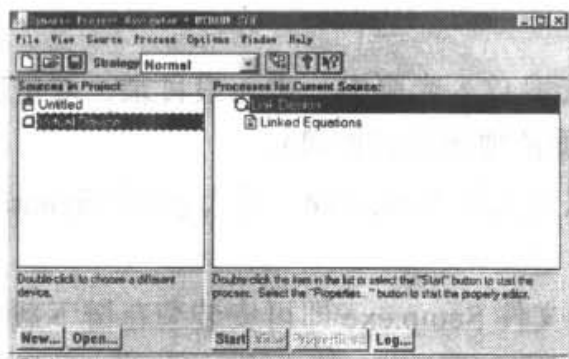


图 6.4.3 新建项目完成后的显示画面

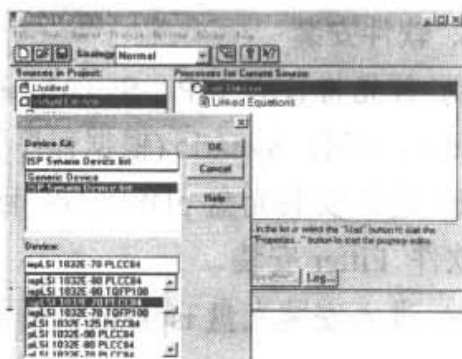



图 6.4.4 选择特定器件窗口

4. 存储信息

选中菜单项 File \ Save 或 Save as, 或者单击工具栏图标  (存盘) 按钮, 即可保存上述文件。若此时选择退出, 则下次进入 ISP Synario 时, 将自动打开前一次的项目文件和相应的设置环境。

一个项目由一个或多个源文件组成。这些源文件可以是原理图文件 (x.sch)、测试向量文件 (x.abv)、ABEL-HDL 文件 (x.abl) 或者是其他文本文件 (x.doc, x.wri, x.txt)。但正如前述, 作为层次设计, 有且仅有一个设计源文件 (x.sch 或 x.abl) 作为顶层文件。若仅有一个设计源文件, 则该文件即为顶层文件。在以下两小节中, 均以一个源文件的设计为示例进行介绍。

6.4.3 ISP Synario 原理图设计

1. 选择原理图输入

选中菜单项 Source \ New, 出现 New Source 对话框, 在框中列出了源文件的几种输入方式, 如图 6.4.5 所示。选择 Schematic, 单击“OK”按钮, 弹出原理图编辑器窗口和一个原理图文件名对话框, 如图 6.4.6 所示, 在对话框中填入原理图文件名 (demo.sch) 即可。

2. 输入原理图

基本过程是从元件库中选出所需要的元件符号, 将它们移到原理图编辑器窗口中的图纸上, 然后用连线将这些元件符号按照设计要求连接起来。可以通过绘图工具 Drawing Toolbar (见图 6.4.7) 或菜单项来完成。

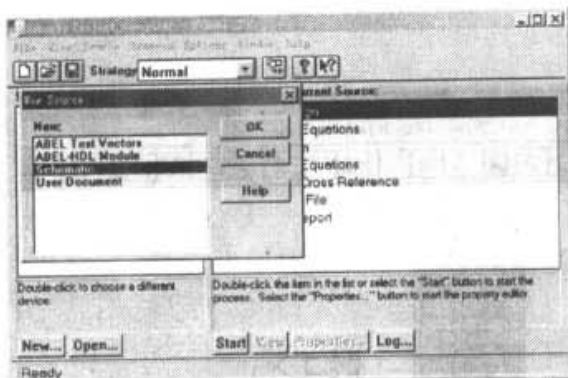



图 6.4.5 原理图输入选择窗口



图 6.4.6 原理图文件命名对话框

(1) 选择元件

单击绘图工具窗口中的元件库  按钮(或打开原理图编辑器窗口中的菜单项 Add \ Symbol...), 出现元件库窗口, 如图 6.4.8 所示。其中包括算术运算电路、门电路、多路选择器、触发器等元件和 I/O 缓冲电路。

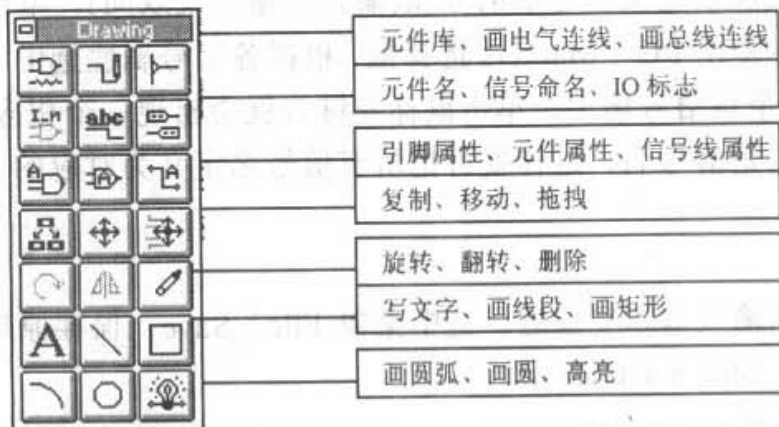




图 6.4.7 绘图工具功能按钮意义说明

对于顶层原理图, 其输入/输出端要与器件的具体引脚相连, 所以原理图中的输入/输出端需与 I/O 缓冲电路连接。I/O 缓冲电路是作为一个元件处理的, 元件符号在元件子库 IOPADS 中。

(2) 电气连接

单击绘图工具中的电气连线按钮 , 将光标放在所需连接的元件一个点上, 单击鼠标左键, 则该点就与连接线上, 然后将光标移至另一个连接点上, 双击鼠标左键, 这时两点间的连接就告完成。重复以上过程, 可把所要求的元件连接线全部画好。

(3) 定义信号命名

单击绘图工具中的信号命名按钮 , 此时在原理图编辑器窗口下方的状态栏中键入信号名。按“Enter”键, 引线名就粘附在光标上随之移动。移至需要命名的 I/O 端(红框上), 单击鼠标左键, 引线名就被安放在那里。重复以上步骤, 可对全部

输入/输出端点进行定义, 如图 6.4.9 所示。

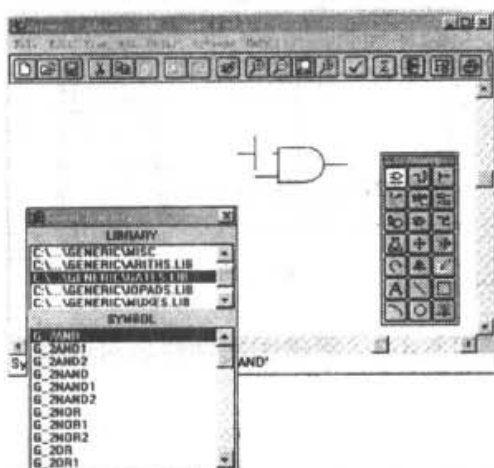


图 6.4.8 元件库窗口

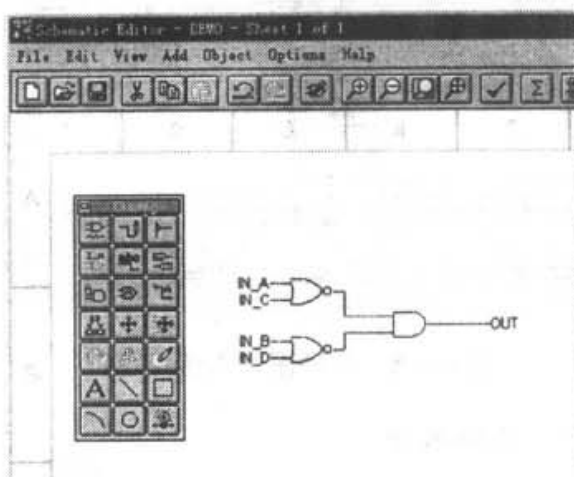



图 6.4.9 信号命名编辑窗口

(4) 定义 I/O 标记

信号命名后, 还应该定义信号名的性质(输入、输出或双向)。单击绘图工具中的 I/O 标记按钮 , 弹出 I/O Markers 选择框, 根据各信号属性选中 Input, Output 或 Bidir。将光标移至该信号端上, 单击鼠标左键, 就会出现一个相应输入/输出或双向标记, 标记框内是信号名。这样就可把所有信号名定义为对应的 I/O 标记, 如图 6.4.10 所示。

(5) 保存原理图

在电路原理图的输入基本完成后, 选中菜单 File \ Save, 保存原理图文件并可退出原理图编辑器, 如图 6.4.11 所示。

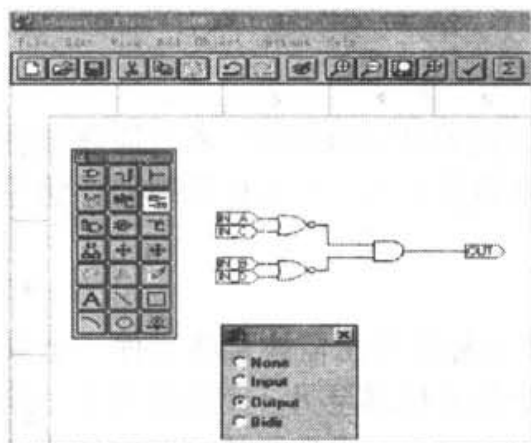


图 6.4.10 定义 I/O 标志窗口

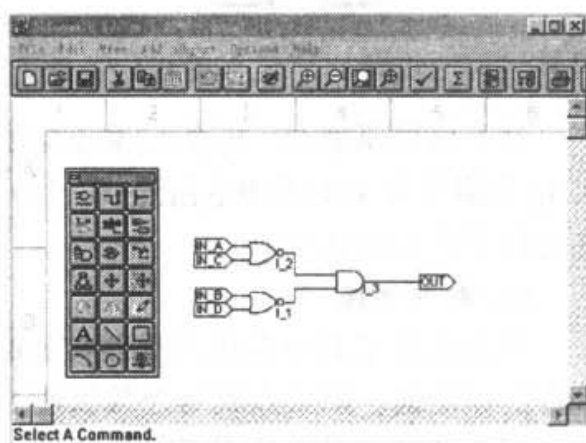


图 6.4.11 完成后的原理图显示画面

3. 原理图编译

电路原理图输入完成后, 回到主界面, 选中原理图源文件名。双击处理过程窗口中的 Compile Schematic 项, 系统对该源文件进行编译。如编译通过, 则在该 Compile Schematic 项前出现一个绿色的记号“√”; 如出现一个黄色的记号“!”,

则表示有警告；如出现一个红色的记号“×”，则表示源文件编辑有错误，编译没通过，并将显示出错误所在及类型，如图 6.4.12 所示。

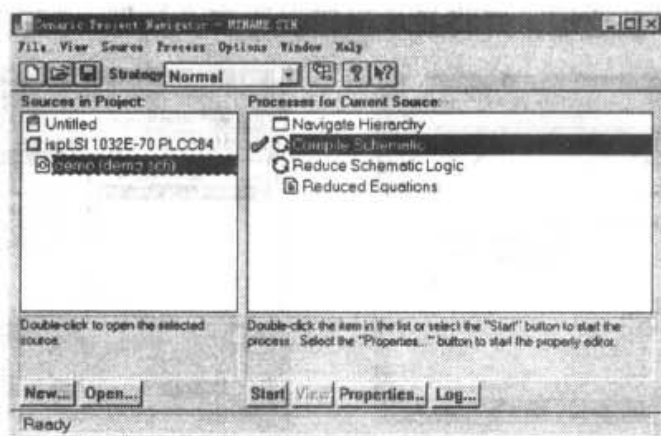


图 6.4.12 原理图编译完成后的显示画面

6.4.4 ISP Synario ABEL-HDL 设计

1. 选择 ABEL-HDL 输入

选中菜单项 Source \ New，出现 New Source 对话框。在框中选择 ABEL-HDL Module，单击“OK”按钮，弹出文本编辑器窗口和一个 New ABEL-HDL Source 对话框，在对话框中填入模块名称(例如 abeldemo)和文件名(例如 abeldemo.abl)。单击“OK”按钮，进入文本编辑器窗口，如图 6.4.13 所示。

2. 编辑 ABEL-HDL 源程序

在文本编辑器窗口中已出现 ABEL-HDL 设计框架和上述键入的模块名。根据设计要求和 ABEL-HDL 语言规则，写入描述电路逻辑关系的文本语句。完成文本编辑后，保存该源文件，进入主界面(工程项目管理器)，如图 6.4.14 和图 6.4.15 所示。

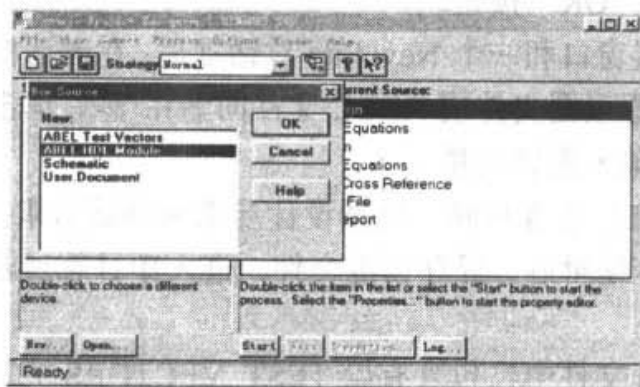


图 6.4.13 选择 ABEL-HDL 设计对话框

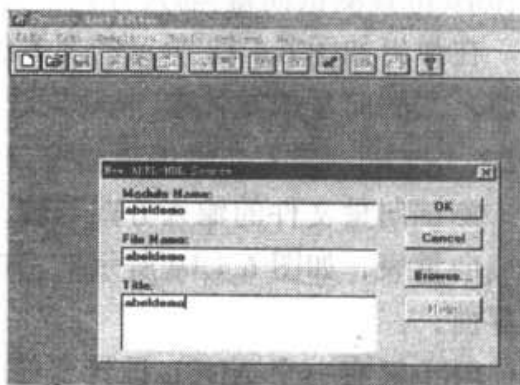


图 6.4.14 ABEL-HDL 模块命名对话框

3. 编译 ABEL-HDL 源文件

在主界面的 Sources in Project 窗口中选中编译的 ABEL 源文件。在右边处理窗口

中双击 Compile Logic 项, 源文件就被编译。如编译通过, 则在 Compile Logic 项前边出现绿色的记号“√”; 或黄色的记号“!”表示有警告; 或红色的记号“×”表示源文件编辑有错误, 编译没通过, 并将显示出错误所在及类型, 如图 6.4.16 所示。

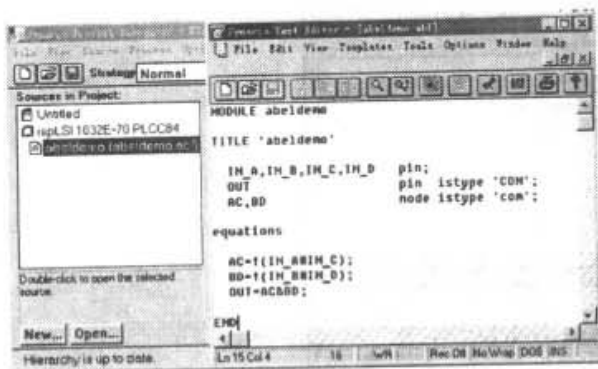


图 6.4.15 ABEL-HDL 模块编辑窗口

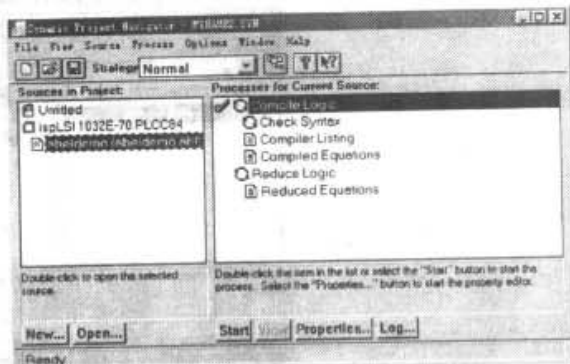


图 6.4.16 ABEL-HDL 模块编译完成的显示画面

6.4.5 设计结果的测试与仿真

利用 ISP Synario 软件进行的设计完成后, 无论是原理图输入还是 ABEL-HDL 源文件输入, 均可以对设计结果作出逻辑功能测试和仿真, 并观察波形。在仿真前需要设计者预先建立好测试向量及测试序列。对 ABEL-HDL 输入文件仿真时, 测试向量即是以关键字 TEST_VECTORS 引导的测试向量段。当仅有一个输入 ABEL 源文件时, 测试向量段可以作为 ABEL 源文件中的一部分, 也可以单独建立一个测试向量文件。在对多个输入源文件构成的层次结构设计或者是对原理图输入文件进行仿真时, 就需要单独建立一个测试向量文件。

1. 建立测试向量

选择菜单项 Source \ New, 弹出如图 6.4.17 所示的对话框。选择 ABEL Test Vectors(仿真测试向量)输入方式, 并单击“OK”按钮。

在弹出如图 6.4.18 所示的文本编辑器窗口和一个 New File 对话框后, 在对话框中填入测试向量的文件名(一般与对应的原理图文件或 ABEL 文件同名), 然后单击“OK”按钮, 进入文本编辑器窗口编辑输入测试向量。

测试向量文件的输入方式与输入 ABEL 语言相同。根据设计要求和测试要求, 完成向量表, 如图 6.4.19 所示。完成文本编辑后, 保存该源文件, 进入项目管理器主界面。

对于仅有一个 ABEL 设计文件, 并且文件中已包含有以 TEST_VECTORS 关键字引导的测试向量段的情况, 在将 ABEL 设计源文件存盘后, 项目管理器左边窗口中自动出现与 ABEL 源文件同名的测试向量源文件。两者后的括号项表明它们来自同一源文件, 参见图 6.4.19。

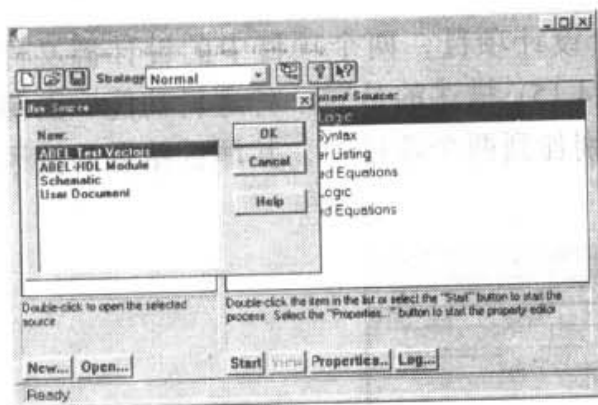


图 6.4.17 建立测试向量对话框

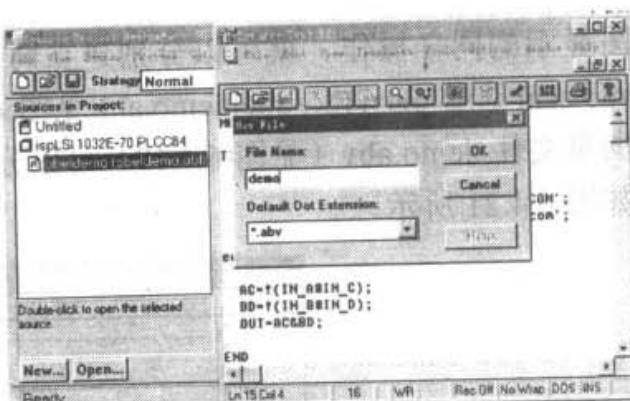


图 6.4.18 输入测试向量的文件名对话框

2. 编译测试向量的源文件

在系统主界面(项目管理器)右边处理窗口中, 双击第一项 Compile Test Vectors (见图 6.4.20) 即对测试向量进行编译。如编译通过, 则在该项前出现绿色“√”标记。

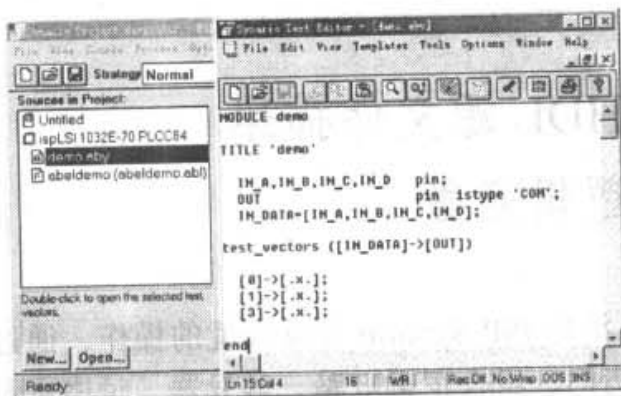


图 6.4.19 测试向量文件编辑窗口

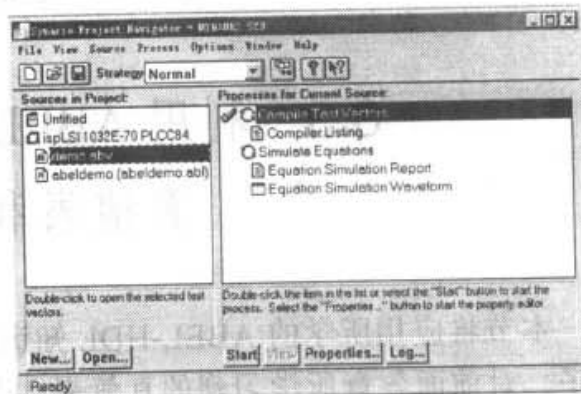


图 6.4.20 测试向量文件编译完成的显示画面

3. 逻辑仿真

参见图 6.4.20, 双击处理窗口中的 Simulate Equations 项, 系统即对设计进行逻辑仿真。仿真通过, 在 Simulate Equations 项的前面出现绿色“√”标记, 否则出现红色“×”标记。通过双击 Equation Simulation Report (仿真报告), 可查看仿真结果或出错情况。

4. 波形图显示

参见图 6.4.20, 双击处理窗口中的 Equation Simulation Waveform 项, 出现波形观察器窗口, 如图 6.4.21 所示。接着选择菜单项 Edit\Show, 弹出一个对话框, 选择需要显示波形的结点或输入/输出端, 单击“Show”按钮, 即可在波形观察器中显示其波形。

从 6.4.3 节到 6.4.5 节, 我们用文字配合屏幕截图方式分步介绍了在 ISP Synario 软件中采用原理图和 ABEL-HDL 进行的设计描述、编译和仿真过程。读者按屏幕

截图显示的同样操作, 就可以分别建立两个设计项目, 两个项目中的设计源文件 Demo.sch(见图 6.4.11)、abeldemo.abl(见图 6.4.15) 描述的是同一种逻辑功能。而测试向量文件 demo.abv(见图 6.4.19) 可以分别加到两个项目中, 其测试结果的波形图如图 6.4.21 所示。

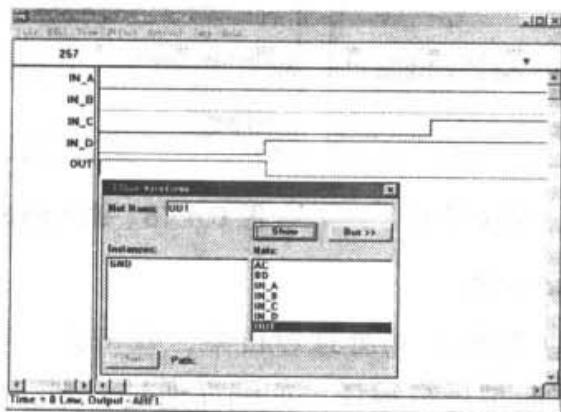


图 6.4.21 测试向量波形显示窗口

6.5 利用 ABEL-HDL 源文件描述 真值表和逻辑函数

本节将应用所学的 ABEL-HDL 知识, 结合 ISP Synario 开发环境的操作, 通过例子, 对前面各章所学习到的真值表、逻辑门、组合逻辑电路、触发器、时序逻辑电路等, 在新的角度上重新归纳和认识。通过学习, 使读者对于采用硬件描述语言进行逻辑设计有更直观的体会。为明确起见, 所有例子都是完整的源程序, 读者可以阅读并直接上机验证。

数字逻辑的基本工具是真值表和逻辑函数。而 ABEL-HDL 的真值表和逻辑运算式, 正是这种基本工具的表达形式。两者之间的直接对应关系, 在语言定义中十分清楚, 所以此处不再展开, 仅以若干例子说明。

例 6.5.1 试用 ABEL-HDL 表示如下逻辑函数:

$$(1) F_1(ABC) = \sum m(0, 1, 4, 5)$$

$$(2) F_2(ABC) = \prod M(1, 5, 6)$$

$$(3) F_3(ABC) = \bar{A}\bar{B}C + A\bar{B}\bar{C} + A\bar{B}C + ABC$$

解 对于以最大项或最小项形式表达的逻辑函数, 在 ABEL-HDL 中以真值表来表示是十分方便的。源程序如下:

```
MODULE ex6_5_1
declarations
```

```

A,B,C      pin;           //定义输入变量
F1,F2,F3    pin istype 'com'; //定义输出变量为组合型
IN_D=[A,B,C];           //定义输入变量的集合

truth_table
// ([输入集合] -> [输出])
([IN_D] -> [F1,F2,F3])    //真值表头
    [0] -> [1,1,0];
    [1] -> [1,0,1];
    [2] -> [0,1,0];
    [3] -> [0,1,0];
    [4] -> [1,1,1];
    [5] -> [1,0,1];
    [6] -> [0,0,0];
    [7] -> [0,1,1];

END

```

我们看到，对于这个3输入3输出的逻辑函数，真值表定义了3输入的8种组合与3个输出间的全部逻辑对应关系。因此，这是一个完整的逻辑函数定义。

例 6.5.2 试用 ABEL-HDL 表示具有随意项的函数

$$F_4(ABC) = \sum m(0,1,5) + \sum \phi(2,3,6)$$

解 当表示一个不完全定义的逻辑函数时,ABEL-HDL 语言可以使用指示字 @Dcset, 事实上, @Dcset 指示了 ABEL 编译程序的优化结果。源程序如下:

```

MODULE ex6_5_2
declarations
    A,B,C      pin;
    F4          pin istype 'com';
    IN_D=[A,B,C];

    @DCset      //声明一个不完全定义的真值表, 未指定的项为随意项。

truth_table ([IN_D] -> [F4])    //真值表描述
    [0] -> [1];
    [1] -> [1];
    [4] -> [0];
    [5] -> [1];
    [7] -> [0];

END

```

例 6.5.3 试用 ABEL-HDL 表示逻辑门: 与、与非、或非、异或的功能。

解 ABEL-HDL 本质上是 EDA 的设计工具, 并不用来作逻辑分析, 但可以利用

用测试向量，画出逻辑输入和逻辑输出间的波形，达到分析目的。

设输入为 A 和 B，逻辑门的功能以逻辑运算来体现。做逻辑运算 $F_5=A*B$ ； $F_6=\overline{A*B}$ ； $F_7=\overline{A+B}$ ； $F_8=A\oplus B$ 。做运算的测试，此时的测试结果应取任意值，即测试结果是实际方程运算的结果。源程序如下：

```
MODULE ex6_5_3
    A,B                pin;
    F5,F6,F7,F8        pin istype 'com';
    X=.X.;
equations              //逻辑方程描述
    F5=A&B;            F6=!(A&B);    F7=!(A#B);    F8=A$B;
test_vectors           //测试向量开始
    ([A,B]->[F5,F6,F7,F8])      //测试表头
    [0,0]->[ X, X, X, X];        //测试结果取任意值
    [1,1]->[ X, X, X, X];
    [1,0]->[ X, X, X, X];
    [0,1]->[ X, X, X, X];
END
```

仿真波形如图 6.5.1 所示，可见 F5、F6、F7 和 F8 分别是输入信号 A、B 的与、与非、或非、异或。

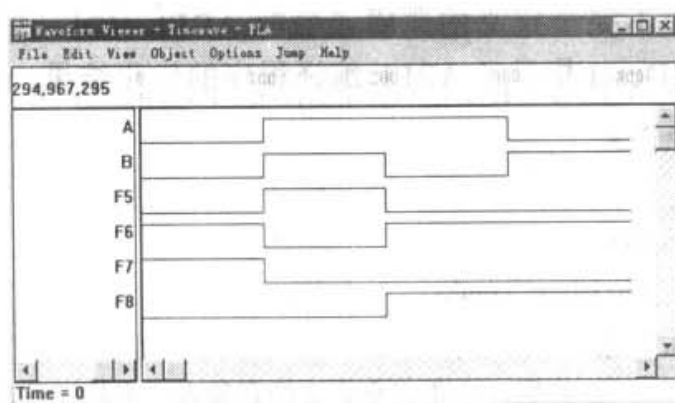


图 6.5.1 逻辑运算的仿真结果

例 6.5.4 试用 ABEL-HDL 实现下列逻辑函数：

$$F_9 = A(A+B+\overline{C})(\overline{A}+C+D)(E+\overline{C}\overline{D})$$

$$F_{10} = A+B+C+\overline{D}+\overline{\overline{E}}$$

$$F_{11} = (A\oplus B)+\overline{A}C$$

解 直接以逻辑方程来实现。提请注意两点，在 EDA 工具中，没有人工的化简过程，化简是在设计的编译和综合阶段，由开发工具自动完成的；在书写逻辑方程时，必须注意运算符的优先级，没有把握时应加上括号以保证原定的运算顺序。

源程序如下:

```

MODULE ex6_5_4
    A,B,C,D,E      pin;
    F9,F10,F11      pin istance 'com';
equations           //逻辑方程描述
    F9 = A&(A#B#!C)&(!A#C#D)&(E#!(C#D));
    F10=A#!(B#C#!(D#!E));
    F11=(A$B)#(!A&C);
End

```

*6.6 利用 ABEL-HDL 源文件描述组合逻辑电路

在前面各章中学习组合逻辑电路和时序逻辑电路时,采用的是传统的分析和设计方法,即以门电路、触发器、组合逻辑集成部件、时序逻辑集成部件为基础,以电路图为依托,以卡诺图和功能表为工具来进行的。下面通过示例介绍以 ABEL 语言为工具来描述的方法,读者可以从中体会语言工具的优势和特点。

组合电路的特点是,任一时刻输出信号的稳态值,仅取决于该时刻的输入信号,而与输入信号作用之前电路所处的状态无关。因此,在电路实现上,它是由各种逻辑门构成,无触发器等记忆元件,也没有输入/输出之间的反馈。用语言实现时,主要是用真值表和逻辑函数。真值表直截了当,逻辑函数表示要借助所学的分析或化简方法,也常用 when—then—else 结构语句直接表述电路功能。下面以部分典型的组合逻辑电路单元为例进行介绍。

例 6.6.1 一个具有编码允许的 8-3 线优先编码器的功能表如表 6.6.1 所示,试用 ABEL-HDL 描述其功能,并用测试向量实现仿真波形显示。

表 6.6.1 8-3 线优先编码器的功能表

EN	输 入								输 出		
	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	Y ₂	Y ₁	Y ₀
0	×	×	×	×	×	×	×	×	0	0	0
1	1	×	×	×	×	×	×	×	1	1	1
1	0	1	×	×	×	×	×	×	1	1	0
1	0	0	1	×	×	×	×	×	1	0	1
1	0	0	0	1	×	×	×	×	1	0	0
1	0	0	0	0	1	×	×	×	0	1	1
1	0	0	0	0	0	1	×	×	0	1	0
1	0	0	0	0	0	0	1	×	0	0	1
1	0	0	0	0	0	0	0	1	0	0	0

解 对于如上优先编码功能的语言描述,可以有以下3种方法。

(1) 直接从功能表,通过 when—then—else 语句表述优先功能。源程序如下:

```

MODULE ex6_1_11                                // 8-3 线优先编码器
declarations
    D7,D6,D5,D4,D3,D2,D1,D0    pin;    //定义输入变量
    COD_EN    pin;                //定义编码控制输入
    Y2,Y1,Y0    pin istype 'com';    //定义输出变量为组合型
    IN_D=[D7,D6,D5,D4,D3,D2,D1];    //定义输入变量的集合
    OUT_Y=[Y2,Y1,Y0];              //定义输出变量的集合
    X=.x.;
equations                                        //逻辑方程描述
    when COD_EN==0 then OUT_Y=0;
    else {
        when D7 then OUT_Y=7;
        else when D6 then OUT_Y=6;
        else when D5 then OUT_Y=5;
        else when D4 then OUT_Y=4;
        else when D3 then OUT_Y=3;
        else when D2 then OUT_Y=2;
        else when D1 then OUT_Y=1;
        else when D0 then OUT_Y=0;}
test_vectors ([COD_EN,IN_D,D0] ->[OUT_Y])
    [0.^H74,X] ->[X];
    [1.^H40,X] ->[X];                // ^H40=^B1000000
    [1.^H41,X] ->[X];  [1.^H61,X] ->[X];    // ^D61=^B1100001
    [1.^H20,X] ->[X];
    [1.^H10,X] ->[X];
    [1.^H08,X] ->[X];
    [1.^H04,X] ->[X];
    [1.^H02,X] ->[X];
    [1.^H01,X] ->[X];
    [1.^H00,1] ->[X];
END

```

其仿真结果如图 6.6.1 所示。

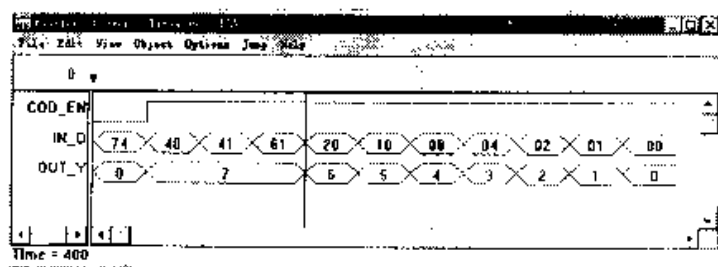


图 6.6.1 例 6.6.1 设计结果测试波形图

(2) 利用逻辑函数来描述也是可行的。用第 3 章所学的组合电路灵活设计法, 不难写出如下输出函数方程:

$$Y_2 = EN \cdot (D_7 + \bar{D}_7 D_6 + \bar{D}_7 \bar{D}_6 D_5 + \bar{D}_7 \bar{D}_6 \bar{D}_5 D_4)$$

$$Y_1 = EN \cdot (D_7 + \bar{D}_7 D_6 + \bar{D}_7 \bar{D}_6 \bar{D}_5 \bar{D}_4 D_3 + \bar{D}_7 \bar{D}_6 \bar{D}_5 \bar{D}_4 \bar{D}_3 D_2)$$

$$Y_0 = EN \cdot (D_7 + \bar{D}_7 \bar{D}_6 D_5 + \bar{D}_7 \bar{D}_6 \bar{D}_5 \bar{D}_4 D_3 + \bar{D}_7 \bar{D}_6 \bar{D}_5 \bar{D}_4 \bar{D}_3 \bar{D}_2 D_1)$$

源程序如下:

```

MODULE ex6_1_12                                // 8-3 线优先编码器
declarations
    D7,D6,D5,D4,D3,D2,D1,D0    pin;           //定义输入变量
    COD_EN    pin;              //定义编码控制输入
    Y2,Y1,Y0    pin istype 'com';           //定义输出变量为组合型
    IN_D=[D7,D6,D5,D4,D3,D2,D1];           //定义输入变量的集合
    OUT_Y=[Y2,Y1,Y0];             //定义输出变量的集合
    X=x.;
equations                                     //逻辑方程描述
    Y2=COD_EN&(D7#!D7&D6#!D7&!D6&D5#!D7&!D6&!D5&D4);
    Y1=COD_EN&(D7#!D7&D6#!D7&!D6&!D5&!D4&D3
                #!D7&!D6&!D5&!D4&!D3&D2);
    Y0=COD_EN&(D7#!D7&!D6&D5#!D7&!D6&!D5&!D4&D3
                #!D7&!D6&!D5&!D4&!D3&!D2&D1);
END

```

第(2)解中的测试向量部分与 (1) 解完全相同, 输出波形也相同, 故省略了。请读者注意, 程序中 Y_1 语句在源程序中是一行写完的 (ABEL-HDL 没有分行语句的规定), 同样 Y_0 语句也如此。

(3) 还可以直接利用功能表作为真值表, 但应注意, 由于功能表中有 8 个输入变量, 却只有 8 行输出定义, 因此, 若采用 @Dcset 做不完全真值表定义, 则在语言的编译优化过程中, 将以逻辑最简为实现标准, 其结果可能会与优先编码的要求相矛盾。如果要切实保证优先编码, 就需要将上表扩充为完全定义的真值表, 即 2^8

= 256 行, 显然这样做其真值表是很繁琐冗长的。读者可自己一试。

例 6.6.2 试用 ABEL-HDL 设计实现 4 位二进制码到 7 段显示的显示译码器。

解 对本例而言直接用真值表设计比较简洁和直观。

```

MODULE ex6_6_2                                //4 位二进制码到 7 段显示译码器
declarations
    D8,D4,D2,D1      pin;                    //定义输入变量
    a,b,c,d,e,f,g     pin istype 'com';      //定义 7 段输出变量为组合型
    IN_D=[D8,D4,D2,D1];                      //定义输入变量的集合
    X=x.;
truth_table ([IN_D] => {a,b,c,d,e,f,g}) //真值表描述
    [^H0] => [1,1,1,1,1,1,0]; //显示 0
    [^H1] => [0,1,1,0,0,0,0]; //显示 1
    [^H2] => [1,1,0,1,1,0,1]; //显示 2
    [^H3] => [1,1,1,1,0,0,1]; //显示 3
    [^H4] => [0,1,1,0,0,1,1]; //显示 4
    [^H5] => [1,0,1,1,0,1,1]; //显示 5
    [^H6] => [1,0,1,1,1,1,1]; //显示 6
    [^H7] => [1,1,1,0,0,0,0]; //显示 7
    [^H8] => [1,1,1,1,1,1,1]; //显示 8
    [^H9] => [1,1,1,1,0,1,1]; //显示 9
    [^HA] => [1,1,1,0,1,1,1]; //显示 A
    [^HB] => [0,0,1,1,1,1,1]; //显示 B
    [^HC] => [1,0,0,1,1,1,0]; //显示 C
    [^HD] => [0,1,1,1,1,0,1]; //显示 D
    [^HE] => [1,0,0,1,1,1,1]; //显示 E
    [^HF] => [1,0,0,0,1,1,1]; //显示 F
END

```

6.7 利用 ABEL-HDL 源文件描述时序逻辑电路

时序电路由组合电路和存储电路构成。其特点是时序电路的输出不仅取决于该时刻的输入信号, 还与该时刻之前电路的状态有关。因此, 在电路实现上, 它是由各种逻辑门加上触发器等构成, 具有输入/输出之间的反馈。用语言实现时, 既要用真值表和逻辑函数, 更多时候要用状态图。当一个时序逻辑电路以状态图描述时,

状态的定义及其化简是必须的,这可以用第5章所学的知识。前节已经提到过,ABEL-HDL中用寄存器来表示存储元件,而在此表达当中,点扩展名的使用是关键,学习中尤其要注意。下面举例说明。

例 6.7.1 用 ABEL-HDL 实现 4 位双向移位寄存器 74LS194。

解 74LS194 是一个通用性很强的双向移位寄存器集成芯片。在时钟脉冲作用下,通过两个功能选择输入端 S_1 、 S_0 ,可以实现左右移位和并行置数的功能。还可以实现保持功能。此外,还具有一个异步的清除端 RD(功能表参见第5章)。本例通过 when 语句来列举功能选择输入端的各个功能,利用方程块加强程序的可读性。实现的源程序如下:

```

MODULE ex6_7_1
TITLE '74LS194: a 4bit bi-dir universal shift_reg'
declarations
    DSR,DSL,D0,D1,D2,D3,S0,S1,RD,CP pin;
    Q0,Q1,Q2,Q3 pin istype 'buffer.reg_D';
    Din=[D3..D0];           //并行置数输入端集合
    Shift_regQ=[Q3..Q0];    //移位寄存器输出端集合
    Select=[S1,S0];         //功能选择输入端集合
    C,X=.c...x.;
equations
    Shift_regQ.clk=CP;       //移位寄存器集合的时钟
    Shift_regQ.rc=!RD;       //移位寄存器异步清除信号,低电平有效
    when Select==0 then Shift_regQ.D=Shift_regQ;           //保持
    when Select==1 then {Q3.D=Q2;Q2.D=Q1;Q1.D=Q0;Q0.D=DSR;} //右移
    when Select==2 then {Q0.D=Q1;Q1.D=Q2;Q2.D=Q3;Q3.D=DSL;} //左移
    when Select==3 then Shift_regQ.D=Din;                 //置数
test_vectors
    ([CP,RD,Select,DSR,DSL,Din] ->[Shift_regQ])
    [X,0,X,X,X,X] ->[X];           //测试异步清零
    [C,1,3,X,X,8] ->[X];           //测试同步置 D3 为 1
    [C,1,3,X,X,4] ->[X];           //测试同步置 D2 为 1
    [C,1,3,X,X,2] ->[X];           //测试同步置 D1 为 1
    [C,1,3,X,X,1] ->[X];           //测试同步置 D0 为 1
    [C,1,0,X,X,X] ->[X];           //测试同步保持
    @repeat 6 {[C,1,1,0,X,X] ->[X];} //测试右移且 DSR=0

```

```
@repeat 6 {[C,1,1,1,X,X]->[X]:} //测试右移且 DSR=1
@repeat 6 {[C,1,2,X,0,X]->[X]:} //测试左移且 DSL=0
@repeat 6 {[C,1,2,X,1,X]->[X]:} //测试左移且 DSL=1
```

END

测试结果如图 6.7.1 所示。

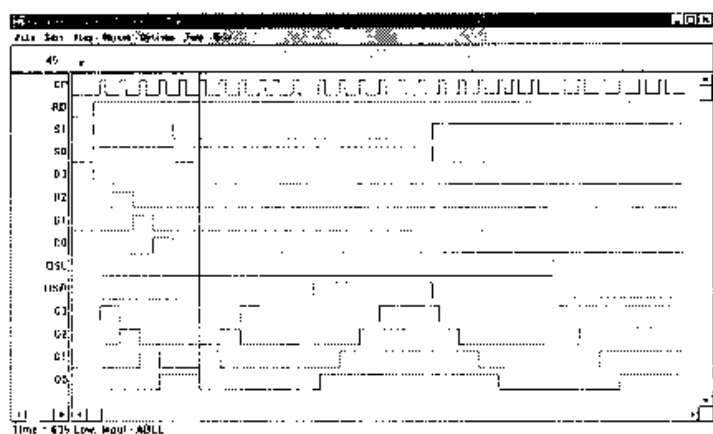


图 6.7.1 例 6.7.1 设计结果测试波形图

例 6.7.2 采用原理图与 ABEL-HDL 混合设计图 6.7.2 所示电路。其中 4 位双向移位寄存器直接应用例 6.7.1 的设计结果。

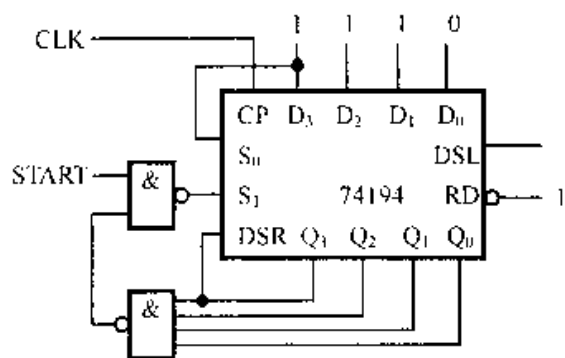


图 6.7.2 例 6.7.2 电路图

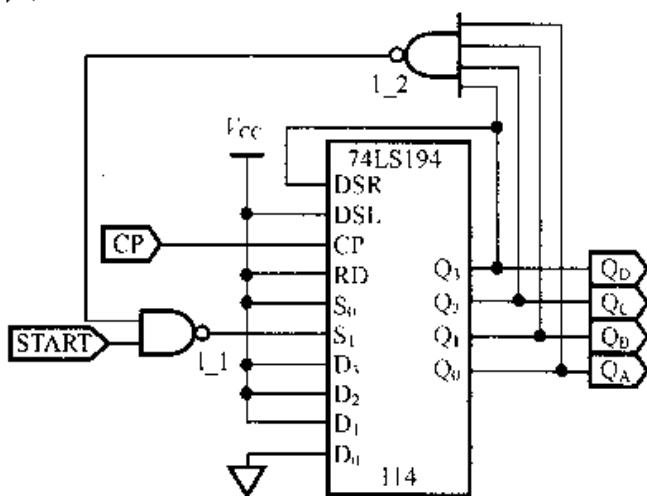


图 6.7.3 例 6.7.2 的顶层原理图设计

解 本例是一个混合设计，也是一个层次设计。可以用图 6.7.3 所示的原理图作为顶层，其中的 4 位双向移位寄存器 74194 作为一个较低层模块来完成解答。实现的过程如下：

(1) 打开 ISP Synario 系统，建立一个新项目 “app74194.syn”；

(2) 在顶层建立原理图文件 “app74194.sch”，在原理图编辑器窗口中，选中菜单项 Add\New Block Symbol，在弹出的对话框中输入如下内容：

Block name: _74LS194

Input Pins: D3,D2,D1,D0,DSL,DSR,RD,CP,S1,S0

Output Pins: Q3,Q2,Q1,Q0

即可在 app74194.sch 中建立起模块 _74LS194, 并具有以上输入/输出引脚。

(3) 继续按图 6.7.3 所示完成原理图 app74194.sch 后存盘, 如图 6.7.4 所示。

(4) 在主界面窗口选中菜单项 Add\New, 建立 ABEL-HDL Module, 其模块名必须是 (2) 中建立的 Block name。本例就是 _74LS194, 并且该模块的输入、输出 pin 的定义也必须和 (2) 中的相同。

```
MODULE _74LS194
```

```
TITLE '74LS194: a 4bit bi-dir universal shift_reg '
```

```
DSR,DSL,D0,D1,D2,D3,S0,S1,RD,CP pin;
```

```
Q0,Q1,Q2,Q3 pin istype 'buffer,reg_D';
```

(5) 参照例 6.7.1 继续完成模块 _74LS194 后存盘。

(6) 在主界面窗口选中菜单项 Add\New, 选择 ABEL Test Vectors, 建立独立的测试向量文件。完成后存盘。

```
MODULE APP74194
```

```
TITLE 'APP74194'
```

```
CP,START pin ;
```

```
QD,QC,QB,QA pin istype 'buffer,reg_D';
```

```
qout=[QD,QC,QB,QA];
```

```
X=.x.;
```

```
test_vectors ([CP,START] ->[qout])
```

```
[.C.,0] ->[X];
```

```
@repeat 20 {[.C.,1] ->[X];}
```

```
END
```

(7) 分别编译顶层原理图、低层 74LS194 (屏幕图中为 74ls194) 模块和测试向量, 如图 6.7.4 所示。

(8) 最后的测试波形如图 6.7.5 所示。

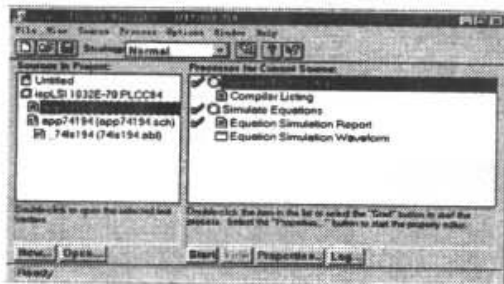


图 6.7.4 例 6.7.2 的层次结构

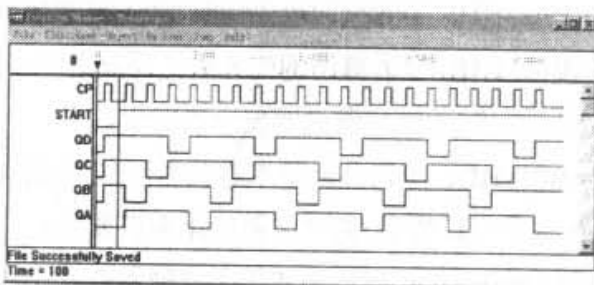


图 6.7.5 例 6.7.2 的测试波形

小 结

EDA 技术的发展和广泛应用是现代电子技术进步的最明显的特征。其中,硬件描述语言又是现代 EDA 技术的重要核心。它使设计者能够以“自顶向下”的设计方法,在系统级的层次上描述系统行为,确定功能框架,仿真测试性能,直至综合优化电路实现。

有多种被广泛使用的硬件描述语言,VHDL、Verilog HDL 已经成为 IEEE 国际标准,得到所有 EDA 软件平台的支持。ABEL-HDL 简单明了,便于掌握,适于教学和入门。

学习一种语言,首先要熟悉它的语言规定,包括基本语法的要求和限制,也包括程序的结构要求和限制。而对于 ABEL-HDL 等硬件描述语言,更要在硬件角度上,学习和掌握其特有的规定。对于逻辑器件和逻辑电路的描述,是本章学习的重点。

学习语言要和上机实践相配合。本章所用 ABEL-HDL 的开发平台是 ISP Synario 软件。在 Synario 系统软件上,可以输入电路原理图、编写 ABEL 源程序、编辑测试向量文件,进行从设计—测试—修改—完成等完整的实际操作过程。上机实践是掌握本章主要内容的重要辅助手段。

本章在已具备数字逻辑基本知识的基础上学习。通过用 ABEL-HDL 语言设计的典型组合逻辑和时序逻辑电路的例子,可以与传统的电路设计方法进行对比,加深对硬件描述语言的认识和体会。

层次设计是现代 EDA 设计的特点,也是硬件描述语言的优势。但认识和掌握它需要较为深入的学习,还需要一定的工程设计实践。

思考题和习题

- 6.1 现代 EDA 技术发展的主要阶段如何?各阶段的特点如何?
- 6.2 归纳各个主要的硬件描述语言的基本特点。
- 6.3 查找因特网上部分与现代 EDA 技术和硬件描述语言相关的资料并归纳小结。
- 6.4 ABEL-HDL 中的标识符如何规定?关键字与标识符有何区别与联系?
- 6.5 ABEL-HDL 中运算符的优先级如何规定?
- 6.6 什么是集合?试举例说明集合的定义。
- 6.7 ABEL-HDL 中有几种基本逻辑器件?如何描述基本逻辑器件?
- 6.8 什么是点扩展名?为什么要使用点扩展名?
- 6.9 用 ABEL-HDL 如何描述如下逻辑器件:

(1) 3 输入与非门; (2) 带使能端的三态 3 输入与非门,使能低电平有效;

- (3) 带有异步清 0 和异步置 1 的 JK 触发器；
- (4) 4 位 D 触发器，带有共同的异步清 0 端，各自独立的异步置 1 端。
- 6.10 一个 ABEL-HDL 模块由哪些部分组成？各使用什么关键字？
- 6.11 ABEL-HDL 程序中定义段的作用是什么？解释下列定义段中的关键字：
- ```

pin: node: istype:

```
- 6.12 解释属性的意义和作用。
- 6.13 ABEL-HDL 程序中逻辑描述段的作用是什么？逻辑描述段中如何应用状态码表示逻辑功能？
- 6.14 ABEL-HDL 程序中向量测试段的作用是什么？
- 6.15 试用 ABEL-HDL 验证异或运算的结果。设  $f_1, f_4, \dots, f_2, f_0 = 0HAF0A$ , (1) 作集合  $set\ 1 = \{f_1, f_0\}$ ；  
(2) 作集合  $set\ 2 = \{f_0, f_5\}$ ；则
- ```

out 1 = set 1[7..4]
out 2 = set 2[7..4]
out 3 = set 1[4..7]
out 4 = set 1(0)
out 5 = set 1(15)

```
- 分别等于何值？
- 6.16 试用 ABEL-HDL 写出如下逻辑函数式
- (1) $F = \overline{A}B + \overline{A}CD + \overline{B}C + \overline{D}$
- (2) $F(ABCD) = \sum m(0, 2, 3, 5, 6, 7, 8, 9) + \sum \phi(10, 11, 12)$
- 6.17 试用 ABEL-HDL 与真工具证明 $A \oplus B \oplus C = A \odot B \odot C$ 。
- 6.18 试用 ABEL-HDL 设计一个 4 输入 4 输出三态逻辑电路。当控制信号 $C=0$ 时，输出状态与输入状态相反； $C=1$ 时，输出状态与输入状态相同。三态控制端 \overline{OE} 低电平有效。给出仿真波形。
- 6.19 试用 ABEL-HDL 设计一个 4 位二进制全加器。给出仿真波形。思考：若要求实现进位超前，应如何设计？
- 6.20 用 ABEL-HDL 分别设计并仿真 D 触发器、JK 触发器和 T 触发器。
- 6.21 试用 ABEL-HDL 描述如图 P6.1(a)所示电路，输入波形如图 P6.1(b)所示，设初始状态 Q 为 0，画出输出 Z 的波形。

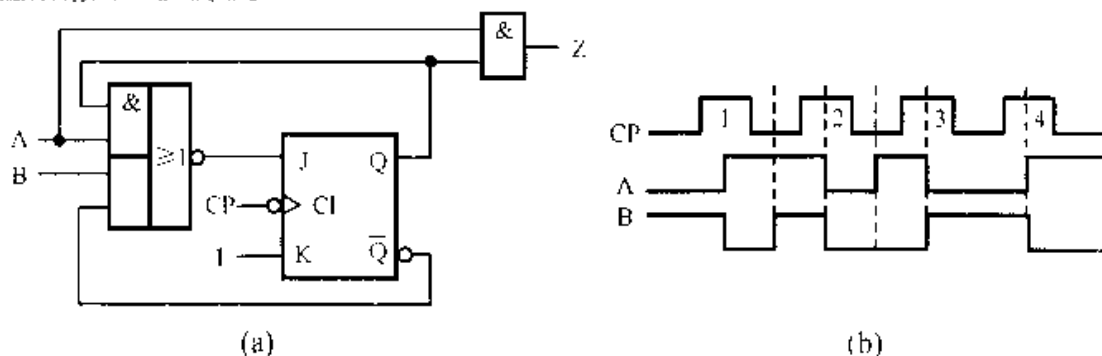


图 P6.1

- 6.22 图 P6.2 是某时序电路的状态转换图, 设电路的初始状态为 **01**, 并设输入序列是 **100110** (自左至右)。试用 ABEL-HDL 对该逻辑功能进行描述, 并仿真求出该电路输出的序列。
- 6.23 试用 ABEL-HDL 实现 4 位数值比较器 74LS85 的逻辑功能。
- 6.24 试以 ABEL-HDL 实现 4 位加法计数器 74LS161。
- 6.25 试以原理图为顶层模块, 采用 ABEL-HDL 实现的 74LS161 为底层模块, 设计一个模 24 的同步计数器。
- 6.26 试以原理图为顶层模块, 采用 ABEL-HDL 分别实现 74LS161 模块和 74LS138 模块 (该模块可参考例 6.6.2) 为底层模块, 设计实现图 P6.3 所示电路。并仿真当 $Q_D Q_C Q_B Q_A = 1111$ 时, 电路输出的 F_1 、 F_2 、 F_3 波形 (画出 12 个 CP 脉冲)。

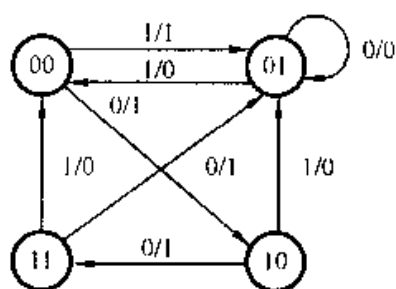


图 P6.2

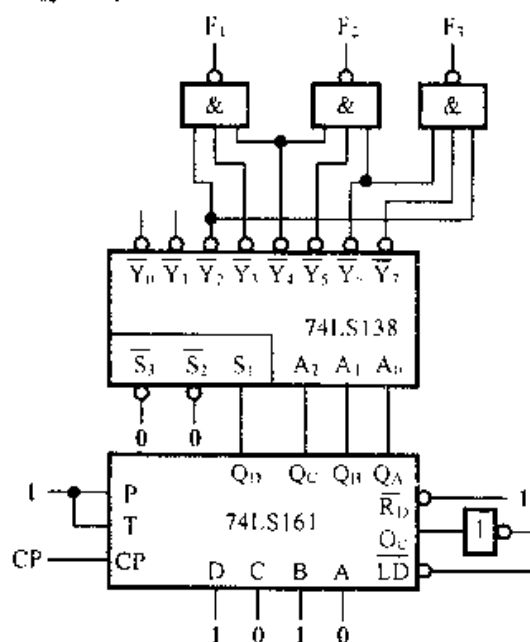


图 P6.3

第 7 章 半导体存储器

内容提要 在大规模集成 LSI 和超大规模集成 VLSI 器件中应用最为广泛的是存储器，它的主要功能是存储信息，自 20 世纪 70 年代以来已经在计算机、数字通信、数据采集与处理、工业自动控制以及人工智能等学科领域获得广泛应用。本章重点介绍各类半导体存储器的结构、工作原理、使用方法及其应用，同时还介绍了存储容量的扩展方法。

7.1 随机存储器(RAM)

存储器分为随机存储器 RAM 和只读存储器 ROM。RAM 可以随机地写入或读出数据，而 ROM 的主要功能是反复读取所存储的内容，不能随意更改内容。

RAM 又可分为 3 大类，其特性与应用范围如表 7.1.1 所示。它们在集成度、速度、功耗及成本等指标上有较宽的覆盖面，因而可以满足不同应用领域的需要。本节先介绍动态 MOS 反相器，它是构成大容量 RAM 的基本单元，然后介绍 RAM 的电路结构和存储单元，最后介绍典型产品的使用方法以及实用中应注意的技术问题。

表 7.1.1 RAM 的特性表

类 型		存储单元	主要指标				特点、应用
			速度	容量	功耗	成本	
双极型 RAM	TTL RAM	双发射极存储单元 肖特基耦合存储单元	高	小	大	高	速度最高，存取时间可小于 10 ns，大型、高速计算机 内存存储器
	PL RAM	PL 存储单元	中	中	中	中	
	ECL RAM	ECL 存储单元	最高	小	大	高	
MOS 型 RAM	静态 RAM	六管静态存储单元	低	大	小	低	外围电路少；控制系统
	动态 RAM	四管、三管、单管存储单元	低	最大	小	最低	外围电路复杂，控制信号 多；计算机外存储器
	CMOS RAM	CMOS 存储单元	低	较大	最小	较低	数据无易失性（电池供 电）；宇航、军工设备
CCD 型 RAM		CCD 存储单元	最低	最大	小	中	结构简单，容量大；大容 量串行存储器

7.1.1 RAM 的基本结构

1. RAM 结构框图

RAM 结构框图如图 7.1.1 所示。它主要由存储矩阵（又称存储体）、地址译码器和读/写电路 3 部分组成。存储矩阵是存储器的主体，其他两部分称为存储器的外围电路。

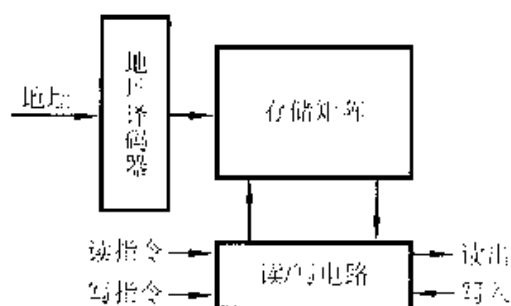


图 7.1.1 RAM 结构框图

存储矩阵是由许多存储单元有规则地排列构成的，每一个存储单元可以存储一位二进制码。对每个存储单元用二进制码编号，即构成存储单元的地址，为了选中给定单元的地址，可以采用一元寻址（又称为字结构或单译码结构），或者二元寻址（又称位结构或双译码结构）。其逻辑框图如 7.1.2 所示，图中，存储矩阵包含 16 个存储单元，所以，需要 16 个地址。图 7.1.2 (a) 是一元寻址，由 4 位地址码便可构成 16 个地址，即 16 条字线，每条字线为 1 电平时便选中相应存储单元。被选中单元通过数据线与读/写电路连接，便可实现对该单元的读出或写入。

图 7.1.2(b) 为二元寻址逻辑图，它有 X 和 Y 两个地址译码器。每个存储单元由 X 字线和 Y 字线控制，只有在 X 和 Y 字线都被选中时才能对该单元读出或写入。二元寻址可以大大减少字线数量。所以，在大容量 RAM 中均采用二元寻址。

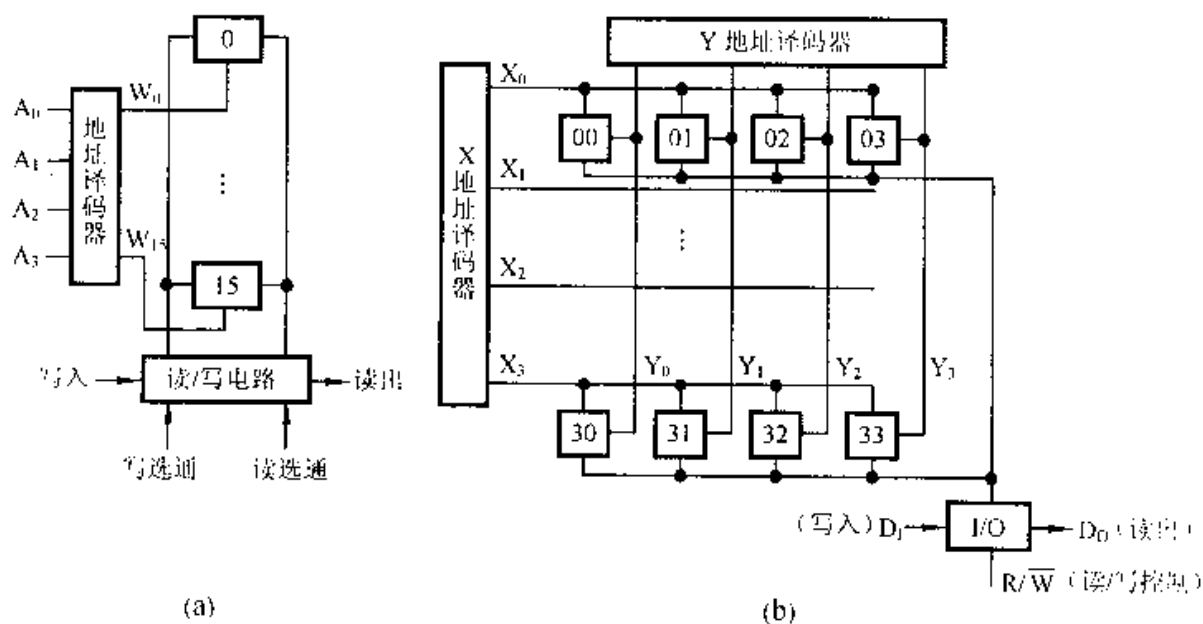


图 7.1.2 RAM 寻址方式

(a) 一元寻址 (b) 二元寻址

2. 存储单元

(1) 双极型存储单元

双极型存储单元如图 7.1.3 所示。 T_1 和 T_2 的一对发射极接同一条字线； T_1 的集电极接数据线 \bar{D} （又称为 1 位线）和 T_2 的集电极分别接数据线 D （又称为 0 位线）。

(a) 维持态。设存储单元为 0 态，并且未被选中。这时字选通脉冲没有出现，所以，字线为低电平。因此 T_1 导通， T_2 截止。而数据线 D 和 \bar{D} 上的电压均为 1.4V，电流经 T_1 流入字线，维持 0 态。即未被选中的存储单元维持状态不变。

(b) 写入态。设写入数据 1。这时，字选脉冲出现，字线为高电平（约为 3.5V），所以，两管的 be_1 结均截止；与此同时，读/写放大器将 1 位线上的电平降低到 0.7V，0 位线上的电平仍保持 1.4V，所以， T_2 开始导通，其 \bar{Q} 电平下降，因而 T_1 电流减少使 Q 电平抬高，它又导致 T_2 电流增加，经这样正反馈后迫使 T_2 饱和， T_1 截止，即数据 1 写入存储单元。当字选脉冲消失后，电流经 T_2 流入字线，维持存储单元 1 态不变。

(c) 读出态。读出时，先出现字选通脉冲，字线电平被抬高到 3.5V。若存储单元为 1 态，则经 T_2 流入字线的电流要流入 D 线，经读/写放大器转换成电平，输出 1 信号；反之， D 线无电流输出，即读出 0 信号。所以，只需鉴别 D 线上有无电流输出便可读出存储信息。

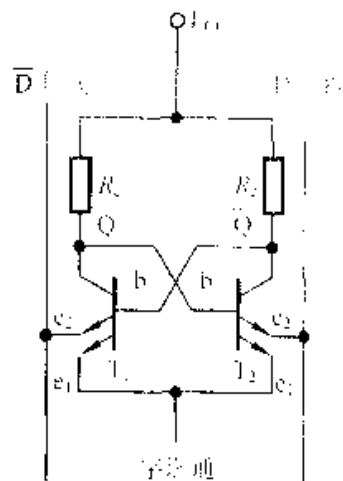


图 7.1.3 双极型存储单元

(2) 静态 MOS 存储单元

图 7.1.4 所示的是静态 MOS 六管存储单元。图中， X_i 和 Y_j 为字线；I/O 为数据写入/输出端； R/\bar{W} 为读/写控制端。当 $R/\bar{W}=0$ 时，进行写入操作；当 $R/\bar{W}=1$ 时，进行读出操作。

电路均由增强型 NMOS 管构成， T_1 、 T_3 和 T_2 、 T_4 两个反相器交叉耦合构成触发器。电路采用二元寻址，当字线 X_i 和 Y_j 均为高电平时， $T_5 \sim T_8$ 均导通，则该单元被选中，若此时 R/\bar{W} 为 1，则电路为读出态，三态门 G_1 、 G_2 被禁止，三态门 G_3 工作，存储数据经数据线 D ，通过三态门 G_3 至 I/O 引脚输出。若 R/\bar{W} 为 0，则三态门 G_1 、 G_2 工作，三态门 G_3 被禁止，由 I/O 输入数据经 G_1 、 G_2 便写入存储单元。

(3) 动态 MOS 存储单元

静态 MOS 存储单元由于管数多，又是有比型电路，故占用芯片大，集成度难以提高。动态 MOS 存储单元是无比型电路，其电路结构有四管、二管和单管等形式。由于占用芯片面积小，有利于制造大容量存储器。

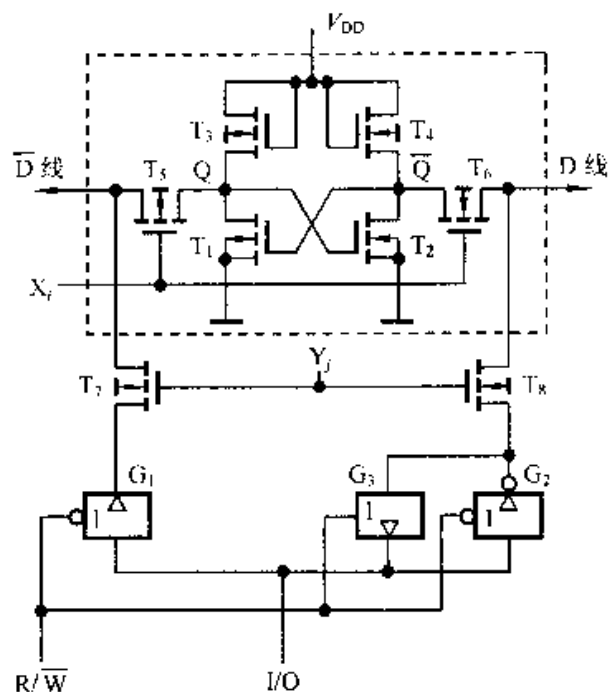


图 7.1.4 六管静态 MOS 存储单元

图 7.1.5 是三管动态存储单元, 图中还画出了该列存储单元公用的写入/刷新控制电路。存储单元由 NMOS 管 T_2 和栅极电容 C 构成, 数据存储在栅极电容 C 之中, T_1 和 T_3 是行字线 X_i 的门控管。若电容 C 上充有足够的电荷, 则 T_2 导通, 该单元呈 0 态; 反之, 该单元呈 1 态。

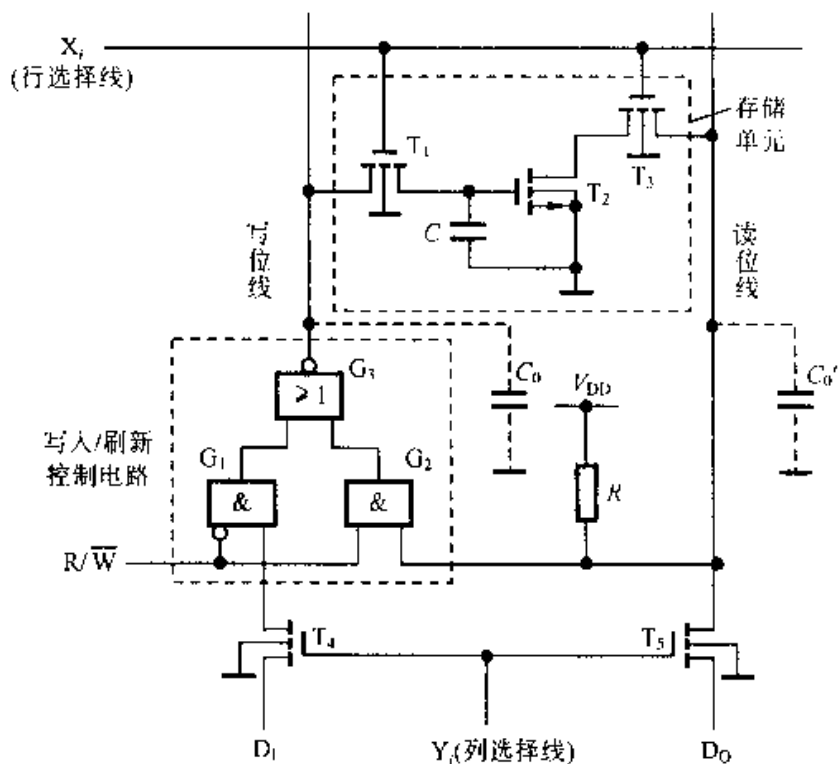


图 7.1.5 三管动态存储单元

在读操作时,行、列字线 X_i 、 Y_j 均为高电平 (令 $R/\overline{W}=1$)。所以 T_3 导通,此时若单元呈 0 态 (即 $C=1$)。则读位线上为低电平;反之,读位线上为高电平。读出数据经 T_5 输至 D_0 端输出。

在写操作时,令 $R/\overline{W}=0$, $X_i=Y_j=1$, 此时门 G_2 被封锁,输入数据 D_1 经 T_4 , 写入刷新控制电路反相后,再经 T_1 写入电容 C 中。所以,当 D_1 为 0 时,电容 C 被充电至 1 电平,该单元便呈 0 态;当 D_1 为 1 时,电容 C 放电至 0 电平,该单元呈 1 态。

动态存储单元存在两个问题:一是需要增设恢复电路。在双极型存储单元和静态 MOS 存储单元中,数据是存储于触发器中,读出之后并不破坏存储单元原来的状态,这种读出方式为非破坏性读出。而动态存储单元的数据是存储于电容 C 之中,当读出操作时, T_1 也是导通,所以,电容 C 中存储电荷要向电容 C_0 上转移。由于 C_0 是列线上所有存储单元分布电容的总和,因此 $C_0 > C$ 。每次读出数据之后,存储数据被破坏了。这种读出方式称为破坏性读出,为了保存数据,在每次读出之后要采取恢复措施,所以,要增加恢复电路。

二是需要增加刷新电路。因为电容 C 中存储的电荷会由于漏电流而逐渐减少,所以,要周期性地刷新存储数据。图中,写入/刷新控制电路就是完成恢复和刷新功能,其工作过程如下:

在读出操作时, $R/\overline{W}=1$ 。读出数据的一路经 T_5 输至 D_0 端输出;另一路经写入刷新控制电路对存储单元完成恢复功能。例如,单元存储 0,则读出时, \uparrow G_3 输出为 1 电平,经 T_1 对电容 C 充电至 1 电平,所以, T_2 导通使该单元恢复了原来的状态。

刷新是逐行进行的,一行的所有存储单元同时得到刷新。例如,对 i 行所有单元刷新,则令 $X_i=1$ ($Y_j=0$) 和 $R/\overline{W}=1$ 。此时第 i 行的所有存储单元都被选中,读位线上的数据通过写入/刷新控制电路对电容 C 进行刷新。在周期性地逐行刷新过程中,由于 $Y_j=0$,所以数据不会输出。

图 7.1.6 是单管动态 MOS 存储单元,数据储存在存储电容 C_1 中,只设置了一条数据线连接那一列的所有单元。若要访问该单元,则令 $X_i=Y_j=1$ 。即选中该单元地址,通过 T_1 把数据线电平加至 C_1 便完成写入;或者把 C_1 上的电平通过 T_1 传至数据线上便完成读出。单管存储单元存在两个问题。一是外围电路复杂,因为它的读出是破坏性的,这是由于 C_1 不可能做得很大(节省芯片面积),而 C_2 是列线上所有单元分布电容的总和,因此 $C_2 > C_1$,每次读出操作之后存储内容就被破坏了,为了保存数据则在

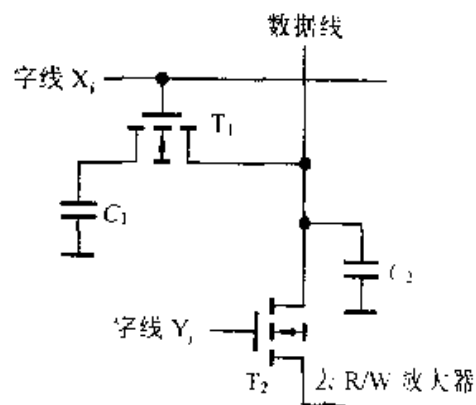


图 7.1.6 动态单管存储单元

每次读出之后要采取恢复措施，因而增设恢复电路。又由于有漏电流， C_1 上电容会逐渐减少，所以，要周期性地刷新存储数据，为此又必须增设刷新电路；二是需要高灵敏度的读出放大器，这是由于在读出时， C_2 与 C_1 并联会使数据线上的1电位降低的原故。这些技术问题早在20世纪80年代已妥善解决了，256K位D-RAM和1M位D-RAM都是采用单管存储单元。

7.1.2 RAM 的典型产品介绍

1. 超高速 ECL-RAM

双极型 ECL-RAM 是超高速随机存储器，其存储单元是采用超高速的 ECL 电路，典型存取时间为 10 ns，它可以将数据高速存入慢速取出，或者低速存入高速取出，输出为射极跟随器(或/或非门结构)，易于实现线或功能，扩展容量很简便。

CE10474 是 1K×4 位 RAM。图 7.1.7(a)为结构图，图(b)为逻辑框图。图中， \overline{CS} 为选通端， \overline{WE} 为写使能端， $D_1\sim D_4$ 为数据输入端， $O_1\sim O_4$ 为数据输出端， $A_0\sim A_9$ 为地址输入端。采用二元寻址结构，功能表如表 7.1.2 所示。

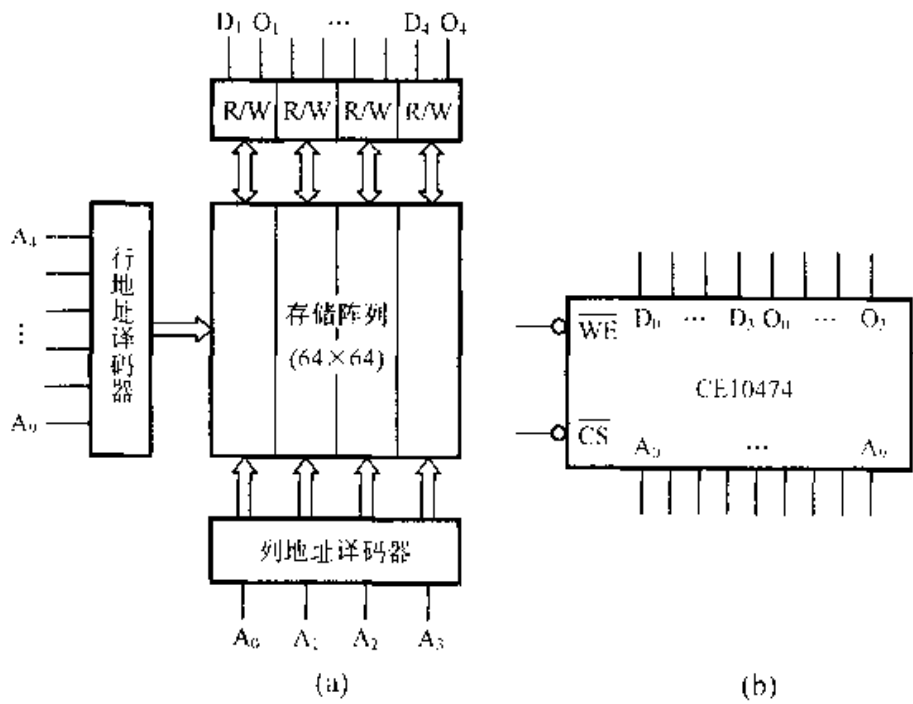


图 7.1.7 ECL-RAM CE10474
(a) 结构图 (b) 逻辑框图

2. 静态 MOS-RAM(简称 SRAM)

SRAM 产品种类繁多，在容量与功耗等指标上有很宽的覆盖面，可供不同场合应用，但其基本电路结构大同小异。图 7.1.8 是 W2114 SRAM(1K×4 位)，这是一种

典型结构, 5G2112(256×4 位)和美国的 HM6116(2K×8 位)都是采用这种结构。它采用 HC-MOS 工艺(H 表示高速), 所以, 具有高速、低功耗、单一 5V 电源、外围电路简单、输入和输出引脚公用、三态输出、使用非常简便以及存取时间短(为 100ns)的特点。

表 7.1.2 CE10474 RAM 功能表

工作方式	输 入			输 出
	\overline{CS}	\overline{WE}	D	O
保持	1	×	×	0 ¹
读出	0	1	×	O ₁ ~O ₄
写入	0	0	D ₁ ~D ₄	0 ¹

①该器件分别设置输入端 D₁~D₄和输出端 O₁~O₄。在保持方式和写入方式时, 输出端不输出数据, O₁~O₄均为 0。

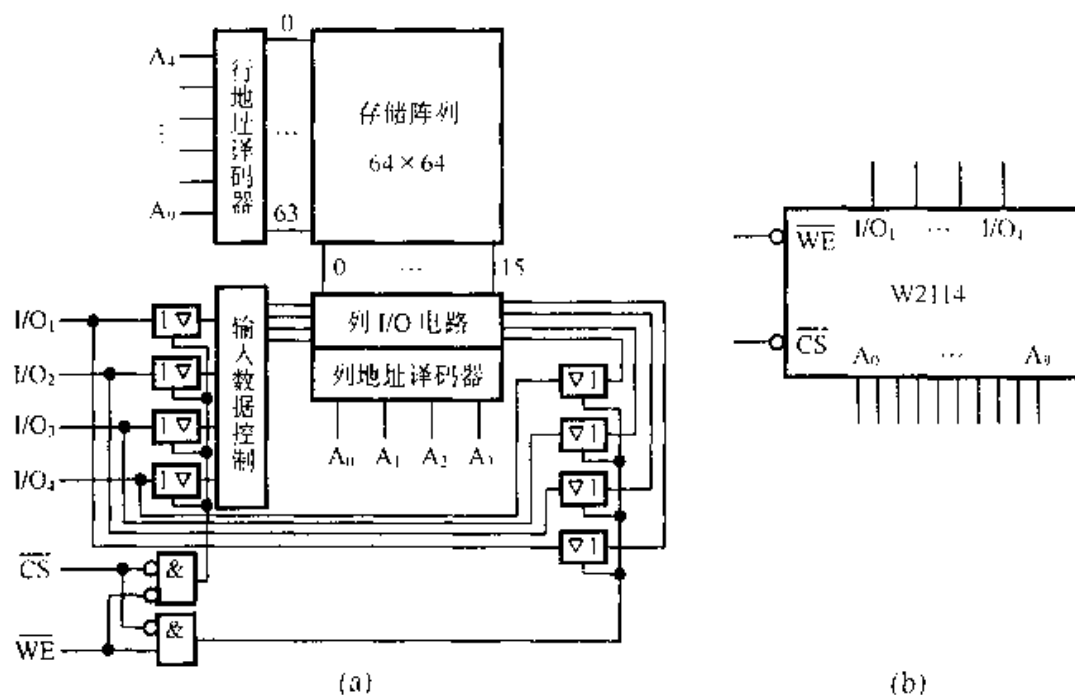


图 7.1.8 W2114 SRAM

(a) 结构图 (b) 逻辑框图

\overline{CS} 为选通端, \overline{WE} 为写使能控制端, $A_0 \sim A_9$ 为地址, $I/O_1 \sim I/O_4$ 为输入/输出。显然可见为二元寻址和三态输出结构。

\overline{CS} 为低电平有效, 电路选通之后, 若要写入操作, 则令 $\overline{WE}=0$, 输入三态门被选通(高电平有效), 数据通过输入控制电路被写入; 与此同时, 输出三态门关闭, 切断了输出与数据总线的联系。若要读出, 则令 $\overline{WE}=1$, 输入三态门关闭而输出三态门被选通, 因而存储数据被读出。功能表示于表 7.1.3。

表 7.1.3 W2114 SRAM 功能表

\overline{CS}	\overline{WE}	I/O	工作方式
1	×	高阻	未选中
0	0	0	写入 0
0	0	1	写入 1
0	1	$D_7 \sim D_0$	读出数据

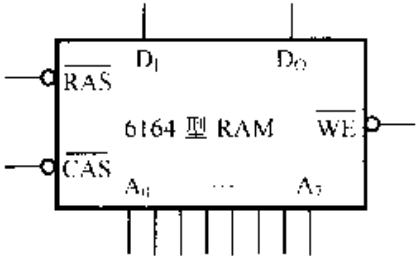


图 7.1.9 4164 型 RAM 逻辑框图

3. 动态 MOS -RAM(简称 DRAM)

大容量 DRAM 都是采用单管存储单元。为了在提高集成度的同时减少器件引脚数目，所以，目前的大容量 DRAM 多采用 1 位输入、1 位输出和地址分时输入方式。

4164 型 RAM 的逻辑框图如图 7.1.9 所示。图中， \overline{WE} 为写使能信号输入端； D_1 为数据输入端， D_0 为数据输出端； \overline{RAS} 和 \overline{CAS} 分别为行、列地址选通端； $A_0 \sim A_7$ 为地址码输入端。

4164DRAM 容量为 $64K \times 1$ 位，采用单管存储单元。功耗低、单一电源+5V、并且输入/输出与 TTL 兼容。

4164DRAM 采用地址分时输入方式。即把 16 位地址码分为行地址（8 位）和列地址（8 位），将行地址和列地址分两次从同一组引脚 $A_0 \sim A_7$ 输入、分时操作由 \overline{RAS} 和 \overline{CAS} 两个信号控制。首先令 $\overline{RAS}=0$ ，将行地址码 $A_0 \sim A_7$ 输入在地址锁存器内；然后令 $\overline{CAS}=0$ ，将列地址码 $A_8 \sim A_{15}$ 输入地址锁存器内。

7.1.3 RAM 的容量扩展

单片 RAM 容量总是有限的，所以，在 RAM 设计中已考虑到增设多个选通端和方便容量扩展的输出结构，如三态输出或 OC 输出结构，ECL 型的射随输出结构等。

1. 字长扩展

字数保持不变，而增加每个字的位数称为字长扩展，又称为位扩展。例如，利用两片容量为 256×4 位 RAM 可以扩展为 256×8 位 RAM。为此，只需将它们的选通端、输出使能端和地址输入都并接起来，输入/输出保持独立便可。其中，一片为高 4 位，另一片为低 4 位，从而每字扩展为 8 位(请读者自画连接图)。

2. 地址扩展

增加字数而每字位数保持不变称为地址扩展，又称为字扩展。例如，用 4 片 CM5101RAM (256×4 位)可以构成 $1K \times 4$ 位 RAM。图 7.1.10 是 CM5101 的逻辑框图，连接图示于图 7.1.11。请注意该器件设置有两个选通端，其中， \overline{CS}_1 为低电平

选通, $\overline{CS_2}$ 为高电平选通; \overline{DIR} 为不允许输出端, 高电平有效; \overline{WE} 为写使能端, 低电平有效(器件类型不同, 符号定义及其有效电平可能不同, 实用中必须注意这一差异)。

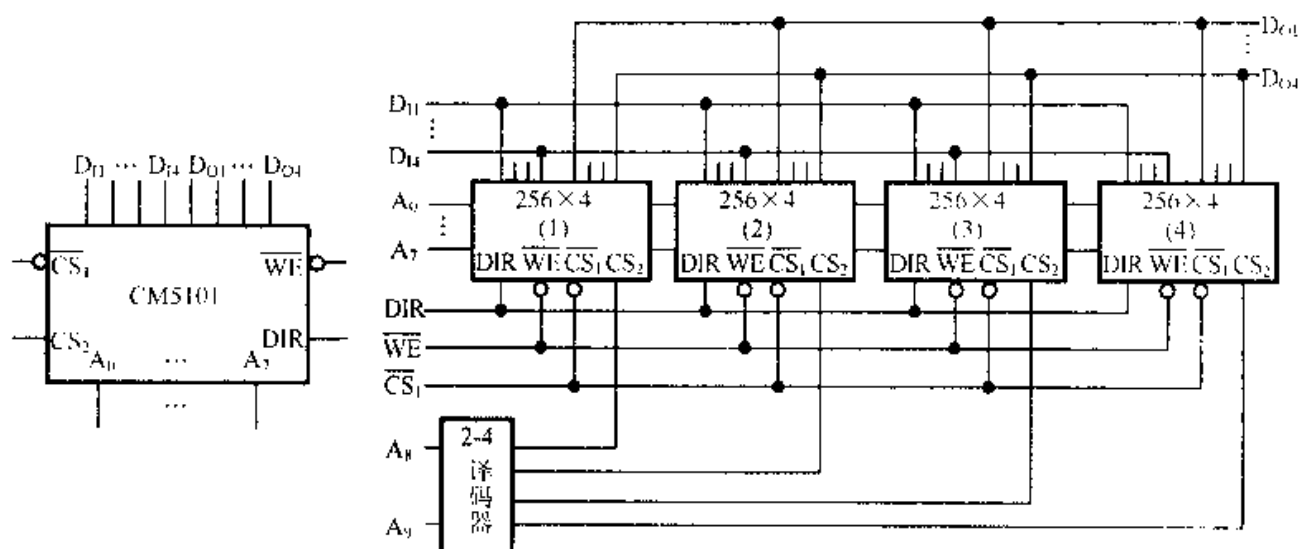


图 7.1.10 CM5101 型 RAM

图 7.1.11 地址扩展连接图

地址 $A_0 \sim A_7$ 并行加在 4 片 RAM 上, 每片 RAM 的对应输出连接在一起。地址 $A_8 \sim A_9$ 加在 2-4 线译码器上, 由译码器输出分别控制 4 片 RAM 的选通端 $\overline{CS_2}$; 它们的 \overline{WE} 端, \overline{DIR} 端都分别并接起来, $\overline{CS_1}$ 并接起来作为 $1K \times 4$ 位 RAM 的选通端。这样便使译码器的每一种输出只有其中一片 RAM 工作, 其他 3 片输出为高阻状态。换言之, 在地址 $A_0 \sim A_9$ 的每一种组合状态下都得到 4 位数据输出。

实际应用中, 经常是需要同时进行地址与字长扩展, 例如, 存储体总容量为 $16K \times 16$ 位, 若用 $2K \times 8$ 位 RAM 构成, 则必须用 16 片 $2K \times 8$ 位 RAM 同时进行地址和字长扩展, 原理如上述, 将两种扩展方法结合起来便可实现了, 请读者自己画出连接图。

7.2 只读存储器(ROM)

只读存储器是存储固定内容的存储器, 工作时一般不改变存储内容, 只是进行反复读出操作。它广泛用来存储固定函数、固定程序或指令、通信系统的模拟信号源、字符及汉字库等。

ROM 种类繁多, 如以使用元件来分类, 则有二极管 ROM、TTL-ROM、MOS-ROM; 如以内容的更新情况来分, 大致可分为

(a) 固定内容只读存储器 ROM: 它的内容由生产厂在制造过程中存入, 不可更新。其特点是集成度高、可靠性高、成本低。

(b) 可编程序只读存储器 PROM: 它的存储单元在制造时全部做成 1 态(或 0 态), 可写入用户编制的程序, 一旦写入后便不可更改, 即只能实现“一次写入”。其特点是通用性强, 灵活性大; 但是“一次写入”操作麻烦, 目前应用较少。

(c) 可擦可编只读存储器 EPROM: 可以多次更改内容, 一般是采用紫外线照射抹去原先存储内容, 再重新写入新内容。它比 PROM 通用性更强, 使用灵活性更大, 所以应用范围广泛。

(d) 电可擦可编只读存储器 E²PROM: 可多次更改内容, 不过它勿需用紫外线照射, 它的改写是由高电压或者由控制端的逻辑电平来完成, 所以使用最方便。深受用户欢迎, 在产品开发过程中使用这类存储器者日渐增多。

(e) 快闪只读存储器: 可以多次更改内容, 同 E²PROM 类似, 可以在工作电压下擦除。它的集成度高、容量大、擦除及改写速度快。

7.2.1 只读存储器(ROM)

1. TTL-ROM

单个信息只有 0 和 1 两种状态, 故只需用一只晶体管来作一个存储单元。当晶体管接入电路时, 在字线选通后晶体管导通, 与之连接的位线有电流输出, 这表示该单元为 1 态; 反之, 若晶体管发射极开路, 字线选通后位线无电流输出, 这表示单元为 0 态, 如图 7.2.1(a)所示。

ROM 结构与 RAM 类似, 但是这种固定内容只读存储器无写入电路, 图 7.2.1(b)是 4×4 位 ROM 的逻辑图。存储矩阵中的每一行有 4 只晶体管, 它们的基极与一条字线相连, 所以, 每一行代表一个字, 每个字有 4 位数据。假如某单元是 1, 则相应晶体管发射极与位线连接, 反之则断开。读出时首先要寻址, 被选中字线呈高电平, 发射极与位线连接的晶体管导通, 因此位线有电流输出, 这些单元为 1; 另一些发射极与位线断开的晶体管是截止的, 相应位线无电流输出。这些单元为 0, 被选中字的 4 位信息便同时读出。没有选中的字, 其晶体管基极都是低电平, 无论发射极与位线是否连接都不会在相应位线上产生输出电流。

在产品中, 存储矩阵的每一行晶体管都是用一个多发射极晶体管来代替的, 如图 7.2.1(c)所示。矩阵中晶体管发射极与位线是否连接应该由用户程序确定, 但是只能由生产厂用掩模工艺完成。器件制造出来之后, 存入的内容不能更改, 所以为固定内容 ROM; 假如在制造时将所有晶体管发射极与位线之间串接一个低熔点的熔丝, 如图 7.2.1(d)所示, 则出厂产品的所有存储单元为 1 状态, 用户可依据自己的程序要求, 对存储数据 0 的单元进行“烧断”处理, 使它们由 1 改变为 0, 从而

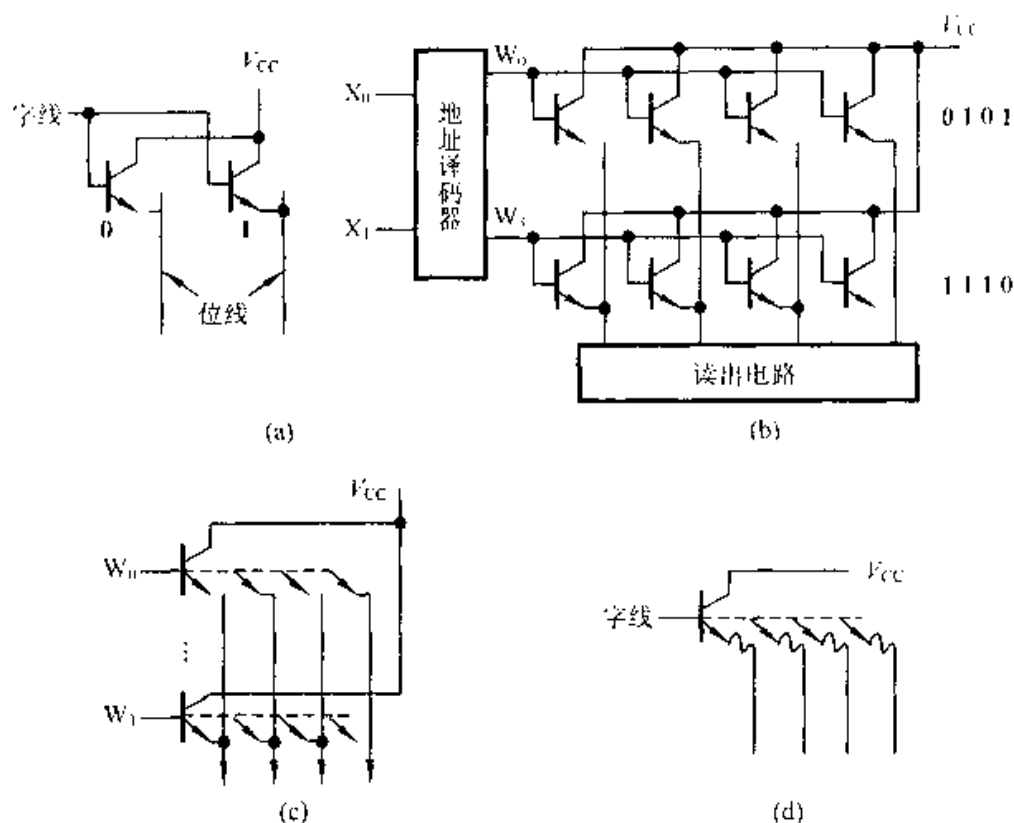


图 7.2.1 ROM 结构图

(a) 存储单元 (b) 4×4 ROM 结构图 (c) 多发射极结构 (d) 熔丝结构

完成一次写入。用户一旦写入内容便无法再改动内容，所以为 PROM。双极型固定 ROM 只适用于大批量定型产品，由工厂定做，而 PROM 已成为过渡性产品，实用中很少采用，所以对它们不再深入讨论。

2. MOS-ROM

MOS-ROM 中，存储单元用有无 MOS 管来表示，若有 MOS 管则该单元为 1，反之为 0。图 7.2.2 是 1024×4 位 NMOS-ROM 电路结构图，输入有 10 位地址码，经地址译码器输出 1024 条字线，每字有 4 位码输出(即 4 条位线)，所以，在存储矩阵中包含有 1024×4 个交叉点，在交叉处制作一个 NMOS 管则表示该单元为 1，反之表示该单元为 0。图中还表示出 NMOS 管在字线与位线之间如何连接。

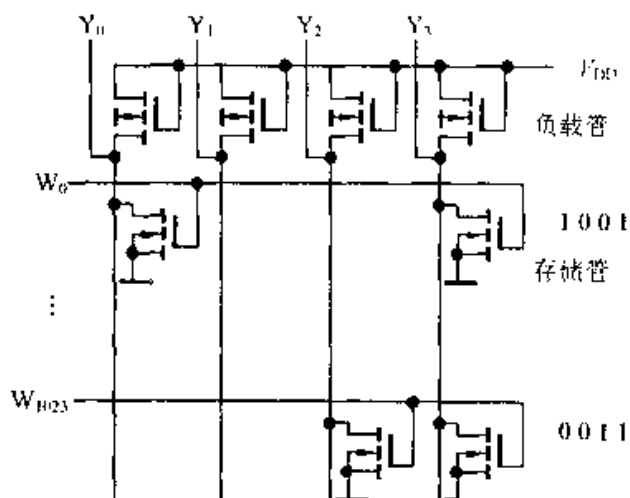


图 7.2.2 NMOS-ROM 结构图

例如, 字线 W_0 为高电平, 位线 Y_0 和 Y_3 为低电平, 而位线 Y_1 和 Y_2 为高电平, 经读出放大器就可鉴别出字线 W_0 的存储数据 $Y_0Y_1Y_2Y_3$ 为 **1001**; 同理, 字线 W_{1023} 存储数据为 **0011**。

这种 MOS-ROM 为固定内容 ROM, 只能由工厂采用掩模工艺制造, 其中, MOS 管也可以采用 CMOS 管, 以降低功耗。它主要用于大批量定型产品, 例如, 按国标 GB-2312-80 规定的—/二级汉字库——GB5199(汉字点阵 15×16), 容量为 4M 位, 读取时间为 250ns, 操作功耗为 220mW, 备用功耗为 550 μ W。

7.2.2 可擦可编程只读存储器(EPROM)

EPROM 是可以多次擦除和改写内容的 ROM, 它的存储单元采用叠层栅 (简称叠栅) MOS 管 (SIMOS 管), 图 7.2.3(a) 是 SIMOS 管的结构示意图。它是一个 N 沟道型的 MOS 管, 设有两个栅极 G 和 G_f 。 G 称为控制栅, 其功能与 NMOS 的栅极相同; 而 G_f 称为浮栅, 因为它完全埋在绝缘层 SiO_2 之中, 处于“浮置”状态, 它的功能是长期保存注入电荷。

浮栅上未注入电荷之前, 在控制栅 G 上加入正常工作电压 +5V 时, 漏-源之间可以形成导电沟道, SIMOS 管为导通状态, 相当于存储 **1**; 反之, 若在浮栅上注入了负电荷之后, 必须在栅极上加入更高的电压才能抵消负电荷的影响而形成导电沟道, 因此在正常工作电压时, SIMOS 管不会导通, 相当于存储 **0**。所以, 由这种存储单元构成的 EPROM 产品全部存储 **1**。

用户编程时, 对于需要写 **0** 的单元可在漏-源间加上高电压 (+25V), 由于沟道电场足够强而产生雪崩现象, 由此产生一些高能电子。此时控制栅加上高压脉冲 (+25 V), 在它的吸引之下, 便有一些高能电子穿过绝缘层而注入浮栅, 当高电压消失后, 浮栅上注入电子被绝缘栅包围, 泄漏电流极小, 所以, 一旦浮栅上注入电子就能保存相当长时间 (一般为 10 年)。

若要改写内容, 则先需用紫外线照射, 使浮栅上的注入电子获得足够的光子能量, 从而穿过绝缘层回到衬底中, 这样可使注入到浮栅上的电子消失, 全部存储单元又恢复为全 **1** 状态, 然后用前面介绍的用户编程方法重新写入新内容。有一种擦刷机专门用来擦除 EPROM 芯片, 通电 20 分钟便完成了擦除操作, 图 7.2.3(b) 是 SIMOS 管符号。

图 7.2.3(c) 所示的是 EPROM 叠栅存储单元, 在字线 W_i 为高电平时, 若浮栅上没有注入电子, 则位线 D_j 为低电平; 若浮栅上注入电子, 则位线 D_j 为高电平, 经过读出放大器便可鉴别出存储内容。

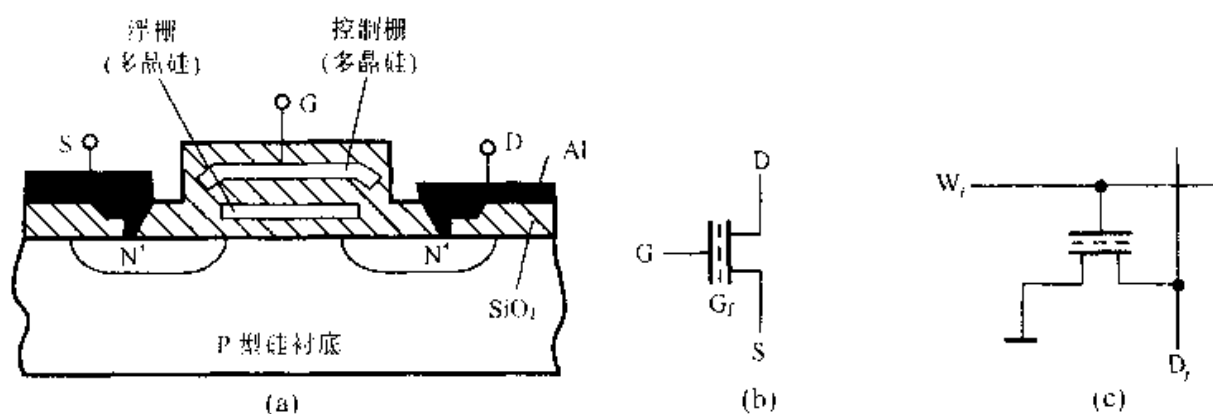


图 7.2.3 EPROM 存储单元

(a) SIMOS 管结构图 (b) SIMOS 管符号 (c) EPROM 存储单元

目前常用的 EPROM 是 2716 ($2K \times 8$ 位)、2732 ($4K \times 8$ 位)、2764 ($8K \times 8$ 位) 和 27128 ($16K \times 8$ 位)。它们的存储单元都是采用 SIMOS 管, 电路结构相同仅存储容量有差别, 所以, 使用方法和电参数也类似。输入/输出与 TTL 兼容, 读取时间为 350ns, 操作功耗为 525mW, 备用功耗为 132mW。

图 7.2.4(a) 是 2716 的结构图, 图(b)为逻辑框图。

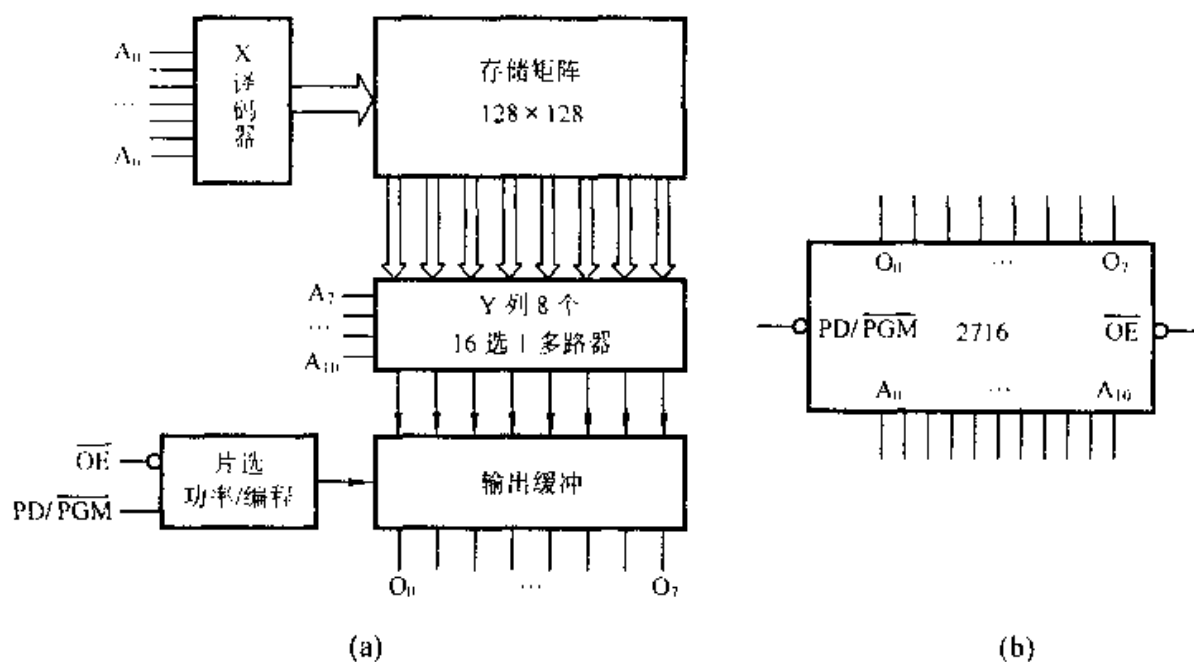


图 7.2.4 2716

(a) 结构图 (b) 逻辑框图

该电路的存储矩阵为 128×128 方阵, 地址码有 11 位, 其中, 7 位用于行地址译码器, 4 位用于列地址译码器。 \overline{OE} 为输出允许端, 低电平有效; $\overline{PD/PGM}$ 为功率/编程控制端, 它的功能是在芯片不工作时使电路工作在功耗下降方式, 即令 $\overline{PD/PGM}$ 为高电平, 电路输出为高阻状态以降低功耗, $\overline{PD/PGM}$ 为低电平, 电路为

工作状态。

读出时，寻址后存储矩阵即可输出数据，但是，能否输出到外部数据总线上去还需由 \overline{OE} 和 $\overline{PD/PGM}$ 信号来确定。

假如要改写内容，则先用擦抹机将原有内容擦去后再写新内容。此时，应令 $V_{pp}=22\sim 25V$ ， $\overline{OE}=1$ 。在提供地址和输入数据后，在 $\overline{PD/PGM}$ 端加入一个正脉冲便完成写入，写入结束后， $\overline{PD/PGM}$ 回到低电平。为了检查写入内容是否正确，可以在保持 $V_{pp}=22\sim 25V$ 的条件下进行一次读出操作，即令 \overline{OE} 和 $\overline{PD/PGM}$ 同时为低电平，检查读出数据是否与写入内容一致。

电路的各种工作模式如表 7.2.1 所示。

表 7.2.1 2716 工作模式表

模 式	$\overline{PD/PGM}$	\overline{OE}	V_{pp}/V	V_{cc}/V	输 出
读	0	0	+5	+5	数据读出
未选中	×	1	+5	+5	高 阻
功率下降	1	×	+5	+5	高 阻
编 程	正脉冲	1	+25	+5	数据输入
程序校验	0	0	+25	+5	数据读出
程序禁止	0	1	+25	+5	高 阻

我们在使用 EPROM 芯片时发现有两个问题应给予注意。

(a) 内容保存时间：浮栅上空穴保存时间与温度和光照有关，在常温(20℃)和避免光照条件下可保存 10 年；在 70℃ 高温环境中只能保存一年；在阳光照射下则只能保存几周；长期暴露在日光灯照射的环境下则只能保存三年。所以，必须注意工作环境，在内容写入之后，应该用不透光的纸将芯片上的擦除窗口覆盖。

(b) V_{pp} 电压调整：不同公司产品的 V_{pp} 电压值有较大差异，一般为 20~25V，在产品说明书或芯片表面上都有标注。但是，在利用编程器按照标注电压进行写入操作时往往碰到写不进去的情况。这时应该耐心地在标注电压附近调整电源电压 V_{pp} ，有可能在偏离芯片标注电压之下完成写入操作。

7.2.3 电可擦可编程只读存储器(E²PROM)

E²PROM 是电可擦可编程 ROM，它的擦除不需借助紫外线照射，只需在高电压或者工作电压下就可进行擦除，这比 EPROM 简便，同时它还具有字擦除及字改写功能。它没有石英玻璃窗口，有利于数据保存和降低成本。由于这些优点，它已在

许多实用场合取代了 EPROM。

E^2 PROM 的存储单元采用浮栅隧道氧化层 MOS 管, 简称 Flotox 管, 结构如图 7.2.5(a)所示。它是另一种类型的叠栅 MOS 管, 也有两个栅极, 上面有引出线的栅极为控制栅 G , 埋于绝缘层中的栅极称浮栅 G_f , 浮栅与漏极区之间有一极小面积的 SiO_2 绝缘层, 非常薄 ($10\sim 15nm$), 称为隧道区。图 7.2.5(b)所示的是 Flotox 管的符号, 图 7.2.5(c)所示的是 E^2 PROM 存储单元。

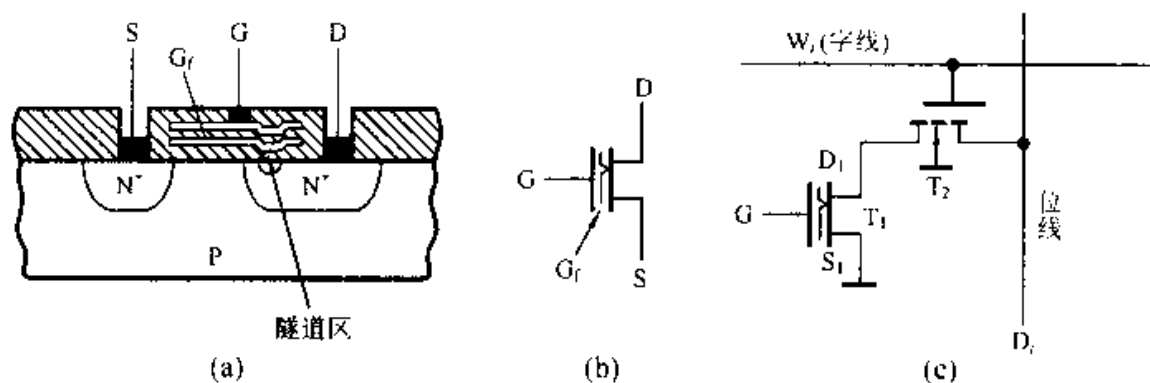


图 7.2.5 E^2 PROM 存储单元

(a) Flotox 管结构示意图 (b) 符号 (c) 存储单元

在图(c)电路中, 浮栅注入电子是利用隧道效应进行的, 令 $W_j=1$, $D_i=0$, 则 T_2 导通, T_1 漏极 D_i 近于 0 电平。这时在控制栅 G_1 上加 21V 正脉冲, 则在浮栅与漏极区之间极薄绝缘层内出现隧道, 通过隧道效应使电子注入浮栅, 正脉冲消失后, 浮栅将长期保存这些电子, 定义为 1 状态。若要把控制栅接 0 电平, 令 $W_j=1$, 在 D_i 加上 21V 正脉冲, 则浮栅上的电子通过隧道返回衬底, 浮栅上就没有注入电子, 定义为 0 状态。目前, E^2 PROM 允许改写次数达到 1 万次, 数据保存时间达到 10 年。

在读出操作时, 浮栅上注入电子的存储单元, T_1 不能导通, 在位线 D_i 上读出 1; 如果在浮栅上没有注入电子, 则 T_1 导通, 在位线 D_i 上读出 0。

E^2 PROM 需用高压脉冲擦写, 一般要用专用编程器来完成。但是, 有些芯片内部设置有升压电路, 读、写及擦除均在工作电压下进行, 例如 X2816、X2864 芯片。

X2816($2K \times 8$ 位) E^2 PROM 逻辑框图为图 7.2.6, 同类产品还有 X2864($8K \times 8$ 位)。写入时间只需 10ns, 读取时间 300ns, 操作电流 110mA, 备用电流 40mA, 输入/输出与 TTL 兼容。

内容改写是由读/写控制端 \overline{WE} 逻辑电平控制, 当 \overline{WE} 为 1 时进行读出操作, 当 \overline{WE} 为 0 时进行写入操作, 可见改写内容非常方便, 而且具有在线编程的独特功能, 存储内容的保存时间达

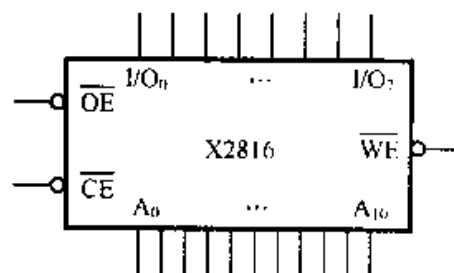


图 7.2.6 X2816 型 E^2 PROM

10 年以上, 而改写次数已提高到 100 次以上, 有些产品已达到 10000 次。由于这些优点使它的应用范围日渐扩大, 成为所有 ROM 器件中的佼佼者。

*7.2.4 快闪只读存储器 (U 盘)

E^2 PROM 存储单元用了两只 MOS 管 (见图 7.2.5(c)), 这就限制了集成度的进一步提高; EPROM 存储单元只用一只 SIMOS 管, 但是擦除及改写内容很不方便。快闪存储器兼有 EPROM 结构简单和 E^2 PROM 擦除及改写方便的优点, 是一种集成度高、容量大、成本低和使用方便的新型器件。

快闪存储器采用叠栅 MOS 管作为存储单元。图 7.2.7(a) 是叠栅 MOS 管的结构示意图, 图(b)所示的是它的符号, 图(c)所示的是存储单元。

叠栅 MOS 管的结构与 EPROM 中的 SIMOS 管相似, 两者的区别在于浮栅与衬底间氧化层的厚度不同, 前者仅为 SIMOS 管的 $1/3$ 左右; 此外, 浮栅与源极重叠的面积很小, 有利于产生隧道效应。

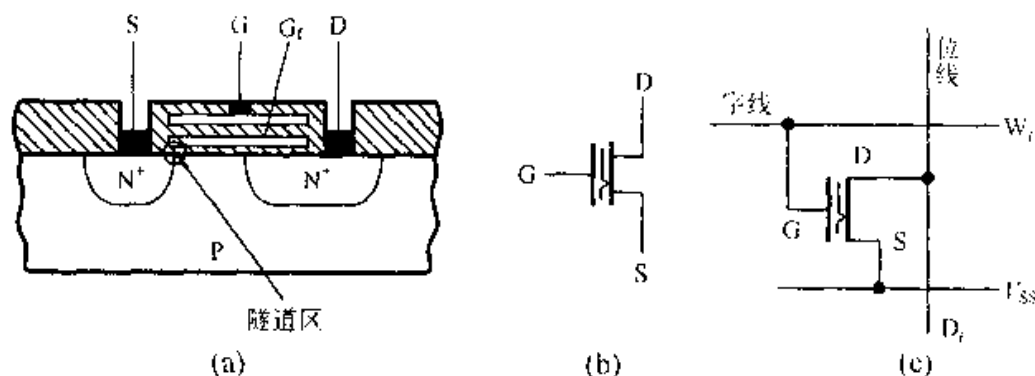


图 7.2.7 快闪存储单元

(a) 叠栅管结构示意图 (b) 符号 (c) 存储单元

写入方法与 EPROM 相同, 即利用雪崩注入的方法使浮栅充电, 相当于存储 0; 浮栅未注入电子, 相当于存储 1。

读出时, 令 $W_t=1$, $V_{ss}=0$ 。若浮栅上没有注入电子, 则叠栅 MOS 管导通, 位线 D_i 输出低电平; 反之, 叠栅 MOS 管截止, 位线 D_i 输出高电平, 故只需鉴别位线 D_i 的电平便可读出存储内容。

擦除方法与 E^2 PROM 类似, 它是利用隧道效应来完成的。此时, 控制栅 G 为 0 电平, 源极加入高压脉冲 (12V), 在浮栅与源区间很小的重叠区域产生隧道效应, 使浮栅上的电荷经隧道返回衬底。由于有的芯片是将所有源极连在一起, 或者有的芯片是分段将源极连在一起, 所以, 只能实现按段擦除或整片擦除, 而不能像 E^2 PROM 那样实现字擦除。

快闪只读存储器目前有两类产品: 一类是 12V 电压擦除编程器件, 如美国 AMD

公司的 28F 系列；另一类是 5V 电压擦除编程器件，如该公司的 29F 系列，产品的存储容量分为 1M 位、4M 位、8M 位和 16M 位。目前，已有 256M 位和 1000M 位的芯片面世。

*7.2.5 ROM 的应用

1. 查表 ROM

对数、指数、乘方、三角函数等常规计算是计算机所必备的功能，可以设计一个 ROM 表，它的输入与输出有固定的函数关系。将自变量以地址码的形式输至 ROM，在其输出端便得到函数值。显然，凡是能写出真值表的任何计算都可以用 ROM 表来实现。

2. 码制转换

ROM 可看成一个代码转换系统，见图 7.2.8(a)，因而可以方便地实现码制转换。设该系统有 M 位输入代码转换为 N 位输出代码，其中， N 可小于、等于或大于 M 。对于一个确定的 M 位代码可以通过 ROM 得到对应的 N 位输出代码，如图 7.2.8(b) 所示。因为有 M 位输入代码，所以 ROM 字线为 $2^M = \mu$ ，即地址线有 μ 条 ($W_0 \sim W_{\mu-1}$)。把存储矩阵作为编码器，将每一个字线编成所希望的输出代码就实现了码制转换。

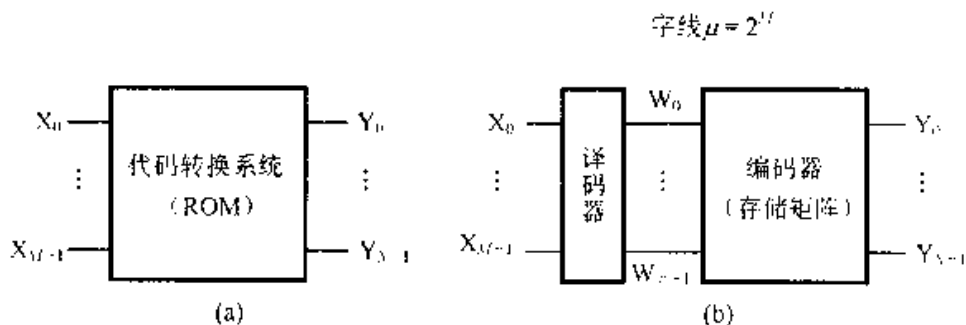


图 7.2.8 码制转换

(a) 示意图 (b) 结构图

例如，用 ROM 实现二进制码至格雷码的转换，其真值表如表 3.3.2 所示(参见 3.3 节)，ROM 的输入是 4 位二进制码 $X_0 \sim X_3$ ，在译码器中译为字线 $W_0 \sim W_{15}$ ，每条字线选通时便可输出相应的格雷码。由表 3.3.2 得到

$$Y_0 = W_1 + W_2 + W_3 + W_6 + W_9 + W_{10} + W_{13} + W_{14}$$

$$\vdots$$

$$Y_3 = W_8 + W_9 + W_{10} + W_{11} + W_{12} + W_{13} + W_{14} + W_{15}$$

上式很容易用二极管、多发射极晶体管或者 MOS 管构成的存储矩阵来实现，请读者画出由其中一种元件构成存储矩阵的连接图。

3. 脉冲序列发生器

在数字系统中，为了进行调试或控制，经常需要几个脉冲序列，我们可以用具有 n 位输出的 ROM 并借助 4 位二进制计数器来改变地址，随着计数器状态从 0000 至 1111 循环变化，则字线选通的顺序按 $W_0 \rightarrow W_1 \rightarrow W_{15} \rightarrow W_0$ 次序循环变化。其连接如图 7.2.9 所示。从 ROM 输出端便可得到 4 个同步的脉冲序列。如果依据脉冲序列正确地给 ROM 编程，则可得所需要的 4 个脉冲序列 $F_1 \sim F_4$ 。

4. 伪彩色处理电路

在图像处理技术中，若想将一幅黑白图像变成彩色图像显示，则可以用 ROM 构成加彩处理电路。例如，在研制数字超声电视测井系统中^[26]，为了显示彩色的井壁图像，对检测的井壁回波信号进行了加彩处理，实际上就是用 3 片 ROM 进行代码转换，如图 7.2.10 所示。其中，3 片 ROM 将代表黑白图像的一组代码按照不同的转换模式(即编程不同)进行转换，获得代表红、绿、蓝等三种色彩的信号加入监视器，即可实现伪彩色的井壁图像显示。顺便指出，转换模式需要经过多次实验不断修正才能获得令人满意的效果。

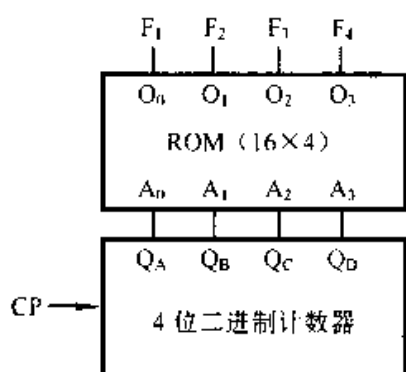


图 7.2.9 脉冲序列发生器

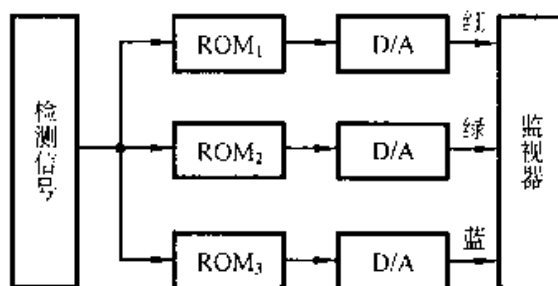


图 7.2.10 伪彩色电路逻辑框图

5. 字符发生器

用 ROM 可以非常简便地产生中、外文字符或简单的图案，因而在宣传、信息报导以及广告行业获得广泛应用。如果与单片机结合起来使用，则能实现快速更改信息内容或者循环显示多幅广告。例如，用 RAM 显示“Z”，采用 7×5 位矩阵构成，如图 7.2.11 所示。图中，每一列有一个公共输出端，经输出缓冲器至光栅矩阵，选中某行时，该行内容就以光点反映在光栅矩阵上，存 1 单元应于光栅上出现亮点。若地址码周而复始地循环改变，各行内容就相继出现在输出端，光栅上就显示出字样“Z”。显然，显示字符越多则要求 ROM 容量越大，若要显示 n 个字符，则 ROM 总容量为 $n \times 7 \times 5$ 位。

产生多字符的 ROM，地址码分为两组，一组是完成逐行扫描的行译码；另一组是选择字符的字特征码，每个字符都对应着一个字符特征码。图 7.2.12 是 64 字符发生器的逻辑框图， $A_0 \sim A_2$ 是行扫描译码输入， $A_0 \sim A_2$ 的循环变化便完成被选字

符的逐行扫描, $B_0 \sim B_5$ 是字特征码输入, 决定显示什么字符。

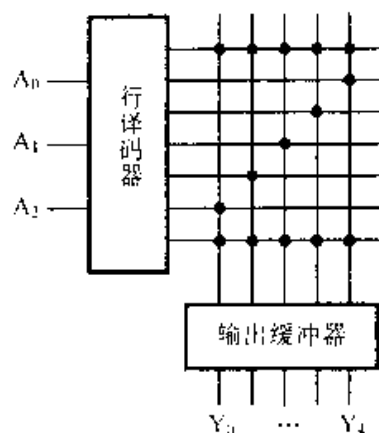


图 7.2.11 字符发生器结构图

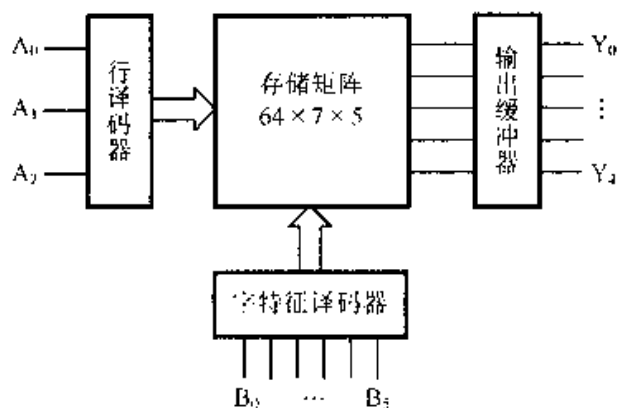


图 7.2.12 多字符发生器结构图

7.3 双端口随机存储器 (DPRAM)

7.3.1 电路结构

双端口随机存储器 DPRAM 是在基本静态 RAM 基础上扩展而成, 电路逻辑结构如图 7.3.1 所示。

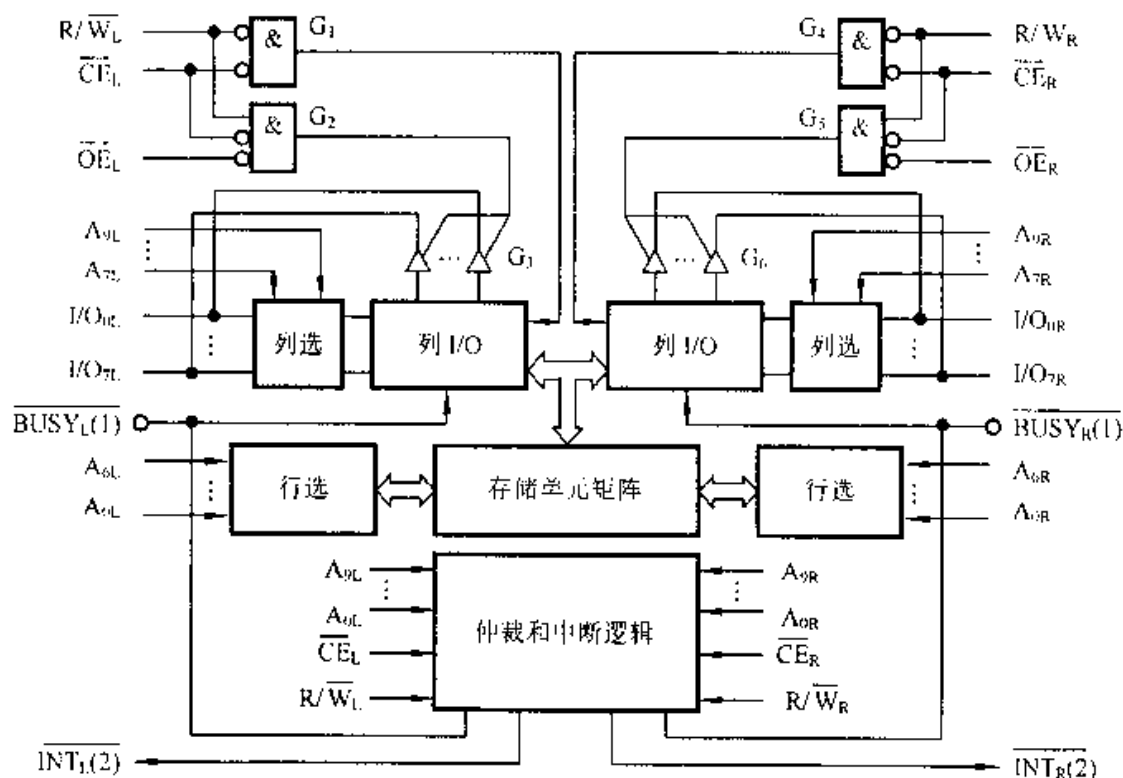


图 7.3.1 双口 RAM 电路逻辑结构图

电路设置有两对地址端口和数据端口, 对称地分布在左、右两边。此外, 有一个公共的存储单元矩阵和一个仲裁和中断逻辑。

数据的读/写是在公共的存储单元矩阵上完成的。两组地址及控制逻辑允许两个口在存储单元矩阵上独立地进行读/写操作; 而两组 I/O 口分别作为两口的数据写入或读出。如果两口的读/写操作是指向同一个存储单元, 则由“仲裁逻辑”来判定哪个口有效; “中断逻辑”提供端口之间或系统之间的数据通信。

7.3.2 工作原理

1. 读/写操作

两个口独立的读/写操作同静态 RAM 的相同。例如, 若为左口读操作, 令写使能信号 $R/\overline{W}_L=1$, 片选通信号 $\overline{CE}=0$, 输出允许信号 $\overline{OE}_L=0$, 则 G_2 输出 1 电平选通 8 个输出三态门 G_3 ; G_1 输出 0 电平使 8 个输入三态门关闭 (图中未画出, 参见图 7.1.8(a))。所以, 被地址 $A_{0L} \sim A_{9L}$ 选中的存储单元的 8 位数据便由 $I/O_{0L} \sim I/O_{7L}$ 读出。如果是左口写操作, 令 $R/\overline{W}_L=0$, $\overline{OE}_L=0$, $\overline{OE}_L=1$, 则作类似分析可知: 8 位数据从 $I/O_{0L} \sim I/O_{7L}$ 写入地址 $A_{0L} \sim A_{9L}$ 选中的存储单元。

2. 仲裁逻辑 (\overline{BUSY})

如果发生两个口同时对某个存储单元读/写, 则由仲裁逻辑来判定哪个口有效。例如, 当右口的地址信号略早于左口的地址信号时, 则右口操作有效, 同时仲裁逻辑将左口“忙” (\overline{BUSY}_L) 标志置为低电平, 这个 \overline{BUSY}_L 低电平信号可以提供保持本次操作和地址信息之用。当右口读/写完毕, 仲裁逻辑将 \overline{BUSY}_L 置为高电平后, 左口便可读/写了。

3. 中断逻辑 (\overline{INT})

中断逻辑是一个可供选用的功能, 输出信号为 \overline{INT}_L 和 \overline{INT}_R , 它的功能是提供中断信号以便执行中断服务程序。在使用中断功能时, 存储矩阵中的 03FEH 和 03FFH 单元被作为“标志”(或称“邮箱”。在右口写完 03FEH 后, 中断逻辑将 \overline{INT}_L 置于低电平, 并由它启动中断服务程序。当主机从左口读完 03FEH 单元之后, 中断逻辑将 \overline{INT}_L 置为高电平, 表示中断服务程序执行完毕, 右口又可进行读/写操作了; 同样, 在左口写完了 03FFH 单元之后, \overline{INT}_R 将被置为低电平, 并由它启动中断服务程序, 当主机从右口读完 03FFH 单元之后, \overline{INT}_R 被置为高电平, 表示中断服务程序执行完毕, 左口便可进行读/写操作了。“中断逻辑”使两个系统之间的数据通信更简便。顺便指出, 若不使用中断功能时, 03FEH 和 03FFH 单元仍作普通的存储单元。

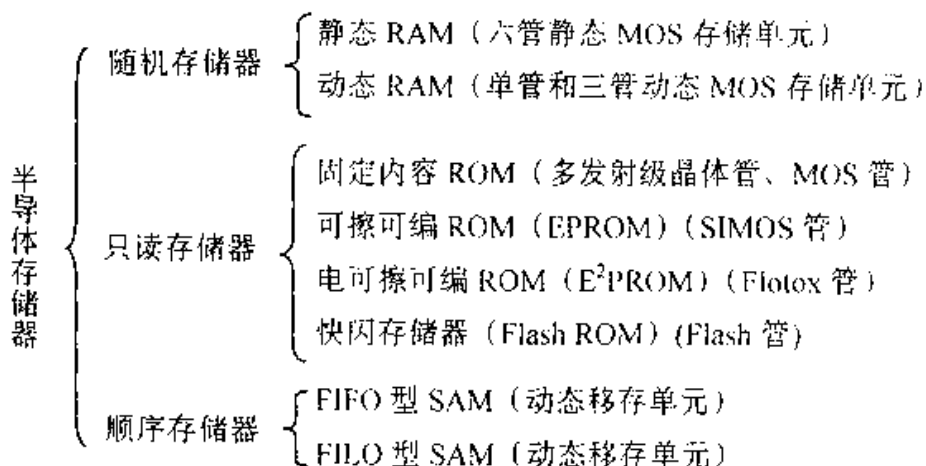
小 结

存储器是应用最广泛的逻辑器件之一。自 1970 年第一片存储器问世以来(1K 位),其集成度和工作速度的提高非常迅速,目前已有 1M 位和 16M 位随机存储器投放市场,它们为计算机快速升级、更新换代创造了条件。

在半导体存储器中采用按地址存放数据的方法。只有被地址指定的存储单元才能与输入/输出端接通,可以对这些指定单元进行读/写操作。所以,存储器的电路结构必须包含有地址译码器、存储矩阵和输入/输出电路。

虽然存储器的存储容量逐年增加,但是在计算机或大型数字系统中,往往会出现单片存储器的容量不能满足要求的情况。这时可以采用字扩展及位扩展的方法,采用多片存储器构成存储容量更大的存储体。因此,存储器容量的扩展方法是一个很重要的和实用的方法。

存储器种类繁多。下面将本章所介绍的半导体存储器类型汇总如下。



思考题和习题

7.1 解释下列名词:

字线、位线、存储矩阵、存储容量、静态 MOS-RAM、动态 MOS-RAM。

7.2 ROM 与 RAM 有何异同?它们各用于什么场合。

7.3 动态 MOS 电路有何优缺点,它为什么在 LSI 和 VLSI 器件中获得广泛应用。

7.4 举例说明 RAM 的容量扩展方法。

7.5 固定内容的 ROM、PROM、EPROM、E²PROM 之间有所异同,它们各应用于什么场合。

7.6 指出下列存储器各有多少存储单元,应设置几根地址线和数据线。

(1) 64K×1

(2) 256 K×4

(3) 1M×1

7.7 设存储器的起始地址为全 0, 试指出下列存储器的最高地址为多少? 分别用二进制和十进制表示最高地址。

(1) $2\text{K} \times 1$

(2) $16\text{K} \times 4$

(3) $256\text{K} \times 16$

7.8 试确定用 ROM 实现下列逻辑函数所需容量:

(1) 实现两个 4 位二进制相乘的乘法器。

(2) 将 8 位二进制数转换为 8421BCD 法的转换电路。

7.9 试用 TTL-ROM 实现 4 位二进制码至格雷码的转换电路。

7.10 64×4 ROM 的逻辑框图示于图 P7.1, 试画出由它构成 128×8 ROM 的连接图。

7.11 一个 4096 位 DRAM, 存储矩阵采用 64 行 \times 64 位结构。设每个存储单元刷新时间为 400 ns, 问需多少时间才能将全部存储单元刷新一遍。

7.12 试用图 P7.1 所示 64×4 ROM 扩展为 256×8 ROM。

7.13 图 P7.2 所示组合电路, 若用 ROM 实现, 其 ROM 容量为多大?

7.14 试用 ROM 设计产生图 P7.3 所示 4 路周期信号的逻辑电路, 画出逻辑框图。

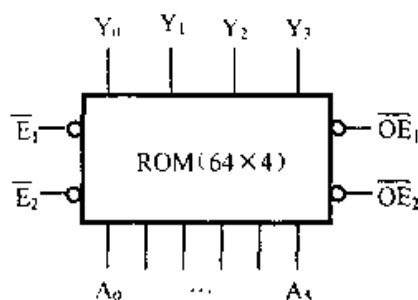


图 P7.1

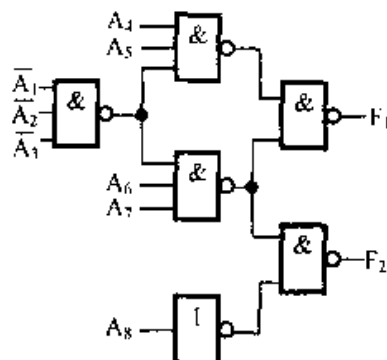


图 P7.2

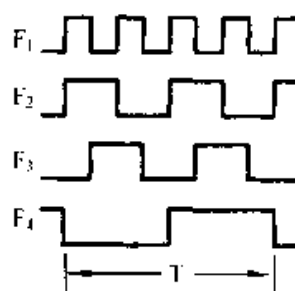


图 P7.3

第 8 章 可编程逻辑器件

内容提要 可编程逻辑器件是一种由用户定义和设置逻辑功能的器件，自 20 世纪 70 年代后期开始出现，至今不过短短 20 多年。但是，由于这类器件集成度和功能密度高、设计灵活方便、工作速度快、可靠性高及保密性强等优点，发展速度十分迅猛。目前，由用户编程就可获得板级、甚至系统级芯片。所以应用领域日渐扩大，已经成为设计数字系统的首选器件，其市场占有率逐年增加，而且这种发展趋势仍以极快速度继续上升。本章重点介绍 GAL、FPLA、CPLD 和 FPGA 等可编程逻辑器件的电路结构、工作原理、性能指标以及编程方法。

8.1 可编程逻辑器件的结构和分类

8.1.1 PLD 器件表示方法

PLD 器件一般属于 LSI 或 VLSI 器件之列，如已投放市场的 FPGA XC3090 内含有 9000 个等效门。所以，一般的逻辑电路表示方法很难描述它们的电路结构，也不便于表示内部配置与逻辑图之间的关系，因此，对 PLD 器件已有一些新的表示方法^[16~20]。

1. 基本门的符号

输入或输出缓冲器、反馈缓冲器都采用互补输出结构，如图 8.1.1(a)所示^①。与门表示方法如图 8.1.1(b)所示。输入变量称为输入项，输出函数称为乘积项；或门表示方法见图 8.1.1(c)，它的输出 G 称为和项。

2. 连接符号

PLD 电路中阵列交叉点上的连接有 3 种表示方法，如图 8.1.2 所示，其中，硬连线连接即为固定连接；编程连接，即通过编程可以使该交叉点连接起来。

3. 缺省状态符号

与门的全部输入项都通过编程接通，称为与门的缺省状态。如图 8.1.3 中，乘积项 $D = A \cdot \bar{A} \cdot B \cdot \bar{B} = 0$ ，即 D 为缺省状态。缺省状态符号如乘积项 E 所示(与门符号上加“×”)。显然乘积项 $D = E = 0$ 。但是，必须注意乘积项 F 与它们不同，因为没有一

① 为了方便使用，本章采用目前国际、国内流行的图形符号。

个输入项接通(即输入端都“悬空”),所以,乘积项 F 为“悬浮”的 1 状态。

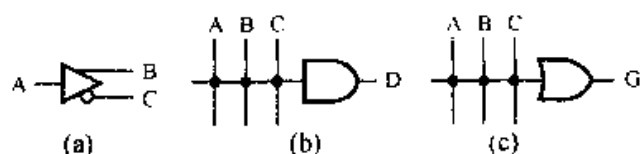


图 8.1.1 基本门符号

(a) 缓冲器 (b) 与门 (c) 或门

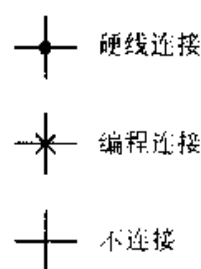


图 8.1.2 连接符号

4. 编程控制的多路开关表示方法

编程控制的 2 选 1 多路器如图 8.1.4 所示,其中,图(a)与图(b)是等效的,只是图(b)省略了控制线,表示由用户编程确定选 A 或 B 输出,即称为编程控制的多路开关。同理,图(c)表示编程控制的 3 选 1 多路器。

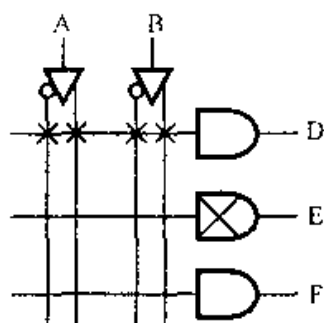


图 8.1.3 缺省状态符号

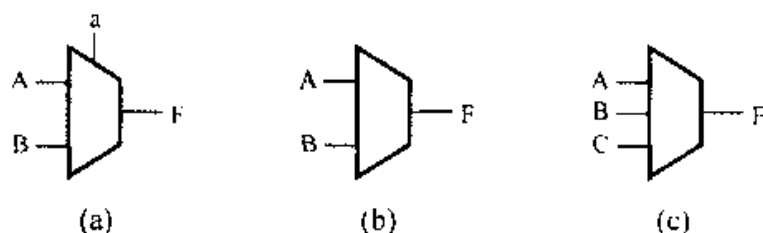


图 8.1.4 编程多路开关符号

(a) 二路开关之一 (b) 二路开关之二 (c) 三路开关

5. 极性表示方法

图 8.1.5 表示编程控制的 2 路开关的输入是反相的,由编程来选择 CK 或 \overline{CK} 信号输出。

6. 多路控制的三态缓冲器

图 8.1.6 表示受到两路控制的三态缓冲器,其中, a 是控制它的选通信号; b 是控制另一个可更改的功能,例如,转换速度为快与慢两种,用户对 b 编程可以实现缓冲器是快速或是慢速转换工作。

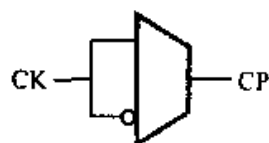


图 8.1.5 编程极性控制

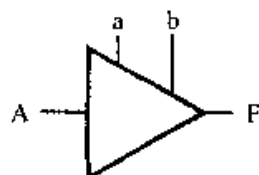


图 8.1.6 编程多路控制三态缓冲器

8.1.2 PLD 的基本结构

PLD 器件种类繁多,但它的基本结构只有 3 种,在它们的基础上再增加一些输入或输出宏单元、一些反馈或控制电路及一些辅助功能电路,便可构成应用范围非常广泛的各種类型 PLD 器件。

1. PROM 结构

这种结构的与门阵列是固定连接,而或门阵列是可编程的。前面介绍的 ROM 器件即采用这种结构。由图 8.1.7 所示编程后的电路可写出 3 个和项为

$$O_0 = \bar{I}_2 \bar{I}_1 \bar{I}_0 + I_2 \bar{I}_1 \bar{I}_0$$

$$O_1 = \bar{I}_2 I_1 \bar{I}_0 + I_2 \bar{I}_1 \bar{I}_0 + \bar{I}_2 \bar{I}_1 I_0$$

$$O_2 = I_2 \bar{I}_1 \bar{I}_0 + I_2 I_1 \bar{I}_0 + I_2 I_1 I_0$$

与门阵列愈大,开关延迟时间愈长,速度就愈不高。所以,与阵列较大时,只能作存储器使用,此时输入项 $I_0 \sim I_2$ 即为地址,而输出 $O_0 \sim O_2$ 即为输出数据。只有小规模与阵列才能有效地构成 PLD 器件,因为这时开关延迟时间短,速度就较高。

2. PAL 结构

这种结构的与门阵列是可编程的,而或门阵列是固定连接的,即每个输出和项是固定为若干乘积项之和。如图 8.1.8 所示。每个和项由相应的两个乘积项构成,而每个乘积项由编程确定。

由图中的编程结构可以写出 3 个和项为

$$O_0 = I_1 \bar{I}_0 + \bar{I}_2 \bar{I}_1 \bar{I}_0$$

$$O_1 = I_1 I_0 + \bar{I}_2 I_0$$

$$O_2 = I_2 \bar{I}_1 I_0 + I_2 \bar{I}_1 \bar{I}_0$$

3. PLA 结构

这种结构的与门阵列及或门阵列都是可编程的,如图 8.1.9 所示。由图中的编程结构可以写出 3 个和项为

$$O_0 = I_2 \bar{I}_1 + I_2 I_1 I_0$$

$$O_1 = \bar{I}_2 \bar{I}_1 \bar{I}_0 + I_2 \bar{I}_1 I_0 + I_2 I_1 I_0$$

$$O_2 = I_2 I_1 + \bar{I}_2 \bar{I}_1 \bar{I}_0$$

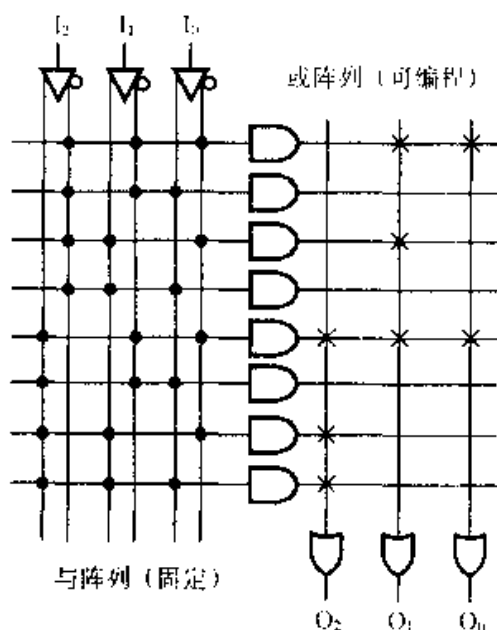


图 8.1.7 PROM 结构图

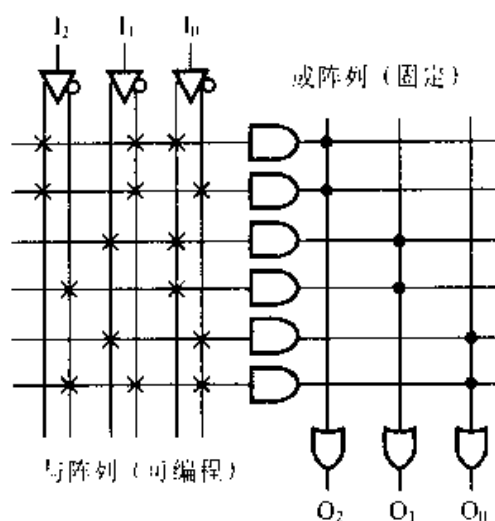


图 8.1.8 PAL 结构图

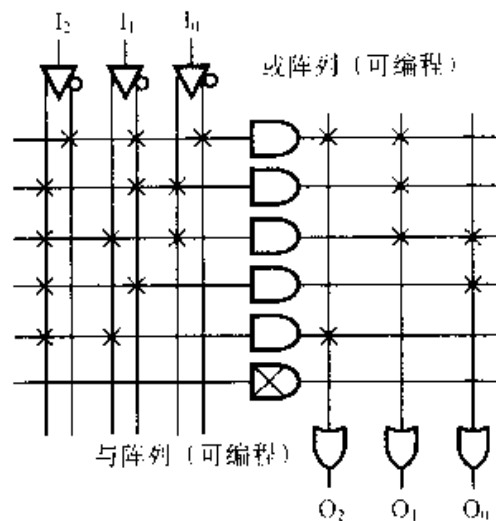


图 8.1.9 PLA 结构图

8.1.3 PLD 的分类

目前，PLD 器件种类繁多，而且国内资料中的名称和分类也不完全一致，本书是依据器件采用的基本结构和所实现的逻辑功能进行分类，将其分为以下几种主要类型。

1. 可编程只读存储器和随机存储器

这种器件从 20 世纪 70 年代问世以来，是 PLD 中最先开发的器件，它的主要用途还是作存储器使用，至今仍然保持着旺盛的生命力。在第 7 章中已作过详细介绍。

2. 可编程阵列逻辑(PAL)

PAL 器件是 20 世纪 70 年代末期出现的面向用户的 PLD 器件。目前的产品中，每个或门有 7~8 个乘积项，输出设置了 4 种电路结构可供选择。PAL 系列品种繁多，覆盖面宽，用户可依据实际需要灵活选择最佳的 PAL 器件。它还增设了一个保密熔丝(保密位)，在验证编程正确无误后将其“烧断”，便无法读出内部编程内容，可有效地防止电路泄密。这类器件采用熔丝工艺，所以是一次性编程器件，编程完成后，其内容就不能更改了。目前，由于 GAL 器件可与之兼容，在各项指标及使用方法上都优于它，因而 PAL 应用范围日渐缩小了，所以本书不作重点介绍。

3. 可编程通用阵列逻辑(GAL)

GAL 器件是 1985 年开始推出的一种新型 PLD。它采用 PAL 结构，在 PAL 的基础上增加了供用户编程组态的输出宏单元，因此，设计逻辑电路的灵活性大大提高了。它采用电擦除工艺(E^2 CMOS 工艺)，可以多次改写内容，并且可与 PAL 兼容。由于这些优点，它已在许多应用领域取代了 PAL 器件。

4. 可编程逻辑阵列(PLA)

PLA 器件采用 PLA 结构, 由于与阵列及或阵列都可由用户编程, 所以设计灵活性更大。这类器件早在 20 世纪 70 年代便开始研制, 但主要产品是面向工厂编程的掩模 PLA, 即用户依据自己的程序绘出“PLA 阵列图”交给工厂, 由工厂用掩模工艺做出二极管图像。它适用于大批量定型产品, 显然它是一次性编程器件, 不适用于产品开发。面向用户编程的现场可编程逻辑阵列 FPLA 在 20 世纪 80 年代中期开始出现, 并且融入了 GAL 器件中的技术, 具有 GAL 器件的优点, 因而厂商和许多资料将 FPLA 产品, 如 G39V18、GAL6001 等都列入 GAL 器件之中。也有资料称为 PLA 型 GAL, 这些产品都是采用 PLA 电路结构, 即与阵列及或阵列均可编程, 应归类于 FPLA 更合理。

5. 复杂可编程逻辑器件 (CPLD)

前面介绍的几种 PLD 器件的集成度和功能密度都较低, 属于低密度可编程逻辑器件, 一般只能完成逻辑部件的功能, 而不能构成一个数字系统。

随着集成工艺和微电子技术的发展, 从 20 世纪 80 年代中期开始, 出现了采用 CMOS 工艺、EPROM、E²PROM、FLASH 和 SRAM 等编程技术构成了高密度可编程逻辑器件, 可以实现一个中等规模的数字系统, 是一种系统级芯片。但是从电路结构上来看, 这类器件还是在 GAL 基础上加以改进和扩展而成的。

6. 现场可编程门阵列(FPGA)

FPGA 器件是 1985 年推出的新一代产品, 属于高密度 PLD 器件, 也是一种系统级芯片, 但是, 它的电路结构与 GAL 及其 GAL 扩展型器件不同, 它由许多可编程逻辑单元组成, 用户可通过对这些逻辑单元编程来实现逻辑功能, 又通过编程技术将这些逻辑单元连接成所需要的数字系统。

8.1.4 可编程逻辑器件的优点

面向用户的 PLD 器件自 20 世纪 70 年代末期出现以来, 由于它设计的灵活性和应用的方便性, 已越来越受到用户的青睐。它的应用范围日渐扩大, 如计算机主板及各种功能卡, 单板机开发系统, 软件保密卡, 无线电话及通信设备等。其市场占有率逐年增加, 其原因在于它具有标准器件无法比拟的优点, 归纳起来有以下几个方面。

1. 功能密度高

功能密度是指在给定的芯片面积中可集成的逻辑功能, PLD 器件的功能密度远远高于中、小规模标准器件。目前, 一般 PAL 和 GAL 器件可代替 5~20 片中、小规模标准器件, 而 FPGA、CPLD 可代替几百片。由于系统的芯片数量大大减少, 系统体积、成本、产品开发周期都会下降, 而系统的性能和可靠性却大有提高。

2. 设计灵活方便

采用可多次编程的 PLD 器件进行产品开发,能给用户带来设计上的极大灵活性,用户通过编程来实现逻辑功能,又可进行仿真与修改,直到用户满意为止。这些都是借助开发工具在计算机上进行的,所以,设计、仿真以及修改都非常方便,可以发挥设计者的独创性,开发出最佳产品。系统级芯片 CPLD 和 FPGA 的出现与应用,已将数字系统的设计、加工和调试融为一体,这就彻底更新了传统的设计和生产方式,具有开发费用低廉、研制周期短和研制人员劳动强度低等优点。

3. 系统可靠性高

由于系统所包含的芯片及印制板的数量减少了,芯片的布线以及印制板之间的连线均得到简化,因而提高了系统可靠性。

4. 保密性强

目前广泛使用的 PLD 器件都增加了加密位,在系统研制成功后将其“烧断”,芯片内的编程数据便无法读出,因而系统具有良好的保密性能。

8.2 可编程通用阵列逻辑(GAL)

GAL 器件是在 PAL 器件基础上发展的新型器件,它们的基本结构是相同的,都是采用 PAL 型结构,即与阵列可编程,而或阵列是固定连接。不同的是 PAL 的输出是几种固定组态结构供不同型号产品选用,因而 PAL 的品种繁多;而 GAL 的输出设置了可编程宏单元,由用户编程来确定输出宏单元的组态结构,设计灵活又方便,并且品种大为简化。GAL 采用 E^2 CMOS 工艺,可以方便地擦除和改写内容,因而比 PAL 更适合在研制开发过程中应用。目前,厂商已做到使 GAL 与 PAL 完全兼容(包括功能、熔丝图以及引脚等),所以, GAL 已逐渐取代了 PAL。

目前,市场出售的 GAL 依其电路结构分为两类:

一类采用 PAL 型结构,如 G16V8、G20V8、ispG16V8 都是属于这一类,这类产品具有相同的通用结构,其差别仅在于 G20V8 扩展了与阵列,而 ispG16V8 增加了在线编程和实时诊断功能;

另一类采用 PLA 型结构,与阵列及或阵列都是由用户编程。属于这一类的产品有 G39V18 和 GAL6001。

但是,正如前文所述理由,PLA 型 GAL 应归类于 FPLA 器件。

本节以典型产品 G16V8 为例对 GAL 器件的通用结构进行重点介绍,然后介绍编程中应该注意的几个技术问题。

8.2.1 GAL 器件的电路结构

1. GAL 的通用结构

不同型号的 GAL 器件的电路结构类似，只是个别辅助功能有差异。G16V8 电路结构就是一种典型的通用结构，如图 8.2.1 所示。在工程应用中常常只画出它的输入和输出部分，称为功能框图。电路中包含输入缓冲器（左面 8 个）、输出三态缓冲器（右面 8 个）、与阵列、输出反馈/输入缓冲器（中间 8 个）、输出逻辑宏单元 OLMC (Output Logic Macro Cell)，单元中含有或阵列，此外还有时钟和输出选通信号的输入缓冲器（1 脚、11 脚）。与阵列有 64 个与门（ 8×8 乘积项），每个乘积项有 32 个输入项，故与阵列通常表示为 64×32 阵列。CLK 为公共时钟，OE 为公共三态控制。G16V8 有 8 个引脚（2~9）作固定输入，还可能有其他 8 个（最大值）引脚配置成输入，因此，最多有 16 个输入，而输

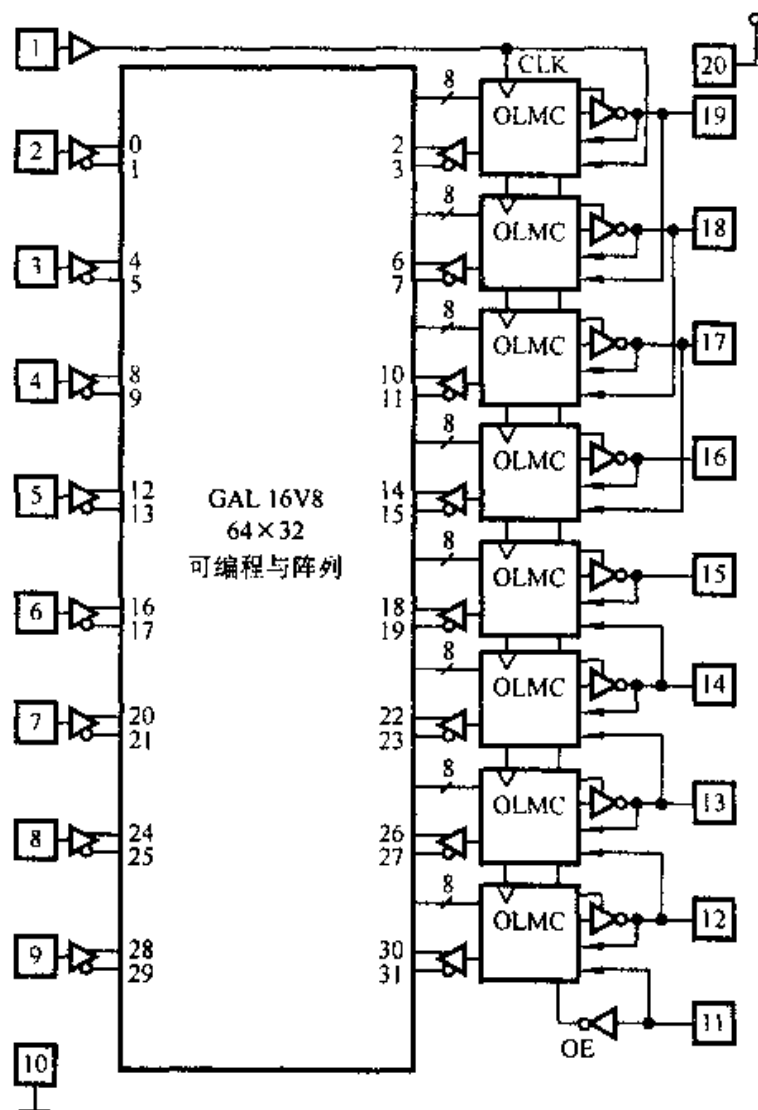


图 8.2.1 GAL 16V8 功能框图

出最多为 8 个。这就是型号中两个数字(16, 8)之含义(G39V18 除外)。逻辑图如图 8.2.2 所示。

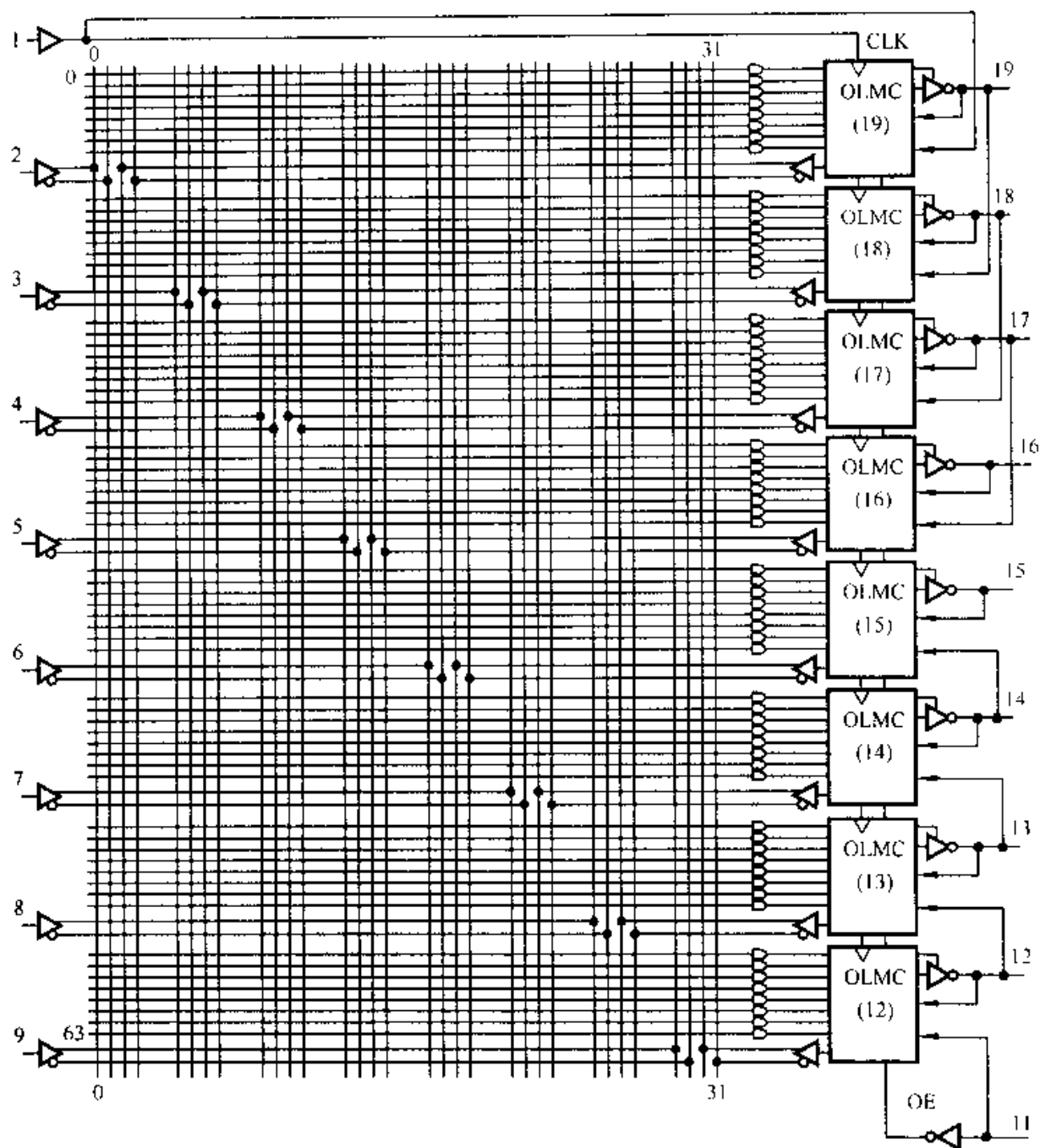


图 8.2.2 GAL 16V8 逻辑图

2. 输出逻辑宏单元(OLMC)

OLMC 内部结构示于图 8.2.3。它由一个 8 输入或门构成或阵列，每个或门输入由一个乘积项提供，因此，或门输出(和项)为有关乘积项之和；异或门用来控制输出信号极性，当 $XOR(n)$ 为 1 时(n 为芯片的引脚号)，异或门完成输出反相功能。D 触发器对异或门输出实现寄存功能，使 OLMC 能适用于时序电路。图

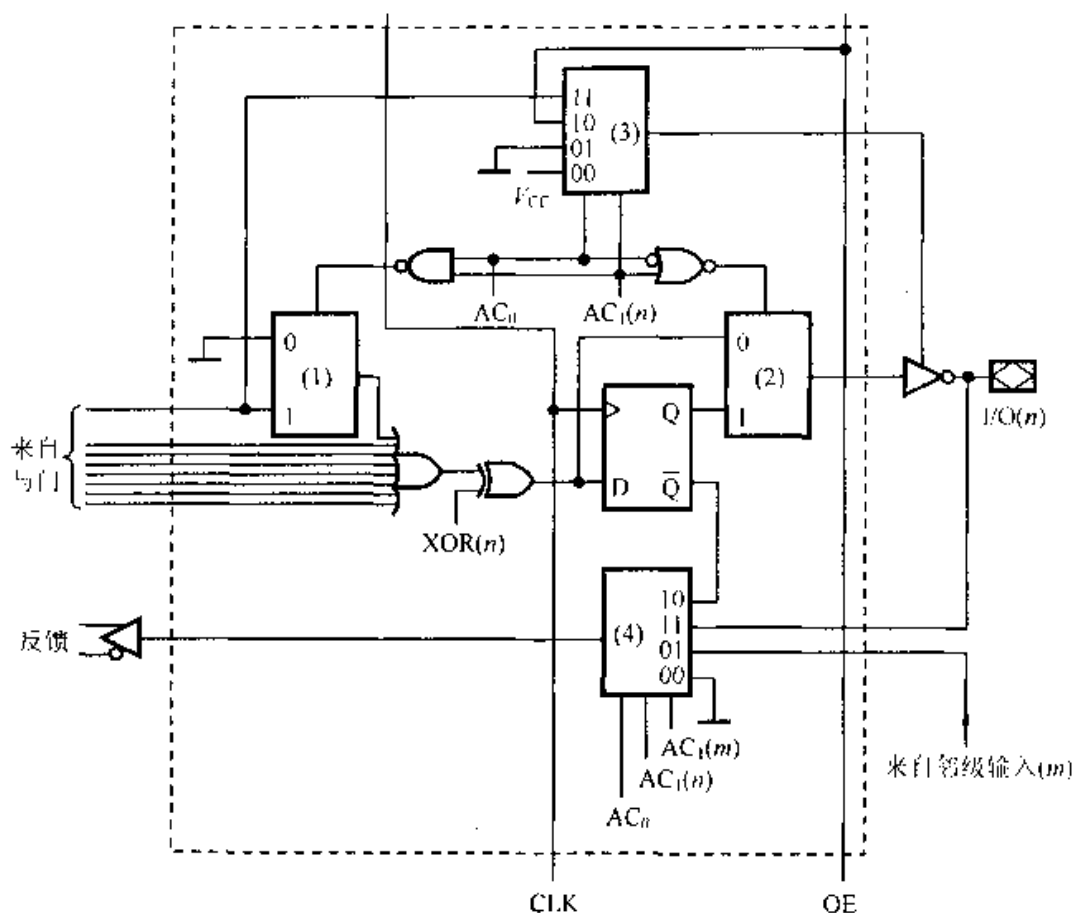


图 8.2.3 OLMC 内部结构图

图中，(1)为乘积项多路开关 PTMUX；(2)为输出多路开关 OMUX；

(3)为三态多路开关 TSMUX；(4)为反馈多路开关 FMUX

中含有 4 个多路开关(即两个 4 选 1 多路器，两个 2 选 1 多路器)，它们是

(a) 乘积项多路开关 PTMUX。它用于控制第一乘积项 PT1 的功能。当选择 1 通道时，PT1 成为或门的一个输入，故或门有 8 个输入项；反之，或门只有 7 个输入项。

(b) 输出多路开关 OMUX。它用来选择输出信号是组合输出(选 0 通道)，或是寄存(又称时序)输出(选 1 通道)。

(c) 三态多路开关 TSMUX。它选择输出三态缓冲器的选通信号，由 AC_0 和 $AC_1(n)$ 的编程确定。顺便提醒读者注意：图中多路器的输入通道是用二进制编码表示，并且这个编码与地址相对应，这样标注可以方便读图。

(d) 反馈多路开关 FMUX。它用来选择反馈信号，也是由 AC_0 、 $AC_1(n)$ 编码确定($AC_1(m)$ 的功能另有说明)。

3. 行地址映射

图 8.2.4 是行地址映射图。行地址 0~31 对应与阵列，每一行有 64 位数据(注意数据配置)；第 32 行为电子标签字，它是 64 位可编程存储器，用于存储用户定义的数据。

据：第 33~59 行为厂商保留，用户不能使用；第 60 行为结构控制字，共有 82 位；第 61 行为加密位，只含一位；第 62 行为厂商保留；第 63 行也只含一位，用于整体擦除。

4. 结构控制字(ACW)

结构控制字共有 82 位，它对于正确使用 GAL 是至关重要的，因为它控制 OLMC 的组态结构和输出极性。换言之，它控制了芯片能实现的逻辑功能。图 8.2.5 是该字的各位数据定义及其配置图。AC₀(一位)是 4 个多路开关的控制码，应特别注意的是它为 8 个 OLMC 所公用；AC_i(n)是 4 个多路开关的另一个控制码，应注意的 n

为引脚号，与 OLMC 对应，故有 8 位；SYN 是时序控制位，只有一位，它描述整个芯片是实现时序电路还是组合电路。只要 8 个 OLMC 中有一个为时序(寄存)输出，则应令 SYN 为 0；只有芯片的输出均为组合输出，才能令 SYN 编程为 1。该位对 8 个 OLMC 的组态结构有很重要的作用；XOR(n)是控制异或门的输出极性，共有 8 位。例如，XOR(12)便是控制引脚 12 上的 OLMC 中异或门输出极性；64 位乘积项禁止位分别对应与阵列中 64 个乘积项 PT₀~PT₆₃，每一位可确定相应乘积项是否编程，为 1 表示要编程，为 0 表示无需编程，有利于加快编程速度。

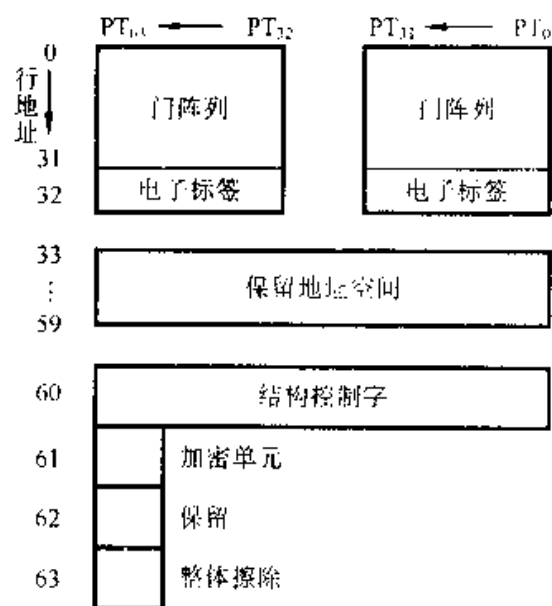


图 8.2.4 行地址映射图

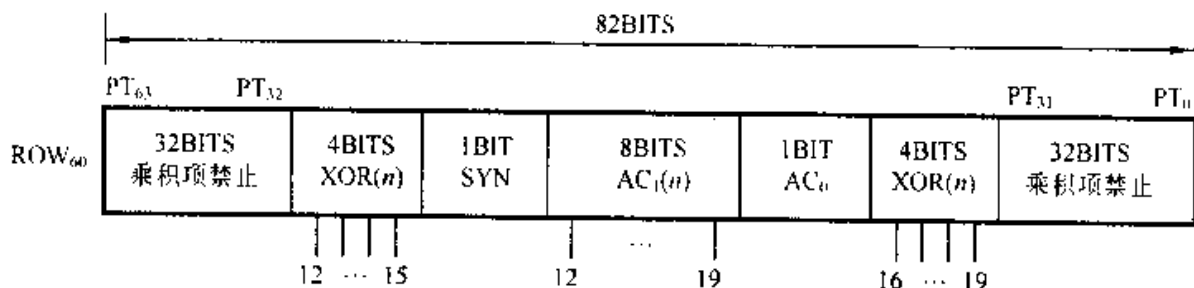


图 8.2.5 结构控制字

5. 电子标签字(ES)

电子标签字 ES 共有 64 位，存储用户定义的数据，这些数据与保密位无关，随时可以读出。用户定义时，最好是把它分为几个字段，并保留一些字段供将来扩充之用，常用的信息字段信息有厂商识别代码、编程日期、程序员代码、电路结构代码等。用户可以分配字段数，也可以配置每个字段的位数，应用十分灵活方便。图 8.2.6 是一种 ES 的配置图可供参考。

当 GAL 整体擦除时，ES 行也同时被擦除，在重新写入内容时应写入新的 ES 字。

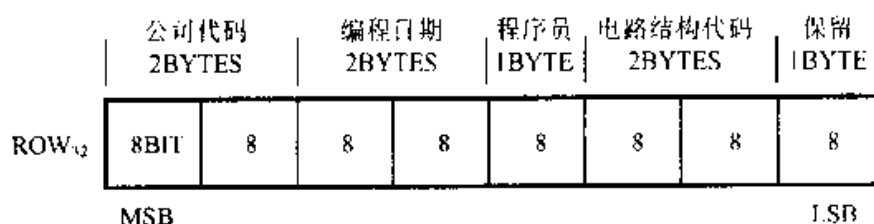


图 8.2.6 电子标签段配置示例图

ES 字的设置不仅可省去手工标签,降低成本和缩短产品开发周期,而且有利于资料管理、软件修改及产品质量跟踪等。例如,在开发过程中,为了做到精益求精,经常出现多种改进版本,这时只要每一版本都在 ES 中写入相应识别码。用它便可对其他版本芯片进行检验,如有人将老版本芯片插入用户系统而造成性能指标下降,只要调出它们的 ES 字,核对电路结构标识码,即可发现插错芯片了。

6. 加密单元

加密单元只有一位,一旦加密,就禁止程序读出,因而无法了解芯片内电路结构,可以防止电路泄密。只有在整体擦除时才能将该单元擦除。

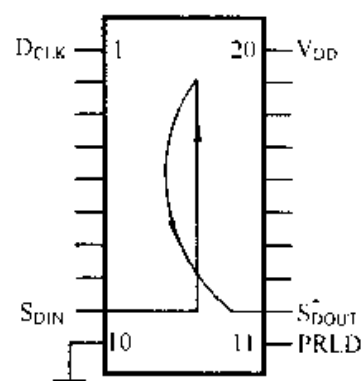
7. 整体擦除位

在编程周期中,对该行寻址并执行擦除功能,就可同时擦除门阵列、ACW 字、ES 字和保密位,即返回到起始状态(相当于清除功能),故称整体擦除。

8. 输出寄存器的预置

因 GAL 中有一个完成时序电路的逻辑部件,显然对它进行自启动功能的检验是至关重要的。为此,应将它强迫进入无效状态,检验在工作条件下能否进入正常运行状态。在 GAL 中设置了一个电路,可以断开反馈电路,使输出寄存器可以预置为任意状态,当然无效状态也包含在其中。图 8.2.7 是寄存器预置的引脚图。当预置脚 PRLD(11 脚)加 15V 电压,输入及输出缓冲器同时被选通,然后在时钟 D_{CLK} 上升沿出现时,数据便移入寄存器,预置操作完成。

应该指出的是预置数据是由下至上移位的,只有配置为时序输出的 OLMC(是否相邻无关)才被预置。例如,一个十进制计数器,有 4 个 OLMC 被配置为时序输出,那么,只要 4 个时钟便可将预置状态移入相应寄存器。若令预置状态为无效状态,则在断开 PRLD 脚的高压后,芯片恢复正常工作状态便可检验其自启动功能。应注意 S_{DOUT} 是漏极开路输出,使用时应加外接电阻(10kΩ)至 V_{DD}。有些资料将这种状态预置称为预加载功能。



*该引脚通过 10kΩ 电阻接至 V_{DD}

图 8.2.7 输出寄存器预置引脚配置图

9. 上电复位

GAL 中设置有上电复位电路。上电后, 所有寄存器 Q 端被置为 0 电平。假如这些寄存器被 OE 选通, 则相应输出引脚便保持在 1 电平, 而与 XOR(*n*)无关。反馈源均来自 \overline{Q} 端, 故保持在 1 电平。由于复位后这些状态为已知的, 所以、有利于时序电路的设计简化。

8.2.2 输出逻辑宏单元(OLMC)的组态结构

OLMC 的组态结构由 3 个控制参量确定, 即 SYN, AC_0 , $AC_1(n)$ 。必须注意在一个芯片的 ACW 字中, 只设置了 8 个 OLMC 公用的 SYN 和 AC_0 , 而 $AC_1(n)$ 是分别设置的控制参量。这一点对于设计 OLMC 组态结构非常重要。对于 3 个参量而言, 有 8 种不同组合, 所以, 应有 8 种工作模式。但是, 由于 SYN、 AC_0 是公用的, 故其中有些工作模式是不可能存在的。例如, 在 $SYN \cdot AC_0 \cdot AC_1(n)$ 为 000 和 001 模式中, $SYN=0$, 即表示 8 个 OLMC 中至少有一个为时序输出, 并且该单元的三态输出(见图 8.2.3)由输出选通 OE 控制, 为此应令 $AC_0 \cdot AC_1(n)=10$ 。这显然与 000 和 001 模式中要求 $AC_0=0$ 冲突, 因而这两种模式不可能存在。同理 110 模式也不可能存在(原因留给读者分析)。因此, 只有 5 种工作模式, 它们的组态结构及其特性如表 8.2.1 所示。

表 8.2.1 OLMC 5 种组态模式表

$SYN \cdot AC_0 \cdot AC_1(n)$	配置功能	芯片逻辑功能	单元逻辑类型	说 明
101	专用输入模式	组合型	组合单元	本单元未输出: 芯片为组合型, 反馈来自邻级输出
100	专用输出模式	组合型	组合单元	本单元仅作输出: 芯片为组合型; 反馈多路开关可被邻级借用
111	组合 I/O 模式	组合型	组合单元	本单元引脚为 I/O, 芯片为组合型, 反馈来自本级输出, 或门输入只有 7 个乘积项
010	时序模式	时序型	时序单元	芯片为时序型, 输出三态由 OE 选通; 反馈来自本级 \overline{Q}
011	时序 I/O 模式	时序型	组合单元	本单元为组合, 它的引脚为 I/O, 输出缓冲器由第一乘积项控制; 芯片为时序型; 反馈来自本级输出

1. 101 模式

简化电路如图 8.2.8(a)所示。该单元输出为高阻态, 本单元不能输出, 但其

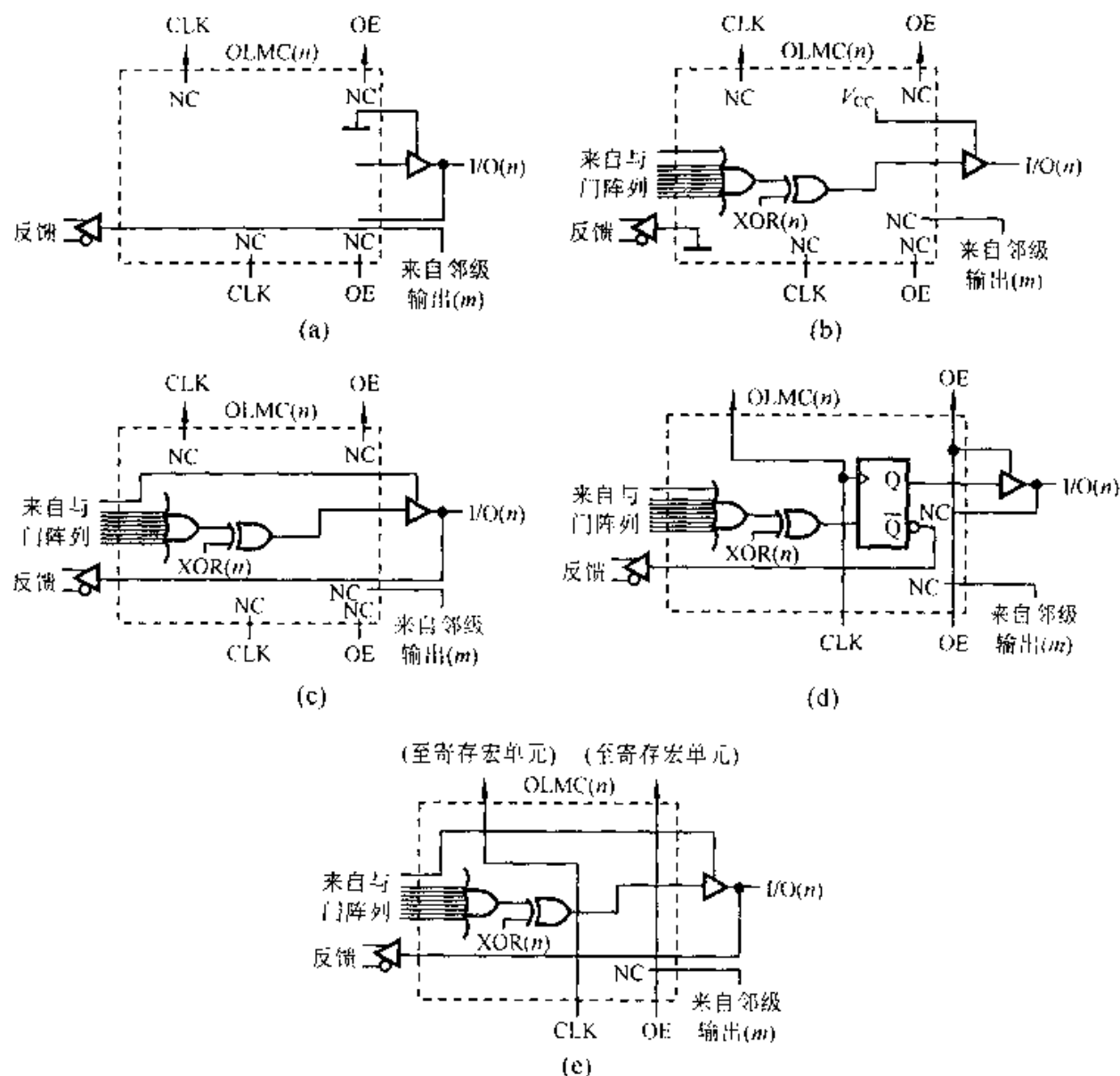


图 8.2.8 OLMC 组态结构图

(a) 专用输入模式 (b) 专用输出模式 (c) 组合 I/O 模式
(d) 时序输出模式 (e) 时序 I/O 模式

他单元可配置成组合输出, 故芯片为组合型。1 脚与 11 脚(CK 和 OE 线)均为输入数据端, 反馈来自邻级输出。故称它为相邻输入模式(专用输入模式)。

2. 100 模式

简化电路示于图 8.2.8(b)。本单元为组合输出, 引脚不能作输入使用, 其他单元只能配置成组合输出, 故芯片为组合型(因为 $SYN=1$)。1 与 11 脚可作输入使用, 反馈开关未用, 可被邻级借用。由于本单元引脚不能作输入, 故称之为输出模式(专用输出模式)。

3. 111 模式

简化电路如图 8.2.8(c)所示。本单元既可作输入, 又可作输出, 由该单元的

一乘积项确定。由于第一乘积项已作选通控制使用,故或门只有 7 个乘积项。本单元引脚为 I/O,反馈来自本级的输出,芯片为组合型。由于引脚为 I/O 而芯片为组合型,故称之为组合 I/O 模式(选通组合输出模式)。

4. 010 模式

由于 $SYN=0$,芯片中至少有一个为时序输出,又由于 $AC_0 \cdot AC_1(n)=10$,故本单元为时序输出。其他单元可配置成时序输出(010 模式),也可配置成组合输出(011 模式)。因本单元为时序输出故称之为时序模式(存储输出模式)。简化电路如图 8.2.8(d)。

5. 011 模式

本单元为组合输出,输出缓冲器由第一乘积项控制,所以本单元引脚为 I/O,反馈来自本级输出。或门输入只有 7 个乘积项。CK 和 OE 提供其他单元使用,芯片为时序型。由于引脚为 I/O,芯片为时序型,故称为 I/O 时序模式。简化电路示于图 8.2.8(e)。

*8.2.3 编程应用中的几个技术问题

GAL 的编程就决定了该芯片的逻辑功能,因此,在编程中对于工作模式的选用、反馈开关的借用、某些 OLMC 单元的特殊性及限制条件等必须给予足够重视。否则,就会出现错误,难以达到预期目的,或者产生设计水平较低的芯片。下面介绍几个在 GAL 编程中经常碰到的问题供读者参考。

1. OLMC 组态模式的制约与兼容

OLMC 的工作状态由 ACW 字中的 18 位数据唯一确定(参见图 8.2.5)。但是,这些代码的取值并非可以任意选取,它们之间存在着多种制约与兼容关系。对此必须给予重视。18 位代码中 $XOR(n)$ 是决定输出信号极性,它对组态(工作)模式不产生影响,所以,主要还是讨论 SYN , AC_0 , $AC_1(n)(n=12\sim19)$ 这 10 个代码取值关系。在前文介绍工作模式时已有涉及,但对 GAL 编程应用极为重要,因此,有必要进一步讨论。

OLMC 有 5 种组态模式,但并不是对 8 个 OLMC 可以任意选用,其原因是芯片中只设置了公用的 SYN 和 AC_0 ,因而每个 OLMC 组态模式的选用就受到某些制约。例如,本单元已选用 100 模式构成组合输出,由于 $SYN=1$,那么,就限制了其他单元不能选 010 和 011 模式;但是否就可选用 101 和 111 模式呢?前者可以选用,而 111 模式不能选用,因为 $AC_0=1$,它与已使用的 100 模式($AC_0=0$)无法兼容;同理,若本单元已选用 010 模式以构成时序单元,其他单元不能选用 100, 101, 111 等模式,但可选用 011 模式(当然还可选 010 模式,同类模式肯定兼容)。由此可归纳出选用模式的原则,即

(a) 同一芯片中, SYN 和 AC_0 相同的模式可以兼容,其他都是不兼容的。即 010

模式与 011 模式兼容；101 模式与 100 模式兼容；除此之外都是互相制约的。

(b) 111 模式与其他模式都不兼容，故只要有一个 OLMC 选用了它，则其他单元也只有选用 111 模式。

2. 相邻单元对反馈开关的借用

在图 8.2.8(b)所示的 100 模式中，反馈开关接地，从逻辑功能上它未被使用，因而可以提供给邻级借用。为了便于借用，GAL 中有如下规定。

规定 1 在 OLMC 工作为 100 模式时，如有邻级借用反馈开关，则反馈开关由 AC_0 、 $AC_1(m)$ (m 表示与 n 相邻) 控制，其中， $AC_0 = \overline{SYN} = 0$ ； $AC_1(m) = SYN = 1$ (图 8.2.3 中反馈开关的控制处同时标注 $AC_1(n)$ 和 $AC_1(m)$ ，其原因就在于此)。但是，必须注意其他 3 个多路开关的控制码仍为 $AC_0 \cdot AC_1(n) = 00$ 。

由于工作为 100 模式，这时 CK(1 脚)和 OE(11 脚)总是安排为输入脚，它们只有借用 OLMC₁₉ 和 OLMC₁₂ 的反馈开关进入与门阵列。为此，OLMC₁₉ 和 OLMC₁₂ 都必须工作于 101 模式 (参见图 8.2.8(a))，其引脚 19 和 12 可作输入。它们分别借用 OLMC₁₈ 和 OLMC₁₃ 的反馈开关，因而，它们必须工作于 100 模式才能提供这种借用。如此类推，直至引脚 17 和引脚 14 借用 OLMC₁₆ 和 OLMC₁₅ 的反馈开关。显然，引脚 16 和 15 已无相邻反馈开关可借用，因而，引脚 16 和 15 不能作为输入脚。故 GAL 中有：

规定 2 在使用 100 模式和 101 模式的芯片中，无论 OLMC₁₆ 和 OLMC₁₅ 的反馈开关是否被借用，引脚 16 和 15 规定只可作输出使用 (此规定另一原因是为了与 PAL 器件兼容)。

在实际编程中，若违反上述规定，编译软件将不接受。例如，图 8.2.9 就是借用反馈开关的示意图。OLMC₁₉ 工作在 101 模式，1 脚信号借用它进入与阵列；19 脚输入便只有借用 OLMC₁₈ 的反馈开关，因此，必须令它工作在 100 模式，其反馈开关控制码为 01 (规定 2)，18 脚作为输出。在本例中并没有借用 OLMC₁₆ 和 OLMC₁₅，但如果设计者把 16 和 15 脚作输入使用，编译软件将不接受。

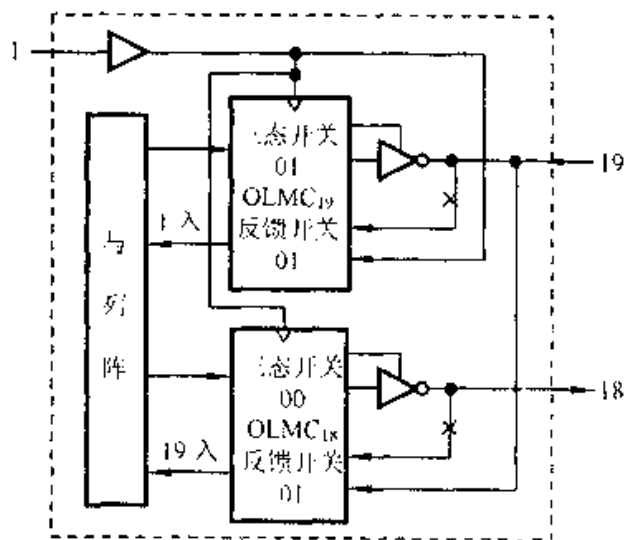


图 8.2.9 借用反馈开关示意图

3. 或门阵列的扩展

目前 GAL 产品的或门阵列输入项只有 7~8 个 (与工作模式有关)，即输出函数中的乘积项不能超过 8 个。然而，在复杂的系统中也会出现超过的情况。这时就需要进行或门阵列扩展。有两种方法进行这种扩展。

(1) 内部扩展法

利用 OLMC 所具有的 I/O 功能, 用一个 OLMC 完成一部分乘积项之或, 然后反馈至与阵列, 用第二个 OLMC 完成所有乘积项之或。例如, $G=P_0+P_1+\cdots+P_9$ (P_i 表示一个乘积项)。

令 $OLMC_{18}$ 工作在 111 模式, 其引脚 18 为 I/O 引脚, 输出反馈至与阵列, 设它完成 $I/O_{18}=P_0+P_1+P_2$ 。

$OLMC_{17}$ 肯定工作在 111 模式, 令其实现 $I/O_{17}=G=I/O_{18}+P_3+\cdots+P_9=P_0+\cdots+P_9$ 。

这种方法的优点是无需外加器件, 其缺点是限制了芯片工作模式, 所有单元都为 111 模式, 或门输入只有 7 个乘积项。

(2) 外部扩展法

这种方法也是利用两个 OLMC 来完成或门阵列扩展, 但不利用 I/O 引脚反馈, 而且是外接一个或门来实现。它的优点是不影响芯片内其他单元的模式, 因而不必为扩展或门阵列又重新设计; 其缺点是硬件电路略为复杂。

无论采用哪种扩展方法, 都必须考虑扩展阵列会引入附加延迟, 芯片的最高工作频率会有所下降。

*8.2.4 用 GAL 设计逻辑电路

下面以设计 BCD 码计数器和 7 段译码显示电路为例, 介绍用 GAL 器件设计数字电路的基本方法。该电路只有一个输入信号, 即计数器的时钟 CLK, 其输出用以驱动 7 段译码器的 7 根信号线, 分别编号为 A, B, ..., G。设计的电路主要由两个部分组成, 一部分是十进制的计数器, 另一部分是 7 段数码管译码器; 十进制计数器的输出作为译码器的输入。

GAL 器件逻辑功能设计可采用 ABEL 语言实现, 此设计的 ABEL 语言源码如下所示:

```
MODULE Decoder
CLK                PIN;
A,B,C,D,E,F,G     PIN ISTYPE 'COM';
Q0,Q1,Q2,Q3       PIN ISTYPE 'REG';
Count={Q3..Q0};
LED=[A,B,C,D,E,F,G];

EQUATIONS
Count.CLK=CLK;
WHEN(Count==^H9) THEN Count:=0;
ELSE Count:=Count.FB+1;
```



```

truth_table(Count->LED)
0->[1,1,1,1,1,1,0];
1->[0,1,1,0,0,0,0];
2->[1,1,0,1,1,0,1];
3->[1,1,1,1,0,0,1];
4->[0,1,1,0,0,1,1];
5->[1,0,1,1,0,1,1];
6->[1,0,1,1,1,1,1];
7->[1,1,1,0,0,0,0];
8->[1,1,1,1,1,1,1];
9->[1,1,1,1,0,1,1];
.X->[0,0,0,0,0,0,0];

test_vectors([CLK]->[Count,LED])
@REPEAT 11 {[.C.]>[.X,.X];}
END

```

图 8.2.10 给出了该设计电路的工作波形,由图可以看到,当时钟上升沿到来时,计数器的值加 1; 数码管译码器输出信号与计数值相对应。

在 Synario 环境中,选择 CAL 器件 GAL26V12C,在 Process for current source 中双击 Fit Design,可完成 GAL 器件中的布局布线,并报告相关信息,如芯片资源利用率,输入/输出信号分配的管脚等。再选择 Create Fuse Map 生成熔丝图文件,用以对器件编程。器件顶层管脚分配如图 8.2.11 所示,由图可见,信号管脚与 ABEL 源程序中指定的管脚一致,其中 CLK 信号作为时钟输入,应指定在 GAL 器件的时钟输入端。

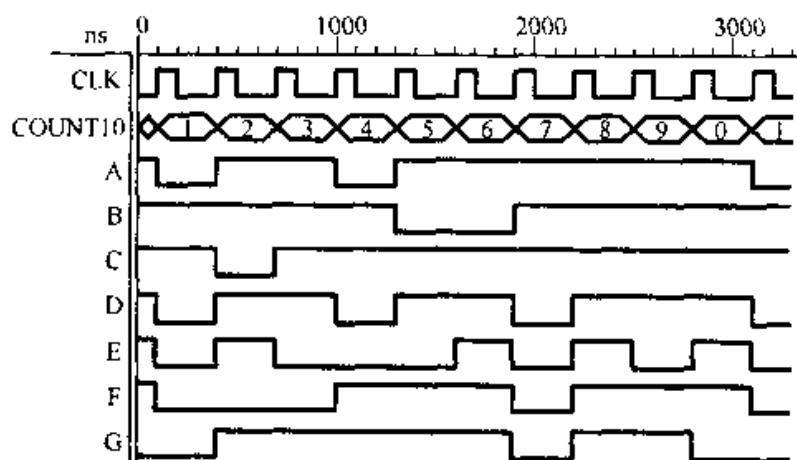


图 8.2.10 BCD 码计数器和 7 段译码
显示仿真波形图

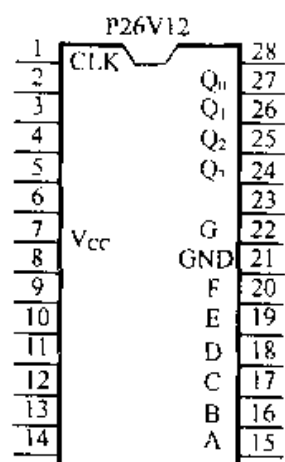


图 8.2.11 GAL P26V12 器件
顶层管脚分配

8.3 可编程逻辑阵列(PLA)

可编程逻辑阵列 PLA 是一类应用十分广泛的 PLD 器件。它采用 PLA 型结构，其与阵列及或阵列都可由用户编程，因而给功能更复杂的逻辑电路设计带来极大的设计灵活性。PLA 分为两类：一类是面向工厂的掩模 PLA，它适合于大批量定型产品，由工厂制作；另一类是面向用户的 FPLA，称为现场可编程逻辑阵列，适合于开发应用。它们的电路结构与工作原理是相同的，所以，本节主要介绍电路结构、工作原理以及掩模 PLA 的电路表示方法(PLA 阵列图)。

8.3.1 PLA 的工作原理

PLA 的与、或两阵列都是可编程的。所以，可以依据逻辑函数中所包含的乘积项进行与阵列编程，即与阵列产生所有需要的乘积项；然后对或阵列编程实现相应乘积项之或。例如，实现函数

$$F_1 = \overline{A}BC + B\overline{C} = P_1 + P_2$$

$$F_2 = \overline{B}\overline{C} + ABC = P_2 + P_3$$

$$F_3 = \overline{A}B\overline{C} + B\overline{C} = P_2 + P_4$$

用与阵列产生 3 个函数中所需要的乘积项 $\overline{A}BC, B\overline{C}, ABC, \overline{A}B\overline{C}$ ；或阵列实现每个函数的相应乘积项之或，如图 8.3.1(a)所示。

面向工厂的掩模 PLA 是由工厂采用掩模技术做出二极管(或其他存储元件)，显然这种常规表示方法及 PLD 中的表示方法都不方便。因此，采用更为简化的阵列逻辑图。它既便于表示规模更大及逻辑复杂的电路，又能使客户与厂商双方对 PLA 内部功能一目了然。

阵列逻辑图(简称阵列图)如图 8.3.1(b)所示，图中，6 条水平线表示输入线。将每个与门及或门都简化成一条线，即 4 条垂直线表示 4 个与门；3 条水平线表示 3 个或门。表示输入变量的水平线同代表与门的垂直线的交叉处标有“黑点”，就表示该处相应纵横线上接有二极管，即该变量参加相与；同理，表示乘积项的垂直线同代表或门的水平线的交叉处标有“黑点”，就表示该乘积项参加相或。符号“↓”表示与门输出；符号“→”表示或门输出。显然这种简化的阵列图便于绘制，同时又使人们对 PLA 内部功能一目了然。

用 PLA 逻辑阵列可以方便地设计组合电路及时序电路，只需完成阵列图的设计，便可交给工厂制造。

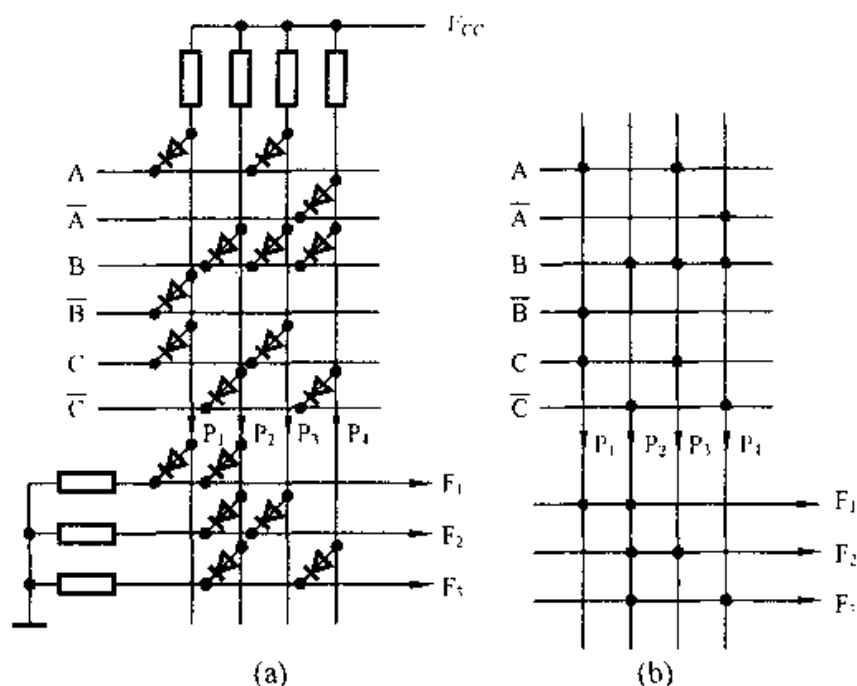


图 8.3.1 PLA 电路举例

(a) 电路图 (b) 逻辑阵列图

8.3.2 用 PLA 设计逻辑电路

1. 用 PLA 设计组合逻辑电路

任何组合逻辑电路都可以用一组逻辑函数的与或式来表示。因此，用与阵列产生所有与项，用或阵列产生每个函数的相应与项之或。图 8.3.2 是用 PLA 设计的一位全加器的阵列图。它是依据式(3.5.2)设计出来的，读者可分析它的设计过程。

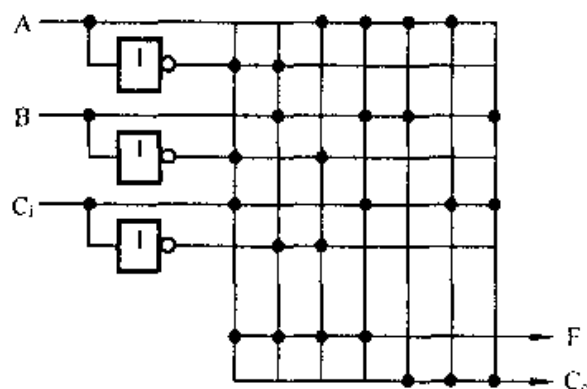


图 8.3.2 用 PLA 设计一位全加器

2. 用 PLA 设计时序逻辑电路

用 PLA 设计时序电路，除了与、或阵列外，还应增加一组触发器作为反馈网络。它的输入由或阵列提供；它的输出反馈至与阵列，并且同外输入 $X_1 \sim X_n$ 一起

产生所有需要的与项。逻辑框图如图 8.3.3(a)所示。

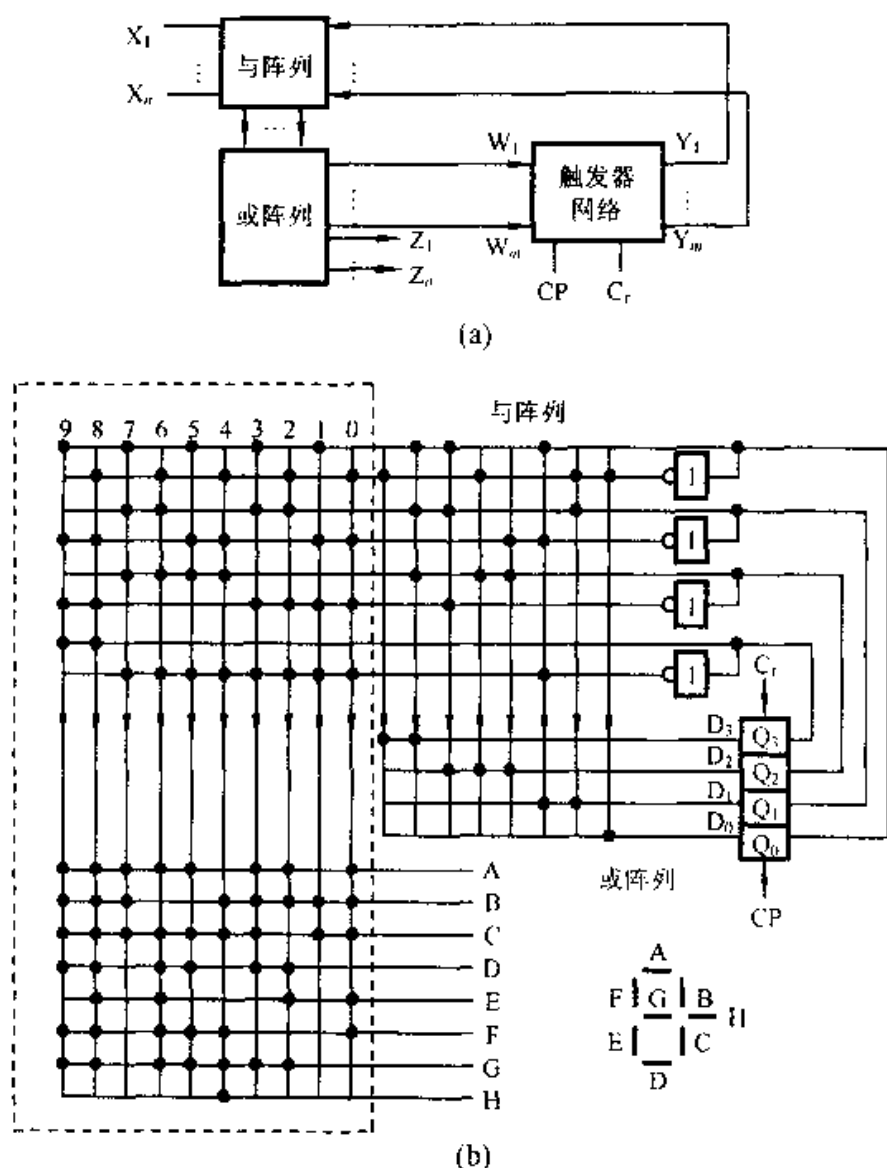


图 8.3.3 用 PLA 设计时序电路

(a) 逻辑框图 (b) 用 PLA 设计 BCD 码十进制计数器

图 8.3.3(b)是用 PLA 设计 BCD 计数器和 8 段译码显示器的阵列图。BCD 码由触发器的状态 $Q_0 \sim Q_3$ 表示, 并且 $Q_0 \sim Q_3$ 反馈至与阵列的输入端。4 个 D 触发器的激励方程可以用时序电路设计方法获得, 用 D 触发器构成十进制计数器的激励方程(请读者自己求出激励方程)为

$$\left. \begin{aligned} D_3 &= Q_2 Q_1 Q_0 + Q_3 \bar{Q}_0 \\ D_2 &= \bar{Q}_2 Q_1 Q_0 + Q_2 \bar{Q}_1 + Q_2 \bar{Q}_0 \\ D_1 &= \bar{Q}_3 \bar{Q}_1 Q_0 + Q_1 \bar{Q}_0 \\ D_0 &= \bar{Q}_0 \end{aligned} \right\} \quad (8.3.1)$$

上式很容易用 PLA 实现, 如图 8.3.3(b)右边部分。用组合电路设计方法可得

$$\left. \begin{aligned}
 A &= 0+2+3+5+6+7+8+9 \\
 B &= 0+1+2+3+4+7+8+9 \\
 C &= 0+1+3+4+5+6+7+8+9 \\
 D &= 0+2+3+5+6+8+9 \\
 E &= 0+2+6+8 \\
 F &= 0+4+5+6+8+9 \\
 G &= 2+3+4+5+6+8+9 \\
 H &= 4
 \end{aligned} \right\} \quad (8.3.2)$$

上式中十进制数 0~9 按 BCD 编码, 即 $0 = \overline{Q_3}\overline{Q_2}\overline{Q_1}\overline{Q_0}$, ..., $9 = Q_3\overline{Q_2}\overline{Q_1}Q_0$ 。

用 PLA 阵列很容易实现式 (8.3.2) 所表示的 8 段显示译码器的输出方程, 如图 8.3.3 (b) 左边部分。

本例中包含了已介绍的逻辑函数化简, 组合及时序电路的设计方法和 PLA 等内容, 读者可以通过设计该电路来达到巩固所学知识, 以及培养综合应用这些知识的能力。

8.4 复杂可编程逻辑器件 (CPLD)

CPLD 是在 GAL 基础上发展起来的高密度 PLD 器件, 它们一般采用 CMOS 工艺和 EPROM、E²PROM 和 Flash 等编程技术, 具有较高的密度和较低的功耗。目前, 主要的 PLD 厂商提供的 CPLD 器件, 结构上大致相同, 但又有着各自的特色。下面以 Altera 公司的 MAX 7000S 系列和 Lattice 公司的 ispLSI 1000 系列器件为例介绍 CPLD 的内部结构。

8.4.1 MAX 7000S 系列 CPLD 器件

Altera 公司的 MAX 7000S 系列是高密度、高性能的 PLD, 采用先进的 CMOS 工艺, 由 E²PROM 保存编程数据, 支持在系统可编程 ISP (In System Programmable) 技术。其内部可提供 600~5000 个可用门, 管脚到管脚 (Pin to Pin) 只有 5ns 延时, 计数器的速度可达到 175.4MHz。MAX7000S 系列器件还提供了可编程的速度、功耗优化功能, 使设计中速度要求高的部分运行在高速、高功耗的状态, 而其他部分则处于低速、低功耗的状态。利用该优化特性配置一个或多个宏单元, 在仅增加很小的延时的情况下可使器件功耗降到 50%, 甚至更低。该系列器件还能通过设置降低输出缓冲的摆率 (Slew Rate), 达到降低信号瞬时噪声之目的可通过 3.3V 或 5V 电源供电。

MAX 7000S 系列器件的内部结构如图 8.4.1 所示, 主要组成部分包括:

(a) 逻辑阵列块 LAB (Logic Array Block), 实现用户设计的逻辑功能, 内部包含 16 个逻辑宏单元 (Macro Cell)。

(b) 扩展乘积项 (Expander Product Terms), 有共享和并联两种方式, 用于实现比较复杂的组合逻辑函数。

(c) 可编程连线阵列 PIA (Programmable Interconnect Array), 为 LAB 之间的信号提供连接所需的通道。

(d) I/O 控制块, 可配置为输入、输出和双向 3 种工作模式。

(e) 特定用途的输入管脚, 包括两个全局时钟管脚 $GCLK_1$ 和 $GCLK_2$, 全局输出使能管脚 OE_1 和 OE_2 , 以及全局清零管脚 $GCLR_n$, 其中 $GCLK_2$ 和 OE_1 共用一个输入管脚。全局时钟信号可以通过 2 选 1 选择器后直接或者经过反相后送到 LAB 的宏单元, 作为触发器的时钟输入, 也可以送到 PIA, 由 PIA 将时钟送到各个 LAB, 作为 LAB 的乘积项的输入信号; OE_1 和 OE_2 送到 PIA, 驱动 6 个输出使能信号, 用以控制 I/O 控制块; CLR_n 既可直接作为触发器的清零信号, 也可送到 PIA 后作为乘积项的输入。

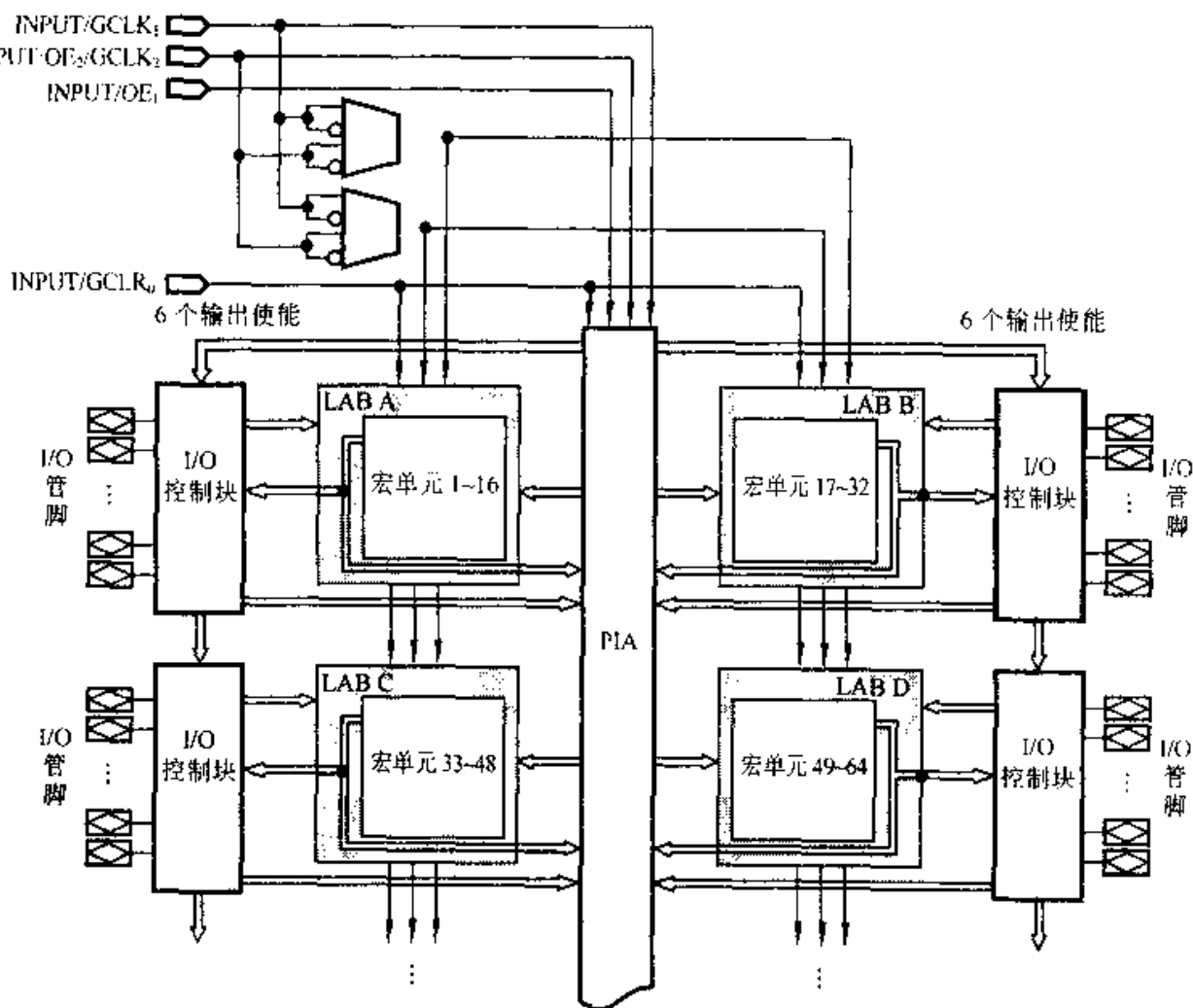


图 8.4.1 MAX7000S 系列器件内部结构

下面以 EPM 7064S 器件为例介绍其内部结构。

1. 逻辑阵列块 LAB

MAX 7000S 系列器件内部的基本组成是 LAB, EPM 7064S 中有 4 个 LAB; LAB 之间通过 PIA 相互传递信号; 每个 LAB 内部包含 16 个宏单元, 因此, LAB 可较为独立地实现复杂逻辑功能; 在 LAB 内部可以完成复杂的电路连接关系, 从而使 LAB 之间的联系信号尽量减少, 降低对 PIA 资源的要求。

宏单元是组成 LAB 的基本单元, 通过配置, 能实现组合逻辑或时序逻辑功能, 其逻辑图如图 8.4.2 所示。其内部包含 3 个功能模块, 分别是逻辑阵列 (Logic Array), 乘积项选择矩阵 (Product-Term Select Matrix) 和可编程的触发器。

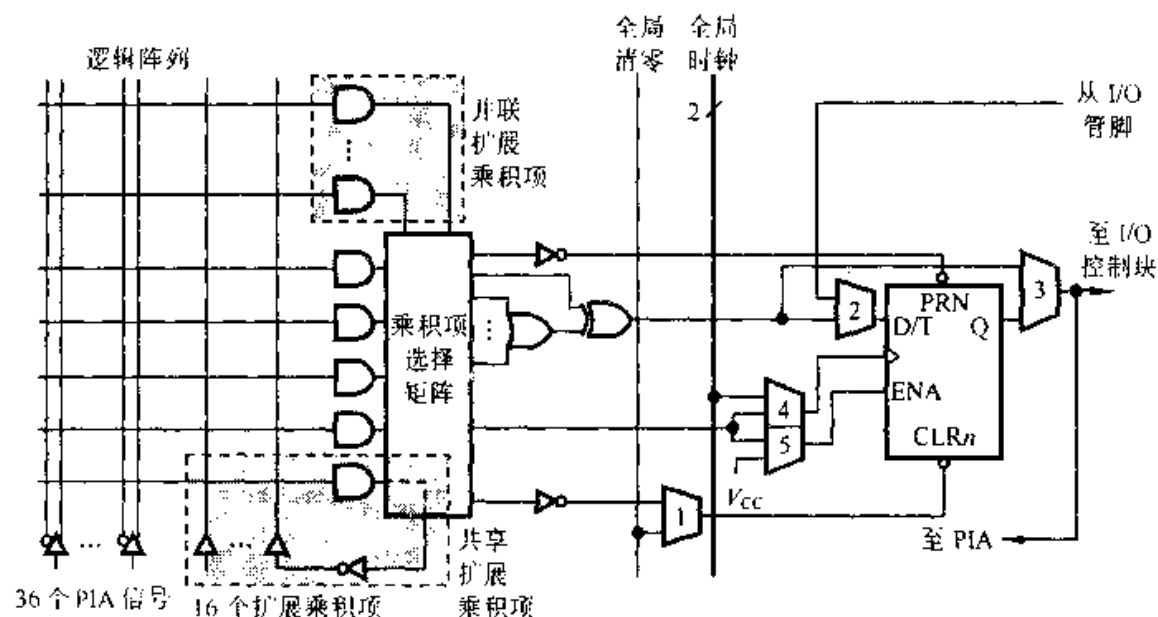


图 8.4.2 宏单元内部结构

每个宏单元的逻辑阵列提供了 5 个乘积项, 用以实现信号的组合逻辑功能, 乘积项的输入信号来源有 3 个, 一个是输入管脚的信号, 二是从扩展乘积项反馈的信号, 三是从输出管脚反馈的信号。

乘积项选择矩阵或者把逻辑阵列提供的乘积项分配到或门和异或门的输入端, 以实现组合逻辑函数; 或者把这些乘积项作为宏单元的辅助输入, 提供可编程触发器的清零、置位、时钟和时钟使能等控制信号。异或门用于控制输出信号的极性, 其一个输入为或门的输出信号, 另一个是乘积项选择矩阵输出的控制信号, 当控制信号为 1 时, 异或门将或门输出信号反相, 当控制信号为 0 时, 异或门的输出与或门输出信号一致。

可编程触发器模块主要包含一个可独立编程的触发器, 该触发器可通过编程实现 D 型、T 型、JK 型和 RS 型触发器。Altera 的开发软件将选择最合适的触发器类型来优化器件资源的利用。有 5 个选择器 (如图 8.4.2 所示的选择器编号) 用以选

择触发器的各个输入端的信号, 包括

(a) 触发器异步清零信号选择器 1。该选择器输入信号有两个, 一个是全局清零管脚输入的信号, 另一个是由其他信号通过组合逻辑后由乘积项选择矩阵输出的信号, 其输出作为触发器的异步清零端输入。

(b) 触发器输入信号选择器 2。其输入信号一个是异或门的输出信号, 另一个是由输入管脚直接输入的信号, 其输出接到触发器的数据输入端。

(c) 触发器输出信号选择器 3。用来选择输出信号模式, 该选择器输入信号有乘积项选择矩阵直接输出的信号和触发器的输出信号。当选择乘积项直接输出信号时, 相当于把触发器旁路, 此时该宏单元实现的是组合逻辑功能, 如果选择触发器的输出信号, 则宏单元完成的是时序逻辑功能。

(d) 触发器时钟选择器 4 和使能信号选择器 5。选择器 4 用以选择触发器的时钟信号, 其输入信号包括全局时钟信号和乘积项输出信号, 其输出送到触发器时钟端; 选择器 5 选择触发器的使能信号, 其输入信号包括乘积项选择矩阵输出的信号和高电平, 输出接到触发器的使能端。通过控制这两个选择器, 使触发器工作在 3 个不同的时钟模式:

➤ 当选择器 4 选择全局时钟信号, 选择器 5 选择高电平时, 触发器工作在全局时钟模式, 这种模式可以得到最佳的速度性能:

➤ 当选择器 4 选择全局时钟信号, 选择器 5 选择乘积项输出信号时, 触发器工作在有使能的全局时钟模式, 该模式的触发器时钟仍从全局时钟管脚引入, 但还为触发器提供了使能信号, 该模式不仅仍能达到全局时钟模式的速度性能, 还具有更高的稳定性:

➤ 当选择器 4 选择乘积项输出信号, 触发器 5 选择高电平时, 触发器工作在乘积项时钟模式下。该模式下, 触发器的速度性能和稳定性相对较低。当编程数据下载后, 这些选择器的选择信号将被设定成固定值, 因此, 该宏单元的工作逻辑就可确定。此外, 乘积项选择矩阵中还可以输出触发器的异步置位信号。

2. 扩展乘积项

虽然大多数的组合逻辑函数可以由每个宏单元提供的 5 个输入乘积项来实现, 但对于更加复杂的逻辑功能, 需要附加的乘积项来实现。MAX7000S 器件中允许在同一个 LAB 中的每个宏单元中的乘积项, 以共享和并联扩展乘积项的形成, 作为附加的乘积项供其他 LAB 使用。

有两种形式的扩展乘积项用来补充宏单元的逻辑资源: 共享扩展项和并联扩展项。

(1) 共享扩展项

共享扩展项如图 8.4.3 所示。每个 LAB 有 16 个宏单元, 每个宏单元的乘积项选择矩阵都提供一个未被使用的乘积项, 将它们反相之后反馈到逻辑阵列, 因此一个 LAB 中共有 16 个共享扩展项。这些扩展项可以被本 LAB 中的任一宏单元使用,

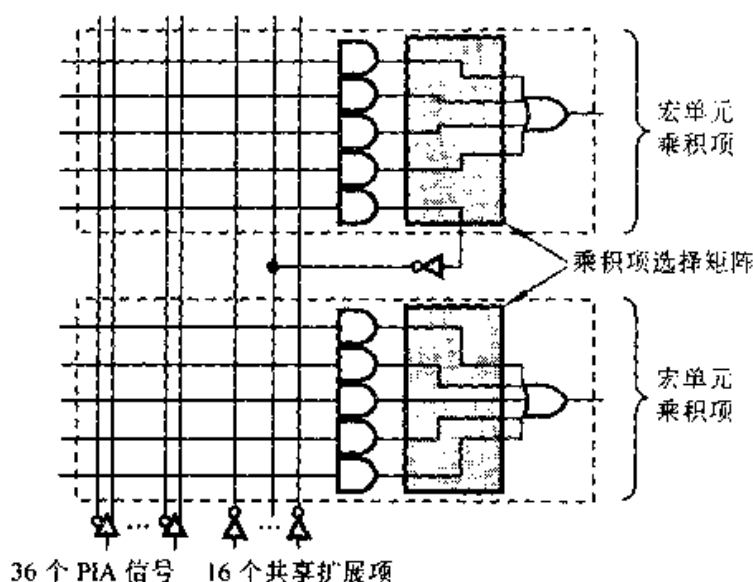


图 8.4.3 共享扩展项

用以实现比较复杂的逻辑函数。但采用共享乘积项后，信号经过一个宏单元的乘积项选择矩阵之后又送到与或阵列，将会增加一定的传输延时。

(2) 并联扩展项

关联扩展项如图 8.4.4 所示。并联扩展项是某个宏单元可以使用其相邻的一些宏单元中没有使用的乘积项，来实现快速、复杂的逻辑功能。使用并联扩展项，可以使多达 20 个乘积项直接馈送到宏单元的或门输入端，其中 5 个乘积项来自于该宏单元本身，15 个来自于同一个 LAB 中的邻近宏单元提供的并联扩展项。

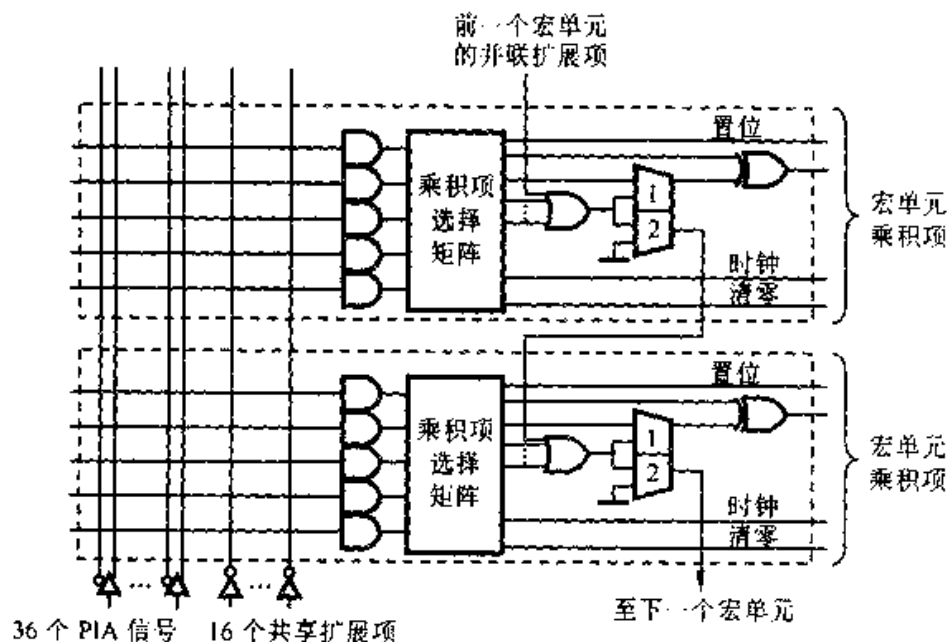


图 8.4.4 并联扩展项

开发软件对 LAB 内部的宏单元进行编号，其中宏单元 1~8 为一组，宏单元 9~

16 为一组形成两条并联扩展项的“借出”和“借入”链，一个宏单元只能从同组中的比其编号小的宏单元借入并联扩展项。例如，当编号为 5 的宏单元要实现 14 个乘积项，则该宏单元本身将提供 5 个乘积项，并且还要使用 2 个并联扩展项，编号 3 的宏单元将提供 4 个乘积项，其输出作为并联扩展项接到编号为 4 的宏单元的或门，与该宏单元提供的 5 个乘积项相或之后，作为并联扩展项送到宏单元 5。由于并联扩展项与本宏单元提供的乘积项相比，其传输路径较长，因此会增加一定的传输延时，而且扩展项越多，延时也就越大。

宏单元中通过两个 2 选 1 的选择器来控制该宏单元是独立实现逻辑功能，还是为相邻宏单元提供并联扩展项。选择器 1 的两个输入信号分别来自乘积项选择矩阵和或门输出，选择器 2 的输入信号为或门输出信号和低电平，两个选择器共用一个控制信号。当选择器 1 选择乘积项选择矩阵的信号时，选择器 2 的选择或门输出的信号，此时选择器 2 的输出将作为其相邻宏单元的并联扩展项输入；若选择器 1 选择或门输出信号，选择器 2 选择为低电平，则不会对相邻宏单元的或门逻辑功能产生影响。

3. 可编程连线阵列

PIA 为器件中的各个 LAB 之间的信号提供连接的通道，MAX7000S 器件中，所有的特定输入管脚、输入/输出管脚和宏单元的输出信号均馈送到 PIA，PIA 可以把这些信号送到器件的任一地方。但只有 LAB 确实需要的信号，PIA 才会将其选通到 LAB。

PIA 选通方法如图 8.4.5 所示，PIA 上的信号分别送到二输入与门的一个输入端，与门的另一个输入端由编程控制，各与门的输出连接到或门的输入，或门的输出为送至 LAB 的信号。

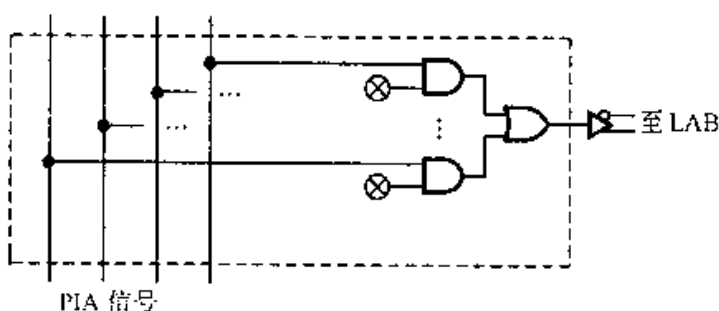


图 8.4.5 PIA 选通方法

注：⊗表示由编程控制

4. 输入/输出控制块

输入/输出控制块允许每个 I/O 管脚独立地配置为输入、输出和双向工作模式。其结构如图 8.4.6 所示。所有 I/O 管脚都有一个三态缓冲器，它由一个 8 选择 1 的选择器来控制其控制信号。该控制信号可以是全局的 6 个输出使能信号，也可以是地

(GND) 或高电平 (V_{CC})。

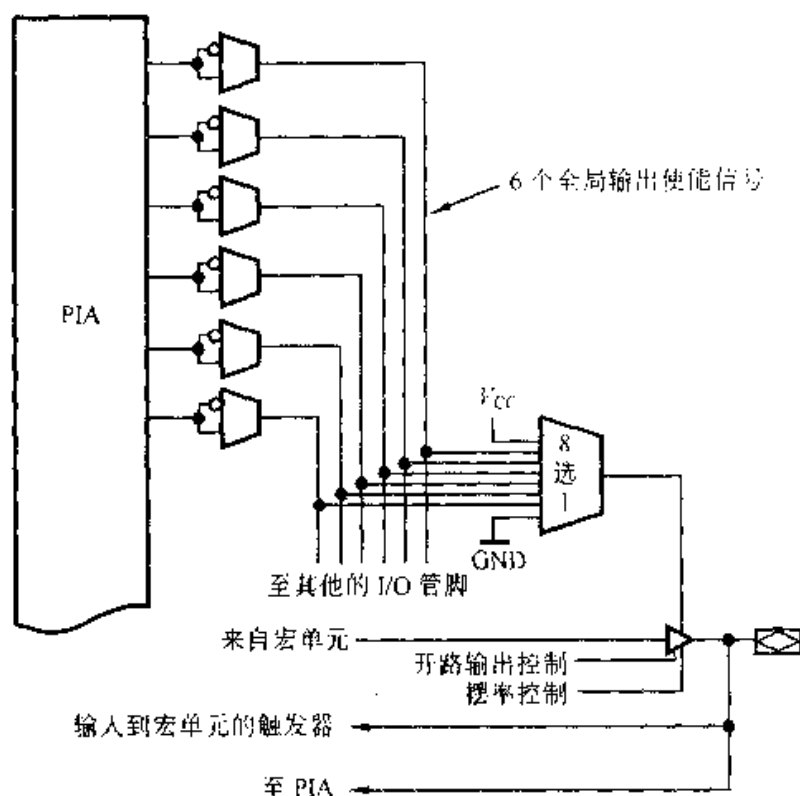


图 8.4.6 输入/输出控制块结构

(1) 输入模式

当三态缓冲器的控制信号接到 GND 时，I/O 管脚的输出为三态，此时该 I/O 管脚配置为输入模式，信号从输入管脚输入后，或者直接送到宏单元作为寄存器的输入信号，或者送到 PIA，由 PIA 送到器件中的 LAB，作为乘积项的输入信号。

(2) 输出模式

当三态缓冲器的使能信号接到 V_{CC} 时，其输出始终有效，I/O 管脚工作在输出模式。输出信号从宏单元输出后送到三态缓冲器的输入，由于缓冲器的使能信号为 V_{CC} ，故可直接送到 I/O 管脚上。通过三态缓冲器还可以控制是否开路输出，以及输出信号的摆率。

开路输出使器件能提供系统级的控制信号，如中断和写使能信号，而且还提供“线或”的功能。利用外部的接到 5V 的上拉电阻，MAX7000S 能够输出与 CMOS 电压(5V)兼容的信号。

输出缓冲器还可调整输出信号的摆率，用以设置输出信号噪声/速度性能。较高的摆率可以提高系统的速度，但同时会引入较大的噪声；较低的摆率可以减少系统的噪声，但会使信号传输时延增加 4~5ns。

(3) 双向模式

当三态缓冲器的使能信号不是接到固定电平，而是由全局使能信号来控制时，

I/O 管脚工作在双向模式。当使能信号为高电平时, 信号由宏单元输出到管脚, 当使能信号为低电平时, 信号由 I/O 管脚送到器件内部。

8.4.2 ispLSI 1032EA 器件

ispLSI 1032EA 器件是高密度可编程逻辑器件, 内部包含 192 个寄存器, 64 个通用 I/O 管脚, 8 个特定输入管脚, 4 个特定时钟输入管脚以及一个全局布线池 GRP(Global Routing Pool)。其结构如图 8.4.7 所示, 包含以下组成部分。

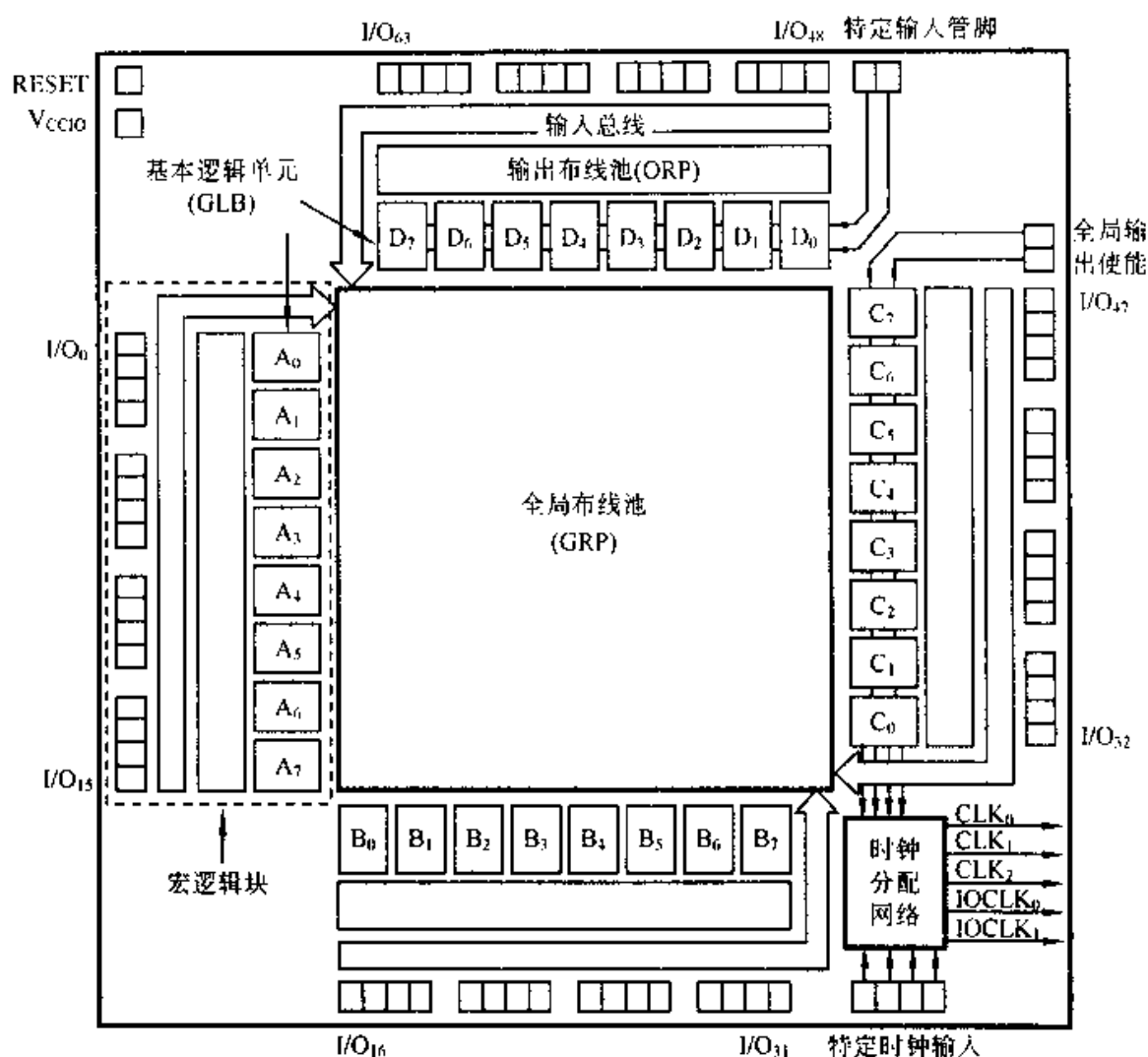


图 8.4.7 ispLSI 1032EA 内部结构

(a) 宏逻辑块 (Megablock)。每个宏逻辑块内部包含 8 个基本的逻辑单元 GLB(Generic Logic Block), 16 个 I/O 单元和输出布线池 ORP(Output Routing Pool)。

(b) 全局布线池 GRP。这是 ispLSI 系列器件中的一种专用内部结构, 它提供高速的内部连线, 将任意的 GLB 输出或 I/O 单元的输入连接到任意的 GLB 的输入端,

采用此特殊结构后, GRP 造成的信号延时就是固定的并且可预测的。

(c) 时钟分配网络 CDN(Clock Distribution Network)。用以选择器件内部使用的时钟。

1. 宏逻辑块

宏逻辑块是 ispLSI 1032EA 器件中的一个大的结构单位, 器件内部包含 4 个宏逻辑块, 每个宏逻辑块中包含 8 个 GLB, 16 个通用 I/O 单元, 两个特定输入管脚和 1 个 ORP, 其结构如图 8.4.8 所示。ORP 把 GLB 的输出信号连接到配置为输出或双向模式的 I/O 管脚。宏逻辑块的输入信号分为两类, 其处理方法也有所不同, 一类是通用 I/O 单元作为输入时, 其输入信号直接送入 GRP, 由 GRP 将其送到器件中的任意的 GLB; 二是每个宏逻辑块有两个特定输入信号, 它们被直接送到本宏逻辑块中的 8 个 GLB, 作为与阵列的输入。

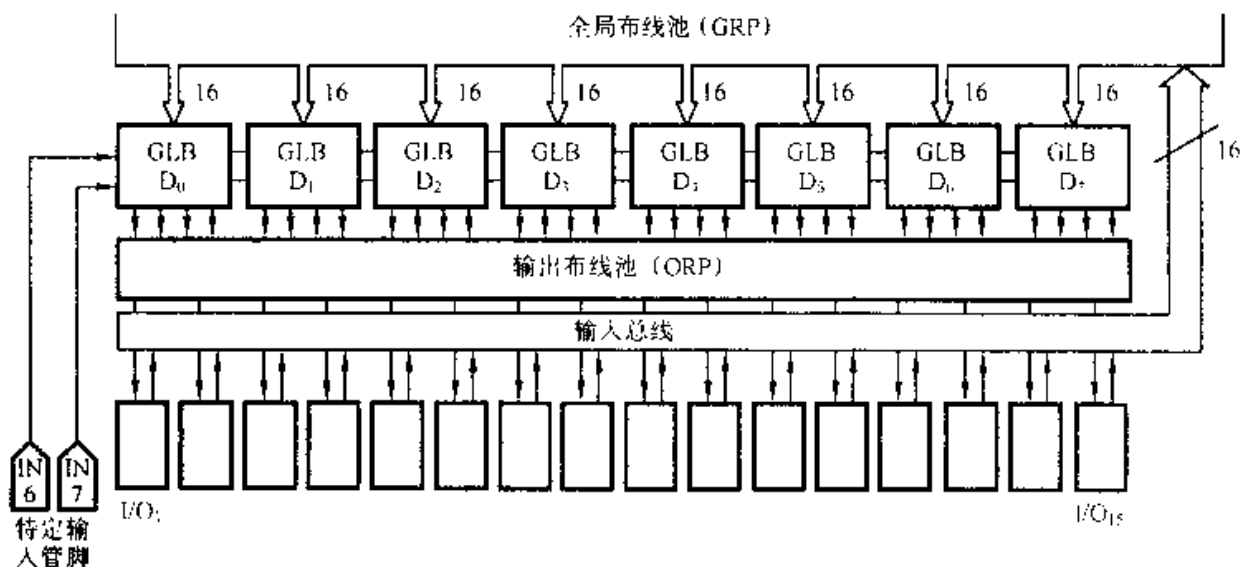


图 8.4.8 宏逻辑块结构

(1) 通用逻辑块

ispLSI 1032EA 的基本逻辑单元是 32 个通用逻辑块, 分别属于 4 个宏逻辑块, 分别标记为 $A_0 \cdots A_7$, $B_0 \cdots B_7$, \cdots , $D_0 \cdots D_7$, 其结构如图 8.4.9 所示。每个 GLB 包含一个 18 输入的可编程的与/或/异或阵列, 乘积项共享阵列, 4 个可配置成组合模式或者寄存器模式的触发器, 以及控制功能块。

(a) 与阵列: 共有 18 根输入信号线和 20 个乘积项输出, 能够实现 GLB 的 18 个输入信号的组合逻辑函数, 18 个输入信号中有 16 个输入来自于 GRP, 两个来自特定的全局输入管脚。输入信号通过互补缓冲器进入与阵列, 为乘积项提供两种极性的信号。

(b) 乘积项共享阵列: 相当于与或阵列中的或阵列, 共有 4 个或门, 其输入分别是 3, 3, 4, 6 个乘积项。或门的输出再经过一级乘积项共享阵列, 如果输出的函数比较复杂, 需要实现的乘积项多于 6 个, 则由该阵列将两个或两个以上的或门

输出合并,再输出到下一级电路。

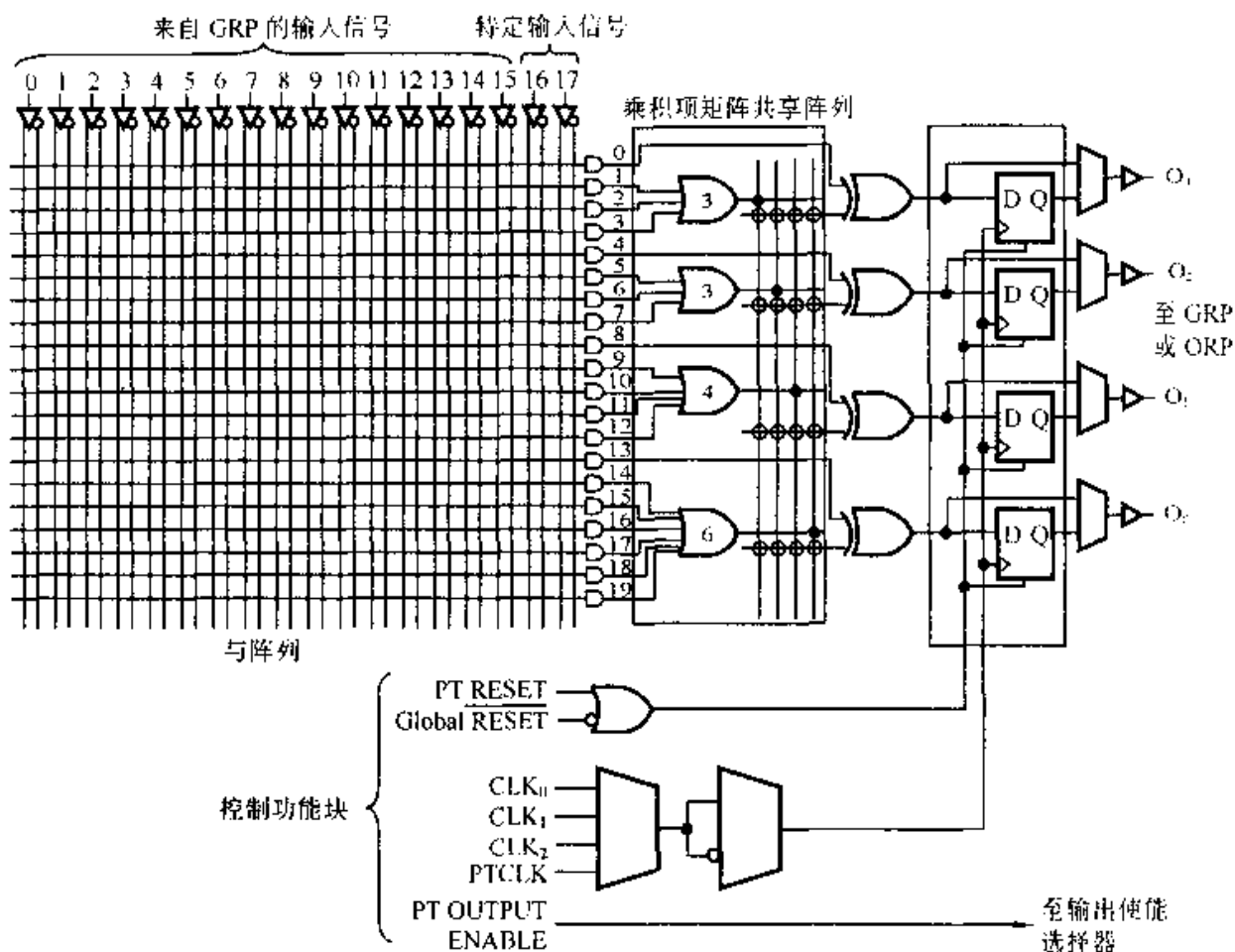


图 8.4.9 GLB 内部结构

(c) 可配置寄存器: 包含 4 个带异或门输入的 D 触发器, 异或门可用来实现异或逻辑功能, 也可用来配置 D 触发器, 使之完成 JK 触发器或者 T 触发器的功能, 这样可大大简化计数器、比较器和算术逻辑单元的设计。异或门的输出信号和寄存器的输出信号分别接到 2 选 1 选择器的两个输入端, 如果选择前者输出, 则寄存器被旁路, 实现的是组合逻辑功能, 如果选择后者, 则实现的是时序逻辑功能。

(d) 控制功能块: GLB 的输出操作由控制功能块的输出信号控制。它通过一个 4 选 1 的选择器来选择寄存器的时钟, 选择器的输入时钟来自于 CDN 的 CLK₀、CLK₁、CLK₂ 和本 GLB 的乘积项, 然后通过一个 2 选 1 选择器来选择时钟信号的极性; 寄存器的复位信号由全局复位管脚和 GLB 乘积项产生的复位信号相或后产生; I/O 单元输出使能信号由 GLB 的一个乘积项提供。

GLB 的输出或者接到 GRP, 由 GRP 将其送到其他的 GLB, 或者输出到 ORP, 由其送到输出管脚。

(2) I/O 单元

ispLSI 1032EA 内部的 64 个 I/O 单元, 分别属于 4 个宏逻辑块, 每个 I/O 单元都

直接连接到 I/O 管脚，每个 I/O 单元都可被独立编程为组合输入、寄存器输入、缓冲输入、输出或者带三态控制的双向 I/O 管脚。此外，所有输出都可进行极性选择。其内部结构如图 8.4.10 所示。

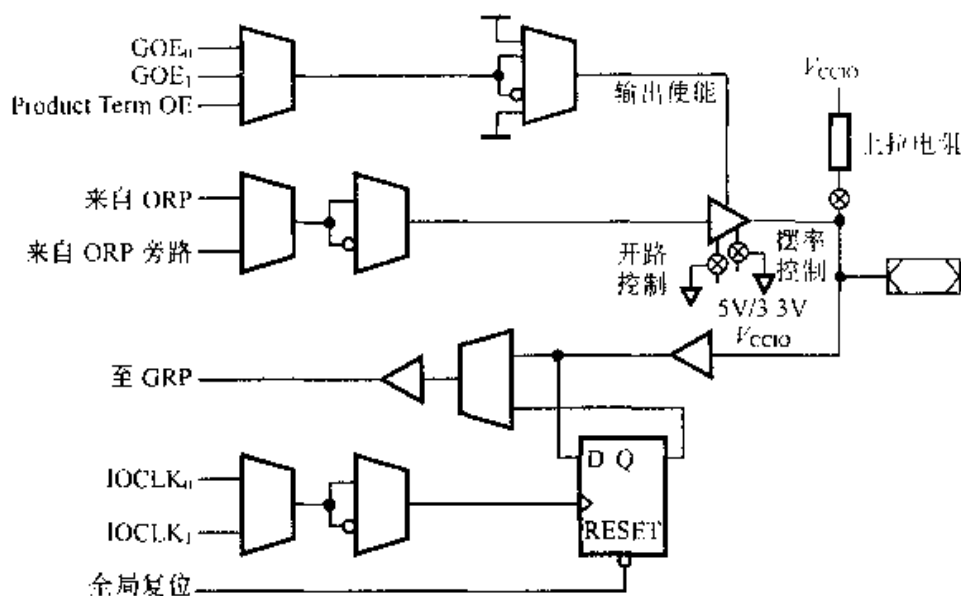


图 8.4.10 I/O 单元内部结构

I/O 单元的工作状态由输出使能信号控制，与 MAX 7000S 器件相似，输出使能信号在全局输出使能信号 GOE_0 、 GOE_1 和 $PTOE$ （乘积项提供的使能信号）之中选择。工作在输入状态时，输入信号通过选择器在其两种输入模式之间选择，一是直接输入到 GRP，二是经过触发器后再输入到 GRP。当采用触发器输入时，输入信号送到 D 触发器的输入端，触发器的时钟由 $IOCLK_0$ 或 $IOCLK_1$ 来控制，触发器的复位受到全局复位信号的控制。在输出状态时，与 MAX 7000S 相同，亦可为每个管脚配置为开路输出，以及独立设置输出信号的摆率。此外，管脚上还可配置的内部上拉电阻，接到 V_{CCIO} ，如果该 I/O 单元没有被使用，则上拉电阻自动使该 I/O 单元保持在 V_{CCIO} 的电平，以提高系统的抗噪声性能。

2. 时钟分配网络

CDN 的结构如图 8.4.11 所示，CDN 的输入包括 4 个全局时钟管脚 ($Y_0 \sim Y_3$) 和特定的 GLB (ispLSI 1032EA 器件中为 C_0) 的 4 个输出信号 ($O_0 \sim O_3$)，它们经过 CDN 后输出 5 个全局时钟信号 ($CLK_0 \sim CLK_2$, $IOCLK_0$, $IOCLK_1$)，其中 $CLK_0 \sim CLK_2$ 可被所有的 GLB 使用， $IOCLK_0$ 和 $IOCLK_1$ 用来作为 I/O 单元触

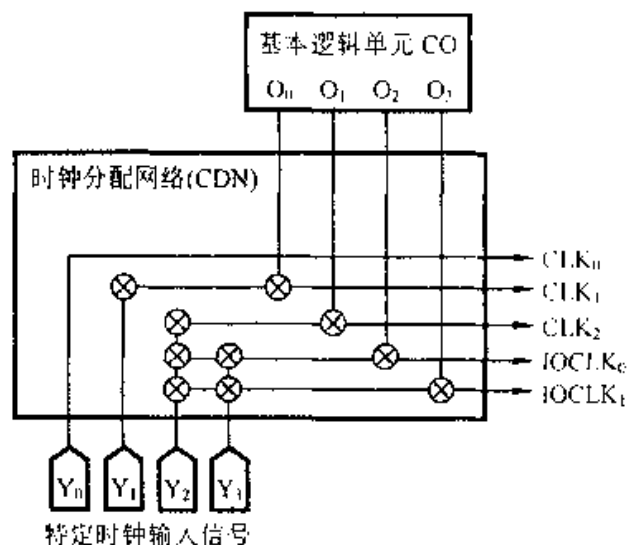


图 8.4.11 时钟分配网络

发器的时钟,图中有⊙的点表示可编程,由图可以看出, Y_0 管脚输入的时钟直接输出到 CLK_0 ,而 CLK_1 的输出则是对 Y_1 的输入和 $GLBC_0$ 的输出 O_0 信号的编程选择的结果; CLK_2 、 $IOCLK_0$ 和 $IOCLK_1$ 的选择灵活性就更大,因此,也就能为系统提供更加丰富的时钟信号。

8.5 现场可编程门阵列(FPGA)

GAL器件虽然具有功能密度较大、设计灵活、使用方便及保密性强等优点,这些是标准器件无法比拟的。但是,它仍存在两个问题:一是集成度(功能密度)不高,一般只含有8个输出宏单元,最多也只有10个;二是I/O引脚数太少,最大输入数为16或20个,规模最大的G39V18也只有22个。因而它只能替代(5~20)个中、小规模的标准器件,难以构成功能复杂的逻辑部件,当然更谈不上构成一个小型数字系统了。随着数字技术应用领域的不断扩展,人们迫切希望获得功能密度大、功耗低、成本低、体积小而可靠性高的集成器件。1985年研制成功的FPGA在较大范围内满足了人们的要求。它仍然是一种由用户编程来实现逻辑功能的器件,不过它的功能密度已相当高,目前,FPGA产品可替代100~200片标准器件,或者20~40片GAL器件,它们的I/O引脚数多达100余条甚至200余条。所以,一片FPGA芯片可以替代多个逻辑功能十分复杂的逻辑部件,或者一个小型数字系统。自EPGA问世以来,已在全球掀起一股研究、开发与应用的热潮,已在许多领域中获得广泛的应用。

8.5.1 FPGA 的电路结构

1. FPGA 结构框图

在前面介绍的几种PLD器件(GAL、FPLA、CPLD)中,其基本电路都是采用与、或阵列加上输出逻辑宏单元的结构模式。而FPGA的电路结构与它们完全不同,它是采用逻辑单元阵列结构,简称LCA型电路结构。

LCA电路主要由3部分组成,即逻辑单元阵列(Configurable Logic Block简称可配置逻辑块CLB)、I/O单元、互连资源。芯片中央是CLB矩阵,四周配置与芯片引脚相连的I/O单元,CLB矩阵的行或列间构成互连资源(称为通道结构),如图8.5.1所示。它的核心是可以通过编程配置的CLB,故称逻辑单元阵列型电路。

这类电路是通过对CLB编程实现逻辑功能;通过对I/O单元编程确定输入或输出结构;通过对互连资源编程实现CLB之间、CLB与I/O单元之间、I/O单元之间的互连关系,从而实现了用户所需要的逻辑功能。

目前, 市场上提供这类电路的公司及其产品有

(1) Xilinx 公司

XC 2000 系列、XC 3000 系列和 XC 4000 系列。

(2) Altera 公司

FLEX 8000 系列、FLEX 10K 系列、FLEX 20K 系列。

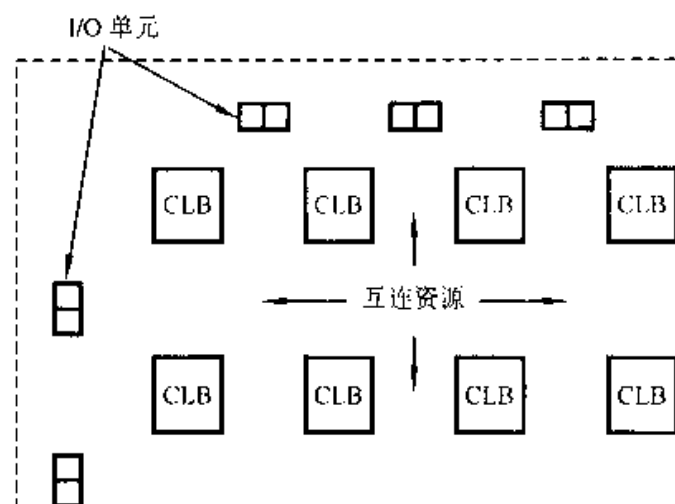


图 8.5.1 LCA 结构示意图

下面以 XC 3020 芯片为例来介绍 FPGA 的电路结构。

图 8.5.2 是 XC 3020 芯片的结构框图。其主要组成部分如下。

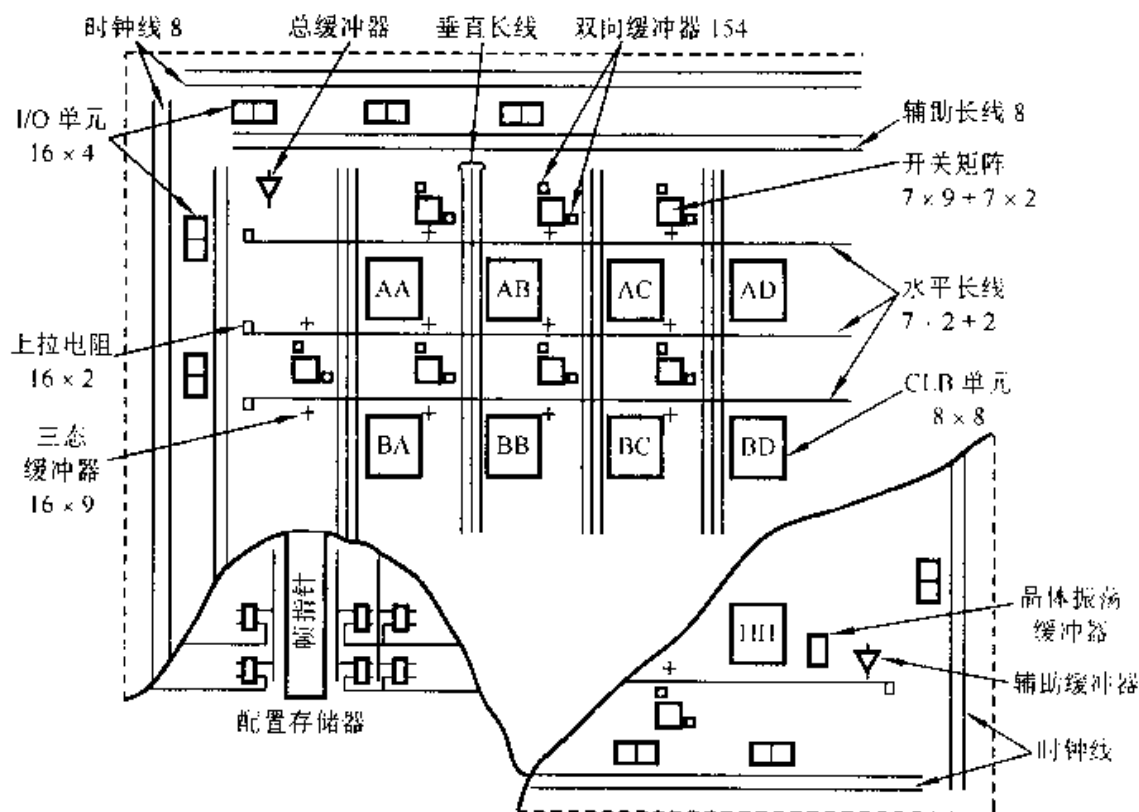


图 8.5.2 LCA 结构图

(a) CLB 单元。它完成用户要求的逻辑功能。64 个 CLB 单元排列成 8×8 方阵配置于中央位置。

(b) I/O 单元。它提供内部逻辑与引脚的接口，配置在芯片四周，每一边 16 个，共有 64 个。

(c) 互连资源。它构成内部逻辑的互连网络。在 CLB 单元行（列）间距或 CLB 单元与 I/O 单元行（列）间距之间的芯片称为行（列）通道，或称水平（垂直）通道。在行或列通道中配置了丰富的互连资源，可以方便地完成 CLB 单元之间或 CLB 单元与 I/O 单元之间的连接，即构成互连网络。其中包含有开关矩阵、长线、各种缓冲器等。

开关矩阵是构成互连网络的主要部件，共有 9 行，配置在行通道中。首末两行只有 7 个开关矩阵，其余各行有 9 个，共计 77 个开关矩阵。在每一行通道中配置有两条水平长线，在开关矩阵的上、下两侧。但是，首末两行通道中只有一条，开关矩阵与 I/O 相邻的一侧没有水平长线，故共有 16 条水平长线。每条水平长线的首尾两端设置有可编程上拉电阻；每一条水平长线上配置有 9 个可编程三态缓冲器，共有 154 个。在每个开关矩阵的左上方和右下方各有一个双向隔离缓冲器 BIDI（功能另有说明）。在每个列通道中有三条垂直长线。此外，在四周最外层通道设置有两条辅助长线，共有八条。

(d) 配置存储器。它用来存储用户的配置程序（配置码），以实现对用户逻辑功能、互连网络以及芯片正常工作的控制。它按一定规律散布于芯片内。

(e) 时钟线。在四周均配置有两条时钟线，可被总缓冲器或辅助缓冲器驱动，也可以被互连网络驱动。

(f) 总（系统）缓冲器。设置在芯片左上角，其主要功能是提供时钟驱动。

(g) 辅助缓冲器。设置在芯片右下角。其主要功能也是提供时钟驱动。

(h) 晶体振荡缓冲器。其主要功能是构成晶体振荡器。

2. CLB 逻辑单元

可配置的逻辑单元（CLB）构成 8×8 方阵，配置于芯片中央。用户对 CLB 编程便可完成用户逻辑功能。编程数据包含在配置程序之中。

CLB 单元逻辑图如图 8.5.3 所示。它包含有组合网络，两个 D 触发器和一些完成清除、时钟极性选择和输出选择等辅助电路。芯片中所有触发器的系统清除端都相连，并受引脚 $\overline{\text{RESET}}$ 控制。其他输入端由互连网络驱动。输出端 X 和 Y 驱动互连网络。

(1) 组合网络

组合网络有 5 个外输入变量和两个内输入变量 Q_X 和 Q_Y ；有两个输出端 F 和 G（注意：不一定是两个独立的函数）。组合网络通过查找表实现逻辑函数。从 7 个输入变量中选取不同组合的输入构成查找表的地址，输出 F 和 G 则依据工作模式，

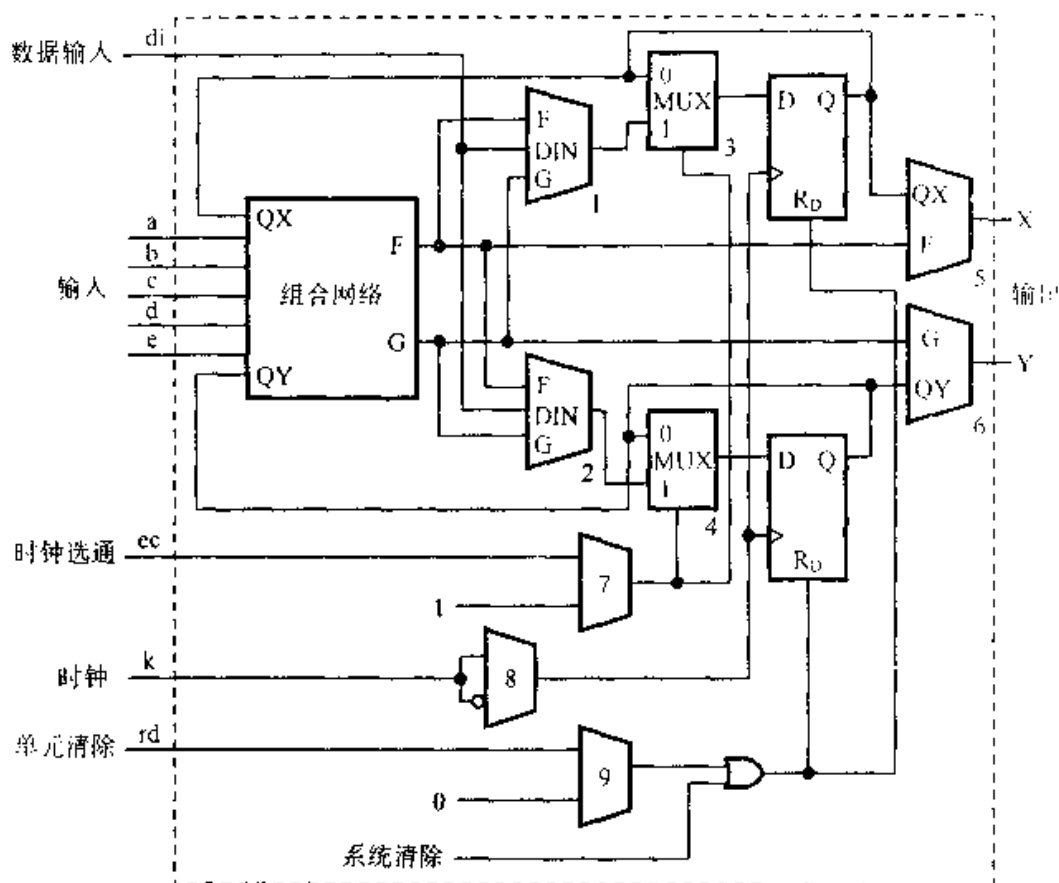


图 8.5.3 CLB 单元逻辑图

可以是两个独立的函数，也可以是同一个函数。组合网络有 3 种工作模式。

(a) FG 模式。

其电路如图 8.5.4(a)所示。从 7 个输入变量中选取 4 个作输入，则可产生两个独立的函数 F 和 G。其输入变量的选用规则为：第一个变量选用 A；第二个变量从 B、Q_X、Q_Y中选取；第三个变量从 C、Q_X、Q_Y中选取；第四个变量从 D 和 E 中选取。

(b) F 模式。

其电路示于图 8.5.4(b)。它产生一个 5 变量的逻辑函数。其输入变量的选用规则为：A、D 和 E 一定选用；另从 B、Q_X、Q_Y和 C、Q_X、Q_Y中各选取一个变量。

(c) FGM 模式。

其电路示于图 8.5.4(c)。这种工作模式是先利用 FG 模式产生两个独立的函数 F 和 G，然后再通过输入 E 来选择其中一个输出。显然输出是一个 6 或 7 变量的函数。

在后面两种模式中，F 和 G 是同一个函数，但是，这种对称配置为优化设计 CLB 单元的输出与其他 CLB 单元或 I/O 单元的连接提供了的条件，有利于获得最佳连接效果。

组合网络的传输延时与所生成的逻辑函数有关，因而输入变量的变化（逻辑电平转换）可能产生逻辑冒险，这一点在进行电路调试和实验模拟时必须引起重视。

(2) 工作原理

组合网络输出 F 与 G 接到三路开关 1 和 2, 它们的另一路信号来自数据输入端 di; F 和 G 还分别为两路开关 5 和 6 提供输入信号。编程三路开关, 选择其中一路信号输至两路开关 3 和 4 的 1 通道, 其 0 通道分别由两个触发器的 Q 端提供信号。若两路开关 5 (6) 编程选择 F (G) 输出, 则构成组合输出; 若选择 Q_X (Q_Y) 输出则构成时序 (寄存) 输出, 此时有两种工作方式:

(a) 时钟使能方式。

两路开关 7 编程选择 1 电平 (ENABLE) 则两个触发器的输入信号来自输入数据 (di) 或者组合网络输出 F 或 G, 此时 CLB 输出 X 和 Y 随时钟节拍实现时序输出, 即为时钟使能工作方式。

(b) 时钟禁止方式。

两路开关 7 选择 ec 通道 (ENABLE CLOCK) 由互连网络驱动 ec 为 1 电平, 则工作在时钟使能方式; 若 ec 为 0, 则触发器实现 Q 状态的循环, 即 Q 状态不变, 因此, 称为时钟禁止方式。

两个触发器设置了系统异步清除和本单元异步清除两种功能。当引脚 $\overline{\text{RESET}}$ 为 0 时, 芯片内部系统异步清除总线 (GLOBAL RESET) 为 1, 故本单

元触发器与其他所有触发器均被清除, 显然 $\overline{\text{RESET}}$ 信号的优先权最高; 当 $\overline{\text{RESET}}$ 为 1 时, GLOBAL RESET 为 0。此时又有两处情况: 若两路开关 9 选择 0 电平 (INHIBIT), 则或门输出为 0, 即为禁止异步清除状态; 两路开关 9 选 rd, 由互连网络驱动 rd 为 1 电平, 则实现本单元的异步清除。

时钟源来自互连网络, 经过编程两路开关 8 可以获得同相或反相时钟信号, 用户可以编程互连网络选择公用时钟, 也可以为该单元选用单独使用的时钟源。

3. I/O 单元

用户编程的 I/O 单元示于图 8.5.5。它提供了内部逻辑与芯片引脚的接口功能。

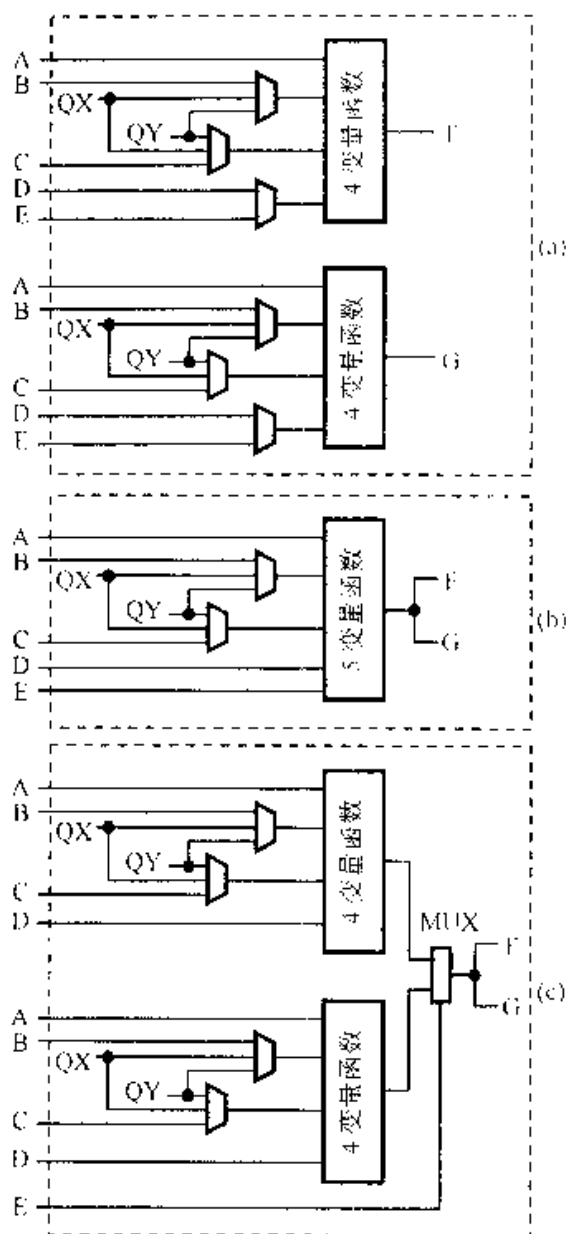


图 8.5.4 组合网络的工作模式

(a) FG 模式 (b) F 模式 (c) FGM 模式

主要包含有输入通道（又称为输入缓冲区）和输出通道（又称为输出缓冲区）。输入通道由输入缓冲器、输入寄存器（或锁存器）和两只钳位二极管构成；输出通道由输出寄存器、输出缓冲器以及输出极性选择、输出三态控制、输出速率转换和一个上拉电阻等部分组成。

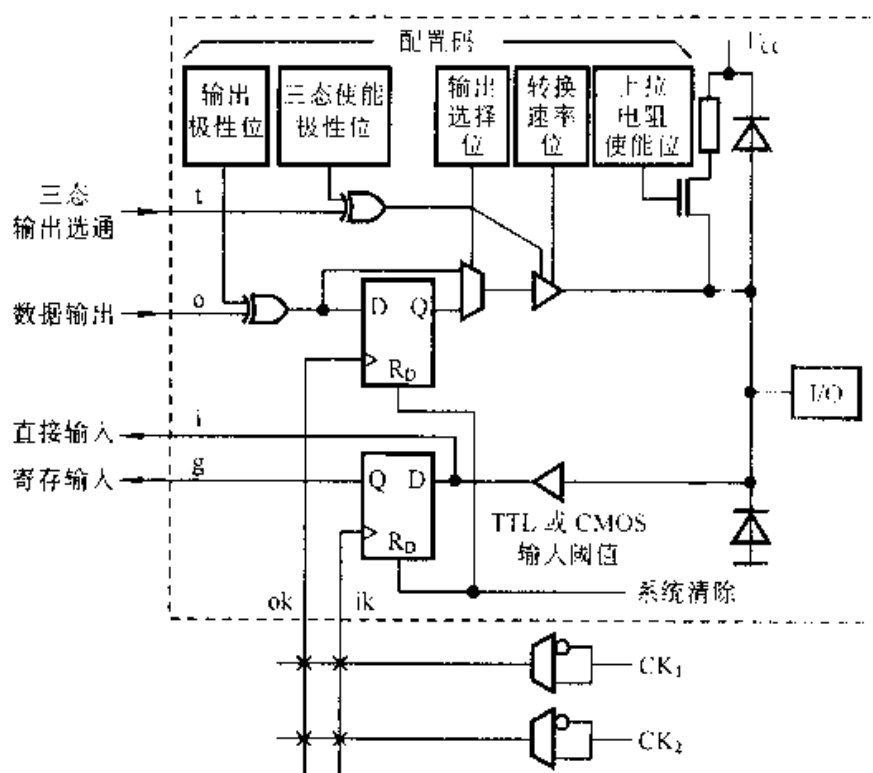


图 8.5.5 I/O 单元

该单元输出设置有三态输出使能端 t，数据输出（OUT）端 o，两个时钟输入端 ok, ik；输入设置有直接输入端 i (DIRECT IN)，寄存输入端 q (REGISTERED)。此外，内部还有系统异步清除端 GLOBAL RESET，它受引脚 RESET 控制。其他输入或输出端均与互连网络连接。

(1) 输入通道

输入信号来自引脚 I/O，若 I/O 单元被编程为输入模式，则信号输至互连网络；若 I/O 单元被编程为 I/O 模式，则信号来自本单元的输出（它同时进入互连网络）。两只钳位二极管具有静电保护和防止输入电流产生触发噪声。输入缓冲器由用户编程，可将引脚 I/O 输入信号转为 TTL 电平或 CMOS 电平，作为直接输入信号；或者输到触发器输入端 D，其输出信号就是互连网络的寄存输入。顺便指出，触发器可由用户编程为寄存器或锁存器。

(2) 输出通道

输出三态使能信号和输出数据均来自互连网络。输出电路由一组配置码（存放在配置存储器中）控制。其中，OUT INVERT 控制输出极性，3-STATE INVERT 位

控制三态使能信号反相；OUTPUT SELECT 位控制输出选择；SLEW RATE 位控制转换速率，1 电平选用高速，0 电平选用低速；PASSIVE PULL UP 位控制上拉电阻使能。若输出缓冲器使能，则总会有一个直接输出或是寄存输出信号从引脚 I/O 输出（进入互连网络），并提供 4mA 的驱动电流和 TTL 与 CMOS 兼容的逻辑电平。各部分电路的工作原理与 CLB 中类似，不再赘述。

(3) 配置 I/O 单元的几个技术问题

在配置 I/O 单元时，构造码的组成是至关重要的，因为它决定了 I/O 单元的电路结构和输出信号的极性。其中，有一部分参数是由设计输入在网表文件中自动生成的（CLB 单元、互连资源的编程数据也与此类似）；而另一部分参数是在设计时由用户定义的（即回答计算机的提问）。如 I/O 单元构造码中 SLEW RATE 位、输入通道中输入缓冲器输入阈值控制位等。用户定义这些控制参量就实现了某些功能选择，有时也各有利弊，应依据实际应用环境来确定。

(a) 输出缓冲器转换速度选择。配置码中转换速率控制位 SLEW RATE 由用户定义。选用 1 电平则为高速；选用 0 电平则为低速。在缺省条件下开发系统定义为低速。它主要是控制输出缓冲器输入端至引脚 I/O 之间的传输延时。选择高速可以缩短传输延时，有利于改善临界时间，避免出现逻辑冒险；选择低速可以减少非临界输出的容性负载上的尖峰电流和系统噪声。它的选择主要取决于芯片工作速度。

(b) 上拉电阻的使用。引脚 I/O 不能出现浮置状态，因为对于 CMOS 输入电路易产生寄生振荡，导致功耗增加和系统噪声增强。所以，I/O 单元未使用的 I/O 引脚就一定要接入上拉电阻，即上拉电阻使能（PASSIVE PULL UP 位为 1 电平），以便为 I/O 引脚提供稳定的 1 电平，避免出现浮置状态。

(c) 时钟极性选择。输入触发器可以编程为 D 型寄存器或锁存器，而输出触发器只能编程为 D 型寄存器。它们的时钟极性可编程选择，可编程选择上升沿或是下降沿触发；高电平透明或是低电平透明。

4. 配置存储器

配置存储器用于存储配置程序（配置码）。开发系统将所有配置数据（CLB, I/O, 互连网络及其他控制参数）按帧结构配置（见图 8.5.14）组织在用户程序（即配置程序）之中。装载配置程序有多种方法，这将在下一节进行介绍。

配置存储器采用静态 CMOS 存储单元，如图 8.5.6 所示。它由两个 CMOS 反相器和一个控制读/写的传输管构成。在配置期间，单元进行写操作，数据存入两个互补输出端 Q 和 \bar{Q} ；在进行程序校验时需执行回读（READBACK）操作，数据由数据端

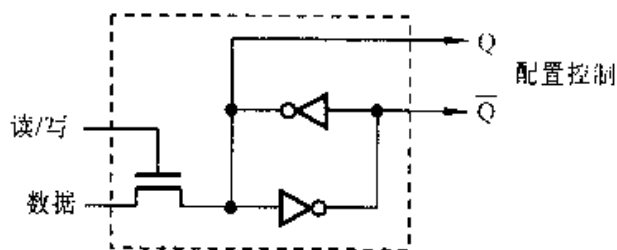


图 8.5.6 配置存储单元

取出。显然,在进行这两种操作时,传输管都应编程为导通状态。在芯片正常操作(工作状态)期间,存储单元的数据(Q , \bar{Q})完成结构控制,无需对它进行读/写。所以,传输管应编程为开路状态,有利于存储单元的工作稳定性,这是比其他类型的存储单元优越之处;同时,在设计时已考虑到高可靠性和低噪声性能,不会受电源漂移和强辐射影响。即使是在最恶劣的工作环境下,存储单元也能保持稳定的性能。

配置存储单元阵列是按一定规律散布于芯片内(参见图 8.5.2)。配置数据按帧结构写入,故对每帧数据要设置地址指针。

5. 其他辅助电路

(1) 总缓冲器

在芯片左上角设置有一个总缓冲器,它具有很强的驱动能力,并且具有无偏移和高扇出性能。由它驱动一个系统网络(GLOBAL NET),可以为所有 CLB 单元(k 端)提供同步时钟;这个系统网络还可以驱动配置于四周的时钟线,为 I/O 单元提供同步时钟源。但从它所处位置考虑,直接从左边第 2 条时钟线连接总缓冲器,更有利于提高速度。

(2) 辅助缓冲器

在芯片右下角设置有一个辅助缓冲器,它也具有较强驱动能力,并具有无偏移和高扇出性能。它驱动一条水平长线,每个垂直通道中的 3 条垂直长线都可以通过编程与之连接,构成提供 CLB 单元的时钟源。由于它处在右下角,所以,最右边垂直通道的第 3 条垂直长线连接辅助缓冲器更为有效。

(3) 晶体振荡器

图 8.5.7 是晶体振荡器电路。虚线内为芯片内包含的电路,主要有两个高速反

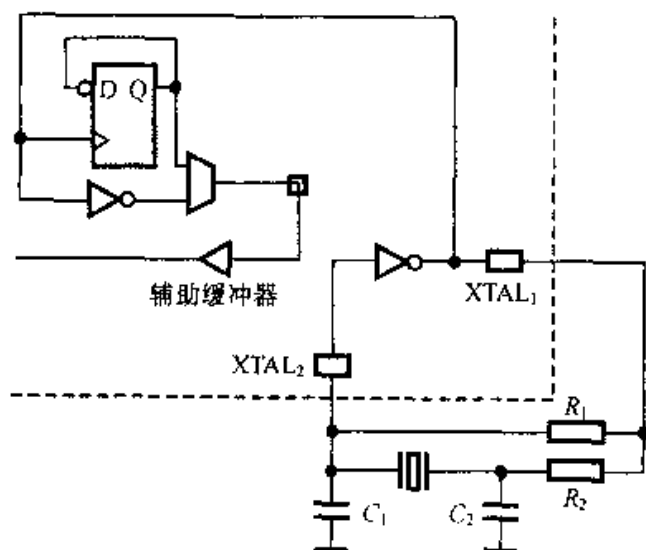


图 8.5.7 晶体振荡器电路图

相放大器, 一个寄存器和一个两路开关, 此外还得用辅助驱动器驱动水平长线提供晶振源; 虚线外为外接的晶体和振荡元件, 它们确定了振荡频率 f_c 。振荡频率范围为 1MHz~20MHz。元件值的选择如表 8.5.1 所示。

表 8.5.1 晶体振荡器参数

f_c/MHz	$R_1/\text{M}\Omega$	$R_2/\text{k}\Omega$	$C_1, C_2/\text{pF}$
1~20	0.5~1	0~1	10~40

构成晶体振荡器时, 利用了两个引脚 (XTAL_1 和 XTAL_2) 外接晶体, 故此时这两个 I/O 引脚不能作它用。当不构成晶体振荡器时, 两个 I/O 引脚可作为用户 I/O。

使用振荡器时, 应考虑编程两路开关可以获得 f_c 和 $f_c/2$ (触发器接为计数触发器)。由于经过二分频后的信号波形具有较好的对称性, 所以, 在工作速度满足要求的条件下应优先选用 $f_c/2$ 作为时钟源。

8.5.2 FPGA 的连接方法

在行和列通道中配置有丰富的互连资源, 为系统提供互连网络以传输各种信号, 如时钟、逻辑、控制等信号。连接是通过带有可编程控制的传输管 (参见图 8.5.6) 的金属线完成的。每个传输管用一位控制码控制, 形成可编程连接点 PIP (Programmable Interconnection Point)。开关矩阵可以方便地完成 CLB 单元与金属线的连接, 图 8.5.8 便是通过开关矩阵和 PIP 连接点实现 CLB 单元之间连接的例子。开发系统既提供依据设计输入进行自动布线的工作方式, 同时也提供人工干预的辅助连接方式, 这有利于进一步优化设计。

为了满足各种网络连接的需要, 有 3 种连接方式可供选择。

1. 通用连接方式

通用连接是借助于开关矩阵完成的, 图 8.5.9 为开关矩阵连接图。在水平与垂直通道的交汇处都有一个开关矩阵, 开关矩阵之间都有 5 条水平线和 5 条垂直线互相连接 (注意: 这与配置在与四周 I/O 单元相邻的开关矩阵略有不同, 它们只有 3 个方向的连接), 对开关矩阵编程便可完成它们的连接。图 8.5.10 是开发系统提供的 20 种连接模式。这种连接可以选择自动布线完成, 也可选用辅助设计完成, 即用编程器选择一对需要的开关矩阵引脚来完成。用开发系统中的显示开关矩阵命令 “Show-Matrix” 可以将它们显示出来。

2. 直接连接方式

在相邻 CLB 单元之间或 CLB 单元与相邻 I/O 单元之间进行直接连接是最方便和有效的, 因它可以减少传输时间, 有利于提高工作速度。但是, 必须注意相邻 CLB

单元之间进行直接连接应遵循的规范：如图 8.5.11 所示，CLB 单元的输出 X 可以与右侧相邻 CLB 单元的输入 b 相连，也可以与左侧相邻 CLB 单元的输入 c 相连；CLB 单元的输出 Y 可以与上方相邻 CLB 单元的 d 相连，也可以与下方相邻 CLB 单元的 a 相连。

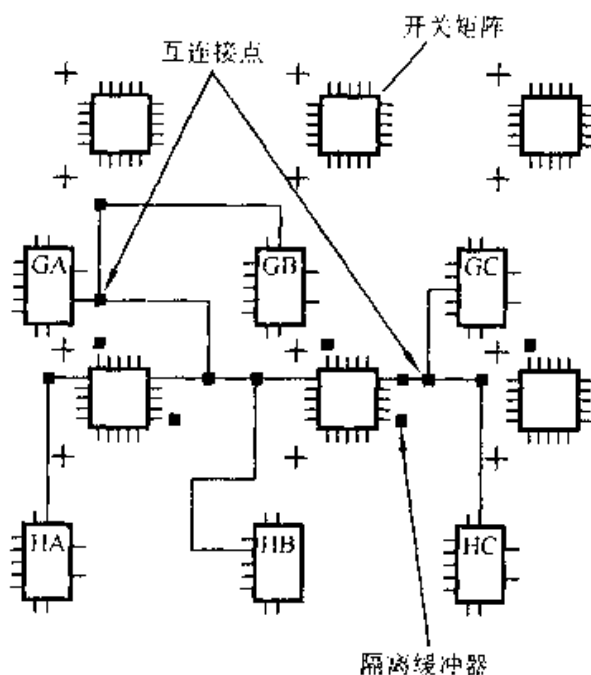


图 8.5.8 LCA 连接示意图

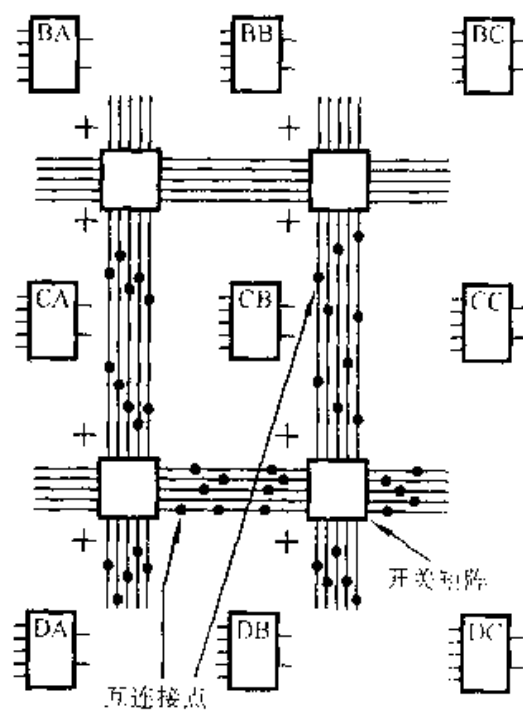


图 8.5.9 通用连接方式

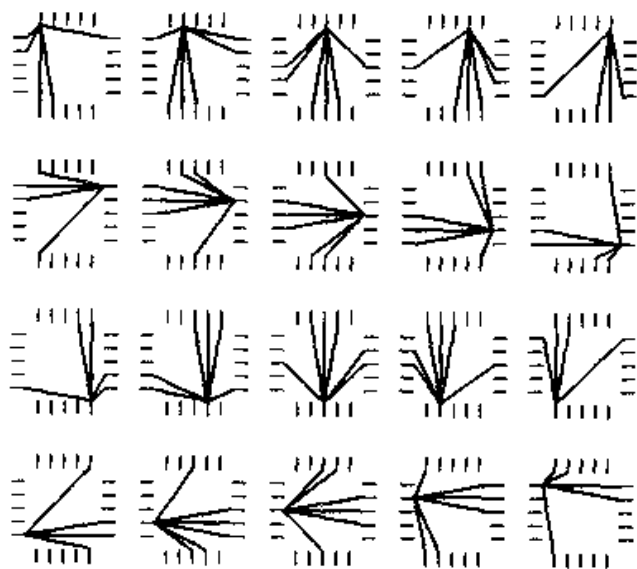


图 8.5.10 开关矩阵连接模式

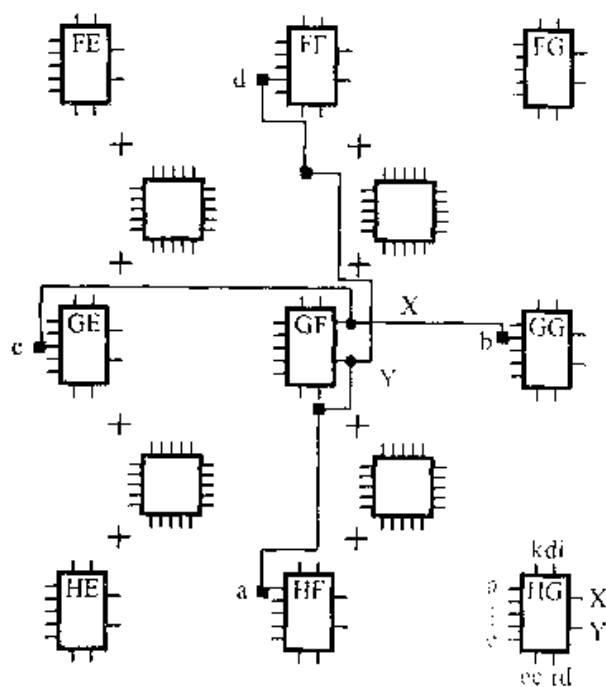


图 8.5.11 直接连接方式

3. 长线连接方式

(1) 长线的传输功能

当信号需要传输到较远距离或者供多个单元使用时,采用长线连接最方便和有效。在行(列)通道设置有水平(垂直)长线两(3)条,利用它们可实现长线连接。长线可为 CLB 单元或 I/O 单元的输出驱动;也可以被总缓冲器或辅助缓冲器驱动,提供公用的同步时钟源。

在使用水平长线时,应注意隔离缓冲器的应用,在每个开关矩阵的左上方和右下方配置有一对双向隔离缓冲器 BIDI,利用它们可以为水平长线的信号传输提供隔离功能,所以,应注意在信号进出水平长线之前先接入双向隔离缓冲器。在布线确定之后,开发系统将自动使用相关的双向缓冲器使能。用开发系统中的显示双向缓冲器命令“Show BIDI”可以将它们显示出来。

(2) 水平长线的逻辑功能

在水平长线上设置有 9 个三态缓冲器,其首、尾两端带有可编程上拉电阻。利用它们可以实现水平长线的逻辑功能。显然,这时的水平长线就具有产生逻辑信号和传输这些信号的双重功能了。图 8.5.12 是水平长线实现线与功能的电路,逻辑式为

$$Z = \overline{D_A} \cdot \overline{D_B} \cdot \overline{D_C} \cdots \overline{D_N}$$

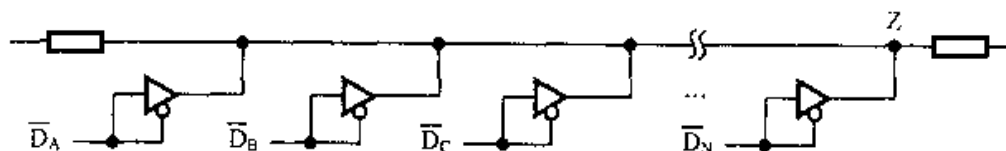


图 8.5.12 长线的线与功能

三态缓冲器为 0 电平选通,只要 $\overline{D_A} \sim \overline{D_N}$ 中有一个使能,则水平长线上为 0 电平;当 $\overline{D_A} \sim \overline{D_N}$ 全为 1 电平时,所有三态缓冲器为禁止状态,其输出均为高阻抗,上拉电阻编程使能,因而水平长线上有稳定的 1 电平,即长线实现线与功能。图 8.5.13 是水平长线实现 N 通道多路器电路,逻辑式为

$$Z = D_A \cdot \overline{A} + D_B \cdot \overline{B} + D_C \cdot \overline{C} + \cdots + D_N \cdot \overline{N}$$

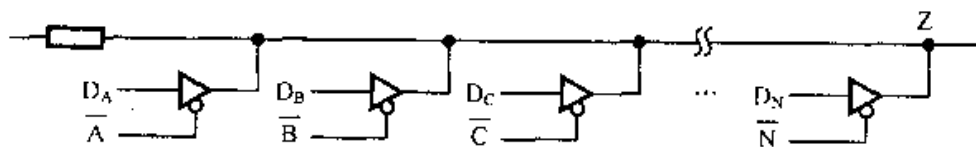


图 8.5.13 长线的多路选择器功能

当 \overline{A} 为 0 电平时(注意: $\overline{B} \sim \overline{N}$ 均为 1 电平), D_A 通道选通,水平长线传输 D_A ;

当 $\bar{A} \sim \bar{N}$ 均为 1 电平时, 所有三态缓冲器输出为高阻抗, 上拉电阻编程使能为水平长线提供稳定的 1 电平。使用图 8.5.13 所示的电路时, 必须注意任何时候最多只允许一个使能信号有效, 关于这一点已在组合逻辑电路中讨论了。

8.5.3 FPGA 的编程和加载方法

FPGA 芯片的编程过程也就是完成芯片内部各部分配置的过程, 不同系列产品的编程方法都类似, 但由于芯片规模不同, 其编程数据长度 (数据帧数及 bit/帧) 有差异。完成程序数据装入芯片配置存储器的过程称为加载过程或配置过程。本书仍以 XC 3000 系列为例介绍 FPGA 的编程和加载方法。

1. 程序数据的数据结构

程序数据 (程序) 由引导数据 (程导头) 和配置数据两部分组成。其中, 配置数据用于实现内部逻辑功能和连接, 它们可以采用几种加载模式装入芯片中的配置存储器。其数据结构如图 8.5.14 所示。引导数据共 40 位, 它们是识别码 11111111 (8 位), 起始码 0010 (4 位), 24 位量程长度码, 启动码 1111 (4 位); 配置数据为帧结构, 每一帧含有起始位 0 (1 位), 71 位数据, 结束位 111 (3 位), 所以, 一帧数据为 75 位, 配置数据的末尾是结束码 1111 (4 位)。

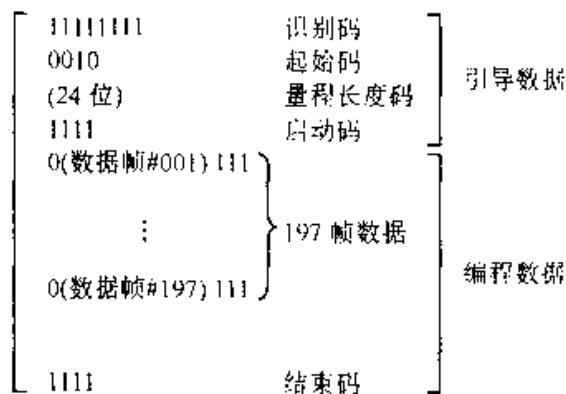


图 8.5.14 XC 3020 数据结构图

Xilinx 公司产品 (包括将要开发的新产品) 的数据结构都相同, 这样可以保持同一公司产品的一致性。但是, 不同型号产品的数据帧数及位/帧则有差异。表 8.1 列出各系列产品的编程数据和 PROM 容量。例如, XC 3020 有 197 帧配置数据和 4 位结束位, 另加 40 位引导数据, 所以, 程序数据总容量为 $(40+197 \times 75+4)$ 位 = 14819 位。一般习惯上将程序数据总容量称为 PROM 容量, 而编程数据的容量则不包含程序头 40 位。XC 3020 的编程数据为 $(197 \times 75+4)$ 位 = 14779 位。

2. 程序数据的加载方法

(1) 加载流程图

上电之后或使用配置 (编程) 指令之后, FPGA 芯片内部的上电-复位电路被接通, 当电源上升到 LCA 开始工作的电压 2.5~3V 时, 芯片进入初始化 (预置) 状态, 所有 I/O 单元的输出缓冲器均为禁止状态, 同时为用户 I/O 引脚接通上拉电阻, 故所有用户 I/O 引脚为固定的高电平。图 8.5.15 是加载流程的状态图。在上电初始化完成之后, 进入清除内部配置存储器状态。

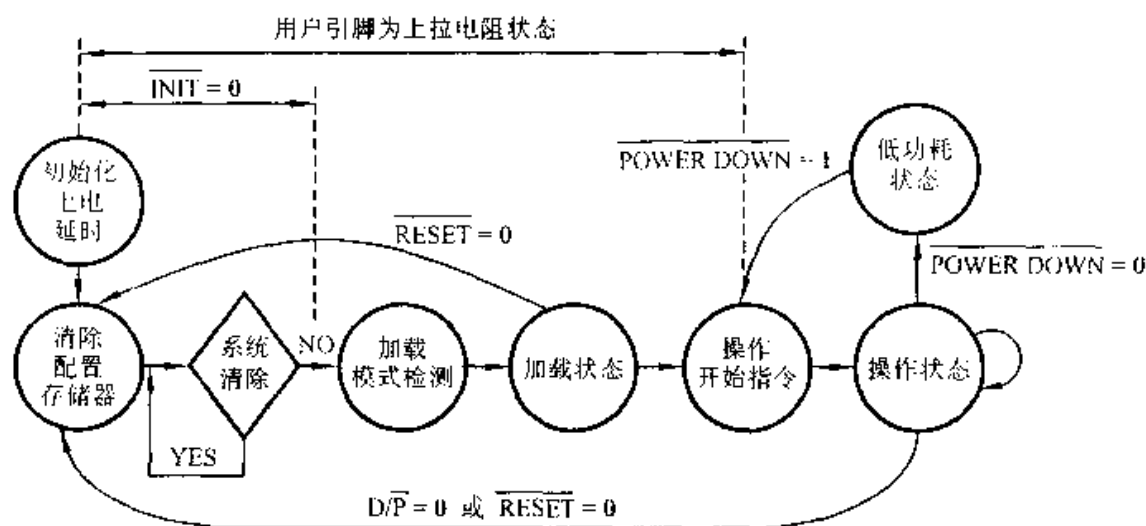


图 8.5.15 加载流程图

在初始化和清除状态期间，预置信号 $\overline{\text{INIT}}$ 为低电平，在完成之后变为高电平。然后检测系统复位信号 $\overline{\text{RESET}}$ 是否有效：如有效（0 电平）则回到检测状态；若无效（1 电平）则转换到加载模式选择。它是通过对 3 个模式引脚 $M_0 \sim M_2$ 进行电平检测来确定加载模式，从而进入程序数据的加载状态的。假如加载过程尚未结束，在引脚 $\overline{\text{RESET}}$ 出现低电平的复位信号，则返回至清除状态，将清除已装载的那些配置存储器，然后再采样 $\overline{\text{RESET}}$ 和状态线，重新进入加载状态；如果加载完毕，则会产生一个工作开始信号（START-UP）便进入操作状态，实现用户逻辑功能。

若芯片暂时不工作，但需保留配置数据，则可令 $\overline{\text{POWER-DOWN}}$ 为低电平，使芯片进入功率下降状态，以节省功耗；如果芯片重新开始工作，则令 $\overline{\text{POWER-DOWN}}$ 为高电平，使芯片经过 START-UP 便进入逻辑操作状态。假如在 $\overline{\text{RESET}}$ 或者 $\overline{\text{D/P}}$ 引脚上出现低电平，则芯片返回到清除状态，同上述过程一样，在清除所有配置存储器之后，重新进入加载状态。

(2) 加载模式

程序数据可以存放在串行存储器或 EPROM 中，也可以存放在微处理器或计算机存储器中。因此，需要提供不同的加载模式来完成加载过程，XC 3020 芯片有 5 种加载模式，如表 8.5.2 所示。程序数据可以按位串行加载，也可以按字节并行加载，这取决于加载模式。加载之前，3 个模式引脚 $M_0 \sim M_2$ 被供给相应逻辑电平，对它们采样便完成加载模式选择。

(a) 主串加载模式（简称主串模式）。

当 $M_0M_1M_2=000$ 时，加载为主串模式。程序数据存放在 Xilinx 公司生产的专用串行配置存储器（SCP）之中，SCP 有两种产品，XC 1736A（36288×1 位）和 XC 1765（65536×1 位）。它们是一种简便的 8 脚芯片，其引脚图如图 8.5.16 所示，其中， $\overline{\text{CE}}$ 为片选端；CLK 为时钟输入端； $\overline{\text{RESET/OE}}$ 为清除/输出允许端； $\overline{\text{CEO}}$ 为输出使能端；

表 8.5.2 5 种加载模式表

$M_0M_1M_2$	模 式	说 明
000	主串	位串联
001	主低	字节, 地址 0000 1
010	—	备用
011	主高	字节, 地址 FFFF 1
100	—	备用
101	外设	字节
110	—	备用
111	从	位串联

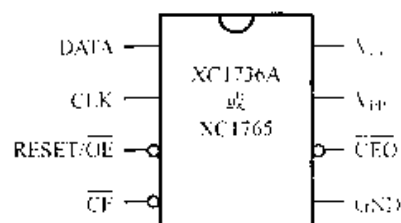


图 8.5.16 SCP 存储器引脚图

V_{pp} 编程电压为 +15V, 读出操作时工作电压为 +5V; DATA 为三态数据输出端。主串加载模式连接图如图 8.5.17 所示, LCA 芯片为 SCP 提供配置时钟 CCLK, SCP 为 LCA 提供程序数据。系统加载之前先清除, 即在 RESET 端加负脉冲, 使 LCA 芯片内部清除, 反相后的正脉冲将 SCP 的地址计数器和位计数器清 0, 然后编程数据逐位输进 LCA。当存储数据读完时, \overline{CEO} 输出低电平, 而 DATA 端呈现高阻抗,

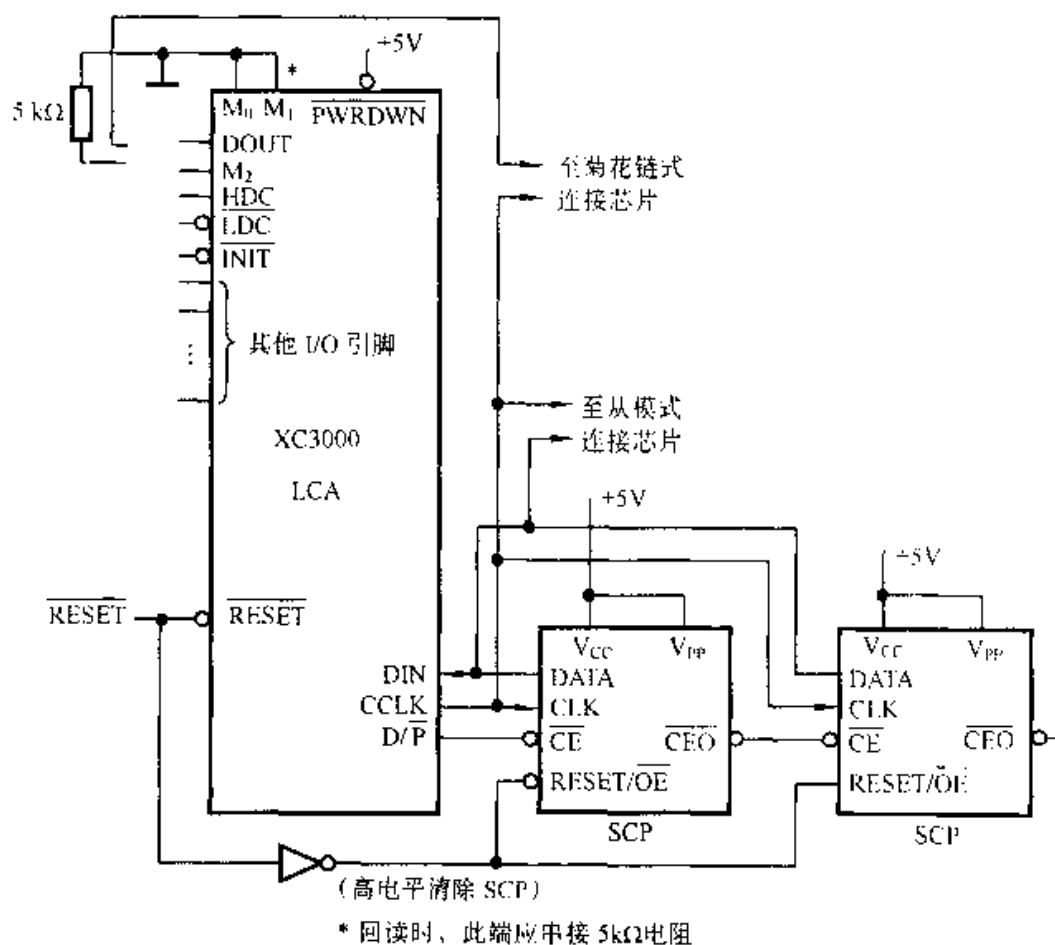


图 8.5.17 主串加载模式连接图

这种结构可以方便地构成多路 SCP 的串联加载。由 \overline{CEO} 输出低电平选通下一级的片选端，可以继续将下一级的 SCP 存储数据写入 LCA 芯片。

(b) 主并加载模式（简称为主并模式或主模式）。

主模式由 LCA 提供 16 位地址、读取时钟 RCLK 和控制信号 D/\overline{P} （或 \overline{LDC} ）。在配置期间， D/\overline{P} 或 \overline{LDC} 为低电平，可以作为外部存储器 EPROM 的片选信号和输出允许的使能信号。数据按数据字节读入后，在内部配置时钟的协助下变换为串行数据，当读入一个数据字节后，每个字节中的最低位 D_0 就是串行数据的第一位。主模式连接图如图 8.5.18 所示。

主模式又分为主低模式和主高模式。当 $M_0M_1M_2=001$ 时，工作在主低模式，16 位地址从 0000 到高地址读取存储器；当 $M_0M_1M_2=011$ 时，工作在主高模式，16 位地址从 FFFF 到低地址读取存储器。这两种模式使 LCA 芯片能与微处理器等器件分享外部存储器。例如，若一个微处理器从地址 0000 开始执行操作，那么，LCA 就可以采用主高模式加载。此时，信号 D/\overline{P} 还可作为微处理器的复位信号，直到配置结束。一旦配置完毕就允许微处理器开始工作。

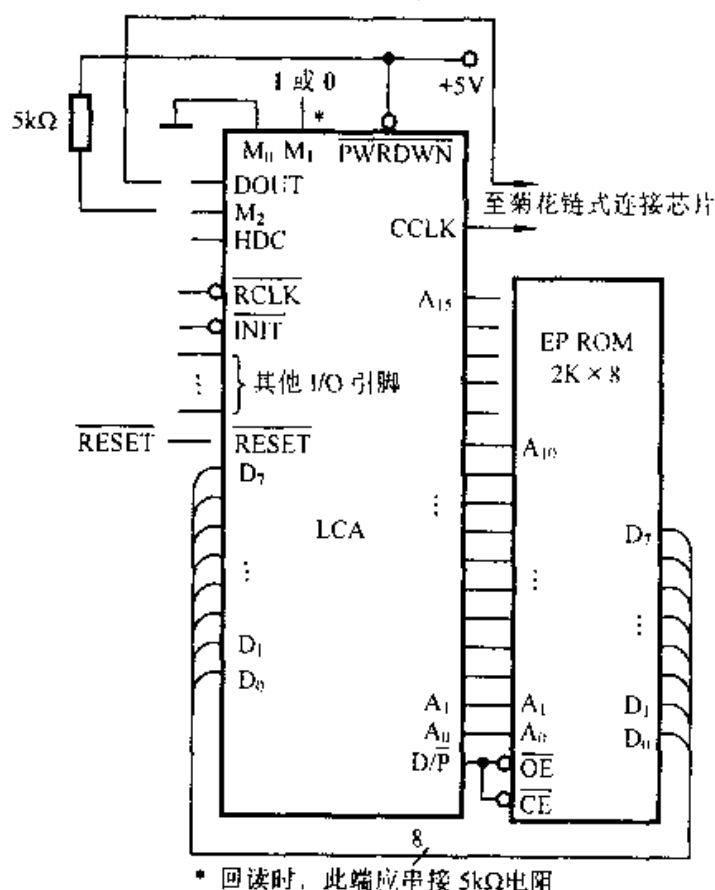


图 8.5.18 主并加载模式连接图

除了主串模式和主并模式外，还有外设模式、从串模式以及实现多片芯片同时加载的菊花链模式。有兴趣的读者可参阅文献[18]。

*8.6 用 CPLD、FPGA 设计数字系统

8.6.1 大规模可编程逻辑器件的设计流程

用 CPLD 和 FPGA 器件设计数字系统一般可以分为设计输入、设计处理和器件编程 3 个步骤,与之相对应的有 3 个验证的过程,分别是功能仿真、时序仿真和在线测试。

1. 设计步骤

(1) 设计输入

在进行设计输入时,由设计者对器件的逻辑功能进行描述,设计输入有多种方式,电路图和 HDL 硬件描述语言是最常用的两种形式。电路图和硬件描述语言有各自的优缺点,硬件描述语言的设计简单,但不适合描述模块间的连接关系;电路图可以很直观地描述连接关系,但在设计逻辑功能时则非常繁琐。为了提高工作效率,应结合具体应用场合来选择合适的输入方式。设计的最上层称为顶层设计,用来描述模块功能分割,以及各模块之间的接口关系,一般可采用电路图的方法来设计。而实际的逻辑功能实现则由下一层模块来完成,采用硬件描述语言来设计则比较方便。

(2) 设计处理

设计处理指从设计输入的文件到最后生成熔丝图文件(CPLD)或者位流文件(FPGA)的整个编译过程。设计处理为设计的核心环节,由计算机软件来完成,设计者仅需设置与“设计实现策略”相关的参数来控制编译的过程。设计处理包括以下几项工作。

(a) 优化:化简逻辑,把逻辑描述转变为最适合在器件中实现的形式。

(b) 合并:将模块化设计中产生的多个文件合并为一个网表文件,使层次设计平面化。

(c) 映射:把设计分为多个适合器件内部逻辑资源实现的逻辑小块的形式,如对于 Altera 和 MAX 7000S 系列器件,要适合用 LAB 和宏单元来实现,而对于 ispLSI 1000 系列器件,要适合用 GLB 来实现。

(d) 布局:将已分割的逻辑小块放到器件内部逻辑资源的具体位置,尽量做到减少各个模块之间的连线。

(e) 布线:利用器件的连线资源,完成各个功能块之间的信号连接。

(f) 编程数据文件:为设计处理最后的结果,该文件将下载到器件中,以配置器件的功能。

(3) 器件编程

器件编程是将设计处理生成的编程数据文件下载到具体的 PLD 器件中,使其按照设计的功能工作。器件编程需要满足一定的条件,如编程电压、编程时序和编程算法等。普通的 CPLD 器件需要专用的编程器来完成器件编程;基于 SPAM 的 FPGA 则可由 PROM 或微处理器来进行器件编程;具有在线可编程性能的 CPLD 和 FPGA 器件,则可以利用计算机的并行口通过 JTAG 电缆,直接对焊接在电路板上的器件编程。

2. 设计校验

设计校验实际上与设计过程是同步进行的,包括功能仿真、时序仿真和在线测试 3 个过程。在设计输入阶段,进行功能仿真来验证设计的功能是否符合设计的要求。时序仿真是在选择了具体的器件并完成了布局、布线后进行的仿真,由于选择不同的器件、不同的布局、布线方案会给设计带来极大的影响,因此,时序仿真对于分析定时关系、估计设计性能是非常必要的。器件编程后,需要在线测试器件的功能和性能指标,看其是否达到最终目标。

设计仿真是 PLD 器件开发的一个重要概念,也是 PLD 器件开发的必不可少的一个过程,借助计算机的仿真软件,设计者可以不必通过实际的器件就能够对要实现的逻辑功能和定时关系进行仿真,并对整个系统的性能进行估计,从而确定设计中是否存在错误,信号的延时是否达到要求,及时地了解错误并改正错误,从而节省开发时间。

8.6.2 用 CPLD、FPGA 设计数字跑表

1. 功能描述

数字跑表是体育比赛中常用的计时器,它通过按键来控制计时的起点和终点。设计的首要任务是确定数字跑表要实现的具体功能,如下所述:

- (a) 系统输入包括 1 个 1kHz 的时钟和两个按键输入 KEY1 和 KEY2;
- (b) 时间精度要求为 10ms,即时间计数器的时钟应为 100Hz;
- (c) 输出到显示电路的信号为 6 组 BCD 码,分别为分钟的十位、分钟的个位、秒的十位、秒的个位、1/10 秒位的 1/100 秒位;
- (d) 时间的计数范围从 00:00:00 到 59:59:99;
- (e) 根据运行状态的不同,KEY1 为清零和暂停键,KEY2 键为启动和停止功能键。

小型系统可采用 CPLD 或 FPGA 来实现,其基本设计流程类似。下面以 CPLD 为例来介绍采用超大规模可编程逻辑器件设计数字系统的方法。

2. 模块的分割

在明确设计要实现的功能的基础上,需要对功能进行模块的划分,使每个模块

的功能相对独立,并确定各个模块之间联系的信号。下面按照自顶向下的设计方法划分数字跑表的功能,如图 8.6.1 所示。

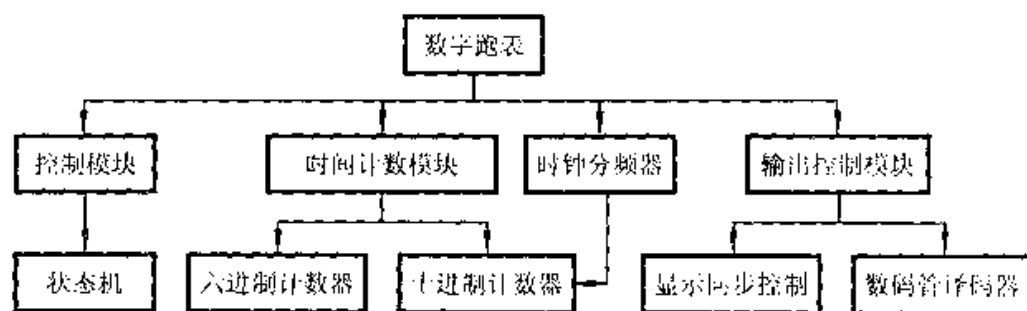


图 8.6.1 数字跑表功能层次划分

控制模块的作用是控制时间计数器的工作状态,两个按键信号用来启动、暂停、清除时间计数器,并且控制显示的刷新。其功能的核心是完成对状态机的控制、产生计数器的控制信号。

时间计数器对时钟脉冲计数,本系统的计时精度要求为 10ms,产生显示的 6 个数码管的 4 位 BCD 码,需要用到十进制计数器和六进制计数器。

为了能够控制显示输出信号与计数器输出之间的同步关系,将计数器信号先送到显示同步控制模块,再由其输出到数码管译码器进行译码。

时钟分频器产生其他模块所需要的时钟信号,系统输入的时钟频率为 1kHz,而其他模块所需要的时钟频率主要包括 2 个:一是时间计数器的时钟,应为 100Hz;另一作为控制电路中状态机的状态转换时钟,该时钟的快慢决定对按键反应速度的快慢,因此,定为 1kHz,不需要分频。为此,需要利用十进制计数器从 1kHz 时钟得到 100Hz 时钟。

3. 模块之间接口的描述

依据模块的划分,下一步需要确定模块之间的接口信号。

控制模块需要产生计数器的控制信号,包括计数使能信号 ENABLE、计数器复位信号 RESET 和输出控制模块的显示刷新信号 REFRESH。

时间计数模块由时钟分频模块输出的 100kHz 信号作为时钟计数,由控制模块输出的 RESET 和 ENABLE 信号复位和使能。其输出的计数值送入输出控制模块。

控制模块产生的 REFRESH 作为显示同步控制模块的使能信号,当 REFRESH 信号有效时,其输出与计数模块的输出保持同步,当 REFRESH 信号无效时,其输出保持不变。

时钟分频模块则是将系统提供的 1kHz 信号 10 分频后送到时间计数器作为计数时钟。此外,还向控制模块的状态机提供运行需要的时钟。

由上所述可以了解数字跑表的各个模块的功能,从而确定数字跑表的结构模块之间的相互联系,其功能框图如图 8.6.2 所示。

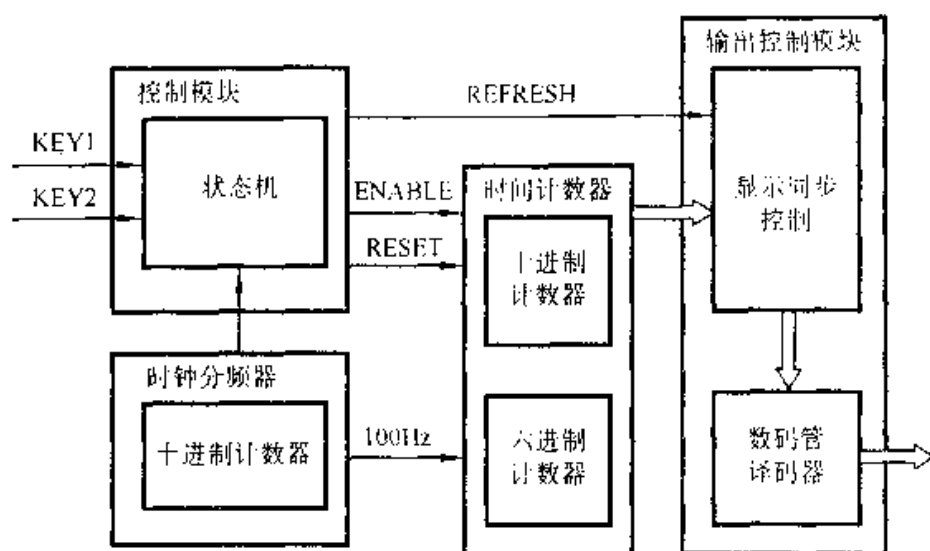


图 8.6.2 跑表功能框图

4. 模块的设计

(1) 顶层电路的设计

顶层设计为了能清晰的描述各模块之间的连接方式，采用电路图的方式，如图 8.6.3 所示。CONTROL 模块为控制器模块，TIME COUNT 为时间计数器模块，COUNT10 用于对系统的 1kHz 时钟进行 10 分频得到 100Hz 时钟，DISP CTRL 模块为显示控制模块。CONTROL 模块受 KEY1 和 KEY2 按键输入控制，其输出的 RESET 和 ENABLE 信号去控制时间计数器，REFRESH 信号控制显示同步控制电路。顶层图中定义了 CONTROL、TIME COUNT、COUNT10 和 DISPCTRL 等 4 个模块的外部特性及相互之间的连接关系，下面进一步描述 CONTROL、TIME COUNT、COUNT10 和 DISPCTRL 等底层模块的内部逻辑功能设计。

(2) 控制器模块

跑表运行一共有 3 个状态，其状态转换图如图 8.6.4 所示。在上电时处于起始状态 S_0 ，此时秒表的计数器不工作。若在该状态下按下 KEY1，则状态保持不变，将计数器的值清零；按下 KEY2 键则状态转至 S_1 ，计数器开始计数。 S_1 为运行状态，秒表以 100Hz 的时钟计数，输出的 6 组 BCD 码随着当前计数值而改变。若在该状态下按下 KEY2，则状态回到 S_0 ，计数暂停；按下 KEY1 键状态转至 S_2 。 S_2 为保持状态，计数器仍然工作，但输出的 6 组 BCD 码并不与计数器的值同步改变，而是每按下一次 KEY2 键刷新一次。若在该状态下按下 KEY1，则状态回到 S_1 。

状态机输出的 3 个信号与其输入信号和状态有着密切的关系，其中 ENABLE 信号在状态 S_0 时无效，计数器停止工作，在状态 S_1 和 S_2 时都应有效，计数器被使能；RESET 信号是在状态 S_0 时按下 KEY1 键将产生一个脉冲信号，将当前计数器的值清零；REFRESH 信号在 S_0 和 S_1 状态应保持有效，使显示值与计数器值保持一致，而在 S_2 状态下，则应是一脉冲信号，每按下一次 KEY2 则产生一个刷新脉冲。

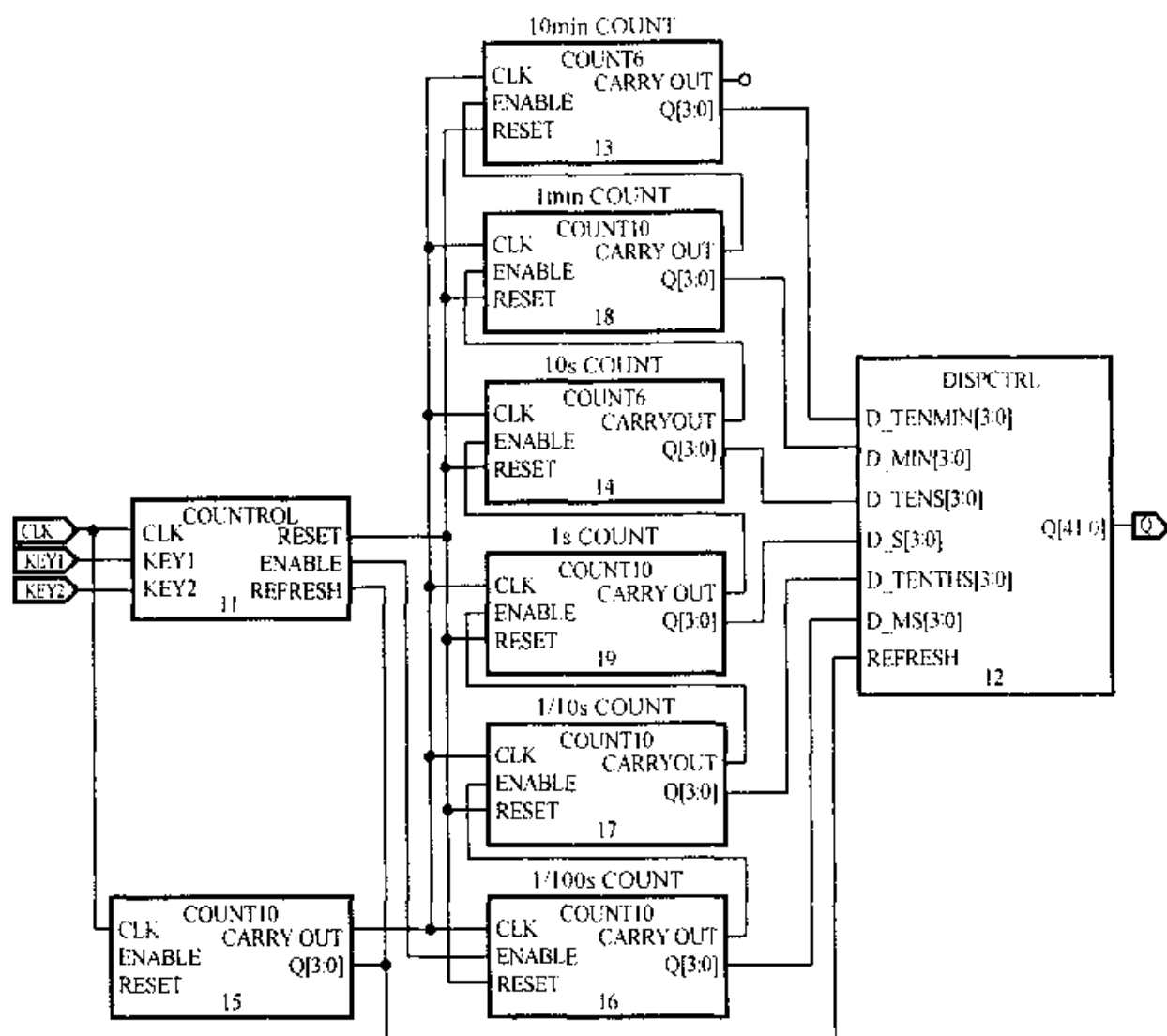


图 8.6.3 顶层电路图

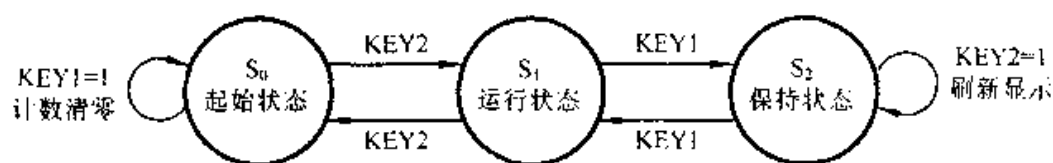


图 8.6.4 秒表状态转移图

用 ABEL-HDL 编写的控制模块实现的核心是状态机，输出根据输入信号和状态而改变，源文件如下：

```
MODULE Control
```

```
Clk,Key1,Key2      PIN;
```

```
En, Reset,Refresh  PIN;
```

```
SS1,SS0            PIN istype'reg';
```

```
SS=[SS1,SS0];
```

```

S0:=^b00;S1:=^b01;S2:=^b10;

EQUATIONS
SS.clk=Clk;

STATE_DIAGRAM SS
  STATE S0:
    En=0;    Reset=Key1;    Refresh=1;
  CASE Key1 & !Key2:S0;
    !Key1 & Key2:S1;
    !Key1 & !Key2:S0;
  ENDCASE;

  STATE S1:
    En=1;    Reset=1;    Refresh=1;
  CASE Key1 & !Key2:S2;
    !Key1 & Key2:S0;
    !Key1 & Key2:S1;
  ENDCASE;

  STATE S2:
    En=1;    Reset=1;    Refresh=Key2;
  CASE Key1 & !Key2:S1;
    !Key1 & Key2:S2;
    !Key1 & !Key2:S2;
  ENDCASE;

  test_vectors([clk,Key1,Key2] ->{SS,En,Reset,Refresh})
    [.c.,1,0] ->[.x.,.x.,.x.,.x.];
    [.c.,0,1] ->[.x.,.x.,.x.,.x.];
    [.c.,0,0] ->[.x.,.x.,.x.,.x.];
    [.c.,1,0] ->[.x.,.x.,.x.,.x.];
    [.c.,0,0] ->[.x.,.x.,.x.,.x.];
    [.c.,0,1] ->[.x.,.x.,.x.,.x.];
    [.c.,1,0] ->[.x.,.x.,.x.,.x.];
    [.c.,0,1] ->[.x.,.x.,.x.,.x.];

END

```

图 8.6.5 给出控制模块的仿真波形，可以看出，状态机的转换和输出信号的波形均符合状态转换图的要求。

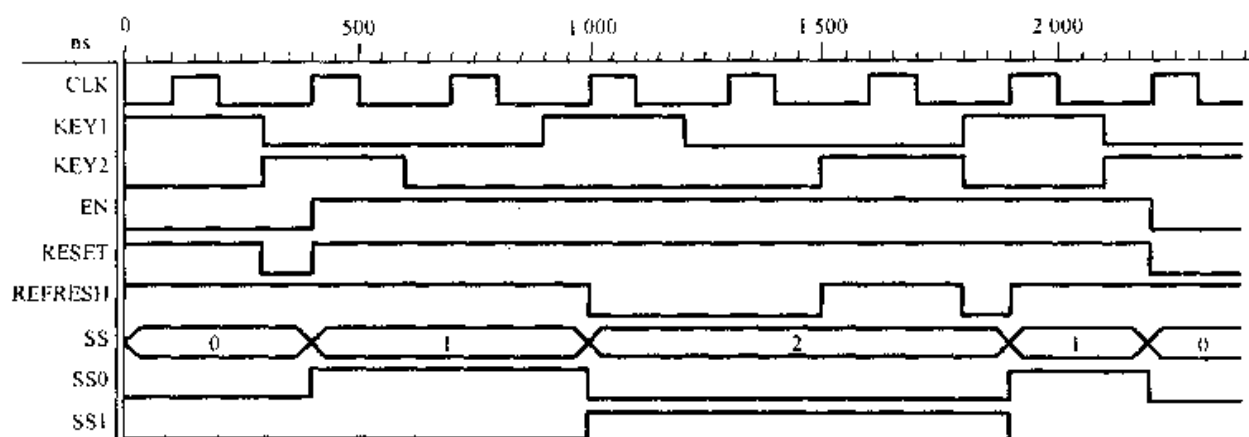


图 8.6.5 控制模块仿真波形

(3) 时间计数器与显示控制模块

时间计数器包括 4 个十进制计数器和两个六进制计数器。每个计数器的计数时钟均为 100Hz，而低位计数器输出的进位信号经整形为 1 个时钟宽度的正脉冲后作为高位计数器的使能信号。所有计数器都由 RESET 信号统一进行复位。计数器的输出送到显示同步控制模块，该模块由锁存器来实现，当其控制信号 REFRESH 为 1 时，输出与输入相同，当控制信号为 0 时，保持当前输出值不变。实现的逻辑框图如图 8.6.6 所示。

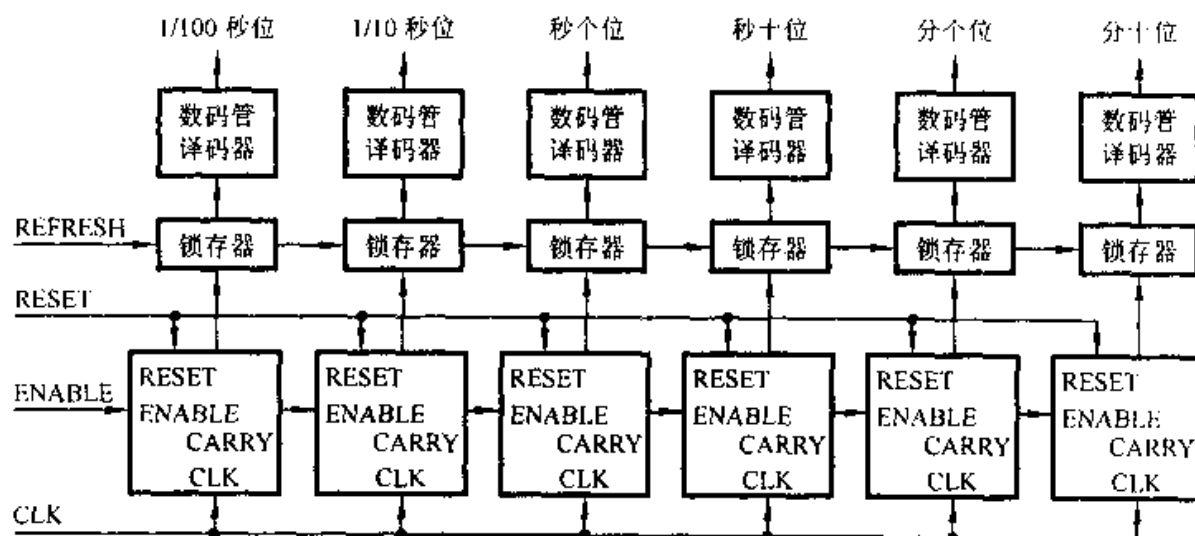


图 8.6.6 时间计数与显示控制模块框图

十进制计数器和六进制计数器的实现基本相同，下面仅给出十进制计数器的 ABEL-HDL 源码。

```

MODULE count10
    Clk,En,Reset      PIN;
    Q0,Q1,Q2,Q3       PIN ISTYPE 'REG';
    CY                 PIN;

```

```

Count=[Q3..Q0];

EQUATIONS
Count.clk=Clk;
Count.ar=!Reset;

WHEN((Count==^H9) & En) THEN Count :=0;
ELSE WHEN(!En) THEN Count := Count.FB;
ELSE WHEN(En) THEN Count := Count.FB+1;

CY=(Count == ^H9) & En;

test_vectors ([En,Reset,Clk] ->[Count,CY])
  @REPEAT 11 {[1,1,.c.] ->[.x,.x.];}
    [0,1,.c.] ->[.x,.x.];
    [1,0,.c.] ->[.x,.x.];

END

```

图 8.6.7 给出十进制计数器仿真的波形，可以看出，当 EN 和 RESET 均为高电平时，计数器正常计数，当 EN 为低电平时，计数器的值保持不变，当 RESET 为低电平时，计数器被清零，且当计数值为 9 时，产生进位脉冲 CY。由波形可以看出，计数器的设计满足设计要求。

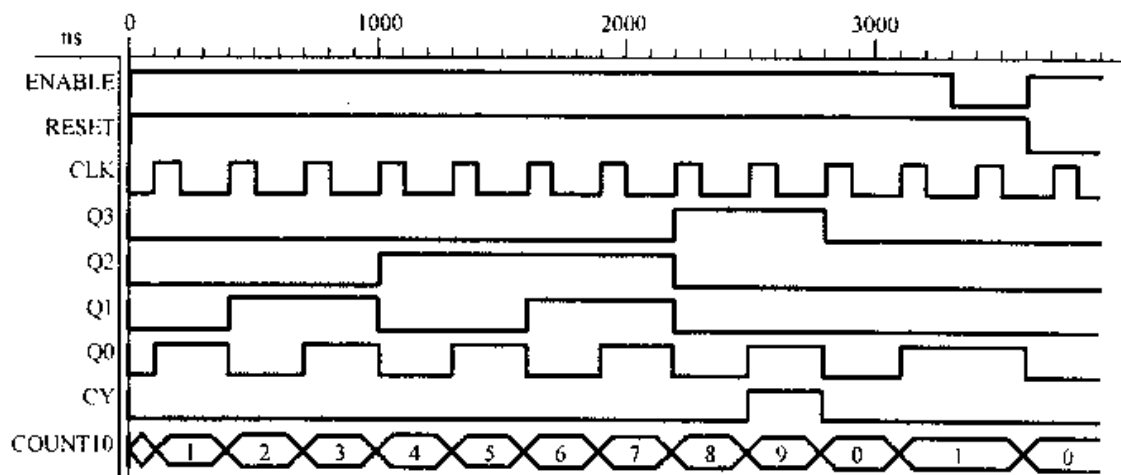


图 8.6.7 十进制计数器仿真波形

锁存器的功能是其使能信号有效时，其输出与输入保持一致，当使能信号无效时，输出保持当前值不变。用 ABEL-HDL 实现锁存器的源码如下。

```

MODULE LATCH4

Refresh      PIN;
D3..D0      PIN;
Q3..Q0      PIN ISTYPE 'com';

```

```

InData=[D3..D0];
OutData=[Q3..Q0];

EQUATIONS
WHEN(Refresh == 1) THEN OutData = InData;
ELSE WHEN(Refresh == 0) THEN OutData = OutData;

TEST_VECTORS([Refresh,InData] ->[OutData])
    [1,^b0101] ->[.x.];
    [1,^b1010] ->[.x.];
    [0,^b1100] ->[.x.];
    [1,^b0011] ->[.x.];
    [0,^b0001] ->[.x.];
    [0,^b1011] ->[.x.];
    [1,^b1000] ->[.x.];
    [1,^b0111] ->[.x.];

END

```

图 8.6.8 给出了锁存器的仿真波形,可以看出,当 REFRESH 信号为高电平时,输出与输入保持一致,当 REFRESH 信号为低电平时,输出信号保持不变。

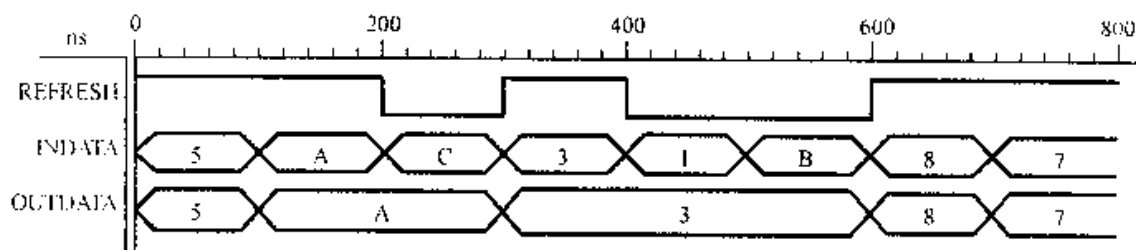


图 8.6.8 锁存器仿真波形

数码管译码器的 ABEL 程序设计参见 8.2.4 节。

(4) 时钟分频电路

时钟分频器对 1kHz 的时钟 10 分频产生 100Hz 的时钟,它的实现与十进制计数器完全一致,可共用其设计代码,计数器的进位输出端即为 100Hz 的时钟。

5. 编程数据的下载

在完成上述文件的输入、编译和仿真后,确认设计的功能已经达到要求,就可以将设计的结果生成下载文件,下载到器件中运行。下面以 Synario 环境为例介绍其操作的具体步骤:

(a) 在器件源中选择 ispLSI 1032EA-100LT100 器件;

(b) 在 Process for current source 过程中,双击 Constraint Editor 打开在 Constraint Editor 程序,分配器件的管脚;

(c) 在 Process for current source 过程中, 双击 Fit Design, 生成熔丝图 JEDEC 文件;

(d) 打开 ispVMSys, 设置 Chain Configuration, 通过 ISP 下载电缆将 JEDEC 文件加载到器件中。此时, 器件按设计的功能运行, 就可以在系统上通过实际运行来检测设计的功能。

小 结

PLD 器件可分为低密度 PLD 和高密度 PLD。目前, 最常用的低密度 PLD 是 GAL 和 FPLA (一般产品目录将其列入 GAL, 有的资料称之为“FPLA 型 GAL”或“新一代 GAL 器件”)。它们的集成度较低, 一般在千门以下, 因此, 功能密度较低, 只能构成板级 (部件级) 芯片。但是, 由于它们含有丰富的与、或阵列资源, 非常适合设计组合逻辑电路。

常用的高密度 PLD 是 CPLD 和 FPGA, 它们的集成度和功能密度都很高, 一般在数千门以上, 目前, 已有 100 万个等效门的商品 (Altera 公司产品 APEX20K), 可构成系统级芯片。高密度 PLD 在电路结构上分为两类: 一类是在 GAL 基础上改进及扩展, 一般称为 GAL (或 PAL) 扩展型结构, 例如, MAX 7000 系列和 ispLSI 1000 系列~ispLSI 3000 系列; 另一类是逻辑单元型结构 (简称为 LCA), 例如, XC 2000 系列~XC 4000 系列。表 8.1~表 8.3 列出了 3 个主要供应商提供的部分产品的主要性能指标。

表 8.1 Xilinx 公司的 FPGA 器件

系列	型号	CLB 数	寄存器 数 量	可用 I/O	可用门数	编程数 据/位	PROM 容量/位	说 明
XC	XC2064	64	122	58	1200	12004	12044	① 系统工作频率为 35MHz ② CMOS 工艺 ③ SRAM 门阵列
	XC2018	100	174	74	1800	17840	17880	
	XC3020	64	256	64	2000	14779	14819	
	XC3030	100	360	80	3000	22176	22216	
	XC3042	144	480	96	4200	30784	30824	
	XC3064	224	688	126	6400	46064	46104	
	XC3090	320	928	144	9000	64160	64200	
	XC4002	64	256	64	2000	31628	31668	
	XC4006	256	768	128	6000	119792	119832	
	XC4010	400	1120	160	10000	178096	178136	
	XC4020	784	2016	224	20000	329264	329304	
	XC4025	1024	2560	256	25000	422128	422168	

表 8.2 Altera 公司的 FPGA 器件

系 列	型 号	等效门	可用 I/O	CLB(或 宏单元)	寄存器	RAM Bus
FLEX 8000	EPF 8282A	2500	78	208	282	无
	EPF 8452A	4000	120	336	452	
	EPF 8820A	8000	152	672	1188	
	EPF 81500A	16000	208	1296	1500	
FLEX 10K	EPF 10K10	10000	134	576	720	6144
	EPF 10K30	20000	246	1728	1968	12288
	EPF 10K70	70000	358	3744	4096	20480
	EPF 10K100	100000	406	4992	5392	24576
EP 20K	EP 20K200C	200000	376	8320	8320	106496
	EP 20K400C	400000	488	16640	16640	212992
	EP 20K600C	600000	588	24320	24320	311296
	EP 20K1000C	1000000	708	38400	38400	327680
MAX 7000	EPM 7032	600	36	32	32	无
	EPM 7096	1800	68/84/100	64	64	
	EPM 7192	3750	124	192	192	
	EPM 7256	5000	132/164	256	256	
MAX 9000	EPM 9320	6000	60/132/168	320	320	无
	EPM 9400	8000	59/139/184	400	400	
	EPM 9480	10000	117/175/200	480	480	
	EPM 9560	12000	153/191/216	560	560	

表 8.3 Lattice 公司的 FPGA 器件

系 列	型 号	等效门	可用 I/O	宏单元	寄存器	f_{\max}/MHz
ispLSI 1000	1016	2000	36	64	96	110
	1024	4000	54	96	144	90
	1032	6000	72	128	192	90
	1048/c	8000	106/108	192	288	80/70
ispLSI 2000	2032	1000	34	32	32	154
	2064	2000	68	64	64	125
	2096	4000	102	96	96	125
	2128	6000	136	128	128	100
ispLSI 3000	3192	8000	192	192	288	100
	3256	11000	128	256	384	77
	3320	14000	160	320	480	77

CPLD 和 FPGA 融合了 GAL 和 ASIC 的优点,在一定程度上又克服了它们的缺点。因此,对它们的开发应用必将有力地推动数字设备向小型化、低功耗、高可靠性和开发周期短的方向发展。

思考题和习题

- 8.1 名词解释：
互连资源，编程数据，PROM 容量，特征频率，系统频率，等效门。
- 8.2 在 FPGA 产品中，为什么 LCA 系列应用最多？
- 8.3 CPLD 和 FPGA 与 GAL 和 ASIC 相比较，有何优点？
- 8.4 与采用中小规模集成电路设计数字电路相比，采用 CPLD/FPGA 器件设计数字电路有何优点？
- 8.5 比较 MAX 7000S 与 ispLSI 1032EA 器件的内部结构，总结 CPLD 内部结构的基本特征。
- 8.6 试用 PLA 可编程逻辑阵列实现下列码组转换：
(1) 二进制码至余 3 码 (2) 余 3 码至格雷码
- 8.7 试用 PLA 阵列实现下列函数，并要求阵列点数最少：
(1) $F_1 = A\bar{C}\bar{D} + CD + ABD + \bar{A}BC$ (2) $F_2 = A\bar{B}CF + \bar{B}\bar{D} + \bar{B}EF$
(3) $F_3 = ACF + \bar{B}EF + \bar{B}\bar{D} + ACF$
- 8.8 用 PLA 阵列设计余 3 码十进制计数器和 8 段显示译码器，并要求一旦出现非法码组时，8 段显示译码器能驱动显示电路显示“+”字符。
- 8.9 图 P7.2 所示组合电路，若用 PLA 阵列来实现，画出逻辑阵列图其存储容量为多大？
- 8.10 用 GAL 16V8 设计一个 3-8 线译码器。
- 8.11 用 GAL 16V8 设计一个 BCD 码计数器。
- 8.12 用 ispLSI 1032 设计一个 1 位十进制数 (8421 BCD 码表示) 加法器。
- 8.13 利用 XC 3020 的开关矩阵工作模式 (参见图 8.5.10) 完成图 P8.1 (d) A 至 B 点的连接图。

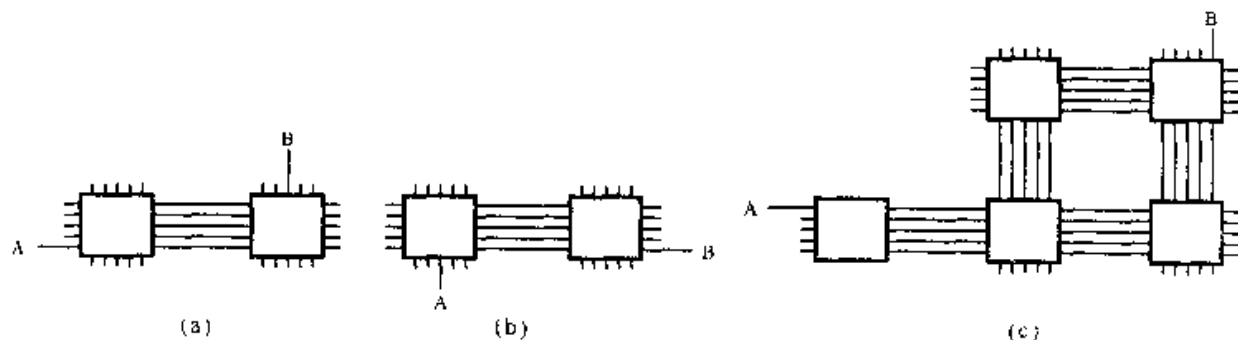


图 P8.1

- 8.14 水平长线和垂直长线有何功能？使用它们应注意什么问题？
- 8.15 依据什么规则来选用 FPGA 芯片内的互连方式。
- 8.16 XC 2064 为 160 帧，每帧数据为 75 位，问编程数据和 PROM 容量各为多少位？画出数据结构图。
- 8.17 XC 3090 有 PROM 容量为 64200 位，已知每帧数据为 172 位，问该芯片有多少帧；画出帧结构图。

第 9 章 脉冲单元电路

内容提要 本章主要讨论矩形脉冲波形的产生、变换及整形的单元电路。分别介绍由集成运算放大器和 555 定时器构成的单稳态触发器、施密特触发器和自激多谐振荡器等。着重介绍电路的结构和工作原理、工作波形图以及主要技术指标；同时也介绍了它们的应用。

9.1 概 述

9.1.1 脉冲信号与脉冲电路

在绪论中曾指出：模拟信号分为正弦信号和脉冲信号两类。因此，从广义上讲，凡不具有连续正弦波形状的信号都可以统称为脉冲信号。图 9.1.1 中示出了几种最常见的脉冲信号，其共同的特点是整个波形是由若干个暂态过程段所组成的。

脉冲电路是指产生、传输、变换和处理脉冲信号的电路。它们一般由开关电路(晶体管或场效应管)和惰性网络(RC 或 RL)组成，也可以由运算放大器或 555 定时器和 RC 组成，后者是本章介绍的主要内容。

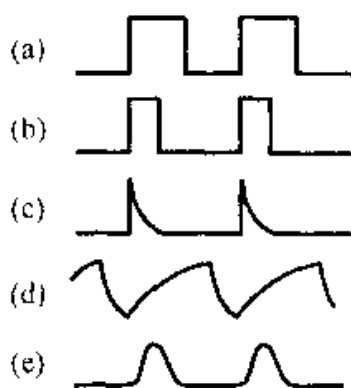


图 9.1.1 脉冲波形

(a) 方波 (b) 矩形波 (c) 尖脉冲
(d) 锯齿波 (e) 钟形脉冲

9.1.2 脉冲信号的主要参数

脉冲信号种类繁多，因此，描述它们特性的参数也有些差异。方波和矩形脉冲是使用最广泛的脉冲信号，下面以它们为例来介绍脉冲信号的主要参数。

理想的方波和矩形脉冲的突变部分是瞬时的，不占用时间。但实际波形中，波形的上升或下降总是需要一定时间的，如图 9.1.2 所示。图中描述的矩形脉冲特性的主要参数定义如下。

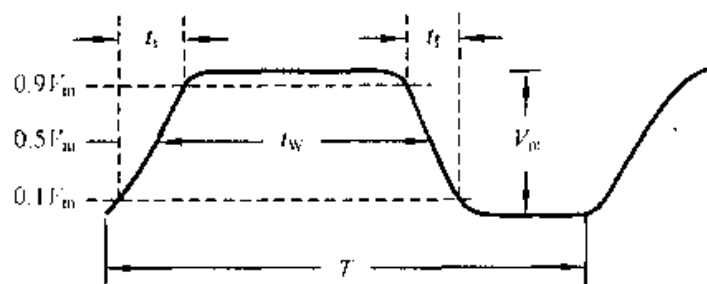


图 9.1.2 实际的矩形脉冲波形

- 脉冲幅度 V_m : 脉冲电压变化的最大幅度。
- 脉冲宽度 t_w : 从脉冲前沿上升到 $0.5V_m$ 起, 到脉冲后沿到达 $0.5V_m$ 为止的时间。
- 上升时间 t_r : 脉冲上升沿从 $0.1V_m$ 上升到 $0.9V_m$ 所需要的时间。
- 下降时间 t_f : 脉冲下降沿从 $0.9V_m$ 下降到 $0.1V_m$ 所需要的时间。
- 脉冲周期 T : 周期性重复的脉冲序列中, 两个相邻脉冲之间的时间间隔
- 占空比 q : 脉冲宽度与脉冲周期的比值, 即 $q=t_w/T$ 。

*9.1.3 集成运算放大器的传输特性

集成运算放大器是一种高性能的直接耦合放大器, 是线性集成电路中应用最为普遍的一种器件。它具有很大的开环电压增益(达 60~120dB), 很高的输入阻抗(达数十千欧~上百兆欧), 输出阻抗很小(一般为百欧数量级), 失调小、温漂小, 而且便于灵活运用, 故用途极广。通过改变集成运算放大器的输入和反馈电路的形式和参数, 可实现不同的运算关系, 从而满足各种不同的实际需要。若将输出反馈到反相输入端, 则可以构成各种负反馈放大器、积分器、微分器、限幅器等; 若将输出反馈到同相输入端, 利用正反馈特性则可构成各种脉冲单元电路, 如本章将要介绍的施密特触发器、单稳态触发器和自激多谐振荡器等。因此, 我们先简要介绍集成运算放大器(简称为集成运放)的传输特性。

如果在集成运算放大器的反相端加直流参考电压 V_R , 而在其同相端加输入电压 v_i , 如图 9.1.3(a)所示, 则其输出电压与输入电压的关系曲线——传输特性如图 9.1.3(b)所示。图中, 横坐标是 $v_D=v_i-V_R$, 以 mV 为单位; 纵坐标是输出电压 v_O , 以 V 为单位。参考电压可以是正值, 也可以是负值或零, 图 9.1.3(a)中给出的 V_R 为正值。这实际上就是比较器的基本电路。

由图 9.1.3(a)所示的电路可见: 线性组件是处于开环工作状态。由于集成运放具有很高的开环电压增益, 所以, 只要放大器的两个输入之间有微小的差动输入电压 v_D , 它的输出级就进入饱和工作状态, 其输出电压将达到饱和值(V_{OH} 或 V_{OL})。线性放大区的输入电压范围通常只有几 mV。例如, 设 $|V_{OH}|=|V_{OL}|=10V$, 开环增益

$A_v=10^4$, 则为了使输出电压从负向饱和值 V_{OL} 翻转到正向饱和值 V_{OH} 所需的输入电压值仅 2mV , 超过了这个范围, 输出级就处于饱和工作状态, 输出电压就偏向其饱和值而不再随输入电压变化。

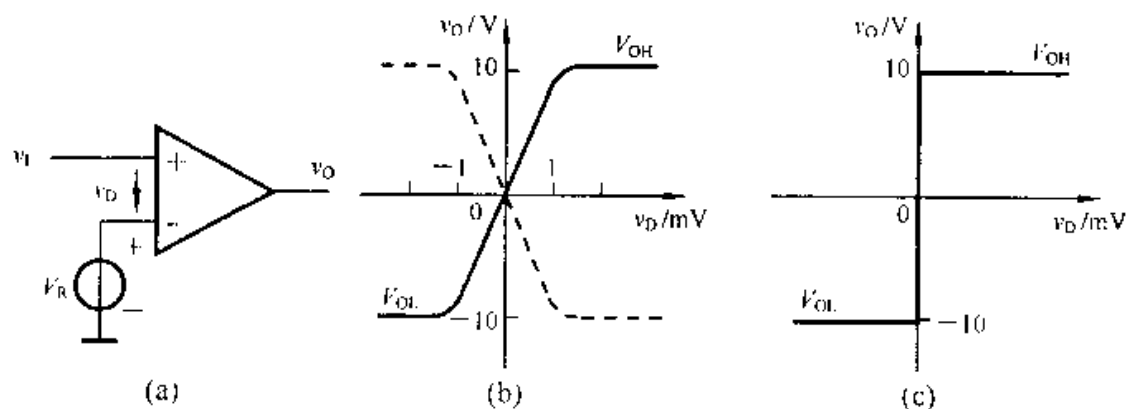


图 9.1.3 集成运算放大器的传输特性

(a) 电路图 (b) 传输特性 (c) 等效传输特性

若将图 9.1.3(b)所示的传输特性的横坐标改成以 V 为单位, 则范围以 mV 计的线性放大区几乎被压缩成一条垂直线, 如图 9.1.3(c)所示。由此可见, 用集成运放作为开关元件或比较器是相当合适的, 输入电压 v_i 以参考电压 V_R 作为分界:

当 $v_i < V_R$, 即 $v_D = v_i - V_R < 0$ 时, 组件输出为低电平, $v_o = V_{OL}$;

当 $v_i > V_R$, 即 $v_D = v_i - V_R > 0$ 时, 组件输出为高电平, $v_o = V_{OH}$ 。

如果将参考电压接到同相端, 输入电压 v_i 接到反相端, 则可以得到另一条传输特性曲线, 如图 9.1.3(b)虚线所示。

如果参考电压 $V_R=0$, 则输入电压 v_i 每次过零时, 输出电压就要产生突变, 这实际上就是一个过零比较器。

作开关用的线性集成电路称为电压比较器, 如国产 BG307, 其输出高电平约为 3.1V , 输出低电平约为 -0.5V , 便于与 TTL 数字电路相匹配。

9.2 施密特触发器

9.2.1 集成门构成的施密特触发器

1. 工作原理与回差特性

(1) 工作原理

图 9.2.1(a)所示的是由集成与非门组成的施密特触发器, 门 G_1 、 G_2 组成基本 RS 触发器, 二极管 D 起电平偏移作用。

(a) 第一稳定状态：假定输入信号 v_i 为三角波，如图 9.2.1(b)所示。设起始状态 v_i 为低电平，则门 G_2 、 G_3 输出端均为高电平，由于门 G_1 的两个输入端均为高电平，因此，输出端 Q 为低电平，这是电路的第一稳态($Q=0$, $\bar{Q}=1$)。

(b) 第一次翻转：当 v_i 上升到 $0.7V$ 时，由于二极管的电平偏移作用，使门 G_2 的输入端 R 上的电压为 $v_i+v_D=(0.7+0.7)V=1.4V$ ，但由于门 G_3 仍处于关闭状态、 v_{O3} 为高电平， Q 端输出仍为低电平，反馈到 G_2 门的另一个输入端，使门 G_2 仍维持关门状态。当 v_i 继续上升到 $V_{TH}=1.4V$ 时，门 G_3 开通， v_{O3} 为低电平，使 S 端低于 V_T ，门 G_1 关闭，其输出跳变到高电平，从而使基本 RS 触发器状态翻转，电路进入第二稳态($Q=1$, $\bar{Q}=0$)。

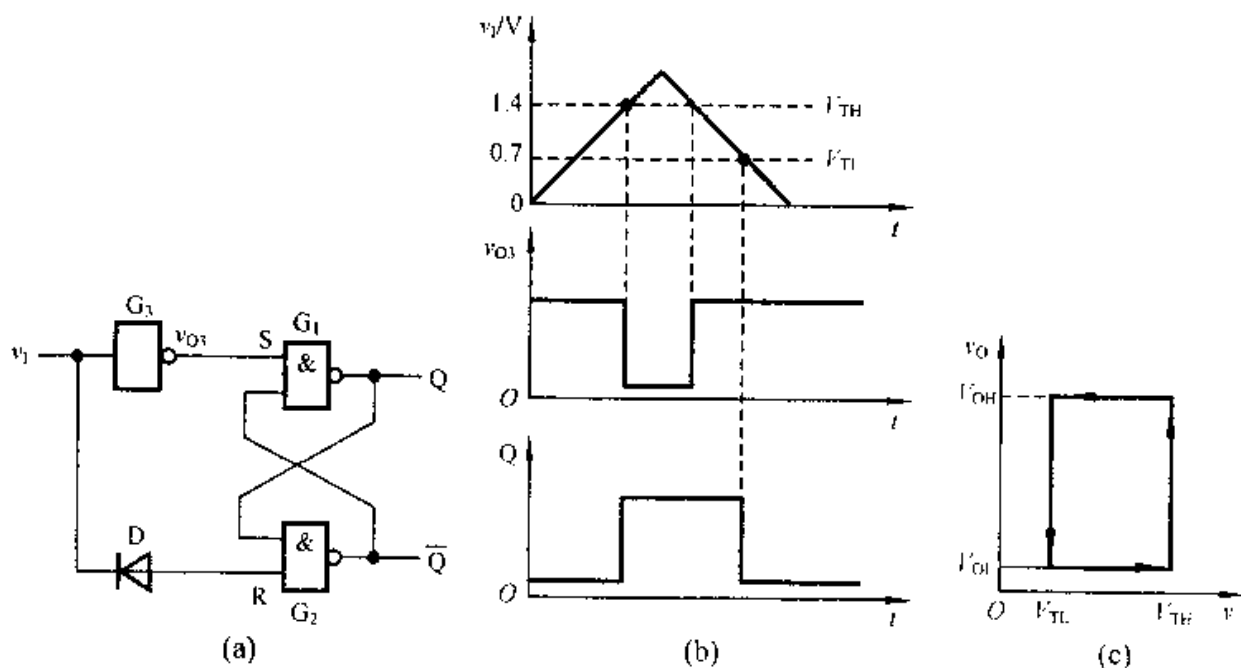


图 9.2.1 集成门构成的施密特触发器
(a) 电路图 (b) 波形图 (c) 传输特性

(c) 第二稳定状态：当电路进入第二稳态后， v_i 继续升高。当 v_i 上升到三角波的最大值后开始下降，但只要 v_i 大于 $V_{TH}=1.4V$ ，电路就维持 G_2 门开通、 G_1 门关闭的状态不变。

(d) 第二次翻转：当 v_i 下降到 V_{TH} 时， G_3 门关闭， v_{O3} 为高电平，但由于此时 R 端的电压等于 $(1.4+0.7)V=2.1V$ ， G_2 门仍处于开通状态。当 v_i 继续下降到 $V_{TL}=0.7V$ 时，使 R 端的电平低于 V_T ， G_2 门关闭，其输出跳变到高电平，引起 RS 触发器的状态第二次翻转，回到第一稳态， $Q=0$, $\bar{Q}=1$ 。

施密特触发器具有两种稳定的工作状态，它处于哪一种稳态，取决于输入信号的电平。图 9.2.1(c)示出其电压传输特性。当 v_i 上升到 V_{TH} 时，电路发生翻转；当 v_i 下降到 V_{TL} 时，电路状态又会发生翻转，显然两次翻转所对应的输入电平值是不同的。

(2) 回差特性

由上面分析可见, 当 v_i 上升到 $V_{TH}=1.4V$, 触发器由第一稳态翻转到第二稳态, 但是当 v_i 下降到 V_{TH} 时, 电路仍维持在第二稳态不变, 只有当 v_i 降到 $V_{TL}=0.7V$ 时, 触发器才发生第二次翻转, 返回第一稳态。称 V_{TH} 为上限触发电平(或接通电位), V_{TL} 为下限触发电平(或断开电位), 它们之间的差值称为回差电压(或滞后电压), 用 ΔV 表示, 即

$$\Delta V = V_{TH} - V_{TL}$$

回差特性是施密特触发器的一个重要特性。对回差特性的要求视其应用场合而有所不同。当用它作整形和波形变换时, 希望回差电压越小越好, 如图 9.2.2(b)所示。若回差电压大于信号电压幅度, 则电路一旦翻转后就不能再次翻转, 也就得不到矩形波输出了, 如图 9.2.2(c)所示。

当利用回差特性消除干扰时, 就要适当选择电路参数, 保证有一定回差电压, 以消除叠加在输入信号上的干扰, 输出波形如图 9.2.3 所示。图(b)表示 $\Delta V'$ 太小导致输出错误波形; 图(c)表示 ΔV 大于顶部干扰信号幅度时的输出波形。

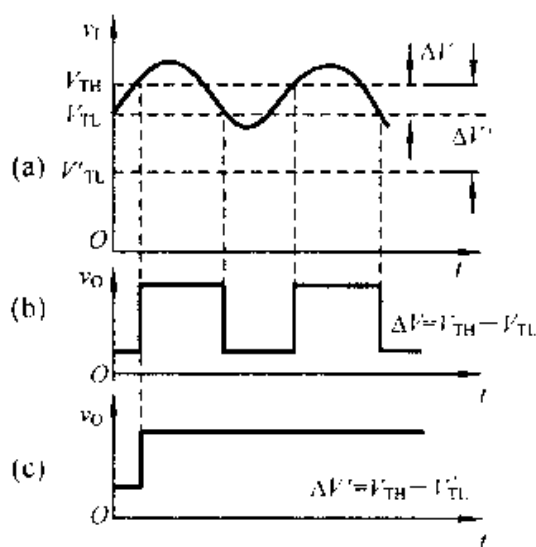


图 9.2.2 回差电压在波形变换中的作用

- (a) 具有不同回差电压的输入信号
- (b) 回差电压小时的输出信号
- (c) 回差电压大时的输出信号

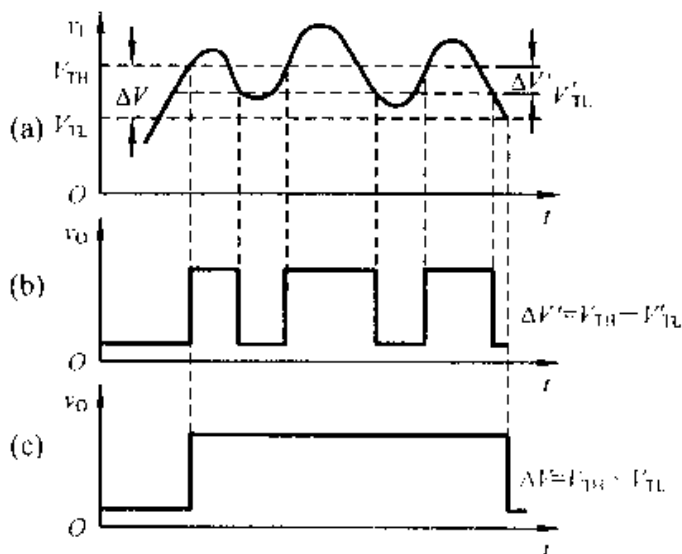


图 9.2.3 利用回差电压抗干扰

- (a) 具有顶部干扰的输入信号
- (b) 回差电压小时的输出波形
- (c) 回差电压大于顶部干扰时输出波形

2. 施密特触发器的应用

施密特触发器的主要用途有: 波形变换、整形、比较、鉴幅等。

(a) 波形变换及整形: 利用施密特触发器可以将正弦波、三角波以及各种周期性的不规则波形变换成规范的矩形波, 因此, 这种电路常被用作计数器的输入整形

电路,使输入信号经整形后变成边沿陡峭的矩形脉冲,作为计数器的时钟脉冲,如图 9.2.2(b)所示。

(b) 抑制干扰:利用施密特触发器的回差特性可以抑制叠加在输入信号上的干扰,使输出波形变成理想的矩形波,如图 9.2.3(c)所示。

(c) 脉冲幅度鉴别:在脉冲信号的传输过程中,经常混有噪声信号,为了选出有用的信号而除去噪声,可以利用施密特电路的特性,使电路的上限触发电平 V_{TH} 等于规定的鉴幅电压 V ,这样,只有幅度超过 V 的信号脉冲才能使电路产生翻转,电路才有相应的输出,而幅度小于 V 的噪声则无输出,如图 9.2.4 所示。

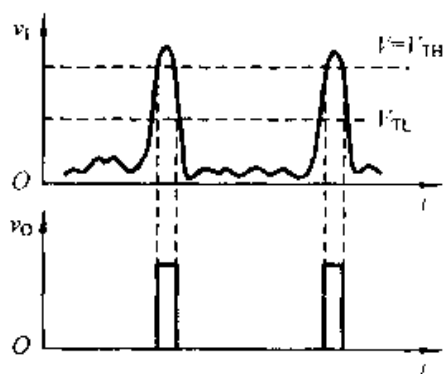


图 9.2.4 幅度鉴别示意图

9.2.2 集成运放构成的施密特触发器

1. 电路及工作过程

由集成运放所构成的施密特触发器及其波形图如图 9.2.5 所示。这实际上是一个具有正反馈的电压比较器,输出电压通过 R_1 和 R_2 组成的分压器加到放大器的同相端形成正反馈。通常电阻 R_2 的值远小于集成运放的输入电阻,因此,集成运放的输入电阻对分压值影响很小, R_1 、 R_2 分压点 P 点的电压 $v_P = v_O R_2 / (R_1 + R_2)$,电路的正反馈系数为

$$F = R_2 / (R_1 + R_2)$$

所以,环路增益(闭环增益)为

$$A_{VF} = A_V \cdot F$$

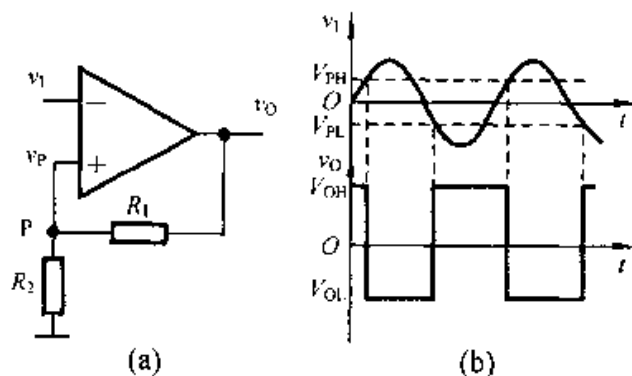


图 9.2.5 集成运放构成的施密特触发器

(a) 电路图 (b) 波形图

当外加输入电压 v_i 低于同相输入端电压, 即 $v_i < v_p$ 时, 则放大器输出为高电平, $v_o = V_{OH}$, 因此, 同相端的电压

$$v_p = V_{PH} = V_{OH} R_2 / (R_1 + R_2)$$

当输入电压 v_i 增加到略大于 V_{PH} 时, 放大器进入线性放大区, 由于电路的闭环增益大于 1, 即 $A_v R_2 / (R_1 + R_2) > 1$, 使电路迅速翻转到另一个状态, 输出低电平, $v_o = V_{OL}$ 。由于 v_o 的降低, 同相端的电位也相应降低到 $v_p = V_{PL} = V_{OL} R_2 / (R_1 + R_2)$, 只要 $v_i > V_{PL}$, 电路就稳定在这个状态。当 v_i 下降到略低于 V_{PL} 时, 电路就又产生翻转, 回到输出 $v_o = V_{OH}$ 的状态。

2. 回差

当施密特触发器的输出由高电平 V_{OH} 转换到低电平 V_{OL} 时的输入转换电平 $V_{TH} = V_{PH}$ (上限触发电平), 即

$$V_{TH} = V_{OH} R_2 / (R_1 + R_2)$$

当施密特触发器的输出由低电平 V_{OL} 转换到高电平 V_{OH} 时的输入转换电平 $V_{TL} = V_{PL}$ (下限触发电平), 即

$$V_{TL} = V_{OL} R_2 / (R_1 + R_2)$$

上、下限触发电平之差称为回差, 以 ΔV 表示:

$$\Delta V = V_{TH} - V_{TL} = (V_{OH} - V_{OL}) R_2 / (R_1 + R_2)$$

上式表明: 调节反馈系数 $R_2 / (R_1 + R_2)$ 大小便可控制回差电压大小, 因此, 该电路适用于对回差电压有不同要求的各种场合, 并且调节十分简便。

上面所讨论的施密特触发器具有线路简单、回差容易控制的优点, 但电路的精度和稳定度不够高。这是因为触发器的上、下限触发电平与集成运放的输出电平有很大关系。由于受电源电压的影响, V_{OH} 和 V_{OL} 不太稳定也不够精确, 使施密特电路的上、下限触发电平的精度和稳定度也较差。在电路上采取一些措施, 例如, 在输出端加上由电阻 R_3 和两个背靠背连接的稳压管 D_{W1} 和 D_{W2} , 使输出经过稳压后再反馈 (如图 9.2.6 所示), 可提高 V_{PH} 和 V_{PL} 的精度和稳定度。

由于施密特触发器具有良好的波形整形功能, 所以, 有许多集成芯片都用它作为输入电路, 并且在手册上都会标注。例如, 74LS13 是 4 输入双与非门 (施密特触发); CC40106 是六反相器 (施密特触发) 等。

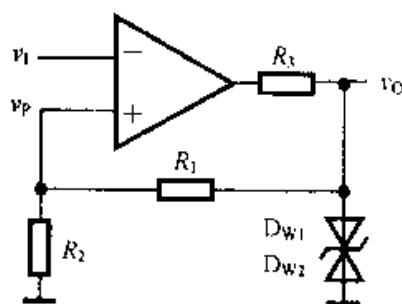


图 9.2.6 改进型的施密特触发器

9.3 单稳态触发器

单稳态触发器具有一个稳定状态和一个暂稳状态。在触发信号没有加入之前,电路处于稳定状态,在外加触发脉冲作用下,电路可转换成另一个状态,但这个状态只是暂时维持的,经过一段时间之后,电路将自动返回到原来的稳定状态。

单稳态触发器在脉冲数字系统中的应用十分广泛。它主要用作脉冲的整形,把波形不规则的脉冲改造成宽度和幅度都一致的脉冲,也可以用作脉冲的延时、定时以及清除噪声电路等。

9.3.1 集成运放构成的单稳态触发器

1. 电路结构

由集成运算放大器构成的单稳态触发器如图 9.3.1(a)所示。电阻 R_3 与稳压管 D_{W1} 、 D_{W2} 构成双向对称限幅电路,使输出电压 v_O 为 $+V_W$ 或 $-V_W$ 。输出电压一方面经过电阻 R_1 和 R_2 组成的分压器反馈到同相输入端形成一定的正反馈,另一方面又经 R 和 C 构成的积分器反馈到反相输入端。二极管 D_1 并在积分电容 C 两端,使得当 $v_O = +V_W$ 时,反相端的电位等于二极管的正向压降,即 $v_N = V_D$ 。 C_T 、 R_T 组成微分电路,用来将输入的触发方波信号微分成正、负尖脉冲。二极管 D_T 只让负脉冲通过去触发单稳态电路(正脉冲被限幅除掉)。

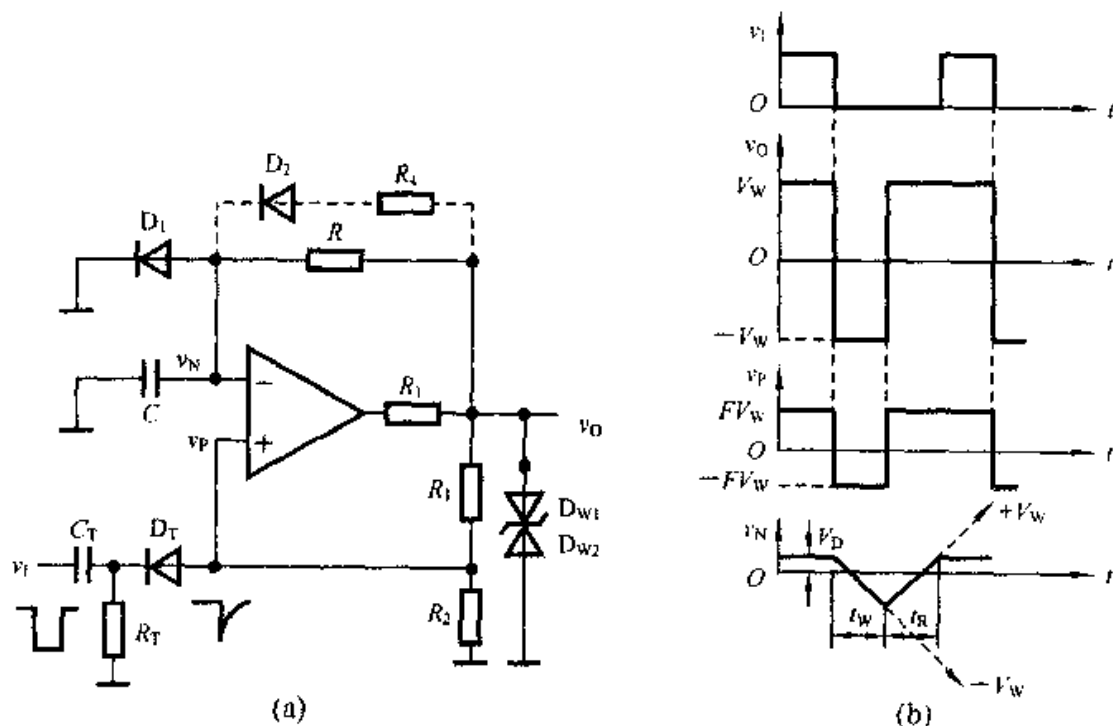


图 9.3.1 集成运放构成的单稳态触发器

(a) 电路图 (b) 波形图

该电路正常工作的条件是：适当选取电阻 R_1 、 R_2 之值，使得在没有外界触发的情况下，同相端的电位 v_P 大于二极管的正向压降，即 $v_P = +R_2/(R_1+R_2)V_W = +FV_W > V_D$ ，输出电压保持在 $v_O = +V_W$ ，这就是电路的稳定状态。

当外加触发脉冲到来时，只要负触发脉冲幅度大于 $(FV_W - V_D)$ ，就能使单稳态电路翻转，使输出电压从 $+V_W$ 转换成 $-V_W$ ，同相端的电位 v_P 也相应地从 $+FV_W$ 迅速跳变到 $-FV_W$ 。于是电路进入暂稳态，二极管 D_1 因反偏而截止，电容 C 通过电阻 R 向 $-V_W$ 放电。随着 C 的放电， v_N 逐渐下降，当 v_N 降到略低于 $-FV_W$ 时，电路又迅速翻转，使输出由 $-V_W$ 突变为 $+V_W$ ，暂稳态结束，电路进入恢复期。输出电压 $-V_W$ 通过 R 向 C 充电， v_N 逐渐上升，当 v_N 上升到等于 $+V_D$ 时，二极管 D_1 导通，将 v_N 钳在 V_D ，电路恢复到起始稳态，等待下一次触发脉冲的到来。各点波形如图 9.3.1(b) 所示。

2. 波形参数

输出脉冲宽度可以用三要素法求出。在暂稳态期间，电容 C 放电： $v_N(0) = V_D$ ， $v_N(\infty) = -V_W$ ， $v_N(t_W) = -FV_W$ ， $\tau_d = RC$ 。因此，

$$t_W = \tau_d \ln \frac{-V_W - V_D}{-V_W + FV_W} = RC \ln \frac{V_W + V_D}{V_W(1-F)} = RC \ln \frac{V_W + V_D}{V_W} \cdot \frac{R_1 + R_2}{R_1}$$

若取 $R_1 = R_2$ ，且通常 $V_W \gg V_D$ ，则

$$t_W \approx RC \ln 2 = 0.69RC$$

恢复时间 t_R 同样可用三要素法求出。在恢复期，电容 C 充电： $v_N(0) = -FV_W$ ， $v_N(\infty) = +V_W$ ， $v_N(t_R) = +V_D$ ， $\tau_c = RC$ 。因此，

$$t_R = \tau_c \ln \frac{V_W + FV_W}{V_W - V_D} \approx RC \ln \frac{V_W(1+F)}{V_W} = RC \ln(1+F)$$

若取 $R_1 = R_2$ ，则 $t_R \approx RC \ln 1.5$ 。

该电路具有温度稳定性好、脉冲宽度调节范围大、调试简便等优点；缺点是所需要的恢复时间与输出脉冲宽度相当，这是因为电容 C 的充放电时间常数相同的缘故。为了缩短恢复期，可以在 R 两端并联一个由二极管 D_2 和电阻 R_4 组成的串联支路，使得充电时间常数为 $\tau_c = (R//R_4)C$ ，只要 R_4 选择得较小， τ_c 就比较小，电路就可以迅速恢复到稳定状态。

9.3.2 单片集成单稳态触发器

用集成门也可构成品种繁多的单稳态触发器。其产品分为不可重复触发和可重复触发两类。属于不可重复触发的产品有 74121、74221、74LS221 和 CMOS 电路 CC74HC123 等；属于可重复触发的产品有 74122、74LS122、74123、74LS123 和 CMOS 电路 CC14528、CC14538 等。使用这些器件时只需外接定时元件即可，多数器件内

部还设置有上升沿触发与下降沿解发的控制或异步清除功能，使用非常方便。

不可重复触发单稳态电路一旦被触发进入暂稳态后，再加入触发脉冲是无效的，必须在暂稳态结束之后才接受下一个触发脉冲，重新进入暂稳态，如图 9.3.2(a) 所示。可重复触发单稳态触发器在被触发进入暂稳态之后，若再次加入触发脉冲，则这些触发脉冲是有效的，电路将重新被触发，使输出脉冲再继续维持 t_W 宽度，如图 9.3.2(b) 所示，因此，输出脉冲宽度将为 $\Delta t+t_W$ 。

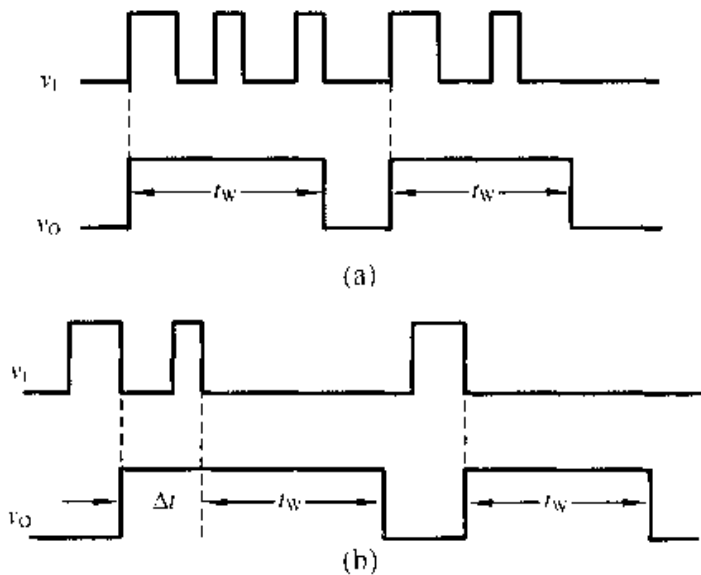


图 9.3.2 两种单稳态电路工作波形

(a) 不可重复触发单稳态触发器工作波形 (b) 可重复触发单稳态触发器工作波形

本节对单片集成单稳态触发器的内部电路结构不作深入讨论，主要介绍器件的外部特性、功能表和使用方法。

1. 不可重复触发集成单稳态触发器

74121 型单稳态触发器是由集成门构成的不可重复触发单稳态触发器。逻辑框图如图 9.3.3 所示，功能表如表 9.3.1 所示。 \overline{A}_1 、 \overline{A}_2 是负跳变触发脉冲输入端；B

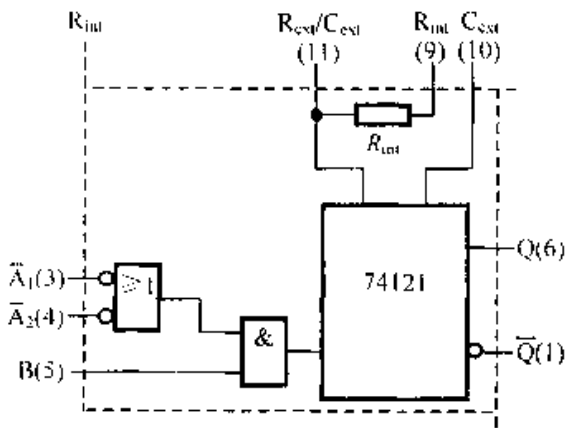


图 9.3.3 74121 逻辑框图

表 9.3.1 74121 的功能表

输 入			输 出	
\overline{A}_1	\overline{A}_2	B	Q	\overline{Q}
0	ϕ	1	0	1
ϕ	0	1	0	1
ϕ	ϕ	0	0	1
1	1	ϕ	0	1
1	1	1	1	0
1	1	1	1	0
1	1	1	1	0
0	ϕ	1	1	0
ϕ	0	1	1	0

是正跳变触发脉冲输入端； Q 和 \bar{Q} 是互补输出端，稳态时 $Q=0$ ， $\bar{Q}=1$ 。在触发脉冲作用下， Q 端输出正脉冲， \bar{Q} 端输出负脉冲。 R_{int} 是内定时电阻的引出线端， C_{ext} 和 R_{ext}/C_{ext} 是外接定时元件的接线端。该组件的特点如下。

(a) 输出脉冲宽度仅是定时元件的函数，输入脉冲宽度可以比输出脉冲宽，也可以比输出脉冲窄。

当不用外接定时元件时(即 R_{int} 接 V_{CC} ， C_{ext} 和 R_{ext}/C_{ext} 端开路，如图 9.3.4(a)所示)，可得到宽度典型值为 $30\text{ns}\sim 35\text{ns}$ 的输出脉冲。

当用外接定时元件时(即在 C_{ext} 和 R_{ext}/C_{ext} 端之间外接电容，在 R_{ext}/C_{ext} 和 V_{CC} 端之间外接一电阻 R_{ext} ，如图 9.3.4(b)所示)，适当选择定时元件的参数值，输出脉冲宽度可在 $40\text{ns}\sim 28\text{s}$ 之间变化。输出脉冲宽度 t_W 由下式确定：

$$t_W \approx 0.7 R_{ext} C_{ext}$$

定时电容在 $10\text{pF}\sim 10\mu\text{F}$ 之间变化， R_{ext} 在 $2\text{k}\Omega\sim 40\text{k}\Omega$ 之间变化。

(b) 器件内部设有补偿电路，脉冲宽度的稳定性高，因为环境温度和 V_{CC} 对脉冲宽度影响较小。

(c) 如果 R_{ext} 使用最大推荐值，占空比(即输出脉宽与重复周期之比 t_W/T)可高达 90%，且工作无颤动。

(d) 有禁止功能。当负跳变触发脉冲从 A_1 或 A_2 端输入时， B 端是电平控制端， B 端为高电平时允许触发， B 端为低电平时禁止触发；当正跳变触发脉冲从 B 端输入时， A_1 和 A_2 端是电平控制端，两者中至少有一端为 0 才允许触发。

(e) B 输入端采用施密特触发输入电路，电路的抗干扰性强。

2. 可重复触发集成单稳态触发器

74122 型可重复触发单稳态触发器是 TTL 电路器件，逻辑框图如图 9.3.5 所示。该组件具有可重复触发功能和异步置 0 功能。 \bar{R}_D 为异步清 0 端； \bar{A}_1 和 \bar{A}_2 是负跳变触发脉冲输入端； B_1 和 B_2 是正跳变触发脉冲输入端； Q 和 \bar{Q} 是互补输出端，稳态

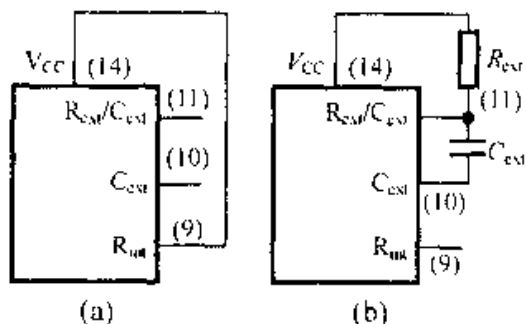


图 9.3.4 74121 连接图

(a) 不增加定时元件

(b) 增加定时元件

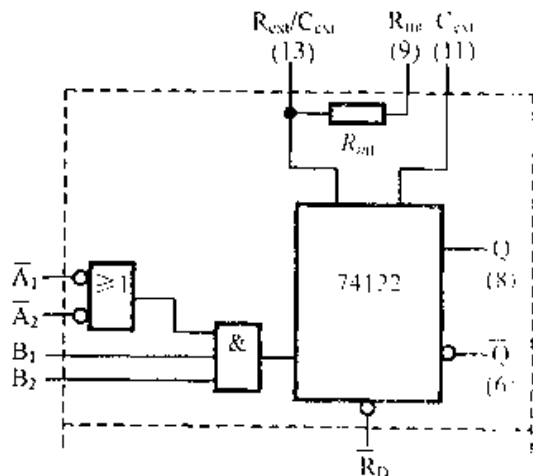


图 9.3.5 74122 逻辑框图

时 Q 为低电平, \bar{Q} 为高电平。在触发脉冲有效时, Q 端输出正脉冲, \bar{Q} 端输出负脉冲。输出脉冲宽度 $t_w \approx 0.7R_{ext}C_{ext}$ (R_{ext} 、 C_{ext} 为外加定时电阻、电容); R_{int} 是内接定时电阻的引出端; C_{ext} 和 R_{ext}/C_{ext} 是外接定时元件的接线端。

74122 的功能表如表 9.3.2 所示。该组件的特点是

(a) 输出脉冲宽度由外接定时元件参数确定。定时电阻 R_{ext} 范围为 $5 \sim 25k\Omega$; 定时电容 C_{ext} 在 $10pF$ 以上。输出脉冲宽度在 $40ns \sim 40s$ 之间变化。

(b) 具有负跳变触发和正跳变触发两种输入端, 并均兼有禁止功能。当负跳变触发脉冲从 \bar{A}_1 或 \bar{A}_2 加入时, B_1 和 B_2 是电平控制端, 它们之中只要有一个为低电平

则禁止触发, 如表 9.3.2 中的第 3 和第 4 行所示; 当正跳变触发脉冲从 B_1 或 B_2 加入时, \bar{A}_1 和 \bar{A}_2 是高电平, 则禁止触发, 如表中第 2 行所示。

(c) 具有异步清 0 功能。若 \bar{R}_0 端有负脉冲出现, 则 Q 端为低电平。在负脉冲结束时, 若 \bar{A}_1 、 \bar{A}_2 、 B_1 和 B_2 未构成触发条件时, 则 Q 端维持低电平; 若它们构成触发条件, 则负脉冲的上跳沿将是有效触发, 所以, 从负脉冲结束之时算起, Q 端输出正常宽度的正脉冲, 如表中最末两行的所示。

(d) 利用重触发功能可以增加脉冲宽度, 见图 9.3.6(a); 利用异步清 0 功能可以

表 9.3.2 74122 的功能表

输 入					输 出	
\bar{R}_0	\bar{A}_1	\bar{A}_2	B_1	B_2	Q	\bar{Q}
0	x	x	x	x	0	1
x	1	1	x	x	0	1
x	x	x	0	x	0	1
x	x	x	x	0	0	1
1	0	x	\downarrow	1	\uparrow	\downarrow
1	0	x	1	\downarrow	\uparrow	\downarrow
1	x	0	\downarrow	1	\uparrow	\downarrow
1	x	0	1	\downarrow	\uparrow	\downarrow
1	1	\downarrow	1	1	\uparrow	\downarrow
1	\downarrow	\downarrow	1	1	\uparrow	\downarrow
1	\downarrow	1	1	1	\uparrow	\downarrow
\downarrow	0	x	1	1	\uparrow	\downarrow
\downarrow	x	0	1	1	\uparrow	\downarrow

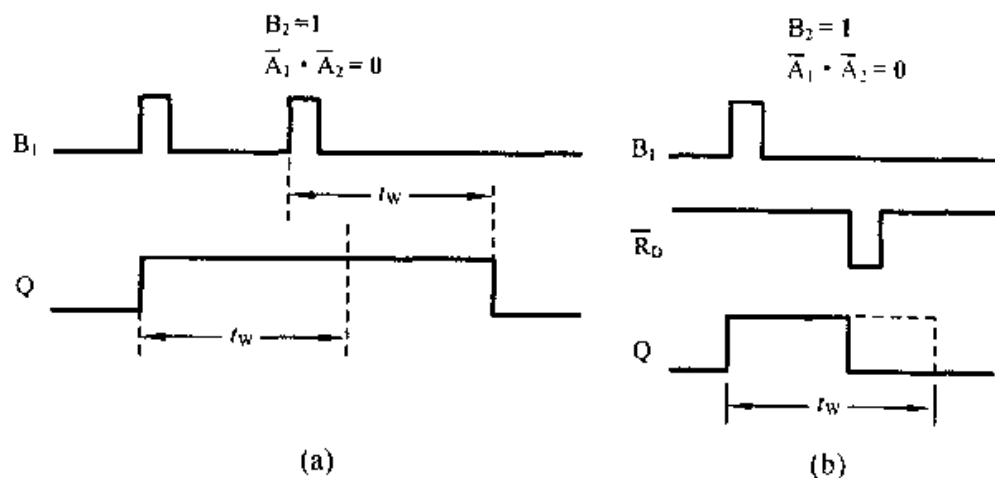


图 9.3.6 输出脉冲宽度控制

(a) 利用重触发扩展脉冲宽度 (b) 利用异步清 0 缩短脉冲宽度

缩短输出脉冲宽度,如图 9.3.6(b)所示。

(e) 器件内部设有补偿电路,脉冲宽度的稳定性高,因为 V_{CC} 和环境温度的变化对脉冲宽度影响较小。

9.3.3 单稳态触发器的应用

单稳态触发器是脉冲电路中应用十分广泛的脉冲单元电路,它的主要应用有脉冲整形、定时、脉冲延时、消除噪声、构成自激多谐振荡器等。

1. 定时

单稳态触发器能产生一定宽度的矩形脉冲,若利用它作为定时信号去控制另一个电路,则可在其 t_w 宽度内有效操作。例如,用矩形脉冲作为与门的控制信号,则只有在 t_w 时间内,信号 v_A 才能通过与门。其电路及波形分别示于图 9.3.7(a)和(b)。

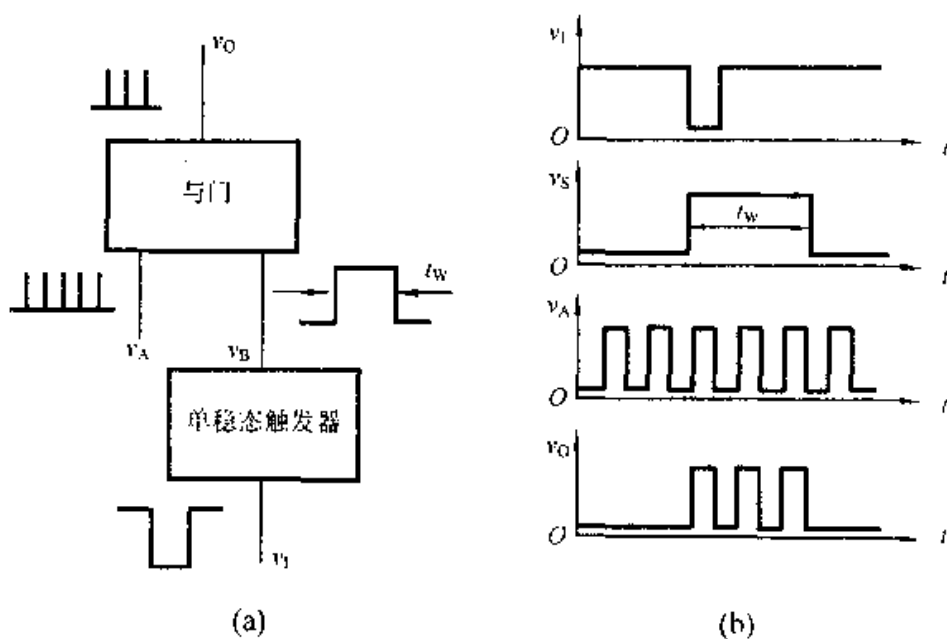


图 9.3.7 单稳态触发器作定时电路的应用

(a) 逻辑图 (b) 波形图

2. 脉冲延时

脉冲延时电路框图如图 9.3.8(a)所示,图(b)所示是波形图。

在单稳电路 I 中,产生脉宽 t_{w1} 的矩形脉冲 v_A ; 在单稳电路 II 中,利用 v_A 的下降沿触发产生脉宽 t_{w2} 的矩形脉冲 v_O 。显然,单稳电路 I 起了延时作用;单稳电路 II 是产生输出脉冲,脉宽为 t_{w2} 。

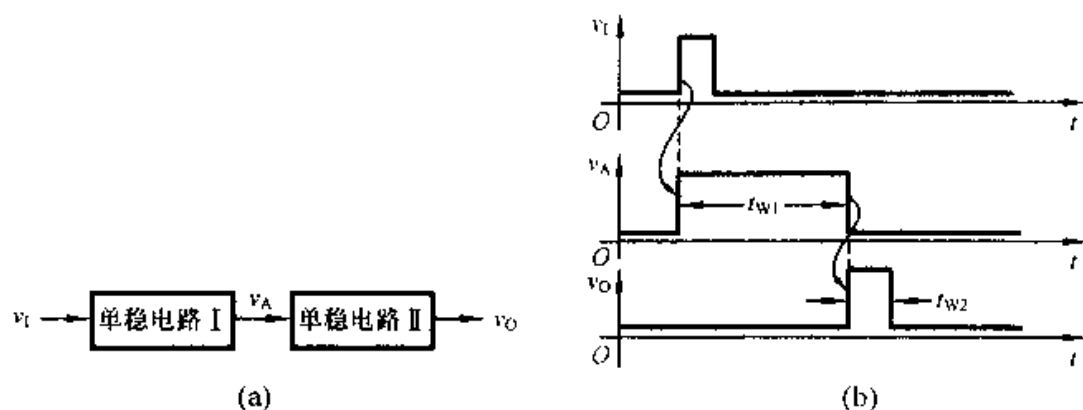


图 9.3.8 脉冲延时电路
(a) 电路框图 (b) 波形图

3. 消除噪声

噪声一般多为尖脉冲，脉宽较窄。利用单稳电路产生脉宽 t_W 的矩形脉冲，并要求 t_W 大于噪声尖脉冲而小于信号脉宽，即可消除噪声。其电路及波形分别示于图 9.3.9(a)和(b)。

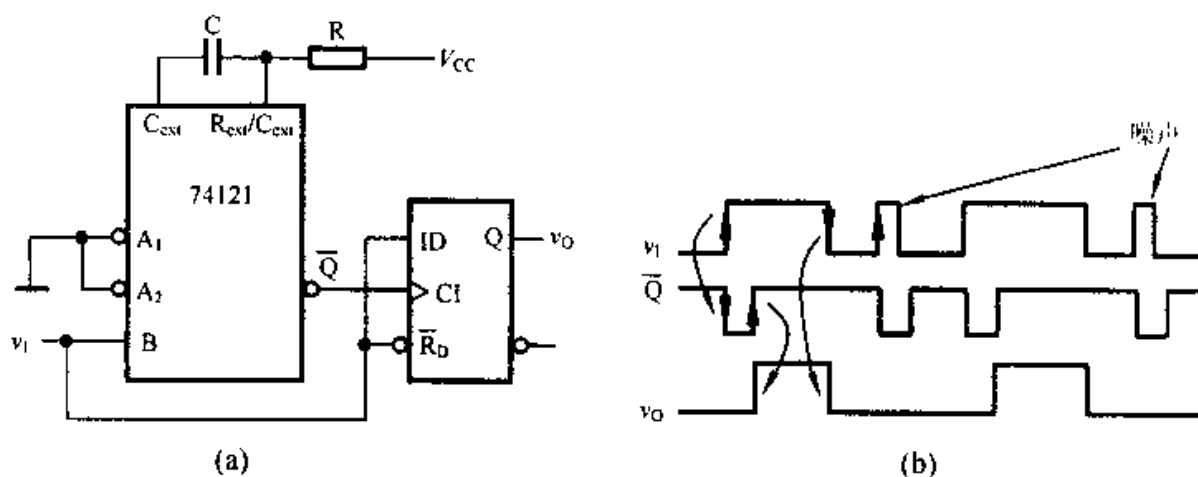


图 9.3.9 噪声消除电路
(a) 逻辑图 (b) 波形图

9.4 多谐振荡器

多谐振荡器是一种自激振荡器，在接通电源之后，无需外加触发信号便能产生方波或矩形脉冲。由于矩形脉冲波或方波中含有丰富的高次谐波，所以称它为多谐振荡器。这种电路没有稳态，只有两个暂稳状态，所以又称为无稳态电路。

9.4.1 集成运放构成的多谐振荡器

集成运放构成的多谐振荡器如图 9.4.1(a)所示。

工作原理 在接通电源的瞬间，输出电压可能偏向于正向饱和，也可能偏向于负向饱和，这是带有偶然性的。假设起始状态为集成运放输出高电平， $v_O = +V_W$ ，此时同相输入端的电位为

$$v_P = +\frac{R_2}{R_1 + R_2} V_W = +FV_W$$

同时 $+V_W$ 通过电阻 R 对电容 C 进行充电，使反相输入端的电压 v_N 以指数规律上升，并趋向于 $+V_W$ 。当 v_N 上升到略超过 $+FV_W$ 时，电路产生正反馈积累过程，使输出电压转换到低电平状态， $v_O = -V_W$ 。同相端的电位 v_P 也相应降到 $-FV_W$ 。

当输出电压突降至 $-V_W$ 时，电容 C 就通过 R 放电，使 v_N 从 $+FV_W$ 开始以指数规律下降，并趋向于 $-V_W$ 值。当 v_N 下降到略低于 $-FV_W$ 时，电路又产生正反馈积累过程，输出电压又转换到高电平状态， $v_O = +V_W$ 。

在 v_O 变成高电平后， C 又通过 R 充电，重复上述过程，就产生自激振荡，输出方波信号。各点波形如图 9.4.1(b)所示。

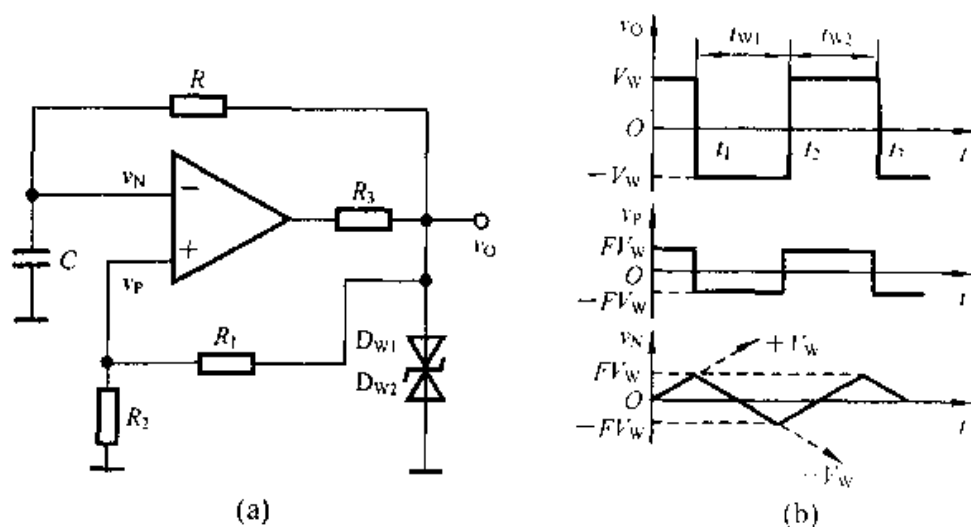


图 9.4.1 集成运放构成的自激多谐振荡器

(a) 电路图 (b) 波形图

工作周期 利用三要素法可以求出两个暂稳态的持续时间 t_{W1} 和 t_{W2} ：

因为 $v_N(0) = +FV_W$, $v_N(\infty) = -V_W$, $v_N(t_{W1}) = -FV_W$

所以 $t_{W1} = RC \ln \frac{-V_W - FV_W}{-V_W + FV_W} = RC \ln \frac{1+F}{1-F}$

因为 $v_N(0) = -FV_W$, $v_N(\infty) = +V_W$, $v_N(t_{W2}) = +FV_W$

所以

$$t_{W2} = RC \ln \frac{V_W + FV_W}{V_W - FV_W} = RC \ln \frac{1+F}{1-F}$$

$$\text{因此, 工作周期 } T = t_{W1} + t_{W2} = 2RC \ln \frac{1+F}{1-F} = 2RC \ln \left(1 + \frac{2R_2}{R_1}\right) \quad (9.4.1)$$

若适当选取 R_1 和 R_2 的值, 使反馈系数 $F=0.47$, 则振荡周期可简化为 $T=2RC$ 。电路特点如下:

(a) 调试方便, 改变 R_2 或 R 都可以很方便地调整电路的振荡频率。

(b) 振荡周期与输出幅度无关, 振荡频率的稳定性主要取决于电容器和稳压管的稳定性, 即使线性组件的性能较差, 其频率漂移也很小。用一般的通用型集成运放, 按图 9.4.1(a) 所示的电路连接, 在 10Hz~10kHz 的低频范围内, 都能产生稳定的方波信号输出。如果选择性能好的高速集成运放, 则振荡频率可以达到 100kHz 以上。

利用集成运放也可以构成不对称的自激多谐振荡器, 电路如图 9.4.2(a) 所示, 其工作原理与图 9.4.1 所示电路的原理完全相同, 只是由于二极管 D_A 、 D_B 的存在, 使电容 C 的充电回路与放电回路分开, 因此, 得到不对称的方波输出。

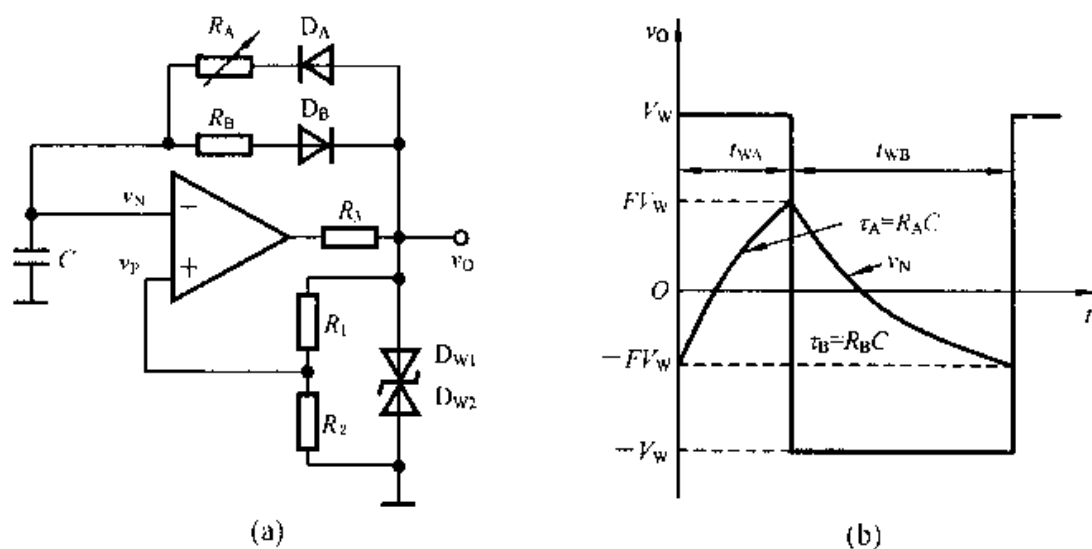


图 9.4.2 不对称的自激多谐振荡器

(a) 电路图 (b) 波形图

9.4.2 施密特触发器构成的多谐振荡器

由施密特触发器构成的多谐振荡器如图 9.4.3(a) 所示。图中, RC 组成积分电路, 将施密特触发器的反相输出端经 RC 积分电路接至输入端便构成多谐振荡器了。

当接通电源之后, 由于电容 C 上初始电压为零, 故输出高电平, 因此, 经 R 向电容 C 充电, 当充到输入电压为 $v_I = V_{TH}$ 时, 输出跳变为低电平。因此, 电容 C 又

经电阻 R 放电, 当放电至 $v_i = V_{TL}$ 时, 输出又跳变至高电平。因此, 电容 C 又开始充电, 如此反复, 形成了多谐振荡。工作波形如图 9.4.3(b) 所示。

施密特触发器若采用 CC40106 型六反相器(见图 9.2.5)组成, 该组件是 CMOS 器件, 设 $V_{OH} \approx V_{DD}$, $V_{OL} \approx 0$, 则依据图 9.4.3(b) 所示的波形可计算振荡周期 T 。

求 t_{W1} :

$$v_C(0) = V_{TL}, \quad v_C(\infty) = V_{DD}, \quad v_C(t_{W1}) = V_{TH}$$

$$t_{W1} = RC \ln \frac{v_C(\infty) - v_C(0)}{v_C(\infty) - v_C(t_{W1})} = RC \ln \frac{V_{DD} - V_{TL}}{V_{DD} - V_{TH}}$$

求 t_{W2} :

$$v_C(0) = V_{TH}, \quad v_C(\infty) = 0, \quad v_C(t_{W2}) = V_{TL}$$

$$t_{W2} = RC \ln \frac{v_C(\infty) - v_C(0)}{v_C(\infty) - v_C(t_{W2})} = RC \ln \frac{-V_{TH}}{-V_{TL}} = RC \ln \frac{V_{TH}}{V_{TL}}$$

振荡器的工作周期

$$\begin{aligned} T = t_{W1} + t_{W2} &= RC \ln \frac{V_{DD} - V_{TH}}{V_{DD} - V_{TL}} + RC \ln \frac{V_{TH}}{V_{TL}} \\ &= RC \ln \left(\frac{V_{DD} - V_{TH}}{V_{DD} - V_{TL}} \times \frac{V_{TH}}{V_{TL}} \right) \end{aligned} \quad (9.4.2)$$

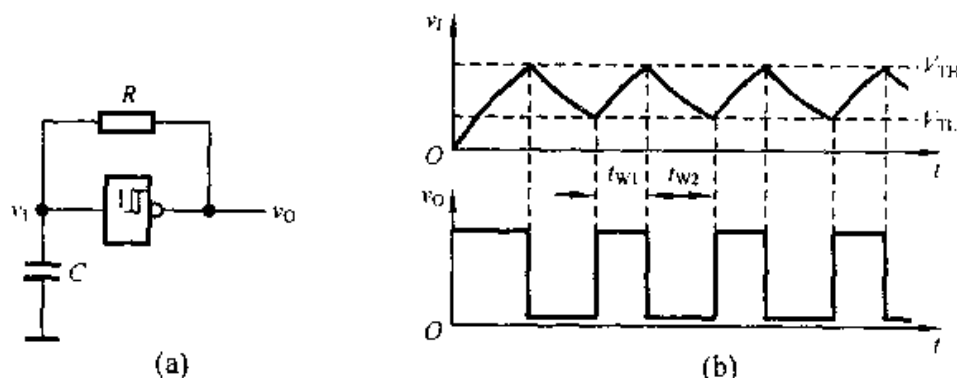


图 9.4.3 用施密特触发器构成的多谐振荡器

(a) 逻辑图 (b) 波形图

9.4.3 单稳态触发器构成的多谐振荡器

用两片 74121 单稳态触发器构成多谐振荡器如图 9.4.4 所示, 图中, S 为起振控制开关。

设电路处于 $Q_1=0$, $Q_2=0$ 时, 将开关 S 断开, 则单稳 I 的 B 端由 0 跳变到 1 触发 (参见表 9.3.1), Q_1 输出正脉冲, 其脉宽为 $0.7R_1C_1$; Q_1 的输出正脉冲下降沿出现时, 单稳 II 的 A_1 端得到触发, Q_2 输出正脉冲。 Q_2 输出正脉冲的下降沿又触发单

稳 I。如此反复便形成多谐振荡，其振荡周期

$$T=0.7(R_1C_1+R_2C_2) \quad (9.4.3)$$

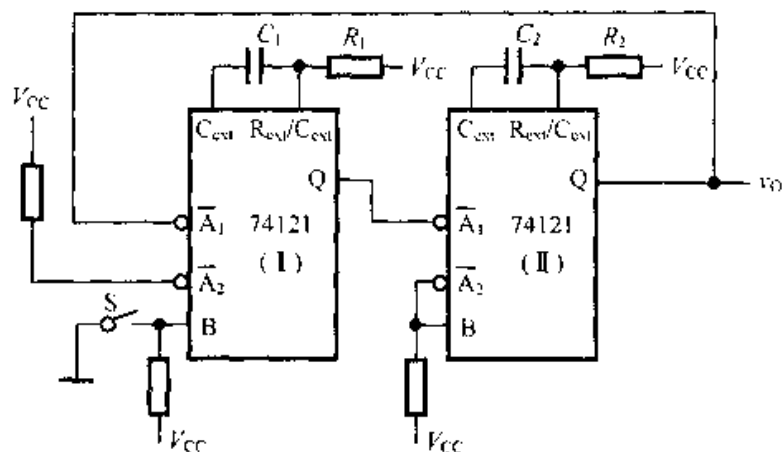


图 9.4.4 由单稳态构成的多谐振荡器

*9.4.4 石英晶体振荡器

在要求多谐振荡器的频率稳定度较高的条件下，可以采用晶体来稳频，图 9.4.5 就是一种石英晶体振荡器电路。门 G_1 和 G_2 构成多谐振荡器，门 G_3 作为整形电路，门 G_2 输出至门 G_1 输入的反馈支路中串接了石英晶体。由于石英晶体有极其稳定的串联谐振频率 f_s ，在此频率的两侧，石英晶体的电抗值迅速增加，所以，若把石英晶体串入两级正反馈电路的反馈支路中，则振荡器只有在 f_s 频率上满足起振条件。因此，这种振荡器的工作频率取决于石英晶体本身的谐振频率，因而这种振荡器的频率稳定度很高，可达 10^{-7} 左右。

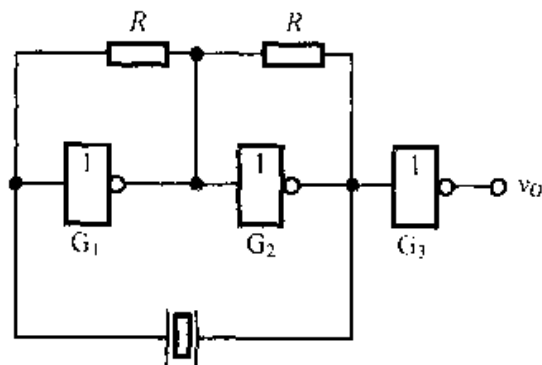


图 9.4.5 晶体振荡器

9.5 555 定时器及其应用

9.5.1 555 定时器的电路结构与功能

555 定时器是一种多用途的数字-模拟混合集成电路，它最大的优点在于只需外接少量的 R、C 元件即可方便地构成单稳态触发器、自激多谐振荡器、施密特触发

器, 广泛应用于波形产生与变换、测量与控制、家电、电玩等领域。

图 9.5.1(a) 是国产双极型定时器 CB555 的电路结构图, 图(b) 是逻辑框图。它由 3 个 $5\text{ k}\Omega$ 的电阻串联分压形成取样电路(产品因此得名), 运放 C_1 、 C_2 构成电压比较器, 基本 RS 触发器和集电极开路的放电管 T_D 3 部分组成。

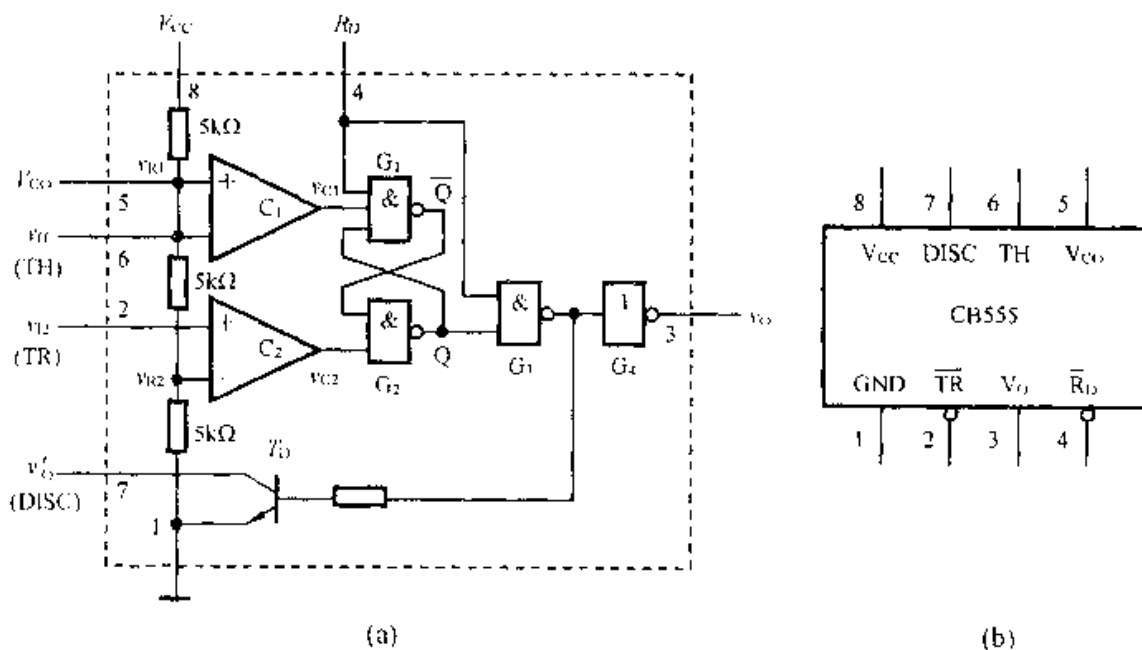


图 9.5.1 CB555 定时器
(a) 电路结构图 (b) 逻辑框图

v_{I1} 是比较器 C_1 的输入端(也称阈值端, 用 TH 标注), v_{I2} 是比较器 C_2 的输入端(也称触发端, 用 TR 标注)。 C_1 和 C_2 的参考电压(电压比较的基准) V_{R1} 和 V_{R2} 由 V_{CC} 经 3 个 $5\text{ k}\Omega$ 电阻分压给出。在控制电压输入端 V_{CO} 悬空时, $V_{R1}=2V_{CC}/3$, $V_{R2}=V_{CC}/3$ 。如果 V_{CO} 外接固定电压, 则 $V_{R1}=V_{CO}$, $V_{R2}=V_{CO}/2$ 。

\bar{R}_D 是置零输入端。只要在 \bar{R}_D 端加上低电平, 输出端 v_O 便立即被置成低电平, 不受其他输入端状态的影响。正常工作时必须使 \bar{R}_D 处于高电平。

由图 9.5.1(a) 可知,

当 $v_{I1} > V_{R1}$ 、 $v_{I2} > V_{R2}$ 时, 比较器 C_1 的输出 $v_{C1}=0$, 比较 C_2 的输出为 1, 基本 RS 触发器被置 0, T_D 导通, 同时 v_O 为低电平。

当 $v_{I1} < V_{R1}$ 、 $v_{I2} > V_{R2}$ 时, $v_{C1}=1$, $v_{C2}=1$, 触发器的状态保持不变, 因而 T_D 和输出的状态也维持不变。

当 $v_{I1} < V_{R1}$ 、 $v_{I2} < V_{R2}$ 时, $v_{C1}=1$, $v_{C2}=0$, 故触发器被置 1, v_O 为高电平, 同时 T_D 截止。

当 $v_{I1} > V_{R1}$ 、 $v_{I2} < V_{R2}$ 时, $v_{C1}=0$, $v_{C2}=0$, 触发器处于 $Q=\bar{Q}=1$ 的状态, v_O 处于高电平, 同时 T_D 截止。

表 9.5.1 是 CB555 的功能表。

为了提高电路的带负载能力, 在输出端还设置了缓冲器 G_4 。如果将 v_O 端经过电阻接到电源上, 则只要这个电阻的阻值足够大, v_O 为高电平时 v'_O 也一定是高电平, v_O 为低电平时 v'_O 也一定是低电平。555 定时器可在很宽的电源电压范围内工作, 并可承受较大的负载电流。双极型 555 定时器的电源电压范围为 5~16V, 最大的负载电流达 200mA, CMOS 型 7555 定时器的电源电压范围为 3~18V, 但最大负载电流在 4mA 以下。

表 9.5.1 CB555 功能表

输 入			输 出	
$\overline{R_D}$	v_{I1}	v_{I2}	v_O	T_D 状态
0	x	x	低	导通
1	$> \frac{2}{3}V_{CC}$	$> \frac{1}{3}V_{CC}$	低	导通
1	$< \frac{2}{3}V_{CC}$	$> \frac{1}{3}V_{CC}$	不变	不变
1	$< \frac{2}{3}V_{CC}$	$< \frac{1}{3}V_{CC}$	高	截止
1	$> \frac{2}{3}V_{CC}$	$< \frac{1}{3}V_{CC}$	高	截止

9.5.2 用 555 定时器构成施密特触发器

根据 555 定时器的电路结构, 如果使 v_{C1} 和 v_{C2} 的低电平信号发生在输入电压信号的不同电平, 那么, 输出与输入之间的关系将构成施密特触发器。因此, 可将 555 定时器的 v_{I1} (6 脚) 与 v_{I2} (2 脚) 连在一起形成 v_I , 在 V_{CO} (5 脚) 端接 $0.01 \mu F$ 到地, 以提高比较电压的稳定性, 如此即构成施密特触发器如图 9.5.2 所示。

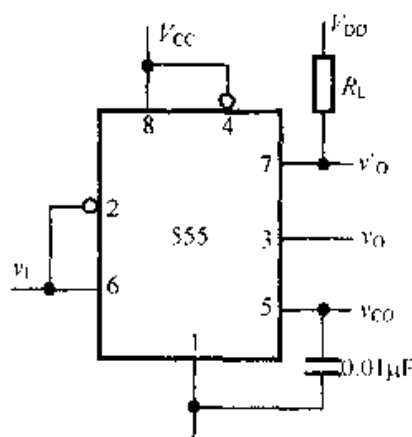


图 9.5.2 用 555 构成施密特触发器

工作原理 当 $v_I < V_{CC}/3$ 时, $v_{C1}=1$, $v_{C2}=0$, $Q=1$, 故 $v_O=V_{OH}$;

当 $V_{CC}/3 < v_I < 2V_{CC}/3$ 时, $v_{C1}=v_{C2}=1$, 故 $v_O=V_{OH}$ 保持不变;

当 $v_I > 2V_{CC}/3$ 以后, $v_{C1}=0$, $v_{C2}=1$, $Q=0$, 故 $v_O=V_{OL}$, 发生一次翻转。因此, $V_{T+}=2V_{CC}/3$, 称为上限触发电平。此过程是 v_I 从 0 逐渐升高的过程。

当 $V_{CC}/3 < v_I < 2V_{CC}/3$ 时, $v_{C1}=v_{C2}=1$, 故 $v_O=V_{OL}$ 保持不变。

当 $v_I < V_{CC}/3$ 以后, $v_{C1}=1$, $v_{C2}=0$, $Q=1$, 故 $v_O=V_{OH}$, 又发生一次翻转。因此, $V_{T-}=V_{CC}/3$, 称为下限触发电平。此过程是 v_I 从高于 $2V_{CC}/3$ 开始逐渐下降的过程。由此得到电路的回差电压

$$\Delta V_T = V_{T+} - V_{T-} = \frac{1}{3}V_{CC}$$

图 9.5.3 是施密特触发器的电压传输特性, 它是一个典型的反相输出施密特触发特性。如果参考电压由外接的电压 V_{CO} 提供, 则 $V_{TH}=V_{CO}$, $V_{TL}=V_{CO}/2$,

$\Delta V = V_{CO}/2$ 。通过改变 V_{CO} 的值可以调节回差电压。

图 9.5.4 所示的就是一个将不规则的模拟信号波形反相转换成规则的矩形波的实例。

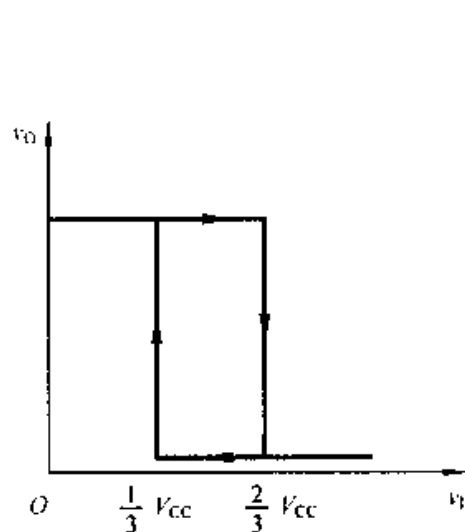


图 9.5.3 施密特触发器的电压传输特性

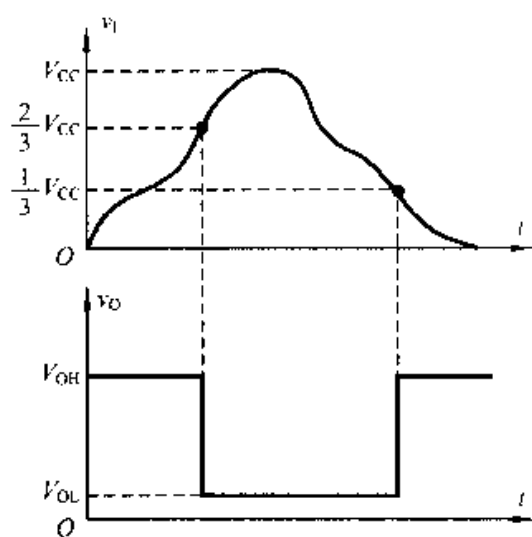


图 9.5.4 施密特触发器的应用

9.5.3 用 555 定时器构成单稳态触发器

根据 555 定时器的结构，如果将 v_{i2} (2 脚) 作为 v_i 端加一个低电平触发信号。同时将内部放电管的集电极 c 极 (7 脚) 与外接电阻 R 连接并引入到 v_{i1} 端 (6 脚)，①③脚之间的电容保留，即构成单稳态触发器。如图 9.5.5 所示。

如果没有触发信号时 v_i 处于高电平，那么稳态时这个电路一定处于 $v_{C1}=v_{C2}=1$ ， $Q=0$ ， $v_O=0$ 的状态。假定接通电源后触发器停在 $Q=0$ 的状态，则 T_D 导通 $v_C \approx 0$ ，故 $v_{C1}=v_{C2}=1$ ， $Q=0$ 及 $v_O=0$ 的状态将稳定的维持不变。

如果接通电源后触发器停在 $Q=1$ 的状态，这时 T_D 一定截止， V_{CC} 便经 R 向 C 充电。当充到 $v_C=2V_{CC}/3$ 时， v_{C1} 变为 0，于是将触发器置 0。同时 T_D 导通，电容 C 经 T_D 迅速放电，使 $v_C \approx 0$ 。此后由于 $v_{C1}=v_{C2}=1$ ，触发器保持 0 状态不变，输出也相应稳定在 $v_O=0$ 的状态。

因此，通电后电路自动地停在 $v_O=0$ 的稳态。

当触发脉冲的下降沿到达时，使 v_O 跳变到 $V_{CC}/3$ 以下时，使 $v_{C2}=0$ (此时 $v_{C1}=1$)，触发器被置 1， v_O 跳变为高电平，电路进入暂稳态。此时 T_D 截止， V_{CC} 经过 R 开始向电容 C 充电。

当充至 $v_C=2V_{CC}/3$ 时， v_{C1} 变成 0。如果此时输入端的触发脉冲已消失， v_i 回到了高电平。触发器将被置 0，于是输出返回到 $v_O=0$ 的状态。同时 T_D 又变为导通状态，电容 C 经 T_D 迅速放电，直至 $v_C \approx 0$ ，电路恢复到稳态。图 9.5.6 所示的是在触

发信号作用下 v_C 和 v_O 的相应波形。

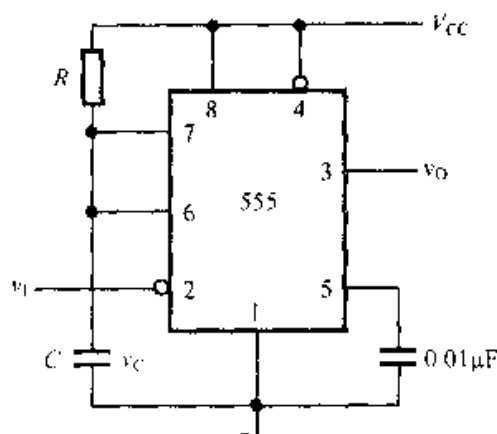


图 9.5.5 用 555 定时器构成的单稳态触发器

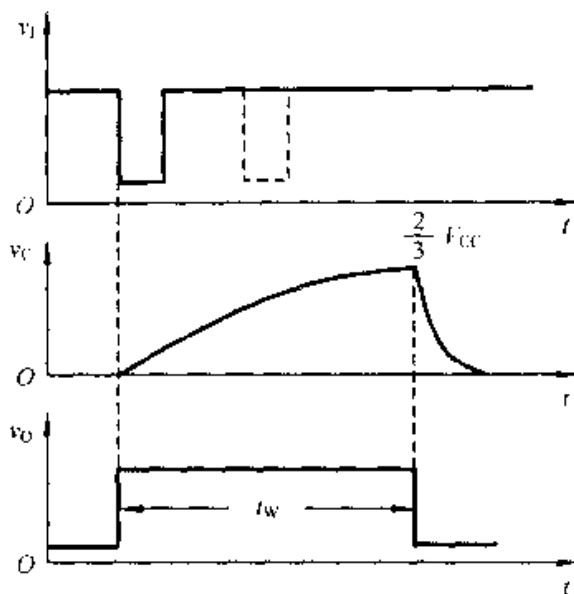


图 9.5.6 触发信号作用下的电压波形图

输出脉冲的宽度 t_W 等于暂稳态的持续时间,而暂稳态的持续时间取决于外接电阻 R 和电容 C 的大小。由图 9.5.6 可知, t_W 等于电容电压在充电过程中从 0 上升到 $2V_{CC}/3$ 所需要的时间,因此得到

$$t_W = RC \ln \frac{V_{CC} - 0}{V_{CC} - \frac{2}{3}V_{CC}} = RC \ln 3 = 1.1RC \quad (9.5.1)$$

通常 R 的取值在几百欧姆到几兆欧姆之间,电容的取值范围为几百皮法到几百微法, t_W 的范围为几微秒到几分钟。但必须注意,随着 t_W 的宽度的增加,它的精度和稳定度也将下降。

9.5.4 用 555 定时器构成自激多谐振荡器

根据 555 定时器的结构,将 v_{I1} 和 v_{I2} 连在一起接成施密特触发器,然后再将输出信号经 RC 积分电路接回到输入端即可。(T_D 与外接电阻 R_1 接成反相器,其集电极电平与 v_O 完全相同)图 9.5.7(a)中就是从 7 脚经 R_2 、 C 引到输入端 2、6 脚的。

在电路接通时,由于电容 C 还未充电,所以 v_C 为低电平,比较器 C_1 输出 v_{C1} 为高电平,比较器 C_2 的输出 v_{C2} 为低电平,门 G_3 输出为低电平,电路输出 v_O 为高电平。此时放电管 T_D 截止, V_{CC} 通过电阻 R_1 和 R_2 对电容 C 充电,电路进入暂稳态。

在暂稳态期间,随着电容 C 的充电, v_C 电位不断升高,当 $v_C \geq 2V_{CC}/3$ 时,比较器 C_1 输出 v_{C1} 为低电平,使门 G_3 输出为高电平,使得 v_O 翻转为低电平,电路发生一次翻转。同时 T_D 在门 G_3 为高电平时导通,电容 C 通过 R_2 、 T_D 放电,电路进

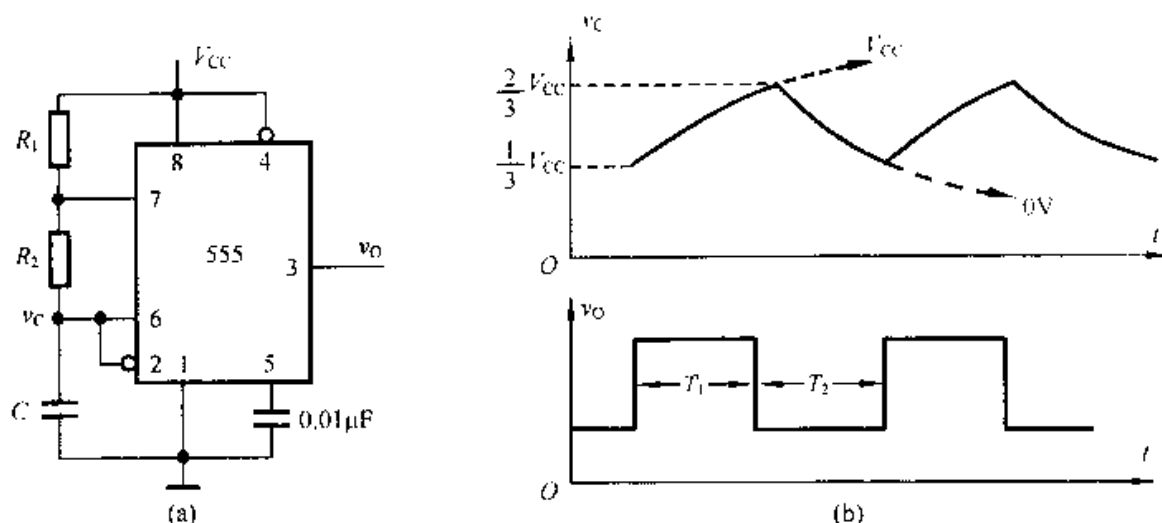


图 9.5.7 用 555 定时器构成的多谐振荡器

(a) 电路结构 (b) 工作波形

入另一暂稳态。在此期间，随着电容 C 的放电，使 v_C 电位逐步下降。当 v_C 下降至 $v_C \leq V_{CC}/3$ 时，比较器 C_2 输出 v_{C2} 为低电平，使门 G_3 输出为低电平，电路又一次自动发生翻转。

由于门 G_3 输出低电平， T_D 截止，电源 V_{CC} 又通过 R_1 和 R_2 对电容 C 充电，如此周而复始，形成多谐振荡，工作波形如图 9.5.7(b) 所示。

由分析可知，在电容充电时，暂稳态持续时间

$$T_{W1} = 0.7(R_1 + R_2)C \quad (9.5.2)$$

在电容 C 放电时，暂稳态持续时间

$$T_{W2} = 0.7R_2C \quad (9.5.3)$$

因此，电路输出矩形脉冲的周期

$$T = T_{W1} + T_{W2} = 0.7(R_1 + 2R_2)C \quad (9.5.4)$$

输出矩形脉冲的占空比

$$q = \frac{t_{W1}}{T} = \frac{R_1 + R_2}{R_1 + 2R_2} \quad (9.5.5)$$

例 9.5.1 试用 CB555 定时器设计一个多谐振荡器，要求振荡周期为 1s，输出脉冲幅度大于 3V 而小于 5V，输出脉冲的占空比 $q = 2/3$ 。

解 由 CB555 的特性参数可知，当电源电压取 5V 时，在 100mA 的输出电流下输出电压的典型值为 3.3V，所以，取 $V_{CC} = 5V$ 可以满足对输出脉冲幅度的要求。若采用图 9.5.7(a) 所示电路，根据式(9.5.5)可知

$$q = \frac{R_1 + R_2}{R_1 + 2R_2} = \frac{2}{3}$$

故得到 $R_1 = R_2$ 。

又由式(9.5.4)知

$$T=0.7(R_1+2R_2)C=1$$

若取 $C=10\mu\text{F}$ ，则代入上式解出

$$R_1=R_2=\frac{1}{3\times 0.7\times 10\times 10^{-6}}=48\text{k}\Omega$$

因 $R_1=R_2$ ，所以取两只 $47\text{k}\Omega$ 的电阻与一个 $2\text{k}\Omega$ 的电位器串联，即得到图 9.5.8 所示的设计电路。

在由 555 定时器构成的多谐振荡器中，若定时器的电压控制端 5 脚不经电容接地，而是外加一个可变的电压源，则通过调节该电压源的值，可以改变定时器触发电位和阈值电位的大小。外加电压越大，振荡器输出脉冲周期越大，即频率越低；外加电压越小，振荡器输出脉冲周期越小，即频率越高。这样，多谐振荡器就实现了将输入电压大小转换成输出频率高低的电压-频率转换器的功能。

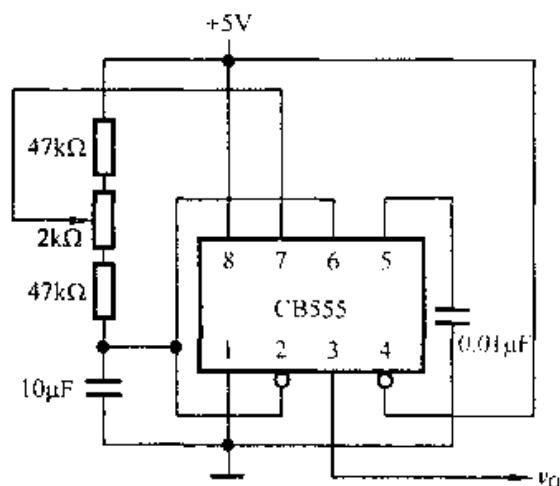


图 9.5.8 例 9.5.1 多谐振荡器的电路图

小 结

脉冲单元电路是产生或变换脉冲波形的电路，这些电路都是由开关元件和惰性网络两部分组成的。开关元件的主要功能是接通或断开电路，使之产生暂态过程；惰性网络的功能是控制暂态过程的形状和快慢。

施密特触发器和单稳态触发器是最常用的两种整形电路。前者的输出高、低电平随输入信号的电平变化，所以，输出脉冲宽度由输入信号决定；后者的输出脉冲宽度完全取决于电路参数，输入信号只起触发作用。另一类脉冲信号产生电路是自激多谐振荡器，它无需外加输入信号，只要接通电源就可以产生矩形脉冲。

555 定时器是一种数-模混合集成芯片，它可以很方便地构成各种脉冲单元电路，还可以接成各种实用电路，是一种用途很广的集成芯片；由集成门构成的脉冲单元电路具有外接元件少、接线简单、速度快以及逻辑电平与其他逻辑电路兼容等优点。由集成运放构成的脉冲单元电路具有稳定性好、精度高等优点，其缺点是工作速度较低，同其他逻辑电路的电平匹配需加电平转换电路。

目前，脉冲单元电路的发展趋势也是集成化，其单片集成的产品已经很多，并且电路类型也逐年增加，所以，要尽可能选用单片集成的脉冲单元电路，使产品的设计与开发周期缩短，产品的市场竞争力提高。

思考题和习题

- 9.1 试述施密特触发器的工作特点和主要应用。
- 9.2 试述单稳态触发器的工作特点和主要用途。
- 9.3 试述多谐振荡器的工作特点和主要用途。
- 9.4 555 定时器具哪些应用特点？其典型应用有哪几种？
- 9.5 为什么单稳态触发器经外界触发后，不能长期稳定在另一种状态？它的“暂稳状态”是怎样维持的？
- 9.6 施密特触发电路如图 P9.1 所示。若 $|V_Z| = 4\text{V}$, $A_V = 5000$, 且 $V_{PH} = 6.5\text{V}$, $V_{PL} = 5.5\text{V}$ 。试求：(1) R_2/R_1 之值；(2) 环路增益 A_{VF} ；(3) V_{RO} 。
- 9.7 由集成运放构成的多谐振荡器如图 P9.2 所示，试估算输出电压 v_O 的矩形波周期。

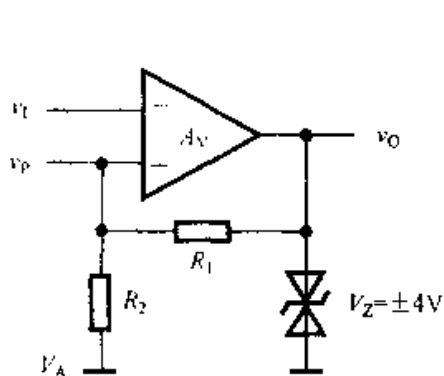


图 P9.1

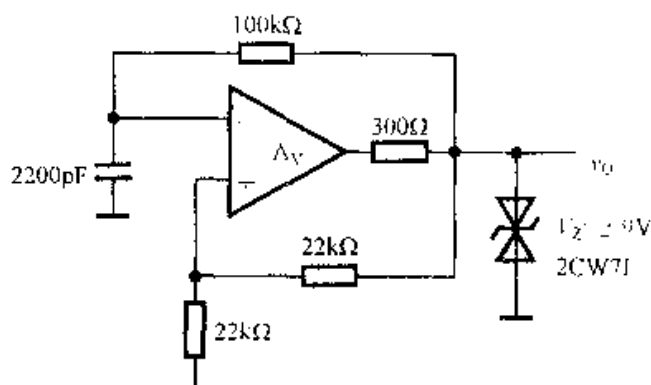


图 P9.2

- 9.8 施密特触发电路如图 P9.3 所示。设： $R_1 = 10\text{k}\Omega$, $R_2 = 100\Omega$, 开环增益 $A_V = 5000$, $V_R = 1\text{V}$; $V_{OH} = 5\text{V}$, $V_{OL} \approx -5\text{V}$ 。
- (1) 求反馈系数 F ；(2) 求环路增益 A_{VF} ；(3) 求回差电压 ΔV ，并画出电压传输特性。
- 9.9 由集成单稳态触发器 74LS121 构成的脉冲产生电路如图 P9.4 所示。试计算输出脉冲宽度的可调范围。

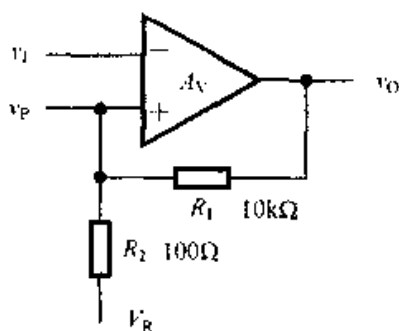


图 P9.3

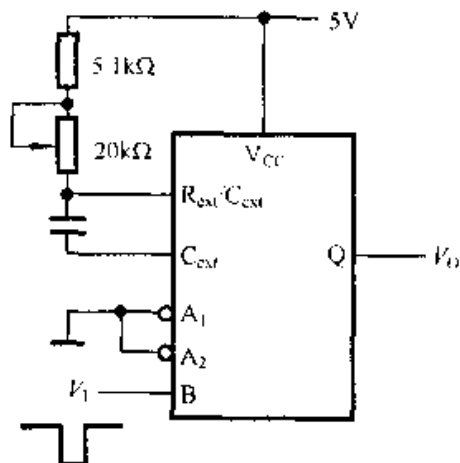


图 P9.4

- 9.10 由 74LS121 构成消除尖脉冲干扰(噪声)的电路如图 P9.5(a)所示(参见图 9.3.9)。试说明它消除尖脉冲干扰的工作原理;假设输入波形如图 P9.5(b)所示,请画出 \bar{Q} 和 V_O 端的工作波形图;若尖脉冲与输入脉冲 V_i 的下跳沿间隔为 $30\mu\text{s}$, 输入脉冲低电平为 $100\mu\text{s}$, 试估算 R 的取值范围。

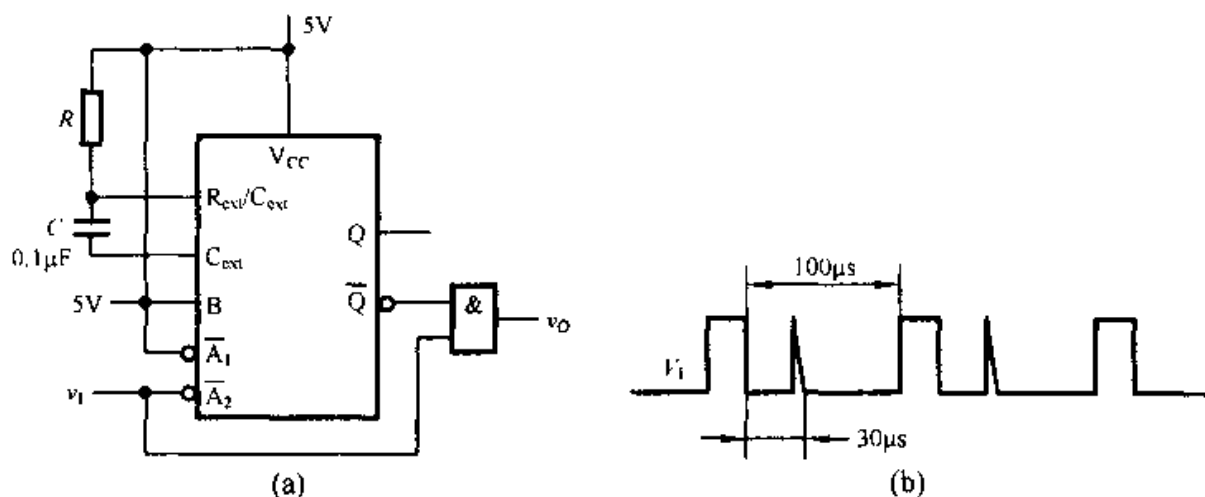


图 P9.5

- 9.11 由施密特触发器构成的多谐振荡器如图 9.4.3 所示, 施密特触发器采用 CMOS 电路, 并设 $V_{DD}=10\text{V}$, $V_{TH}=6.3\text{V}$, $V_{TL}=2.7\text{V}$, $R=10\text{k}\Omega$, $C=0.01\mu\text{F}$, 试求该电路的振荡周期。
- 9.12 图 P9.6 是由 555 定时器构成的施密特触发器, 若电源 $V_{CC}\approx 9\text{V}$, 则在不考虑 V_{CO} 影响的情况下, 其正、负向阈值电压 V_+ 、 V_- 及回差电压 ΔV 各为何值?
- 9.13 在图 P9.7 所示的由 555 定时器构成的单稳态触发器中, R_1 、 C_1 构成什么环节, 起什么作用? 在什么情况下可以省略该环节?

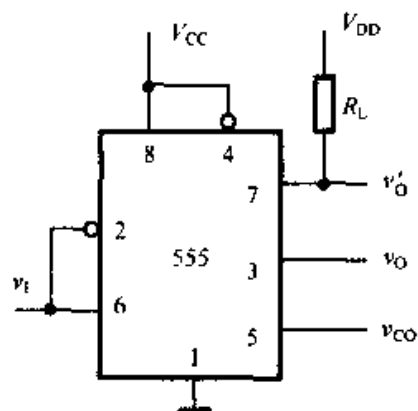


图 P9.6

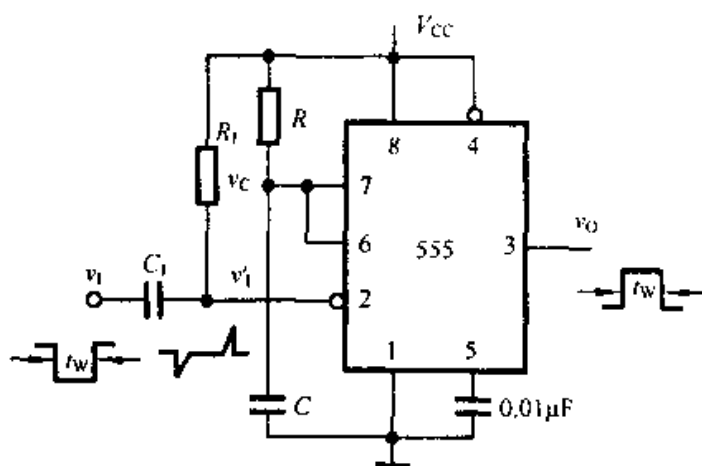


图 P9.7

- 9.14 某电路如图 P9.8 所示, 设外接二极管为理想二极管, 试求占空比 q 的变化范围及工作频率 f , 并画出 v_C 、 v_O 的对应波形。

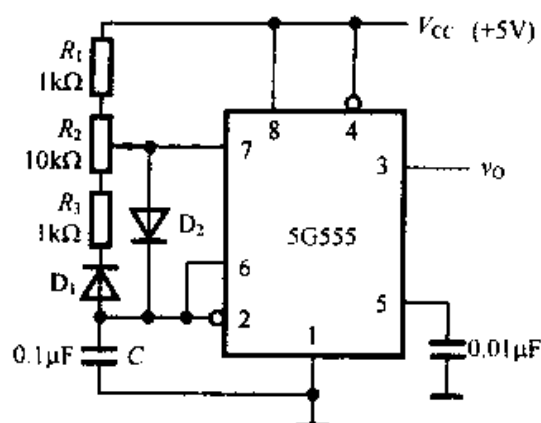


图 P9.8

- 9.15 分析如图 P9.9 所示电路, 简述其电路的组成及工作原理。若要求扬声器在开关 S 按下后, 以 1.2kHz 频率持续响 10s, 试确定图中 R_1 、 R_2 的值。

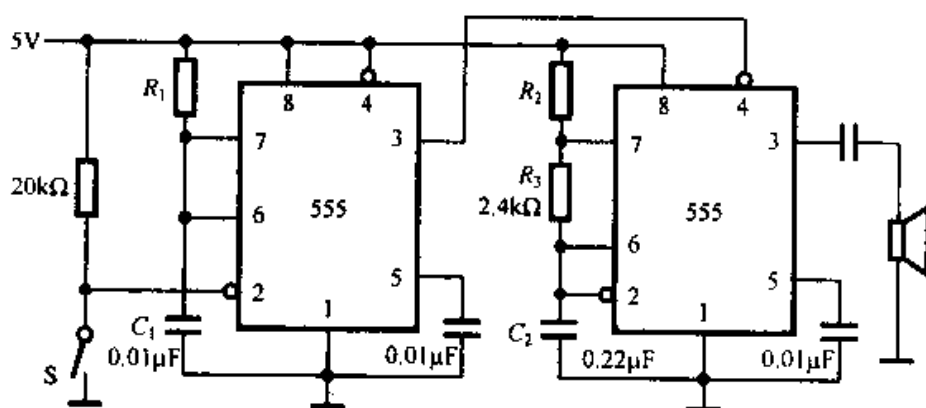


图 P9.9

第 10 章 模数及数模转换技术

内容提要 本章讨论模数及数模转换的基本原理，重点讨论常用的几种模数转换器及数模转换器，同时介绍通用性较强的几种单片集成模数转换器及数模转换器的电路结构、工作原理和使用中应注意的问题。

10.1 模数及数模转换技术概述

数字系统已广泛应用于各种学科领域，乃至国民经济生活中的方方面面，微型计算机就是一个典型的数字系统。但是，数字系统只能对输入的数字信号进行处理，其输出也是数字信号。在工业和生活中的许多物理量都是连续变化的模拟量，如温度、压力、流量、速度等。这些模拟量可以通过传感器或换能器变成与之对应的电压、电流或频率等电模拟量。为了实现数字系统对这些电模拟量进行检测、运算和控制，就需要一个模拟量与数字量之间相互转换的过程。即要求把模拟量转换为数字量，以便数字系统进行工作；数字系统的输出数字量也需要转换为模拟量才便于应用。我们把模拟量转换为数字量的过程称为模数转换，完成这种转换的电路称为模数转换器 (Analog to Digital Converter, 简称 ADC)；把数字量转换为模拟量的过程称为数模转换，完成这种转换的电路称为数模转换器 (Digital to Analog Converter, 简称 DAC)。图 10.1.1 是一个典型数字控制系统的框图。

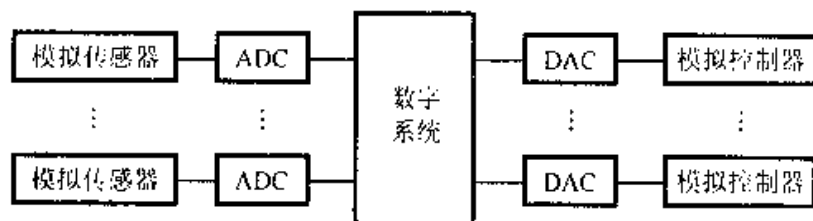


图 10.1.1 数字控制系统框图

模拟传感器的输入为模拟量，其输出为模拟电压或电流，它们作为 ADC 的输入，ADC 输出为相应的数字量。这些数字量经过数字系统处理后输出数字量，再经 DAC 变换为模拟量送到模拟控制器以实现对各种物理量的控制。

如果一个系统只包含有计算机、DAC 和模拟控制器，则通常称它为程序控制系统；如果只包含模拟传感器、ADC 和计算机，则通常称它为数据处理系统。

模数及数模转换技术是数字技术的一个重要分支。目前，已生产了单片集成的和具有不同性能指标的系列产品可供用户选用，为了适应数字系统发展的需要，提

高 ADC 和 DAC 芯片的转换速度和转换精度已成为今后发展的主要方向。

10.2 数模转换器

10.2.1 数模转换器的基本原理

数字量是用按数位组合的一组代码来表示的, 或者用按权展开式来表示的。每一位代码都有一定的“权”, 所以, 为了将数字量转换为模拟量, 必须将每一位代码按“权”值转换成相应模拟量, 然后将代表各位的模拟量相加, 这样便得到与数字量成正比的模拟量。这种转换器通常是将各位代码同时转换, 故称为并行 DAC。它的转换速度快, 一般仅为几十纳秒。

设输入数字量为 D , 输出模拟量为 A , 则 DAC 的一般关系式为

$$A=kD$$

式中, k 为比例系数, 它是一个常数; D 为 n 位二进制数, 即有 $D=\sum_{i=0}^{n-1} a_i 2^i$ 。

DAC 输入数字量与输出模拟量的对应关系称为 DAC 的转换特性。图 10.2.1 所示是单极性(不带符号位)4 位二进制 DAC 的转换特性。图中, 二进制代码 1111 对应的输出电压称为满度电压 v_{Omax} , 且令 $v_{Omax}=15V$, 则 4 位二进制 DAC 有 16 个输出的模拟电压, 即从 0V 变化到 15V。由图可见: 输出模拟量可以分为 (2^4-1) 个阶梯等级, 当最大输出电压确定之后, 输入数字量的位数愈多, 则输出模拟量的阶梯间隔愈小, 即相邻两组代码转换出来的模拟量之差值愈小, 就表示转换器的分辨率愈高。

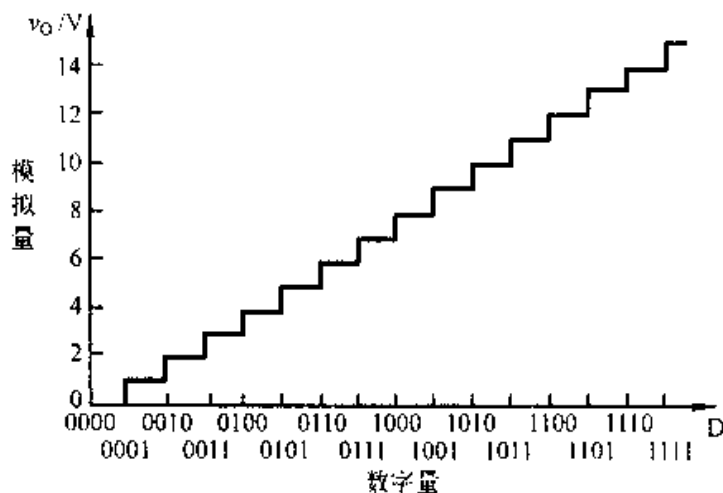


图 10.2.1 DAC 输出特性

图 10.2.2 是双极性 4 位二进制 DAC 的转换特性。输入数字信号的最高位为符号位, 0 表示正值, 1 表示负值。图中, 假设输出电压范围为 $-7V \sim +7V$, 输入数字信号为原码。

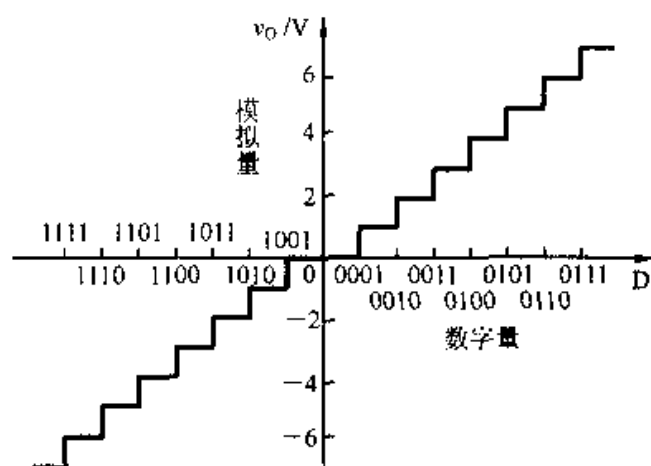


图 10.2.2 双极性 DAC 输出特性

图 10.2.3 是一个 n 位 DAC 框图。输入数字信号先存放在寄存器内，寄存器并行输出的各位数码驱动一个模拟开关；通过模拟开关将基准电压按位切换到电阻解码网络，使输出模拟电压与该位数码所具有的“权”值相对应。

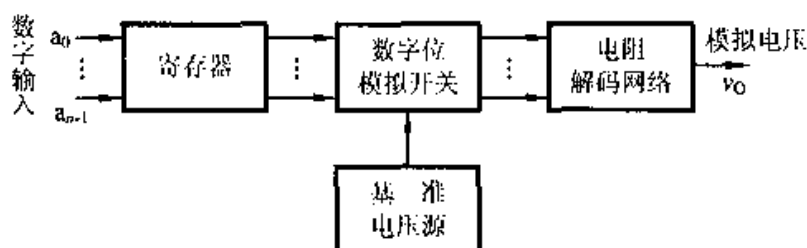


图 10.2.3 DAC 框图

DAC 分为电压型和电流型两大类。电压型 DAC 有权电阻网络、T 型电阻网络和树形开关网络等；电流型 DAC 有权电流型电阻网络和倒 T 型电阻网络等。

10.2.2 电压型数模转换器

1. 权电阻网络 DAC

权电阻网络 DAC 电路如图 10.2.4 所示。它由寄存器、模拟电子开关、权电阻解码网络、运算放大器和基准电源组成。

(a) 寄存器：在寄存指令作用下，将输入数字量 $a_{n-1} \sim a_0$ 存入寄存器，一直要存储到下一个指令到来为止。

(b) 模拟电子开关：寄存器输出的数码控制相应的模拟电子开关，即 a_i 控制 $S_i (i=0 \sim n-1)$ 。当 a_i 为 1 时， S_i 接至基准电压 V_R ；当 a_i 为 0 时， S_i 接地。

(c) 权电阻解码网络：对于 n 位二进制代码，则需要 n 个电阻来组成该网络。网络中各支路的电阻值 $R_0 \sim R_{n-1}$ 必须按二进制的位权大小成比例减小，即 a_i 位相应

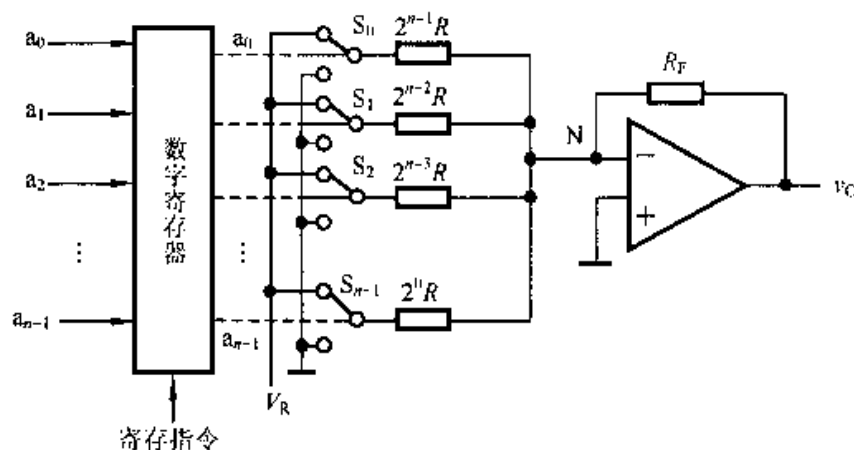


图 10.2.4 权电阻网络 DAC 电路

支路的电阻为 $R_i = 2^{n-1-i} R$ 。例如，最高位 a_{n-1} 相应支路的电阻值最小，其值为 $2^0 R$ ，所以，在 a_{n-1} 为 1 时， S_{n-1} 便接至基准电压 V_R ，该支路就给出最大电流。显然可见：各支路中电流按 $2^i (i=0, \dots, n-1)$ 规律变化，位权值愈高，支路中电流愈大。

(d) 运算放大器：作为权电阻网络的负载，将该网络输出的总电流转换为相应的模拟电压输出。调节转换系数可以通过改变反馈电阻 R_F 的大小来实现。

下面对输出模拟电压 v_O 进行定量分析。

设输入数字量

$$D = a_{n-1} \cdots a_0 = \sum_{i=0}^{n-1} a_i 2^i \quad (10.2.1)$$

当输入数码中某一位 a_i 为 1 时， S_i 接基准电压 V_R ，所以，流向求和点 N 的电流

$$I_i = \frac{V_R}{R_i} = \frac{V_R}{2^{n-1-i} R} = \frac{V_R}{2^{n-1} R} 2^i \quad (10.2.2)$$

当输入数码 a_i 为 0 时， S_i 接地，所以 $I_i = 0$ 。由此可将 a_i 位产生的电流写成

$$I_i = \frac{V_R}{2^{n-1} R} a_i 2^i \quad (10.2.3)$$

根据叠加原理，流向求和点 N 的总电流

$$I = \sum_{i=0}^{n-1} I_i = \sum_{i=0}^{n-1} \frac{V_R}{2^{n-1} R} a_i 2^i = \frac{V_R}{2^{n-1} R} \sum_{i=0}^{n-1} a_i 2^i \quad (10.2.4)$$

通过运算放大器，输出电压 $v_O = -IR_F = \frac{-R_F}{R} \times \frac{V_R}{2^{n-1}} \sum_{i=0}^{n-1} a_i 2^i$ ，若选择 $R_F = R$ ，则

$$v_O = -\frac{V_R}{2^{n-1}} \sum_{i=0}^{n-1} a_i 2^i \quad (10.2.5)$$

上式表明输出模拟电压的大小与输入的数字量成正比，即完成数模转换功能。

权电阻网络型 DAC 的优点是简单、直观；主要缺点是电阻品种多且阻值范围太宽。例如，一个 10 位 DAC，若最高位电阻为 $2\text{k}\Omega$ ，则最低位电阻为 $2^9 \times 2\text{k}\Omega = 1\text{M}\Omega$ ，在

如此宽的范围内要保证各种阻值的精度以及相近的温度系数是十分困难的, 所以, 这类 DAC 的精度难以保证。

2. T 型电阻网络 DAC

在 T 型电阻网络 DAC 电路中, R - $2R$ T 型电阻网络 DAC 是最常见的一种, 如图 10.2.5 所示。

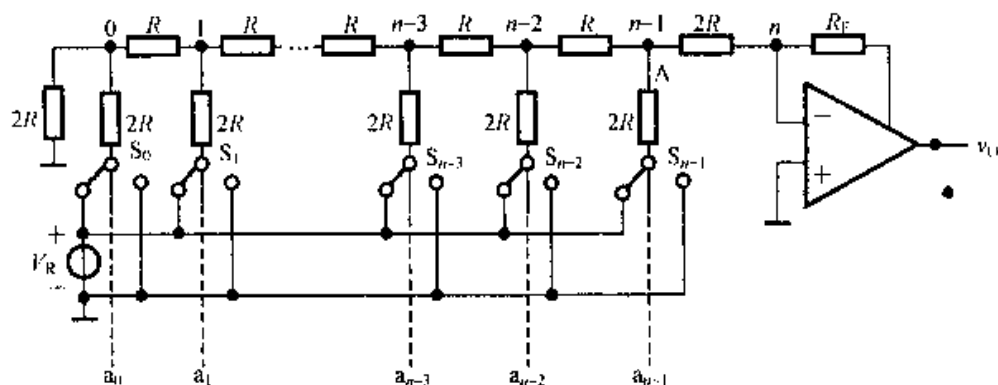


图 10.2.5 R - $2R$ T 型电阻网络 DAC 电路

T 型电阻网络的特点是网络中任何一个节点向 3 个支路方向看的对地等效电阻均为 $2R$ 。例如, 节点 $(n-1)$, 向右看对地的等效电阻为 $2R$ (一端为虚地); 向下看对地的等效电阻也是 $2R$; 向左看进去的等效电阻不能直观看出来, 但这是一个简单的电阻并串联问题, 读者不难从节点 0 开始, 画出节点 1 的等效电路, 并依此类推, 逐个节点向右推进, 即可发现节点 $(n-1)$ 向左看去的等效电阻还是 $2R$ 。利用 R - $2R$ T 型电阻网络的这个特点很容易求出任何一位数码 $a_i=1$ 时, 其对应节点的电压 $v_i=V_R/3$; 该节点电压每向求和点 N 传递一个节点就衰减一半。例如 $a_0=1$, 则节点 0 的电压和传递至节点 1 的等效电路如图 10.2.6 所示。所以, 任意一节点 i 的电压传递到 $(n-1)$ 节点的电压为

$$a_i \times \left(\frac{V_R}{3} \right) \times \left(\frac{1}{2} \right)^{n-1-i}$$

依据叠加原理, 节点 $(n-1)$ 的总电压

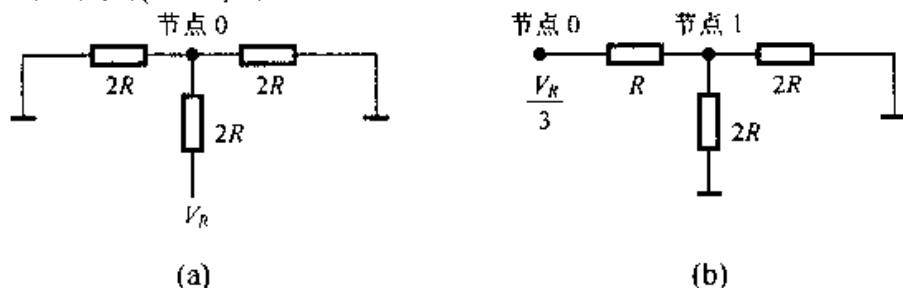


图 10.2.6 $a_0=1$ 时等效电路

(a) 节点 0 的等效电路 (b) 节点 1 的等效电路

$$V_A = \sum_{i=0}^{n-1} a_i \times \frac{V_R}{3} \times \left(\frac{1}{2}\right)^{n-1-i} = \left(\frac{V_R}{3}\right) \left(\frac{1}{2}\right)^{n-1} \sum_{i=0}^{n-1} a_i 2^i \quad (10.2.6)$$

求和运算放大器的输出为

$$V_O = -\frac{R_F}{2R} V_A = -\frac{R_F}{3R} \times \frac{V_R}{2^n} \sum_{i=0}^{n-1} a_i 2^i \quad (10.2.7)$$

式中, 选择 $R_F=3R$, 则有

$$V_O = -\frac{V_R}{2^n} \sum_{i=0}^{n-1} a_i 2^i \quad (10.2.8)$$

式(10.2.7)和式(10.2.8)都表明转换器的输出模拟电压的大小与输入数字量成正比, 即实现数模转换功能。

T型电阻网络需要较多电阻, 但品种只有两种(R 、 $2R$), 制作时容易保证精度和相近的温度系数, 有利于提高转换精度。

3. 树形开关网络 DAC

MOS 管作为开关使用, 关断性能好而且功耗低, 因此, 可用 MOS 管构成树状开关网络来实现数模转换, 如图 10.2.7 所示。图中开关由 MOS 管构成, 它们的工作状态由输入的数码控制; 当 a_i 为 1 时, 对应 a_i 的开关接通; 当 a_i 为 0 时, 对应 \bar{a}_i 的开关接通。由图可得到

$$V_O = V_R \left(\frac{1}{2} a_2 + \frac{1}{4} a_1 + \frac{1}{8} a_0 \right) = \frac{V_R}{2^n} \sum_{i=0}^2 a_i 2^i$$

由此推广到 n 位树型开关网络 DAC, 可知其输出电压

$$V_O = \frac{V_R}{2^n} \sum_{i=0}^{n-1} a_i 2^i \quad (10.2.9)$$

式 10.2.9 表明输出模拟电压与输入数字量成正比, 完成数模转换功能。

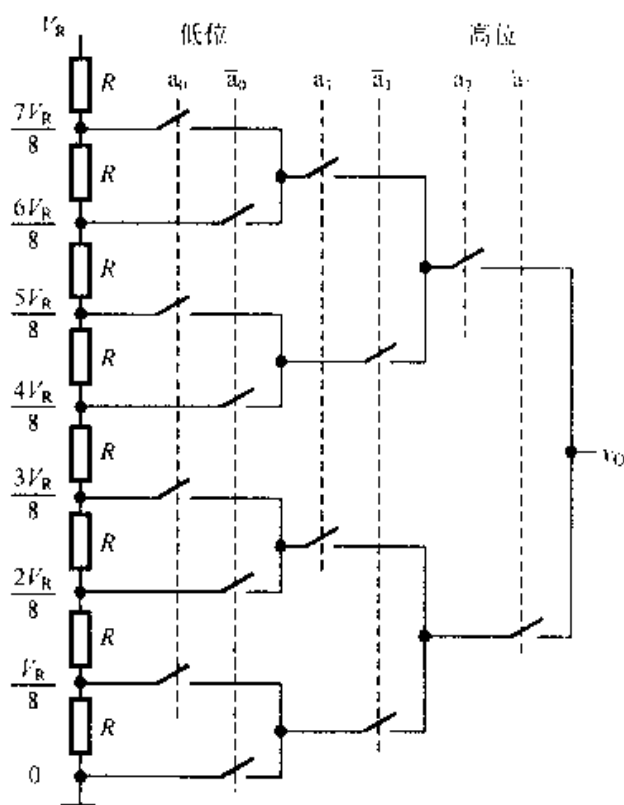


图 10.2.7 树形开关网络 DAC

*10.2.3 电流型数模转换器

电压型 DAC 的缺点是转换速度低, 这是由于电子开关工作于饱和或截止状态, 切换速度慢、信号摆幅大, 因而分布电容充放电时间较长。电流型 DAC 克服了这

个缺点，它的转换速度高。电流型 DAC 有倒 T 型电阻网络 DAC 和权电流型 DAC 等类型。

图 10.2.8 是电流型倒 T 型电阻网络 DAC 的电路图。它的特点是电流开关直接与运算放大器的求和端 N 连接，由于运算放大器的输入端为虚地。所以，无论开关置于什么位置，开关端点的电压近似不变，因而也就不存在分布电容充放电的问题，这就是电流型倒 T 型网络能提高转换速度的原因。

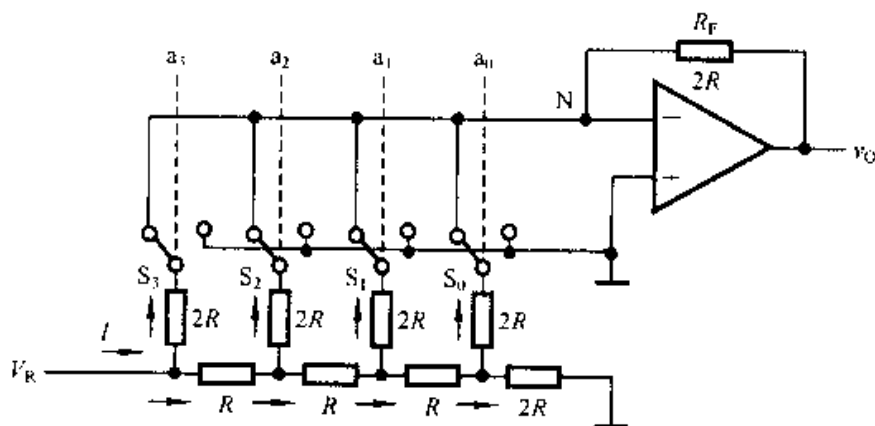


图 10.2.8 电流型倒 T 型电阻网络

由于运算放大器的 N 点为虚地，故无论开关状态如何，各 $2R$ 电阻上端均相当于接地，而从任一节点向右看的对地等效电阻均为 $2R$ 。所以，从电源 V_R 流出的电流为

$$I = \frac{V_R}{R}$$

从 i 节点到 $i-1$ 节点的电流减少一半，即流入开关 S_3 、 S_2 、 S_1 、 S_0 的电流分别为 $I/2$ 、 $I/4$ 、 $I/8$ 、 $I/16$ 。所以，流入求和点 N 的电流

$$\begin{aligned} I_N &= a_3 \frac{I}{2} + a_2 \frac{I}{4} + a_1 \frac{I}{8} + a_0 \frac{I}{16} \\ &= \frac{1}{2^4} \times \frac{V_R}{R} (a_3 \times 2^3 + a_2 \times 2^2 + a_1 \times 2^1 + a_0 \times 2^0) \\ &= \frac{1}{2^4} \times \frac{V_R}{R} \sum_{i=0}^3 a_i 2^i \end{aligned} \quad (10.2.10)$$

运算放大器输出模拟电压(令 $R_F=R$)

$$v_O = -I_N R_F = -\frac{V_R}{2^4} \sum_{i=0}^3 a_i 2^i \quad (10.2.11)$$

由此可推广到 n 位电流型倒 T 型电阻网络 DAC，其输出

$$v_O = -\frac{V_R}{2^n} \sum_{i=0}^{n-1} a_i 2^i \quad (10.2.12)$$

10.2.4 中规模集成数模转换器

由于集成技术的发展,目前,生产的单片集成 DAC 内不仅有电流开关、偏置电路,而且还包含了 T 型电阻网络及基准电源控制电路等。品种繁多,各具特色。这里仅介绍几种通用性较强的芯片,而且重点是讨论它们的外部特性和使用方法,而不过多地去讨论芯片内部的电路结构。

1. AD7541 D/A 转换器

AD7541 DAC 电路原理图如图 10.2.9(a)所示。芯片内含有倒 T 型电阻网络和 12 个 CMOS 模拟开关,输入为 12 位二进制码。该转换器内部不包含运算放大器,外接一个运算放大器便构成了一个完整的 DAC,它们的连接方法如图 10.2.9(b)所示。

AD741 是运算放大器,其输出电压幅度等于 $\pm 10\text{V}$; R_1 为倒 T 型网络的补偿电阻,用以调节网络的输出电流; R_2 是反馈电阻 R_F (片内)的补偿电阻,用于补偿 R_F 阻值的偏差。

由图 10.2.9(a)可知:

$$R=10\text{k}\Omega \quad R_F=10\text{k}\Omega=R$$

$$I_N = \frac{1}{2^{12}} \times \frac{V_R}{R} \sum_{i=0}^{11} a_i 2^i$$

因此
$$v_O = -I_N R_F = -\left(\frac{1}{2^{12}} \times \frac{V_R}{R} \sum_{i=0}^{11} a_i 2^i \right) R = \frac{-V_R}{2^{12}} \sum_{i=0}^{11} a_i 2^i \quad (10.2.13)$$

例如, $V_R = -10\text{V}$, $a_0 \sim a_{11}$ 均为 1, 即可得满度输出电压

$$v_{O\max} = -\left(\frac{1}{2^{12}} \right) (-10\text{V}) (2^{11} + 2^{10} + \dots + 2^0) = \frac{10\text{V}}{2^{12}} \times (2^{12} - 1) = 9.99975\text{V}$$

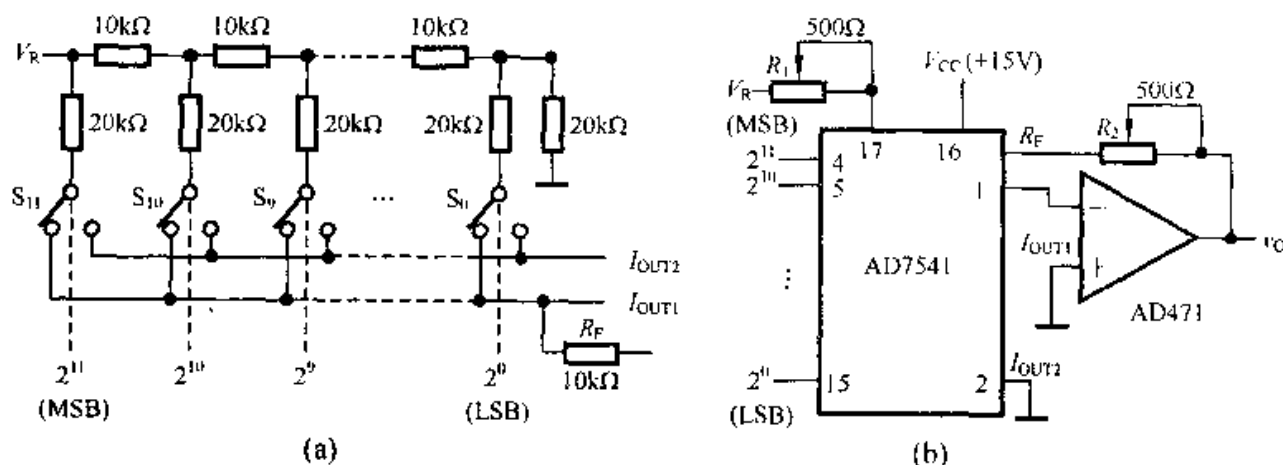


图 10.2.9 AD7541 型 DAC

(a) 电路图 (b) 连接图

2. DAC 0832 D/A 转换器

有一类 DAC 带有与微机连接的接口, 以便于直接与微机的 CPU 总线连接而无须附加逻辑电路。在带有微机或单片机的数字系统中应用十分广泛, 典型产品是 DAC0832, 其他还有 AD7224、AD7225 等。

DAC0832 逻辑框图如图 10.2.10 所示。它包含有两级缓冲寄存器以及 DAC。在使用时应外接运算放大器。主要信号说明如下。

- \overline{CS} : 片选信号, 低电平有效。
- \overline{WR}_1 : 写使能信号, 允许数据存入输入锁存器。低电平有效。
- \overline{WR}_2 : 写使能信号, 允许输入锁存器的内容存入数据寄存器, 低电平有效。
- \overline{LE} : 输入锁存器的使能信号, 高电平有效。
- I_{OUT1} 和 I_{OUT2} : DAC 的输出电流。
- \overline{XFER} : 传输控制信号, 低电平有效。
- V_{REF} : 基准电压, 范围 $-10V \sim +10V$ 。

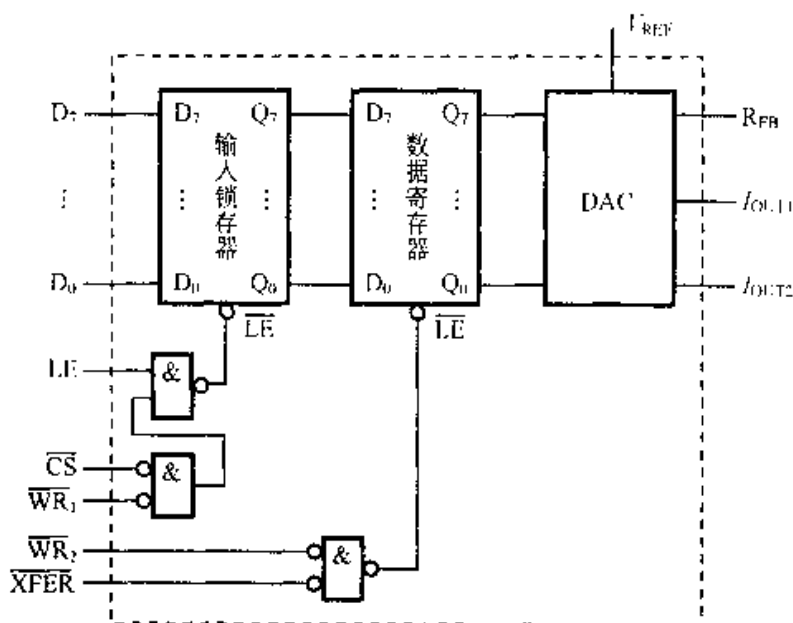


图 10.2.10 DAC0832 逻辑框图

芯片主要操作说明如下: 在片选 \overline{CS} 、锁存使能 \overline{LE} 、写使能 \overline{WR}_1 同时有效时, 数据总线上的数据 $D_0 \sim D_7$ 存入输入锁存器; 当写使能 \overline{WR}_2 和传输控制 \overline{XFER} 有效时, 输入锁存器的内容便存入数据寄存器。D/A 转换开始, 大约经过 $1\mu s$ 之后在输出端便建立了稳定电流输出。

DAC0832 可以构成 3 种工作方式: 单级缓冲、双级缓冲、直接驱动。图 10.2.11 是 3 种工作方式的连接图。图(a)为单级缓冲方式, 数据寄存器处于直通状态, 输入锁存器处于受控状态; 图(b)为双级缓冲方式, 两级寄存器都处于受控状态, 所以输入数据要经过两级缓冲才能实现 D/A 转换; 图(c)为直接驱动方式, 两级寄存器均处

于直通状态，只要有数据输入便可进行 D/A 转换。

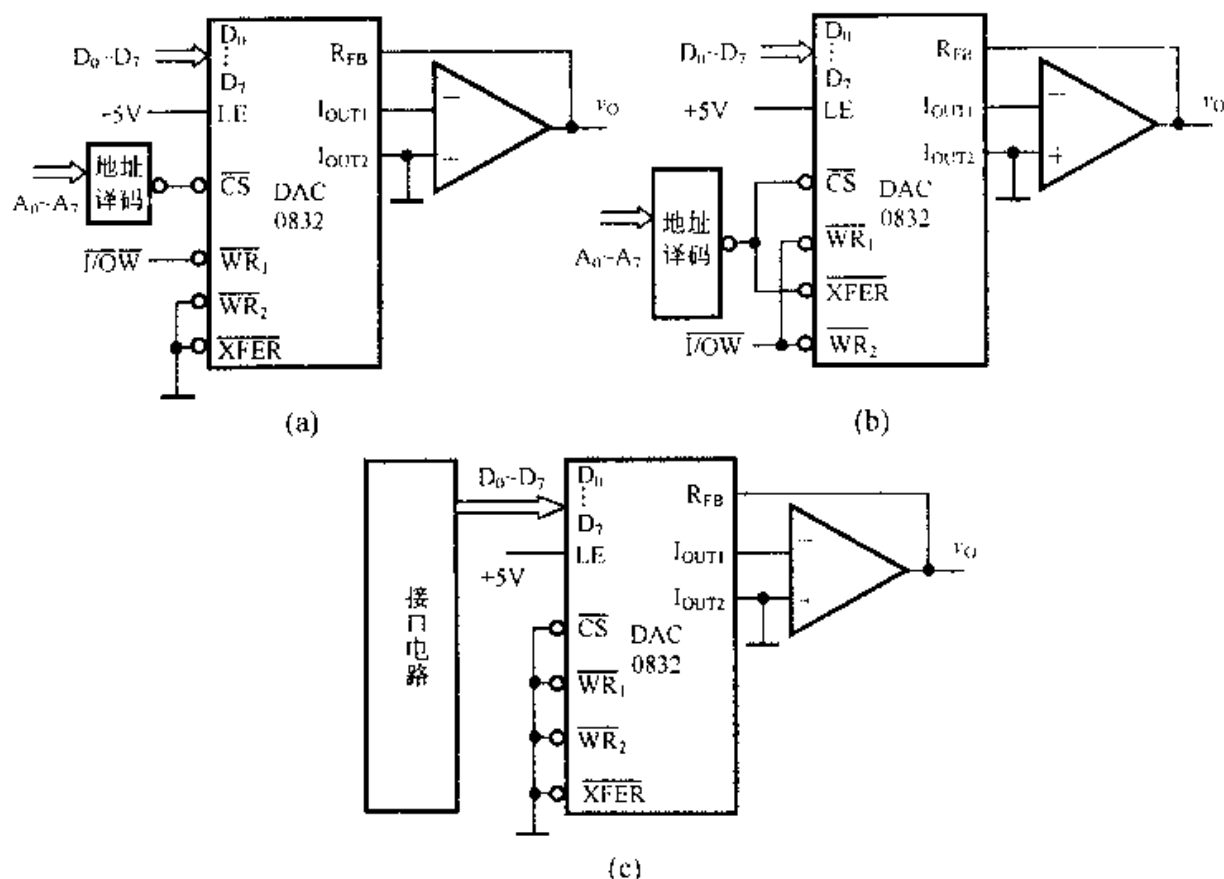


图 10.2.11 DAC0832 的工作方式

(a) 单级缓冲方式 (b) 双级缓冲方式 (c) 直接驱动方式

***例 10.2.1** 用 DAC0832 设计一个阶梯波发生器。要求该阶梯波在一定的时间范围内每隔一段时间，输出幅度递增一个恒定值。如图 10.2.12 所示波形中，每隔 2ms 输出幅度增长 15LSB，经 20ms 后重新循环。

解 如图 10.2.13 所示，将 DAC0832 工作在单缓冲方式下，#BFFFH 是 DAC0832

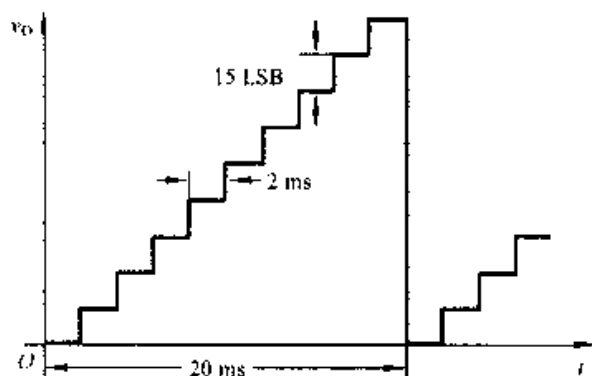


图 10.2.12 阶梯波波形图

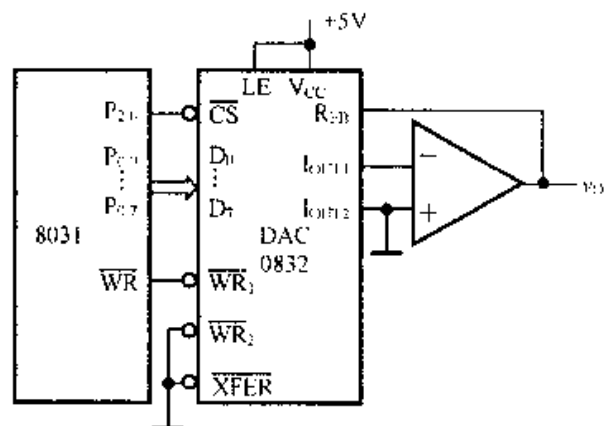


图 10.2.13 阶梯波信号发生器接线图

作为 8031 单片机外围芯片的端口地址。8031 单片机在执行 $\text{MOVX}@\text{DPTR}, \text{A}$ 时, P2.6 和 $\overline{\text{WR}}$ 变低, 使 DAC0832 有效工作, 存在累加器 A 中的阶梯波幅度数 (8 位数字信号) 通过 8031 单片机 P_0 口送入 DAC0832 的 $\text{D}_0\sim\text{D}_7$ 端, DAC0832 及附加电路完成 D/A 转换后输出模拟电压 V_0 。所需的 2ms 延迟通过延迟程序 DELAY_2 获得。

产生阶梯波的延迟程序如下:

```
START: MOV A, #00H;
        MOV DPTR, #BFFFH ; D/A 转换器地址送地址寄存器 DPTR
        MOV R1, #0AH      ; 台阶数为 10
LOOP:   MOVX @DPTR, A      ; 送数据至 D/A 转换器
        CALL DELAY_2       ; 2ms 延迟
        DJNZ R1, NEXT1     ; 不到 10 个台阶转移
        SJMP START         ; 产生下一个周期
NEXT1:  ADD A, #15          ; 台阶增幅
        SJMP LOOP          ; 产生下一个台阶
DELAY_2: MOV 20H, #498     ; 开始 2ms 延迟程序
AGAIN:  NOP
        NOP
        DJNZ 20H, AGAIN
        RET
```

*3. AD558D/A 转换器

AD558 是美国模拟器件公司生产的 8 位电压输出 D/A 转换器, 片内含有输出运算放大器、与微处理器完全兼容的接口和高精度参考电压源, 因此, 不需外接元件和调整即可与 MCS-51 单片机直接接口。AD558 这些齐全的功能使其应用比 DAC0830 系列更简便。

(1) AD558 的技术特性

(a) 分辨率: 8 位, 数据输入及控制逻辑与微处理器总线完全兼容, 故能直接与 8 位或 16 位微处理器接口。

(b) 输出形式: 电压输出 ($0\sim+2.56\text{V}$ 和 $0\sim+10\text{V}$ 两种标定范围)。

(c) 内有精密能隙电压基准, 为确保精度, 厂家已在出厂时标定好芯片, 不需用户再调。用户只要选择连接方式, 就可确定输出电压的量程范围。

(d) 工作电源: 单一电源电压 $+5\sim+15\text{V}$ 。

(e) 电压建立时间: 高速输出运算放大器使电压输出在 800ns 内建立到 $\pm 1/2\text{LSB}$ (量程为 $0\sim+2.56\text{V}$), 在 $2\mu\text{s}$ 内建立到 $\pm 1/2\text{LSB}$ (量程为 $0\sim+10\text{V}$)。

(f) 低功耗: $V_{\text{CC}}=5\text{V}$ 时, 典型值为 75mW ; $V_{\text{CC}}=15\text{V}$ 时, 典型值为 225mW 。

AD558 的电路结构图、逻辑框图和引脚图分别如图 10.2.14(a)、(b)和(c)所示。

其中, 控制逻辑和与单片机接口的电路是由 I^2L (Integrated Injection Logic) 工艺制造, 因而具有高密度、低功耗和高速等优良特性。

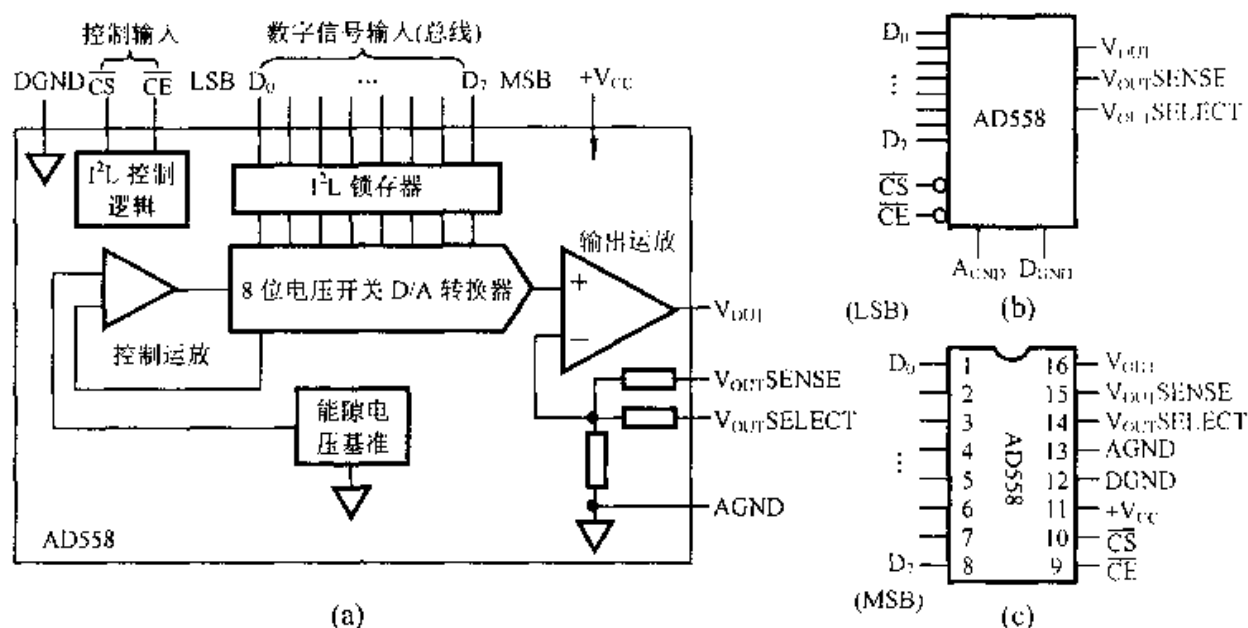


图 10.2.14 AD558 D/A 转换器

(a) 电路结构图 (b) 逻辑框图 (c) 引脚图

AD558 主要引脚功能说明如下:

- $D_0 \sim D_7$: 8 位数据输入。
- \overline{CE} : 芯片允许信号, 低电平有效。
- \overline{CS} : 片选信号, 低电平有效。
- V_{CC} : 供电电源, $+5 \sim +15V$ 。
- DGND: 数字地。
- AGND: 模拟地。

➤ $V_{OUTSELECT}$: 输出量程选择端, 如果该引脚和 AGND 相连, 则输出电压为 $0 \sim +10V$ 。如果它和引脚 15 ($V_{OUTSENSE}$) 相连, 则输出电压为 $0 \sim +2.56V$ 。量程转换连接图如图 10.2.15(a)和(b)所示。

➤ $V_{OUTSENSE}$: 输出运放检测引脚。特殊用途时可接补偿电阻以提供大的驱动电流。

➤ V_{OUT} : 运放电压输出引脚。

(2) AD558 与 8031 单片机连接方法

AD558 与 8031 单片机的硬件接口电路如图 10.2.16 所示。

图中, 引脚 \overline{CE} 与 8031 的 \overline{WR} 线直接相连, 片选 \overline{CS} 通过译码得到。引脚 14 与引脚 15 直接相连, 选择的电压输出量程为 $0 \sim +2.56V$ 。

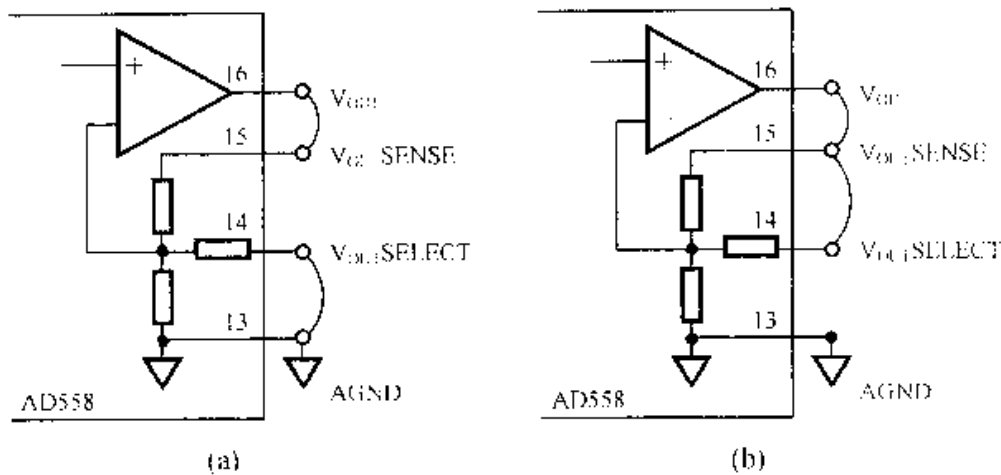


图 10.2.15 输出量程选择连接图

(a) 输出电压范围 0V~+10V (b) 输出电压范围 0V~+2.56V

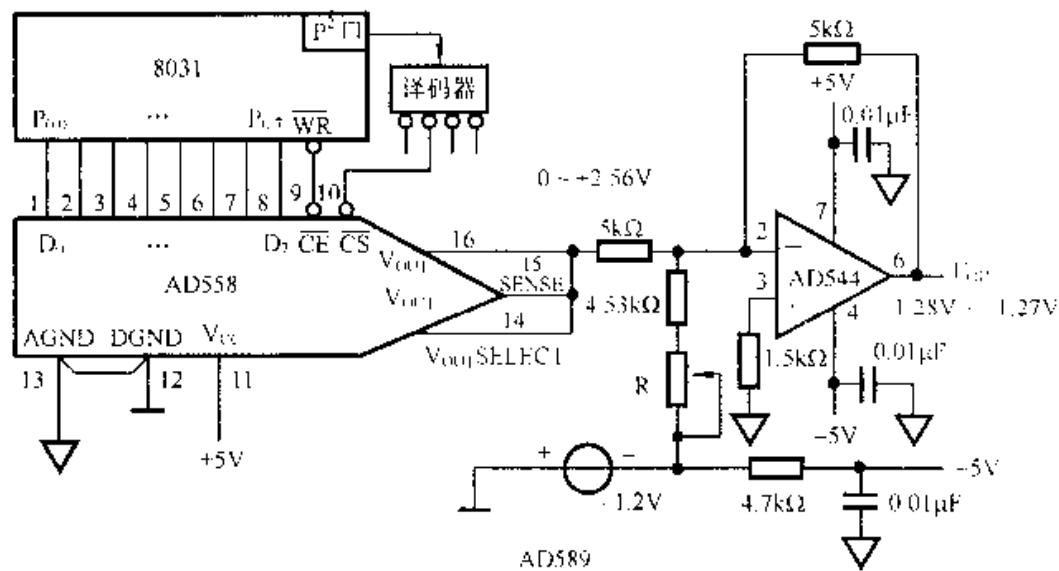


图 10.2.16 AD558 与 8031 单片机的接口电路

由于 AD558 只能提供单极性的电压输出 (0~+2.56V 和 0~+10V)。所以, 要获得双极性输出还需再接一级运放, 加上一个负电源, 通过适当的输出偏置和比例调节才可获得双极性输出。图 10.2.16 所示电路中就是通过另加一个 -5V 电源获得 $\pm 1.28\text{V}$ 输出电压的。偏置由 1.2V 的精密电压基准 AD589 提供, 输入编码与 V_{OUT} 的关系如表 10.2.1 所示。

表 10.2.1

输入编码	V_{OUT}/V
00000000	+1.28
10000000	0
11111111	-1.27

10.2.5 数模转换器的性能指标

1. 分辨率

分辨率 R 反映了输出模拟电压的最小增量, 即表明 DAC 输入一个最低有效位

(LSB)而在输出端上模拟电压的变化量。例如,某DAC的满度输出为10V,输入为16位二进制码,其分辨率

$$R = \frac{10\text{V}}{2^{16} - 1} = 152\mu\text{V}$$

它表示输入一个最低有效位所对应的输出的最小变化量为152 μV 。

2. 线性度

在理想情况下,DAC的数字输入量作等量增加时,其模拟输出电压也应作等量增加,如图10.2.17中斜线所示。但是,实际输出往往有偏离,如图中黑点所示。实际值与理想值的最大偏差 ε 定义为线性误差。若用 Δ 表示输入数码最低有效位发生变化时所引起的输出模拟量的变化量,则线性度一般以 ε/Δ 来表示。手册中给出的典型值为 $\pm \frac{1}{2}\text{LSB}$,即表示 $|\varepsilon| \leq \Delta/2$ 。

3. 精度

精度是表示实际的模拟输出电压与理想的模拟输出电压相差的程度。除线性度不好会影响精度外,元件的精度及稳定性、基准电源的波动、运算放大器增益及偏置的变化等都会影响精度。例如,一个DAC的精度为 $\pm 1\%$,若对应某一个数字量的理论输出模拟电压为10V,则实际的输出模拟电压是9.9~10.1V范围内的某个值。

4. 建立时间

建立时间(又称稳定时间)是描述转换速度的一个参数。建立时间是从输入新的数字量至输出模拟电压稳定在所规定的误差范围内所需要的时间。通常应考虑最坏的情况,即将输入数字量从“全0”变为“全1”时,输出 v_o 从0V变到满度输出电压的规定误差范围(一般为 $\pm \Delta/2$ 或者表示为 $\pm \frac{1}{2}\text{LSB}$)内所需的时间定义为建立时间 t_{set} ,如图10.2.18所示。

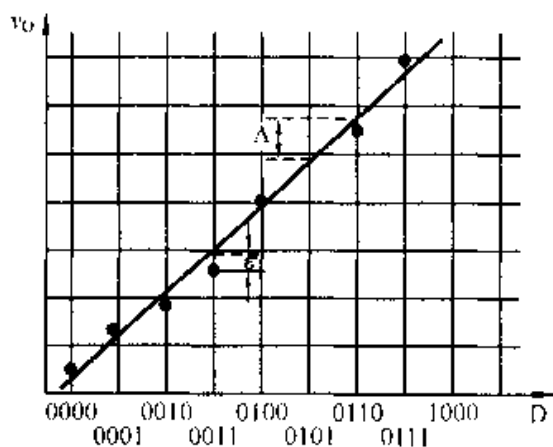


图 10.2.17 线性误差

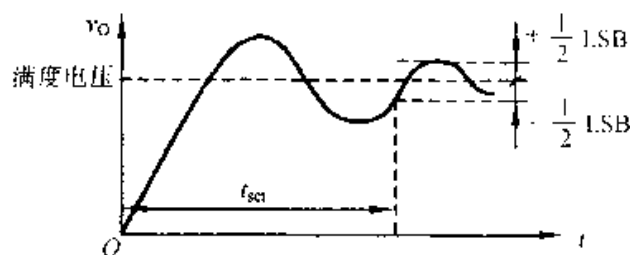


图 10.2.18 建立时间

由于建立时间与误差范围有关,所以,具有高分辨率的DAC,因误差范围窄故

建立时间长。例如, 8 位 DAC 的建立时间为 $1\mu\text{s}$ (典型值), 而 16 位 DAC 的建立时间(典型值)为 $50\mu\text{s}$ 。

10.3 模数转换器

10.3.1 模数转换的基本原理

模拟信号转换为数字信号, 一般分为 4 个步骤进行, 即取样、保持、量化和编码。前两个步骤在取样-保持电路中完成; 后两个步骤则在 ADC 中完成。

1. 取样-保持

取样(采样)是对一个时间上和量值上均是连续变化的模拟量按一定的时间间隔抽取样值。因为 ADC 转换一次需要一定的时间, 所以, 将模拟信号转换为数字信号实际上只能实现将模拟信号的有限个取样值转换为数字信号, 这就需要对模拟信号进行取样。为了保证转换的准确性, 要求在转换过程中取样值保持不变, 这就是保持过程。

模拟信号的取样-保持过程如图 10.3.1 所示。图(a)是取样-保持电路框图, 图(b)是取样-保持过程的波形图。图中, $v(t)$ 为输入的模拟信号, $s(t)$ 为取样信号(取样脉冲), 在时间间隔 τ 内 $s(t)$ 为 1, 在时间间隔 $(T_s - \tau)$ 内 $s(t)$ 为 0。 T_s 为取样信号 $s(t)$ 的周期; $y(t)$ 为取样后的取样输出信号; $y'(t)$ 为取样-保持电路的输出信号。

图 10.3.1(a)电路中的取样器, 实际上是一个受控的电子开关, 它在时间 τ 内是接通状态, 取样器的输出跟踪输入信号, 而在时间 $(T_s - \tau)$ 内是断开状态, 所以取样器的输出为 0。这个开关以周期 T_s 重复动作, 取样器的传递函数

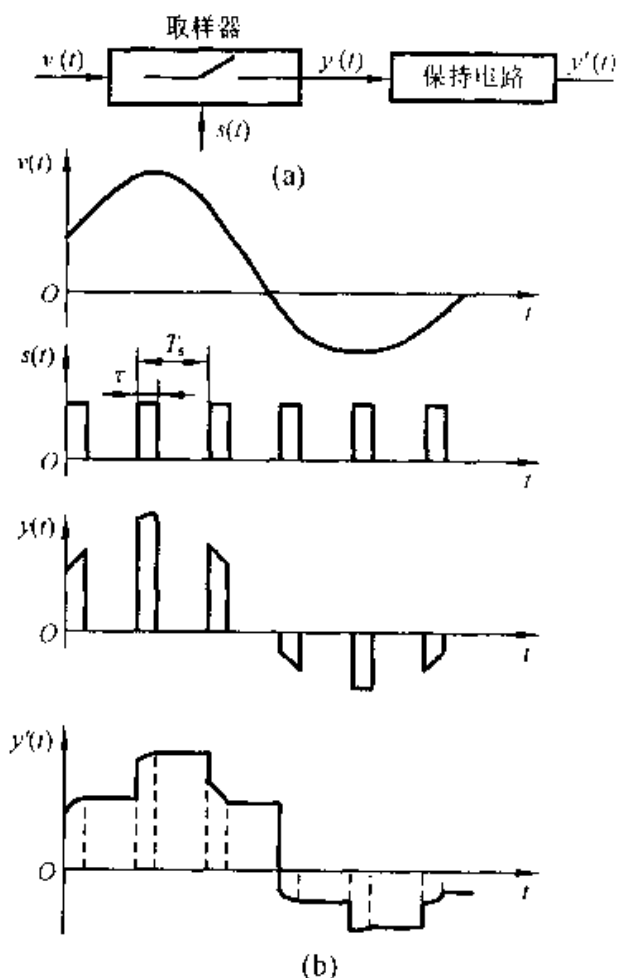


图 10.3.1 取样-保持过程
(a) 电路框图 (b) 波形图

$$y(t) = v(t) \times s(t) = \begin{cases} v(t) & 0 \leq t \leq \tau \\ 0 & \tau \leq t \leq T_s - \tau \end{cases}$$

保持电路输出时,

$$y'(t) = \begin{cases} v(t) & 0 \leq t \leq \tau \\ v(\tau) & \tau < t \leq T_s - \tau \end{cases} \quad (10.3.1)$$

式中, $v(\tau)$ 为 $t = \tau$ 时刻的取样值。

为了使有限个取样值能够很好地代表输入的模拟信号, 对取样信号的频率有一定的要求, 取样定理^①指出: 为了从取样后的输出信号 $y(t)$ 中完全恢复输入的模拟信号 $v(t)$, 则取样信号 $s(t)$ 的重复频率 f_s 至少为 $v(t)$ 的最高有效频率的两倍。

2. 量化与编码

取样-保持电路的输出信号仍然是模拟信号, 若用一个测量单位去测量并取其整数, 然后将这个整数值用一组二进制代码表示, 这就是量化-编码过程。一般把取整量的过程称为量化, 把用二进制代码表示量化值的过程称为编码。

量化过程是一种非线性过程, 它是将幅度连续变化的输入信号变换成一组离散的输出电压。量化过程中采用的测量单位称为量化单位, 对于不足一个量化单位的剩余电压有两种近似处理方式。其一, “只舍不入” 量化方式。若用 q 表示量化单位, 则对于大于 0 而小于 $1q$ 的电压作 $0q$ 处理, 对于大于或等于 $1q$ 而小于 $2q$ 的电压作 $1q$ 处理, 依此类推。其二, “有舍有入” 量化方式。这种量化方式的输入-输出特性如图 10.3.2 所示。由图可见: 对于大于 $(-q/2)$, 小于 $q/2$ 的电压作 $0q$ 处理, 对于大于或等于 $q/2$, 小于 $1.5q/2$ 的电压作 $1q$ 处理, 依此类推。

无论哪一种量化方式都会有一定的误差, 称之为量化误差 ε 。“只舍不入” 的 ε 总是为正的, 且最大量化误差 $\varepsilon_{\max} \approx q$ 。“有舍有入” 量化方式的量化误差有正负之分, 且最大量化误差的绝对值为 $|\varepsilon_{\max}| = q/2$ 。显然, “有舍有入” 量化方式的量化误差比前一种的小些。 q 愈小, 量化误差也愈小。

量化-编码过程是在两个取样脉冲之间的间隔时间内完成的, 因此, 量化-编码过程所需要的时间就是决

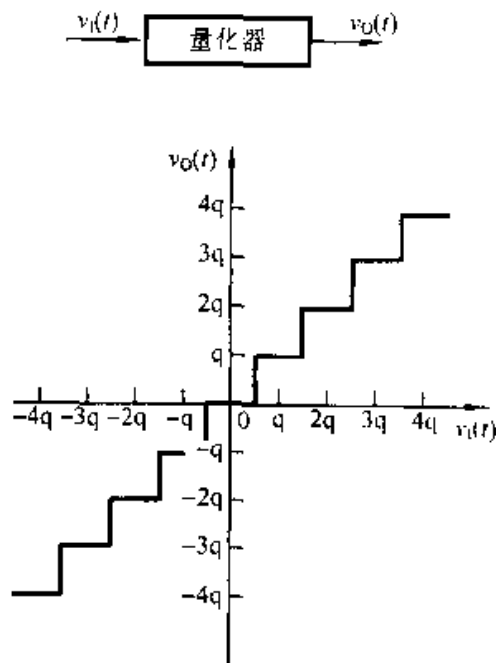


图 10.3.2 有舍有入量化方式的输入-输出特性

① 参阅有关数字信号处理方面的教材。

定 ADC 工作速度的最主要因素。实现模数转换的方案很多,因而 ADC 的电路形式也很多。下面将要讨论的各种模数转换方案,实际上是对各种量化和编码方案进行讨论。有关取样-保持电路内容将在专业课中介绍。

ADC 电路可分为以下两大类。

(a) 直接转换型:通过一组基准电压与取样-保持电路的输出信号进行比较,直接转换为数字量。这类电路的特点是转换速度快,转换精度容易保证。这类转换器广泛用于控制电路。

(b) 间接转换型:它是将取样-保持电路输出的信号首先转换成时间或频率,然后再将时间或频率转换成数字量。这类电路的特点是转换速度低,转换精度较高,抑制干扰能力强,它们在测试仪表中获得广泛应用。下面讨论直接转换型模数转换器,有关间接转换型模数转换器的内容可参阅文献[1、3]。

10.3.2 直接转换型模数转换器

1. 并行 ADC

并行 ADC 又称为快速 ADC,它输出的各位数码是一次形成的,所以,它是转换速度最快的一类 ADC。图 10.3.3 是 3 位数码输出的并行 ADC 的逻辑电路图,由 4 部分组成。

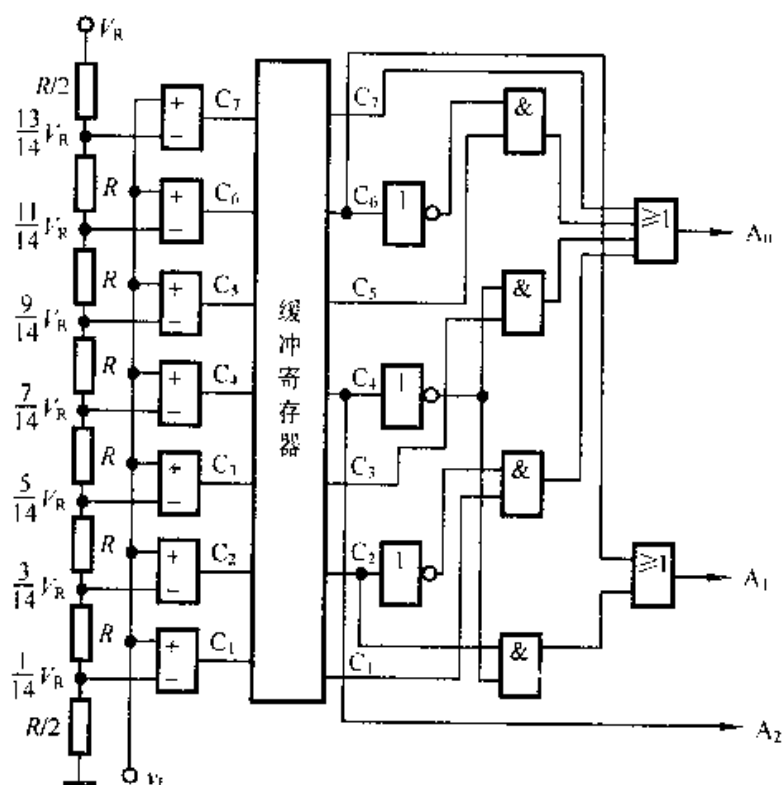


图 10.3.3 3 位二进制代码并行 ADC 逻辑图

(a)电阻分压器。它由8个电阻串联以产生不同数值的参考电压,即形成各种量化电平。需要指出:8个电阻使用的阻值不同会对量化方式或者量化误差产生影响。如图10.3.4所示。在图(a)中,8个电阻相同,阻值均为 R ,所以产生的量化电平为 $V_R/8, \dots, 7V_R/8$ 。两个量化电平之间相差 $V_R/8$,即量化单位 $q=V_R/8$ 。显然,这是采用“只舍不入”量化方式。图(b)和图(c)是采用“有舍有入”量化方式,在图(b)中,上、下两个电阻为 $R/2$,其余均为 R 。所以,产生的量化电平为 $V_R/14, 3V_R/14, \dots, 13V_R/14$,量化单位 $q=V_R/7$ 。因此,各量化电平对应的量化单位为 $0.5q, 1.5q, \dots, 6.5q$ 。在图(c)中,最下面的电阻为 $R/2$,其余均为 R 。所以,产生的量化电平为 $V_R/15, 3V_R/15, \dots, 13V_R/15$,量化单位 $q=2V_R/15$ 。各量化电平对应的量化单位与图(b)相同。其中图(c)电路产生的量化误差最小。

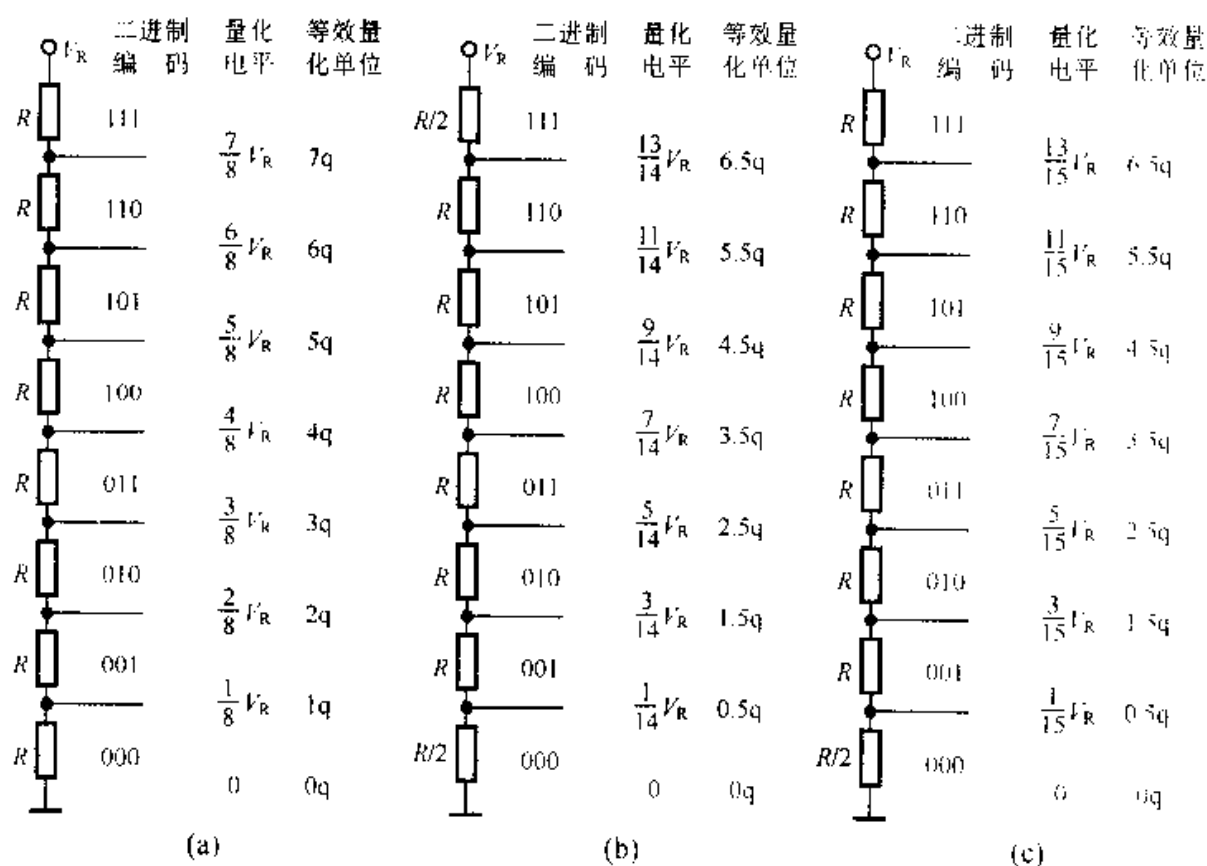


图 10.3.4 量化方式

(a) “只舍不入”量化方式 (b) “有舍有入”量化方式之一 (c) “有舍有入”量化方式之二

(b)比较器。每个比较器的(-)端接相应的量化电平, (+)端全部接入取样-保持电路的输出信号(参见图10.3.1中的 $y'(t)$ 信号),这个信号是要进行量化的信号,在电路中用 v_i 表示。当 v_i 大于或等于量化电平时,比较器输出 C 为1电平;当 v_i 小于量化电平时,比较器的输出 C 为0电平。在参考电压 V_R 范围内,输入信号 v_i 与比较器输出的关系如表10.3.1所示。

表 10.3.1 3 位并行 ADC 代码输出表

v_i	C_7	C_6	C_5	C_4	C_3	C_2	C_1	A_2	A_1	A_0
$1/14V_R > v_i \geq 0$	0	0	0	0	0	0	0	0	0	0
$3/14V_R > v_i \geq 1/14V_R$	0	0	0	0	0	0	1	0	0	1
$5/14V_R > v_i \geq 3/14V_R$	0	0	0	0	0	1	1	0	1	0
$7/14V_R > v_i \geq 5/14V_R$	0	0	0	0	1	1	1	0	1	1
$9/14V_R > v_i \geq 7/14V_R$	0	0	0	1	1	1	1	1	0	0
$11/14V_R > v_i \geq 9/14V_R$	0	0	1	1	1	1	1	1	0	1
$13/14V_R > v_i \geq 11/14V_R$	0	1	1	1	1	1	1	1	1	0
$V_R > v_i \geq 13/14V_R$	1	1	1	1	1	1	1	1	1	1

(c) 缓冲寄存器。由于比较器的延迟时间可能有差异，所以，利用缓冲寄存器来寄存比较的结果，供编码使用。

(d) 编码逻辑电路。其功能是将寄存器输出的信号编译成相应的二进制代码。表 10.3.1 中给出了编码表，由表可知编码器是一个 7 输入 3 输出组合电路。若用传统的“四步法”设计则需应用 7 变量卡诺图，其设计工作量大且设计过程十分繁琐；若采用灵活设计法来设计就能简便地得到结果。仔细分析编码表可直接写出

$$\left. \begin{aligned} A_2 &= C_4 \\ A_1 &= C_2 \overline{C_4} + C_6 \\ A_0 &= C_7 + C_5 \overline{C_6} + C_3 \overline{C_4} + C_1 \overline{C_2} \end{aligned} \right\} \quad (10.3.2)$$

由上式可画出编码器电路，如图 10.3.3 中右边部分所示。

并行 ADC 的优点是具有最快的转换速度，缺点是需要较多的比较器。对于输出 n 位二进制代码的并行 ADC，需要比较器为 $(2^n - 1)$ 个。所以，当输出二进制代码的位数增多时，比较器和其他硬件数量就增多，设备体积增大，造价就昂贵。这类转换器适用于高速度、低精度要求的场合。

2. 并串型 ADC

并行 ADC 输出的各位数码是一次形成的，而下面将要介绍的串行 ADC(如逐次比较型 ADC)输出的各位数码是逐位形成的。并串型 ADC 是这二者的结合，它是将 n 位 ADC 分为两组(或多组)，组内采用并行转换方式，而组间采用串行转换方式。

例如，一个 8 位 ADC，可以分为两组 4 位代码，一组转换高 4 位，送到缓冲寄存器高 4 位。然后，第二组转换低 4 位，最后由时钟脉冲控制 8 位码从缓冲寄存器输出。8 位并串型 ADC 逻辑框图如图 10.3.5 所示。

输入信号 v_i 首先在高 4 位并行 ADC 中转换为 4 位二进制代码输出，送至缓冲寄存器的高 4 位。在转换过程中，量化单位采用 $q_1 = V_{R1}/16$ (V_{R1} 为基准电压)，同时将产生的高 4 位码经 4 位 DAC 转换为模拟信号 v'_R ，它与经过延迟后的同一个输入

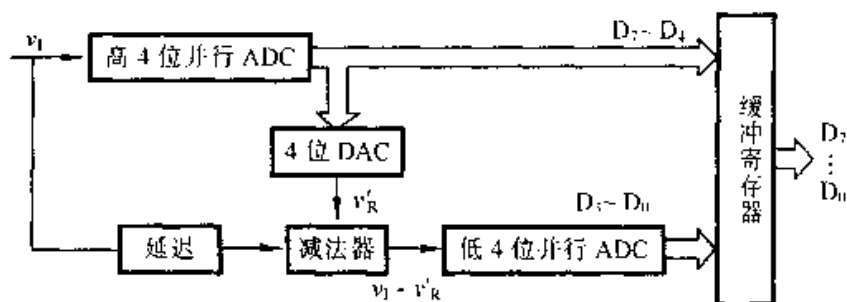


图 10.3.5 并串型 ADC 逻辑框图

信号 v_i 在相减器中相减。由于高 4 位并行 ADC 是采用“只舍不入”量化方式，其最大量化误差为 q_1 ，所以，相减电路输出的最大值为 q_1 。再将相减结果送到低 4 位并行 ADC 中进行转换，这时可以采用“只舍不入”量化方式，也可以采用“有舍有人”量化方式。在本例中采用“只舍不入”量化方式，则量化单位

$$q_2 = \frac{V_{R2}}{2^4} = \frac{q_1}{2^4} = \frac{V_{R1}}{2^8}$$

式中， V_{R2} 为低 4 位并行 ADC 的基准电压，且有 $V_{R2} = q_1$ 。

8 位并串型 ADC 的分辨率取决于低 4 位并行 ADC 的分辨率，所以该转换器的分辨率为 $V_{R1}/2^8$ 。

现举一例来说明其工作过程。

设基准电压 $V_{R1} = 8V$ ，输入信号 $v_i = 5.4V$ 。输入信号 v_i 先加至高 4 位并行 ADC 进行转换，量化单位 $q_1 = V_{R1}/16 = 0.5V$ ，对输入信号 $5.4V$ 量化的结果为 $10q_1$ ，相应的二进制代码为 **1010**。该组代码经 4 位 DAC 转换，其输出模拟电压 $v'_R = 5V$ 。它与延迟后的输入信号在相减器中相减，其差值为 $0.4V$ ，这就是低 4 位并行 ADC 的输入信号。低 4 位并行 ADC 的量化单位应取

$$q_2 = \frac{V_{R2}}{2^4} = \frac{q_1}{2^4} = \frac{V_{R1}}{2^8} = 31.25mV$$

$0.4V$ 输入电压被量化为 $12q_2$ ，所以，低 4 位二进制代码为 **1100**。缓冲寄存器输出 8 位二进制代码为 **10101100**。

并串型 ADC 的主要优点是比同样位数的并行 ADC 所需硬件大大减少。例如，8 位并行 ADC 需要 255 个比较器，而 8 位并串型 ADC 需要比较器的数量为 $(2^4 - 1) \times 2 = 30$ 。

但是，它的转换速度比并行 ADC 的低些，所以，它应用于中等速度、高精度要求的场合。

3. 逐次比较型 ADC

逐次比较型 ADC，又称为逐次逼近型 ADC。它是按串行方式工作的，即转换器输出的各位数码是逐位形成的。图 10.3.6 是 4 位逐次比较型 ADC 的逻辑框图。它由下列几部分组成。

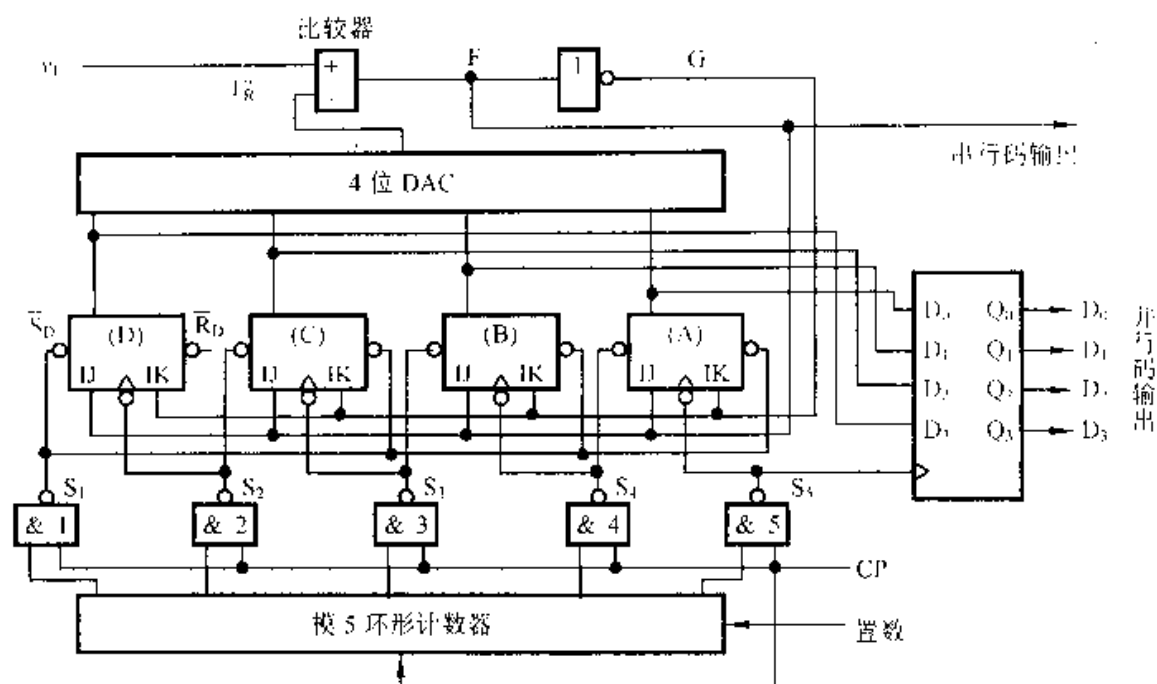


图 10.3.6 逐次比较型 ADC

(a) 4 位 DAC: 它的作用是产生一组与输入数码对应的参考电压 v_R , 送到比较器与输入信号 v_I 进行比较。输入数码 $Q_D \sim Q_A$ 与输出参考电压 v_R 之间的关系如表 10.3.2 所示。

表 10.3.2 4 位 DAC 输出表

Q_D	Q_C	Q_B	Q_A	V'_R	Q_D	Q_C	Q_B	Q_A	V'_R
0	0	0	0	0	1	0	0	0	$8/16V_R$
0	0	0	1	$1V_R/16$	1	0	0	1	$9/16V_R$
0	0	1	0	$2V_R/16$	1	0	1	0	$10/16V_R$
0	0	1	1	$3V_R/16$	1	0	1	1	$11/16V_R$
0	1	0	0	$4V_R/16$	1	1	0	0	$12/16V_R$
0	1	0	1	$5V_R/16$	1	1	0	1	$13/16V_R$
0	1	1	0	$6V_R/16$	1	1	1	0	$14/16V_R$
0	1	1	1	$7V_R/16$	1	1	1	1	$15/16V_R$

(b) 电压比较器：它是将输入信号与 4 位 DAC 输出的参考电压进行比较，若 $v_I \geq v_R$ ，则比较器输出 C 为 1；当 $v_I < v_R$ 时，C 为 0。

(c) 模 5 环形计数器：它的作用是产生比较用的命令信号 $S_1 \sim S_5$ 。在时钟脉冲的作用下产生周期性的命令信号 $S_1 \sim S_5$ ，其工作波形如图 10.3.7 所示。

(d) JK 触发器：在命令信号作用下，存储每次比较的结果，并向 DAC 提供输入数码。

(c) 寄存器：由 4 个 D 触发器构成，在一次转换完成之后，由命令信号 S_1 触发。

并行输出二进制代码。

现举一例来说明其工作过程。

假设:4位DAC的基准电压 $V_R=8V$, 采样-保持电路后的信号电压 $v_i=6.55V$ 。

首先将环形计数器预置在起始状态, 即令 $Q_1Q_2Q_3Q_4Q_5=10000$; 然后在CP脉冲作用下产生命令 S_1 , 将JK触发器的状态置为 $Q_DQ_CQ_BQ_A=1000$, 则DAC输出参考电压为 $v'_R=V_R/2=4V$ 。由于 $v_i > v'_R$, 所以 $C=1, \bar{C}=0$ 。各级JK触发器的输入端为 $J_i=1$ 和 $K_i=0, i=A\sim D$ 。

在命令 S_2 作用下, 下跳沿将触发器D触发, 使 $Q_D=1$, 同时还使触发器C置1。所以, $Q_DQ_CQ_BQ_A=1100$, 经DAC转换后输出参考电压 $v'_R=(12/16)V_R=6V$ 。由于 $v_i > v'_R$, 所以 $C=1$ 和 $\bar{C}=0$ 。各级JK触发器的输入端为 $J_i=1$ 和 $K_i=0$ 。

在命令 S_3 作用下, 触发器C被触发, 使 $Q_C=1$, 同时还将触发器B置1。所以, $Q_DQ_CQ_BQ_A=1110$, 经DAC转换后输出参考电压 $v'_R=(14/16)V_R=7V$ 。由于 $v_i < v'_R$, 所以 $C=0$ 和 $\bar{C}=1$ 。各级JK触发器的输入端为 $J_i=0$ 和 $K_i=1$ 。

在命令 S_4 作用下, 触发器B被触发, 使 $Q_B=0$, 同时还使触发器A置1。所以, $Q_DQ_CQ_BQ_A=1101$, 经DAC转换后输出参考电压 $v'_R=(13/16)V_R=6.5V$ 。由于 $v_i > v'_R$, 所以 $C=1$ 和 $\bar{C}=0$ 。各级JK触发器的输入端为 $J_i=1$ 和 $K_i=0$ 。

在命令 S_5 到来时, 其下跳沿触发触发器A, 使 $Q_A=1$ 。所以, 4级JK触发器的状态为 $Q_DQ_CQ_BQ_A=1101$ 。而它的上升沿将4级JK触发器的1101状态存入到寄存器, 寄存器输出 $Q_3Q_2Q_1Q_0=1101$ 。这就是输入信号 $v_i=6.55V$ 的二进制代码。显然, 寄存器输出的是并行二进制代码, 从以上分析可以看出, 比较器输出端C所顺序输出的是1101串行的二进制代码。

这类转换器在增加二进制代码位数时, 它所需要的硬件成本增加不多。所以, 精度容易达到较高要求。但是它完成一次转换所需的时间为 $(n+1)T_{CP}$ 。其中, n 为二进制代码的位数, T_{CP} 为时钟脉冲周期, 因此, 转换时间随着代码位数 n 增加而加大。它的转换速度比并行和并串型低, 它适用于低速度、高精度要求的场合。

*10.3.3 间接转换型模数转换器

目前, 常用的16位精度以下间接A/D转换器可分为两大类: 电压-时间变换型(V-T变换型)和电压-频率变换型(V-F变换型)。V-T变换型是指ADC电路将输

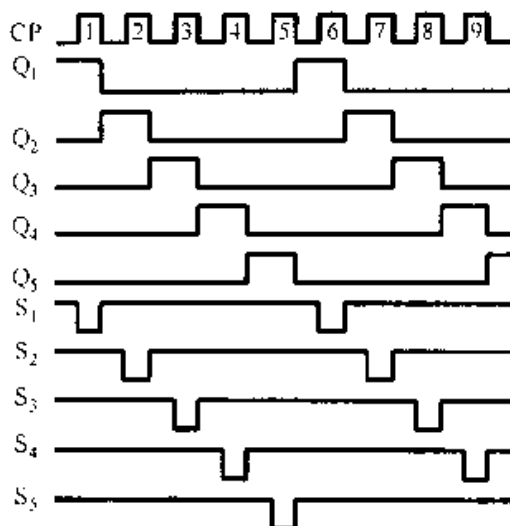


图 10.3.7 模5环形计数器的工作波形图

入的模拟电压 V_i 转换成与之大小成正比的时间间隔 t ，并用计数器对 t 进行计数。计数时钟为固定频率的时钟信号，所得到的计数结果即是正比于输入模拟电压 V_i 的数字量。V-F 变换型 ADC 则是先把输入的模拟电压 V_i 转换成与之大小成正比的频率信号，然后在固定的时间间隔里对得到的频率信号进行计数，计数结果即是正比于输入模拟电压 V_i 的数字量。V-F 变换型中最常应用的是双积分型转换器。它的主要优点是工作性能稳定、精度高、抗干扰能力强，但是，转换时间较长，因而工作速度较低。

常用的高精度 A/D 转换器是指精度在 16 位以上的 A/D 转换器。现在的高精度 ADC 大多属于 Σ - Δ 型 ADC，它是间接转换型 ADC 的一种。

一阶 Σ - Δ 型 ADC 是 Σ - Δ 型 ADC 中最简单的一种。它包含 1 位 ADC（通常称为锁存比较器）、积分器、1 位 DAC、加法器、数字低通滤波器和采样器。其中， Σ - Δ 调制器的作用是对量化噪声整形，以便使其处在超出数字输出滤波器的通带范围。一阶 Σ - Δ 型 ADC 的工作原理如下：假定 V_{IN} 处为直流输入。积分器始终不断地在节点 A 处斜坡式上升或斜坡式下降。比较器的输出端经 1 位 DAC 反馈到节点 B 处的求和输入端。从比较器输出端经 1 位 DAC 回到求和点的负反馈回路迫使节点 B 上的平均直流电压等于 V_{IN} 。这意味着 DAC 的平均输出电压必须等于输入电压 V_{IN} 。DAC 的平均输出电压由来自比较器输出端的 1 位数据流中二进制数据 1 的密度控制。随着输入信号向 $+V_{REF}$ 正向增大，串行位流中 1 的数量增加，0 的数量减少。类似地，随着输入信号向 $-V_{REF}$ 负向增大，串行位流中 1 的数量减少，0 的数量增加。上述分析表明：输入电压的平均值包含在比较器输出的串行位流中。数字滤波器和采样器对这种串行位流进行处理，从而形成最终输出数据。希望进一步了解 Σ - Δ 型 ADC 的读者可参阅文献[12]。

10.3.4 模数转换器的性能指标

模数转换器的性能有许多参数来描述，其中几个主要参数介绍如下。

1. 分辨率

ADC 的分辨率是指输出数字量变化一个最低有效位(LSB)所需要的输入模拟电压的变化量。例如，ADC 输入模拟电压变化范围为 0~10V，输出为 10 位码，则分辨率

$$R_{es} = \frac{\Delta V}{2^n - 1} = \frac{10V}{2^{10} - 1} = 9.77\text{mV}$$

2. 精度

ADC 的精度决定于量化误差及系统内其他误差之总和。一般精度指标为满量程的 $\pm 0.02\%$ ，高精度指标为满量程的 0.001% 。

3. 转换时间

转换时间是描述工作速度的指标，典型值为 $50\mu\text{s}$ ，高速 ADC 的转换时间为

50ns。

4. 输入阻抗

输入阻抗值与 ADC 的类型有关, 典型值的输入电阻为 $1\text{k}\Omega \sim 1\text{M}\Omega$; 输入电容为几十 pF。

5. 稳定度

ADC 的精度一般受温度影响。典型的温度误差系数为每度(摄氏)占满量程的百万分之二十。

*10.4 集成模数转换器

随着数字技术在各学科领域和国民经济的各行各业中普遍应用, 对 ADC 和 DAC 的需求大量增加, 而且要求性能指标有较宽覆盖面, 以便适应不同场合应用的要求。近几年, ADC 和 DAC 的生产已进入全集成化阶段, 同时在转换速度和转换精度等主要指标上有了重大突破, 还开发了一些具有与计算机直接接口功能的芯片。本节介绍一种在自动数据采集和处理系统中应用十分广泛的集成模数转换器 ADC0809 和另一种高精度集成模数转换器 AD7710。

1. ADC0809

ADC0809 是 8 位逐次比较型 ADC。它是属于 CMOS 器件, 输出逻辑电平与 TTL 电平兼容, 并且输出设置有三态输出锁存器, 便于同微机接口; 输入设置有 8 个通道, 可以选择其中任一通道的模拟信号进行转换。图 10.4.1 是它的逻辑框图。电路可分为 3 部分: 模拟多路器、模数转换器和控制器。

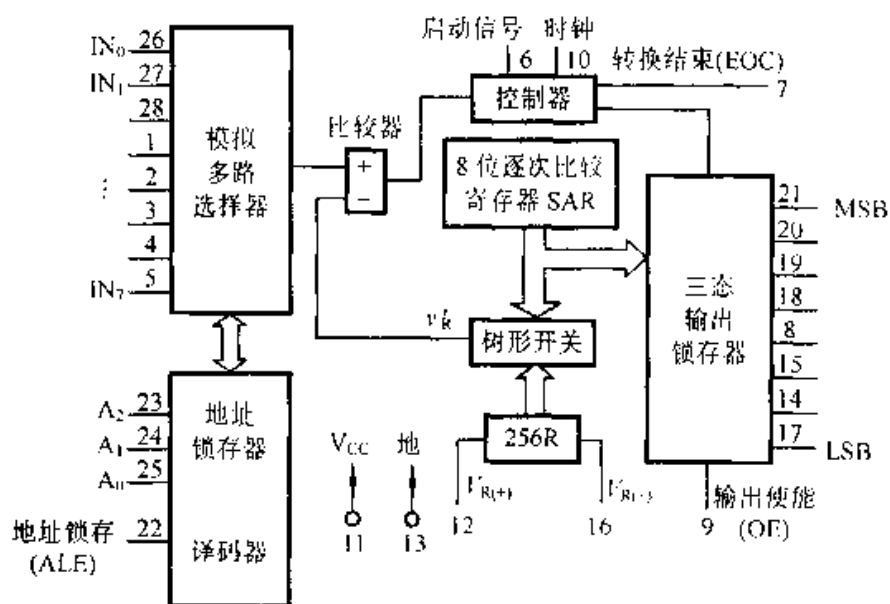


图 10.4.1 ADC0809 逻辑框图

(1) 模拟多路器

ADC0809 可以处理来自 8 个不同信源的模拟信号,这种功能是由 8 通道模拟多路器和地址锁存译码器共同完成的。在地址锁存使能端 ALE 的正脉冲作用下,将地址线 $A_2A_1A_0$ 上的选通地址锁存在锁存器中,并经 3-8 线译码器使相应输入通道的模拟开关接通。地址与输入通道的关系如表 10.4.1 所示。

(2) 模数转换器

模数转换器的基准电源由 256R 电阻网络和外加基准电压 $V_R(+)$ 、 $V_R(-)$ 产生,然后由树形开关选择一个量化电平输出,并加至比较器与输入模拟电压进行比较。树形开关的状态由逐次比较寄存器 SAR 的输出控制,8 位 SAR 有 256 种输出状态,每一种输出状态控制树形开关选择一个与其对应的量化电平输出。比较器的比较结果由 SAR 记忆下来,即比较器的输出改变了 SAR 的输出状态,因此,它控制树形开关又选择另一个量化电平输出,并在比较器中与输入模拟电压再进行比较。如此继续下去,直到转换过程结束为止。

从上述分析可以看出:比较器每次比较结果在 SAR 中进行迭代, n 位转换需要 n 次迭代。显然 SAR 的作用与图 10.3.7 中 4 级 JK 触发器的功能类似,而树形开关的作用相当于一个 8 位 DAC(关于树形开关网络的连接方法参阅 10.3 节)。此外,从图 10.4.2 可以看出:电阻分压器的上、下两个电阻为 $R/2$,其余均为 R ,因此,该电路是采用“有舍有入”量化方式,其最大量化误差为 $\pm \frac{1}{2} \text{LSB}$ 。

表 10.4.1 地址与输入通道关系表

地 址			输入通道
A_2	A_1	A_0	
0	0	0	IN_0
0	0	1	IN_1
0	1	0	IN_2
0	1	1	IN_3
1	0	0	IN_4
1	0	1	IN_5
1	1	0	IN_6
1	1	1	IN_7

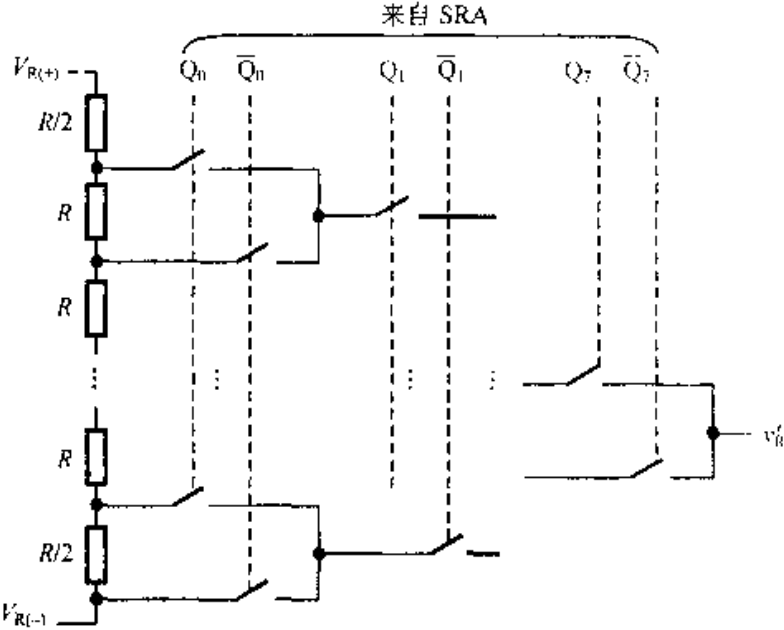


图 10.4.2 256R 树形开关网络 DAC

2. AD7710

AD7710 是一种 Σ - Δ 型的模数转换器。它是美国模拟器件公司 (简称 AD 公司)

推出的一个高精度产品,分辨率高达24位。同一系列的还有AD7711~AD7713。它的性能指标优良,功能齐全,主要应用于各种高精度仪表中,可为低频测量应用提供一个完整的模拟前端接口:直接从传感器接受微弱信号并转换成数字信号,在片内含有数字滤波器,可以通过软件设置滤波器的截止频率和建立时间。

AD7710的主要特点如下:

- (a) 双向微控制器的串行接口;
- (b) 具有截止频率可编程的低通滤波器;
- (c) 具有读和写校准系数的功能;
- (d) 允许使用者读或写芯片上的校准寄存器;
- (e) 允许单电源工作和双电源工作。

AD7710电路结构如图10.4.3(a)所示,图(b)是逻辑框图,图(c)为引脚图。

主要引脚的功能说明如下。

➤ SCLK: 串行时钟脉冲。当MODE是高电平时, SCLK脚输出串行时钟脉冲,器件为内时钟模式;当MODE是低电平时,外部串行时钟脉冲从此脚输入。

➤ MCLK IN: 器件的主时钟信号。

➤ MCLK OUT: 器件的主时钟信号。主时钟由石英晶体振荡器产生时,石英晶体连接在MCLK IN与MCLK OUT之间。

➤ A_0 : 地址输入。低电平时,控制寄存器的读/写;高电平时,进行数据寄存器和校准寄存器的存取。

➤ $\overline{\text{SYNC}}$: 逻辑输入。在含有多片AD7710的系统中,用于对各个AD7710进行同步控制。

➤ MODE: 逻辑输入。高电平时,为内时钟模式。低电平时,为外时钟模式。

➤ $\text{AIN}_{1(+)}$: 通道(1)的模拟输入。 $\text{AIN}_{1(-)}$: 通道(1)的模拟输入。

➤ $\text{AIN}_{2(+)}$: 通道(2)的模拟输入。 $\text{AIN}_{2(-)}$: 通道(2)的模拟输入。

➤ $\text{REF IN}_{(+)}$: 参考电压输入端。 $\text{REF IN}_{(-)}$: 参考电压输入端。

➤ REF OUT: 内部2.5V参考电压输出端。

➤ $\overline{\text{TFS}}$: 发送帧同步信号,低电平有效。在脉冲下降沿后串行数据有效。

➤ $\overline{\text{RFS}}$: 接收帧同步信号,低电平有效。

➤ $\overline{\text{DRDY}}$: 逻辑输出,其输出低电平表示新的输出数字信号已准备完毕,可用于传输。输出数字信号传输完成后,其输出又从低电平回到高电平。

➤ SDATA: 串行数据。

图10.4.4所示为AD7710单电源工作的典型接线图。

AD7710与8031的接口连接如图10.4.5所示。AD7710工作在外时钟模式,而8031则工作在模式0的串行接口模式。AD7710的 $\overline{\text{DRDY}}$ 与8031的 $\text{P}_{1.2}$ 口相连,8031通过查询的方式检测 $\overline{\text{DRDY}}$ 的状态。如果需要用中断的方式,可以将 $\overline{\text{DRDY}}$ 连接到

8031 的 $\overline{\text{INT}}_0$ 或 $\overline{\text{INT}}_1$ 端。在 8031 向 AD7710 写命令字时, 8031 的串口首先输出的是 LSB, 而 AD7710 应该接受的却是 MSB, 因此, 8031 在写命令字时应作调整。

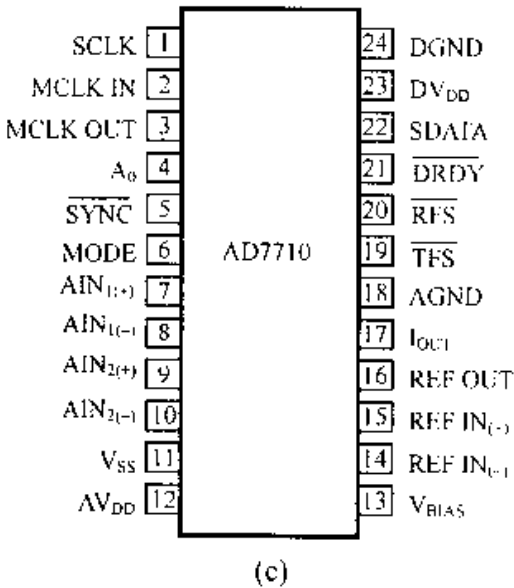
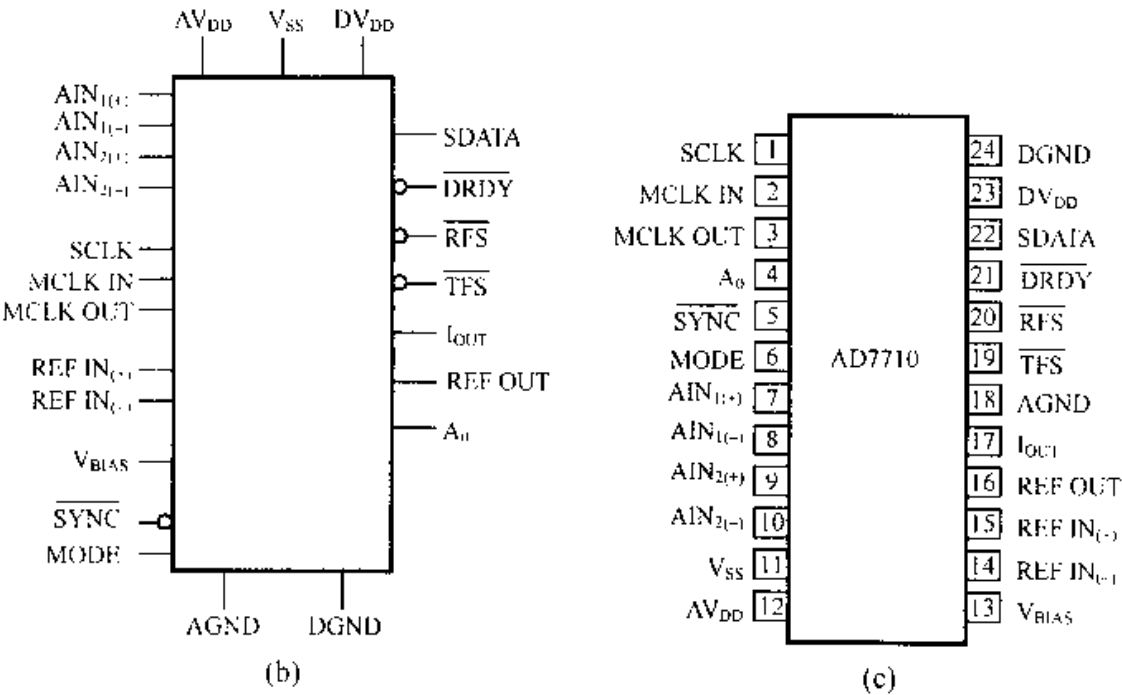
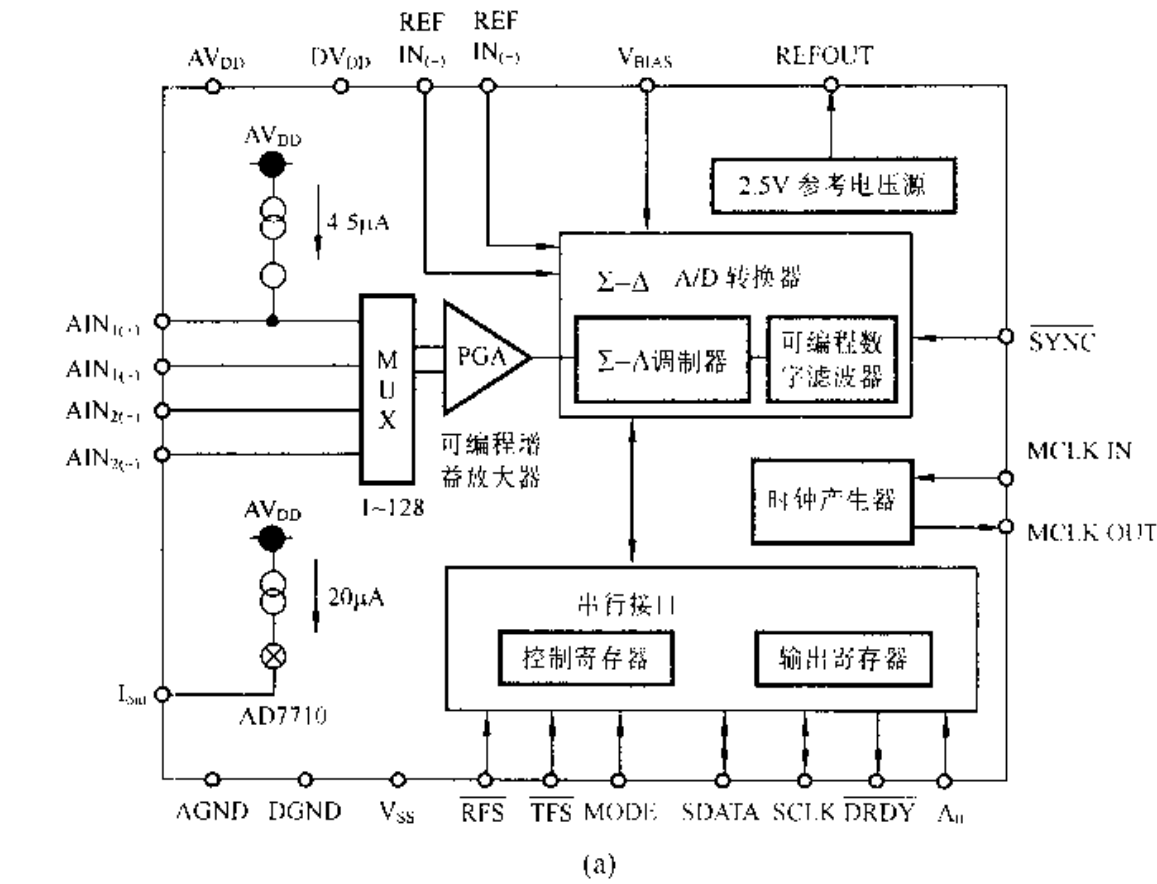


图 10.4.3 AD7710 模数转换器

(a) 电路结构图 (b) 逻辑框图 (c) 引脚图

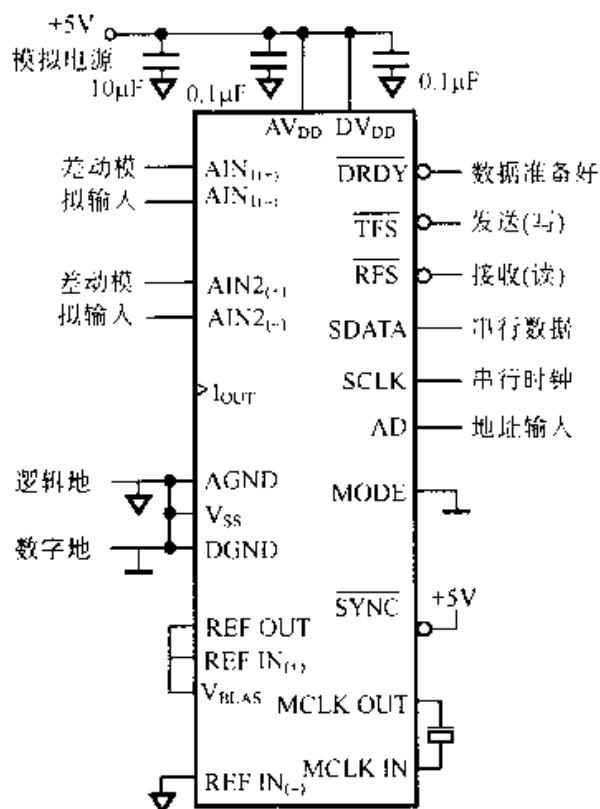


图 10.4.4 AD7710 单电源工作典型接线图

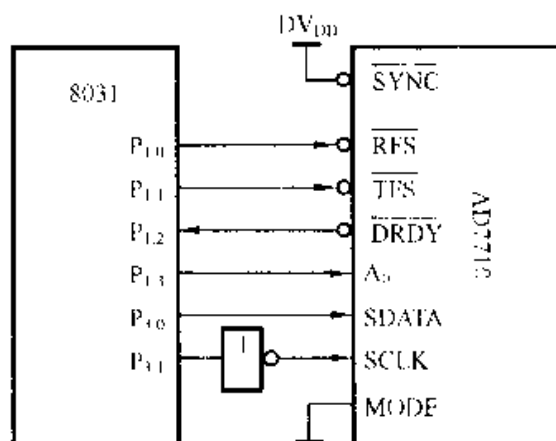


图 10.4.5 AD7710 与 8031 单片机接口连接图

同样，在读取 AD7710 的数据时，AD7710 首先输出的是 MSB，而 8031 则期望先读到 LSB，因此，8031 在读取数据后需对数据进行调整。

下面是 8031 从 AD7710 中读取数据的程序：

```

MOV SCON, #00010000B    ; 设置 8031 的串行口
MOV IE, #00010000B      ; 关中断
SETB P1.0                ; 设置 P1.0，用作  $\overline{\text{RFS}}$ 
SETB P1.1                ; 设置 P1.1，用作  $\overline{\text{TFS}}$ 
SETB P1.3                ; 设置 P1.3，用作  $\text{A}_0$ 
MOV R1, #03H             ; 读 3 个字节的数据
MOV R0, #30H             ; 数据缓冲区首址
WAIT: JB P1.2, WAIT       ; 查询 AD7710
CLR P1.0                 ; 使  $\overline{\text{RFS}}$  变低
CLR RI                   ; 清 RI 标志，准备接受数据
READ1: JNB RI, READ1      ; 未接收完一帧，循环等待
MOV A, SBUF              ; 从串口缓冲器读数据
MOV C, ACC.0             ; 调整数据
MOV 07H, C

```

MOV C, ACC.1	
MOV 06H, C	
MOV C, ACC.2	
MOV 05H, C	
MOV C, ACC.3	
MOV 04H, C	
MOV C, ACC.4	
MOV 03H, C	
MOV C, ACC.5	
MOV 02H, C	
MOV C, ACC.6	
MOV 01H, C	
MOV C, ACC.7	
MOV 00H, C	
MOV A, 20H	; 得到调整后的数据
MOV @R0, A	; 数据送缓冲区
INC R0	; 修改数据缓冲区地址
DJNZ R1, WAIT	; 判断是否读完 3 字节数据
RET	; 读完 3 字节数据, 返回

小 结

D/A 转换器和 A/D 转换器是数字系统与模拟系统之间的接口, 随着数字设备和计算机的广泛应用, 数模及模数转换技术得到迅速发展, 各种单片集成的 D/A 转换器和 A/D 转换器不断出现。本章介绍了常见的几种 D/A 转换器和 A/D 转换器的工作原理和实例。

在 D/A 转换器产品中, $R-2R$ T 型电阻网络 D/A 转换器是目前应用较多的一种电路。它由于电阻品种少, 易于提高精度, 并且电流型倒 T 型电阻网络 D/A 转换器具有较高的工作速度。把接口部件与 D/A 转换器一同集成在芯片上, 即可构成与微机兼容的单片集成 D/A 转换器, 目前, 这类芯片应用十分广泛。

在控制系统中, 采用直接转换型 A/D 转换器居多。其中, 全并型工作速度最高, 但电路结构复杂且成本很高; 并串型工作速度中等, 但硬件数量和成本大大减少, 所以, 它是兼顾到速度与成本的一种较理想的折衷方案; 逐次比较型 A/D 转换器工作速度较低, 但是转换精度高, 是集成 A/D 转换器中使用较多的一种电路; 双积分

型转换器工作速度较低,但是转换精度高、工作性能稳定、抗干扰能力强。是间接转换型模数转换器中使用较多的一种电路。

在选用 A/D 转换器和 D/A 转换器时,工作速度和转换精度是最重要的技术指标。

思考题和习题

10.1 解释下列名词:

分辨率、转换精度、转换时间、量化、量化单位。

10.2 一个 D/A 转换器应包含哪几部分,它们的功能是什么?

10.3 设满度输出电压为 +5V,问二进制梯形网络需用多少位才能达到 1mV 的分辨率。

10.4 设有一个 6 位 T 型 DAC 的基准电压为 V_R ,求下列输入数字量对应输出模拟电压为多少?

(1) 101001;

(2) 111011;

(3) 110001。

10.5 已知 12 位二进制 D/A 转换器满度输出为 10V。试确定它的分辨率和对应一个最低位的电压。

10.6 8 位单极性 D/A 转换器满度输出为 -8V。求输入为 11100000 时,输出模拟电压为多少?

10.7 8 位双极性 D/A 转换器满度输出为 $\pm 5V$ 。采用符号-绝对值表示法,问数字输入为 10110000 及 10000000 时,输出模拟电压多大?

10.8 模数转换包括哪几个工作步骤,它们完成的功能是什么?

10.9 A/D 转换器的实现方案有哪几种,它们各有何优缺点。

10.10 画出输出为 4 位二进制数码的并行 A/D 转换器的逻辑框图,并推导编码电路的逻辑表达式。

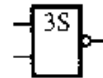
10.11 3 位并行 A/D 转换器采用有舍有入量化方式,设参考电压为 $V_R=7V$,输入模拟电压 $v_i=3.4V$ 。试求 3 位并行 A/D 转换器的输出数码?

10.12 画出输出为 8 位二进制数码的并串型 A/D 转换器的逻辑框图。设输入模拟电压 $v_i=9.4V$,参考电压 $V_R=16V$ 。求输出数字量并确定该转换器的分辨率。

10.13 要求 A/D 转换器能分辨 0.0025V 的电压变化,其满度输出所对应的输入电压为 9.9976V,问该转换器至少应有多少位。

10.14 设逐次比较型 A/D 转换器的参考电压为 8V,输入模拟电压有 2.55V。问该转换器输出的 4 位码及 6 位码分别是什么?

附录一 常用逻辑单元图形符号对照表

序号	名称与说明	沿用符号	国标符号
01	与门		
02	与非门		
03	或门		
04	或非门		
05	输出无放大的缓冲单元		
06	非门		
07	异或门		
08	同或门（异或非门）		
09	集电极开路输出		
10	与非门（L型OC）		
11	反相缓冲器驱动器（L型OC）		
12	与非缓冲器（3S）		

续表

序号	名称与说明	沿用符号	国标符号
13	与非门 (带有施密特触发器)		
14	半加器		
15	全加器		
16	与或非门		
17	基本 RS 触发器		
18	脉冲触发主从 JK 触发器		
19	维持-阻塞型 D 触发器		
20	正边沿型 D 触发器		
21	负边沿型 JK 触发器		

注：(1) 沿用符号是原四机部颁布的标准：BG3432-82、BG3435-82。

(2) 国标符号是国家标准局颁布的国家标准：BG4.728.12.85。

附录二 本书中的文字符号和图形符号及其说明

(一) 文字符号及其说明

1. 电压符号

v_i	输入电平	v_o	输出电平
V_{OL}	输出低电平	V_{OH}	输出高电平
V_{CC}	电源电压(晶体管器件)	V_{DD}	电源电压(MOS 器件)
V_{NH}	输入高电平抗干扰容限	V_{NL}	输入低电平抗干扰容限
v_{be}	三极管基极相对于发射极的电压	v_{ce}	三极管集电极相对于发射极的电压
v_{DS}	MOS 管漏极相对于源极的电压	v_{GS}	MOS 管栅极相对于源极的电压
V_T	门电路的阈值电压	V_{TN}	N 沟道 MOS 管的开启电压
V_{TP}	P 沟道 MOS 管的开启电压	V_{IN}	N 沟道 MOS 管的夹断电压
V_{IP}	P 沟道 MOS 管的夹断电压	v_N	运算放大器反相端电压
v_p	运算放大器同相端电压	V_{TH}	施密特触发器的上限触发电平
V_{TL}	施密特触发器的下限触发电平	ΔV	施密特触发器的回差电压
V_{on}	开门电平	V_{off}	关门电平
V_m	脉冲幅度		

2. 电流符号

i_c	集电极电流	i_b	基极电流
i_e	发射极电流	I_{cs}	集电极饱和电流
I_{bs}	基极饱和电流	i_i	输入电流
I_{IL}	低电平输入电流	I_{IH}	高电平输入电流
i_o	输出电流	I_{OL}	低电平输出电流
I_{OH}	高电平输出电流	I_{RL}	负载电流
i_{DS}	漏极电流	i_{GS}	栅极电流

3. 功率符号

P	平均功耗	P_l	空载导通功耗
P_{II}	空载截止功耗	P_S	静态功耗
P_D	动态功耗	P_C	负载电容充、放电功耗
P_T	瞬时导通功耗		

4. 电阻、电容符号

C_L	负载电容	C_O	分布电容
C_{ext}	外接电容	R_{ext}	外接电阻
R_i	输入电阻	R_O	输出电阻

R_L 负载电阻 R_{off} 器件截止时内阻 R_{on} 器件导通时内阻**5. 时间符号** t_{pd} 平均传输延迟时间 t_{PLH} 截止延迟时间 t_r 上升时间 t_f 下降时间 t_{off} 关闭时间 t_R 恢复时间 t_{PHL} 导通延迟时间 t_d 延迟时间 t_s 存储时间 t_{on} 开启时间 t_w 脉冲宽度 t_{set} 建立时间**6. 器件及引脚符号**

D 二极管

G 门

 T_N N沟道 MOS 管 T_{LP} PMOS 负载管 T_G 传输门

T 三极管

S 开关

 T_P P沟道 MOS 管 T_{ON} NMOS 输出管

CP、CLK 时钟

EN、G、S、CS、CE、 E_T 、 E_P 选通（使能）

OE 输出允许

LD 置数控制

M 模式控制

 D_{SR} 右移串行输入 O_C 进位输出 O_{CR} 串行输出

RAS 行地址选通

 X_i 行选通 O_C/O_B 进位/借位输出 $C_i(CLR)$ 清零（清除、复位） SH/\overline{LD} 移位/置数 M_A 、 M_B 模式选择 D_{SL} 左移串行输入 O_B 借位输出

WE 写使能

CAS 列地址选通

 Y_j 列选通**(二) 图形符号及其说明**

本书中小规模集成器件的逻辑符号以及中、大规模集成器件的逻辑电路均采用国标符号（简称国标，即国家标准 BG4.728.12—85）绘制。为了方便教学和图形绘制，在介绍中，大规模集成器件构成的应用电路中，本书采用逻辑框图来表示这些器件，这些逻辑框图与原电子工业部颁布的逻辑符号是一致的。在这些逻辑框图中，凡是输入、输出和控制信号以低电平作为有效信号时（即“低电平有效”），在信号线上加有小圆圈，并在相应的符号上加有非号（“-”），这些带非号的符号可以标注在框内或框外。需要指出：中、大规模集成器件的逻辑框图与国标的逻辑符号有一些区别，读者可查阅“常用逻辑单元图形符号对照表”（附录一）。

附录三 汉英名词、缩写词对照表

二 画

二进制/binary
二进制数/binary number
二进制地址/binary address
二-十进制编码/binary coded decimal (BCD)
二极管/diode
发光二极管/light emitting diode(LED)
肖特基二极管/schottky diode
十进制/decimal
十进制数/decimal number
八进制/octal
八进制数/octal number
十六进制/hexadecimal
十六进制数/hexadecimal number

三 画

三极管/bipolar junction transistor (BJT)
肖特基三极管/schottky transistor
多发射极三极管/multiemitter transistor
门/gate
与门/AND gate
或门/OR gate
非门/NOT gate
与非门/NAND gate
异或门/exclusive OR gate
同或门/exclusive NOR gate
三态门/three state gate
集电极开路门/open collector gate
双向传输门/bilateral transmission gate
上升时间/rise time
上升沿/rise edge
下降时间/fall time
下降沿/fall edge

四 画

计数器/counter
二进制计数器/binary counter
十进制计数器/decimal counter

可逆计数器/up-down counter
异步计数器/synchronous counter
同步计数器/asynchronous counter
计算机/computer
比较器/comparator
开启电压/threshold voltage
开关时间/switching time
开关特性/switching characteristics
反相器/inverter
反码/one's complement
反演规则/complementary operation theorem
反向恢复时间/reverse recovery time
双稳态/bistable
双稳态多谐振荡器/bistable astable
multivibrator
分配器/distributor
分辨率/resolution

五 画

布尔代数/boolean algebra
半导体/semiconductor
加法器/adder
半加法器/half adder
全加法器/full adder
串行进位加法器/serial carry adder
超前进位加法器/look-ahead carry adder
发射极/emitter
发射极耦合逻辑/emitter coupled logic
卡诺图/karnaugh map
可编程逻辑器件/programmable logic device
(PLD)
复杂可编程逻辑器件/complex programmable
logic device(CPLD)
可编程阵列逻辑/programmable array logic
(PAL)
可编程逻辑阵列/programmable logic array
(PLA)

可编程逻辑模块/programmable logic block
(PLB)

六 画

多谐振荡器/astable multivibrator
字/word
结构控制字/architecture control word
电子标签字/electronic signature word (ES)
权/weight
回差电压/backlash voltage
行选择线/row-select line
列选择线/column-select line
存储器/memory
串行存储器/serial access memory (SAM)
随机存储器/random access memory (RAM)
双端口随机存储器/dual-port RAM(DPRAM)
静态存储器/static memory (SRAM)
动态存储器/dynamic memory (DRAM)
闪存存储器/flash memory
半导体存储器/semiconductor memory
只读存储器/read only memory(ROM)
快闪只读存储器/flash memory
可编程只读存储器/programmable read only
memory (PROM)
串行配置(可编程只读)存储器/serial
configuration PROM(SCP)
可擦可编只读存储器/erasable programmable
read only memory (EPROM)
电可擦可编只读存储器/electrically erasable
programmable read only memory (E²PROM)
存储单元/memory cell
静态存储单元/static memory cell
动态存储单元/dynamic memory cell
存储时间/storage time
传输延迟时间/propagation delay time
在系统可编程/in-system programmability(ISP)
地址/address
先进先出/first-in first-out(FIFO)
先入后出/first-in last-out(FILO)

七 画

译码器/decoder
7 段译码器/Segment decoder
二进制译码器/binary decoder
二-十进制译码器/BCD decoder
时序逻辑电路/sequential logic circuit
时钟/clock (CLK)
时钟脉冲/clock pulse (CP)
时钟频率/clock frequency
配置时钟/configuration clock (CCLK)
时间常数/Time constant
进位/carry
位线/bit line
补码/complement code
余 3 码/excess three code
运算电路/arithmetic circuit
驱动方程/driving equation
状态/state
状态表/state table
状态图/state diagram
状态方程/state equation
状态分配/state assignment
状态简化/reduction of state
次状态/next state
现状态/present state

八 画

组合函数/combinatorial function
组合逻辑电路/combinational logic circuit
现场可编程门阵列/field programmable Gate
array(FPGA)
定时器/timer
奇偶校验/odd-even check
参考电压/reference voltage
函数发生器/function generator
单位间距码/unit distance code
建立时间/setup time
取样时间/sampling time
放大器/amplifier
金属-氧化物-半导体/metal oxide

十一画

semiconductor (MOS)
互补 MOS 管/complemental metal oxide
semiconductor (CMOS)
MOS 场效应管/ metal oxide semiconductor
field-effect transistor (MOSFET)
叠层栅 MOS 管/stacked-gate injection metal
oxide semiconductor (SIMOS)

九 画

显示器/display
7 段显示器/seven segment display
液晶显示器/liquid crystal display(LCD)
字符显示器/character mode display
7 段字符显示器/seven-segment character mode
display
复位/reset
栅极/gate
总线/bus
保持时间/hold time
恢复时间/recovery time
相邻项/adjacency
选择器/selector
选通脉冲/gate pulse

十 画

真值表/truth table
特征方程/characteristic equation
扇入/fan in
扇出/fan out
竞争-冒险/race and hazard
借位/borrow
通用阵列逻辑/generic array logic(GAL)
积分电路/Integrating circuit
积分器/integrator
格雷码/gray code
流程图/flow chart
乘积项/product term
逐次比较型转换器/successive approximation
converter
原变量/uncomplemented
起始状态/initial state

常量/constant
阈值电压/threshold voltage
逻辑/logic
逻辑电平/ logic level
逻辑函数/ logic function
逻辑变量/ logic variables
逻辑符号/ logic symbol
逻辑表达式/ logic expression
逻辑门/ logic gate
逻辑电路/ logic circuit
逻辑代数/ logic algebra
逻辑和/ logic sum
逻辑积/ logic product
逻辑运算/ logic operation
逻辑设计/ logic Design
逻辑单元陈列/Logic cell array(LCA)
寄存器/register
移位寄存器/shift register
双向移位寄存器/bidirectional shift register
寄存器传输语言/ register transfer language
(RTL)
减法器/subtractor
基极/base
缓冲器/buffer
总(系统)缓冲器/global buffer
辅助缓冲器/alternate buffer
晶体振荡缓冲器/crystal oscillator buffer
双向隔离缓冲器/bidirectional interconnect
buffer (BIDI)

十二画

最小项/miniterm
最大项/maxterm
最低有效位/least significant bit(LSB)
最高有效位/most significant bit(MSB)
量化/quantification
量化误差/quantification error
集电极/collector
集电极开路输出/open collector output

编码/coding
 编码器/encoder
 优先编码器/priority encoder
 锁存器/latch
 集成电路/integrated circuit(IC)
 专用集成电路/application specific integrated circuit (ASIC)
 小规模集成/small scale integration (SSI)
 中规模集成/medium scale integration (MSI)
 大规模集成/Large scale integration (LSI)
 超大规模集成/very large scale integration (VLSI)
 集成注入逻辑/integrated injection logic(I²L)
 硬件描述语言/hardware description language (HDL)
 循环码/cyclic code
 循环回路/recirculating loop
 等价/equivalent
 晶体管-晶体管逻辑/transistor-transistor logic (TTL)

十三画

数字电路/digital circuit
 数字系统/digital system
 数字技术/digital technique
 数字测量/digital measurement
 数制/number system
 数码管/Nixie light
 数值比较器/digital comparator
 数据选择器/multiplexer
 数-模转换/digital to analog conversion
 数-模转换器/digital to analog convertor (DAC)
 权电阻数-模转换器/weighted resistor digital to analog convertor
 权电流数-模转换器/weighted current digital to analog convertor

开关树型数-模转换器/switch tree type digital to analog convertor
 倒 T 型电阻数-模转换器/inverted ladder resistor digital to analog convertor
 置位/set
 源极/source
 触发器/flip-flop
 主从触发器/master-slave flip-flop
 边沿触发器/edge-triggered flip-flop
 单稳态触发器/monostable multivibrator
 施密特触发器/schmitt trigger
 触发/trigger
 简化/simplification

十四画

模拟电路/analog circuit
 模-数转换/analog to digital conversion
 模数转换器/analog to digital converter(ADC)
 漏极/drain
 漏极开路门电路/open drain gate(OD 门)
 截止/cut-off
 稳定状态/stable state
 模拟开关/analog'switch
 算术逻辑单元/arithmetic logic unit(ALU)

十五画

摩根定律/de morgan's theorem
 蕴涵/implicant

十六画

噪声容限/noise margin
 激励表/excitation table

十八画

瞬态/transient state
 覆盖/covering
 翻转/turnover

参 考 文 献

- 1 曹汉房,陈耀奎编著.数字技术教程.北京:电子工业出版社,1995
- 2 王毓银主编.数字电路逻辑设计(第三版).北京:高等教育出版社,1999
- 3 刘宝琴编著.数字电路与系统.北京:清华大学出版社,1993
- 4 康华光主编.电子技术基础(数字部分)(第四版).北京:高等教育出版社,2000
- 5 曹汉房编.数字电路的设计与解题技巧.北京:高等教育出版社,1988
- 6 张端编.数字电路与逻辑设计.北京:高等教育出版社,1985
- 7 蒋大宗等编.数字逻辑.北京:电子工业出版社,1984
- 8 罗朝杰编著.数字逻辑设计基础.北京:人民邮电出版社,1982
- 9 顾德仁主编.脉冲与数字电路.第二版.北京:高等教育出版社,1984
- 10 阎石主编.数字电子技术基础.北京:高等教育出版社,2001
- 11 孙涵芳,徐爱卿编著.可编程逻辑器件.北京:航空航天大学出版社,1990
- 12 刘书明,刘斌编著.高性能模数与数模转换器件.西安:西安电子科技大学出版社,2000
- 13 陈俊亮编著.数字电路逻辑设计.北京:人民邮电出版社,1980
- 14 蓝江桥、李跃华主编.数字系统与逻辑设计.武汉:湖北科技出版社,1997
- 15 [美]J.米尔曼著.微电子学:数字和模拟电路及系统(上、中、下册).清华大学电子学教研组译.北京:人民教育出版社,1980~1982
- 16 Xilinx:The Programmable Gate Array Date Book,Xilinx,Inc,1992
- 17 Xilinx:LCA Programmable Gate Array User's Guide,Xilinx,Inc,1990
- 18 Xilinx:The Programmable Gate Array Development System DATA BOOK, Xilinx, Inc,1990
- 19 Xilinx:XACT Development System Design Interface User's Guide, Xilinx, Inc,1991
- 20 David Winkel,Franklin Prosser:The Art of Digital Design,Englewood Cliffs, Prentice-Hall, Inc,1980
- 21 Frederick J. Hill,Gerald R.Peterson:Introduction to Switching Theory and Logical Design (Third Edition), John Wiley & Sons,1981
- 22 Joseph D.Greenfield:Practical Digital Design Using IC_s(Second Edition), Prentice-Hall, Inc,1983
- 23 Friedmen AD. Logical Design of Degital System,Pitman,1978
- 24 曹汉房.多输出逻辑函数简化方法的探讨.武汉:华中理工大学学报,第19卷第4期.1991
- 25 曹汉房,屈万里.实现数字调相系统模拟信号源的一种新方法.华中理工大学学报,第19卷,1991年增刊(I)
- 26 曹汉房,屈万里.数字超声电视测井实验模拟系统.武汉:华中理工大学学报,第19卷,1991年增刊(I)
- 27 赵保经主编.中国集成电路大全.北京:国防工业出版社,1986
- 28 张端主编.实用电子电路手册(数字电路分册).北京:高等教育出版社,1992