



“十二五”普通高等教育本科国家级规划教材



面向 21 世纪 课 程 教 材

Textbook Series for 21st Century

电子技术基础

数字部分

第六版

华中科技大学电子技术课程组 编

主 编 康华光

副主编 秦 鑫 张 林

高等教育出版社
HIGHER EDUCATION PRESS



“十二五”普通高等教育本科国家级规划教材



面
Text

电子技术基础

数字部分

第六版

华中科技大学电子技术课程组 编

主 编 康华光

副主编 秦 璞 张 林

DIANZIJISHU JICHU SHUZIBUFEN



高等教育出版社·北京

HIGHER EDUCATION PRESS BEIJING

内容简介

本书被评为“十二五”普通高等教育本科国家级规划教材。上一版为普通高等教育“十五”国家级规划教材，第四版为面向 21 世纪课程教材，荣获 2002 年全国普通高等学校优秀教材一等奖。

本次修订弱化了中规模集成芯片的应用，将组合与时序单元电路作为宏模型介绍，加强应用 FPGA、CPLD 进行数字系统设计的内容。加强低电源电压器件及其接口内容，消减 TTL 系列的内容。增加 CMOS 通用电路中小逻辑与宽总线内容的介绍。加强应用 Verilog 语言描述组合及时序单元电路的例题。

全书共 11 章，分别是：数字逻辑概论，逻辑代数与硬件描述语言基础，逻辑门电路，组合逻辑电路，锁存器和触发器，时序逻辑电路，半导体存储器，CPLD 和 FPGA，脉冲波形的变换与产生，数模与模数转换器，数字系统设计基础。附录中列出 EDA 工具 Quartus II 9.0 简介，电气简图用图形符号——二进制逻辑单元(GB/T 4728.12—1996)简介，常用逻辑符号对照表。

本书可作为高等学校电气类、电子信息类、自动化类等专业“数字电子技术基础”课程的教材，也可供相关工程技术人员参考。

图书在版编目(CIP)数据

电子技术基础·数字部分/康华光主编;华中科技大学电子技术课程组编.--6 版.--北京:高等教育出版社,2014.1

ISBN 978-7-04-038004-0

I .①电… II .①康…②华… III .①电子技术-高等学校-教材 IV . ①TN

中国版本图书馆 CIP 数据核字(2013)第 164562 号

策划编辑 韩颖	责任编辑 韩颖	封面设计 王雎	版式设计 马敬茹
插图绘制 尹莉	责任校对 胡美萍	责任印制 刘思涵	

出版发行	高等教育出版社	网 址	http://www.hep.edu.cn
社 址	北京市西城区德外大街 4 号		http://www.hep.com.cn
邮政编码	100120	网上订购	http://www.landraco.com
印 刷	唐山市润丰印务有限公司		http://www.landraco.com.cn
开 本	787mm×1092mm 1/16		
印 张	36.25	版 次	1979 年 3 月第 1 版
字 数	820 千字		2014 年 1 月第 6 版
购书热线	010-58581118	印 次	2014 年 1 月第 1 次印刷
咨询电话	400-810-0598	定 价	53.00 元

本书如有缺页、倒页、脱页等质量问题，请到所购图书销售部门联系调换

版权所有 侵权必究

物 料 号 38004-00

作 者 声 明

未经本书作者和高等教育出版社书面许可,任何单位和个人均不得以任何形式将《电子技术基础 数字部分》(第六版)中的习题解答后出版,不得翻印或在出版物中选编、摘录本书的内容;否则,将依照《中华人民共和国著作权法》追究法律责任。

第六版序

《电子技术基础(模拟、数字)》，是电子电气类专业的技术基础课程教材，该套教材自1979年春由高等教育出版社出版发行以来，深受广大读者的欢迎。根据当前电子技术发展的新形势，在第五版的基础上，推陈出新。如今电子技术发展的现实是，MOSFETs器件在电子产品中已占统治地位。为了适应这一发展形势，新版教材大力加强了MOSFET的相关内容。现就模拟和数字两部分提出如下的修订思路。

一、模拟部分

1. 运算放大器是模拟部分的核心内容。在第2章中，首先把它理想化，称之为理想运算放大器，实际的运放将在第7章(模拟集成电路)中讲述。这样的安排是为了让学生易于入门，分散难点，也为了让教师根据各专业的要求作相应的选择。讲完第2章后，即可在教师的指导下进行运放的基本实验，可使学生对该课程产生兴趣，并有初步的成就感。
2. 由于半导体材料和器件制造工艺的进步，场效应管(MOSFETs)与双极结型三极管(BJTs)相比显示出新的优越性而获得较广泛的应用。考虑到历史和现实将第4、5两章写成相互独立的内容，教师可以自由选择其中任一章先讲，后讲的章节可以加快进度。
3. 频率响应一章，除一般知识外，可就MOSFETs和BJTs的相关电路有选择性地讲述。例如重点介绍MOS管及共源放大电路的高低频响应，最后介绍扩展频带的方法。
4. 模拟集成电路一章的内容丰富，可有选择性地讲述MOSFETs和BJTs的相关电路。至于运算放大器，也可按同样方法处理。

5. 反馈电子电路是电子电路的重要内容，通过大量的例题和习题以阐明负反馈的基本概念与分析方法，对反馈电路的稳定问题也作了简明分析。

二、数字部分

1. 现代数字电路和系统基本上不再使用中规模集成芯片搭建，而是采用CPLD或FPGA实现，甚至将系统集成在单一芯片上。其设计过程是将组合与时序单元电路作为基本模块由高层调用。因此，教材力求在弱化中规模集成芯片应用的同时，将组合与时序单元电路作为宏模型介绍。
2. 便携设备的发展要求CMOS集成电路的电源电压越来越低，导致低电压、超低电压器件的广泛使用。教材加强了低电源电压器件及其接口内容介绍，同时削减了TTL系列的内容。
3. 增加了CMOS通用电路中小尺寸逻辑与宽总线内容的介绍。小尺寸逻辑芯片是用来修改完善大规模集成芯片之间连线或外围电路的。与中规模器件相比，体积更小，速度更快。宽总线芯片是为满足计算机总线驱动而产生的。

4. 为了便于学生掌握 Verilog 描述单元电路的方法, 加强了 Verilog 描述组合及时序单元电路的例题。

5. 当用指定器件实现电路设计时, 力求成本低、速度快。介绍了 EDA 工具实现优化设计时, 需要用到多乘积项的共用、提取公因子、函数分解等方法。

6. 增加了时钟同步状态机的同步问题。当数字系统的结构复杂、工作速度快时, 时钟同步问题也越来越突出。由时钟偏移等问题引起触发器误翻转会造成系统的误动作。因此要在设计上避免这类问题的出现。

在本版修订工作中, 重新改编了例题、复习思考题和习题, 以利读者深入理解教材内容。SPICE 部分和 Verilog 语言部分的内容, 供各校师生灵活选用。

参加本版模拟部分修订工作的有张林(第 1、3、12 章及附录 A、B、C)、王岩(第 2、7、11 章)、陈大钦(第 4、9、10 章)、杨华(第 5、6、8 章)等。参加数字部分修订工作的有秦臻(第 1、3、4、11 章及附录 B、C)、罗杰(第 2 章及附录 A)、瞿安连(第 5、6 章)、张林(第 7、8 章)、彭容修(第 9、10 章)。康华光为主编, 负责全书的策划、组织和定稿。陈大钦、张林为模拟部分的副主编; 秦臻、张林为数字部分的副主编, 协助主编工作。此外, 张林还完成了模拟电路的 SPICE 分析; 罗杰还完成了数字电路的 Verilog 语言描述。

电子技术基础是一门实践性很强的课程, 与本教材配套的实验教材是由高等教育出版社出版的, 陈大钦、罗杰主编的《电子技术基础实验》。

本书由哈尔滨工业大学蔡惟铮教授主审, 参加审阅的还有王淑娟教授、杨春玲教授和王立欣教授。他们认真审阅了本书, 提出了不少中肯的修改意见, 在此表示衷心的感谢。第五版发行期间, 承全国各兄弟院校师生给我们以鼓励, 寄来了不少宝贵意见和建议, 编者在此一并致以谢忱。

康 华 光

2013 年 5 月于武汉华中科技大学

第五版序

当代电子技术的迅速发展,为人们的文化、物质生活提供了优越的条件,数码摄像机、家庭影院、空调、电子计算机等,都是典型的电子技术应用实例,可谓琳琅满目、异彩纷呈。至于电子技术在科技领域的应用,更是起着龙头作用,例如通信工程、测控技术、空间科学等比比皆是。而计算机的普及,也为大学生们提供了良好的学习平台。

本版是在前版的基础上修订而成,在修订过程中,参考了教育部组织编写的《电子技术基础(A)课程基本要求》,提出了如下的思路:精选内容,推陈出新;讲清基本概念、基本电路的工作原理和基本分析方法。对于较简单的电路,可用手工的方法进行近似计算;对于较复杂的电路,则可利用计算机及相应的软件进行仿真分析和设计。具体考虑有如下几点:

1. 简述信号与电子系统的概念,为学习模拟电路和数字电路提供引导性的背景知识。
2. 由于微电子学与制造工艺的进步,特别是在数字电路中,与双极型器件的性能相比,MOS 器件具有明显的优势。
3. 在模拟电路中增加了器件建模的内容,并利用 SPICE 软件对电路作具体的仿真分析与设计。在数字电路中增加了用 Verilog 语言建模的内容,借助 Quartus II 集成开发软件对电路进行仿真分析与设计。

现在,硬件与软件之间的界限越来越模糊,模拟电路或数字电路均属硬件,在利用软件对电路进行辅助设计时,不能轻视硬件,应引导学生全面发展。

4. 重新改编了例题、复习思考题和习题,以便读者深入理解教材内容。SPICE 部分和 Verilog 语言部分的内容,供各院校师生灵活选用。

参加本版模拟部分修订工作的有张林(第 1、3、11 章及附录 A、B、C)、王岩(第 2、6、10 章)、杨华(第 4、7 章)、陈大钦(第 5、8、9 章)等。参加数字部分修订工作的有秦臻(第 1、3、4、10 章及附录 A、C、D)、罗杰(第 2 章及附录 B)、瞿安连(第 5、6 章)、张林(第 7 章)、彭容修(第 8、9 章)等。康华光为主编,负责全书的策划、组织和定稿。陈大钦、张林为模拟部分的副主编,邹寿彬、秦臻为数字部分的副主编,协助主编工作。此外,张林还完成了模拟电路的 SPICE 分析;罗杰还完成了数字电路的 Verilog 语言描述。

本书由哈尔滨工业大学蔡惟铮教授主审,参加模拟部分审阅的为王淑娟教授,数字部分审阅的为杨春玲教授,在此表示衷心的感谢。第四版发行期间,承全国各兄弟院校师生给我们以鼓励,寄来了不少宝贵意见和建议,编者在此一并致以谢忱。

康华光

2005 年 7 月于武汉华中科技大学

第四版序

在电子技术日新月异的形势下,为了培养面向 21 世纪的电子技术人才,本书在第三版的基础上,经过教学改革与实践,对其内容作了较大的修改和更新,使之更符合电子信息时代的要求。在修订过程中,依照 1995 年教育部(原国家教委)颁发的《高等工业学校电子技术基础课程教学基本要求》,提出了如下的总思路:精选内容,推陈出新;讲清基本概念、基本电路的工作原理和基本分析方法。对主要的技术指标,采用工程近似方法进行计算,至于更全面的分析与设计则可借助 CAD 技术来实现。这将有利于读者开拓思路。具体考虑有如下几点:

1. 加强电子系统与信号的概念,为学习模拟电路和数字电路提供了引导性的背景知识。
2. 增加了部分新器件的内容,如砷化镓场效应管(MSFET)、VMOS 功率器件、BiCMOS 门电路、现场可编程逻辑器件(如 CPLD、FPGA)等,以适应新技术发展的需要。
3. 将三端有源器件(BJT、FET)的 6 种电路组态(共射、共集、共基和共源、共漏、共栅)归结为 3 种通用的电路组态,即反相电压放大器、电压跟随器和电流跟随器,这就有利于电子电路的分析与综合,也为学习和使用 BiFET 和 BiCMOS 等一类新型集成电路器件奠定了基础。
4. 根据当前教学上的需要与设备条件的可能性,模拟部分增设了“电子电路的计算机辅助分析与设计”一章;为电子电路仿真与设计作了引导性的介绍,并利用 PSPICE 软件解题;数字部分增设了“数字系统设计基础”一章,将硬件描述语言 ABEL 和 ISP Synario 软件作为数字系统设计的入门性工具,并附有设计实例。
5. 为便于读者深入理解教材内容,加强了例题,其中部分电路具有实用性。同时也重编了具有启发意义的复习思考题和习题,并附有少量的 CAD 例题和习题供各校师生灵活选用。

参加本版模拟部分修订工作的有瞿安连(第 1 章)、康华光(第 2、3、7 章)、陈大钦(第 4、5、8、9 章)、王岩(第 6、8、10 章)、张林(第 11 章及附录)等同志。参加数字部分修订工作的有康华光(第 1、2 章及附录 A、B、C、E、F、G)、邹寿彬(第 3、4、5 章)、杨华(第 6、7 章)、张林、李玲(第 8 章)、彭容修(第 9、10 章)、秦臻(第 11 章)、罗杰(附录 D 和第 8 章的部分内容)。康华光同志为主编,负责全书的策划、组织和定稿。陈大钦和邹寿彬同志分别为模拟部分和数字部分的副主编,协助主编工作。此外,杨华同志负责重编了模拟部分第 2、3、7 章的习题和第 1 章的校订工作;张林和罗杰二位同志协助有关各章的编者,完成了全书的 CAD 例题和习题的解答工作。

本书由东南大学衣承斌教授主审,参加审阅的,模拟部分为刘京南教授、李桂安副教授;数字

/ II / 第四版序

部分为皇甫正贤教授、戴义宝副教授。第三版发行期间，承全国各兄弟院校师生给我们以鼓励，寄来了不少宝贵意见和建议，编者在此一并致以衷心的谢意。

编 者

1998年7月于武汉华中理工大学

第三版序

自本书第一版问世以来,已经历了近十年。在这期间,电子技术领域发生了迅猛而巨大的变化。新技术革命和教学改革的不断深入,促使本教材不断改进完善,第三版现在与读者见面了。

新版是在第二版的基础上,经过改革试验、总结提高、修改增删而成的。在修订工作中,依照1987年经国家教委批准的《高等工业学校电子技术课程教学基本要求》,在保证基本教学内容的前提下,为适应电子技术不断发展的新形势和教学上的灵活性以及因材施教的需要,本版适当增加了部分加宽加深的选讲内容,具体考虑如下:

1. 新版在体系上作了较大的调整。在模拟部分中,将“模拟集成电路”一章的位置提前,以致有可能在“反馈放大器”以及后续各章中,均以模拟集成电路为对象进行讨论,这就形成了以模拟集成电路为主干的体系。数字部分则直接以小规模数字集成电路引路,逐步向中大规模集成电路深入,几乎大部分内容都纳入“组合逻辑”和“时序逻辑”两大类电路之中。
2. 在保证基本理论完整性的原则下,删去或精简了一些分立元件电路内容,增强了集成电路的应用,并引入模拟乘法器、开关电容滤波器、压控振荡器、锁相环、直流变换器、门阵列、算术逻辑单元、动态存储器、集成A/D与D/A转换器等新技术内容。
3. 为了开拓学生的知识广度,新增了“调制与解调”一章。
4. 本书数字部分的内容安排与讲述方法,注意到了与“微处理器基础”的密切联系,以利于压缩学时,提高教学效果。
5. 为了贯彻理论联系实际的原则,书中以不同的方式,安排了一定数量的电路实例,并注意阅读电子电路图和查阅电子器件手册的训练。
6. 教材正文与例题、习题紧密配合。例题是正文的补充。某些内容则有意地让读者通过习题来掌握,以调节教学节律,利于理解深化。
7. 在编排上,对于加宽加深的内容,均注有*号,以便于教师选讲和读者自学参考。

本版仍沿用从模拟到数字的体系,若有需要,亦可按从数字到模拟的体系讲授,只需将模拟部分的“半导体二极管和三极管”一章移到数字部分之前讲授即可。

参加新版模拟部分修订工作的有汤之璋(第1章)、康华光(第1、2、6、7章)、王岩(第5、8、11章)和陈大钦(第3、4、8、9、10章及附录A)等同志。参加数字部分修订工作的有康华光(第1、2章)、邹寿彬(第3、4、7章)和赵德宝(第5、6章及附录A)等同志。康华光同志为主编,负责全书的组织和定稿。陈大钦和邹寿彬同志分别为模拟和数字部分的副主编,协助主编工作。在修订过程中,得到了汤之璋教授的支持与帮助。赵德宝、瞿安连、肖锡湘同志协助校订了模拟部分的原稿。陈大钦、瞿安连同志协助校订了数字部分的原稿。丁素芳、罗杰、杨晓安和汪菊华等同志

/ II / 第三版序

绘制了全书的插图。教研室的其他同志也参加了部分工作。

本书由南京工学院李士雄教授主审，负责组织审稿工作的为衣承斌副教授，参加审阅的，模拟部分为衣承斌、陈黎明、陈天授副教授，李桂安讲师；数字部分为丁康源副教授，郑虎申、严振祥、皇甫正贤讲师。在第二版发行期间，承全国许多师生给我们以鼓励，寄来了不少宝贵意见和建议，编者谨此一并致以谢忱。

本版虽有所改进提高，但离教学改革的要求尚远。敬希读者予以批评指正。

编 者

1987年8月于武昌华工园

第二版序

本书是在第一版的试用基础上，并按照高等工业学校《电子技术基础教学大纲》（草案）（四年制自动化类和电力类专业试用），总结提高、修改增删而成的。主要做了下列几方面的工作：（1）从本课程的目的和任务出发，在保证打好基础的前提下，精选了内容，例如删去了“电子电路的计算机辅助分析”一章，适当精简了器件内部的物理过程、放大器的频率特性分析、分立元件电路以及设计方面的内容等，在篇幅上有较大的缩减；（2）删繁就简改写了第二、四、六章的大部分内容。同时，将第一版的第九、十章各分为两章，以利于教学；（3）增加了部分新内容，如集成运算放大器的应用电路，中规模数字集成电路等；（4）加强了电路分析方法，如用“虚短”的概念分析集成运算放大器的线性应用电路；在数字电路中，突出了组合逻辑与时序逻辑电路的分析方法；（5）近几年来，由于大规模集成电路的飞速发展，出现了微处理机对各个科学技术领域的渗透，为此，我们充实了“MOS 数字集成电路”一章的内容；（6）重新整理并增删了各章所附的思考题和习题。此外，在编排上，把基本内容排大字，选讲内容排小字，自学参考内容既排小字，又带*号。

本版各章基本上由原编者修订，参加的人员有汤之璋、康华光、陈婉儿、王岩、陈大钦、邹寿彬、朱立群等同志，全书由康华光同志定稿。在修订过程中，得到了汤之璋教授的帮助与指导，陈婉儿同志协助校阅了第一至第六章的书稿，肖锡湘、陈晓天、丘小云、石友惠、罗玉兰以及其他同志参加了许多工作。

本书由南京工学院李士雄教授主审，参加审阅工作的还有陈天授、陈黎明、皇甫正贤、郑虎申等同志；在本书第一版的试用期间，承全国有关兄弟院校的师生寄来不少宝贵意见和建议，编者在此深表谢忱。

本版内容虽有所改进，但离教学要求尚有差距，恳请使用本教材的师生和其他读者予以批评指正，以便不断提高。

编 者

1982年10月于武汉

初 版 序

本书是根据高等学校工科基础课电工、无线电类教材编写会议(1977年11月合肥会议)所制订的“电子技术基础”(电力类)教材编写大纲编写的。在编写过程中,我们力图以马列主义、毛泽东思想为指导,运用辩证唯物主义观点和方法来阐明本学科的规律。

“电子技术基础”是电力工程类各专业的一门技术基础课,它是研究各种半导体器件的性能、电路及其应用的学科。从本学科内容大的方面来划分,本书上、中两册属模拟电子技术,下册属数字电子技术;前者主要是讨论线性电路,后者则着重讨论脉冲数字电路。

教材中注意总结我们近年来的教学实践经验,加强了基础理论,如加强了半导体的物理基础和电路的基本分析方法;同时也注意吸取国内外的先进技术,如加强了线性集成电路和数字集成电路(包括中、大规模集成电路)的原理和应用,新增了电子电路的计算机辅助分析等内容。

在内容的安排上,注意贯彻从实际出发,由浅入深、由特殊到一般、从感性上升到理性等原则。通过各种半导体器件及其电路来阐明电子技术中的基本概念、基本原理和基本分析方法。对于基本的和常用的半导体电路(包括脉冲数字电路),除了作定性的分析外,还介绍了工程计算或设计方法。为了加深对课堂知识的理解,列举了若干电路实例,并配有一定数量的例题、思考题和习题。

在使用本教材时,请注意下列几点:

(1) 本课程是在学完普通物理学和电工原理的大部分内容之后开设的,课程之间的相互配合和衔接非常重要。例如,在第一章用能带理论来解释半导体内两种载流子——电子和空穴的导电规律时,应以普通物理学中讲的固体能带理论为基础;又如在分析放大器时,既讨论了稳态分析(频域),也介绍了瞬态分析(时域),在“运算放大器”一章中,又有积分、微分电路以及其他应用,这些内容应以电工原理中的无源线性电路的瞬态分析为基础,只有配合得好,才能取得满意的效果。

(2) 本教材是按课程总学时数约200(包括实验课等环节)而编写的,除了基本内容之外,还编入了部分较深入的内容,这些内容均在标题前注有星号(*)或用小字排印,自成体系。不同专业可按学时多少,由教师灵活选择,也可供读者自学参考。

(3) 课程中各个教学环节的配合十分重要,除了课堂讲授外,还必须通过习题课和实验课等环节加以补充,有些内容可以把这几个环节有机地结合起来。对于实验课,必须予以高度重视,通过实验课,不仅可以验证理论,加深对理论知识的理解,更重要的是,可以学会电子测试技术,使理论紧密结合实践。

参加本书编写工作的有汤之璋(第一章)、陈婉儿(第一、二、九章)、陈大钦(第三、五、十

章)、康华光(第四、十一章)、王岩(第六、七、十三章)、林家瑞(第六章)、邹寿彬(第八、十二章)、周劲青(第十一章)和江庚和(第十三章)等同志,最后由康华光同志定稿。在编写过程中,张瑾、朱立群、赵月怀、肖锡湘、杨华、石友惠、汪菊华、罗玉兰以及其他同志参加了许多工作,给予很大支持。

本书由南京工学院李士雄副教授主审,参加主审工作的还有江正战、张志明、衣承斌、陈黎明和丁康源等同志。

在武汉和南京举行的审稿会上,承西安交通大学沈尚贤教授、清华大学童诗白教授、浙江大学邓汉馨副教授、上海交通大学徐俊荣副教授以及重庆大学、山东工学院、沈阳机电学院、合肥工业大学、大连工学院、湖南大学、华南工学院、同济大学、哈尔滨工业大学、天津大学、太原工学院和昆明工学院等兄弟院校的教师代表对初稿进行了认真的审阅,并提出了许多宝贵的意见。

在编写本书第八章(电子电路的计算机辅助分析)的过程中,承中国科学院湖北岩体土力学研究所计算机室协助解题。

对所有为本教材进行审阅并提出宝贵意见以及在编写出版过程中给予热情帮助和支持的同志们,我们在此一并表示衷心的感谢。

由于我们的水平有限,加之时间比较仓促,书中错误和不妥之处,在所难免,殷切希望使用本教材的师生及其他读者,给予批评指正。

编 者
1979年3月

本书常用符号表

A_0, A_1, A_2, \dots	第 0、1、2、…位译码器地址输入	$EI; EO$	使能输入；使能输出
$A > B, A = B, A < B$	数字比较器大于、等于、小于	FF_n	时序电路中触发器 n
BCD	二-十进制码	f_A	异步输入信号翻转的平均频率
C	传输门高电平控制信号	f_{CP}	时钟频率
CAS	动态存储器列地址选通信号	f_{max}	最高时钟频率
CE	存储器片选输入端	G	逻辑门
CEP	计数器并行进位允许输入端	G	进位产生变量
CET	计数器进位允许输入端	I	时序电路的输入信号向量
CP	时钟脉冲	I_{BS}	临界饱和基极电流
CP_n	时序电路中第 n 个触发器的时钟信号	i_s	基极电流瞬时值(交、直流混合量)
cp_n	异步时序电路第 n 个触发器的时钟有效标志	I_{CS}	集电极饱和电流
CR	清零输入端	i_c	集电极电流瞬时值
CS	片选信号输入	i_b	漏极电流瞬时值
C_{re}	外接电容端	I_m	高电平输入电流
C_{rd}	CMOS 电路的功耗电容	I_u	低电平输入电流
C_L	负载电容	I_{on}	高电平输出电流
\bar{C}	传输门低电平控制信号	I_{ol}	低电平输出电流
D	数据	I_{oz}	流过截止 MOS 管的漏电流
D	数据输入	J	JK 触发器的 J 输入端
D	D 锁存器或触发器的 D 输入端	K	JK 触发器的 K 输入端
DP	延时-功耗积	K_n	MOS 管电导常数
D_{po}	移位寄存器并行输出	k'_n	MOS 管制造工艺常数
D_{si}	移位寄存器串行输入端	LE	锁存允许输入端
D_{sl}	移位寄存器左移串行输入端	$MTBF_{sync}$	同步器的平均失效间隔时间
D_{so}	移位寄存器串行输出端	N_f	扇入数
D_{sr}	移位寄存器右移串行输入端	N_o	扇出数
E	使能控制端	N_{sync}	每秒进入介稳态的平均次数
E	锁存器使能输入端	O	时序电路的输出信号向量
E	时序电路激励信号向量	OE	储存器输出使能输入端
EC	计数允许输入端	PE	并行置数允许输入端
		P_0	功耗
		P_T	CMOS 管导通功耗

P_t	CMOS 管带容性负载的动态功耗	t_{pd}	低电平到高电平的传输延迟时间
Q	锁存器或触发器的输出端	t_{pd}	平均传输延迟时间
Q^*	触发器的现态	t_{su}	建立时间
Q^{**}	触发器的次态	t_s	脉冲宽度
q	占空比	t_r	上升时间
R	SR 锁存器或 SR 触发器的复位(置 0)输入端	V_{RE}	三极管发射结电压
RAS	动态储存器行地址选通信号	V_{CES}	TTL 电路电源电压
$RESET$	时序电路的复位信号	V_{CE}	BJT 的饱和压降
R_b	锁存器或触发器的直接复位(置 0)	V_{DD}	三极管集电结电压
	输入端	V_{DS}	CMOS 电路电源电压
r_{ds}	MOS 管漏源之间的交流等效电阻	V_{GS}	MOS 管漏源之间电压
R_L	负载电阻	V_I	MOS 管栅源之间电压
R_m	MOS 管导通时的电阻	v_i	输入电压瞬时值
R_p	OD 门外接的上拉电阻	V_{HS}	介稳态点电压
S	SR 锁存器或 SR 触发器的置位(置 1)输入端	V_{NH}	高电平噪声容限电压
S	时序电路的状态向量	V_{NL}	低电平噪声容限电压
S_b	锁存器或触发器的直接置位(置 1)	V_o	输出电压
	输入端	V_{OH}	输出电压瞬时值
S^*	时序电路的现态	V_{OL}	输出高电平时的电压
S^{**}	时序电路的次态	V_{REF}	输出低电平时的电压
T	周期	V_p	参考电压
T	T 触发器的 T 输入端	V_{sw}	耗尽型 MOS 管夹断电压
T_A	异步输入信号翻转的平均周期	V_T	逻辑摆幅
TC	计数器进位输出	V_T	增强型 MOS 管开启电压
T_{cr}	时钟周期	V_{TH}	ECL 电路牵引电压
T_N	N 沟道 MOSFET	V_{Tr+}	阈值电压
T_P	P 沟道 MOSFET	V_{Tr-}	施密特触发特性的正向阈值电压
t	时间	WE	施密特触发特性的负向阈值电压
t_{compl}	同步时序电路中组合电路的延迟时间	W/L	储存器写使能输入端
		\times	MOS 管沟道的宽长比
t_{tripd}	同步时序电路中触发器的延迟时间	δ	任意态, 无关项
t_H	保持时间	τ	同步时序电路的时钟偏移量
t_{lstd}	门延迟时间		时间常数
t_f	下降时间		电平触发信号
t_{pt}	时钟抖动时间		上升沿触发信号
t_{pdm}	高电平到低电平的传输延迟时间		下降沿触发信号

郑重声明

高等教育出版社依法对本书享有专有出版权。任何未经许可的复制、销售行为均违反《中华人民共和国著作权法》，其行为人将承担相应的民事责任和行政责任；构成犯罪的，将被依法追究刑事责任。为了维护市场秩序，保护读者的合法权益，避免读者误用盗版书造成不良后果，我社将配合行政执法部门和司法机关对违法犯罪的单位和个人给予严厉打击。社会各界人士如发现上述侵权行为，希望及时举报，本社将奖励举报有功人员。

反盗版举报电话 (010)58581897 58582371 58581879

反盗版举报传真 (010)82086060

反盗版举报邮箱 dd@ hep. com. cn

通信地址 北京市西城区德外大街 4 号 高等教育出版社法务部

邮政编码 100120

目 录

1 数字逻辑概论	1
1.1 数字信号与数字电路	1
1.1.1 数字技术的发展及其应用	1
1.1.2 数字集成电路的分类及 特点	3
1.1.3 模拟信号和数字信号	7
1.1.4 数字信号的描述方法	7
1.2 数制	12
1.2.1 十进制	12
1.2.2 二进制	13
1.2.3 十-二进制之间的转换	15
1.2.4 十六进制和八进制	18
1.3 二进制数的算术运算	20
1.3.1 无符号二进制数的算术 运算	20
1.3.2 带符号二进制数的减法 运算	22
1.4 二进制代码	25
1.4.1 二-十进制码	25
1.4.2 格雷码	26
1.4.3 ASCII 码	28
1.5 二值逻辑变量与基本逻辑 运算	29
1.6 逻辑函数及其表示方法	33
1.6.1 逻辑函数的几种表示方法	33
1.6.2 逻辑函数表示方法之间的 转换	35
小 结	37
习 题	37
2 逻辑代数与硬件描述语言基础	41
2.1 逻辑代数的基本定律和规则	41
2.1.1 逻辑代数的基本定律和恒 等式	41
2.1.2 逻辑代数的基本规则	44
2.2 逻辑函数表达式的形式	45
2.2.1 逻辑函数表达式的基本 形式	45
2.2.2 最小项与最小项表达式	46
2.2.3 最大项与最大项表达式	47
2.3 逻辑函数的代数化简法	50
2.3.1 逻辑函数的最简形式	50
2.3.2 逻辑函数的代数化简法	51
2.4 逻辑函数的卡诺图化简法	53
2.4.1 用卡诺图表示逻辑函数	54
2.4.2 用卡诺图化简逻辑函数	56
2.5 硬件描述语言 Verilog HDL 基础	60
2.5.1 Verilog 语言的基本语法 规则	61
2.5.2 变量的数据类型	63
2.5.3 运算符及其优先级	64
2.5.4 Verilog 内部的基本门级 元件	67
2.5.5 Verilog 程序的基本结构	69
2.5.6 逻辑功能的仿真与测试	73
小 结	73
习 题	74
3 逻辑门电路	78

/ II / 目录

3.1 逻辑门电路简介	78	3.8.1 各系列逻辑门电路之间的 接口问题	124
3.1.1 各种逻辑门电路系列简介	78	3.8.2 逻辑门电路驱动其他负载 时的接口	129
3.1.2 开关电路	80	3.8.3 抗干扰措施	131
3.2 基本 CMOS 逻辑门电路	81	3.8.4 CMOS 通用逻辑电路中的小 尺寸逻辑和宽总线系列	132
3.2.1 MOS 管及其开关特性	81	3.9 用 Verilog HDL 描述 CMOS 门 电路	136
3.2.2 CMOS 反相器	86	3.9.1 CMOS 门电路的 Verilog 建模	136
3.2.3 其他基本 CMOS 逻辑门 电路	90	3.9.2 CMOS 传输门电路的 Verilog 建模	137
3.2.4 CMOS 传输门	92	小结	138
3.3 CMOS 逻辑门电路的不同输出 结构及参数	94	习题	139
3.3.1 CMOS 逻辑门的保护和缓冲 电路	94	4 组合逻辑电路	148
3.3.2 CMOS 漏极开路门和三态输 出门电路	96	4.1 组合逻辑电路的分析	148
3.3.3 CMOS 逻辑门电路的重要技 术参数	102	4.1.1 组合逻辑电路的定义	148
3.4 类 NMOS 和 BiCMOS 逻辑门 电路	109	4.1.2 组合逻辑电路的分析 方法	148
3.4.1 类 NMOS 门电路	109	4.2 组合逻辑电路的设计	151
3.4.2 BiCMOS 门电路	111	4.2.1 组合逻辑电路的设计 过程	151
3.5 TTL 逻辑门电路	111	4.2.2 组合逻辑电路的优化 实现	155
3.5.1 BJT 的开关特性	111	4.3 组合逻辑电路中的竞争- 冒险	158
3.5.2 TTL 反相器的基本电路	112	4.3.1 产生竞争-冒险的原因	158
3.5.3 改进型 TTL 门电路—— 抗饱和 TTL 门电路	113	4.3.2 消去竞争-冒险的方法	159
3.5.4 TTL 系列门电路特性参数 比较	116	4.4 若干典型的组合逻辑电路	161
3.6 ECL 逻辑门电路	117	4.4.1 编码器	161
3.7 逻辑描述中的几个问题	119	4.4.2 译码器/数据分配器	166
3.7.1 正负逻辑问题	119	4.4.3 数据选择器	179
3.7.2 基本逻辑门的等效符号及其 应用	120	4.4.4 数值比较器	186
3.8 逻辑门电路使用中的几个实际 问题	123	4.4.5 算术运算电路	190
4.5 组合可编程逻辑器件	197		

4.5.1 PLD 的结构、表示方法及 分类	198	5.5.5 D 触发器逻辑功能的 转换	259
4.5.2 组合逻辑电路的 PLD 实现	202	5.6 用 Verilog HDL 描述锁存器 和触发器	260
4.6 用 Verilog HDL 描述组合逻 辑电路	206	5.6.1 时序逻辑电路建模基础	260
4.6.1 组合逻辑电路的行为级 建模	206	5.6.2 锁存器和触发器的 Verilog 建模实例	262
4.6.2 分模块、分层次的电路 设计	214	小 结	265
小 结	217	习 题	266
习 题	218		
5 锁存器和触发器	231		
5.1 基本双稳态电路	231	6 时序逻辑电路	273
5.2 SR 锁存器	232	6.1 时序逻辑电路的基本概念	273
5.2.1 基本 SR 锁存器	232	6.1.1 时序逻辑电路的基本结构 与分类	274
5.2.2 门控 SR 锁存器	237	6.1.2 时序逻辑电路功能的 表达	276
5.3 D 锁存器	239	6.2 同步时序逻辑电路的分析	282
5.3.1 D 锁存器的电路结构	239	6.2.1 分析同步时序逻辑电路的 一般步骤	282
5.3.2 典型的 D 锁存器集成 电路	241	6.2.2 同步时序逻辑电路分析 举例	282
5.3.3 D 锁存器的动态特性	243	6.3 同步时序逻辑电路的设计	289
5.4 触发器的电路结构和工作 原理	245	6.3.1 设计同步时序逻辑电路的 一般步骤	289
5.4.1 主从 D 触发器的电路结构 和工作原理	246	6.3.2 同步时序逻辑电路设计 举例	291
5.4.2 典型的主从 D 触发器集成 电路	247	6.3.3 同步时序逻辑电路中的 时钟偏移	302
5.4.3 主从 D 触发器的动态 特性	248	6.4 异步时序逻辑电路的分析	305
5.4.4 其他电路结构的触发器	250	6.5 若干典型的时序逻辑电路	310
5.5 触发器的逻辑功能	254	6.5.1 寄存器和移位寄存器	310
5.5.1 D 触发器	254	6.5.2 计数器	316
5.5.2 JK 触发器	255	6.6 简单的时序可编程逻辑器件	
5.5.3 T 触发器	257	GAL	331
5.5.4 SR 触发器	258	6.6.1 GAL 的结构	331
		6.6.2 GAL 中的输出逻辑宏 单元	333

6.6.3 GAL 的结构控制字	337	的设计例题	406
6.7 用 Verilog HDL 描述时序逻辑		小 结	408
电路	338	习 题	408
6.7.1 移位寄存器的 Verilog 建模	339	9 脉冲波形的变换与产生	410
建模	339	9.1 单稳态触发器	410
6.7.2 计数器的 Verilog 建模	340	9.1.1 用门电路组成的单稳态触 发器	410
状态图的 Verilog 建模	343	9.1.2 集成单稳态触发器	412
数字钟的 Verilog 建模	346	9.1.3 单稳态触发器的应用	417
小 结	349	9.2 施密特触发器	419
习 题	350	9.2.1 用门电路组成的施密特触 发器	419
7 半导体存储器	364	9.2.2 集成施密特触发器	422
7.1 只读存储器	364	9.2.3 施密特触发器的应用	423
7.1.1 ROM 的基本结构	365	9.3 多谐振荡器	425
7.1.2 二维译码与存储阵列	366	9.3.1 门电路组成的多谐振 荡器	425
7.1.3 可编程 ROM	367	9.3.2 用施密特触发器构成多谐 振荡器	427
7.1.4 ROM 读操作实例	368	9.3.3 石英晶体多谐振荡器	429
7.1.5 ROM 应用举例	371	9.4 555 定时器及其应用	431
7.2 随机存取存储器	373	9.4.1 555 定时器	431
7.2.1 SRAM	373	9.4.2 用 555 组成的施密特触 发器	432
7.2.2 同步 SRAM	378	9.4.3 用 555 组成的单稳态触 发器	434
7.2.3 DRAM	381	9.4.4 用 555 组成的多谐振 荡器	436
7.2.4 存储容量的扩展	384	小 结	438
7.2.5 RAM 应用举例	385	习 题	438
小 结	388	10 数模与模数转换器	445
习 题	388	10.1 D/A 转换器	445
8 CPLD 和 FPGA	391	10.1.1 D/A 转换器的输入/输出 特性及其结构框图	445
8.1 复杂可编程逻辑器件(CPLD)		10.1.2 D/A 转换器的基本原理	446
简介	391	10.1.3 倒 T 形电阻网络 D/A 转	
8.2 现场可编程门阵列(FPGA)	395		
8.2.1 FPGA 中编程实现逻辑			
功能的基本原理	395		
8.2.2 FPGA 的结构简介	397		
8.3 可编程逻辑器件开发过程			
简介	402		
8.4 用 EDA 技术和可编程器件			

换器	447	11.3.3 用可编程逻辑器件实 现交通灯控制系统	491
10.1.4 权电流型 D/A 转换器	450	11.4 数字密码锁	500
*10.1.5 权电容网络 D/A 转换器	452	11.4.1 数字密码锁的 ASM 图	500
10.1.6 D/A 转换器的输出方式	454	11.4.2 用典型电路基本模块实现 数字密码锁	503
10.1.7 D/A 转换器的主要技术 指标	456	11.4.3 用可编程逻辑器件实现数字 密码锁	506
10.1.8 D/A 转换器的应用	458	小结	509
10.2 A/D 转换器	460	习题	510
10.2.1 A/D 转换的一般工作 过程	460	附录 A EDA 工具 Quartus II 9.0	
10.2.2 并行比较型 A/D 转 换器	463	简介	513
10.2.3 逐次比较型 A/D 转 换器	464	A.1 Quartus II 9.0 软件主界面	513
10.2.4 双积分式 A/D 转换器	467	A.2 Quartus II 的设计流程	514
10.2.5 A/D 转换器的主要技术 指标	470	A.3 设计与仿真的过程	518
10.2.6 集成 A/D 转换器及其 应用	470	A.3.1 建立新的设计项目	518
小结	473	A.3.2 输入设计文件	520
习题	474	A.3.3 编译设计文件	521
11 数字系统设计基础	477	A.3.4 设计项目的仿真验证	522
11.1 数字系统概述	477	A.4 引脚分配与器件编程	526
11.1.1 数字系统的组成	477	A.4.1 引脚分配	526
11.1.2 数字系统的设计方法	478	A.4.2 对目标器件编程	528
11.1.3 数字系统的实现	480	附录 B 电气简图用图形符号——二进 制逻辑单元(GB/T 4728.12— 1996)简介	532
11.2 算法状态机	481	B.1 二进制逻辑单元图形符号的 组成	532
11.2.1 ASM 图形符号	481	B.2 限定性符号	533
11.2.2 ASM 图与状态图	483	B.3 关联标注法	536
11.3 交通信号灯控制系统	486	附录 C 常用逻辑符号对照表	538
11.3.1 交通信号灯控制系统 ASM 图	486	部分习题答案	540
11.3.2 用典型电路基本模块 实现交通灯控制系统	488	索引(汉英对照)	547
		参考文献	552

1

>>>

数字逻辑概论

引言

随着现代电子技术的发展,人们正处于一个信息时代。每天都要通过电视、广播、通信、互联网等多种媒体获取大量的信息。而现代信息的存储、处理和传输越来越趋于数字化。在人们的日常生活中,常用的计算机、电视机、音响系统、视频记录设备、远距离通信等电子设备或电子系统中,无一不采用数字电路或数字系统。因此,数字电子技术的应用越来越广泛。

本章首先介绍数字技术的发展及应用、数字集成电路的分类及特点、模拟信号和数字信号以及数字信号的描述方法。然后讨论数制、二进制的算术运算、二进制代码和数字逻辑的基本运算。

1.1 数字信号与数字电路

1.1.1 数字技术的发展及其应用

电子技术是 20 世纪以来发展最迅速、应用最广泛的技术,已经渗透到人类生活的各个方面。特别是数字电子技术,取得了令人瞩目的进步。

电子技术的发展是以电子器件的发展为基础的。20 世纪初直至中叶,主要使用真空管,也称电子管。随着固体微电子学的进步,第一只晶体管于 1947 年问世,开创了电子技术的新领域。随后,在 20 世纪 60 年代初,模拟和数字集成电路相继上市。20 世纪 70 年代微处理器的问世,使电子器件及其应用出现了崭新的局面。20 世纪 80 年代末,微处理器每个芯片的晶体管数目突破百万大关。到 20 世纪 90 年代末,可以制造包含千万个晶体管的芯片。当前的制造技术可以在芯片上集成几十亿个晶体管。在过去的 40 年间,集成电路的集成度和性能以惊人的速度发展着,印证了摩尔定理,即单个芯片上集成的晶体管数量每 18 个月就翻一番。以集成电路为核心的电子信息产业已超过了汽车、石油和钢铁工业成为第一大产业。

数字技术应用的典型代表是电子计算机,它是伴随着电子技术的发展而发展的,经历了电子管、晶体管、集成电路和超大规模集成电路四个发展阶段,其体积越来越小,功能越来越强,价格越来越低,应用范围越来越广,目前正朝着智能化方向发展。计算机技术的影响已遍及人类经济生活的各个领域,掀起了一场“数字革命”。数字技术被广泛地应用于广播、电视、通信、医学诊断、测量、控制、文化娱乐以及家庭生活等方面。由于数字信息具有便于存储、处理和传输的特点,使得许多传统使用模拟技术的领域转而运用数字技术。

音频信息存储 20世纪90年代以前,声音的存储主要以模拟信号的方式将声波记录在唱片或磁带上,而两者的携带和保存都不方便。现在是将声音转换成数字信号存储在CD(Compact Disc)上。存储音乐的CD通常使用的取样频率为44.1kHz,量化位数为16位,同时有两股声音被录制(分别流向立体声系统的两个扬声器),而且可以存储长达74 min的音乐。这样一张CD盘上储存的数字信息总量超过700 MB。CD盘可用于存储不同格式的数据,最常见的格式是音频数据和计算机数据。

视频信息存储 DVD(Digital Video Disk或Digital Versatile Disk)普及之前,早期的视频信息存储主要以记录模拟信号的录像带为主,而录像带不便于保存和传输。DVD与CD外观类似,但在数据存储技术、数据容量及功能等方面都有本质的区别。CD存储无数据压缩的音频信息,而DVD采用MPEG2的压缩技术,以数字格式存储音频和视频信息。DVD可以分为单面单层、单面双层、双面单层和双面双层四种物理结构,其数据容量非常大,画质和音质好。仅单面单层、直径12 cm的光盘就能存储4.7 GB数据、影片播放时间可达133 min。双面双层存储的数据可高达17 GB之多,影片播放时间可达8 h以上。因此,DVD已成为家庭影院的重要组成部分。

MPEG(Moving Picture Experts Group)是世界数字视频和音频压缩比的标准化组织制定的,用于多媒体运动图像和伴音的数据压缩编码的国际标准。MPEG标准主要包括MPEG1、MPEG2、MPEG4、MPEG7和MPEG21等,以适用于不同带宽和数字影像质量的要求。MPEG的压缩比,最高可达200:1,而且在提供高压缩比的同时,对数据的损失很小。MPEG标准改变了以模拟方式记录声音和影像的传统方法,令视听传播进入到数字化时代。

照相机 传统的模拟相机是用卤化银感光胶片记录影像,胶片成像过程需要严格的加工工艺和技术,而且胶片不便于保存和传输。数码相机是将影像的光信号转换为数字信号,以像素阵列的形式进行存储。存储的信息包括色彩、光强度和位置等。例如 1024×768 的像素阵列中,每个像素的红、绿、蓝三原色均是8位,则该阵列的数据位数超过1800万。如果用JPEG图形格式进行压缩处理,压缩比率采用20:1,处理后的数据量只为原来的5%,便于进行网络的远距离传输。如今,数码相机已完全取代模拟胶片相机。

JPEG(Joint Photographic Experts Group)是国际标准化组织(International Standard Organization,ISO)和国际电报电话咨询委员会(Consultative Committee for International Telegraph and Telephone,CCITT)联合制定的静止图像压缩编码标准。JPEG有多种压缩级别,压缩比率通常在10:1到40:1之间,是目前较流行的静止图像压缩文件格式。

交通灯控制系统 交通灯于1920年问世。早期的红黄绿灯是用机电定时器控制的。后来

使用由电子线路和继电器构成的控制器,根据道路上传感器检测的信号进行控制,可靠性低,故障率高。现在的交通灯由计算机控制,可以将监测系统检测到的车辆流量信息送到系统计算机,经计算后进行合理的时间分配。如某路口东西方向堵塞,则将该路口东西方向的绿灯自动延时,并将附近区域东西方向的红灯也自动地延时,堵塞解除后,信号灯恢复正常状态。

随着微电子技术的发展,将会有更多的数字电子产品陆续问世。数字技术的发展、计算机的应用正在改变着人类的生产方式、生活方式及思维方式,它使得工业自动化、农业现代化、办公自动化和通信网络化成为现实。但是,无论数字技术如何发展,也不能代替模拟技术。自然界中绝大多数物理量都是模拟量,数字技术不能直接接受模拟信号进行处理,也无法将处理后的数字信号直接送到外部物理世界。因此,模拟技术在电子系统中是不可缺少的。由于模拟技术难度远高于数字技术,其发展自然较慢。实际电子系统一般是模拟电路和数字电路的结合,在发展数字技术的同时,也应重视模拟技术的发展。

1.1.2 数字集成电路的分类及特点

电子电路按功能分为模拟电路和数字电路。根据电路的结构特点及其对输入信号的响应规则的不同,数字电路可分为组合逻辑电路和时序逻辑电路。数字电路中的电子器件,如二极管、三极管(FET、BJT)处于开关状态,时而导通,时而截止,构成电子开关。这些电子开关是组成逻辑门电路的基本器件。逻辑门电路又是数字电路的基本单元。如果将这些门电路集成在一片半导体芯片上就构成数字集成电路。

1. 数字集成电路的分类

数字电路的发展历史与模拟电路一样,经历了由电子管、晶体管等半导体分立器件到集成电路的发展过程。由于集成电路的发展非常迅速,很快占有主导地位,因此,数字电路主流形式是数字集成电路。从20世纪60年代开始,数字集成器件以双极型工艺制成了小规模逻辑器件,随后发展到中规模;20世纪70年代末,微处理器的出现,使数字集成电路的性能发生了质的飞跃;从20世纪80年代中期开始,专用集成电路(Application Specific Integrated Circuit, ASIC)制作技术已趋成熟,标志着数字集成电路发展到了新的阶段;20世纪90年代中期,片上系统(System on Chip, SoC)设计技术可以将复杂的电子系统全部集成在单一芯片上,使集成电路设计向集成系统设计转变,预示着集成电路出现了从量变到质变的突破。

ASIC是根据用户特定要求和电子系统的特定需要而设计制造的专用集成电路。与通用集成电路相比具有体积小、重量轻、功耗低、速度高、成本低、保密性强的优点。ASIC芯片的制作可以采用全定制或半定制的方法。全定制适用于生产批量的成熟产品,由半导体生产厂家制造。对于生产批量小或研究试制阶段的产品,可以采用半定制方法。目前最为流行的半定制方法是用复杂可编程逻辑器件(Complex Programmable Logic Device, CPLD)和现场可编程门阵列(Field Programmable Gate Array, FPGA)来进行ASIC设计。用户通过软件编程,将自己设计的数字系统制作在厂家生产的CPLD或FPGA芯片上,便得到所需的系统级芯片。

SoC是将电子系统中所有不同的功能块集成在一个芯片中,称为片上系统,如手机芯片、数

字电视芯片、DVD 芯片等。SoC 芯片的规模远大于普通的 ASIC, 其设计方法进一步分工细化, 出现了 IP(Intellectual Property)设计和 SoC 系统设计。IP 内核模块是一种已经过验证的、可重利用的、具有某种确定功能的模块。将数字电路中常用的但比较复杂的功能块, 如微处理器、微控制器、滤波器、嵌入式存储器、数字信号处理器等设计成可修改参数的模块, 让用户直接调用这些模块, 这样可以大大减轻工程师的负担, 避免重复劳动。随着 CPLD/FPGA 的规模越来越大, 设计越来越复杂, 使用 IP 核是一个发展趋势。

衡量集成电路有两个主要的参数:集成度与特征尺寸。集成度是指每一芯片所包含的门的个数。特征尺寸是指集成电路中半导体器件加工的最小线条宽度。集成度与特征尺寸是相关的。当集成电路芯片的面积一定时,集成度越高,特征尺寸就越小。所以,特征尺寸也成为衡量集成电路设计和制造技术水平高低的重要指标。

在过去的几十年中,以硅为主要加工材料的微电子制造工艺从开始的微米级加工水平,经历亚微米级($0.8 \sim 0.35 \mu\text{m}$)、深亚微米级($0.25 \mu\text{m}$ 以下)技术的发展,到现在的纳米级($0.05 \mu\text{m}$ 以下)技术,集成电路芯片集成度越来越高,成本越来越低。

从集成度来说,数字集成电路可分为小规模(Small Scale Integration,SSI)、中规模(Medium Scale Integration,MSI)、大规模(Large Scale Integration,LSI)、超大规模(Very Large Scale Integration,VLSI)和甚大规模(Ultra Large Scale Integration,ULSI)五类。表 1.1.1 列出了数字集成电路的集成度分类。

表 1.1.1 数字集成电路的集成度分类

分类	门的个数	典型集成电路
小规模	最多 12 个	逻辑门、触发器
中规模	12 ~ 99	计数器、加法器
大规模	100 ~ 9999	小型存储器、门阵列
超大规模	10 000 ~ 99 999	大型存储器、微处理器
甚大规模	10^6 以上	可编程逻辑器件、多功能专用集成电路

逻辑门是数字集成电路的主要单元电路,按照结构和工艺分为双极型、MOS 型和双极-MOS 型。三极管-三极管(Transistor-Transistor Logic, TTL)逻辑门电路问世较早,其工艺经过不断改进,曾广泛应用于中小规模集成电路设计中,目前的使用范围大大缩小。随着金属-氧化物-半导体(MOS)工艺特别是 CMOS(Complementary Metal-Oxide-Semiconductor)工艺的发展,大大提高了电路的集成度和工作速度,且明显降低了功耗,因此 TTL 的主导地位已被 CMOS 器件所取代。

2. 数字集成电路的特点

与模拟电路相比,数字电路主要有下列优点。

- (1) 稳定性高,抗干扰能力强

一般而言,对于一个给定的输入信号,如果噪声信号不使其超过阈值,数字电路的输出状态不变。因此数字电路的工作可靠,稳定性好,抗干扰能力强。而模拟电路的输出则随着外界温度和电源电压的变化,以及器件的老化等因素而发生变化。

(2) 易于设计

数字电路又称为数字逻辑电路,它主要是对用 **0** 和 **1** 表示的数字信号进行逻辑运算和处理,不需要复杂的数学知识,广泛使用的数学工具是逻辑代数。数字电路只要能够可靠地区分 **0** 和 **1** 两种状态就可以正常工作。因此,数字电路的分析与设计相对较容易。

(3) 便于集成,成本低廉

数字电路便于集成化生产,成本低廉,体积小,通用性强,容易制造,进而使集成芯片广泛应用于数字电路。

(4) 可编程性

现代数字系统的设计,大多采用可编程逻辑器件,即厂家生产的一种半成品芯片。用户根据需要用硬件描述语言(Hardware Description Language, HDL)在计算机上完成电路设计和仿真,并写入芯片,这为用户研制开发产品带来了极大的方便和灵活性。

(5) 高速度,低功耗

随着集成电路工艺的发展,数字器件的工作速度越来越高,而功耗越来越低。集成电路中单管的开关速度可以做到小于 10^{-11} s。整体器件中,信号从输入到输出的传输时间小于 10^{-9} s。百万门以上超大规模集成芯片的功耗,可以低于毫瓦级。

(6) 便于存储、传输和处理

数字信号由 **0** 和 **1** 编码组成,可以用数字存储器对其进行存储、传输。将数字系统与计算机连接,便可利用计算机对数字信号进行处理和控制。

由于具有这些优点,可以肯定,数字电路在众多领域取代模拟电路的趋势将会继续发展下去。

3. 数字电路的分析、设计与测试

(1) 数字电路的分析方法

数字电路处理的是数字信号,电路中的半导体器件工作在开关状态,如晶体管工作在饱和区或截止区,所以不能采用模拟电路使用的小信号模型等方法进行分析。数字电路又称为逻辑电路,在电路结构、功能和特点等方面均不同于模拟电路,因而,其分析方法与模拟电路完全不同。数字电路的主要研究对象是电路的输出与输入之间的逻辑关系,所采用的分析工具是逻辑代数,表达电路输出与输入的关系主要用真值表、功能表、逻辑表达式或波形图。

随着计算机技术的发展,借助电子设计自动化(Electronic Design Automation, EDA)工具,可以更直观、更快捷、更全面地对电路进行分析。EDA 软件可以用于模拟电路、数字电路或模数混合电路进行仿真分析。用 EDA 软件进行电路的功能仿真时,可以显示逻辑仿真的波形结果,以检查逻辑错误。如果进行时序仿真时,可以设置器件及连线的延迟时间,以便检测电路中存在的冒险竞争、时序错误等问题。

(2) 数字电路的设计方法

数字电路的设计是从给定的逻辑功能要求出发,确定输入、输出变量,选择适当的逻辑器件,设计出符合要求的逻辑电路。设计过程一般有方案的提出、验证和修改三个阶段。设计方式分为传统的方式和基于EDA软件的设计方式。传统的硬件电路设计全过程都是由人工完成,硬件电路的验证和调试是在电路构成后进行的,电路存在的问题只能在验证后发现。如果存在的问题较大,有可能重新设计电路,因而设计周期长,资源浪费大,不能满足大规模集成电路设计的要求。基于EDA软件的设计方式是借助于计算机来快速准确地完成电路的设计。设计者提出方案后,利用计算机进行逻辑分析、性能分析、时序测试,如果发现错误或方案不理想,可以重复上述过程直至得到满意的电路,然后进行硬件电路的实现。这种方法提高了设计质量,缩短了设计周期,节省了设计费用,提高了产品的竞争力。因此EDA软件已成为设计人员不可或缺的有力工具。

EDA软件的种类较多,大多数软件包含以下主要工具。

原理图输入 设计者可以如同在纸上画电路一样,将逻辑电路图输入到计算机,软件自动检查电路的接线、电源及地线的连接、信号的连接等。

HDL文本输入 硬件描述语言是用文本的形式描述硬件电路的功能、信号连接关系以及时序关系。它虽然没有图形输入那么直观,但功能更强,可以进行大规模、多个芯片的数字系统的设计。常用的HDL语言有VHDL(Very High Speed Integrated Circuit)和Verilog HDL等。

测试平台 当逻辑电路的设计输入到计算机后,需要在测试平台上编写或绘制激励信号,以便测试验证电路逻辑功能或时序关系的正确性。

仿真和综合工具 仿真工具包括对电路的功能仿真和时序仿真。功能仿真用于验证电路的功能和逻辑关系是否正确。时序仿真考虑门及连线的延时,验证系统内部工作过程及输入输出的时序关系是否满足设计要求。

综合工具 将HDL描述的电路的逻辑关系,转换为门和触发器等元件及其相互连接的电路形式。

(3) 数字电路的测试技术

数字电路在正确设计和安装后,必须经过严格的测试方可使用。测试中小规模数字电路时通常使用下列基本仪器设备。

数字电压表 用来测量电路中各点的电压,并观察其测试结果是否与理论分析一致。

数字示波器 常用来观察电路各点的波形。一个复杂的数字系统,在系统时钟信号的激励下,有关逻辑关系可以从波形图中得到验证。逻辑分析仪是一种专用示波器,例如,它可以同时显示8~32位的数字波形,非常有利于对整体电路各部分之间的逻辑关系进行分析。

集成电路的复杂程度越高,测试越困难,对于系统芯片SoC的测试成本几乎占芯片成本的一半。大规模集成电路的测试系统由测试硬件和测试软件组成。测试硬件除计算机外,还有测试仪和控制测试仪的处理器。测试软件包括操作系统、开发系统、运行系统和报告系统等。集成电路测试面临的最大挑战是如何降低测试成本。

1.1.3 模拟信号和数字信号

1. 模拟信号

人们在自然界感知的许多物理量中,有一些物理量如速度、压力、温度、声音、重量以及位置等具有一个共同的特点,即它们在时间上是连续变化的,幅值上也是连续取值的。这种连续变化的物理量称为模拟量,表示模拟量的信号称为模拟信号,处理模拟信号的电子电路称为模拟电路。在工程技术上,为了便于处理和分析,通常用传感器将模拟量转换为与之成比例的电压或电流信号,然后再送到电子系统中进一步处理。图 1.1.1(a)中给出了由热电偶得到的一个模拟电压信号波形。

2. 数字信号

与模拟量相对应的另一类物理量称为数字量。它们是在一系列离散的时刻取值,数值的大小和每次的增减都是量化单位的整数倍,即它们是一系列时间离散、数值也离散的信号。表示数字量的信号称为数字信号。将工作于数字信号下的电子电路称为数字电路。例如用温度计测量某一天内的温度变化,仅在整点时刻读取数据,并且对数据进行量化,若某次温度计的读数为 $30.35\cdots^{\circ}\text{C}$,取 1°C 作为量化单位,则温度记录值为 30°C 。这样,一天内的温度记录在时间上和数值上都是离散的,温度是以 1°C 为单位增加或减少。显然,用数字信号也可以表示其他各种物理量的大小,只是存在着一定的误差,误差取决于量化单位大小的选择。

随着计算机的广泛应用,绝大多数电子系统都采用计算机对信号进行处理。由于计算机无法直接处理模拟信号,所以需要将模拟信号转换为数字信号。

3. 模拟量的数字表示

图 1.1.1 所示为转换过程中的各种波形图,图 1.1.1(a)所示是模拟电压信号。首先对模拟信号取样。图 1.1.1(b)所示为模拟信号通过取样电路后,变成时间离散、幅值连续的取样信号, t_0, t_1, t_2, \dots 为取样时间点。这里,幅值连续是指各取样点的幅值没有量化,仍然与对应的模拟信号的幅值相同,如图 1.1.1(a)和图 1.1.1(b)中 t_1 处的幅值均为模拟量 $0.915\cdots\text{V}$ 。然后对取样信号进行量化,即数字化。选取一个量化单位,将取样信号除以量化单位并取整数结果,得到时间离散、数值也离散的数字量。最后对得到的数字量进行编码,生成用 **0** 和 **1** 表示的数字信号,如图 1.1.1(c)所示。图中以 0.1 V 作为量化单位,对 t_1 处的幅值 $0.915\cdots\text{V}$ 进行量化,量化后数值为 9。该值用 8 位二进制数表示为 00001001。如果取样点足够多,量化单位足够小,数字信号可以较真实地反映模拟信号。关于模数和数模转换的详细讨论见本书第 10 章的内容。

1.1.4 数字信号的描述方法

模拟信号可以用数学表达式或波形图等表示。数字信号则用 **0**、**1** 两种值表示,即二值数字逻辑 (Binary Digital Logic);或高、低电平组成的数字波形,即逻辑电平 (Logic Level) 表示。

1. 二值数字逻辑和逻辑电平

在数字电路中,可以用 **0** 和 **1** 组成的二进制数表示数量的大小,也可以用 **0** 和 **1** 表示两种不

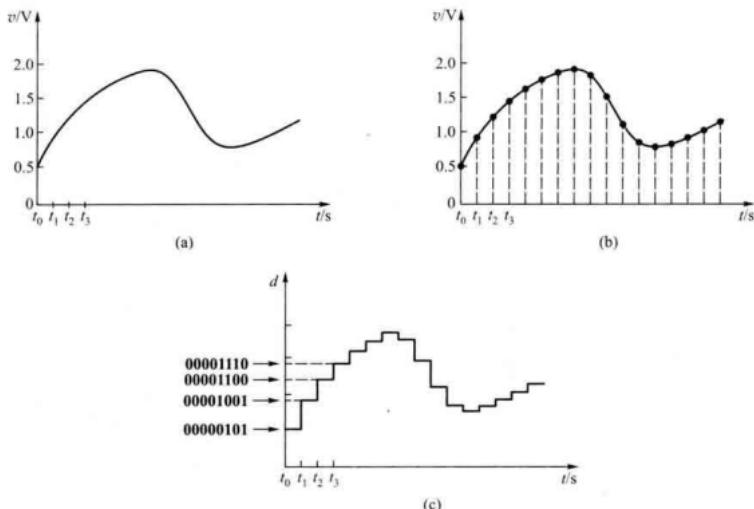


图 1.1.1 模拟量的数字表示

(a) 模拟电压信号 (b) 取样信号 (c) 数字信号

同的逻辑状态。当表示数量时,两个二进制数可以进行数值运算,常称为算术运算,将在 1.3 节介绍。当用 **0** 和 **1** 描述客观世界存在的彼此相互关联又相互对立的事物时,例如,是与非,真与假,开与关,低与高,通与断等等,这里的 **0** 和 **1** 不是数值,而是逻辑 **0** 和逻辑 **1**。这种只有两种对立逻辑状态的逻辑关系称为**二值数字逻辑**或简称**数字逻辑**。

在电路中,可以很方便地用电子器件的开关来实现二值数字逻辑。也就是以高、低电平分别表示逻辑 **1** 和 **0** 两种状态。在分析实际数字电路时,考虑的是信号之间的逻辑关系,只要能区别出表示逻辑状态的高、低电平,就可以忽略高、低电平的具体数值。表 1.1.2 所示为一种类型 CMOS 器件的电压范围与逻辑电平之间的关系。当信号电压在 3.5 ~ 5 V 范围内,都表示高电平;在 0 ~ 1.5 V 范围内,都表示低电平。这些表示数字电压的高、低电平通常称为**逻辑电平**。应当注意,逻辑电平不是物理量,而是物理量的相对表示。

表 1.1.2 电压范围与逻辑电平的关系

电压	二值逻辑	电平
3.5 ~ 5 V	1	H(高电平)
0 ~ 1.5 V	0	L(低电平)

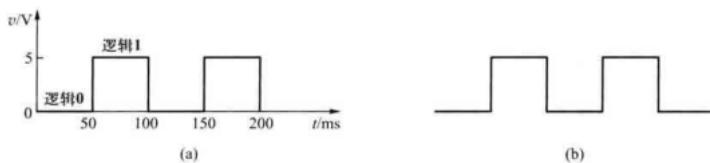


图 1.1.2 数字波形
(a) 标明时间及幅值的数字波形 (b) 数字波形的常规表示

图 1.1.2 所示为用逻辑电平描述的数字波形,用逻辑 0 表示低电平,逻辑 1 表示高电平。其中图 1.1.2(a)所示波形标出时间及幅值。通常在分析一个数字系统时,由于电路采用相同的逻辑电平标准,一般可以不标出高、低电平的电压值。时间轴也可以不标,如图 1.1.2(b)所示。

2. 数字波形

(1) 数字波形的两种类型

数字波形是逻辑电平对时间的图形表示。数字信号有两种传输波形,一种是非归零型,另一种是归零型。在图 1.1.3 中,一定的时间间隔 T ,称为 1 位(1bit),或者一拍。如果在一个时间拍内用高电平代表 1,低电平代表 0,称为非归零型,如图 1.1.3(a)所示。如果在一个时间拍内有脉冲代表 1,无脉冲代表 0,称为归零型,如图 1.1.3(b)所示。两者的区别在于高电平的表示方法不同,在一个时间拍内,非归零型信号持续为高电平,而归零型信号的高电平持续一段时间后会归零。只有作为时序控制信号使用的时钟脉冲是归零型,除此之外的大多数数字信号基本都是非归零型,非归零型信号使用较为广泛。

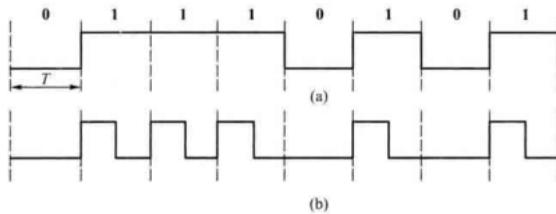


图 1.1.3 数字信号的传输波形
(a) 非归零型信号 (b) 归零型信号

数字信号只有两种取值,故称为二值信号,数字波形又称为二值位形图。非归零型信号的每位数据占用一拍时间。每秒钟所传输数据的位数称为数据率或比特率(Bit Rate)。

例 1.1.1 某通信系统每秒钟传输 1544000 位(1.544 兆位)数据,求每位数据的时间。

解:按题意,每位数据的时间为

$$\left(\frac{1.544 \times 10^6}{1 \text{ s}}\right)^{-1} = 647.67 \times 10^{-9} \text{ s} \approx 648 \text{ ns}$$

(2) 周期性和非周期性

与模拟波形相同,数字波形亦有周期性与非周期性之分。图 1.1.4(a)、(b) 所示分别为周期性与非周期性数字波形举例。

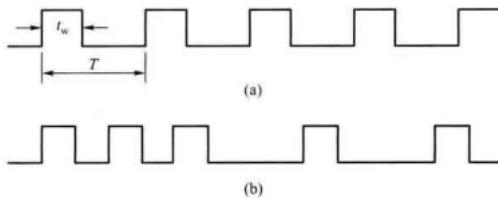


图 1.1.4 数字波形

(a) 周期性数字波形 (b) 非周期性数字波形

周期性数字波形常用周期 T 和频率 f 来描述。脉冲波形的脉冲宽度用 t_w 表示, 它表示脉冲的作用时间。另一个重要参数是占空比 q , 它表示脉冲宽度 t_w 占整个周期 T 的百分数, 常用下式来表示

$$q(\%) = \frac{t_w}{T} \times 100\% \quad (1.1.1)$$

当占空比为 50% 时, 称此时的矩形脉冲为方波, 即 0 和 1 交替出现并持续占有相同的时间。

例 1.1.2 设周期性数字波形的高电平持续 6 ms, 低电平持续 10 ms, 求占空比 q 。

解: 因数字波形的脉冲宽度 $t_w = 6 \text{ ms}$, 周期 $T = 6 \text{ ms} + 10 \text{ ms} = 16 \text{ ms}$ 。

$$q = \frac{6 \text{ ms}}{16 \text{ ms}} \times 100\% = 37.5\%$$

(3) 实际数字信号波形

在实际的数字系统中, 数字信号并没有那么理想。当矩形脉冲从低电平跳变到高电平, 或从高电平跳到低电平时, 边沿没有那么陡峭, 而要经历一个过渡过程, 分别用上升时间 t_r 和下降时间 t_f 描述, 如图 1.1.5 所示。若脉冲幅值为 V_m , 将矩形脉冲从 10% V_m 到 90% V_m 时所经历的时

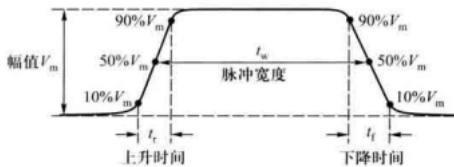


图 1.1.5 非理想脉冲波形

间称为上升时间 t_r 。下降时间则相反,将矩形脉冲从 $90\% V_m$ 下降到 $10\% V_m$ 时所经历的时间称为下降时间 t_f 。将脉冲上升沿的 $50\% V_m$ 到下降沿的 $50\% V_m$ 两个时间点所跨越的时间称为脉冲宽度 t_w ,对于不同类型的器件和电路,其上升和下降时间各不相同。一般数字信号上升和下降时间的典型值约为几纳秒(ns)。

例 1.1.3 试绘出一幅值为 5 V 的脉冲波形,设它的占空比为 50%,脉冲宽度为 100 ns,上升时间为 10 ns,而下降时间为 20 ns。

解:根据题意,所绘出的脉冲波形如图 1.1.6 所示。

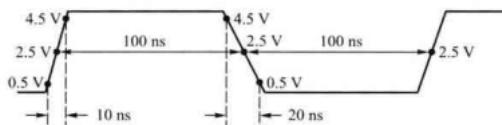


图 1.1.6 例 1.1.3 的波形图

本书所用的数字波形大多数将画成理想波形。实际上,每一波形均有上升时间 t_r 和下降时间 t_f 。当我们关注各信号之间的逻辑关系时,不必在每一波形上表示出上升和下降时间。只有当研究数字电路的翻转特性时,才绘出各信号的上升时间和下降时间。

(4) 波形图、时序图或定时图

数字波形是表达数字电路动态特性的有效工具之一。将数字电路输入变量的每一种取值与相应的输出值按照时间顺序依次排列得到的图形,称为波形图(Waveform)。

在时序电路中,电路的状态和输出对时钟脉冲序列和输入信号响应的波形图称为时序图或定时图(Timing Diagram)。时序图用来表示多个输入信号的先后顺序,以及输出如何对输入信号产生响应的过程。通常时序图侧重描述电路逻辑功能,而定时图侧重各个信号的先后顺序以及时间量。为便于比较,图 1.1.7 所示为 D 触发器的波形图。图中 CP 为时钟控制输入,D 为数据输入,Q 为输出。在图 1.1.7(b) 中为保证电路稳定工作,D 输入端的信号比 CP 上升沿提前至少

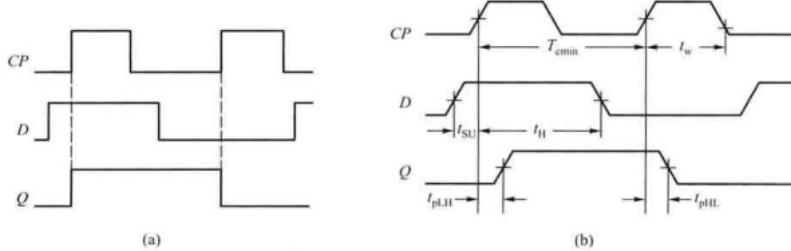


图 1.1.7 D 触发器的波形图

(a) 时序图 (b) 定时图

t_{st} (建立时间),并且要求其在CP上升沿来之后保持至少 t_h (保持时间),其他时间参数在后续章节介绍。通常数字集成电路,例如存储器和时序逻辑器件等均附有时序图,以便于进行数字系统的分析、设计和应用。

复习思考题

- 1.1.1 衡量集成电路有哪两个主要的参数?
- 1.1.2 数字电路从集成度来看可分为几大类?
- 1.1.3 按照结构和工艺分类数字电路有哪两种主要系列?
- 1.1.4 数字电路有哪些优点?
- 1.1.5 为什么数字逻辑称为二值数字逻辑?
- 1.1.6 实际数字波形和理想数字波形有什么不同?

1.2 数 制

人们在日常生活中经常遇到计数问题,并且习惯于用十进制数。而在数字系统,例如计算机中,通常采用二进制数,有时也采用十六进制数或八进制数。这种多位数码的构成方式以及从低位到高位的进位规则称为数制。

1.2.1 十进制

众所周知,中国的珠算盘是一个十进制计数器。任何一个数都可以用0,1,2,3,4,5,6,7,8,9这10个数码中的一个或几个,按一定的规律排列起来表示,其计数规律是“逢十进一”即 $9+1=10$,其中左边的“1”为十位数,右边的“0”为个位数,也就是 $10=1\times10^1+0\times10^0$ 。所谓十进制就是以10为基数的计数体制。

这样,每一数码处于不同的位置时,它所代表的数值是不同的。例如,十进制数4587.29可以表示为

$$4587.29 = 4 \times 10^3 + 5 \times 10^2 + 8 \times 10^1 + 7 \times 10^0 + 2 \times 10^{-1} + 9 \times 10^{-2}$$

式中 10^3 、 10^2 、 10^1 和 10^0 分别为千位、百位、十位和个位数码的权,而小数点以右数码的权值是10的负幂。这与珠算盘横梁上所标示的个、十、百、千的位权是相同的。

一般地说,任意十进制数可表示为

$$(N)_D \text{①} = \sum_{i=-\infty}^{+\infty} K_i \times 10^i \quad (1.2.1)$$

式中 K_i 为基数“10”的第*i*次幂的系数,它可以是0~9中任何一个数字。

① 下标 D(Decimal) 表示十进制。

如果将式(1.2.1)中的10用字母R来代替,就可以得到任意进制数的表达式

$$(N)_R = \sum_{i=-\infty}^{\infty} K_i \times R^i \quad (1.2.2)$$

式中 K_i 是第*i*次幂的系数,根据基数R的不同,它的取值为0到R-1个不同的数码。例如对于十进制数,R为10,所以 K_i 的取值为0~9共10个数码。

数字电路不便于存储或处理十进制数。因为构成数字电路的基本思路是把电路的状态与数码对应起来。二进制只有0和1两个数码,分别用数字电路的两种状态来表示。而十进制的10个数码要求电路有10个完全不同的状态,这样使得电路很复杂,因此在数字电路中不直接处理十进制数。

1.2.2 二进制

1. 二进制的表示方法

二进制数中,只有0和1两个数码,并且计数规律是“逢二进一”,即1+1=10(读为“壹零”)。必须注意,这里的“10”与十进制数的“10”是完全不同的,它并不代表数“拾”。左边的“1”表示 2^1 的系数,右边的“0”表示 2^0 的系数,也就是 $10=1\times2^1+0\times2^0$ 。因此,所谓二进制就是以2为基数的计数体制。

根据式(1.2.2),任意二进制数可表示为

$$(N)_B \text{①} = \sum_{i=-\infty}^{\infty} K_i \times 2^i \quad (1.2.3)$$

式中 K_i 为基数“2”的第*i*次幂的系数,它可以是0或者1。这样式(1.2.3)也可以作为二进制数转换为十进制数的转换公式。

例 1.2.1 试将二进制数(1010110)_B转换为十进制数。

解: 将每一位二进制数与其位权相乘,然后相加便得相应的十进制数。

$$(1010110)_B = 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = (86)_B$$

2. 二进制的优点

与十进制相比较,二进制具有一定的优点,因此它在计算机技术中被广泛采用。

(1) 二进制的数字电路简单可靠,所用元件少

二进制数的每一位只有0和1两个状态,可用任何具有两个不同稳定状态的元件来表示,如BJT的饱和与截止,继电器触点的闭合与断开,灯的亮和不亮等。只要规定其中一种状态表示1,另一种状态表示0,就可以表示二进制数。这样,数码的存储、分析和传输,就可以用简单而可靠的方式进行。

(2) 二进制的基本运算规则简单,运算操作方便

二进制只有两个数码0和1,基本运算规则简单。

① 下标B(Binary)表示二进制。

但是,采用二进制也有缺点。用二进制表示一个数时,位数太多,例如,十进制数 49 表示为二进制时,即为 **110001**,不便于书写和阅读,使用不方便。因此,通常在阅读和书写时采用十进制数。在送入计算机时,计算机先将十进制数转换成数字系统能接受的二进制数;而在运算结束后,再将二进制转换为十进制数,输出十进制结果。

3. 二进制的波形表示

在数字电路中,二值数据常用数字波形来表示。图 1.2.1 所示为 16 位数据 **0010111100111010** 的波形表示。数据的波形表示比较直观,也便于用示波器进行监视。图 1.2.2 所示为一计数器的波形,图中最左列标出了二进制数的位权($2^0, 2^1, 2^2, 2^3$)以及最低位(Least Significant Bit, LSB)和最高位(Most Significant Bit, MSB),最后一行标出了从 0 到 15 的等效十进制数。

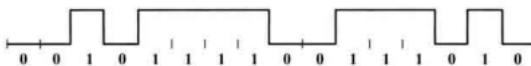


图 1.2.1 16 位数据的波形表示

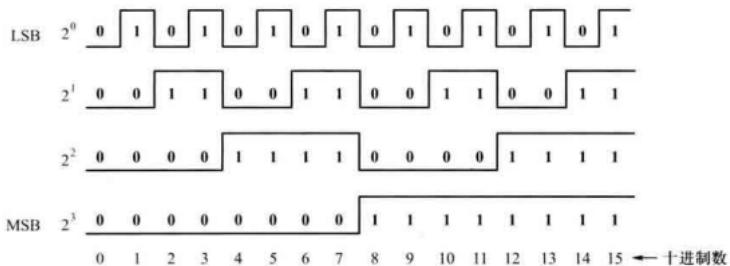


图 1.2.2 用二进制数表示 0~15 波形图

从图 1.2.2 还可看出,每一位的波形均为方波,其占空比均为 50%,但从低位到高位,每一波形的频率逐位减半。

4. 二进制数据的传输

二进制数据从一处传输到另一处,可以采用串行的方式或并行的方式。对于串行的方式,一组数据在时钟脉冲的控制下逐位传送。串行方式所需的设备简单,只需一根导线和一共同接地端即可。两台计算机之间,或计算机通过电话线与网络连接均采用这种方式。

二进制数据作串行传输的示意图如图 1.2.3 所示,图 1.2.3(a) 表示二进制数据 **00110110** 从计算机 A 串行传送到计算机 B。图 1.2.3(b) 是在时钟脉冲 CP 的控制下,数据由最高位 MSB 到最低位 LSB 依次传输的波形图。注意每传送一位数需要一个时钟周期,并且在时钟脉冲的下降沿完成。

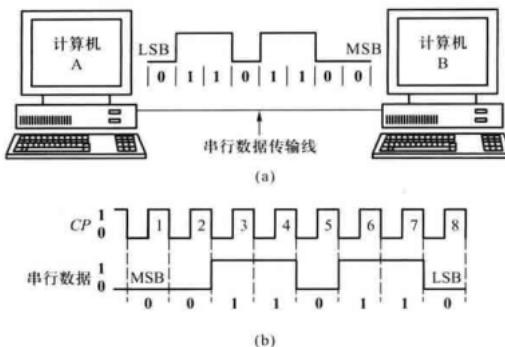


图 1.2.3 二进制数据的串行传输

(a) 两台计算机之间的串行通信 (b) 二进制数据的串行表示

若要求传输速度快,则可采用并行传输的方式,即将一组二进制数据同时传送。图 1.2.4 为并行传输数据的示意图。图 1.2.4(a)所示为一台打印机从一台计算机以 8 位数据并行的方式获取数据。传输 8 位数据所需的时间为一个时钟脉冲的周期,只有串行传输时间的 $\frac{1}{8}$ 。但所需设备复杂,需用 8 条传输线和其他部件。并行传输在数字系统中是一种常用的技术。

1.2.3 十—二进制之间的转换

既然同一个数可以用二进制和十进制两种不同形式来表示,那么两者之间必然有一定的转换关系。前面已经介绍了二进制数转换为十进制数的方法,是将每位二进制数与其权相乘,然后相加便得到相应的十进制数。十进制数转换为二进制数时,整数部分和小数部分的方法不同,下面分别介绍。

对于整数部分可写成

$$(N)_b = b_n \times 2^n + b_{n-1} \times 2^{n-1} + \cdots + b_1 \times 2^1 + b_0 \times 2^0 \quad (1.2.4)$$

式中 $b_n, b_{n-1}, \dots, b_1, b_0$ 是二进制数各位数字。将等式两边分别除以 2, 得

$$\frac{1}{2}(N)_b = b_n \times 2^{n-1} + b_{n-1} \times 2^{n-2} + \cdots + b_1 \times 2^0 + \frac{b_0}{2} \quad (1.2.5)$$

由此可知,将十进制数除以 2,其余数为 b_0 。得到的商为

$$b_n \times 2^{n-1} + b_{n-1} \times 2^{n-2} + \cdots + b_1$$

可以写成

$$b_n \times 2^{n-1} + b_{n-1} \times 2^{n-2} + \cdots + b_1 = 2 \left(b_n \times 2^{n-2} + b_{n-1} \times 2^{n-3} + \cdots + b_2 + \frac{b_1}{2} \right) \quad (1.2.6)$$

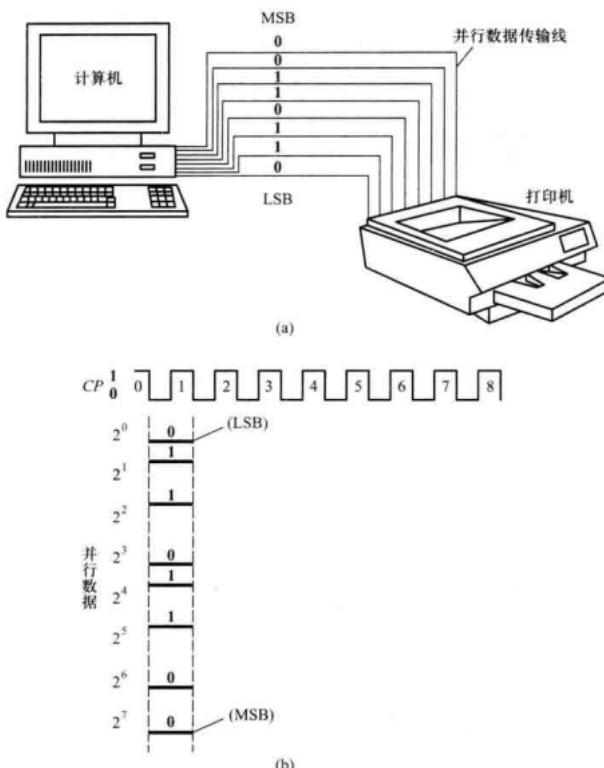


图 1.2.4 并行传输数据的示意图
 (a) 计算机与打印机之间的并行通信 (b) 二进制数据的并行表示

将式(1.2.6)再除以2,其余数为 b_1 。不难推知,将十进制整数每除以一次2,就可根据余数得到二进制数的1位数字。因此,只要连续除以2直到商为0,就可由所有的余数求出二进制数。

例 1.2.2 将十进制数 $(37)_D$ 转换为二进制数。

解: 根据上述原理,可将 $(37)_D$ 按如下的步骤转换为二进制数:

$$\begin{array}{r}
 2 \mid 37 \cdots \text{余 } 1 \cdots \cdots b_0 \\
 2 \mid 18 \cdots \cdots \text{余 } 0 \cdots \cdots b_1 \\
 2 \mid 9 \cdots \cdots \text{余 } 1 \cdots \cdots b_2 \\
 2 \mid 4 \cdots \cdots \text{余 } 0 \cdots \cdots b_3 \\
 2 \mid 2 \cdots \cdots \text{余 } 0 \cdots \cdots b_4 \\
 2 \mid 1 \cdots \cdots \text{余 } 1 \cdots \cdots b_5 \\
 0
 \end{array}$$

由上得

$$(37)_0 = (100101)_B$$

当十进制数较大时,不必逐次除2,而是将十进制数和与其相当的2的幂项对比,使转换过程得到简化。

例 1.2.3 将 $(133)_0$ 转换为二进制数。

解:由于 2^7 为128,而 $133-128=5=2^2+2^0$,所以对应二进制数 $b_7=1, b_2=1, b_0=1$,其余各系数均为0,所以得

$$(133)_0 = (10000101)_B$$

值得指出,多数计算机或数字系统中只处理4、8、16、32或64位等的二进制数据,因此,数据的位数需配成规格化的位数,如例1.2.2中转换结果为**100101**,如将它配成8位,则相应的高幂项应填以**0**,其值不变,即

$$\mathbf{100101 = 00100101}$$

对于二进制的小数部分可写成

$$(N)_0 = b_{-1} \times 2^{-1} + b_{-2} \times 2^{-2} + \cdots + b_{-(n-1)} \times 2^{-(n-1)} + b_{-n} \times 2^{-n} \quad (1.2.7)$$

将上式两边分别乘以2,得

$$2 \times (N)_0 = b_{-1} \times 2^0 + b_{-2} \times 2^{-1} + \cdots + b_{-(n-1)} \times 2^{-(n-2)} + b_{-n} \times 2^{-(n-1)} \quad (1.2.8)$$

由此可见,将十进制小数乘以2,所得乘积的整数即为 b_{-1} 。不难推知,将十进制小数每次去掉上次所得积中的整数再乘以2,直到满足误差要求进行“四舍五入”为止,就可完成将十进制小数转换成二进制小数。

例 1.2.4 将 $(0.706)_0$ 转换为二进制数,要求其误差不大于 2^{-10} 。

解:按上面介绍的方法计算,可得 $b_{-1}, b_{-2}, \dots, b_{-9}$ 如下:

$$0.706 \times 2 = 1.412 \cdots \cdots 1 \cdots \cdots b_{-1}$$

$$0.412 \times 2 = 0.824 \cdots \cdots 0 \cdots \cdots b_{-2}$$

$$0.824 \times 2 = 1.648 \cdots \cdots 1 \cdots \cdots b_{-3}$$

$$0.648 \times 2 = 1.296 \cdots \cdots 1 \cdots \cdots b_{-4}$$

$$0.296 \times 2 = 0.592 \cdots \cdots 0 \cdots \cdots b_{-5}$$

$$0.592 \times 2 = 1.184 \cdots \cdots 1 \cdots \cdots b_{-6}$$

$$0.184 \times 2 = 0.368 \cdots \cdots 0 \cdots \cdots b_{-7}$$

$$\begin{aligned}0.368 \times 2 &= 0.736 \cdots \mathbf{0} \cdots b_{-8} \\0.736 \times 2 &= 1.472 \cdots \mathbf{1} \cdots b_{-9} \\0.472 \times 2 &= 0.944 \cdots \mathbf{0} \cdots b_{-10}\end{aligned}$$

由于最后的小数大于 0.5，根据“四舍五入”的原则， b_{-10} 应为 1。所以， $(0.706)_b = (0.1011010011)_b$ ，其误差 $\varepsilon < 2^{-10}$ 。

1.2.4 十六进制和八进制

对于同一个数，用二进制数表示比用十进制数表示需要的位数多，不便书写和记忆，因此在计算机资料中常采用十六进制数或八进制数来表示。

1. 十六进制

十六进制数采用 16 个数码，分别为 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F。其中 A, B, C, D, E, F 依次相当于十进制数中的 10, 11, 12, 13, 14, 15。十六进制数是“逢十六进一”，见表 1.2.1，它是以 16 为基数的计数体制。

根据式(1.2.2)，任意十六进制数可表达如下：

$$(N)_H \text{ ① } = \sum_{i=-\infty}^{\infty} K_i \times 16^i \quad (1.2.9)$$

式中 K_i 为基数 16 的第 i 次幂的系数。

例 1.2.5 将十六进制数 4E6 转换为十进制数。

$$\text{解: } (4E6)_H = 4 \times 16^2 + 14 \times 16^1 + 6 \times 16^0 = 1254$$

2. 十六-二进制之间的转换

4 位二进制数有 16 个状态，而 1 位十六进制数有 16 个不同的数码，因此二进制转换成为十六进制非常简单，以小数点为基准，整数部分从右到左每 4 位一组，不足 4 位的在高位补 0；小数部分从左到右每 4 位一组，不足 4 位的在低位补 0。每 4 位一组的二进制数就表示 1 位十六进制数。

例 1.2.6 将二进制数 101110010111.01001 转换为十六进制数。

解：将 4 位二进制数分为一组，用相应的十六进制数代替即得

$$(0101 \quad 1100 \quad 1011. \quad 0100 \quad 1000)_B = (5CB.48)_H$$

例 1.2.7 将十六进制数 F15.6 转换为二进制数。

解：将每位十六进制数用 4 位二进制数代替即得相应的二进制数

$$(F15.6)_H = (1111 \quad 0001 \quad 0101. \quad 011)_B$$

3. 八进制

同样，八进制数由 0 ~ 7 共 8 个数码表示，且“逢八进一”。任意八进制可表示为

① 下标 H(Hexadecimal) 表示十六进制。 $(59)_H$ 也可写成 59H。

$$(N)_8 \stackrel{(1)}{=} \sum_{i=-\infty}^{\infty} K_i \times 8^i \quad (1.2.10)$$

式中 K_i 为基数 8 的第 i 次幂的系数。

4. 八-二进制之间的转换

同理,对于八进制数,可将 3 位二进制数分为一组,对应于 1 位八进制数。如将例 1.2.6 中的二进制数变换为八进制数,则可写成

$$(010 \ 111 \ 001 \ 011, \ 010 \ 010)_8 = (2713.22)_0$$

至于十进制数变换为十六进制数,可先将十进制数变换为二进制数,再由二进制数转换为十六进制数。

为便于对照,将十进制、二进制、八进制及十六进制之间的关系列于表 1.2.1 中。

表 1.2.1 几种数制之间的关系对照表

十进制数	二进制数	八进制数	十六进制数
0	0 0 0 0 0	0	0
1	0 0 0 0 1	1	1
2	0 0 0 1 0	2	2
3	0 0 0 1 1	3	3
4	0 0 1 0 0	4	4
5	0 0 1 0 1	5	5
6	0 0 1 1 0	6	6
7	0 0 1 1 1	7	7
8	0 1 0 0 0	10	8
9	0 1 0 0 1	11	9
10	0 1 0 1 0	12	A
11	0 1 0 1 1	13	B
12	0 1 1 0 0	14	C
13	0 1 1 0 1	15	D
14	0 1 1 1 0	16	E
15	0 1 1 1 1	17	F
16	1 0 0 0 0	20	10
17	1 0 0 0 1	21	11
18	1 0 0 1 0	22	12
19	1 0 0 1 1	23	13
20	1 0 1 0 0	24	14

① 下标 O(Octal) 表示八进制。为了更清楚起见,某些文献中也用下标 2,8,10 及 16 分别表示二进制、八进制、十进制及十六进制。

复习思考题

- 1.2.1 为什么在计算机或数字系统中通常采用二进制数?
- 1.2.2 在二进制数中,其位权的规律是什么?
- 1.2.3 十六进制数主要用于何种场合?
- 1.2.4 二进制与十六进制之间如何进行转换?

1.3 二进制数的算术运算

在数字电路中,0 和 1 既可以表示逻辑状态,又可以表示数量大小。当表示数量时,两个二进制数可以进行算术运算。下面介绍无符号二进制数和有符号二进制数的算术运算。

1.3.1 无符号二进制数的算术运算

二进制数的加、减、乘、除四种运算的运算规则与十进制数类似,两者唯一的区别在于进位或借位规则不同。

1. 二进制加法

无符号二进制的加法规则是:

$$0+0=0, 0+1=1, 1+1=\boxed{1}0$$

方框中的 1 是进位位,表示两个 1 相加“逢二进一”。

例 1.3.1 计算两个二进制数 1010 和 0101 的和。

解:

$$\begin{array}{r} 1010 \\ + 0101 \\ \hline 1111 \end{array}$$

所以

$$1010+0101=1111$$

无符号二进制数的加法运算是基础,数字系统中的各种算术运算都将通过它来进行。

2. 二进制减法

无符号二进制数的减法规则是:

$$0-0=0, 1-1=0, 1-0=1, 0-1=\boxed{1}1$$

方框中的 1 是借位位,表示 0 减 1 时不够减,向高位借 1。

例 1.3.2 计算两个二进制数 1010 和 0101 的差。

解:

$$\begin{array}{r}
 1010 \\
 -0101 \\
 \hline
 0101
 \end{array}$$

所以

$$1010 - 0101 = 0101$$

由于无符号二进制数中无法表示负数，则要求被减数一定大于减数。

3. 乘法运算和除法运算

无符号二进制数的乘法规则是：

$$0 \times 0 = 0, 0 \times 1 = 0, 1 \times 0 = 0, 1 \times 1 = 1$$

无符号二进制数的除法规则是：

$$0 \div 1 = 0, 1 \div 1 = 1$$

注意，除数不能为 0，否则无意义。

例 1.3.3 计算两个二进制数 1010 和 0101 的积。

解：

$$\begin{array}{r}
 1010 \\
 \times 0101 \\
 \hline
 1010 \\
 0000 \\
 1010 \\
 \hline
 110010
 \end{array}$$

所以

$$1010 \times 0101 = 110010$$

由上述运算过程可见，乘法运算是由左移被乘数与加法运算组成的。

例 1.3.4 计算两个二进制数 1010 和 111 之商。

解：

$$\begin{array}{r}
 1.011 \\
 111 \sqrt{1010} \\
 111 \\
 \hline
 1100 \\
 111 \\
 \hline
 1010 \\
 111 \\
 \hline
 11 \cdots \text{余数}
 \end{array}$$

所以

$$1010 \div 111 = 1.011 \text{ 余 } 11$$

由上述运算过程可见,除法运算是由右移除数与减法运算组成的。

1.3.2 带符号二进制数的减法运算

前面只考虑了二进制数的正数,当涉及负数时,就要用有符号的二进制数表示。在定点运算的情况下,二进制数的最高位(即最左边的位)表示符号位,且用0表示正数,用1表示负数。其余部分为数值位。例如

$$(+11)_B = (\boxed{0} \ 1011)_B$$

$$(-11)_B = (\boxed{1} \ 1011)_B$$

以上是用原码的形式表示数值位。在数字电路或系统中,为简化电路,常将负数用补码表示,以便将减法运算变为加法运算。下面首先介绍补码的概念,然后举例说明负数的求补方法及减法运算,同时为了便于得到补码,引入反码的概念。

1. 二进制数的补码表示

若基数为R,位数为n的原码N,其补码为

$$N_{\#} = R^n - N \quad (1.3.1)$$

首先以常用的十进制数为例,2和46的补码分别为 $10-2=8$ 和 $10^2-46=54$ 。

例 1.3.5 利用补码分别计算出 $8-2$ 和 $82-46$ 。

解:根据式(1.3.1)可知 $-N = N_{\#} - R^n$,所以

$$8-2 = 8+2_{\#}-10 = 8+8-10 = 6$$

$$82-46 = 82+46_{\#}-10^2 = 82+54-100 = 36$$

上面计算的是无符号十进制数的补码。对于无符号二进制数,同样可以利用式(1.3.1)进行补码的计算。当考虑负数情况时,带符号二进制数补码的计算方法如下:

补码或反码的最高位为符号位,正数为0,负数为1。

当二进制数为正数时,其补码、反码与原码相同。

当二进制数为负数时,将原码的数值位逐位求反(即得到反码),然后在最低位加1得到补码。

例 1.3.6 分别计算出 $A=+6$ 和 $B=-6$ 的4位二进制的原码、反码和补码。

解: A 和 B 的绝对值均为6。除最高位为符号位外,还有3位为数值位。其原码、反码和补码分别为

$$A_{原} = \boxed{0} \ 110 \qquad B_{原} = \boxed{1} \ 110$$

$$A_{反} = \boxed{0} \ 110 \qquad B_{反} = \boxed{1} \ 001$$

$$A_{\#} = \boxed{0} \ 110 \qquad B_{\#} = \boxed{1} \ 010$$

4位带符号的二进制数的原码、反码和补码对照表如表1.3.1所示。对于-0和+0,原码和反码用不同的4位二进制数表示,而补码则是相同的。因此,它们表示的数值范围分别为,原码

是 $-7 \sim +7$, 反码也是 $-7 \sim +7$, 补码是 $-8 \sim +7$ 。由此可以推知, 对于 n 位带符号的二进制数的原码、反码和补码的数值范围分别为:

$$\text{原码} \quad -(2^{n-1}-1) \sim +(2^{n-1}-1)$$

$$\text{反码} \quad -(2^{n-1}-1) \sim +(2^{n-1}-1)$$

$$\text{补码} \quad -2^{n-1} \sim +(2^{n-1}-1)$$

表 1.3.1 4 位二进制数原码、反码、补码对照表

十进制数	二进制数		
	原码	反码	补码
-8	—	—	1 0 0 0
-7	1 1 1 1	1 0 0 0	1 0 0 1
-6	1 1 1 0	1 0 0 1	1 0 1 0
-5	1 1 0 1	1 0 1 0	1 0 1 1
-4	1 1 0 0	1 0 1 1	1 1 0 0
-3	1 0 1 1	1 1 0 0	1 1 0 1
-2	1 0 1 0	1 1 0 1	1 1 1 0
-1	1 0 0 1	1 1 1 0	1 1 1 1
-0	1 0 0 0	1 1 1 1	0 0 0 0
+0	0 0 0 0	0 0 0 0	0 0 0 0
+1	0 0 0 1	0 0 0 1	0 0 0 1
+2	0 0 1 0	0 0 1 0	0 0 1 0
+3	0 0 1 1	0 0 1 1	0 0 1 1
+4	0 1 0 0	0 1 0 0	0 1 0 0
+5	0 1 0 1	0 1 0 1	0 1 0 1
+6	0 1 1 0	0 1 1 0	0 1 1 0
+7	0 1 1 1	0 1 1 1	0 1 1 1

2. 二进制补码的减法运算

采用补码的形式, 可以很方便地进行带符号二进制数的减法运算。减法运算的原理是减去一个正数相当于加上一个负数, 即 $A-B=A+(-B)$, 对 $(-B)$ 求补码, 然后进行加法运算。至于乘法和除法运算, 从 1.3.1 节介绍知道, 可以采用移位与加法或减法的组合完成。

进行二进制补码的加法运算时, 必须注意被加数补码与加数补码的位数相等, 即让两个二进制数补码的符号位对齐。通常两个二进制数的补码采用相同的位数表示。

例 1.3.7 试用 4 位二进制补码计算 $5-2$ 。

解：因为 $(5-2)_B = (5)_B + (-2)_B$

$$\begin{array}{r}
 = 0101 + 1110 \\
 = 0011 \\
 \hline
 \end{array}
 \quad \begin{array}{r}
 0101 \\
 + 1110 \\
 \hline
 [1] 0011
 \end{array}$$

所以 $5-2=3$ 自动丢弃 \leftarrow

两个二进制补码相加时，方框中的 1 是进位位，在计算中会自动丢失，因为运算是以 4 位二进制补码表示的，计算结果仍然保留 4 位数。

3. 溢出

例 1.3.8 试用 4 位二进制补码计算 $5+7$ 。

解：因为 $(5+7)_B = (5)_B + (7)_B$

$$\begin{array}{r}
 = 0101 + 0111 \\
 = 1100 \\
 \hline
 \end{array}
 \quad \begin{array}{r}
 0101 \\
 + 0111 \\
 \hline
 [1] 1000
 \end{array}$$

计算结果 **1100** 表示 -4 ，而实际正确的结果应该为 12 。错误产生的原因在于 4 位二进制补码中，有 3 位是数值位，它所表示的范围为 $-8 \sim +7$ ，而本题的结果需要 4 位数值位表示，因而产生溢出。解决溢出的办法是进行位扩展，即用 5 位以上的二进制补码表示，就不会产生溢出了。

4. 溢出的判别

两个符号相反的数相加不会产生溢出，但两个符号相同的数相加有可能产生溢出。如何判断是否产生溢出？根据下列同符号 4 位二进制补码的计算结果可以推知。

$+4$ $+)+3$ \hline $+7$	0100 $+0011$ \hline $[0]0111$ (a)	-5 $+)-3$ \hline -8	1011 $+1101$ \hline $[1]1000$ (b)
$+2$ $+)+6$ \hline $+8$	0010 $+0110$ \hline $[0]1000$ (c)	-3 $+)-6$ \hline -9	1101 $+1010$ \hline $[1]0111$ (d)

4 位二进制补码表示的数值范围为 $-8 \sim +7$ ，所以 (a) 和 (b) 没有产生溢出，结果是正确的。(c) 和 (d) 的运算结果应分别是 $+8$ 和 -9 ，均超过了允许的范围，产生溢出。比较 4 种情况可以看出，当方框中的进位位与和数的符号位（即 b_3 位）相同时，则运算结果是错误的，产生溢出。

复习思考题

1.3.1 为什么说二进制数的加法运算是算术运算基础？

1.3.2 二进制数乘法、除法运算过程各有什么规律？

1.3.3 说明反码与补码之间的关系。

1.3.4 简要说明由加补码完成减法运算的原理。

1.3.5 说明溢出产生的原因及解决的办法。

1.4 二进制代码

数值和文字符号(包括控制符)是数字系统中处理最多的两类信息。数值信息的表示方法如前所述。文字符号信息通常也采用二进制数码表示,这些数码并不表示数量的大小,仅仅区别不同事物而已。这些特定的二进制数码称为代码。以一定的规则编制代码,用以表示十进制数值、字母、符号等的过程称为编码。将代码还原成所表示的十进制数、字母、符号等的过程称为解码或译码。

若所需编码的信息有 N 项,则需要的二进制数码的位数 n 应满足如下关系

$$2^n \geq N$$

1.4.1 二-十进制码

二-十进制码就是用 4 位二进制数来表示 1 位十进制数中的 0~9 十个数码,即二进制编码的十进制码(Binary-Coded-Decimal,BCD 码)。4 位二进制数有 16 种不同的组合方式,即 16 种代码,根据不同的规则从中选择 10 种来表示十进制的 10 个数码。其方案有很多种。表 1.4.1 列出了几种常用的 BCD 码。

表 1.4.1 几种常见的 BCD 码

十进制数	有权码			无权码	
	8421 码	2421 码	5421 码	余 3 码	余 3 循环码
0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 1 1	0 0 1 0
1	0 0 0 1	0 0 0 1	0 0 0 1	0 1 0 0	0 1 1 0
2	0 0 1 0	0 0 1 0	0 0 1 0	0 1 0 1	0 1 1 1
3	0 0 1 1	0 0 1 1	0 0 1 1	0 1 1 0	0 1 0 1
4	0 1 0 0	0 1 0 0	0 1 0 0	0 1 1 1	0 1 0 0
5	0 1 0 1	1 0 1 1	1 0 0 0	1 0 0 0	1 1 0 0
6	0 1 1 0	1 1 0 0	1 0 0 1	1 0 0 1	1 1 0 1
7	0 1 1 1	1 1 0 1	1 0 1 0	1 0 1 0	1 1 1 1
8	1 0 0 0	1 1 1 0	1 0 1 1	1 0 1 1	1 1 1 0
9	1 0 0 1	1 1 1 1	1 1 0 0	1 1 0 0	1 0 1 0

8421BCD 码是最常用的一种 BCD 码。它是由 4 位自然二进制数 0000(0) 到 1111(15) 16 种组合的前 10 种组合,即 0000(0)~1001(9) 构成,其余 6 种组合是无效的。其编码中每位的值都是固定数,称为位权。 b_3 位的权为 $2^3=8$, b_2 位的权为 $2^2=4$, b_1 位的权为 $2^1=2$, b_0 位的权为 $2^0=1$,因此称为 8421BCD 码。它属于有权码。

2421 码也是有权码。对应 b_3, b_2, b_1 和 b_0 的权分别是 2、4、2 和 1。它的特点是，将任意一个十进制数 N 的代码各位取反，所得代码正好表示 N 的 9 的补码。例如 2 的代码 0010 各位取反所得代码 1101 正好是 $9 - 2 = 7$ 的代码。这种特性称为自补性。具有自补特性的代码称为自补码。

5421 码也是有权码，它各位的权依次为 5、4、2、1。在一般情况下，有权码的十进制数与二进制数之间可用下式来表示

$$(N)_0 = W_3 b_3 + W_2 b_2 + W_1 b_1 + W_0 b_0 \quad (1.4.1)$$

式中 $W_3 \sim W_0$ 为二进制码中各位的权。

余 3 码是自补码，与 2421 码有类似的自补性。当两个十进制数之和是 10 时，相应的二进制数之和是 16。例如 1 和 9、2 和 8、…、6 和 4。该特点便于求 10 的补码。余 3 码也是无权码，它的每一位没有一定的权值，不能用式(1.4.1)来表示其编码关系，但其编码可以由 8421 码加 3(**0011**)得出。

余 3 循环码也是一种无权码，它的特点是具有相邻性，任意两个相邻代码之间仅有 1 位取值不同，例如 4 和 5 两个代码 0100 和 1100 仅 b_3 不同。余 3 循环码可以看成是将格雷码首尾各 3 种状态去掉后得到的。下面介绍格雷码。

1.4.2 格雷码

格雷码(Gray Code)也是一种常见的无权码，表 1.4.2 给出了 4 位格雷码的编码顺序。它也具有相邻性，即两个相邻代码之间仅有 1 位取值不同，并且 0 和最大数($2^4 - 1$)之间也只有 1 位不同，因此它是一种循环码。格雷码的这个特点使它在代码形成和传输时引起的误差较小。因而常用于将模拟量转换成用连续二进制序数列表示数字量的系统中。当模拟量发生微小变化而引起数字量从一位变化到相邻位时，例如从十进制数的 3 到 4，格雷码变化是从 **0010** 到 **0110**，只有 b_2 位从 **0** 变成 **1**，其余 3 位保持不变。如果对于自然二进制码，其变化是从 **0011** 到 **0100**，有 3 位发生变化，如果 b_2 位从 **0** 到 **1** 变化所需的时间，比 b_1 和 b_0 从 **1** 变到 **0** 的时间长，则在转换过程中，会出现瞬间错误数码 **0000**。而格雷码可以避免错误数码的出现。

格雷码的缺点是不能直接进行算术运算。这是因为格雷码是无权码，其每一位的权值不是固定的。

表 1.4.2 格雷码

十进制数	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							
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

续表

十进制数	二进制码				格雷码			
	b_3	b_2	b_1	b_0	G_3	G_2	G_1	G_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

格雷码和二进制码之间经常需要转换,具体转换方法如下。

1. 二进制码到格雷码的转换

- (1) 格雷码的最高位(最左边)与二进制码的最高位相同。
- (2) 从左到右,逐一将二进制码相邻的2位相加(舍去进位),作为格雷码的下一位。

例 1.4.1 将二进制码 1011 转换成格雷码。

解: 转换过程如下:

$$\begin{array}{ccccccc}
 1 & - & + & 0 & - & + & 1 & - & + & 1 \\
 \downarrow & & \downarrow & & \downarrow & & \downarrow & & \\
 1 & & 1 & & 1 & & 0 & & \\
 & & & & & & & & \text{格雷码}
 \end{array} \quad \begin{array}{l} \text{二进制码} \\ \qquad \qquad \qquad \qquad \qquad \qquad \qquad \end{array}$$

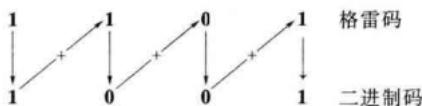
所以,格雷码为 1110。

2. 格雷码到二进制码的转换

- (1) 二进制码的最高位(最左边)与格雷码的最高位相同。
- (2) 将产生的每一位二进制码,与下一位相邻的格雷码相加(舍去进位),作为二进制码的下一位。

例 1.4.2 将格雷码 1101 转换成二进制码。

解: 转换过程如下:



所以,二进制码为 1001。

格雷码和二进制码相互转换的另一种方法是用异或表达式(参见 4.2.1 节)。

1.4.3 ASCII 码

计算机不仅用于处理数字,而且用于处理字母、符号等文字信息。人们通过键盘上的字母、符号和数值向计算机发送数据和指令,每一个键符可用一个二进制码来表示,美国标准信息交换码(American Standard Code for Information Interchange, ASCII)是目前国际上最通用的一种字符码。它是用 7 位二进制码来表示 128 个十进制数、英文大小写字母、控制符、运算符以及特殊符号,如表 1.4.3A 所示,其中一些字符的含义如表 1.4.3B 所示。

表 1.4.3A ASCII 码

$b_3 b_2 b_1 b_0$	$b_6 b_5 b_4$							
	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
0 0 0 0	NUL	DLE	SP	0	@	P	'	p
0 0 0 1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	STX	DC2	"	2	B	R	b	r
0 0 1 1	ETX	DC3	#	3	C	S	c	s
0 1 0 0	EOT	DC4	\$	4	D	T	d	t
0 1 0 1	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	ACK	SYN	&	6	F	V	f	v
0 1 1 1	BEL	ETB	*	7	G	W	g	w
1 0 0 0	BS	CAN	(8	H	X	h	x
1 0 0 1	HT	EM)	9	I	Y	i	y
1 0 1 0	LF	SUB	*	:	J	Z	j	z
1 0 1 1	VT	ESC	+	;	K	[k]
1 1 0 0	FF	FS	,	<	L	\	l	
1 1 0 1	CR	GS	-	=	M]	m	!
1 1 1 0	SO	RS	.	>	N	^	n	~
1 1 1 1	SI	US	/	?	O	—	o	DEL

表 1.4.3B ASCII 码中各字符的含义

字符	含 义	字符	含 义
NUL	Null 空白	DC1	Device control 1 设备控制 1
SOH	Start of heading 标题开始	DC2	Device control 2 设备控制 2
STX	Start of text 文本开始	DC3	Device control 3 设备控制 3
ETX	End of text 文本结束	DC4	Device control 4 设备控制 4
EOT	End of transmission 传输结束	NAK	Negative acknowledge 否认
ENQ	Enquiry 询问	SYN	Synchronous idle 同步空转
ACK	Acknowledge 确认	ETB	End of transmission block 块传输结束
BEL	Bell 报警	CAN	Cancel 取消
BS	Backspace 退一格	EM	End of medium 纸尽
HT	Horizontal tab 水平列表	SUB	Substitute 替换
LF	Line feed 换行	ESC	Escape 脱离
VT	Vertical tab 垂直列表	FS	File separator 文件分隔符
FF	Form feed 走纸	GS	Group separator 组分隔符
CR	Carriage return 回车	RS	Record separator 记录分隔符
SO	Shift out 移出	US	Unit separator 单元分隔符
SI	Shift in 移入	SP	Space 空格
DLE	Data link escape 数据链路换码	DEL	Delete 删除

复习思考题

- 1.4.1 什么叫 BCD 码？试列举几种常用的 BCD 码。
- 1.4.2 为什么 8421BCD 码用的较普遍？
- 1.4.3 格雷码有什么特点，用于什么场合？
- 1.4.4 格雷码与二进制码之间如何进行转换？
- 1.4.5 什么是 ASCII 码？试说明它的用途。

1.5 二值逻辑变量与基本逻辑运算

当 **0** 和 **1** 表示逻辑状态时，两个二进制数码按照某种指定的因果关系进行的运算称为逻辑运算。逻辑运算与算术运算完全不同，它所使用的数学工具是逻辑代数（又称为布尔代数）。与普通代数一样，它由逻辑变量和逻辑运算组成。变量可以用 A, B, C, x, y, z 等字母组成。所不同的是，在普通代数中，变量的取值可以是任意的，而在逻辑代数中，逻辑变量只有两个可取的值，即 **0** 和 **1**，因而称为二值逻辑变量。这里，**0** 和 **1** 并不表示数量的大小，而是用来表示完全对立的

逻辑状态。

在逻辑代数中,有与、或、非三种基本的逻辑运算。众所周知,运算是一种函数关系,它可以用语言描述,亦可用逻辑代数表达式描述,还可用表格或图形来描述。输入逻辑变量所有取值的组合与其对应的输出逻辑函数值构成的表格,称为真值表。用规定的逻辑符号表示的图形称为逻辑图。下面分别讨论三种基本的逻辑运算。

1. 与运算

图 1.5.1(a)所示为一个简单的与逻辑电路,电源 E 通过开关 A 和 B 向灯供电,只有 A 与 B 同时闭合时,灯才亮。 A 和 B 中只要有一个断开或者两个均断开时,则灯不亮。在这个电路中,开关 A 、 B 与灯 L 的逻辑关系是:“只有当一件事的几个条件全部具备之后,这件事才发生”。这种关系称为与逻辑。如果用二值逻辑 0 和 1 来表示开关和灯的状态,设开关断开和灯不亮均用 0 表示,而开关闭合和灯亮均用 1 表示,则可得出其真值表,如表 1.5.1 所示。其中 L 表示灯的状态。若用逻辑表达式来描述,则可写为

$$L = A \cdot B \quad (1.5.1)$$

式中小圆点“·”表示 A 、 B 的与运算,也称为逻辑乘。在不致引起混淆的前提下,乘号“·”被省略。在某些文献中,也采用符号 \wedge 、 \cap 表示与运算。能实现与运算的逻辑电路称为与门,其逻辑符号^①如图 1.5.1(b) 和图 1.5.1(c) 所示。图 1.5.1(b) 所示为特异形符号,图 1.5.1(c) 所示为矩形符号。

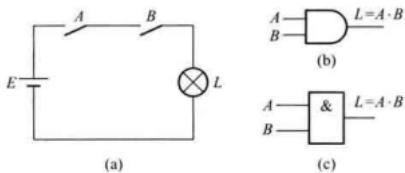


图 1.5.1 与逻辑运算

(a) 电路图 (b) 特异形符号 (c) 矩形符号

表 1.5.1 与逻辑真值表

A	B	$L = A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

2. 或运算

图 1.5.2(a)所示为一个简单的或逻辑电路,电源 E 通过开关 A 或 B 向灯供电。只要开关 A 或 B 闭合或两者均闭合,则灯亮。而当 A 和 B 均断开时,则灯不亮。此电路中,开关 A 、 B 与灯 L 的逻辑关系是:“当一件事情的几个条件中只要有一个条件得到满足,这件事就会发生”。这种关系称为或逻辑。或是指 A 闭合或 B 闭合,即任一个条件具备的意思。仿照前述,可以得出用 0、1 表示的或逻辑真值表如表 1.5.2 所示。若用逻辑表达式来描述,则可写为

$$L = A + B \quad (1.5.2)$$

^① 电气和电子工程师协会 (Institute of Electrical and Electronics Engineers, 简称 IEEE) 标准中,门电路的图形符号有两种,矩形符号和特异形符号(见附录 C)。中国国家标准 (GB/T4728.12—1996) 采用矩形符号。本书所用的逻辑图形符号为特异形符号。

式中符号“+”表示 A, B 的或运算，也表示逻辑加。在某些文献中，也采用符号 V、U 来表示或运算，能实现或运算的逻辑电路称为或门，其逻辑符号如图 1.5.2(b) 和图 1.5.2(c) 所示。图 1.5.2(b) 所示为特异形符号。图 1.5.2(c) 所示为矩形符号。

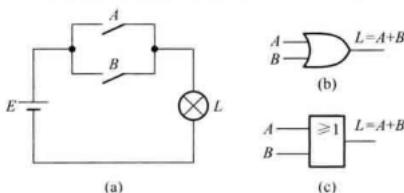


图 1.5.2 或逻辑运算

(a) 电路图 (b) 特异形符号 (c) 矩形符号

表 1.5.2 或逻辑真值表

A	B	$L = A + B$
0	0	0
0	1	1
1	0	1
1	1	1

3. 非运算

如图 1.5.3(a) 所示，电压 V 通过一继电器触点向灯供电，NC 为继电器 A 的动断触点，即通常 A 不通电，动断触点闭合，灯亮；而当 A 通电时，动断触点断开，灯不亮。由此可得出其逻辑关系为：“一件事情的发生是以其相反的条件为依据”。这种逻辑关系称为非逻辑。若用 0 和 1 来表示继电器和灯的状态， A 不通电和灯不亮用 0 表示，而 A 通电和灯亮用 1 表示，则得出非逻辑的真值表如表 1.5.3 所示。读者很容易看到， L 与 A 总是处于相反的逻辑状态。若用逻辑表达式来描述，则可写为

$$L = \bar{A} \quad (1.5.3)$$

式中字母 A 上方的短线“~”表示非运算。在某些文献中，也采用“~”、“ \neg ”或“‘’来表示非运算。在逻辑运算中，通常将 A 称为原变量，而将 \bar{A} 称为反变量或非变量。

能实现非运算的电路称为非门，也可以称为反相器，其逻辑符号如图 1.5.3(b)、(c) 所示，小圆圈表示非运算。

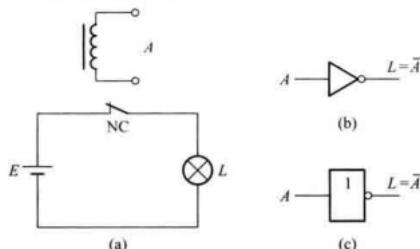


图 1.5.3 非逻辑运算

(a) 电路图 (b) 特异形符号 (c) 矩形符号

表 1.5.3 非逻辑真值表

A	$L = \bar{A}$
0	1
1	0

上述与、或逻辑运算可以推广到多变量的情况

$$L = A \cdot B \cdot C \cdots \quad (1.5.4)$$

$$L = A + B + C + \cdots \quad (1.5.5)$$

4. 几种常用逻辑运算

在实际逻辑运算中,除了与、或、非三种基本运算外,还经常使用一些其他的逻辑运算,例如与非、或非、异或和同或。

与非运算是与运算和非运算的组合。逻辑符号和真值表分别如图 1.5.4 和表 1.5.4 所示。逻辑表达式可写成

$$L = \overline{A \cdot B} \quad (1.5.6)$$

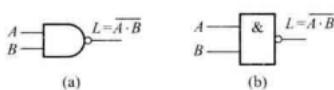


图 1.5.4 与非逻辑符号
(a) 特异形符号 (b) 矩形符号

表 1.5.4 与非逻辑真值表

A	B	L
0	0	1
0	1	1
1	0	1
1	1	0

或非运算是或运算和非运算的组合。逻辑符号和真值表分别如图 1.5.5 和表 1.5.5 所示。逻辑表达式可写成

$$L = \overline{A+B} \quad (1.5.7)$$

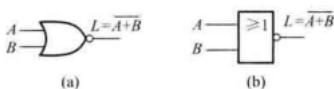


图 1.5.5 或非逻辑符号
(a) 特异形符号 (b) 矩形符号

表 1.5.5 或非逻辑真值表

A	B	L
0	0	1
0	1	0
1	0	0
1	1	0

异或的逻辑关系是:当两个输入状态相同时,输出为 0;当两个输入状态不同时,输出为 1。逻辑符号和真值表分别如图 1.5.6 和表 1.5.6 所示。逻辑表达式为

$$L = \overline{AB} + A\overline{B} = A \oplus B \quad (1.5.8)$$

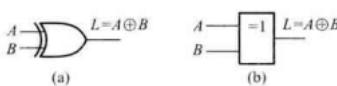


图 1.5.6 异或逻辑符号
(a) 特异形符号 (b) 矩形符号

表 1.5.6 异或逻辑真值表

A	B	L
0	0	0
0	1	1
1	0	1
1	1	0

同或和异或的逻辑关系刚好相反:当两个输入状态相同时,输出为 1;当两个输入状态不同

时,输出为 0。逻辑符号和真值表分别如图 1.5.7 和表 1.5.7 所示。逻辑表达式为

$$L = AB + \overline{A} \overline{B} = A \odot B \quad (1.5.9)$$

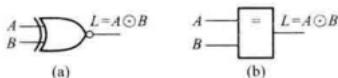


图 1.5.7 同或逻辑符号

(a) 特异形符号 (b) 矩形符号

表 1.5.7 同或逻辑真值表

A	B	L
0	0	1
0	1	0
1	0	0
1	1	1

复习思考题

1.5.1 试列举几种相关的实例说明与、或、非逻辑关系,并列出 3 种逻辑运算的表达式。

1.5.2 根据真值表判断异或和同或具有怎样的逻辑关系?

1.6 逻辑函数及其表示方法

从上一节介绍的逻辑运算中可以知道,逻辑变量分为两种:输入逻辑变量和输出逻辑变量。描述输入逻辑变量和输出逻辑变量之间的因果关系称为逻辑函数。由于逻辑变量是只取 0 或 1 的二值逻辑变量,因此逻辑函数也是二值逻辑函数。

1.6.1 逻辑函数的几种表示方法

一般来说,一个比较复杂的逻辑电路,往往是受多种因素控制的,就是说有多个逻辑变量。输出变量与输入变量之间的逻辑函数的描述方法有真值表、逻辑函数表达式、逻辑图、波形图和卡诺图等。下面举一个简单实例介绍前四种逻辑函数的表示。

图 1.6.1 是一个控制楼梯照明灯的电路,单刀双掷开关 A 装在楼下,B 装在楼上,这样在楼下开灯后,可在楼上关灯;同样,也可在楼上开灯,而在楼下关灯。因为只有当两个开关都向上扳或向下扳时,灯才亮;而一个向上扳、另一个向下扳时,灯就不亮。

1. 真值表

将输入变量所有可能的取值与相应的函数值列成表格,就得到真值表。

图 1.6.1 电路的逻辑关系可用真值表来描述。设 L 表示灯的状态,即 $L=1$ 表示灯亮, $L=0$ 表示灯不亮。用 A 和 B 表示开关 A 和开关 B 的位置,用 1 表示开关向上扳,用 0 表示开关向下扳。A,B 的取值有 4 种组合,00,01,10,11,对每一种组合确定输出 L 的值,则 L 与 A,B 逻辑关系的真值表如表 1.6.1 所示。

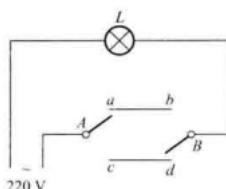


图 1.6.1 逻辑电路举例

表 1.6.1 图 1.6.1 的真值表

A	B	L
0	0	1
0	1	0
1	0	0
1	1	1

2. 逻辑函数表达式

逻辑表达式是用与、或、非等运算组合起来，表示逻辑函数与逻辑变量之间关系的逻辑代数式。

由表 1.6.1 所示真值表可知，在 A, B 状态的四种不同组合中，只有第一 ($A=B=0$) 和第四 ($A=B=1$) 两种组合才能使灯亮 ($L=1$)。这两种组合所对应的乘积项 \overline{AB} 和 AB 为 1，使 $L=1$ 。即输出 L 为两个乘积项之和，即

$$L = \overline{A}\overline{B} + AB \quad (1.6.1)$$

根据真值表写逻辑表达式的方法：逻辑变量之间是与的关系，而输出状态之间的组合则是或的关系。对于变量 A, B 或输出 L ，凡取 1 值的用原变量表示，取 0 值用反变量表示。

3. 逻辑图

用与、或、非等逻辑符号表示逻辑函数中各变量之间的逻辑关系所得到的图形称为逻辑图。

将式(1.6.1)中所有的与、或、非运算符号用相应的逻辑符号代替，并按照逻辑运算的先后次序将这些逻辑符号连接起来，就得到图 1.6.1 所示电路所对应的逻辑图如图 1.6.2(a)所示。式(1.6.1)表示的是同或逻辑关系，为简便起见，也可以用同或逻辑符号表示，得到图 1.6.2(b) 所示逻辑图。

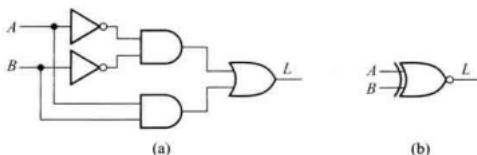


图 1.6.2 图 1.6.1 电路的逻辑图

(a) 由与、或、非逻辑符号构成的逻辑图 (b) 由同或逻辑符号构成的逻辑图

4. 波形图

对输入变量随时间变化的每一种取值，求出相应的输出值。并将输入和输出关系按时间顺

序依次排列得到的图形，称为波形图。

图 1.6.3 所示的波形图中，在 t_1 时间段内， A, B 输入端均为低电平 0，根据式(1.6.1)或表 1.6.1 可知，此时输出 L 为高电平 1。依照此方法，可得出 t_2, t_3 和 t_4 时间段内输出 L 的波形图。从图中可以直观地看出，对于同或逻辑关系，只要输入 A 和 B 相同，输出为 1； A 和 B 不相同时，输出为 0。

上述四种不同的表示方法所描述的是同一逻辑关系，因此它们之间有着必然的联系，可以从一种表示方法，得到其他表示方法。

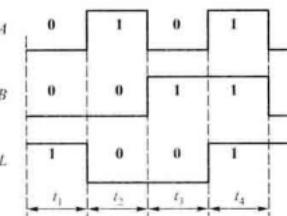


图 1.6.3 图 1.6.1 所示电路的波形图

1.6.2 逻辑函数表示方法之间的转换

逻辑函数的表示方法归纳起来有六种：真值表、逻辑函数表达式、逻辑图、波形图、卡诺图和 HDL。卡诺图及 HDL 表示方法将在下一章介绍。

同一逻辑问题可以用六种不同方法描述。因此，这些方法本质是相通的，可以相互转换。这里只介绍真值表与逻辑表达式及逻辑图之间的转换。

1. 真值表到逻辑图的转换

通常从给定的真值表不能直接得到逻辑图。首先根据真值表写出逻辑表达式，依照逻辑表达式画出逻辑图。

转换步骤：

- (1) 根据真值表写出逻辑表达式。
- (2) 用公式法或卡诺图法化简得到简化的逻辑表达式。
- (3) 根据逻辑表达式画出逻辑图。

例 1.6.1 已知真值表如表 1.6.2 所示，试画出它的逻辑图。

解：(1) 写出真值表中使输出为 1 的所有输入变量乘积项，凡取 1 值的用原变量表示，取 0 值的用反变量表示。然后将这些乘积项相加，即得到输出逻辑函数表达式。

当输入为 $A=0, B=1, C=1$ 时，使乘积项 $\bar{A}BC=1$ ，此时 $L=1$ 。同理乘积项 $AB\bar{C}$ 使 $L=1$ 。输出逻辑函数表达式就是这两个乘积项之和，即

$$L = \bar{A}BC + AB\bar{C} \quad (1.6.2)$$

(2) 公式法或卡诺图法化简将在第 2 章介绍，式(1.6.2)不需要化简。

(3) 将式(1.6.2)中所有的与、或、非运算符号用相应的逻辑符号代替，并按照逻辑运算的先后次序将它们连接起来，得到如图 1.6.4 所示的逻辑图。

表 1.6.2 例 1.6.1 的真值表

A	B	C	L
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

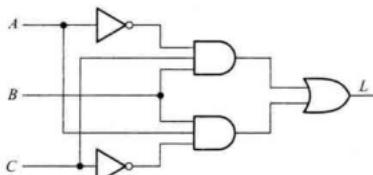


图 1.6.4 例 1.6.1 的逻辑图

2. 逻辑图到真值表的转换

从逻辑图不能直接得到真值表, 转换步骤如下:

(1) 从逻辑图的输入端到输出端, 逐级写出每个逻辑符号输出端的表达式, 直到写出最后的输出变量的逻辑表达式。

(2) 化简变换, 求简化的逻辑表达式。

(3) 将输入变量可能的取值逐个代入表达式进行计算, 并将结果列表, 即得真值表。

例 1.6.2 已知逻辑图如图 1.6.5 所示, 列出它的真值表。

解: 从输入端开始写出每个逻辑符号输出端的表达式, 得到最后的表达式为

$$L = \bar{A}B + A\bar{B}$$

该表达式不需要化简变换。

输入变量 A、B 取值有四种组合, 其中 $\bar{A}B$ 和 $A\bar{B}$ 使 L 为 1, 得到真值表如表 1.6.3 所示。

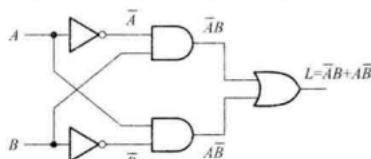


图 1.6.5 例 1.6.2 的逻辑图

表 1.6.3 例 1.6.2 的真值表

A	B	L
0	0	0
0	1	1
1	0	1
1	1	0

复习思考题

1.6.1 逻辑函数都有哪些表示方法?

1.6.2 你能根据给出的一种逻辑函数表示方法转换为其他的描述方式吗?

小 结

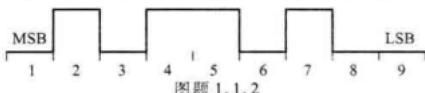
- 由于模拟信号具有连续性,实用上难于存储、分析和传输;数字电路或数字系统传输和处理**0** 和 **1** 表示的信息,因此,较易克服这些困难。
- 用**0** 和 **1** 可以组成二进制数表示数量的大小,也可以表示对立的两种逻辑状态。数字系统中常用二进制数来表示数值。所谓二进制是以 2 为基数的计数体制。
- 十六进制是二进制的简写,它是以 16 为基数的计数体制,常用于数字电子技术、微处理器、计算机和数据通信中。任意一种格式的数可以在十六进制、二进制和十进制之间相互转换。
- 与十进制数类似,二进制数也有加、减、乘、除四种运算,加法是各种运算的基础。二进制数可以用原码、反码或补码表示。在数字系统或计算机中常采用二进制补码表示有符号的数,并进行有关运算。
- 特殊二进制码常用来表示十进制数。如 8421 码、2421 码、5421 码、余 3 码、余 3 循环码、格雷码等。也有用 7 位二进制数来表示符号-数字混合码,如 ASCII 码。
- **与**、**或**、**非** 是逻辑运算中的三种基本运算,其他的逻辑运算可以由这三种基本运算构成。数字逻辑是计算机的基础。逻辑函数的描述方法有真值表、逻辑函数表达式、逻辑图、波形图和卡诺图等。

习 题

1.1 数字信号与数字电路

1.1.1 试按表 1.1.1 所列的数字集成电路的分类为依据,指出下列 IC 器件属于何种集成度器件:(1) 微处理器;(2) 计数器;(3) 加法器;(4) 逻辑门;(5) 4 位兆存贮器。

1.1.2 一数字信号波形如图题 1.1.2 所示,试问该波形所代表的二进制数是什么?



图题 1.1.2

1.1.3 试绘出下列二进制数的数字波形,设二进制数为串行方式,从左到右逻辑 **1** 的电压为 5 V,逻辑 **0** 的电压为 0 V。

(1) 0011001110011 (2) 0111010 (3) 11110111101

1.1.4 一周期性数字波形如图题 1.1.4 所示,试计算:(1) 周期;(2) 频率;(3) 古空比。

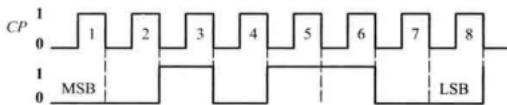


图题 1.1.4

1.2 数制

1.2.1 一数字波形如图题 1.2.1 所示,时钟频率为 4 kHz,试确定:

- (1) 它所表示的二进制数;(2) 串行方式传送 8 位数据所需要的时间;(3) 以 8 位并行方式传送数据所需要的时间。



图题 1.2.1

1.2.2 将下列二进制数转换为十进制数:

- (1) $(1011)_b$ (2) $(10111)_b$ (3) $(101101)_b$ (4) $(1001011)_b$

1.2.3 将下列二进制数转换为十进制数:

- (1) $(0.101)_b$ (2) $(0.0011)_b$ (3) $(0.10101)_b$ (4) $(0.100111)_b$

1.2.4 将下列二进制数转换为十进制数:

- (1) $(10.01)_b$ (2) $(110.101)_b$ (3) $(1110.1001)_b$ (4) $(10111.01101)_b$

1.2.5 将下列二进制数转换为八进制和十六进制数:

- (1) $(101001)_b$ (2) $(1011.0101)_b$ (3) $(11010.11001)_b$ (4) $(110111.011101)_b$

1.2.6 将下列十进制数转换为二进制数、八进制数和十六进制数:

- (1) $(43)_b$ (2) $(59)_b$ (3) $(127)_b$ (4) $(234)_b$

1.2.7 将下列十进制数转换为二进制数、八进制数和十六进制数(要求二进制数保留小数点后 8 位):

- (1) $(0.9)_b$ (2) $(0.34)_b$ (3) $(0.028)_b$ (4) $(0.7182)_b$

1.2.8 将下列十进制数转换为二进制数、八进制数和十六进制数(要求转换误差不大于 2^{-4}):

- (1) $(4.8)_b$ (2) $(15.27)_b$ (3) $(254.35)_b$ (4) $(1002.456)_b$

1.2.9 将下列十六进制数转换为二进制数:

- (1) $(7)_b$ (2) $(D4)_b$ (3) $(23F.45)_b$ (4) $(A040.51)_b$

1.2.10 将下列十六进制数转换为十进制数:

- (1) $(13)_b$ (2) $(103.2)_b$ (3) $(A3.0C)_b$ (4) $(A45D.0BC)_b$

1.2.11 试比较下列各数,并指出从大到小的排列顺序:

- (1) $(74)_b$ (2) $(1101101)_b$ (3) $(165)_b$ (4) $(10E)_b$

1.3 二进制数的算术运算

1.3.1 写出下列二进制数的原码、反码和补码:

- (1) $(+1110)_b$ (2) $(+10110)_b$ (3) $(-1110)_b$ (4) $(-10110)_b$

1.3.2 写出下列有符号二进制补码所表示的十进制数:

- (1) 01011 (2) 0010111 (3) 11101000 (4) 11011001

1.3.3 试用 8 位二进制补码表示下列十进制数:

- (1) +11 (2) +68 (3) -24 (4) -90

1.3.4 试用 8 位二进制补码计算下列各式,并用十进制数表示结果:

- (1) $12+9$ (2) $11-3$ (3) $-29-25$ (4) $-120+30$

1.4 二进制代码

- 1.4.1 将下列十进制数转换为 8421BCD 码：

- (1) 43 (2) 127 (3) 254.25 (4) 2.718

- 1.4.2 将下列数码分别作为自然二进制数和 8421BCD 码时，求出相应的十进制数。

- (1) 10010111 (2) 100010010011 (3) 000101001001 (4) 10000100.10010001

- 1.4.3 将下列二进制数转换为格雷码：

- (1) 101 (2) 1011 (3) 11010 (4) 101101

- 1.4.4 将下列格雷码转换为二进制数：

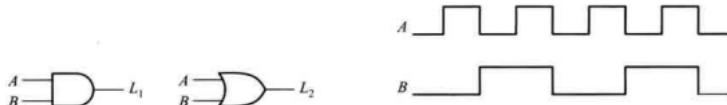
- (1) 110 (2) 1111 (3) 10110 (4) 110111

- 1.4.5 试用十六进制数写出下列字符的 ASCII 码的表示：

- (1) + (2) @ (3) you (4) 43

1.5 二值逻辑变量与基本逻辑运算

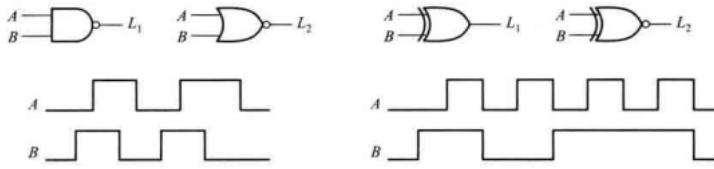
- 1.5.1 在图题 1.5.1 中，已知输入信号 A、B 的波形，画出各逻辑门输出的波形。



图题 1.5.1

- 1.5.2 在图题 1.5.2 中，已知输入信号 A、B 的波形，画出各逻辑门输出的波形。

- 1.5.3 在图题 1.5.3 中，已知输入信号 A、B 的波形，画出各逻辑门输出的波形。



图题 1.5.2

图题 1.5.3

1.6 逻辑函数及其表示方法

- 1.6.1 已知逻辑函数的真值表如表题 1.6.1 所示，试写出其逻辑函数表达式。

表题 1.6.1 题 1.6.1 的真值表

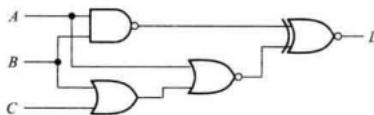
A	B	C	L
0	0	0	1
0	0	1	0
0	1	0	0

续表

A	B	C	L
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

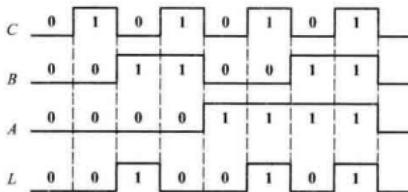
1.6.2 已知逻辑函数 $L = A + \overline{B}C + \overline{A}B\overline{C}$, 试求其对应的真值表。

1.6.3 已知逻辑图如图题 1.6.3 所示, 试求 L 的逻辑函数表达式。



图题 1.6.3

1.6.4 已知逻辑函数 L 的波形图如图题 1.6.4 所示, 试求其真值表和逻辑函数表达式。



图题 1.6.4

2 >>>

逻辑代数与硬件描述语言基础

引
言

本章将首先介绍分析和设计数字电路的数学工具——逻辑代数^①,从逻辑变量、基本定律和定理、逻辑函数及其化简方法逐步加以讨论。然后介绍在数字电路仿真和设计中使用的一种硬件描述语言——Verilog HDL 的基础知识。

2.1 逻辑代数的基本定律和规则

逻辑代数是 1854 年问世的,最早用于开关和继电器网络的分析及化简,随着半导体器件制造工艺的发展,各种性能良好的微电子开关器件不断涌现,逻辑代数成为分析和设计逻辑电路不可缺少的数学工具。利用这种数学工具,可以把逻辑电路输入和输出之间的关系用代数方程表示出来。

逻辑代数有一系列的定律、定理和规则,用它们对数学表达式进行处理,可以完成对逻辑电路的化简、变换、分析和设计。

2.1.1 逻辑代数的基本定律和恒等式

表 2.1.1 列出了常用的逻辑代数基本定律和恒等式。等式中的字母(例如 A, B, C)为逻辑变量,其值可以取 0 或 1,代表逻辑信号的两种可能状态之一。

表 2.1.1 逻辑代数定律、定理和恒等式

基本定律	或	与	非
$A + 0 = A$		$A \cdot 1 = A$	
$A + 1 = 1$		$A \cdot 0 = 0$	
$A + A = A$		$A \cdot A = A$	$\overline{\overline{A}} = A$

^① 又称布尔代数,是英国数学家 G. Boole (1815—1864) 提出的。

续表

基本定律	或	与	非
	$A + \bar{A} = 1$	$A \cdot \bar{A} = 0$	
交换律	$A + B = B + A$	$A \cdot B = B \cdot A$	
结合律	$(A + B) + C = A + (B + C)$	$(A \cdot B) \cdot C = A \cdot (B \cdot C)$	
分配律	$A \cdot (B + C) = A \cdot B + A \cdot C$	$A + B \cdot C = (A + B) \cdot (A + C)$	
吸收律	$A + A \cdot B = A$	$A \cdot (A + B) = A$	
反演律(摩根定理①)	$\overline{A + B + C \dots} = \bar{A} + \bar{B} + \bar{C} + \dots$	$\overline{A + B + C \dots} = \bar{A} \cdot \bar{B} \cdot \bar{C} \dots$	
其他常用恒等式	$A + \bar{A} \cdot B = A + B$ $A \cdot B + \bar{A} \cdot C + B \cdot C = A \cdot B + \bar{A} \cdot C + A \cdot B + \bar{A} \cdot C + B \cdot C + D = A \cdot B + \bar{A} \cdot C + D$	$A \cdot (\bar{A} + B) = A \cdot B$	

表 2.1.1 中除了逻辑非运算外, 其他基本定律或恒等式是成对出现的, 具有对偶性。用完全归纳法可以证明表 2.1.1 所列等式的正确性, 方法是: 列出等式左边函数与右边函数的真值表, 如果等式两边的真值表相同, 说明等式成立。

例如, 要证明恒等式 $A + \bar{A} \cdot B = A + B$ 时, 按变量 A, B 所有可能的取值情况列出真值表如表 2.1.2 所示。表中第 3 列和第 4 列的结果相同, 故等式 $A + \bar{A} \cdot B = A + B$ 成立。

表 2.1.2 恒等式 $A + \bar{A} \cdot B = A + B$ 的证明

A	B	$\bar{A} \cdot B$	$A + \bar{A} \cdot B$	$A + B$
0	0	0	$0 + 0 = 0$	0
0	1	1	$0 + 1 = 1$	1
1	0	0	$1 + 0 = 1$	1
1	1	0	$1 + 0 = 1$	1

表 2.1.1 中的常用恒等式可以用其他基本定律加以证明, 下面证明其中的一条。

$$\begin{aligned}
 A \cdot B + \bar{A} \cdot C + B \cdot C &= A \cdot B + \bar{A} \cdot C + (A + \bar{A}) \cdot B \cdot C && \text{(运用定律 } A + \bar{A} = 1 \text{)} \\
 &= A \cdot B + \bar{A} \cdot C + A \cdot B \cdot C + \bar{A} \cdot B \cdot C && \text{(运用定律 } A \cdot (B + C) = A \cdot B + A \cdot C \text{)} \\
 &= A \cdot B \cdot (1 + C) + \bar{A} \cdot C \cdot (1 + B) && \text{(运用定律 } A \cdot (B + C) = A \cdot B + A \cdot C \text{)} \\
 &= A \cdot B + \bar{A} \cdot C && \text{(运用定律 } A + 1 = 1 \text{ 和 } A \cdot 1 = A \text{)}
 \end{aligned}$$

① 系 De Mogen 的译称。

为了进一步简化表达式的形式，在不会产生混淆的前提下，可以省略与运算符“·”，因此上面的表达式可以写成

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

这个恒等式说明，若两个乘积项中分别包含因子 A 和 \bar{A} ，而这两个乘积项的其余因子组成第三个乘积项时，则第三个乘积项是多余的，可以消去。

在以上所有定律中，数学家摩根提出的反演律具有特殊重要的意义。反演律又称为摩根定理，它经常用于求一个原函数的非函数或者对逻辑函数进行变换。以两个变量为例，证明摩根定理的正确性，即 $A \cdot B = \bar{A} + \bar{B}$ 和 $A + B = \bar{A} \cdot \bar{B}$ 。将 A, B 所有可能的取值情况列出，并求出相应表达式的结果，得到表 2.1.3 所示的真值表。分别将表中第 3 列和第 4 列进行比较、第 5 列和第 6 列进行比较，可见上面每个等式两边的真值表相同，故等式成立。

表 2.1.3 摩根定理的证明

$A \cdot B$	$\bar{A} \cdot \bar{B}$	$\overline{A \cdot B}$	$\overline{\bar{A} \cdot \bar{B}}$	$\overline{\overline{A \cdot B}}$	$\overline{\bar{A} \cdot \bar{B}}$
0 0	1 1	$\overline{0 \cdot 0} = 1$	1	$\overline{0 \cdot 0} = 1$	1
0 1	1 0	$\overline{0 \cdot 1} = 1$	1	$\overline{0 \cdot 1} = 1$	0
1 0	0 1	$\overline{1 \cdot 0} = 1$	1	$\overline{1 \cdot 0} = 1$	0
1 1	0 0	$\overline{1 \cdot 1} = 0$	0	$\overline{1 \cdot 1} = 0$	0

上面这些基本定理对化简逻辑函数表达式十分有用，也就是说，可以用来减少表达式中乘积项的数量。

例 2.1.1 试化简下列逻辑函数表达式：

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

解：运用分配律将表达式展开，得到

$$\begin{aligned} L &= A \cdot \bar{A} + A \cdot B + B \cdot \bar{A} + B \cdot B \\ &= 0 + A \cdot B + B \cdot \bar{A} + B \quad (\text{运用定理 } A \cdot \bar{A} = 0 \text{ 和 } A \cdot A = A) \\ &= A \cdot B + B \cdot \bar{A} + B \quad (\text{运用定理 } A+0=A) \\ &= B \cdot (A+\bar{A}+1) \quad (\text{运用定理 } A \cdot (B+C) = A \cdot B + A \cdot C) \\ &= B \cdot 1 = B \quad (\text{运用定理 } A+1=1 \text{ 和 } A \cdot 1=A) \end{aligned}$$

本节所列出的基本公式反映了逻辑关系，而不是数量之间的关系，在运算中不能简单套用初等代数的运算规则。如初等代数中的移项规则就不能用，这是因为逻辑代数中没有减法的缘故。这一点在使用时必须注意。

2.1.2 逻辑代数的基本规则

1. 代入规则

在任何一个逻辑等式中,如果用一个函数代替等式两边出现的某变量 A ,则等式依然成立,这个规则称为代入规则。

例如,在 $B \cdot (A+C) = B \cdot A + B \cdot C$ 中,将所有出现 A 的地方都用函数 $E+F$ 代替,则等式仍成立,即得

$$B \cdot [(E+F)+C] = B \cdot (E+F) + B \cdot C = B \cdot E + B \cdot F + B \cdot C$$

对所有基本定律或定理都可以使用代入规则。例如对于二变量的摩根定理 $\overline{A \cdot B} = \overline{A} + \overline{B}$,若用 $L = C \cdot D$ 替代等式中的 A ,则 $\overline{(C \cdot D) \cdot B} = \overline{C \cdot D} + \overline{B} = \overline{C} + \overline{D} + \overline{B}$,以此类推,任意多个变量的摩根定理都成立。

2. 反演规则

根据摩根定理,由原函数 L 的表达式,求它的非函数 \overline{L} 时,可以将 L 中的与(·)换成或(+),或(+)换成与(·);再将原变量换为非变量(如 A 换成 \overline{A}),非变量换为原变量;并将 1 换成 0,0 换成 1;那么所得的逻辑函数式就是 \overline{L} 。这个规则称为反演规则。

利用反演规则,可以比较容易地求出一个原函数的非函数。运用反演规则时必须注意以下两个原则:

(1) 保持原来的运算优先级,即先进行与运算,后进行或运算。并注意优先考虑括号内的运算。

(2) 对于非变量以外的非号应保留不变。

例 2.1.2 试求 $L = \overline{A} \cdot \overline{B} + C \cdot D + 0$ 的非函数 \overline{L} 。

解:按照反演规则,得

$$\overline{L} = (\overline{A+B}) \cdot (\overline{\overline{C}+\overline{D}}) \cdot \overline{1} = (\overline{A+B}) \cdot (\overline{\overline{C}+\overline{D}})$$

例 2.1.3 试求 $L = A + B \cdot \overline{C} + D + \overline{E}$ 的非函数 \overline{L} 。

解:按照反演规则,并保留非变量以外的非号不变,得

$$\overline{L} = \overline{A} \cdot (\overline{B+C}) \cdot \overline{\overline{D+E}}$$

3. 对偶规则

设 L 是一个逻辑表达式,若把 L 中的“与、或互换,0、1 互换”,那么就得到一个新的逻辑函数式,这就是 L 的对偶式,记作 L' 。与、或互换就是把与(·)换成或(+),或(+)换成与(·);0、1 互换就是把 1 换成 0,0 换成 1。变换时需注意保持原式中“先括号、然后与、最后或”的运算顺序。

例如, $L = (\overline{A+B}) \cdot (\overline{A+C})$, 则 $L' = A \cdot \overline{B} + A \cdot \overline{C}$ 。

对偶规则为：当某个逻辑表达式相等，则它们的对偶式也相等。

对偶性意味着逻辑代数中每个逻辑恒等式可以用两种不同的表达式进行表示。即任何一个定理的对偶表达式也是正确的。例如，表 2.1.1 中吸收律 $A + \bar{A} \cdot B = A + B$ 成立，利用对偶规则，它的对偶式 $A \cdot (\bar{A} + B) = A \cdot B$ 也是成立的。

复习思考题

2.1.1 (1) $A+1=?$ (2) $A \cdot 0=?$ (3) $A \cdot \bar{A}=?$ (4) $A+A=?$

2.1.2 试用完全归纳法证明 $A+B \cdot C=(A+B) \cdot (A+C)$ 。

2.1.3 写出三变量摩根定理的表达式，并用完全归纳法进行证明。

2.2 逻辑函数表达式的形式

任何一个逻辑函数，其表达式的形式都不是唯一的。下面先介绍逻辑函数的“与-或”、“或-与”表达式，接着介绍最小项的概念及逻辑函数的“最小项之和”表达式，然后再介绍最大项的概念及逻辑函数的“最大项之积”表达式。

2.2.1 逻辑函数表达式的基本形式

1. 与-或表达式

与-或表达式是指由若干与项进行或逻辑运算构成的表达式。例如有一个逻辑函数式为

$$L=A \cdot C + \bar{C} \cdot D$$

式中， $A \cdot C$ 和 $\bar{C} \cdot D$ 两项都是由与（逻辑乘）运算把变量连接起来的，故称为与项（或乘积项），然后将这两个与项用或运算符连接起来，称这种类型的表达式为与-或式，或称之为“积之和（Sum of Products，简称 SOP）”表达式。

2. 或-与表达式

或-与表达式是指由若干或项进行与逻辑运算构成的表达式。例如有一个逻辑函数式为

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

式中， $(A+C)$ 和 $(B+\bar{C})$ 两项是由或（逻辑加）运算符把变量连接起来的，故称为或项， D 是单个变量，可以认为是 $(D+0)$ ，然后将这三个或项用与运算符连接起来，称这种类型的表达式为或-与式，或称之为“和之积（Products of Sum，简称 POS）”表达式。

通常可以将逻辑函数式表示成混合形式。例如，函数 $L=A \cdot (B \cdot C + \bar{B} \cdot \bar{C}) + A \cdot (B \cdot \bar{C} + \bar{B} \cdot C)$ ，既不是与-或式，也不是或-与式，但经过变换可以转化成上述两种基本形式。

2.2.2 最小项与最小项表达式

逻辑函数的最小项表达式是建立在最小项基础之上的,下面先介绍最小项的定义和性质。

1. 最小项的定义和性质

对于有 n 个变量的逻辑函数,若有一个乘积项包含了全部的 n 个变量,每个变量都以它的原变量或非变量的形式在乘积项中出现,且仅出现一次,则称该乘积项为最小项。例如,三个变量 A, B, C 的最小项有 $\bar{A}\bar{B}\bar{C}$, $AB\bar{C}$, ABC 等,而 $\bar{A}B$, $A\bar{B}\bar{C}\bar{A}$, $A(B+C)$ 等则不是最小项。

一般 n 个变量的最小项应有 2^n 个, 最小项通常用 m_i 表示, 下标 i 即最小项编号, 用十进制数表示。将最小项中的原变量用 **1** 表示, 非变量用 **0** 表示, 可得到最小项的编号。例如, 以三个变量的乘积项 \overline{ABC} 为例, 它的二进制取值为 **011**, 可以用十进制数 3 表示, 所以把最小项 \overline{ABC} 记作 m_3 。三个变量 A, B, C 的全部 8 个最小项及其最小项的代表符号如表 2.2.1 所示。

表 2.2.1 三变量最小项、最大项编号表

行号	变量取值			最小项	最大项
	A	B	C		
0	0	0	0	$m_0 = \overline{A} \overline{B} \overline{C}$	$M_0 = A + B + C$
1	0	0	1	$m_1 = \overline{A} \overline{B} C$	$M_1 = A + B + \overline{C}$
2	0	1	0	$m_2 = \overline{A} B \overline{C}$	$M_2 = A + \overline{B} + C$
3	0	1	1	$m_3 = \overline{A} B C$	$M_3 = A + \overline{B} + \overline{C}$
4	1	0	0	$m_4 = A \overline{B} \overline{C}$	$M_4 = \overline{A} + B + C$
5	1	0	1	$m_5 = A \overline{B} C$	$M_5 = \overline{A} + B + \overline{C}$
6	1	1	0	$m_6 = A B \overline{C}$	$M_6 = \overline{A} + \overline{B} + C$
7	1	1	1	$m_7 = A B C$	$M_7 = \overline{A} + \overline{B} + \overline{C}$

为了分析最小项的性质，我们列出三个变量 A, B, C 所有最小项的真值表，如表 2.2.2 所示。

表 2.2.2 三变量最小项真值表

观察表 2.2.2 可以看出, 最小项具有下列性质。

(1) 任意一个最小项, 输入变量只有一组取值使其值为 1, 而其他各组取值均使其为 0。并且, 最小项不同, 使其值为 1 的输入变量取值也不同。以 $AB\bar{C}$ 为例, 只有当 ABC 取 110 时, 最小项 $m_6 = AB\bar{C} = 1$, 取其他值时, m_6 均为 0。

(2) 任意两个不同的最小项之积为 0。

(3) 所有最小项之和为 1。

2. 最小项表达式

由若干最小项相或构成的逻辑表达式称为最小项表达式, 也称为标准与-或表达式。

例 2.2.1 将逻辑函数 $L(A, B, C) = AB + \bar{A}C$ 变换成最小项之和表达式。

解: 利用公式 $A + \bar{A} = 1$, 将逻辑函数中的每一个乘积项都化成包含所有变量 A, B, C 的项, 即

$$\begin{aligned} L(A, B, C) &= AB + \bar{A}C = AB(C + \bar{C}) + \bar{A}(B + B)\bar{C} \\ &= ABC + AB\bar{C} + \bar{A}BC + \bar{A}\bar{B}\bar{C} \end{aligned}$$

此式为四个最小项之和, 是一个标准“与-或”表达式。为了简便, 在表达式中常用最小项的编号表示, 上式又可写为

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

由此可见, 任何一个逻辑函数都能变换为唯一的最小项表达式。

例 2.2.2 将逻辑函数 $L(A, B, C) = (\overline{AB + \bar{A}B + C})\overline{AB}$ 变换成最小项表达式。

解: (1) 多次利用摩根定理去掉非号, 直至最后得到一个只在单个变量上有非号的表达式, 即

$$\begin{aligned} L(A, B, C) &= (\overline{AB + \bar{A}B + C})\overline{AB} = (\overline{AB + \bar{A}B + C}) + AB \\ &= (\overline{AB} \cdot \overline{\bar{A}B} \cdot C) + AB = (\overline{A + B})(A + B)C + AB \end{aligned}$$

(2) 利用分配律去掉括号, 直至得到一个与-或表达式

$$\begin{aligned} L(A, B, C) &= (\overline{A + B})(A + B)C + AB \\ &= \overline{ABC} + A\bar{B}C + AB \end{aligned}$$

(3) 式中 AB 不是最小项, 用 $(C + \bar{C})$ 进行配项, 可得

$$\begin{aligned} L(A, B, C) &= \overline{ABC} + A\bar{B}C + AB(C + \bar{C}) \\ &= \overline{ABC} + A\bar{B}C + ABC + AB\bar{C} \\ &= m_3 + m_5 + m_6 + m_7 \\ &= \sum m(3, 5, 6, 7) \end{aligned}$$

2.2.3 最大项与最大项表达式

逻辑函数的最大项表达式是建立在最大项基础之上的, 下面先介绍最大项的定义和性质。

1. 最大项的定义和性质

对于有 n 个变量的函数来说, 若有一个或项包含了全部的 n 个变量, 每个变量都以它的原变量或非变量的形式在或项中出现, 且仅出现一次, 则称该或项为最大项。

一般 n 个变量的最大项应有 2^n 个。最大项通常用 M_i 表示, 下标编号 i 用于区别不同的最大项。对于一个最大项, 输入变量只有一组二进制数使其取值为 0, 与该二进制数对应的十进制数就是该最大项的下标编号。

三个变量 A, B, C 的全部 8 个最大项及其最大项的代表符号如表 2.2.1 所示。表中每一行列出的二进制数都使与其对应的最大项的值为 0。例如, A, B, C 的取值为 010, 其对应十进制数为 2, 它使最大项 $(A+B+C)=0$, 所以把 $(A+B+C)$ 记作 M_2 。

根据最大项的定义, 得到最大项具有下列性质:

- (1) 任意一个最大项, 输入变量只有一组取值使得它的值为 0, 而在变量取其他各组值时, 这个最大项的值都是 1。并且, 最大项不同, 使其值为 0 的输入变量取值也不同。
- (2) 任意两个不同的最大项之和为 1。
- (3) 所有最大项之积为 0。

2. 最小项和最大项的关系

根据最小项和最大项的性质可知, 相同变量构成的最小项与最大项之间存在互补关系, 即

$$m_i = \overline{M}_i \quad \text{或者} \quad M_i = \overline{m}_i$$

例如, $m_2 = \overline{AB}\ \overline{C}$, 则 $\overline{m_2} = \overline{\overline{AB}}\ \overline{\overline{C}} = A+B+C = M_2$ 。

例 2.2.3 将逻辑函数 $L(A, B, C) = A \cdot B + \overline{A} \cdot C$ 变换成最大项之积表达式。

解: (1) 多次利用摩根定律, 将函数表达式变换成或-与表达式, 即

$$\begin{aligned} L(A, B, C) &= \overline{A \cdot B + \overline{A} \cdot C} = \overline{(\overline{A} + B)} \cdot \overline{(A + \overline{C})} \\ &= \overline{\overline{A} \cdot A + A \cdot \overline{B} + \overline{A} \cdot C + \overline{B} \cdot \overline{C}} \\ &= (\overline{A} + B) \cdot (A + C) \cdot (B + C) \end{aligned}$$

(2) 利用公式 $\overline{A} \cdot A = 0$ 和 $A + B \cdot C = (A + B) \cdot (A + C)$, 将或-与式中非最大项扩展成最大项, 即

$$\begin{aligned} L(A, B, C) &= (\overline{A} + B + 0) \cdot (A + C + 0) \cdot (B + C + 0) \\ &= (\overline{A} + B + C \cdot \overline{C}) \cdot (A + C + B \cdot \overline{B}) \cdot (B + C + A \cdot \overline{A}) \\ &= (\overline{A} + B + C) \cdot (\overline{A} + B + \overline{C}) \cdot (A + C + B) \cdot (A + C + \overline{B}) \cdot (B + C + A) \cdot (B + C + \overline{A}) \\ &= (\overline{A} + B + C) \cdot (\overline{A} + B + \overline{C}) \cdot (A + B + C) \cdot (A + \overline{B} + C) \end{aligned}$$

此式为四个最大项之积, 是一个标准“或-与”表达式。

为了简便，在表达式中常用最大项（或最大项的下标编号）表示，上式又可写为

$$L(A, B, C) = M_4 \cdot M_5 \cdot M_6 \cdot M_2 = \prod M(0, 2, 4, 5)$$

由此可见，任何一个逻辑函数经过变换，都能表示成唯一的最大项表达式。

例 2.2.4 一个逻辑电路有三个输入逻辑变量 A, B, C ，它的真值表如表 2.2.3 所示，试写出该逻辑函数的最小项表达式和最大项表达式。

表 2.2.3 例 2.2.4 的真值表

行号	输入变量			输出
	A	B	C	L
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	$1 \rightarrow m_3$
4	1	0	0	0
5	1	0	1	$1 \rightarrow m_5$
6	1	1	0	$1 \rightarrow m_6$
7	1	1	1	0

解：(1) 根据真值表求最小项表达式的一般方法是：

- ① 写出使函数 $L=1$ 的各行所对应的最小项；
- ② 将这些最小项相加，即得到最小项表达式。

$$L(A, B, C) = m_3 + m_5 + m_6$$

$$= \sum m(3, 5, 6)$$

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

(2) 根据真值表求最大项表达式的一般方法是：

- ① 写出使函数 $L=0$ 的各行所对应的最大项；
- ② 将这些最大项相乘，即得到最大项表达式。

$$L(A, B, C) = M_0 \cdot M_1 \cdot M_2 \cdot M_4 \cdot M_7$$

$$= \prod M(0, 1, 2, 4, 7)$$

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

由上面的推导可知，由未出现在最小项表达式中的各标号组成的大项之积就构成了最大项表达式，反之亦然。

复习思考题

- 2.2.1** 对于有 n 个变量的函数来说，它的最小项有多少个？每个最小项中有多少个变量？

2.2.2 对于 n 个变量的函数,任意两个最小项的乘积是多少? 全体最小项之和是多少?

2.2.3 相同变量构成的最小项与最大项之间存在什么关系? 试以两个变量为例进行说明。

2.2.4 已知逻辑函数的最小项表达式 $L(A,B,C,D) = \sum m(1,2,3,7,8,9,10,11,12,14)$, 试写出其最大项表达式。

2.3 逻辑函数的代数化简法

在设计逻辑电路时,根据实际要求直接归纳出来的逻辑函数式往往不是最简的形式,这就需要对逻辑函数式进行化简,其目的是为了降低电路实现的成本,以较少的门实现电路。

例如,图 2.3.1(a) 所示电路可以简化为图 2.3.1(b) 所示的电路,两个电路具有完全相同的逻辑功能。显然,简化电路使用较少的门,比原来电路的体积小且成本低。简化电路的连线较少,减少了电路可能潜在的故障,可靠性得到进一步的改善。

简化逻辑函数的方法很多,本节将介绍代数化简法,下一节介绍卡诺图法。

2.3.1 逻辑函数的最简形式

一个逻辑函数可以有多种不同的逻辑表达式,如与-或表达式、与非-与非表达式、或-与表达式、或非-或非表达式以及与-或-非表达式等。例如:

$$\begin{aligned}
 L &= AC + \overline{CD} && \text{与-或表达式} \\
 &= \overline{\overline{A} \overline{C} + \overline{C} \overline{D}} && \text{与非-与非表达式} \\
 &= \overline{(A+C)(C+D)} && \text{或-与表达式} \\
 &= \overline{\overline{(A+C)} + \overline{(C+D)}} && \text{或非-或非表达式} \\
 &= \overline{\overline{AC} + \overline{CD}} && \text{与-或-非表达式}
 \end{aligned}$$

以上是同一函数五种不同形式的最简表达式。通常与-或表达式易于转换为其他类型的函数式,所以下面着重讨论与-或表达式的化简。

在若干个具有相同逻辑关系的与-或表达式中,将其中包含的乘积项个数最少,且每个乘积项中变量数最少的表达式称为最简与-或表达式。

逻辑函数化简就是要消去与-或表达式中多余的乘积项和每个乘积项中多余的变量,以得

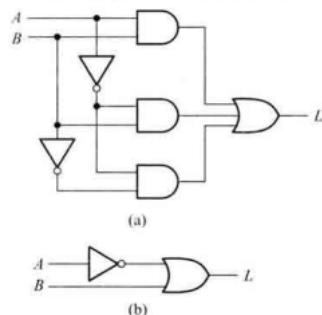


图 2.3.1 具有相同逻辑功能的两种电路实现
(a) 标准积之和的电路 (b) 成本最低的电路

到逻辑函数的最简与-或表达式。有了最简与-或表达式以后,再用公式变换就可以得到其他类型的函数式。

2.3.2 逻辑函数的代数化简法

1. 逻辑函数的化简

代数法是运用逻辑代数中的定理、恒等式或规则对逻辑函数进行化简,这种方法需要一些技巧,没有固定的步骤。下面是经常使用的方法。

(1) 并项法

利用公式 $A+\bar{A}=1$, 将两项合并成一项, 并消去一个变量。

例 2.3.1 试用并项法化简逻辑函数表达式 $L=A(BC+\bar{B}\bar{C})+A(\bar{B}\bar{C}+\bar{B}C)$ 。

$$\begin{aligned} \text{解: } L &= ABC+A\bar{B}\bar{C}+AB\bar{C}+A\bar{B}C \\ &= AB(C+\bar{C})+A\bar{B}(C+\bar{C}) \\ &= A(B+\bar{B})=A \end{aligned}$$

(2) 吸收法

利用公式 $A+AB=A$, 消去多余的项 AB 。根据代入规则, A, B 可以是任何一个复杂的逻辑式。

例 2.3.2 试用吸收法化简逻辑函数表达式 $L=\bar{A}B+\bar{A}BCDE+\bar{A}BCDF$ 。

$$\text{解: } L=\bar{A}B+\bar{A}BCD(E+F)=\bar{A}B$$

(3) 消去法

利用 $A+\bar{A}B=A+B$, 消去多余的因子。

例 2.3.3 试用消去法化简逻辑函数表达式 $L=AB+\bar{A}C+\bar{B}C$ 。

$$\begin{aligned} \text{解: } L &= AB+(\bar{A}+\bar{B})C \\ &= AB+\bar{A}C \\ &= AB+C \end{aligned}$$

(4) 配项法

先利用 $A=A(B+\bar{B})$, 增加必要的乘积项, 再用并项或吸收的办法使项数减少。

例 2.3.4 试用配项法化简逻辑函数表达式 $L=AB+\bar{A}\bar{C}+B\bar{C}$ 。

$$\begin{aligned} \text{解: } L &= AB+\bar{A}\bar{C}+(A+\bar{A})B\bar{C} \\ &= AB+\bar{A}\bar{C}+AB\bar{C}+\bar{A}B\bar{C} \\ &= (AB+AB\bar{C})+(\bar{A}\bar{C}+\bar{A}B\bar{C}) \\ &= AB+\bar{A}\bar{C} \end{aligned}$$

使用配项的方法要有一定的经验, 否则越配越繁。通常对逻辑表达式进行化简, 要综合使用

上述技巧。以下再举一例。

例 2.3.5 化简 $L = AD + A\bar{D} + AB + \bar{A}C + BD + A\bar{B}EF + \bar{B}EF$

$$\text{解: } L = A(D + \bar{D}) + AB + \bar{A}C + BD + A\bar{B}EF + \bar{B}EF \quad (\text{利用 } A + \bar{A} = 1)$$

$$= A + \bar{A}C + BD + \bar{B}EF \quad (\text{利用 } A + AB = A)$$

$$= A + C + BD + \bar{B}EF \quad (\text{利用 } A + \bar{AB} = A + B)$$

2. 逻辑函数形式的变换

如果用门电路实现前面介绍的逻辑函数, 需要用到非门、与门和或门。通常在一片集成电路芯片中只有一种门电路, 为了减少门电路的种类, 需要对逻辑函数表达式进行变换。

例 2.3.6 已知逻辑函数表达式为

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

要求:(1) 试求最简的与-或逻辑函数表达式, 并画出相应的逻辑图;

(2) 仅用与非门画出表达式的逻辑图。

解: $L = \bar{A}B(\bar{D} + D) + A\bar{B}\bar{D} + A\bar{B}(C + C)D \quad (\text{分配律})$

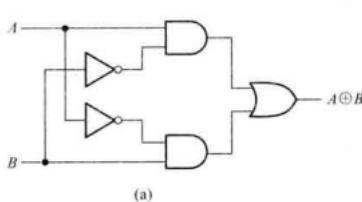
$$= \bar{A}B + A\bar{B}\bar{D} + A\bar{B}D \quad (\text{利用 } A + \bar{A} = 1)$$

$$= \bar{A}B + A\bar{B}(D + \bar{D}) \quad (\text{利用 } A + \bar{A} = 1)$$

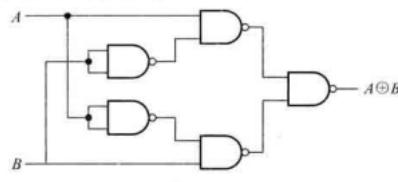
$$= \bar{A}B + A\bar{B} \quad (\text{与-或表达式})$$

$$= \overline{\overline{AB}} + \overline{A\bar{B}} \quad (\text{先利用 } \overline{\overline{A}} = A, \text{ 再用摩根定理})$$

$$= \overline{\overline{AB}} \cdot \overline{A\bar{B}} \quad (\text{与非-与非表达式})$$



(a)



(b)

图 2.3.2 例 2.3.6 的逻辑图

(a) 与-或表达式的实现 (b) 与非门的实现

图 2.3.2(a)是根据最简与-或表达式画出的逻辑图, 它用到与门、或门和非门三种类型的门; 而图 2.3.2(b)是根据与非-与非表达式画出的逻辑图, 它只用到两输入端与非门一种类型。注意, 图 2.3.2(b)中用两输入与非门来实现非门(即将两个输入端连接在一起)。

将与-或表达式变换为与非-与非表达式时, 首先对与-或表达式取两次非, 然后按照摩根定

理分开下面的非号。将与-或表达式转换成或非-或非表达式时,首先对与-或表达式中的每个乘积项单独取两次非,然后按照摩根定理分开下面的非号。下面再举一例说明逻辑函数的变换。

例 2.3.7 试对逻辑函数表达式 $L = \overline{\overline{A}}\overline{B}\overline{C} + \overline{A}\overline{B}\overline{C}$ 进行变换,仅用或非门画出该表达式的逻辑图。

$$\begin{aligned} \text{解: } L &= \overline{\overline{A}}\overline{B}\overline{C} + \overline{A}\overline{B}\overline{C} \\ &= A + B + \overline{C} + \overline{A} + B + C \quad (\text{摩根定理}) \\ &= \overline{\overline{A + B + \overline{C}}} \\ &= \overline{A + B + \overline{C}} + \overline{A} + B + C \quad (\text{或非-或非表达式}) \end{aligned}$$

逻辑图如图 2.3.3 所示。

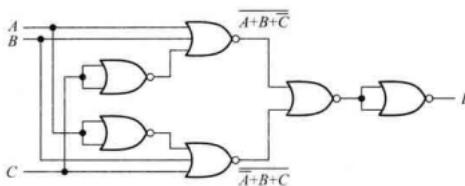


图 2.3.3 例 2.3.7 的逻辑图

复习思考题

2.3.1 对逻辑函数进行化简的目的是什么? 最简与-或表达式的标准是什么?

2.3.2 对逻辑函数进行变换的目的是什么?

2.3.3 用代数法将下列各式化简成最简的与-或表达式:

- | | |
|---|---|
| (1) $AB(\overline{BC} + A)$ | (2) $(A + B)(A\overline{B})$ |
| (3) $\overline{ABC}(\overline{B} + \overline{C})$ | (4) $\overline{A\overline{B} + ABC} + A(B + A\overline{B})$ |

2.4 逻辑函数的卡诺图化简法

利用代数法可使逻辑函数变成较简单的形式,但经代数法化简得到的逻辑表达式是否为最简式较难判断。本节介绍的卡诺图^①法可以比较简便地得到最简的逻辑表达式。

① 美国工程师 Karnaugh 在 20 世纪 50 年代提出。

2.4.1 用卡诺图表示逻辑函数

1. 卡诺图的引出

一个逻辑函数的卡诺图就是将此函数的最小项表达式中的各最小项相应地填入一个特定的方格图内,此方格图称为卡诺图。因此,卡诺图是逻辑函数的一种图形表示。

下面从一变量卡诺图开始讨论,逐步过渡到多变量卡诺图。

对于 n 个变量的逻辑函数有 2^n 个最小项,因此一变量的逻辑函数有2个最小项。设变量为 D ,则其两个最小项是 \bar{D} 和 D ,分别记为 $m_0 = \bar{D}, m_1 = D$ 。用两个相邻的方格表示这两个最小项,如图2.4.1(a)所示。方格上的 \bar{D} 和 D 分别表示非变量和原变量。为简明起见,非变量 \bar{D} 可以不标出,只标出原变量 D ,得到图2.4.1(b)。图2.4.1(c)是另一种更简单的画法,图中 m_0, m_1 只用其下标编号来表示。

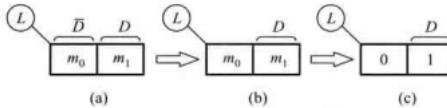


图 2.4.1 一变量卡诺图

二变量逻辑函数的最小项有 $2^2 = 4$ 项,如果逻辑函数的变量用 C, D 表示,则 $m_0 = \bar{C} \cdot \bar{D}, m_1 = \bar{C} \cdot D, m_2 = C \cdot \bar{D}, m_3 = C \cdot D$ 。用四个相邻的方格来表示这四个最小项。

将一变量卡诺图按照图2.4.2(a)所示箭头方向展开,得到二变量卡诺图如图2.4.2(b)所示。新增加的两个方格标以变量 C ,新方格最小项的编号相对原方格编号增加了 $2^{n-1} = 2^{2-1} = 2$,即方格0后面为方格2,方格1后面为方格3。按照展开规律,中间两格 m_1 和 m_3 包含原变量 D ,右边两格 m_1 和 m_2 包含原变量 C ,图中清楚地展示了原变量所包含的方格。

综上所述,可归纳“折叠展开”的法则如下:

- ① 新增加的方格按展开方向应标以新变量;
- ② 新方格内最小项编号应为展开前对应方格编号加上 2^{n-1} 。

按照同样的方法,从二变量卡诺图展开获得三变量卡诺图,如图2.4.3所示。三变量逻辑函数 $L(B, C, D)$ 有8个最小项,可用8个相邻的方格来表示。新增加的四个方格标以变量 B ,并且其最小项编号相对原方格的编号加上 $2^{n-1} = 2^{3-1} = 4$ 。方格所处的位置,与相应的最小项对应,例如,2号方格处于变量为 \bar{B}, C, \bar{D} 的区域,则 $m_2 = \bar{B}C\bar{D}$,余类推。

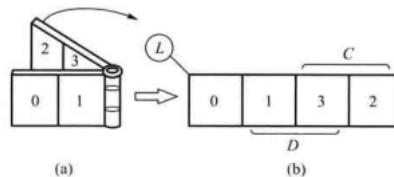


图 2.4.2 二变量卡诺图

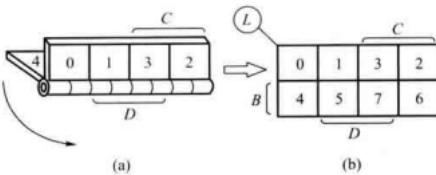


图 2.4.3 三变量卡诺图

同理,可得四变量卡诺图,如图 2.4.4 所示。在使用时,根据方格所在区域对应的变量 A 、 B 、 C 、 D 的取值,可直接填入相应的最小项。

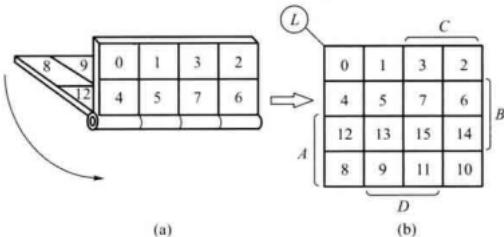


图 2.4.4 四变量卡诺图

2. 卡诺图的特点

卡诺图的特点是:几何位置相邻的最小项在逻辑上也是相邻的。即相邻的两个最小项只有一个变量不同,这是用卡诺图化简逻辑函数的主要依据。

现以四变量卡诺图为例,并将最小项用变量表示出来,如图 2.4.5 所示。图中相邻的方格内只有一个变量不同,例如, $m_4 = \bar{A}B\bar{C}\bar{D}$, $m_5 = \bar{A}B\bar{C}D$,这两个相邻最小项之间的差别仅在于 D 和 \bar{D} , m_5 和 m_{13} 之间的差别在于 A 和 \bar{A} 。要特别指出的是,卡诺图同一行最左端和最右端的方格是相邻的,同一列最上端和最下端两个方格也是相邻的。这个特点说明在几何位置上卡诺图具有上下、左右封闭的特性。

3. 卡诺图的简化表示法

图 2.4.6 所示为图 2.4.5 所示的卡诺图简化形式。变量 A 、 B 、 C 、 D 的每组取值,与对应方格内的最小项编号一一对应。例如, $AB=11, CD=01$,即 1101 对应最小项 m_{13} 。

4. 已知逻辑函数,画出卡诺图

当逻辑函数为最小项表达式时,在卡诺图对应最小项的方格填上 1 ,其余的方格填上 0 (也可用空格表示),就可以得到相应的卡诺图。也就是说,任何逻辑函数都等于其卡诺图中为 1 的方格所对应的最小项之和。

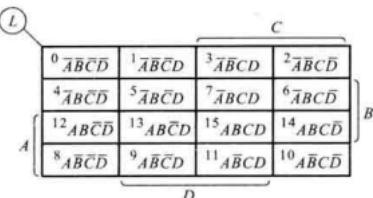


图 2.4.5 填入最小项的卡诺图

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

图 2.4.6 图 2.4.5 的简化表示法

例如,要画出逻辑函数 $L(A,B,C,D) = \sum m(0,1,2,3,4,8,10,11,14,15)$ 的卡诺图时,首先画出四变量卡诺图,并将逻辑函数表达式中最小项对应的方格填入 1,其余填入 0,即可得图 2.4.7 所示的 $L(A,B,C,D)$ 的卡诺图。

当逻辑函数的表达式为其他形式时,可将其变换为最小项表达式后,再作出卡诺图。

例 2.4.1 画出

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

解: (1) 由摩根定理,将上式变换为

$$\begin{aligned}\bar{L} &= ABCD + AB\bar{C}\bar{D} + A\bar{B}C\bar{D} + \bar{A}BC\bar{D} + \bar{A}\bar{B}\bar{C}\bar{D} \\ &= \sum m(15,13,10,6,0)\end{aligned}$$

(2) 因上式为 \bar{L} ,故最小项对应的方格内填入 0,其余填入 1,得到图 2.4.8 所示的卡诺图。

L	CD	00	01	11	10
AB	00	1	1	1	1
01	1	0	0	0	
11	0	0	1	1	
10	1	0	1	1	

图 2.4.7 $L(A,B,C,D)$ 的卡诺图

L	CD	00	01	11	10
AB	00	0	1	1	1
01	1	1	1	0	
11	1	0	0	1	
10	1	1	1	0	

图 2.4.8 例 2.4.1 的卡诺图

2.4.2 用卡诺图化简逻辑函数

1. 化简的依据

卡诺图具有相邻性,若两个相邻的方格均为 1,则这两个最小项之和有一个变量可以被消去,例如,图 2.4.5 所示卡诺图中的方格 5 和方格 7,其最小项之和为 $\bar{A}B\bar{C}D + \bar{A}BC\bar{D} = \bar{A}BD(\bar{C} + C) = \bar{A}BD$,将两个方格中的不相同的因子 C 和 \bar{C} 消去。

若四个相邻的方格为 1,则这四个最小项之和有两个变量可以被消去,如图 2.4.5 所示四变

量卡诺图中的方格 2, 3, 7, 6, 它们的最小项之和

$$\begin{aligned} A\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}CD + \bar{ABC}\bar{D} &= \bar{A}\bar{B}C(D+\bar{D}) + \bar{ABC}(D+\bar{D}) \\ &= \bar{A}\bar{B}C + \bar{ABC} = \bar{AC} \end{aligned}$$

消去了四个方格中不相同的两个因子, 使逻辑表达式得到简化。这就是卡诺图法化简逻辑函数的基本原理。

2. 化简的步骤

用卡诺图化简逻辑函数的步骤如下:

- (1) 将逻辑函数写成最小项表达式。
- (2) 按最小项表达式填卡诺图, 凡式中包含了的最小项, 其对应方格填 **1**, 其余方格填 **0**。
- (3) 找出为 **1** 的相邻最小项, 用虚线(或者细实线)画一个包围圈, 每个包围圈含 2^n 个方格, 写出每个包围圈的乘积项。

- (4) 将所有包围圈对应的乘积项相加。

有时也可以由真值表直接填卡诺图, 以上的(1)、(2) 两步就合为一步。

包围圈的原则:

- (1) 包围圈内的方格数必定是 2^n 个, n 等于 0, 1, 2, 3, …。
- (2) 相邻方格包括上下底相邻, 左右边相邻和四个角两两相邻。
- (3) 同一方格可以被不同的包围圈重复包围, 但新增包围圈中一定要有新的方格, 否则该包围圈为多余。

- (4) 包围圈内的方格数要尽可能多, 包围圈的数目要尽可能少。

化简逻辑函数后, 一个包围圈对应一个乘积项, 包围圈越大, 所得乘积项中的变量越少。包围圈个数越少, 乘积项个数也越少, 得到的与-或表达式也最简。

例 2.4.2 用卡诺图法化简下列逻辑函数, 求 L 的最简与-或表达式。

$$L(A, B, C, D) = \sum m(0, 2, 5, 7, 8, 10, 13, 15)$$

解: (1) 由 L 画出卡诺图, 如图 2.4.9 所示。

- (2) 找出 **1** 的相邻最小项, 用虚线画包围圈, 合并最小项, 得最简与-或表达式为

$$L = BD + \bar{B}\bar{D}$$

例 2.4.3 用卡诺图法化简下列逻辑函数

$$L(A, B, C, D) = (\bar{A}\bar{B} + B\bar{D})\bar{C} + BD(\bar{A}\bar{C}) + \bar{D}(\bar{A} + \bar{B})$$

解: (1) 将逻辑函数化简, 得到与-或表达式。

$$\begin{aligned} L(A, B, C, D) &= \bar{A}\bar{B}\bar{C} + B\bar{C}\bar{D} + BD(A+C) + \bar{D}AB \\ &= \bar{A}\bar{B}\bar{C} + B\bar{C}\bar{D} + ABD + BCD + AB\bar{D} \end{aligned}$$

(2) 由逻辑表达式直接作卡诺图, 如图 2.4.10 所示。

(3) 画包围圈合并最小项, 得简化的与-或表达式为

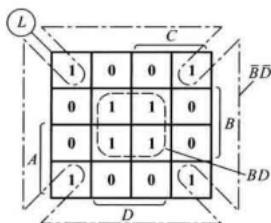


图 2.4.9 例 2.4.2 的卡诺图

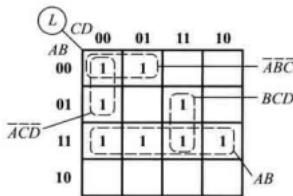


图 2.4.10 例 2.4.3 的卡诺图

$$L = AB + BCD + \bar{A} \bar{B} \bar{C} + \bar{A} \bar{C} \bar{D}$$

注意,此例中包含最小项 m_4 的包围圈还有一种画法,即将 m_4 和 m_{12} 画成一个包围圈,得到下面另一种简化的与-或表达式。可见,包围圈画的不同,得到的表达式也不同。

$$L = AB + BCD + \bar{A} \bar{B} \bar{C} + B \bar{C} \bar{D}$$

利用卡诺图化简时,如果卡诺图中大多数方格为 1,用包围 1 的方法需要画很多包围圈。这时采用包围 0 的方法化简更为简单。即求出非函数 \bar{L} ,再对 \bar{L} 求非,下面举例说明。

例 2.4.4 化简下列逻辑函数

$$L(A, B, C, D) = \sum m(0 \sim 3, 5 \sim 11, 13 \sim 15)$$

解:由 L 画出卡诺图,如图 2.4.11(a) 所示。

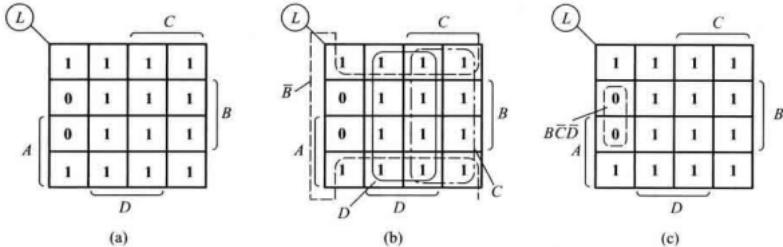


图 2.4.11 例 2.4.4 的卡诺图

方法一:用包围 1 的方法化简,如图 2.4.11(b) 所示,得

$$\bar{L} = \bar{B} + C + D$$

方法二:用包围 0 的方法化简,如图 2.4.11(c) 所示,得

$$\bar{L} = B \bar{C} \bar{D}$$

对 \bar{L} 求非

$$L = \bar{B} + C + D$$

可见,两种方法结果相同。

3. 具有无关项的化简

在实际工作中,当逻辑变量被赋予特定含义时,有一些变量的取值组合根本就不会出现,或者对应于变量的某些取值,其函数值可以是任意的(它的值可以取 0 或取 1),将变量取这些值所对应的最小项称为无关项或任意项。

无关项的值可以取 0 或取 1,具体取什么值,可以根据使函数尽量得到简化而定。下面举一个利用无关项化简的例子。

例 2.4.5 要求设计一个逻辑电路,能够判断 1 位十进制数是奇数还是偶数,当十进制数为奇数时,电路输出为 1;当十进制数为偶数时,电路输出为 0。

解:第一步,列写真值表。用 8421 BCD 码表示十进制数,输入变量用 A, B, C, D 表示,当对应的十进制数为奇数时,函数值为 1,反之为 0,得到表 2.4.1 所示的真值表。

表 2.4.1 例 2.4.5 的真值表

对应十进制数	输入变量				输出 L
	A	B	C	D	
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	1
无关项	1	0	1	0	x
	1	0	1	1	x
	1	1	0	0	x
	1	1	0	1	x
	1	1	1	0	x
	1	1	1	1	x

注意,8421 BCD 码只有十个数,表中 4 位二进制码的后六种组合是无关的,1 位十进制数不包括 10~15,这些无关状态根本不会出现,输入变量的这后六种组合所对应的最小项就是无关项,它们对应的函数值可以任意假设,为 0 为 1 都可以,通常以 x(或 ϕ) 表示。

第二步,根据真值表的内容,填写四变量卡诺图,如图 2.4.12 所示。

第三步,画包围圈,此时应利用无关项,显然,将最小项 m_{13} 、 m_{15} 、 m_{11} 对应的方格视为 1,可以得到最大的包围圈,由此得到逻辑表达式

$$L=D$$

若不利用无关项, $L=\overline{A}D+\overline{B}\overline{C}D$,结果复杂得多。

以上讨论的是从一变量到四变量的卡诺图,并以 4 变量卡诺图为例,分析了逻辑函数的化简方法。至于更多变量,如五变量或六变量卡诺图的应用,可参阅文献[3,4]。

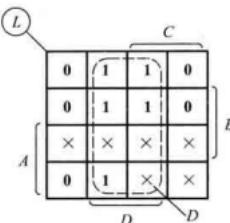


图 2.4.12 例 2.4.5 的卡诺图

复习思考题

2.4.1 什么是无关项?

2.4.2 使用卡诺图化简逻辑函数的依据是什么?

2.4.3 用卡诺图找出下列各个逻辑函数的最简与-或表达式。

$$(1) L(A,B,C,D) = \sum m(1,4,5,7,12,14,15)$$

$$(2) L(A,B,C) = \prod M(0,1,2,4)$$

2.5 硬件描述语言 Verilog HDL 基础

硬件描述语言 HDL 类似于计算机高级程序设计语言(如 C 语言等),它是一种以文本形式来描述数字系统硬件的结构和功能的语言,用它可以表示逻辑电路图、逻辑表达式,还可以表示更复杂的数字逻辑系统所完成的逻辑功能(即行为)。人们还可以用 HDL 编写设计说明文档,这种文档易于存储和修改,适用于不同的设计人员之间进行技术交流,还能被计算机识别和处理,计算机对 HDL 的处理包括两个方面:逻辑仿真和逻辑综合。

逻辑仿真是指用计算机仿真软件对数字逻辑电路的结构和行为进行预测,仿真器对 HDL 描述进行解释,以文本形式或时序波形图形式给出电路的输出。在电路被实现之前,设计人员根据仿真结果可以初步判断电路的逻辑功能是否正确。在仿真期间,如果发现设计中存在的错误,可以对 HDL 描述进行修改,直至满足设计要求为止。

逻辑综合是指从 HDL 描述的数字逻辑电路模型中导出电路基本元件列表以及元件之间的连接关系(常称为门级网表)的过程,即将 HDL 代码转换成真实的硬件电路。它类似于高级程序设计语言中对一个程序进行编译,得到目标代码的过程。所不同的是,逻辑综合不会产生目标代码,而是产生门级元件及其连接关系的数据库,根据这个数据库可以制作出集成电路或印制电路板(Printed Circuit Board,PCB)。

早期较为流行的硬件描述语言是 ABEL^①, 目前, 有两种硬件描述语言符合 IEEE 标准: VHDL 和 Verilog HDL(简称 Verilog)。VHDL 是在 20 世纪 80 年代中期由美国国防部支持开发出来的, 约在同一时期, 由 Gateway Design Automation^② 公司开发出 Verilog 语言。1990 年, Verilog 被公开推向市场, 并成为最流行的描述数字电路的语言。1995 年, Verilog 正式地被批准为 IEEE 的标准, 叫做 1364—1995。该语言的修订增强版引入了一些新的特性, 分别于 2001 年、2005 年被批准为 IEEE 的标准, 叫做 1364—2001 和 1364—2005。修订增强版支持原始 Verilog 版本的所有特性。

尽管这两种语言在很多方面都有所不同, 但在学习逻辑电路时, 设计者使用任何一种语言都可以完成自己的任务, 并且 Verilog 的句法根源出自通用的 C 语言, 较 VHDL 易学易用。作为入门, 本书简要讲解 Verilog 的基本知识, 并介绍逻辑电路综合和设计用到的 Verilog 结构, 对逻辑仿真时用到的语言结构没有进行介绍。

2.5.1 Verilog 语言的基本语法规则

Verilog 语言继承了 C 语言的许多语法结构, 同时又增加了一些新的规则, 本节将介绍 Verilog 语言的基本语法规则。

1. 间隔符

Verilog 的间隔符包括空格符(\b)、TAB 键(\t)、换行符(\n)及换页符。如果间隔符并非出现在字符串中, 则该间隔符被忽略。所以编写程序时, 可以跨多行书写, 也可以在一行内书写。

间隔符主要起分隔文本的作用, 在必要的地方插入适当的空格或换行符, 可以使文本错落有致, 便于阅读与修改。

2. 注释符

Verilog 支持两种形式的注释符: /* --- */ 和 //。其中, /* --- */ 为多行注释符, 用于写多行注释; // 为单行注释符, 以双斜线//开始到行尾结束为注释文字。注释只是为了改善程序的可读性, 在编译时不起作用。

3. 标识符和关键词

给对象(如模块名、电路的输入与输出端口、变量等)取名所用的字符串称为标识符, 标识符通常由英文字母、数字、\$ 符或者下画线组成, 并且规定标识符必须以英文字母或下画线开始, 不能以数字或 \$ 符开头。英文字母的大小写是有区别的。例如, clk、counter8、_net、bus_A 等都是合法的标识符, 2cp、\$latch、a * b 则是非法的标识符; A 和 a 是两个不同的标识符。

关键词为小写的英文字符串, 它是 Verilog 语言本身规定的特殊字符串, 在 Verilog 中有 100 个左右预定义的关键词用来定义该语言的结构。例如, module、endmodule、input、output、wire、reg、and 等都是关键词。关键词不能作为标识符使用。

^① 系 Advanced Boolean Equation Language 的缩写。

^② 该公司于 1989 年被 Cadence 公司收购。

本书为清晰起见,将关键词以粗体字印刷,但语言本身没有这一要求。

4. 逻辑值集合

为了表示数字逻辑电路的逻辑状态,Verilog 语言规定了 4 种基本的逻辑值,如表 2.5.1 所示。

表 2.5.1 4 种逻辑状态的表示

0	逻辑 0 、逻辑假
1	逻辑 1 、逻辑真
x 或 X	不确定的值(未知状态)
z 或 Z	高阻态

5. 常量及其表示

Verilog 中有两种类型的常量:整数型常量和实数型常量。

整数型常量有两种不同的表示方法:一是使用简单的十进制数的形式表示常量,例如:30、-2 都是十进制数表示的常量。用这种方法表示的常量被认为是有符号的常量。二是使用带基数的形式表示常量,其格式为:

<+/-><位宽><基数符号><数值>

其中<+/->表示常量是正整数还是负整数,当常量为正整数时,前面的正号可以省略;<位宽>定义了常量对应的二进制数的宽度;<基数符号>定义了后面<数值>的表示形式,在<数值>表示中,左边为最高有效位,右边为最低有效位。整数型常量可以用二进制数(基数符号为 b 或 B)的形式表示,还可以用十进制数(基数符号为 d 或 D)、十六进制数(基数符号为 h 或 H)和八进制数(基数符号为 o 或 O)的形式表示。例如:3'b101、5'o37、8'he3 分别表示位宽为 3 位的二进制数 101、位宽为 5 位的八进制数 37 和位宽为 8 位的十六进制数 e3,-4'd10、4'b1x0x 分别表示位宽为 4 位的十进制数 10 和位宽为 4 位的二进制数 1x0x。为了增加数值的可读性,可以在数字之间增加下画线,例如:8'b1001_0011 是位宽为 8 位的二进制数 10010011。

实数型常量也有两种表示方法:一是使用简单的十进制记数法,例如:0.1、2.0、5.67 等都是十进制记数法表示的实数型常量。二是使用科学记数法,23.5、1e2、3.6E2、5E-4 等都是使用科学记数法表示的实数型常量,它们以十进制记数法表示分别为 23.5、10.0、360.0 和 0.0005。

为了改善可读性和方便将来修改程序,Verilog 允许用参数定义语句定义一个标识符来代表一个常量,称为符号常量。定义的格式为:

parameter 参数名 1=常量表达式 1,参数名 2=常量表达式 2,⋯;

下面是符号常量的定义实例:

```
parameter BIT=1,BYTE=8,PI=3.14;
parameter DELAY=(BYTE+BIT)/2;
```

6. 字符串

字符串是双撇号内的字符序列,但字符串不允许分成多行书写。在表达式和赋值语句中,字

字符串要转换成无符号整数,用一串 8 位 ASCII 值表示,每一个 8 位 ASCII 码代表一个字符。例如:字符串“ab”等价于 16'h5758。存储字符串“INTERNAL ERROR”,需要定义 8×14 位的变量。

2.5.2 变量的数据类型

在 Verilog 语言中,变量有两大类数据类型:一类是线网类型^①,另一类是寄存器类型^②。

1. 线网类型

线网类型是硬件电路中元件之间实际连线的抽象。线网类型变量的值由驱动元件的值决定。例如,图 2.5.1 所示线网 L 跟与门 G1 的输出相连,线网 L 的值由与门的驱动信号 a 和 b 所决定,即 $L = a \& b$ 。a、b 的值发生变化,线网 L 的值会立即跟着变化。当线网型变量被定义后,没有被驱动元件驱动时,线网的默认值为高阻态 z(线网 trireg 除外,它的默认值为 x)。

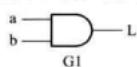


图 2.5.1 线网示意图

常用的线网类型由关键词 **wire** 定义。如果没有明确地说明线网型变量是多少位宽的矢量,则线网型变量的位宽为 1 位。在 Verilog 模块中如果没有明确地定义输入、输出变量的数据类型,则默认认为是位宽为 1 位宽的 **wire** 型变量。**wire** 型变量的定义格式如下:

```
wire [n-1:0] 变量名 1, 变量名 2, ..., 变量名 n;
```

其中,方括号内以冒号分隔的两个数字定义了变量的位宽,位宽的定义也可以用 [n:1] 的形式定义。下面是 **wire** 型变量定义的一些例子:

wire a,b;	// 声明两个线网类型变量 a,b
wire L;	// 声明一个线网类型变量 L
wire [7:0] databus;	// 声明一个 8 位宽的线网类型变量 databus
wire [32:1] busA, busB, busC;	// 声明 3 个位宽为 32 的线网类型变量 busA, busB, busC

线网类型除 **wire** 外,还有 **wand**、**wor**、**tri**、**triand**、**trior**、**trireg** 等。

2. 寄存器类型

寄存器类型表示一个抽象的数据存储单元,它具有状态保持作用。寄存器型变量只能在 **initial** 或 **always** 内部被赋值。寄存器型变量在没有被赋值前,它的默认值是 x。

Verilog 语言中,有 4 种寄存器类型的变量,如表 2.5.2 所示。

表 2.5.2 寄存器类型变量及其说明

寄存器类型	功能说明
reg	用于行为描述中对寄存器型变量的说明
integer	32 位带符号的整数型变量
real	64 位带符号的实数型变量,默认值为 0
time	64 位无符号的时间型变量

① “线网类型”是英文 net type 的译称。

② “寄存器类型”是英文 register type 的译称。

常用的寄存器类型用关键词 **reg** 进行声明。如果没有明确地说明寄存器型变量是多位宽的矢量，则寄存器变量的位宽为 1 位。**reg** 型变量的定义格式如下：

```
reg [n-1:0] 变量名 1, 变量名 2, … , 变量名 n;
```

下面是 **reg** 型变量定义的一些例子：

```
reg clock; // 声明一个寄存器变量 clock
```

```
reg [3:0] counter; // 声明一个 4 位宽的寄存器变量 counter
```

integer、**real** 和 **time** 等 3 种寄存器类型变量都是纯数学的抽象描述，不对应任何具体的硬件电路。**integer** 型变量通常用于对整型常量进行存储和运算，在算术运算中 **integer** 型数据被视为有符号的数，用二进制补码的形式存储。而 **reg** 型数据通常被当作无符号数来处理。注意 **integer** 型变量不能使用位矢量，例如 **integer [3:0] num;** 的定义是错误的。**integer** 型变量的应用举例如下：

```
integer counter; // 声明一个整型变量 counter
```

```
initial
```

```
counter = -1; // 将 -1 以补码的形式存储在 counter 中
```

其中，**initial** 是一种过程语句结构，只有寄存器类型的变量才能在 **initial** 内部被赋值。

real 型变量通常用于对实数型常量进行存储和运算，实数不能定义范围，其默认值为 0。当实数值被赋给一个 **integer** 型变量时，只保留整数部分的值，小数点后面的值被截掉。**real** 型变量的应用举例如下：

```
real delta; // 声明一个实数型变量 delta
```

```
initial
```

```
begin
```

```
delta = 4e10; // 给 delta 赋值
```

```
delta = 2.13;
```

```
end
```

```
integer i; // 声明一个整型变量 i
```

```
initial
```

```
i = delta; // i 得到的是 2 (只将实数 2.13 的整数部分赋给 i)
```

time 型变量主要用于存储仿真时间，它只存储无符号数。每个 **time** 型变量存储一个至少 64 位的时间值。为了得到当前的仿真时间，常调用系统函数 **\$time**。**time** 型变量的应用举例如下：

```
time current_time; // 声明一个时间类型的变量 current_time
```

```
initial
```

```
current_time = $time; // 保存当前的仿真时间到变量 current_time 中
```

2.5.3 运算符及其优先级

1. 运算符

Verilog 语言提供了 30 多个运算符，表 2.5.3 列出了对逻辑电路综合有用的部分运算符。按

照功能可以大致分为算术运算符、逻辑运算符、关系运算符、移位运算符等几大类。按照参与运算的操作数的个数，可以分为单目、双目和三目运算符。双目运算符是指一个运算符需要带两个操作数，即对两个操作数进行运算。单目运算符只对一个操作数进行运算，而三目运算符需要带三个操作数。

表 2.5.3 Verilog HDL 的运算符

类型	符号	功能说明	类型	符号	功能说明
算术运算符	+	二进制加	关系运算符 (双目运算符)	>	大于
	-	二进制减		<	小于
	-	2 的补码		\geq	大于或等于
	*	二进制乘		\leq	小于或等于
	/	二进制除		$= =$	相等
位运算符 (双目运算符)	\sim	按位取反	缩位运算符 (单目运算符)	!	缩位与
	&	按位与		$\sim \&$	缩位与非
		按位或			缩位或
	*	按位异或		$\sim $	缩位或非
	$\sim -$ 或 $- \sim$	按位同或		*	缩位异或
逻辑运算符 (双目运算符)	!	逻辑非	移位运算符 (双目运算符)	$>>$	右移
	$\&\&$	逻辑与		$<<$	左移
	$\ $	逻辑或			
位拼接运算符	[,] [][]	将多个操作数拼接成为一个操作数	条件运算符 (三目运算符)	?:	根据条件表达式是否成立，选择表达式

Verilog 语言中运算符有很多与 C 语言中的类似，下面仅介绍含义有差别的运算符。

缩位运算符是 Verilog 语言中新增加的，它和位运算符是有区别的。位运算是将两个操作数按对应位进行相应的逻辑运算，操作数是几位数，则运算结果也是几位数。而缩位运算是对单个操作数的各位进行相应运算，最后的运算结果是 1 位二进制数。假设变量 A、B 的值分别为 4'b1010 和 4'b1111，表 2.5.4 为这两种不同运算的例子。

表 2.5.4 位运算、缩位运算和逻辑运算的比较表

位运算	$\sim A = 0101$ $\sim B = 0000$	$A \& B = 1010$	$A B = 1111$	$A ^ B = 0101$	$A \sim ^ B = 1010$
缩位运算	$\& A = 1 \& 0 \& 1 \& 0$ $= 0$	$\sim \& A = 1$ $\& B = 1$	$ A = 1$ $\sim B = 0$	$\sim A = 0$ $\sim B = 0$	$\sim \sim A = 1$ $\sim \sim B = 1$

逻辑运算符的运算结果为 1 位, 其中 `&&` 和 `||` 为双目运算符, 逻辑运算符 `!` 为单目运算符。在运算过程中, 如果操作数只有 1 位, 那么 1 代表逻辑 1, 0 代表逻辑 0; 如果操作数由多位组成, 若操作数的每一位都是 0, 则认为该操作数具有逻辑 0 值; 反之, 若操作数中某一位为 1, 则认为该操作数具有逻辑 1 值; 如果任一个操作数为 `x` 或 `z`, 则逻辑运算的结果为不定态 `x`。例如, 设变量 A、B 的值分别为 $2'b10$ 和 $2'b00$, 则 A 为逻辑 1, B 为逻辑 0, 于是 `!A=0, !B=1, A && B=1 && 0=0, A || B=1 || 0=1`。

位拼接运算符的作用是将两个或多个信号的某些位拼接起来成为一个新的操作数, 进行运算操作。例如, 设 $A=1'b1, B=2'b10, C=2'b00$, 若将操作数 B、C 拼接起来, 则得到 $|B, C|=4'b1000$; 若将操作数 A、B 的第 1 位和 C 的第 0 位拼接起来, 则得到一个 3 位矢量的新操作数, 即 $|A, B[1], C[0]|=3'b110$ 。若将操作数 A、B、C 和 $3'b101$ 拼接起来, 则得到一个 8 位矢量的新操作数, 即 $|A, B, C, 3'b101|=8'b11001010$ 。

对同一个操作数重复拼接还可以使用双重大括号构成的运算符 `|||`, 例如 $|4|A||=4'b1111, |2|A|, 2|B|, C|=8'b11101000$ 。注意, 参与拼接的操作数必须标明位宽。

2. 运算符的优先级

表 2.5.5 是 Verilog 运算符的优先级。优先级的顺序从下向上依次增加, 位于顶部行的运算符优先级别最高, 在最底部行的优先级别最低。列在同一行的运算符优先级别相同。所有运算符(条件运算符 `? :` 除外)在表达式中都是从左向右结合的, 使用圆括号 `()` 可以改变运算的先后顺序。为了避免发生误解, 推荐使用圆括号去隔开不同的表达式, 以便代码的含义更明确易懂。

表 2.5.5 Verilog HDL 运算符的优先级

类型	符号	优先级别
取反	<code>!</code>	最高优先级
	<code>~</code> (求 2 的补码)	
算术	<code>*</code> <code>/</code> <code>+</code> <code>-</code>	
移位	<code>^</code> <code>>></code> <code><<</code>	
关系	<code><</code> <code><=</code> <code>></code> <code>>=</code>	
等于	<code>==</code> <code>!=</code>	
缩位	<code>&</code> <code>~&</code> <code>^</code> <code>~^</code> <code> </code> <code>~ </code>	
逻辑	<code>&&</code> <code> </code>	
条件	<code>?:</code>	最低优先级

2.5.4 Verilog 内部的基本门级元件

Verilog 语言包含 12 个预先定义的基本门级元件^①(gate level primitives)模型,如表 2.5.6 所示。门级元件的输出、输入必须为线网类型的变量。当使用这些元件进行逻辑仿真时,仿真软件会根据程序的描述给每个元件中的变量分配逻辑 0、逻辑 1、不确定态 x 和高阻态 z 这 4 个值之一。下面介绍这些元件的用法。

表 2.5.6 Verilog 语言内置的 12 个基本门级元件

元件符号	功能说明	元件符号	功能说明
and	多输入端的与门	nand	多输入端的与非门
or	多输入端的或门	nor	多输入端的或非门
xor	多输入端的异或门	xnor	多输入端的异或非门
buf	多输出端的缓冲器	not	多输出端的反相器
bufif1	控制信号高电平有效的三态缓冲器	notif1	控制信号高电平有效的三态反相器
bufif0	控制信号低电平有效的三态缓冲器	notif0	控制信号低电平有效的三态反相器

1. 多输入门

and、**nand**、**or**、**nor**、**xor** 和 **xnor** 是具有多个输入的逻辑门,它们的共同特点是:只允许有一个输出,但可以有多个输入。图 2.5.2 是与门元件(**and**)模型的示意图,一般的调用形式为:

and A1(out,in1,in2,...,inN);

其中,调用名 A1 可以省略。注意,调用门级元件时,在圆括号中需要列出输入、输出变量,括号中左边的第一个变量必须是输出变量,即与原理图中的输出端口对应。**nand**、**or**、**nor**、**xor**、**xnor** 的调用形式与之类似,不再赘述。

表 2.5.7 ~ 表 2.5.9 是两输入端 **and**、**or** 和 **xor** 元件的逻辑真值表,表中第一行对应于门的一个输入端,表中的第一列对应于门的另一个输入端,行、列输入的运算结果列在表的中间部位。注意,多输入门的输出不可能为高阻态 z。

其余 3 个多输入门(**nand**、**nor**、**xnor**)的真值表分别与上述的类似。



图 2.5.2 多输入与门元件模型

^① 有的文献翻译成门级原语。

表 2.5.7 两输入 and 真值表

and		输入 1			
		0	1	x	z
输入 2	0	0	0	0	0
	1	0	1	x	x
	x	0	x	x	x
	z	0	x	x	x

表 2.5.8 两输入 or 真值表

or		输入 1			
		0	1	x	z
输入 2	0	0	1	x	x
	1	1	1	1	1
	x	x	1	x	x
	z	x	1	x	x

表 2.5.9 两输入 xor 真值表

xor		输入 1			
		0	1	x	z
输入 2	0	0	1	x	x
	1	1	0	x	x
	x	x	x	x	x
	z	x	x	x	x

2. 多输出门

buf、**not** 是具有多个输出端的逻辑门, 它们的共同特点是: 允许有多个输出, 但只有一个输入。一般的调用形式为:

buf B1(out1,out2,⋯,outN,in);

not N1(out1,out2,⋯,outN,in);

其中, 调用名 B1、N1 可以省略。图 2.5.3 所示为多个输出端反相器的示意图, 其逻辑真值表如表 2.5.10 所示。**buf** 逻辑真值表与 **not** 的类似, 但输出值与 **not** 的相反。

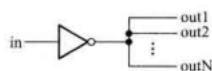


图 2.5.3 门级元件 not 的模型

表 2.5.10 not 真值表

not	输入			
	0	1	x	z
输出	1	0	x	x

3. 三态门^①

bufif1^②、**bufif0**、**notif1** 和 **notif0** 是三态门元件模型, 这些门有一个输出、一个数据输入和一

① 本书第 3 章将介绍三态门的有关内容。

② 该关键词可分两部分理解, buf 是 buffer 的缩写, 表示该元件完成缓冲器的功能; 后面的 if 表示完成该功能所需的条件, 即控制信号为逻辑 1。其他 3 个关键词的理解类同。

个输入控制。如果输入控制信号无效，则三态门的输出为高阻态 z。一般的调用形式为：

bufif1 B1(out,in,ctrl); **bufif0** B0(out,in,ctrl);

notif1 N1(out,in,ctrl); **notif0** N0(out,in,ctrl);

其中，调用名 B1、B0、N1、N0 可以省略。图 2.5.4 所示是 **bufif1** 和 **notif1** 的示意图。



图 2.5.4 三态门元件模型

(a) bufif1 (b) notif1

表 2.5.11 和表 2.5.12 给出了 **bufif1** 和 **notif1** 的逻辑真值表，表中，0/z 表明三态门的输出可能是 0，也可能是高阻态 z，主要由输入数据信号和控制信号的强度决定。有关信号强度的讨论超出了本书的范畴。

表 2.5.11 bufif1 真值表

bufif1		控制输入			
		0	1	x	z
数	0	z	0	0/z	0/z
据	1	z	1	1/z	1/z
输	x	z	x	x	x
入	z	z	x	x	x

表 2.5.12 notif1 真值表

notif1		控制输入			
		0	1	x	z
数	0	z	1	1/z	1/z
据	1	z	0	0/z	0/z
输	x	z	x	x	x
入	z	z	x	x	x

2.5.5 Verilog 程序的基本结构

模块 (module) 是 Verilog 描述电路的基本单元，它可以表示一个简单的门电路，也可以表示功能复杂的数字电路。对数字电路建模时，通常使用 Verilog 的一个或多个模块，不同的模块之间通过端口进行连接。定义模块的基本语法结构如下：

```
module 模块名(端口名 1, 端口名 2, 端口名 3, …);
    端口模式说明 (input, output, inout);
    {参数定义(可选);
        数据类型定义 (wire, reg 等);}
    } 说明部分
```

```
实例化低层次模块或基本门级元件;
连续赋值语句 (assign);
过程块结构 (initial 和 always);
行为描述语句;
} 逻辑功能描述部分,
其排列顺序是任意的。
endmodule
```

定义模块时总是以关键词 **module** 开始，并以关键词 **endmodule** 来结束。“模块名”是模块唯一的标识符，圆括号中以逗号分隔列出的端口名是该模块的输入、输出端口；在 Verilog 中，“端口类型说明”为 **input**(输入)、**output**(输出)、**inout**(双向端口)三者之一，它决定了信号流经端口的方向，凡是在模块名后面圆括号中出现的端口名，都必须明确地说明其端口模式。“参数定义”是将数值常量用符号常量代替，以增加程序的可读性和可修改性，它是一个可选择的语句。“数据类型定义”部分用来指定模块内所用的数据对象为寄存器类型还是连线类型。

接着要对该模块完成的逻辑功能进行描述，通常可以使用三种不同风格描述电路的功能：一是使用实例化低层次模块的方法，即调用其他已定义过的低层次模块对整个电路的功能进行描述，或者直接调用 Verilog 内部预先定义的基本门级元件描述电路的结构，通常将这种方法称为结构描述方式（对仅使用基本门级元件描述电路功能的方式，也称为门级描述方式）；二是使用连续赋值语句(**assign**)对电路的逻辑功能进行描述，通常称之为数据流描述方式，该方式特别便于对组合逻辑电路建模；三是使用过程块语句结构（包括 **initial** 语句结构和 **always** 语句结构两种）和比较抽象的高级程序语句对电路的逻辑功能进行描述，通常称之为行为描述方式。

行为描述侧重于描述模块的逻辑功能，不涉及实现该模块的详细硬件电路结构。设计人员可以选用这三种方式中的任意一种或混合使用几种对电路的功能进行描述，并且这几种描述方式在程序中排列的先后顺序是任意的。这些描述方式将在 4.6 节、5.6 节和 6.7 节做进一步介绍。除此之外，在 3.9 节中将介绍开关级描述方式，有时用这种方法对 MOS 管构成的逻辑电路进行建模，以便仿真 MOS 管电路的逻辑功能，但逻辑综合工具不支持这种建模方式。

例 2.5.1 用 Verilog 语言对图 2.5.5 所示简单逻辑电路进行建模。

解：该电路的逻辑表达式为

$$Y = D_0 \cdot \bar{S} + D_1 \cdot S$$

当 $S=0$ 时，输出 $Y=D_0$ ，当 $S=1$ 时， $Y=D_1$ 。可见，该电路能够根据输入 S 的控制，选择两路输入信号(D_0, D_1)中的一路作为输出。因此，该电路也被称为 2 选 1 数据选择器。

用 Verilog 语言对该电路建模时，有以下三种不同的描述方式。

(1) 门级描述方式

Verilog 语言内部预先定义了 12 个基本门级元件，门级描述就是调用这些“元件”描述逻辑图中的元件以及元件之间的连线关系。2 选 1 数据选择器的门级描述代码如下：

```
//Verilog model of circuit of Figure2.5.5. IEEE 1364—1995 Syntax
module mux2tol( D0,D1,S,Y);
    input D0,D1,S; //输入端口声明
    output Y; //输出端口声明
    wire Snot,A,B; //内部节点声明
```

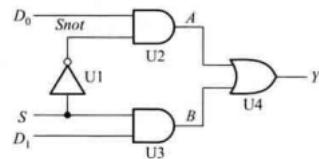


图 2.5.5 简单的门电路

```
//下面对电路的逻辑功能进行描述
not U1(Snot,S); //调用名 U1 可以省略
and U2( A,D0,Snot);
and U3( B,D1,S);
or U4( Y,A,B);
endmodule
```

说明：代码第1行以双斜线（//）开始到本行结尾之间的文本是一个注释，对这个电路进行简单的说明。第2行以关键词 **module** 开始声明了一个模块，**module** 后面跟有模块名（mux2to1）和端口名（D0、D1、S、Y）列表。端口名列表给出了该模块的输入、输出端口，端口用圆括号括起来，多个端口之间以逗号进行分隔。每一条语句以分号结尾。

接着，以关键词 **input** 和 **output** 定义了该模块的输入端口、输出端口。端口的数据类型默认为 **wire** 类型，此处将电路内部的结点信号（Snot、A、B）定义为 **wire** 类型。

电路的结构（即逻辑功能）由基本门级元件（**not**、**and**、**or**）进行描述，每个门级元件后面包含一个调用名（U1、U2 等）和由圆括号括起来、以逗号分隔的输出端口以及输入端口，调用基本门级元件时，Verilog 规定输出端口总是位于圆括号中左边的第 1 个位置，输入端口跟在后面。例如，调用名为 U4 的或门输出端口是 Y、输入端口是 A 和 B。调用名可以直接使用，不需要预先定义，并且调用基本门级元件时，调用名可以省略。最后模块以 **endmodule** 结尾（注意后面没有分号）。由于这个模块描述了电路的逻辑功能，故将该模块称为设计块。

在修订后的 Verilog 标准中，定义模块端口时可以按照如下方式书写：

```
module mux2to1( input D0,D1,S, output Y);
```

在模块的端口列表中，直接声明端口模式，不再需要独立的 **input** 和 **output** 语句，这使得代码变得更加紧凑。

（2）数据流描述方式

数据流描述方式使用的基本语句是连续赋值语句，它由关键词 **assign** 开始，后面跟着由操作数和运算符组成的逻辑表达式。连续赋值语句右边的表达式等同于用该函数实现的逻辑门。

2 选 1 数据选择器的数据流描述代码如下：

```
module mux2to1_dataflow( D0,D1,S,Y); //IEEE 1364—1995 Syntax
  input D0,D1,S; //输入端口声明
  output Y; //输出端口声明
  wire Y; //变量的数据类型声明
  //电路功能描述
  assign Y=( ~S & D0 ) |( S & D1 ); //注意表达式左边变量的数据类型必须是 wire 型
endmodule
```

可见,使用逻辑表达式编写 Verilog 代码更加简明、易懂。通过逻辑综合软件,能够自动地将数据流描述转换成为门级电路。

(3) 行为描述方式

对于功能复杂的逻辑电路,使用门级描述或者数据流描述电路的功能,其工作效率就会很低。另一种方法就是使用更抽象的结构来描述电路的逻辑功能和算法。使用 Verilog 提供的 **always** 语句结构和一些类似于高级语言的语句就可以对电路的行为进行描述。2 选 1 数据选择器的行为描述代码如下:

```
module mux2to1_bh(D0,D1,S,Y); //IEEE 1364—1995 Syntax
    input D0,D1,S; //输入端口声明
    output Y; //输出端口声明
    reg Y; //变量的数据类型声明
    //电路功能描述
    always @ (S or D0 or D1)
        if (S == 1) Y = D1; //也可以写成 if (S) Y = D1;
        else Y = D0; //注意表达式左边变量的数据类型必须是 reg 型
    endmodule
```

上述代码使用 **if-else** 语句对数据选择器的行为特征进行了描述,它没有涉及实现选择器的具体电路结构。**if-else** 语句是 Verilog 语言中的一种过程性语句,在本书 4.6 节将介绍其他类型的过程性语句。

行为描述的标识是 **always** 语句, **always** 是一个循环执行语句,在它后面跟着执行循环体语句的条件@ (S or D0 or D1)(注意后面没有分号),它表示圆括号内的任何一个变量发生变化时,下面的过程赋值语句就会被执行一次,执行完最后一条语句后,执行挂起, **always** 语句再次等待变量发生变化,因此将圆括号内列出的变量称为**敏感变量**。对组合逻辑电路来说,所有的输入信号都是敏感变量,应该被写在圆括号内。注意,过程赋值语句只能给寄存器型变量赋值,因此,程序中将输出变量 Y 的数据类型定义成 **reg**。

典型的 Verilog 设计模块可以包含几个 **always** 语句块,每个 **always** 语句块表示电路模型的一个部分。**always** 语句块的一个重要特性是它内部所包含的语句是按代码排列顺序执行的。这与连续赋值语句 **assign** 是不同的,多条 **assign** 语句是并行执行的。

修订后的 Verilog 标准在敏感变量列表中,可以用逗号代替 **or**,也可以用一个 * 号来代替敏感变量列表中所有输入信号,所以上述代码的更简单写法如下所示:

```
module mux2to1_bh( //Verilog 2001,2005 module port syntax
    input D0,D1,S,
    output reg Y
```

```

):;
always @ ( D0,D1,S) //Verilog 2001,2005 syntax
//可以写成 always @ ( * ) ,或者写成 always @ *
    if ( S) Y=D1;
    else Y=D0;
endmodule

```

2.5.6 逻辑功能的仿真与测试

一旦逻辑电路的设计块完成后,接下来就要测试这个设计块描述的逻辑功能是否正确。为此必须在输入端口加入测试信号,以便从输出端口检测其结果是否正确,这一过程常称为搭建测试平台(Test Bench)。根据仿真软件的不同,搭建测试平台的方法也不同,本书使用 Quartus II 9.1 版软件^①进行仿真,以波形图的方式建立一个矢量波形文件(扩展名为.vwf)作为激励信号。

对例 2.5.1 进行仿真^②时,首先进入 Quartus II 软件,创建一个新的工程设计项目,并使用文本编辑器输入源程序,再对该设计项目进行编译,然后使用波形编辑器创建一个新的矢量波形文件,最后进行逻辑功能仿真,得到图 2.5.6 所示的波形。由图可知,在 0~50 ns 期间,由于 S=0,所以输出 Y 与输入 D0 相同;在 50~100 ns 期间,S=1,故输出 Y 与输入 D1 相同。分析表明该设计块描述的逻辑功能是正确的。

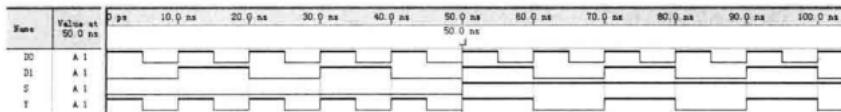


图 2.5.6 例 2.5.1 的仿真输出波形

对一个实际的门电路来说,信号从输入端传到输出端存在着延时,在使用 HDL 进行逻辑功能仿真时,有时需要说明门电路的延时。本书作为 Verilog 的入门,没有介绍这方面的内容。

小 结

- 逻辑代数是分析和设计逻辑电路的数学工具。一个逻辑问题可用逻辑函数来描述,逻辑函数可用真值表、逻辑表达式、卡诺图和逻辑图表达,这四种表示方法各具特点,可根据需要选用。
- 逻辑函数的与-或表达式和或-与表达式是两种常见的形式,但其表达式的形式不是唯一

^① Altera 公司自行研制的软件,为该公司生产的可编程逻辑器件提供了一个集成开发工具。

^② 仿真的详细步骤见附录 A。

的。任一个逻辑函数经过变换,都能得到唯一的最小项表达式或者是最大项表达式。

• 硬件描述语言 HDL 是一种以文本形式来描述数字系统硬件的结构和行为的语言,可以用来表示逻辑电路图、逻辑表达式,还可以表示更复杂的数字逻辑系统所完成的逻辑功能(即行为)。计算机对 HDL 的处理包括两个方面:逻辑仿真和逻辑综合。用 HDL 设计数字系统是当今的一种趋势。

习 题

2.1 逻辑代数的基本定律和规则

2.1.1 用真值表证明下列恒等式:

$$(1) (A \oplus B) \oplus C = A \oplus (B \oplus C)$$

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

2.1.2 用逻辑代数定律证明下列等式:

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

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

$$(3) A + A \overline{B} \overline{C} + \overline{ACD} + (\overline{C} + \overline{D}) E = A + CD + E$$

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

$$(5) AB + \overline{A}C + BCD = AB + \overline{A}C$$

$$(6) (A + B)(B + C)(\overline{A} + C) = (A + B)(\overline{A} + C)$$

2.1.3 应用反演规则和对偶规则,求下列函数的非函数和对偶函数。

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

$$(2) L = AB + \overline{C + D}$$

$$(3) L = \overline{A} \cdot \overline{B} + \overline{A} \cdot B + \overline{C} \cdot D$$

2.2 逻辑函数表达式的形式

2.2.1 将下列各式转换成与-或形式:

$$(1) \overline{A \oplus B} \oplus \overline{C \oplus D}$$

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

$$(3) \overline{\overline{AC} \cdot \overline{BD}} \overline{BC} \cdot \overline{AB}$$

2.2.2 列出逻辑函数 $L(A, B, C) = A \overline{B} + B \overline{C}$ 的真值表,并写出该函数的最小项表达式。

2.2.3 试写出下列各个函数的最小项表达式:

$$(1) L = A \overline{C} \overline{D} + \overline{B} C \overline{D} + ABCD$$

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

$$(3) L = \overline{\overline{AB} + ABD} (\overline{B} + \overline{CD})$$

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

2.2.4 判断下列等式是否成立?

$$(1) \overline{A} \overline{C} + BC + A \overline{B} = \overline{AB} + AC + \overline{B} \overline{C}$$

$$(2) \overline{AC} + AB \overline{C} + \overline{AB} + A \overline{B} = \overline{BC} + A \overline{C} + B \overline{C} + \overline{ABC}$$

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

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

2.2.5 列出逻辑函数 $L(A, B, C) = \overline{A} \cdot C + A \cdot \overline{B} \cdot \overline{C}$ 的真值表, 并写出该函数的最大项表达式。

2.2.6 已知下列函数的最大项表达式, 试写出其最小项表达式。

$$(1) L(A, B) = \prod M(0, 1, 2)$$

$$(2) L(A, B, C) = \prod M(0, 1, 3, 6, 7)$$

2.2.7 已知下列函数的最小项表达式, 试写出其最大项表达式。

$$(1) L(A, B) = \sum m(1, 2)$$

$$(2) L(A, B, C, D) = \sum m(1, 2, 5, 6)$$

2.3 逻辑函数的代数化简法

2.3.1 用代数法将下列各式化简成最简的与-或表达式:

$$(1) \overline{AB} + \overline{A} \overline{B} + \overline{AB} + A \overline{B}$$

$$(2) \overline{(\overline{A} + B)} + \overline{(A + B)} + \overline{(AB)} (\overline{A} \overline{B})$$

$$(3) \overline{B} + ABC + \overline{AC} + \overline{AB}$$

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

$$(5) ABC \overline{D} + ABD + BC \overline{D} + ABCD + B \overline{C}$$

$$(6) AC + \overline{ABC} + \overline{BC} + AB \overline{C}$$

2.3.2 已知逻辑函数表达式为 $L = \overline{ABC} \overline{D}$, 画出实现该式的逻辑电路图, 限使用非门和 2 输入与非门。

2.3.3 画出实现下列逻辑表达式的逻辑电路图, 限使用非门和 2 输入与非门。

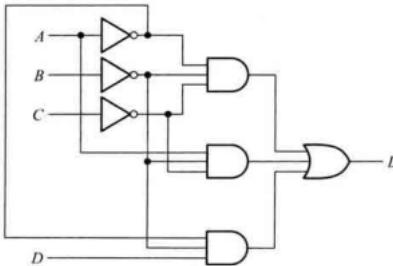
$$(1) L = AB + AC$$

$$(2) L = \overline{D(A+C)}$$

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

2.3.4 已知逻辑函数表达式为 $L = A \overline{B} + \overline{AC}$, 画出实现该式的逻辑电路图, 限使用非门和 2 输入或非门。

2.3.5 写出图题 2.3.5 所示电路的逻辑表达式, 将其转换成与非-与非表达式, 然后画出仅用与非门实现的逻辑图。



图题 2.3.5

2.4 逻辑函数的卡诺图化简法

2.4.1 已知逻辑函数 $L(A, B, C) = A\bar{B} + B\bar{C} + C\bar{A}$, 试用真值表、卡诺图和逻辑图(限用非门和与非门)表示。

2.4.2 已知函数 $L(A, B, C, D)$ 的卡诺图如图题 2.4.2 所示, 请写出函数 L 的最简与-或表达式。

2.4.3 用卡诺图法化简下列各式:

$$(1) \bar{A}\bar{B}CD + AB\bar{C}D + A\bar{B} + A\bar{D} + A\bar{B}C$$

$$(2) \bar{A}\bar{B}C + A\bar{B}\bar{C}D + AB\bar{C}D + ABC$$

$$(3) A\bar{B}CD + D(\bar{B}\bar{C}D) + (A+C)B\bar{D} + \bar{A}(\bar{B}+C)$$

$$(4) L(A, B, C, D) = \sum m(0, 2, 4, 8, 10, 12)$$

$$(5) L(A, B, C, D) = \sum m(0, 1, 2, 5, 6, 8, 9, 10, 13, 14)$$

$$(6) L(A, B, C, D) = \sum m(0, 2, 4, 6, 9, 13) + \sum d(1, 3, 5, 7, 11, 15)$$

$$(7) L(A, B, C, D) = \sum m(0, 4, 6, 13, 14, 15) + \sum d(1, 2, 3, 5, 7, 9, 10, 11)$$

2.4.4 用卡诺图化简法, 求下列函数的最简与-或表达式。

$$(1) L(A, B, C, D) = A\bar{C} + AD + \bar{B}\bar{C} + \bar{B}D$$

$$(2) L(A, B, C, D) = \sum m(3, 4, 5, 7, 13, 14, 15)$$

$$(3) L(A, B, C, D) = (\bar{A}+D)(B+\bar{D})(A+B)$$

$$(4) L(A, B, C, D) = \prod M(3, 4, 6, 7, 11, 12, 13, 14, 15)$$

2.5 硬件描述语言 Verilog HDL 基础

2.5.1 在 Verilog 语言中, 下列标识符是否正确?

(1) system1 (2) 2reg (3) FourBit_Adder (4) exec\$ (5) _2tolmux

2.5.2 Verilog 语言规定的 4 种基本逻辑值是什么?

2.5.3 在 Verilog 程序中, 如果没有说明输入、输出变量的数据类型, 试问它们的数据类型是什么?

2.5.4 填空题:

(1) 问下列运算的二进制值是多少?

`reg [3:0] m;`

`m = 4'b1010; //|2|m|的二进制值是 _____;`

(2) 假设 `m = 4'b0101`, 按要求填写下列运算的结果:

`&m = _____, !m = _____,`

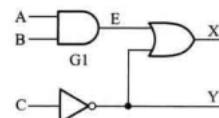
`~m = _____, ^~m = _____。`

2.5.5 下列 Verilog 程序描述了图题 2.5.5 所示的电路, 但程序中每一行有一个语法错误, 请改正。(注意: 基本门级元件的调用名可以省略。)

```
module Ex1(A,B,C,X,Y)
  input A,B,C
  output X,Y
  reg E;
  and G1(A,B,E);
  NOT (Y,C);
```

		L	CD	00	01	11	10
AB							
00				1			1
01		1					
10		1					
11				1			1

图题 2.4.2



图题 2.5.5

```
OR ( X,E,Y);
```

```
endmodule;
```

2.5.6 根据下面的 HDL 描述,用 Quartus II 软件对电路进行逻辑功能仿真,给出仿真波形。再画出该模块的逻辑电路图。

```
module circuit( A,B,L);
    input A,B;
    output L;
    wire a1,a2,Anot,Bnot;
    and G1( a1,A,B );
    and G2( a2,Anot,Bnot );
    not ( Anot,A );
    not ( Bnot,B );
    or ( L,a1,a2 );
endmodule
```

3

>>>

逻辑门电路

引言

在第1章我们介绍了与、或、非三种基本逻辑运算，并介绍了逻辑变量与逻辑函数的关系。对于逻辑门只给出了逻辑符号，没有涉及内部的结构和原理。

本章讨论的门电路是数字电路的基本逻辑单元。它们包括CMOS门电路、双极结型三极管(Bipolar Junction Transistor, BJT)构成的TTL门电路，以及其他门电路。门电路中的MOS管或BJT管工作在开关状态。对于CMOS或TTL门电路，首先介绍晶体管的开关特性，然后介绍由它们构成的基本逻辑门的电路结构和工作原理，着重阐述了其逻辑功能和外部输入特性、输出特性，以及其他电气特性，以便正确使用这些门电路。

3.1 逻辑门电路简介

3.1.1 各种逻辑门电路系列简介

实现基本逻辑运算和常用逻辑运算的单元电路称为门电路。逻辑门电路是组成各种数字电路的基本单元电路。将构成门电路的元器件制作在一块半导体芯片上，再封装起来，便构成了集成门电路。按照制造门电路晶体管的不同，分为MOS型、双极型和混合型。MOS型集成逻辑门有CMOS、NMOS、PMOS，双极型集成逻辑门主要有TTL和ECL，混合型集成逻辑门有BiCMOS。

CMOS逻辑门电路是目前使用最广泛、占主导地位的集成电路。早期的CMOS与TTL逻辑门相比，CMOS速度慢、功耗低，而TTL主要是速度快，但功耗大。后来随着制造工艺的不断改进，CMOS电路的集成度、工作速度、功耗和抗干扰能力远优于TTL。因此，几乎所有的CPU、存储器、PLD器件和专用集成电路(ASIC)都采用CMOS工艺制造，且费用较低。因此，出现种类繁多的CMOS逻辑系列。图3.1.1所示为CMOS发展过程中部分典型逻辑门系列，以及相对前一个系列在速度和功耗等方面改进。

早期生产的CMOS门电路为4000系列，其工作速度较慢，与TTL不兼容，但功耗低、工作电

压范围宽、抗干扰能力强。随后出现了高速 CMOS 器件 HC/HCT 系列。与 4000 系列相比，其工作速度快、带负载能力强。HCT 系列与 TTL 兼容，可与 TTL 器件互换使用。另一种 CMOS 系列是 AHC/AHCT 系列，其工作速度达到 HC/HCT 系列两倍之多。

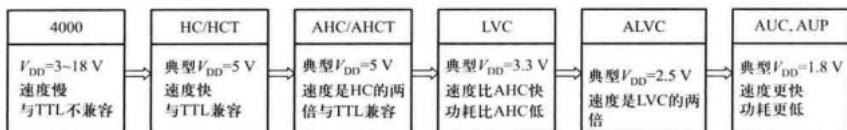


图 3.1.1 部分典型 CMOS 逻辑系列简介图

近年来，随着便携式设备（如笔记本电脑、数码相机、手机等）的发展，要求使用体积小、重量轻、功耗低的半导体器件，因此先后推出了低电压 CMOS 器件 LVC (Low-Voltage CMOS Logic) 系列，速度和性能比 LVC 更好的 ALVC (Advanced Low-Voltage CMOS Logic) 系列，超低电压 CMOS 器件 AUC (Advanced Ultra-Low-Voltage CMOS Logic) 系列，以及低功耗 CMOS 器件 AUP (Advanced Ultra-Low-Power CMOS Logic) 系列，并且半导体制造工艺的进步使它们的成本更低、速度更快。不同的 CMOS 系列器件对电源电压要求不一样，表 3.1.1 列出几种 CMOS 集成电路的电源电压范围和所允许的最大电源电压。

表 3.1.1 几种 CMOS 电路的电源电压值^①

参数 \ 类型	4000	74HC	74HCT	74LVC	74ALVC	74AUC
电源电压范围/V	3~18	2~6	4.5~5.5	1.65~3.6	1.65~3.6	0.8~2.7
电源最大电压额定值/V	20	7	7	6.5	4.5	3.6

中小规模集成电路芯片的名称以 54 或 74 开始，后加不同系列缩写字母及数字表示，如 54/74HC00。中间字母表示不同系列，如 HC 系列。最后的数字表示不同逻辑功能芯片的编号，如 00 表示 4 个 2 输入与非门，即一个芯片中封装了 4 个与非门，如图 3.1.2 所示。图 3.1.2 (a) 所示为双列直插 (Dual-Inline Package, DIP) 封装的芯片，图 3.1.2 (b) 所示为 74HC00 引脚排列图。54 和 74 系列的区别是 54 系列适用的温度范围更宽，测试和筛选标准更严格。

CMOS 是数字逻辑电路的主流工艺技术，但 CMOS 技术却不适合用在射频和模拟电路中。因此 BiMOS 成为射频系统中用得最多的工艺技术。BiCMOS 集成电路是将 BJT 的高速性和高驱动能力，以及 CMOS 的高密度、低功耗和低成本等优点结合起来，并且既可用于数字集成电路，也可用于模拟集成电路。BiMOS 技术主要用于高性能集成电路的生产。BiCMOS 门电路有 BCT

^① 本书引用的典型集成电路资料均根据 Texas Instruments 公司网站 www.ti.com 和 Philips Semiconductors 公司网站 (www.semiconductors.philips.com) 上所提供的 Data Sheet，谨此致谢。

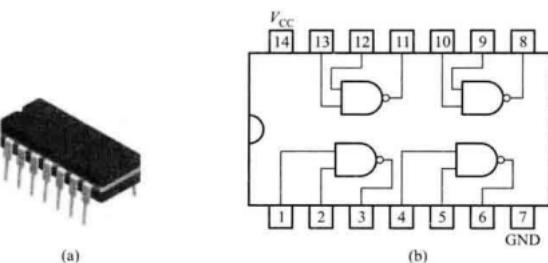


图 3.1.2 2 输入与非门 74HC00

(a) 双列直插封装 (b) 引脚排列图

(BiCMOS Bus-Interface Technology) 和 ABT(Advanced BiCMOS Technology) 系列, 低电压供电的 LVT(Low-Voltage BiCMOS Technology) 系列和速度性能更好的 ALVT(Advanced Low-Voltage BiCMOS Technology) 系列等。

TTL 是应用最早, 技术比较成熟的集成电路, 曾被广泛使用。大规模集成电路的发展, 要求每个逻辑单元电路的结构简单, 并且功耗低。TTL 电路不满足这个条件, 因此逐渐被 CMOS 电路所取代, 退出其主导地位。由于 TTL 技术在整个数字集成电路设计领域中的历史地位和影响, 目前主要应用于教育或是简单的中小规模数字电路。

最早的 TTL 门电路是 74 系列。后来为改善工作速度和功耗, 使用肖特基三极管, 生产出 74S 系列。之后推出 74LS 系列, 其速度与 74 系列相当, 但功耗却降低到 74 系列的 $1/5$ 。74LS 系列曾广泛应用于中、小规模集成电路。随着集成电路的发展, 生产出进一步改进的 74AS 和 74ALS 系列。74AS 系列与 74S 系列相比, 功耗相当, 但速度却提高了两倍。74ALS 系列将 74LS 系列的速度和功耗又进一步改善。而 74F 系列的速度和功耗介于 74AS 和 74ALS 之间, 应用于速度要求较高的 TTL 逻辑门电路。

射极耦合逻辑门(Emitter-Coupled Logic, ECL)也是一种双极型数字集成电路, 其基本器件是差分对管。在饱和型的 TTL 电路中, 三极管作为开关在饱和区和截止区切换, 其退出饱和区需要的时间较长。而 ECL 电路中三极管导通时未进入饱和状态, 因此工作速度极高。但 ECL 器件功耗比较高, 不适合制成大规模集成电路, 因此不像 CMOS 或 TTL 系列被广泛使用。ECL 电路主要用于高速或超高速数字系统或设备中。

3.1.2 开关电路

在二值数字逻辑中, 逻辑变量的取值不是 0 就是 1。在数字电路中, 与其对应的是电子器件的“闭合”和“断开”两种状态。图 3.1.3 所示为开关电路示意图。当开关 S 断开时, 输出电压 $v_o = V_{cc}$, 输出逻辑 1, 如图 3.1.3(a) 所示。反之, 当开关 S 接通时, 输出电压 $v_o = 0$, 输出逻辑 0, 如图 3.1.3(b) 所示。

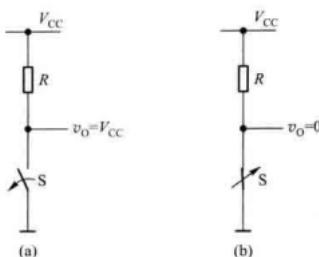


图 3.1.3 开关电路示意图

(a) 输出逻辑 1 (b) 输出逻辑 0

早期的开关由继电器构成，后来使用 BJT 或 MOS 管作为开关。BJT 或 MOS 管相当于一个受控开关，当其工作在截止状态时，相当于开关断开，输出高电平；当其工作在饱和状态时，相当于开关闭合，输出低电平。

复习思考题

- 3.1.1 CMOS 逻辑门电路有哪些优点？
- 3.1.2 74HC 与 74HCT 系列的区别是什么？
- 3.1.3 现在的 CPU、PLD 和 ASIC 芯片主要采用什么工艺制造？
- 3.1.4 BiCMOS 电路的特点是什么？

3.2 基本 CMOS 逻辑门电路

3.2.1 MOS 管及其开关特性

CMOS 逻辑门电路是以 MOS 管作为开关器件的。按照导电载流子的不同，MOS 管分为 N 沟道 MOS(NMOS)管和 P 沟道 MOS(PMOS)管。按照导电沟道形成机理的不同分为增强型和耗尽型。

1. N 沟道增强型 MOS 管的结构和工作原理

N 沟道增强型 MOS 管的结构示意图及符号如图 3.2.1 所示。它是在 P 型衬底上，用扩散法制作两个高掺杂浓度的 N 区。然后在 P 型硅表面生长一层很薄的二氧化硅绝缘层，并在二氧化硅表面及两个 N 型区各安置一个电极，形成栅极 g、源极 s 和漏极 d。由于栅极被绝缘，其电阻高达 $10^{12} \sim 10^{15} \Omega$ 。通常将衬底与源极相连，或接地电位，以防止有电流从衬底流入源极和导电

沟道。

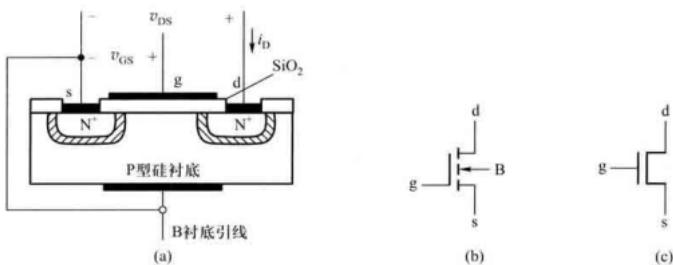


图 3.2.1 N 沟道增强型 MOS 管

(a) 结构示意图 (b) 标准符号 (c) 简化符号

如果栅极和源极之间所加电压 $v_{GS}=0$, 则源区、衬底和漏区形成的两个 PN 结背靠背串联, d、s 间不导通, $i_D=0$ 。

当栅源之间加正向电压 v_{GS} , 且 $v_{GS} \geq V_T$ (V_T 为开启电压) 时, 栅极和衬底之间形成足够强的电场, 吸引衬底中的少数载流子(电子), 使其聚集在栅极下的衬底表面, 形成 N 型反型层, 该反型层就构成了 d、s 间的导电沟道。若此时漏极和源极之间加电压 v_{DS} , 将有漏极电流 i_D 产生, 如图 3.2.2 所示。这种在 $v_{GS}=0$ 时不存在导电沟道, v_{GS} 必须增强到足够大时才形成导电沟道的场效应管, 称为 N 沟道增强型 MOS 管。

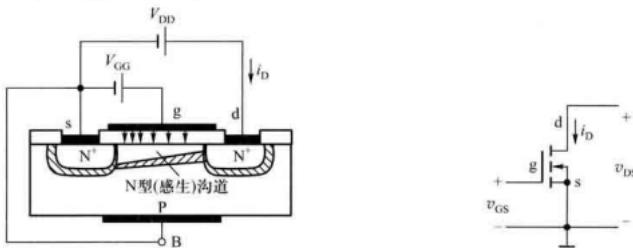


图 3.2.2 外加电压使 MOS 管形成导电沟道

图 3.2.3 共源极连接

2. N 沟道增强型 MOS 管的输出特性和转移特性

MOS 管可视为二端口网络, 如图 3.2.3 所示, 栅-源为输入端口, 漏-源为输出端口, 源极为公共端, 故称共源极连接。当端口电压不同时, 回路电流也将发生变化。因此用 $I-V$ 特性曲线反映电压与电流的关系。MOS 管的 $I-V$ 特性包括输出特性和转移特性, 分别如图 3.2.4(a)、(b) 所示。

输出特性曲线是指栅源电压 v_{GS} 一定的情况下, 漏极电流 i_D 与漏源电压 v_{DS} 之间的关系。输

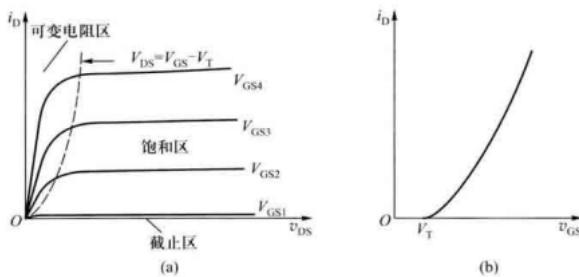


图 3.2.4 N 沟道增强型 MOS 管的特性曲线

(a) 输出特性曲线 (b) 转移特性曲线

出特性曲线分为三个工作区：截止区、饱和区和可变电阻区。

当 $v_{GS} < V_T$ 时，导电沟道尚未形成， $i_D = 0$ ，漏-源间电阻很大，可达 $10^9 \Omega$ 以上，相当于断开，MOS 管处于截止工作状态，特性曲线的该区域称为截止区。

当 $v_{GS} \geq V_T$ 时，产生导电沟道，外加 v_{DS} 较小时， i_D 随 v_{DS} 呈线性增长。此时 MOS 管可以看成一个受 v_{GS} 控制的可变电阻 r_{ds} ， v_{GS} 越大，输出特性曲线越倾斜，等效电阻越小。因此，该区域称为可变电阻区。 r_{ds} 由下式确定：

$$r_{ds} = \frac{dv_{DS}}{di_D} \Big|_{v_{GS}=\text{const}} = \frac{1}{2K_n(v_{GS}-V_T)} \quad (3.2.1)$$

式中 $K_n = \frac{k'_n}{2} \cdot \frac{W}{L}$ 称为电导常数， k'_n 对于特定制造工艺是常数， W/L 为 MOS 管沟道的宽长比。

由式(3.2.1)可知，为使 r_{ds} 尽可能小，应当使 v_{GS} 尽可能大。

当 v_{DS} 继续增加到一定数值使 $v_{DS} = v_{GS} - V_T$ 时，沟道在靠近漏极处开始消失，称为预夹断。随着 v_{DS} 继续增加， i_D 几乎不再增加，此时的区域称为饱和区。因此，当 $v_{DS} < v_{GS} - V_T$ 时，N 沟道 MOS 管工作于可变电阻区。当 $v_{DS} \geq v_{GS} - V_T$ 时，MOS 管工作于饱和区。

转移特性是指在漏源电压 v_{DS} 一定的条件下，栅源电压 v_{GS} 对漏极电流 i_D 的控制作用。当 MOS 管工作在饱和区时，由于 v_{DS} 对 i_D 的影响很小，所以不同的 v_{DS} 所对应的转移特性曲线基本重合，可以用一条曲线来表示。这条曲线与横坐标的交点即为开启电压 V_T 。

3. 其他类型的 MOS 管

(1) P 沟道增强型 MOS 管

与 N 沟道 MOS 管相反，P 沟道 MOS 管是在 N 型衬底上

制作两个高浓度的 P 区，导电沟道为 P 型，载流子为空穴。

其符号如图 3.2.5 所示。通常将衬底与源极相连，或接电源。

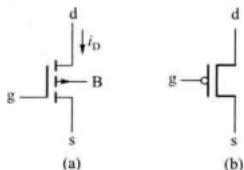


图 3.2.5 P 沟道增强型 MOS 管符号

(a) 标准符号 (b) 简化符号

为吸引空穴形成导电沟道, 棚极接电源负极, 与衬底相连的源极接电源的正极, 即 v_{GS} 为负值, 因此开启电压 V_T 也为负值。而 i_D 的实际方向为流出漏极, 与通常的假定方向正好相反。图 3.2.6 所示为 P 沟道增强型 MOS 管的输出特性和转移特性曲线。

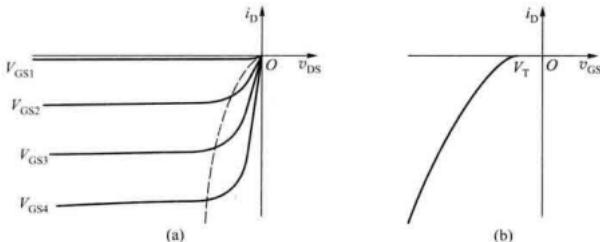
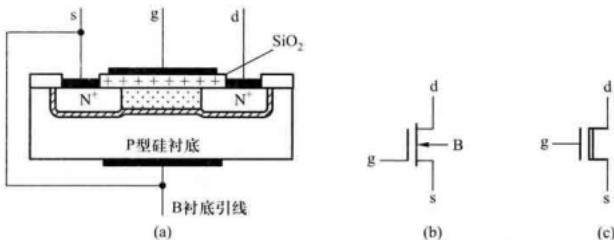


图 3.2.6 P 沟道增强型 MOS 管的特性曲线

(a) 输出特性曲线 (b) 转移特性曲线

(2) N 沟道耗尽型 MOS 管

N 沟道耗尽型 MOS 管的结构与增强型基本相同, 最大的区别是在生产过程中, 在 SiO_2 绝缘层中掺入大量正离子, 其结构示意图与符号如图 3.2.7 所示。 $v_{GS}=0$ 时, 由于正离子的作用, 将电子吸引到棚极下面的衬底表面形成 N 型沟道。当 $v_{GS}>0$ 时, 沟道变宽。在 v_{DS} 作用下, i_D 的数值更大。 v_{GS} 为负, 沟道变窄, i_D 减小, 当 v_{GS} 达到某一负值 V_p 时, 沟道完全被夹断, 即使加 v_{DS} , 也不会有漏极电流 i_D , MOS 管截止, V_p 称为夹断电压。

图 3.2.7 N 沟道耗尽型 MOS 管
(a) 结构示意图 (b) 标准符号 (c) 简化符号

(3) P 沟道耗尽型 MOS 管

P 沟道耗尽型 MOS 管的结构与增强型基本相同, 但在 SiO_2 绝缘层中掺入大量负离子, 形成 P 型导电沟道, 其夹断电压 V_p 为正值。 v_{GS} 可以是负值、零或正值。 v_{GS} 为负值时 i_D 增加, v_{GS} 为正值时 i_D 减小。当 v_{GS} 达到 V_p 时, 沟道完全被夹断, MOS 管截止。P 沟道耗尽型 MOS 管的符号如

图 3.2.8 所示。

4. MOS 管开关电路

用 N 沟道增强型 MOS 管替代图 3.1.3 所示的开关 S 构成的电路如图 3.2.9 所示。MOS 管的作用对应于有触点开关 S 的“断开”和“闭合”，但在速度和可靠性方面比机械开关优越得多。

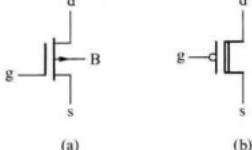


图 3.2.8 P 沟道耗尽型 MOS 管符号
(a) 标准符号 (b) 简化符号

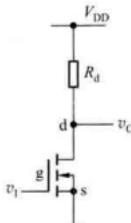


图 3.2.9 MOS 管开关电路

当 $v_i < V_T$ 时，MOS 管处于截止状态，输出电压 $v_o = V_{DD}$ 。此时器件不消耗功率。

当 $v_i > V_T$ ，并且足够大时，使得 MOS 管工作在可变电阻区。 d, s 之间的导通电阻 R_{on} 可由式(3.2.1)确定。 v_{GS} 的取值足够大时， R_{on} 很小，使得 R_d 远远大于 R_{on} ，电路输出为低电平。

由此可见，MOS 管相当于一个由 v_{GS} 控制的无触点开关。当输入为低电平时，MOS 管截止，相当于开关“断开”，输出为高电平，其等效电路如图 3.2.10(a) 所示；当输入为高电平时，MOS 管工作在可变电阻区，相当于开关“闭合”，输出为低电平，其等效电路如图 3.2.10(b) 所示。MOS 管导通时的等效电阻 R_{on} 约在 $1 \text{ k}\Omega$ 以内。当 v_i 足够大时，可以使 R_{on} 为 $25 \sim 200 \Omega$ 。

5. MOS 管开关电路的动态特性

在图 3.2.9 所示 MOS 管开关电路的输入端，加一个理想的脉冲波形，如图 3.2.11(a) 所示。由于 MOS 管中栅极与衬底间电容 C_{sb} （即数据手册中的输入电容 C_I ）、漏极与衬底间电容 C_{db} 、栅极与漏极电容 C_{gd} 以及导通电阻等的存在，使其在导通和闭合两种状态之间转换时，不可避免地受到电容充、放电过程的影响。输出电压 v_o 的波形已不是与输入一样的理想脉冲，如图 3.2.11(b) 所示。上升沿和下降沿的变化都变得缓慢了，而且输出 v_o 的变化滞后于输入 v_i 的变化。 t_{pH} 为输出 v_o 由高电平跳变为低电平的传输延迟时间， t_{pHL} 为输出 v_o 由低电平跳变为高电平的传输

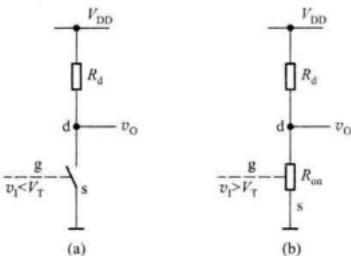


图 3.2.10 MOS 管的开关等效电路
(a) 截止时的等效电路 (b) 导通时的等效电路

延迟时间。

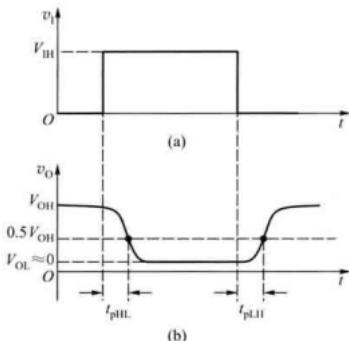


图 3.2.11 MOS 管开关电路电压波形
(a) 输入电压波形 (b) 输出电压波形

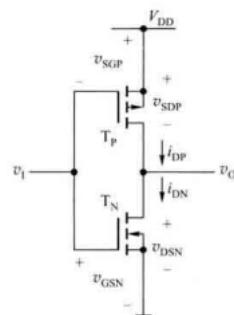


图 3.2.12 CMOS 反相器

图 3.2.9 所示电路中 R_d 的作用是：当输入为高电平时，流过导通 NMOS 管的电流很大， R_d 起限流作用，但此时消耗在其上的功率也很大。为了克服这个缺点，用另一个 PMOS 管替代电阻 R_d ，就构成了 CMOS 反相器。

3.2.2 CMOS 反相器

由 N 沟道和 P 沟道增强型 MOS 管组成的电路称为互补 MOS 或 CMOS 电路。CMOS 反相器是构成 CMOS 逻辑电路的基本单元电路之一，另一个基本单元电路——传输门将在 3.2.4 节介绍。下面讨论 CMOS 反相器的工作原理。

CMOS 反相器电路如图 3.2.12 所示，由两只增强型 MOS 管组成，其中 T_N 为 N 沟道 MOS 管， T_P 为 P 沟道 MOS 管。两只 MOS 管的栅极连在一起作为输入端；它们的漏极连在一起作为输出端。按照图中标明的电压与电流方向， $v_i = v_{GSN}$, $v_o = v_{DSN}$ ，并设 $i_{DN} = i_{DP} = i_0$ 。为了电路能正常工作，要求电源电压 V_{DD} 大于两只 MOS 管的开启电压的绝对值之和，即 $V_{DD} > (V_{TN} + |V_{TP}|)$ 。为方便叙述，将 N 沟道和 P 沟道增强型 MOS 管的开启电压分别用 V_{TN} 和 V_{TP} 表示。

1. 工作原理

首先，我们考虑两种极限情况：当 v_i 处于逻辑 0 时，相应的电压近似为 0 V；而当 v_i 处于逻辑 1 时，相应的电压近似为 V_{DD} 。

当输入为高电平 $v_i = V_{DD}$ 时，根据图 3.2.12 所示电路图可知， $v_{GSN} = V_{DD} > V_{TN}$, T_N 管导通，并且导通电阻很低，通常为几百欧；而 $v_{SGP} = 0 < |V_{TP}|$, T_P 管截止，等效电阻很高，可达兆欧级以上。因此，反相器输出为低电平，输出电压 $v_o = V_{OL} \approx 0$ ，而通过两管的电流接近于零。这就是说，电路的功耗极低。

当输入为低电平 $v_i = 0$, 即 $v_{GSN} = 0 < V_{TN}$, T_N 管截止; 而 $v_{SGP} = V_{DD} > |V_{TP}|$, T_P 管导通。反相器输出为高电平, 输出电压 $v_o = V_{on} \approx V_{DD}$ 。

由此可知, 当输入 $v_i = V_{DD}$ 时, 输出 $v_o \approx 0$; 而输入 $v_i = 0$, 输出 $v_o \approx V_{DD}$ 。输出与输入之间为逻辑非的关系, 也称非门为反相器 (Inverter)。CMOS 反相器近似于一个理想的逻辑单元, 其输出电压接近于零或 $+V_{DD}$ 。

通过上述分析可以看出 CMOS 反相器具有以下几个重要特点。

第一, 当反相器处于稳态时, 无论输入 v_i 是高电平还是低电平, T_N 和 T_P 中总是处于一个导通而另一个截止的状态。截止管的等效电阻很大, 流过 T_N 和 T_P 的静态电流非常小。因此 CMOS 反相器的静态功耗非常低, 几乎为零。这是 CMOS 电路最显著的优点。

第二, MOS 管导通电阻很低, 而截止电阻很高。当反相器输出端接电容性负载, 其输出端由低电平变为高电平时, T_P 导通, 电源通过其导通电阻给负载电容提供快速充电回路。反相器输出电平由高变低时, T_N 导通, 负载电容通过其较小的导通电阻快速放电。与图 3.2.9 所示的电路相比较, CMOS 反相器的开关速度更快, 具有更强的带负载能力。

第三, MOS 管栅极是绝缘的, $I_G \approx 0$, 反相器的输入电阻非常高。因此, 理论上, 反相器可以驱动任意数目的同类门而不会对输出电平造成影响。但是, 负载门输入端具有杂散电容, 负载门数目的增加将使反相器的负载电容增加, 从而影响反相器的开关速度。

2. 电压传输特性和电流传输特性

CMOS 反相器的电压传输特性是指其输出电压 v_o 随输入电压 v_i 变化的关系曲线, 如图 3.2.13(a) 所示。电流传输特性是指漏极电流 i_D 随输入电压 v_i 变化的曲线, 如图 3.2.13(b) 所示。根据 T_N 和 T_P 两管工作情况的不同, 可将传输特性曲线分为五段。

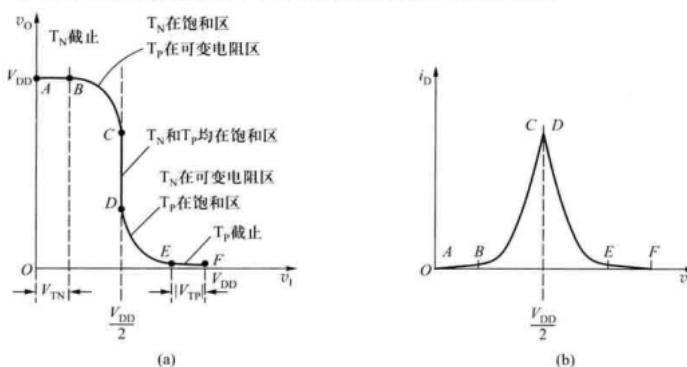


图 3.2.13 CMOS 反相器的传输特性

(a) 电压传输特性 (b) 电流传输特性

在传输特性曲线的 AB 或 EF 段, 根据前述 CMOS 反相器工作原理的两种极限情况分析可知, 不论输出为高电平或低电平, 总有一只 MOS 管工作在截止区, 因此, 流过两管的电流接近零值。

在 BC 或 DE 段, T_N 和 T_P 两管中, 总有一个工作在饱和区, 另一个工作在可变电阻区, 此时输出电流比较大, 传输特性变化比较快, 两管在 $v_i = V_{DD}/2$ 处转换状态。因此, 将状态转换处的电压定义为 CMOS 反相器的阈值电压 V_{TH} ^①, $V_{TH} = V_{DD}/2$ 。

在 CD 段, 由于 T_N 和 T_P 两管均工作在饱和区, 此时 $v_i = V_{DD}/2$, 电流 i_D 达到最大值。

从 B 到 E 之间, 即 $V_{TH} < v_i < V_{DD} - |V_{TP}|$ 时, T_N 和 T_P 处于同时导通的过渡区域, 传输特性变化急剧, 产生一个较大的电流尖峰。因而导致有较大的功耗。使用时应避免使两管长时间工作在此区域, 以防止功耗过大而损坏。

3. 输入逻辑电平

由图 3.2.13(a) 可见, 当输入电压从 0 V 开始逐渐增加时, 输出高电平维持一段时间没有改变。同样, 当输入电压由 V_{DD} 开始降低时, 输出低电平也维持一段时间没有改变。因此, 在反相器的输出逻辑状态没有发生明显改变时, 输入高、低电平允许有一个波动范围, 如图 3.2.14(a) 所示。对于不同系列的集成电路, 输入高、低电平所对应的电压范围也不同。因此, 各种集成门电路都规定了输入低电平的上限值 $V_{IL(max)}$ 和输入高电平的下限值 $V_{IH(min)}$ 。以 74LVC 系列 CMOS 逻辑电路为例, 在 3.3 V 工作电源条件下, $V_{IL(max)} = 0.8 \text{ V}$, $V_{IH(min)} = 2.0 \text{ V}$, 即输入电压在 0 ~ 0.8 V 电压范围内, 输出高电平; 输入电压在 2.0 ~ 3.3 V 范围内, 输出低电平。

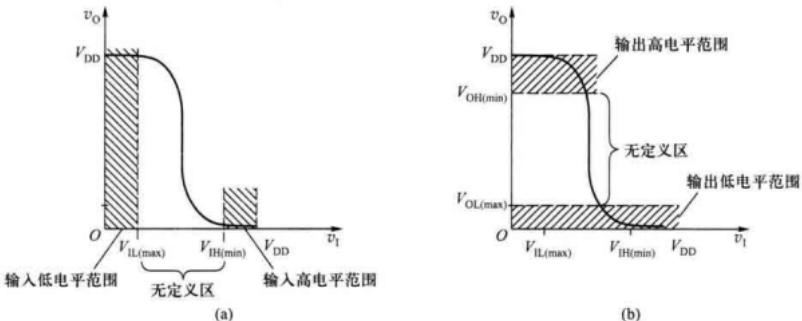


图 3.2.14 门电路输入和输出逻辑电平范围

(a) 输入高、低电平 (b) 输出高、低电平

① V_{TH} 系 Threshold voltage 之意。

4. 输出逻辑电平

对于图 3.2.12 所示的 CMOS 反相器,输出高、低电平值也允许有一个波动范围,如图 3.2.14(b)所示。当输入为高电平时,则 $v_{GSN} = v_i = V_{IH}$, T_N 管导通, T_P 管截止,输出为低电平,即 $v_o = V_{OL}$,等效电路如图 3.2.15(a)所示。图中 R_L 为 CMOS 反相器所带负载的等效电阻。此时负载电流 I_{OL} 从负载流向 T_N 管,称为灌电流负载。

当 v_i 足够大时, T_N 管工作在输出特性曲线的可变电阻区($v_{DS} < v_{GS} - V_T$),如图 3.2.4(a)所示。此时 T_N 管的导通电阻 $R_{on(N)}$ 与 v_{GSN} 的关系可由式(3.2.1)确定。

需要特别注意,由于 $V_{OL} = R_{on(N)} I_{OL}$,所以 V_{OL} 的值与 T_N 管导通电阻 $R_{on(N)}$ 和负载电流 I_{OL} 两个因素有关。若 $R_{on(N)}$ 不变, V_{OL} 随着 I_{OL} 增加(由负载 R_L 引起)而升高。若 I_{OL} 不变,由式(3.2.1)可知,当 v_i 的高电平值越低, v_{GS} 越小,则 $R_{on(N)}$ 越大,导致 V_{OL} 也越高。因此,集成电路数据手册给出了在不同 I_{OL} 条件下,输出低电平的值 V_{OL} 。例如低电压 CMOS 系列与非门 74LVC00 在 3.3 V 工作电源条件下,当 $I_{OL} = 100 \mu A$ 时, $V_{OL(max)} = 0.1 V$ 。这里 $V_{OL(max)}$ 的下标 max 的含义是,由于器件参数的分散性,即使在相同条件下,相同型号器件的输出低电平的值也略有区别,但它们的 V_{OL} 值不超过 $V_{OL(max)}$ 。

同理,当输入为低电平时, T_N 管截止, T_P 管导通,输出为高电平,即 $v_o = V_{OH}$,等效电路如图 3.2.15(b)所示。 R_L 为 CMOS 反相器所带负载的等效电阻, $R_{on(P)}$ 为 T_P 管的导通电阻。此时负载电流 I_{OH} 从 T_P 管流向负载,称为拉电流负载。

V_{OH} 的值与 T_P 管导通电阻 $R_{on(P)}$ 和负载电流 I_{OH} 两个因素有关。由于 $V_{OH} = V_{DD} - R_{on(P)} I_{OH}$,随着 I_{OH} 的增加(由 R_L 引起), V_{OH} 将降低。集成电路数据手册给出了不同 I_{OH} 条件下,输出高电平的值 V_{OH} 。74LVC00 器件在 3.3 V 工作电源条件下,当 $I_{OH} = -100 \mu A$ (规定电流从器件流出为负)时, $V_{OH(min)} = 3.1 V$;当 $I_{OH} = -12 mA$ 时, $V_{OH(min)} = 2.7 V$ 。这里 $V_{OH(min)}$ 是指相同条件下,相同型号器件的输出高电平的值也略有区别,但它们的 V_{OH} 值不低于 $V_{OH(min)}$ 。

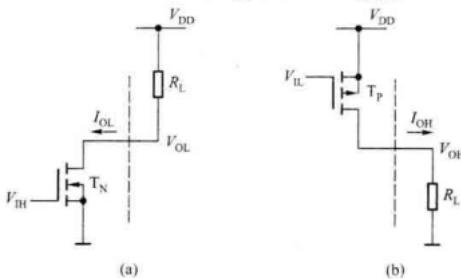


图 3.2.15 CMOS 反相器的输出特性
(a) 输出低电平等效电路 (b) 输出高电平等效电路

5. 工作速度

CMOS 反相器或 CMOS 电路用于驱动其他 MOS 器件时, 其负载的阻抗是电容性的, 如图 3.2.16(a) 所示。当 $v_i = 0$ 时, T_N 截止, T_P 导通, 由 V_{DD} 通过 T_P 向负载电容 C_L 充电, 如图 3.2.16(b) 所示。此时, $|V_{GSP}| = V_{DD}$, 根据式(3.2.1)可知, T_P 的导通电阻较小, 充电回路的时间常数较小。类似地, 当 $v_i = V_{DD}$ 时, T_N 导通, T_P 截止, 电容 C_L 的放电回路如图 3.2.16(c) 所示。由于电路具有互补对称的性质, T_N 与 T_P 的导通电阻相等, 因此, 传输延迟时间 t_{PLH} 和 t_{PHL} 是基本相等的。

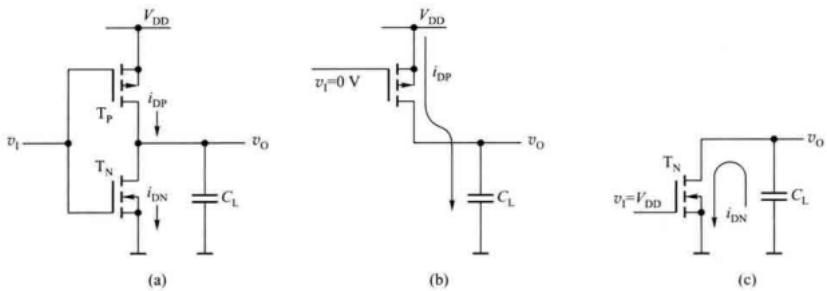


图 3.2.16 CMOS 反相器在电容负载下的工作情况

(a) 电路图 (b) 负载电容充电 (c) 负载电容放电

3.2.3 其他基本 CMOS 逻辑门电路

CMOS 系列基本逻辑门电路中, 除上述介绍的非门(反相器)外, 还有与门、或门、与非门、或非门、异或门等电路。下面重点介绍与非门和或非门。

1. 与非门电路

图 3.2.17 是 2 输入端 CMOS 与非门电路, 其中包括两个串联的 N 沟道增强型 MOS 管和两个并联的 P 沟道增强型 MOS 管。每个输入端连到一个 N 沟道和一个 P 沟道 MOS 管的栅极。电路输出与输入信号逻辑关系及各个 MOS 管的工作状态如表 3.2.1 所示。当输入端 A、B 有一个为低电平时, 就会使与它相连的 NMOS 管截止, PMOS 管导通, 输出为高电平; 仅当 A、B 全为高电平时, 才会使两个串联的 NMOS 管都导通, 使两个并联的 PMOS 管都截止, 输出为低电平。

因此, 这种电路具有与非的逻辑功能, 即

$$L = \overline{A \cdot B}$$

n 个输入端的与非门必须有 n 个 NMOS 管串联和 n 个 PMOS 管并联。

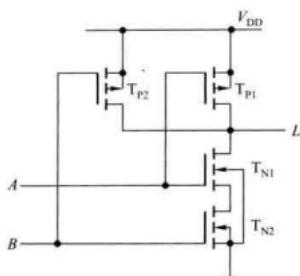


图 3.2.17 CMOS 与非门

表 3.2.1 与非门输入输出关系及各 MOS 管工作状态

A	B	T_{N1}	T_{P1}	T_{N2}	T_{P2}	L
0	0	截止	导通	截止	导通	1
0	1	截止	导通	导通	截止	1
1	0	导通	截止	截止	导通	1
1	1	导通	截止	导通	截止	0

2. 或非门电路

图 3.2.18 是 2 输入端 CMOS 或非门电路，其中包括两个并联的 N 沟道增强型 MOS 管和两个串联的 P 沟道增强型 MOS 管。

电路输出与输入信号逻辑关系及各个 MOS 管的工作状态如表 3.2.2 所示。当输入端 A、B 只要有一个为高电平时，就会使与它相连的 NMOS 管导通，而 PMOS 管截止，输出为低电平；仅当 A、B 全为低电平时，两个并联 NMOS 管都截止，两个串联的 PMOS 管都导通，输出为高电平。

因此，这种电路具有或非的逻辑功能，其逻辑表达式为

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

显然，n 个输入端的或非门必须有 n 个 NMOS 管并联和 n 个 PMOS 管串联。

从以上 CMOS 与非门和或非门电路可知，输入端的数目越多，则串联的管子也越多。若串联的管子全部导通时，其总的导通电阻会增加，以致影响输出电平，使与非门的低电平升高，使或非门的高电平降低。因此 CMOS 逻辑门电路的输入端不宜过多。

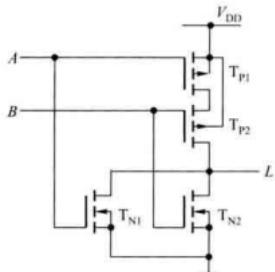


表 3.2.2 或非门输入输出关系及各 MOS 管工作状态

A	B	T_{N1}	T_{P1}	T_{N2}	T_{P2}	L
0	0	截止	导通	截止	导通	1
0	1	截止	导通	导通	截止	0
1	0	导通	截止	截止	导通	0
1	1	导通	截止	导通	截止	0

图 3.2.18 CMOS 或非门

例 3.2.1 试分析图 3.2.19 所示的 CMOS 逻辑门电路,写出其逻辑表达式,说明其逻辑功能。

解: 从电路结构可以看出,虚线左边是或非门结构。虚线右边的两个 NMOS 管串联相与,然后与另一个 NMOS 管并联相或,并且三个 PMOS 管也有相应的连接,所以是与或非门结构。或非门的输出 $X = \overline{A+B}$ 。而与或非门的输出 L 为

$$L = \overline{A \cdot B + X} = \overline{A \cdot B + \overline{A+B}} = \overline{A \cdot B + \overline{A} \cdot \overline{B}} = \overline{A \odot B} = A \oplus B$$

电路实现异或逻辑功能。如果在异或门的后面增加一级反相器就构成异或非门,此时其表达式为 $\overline{L} = A \cdot B + \overline{A} \cdot \overline{B}$, 可以实现同或逻辑功能。

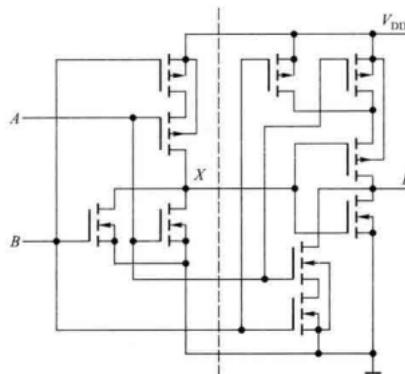


图 3.2.19 例 3.2.1 的 CMOS 逻辑电路

3.2.4 CMOS 传输门

传输门(Transmission Gate, TG)的应用比较广泛,不仅可以作为基本单元电路构成各种逻辑电路,用于数字信号的传输,而且可以在取样-保持电路、斩波电路、模数和数模转换等电路中传输模拟信号,因而又称为模拟开关。

1. 传输门的结构及工作原理

CMOS 传输门由一个 P 沟道和一个 N 沟道增强型 MOS 管并联而成,如图 3.2.20(a)所示。图 3.2.20(b)是它的逻辑符号。 T_N 和 T_P 是结构完全对称的,衬底的引线与普通 MOS 管不同。所以栅极的引出端画在符号横线的中间。它们的漏极和源极可以互换,因而传输门的输入和输出端可以互换使用,即为双向器件。设它们的开启电压 $V_{TN} = |V_{TP}| = V_T$, C 和 \bar{C} 是一对互补的控制信号。关于衬底连接的原则:NMOS 管的衬底连接到电路中最低电位点,而 PMOS 管的衬底连接到电路中最高电位点,这是为了防止电流从漏极直接流入衬底,使衬底与漏源极之间形成的

PN 结反向偏置。因此,将 N 沟道 MOS 管的衬底连接到地电位, P 沟道的衬底接 $+V_{DD}$ 电压。

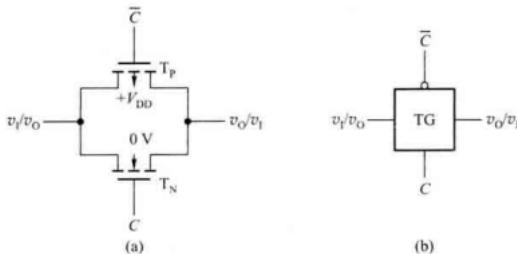


图 3.2.20 CMOS 传输门

(a) 电路结构 (b) 逻辑符号

传输门的工作情况如下:当 C 端接 0 V , \bar{C} 端接 $+V_{DD}$ 时,输入信号 v_I 的取值在 $0 \sim +V_{DD}$ 范围内, T_N 和 T_P 同时截止,输入和输出之间呈高阻态,传输门是断开的。

当 C 端接 $+V_{DD}$, \bar{C} 端接 0 V 时, v_I 在 $0\text{ V} \sim +(V_{DD} - V_T)$ 的范围内, T_N 导通。 v_I 在 $+V_T \sim +V_{DD}$ 的范围内, T_P 将导通。由此可知,当 v_I 在 $0\text{ V} \sim +V_{DD}$ 之间变化时, T_N 和 T_P 至少有一个导通,使 v_I 与 v_O 之间的导通电阻很小,传输门导通。

进一步分析还可看到,当输入电压变化时,使两管的栅源电压 v_{GS} 均发生变化。而 MOS 管漏源间的等效电阻是 v_{GS} 的函数,如式(3.2.1)所示。因此,两管漏源间的等效电阻随输入电压的变化而变化。一管导通的程度越深,另一管的导通程度则相应地减小。也就是当一管的等效电阻减小,则另一管的等效电阻就增加。由于具有互补作用的两管并联在一起,使传输门导通电阻的变化相对各单管等效电阻的变化小得多,这是传输门的优点。

2. 模拟开关

当 CMOS 传输门用作模拟开关时,若输入信号的变化范围为 $-V_{SS}$ 到 $+V_{DD}$,则 T_N 和 T_P 的衬底分别接 $-V_{SS}$ 和 $+V_{DD}$ 。互补控制端 C 和 \bar{C} 的控制电压分别为 $+V_{DD}$ 和 $-V_{SS}$,传输门导通。 C 和 \bar{C} 的控制电压分别为 $-V_{SS}$ 和 $+V_{DD}$,传输门断开。

模拟开关的导通电阻与输出端的负载构成分压器,输出电压是两者对输入电压分压产生的。因此,导通电阻的稳定可以使输出电压随输入电压的变化呈线性关系。但模拟开关的导通电阻不是恒定的,因此,推出了多种改进的电路,其目的是为了使导通电阻尽可能小,并且在输入信号的变化范围内使导通电阻尽可能保持不变。有些精密 CMOS 模拟开关的导通电阻已达到几欧。

一般的模拟开关导通电阻值在数百欧以下,当它与输入阻抗为兆欧级的运放或输入电阻达 $10^{10}\Omega$ 以上的 MOS 电路串接时,可以忽略不计。

3. 传输门在数字电路的应用

CMOS 传输门除了作为传输模拟信号的开关外,由于它的传输延迟时间短、结构简单,也作

为基本单元电路,用于构成各种逻辑电路,如数据选择/分配器、触发器等。

由 CMOS 传输门构成的异或门电路如图 3.2.21 所示,输入信号 B 作为传输门的控制信号。当 $B=0$ 时, TG₁ 截止,TG₂ 导通, $L=A$; 当 $B=1$ 时, TG₁ 导通,TG₂ 截止, $L=\bar{A}$ 。 L 与 A 、 B 是异或逻辑关系,即 $L=A\bar{B}+\bar{A}B=A\oplus B$ 。

由 CMOS 传输门构成的 2 选 1 数据选择器如图 3.2.22 所示。当控制端 $C=0$ 时,输入端 A 的信号被传到输出端, $L=A$ 。而当 $C=1$ 时, $L=B$ 。

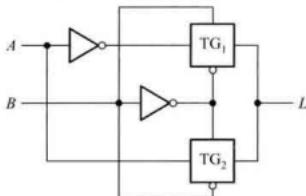


图 3.2.21 传输门构成的异或门

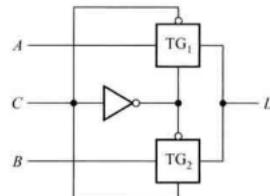


图 3.2.22 传输门构成的数据选择器

复习思考题

- 3.2.1 在数字电路中 MOS 管工作在输出特性的什么区域,如何保证其工作在这些区域?
- 3.2.2 开启电压 V_T 的含义是什么?
- 3.2.3 N 沟道增强型 MOS 管的导通电阻 R_{on} 与 v_{GS} 有怎样的关系?
- 3.2.4 CMOS 反相器的静态功耗是指什么?为什么其静态功耗几乎等于零?
- 3.2.5 CMOS 反相器在什么时候电流 i_o 达到最大值?
- 3.2.6 CMOS 电路的输出低电平 V_{OL} 与负载电流 I_{OL} 有怎样的关系?输出高电平 V_{OH} 与负载电流 I_{OH} 又有怎样的关系?
- 3.2.7 增加 CMOS 与非门输入端会增加串联的 NMOS 管数目,对电路输出电平有什么影响?增加或非门的输入端呢?
- 3.2.8 为什么 CMOS 传输门又称为模拟开关?传输门在数字电路和模拟电路中的连接方式有什么不同?

3.3 CMOS 逻辑门电路的不同输出结构及参数

实际 CMOS 逻辑门电路的输入和输出端都有保护电路和缓冲电路。另外,使用时如果需要将两个 CMOS 逻辑门的输出端连在一起,则需要选择漏极开路的逻辑门。如果希望对 CMOS 逻辑门电路的输出加以控制,则选择三态输出逻辑门电路。

3.3.1 CMOS 逻辑门的保护和缓冲电路

CMOS 门电路的输入端是 MOS 管的栅极,在栅极与沟道之间是很薄的 SiO₂ 层,小于

$0.1 \mu\text{m}$, 极易被击穿。而输入电阻高达 $10^{12} \Omega$ 以上, 输入电容为几皮法。电路在使用前输入端是悬空的, 只要外界有很小的静电源, 都会在输入端积累电荷而将栅极击穿。因此, CMOS 电路都增加了保护电路。

当逻辑门电路的输入端数目不同, 或所带负载门的数目不同时, 其输入、输出特性也不同, 会给电路设计工作带来麻烦, 同时要求输出端具有一定驱动能力。因此, CMOS 逻辑门通常用反相器作输入或输出缓冲电路。由于这些缓冲电路具有统一的参数, 使得集成逻辑门电路的输入和输出特性不再因内部逻辑不同而发生变化, 从而使电路的性能得到改善。

实际 CMOS 逻辑门通常有输入、输出保护电路和缓冲电路。以 74HC/HCT 系列为例, 其电路结构如图 3.3.1 所示。图中的基本逻辑功能电路可以是前面介绍的反相器、与非门、或非门或者它们的组合电路。

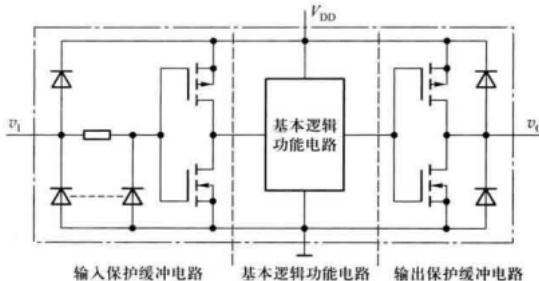


图 3.3.1 实际集成 CMOS 门电路结构图

1. 输入保护电路

图 3.3.2 所示为输入保护电路和输入缓冲电路。图中 C_N 和 C_P 分别表示 T_N 和 T_P 的栅极等效电容, D_1 和 D_2 是正向导通压降为 $V_{DF} = 0.5 \sim 0.7 \text{ V}$ 的二极管, D_2 是分布式二极管结构, 用虚线和两个二极管表示。这种分布式二极管结构可以通过较大的电流, 使得输入引脚上的静电荷得以释放, 从而保护了 MOS 管的栅极绝缘层。二极管的反向击穿电压约为 30 V , 小于栅极 SiO_2 层的击穿电压。

输入电压在正常范围内 ($0 \leq v_i \leq V_{DD}$) , 保护电路不起作用。当 $v_i > (V_{DD} + V_{DF})$ 或 $v_i < -V_{DF}$ 时, MOS 管的栅极电位被限制在 $-V_{DF}$ 到 $(V_{DD} + V_{DF})$ 之间, 使栅极的 SiO_2 层不会被击穿。如果输入电平发生突变时的过冲电压超出上述输入电压范围, 可能使二极管 D_1 或 D_2 首先被击穿。当过冲时间较短时, 二极管仍能恢复

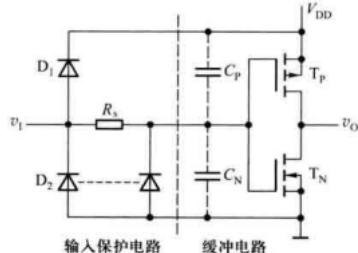


图 3.3.2 输入保护电路及缓冲电路

工作;当过冲时间较长或过冲电压很大时,可能损坏二极管,进而使 MOS 管栅极被击穿。

另外,电阻 R_s 和 MOS 管的栅极电容组成积分网络,使输入信号的过冲电压延迟一段时间才作用到栅极上,而且幅度有所衰减。为减小这种延迟对电路动态性能的影响, R_s 值不宜过大,通常常 R_s 为 $2\text{ k}\Omega$ 左右。

逻辑门电路输出端也接入静电保护二极管,确保输出不超出正常的工作范围。

2. 反相缓冲电路

图 3.3.3 所示为带缓冲级的 CMOS 与非门电路的逻辑符号。由于输入、输出端加了反相器作为缓冲电路,所以电路输出与输入的逻辑关系也发生了变化。图中的基本逻辑电路是或非门,增加了缓冲器后的逻辑关系为与非功能,即

$$L = \overline{\overline{A} + \overline{B}} = \overline{A \cdot B}$$

3.3.2 CMOS 漏极开路门和三态输出门电路

前面讨论了具有输入、输出缓冲电路的 CMOS 集成电路。如果从输出端看,还有另外两种输出结构的 CMOS 门电路,漏极开路门(Open Drain, OD)和三态输出门(Tristate Logic, TSL)。下面分别加以讨论。

1. CMOS 漏极开路门

(1) 漏极开路门的结构及工作原理

通常 CMOS 门电路都有反相器作输出缓冲电路。而在工程实践中,有时需要将两个门的输出端并联以实现与逻辑的功能称为线与,或者用于驱动大电流负载,或者实现逻辑电平变换。现在来考察一种情况,如果将两个 CMOS 与非门 G_1 和 G_2 的输出端连接在一起,如图 3.3.4 所示,并设 G_1 的输出处于高电平, T_{N1} 截止, T_{P1} 导通; 而 G_2 的输出为低电平, T_{N2} 导通, T_{P2} 截止。这样,从 G_1 的 T_{P1} 到 G_2 的 T_{N2} 将形成一低阻通路,从而产生很大的电流,有可能导致器件的损毁,并且无法确定输出是高电平还是低电平。这一问题可以采用 OD 门来解决。

所谓漏极开路是指 CMOS 门电路的输出电路只有 NMOS 管,并且它的漏极是开路的。漏极开路的与非门电路及逻辑符号如图 3.3.5(a) 和 3.3.5(b) 所示,其中图标“ Δ ”表示漏极开路之意。

使用 OD 门时必须在漏极和电源 V_{DD} 之间,外接一个上拉电阻 R_p ^①。图 3.3.6 所示为两个 OD 与非门实现线与。将两个门电路输出端接在一起,通过上拉电阻接电源。由图 3.3.6(a) 可见,当两个与非门的输出全为 1 时,输出为 1; 只要其中一个为 0 时,输出为 0。所以该电路的输出符合与逻辑功能,即 $L = \overline{\overline{AB} \cdot \overline{CD}}$ 。图 3.3.6(b) 所示为两个 OD 与非门线与的逻辑图。

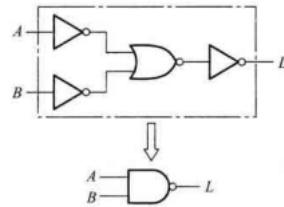


图 3.3.3 带缓冲级的 CMOS 与非门的逻辑图

① 上拉电阻系 pull-up resistor 的译称。

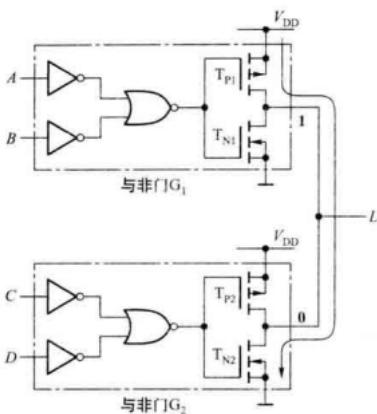


图 3.3.4 普通 CMOS 与非门输出端相连

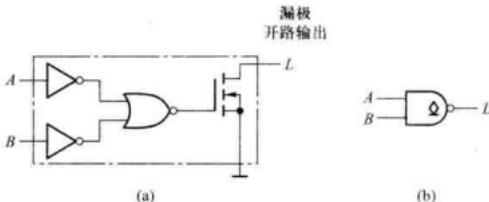


图 3.3.5 漏极开路(OD)与非门

(a) 电路结构 (b) 逻辑符号

(2) 上拉电阻对 OD 门动态性能的影响

当其他门电路作为 OD 门的负载时，OD 门称为驱动门，其后所接的门电路称为负载门。由于驱动门的输出电容、负载门的输入电容以及接线电容的存在，上拉电阻 R_p 的大小必将影响 OD 门的开关速度， R_p 的值越小，负载电容的充电时间常数也越小，因而开关速度越快。但上拉电阻不能任意地减小，它必须保证 OD 门输出端的电流不能超过允许的最大值 $I_{OL(max)}$ 。对于 74HC/HCT 系列 CMOS 门电路， $I_{OL(max)} = 4 \text{ mA}$ ，因此 R_p 必须大于 $V_{DD}/I_{OL(max)} = 5 \text{ V}/4 \text{ mA} = 1.25 \text{ k}\Omega$ 。与普通 CMOS 电路相比， R_p 的值比 PMOS 管导通电阻大，因而，OD 门从低电平到高电平的转换速度比普通 CMOS 门慢。

图 3.3.7 所示为 OD 门驱动电容性负载的工作情况，取上拉电阻 R_p 为 $1.5 \text{ k}\Omega$ 。由于输出状态发生变化时，电容的充、放电作用会对输出波形产生影响，所以主要考虑负载电容 C_L ，并假设

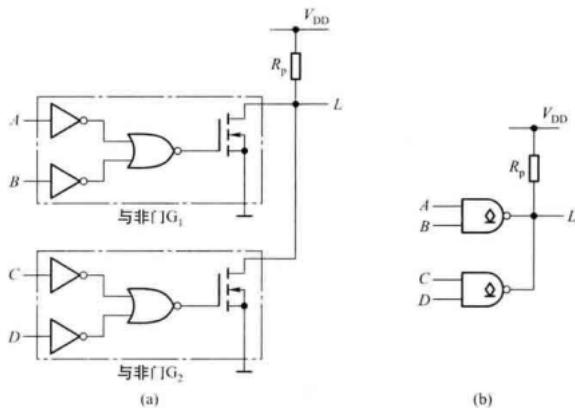


图 3.3.6 漏极开路(OD)与非门的线与
(a) 线与连接图 (b) 逻辑图

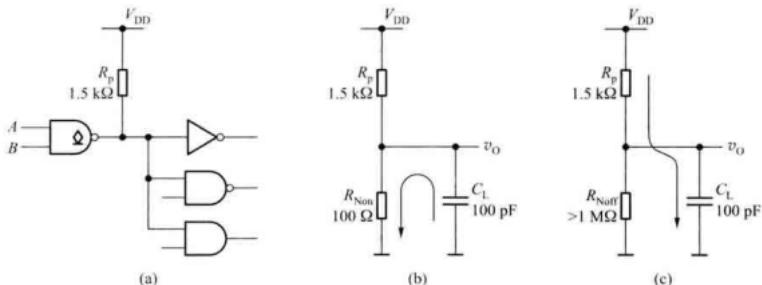


图 3.3.7 OD 门驱动电容性负载
(a) 逻辑图 (b) 输出为低电平时的等效电路 (c) 输出为高电平时的等效电路

C_L 为 100 pF, 下面分两种情况进行讨论。

当 OD 门输出由高电平变为低电平时, 其等效电路如图 3.3.7(b) 所示, 如果 NMOS 管导通时的等效电阻 $R_{on(N)}$ 为 100 Ω, 因此充了电的电容主要通过 NMOS 管放电, 放电时间常数 $\tau_{hl} = 100 \Omega \times 100 \text{ pF} = 10 \text{ ns}$ 。

当输出由低电平跳变为高电平时, 其等效电路如图 3.3.7(c) 所示, 此时电源通过 R_p 向 C_L 充电, 充电时间常数 $\tau_{lh} = 1.5 \text{ k}\Omega \times 100 \text{ pF} = 150 \text{ ns}$, 导致输出波形的上升沿时间很长。因此, 当工作速度较快时, 应尽量避免用 OD 门驱动大的电容性负载。

(3) 上拉电阻的计算

选择上拉电阻 R_p 的值时要考虑多种因素。一方面,从上面分析可知,如果负载具有电容性, R_p 的值越小,电容的充电时间常数也越小,因而开关速度越快,但功耗也越大。另一方面,多个 OD 门的输出端线与在一起,当只有一个门导通,输出为低电平,其他门均截止时,负载电流将全部流向导通的 OD 门,这是一种最不利的情况,此时上拉电阻 R_p 具有限制电流的作用,其取值不能太小。应保证 I_{OL} 不超过额定值 $I_{OL(max)}$ 。下面分两种情况计算 R_p 的最小值 $R_{p(min)}$ 及最大值 $R_{p(max)}$ 。

① 当输出为低电平时,并联的 OD 门中只有一个导通,参看图 3.3.8(a)。

为保证导通 OD 门的输出电流 $I_{OL} \leq I_{OL(max)}$,对于所有截止的 OD 门,忽略流过截止 NMOS 管的漏电流 I_{OZ} 。于是, $I_{OL} = I_{Rp} + I_{IL(total)} \leq I_{OL(max)}$, 并且 $V_{OL} = V_{OL(max)}$ 。求得 R_p 上的压降为 $V_{DD} - V_{OL(max)}$, 则流过 R_p 的电流为 $(V_{DD} - V_{OL(max)}) / R_p = I_{OL(max)} - I_{IL(total)}$ 。此时 R_p 应满足方程

$$R_p \geq \frac{V_{DD} - V_{OL(max)}}{I_{OL(max)} - I_{IL(total)}}$$

因此 R_p 的最小值 $R_{p(min)}$ 可按下式来确定:

$$R_{p(min)} = \frac{V_{DD} - V_{OL(max)}}{I_{OL(max)} - I_{IL(total)}} \quad (3.3.1)$$

式中: V_{DD} ——直流电源电压;

$V_{OL(max)}$ ——驱动门 V_{OL} 最大值;

$I_{OL(max)}$ ——驱动门 I_{OL} 最大值;

$I_{IL(total)}$ ——负载门低电平输入电流 I_{IL} 总和, $I_{IL(total)} = nI_{IL}$ 。这里需要注意 n 的取值,对于负载为 CMOS 门电路或者 TTL 或非门, n 为并联的输入端数目;对于 TTL 与非门负载, n 为负载门的个数,而不是输入端的数目,这是由于 TTL 与非门输入端的结构所致,有关 TTL 与非门的内容参见 3.5.3 节。

② 当所有 OD 门输出均为高电平时,参看图 3.3.8(b)。

为使得输出高电平不低于规定的 V_{OH} 的最小值,即 $V_{OH} \geq V_{OH(min)}$, 则 R_p 的选择不能过大。流过 R_p 的电流 $I_{Rp} = (V_{DD} - V_{OH}) / R_p$, 应满足 $I_{Rp} = I_{OZ(total)} + I_{IH(total)}$ 。则 $V_{OH} = V_{DD} - R_p(I_{OZ(total)} + I_{IH(total)}) \geq V_{OH(min)}$ 。

因此, R_p 的最大值 $R_{p(max)}$ 可按下式来确定:

$$R_{p(max)} = \frac{V_{DD} - V_{OH(min)}}{I_{OZ(total)} + I_{IH(total)}} \quad (3.3.2)$$

式中: $V_{OH(min)}$ ——驱动门 V_{OH} 最小值;

$I_{OZ(total)}$ ——全部驱动门输出高电平时的漏电流总和;

$I_{IH(total)}$ ——负载门高电平输入电流 I_{IH} 总和。 $I_{IH(total)} = nI_{IH}$, n 为负载门并联的输入端数目。

实际上, R_p 的值选在 $R_{p(min)}$ 和 $R_{p(max)}$ 之间,若要求电路速度快,选用 R_p 的值接近 $R_{p(min)}$ 的标准值。若要求电路功耗小,选用 R_p 的值接近 $R_{p(max)}$ 的标准值。

式(3.3.1)和式(3.3.2)中已考虑电流的方向,因此,所有电流参数均取正值。

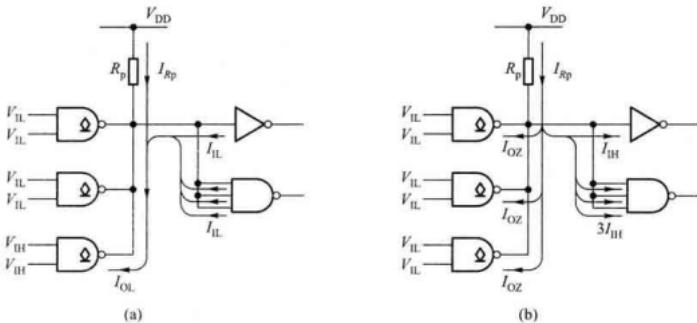


图 3.3.8 计算 OD 门上拉电阻 R_p 的工作情况

(a) $R_{p(\min)}$ 的工作情况 (b) $R_{p(\max)}$ 的工作情况

例 3.3.1 设 74HC03 中的 3 个漏极开路与非门作线与连接, 驱动 74HC04 中的 1 个反相器和 74HC10 中的 1 个三输入与非门, 电路如图 3.3.8 所示, 试确定一合适大小的上拉电阻 R_p 。已知 $V_{DD}=5\text{ V}$, OD 门输出低电平 $V_{OL(\max)}=0.33\text{ V}$ 时的输出电流 $I_{OL(\max)}=4\text{ mA}$, 输出高电平 $V_{OH(\min)}=4.4\text{ V}$ 时的漏电流 $I_{OZ}=5\text{ }\mu\text{A}$ 。负载门高电平和低电平输入电流最大值 $I_{IH(\max)}=I_{IL(\max)}=1\text{ }\mu\text{A}$ 。

解: (1) 当线与的 OD 门输出为低电平时, 式(3.3.1)中

$$I_{IL(\text{total})} = nI_{IL(\max)} = 4 \times 1\text{ }\mu\text{A} = 0.004\text{ mA}$$

$$\text{得 } R_{p(\min)} = \frac{V_{DD} - V_{OL(\max)}}{I_{OL(\max)} - I_{IL(\text{total})}} = \frac{5\text{ V} - 0.33\text{ V}}{4\text{ mA} - 0.004\text{ mA}} \approx 1.17\text{ k}\Omega$$

(2) 当线与的 OD 门输出为高电平时, 式(3.3.2)中

$$I_{OZ(\text{total})} = 3 \times 5\text{ }\mu\text{A} = 0.015\text{ mA}$$

$$I_{IH(\text{total})} = 4 \times 1\text{ }\mu\text{A} = 0.004\text{ mA}$$

$$\text{得 } R_{p(\max)} = \frac{V_{DD} - V_{OH(\min)}}{I_{OZ(\text{total})} + I_{IH(\text{total})}} = \frac{5\text{ V} - 4.4\text{ V}}{0.015\text{ mA} + 0.004\text{ mA}} \approx 31.58\text{ k}\Omega$$

根据上述计算, R_p 的值可在 $1.17\text{ k}\Omega$ 至 $31.58\text{ k}\Omega$ 之间选取。为使电路有较快的开关速度, 可选用一标准值为 $2\text{ k}\Omega$ 的电阻。

除了可以实现线与的逻辑功能外, OD 门也用来驱动发光二极管。发光二极管发光时需要的电流较大。图 3.3.9(a)所示为用 74HC05 中的 1 个漏极开路反相器驱动发光二极管。发光二极管发光时要求有几毫安的电流通过, 74HC/HCT 系列 CMOS 门电路的最大灌电流或拉电流为 4 mA 。当输入为高电平时, 输出为低电平, 此时发光二极管发光, 否则输出为高电平时二极管熄灭。若驱动指示灯($12\text{ V}, 20\text{ mA}$), 74HC/HCT 系列门电路不能满足要求, 可以选用 74AC05 或 74ACT05, 其灌电流为 24 mA 。

OD 门电路的另一个功能是实现逻辑电平变换,例如,可将 3.3 V 高电平转换为 5 V 高电平,如图 3.3.9(b) 所示。

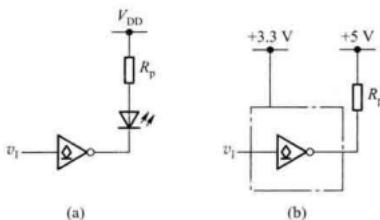


图 3.3.9 OD 门电路的应用
(a) 驱动发光二极管 (b) 逻辑电平变换

2. 三态输出门电路

利用 OD 门虽然可以实现线与的功能,但外接电阻 R_p 的选择要受到一定的限制,因此影响了工作速度。同时它省去了 PMOS 有源负载,使得带负载能力下降。为保持互补输出级的优点,又可以与总线连接,人们又开发了一种三态输出门电路,它的输出除了具有一般门电路的两种状态,即输出高、低电平外,还具有高输出阻抗的第三状态,称为高阻态,又称为禁止态。

图 3.3.10(a) 所示为高电平使能的三态输出缓冲电路,其中 A 是输入端, L 为输出端, EN (Enable) 是控制信号输入端,也称为使能端,图 3.3.10(b) 是它的逻辑符号。

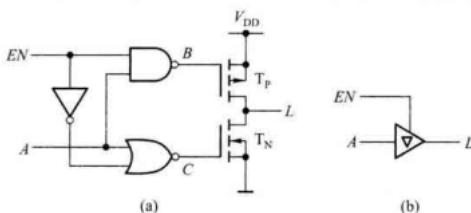


图 3.3.10 高电平使能三态输出门电路
(a) 电路结构 (b) 逻辑符号

当使能端 $EN=1$ 时,如果 $A=0$,则 $B=1, C=1$,使得 T_N 导通,同时 T_P 截止,输出端 $L=0$;如果 $A=1$,则 $B=0, C=0$,使得 T_N 截止, T_P 导通,输出端 $L=1$ 。

当使能端 $EN=0$ 时,不论 A 的取值为何,都使得 $B=0, C=0$,则 T_N 和 T_P 均截止,电路的输出端既不是低电平,又不是高电平,而是开路,这就是第三种高阻工作状态。

由以上分析可知,当 EN 为有效的高电平时,电路处于正常逻辑工作状态, $L=A$ 。而当 EN 为低电平时,电路处于高阻状态。三态输出门电路的真值表如表 3.3.1 所示,其中 \times 表示 A 可以是

0 或 **1**。

三态输出门电路主要用于总线传输,如计算机或微处理器系统,其连接形式如图 3.3.11 所示。任何时刻只有一个门电路的使能端 EN 为 **1**,该门电路的信号被传到总线上,而其他三态输出电路处于高阻状态。这样就可以按一定顺序将各个门电路的输出信号分时送到总线上。

表 3.3.1 三态输出门的真值表

使能 EN	输入 A	输出 L
1	0	0
1	1	1
0	\times	高阻

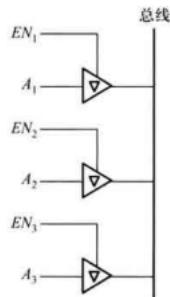


图 3.3.11 三态输出电路构成总线传输结构

在实际应用中,除上述介绍的三态输出电路外,还有其他不同形式的电路结构(见习题 3.3.7 所示)。使能端可以是高电平或低电平有效,输出与输入可以是同相或反相,其目的都是为用户提供一个合适的动态特性。例如,接到总线上的三态输出电路,在任何时刻只有一个使能端为有效信号,这就要求某个与总线进行数据传输的三态门必须关断以后,另一个三态门才允许与总线进行数据传输。即从高阻态到高电平(或低电平)输出的转换时间,略大于从高电平(或低电平)到高阻态的转换时间。这样,控制系统给出的使能信号,使前一个电路进入高阻状态以后,后一个电路的输出信号才送到总线上,以避免两个不同的信号在总线上引起冲突。

3.3.3 CMOS 逻辑门电路的重要技术参数

生产逻辑门电路的厂家,通常都会为用户提供逻辑器件的数据手册。对于不同系列的 CMOS 电路,只要型号最后的数字相同,它们的逻辑功能就一样,但是电气性能参数有所不同。手册中一般都要给出门电路的电压传输特性 $v_i - v_o$ 、输入和输出的高、低电压,噪声容限,传输延迟时间,功耗等。除传输特性外,其他各项技术参数分别介绍如下:

1. 输入和输出的高、低电平

前已讨论,数字电路中的高、低电压常用高、低电平来描述,并规定在正逻辑体制中,用逻辑 **1** 和 **0** 分别表示高、低电平。当逻辑电路的输入信号在一定范围内变化时,输出电压并不会改变,因此逻辑 **1** 或 **0** 对应一定的电压范围。不同系列的集成电路,输入和输出为逻辑 **1** 或 **0** 所对应的电压范围也不同。生产厂家的数据手册中一般都给出四种逻辑电平参数:输入低电平的上限值 $V_{IL(max)}$ 、输入高电平的下限值 $V_{IH(min)}$ 、输出低电平的上限值 $V_{OL(max)}$ 和输出高电平的下限

值 $V_{OH(min)}$ 。

表 3.3.2 列出了几种 CMOS 集成电路在典型工作电压时的输入高、低电压值以及在规定输出电流 I_o 条件下的输出电压值。这里以带 CMOS 负载的 74××04 非门参数为例。4000、74HC 和 74HCT 系列工作电压为 5 V, 低电压 74LVC 系列典型工作电压为 3.3 V, 超低电压 74AUC 系列典型工作电压为 1.8 V。

表 3.3.2 几种 CMOS 系列非门的输入和输出电压值及输入噪声容限

参数/单位 \ 类型	4000 $(V_{DD} = 5 \text{ V})$ $(I_o = 1 \text{ mA})$	74HC $(V_{DD} = 5 \text{ V})$ $(I_o = 0.02 \text{ mA})$	74HCT $(V_{DD} = 5 \text{ V})$ $(I_o = 0.02 \text{ mA})$	74LVC $(V_{DD} = 3.3 \text{ V})$ $(I_o = 0.1 \text{ mA})$	74AUC $(V_{DD} = 1.8 \text{ V})$ $(I_o = 0.1 \text{ mA})$
$V_{IL(max)} / \text{V}$	1.0	1.5	0.8	0.8	0.6
$V_{OL(max)} / \text{V}$	0.05	0.1	0.1	0.2	0.2
$V_{IH(min)} / \text{V}$	4.0	3.5	2.0	2.0	1.2
$V_{OH(min)} / \text{V}$	4.95	4.9	4.9	3.1	1.7
高电平噪声容限 (V_{NH} / V)	0.95	1.4	2.9	1.1	0.5
低电平噪声容限 (V_{NL} / V)	0.95	1.4	0.7	0.6	0.4

2. 噪声容限

噪声容限表示门电路的抗干扰能力。二值数字逻辑电路的优点在于它的输入信号允许一定的容差。在数字系统中, 各逻辑电路之间的连线可能会受到各种噪声的干扰, 如信号传输引起的噪声、信号的高低电平转换引起的噪声, 或者邻近开关信号所引起的随机脉冲的噪声。这些噪声会叠加在工作信号上, 只要其幅度不超过逻辑电平允许的最小值或最大值, 则输出逻辑状态不会受影响。通常将这个最大噪声幅度称为噪声容限。电路的噪声容限越大, 其抗干扰能力越强。

图 3.3.12 所示为噪声容限定义的示意图。前一级驱动门电路的输出, 就是后一级负载门电路的输入。则输入高电平的噪声容限

$$V_{NH} = V_{OH(min)} - V_{IH(min)} \quad (3.3.3)$$

V_{NH} 反映了驱动门输出高电平时, 容许叠加在其上的负向噪声电压的最大值。类似地, 输入低电平的噪声容限

$$V_{NL} = V_{IL(max)} - V_{OL(max)} \quad (3.3.4)$$

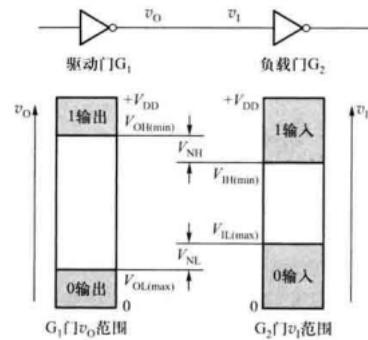


图 3.3.12 输入噪声容限示意图

V_{NL} 反映了驱动门输出低电平时,容许叠加在其上的正向噪声电压的最大值。根据 74HC 系列 CMOS 集成电路在 5 V 典型工作电压时的参数 ($I_o = 0.02 \text{ mA}$), 求得其输入高、低电平的噪声容限分别为:

$$\text{高电平的噪声容限 } V_{NH} = V_{OH(\min)} - V_{IH(\min)} = 4.9 \text{ V} - 3.5 \text{ V} = 1.4 \text{ V}$$

$$\text{低电平的噪声容限 } V_{NL} = V_{IL(\max)} - V_{OL(\max)} = 1.5 \text{ V} - 0.1 \text{ V} = 1.4 \text{ V}$$

其他 CMOS 系列的高、低电平的噪声容限列于表 3.3.2。

3. 传输延迟时间

传输延迟时间是表征门电路开关速度的参数, 它说明门电路在输入脉冲波形的作用下, 其输出波形相对于输入波形延迟了多长时间, 其数值与电源电压 V_{DD} 及负载电容的大小有关。

当非门电路的输入端加入一脉冲波形, 其相应的输出波形如图 3.3.13 所示。通常输出波形下降沿、上升沿的中点与输入波形对应沿中点之间的时间间隔, 分别用 t_{PLH} 和 t_{PHL} 表示, 由于 CMOS 门电路输出级的互补对称性, 其 t_{PLH} 和 t_{PHL} 相等。

有时也采用平均传输延迟时间这一参数, 即 $t_{pd} = (t_{PLH} + t_{PHL}) / 2$ 。例如, CMOS 与非门 74HC00 在 5 V 典型工作电压时 $t_{PLH} = t_{PHL} = 7 \text{ ns}$, $t_{pd} = (7+7) \text{ ns}/2 = 7 \text{ ns}$ 。在图 3.3.13 中还标出了上升时间 t_r 和下降时间 t_f 。

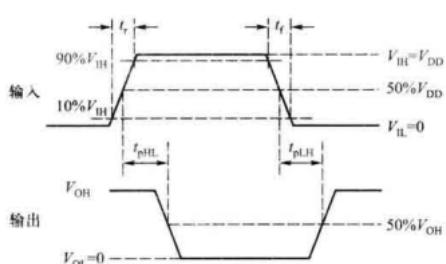


图 3.3.13 门电路传输延迟波形图

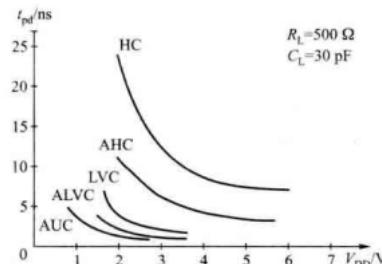


图 3.3.14 几种 CMOS 系列传输延迟时间与电源电压关系

图 3.3.14 所示为几种 CMOS 集成电路的传输延迟时间 t_{pd} 与电源电压 V_{DD} 的关系曲线。由图可见, 当电源电压增加时, 传输延迟时间减少, 可提高工作速度。74AHC 系列的速度达到了 74HC 系列的两倍, 而低电压 74LVC、74ALVC 和超低电压 74AUC 系列的电源电压更低, 传输延迟时间更短, 工作速度更快。

4. 功耗

功耗是门电路重要参数之一。功耗有静态和动态之分。所谓静态功耗是指电路输出没有状态转换时的功耗。静态时, CMOS 电路的电流非常小, 使得静态功耗非常低, 所以 CMOS 电路广

泛应用于要求功耗较低或电池供电的设备,如便携计算机、手机和掌上电脑等。这些设备在没有输入信号时,功耗非常低。

CMOS 电路在输出发生状态转换时的功耗称为动态功耗。它主要由两部分组成。一部分是电路输出状态转换瞬间 MOS 管的导通功耗。由图 3.2.13 所示的 CMOS 反相器电压和电流传输特性可知,当输出电压由高到低或由低到高变化过程中,在短时间内,NMOS 管和 PMOS 管均导通,从而导致有较大的电流从电源 V_{DD} 经导通的 NMOS 管和 PMOS 管流入地。这部分功耗可由下式表示:

$$P_T = C_{PD} V_{DD}^2 f$$

式中 f 为输出信号的转换频率; V_{DD} 为供电电源; C_{PD} 称为功耗电容 (Power Dissipation Capacitance),与电源电压和工作频率有关,可以在数据手册中查到。74HC04 非门的 C_{PD} 为 21 pF ($V_{DD} = 5$ V), 74LVC04 为 8 pF ($V_{DD} = 3.3$ V)。

动态功耗的另一部分是因为 CMOS 管的负载通常是电容性的,当输出由高电平到低电平,或者由低电平到高电平转换时,会对电容进行充、放电,这一过程将增加电路的损耗。这部分动态功耗为

$$P_L = C_L V_{DD}^2 f$$

式中 C_L 为负载电容。由此得到 CMOS 电路总的动态功耗为

$$P_D = (C_{PD} + C_L) V_{DD}^2 f \quad (3.3.5)$$

从式(3.3.5)可见,CMOS 动态功耗正比于转换频率和电源电压的平方。当工作频率增加时,CMOS 门的动态功耗会线性增加。一个典型的 CMOS 门电路的静态功耗为 0.01 mW。当工作频率达到 1 MHz 时,功耗增加到 0.5 mW。当频率为 10 MHz 时,功耗为 5 mW。可见,其功耗主要取决于动态功耗。在设计 CMOS 电路时,为降低功耗,可选用低电源电压器件,如 3.3 V 的 74LVC 系列、1.8 V 的 74AUC 系列或超低功耗 74AUP 系列。

5. 延时-功耗积

从上述对传输延迟时间和功耗的讨论可知,若增加电源电压,电路的工作速度变快,但功耗也随之增加。理想的数字电路或系统,既要速度高,又要功耗低。在工程实践中要实现这种理想情况是较难的。高速数字电路往往要以较大的功耗为代价。一种衡量这种性能的综合性指标叫做延时-功耗积,用符号 DP 表示,单位为 J(焦[耳]),即

$$DP = t_{pd} P_D$$

式中 $t_{pd} = (t_{PLH} + t_{PHL}) / 2$; P_D 为门电路的功耗,由式(3.3.5)进行计算。一个逻辑门器件的 DP 值越小,表明它的特性越接近理想情况。表 3.3.3 所示为不同系列 CMOS 非门在典型工作电压下,负载电容为 15 pF,工作频率为 10 MHz 时的 DP 值,其中功耗电容 C_{PD} 和传输延迟时间 t_{pd} 可在数据手册中查到。由此可见,74AHC 系列的性能优于 74HC 系列,而 74LVC 系列和 74AUC 系列比前两者性能更好。

表 3.3.3 几种 CMOS 系列非门的 DP 性能比较

系列 参数/单位	74HC04 ($V_{DD} = 5$ V)	74AHC04 ($V_{DD} = 5$ V)	74LVC04 ($V_{DD} = 3.3$ V)	74AUC04 ($V_{DD} = 1.8$ V)
功耗电容 C_{PD}/pF	21	12	8	17
传输延迟时间 t_{PD}/ns ($C_L = 15 \text{ pF}$)	6	3.8	2.5	0.8
功耗 P_0/mW (10 MHz)	9	6.8	2.5	1
延时功耗积 DP/pJ	54	25.84	6.25	0.8

6. 扇入与扇出数

门电路的扇入数取决于它的输入端的个数,例如一个 3 输入端的与非门,其扇入数 $N_I = 3$ 。

门电路的扇出数是指其在正常工作情况下、所能带同类门电路的最大数目。扇出数的计算则稍复杂些,要考虑两种情况,一种是拉电流负载,另一种是灌电流负载。

(1) 拉电流工作情况

图 3.3.15(a)所示为拉电流负载的情况。当驱动门的输出端为高电平时,将有电流 I_{on} 从驱动门拉出而流入负载门,负载门的输入电流为 I_{in} 。当负载门的个数增加时,总的拉电流将增加,会引起输出高电平的降低。但不得低于输出高电平的下限值,这就限制了负载门的个数。这样,输出为高电平时的扇出数可表示为

$$N_{on} = \frac{I_{on}(\text{驱动门})}{I_{in}(\text{负载门})} \quad (3.3.6)$$

(2) 灌电流工作情况

图 3.3.15(b)所示为灌电流负载的情况。当驱动门的输出端为低电平时,电流 I_{ol} 流入驱动门,它是负载门输入端电流 I_{il} 之和。当负载门的个数增加时,总的灌电流 I_{ol} 将增加,同时也将引起输出低电平 V_{ol} 的升高。在保证不超过输出低电平的上限值时,驱动门所能驱动同类门的个数由下式决定:

$$N_{ol} = \frac{I_{ol}(\text{驱动门})}{I_{il}(\text{负载门})} \quad (3.3.7)$$

以上 N_{on} 和 N_{ol} 的计算公式没有考虑电容性负载。

一般逻辑器件的数据手册中,并不给出扇出数,而需计算或用实验的方法求得,并注意在设计时留有余地,以保证数字电路或系统能正常地运行。在实际的工程设计中,如果 $N_{ol} \neq N_{on}$,则取二者中的最小值。

为便于比较,表 3.3.4 给出了几种 CMOS 系列 2 输入与非门的输入和输出电压及电流参数的典型值。大多数 CMOS 器件有两类负载规格参数。一类对应于“CMOS 负载”,即输出端与其他 CMOS 门相连,此时输出端只有很小的电流,因此,高电平输出值接近 V_{DD} ,而低电平输出值接近 0 V。另一类对应“TTL 负载”,即输出端与电阻性负载(TTL 输入或其他器件)相连,此时输出

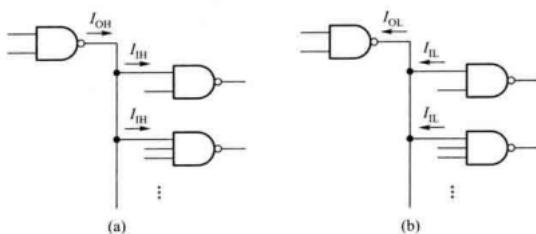


图 3.3.15 扇出数的计算

(a) 拉电流负载 (b) 灌电流负载

端有较大的电流。

表 3.3.4 几种 CMOS 系列 2 输入与非门的电压与电流参数

系列 参数/单位		74HC00	74HCT00	74AHC00	74AHCT00	74LVC00	74ALVC00	74AUC00
$I_{OH(max)}$ /μA		1	1	1	1	5	5	5
$I_{OL(max)}$ /μA		-1	-1	-1	-1	-5	-5	-5
$I_{OH(max)}$ /mA	CMOS 负载	-0.02	-0.02	-0.05	-0.05	-0.1	-0.1	-0.1
	TTL 负载	-4	-4	-8	-8	-24	-24	-9
$I_{OL(max)}$ /mA	CMOS 负载	0.02	0.02	0.05	0.05	0.1	0.1	0.1
	TTL 负载	4	4	8	8	24	24	9
$V_{OH(max)}$ /V		3.5	2	3.5	2	2	2	1.7
$V_{OL(max)}$ /V		1.5	0.8	1.5	0.8	0.8	0.8	0.7
$V_{OH(min)}$ /V	CMOS 负载	4.9	4.9	4.9	4.9	2.8	2.8	2.2
	TTL 负载	4.4	4.4	4.4	4.4	2.2	2	1.8
$V_{OL(max)}$ /V	CMOS 负载	0.1	0.1	0.1	0.1	0.2	0.2	0.2
	TTL 负载	0.33	0.33	0.44	0.44	0.55	0.55	0.6

注：表中的参数参考 Texas Instruments 公司在互联网站上提供的集成电路产品数据，以及文献[1]。74HC/HCT 和 74AHC/AHCT 是 $V_{DD}=5$ V 时的参数，74LVC/ALVC 是 $V_{DD}=3$ V 时的参数，74AUC 是 $V_{DD}=2.7$ V 时的参数。更详细的参数，可查阅有关器件的数据手册。

例如，根据表 3.3.4 可知 HC 系列 CMOS 门电路参数：当高电平输出为 4.9 V 时，输出电流 $I_{OH}=-20 \mu\text{A}$ ；低电平输出 0.1 V 时， $I_{OL}=20 \mu\text{A}$ ，输入电流 $I_{IH}=1 \mu\text{A}$ ， $I_{IL}=-1 \mu\text{A}$ 。数据前的负号表示电流从器件流出，否则电流流入器件，计算时只取绝对值。所以根据式 (3.3.6) 和式

(3.3.7) 计算出 $N_{\text{on}} = N_{\text{ot}} = 20 \mu\text{A}/1 \mu\text{A} = 20$, 即最多可接同类电路的输入端数为 20 个。

如果允许其高电平输出降至 4.4 V, 并且低电平为 0.33 V, 则 I_{on} 和 I_{ot} 分别为 -4 mA 和 4 mA, 此时计算出的扇出数为 4 000, 实际不可能达到这么大的数。

需要注意的是, 上述讨论的是 CMOS 电路静态扇出数, 即直流扇出数。当门电路的工作频率比较低时, 可以采用直流扇出数计算带负载能力。

如果考虑电容性负载, 假设每个负载门输入端的电容均为 15 pF, 20 个负载门的等效电容为 300 pF, 而 4 000 个负载门的等效电容为 60 000 pF。当驱动门的负载电容比较大时, 电容的充放电电流不能忽略。

当 CMOS 门驱动同类门, 并且输出状态产生由高到低或由低到高变化时, 由图 3.2.13(b) 所示的 CMOS 反相器电流传输特性可知, 负载门的电流很大。驱动门的输出电流会对负载门输入端的杂散电容进行充放电。驱动门为高电平时, 会向负载门的输入电容充电; 而驱动门为低电平时, 充电的电容会通过驱动门输出电阻放电。此时驱动门对负载门输入端杂散电容的充放电能力可以定义为交流扇出数。

交流扇出数很难像直流扇出数那样简单地计算出来。它不仅与负载电容的大小有关, 还与门电路的工作频率有关。增加负载门数量将导致总电容值的增加, 致使充放电时间增加, 并使输出电压的上升时间和下降时间大于规定的数值, 从而影响门电路的开关速度。因此, 工作频率增加时, 为保证电路正常工作, 应减少扇出数。

如果驱动门的负载大于其扇出能力, 当输出低电平时, 低电平的数值高于 $V_{\text{OL(max)}}$; 当输出高电平时, 高电平的值低于 $V_{\text{OH(min)}}$ 。电路不能正常工作。

这里考虑每个负载门只有一个输入端与驱动门相接, 如果每个负载门有两个以上的输入端接入驱动门, 则扇出数实为输入端数目。

另外, 不同门电路的参数可能与表 3.3.4 给出的典型值不同, 因此当分析实际问题时, 必须查阅生产厂商提供的数据表。

CMOS 器件有许多不同系列产品, 各系列产品的参数也有很多。对于设计者, 比较重要的参数是速度和功耗。

复习思考题

-
- 3.3.1 为什么要在 CMOS 逻辑门电路输入、输出端加保护和缓冲电路?
 - 3.3.2 CMOS 逻辑门的输入端分别通过 1 kΩ 和 100 kΩ 的电阻接地时, 其输入电压都是多少?
 - 3.3.3 普通的两个 CMOS 逻辑门电路的输出端能否连接在一起, 为什么?
 - 3.3.4 漏极开路门和三态门的特点是什么, 它们各用在什么场合?
 - 3.3.5 如何确定漏极开路门的上拉电阻? 它对开关速度有无影响?
 - 3.3.6 当增加电源电压时, CMOS 电路的传输延迟时间和功耗如何变化? 当增加工作频率时, 功耗如何变化?
 - 3.3.7 CMOS 电路的功耗电容 C_{pp} 的物理意义是什么?

3.4 类 NMOS 和 BiCMOS 逻辑门电路

3.4.1 类 NMOS 门电路

MOS 数字集成电路的发展经历了由 PMOS、NMOS 到 CMOS 的过程，其中 PMOS 电路问世最早。PMOS 管是以空穴为导电载流子，而 NMOS 管以电子为导电载流子，由于空穴的迁移率比电子低，因此，NMOS 电路的工作速度比 PMOS 电路快，而且 PMOS 使用负电源，与 TTL 电路不匹配，所以 PMOS 电路被 NMOS 电路取代。

NMOS 电路的工作速度快，几何尺寸小，而且生产工艺水平也不断提高和完善，所以某些特殊应用中采用 NMOS 电路更好。后来发展的 CMOS 电路有静态功耗低、抗干扰能力强等诸多优点而成为主流器件。但是 CMOS 门电路每增加一个输入端就要增加一个 NMOS 管和一个 PMOS 管，而且空穴的迁移率比电子的迁移率低，为获得同样的导通电阻和电流，PMOS 管所需的芯片面积更大。为减少电路中 PMOS 管的数目，在对性能要求不太高，并且希望芯片面积尽可能小的情况下，仍然采用 NMOS 电路。

NMOS 逻辑门电路全部由 N 沟道 MOS 管构成。NMOS 反相器是 NMOS 逻辑门电路的基本电路形式，它的工作管为增强型 MOS 管，而负载管可以是增强型也可以是耗尽型 MOS 管。早期的 NMOS 电路采用增强型负载管，如图 3.4.1(a) 所示。由于增强型负载的 NMOS 电路开关速度比耗尽型的慢，在后来的 NMOS 电路中，采用耗尽型负载管，如图 3.4.1(b) 所示。生产耗尽型负载管需要增加一道离子注入工艺，并且衬底效应使其 $I-V$ 特性偏离了恒流特性 ($v_{GS}=0$ ，耗尽型 MOS 管为恒流源负载) 使电路性能不够理想。但某些特殊应用中仍然使用耗尽型负载管的 NMOS 电路。类 NMOS (Pseudo NMOS) 电路采用 PMOS 增强型负载管，栅极接地使负载管处于常通状态。相对耗尽型负载的 NMOS 电路，类 NMOS 电路有许多性能上的改进，噪声容限变大。类 NMOS 电路的另一个优点是与 CMOS 电路相匹配。

1. 类 NMOS 反相器

类 NMOS 反相器电路结构如图 3.4.2 所示，其中工作管为 NMOS 管，负载管为 PMOS 管。PMOS 管的栅极接地，所以始终处于导通状态。

当输入 $v_i=0$ 时，NMOS 管截止，PMOS 管处于可变电阻区， $v_{DS2} \approx 0$ ，输出 $v_o \approx +V_{DD}$ ，此时电路的电流几乎为 0，静态功耗为 0。

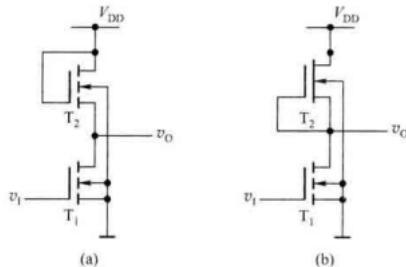


图 3.4.1 NMOS 反相器
(a) 增强型负载 (b) 耗尽型负载

当输入 $v_1 = +V_{DD}$ 时, NMOS 管和 PMOS 管都导通, NMOS 管工作在可变电阻区, PMOS 管工作在饱和区, 输出 v_0 应为低电平, 其电压由 NMOS 管和 PMOS 管的导通电阻值分压比决定。通常在制造工艺上使 NMOS 管的导通电阻远小于 PMOS 管的导通电阻, 使输出为低电平。此时的静态功耗不为 0。因此, 类 NMOS 反相器多用于输出通常为高电平的电路。

2. 类 NMOS 与非门和或非门

图 3.4.3 为类 NMOS 与非门电路, 其中 T_{N1} 和 T_{N2} 为工作管, 两者串联; T_P 为负载管。当输入 A, B 中任一个为低电平时, 与它对应的 NMOS 管截止, 输出为高电平; 仅当 A, B 全为高电平时, 所有工作管都导通, 输出才为低电平。可见电路具有与非功能, 即

$$L = \overline{A \cdot B}$$

图 3.4.4 为类 NMOS 或非门电路, 其中工作管 T_{N1} 和 T_{N2} 并联, T_P 为负载管。当输入 A, B 中任一个为高电平时, 与它对应的 NMOS 管导通, 输出为低电平; 仅当 A, B 全为低电平时, 所有工作管都截止, 输出才为高电平。电路具有或非功能, 即

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

值得注意的是, 当增加与非门输入端数目时, 串联的管子也随之增加。当输入全为高电平时, 各管的导通电阻串联, 使低电平输出电压升高, 以致破坏正常逻辑功能。而或非门的工作管是并联的, 增加 NMOS 管的数目不会影响低电平输出电压的稳定, 因而类 NMOS 电路多以或非门为基础, 构成各种功能的逻辑电路。

与 CMOS 电路相比, 类 NMOS 电路减少了 PMOS 管数目, 输入端数目越多, 减少的 PMOS 管数目也越多。

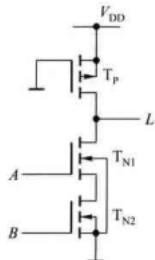


图 3.4.3 类 NMOS 与非门电路

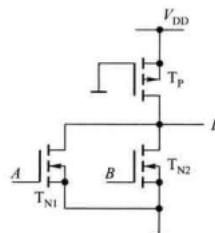


图 3.4.4 类 NMOS 或非门电路

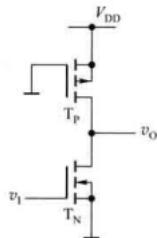


图 3.4.2 类 NMOS 反相器

3.4.2 BiCMOS 门电路

BiCMOS 电路的特点在于采用了双极型 BJT 管作为 CMOS 电路的输出级。因此这种电路结合了 MOS 管高集成度、低功耗和双极型管速度快、驱动力强的优势，受到用户的重视。

基本的 BiCMOS 反相器电路结构如图 3.4.5 所示。其中，MOS 管用符号 M 表示，BJT 用 T 表示。 M_p 、 M_N 、 M_1 和 M_2 组成输入级， T_1 和 T_2 构成推拉式输出级。输入信号 v_i 同时作用于 M_p 和 M_N 的栅极。输入信号 v_i 为高电平时， M_N 、 M_1 和 T_1 导通， M_p 、 M_2 和 T_1 截止，输出 v_o 为低电平；而当 v_i 为低电平时，情况则相反， M_p 、 M_2 和 T_2 导通， M_N 、 M_1 和 T_2 截止，输出 v_o 为高电平。当输出端接有比较大的容性负载时，双极型三极管输出级能为其提供足够大的充电电流。同理，已充电的电容负载也能迅速地通过 T_2 放电。

电路中 M_1 和 M_2 的作用是加快 T_1 和 T_2 由饱和导通翻转到截止的过程，使 T_1 和 T_2 的基区存储电荷通过 M_1 和 M_2 释放。当 v_i 为高电平时， M_1 导通， T_1 基区的存储电荷迅速消散。同理，当 v_i 为低电平时，电源电压 V_{DD} 通过 M_p 以激励 M_2 ，使 M_2 导通，显然， T_2 基区的存储电荷通过 M_2 而消散。从而，门电路的开关速度可得到改善。

根据前述的 CMOS 门电路的结构和工作原理，同样可以用 BiCMOS 技术实现与非门或或非门，具体电路见题 3.4.3 和题 3.4.4。

复习思考题

3.4.1 为什么类 NMOS 门电路多以或非门作为基本门电路？

3.4.2 为什么 BiCMOS 电路的开关速度比较快？

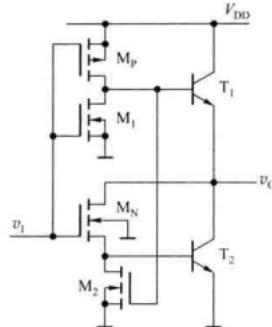


图 3.4.5 基本的 BiCMOS 反相器电路

3.5 TTL 逻辑门电路

3.5.1 BJT 的开关特性

1. BJT 的开关作用

由 BJT 构成的开关电路如图 3.5.1(a) 所示，图中 BJT 为 NPN 型硅管。当输入为低电平时，例如 $v_i = V_{IL} = 0 \text{ V}$ 时，BJT 的发射结为零偏 ($v_{BE} = 0$)，集电结为反向偏置 ($v_{BC} < 0$)，只有很小的漏电流流过 PN 结，故 $i_B = 0$, $i_C \approx 0$, $v_o = V_{CE} \approx V_{CC}$ ，对应于图 3.5.1(b) 中的 A 点。这时集电极回路中

的 c、e 极之间近似于开路，相当于开关断开一样，输出为高电平。BJT 的这种工作状态称为截止^①。

当输入为高电平时，例如 $v_i = V_{in} = V_{cc}$ 时，调节 R_b ，使 $i_B = V_{cc}/(\beta R_e)$ ，则 BJT 工作在图 3.5.1 (b) 中的 B 点，集电极电流 i_c 已接近最大值 V_{cc}/R_e ，称为集电极饱和电流 $I_{cs} \approx V_{cc}/R_e$ ，对应的基极电流称为基极临界饱和电流 $I_{bs} \approx V_{cc}/(\beta R_e)$ 。如果再增加 i_B ，则饱和程度加深，但 i_c 基本上保持在 I_{cs} 不再增加，集电极、发射极之间的电压 v_{ce} 约为 0.2~0.3 V，该电压称为 BJT 的饱和压降 V_{ces} ，它也基本上不随 i_B 增加而改变。由于 V_{ces} 很小，集电极回路中的 c、e 极之间近似于短路，相当于开关闭合一样，输出为低电平。

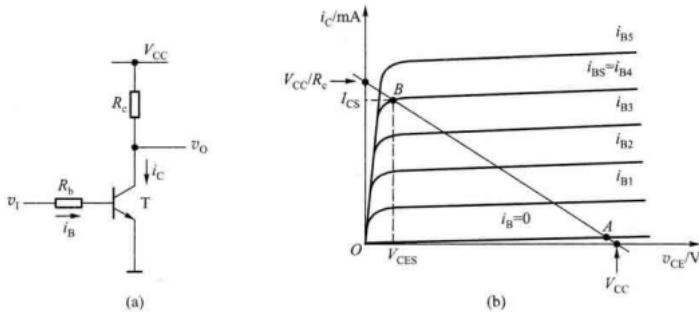


图 3.5.1 BJT 管开关电路及输出特性

(a) BJT 管开关电路 (b) BJT 管输出特性

2. BJT 的开关特性

BJT 的开关过程是其在饱和与截止两种状态之间的相互转换，也是内部电荷“建立”和“消散”的过程。因此，需要一定的时间才能完成。

当图 3.5.1(a) 所示开关电路的输入端加入一个数字脉冲信号，输入信号的上升沿使 BJT 从截止到饱和变化，需要时间建立基区电荷以形成饱和电流。输入信号的下降沿使 BJT 从饱和到截止变化，需要时间消散基区存储的电荷。因此，输出电压 v_o 的变化滞后于输入电压 v_i 的变化。

3.5.2 TTL 反相器的基本电路

由于基本 BJT 反相器的动态性能不理想，为改善其动态性能，增加若干元器件构成 TTL 反相器的基本电路，如图 3.5.2 所示。该电路由三部分组成， T_1 组成电路的输入级， T_3 、 T_4 和二极管 D 组成输出级，以及由 T_2 组成的中间级作为输出级的驱动电路，将 T_2 的单端输入信号 v_{12} 转换为互补的双端输出信号 v_{13} 和 v_{14} ，以驱动 T_3 和 T_4 。

^① 当 v_{BE} 小于开启电压时，管子即已进入截止区。

当输入 $v_i = V_{IL} = 0.2 \text{ V}$ 时, T_1 的发射结导通, 其基极电压为 $v_{B1} = V_{IL} + V_{BE1} = 0.9 \text{ V}$, 该电压作用于 T_1 的集电结和 T_2 、 T_3 的发射结上, 所以 T_2 、 T_3 都截止, 而 T_4 和 D 导通, 输出为高电平。由于 T_2 管截止, R_{e2} 上的压降忽略不计, 则 $v_o = V_{OH} = V_{CC} - V_{BE4} - V_D = 3.6 \text{ V}$ 。

当输入 $v_i = V_{IH} = 3.6 \text{ V}$ 时, V_{CC} 通过 R_{BL} 和 T_1 的集电结向 T_2 、 T_3 提供基极电流, 使 T_2 、 T_3 饱和导通, 此时 $v_{B1} = V_{BC1} + V_{BE2} + V_{BE3} = 2.1 \text{ V}$, 使 T_1 的发射结反向偏置, 而集电结正向偏置。所以 T_1 处于发射结和集电结倒置的放大状态。由于 T_2 和 T_3 饱和, 使 $v_{C2} = V_{CES2} + V_{BE3} = 0.9 \text{ V}$ 。该电压作用于 T_4 的发射结和二极管 D 两个 PN 结上, 显然 T_4 和 D 均截止。 T_4 和 D 截止, 且 T_3 饱和导通, 使输出为低电平, $v_o = v_{C3} = V_{CES3} = 0.2 \text{ V}$ 。

上述电路实现了反相器的逻辑关系。

输入级的作用是用来提高工作速度。当电路的输入电压由高到低变化时, T_1 由倒置的放大状态转换为放大状态, 使 T_2 的基极电流增加, 加快抽走多余的存储电荷而达到截止。 T_2 的迅速截止, 一方面使 T_4 的导通加快, 另一方面使 T_3 的截止加快, 从而加快了状态转换。

采用推拉式输出级^①以提高开关速度和带负载能力。输出级的两个管子总是一个导通而另一个截止, 因此降低了静态功耗。当输出为低电平时, T_4 截止, T_3 饱和导通, 其饱和电流全部用来驱动负载。当输出为高电平时, T_3 截止, 由 T_4 组成电压跟随器的输出电阻很小, 因此带负载能力也较强。当输出端接有电容性负载时, T_3 或 T_4 饱和导通电阻很低, 对电容充、放电时间常数很小, 使输出电压波形的上升沿和下降沿都很好。

3.5.3 改进型 TTL 门电路——抗饱和 TTL 门电路

上述 TTL 反相器中 BJT 导通时工作在深度饱和状态是产生传输延迟时间的主要原因, 为此推出了许多改进电路。改进电路的目的是提高工作速度和降低功耗, 但两者对电路的要求是矛盾的, 常用延时功耗积 DP 进行综合评价。改进效果比较明显的是抗饱和 TTL 电路。这种电路采用肖特基势垒二极管 (Schottky-Barrier-Diode, SBD) 的钳位作用来限制 BJT 导通时的饱和深度。

最早的肖特基 (74S) 系列使用肖特基三极管和二极管, 将电路的 DP 值降到原来的 $1/2$ 。低功耗肖特基 (74LS) 系列的 DP 值为 74S 系列的 $1/3$ 。后来对集成电路工艺进一步改进, 生产出改进的 74AS 和 74ALS 系列。74AS 系列的速度是 74S 系列的 2 倍。74ALS 系列又进一步改善了速度和功耗。快速 TTL 系列 (74F) 使用新工艺减小了器件尺寸和结电容, 进一步降低了传输延迟时间。

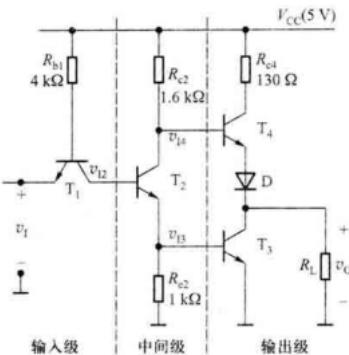


图 3.5.2 TTL 反相器的基本电路

^① 又称为带有源上拉 (active pull-up) 电路或图腾 (totem pole) 输出级。

1. 肖特基势垒二极管工作原理

肖特基势垒二极管是一种利用金属和半导体相接触，在交界面形成势垒的二极管。利用金属铝和N型硅半导体相接触形成的势垒二极管的工作特点如下：

(1) 它和PN结一样，同样具有单向导电性，这种铝-硅势垒二极管(Al-SiSBD)导通电流的方向是从铝到硅。

(2) Al-SiSBD的导通阈值电压较低，约为0.4~0.5V，比普通硅PN结约低0.2~0.3V。

(3) 势垒二极管是多数载流子参与导电，没有少数载流子的积累，因而从正向导通到反向截止，没有内部电荷的建立和消散过程，使转换速度加快。

BJT工作在饱和时，发射结和集电结都处在正向偏置，集电结正向偏置电压越大，则表明饱和程度越深。为了限制BJT的饱和深度，在BJT的基极和集电极并联上一个导通阈值电压较低的肖特基二极管，如图3.5.3(a)所示。通常把它们看成一个器件，并用图3.5.3(b)所示符号表示。当BJT集电结的正向偏压达到SBD的导通阈值电压时，这个二极管首先导通，使集电结正向偏压钳制在0.4V左右，如果流向基极的电流增大，企图使集电结正向偏压加大时，则一部分电流就会通过肖特基二极管直接流向集电极，而不会使BJT基极电流过大，因此，肖特基二极管起到了抵抗BJT过饱和的作用，因而这种电路就称为抗饱和电路，它能使电路的开关时间大为缩短。

2. 低功耗肖特基TTL反相器

以低功耗肖特基TTL反相器74LS04为例，该芯片中有6个反相器，图3.5.4为其中的一个反相器，与基本的TTL反相器电路相比，作了若干改进。首先是除 T_5 外，其余的BJT均采用SBD钳位，以达到明显的抗饱和效果。其次输入级用SBD代替BJT管，由于SBD没有电荷存储效应，有利于提高工作速度。

肖特基TTL门电路还从以下三点对基本TTL电路(图3.2.7)的性能作了改进：

(1) 基本TTL电路中的二极管D和 T_4 由 T_4 和 T_5 所组成的复合管所代替，当输出由低电平向高电平过渡时，由于复合管电路的电流增益很大，输出电阻很小，从而减小了电路对负载电容的充电时间。

(2) 电路输入端所加的SBD D_2 ，用来减小门电路之间的连线而引起的杂散信号，并防止输入信号反向过冲使 T_1 电流过大而损坏。

(3) 由 T_2 与 R_4 、 R_5 组成的有源电路代替了基本TTL电路中的 R_{o2} (1kΩ)。当反相器输入端由低电平转向高电平时， T_1 由截止变为导通，由于 T_2 的基极回路串接了电阻 R_4 ， T_2 的导通滞后于 T_1 ，使 T_1 以较大的电流驱动 T_3 ，从而加快了 T_3 的导通过程。随后， T_2 开始导通，将对 T_3 的基极电流产生分流作用，减轻了 T_3 饱和程度，当电路再次翻转时， T_3 能很快地截止。因而，有源电阻缩短了门电路的转换时间，使其电压传输特性得到改善。

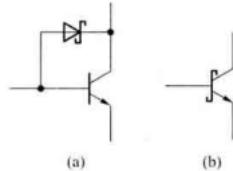


图3.5.3 带有肖特基二极管钳位的BJT
(a) 电路结构形式 (b) 符号

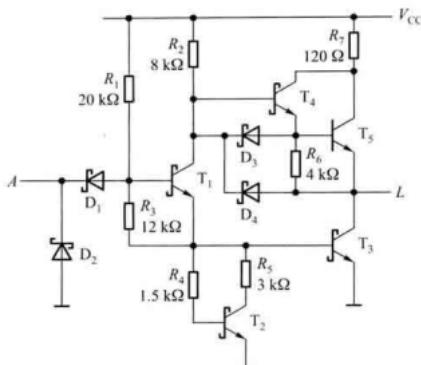


图 3.5.4 74LS04 中的一个反相器电路

3. 其他 TTL 门电路

TTL 系列逻辑门电路中,除上述介绍的反相器外,还有与非门、或非门等。74LS 系列 2 输入与非门 74LS00 和 2 输入或非门 74LS02 电路分别如图 3.5.5 和图 3.5.6 所示。

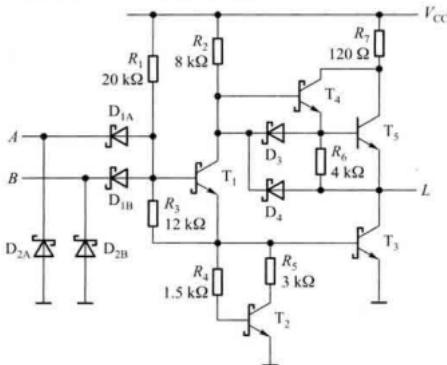


图 3.5.5 74LS00 中的一个与非门电路

与 CMOS 逻辑门电路类似,TTL 门电路也有另外两种不同输出结构形式的电路,集电极开路(Open Collector,OC)门和三态输出门。OC 门是将 TTL 门电路输出级的集电极开路,只有外接上拉电阻电路才能正常工作。外接电阻的计算与 OD 门类似。TTL 三态门也是在普通门电路的基础上,增加控制电路构成的,电路的输出状态有高电平、低电平和高阻三种状态。OC 门和三态门的 TTL 具体电路这里不再赘述。

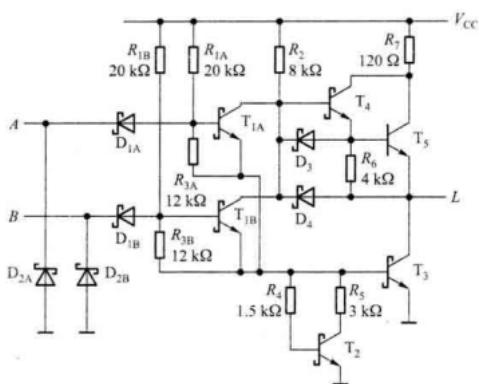


图 3.5.6 74LS02 中的一个或非门电路

3.5.4 TTL 系列门电路特性参数比较

目前 TTL 系列的使用在逐渐减少。TTL 系列电路的供电电源都是 +5 V，并且都是兼容的。但每个系列的速度、功耗等参数各有特点。表 3.5.1 列出了几种 TTL 系列 2 输入与非门电路输入和输出的高、低电平及电流，传输延迟时间 t_{pd} 和功耗 P_p 等特性参数，以资比较。

表 3.5.1 几种 TTL 系列门电路的特性参数比较

参数 \ 系列	74S	74LS	74AS	74ALS	74F
$V_{IL(max)}/V$	0.8	0.8	0.8	0.8	0.8
$V_{OL(max)}/V$	0.5	0.5	0.5	0.5	0.5
$V_{IH(min)}/V$	2.0	2.0	2.0	2.0	2.0
$V_{OH(max)}/V$	2.7	2.7	2.7	2.7	2.7
$I_{IL(max)}/mA$	-2.0	-0.4	-0.5	-0.1	-0.6
$I_{OL(max)}/mA$	20	8	20	8	20
$I_{IH(max)}/\mu A$	50	20	20	20	20
$I_{OH(max)}/mA$	-1.0	-0.4	-2.0	-0.4	-1.0
传输延迟时间 t_{pd}/ns	3	9.5	1.7	4	3
每门功耗 P_p/mW	19	2	8	1.2	4
延时-功耗积 DP/pJ	57	19	13.6	4.8	12

复习思考题

- 3.5.1 在数字电路中, BJT 管作为开关使用时, 工作在其输出特性曲线的什么区?
- 3.5.2 基本 BJT 反相器的动态性能存在什么问题? 与 BJT 反相器相比, 为什么 TTL 反相器可以提高电路的开关速度?
- 3.5.3 抗饱和 TTL 电路为什么可以提高开关速度?

* 3.6 ECL 逻辑门电路

由于 TTL 门电路中 BJT 工作在饱和状态, 开关速度受到了限制。上一节介绍的肖特基 TTL 电路是一种抗饱和电路。还有一种结构方式完全不同的电路, 将 BJT 从饱和型变为非饱和型, 从根本上提高速度。射极耦合逻辑门 ECL 就是这种非饱和型高速数字集成电路, 它的平均传输延迟时间可低于 1 ns, 是目前双极型电路中速度最高的, 主要用于高速或超高速数字系统中。与其他系列逻辑门相比, 其输出电压在高电平和低电平之间的摆幅非常小, 可低于 0.8 V。

ECL 逻辑电路有 10K 和 100K 两种系列。ECL10K 系列的传输延迟时间为 2 ns, 每门功耗为 25 mW, 延时功耗积 DP 为 50 pJ。ECL100K 系列的传输延迟时间为 0.75 ns, 每门功耗为 40 mW, 延时功耗积 DP 为 30 pJ。

1. ECL 门电路的结构及工作原理

图 3.6.1 所示为 2 输入 ECL 或/或非门电路及逻辑符号。电路由三部分组成: 差分输入、基准电压及射极输出电路。通常 $V_{CC1} = V_{CC2} = 0$, $V_{EE} = -5.2$ V, V_T 为牵引电源, 可以取 -5.2 V 或 -2 V 等数值。MC10102 为 10K 系列 2 输入或非门, 在测试温度 +25°C 时, 输入和输出高、低电平电压范围分别为: $V_{IH} = (-1.105 \sim -0.81)$ V, $V_{IL} = (-1.85 \sim -1.475)$ V, $V_{OH} = (-0.96 \sim -0.81)$ V, $V_{OL} = (-1.85 \sim -1.65)$ V。下面分析当输入信号的高、低电平分别为 $V_{IH} = -0.9$ V, $V_{IL} = -1.75$ V 时, 电路的工作情况。

T_1 、 T_2 组成多端输入, 并与 T_3 管组成射极耦合差分电路。 T_4 组成一个简单的电压跟随器, 它为 T_3 提供一个参考电压 V_{REF} 。由于 ECL 中 PN 结的正向导通压降为 0.8 V, 可以计算出 T_4 的基极电压 $V_{B4} = -0.55$ V, $V_{REF} = -1.35$ V。为了补偿温度漂移, 在 T_4 的基极回路接入了两个二极管。 T_5 和 T_6 各组成电压跟随器, R_{L1} 和 R_{L2} 为外接负载或下一级门电路的输入电阻。射极输出器的作用是移动电平值, 使得输出端的高、低电平与输入端的高、低电平电压匹配, 并提高了带负载能力。

(1) 输入端 A、B 都接低电平 (-1.75 V)

由于 T_3 的基极电位比 T_1 的高, 因此 T_3 优先导通, 使发射极的电位 $v_E = V_{REF} - V_{BE3} = -2.15$ V,

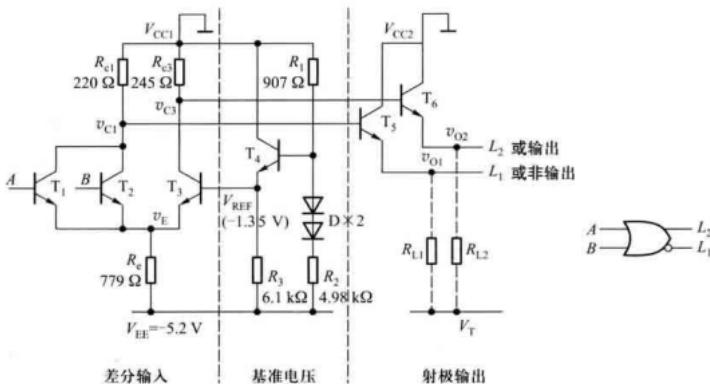


图 3.6.1 ECL 或/或非门电路结构及逻辑符号

这时 T_1 、 T_2 管发射结压降只有 0.4 V，因此 T_1 、 T_2 同时截止。若忽略 T_5 基极电流在 R_{e1} 上的压降，则 $v_{c1} = 0$ V， $v_{o1} = -V_{BE1} = -0.8$ V，即 L_1 输出为 ECL 逻辑高电平。

T_3 导通，流过 R_e 的电流由 T_3 提供， $i_e = (v_E - V_{EE})/R_e \approx 3.9$ mA。忽略 T_6 基极电流，可以求得 T_3 集电极电位 $v_{c3} = -i_e R_{e3} = -0.96$ V。 $v_{o2} = v_{c3} - V_{BE2} = -1.76$ V，即 L_2 为逻辑低电平。

另外，导通的 T_3 管集电结反偏，所以 T_3 处于放大状态，而不是饱和状态。

(2) 输入端 A 、 B 中有一个接高电平(设 A 接高电平-0.9 V)

由于 $v_A > V_{REF}$ ，所以 T_1 优先导通，使 $v_E = v_A - V_{BE1} = -1.7$ V，此时加到 T_3 管发射结压降只有 0.35 V，故 T_3 截止。忽略 R_{e3} 上的压降， $v_{o2} = -0.8$ V， L_2 为高电平输出。 T_1 导通使 R_e 的电流为 $i_e = (v_E - V_{EE})/R_e = 4.5$ mA，该电流在 R_{e1} 上产生压降，使 T_5 集电极电位 $v_{c1} = -i_e R_{e1} = -1$ V， $v_{o1} = -1.78$ V， L_1 为低电平输出。同样， T_1 的集电结近似为零偏，也未工作在饱和状态。

上述分析可见， v_{c1} 和 v_{c3} 输出的高、低电平与输入信号的高、低电平不匹配，无法驱动下一级门电路。因此， T_5 和 T_6 组成的射极输出器将其转换成所需的电平值。

由于 T_1 和 T_2 管是并联在一起的，只要 A 、 B 中有一个接高电平，都会使 L_1 为低电平，而 L_2 为高电平。因此

$$L_1 = \overline{A+B} \quad \text{或非输出}$$

$$L_2 = A+B \quad \text{或输出}$$

即 ECL 门的基本逻辑功能是同时具备或/或非输出，称之为互补逻辑输出，逻辑符号如图 3.6.1 所示。

不论是哪个 BJT 导通，所形成的发射极电流 i_e 都是很接近的，输入信号就像开关一样，控制 T_1 、 T_2 或 T_3 给 R_e 提供 i_e 这个电流，所以 ECL 电路又称为电流开关型逻辑(Current-MODE Logic)。

CML)。

以上所述的是具有 A、B 两个输入端的或非电路,只要增加相同类型的 BJT 与 T_1 并联,就能增加门电路的输入端。

2. ECL 门的工作特点

(1) 电路的输入级采用差分输入的形式,因此具有抑制漂移的能力。

(2) BJT 工作在截止区或放大区,集电极电位总高于基极电位,这就避免了 BJT 因工作在饱和状态而产生的存储电荷问题。

(3) 不论是哪个 BJT 导通,所形成的发射极电流 i_e 都几乎相等,因此电路在进行状态转换时,电源的电流保持常数。而 CMOS 或 TTL 电路进行状态转换时,从电源到地之间电路的电流产生一个尖峰,这个电流是产生噪声的重要根源。因此,ECL 电路内部开关噪声低。

(4) 逻辑电平的电压摆幅小。电压摆幅是指输出(或输入)电压最大值与最小值之差。电压摆幅小致使集电极输出电压的变化小,这不仅有利于电路的转换,而且可采用很小的集电极电阻 R_o 。同时, T_3 和 T_6 组成的电压跟随器进一步减小了输出电阻。因此 ECL 门的输出电阻很低,使输出回路的时间常数比一般饱和型电路小,因而开关速度快,常用于高速系统中。

(5) 互补输出为简化逻辑设计带来方便。

它的主要缺点是制造工艺要求高,功耗大。其逻辑摆幅小,噪声容限只有 0.2 V 左右,因而抗外界干扰能力较弱。而且由于输出电压为负值,若需与其他门电路接口,需用专门的电平移动电路。

双极型逻辑门电路除了 TTL 和 ECL 之外,尚有集成注入逻辑门电路(Integrated-Injection Logic, IIL 或 I^2L)。由于 IIL 电路简单,易于在硅片上实现高集成度的器件,因而在大规模和超大规模集成电路中得到应用,但很少用来制作中、小规模电路。由于它的高、低电平电压差值很小,抗干扰能力较差,因而这种门电路的推广受到限制。

复习思考题

3.6.1 与 TTL 系列相比,ECL 系列门电路的主要优点是什么?

3.6.2 列举出 ECL 系列门电路的两个主要缺点。

3.7 逻辑描述中的几个问题

3.7.1 正负逻辑问题

1. 正负逻辑的规定

在数字电路中,可以采用两种不同的逻辑体制表示电路输入和输出的高、低电平。在前面讨

论时,将高电平用逻辑 1 表示,低电平用逻辑 0 表示,这种表示方法称为正逻辑体制。如果将高电平用逻辑 0 表示,低电平用逻辑 1 表示,则这种表示方法称为负逻辑体制。

对于同一电路的输入与输出关系的描述,可以采用正逻辑,也可以采用负逻辑。正逻辑和负逻辑两种体制不牵涉到逻辑电路本身的结构问题,但根据所选正负逻辑的不同,即使同一电路也具有不同的逻辑功能。例如某个逻辑门电路的输入和输出电平如表 3.7.1 所示,其中 H 和 L 分别表示高、低电平。如果采用正逻辑体制,令 H 为 1,L 为 0,得到如表 3.7.2 所示的真值表,它表示与非逻辑关系 $L = \overline{A \cdot B}$ 。如果采用负逻辑体制,令 H 为 0,L 为 1,得到如表 3.7.3 所示的真值表,它表示或非逻辑关系 $L = \overline{A+B}$ 。因此,正逻辑的与非门等效于负逻辑的或非门。正逻辑和负逻辑只是看问题的角度或分析问题的方法不同而已,问题的实质是不变的,即电路输入与输出的电平关系始终是不变的。本书如无特殊说明,一律采用正逻辑,即规定高电平为逻辑 1,低电平为逻辑 0。

表 3.7.1 某电路输入与输出电平表

A	B	L
L	L	H
L	H	H
H	L	H
H	H	L

表 3.7.2 正与非门真值表

A	B	L
0	0	1
0	1	1
1	0	1
1	1	0

表 3.7.3 负或非门真值表

A	B	L
1	1	0
1	0	0
0	1	0
0	0	1

2. 正负逻辑的等效变换

工程实践中,电路描述一般采用正逻辑体制,负逻辑体制用的比较少。上述分析可知,正逻辑的与非对应负逻辑的或非;同理,正逻辑的或非与负逻辑的与非相对应。因此,如果需要,可以按下列方式进行两种逻辑体制的互换:

与非 \Leftrightarrow 或非

与 \Leftrightarrow 或

非 \Leftrightarrow 非

3.7.2 基本逻辑门的等效符号及其应用

1. 基本逻辑门的等效符号

利用摩根定理对基本逻辑运算进行变换,可以得到不同形式的表达式。例如与非逻辑运算的表达式可以写成

$$L = \overline{\overline{AB}} = \overline{A} + \overline{B}$$

由此,可以得到与非门的等效符号如图 3.7.1 所示。输入端的小圆圈表示先对信号作非运算,然后进行或运算。



图 3.7.1 与非门及其等效符号

对于或非运算的逻辑表达式,可以写成

$$L = \overline{A+B} = \overline{A} \cdot \overline{B}$$

所以得到其等效符号如图 3.7.2 所示。



图 3.7.2 或非门及其等效符号

同理,利用摩根定理对与门和或门的逻辑表达式进行变换,可以得到它们的等效符号分别如图 3.7.3 和图 3.7.4 所示。



图 3.7.3 与门及其等效符号



图 3.7.4 或门及其等效符号

上述各图所示的逻辑符号及其等效符号,是在同一逻辑体制下,用两种不同的方式描述同一逻辑运算。因此,不能将等效符号看成是负逻辑体制或者负逻辑表示方法。本书采用正逻辑体制,所以对于输入和输出均是高电平为 1,低电平为 0。可以用真值表验证各逻辑符号及其等效符号是等价的。

2. 逻辑门等效符号的应用

与门是由与非门和非门级联构成的,或门是由或非门和非门构成的。与非门和或非门比与门和或门电路结构简单,速度快。通常采用与非门或者或非门实现电路。利用逻辑门等效符号对逻辑电路进行变换,在不改变电路逻辑功能的前提下,可以简化电路,减少实现电路的门的种类,提高工作速度。

图 3.7.5(a)所示电路由两级组成,第一级是两个与门,第二级是一个或门。

利用摩根定理 $\overline{\overline{X}} = X$,在图 3.7.5(a)中间连线的两端各加一个圆圈,相当于进行两次非运算,但并没有改变电路的功能,得到图 3.7.5(b)所示电路。然后将图 3.7.5(b)所示电路第二级与非门的等效符号用与非门符号代替,就可以得到图 3.7.5(c)所示电路。该电路的工作速度比图

3.7.5(a) 所示电路快。

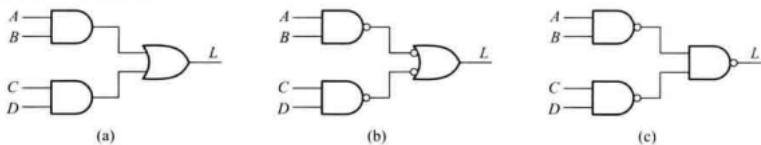


图 3.7.5 或门及其等效符号

(a) 逻辑电路 (b) 逻辑电路的等效变换 (c) 用与非门替代等效符号

3. 逻辑门等效符号强调低电平有效

在 3.3.2 节介绍三态门时,就涉及有效电平的概念。三态门的使能控制信号可以是高电平有效,也可以是低电平有效。对于高电平使能的三态门,当使能端信号为 1 时,电路处于正常逻辑工作状态;对于低电平使能的三态门,当使能端信号为 0 时,电路正常工作。有效电平的概念不止限于使能端信号。在实际电路,特别是大规模集成芯片中,任何输入或者输出信号都有可能是高电平有效,或者是低电平有效。所谓低电平有效,是指当信号为低电平时,电路完成规定的操作;而高电平有效,是指信号为高电平时,电路完成规定的操作。

图 3.7.6 所示是一个可以控制数据传输的电路。其中集成芯片 IC 的使能端 \overline{EN} 要求低电平有效,电路的两个控制信号分别是请求信号 RE 和允许信号 \overline{AL} 。图中, G_2 门是输入、输出均为低有效的与门。根据图 3.7.4 可知, G_2 门实际是或门的等效符号。这里之所以采用等效符号是为了强调低电平有效,以便于理解实际电路中,请求信号 RE 、允许信号 \overline{AL} 以及 IC 芯片的使能信号 \overline{EN} 之间的逻辑关系。当请求信号 RE 为高电平信号,允许信号 \overline{AL} 为低电平信号时, G_2 门的两个输入端均为有效信号,即低电平,则产生一个有效的输出信号,即 \overline{L} 为低电平,使 \overline{EN} 为低电平,允许 IC 传输数据。

信号名称 \overline{EN} , \overline{AL} 和 \overline{L} 上面的横线表示该信号是低电平有效,在进行逻辑运算时,应该作为一个整体符号。如果在运算过程中,变量上面的“-”号参与运算,则在画逻辑电路图、或者验证真值表时,应将其还原为低有效符号。

需要注意的是,如果一根连线的两端都有圆圈,并且都包含有非运算的含义,可以用“圈圈相消”进行电路化简,如图 3.7.5 所示。但在图 3.7.6 中, G_2 门的输出与使能端之间的连线两端也有圆圈,这两个圆圈不能抵消,因为集成芯片 IC 使能端的圆圈的功能在芯片内部,不能被化简掉。

如果要求请求信号 RE 和允许信号 AL 均为高电平有效,而芯片 IC 的使能端 \overline{EN} 仍为低有效,可以采用如图 3.7.7(a) 所示的控制电路。 G_2 门可以看成是输入为高电平有效,输出为低电平有效的与门。

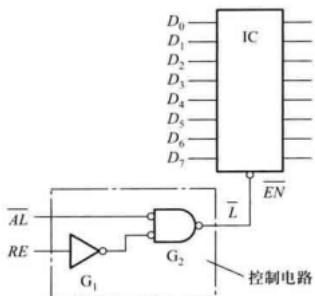


图 3.7.6 数据传输控制电路

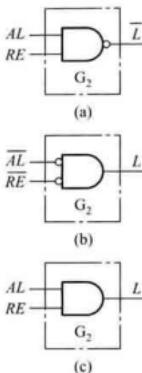


图 3.7.7 几种不同的控制电路

(a) 与非门实现 (b) 或非门实现 (c) 与门实现

如果要求请求信号 \overline{RE} 和允许信号 \overline{AL} 均为低电平有效,而芯片 IC 的使能端 EN 为高有效,则采用如图 3.7.7(b) 所示的控制电路。 G_2 门可以看成是输入为低电平有效,输出为高电平有效的与门。根据图 3.7.2 可知, G_2 门是或非门的等效符号。

同理,如果要求请求信号 RE 和允许信号 AL 均为高电平有效,芯片 IC 的使能信号也为高电平有效,则采用如图 3.7.7(c) 所示的控制电路。 G_2 门为输入、输出均是高电平有效的与门。

复习思考题

3.7.1 列出正逻辑体系或非门和负逻辑体系与非门的真值表,并说明两者的等效关系。

3.7.2 为什么说逻辑符号及其等效符号不是正负逻辑关系?

3.7.3 对于小圆圈画在输入端或者画在输出端的非门,其逻辑运算结果是否相同,所表达的含义是否相同?

3.8 逻辑门电路使用中的几个实际问题

以上重点讨论了 CMOS 系列门电路,同时还介绍其他几种逻辑门电路。在实际应用中,可以根据电源电压、传输延迟时间、功耗、噪声容限、带负载能力等要求来选择相应的门电路。有时需要混合使用不同系列的逻辑门电路,此时将遇到不同逻辑门电路之间的接口问题。另外,门电路与负载之间的匹配也是需要考虑的问题之一。下面对几个实际问题进行讨论。

3.8.1 各系列逻辑门电路之间的接口问题

CMOS 电路的动态功耗为 $P_d = (C_{PD} + C_L) V_{DD}^2 f$ 。为了减小功耗,采用低电源电压器件。同时电路的集成度不断提高使晶体管尺寸越来越小,CMOS 的栅极与源极、栅极与漏极间的绝缘层也越来越薄,不足以承受 5 V 电源电压。JEDEC(Join Electron Device Engineering Council) 固态技术协会选择 3.3 V、2.5 V、1.8 V、1.5 V 和 1.2 V 等一系列低电压作为各种低电压逻辑器件的供电电源。并规定了这些供电电源下输入和输出高、低逻辑电平的电压值。

图 3.8.1 给出了各个系列在给定电源电压下四个逻辑电平参数: 输入低电平的上限值 $V_{IL(max)}$ 、输入高电平的下限值 $V_{IH(min)}$ 、输出低电平 ($I_{OL} = 4 \text{ mA}$) 的上限值 $V_{OL(max)}$ 、输出高电平 ($I_{OH} = 4 \text{ mA}$) 的下限值 $V_{OH(min)}$, 便于进行逻辑电平的快速比较。为简单起见, 图中省略了下标 min 和 max。

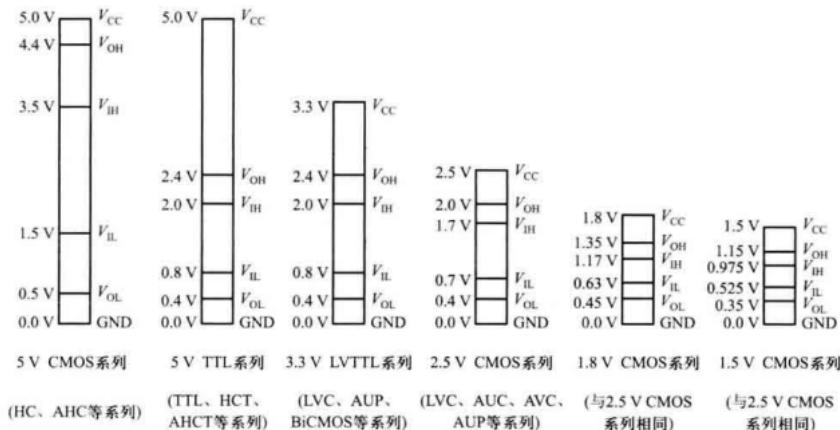


图 3.8.1 几种逻辑电平比较

在图 3.8.1 中, 3.3 V LVTTL 是指 3.3 V CMOS 低电压器件所带 TTL 负载时的电压值。逻辑门的负载不同时, 其输出高、低电平的下限值和上限值也不同, 当其输出电流 I_o 低于 100 μA 称为带 CMOS 负载, 此时 $V_{OH} = 3.1 \text{ V}$, $V_{OL} = 0.2 \text{ V}$ 。当 I_o 比较大时称为带 TTL 负载。图中所示为 $I_o = 4 \text{ mA}$ 时的 $V_{OH} = 2.4 \text{ V}$, $V_{OL} = 0.4 \text{ V}$ 。

图 3.8.1 给出的是典型的逻辑电平值, 不同系列电路的具体参数可能与典型值不同, 分析具体的实际问题时, 必须查阅生产厂商提供的数据表。

现在的很多数字电路或数字系统中, 同时采用不同供电电压的逻辑器件, 主要是为了降低成本, 兼顾新老器件或设备的应用。不同系列逻辑器件的混合使用, 需要考虑它们的匹配问题。

当一个系统中使用两种不同系列逻辑门电路时,由于它们的电压和电流参数各不相同,首先要考虑门电路的输入或输出电平是否超过数据手册规定的极值,即

$$V_{I(\min)} \leq V_I \leq V_{I(\max)} \quad (3.8.1)$$

$$V_{O(\min)} \leq V_O \leq V_{O(\max)} \quad (3.8.2)$$

如果不满足上述两个条件,有可能损坏芯片。

第二是逻辑电平兼容性问题,驱动器件的输出电压必须满足负载器件所要求的高电平或者低电平输入电压的范围。即

$$V_{OH(\min)} \geq V_{IH(\min)} \quad (3.8.3)$$

$$V_{OL(\max)} \leq V_{IL(\max)} \quad (3.8.4)$$

第三是逻辑门电路的扇出问题,即驱动器件必须能对负载器件提供足够的灌电流或者拉电流。

灌电流情况下应满足 $|I_{OL(\max)}| \geq |I_{IL(\text{total})}|$ (3.8.5)

拉电流情况下应满足 $|I_{OH(\max)}| \geq I_{IH(\text{total})}$ (3.8.6)

其余如噪声容限、输入和输出电容以及开关速度等参数在某些设计中也必须予以考虑。下面分别就不同供电电压的逻辑电路之间的接口问题和 5 V 供电电压的 CMOS 电路与 TTL 电路之间的接口问题进行讨论。

1. 5 V CMOS 电路与 3.3 V CMOS 电路的接口

两种供电电源的 CMOS 电路相连时,首先要考虑它们输入输出电平是否满足式(3.8.1)和式(3.8.2)的要求。

(1) 输入电压的极值 $V_{I(\max)}$ 和 $V_{I(\min)}$

有些逻辑门电路允许输入 V_I 超过电源电压 V_{DD} ,而有些不允许。

例如:74HC 系列门电路的最大输入 $V_{I(\max)} = V_{DD} + 0.5 \text{ V}$,即输入 V_I 不能超过电源电压 V_{DD} 。图 3.8.2 所示为 74HC 系列的输入保护电路。正常工作时,不论输入为高电平还是低电平,二极管 D_1 和 D_2 均截止,电路输入端电阻很高。当输入电压超过 V_{DD} 时, D_1 导通,有很大的电流流过 D_1 ,并且使得该支路的等效电阻很小,电路不能正常工作。

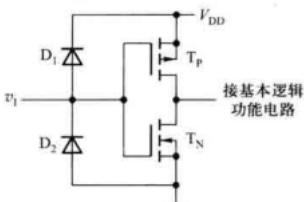


图 3.8.2 74HC 系列的输入电路

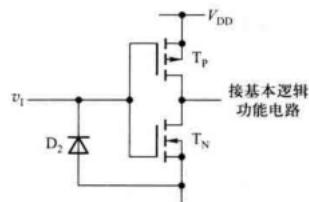


图 3.8.3 74AHC 系列的输入电路

74AHC 系列 $V_{I(\max)} = 7 \text{ V}$ 。当该系列采用 3.3 V 电源电压时,允许输入电压 $V_I > 3.3 \text{ V}$,只要小于 7 V 即可。74AHC 系列输入保护电路如图 3.8.3 所示。该电路去掉了输入到电源之间的

二极管,因此允许 V_I 高于 V_{DD} ,同时要求电路的 MOS 管能够承受足够高的电压。

输入电压的极小值 $V_{I(min)}$ 通常为 0 V。如果考虑输入保护二极管的作用时, $V_{I(min)} = -0.5$ V。

(2) 输出电压的极值 $V_{O(max)}$ 和 $V_{O(min)}$

有些逻辑门电路允许输出 V_O 超过电源电压 V_{DD} ,而有些则不允许。特别是三态输出的 5 V 和 3.3 V 两种逻辑电路接到同一个数据总线时,如果 5 V 逻辑电路输出高电平,而 3.3 V 逻辑电路处于高阻态。5 V 高电平输出信号通过总线也连接到 3.3 V 逻辑电路的输出端,此时需要考虑输出端允许的极限值。

例如:74HC 和 74AHC 系列电路的最大输出 $V_{O(max)} = V_{DD} + 0.5$ V,即输出 V_O 不能超过电源电压 V_{DD} 。图 3.8.4 所示为 74HC/74AHC 系列的三态输出电路,电路供电电源为 3.3 V。正常输出高阻态时, T_p 和 T_N 管均截止。当 5 V 电压加到输出端时,使二极管 D_1 正偏导通, T_p 的漏极与源极之间形成一个低阻通路,并且产生很大的电流,因此不允许输出端电压超过 $V_{O(max)} = V_{DD} + 0.5$ V。

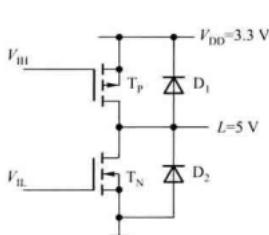


图 3.8.4 74HC 和 74AHC 系列的三态输出电路

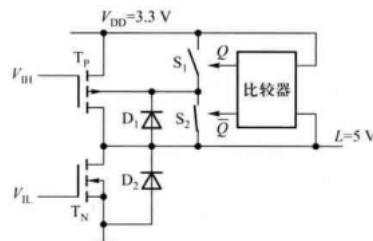


图 3.8.5 74LVC 系列的三态输出电路

对于 74LVC/AUC/AUP 等系列的输出端为高阻或电路电源关闭的情况下, $V_{O(max)} = 6.5$ V。即 $V_{DD} = 3.3$ V 时,其输出电压 V_O 可以高于 3.3 V,只要小于 6.5 V 即可。图 3.8.5 所示为带保护电路的 74LVC 系列的三态输出电路。当输出端电压高于 $V_{DD} + 0.5$ V 时,比较器使 S_1 开路, S_2 闭合,电流通路消失,其输出仍然保持高阻态,就能与 5 V 器件相接。

输出电压的极小值 $V_{O(min)}$ 通常为 0 V,如果考虑输入保护二极管的作用时, $V_{O(min)} = -0.5$ V。

(3) 逻辑电平兼容性及输入输出电流匹配问题

根据图 3.8.1 所示的电压数据可见,5 V CMOS 系列驱动 3.3 V CMOS 系列时, $V_{OH(min)} = 4.4$ V, $V_{IH(min)} = 2.0$ V, $V_{OL(max)} = 0.5$ V, $V_{IL(max)} = 0.8$ V, 满足式(3.8.3)和式(3.8.4)对逻辑电平的要求。

根据表 3.3.4 给出的电流参数可知,5 V CMOS 系列的高电平最大输出电流 $I_{OH(max)}$ 和低电平最大输出电流 $I_{OL(max)}$ 都在 20 μ A 以上,3.3 V 74LVC/ALVC 系列的高、低电平最大输入电流 $I_{IH(max)}$ 和 $I_{IL(max)}$ 均为 5 μ A,只要负载数目适当,即可满足式(3.8.5)和式(3.8.6)对输入输出电流的要求。因此,5 V CMOS 系列可以直接驱动 3.3 V CMOS 系列。

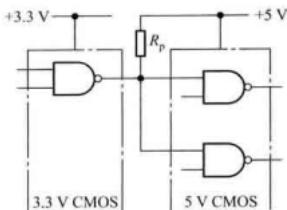


图 3.8.6 用上拉电阻提高 3.3 V CMOS 电路的输出高电平

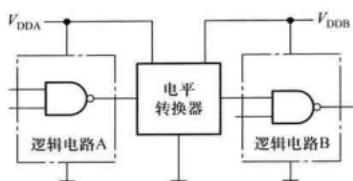


图 3.8.7 电平转换器用做接口

当用 3.3 V CMOS 系列驱动 5 V CMOS 系列时, 根据图 3.8.1 所示的电压数据, $V_{OL(max)} = 0.4 \text{ V}$, $V_{IL(max)} = 1.5 \text{ V}$, 满足式(3.8.4)的要求。但是 $V_{OH(min)} = 2.4 \text{ V}$, 低于 5 V CMOS 系列的 $V_{IH(min)} = 3.5 \text{ V}$, 不满足式(3.8.3)的要求。为了解决这一矛盾, 最简单的方法是在 3.3 V CMOS 电路的输出端与 +5 V 电源之间接一个上拉电阻 R_p , 如图 3.8.6 所示。上拉电阻的值取决于负载器件的数目以及驱动门和负载门的电流参数, 可以用 OD 门外接上拉电阻的计算方法进行计算。但必须注意, 此时 $V_{OH(min)} < V_{IH(min)}$ 。为保证负载门输入高电平的要求, 应将式(3.3.2)中的 $V_{OH(min)}$ 换成 $V_{IH(min)}$ 进行计算。当驱动门电路输出为高电平时, 如图 3.8.5 所示的输出结构, T_N 和 T_{P1} 、 T_{P2} 管均截止, 因此有

$$V_{OH} = V_{DD} - R_p(I_{OZ} + I_{IH(total)}) \quad (3.8.7)$$

I_{OZ} 为 3.3 V CMOS 门电路输出级 T_N 管均截止时的漏电流, $I_{IH(total)}$ 为流入全部 5 V CMOS 负载电路的电流, 这两个电流的数值都很小, 如果 R_p 取值不太大, 驱动门输出将被提高至接近 V_{DD} 。

除了用上拉电阻的方法外, 还可以采用 OD 门或专门的逻辑电平转换器。

2. 低电压 CMOS 电路之间的接口

3.3 V CMOS 系列驱动 2.5 V CMOS 系列时, 从图 3.8.1 可见, $V_{OH(min)} = 2.4 \text{ V}$, $V_{IH(min)} = 1.7 \text{ V}$, $V_{OL(max)} = 0.4 \text{ V}$, $V_{IL(max)} = 0.7 \text{ V}$, 满足式(3.8.3)和式(3.8.4)对逻辑电平的要求。如果 2.5 V CMOS 系列允许输入最大值 $V_{I(max)}$ 为 3.3 V, 可以由 3.3 V CMOS 系列直接驱动 2.5 V CMOS 系列。当用 2.5 V CMOS 系列驱动 3.3 V CMOS 系列时, $V_{OH(min)} = 2 \text{ V}$, $V_{IH(min)} = 2 \text{ V}$, 此时高电平噪声容限为 0, 这在实际应用中是不允许的。

为满足不同系列逻辑电路之间的接口需求, 特别是 2.5 V 以下逻辑电路之间的接口, 通常使用专门的逻辑电平转换器, 如图 3.8.7 所示。电平转换器有两个独立的电源 V_{DDA} 和 V_{DDB} , 分别与两种系列逻辑电路的电源相接。例如 74LVC1T45 是小逻辑封装(参看 3.8.4 节)的 1 位双向逻辑电平转换器, 并具有驱动总线的三态输出结构, 可以实现 5 V、3.3 V、2.5 V、1.8 V 或 1.5 V 逻辑电平之间的相互转换, 电路如图 3.8.8 所示。当控制端 DIR 接 V_{DDA} 时, 则 A 端信号传到 B 端, DIR 接地时, 则 B 端信号传到 A 端。

表 3.8.1 74ALVC164245 的功能表

输入		操作
\overline{OE}	DIR	
L	L	数据 B 传送到 A
L	H	数据 A 传送到 B
H	x	隔离

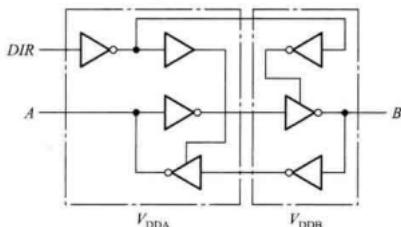


图 3.8.8 74LVC1T45 电平转换器内部电路图

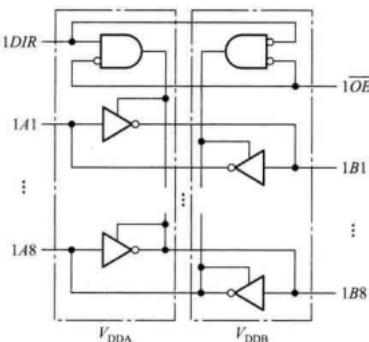


图 3.8.9 74ALVC164245 电平转换器内部电路图

如果同时需要多位逻辑电平转换器, 可使用宽总线封装的芯片。74ALVC164245 是 16 位双向逻辑电平转换器, 具有驱动总线的三态输出结构。集成芯片内部是将 16 位电平转换器分成 2 组, 每组有 8 位, 组与组之间相互独立。每组结构如图 3.8.9 所示。**74ALVC164245 的功能表**如表 3.8.1 所示, 控制端 DIR 和 \overline{OE} 接 V_{DDA} 一边的电源或地, 以控制信号的传输方向。使用时可根据需要连接成 16 位或两组 8 位, 两个独立的电源 V_{DDA} 和 V_{DBB} 可接 1.4 ~ 3.6 V 的电压, 实现 3.3 V、2.5 V、1.8 V 或 1.5 V 逻辑电平之间的相互转换。电源电压为 3.3 V 时, 输出电流可达到 12 mA, 电源电压为 2.5 V 时输出电流可以达到 8 mA。

* 3.5 V CMOS 电路与 TTL 电路的接口

(1) 5 V CMOS 电路驱动 TTL 电路

由图 3.8.1 可知 5 V CMOS 电路参数 $V_{OH(min)} = 4.4$ V, $V_{OL(max)} = 0.5$ V。TTL 电路参数 $V_{IH(min)} = 2.0$ V, $V_{IL(max)} = 0.8$ V。两者的逻辑电平参数满足式(3.8.3)和式(3.8.4)的要求。从表 3.3.4 和表 3.5.1 给出的电流参数可知, 74HC/HCT 系列的 $I_{OH(max)}$ 和 $I_{OL(max)}$ 均为 4 mA, 74AHC/AHCT 系列的 $I_{OH(max)}$ 和 $I_{OL(max)}$ 均为 8 mA, TTL 系列的输入电流 $I_{IH(max)}$ 低于 50 μ A, $I_{IL(max)}$ 低于 2 mA。只要负载数目适当, 即可满足式(3.8.5)和式(3.8.6)对灌电流和拉电流的要求。因此, 5 V CMOS

系列可以直接驱动 TTL 电路。

(2) TTL 电路驱动 5 V CMOS 电路

根据表 3.3.4 和表 3.5.1 给出的电压参数可知, TTL 系列的 $V_{OH(\min)} = 2.7 \text{ V}$, $V_{OL(\max)} = 0.5 \text{ V}$, 74HC 和 74AHC 系列的 $V_{IH(\min)} = 3.5 \text{ V}$, $V_{IL(\max)} = 1.5 \text{ V}$, 低电平的输出与输入电平兼容, 但高电平不满足式(3.8.3)的要求。可以在 TTL 输出端与 +5 V 电源之间接一个上拉电阻(参考图 3.8.6 所示电路), 或采用 OC 门输出结构的电路。

74HCT 和 74AHCT 系列是为了便于与 TTL 直接接口而设计的 CMOS 系列, 其 $V_{OH(\min)}$ 值降为 2 V。根据表 3.3.4 和表 3.5.1 给出的电压和电流参数可以验证, 将 TTL 系列电路与 74HCT 和 AHCT 系列连接时, 满足式(3.8.3)~式(3.8.6)对逻辑电平兼容性和输入输出电流匹配的要求。因此, TTL 系列可以直接驱动 74HCT 和 74AHCT 系列 CMOS 电路。

*4.3.3 V CMOS 电路与 TTL 电路的接口

从图 3.8.1 可见, TTL 电路与 3.3 V CMOS 电路电平是兼容的, 满足式(3.8.3)和式(3.8.4)的要求。根据表 3.3.4 和表 3.5.1 给出的电流参数可知, 74LVC 电路驱动 TTL 电路时, $I_{OH(\max)} = I_{OL(\max)} = 24 \text{ mA}$, TTL 系列的 $I_{IH(\max)}$ 和 $I_{IL(\max)}$ 低于 0.6 mA, 电流相匹配, 满足式(3.8.5)和式(3.8.6)的要求。74LVC 电路可以直接驱动 TTL 电路。

当 TTL 电路驱动 74LVC 电路时, 同样验证电流相匹配, 并且 74LVC 电路最大输入 $V_{I(\max)} = 6.5 \text{ V}$, TTL 电路可以直接驱动 74LVC 电路。

其他 3.3 V CMOS 电路与 TTL 电路接口时, 同样需要验证输入输出电压的最大值, 考虑电平兼容性及电流的匹配问题。

3.8.2 逻辑门电路驱动其他负载时的接口

1. 驱动显示器件

在数字电路中, 往往需要用发光二极管来显示信息, 如电源接通或者断开的指示、七段数码显示或图形符号显示等。

图 3.8.10 所示为反相器驱动一发光二极管 LED 电路。电路中串接了一限流电阻 R 以保护 LED。限流电阻的大小可分别按下面两种情况来计算。

对于图 3.8.10(a)所示电路, 当门电路的输出为高电平时, LED 发光, 则

$$R = \frac{V_{OH} - V_F}{I_D} \quad (3.8.8)$$

反之, 图 3.8.10(b)所示电路, 当门电路的输出为低电平时 LED 发光, 故有

$$R = \frac{V_{CC} - V_F - V_{OL}}{I_D} \quad (3.8.9)$$

以上两式中, I_D 为 LED 的电流, V_F 为 LED 的正向压降, V_{OH} 和 V_{OL} 为门电路的输出高、低电平

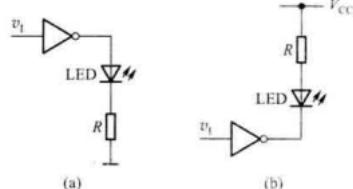


图 3.8.10 反相器驱动 LED 电路

电压值。

例 3.8.1 试用 74HC04 六个 CMOS 反相器中的一个作为接口电路, 当门电路的输入为高电平时, 使 LED 导通发光。已知 LED 正常发光需要几毫安的电流, 并且导通时的压降 V_F 为 1.6 V。

解: 根据表 3.3.4 中 74HC 系列的数据, 当 $V_{ce} = 5$ V 时, $V_{ol} = 0.33$ V, $I_{ol(max)} = 4$ mA, 因此 I_o 取值不能超过 4 mA。根据式(3.8.9)计算限流电阻的最小值为

$$R = \frac{(5 - 1.6 - 0.33) \text{ V}}{4 \text{ mA}} = 768 \Omega$$

相应的电路如图 3.8.10(b) 所示。

2. 驱动机电性负载

在工程实践中, 往往会遇到用各种数字电路来控制机电性系统的功能, 如控制电动机的转角和转速, 继电器的接通与断开, 流体系统中阀门的开通和关闭, 自动生产线中的机械手多参数控制等。这些机电系统所需的工作电压和工作电流比较大, 即使微型继电器的驱动电流也会在 10 mA 以上。要使这些机电系统正常工作, 必须扩大驱动电路的输出电流以提高带负载能力, 而且必要时要实现电平转移。

如果负载所需的电流不特别大, 例如微型继电器, 可以将两个反相器并联后的拉电流作为驱动电流, 如图 3.8.11 所示。二极管 D 用来钳位电感产生的反电动势。即使封装在同一芯片内的两个反相器的参数也有差别, 因此, 并联后总的驱动电流略小于单个门最大负载电流的两倍。

如果负载所需的电流比较大, 达到几百毫安, 则需要在数字电路的输出端与负载之间接入一个功率驱动器, 称之为外围驱动器。功率驱动器有多种类型, 但电路结构一般都具有以下两个特点: 一是采用集电极开路输出结构, 其输出高电平几乎等于外加电压, 通过调节外加电压来满足不同负载对高电平电压的要求。二是驱动电路的输出晶体管具有较强的带负载能力, 能提供较大的电流。具体外围驱动器的电路结构可查阅有关数据手册。

图 3.8.12 所示为达林顿晶体管阵列 ULN2003A 作为驱动机电性负载的功率驱动器。ULN2003A 集成芯片中有 7 组 OC 反相器和二极管 D 组成的电路, 这里只用了 1 组。若需更大的电流也可以将 2 组并联使用。ULN2003A 的输入与 TTL 或 5 V CMOS 集成电路兼容, 输出端可直接用于驱动机电系统, 其最大驱动电流为 500 mA, 最高驱动电压为 50 V。

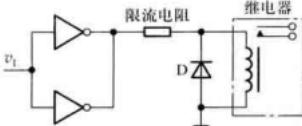


图 3.8.11 用并联逻辑门提高驱动能力

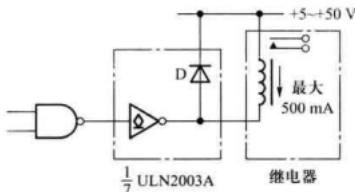


图 3.8.12 用功率驱动器提高驱动能力

3.8.3 抗干扰措施

利用逻辑门电路(CMOS或TTL)作具体的电路设计时,还应当注意下列几个实际问题。

1. 多余输入端的处理措施

集成逻辑门电路在使用时,一般不让多余的输入端悬空,以防引入干扰信号。对多余输入端的处理以不改变电路工作状态及稳定可靠为原则,如图3.8.13所示。一是将它与其他输入端并接在一起。二是根据逻辑要求,与门或者与非门的多余输入端通过 $1\sim3\text{ k}\Omega$ 电阻接正电源,对CMOS电路可以直接接电源。或门或者或非门的多余输入端接地。对于高速电路的设计,输入端并接会增加等效的电容性负载,而使信号的传输速度下降,最好采用图3.8.13所示的后两种方法。

特别是CMOS电路的多余输入端绝对不能悬空。由于它的输入电阻很大,容易受到静电或工作区域工频电磁场引入电荷的影响,从而破坏电路的正常工作状态。

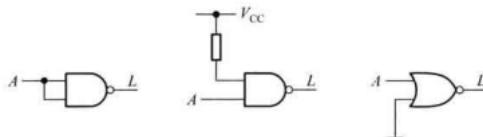


图3.8.13 多余输入端的处理电路

2. 去耦合滤波电容

数字电路或数字系统往往由多片逻辑电路构成。它们由一公共的直流电源供电。这种电源是非理想的,一般由整流稳压电路供电,具有一定的内阻抗。当数字电路在高、低状态之间交替变换时,会产生较大的脉冲电流或尖峰电流。当它们流经公共的内阻抗时,必将产生相互影响,甚至使逻辑功能发生错乱。一种常用的处理方法是采用去耦合滤波电容,用 $10\sim100\mu\text{F}$ 的大电容器接在直流电源与地之间,滤除干扰信号。除此以外,对于每一集成芯片的电源与地之间接一个 $0.1\mu\text{F}$ 的电容器以滤除开关噪声。

3. 接地和安装工艺

正确的接地技术对于降低电路噪声是很重要的。方法是将电源地与信号地分开,先将信号地汇集在一点,然后将二者用最短的导线连在一起,以避免含有多种脉冲波形(含尖峰电流)的大电流引到某数字器件的输入端而破坏系统正常的逻辑功能。此外,当系统中同时有模拟和数字两种器件时,同样需将二者的地分别连在一起,然后再选用一个合适共同点接地,以避免二者之间的影响。必要时,也可设计模拟和数字两块电路板,各备直流电源,然后将二者的地恰当地连接在一起。在印制电路板的设计或安装中,要注意连线尽可能短,以减少接线电容产生寄生反馈而引起的寄生振荡。这方面更详细的介绍,可参阅有关文献。某些典型电路应用设计也可参考集成数字电路的数据手册。

3.8.4 CMOS 通用逻辑电路中的小尺寸逻辑和宽总线系列

CMOS 逻辑集成器件从 20 世纪 60 年代至今,特别是近 20 年来,由于制造工艺的不断改进,在提高集成度、缩短传输延迟时间和减小单元电路功耗等方面取得了很大的进步,生产出种类繁多的标准化、系列化的 CMOS 通用集成电路产品。每一种新的通用集成逻辑器件系列的生产,都是运用新技术对逻辑电路性能进行改进的结果。正如前面介绍的 4000 系列、HC/HCT 系列、AHC/AHCT 系列,以及低电压 LVC 系列、AUC 系列和 AUP 系列等。

根据逻辑功能的特点,可将数字集成芯片分为通用型和专用型两类。中、小规模数字集成芯片都属于通用型集成芯片,它们的通用性很强。由多个不同的通用集成芯片连接起来可以构成各种数字电路或数字系统。通用型集成芯片的逻辑功能是固定的,不能为设计某一特定逻辑电路而改变。芯片内包含的逻辑门数量较少,构成大型逻辑电路时所用芯片种类和数量多、体积大、可靠性差。如果将所设计的数字系统制作在一块半导体芯片上,就构成了具有专门用途的集成芯片,即专用集成电路(ASIC)。ASIC 芯片减小了电路的体积、重量和功耗,提高了可靠性。但设计和生产 ASIC 电路的成本高、周期较长,并且用户不能修改。为此,半导体厂家生产出可编程逻辑器件。

可编程逻辑器件是通用逻辑器件,其逻辑功能可以由用户自己设定。用户通过软件编程可以实现所有通用集成单元电路的功能,也可以满足一般的数字系统设计需要。可编程逻辑器件的应用使得传统通用逻辑芯片失去了市场。

作为大规模可编程逻辑器件的补充或接口电路,小尺寸逻辑(Little Logic)系列应运而生。相比传统通用逻辑芯片,小尺寸逻辑芯片的体积更小。小尺寸逻辑芯片不是用来构成电路或系统,而是用来修改或完善大规模集成芯片之间连线或外围电路的连线。另外,微处理器和计算机的进一步发展,要求性能更优越的总线驱动器件,出现了宽总线(Widebus)系列。实际应用中,各种高速服务器、通信和网络设备也需要不同规格的小尺寸逻辑或宽总线逻辑器件,以满足它们的灵活性要求。因此,小尺寸逻辑和宽总线电路成为通用型逻辑器件的主流器件,但比传统的通用集成电路系列的品种少得多。

1. 小尺寸逻辑电路

小尺寸逻辑器件是将 1 个、2 个或 3 个等少数几个门封装在一个芯片中构成小封装的超小规模集成电路。通常作为大规模集成电路的补充电路,用以完善电路的设计,或作为大规模集成芯片之间的接口电路。

以 2 输入与非门为例,传统的通用集成电路是将 4 个门封装在一个芯片中,如图 3.8.14 所示。小尺寸逻辑电路是将其分解成 1 个或 2 个门分别封装,面积分别减少了 97.3% 和 95.7%,如图 3.8.15 所示。集成芯片的封装技术有许多种类,TSSOP(Thin Shrink Small Outline Package) 和 Micro QFN(Quad Flat Non-leaded Package) 都是表面贴装型封装技术。TSSOP 采用表面安装技术,在芯片周围做引脚直接附在 PCB 板的表面。Micro QFN 封装配置有接触电极,无引脚,贴装面积很小。Nano Star 使用管芯作为封装,是当前最小的封装。

采用小尺寸逻辑电路可以减小体积,提高工作速度。例如,某电路需要插入一个2输入与非门,当使用传统的通用集成芯片时,就会有3个与非门闲置不用造成浪费。即使需要多个门,如果需要分布在较远的位置,会造成布线困难。小尺寸逻辑芯片不仅大大减少了面积,而且可以降低布线对电路工作性能的影响。

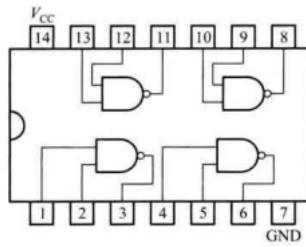


图 3.8.14 传统封装的 2 输入与非门

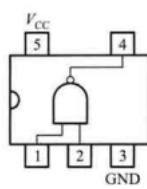


图 3.8.15 小尺寸逻辑封装的 2 输入与非门
面积为 0.9 mm²

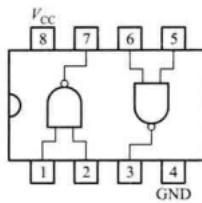


图 3.8.15 小尺寸逻辑封装的 2 输入与非门
面积为 1.4 mm²

小尺寸逻辑器件有5脚封装、6脚封装和8脚封装的系列芯片等等。图3.8.15所示为2输入与非门的单门和双门封装。单门、双门和三门系列分别以1G、2G和3G表示。逻辑电平转换电路的小尺寸逻辑器件以1T或2T分别表示以1位或2位传输。CMOS通用集成电路AHC/AHCT系列、LVC系列、AUC系列和AUP系列都有小尺寸逻辑产品。对于2输入与非门,传统AUC系列的型号为74AUC00,小尺寸逻辑单门封装的型号为74AUC1G00,双门封装的型号为74AUC2G00。74LVC1T45是小尺寸逻辑封装的1位逻辑电平转换器。

例 3.8.2 在小逻辑电路中,有一种广泛应用的可配置多功能门(Configurable Multiple-Function Gate)电路74AUP1G97,如图3.8.16所示(输入端的3个非门具有施密特特性,施密特电路在第9章介绍,这里可将其视为普通非门进行逻辑功能分析)。通过简单的外部连线就可以实现表3.8.2所示的多种功能,试写出不同输入组合时的输出L的表达式。

解:根据电路结构得出其输出与输入的表达式为

$$L = I_1 I_3 + I_2 \bar{I}_3 \quad (3.8.10)$$

将每一组输入代入式(3.8.10),化简得:

(1) $L = AC + B \bar{C}$,2选1数据选择器,C为控制端,A,B为数据输入。数据选择器将在第4章介绍。

(2) $L = AB$,与运算。

(3) $L = A + \bar{C}$,A与 \bar{C} 进行或运算。

(4) $L = A + \bar{C} + \bar{AC}$, \bar{A} 与C进行与非运算。

(5) $L = B \bar{C}$,B与 \bar{C} 进行与运算。

(6) $L = \overline{\overline{B} + C}$, \overline{B} 与 C 进行或非运算。

(7) $L = A + B$, 或逻辑。

(8) $L = \overline{C}$, 非运算。

(9) $L = A$, 缓冲器。

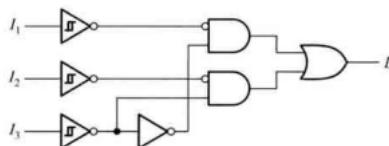


图 3.8.16 例 3.8.2 的小尺寸逻辑 74AUP1G97 内部电路

表 3.8.2 74AUP1G97 实现的逻辑功能

序号	输入连接方式			功 能
	I_1	I_2	I_3	
1	A	B	C	2 选 1 数据选择器
2	A	B	B	2 输入与门
3	A	$+V_{DD}$	C	带一个反相输入的 2 输入或门
4	A	$+V_{DD}$	C	带一个反相输入的 2 输入与非门
5	地	B	C	带一个反相输入的 2 输入与门
6	地	B	C	带一个反相输入的 2 输入或非门
7	A	B	A	2 输入或门
8	地	$+V_{DD}$	C	反相器
9	A	地	$+V_{DD}$	同相缓冲器

2. 宽总线电路

随着计算机、网络、信息传输设备和其他电子系统的发展，要求提高电路的集成度，增加数据传输速度，降低功耗，宽总线电路应运而生。它不仅包含 CMOS，而且包含 TTL 和 BiCMOS 系列的多种产品，其中包括单向和双向总线驱动、多路选择器/分配器、锁存器或触发器等。

宽总线是指将多个相同的单元电路封装在一起，以减少体积、改善电路性能，满足数据总线传输的需求。宽总线驱动器即是将多个三态输出缓冲器（或反相器）集成在一个芯片中。随着总线位数的加宽，出现了 8 位、16 位、32 位等集成电路芯片。例如 16 位总线驱动器 74AUC16240 集成芯片内部有 16 个三态输出缓冲器，分成 4 组。每组由 4 个高电平使能的三态缓冲器和一个输入低有效的使能控制门构成。组与组之间相互独立，如图 3.8.17 所示。74AUC16240 的功能

表如表 3.8.3 所示。使用时可根据需要连接成 16 位、两组 8 位或其他形式,作为总线的数据接收或传输、存储器地址驱动、时钟驱动等。有些总线驱动电路还带有逻辑电平转换功能,可以作为两种工作电源的接口电路。

表 3.8.3 74AUC16240 的功能表

使能 \overline{OE}	输入 A	输出 Y
L	H	L
L	L	H
H	x	高阻

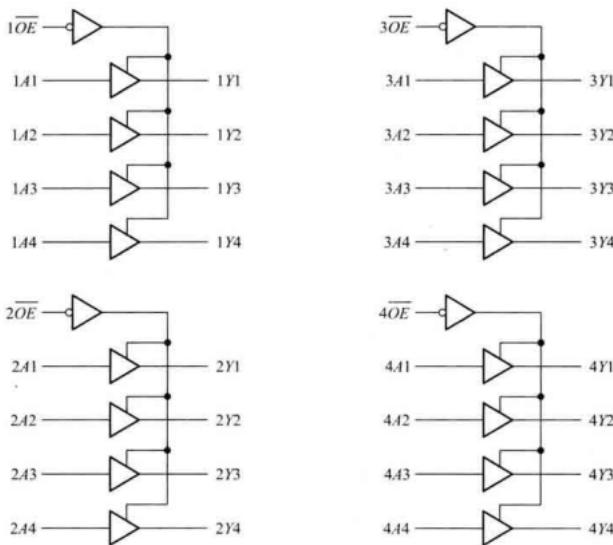


图 3.8.17 宽总线 74AUC16240 逻辑图

复习思考题

- 3.8.1 不同系列逻辑电路相互连接时,需要满足什么条件?
- 3.8.2 当 3.3 V 和 5 V 两种 CMOS 门电路连到同一数据总线时,需要注意什么问题?
- 3.8.3 3.3 V CMOS 电路驱动 5 V CMOS 电路时,如何解决高电平参数不兼容问题?
- 3.8.4 当负载所需的电流比较大时,如何增加驱动电流?

- 3.8.5 为什么 CMOS 电路的多余输入端绝对不能悬空?
 3.8.6 什么是通用型集成电路？什么是专用型集成电路?
 3.8.7 小尺寸逻辑和宽总线电路的特点各是什么？

3.9 用 Verilog HDL 描述 CMOS 门电路

用 Verilog 语言对 MOS 管构成的数字开关逻辑电路建模，常称为开关级建模，这是 Verilog 语言提供的最低层次的描述。由于 Quartus II 软件不支持 Verilog 语言内置的开关级元件，所以本节介绍的开关级电路模型需用其他的通用仿真器（如 ModelSim、Active HDL、Verilog_XL 等）仿真。

3.9.1 CMOS 门电路的 Verilog 建模

为了对数字开关逻辑电路建模，Verilog 提供了 10 多个内置的基本开关元件。关键词 **nmos**、**pmos** 分别定义了最基本的 NMOS 管和 PMOS 管模型，调用时按照下列格式说明它的三个端口信号。

nmos N1 (漏极, 源极, 控制栅极);

pmos P1 (漏极, 源极, 控制栅极);

对 NMOS 元件，如果控制栅极信号为 1，则 NMOS 开关导通，信号能够从管子的源极传输到漏极，如果控制栅极信号为 0，则输出呈现高阻值 z。类似地，如果控制栅极信号为 0 时，PMOS 开关导通，否则，PMOS 开关的输出呈现高阻值 z。由于 **nmos**、**pmos** 是基本元件，故调用名 N1、P1 可以省略。

关键词 **rnmos**、**rpmos** 分别是 NMOS 管和 PMOS 管另一种模型，前面的字母 r 说明 MOS 管的输入端和输出端之间存在着电阻，当信号从 MOS 管的输入传输至输出时，信号的幅度会衰减，它们的用法与 **nmos**、**pmos** 元件相同。

Verilog 语言中用关键词 **supply1**、**supply0** 分别定义了电源线和地线。**supply1** 与电路图中的 V_{dd} 等效，在整个仿真期间将线网置逻辑 1；**supply0** 与电路图中的地线或 V_{ss} 等效，在整个仿真期间将线网置逻辑 0。它们的用法如下：

supply1 Vdd;

supply0 GND;

例 3.9.1 根据图 3.2.17 所示电路，用 Verilog 中开关级建模方法对 CMOS 与非门电路进行描述。

解：在 2 输入 CMOS 与非门中，两个 PMOS 管并联，且源极都与电源 Vdd 连接，漏极都与输出 L 相连，两个管子的栅极分别与输入 A、B 相连；两个 NMOS 管串联，有一个公共节点 W1，第一个 NMOS 管的漏极与输出 L 相连，第二个 NMOS 管的源极与地线 GND 相连，两个 NMOS 管的栅极分别与输入 A、B 相连。电路的描述代码如下：

```
module NAND2(L,A,B); //IEEE 1364—1995 Syntax
```

```



```

3.9.2 CMOS 传输门电路的 Verilog 建模

Verilog 语言中用关键词 **cmos** 定义了基本传输门元件模型,它有一个输出端、一个输入端和两个控制端,如图 3.2.20(a)所示,习惯上也称之为 **cmos** 开关。其用法如下:

cmos C1(输出信号,输入信号, T_n 管控制信号, T_p 管控制信号):

通常 T_n 管控制信号和 T_p 管控制信号彼此是互补的,当 T_n 管控制信号为 1, T_p 管控制信号为 0 时,CMOS 开关导通;如果 T_n 管控制信号为 0, T_p 管控制信号为 1 时,CMOS 开关的输出呈现出高阻抗 z。CMOS 开关的电源和地通常与 MOS 管的衬底相连,故调用 **cmos** 元件时,不需要考虑电源与地的连接问题。调用名 C1 可以省略。

关键词 **r_cmos** 定义了传输门元件另一种模型,字母 r 说明传输门元件的输入端和输出端之间存在着电阻,当信号通过时会造成幅度衰减。注意,Verilog 语言中定义的两种传输门元件是单向传输信号的,与前面介绍的实际传输门元件存在一定差异。

例 3.9.2 根据图 3.2.21 所示电路,用 Verilog 中开关级建模方法对由 CMOS 传输门构成的异或门电路进行描述。

解:程序两次调用在开关级自定义的下层模块 inverter,其调用名为 V1、V2(注意,调用下层模块时,调用名不能省略),完成反相的功能。还调用了两个 **cmos** 开关元件,它们是 Verilog 内置的基本元件,故调用名可以被省略。

```

//CMOS 传输门构成的异或门(参考图 3.2.21)
module mymux2to1(A,B,L); //IEEE 1364—1995 Syntax
    input A,B; //输入端口声明
    output L; //输出端口声明
    wire Anot,Bnot; //声明模块内部的连接线
    inverterV1(Anot,A); //实例化,调用底层模块 inverter
    inverterV2(Bnot,B);

```

```

cmos ( L, Anot, B, Bnot ); //实例化,调用内部开关元件
cmos ( L, A, Bnot, B ); // ( output, input, ncontrol, pcontrol )
endmodule

//CMOS 反相器(参考图 3.2.12)
module inverter( Vo, Vi ); //IEEE 1364—1995 Syntax
    input Vi; //输入端口声明
    output Vo; //输出端口声明
    supply1 Vdd;
    supply0 GND;
    pmos ( Vo, Vdd, Vi ); //实例化,调用内部开关元件
    nmos ( Vo, GND, Vi ); // ( 漏极,源极,控制栅极 )
endmodule

```

小结

- 逻辑门电路是组成各种数字电路的基本单元电路。按照构成门电路元器件的不同,分为MOS型、双极型和混合型。MOS型集成逻辑门有CMOS、NMOS、PMOS,双极型集成逻辑门主要有TTL、ECL,混合型有BiCMOS等。不论哪一种逻辑门电路,作为开关器件的MOS管或BJT管起关键作用。影响它们开关速度的主要因素是器件内部各电极之间的结电容。
- CMOS逻辑门是目前使用最广泛的集成电路,工艺技术的改进使其向着低电压、超低电压和低功耗器件的方向发展。CMOS电路的优点是集成度高,功耗低,扇出数大,噪声容限大,开关速度快。
- CMOS电路的输入和输出具有保护电路和缓冲电路,因此,同一系列不同逻辑功能的电路,具有相同的特性和统一的参数。不同系列的逻辑电路,只要型号最后的数字相同,它们的逻辑功能就一样,但是电气性能参数有所不同。使用时要注意查阅具体的参数。从输出端的结构看,CMOS电路有普通缓冲器输出、漏极开路输出或三态输出电路。
- 类NMOS逻辑门电路结构简单,几何尺寸小,易于集成化,在某些特殊应用中更具优势。BiCMOS电路结合了MOS电路和TTL电路的优势,其开关速度较快,功耗亦较低。
- CMOS系列已取代TTL系列成为数字电路和系统的首选。在某些特殊场合及BiCMOS电路中,可能会用到TTL门电路。TTL门电路由若干BJT和电阻组成。输出级采用推拉式结构可以提高开关速度,增强带负载能力。利用肖特基二极管构成抗饱和TTL电路,可以提高开关速度。
- ECL逻辑门电路是以差分放大电路为基础的,电路中的BJT不工作在饱和区,因而开关速度较快。其缺点是功耗较大,噪声容限低。

- 在逻辑体制中有正、负逻辑的规定，本书主要采用正逻辑体制。逻辑门的等效符号常用于简化电路的分析和设计。
- 在逻辑门电路的实际应用中，有可能遇到不同系列门电路之间、门电路与负载之间的接口技术问题，以及抗干扰工艺问题。数字电路设计工作者应能正确分析与解决这些问题。
- 用 Verilog 可以对 MOS 管构成逻辑门电路描述，即开关级建模。

习 题

3.1 逻辑门电路简介

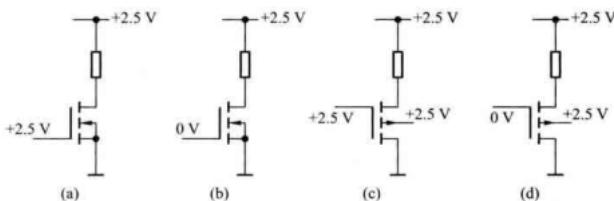
3.1.1 按照制造门电路晶体管的不同，集成门电路分为哪几种类型？各种类型的代表是什么？

3.1.2 为什么要发展低电压和超低电压 CMOS 器件？

3.1.3 数字逻辑变量可以取什么值？晶体管在数字电路中工作在什么状态？

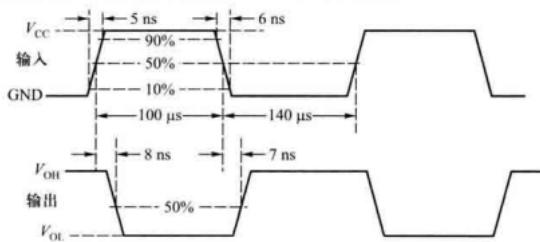
3.2 基本 CMOS 逻辑门电路

3.2.1 已知图题 3.2.1 所示各 MOS 管的 $|V_T| = 0.5$ V，忽略电阻上的压降，试分别确定它们的工作状态（导通或截止）。



图题 3.2.1

3.2.2 一个反相器的输入和输出波形如图题 3.2.2 所示，试确定：



图题 3.2.2

(1) 输入信号的周期和频率；(2) 输入信号的上升时间 t_r 和下降时间 t_f ；(3) 输出由高电平变为低电平的

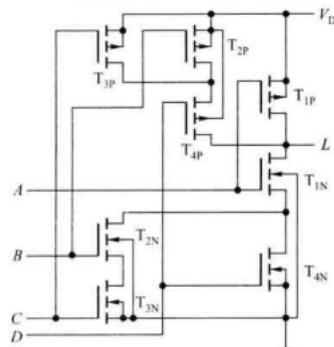
传输延迟时间 t_{pH} 和由低电平变为高电平的传输延迟时间 t_{pLH} 。

3.2.3 为什么说 74LVC 系列 CMOS 与非门在+3.3 V 电源工作时,输入端在以下四种接法下都属于逻辑 0 (74LVC 系列输入、输出电压值参考表 3.3.2):

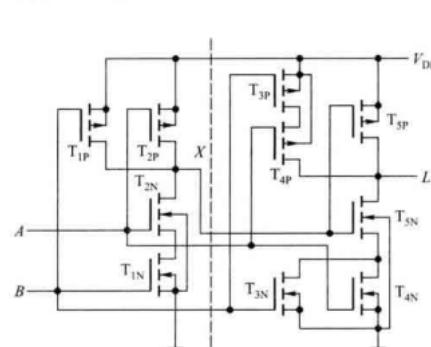
(1) 输入端接地; (2) 输入端接低于 0.8 V 的电源; (3) 输入端接同类与非门的输出低电平 0.2 V; (4) 输入端到地之间接 10 kΩ 的电阻。

3.2.4 试分析图题 3.2.4 所示电路,写出 L 的逻辑表达式。

3.2.5 试分析图题 3.2.5 所示的电路,写出 L 的逻辑表达式,说明它实现什么逻辑功能。



图题 3.2.4

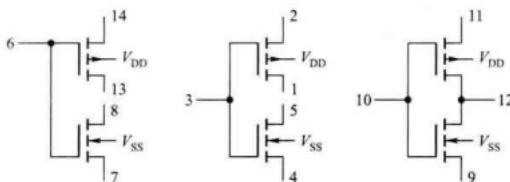


图题 3.2.5

3.2.6 试画出实现下列逻辑功能的 CMOS 电路图。

$$(1) L = \overline{AB+CD} \quad (2) L = \overline{(A+B)(C+D)}$$

3.2.7 CMOS 集成芯片 4007 中包含两个互补对和一个反相器,如图题 3.2.7 所示,试分别连接:(1) 三个反相器;(2) 3 输入端或非门;(3) 3 输入端与非门;(4) 或与非门 [$L = \overline{C(A+B)}$] ;(5) 传输门(一个非门控制两个传输门分时传送)。



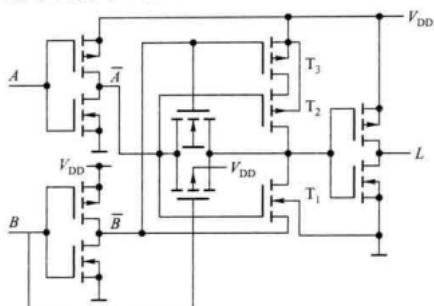
图题 3.2.7

3.2.8 试分析图题 3.2.8 所示 CMOS 门电路,写出 L 的逻辑表达式,说明它实现什么逻辑功能。

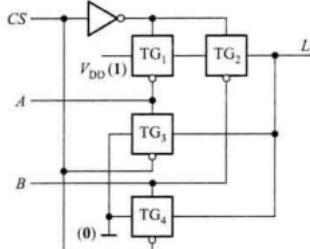
3.2.9 试用两个传输门和两个反相器实现同或逻辑功能,并画出逻辑电路。

3.2.10 由 CMOS 传输门构成的电路如图题 3.2.10 所示,CS 为控制端,A,B 为输入,L 为输出,试列出其真

值表,说明该电路的逻辑功能。



图题 3.2.8



图题 3.2.10

3.3 CMOS 逻辑门电路的不同输出结构及参数

3.3.1 在漏极开路门电路中,输出上拉电阻值的大小对逻辑门有什么影响?

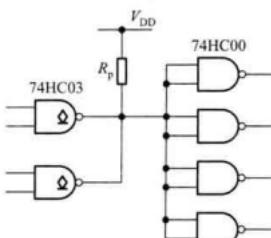
3.3.2 试判断下列哪些 CMOS 门可以将输出端并接使用:(1) 普通的互补输出;(2) 漏极开路输出;(3) 三态输出。

3.3.3 试用普通反相器和 OD 输出反相器实现下列表达式,并画出逻辑电路图。

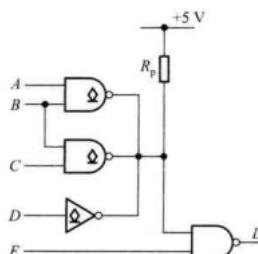
$$(1) L = A \bar{B} C; (2) L = AB \bar{C} \bar{D}$$

3.3.4 用 74HC03 中 2 个漏极开路与非门及 74HC00 中的 4 个与非门构成的电路如图题 3.3.4 所示。试确定上拉电阻 R_p 的取值范围。已知 $V_{DD} = 5$ V, OD 门输出低电平 $V_{OL(max)} = 0.33$ V 时的输出电流 $I_{OL(max)} = 4$ mA, 输出高电平 $V_{OH(min)} = 4.4$ V 时的漏电流 $I_{OZ} = 5 \mu\text{A}$ 。负载门高电平和低电平输入电流最大值 $I_{H(max)} = I_{L(max)} = 1 \mu\text{A}$ 。

3.3.5 用 74HC03 中三个漏极开路与非门和 74LS00 中一个 TTL 与非门实现图题 3.3.5 所示的电路,试写出输出与输入的逻辑关系式,并计算 $R_{p(min)}$ 和 $R_{p(max)}$ 。已知驱动门输出低电平 $V_{OL(max)} = 0.33$ V 时的最大电流 $I_{OL(max)} = 4$ mA, 输出高电平 $V_{OH(min)} = 4.4$ V 时的漏电流 $I_{OZ} = 5 \mu\text{A}$ 。负载门高电平和低电平输入电流分别为 $I_{Hl} = 0.02 \text{ mA}$, $I_{Ll} = 0.4 \text{ mA}$ 。



图题 3.3.4

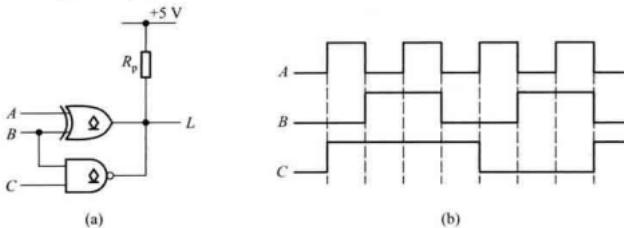


图题 3.3.5

3.3.6 由 OD 异或门和 OD 与非门构成的电路及输入电压波形如图题 3.3.6 所示。

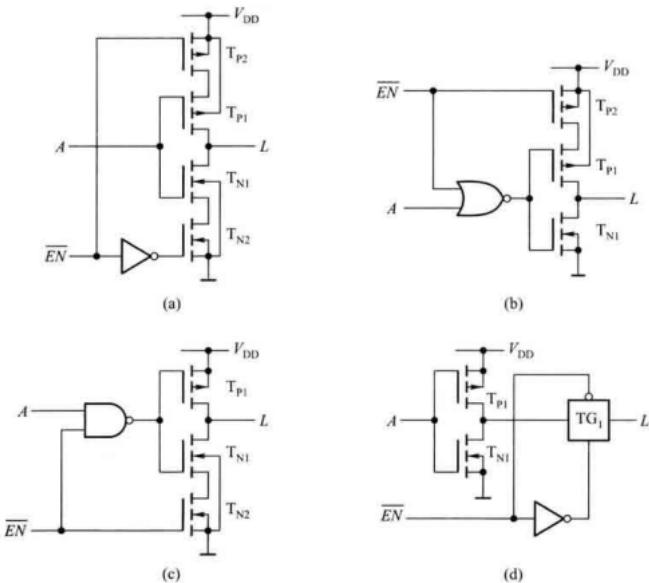
(1) 试写出输出与输入的逻辑关系式,画出输出电压波形。

(2) 已知输出低电平 $V_{OL(max)} = 0.33$ V 时的最大输出电流 $I_{OL(max)} = 4$ mA, 输出高电平 $V_{OH(min)} = 4.4$ V 时的漏电流 $I_{OZ} = 5$ μ A, 计算 $R_{p(min)}$ 和 $R_{p(max)}$ 。



图题 3.3.6

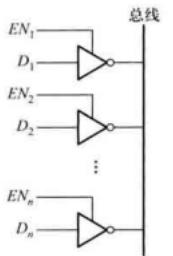
3.3.7 试分析图题 3.3.7 所示的 CMOS 电路,说明它们的逻辑功能。



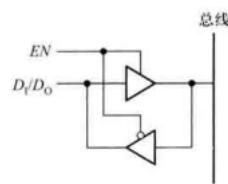
图题 3.3.7

3.3.8 图题 3.3.8 表示三态门用于总线传输的示意图,图中 n 个三态门的输出连接到数据传输总线, D_1 、 D_2 、 \dots 、 D_n 为数据输入端, EN_1 、 EN_2 、 \dots 、 EN_n 为使能信号输入端。试问:(1) 如何控制 EN 信号,以便数据 D_1 、 D_2 、 \dots 、 D_n 能通过该总线进行正常传输? (2) EN 信号能否有两个或两个以上同时有效? 如果 EN 出现两个或两个以上有效,可能发生什么情况? (3) 如果所有 EN 信号均无效,总线处在什么状态? (4) 如果三态门的开通比断开要快,可能发生什么情况?

3.3.9 三态门与总线的连接方式如图题 3.3.9 所示,试分析电路的逻辑功能。



图题 3.3.8



图题 3.3.9

3.3.10 根据表题 3.3.10 所列的三种逻辑门电路的技术参数,试选择一种最适合工作在高噪声环境下的门电路。

表题 3.3.10 逻辑门电路的技术参数表

	$V_{OH(\min)}/V$	$V_{OL(\max)}/V$	$V_{IH(\min)}/V$	$V_{IL(\max)}/V$
逻辑门 A	2.4	0.4	2	0.8
逻辑门 B	3.5	0.2	2.5	0.6
逻辑门 C	4.2	0.2	3.2	0.8

3.3.11 CMOS 反相器的负载电容 $C_L = 100 \text{ pF}$,功耗电容 $C_{pp} = 15 \text{ pF}$,电源电压 $V_{DD} = 3.3 \text{ V}$,输入矩形波的频率为 1 MHz,试计算反相器的动态功耗。

3.3.12 当 74 HC 系列 CMOS 电路的电源电压由 5 V 降至 3.3 V,其功耗下降了多少百分比?

3.3.13 根据表题 3.3.13 所列的三种逻辑门电路的技术参数,计算出它们的延时-功耗积,并指出哪一种逻辑门的性能最好。

表题 3.3.13 逻辑门电路的技术参数表

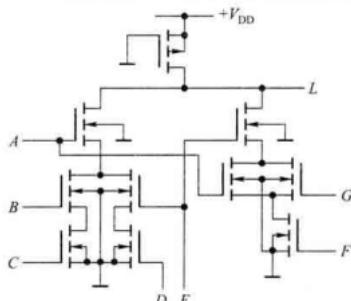
	t_{PH}/ns	t_{PHL}/ns	P_D/mW
逻辑门 A	1	1.2	16
逻辑门 B	5	6	8
逻辑门 C	10	10	1

3.3.14 根据表 3.3.4 和表 3.5.1 所示的电流值,求下列情况下逻辑门的扇出数:(1) 74LVC 门驱动同类门 (2) 74AHCT 门驱动 74ALS 门。

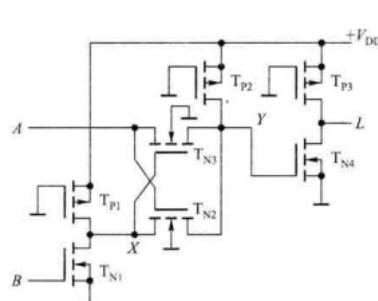
3.4 类 NMOS 和 BiCMOS 逻辑门电路

3.4.1 写出图题 3.4.1 所示电路的逻辑表达式。

3.4.2 写出图题 3.4.2 所示电路的逻辑表达式。



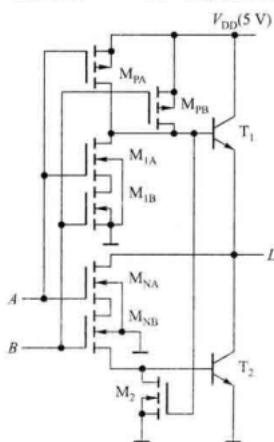
图题 3.4.1



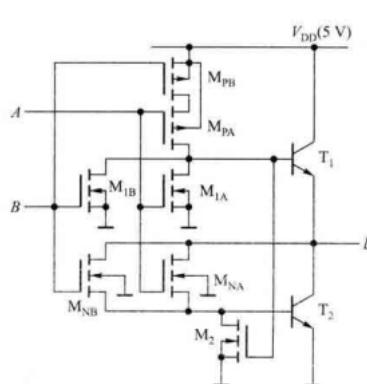
图题 3.4.2

3.4.3 图题 3.4.3 所示为 2 输入端 BiCMOS 与非门电路,试分析该电路是怎样实现与非逻辑关系($L = \overline{A \cdot B}$)的。

3.4.4 分析图题 3.4.4 所示电路的工作原理,写出 L 的逻辑表达式。



图题 3.4.3



图题 3.4.4

3.5 TTL 逻辑门电路

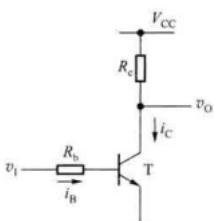
3.5.1 由BJT构成的反相器如图题3.5.1所示, $V_{CC} = +5\text{ V}$, $V_{BE} = 0.7\text{ V}$, $\beta = 100$ 。当输入 v_1 为 5 V 时, 输出为 0.2 V, 试计算 R_b/R_c 的最大比值。

3.5.2 为什么说 TTL 与非门的输入端在以下四种接法下,都属于逻辑 1:

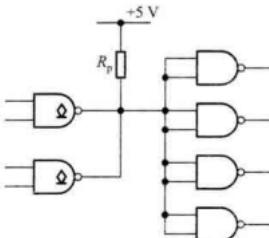
- (1) 输入端悬空; (2) 输入端接高于 2 V 的电源; (3) 输入端接同类与非门的输出高电平 3.6 V; (4) 输入端到地之间接 10 kΩ 的电阻。

3.5.3 设有一个 74ALS04 反相器驱动两个 74ALS04 反相器和 4 个 74LS04 反相器。(1) 驱动门是否超载? (2) 若超载,试提出改进方案;若未超载,则还可增加几个 74LS04 门?(器件参数参考表 3.5.1)。

3.5.4 图题 3.5.4 所示为集电极开路门驱动 4 个 TTL 逻辑门。已知 OC 门输出管截止时的漏电流 $I_{OZ} = 0.1\text{ mA}$, 输出低电平 $V_{OL(\min)} = 0.5\text{ V}$ 时的输出电流 $I_{OL(\max)} = 8\text{ mA}$, 负载门的电流参数为: $I_{IL} = 0.5\text{ mA}$, $I_{IH} = 0.02\text{ mA}$ 。要求输出高电平 $V_{OH} \geq 3\text{ V}$, 试计算上拉电阻 R_p 的值。



图题 3.5.1



图题 3.5.4

3.5.5 将题 3.5.4 中的负载门全部改成 2 输入或非门, 其他参数不变, 重新计算上拉电阻 R_p 的值。

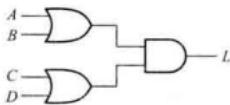
3.6 ECL 逻辑门电路

3.6.1 某 ECL 门电路在 25°C 时的参数为: $V_{IL(\max)} = -1.475\text{ V}$, $V_{IH(\min)} = -1.105\text{ V}$, $V_{OL(\max)} = -1.630\text{ V}$, $V_{OH(\min)} = -0.980\text{ V}$ 。试计算其噪声容限。

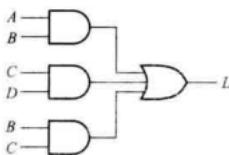
3.7 逻辑描述中的几个问题

3.7.1 试对图题 3.7.1 所示电路的逻辑门进行变换, 使其可以用单一的或非门实现。

3.7.2 电路如图题 3.7.2 所示, 试用与非门实现。



图题 3.7.1



图题 3.7.2

3.8 逻辑门电路使用中的几个实际问题

3.8.1 当供电电源不同的两种门电路相互连接时, 要考虑哪些电压和电流参数? 这些参数应满足怎样的

关系?

3.8.2 若只考虑逻辑电平的兼容性,根据表3.3.4所示的参数值,当用5V供电的74AHC00驱动3V供电的74LVC00时,是否需要接口?计算其扇出数。当用74LVC00驱动74AHC00时,是否需要接口?

3.8.3 根据图3.8.1所示的逻辑电平值,计算3.3V CMOS系列驱动5V TTL系列的高、低电平噪声容限(假设3.3V CMOS系列能承受5V电压)。

3.8.4 根据图3.8.1所示的逻辑电平值,分别计算3.3V CMOS系列驱动2.5V CMOS系列或2.5V CMOS系列驱动3.3V CMOS系列的高、低电平噪声容限(假设2.5V CMOS系列输入输出均能承受3.3V电压)。

3.8.5 根据图3.8.1所示的逻辑电平值,分别计算3.3V或2.5V CMOS系列驱动1.8V CMOS系列的高、低电平噪声容限(假设1.8V CMOS系列输入可承受3.3V电压)。

3.8.6 复习一下CMOS门的输出电路。若CMOS的输出级超载时,电路会出现什么现象?用什么仪器进行判断?

3.8.7 根据表3.3.4和表3.5.1所示参数,判断用74LS系列TTL电路去驱动74HC系列CMOS电路时,试简述其设计思路,判断是否需要接口电路,计算其扇出数,并对接口电路就开关速度和功耗两方面作出评价(设用一个74LS逻辑门作为驱动器件,并且它输出高电平时的漏电流为0.2mA)。

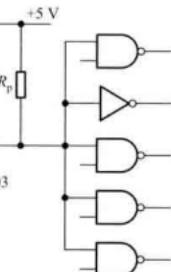
3.8.8 当用74ALS系列TTL去驱动74HC系列CMOS时。重复题3.8.7。

3.8.9 图题3.8.9所示为集电极开路门74LS03驱动5个CMOS逻辑门。已知OC门输出管截止时的漏电流 $I_{OZ} = 0.2\text{ mA}$;负载门的参数为: $V_{H(\min)} = 4\text{ V}$, $V_{H(\max)} = 1\text{ V}$, $I_H = I_M = 1\text{ }\mu\text{A}$ 。试计算上拉电阻的值。

3.8.10 根据表3.3.4和表3.5.1所示参数,用74HC系列CMOS去驱动74LS系列TTL门电路时,试简述其设计思路,指出是否需要加接口电路,并就开关速度和功耗两方面对接口电路进行评价。

3.8.11 当用74HC系列CMOS驱动ALS系列TTL时,重复题3.8.10。

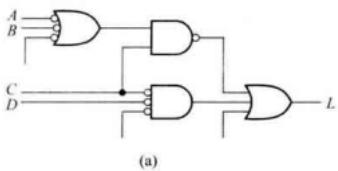
3.8.12 设计一发光二极管(LED)驱动电路,设LED的参数为 $V_F = 2.5\text{ V}$, $I_F = 4.5\text{ mA}$;若 $V_{CC} = 5\text{ V}$,当LED发光时,电路的输出为低电平,选用集成门电路的型号,并画出电路图。



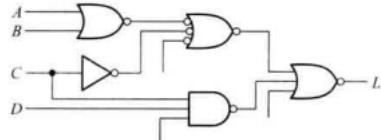
图题3.8.9

3.8.13 由CMOS门构成的电路如图题3.8.13所示,试对多余输入端进行处理,并写出 L 的逻辑表达式。

3.8.14 由TTL门构成的电路如图题3.8.13时,重复题3.8.13。



(a)



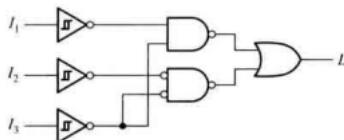
(b)

图题3.8.13

3.8.15 可配置多功能门电路74LVC1G57的内部结构如图题3.8.15所示(输入端的3个非门具有施密特特性,施密特电路在第9章介绍,这里可将其视为普通非门进行逻辑功能分析)。根据需要将 I_1 , I_2 , I_3 接输入变

量、地或 $+V_{cc}$ 就可以实现不同逻辑功能,试确定输入端的连接方式以便分别实现:(1) $L = AB$;(2) $L = \overline{A+B}$;

$$(3) L = AB + \overline{A} \overline{B}$$



图题 3.8.15

3.9 用 VerilogHDL 描述 CMOS 门电路

3.9.1 试用 Verilog 提供的基本开关元件对图 3.2.18 所示的或非门电路进行描述。

3.9.2 试用 Verilog 提供的基本开关元件对图 3.2.19 所示的异或门电路进行描述。

4 >>>

组合逻辑电路

引言

数字系统中常用的各种数字部件,就其结构和工作原理而言可分为两大类,即组合逻辑电路和时序逻辑电路。本章首先介绍组合逻辑电路的定义、分析方法和设计方法,并阐述竞争-冒险产生的原因及消除方法。然后讨论常用的几种组合逻辑电路的功能及基本应用,它们包括编码器和译码器、数据选择器和数据分配器、数值比较器、算术运算单元等。最后介绍组合逻辑电路的 Verilog HDL 描述以及用可编程逻辑器件 PLD 的实现方法。

4.1 组合逻辑电路的分析

4.1.1 组合逻辑电路的定义

对于一个逻辑电路,其输出状态在任何时刻只取决于同一时刻的输入状态,而与电路原来的状态无关,这种电路被定义为组合逻辑电路。组合逻辑电路的一般框图如图 4.1.1 所示,其输出与输入之间的逻辑关系可用如下的逻辑函数来描述,即

$$L_i = f(A_1, A_2, \dots, A_n) \quad (i=1, 2, \dots, m) \quad (4.1.1)$$

式中, A_1, A_2, \dots, A_n 为输入变量。

组合逻辑电路的结构具有如下的特点:

- (1) 输出、输入之间没有反馈延迟通路;
- (2) 电路中不含具有记忆功能的元件。

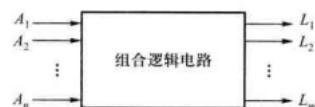


图 4.1.1 组合逻辑电路的一般框图

4.1.2 组合逻辑电路的分析方法

分析组合逻辑电路的目的是,对于一个给定的逻辑电路,确定其逻辑功能。分析组合逻辑电

路的步骤大致如下：

- (1) 根据逻辑电路,从输入到输出,写出各级逻辑函数表达式,直到写出输出信号与输入信号的逻辑函数表达式;
- (2) 将各逻辑函数表达式化简和变换,以得到最简单的表达式;
- (3) 根据简化后的逻辑表达式列出真值表;
- (4) 根据真值表和简化后的逻辑表达式对逻辑电路进行分析,最后确定其功能。

下面举例说明组合逻辑电路的分析方法。

例 4.1.1 已知逻辑电路如图 4.1.2 所示,分析该电路的功能。

解: (1) 根据逻辑电路可写出输出端的逻辑函数表达式,为方便起见,电路中标出了中间变量 Z 。电路由两个异或门构成,因此

$$\begin{aligned} Z &= A \oplus B \\ L &= Z \oplus C = (A \oplus B) \oplus C \end{aligned}$$

该表达式无须化简和变换。

(2) 列写真值表。将 3 个输入变量的 8 种可能的组合一一列出。分别将每一组变量的取值代入逻辑函数表达式,然后算出中间变量 Z 值和输出 L 值,填入表中,如表 4.1.1 所示。

表 4.1.1 例 4.1.1 的真值表

A	B	C	Z	L
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	1	0
1	0	0	1	1
1	0	1	1	0
1	1	0	0	0
1	1	1	0	1

(3) 确定逻辑功能。分析真值表后可知,当 A, B, C 三个输入变量的取值中有奇数个 1 时, L 为 1, 否则 L 为 0。所以该电路称为奇校验电路,用于检查 3 位二进制码的奇偶性。

如果在上述电路的输出端再加一级反相器,当输入电路的二进制码中含有偶数个 1 时,输出为 1,则称此电路为偶校验电路。

波形图可以比较直观地反映输入与输出之间的逻辑函数关系,对于比较简单的组合逻辑电路,也可用画波形图的方法进行分析。为了避免出错,通常是根据输入波形的变化分段,然后逐段画出输出波形,例如,第一段的输入信号 A, B, C 的值均为 0,代入表达式中得出 Z 的值,然后

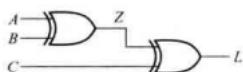


图 4.1.2 例 4.1.1 的逻辑电路

得出 L 的值,如图 4.1.3 所示。最后根据波形图确定输入和输出的逻辑功能。

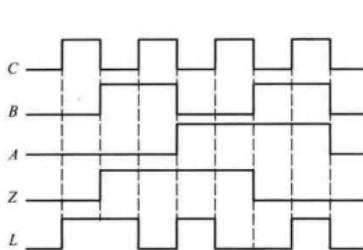


图 4.1.3 例 4.1.1 的波形分析图

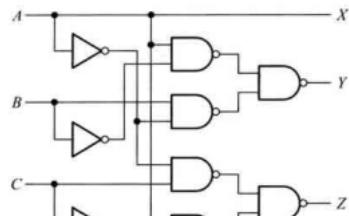


图 4.1.4 例 4.1.2 的逻辑电路

例 4.1.2 试分析图 4.1.4 所示组合逻辑电路的逻辑功能。

解: (1) 根据逻辑电路可写出各输出端的逻辑表达式,并进行化简和变换。

$$X = A$$

$$\overline{Y} = \overline{AB} \cdot \overline{AB} = \overline{AB} + \overline{AB}$$

$$\overline{Z} = \overline{AC} \cdot \overline{AC} = \overline{AC} + \overline{AC}$$

(2) 列写真值表,如表 4.1.2 所示。

表 4.1.2 例 4.1.2 的真值表

输入			输出		
A	B	C	X	Y	Z
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	1	1
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	1	0	0

(3) 确定逻辑功能。分析真值表可知,输出最高位 X 与输入最高位 A 相等。当 A 为 0 时,输出 Y, Z 分别与所对应的输入 B, C 相同;而当 A 为 1 时,输出 Y, Z 分别是输入 B, C 取反。这个电

路逻辑功能是对输入的二进制码求反码。最高位为符号位, **0** 表示正数, **1** 表示负数, 正数的反码与原码相同; 负数的数值部分是在原码的基础上逐位求反。

复习思考题

- 4.1.1 什么是组合逻辑电路?
- 4.1.2 列出分析组合逻辑电路的步骤。
- 4.1.3 组合逻辑电路都能用波形分析法进行分析吗?

4.2 组合逻辑电路的设计

组合逻辑电路的设计与分析过程相反, 对于提出实际逻辑问题, 得出满足这一逻辑问题的逻辑电路。电路设计的首要任务是满足功能要求, 其次是优化, 即用指定的器件实现逻辑函数时, 力求成本低并且工作速度快。前面介绍的用代数法和卡诺图法来化简逻辑函数, 是为了获得最简的逻辑表达式。根据最简表达式获得最低成本电路。电路的实现可以采用各种门电路, 也可以利用数据选择器等基本模块, 或者可编程逻辑器件。因此, 逻辑函数的化简也要结合所选用的器件进行, 有时还需要一定的变换, 以便满足优化实现的要求。

4.2.1 组合逻辑电路的设计过程

组合逻辑电路的设计步骤大致如下:

(1) 明确实际问题的逻辑功能。许多实际设计要求是用文字描述的, 因此, 需要确定实际问题的逻辑功能, 并确定输入、输出变量数及表示符号。

(2) 根据对电路逻辑功能的要求, 列出真值表;

(3) 由真值表写出逻辑表达式;

(4) 简化和变换逻辑表达式, 从而画出逻辑图。

下面举例说明设计组合逻辑电路的方法和步骤。

例 4.2.1 某火车站有特快、直快和慢车三种类型的客运列车进出, 试设计一个指示列车等待进站的逻辑电路。当有两种或以上的列车等待进站时, 要求发出信号, 提示工作人员安排列车进站事宜。

解: (1) 明确逻辑功能。

设变量 A, B, C 分别表示特快、直快和慢车, 并规定有进站请求时为 **1**, 没有请求时为 **0**。输出变量 L 表示进站状况, 有两种或以上的列车等待进站时, L 为 **1**, 否则为 **0**。

根据题意列出真值表, 如表 4.2.1 所示。

(2) 根据真值表写出输出逻辑表达式。

根据 1.6 节介绍的方法,逻辑变量之间是与的关系,而输出状态之间的组合则是或的关系。对于输入或输出变量,凡取 1 值的用原变量表示,取 0 值用反变量表示。则

$$L = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + A\bar{B}\bar{C} + ABC \quad (4.2.1)$$

(3) 简化逻辑表达式。

用公式法或卡诺图化简法对式(4.2.1)化简,得

$$L = AB + AC + BC \quad (4.2.2)$$

式(4.2.2)为最简与-或式,用与门和或门实现两级与-或结构的最简电路如图 4.2.1 所示。



图 4.2.1 例 4.2.1 逻辑图

表 4.2.1 例 4.2.1 的真值表

输入			输出
A	B	C	L
0	0	0	0
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

在不同的数字系统中,可能采用不同的码制对信息进行编码和处理。如果在采用不同码制的两个数字系统之间进行信息传输,则需要一个码转换电路,以保证两者之间的相互匹配。

例 4.2.2 试设计一个码转换电路,将 4 位格雷码转换为自然二进制码。可以采用任何逻辑门电路来实现。

解: (1) 明确逻辑功能,列出真值表。

设电路的 4 个输入变量为 G_3, G_2, G_1 和 G_0 ,4 个输出变量为 B_3, B_2, B_1 和 B_0 。当输入格雷码按照从十进制数 0 到 15 递增排序时,对应输出的自然二进制码如表 4.2.2 所示。

(2) 画出各输出函数的卡诺图,如图 4.2.2 所示。如果按照表 4.2.2 所示从上到下的顺序,填写卡诺图的顺序依次为:第一行从左到右,第二行从右到左,第三行从左到右,第四行从右到左。如 B_1 的第一行从左到右为 0011,第二行从右到左为 0011,以此类推。由卡诺图写出各输出逻辑表达式,并化简和变换,如式(4.2.3)所示。

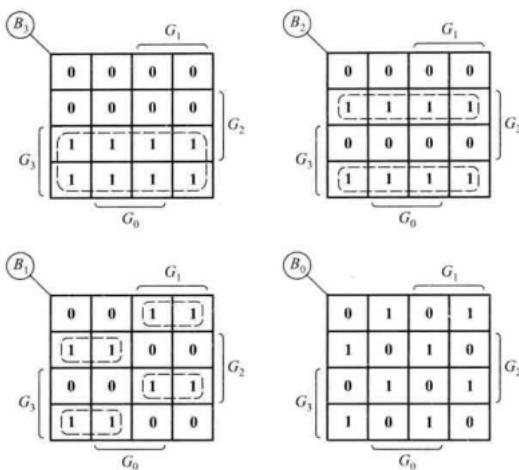


图 4.2.2 例 4.2.2 的卡诺图

表 4.2.2 例 4.2.2 的真值表

十进制数	输入				输出			
	G ₃	G ₂	G ₁	G ₀	B ₃	B ₂	B ₁	B ₀
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	1	0	0	1	0
3	0	0	1	0	0	0	1	1
4	0	1	1	0	0	1	0	0
5	0	1	1	1	0	1	0	1
6	0	1	0	1	0	1	1	0
7	0	1	0	0	0	1	1	1
8	1	1	0	0	1	0	0	0
9	1	1	0	1	1	0	0	1
10	1	1	1	1	1	0	1	0
11	1	1	1	0	1	0	1	1
12	1	0	1	0	1	1	0	0
13	1	0	1	1	1	1	0	1
14	1	0	0	1	1	1	1	0
15	1	0	0	0	1	1	1	1

$$\left\{ \begin{array}{l} B_3 = G_3 \\ B_2 = G_3 \overline{G_2} + \overline{G_3} G_2 \\ = G_3 \oplus G_2 \\ = B_3 \oplus G_2 \\ B_1 = G_3 \overline{G_2} \overline{G_1} + \overline{G_3} G_2 \overline{G_1} + G_3 G_2 G_1 + \overline{G_3} \overline{G_2} G_1 \\ = (G_3 \overline{G_2} + \overline{G_3} G_2) \overline{G_1} + (\overline{G_3} \overline{G_2} + \overline{G_3} G_2) G_1 \\ = G_3 \oplus G_2 \oplus G_1 \\ = B_2 \oplus G_1 \\ B_0 = \overline{G_3} \overline{G_2} \overline{G_1} G_0 + \overline{G_3} \overline{G_2} G_1 \overline{G_0} + \overline{G_3} G_2 \overline{G_1} \overline{G_0} + \overline{G_3} G_2 G_1 G_0 \\ + G_3 G_2 \overline{G_1} G_0 + G_3 G_2 G_1 \overline{G_0} + G_3 \overline{G_2} \overline{G_1} \overline{G_0} + G_3 \overline{G_2} G_1 G_0 \\ = G_3 \oplus G_2 \oplus G_1 \oplus G_0 \\ = B_1 \oplus G_0 \end{array} \right. \quad (4.2.3)$$

因此, n 位格雷码 $G_{n-1} G_{n-2} G_{n-3} \dots G_0$ 转换为 n 位二进制码 $B_{n-1} B_{n-2} B_{n-3} \dots B_0$ 的公式为

$$\left\{ \begin{array}{l} B_{n-1} = G_{n-1} \\ B_{n-2} = B_{n-1} \oplus G_{n-2} \\ B_{n-3} = B_{n-2} \oplus G_{n-3} \\ \vdots \\ B_0 = B_1 \oplus G_0 \end{array} \right. \quad (4.2.4)$$

以同样的方法, 也可以得到 n 位二进制码 $B_{n-1} B_{n-2} B_{n-3} \dots B_0$ 转换为 n 位格雷码 $G_{n-1} G_{n-2} G_{n-3} \dots G_0$ 的公式为

$$\left\{ \begin{array}{l} G_{n-1} = B_{n-1} \\ G_{n-2} = B_{n-1} \oplus B_{n-2} \\ G_{n-3} = B_{n-2} \oplus B_{n-3} \\ \vdots \\ G_0 = B_1 \oplus B_0 \end{array} \right. \quad (4.2.5)$$

(3) 根据式(4.2.3)的逻辑表达式, 可画出逻辑图如图 4.2.3 所示。

从以上逻辑表达式和逻辑图可以看出, 用异或门代替与门和或门能使逻辑电路比较简单。在化简和变换逻辑表达式时, 注意综合考虑, 使各式中的相同乘积项尽可能多, 使某些输出作为另一些逻辑门的输入。例如, 利用 B_2 作为 B_1 的一个输入, B_1 又作为 B_0 的一个输入, 这样可以减少门电路的数目, 降低实现电

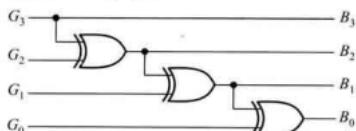


图 4.2.3 例 4.2.2 的逻辑图

路的成本。

4.2.2 组合逻辑电路的优化实现

逻辑函数的优化实现是用指定芯片中特定资源实现逻辑函数,使电路的成本低并且工作速度快。优化实现可以使用基本逻辑门电路或由门电路构成的常用组合逻辑电路模块,这些模块可以作为EDA工具的库元件,由规模更大的电路调用。采用的芯片类型不同,实现电路的优化方式也不同。手工设计过程中由设计者进行优化,自动设计过程中由EDA工具实现优化。EDA工具的优化策略和实现过程庞大而复杂,由计算机在后台完成。对用户而言有许多可选择的特性和控制选项,因此,我们要了解EDA工具所做的工作。作为入门,这一节简单介绍EDA工具实现优化设计时用到的一些技术。

前面介绍的组合逻辑电路的设计过程中,对满足功能要求的逻辑函数进行代数法或卡诺图法进行化简,通常得到最简与-或式。可以用与门和或门构成最简的两级与-或结构电路。但实际电路设计不仅要满足功能要求,还要根据所指定的芯片进行优化实现,并且确保能够物理实现。因此要对逻辑函数的最简与-或式进行变换。变换的宗旨是在满足设计要求的前提下,减少所用芯片资源的数目和连线,使电路得到简化。

如果使用基本门电路为优化资源,需要将最简与-或式转换成与指定门电路相适应的形式。如使用与非门,需要将逻辑函数转换成与非-与非式。各种常用组合逻辑电路的输出与输入的关系都可以写成一个逻辑函数式,不同功能的电路模块,逻辑函数式的形式也不同。CPLD实现逻辑函数是基于与-或式和输出宏单元。FPGA实现逻辑函数是基于查找表(Look-up table,LUT),即数据选择器。基于组合逻辑电路模块及可编程逻辑器件的优化实现在随后的章节介绍。

1. 单输出电路

以基本门电路作为可用资源,用两级与-或结构实现最简函数

$$L = AB + CD \quad (4.2.6)$$

得到如图4.2.4(a)所示电路。如果要求用与非门优化实现该逻辑电路,需要对式(4.2.6)所示逻辑函数式进行变换,用两次求反的方法得到

$$\begin{aligned} L &= AB + CD \\ &= \overline{\overline{AB}} \cdot \overline{\overline{CD}} \end{aligned} \quad (4.2.7)$$

由此可画出逻辑图,如图4.2.4(b)所示。相同输入端的与非门相对与门或者或门而言,所用晶体管少,速度快,实际中多选择使用与非门。或非门相对于或门和与门而言速度快。

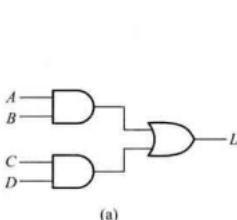
2. 多输出电路

在实际的数字系统中,通常都有多个逻辑函数输出,化简时需要将多个输出函数作为整体考虑,使各式中的相同乘积项尽可能多,以减少门的个数。如两个逻辑函数

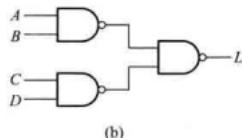
$$L_1 = AB + \overline{AC} + BD$$

$$L_2 = AB + \bar{A}C + ABD$$

如果直接分别实现这两个逻辑函数,需要 6 个与门和两个或门(不考虑非门)。如果考虑共享相同乘积项则只需要 4 个与门和两个或门,逻辑电路如图 4.2.5 所示。



(a)



(b)

图 4.2.4 以门电路为资源优化实现逻辑函数

(a) 与门和或门结构 (b) 与非门结构

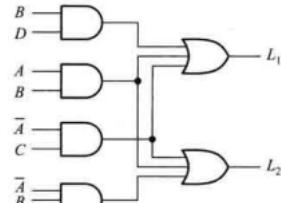


图 4.2.5 共享相同乘积项的逻辑电路

3. 多级逻辑电路

逻辑门电路的优化实现除了要考虑降低成本外,还必须保证能够实现。前面介绍的单输出或多输出都可以用两级与-或结构实现。当输入变量数增加时,逻辑门输入端的数目将增多。如果所需逻辑门输入端数超过物理实现的可能,则需要变换逻辑表达式,以减少逻辑门的扇入数。变换方法可以是提取公因子或函数分解等。

(1) 提取公因子

对于逻辑函数

$$L = ABCD + AB\bar{C}E + ABDF$$

用与-或两级电路实现上述函数需要 3 个 4 输入与门和一个 3 输入或门,如图 4.2.6 所示。如果限定逻辑门的扇入数为 3,需要变换逻辑表达式,提取公因子得

$$L = AB(CD + \bar{C}E + DF)$$

实现上式逻辑门的最大扇入数为 3,满足限定条件。但电路的级数增加到 3 级,如图 4.2.7 所示。

表达式中每个变量对应一根信号线。提取公因子减少了连线的数目,因此减少了逻辑电路连线的复杂度。

(2) 函数分解

对于多变量逻辑函数,减少扇入数除了提取公因子外,还可以用函数分解方法减少子函数的变量数目。

对于逻辑函数

$$L = \bar{A}\bar{B}C + \bar{A}\bar{B}\bar{C} + ABD + \bar{A}\bar{B}D$$

用与-或两级电路实现上述函数需要 4 个 3 输入与门和一个 4 输入或门,如图 4.2.8 所示。

如果限定逻辑门的扇入数为3,需要对上式进行变换,函数分解得到

$$\begin{aligned} L &= (\bar{A}\bar{B} + \bar{A}B)C + (AB + \bar{A}\bar{B})D \\ &= (\bar{A}\bar{B} + \bar{A}B)C + (\bar{A}\bar{B} + AB)D \end{aligned}$$

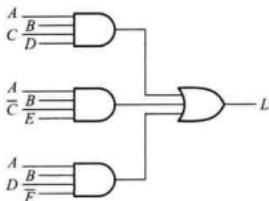


图 4.2.6 提取公因子前的与-或两级电路

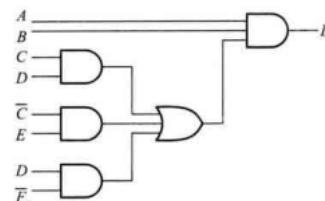


图 4.2.7 提取公因子后的 3 级电路

实现上述逻辑门的最大扇入数为2,满足限定条件。但电路级数增加到5级,如图4.2.9所示。

提取公因子或函数分解可以将与-或两级电路转换为多级电路,减少逻辑门的扇入数,以满足设计要求。但电路级数的增加可能会使总延时增加。因此变换后电路性能的好坏需综合评定。

上述介绍的逻辑电路优化实现是基于代数法或卡诺图法化简进行的,只适合函数的变量数较少(不大于5)时的手工化简,当变量数较多时,将优化实现的策略写入计算机程序,由计算机自动完成。

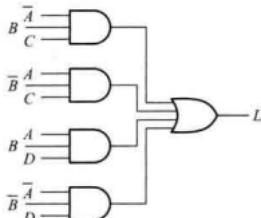


图 4.2.8 函数分解前的与-或两级电路

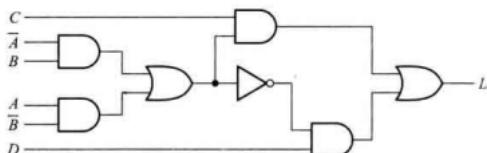


图 4.2.9 函数分解后的 5 级电路

复习思考题

- 4.2.1 列出设计组合逻辑电路的步骤。
- 4.2.2 为什么说在组合逻辑电路设计中正确列出真值表是最关键的一步?
- 4.2.3 什么是逻辑函数的优化实现?这里介绍的优化实现的策略有哪些?

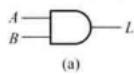
4.3 组合逻辑电路中的竞争-冒险

前面进行组合逻辑电路的分析和设计时,都没有考虑逻辑门的延迟时间对电路产生的影响,并且认为电路的输入和输出均处于稳定的逻辑电平。实际上,信号经过逻辑门电路都需要一定的时间。由于不同路径上门的级数不同,信号经过不同路径传输的时间不同。或者门的级数相同,而各个门延迟时间的差异,也会造成传输时间的不同。因此,电路在信号电平变化瞬间,可能与稳态下的逻辑功能不一致,产生错误输出,这种现象就是电路中的竞争-冒险。

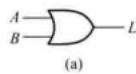
4.3.1 产生竞争-冒险的原因

下面通过两个简单电路的工作情况,说明产生竞争-冒险的原因。图 4.3.1(a)所示的与门在稳态情况下,当 $A=0, B=1$ 或者 $A=1, B=0$ 时,输出 L 始终为 0。如果信号 A, B 的变化同时发生,则能满足要求。若由于前级门电路的延迟差异或其他原因,致使 B 从 1 变为 0 的时刻,滞后于 A 从 0 变为 1 的时刻,因此,在很短的时间间隔内,与门的两个输入端均为 1,其输出端出现一个高电平窄脉冲(干扰脉冲),如图 4.3.1(b)所示,图中考虑了与门的延迟。

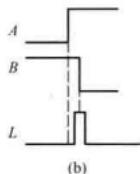
同理,图 4.3.2(a)所示的或门在稳态情况下,当 $A=0, B=1$ 或者 $A=1, B=0$ 时,输出 L 始终为 1。而当 A 从 0 变为 1 时刻,滞后于 B 从 1 到 0 的时刻,则在很短的时间间隔内,或门的两个输入端均为 0,使输出出现一个低电平窄脉冲,如图 4.3.2(b)所示。



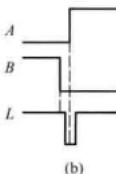
(a)



(a)



(b)



(b)

图 4.3.1 产生正跳变脉冲的竞争-冒险 图 4.3.2 产生负跳变脉冲的竞争-冒险
 (a) 逻辑电路 (b) 工作波形

下面进一步分析组合逻辑电路产生的竞争-冒险。图 4.3.3(a)所示的逻辑电路中,它的输出逻辑表达式为 $L=AC+\bar{BC}$ 。由此式可知,当 A 和 B 都为 1 时,表达式简化成两个互补信号相加,即 $L=C+\bar{C}$,此时,该电路可能存在竞争-冒险。由图 4.3.3(b)所示的波形图可以看出,在 C 由 1 变 0 时, \bar{C} 由 0 变 1 有一延迟时间, G_2 和 G_3 的输出 AC 和 \bar{BC} 分别相对于 C 和 \bar{C} 均有延迟, AC 和

\bar{BC} 经过 G_4 的延迟而使输出出现一负跳变的窄脉冲。

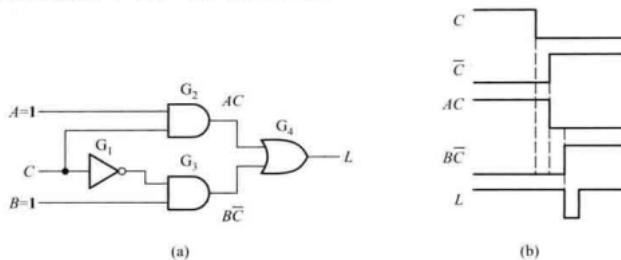


图 4.3.3 组合逻辑电路的竞争-冒险

(a) 逻辑电路 (b) 工作波形

综上所述,一个逻辑门的两个输入端的信号同时向相反方向变化,而变化的时间有差异的现象,称为竞争。两个输入端可以是不同变量所产生的信号,但其取值的变化方向是相反的,参见图 4.3.1 和图 4.3.2 中的 AB 及 $A+B$ 。也可以是在一定条件下,门电路输出端的逻辑表达式简化成两个互补信号相乘或者相加,即 $L=A \cdot \bar{A}$ 或者 $L=A+\bar{A}$,如图 4.3.3 所示。由竞争而可能产生输出干扰脉冲的现象称为冒险。

在考虑延迟的条件下,若与门的两个输入 A 和 \bar{A} ,其中一个先从 0 变 1 时,则 $A \cdot \bar{A}$ 会向其非稳定值 1 变化,此时会产生冒险;若或门的两个输入 A 和 \bar{A} ,其中一个先从 1 变 0 时,则 $A+\bar{A}$ 会向其非稳定值 0 变化,也会产生冒险。两者之间存在对偶关系。

值得注意的是,有竞争现象时不一定都会产生干扰脉冲,如图 4.3.2(a)中,如果 A 从 0 变为 1 时刻没有滞后信号 B 的变化,则输出不会产生冒险。在一个复杂的逻辑系统中,由于信号的传输路径不同,或者各个信号延迟时间的差异、信号变化的互补性以及其他一些因素,很容易产生竞争-冒险现象。因此在电路设计中应尽量减小冒险产生。

4.3.2 消去竞争-冒险的方法

针对上述分析,可以采取以下措施来消去竞争-冒险现象。

1. 发现并消去互补相乘项

例如,函数式 $F=(A+B)(\bar{A}+C)$,在 $B=C=0$ 时, $F=\bar{A}A$ 。若直接根据这个逻辑表达式组成逻辑电路,如图 4.3.4(a)所示,则可能出现竞争-冒险。如将该式变换为 $F=\bar{A}\bar{A}+AC+\bar{A}B+BC=AC+\bar{A}B+BC$,这里已将 $\bar{A}A$ 消掉。根据这个表达式组成逻辑电路,如图 4.3.4(b)所示,就不会出现竞争-冒险。

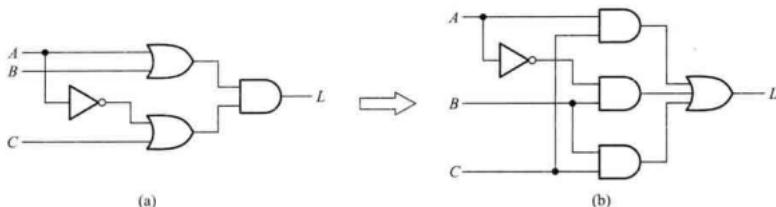


图 4.3.4 逻辑表达式的不同实现方法
(a) 存在竞争-冒险的电路 (b) 无竞争-冒险的电路

2. 增加乘积项以避免互补项相加

对于图 4.3.3(a)所示的逻辑电路,可以根据常用恒等式增加乘积项,将输出逻辑表达式 $L = AC + \bar{B}C$ 变为 $L = AC + \bar{B}C + AB$, 卡诺图如图 4.3.5 所示,对应的逻辑电路如图 4.3.6 所示。当 $A=B=1$ 时,根据逻辑表达式有 $L = C + \bar{C}+1$, 不会只出现互补项相加的情况。而此时电路中, G_5 输出为 1,使 G_4 输出亦为 1,这就消除了 C 的状态变化对输出状态的影响,从而消去了竞争-冒险。

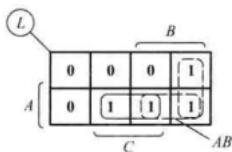


图 4.3.5 增加了乘积项 AB 的卡诺图

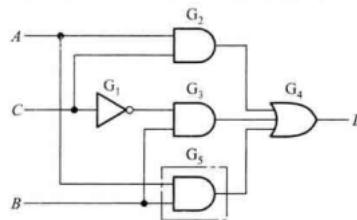


图 4.3.6 增加了乘积项 AB 的逻辑电路

3. 输出端并联电容器

对于工作速度不高的逻辑门构成的电路,为了消去竞争-冒险产生的干扰窄脉冲,可以在输出端并联一滤波电容,其容量为 $4\sim20\text{ pF}$ 之间,如图 4.3.7(a)所示, R_o 是逻辑门电路的输出电阻。若在图 4.3.3(a)所示电路的输出端并联电容 C ,当 $A=B=1,C$ 的波形与图 4.3.3 相同的情况下,得到如图 4.3.7(b)所示的输出波形。显然,电容对窄脉冲起到平波的作用,消除输出端出现的逻辑错误,但同时也使输出波形上升沿或下降沿变得缓慢。

以上介绍了产生竞争-冒险的原因和克服竞争-冒险的方法。现代数字电路或数字系统的分析与设计,可以借助计算机进行时序仿真,检查电路是否存在竞争-冒险现象。仿真时,由于逻辑门电路的传输延迟时间是采用软件设定的标准值或设计者自行设定的值,与电路的实际工作情况有差异,最终要在实验中检查验证。因此,要能很好地解决这一问题,还必须在实践中积累和总结经验。

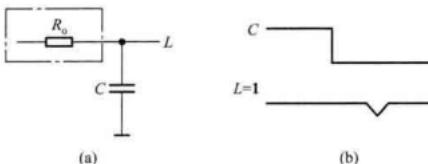


图 4.3.7 并联电容器消去竞争冒险

(a) 电路 (b) 输出波形

复习思考题

- 4.3.1 什么是组合逻辑电路中的竞争-冒险?
4.3.2 列出三种消去组合逻辑电路竞争-冒险的方法。

4.4 若干典型的组合逻辑电路

半导体制作工艺的发展,将许多常用的组合逻辑电路制成了中规模集成芯片,曾被广泛应用。在现代数字电路和数字系统的设计中,这些典型的组合逻辑电路经常被当做基本模块,很多EDA工具将这些基本模块作为库元件,也可以用HDL语言描述这些模块的功能,由高层设计调用,以便构建所需要的逻辑电路。这些模块包括编码器、译码器、数据选择器、数据分配器、数值比较器、算术运算单元等。下面着重分析它们的工作原理及基本应用方法。

4.4.1 编码器

本节和下一节分别讨论编码器和译码器。编码和译码问题在日常生活中经常遇到。例如,你购买一台移动电话时,通信公司给你的电话设定一个号码,叫做编码。显然,这个特定的号码与你的姓名是等同的,任何人拨打你的移动电话号码,都能够找到你,这叫做译码。

1. 编码器的定义与工作原理

数字系统中存储或处理的信息,常常是用二进制码表示的。用一个二进制代码表示特定含义的信息称为编码。具有编码功能的逻辑电路称为编码器。图 4.4.1 所示为二进制编码器的结构图,它有 n 位二进制码输出,与 2^n 个输入相对应。编码器有普通编码器和优先编码器之分。普通编码器任何时刻只允许一个输入信号有效,否则将产生错误输出。优先编码器允许多个输入信号同时有效,输出是对优先级别高的输入信号进行编码。



图 4.4.1 二进制编码器结构框图

(1) 普通编码器

4 线-2 线普通编码器真值表如表 4.4.1 所示, 根据真值表设计编码器电路。4 个输入 I_0 到 I_3 为高电平有效, 输出是 2 位二进制代码 $Y_1 Y_0$, 任何时刻 $I_0 \sim I_3$ 中只能有一个取值为 1, 并且有一组对应的二进制代码输出。除表中列出 4 个输入变量的 4 种取值组合有效外, 其余 12 种组合所对应的输出均应为 0。由真值表可以得到如下逻辑表达式:

$$\begin{aligned} Y_1 &= \overline{I_0} \overline{I_1} \overline{I_2} \overline{I_3} + \overline{I_0} \overline{I_1} \overline{I_2} I_3 \\ Y_0 &= \overline{I_0} I_1 \overline{I_2} \overline{I_3} + I_0 \overline{I_1} \overline{I_2} I_3 \end{aligned} \quad (4.4.1)$$

任何时刻只有一个输入为 1, 如果将表 4.4.1 中没有列出的 12 种组合所对应的输出看作无关项, 并化简得

$$\begin{aligned} Y_1 &= I_2 + I_3 \\ Y_0 &= I_1 + I_3 \end{aligned} \quad (4.4.2)$$

式(4.4.2)是考虑了无关项的简化结果, 比式(4.4.1)简单。根据式(4.4.2)的逻辑表达式画出逻辑图如图 4.4.2 所示。

表 4.4.1 4 线-2 线编码器真值表

输入				输出	
I_0	I_1	I_2	I_3	Y_1	Y_0
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

应当特别注意, 这一电路实现正常编码时, 对输入信号有严格的限制, 即任何时刻 $I_0 \sim I_3$ 中只能并且必须有一个取值为 1。例如, 当 I_1, I_2 同时为 1 时, 输出出现错误编码 $Y_1 Y_0 = 11$ 。因为正常编码时, 输出 11 表示 I_3 为 1。

在实际应用中, 经常会遇到两个以上的输入同时为有效信号的情况。因此, 必须根据轻重缓急, 事先规定好这些输入编码的先后次序, 即优先级别。识别这类请求信号的优先级别并进行编码的逻辑电路称为优先编码器。

(2) 优先编码器

4 线-2 线优先编码器的功能表如表 4.4.2 所示, 根据真值表设计优先编码器电路。由表 4.4.2 可知 $I_0 \sim I_3$ 的优先级别。例如, 对于 I_0 , 只有当 I_1, I_2, I_3 均为 0, 即均无有效电平输入, 且 I_0 为 1 时, 输出为 00。对于 I_3 , 无论其他 3 个输入是否为有效电平输入, 输出均为 11。由此可知 I_3

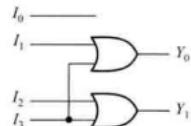


图 4.4.2 4 线-2 线编码器逻辑图

的优先级别高于 I_0 的优先级别,且这 4 个输入的优先级别由高到低的次序依次为 I_3, I_2, I_1, I_0 。优先编码器允许 2 个以上的输入同时为 1,但只对优先级别比较高的输入进行编码。

由表 4.4.2 可以得出该优先编码器的逻辑表达式为

$$Y_1 = I_2 \bar{I}_3 + I_3 = I_2 + I_3$$

$$Y_0 = I_1 \bar{I}_2 \bar{I}_3 + I_3 = I_1 \bar{I}_2 + I_3$$

上述两种类型的编码器均存在一个问题,当电路所有的输入为 0 时,输出 $Y_1 Y_0$ 均为 0。而当 I_0 为 1 时,输出 $Y_1 Y_0$ 也全为 0,即输入条件不同而输出代码相同。这两种情况在实际中必须加以区分,解决的方法将在下面例题中介绍。

表 4.4.2 4 线-2 线优先编码器真值表

输入				输出	
I_0	I_1	I_2	I_3	Y_1	Y_0
1	0	0	0	0	0
x	1	0	0	0	1
x	x	1	0	1	0
x	x	x	1	1	1

例 4.4.1 键盘输入逻辑电路就是由编码器组成的。图 4.4.3 是用十个按键和门电路组成的 8421 码编码器,其真值表如表 4.4.3 所示,十个按键 $\bar{S}_0 \sim \bar{S}_9$ 分别对应十进制数 0 ~ 9,编码器的输出为 $ABCD$ 和 GS ,试分析该电路的工作原理及 GS 的作用。

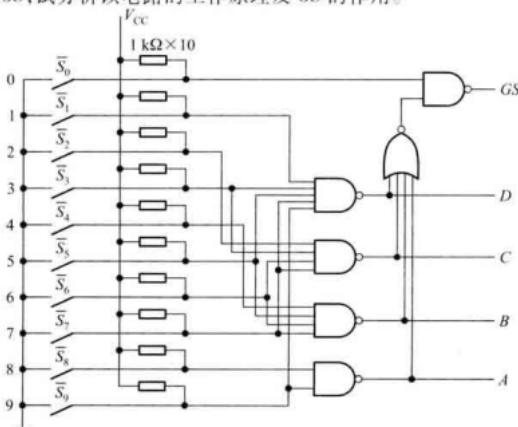


图 4.4.3 用十个按键和门电路组成的 8421BCD 码编码器

表 4.4.3 十个按键 8421BCD 码编码器真值表

输入										输出				
\bar{S}_9	\bar{S}_8	\bar{S}_7	\bar{S}_6	\bar{S}_5	\bar{S}_4	\bar{S}_3	\bar{S}_2	\bar{S}_1	\bar{S}_0	A	B	C	D	GS
1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
1	1	1	1	1	1	1	1	1	0	0	0	0	0	1
1	1	1	1	1	1	1	1	0	1	0	0	0	1	1
1	1	1	1	1	1	1	0	1	1	0	0	1	0	1
1	1	1	1	1	1	0	1	1	1	0	0	1	1	1
1	1	1	1	1	0	1	1	1	1	0	1	0	0	1
1	1	1	1	0	1	1	1	1	1	0	1	0	1	1
1	1	1	0	1	1	1	1	1	1	0	1	1	0	1
1	0	1	1	1	1	1	1	1	1	1	0	0	0	1
0	1	1	1	1	1	1	1	1	1	1	0	0	1	1

解：对真值表和逻辑电路进行分析，可得知：①该编码器为输入低电平有效，因此用带非号的变量表示；②在按下 $\bar{S}_0 \sim \bar{S}_9$ 中任意一个键时，即输入信号中有一个为低电平时，ABCD 输出信号为该键的 BCD 编码，同时 $GS = 1$ ，表示有信号输入。而只有 $\bar{S}_0 \sim \bar{S}_9$ 均为高电平时 $GS = 0$ ，表示无信号输入，此时的输出代码 0000 为无效代码。由此解决了图 4.4.2 所示电路存在的问题，即输入条件不同而输出代码相同。

2. 典型编码器电路

8 线-3 线优先编码器 CD4532 是 CMOS 中规模集成电路，其功能如表 4.4.4 所示。

表 4.4.4 优先编码器 CD4532 功能表

输入									输出				
EI	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	Y_2	Y_1	Y_0	GS	EO
0	x	x	x	x	x	x	x	x	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	1
1	1	x	x	x	x	x	x	x	1	1	1	1	0
1	0	1	x	x	x	x	x	x	1	1	0	1	0
1	0	0	1	x	x	x	x	x	1	0	1	1	0
1	0	0	0	1	x	x	x	x	1	0	0	1	0
1	0	0	0	0	1	x	x	x	0	1	1	1	0
1	0	0	0	0	0	1	x	x	0	1	0	1	0
1	0	0	0	0	0	0	1	x	0	0	1	1	0
1	0	0	0	0	0	0	0	1	0	0	0	1	0

从功能表可以看出，该编码器有 8 个信号输入端，3 个二进制码输出端，输入和输出均以高

电平作为有效电平,而且输入优先级别由高到低的次序依次为 I_7, I_6, \dots, I_0 。此外,为便于多个芯片连接起来扩展电路的功能,还设置了高电平有效的输入使能端 EI 和输出使能端 EO ,以及优先编码工作状态标志 GS 。

当 $EI=1$ 时,编码器工作;而当 $EI=0$ 时,禁止编码器工作,此时不论 8 个输入端为何种状态,3 个输出端均为低电平,且 GS 和 EO 均为低电平。

只有在 EI 为 1,且所有输入端都为 0 时, EO 输出为 1,它可与另一片相同器件的 EI 连接,以便组成更多输入端的优先编码器。

GS 的功能是,当 EI 为 1,且至少有一个输入端有高电平信号输入时, GS 为 1,表明编码器处于工作状态,否则 GS 为 0,由此可以区分当电路所有输入端均无高电平输入,或者只有 I_0 输入端有高电平时, $Y_2 Y_1 Y_0$ 均为 000 的情况。

根据功能表推导出各输出端的逻辑表达式为

$$\begin{cases} Y_2 = EI(I_7 + I_6 + I_5 + I_4) \\ Y_1 = EI(I_7 + I_6 + \bar{I}_5 \bar{I}_4 I_3 + \bar{I}_5 \bar{I}_4 I_2) \\ Y_0 = EI(I_7 + \bar{I}_6 I_5 + \bar{I}_6 \bar{I}_4 I_3 + \bar{I}_6 \bar{I}_4 \bar{I}_2 I_1) \\ EO = EI(\bar{I}_7 \bar{I}_6 \bar{I}_5 \bar{I}_4 \bar{I}_3 \bar{I}_2 \bar{I}_1 \bar{I}_0) \\ GS = EI(I_7 + I_6 + I_5 + I_4 + I_3 + I_2 + I_1 + I_0) \end{cases} \quad (4.4.3)$$

由式(4.4.3)可以画出逻辑图(此处省略),逻辑符号如图 4.4.4 所示。

例 4.4.2 用两片 8 线-3 线编码器 CD4532 组成 16 线-4 线优先编码器,其逻辑图如图 4.4.5 所示,试分析其工作原理。

解:根据 CD4532 的功能表,对逻辑图进行分析得出:

(1) 当 $EI_1 = 0$ 时,片(1)禁止编码,其输出端 $Y_2 Y_1 Y_0$ 为 000,而且 GS_1, EO_1 均为 0。同时 EO_1 使 $EI_0 = 0$,片(0)也禁止编码,其输出端及 GS_0, EO_0 均为 0。由电路图可知 $GS = GS_1 + GS_0 = 0$,表示此时电路输出端的代码 $L_3 L_2 L_1 L_0 = 0000$ 是非编码输出。

(2) 当 $EI_1 = 1$ 时,片(1)允许编码,若 $A_{15} \sim A_8$ 均无有效电平输入,则 $EO_1 = 1$,使 $EI_0 = 1$,从而允许片(0)编码,因此片(1)的优先级高于片(0)。

此时 $A_{15} \sim A_8$ 没有有效电平输入,片(1)的输出均为 0,使 4 个或门都打开, L_2, L_1, L_0 取决于片(0)的输出,而 $L_3 = GS_1$ 总是等于 0,所以输出代码在 0000 ~ 0111 之间变化。若只有 A_0 有高电平输入,输出为 0000,若 A_7 及其他输入同时有高电平输入,则输出为 0111。 A_0 的优先级别最低。

(3) 当 $EI_1 = 1$ 且 $A_{15} \sim A_8$ 中至少有一个为高电平输入时, $EO_1 = 0$,使 $EI_0 = 0$,片(0)禁止编码,此时 $L_3 = GS_1 = 1, L_2, L_1, L_0$ 取决于片(1)的输出,输出代码在 1000 ~ 1111 之间变化,并且 A_{15} 的优先级别最高。

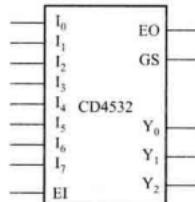


图 4.4.4 优先编码器 CD4532 的逻辑符号

电路实现了 16 位输入的优先编码，优先级别从 A_{15} 至 A_0 依次递减。

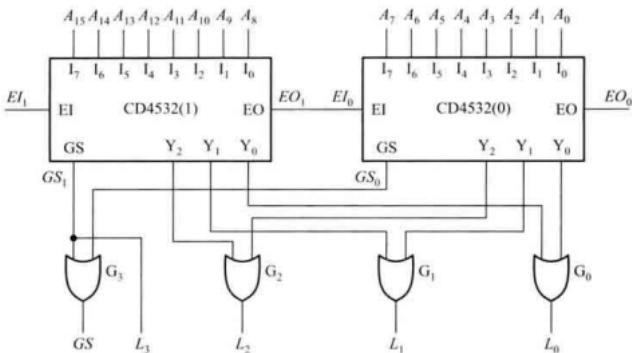


图 4.4.5 例 4.4.2 的逻辑图

复习思考题

4.4.1.1 普通编码器和优先编码器的区别是什么？

4.4.1.2 优先编码器 CD4532 的输入、输出信号是高电平有效还是低电平有效？

4.4.1.3 说明 CD4532 的输入信号 EI 和输出信号 GS, EO 的作用。

4.4.2 译码器/数据分配器

在数字系统中，经常需要将一种代码转换为另一种代码，以满足特定的需要，完成这种功能的电路称为码转换电路。译码器和编码器都是码转换电路。

1. 译码器的定义与功能

译码是编码的逆过程，它的功能是将具有特定含义的二进制码转换成对应的输出信号，具有译码功能的逻辑电路称为译码器。

译码器可分为两种类型，一种是将一系列代码转换成与之一一对应的有效信号。这种译码器可称为二进制译码器或唯一地址译码器，它常用于计算机中对存储器单元地址的译码，即：将每一个地址代码转换成一个有效信号，从而选中对应的单元。另一种是将一种代码转换成另一种代码，所以也称为代码变换器。

图 4.4.6 表示二进制译码器的一般框图，它具有 n 个输入端、 2^n 个输出端和一个使能输入端。在使能输入端为有效电平时，对应每一

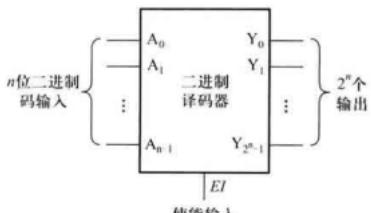


图 4.4.6 二进制译码器框图

组输入代码,只有其中一个输出端为有效电平,其余输出端则为相反电平。输出信号可以是高电平有效,也可以是低电平有效。

下面以2线-4线译码器为例,分析译码器的工作原理和电路结构。

2输入变量 A_1, A_0 共有4种不同状态组合,因而译码器有4个输出信号 $\bar{Y}_0 \sim \bar{Y}_3$,并且输出为低电平有效,真值表如表4.4.5所示。

表4.4.5 2线-4线译码器真值表

输入		输出				
\bar{E}	A_1	A_0	\bar{Y}_0	\bar{Y}_1	\bar{Y}_2	\bar{Y}_3
1	x	x	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

另外设置了使能控制端 \bar{E} ,当 \bar{E} 为1时,无论 A_1, A_0 为何种状态,输出全为1,译码器处于非工作状态。而当 \bar{E} 为0时,对应于 A_1, A_0 的一种输入状态,其中只有一个输出端为0,其余各输出端均为1。例如, $A_1 A_0 = 00$ 时, \bar{Y}_0 为0, $\bar{Y}_1 \sim \bar{Y}_3$ 均为1。由此可见,译码器是通过输出端的逻辑电平以识别不同的代码。

根据真值表可写出各输出端的逻辑表达式为

$$\begin{cases} \bar{Y}_0 = \overline{\bar{E}} \overline{A_1} \overline{A_0} \\ \bar{Y}_1 = \overline{\bar{E}} \overline{A_1} A_0 \\ \bar{Y}_2 = \overline{\bar{E}} A_1 \overline{A_0} \\ \bar{Y}_3 = \overline{\bar{E}} A_1 A_0 \end{cases} \quad (4.4.4)$$

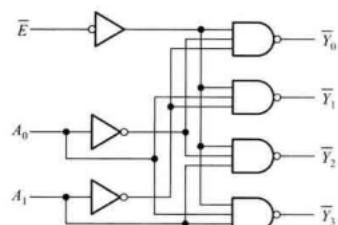


图4.4.7 2线-4线译码器逻辑图

根据逻辑表达式画出逻辑图如图4.4.7所示。

2. 典型译码器电路及应用

(1) 二进制译码器

典型的二进制译码器有2线-4线译码器和3线-8线译码器。

3线-8线译码器的逻辑图如图4.4.8(a)所示,逻辑符号如图4.4.8(b)所示。该译码器有3

位二进制输入 A_2, A_1, A_0 , 它们共有 8 种组合状态, 即可译出 8 个输出信号 $\bar{Y}_0 \sim \bar{Y}_7$, 输出为低电平有效。此外, 还设置了 3 个使能输入端 E_3, \bar{E}_2 和 \bar{E}_1 , 并且 $E = E_3 \bar{E}_2 \bar{E}_1$, 为扩展电路的功能提供了方便。由逻辑图写出:

$$\left\{ \begin{array}{l} \bar{Y}_0 = \overline{E \bar{A}_2 \bar{A}_1 \bar{A}_0} \\ \bar{Y}_1 = \overline{E \bar{A}_2 \bar{A}_1 A_0} \\ \bar{Y}_2 = \overline{E \bar{A}_2 A_1 \bar{A}_0} \\ \bar{Y}_3 = \overline{E \bar{A}_2 A_1 A_0} \\ \bar{Y}_4 = \overline{E A_2 \bar{A}_1 \bar{A}_0} \\ \bar{Y}_5 = \overline{E A_2 \bar{A}_1 A_0} \\ \bar{Y}_6 = \overline{E A_2 A_1 \bar{A}_0} \\ \bar{Y}_7 = \overline{E A_2 A_1 A_0} \end{array} \right. \quad (4.4.5)$$

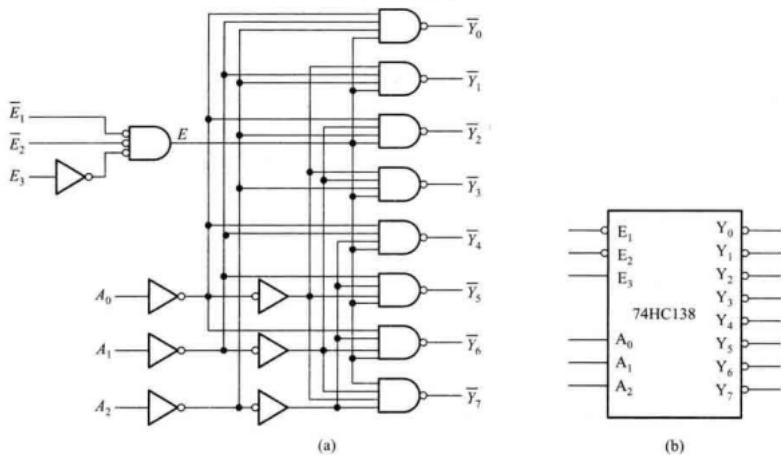


图 4.4.8 3 线-8 线译码器
(a) 逻辑图 (b) 逻辑符号

当 $E_3 = 1$, 且 $\bar{E}_2 = \bar{E}_1 = 0$ 时, $E = 1$, 带入式(4.4.5) 可得

$$\bar{Y}_0 = \bar{m}_0, \bar{Y}_1 = \bar{m}_1, \bar{Y}_2 = \bar{m}_2, \bar{Y}_3 = \bar{m}_3, \bar{Y}_4 = \bar{m}_4, \bar{Y}_5 = \bar{m}_5, \bar{Y}_6 = \bar{m}_6, \bar{Y}_7 = \bar{m}_7$$

译码器的输出包含了输入 A_2, A_1, A_0 组成的所有最小项。

根据式(4.4.5)可以列出3线-8线译码器功能表如表4.4.6所示。

利用3线-8线译码器可以构成4线-16线、5线-32线或6线-64线译码器。

表4.4.6 3线-8线译码器功能表

输入			输出										
E_3	\bar{E}_2	\bar{E}_1	A_2	A_1	A_0	\bar{Y}_0	\bar{Y}_1	\bar{Y}_2	\bar{Y}_3	\bar{Y}_4	\bar{Y}_5	\bar{Y}_6	\bar{Y}_7
x	1	x	x	x	x	1	1	1	1	1	1	1	1
x	x	1	x	x	x	1	1	1	1	1	1	1	1
0	x	x	x	x	x	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

集成3线-8线译码器有CMOS(如74HC138)和TTL(如74LS138)的产品，两者在逻辑功能上没有区别，只是电性能参数不同，用74x138表示两者中任意一种。74x139是双2线-4线，两个独立的译码器封装在一个集成芯片中，其中之一的逻辑符号如图4.4.9(a)所示。

逻辑符号说明 74x139逻辑符号框外部的 $\bar{E}, \bar{Y}_0 \sim \bar{Y}_3$ 作为变量符号，表示外部输入或输出

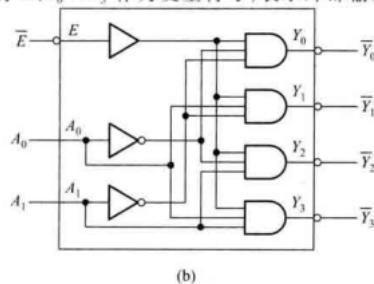
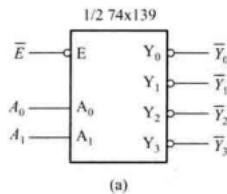


图4.4.9 74x139逻辑符号及结构

(a) 逻辑符号 (b) 逻辑符号结构

信号名称,字母上面的“—”号说明该输入或输出是低电平有效,如图 4.4.9(b)所示。符号框内部的输入、输出变量表示其内部的逻辑关系,全部为高电平有效。当输入或输出为低电平有效时,符号框外部逻辑变量 $\bar{E} \sim \bar{Y}_0 \sim \bar{Y}_3$ 的逻辑状态与符号框内相应的 $E \sim Y_0 \sim Y_3$ 的逻辑状态相反。在推导表达式的过程中,如果低有效的输入或输出变量上面的“—”号参与运算,则在画逻辑图或验证真值表时,注意将其还原为低有效符号。

例 4.4.3 试用四片 3 线-8 线译码器(74HC138)和一片 2 线-4 线译码器(74HC139)构成 5 线-32 线译码器,输入为 5 位二进制码 $B_4B_3B_2B_1B_0$,对应输出 $\bar{L}_0 \sim \bar{L}_{31}$ 为低电平有效信号。

解:首先列出 5 线-32 线译码器的真值表如表 4.4.7 所示。

表 4.4.7 5 线-32 线译码器真值表

输入					输出											
B_4	B_3	B_2	B_1	B_0	\bar{L}_0	\bar{L}_1	\bar{L}_2	\bar{L}_3	\bar{L}_4	...	\bar{L}_{27}	\bar{L}_{28}	\bar{L}_{29}	\bar{L}_{30}	\bar{L}_{31}	
0	0	0	0	0	0	1	1	1	1	...	1	1	1	1	1	1
0	0	0	0	1	1	0	1	1	1	...	1	1	1	1	1	1
0	0	0	1	0	1	1	0	1	1	...	1	1	1	1	1	1
0	0	0	1	1	1	1	1	0	1	...	1	1	1	1	1	1
0	0	1	0	0	1	1	1	1	0	...	1	1	1	1	1	1
⋮					⋮											
1	1	0	1	1	1	1	1	1	1	...	0	1	1	1	1	1
1	1	1	0	0	1	1	1	1	1	...	1	0	1	1	1	1
1	1	1	0	1	1	1	1	1	1	...	1	1	0	1	1	1
1	1	1	1	0	1	1	1	1	1	...	1	1	1	0	1	1
1	1	1	1	1	1	1	1	1	1	...	1	1	1	1	1	0

从表中可以看出,当 $B_4B_3=00$,而 $B_2B_1B_0$ 从 000 变化到 111 时,对应 $\bar{L}_0 \sim \bar{L}_7$ 中有一个输出为 0,其余输出全为 1,因此 4 片 3 线-8 线译码器(74HC138)中,设置片(0)为译码状态,其余 3 片为禁止译码状态,对应的输出 $\bar{L}_8 \sim \bar{L}_{31}$ 全为 1。

以此类推,当 $B_4B_3=01$, $B_2B_1B_0$ 从 000 变化到 111 时, $\bar{L}_8 \sim \bar{L}_{15}$ 分别输出有效信号,此时设置片(1)为译码状态。当 $B_4B_3=10$ 和 11 时,分别设置片(2)和片(3)为译码状态。因此,将 5 位二进制码的低 3 位 $B_2B_1B_0$ 分别与 4 片 3 线-8 线译码器的 3 个地址输入端 $A_2A_1A_0$ 并接在一起。

高位 B_4B_3 有 4 种状态的组合,因此接入 2 线-4 线译码器(74HC139)的两个地址输入端 A_1A_0 ,2 线-4 线译码器的 4 个低有效输出信号分别接入 4 片 3 线-8 线译码器的低使能输入端,使它们在 B_4B_3 的控制下轮流工作在译码状态。这样就得到 5 线-32 线译码器,逻辑图如图 4.4.10 所示。

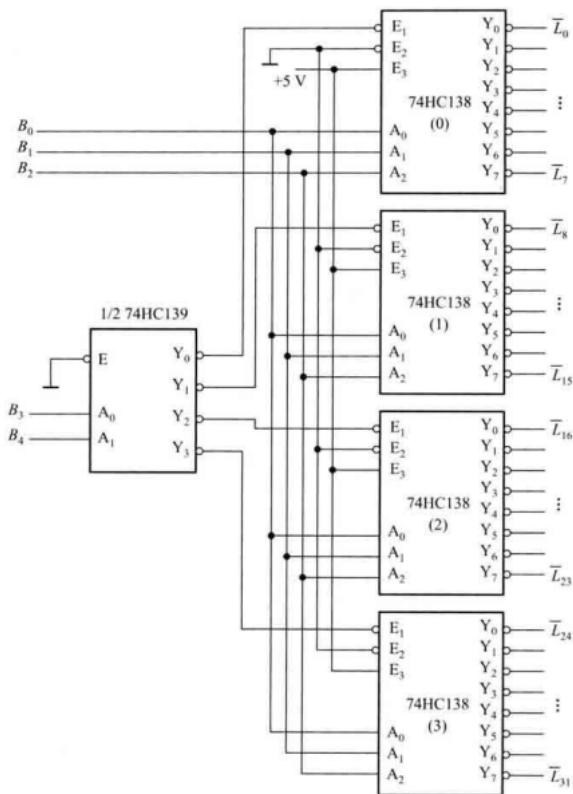


图 4.4.10 例 4.4.3 的逻辑图

例 4.4.4 用 3 线-8 线译码器(74HC138)和必要的逻辑门实现函数 $L = \overline{AC} + AB$ 。

解：当控制端接有效电平时，译码器的输出是 3 个输入变量的全部最小项。因此，首先将函数式变换为最小项之和的形式

$$L = \overline{ABC} + \overline{ABC} + \overline{ABC} + ABC = m_0 + m_2 + m_6 + m_7$$

将输入变量 A, B, C 分别接入 A_2, A_1 和 A_0 端，并将使能端接有效电平。由于译码器输出是低电平有效，所以将最小项变换为反函数的形式

$$L = \overline{\overline{m_0}} \cdot \overline{\overline{m_1}} \cdot \overline{\overline{m_6}} \cdot \overline{\overline{m_7}} = \overline{\overline{Y_0}} \cdot \overline{\overline{Y_1}} \cdot \overline{\overline{Y_6}} \cdot \overline{\overline{Y_7}}$$

在译码器的输出端加一个与非门, 将这些最小项组合起来, 便可实现 3 变量组合逻辑函数, 如图 4.4.11 所示。

(2) 二-十进制译码器

在第一章已经讨论过 8421BCD 码, 对应于 0~9 的十进制数, 由 4 位二进制数 **0000 ~ 1001** 表示。由于人们不习惯于直接识别二进制数, 所以采用

二-十进制译码器来解决。这种译码器应有 4 个输入端,10 个输出端。它的真值表如表 4.4.8 所示,其输出为低电平有效。当输入超过 8421BCD 码的范围时(即 **1010 ~ 1111**),输出均为高电平,即没有有效译码输出。

根据真值表 4.4.8, 可以写出二-十进制译码器输出与输入的逻辑表达式:

$$\begin{cases} \bar{Y}_0 = \overline{\bar{A}_3 \bar{A}_2 \bar{A}_1 \bar{A}_0} & \bar{Y}_1 = \overline{\bar{A}_3 \bar{A}_2 \bar{A}_1 A_0} \\ \bar{Y}_2 = \overline{\bar{A}_3 \bar{A}_2 A_1 \bar{A}_0} & \bar{Y}_3 = \overline{\bar{A}_3 \bar{A}_2 A_1 A_0} \\ \bar{Y}_4 = \overline{\bar{A}_3 A_2 \bar{A}_1 \bar{A}_0} & \bar{Y}_5 = \overline{\bar{A}_3 A_2 \bar{A}_1 A_0} \\ \bar{Y}_6 = \overline{\bar{A}_3 A_2 A_1 \bar{A}_0} & \bar{Y}_7 = \overline{\bar{A}_3 A_2 A_1 A_0} \\ \bar{Y}_8 = \overline{A_3 \bar{A}_2 \bar{A}_1 \bar{A}_0} & \bar{Y}_9 = \overline{A_3 \bar{A}_2 \bar{A}_1 A_0} \end{cases} \quad (4, 4, 6)$$

表 4.4.8 二-十进制译码器真值表

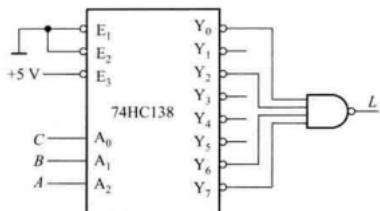


图 4.4.11 例 4.4.4 的逻辑图

续表

数目	BCD 输入				输出									
	A_3	A_2	A_1	A_0	\bar{Y}_0	\bar{Y}_1	\bar{Y}_2	\bar{Y}_3	\bar{Y}_4	\bar{Y}_5	\bar{Y}_6	\bar{Y}_7	\bar{Y}_8	\bar{Y}_9
11	1	0	1	1	1	1	1	1	1	1	1	1	1	1
12	1	1	0	0	1	1	1	1	1	1	1	1	1	1
13	1	1	0	1	1	1	1	1	1	1	1	1	1	1
14	1	1	1	0	1	1	1	1	1	1	1	1	1	1
15	1	1	1	1	1	1	1	1	1	1	1	1	1	1

由式(4.4.6)可以画出逻辑图(此处省略)。

二-十进制译码器应用电路如图 4.4.12 所示。电路的输出分别接在标有十进制数的灯泡。当输入一组 8421BCD 码时, 对应输出端为低电平, 点亮与之相连的灯泡。例如, 当输入 BCD 码 $A_3A_2A_1A_0 = 0110$ 时, 输出 $\bar{Y}_6 = 0$, 它对应于十进制数 6, 其余输出为高电平。

(3) 七段显示译码器

在数字测量仪表和各种数字系统中, 都需要将数字量直观地显示出来, 数字显示电路通常由译码驱动器和显示器等部分组成。数码显示器就是用来显示数字、文字或符号的器件。七段式数字显示器是目前常用的显示方式, 图 4.4.13 表示七段式数字显示器利用不同发光段的组合, 显示 0~9 阿拉伯数字。有些数码显示器增加了一段, 作为小数点。

日常生活中普遍使用七段式数字显示器, 也称为七段数码管。常见的七段显示器有发光二极管和液晶显示器两种, 这里主要介绍前者。发光二极管构成的七段显示器有两种, 共阴极和共阳极电路, 如图 4.4.14 所示。共阴极电路中, 八个发光二极管的阴极连在一起接低电平, 需要某一段发光, 就将相应二极管的阳极接高电平。共阳极显示器的驱动则刚好相反。

为了使数码管能显示十进制数, 必须将十进制数的代码经译码器译出, 然后经驱动器点亮对应的段。例如, 对于 8421 码的 0011 状态, 对应的十进制数为 3, 则译码驱动器应使 a, b, c, d, g 各段点亮。译码器的功能就是, 对应于某一组数码输入, 相应的几个输出端有有效信号输出。

常用的七段显示译码器有两类, 一类译码器输出高电平有效信号, 用来驱动共阴极显示器, 另一类输出低电平有效信号, 以驱动共阳极显示器。下面介绍输出高电平有效的七段显示译码器(74HC4511)。

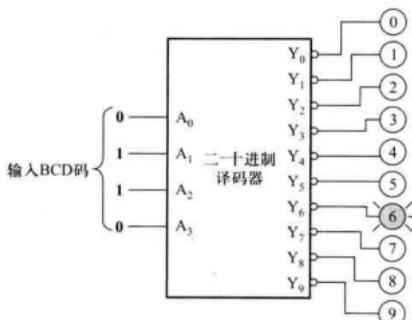


图 4.4.12 二-十进制译码器应用电路

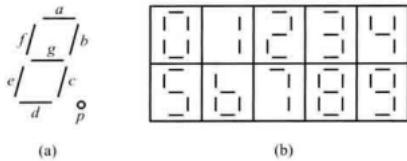


图 4.4.13 七段数字显示器发光段组合图

(a) 分段布置图 (b) 段组合图

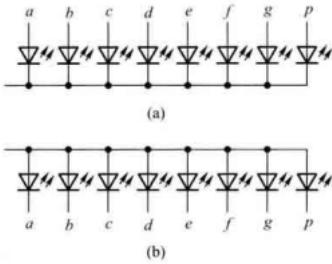


图 4.4.14 二极管显示器等效电路

(a) 共阴极电路 (b) 共阳极电路

七段显示译码器功能表如表 4.4.9 所示。当输入 $D_3D_2D_1D_0$ 接 8421BCD 码时, 输出高电平有效, 用以驱动共阴极显示器。当输入为 1010~1111 六个状态时, 输出全为低电平, 显示器无显示。该显示译码器设有三个辅助控制端 LE 、 \overline{BL} 、 \overline{LT} , 以增强器件的功能, 现分别简要说明如下:

① 灯测试输入 \overline{LT}

当 $\overline{LT}=0$ 时, 无论其他输入端是什么状态, 所有输出 $a \sim g$ 均为 1, 显示器显示字形 \square 。该输入端常用于检查译码器本身及显示器各段的好坏。

② 灭灯输入 \overline{BL}

当 $\overline{BL}=0$, 并且 $\overline{LT}=1$ 时, 无论其他输入端是什么电平, 所有输出 $a \sim g$ 均为 0, 所以字形熄灭。该输入端用于将不必要显示的零熄灭, 例如一个 6 位数字 023.050, 将首、尾多余的 0 熄灭, 则显示为 23.05, 使显示结果更加清楚。

③ 锁存使能输入 LE

在 $\overline{BL}=\overline{LT}=1$ 的条件下, 当 $LE=0$ 时, 锁存器不工作, 译码器的输出随输入码的变化而变化; 当 LE 由 0 跳变为 1 时, 输入码被锁存, 输出只取决于锁存器的内容, 不再随输入的变化而变化。有关锁存器的内容将在第五章介绍。

表 4.4.9 七段显示译码器(74HC4511)功能表

十进制 或功能	输入							输出							字形
	LE	\overline{BL}	\overline{LT}	D_3	D_2	D_1	D_0	a	b	c	d	e	f	g	
0	0	1	1	0	0	0	0	1	1	1	1	1	1	0	□
1	0	1	1	0	0	0	1	0	1	1	0	0	0	0	1
2	0	1	1	0	0	1	0	1	1	0	1	1	0	1	2
3	0	1	1	0	0	1	1	1	1	1	1	0	0	1	3
4	0	1	1	0	1	0	0	0	1	1	0	0	1	1	4

续表

十进制 或功能	输入						输出						字形		
	LE	\overline{BL}	\overline{LT}	D_3	D_2	D_1	D_0	a	b	c	d	e	f	g	
5	0	1	1	0	1	0	1	1	0	1	1	0	1	1	5
6	0	1	1	0	1	1	0	0	0	1	1	1	1	1	6
7	0	1	1	0	1	1	1	1	1	1	0	0	0	0	7
8	0	1	1	1	0	0	0	1	1	1	1	1	1	1	8
9	0	1	1	1	0	0	1	1	1	1	1	0	1	1	9
10	0	1	1	1	0	1	0	0	0	0	0	0	0	0	熄灭
11	0	1	1	1	0	1	1	0	0	0	0	0	0	0	熄灭
12	0	1	1	1	1	0	0	0	0	0	0	0	0	0	熄灭
13	0	1	1	1	1	0	1	0	0	0	0	0	0	0	熄灭
14	0	1	1	1	1	1	0	0	0	0	0	0	0	0	熄灭
15	0	1	1	1	1	1	1	0	0	0	0	0	0	0	熄灭
灯测试	x	x	0	x	x	x	x	1	1	1	1	1	1	1	11
灭 灯	x	0	1	x	x	x	x	0	0	0	0	0	0	0	熄灭
锁 存	1	1	1	x	x	x	x				*			*	

* 此时输出状态取决于 LE 由 0 跳变为 1 时 BCD 码的输入。

根据表(4.4.9)所示功能表,如果不考虑控制端 LE 、 \overline{BL} 、 \overline{LT} 的作用,可以画出 $a \sim g$ 每个字段与 D_3 、 D_2 、 D_1 、 D_0 的卡诺图,并求出每一个字段的最简逻辑表达式。图 4.4.15 所示为 a 段的卡诺图,并求出 a 段的最简逻辑表达式

$$a = \bar{D}_3 \bar{D}_2 D_1 + D_3 \bar{D}_2 \bar{D}_1 + \bar{D}_3 D_2 D_0 + \bar{D}_3 \bar{D}_2 \bar{D}_0$$

当考虑 \overline{BL} 、 \overline{LT} 的作用时, a 段的表达式为

$$a = (\bar{D}_3 \bar{D}_2 D_1 + D_3 \bar{D}_2 \bar{D}_1 + \bar{D}_3 D_2 D_0 + \bar{D}_3 \bar{D}_2 \bar{D}_0) \overline{BL} + \overline{LT}$$

以此类推,可以写出其他字段的最简逻辑表达式。根据各段的表达式可以画出逻辑图(此处省略)。

如果在 D_3 、 D_2 、 D_1 、 D_0 端各增加一个锁存器, LE 为锁存器的控制信号,就可以实现锁存使能输入的控制。

七段显示译码器(74HC4511)与七段显示器的连接方式如图 4.4.16 所示。

例 4.4.5 由 2 线-4 线译码器(74HC139)、显示译码器(74HC4511)和 4 个七段显示器构成的 4 位动态显示电路如图 4.4.17 所示,试分析其工作原理。

解: 如果每位显示器配一个显示译码器驱动,当显示器的位数比较多时,电路的复杂程度会增加。采用动态扫描多位显示器,可以减少显示译码器数目,简化电路连线。

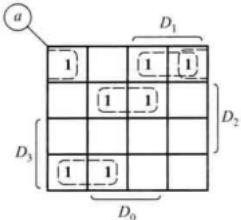


图 4.4.15 a 段的卡诺图

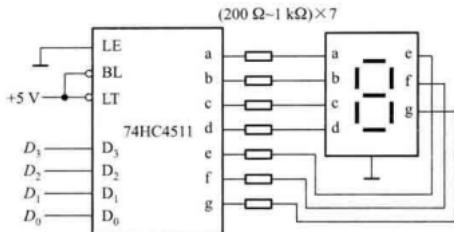


图 4.4.16 七段显示译码器与显示器的连接

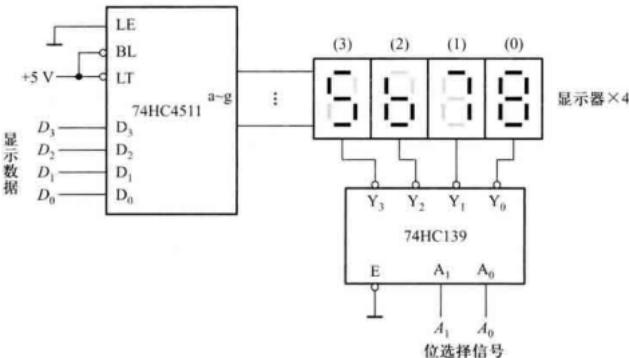


图 4.4.17 例 4.4.5 的译码显示电路

图中 4 个显示器的 $a \sim g$ 段分别接在一起，并与显示译码器的输出 $a \sim g$ 分别相连。将要显示的数据组分别送到显示译码器的输入端 $D_3D_2D_1D_0$ 。2 线-4 线译码器的选择信号 A_1A_0 控制输出 $\bar{Y}_3 \sim \bar{Y}_0$ 依次产生低电平，使 4 个显示器轮流显示。例如要显示 4 位数 5678。位选择信号 $A_1A_0 = 00$ 时， $\bar{Y}_0 = 0$ ，将数据 $D_3D_2D_1D_0 = 1000$ ，经过译码器 74HC4511 送到第(0)位显示器显示 8。当 A_1A_0 分别为 01、10、11 时，将数据 0111、0110 和 0101 分别送到第(1)、(2)、(3)位显示器，分别显示 7、6、5。当以一定的频率重复此过程，在 4 个显示器上分别显示稳定的数字 5678。位选择信号 A_1A_0 必须与显示的数据 $D_3D_2D_1D_0$ 同步变化。

人的视觉暂留时间有一定范围。当显示器频率变化太高时，前一个数字的余辉没消失，又开始显示后一个数字，显示的数码不清晰。当太低时，会使显示闪烁。通常每位显示器频率 f_c 的变化范围为

$$25 < f_c < 100$$

f_c 的单位为 Hz。

3. 数据分配器

数据分配是将公共数据线上的数据根据需要送到不同的通道上去, 实现数据分配功能的逻辑电路称为数据分配器。它的作用相当于多个输出的单刀多掷开关, 其示意图如图 4.4.18 所示。

数据分配器可以用带有使能端的二进制译码器实现。如用 3 线-8 线译码器可以把 1 个数据信号分配到 8 个不同的通道上去。用 3 线-8 线译码器(74HC138)作为数据分配器的逻辑原理图如图 4.4.19 所示。将 E_2 接低电平, E_3 作为使能端, A_2 、 A_1 和 A_0 作为选择通道地址输入, \bar{E}_1 作为数据输入。例如, 当 $E_3 = 1$, $A_2A_1A_0 = 010$ 时, 由功能表(表 4.4.6)可得 \bar{Y}_2 的逻辑表达式

$$\bar{Y}_2 = (\bar{E}_3 \cdot \bar{E}_2 \cdot \bar{E}_1) \cdot \bar{A}_2 \cdot A_1 \cdot A_0 = \bar{E}_1$$

而其余输出端均为高电平。因此, 当地址 $A_2A_1A_0 = 010$ 时, 只有输出端 \bar{Y}_2 得到与输入相同的 data 波形。改变 $A_2A_1A_0$ 的取值可以将数据送到不同的输出端。

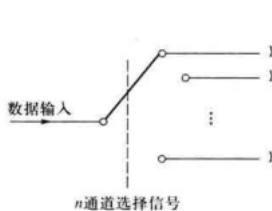


图 4.4.18 数据分配器示意图

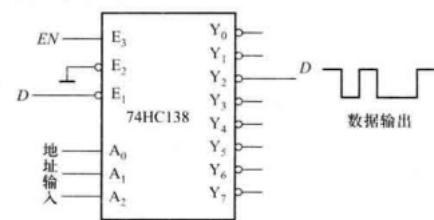


图 4.4.19 用 3 线-8 线译码器作为数据分配器

数据分配器的用途比较多, 比如用它将一台 PC 机与多台外部设备连接, 将计算机的数据分送到外部设备中。它还可以与计数器结合组成脉冲分配器, 用它与数据选择器连接组成分时数据传送系统。

例 4.4.6 试用门电路设计一个具有低电平使能控制的 1 线-4 线数据分配器, 使能信号无效时, 电路所有的输出为高阻态。当通道选择信号将 1 路输入信号连接到其中 1 路输出端时, 其他输出端为高阻状态。

解: (1) 明确逻辑功能, 列出真值表。

根据题意可知, 电路有一个数据输入端 In , 一个低有效的使能端 \bar{E} , 两个通道选择输入端 S_1 、 S_0 , 以及 4 个输出端 Y_3 ~ Y_0 。电路的真值表如表 4.4.10 所示。

(2) 根据真值表写出逻辑表达式。

电路的每个输出端有 3 种状态(0, 1, z), 因此电路的输出级必须由 4 个三态门 G_3 、 G_2 、 G_1 、 G_0 组成。三态门的工作状态由其控制信号决定。设 C_3 、 C_2 、 C_1 、 C_0 分别为 4 个三态门的控制信号。

三态门的控制信号由使能端 \bar{E} 、通道选择输入端 S_1 、 S_0 共同作用产生。

根据表 4.4.10 所示的输入与输出的逻辑关系, 可以写出每个三态门控制端的表达式。

表 4.4.10 例 4.4.6 的 1 线-4 线数据分配器真值表

输 入		输 出				
\bar{E}	S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	z	z	z	In
0	0	1	z	z	In	z
0	1	0	z	In	z	z
0	1	1	In	z	z	z
1	x	x	z	z	z	z

例如, 当 $\bar{E} = 0, S_1 = S_0 = 0$ 时, $C_0 = 1$, 门 G_0 工作, 使得输出 $Y_0 = In$ 。控制端 C_0 的逻辑表达式为: $C_0 = \bar{E} \cdot \bar{S}_1 \cdot \bar{S}_0$ 。其他控制端 $C_3 = C_2 = C_1 = 0$, 对应的三态门输出高阻态。

以此类推, 写出其他三态门控制信号的逻辑表达式为:

$$C_1 = \bar{E} \cdot \bar{S}_1 \cdot S_0, \quad C_2 = \bar{E} \cdot S_1 \cdot \bar{S}_0, \quad C_3 = \bar{E} \cdot S_1 \cdot S_0.$$

(3) 画逻辑电路。

电路的输出级由 4 个三态门组成, 其控制信号是由 2 线-4 线译码器输出确定。译码器的输入为 S_1 和 S_0 及使能端 \bar{E} , 译码器的输出为 C_3, C_2, C_1 和 C_0 。画出完整的逻辑电路如图 4.4.20 所示。注意, \bar{E} 表示低电平有效, 看做完整的变量符号。

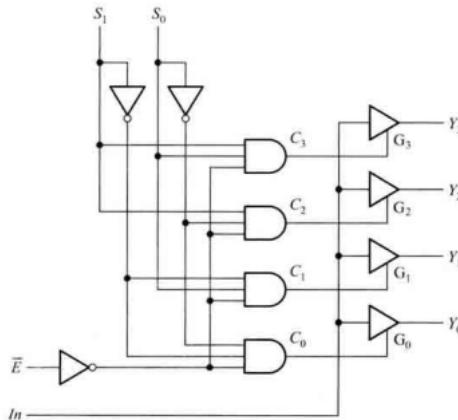


图 4.4.20 例 4.4.6 的逻辑电路

复习思考题

4.4.2.1 什么是译码？什么是二进制译码器或唯一地址译码？

4.4.2.2 发光二极管构成的七段显示器有哪两种，七段显示译码器有哪两类？

4.4.2.3 何种译码器可以作为数据分配器使用？为什么？

4.4.2.4 3 线-8 线译码器(74HC138)作为数据分配器时，对于 \bar{E}_1 、 \bar{E}_2 、 E_3 的设置办法，除了图 4.4.19 所示外，你还有别的什么办法？

4.4.3 数据选择器

1. 数据选择器的定义与功能

数据选择是指经过选择，把多路数据中的某一路数据传送到公共数据线上，实现数据选择功能的逻辑电路称为数据选择器。它的作用相当于多个输入的单刀多掷开关，其示意图如图 4.4.21 所示。

在 3.2.4 节已经介绍了由传输门构成的 2 选 1 数据选择器。这里介绍将 2 选 1 数据选择器作为基本模块，构成更大规模的电路。图 4.4.22 所示为与门和或门构成的 2 选 1 数据选择器电路及逻辑符号，该符号常在大规模逻辑电路中使用。2 选 1 数据选择器选择输入端 S 决定输出 Y 等于 D_0 还是 D_1 ，真值表如表 4.4.11 所示。输出的逻辑函数式为

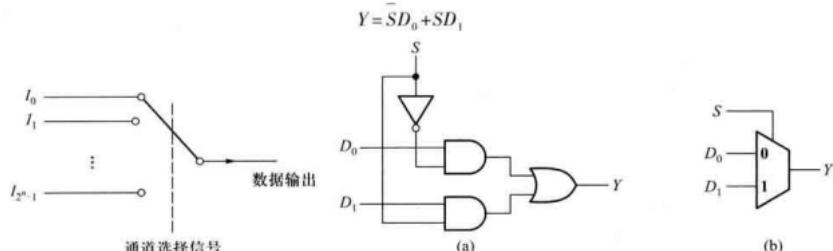


图 4.4.21 数据选择器示意图

图 4.4.22 2 选 1 数据选择器

(a) 逻辑电路 (b) 逻辑符号

表 4.4.11 2 选 1 数据选择器真值表

选择输入	输出
S	Y
0	D_0
1	D_1

4 选 1 数据选择器对 4 个数据源进行选择，需要两位选择输入 S_1S_0 。当 S_1S_0 取 00、01、10、11

时, 分别控制 4 个数据通道的开关。任何时候 S_1, S_0 只有一种可能的取值, 使对应的一路数据通过, 送达 Y 端。4 选 1 数据选择器的真值表如表 4.4.12 所示, 输出端逻辑函数式为

$$\begin{aligned} Y &= \bar{S}_1 \bar{S}_0 D_0 + \bar{S}_1 S_0 D_1 + S_1 \bar{S}_0 D_2 + S_1 S_0 D_3 \\ &= \bar{S}_1 (\bar{S}_0 D_0 + S_0 D_1) + S_1 (\bar{S}_0 D_2 + S_0 D_3) \\ &= \bar{S}_1 Y_0 + S_1 Y_1 \end{aligned}$$

用 3 个 2 选 1 数据选择器构成两级电路, 第 1 级两个数据选择器分别实现 $Y_0 = \bar{S}_0 D_0 + S_0 D_1$ 和 $Y_1 = \bar{S}_0 D_2 + S_0 D_3$ 。第 2 级实现 $Y = \bar{S}_1 Y_0 + S_1 Y_1$, 其电路结构及逻辑符号分别如图 4.4.23(a)、(b) 所示。

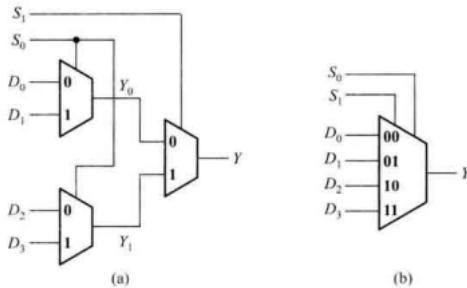


图 4.4.23 4 选 1 数据选择器逻辑图

(a) 由 2 选 1 数据选择器构成的 4 选 1 数据选择器 (b) 逻辑符号

表 4.4.12 4 选 1 数据选择器真值表

选择输入		输出	
S_1	S_0	Y	
0	0	0	D_0
0	1	1	D_1
1	0	0	D_2
1	1	1	D_3

同样原理, 可以构成更多输入通道的数据选择器。被选数据源越多, 所需选择输入端的位数也越多, 若选择输入端为 n , 可选输入通道数为 2^n 。

2. 典型数据选择器电路及应用

数据选择器的应用非常广泛, 并且是构成 FPGA 器件内部查找表(LUT)的基本单元。常用

的数据选择器有 2 选 1 数据选择器、4 选 1 数据选择器、8 选 1 数据选择器、16 选 1 数据选择器等。还有一些数据选择器具有三态输出功能，除了正常的 0 或 1 输出之外，当使能输入端为无效信号时，输出为高阻状态。利用这一特点，可以将多个数据选择器的输出端线与连在一起，共用一根数据传输线，而不会出现相互干扰问题。

(1) 用数据选择器实现逻辑函数

以 4 选 1 数据选择器为例，它有 2 个选择输入 S_1, S_0 和 4 个数据输入 D_0, D_1, D_2 和 D_3 ，其输出与输入的关系式可以写成

$$Y = \bar{S}_1 \bar{S}_0 D_0 + \bar{S}_1 S_0 D_1 + S_1 \bar{S}_0 D_2 + S_1 S_0 D_3 = \sum_{i=0}^3 m_i D_i$$

式中 m_i 是选择输入端 S_1, S_0 构成的最小项。数据输入作为控制信号，当 $D_i = 1$ 时，其对应的最小项 m_i 在表达式中出现，当 $D_i = 0$ 时，对应的最小项就不出现。利用这一点将函数变换为最小项表达式，函数的变量接入选择输入端，就可以实现组合逻辑函数。

例 4.4.7 试用数据选择器实现下列逻辑函数：

(1) 用 4 选 1 数据选择器实现 $L_0 = \bar{A}B + A\bar{B}$

(2) 用 4 选 1 数据选择器实现 $L_1 = AB + AC + BC$

(3) 用 2 选 1 数据选择器和必要的逻辑门实现 $L_1 = AB + AC + BC$

解：(1) 把所给的函数式变换为最小项表达式

$$L_0 = \bar{A}B + \bar{A}\bar{B} + A\bar{B} + AB = m_1 + m_2$$

变量 A, B 分别接 4 选 1 数据选择器的两个选择端 S_1 和 S_0 。 L_0 中出现的最小项 m_1, m_2 对应的数据输入端 D_1, D_2 都应该等于 1，而没有出现的最小项 m_0, m_3 对应的数据输入端 D_0, D_3 都应该等于 0。由此可画出逻辑图如图 4.4.24(a) 所示。

(2) 根据 L_1 的函数式列出真值表如表 4.4.13 所示。将变量 A, B 接入 4 选 1 数据选择器选择输入端 S_1 和 S_0 。将变量 C 分配在数据输入端。从表中可以看出输出 L_1 与变量 C 的关系。当 $AB = 00$ 时选通 D_0 ，而此时 $L_1 = 0$ ，所以数据端 D_0 接 0；当 $AB = 01$ 时选通 D_1 ，由真值表得此时 $L_1 = C$ ，即 D_1 应接 C ；当 AB 为 10 和 11 时， D_2 和 D_3 分别接 \bar{C} 和 1。因此得到逻辑电路如图 4.4.24(b) 所示。

(3) L_1 的真值表如表 4.4.14 所示。2 选 1 数据选择器只有一个选择输入端，将变量 A 接入选择输入端。根据表中 L_1 和 B, C 的关系，当 $A = 0$ 时，可以求出 $L_1 = BC$ ，即数据端 $D_0 = BC$ 。同理求出 $D_1 = B + \bar{C}$ 。也可以将逻辑函数变换为 $L_1 = \bar{A}BC + A(B + \bar{C})$ ，求得 D_0 和 D_1 。将变量 B, C 用逻辑门组合后接入数据端如图 4.4.24(c) 所示，这样可以实现变量数更多的逻辑函数。

由此可知，用一个具有 n 位选择输入的数据选择器，实现变量数不大于 $n+1$ 的逻辑函数时，每一个数据输入端可以接 0、1、单变量或它的非。当变量数大于 $n+1$ 时，可以用多个数据选择器扩展使用，也可以附加其他门电路后连接到数据输入端。

表 4.4.13 例 4.4.7(2) 的真值表

输入			输出	
A	B	C	L_1	
0	0	0	0	$L_1 = 0$
0	0	1	0	
0	1	0	0	$L_1 = C$
0	1	1	1	
1	0	0	1	$L_1 = \bar{C}$
1	0	1	0	
1	1	0	1	$L_1 = 1$
1	1	1	1	

表 4.4.14 例 4.4.7(3) 的真值表

输入			输出	
A	B	C	L_1	
0	0	0	0	
0	0	1	0	
0	1	0	0	$L_1 = BC$
0	1	1	1	
1	0	0	1	
1	0	1	0	
1	1	0	1	$L_1 = B + \bar{C}$
1	1	1	1	

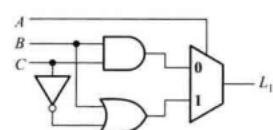
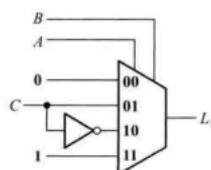
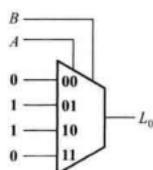


图 4.4.24 用数据选择器实现逻辑函数

(a) 用 4 选 1 数据选择器实现 L_0 (b) 用 4 选 1 数据选择器实现 L_1 (c) 用 2 选 1 数据选择器实现 L_1

(2) 用数据选择器构成查找表 LUT

构成 FPGA 基本单元的逻辑块主要是查找表 LUT。LUT 实质是一个小规模的存储器,以真值表的形式实现给定的逻辑函数。3 输入 LUT 的结构及逻辑符号如图 4.4.25 所示,由 7 个 2 选 1 数据选择器组成 3 级电路,第 1 级的数据输入端接存储单元(存储单元在第 7 章介绍),每个存储单元可以存放一个逻辑值 0 或 1,由编程决定取 0 还是 1。4 输入的 LUT 则由 15 个 2 选 1 数据选择器组成 4 级电路,4 个选择输入端对 16 个存储单元进行选择。当输入端增加时,LUT 使用的数据选择器呈几何增长。FPGA 中 LUT 的输入端数一般为 4 个。

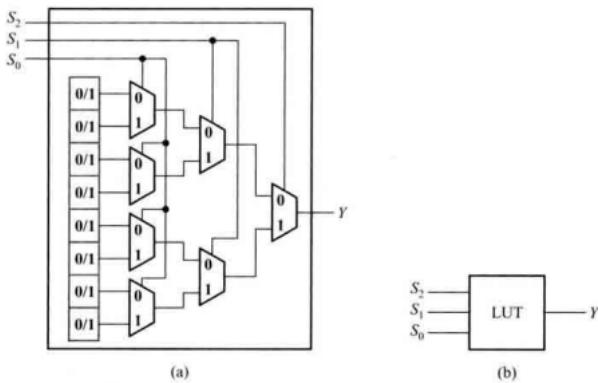


图 4.4.25 3 输入 LUT 结构图及符号

(a) 结构图 (b) 符号

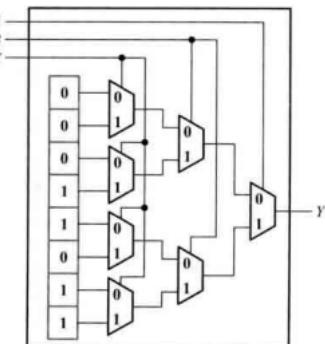
图 4.4.26 所示为 3 输入 LUT 实现例 4.4.8 中的函数 $L_1 = AB + \bar{AC} + BC$ 。根据表 4.4.13 所示真值表中 L_1 的取值,对存储单元编程,每个存储单元对应真值表中相应行的输出值。输入变量 A, B, C 接入选择端,根据它们的取值从 8 个存储单元中选择一个作为输出。

用 FPGA 实现大规模逻辑电路时,变量数比较多,需要将多个 LUT 扩展连接。首先要对复杂逻辑函数进行函数分解变换,以便能够满足单个 LUT 对变量数的要求。

例 4.4.8 试用 3 输入 LUT 实现逻辑函数 $L = \overline{BC} + \overline{ABC} + \overline{BCD} + \overline{ABD}$

解: 将逻辑函数进行变换

$$L = \overline{B}(\overline{C} + \overline{AD}) + B(\overline{AC} + \overline{CD})$$

图 4.4.26 用 LUT 实现给定的函数 L

$$\begin{aligned}
 &= \bar{B}(\overline{AC + \bar{CD}}) + B(AC + \bar{CD}) \\
 &= \bar{B}Y + BY
 \end{aligned}$$

该电路需要两个 3 输入 LUT，片(0)实现 $Y = \bar{C} + \bar{AD}$ ，片(1)实现 $L = \bar{BY} + \bar{BY}$ 。由于 $Y(A, C, D) = (\bar{C} + \bar{AD}) = \sum m(0, 1, 3, 4, 5)$ ，LUT(0)中数据输入存储单元 0, 1, 3, 4, 5 为 1，其余为 0。片(1)的 3 个选择输入端只接 B, Y 两个变量，最高位地址输入端没有使用。查找表将没有使用的输入端作为无关项对待， $L(x, B, Y) = \bar{BY} + \bar{BY} = \sum m(1, 2, 5, 6)$ 。

查找表没有使用的输入端可以接 0，也可以接 1，这里接 0。因此， $L(0, B, Y) = \bar{BY} + \bar{BY} = \sum m(1, 2)$ ，LUT(1)中存储单元 1, 2 为 1，其余为 0。用 LUT 实现的框图如图 4.4.27 所示。

对变量数比较多的复杂逻辑函数，很难用手工进行分解变换，用户可以借助 CAD 工具。另外，如何用尽量少的 LUT 实现给定的逻辑函数是一个优化问题，限于篇幅这里不进行讨论。

(3) 数据选择器、数据分配器与总线的连接

数据选择器又称为多路复用器，数据分配器又称为多路分接器（或解复用器）。由多路复用器、多路分配器和传输总线连接构成的信号传输系统示意图如图 4.4.28 所示。该系统将多个数据信号中的一路信号连接到总线，经过远距离传送后，由多路分配器再将总线上的数据分配到被选中的多个目的地之一。这种信息传输的基本原理在通信系统、计算机网络系统以及计算机内部各功能部件之间的信息转送等都有广泛的应用。

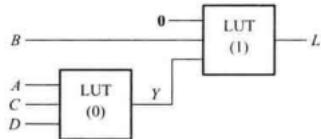


图 4.4.27 例 4.4.8 函数的 LUT 实现

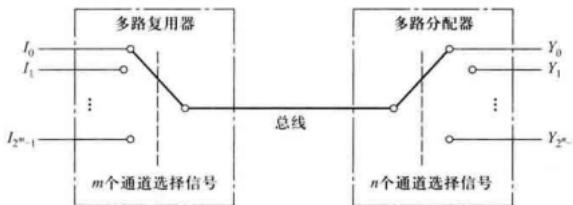


图 4.4.28 多路复用器、多路分配器、总线构成的信号传输系统示意图

(4) 集成数据选择器

数据选择器集成芯片有很多种，74HC151 是一种典型的 CMOS 集成数据选择器。它有 3 个地址输入端 S_2, S_1, S_0 ，可选择 $D_0 \sim D_7$ 共 8 个数据源，具有两个互补输出端，同相输出端 Y 和反相

输出端 \bar{Y} ,还有一个使能输入端 \bar{E} ,功能表如表 4.4.15 所示。当 $\bar{E} = 0$ 时数据选择器工作, $\bar{E} = 1$ 时数据选择器禁止工作,输出被封锁。

表 4.4.15 74HC151 的功能表

输入				输出	
使能 \bar{E}	S_2	S_1	S_0	Y	\bar{Y}
1	x	x	x	0	1
0	0	0	0	D_0	\bar{D}_0
0	0	0	1	D_1	\bar{D}_1
0	0	1	0	D_2	\bar{D}_2
0	0	1	1	D_3	\bar{D}_3
0	1	0	0	D_4	\bar{D}_4
0	1	0	1	D_5	\bar{D}_5
0	1	1	0	D_6	\bar{D}_6
0	1	1	1	D_7	\bar{D}_7

使用时,当数据选择器的大小不满足实际需求时,需要进行扩展。

位的扩展 上面所讨论的是一位数据选择器,如需要选择多位数据时,可由几个一位数据选择器并联组成,即将它们的使能端连在一起,相应的选择输入端连在一起。两位 8 选 1 数据选择器的连接方法如图 4.4.29 所示。当需要进一步扩充位数时,只需相应地增加器件的数目。

字的扩展 我们可以把数据选择器的使能端作为地址选择输入,将两片 74LS151 连接成一个 16 选 1 的数据选择器,其连接方式如图 4.4.30 所示。16 选 1 的数据选择器的地址选择输入有 4 位,其最高位 D 与一个 8 选 1 数据选择器的使能端连接,经过一反相器反相后与另一个数据选择器的使能端连接。低 3 位地址选择输入端 CBA 由两片 74HC151 的地址

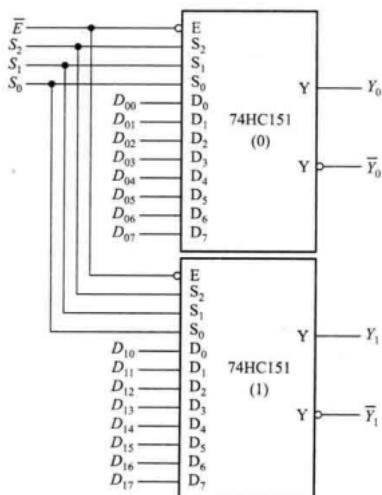


图 4.4.29 两位 8 选 1 数据选择器的连接方法

选择输入端相对应连接而成。

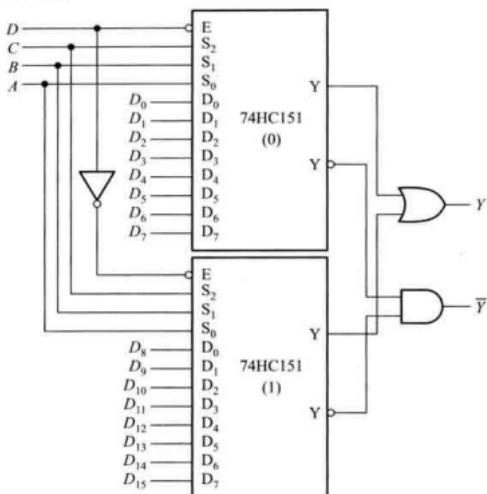


图 4.4.30 用两片 8 选 1 数据选择器连接成 16 选 1 数据选择器

复习思考题

- 4.4.3.1 试用十六进制数的方式写出 16 选 1 的数据选择器的各地址码。
- 4.4.3.2 构成 16 选 1 数据选择器需要多少个 2 选 1 数据选择器, 电路有几级? 试画出逻辑电路。
- 4.4.3.3 试说明 FPGA 中查找表 LUT 实现逻辑函数的原理。
- 4.4.3.4 用 32 选 1 数据选择器选择数据, 若选择的输入数据为 $D_{20}, D_{17}, D_{18}, D_{27}, D_{31}$, 试依次写出对应的地址码。

4.4.4 数值比较器

1. 数值比较器的定义及功能

在数字系统中, 特别是在计算机中常需要对两个数的大小进行比较。数值比较器就是对两个二进制数 A, B 进行比较的逻辑电路, 比较结果有 $A > B, A < B$ 以及 $A = B$ 三种情况。

(1) 1 位数值比较器

1 位数值比较器是多位比较器的基础。当 A 和 B 都是 1 位二进制数时, 它们只能取 0 或 1 两种值, 由此可写出 1 位数值比较器的真值表, 如表 4.4.16 所示。

由真值表得到如下逻辑表达式:

$$\begin{cases} F_{A>B} = \bar{A}\bar{B} \\ F_{A<B} = \bar{A}B \\ F_{A=B} = \bar{A}\bar{B} + AB \end{cases} \quad (4.4.6)$$

由以上逻辑表达式可画出图 4.4.31 所示的逻辑电路。

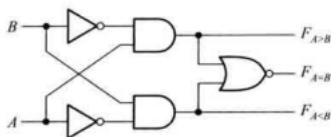


图 4.4.31 1 位数值比较器逻辑图

表 4.4.16 一位数值比较器真值表

输入		输出		
A	B	$F_{A>B}$	$F_{A<B}$	$F_{A=B}$
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

(2) 两位数值比较器

现在分析比较两位二进制数 A_1A_0 和 B_1B_0 的情况, 用 $F_{A>B}$ 、 $F_{A<B}$ 和 $F_{A=B}$ 表示比较结果。当高位 (A_1, B_1) 不相等时, 无需比较低位 (A_0, B_0), 高位比较的结果就是两个数的比较结果。当高位相等时, 两数的比较结果由低位比较的结果决定。利用 1 位数值的比较结果, 可以列出简化的真值表, 如表 4.4.17 所示。

表 4.4.17 两位数值比较器真值表

输入		输出				
A_1	B_1	A_0	B_0	$F_{A>B}$	$F_{A<B}$	$F_{A=B}$
$A_1 > B_1$		×		1	0	0
$A_1 < B_1$		×		0	1	0
$A_1 = B_1$		$A_0 > B_0$		1	0	0
$A_1 = B_1$		$A_0 < B_0$		0	1	0
$A_1 = B_1$		$A_0 = B_0$		0	0	1

由表 4.4.17 可以写出如下逻辑表达式:

$$\begin{aligned} F_{A>B} &= A_1\bar{B}_1 + (\bar{A}_1\bar{B}_1 + A_1B_1)A_0\bar{B}_0 \\ &= F_{A_1>B_1} + F_{A_1=B_1} \cdot F_{A_0>B_0} \\ F_{A<B} &= \bar{A}_1B_1 + (\bar{A}_1\bar{B}_1 + A_1B_1)\bar{A}_0B_0 \end{aligned} \quad (4.4.7)$$

$$= F_{A_1 > B_1} + F_{A_1 = B_1} \cdot F_{A_0 < B_0}$$

$$F_{A=B} = F_{A_1 = B_1} \cdot F_{A_0 = B_0}$$

根据上式画出逻辑图,如图 4.4.32 所示。电路利用了 1 位数值比较器的输出作为中间结果。它所依据的原理是,如果两位数 $A_1 A_0$ 和 $B_1 B_0$ 的高位不相等,则高位比较结果就是两数比较结果,与低位无关。这时,高位输出 $F_{A_1 > B_1} = 0$,使与门 G_1, G_2, G_3 均封锁,而或门都打开,低位比较结果不能影响或门,高位比较结果则从或门直接输出。如果高位相等,即 $F_{A_1 = B_1} = 1$,使与门 G_1, G_2, G_3 均打开,同时由 $F_{A_1 > B_1} = 0$ 和 $F_{A_1 < B_1} = 0$ 作用,或门也打开,低位的比较结果直接送达输出端,即低位的比较结果决定两数的大、小或者相等。

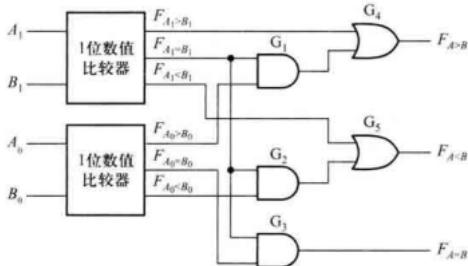


图 4.4.32 两位数值比较逻辑图

用以上方法可以构成更多位数值的比较器。

2. 典型数值比较器

常用的比较器有 4 位数值比较器、8 位数值比较器等。

(1) 4 位数值比较器

4 位数值比较器是对两个 4 位二进制数 $A_3 A_2 A_1 A_0$ 与 $B_3 B_2 B_1 B_0$ 进行比较, 比较原理与 2 位比较器的相同。从 A 的最高位 A_3 和 B 的最高位 B_3 进行比较, 如果它们不相等, 则该位的比较结果可以作为两数的比较结果。若最高位 $A_3 = B_3$, 则再比较次高位 A_2 和 B_2 , 余此类推。显然, 如果两数相等, 那么, 必须将比较进行到最低位才能得到结果。可以得出:

$$\begin{aligned} F_{A>B} &= F_{A_3 > B_3} + F_{A_3 = B_3} F_{A_2 > B_2} + F_{A_3 = B_3} F_{A_2 = B_2} F_{A_1 > B_1} + F_{A_3 = B_3} F_{A_2 = B_2} F_{A_1 = B_1} F_{A_0 > B_0} \\ &\quad + F_{A_3 = B_3} F_{A_2 < B_2} F_{A_1 = B_1} F_{A_0 = B_0} I_{A>B} \\ F_{A<B} &= F_{A_3 < B_3} + F_{A_3 = B_3} F_{A_2 < B_2} + F_{A_3 = B_3} F_{A_2 = B_2} F_{A_1 < B_1} + F_{A_3 = B_3} F_{A_2 = B_2} F_{A_1 = B_1} F_{A_0 < B_0} \\ &\quad + F_{A_3 = B_3} F_{A_2 = B_2} F_{A_1 = B_1} F_{A_0 = B_0} I_{A<B} \\ F_{A=B} &= F_{A_3 = B_3} F_{A_2 = B_2} F_{A_1 = B_1} F_{A_0 = B_0} I_{A=B} \end{aligned} \tag{4.4.8}$$

$I_{A>B}, I_{A<B}$ 和 $I_{A=B}$ 称为扩展输入端, 是来自低位的比较结果。扩展输入端与其他数值比较器的输出连接, 以便组成位数更多的数值比较器。若仅对 4 位数进行比较时, 应对 $I_{A>B}, I_{A<B}, I_{A=B}$ 进行适当处理, 即 $I_{A>B} = I_{A<B} = 0, I_{A=B} = 1$ 。

74HC85 是 4 位 CMOS 集成数值比较器。

(2) 数值比较器的位数扩展

数值比较器的扩展有串联和并联两种方式。图 4.4.33 表示两个 4 位数值比较器串联而成为一个 8 位的数值比较器。我们知道,对于两个 8 位数,若高 4 位相同,它们的大小则由低 4 位的比较结果确定。因此,低 4 位的比较结果应作为高 4 位的条件,即低 4 位比较器的输出端应分别与高 4 位比较器的 $I_{A>B}$ 、 $I_{A<B}$ 和 $I_{A=B}$ 端连接。

当位数较多且要满足一定的速度要求时,可以采取并联方式。图 4.4.34 表示 16 位并联数值比较器的原理图。由图可以看出这里采用两级比较方法,将 16 位按高低位次序分成四组,每组 4 位,各组的比较是并行进行的。将每组的比较结果再经 4 位比较器进行比较后得出结果。显然,从数据输入到稳定输出只需 2 倍的 4 位比较器延迟时间,若用串联方式,则 16 位的数值比较器从输入到稳定输出需要约四倍的 4 位比较器的延迟时间。

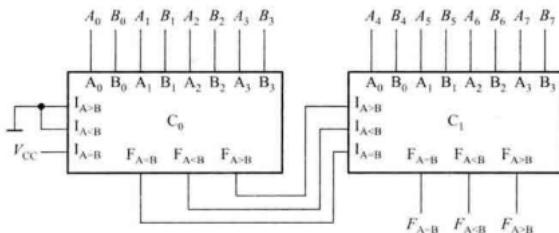


图 4.4.33 串联方式扩展数值比较器的位数

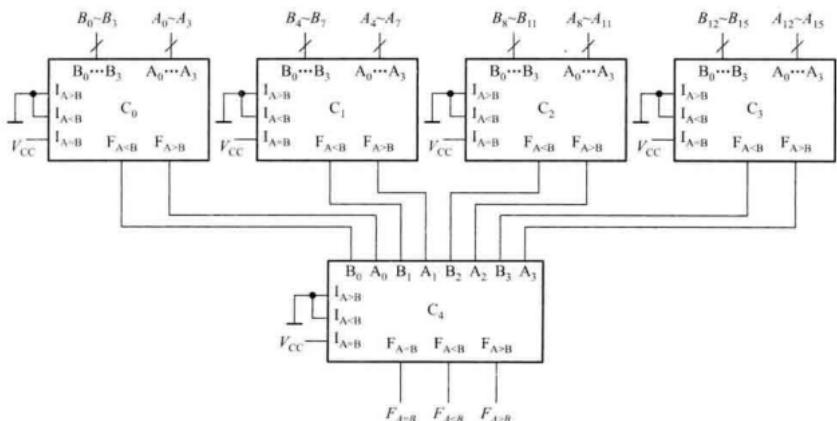


图 4.4.34 并联方式扩展数值比较器的位数

复习思考题

4.4.4.1 比较器的三个输入端 $I_{A>B}$ 、 $I_{A<B}$ 、 $I_{A=B}$ 有何作用？

4.4.4.2 用两个 4 位比较器串联，连接成 8 位数值比较器时，低位的 $I_{A>B}$ 、 $I_{A<B}$ 、 $I_{A=B}$ 端应作何处理？

4.4.5 算术运算电路

算术运算是数字系统的基本功能，更是计算机中不可缺少的组成单元。在第 1 章介绍了二进制数的算术运算，下面介绍实现加法运算和减法运算的逻辑电路。

1. 半加器和全加器

(1) 半加器

半加器和全加器是算术运算电路中的基本单元，它们是完成 1 位二进制数相加的一种组合逻辑电路。

如果只考虑了两个加数本身，而没有考虑低位进位的加法运算，称为半加。实现半加运算的逻辑电路称为半加器。两个 1 位二进制的半加运算可用表 4.4.18 所示的真值表表示，其中 A 、 B 是两个加数， S 表示和数， C 表示进位数。由真值表可得逻辑表达式

$$\begin{cases} S = \bar{A}\bar{B} + \bar{A}B \\ C = AB \end{cases} \quad (4.4.9)$$

表 4.4.18 半加器真值表

输入		输出	
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

由上述表达式可以得出由异或门和与门组成的半加器，如图 4.4.35(a) 所示，图 4.4.35(b) 是半加器的图形符号。

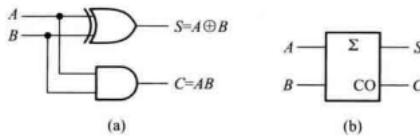


图 4.4.35 半加器

(a) 逻辑图 (b) 半加器符号

(2) 全加器

全加器能进行被加数、加数和来自低位的进位信号相加，并根据求和结果给出该位的进位信号。

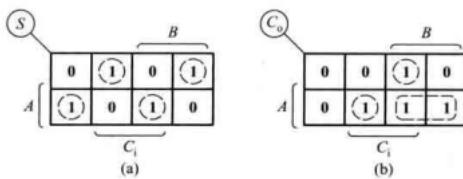
根据全加器的功能，可列出它的真值表，如表 4.4.19 所示。其中 A 和 B 分别是被加数及加数， C_i 为低位进位数， S 为本位和数（称为全加和）以及 C_o 为向高位的进位数。为了求出 S 和 C_o 的逻辑表达式，首先分别画出 S 和 C_o 的卡诺图，如图 4.4.36 所示，其中 C_o 的包围圈是为了便于利用 $A \oplus B$ 结果，得出下列表达式：

$$\left\{ \begin{array}{l} S = \overline{ABC}_i + \overline{ABC}_i + \overline{BC}_i + ABC_i \\ \quad = A \oplus B \oplus C_i \\ \\ C_o = AB + \overline{ABC}_i + \overline{BC}_i \\ \quad = AB + (A \oplus B) C_i \end{array} \right. \quad (4.4.10)$$

由式(4.4.10)可以画出 1 位全加器的逻辑图，如图 4.4.37(a)所示，它是由两个半加器和一个或门构成的，图 4.4.37(b)是它的图形符号。

表 4.4.19 全加器真值表

输入			输出	
A	B	C_i	S	C_o
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

图 4.4.36 全加器的 S 和 C_o 卡诺图

(a) S 的卡诺图 (b) C_o 的卡诺图

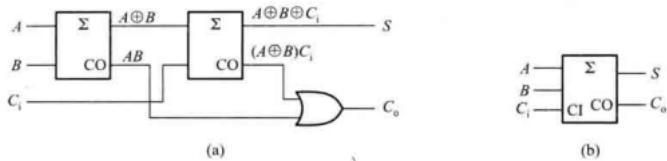


图 4.4.37 全加器
(a) 逻辑图 (b) 全加器符号

2. 多位数加法器

(1) 串行进位加法器

若有多位数相加，则可采用并行相加串行进位的方式来完成。例如，有两个 4 位二进制数 $A_3A_2A_1A_0$ 和 $B_3B_2B_1B_0$ 相加，可以采用 4 个全加器构成 4 位数加法器，其原理图如图 4.4.38 所示。将低位的进位输出信号接到高位的进位输入端，因此，任 1 位的加法运算必须在低 1 位的运算完成之后才能进行，这种进位方式称为串行进位。这种加法器的逻辑电路比较简单，但它的运算速度不高。为克服这一缺点，可以采用超前进位等方式。

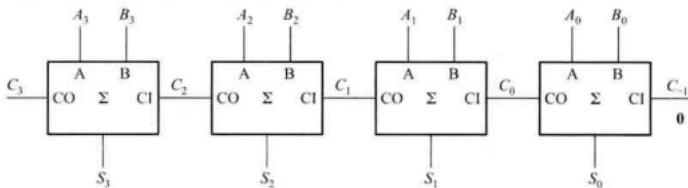


图 4.4.38 四位串行进位全加器

(2) 超前进位加法器

由于串行进位加法器的速度受到进位信号的限制，人们又设计了一种多位数超前进位加法逻辑电路，使每位的进位只由被加数和加数决定，而与低位的进位无关。下面介绍超前进位的概念。

由式(4.4.10)，并考虑多位数值相加时，全加器的和数 S_i 和进位 C_i 的逻辑表达式

$$S_i = A_i \oplus B_i \oplus C_{i-1} \quad (4.4.11)$$

$$C_i = A_i B_i + (A_i \oplus B_i) C_{i-1} \quad (4.4.12)$$

定义两个中间变量 G_i 和 P_i

$$G_i = A_i B_i \quad (4.4.13)$$

$$P_i = A_i \oplus B_i \quad (4.4.14)$$

当 $A_i = B_i = 1$ 时， $G_i = 1$ ，由式(4.4.9)得 $C_i = 1$ ，即产生进位，所以 G_i 称为产生变量。若 $P_i = 1$ ，则 $A_i B_i = 0$ ，由式(4.4.12)得 $C_i = C_{i-1}$ ，即 $P_i = 1$ 时，低位的进位能传送到高位的进位输出端，故

P_i 称为传输变量。这两个变量都与进位信号无关。

将式(4.4.13)和(4.4.14)代入式(4.4.11)和(4.4.12), 得

$$S_i = P_i \oplus C_{i-1} \quad (4.4.15)$$

$$C_i = G_i + P_i C_{i-1} \quad (4.4.16)$$

由式(4.4.16)得各位进位信号的逻辑表达式如下:

$$C_0 = G_0 + P_0 C_{-1} \quad (4.4.17a)$$

$$C_1 = G_1 + P_1 C_0 = G_1 + P_1 G_0 + P_1 P_0 C_{-1} \quad (4.4.17b)$$

$$C_2 = G_2 + P_2 C_1 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{-1} \quad (4.4.17c)$$

$$C_3 = G_3 + P_3 C_2 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_{-1} \quad (4.4.17d)$$

由式(4.4.17)可知, 因为进位信号只与变量 G_i 、 P_i 和 C_{-1} 有关, 而 C_{-1} 是向最低位的进位信号, 其值为 0, 所以各位的进位信号都只与两个加数有关, 它们是可以并行产生的。用与门和或门即可实现式(4.4.17)所表示的超前进位产生电路, 如图 4.4.39 所示。根据超前进位概念构成的 4 位加法器的结构示意图如图 4.4.40 所示。

超前进位加法器大大提高了运算速度。但是, 随着加法器位数的增加, 超前进位逻辑电路越来越复杂。74HC283 是 4 位超前进位加法器, 如果进行更多位数的加法, 则需要进行扩展。例如用 74HC283 实现 8 位二进制数相加, 两片 4 位加法器的连接方法如图 4.4.41 所示。该电路的级联是串行进位方式, 低位片(0)的进位输出连到高位片(1)的进位输入。当级联数目增加时, 会影响运算速度, 可采用并行进位的级联方式加以改进。

并行进位级联是指片与片之间采用超前进位连接方式。4 片加法器采用并行进位的连接示意图如图 4.4.42 所示。令加法器片(0)~(4)的产生变量和传输变量分别为 g_0 、 p_0 、 g_1 、 p_1 、 g_2 、 p_2 和 g_3 、 p_3 , 并且令式(4.4.17d)中

$$p_0 = P_3 P_2 P_1 P_0$$

$$g_0 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0$$

因此, 式(4.4.17d)可以写成

$$C_3 = G_3 + P_3 C_2 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_{-1}$$

$$= g_0 + p_0 C_{-1}$$

同样, 加法器片(1)、(2)和(3)的进位信号分别为

$$C_7 = g_1 + p_1 C_3 = g_1 + p_1 (g_0 + p_0 C_{-1})$$

$$C_{11} = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 C_{-1}$$

$$C_{15} = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 C_{-1}$$

从以上各个进位表达式可知, 进位信号 C_3 、 C_7 、 C_{11} 和 C_{15} 只由被加数、加数和 C_{-1} 决定, 而与其他低位的进位无关。它们只用于高位片的求和, 例如计算 $S_7 \sim S_4$ 需要 C_3 。

从输入加数、被加数到产生 g_i 、 p_i , 需要三级门延迟, 再经过两级门(进位产生电路的与门和或门)的延迟才能产生进位信号 C_3 、 C_7 、 C_{11} 和 C_{15} 。从进位信号 C_3 、 C_7 、 C_{11} 分别输入片(1)、(2)、(3), 到各片内部进位信号的建立, 需要再加上两级门延迟, 产生每一个和还需一个异或门的延

退。因此,完成16位数总共需要8级门的延迟,当位数增加时,延迟时间几乎不再增加。如果采用串行进位连接,每一片加法器有4级门延迟,4片共有16级门延迟,n片将有 $4n$ 级门延时。超前进位级联方式提高了电路的工作速度,但也增加了电路结构的复杂程度。

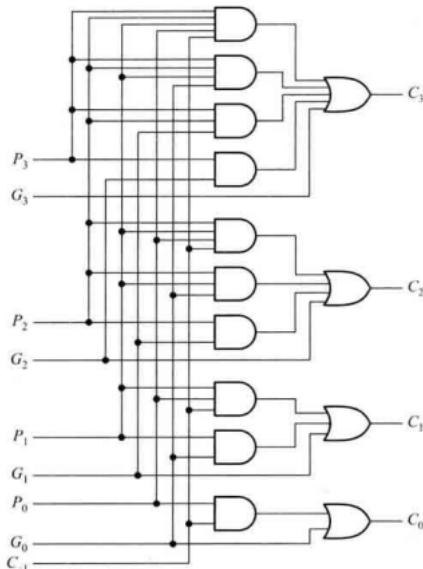


图 4.4.39 超前进位产生电路

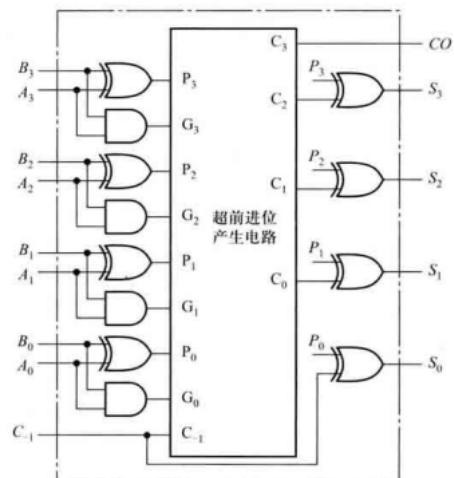


图 4.4.40 超前进位加法器结构示意图

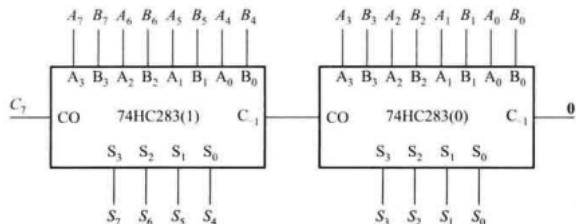


图 4.4.41 加法器串行进位扩展连接方式

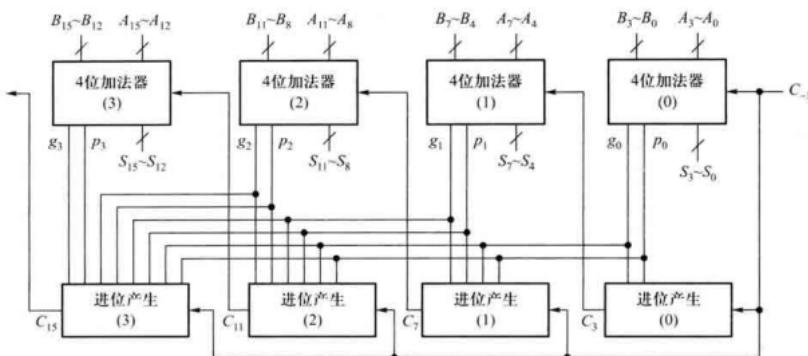


图 4.4.42 加法器并行进位扩展连接方式

3. 减法运算

由第1章介绍的二进制数算术运算可知,减法运算的原理是将减法运算转换成补码的加法运算进行的。如果前面介绍的加法运算器既能实现加法运算,又可实现减法运算,就可以简化数字系统结构。

若 n 位二进制的原码为 $N_{\text{原}}$, 则与它相对应的补码为

$$N_{\text{补}} = 2^n - N_{\text{原}} \quad (4.4.18)$$

补码与反码的关系式

$$N_{\text{补}} = N_{\text{反}} + 1 \quad (4.4.19)$$

设两个数 A, B 相减, 利用式(4.4.18)和式(4.4.19)可得

$$A - B = A + B_{\text{反}} - 2^n = A + B_{\text{反}} + 1 - 2^n \quad (4.4.20)$$

式(4.4.20)表明, A 减 B 可由 A 加 B 的补码并减 2^n 完成。4位减法运算电路如图 4.4.43(a)所示, 具体原理如下。

由 4 个反相器将 B 的各位反相(求反), 并将进位输入端 C_{-1} 接逻辑 1 以实现加 1, 由此求得 B 的补码。加法器相加的结果为 $(A + B_{\text{反}} + 1)$ 。

由于 $2^n = (10000)_2$, 相加结果与 2^n 相减只能由加法器进位输出信号完成。当进位输出信号为 1 时, 它与 2^n 的差为 0; 当进位输出信号为 0 时, 它与 2^n 的差值为 1, 同时还应发出借位信号。因此, 只要将进位信号反相即实现了减 2^n 的运算, 反相器的输出 V 为 1 时需要借位, 故 V 也可作为借位信号。下面分两种情况分析减法运算过程。

(1) $A - B \geq 0$ 的情况。设 $A = 0101, B = 0001$ 。

求补相加演算过程如下:

$$\begin{array}{r}
 \begin{array}{r} 0101 \\ 1110 \\ + \quad \quad 1 \\ \hline 10100 \end{array} \\
 \downarrow \\
 \begin{array}{l} \text{(进位反相)} \\ \text{(借位) } \rightarrow \end{array}
 \end{array}$$

直接作减法演算，则有

$$\begin{array}{r}
 \begin{array}{r} 0101 \\ - 0001 \\ \hline 0100 \end{array}
 \end{array}$$

比较两种运算结果，它们完全相同。在 $A-B \geq 0$ 时，所得的差值就是差的原码，借位信号为 0。

(2) $A-B < 0$ 的情况。设 $A = 0001, B = 0101$ 。

求补相加演算过程如下：

$$\begin{array}{r}
 \begin{array}{r} 0001 \\ 1010 \\ + \quad \quad 1 \\ \hline 01100 \end{array} \\
 \downarrow \\
 \begin{array}{l} \text{(进位反相)} \\ \text{(借位) } \rightarrow \end{array}
 \end{array}$$

直接作减法演算，则有

$$\begin{array}{r}
 \begin{array}{r} 0001 \\ - 0101 \\ \hline - 0100 \end{array} \\
 \downarrow \\
 \begin{array}{l} \text{(符号) } \rightarrow \end{array}
 \end{array}$$

比较两种运算结果可知，前者正好是后者的绝对值的补码，借位信号 V 为 1 时表示差值为负数， V 为 0 时差为正数。若要求差值以原码形式输出，则还需进行变换。由式(4.4.18)可知，将补码再求补得原码。

求补逻辑电路如图 4.4.43(b) 所示，它和图 4.4.43(a) 共同组成输出为原码的完整的 4 位减法运算电路。由图 4.4.43(a) 所得的差值输入到异或门的一个输入端，而另一端输入端由借位信号 V 控制。当 $V=1$ 时， $D_3 \sim D_0$ 反相，并与 $C_{-1}=1$ 相加，实现求补运算； $V=0$ 时， $D_3 \sim D_0$ 不反相，加法器也不实现加 1 运算，维持原码。

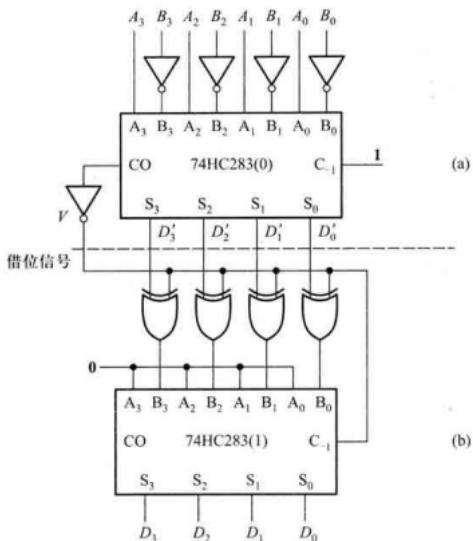


图 4.4.43 输出为原码的 4 位减法运算逻辑图

(a) 4 位减法运算逻辑图 (b) 输出求补逻辑图

复习思考题

- 4.4.5.1 什么是半加器？什么是全加器？
- 4.4.5.2 超前进位加法器和串行进位加法器的区别是什么？
- 4.4.5.3 图 4.4.38 所示 4 位串行进位加法器，完成一次运算有多少级门延迟？
- 4.4.5.4 说明反码和补码之间的关系。
- 4.4.5.5 简要说明由加补码完成减法运算的原理。

4.5 组合可编程逻辑器件

早期通用型的小规模门电路和中规模组合逻辑集成器件性能好、结构简单，是组成数字系统的基本构件。但是这些器件的逻辑功能是固定的，所包含的逻辑门数量较少。当构造一个大型复杂的数字系统时，过多的器件可能导致功耗高、占用空间大和系统可靠性差等问题。利用可编程逻辑器件 (Programmable Logic Device, PLD) 进行数字系统的设计，可以较好地解决以上问题。

PLD 是一种可以由用户定义和设置逻辑功能的器件。该类器件具有逻辑功能实现灵活、集成度高、处理速度快和可靠性高等特点。

4.5.1 PLD 的结构、表示方法及分类

1. PLD 的结构

PLD 的结构框图如图 4.5.1(a) 所示。与阵列和或阵列是它的基本组成部分，通过对与、或阵列的编程实现所需的逻辑功能。输入电路是由输入缓冲器组成，通过它可以得到驱动能力强，并且互补的输入变量送到与阵列。有些 PLD 器件的输出电路包含称为宏单元的电路。宏单元可以被编程为组合输出和时序输出两种方式，组合方式的或阵列经过三态门输出，时序方式的或阵列经过触发器和三态门输出。有些电路可以根据需要将输出反馈到与阵列的输入，以增加器件的灵活性。图 4.5.1(b) 所示为 PLD 基本电路结构，为简明，将输出三态门省略。

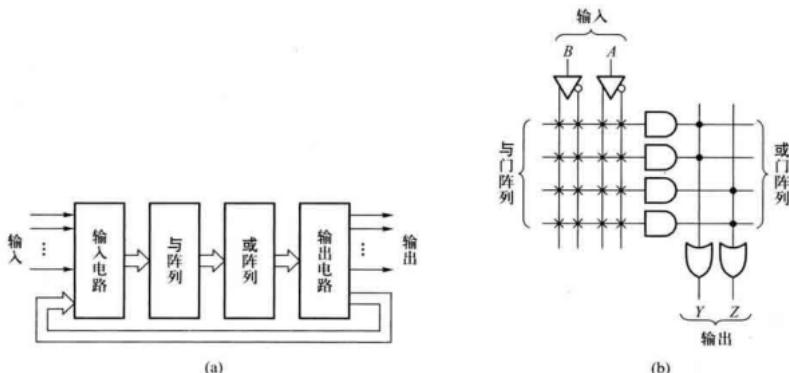


图 4.5.1 PLD 结构图
(a) 结构框图 (b) 基本电路结构

2. PLD 的表示方法

为了便于绘制与、或阵列的结构图，采用一种简化的表示方法，该方法的各种符号及含义如下。

(1) 连接方式

在图 4.5.1(b) 所示基本的 PLD 结构中，门阵列的每个交叉点称为“单元”，单元的连接方式共有三种情况，如图 4.5.2 所示：

- 硬线连接：硬线连接是固定连接，不可以编程改变。
- 可编程“接通”单元：依靠用户编程来实现“接通”连接。
- 可编程“断开”单元：编程实现断开状态。这种单元又称为被

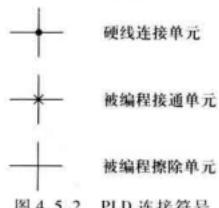


图 4.5.2 PLD 连接符号

编程擦除单元。

(2) 基本门电路的表示方式

PLD 中基本门电路符号如图 4.5.3 所示。图 4.5.3(a) 和图 4.5.3(b) 分别为与门 $L_1 = ABC$ 和或门 $L_2 = A + B + C$ 。由于 $L_3 = A \cdot \bar{A} \cdot B \cdot \bar{B} = 0$, 图 4.5.3(c) 给出了与门输出恒等于 0 的简化画法。图 4.5.3(d) 中与门的所有输入项均不接通, 保持“悬浮”的 1 状态, 即 $L_4 = 1$ 。图 4.5.3(e) 为具有互补输出的缓冲器。图 4.5.3(f) 为三态输出缓冲器。

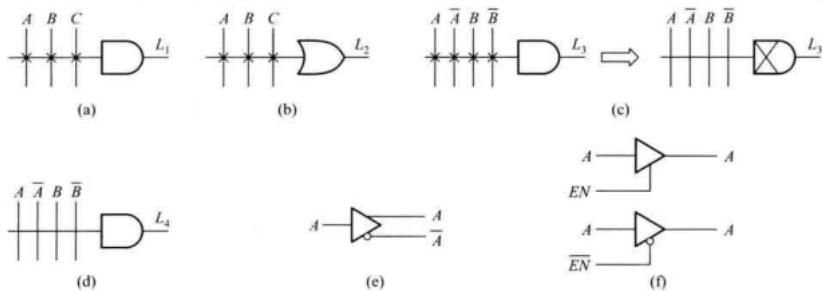


图 4.5.3 可编程器件中基本门电路的符号
 (a) 与门 (b) 或门 (c) 输出恒等于 0 的与门
 (d) 输出为 1 的状态 (e) 互补输出的缓冲器 (f) 三态输出缓冲器

(3) 编程连接技术

早期的 PLD 采用双极型连接技术, 对于图 4.5.4(a) 的逻辑电路, 单元的连接是由一个二极管与金属熔丝串接在一起, 如图 4.5.4(b) 所示。编程时, 用比工作电流大许多的电流, 将不需要连接的熔丝烧断。由于熔丝烧断后不能恢复, 这种方法只能进行一次编程。在门阵列中表示“单元”连接状况的阵列图, 称为熔丝图, 并且这个名称被沿用下来。

在 CMOS 的 PLD 中, 常采用可擦除的编程方法, 即可以对器件进行多次编程。可擦除 CMOS 技术用浮棚 MOS 管代替“熔丝”, 如图 4.5.4(c) 所示, 图中 MOS 管为浮棚 MOS 符号, 它可以是叠栅注入 MOS、浮棚隧道氧化层 MOS 或快闪叠栅 MOS 管中的任意一种。未经编程的浮棚 MOS 管, 与普通 N 沟道增强型 MOS 管一样, 当栅极加正常的逻辑高电平时, 管子处于导通状态, 否则截止。而经过编程处理的浮棚 MOS 管, 始终处于截止状态, 相当于“熔丝”断开一样。

根据图 4.5.4(a) 电路的逻辑要求, 经过编程将图 4.5.4(c) 中的 T_2, T_4 断开。由于输入信号接在 MOS 管的栅极, 所以输入信号低电平有效, 即 $A = 0, \bar{A} = 1$ 时, T_1 导通, 无论输入信号 C 为何值, $L = 0$ 。当 A, C 均为 1 时, T_1, T_3 均截止, $L = 1$ 。所以该电路实现与逻辑 $L = AC$ 。另外在图 4.5.4(c) 中, 上拉电阻实际是由 P 沟道 MOS 管构成的。

(4) 浮棚 MOS 管开关

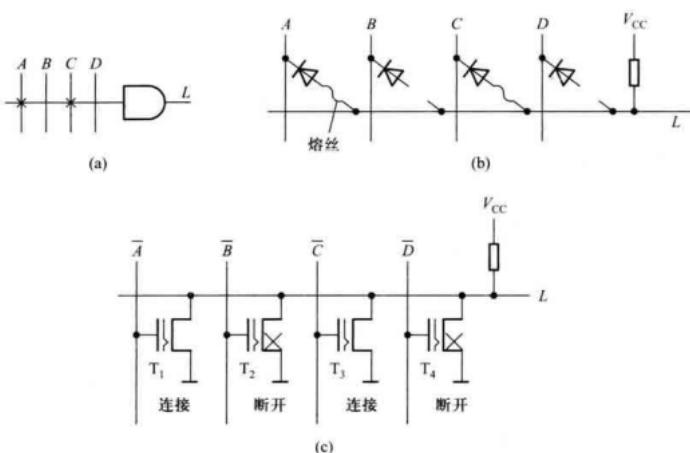


图 4.5.4 与门电路

(a) PLD 表示的与门 (b) 熔丝工艺的与门原理图 (c) CMOS 工艺的与门原理图

上面介绍了可擦除 PLD 采用浮栅 MOS 管,而浮栅 MOS 管又分为叠栅注入 MOS 管(Stacked-gate Injection Metal Oxide Semiconductor,SIMOS 管)、浮栅隧道氧化层 MOS 管(Floating-gate Tunnel Oxide,Flotox 管)和快闪(Flash)叠栅 MOS 管。不同的浮栅 MOS 管连接的 PLD,编程信息的擦除方法也不同。SIMOS 管连接的 PLD,采用紫外光照射擦除;Flotox 管和快闪叠栅 MOS 管,采用电擦除方法。

a. SIMOS 管开关

SIMOS 管的结构及符号如图 4.5.5(a)所示。它是一个 N 沟道增强型 MOS 管,有两个多晶硅栅极,控制栅 g_c 和浮置栅 g_f 。浮栅被绝缘的 SiO_2 包围着。编程处理前,浮栅上没有电荷,与普通 MOS 管一样。当控制栅加正常工作的高电平时,SIMOS 管处于导通状态,此时的开启电压为 V_{th} ,转移特性如图 4.5.5(b)所示。编程时漏源间加几十伏的正电压,漏极与衬底间的 PN 结产生雪崩击穿,若同时在控制栅加正脉冲电压,则雪崩产生的高能电子,在栅极电场的作用下,穿过 SiO_2 层注入到浮栅上。编程电压撤除后,因浮栅被绝缘层包围,注入的电子无放电通路,可以长期保留,此时 SIMOS 管的开启电压升高到 V_{t2} ,特性曲线右移如图 4.5.5(b)所示。因此,控制栅加正常逻辑高电平也不能达到其开启电压,SIMOS 管始终截止,相当于断开一样。

擦除的方法是用紫外线灯照射器件 20 min,则 SiO_2 层中将产生电子-空穴对,为浮栅上的电子提供泄放通道,使之放电,SIMOS 管恢复到编程前的状态。

b. Flotox 管开关

Flotox 管的结构及符号如图 4.5.6 所示。它与 SIMOS 管相似,不同的只是其浮栅与漏区间

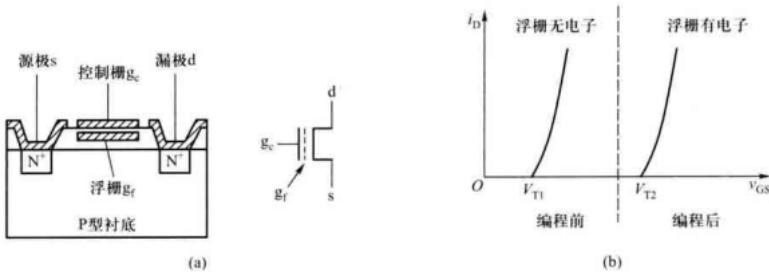


图 4.5.5 叠栅注入(SIMOS)管

(a) 结构及符号 (b) 浮栅上累积电子与开启电压的关系

有一个极薄的氧化层(仅 $0.01\text{ }\mu\text{m}$), 称为隧道区。当隧道区的电场强度足够大时, 漏区与浮栅之间便出现导电隧道, 在电场的作用下, 电子通过隧道形成电流, 这种现象称为隧道效应。同样, 编程处理前, 浮栅上没有电荷, 与普通 MOS 管一样。编程时源极、漏极均接地, 控制栅加 20 V 的脉冲电压, 隧道区产生强电场, 吸引漏区的电子通过隧道到达浮栅。撤除编程电压后, 浮栅上的电子可以长久保留, 此时 Flotox 管的开启电压升高, 在正常逻辑电平下, 始终关断。

如果需要擦除编程信息, 将 Flotox 管漏极加 20 V 正脉冲电压, 控制栅接地, 则浮栅上的电子在电场的作用下通过隧道回到漏区, Flotox 管恢复到编程前的状态, 从而达到擦除编程信息的目的。与 SIMOS 管相比, Flotox 管的编程和擦除都是通过在漏极和控制栅上加脉冲电压, 向浮栅注入和清除电荷的速度快、操作简单, 用户可以在电路板上实现在线操作。实际上编程和擦除是同时进行的, 每次编程时, 是以新的信息代替原来的信息。Flotox 管也广泛用于各种大规模可编程器件, 作为可编程开关。

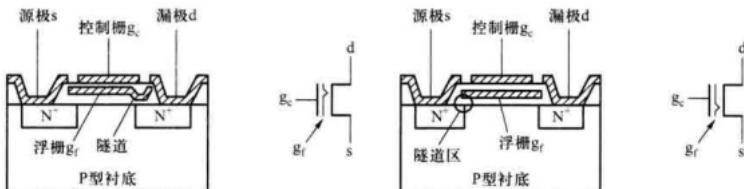


图 4.5.6 Flotox 管的结构及符号

图 4.5.7 快闪叠栅 MOS 管的结构及符号

c. 快闪叠栅 MOS 管开关

快闪叠栅 MOS 管的结构及符号如图 4.5.7 所示。它与 SIMOS 管结构类似。两者最大的区别在于快闪叠栅 MOS 管的浮栅到 P 型衬底间的氧化绝缘层比 SIMOS 管的更薄。因此, 其擦除方式是通过浮栅与源极之间超薄氧化层的电子隧道效应进行擦除。而 SIMOS 管的浮栅与源极

之间氧化层较厚,电场不足以产生隧道效应,所以用紫外线或X射线照射,使浮栅上的电子获得足够的能量回到衬底。

对快闪叠栅MOS管编程时,漏极接正电压(6V),源极接地,同时在控制栅加12V正脉冲电压,向浮栅注入电子的方式与SIMOS相同。编程后,快闪叠栅MOS管在正常逻辑条件下相当于断开。

快闪叠栅MOS管擦除编程信息的方式类似于Flotox管,将漏极加12V正脉冲电压,控制栅接地,即可利用隧道效应使浮栅放电而擦除编程信息。快闪叠栅MOS管既有SIMOS管结构简单、工作可靠的优点,又有Flotox管隧道效应带来的速度快、操作简单的特点,因而被广泛使用。

3. PLD的分类

PLD的分类方法有多种。按照PLD门电路的集成度,可以分为低密度和高密度器件,1000门以下为低密度,例如可编程只读存储器(Programmable Read Only Memory,PROM)、可编程逻辑阵列(Programmable Logic Array,PLA)、可编程阵列逻辑(Programmable Array Logic,PAL)和通用阵列逻辑(Generic Array Logic,GAL)等;1000门以上的为高密度,如CPLD、FPGA等。

按照PLD的结构体系,可分为简单PLD(如PAL、GAL),复杂可编程逻辑器件CPLD和现场可编程门阵列FPGA。CPLD和FPGA将在第8章介绍。

按照PLD中的与、或阵列是否可编程分为三种,分别如图4.5.8(a)、4.5.8(b)和4.5.8(c)所示,PROM的与阵列固定,或阵列可编程;PLA的与阵列、或阵列均可编程;PAL和GAL等的与阵列可编程,或阵列固定。

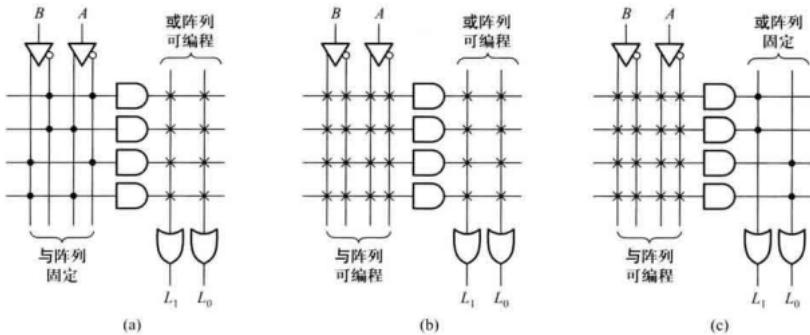


图4.5.8 PLD的分类

(a) PROM的基本电路结构 (b) PLA的基本电路结构 (c) PAL的基本电路结构

4.5.2 组合逻辑电路的PLD实现

任何组合逻辑关系都可以转换成与或表达式,因此通过PLD的与、或阵列可以实现任何一

个逻辑函数。

从图 4.5.8(a)可以看出,PROM 的与阵列是将输入变量的全部最小项译出来了,如果用它来实现逻辑函数,往往只用到一部分最小项,芯片的利用率不高,因此很少做为 PLD 器件使用。

1. 可编程逻辑阵列 PLA

PLA 就是为解决 PROM 实现逻辑函数时芯片利用率不高的问题而设计的。由于它的与、或阵列均可编程,所以将逻辑函数化简后再实现,可以有效地提高芯片的利用率。典型的集成 PLA(82S100)有 16 个输入变量、48 个乘积项、8 个输出端。

例 4.5.1 由 PLA 构成的逻辑电路如图 4.5.9 所示,试写出该电路的逻辑表达式,并确定其逻辑功能。

解:(1)由图 4.5.9 可知,该电路有 7 个与项,根据或阵列得到输出逻辑表达式:

$$L_0 = \overline{ABC} + \overline{ABC} + \overline{BC} + ABC$$

$$L_1 = AB + AC + BC$$

(2)列出真值表如表 4.5.1 所示。

(3)由真值表看出该电路实现全加器的功能,A、B、C 分别为加数、被加数和低位进位数。 L_0 为和数, L_1 为向高位的进位数。

尽管 PLA 的灵活性比 PROM 提高了,但是由于缺少高质量的支撑软件和编程工具,并且价格较贵,因而使用不广泛。

表 4.5.1 例 4.5.1 电路的真值表

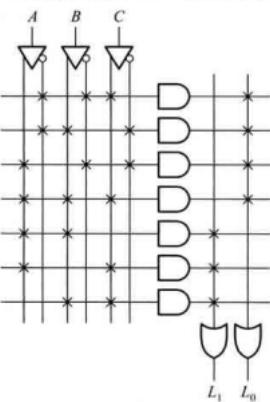


图 4.5.9 例 4.5.1 的 PLA 电路

输入			输出	
A	B	C	L_1	L_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

2. 可编程阵列逻辑 PAL

可编程阵列逻辑器件 PAL 是 20 世纪 70 年代后期推出的 PLD 器件, 早期采用双极型熔丝技术实现编程。除输入缓冲器外, PAL 由可编程的与阵列、固定的或阵列和输出电路组成。由于只有与阵列可编程, 因此 PAL 的编程相对简单。各种型号 PAL 的门阵列规模有大有小, 但基本结构类似。

图 4.5.10 所示为一简单的 PAL 结构图, 它有 4 组 10×3 位的可编程与阵列, 4 个输入信号和 1 个输出反馈信号产生 10 个与阵列的输入变量。每 3 个乘积项构成一组固定或阵列, 共有 4 组输出。

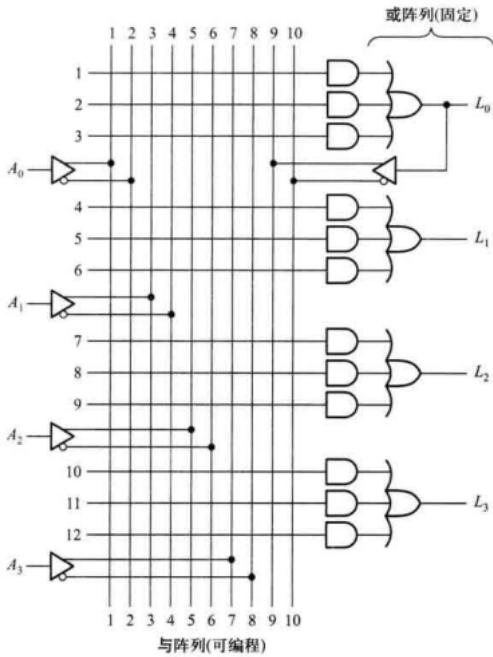


图 4.5.10 PAL 的基本电路结构图

PAL 与阵列所有交叉点都由熔丝连通(图 4.5.10 中所有交叉点上省略了“ \times ”), 编程时保留有用的熔丝, 断开无用的熔丝, 就得到所需的电路。

例 4.5.2 用图 4.5.10 所示 PAL 实现下列逻辑函数

$$\begin{cases} L_0 = ABC + \bar{A}\bar{B}\bar{C}\bar{D} \\ L_1 = \bar{A}\bar{B}C + A\bar{C}D + B\bar{C}\bar{D} \\ L_2 = \bar{A}\bar{B}\bar{C} + A\bar{B}C \\ L_3 = \bar{A}\bar{B}C + \bar{A}\bar{B}\bar{C}\bar{D} + B\bar{C}\bar{D} + A\bar{C}\bar{D} \end{cases}$$

解：这组逻辑函数均为简化的逻辑表达式，有 4 个输入变量，4 个输出。固定或阵列的每一个输出包含 3 个乘积项，而 $L_0 \sim L_2$ 三个表达式各包含 3 个以下乘积项，满足输出端对乘积项数目要求，可以直接编程实现。

L_3 表达式包含 4 个乘积项，不能直接编程实现，但其前两项正好为 L_0 ，即

$$\begin{aligned} L_3 &= \bar{A}\bar{B}C + \bar{A}\bar{B}\bar{C}\bar{D} + B\bar{C}\bar{D} + A\bar{C}\bar{D} \\ &= L_0 + \bar{B}\bar{C}D + A\bar{C}\bar{D} \end{aligned}$$

因此，将 L_0 反馈到输入端作为 L_3 的输入就可以实现。编程后的逻辑图如图 4.5.11 所示。

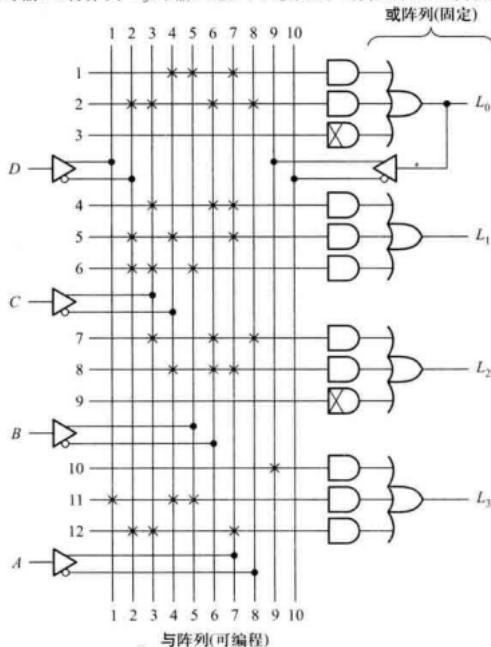


图 4.5.11 例 4.5.2 中编程后的 PAL 电路

为了扩展电路的功能并增加使用的灵活性,还有多种形式的输出、反馈电路结构。后期 PAL 器件采用了可擦除 CMOS 编程单元满足修改电路的需要。与中、小规模组合逻辑集成器件相比, PAL 的通用性好,速度和集成度均有所提高,设计和使用的灵活性得到改善。但是,PAL 器件输出电路结构的类型较多且不统一,给设计和使用带来不便,并且用 PAL 实现的时序电路非常有限。为克服这一缺陷,推出了采用可擦除 CMOS 编程技术的 GAL 器件,将在后面 6.6 节中介绍。

复习思考题

- 4.5.1 PLD 由哪几部分组成?
- 4.5.2 PLD 编程连接技术分哪几类?
- 4.5.3 浮棚 MOS 管编程信息的擦除方法主要有哪些?
- 4.5.4 列举四种类型的 PLD 器件,并简单说明其结构。

4.6 用 Verilog HDL 描述组合逻辑电路

2.5 节介绍了用 Verilog 语言描述电路的基本知识,本节将进一步讨论组合逻辑电路的行为级建模以及分模块、分层次的结构化建模方法。行为级建模用于描述一个电路输入、输出的功能特性,即所设计的电路是干什么的,而不是怎样去实现硬件电路。

4.6.1 组合逻辑电路的行为级建模

行为级建模一般使用 **always** 结构和过程赋值语句、条件语句(**if-else**)、多路分支语句(**case-endcase**)和**for** 循环语句等。

1. 条件语句

条件语句就是根据条件表达式的真假,确定下一步进行的运算。Verilog 语言中有 3 种形式的 **if** 语句,一般用法如下:

```
if ( condition_expr) true_statement;
或
if ( condition_expr) true_statement;
else false_statement;
或
if ( condition_expr1) true_statement1 ;
else if ( condition_expr2) true_statement2 ;
else if ( condition_expr3) true_statement3 ;
.....
else default_statement;
```

if 后面的条件表达式一般为逻辑表达式或关系表达式。执行 **if** 语句时,首先计算表达式的值,若结果为 0、x 或 z,按“假”处理;若结果为 1,按“真”处理,并执行相应的语句。注意,在第三种形式中,从第一个条件表达式 condition_expr1 开始依次进行判断,直到最后一个条件表达式被判断完毕,如果所有的表达式都不成立,才会执行 **else** 后面的语句。这种判断上的先后次序,本身隐含着一种优先级关系,在使用时应予注意。

例 4.6.1 根据表 4.4.11 所示的真值表,使用 **if-else** 语句对 4 选 1 数据选择器的行为(功能)进行描述。

解:假设这个程序模块的名称为 mux4to1_bh,输出端口为 Y,输入端口使用一个 4 位的向量 D 来代替原来的 D_3, D_2, D_1, D_0 ,选择输入端口使用一个 2 位的向量 S 来代替原来的 S_1, S_0 ,它的 4 个可能值用 2 位二进制数来表示。于是,得到 4 选 1 数据选择器的行为描述代码如下:

```
module mux4to1_bh(D,S,Y); //Verilog 1995 module port syntax
  input [3:0] D; //输入端口声明
  input [1:0] S; //输入端口声明
  output reg Y; //输出端口及变量的数据类型声明
  always @ (D,S) //电路功能描述
    if (S == 2'b00)      Y = D[0];
    else if (S == 2'b01)  Y = D[1];
    else if (S == 2'b10)  Y = D[2];
    else                  Y = D[3];
endmodule
```

注意,过程赋值语句只能给寄存器型变量赋值,因此,程序中将输出变量 Y 的数据类型定义成 **reg**。

2. 多路分支语句

case 语句是一种多分支条件选择语句,一般形式如下:

```
case (case_expr)
  item_expr1 : statement1;
  item_expr2 : statement2;
  .....
  default: default_statement; //default 语句可以省略
endcase
```

执行时,首先计算 **case_expr** 的值,然后依次与各分支项中表达式的值进行比较,如果 **case_expr** 的值与 **item_expr1** 的值相等,就执行语句 **statement1**,依次类推,如果 **case_expr** 的值与所有列出来的分支项的值都不相等,则执行语句 **default_statement**。

注意,(1)每个分支项中的语句可以是单条语句,也可以是多条语句。如果是多条语句,必须在多条语句的最前面写上关键词 **begin**,在这些语句的最后写上关键词 **end**,这样多条语句就组成了一个整体,称之为顺序语句块。在分支语句中,可以嵌入 **if-else** 语句,也可以嵌入另一条

case 语句。

(2) 每个分支项表达式的值必须各不相同,一旦判断到与某分支项的值相同并执行相应语句后, **case** 语句的执行便结束了。

(3) 如果某几个连续排列的分支执行同一条语句,则这几个分支项表达式之间可以用逗号分隔,将语句写在这几个分支项表达式的最后一个中。

例 4.6.2 对 4 选 1 数据选择器的行为进行描述。要求电路有一个使能输入端 En ,当 $En = 0$ 时允许数据选择器工作, $En = 1$ 时禁止数据选择器工作,输出为 0。

解: 可以混合使用 **if-else** 和 **case** 语句对使能输入端的 4 选 1 数据选择器进行描述。首先使用 **if-else** 语句对使能输入端 En 进行判断,然后使用 **case** 语句对输入的选择条件进行比较。2 位选择输入向量 S 的 4 个可能值可以用十进制数来表示。其代码如下:

```
module mux4to1_bh(          //Verilog 2001,2005 module port syntax
    input En;
    input [3:0] D,           //输入端口声明
    input [1:0] S,           //输入端口声明
    output reg Y            //输出端口及变量的数据类型声明
);
    always @ ( D, S, En)      //Verilog 2001,2005 syntax
    begin
        if (En == 1) Y = 0;   //当 En 为 1 时,输出为 0
        else               //当 En 为 0 时,实现选择器的功能
            case (S)
                2'd0: Y = D[0];
                2'd1: Y = D[1];
                2'd2: Y = D[2];
                2'd3: Y = D[3];
            endcase
    end
endmodule
```

例 4.6.3 根据表 4.4.9 所示的功能表,对共阴极的七段显示译码器的行为进行描述。

解: 根据电路的真值表对电路进行描述时使用 **case** 语句比较方便。程序①中用位拼接运算符将输出端口 a、b、c、d、e、f、g 拼接起来成为一个 7 位向量,将 4 位 BCD 码输入 D_3, D_2, D_1, D_0 拼接成一个 4 位向量。**case** 语句的分支项按顺序排列,最后一个分支项 **default** 把其他非 8421 BCD 码输入的情况设置成全零输出,可以省去所有无用输入码的显示。

注意, **case** 语句中列出的各个条件是不存在优先权差别的。

① 一个数中增加下画线,可以改善可读性。

```

module seg7_decoder(          //Verilog 2001,2005 module port syntax
    input LE,BL,LT,D3,D2,D1,D0, //输入端口声明
    output reg a,b,c,d,e,f,g   //输出端口及变量的数据类型声明
);
always @ ( * ) //Verilog 2001 语法,用 * 号表示电路的所有输入信号,参考 2.5.5 节的说明
begin
    if( LT==0)           |a,b,c,d,e,f,g|=7'b111_1111; //让显示器的 7 段都发光,显示 8
    else if ( BL==0)      |a,b,c,d,e,f,g|=7'b000_0000; //让显示器的 7 段都熄灭
    else if ( LE==1)      |a,b,c,d,e,f,g|=|a,b,c,d,e,f,g|; //锁存显示
    else
        case ( [D3,D2,D1,D0] )          //根据输入的 8421BCD 码,实现显示译码器的功能
            4'd0: |a,b,c,d,e,f,g|=7'b111_1110; //7e
            4'd1: |a,b,c,d,e,f,g|=7'b011_0000; //30
            4'd2: |a,b,c,d,e,f,g|=7'b110_1101; //6d
            4'd3: |a,b,c,d,e,f,g|=7'b111_1001; //79
            4'd4: |a,b,c,d,e,f,g|=7'b011_0011; //33
            4'd5: |a,b,c,d,e,f,g|=7'b101_1011; //5b
            4'd6: |a,b,c,d,e,f,g|=7'b001_1111; //1f
            4'd7: |a,b,c,d,e,f,g|=7'b111_0000; //70
            4'd8: |a,b,c,d,e,f,g|=7'b111_1111; //7f
            4'd9: |a,b,c,d,e,f,g|=7'b111_1011; //7b
            default: |a,b,c,d,e,f,g|=7'b000_0000; //非 8421 BCD 码输入时,不显示
        endcase
    end
endmodule

```

例 4.6.4 根据表 4.4.10 所示功能表,使用 case 语句对 1 路-4 路数据分配器的行为进行描述。

解: 该例给出了 case 语句的嵌套使用情况,在使能信号 E=1 的分支语句中,嵌入另一条 case 语句对选择条件 ([S1,S0]) 进行比较,从而确定电路的输出。

```

module Demux1_to_4 (          //Verilog 2001,2005 module port syntax
    output reg Y0,Y1,Y2,Y3,    //输出端口及变量的数据类型声明
    input In,                  //输入端口声明
    input S1,S0,E             //输入端口声明
);
always @ ( S1 or S0 or In or E)
    case( E )
        1'b1; case( [S1,S0] )
            2'b00: begin Y0 = In; Y1 = 1'bz; Y2 = 1'bz; Y3 = 1'bz; end

```

```

2'b01; begin Y0 = 1'bz; Y1 = In; Y2 = 1'bz; Y3 = 1'bz; end
2'b10; begin Y0 = 1'bz; Y1 = 1'bz; Y2 = In; Y3 = 1'bz; end
2'b11; begin Y0 = 1'bz; Y1 = 1'bz; Y2 = 1'bz; Y3 = In; end
endcase
default; begin Y0 = 1'bz; Y1 = 1'bz; Y2 = 1'bz; Y3 = 1'bz; end
endcase
endmodule

```

在 Verilog 语言中, **case** 语句还有另外两种形式, 用关键词 **cased** 和 **casez** 表示, 以便处理表达式值中含有逻辑值 **x** 和 **z** 的情况。**casez** 将比较双方表达式(分支项表达式和 **case_expr** 控制表达式)中出现 **z** 的位当作不用关心的位来处理, 在分支项表达式中所有为 **z** 的位也可以用? 来表示。而 **cased** 则认为表达式中所有的 **z** 和 **x** 都为无关项。

例 4.6.5 根据表 4.4.4 所示功能表, 对 8 线-3 线优先编码器的行为进行描述。

解: 程序首先使用 **cased** 语句描述了带使能控制端优先级编码器。这个模块与集成电路 CD4532 的功能类似, 当 **EI=1** 时, 才对 **cased** 语句中的条件项进行比较。当至少有一个输入数据为 1 时, 编码器的输出 **GS** 必然为 1 且 **EO** 必然为 0, 在 **cased** 语句执行之前, 这两个输出信号即被设置。若 8 个分支表达式的值没有一个与 I 的值匹配, 将会执行 **default** 分支后面语句, 将输出设置成默认值。注意, **cased** 忽略了比较双方表达式各个位中的 **x** 和 **z**, 根据比较的先后次序, 使得 **cased** 也存在隐含的优先级。

程序后面给出了优先编码器的另一种描述方式, 使用的 **if-else** 语句本身隐含着优先级关系。

```

module encoder8to3_bh(          //Verilog 2001,2005 module port syntax
    input EI,                  //输入端口声明
    input [7:0] I,              //输入端口声明
    output reg [2:0] Y,         //输出端口及变量的数据类型声明
    output reg GS,EO           //输出端口及变量的数据类型声明
);
always @ ( EI,I ) //Verilog 2001,2005 syntax
begin
    if ( EI==0 ) begin Y = 3'd0; GS=0; EO = 0;end
    else          //当 EI=1 时, 实现优先编码器的功能
        begin
            GS=1; EO = 0;//编码器的输入信号有效时, 设定 GS,EO 的输出值
            casex (I)      //逻辑值 x 表示输入信号中哪些位可以为任意值
                8'b1xxx xxxx; Y = 3'd7; //只要 I 的最高位等于 1, 就执行该语句
                8'b01xx xxxx; Y = 3'd6;
                8'b001x xxxx; Y = 3'd5;
                8'b0001 xxxx; Y = 3'd4;
            endcase
        end
    end
end

```

```

8'b0000_1xxx; Y = 3'd3;
8'b0000_01xx; Y = 3'd2;
8'b0000_001x; Y = 3'd1;
8'b0000_0001; Y = 3'd0;
default: begin Y = 3'd0; GS=0; EO = 1;end //处理编码器输入信号无效的情况
endcase
end
end
/* 下面给出另一种描述方式
begin Y = 3'd0; GS=0; EO = 0;end
    else //当 EI=1 时,实现优先编码器的功能
        begin
            GS=1; EO = 0; //编码器的输入信号有效时,设定 GS,EO 的输出值
            if(I[7]) Y = 3'd7; else //只要 I 的最高位等于 1,就执行该语句
                if(I[6]) Y = 3'd6; else
                if(I[5]) Y = 3'd5; else
                if(I[4]) Y = 3'd4; else
                if(I[3]) Y = 3'd3; else
                if(I[2]) Y = 3'd2; else
                if(I[1]) Y = 3'd1; else
                if(I[0]) Y = 3'd0;
            else begin Y = 3'd0; GS=0;EO = 1;end //处理编码器输入信号无效的情况
        end
    end
*/
endmodule

```

3. for 循环语句

for 语句的一般形式如下：

```
for ( initial_assignment ; condition ; step_assignment ) statement;
```

initial_assignment 为循环变量的初始值, condition 给出了循环的条件, 只要条件为真, 就执行过程赋值语句 statement, 当循环条件不成立时, 循环结束, 执行 **for** 后面的语句。step_assignment 为循环变量的步长, 每次迭代后, 循环变量将增加或减少一个步长。

注意,(1)循环变量必须为 **integer**(整数)型变量。

(2) 过程语句 statement 可以是过程赋值语句、条件语句和多路分支语句。也可以是另一条循环语句, 即循环语句可以嵌套。

例 4.6.6 用 Verilog 描述一个具有使能输入端的 3 线-8 线译码器的行为, 要求输出为低电平有效。

解: 该模块实现的功能与集成电路 74HC138 类似, 但只有一个高电平有效的使能信号。当使能信号 $En=1$ 时, 针对循环变量($k=0, 1, \dots, 7$)的变化, 重复执行 **if-else** 语句 8 次。每一次迭代实现一个不同的子电路, 例如 $A=0$ 时, 设置 $Y[0]=0$; $A=1$ 时, 设置 $Y[1]=0, \dots$, 其余的依次类推。

```
module decoder3to8_bh(          //Verilog 2001,2005 module port syntax
    input [2:0] A,           //输入端口声明
    input En,                //输入端口声明
    output reg [7:0] Y       //输出端口及变量的数据类型声明
);
    integer k;              //声明一个整型变量 k
    always @ (A,En)          //Verilog 2001,2005 syntax
    begin
        Y = 8'b1111_1111;    //设置译码器输出的默认值
        for(k = 0; k <= 7; k = k+1)//下面的 if-else 语句循环 8 次
            if ((En == 1) && (A == k))
                Y[k] = 0;        //当使能 En=1 时, 根据输入 A 进行译码
            else
                Y[k] = 1;        //处理使能无效或输入无效的情况
    end
endmodule
```

例 4.6.7 试用 **for** 循环语句对 n 位串行加法器的行为进行描述。

解: 程序使用 **for** 语句描述了一个通用的串行进位电路, 通过改变 **parameter** 语句中定义的参数 n 的数值, 该加法器可以对任意位宽数据进行加法运算。该例中 n 值被设置为 32, 所以此例代码实现的是 32 位加法器。

该模块的 **for** 循环由 **begin** 和 **end** 之间的两条语句定义, 这两条语句定义了循环变量为 k 的那一级加法器所对应的求和函数以及进位函数。 k 的范围是 0 到 $n-1$, 它的值每经过一次循环就递增 1。注意, 在 Verilog 中, 没有 C 语言中的“`++`”和“`--`”运算符, 因此 `k++` 和 `k--` 是错误的。

```
module addern_bh
#( parameter n=32);           //定义一个符号常量 n
)
//Verilog 2001,2005 module port syntax
input [n-1:0] A,B,           //输入端口声明
input Cin,                  //输入端口声明
output reg [n-1:0] SUM,      //输出端口及变量的数据类型声明
output reg Cout;             //输出端口及变量的数据类型声明
```

```

);
reg [n:0] C;           //声明中间变量
integer k;             //声明一个整型变量 k
always @ ( A,B,Cin)   //Verilog 2001,2005 syntax
begin
    C[0] = Cin;
    for(k = 0; k < n; k = k+1) //下面 begin…end 之间的语句循环 n 次
        begin
            SUM[k] = A[k] ^ B[k] ^ C[k];           //求和
            C[k+1] = (A[k] & B[k]) | (A[k] & C[k]) | (B[k] & C[k]); //求进位
        end
    Cout = C[n];           //输出进位信号
end
endmodule

```

例 4.6.8 根据表 4.4.10 所示功能表,对 2 选 1 数据选择器的行为进行描述。

解: 对 2 选 1 数据选择器的行为进行描述的方法很多,例 2.5.1 就介绍过。在这里介绍用条件运算符进行建模的方法。条件运算符可以用在连续赋值语句中,也可以用在 **always** 结构的过程语句中。

```

module mux2to1 (
    input D1,D0,S,
    output Y
);
    assign Y = S ? D1 : D0; //用连续赋值语句和条件运算符建模
endmodule
//下面给出另一种描述方式
module mux2to1 (
    input D1,D0,S,
    output reg Y
);
    always @ ( D1,D0,S ) //用 always 语句和条件运算符建模
        L = S ? D1 : D0;
endmodule

```

从上面的例子来看,数据流建模也是根据电路的逻辑功能进行描述,不必考虑电路组成以及元件之间的连接,也是属于电路行为描述范畴。下面再举两例,说明数据流建模方法。

例 4.6.9 根据表 4.4.5 所示的真值表,用数据流建模方法对 2 线-4 线译码器的行为进行描述。

解: 2 线-4 线译码器的逻辑功能由 4 个逻辑表达式组成的连续赋值语句进行描述,每一个

表达式与译码器的一个输出相对应。每一个表达式可以单独写成一条语句，后面用分号结尾，也可以像程序中示意的那样 4 个表达式合起来作为一条 **assign** 语句，但要注意，此时各个表达式之间以逗号作为分隔符，最后的一个表达式以分号作为一条语句的结束。

```
module decoder_df (          //Verilog 2001,2005 module port syntax
    input A1,A0,E,
    output [3:0] Y
);
    assign Y[0] = ~ (~A1 & ~A0 & ~E), //多条赋值语句列表的情况
    Y[1] = ~ (~A1 & A0 & ~E),
    Y[2] = ~ (A1 & ~A0 & ~E),
    Y[3] = ~ (A1 & A0 & ~E);
endmodule
```

例 4.6.10 试用数据流建模方法对 4 位加法器的行为进行描述。

解：4 位加法器的逻辑功能用一条连续赋值语句进行描述，加号“+”指明是二进制的加法，由于被加数和加数都是 4 位的，而低位来的进位为 1 位，所以运算的结果可能为 5 位，用 **| Cout, Sum |** 拼接起来表示。

```
module binary_adder (          //Verilog 2001,2005 module port syntax
    input [3:0] A,B,
    input Cin,
    output [3:0] SUM,
    output Cout
);
    assign |Cout,SUM| = A + B + Cin;
endmodule
```

4.6.2 分模块、分层次的电路设计

在电路设计中，可以将两个或多个模块组合起来描述电路逻辑功能，通常称之为分模块、分层次的电路设计，自上而下(Top-down)和自下而上(Bottom-up)是两种常用的设计方法。在自上而下设计中，先定义顶层模块，然后再定义顶层模块中用到的子模块。而在自下而上设计中，底层的各个子模块首先被确定下来，然后将这些子模块组合起来构成顶层模块。

以图 4.4.36 所示的全加器电路为例，4 位串行进位全加器被认为是一个顶层电路模块，它由 4 个全加器的子模块构成，而每个全加器又可以由两个半加器和一个或门构成，图 4.6.1 明确地表示了构成 4 位全加器的 3 个层次。如果用自上而下的设计方法，首先定义 4 位串行进位全加器这个顶层模块，然后定义 1 位的全加器，最后定义底层的半加器子模块。如果用自下而上的设计方法，则设计次序相反。

例 4.6.11 根据图 4.6.1 所示的层次结构框图，使用自下而上的方法描述 4 位全加器的逻

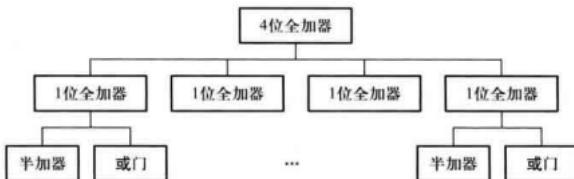


图 4.6.1 4 位全加器的层次结构框图

辑功能。

解：4 位全加器的结构化描述代码如下：

//一位半加器的描述(参考图 4.4.33)

```

module halfadder (S,C,A,B); //IEEE 1364—1995 Syntax
  input A,B;
  output S,C;
  xor (S,A,B);
  and (C,A,B);
endmodule
  
```

//一位全加器的描述(参考图 4.4.35)

```

module fulladder (Sum,Co,A,B,Ci);
  input A,B,Ci;
  output Sum,Co;
  wire S1,D1,D2; //内部节点信号声明
  halfadder H1(.B(B),.S(S1),.C(D1),.A(A)); //实例引用底层模块 halfadder
  halfadder H2(.A(S1),.B(Ci),.S(Sum),.C(D2)); //端口信号按照名称对应关联
  or g1(Co,D2,D1);
endmodule
  
```

//四位全加器的描述(参考图 4.4.36)

```

module _4bit_adder (S,C3,A,B,C_1);
  input [3:0] A,B;
  input C_1;
  output [3:0] S;
  output C3;
  wire C0,C1,C2; //声明模块内部的连接线
  fulladder U0_FA (S[0],C0,A[0],B[0],C_1); //实例引用模块 fulladder
  fulladder U1_FA (S[1],C1,A[1],B[1],C0); //端口信号按照位置顺序对应关联
  fulladder U2_FA (S[2],C2,A[2],B[2],C1);
  
```

```
fulladder U3_FA (S[3],C3,A[3],B[3],C2);
endmodule
```

例 4.6.12 所示的 4 位全加器总共使用了 3 个模块。首先,通过实例引用基本门级元件 **xor**、**and** 定义了底层的半加器模块 **halfadder**,接着实例引用两个半加器模块 **halfadder** 和一个基本或门元件 **or** 组合成为全加器模块 **fulladder**,最后实例引用 4 个 1 位的全加器模块 **fulladder** 构成 4 位全加器的顶层模块。可见,当一个模块被其他模块实例引用时,就形成了层次化结构。这种层次表明了引用模块与被引用模块之间的关系,引用模块称为父模块,被引用的模块称为子模块,即包含子模块的模块是父模块。

模块实例引用语句的格式如下:

```
module_name instance_name (port_associations);
```

其中,**module_name** 为设计模块名,**instance_name** 为实例引用名,**port_associations** 为父模块与子模块之间端口信号的关联方式,通常有位置关联法和名称关联法。

父模块引用子模块时,通过模块名完成引用过程,且实例引用名不能省略。例 4.6.11 所示的 4 位全加器顶层模块 **_4bit_adder** 是用 4 条实例引用语句描述的,每一条语句的开头都是被引用模块的名字 **fulladder**,后面紧跟着的是实例引用名(例如,**U0_FA**、**U1_FA** 等),且实例引用名在父模块中必须是唯一的。

父模块与子模块的端口信号是按照位置(端口排列次序)对应关联的。父模块引用子模块时可以使用一套新端口,也可以使用同名的旧端口,但必须注意端口的排列次序。例如,在引用语句 **fulladder U0_FA (S[0],C0,A[0],B[0],C_1)** 中,信号 **S[0]** 与模块(**fulladder**)的端口 **S** 连接,信号 **C0** 与模块的端口 **C0** 连接,信号 **A[0]** 与模块的端口 **A** 连接,信号 **B[0]** 与模块的端口 **B** 连接,信号 **C_1** 与模块的端口 **C1** 连接。对于端口较少的 Verilog 模块,使用这种方法比较方便。

当端口较多时,建议使用名称关联的方法。图 4.6.2 是上面第 2 个模块调用半加器时的情况,带有圆点的名称(如 **S.**、**C** 等)是定义子模块时使用的端口名称,也称为形式名称,类似于 C 语言中子函数的形参,写在圆括号内的名称(如 **Sum**、**D2** 等)是父模块中使用的新名称,也称为实际名称,类似于 C 语言中的实参。用这种方法实例引用子模块时,直接通过名称建立模块端口的连接关系,不需要考虑端口的排列次序。另外,端口关联时允许某些端口不连接,方法是:让不需要连接的端口位置为空白即可。在进行逻辑综合时,未连接输入端口的值被设置为高阻态(**z**),未连接的输出端口表示该端口没有被使用。

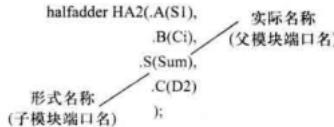


图 4.6.2 模块端口的名称关联法

在顶层模块 **_4bit_adder** 中,每一个 **fulladder** 后面的端口名称隐含地表示了这些子模块是如

何相互连接在一起的。例如,子模块 U0_FA 中的进位输出 CO 被连接到子模块 U1_FA 中作为进位输入信号。

关于模块引用的几点注意事项:

(1) 模块只能以实例引用的方式嵌套在其他模块内,嵌套的层次是没有限制的。但不能在一个模块内部使用关键词 **module** 和 **endmodule** 去定义另一个模块,也不能以循环方式嵌套模块,即不能在 **always** 语句内部引用子模块。

(2) 实例引用的子模块可以是一个设计好的 Verilog 设计文件(即一个设计模块),也可以是 FPGA 元件库中一个元件或嵌入式元件功能块,或者是用别的 HDL 语言(如 VHDL、AHDL 等)设计的元件,还可以是 IP(Intellectual Property,知识产权)核模块。

(3) 在一条实例引用子模块的语句中,不能一部分端口用位置关联,另一部分端口用名称关联,即不能混合使用这两种方式建立端口之间的连接。

小 结

- 根据电路的结构和工作特点,将数字电路分为两大类,即组合逻辑电路和时序逻辑电路。
- 组合逻辑电路的特点是,其输出状态在任何时刻只决定于同一时刻的输入状态。它可由逻辑门电路以及可编程器件(PLD)等组成。组合逻辑电路在形式和功能上种类繁多,但其分析方法和设计方法具有共同特点。因此,学习的重点是掌握一般的分析方法和设计方法。
- 分析组合逻辑电路的目的是确定已知电路的逻辑功能,其步骤大致是:写出各输出端的逻辑表达式→化简和变换逻辑表达式→列出真值表→确定功能。
- 设计组合逻辑电路的目的是根据提出的实际问题,设计出逻辑电路。设计步骤大致是:明确逻辑功能→列出真值表→写出逻辑表达式→逻辑化简和变换→画出逻辑图。
- 组合逻辑电路优化实现的技术包括:多输出的化简变换,用提取公因子或函数分解来减少扇入数。
- 典型的组合逻辑电路包括编码器、译码器、数据选择器、数值比较器、加法器和算术运算单元等。这些组合逻辑电路除了具有其基本功能外,通常还具有输入使能、输出使能、输入扩展、输出扩展功能,使其功能更加灵活,便于构成较复杂的逻辑系统。在用可编程逻辑器件设计逻辑系统时,这些典型的组合逻辑电路经常被当做特殊功能模块调用。
- 逻辑函数的优化实现是用指定芯片中特定资源实现逻辑函数。优化实现可以使用门电路,常用组合逻辑电路模块或可编程逻辑器件。用基本门电路进行优化实现时,需要对逻辑表达式进行变换,以满足芯片资源的要求。用可编程逻辑器件时,优化实现过程由 EDA 工具自动完成。组合逻辑电路模块可以作为 EDA 工具的库元件,由高层电路设计调用。用组合逻辑电路模块进行优化实现时,所应用的原理和步骤与用门电路时基本一致,但也有其特殊之处。
 - ① 对逻辑表达式的变换与化简应尽可能与组合逻辑模块的形式一致,而不是尽量简化。
 - ② 在满足设计要求的前提下,尽量选用简单的模块,模块数也尽可能少。

③ 如果只需一个模块就可以满足要求，则需要对有关使能、扩展或者多余输入端等作适当的处理。如果一个模块不能满足要求，则需要进行扩展，直接将若干个模块组合或者由适当的逻辑门将若干个模块组合起来。

- 电路在信号电平变化的瞬间经常会产生竞争-冒险现象，在电路设计过程中要采取措施避免产生竞争-冒险。

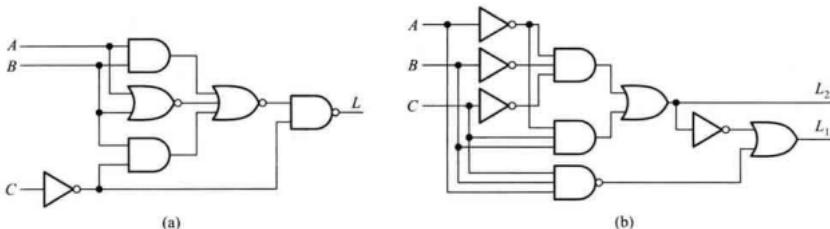
- 可编程逻辑器件(PLD)由用户定义和设置逻辑功能，可以实现各种组合逻辑电路。其特点是结构灵活、集成度高、速度快和可靠性高等。

- 用 Verilog 对组合逻辑电路建模时有三种不同的描述风格，即行为级建模、数据流建模和分模块分层次的结构化建模，其中数据流建模方式也属于行为建模的范畴。而结构化的建模方式通常用于规模较大的复杂逻辑电路。

习 题

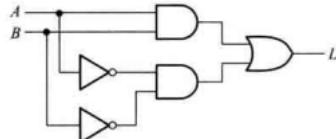
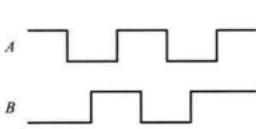
4.1 组合逻辑电路的分析

4.1.1 写出图题 4.1.1 所示电路对应的真值表。



图题 4.1.1

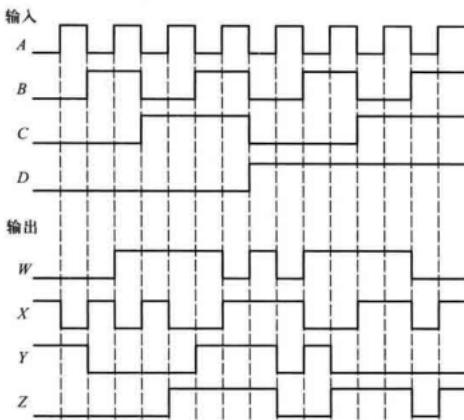
4.1.2 组合逻辑电路及输入波形(A,B)如图题 4.1.2 所示，试写出输出端的逻辑表达式并画出输出波形。



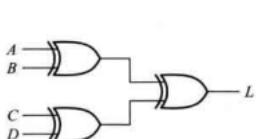
图题 4.1.2

4.1.3 设有四种组合逻辑电路，它们的输入波形(A,B,C,D)如图题 4.1.3 所示，其对应的输出波形为 W、X、Y、Z，试分别写出它们的简化逻辑表达式。

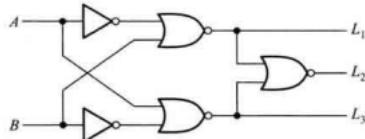
4.1.4 试分析图题 4.1.4 所示逻辑电路的功能。



图题 4.1.3



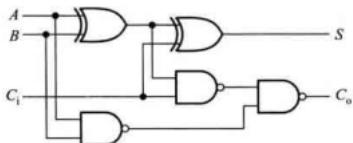
图题 4.1.4



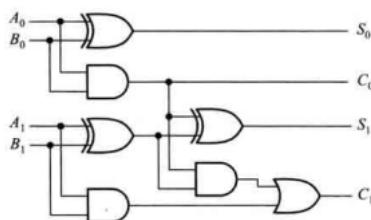
图题 4.1.5

4.1.5 逻辑电路如图题 4.1.5 所示,试分析其逻辑功能。

4.1.6 试分析图题 4.1.6 所示逻辑电路的功能。



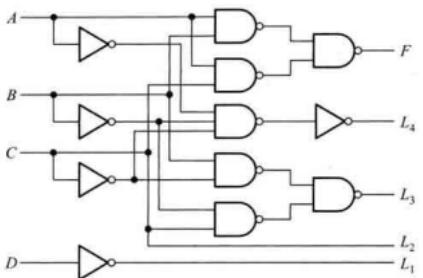
图题 4.1.6



图题 4.1.7

4.1.7 分析图题 4.1.7 所示逻辑电路的功能。

4.1.8 分析图题 4.1.8 所示逻辑电路的功能。



图题 4.1.8

4.2 组合逻辑电路的设计

4.2.1 试用 2 输入与非门设计一个 3 输入的组合逻辑电路。当输入的二进制码小于 3 时,输出为 0;输入大于等于 3 时,输出为 1。

4.2.2 试设计一个 4 位的奇偶校验器,即当 4 位数中有奇数个 1 时输出为 0,否则输出为 1。可以采用各种逻辑功能的门电路来实现。

4.2.3 试设计一个具有控制端 C 的 4 输入、4 输出逻辑电路。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。可以采用各种逻辑功能的门电路来实现。

4.2.4 试设计一可逆的 4 位码转换电路。当控制信号 $C=1$ 时,它将 8421 码转换为格雷码; $C=0$ 时,它将格雷码转换为 8421 码。可以采用任何门电路来实现。

4.2.5 试设计一组合逻辑电路,能够对输入的 4 位二进制数进行求反加 1 的运算。可以采用任何门电路来实现。

4.2.6 某足球评委会由一位教练和三位球迷组成,对裁判员的判罚进行表决。当满足以下条件时表示同意:有三人或三人以上同意,或者有两人同意,但其中一人是教练。试用 2 输入与非门设计该表决电路。

4.2.7 某雷达站有三部雷达 A, B, C ,其中 A 和 B 功率消耗相等, C 的功率是 A 的两倍。这些雷达由两台发电机 X 和 Y 供电,发电机 X 的最大输出功率等于雷达 A 的功率消耗,发电机 Y 的最大输出功率是 X 的 3 倍。要求设计一个逻辑电路,能够根据各雷达的启动和关闭信号,以最节约电能的方式启、停发电机。可以采用任何门电路来实现。

4.2.8 某火车站有特快、直快和慢车三种类型的客运列车进出,试用 2 输入与非门和反相器设计一个指示列车等待进站的逻辑电路,3 个指示灯一、二、三号分别对应特快、直快和慢车。列车的优先级别依次为特快、直快和慢车,要求当特快列车请求进站时,无论其他两种列车是否请求进站,一号灯亮。当特快没有请求,直快请求进站时,无论慢车是否请求,二号灯亮。当特快和直快均没有请求,而慢车有请求时,三号灯亮。

4.2.9 设计一个电路,能实现表题 4.2.9 所示的逻辑功能,可以采用任何门电路来实现。化简变换时考虑多输出函数的公共乘积项,以减少门的数目。

表题 4.2.9 的真值表

A	B	C	L_1	L_2
0	0	0	0	1
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	0	0

4.2.10 设计一 2 位二进制数相加的逻辑电路,可以用任何门电路实现。化简变换时考虑多输出函数的公共乘积项,以减少门的数目。提示:

$$\begin{array}{r} A_1 \quad A_0 \\ + B_1 \quad B_0 \\ \hline C_1 \quad S_1 \quad S_0 \end{array}$$

A_1 、 A_0 和 B_1 、 B_0 分别为被加数和加数, S_1 、 S_0 为相加的和, C_1 为进位位。

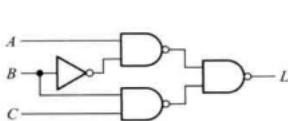
4.3 组合逻辑电路中的竞争-冒险

4.3.1 判断下列逻辑函数是否有可能产生竞争-冒险,如果可能应如何消除。

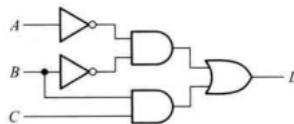
- (1) $L_1(A, B, C, D) = \sum m(5, 7, 13, 15)$
- (2) $L_2(A, B, C, D) = \sum m(5, 7, 8, 9, 10, 11, 13, 15)$
- (3) $L_3(A, B, C, D) = \sum m(0, 2, 4, 6, 8, 10, 12, 14)$
- (4) $L_4(A, B, C, D) = \sum m(0, 2, 4, 6, 12, 13, 14, 15)$

4.3.2 判断图题 4.3.2 所示电路是否会产生竞争-冒险。

4.3.3 判断图题 4.3.3 所示电路在什么条件下产生竞争-冒险,怎样修改电路能消除竞争-冒险?



图题 4.3.2



图题 4.3.3

4.3.4 画出下列逻辑函数的逻辑图,电路在什么条件下产生竞争-冒险,怎样修改电路能消除竞争-冒险。

$$L(A, B, C) = (A+B)(B+C)$$

4.4 若干典型的组合逻辑电路

4.4.1 优先编码器 CD4532 的输入端 $I_1 = I_3 = I_5 = 1$, 其余输入端均为 0, 试确定其输出 $Y_2 Y_1 Y_0$ 。

4.4.2 试用与非门设计一 4 输入的优先编码器,要求输入、输出及工作状态标志均为高电平有效。列出真

值表,画出逻辑图。

4.4.3 优先编码器 74HC147 的功能表如表题 4.4.3 所示,试用 74HC147 和适当的门构成输入为低有效的 $\bar{I}_0 \sim \bar{I}_4$, 输出为低电平有效的 8421BCD 码,并具有编码输出标志的编码器。

表题 4.4.3 优先编码器 74HC147 功能表

4.4.4 试用 74HC147 设计键盘编码电路，十个按键分别对应十进制数 0~9，编码器的输出为 8421BCD 码。要求按键 9 的优先级别最高，并且有工作状态标志，以说明没有按键按下和按键 0 按下两种情况。

4.4.5 试用两片 74HC138 构成 4 线-16 线译码器, 输入为 4 位二进制码 $B_3B_2B_1B_0$, 输出为 $\bar{L}_0 \sim \bar{L}_{15}$ 为低电平有效信号。

4.4.6 译码器的真值表如表题 4.4.6 所示, 试用 74HC138 实现该译码器。

表题 4.4.6 的真值表

4.4.7 2 线-4 线译码器 74x139 的输入为高电平有效, 使能输入及输出均为低电平有效。试用 74x139 构成 4 线-16 线译码器。

4.4.8 用译码器 74HC138 和适当的逻辑门实现函数 $F = \overline{ABC} + \overline{ABC} + \overline{ABC} + ABC$ 。

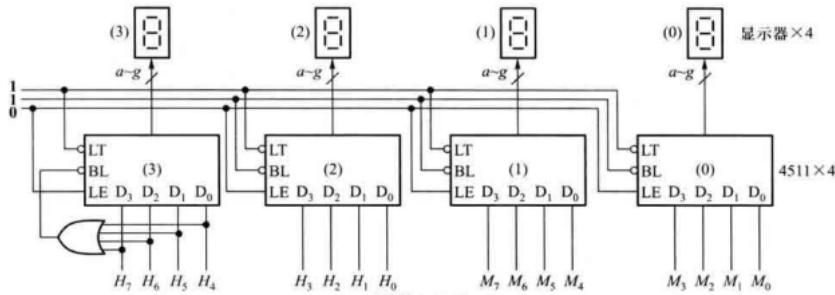
4.4.9 试用一片 74HC138 实现函数 $L(A, B, C, D) = \overline{ABC} + ACD$ 。

4.4.10 应用 74HC138 和其他逻辑门设计一地址译码器, 要求地址范围是十六进制 00~3F。

4.4.11 指出题 4.4.10 中对应十六进制地址码 07, 0E, 13, 2C, 3B 的输入。

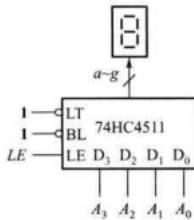
4.4.12 试用译码器 74HC138 和必要的与非门, 设计一个乘法器电路, 实现两位二进制数相乘, 并输出结果。

4.4.13 由 74HC4511 构成 24 小时及分钟的译码电路如图题 4.4.13 所示, 试分析小时高位是否具有零熄灭功能。

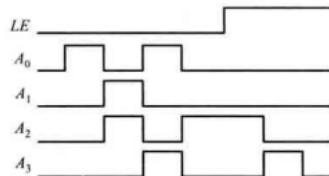


图题 4.4.13

4.4.14 七段显示译码电路如图题 4.4.14(a) 所示, 对应图题 4.4.14(b) 所示输入波形, 试确定显示器显示的字符序列是什么?



(a)

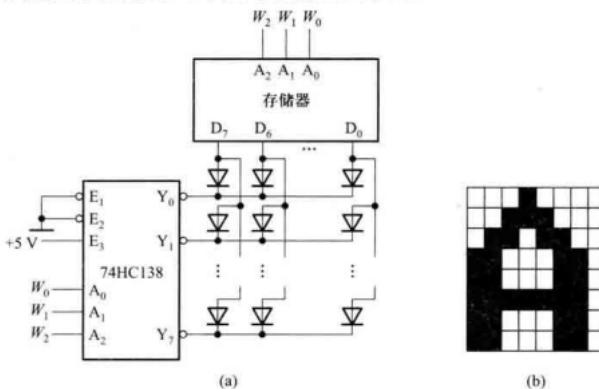


(b)

图题 4.4.14

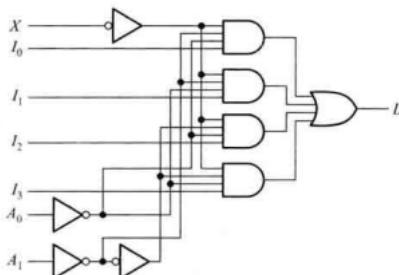
4.4.15 图题 4.4.15 所示为 8×8 个 LED 阵列显示示意图。3 线-8 线译码器控制逐行扫描, 从上到下每次显示一行。存储阵列共有 8×8 个存储单元, 每个单元存放 1 位显示的数据, 需要显示的点存 1, 否则存 0。地址线 $W_2 W_1 W_0$ 从 000 到 111 变化时, 每次将一组 8 个数据送到输出端, 控制发光二极管, 需要发光的二极管接 1, 否

则接 0。如要显示的字型如图题 4.4.15(b) 所示, 试写出存储器存放的数据。若人的视觉暂留时间为 0.05 s, 在满足 LED 阵列图像稳定不闪烁的情况下, 试计算地址变换的最低频率。

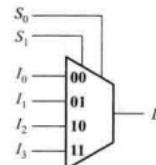


图题 4.4.15

4.4.16 数据选择器如图题 4.4.16 所示, 并行输入数据 $I_3 I_2 I_1 I_0 = 1010$, 控制端 $X=0, A_1 A_0$ 的态序分别为 00, 01, 10, 11, 试画出输出端 L 的波形。



图题 4.4.16



图题 4.4.17

4.4.17 数据选择器如图题 4.4.17 所示, 当 $I_3=0, I_2=I_1=I_0=1$ 时, 有 $L=\bar{S}_1+S_1\bar{S}_0$ 的关系, 证明该逻辑表达式的正确性。

4.4.18 应用图题 4.4.17 所示的电路产生逻辑函数 $F=S_1+S_0$ 。

4.4.19 设计一 4 选 1 数据选择器。数据输入是 I_0, I_1, I_2, I_3 , 数据输出是 Y , 4 个控制信号为 S_3, S_2, S_1, S_0 。要求只当 $S_i=1$ 时, I_i 与 Y 接通, 且由另一控制信号 E 作为该选择器的使能信号。

(1) 画出由反相器、两输入与门或或门实现的逻辑电路。

(2) 选择一合适的三态门作为输出级。

4.4.20 由数据选择器构成的电路如图题 4.4.20 所示, 试写出输出端的逻辑函数式。

4.4.21 由 2 选 1 数据选择器构成的电路如图题 4.4.21 所示, 其数据输入端和选择输入端可接 0、1 或输入变量, 试用该电路实现下列逻辑函数。

$$(1) L(A, B) = AB + C$$

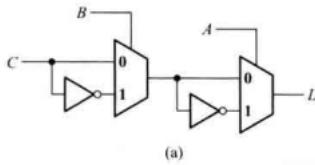
$$(2) L(A, B, C) = AC + BC$$

4.4.22 试用 4 选 1 数据选择器产生逻辑函数:

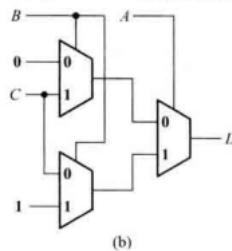
$$(1) L(A, B) = \bar{A}\bar{B} + AB$$

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

4.4.23 用 2 选 1 数据选择器构建一个 4 输入查找表 LUT, 画出电路结构图。

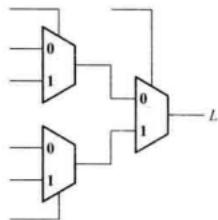


(a)



(b)

图题 4.4.20



图题 4.4.21

4.4.24 试用 3 输入 LUT 实现下列逻辑函数, 画出电路结构, 标明输入端存储单元的数值。

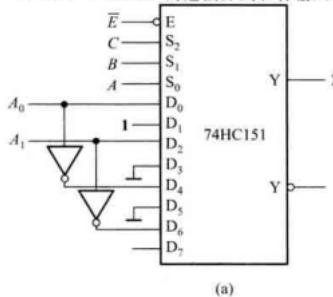
$$(1) L = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C}$$

$$(2) L = A \oplus B \oplus C$$

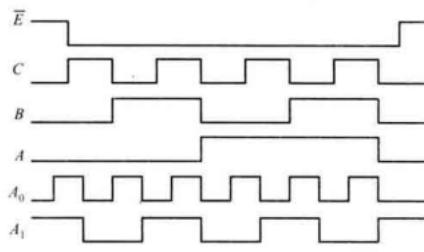
4.4.25 试用 3 输入 LUT 实现下列逻辑函数, 画出 LUT 符号构成的结构图, 说明各 LUT 输入端存储单元的数值。

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

4.4.26 74HC151 的连接方式和各输入端的输入波形如图题 4.4.26 所示, 画出输出端 Y 的波形。



(a)



(b)

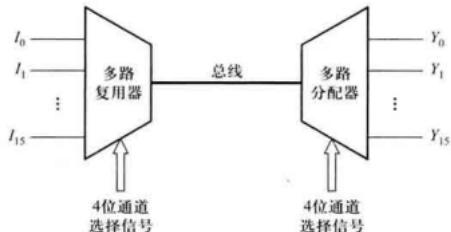
图题 4.4.26

4.4.27 应用 74HC151 实现如下逻辑函数：

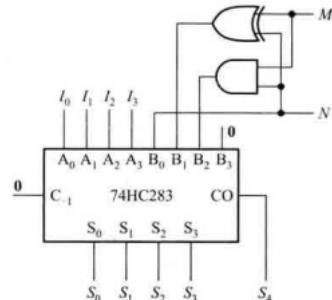
$$(1) L = \overline{ABC} + \overline{AB}C + A\overline{BC}$$

$$(2) L = (A \odot B) \odot C$$

4.4.28 应用数据选择器 74HC151 和 3 线-8 线译码器 74HC138 设计一个数据传输电路，其功能是在 4 位通道选择信号的控制下，能将 16 个输入数据中的任何一个传送到 16 个输出端中相对应的一个输出端，其逻辑电路示意图如图题 4.4.28 所示。



图题 4.4.28



图题 4.4.37

4.4.29 试用三个 3 输入端与门和一个或门实现“ $A > B$ ”的比较电路， A 和 B 均为两位二进制数。

4.4.30 试用五个 2 输入端或门、一个与门和两个非门实现语句“ $A > B$ ”， A 和 B 均为两位二进制数。

4.4.31 试设计一个八位相同数值比较器，当两数相等时，输出 $L=1$ ，否则 $L=0$ 。

4.4.32 试用数值比较器 74HC85 设计一个 8421BCD 码有效性测试电路，当输入为 8421BCD 码时，输出为 1，否则为 0。

4.4.33 试用数值比较器 74HC85 和必要的逻辑门设计一个余 3 码有效性测试电路，当输入为余 3 码时，输出为 1，否则为 0。

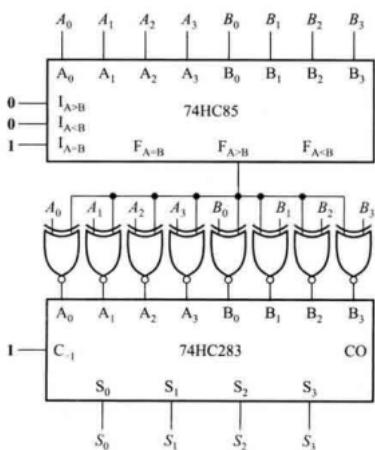
4.4.34 试用反相器、与门和或门设计 1 位二进制全加器。

4.4.35 试用 8 选 1 数据选择器 74HC151，实现 1 位二进制全加器。

4.4.36 仿照半加器和全加器的设计方法，试设计一半减器和一全减器，所用的门电路由自己选定。

4.4.37 由 4 位数加法器 74HC283 构成的逻辑电路如图题 4.4.37 所示， M 和 N 为控制端，试分析该电路的功能。

4.4.38 逻辑电路如图题 4.4.38 所示，试分析该电路的功能。

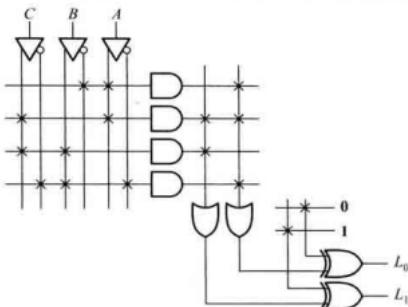


图题 4.4.38

4.4.39 试用若干片 74HC283 构成一个 12 位二进制加法器,画出连接图。此加法器能否用来构成超前进位的级连方式,为什么?

4.5 组合可编程逻辑器件

4.5.1 一个可编程逻辑阵列 PLA 电路如图题 4.5.1 所示。试写出输出逻辑函数表达式。



图题 4.5.1

4.5.2 试用可编程逻辑阵列 PLA 实现下列逻辑函数,并考虑尽量减少乘积项数目。

$$L_0(A, B, C) = \sum(0, 1, 2, 4), L_1(A, B, C) = \sum(0, 5, 6, 7)$$

4.5.3 试用图 4.5.10 所示的可编程阵列逻辑 PAL,实现表题 4.5.3 所示真值表给出的逻辑关系。

表题 4.5.3

A	B	C	L_1	L_2	L_3	L_4
0	0	0	0	1	0	0
0	0	1	1	1	1	1
0	1	0	1	0	1	1
0	1	1	0	1	0	1
1	0	0	1	0	1	0
1	0	1	0	0	0	1
1	1	0	1	1	1	0
1	1	1	0	1	1	1

4.5.4 试用图 4.5.10 所示的可编程阵列逻辑 PAL,实现码转换电路,输入为 4 位 8421BCD 码,输出为余 3 码。

4.6 用 Verilog HDL 描述组合逻辑电路

4.6.1 试根据图 4.4.3 所示的逻辑图,写出 8421BCD 码编码器的 Verilog 门级描述。

4.6.2 根据下面的 Verilog 描述,画出数字电路的逻辑图,写出逻辑电路的输出表达式。

(1) **module** Circuit_A (AgtB, AltB, AeqB, A, B);

```

input [1:0] A,B;
output AgtB,AhB,AeqB;
nor( AgtB,AhB,AeqB );
or ( AhB,w1,w2,w3 );
and ( AeqB,w4,w5 ),
    ( w1,w6,B[1]),
    ( w2,w6,w7,B[0]),
    ( w3,w7,B[1],B[0]);
not ( w6,A[1]),
    ( w7,A[0]);
xnor ( w4,A[1],B[1]);
xnor ( w5,A[0],B[0]);
endmodule

(2) module Circuit_B ( input A,B, output Y1,Y2);
    assign Y1 = A & B;
    or ( Y2,A ,B);
endmodule

```

4.6.3 使用连续赋值语句,写出由下列逻辑函数定义的逻辑电路的 Verilog 描述。

$$(1) L_1 = (B+C)(\bar{A}+D)\bar{B}$$

$$(2) L_2 = (\bar{B}C+ABC+\bar{B}\bar{C})(A+\bar{D})$$

$$(3) L_3 = C(AD+B)+\bar{A}B$$

4.6.4 图 4.2.3 是一个码制变换器,将输入的格雷码转换成二进制码输出,试用 Verilog 数据流方式描述该码制变换器的功能。然后用 Quartus II 软件进行逻辑功能仿真,并给出仿真波形。

4.6.5 根据功能表 4.4.2,写出 4 线-2 线优先编码器的行为级描述。然后用 Quartus II 软件进行逻辑功能仿真,并给出仿真波形。

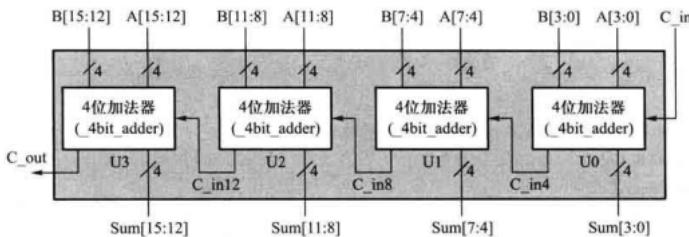
4.6.6 说明下列 Verilog 程序所描述电路的功能,并画出逻辑图。

```

module Circuit_A (
    input [1:0] A,B,
    input S,E,
    output[1:0] Y
);
    assign Y = E ? (S ? A : B) : 'bz ;
endmodule

```

4.6.7 一个 16 位的加法器的组成框图如图题 4.6.7 所示,它由 4 个 4 位的加法器级联而成,每个单元所产生的进位从最低位开始逐级传递至下一级的进位输入端。要求直接引用例 4.6.11 中的 4 位加法器模块_4bit_adder 对 16 位加法器的行为进行描述,然后用 Quartus II 软件对整个电路进行逻辑功能仿真,并给出仿真波形。



图题 4.6.7

4.6.8 根据图 4.4.5 所示的逻辑图, 使用分模块、分层次设计方法, 对 16 线-4 线优先编码器的行为进行描述。要求如下:

- (1) 首先根据功能表 4.4.4 写出 8 线-3 线优先编码器行为级描述, 并用 Quartus II 软件对该模块进行逻辑功能仿真, 并给出仿真波形。
- (2) 然后调用上面设计的编码器子模块和基本门级元件, 完成 16 线-4 线优先编码器的建模。
- (3) 最后用 Quartus II 软件对整个电路进行逻辑功能仿真, 并给出仿真波形。

4.6.9 试根据图 4.4.31 和图 4.4.32 所示的数值比较器逻辑图, 使用分模块、分层次设计方法, 对两位数值比较器的行为进行描述。要求如下:

- (1) 首先根据图 4.4.31 对 1 位数值比较器的行为进行描述, 并用 Quartus II 软件对该模块进行逻辑功能仿真, 并给出仿真波形。
- (2) 然后调用上面设计的 1 位比较器模块和基本门级元件, 完成两位数值比较器的建模。
- (3) 最后用 Quartus II 软件对整个电路进行逻辑功能仿真, 并给出仿真波形。

4.6.10 下面是用分层次方法设计的 4 位串行全加器程序。设计者首先完成了 1 位全加器(模块名为_1bitAdder)的建模和仿真, 结果是正确的; 然后在顶层调用 4 个 1 位全加器模块组合成为 4 位全加器(模块名为_4bitAdder), 结果编译未能通过, 试参照图 4.4.36 所示组成框图分析下列程序中存在的错误, 并进行改正。

```
module _4bitAdder( A,B,Cin,Sum,Cout );
    input [3:0] A,B;
    input Cin;
    output [3:0] Sum;
    output Cout;
    reg Cout;
    reg [4:0] temp;
    always @ ( A or B or Cin)
    begin
        temp[0] = Cin;
        _1bitAdder u0( A[0],B[0],temp[0],Sum[0],temp[1]);
        _1bitAdder u1( A[1],B[1],temp[1],Sum[1],temp[2]);
        _1bitAdder u2( A[2],B[2],temp[2],Sum[2],temp[3]);
        _1bitAdder u3( A[3],B[3],temp[3],Sum[3],temp[4]);
    end
endmodule
```

/230/ 4 组合逻辑电路

```
Cout = temp[ 4 ] ;  
end  
endmodule  
  
module _1bitAdder ( A,B,Ci,Sum,Co ) ; //此模块正确  
    input A,B,Ci;  
    output Sum,Co;  
    assign Sum = A'B'Ci;  
    assign Co = (A & B) |( B & Ci) |( A & Ci) ;  
endmodule
```

5

>>>

锁存器和触发器

引
言

大多数数字系统中,除了需要具有逻辑运算和算术运算功能的组合逻辑电路外,还需要具有存储功能的电路。组合电路与存储电路结合构成时序逻辑电路,简称时序电路。本章将讨论实现存储功能的两种逻辑单元电路:锁存器和触发器,着重讨论它们的工作原理与电路结构,以及所实现的不同逻辑功能。此外,本章还将讨论用Verilog HDL描述锁存器与触发器的方法。

5.1 基本双稳态电路

将两个非门 G_1 和 G_2 接成如图 5.1.1 所示的交叉耦合形式,则构成最基本的双稳态电路。

从图 5.1.1 所示电路的逻辑关系可知,若 $Q=0$, 经非门 G_2 反相, 则 $\bar{Q}=1$ 。 \bar{Q} 反馈到 G_1 输入端, 又保证了 $Q=0$ 。由于两个非门首尾相接的逻辑锁定, 因而电路能自行保持在 $Q=0, \bar{Q}=1$ 的状态, 形成第一种稳定状态。反之, 若 $Q=1, \bar{Q}=0$, 则形成第二种稳定状态。在两种稳定状态中, 输出端 Q 和 \bar{Q} 总是逻辑互补的。可以定义 $Q=0$ 为整个电路的 0 状态, $Q=1$ 则是 1 状态。电路进入其中任意一种逻辑状态都能长期保持下去, 并可以通过 Q 端电平检测出来, 因此, 它具有存储 1 位二进制数据的功能。

像图 5.1.1 所示电路这样, 具有 0、1 两种逻辑状态, 一旦进入其中一种状态, 就能长期保持不变的单元电路, 称为双稳态存储电路, 简称双稳态电路。本章所讨论的锁存器和触发器均属于双稳态电路。

可以看出, 图 5.1.1 所示双稳态电路的功能极不完备。在接通电源后, 它可能随机进入 0 状态或 1 状态, 因为没有控制机构, 所以也无法在运行中改变和控制它的状态, 从而不能作为存储

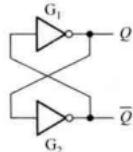


图 5.1.1 最基本的双稳态电路

电路使用。但是,该电路是各种锁存器、触发器等存储单元的基础。

复习思考题

5.1.1 为什么图 5.1.1 所示的电路能长期保持状态不变?

5.2 SR 锁存器

锁存器(Latch)是一种对脉冲电平敏感的双稳态电路,它具有 **0** 和 **1** 两个稳定状态,一旦状态被确定,就能自行保持,直到有外部特定输入脉冲电平作用在电路一定位置时,才有可能改变状态。这种特性可以用于置入和存储 1 位二进制数据。本节首先讨论 **SR** 锁存器。

5.2.1 基本 SR 锁存器

1. 基本 SR 锁存器的工作原理

将图 5.1.1 中双稳态电路的非门换成或非门,则构成图 5.2.1(a)所示的基本 **SR** 锁存器,它是一种具有最简单控制功能的双稳态电路。图中,**S** 和 **R** 是两个输入端,**Q** 和 \bar{Q} 是两个输出端。定义 $Q=0$ 且 $\bar{Q}=1$ 为整个锁存器的 **0** 状态, $Q=1$ 且 $\bar{Q}=0$ 则是锁存器的 **1** 状态。下面根据 **S**、**R** 的 4 种输入状态组合来分析它的工作原理。

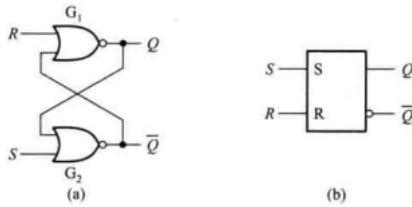


图 5.2.1 用或非门构成基本 SR 锁存器

(a) 逻辑电路 (b) 逻辑符号

① $S=R=0$

根据图 5.2.1(a)的逻辑电路可以列出 $Q=\overline{R+\bar{Q}}=\overline{0+1}=\overline{1}=0$, $\bar{Q}=\overline{S+Q}=\overline{0+0}=\overline{0}=1$ 。显然,**S**、**R** 两信号对输出 **Q** 和 \bar{Q} 不起作用,电路状态保持不变,功能与图 5.1.1 所示的双稳态电路相同,因此可存储 1 位二进制数据。

② $S=0, R=1$

对于或非门而言, $S=0$ 不会影响 G_2 的输出状态,而 $R=1$ 作用于 G_1 则不然,所以必须首先确

定 G_1 输出端 Q 的状态。根据电路可得: $Q = R + \bar{Q} = 1 + \bar{Q} = 0$ 。该信号再反馈到 G_2 输入端, 于是得: $\bar{Q} = S + Q = 0 + 0 = 0$ 。根据定义, 锁存器这时的状态为 **0**。

再从电路的动态变化进行分析。如果图 5.2.1(a) 所示电路此前的状态为 **1**, 即 $Q = 1, \bar{Q} = 0$, 在 R 端出现逻辑 **1** 电平瞬间, 将使 Q 端输出电压下降并作用于 G_2 的输入端, 随即引发 \bar{Q} 端电压上升。一旦 Q 和 \bar{Q} 端均跨越逻辑阈值电平, 便迅速转换为 $Q = 0, \bar{Q} = 1$, 电路状态由 **1** 翻转为 **0**; 反之, 如果此前电路状态为 **0**, 即 $Q = 0, \bar{Q} = 1$, 则 $R = 1$ 的出现不改变其状态。

总之, $S = 0, R = 1$ 将使锁存器置 **0**, 因此将 R 端称为复位(或置 **0**)输入端。当 $R = 1$ 信号消失(即回到 **0**), 电路则进入前述①的状态, 使锁存器的 **0** 状态得以保持。

③ $S = 1, R = 0$

电路是对称的, $S = 1, R = 0$ 将首先使 $\bar{Q} = S + Q = 0$, 继而 $Q = R + \bar{Q} = 1$, 锁存器置 **1**。 S 端称为置位(或置 **1**)输入端。当 $S = 1$ 信号消失, 同样可使锁存器的 **1** 状态得以保持。

④ $S = R = 1$

无论 Q 和 \bar{Q} 原来是什么状态, $S = R = 1$ 将强制 $Q = R + \bar{Q} = 1 + \bar{Q} = 0, \bar{Q} = S + Q = 1 + Q = 0$, 锁存器处在既非 **1**, 又非 **0** 的非定义状态。若 S 和 R 同时回到 **0**, 则无法确定锁存器将落入 **1** 状态还是 **0** 状态。由于电路存在制造误差, G_1, G_2 两门的延迟时间总是有微小差别, 若 G_1 的延迟时间稍短, 在 S 和 R 同时跳变到 **0** 时, Q 端会抢先跳变为 **1**, 迫使 $\bar{Q} = 0$; 反之, 若 G_2 延迟时间稍短, 锁存器则进入 **0** 状态。所以, 实际的电路在这种情况下总是倒向电路设计者无法预知的一个固定状态。为保证锁存器始终工作于定义状态, 输入信号应遵守 $SR = 0$ 的约束条件, 也就是说不允许 $S = R = 1$ 。

由上述分析可得基本 SR 锁存器的功能表, 如表 5.2.1 所示。表中的 4 行内容分别对应于前述①~④的输入和输出状态。

表 5.2.1 用或非门构成基本 SR 锁存器的功能表

S	R	Q	\bar{Q}	功能
0	0	不变	不变	保持
0	1	0	1	置 0
1	0	1	0	置 1
1	1	0	0	非定义状态

图 5.2.1(b) 所示为基本 SR 锁存器的逻辑符号, S 和 R 分别为置位端和复位端, Q 和 \bar{Q} 为互补的两个输出端, 其中 \bar{Q} 输出锁存器的非状态, 所以用小圆圈示之。这样, 不通过图 5.2.1(a) 的逻辑门电路, 仅从抽象的逻辑符号也可以理解基本 SR 锁存器各输入、输出信号之间的逻辑关系。

基本 SR 锁存器的数据保持、置 **0** 和置 **1** 功能, 是一个可实际应用的存储单元最基本的逻辑

功能。基本 SR 锁存器的典型工作波形如图 5.2.2 所示。

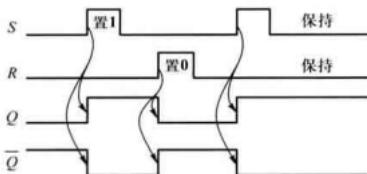


图 5.2.2 基本 SR 锁存器的典型工作波形

例 5.2.1 设图 5.2.1(a) 所示基本 SR 锁存器的初始状态为 1, 其 S、R 端输入波形如图 5.2.3 中虚线上面所示, 试画出对应的 Q 和 \bar{Q} 波形。

解: 根据表 5.2.1 可以画出 Q 和 \bar{Q} 端的波形如图 5.2.3 中虚线下面所示。

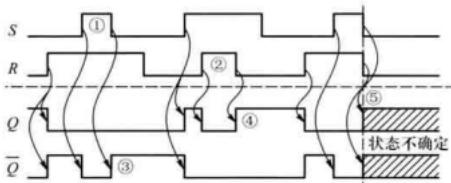


图 5.2.3 例 5.2.1 的波形

需要注意, 虽然图 5.2.3 中①、②两处输入信号越出了 SR 锁存器的约束条件, 出现 $S=R=1$ 使 $Q=\bar{Q}=1$ 的情况, 但是, 如果 S 和 R 的 1 电平不同时撤消, 此后的输出状态仍然是可以确定的, 如图中③、④所示。而在⑤处, 由于 S 和 R 的高电平同时撤消, 所以锁存器以后的状态将无法确定, 从而失去对它的控制, 在实际应用中必须避免出现这种情况。

2. 基本 SR 锁存器的动态特性

此前的讨论仅考虑了电路的逻辑关系, 没有涉及门电路输出信号对输入信号的时间延迟, 即电路的动态特性。而构成图 5.2.1(a) 所示电路的两个或非门在工作时都存在一定的传输延迟, 当输入信号 S 或 R 变为高电平后, 输出信号 Q 或 \bar{Q} 需要经过一定延迟才会产生变化。这种延迟有时会影响到被其驱动的后续电路的动作, 可能造成错误的逻辑输出或出现工作不稳定的情况。此外, 为保证锁存器状态可靠转换, 对输入信号也需要有一定的时间要求。

定时图是表达时序电路动态特性的工具之一, 它表达了电路动作过程中, 输出对输入信号响应的延迟时间, 以及对各输入信号的时间要求。图 5.2.4 是基本 SR 锁存器的定时图。图中, 脉冲信号的上升沿和下降沿均用斜线表达, 表示存在一定的上升时间和下降时间, 脉冲沿的基准时间定位在上升沿和下降沿的 50% 处。

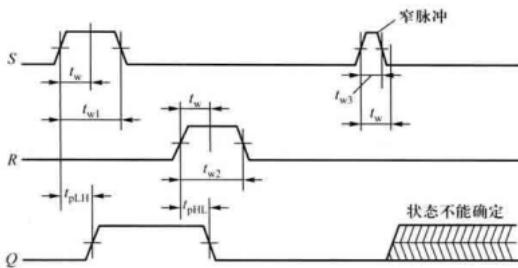


图 5.2.4 基本 SR 锁存器的定时图

传输延迟时间 t_{plH} 和 t_{phL}

如图 5.2.4 所示, 当置 1 信号 S 上升为高电平时, 需要一定的传输延迟时间 t_{plH} 之后, Q 端才转换为高电平。同样, 置 0 信号 R 作用于电路, Q 端电平也经一定的传输延迟时间 t_{phL} 才变化为 0。 \bar{Q} 端的变化相对于输入信号 S 或 R 的变化也存在一定的传输延迟。这里, 把 t_{plH} 和 t_{phL} 定义为基本 SR 锁存器的传输延迟时间。对于具体电路, 由于信号通过的路径不同, t_{plH} 和 t_{phL} 一般不完全相等。

脉冲宽度 t_w

基本 SR 锁存器工作时, 必须保证 S 和 R 的高电平脉冲宽度不小于某一最小值 t_w 。例如, 图 5.2.4 中的 t_{w1} 和 t_{w2} 均满足要求, 从而电路能可靠地实现翻转。如果加在 S 或 R 端的脉冲宽度过窄, 如图 5.2.4 所示宽度为 t_{w3} 的窄脉冲, 在 Q 端电压尚未越过逻辑阈值电平时, S 端的高电平就被撤除, 电路可能又回到原来的状态, 或者使 Q 的最终状态不能确定。所以基本 SR 锁存器应用中要求输入信号 S 和 R 的脉冲宽度必须不小于一个最低限值 t_w , 才能保证在 S 或 R 脉冲作用之后有确定的状态。

3. 用与非门构成的基本 SR 锁存器

基本 SR 锁存器也可以用与非门构成, 其逻辑原理图和逻辑符号如图 5.2.5 所示。图 5.2.5 (a) 中的两个与非门是用其等效符号表示的, 由图可分析出当 \bar{S}, \bar{R} 为不同输入状态组合时锁存器的状态, 如表 5.2.2 所示。

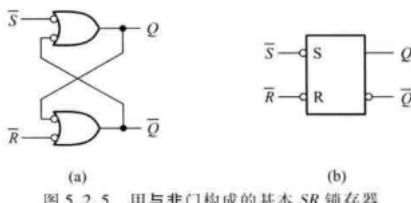


图 5.2.5 用与非门构成的基本 SR 锁存器

(a) 逻辑电路 (b) 逻辑符号

表 5.2.2 用与非门构成基本 SR 锁存器的功能表

\bar{S}	\bar{R}	Q	\bar{Q}	功能
1	1	不变	不变	保持
1	0	0	1	置 0
0	1	1	0	置 1
0	0	1	1	非定义状态

当输入为 $\bar{S}=\bar{R}=0$ 时, $Q=1$, $\bar{Q}=1$, 该锁存器处于非定义状态, 因此工作时应当受到 $\bar{S}+\bar{R}=1$ 的条件约束, 即同样应遵守 $SR=0$ 的约束条件。

与前述或非门构成的基本 SR 锁存器不同, 这种锁存器的输入信号 \bar{S} 和 \bar{R} 以逻辑 0 作为有效作用信号, 因而在图 5.2.5(b) 所示逻辑符号中, 在输入端用小圆圈表示。为了区别, 这种锁存器有时也称为基本 $\bar{S}\bar{R}$ 锁存器。

4. 基本 SR 锁存器的应用

基本 SR 锁存器可以应用于数字系统中某些特定标志的设置。例如, 当某种预设逻辑条件具备时, 电路可以通过 S 端将基本 SR 锁存器置 1, 标志预设事件已经发生; 而当另一种相悖的预设逻辑条件满足时, 则可通过 R 端将其置 0。下面的例题是另外一种应用。

例 5.2.2 运用基本 SR 锁存器消除机械开关触点抖动引起的脉冲输出。

解: 机械开关(例如按键、拨动开关、继电器等)常常用作数字系统的逻辑电平输入装置。由于机械开关接通或断开瞬间的弹性振颤, 触点会在短时间内多次接通和断开, 出现如图 5.2.6 所示的“抖动”现象, 使 v_o 的逻辑电平多次在 0 和 1 之间跳变, 导致错误逻辑输入数字系统。机械开关触点振颤的延续时间因开关结构、几何形状和尺寸以及材料的不同而不同, 从数毫秒到上百毫秒不等。在设计数字系统时, 通常需要采用硬件方法(如本例)或软件方法来克服其不良影响。

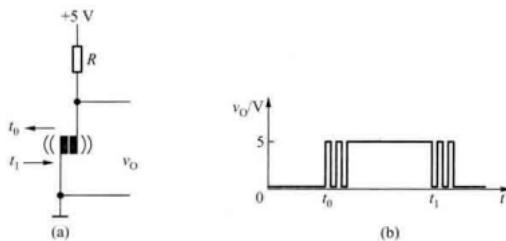


图 5.2.6 机械开关的“抖动”现象

(a) 开关在 t_0 时断开, t_1 时接通 (b) 实际输出波形

图 5.2.7(a)是解决机械开关抖动现象的一种硬件方案,它利用基本 SR 锁存器的存储作用消除开关触点振动所产生的影响,称为去抖动电路。图 5.2.7(b)中横虚线上面是图 5.2.7(a)所示电路中 \bar{S} 和 \bar{R} 端的波形,表示了单刀双掷开关 S 由 B 拨向 A,然后又拨回 B 的过程。初始时,开关 S 的动触点与 B 点接通,锁存器的状态为 0。在开关 S 拨向 A,动触点脱离 B 点瞬间产生的抖动,并不影响锁存器的状态。在动触点悬空瞬间, $\bar{S} = \bar{R} = 1$, Q 仍维持为 0。当它第一次触碰 A 点时,便使 $\bar{S} = 0$, Q 端状态立即翻转为 1。此后,即使触点抖动,使 \bar{S} 端再次出现高、低电平的跳变,也不会改变 $Q = 1$ 的状态。由于电路是对称的,开关由 A 拨向 B 与前述的情况类似。于是得到 Q 端波形,如图 5.2.7(b)中横虚线下面所示。可以看到,在开关每次变化时,锁存器只翻转一次,不存在抖动波形。图 5.2.7(a)所示去抖动电路特别适用于需要对机械开关状态进行计数的场合,它可以消除开关触点抖动造成的误计数。

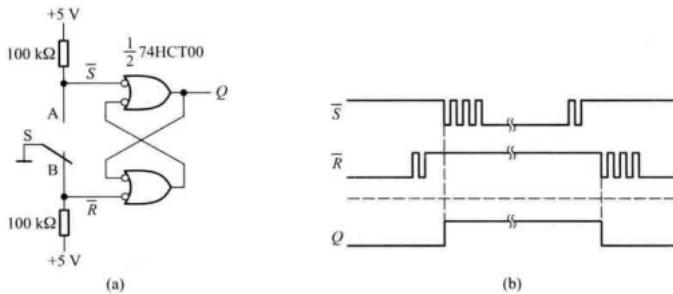


图 5.2.7 用基本 SR 锁存器构成机械开关去抖动电路

(a) 电路 (b) 波形图

5.2.2 门控 SR 锁存器

1. 门控 SR 锁存器的逻辑功能

前面所讨论的基本 SR 锁存器的输出状态是由输入信号 S 或 R 直接控制的,而图 5.2.8(a)所示电路在基本 SR 锁存器输入端增加了一对逻辑门 G_3 、 G_4 ,用使能信号 E 控制锁存器在某一指定时刻,根据 S 、 R 输入信号确定输出状态。这种锁存器称为门控 SR 锁存器。通过控制 E 端电平,可以实现多个锁存器同步的数据锁存。

从图 5.2.8(a)可以看出,当 $E = 0$ 时, $Q_3 = Q_4 = 0$, S 、 R 端的逻辑状态不会影响到锁存器的状态;当 $E = 1$ 时, S 、 R 端的信号被传送到基本 SR 锁存器的输入端,从而可确定 Q 和 \bar{Q} 端的状态,其功能与表 5.2.1 一致。若 $E = 1$ 时输入信号 $S = R = 1$,则 $Q = \bar{Q} = 0$,锁存器将处于非定义的逻辑状态。当 E 恢复为 0 时,由于 Q_3 、 Q_4 同时回到 0,将不能确定锁存器的状态。因此,应用这种锁存器必须更严格地遵守 $SR = 0$ 的约束条件。由于约束条件造成的应用限制,因而很少有独立的门

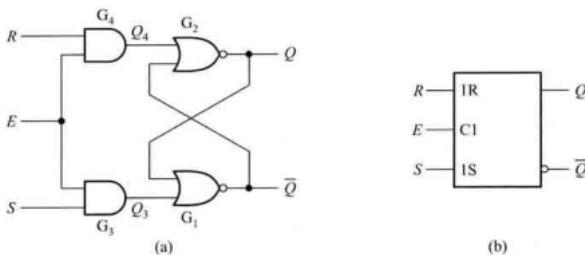


图 5.2.8 门控 SR 锁存器

(a) 逻辑电路 (b) 逻辑符号

控 SR 锁存器产品。但是,在许多中、大规模集成电路中时常应用这种锁存器,或用它构成触发器或存储器。所以,SR 锁存器仍是重要的基本逻辑单元。

图 5.2.8(b)所示是门控 SR 锁存器的逻辑符号。其方框内用 C1 和 1R、1S 表达内部逻辑之间的关联关系。C 表示这种关联属于控制类型,其后缀用标识序号“1”表示该输入的逻辑状态对所有以“1”作为前缀的输入起控制作用。这里置位和复位输入均受 C1 的控制,故 S 和 R 之前分别以标识序号“1”作为前缀。图 5.2.8(b)中所示的两个输出端 Q 和 \bar{Q} ,其意义与图 5.2.1 (b)所示基本 SR 锁存器相同。

例 5.2.3 图 5.2.8(a)所示门控 SR 锁存器的 E 、 S 、 R 的波形如图 5.2.9 中横虚线上面所示,设锁存器的初始状态为 $Q=0$, $\bar{Q}=1$,试画出 Q_3 、 Q_4 、 Q 和 \bar{Q} 的波形。

解: 从图 5.2.8(a)的逻辑电路图得 $Q_3 = SE$, $Q_4 = RE$ 。于是,可根据 E 、 S 和 R 的波形画出 Q_3 和 Q_4 的波形。图 5.2.8(a)中 G_1 、 G_2 构成基本 SR 锁存器,再根据表 5.2.1 即可画出 Q 和 \bar{Q} 的波形。全部波形如图 5.2.9 所示。

2. CMOS 集成电路中的门控 SR 锁存器

在集成电路中,往往根据具体条件,尽量应用简化电路来实现所要求的逻辑功能。例如,图 5.2.10 所示是一种 CMOS 集成电路中常用的门控 SR 锁存器晶体管级电路,它仅用 6 个 NMOS 管和两个 PMOS 管便实现了图 5.2.8(a)所示的两个与门和两个或非门的逻辑功能,而没有使用标准 CMOS 门电路,从而省却了一些 PMOS 晶体管。由于一般 CMOS 与或非门中的 PMOS 管占据芯片的面积远大于相应的 NMOS 管^①,所以

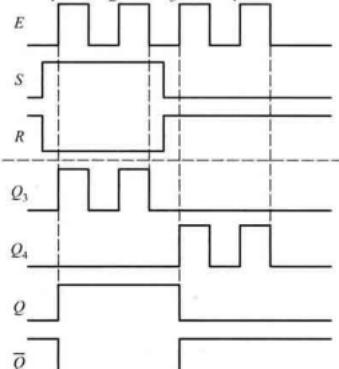


图 5.2.9 例 5.2.3 的波形图

^① Sedra, A. S. and Smith, K. C.; Microelectronic Circuits, 6th Ed., New York: Oxford University Press, Inc., 2010. p. 1117 ~ 1121

图 5.2.10 所示电路的简化有效缩小了锁存器在集成电路芯片中所占的空间。在正常逻辑状态下,该电路只在状态转换瞬间存在一定的工作电流,静态功耗极微。但需要注意,如果在 $E=1$ 的同时 $S=R=1$,则 $T_1 \sim T_3$ 和 $T_8 \sim T_7$ 均处于导通状态,将使电路功耗剧增。因此,在集成电路结构设计时就必须考虑到严格遵守 $SR=0$ 的约束条件,保证在任何时候都不出现 $S=R=1$ 的情况。

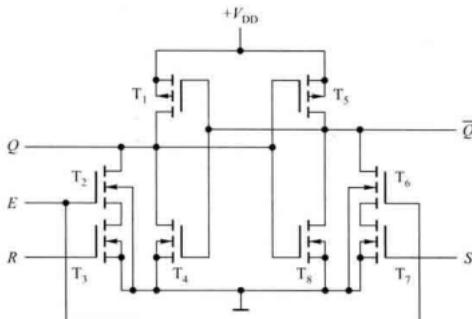


图 5.2.10 CMOS 门控 SR 锁存器电路

由于图 5.2.10 所示电路不是用标准逻辑门电路构成的,所以在考虑它的动态特性时,按图 5.2.8(a)所示的与门和或非门去逐级累加计算传输延迟时间是脱离实际的。应用中,一般通过查阅芯片的数据手册直接了解相关参数。

复习思考题

5.2.1 图 5.2.1(a)所示基本 SR 锁存器和图 5.2.8(a)所示门控 SR 锁存器在电路结构、数据锁存动作上有什么区别?为什么它们在工作中均须遵循 $SR=0$ 的约束条件?

5.2.2 为什么图 5.2.10 中的 CMOS 门控 SR 锁存器电路可以节省一些晶体管?这样的电路有什么优点和应用上的注意事项?

5.3 D 锁存器

5.3.1 D 锁存器的电路结构

与 SR 锁存器不同,D 锁存器在工作中不存在非定义状态,因而得到广泛应用。目前,CMOS 集成电路主要采用传输门控 D 锁存器和逻辑门控 D 锁存器两种电路结构形式,特别是前者电路结构简单、在芯片中占用面积小而更受青睐。

1. 传输门控 D 锁存器

在图 5.1.1 的双稳态电路中插入两个传输门 TG_1 和 TG_2 , 则可构成如图 5.3.1(a) 所示的传输门控 D 锁存器, 图 5.3.1(b) 所示是它的逻辑符号。

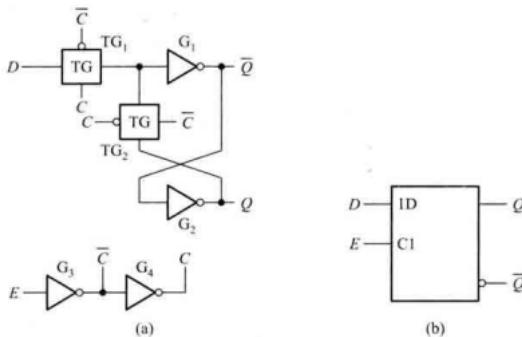


图 5.3.1 传输门控 D 锁存器

(a) 逻辑电路 (b) 逻辑符号

D 锁存器有两个输入端: 使能端 E 和数据输入端 D 。当 $E=1$ 时, $\bar{C}=0, C=1, TG_1$ 导通, TG_2 断开, 如图 5.3.2(a) 所示。输入数据 D 经 G_1, G_2 两个非门, 使 $Q=D, \bar{Q}=\bar{D}$ 。显然, 这时 Q 端跟随输入信号 D 的变化。当 $E=0$ 时, $\bar{C}=1, C=0, TG_1$ 断开, TG_2 导通, 如图 5.3.2(b) 所示, 其原理与图 5.1.1 所示双稳态电路相同。由于 G_1, G_2 输入端存在的分布电容对逻辑电平有暂短的保持作用, 在两个传输门状态转换瞬间并不影响电路的输出状态。之后, 电路将被锁定在 E 信号由 1 变 0 前瞬间 D 信号所确定的状态, 在 $E=0$ 的条件下可保持锁存器状态不变, 使 1 位二进制数据得以存储。表 5.3.1 概括了 D 锁存器的功能。由于这种锁存器在 $E=1$ 时 Q 端可跟随 D 端的逻辑状态变化, 故又称为透明锁存器。

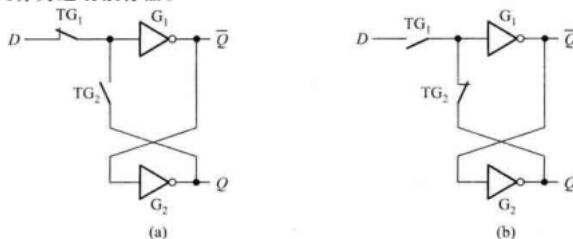


图 5.3.2 传输门的开关状态

(a) $E=1$ 时的状态 (b) $E=0$ 时的状态

表 5.3.1 D 锁存器的功能表

E	D	Q	\bar{Q}	功能
0	x	不变	不变	保持
1	0	0	1	置 0
1	1	1	0	置 1

2. 逻辑门控 D 锁存器

图 5.3.3 所示为逻辑门控 D 锁存器的逻辑电路, 它在门控 SR 锁存器的 S 和 R 输入端之间连接了一个非门 G_5 , 从而保证了 $SR=0$ 的约束条件, 消除了可能出现的非定义状态。读者可仿照前述方法自行分析, 并用表 5.3.1 来验证图 5.3.3 所示电路的逻辑功能。由于它的逻辑功能与传输门控 D 锁存器完全相同, 所以逻辑符号亦相同。

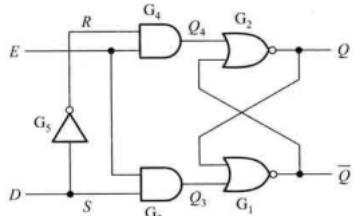


图 5.3.3 逻辑门控 D 锁存器的逻辑电路

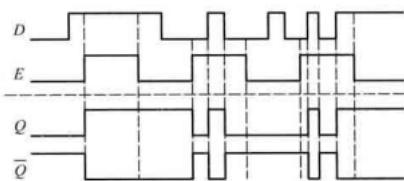


图 5.3.4 例 5.3.1 的波形图

例 5.3.1 设图 5.3.1(a)所示电路的初始状态为 $Q=0$, 输入信号 D,E 的波形如图 5.3.4 中横虚线上面所示, 画出 Q 和 \bar{Q} 的输出波形。

解: 根据图 5.3.2(a)、(b), 当 $E=1$ 时, Q 端波形跟随 D 端变化; 当 E 跳变为 0 时, 锁存器保持此前瞬间的状态, 可以画出 Q 和 \bar{Q} 波形如图 5.3.4 中横虚线下面所示。

5.3.2 典型的 D 锁存器集成电路

1. 74HC/HCT373

图 5.3.5 为中规模集成的 CMOS 八 D 锁存器 74HC/HCT373 的内部逻辑图, 其核心电路是 8 个如图 5.3.1(a)所示的传输门控 D 锁存器。8 个锁存器共用同一对互补的传输门控制信号 C 和 \bar{C} , 这对信号又由锁存使能信号 LE 所驱动。当 LE 为高电平时, 允许各 D 锁存器的输出跟随相应输入信号的变化; LE 为低电平时则保持状态不变。8 个 D 锁存器输出端都带有三态门, 当输出三态门使能信号 \overline{OE} 为低电平时, 三态门有效, 输出锁存的信号; 当 \overline{OE} 为高电平时, 输出处于高阻状态。这种三态输出电路, 一方面提高了对负载的驱动能力, 在锁存器与输出负载之间起到

隔离作用,避免因负载变化而影响锁存器的动态特性;更重要的是使74HC/HCT373可以方便地应用于微处理器或计算机的总线传输电路。

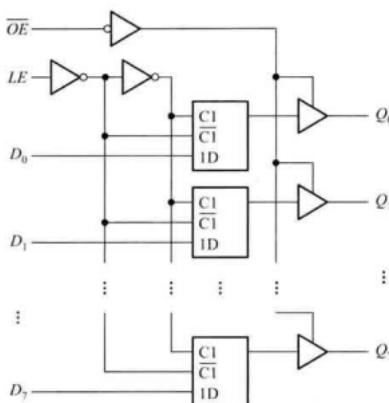


图 5.3.5 74HC/HCT373 八 D 锁存器的内部逻辑图

根据 LE 和 \overline{OE} 的不同逻辑电平,74 HC/HCT 373 可分为三种工作模式:① 使能和读锁存器(传送模式);② 锁存和读锁存器;③ 禁止输出。表 5.3.2 所示为其功能表。

表 5.3.2 74HC/HCT373 的功能表

工作模式	输入			内部锁存器状态	输出 Q_N
	\overline{OE}	LE	D_N		
使能和读锁存器 (传送模式)	L	H	L	L	L
	L	H	H	H	H
锁存和读锁存器	L	L	L^*	L	L
	L	L	H^*	H	H
禁止输出	H	×	×	×	高阻

注: D_N 和 Q_N 的下标表示第 N 位锁存器数据。 L^* 和 H^* 表示使能信号 LE 的电平由高变低之前瞬间 D_N 的电平。

2. 74LVC32373A

图 5.3.6 所示为“宽总线”锁存器 74LVC32373A 的内部简化逻辑图,芯片中集成了 32 个三态输出的 D 锁存器,它们分为 4 组,每一组的逻辑功能都相当于一个 74HC373,各组间既可同步工作,又可独立运行,异步工作。这种结构给芯片的总线应用带来很大的灵活性。除此之外,一般还能找到 16 位、20 位等“宽总线”D 锁存器集成电路产品,以适用于不同总线宽度的微处理器机

系统。

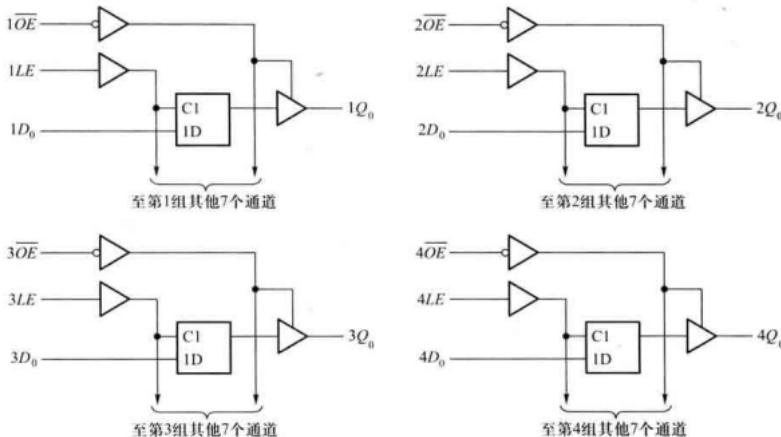


图 5.3.6 “宽总线”D 锁存器 74LVC32373A 的内部简化逻辑图

3. 74LVC1G373

图 5.3.7 所示为“小逻辑”锁存器 74LVC1G373 的内部逻辑图, 它的封装中只有一路 D 锁存器, 其逻辑功能只相当于 74HC373 中的一个通道。

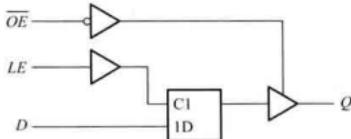


图 5.3.7 “小逻辑”D 锁存器 74LVC1G373 的内部逻辑图

5.3.3 D 锁存器的动态特性

图 5.3.8 所示是 D 锁存器的定时图, 对于传输门控和逻辑门控两种电路结构的 D 锁存器都是适用的, 只是具体参数值有所差异。下面对各参数进行说明。

传输延迟时间 t_{pd}

t_{pd} 是输出信号对输入信号的响应延迟时间, 对于 D 锁存器则是指 D 信号和 E 信号共同作用后, Q

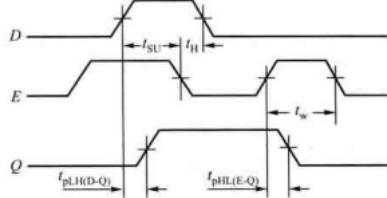


图 5.3.8 D 锁存器的定时图

(或 \bar{Q}) 端响应的延迟时间。图 5.3.8 中所示 $t_{pLH(D-Q)}$ 是输出 Q 从低电平到高电平对 D 信号的延迟时间, $t_{pHL(E-Q)}$ 则是 Q 从高电平到低电平对 E 信号的延迟时间。根据不同的输入状态, 还存在图中没有显示的 $t_{pLH(D-Q)}$ 和 $t_{pHL(E-Q)}$ 。对于 CMOS 集成电路, 因为输出信号对各输入信号的延迟相差不多, 有时统一以 t_{pLH} 和 t_{pHL} 表达, 更经常的是取平均传输延迟时间: $t_{pd} = (t_{pLH} + t_{pHL})/2$ 。

建立时间 t_{su}

信号 D 的逻辑电平必须在使能信号 E 下降沿到来之前建立起来, 才能保证正确地锁存。 t_{su} 表示 D 信号对 E 下降沿的最少时间提前量。

保持时间 t_h

在 E 电平下降后, D 信号不允许立即撤除, 否则不能确保数据的锁存。 t_h 表示 D 信号电平在 E 电平下降后需要继续保持的最少时间。

脉冲宽度 t_w

为保证 D 信号正确传送到 Q 和 \bar{Q} , 要求 E 信号的脉冲宽度不小于 t_w 。

上述 t_{su} 、 t_h 和 t_w 是对输入信号的时间要求。如果电路运行中达不到要求, 则会分别出现如图 5.3.9 所示的情况, 可能导致 D 锁存器不确定的逻辑输出。

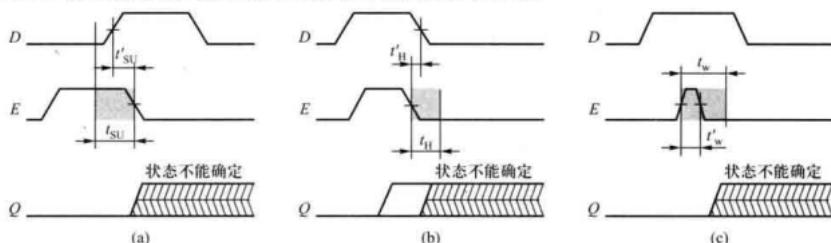


图 5.3.9 t_{su} 、 t_h 和 t_w 达不到要求时可能导致 D 锁存器不确定的逻辑输出

(a) $t'_{su} < t_{su}$ (b) $t'_{hl} < t_h$ (c) $t'_w < t_w$

在生产厂家提供的数据手册中, 都会给出这些时间关系的明确限度, 表 5.3.3 示出环境温度在 $-40^{\circ}\text{C} \sim +85^{\circ}\text{C}$ 范围的几种典型 D 锁存器的定时参数。

表 5.3.3 几种典型的 D 锁存器定时参数

型号	功能	工作电压 V_{cc}/V	t_{pd}/ns^*				t_{su}/ns	t_h/ns	t_w/ns
			$D-Q$		$LE-Q$				
			最小	最大	最小	最大	最小	最小	最小
74HCT373	八 D 锁存器	4.5		38		44	13	12	20
74LVC32373A	三十二 D 锁存器	3.3 ± 0.3	1.6	4.2	2.1	4.6	1.7	1.2	3.3
74LVC1G373	单 D 锁存器	3.3 ± 0.3	1	5.4	1	5.5	1.5	1.5	3

* 测试条件: Q 端负载电容 $C_L = 50 \text{ pF}$ 。

(摘自 <http://www.ti.com/>)

在设计工作中,对电路的动态特性必须予以充分重视,特别是电路工作在接近定时极限的高频条件下,对上述时间关系更要注意留有充分余地。否则,电路在长期工作时会发生原因难以查明的偶发性逻辑错误,或因环境条件改变(如温度变化)而出现工作不稳定的情况。

复习思考题

- 5.3.1 传输门控和逻辑门控 D 锁存器在电路结构和工作原理上有何不同?
- 5.3.2 D 锁存器工作时对使能信号 E 和输入信号 D 在定时上有什么要求? 输出信号 Q 的变化对输入信号 D 、使能信号 E 有何定时特性?

5.4 触发器的电路结构和工作原理

本节讨论另一种对脉冲边沿敏感的双稳态电路。如前所述, D 锁存器在使能信号 E 为逻辑 1 期间更新状态,在图 5.4.1(a)所示的波形图中以加粗部分表示这个敏感时段。在这期间,它的输出会随输入信号变化。而很多时序电路要求存储电路只对时钟信号的上升沿或下降沿敏感,而在其他时刻保持状态不变,例如 6.5 节中将要讨论的移位寄存器和计数器。这种对时钟脉冲边沿敏感的状态更新称为触发,具有触发工作特性的存储单元称为触发器。电路结构不同的触发器对时钟脉冲的敏感边沿可能不同,分为上升沿触发和下降沿触发。本书以 CP (Clock Pulse) 命名上升沿触发的时钟信号,触发边沿如图 5.4.1(b) 波形中的箭头所示;以 \overline{CP} 命名下降沿触发的时钟信号,触发边沿如图 5.4.1(c) 中箭头所示。

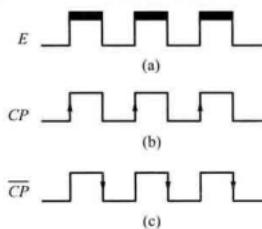


图 5.4.1 锁存器对使能信号的响应和触发器对时钟信号的响应

(a) 对脉冲高电平敏感 (b) 对脉冲上升沿敏感 (c) 对脉冲下降沿敏感

在 Verilog HDL 中,对脉冲电平敏感的锁存器和脉冲边沿敏感的触发器的描述语句是不同的,这一点将在 5.6 节中说明。正因为如此,这里要特别强调锁存器与触发器在概念上的差异。

目前应用的触发器主要有三种电路结构:主从触发器、维持阻塞触发器和利用传输延迟的触发器。由于 CMOS 主从结构的 D 触发器在芯片上占用的面积最小,逻辑设计方法也较简单,在

大规模 CMOS 集成电路,特别是可编程逻辑器件(如 CPLD、FPGA)和专用集成电路(ASIC)中得到普遍应用,因而在目前的工程实践中也会更多地面对这种 D 触发器。下面重点讨论 CMOS 主从 D 触发器。

5.4.1 主从 D 触发器的电路结构和工作原理

将两个如图 5.3.1(a)所示 D 锁存器级联,则构成典型的 CMOS 主从 D 触发器,如图 5.4.2 所示。图中左边的锁存器称为主锁存器,右边的称为从锁存器。主锁存器与从锁存器的使能信号相位相反,利用两个锁存器的交互锁存,则可实现存储数据和输入信号之间的隔离。

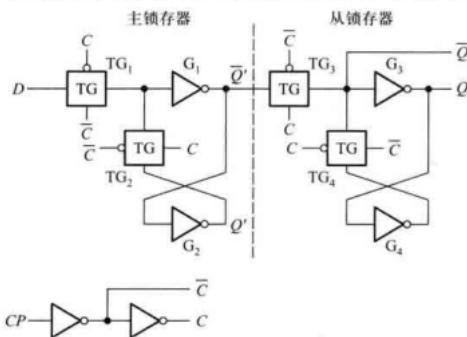


图 5.4.2 CMOS 主从 D 触发器的逻辑电路

主从 D 触发器工作过程分为以下两个节拍。

① 当时钟信号 $CP=0$ 时, $\bar{C}=1, C=0$, 使 TG_1 导通, TG_2 断开, D 端输入信号进入主锁存器, 这时 Q' 跟随 D 端的状态变化, 使 $Q'=D$ 。例如, D 为 1 时, 经 TG_1 传到 G_1 的输入端, 使 $\bar{Q}'=0, Q'=1$ 。同时由于 TG_3 断开, 切断了从锁存器与主锁存器之间的联系, 而 TG_4 导通, 使 G_3 的输入端和 G_4 的输出端经 TG_4 连通, 构成最基本的双稳态电路, 使从锁存器维持原来的状态, 即触发器的输出状态不变。

② 当 CP 由 0 跳变到 1 后, $\bar{C}=0, C=1$, 使 TG_1 断开, 从而切断了 D 端与主锁存器的联系, 同时 TG_2 导通, 将 G_1 的输入端和 G_2 的输出端连通, 主锁存器锁存 CP 跳变前 D 端的数据。这时, TG_3 导通, TG_4 断开, Q' 端信号传送到 Q 端。若 $Q'=1, \bar{Q}'=0$, 经 TG_3 传送给 G_3 的输入端, 于是 $\bar{Q}=0, Q=1$ 。

可见, 从锁存器在工作中是跟随主锁存器的状态变化的, 触发器因之冠名主从(Master-Slave)。它的状态转换发生在 CP 信号上升沿到来后的瞬间, 输出状态由 CP 信号上升沿到达前瞬间的数据信号 D 所决定, 从功能上考虑称为 D 触发器。如果以 Q^{**} 表示 CP 信号上升沿到达

后触发器的状态，则 D 触发器的特性可以用下式来表达

$$Q^{n+1} = D \quad (5.4.1)$$

该式称为 D 触发器的特性方程。它反映了触发器在时钟信号作用后的状态与此前输入信号 D 的关系。

5.4.2 典型的主从 D 触发器集成电路

1. 74HC/HCT74

图 5.4.3 所示是一种 CMOS 集成 D 触发器的内部逻辑电路。由于实际应用中有时需要对触发器进行异步（即与图中 CP 信号无关）置位、复位，所以电路中引入了直接置 0 端 \bar{R}_D 和直接置 1 端 \bar{S}_D 。这两个信号经非门缓冲后，分别送入主锁存器和从锁存器。当 $CP=1$ 时， TG_1 、 TG_4 断开而 TG_2 、 TG_3 导通，或非门 G_1 和 G_2 构成基本 SR 锁存器，可以把 \bar{R}_D 或 \bar{S}_D 信号锁存在主锁存器，进而传送到 Q 和 \bar{Q} 端。当 $CP=0$ 时， TG_3 断开 TG_4 导通，或非门 G_3 和 G_4 构成基本 SR 锁存器，同样可把 \bar{R}_D 或 \bar{S}_D 信号锁存到 Q 和 \bar{Q} 端。显然， \bar{S}_D 和 \bar{R}_D 对触发器的状态有优先控制权。只有当 $\bar{S}_D = \bar{R}_D = 1$ 时，触发器才能被 CP 上升沿触发，按 D 端逻辑值更新状态。

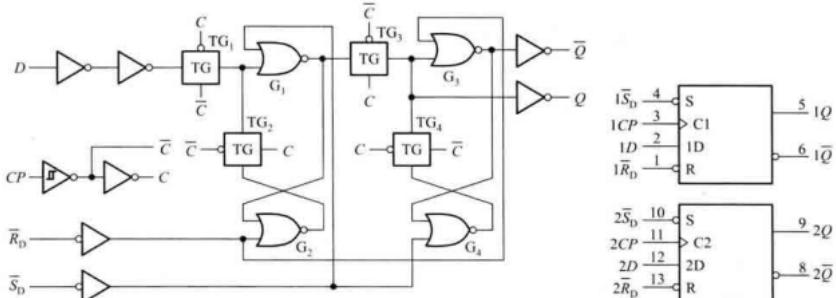


图 5.4.3 74HC/HCT74 中 D 触发器的逻辑图

图 5.4.4 74HC/HCT74 的逻辑符号

由图可见，触发器的所有输入、输出都设置了缓冲电路，这些措施提高了电路工作的稳定性。为了避免 CP 脉冲上升沿或下降沿在跨越阈值电平时的噪声引发触发器的误触发，电路在 CP 输入端特别增置了施密特反相器，以提高抗干扰能力（施密特电路的抗干扰原理请参见第 8 章）。

双 D 触发器 74HC/HCT74 芯片中集成了两个如图 5.4.3 所示的 D 触发器，它们之间相互独立，其逻辑符号见图 5.4.4。图中方框内侧的“>”符号表示电路对 CP 信号的脉冲边沿敏感，C1 和 C2 分别关联控制着 1D 和 2D。表 5.4.1 是 74HC/HCT74 中触发器的功能表。表的上半部分是 \bar{S}_D 、 \bar{R}_D 与输出信号的关系。其中，当 \bar{S}_D 和 \bar{R}_D 均为低电平时，输出 Q 和 \bar{Q} 均为高电平，若 \bar{S}_D 、 \bar{R}_D

同时恢复高电平，则不能确定触发器此后的状态，因而 $R_b S_b = 0$ （即 $\bar{S}_b + \bar{R}_b = 1$ ）仍为约束条件。表 5.4.1 的下半部分是在 \bar{S}_b, \bar{R}_b 均为高电平时的动态功能表。符号“↑”表示 CP 脉冲上升沿触发， Q^{n+1} 和 \bar{Q}^{n+1} 分别为 CP 脉冲上升沿到达后 Q 和 \bar{Q} 端的状态。

表 5.4.1 74HC/HCT74 的功能表

输入				输出	
\bar{S}_b	\bar{R}_b	CP	D	Q	\bar{Q}
L	H	×	×	H	L
H	L	×	×	L	H
L	L	×	×	H	H
\bar{S}_b	\bar{R}_b	CP	D	Q^{n+1}	\bar{Q}^{n+1}
H	H	↑	L*	L	H
H	H	↑	H*	H	L

L* 和 H* 表示 CP 脉冲上升沿到来之前瞬间的电平。

2. 74LVC1G79

小逻辑系列的 D 触发器 74LVC1G79 的内部逻辑图如图 5.4.5 所示，其逻辑已达到最简。

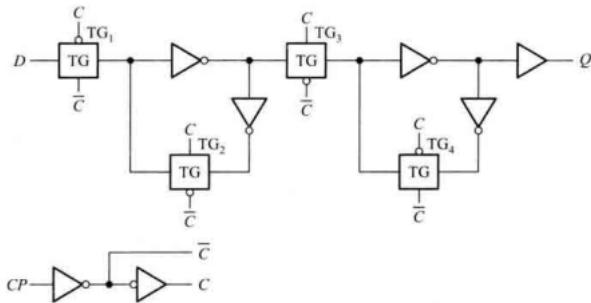


图 5.4.5 74LVC1G79 的内部逻辑图

与锁存器类似， D 触发器同样有宽总线产品，如 74LVC32374，不再赘述。

5.4.3 主从 D 触发器的动态特性

图 5.4.6 的定时图显示了 CP 上升沿触发的 D 触发器的动态特性，它反映了输出对时钟信号响应的延迟时间，以及对输入逻辑信号和时钟信号的定时要求。下面以前述上升沿触发的 D 触发器为例进行说明。

传输延迟时间 t_{pd}

时钟脉冲 CP 上升沿至输出端新状态稳定建立起来的时间定义为 D 触发器的传输延迟时间。图 5.4.6 所示波形中, t_{pLH} 是输出 Q 从低电平到高电平的延迟时间, t_{pHL} 则是 Q 从高电平到低电平的延迟时间。应用中更多地取它们的平均值, 即平均传输延迟时间 $t_{pd} = (t_{pLH} + t_{pHL})/2$ 。

建立时间 t_{su}

输入信号 D 的变化会引起触发器内部电路逻辑电平的一系列变化, 为保证相关电路建立起稳定的状态, 信号 D 必须提前于时钟信号 CP 的上升沿(对上升沿触发的触发器而言)就稳定在指定的逻辑电平上, 以使触发器状态得到正确的转换。该提前时间的最小值即建立时间 t_{su} 。

保持时间 t_h

信号 D 在 CP 上升沿到来之后还应保持一定时间, 才能保证 D 状态可靠地传送到 Q 和 \bar{Q} 端, 该时间的最小值称为保持时间 t_h 。由于技术的进步, 已有多种触发器把保持时间几乎降到 0。这项特性在高速移位寄存器或计数器中是十分重要的。

触发脉冲宽度 t_w

为保证可靠触发, 要求时钟脉冲 CP 的宽度不小于 t_w , 以保证内部门电路有足够的时间实现正确的翻转。

最高时钟频率 f_{cmax}

触发器所能响应的时钟脉冲 CP 的最高频率, $f_{cmax} = 1/T_{cmin}$ 。因为 CP 无论在高电平还是在低电平期间, 触发器内部都要完成一系列动作, 存在一定的时间延迟, 所以对于 CP 最高工作频率有一个限制。

对于特定型号的 D 触发器, 上述动态特性参数在生产厂家的数据手册中都会给出。表 5.4.2 所示为环境温度在 $-40^{\circ}\text{C} \sim +85^{\circ}\text{C}$ 范围的几种典型 D 触发器的定时参数。

表 5.4.2 几种典型的 D 触发器定时参数

型号	功能	工作电压 V_{cc}/V	t_{pd}/ns^*		t_{su}/ns	t_h/ns	t_w/ns	f_{cmax}/MHz
			最小	最大				
74HC74	双 D 触发器	4.5	44	25	0	20	25	
74HC374	八 D 触发器	4.5	45	25	5	20	24	
74AUC1G74	单 D 触发器	2.5 ± 0.2	0.8	1.2	0.4	0.3	1	275
74LVC1G79	单 D 触发器	3.3 ± 0.3	1	4	1.3	1	2.5	160
74AUC1G79	单 D 触发器	2.5 ± 0.2	0.3	1.3	0.5	0.1	1.7	275

* 测试条件: 表中 74HC 系列逻辑电路的 Q 端负载电容为 $C_L = 50 \text{ pF}$, 其余逻辑系列为 $C_L = 15 \text{ pF}$ 。

(摘自 <http://www.ti.com/>)

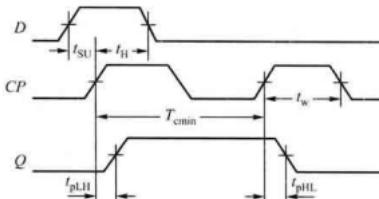


图 5.4.6 D 触发器定时图

5.4.4 其他电路结构的触发器

1. 维持阻塞触发器

维持阻塞结构的 D 触发器逻辑电路如图 5.4.7 所示。该触发器由 6 个与非门构成，其中， G_1 、 G_2 、 G_3 和 G_4 响应外部输入信号 D 和时钟信号 CP ，所产生的 \bar{S} 和 \bar{R} 信号控制由 G_5 、 G_6 构成的基本 $\bar{S}\bar{R}$ 锁存器的状态，也就是整个触发器的状态。下面分析其工作原理。

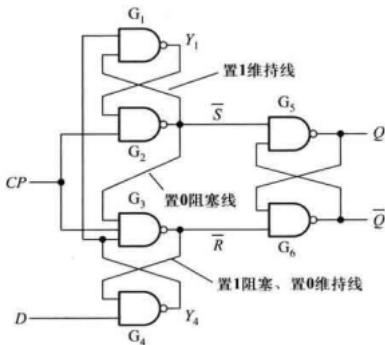


图 5.4.7 维持阻塞 D 触发器的逻辑电路

① 当 $CP = 0$ 时，与非门 G_2 和 G_3 被低电平封锁，它们的输出 $\bar{S}=\bar{R}=1$ ，使 G_5 、 G_6 组成的输出锁存器处于保持状态，触发器的输出 Q 和 \bar{Q} 不变。同时， \bar{S} 和 \bar{R} 的高电平分别反馈到 G_1 和 G_4 输入端，将这两个门打开，使 $Y_4=\bar{D}$ ， $Y_1=\bar{Y}_4=D$ ， D 信号进入触发器，为触发器状态更新做好准备。

② 当 CP 由 0 跳变到 1 后瞬间， G_2 和 G_3 打开， \bar{S} 和 \bar{R} 分别由 Y_1 和 Y_4 的状态所决定，即 $\bar{S}=\bar{Y}_1=D^*$ ， $\bar{R}=\bar{Y}_4=D^*$ 。这里， D^* 表示 CP 脉冲上升沿到来之前 D 的逻辑值。可以看出， \bar{S} 和 \bar{R} 的逻辑值始终是互补的，即二者中仅有 1 为 0。如果 $D^*=1$ ，则 $\bar{S}=\bar{D}^*=0$ ， $\bar{R}=D^*=1$ ，将输出基本 $\bar{S}\bar{R}$ 锁存器置 1，即触发器状态更新为 $Q^{**}=1$ 。在 $CP=1$ 期间， $\bar{S}=0$ 将 G_1 和 G_3 封锁。其中， \bar{S} 至 G_1 输入端的反馈线使 $Y_1=1$ ，起到维持 $\bar{S}=0$ 的作用，从而维持了触发器的 1 状态，故称之为置 1 维持线。虽然 D 信号在 $CP=1$ 期间可能由 1 跳变为 0，使 $Y_4=1$ ，但 \bar{S} 至 G_3 输入端的反馈线将阻止 $\bar{R}=1$ 状态的改变，从而阻塞了 D 端输入的置 0 信号。故称该反馈线为置 0 阻塞线。反之，假若 $D^*=0$ ，则 $\bar{S}=1$ ， $\bar{R}=0$ ，触发器状态更新为 $Q^{**}=0$ 。在 $CP=1$ 期间， \bar{R} 至 G_4 输入端反馈线上的信号将 G_4 封锁，使 $Y_4=1$ ，既阻塞了 $D=1$ 信号进入触发器的路径，又通过 $\bar{R}=Y_4 \cdot \bar{S} \cdot CP=0$ ，将

触发器维持在 0 状态。故这根反馈线称为置 1 阻塞、置 0 维持线。

待 CP 高电平结束, 电路又回到①所描述的状况。可以看出, 触发器只在 CP 由 0 跳变到 1 瞬间更新状态, 其余时间均处于保持不变的状态。

虽然维持阻塞 D 触发器的电路结构与前述主从 D 触发器完全不同, 但这两个电路所实现的逻辑功能是完全相同的, 都是在 CP 脉冲上升沿到来后瞬间转换输出状态, 将输入信号 D 传递到 Q 端并保持下去。因此, 它们使用同一逻辑符号和特性方程[即式(5.4.1)]来表达, 并具有十分相似的动态特性。

2. 利用传输延迟的触发器

一种利用传输延迟实现的 JK 触发器电路结构如图 5.4.8 所示。由 G_{11} 、 G_{12} 、 G_{13} 和 G_{21} 、 G_{22} 、 G_{23} 构成两个与或非门, 它们交叉耦合进一步构成 SR 锁存器作为触发器的输出电路, 而与非门 G_3 和 G_4 则构成触发器的输入电路接收输入信号 J 、 K 。另外, 在集成电路的工艺上保证 G_3 和 G_4 门的传输延迟时间大于 SR 锁存器的翻转时间。该触发器的工作原理如下。

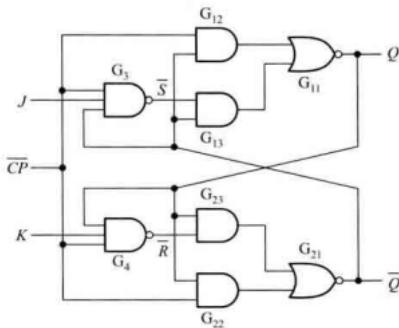


图 5.4.8 利用传输延迟的 JK 触发器的逻辑电路

① $\overline{CP} = 0$ 时, 不仅封锁了 G_{12} 、 G_{22} , 而且也封锁了 G_3 、 G_4 , 不论 J 、 K 为何状态, \bar{S} 、 \bar{R} 均为 1, 使 G_{13} 、 G_{23} 打开, 于是 G_{11} 和 G_{21} 形成交叉耦合的锁定状态, 输出 Q 、 \bar{Q} 保持原状态不变。

② \overline{CP} 由 0 变 1 后瞬间, G_{12} 、 G_{22} 两门传输延迟时间较短, 首先打开, 使 G_{11} 和 G_{21} 继续处于锁定状态, 输出仍保持不变。经过 G_3 、 G_4 稍长时间的传输延迟, \bar{S} 、 \bar{R} 才反映出信号 J 、 K 的作用。设 CP 由 0 到 1 跳变前触发器的状态为 Q'' , 根据图 5.4.8, 在此后的 $\overline{CP} = 1$ 期间

$$Q = \overline{\overline{CP} \cdot Q'' + \bar{S} \cdot Q''} = Q'' \quad (5.4.2)$$

$$\bar{Q} = \overline{\overline{CP} \cdot Q'' + R \cdot Q''} = \bar{Q}'' \quad (5.4.3)$$

说明触发器状态仍与 CP 跳变前相同。同时

$$\bar{S} = \overline{\overline{CP} \cdot J \cdot \bar{Q}^n} = \overline{J \bar{Q}^n} \quad (5.4.4)$$

$$\bar{R} = \overline{\overline{CP} \cdot K \cdot Q^n} = \overline{KQ^n} \quad (5.4.5)$$

无论 J, K 为何值, 若 $Q^n = 1$, 则从式(5.4.4)可得 $\bar{S} = 1$; 反之, $Q^n = 0$, 则从式(5.4.5)可得 $\bar{R} = 1$; 即 \bar{S}, \bar{R} 不可能同时为 0。这时, 电路已接收输入信号 J, K , 为触发器状态更新做好了准备。

(3) \overline{CP} 由 1 变 0 后的瞬间, G_{12}, G_{22} 两门首先关闭, 均输出为 0。

而 G_3, G_4 两门的延迟使 \bar{S}, \bar{R} 尚未变化, 此时仍为 $\bar{S} = \overline{J \bar{Q}^n}, \bar{R} = \overline{KQ^n}$, 输出 SR 锁存器可简化为图 5.4.9 所示电路, 其状态由 \bar{S}, \bar{R} 确定, 于是, 触发器状态由前一状态转换为下一状态。随着 G_3, G_4 的延迟关闭, \bar{S}, \bar{R} 均转换为 1, 触发器又进入①所分析的情况。

由于这种触发器的状态更新发生在时钟脉冲由 1 变 0 瞬间, 即时钟脉冲的下降沿, 属于如图 5.4.1(e) 所示波形触发的触发器, 所以我们从一开始就以 \overline{CP} 来表示这种触发器的时钟信号。为了区别 \overline{CP} 下降沿到来前、后触发器的状态, 以 Q^n 表示触发器现在的状态, 以 Q^{n+1} 表示下一状态, 根据图 5.4.9 的电路可得

$$Q^{n+1} = \overline{\overline{SR} \bar{Q}^n} = \overline{J \bar{Q}^n \bar{K} Q^n Q^n} \quad (5.4.6)$$

应用摩根定理进行整理, 得

$$Q^{n+1} = \overline{J \bar{Q}^n} + \overline{K Q^n} \quad (5.4.7)$$

上式称为 JK 触发器的特性方程。式中可见, Q^{n+1} 是 Q^n 和输入信号 J, K 的函数。

在 JK 触发器的动态特性中, 输出信号 Q 和 \bar{Q} 对时钟信号 \overline{CP} 下降沿亦存在传输延迟时间 t_{pd} , 输入信号 J, K 的逻辑电平至少应在时钟信号 \overline{CP} 下降沿到来之前 t_{su} 建立起来, 并至少保持一段时间 t_H , 对 \overline{CP} 脉冲的低电平宽度 t_w 和最高工作频率 f_{max} 也有一定的要求。这些参数均与前述 D 触发器类似, 不再赘述。

虽然在现代时序电路设计中使用 JK 触发器的机会已大大减少, 但是, 因为用它们构成时序电路所需的外部驱动电路有时比用 D 触发器简单, 所以至今还有应用。在较新的 CMOS 集成电路系列中也保留了专门的 JK 触发器芯片产品。

图 5.4.10 所示为 CMOS 双 JK 触发器 74LVC112A 中一个触发器的内部逻辑图, 与图 5.4.8 相比, 只不过改变了各个门电路在图中布局的位置, 并增加了直接置 1 端 \bar{S}_D 和直接置 0 端 \bar{R}_D 。74LVC112A 的逻辑符号如图 5.4.11 所示, 方框内侧的“>”符号和方框外侧与其相邻的圆圈共同表示该触发器对时钟信号的脉冲下降沿敏感。方框内 C1 与 1J, 1K 控制关联, 而 C2 则与 2J, 2K 控制关联。表 5.4.3 所示是 74LVC112A 中触发器的功能表。

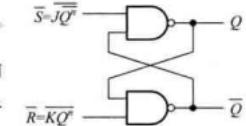


图 5.4.9 由 1 变 0 后瞬间输出 SR 锁存器的简化电路

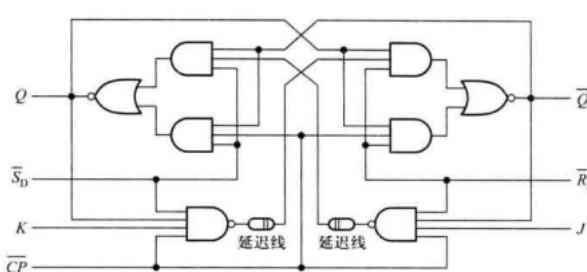


图 5.4.10 74LVC112A 中利用传输延迟的 JK 触发器逻辑图

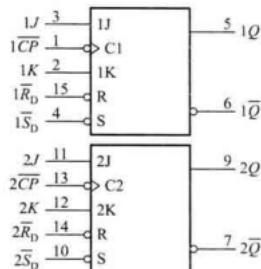


图 5.4.11 74LVC112A 的逻辑符号

表 5.4.3 74LVC112A 中 JK 触发器的功能表

输入					输出	
\bar{S}_b	\bar{R}_b	\bar{CP}	J	K	Q	\bar{Q}
L	H	×	×	×	H	L
H	L	×	×	×	L	H
L	L	×	×	×	H	H
\bar{S}_b	\bar{R}_b	\bar{CP}	J	K	Q^{n+1}	\bar{Q}^{n+1}
H	H	↓	L*	L*	Q^n	\bar{Q}^n
H	H	↓	L*	H*	L	H
H	H	↓	H*	L*	H	L
H	H	↓	H*	H*	\bar{Q}^n	Q^n

L* 和 H* 表示 \bar{CP} 脉冲下降沿到来之前瞬间的电平。

复习思考题

5.4.1 触发器有哪几种常见的电路结构？试归纳它们的工作原理和动作特点。

5.4.2 图 5.4.2 所示 D 触发器要求输入信号 D 与时钟信号 CP 有什么样的定时关系？输出信号 Q 和 \bar{Q} 与 CP 有什么样的定时关系？

5.4.3 试画出图 5.4.8 所示 JK 触发器的定时图，注意定时图中各信号与时钟脉冲 \bar{CP} 的哪一个边沿有定时关系。

5.5 触发器的逻辑功能

在 5.4 节中以两种 D 触发器和一种 JK 触发器为例,介绍了触发器的不同电路结构,本节将进一步讨论触发器的逻辑功能。触发器在每次时钟触发沿到来之前的状态称为现态,之后的状态称为次态。所谓触发器的逻辑功能,是指以输入信号和现态为变量,以次态为函数的逻辑关系,可以用特性表、特性方程或状态图来描述这种关系。按照触发器的逻辑功能,通常分为 D 触发器、 JK 触发器、 T 触发器和 SR 触发器等几种不同类型。它们的逻辑符号见图 5.5.1,各逻辑方框内分别标明了时钟信号与不同输入信号的控制关联。

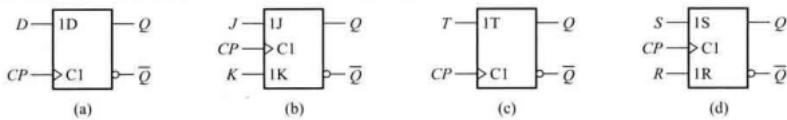


图 5.5.1 不同逻辑功能触发器的逻辑符号

(a) D 触发器 (b) JK 触发器 (c) T 触发器 (d) SR 触发器

需要指出的是,逻辑功能和电路结构是两个不同的概念。某一种逻辑功能的触发器可以用不同的电路结构来实现,如前述两种不同电路结构而功能完全相同的 D 触发器;同时,以某一种基本电路结构为基础,也可以构成不同逻辑功能的触发器,例如 5.5.5 节将要讨论的 D 触发器逻辑功能的转换。对于某种特定的电路结构,只不过是可能更易于实现某一逻辑功能而已。在本节讨论触发器的逻辑功能时,可以暂时不考虑其内部的电路结构。

5.5.1 D 触发器

1. 特性表

以输入信号和触发器的现态为变量,以次态为函数,描述它们之间逻辑关系的真值表称为触发器的特性表。 D 触发器的特性表如表 5.5.1 所示,表中对触发器的输入信号 D 和现态 Q^n 的每种组合都列出了相应的次态 Q^{n+1} 。

表 5.5.1 D 触发器的特性表

D	Q^n	Q^{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

2. 特性方程

触发器的逻辑功能也可以用逻辑表达式来描述, 称为触发器的特性方程。根据表 5.5.1 可以列出 D 触发器的特性方程:

$$Q^{n+1} = D \quad (5.5.1)$$

该式与 5.4 节中从触发器电路结构导出的式(5.4.1)完全相同。

3. 状态图

触发器的功能还可以用图 5.5.2 所示的状态图更为形象地表示。它同样可以由 D 触发器的特性表导出。图中, 圆圈内为触发器的状态 Q, 分别表示为 0 和 1 的两个圆圈代表了触发器的两个状态; 4 根带箭头的方向线表示状态转换的方向, 分别对应特性表中的 4 行, 方向线的起点为触发器的现态 Qⁿ, 箭头指向相应的次态 Qⁿ⁺¹; 方向线旁边标出了状态转换的条件, 即输入信号 D 的逻辑值。

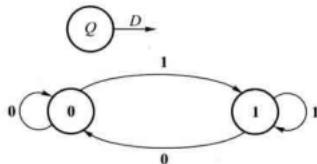


图 5.5.2 D 触发器的状态图

由特性表、特性方程或状态图均可看出, 当 D=0 时, D 触发器的下一状态将被置 0($Q^{n+1}=0$); 当 D=1 时, 将被置 1($Q^{n+1}=1$)。在时钟脉冲的两个触发沿之间, 触发器状态保持不变, 即存储 1 位二进制数据。

5.5.2 JK 触发器

1. 特性表

表 5.5.2 是 JK 触发器的特性表, 其中列出了触发器的输入信号 J、K 和现态 Qⁿ 在不同组合条件下的次态值。

表 5.5.2 JK 触发器的特性表

J	K	Q^n	Q^{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

2. 特性方程

从表 5.5.2 可以写出 JK 触发器次态的逻辑表达式, 经化简可得其特性方程如下:

$$Q^{n+1} = J \overline{Q^n} + \overline{K} Q^n \quad (5.5.2)$$

与由触发器电路结构导出的式(5.4.7)完全一致。

3. 状态图

JK 触发器的状态图如图 5.5.3 所示, 它可从表 5.5.2 导出。与 D 触发器的状态图在形式上的差别是它有两个输入变量, 所以每根方向线旁都标有两个逻辑值, 分别为 J, K 的值。可以注意到, 在每一个转换方向上, J, K 中总有一个是无关变量。例如, 表 5.5.2 的第 5 行和第 7 行, $Q^n = 0$ 转换为 $Q^{n+1} = 1$, 条件是 $J = 1$, 而 K 既可以取 0, 也可以取 1, 故状态图中的转换条件则以 $1 \times$ 表示。所以, 状态图中的 4 根方向线实际对应表中 8 行。

由上面的描述可以看出, 当 $J = K = 0$ 时, JK 触发器状态保持不变 ($Q^{n+1} = Q^n$); 当 $J = 0, K = 1$ 时, 触发器的下一状态将被置 0 ($Q^{n+1} = 0$); 当 $J = 1, K = 0$ 时, 将被置 1 ($Q^{n+1} = 1$); 当 $J = K = 1$ 时, 触发器的下一状态将翻转 ($Q^{n+1} = \overline{Q^n}$)。在所有逻辑类型的触发器中, JK 触发器具有最强的逻辑功能, 在外部 J, K 信号控制下, 它能执行保持、置 0、置 1 和翻转四种操作, 并可用简单的附加电路转换为其他功能的触发器。

例 5.5.1 设下降沿触发的 JK 触发器时钟脉冲 \overline{CP} 和 J, K 信号的波形如图 5.5.4 中横虚线上面所示, 试画出输出端 Q 的波形。设触发器的初始状态为 0。

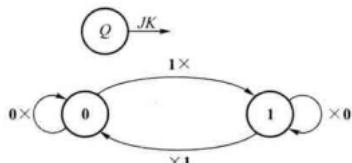


图 5.5.3 JK 触发器的状态图

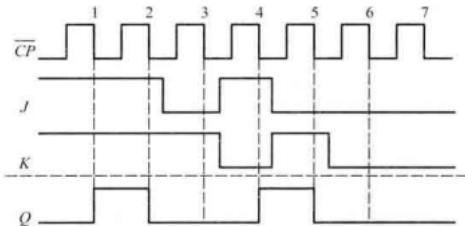


图 5.5.4 例 5.5.1 的波形图

解: 根据 JK 触发器的特性表、特性方程或状态图都可画出 Q 端波形, 如图 5.5.4 中横虚线下面所示。

从图 5.5.4 可以看出, 在第 1、2 个 \overline{CP} 脉冲作用期间, J, K 均为 1, 每输入一个脉冲, Q 端的状态就改变一次, 即触发器翻转一次。触发器的这种工作状态称为计数状态; 由触发器翻转的次数可以计算出时钟脉冲的个数。同时, Q 端的方波频率是时钟脉冲频率的二分之一。若以 CP (或

\overline{CP} 为输入信号, Q 为输出信号, 则一个触发器可作为二分频电路, 两个触发器级联可获得四分频, 其余类推。

5.5.3 T 触发器

在某些应用中, 需要对上述计数功能进行控制; 当控制信号 $T=1$ 时, 每来一个 CP (或 \overline{CP}) 脉冲, 它的状态翻转一次; 而当 $T=0$ 时, 则不对 CP (或 \overline{CP}) 信号作出响应而保持状态不变。具备这种逻辑功能的触发器称为 T (Toggle) 触发器。

1. 特性表

T 触发器的特性表如表 5.5.3 所示。

表 5.5.3 T 触发器的特性表

T	Q^n	Q^{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

2. 特性方程

从表 5.5.3 可以写出 T 触发器的特性方程:

$$Q^{n+1} = T \overline{Q^n} + \overline{T} Q^n \quad (5.5.3)$$

3. 状态图

T 触发器的状态图如图 5.5.5 所示。

由此可知, T 触发器的功能是: $T=1$ 时为翻转状态, $Q^{n+1} = \overline{Q^n}$; $T=0$ 时为保持状态, $Q^{n+1} = Q^n$ 。

比较式(5.5.3)和式(5.5.2), 如果令 $J=K=T$, 则两式等效。事实上只要将 JK 触发器的 J 、 K 端连接在一起作为 T 输入端, 就可实现 T 触发器的功能, 因此, 在小规模集成电路产品中没有专门的 T 触发器, 如果有需要, 可用其他功能的触发器转换。

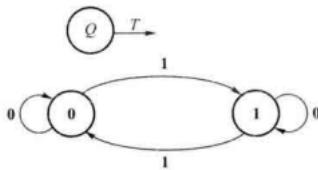


图 5.5.5 T 触发器的状态图

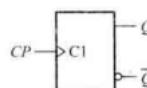


图 5.5.6 T' 触发器的逻辑符号

4. T' 触发器

当 T 触发器的 T 输入端固定接高电平时(即 $T=1$)，则式(5.5.3)变为

$$Q^{n+1} = \overline{Q^n} \quad (5.5.4)$$

也就是说，时钟脉冲每作用一次，触发器翻转一次。这种特定的 T 触发器常在集成电路内部逻辑图中出现，其输入只有时钟信号，有时称为 T' 触发器。上升沿触发的 T' 触发器逻辑符号如图 5.5.6 所示。

5.5.4 SR 触发器

仅有置位、复位功能的触发器称为 SR 触发器，它的特性表如表 5.5.4 所示。从表中可以看出， $S=R=1$ 时，触发器的次态是不能确定的，如果出现这种情况，触发器将失去控制。因此， SR 触发器的使用必须遵循 $SR=0$ 的约束条件。从特性表可导出表达次态与现态、输入信号关系的表达式

$$\begin{cases} Q^{n+1} = \overline{S} \overline{R} \overline{Q^n} + \overline{S} R Q^n + \overline{S} \overline{R} Q^n = \overline{S} \overline{R} + \overline{S} R Q^n \\ SR = 0 \text{ (约束条件)} \end{cases}$$

由于约束条件限制，当 $S=1$ 时， $R=0$ ， $Q^{n+1}=S$ ；当 $S=0$ 时， $Q^{n+1}=\overline{R}Q^n$ 。上式可进一步化简，于是得到特性方程

$$\begin{cases} Q^{n+1} = S + \overline{R} Q^n \\ SR = 0 \quad (\text{约束条件}) \end{cases} \quad (5.5.5)$$

表 5.5.4 SR 触发器的特性表

S	R	Q^n	Q^{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	不确定
1	1	1	不确定

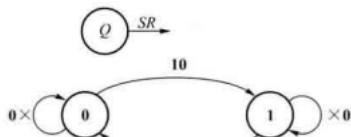


图 5.5.7 SR 触发器的状态图

从特性表也可以导出状态图，如图 5.5.7 所示。

事实上，生产厂家罕有专门的 SR 触发器芯片提供，它主要出现于集成电路的内部结构，需要单独使用 SR 触发器时，可以 JK 触发器直接代用。比较图 5.5.3 和图 5.5.7 的状态图，令 $J=S, K=R$ ，便可用 JK 触发器实现 SR 触发器的全部有效功能。

5.5.5 D 触发器逻辑功能的转换

如前所述,D触发器和JK触发器具有较完善的功能,有很多独立的中、小规模集成电路产品,而T触发器和SR触发器可以很容易用D触发器或JK触发器转换而成。下面将仅讨论应用普遍的D触发器逻辑功能的转换,其他触发器之间逻辑功能的相互转换,有些在前文中已述及,有些则请读者采用类似的方法举一反三。

1. D触发器构成JK触发器

比较D触发器和JK触发器的特性方程,即式(5.5.1)和式(5.5.2),可以令

$$Q^{n+1} = D = J \bar{Q}^n + \bar{K}Q^n \quad (5.5.6)$$

按上式,可得电路如图5.5.8所示,电路特性符合JK触发器的特性方程,从而能实现JK触发器的所有功能。有些CMOS主从结构的集成JK触发器就是采用类似方式实现的。

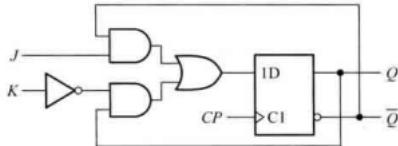


图 5.5.8 用 D 触发器实现 JK 触发器的逻辑功能

2. D触发器构成T触发器

用构成JK触发器相同的方法,令

$$Q^{n+1} = D = T \bar{Q}^n + \bar{T}Q^n = T \oplus Q^n = T \odot \bar{Q}^n \quad (5.5.7)$$

只需在D输入端前增加一个异或门或者同或门即可实现,于是得到如图5.5.9(a)、(b)所示的两种T触发器逻辑电路。

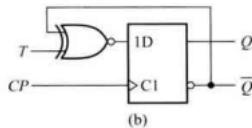
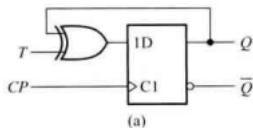


图 5.5.9 用 D 触发器实现 T 触发器的逻辑功能

(a) 用异或门实现 (b) 用同或门实现

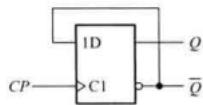


图 5.5.10 用 D 触发器实现 T' 触发器的逻辑功能

3. D触发器构成T'触发器

比较式(5.5.1)和式(5.5.4)可得 $Q^{n+1} = D = \bar{Q}^n$,于是,画出用D触发器构成的T'触发器如图5.5.10所示。

复习思考题

- 5.5.1 触发器的逻辑功能和电路结构之间有什么关系?
- 5.5.2 写出 D 触发器、JK 触发器、T 触发器和 SR 触发器的特性方程,并画出它们的状态图。
- 5.5.3 试用 JK 触发器实现 D 触发器、T 触发器、T' 触发器以及 SR 触发器的逻辑功能。

5.6 用 Verilog HDL 描述锁存器和触发器

4.6 节介绍了组合逻辑电路的行为级建模,本节将讨论时序电路的基本部件——锁存器和触发器的建模,首先介绍有关的基础知识,然后以实例进行说明。

5.6.1 时序逻辑电路建模基础

在 Verilog 中,行为级描述主要使用由关键词 **initial** 或 **always** 定义的两种结构类型的语句。一个模块的内部可以包含多个 **initial** 或 **always** 语句,仿真时这些语句同时并行执行,即与它们在模块内部排列的顺序无关,都从仿真的 0 时刻开始执行。

initial 语句是一条初始化语句,仅执行一次,在测试模块中用它对激励信号进行描述,在硬件电路的行为描述中,有时为了仿真的需要,也用 **initial** 语句给寄存器变量赋初值。**initial** 语句主要是一条面向仿真的过程语句,逻辑综合工具不支持 **initial** 语句,因而不作介绍。

always 本身是一个无限循环语句,即不停地循环执行其内部的过程语句,直到仿真过程结束。但用它来描述硬件电路的逻辑功能时,通常在 **always** 后面紧跟着循环的控制条件,所以 **always** 语句的一般用法如下:

```
always @ (事件控制表达式)
begin
    块内部局部变量的定义;
    过程赋值语句;
end
```

这里,“事件控制表达式”也称为敏感事件表,即等待确定的事件发生或某一特定的条件变为“真”,它是执行后面过程赋值语句的条件。“过程赋值语句”左边的变量必须被定义成 **reg** 数据类型,右边变量可以是任意数据类型。**begin** 和 **end** 将多条过程赋值语句包围起来,组成一个顺序语句块,块内的语句按照排列顺序依次执行,最后一条语句执行完后,执行挂起,然后 **always** 语句处于等待状态,等待下一个事件的发生。注意,当 **begin** 和 **end** 之间只有一条语句,且没有定义局部变量时,则关键词 **begin** 和 **end** 可以被省略。

在 Verilog 中,将逻辑电路中的敏感事件分为两种类型:电平敏感事件和边沿触发事件。在组合电路中,输入信号的变化直接会导致输出信号的变化;时序电路中的锁存器输出在使能信号

为高电平时亦随输入电平而变化,如图 5.4.1(a)所示波形。这种对输入信号电平变化的响应称为电平敏感事件。

例如,例 4.6.1 中对 4 选 1 数据选择器的描述语句

```
always @ ( D, S )
```

说明 S 或 D 中任意一个信号的电平发生变化(即有电平敏感事件发生),后面的过程赋值语句将会执行一次。

而触发器状态的变化仅仅发生在时钟脉冲的上升沿或下降沿,如图 5.4.1(b)、(c)所示波形。Verilog 中分别用关键词 **posedge**(上升沿)和 **negedge**(下降沿)进行说明,这就是边沿敏感事件。例如,语句

```
always @ ( posedge CP or negedge CR ) //Verilog 1995 syntax
```

说明在时钟信号 CP 的上升沿到来或在清零信号 CR 跳变为低电平时,后面的过程语句将被执行。注意,关键词 **negedge** 不能省略,因为敏感事件列表中不能同时包含电平敏感事件和边沿触发事件。

修订后的 Verilog 语言在敏感事件列表中,允许用逗号进行分隔多个不同的事件。例如,上面的语句可以改写成下面的形式:

```
always @ ( posedge CP, negedge CR ) //Verilog 2001,2005 syntax
```

always 语句内部的过程赋值语句有两种类型:阻塞型赋值语句(Blocking Assignment Statement)和非阻塞型赋值语句(Non-Blocking Assignment Statement)。所使用的赋值符分别为“=”和“<=”,通常称“=”为阻塞赋值符,“<=”为非阻塞赋值符。在串行语句块中,阻塞型赋值语句按照它们在块中排列的顺序依次执行,即前一条语句没有完成赋值之前,后面的语句不能被执行,换言之,前面的语句阻塞了后面语句的执行,这与在一个公路收费站,汽车必须排队顺序前进缴费有些类似。例如,下面两条阻塞型赋值语句的执行过程是:首先执行第一条语句,将 A 的值赋给 B,接着执行第二条语句,将 B 的值(等于 A 值)加 1,并赋给 C,执行完后,C 的值等于 A+1。也就是说,如果使用“=”给一个变量 B 赋了新值,则该块后面的语句若用到该变量 B 时将使用新值。

```
begin
    B = A;
    C = B+1;
end
```

Verilog 中由“<=”符号构成的非阻塞型赋值语句的执行过程是:首先计算语句块内部所有右边表达式的值,然后完成对左边寄存器变量的赋值操作,这些操作是并行执行的。例如,下面两条非阻塞型赋值语句的执行过程是:首先计算所有表达式右边的值并分别存储在暂存器中,即 A 的值被保存在一个暂存器中,而 B+1 的值被保存在另一个暂存器中,在 **begin** 和 **end** 之间所有非阻塞型赋值语句的右边表达式都被计算并存储后,对左边寄存器变量的赋值操作才会进行。这样,C 的值等于 B 的原始值(而不是 A 的值)加 1。也就是说,**always** 块中所有非阻塞赋值语

句在求值时,所用的值全部是进入 **always** 块时各个变量的原始值。

```
begin
    B <= A;
    C <= B+1;
end
```

综上所述,阻塞型赋值语句和非阻塞型赋值语句的主要区别是完成赋值操作的时间不同,前者的赋值操作是立即执行的,即执行后一句时,前一句的赋值已经完成;而后的赋值操作要到顺序块内部的多条非阻塞型赋值语句运算结束时,才同时并行完成赋值操作,一旦赋值操作完成,语句块的执行也就结束了。需要注意的是,在可综合的电路设计中,一个语句块的内部只允许出现唯一一种类型的赋值语句,而不允许阻塞型赋值语句和非阻塞型赋值语句二者同时出现。在时序电路的设计中,建议采用非阻塞型赋值语句。

5.6.2 锁存器和触发器的 Verilog 建模实例

本节给出一些锁存器和触发器的行为级描述实例。

例 5.6.1 根据功能表 5.3.1,对 D 锁存器的行为进行描述。

解:对于锁存器来说,当输入控制信号处于有效电平时(即 $E = 1$ 时),其输出 Q 跟随输入信号 D 的变化;当控制信号无效时,输出 Q 保持不变。所以在 **always** 语句中@ 符号之后的“事件控制表达式”使用了电平敏感事件,说明如果输入信号 E 或者 D 发生变化,就会执行一次后面的 **if** 语句,但只有 E 为逻辑 1 时,输入 D 的变化才能传送到输出 Q ;否则,输出 Q 将保持不变。

注意,由于输出 Q 在过程语句中被赋值,所以必须将它声明为 **reg** 类型的变量。

```
module D_latch (Q, QN, D, E); //IEEE 1364—1995 Syntax
    input D, E; //输入端口声明
    output reg Q; //输出端口声明
    output QN; //输出端口声明
    assign QN = ~ Q; //Continuous assignment
    always @ (E or D)
        if (E) Q <= D; //Same as: if (E == 1)
endmodule
```

例 5.6.2 分析下列程序,说明它所完成的逻辑功能。

(1)

```
module DFF (output reg Q, input D, input CP); // Verilog 2001, 2005 syntax
    always @ (posedge CP) //Elementary D flip-flop
        Q <= D;
endmodule
```

(2)

```

module async_set_rst_DFF (//Verilog 2001, 2005 syntax
    output reg Q, QN, //D flip-flop with asynchronous set and reset
    input D, CP, Sd, Rd
);
always @ (posedge CP, negedge Sd, negedge Rd)
    if (~Sd || ~Rd)
        if (~Sd) begin Q <= 1'b1; QN <= 1'b0; end
        else begin Q <= 1'b0; QN <= 1'b1; end
    else
        begin Q <= D; QN <= ~D; end
endmodule

```

(3)

```

module sync_rst_DFF (//Verilog 2001, 2005 syntax
    output reg Q, // D flip-flop with synchronous reset.
    input D, CP, Rd
);
always @ (posedge CP)
    if (~Rd) Q <= 1'b0;
    else Q <= D;
endmodule

```

解：例 5.6.2 对上升沿触发的 D 触发器的行为进行了描述。

(1) 第一个模块对基本 D 触发器的行为进行了描述。该模块的输出信号为 Q, 输入信号为 D 和 CP。在 always 语句@ 符号之后的“事件控制表达式”中使用了边沿触发事件, 即 posedge CP, 使其后的语句 $Q \leq D$ 仅在 CP 上升沿期间将 D 的值赋给 Q, 而在其他任何时间, 无论 D 信号如何变化, 都不能改变 Q 的状态。

(2) 第二个模块对具有异步直接置 1、置 0 功能的 D 触发器的行为进行了描述。在 always 语句的“事件控制表达式”中, 比前一模块增加了两个异步触发事件 negedge Sd 和 negedge Rd。在这种表达式中, 可以有一个或多个异步事件, 但必须有一个事件是时钟事件, 在 IEEE 1364—1995 标准中, 要求多个事件之间用关键词 or 进行连接, 修订后的标准可以用逗号隔开多个异步事件。这个模块中的触发事件表示, 在输入信号 CP 的上升沿到来, 或 Sd 或 Rd 跳变为低电平时, 后面的 if-else 语句就会被执行一次。negedge Sd 和 negedge Rd 是两个异步事件, 它与 **if**(~Sd || ~Rd) 语句相匹配。如果条件成立, 接下来, 如果 Sd 为逻辑 0(**if**(~Sd)), 则将输出 Q 置 1, QN 置 0; 否则将执行 else 后面的两条语句, 将输出 Q 置 0, QN 置 1; 如果 Sd 和 Rd 均不为 0, 当时钟 CP 上升沿到来时, 将输入 D 传送到输出 Q, ~D 传送到 QN。从语句执行的顺序来看, 如果直接置 1、置 0 和时钟上升沿三个事件同时发生, 则置 1 事件拥有最高的优先级别, 其次是

置 0 事件,时钟上升沿事件的优先级别最低。

(3) 第三个模块对具有同步置 0 功能的 D 触发器的行为进行了描述。即置 0 信号 Rd 要在 CP 脉冲上升沿到来后才起作用。于是,在 **always** 语句中@ 符号之后的“事件控制表达式”中只有一个时钟事件,它表示只有在 CP 的上升沿到来时,后面的 **if-else** 语句才会被执行,此时首先检查 Rd 信号,如果 Rd 为逻辑 0,则将输出 Q 置 0;否则,将输入 D 传给输出 Q。显然,在该语句块中,置 0 信号 Rd 仍具有优先权,只有 Rd=1 时,才有可能执行 $Q \leftarrow D$ 语句。

例 5.6.3 根据特性表 5.5.2,对下降沿触发的 JK 触发器行为进行描述。

解:根据 JK 触发器的特性表,使用 **case** 多路分支语句进行描述。这里,将输入变量 J、K 拼接起来成为一个两位二进制变量($|J, K|$),它的值可能是二进制数 00、01、10、11,**case** 语句后面的 4 条分支语句正好说明了在时钟信号 CP 下降沿作用后,触发器的次态。

```
module JK_FF ( output reg Q, output Qnot, input J, K, CP ); //Verilog 2001, 2005 syntax
  assign Qnot = ~ Q ; //Continuous assignment
  always @ ( negedge CP )
    case ( |J, K| ) //Switch based on concatenation of J and K signals
      2'b00: Q <= Q;
      2'b01: Q <= 1'b0;
      2'b10: Q <= 1'b1;
      2'b11: Q <= ~ Q;
    endcase
  endmodule
```

例 5.6.4 分析下列程序,说明阻塞赋值语句和非阻塞赋值语句之间的区别。

(1) module Circuit_A (output reg f, g, input A, B, C, CP); always @ (posedge CP) begin f = A & B; g = f C; end endmodule	(2) module Circuit_B (output reg f, g, input A, B, C, CP); always @ (posedge CP) begin f <= A & B; g <= f C; end endmodule
--	--

解:模块 Circuit_A 使用阻塞赋值语句建模,因为 **always** 语句@ 符号之后的敏感事件表中使用了边沿触发事件,即 **posedge CP**,所以 f 和 g 都是 D 触发器的输出。由于用的是阻塞赋值,所以由 $f = A \& B$ 产生的新 f 值,在接下来的语句 $g = f | C$ 中就会被使用,这意味着 $g = f | C$ 实际

上与 $g = A \& B + C$ 是等效的。综合后得到的电路如图 5.6.1 所示。

模块 Circuit_B 使用非阻塞赋值语句建模, 进入 **always** 语句之后, 会使用各个变量的原始值将两个表达式右边的值计算出来并存入暂存器, 在 **always** 语句结束时 f 和 g 的值被同时更新。这意味着计算 $g \leq f + C$ 时使用了 f 的前一个值, 即或门的一个输入端应该连接到触发器的输出端 f。综合后得到的电路如图 5.6.2 所示。

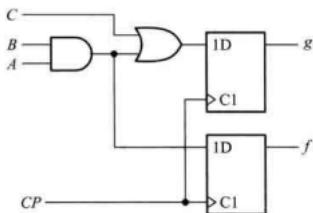


图 5.6.1 模块 Circuit_A 的电路

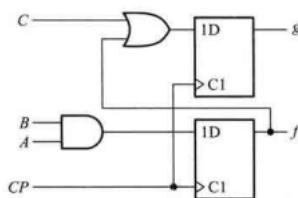


图 5.6.2 模块 Circuit_B 的电路

假设分别将两个模块中给 f 和 g 赋值的两条语句的次序颠倒一下, 那么会综合成什么样的电路呢?

对模块 Circuit_A 来说, 由于用的是阻塞赋值, **begin-end** 之间的语句是顺序执行的, 语句次序颠倒后, 首先执行 $g = f + C$, 此时 f 的值就是进入 **always** 语句时的值, 即 CP 上升沿到来之前的值, 所以综合后生成的电路会变成图 5.6.2 所示的那样。对模块 Circuit_B 来说, 由于用的是非阻塞赋值, **begin-end** 之间的语句是并行执行的, 所以语句次序颠倒对综合器生成电路没有任何影响。

可见, 阻塞赋值对语句的顺序具有依赖性, 用它对时序电路建模存在一定的风险, 所以, 对时序逻辑电路时, 建议使用非阻塞赋值语句。

复习思考题

- 5.6.1 在 Verilog 中, **initial** 语句和 **always** 语句的主要区别是什么?
- 5.6.2 在 **always** 语句中对电平敏感事件和边沿敏感事件的描述有何不同?
- 5.6.3 阻塞型赋值和非阻塞型赋值有何区别?
- 5.6.4 试用 Verilog 描述一个基本 SR 锁存器和一个下降沿触发的 SR 触发器。

小 结

- 双稳态电路存在两个稳定状态, 从而可存储、记忆 1 位二进制数据。锁存器和触发器都属于双稳态电路, 它们是时序电路的基本存储单元。置 0、置 1 和数据保持是一个存储单元最基

本的逻辑功能。

- 锁存器是对脉冲电平敏感的存储电路，它们在一定电平作用下才能改变状态。基本 SR 锁存器由输入信号电平直接控制其状态，门控 SR 锁存器在使能信号有效的条件下由输入信号决定其状态。

- D 锁存器在工作中没有约束条件，因而应用较广泛。在使能信号作用期间，D 锁存器输出跟随输入信号 D 而变化。其中传输门控 D 锁存器在中、大规模 CMOS 集成电路中应用较多，是构成触发器的基础电路。

- 触发器是对时钟脉冲边沿敏感的存储电路，它们在时钟脉冲的上升沿或下降沿作用下更新状态。目前流行的触发器电路主要有主从、维持阻塞和利用传输延迟等几种电路结构，它们的工作原理各不相同。

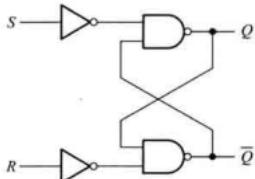
- 触发器按逻辑功能分类有 D 触发器、JK 触发器、T 触发器和 SR 触发器等几种类型。它们的功能可用特性表、特性方程和状态图来描述。触发器的电路结构与逻辑功能没有必然联系。例如 JK 触发器既有主从结构的，也有维持阻塞或利用传输延迟结构的。每一种逻辑功能的触发器都可以通过适当增加外部连线和门电路转换为其他逻辑功能的触发器。

- 可以用 Verilog 语言对锁存器和触发器的行为特性进行建模。Verilog 语言提供了两种赋值符：阻塞型赋值和非阻塞型赋值。对组合逻辑电路建模时，建议使用阻塞型赋值；对时序电路建模时，建议使用非阻塞型赋值。

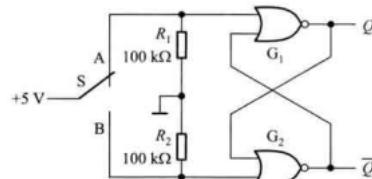
习题

5.2 SR 锁存器

5.2.1 分析图题 5.2.1 所示电路的功能，列出功能表。



图题 5.2.1



图题 5.2.2

5.2.2 用 CMOS 电路 74HCT02 或非门构成消除机械开关抖动影响的电路如图题 5.2.2 所示，试画出在开关 S 由位置 A 到 B 时 Q 和 \bar{Q} 端的波形。如改用 TTL 电路 74LS02 实现， R_1, R_2 取值的大致范围为多少？整个电路的功耗会发生什么变化？

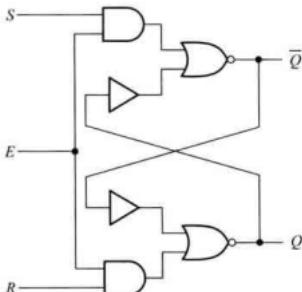
5.2.3 SR 锁存器如图 5.2.1(a) 所示，其初始状态为 $Q=0, \bar{Q}=1$ ，图中两个或非门的传输延迟时间均为

10 ns-SR 锁存器的输入波形如图题 5.2.3 所示,虚线表示的时间间隔为 10 ns,画出考虑或非门传输延迟时间的输出波形。假定输入和输出信号的上升和下降时间均为 0。

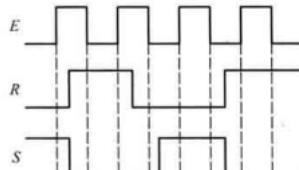


图题 5.2.3

5.2.4 由与或非门组成的 SR 锁存器如图题 5.2.4 所示,试分析其工作原理并列出功能表。



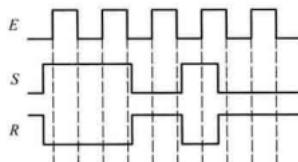
图题 5.2.4



图题 5.2.5

5.2.5 图题 5.2.4 所示锁存器的 E 、 R 、 S 端的输入信号波形如图题 5.2.5 所示,试画出 Q 和 \bar{Q} 端的波形。设初始状态 $Q=0$ 。

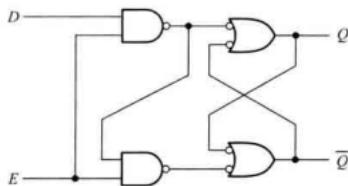
5.2.6 若图 5.2.8(a) 所示电路的初始状态为 $Q=1$, E 、 S 、 R 端的输入信号如图题 5.2.6 所示,试画出相应 Q 和 \bar{Q} 端的波形。



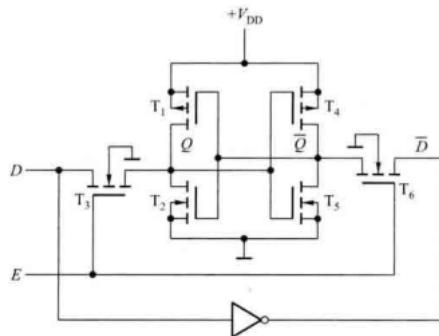
图题 5.2.6

5.3 D 锁存器

5.3.1 列出图题 5.3.1 所示电路的功能表,并与表 5.3.1 比较,证明该电路的逻辑功能与图 5.3.1(a) 和图 5.3.3 所示的锁存器相同。与图 5.3.3 所示电路比较,图题 5.3.1 所示电路有何优点?



图题 5.3.1



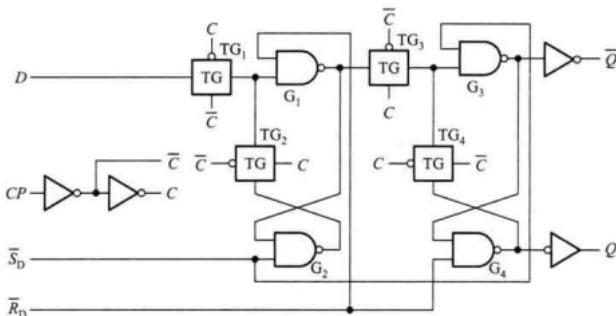
图题 5.3.2

5.3.2 一种简单的 D 锁存器晶体管电路如图题 5.3.2 所示, 试列出该电路的功能表。这种电路广泛用于 CMOS 静态随机存储器 (SRAM) 中作为基本存储单元。

5.3.3 试用 1 片八 D 锁存器 74HC373 设计一个能锁存两组 BCD 码信号的锁存电路。假定三态输出使能端 $\overline{OE} = 0$, 锁存器原输出为 $Q_7, Q_6, Q_5, Q_4 = 1001(9_b)$ 和 $Q_3, Q_2, Q_1, Q_0 = 0100(4_b)$, 输入为 $D_7, D_6, D_5, D_4 = 1001(9_b)$ 和 $D_3, D_2, D_1, D_0 = 0101(5_b)$, 画出锁存器锁存新数据前、后使能端 LE 应输入的波形和相应 Q_0 的波形。74HC373 的内部逻辑图请参见图 5.3.5。

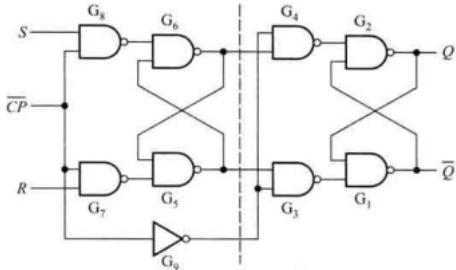
5.4 触发器的电路结构和工作原理

5.4.1 触发器的逻辑电路如图题 5.4.1 所示, 确定其属于何种电路结构的触发器并分析工作原理。

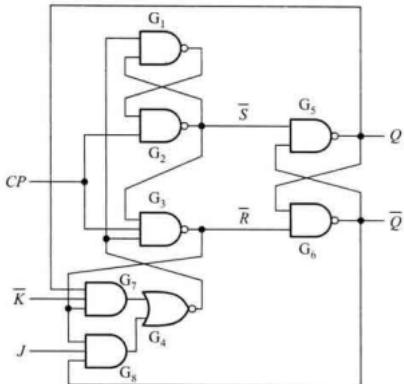


图题 5.4.1

5.4.2 触发器的逻辑电路如图题 5.4.2 所示, 确定其应属于何种电路结构的触发器。



图题 5.4.2



图题 5.4.3

5.4.3 触发器的逻辑电路如图题 5.4.3 所示, 确定其属于何种电路结构的触发器。

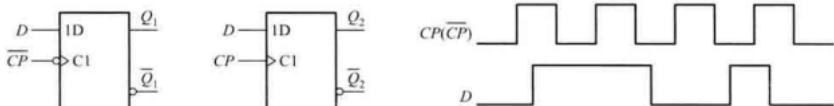
5.4.4 根据对图 5.4.8 所示电路的分析, 列出其功能表。

5.4.5 图 5.4.7 所示维持阻塞 D 触发器在 $D=1$ 时保持时间 t_H 为零, 分析其原因。

5.4.6 图 5.4.8 所示触发器电路的动态特性参数为: $t_{st}=5 \text{ ns}$, $t_{phL}=10 \text{ ns}$, $t_{phH}=15 \text{ ns}$ 。画出这种触发器的定时图。

5.5 触发器的逻辑功能

5.5.1 上升沿触发和下降沿触发的 D 触发器逻辑符号及时钟信号 $CP(\overline{CP})$ 和输入信号 D 的波形如图题 5.5.1 所示。分别画出它们的 Q 端波形。设触发器的初始状态为 0。



图题 5.5.1

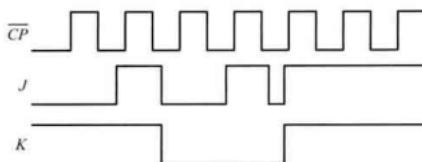
5.5.2 试用卡诺图化简表 5.5.2 表达的逻辑关系, 并将结果与式(5.5.2)核对。

5.5.3 设下降沿触发的 JK 触发器初始状态为 0, \overline{CP}, J, K 信号如图题 5.5.3 所示, 试画出触发器 Q 端的输出波形。

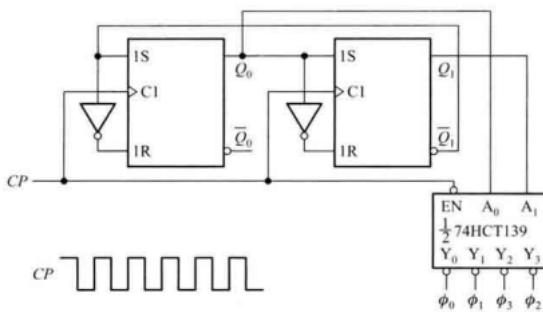
5.5.4 逻辑电路如图题 5.5.4 所示, 初始状态为 $Q_0=Q_1=0$, 试画出在 CP 作用下, ϕ_0, ϕ_1, ϕ_2 和 ϕ_3 的波形。

5.5.5 电路如图题 5.5.5 所示, 设各触发器的初始状态为 0, 画出在 CP 脉冲作用下 Q 端波形。

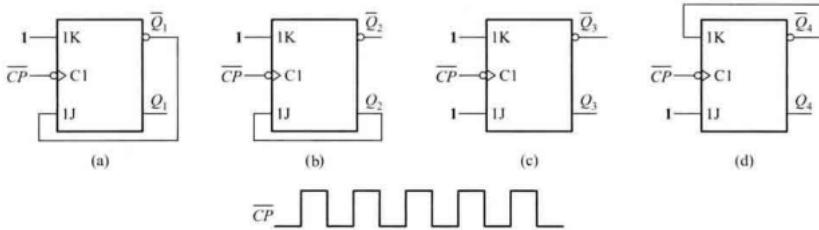
5.5.6 逻辑电路如图题 5.5.6 所示, 已知 \overline{CP} 和 X 的波形, 试画出 Q_1 和 Q_2 的波形。触发器的初始状态均为 0。



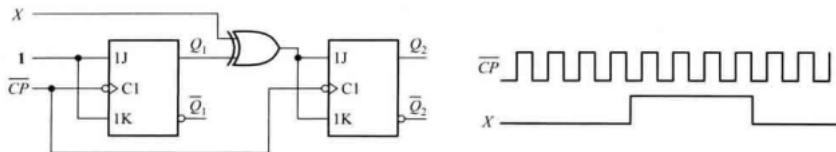
图题 5.5.3



图题 5.5.4

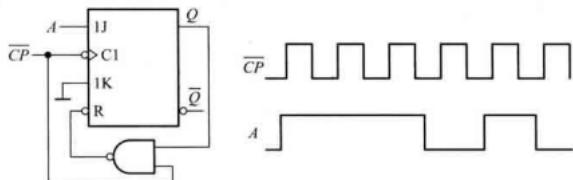


图题 5.5.5



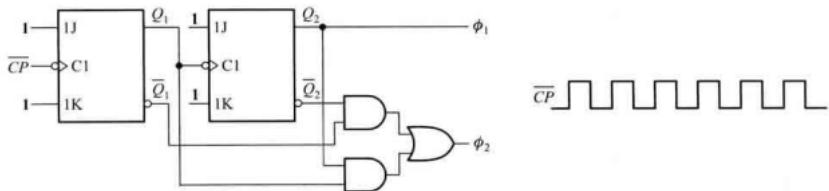
图题 5.5.6

5.5.7 逻辑电路如图题 5.5.7 所示, 已知 \overline{CP} 和 A 的波形, 画出触发器 Q 端的波形, 设触发器的初始状态为 0。



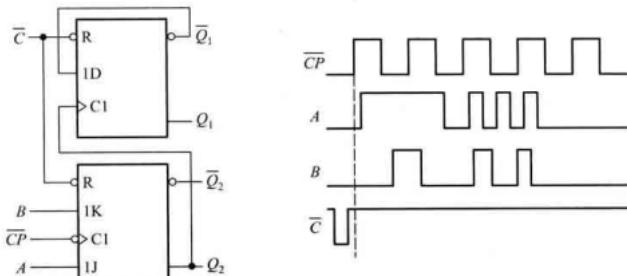
图题 5.5.7

5.5.8 两相脉冲产生电路如图题 5.5.8 所示, 试画出在 \overline{CP} 作用下 ϕ_1 、 ϕ_2 的波形, 并说明 ϕ_1 和 ϕ_2 的时间关系。各触发器的初始状态为 0。



图题 5.5.8

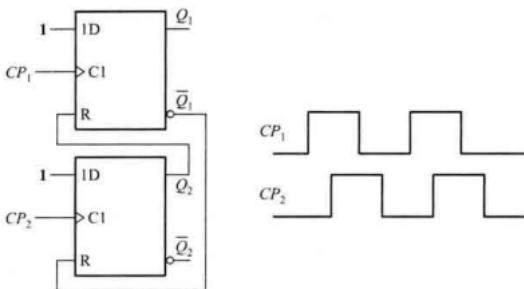
5.5.9 逻辑电路和各输入信号波形如图题 5.5.9 所示, 画出两触发器 Q 端的波形。两触发器的初始状态均为 0。



图题 5.5.9

5.5.10 逻辑电路和输入信号波形如图题 5.5.10 所示, 画出各触发器 Q 端的波形。触发器的初始状态均为 0。

5.5.11 试用 T 触发器和适当的组合逻辑实现 D 触发器的逻辑功能。



图题 5.5.10

5.5.12 试用 T 触发器和适当的组合逻辑实现 JK 触发器的逻辑功能。

5.6 用 Verilog HDL 描述锁存器和触发器

5.6.1 试说明下列程序所完成的逻辑功能，并画出它的逻辑图。

```
module d_latch_rst (Rd, control, D, Q);
    input Rd, control, D;
    output Q;
    reg Q;
    always @ ( Rd or control or D )
        if ( ~Rd ) Q <= 1'b0;
        else if ( control )
            Q <= D;
endmodule
```

5.6.2 试用行为建模方式描述一个下降沿触发的 D 触发器，要求具有异步置零功能，即置零信号变为低电平时，将触发器的输出置零。

5.6.3 阅读下列两个程序，画出它们的逻辑图。

(1)

```
module DFF1 ( Qa, Qb, D, CP );
    input D, CP;
    output Qa, Qb;
    reg Qa, Qb;
    always @ ( posedge CP )
        begin
            Qa = D;
            Qb = Qa;
        end
endmodule
```

(2)

```
module DFF2 ( Qa, Qb, D, CP );
    input D, CP;
    output Qa, Qb;
    reg Qa, Qb;
    always @ ( posedge CP )
        begin
            Qa <= D;
            Qb <= Qa;
        end
endmodule
```

6 >>>

时序逻辑电路

引言

本章将在组合逻辑电路和锁存器、触发器的基础上讨论时序电路。在第4章所讨论的组合逻辑电路中,任一时刻的输出信号仅仅由该时刻的输入信号所决定,而时序电路在任一时刻的输出信号不仅与当时的输入信号有关,而且与电路原来的状态有关。也就是说,时序电路中除具有逻辑运算功能的组合电路外,还必须有能够记忆电路状态的存储单元或延迟单元,这些存储或延迟逻辑单元主要由锁存器或触发器组成。

本章首先介绍时序电路的基本概念,然后讨论其分析与设计方法。其中,重点侧重于同步时序电路的分析与设计。在明确基本概念和基本方法之后,介绍逻辑设计中常用的典型时序电路模块,以及简单的时序可编程逻辑器件GAL。最后通过简单实例介绍用Verilog描述时序电路的方法。

6.1 时序逻辑电路的基本概念

日常生活中,时序逻辑的实例并不鲜见,例如楼房电梯的控制便是一个典型的时序逻辑问题。电梯的控制电路需要根据电梯内和各楼层入口处的按键信号,以及电梯当前的状态来决定电梯的升降,同时将电梯当前所处楼层信号输出到电梯内外。这里,按键信号和所到达的楼层信号是时序逻辑的“输入信号”,升降信号和到达楼层的显示则是其“输出信号”。显然,控制电路中必须具有存储单元,以记忆当前电梯所在的楼层。可以定义电梯目前所处楼层为现在的状态,简称“现态”,将要到达的楼层为下一个状态,简称“次态”,楼层的变换即为“状态转换”。电梯的升降不仅取决于当前各按键的输入信号,而且取决于电梯运转的历史状态。例如,电梯正在从低楼层向高楼层上升,若这时电梯内已有人按下更高楼层的按键,或更高的楼层有人召唤,电梯则应向高一层转换其状态而暂时忽略电梯内要求下楼的按键输入或低楼层的召唤信号。这种确定电梯状态如何转换的信号称为“激励信号”。

诸如上例中的输入信号、输出信号、激励信号以及现态、次态及其状态转换的关系是时序电

路研究的主要内容。

6.1.1 时序逻辑电路的基本结构与分类

1. 时序电路的基本结构

时序电路的基本结构如图 6.1.1 所示, 它由完成逻辑运算的组合电路和起记忆作用的存储电路两部分构成, 其中, 存储电路由触发器或锁存器组成。为了方便, 图中各组逻辑变量均以向量形式表示, 其中, $I = (I_1, I_2, \dots, I_n)$ 为输入信号, $O = (O_1, O_2, \dots, O_j)$ 为输出信号, $E = (E_1, E_2, \dots, E_k)$ 为驱动存储电路转换为下一状态的激励信号, 而 $S = (S_1, S_2, \dots, S_m)$ 为存储电路状态, 称为状态信号, 亦称为状态变量, 它表示时序电路当前的状态, 简称现态。状态变量 S 被反馈到组合电路的输入端, 与输入信号 I 一起决定时序电路的输出信号 O , 并产生对存储电路的激励信号 E , 从而确定电路的下一状态, 即次态。于是, 上述 4 组变量间的逻辑关系可用下列三个向量函数形式的方程来表达:

$$E = f(I, S) \quad (6.1.1)$$

$$S^{n+1} = g(E, S^n) \quad (6.1.2)$$

$$O = h(I, S) \quad (6.1.3)$$

式(6.1.1)表达了激励信号与输入信号、状态变量的关系, 称为时序电路的激励方程。式(6.1.2)表达了存储电路从现态到次态的转换, 故称为状态转换方程, 简称转换方程。其中, S^n 和 S^{n+1} 分别表示存储电路的现态和次态。而式(6.1.3)表达了时序电路的输出信号与输入信号、状态变量的关系, 称为输出方程。上述三个向量函数形式的方程分别对应于表达时序电路的三个基本方程组: 激励方程组、转换方程组和输出方程组。

如上所述, 时序电路是状态依赖的, 故又称为状态机(State Machine, SM)。本章将只限于讨论有限数量的存储单元构成的状态机, 因而其状态数也是有限的, 称为有限状态机(Finite State Machine, FSM)。

由上述基本结构和方程可见, 时序电路具有以下主要特征:

① 时序电路由组合电路和存储电路组成。

② 时序电路的状态与时间因素相关, 即时序电路在任一时刻的状态变量不仅是输入信号的函数, 而且还是电路以前状态的函数, 并由当前输入变量和状态决定电路的下一状态。

③ 时序电路的输出信号由输入信号和电路状态共同决定。

2. 异步与同步时序电路

时序电路可分为异步时序电路和同步时序电路两大类。

若电路中触发器的时钟输入端没有连接在统一的时钟脉冲上, 或电路中没有时钟脉冲(如 SR 锁存器构成的时序电路), 从而, 电路中各存储单元的状态更新不是同时发生的, 则这种电路称为异步时序电路。根据电路是对脉冲边沿敏感还是对电平敏感, 异步时序电路又分为脉冲异

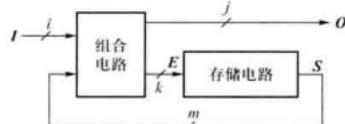


图 6.1.1 时序电路的基本结构

步时序电路(由触发器构成)和电平异步时序电路(由锁存器构成)两种。异步时序电路的状态转换取决于以任意时间间隔变化的输入信号序列,若有多个信号输出,其存储电路的状态转换因存在时间差异而可能造成短时间输出状态的不稳定,而且这种不稳定的状态有时是不容易判定的,常常给电路设计和调试带来困难。

与异步时序电路不同,同步时序电路中存储电路状态的转换是在同一时钟脉冲源的同一边沿作用下同步动作的,它也称作时钟同步状态机(Clocked Synchronous SM),其结构如图 6.1.2 所示。

同步时序电路的存储电路一般用触发器实现,所有触发器的时钟输入端都应接在同一个时钟脉冲源上,而且它们的时钟脉冲触发沿也都应一致。因此,所有触发器的状态更新是在同一时刻,其输出状态变换的时间差异很小。在时钟脉冲两次作用的间隔

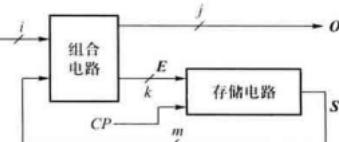


图 6.1.2 同步时序电路的结构

期间,从触发器输入到状态输出的通路被切断,即使此时输入信号发生变化,也不会改变各触发器的输出状态,所以较少发生因状态转换不同步而引起的输出状态不稳定的现象。更重要的是,其电路的状态很容易用固定周期的时钟脉冲边沿清楚地分离为序列步进。其中,每一个步进都可以通过输入信号和所有触发器的现态单独进行分析,从而有一套较系统、易掌握的分析和设计方法,电路行为也很容易用 HDL 来描述。所以,目前结构较复杂、运行速度较高的时序电路广泛采用同步方式来实现。很多大规模可编程逻辑器件的应用电路和专用集成电路的设计,也都采用同步时序电路的结构。

3. 米利型和穆尔型时序电路

米利(Mealy)型时序电路的结构如图 6.1.3 所示,事实上是将图 6.1.2 中的组合电路拆解成激励组合电路和输出组合电路两部分。米利型时序电路的输出信号 O 是状态变量 S 和输入信号 I 二者的函数,即 $O = h(I, S)$ 。这种时序电路在时钟脉冲的两个触发沿之间,输出信号随时可能受到非时钟同步的输入信号作用而发生变化,从而影响电路输出的同步性。

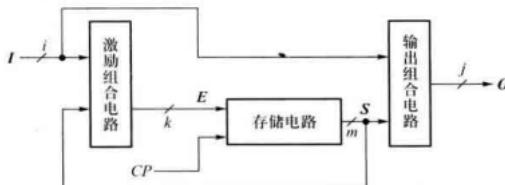


图 6.1.3 米利型时序电路的结构

穆尔(Moor)型时序电路是米利型时序电路的一种特例,它的输出信号 O 仅仅是状态变量 S 的函数,即 $O = h(S)$,其结构如图 6.1.4 所示。穆尔型时序电路的输出信号只取决于与时钟同步

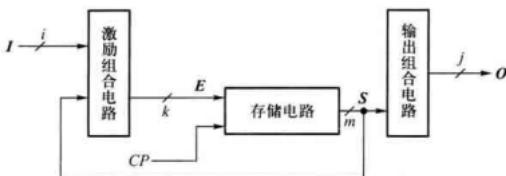


图 6.1.4 穆尔型时序电路的结构

的各触发器的状态，在时钟脉冲触发沿的间隔期间，不受非同步的输入信号影响。实际上，米利型时序电路中有时也有一个或多个输出可能是穆尔型的，即它们的输出只取决于触发器的状态。

在现代高速时序电路设计中，一般尽量采用穆尔型时序电路结构，以利于后续高速电路的同步。在米利型时序电路的输出端增加一级存储电路，构成“流水线输出”形式，是将其转化为穆尔型电路的最简单的方法，电路结构如图 6.1.5 所示。需要注意的是，流水线存储电路将把输出信号延迟一个时钟周期。虽然流水线输出电路增加了一些逻辑元件，但它的输出信号同步特性更好，具有更好的稳定性和抗干扰性能。在大规模集成电路技术成熟的今天，节省逻辑元件的数量经常不再是逻辑电路设计者追求的唯一目标，电路工作的稳定性和可靠性，以及工作速度的提高才是更重要的要求。

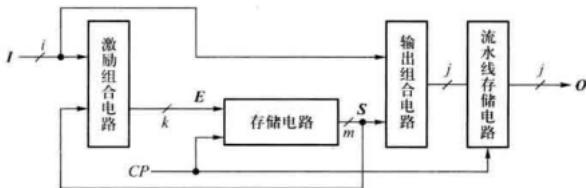


图 6.1.5 流水线输出的米利型时序电路的结构

6.1.2 时序逻辑电路功能的表达

时序电路的功能可用逻辑方程组、转换表、状态表、状态图和时序图等形式来表达，也可以用 HDL 语言描述。从理论上讲，有了激励方程组、转换方程组和输出方程组，时序电路的功能就被唯一地确定了。但是，对于许多时序电路而言，仅从这三组方程还不易判断其逻辑功能，在设计时序电路时，也往往很难根据给出的逻辑需求直接写出这三组方程。因此，还需要用能够直观反映电路状态变化序列全过程的转换表、状态表和状态图来帮助。三组方程、转换表、状态表和状态图之间可以实现相互转换，根据其中任意一种表达方式，都可以画出时序图。下面通过实例介绍时序电路逻辑功能的前五种表达方法，使用 Verilog HDL 描述时序电路的方法将在 6.7 节讨论。

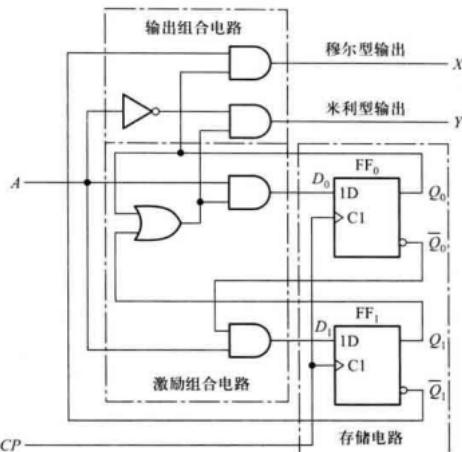


图 6.1.6 同步时序电路一例

考虑图 6.1.6 所示电路, 它由组合电路与存储电路两大部分组成。其中, 存储电路由两个 D 触发器 FF_1 、 FF_0 构成, 二者共用一个时钟信号 CP , 从而构成一个同步时序电路。组合电路又分为激励电路和输出电路两部分。电路的输入信号为 A , 输出信号为 X 、 Y 。对触发器的激励信号分别为 D_1 和 D_0 , Q_1 、 Q_0 为电路的状态变量。从图中可以看出, 输出信号 Y 是状态变量 Q_1 、 Q_0 和输入信号 A 的函数, 所以从总体上看, 这是一个米利型时序电路, 但是, 输出信号 X 纯粹由状态变量 Q_1 、 Q_0 决定, 电路中又存在一个穆尔型输出 X 。

1. 逻辑方程组

(1) 激励方程组

根据图 6.1.6 中的组合电路, 可写出对两个 D 触发器的激励方程组

$$D_0 = (Q_1 + Q_0)A \quad (6.1.4)$$

$$D_1 = \overline{Q_0}A \quad (6.1.5)$$

(2) 转换方程组

将式(6.1.4)和式(6.1.5)分别代入 D 触发器的特性方程 $Q^{n+1} = D$, 于是, 得到转换方程组

$$Q_0^{n+1} = (Q_1^n + Q_0^n)A \quad (6.1.6)$$

$$Q_1^{n+1} = \overline{Q_0^n}A \quad (6.1.7)$$

以上两式表明, 触发器的次态 Q_0^{n+1} 和 Q_1^{n+1} 是输入变量 A 和触发器现态 Q_0^n 、 Q_1^n 的函数。

(3) 输出方程组

图 6.1.6 的逻辑图中有两个输出变量 X, Y , 可根据输出组合电路得到输出方程组

$$X = \overline{Q_1} Q_0 \quad (6.1.8)$$

$$Y = (Q_1 + Q_0) \overline{A} \quad (6.1.9)$$

显然, X 是穆尔型输出, Y 是米利型输出。

上述三组方程中, 激励方程组和输出方程组表达了时序电路中全部组合电路的特性, 而转换方程组则表达了存储电路从现态到次态的状态转换特性。转换方程两边的状态变量, 分别以上标 n 表示现态, 上标 $n+1$ 表示次态, 以区别这两种不同的状态。

2. 转换表

与组合电路分析方法类似, 根据逻辑表达式(6.1.6)、(6.1.7)和式(6.1.8)、(6.1.9)可以列出真值表如表 6.1.1 所示。真值表的输入变量为 Q_1^n, Q_0^n 和 A , 输出变量为 Q_1^{n+1}, Q_0^{n+1} 和 X, Y 。由于该真值表反映了触发器从现态到次态的转换, 故称为状态转换真值表。一般来说, 有 m 位状态变量和 i 位输入信号, 就存在 2^{m+i} 种状态-输入组合, 真值表就应有 2^{m+i} 行。

表 6.1.1 图 6.1.6 电路的状态转换真值表

Q_1^n	Q_0^n	A	Q_1^{n+1}	Q_0^{n+1}	X	Y
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	0	0	1
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	1	0	1	0	0

在分析和设计时序电路时, 更常用的是转换表, 如表 6.1.2 所示。它与表 6.1.1 完全等效, 但形式更紧凑明了。表 6.1.2 用矩阵形式表达出在不同现态和输入条件下, 电路的状态转换和输出逻辑值。表中, 输出信号 X 是穆尔型输出, 故将其与现态 $Q_1^n Q_0^n$ 对应的逻辑值单列一栏; 输出信号 Y 的逻辑值不仅取决于 Q_1^n 和 Q_0^n , 而且会跟随 A 变化, 所以表示在 $A=0$ 和 $A=1$ 两栏的斜线后面。需要注意的是, 虽然它与斜线前的次态 $Q_1^{n+1} Q_0^{n+1}$ 列在一起, 但仍是现态 $Q_1^n Q_0^n$ 和输入 A 的函数。

表 6.1.2 图 6.1.6 电路的转换表

$Q_1^* Q_0^*$	$Q_1^{n+1} Q_0^{n+1} / Y$		X
	$A = 0$	$A = 1$	
00	00/0	10/0	0
01	00/1	01/0	1
10	00/1	11/0	0
11	00/1	01/0	0

3. 状态表

转换表以各触发器逻辑值的编码表示时序电路的状态。在分析一个电路时,给每个编码状态分别赋予一个具体名称有时更便于与实际问题结合进行分析和记忆;时序电路设计过程中,在尚未进行状态分配之前,也必须首先给各个状态命名,以表达状态之间的转换关系。在本例中,可以简单地命名 4 个状态为 **00** = a , **01** = b , **10** = c , **11** = d 。当然,也可以用意义更明显的中、英文单词或数字来对状态命名。将表 6.1.2 中 $Q_1 Q_0$ 的状态用其状态名代替,则得到表 6.1.3 所示的状态表。表中, S^n 表示现态, S^{n+1} 表示次态,斜线后的逻辑值为输出变量 Y 的值。

和转换表相比,使用状态表更容易理解时序电路的行为和结果。如果状态命名合理,即使较复杂的时序电路,直接通过状态名就可以得到各状态的实际意义和转换关系。但是,状态表中的状态变量没有标明编码,与时序电路的 3 个逻辑方程组及实际电路图难以联系,这方面的信息少于转换表。转换表和状态表虽然形式和内容相似,然而在应用上是有差别的。

表 6.1.3 图 6.1.6 电路的状态表

S^n	S^{n+1} / Y		X
	$A = 0$	$A = 1$	
a	$a/0$	$c/0$	0
b	$a/1$	$b/0$	1
c	$a/1$	$d/0$	0
d	$a/1$	$b/0$	0

4. 状态图

将表 6.1.3 的状态表转换为如图 6.1.7(a) 所示的状态图,可以更直观形象地表示出时序电路运行中的全部状态、各状态间相互转换的关系以及转换的条件和结果。图中,每一个圆圈都对应着一个状态,圆圈中标出状态名;每一个带箭头的方向线都表示一个转换,箭头指示出状态转换的方向。当方向线的起点和终点都在同一个圆圈上时,则表示状态不变。引起该状态转换的输入变量逻辑值标在方向线旁斜线左侧,如图中 A 值。米利型输出变量的逻辑值标在方向线旁

斜线右侧,是本次状态转换前的逻辑值,如图中 Y 值,它由方向线起点的状态和斜线前的输入变量共同决定。而穆尔型输出变量的逻辑值则标在圆圈内的状态名后,因为状态一旦确定,其输出值随之确定,如图中 X 值。当设计时序电路时,首先需要画出这种形式的状态图,以明确状态的数目、状态转换的方向以及状态转换的条件和相应的输出信号。

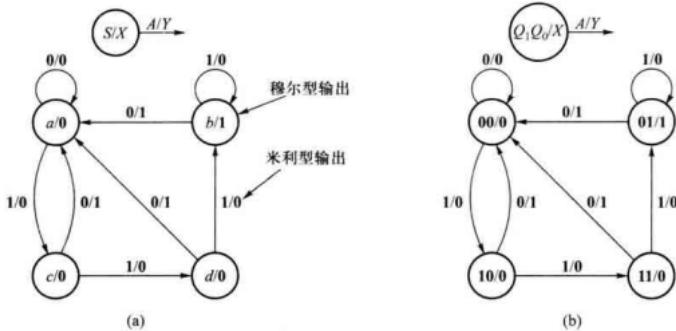


图 6.1.7 图 6.1.6 所示同步时序电路的状态图

(a) 与状态表对应的状态图 (b) 与转换表对应的状态图

状态图也可以与转换表相对应,如图 6.1.7(b) 所示,圆圈中以二进制编码表示状态,在分析时序电路时可先得到这种形式的状态图,然后再进一步研究其功能。

需要强调指出,图 6.1.7(a) 或 (b) 中状态转换的方向,取决于电路中下一个时钟脉冲触发沿到来前瞬间的输入信号。如果在此之前输入信号发生改变,则状态转换的方向也会立即改变。例如图 6.1.7(a) 中,当处于状态 c 时,如果输入 A 保持为 1 ,则输出 Y 为 0 ,下一状态将转换为 d ;若在下一个时钟脉冲触发沿到来之前, A 由 1 变化为 0 ,则 Y 立即变化为 1 ,下一状态将转换为 a 。

绘制状态图时需要注意不能漏画方向线。原则上,以某一状态为起点的方向线数量应为 2^i 根, i 为电路输入变量的数目,即输入变量的每一种组合应当对应 1 根方向线。例如,在图 6.1.7 的状态图中只有一个输入变量 A ,共有 2^1 种组合: $A=0$ 和 $A=1$,所以从每个状态都引出两根方向线。

当出现输入变量不同而方向线的起点和终点一致时,可以合并方向线。例如 5.5.2 小节中讨论的 JK 触发器状态图,它有两个输入变量 J, K ,以每个状态为起点的方向线原应当有 $2^2=4$ 根,经合并后仅余两根,如图 5.5.3 所示,从而使状态图得到简化。以图中触发器状态 $Q=0$ 为例,转换到状态 $Q=1$ 的方向线原有两根,输入变量的组合分别为 $JK=10$ 和 $JK=11$,即同一方向上 K 是无关变量,于是可将两根方向线合并为一根,输入变量表示为 $JK=1x$ 。其余方向线的合并类推,最后得到简化的状态图。

因为人类对图形的理解能力远高于相应的表格、文字和公式,所以状态图在日益复杂的时序

电路设计中愈显重要。从状态图出发,可以直接列出状态表,进而列出转换表。EDA 软件一般都可以以某种形式接受转换表的输入,有些软件甚至能以某种形式接受状态表或状态图的输入。这样,在 6.3 节所讨论的同步时序电路的设计,从状态图以后的很多工作都可以由软件自动完成,状态图的描绘才是一项真正具有挑战性的工作。

5. 时序图

与组合电路一样,波形图能直观地表达时序电路中各信号在时间上的对应关系,通常把时序电路的状态、输出对输入信号(包括时钟信号)响应的波形图称为时序图。它不仅便于电路调试时检查逻辑功能、排查故障或差错,而且在运用 HDL 设计电路时可用于电路的仿真。从逻辑方程组、转换表或状态图都可以导出时序图。图 6.1.6 所示时序电路的时序图如图 6.1.8 所示。

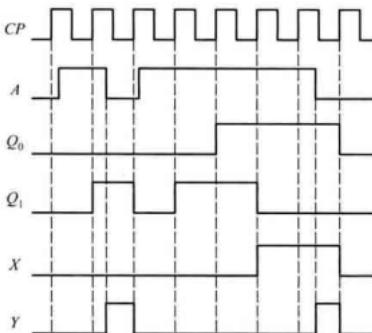


图 6.1.8 图 6.1.6 电路的时序图

使用时序图时需要注意,有时它并不完全表达出电路状态转换的全部过程,而是根据需要仅画出部分典型的波形图,如图 6.1.8 中就没有表达出当状态 Q_1Q_0 为 11 而输入 A 为 0 时的状态转换波形和输出波形。

复习思考题

6.1.1 时序电路由哪几部分组成? 它和组合电路在逻辑功能和结构上有什么区别?

6.1.2 异步时序电路与同步时序电路有哪些不同的特性?

6.1.3 表达时序电路的逻辑功能有哪几种方法? 哪种方法可直接从逻辑图得出? 哪几种方法可直接显示出时序电路状态转换的全部过程? 哪种方法可观察到电路的输入信号、触发器状态和输出信号的波形并在实验和 HDL 仿真时应用?

6.1.4 试从表 6.1.1 的状态转换真值表推导出该例的激励方程组、转换方程组和输出方程组。

6.1.5 试从图 6.1.7(a) 和 (b) 所示状态图分别推导其状态表和转换表。

6.1.6 米利型和穆尔型时序电路的输出特性有何不同？米利型输出和穆尔型输出在转换表、状态表和状态图中的表达有何不同？

6.2 同步时序逻辑电路的分析

同步时序电路的分析实际上是一个读图、识图的过程：按照给定的时序电路，通过分析其状态和输出信号在输入变量和时钟作用下的转换规律，理解其逻辑功能和工作特性。下面首先介绍分析同步时序电路的一般步骤，然后通过例题加深对分析方法的理解。

6.2.1 分析同步时序逻辑电路的一般步骤

(1) 根据给定的同步时序电路导出下列逻辑方程组：

- ① 对每个触发器导出激励方程，组成激励方程组；
- ② 将各触发器的激励方程代入相应触发器的特性方程，得到各触发器的转换方程，组成转换方程组；

③ 对应每个输出变量导出输出方程，组成输出方程组。

上述①和③推导出同步时序电路中全部组合电路的特性，而②则推导出电路的状态转换特性。

(2) 根据转换方程组和输出方程组，列出电路的转换表或状态表，画出状态图和时序图。

(3) 确定电路的逻辑功能，必要的话，可用文字详细描述。

上述步骤是分析同步时序电路的一般化过程，实际分析中可根据具体情况增、减执行，最终达到明确电路的功能即可。

6.2.2 同步时序逻辑电路分析举例

例 6.2.1 分析图 6.2.1 所示同步时序电路的逻辑功能。

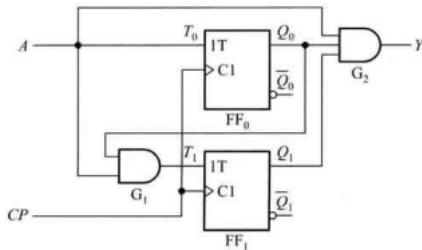


图 6.2.1 例 6.2.1 的逻辑电路图

解：这是一个由两个 T 触发器和两个与门组成的米利型同步时序电路，分析如下。

(1) 根据电路列出三个方程组

① 激励方程组

$$T_0 = A$$

$$T_1 = AQ_0$$

② 转换方程组

T 触发器的特性方程为 $Q^{n+1} = T \oplus Q^n = T \overline{Q^n} + \bar{T}Q^n$ ，将两个激励方程分别代入触发器的特性方程，即得转换方程组

$$Q_0^{n+1} = A \oplus Q_0^n$$

$$Q_1^{n+1} = (AQ_0^n) \oplus Q_1^n$$

③ 输出方程组

$$Y = AQ_1 Q_0$$

(2) 列出转换表

首先将电路可能出现的现态和输入变量在转换表中列出，本例中需要将 **00**, **01**, **10**, **11** 四个可能的现态列在“ $Q_1^n Q_0^n$ ”栏目中，并把输入 $A = 0$ 和 $A = 1$ 列在“ $Q_1^{n+1} Q_0^{n+1} / Y$ ”栏目下；然后将现态和输入逻辑值一一代入上述转换方程组和输出方程组，分别求出次态和输出逻辑值。例如，将 $Q_1^n = Q_0^n = A = 0$ 分别代入两个转换方程，得到 $Q_1^{n+1} = 0$ 和 $Q_0^{n+1} = 0$ ；将 $Q_1^n = Q_0^n = A = 0$ 代入输出方程，得到 $Y = AQ_1 Q_0 = 0$ 。于是可在转换表“ $Q_1^{n+1} Q_0^{n+1} / Y$ ”栏目下， $A = 0$ 这一列的第一行填入 **00/0**。其余以此类推，最后列出的转换表如表 6.2.1 所示。

表 6.2.1 例 6.2.1 的转换表

$Q_1^n Q_0^n$	$Q_1^{n+1} Q_0^{n+1} / Y$	
	$A = 0$	$A = 1$
0 0	0 0/0	0 1/0
0 1	0 1/0	1 0/0
1 0	1 0/0	1 1/0
1 1	1 1/0	0 0/1

(3) 画出状态图

由转换表即可画出状态图，如图 6.2.2 所示。

(4) 画出时序图

设电路的初始状态为 $Q_1 Q_0 = 00$ ，根据转换表和状态图，可画出在一系列 CP 脉冲作用下电路的时序图，如图 6.2.3 所示。

(5) 逻辑功能分析

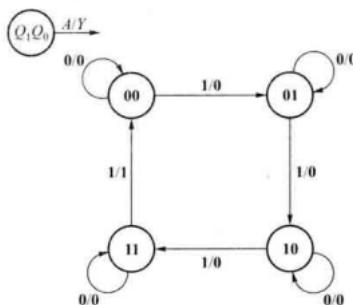


图 6.2.2 例 6.2.1 的状态图

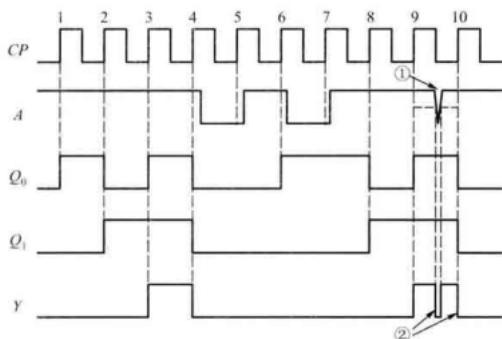


图 6.2.3 例 6.2.1 的时序图

观察状态图和时序图可知,图 6.2.1 的电路是一个由信号 A 控制的可控二进制计数器, CP 为计数脉冲。当 $A=0$ 时停止计数, 电路状态保持不变; 当 $A=1$ 时, 在 CP 上升沿到来后电路状态值加 1, 一旦计数到 **11** 状态, Y 输出 **1**, 且电路状态将在下一个 CP 上升沿回到 **00**。输出信号 Y 的下降沿可用于触发进位操作。

该电路亦可作为序列信号检测器, 用来检测同步脉冲信号序列 A 中 **1** 的个数, 一旦检测到四个 **1** 状态(这四个 **1** 状态可以不连续), 电路输出 Y 则出现一次从 **1** 到 **0** 的跳变。

观察图 6.2.3 的时序图, 在第 9 个和第 10 个 CP 脉冲之间, 输入信号 A 出现短时间的 **0** 电平“毛刺”, 如图中箭头①所示, 结果引起输出 Y 也相应变化。倘若信号 A 的这个“毛刺”是外界干扰造成的(输入信号的引线有时可能较长, 易捡拾干扰信号), 计数器将输出两次

进位触发脉冲沿,如图 6.2.3 中箭头②所示。如果删除图 6.2.1 中 A 和与门 G_2 输入之间的连线,将电路转化为穆尔型,则能使输出信号 Y 仅取决于电路的状态,其变化始终与时钟同步,而输入信号 A 影响电路状态的时间仅限于 CP 脉冲上升沿前后的瞬间,发生错误的概率将降低到较低的程度,从而提高了电路的抗干扰性能。当然,修改后的电路逻辑功能也将稍有变化。

例 6.2.2 分析图 6.2.4 所示同步时序电路。

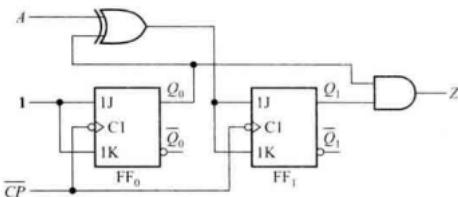


图 6.2.4 例 6.2.2 的逻辑电路图

解: 这是一个由两个下降沿触发的 JK 触发器、一个异或门及一个与门组成的穆尔型同步时序电路。

(1) 根据电路列出三个方程组

① 激励方程组

$$J_0 = K_0 = 1$$

$$J_1 = K_1 = A \oplus Q_0$$

② 转换方程组

将两个激励方程分别代入 JK 触发器的特性方程,得到两个触发器的转换方程

$$\begin{aligned} Q_0^{n+1} &= J_0 \overline{Q_0^n} + \overline{K_0} Q_0^n = \overline{Q_0^n} \\ Q_1^{n+1} &= J_1 \overline{Q_1^n} + \overline{K_1} Q_1^n \\ &= (A \oplus Q_0^n) \overline{Q_1^n} + A \oplus \overline{Q_0^n} Q_1^n \\ &= A \oplus Q_0^n \oplus Q_1^n \end{aligned}$$

③ 输出方程组

$$Z = Q_1 Q_0$$

(2) 列出转换表

根据转换方程组和输出方程组可以列出转换表如表 6.2.2 所示。

表 6.2.2 例 6.2.2 的转换表

$Q_1^* Q_0^*$	$Q_1^{n+1} Q_0^{n+1}$		Z
	$A = 0$	$A = 1$	
0 0	0 1	1 1	0
0 1	1 0	0 0	0
1 0	1 1	0 1	0
1 1	0 0	1 0	1

(3) 画出状态图

根据转换表可以画出状态图如图 6.2.5 所示。

(4) 画出时序图

设电路的初始状态为 $Q_1 Q_0 = 00$, 根据转换表和状态图, 可画出在一系列 \overline{CP} 脉冲作用下的时序图如图 6.2.6 所示。

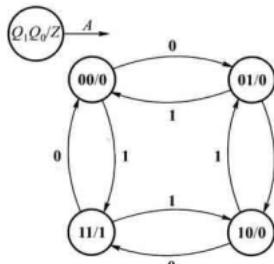


图 6.2.5 例 6.2.2 的状态图

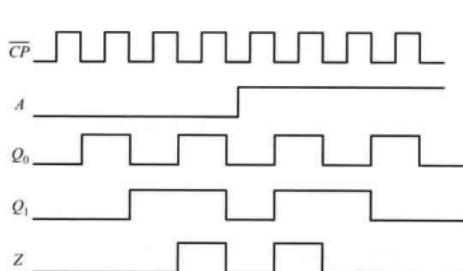


图 6.2.6 例 6.2.2 电路的时序图

(5) 逻辑功能分析

由状态图可以看出, 图 6.2.4 所示电路是一个可逆二进制计数器。当 $A = 0$ 时, 进行递增计数, 每来一个时钟脉冲, 计数器值 $Q_1 Q_0$ 加 1, 依次为 $00 \rightarrow 01 \rightarrow 10 \rightarrow 11$ 。每经过 4 个时钟脉冲作用后, 电路的状态循环一次。当 $A = 1$ 时, 进行递减计数, 依次为 $11 \rightarrow 10 \rightarrow 01 \rightarrow 00$ 。Z 端在 $Q_1 Q_0$ 为 11 时输出 1。在进行递增计数时, 可以利用 Z 信号的下降沿触发进位操作; 在递减计数时则可用 Z 信号的上升沿触发借位操作。有关计数器的详细内容将在 6.5.2 节讨论。

例 6.2.3 分析图 6.2.7 所示的同步时序电路。

解: 由图 6.2.7 可见, 该同步时序电路没有输入信号, 输出为三个触发器的状态, 是穆尔型时序电路。

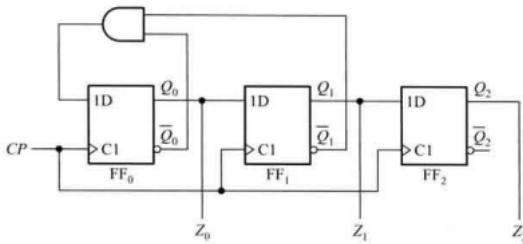


图 6.2.7 例 6.2.3 的逻辑电路图

(1) 根据电路列出逻辑方程组

由于使用 D 触发器，其特性方程为 $Q^{n+1} = D$ ，因此，可以根据逻辑电路直接列出转换方程组。

① 转换方程组

$$Q_0^{n+1} = D_0 = \overline{Q}_1^n \overline{Q}_0^n$$

$$Q_1^{n+1} = D_1 = Q_0^n$$

$$Q_2^{n+1} = D_2 = Q_1^n$$

② 输出方程组

$$Z_0 = Q_0$$

$$Z_1 = Q_1$$

$$Z_2 = Q_2$$

(2) 列出转换表

由于该电路的输出 Z_2, Z_1, Z_0 就是各触发器的状态，所以转换表中可不再单列输出栏。并且电路中没有输入信号，其转换表可简化为表 6.2.3 所示形式。

表 6.2.3 例 6.2.3 的转换表

$Q_2^n Q_1^n Q_0^n$	$Q_2^{n+1} Q_1^{n+1} Q_0^{n+1}$
0 0 0	0 0 1
0 0 1	0 1 0
0 1 0	1 0 0
0 1 1	1 1 0
1 0 0	0 0 1
1 0 1	0 1 0
1 1 0	1 0 0
1 1 1	1 1 0

(3) 画出状态图

根据转换表可画出电路的状态图如图 6.2.8 所示。由图可见,001、010、100 三个状态形成闭合回路,电路正常工作时,其状态总是按照回路中的箭头方向循环变化。这三个状态构成了有效序列,称它们为有效状态,其余的五个状态则称为无效状态。从状态图还可以看出,无论电路的初始状态如何,经过若干 CP 脉冲之后,总能进入有效状态。若电路能从无效状态经一定过程自动进入有效状态,则称其为具有自校正能力。因此,该电路是具有自校正能力的同步时序电路。

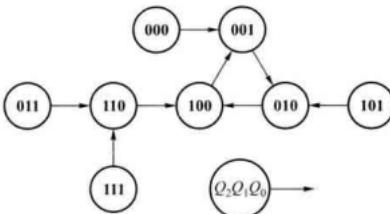


图 6.2.8 例 6.2.3 的状态图

(4) 画出时序图

设电路的初始状态为 $Q_2Q_1Q_0 = 000$, 根据转换表或状态图, 可画出时序图如图 6.2.9 所示。

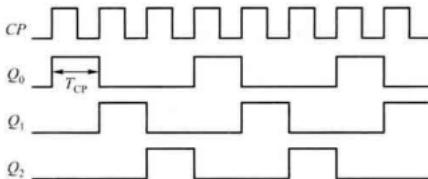


图 6.2.9 例 6.2.3 的时序图

(5) 逻辑功能分析

仅由转换表是不容易观察该电路逻辑功能的,而由状态图可见,电路的有效状态是三位循环码。从时序图可以看出,电路正常工作时,各触发器的 Q 端轮流出现一个脉冲信号,其宽度为一个 CP 周期,即 $1T_{cp}$,循环周期为 $3T_{cp}$,这个动作可以看作是在 CP 脉冲作用下,电路把宽度为 $1T_{cp}$ 的脉冲依次分配给 Q_0 、 Q_1 、 Q_2 各端,因此,电路的功能为脉冲分配器或节拍脉冲产生器。

复习思考题

6.2.1 同步时序电路的分析过程可分为哪几个步骤?

6.2.2 在分析同步时序电路时,激励方程组、转换方程组和输出方程组是怎样导出的?

6.2.3 怎样通过转换方程组和输出方程组得到转换表和状态表?进而如何导出状态图和时序图?

6.3 同步时序逻辑电路的设计

时序电路设计又称为时序电路综合,其任务是根据给定的逻辑功能需求,选择适当的逻辑器件,设计出符合要求的时序电路。本节讨论的用触发器及门电路设计同步时序电路的方法是时序电路设计的基础,也是用 Verilog HDL 对时序电路进行描述及使用可编程逻辑器件设计时序电路的基础。了解这些设计方法,亦有助于理解已有集成时序电路产品的内部结构和工作原理。

6.3.1 设计同步时序逻辑电路的一般步骤

设计同步时序电路的一般过程如图 6.3.1 所示。

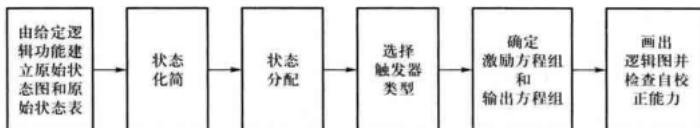


图 6.3.1 同步时序电路的设计过程

下面对设计过程中的主要步骤加以说明。

1. 由给定的逻辑功能建立原始状态图和原始状态表

通常,所要设计的时序电路的逻辑功能是通过文字、图形或波形图来描述的,首先必须把它们变成规范的状态图和状态表。这种直接从图文描述得到的状态图和状态表分别称为原始状态图和原始状态表。这个过程是对实际问题进行分析的过程,具体做法是:

(1) 明确电路的输入条件和相应的输出要求,分别确定输入变量和输出变量的数目和名称。同步时序电路的时钟脉冲 CP (或 \overline{CP})一般不作为输入变量考虑。

(2) 找出所有可能的状态以及状态转换之间的关系和输入条件。不同的状态以字符来命名。可以假定一个初始状态,以该状态作为现态,根据输入条件确定输出及次态。以此类推,直到把每一个状态的输出和向下一个可能转换的状态全部找出后,则建立起原始状态图。

(3) 根据原始状态图建立原始状态表。

由于以后所有的设计步骤都将在原始状态图或原始状态表的基础上进行,只有在它们全面、正确反映给定设计要求的条件下,才有可能获得成功的设计结果。

2. 状态化简

原始状态图或原始状态表很可能隐含多余的状态,去除多余状态的过程称为状态化简,其目

的是减少电路中触发器及门电路的数量,但不能改变原始状态图或原始状态表所表达的全部逻辑功能。状态化简建立在等价状态的基础上:如果两个状态分别作为现态,其任何相同输入所建立的次态及产生的输出均完全相同,则这两个状态称为等价状态。凡是两个等价状态都可以合并成一个状态而不改变输入-输出关系。我们将通过 6.3.2 节的实例具体说明。

3. 状态分配

对每个状态指定一个特定的二进制代码,称为状态分配或状态编码。编码方案不同,设计出的电路结构也就不同。编码方案选择得当,设计出的电路可能工作更可靠,也可能结构相对简单。

首先,要确定状态编码的位数。同步时序电路的状态取决于触发器的状态组合,触发器的个数 N 即状态编码的位数。 N 与所要求的状态数 M 须满足如下关系

$$M \leq 2^N \quad (6.3.1)$$

其次,要对每个状态确定编码。从 2^N 个状态中取 M 个状态组合可能存在多种不同方案,随着 N 值的增大,编码方案的数目会急剧增多^①,面对大量的编码方案是难以一一进行仔细比较的。一般来说,选取的编码方案应该有利于所选触发器的激励方程及输出方程的化简以及电路的稳定可靠。有时,遵循状态变化的顺序,以二进制数自然递增顺序编码可简化电路。而使用具有一定特征的编码,如格雷码,则有利于降低输出信号产生竞争-冒险的可能性。

状态分配完成,则可将状态表中的状态名替换为状态编码,得到转换表。

4. 选择触发器类型

触发器类型选择的余地实际上是非常小的。小规模集成电路的触发器产品,大多是 D 触发器和 JK 触发器,选择具有较强逻辑功能的 JK 触发器,有时可简化激励电路。如前所述,很多可编程逻辑器件中采用 D 触发器来实现时序电路设计,如有特殊要求,用 D 触发器也非常容易构成其他逻辑功能的触发器。

5. 确定激励方程组和输出方程组

根据转换表,用卡诺图或其他方式对逻辑函数进行化简,可求得电路的激励方程组和输出方程组。这两个方程组决定了同步时序电路的组合电路部分。

6. 画出逻辑图,并检查自校正能力

按照前一步导出的激励方程组和输出方程组,可画出接近工程实现的逻辑电路图。

有些同步时序电路设计中会出现没有用到的无效状态,当电路上电后有可能陷入这些无效状态而不能退出,因此,设计的最后一步应检查电路是否能进入有效状态,即是否具有自校正能力。如果不能自校正,则需修改设计。

有些时序电路要求必须从指定的初始状态开始工作,而不允许从任何其他状态启动。这时,应利用触发器的直接置 0、置 1 功能,在开始工作之前先将电路置为有效状态。图 6.3.2(a) 所示

^① 研究证明,从 N 位编码中取 M 个状态,其可能的状态分配方案数目为 $S = \frac{2^N}{(2^N - M)!}$ 。例如, $N=3, M=5$,其可能的编码方案总数达 6 720 之多!

为一低电平手动复位信号产生电路,当按键开关按下,则产生低电平复位信号 \overline{RESET} ,按键抬起,系统则进入正常工作状态。图 6.3.2(b)是一种上电自动复位电路,它利用电容两端电压不能突变的原理,在上电之初将复位信号 \overline{RESET} 保持在低电平,将系统预置为初始状态。随着电源 $+V_{CC}$ 通过电阻 R 对电容 C 充电,使 C 两端电压逐渐升高,经过一段时间之后, \overline{RESET} 信号才跨越逻辑阈值转变为高电平,使系统脱离复位状态而进入正常工作状态。虽然设计了复位电路的时序电路在正常工作时可不考虑自校正问题,但是,如果外界干扰等偶发因素使时序电路进入无效的循环状态,则可能出现所谓的“死机”现象,这是逻辑电路设计者不得不预先考虑的问题。

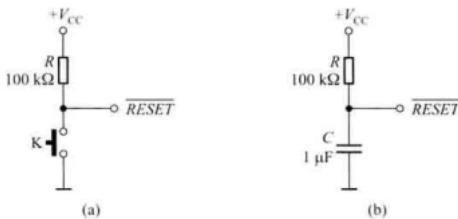


图 6.3.2 复位电路
(a) 手动复位电路 (b) 上电自动复位电路

需要说明的是,上述步骤是设计同步时序电路的一般化过程,实际设计中并不是每一步都必须执行,可根据具体情况简化或省略一些步骤,这通过下面的设计实例可以看出。

6.3.2 同步时序逻辑电路设计举例

例 6.3.1 用 D 触发器设计一个同步递增 8421 BCD 计数器。

解:计数器实际上是对时钟脉冲进行计数,每到来一个时钟脉冲触发沿,计数器改变一次状态。在每个时钟脉冲作用下,计数器的状态输出编码值加 1,编码顺序与 8421 BCD 码一致。每经过 10 个时钟脉冲,计数器完成一个计数周期,共有 10 个状态,可用 0~9 十个数字命名各个状态。由于电路的状态数、状态转换关系及状态编码等都是明确的,因此设计过程较简单,没有必要拘泥于上一小节所述的设计步骤。

(1) 列出转换表

10 个状态的计数器共需要 4 个 D 触发器构成。由于 D 触发器功能较简单,可将激励信号同时列入转换表中,如表 6.3.1 所示。这是一个穆尔型时序电路,输出变量即状态编码。

表 6.3.1 同步 8421 BCD 计数器的转换表

Q_3^n	Q_2^n	Q_1^n	Q_0^n	$Q_3^{n+1}(D_3)$	$Q_2^{n+1}(D_2)$	$Q_1^{n+1}(D_1)$	$Q_0^{n+1}(D_0)$
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	0	0	1	0	1
0	1	0	1	0	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	0	1
1	0	0	1	0	0	0	0

(2) 确定激励方程组

按表 6.3.1 可画出各触发器激励信号的卡诺图如图 6.3.3 所示。

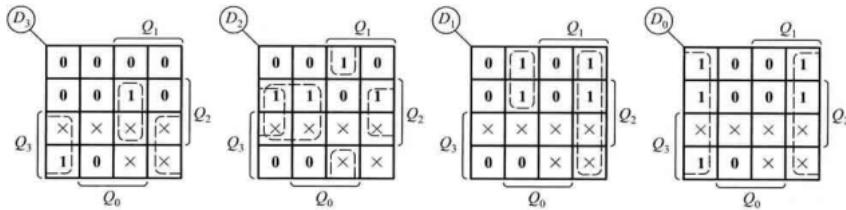


图 6.3.3 例 6.3.1 的卡诺图

4 个触发器可组合 16 个状态(0000 ~ 1111)，其中有 6 个状态(1010 ~ 1111)在 8421 BCD 计数器中是无效状态，在图 6.3.3 的卡诺图中以无关项 X 表示。于是，得到激励方程组(在本例中也是转换方程组)

$$Q_3^{n+1} = D_3 = Q_3^n \overline{Q_0^n} + Q_2^n Q_1^n Q_0^n$$

$$Q_2^{n+1} = D_2 = Q_2^n \overline{Q_1^n} + Q_2^n \overline{Q_0^n} + \overline{Q_2^n} Q_1^n Q_0^n$$

$$Q_1^{n+1} = D_1 = Q_1^n \overline{Q_0^n} + Q_3^n \overline{Q_1^n} Q_0^n$$

$$Q_0^{n+1} = D_0 = \overline{Q_0^n}$$

(3) 画出逻辑图

根据激励方程组可画出逻辑图如图 6.3.4 所示。图中,各触发器的直接置 0 端为低电平有效,计数工作时,电路的 \overline{RESET} 输入端应保持为高电平。

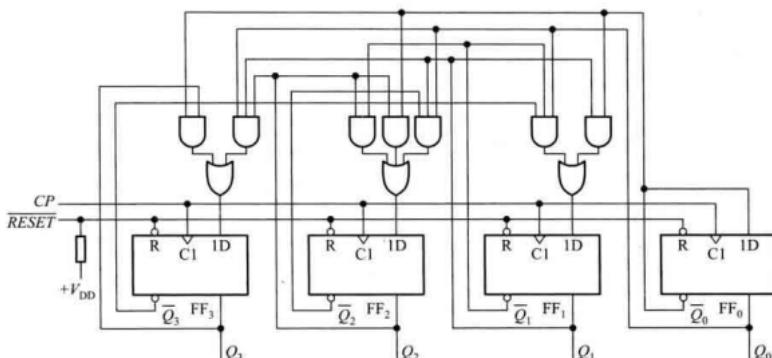


图 6.3.4 例 6.3.1 的逻辑电路

(4) 检查自校正能力

该电路有 6 个无效状态: **1010, 1011, 1100, 1101, 1110** 和 **1111**, 分别以它们作为现态, 代入电路的转换方程组而求其次态。如果还没有进入有效状态, 再以新的状态作为现态求下一个次态, 以此类推, 看最终能否进入有效状态。结果证明, 这 6 个状态在一或两个时钟周期后全部都能进入有效循环状态。

(5) 画出状态图

图 6.3.4 所示电路的完全状态图如图 6.3.5 所示。

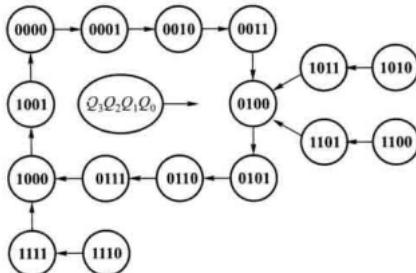


图 6.3.5 图 6.3.4 所示电路的完全状态图

如果要求电路必须从 **0000** 开始计数, 则可将前述复位电路连接在 **RESET** 输入端。在开始计数前使 **RESET** 产生低电平脉冲, 强制 4 个触发器进入 **0000** 的初始状态, 待 **RESET = 1** 后再开始计数。

例 6.3.2 试设计一序列编码检测器, 当检测到输入信号出现 **110** 序列编码(按自左至右的顺序)时, 电路输出为 **1**, 否则输出为 **0**。要求用同步时序电路实现。

解: (1) 由给定的逻辑功能建立原始状态图和原始状态表

从给定的逻辑功能可知, 电路有一个输入信号 **A** 和一个输出信号 **Y**, 电路功能是对输入信号 **A** 的编码序列进行检测, 一旦检测到信号 **A** 出现连续编码为 **110** 序列时, 输出为 **1**, 检测到其他编码序列, 则输出均为 **0**。

设电路的初始状态为 **a**, 如图 6.3.6 中大箭头所指。在此状态下, 电路输出 **Y = 0**, 这时可能的输入有 **A = 0** 和 **A = 1** 两种情况。当 **CP** 脉冲相应边沿到来时, 若 **A = 0**, 则是收到 **0**, 应保持在状态 **a** 不变; 若 **A = 1**, 则转向状态 **b**, 表示电路收到一个 **1**。当在状态 **b** 时, 若输入 **A = 0**, 则表明连续输入编码为 **10**, 不是 **110**, 则应回到初始状态 **a**, 重新开始检测; 若 **A = 1**, 则进入状态 **c**, 表示已连续收到两个 **1**。在状态 **c** 时, 若 **A = 0**, 表明已收到序列编码 **110**, 则输出 **Y = 1**, 并进入状态 **d**; 若 **A = 1**, 则收到的编码为 **111**, 应保持在状态 **c** 不变, 看下一个编码输入是否为 **A = 0**; 由于尚未收到最后的 **0**, 故输出仍为 **0**。在状态 **d**, 若输入 **A = 0**, 则应回到状态 **a**, 重新开始检测; 若 **A = 1**, 电路应转向状态 **b**, 表示在收到 **110** 之后又重新收到一个 **1**, 已进入下一轮检测; 在 **d** 状态下, 无论 **A** 为何值, 输出 **Y** 均为 **0**。根据上述分析, 可以得出如图 6.3.6 所示的原始状态图和表 6.3.2 的原始状态表。

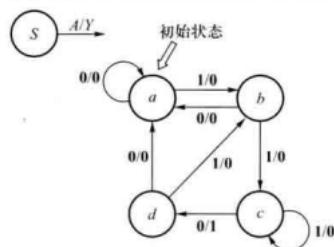


图 6.3.6 例 6.3.2 的原始状态图

表 6.3.2 例 6.3.2 的原始状态表

S^n	S^{n+1}/Y	
	$A = 0$	$A = 1$
a	a/0	b/0
b	a/0	c/0
c	d/1	c/0
d	a/0	b/0

(2) 状态化简

观察表 6.3.2 中 **a** 和 **d** 两行可以看出, 在 **A = 0** 和 **A = 1** 时, 分别具有相同的次态 **a**、**b** 及相同的输出 **0**, 因此, **a** 和 **d** 是等价状态, 可以合并。这里选择去除 **d** 状态, 并将其他行中的次态 **d** 改为 **a**。于是, 得到化简后的状态表如表 6.3.3 所示, 状态图亦可作相应化简。从实际物理意义看

也不难理解这种化简：当进入 c 状态后，电路已连续接收到两个 1 ，这时输入若为 0 ，则意味着已接收到编码 **110**，下一步电路应回到初始状态 a ，以准备新一轮检测，原始状态表中的 d 状态显然是多余的。

表 6.3.3 例 6.3.2 经化简的状态表

S^n	S^{n+1}/Y	
	$A = 0$	$A = 1$
a	$a/0$	$b/0$
b	$a/0$	$c/0$
c	$a/1$	$c/0$

(3) 状态分配

化简后的状态有三个，可以用两位二进制代码组合(**00**, **01**, **10**, **11**)中的任意三个代码表示，用两个触发器组成电路。观察表 6.3.3，当输入信号 $A = 1$ 时，有 $a \rightarrow b \rightarrow c$ 的变化顺序，当 $A = 0$ 时，又存在 $c \rightarrow a$ 的变化。综合两方面考虑，这里采取 **00** → **01** → **11** → **00** 的变化顺序，可能会使其中的组合电路相对简单。于是，令 $a = 00$, $b = 01$, $c = 11$ ，状态分配后得到的状态图如图 6.3.7 所示。

(4) 选择触发器类型

选用逻辑功能较强的 JK 触发器可能得到较简化的组合电路。

(5) 确定激励方程组和输出方程组

用 JK 触发器设计时序电路时，电路的激励方程需要间接导出。表 5.5.2 提供了 JK 触发器在不同现态和输入条件下所对应的次态。而在时序电路设计时，状态表已列出现态到次态的转换关系，希望推导出触发器的激励条件。所以需将特性表做适当变换，以给定的状态转换为条件，列出所需要的输入信号。这样的表格称为激励表。根据表 5.5.2 建立的 JK 触发器激励表如表 6.3.4 所示。表中的 \times 表示其逻辑值与该行的状态转换无关。

表 6.3.4 JK 触发器的激励表

Q^n	Q^{n+1}	J	K
0	0	0	\times
0	1	1	\times
1	0	\times	1
1	1	\times	0

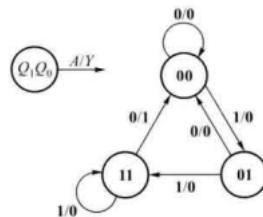


图 6.3.7 例 6.3.2 状态分配后的状态图

根据图 6.3.7 和表 6.3.4 可以列出状态转换真值表及两个触发器所要求的激励信号如表

6.3.5 所示。据此,分别画出两个触发器的输入 J, K 和电路输出 Y 的卡诺图,如图 6.3.8 所示。图中,不使用的状态均以无关项 \times 填入。化简后得到激励方程组

$$J_1 = A Q_0 \quad K_1 = \bar{A}$$

$$J_0 = \bar{A} \quad K_0 = \bar{A}$$

和输出方程

$$Y = \bar{A} Q_1$$

表 6.3.5 例 6.3.2 的状态转换真值表及激励信号

Q_1^n	Q_0^n	A	Q_1^{n+1}	Q_0^{n+1}	Y	激励信号			
						J_1	K_1	J_0	K_0
0	0	0	0	0	0	0	\times	0	\times
0	0	1	0	1	0	0	\times	1	\times
0	1	0	0	0	0	0	\times	\times	1
0	1	1	1	1	0	1	\times	\times	0
1	1	0	0	0	1	\times	1	\times	1
1	1	1	1	1	0	\times	0	\times	0

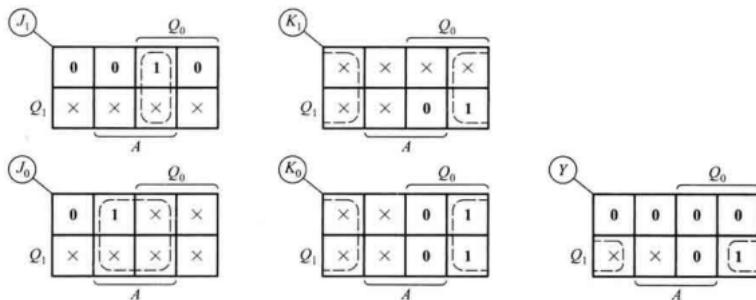


图 6.3.8 激励信号及输出信号的卡诺图

(6) 画出逻辑图

根据激励方程组和输出方程画出逻辑图,如图 6.3.9 所示。

(7) 检查自校正能力

最后还应检查该电路的自校正能力。当电路进入无效状态 **10** 后,由激励方程组和输出方程可知,若 $A=0$,则次态为 **00**;若 $A=1$,则次态为 **11**,电路能自动进入有效序列。但从输出来看,若

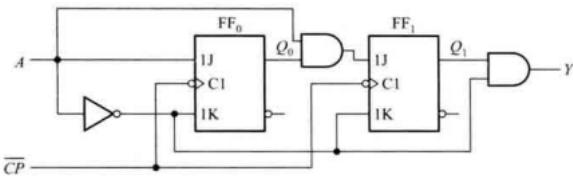


图 6.3.9 例 6.3.2 的逻辑图

电路在无效状态 **10**, 当 $A=0$ 时, 输出错误地出现 $Y=1$ 。为此, 需要对输出方程做适当修改, 即将图 6.3.8 中输出信号 Y 的卡诺图里无关项 $Q_1\bar{Q}_0\bar{A}$ 不画在包围圈内, 则输出方程变为 $Y=Q_1Q_0\bar{A}$ 。根据此式对图 6.3.9 也做相应的修改即可。

如果发现所设计的电路不能自校正, 则应修改设计。方法是: 在激励信号卡诺图的包围圈中, 对无关项 \times 的处理作适当修改, 即原来取 **1** 圈入包围圈的, 可试取 **0** 而不圈入包围圈, 与上述对输出 Y 的处理方法类似。于是, 得到新的激励方程组和逻辑图, 然后再检查其自校正能力, 直到能自校正为止。

例 6.3.3 试用 D 触发器设计一同步时序电路, 实现图 6.3.10 所示原始状态图的逻辑功能。

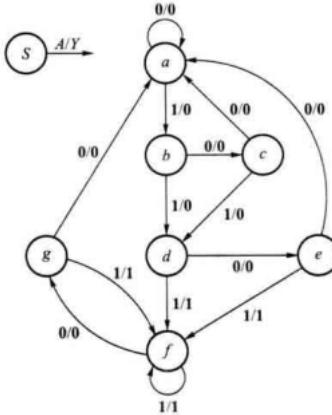


图 6.3.10 例 6.3.3 的原始状态图

解: (1) 列出原始状态表

根据图 6.3.10 列出原始状态表如表 6.3.6 所示。

表 6.3.6 例 6.3.3 的原始状态表

S^n	S^{n+1}/Y	
	$A = 0$	$A = 1$
a	$a/0$	$b/0$
b	$c/0$	$d/0$
c	$a/0$	$d/0$
d	$e/0$	$f/1$
e	$a/0$	$f/1$
f	$g/0$	$f/1$
g	$a/0$	$f/1$

(2) 状态化简

观察表 6.3.6 发现, 状态 e, g 是等价状态, 可以合并。表 6.3.7 示出第一步化简的结果: 将状态 g 一行删除, 并用状态 e 替换表 6.3.6 中“ S^{n+1}/Y ”栏中的状态 g 。

表 6.3.7 表 6.3.6 的第一步化简

S^n	S^{n+1}/Y	
	$A = 0$	$A = 1$
a	$a/0$	$b/0$
b	$c/0$	$d/0$
c	$a/0$	$d/0$
d	$e/0$	$f/1$
e	$a/0$	$f/1$
f	$e/0$	$f/1$

再观察表 6.3.7, 又出现状态 d 和 f 是等价的, 状态 f 亦可去除, 代之以 d 。于是, 得到表 6.3.8 所示状态表。检查该表, 已不存在等价状态, 因此是最简状态表。根据表 6.3.8 画出的状态图如图 6.3.11 所示。

表 6.3.8 例 6.3.3 的最简状态表

S^n	S^{n+1}/Y	
	$A = 0$	$A = 1$
a	$a/0$	$b/0$
b	$c/0$	$d/0$
c	$a/0$	$d/0$
d	$c/0$	$d/1$
e	$a/0$	$d/1$

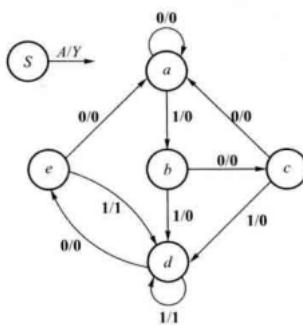


图 6.3.11 化简后的状态图

上述状态化简过程将原有的 7 个状态化简为 5 个,而输入-输出特性与原始状态表和原始状态图完全相同。

(3) 状态分配

表 6.3.8 中列出 5 个状态,最简单的状态分配是使用自然二进制码,取二进制计数序列的前 5 个连续编码,如表 6.3.9 中的状态分配方案 1。结合图 6.3.11 可以看出,这种二进制编码值的递增顺序基本上与相应的状态转换顺序一致。对于较简单的时序电路,按这种状态分配方案构成的电路,其组合电路将可能相对简单一些。

表 6.3.9 例 6.3.3 的三种状态分配方案

状态	方案 1 自然二进制码	方案 2 格雷码	方案 3 “一对一”
a	0 0 0	0 0 0	0 0 0 0 1
b	0 0 1	0 0 1	0 0 0 1 0
c	0 1 0	0 1 1	0 0 1 0 0
d	0 1 1	0 1 0	0 1 0 0 0
e	1 0 0	1 1 0	1 0 0 0 0

表 6.3.9 中所列方案 2 为格雷码方案。如果状态图示出的状态转换顺序是简单的从 a 到 e，那么它从一个状态转换到下一状态仅有一个触发器改变状态，使用这种方案将降低电路发生竞争-冒险的可能，提高电路的可靠性。但遗憾的是，本例的状态转换并不是简单的单循环序列。

表 6.3.9 中的方案 3 为“一对一”编码方案^①。虽然这将使用较多的触发器，但它的编码方式非常简单，可有效地简化组合电路，并换得工作可靠性和工作速度的提高。在大规模可编程逻辑器件，如 FPGA 中，触发器数量较多而门逻辑相对较少，“一对一”的编码方案有时反而更有利提高器件资源的利用率。

本例按表 6.3.9 中方案 1 分配状态，得到的转换表如表 6.3.10 所示。

表 6.3.10 例 6.3.3 的转换表

$Q_2^n Q_1^n Q_0^n$	$Q_2^{n+1} Q_1^{n+1} Q_0^{n+1} / Y$	
	$A = 0$	$A = 1$
0 0 0	0 0 0 / 0	0 0 1 / 0
0 0 1	0 1 0 / 0	0 1 1 / 0
0 1 0	0 0 0 / 0	0 1 1 / 0
0 1 1	1 0 0 / 0	0 1 1 / 1
1 0 0	0 0 0 / 0	0 1 1 / 1

(4) 确定激励方程组和输出方程组

该电路需要用 3 个 D 触发器实现，它们的输出分别为 Q_2 、 Q_1 、 Q_0 。于是，由表 6.3.10 可得状态转换真值表如表 6.3.11 所示。

由于 D 触发器的特性方程为 $Q^{n+1} = D$ ，所以根据表 6.3.11，可以画出 D_2 、 D_1 、 D_0 和 Y 的卡诺图如图 6.3.12 所示，未应用的状态可作为无关项×填入。

^① 有些国外文献称之为“One-Hot Encoding”，国内某些文献直译为“1 位热码编码”。

表 6.3.11 例 6.3.3 的状态转换真值表

Q_2^n	Q_1^n	Q_0^n	A	$Q_2^{n+1}(D_2)$	$Q_1^{n+1}(D_1)$	$Q_0^{n+1}(D_0)$	Y
0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	0
0	0	1	0	0	1	0	0
0	0	1	1	0	1	1	0
0	1	0	0	0	0	0	0
0	1	0	1	0	1	1	0
0	1	1	0	1	0	0	0
0	1	1	1	0	1	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	1	1	1

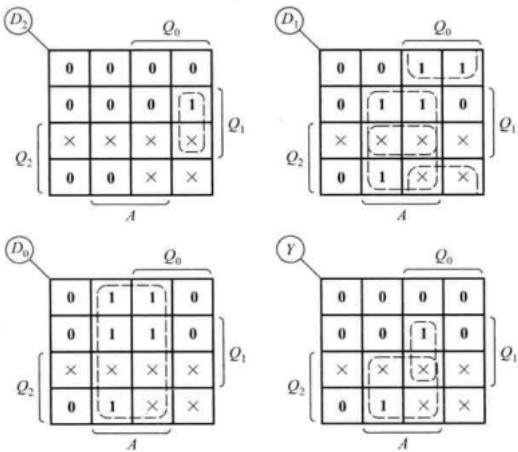


图 6.3.12 表 6.3.11 的卡诺图

经卡诺图化简, 得到三个触发器的激励方程组

$$D_2 = Q_1^n Q_0^n \bar{A}$$

$$D_1 = \overline{Q_1} Q_0 + Q_1 \overline{A} + Q_2 A$$

$$D_0 = A$$

和输出方程

$$Y = Q_1 Q_0 A + Q_2 A$$

经过几次实践之后,实际上可不必列状态转换真值表而直接从转换表得到卡诺图。

(5) 画出逻辑图

根据激励方程组和输出方程,可以画出图 6.3.13 所示的逻辑图。电路中引出了直接复位端 *RESET*,可外接复位电路,强制电路在开启上电时从状态 **000** 开始工作。

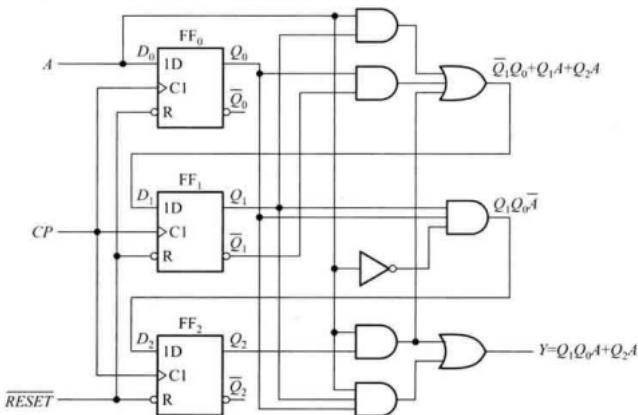


图 6.3.13 例 6.3.3 的逻辑图

(6) 检查自校正能力

该电路有 3 个无效状态:**101**、**110** 和 **111**,需要评估电路受到外界干扰时万一落入这 3 个无效状态时的影响。将 **101**、**110** 和 **111** 分别作为现态,与可能出现的所有输入信号一起分别代入电路的转换方程组而求其次态。结果证明,无论在何种情况下,这 3 个状态在一个时钟周期后全部都能进入有效状态。于是,可进一步画出完全状态图如图 6.3.14 所示。

6.3.3 同步时序逻辑电路中的时钟偏移

一般的时序电路都含有多个触发器,特别是规模较大的时序电路(如 PLD 中),其触发器的数量将会更多。同步时序电路要求时钟脉冲必须同时连接到每个触发器的时钟输入端。当电路规模较大时,由于实际电路制版时触发器的分布位置和布线等限制,每个触发器的时钟信号线的长度可能存在较大差异。另外,当触发器数量较多时,由于公共时钟信号驱动电路的负载能力有

限(即扇出的限制),经常需要在某些位置添加缓冲器来提高时钟信号线的驱动能力,由此造成时钟脉冲传输到不同触发器的延迟时间不同。这种从同一时钟源出发的时钟脉冲,通过不同路径到达每个触发器的时间不同而产生的偏差称为时钟偏移。随着工艺水平的提高,触发器的传输延迟时间大大缩短(为 $1\sim2\text{ ns}$,参见表5.4.2),很有可能出现小于时钟偏移的情况,此时,将导致同步时序电路出现错误动作。

例如某同步时序逻辑电路中的2个触发器电路如图6.3.15(a)所示,触发器 FF_0 的输出 Q_0 作为触发器 FF_1 的输入,且假设它们的连线很短,可以忽略其延迟。另外,假设受元器件位置及布线等的限制,由 CP 到 FF_1 的时钟 CP_1 的传输路径很

长,有较大延迟,即时钟偏移较大。在不考虑时钟偏移时,电路按照设计初衷正常工作,第1个 CP 脉冲有效沿到来时,输入信号 A 的状态存入 FF_0 。第2个 CP 脉冲到来时, A 的状态才传递到 FF_1 ,其工作波形应如图6.3.15(b)所示(设触发器初态 $Q_1 Q_0 = 00$)。但是,实际电路存在时钟偏移,如果时钟偏移量超过触发器 FF_0 的传输延迟时间 t_{pd} ,即 FF_1 的时钟 CP_1 在 Q_0 翻转后才到

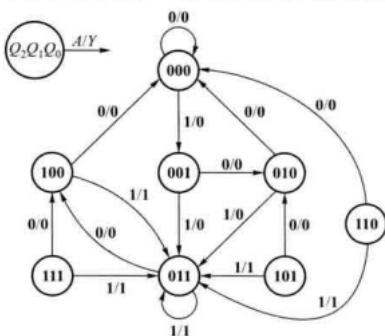


图 6.3.14 例 6.3.3 的完全状态图

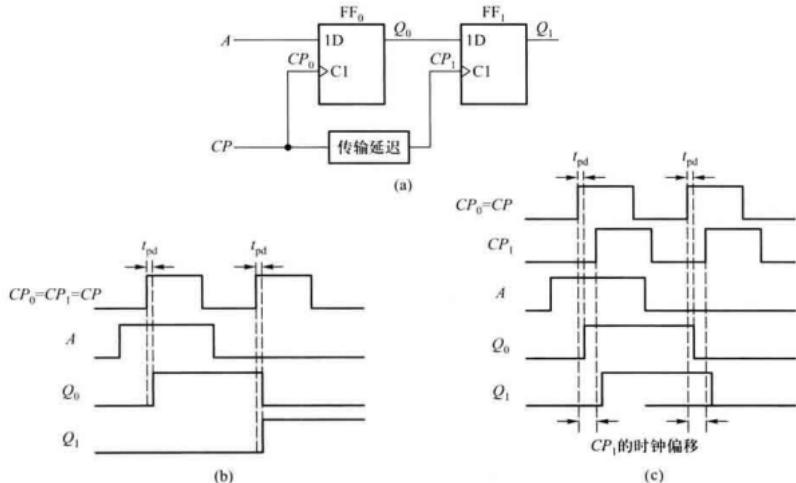


图 6.3.15 某同步时序逻辑电路工作情况

(a) 逻辑电路 (b) 无时钟偏移时的时序图 (c) 由时钟偏移引起的错误时序图

达,如图 6.3.15(c)所示,则 Q_1 的波形将与正常工作时(图 6.3.15(b))不同,即 A 的状态在第 1 个 CP 脉冲到来时就已经传递到了 FF_1 ,出现了逻辑错误。

由此看出,严重的时钟偏移很有可能造成同步时序电路的工作错误。为了消除 6.3.15(a) 中时钟偏移带来的影响,一方面尽可能缩短 CP_1 的传输路径,减小其延迟;另一方面,也可以在 CP_0 的传输路径上添加缓冲器,延长其延迟,从而减小两触发器的时钟偏移。

另外,不同的时钟传输路径上的负载(特别是容性负载)不平衡时,将导致不同路径上时钟脉冲的上升沿和下降沿的斜率不同,也会引起时钟偏移。因此,设计时钟布线网络时,还需要考虑各传输路径上负载的平衡问题。

通过以上分析,引起时钟偏移的主要原因可归纳为:

- (1) 各触发器时钟传输路径的长度不同;
- (2) 各触发器时钟传输路径上经过的缓冲器的数量不同;
- (3) 各触发器时钟传输路径上的负载不平衡。

在设计大规模的同步时序逻辑电路时,为了减少时钟偏移对电路正常工作的影响,应尽可能减小各触发器的时钟偏移。图 6.3.16 是一种可以有效减小时钟偏移的 H 型时钟布线网络。该方案的设计思路是使同步时钟信号传输到每个触发器 CP 端的路径尽可能相同。为简洁起见,图中未画出完整的触发器,而仅以 FF_i 替代。考虑到系统时钟 CP 扇出数的限制,网络中添加了 4 个缓冲器,每个缓冲器驱动 8 个触发器负载。当然,在实际电路设计时,可根据触发器 CP 端的负载效应和缓冲器的驱动能力来安排缓冲器的位置和数量。

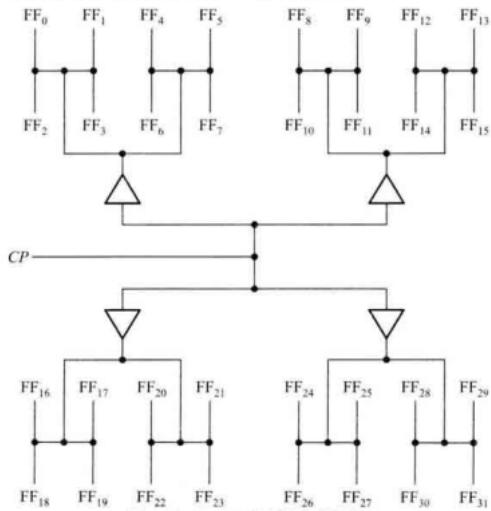


图 6.3.16 H 型时钟布线网络

复习思考题

- 6.3.1 同步时序电路的设计过程可分为哪几个步骤?
- 6.3.2 什么是原始状态图和原始状态表?怎样建立原始状态图和原始状态表?
- 6.3.3 什么是等价状态?若状态 a 与 b 等价, b 又与 c 等价,那么状态 a 与 c 等价吗?
- 6.3.4 同步时序电路中触发器的数目与状态数目有何关系?与状态分配又有何关系?
- 6.3.5 激励方程组和输出方程组是如何确定的?它们可决定同步时序电路中哪些电路?
- 6.3.6 如何检查同步时序电路的自校正能力?如果要求电路从特定状态开始工作,应如何处理?
- 6.3.7 造成时钟偏移的原因是什么?一般如何解决?

6.4 异步时序逻辑电路的分析

本节主要讨论用触发器构成的脉冲异步时序电路的分析方法。异步时序电路与同步时序电路的主要区别在于电路中没有统一的时钟脉冲,因而各存储电路不是同时更新状态,状态之间没有准确的时间分界。在分析脉冲异步时序电路时必须注意以下几点。

(1) 分析状态转换时必须考虑各触发器的时钟信号作用情况

异步时序电路中,由于各个触发器只有在其 CP_n (或 \overline{CP}_n ,下标 n 表示电路中第 n 个触发器) 端的信号有效时,才有可能改变状态。因此,在分析状态转换时,首先应根据给定的电路列出各个触发器的时钟信号方程组,根据该方程组的各表达式分别确定相应触发器的 CP_n (或 \overline{CP}_n) 端是否有时钟信号的作用,并用变量 cp_n 表示:有作用,则令 $cp_n = 1$;否则 $cp_n = 0$ 。对于上升沿触发的触发器,当其 CP_n 端的信号由 0 变 1 时 $cp_n = 1$;反之,对下降沿触发的触发器,则在 \overline{CP}_n 信号由 1 变 0 时 $cp_n = 1$ 。然后再根据激励信号确定那些 $cp_n = 1$ 的触发器的次态, $cp_n = 0$ 的触发器则保持原有状态不变。这里, cp_n 不是一个逻辑变量,但参与各转换方程的逻辑运算。

(2) 每一次状态转换必须从输入信号所能触发的第一个触发器开始逐级确定

同步时序电路的分析可以从任意一个触发器开始推导状态的转换,而异步时序电路每一次状态转换的分析必须从输入信号所能作用的第一个触发器开始推导,确定它的状态变化,然后根据它的输出信号分析下一个触发器的时钟信号以确定 cp_n 的值,进一步决定该触发器是否发生状态转换。像这样依次逐级分析,直到最后一个触发器。待全部触发器的转换状态导出后,才能最终确定电路的次态,填入转换表或状态图。

(3) 每一次状态转换都有一定的时间延迟

同步时序电路的所有触发器几乎是同时更新状态的,与之不同,异步时序电路各个触发器之间的状态更新存在一定的延迟,也就是说,从现态 S^n 到次态 S^{n+1} 的转换过程中有一段各触发器分别延迟更新状态的“不稳定”时间。在此期间,整个电路的状态是不确定的。只有当全部触发

器的状态更新完毕,电路才进入新的“稳定”状态,即次态 S^{n+1} 。因此,异步时序电路的输入信号(包括时钟信号)必须等待电路进入稳定状态之后才允许发生改变,否则电路会处在不确定的状态。由于上述延迟时间的存在,对于同一系列的逻辑电路,类似功能的同步时序电路的速度要快于异步时序电路。

下面以两个实例来说明异步时序电路的分析过程。

例 6.4.1 分析图 6.4.1 所示逻辑电路图。

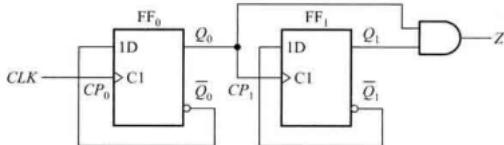


图 6.4.1 例 6.4.1 的逻辑电路图

解: 在图 6.4.1 的电路中,两触发器 FF_0 和 FF_1 的 CP_0 和 CP_1 未共用时钟信号,故属于异步时序电路。

(1) 列出各逻辑方程组

① 时钟信号方程组

根据逻辑图列出各触发器的时钟信号表达式

$$CP_0 = CLK$$

$$CP_1 = Q_0$$

② 激励方程组

$$D_0 = \bar{Q}_0$$

$$D_1 = \bar{Q}_1$$

③ 转换方程组

这时需要考虑各触发器时钟信号 CP_n 的作用:只有 $cp_n = 1$ 发生后,相应触发器才可能转换状态,当 $cp_n = 0$,即 $\overline{cp_n} = 1$ 时,触发器应保持原态。因此,触发器的特性方程中应引入 cp_n 而改写为如下的转换方程组:

$$Q_0^{n+1} = D_0 cp_0 + Q_0^n \overline{cp_0} = \bar{Q}_0^n cp_0 + Q_0^n \overline{cp_0} \quad (6.4.1)$$

$$Q_1^{n+1} = D_1 cp_1 + Q_1^n \overline{cp_1} = \bar{Q}_1^n cp_1 + Q_1^n \overline{cp_1} \quad (6.4.2)$$

式(6.4.1)和式(6.4.2)中右边的第二项表示,当触发器没有时钟信号作用时,状态保持不变。

④ 输出方程组

$$Z = Q_1 Q_0$$

(2) 列出转换表

列转换表的方法与同步时序电路分析过程基本相似,只是还应注意各触发器是否存在 $cp_n = 1$,因此,可在转换表中增加 cp_0, cp_1 两列。对应于输入信号 CLK 的每一个上升沿, $cp_0 = 1$, 将 cp_0 和 FF_0 的现态 Q_0^n 代入式(6.4.1),从而得到 Q_0^{n+1} 。因为只有对应于 Q_0 由 0 到 1 的跳变,才有 $cp_1 = 1$,而对应于 Q_0 由 1 到 0 的转换, $cp_1 = 0$,所以在每一次 Q_0 的状态转换后,应首先根据 FF_1 时钟信号的逻辑表达式确定 cp_1 ,然后将 FF_1 的现态 Q_1^n 和相应的 cp_1 代入式(6.4.2),求得其次态 Q_1^{n+1} 。如果从 $Q_1^n Q_0^n$ 为 00 开始推导,可以得到如表 6.4.1 所示的转换表。

表 6.4.1 例 6.4.1 的转换表

Q_1^n	Q_0^n	cp_1	cp_0	Q_1^{n+1}	Q_0^{n+1}	Z
0	0	1	1	1	1	0
0	1	0	1	0	0	0
1	0	1	1	0	1	0
1	1	0	1	1	0	1

(3) 画出状态图和时序图

由表 6.4.1 所示的转换表可画出如图 6.4.2 所示的状态图。

根据状态图和具体触发器的传输延迟时间 t_{plh} 和 t_{phl} ,可以画出时序图,如图 6.4.3 所示。可以看出,由于两个触发器异步翻转之间存在延迟,电路有短时间存在着不确定的状态,如果使用 74HCT74 双 D 触发器实现图 6.4.1 所示的电路,这段时间大约在 40 ns 左右。

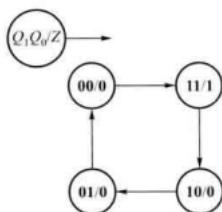


图 6.4.2 例 6.4.1 的状态图

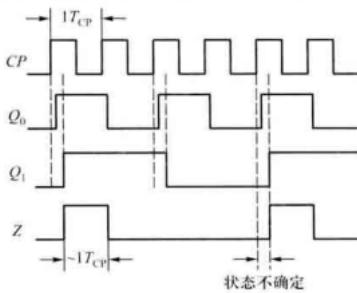


图 6.4.3 例 6.4.1 的时序图

(4) 逻辑功能分析

由状态图和时序图可知,该电路是一个异步二进制递减计数器, Z 信号的上升沿可触发借位操作。也可把它看作为一个序列信号发生器,输出序列脉冲信号 Z 的重复周期为 $4T_{CP}$, 脉宽约为 $1T_{CP}$ 。

例 6.4.2 分析图 6.4.4 所示逻辑电路。

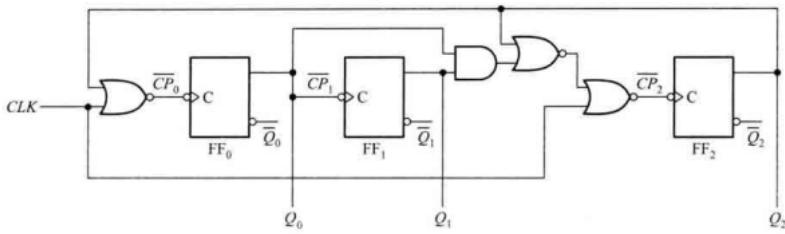


图 6.4.4 例 6.4.2 的逻辑电路图

解：这是由 3 个下降沿触发的 T' 触发器构成的异步时序电路。只要相应触发器的时钟输入端 \overline{CP}_n 出现一次从 1 到 0 的跳变，其状态就会翻转一次。下面按步骤进行分析。

(1) 列出各逻辑方程组

① 时钟信号方程组

根据逻辑图列出各触发器时钟信号的逻辑表达式

$$\overline{CP}_0 = \overline{Q_2} + \overline{CLK} = \overline{Q_2} \overline{CLK} \quad (6.4.3)$$

$$\overline{CP}_1 = Q_0 \quad (6.4.4)$$

$$\overline{CP}_2 = \overline{\overline{Q_0}Q_1 + Q_2 + \overline{CLK}} = (\overline{Q_0}Q_1 + Q_2) \overline{CLK} \quad (6.4.5)$$

② 转换方程组

引入 cp_n 后， T' 触发器的特性方程 $Q_n^{n+1} = \overline{Q_n^n}$ 应改写为如下转换方程

$$Q_0^{n+1} = \overline{Q_0^n} cp_0 + Q_0^n \overline{cp_0} \quad (6.4.6)$$

$$Q_1^{n+1} = \overline{Q_1^n} cp_1 + Q_1^n \overline{cp_1} \quad (6.4.7)$$

$$Q_2^{n+1} = \overline{Q_2^n} cp_2 + Q_2^n \overline{cp_2} \quad (6.4.8)$$

注意：此例中每当 \overline{CP}_n 发生由 1 到 0 的跳变时 $cp_n = 1$ 。

③ 输出方程组

即三个触发器的输出信号 Q_2, Q_1, Q_0 。

(2) 列出转换表

从现态 $Q_2 = Q_1 = Q_0 = 0$ 开始列转换表。应从 CLK 所能触发的第一个触发器 FF_0 开始推导其次态。首先确定 cp_0 ：根据式(6.4.3)，由于 $Q_2 = 0, CLK$ 信号 0 → 1 的跳变必然使 \overline{CP}_0 产生一个 1 → 0 的跳变，所以 $cp_0 = 1$ 。然后将 cp_0 和现态 $Q_0^n = 0$ 代入式(6.4.6)，得到 $Q_0^{n+1} = 1$ 。类似地，根据式(6.4.5)，由于 $Q_2 = Q_1 = Q_0 = 0, \overline{CP}_2$ 为 0，此时 CLK 的任何变化都不会使 \overline{CP}_2 产生下降沿，故 $cp_2 = 0, FF_2$ 不会改变状态， $Q_2^{n+1} = 0$ 。这时，再根据式(6.4.4)确定 cp_1 ：因为 Q_0 是 0 → 1 跳变，所以

$cp_1 = 0$, Q_1 也将保持原状。 CLK 信号第一个上升沿到来后, 电路状态改变为 **001**。以此类推, 可得电路的转换表, 如表 6.4.2 所示。

表 6.4.2 例 6.4.2 的转换表

Q_2^n	Q_1^n	Q_0^n	cp_2	cp_1	cp_0	Q_2^{n+1}	Q_1^{n+1}	Q_0^{n+1}
0	0	0	0	0	1	0	0	1
0	0	1	0	1	1	0	1	0
0	1	0	0	0	1	0	1	1
0	1	1	1	1	1	1	0	0
1	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	0	1
1	1	0	1	0	0	0	1	0
1	1	1	1	0	0	0	1	1

(3) 画出状态图

由表 6.4.2 所示的转换表可画出状态图, 如图 6.4.5 所示。该图表明, 当电路处于循环外的状态(**101**、**110** 和 **111** 三个状态)时, 在 CLK 信号出现第一个上升沿后, 电路便能进入有效循环状态。

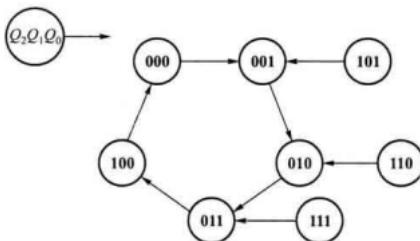


图 6.4.5 例 6.4.2 的状态图

(4) 逻辑功能分析

由状态图可知, 图 6.4.4 所示电路是一个具有 **000 ~ 100** 五个连续二进制编码状态的异步计数器。

复习思考题

- 6.4.1 异步时序电路的分析与同步时序电路的分析主要有哪些不同之处？
- 6.4.2 分析异步时序电路时，为什么需要列出触发器的时钟信号方程组？
- 6.4.3 为什么在分析异步时序电路的状态转换时，必须按信号作用的顺序对每个触发器进行逐个推导，才能确定电路的次态？
- 6.4.4 异步时序电路在每次状态转换时为什么会出现短时间的不稳定状态？这种不稳定的延续时间与门电路和触发器的延迟时间有什么关系？此外还可能与哪些因素相关？
- 6.4.5 为什么异步时序电路的输入信号必须在电路转换状态稳定之后才允许发生改变？

6.5 若干典型的时序逻辑电路

本节介绍在数字系统中广泛应用的几种典型时序电路——寄存器、移位寄存器和计数器的电路结构和工作原理，它们是时序电路的基本逻辑部件，与各种组合电路一起，可以构成逻辑功能极其复杂的数字系统。寄存器、移位寄存器和计数器有很多种类的中规模集成电路，可直接用于较简单的数字系统。对于较复杂的时序电路设计，应选择可编程逻辑器件实现，大批量生产的电子产品采用专用集成电路则可降低成本。本节提及的几种中规模集成电路，都经过精心设计和多年改进，具有较完善的逻辑功能及电路性能，至今仍广泛应用。它们的设计思想和电路结构，可作为可编程逻辑器件或专用集成电路设计的范例。在一些逻辑集成开发软件中也包含这些电路的“宏模块”，直接引用可简化数字系统设计。总之，充分理解这些典型时序电路的工作原理和电路结构并灵活应用，对于各种数字系统的设计都有裨益。

6.5.1 寄存器和移位寄存器

1. 寄存器

寄存器是数字系统中用来存储二进制数据的逻辑部件。1个触发器可存储1位二进制数据，存储 N 位二进制数据的寄存器需要用 N 个触发器组成。

由8个触发器构成的8位寄存器的逻辑图如图6.5.1所示。图中， $D_7 \sim D_0$ 是8位数据输入端，在 CP 脉冲上升沿作用下， $D_7 \sim D_0$ 端的数据同时存入相应的触发器，是一种极为简单的同步时序电路。当输出使能控制信号 $\overline{OE} = 0$ 时，触发器存储的数据通过三态门输出端 $Q_7 \sim Q_0$ 并行输出。表6.5.1是典型的中规模集成8位寄存器74HC/HCT374的功能表。

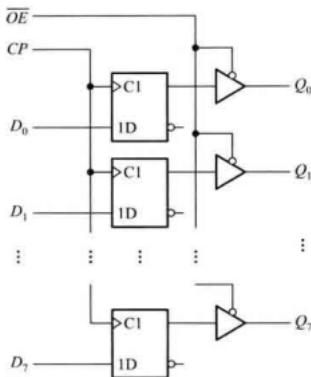


图 6.5.1 带输出缓冲器的 8 位寄存器

表 6.5.1 74HC/HCT374 的功能表

工作模式	输入			内部触发器 Q_N^{N+1}	输出 $Q_0 \sim Q_7$
	\overline{OE}	CP	D_N		
存入和读出数据	L	↑	L*	L	相应内部触发器的状态
	L	↑	H*	H	
存入数据, 禁止输出	H	↑	L*	L	高阻
	H	↑	H*	H	高阻

注: D_N 和 Q_N^{N+1} 的下标表示第 N 位触发器。L* 和 H* 表示 CP 脉冲上升沿之前瞬间 D_N 的电平。

从存储数据的角度看, 5.3.2 小节介绍的八 D 锁存器 74HC/HCT373 与此处的 8 位寄存器 74HC/HCT374 具有类似的逻辑功能。两者的区别在于, 前者是电平敏感电路, 而后者是脉冲边沿敏感电路。它们有不同的应用场合, 主要取决于控制信号与数据信号之间的定时关系, 以及控制存储数据的方式。如果数据状态的更新可能出现在控制信号(锁存器的 LE)开始有效之后, 则只能使用锁存器, 它不能保证各个锁存器输出同时更新状态; 如果能确保输入数据的更新在控制信号(寄存器的 CP)触发边沿出现之前稳定, 并要求输出同时更新状态, 则可选择寄存器。一般来说, 寄存器比锁存器具有更好的同步性能和抗干扰性能。

与“宽总线”锁存器类似, 寄存器也有 16 位、18 位、20 位和 32 位的“宽总线”集成电路。为了应用的灵活方便, 一般都把其中的 D 触发器分为若干组, 各组间既可同步工作, 又可异步工作。另外, 也有 1~2 个触发器的“小逻辑”寄存器产品。

2. 移位寄存器

(1) 基本的移位寄存器

如果将若干个触发器级联成如图 6.5.2 所示电路，则构成基本的移位寄存器。图中是一个 4 位移位寄存器，串行二进制数据从输入端 D_{si} 输入，左边触发器的输出作为右邻触发器的数据输入。若将串行数码 $D_3 D_2 D_1 D_0$ 从高位 (D_3) 至低位 (D_0) 按时钟脉冲间隔依次送到 D_{si} 端，经过第一个时钟脉冲后， $Q_0 = D_3$ 。由于跟随 D_3 后面的是 D_2 ，因此经过第二个时钟脉冲后，触发器 FF_0 的状态移入触发器 FF_1 ，而 FF_0 转变为新的状态，即 $Q_1 = D_3, Q_0 = D_2$ 。以此类推，可得该移位寄存器的功能如表 6.5.2 所示（ \times 表示不确定状态）。由表可知，输入数码依次由左侧触发器移到右侧触发器。经过 4 个时钟脉冲后，4 个触发器的输出状态 $Q_3 Q_2 Q_1 Q_0$ 与输入数码 $D_3 D_2 D_1 D_0$ 相对应。为了加深理解，在图 6.5.3 中画出了数码 1101（即 $D_3 = 1, D_2 = 1, D_1 = 0, D_0 = 1$ ）在移位寄存器中移位的波形，经过 4 个时钟脉冲后，1101 出现在触发器的输出端 $Q_3 Q_2 Q_1 Q_0$ 。这样，就将串行输入数据转换为并行输出数据 D_{po} 。

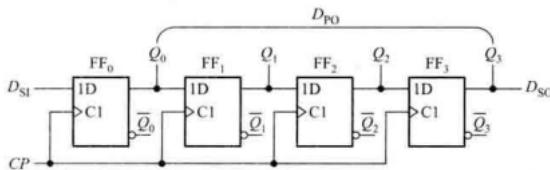


图 6.5.2 用 D 触发器构成的 4 位移位寄存器

表 6.5.2 图 6.5.2 电路的功能

CP	Q_0	Q_1	Q_2	Q_3
第一个 CP 脉冲之前	\times	\times	\times	\times
1	D_3	\times	\times	\times
2	D_2	D_3	\times	\times
3	D_1	D_2	D_3	\times
4	D_0	D_1	D_2	D_3

在图 6.5.3 中还画出了第 5 到第 8 个时钟脉冲作用下，输入数码在寄存器中移位的波形。由图可见，在第 8 个时钟脉冲作用后，数码已从串行数据输出端 D_{so} （即 Q_3 端）全部移出寄存器。也就是说，随着时钟信号的推移，从输出端 D_{so} 可得到 1101 的串行输出。

一般来说， N 位移位寄存器要由 N 个触发器构成，需要 $N \cdot T_{cp}$ 来完成串行到并行的数据转换，同样也需要 $N \cdot T_{cp}$ 来实现并行到串行的数据输出。这里， T_{cp} 为时钟周期。

从上述操作可以看出，移位寄存器只能用脉冲边沿敏感的触发器，而不能用电平敏感的锁存器来构成，因为在时钟脉冲高电平期间，锁存器输出跟随输入变化的特性将使移位操作失去控制。显然，移位寄存器属于同步时序电路。

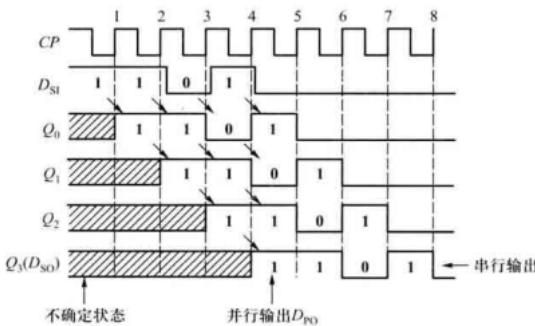


图 6.5.3 图 6.5.2 电路的时序图

(2) 多功能双向移位寄存器

① 工作原理

有时需要对移位寄存器的数据流向加以控制, 实现数据的双向移动, 其中一个方向称为右移, 另一个方向则为左移, 这种移位寄存器称为**双向移位寄存器**。由于国家标准规定, 逻辑图中最低有效位(LSB)到最高有效位(MSB)的电路排列顺序应从上到下, 从左到右。因此, 定义移位寄存器中的数据从低位触发器移向高位为右移, 反之则为左移。这一点与通常计算机程序中的规定相反, 后者从自然二进制数的排列考虑, 将数据移向高位定义为左移, 反之为右移。

为了扩展逻辑功能和增加使用的灵活性, 在双向移位基础上, 还可以增加并行输入、并行输出等功能, 构成多功能移位寄存器, 其工作模式的简化示意图如图 6.5.4 所示。

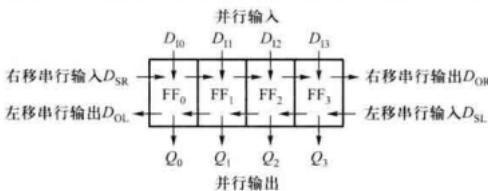


图 6.5.4 多功能移位寄存器工作模式简图

图 6.5.5 是实现数据保持、右移、左移、并行输入和并行输出的一种电路方案。图中的 D 触发器 FF_m 是 N 位移位寄存器中的第 m 位触发器, 在其数据输入端插入了一个 4 选 1 数据选择器, 用两位编码选择输入信号 S_1, S_0 控制触发器输入信号 D_m 的来源。当 $S_1 = S_0 = 0$ 时, 选择该触发器本身的输出 Q_m , 次态为 $Q_m^{n+1} = D_m = Q_m^n$, 触发器保持状态不变; 当 $S_1 = 0, S_0 = 1$ 时, 触发器 FF_{m-1} 的输出 Q_{m-1} 被选中, 故 CP 脉冲上升沿到来时, FF_m 存入 FF_{m-1} 此前的逻辑值, 即 $Q_m^{n+1} = Q_{m-1}^n$;

而 $Q_{m+1}^{n+1} = Q_m^n$, 从而实现右移功能; 类似地, 当 $S_1 = 1, S_0 = 0$ 时, 数据选择器选择 Q_{m+1} , 实现左移功能; 而当 $S_1 = S_0 = 1$ 时, 则选择并行输入数据 D_{lm} , 其次态 $Q_m^{n+1} = D_{lm}$, 从而完成并行数据的置入。上述四种功能概述于表 6.5.3 中。此外, 在各触发器的输出端 $Q_{m-1} \sim Q_0$ 可以得到 N 位并行数据的输出。

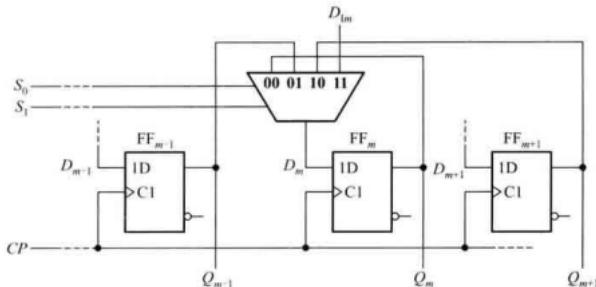


图 6.5.5 实现多种功能双向移位寄存器的一种方案

表 6.5.3 图 6.5.5 的功能表

控制信号		功 能
S_1	S_0	
0	0	保 持
0	1	右 移
1	0	左 移
1	1	并行输入

② 典型的多功能 4 位双向移位寄存器

采用图 6.5.5 方案实现数据保持、右移、左移、并行输入和并行输出的一种多功能典型的 4 位双向移位寄存器如图 6.5.6 所示。它采用了 4 个 SR 触发器分别配以非门构成 4 个 D 触发器, 连接在各触发器 1R 端上的与或非门则实现了数据选择器的功能。设连接在触发器 1R 端的与或非门输出为 \bar{D} , 1S 端的输入则为 D , 将二者分别代入 SR 触发器的特性方程, 得

$$Q^{n+1} = S + \bar{R}Q^n = D + \bar{D}Q^n = D$$

于是将 SR 触发器转换为 D 触发器, 功能与图 6.5.5 完全一致。图中, D_{sr} 是右移串行数据输入端, D_{sl} 是左移串行数据输入端, $D_{10} \sim D_{13}$ 是并行数据输入端, \overline{CR} 连接到每个触发器的直接置 0 端, 可控制寄存器异步清零。图中输入缓冲器可以减轻前级的负载, 输出缓冲器可以提高驱动能力。图 6.5.6 所示实际是 74HC/HCT194 的内部逻辑图, 表 6.5.4 是它的功能表。表中内容: 第

1 行表示寄存器异步清零操作, 第 2 行为保持状态, 第 3、4 行为串行数据右移操作, 第 5、6 行为串行数据左移操作, 第 7 行是并行数据的同步置入操作。

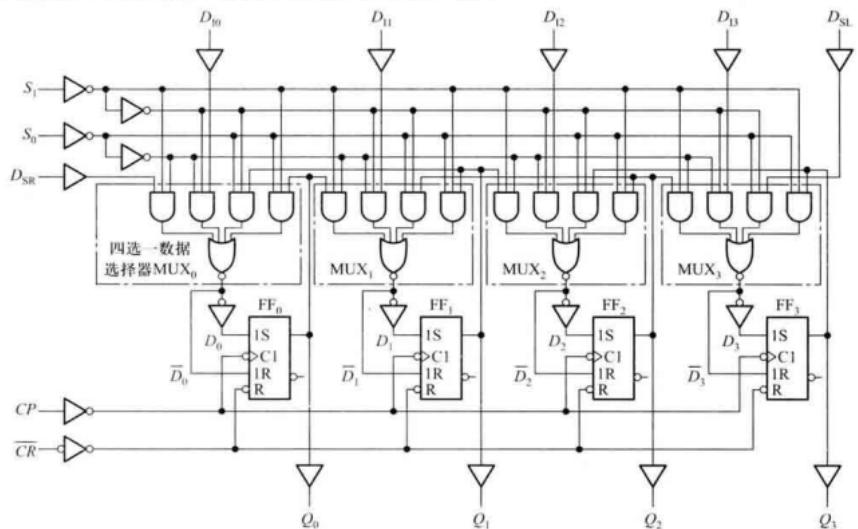


图 6.5.6 多功能 4 位双向移位寄存器 74HC/HCT194 的逻辑图

表 6.5.4 多功能 4 位双向移位寄存器 74HC/HCT194 的功能表

清零	控制信号		时钟 CP	输入				输出				功 能	
				串行输入		并行输入		Q_0^{n+1}	Q_1^{n+1}	Q_2^{n+1}	Q_3^{n+1}		
CR	S ₁	S ₀		右移 D_{sr}	左移 D_{sl}	D_{10}	D_{11}	D_{12}	D_{13}	L	L	L	L
L	x	x	x	x	x	x	x	x	x	L	L	L	L
H	L	L	x	x	x	x	x	x	x	Q_0^n	Q_1^n	Q_2^n	Q_3^n
H	L	H	↑	L	x	x	x	x	x	L	Q_0^n	Q_1^n	Q_2^n
H	L	H	↑	H	x	x	x	x	x	H	Q_0^n	Q_1^n	Q_2^n
H	H	L	↑	x	L	x	x	x	x	Q_1^n	Q_2^n	Q_3^n	L
H	H	L	↑	x	H	x	x	x	x	Q_1^n	Q_2^n	Q_3^n	H
H	H	H	↑	x	x	D_{10}^*	D_{11}^*	D_{12}^*	D_{13}^*	D_{10}	D_{11}	D_{12}	D_{13}

注: D_{1x}^* 表示 CP 脉冲上升沿之前瞬间 D_{1x} 的电平。

有时要求在移位过程中,数据仍保持在寄存器中不丢失。此时,只要将移位寄存器最高位的输出接至最低位的输入,或将最低位的输出接至最高位的输入,便可实现这个功能,称为环形移位寄存器。它亦可作计数器使用,称为环形计数器,将在 6.5.2 节中讨论。

6.5.2 计数器

计数器是最常用的时序电路之一,它们不仅可用于对脉冲进行计数,还可用作分频、定时、产生节拍脉冲以及其他时序信号。计数器的种类不胜枚举,按触发器动作分类,可分为同步计数器和异步计数器;按编码数值增减分类,可分为递增计数器、递减计数器和可逆计数器;按编码分类,又可分为二进制码计数器、BCD 计数器、循环码计数器等。计数器运行时,从某一状态开始依次遍历不重复的各个状态后完成一次循环,所经过的状态总数称为计数器的模 (Modulo), 并用 M 表示。若某个计数器在 n 个状态下循环计数,通常则称之为模 n 计数器,或 $M=n$ 计数器。例如一个在 60 个不同状态中循环转换的计数器,就可称为模 60, 或 $M=60$ 计数器。有时也把模 n 计数器称为 n 进制计数器。

1. N 位二进制计数器

按照二进制数自然递增或递减编码的计数器称为二进制计数器 (Binary Counter), N 位二进制计数器的模为 2^N , 由 N 个触发器组成。

(1) N 位异步二进制计数器

图 6.5.7 是一个 4 位异步二进制计数器的逻辑图, 它由 4 个下降沿触发的 T' 触发器组成。计数脉冲 \overline{CP} 加至触发器 FF_0 的时钟脉冲输入端, 每输入一个计数脉冲, FF_0 翻转一次。 FF_1 、 FF_2 和 FF_3 都以前级触发器的 Q 端输出作为触发信号, 当 Q_0 由 1 变 0 时, FF_1 翻转, 其余类推。分析其工作过程, 不难得到输出波形如图 6.5.8 所示。由图可见, 从初态 **0000** (可由 \overline{CR} 输入低电平脉冲使 4 个触发器全部置 0) 开始, 每输入一个计数脉冲, 计数器的状态就按二进制编码值递增 1, 输入第 16 个计数脉冲后, 计数器又回到 **0000** 状态。显然, 该计数器以 16 个 \overline{CP} 脉冲构成一个计数周期, 是模 16 ($M=16$) 递增计数器。其中, Q_0 的频率是 \overline{CP} 的 $1/2$, 即实现了 2 分频, Q_1 得到 \overline{CP} 的 4 分频, 以此类推, Q_2 、 Q_3 分别对 \overline{CP} 进行了 8 分频和 16 分频, 因而, 计数器也可作为分频器使用。

异步二进制计数器的工作原理和结构均较简单, 但各触发器不是在同一时钟沿作用下同时翻转, 而是逐级脉动翻转实现计数进位的, 故亦称之为纹波计数器 (Ripple Counter)。图 6.5.8 中的虚线是考虑了触发器逐级翻转过程中平均传输延迟时间 t_{pd} 的波形。由于各触发器不在同一时间翻转, 因此, 若用这种计数器驱动组合逻辑电路, 则可能出现瞬间错误的逻辑输出。例如, 当计数值从 **0111** 加 1 时, Q_3 、 Q_2 、 Q_1 、 Q_0 要先后经过 **0110**、**0100**、**0000** 几个状态, 才最终翻转为 **1000**。

另外, 当计数脉冲频率很高时, $Q_3 \sim Q_0$ 甚至会出现编码输出分辨不清的情况。对于一个 N 位二进制异步计数器来说, 从一个计数脉冲作用在第一个触发器开始, 到第 N 个触发器翻转达到稳定状态为止, 需要经历的时间为 Nt_{pd} 。为了保证正确地检出计数器的输出状态, 必须满足

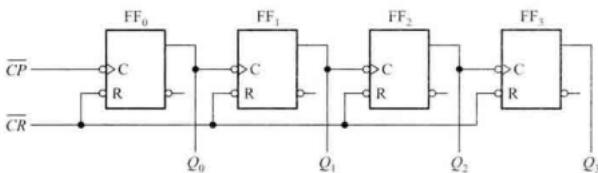


图 6.5.7 4 位异步二进制计数器逻辑图

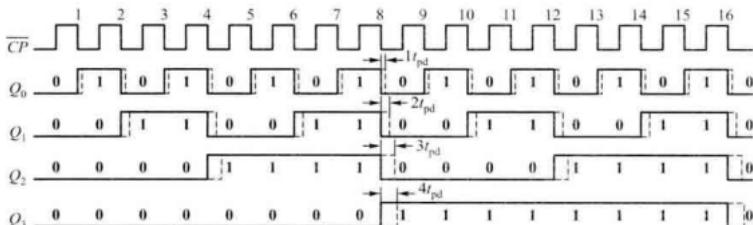


图 6.5.8 4 位异步二进制计数器时序图

$T_{cp} \gg N t_{pd}$ 的条件, 其中, T_{cp} 为计数脉冲 \overline{CP} 的周期。

(2) N 位同步二进制计数器

为了提高计数速度, 可采用同步计数器。其特点是, 计数脉冲作为时钟信号同时接于所有触发器的时钟脉冲输入端。在每次计数脉冲沿到来之前, 根据当前计数器状态, 利用组合逻辑电路产生触发器的激励条件。当计数脉冲沿到来时, 所有应翻转的触发器同步翻转, 同时, 所有应保持原状态的触发器不发生变化。由于不存在异步计数器那种纹波进位造成的延迟时间积累, 所以能取得较高的计数速度, 输出编码也不会发生纹波进位时的那种混乱。显然, 同步计数器是一种同步时序电路。

① 工作原理

表 6.5.5 是 4 位二进制计数器的转换表^①。观察该表递增计数一栏可以看出, Q_0 在每个计数脉冲作用后都要翻转一次; Q_1 只在 $Q_0=1$ 的条件下接受计数脉冲作用而翻转; Q_2 则在 $Q_0=Q_1=1$ 的次态翻转; Q_3 在 $Q_0=Q_1=Q_2=1$ 的次态翻转。当用 T 触发器实现 4 位同步二进制递增计数器时, 可根据上述激励条件, 确定每个 T 触发器的激励方程为

① 表 6.5.5 是时序电路转换表的简化形式, 只给出在一系列时钟脉冲信号作用下电路状态的转换顺序, 若以某行状态作为现态, 下一行则为次态。

$$\begin{cases} T_0 = \mathbf{1} \\ T_1 = Q_0 \\ T_2 = Q_1 Q_0 \\ T_3 = Q_2 Q_1 Q_0 \end{cases}$$

即除了触发器 FF_0 的 $T_0 = \mathbf{1}$ 外, 其他触发器都是在比其低的各位均为 $\mathbf{1}$ 时准备好翻转条件。由此类推, 可以扩展到更多的位数。当用 N 个 T 触发器构成 N 位同步二进制递增计数器时, 各触发器的激励方程为

$$\begin{cases} T_0 = \mathbf{1} \\ T_i = Q_{i-1} Q_{i-2} \cdots Q_1 Q_0 = \prod_{j=0}^{i-1} Q_j \quad (i=1, 2, \dots, N-1) \end{cases} \quad (6.5.1)$$

表 6.5.5 4 位二进制计数器的转换表

计数顺序	递增计数	递减计数
	$Q_3 Q_2 Q_1 Q_0$	$Q_3 Q_2 Q_1 Q_0$
0	0 0 0 0	0 0 0 0
1	0 0 0 1	1 1 1 1
2	0 0 1 0	1 1 1 0
3	0 0 1 1	1 1 0 1
4	0 1 0 0	1 1 0 0
5	0 1 0 1	1 0 1 1
6	0 1 1 0	1 0 1 0
7	0 1 1 1	1 0 0 1
8	1 0 0 0	1 0 0 0
9	1 0 0 1	0 1 1 1
10	1 0 1 0	0 1 1 0
11	1 0 1 1	0 1 0 1
12	1 1 0 0	0 1 0 0
13	1 1 0 1	0 0 1 1
14	1 1 1 0	0 0 1 0
15	1 1 1 1	0 0 0 1
16	0 0 0 0	0 0 0 0

由表 6.5.5 递减计数一栏可以看出, Q_0 同样在每个计数脉冲沿到来时都要翻转一次; 而其

他位都是在比其低的各位均为 0 时准备好翻转条件, 在次态完成翻转, 实现二进制计数的递减借位。当用 T 触发器实现 4 位同步二进制递减计数器时, 激励方程为

$$\left\{ \begin{array}{l} T_0 = 1 \\ T_1 = \bar{Q}_0 \\ T_2 = \bar{Q}_1 \bar{Q}_0 \\ T_3 = \bar{Q}_2 \bar{Q}_1 \bar{Q}_0 \end{array} \right.$$

对于 N 位同步二进制递减计数器, 类似地有

$$\left\{ \begin{array}{l} T_0 = 1 \\ T_i = \bar{Q}_{i-1} \bar{Q}_{i-2} \cdots \bar{Q}_1 \bar{Q}_0 = \prod_{j=0}^{i-1} \bar{Q}_j \quad (i = 1, 2, \dots, N-1) \end{array} \right. \quad (6.5.2)$$

根据式(6.5.1)和式(6.5.2)可得由 T 触发器构成的 4 位同步二进制计数器逻辑图如图 6.5.9 所示。其中, 图(a)为递增计数器, 图(b)为递减计数器。

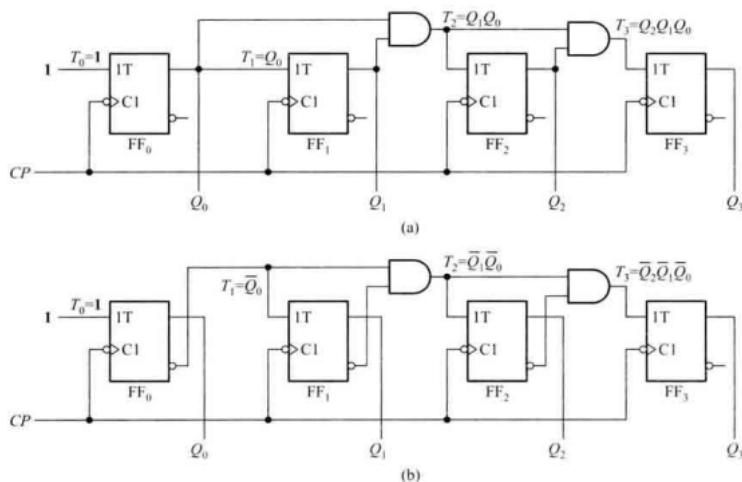


图 6.5.9 4 位同步二进制计数器

(a) 递增计数器 (b) 递减计数器

图 6.5.10 所示是图 6.5.9(a)所示计数器的时序图, 其中虚线是考虑到触发器传输延迟时间 t_{pd} 的波形。由图 6.5.10 可知, 在同步计数器中, 由于计数脉冲 CP 同时作用于各触发器, 所有触发器的状态更新几乎是同时进行的, 都比计数脉冲沿的作用时间滞后一个 t_{pd} 。因此, 输出状

态比异步二进制计数器稳定。对于同一逻辑系列、位数相同的计数器，同步计数器的工作速度一般高于异步计数器。

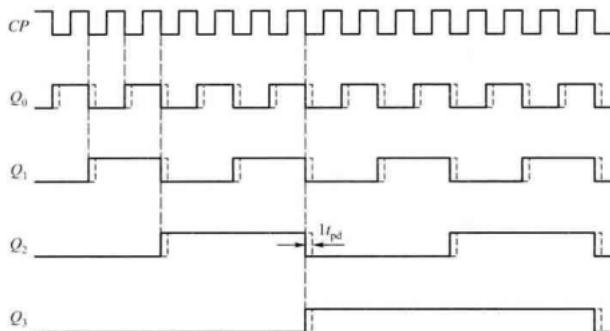
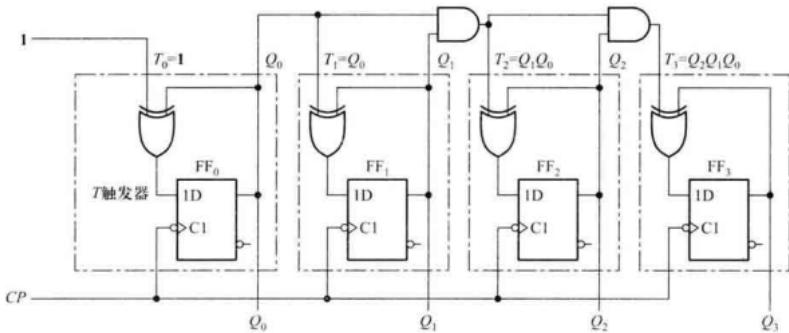


图 6.5.10 4 位同步二进制递增计数器时序图

当用 D 触发器实现二进制计数器时，可根据 5.5 节图 5.5.9(a) D 触发器与 T 触发器的关系，得到 D 触发器构成的 4 位二进制递增计数器如图 6.5.11 所示。其中，4 个点画线方框内为等效的 T 触发器。

图 6.5.11 D 触发器构成的 4 位同步二进制递增计数器

② 同步计数器的进位方法

递增计数器的低位计满后激励高位触发器翻转的操作称为计数进位。图 6.5.11 所示电路

的进位信号是通过一系列串行的逻辑门逐位向高位传递的，这种计数器称为同步串行计数器。如果一个计数器中有 N 位触发器，将它们按二进制数位从低位向高位编号，最低位输出为 Q_0 ，那么， Q_0 的进位信号至少要经过 $N-2$ 个等效逻辑门的延迟，才可能到达最高位触发器的输入端。假设每个等效逻辑门的延迟时间相同，均为 t_{Gpd} ，那么计数器中组合电路的最大延迟时间为 $t_{\text{COMpd(max)}} = (N-2)t_{\text{Gpd}}$ 。显然， $t_{\text{COMpd(max)}}$ 将随着计数器的位数增多而增大。当计数器位数较多时，会降低电路的运行质量，或被迫降低电路的工作频率。

另一种计数器是同步并行计数器，电路结构如图 6.5.12 所示，其激励电路由 G_1, G_2, G_3 三个与门实现。和图 6.5.11 所示电路相比，虽然图 6.5.12 中各触发器的激励方程与前者相同，但后者在一次状态转换完成后，只需经过相当于一个等效逻辑门的传输延迟时间 t_{Gpd} ，就可将激励信号传递到所有高位触发器的 T 输入端，迎接下一个 CP 触发沿的到来。即无论计数器有多少位，电路的 $t_{\text{COMpd(max)}}$ 都等于一个等效逻辑门的传输延迟时间 t_{Gpd} 。因此，这种进位方式可以大大提高计数器的工作频率，位数越多，提高越明显。然而，随着计数器位数的增多，低位触发器的扇出和高位触发器激励电路的扇入会增多，例如，从图 6.5.12 所示电路中 Q_0 的扇出和 G_3 门的扇入就可看出这一点。

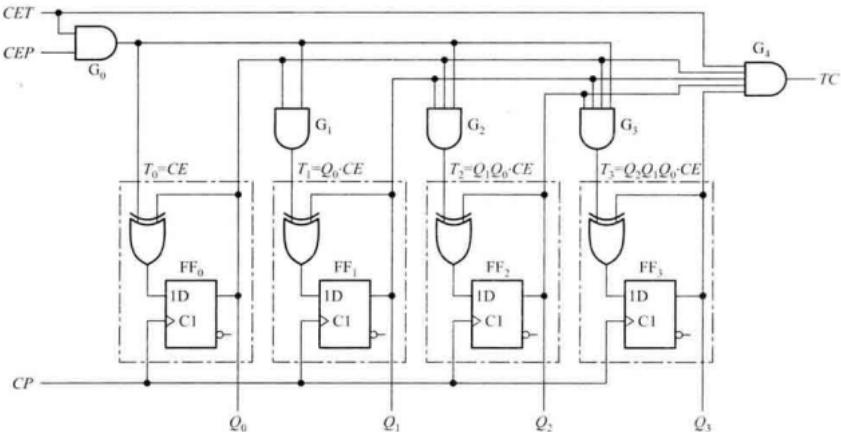


图 6.5.12 并行进位的同步 4 位二进制递增计数器

当计数器规模达到 4 位以上时，工程上常常采用折中处理的方案，将并行进位和串行进位相结合。方法是将多位计数器分为若干组，组内的触发器采用并行进位，而组与组之间的进位则以串行实现。图 6.5.13 示出按这种方案实现 16 位同步二进制计数的一个实例。图中使用了 4 组图 6.5.12 所示电路，各组从低位到高位分别以 $C_0 \sim C_3$ 命名。需要注意的是每一组中串行进位允许信号 CET 、并行进位允许信号 CEP 和进位输出信号 TC 的连接方法。其中， $TC = Q_0 Q_1 Q_2 \cdot CET$ ，是

向高位组的进位信号。对于 C_0 组, CET 与 CEP 相连, 可作为整个计数器的使能输入 CE , 当 $CE=1$ 时允许计数。其他各组的 CEP 均接收 C_0 组计数器的进位信号 TC_0 ; C_1 组 CET 直接与计数允许输入相连, C_2 、 C_3 组的 CET 接收相邻低位组的进位信号。电路工作时, 只有在 C_0 组发出进位信号, 即 $TC_0=1$ 后, 才允许所有高位组内部进行并行进位操作; 高位各组的进位信息则通过各自的 G_4 直接上传。很容易看出, C_0 组 Q_0 的翻转信号, 只需经过该组的 G_4 和 C_3 组的 G_0 、 G_3 三个等效逻辑门的延迟, 历时大约 $3t_{Gpd}$, 就能抵达整个计数器的最高位触发器, 也就是 C_3 组的 FF_3 。进一步分析可以证明, 电路中所有触发器的激励电路延迟都不会大于 $3t_{Gpd}$, 即 $t_{COMpl(max)} \approx 3t_{Gpd}$ 。因而, 这种并、串行结合的进位方法既具有较短的进位延迟, 又同时兼顾了电路的复杂程度。

图 6.5.13 所示的进位方法, 对于设计大型同步计数器是具有普遍意义的, 包括 PLD 和 ASIC 的应用。例如, 在高速数据采集系统中, 经常需要用计数器对 1k 以上“字节(8 bit)”或“字(16 bit)”的存储器实现与取样同步的高速地址扫描, 以便及时将采集的数据顺序存入存储器。用于这种目的的二进制计数器, 位数一般都超过 10 位, 如果采用并、串行结合的进位方式, 则可提高电路工作的可靠性, 或充分发挥已有电路的资源。

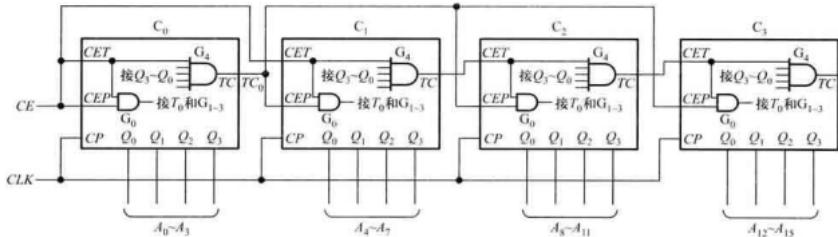


图 6.5.13 以图 6.5.12 所示电路实现的并行进位和串行进位结合的 16 位同步二进制计数器

③ 异步清零和同步并行置数

图 6.5.14 所示电路是功能更为完备的 4 位同步二进制计数器的逻辑图。

在许多应用中, 需要计数器从零状态开始工作, 为此, 在图 6.5.14 所示电路中增加了直接清零输入端 \overline{CR} 。可以看到, 清零信号 \overline{CR} 直接作用于 4 个触发器的置 0 端。当异步的 \overline{CR} 为低电平时, 将使电路中所有触发器无条件置 0, 而不管其他输入端是何状态(包括时钟信号 CP), 因此, $\overline{CR}=0$ 对计数器状态有优先级最高的控制权, 其他各种操作都是在 $\overline{CR}=1$ 的条件下才能执行。

计数器应用时, 有时需要将计数器预置为某种状态, 然后再从该状态开始计数。图 6.5.14 所示电路以图 6.5.12 为基础, 在每个 D 触发器的输入端前插入一个 2 选 1 数据选择器, 以选择从数据输入端 D_N 置入数据, 还是执行正常的计数功能, 并用一个公共置数控制端 \overline{PE} 进行控制。

由图 6.5.14 看出, $D_3 D_2 D_1 D_0$ 为 4 位并行数据输入端, 当 $\overline{PE} = 0$ 时, 数据选择器选择并行数据送至相应触发器的 D 输入端, 这样, 在下一个 CP 沿到来时, 输入端 $D_3 D_2 D_1 D_0$ 的数据被并行置入各触发器。这种在时钟脉冲作用下的并行置数称为同步置数。而当 \overline{PE} 恢复高电平时, 4 个 2 选 1 数据选择器选择相应异或门的输出信号送至触发器 D 输入端, 计数器恢复正常计数功能, 于是, 可在预置数据的基础上继续递增计数。

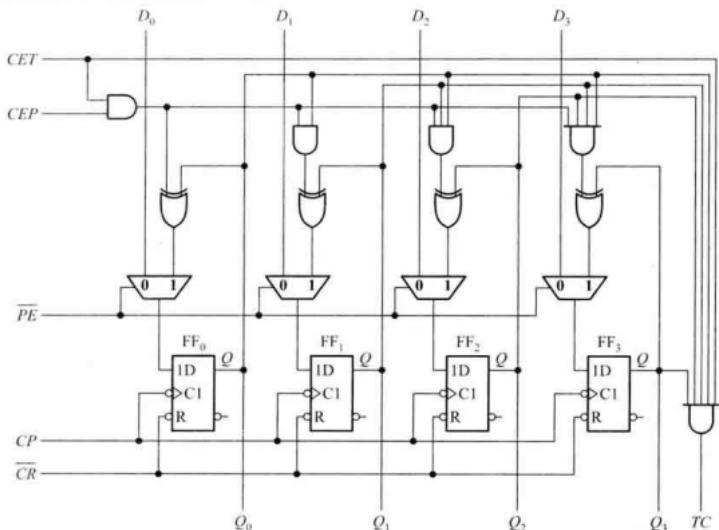


图 6.5.14 具有清零和并行置数功能的 4 位同步二进制递增计数器

由图 6.5.14 看出, $\overline{PE} = 0$ 具有次高优先级, 仅低于 $\overline{CR} = 0$, 计数和保持状态的操作都要求 $\overline{PE} = 1$ 。

④ 典型集成同步二进制计数器

74LVC161 是一种典型的高性能、低功耗 CMOS 4 位同步二进制递增计数器, 芯片的逻辑功能和实现原理与图 6.5.14 所示逻辑图相同。它的时钟最高工作频率可达 200MHz。图 6.5.15 所示为 74LVC161 的引脚功能图, 所有引脚名称与图 6.5.14 所示输入/输出信号线一致。表 6.5.6 示出它的功能表。

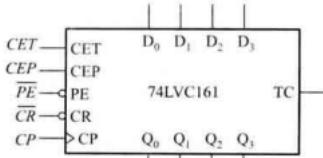


图 6.5.15 74LVC161 引脚功能图

表 6.5.6 74LVC161 的功能表

输入								输出					
清零	预置	使能		时钟	预置数据输入				Q_3	Q_2	Q_1	Q_0	进位
		\overline{CR}	\overline{PE}		CEP	CET	CP	D_3	D_2	D_1	D_0	TC	
L	x	x	x	x	x	x	x	x	L	L	L	L	L
H	L	x	x	↑	D_3^*	D_2^*	D_1^*	D_0^*	D_3	D_2	D_1	D_0	#
H	H	L	x	x	x	x	x	x	保	持			#
H	H	x	L	x	x	x	x	x	保	持			L
H	H	H	H	↑	x	x	x	x	计	数			#

注： D_n^* 表示 CP 脉冲上升沿之前瞬间 D_n 的电平。#表示只有当 $Q_3Q_2Q_1Q_0 \cdot CET = 1$ (正逻辑体系) 时， TC 输出为高电平，其余均为低电平。

74LVC161 的典型时序图如图 6.5.16 所示。图中，当清零信号 $\overline{CR} = 0$ 时，各触发器置 0。当 $\overline{CR} = 1$ 时，若 $\overline{PE} = 0$ ，在下一个时钟脉冲上升沿到来后，各触发器的输出状态与预置的输入数据相同。在 $\overline{CR} = \overline{PE} = 1$ 的条件下，若 $CEP = CET = 1$ ，则电路处于计数状态。图中从预置的 1100 开始计数，直到 $CEP \cdot CET = 0$ ，计数状态结束。此后处于禁止计数的保持状态： $Q_3Q_2Q_1Q_0 = 0010$ 。进位信号 TC 只有在 $Q_3Q_2Q_1Q_0 = 1111$ 且 $CET = 1$ 时输出为 1，其余时间均为 0。

与 74LVC161 工作原理和功能相似的集成计数器还有 74LVC163，其差别仅在于清零方式不同，后者为同步清零，即清零信号 \overline{CR} 的低电平必须等待下一个 CP 沿作用后才能完成清零操作。另外，与它们结构类似的计数器还有 74LVC160 和 74LVC162，这两种计数器是 8421 BCD 计数器，前者为异步清零，后者为同步清零。

2. 其他模数的计数器

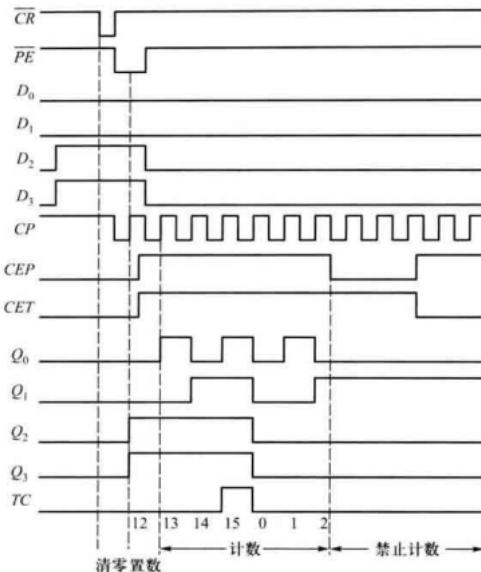
除了模为 2^3 的二进制计数器外，工程上还经常需要很多其他模数的计数器。这类计数器的状态中都或多或少存在无效状态，它们也分为同步和异步、递增、递减和可逆等各种类型。其中，最常用的是 BCD 计数器。

(1) 异步 BCD 计数器

在例 6.4.2 中曾分析过一个具有 000 ~ 100 五个连续二进制编码状态的异步计数器。如果在该电路基础上增加一级触发器，便可构成异步 BCD 计数器，如图 6.5.17 所示。为了应用的灵活性，除清零信号 CR 外，这个触发器和五状态计数器的输入、输出端均是独立引出的。

例 6.5.1 将图 6.5.17 所示电路按以下两种方式连接：① \overline{CP}_0 接计数脉冲信号；将 Q_0 与 CP_1 相连。② \overline{CP}_1 接计数脉冲信号，将 Q_3 与 \overline{CP}_0 相连。试分析它们的逻辑输出状态。

解：按①方式连接时，计数脉冲先进行二分频，然后进行五分频。从 0000 状态开始，依次分析，得到的转换表如表 6.5.7 左边所示。 $Q_3Q_2Q_1Q_0$ 输出为 8421 BCD 码，构成 8421 BCD 计



复位清零：置数：1100；计数：1101→1110→1111→0000→0001→0010；禁止计数

图 6.5.16 74LVC161 的典型时序图

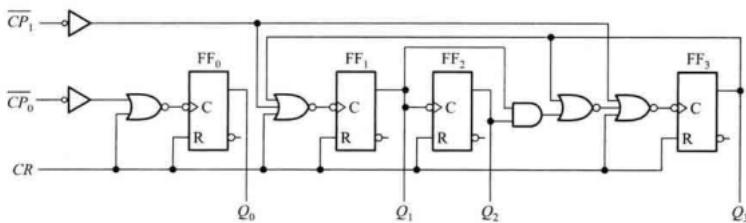


图 6.5.17 一种异步 BCD 计数器的逻辑图

数器。

按②方式连接时，计数脉冲先进行五分频，然后再二分频。分析得到的转换表列于表 6.5.7 右边。可以看出， Q_0 的权值等于 5， Q_3, Q_2, Q_1 的权值分别为 4、2、1，这种编码称为 5421 BCD 码。因此，电路构成 5421 BCD 计数器。

表 6.5.7 例 6.5.1 的两种连接方式的转换表

计数顺序	连接方式①(8421 码)				连接方式②(5421 码)			
	Q_3	Q_2	Q_1	Q_0	Q_0	Q_3	Q_2	Q_1
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	0
3	0	0	1	1	0	0	1	1
4	0	1	0	0	0	1	0	0
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

(2) 用集成计数器构成任意模数计数器

任意模数的计数器可以用厂家定型的集成计数器产品外加适当的电路连接而成。用模 m 的集成计数器构成模 n 计数器时, 如果 $m > n$, 则只需一个模 m 集成计数器; 如果 $m < n$, 则要用多个模 m 计数器来构成。下面结合例题分别介绍这两种情况的实现方法。

例 6.5.2 用 74LVC161 构成模 9 递增计数器。

解: 模 9 计数器有 9 个状态, 而 74LVC161 在计数过程中有 16 个状态。因此属于 $m > n$ 的情况。如果设法跳过多余的 7 个状态, 则可实现模 9 计数器。通常用两种方法实现, 即反馈清零法和反馈置数法。

① 反馈清零法

反馈清零法适用于有清零输入端的集成计数器。74LVC161 具有异步清零功能, 在其计数过程中, 不管它的输出处于哪一状态, 只要在异步清零输入端加一低电平, 使 $\overline{CR} = 0$, 74LVC161 的输出会立即从那个状态回到 0000 状态。清零信号消失后 ($\overline{CR} = 1$), 74LVC161 又从 0000 状态开始重新计数。

图 6.5.18(a) 所示的模 9 计数器, 就是借助 74LVC161 的异步清零功能实现的。图 6.5.18(b) 是其主循环状态图。由图可知, 74LVC161 从 0000 状态开始计数, 当第九个 CP 脉冲上升沿到达时, 输出 $Q_3Q_2Q_1Q_0 = 1001$, 通过一个与非门译码后, 反馈给 \overline{CR} 端一个清零信号, 立即使 $Q_3Q_2Q_1Q_0$ 返回到 0000 状态。此刻, 产生清零信号的条件已消失, \overline{CR} 端随之变为高电平, 74LVC161 重新从 0000 状态开始新的计数周期。这样就跳过了 1001 ~ 1111 七个状态, 构成模 9

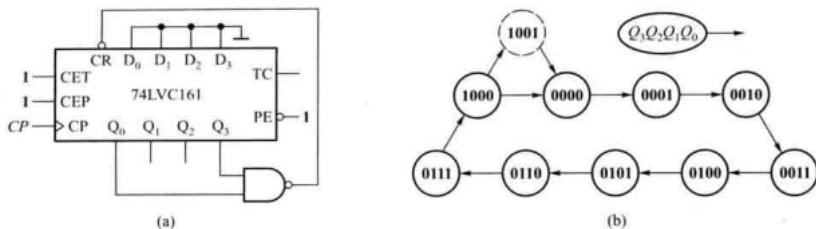


图 6.5.18 用反馈清零法将 74LVC161 接成模 9 计数器

(a) 电路图 (b) 主循环状态图

计数器。需要说明,电路是在进入 **1001** 状态后才被置成 **0000** 的,因此在主循环状态图中用虚线圈表示 **1001** 状态。虽然该状态存在的时间很短,且处于 **0000** 状态所占的时钟周期内,但是,如果 Q_0 端通过某种方式另接有其他时序电路,则应充分考虑 $Q_0=1$ 的窄脉冲是否会引起错误的逻辑输出,例如触发器被误触发等故障。

具有同步清零功能的模 m 集成计数器也可用反馈清零法构成模 n 计数器。这里不再举例,读者可自行分析两者的差异。

② 反馈置数法

反馈置数法适用于具有预置数功能的集成计数器。对于具有同步预置功能的计数器而言,在其计数过程中,可以将它输出的任何一个状态通过译码,产生一个预置控制信号反馈至预置控制端,在下一个 CP 脉冲作用后,就会把数据输入端 D_3, D_2, D_1, D_0 的状态置入计数器。预置控制信号消失后,计数器就从被置入的状态开始重新计数。

图 6.5.19(a) 和图 6.5.20 所示电路,都是借助 74LVC161 的同步预置功能,采用反馈置数法构成模 9 递增计数器的。其中图 6.5.19(a) 所示电路的接法是把输出 $Q_3Q_2Q_1Q_0 = 1000$ 的状态经译码产生预置信号 0 反馈至 \overline{PE} 端,在下一个 CP 脉冲上升沿到达时置入 **0000** 状态。图 6.5.19(b) 是图 6.5.19(a) 所示电路的主循环状态图。其中 **0001 ~ 1000** 这 8 个状态是

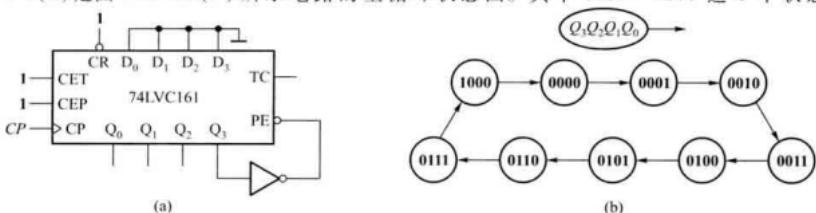


图 6.5.19 用反馈置数法将 74LVC161 接成模 9 计数器

(a) 电路图 (b) 主循环状态图

74LVC161 进行加 1 计数实现的，**0000** 是由反馈（同步）置数得到的。由此可以推知，在图 6.5.19(a) 中，反馈置数操作可在 74LVC161 计数循环状态 **0000 ~ 1111** 中的任何一个状态下进行。例如可将 $Q_3 Q_2 Q_1 Q_0 = 1111$ 状态的译码信号加至 \overline{PE} 端，这时，数据输入端应接为 $D_3 D_2 D_1 D_0 = 0111$ ，计数器将在 **0111 ~ 1111** 九个状态间循环。

图 6.5.20 所示电路的接法是将 74LVC161 计数到 **1111** 状态时产生的进位信号反相后，反馈到预置端。预置数据输入端应置成 **0111** 状态。该电路从 **0111** 状态开始加 1 计数，输入第 8 个 CP 脉冲后到达 **1111** 状态，此时 $TC = Q_3 \cdot Q_2 \cdot Q_1 \cdot Q_0 \cdot CET = 1, \overline{PE} = 0$ ，在第 9 个 CP 脉冲作用后， $Q_3 Q_2 Q_1 Q_0$ 被置成 **0111** 状态，同时使 $TC = 0, \overline{PE} = 1$ 。新的计数周期又从 **0111** 开始。

具有异步置数功能的模 m 集成计数器也可用反馈置数法构成模 n 计数器。读者可自行分析它与上述同步置数的差异。

例 6.5.3 用图 6.5.17 所示异步 BCD 计数器构成模 24 计数器。

解：运用反馈清零法实现。因为 BCD 计数器 $M=10$ ，小于 24，所以需要用两个 BCD 计数器 C_0 和 C_1 实现。先将 C_0 和 C_1 分别接成 8421 BCD 计数器，然后将它们级联，接成模 100 计数器，其中 C_1 构成十位数计数器， C_0 构成个位数计数器，其电路如图 6.5.21 所示。在此基础上，借助与门译码和计数器异步清零功能，将 C_0 的 Q_2 和 C_1 的 Q_1 分别接至与门的输入端。工作时，在第 24 个计数脉冲作用后，十位数计数器输出为 **0010**，个位数计数器输出为 **0100**（十进制数 24）， C_1 的 Q_1 与 C_0 的 Q_2 同时为 1，使与门输出高电平。它作用在计数器 C_0 和 C_1 的清零端 CR （高电平有效），使计数器立即返回到 **0000 0000** 状态。状态 **0010 0100** 仅在瞬间出现一下，计数器的有效状态为 24 个，形成模 24 计数器。

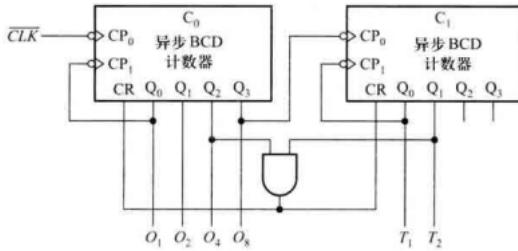


图 6.5.21 例 6.5.3 的电路图

这种连接方式可称为整体反馈清零法，其原理与 $m > n$ 时的反馈清零法相同。也可以用具有预置数据功能的集成计数器，采取整体反馈置数的方法构成模 24 计数器，其原理与 $m > n$ 时的反

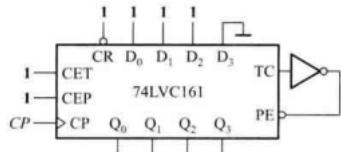


图 6.5.20 另一种反馈置数法的电路

锁置数法相似。读者可自行分析或设计。

(3) 环形计数器

① 基本环形计数器

如果将图 6.5.2 中移位寄存器的 $D_{SO}(Q_3)$ 与 D_{SI} 相连, 则构成环形计数器, 如图 6.5.22(a) 所示。若事先通过 \overline{PE} 端施加低电平脉冲, 在 4 个触发器内置入数据 $Q_3Q_2Q_1Q_0 = 0001$, 那么环形计数器在 CP 脉冲作用下, 将会有如图 6.5.22(b) 所示的 4 个状态, 于是, 电路成为模 4 计数器。图 6.5.22(c) 所示为在 4 个 CP 脉冲作用下的波形。可以看出, 这种计数器不必译码就能直接输出 4 个状态的译码信号, 亦不存在普通译码电路输出易出现的竞争-冒险现象。

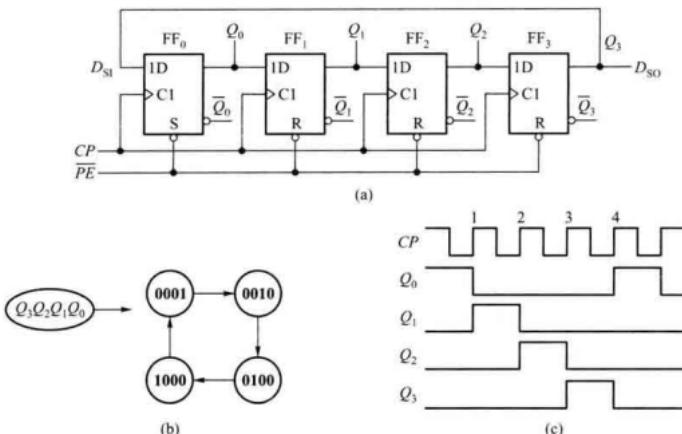


图 6.5.22 基本环形计数器的电路和状态变化

(a) 模 4 环形计数器 (b) 状态图 (c) 波形图

② 扭环形计数器

上述基本环形计数器的状态利用率不高, 4 个触发器只有 4 个计数状态。若将图 6.5.2 所示电路中的 \overline{Q}_3 与 D_{SI} 相连, 则构成扭环形计数器, 亦称为约翰逊计数器 (Johnson counter), 电路的状态将增加一倍。图 6.5.23(a) 所示是 5 个触发器构成的扭环形模 10 计数器, 图 6.5.23(b) 所示是其状态图, 表 6.5.8 列出了它的十个状态以及对各状态译码的逻辑表达式。可以看出, 它的译码电路将十分简单。由于每一次时钟脉冲触发沿到来后只有一个触发器改变状态, 所以译码输出不会出现竞争-冒险。图 6.5.23(a) 中附加的逻辑门使电路能够自校正, 请读者自行验证。

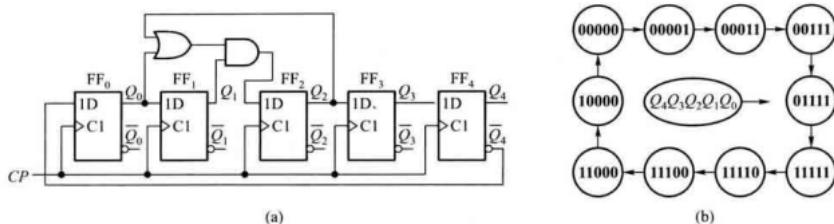


图 6.5.23 模 10 扭环形计数器

(a) 逻辑图 (b) 状态图

表 6.5.8 图 6.5.23 所示扭环形计数器的十个状态及其译码

计数顺序	Q_4	Q_3	Q_2	Q_1	Q_0	状态译码
0	0	0	0	0	0	$\bar{Q}_4 \bar{Q}_0$
1	0	0	0	0	1	$\bar{Q}_4 Q_0$
2	0	0	0	1	1	$\bar{Q}_4 Q_1$
3	0	0	1	1	1	$\bar{Q}_4 Q_2$
4	0	1	1	1	1	$\bar{Q}_4 Q_3$
5	1	1	1	1	1	$Q_4 Q_0$
6	1	1	1	1	0	$Q_4 \bar{Q}_0$
7	1	1	1	0	0	$Q_4 \bar{Q}_1$
8	1	1	0	0	0	$Q_4 \bar{Q}_2$
9	1	0	0	0	0	$Q_4 \bar{Q}_3$

74HC4017 是一种典型的模 10 扭环形计数器, 芯片中含有计数器的十个有效状态的译码电路, 从引脚可直接得到 $D_0 \sim D_9$ 无竞争-冒险的译码输出。

复习思考题

6.5.1 电平敏感的锁存器和脉冲边沿敏感的寄存器在数据寄存操作上有何不同?

6.5.2 用 D 锁存器能否构成移位寄存器或计数器? 为什么?

6.5.3 如何用双向移位寄存器实现二进制数据的 $\times 2^n$ 和 $\div 2^n$ 运算?

6.5.4 同步和异步二进制计数器各有什么特点？在需要高速计数并对计数值译码时应采用哪种计数器？

6.5.5 电子钟表中的石英晶体振荡器一般采用 32.768 kHz 的振荡频率，需要使用多少位二进制计数器分频才能得到周期为 1s 的秒信号？

6.5.6 如何用具有同步清零功能的二进制计数器采取反馈清零法来构成模 n 计数器？

6.5.7 试用 10 个二输入端与门为图 6.5.23(a) 所示电路设计十个状态的译码电路，并说明为什么它们的输出不会出现竞争-冒险。

6.6 简单的时序可编程逻辑器件 GAL

目前应用的时序可编程逻辑器件(Sequential Programmable Logic Device,SPLD)主要有 GAL、CPLD 和 FPGA 等。工程设计中，应根据实际需求的简繁，选择使用 GAL、CPLD 或 FPGA 等不同类型的时序 PLD。

本节将讨论的 GAL，其集成度在 1000 个门以下，属于简单、低密度型时序 PLD。对这种简单 PLD 的结构、工作原理和设计过程的理解，将有助于更复杂的 PLD 的原理理解和应用。有关 CPLD 和 FPGA 的详细讨论将在第 8 章进行。

6.6.1 GAL 的结构

这里以较简单的 GAL16V8 为例，说明 GAL 器件的结构。

1. 电路组成

图 6.6.1 为 GAL16V8 的逻辑结构图，它主要由以下五部分组成：

① 由 8 组与门 $A_1 \sim A_8$ 共 8×8 个与门构成的可编程与逻辑阵列，形成 64 个乘积项，32 根列线通过编程可接至其中任意与门的输入端。32 根列线连接 16 对两两互补的变量，其中 8 对接输入缓冲器的输出(列线编号为 0-1, 4-5, 8-9, 12-13, 16-17, 20-21, 24-25, 28-29)，另外 8 对接反馈/输入缓冲器的输出 $FB_1 \sim FB_8$ 。

② 8 个输出逻辑宏单元(Output Logic Macro Cell,OLMC) OLMC12 ~ OLMC19，每个单元中都包括固定的或逻辑阵列。

③ 10 个输入缓冲器，固定接信号输入端，其中 8 路(引脚 $P_2 \sim P_9$)缓冲器 $IB_2 \sim IB_9$ 输出的互补信号直接引入与逻辑阵列，而引脚 P_1 可作为系统时钟 CLK 的输入端，引脚 P_{10} 输入的信号可作为输出三态缓冲器的系统使能控制信号 OE 。

④ 8 个连接 OLMC 的输出三态缓冲器 $OB_1 \sim OB_8$ 。若三态缓冲器配置为输出高阻状态，所连接的相应引脚亦能作为输入端。

⑤ 8 个反馈/输入缓冲器 $FB_1 \sim FB_8$ ，它们的输入信号来自各 OLMC，其互补输出信号锁送到与逻辑阵列。

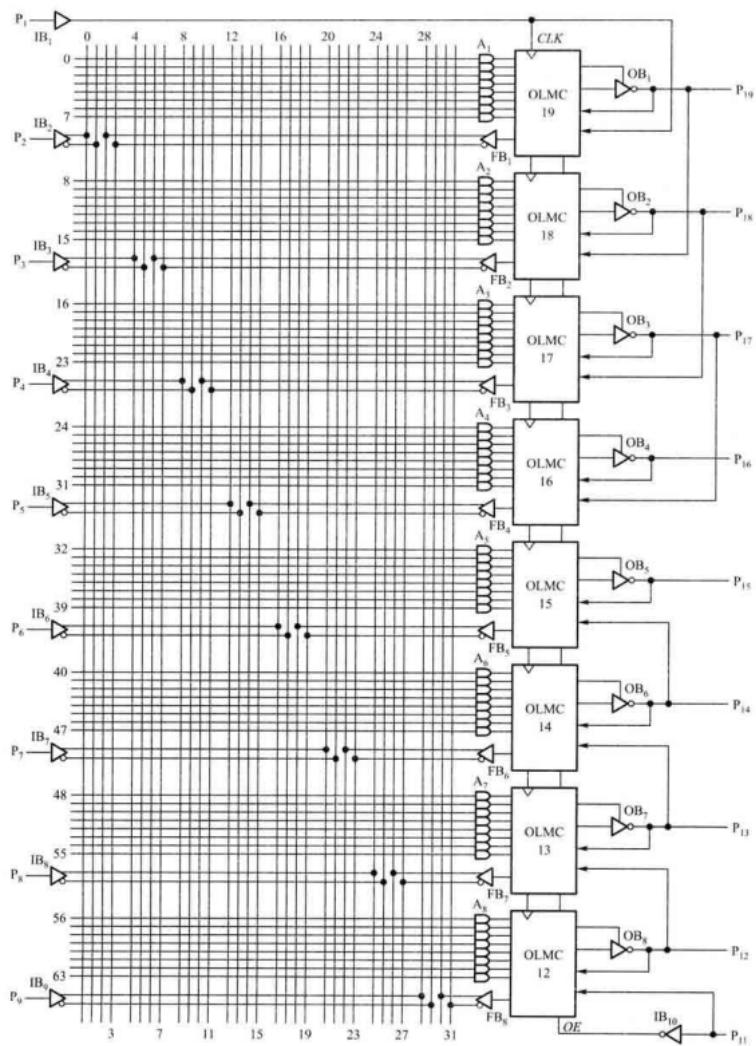


图 6.6.1 GAL16V8 的逻辑结构图

2. 与阵列编程单元

图 6.6.1 中 64 个与门的输入引线都和标有 0~31 的 32 根列线交叉，因此每个与门有 32 个输入端。在每个交叉点上都设有 E²CMOS(Electrical Erasable CMOS, E²CMOS) 编程单元。它采用了 4.5.1 节介绍的 Flotox 管。实现对乘积项(Product Term, PT)编程的原理电路如图 6.6.2 所示。图中左边是实际电路，只画出了 A、B、C 三根列线，省略了与它们互补的列线，且每根列线只带了一个编程单元。图中，T₁、T₃、T₅ 为逻辑功能管，T₂、T₄、T₆、T₇ 为 Flotox 编程管。编程时，编程电压加在 Flotox 管栅极对其进行编程(未画编程电路)；在非编程状态时，Flotox 管的栅极不加逻辑电压，列线上的逻辑电压加在 T₁、T₃、T₅ 管的栅极。

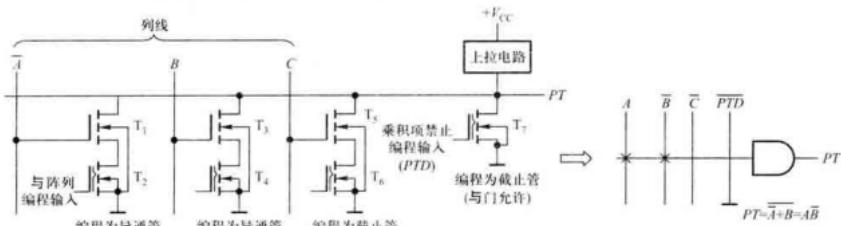


图 6.6.2 编程单元的原理电路

图 6.6.2 的电路图中，因为 T₇ 编程为截止管，不影响 PT 输出，故乘积项处于有效状态；且因 T₆ 也编程为截止管，所以列线 C 上的逻辑电平与输出 PT 无关。而 T₂ 和 T₄ 编程为导通管，则 T₁ 和 T₃ 构成或非门(PT 端有电平上拉电路)，于是得到 $PT = \overline{A+B} = A\bar{B}$ 。此例中，因为 A, \overline{B} 和 PT 之间有编程连接，所以在图 6.6.2 右边的简化逻辑图中，相应的交叉点以×标记。需要注意的是左、右两图中输入变量是互补的。由于所有引入与阵列的变量全部是成对的互补变量，所以，这样安排电路并没有增加与阵列的复杂性。当该乘积项是冗余项时，T₇ 应编程为导通管，使 $PT = 0$ ，则该乘积项输出为 0，不再影响它所驱动的或逻辑，表明它被禁止。

除了与阵列编程单元外，GAL 中的其他编程单元也都采用如图 6.6.2 中 T₇ 那样的 E²CMOS 电路，可编程为导通管，也可以编程为截止管。需要时，可用电擦除方式消除已编程的结构配置，然后再重新编程。这给时序电路的设计、调试和修改带来极大的方便。

6.6.2 GAL 中的输出逻辑宏单元

1. 从组合 PAL 到寄存器型 PAL

如前所述，时序电路由组合电路和存储单元组成。在第 4 章中已讨论过组合 PLD，若在与-或阵列和输出缓冲电路之间插入触发器，则组成如图 6.6.3 所示的电路，构成时序可编程逻辑电路。图中，触发器的 D 输入端被一组与-或阵列的输出所激励，其 Q 端通过三态缓冲门接输出引线，而 \overline{Q} 端输出经互补缓冲电路将触发器的状态信息反馈到与逻辑阵列的可编程接点。这样一

种结构,不仅使 D 触发器可以寄存与-或逻辑阵列的输出状态,而且通过含有状态信息的乘积项 PT ,为触发器提供了从现态向次态转换的条件,可方便地组成不同逻辑功能的时序电路。在早期的“寄存器型 PAL”中,就集成了多组类似于图 6.6.3 所示的电路。其中,各三态缓冲门使能端都连接在同一输出使能线上,受外部输入的使能信号 OE 控制。各触发器的时钟信号输入也都连接到同一公共时钟 CLK 上,使一组触发器能同时更新状态,构成同步时序电路。

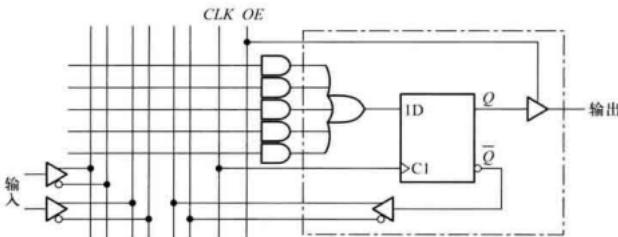


图 6.6.3 寄存器型 PAL 的输出电路

2. 在寄存器型 PAL 基础上改进的 GAL

GAL 是在寄存器型 PAL 基础上发展起来的增强型器件,其内部逻辑仍采用组合 PLD 的与-或结构,但在输出电路方面做了改进。在 GAL 中,每一组可编程的与门阵列驱动的是一个包括或门在内的输出逻辑宏单元 OLMC,它使电路的通用性更强,输出方式更灵活。GAL16V8 中 OLMC 的逻辑结构如图 6.6.4 所示。对照图 6.6.3 可以看出,该 OLMC 是在图 6.6.3 中点画线框内所示输出电路的基础上,增加了若干数据选择器 MUX 和一些门电路,电路设计者可根据需要将它们编程配置为不同结构,以实现多种结构的组合电路和时序电路。

3. OLMC 的电路结构和功能

图 6.6.4 中点画线框内所示为 OLMC 内部电路,主要由 1 个上升沿触发的 D 触发器、4 个数据选择器 MUX、一个固定的 8 输入端或门以及一些门电路组成。图中所标注的变量分为两大类:一类是与其他 OLMC 共用的变量,如 CLK 、 OE 、 AC_0 等;另一类属于该 OLMC 专用或来自相邻 OLMC 的变量,如 $AC_1(n)$ 、 $XOR(n)$ 、 $OR(n)$ 、 $D(n)$ 、 $I/O(n)$ 、 $AC_1(m)$ 、 $I/O(m)$ 等,其中括号中的 n 表示该 OLMC 的编号,而 m 则是与 n 相邻的一个 OLMC 的编号。图中,“ \oplus ”表示 E²COSMOS 编程单元,通过编程来确定相应的逻辑变量为 0 还是 1。一旦对芯片编程完毕,它们的逻辑值就不可更改,除非将芯片整体擦除后重新编程。

(1) 输入电路

一个 OLMC 的或门与同一芯片中其他 OLMC 的或门构成了 GAL 的或逻辑阵列,再和输入的与门阵列一起构成基本的与-或组合逻辑阵列。OLMC 中或门的第 2~8 输入端分别直接和与阵列的一个与门输出端相连,即接收一个乘积项,而第 1 输入端接乘积项选择器 PT MUX 的输出。两个由编程单元确定逻辑值的变量 AC_0 和 $AC_1(n)$ 控制着 PT MUX 和三态控制选择器 TS MUX。

除了 OLMC12 和 OLMC19 两个输出逻辑宏单元有些特殊外,当编程使 $AC_0 \cdot AC_1(n) = 1$ 时,第 1 乘积项通过 PT MUX 连接到或门的第 1 输入端,此时或门有 8 个乘积项输入,否则或门的第 1 输入端接 0,第 1 乘积项对或门的输出没有影响。在 $AC_0 = AC_1(n) = 1$ 时,第 1 乘积项将通过 TS MUX 接至输出三态缓冲器的使能端,当该乘积项为 1 时,三态缓冲器使能。

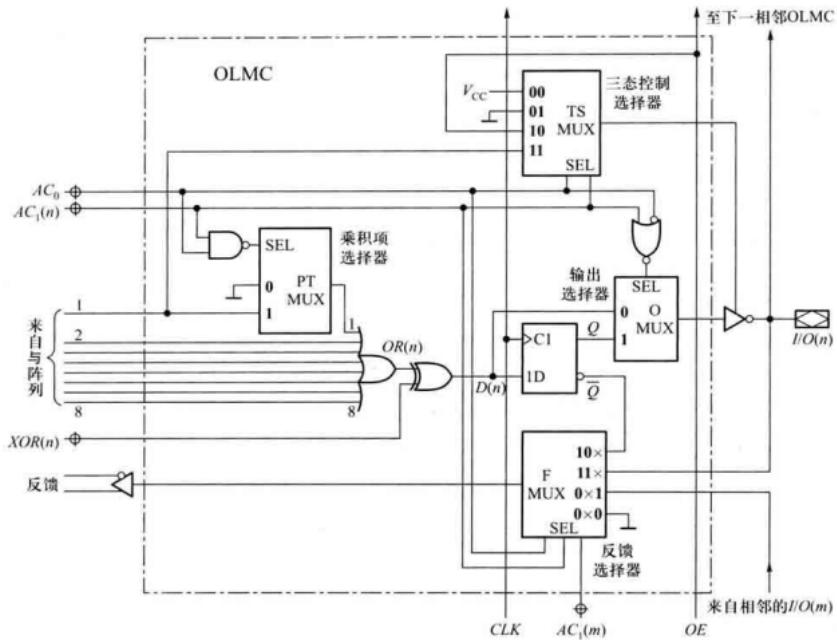


图 6.6.4 GAL16V8 中的 OLMC

注：⊕表示 E²CMOS 编程单元，×表示无关位

(2) 原变量/非变量输出控制电路

异或门将或门输出信号 $OR(n)$ 与 $XOR(n)$ 进行异或运算后, 激励 D 触发器或直接输出。

$XOR(n)$ 的逻辑值由编程单元确定。对于异或门有 $D(n) = OR(n) \overline{XOR(n)} + \overline{OR(n)} XOR(n)$, 若 $XOR(n) = 0$, 则 $D(n) = OR(n)$, 若 $XOR(n) = 1$, 则 $D(n) = \overline{OR(n)}$ 。可见, 通过对 $XOR(n)$ 编程, 便可控制输出信号等于或门的输出, 或者为或门输出的非。

(3) 输出电路

输出选择器 O MUX 是 2 选 1 数据选择器, 用于选择输出信号的来源。当 AC_0 和 $AC_1(n)$ 满

是 $\overline{AC_0+AC_1(n)}=0$ 时, 选择异或门输出, 其信号直接送达输出缓冲器, 意味着 OLMC 是组合电路输出; 反之, 当 $\overline{AC_0+AC_1(n)}=1$ 时, O MUX 选择 D 触发器的 Q 端, 异或门的输出信号必须经过 D 触发器寄存, 才到达输出缓冲器, 即构成时序电路输出。

输出三态缓冲器的控制信号来源由三态选择器 TS MUX 选择, 它是一个 4 选 1 数据选择器, 4 个数据输入端受 AC_0 和 $AC_1(n)$ 的控制, 可分别选择 V_{cc} 、地、外部输入 OE 或内部与门阵列的第一乘积项作为输出三态缓冲器的控制信号, 如表 6.6.1 所示。

表 6.6.1 TS MUX 的控制功能

控制信号		TS MUX 的输出	输出三态缓冲器工作状态
AC_0	$AC_1(n)$		
0	0	1(V_{cc})	使能
0	1	0(地)	高阻
1	0	OE	$OE=1$: 使能 $OE=0$: 高阻
1	1	第 1 乘积项	第 1 乘积项 = 1: 使能 第 1 乘积项 = 0: 高阻

(4) 反馈电路

反馈电路包括反馈选择器 F MUX 和反馈/输入缓冲器, 它将不同来源的信号馈送到与门阵列。F MUX 是一个 4 选 1 数据选择器, 根据 3 个控制信号 AC_0 、 $AC_1(n)$ 和 $AC_1(m)$ 的编程逻辑值, 分别选择来自地(逻辑 0)、相邻 OLMC 的输出 $I/O(m)$ 、本级 OLMC 输出 $I/O(n)$ 或本级 D 触发器的 \bar{Q} 信号送至与门阵列。上述 3 个控制信号的每项选择编码中, 都包含了一位逻辑值可 0、可 1 的无关位。

下面的例题分别列举了一个组合电路和一个时序电路的 OLMC 配置实例。

例 6.6.1 将编程单元 AC_0 、 $AC_1(n)$ 编程为 ① $AC_0=AC_1(n)=1$ 和 ② $AC_0=1$ 、 $AC_1(n)=0$, 试画出编程后 OLMC 的功能逻辑图。

解: ① 当 $AC_0=AC_1(n)=1$ 时, PT MUX 选择接地, 第 1 个乘积项通过 TS MUX 接输出三态缓冲器的使能端, O MUX 将输出缓冲器直接与异或门输出相连, F MUX 将 $I/O(n)$ 接至反馈/输入缓冲器。得到的功能逻辑图如图 6.6.5 所示。此时 OLMC 配置为组合电路。

② 当 $AC_0=1$ 、 $AC_1(n)=0$ 时, PT MUX 选择第 1 个乘积项接入或门, TS MUX 选择 OE 控制输出缓冲器使能, O MUX 选择 D 触发器的 Q 输出, F MUX 选择将触发器的 \bar{Q} 接至反馈/输入缓冲器, 得到的功能逻辑图如图 6.6.6 所示。芯片中若有几个同样配置的 OLMC, 它们共用一个时钟信号 CLK , 可构成同步时序电路。

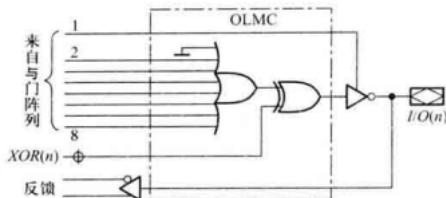


图 6.6.5 OLMC 配置为组合电路

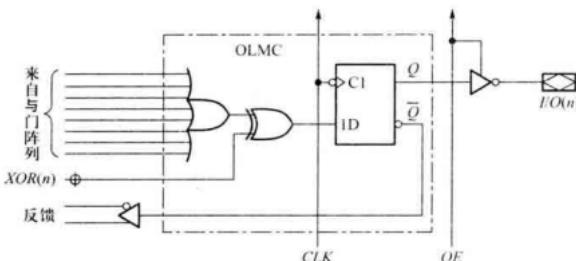


图 6.6.6 OLMC 配置为时序电路

6.6.3 GAL 的结构控制字

如前所述,通过编程可将 OLMC 配置成各种不同的逻辑功能。由图 6.6.4 看出,每个 OLMC 有 2 个编程单元 $AC_i(n)$ 和 $XOR(n)$,另外有一个全局编程单元 AC_0 。而 GAL16V8 共有 8 个 OLMC(参见图 6.6.1),所以共有 8 个 $AC_i(n)$ 编程单元和 8 个 $XOR(n)$ 编程单元。另外, GAL16V8 中的 64 个与门每一个都有一个乘积项禁止编程单元(参见图 6.6.2),共 64 个。将这些编程单元组合在一起,构成所谓的结构控制字,每个编程单元占据其中 1 位。GAL16V8 的结构控制字共有 82 位,它们的定义如图 6.6.7 所示。图中 $XOR(n)$ 和 $AC_i(n)$ 字段下面的数字为 n 的值,表示所控制的 OLMC 编号。

除前面已介绍的编程单元外,结构控制字中还有一个关键的同步控制单元 SYN 。当 $SYN=1$ 时,芯片中所有的 OLMC 均配置为组合输出模式;若 $SYN=0$,则 OLMC 可配置为寄存器输出模式(即时序电路模式),亦可配置为组合输出,但至少有一个 OLMC 配置为寄存器输出模式。此外,对于 OLMC(12)和 OLMC(19)两个宏单元, \overline{SYN} 代替 AC_0 , SYN 代替 $AC_i(m)$,这使得 $SYN=1$ 时, P_1 脚和 P_{11} 脚成为输入端,将输入信号通过这两个 OLMC 中的 F MUX 和反馈/输入缓冲器引入与逻辑阵列,使这两引脚的逻辑功能与 $P_2 \sim P_9$ 引脚相同。当初这种安排是为了兼容早期的很多

PAL 器件。

通过对结构控制字的编程,便可确定 GAL 的工作模式。

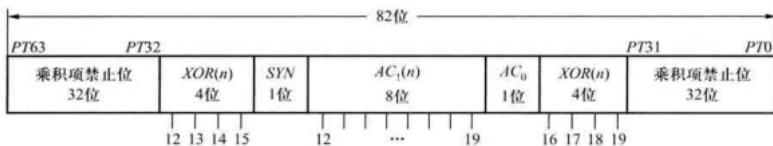


图 6.6.7 GAL16V8 的结构控制字

除了结构控制字外,GAL16V8 内部还附有 1 个加密编程单元和 64 位电子标签编程单元。前者被编程后就禁止对门阵列作进一步的编程或读出验证,以防止未经允许的改写和保护电路的知识产权。加密编程单元只有在全部编程单元被整体擦除后才被擦除。但电子标签记录的信息不受加密单元状态的影响,加密后仍随时可以读出。电子标签可供产品制造者记录各种识别信息,如产品制造商识别码、编程日期、线路形式识别码、产品序列码等,以便识别型号相同而编程后功能不同的 GAL 芯片,以及进行产品的质量跟踪。另外,还有一个整体擦除编程单元,在编程周期对该单元执行擦除命令,则将前述全部编程单元整体擦除,包括电子标签和加密单元,芯片返回到初始状态。

为了实现对 GAL16V8 的编程和回读验证编程状态,芯片中设置有编程硬件电路。对 GAL 的编程需要在开发系统(包括硬件编程器和软件)的控制下完成。其开发过程将在 8.3 节统一介绍。

从上述有关讨论可以看出,GAL16V8 虽然比此前的 PAL 等可编程器件的功能有很大提高,但是,它的应用仍受到 OLMC 有限几种工作模式的限制,而且其电路规模仅相当于中规模集成电路芯片,当需要实现更大规模的数字系统时,应当选用集成度更高、编程功能更强的可编程逻辑器件,如 CPLD、FPGA 等。

复习思考题

- 6.6.1 GAL 中的输出三态缓冲器可由哪几个信号控制?
- 6.6.2 在 GAL 的 OLMC 中可以将哪几个信号反馈到与门阵列中?
- 6.6.3 在时序电路设计中,GAL 的应用受到哪些局限?

6.7 用 Verilog HDL 描述时序逻辑电路

在 4.6 节和 5.6 节分别介绍了用 Verilog 描述组合电路和触发器的方法,本节将通过一些例

题讨论时序逻辑电路的行为描述方法。

6.7.1 移位寄存器的 Verilog 建模

例 6.7.1 根据 6.5.4 所示的功能表, 试用 Verilog 语言描述一个 4 位双向移位寄存器的行为。

解:

```
module shift74x194_beh( //Verilog 2001,2005 syntax
    input S1,S0,           //选择输入端口声明
    input Dsl,Dsr,         //串行数据输入
    input CP,CR,           //时钟和清零输入
    input[3:0]D,           //并行数据输入
    output reg[3:0]Q        //输出端口及变量的数据类型声明
);
    always @ ( posedge CP, negedge CR) //Verilog 2001,2005 syntax
        if( ~CR) Q<=4'b0000;          //实现异步清零功能
        else
            case( |S1,S0| )
                2'b00:Q<=Q;           //输出保持不变
                2'b01:Q<=|Q[2:0],Dsr|;   //右移
                2'b10:Q<=|Dsl,Q[3:1]|;   //左移
                2'b11:Q<=D;             //并行置数
            endcase
endmodule
```

程序首先声明了模块的输入、输出端口变量, 然后使用 **always** 语句结构描述了模块完成的 5 种逻辑功能: 异步置零、同步置数、左移、右移和保持原状态不变。其功能与图 6.5.6 所示的 74HC/HCT194 类似。

当清零信号 CR 跳变到低电平时, 寄存器的输出被异步置 0; 当 CR=1 时, 与时钟信号有关的 4 种功能由 **case** 语句中的两个选择输入信号 S1、S0 决定(在 **case** 后面 S1、S0 被拼接成两位矢量)。移位由串行输入和 3 个触发器的输出拼接起来进行描述, 例如:

$$Q<=|Dsl,Q[3:1]|;$$

说明了左移操作, 即在时钟信号 CP 上升沿到来时, 将左移输入端 Dsl 的数据直接传给输出 Q[3], 而 CP 上升沿到来之前触发器输出端的数据(现态)左移 1 位, Q[3:1] 传给 Q[2:0](即 Q[3]→Q[2], Q[2]→Q[1], Q[1]→Q[0]), 于是, 完成将数据左移 1 位的操作。

注意, 例 6.7.1 中所描述的右移和左移方向与图 6.5.6 中的逻辑图一致, 与计算机原理中的规定相反。

6.7.2 计数器的 Verilog 建模

下面介绍同步二进制计数器、异步二进制计数器和非二进制计数器的 Verilog 建模。

1. 同步二进制计数器

例 6.7.2 根据表 6.5.6 所示的功能表,试用 Verilog 语言描述一个 4 位计数器的行为。

解: 4 位计数器的 Verilog 语言描述如下所示。在该模块中混合使用了 **assign** 语句和 **always** 语句, **assign** 语句描述了组合电路中由与门产生的使能控制信号 CE(中间节点)和进位输出信号 TC, 当计数器计数到最大值 15 时, TC=1。

always 语句描述了计数器的逻辑功能,当 CR 信号跳变到低电平(由 **negedge** CR 描述)时,计数器的输出被置 0;否则,当 CR=1 时,在 CP 上升沿作用下,完成其他三种功能:同步置数、加 1 计数和保持原有状态不变。注意, **if** 语句隐含着各个控制信号 CR、PE、CE 操作的优先级别。逻辑综合工具根据该例的行为模型将产生图 6.5.13 所示的电路。

```

module counter74x161_beh(          //Verilog 2001,2005 syntax
    input CEP,CET,PE,CP,CR,        //输入端口声明
    input[3:0]D,                  //并行数据输入
    output TC,                   //进位输出
    output reg[3:0]Q            //数据输出端口及变量的数据类型声明
);
wire CE;                      //中间变量声明
assign CE=CEP & CET;           //CE=1 时,计数器计数
assign TC=CET & PE & (Q==4'b1111); //产生进位输出信号
always @ ( posedge CP, negedge CR) //Verilog 2001,2005 syntax
    if( ~CR) Q<=4'b0000;         //实现异步清零功能
    else if( ~PE) Q<=D;          //PE=0,同步装入输入数据
    else if( CE) Q<=Q+1'b1;     //加 1 计数
    else Q<=Q;                  //输出保持不变
endmodule

```

例 6.7.3 试用 Verilog 语言描述一个带使能端和同步置数端的通用可逆计数器的行为。

解: 一个通用的可逆计数器 Verilog 代码如下所示。其中参数 n(默认设为 4)用来设置二进制计数器的位数。当 Up_down=1 时,整型变量 direction 为 1,计数器为递增计数;Up_down=0 时,direction 为 -1,计数器为递减计数。

```

module updowncount_beh(          //Verilog 2001,2005 syntax
    parameter n=4
)

```

```

(
  input Load, Up_down, En, CP,           //输入端口声明
  input[n-1:0] D,                      //并行数据输入端口声明
  output reg[n-1:0] Q                 //数据输出端口及变量的数据类型声明
);
integer direction;                   //声明一个整型变量 direction
always @ ( posedge CP)
begin
  if( Up_down)
    direction<=1;                     //Up_down=1, 整型变量 direction 为 1
  else
    direction<=-1;                   //Up_down=0, 整型变量 direction 为 -1
  if( Load)
    Q<=D;                          //Load=1, 同步装入输入数据
  else if( En)
    Q<=Q+direction;                //实现加 1 或减 1 计数功能
  else
    Q<=Q;                          //输出保持不变
end
endmodule

```

2. 异步二进制计数器

例 6.7.4 根据图 6.5.7 所示原理图, 试用 Verilog 语言描述一个 4 位异步二进制计数器的行为。

解: 异步计数器的结构化描述如下。第一个模块通过 4 次调用第二个模块完成计数功能, 作为底层的第二个模块是带有异步置零功能的 D 触发器。

在第一个模块中, 第 1 个触发器 FF0 的时钟信号接外部输入 CP 端, 其输出 Q0 经反相(用 $\sim Q0$ 表示)后与 D 输入相连(在 FF0 中用 $\sim Q0$ 取代 D), 构成 T' 触发器。第 2 个触发器 FF1 的时钟信号接前一个触发器的输出端(用 Q0 取代 CP), 其输出 Q1 经反相后与 D 输入相连(在 FF1 中用 $\sim Q1$ 取代 D)。类似地, 将 4 个触发器级联在一起构成异步二进制计数器, 其原理图与图 6.5.7 类似。

```

module ripplecounter(
  output Q0, Q1, Q2, Q3, //输出端口声明
  input CP, CR           //输入端口声明
);
  D_FF FF0( Q0, ~Q0, CP, ~CR ); //实例化 D 触发器
  D_FF FF1( Q1, ~Q1, Q0, ~CR );

```

```

D_FF FF2( Q2, ~Q2, Q1, ~CR);
D_FF FF3( Q3, ~Q3, Q2, ~CR);
endmodule

module D_FF( output reg Q, input D, CP, Rd );//D 触发器模块
  always @ ( negedge CP, negedge Rd )
    if( ~Rd) Q<=1'b0; //异步清零
    else Q<=D; //Rd=1,将输入数据送到输出端口
endmodule

```

3. 非二进制计数器

例 6.7.5 试用 Verilog 语言描述一个带有使能端和异步清零端的同步模 10 计数器。

解：同步模 10 计数器的 Verilog 代码如下。当清零信号 CR 跳变到低电平(由 negedge CR 描述)时,计数器的输出被置 0;否则,当 CR=1,且使能信号 CE=1 时,在 CP 上升沿作用下,若计数值小于 9,则计数器的值加 1;若计数值大于或者等于 9,计数器的输出被置零。当 CR=1,但 CE=0 时,计数器保持原来的状态不变。注意,电路的功能描述与具体的硬件电路结构是无关的。

```

module m10_counter(          //Verilog 2001,2005 syntax
  output reg[3:0] Q,          //输出端口及变量的数据类型声明
  input CE,CP,CR             //输入端口声明
);
  always @ ( posedge CP, negedge CR)
    if( ~CR) Q=4'b0000; //异步清零
    else if( CE)           //CE=1,由下面的 if-else 语句完成模 10 计数
      begin
        if( Q>=4'b1001)
          Q<=4'b0000;
        else
          Q<=Q+1'b1;
      end
    else
      Q<=Q; //输出保持不变(默认情况)
endmodule

```

4. 分频器

例 6.7.6 假设有一个 50 MHz 时钟信号源,试用 Verilog 设计一个分频电路,以产生 1 Hz 的秒脉冲输出,要求输出信号的占空比为 50%。

解：设计一个模数为 $25 * 10^6$ 的二进制递增计数器,其计数范围是 0 ~ 24999999,每当计数器计到最大值时,输出信号翻转一次,即可产生 1 Hz 的秒脉冲。其程序如下。

```

module Divider50MHz(
parameter CLK_Freq=50000000,           //50MHz 时钟输入
parameter OUT_Freq=1,                   //1 Hz 时钟输出
)
(
input      CR,CLK_50M,                //输入端口声明
output reg CLK_1HzOut                 //输出端口及变量的数据类型声明
);
reg[24:0] Count_DIV;                  //内部节点
always @ ( posedge CLK_50M or negedge CR)
begin
if( ! CR) begin
    CLK_1HzOut<=0;                  //输出信号被异步清零
    Count_DIV<=0;                  //分频器的输出被异步清零
end
else begin
    if( Count_DIV<( CLK_Freq/2 * OUT_Freq ) )
        Count_DIV<=Count_DIV+1'b1;   //分频计数器增1 计数
    else begin
        Count_DIV     <=0;          //分频器的输出被清零
        CLK_1HzOut   <= ~ CLK_1HzOut; //输出信号取反
    end
end
end
endmodule

```

6.7.3 状态图的 Verilog 建模

描述状态图的方法很多,下面通过两个例题进行说明。

例 6.7.7 试用 Verilog 语言描述图 6.7.1 所示状态图的行为。

解: 图 6.7.1 所示状态图的行为描述如下。程序首先对电路的输入、输出、时钟以及清零信号进行声明,对保存电路状态值的触发器用标识符 state(中间节点) 进行声明,并使用参数定义语句 **parameter** 定义了电路的四种状态,即 $S0 = 2'b00$ 、 $S1 = 2'b01$ 、 $S2 = 2'b10$ 和 $S3 = 2'b11$ 。

接着,用一个 **always** 语句描述时钟控制的状态转换以及电路的次态逻辑。在异步复位信号 CR 为 0 时,电路进入初始状态 $S0$;在没有复位信号且时钟信号 CP 上升沿到来时,根据电路当前状态和输入信号,电路转换到下一个状态,这是用 **case** 语句进行描述的,在 **case** 语句的多个分支中包含了每个状态的行为。

最后用 **assign** 语句描述了电路的输出,该电路的输出与电路现态相同。注意,在 CP 信号的两个上升沿之间,输入信号 Data 的变化对输出信号是没有影响的。

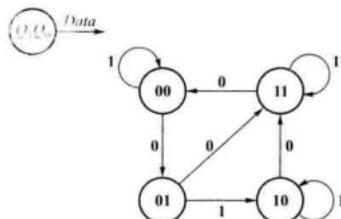


图 6.7.1 一个穆尔型时序电路的状态图

```

module MooreMachine(           //Verilog 2001,2005 syntax
input Data,CP,CR,             //输入端口声明
output[1:0] Q                //输出端口声明
);
    reg[1:0] state; //声明两个寄存器变量 state[1] 和 state[0],记忆电路现态
    parameter S0=2'b00,S1=2'b01,S2=2'b10,S3=2'b11; //电路的状态定义
    always @ ( posedge CP, negedge CR)
        begin
            if( ~ CR) state<=S0; //CR = 0,让电路进入初始状态 S0
            else
                case( state )           //电路状态转换
                    S0 : if( ~ Data) state<=S1 ;
                    S1 : if( Data) state<=S2 ; else state<=S3 ;
                    S2 : if( ~ Data) state<=S3 ;
                    S3 : if( ~ Data) state<=S0 ;
                endcase
        end
        assign Q=state; //电路的输出与电路的现态相同
    endmodule

```

上例描述状态图的方法仅仅用了一条 **always** 语句,看起来非常简单,但电路结构却不够清晰,电路的现态和次态均用变量 state 表示,并且该方法对米利型时序电路不适用,因为米利型电路输出可能直接受到与时钟无任何关系的输入信号的作用。建议用两条 **always** 语句对状态图建模,下面举例说明。

例 6.7.8 图 6.3.7 是用于检测连续输入序列 110 的状态图,试用 Verilog 描述该状态图所

表示的行为。

解：该状态图的 Verilog 描述如下。

```

module Mealy_sequence_detector( //Verilog 2001,2005 syntax
    input A,CP,CR,           //输入端口声明
    output reg Y           //输出端口及变量的数据类型声明
);
    reg[1:0] current_state,next_state;//中间变量声明
    parameter S0=2'b00,S1=2'b01,S2=2'b11;//定义状态
    always @ ( negedge CP,negedge CR)//触发器状态转换
    begin
        if( ~CR) current_state<=S0; //让电路进入初始化状态 S0
        else current_state<=next_state;//电路发生状态转换(相当于 D 触发器 Q=D)
    end
    always @ ( current_state,A) //输出逻辑和次态逻辑描述
    begin
        case( current_state)
            Y=0;           //电路输出
            S0;begin next_state=(A==1)? S1:S0;end
            S1;begin next_state=(A==1)? S2:S0;end
            S2;if(A==1)
                begin next_state<=S2;end
            else
                begin Y=1;next_state<=S0;end
            default;begin next_state<=S0;end
        endcase
    end
endmodule

```

电路功能描述使用了两条并行执行的 **always** 语句,通过公用变量(**next_state**)相互交换信息。第一条 **always** 语句使用边沿触发事件描述电路的同步时序逻辑部分,第二条 **always** 语句使用电平敏感事件描述组合逻辑部分。

第一条 **always** 语句说明了异步复位到初始状态 S0 和同步时钟控制的操作,语句

current_state<=next_state;

仅在时钟 CP 的下降沿被执行,这意味着第二条 **always** 语句内部 **next_state** 值的变化会在时钟 CP 下降沿到来时被传送给 **current_state**。

第二条 **always** 语句把现态 **current_state** 和输入数据 Data 作为敏感变量,只要其中的任何一个变量发生变化,就会执行顺序语句块内部的 **case** 语句,跟在 **case** 语句后面的各分支项说明了

图 6.3.7 中状态的转换以及输出信号。

注意,在 Mealy 型电路中,当电路处于任何给定的状态时,如果输入信号 A 发生变化,则输出信号 Y 也会跟着变化。

综上所述,利用 Verilog HDL 语言描述状态图主要包含四部分内容:

(1) 利用参数定义语句 **parameter** 描述状态机中各个状态的名称,并指定状态编码。例如,对序列检测器的状态分配可以使用最简单的自然二进制码,其描述如下:

```
parameter S0=2'b00,S1=2'b01,S2=2'b10,S3=2'b11;
```

或者, **parameter[1:0]** S0=2'b00,S1=2'b01,S2=2'b10,S3=2'b11;

注意,在语法上使用 $S2=3$ 的形式定义状态也是正确的,但存储“3”这个整数至少要使用 32 位寄存器,而存储 2^3 只需要两位寄存器,所以例题中使用的定义方式更好一些。上面第二种方式的定义中,明确地指出使用两个状态触发器,对逻辑综合更为有利。

(2) 使用 **always** 语句描述时钟控制的同步时序逻辑部分,实现电路状态存储。

(3) 使用 **always** 语句和 **case** 语句(也可以采用 **if-else** 等价语句)描述电路的组合逻辑部分,实现状态转换逻辑。

(4) 描述状态机的输出逻辑。

6.7.4 数字钟的 Verilog 建模

例 6.7.9 设计一个具有时、分、秒计时的数字钟电路,按 24 小时制计时。要求:

- (1) 输出时、分、秒的 8421BCD 码,计时输入脉冲频率为 1Hz;
- (2) 具有分、时校正功能,校正输入脉冲频率为 1Hz;
- (3) 采用分层次、分模块的方法,用 Verilog 语言进行设计。

解:(1) 设计分析

数字钟的组成框图如图 6.7.2 所示,它由两个 60 进制计数器、1 个 24 进制计数器和两个 2 选 1 选择器共 5 个模块构成,三个计数器公用一个时钟信号 CP,为同步 8421BCD 码输出的计数器,在秒、分、小时计数器的输出端接上虚线框内的译码显示电路,即可显示出数码(即时间)。

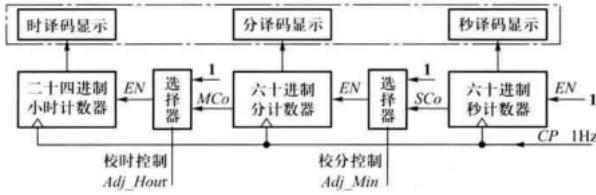


图 6.7.2 数字钟组成框图

图中两个选择器分别用于选择分钟计数器和小时计数器的使能控制信号。对时间进行校正

时，在控制端(Adj_Hour , Adj_Min)的作用下，使能信号接高电平，此时每来1个时钟信号，计数器加1计数，从而实现对小时和分钟的校正。正常计时时，使能信号来自于低位计数器的输出，即秒钟计数器计到59秒时，产生输出信号($SCo=1$)使分钟计数器加1，分钟、秒钟计数器同时计到最大值(59分59秒)时，产生输出信号($MCo=1$)使小时计数器加1。

(2) 逻辑设计

实现上述功能的 Verilog HDL 程序如下。整个程序分为两个层次 4 个模块，其层次结构如图 6.7.3 所示。底层由 3 个模块组成，六进制计数器模块 (counter6.v)、十进制计数器模块 (counter10.v) 和 24 进制计数器模块 (counter24.v)，顶层有 1 个模块 (top_clock.v)，顶层模块调用底层的 3 个模块完成数字钟的计时功能。其中，底层的六进制计数器模块和十进制计数器模块分别被调用两次，构成六十进制的秒钟计数器和分钟计数器。

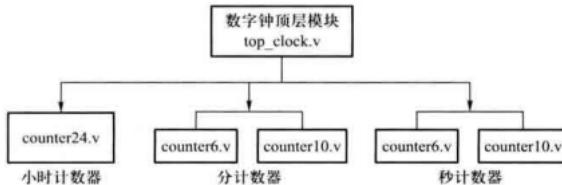


图 6.7.3 数字钟程序的层次结构图

```

//*****时钟顶层模块(top_clock.v)*****
module top_clock(          //Verilog 2001,2005 module port syntax
    input  CP,nCR,EN,Adj_Min,Adj_Hour, //输入端口变量声明
    output[ 7:0 ] Hour,Minute,Second //输出端口变量声明
);
    supply1 Vdd;
    wire SecH_EN,MinL_EN,MinH_EN,Hour_EN; //中间节点变量声明
    //60进制秒计数器:调用10进制和6进制底层模块构成
    counter10 U1(Second[ 3:0 ],nCR,EN,CP);      //秒计数器个位
    counter6 U2(Second[ 7:4 ],nCR,SecH_EN,CP); //秒计数器十位
    //产生分钟计数器使能信号。Adj_Min=1,校正分钟;Adj_Min=0,分钟正常计时
    assign MinL_EN = Adj_Min? Vdd: (Second == 8'h59);
    assign MinH_EN = ( Adj_Min && ( Minute[ 3:0 ] == 4'h9 ) ) || ( Minute[ 3:0 ] == 4'h9 ) && ( Second == 8'h59 );
    //60进制分钟计数器:调用10进制和6进制底层模块构成
    counter10 U3( Minute[ 3:0 ],nCR,MinL_EN,CP); //分计数器个位
    counter6 U4( Minute[ 7:4 ],nCR,MinH_EN,CP); //分计数器十位
    //产生小时计数器使能信号。Adj_Hour=1,校正小时;Adj_Hour=0,小时正常计时

```

```

assign Hour_EN = Adj_Hour? Vdd:(( Minute == 8'h59) && (Second == 8'h59));
//24 进制小时计数器:调用 24 进制底层模块构成
counter24 U5(Hour[7:4],Hour[3:0],nCR,Hour_EN,CP); //小时计数器
endmodule

//***** counter10.v(BCD:0~9) *****
module counter10(
    input CP,nCR,EN,
    output reg[3:0]Q
);
    always @ (posedge CP, negedge nCR)
    begin
        if( ~nCR) Q<=4'b0000; //nCR=0,计数器被异步清零
        else if( ~EN) Q<=Q; //EN=0,暂停计数
        else if( Q == 4'b1001) Q<=4'b0000;
        else Q<=Q+1'b1; //计数器增1计数
    end
endmodule

//***** counter6.v(BCD:0~5) *****
module counter6(
    input CP,nCR,EN,
    output reg[3:0]Q
);
    always @ (posedge CP, negedge nCR)
    begin
        if( ~nCR) Q<=4'b0000; //nCR=0,计数器被异步清零
        else if( ~EN) Q<=Q; //EN=0,暂停计数
        else if( Q == 4'b0101) Q<=4'b0000;
        else Q<=Q+1'b1; //计数器增1计数
    end
endmodule

//***** counter24.v(BCD:0~23) *****
module counter24(
    input CP,nCR,EN,
    output reg[3:0]CntH,CntL,
);

```

```

always @ ( posedge CP, negedge nCR)
begin
    if( ~nCR)
        | CntH,CntL| <= 8'h00;
    else if( ~EN)
        | CntH,CntL| <= | CntH,CntL| ;
    else if( ( CntH>2 ) || ( CntL>9 ) || ( ( CntH == 2 ) && ( CntL>=3 ) ) )
        | CntH,CntL| <= 8'h00;//出错处理
    else if( ( CntH == 2 ) && ( CntL<3 ) )           //小时(20~23)的计数
        begin   CntH<=CntH;   CntL<=CntL+1'b1;end
    else if( CntL == 9 ) //小时十位计数
        begin   CntH<=CntH+1'b1;   CntL<=4'b0000;end
    else               //小时个位计数
        begin   CntH<=CntH;   CntL<=CntL+1'b1;end
    end
endmodule

```

复习思考题

6.7.1 试用 Verilog 描述图 6.5.1 所示的寄存器。

6.7.2 异步计数器和同步计数器的描述方式有何不同?

6.7.3 状态图的建模方法与前述计数器的建模有何区别? 试用状态图的建模方法描述一个 4 位同步二进制递增计数器。

小 结

- 时序电路一般由组合电路和存储电路两部分构成。它们在任一时刻的输出不仅是当前输入信号的函数,而且还与电路原来的状态有关。时序电路可分为同步和异步两大类。逻辑方程组、转换表、状态表、状态图和时序图从不同方面表达了时序电路的逻辑功能,是分析和设计时序电路的主要工具。

- 同步时序电路是目前广泛应用的时序电路,是本章重点讨论的内容。同步时序电路的分析,首先按照给定电路列出各逻辑方程组、进而列出转换表或状态表、画出状态图和时序图,最后分析得到电路的逻辑功能。同步时序电路的设计,首先根据逻辑功能的需求,推导出原始状态图或原始状态表,有必要时需进行状态化简,继而对状态进行编码得到转换表,然后根据转换表导出激励方程组和输出方程组,最后画出逻辑图完成设计任务。

- 时序电路的功能、结构和种类繁多。本章仅对寄存器和计数器等几种典型的时序电路模块进行了较详细的讨论。应用这些电路模块,能设计出各种不同功能的电子系统。这些模块的

内部电路结构可作为设计其他逻辑电路的范例,也可作为初学者使用 HDL 以抽象方式描述时序电路时的实际映象。

• 本章分别在 6.6 节和 6.7 节介绍了简单时序可编程逻辑器件 GAL 和用 Verilog 描述时序电路的方法,它们是电子技术和计算机技术以及 EDA 技术高度发展的结果。这两节所讨论的方法和器件,尚需参考有关软件的详细文献资料并配合上机实验才能达到实际应用的程度。

习 题

6.1 时序逻辑电路的基本概念

6.1.1 已知一时序电路的状态表如表题 6.1.1 所示, A 为输入信号, 试作出相应状态图。

表题 6.1.1

S^n	S^{n+1}/Z	
	$A = 0$	$A = 1$
a	$d/1$	$b/0$
b	$d/1$	$c/0$
c	$d/1$	$a/0$
d	$b/1$	$c/0$

6.1.2 已知状态表如表题 6.1.2 所示, 输入为 $A_1 A_0$, 试作出相应状态图。

表题 6.1.2

S^n	S^{n+1}/Z			
	$A_1 A_0 = 00$	$A_1 A_0 = 01$	$A_1 A_0 = 10$	$A_1 A_0 = 11$
S_0	$S_0/0$	$S_1/0$	$S_3/0$	$S_2/1$
S_1	$S_1/0$	$S_2/1$	$S_1/1$	$S_0/0$
S_2	$S_2/0$	$S_1/0$	$S_3/0$	$S_3/0$
S_3	$S_3/0$	$S_0/1$	$S_2/0$	$S_2/0$

6.1.3 已知状态图如图题 6.1.3 所示, 试列出它的转换表。

6.1.4 试画出 101 序列检测器的状态图, 已知此检测器的输入序列、输出序列如下:

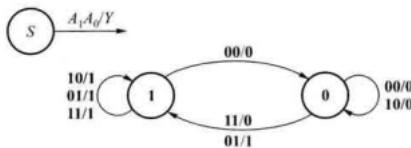
(1) 输入 $A: 0\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1$

输出 $Z: 0\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 1$

(2) 输入 $A: 0\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0$

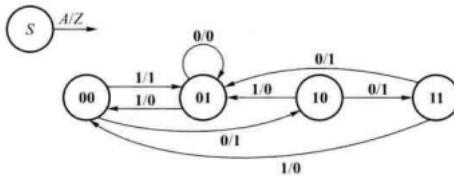
输出 $Z: 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0$

6.1.5 图题 6.1.5 所示是某时序电路的状态图, 设电路的初始状态为 01, 当序列 $A=100110$ 自左至右输入



图题 6.1.3

时,求该电路输出 Z 的序列。



图题 6.1.5

6.1.6 已知某时序电路的状态表如表题 6.1.6 所示,输入为 A ,试画出它的状态图。如果电路的初始状态为 b ,输入信号 A 依次是 **0 1 0 1 1 1 1**,试求其相应的输出。

表题 6.1.6

S^*	S^{*+1}/Z	
	$A=0$	$A=1$
a	$a/0$	$b/0$
b	$a/1$	$d/1$
c	$b/1$	$e/1$
d	$d/0$	$c/0$
e	$b/1$	$a/1$

6.1.7 已知某同步时序电路含有两个上升沿触发的 D 触发器,其激励方程组为

$$D_0 = X_2 X_1 + X_1 Q_0 + X_2 Q_0$$

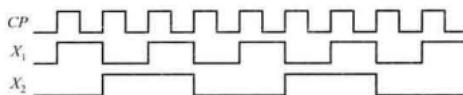
$$D_1 = X_1 \oplus X_2 \oplus Q_0$$

输出方程为

$$Z = Q_1$$

列出转换表,画出状态图,并分析其逻辑功能。若输入信号的波形如图题 6.1.7 所示,且电路的初始状态为 **00**,试画出 Q_1, Q_0 的波形。

6.1.8 已知转换表如表题 6.1.8 所示,若电路的初始状态为 $Q_1 Q_0 = 00$,输入信号 A 的波形如图题 6.1.8 所

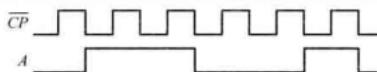


图题 6.1.7

示,输出信号为 Z ,试画出 $Q_1 Q_0$ 的波形(设触发器为下降沿触发)。

表题 6.1.8

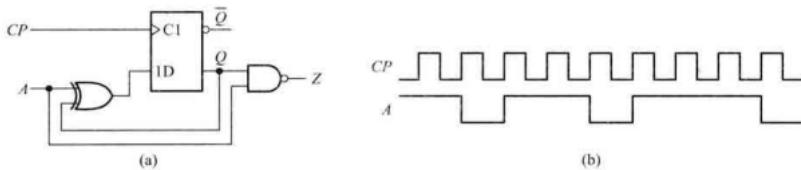
$Q_1^* Q_0^*$	$Q_1^{n+1} Q_0^{n+1}/Z$	
	$A = 0$	$A = 1$
0 0	0 1/1	1 1/1
0 1	1 0/0	1 0/0
1 0	1 0/0	1 1/0
1 1	0 1/1	0 0/1



图题 6.1.8

6.2 同步时序逻辑电路的分析

6.2.1 试分析图题 6.2.1(a)所示时序电路,列出转换表并画出状态图。设电路的初始状态为 0,试画出在图题 6.2.1(b)所示波形作用下, Q 和 Z 的波形图。



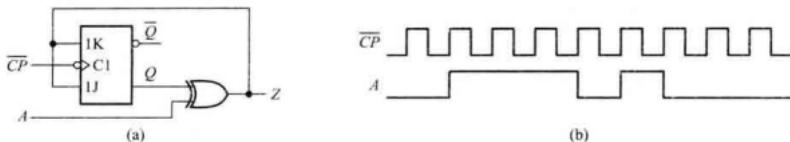
图题 6.2.1

6.2.2 试分析图题 6.2.2(a)所示时序电路,列出转换表并画出状态图。设电路的初始状态为 0,画出在图题 6.2.2(b)所示波形作用下, Q 和 Z 的波形图。

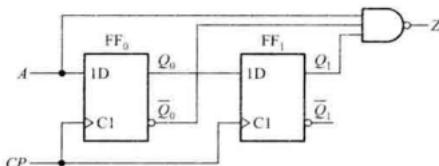
6.2.3 试分析图题 6.2.3 所示同步时序电路,画出状态图。

6.2.4 试分析图题 6.2.4 所示电路,写出它的激励方程组、转换方程组和输出方程,列出转换表并画出状态图。

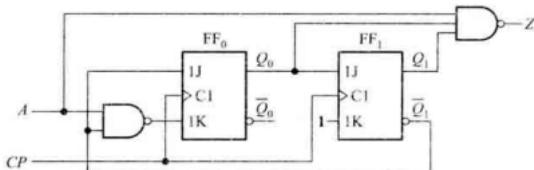
6.2.5 试分析图题 6.2.5 所示同步时序电路,写出各触发器的激励方程、电路的转换方程组和输出方程,列出转换表,画出状态图。



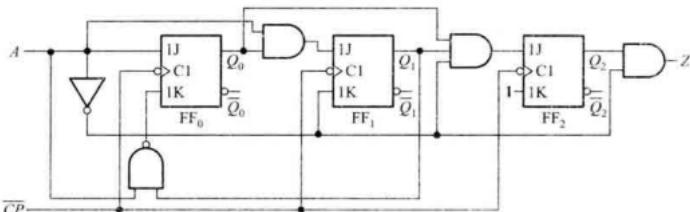
图题 6.2.2



图题 6.2.3

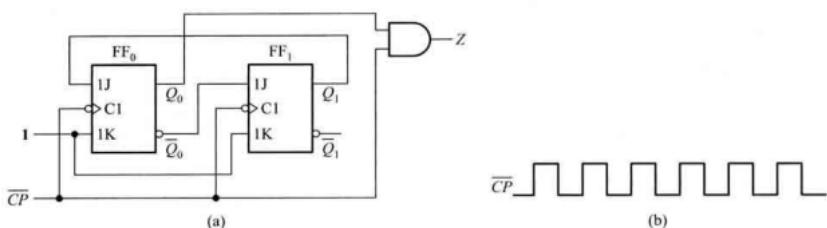


图题 6.2.4



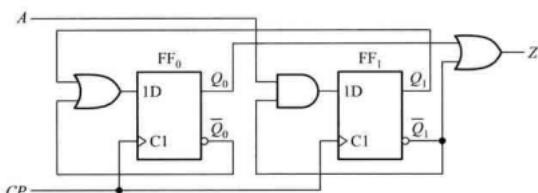
图题 6.2.5

6.2.6 试画出图题 6.2.6(a)所示同步时序电路的状态图，并画出对应于 CP 的 Q_1, Q_0 波形，以及输出 Z 的波形，设电路的初始状态为 00。



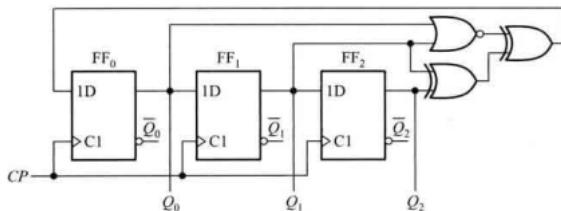
图题 6.2.6

6.2.7 试分析图题 6.2.7 所示同步时序电路,写出激励方程组、转换方程组和输出方程,列出转换表并画出状态图。



图题 6.2.7

6.2.8 试分析图题 6.2.8 所示同步时序电路,写出激励方程组、转换方程组,列出转换表并画出状态图。



图题 6.2.8

6.3 同步时序逻辑电路的设计

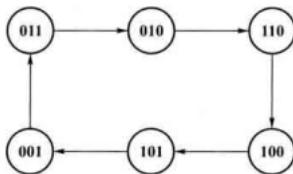
6.3.1 用 JK 触发器设计一同步时序电路,其转换表如表题 6.3.1 所示。

表题 6.3.1

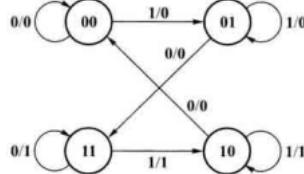
$Q_1^n Q_0^n$	$Q_1^{n+1} Q_0^{n+1}/Y$	
	$A = 0$	$A = 1$
00	01/0	11/0
01	10/0	00/0
10	11/0	01/0
11	00/1	10/1

6.3.2 某同步时序电路的状态图如图题 6.3.2 所示, 试写出用 D 触发器设计时的最简激励方程组。

6.3.3 试用上升沿触发的 JK 触发器设计一同步时序电路, 其状态图如图题 6.3.3 所示, 要求电路使用的门电路最少。

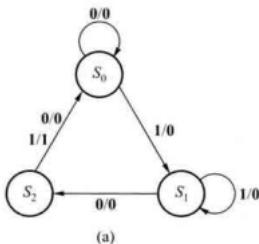


图题 6.3.2

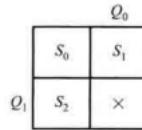


图题 6.3.3

6.3.4 试用下降沿触发的 D 触发器设计一同步时序电路, 其状态图如图题 6.3.4(a) 所示, S_0, S_1, S_2 的编码如图题 6.3.4(b) 的卡诺图所示。



(a)



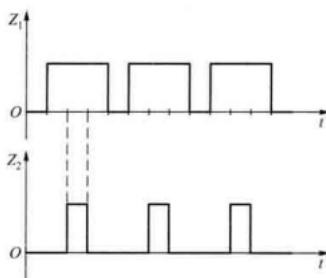
(b)

图题 6.3.4

6.3.5 试用下降沿触发的 JK 触发器和最少的门电路, 实现图题 6.3.5 所示的 Z_1 和 Z_2 输出波形。

6.3.6 试用上升沿触发的 D 触发器设计一个 1101 序列检测器, 输入为串行编码序列, 输出为检出信号。

6.3.7 试用 D 触发器设计一同步时序电路, 其状态表如表题 6.3.7 所示。



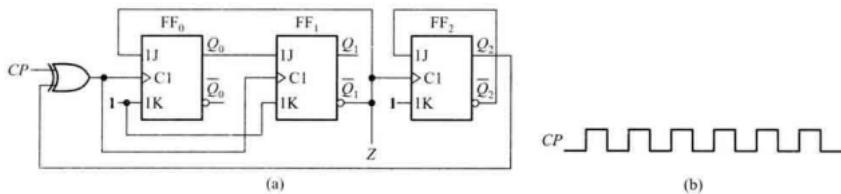
图题 6.3.5

表题 6.3.7

\$S^n\$	\$S^{n+1}\$		Z
	\$A = 0\$	\$A = 1\$	
a	b	d	0
b	c	b	0
c	b	a	1
d	b	c	0

6.4 异步时序逻辑电路的分析

6.4.1 一时序电路如图题 6.4.1(a) 所示, 试画出在 \$CP\$ 作用下, \$Q_0, Q_1, Q_2\$ 和 Z 端的波形, 设各触发器的初始状态均为 0。

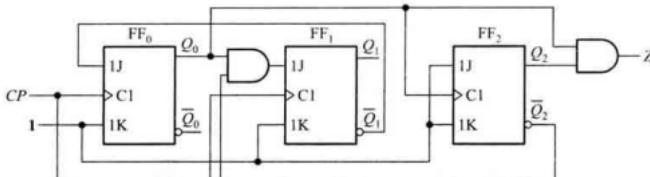


图题 6.4.1

6.4.2 分析图题 6.4.2 所示时序电路 [\$CP\$ 脉冲同图题 6.4.1(b)]。

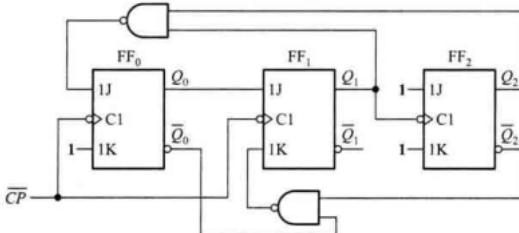
- (1) 写出各触发器的 \$CP\$ 信号方程和激励方程;
- (2) 写出电路的转换方程组和输出方程;
- (3) 列出转换表并画出状态图;

(4) 画出电路的时序图。



图题 6.4.2

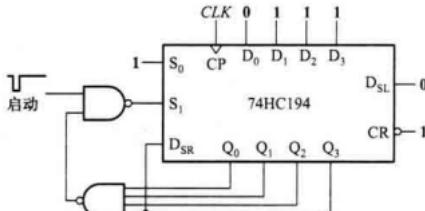
6.4.3 试分析图题 6.4.3 所示时序电路的逻辑功能。



图题 6.4.3

6.5 若干典型的时序逻辑电路

6.5.1 试画出图题 6.5.1 所示电路的输出($Q_3 \sim Q_0$)波形，并分析该电路的逻辑功能。

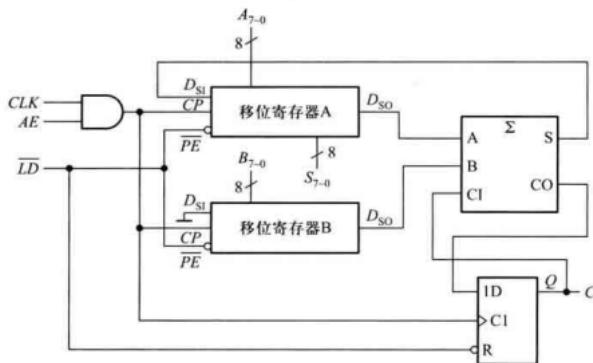


图题 6.5.1

6.5.2 试用两片 74HC194 构成 8 位双向移位寄存器。

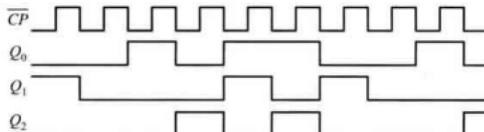
6.5.3 用一个全加器和一个 D 触发器及两个 8 位移位寄存器 A、B 构成的 8 位串行加法电路如图题 6.5.3 所示。图中， CLK 为时钟输入端； \overline{LD} 为置数控制输入端，当 $\overline{LD}=0$ 时，8 位被加数 $A_{7\sim 0}$ 和 8 位加数 $B_{7\sim 0}$ 将分别进入移位寄存器 A 和 B； AE 为加运算控制端，当 $AE=1$ 时，进行串行加法运算，输入 8 个时钟脉冲后恢复为 0； $S_{7\sim 0}$ 为

8位和输出端; C 为进位输出端。移位寄存器A、B的 CP 端为时钟输入端, \overline{PE} 端为并行置数控制端, D_{SI} 和 D_{SO} 端分别为串行数据输入端、输出端。试分析电路的工作原理。



图题 6.5.3

6.5.4 在某计数器的输出端观察到如图题 6.5.4 所示的波形, 试确定该计数器的模。



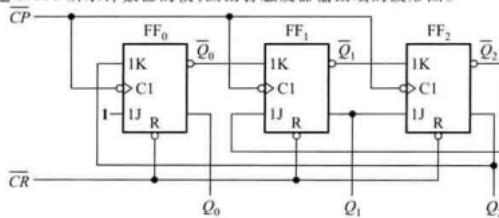
图题 6.5.4

6.5.5 试用下降沿触发的 JK 触发器组成 4 位异步二进制递减计数器, 画出逻辑图。

6.5.6 试用下降沿触发的 D 触发器组成 4 位异步二进制递增计数器, 画出逻辑图。

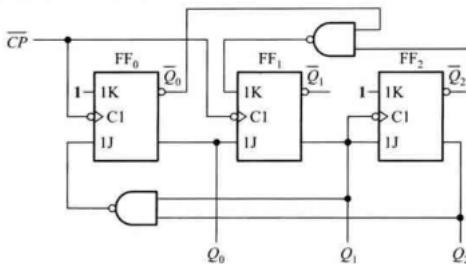
6.5.7 试用上升沿触发的 D 触发器及门电路组成 3 位同步二进制递增计数器, 画出逻辑图。

6.5.8 试分析图题 6.5.8 所示计数器的模, 画出各触发器输出端的波形图。



图题 6.5.8

6.5.9 计数器电路如图题 6.5.9 所示, 试画出它的状态图, 并确定它的模。

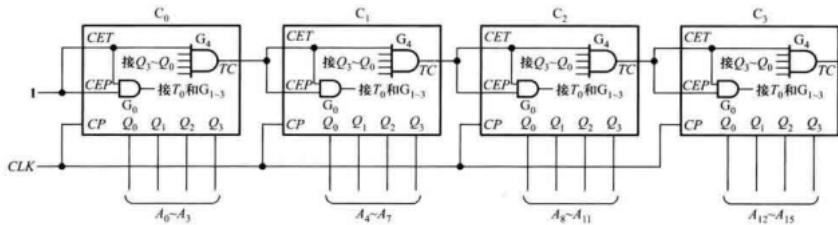


图题 6.5.9

6.5.10 试用上升沿触发的 D 触发器和门电路设计一个同步模 3 递减计数器。

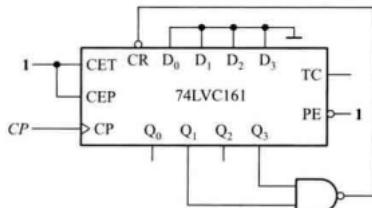
6.5.11 试用 JK 触发器设计一个同步模 6 递增计数器。

6.5.12 图题 6.5.12 所示为一个模 2^{16} 递增计数器的进位原理图, 试与图 6.5.13 所示的计数器进位方式进行比较, 估计它们的激励电路传输延迟时间的最大值 $t_{\text{compd(max)}}$ (以等效逻辑门的平均延迟时间 t_{cpd} 为单位时间)。



图题 6.5.12

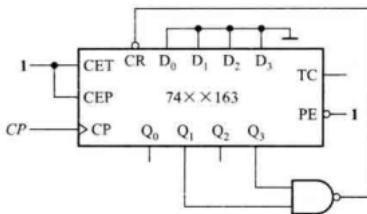
6.5.13 试分析图题 6.5.13 所示计数器, 画出它的状态图, 并确定它的模。



图题 6.5.13

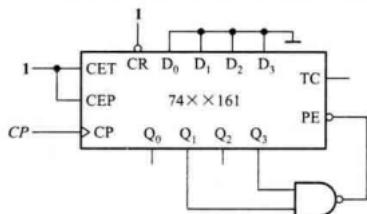
6.5.14 试分析图题 6.5.14 所示计数器, 画出它的状态图, 并确定它的模。(74xx163 是具有同步清零功

能的 4 位同步二进制递增计数器, 其他功能与 74×161 相同)



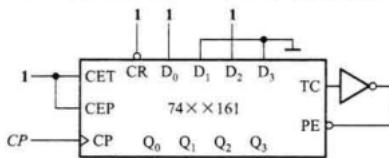
图题 6.5.14

6.5.15 试分析图题 6.5.15 所示计数器, 画出它的状态图, 并确定它的模。



图题 6.5.15

6.5.16 试分析图题 6.5.16 所示计数器, 画出它的状态图, 并确定它的模。



图题 6.5.16

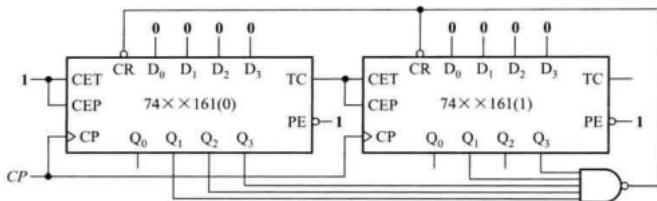
6.5.17 试用 $74HC161$ 设计一个计数器, 其计数序列以自然二进制数 $1001 \sim 1111$ 顺序循环。

6.5.18 试分析图题 6.5.18 所示计数器, 确定它的模。

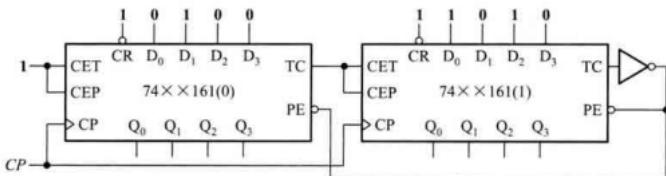
6.5.19 试分析图题 6.5.19 所示计数器, 确定它的模。

6.5.20 试用 74×161 构成同步模 24 计数器, 要求采用两种不同的方法。

6.5.21 试用 4 个具有复位功能的 D 触发器设计一个扭环型计数器, 用复位方式将计数器初始状态置为 $Q_3Q_2Q_1Q_0=0000$, 并用 8 个二输入端与门对它的 8 个计数状态译码, 画出逻辑电路图。



图题 6.5.18



图题 6.5.19

6.6 简单的时序可编程逻辑器件 GAL

6.6.1 对于图 6.6.4 所示的 OLMC, 试画出当 $AC_0 = 1, AC_1(n) = 1, XOR(n) = 1$ 时的等效逻辑电路。

6.7 用 Verilog HDL 描述时序逻辑电路

6.7.1 根据图 6.5.2 所示的逻辑图, 试用 Verilog 语言描述 4 位移位寄存器的功能。然后用 Quartus II 软件进行逻辑功能仿真, 并给出仿真波形。

6.7.2 阅读下列程序, 说明它所完成的功能。

```
module shiftn(Q,PData,SerialData,Load,CP);
    input SerialData;           //Serial Data inputs
    input CP,Load;              //Clock and Load control
    input[n-1:0] PData;         //Parallel Data input
    output reg[n-1:0] Q;        //Register output
    parameter n=8;
    integer k;
    always @ ( posedge CP)
        if( Load) Q<=PData;    //Parallel load input
        else
            begin
                for(k=0;k<n-1;k=k+1)
                    Q[k]<=Q[k+1];
                Q[n-1]<=SerialData;
            end
endmodule
```

```

    end
endmodule

```

6.7.3 试用 Verilog 语言描述一个 4 位二进制可逆计数器的行为。要求如下：

(1) 电路具有 5 种功能, 即异步清零、同步置数、递增计数、递减计数和保持原有状态不变。且要求计数器能输出进位信号和借位信号, 即当计数器递增计数到最大值时, 产生一个高电平有效的进位信号 C_o ; 当计数器递减计数到最小值 0 时, 产生一个高电平有效的借位信号 B_o 。

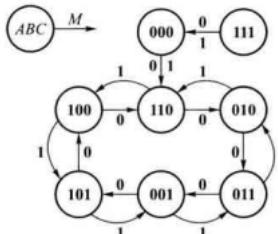
(2) 用 Quartus II 软件进行逻辑功能仿真, 并给出仿真波形。

6.7.4 试用 Verilog 语言描述一个变模计数器, 在 S 和 T 的控制下, 实现同步模 5、模 8、模 10 和模 12 计数, 其模数控制表如表题 6.7.4 所示, 并要求具有异步清零和暂停计数的功能。然后用 Quartus II 软件进行逻辑功能仿真, 并给出仿真波形。

表题 6.7.4 计数器的模数控制表

控制信号 $S \ T$	模 数
0 0	模 5 计数
0 1	模 8 计数
1 0	模 10 计数
1 1	模 12 计数

6.7.5 某电路的状态图如图题 6.7.5 所示, 图中, M 为控制变量, 当 $M=0$ 时, 电路按顺时针方向所指的状态进行转换; 当 $M=1$ 时, 则按逆时针方向进行状态转换。试用 Verilog 描述该电路的功能, 并用 Quartus II 软件进行逻辑功能仿真, 并给出仿真波形。



图题 6.7.5

6.7.6 设计一个序列检测器电路。功能是检测出串行输入数据 Data 中的 4 位二进制序列 0101(自左至右输入), 当检测到该序列时, 输出 $Out=1$; 没有检测到该序列时, 输出 $Out=0$ 。要求:

(1) 给出电路的状态编码, 画出状态图(注意考虑序列重叠的可能性, 如 010101, 相当于出现两个 0101 序列)。

(2) 用 JK 触发器和门电路来设计此电路。

(3) 用 Verilog 的行为描述方式描述该电路的功能。

(4) 然后用 Quartus II 软件进行逻辑功能仿真，并给出仿真波形。

6.7.7 试用 Verilog 的行为描述方式写出数字钟的小时时间计数器程序。要求如下：

(1) 计数器的功能是从 1 开始计数到 12，然后又从 1 开始，周而复始运行。计数器的输出为 8421 BCD 码。

(2) 要求该计数器带有复位端 CR 和计数控制端 EN。当 CR 为低电平时，计数器复位，其输出为 1；当 CR 和 EN 均为高电平时，计数器处于计数状态；当 CR 为高电平但 EN 为低电平时，计数器暂停计数。

(3) 然后用 Quartus II 软件进行逻辑功能仿真，并给出仿真波形。

6.7.8 在例 6.7.9 的基础上，将计时输入脉冲频率改为 1 kHz，要求增加以下功能：

(1) 具有仿电台整点报时功能。即每逢 59 分 51 秒、53 秒、55 秒、57 秒输出 500 Hz 低音频信号，在 59 分 59 秒时输出 1 kHz 高音频信号，输出音频信号的持续时间为 1 s，高音频信号结束时，正好为整点。

(2) 增加闹钟功能，最长闹铃时间为 1 min。闹钟的闹铃时刻可以任意设置（只要求对小时、分钟进行设置）；闹铃信号为 500 Hz 和 1 kHz 的方波信号，两种频率的信号交替输出，且均持续 1 s。设置一个停止闹铃的控制键，可以停止输出闹铃信号。

(3) 采用分层次、分模块的方法，用 Verilog 语言进行设计，并给出各模块的仿真波形图。

半导体存储器

引言

半导体存储器用来存储大量的二值数据,它是计算机等大型数字系统中不可缺少的组成部分,同时也广泛用于小型便携的数码产品中,如数码相机、手机、MP3 播放器等。按照集成度划分,半导体存储器属于大规模集成电路。

本章介绍半导体存储器的基本结构和工作原理,以及各类存储器的特点。用户实际使用时更需要关注存储器的读写控制时序(定时图)和容量扩展等内容,本章对此也作了介绍。

7.1 只读存储器

目前,半导体存储器基本上可以分为两大类,即只读存储器(Read-Only memory, ROM)和随机存取存储器(Random Access Memory, RAM, 又称为读写存储器)。RAM 中的“随机”是指存储器的内容可以以任何顺序存取,而不管前一次存取的是哪一个位置。两者最主要的差别是,正常工作时,即可从 RAM 中读出数据,也可向 RAM 中写入数据,而 ROM 只能读出数据。断电以后, RAM 中所存的数据将全部丢失,即具有易失性;而 ROM 中的数据可以长久保存。

RAM 又可分为静态 RAM(Static Random Access Memory, SRAM)和动态 RAM(Dynamic Random Access Memory, DRAM)。SRAM 中的存储单元相当于一个锁存器,有 0、1 两个稳态;DRAM 则是利用电容器存储电荷来保存 0 或 1 的,因此需要定时对其进行刷新,否则随着时间推移,电容器中存储的电荷将逐渐消散。

根据是否允许用户对 ROM 写入数据,又可将 ROM 分为固定 ROM(或掩模 ROM)和可编程 ROM(Programmable Read-Only Memory, PROM)。PROM 又可分为一次可编程 ROM(PROM),光可擦除可编程 ROM(Erasable Programmable Read-Only Memory, EPROM),电可擦除可编程 ROM(Electrical Erasable Programmable Read-Only Memory, E²PROM)和快闪存储器(Flash Memory)。

RAM 一般用在需要频繁读写数据的场合,如计算机系统中的数据缓存。ROM 常用于存放

系统程序、数据表、字符代码等不易变化的数据。 E^2PROM 和 Flash Memory 则广泛用于各种存储卡中,例如公交车的 IC 卡,数码相机中的存储卡,移动存储卡(USB 存储卡,也称 U 盘)等。

7.1.1 ROM 的基本结构

ROM 是一种永久性数据存储器,其中的数据一般由专用的装置写入,数据一旦写入,不能随意改写,在切断电源之后,数据也不会消失。

存储器由存储阵列、地址译码器和输出控制电路三部分组成,其结构如图 7.1.1 所示。

存储阵列由许多存储单元组成,每个存储单元存放 1 位二值数据。通常存储单元排列成矩阵形式,且按一定位数进行编组,每次读出一组数据,这里的组称为字。一个字中所含的位数称为字长。为了区别各个不同的字,给每个字赋予一个编号,称为地址。构成字的存储单元也称为地址单元。地址译码器将输入的地址代码译成相应的地址信号,从存储矩阵中选出相应的存储单元,并将其中的数据送到输出控制电路。地址单元的个数 N 与二进制地址码的位数 n 满足关系式 $N=2^n$ 。

输出控制电路一般都包含三态缓冲器,以便与系统的数据总线连接。当有数据读出时,可以有足够的能力驱动数据总线;没有数据输出时,输出高阻态不会对数据总线产生影响。

图 7.1.2 是一个 ROM 的结构示意图,其中存储阵列由字线和位线交叉处的二极管构成。该存储器有 4 个地址单元(4 个字),字长为 4 位。

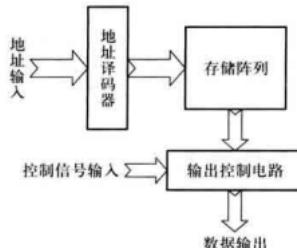


图 7.1.1 ROM 的基本结构框图

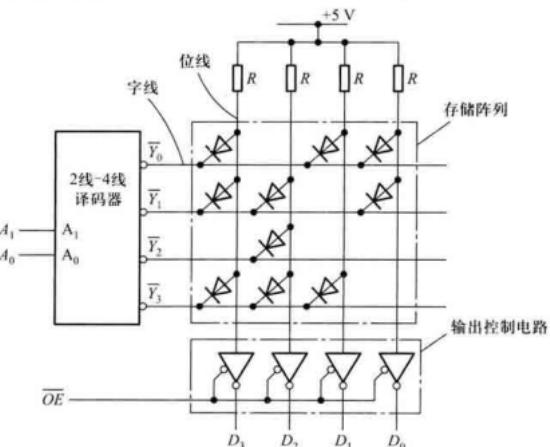


图 7.1.2 ROM 结构示意图

读操作时,如果给定的地址码为 $A_1A_0=01$,译码器的 $\bar{Y}_0 \sim \bar{Y}_3$ 中只有 \bar{Y}_1 为低电平,则 \bar{Y}_1 字线

与所有位线交叉处的二极管导通,使相应的位线变为低电平,而交叉处没有二极管的位线仍保持高电平。此时,若输出使能控制信号 $\overline{OE} = 0$,则位线电平经缓冲器反相输出,使 $D_3 D_2 D_1 D_0 = 1101$ 。因此,ROM 属于组合电路,给定一组输入(地址),便可得到一组输出(内容)。该 ROM 的 4 个地址内所存储的数据如表 7.1.1 所示。

表 7.1.1 图 7.1.2 ROM 存储的内容

输出使能控制 \overline{OE}	地址 $A_1 A_0$	内 容 $D_3 D_2 D_1 D_0$
0	0 0	1 0 1 1
0	0 1	1 1 0 1
0	1 0	0 1 0 0
0	1 1	1 1 1 0
1	xx	高 阻

以上分析可知,字线与位线交叉处相当于一个存储单元,此处若有二极管存在,则存储单元存有 1 值,否则为 0 值。

在实际应用中,常以字数和字长的乘积表示存储器的容量(也称为密度),存储器的容量越大,意味着能存储的数据越多。例如,一个容量为 256×4 位的存储器,有 256 个字,字长为 4 位,总共有 1024 个存储单元。存储容量较大时,字数通常采用 K、M 或 G 为单位。其中 $1K = 2^{10} = 1024$, $1M = 2^{20} = 1024K$, $1G = 2^{30} = 1024M$ 。图 7.1.2 所示 ROM 的容量为 $2^2 \times 4$ 位。

7.1.2 二维译码与存储阵列

如果采用图 7.1.2 中的译码方式构造一个 $2^8 \times 1$ 位的 ROM,则需要一个 8 线-256 线译码电路。若采用一级门电路译码,则该译码电路至少需要 256 个 8 输入与非门和 16 个反相缓冲器,电路非常庞大。在实际的 ROM 中,常采用行译码和列译码的二维译码结构来减小译码电路的规模,如图 7.1.3 所示。

4 线-16 线译码器为行译码器,它需要 16 个 4 输入与门和 8 个反相缓冲器。该译码电路输出高电平有效。16 线-1 线数据选择器构成列译码器,需要 16 个 5 输入与门、一个 16 输入或门和 8 个反相缓冲器^①。可见采用二维译码结构,译码电路的规模大大减小了。

图 7.1.3 中的存储阵列由行线和列线交叉处的 MOS 管^②构成。当给定的地地址码为 $A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0 = 00010001$ 时, $A_7 A_6 A_5 A_4$ 经译码器译码, 输出 Y_1 行线为高电平, 则栅极与 Y_1 相连的 MOS 管导

① 译码器和数据选择器的电路结构可参阅 4.4.2~4.4.3 节。

② 为简明起见,本章 MOS 管采用简化符号。

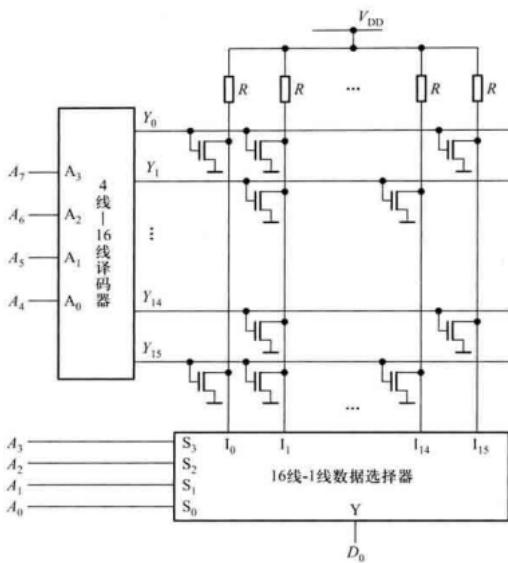


图 7.1.3 用 MOS 管构成存储单元的 ROM 结构示意图

通,使 I_1, I_{14} 的位线变为低电平;而交叉处未连接 MOS 管的位线仍保持高电平,如 I_0, I_{15} 的位线仍为高电平。而此时地址码的低 4 位 $A_3, A_2, A_1, A_0 = 0001$,数据选择器选择 I_1 位线输出,即 $D_0 = I_1 = 0$ 。如果 Y_i 行线和 I_i 位线交叉处没有 MOS 管, $D_0 = I_i = 1$ 。一般数据选择器的输出 D_0 还需经反相输出缓冲器再输出。由此看出,4 线-16 线译码器实现行的选择,16 线-1 线数据选择器实现列的选择,从而完成行和列的译码。ROM 通过行和列交叉点上是否连有 MOS 管来存储 0 和 1。

7.1.3 可编程 ROM

由图 7.1.2 和图 7.1.3 看出,存储阵列可用二极管构成,也可用 MOS 管或 BJT 管构成。这类 ROM 也称为固定 ROM 或掩模 ROM,器件制造商根据用户提供的 ROM 存储内容,在制造时,利用掩模技术把数据写入存储器中(构建存储阵列中字线与位线交叉处二极管的有、无),一旦 ROM 制成,其存储的数据也就固定不变了。

另外,存储阵列也可采用 4.5.1 节介绍的带金属熔丝的二极管、SIMOS 管、Flotox MOS 管和快闪叠栅 MOS 管等,制成各种可编程 ROM。一次可编程存储器 PROM 的存储阵列由带金属熔丝的二极管构成。出厂时,PROM 存储内容全为 1(或者全为 0),用户可以根据要写入的数据,利用编程软件生成编程数据(也称为熔丝图),再通过通用或专用的编程器,根据熔丝图将某些单

元的熔丝烧断,来改写存储的内容。由于熔丝烧断后不能恢复,因此 PROM 只能改写一次。

光可擦除可编程存储器 EPROM 的存储阵列由 SIMOS 管构成,其数据写入需要通用或专用的编程器。EPROM 芯片的封装外壳装有透明的石英盖板,用紫外线或 X 射线照射 15~20 min,便可擦除其全部内容。擦除后可重新写入数据。如今大多数 PROM 实际上是不装透明石英盖板的 EPROM,因而无法擦除,只能写入一次,也称 OTP(One Time Programmable)EPROM。

电可擦除可编程存储器 E²PROM 和快闪存储器阵列分别由 Flotox MOS 管和快闪叠栅 MOS 管构成。E²PROM 既具有 ROM 的非易失性,又具有写入功能。改写过程就是电擦除过程(在线擦除,即不需要将芯片从电路系统中取出。可重复擦写 1 万次以上),改写以字为单位进行。目前,大多数 E²PROM 芯片内部都备有升压电路。因此,只需提供单电源供电,便可进行读、擦除/写操作,这为数字系统的设计和在线调试提供了极大方便。与 EPROM 相比,E²PROM 的存储单元电路复杂,所以集成度低。

快闪存储器的擦除和写入是分开进行的,通过在快闪叠栅 MOS 管的源极加正电压完成擦除操作,而在 MOS 管的栅极加高的正电压完成写入操作。因此写入前,首先要进行擦除。由于快闪存储器的存储单元结构简单(只需要一个快闪叠栅 MOS 管),所以集成度较 E²PROM 高。

上述可编程 ROM 的发展,实际上已经改变了 ROM 最初只读存储器的含义,而既有读功能,又有写功能。特别是快闪存储器具有的大容量、可读写、非易失性特点,使之广泛应用于各种数码产品中。为便于比较,将几种 ROM 性能列于表 7.1.2 中。

表 7.1.2 几种 ROM 性能比较

	快闪存储器	ROM	EPROM	E ² PROM
非易失性	是	是	是	是
高密度	是	是	是	否
单管存储单元	是	是	是	否
在系统可写	是	否	否	是

由前面所述 ROM 的结构看出,地址码的输入和数据的输出都是并行的。实际上,还有串行输入输出的 ROM,其地址为串行输入,数据也是串行输出的(PROM 也有数据输入)。而且经常是地址和数据分时共用同一个引脚。由于串行结构大大减少了芯片的引脚数,所以其体积可以做得很小。目前,该结构的存储器已在便携设备中得到广泛应用,如日常生活中的各种 IC 卡就属于这种结构。

7.1.4 ROM 读操作实例

下面通过一个 PROM 芯片例子 AT27C010,了解 ROM 的具体使用。

1. PROM 芯片 AT27C010 简介

该芯片是美国 Atmel 公司^①生产的 128K×8 位的 OTP EPROM。在读工作方式下,采用 5V 电源,读出时间最短为 45 ns,静态时工作电流小于 10 μA。图 7.1.4 是它的框图和引脚图。图中 V_{CC} 是读操作时的工作电压, V_{PP} 是数据写入时的编程电压(编程写入时, $V_{PP} = 13 V$)。 \overline{OE} 是输出使能信号, \overline{PGM} 是编程选通信号。为便于控制还加有片选信号 \overline{CE} 。NC 为空脚。AT27C010 的工作模式如表 7.1.3 所示。

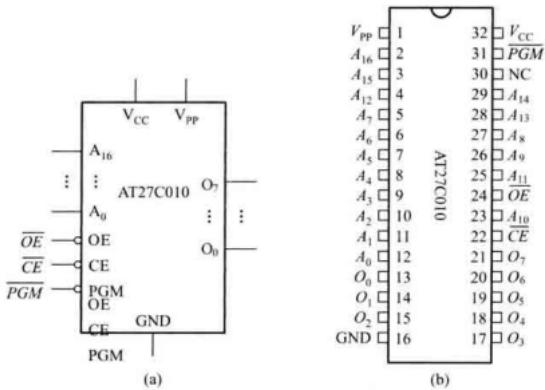


图 7.1.4 128K×8 位的 OTP EPROM AT27C010

(a) AT27C010 框图 (b) AT27C010 双列直插式封装的引脚图

表 7.1.3 工作模式

工作模式	\overline{CE}	\overline{OE}	\overline{PGM}	$A_{16} \sim A_0$	V_{PP}	$O_7 \sim O_0$
读	0	0	X	A_i	X	数据输出
输出无效	X	1	X	X	X	高阻
等待	1	X	X	A_i	X	高阻
快速编程	0	1	0	A_i	V_{PP}	数据输入
编程校验	0	0	1	A_i	V_{PP}	数据输出

^① Atmel Corporation 的主要产品种类有存储器、智能卡 IC 等。AT27C010 资料选自该公司数据文档 0321K - EPROM - 4/2004。

2. 读操作及定时图

为了保证存储器准确无误地工作, 加到存储器的地址和控制信号必须遵守规定的时序要求, AT27C010 的读时序如图 7.1.5 所示, 阴影部分表示不确定状态。

读出过程操作如下:

- (1) 欲读取单元的地址加到存储器的地址输入端;
- (2) 加入有效的片选信号 \overline{CE}
- (3) 使输出使能信号 \overline{OE} 有效, 经过一定延时后, 有效数据出现在数据线上;
- (4) 让片选信号 \overline{CE} 或输出使能信号 \overline{OE} 无效, 经过一定延时后, 数据线呈高阻态, 本次读出结束。

图 7.1.5 中 t_{AA} 为地址存取时间, 表示从地址信号加到存储器上到数据输出端有稳定的数据输出所需要的时间。此前必须保证 \overline{CE} 和 \overline{OE} 已经为 0。 t_{CE} 为片选取时间, 表示在地址输入端已经有稳定地址和输出使能信号 \overline{OE} 有效的条件下, 从片选信号 \overline{CE} 有效到数据稳定输出所需时间。 t_{OE} 为输出使能时间, 表示在地址输入端已经有稳定地址和片选信号 \overline{CE} 有效的条件下, 从输出使能信号 \overline{OE} 有效到数据稳定输出所需时间。显然在进行读操作时, 只有在地址、 \overline{CE} 和 \overline{OE} 均有效, 且延迟时间均满足 t_{AA} 、 t_{CE} 和 t_{OE} 以后, 被读单元的内容才能稳定地出现在数据输出端。而当地址失效后, 数据输出端上的数据保持 t_{OH} 后才失效。 \overline{CE} 或 \overline{OE} 失效后, 经延迟 t_{OZ} (输出失效时间) 后数据端呈高阻态。

AT27C010 延迟时间的一组典型值为: $t_{AAmax} = 45 \text{ ns}$, $t_{CEmax} = 45 \text{ ns}$, $t_{OEmax} = 20 \text{ ns}$, $t_{OZmax} = 20 \text{ ns}$, $t_{OHmin} = 7 \text{ ns}$ 。

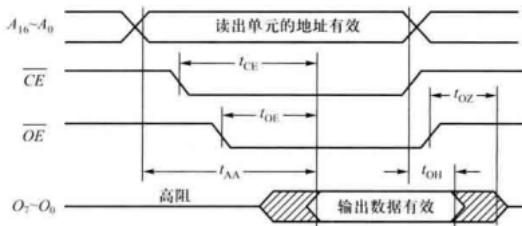


图 7.1.5 AT27C010 的读操作定时图

由于 PROM 的数据写入均由专用或通用编程器完成, 一般无需用户直接对其引脚进行操作, 所以此处不再介绍数据写入时序, 若有需要, 可参阅制造商的产品数据手册。

7.1.5 ROM 应用举例

ROM 常用于存放系统的运行程序或固定不变的数据。此外,由于 ROM 是一种组合逻辑电路,因此可以用它来实现各种组合逻辑函数,特别是多输入、多输出的逻辑函数。设计实现时,只需列出真值表,逻辑函数的输入作为地址,输出作为存储内容,将内容按地址写入 ROM 即可。下面举例说明 ROM 的简单应用。

用 ROM 实现二进制码与格雷码相互转换的电路如图 7.1.6 所示。该电路需要用 ROM 的 5 根地址线和 4 根数据线。连接地址线最高位的 C 为转换方向控制位,待转换的代码由 I_3, I_2, I_1, I_0 输入,转换后的代码由 O_3, O_2, O_1, O_0 输出。当 $C=0$ 时,实现二进制码到格雷码的转换;而当 $C=1$ 时,实现格雷码到二进制码的转换。ROM 中的内容如表 7.1.4 所示(两种码的关系参见 1.4.2 节)。该 ROM 的容量至少为 $2^5 \times 4$ 位。

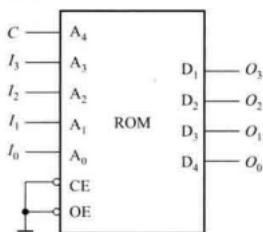


图 7.1.6 用 ROM 实现二进制码与格雷码相互转换

表 7.1.4 ROM 中的内容

C (A_4)	$I_3\ I_2\ I_1\ I_0$ ($A_3A_2A_1A_0$) 二进制码	$O_3O_2O_1O_0$ ($D_3D_2D_1D_0$) 格雷码	C (A_4)	$I_3\ I_2\ I_1\ I_0$ ($A_3A_2A_1A_0$) 格雷码	$O_3O_2O_1O_0$ ($D_3D_2D_1D_0$) 二进制码
0	0 0 0 0	0 0 0 0	1	0 0 0 0	0 0 0 0
0	0 0 0 1	0 0 0 1	1	0 0 0 1	0 0 0 1
0	0 0 1 0	0 0 1 1	1	0 0 1 0	0 0 1 1
0	0 0 1 1	0 0 1 0	1	0 0 1 1	0 0 1 0
0	0 1 0 0	0 1 1 0	1	0 1 0 0	0 1 1 1
0	0 1 0 1	0 1 1 1	1	0 1 0 1	0 1 1 0
0	0 1 1 0	0 1 0 1	1	0 1 1 0	0 1 0 0

续表

C (A_4)	$I_3\ I_2\ I_1\ I_0$ ($A_3A_2A_1A_0$) 二进制码	$O_3O_2O_1O_0$ ($D_3D_2D_1D_0$) 格雷码	C (A_4)	$I_3\ I_2\ I_1\ I_0$ ($A_3A_2A_1A_0$) 格雷码	$O_3O_2O_1O_0$ ($D_3D_2D_1D_0$) 二进制码
0	0 1 1 1	0 1 0 0	1	0 1 1 1	0 1 0 1
0	1 0 0 0	1 1 0 0	1	1 0 0 0	1 1 1 1
0	1 0 0 1	1 1 0 1	1	1 0 0 1	1 1 1 0
0	1 0 1 0	1 1 1 1	1	1 0 1 0	1 1 0 0
0	1 0 1 1	1 1 1 0	1	1 0 1 1	1 1 0 1
0	1 1 0 0	1 0 1 0	1	1 1 0 0	1 0 0 0
0	1 1 0 1	1 0 1 1	1	1 1 0 1	1 0 0 1
0	1 1 1 0	1 0 0 1	1	1 1 1 0	1 0 1 1
0	1 1 1 1	1 0 0 0	1	1 1 1 1	1 0 1 0

另外,还可以用 ROM 实现两个 4 位二进制数的乘法运算。两个 4 位二进制数分别作为地址的高 4 位和低 4 位,他们的积需要 8 位。这样选用容量为 $2^8 \times 8$ 位的 ROM 便可实现该乘法运算,读者可以试着自行确定 ROM 中的内容。

随着集成电路技术的发展,ROM 的价格已变得相对低廉,加之用它实现逻辑函数非常简单易行,所以在某些情况下,用该法实现逻辑函数不失为一种有效的方法。实际上,8.2 节的现场可编程门阵列就是借鉴这种方法实现逻辑函数的。

复习思考题

- 7.1.1 什么是存储器的数据非易失性?
- 7.1.2 在存储器的结构中,什么叫“字”?什么叫“字长”?如何标注存储器的容量?
- 7.1.3 一个存储容量为 256×8 位的 ROM,其地址码应为多少位?
- 7.1.4 哪几种 ROM 具有多次擦除重写功能?哪种 ROM 的擦除过程就是数据写入过程?
- 7.1.5 用 ROM 实现无符号 16 位二进制数的加/减运算,要求有加/减模式控制、低位的进位输入以及进位输出。问:该 ROM 需要有多少根地址线?多少根数据线?其存储容量为多少?

7.2 随机存取存储器

RAM 是另一大类存储器, 它与 ROM 的最大区别就是数据易失性, 一旦失去电源供电, 所存储的数据立即丢失。最大优点是可以随时快速地从其中任一指定地址读出(取出)或写入(存入)数据。RAM 又分为静态 RAM(SRAM)和动态 RAM(DRAM)。下面首先介绍 SRAM。

7.2.1 SRAM

1. SRAM 的基本结构及输入输出

静态 RAM 常称为 SRAM, 它的基本结构与 ROM 类似, 由存储阵列、地址译码器和输入/输出控制电路三部分组成。SRAM 的一般结构框图如图 7.2.1 所示。

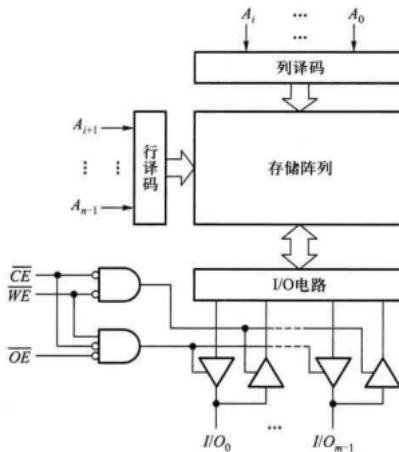


图 7.2.1 SRAM 的结构框图

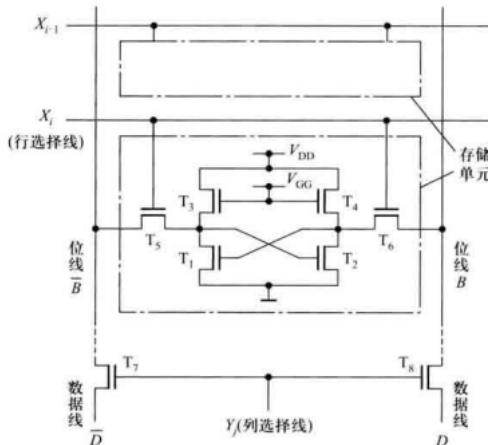
其中 $A_0 \sim A_{n-1}$ 是 n 根地址线, $I/O_0 \sim I/O_{m-1}$ 是 m 根双向数据线, 其容量为 $2^n \times m$ 位。 \overline{OE} 为输出使能信号, \overline{WE} 是写使能信号, \overline{CE} 为片选信号。只有在 $\overline{CE} = 0$ 时, RAM 才能进行正常读写操作, 否则, 三态缓冲器均为高阻, SRAM 不工作。为降低功耗, 一般 SRAM 中都设计有电源控制电路(图中未画), 当片选信号 \overline{CE} 无效时, 将降低 SRAM 内部的工作电压, 使其处于微功耗状态。I/O 电路主要包含数据输入驱动电路和读出放大器, 以使 SRAM 的内外电平能更好地匹配。SRAM 的工作模式如表 7.2.1 所示。

表 7.2.1 SRAM 的工作模式

工作模式	\overline{CE}	\overline{WE}	\overline{OE}	$I/O_0 \sim I/O_{n-1}$
保持(微功耗)	1	x	x	高阻
读	0	1	0	数据输出
写	0	0	x	数据输入
输出无效	0	1	1	高阻

2. SRAM 存储单元

SRAM 与 ROM 最主要的差别是存储单元。SRAM 的存储单元是由锁存器构成的，因此它属于时序逻辑电路。图 7.2.2 画出了存储阵列中第 j 列、第 i 行的存储单元结构示意图。虚线框中为六管 SRAM 存储单元，其中 NMOS 管 $T_1 \sim T_4$ 构成一个 RS 锁存器用来存储 1 位二值数据。 X_i 为行译码器的输出， Y_j 为列译码器的输出。 T_5, T_6 为本单元控制门，由行选择线 X_i 控制。 $X_i = 1$ ， T_5, T_6 导通，锁存器与位线接通； $X_i = 0$ ， T_5, T_6 截止，锁存器与位线隔离。 T_7, T_8 为一列存储单元公用的控制门，用于控制位线与数据线的连接状态，由列选择线 Y_j 控制。显然，当行选择线和列选择线均为高电平时， $T_5 \sim T_8$ 都导通，锁存器的输出才与数据线接通，该单元才能通过数据线传送数据。因此，存储单元能够进行读/写操作的条件是，与它相连的行、列选择线均须呈高电平。

图 7.2.2 第 j 列、第 i 行的存储单元结构示意图

由此可见，SRAM 中数据由锁存器记忆，只要不断电，数据就能永久保存。

3. 读写操作及定时图

SRAM 工作时有读和写两种操作,他们是分时进行的,下面根据表 7.2.1 的工作模式分别加以介绍。

读操作时,有效地址要加到地址输入端,片选信号 \overline{CE} 为低电平,写使能信号 \overline{WE} 为高电平,输出使能信号 \overline{OE} 为低电平,SRAM 中的数据被送到数据线上。由于实际的逻辑电路都存在延时,所以从给定输入信号到数据输出需要一定的时间。SRAM 典型的读操作定时图如图 7.2.3 所示。图中左侧是由地址控制的读操作, \overline{CE} 和 \overline{OE} 始终保持低电平,此时数据的输出只与地址有关。右侧是 \overline{CE} 控制的读操作,此前地址已经有效或与 \overline{CE} 同时有效。图中定时参数的含义见表 7.2.2^①,并给出了 Cypress 公司生产的 SRAM 产品 CY7C1062DV33 的参数值。只有在满足相关参数最低要求时,才能顺利完成读操作。器件制造商在数据手册中会提供这些参数的具体数值。

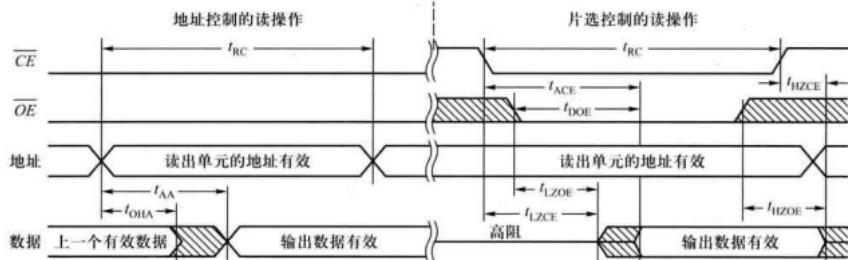


图 7.2.3 SRAM 典型的读操作定时图 ($\overline{WE}=1$)

表 7.2.2 读操作定时参数含义

参数	说明	CY7C1062DV33 参数值样例 (单位:ns)	
		最小值	最大值
t_{RC}	读周期。连续进行两次读操作所需要的最长时间间隔	10	
t_{AA}	地址存取时间。从地址有效到数据稳定输出的延迟时间		10
t_{OHA}	输出保持时间。地址变化后,输出数据仍然保持有效的时间	3	
t_{ACE}	片选存取时间。片选信号有效到数据稳定输出的延迟时间		10

① 不同制造商定义的参数有所不同,但基本类似。此处参照了 Cypress 公司的参数定义。

续表

参数	说明	CY7C1062DV33 参数值样例 (单位:ns)	
		最小值	最大值
t_{DOE}	输出使能时间。输出使能信号有效到数据稳定输出的延迟时间,通常 $t_{DOE} < t_{ACK}$		5
t_{LZOE}	输出使能高阻维持时间。输出使能有效后,输出仍维持高阻的时间	1	
t_{LZCE}	片选高阻维持时间。片选有效后,输出仍维持高阻的时间	3	
t_{HZCE}	片选保持时间。片选无效后,输出数据仍保持有效的时间,通常 $t_{HZCE} < t_{LZCE}$		5
t_{HZOE}	输出使能保持时间。输出使能无效后,输出数据仍保持有效的时间,通常 $t_{HZOE} < t_{LZOE}$		5

写操作时,有效地址要加到地址输入端,片选信号 \overline{CE} 有效,写使能信号 \overline{WE} 有效时,数据线上的数据被写入 SRAM 中。SRAM 典型的写操作定时图如图 7.2.4 所示。图中左侧为 \overline{CE} 控制写入时的情况;右侧为 \overline{WE} 控制写入时的情况。定时参数的含义如表 7.2.3 所示。同样,在满足参数最低要求时,才能正确完成写操作。

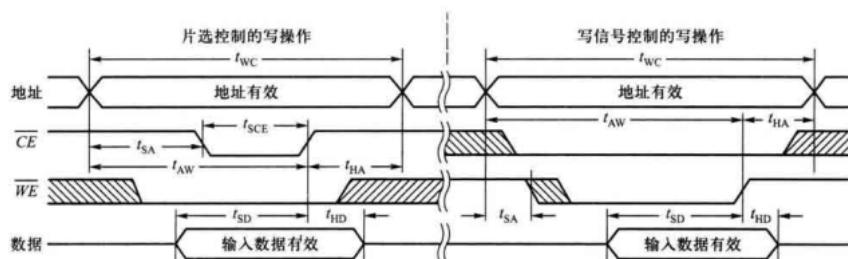
图 7.2.4 SRAM 典型的写操作定时图 ($OE=1$)

表 7.2.3 写操作定时参数含义

参数	说明	CY7C1062DV33 参数值样例 (单位:ns)	
		最小值	最大值
t_{WC}	写周期	10	
t_{SA}	地址建立时间。反映在写控制信号有效前,地址必须稳定一段时间	0	
t_{AW}	写结束前地址保持时间	7	
t_{SCE}	片选持续时间	7	
t_{SD}	写结束前数据建立时间。反映在写信号失效前,数据线上的数据应保持稳定的时间	5.5	
t_{HA}	写结束后地址维持时间	0	
t_{HD}	写结束后数据维持时间	0	

大多数 SRAM 的读周期和写周期是相等的,根据器件工作速度不同其时长也不同,一般为几纳秒到几十纳秒。

截止到 2010 年 10 月,Cypress 公司已有容量达 64Mbit 的产品提供,存取时间最短 10 ns。

另外,一种称作先进先出(First-in First-out, FIFO)的 SRAM 也得到广泛应用。所谓先进先出是指先存入的数据也先被读出,不能随机存取。意味着 FIFO 的读写地址是按一定顺序变化的。为了实现先进先出的读写功能,FIFO 在结构设计上与普通 SRAM 有较大差别,它的数据写入端口和数据读出端口是分开的,并且没有地址输入端口。读写数据的地址由 FIFO 内部的读地址指针计数器和写地址指针计数器来确定。写操作时,写控制信号有效,输入端口的数据写入到由写指针指向的地址单元中,然后写指针计数器加 1。读操作与此类似。该存储器必须先写入数据后,才能读出数据。当 FIFO 中的数据被读完后,表示“空”状态的输出信号有效。若写入 FIFO 中的数据没有被及时读出,可能会出现被写“满”的情况,此时表示“满”状态的输出信号有效。“空”和“满”状态信号为用户控制 FIFO 的读写提供了方便。

FIFO SRAM 多用来做数据缓冲器,特别适合用于需要长时间、不间断、高速数据采集的缓冲器。

还有一种不同于 FIFO 的双口(Dual-Port)SRAM。这种存储器有两套完全独立完整的地址、数据和控制端口,共用同一个存储阵列。两个端口都能进行读写操作,不像 FIFO 那样数据只能一个口进一个口出,而且能根据各自端口的地址随机存取。当出现两个端口同时访问相同的地址单元时,SRAM 内部仲裁电路将根据两个端口先后访问的微小时差,决定先完成哪个端口的访问,并输出状态信号告知另一端口将延迟其访问。双口 SRAM 为数据缓存提供了更便利的条件。

与 ROM 类似,SRAM 也有串行结构。由于串行 SRAM 芯片减少了引脚数,所以可以做得非常小,通常用在速度要求不高,而体积要求非常小的系统中。

7.2.2 同步 SRAM

1. 同步 SRAM 的基本结构及工作原理简介

随着各种数据密集型应用(如互联网中的交换机、路由器、服务器,以及通信领域的无线基站和测试设备等)对速度要求的不断提高, RAM 和其他大规模及超大规模集成电路(如 Intel 的奔腾处理器)一样,近些年也得到快速发展,同步 SRAM(Synchronous SRAM,SSRAM)是在 SRAM 基础上发展起来的一种更适合高速存取的 RAM。同步 SRAM 与 SRAM 最主要的差别是,前者的读写操作是在时钟脉冲节拍控制下完成的。因此,同步 SRAM 最明显的标志是有时钟脉冲输入端。为便于区别,上节介绍的 SRAM 也称为异步 SRAM(Asynchronous SRAM,ASRAM)。

同步 SRAM 的基本结构如图 7.2.5 所示。为了能更好地实现同步操作,与 SRAM 相比,同步 SRAM 在结构上最大的不同是在电路内部增加了包括地址、数据、读写控制等各种输入信号的寄存器。除输出使能控制信号 \overline{OE} 外,所有输入均在时钟脉冲 CP 的上升沿被取样,存入寄存器。

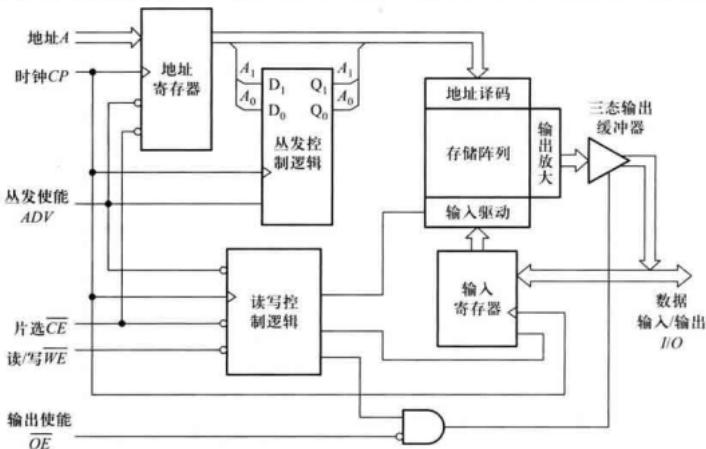


图 7.2.5 同步 SRAM 的基本结构

由图 7.2.3 和图 7.2.4 异步 SRAM 的读写定时图看到,相关信号的定时参数必须满足一定要求才能正确实现读写操作。特别是在高速存取的应用场合,必须精心设计异步 SRAM 的地址、数据和读写控制等信号的时序。而同步 SRAM 是在时钟脉冲控制下完成读写操作的。在时钟有效沿到来时,地址、数据、控制等信号被锁存到同步 SRAM 内部的寄存器中,读写过程的延

时等待均被限制在时钟作用间隔内。这种工作方式使其应用变得非常简便,用户只要围绕时钟的有效沿设计所有的输入信号即可。另外,可以对芯片内部的各种时延进行优化设计,使得同步 SRAM 的读写速度高于异步 SRAM。

同步 SRAM 具有的另一个特点是从发(Burst,也译为突发)读写操作模式^①。在该工作模式下,只要外部给定读写存储单元的首地址,在时钟脉冲作用下,由内部地址计数器提供首地址后的一组连续地址,就可以连续读写接下来的若干个地址单元,而不再需要外部输入地址。同步 SRAM 的这种从发模式,在连续读/写多个字时,可以减少外部地址总线的占用时间,提高读写效率。根据内部地址计数器的位数不同,从发读写的字数可以是 2、4、8 个字不等。

图 7.2.5 中的从发控制逻辑电路中包含一个 2 位的二进制地址计数器(也称为从发计数器),从发模式下可以连续读写 4 个字。外部地址码的最低 2 位 A_1A_0 经过该电路后再送至地址译码电路。 ADV 是从发使能控制信号。同步 SRAM 的读写工作模式如表 7.2.4 所示。

表 7.2.4 同步 SRAM 的读写工作模式

工作模式	CP	ADV	\overline{CE}	\overline{WE}	\overline{OE}	I/O
一般读	↑	0	0	1	0	数据输出
一般写	↑	0	0	0	×	数据输入
从发读	↑	1	0	1	0	数据输出
从发写	↑	1	0	0	×	数据输入

当 $ADV=1$ 时,地址寄存器不接收外部新地址而保持上一次存入的地址,在 CP 下一个上升沿到来时,从发计数器在上一个 A_1A_0 基础上加 1 产生下一个地址,并在其他控制信号作用下进行读/写。如果保持 $ADV=1$,则 2 位从发计数器可以连续生成 4 个不同的地址。如果超过 4 个 CP 周期仍保持 $ADV=1$ (不读入新的外部地址),则按从发计数器循环产生的地址进行读/写操作。

读操作时,控制信号和地址输入在 CP 的上升沿被取样并存入相应寄存器,地址寄存器的输出和从发计数器产生的 A_1A_0 一并送入地址译码电路译码,同时,在读写控制逻辑电路和输出使能 \overline{OE} 的共同作用下,在下一个 CP 有效沿到来前,存储阵列中的数据被送到 I/O 数据线上输出,也就是说, I/O 线上的数据是前一个 CP 沿作用的结果。在下一个 CP 有效沿作用后,从发计数器产生 A_1A_0 的下一个地址,相应单元的数据被送到 I/O 数据线上,以此类推。

当在 CP 的上升沿取样到 $\overline{WE}=0$ 时为写操作,在接下来的 CP 上升沿才将 I/O 数据线上的数据锁存到输入寄存器中,并由读写控制逻辑电路控制输入驱动电路,将输入寄存器中的数据写入存储阵列。地址的生成与读操作相同。写操作时,读写控制逻辑电路将自动屏蔽输出使能信号

^① 在同步的静态和动态 RAM 中均有从发模式。

\overline{OE} ,使三态输出缓冲器呈现高阻态。

当 $ADV=0$ 时,为一般模式读写,此时 A_1A_0 不受从发控制逻辑电路影响,直接送至地址译码电路,同步 SRAM 按外部给定的地址进行读/写。

目前,同步 SRAM 已广泛应用于各种同步工作的数字系统中,特别是与处理器一同工作的系统。如个人电脑中的超高速缓冲存储器(Cache)。

2. 其他同步 SRAM

高速、高密度、低功耗早已成为 RAM 发展的永恒主题。在同步 SRAM 之后,各大 RAM 厂商又先后开发出双倍数据传输率(Double Data Rate,DDR)SRAM 和四倍数据传输率(Quad Data Rate,QDR)SRAM。

上述同步 SRAM 只在时钟的上升沿传输数据,并且共用读/写数据总线,读和写只能分时进行。这种同步 SRAM 也称为单倍数据传输率(Single Data Rate,SDR)同步 SRAM。DDR 同步 SRAM 是在同步 SRAM 基础上经过改进,在每个时钟周期的上升沿和下降沿各传输一次数据,这样数据传输效率提高了一倍,但是读写仍不能同时进行。

QDR 同步 SRAM 进一步改进了结构,为读和写操作分别提供独立的接口,不但在每个时钟周期的上升沿和下降沿共传输两次数据,而且每次读写能够同时进行,避免了数据总线的争抢,使数据传输效率比同步 SRAM 提高了两倍。

对 DDR 和 QDR 某些性能进行改善后的产品称为 DDR II 和 QDR II 同步 SRAM。目前,Cypress 公司采用 $0.065\text{ }\mu\text{m}$ 工艺技术生产的同步 SRAM 最高容量已达 144 MBit ,最高时钟工作频率达到 550 MHz 。

表 7.2.5 中列出了几种 SRAM 产品的几个主要指标。

表 7.2.5 几种 SRAM 产品的主要指标

型号	类型	容量 (密度)	字×位	存取时间 ns	时钟频率 MHz	工作 电压	工作 电流	厂家
M68AF031A	SRAM	256 KBit	32K×8	55	—	5V	50mA	STMicroelectronics
K6X0808C1D	SRAM	256 KBit	32K×8	55	—	5V	25mA	Samsung
CY7C1062DV33	SRAM	16 MBit	512K×32	10	—	3.3V	150mA	Cypress
IDT71V3556	SSRAM	4.5MBit	128K×36	5	200	3.3V	400mA	IDT
μ PD4481181GF-A65	SSRAM	8MBit	512K×16	6.5	133	3.3V	250mA	NEC
CY7C1308CV25	DDR	9MBit	256K×36	6	167	2.5V	650mA	Cypress
μ PD44164185F5-E40	DDR II	18MBit	1M×18	4	250	1.8V	650mA	NEC
CY7C1513V18	QDR II	72MBit	4M×18	4	250	1.8V	—	Cypress

注:对于 DDR,DDR II 和 QDR II,工作电压指核心工作电压。

7.2.3 DRAM

1. DRAM 存储单元

图 7.2.2 所示的 SRAM 存储单元由 6 个 MOS 管构成, 所用的管子数目多、功耗大, 集成度受到限制, 动态 RAM 克服了这些缺点。动态 RAM 也称为 DRAM, 其存储单元由一个 MOS 管和一个容量较小的电容器构成, 如图 7.2.6 中虚线框内所示。它利用电容器的电荷存储效应来存储数据 0 或 1。当电容 C 充有电荷、呈现高电压时, 相当于存有 1 值, 反之为 0 值。MOS 管 T 相当于一个开关, 当行选择线为高电平时, T 导通, C 与位线连通, 反之则断开。由于电路中漏电流的存在, 电容器上存储的数据(电荷)不能长久保存, 因此必须定期给电容补充电荷, 以免存储数据丢失, 这种操作称为刷新(Refresh)或再生。

写操作时, 行选线 X 为高电平, T 导通, 电容器 C 与位线 B 连通。同时读写控制信号 \overline{WE} 为低电平, 输入缓冲器被选通, 数据 D_1 经缓冲器和位线写入存储单元(外部输入/输出引脚上的数据经列选通电路送至 D_1)。图中未画出列选通电路。如果 D_1 为 1, 则向电容器充电, 反之电容器放电。未选通的缓冲器呈高阻态。

读操作时, 行选线 X 为高电平, T 导通, 电容器 C 与位线 B 连通。此时读写控制 \overline{WE} 为高电平, 输出缓冲器/灵敏放大器被选通, C 中存储的数据通过位线和缓冲器由 D_0 输出(D_0 再经列选通电路送至最终的输出引脚)。由于读出时会消耗 C 中的电荷, 存储的数据被破坏, 故每次读出后, 必须及时对读出单元刷新, 即此时刷新控制 R 也为高电平, 则读出的数据又经刷新缓冲器和位线对电容器 C 进行刷新。

除了读、写操作可以对存储单元进行刷新外, 刷新操作也可以通过只选通行选择线来实现。例如, 当行选线 X 为高电平, 且 \overline{WE} 亦为高电平时, C 上的数据经 T 到达位线 B, 然后经输出缓冲器和刷新缓冲器对存储单元刷新, 此时的刷新是整行刷新。由于未进行列选通, 所以 D_0 的数据不会送至最终的输出引脚。实际上, 刷新操作时输出缓冲器和刷新缓冲器环路构成一正反馈, 如果位线为高电平, 则将位线电平拉向更高; 反之则使位线电平降得更低。

由于存储单元电容的容量很小, 所以在位线容性负载较大时, C 中存储的电荷(C 存有 1 时)可能还未将位线拉至高电平时便耗尽了, 由此出现读出错误。为避免出现这种情况, 通常在读之前先将位线电平预置为高、低电平的中间值。这样, T 导通时, 根据电容 C 存储的 0 还是 1, 会将位线拉向低电平或高电平。位线电平的这种变化经灵敏放大器放大, 可以准确得到 C 所存储的逻辑值。

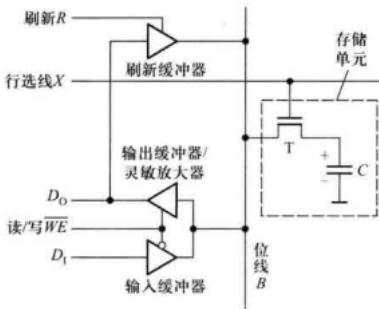


图 7.2.6 动态存储单元及基本操作原理

2. DRAM 的基本结构和操作时序

由于 DRAM 的集成度很高, 存储容量大, 因此需要较多的地址线。为减少引线数目, DRAM 大都采用行、列地址分时送入的方法。例如, 对于一个 1M 字的存储器, 有 2^{20} 个地址, 即有 20 根地址线。采用行、列地址分时送入时, 只需要 10 根地址线。DRAM 的基本结构如图 7.2.7 所示, 其内部设有行、列两个地址寄存器。行、列地址分别由行地址选通信号 \overline{RAS} (Row Address Strobe) 和列地址选通信号 \overline{CAS} (Column Address Strobe) 控制, 送入各自的寄存器。此外, DRAM 内部还设有刷新计数器和刷新控制及定时电路, 由此可以自动产生行地址进行刷新。

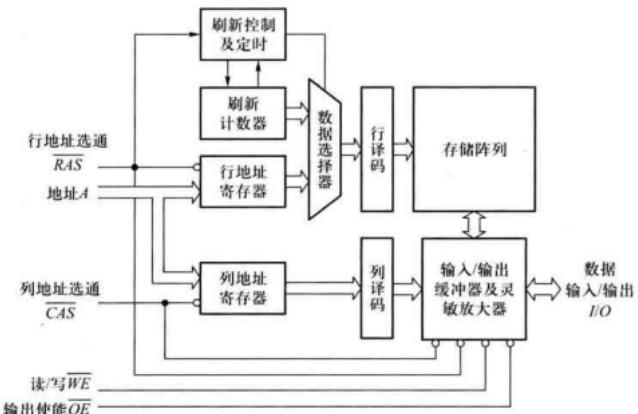


图 7.2.7 DRAM 的基本结构

DRAM 的操作方式比 SRAM 要复杂些, 这里只举出几种典型操作。

读/写操作

读/写操作时, 首先 \overline{RAS} 和 \overline{CAS} 先后变为低电平, 将行和列地址分别送入相应地址寄存器。然后在读/写控制信号 \overline{WE} 作用下完成读/写操作。读操作时, 输出使能 \overline{OE} 应为低电平。操作时序如图 7.2.8(a) 所示。

页模式操作

所谓“页”是指同一行的所有列构成的存储单元。页模式下的读/写操作与一般读/写操作的差别在于不改变行地址, 而只改变列地址。但行地址选择 \overline{RAS} 必须始终保持低电平。页模式可以显著提高读写速度, 其读操作时序如图 7.2.8(b) 所示。

\overline{RAS} 只刷新操作

该操作只刷新行地址指定行的所有存储单元, 不进行任何实际的读/写操作。在整个操作周

期 \overline{CAS} 要保持高电平, 其时序如图 7.2.8(c) 所示。该操作一次只刷新一行, 且需要外部地址计数器提供刷新地址。

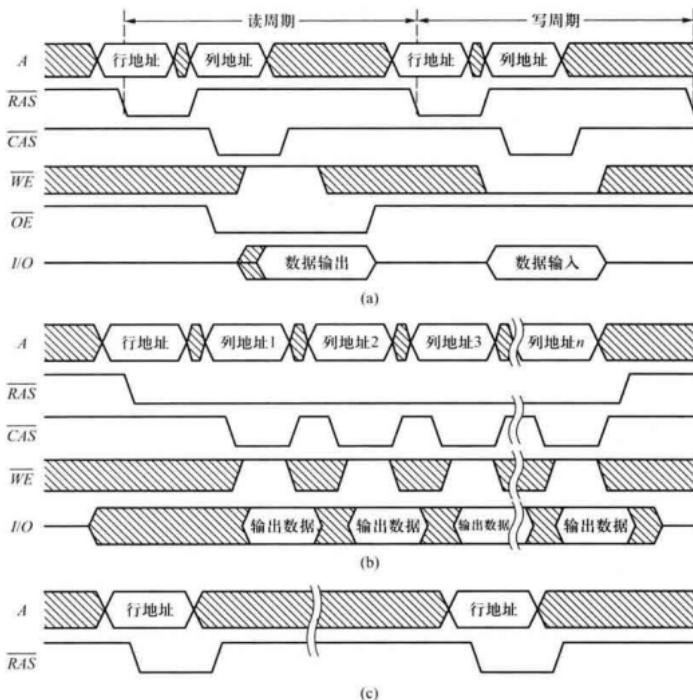


图 7.2.8 DRAM 操作定时图

(a) 读/写操作 (b) 页模式读操作 ($\overline{OE} = 0$) (c) RAS 只刷新操作 ($\overline{CAS} = \overline{WE} = 1$)

\overline{CAS} 先于 \overline{RAS} 有效的刷新操作

执行该操作时, \overline{CAS} 首先变为低电平, 然后 \overline{RAS} 变为低电平。此时, DRAM 内部的刷新控制及定时电路, 控制刷新计数器连续生成刷新地址进行刷新操作。

一般的 DRAM 每行刷新的间隔时间为 $15.6\text{ }\mu\text{s}$ (目前也有 $7.8\text{ }\mu\text{s}$ 的), 典型的刷新操作时间小于 100 ns 。刷新时间只占刷新周期的 0.64% , 所以 DRAM 用于读/写操作的时间实际上超过 99% 。

与 SRAM 的发展类似, DRAM 也有同步 DRAM (Synchronous DRAM, SDRAM)、DDR 同步

DRAM 和 QDR 同步 DRAM。由于 DRAM 的存储单元结构简单,其集成度远高于 SRAM,所以同等容量情况下,DRAM 更廉价。

目前,改进型 DDR II(二代)和 DDR III(三代)同步 DRAM 已成为个人电脑的主流内存。

7.2.4 存储容量的扩展

目前,尽管各种容量的存储器产品已经很丰富,且最大容量已达 1GBit 以上,用户能够方便地选择满足需要的芯片。但是,只用单个芯片不能满足存储容量要求的情况仍然存在。个人电脑中的内存条就是一个典型的例子,它由焊在一块印刷电路板上的多个芯片组成。此时,便涉及存储容量的扩展问题。

扩展存储容量的方法可以通过增加字长(位数)或字数来实现。

1. 字长(位数)的扩展

通常 RAM 芯片的字长为 1 位、4 位、8 位、16 位和 32 位等。当实际存储器系统的字长超过 RAM 芯片的字长时,需要对 RAM 实行位扩展。

位扩展可以利用芯片的并联方式实现,即将 RAM 的地址线、读/写控制线和片选信号对应地并联在一起,而各个芯片的数据输入/输出端作为字的各个位线。如图 7.2.9 所示,用 4 个 4K×4 位 RAM 芯片可以扩展成 4K×16 位的存储系统。

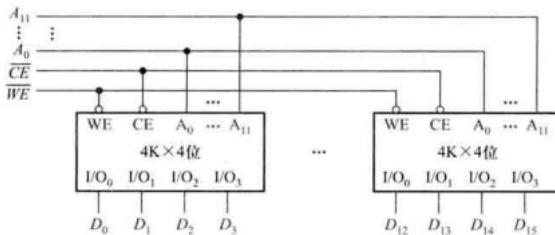
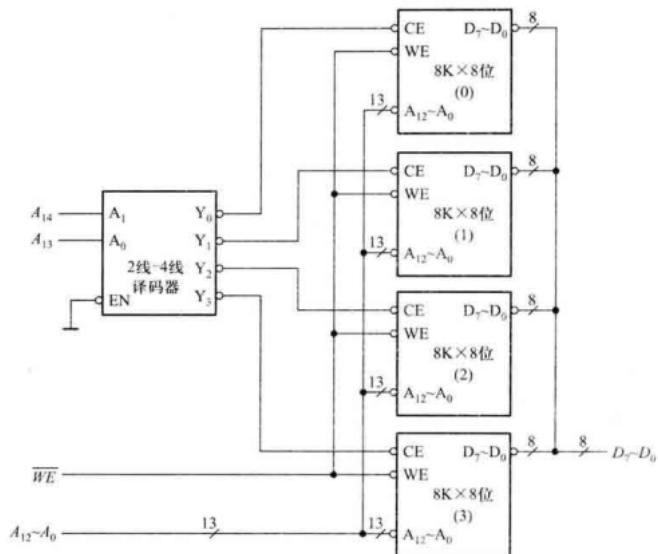


图 7.2.9 用 4K×4 位 RAM 芯片构成 4K×16 位的存储器系统

2. 字数的扩展

字数的扩展可以利用外加译码器控制存储器芯片的片选使能输入端来实现。例如,利用 2 线-4 线译码器将 4 个 8K×8 位的 RAM 芯片扩展为 32K×8 位的存储器系统。扩展方式如图 7.2.10 所示。图中,存储器扩展所要增加的地址线 A_{14} 、 A_{13} 与译码器的输入相连,译码器的输出 $Y_0 \sim Y_3$ 分别接至 4 片 RAM 的片选信号控制端 CE 。这样,当输入一个地址码 ($A_{12} \sim A_0$) 时,只有一片 RAM 被选中,从而实现了字的扩展。

实际应用中,常将两种方法相互结合,以达到同时扩展字和位的要求。可见,无论需要多大容量的存储器系统,均可利用容量有限的存储器芯片,通过位数和字数的扩展来构成。

图 7.2.10 用 $8K \times 8$ 位 RAM 芯片构成 $32K \times 8$ 位的存储器系统

7.2.5 RAM 应用举例

4.4 节中的例 4.4.6 已经介绍了七段显示器构成的 4 位动态显示电路的工作原理。实际上，目前应用更多的是 LED 点阵显示屏。这里结合 LED 阵列动态显示电路，介绍 RAM 在其中的应用。

一种 LED 阵列动态显示原理电路如图 7.2.11 所示。显示屏由 8×32 个 LED 构成，行线和列电平控制交叉点上 LED 的“亮”和“灭”。行线的电平由 RAM 中读出的数据经三态驱动器控制，而列线的电平由 5 线-32 线译码器控制。二进制计数器的输出一方面控制译码器，进行自左而右的逐列扫描；另一方面为 RAM 提供地址，将数据读出。三态缓冲器用于 RAM 内容更新时的数据输入。正常显示时，数据选择器将计数器输出提供给 RAM 作为地址。更新 RAM 内容时，将外部地址提供给 RAM。

当更新控制信号 $\overline{NEW}=1$ 时，电路处于动态显示工作状态。此时，三态缓冲器输出高阻，三态驱动器工作，RAM 处于读操作状态。数据选择器 $S=1$ ，使 $Y_4 \sim Y_0 = B_4 \sim B_0$ ，即将计数器的输出作为 RAM 的地址，RAM 输出的 8 位数据经三态驱动器驱动显示屏的 8 根行线。与此同时，计数器的输出 $Q_4 \sim Q_0$ 也通过译码器使显示屏的 1 根列线为低电平，这样，高电平行线与该低电平列

线交叉点上的 LED 被点亮,而低电平行线与该列线交叉点上的 LED 则不亮,从而控制 1 列 LED 的显示。例如,图 7.2.11 的显示屏要显示“CALL”时,计数器输出 $Q_4 \sim Q_0 = 00000$ 时,使译码器 $\bar{Y}_0 = 0$,同时通过数据选择器使 RAM 地址也为 00000,该地址单元的数据 00111000 分别控制 \bar{Y}_0 列 O_7, O_6, O_2, O_1, O_0 行的 LED 为“灭”状态; O_5, O_4, O_3 行的 LED 为“亮”状态。类似地,在下一个 CP 到来时,计数器加 1 使 $Q_4 \sim Q_0 = 00001$,则 $\bar{Y}_1 = 0$,同时 RAM 中 00001 地址单元的数据 00111110 控制 \bar{Y}_1 列的显示,即该列的 O_5, O_4, O_3, O_2, O_1 行“亮”。以此类推,在 CP 作用下,显示屏自左向右进行逐列扫描。完成一屏扫描后,在下一个 CP 到来时,计数器输出 $Q_4 \sim Q_0$ 由 11111 变为 00000,开始下一屏扫描,如此循环往复。

由上述 LED 阵列动态显示原理可知,显示屏上要显示的字符,以点阵方式存储在 RAM 中,每个点占用 1 个存储单元。需要发光的点存为 1,不发光的点存为 0。因此,8 行×32 列的显示阵列需要 32×8 位 RAM 存放一屏的数据。根据显示字符形状确定 RAM 中需要存储的点阵数据。如要显示“CALL”时,RAM 中的点阵数据应如表 7.2.6 所示。如果向 RAM 中写入不同的点阵数据,则可显示不同的字符。

表 7.2.6 显示“CALL”时 RAM 中的点阵数据

地址 $A_4 A_3 A_2 A_1 A_0$	内容 $D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0$	地址 $A_4 A_3 A_2 A_1 A_0$	内容 $D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0$
0 0 0 0 0	0 0 1 1 1 0 0 0	1 0 0 0 0	1 1 1 1 1 1 1 0
0 0 0 0 1	0 1 1 1 1 1 0 0	1 0 0 0 1	1 1 1 1 1 1 1 0
0 0 0 1 0	1 1 0 0 0 1 1 0	1 0 0 1 0	1 0 0 0 0 0 0 0
0 0 0 1 1	1 0 0 0 0 0 1 0	1 0 0 1 1	1 0 0 0 0 0 0 0
0 0 1 0 0	1 0 0 0 0 0 0 1 0	1 0 1 0 0	1 0 0 0 0 0 0 0
0 0 1 0 1	1 1 0 0 0 1 1 0	1 0 1 0 1	1 0 0 0 0 0 0 0
0 0 1 1 0	0 1 0 0 0 1 0 0	1 0 1 1 0	1 0 0 0 0 0 0 0
0 0 1 1 1	0 0 0 0 0 0 0 0	1 0 1 1 1	0 0 0 0 0 0 0 0
0 1 0 0 0	1 1 1 1 0 0 0 0	1 1 0 0 0	1 1 1 1 1 1 1 0
0 1 0 0 1	1 1 1 1 1 0 0 0	1 1 0 0 1	1 1 1 1 1 1 1 0
0 1 0 1 0	0 0 1 0 1 1 0 0	1 1 0 1 0	1 0 0 0 0 0 0 0
0 1 0 1 1	0 0 1 0 0 1 1 0	1 1 0 1 1	1 0 0 0 0 0 0 0
0 1 1 0 0	0 0 1 0 1 1 0 0	1 1 1 0 0	1 0 0 0 0 0 0 0
0 1 1 0 1	1 1 1 1 1 0 0 0	1 1 1 0 1	1 0 0 0 0 0 0 0
0 1 1 1 0	1 1 1 1 0 0 0 0	1 1 1 1 0	1 0 0 0 0 0 0 0
0 1 1 1 1	0 0 0 0 0 0 0 0	1 1 1 1 1	0 0 0 0 0 0 0 0

当 $\overline{NEW}=0$ 时, 电路处于 RAM 内容更新状态。此时, 更新数据经三态缓冲器送至 RAM 的数据端口, 三态驱动器为高阻, RAM 处于写操作状态。数据选择器 $S=0$, 使 $Y_4 \sim Y_0 = A_4 \sim A_0$, 即 RAM 的地址 $A_4 \sim A_0$ 来自外部更新地址, 从而将新数据写入 RAM 中。只要能保证更新内容在一个 CP 周期内全部写入 RAM 中, 就不会对动态显示造成影响(图 7.2.11 中未给出更新控制电路)。实际上, 如果采用双口 RAM 代替图 7.2.11 中的 RAM, 就可以省去三态缓冲器和数据选择器, 并且显示内容的更新和显示屏的动态显示也可以各自独立工作而互不相关。

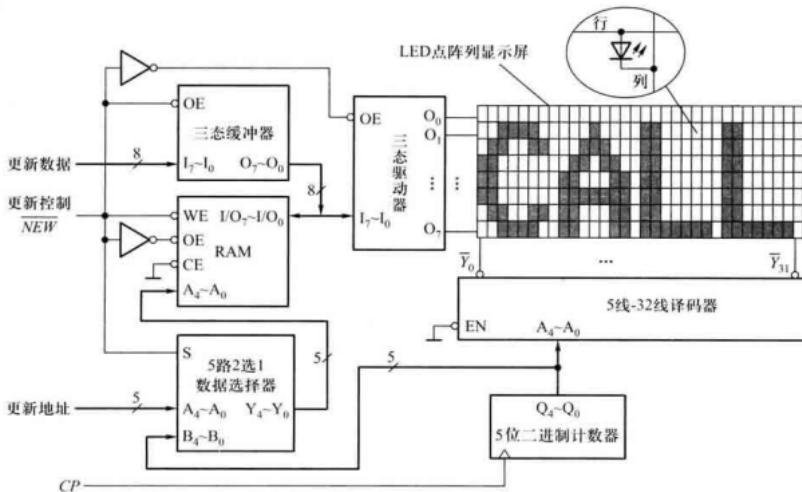


图 7.2.11 8×32 LED 点阵动态显示原理

若动态显示扫描频率为 31.25 Hz, 即扫描周期为 32 ms, 则扫描 32 列显示屏时每列持续时间为 1 ms。即要求 CP 脉冲的周期为 1 ms。更新显示内容时, 只要在 1 ms 内完成 RAM 中 32 个地址单元点阵数据的写入即可。

除了可以显示字符外, LED 点阵显示器还可以显示各种图案。只要向 RAM 中写入图案的点阵数据, LED 阵列便可显示出希望的图案。

复习思考题

7.2.1 DRAM 中存储的数据如果不进行周期性的刷新, 其数据将会丢失; 而 SRAM 中存储的数据无需刷新, 只要电源不断电就可以永久保存, 为什么?

7.2.2 一般情况下, DRAM 的集成度比 SRAM 的集成度高, 为什么?

7.2.3 同步 SRAM 与异步 SRAM 的主要差别是什么? 它有什么优点?

7.2.4 同步 SRAM 中从发模式读/写操作的特点是什么?

7.2.5 用容量为 $16K \times 1$ 位存储器芯片构成一个 $32K \times 8$ 位的存储系统, 需要多少根地址线? 多少根数据线? 多少个 $16K \times 1$ 位的存储器芯片?

小 结

- 半导体存储器是现代数字系统特别是计算机中的重要组成部分, 它可分为 ROM 和 RAM 两大类, 属于 MOS 工艺制成的大规模集成电路。

- ROM 是一种非易失性的存储器, 它存储的是固定数据, 一般只能被读出。根据数据写入方式的不同, ROM 又可以分为固定 ROM 和可编程 ROM。可编程 ROM 又可以细分为 PROM、EPROM、 E^2PROM 和快闪存储器等。特别是 E^2PROM 和快闪存储器可以进行电擦写, 已兼有了 RAM 的特性。

- RAM 是一种时序逻辑电路, 具有记忆功能。它存储的数据随电源断电而消失, 因此是一种易失性的读写存储器。它包含有 SRAM 和 DRAM 两种类型, 前者用触发器记忆数据, 后者靠 MOS 管栅极电容存储数据。因此, 在不停电的情况下, SRAM 的数据可以长久保持, 而 DRAM 则必须定期刷新。

- 无论是 SRAM 还是 DRAM, 目前都有在时钟脉冲作用下工作的同步 RAM (SSRAM 和 SDRAM), 而且同步 RAM 已成为主流存储器。在此基础上发展起来的 DDR、DDR II 和 QDR 等 RAM 也已愈来愈多地应用于计算机内存、显存和通信设备中。

习 题

7.1 只读存储器

7.1.1 指出下列存储系统各具有多少个存储单元, 至少需要几根地址线和数据线。

- (1) $64K \times 1$ (2) $256K \times 4$ (3) $1M \times 1$ (4) $128K \times 8$

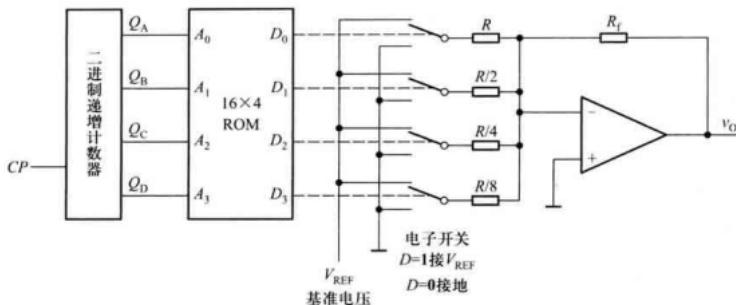
7.1.2 设存储器的起始地址为全 0, 试指出下列存储系统的最高地址的十六进制地址码为多少?

- (1) $2K \times 1$ (2) $16K \times 4$ (3) $256K \times 32$

7.1.3 试确定用 ROM 实现下列逻辑函数时所需的容量: (1) 实现两个 3 位二进制数相乘的乘法器; (2) 将 8 位二进制数转换成十进制数(用 BCD 码表示)的转换电路。

7.1.4 用一片 128×8 位的 ROM 实现各种码制之间的转换。要求用从第 0 个地址单元(全 0 地址码)开始的前 16 个地址单元中的 10 个, 实现 8421BCD 码到余 3 码的转换; 用从第 17 个地址单元开始接下来的 16 个单元中的 10 个, 实现余 3 码到 8421BCD 码的转换。试求: (1) 列出 ROM 的地址与内容对应关系的真值表; (2) 确定输入变量和输出变量与 ROM 地址线和数据线的对应关系; (3) 简要说明将 8421BCD 码的 0101 转换成余 3 码和将余 3 码的 1001 转换成 8421BCD 码的过程。

7.1.5 利用 ROM 构成的任意波形发生器如图题 7.1.5 所示, 改变 ROM 的内容, 即可改变输出波形。当 ROM 的内容如表题 7.1.5 所示时, 画出输出端随 CP 变化的波形。



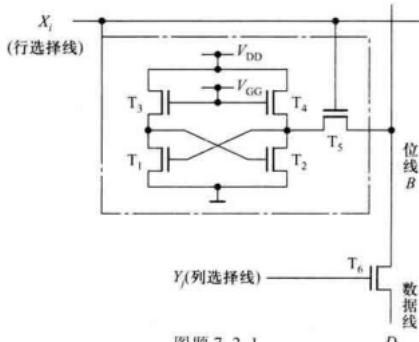
图题 7.1.5

表题 7.1.5

$A_3 A_2 A_1 A_0$	$D_3 D_2 D_1 D_0$	$A_3 A_2 A_1 A_0$	$D_3 D_2 D_1 D_0$
0 0 0 0	0 1 0 0	1 0 0 0	0 1 0 0
0 0 0 1	0 1 0 1	1 0 0 1	0 0 1 1
0 0 1 0	0 1 1 0	1 0 1 0	0 0 1 0
0 0 1 1	0 1 1 1	1 0 1 1	0 0 0 1
0 1 0 0	1 0 0 0	1 1 0 0	0 0 0 0
0 1 0 1	0 1 1 1	1 1 0 1	0 0 0 1
0 1 1 0	0 1 1 0	1 1 1 0	0 0 1 0
0 1 1 1	0 1 0 1	1 1 1 1	0 0 1 1

7.2 随机存取存储器

7.2.1 一个 CMOS 存储单元如图题 7.2.1 所示, 试分析其工作原理。



图题 7.2.1

7.2.2 设同步 SRAM 工作在从发模式下,若数据写入的首地址为 0094H,问接下来的 3 个数据将被写入的存储单元的地址分别是多少?

7.2.3 一个有 4096 位的 DRAM,如果存储矩阵为 64×64 结构形式,且每个存储单元刷新时间为 100 ns,则存储单元全部刷新一遍最快需要多长时间?如果刷新每行的最长间隔时间为 15.6 μ s,则该 DRAM 的刷新周期最长为多少?刷新操作所用时间占刷新周期的百分比是多少?

7.2.4 一个有 $1M \times 1$ 位的 DRAM,采用地址分时送入的方法,芯片应具有几根地址线?

7.2.5 试用具有片选使能 \overline{CE} 、输出使能 \overline{OE} 、读/写控制 \overline{WE} 、容量为 $8K \times 8$ 位的 SRAM 芯片,设计一个 $16K \times 16$ 位的存储器系统,试画出其逻辑图。

7.2.6 在图 7.2.11 的 LED 点阵列字符动态显示电路中,(1)若人的视觉暂留时间为 0.05 s,在满足 LED 阵列图像稳定不闪烁的情况下,试计算 CP 脉冲的最低工作频率。(2)若要在显示屏上显示“HELP”,试确定 RAM 中存放的点阵数据。(3)若 LED 阵列改为 16 行 \times 128 列,则需要 RAM 的位数为多少?容量是多少?

CPLD 和 FPGA

引
言

前面有关章节已经介绍了低密度(或简单)的可编程逻辑器件(PLD),如PAL和GAL。与中、小规模逻辑集成器件相比,PLD在集成度、功耗和系统可靠性等方面有显著的优势。但是对于规模更大、更复杂的数字系统,低密度PLD仍然相形见绌。目前,采用集成度更高、功能更复杂的CPLD和FPGA设计实现大型数字系统已成为普遍应用的方法。

由于实际的CPLD和FPGA内部电路规模庞大且非常复杂,所以本章仅从基本原理和一般性问题出发,简要介绍CPLD和FPGA的基本结构、逻辑功能编程实现原理和一般开发过程。以期读者对此有一定了解,并作为进一步学习和运用它们开发数字电路或系统的基础知识。

8.1 复杂可编程逻辑器件(CPLD)简介

前面有关章节已介绍了可编程逻辑器件PAL和GAL,它们都属于简单的PLD。随着微电子技术的发展和应用上的需求,简单PLD在集成度和性能方面难以满足要求,因此集成度更高、功能更强的CPLD便迅速发展起来。

早期的CPLD大多采用EPROM编程技术,其编程过程与简单PLD一样,每次编程需要在专用或通用设备上进行。后来采用E²PROM和快闪存储器技术,使CPLD具有了“在系统可编程(In System Programmability,ISP)”特性。所谓在系统可编程是指未编程的ISP器件可以直接焊接在印制电路板上,然后通过计算机的数据传输端口和专用的编程电缆对焊接在电路板上的ISP器件直接多次编程,从而使器件具有所需要的逻辑功能。这种编程不需要使用专用的编程器,因为已将原来属于编程器的编程电路和升压电路集成在ISP器件内部。ISP技术使得调试过程不需要反复拔插芯片,从而不会产生引脚弯曲变形现象,提高了可靠性,而且可以随时对焊接在电路板上的ISP器件的逻辑功能进行修改,从而加快了数字系统的调试过程。目前,ISP已成

为系统在线远程升级的技术手段。

与简单 PLD(PAL、GAL 等)相比,CPLD 的集成度更高,具有的输入、乘积项和宏单元更多。图 8.1.1 是一般 CPLD 器件的结构框图。其中逻辑块^①就相当于一个 GAL 器件(见 6.6 节),CPLD 中有多个逻辑块,这些逻辑块之间可以使用可编程内部连线实现相互连接。为了增强对 I/O 的控制能力,提高引脚的适应性,CPLD 中还增加了 I/O 控制块。每个 I/O 块中有若干个 I/O 单元。

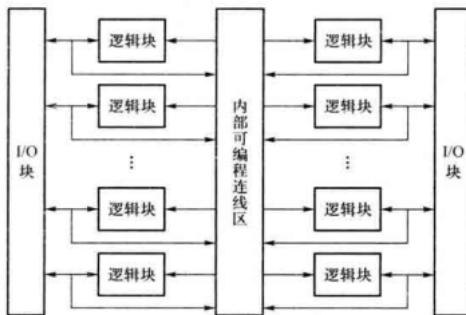


图 8.1.1 CPLD 的结构框图

1. 逻辑块

由前述知识已知,用与-或阵列和触发器可以实现任何组合或时序逻辑函数,这便是 CPLD 实现逻辑功能电路的基本原理。逻辑块就是 CPLD 实现逻辑功能的核心模块。逻辑块的构成如图 8.1.2 所示。它主要由可编程乘积项阵列(即与阵列)、乘积项分配、宏单元三部分组成。与 GAL 相比,逻辑块中增加了乘积项分配电路。对于不同厂商、不同型号的 CPLD,逻辑块中乘积项的输入变量个数 n 和宏单元个数 m 不完全相同。例如,Xilinx 公司的 XC9500 系列中,乘积项输入变量有 36 个,宏单元为 18 个。Altera 公司的 MAX7000 系列有 36 个乘积项输入变量,16 个宏单元。

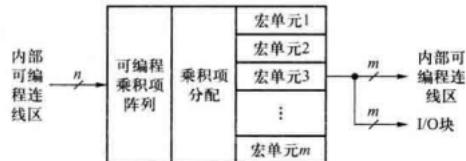


图 8.1.2 逻辑块的构成

^① 各公司 CPLD 中的逻辑块名称不一。如 Function Block, FB(Xilinx), Logic Array Block, LAB(Altera), Generic Logic Block, GLB(Lattice)。

(1) 可编程乘积项阵列

逻辑块中的可编程乘积项阵列与 GAL 中的相似,只是规模上有所差别。此处,乘积项阵列有 n 个输入,可以产生 n 变量的乘积项。一般一个宏单元对应 5 个乘积项,这样,在逻辑块中共有 $5 \times m$ 个乘积项。例如,XC9500 系列的逻辑块中有 90 个 36 变量乘积项,MAX7000 系列的逻辑块中有 80 个 36 变量乘积项。

(2) 乘积项分配

乘积项分配电路是由可编程的数据选择器和数据分配器构成。我们知道,在 GAL 中一个宏单元对应的乘积项是固定的,而在 CPLD 的逻辑块中并不是这样。这里,乘积项阵列中的任何一个乘积项都可以通过可编程的乘积项分配电路,分配到任意一个宏单元中,从而增强了逻辑功能实现的灵活性。在 XC9500 系列 CPLD 中,理论上可以将 90 个乘积项组合到一个宏单元中,产生 90 个乘积项的与-或式,但此时其余 17 个宏单元将不能使用乘积项了。在 Altera 公司生产的 CPLD 中,还有乘积项共享电路,使得同一个乘积项可以被多个宏单元同时使用。

(3) 宏单元

此处的宏单元与 GAL 中的类似,其中包含一个或门、一个触发器和一些可编程的数据选择器及控制门。或门用来实现与-或阵列的或运算。通过对宏单元编程可实现组合逻辑输出、寄存器输出、清零、置位等工作方式。宏单元的输出不仅送至 I/O 单元,还送到内部可编程连线区,以被其他逻辑块使用。

2. 可编程内部连线

可编程内部连线纵横交错地分布在 CPLD 中,其作用是实现逻辑块与逻辑块之间、逻辑块与 I/O 块之间,以及全局信号到逻辑块和 I/O 块之间的连接。连线区的可编程连接也是基于 E²CMOS 单元编程实现的。

不同制造商对可编程内部连线区的称呼也不同,Xilinx 公司的称为 Switch Matrix(开关矩阵),Altera 公司的称为 PIA(Programmable Interconnect Array),Lattice 公司的称为 GRP(Global Routing Pool)。当然,它们之间存在一定的差别,但所承担的任务是相同的。这些连线的编程工作是由开发软件的布线程序自动完成的。

3. I/O 单元

I/O 单元是 CPLD 外部封装引脚和内部逻辑间的接口。每个 I/O 单元对应一个封装引脚,通过对 I/O 单元中可编程单元的编程,可将引脚定义为输入、输出和双向功能。CPLD 的 I/O 单元简化原理框图如图 8.1.3 所示。

I/O 单元中有输入和输出两条信号通路。当 I/O 引脚作输出时,三态输出缓冲器的输入信号来自宏单元,其使能控制信号 OE 由可编程数据选择器 M 选择其来源。其中,有多个全局输出使能控制信号,不同型号的器件,其数量也不同(XC9500 系列中 $r=4$, MAX7000 系列中 $r=6$)。当 OE 为低电平时,输出缓冲器为高阻,I/O 引脚可用作输入,引脚上的输入信号经过输入缓冲器送至内部可编程连线区。

图中 D_1 和 D_2 是钳位二极管,用于 I/O 引脚的保护。另外,通过编程可以使 I/O 引脚接上拉

电阻或接地,也可以控制输出摆率(转换速率 SR)。快速方式可适应频率较高的信号输出,慢速方式则可减小功耗和降低噪声。 V_{CCINT} 是器件内部逻辑电路的工作电压(也称为核心工作电压(Core Voltage)),而 V_{CCIO} 的引入,可以使 I/O 引脚兼容多种电源系统。

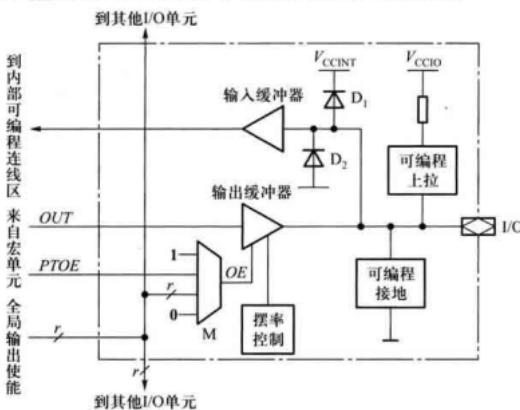


图 8.1.3 I/O 单元的简化结构图

与后面将要介绍的 FPGA 相比,尽管 CPLD 在电路规模和灵活性方面不如 FPGA,但是它的可加密性和传输延时预知性,使得 CPLD 仍广泛应用于数字系统设计中。

目前,各大生产厂商仍不断开发出集成度更高、速度更快、功耗更低的 CPLD 新产品,核心工作电源可以低到 1.8 V。表 8.1.1 列出了几种典型的 CPLD 产品。实际上,Altera 公司的 MAX II 系列 CPLD 中的逻辑块已不再是基于与-或阵列架构,而是基于与 FPGA 类似的 LUT 架构。

表 8.1.1 几种典型的 CPLD 产品

厂商	系列	型号	逻辑块	宏单元个数/ 每逻辑块	等价宏单 元总个数	逻辑块输入 宽度/Bit	最高工作 频率/MHz	工作电 压/V	最大可用 引脚数
Altera	MAX7000A	EPM7128AE	8	16	128	36	192.3	3.3	100
	MAX II	EPM2210G	221	10	1700	26	304	1.8	272
Xilinx	XC9500XL	XC95144XL	8	18	144	54	200	3.3	117
	CoolRunner-II	XC2C512	32	16	512	40	179	1.8	270
Lattice	ispMACH4000V	4128V	8	16	128	36	333	3.3	100
	ispXPLD 5000MX	51024MX	32	32	1024	68	250	1.8	381

复习思考题

8.1.1 CPLD 在结构上可分为哪几个部分？各部分的主要功能是什么？

8.1.2 CPLD 的可编程特性是基于什么编程技术？其有什么特点？

8.2 现场可编程门阵列(FPGA)

FPGA 是另一种可以实现更大规模逻辑电路的可编程器件。它不像 CPLD 那样采用可编程的“与-或”阵列来实现逻辑函数，而是采用曾在 4.4.3 节中介绍的查找表(LUT)工作原理来实现逻辑函数。这种逻辑函数实现原理避开了与-或阵列结构规模上的限制，使 FPGA 中可以包含数量众多的 LUT 和触发器，从而能够实现更大规模、更复杂的逻辑电路。FPGA 的编程机理也不同于 CPLD，它不是基于 E²PROM 或快闪存储器编程技术，而是采用 SRAM 实现电路编程。

随着生产工艺的进步，FPGA 的功能愈来愈强大，性价比愈来愈高，目前已成为数字系统设计的首选器件之一。

8.2.1 FPGA 中编程实现逻辑功能的基本原理

LUT 是 FPGA 实现逻辑函数的基本逻辑单元，它由若干存储单元和数据选择器构成。2 输入 LUT 的结构如图 8.2.1 所示，其中 M₀ ~ M₃ 为 4 个 SRAM 存储单元，他们存储的数据作为数据选择器的输入数据；LUT 的 2 个输入端作为数据选择器的选择信号。该 LUT 可以实现任意 2 变量组合逻辑函数。例如，若要实现逻辑函数 $L = A + \bar{B}$ ，则可列出其真值表如表 8.2.1 所示。用图 8.2.1 的 LUT 实现时，将变量 A 和 B 分别连接 S₁ 和 S₀，同时将真值表 8.2.1 中逻辑函数 L 的 0、1 值按由上到下的顺序分别存入 M₀ ~ M₃ 的 4 个 SRAM 单元中，得到图 8.2.2。此时，LUT 的输出便实现了该逻辑函数，即 $Y = L$ 。由此看出，只要改变 SRAM 单元 M₀ ~ M₃ 中的数据，就可以实现不同的逻辑函数。这就是 FPGA 可编程特性的具体体现，其逻辑功能的编程就好像向 RAM 中写数据一样容易（图中未画编程电路）。同时看到，LUT 相当于以真值表的形式实现给定的逻辑函数。

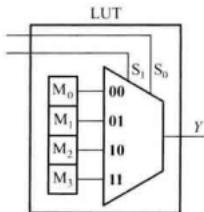


图 8.2.1 2 输入 LUT 结构图

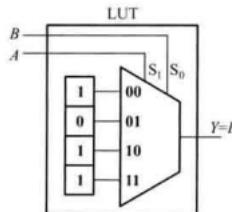


图 8.2.2 实现逻辑函数 L 的 LUT

表 8.2.1 L 的真值表

A	B	L
0	0	1
0	1	0
1	0	1
1	1	1

表 8.2.2 逻辑函数真值表

A	B	F_1	B	C	F_2	F_1	F_2	F
0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	0	1	1
1	0	0	1	0	0	1	0	1
1	1	1	1	1	0	1	1	1

在 FPGA 中实现该逻辑函数时需要完成以下编程任务:(1) 将 FPGA 的 I/O 引脚上的输入变量 A 和 B 通过可编程连线资源连接到 LUT 的 S_1 和 S_0 ; (2) 将真值表中 L 的函数值写入 LUT 中对应的 SRAM 单元中; (3) 将 LUT 的输出 Y 通过可编程连线资源连接到 FPGA 的 I/O 引脚上, 作为逻辑函数 L 的输出。

实际上, 上述的 LUT 也可以看作是一个 4×1 位的 SRAM, S_1S_0 为 2 位地址码输入。这样, 每输入一组 AB 信号进行逻辑运算, 就相当于输入一个地址进行查表, 找出地址对应的内容输出, 在 Y 端便得到该组输入信号逻辑运算的结果, 查找表(LUT)因此得名。这一原理也与 7.1.5 节 ROM 实现组合逻辑函数的原理相同。

目前 FPGA 中大多使用 4~5 个输入、1 个输出的 LUT。当变量数超过一个 LUT 的输入数时, 可以将多个 LUT 扩展连接以满足更多变量数的要求。下面还是以 2 变量 LUT 为例, 说明在 FPGA 中如何扩展连接实现一个 3 变量的逻辑函数。

用一个小规模的 FPGA 实现逻辑函数 $F = F_1 + F_2 = AB + \bar{B}C$, 该逻辑函数的真值表如表 8.2.2 所示。编程后 FPGA 中一部分逻辑块和连线资源的编程状态如图 8.2.3 所示。符号 \times 表示纵横线交叉点上的可编程开关接通。连线资源将 I/O 引脚上的输入 A, B, C 和输出 F 连接到内部逻辑电路中, 而内部逻辑块之间也通过连线资源实现连接。图中上方两个逻辑块被编程实现逻辑函数 $F_1 = AB$ 和 $F_2 = \bar{B}C$, 右下角的逻辑块实现 $F = F_1 + F_2$ 。逻辑块中的 **0, 1** 值表示 LUT 中 SRAM 单元的编程数据, 也就是表 8.2.2 中 F_1, F_2 和 F 的值。

在 FPGA 中, 实现组合逻辑功能的基本电路是 LUT, 而触发器仍然是实现时序逻辑功能的基本电路。在 LUT 的基础上再增加触发器, 便可构成既可实现组合逻辑又可实现时序逻辑的基本逻辑单元电路。FPGA 中就是由很多类似这样的基本逻辑单元来实现各种复杂逻辑功能的。

当用户通过原理图或 HDL 语言描述了一个逻辑电路后, FPGA 开发软件会自动计算逻辑电路的所有可能的结果(真值表), 并把结果写入 SRAM, 这一过程就是所谓的编程。此后, SRAM 中的内容始终保持不变, LUT 就具有了确定的逻辑功能。由于 SRAM 具有数据易失性, 即一旦断电, 其原有的逻辑功能将消失。所以 FPGA 一般需要一个外部的 PROM(或其他非易失性存储器)保存编程数据。上电后, FPGA 首先从 PROM 中读入编程数据进行初始化, 然后才开始正常

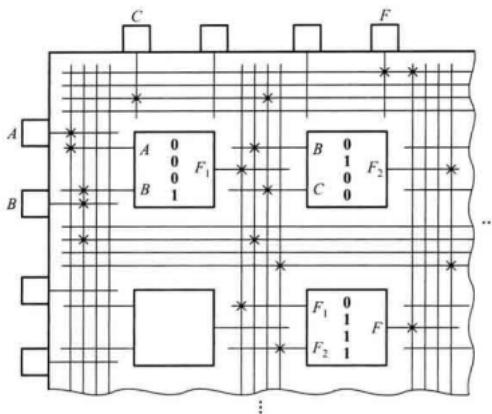


图 8.2.3 已被编程的 FPGA 的一部分

工作。由于 SRAM 中的数据理论上可以进行无限次写入,所以,基于 SRAM 技术的 FPGA 可以进行无限次的编程。

8.2.2 FPGA 的结构简介

目前,FPGA 产品种类繁多,各生产厂商的产品也各不相同。这里仅从构成 FPGA 的最基本原理出发,介绍 FPGA 的基本结构。在实际使用时,用户必须根据所选用的器件型号查阅相关数据手册。

FPGA 的一般结构如图 8.2.4 所示。它至少包含三种最基本资源:可编程逻辑块、可编程连线资源和可编程 I/O 模块。对比图 8.2.4 和图 8.1.1 CPLD 的结构,看似两者非常类似,但是 FPGA 与 CPLD 最大的差别在于实现逻辑功能的原理不同。FPGA 中逻辑块是以 LUT 为基础的,而 CPLD 中的逻辑块是以与-或阵列为基础的。另外编程技术也不同,CPLD 是基于 E²PROM 或快闪存储器的编程技术,而 FPGA 则是基于 SRAM 的编程技术。

FPGA 中的逻辑块排列成二维阵列。规模不同,FPGA 所含逻辑块的数量也不同。连线资源在逻辑块行和列之间纵横分布,它包括纵向和横向连线以及可编程开关(互连开关)。逻辑块用来实现所需要的逻辑功能;连线资源用来实现逻辑块与逻辑块之间、逻辑块与 I/O 块之间的连接;I/O 块是芯片外部引脚数据与内部数据进行交换的接口电路,通过编程可将 I/O 引脚设置成输入、输出和双向等不同的功能。

1. 可编程逻辑块

可编程逻辑块是 FPGA 实现各种逻辑功能的基本单元。为了能实现时序逻辑功能,逻辑块中除了包含 LUT 外,还附加了触发器和一些可编程数据选择器和必要的逻辑门。这些附加电路

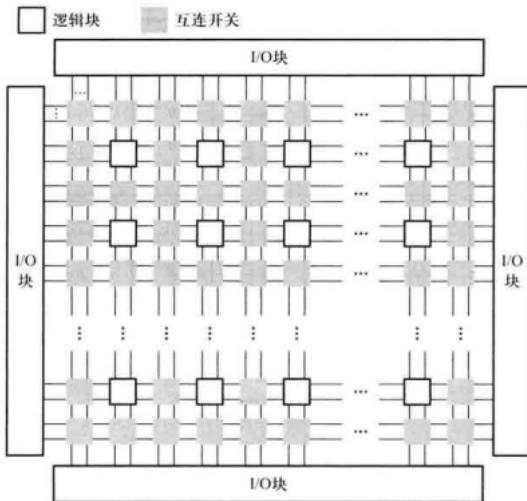


图 8.2.4 FPGA 结构示意图

的作用非常类似于 GAL 或 CPLD 中的宏单元。图 8.2.5 是 Xilinx 公司 Spartan-II 系列 FPGA 的逻辑块主要逻辑电路原理图,其中数据选择器除了 CY、F5、F6 外均为可编程选择器(未画 SRAM 单元)。由图看出,逻辑块有 15 个输入端和 8 个输出端。 $F_1 \sim F_4$ 和 $G_1 \sim G_4$ 为两组 4 变量逻辑函数输入端; SR, CE 和 CLK 为触发器的控制及时钟信号输入端; $FSIN$ 为级联输入端,用于 LUT 的级联扩展; BX, BY 为旁路输入端,信号由此进入逻辑块后可以绕过 LUT 直接传至输出; CIN 为进位链输入端,当与其他逻辑块实现多位数加法运算时, CIN 可作为进位输入。8 个输出包括 2 个组合逻辑输出端 X 和 Y ,2 个寄存器输出端 XQ 和 YQ ,2 个算术运算进位端 XB 和 YB ,1 个级联输出端 $F5$ 和 1 个进位链输出端 $COUT$ 。

通过对逻辑块中的 LUT 和相关数据选择器编程,可以实现所需要的组合逻辑电路。当逻辑电路规模较大时,可以级联扩展多个逻辑块来实现。如果触发器的输出,经可编程连线资源反馈给输入,再经 LUT 产生激励函数驱动触发器的 D 端,可以构成时序逻辑电路。例如,用图 8.2.5 实现 2 位二进制递增计数器时,由表 8.2.3 所示的 2 位二进制递增计数器状态转换,可得 D 触发器的激励方程为

$$Q_1^{n+1} = D_1 = \bar{Q}_1 Q_0 + Q_1 \bar{Q}_0$$

$$Q_0^{n+1} = D_0 = \bar{Q}_0$$

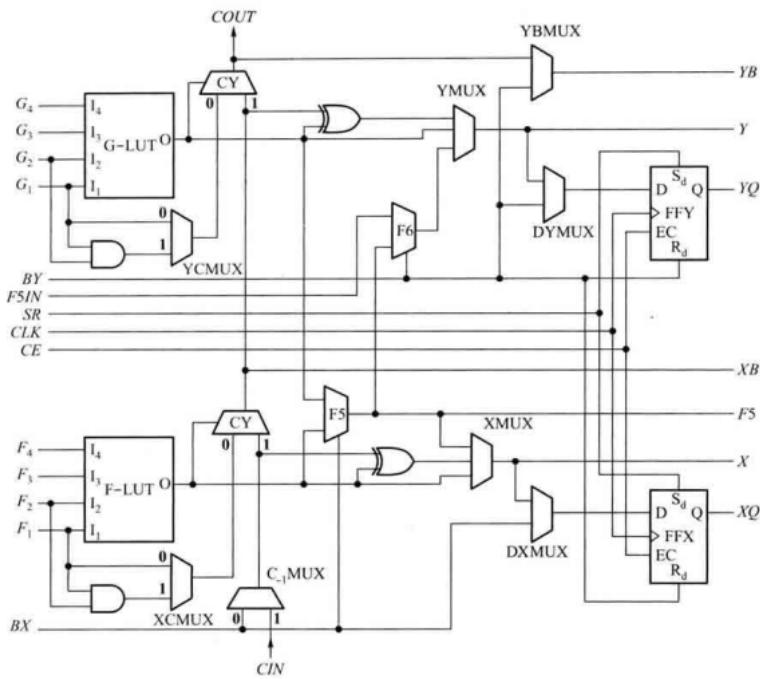


图 8.2.5 Spartan-II 系列逻辑块主要逻辑电路

表 8.2.3 2 位二进制递增计数器状态转换表

Q_1	Q_0	$Q_1^{n+1}(D_1)$	$Q_0^{n+1}(D_0)$
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

用 G-LUT 产生 D_1 , 仅需要用到 2 个输入变量, 高位的 2 个输入接地。此时, G-LUT 相当于图 8.2.1 的 2 输入 LUT, 所以仅需对其中 SRAM 单元的 $M_0 \sim M_3$ 编程。类似地, 用 F-LUT 产生 D_0 , 仅用到 1 个输入变量, F-LUT 相当于 1 个输入的 LUT, 所以仅需对它的 M_0 和 M_1 编程。图

8.2.6 就是对图 8.2.5 编程后, 实现 2 位二进制递增计数器的结果。

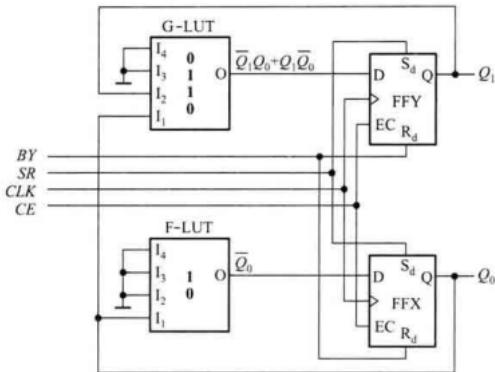


图 8.2.6 逻辑块编程后实现 2 位二进制计数器

由多个逻辑块便可构成更复杂的时序逻辑电路。

由于 4 输入的 LUT 相当于一个 16×1 位的 SRAM，所以逻辑块也可用来做存储器使用，不过此时 LUT 中的内容不再是预先配置好的，而是在正常工作时可以随时读写的，而且 LUT 不能再作为逻辑函数产生器使用。

2. I/O 块

I/O 块是 FPGA 外部封装引脚和内部逻辑间的接口。每个 I/O 单元对应一个封装引脚，通过对 I/O 单元编程，可将引脚分别定义为输入、输出和双向功能。I/O 单元的简化原理图如图 8.2.7 所示。图中的 V_{CC0} 和 V_{REF} 引脚与其他 I/O 单元共用。

I/O 单元中有输入和输出两条信号通路。当 I/O 引脚用作输出时，内部逻辑信号由 O 端进入 I/O 单元，由可编程数据选择器确定是直接送输出缓冲器还是经过触发器 OFF 寄存后再送输出缓冲器。输出缓冲器使能控制信号 T 可以直接控制输出缓冲器，也可以通过触发器 TFF 后再控制输出缓冲器。当 I/O 引脚用作输入时，引脚上的输入信号经过输入缓冲器，可以直接由 I 进入内部逻辑电路，也可以经触发器 IFF 寄存后由 IQ 输入到内部逻辑电路中。没有用到的引脚被预置为高阻态。

可编程延时电路可以控制输入信号进入的时机，保证内部逻辑电路协调工作。其最短延迟时间为零。

3 个触发器均可编程配置为边沿触发或电平触发方式，他们共用一个时钟信号 CLK，但有各自的时钟使能控制信号 EC。通过它们可以实现同步输入/输出。

图 8.2.7 中两个钳位二极管具有瞬时过压保护和静电保护作用。上拉、下拉电阻和弱保持电路（Weak-keeper Circuit）可通过编程配置给 I/O 引脚。弱保持电路监视并跟踪 I/O 引脚输入

电压的变化,当连至引脚总线上所有的驱动信号全部无效时,弱保持电路将维持在引脚最后一个状态的逻辑电平上,可以避免总线处于悬浮状态,消除总线抖动。

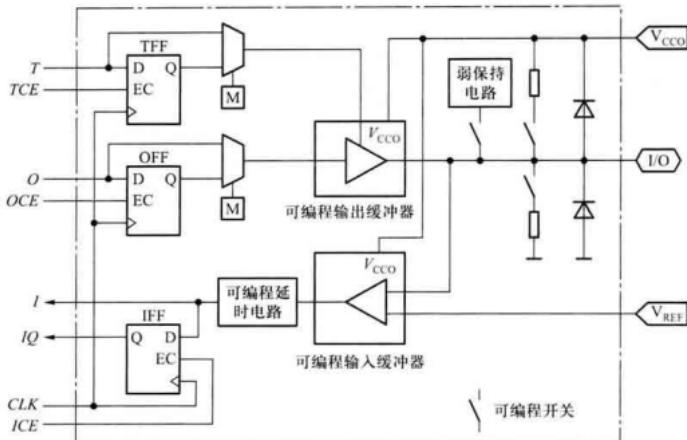


图 8.2.7 简化的 I/O 单元原理框图

I/O 单元中设计了两个电压输入端 V_{CCO} 和 V_{REF} ,以便兼容不同电源系统。为了增强 FPGA 的适应性和灵活性,通常将 I/O 单元进行分组,每组共用一个 V_{CCO} 和 V_{REF} ,不同组可以接不同的 V_{CCO} 和 V_{REF} 。这一措施可以使 FPGA 工作在由不同工作电源构成的复杂系统中,而 FPGA 内部逻辑电路则在其所谓的核心电源 (Core Power Supply) 下工作。 V_{REF} 为逻辑电平的参考电压,在执行某些 I/O 标准时,需要输入 V_{REF} 。

3. 可编程连线资源

可编程连线资源是 FPGA 不可或缺的组成部分,它包括纵向和横向连线以及可编程开关。图 8.2.4 中灰色方块表示的可编程开关(有时也称为可编程开关矩阵)有三种位置:处在和逻辑块相邻的灰色开关用来连接逻辑块和相邻逻辑块之间、逻辑块和连线之间的连接;而处在逻辑块间对角线位置上的开关则用于实现内部各连线之间的连接;与 I/O 块相邻的开关可以实现 I/O 块和逻辑块、I/O 块和内部连线之间的连接。

两种典型的互连开关结构如图 8.2.8 所示,当 NMOS 开关管栅极连接的 SRAM 单元 M 被编程为 1 时,开关管导通,否则开关管断开。图(a)纵横交叉互连开关通过编程,可以决定纵、横连线是否连通,而图(b)6 路互连开关通过编程,可以连通上、下、左、右任意方向,非常灵活。

在实际的 FPGA 芯片中,除了上述的称之为通用连线资源外,通常还有一些特殊用途的连线资源,如时钟信号线和全局控制信号线等。

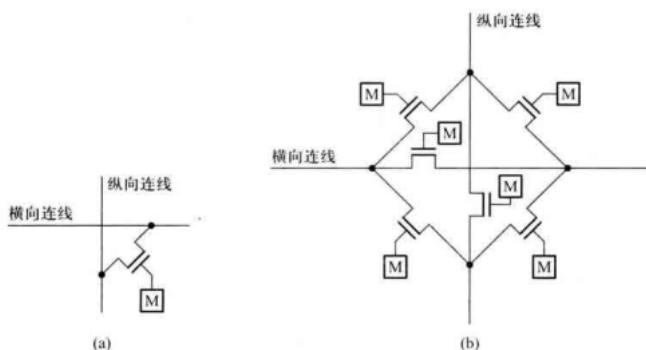


图 8.2.8 两种典型的可编程互连开关

(a) 纵横交叉互连开关 (b) 6 路互连开关

信号的传输延时是限制器件工作速度的根本原因。在 FPGA 的设计过程中,由软件进行优化,确定电路布局的位置和线路选择,以减小传输延迟时间,提高工作速度。

目前在很多高性能的 FPGA 产品中,已集成了乘法器、数字信号处理器、甚至 CPU 等非常复杂的模块,使 FPGA 在数字系统中几乎无所不能。

复习思考题

- 8.2.1 LUT 实现各种组合逻辑功能的原理是什么?
- 8.2.2 FPGA 在结构上主要由哪几个部分组成? 各部分的主要功能是什么?
- 8.2.3 FPGA 与 CPLD 有哪些主要区别? 它们各有什么特点?

8.3 可编程逻辑器件开发过程简介

通过上两节的介绍可以看出,CPLD 各种逻辑功能的实现,都是由其内部的可编程单元控制的。这些单元大多采用 E²PROM 或快闪存储器编程技术。而在 FPGA 中,逻辑块的功能、I/O 块的定义和连线资源的连接,是由他们相应的 SRAM 存储单元中的数据决定的。无论是 CPLD 还是 FPGA,编程过程就是将编程数据写入这些单元的过程。这一过程也称为下载(Download)或配置(Configuration)。编程数据有时也称为配置数据。

可编程器件的开发(编程实现)过程必须具备计算机、编程软件、通用或专用编程器(或编程电缆)三个条件。其一般开发步骤如图 8.3.1 所示。除了第一步和最后一步,其他步骤必须在

可编程开发软件的支持下完成。本书附录 A 介绍的 Quartus II 便是 Altera 公司提供的一种可编程逻辑器件开发软件。



图 8.3.1 可编程器件一般开发步骤

(1) 逻辑电路设计。设计者根据设计要求,运用前面章节介绍的逻辑电路设计方法设计出逻辑电路。

(2) 设计输入。将设计结果以开发软件能够接受的方式输入计算机。大多数开发软件均接受原理图描述方式和 HDL 描述方式。

(3) 逻辑综合。开发软件对设计输入进行逻辑化简和优化,并进行电路图连线错误或语法错误检查。

(4) 功能仿真。通过开发软件提供的仿真功能,可以对所设计的电路进行逻辑功能仿真。如果发现逻辑功能不满足设计要求,则需要修改设计。

(5) 适配。适配是将设计“装配”到选定的 CPLD 或 FPGA 芯片中。该过程需要选择芯片型号,分配器件引脚,然后由开发软件自动对逻辑电路进行划分、布局和布线(也可人工干预),并生成编程数据。

(6) 定时仿真。完成适配后,信号的传递通路也都确定了。定时仿真包含信号传递通路上的门、触发器、编程开关及连线等延时信息的时序仿真。若定时仿真发现不能满足设计要求时,则需更换器件或修改适配的约束条件重新进行适配;若仍不能满足要求,则需要修改最初的设计。

(7) 编程(下载)。将编程数据写入 CPLD 或 FPGA 器件中。由于 CPLD 和 FPGA 的编程实现技术存在差异,所以这两种器件的编程步骤也略有不同。

CPLD 的编程技术是基于 E²PROM 的,且绝大部分 CPLD 器件具有 ISP 功能。编程时,先通过专用编程电缆连接计算机的数据传输端口^①和被编程器件的 ISP 端口,然后通过编程软件发出编程命令,将编程数据写入芯片中,至此,CPLD 的开发过程结束,CPLD 具有了所设计的逻辑功能。

不同制造商生产的 CPLD 的 ISP 接口也不完全相同,但基本上都支持 JTAG^② 标准编程,所以 CPLD 中都设有 JTAG 规定的 4 个 I/O 引脚。各引脚作用如表 8.3.1 所示。图 8.3.2 是 Altera 公司的 MAX7000S 系列 CPLD 器件编程连接示意图。TDO 的数据可以用于编程校验,也可以作

① 可以是并口、串口或 USB 口,视编程软件所支持的端口而定。

② Joint Test Action Group 的缩写,指由该组织制定的 IEEE 1149.1 接口标准。

为多个 CPLD 串行编程时下一个 CPLD 器件的输入数据。多数 CPLD 的这 4 个编程引脚也可以作为用户 I/O 使用。

因为基于 SRAM 编程技术的 FPGA 断电后,其中的编程数据将消失,所以一般 FPGA 的编程步骤并不像 CPLD 那样,直接将编程数据写入 FPGA 中,而是写入与之配套的 PROM 中,以便长期保存。编程时,根据编程数据量,选择相应容量的 PROM 芯片或其他非易失性存储器,通过专用(或通用)编程器将编程数据写入 PROM 中。器件制造商一般都会提供与 FPGA 配套使用的 PROM、EPROM 等芯片。

当然,在电路实验、电路调试或单片机系统中,也可以由计算机直接控制 FPGA 相应的编程引脚,将存储在计算机中的编程数据直接装载到 FPGA 中。

表 8.3.1 JTAG 标准的编程引脚

引脚	功能
TMS	编程模式控制
TCK	编程时钟
TDI	编程数据输入
TDO	编程数据输出

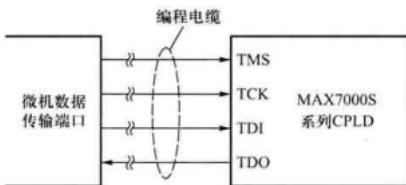


图 8.3.2 ISP 器件编程连接示意图

(8) 编程数据装载。只有基于 SRAM 编程技术的 FPGA 才需要该步骤。编程数据装载是指将存储在 PROM 中的编程数据装载到 FPGA 中的过程。由于第 7 步已经将编程数据写入 PROM 中,所以 FPGA 数据装载步骤不再需要由计算机、编程器及开发软件构成的开发环境。当设计好 PROM 和 FPGA 的连接电路后,电路每次通电时,编程数据的装载通常都可自动完成,无需人工干预。因此设计和实现 PROM 和 FPGA 的电路连接,是完成编程数据装载的前提。在完成定时仿真后就可以根据选定的 FPGA 和 PROM 器件的型号,设计它们的连接电路。

不同制造商的不同型号的 FPGA 产品与 PROM 的连接方式和数据装载控制方式也不同,通常都设有少量专用于数据装载的引脚,并提供几种可选的装载模式。下面以 Spartan-II 系列 FPGA 为例,简要说明装载控制方式和电路实现。

Spartan-II 系列的 FPGA 使用专用数据装载引脚,并结合一些通用 I/O 引脚完成数据装载。装载过程传递的信号包括模式控制、编程使能、数据输入、数据输出、状态指示、时钟等。

当 FPGA 的 3 个专用引脚 M2、M1 和 M0 接不同的逻辑电平时,便可选择一种装载模式进行数据装入。装载模式如表 8.3.2 所示。主模式利用 FPGA 内部振荡器产生装载时钟信号 CCLK 来驱动装有编程数据的 PROM。而从模式则需要外部电路提供时钟信号来驱动 CCLK 和装有编程数据的 PROM。当选择串行模式时,编程数据从 PROM 中以串行方式装入 FPGA 中,此时必须用有串行功能的 PROM。主串装入模式电路连接举例如图 8.3.3 所示。装载使能信号 PROGRAM=0 时开始装载过程。状态信号 DONE 变为高电平后表示装载完成。INIT 输入低电

平时初始化 FPGA。当选择并行模式时,除了时钟信号外,还需提供其他读写控制信号。

FPGA 不仅能够直接从 PROM 中读取编程数据,而且可以由其他微处理器或单片机控制装入编程数据。

表 8.3.2 Spartan-II 系列的 FPGA 装载模式控制

模式	M2 M1 M0	装载前相关引脚是否上拉	CCLK 方向	数据宽度	DOUT
主串	0 0 0	否	输出	1	串行输出
	0 0 1	是			
从串	1 1 0	是	输入	1	串行输出
	1 1 1	否			
从并	0 1 0	是	输入	8	无
	0 1 1	否			

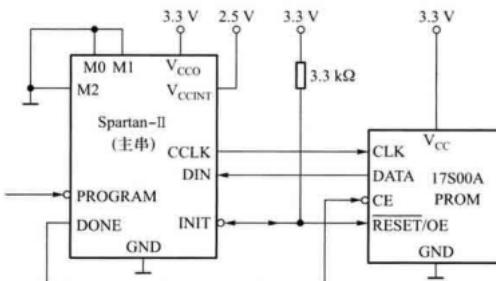


图 8.3.3 主串模式装载电路

近年来,随着半导体工艺的进一步提高,FPGA 的发展速度也相当快,种型号繁多,实际使用时,需要查阅所用产品型号的数据手册设计装载电路。

对简单 PLD 器件 PAL 和 GAL 的编程基本上与 CPLD 相同。

复习思考题

8.3.1 为什么在 FPGA 构成的数字系统中通常要配备一个 PROM 或 EPROM?

8.3.2 Spartan-II 系列的 FPGA 采用从模式装载时,是否需要外部时钟信号?

8.4 用 EDA 技术和可编程器件的设计例题

例 8.4.1 设计一个定时器。要求：

- (1) 定时时间为 24 s, 按递减方式计时, 每隔 1 s, 定时器减 1。
- (2) 设置两个外部控制开关(控制功能如表 8.4.1 所示), 控制定时器的直接复位、启动计时、暂停/连续计时。
- (3) 当定时器递减计时到零(即定时时间到)时, 定时器保持零不变, 同时发出报警信号。
- (4) 输入时钟脉冲的频率为 1 Hz, 输出为 8421BCD 码。

表 8.4.1 定时器功能表

复位/启动 nRST	暂停/连续 nPAUSE	定时器完成的功能
0	x	定时器复位, 置初值 24
1	1	定时器开始计时
1	0	定时器暂停计时

解：(1) 逻辑设计

用计数器对 1 Hz 的时钟信号进行计数, 其计数值即为定时时间。根据设计要求可知, 电路需要输出 2 组 8421BCD 码, 计数器初值为 24, 按递减方式计数, 减到 0 时, 输出报警信号, 并能控制计数器暂停/连续计数, 所以需要设计一个可预置初值的带使能控制端的递减计数器。

实现上述功能的 Verilog HDL 程序如下。HDL 描述分为 3 部分: 第一部分对电路的输入、输出信号进行声明。复位 nRST(低电平有效)、暂停/连续 nPAUSE 和时钟 CP 均为输入信号, TimerH(定时器的十位)、TimerL(定时器的个位)和报警 Alarm 均为输出信号。第二部分用一个连续赋值语句 **assign** 描述了电路报警信号的输出, 即计数器递减到 0 时, 输出报警信号(Alarm=1)。第三部分用一个过程块 **always** 描述计数器的计数、处理操作, 计数器的个位和十位均按 8421BCD 码方式递减计数。

```
/* 例 8.4.1 ***** Basketball24.v ***** */
module basketball24 ( //Verilog 2001,2005 module port syntax
    input nRST, nPAUSE, CP,           //定时器的输入信号声明
    output reg [3:0] TimerH, TimerL, //定时器的输出信号声明
    output Alarm            //定时时间到,输出报警信号 Alarm=1
);
    assign Alarm = (|TimerH, TimerL| == 8'h00) & (nRST == 1'b1); //输出报警信号
    always @ (posedge CP, negedge nRST, negedge nPAUSE) //计数处理
```

```

begin
    if (~nRST)
        | TimerH, TimerL| <= 8'h24; //复位时置初值 24
    else if (~nPAUSE)
        | TimerH, TimerL| <= |TimerH, TimerL|; //暂停计时
    else if (|TimerH, TimerL| == 8'h00) //定时时间到,保持 0 不变
        begin |TimerH, TimerL| <= |TimerH, TimerL|; end
    else if (TimerL == 4'h0)
        begin TimerH <= TimerH - 1'bl; TimerL <= 4'h9; end
    else
        begin TimerH <= TimerH; TimerL <= TimerL - 1'bl; end
    end
endmodule

```

(2) 设计实现

用可编程逻辑器件实现上述设计的过程如下：

① 在 Quartus II 9.1 软件中建立一个新的工程项目，输入上述 HDL 文件，对设计项目进行编译。

② 新建一个仿真波形文件，给出输入信号的激励波形（图中为方便仿真将时钟 CP 的周期指定为 2 ms），对设计项目进行时序仿真，得到如图 8.4.1 所示的波形。

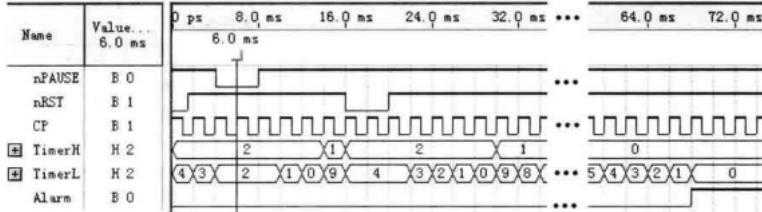


图 8.4.1 定时器的仿真波形图

分析波形图可知，开始和 16~20 ms 时，nRST 为低电平，初值 24 被预置到计数器的输出端 TimerH、TimerL；4~8 ms 时，nPAUSE 为低电平，计数器暂停计数；其他时间 nRST、nPAUSE 均为高电平，计数器在 CP 上升沿到来时递减计数，到 68 ms 之后，计数器一直输出为 0，报警信号 Alarm 为高电平。仿真结果完全符合设计要求。

③ 选定目标器件（如 EPM7032S），将输入、输出信号分配到器件相应的引脚上，然后重新编译设计项目，生成下载文件（文件后缀为.pof 或.sof）。

④ 将下载文件写入到目标器件中,就是一块专用的定时器电路。

小结

- 可编程逻辑器件(PLD)可以由用户自行设计逻辑功能。它们具有集成度高、可靠性高、处理速度快和保密性好等特点。CPLD 是在 GAL 的基础上发展起来的复杂可编程逻辑器件,其电路结构的核心是与-或阵列和触发器,且可以在系统编程(ISP 特性)。

- FPGA 是基于 LUT 实现逻辑函数的可编程器件,且大部分 FPGA 的 LUT 由数据选择器和 SRAM 构成。它以功能很强的逻辑块为基本逻辑单元,可以实现各种复杂的逻辑功能。FPGA 是目前规模最大、密度最高的可编程器件。

- 无论是 CPLD 还是 FPGA,其开发过程必须借助相关开发软件和编程设备才能完成。

习题

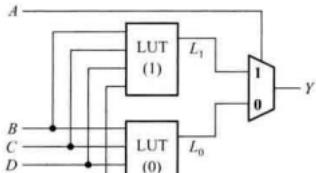
8.1 CPLD

8.1.1 若某 CPLD 中的逻辑块有 36 个输入(不含全局时钟、全局使能控制等),16 个宏单元。理论上,该逻辑块可以实现多少个逻辑函数?每个逻辑函数最多可有多少个变量?如果每个宏单元包含 5 个乘积项,通过乘积项扩展,逻辑函数中所能包含的乘积项数目最多是多少?

8.2 FPGA

8.2.1 电路如图题 8.2.1 所示,LUT 的内容如表题 8.2.1 所示。试写出 Y 的逻辑函数表达式。

表题 8.2.1



图题 8.2.1

B	C	D	E	L_1	L_0	B	C	D	E	L_1	L_0
0	0	0	0	0	1	1	0	0	0	0	1
0	0	0	1	0	0	1	0	0	1	1	0
0	0	1	0	1	0	1	0	1	0	0	0
0	0	1	1	0	0	1	0	1	1	0	1
0	1	0	0	1	0	1	1	0	0	1	0
0	1	0	1	0	0	1	1	0	1	1	0
0	1	1	0	0	1	1	1	1	0	1	0
0	1	1	1	0	1	1	1	1	1	1	1

8.2.2 用 5 个 2 输入 LUT 实现逻辑函数: $f(x_1, x_2, x_3) = x_1\bar{x}_2 + x_1x_3 + x_2\bar{x}_3$ 。用图 8.2.3 的风格标出每个 LUT 的真值表,不必画 FPGA 中的无关连线和编程节点。

8.2.3 用 2 个 2 输入 LUT 实现逻辑函数: $f(x_1, x_2, x_3) = \sum m(2, 3, 4, 6, 7)$ 。用图 8.2.3 的风格标出每个 LUT 的真值表,不必画 FPGA 中的无关连线和编程节点。

8.4 用 EDA 技术和可编程器件的设计例题

8.4.1 试用 Verilog 语言设计一个 1 位十进制数(用 8421BCD 码表示)加法器,用 Quartus II 软件对电路进

行逻辑功能仿真，并给出仿真波形。

8.4.2 设计一个4位数字显示的简易频率计。要求：

- (1) 能够测试100~9999 Hz 正方波信号(幅度为3~5 V)的频率。
- (2) 电路输入的基准时钟为1 Hz, 要求测量值以8421BCD码形式输出。
- (3) 系统有复位按键。
- (4) 采用分层次、分模块的方法,用Verilog语言进行设计。
- (5) 最后用Quartus II软件对电路进行逻辑功能仿真,并给出仿真波形。

9 >>>

脉冲波形的变换与产生

引言

在数字电路中，常常需要各种脉冲波形，例如时序电路中的时钟脉冲、控制过程中的定时信号等。这些脉冲波形的获取，通常有两种方法：一种是用脉冲信号产生电路直接产生；另一种则是将已有信号通过波形变换电路获得。

本章首先介绍用门电路组成的单稳态触发器、施密特触发器、多谐振荡器及其集成电路，然后讨论 555 定时器和用它构成的波形变换、产生电路。

9.1 单稳态触发器

前面介绍的触发器，有 0、1 两个稳定状态，因此，这种电路也被称为双稳态电路。一个双稳态电路可以保存一位二值信息。本章介绍的单稳态触发器与双稳态电路不同，它只有一个稳定状态，并具有如下的工作特点：

- (1) 没有触发脉冲作用时，电路处于一种稳定状态。
- (2) 在触发脉冲作用下，电路会由稳态翻转到暂稳态。暂稳态是一种不能长久保持的状态。
- (3) 由于电路中 RC 延时环节的作用，电路的暂稳态在维持一段时间后，会自动返回到稳态。暂稳态持续时间由 RC 延时环节参数值决定。

单稳态触发器被广泛地应用于脉冲的变换、延时和定时等。

9.1.1 用门电路组成的单稳态触发器

1. 电路组成及工作原理

单稳态触发器可由逻辑门和 RC 电路组成。根据 RC 电路连接方式不同，单稳态触发器分为微分型单稳态触发器和积分型单稳态触发器两种。本章只讨论微分型单稳态触发器。

用不同的 CMOS 门组成的微分型单稳态触发器如图 9.1.1(a)、(b) 所示。两电路中的 RC 电路，均按微分电路的方式连接在 G_1 门的输出端和 G_2 门的输入端。图 9.1.1 中两电路表明，采

用的逻辑门不同，单稳态触发器的触发信号和输出脉冲也不一样。

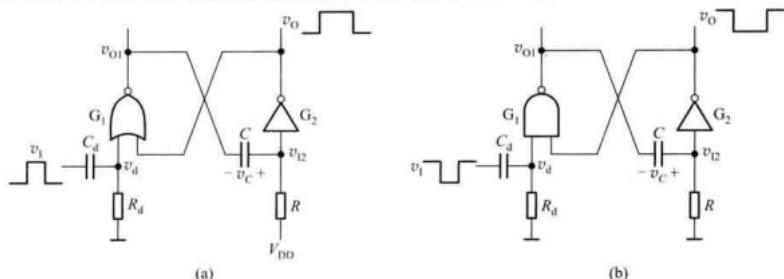


图 9.1.1 CMOS 门电路组成的微分型单稳态触发器

(a) 用或非门和非门构成的微分型单稳态触发器 (b) 用与非门和非门构成的微分型单稳态触发器

下面以如图 9.1.1(a) 所示的微分型单稳态触发器为例，说明单稳态触发器的工作原理。

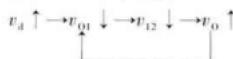
为了便于讨论，本章中将 CMOS 门电路的电压传输特性理想化，且设定 CMOS 门的阈值电压 $V_{TH} \approx \frac{V_{DD}}{2}$ ，输出高电平 $V_{OH} \approx V_{DD}$ ，输出低电平 $V_{OL} \approx 0$ V。

(1) 没有触发信号时，电路处于一种稳定状态

无触发信号作用时， v_1 为低电平。由于 G_2 门的输入端经电阻 R 接 V_{DD} ，故 $v_0 = V_{OL}$ ；这样， G_1 门两输入端均为 0， $v_{01} = V_{OH}$ ，电容器 C 两端的电压接近 0 V，电路处于一种稳定状态。只要没有正脉冲触发，电路就一直保持这一稳态不变。

(2) 外加触发信号，电路由稳态翻转至暂稳态

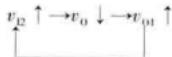
输入触发脉冲，在 R_d 和 C_d 组成的微分电路的输出端得到很窄的正、负脉冲 v_d 。当 v_d 上升到 G_1 门的阈值电压 V_{TH} 时，在电路中产生如下正反馈过程：



这一正反馈过程使 v_{01} 迅速地从高电平跳变为低电平。由于电容 C 两端的电压不能突变， v_{12} 也同时跳变为低电平，并使 v_0 跳变为高电平，电路进入暂稳态。此时，即使 v_d 返回到低电平， v_0 仍将维持高电平。由于电容 C 的存在，电路的这种状态是不能长久保持的，所以将电路此时的状态称之为暂稳态。暂稳态时 $v_{01} \approx 0$, $v_0 \approx V_{DD}$ 。

(3) 暂稳态期间电容器 C 充电，电路自动从暂稳态返回至稳态。

进入暂稳态后，电源 V_{DD} 经电阻 R 和 G_1 门导通的工作管对电容 C 充电， v_{12} 按指数规律升高，当 v_{12} 达到 G_2 门的阈值电压 V_{TH} 时，电路又产生下述正反馈过程：



假如此时触发脉冲已消失,即 v_4 返回到低电平,上述正反馈过程使 v_{01}, v_{12} 迅速跳变到高电平,输出返回到 $v_o \approx 0$ V 的状态。此后,电容通过电阻 R 和 G_2 门的输入保护电路放电,使电容 C 上的电压最终恢复到稳定状态时的初始值,电路从暂稳态自动返回到稳态。

根据上述电路工作过程分析,可画出电路工作时各点电压波形如图 9.1.2 所示。

2. 主要参数的计算

(1) 输出脉冲宽度 t_w

由图 9.1.2 可见,触发信号作用后,输出脉冲宽度 t_w 就是暂稳态维持时间,也就是 RC 电路在充电过程中,使 v_{12} 从 0 V 上升到 V_{TH} 所需时间。

根据 RC 电路过渡过程的分析,可求输出脉冲宽度

$$t_w = RC \ln \frac{v_c(\infty) - v_c(0)}{v_c(\infty) - V_{TH}} \quad (9.1.1)$$

式中 $v_c(0)$ 是电容的起始电压值, $v_c(\infty)$ 是电容的充电的终了电压值。

电容在充电过程中, $v_c(0) = 0$; $v_c(\infty) = V_{DD}$, $\tau = RC$, $V_{TH} = V_{DD}/2$ 。将这些值代入式(9.1.1)

$$\text{得 } t_w = RC \ln \frac{V_{DD} - 0}{V_{DD} - V_{TH}} = RC \ln 2 = 0.69RC \quad (9.1.2)$$

$$\text{可取近似值 } t_w \approx 0.7RC \quad (9.1.3)$$

(2) 恢复时间 t_{re}

暂稳态结束后,还要经过一段恢复时间,让电容 C 上的电荷释放完,才能使电路完全恢复到触发前的起始状态。恢复时间一般认为要经过放电时间常数的 3~5 倍, RC 电路才基本达到稳态。

(3) 最高工作频率 f_{max}

设触发信号 v_1 的周期为 T ,为了使单稳态电路能正常工作,应满足 $T > (t_w + t_{re})$ 的条件,因此,单稳态触发器的最高工作频率为

$$f_{max} = \frac{1}{T_{min}} < \frac{1}{t_w + t_{re}} \quad (9.1.4)$$

9.1.2 集成单稳态触发器

用逻辑门组成的单稳态触发器虽然电路结构简单,但它存在触发方式单一、输出脉宽稳定性差,调节范围小等缺点。为提高单稳态触发器的性能指标,现已制成单片的集成电路。如 74LS122 和 74121 等 TTL 产品和 74HC123、MC14098、MC14528 等 CMOS 产品。集成单稳态触发器,根据电路工作特性不同分为可重复触发和不可重复触发两种,其工作波形分别如图 9.1.3

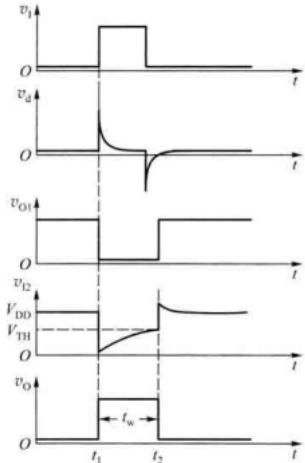


图 9.1.2 微分型单稳态触发器中各点的电压波形图

(a)、(b) 所示。

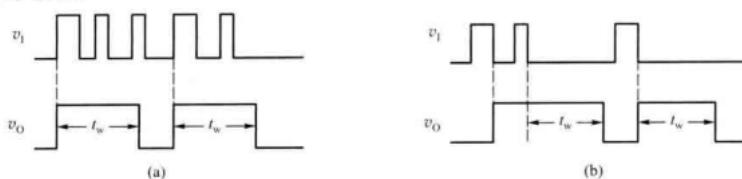


图 9.1.3 两种集成单稳态触发器的工作波形

(a) 前沿触发的不可重复触发单稳 (b) 后沿的触发可重复触发单稳

不可重复触发单稳态触发器，在暂稳态期间，如有触发脉冲加入，电路的输出脉宽不受其影响，仍由电路中的 R 、 C 参数值确定。而可重复触发单稳态电路在暂稳态期间，如有触发脉冲加入，电路会被输入脉冲重复触发，暂稳态将延长，暂稳态在最后一个脉冲的触发沿，再延时 t_s 时间后回到稳态。其输出脉宽根据触发脉冲的输入情况的不同而改变。

1. 不可重复触发的集成单稳态触发器

TTL 集成器件 74121 是一种不可重复触发的集成单稳态触发器，其内部逻辑图如图 9.1.4 所示。电路由触发信号控制电路、微分型单稳态触发器和输出缓冲器三部分组成。

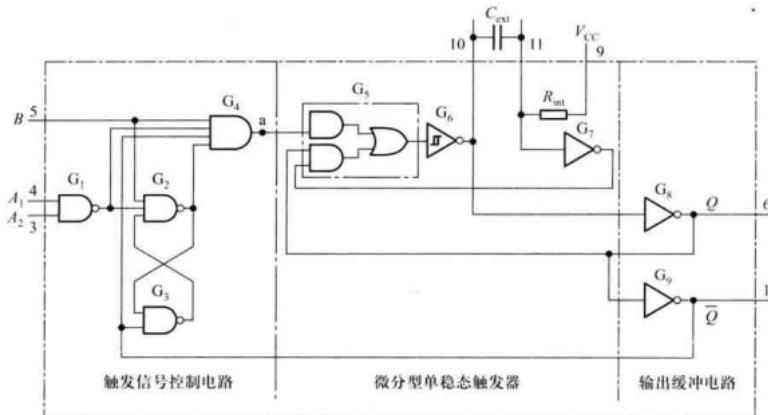


图 9.1.4 单稳态触发器 74121 逻辑图

单稳态触发器 74121 在使用时，要在芯片的 10、11 引脚之间外接电容 C_{ext} （如采用电解电容，电容 C 的正极端接 10 脚）。根据输出脉宽的要求，定时电阻 R 可选择外接电阻 R_{ext} 或芯片内部电阻 R_{int} （ $2\text{ k}\Omega$ ）。单稳态触发器采用内、外部电阻的电路连接如图 9.1.5(a)、(b) 所示。

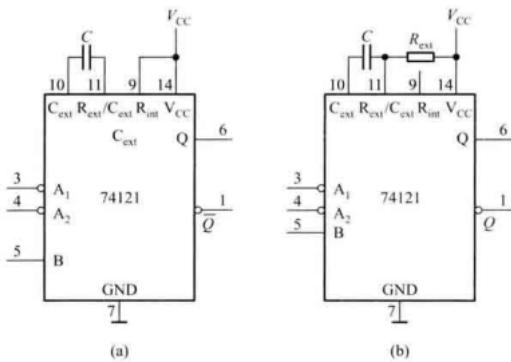


图 9.1.5 74121 定时电容、电阻的连接

(a) 使用内接电阻 R_{int} 的电路连接 (b) 使用外部电阻 R_{ext} 的电路连接

(1) 工作原理

如将图 9.1.4 中, 具有施密特特性的非门 G_6 与 G_5 门合起来, 看成是一个或非门, 它与 G_7 非门、电阻 R_{int} 及电容 C_{ext} 组成微分型单稳态触发器, 其工作原理与 9.1.1 节所介绍的微分型单稳态触发器基本相同。电路只有一个稳态 $Q=0, \bar{Q}=1$ 。当 G_4 门输出端 a 点有正脉冲触发时, 电路进入暂态 $Q=1, \bar{Q}=0$ 。电路输出脉冲的宽度由 R_{int} 和 C_{ext} 大小决定。

G_1, G_2, G_3 和 G_4 组成的触发信号控制电路, 不仅实现了输入信号触发沿可选择, 而且还使电路具有不可重复触发特性。如当 A_1, A_2 当中至少有一个接低电平, 且触发信号从 B 端输入, 电路选择上升沿触发。此时, 由于 G_4 门的其他三个输入端均为高电平, 当 B 端输入信号的上升沿到来时, a 点也随之跳变为高电平, 在正脉冲触发下, 单稳态电路进入暂态, $Q=1, \bar{Q}=0$ 。 \bar{Q} 的低电平使触发信号控制电路中 SR 锁存器的 G_2 门输出为低电平, 于是 G_4 门被封锁, 此时即使有触发信号输入, 也不会有触发信号到达 a 点。只有电路在返回稳态后, 触发信号才能使电路再次被触发。由以上分析可知, 电路具有边沿触发的性质, 且属于不可重复触发的单稳态触发器。

与 9.1.1 节所述电路相同, 电路的输出脉冲宽度为

$$t_w \approx 0.7RC \quad (9.1.5)$$

如需要采用下降沿触发, 则可将 B 输入端接高电平, 触发脉冲从 A_1 或 A_2 输入。电路的工作状态与电路选择上升沿触发时完全相同。

(2) 逻辑功能

74121 的功能如表 9.1.1 所示。

表 9.1.1 74121 功能表

输入			输出	
A_1	A_2	B	Q	\bar{Q}
L	x	H	L	H
x	L	H	L	H
x	x	L	L	H
H	H	x	L	H
H	↓	H	↑	↑
↓	H	H	↑	↑
↓	↓	H	↑	↑
L	x	↑	↑	↑
x	L	↑	↑	↑

由功能表可见，在下述情况下，电路有正脉冲输出：

- ① 若 A_1, A_2 两个输入中有一个或两个为低电平， B 产生由 0 到 1 的正跳变时；
 - ② 若 B 为高电平， A_1, A_2 中有一个或两个产生由 1 到 0 的负跳变时。
- (3) 工作波形

根据 74121 的功能表，可画出图 9.1.5(a) 的工作波形如图 9.1.6 所示。

2. 可重复触发集成单稳态触发器

下面我们以常用 CMOS 集成器件 MC14528 为例，简述可重复触发单稳态触发器的工作原理。该器件的逻辑图如图 9.1.7 所示，图中 R_{ex} 和 C_{ex} 为外接电阻和电容。

MC14528 功能表如表 9.1.2 所示，工作波形如图 9.1.8 所示。

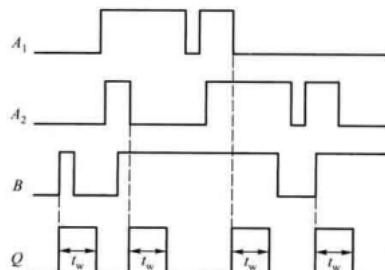


图 9.1.6 74121 集成单稳态触发器的工作波形

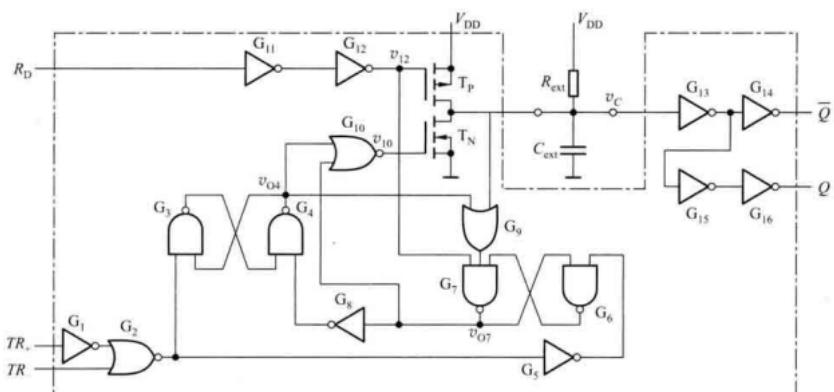


图 9.1.7 集成单稳态触发器 MC14528 逻辑图

表 9.1.2 MC14525 功能表

输入			输出		功能
\bar{R}_D	TR_+	TR_-	Q	\bar{Q}	
L	x	x	L	H	清除
x	H	x	L	H	禁止
x	x	L	L	H	禁止
H	H	↑	↑↓	↑↓	单稳
H	↓	L	↑↓	↑↓	单稳

下面以图 9.1.8 所示工作情况为例,说明电路的工作原理。

在 $t < t_1$ 期间,没有触发信号输入 ($TR_+ = 1, TR_- = 0$), 电路工作在稳定状态。 G_4 门的输出 v_{04} 一定是高电平。这是因为,在接通电源瞬间,电容 C_{ext} 上的电压是低电平,如果 v_{04} 不是高电平,而是低电平,那么, G_9 门的输出也为低电平,这样, G_7 门的输出 v_{07} 为高电平、 G_8 门输出为低电平。所以, v_{04} 只能是高电平。 v_{04} 的高电平使 G_6 、 G_7 门组成的基本 SR 锁存器的输出端 v_{07} 为低电平, G_8 门的输出为高电平。 v_{04} 的高电平状态将保持不变。

由于 G_{10} 门的输出为低电平, G_{12} 的输出高电平,使 T_N 、 T_P 同时截止,电源 V_{DD} 经 R_{ext} 对 C_{ext} 充电,达到稳定态时 $v_C = V_{DD}$, $Q = 0, \bar{Q} = 1$ 。

在 t_1 时刻 TR_- 端输入的正触发脉冲,使 G_3 、 G_4 门组成的基本 SR 锁存器的输出 v_{04} 转换为低电平, G_{10} 门输出高电平, T_N 导通, C_{ext} 放电。当 v_C 下降到 G_{13} 门的阈值电压 V_{TH3} 时, $Q = 1, \bar{Q} = 0$ 电

路进入暂态。此后, v_c 进一步下降, 当 v_c 降至 G_9 门的阈值电压 V_{TH3} 时 ($V_{TH3} > V_{TH9}$), G_9 门输出为低电平, 此电平经 G_7 、 G_8 使 v_{04} 为高电平。于是, T_s 截止, 电容 C_{ext} 又重新开始充电, v_c 上升至 V_{TH3} 时, 电路自动返回到 $Q=0, \bar{Q}=1$ 的稳态。由图 9.1.8 可知, 输出脉宽 t_w 等于 v_c 由 V_{TH3} 放电降至 V_{TH9} 的时间与 v_c 由 V_{TH9} 充电升至 V_{TH3} 两个时间之和。为获得较宽的输出脉冲, 一般都将 V_{TH3} 设计得较高而将 V_{TH9} 设计得较低。

在 t_1 时刻, 电路再一次被触发, 与上述分析相同, 电路进入暂态, C_{ext} 放电, 当 v_c 降至 G_9 门的阈值电压 V_{TH9} 时, 电容 C_{ext} 又重新开始充电, 当 C_{ext} 还没有充电到 V_{TH3} , 电路在 t_4 时刻被再次触发, 使 v_{04} 转换为低电平, G_{10} 门输出高电平, T_N 导通, C_{ext} 又重新放电, 当 v_c 降至 G_9 门的阈值电压 V_{TH9} , G_9 门的输出转换为低电平, v_{04} 为高电平。于是, T_s 截止, 电容 C_{ext} 又重新开始充电, v_c 上升至 V_{TH3} 时, 电路才自动返回到 $Q=0, \bar{Q}=1$ 的稳态。所以, 电路为可重复触发集成单稳态触发器。

9.1.3 单稳态触发器的应用

1. 定时

分析图 9.1.9 定时电路可知, 电路只有在单稳态触发器的输出高电平期间 (t_w 时间内), v_A 信

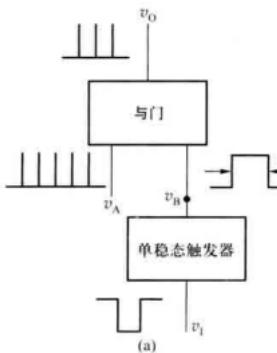
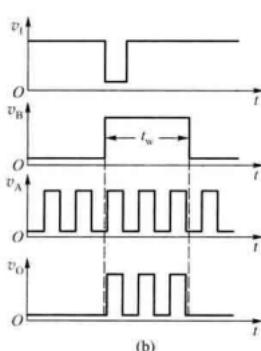


图 9.1.9 单稳态触发器作定时电路的应用

(a) 逻辑框图



(b)

(b) 波形图

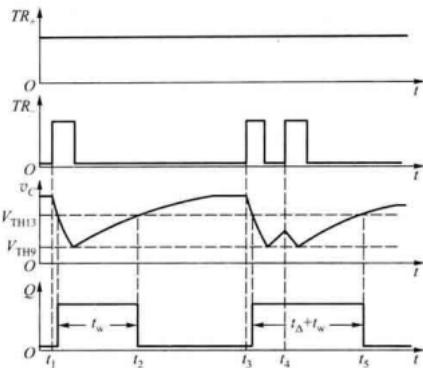


图 9.1.8 MC14528 可重复触发单稳态电路工作波形

号才有可能通过与门。单稳态触发器的 RC 的取值不同,与门的开启时间则不同,通过与门的脉冲个数也就随之改变。

2. 延时

单稳态触发器的另一用途是实现脉冲的延时。用两片 74121 组成的脉冲延时电路和工作波形分别如图 9.1.10(a)、(b) 所示。从波形图可以看出, v_0 脉冲的上升沿,相对输入信号 v_i 的上升沿延迟了 t_{w1} 时间。

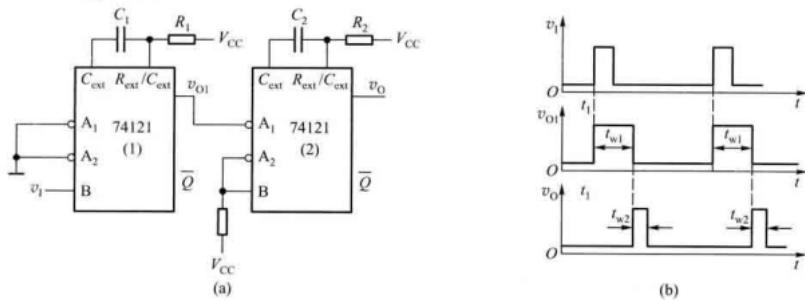


图 9.1.10 用 74121 组成的延时电路及工作波形

(a) 延时电路 (b) 工作波形

3. 噪声消除电路

由单稳态触发器组成的噪声消除电路及工作波形分别如图 9.1.11(a)、(b) 所示,有用的信号一般都有一定的脉冲宽度,而噪声多表现为尖脉冲。从分析结果可见,只要合理地选择 R 、 C 的值,使单稳态电路的输出脉宽大于噪声宽度而小于信号的脉宽,即可消除噪声。

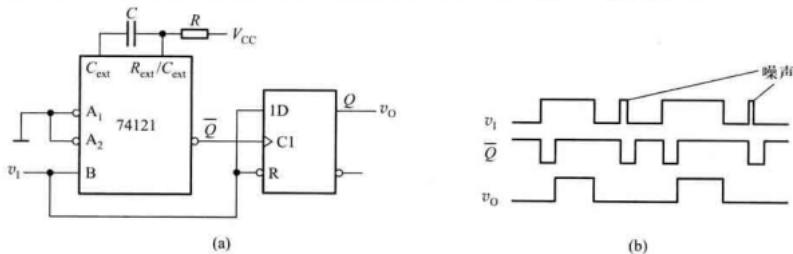


图 9.1.11 噪声消除电路

(a) 逻辑图 (b) 波形图

复习思考题

- 9.1.1 与双稳态触发器相比较,单稳态触发器在电路的基本组成和工作原理方面有什么不同?
- 9.1.2 用与非门和或非门组成的微分型单稳态触发器,它们的触发脉冲和输出脉冲有何区别?
- 9.1.3 图 9.1.1 所示的单稳态触发器的输出脉宽与电路中哪些参数有关?
- 9.1.4 集成单稳态触发器分为哪两类?它们的区别是什么?
- 9.1.5 用集成单稳态触发器 74121 产生输出脉宽为 3 ms 的脉冲信号,如选 $R=2\text{ k}\Omega$ (内部电阻),试问外接电容 C_{ext} 应取何值?

9.2 施密特触发器

施密特触发器常用于波形变换、幅度鉴别等。施密特触发器具有以下工作特点:

(1) 电路属于电平触发,对于缓慢变化的信号仍然适用。当输入信号达到某一电压值时,输出电压会发生跳变。但输入信号在增加过程和减小过程中,使输出状态跳变时,所对应的输入电平不相同。

(2) 由于电路内部的正反馈的作用,当电路输出状态变换时,输出电压波形的边沿很陡直。

根据输入、输出相位关系的不同,施密特触发器分为同相输出和反相输出两种类型。它们的电压传输特性曲线及逻辑符号分别如图 9.2.1(a)、(b) 所示。由于电路的电压传输特性曲线类似于铁磁材料的磁滞回线,磁滞曲线就成为了施密特触发器的符号标志。

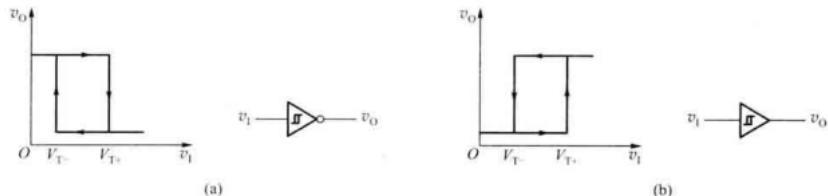


图 9.2.1 施密特电路的传输特性

(a) 反相输出施密特电路的传输特性 (b) 同相输出施密特电路的传输特性及逻辑符号

9.2.1 用门电路组成的施密特触发器

1. 电路组成

用 CMOS 门电路组成的施密特触发器如图 9.2.2 所示。电路中两个 CMOS 反相器串接,分压电阻 R_1 、 R_2 将输出端的电压反馈到 G_1 门的输入端,并对电路产生影响。

2. 工作原理

设 CMOS 反相器的阈值电压 $V_{TH} = \frac{V_{DD}}{2}$, 电路中 $R_1 < R_2$ 。

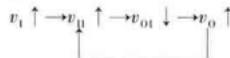
从图 9.2.2 可见, G_1 门的输入电平 v_{II} 决定着电路的输出状态。根据叠加原理有

$$v_{II} = \frac{R_2}{R_1 + R_2} \cdot v_I + \frac{R_1}{R_1 + R_2} \cdot v_{OL} \quad (9.2.1)$$

设输入信号 v_I 为三角波。

当 $v_I = 0$ V 时, $v_{II} \approx 0$ V, G_1 门截止, G_2 门导通, $v_{O1} = V_{OH} \approx V_{DD}$, $v_O = V_{OL} \approx 0$ V。

v_I 从 0 V 电压逐渐增加, 只要 $v_{II} < V_{TH}$, 电路保持 $v_O \approx 0$ V 不变。当 v_I 上升到 $v_{II} = V_{TH}$ 时, G_1 门进入其电压传输特性转折区, 随着 v_{II} 的增加在电路中产生如下正反馈过程:



这样, 电路的输出状态很快从低电平跳变为高电平, $v_O \approx V_{DD}$ 。

我们把在输入信号在上升过程中, 使电路的输出电平发生跳变时所对应的输入电压称为正向阈值电压, 用 V_{T+} 表示。

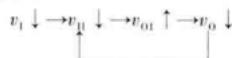
即由式(9.2.1)得

$$v_{II} = V_{TH} = \frac{R_2}{R_1 + R_2} V_{T+} \quad (9.2.2)$$

$$V_{T+} = \left(1 + \frac{R_1}{R_2} \right) V_{TH} \quad (9.2.3)$$

如果 v_{II} 继续上升, 电路在 $v_{II} > V_{TH}$ 后, 输出状态维持 $v_O \approx V_{DD}$ 不变。

如果 v_I 从高电平开始逐渐下降, 当降至 $v_{II} = V_{TH}$ 时, G_1 门又进入其电压传输特性转折区, 随着 v_I 的下降, 电路产生如下的正反馈过程:



电路迅速从高电平跳变为低电平, $v_O \approx 0$ V。

我们将输入信号在下降过程中, 使输出电平发生跳变时所对应的输入电平称为负向阈值电压, 用 V_{T-} 表示。根据式(9.2.1)有

$$v_{II} \approx V_{TH} = \frac{R_2}{R_1 + R_2} V_{T-} + \frac{R_1}{R_1 + R_2} V_{DD}$$

将 $V_{DD} = 2V_{TH}$ 代入可得

$$V_{T-} \approx \left(1 - \frac{R_1}{R_2} \right) V_{TH} \quad (9.2.4)$$

定义正向阈值电压 V_{T+} 与负向阈值电压 V_{T-} 之差为回差电压, 记作 ΔV_T 。由式(9.2.3)和式(9.2.4)可求得

$$\Delta V_T = V_{T+} - V_{T-} \approx 2 \frac{R_1}{R_2} V_{TH} = \frac{R_1}{R_2} V_{DD} \quad (9.2.5)$$

式(9.2.5)表明, 电路的回差电压与 R_1/R_2 成正比, 改变 R_1, R_2 的比值即可调节回差电压的大小。

3. 工作波形及电压传输特性

根据以上分析, 可画出电路的工作波形如图 9.2.3(a) 所示。以 v_o 作为电路的输出端和以 v_{0i} 作为输出端时, 电路的电压传输特性分别如图 9.2.3(b)、(c) 所示。图 9.2.3(b)、(c) 表明, 如以 v_o 作为电路的输出端时, 电路为同相输出施密特触发器; 如以 v_{0i} 作为输出端时, 电路则为反相输出施密特触发器,

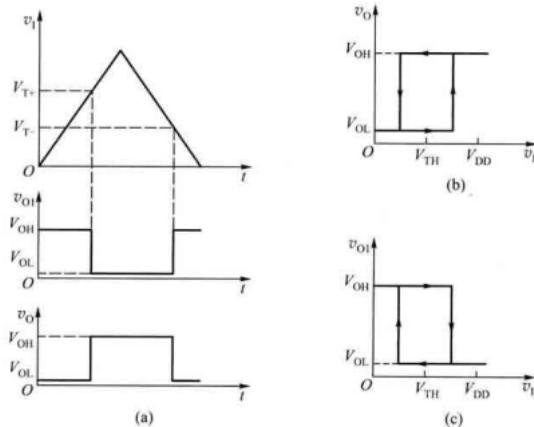


图 9.2.3 施密特触发器工作波形及传输特性

(a) 电路的工作波形 (b) v_o 输出的传输特性 (c) v_{0i} 输出的传输特性

例 9.2.1 在图 9.2.2 所示的电路中, 电源电压 $V_{DD} = 10 \text{ V}$, G_1, G_2 选用 CC4069 反相器, 其负载电流最大允许值 $I_{OH(\max)} = 1.3 \text{ mA}$, 门的阈值电压 $V_{TH} \approx \frac{1}{2} V_{DD} = 5 \text{ V}$, 且 $R_1/R_2 = 0.5$ 。

(1) 求电路 V_{T+} 、 V_{T-} 和 ΔV_T 的值;

(2) 试计算 R_1, R_2 值。

解: (1) 求 V_{T+} 、 V_{T-} 和 ΔV_T

由式(9.2.3)、式(9.2.4)和式(9.2.5)可求出

$$V_{T+} = \left(1 + \frac{R_1}{R_2}\right) V_{TH} = 1.5 \times 5 \text{ V} = 7.5 \text{ V}$$

$$V_{T-} = \left(1 - \frac{R_1}{R_2}\right) V_{TH} = (1 - 0.5) \times 5 \text{ V} = 2.5 \text{ V}$$

$$\Delta V_T = V_{T+} - V_{T-} = 5 \text{ V}$$

(2) 计算 R_1 、 R_2 值

为保证反相器 G_2 输出高电平时的负载电流不超过最大允许值 $I_{OH(max)}$, 应使

$$\frac{V_{OH} - V_{TH}}{R_2} < I_{OH(max)}$$

考虑到 $V_{OH} \approx V_{DD} = 10 \text{ V}$, 故由上式可得

$$R_2 > \frac{10 \text{ V} - 5 \text{ V}}{1.3 \text{ mA}} = 3.85 \text{ k}\Omega$$

当选 $R_2 = 15 \text{ k}\Omega$ 时, 则 $R_1 = \frac{1}{2} R_2 = 7.5 \text{ k}\Omega$ 。

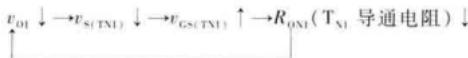
9.2.2 集成施密特触发器

目前, 无论是 CMOS 还是 TTL 电路, 都有单片的集成施密特触发器产品, 它们性能稳定, 应用十分广泛。下面以 CMOS 集成施密特触发器 CC40106 为例介绍其工作原理。图 9.2.4(a)、(b) 分别为 CC40106 的电路图和逻辑符号。集成施密特触发器 CC40106 的内部电路由施密特电路、整形电路和输出电路三部分组成, 核心部分是施密特电路。图中 T_{p1} 、 T_{n4} 和 T_{p5} 、 T_{n5} 组成两个首尾相连的反相器构成整形级, 在 v_{o1} 上升和下降过程中, 利用两级反相器的正反馈作用, 可使输出波形的上升沿和下降沿变得很陡直。输出级为 T_{p6} 和 T_{n6} 组成的反相器, 它不仅能起到与负载隔离的作用, 而且可提高电路的带负载能力。

下面我们着重讨论图 9.2.4 中施密特电路的工作原理。施密特电路由 P 沟道 MOS 管 T_{p1} ~ T_{p3} 、N 沟道 MOS 管 T_{n1} ~ T_{n3} 组成, 设 P 沟道 MOS 管的开启电压为 V_{TP} , N 沟道 MOS 管开启电压为 V_{TN} 。

电路的输入信号 v_i 为三角波。当 $v_i = 0$ 时, T_{p1} 、 T_{p2} 导通, T_{n1} 、 T_{n2} 截止, 电路中 v_{o1} 为高电平 (V_{DD}), v_{o1} 的高电平使 T_{p3} 截止, T_{n3} 导通, 电路为源极跟随器, 此时, T_{n1} 源极的电位 $v_{S(TN1)}$ 较高, 此时 $v_o = V_{OH}$ 。

v_i 电位逐渐升高, 当 $v_i > V_{TN}$ 时, T_{n2} 导通, 由于 T_{n1} 的源极电压较大, 即使 $v_i > V_{DD}/2$, T_{n1} 仍不能导通。随着 v_i 继续升高, T_{p1} 、 T_{p2} 的栅源电压的绝对值减小, 致使 T_{p1} 、 T_{p2} 趋于截止。随着 T_{p1} 、 T_{p2} 的截止, 其内阻急剧增大, 从而使得 v_{o1} 和 $v_{S(TN1)}$ 开始下降。当 $v_i - v_{S(TN1)} \geq V_{TN}$ 时, T_{n1} 开始导通, 并引起如下正反馈过程:



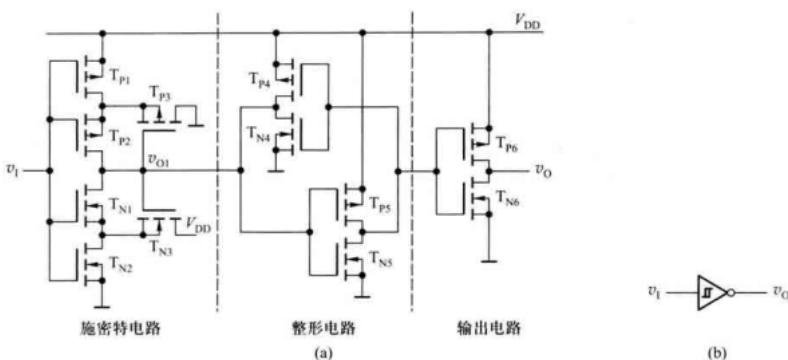


图 9.2.4 集成施密特触发器 CC40106 内部电路及其逻辑符号

(a) 电路图 (b) 逻辑符号

于是, T_{N1} 迅速导通, v_{o1} 随之下降, 致使 T_{P3} 很快导通, 进而使 T_{P1}, T_{P2} 截止, v_{o1} 下降为低电平。 v_i 继续升高, 最终使 T_{P1} 也完全截止, 输出电压 v_o 从高电平跳变为低电平。在 $V_{DD} \gg V_{TN} + |V_{TP}|$ 的条件下, 电路的正向阈值电压 V_{T+} 远大于 $\frac{1}{2}V_{DD}$ 。

同理, 当 v_i 逐渐下降的过程中, 与 v_i 上升过程类似, 电路也会出现一个急剧变化的工作过程, 使电路转换为 v_{o1} 高电平, $v_o = V_{OH}$ 的状态。在 v_i 下降过程中的负向阈值电压 V_{T-} 也远低于 $\frac{1}{2}V_{DD}$ 。

由上述分析可知, 电路在 v_i 上升和下降过程分别有不同的两个阈值电压, 电路为反相输出的施密特触发器。

值得指出的是, 由于集成电路内部器件参数差异较大, 电路的 V_{T+} 和 V_{T-} 的数值有较大的差异, 不同的 V_{DD} 有不同的 V_{T+}, V_{T-} 值, 即使 V_{DD} 相同, 不同的器件也有不同的 V_{T+} 和 V_{T-} 值。

9.2.3 施密特触发器的应用

施密特触发器的应用较广, 择其典型应用列举如下:

1. 波形变换

施密特触发器常用于波形变换, 如将正弦波、三角波等变换成矩形波等。将幅值大于 V_{T+} 的正弦波输入到施密特触发器的输入端, 根据施密特触发器的电压传输特性, 可画出输出电压波形, 如图 9.2.5 所示。结果表明, 通过电路在状态变化过程中的正反馈作用, 施密特触发器

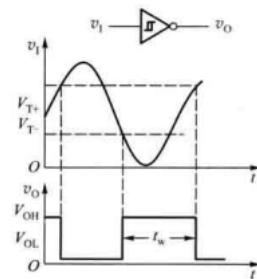


图 9.2.5 用施密特触发器实现波形变换

可将输入变化缓慢的周期信号变成了与其同频率、边缘陡直的矩形波。调节施密特触发器的 V_{T+} 或 V_{T-} , 可改变 v_o 的脉宽。

2. 波形的整形与抗干扰

在工程实际中, 常遇到信号在传输过程中发生畸变的现象。如当传输线上电容较大, 矩形波在传输过程中, 其上升沿和下降沿都会明显地被延缓, 其波形如图 9.2.6 (a) 所示。又如传输线较长, 且接收端的阻抗与传输线的阻抗也不匹配, 则在波形的上升沿和下降沿将产生阻尼振荡, 如图 9.2.6 (b) 所示中的输入信号。

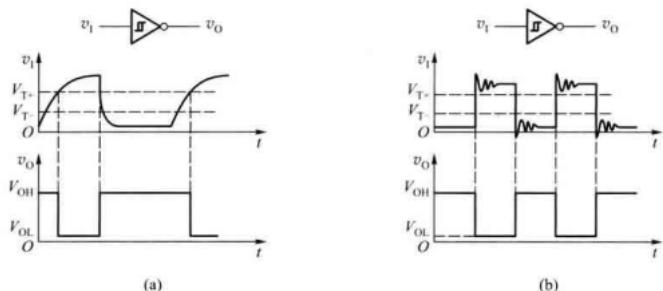


图 9.2.6 用施密特触发器实现脉冲波形的整形

(a) 改善上升沿和下降沿 (b) 消除振荡影响

对上述信号波形在传输过程中产生的畸变, 均可采用施密特触发器整形。图 9.2.6 (a)、(b) 所示施密特触发器工作波形说明, 只要回差电压选择合适, 就可达到理想的整形效果。

施密特触发器还可用于消除干扰。用施密特触发器消除干扰时, 回差电压的选择很重要。例如要消除图 9.2.7 (a) 所示矩形波的顶部干扰, 如回差电压取小了, 顶部干扰无法消除, 输出波形如图 9.2.7 (b) 所示。适当调大回差电压, 才能消除顶部干扰, 得到如图 9.2.7 (c) 所示的理想波形。

3. 幅度鉴别

施密特触发器属电平触发方式, 即其输出状态与输入信号 v_i 的幅值有关。利用这一工作特点, 可将它作为幅度鉴别电路。例如, 在施密特触发器输入端输入一串幅度不等的脉冲, 分析电路可得如图 9.2.8 所示输入、输出波形。图 9.2.8 表明, 只有幅度大于 V_{T+}

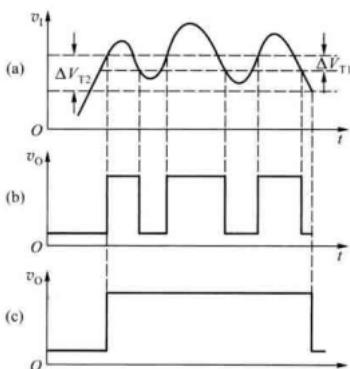


图 9.2.7 利用回差电压抗干扰

(a) 具有顶部干扰的输入信号

(b) 回差电压取值为 ΔV_{T1} 时的输出波形

(c) 回差电压取值为 ΔV_{T2} 时的输出波形

的那些脉冲才会使施密特触发器翻转, v_o 有脉冲输出; 而对于幅度小于 V_{T_1} 的脉冲, 施密特触发器不翻转, v_o 就没有脉冲输出。

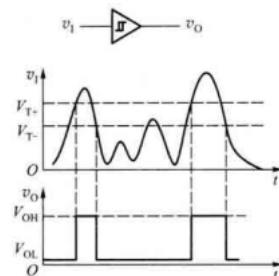


图 9.2.8 用施密特触发器进行幅度鉴别

复习思考题

- 9.2.1 施密特触发器具有怎样的工作特点? 试画出其电压传输特性曲线。
- 9.2.2 如果改变图 9.2.2 中 R_i 的取值, 对电路的传输特性有何影响?
- 9.2.3 为消除图 9.2.7a 中的顶部干扰, 应如何正确选择施密特触发器的正、负向阈值电压?

9.3 多谐振荡器

多谐振荡器是一种自激振荡电路, 它在接通电源后, 不需要外加触发信号, 电路就能自行产生一定频率和一定幅值的矩形波。由于矩形波含有丰富的谐波分量, 所以将它称之为多谐振荡器。多谐振荡器在工作过程中没有稳定状态, 故又被称为无稳态电路。

多谐振荡器的电路形式有多种, 但它们都具有如下共同的结构特点:

电路由开关器件和反馈延时环节组成。开关器件可以是逻辑门、电压比较器、定时器等, 其作用是产生脉冲信号的高、低电平。反馈延时环节一般由 RC 电路组成, 其作用是将输出电压延时后, 再反馈到开关器件的输入端, 以改变输出状态, 得到矩形波。

9.3.1 门电路组成的多谐振荡器

1. 电路组成及工作原理

由 CMOS 门电路和 RC 电路组成的多谐振荡器及其原理图分别如图 9.3.1(a)、(b) 所示。图 9.3.1(b) 中的 D_1 、 D_2 、 D_3 、 D_4 均为保护二极管。

(1) 第一暂稳态及电路自动翻转的过程

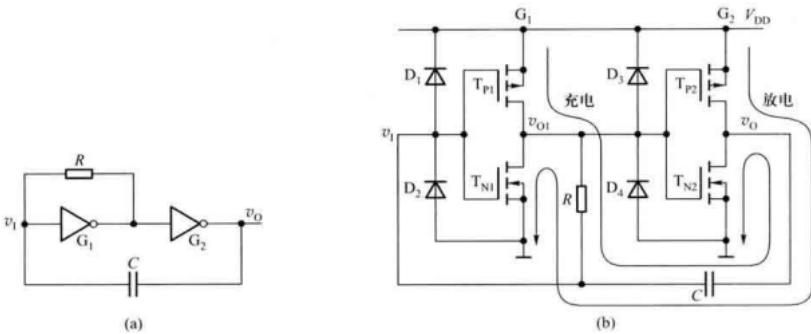


图 9.3.1 多谐振荡器及其原理图

(a) 由 CMOS 门电路组成的多谐振荡器 (b) 多谐振荡器原理图

假定在 $t=0$ 时接通电源, 电容 C 尚未充电, 电路初始状态为第一暂稳态, $v_{01} = V_{\text{on}}$, $v_i = v_o = V_{\text{on}}$ 。此时, 电源经 G_1 门的 T_{p1} 管、 R 和 G_2 门的 T_{n2} 管给电容 C 充电, 如图 9.3.1(b) 所示。随着充电时间的增加, v_i 的值不断上升, 当 v_i 达到 V_{th} 时, 电路发生下述正反馈过程:

$$\begin{array}{c} v_i \uparrow \rightarrow v_{01} \downarrow \rightarrow v_o \uparrow \\ \uparrow \qquad \downarrow \qquad \uparrow \end{array}$$

这一正反馈过程使 $v_{01} = V_{\text{on}}$, $v_o = V_{\text{on}}$, 电路转入第二暂稳态。

(2) 第二暂稳态及电路自动翻转的过程

在电路进入第二暂稳态瞬间, v_o 产生从 0 V 至 V_{dd} 的上跳, 由于电容两端电压不能突变, 则 v_i 也出现相同幅值的上跳, 保护二极管的钳位作用, 使得 v_i 仅上跳至 $V_{\text{dd}} + V_{\text{br}}$, V_{br} 为二极管正向导通压降。随后, 电容 C 通过 G_2 门的 T_{p2} 、电阻 R 和 G_1 门的 T_{n1} 放电, 使 v_i 下降, 当 v_i 降至 V_{th} 后, 电路又产生如下正反馈过程:

$$\begin{array}{c} v_i \downarrow \rightarrow v_{01} \uparrow \rightarrow v_o \downarrow \\ \downarrow \qquad \uparrow \qquad \downarrow \end{array}$$

于是, $v_{01} = V_{\text{on}}$, $v_o = V_{\text{on}}$, 电路又返回到第一暂稳态, $v_{01} = V_{\text{on}}$, $v_o = V_{\text{on}}$ 。此后, 电路重复上述过程。如此周而复始, 电路不断从一个暂稳态翻转到另一个暂稳态, 于是, 在 G_2 门的输出端得到方波信号。电路工作波形图如图 9.3.2 所示。

由上述分析可见, 多谐振荡器的两个暂稳态的转换过程是通过电容 C 充放电作用来实现的。电容的充、放电作用又集中体现在图中 v_i 的变化上。

2. 振荡周期的计算

多谐振荡器的振荡周期与两个暂稳态时间有关, 而两个暂稳态时间又分别由电容的充、放电时间决定。设电路的第一暂稳态和第二暂稳态时间分别为 T_1 , T_2 , 根据以上分析所得电路状态

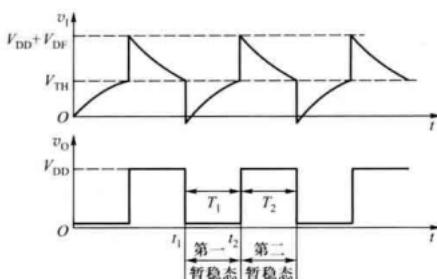


图 9.3.2 多谐振荡器波形图

转换时 v_i 的几个特征值, 可以计算电路的振荡周期。

(1) T_1 的计算

将图 9.3.2 中的 t_1 作为第一暂稳态起点, $T_1 = t_2 - t_1$, $v_i(0^+) = -V_{DF} \approx 0$ V, $v_i(\infty) = V_{DD}$, $\tau = RC$, 根据 RC 电路过渡过程的分析可知, v_i 由 0 V 变化到 V_{TH} 所需要的时间即为 T_1 ,

$$T_1 = RC \ln \frac{V_{DD}}{V_{DD} - V_{TH}} \quad (9.3.1)$$

(2) T_2 的计算

同理, 将 t_2 作为第二暂稳态时间起点有

$$v_i(0^+) = V_{DD} + V_{DF} \approx V_{DD}, v_i(\infty) = 0 \text{ V}, \tau = RC$$

由此可求出

$$T_2 = RC \ln \frac{V_{DD}}{V_{TH}} \quad (9.3.2)$$

所以

$$T = T_1 + T_2 = RC \ln \left[\frac{V_{DD}^2}{(V_{DD} - V_{TH}) \cdot V_{TH}} \right] \quad (9.3.3)$$

将 $V_{TH} = \frac{V_{DD}}{2}$ 代入上式, 得

$$T = RC \ln 4 \approx 1.4RC \quad (9.3.4)$$

9.3.2 用施密特触发器构成多谐振荡器

由于施密特触发器有 V_{T+} 和 V_{T-} 两个不同的阈值电压, 如能使其输入电压在 V_{T+} 和 V_{T-} 之间反复变化, 就可以在输出端得到矩形波。将施密特触发器的输出端经 RC 积分电路接回其输入端, 利用 RC 电路充、放电过程改变输入电压, 即可用施密特触发器构成多谐振荡器。用施密特触发器构成的多谐振荡器如图 9.3.3 所示。

(1) 工作原理

设在电源接通瞬间,电容器C的初始电压为零,输出电压 v_0 为高电平。 v_0 通过电阻R对电容器C充电,当 v_c 达到 V_{T+} 时,施密特触发器翻转, v_0 由高电平跳变为低电平。此后,电容器C又开始放电, v_c 下降,当它下降到 V_{T-} 时,电路又发生翻转, v_0 又由低电平跳变为高电平,C又被重新充电。如此周而复始,在电路的输出端,就得到了矩形波。 v_c 和 v_0 的波形如图9.3.4所示。

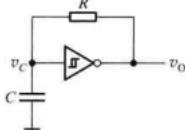


图9.3.3 用施密特触发器构成的多谐振荡器

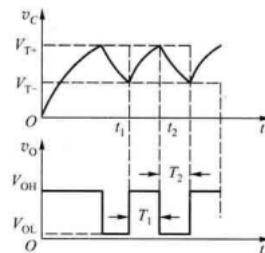


图9.3.4 图9.3.3所示电路的电压波形

(2) 振荡周期的计算

设在图9.3.3中采用CMOS施密特触发器CC40106,已知 $V_{OH} \approx V_{DD}, V_{OL} \approx 0$ V,则图9.3.4中的输出电压 v_0 的周期 $T = T_1 + T_2$,计算如下:

① T_1 的计算

以图9.3.4中 t_1 作为时间起点,则有

$$v_i(0^+) = V_{T+}, v_i(\infty) = V_{DD}, v_i(T_1) = V_{T+}, \tau = RC$$

根据 RC 电路暂态过渡过程公式,可求出

$$T_1 = RC \ln \frac{V_{DD} - V_{T-}}{V_{DD} - V_{T+}} \quad (9.3.5)$$

② T_2 的计算

以图9.3.4中 t_2 作为时间起点,则有

$$v_i(0^+) = V_{T+}, v_i(T_2) = V_{T-}, v_i(\infty) = 0, \tau = RC$$

利用 RC 电路暂态过程公式,可求出

$$T_2 = RC \ln \frac{V_{T+}}{V_{T-}} \quad (9.3.6)$$

③ 振荡周期 T 的计算

$$\begin{aligned} T &= T_1 + T_2 = RC \left(\ln \frac{V_{DD} - V_{T-}}{V_{DD} - V_{T+}} + \ln \frac{V_{T+}}{V_{T-}} \right) \\ &= RC \ln \left(\frac{V_{DD} - V_{T-}}{V_{DD} - V_{T+}} \cdot \frac{V_{T+}}{V_{T-}} \right) \end{aligned} \quad (9.3.7)$$

例 9.3.1 在图 9.3.3 中已知 $R = 10 \text{ k}\Omega$, $C = 0.022 \mu\text{F}$, CMOS 施密特触发器的 $V_{DD} = 5 \text{ V}$, $V_{on} \approx 5 \text{ V}$, $V_{OL} = 0 \text{ V}$, $V_{Ts} = 2.75 \text{ V}$, $V_{T-} = 1.67 \text{ V}$, 试计算输出波形的高电平持续时间 t_{ph} 、低电平持续时间 t_{pl} 和占空比 q 。

解: 电路的输出波形如图 9.3.4 所示。 t_{ph} 、 t_{pl} 实际上就是图 9.3.4 中的 T_1 和 T_2 , 由式(9.3.5)和式(9.3.6)可分别求出

$$\begin{aligned} t_{ph} &= T_1 = RC \ln \frac{V_{DD} - V_{T-}}{V_{DD} - V_{Ts}} \\ &= 10 \text{ k}\Omega \times 0.022 \mu\text{F} \cdot \ln \frac{5 - 1.67}{5 - 2.75} \\ &= 86.2 \mu\text{s} \\ t_{pl} &= T_2 = RC \ln \frac{V_{Ts}}{V_{T-}} \\ &= 10 \text{ k}\Omega \times 0.022 \mu\text{F} \cdot \ln \frac{2.75}{1.67} \approx 110 \mu\text{s} \end{aligned}$$

占空比 q 为

$$q = \frac{t_{ph}}{t_{ph} + t_{pl}} = \frac{86.2}{86.2 + 110} = 0.439 = 43.9\%$$

9.3.3 石英晶体多谐振荡器

在现代数字系统中,往往要求多谐振荡器的振荡频率具有很高的稳定性,否则,就会导致系统不能可靠地工作。式(9.3.3)表明,用门电路组成的多谐振荡器的振荡频率不仅与时间常数 RC 有关,而且还与门电路的阈值电压 V_{TH} 有关。当电源电压波动,温度变化(引起门电路的阈值电压 V_{TH} 变化)时,电路振荡频率稳定性会变得很差,用门电路组成的多谐振荡器很难适应现代数字系统的要求。在对振荡频率稳定性要求很高的电子设备中,只能采用由石英晶体组成的石英晶体振荡器。石英晶体振荡器的振荡频率不仅频率稳定度极高,而且频率范围也很宽,它的频率范围可从几百赫兹到几百兆赫兹。

石英晶体的电路符号及其阻抗频率特性分别如图 9.3.5(a)、(b) 所示。石英晶体阻抗频率特性表明,它的选频特性非常好,将它接入多谐振荡器的正反馈环路中后,只有频率为 f_s 的电压信号容易通过,在电路中形成正反馈,而其他频率信号都被石英晶体衰减,电路的振荡频率就是 f_s 。而 f_s 仅与石英晶体的结晶方向和外尺寸有关,而与电路中的电阻、电容无关。石英晶体振荡器的频率稳定度极高,其频率稳定度 $\Delta f_s/f_s$ 可达 $10^{-10} \sim 10^{-11}$ 。

一种典型的石英晶体振荡电路如图 9.3.6 所示,图中,电阻 R_1 和 R_2 的作用是使反相器 G_1 、 G_2 工作在电压特性曲线的转折区,有利于电路起振。如采用 TTL 门电路, R_1 和 R_2 通常取值为 $0.5 \sim 2 \text{ k}\Omega$ 之间;如采用 CMOS 门电路,其阻值则在 $5 \sim 100 \text{ M}\Omega$ 之间。电容 C_1 、 C_2 为两个反相器之间的耦合电容,电容 C_1 的大小选择应使其在频率为 f_s 时的容抗可以忽略不计,这样,可保证

G_1 和 G_2 之间形成正反馈环路。

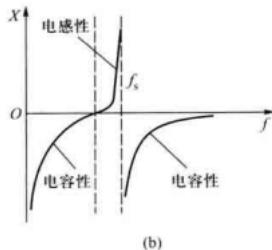


图 9.3.5 石英晶体的电路符号及阻抗频率特性

(a) 电路符号 (b) 阻抗频率特性

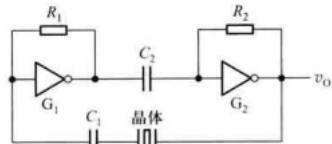
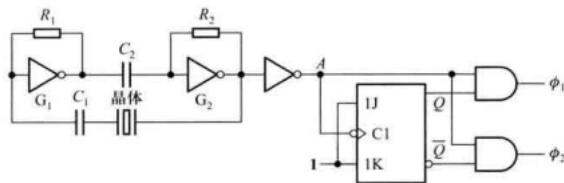


图 9.3.6 石英晶体振荡器

为了改善输出波形, 增强带负载的能力, 通常在振荡器的输出端再加一级反相器。图 9.3.7(a) 所示双相时钟产生电路可作为一个应用实例, 其工作波形如图 9.3.7(b) 所示。



(a)

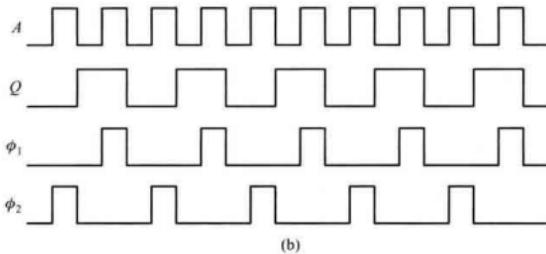


图 9.3.7 双相时钟发生器

(a) 逻辑图 (b) 波形图

复习思考题

9.3.1 简述多谐振荡器的工作特点, 并说明改变电路的哪些参数可以调节振荡频率。

9.3.2 你能用两片 74121 组成多谐振荡器吗？试画出电路图。

9.3.3 石英晶体振荡器有什么工作特点？其振荡频率等于什么？

9.4 555 定时器及其应用

555 定时器是一种模、数混合的中规模集成电路，它使用方便、灵活，应用极为广泛。用它可很方便地组成脉冲的产生、整形、延时和定时电路。

定时器有双极型和 CMOS 两种类型的产品，它们的结构及工作原理基本相同，没有本质的区别。一般说来，双极型定时器的驱动能力较强，电源电压范围为 5~16 V。而 CMOS 定时器的电源电压范围为 3~18 V，它具有低功耗、输入阻抗高等优点。

9.4.1 555 定时器

1. 电路结构

555 定时器的电路结构如图 9.4.1 所示。它由分压器、电压比较器 C_1 和 C_2 、简单 SR 锁存器、放电三极管 T 以及缓冲器 G 组成。阈值输入端 v_{th} 是比较器 C_1 的信号输入端，触发输入端 v_{tr} 是比较器 C_2 的信号输入端。

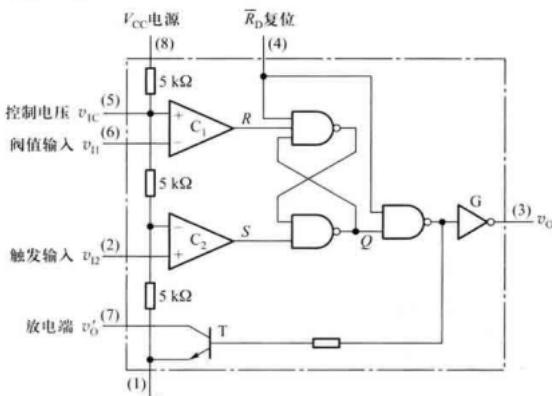


图 9.4.1 555 定时器的电路结构

当控制电压端(5)悬空时(此时，一般该端接上 0.01 μF 左右的滤波电容)，三个 5 k Ω 的电阻串联组成的分压器，为比较器 C_1 、 C_2 分别提供 $\frac{2}{3}V_{cc}$ 和 $\frac{1}{3}V_{cc}$ 的基准电压。如控制电压端(5)外

接电压 v_{ic} , 则比较器 C_1, C_2 的基准电压就变为 v_{ic} 和 $v_{ic}/2$ 。

比较器 C_1 和 C_2 的输出控制 SR 锁存器和放电三极管 T 的状态。

放电三极管 T 为外接电路提供放电通路, 定时器在使用时, T 的集电极(7脚)一般都要外接一个上拉电阻。

\bar{R}_b 为直接复位输入端, 当 \bar{R}_b 为低电平时, 不管其他输入端的状态如何, 输出端 v_o 即为低电平。

当 $v_{11} > \frac{2V_{cc}}{3}, v_{12} > \frac{V_{cc}}{3}$ 时, 比较器 C_1 输出低电平, 比较器 C_2 输出高电平, 简单 SR 锁存器 Q 端置 0, T 导通, 输出端 v_o 为低电平。

当 $v_{11} < \frac{2V_{cc}}{3}, v_{12} < \frac{V_{cc}}{3}$ 时, 比较器 C_1 输出高电平, 比较器 C_2 输出低电平, 简单 SR 锁存器置 1, T 截止, 输出端 v_o 为高电平。

当 $v_{11} < \frac{2V_{cc}}{3}, v_{12} > \frac{V_{cc}}{3}$ 时, 简单 SR 锁存器 $R=1, S=1$, 锁存器状态不变, 电路保持原状态不变。

2. 电路功能

综合上述分析, 可得 555 定时器功能表如表 9.4.1 所示。

表 9.4.1 555 定时器功能表

输入			输出	
阈值输入(v_{11})	触发输入(v_{12})	复位(\bar{R}_b)	输出(v_o)	放电管 T
x	x	0	0	导通
$<2V_{cc}/3$	$<V_{cc}/3$	1	1	截止
$>2V_{cc}/3$	$>V_{cc}/3$	1	0	导通
$<2V_{cc}/3$	$>V_{cc}/3$	1	不变	不变

9.4.2 用 555 组成的施密特触发器

将 555 定时器的阈值输入端和触发输入端相接, 即构成施密特触发器, 电路和简化电路分别如图 9.4.2(a)、(b) 所示。

如果 v_i 由 0 V 开始逐渐增加, 当 $v_i < \frac{V_{cc}}{3}$ 时, 根据 555 定时器功能表可知, 输出 v_o 为高电平; v_i

继续增加, 如果 $\frac{V_{cc}}{3} < v_i < \frac{2V_{cc}}{3}$, 输出 v_o 维持高电平不变; v_i 再增加, 一旦 $v_i > \frac{2V_{cc}}{3}$, v_o 就由高电平跳

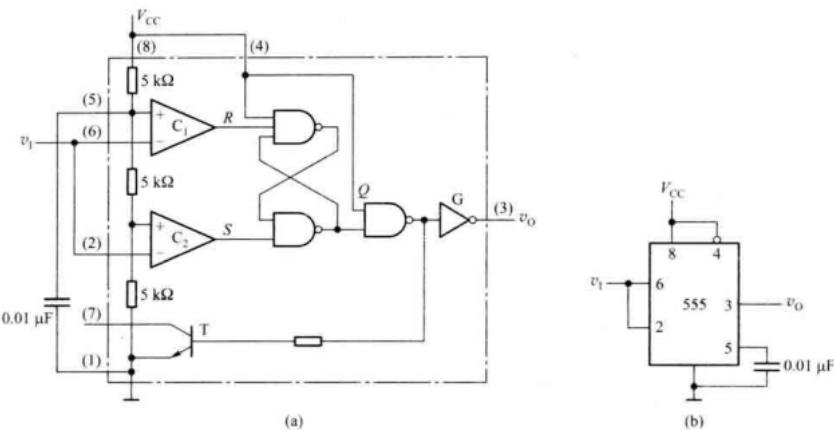


图 9.4.2 用 555 构成的施密特触发器

(a) 电路 (b) 简化电路

变为低电平；之后 v_1 再增加，仍是 $v_1 > \frac{2V_{CC}}{3}$ ，输出保持低电平不变。

如果 v_1 由大于 $\frac{2}{3}V_{CC}$ 的电压值逐渐下降，只要 $\frac{V_{CC}}{3} < v_1 < \frac{2V_{CC}}{3}$ ，电路输出状态不变仍为低电平；只有当 $v_1 < \frac{V_{CC}}{3}$ 时，电路才再次翻转， v_0 就由低电平跳变为高电平。

如果输入 v_1 为三角波，电路的工作波形和电压传输特性曲线分别如图 9.4.3 (a)、(b) 所示。

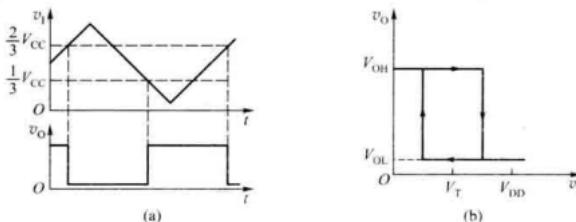


图 9.4.3 施密特触发器的工作波形及电压传输特性曲线

(a) 工作波形图 (b) 电压传输特性曲线

图 9.4.2 所示施密特触发器的正、负向阈值电压分别为 $\frac{2}{3}V_{cc}$ 和 $\frac{1}{3}V_{cc}$ 。不难理解,如施密特触发器控制电压(5脚)端接 V_{in} ,改变 V_{in} 可以调节电路的回差电压。

9.4.3 用 555 组成的单稳态触发器

用 555 构成的单稳态触发器和简化电路分别如图 9.4.4(a)、(b) 所示。

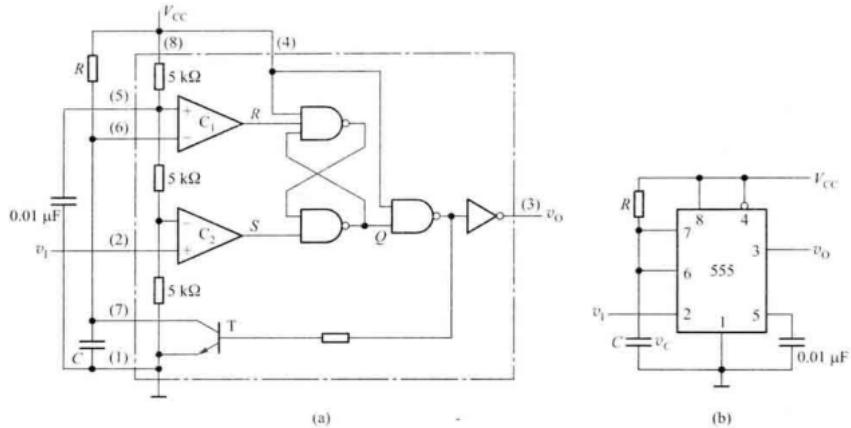


图 9.4.4 用 555 构成的单稳态触发器

(a) 电路 (b) 简化电路

没有触发信号时 v_I 处于高电平 ($v_I > \frac{V_{cc}}{3}$), 如接通电源后 $Q=0, v_o=0$, T 导通, 电容通过 T 放电, 使 $v_c=0, v_o$ 保持低电平不变。如接通电源后 $Q=1$, T 就会截止, 电源通过电阻 R 向电容 C 充电, 当 v_c 上升到 $\frac{2V_{cc}}{3}$ 时, 由于 $R=0, S=1$, 锁存器置 0, v_o 为低电平。此时 T 导通, 电容 C 放电, v_o 保持低电平不变。因此, 电路通电后在没有触发信号时, 电路只有一种稳定状态, $v_o=0$ 。

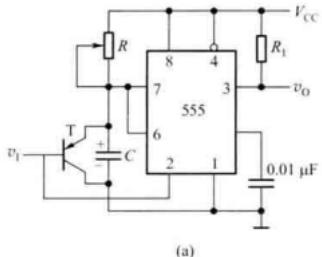
若触发输入端施加触发信号 ($v_I < \frac{V_{cc}}{3}$), 电路的输出状态由低电平跳变为高电平, 电路进入暂稳态, 三极管 T 截止。此后电容 C 充电, 当电容充电至 $v_c=\frac{2V_{cc}}{3}$ 时, 电路的输出电压 v_o 由高电平翻转为低电平, 同时 T 导通, 于是电容 C 放电, 电路返回到稳定状态。电路的工作波形如图 9.4.5 所示。

如果忽略 T 的饱和压降, 则 v_c 从零电平上升到 $\frac{2V_{CC}}{3}$ 的时间, 即为输出电压 v_o 的脉宽 t_w 。

$$\begin{aligned} t_w &= R C \ln \frac{V_{CC} - 0}{V_{CC} - \frac{2}{3} V_{CC}} \\ &= R C \ln 3 = 1.1 R C \end{aligned} \quad (9.4.1)$$

通常 R 的取值在几百欧至几兆欧之间, 电容取值为几百皮法到几百微法。这种电路产生的脉冲宽度可从几个微秒到数分钟, 精度可达 0.1%。由图 9.4.5 可知, 如果在电路的暂稳态持续时间内, 加入新的触发脉冲(如图 9.4.5 中的虚线所示), 则该脉冲对电路不起作用、电路为不可重复触发单稳态触发器。

由 555 定时器构成的可重复触发单稳态电路及工作波形分别如图 9.4.6(a)、(b) 所示。



(a)

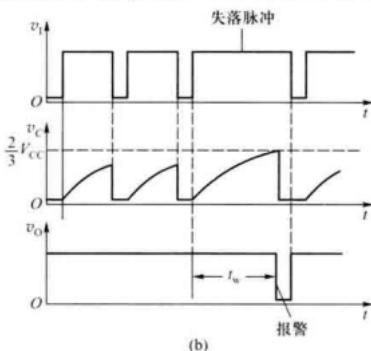


图 9.4.6 可重复触发的单稳态电路

(a) 电路图 (b) 工作波形

当 v_1 输入负向脉冲后, 电路进入暂稳态, 同时三极管 T 导通, 电容 C 放电。输入负脉冲撤除后, 电容 C 充电, 在 v_c 未充到 $\frac{2V_{CC}}{3}$ 之前, 电路处于暂稳态。如果在此期间, 又加入新的触发脉冲, 三极管 T 又导通, 电容 C 再次放电, 输出仍然维持在暂稳态。只有在触发脉冲撤除后, 在输出脉宽 t_w 时间内没有新的触发脉冲, 电路才返回到稳定状态。该电路可用作失落脉冲检测, 或对电机转速或人体的心律进行监视, 如转速不稳或人体的心律不齐时, v_o 的低电平可用作报警信号。

用单稳态电路组成的脉宽调制电路如图 9.4.7(a) 所示。在单稳态电路的电压控制端输入

三角波,当输入电压升高时,电路的阈值电压升高,输出的脉冲宽度随之增加;而当输入电压降低时,电路的阈值电压也降低,输出的脉冲宽度则随之减小。随着输入电压的变化,在单稳态的输出端,就可得到一串随控制电压变化的脉宽调制波形,如图 9.4.7(b) 所示。

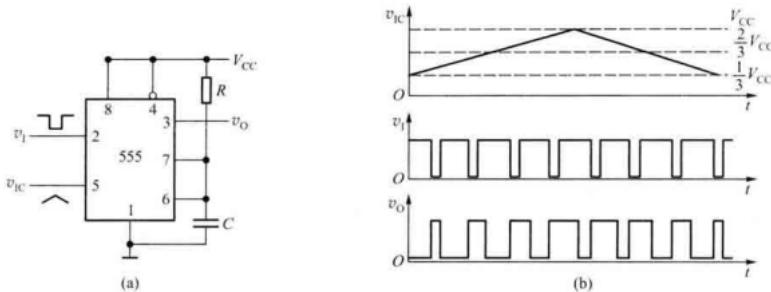


图 9.4.7 脉冲宽度调制器
(a) 电路 (b) 波形图

9.4.4 用 555 组成的多谐振荡器

由 555 定时器构成的多谐振荡器如图 9.4.8(a) 所示。接通电源后,电容 C 被充电,当 v_c 上升到 $\frac{2}{3}V_{cc}$ 时,使 v_o 为低电平,同时 T 导通,此时电容 C 通过 R_2 和 T 放电, v_c 下降。当 v_c 下降到 $\frac{1}{3}V_{cc}$ 时, v_o 翻转为高电平。电容器 C 放电所需的时间为

$$t_{pl} = R_2 C \ln \frac{0 - V_{T-}}{0 - V_{T+}} = R_2 C \ln 2 = 0.7 R_2 C \quad (9.4.2)$$

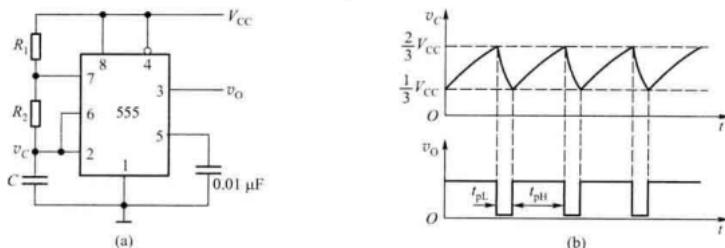


图 9.4.8 多谐振荡器
(a) 电路图 (b) 工作波形

当放电结束时, T 截止, V_{cc} 将通过 R_1, R_2 向电容器 C 充电, v_c 由 $\frac{V_{cc}}{3}$ 上升到 $\frac{2V_{cc}}{3}$ 所需的时间为

$$\begin{aligned} t_{ph} &= (R_1 + R_2) C \ln \frac{V_{cc} - V_{tr-}}{V_{cc} - V_{tr+}} \\ &= (R_1 + R_2) C \ln 2 \approx 0.7 (R_1 + R_2) C \end{aligned} \quad (9.4.3)$$

当 v_c 上升到 $\frac{2V_{cc}}{3}$ 时, 电路又翻转为低电平。如此周而复始, 于是, 在电路的输出端就得到一个周期性的矩形波。电路的工作波形如图 9.4.8 (b) 所示, 其振荡频率为

$$f = \frac{1}{t_{ph} + t_{pl}} \approx \frac{1.43}{(R_1 + 2R_2) C} \quad (9.4.4)$$

由于 555 内部的比较器灵敏度较高, 而且采用差分电路形式, 用 555 构成的多谐振荡器的振荡频率受电源电压和温度变化的影响很小。

图 9.4.8 (a) 所示电路的 $t_{ph} \neq t_{pl}$, 而且占空比固定不变。如要实现占空比可调可采用如图 9.4.9 所示电路。由于电路中二极管 D₁、D₂ 的单向导电特性, 使电容器 C 的充放电回路分开, 调节电位器, 就可调节多谐振荡器的占空比。图中, V_{cc} 通过 R_A 、D₁ 向电容 C 充电, 充电时间为

$$t_{ph} \approx 0.7 R_A C \quad (9.4.5)$$

电容器 C 通过 D₂、 R_B 及 555 中的三极管 T 放电, 放电时间为

$$t_{pl} \approx 0.7 R_B C \quad (9.4.6)$$

因而, 振荡频率为

$$f = \frac{1}{t_{ph} + t_{pl}} \approx \frac{1.43}{(R_A + R_B) C} \quad (9.4.7)$$

电路输出波形的占空比为

$$q(\%) = \frac{R_A}{R_A + R_B} \times 100\% \quad (9.4.8)$$

上面仅讨论了由 555 定时器组成的单稳态触发器、多谐振荡器和施密特触发器, 实际上, 由于 555 定时器的比较器灵敏度高、输出驱动电流大、功能灵活, 因而在电子电路中获得了广泛应用, 限于篇幅, 这里就不一一列举了。

复习思考题

9.4.1 555 定时器中缓冲器 G 有什么作用?

9.4.2 555 定时器的功能表如何? 用它能组成哪些脉冲波形的变换与产生电路?

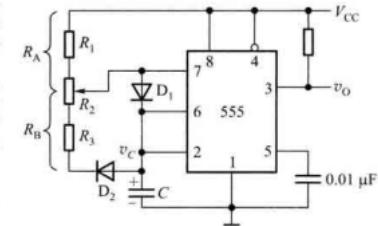


图 9.4.9 占空比可调的方波发生器

9.4.3 如将图 9.4.2(a) 中 555 定时器的控制电压输入端(5 脚)改接电压 V_u 上,试问电路的输出脉宽 t_s 有无改变? t_s 与 V_u 的关系如何?

9.4.4 能否采用改变图 9.4.8(a) 中 555 定时器控制电压的方法调节电路的振荡频率? 为什么? 如要实现振荡和停振可控,你会采用什么方法来实现?

小结

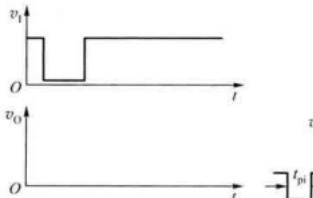
- 集成单稳态触发器分为非重复触发和可重复触发两大类,在暂稳态期间,出现的触发信号对非重复触发单稳态电路的输出脉宽没有影响,而对可重复触发单稳态电路可起到连续触发作用。
- 在数字电路中,施密特触发器实质上是具有滞后特性的逻辑门,它有两个阈值电压。电路状态与输入电压有关,不具备记忆功能。
- 多谐振荡器无需外加输入信号就能在接通电源后自行产生矩形波输出。在频率稳定性要求较高的场合通常采用石英晶体振荡器。
- 定时器是一种应用广泛的集成器件,多用于脉冲产生、整形及定时等。除 555 定时器外,目前还有 556(双定时器)、558(四定时器)等。

习题

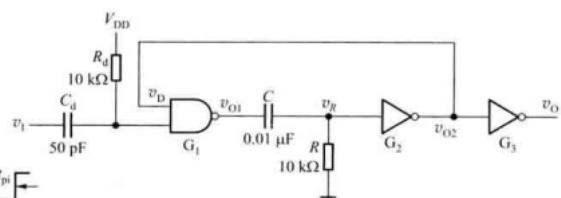
9.1 单稳态触发器

9.1.1 在图 9.1.1(a) 所示微分型单稳态电路中,已知 $C=0.01 \mu\text{F}$, $R=9.1 \text{ k}\Omega$, 电源电压 $V_{DD}=10 \text{ V}$ 。试对应图题 9.1.1 所示的输入波形画出电路输出 v_o 的波形,并计算输出脉宽 t_s 及幅值大小。

9.1.2 用 CMOS 逻辑门组成的微分型单稳态电路如图题 9.1.2 所示。其中 $t_p = 3 \mu\text{s}$, $C_d = 50 \text{ pF}$, $R_d = 10 \text{ k}\Omega$, $C = 0.01 \mu\text{F}$, $R = 10 \text{ k}\Omega$, 试对应地画出 v_t , v_B , v_{O1} , v_R , v_{O2} , v_o 的波形,并求出输出脉冲宽度。



图题 9.1.1



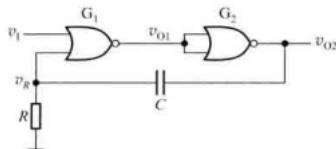
图题 9.1.2

9.1.3 图题 9.1.3 所示电路是用 CMOS 或非门构成的单稳态触发器的另一种形式。试回答下列问题:

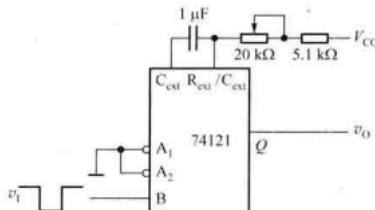
- (1) 分析电路的工作原理;
- (2) 画出加入触发脉冲后 v_{O1} , v_{O2} 及 v_o 的工作波形;
- (3) 写出输出脉宽 t_s 的表达式。

9.1.4 由集成单稳态触发器 74121 组成的延时电路及输入波形如图题 9.1.4 所示。

- (1) 计算输出脉宽的变化范围;
 (2) 解释为什么使用电位器时要串接一个电阻。



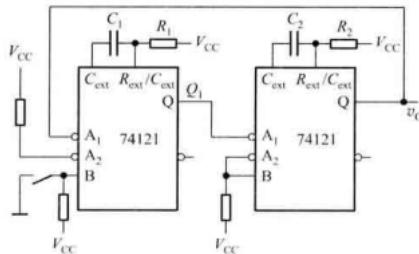
图题 9.1.3



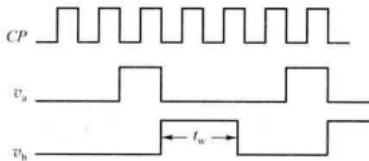
图题 9.1.4

9.1.5 利用两片集成单稳态触发器 74LS121 可构成的多谐振荡器如图题 9.1.5 所示，试说明其工作原理，并计算电路的振荡频率。

9.1.6 某控制系统要求产生的信号 v_a, v_b 与系统时钟 CP 的时序关系如图题 9.1.6 所示。试用计数器 74161、集成单稳 74121 设计该信号产生电路，画出电路图。



图题 9.1.5

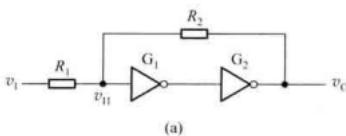


图题 9.1.6

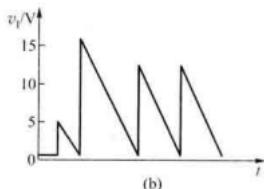
9.2 施密特触发器

9.2.1 在图题 9.2.1 所示的施密特触发器电路中，已知 $R_1 = 10 \text{ k}\Omega, R_2 = 30 \text{ k}\Omega$ 。 G_1, G_2 为 CMOS 反相器，

$$V_{DD} = 15 \text{ V}, V_T = \frac{1}{2} V_{DD}$$



(a)



图题 9.2.1

(1) 试计算电路的上门限触发电平 V_{t+} 、下门限触发电平 V_{t-} 和回差电压 ΔV_t 。

(2) 若电路的输入信号 v_i 波形如图题 9.2.1(b) 所示, 试画出相应的输出电压 v_o 的波形。

9.2.2 图题 9.2.2 所示为一个回差可调的施密特电路, 它是利用射极跟随器的射极电阻来调节回差的。

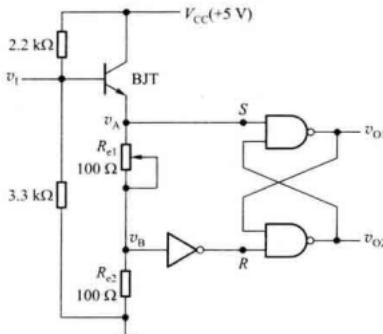
(1) 分析电路的工作原理;

(2) 当 R_s 在 $50 \sim 100 \Omega$ 的范围内变动时, 试计算回差的变化范围。

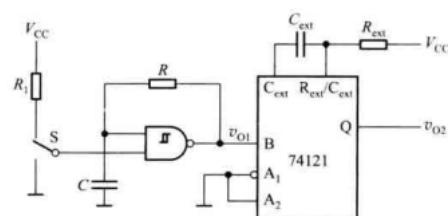
9.2.3 集成施密特电路和集成单稳态触发器 74121 构成如图题 9.2.3 所示电路。已知集成施密特电路的 $V_{DD} = 10 \text{ V}$, $R = 100 \text{ k}\Omega$, $C = 0.01 \mu\text{F}$, $V_{t+} = 6.3 \text{ V}$, $V_{t-} = 2.7 \text{ V}$, $C_{ext} = 0.01 \mu\text{F}$, $R_{ext} = 30 \text{ k}\Omega$ 。

(1) 分别计算 v_{o1} 的周期及 v_{o2} 的脉宽;

(2) 根据计算结果画出 v_{o1} 、 v_{o2} 的波形。



图题 9.2.2



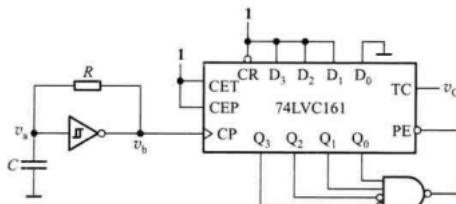
图题 9.2.3

9.2.4 集成施密特触发器和 4 位计数器 74 LVC 161 组成的电路如图题 9.2.4 所示。

(1) 分别说明图中两部分电路的功能;

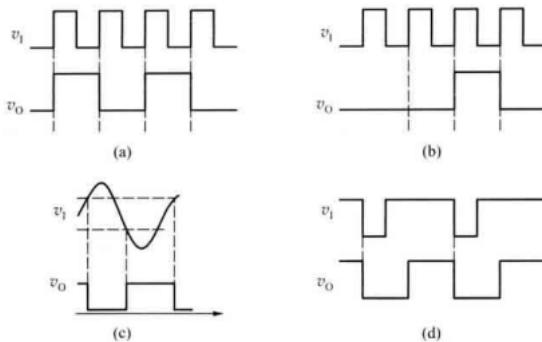
(2) 画出图中 74LVC161 组成的电路的状态图;

(3) 画出图中 v_a 、 v_b 和 v_o 的对应波形。



图题 9.2.4

9.2.5 已知几种电路的输入、输出波形如图题 9.2.5 所示, 试问应分别选取何种电路才能实现如图所示工作波形?



图题 9.2.5

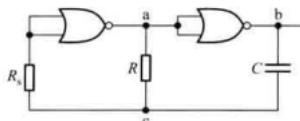
9.3 多谐振荡器

9.3.1 图题 9.3.1 所示电路为 CMOS 或非门构成的多谐振荡器, 图中 $R_s = 10R$ 。

(1) 画出 a、b、c 各点的波形;

(2) 计算电路的振荡周期;

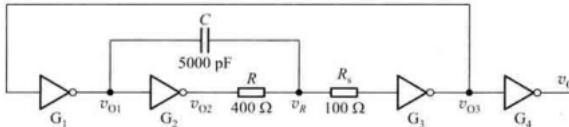
(3) 试问当阈值电压 V_{th} 由 $\frac{V_{DD}}{2}$ 改变至 $\frac{2V_{DD}}{3}$ 时, 电路的振荡频率



如何变化?

9.3.2 RC 环型多谐振荡电路如图题 9.3.2 所示, 试分析电路的振荡过程, 画出 v_{o1}, v_{o2}, v_R 及 v_o 的波形。

图题 9.3.1



图题 9.3.2

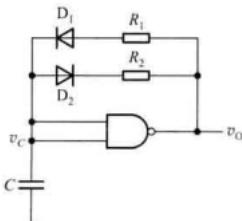
9.3.3 由集成施密特 CMOS 与非门电路组成的脉冲占空比可调多谐振荡器如图题 9.3.3 所示。设电路中 R_1, R_2, C 及 V_{th+}, V_{th-} 的值已知。

(1) 定性画出 v_c 及 v_o 波形;

(2) 写出输出信号 v_o 频率的表达式。

9.4 555 定时器及其应用

9.4.1 图题 9.4.1 所示为一简易触摸开关电路。当手摸金属片时, 发光二极管亮, 经过一定时间后, 发光二极管自动熄灭。试说明其工作原理, 问手摸金属片后发光二极管亮多长时间自动熄灭?

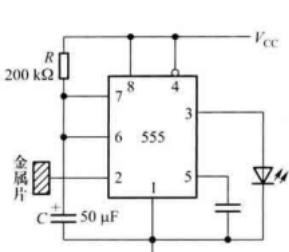


图题 9.3.3

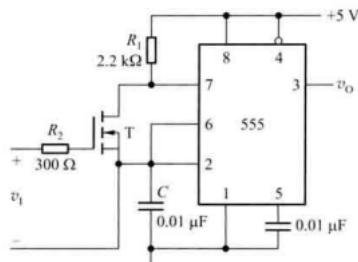
9.4.2 由 555 定时器及场效应管 T 组成的电路如图题 9.4.2 所示, 电路

中 T 工作于可变电阻区, 其导通电阻为 R_{DS} 。

- (1) 说明电路功能;
- (2) 写出输出 v_o 频率的表达式。

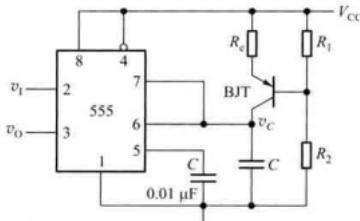


图题 9.4.1



图题 9.4.2

9.4.3 由 555 定时器构成的锯齿波发生器如图题 9.4.3 所示, 三极管 T 和电阻 R_1 、 R_2 、 R_c 构成恒流源, 给定时电容 C 充电, 当触发输入端输入负脉冲后, 画出电容电压 v_c 及 555 输出端 v_o 的波形, 并计算电路的输出脉宽。



图题 9.4.3

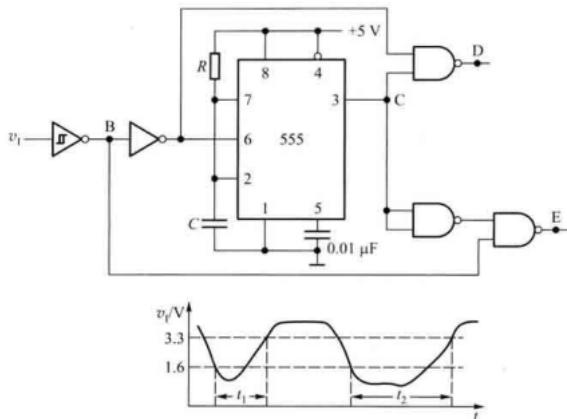
9.4.4 由 555 定时器组成的脉冲宽度鉴别电路及输入 v_i 波形如图题 9.4.4 所示。集成施密特的 $V_{tr} = 3.3 \text{ V}$, $V_{t-} = 1.6 \text{ V}$ 。已知单稳的输出脉宽 t_s 与 t_1 , t_2 关系为 $t_s < t_1 < t_2$ 。对应 v_i 画出电路中 B、C、D、E 各点波形, 并说明 D、E 端输出负脉冲的作用。

9.4.5 图题 9.4.5(a) 所示为心律失常报警电路, 经放大后的心电信号 v_i 如图题 9.4.5(b) 所示, v_i 的幅值 $V_{Im} = 4 \text{ V}$ 。

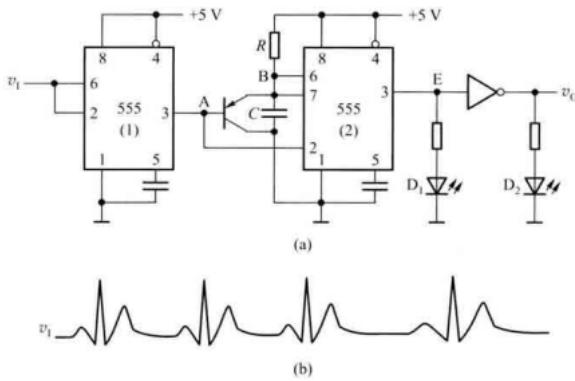
- (1) 对应 v_i 分别画出图中 A、B、E 三点的电压波形;
- (2) 说明电路的组成及工作原理。

9.4.6 一防盗报警电路如图题 9.4.6 所示, a、b 两端被一细铜丝接通, 此铜丝置于小偷必经之处。当小偷闯入室内将铜丝碰断后, 扬声器即发出报警声。(扬声器电压为 1.2 V, 通过电流为 40 mA)。

- (1) 试问 555 定时器接成何种电路?
- (2) 简要说明该报警电路的工作原理。



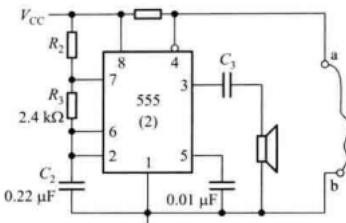
图题 9.4.4



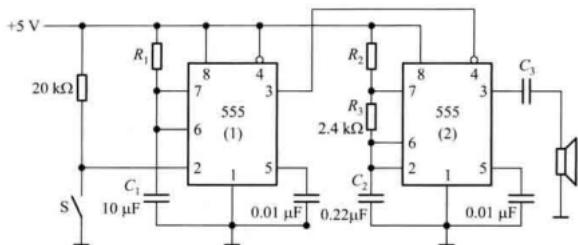
图题 9.4.5

9.4.7 分析图题 9.4.7 所示电路, 简述电路组成及工作原理。若要求扬声器在开关 S 按下后以 1.2 kHz 的频率持续响 10 s, 试确定图中 R_1 , R_2 的阻值。

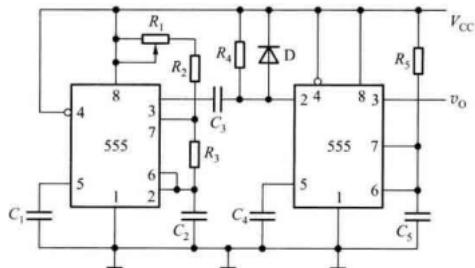
9.4.8 图题 9.4.8 电路为两个 555 定时器构成的频率可调而脉宽不变的方波发生器, 试说明工作原理; 确定频率变化的范围和输出脉宽; 解释二极管 D 在电路中的作用。



图题 9.4.6



图题 9.4.7



图题 9.4.8

数模与模数转换器

引言

随着数字技术,特别是计算机技术的飞速发展与普及,在现代控制、通信及检测领域中,信号的处理无不广泛地采用了计算机技术。由于自然界中的物理量,如压力、温度、位移、液位等都是模拟量,如要用数字技术处理这些模拟信号,则往往需要一种能在模拟信号与数字信号之间起转换作用的电路——模数转换器和数模转换器。

能将模拟信号转换成数字信号的电路称为模数转换器(Analog to Digital Converter,简称ADC或A/D转换器);反之,能把数字信号转换为模拟信号的电路称为数模转换器(Digital to Analog Converter,简称为DAC或D/A转换器)。A/D和D/A转换器是现代计算机系统中不可缺少的基本组成部分。

A/D转换器和D/A转换器在工业控制系统中的作用如图10.0所示。

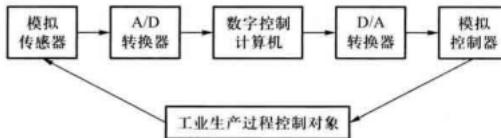


图10.0 A/D转换器与D/A转换器在工业控制系统中的作用

本章主要介绍几种常用D/A与A/D转换器的电路结构、工作原理及其应用。

10.1 D/A转换器

10.1.1 D/A转换器的输入/输出特性及其结构框图

1. D/A转换器的输入/输出特性

D/A转换器输入数字量 D 与输出模拟量 A 之间的转换关系式为

$$A = KD$$

(10.1.1)

式中 K 为比例系数, 它是一个常数。 D 为 n 位二进制数, 可写成

$$D = \sum_{i=0}^{n-1} d_i 2^i \quad (10.1.2)$$

式中 d_i 为第 i 位的二进制数。

理想的 3 位 D/A 转换器的输入/输出特性如图 10.1.1 所示。图中 V_{LSB} 为输入数字量中最有效位(LSB) d_0 变化所引起的输出电压的变化值。图 10.1.1 表明, D/A 转换器输出为阶梯波信号。

2. D/A 转换器的结构框图

n 位 D/A 转换器的一般结构框图如图 10.1.2 所示。数字量以串行或并行方式输入并存储于数码寄存器中, 寄存器的输出驱动对应的数位上的电子开关, 将相应数位的权值送入求和电路。求和电路将各位的权值相加得到与数字量对应的模拟量。

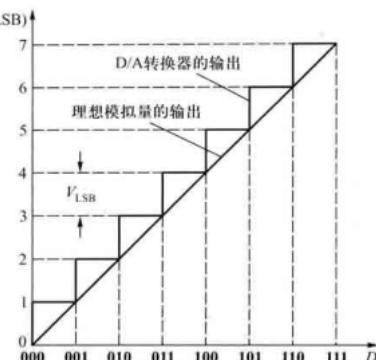


图 10.1.1 理想 D/A 转换器的输入/输出特性



图 10.1.2 n 位 D/A 转换器一般方框图

根据解码网络结构的不同, D/A 转换器有多种类型。如有电阻网络型(如倒 T 形电阻网络 D/A 转换器、T 形电阻网络 D/A 转换器)、电容网络型(如权电容网络 D/A 转换器)和电阻电容混合型及晶体管混合型等。这些 D/A 转换器按数字量的输入方式的不同又分为并行输入 D/A 转换器和串行输入 D/A 转换器两种。

10.1.2 D/A 转换器的基本原理

任何一个二进制数 $D_{n-1} D_{n-2} \cdots D_1 D_0$ 可以按下式转换为十进制数,

$$(N)_B = D_n \times 2^n + D_{n-1} \times 2^{n-1} + \cdots + D_1 \times 2^1 + D_0 \times 2^0$$

式中 $2^n, 2^{n-1}, \cdots, 2^1, 2^0$ 为各数位的权。上式表明, 要实现数模转换, 首先要将输入二进制数中为 1 的每一位代码按其权的大小, 转换成模拟量, 然后将这些模拟量相加, 相加所得的总量就是与数字量成正比的模拟量。根据这个思路, 可得 4 位 D/A 转换器的原理电路如图 10.1.3 所示。电路由寄存器、基准电压、电子开关、权电阻网络、求和电路等组成。

寄存器输出的 4 位二进制码 $D_3 \sim D_0$ 分别控制着开关 $S_3 \sim S_0$, 当 $D_i = 1$ ($i = 0, 1, 2, 3$) 时, 开关

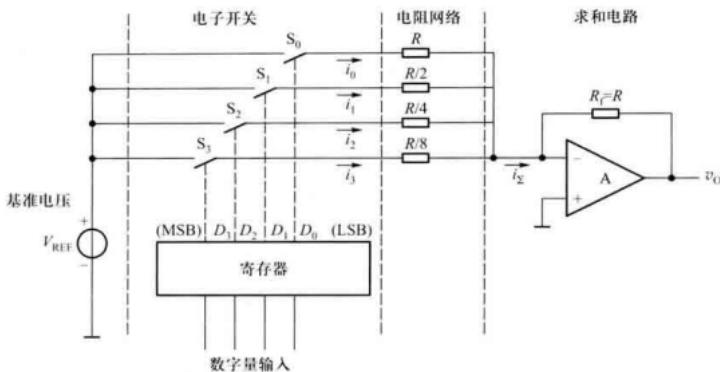


图 10.1.3 D/A 转换器原理电路

S_i 与基准电压 $+V_{REF}$ 接通, 电流 i_i 流入求和电路; 当 $D_i = 0$ 开关断开, $i_i = 0$ 。

根据运算放大器在线性运用条件下, 虚短、虚断的特点, 可得

$$\begin{aligned} v_o &= -i \frac{R_f}{R_i} \\ &= -R_f (i_3 + i_2 + i_1 + i_0) \end{aligned} \quad (10.1.3)$$

若令 $R_f = R$, 可得 $i_3 = \frac{8V_{REF}D_3}{R}$, $i_2 = \frac{4V_{REF}D_2}{R}$, $i_1 = \frac{2V_{REF}D_1}{R}$, $i_0 = \frac{V_{REF}D_0}{R}$,

将它们代入式(10.1.3)则得

$$\begin{aligned} v_o &= -V_{REF}(D_32^3 + D_22^2 + D_12^1 + D_02^0) \\ &= -V_{REF} \sum_{i=0}^3 D_i \cdot 2^i \end{aligned} \quad (10.1.4)$$

结果表明, 电路实现了数模转换。

上述分析表明, 电阻网络中的各支路电流 $i_i = \frac{2^i V_{REF}}{R}$ 为二进制数各位的权值; D_i 控制 S_i , 以决定相应支路的权值电流 i_i 是否流入电流求和网络; 而求和电路则将总电流转换为模拟电压 v_o 输出, 电路实现设计初衷。由于这种电路中每一位电阻的阻值与这一位的“权”相对应, 位权越大, 对应的阻值越小, 所以, 该电路也称为权电阻网络 D/A 转换电路。

10.1.3 倒 T 形电阻网络 D/A 转换器

在单片集成 D/A 转换器中, 使用最多的是倒 T 形电阻网络 D/A 转换器。下面我们以 4 位 D/A 转换器为例说明其工作原理。

1.4 位倒 T 形电阻网络 D/A 转换器

4 位倒 T 形电阻网络 D/A 转换器的原理图如图 10.1.4 所示。电路因电阻解码网络呈倒 T

形而得名。图中,模拟开关 S_i 由输入数码 D_i 控制,当 $D_i = 0$, S_i 接地;当 $D_i = 1$, S_i 接运算放大器反相端,倒 T 形的电阻解码网络与运算放大器 A 组成求和电路。

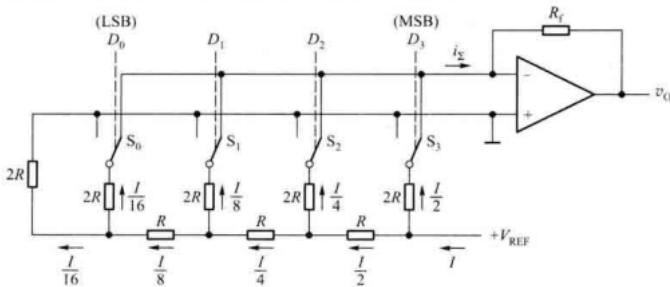


图 10.1.4 4 位倒 T 形电阻网络 D/A 转换器

运放工作在线性运用状态,其反相端虚地。这样,无论模拟开关 S_i 置于何种位置,与 S_i 相连的 $2R$ 电阻总是接“地”,所以,流经每条 $2R$ 电阻支路上的电流与开关状态无关。

分析图 10.1.4 所示电路可发现,从每个节点向左看,每个二端网络的等效电阻均为 R ,与开关相连的 $2R$ 电阻上的电流从高位到低位按 2 的负整数幂递减。如基准电压源提供的总电流为 I ($I = V_{REF}/R$),则流过各开关支路(从右到左)的电流分别为 $I/2$ 、 $I/4$ 、 $I/8$ 、和 $I/16$ 。

于是,可得总电流

$$\begin{aligned} i_{\Sigma} &= \frac{V_{REF}}{R} \left(\frac{D_0}{2^4} + \frac{D_1}{2^3} + \frac{D_2}{2^2} + \frac{D_3}{2^1} \right) \\ &= \frac{V_{REF}}{2^4 \times R} \sum_{i=0}^3 (D_i \cdot 2^i) \end{aligned} \quad (10.1.5)$$

输出电压

$$v_0 = -i_{\Sigma} R_f = -\frac{R_f}{R} \cdot \frac{V_{REF}}{2^4 \times R} \sum_{i=0}^3 (D_i \cdot 2^i) \quad (10.1.6)$$

如将输入数字量扩展到 n 位,可得 n 位倒 T 形电阻网络 D/A 转换器输出模拟量与输入数字量之间的一般关系式为

$$v_0 = -\frac{V_{REF}}{2^n} \cdot \frac{R_f}{R} \left[\sum_{i=0}^{n-1} (D_i \cdot 2^i) \right] \quad (10.1.7)$$

若将式中 $\frac{V_{REF}}{2^n} \cdot \frac{R_f}{R}$ 用 K 表示,方括号内的 n 位二进制数用 N_B 表示,则式(10.1.7)可改写为

$$v_0 = -KN_B \quad (10.1.8)$$

上式表明,对应每一个二进制数 N_B ,在图 10.1.4 电路的输出端都能得到与之成正比的模拟

电压。

由式(10.1.7)可见,要提高D/A转换器的转换精度,电路参数的选择要注意以下几点:

(1) 基准电压 V_{REF} 的精度和稳定性对D/A转换器的精度影响很大,在对精度要求较高的情况下,基准电压可采用带隙基准电压源;

(2) 倒T形电阻网络中 R 和 $2R$ 电阻比值的精度要高;

(3) 每个模拟开关的开关电压降要相等。为实现电流从高位到低位按2的整数倍递减,模拟开关的导通电阻也相应地按2的整数倍递增;

(4) 运放的零点漂移要小。

由于在倒T形电阻网络D/A转换器中,各支路电流是同时直接流入运算放大器的输入端,它们之间不存在传输上的时间差,所以,该电路不仅具有较高的转换速度,而且在动态过程中输出端可能出现的尖脉冲也大为减小。

例 10.1.1 在图10.1.4所示4位倒T形电阻网络D/A转换器中, $R_t=R$, $V_{REF}=10\text{ V}$,试分别求出当 $D_3D_2D_1D_0$ 分别为**1000**,**0100**,**0010**,**0001**时输出电压 v_o 的值。

解:根据式(10.1.6)可得

$$D_3D_2D_1D_0 = \mathbf{1000} \text{ 时}, v_o = -\frac{V_{REF}}{2^4} \times 2^3 = -\frac{V_{REF}}{2} = -\frac{10 \text{ V}}{2} = -5 \text{ V}$$

$$D_3D_2D_1D_0 = \mathbf{0100} \text{ 时}, v_o = -\frac{V_{REF}}{2^4} \times 2^2 = -\frac{V_{REF}}{4} = -\frac{10 \text{ V}}{4} = -2.5 \text{ V}$$

$$D_3D_2D_1D_0 = \mathbf{0010} \text{ 时}, v_o = -\frac{V_{REF}}{2^4} \times 2^1 = -\frac{V_{REF}}{8} = -\frac{10 \text{ V}}{8} = -1.25 \text{ V}$$

$$D_3D_2D_1D_0 = \mathbf{0001} \text{ 时}, v_o = -\frac{V_{REF}}{2^4} \times 2^0 = -\frac{V_{REF}}{16} = -\frac{10 \text{ V}}{16} = -0.625 \text{ V}$$

2. 集成D/A转换器

单片集成D/A转换器产品的种类繁多,性能指标各异。AD公司生产的AD7533是10位CMOS电流开关型D/A转换器,属于电流输出型D/A转换器,它的结构简单,通用性好。用该器件组成的D/A转换电路如图10.1.5所示,图中点画线部分为AD7533的内部电路。

AD7533芯片内只有倒T形电阻网络、CMOS电流开关和反馈电阻($R=10\text{ k}\Omega$)。用AD7533组成D/A转换器时,必须外接运算放大器,其反馈电阻可采用片内电阻($10\text{ k}\Omega$)或外加电阻。

图10.1.5中电子开关 S_i 的实际电路如图10.1.6所示。它是由9个MOS管组成的CMOS模拟开关电路。图中 $T_1 \sim T_3$ 组成电平转移电路,使输入信号能与TTL电平兼容。 T_4, T_5 及 T_6, T_7 组成两个反相器分别作为模拟开关管 T_8, T_9 的驱动电路, T_8, T_9 构成单刀双掷开关。

当 $D_i=1$ 时, T_1 输出低电平, T_4, T_5 反相器输出高电平,而 T_6, T_7 反相器输出低电平,从而使 T_8 截止, T_9 导通, $2R$ 电阻经 T_9 接至运放反相输入端,使电流流入运放。

当 $D_i=0$ 时, T_1 输出高电平, T_4, T_5 反相器输出的低电平使 T_6 截止, T_6, T_7 反相器输出的高电平使 T_8 导通,这样 $2R$ 电阻经 T_8 接地。CMOS模拟开关导通电阻较大,通过工艺设计可控制其大

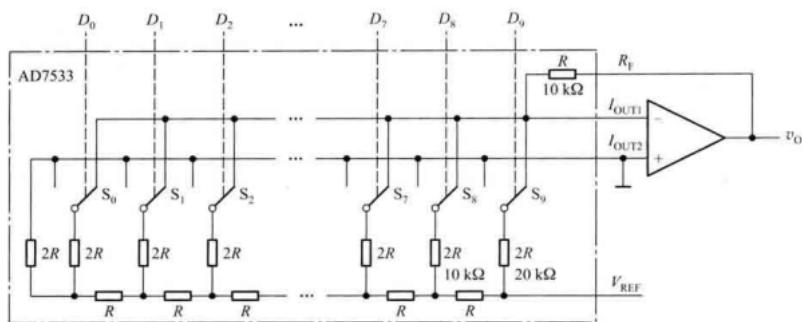


图 10.1.5 AD7533 内部电路

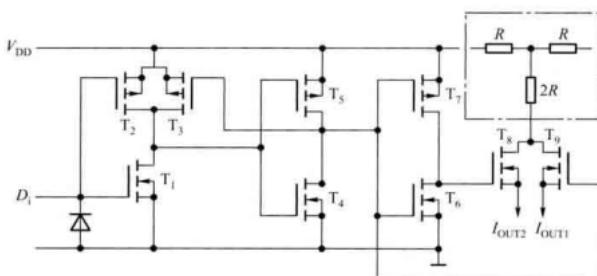


图 10.1.6 CMOS 模拟开关电路

小并计入电阻网络。该电路具有使用简便、功耗低、转换速度较快、温度系数小、通用性强等优点。

10.1.4 权电流型 D/A 转换器

1. 4 位权电流 D/A 转换器

由于实际的倒 T 形电阻网络 D/A 转换器中的模拟开关存在导通电阻和导通电压，这会引起求和电流的误差，从而影响 D/A 转换器的转换精度。如要提高 D/A 转换器的精度，可采用权电流型 D/A 转换器。4 位权电流 D/A 转换器的原理电路如图 10.1.7 所示。图中，用一组恒流源代替了图 10.1.4 中倒 T 形电阻网络，恒流源从高位到低位的电流大小依次为 $\frac{I}{2}, \frac{I}{4}, \frac{I}{8}, \frac{I}{16}$ 。

图 10.1.7 中，当 $D_i = 0$ 时，开关 S_i 接地； $D_i = 1$ 时，开关 S_i 与运放的反相端相联，相应的权电流流入求和电路。分析该电路可得

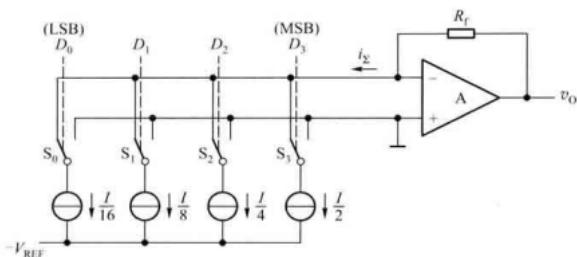


图 10.1.7 权电流 D/A 转换器的原理电路

$$\begin{aligned}
 v_0 &= -i_{\Sigma} = -R_i \left(\frac{I}{2} D_3 + \frac{I}{4} D_2 + \frac{I}{8} D_1 + \frac{I}{16} D_0 \right) \\
 &= \frac{I}{2^4} \cdot R_i (D_3 \cdot 2^3 + D_2 \cdot 2^2 + D_1 \cdot 2^1 + D_0 \cdot 2^0) \\
 &= \frac{I}{2^4} \cdot R_i \sum_{i=0}^3 D_i \cdot 2^i
 \end{aligned} \tag{10.1.9}$$

由于权电流 D/A 转换器中，各支路上的权电流的大小不受开关导通电阻和电压的影响，所以该电路的转换精度较高。

2. 实际的权电流 D/A 转换器

实际的权电流 D/A 转换器电路如图 10.1.8 所示，图中具有电流负反馈的 BJT 组成恒流源电路。

运放 A_2 、 R_1 、 T_1 、 R 和 $-V_{EE}$ 组成基准电流 I_{REF} 产生电路。 A_2 的输出端经 T_1 的 cb 结组成电压并联负反馈电路，以稳定其输出电压 (T_1 的基极电压)。基准电流 I_{REF} 由基准电压 V_{REF} 和电阻 R_1 确定。由于 T_1 和 T_3 具有相同的 V_{BE} ，而发射极回路的电阻相差一倍，所以它们的发射极电流也相差一倍，于是有

$$I_{REF} = \frac{V_{REF}}{R_1} = 2I_{E3} \tag{10.1.10}$$

图中 $T_3 \sim T_0$ 的基极电压相同，只要这些三极管的发射结压降 V_{BE} 相等，则它们的发射极 $e_3 \sim e_0$ 就是等电位，这样，从左到右流过 $2R$ 电阻上的电流就分别为 $\frac{I}{2}, \frac{I}{4}, \frac{I}{8}$ 和 $\frac{I}{16}$ 。

为了保证每个 BJT 的发射结电压相等，电路中 $T_3 \sim T_0$ 采用了多发射极 BJT，其发射极个数分别是 8, 4, 2, 1 (即发射极面积之比为 8 : 4 : 2 : 1)。这样，在各 BJT 的电流比值也为 8 : 4 : 2 : 1 的情况下， $T_3 \sim T_0$ 的射极电流的密度相等，它们的发射结电压 V_{BE} 也就相同，倒 T 形电阻网络每个 $2R$ 电阻的电流值就与图 10.1.7 完全相同。

于是可得输出电压为

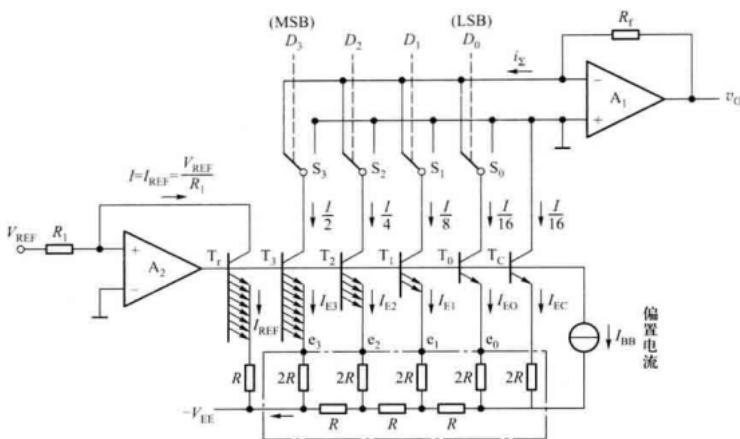


图 10.1.8 实际的权电流 D/A 转换器电路

$$v_O = i_{\Sigma} R_i = \frac{R_i V_{REF}}{2^n R_i} (D_3 \cdot 2^3 + D_2 \cdot 2^2 + D_1 \cdot 2^1 + D_0 \cdot 2^0)$$

可推得 n 位权电流 D/A 转换器的输出电压

$$v_O = \frac{V_{REF}}{R_i} \cdot \frac{R_i}{2^n} \sum_{i=0}^{n-1} D_i \cdot 2^i \quad (10.1.11)$$

式(10.1.11)表明,输出电压仅与基准电压 V_{REF} 和电阻 R_i 有关,而与 BJT、 R 、 $2R$ 电阻无关。由于电路对 BJT 参数及 R 、 $2R$ 取值的要求降低,这对电路集成十分有利。

在权电流 D/A 转换器中,一般都采用了高速电子开关,电路具有较高的转换速度。

10.1.5 权电容网络 D/A 转换器

在 MOS 集成电路中,电容不仅容易制作,而且可通过控制电容的尺寸,严格保持各电容容量之间的比例关系,制作出具有精确比例关系的不同电容。采用 MOS 工艺制造 D/A 转换器时,权电容网络 D/A 转换器是一种常采用的设计方案。

权电容网络 D/A 转换器是一种分压式 D/A 转换器,主要利用电容分压的原理工作。4 位权电容网络 D/A 转换的原理如图 10.1.9 所示,图中,模拟开关 $S_3 \sim S_0$ 分别由数字信号 $D_3 \sim D_0$ 控制,当 $D_i = 1$ 时, S_i 接参考电压 V_{REF} ;当 $D_i = 0$ 时, S_i 接地。图中 $C'_0 = 2^0 C_x$, 与各模拟开关相连接的电容容量依次按 2^i 递减,即 $C_3 = 2^3 C_x$, $C_2 = 2^2 C_x$, $C_1 = 2^1 C_x$, $C_0 = 2^0 C_x$ 。

每次转换的开始前,首先令所有开关(包括 $S_3 \sim S_0$ 和 S_B)接地,使所有电容充分放电完,然后断开 S_B 。在上述过程结束后,再将 4 位数据并行输入到 $D_3 \sim D_0$ 端。

例如, $D_3 D_2 D_1 D_0 = 0100$, 则只有 S_2 接 V_{REF} , S_3, S_1 和 S_0 接地, 图 10.1.9 的等效电路如图 10.1.10 所示。

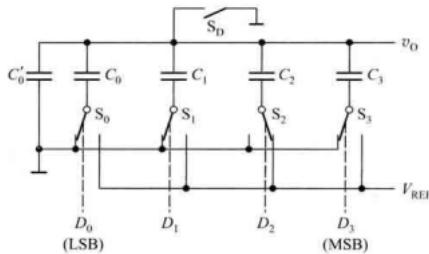


图 10.1.9 4 位权电容网络 D/A 转换器

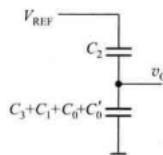


图 10.1.10 图 10.1.9 输入为 0100 时的等效电路

C_2 与 $(C_3 + C_1 + C_0 + C'_0)$ 构成一个电容分压器, V_{REF} 对串联电路快速充电的结果, 使输出电压按串联电容的容量分压, 输出电压为

$$\begin{aligned} v_0 &= \frac{D_2 C_2}{C_3 + C_2 + C_1 + C_0 + C'_0} V_{REF} \\ &= \frac{D_2 C_2}{C_1} V_{REF} \end{aligned} \quad (10.1.12)$$

式中, $C_1 = C_3 + C_2 + C_1 + C_0 + C'_0$

同理, 可写出输入为任意数字量时的输出模拟电压的一般表达式

$$\begin{aligned} v_0 &= \frac{D_3 C_3 + D_2 C_2 + D_1 C_1 + D_0 C_0}{C_1} V_{REF} \\ &= \frac{C_x (D_3 2^3 + D_2 2^2 + D_1 2^1 + D_0 2^0) V_{REF}}{2^4 C_1} \\ &= \frac{V_{REF}}{2^4} (D_3 2^3 + D_2 2^2 + D_1 2^1 + D_0 2^0) \end{aligned} \quad (10.1.13)$$

式(10.1.14)表明, 电路实现了数模转换。

n 位权电容网络 D/A 转换的一般化表达式

$$v_0 = \frac{V_{REF}}{2^n} \sum_{i=0}^{n-1} D_i \cdot 2^i \quad (10.1.14)$$

从上述推导可以看出, 权电容网络 D/A 转换器有如下特点。

(1) 输出电压精度与各电容的容量无关, 而只与它们的容量比值相关。电路对电容容量的大小要求不高。所以, 在 CMOS 集成电路集成技术的支持下, 容易制造出高精度的 D/A 转换器。

(2) 它的输出电压 v_0 稳态值不受模拟开关内阻和参考电源 V_{REF} 内阻的影响, 因而降低了对开关电路和参考电源的要求。

(3) 由于电容能隔离直流,因此权电容网络 D/A 转换器在静态时无功率损耗。

权电容网络 D/A 转换器在使用中也有一定的局限性,随转换分辨率的提高,权电容容量的比值将成倍增大,因此对电容尺寸精度的要求也成倍提高,高位权电容不仅会占用很大芯片面积,影响集成度,而且会增加电容充放电时间,降低电路的转换速度。

10.1.6 D/A 转换器的输出方式

在前面介绍的 D/A 转换器的讨论中,输入的是无符号的二进制数,即二进制数的每一位都是数值码。根据电路形式或参考电压的极性不同,输出电压或为 0 V 到正的满度值,或为 0 V 到负的满度值,D/A 转换器处于单极性输出方式。

采用单极性输出方式时,数字输入量采用自然二进制码,8 位 D/A 转换器单极性输出时,输入数字量与输出模拟量之间的关系如表 10.1.1 所示。

表 10.1.1 8 位 D/A 转换器在单极性输出时的输入/输出关系

数字量								模拟量
MSB				LSB				
1	1	1	1	1	1	1	1	$\pm V_{REF} \left(\frac{255}{256} \right)$
				⋮				⋮
1	0	0	0	0	0	0	1	$\pm V_{REF} \left(\frac{129}{256} \right)$
1	0	0	0	0	0	0	0	$\pm V_{REF} \left(\frac{128}{256} \right)$
0	1	1	1	1	1	1	1	$\pm V_{REF} \left(\frac{127}{256} \right)$
				⋮				⋮
0	0	0	0	0	0	0	1	$\pm V_{REF} \left(\frac{1}{256} \right)$
0	0	0	0	0	0	0	0	$\pm V_{REF} \left(\frac{0}{256} \right)$

实际上,D/A 转换器输入的都是有符号数,这就要求 D/A 转换器按符号的不同,输出为正、负极性的模拟电压,工作于双极性方式。采用双极性输出时常用的编码有 2 的补码、偏移二进制码和符号-数值码(符号位加数值码)等。表 10.1.2 以 8 位为例列出了 2 的补码、偏移二进制码与模拟量之间的对应关系。

表 10.1.2 常用双极性编码

	2 的补码								偏移二进制码								模拟量 v_o/V_{LSB}
	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	127
0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	126
	⋮									⋮					⋮		⋮
0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	-1
	⋮									⋮				⋮		⋮	⋮
1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	-127
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-128

* 表中 $V_{LSB} = V_{REF}/256$

2 的补码的最高位为符号位。符号位为 0 表示“+”号；为 1 表示“-”号，其他位为数值位。对于正数，其补码的数值部分为该二进制数的绝对值；对于负数，其补码的数值部分为该二进制数的绝对值每位取反、末位加 1，零的补码各位都是 0。

下面以输入为 8 位 2 的补码为例，说明双极性输出 D/A 转换器的工作原理。

将表 10.1.1 和表 10.1.2 中偏移二进制码相比较可发现，对于相同的编码，偏移二进制码 D/A 转换器与单极性 D/A 转换器的输出电压拉偏了 $\left(\frac{128}{256}V_{REF}\right)V$ 。如输入 **11111111**，偏移二进制码 D/A 转换器的输出电压为 $+\left(\frac{127}{256}V_{REF}\right)V$ ，而单极性二进制码输出电压为 $+\left(\frac{255}{256}V_{REF}\right)V$ ；又如，输入 **10000000** 时，偏移二进制码输出电压为 0 V，而在单极性二进制码的输出电压为 $-\left(\frac{128}{256}V_{REF}\right)V$ 。根据上述两种编码的输入、输出之间的对应关系，要得到输入为偏移二进制码的双极性 D/A 转换输出电压，只需将单极性 8 位 D/A 转换器的输出电压减去 $\left(\frac{128}{256}V_{REF}\right)V$ 即可。

比较表 10.1.2 中 2 的补码与偏移二进制码可以发现，偏移二进制码和 2 的补码的区别仅只是符号位相反。如要实现输入为 2 的补码 D/A 转换，只需将偏移二进制码 D/A 转换器的高位求反即可。

采用 2 的补码输入的 8 位双极性输出 D/A 转换电路，如图 10.1.11 所示。图中，最高位取反，由 D/A 转换器输出的模拟量 v_i 经 A_2 组成的第二个求和放大器减去 $\left(\frac{128}{256}V_{REF}\right)V$ 后得到极性

正确的输出电压 v_{0+}

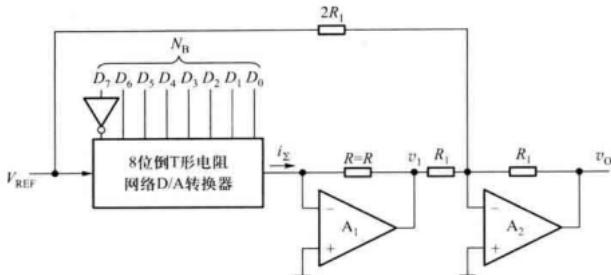


图 10.1.11 具有双极性输出的 D/A 转换器

分析电路可得

$$\begin{aligned}
 v_0 &= -v_1 - \frac{1}{2}V_{\text{REF}} \\
 &= \left(\frac{N_B V_{\text{REF}}}{2^8} + \frac{V_{\text{REF}}}{2} \right) - \frac{V_{\text{REF}}}{2} \\
 &= V_{\text{REF}} \cdot \frac{N_B}{256}
 \end{aligned} \tag{10.1.15}$$

电路输入 2 的补码 N_B 与 v_0 之间满足表 10.1.2 所示的对应关系。

10.1.7 D/A 转换器的主要技术指标

1. 分辨率

分辨率是 D/A 转换器对输入微小量变化敏感程度的表征。其定义为 D/A 转换器输出模拟电压可能被分离的等级数, n 位 D/A 转换器输出模拟量最多有 2^n 个不同值, 例如 8 位 D/A 转换器输出电压能被分离的等级数为 2^8 个。输入数字量位数愈多, 输出电压可分离的等级愈多, 即分辨率愈高。所以, 实际应用中, 往往用输入数字量的位数表示 D/A 转换器的分辨率。

另外, 分辨率也可以用 D/A 转换器最小输出电压(输入数码仅最低有效位为 1, 其余各位均为 0)与最大输出电压(输入数码全为 1)之比给出。

n 位 D/A 转换器的分辨率表示为

$$\text{分辨率} = \frac{V_{\text{LSB}}}{V_m} = \frac{1}{2^n - 1} \tag{10.1.16}$$

式中, V_{LSB} 为最小输出电压, V_m 为最大输出电压。当 V_m 一定, D/A 转换器的位数 n 越大, V_{LSB} 愈小, 即说明该转换器分辨能力愈高。它表示 D/A 转换器在理论上可以达到的精度。如 8 位 D/A 转换器的分辨率可以表示为

$$\frac{1}{2^n - 1} = \frac{1}{256 - 1} \approx 0.0039$$

例 10.1.2 已知 10 位 D/A 转换器满刻度输出电压 $V_m = 10$ V。

(1) 求输入最低位 D_0 对应的输出电压增量 V_{LSB} ;

(2) 如要求分辨的最小电压为 5 mV ($V_{LSB} = 5$ mV), 试问至少应选用多少位的 D/A 转换器。

解: (1) 根据式(10.1.16)可知, 分辨率 $= \frac{1}{2^n - 1} = \frac{V_{LSB}}{V_m}$

将 $n = 10$, $V_m = 10$ V 代入上式得

$$V_{LSB} = \frac{1}{2^{10} - 1} \times V_m = \frac{1}{2^{10} - 1} \times 10 \text{ V} = 0.01 \text{ V}$$

(2) 将 $V_{LSB} = 5$ mV, $V_m = 10$ V 代入式(10.1.16)中可得

$$\frac{1}{2^n - 1} = \frac{0.005}{10} \text{ V} \quad \text{由此得 } n \approx 11$$

由以上分析可知, 在题意给定条件下应选择 11 位 D/A 转换器。

2. 转换精度

由于 D/A 转换器中受到电路元件参数误差、基准电压不稳和运算放大器的零漂等因素的影响, D/A 转换器实际输出的模拟量与理想值之间存在误差。这些误差的最大值定义为转换精度。转换误差有比例系数误差、失调误差和非线性误差等。

(1) 比例系数误差

比例系数误差是指实际转换特性曲线的斜率与理想特性曲线斜率的偏差。如 n 位倒 T 形电阻网络 D/A 转换器, 当 V_{REF} 偏离标准值 ΔV_{REF} 时, 就会在输出端产生误差电压 Δv_o 。由式(10.1.7)可知

$$\Delta v_o = \frac{\Delta V_{REF}}{2^n} \cdot \frac{R_f}{R} \sum_{i=0}^{n-1} D_i \cdot 2^i \quad (10.1.17)$$

由 ΔV_{REF} 引起的误差属于比例系数误差。3 位 D/A 转换器的比例系数误差如图 10.1.12 所示。

(2) 失调误差

该误差为模拟量的实际起始数值与理想起始数值之差, 由运算放大器的零点漂移所引起, 它使输出电压的转移特性曲线发生平移, 3 位 D/A 转换器的失调误差如图 10.1.13 所示。

(3) 非线性误差

这是一种没有一定变化规律的误差, 一般用在满刻度范围内, 偏离理想的转移特性的最大值来表示。引起非线性误差的原因较多, 如电路中的各模拟开关存在不同的导通电压和导通电阻、电阻网络中电阻的误差等都会导致非线性误差。因此, 要获得高精度的 D/A 转换器, 不仅应选择位数较多的高分辨率的 D/A 转换器, 而且电路中还需要选用高稳定度的 V_{REF} 和低零漂的运算放大器等器件与之配合才能达到要求。

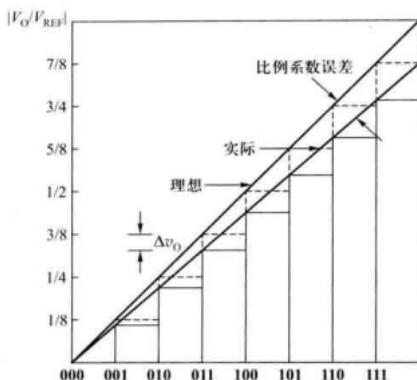


图 10.1.12 3 位 D/A 转换器的比例系数误差

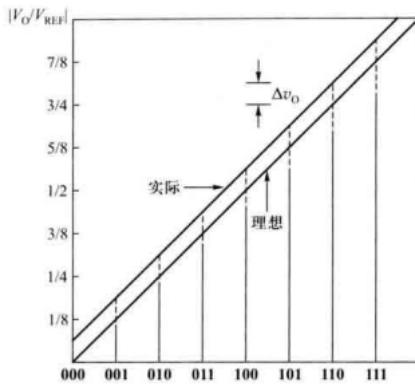


图 10.1.13 3 位 D/A 转换器的失调误差

3. 转换速度

当 D/A 转换器输入的数字量发生变化时, 输出的模拟量并不能立即达到所对应的量值, 它要延迟一段时间。通常我们用建立时间和转换速率两个参数来描述 D/A 转换器的转换速度。

(1) 建立时间 指输入数字量变化时, 输出电压达到规定误差范围所需的时间。一般用 D/A 转换器输入的数字量 N_B 从全 0 变为全 1 时, 输出电压达到规定的误差范围 ($\pm \text{LSB}/2$) 时所需时间表示。

(2) 转换速率 指大信号工作状态下, 模拟输出电压的最大变化率。通常以 $\text{V}/\mu\text{s}$ 为单位表示。该参数与运放的摆率 SR 类似。

4. 温度系数

这是指在输入不变的情况下, 输出模拟电压随温度变化产生的变化量。一般用满刻度输出条件下温度每升高 1℃, 输出电压变化的百分数作为温度系数。

10.1.8 D/A 转换器的应用

在实践中, D/A 转换器应用很广, 它不仅作为数字系统和模拟系统之间的接口电路(如作为微机系统的接口电路), 而且还可用于数字量对模拟信号进行处理。下面我们以数字式可编程增益放大电路和波形产生电路为例说明它的应用。

(1) 数字式可编程增益放大电路

数字式可编程增益控制电路如图 10.1.14 所示, AD7533 与运放接成普通的反相比例放大电路形式。电路中 AD7533 内部的反馈电阻 R 为反相比例放大电路的输入电阻, 而由数字量控制的倒 T 形电阻网络是它的反馈电阻。当输入数字量变化时, 倒 T 形电阻网络的等效电阻随之改变。这样, 在输入电阻一定的情况下, 随着电阻网络的等效电阻变化, 反相比例放大器的增益也

就随之改变。

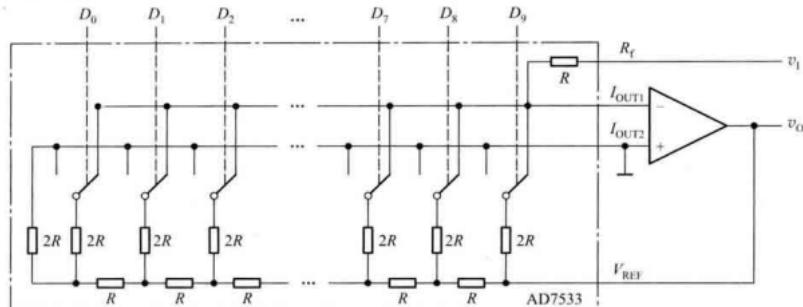


图 10.1.14 数字式可编程增益控制电路

根据运放虚地原理,可以得到

$$\frac{v_I}{R} = \frac{-v_O}{2^{10}R} (D_0 2^0 + D_1 2^1 + \cdots + D_9 2^9)$$

所以

$$A_v = \frac{v_O}{v_I} = \frac{-2^{10}}{D_0 2^0 + D_1 2^1 + \cdots + D_9 2^9}$$

如将 AD7533 芯片中的反馈电阻作为反相运放的反馈电阻,数控 AD7533 的倒 T 形电阻网络连接成运放的输入电阻,读者不难推断出电路为数字式可编程衰减器。

(2) 脉冲波产生电路

由 AD7533、运算放大器及 4 位同步二进制计数器 74LVC163(同步清零)组成的波形产生电路如图 10.1.15 (a) 所示。图中 74LVC163 采用反馈清零法,组成模 10 计数器,D/A 转换器的高

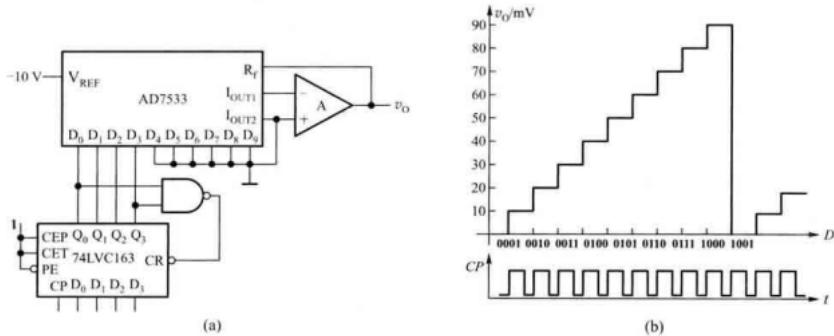


图 10.1.15 波形产生电路

(a) 电路 (b) 工作波形

位 $D_4 \sim D_9$ 均为 0, 低 4 位输入是计数器的输出。在 CP 作用下, $Q_3 Q_2 Q_1 Q_0$ 输出分别为 **0000 ~ 1001**。根据式(10.1.7)计算输出电压的值, 可画出 v_o 的波形如图 10.1.15(b) 所示, 输出波形是有 10 个阶梯的阶梯波。如改变计数器的模, 则改变波形的阶梯数; 如采用可逆计数器, 经滤波后, 在电路的输出端可得到三角波输出。

复习思考题

10.1.1 为使倒 T 形电阻网络 D/A 转换器有足够的精度, 在电路器件及参数选择上应注意些什么问题?

10.1.2 已知某 D/A 转换器满刻度输出电压为 10 V, 试问如要求分辨的最小电压是 1 mV, 应采用多少位的 D/A 转换器?

10.2 A/D 转换器

A/D 转换器要将时间和幅值都连续的模拟量, 转换为时间、幅值都离散的数字量, 一般要经过取样、保持和量化、编码几个过程, 下面分别予以讨论。

10.2.1 A/D 转换的一般工作过程

1. 取样与保持

取样电路将输入模拟量转换为在时间上离散的模拟量。取样过程示意图如图 10.2.1(a)、(b) 所示。图 10.2.1(a) 中, 取样信号 $S(t)$ 控制取样过程, $S(t)$ 高电平期间, 开关导通, 输出信号 $v_o(t)$ 等于输入信号 $v_i(t)$, 而在 $S(t)$ 的低电平期间, 开关关闭, 输出信号 $v_o(t) = 0$ 。电路工作波形如图 10.2.1(b) 所示。从图 10.2.1(b) 可见, 取样信号 $S(t)$ 的频率愈高, 所取信号经低通滤波器后, 愈能真实地复现输入信号。合理的取样频率由取样定理确定。

取样定理: 设取样信号 $S(t)$ 的频率为 f_s , 输入模拟信号 $v_i(t)$ 的最高频率分量的频率为 $f_{i\max}$, 则 f_s 与 $f_{i\max}$ 必须满足下面的关系

$$f_s \geq 2f_{i\max} \quad (10.2.1)$$

一般取 $f_s = 3 \sim 5f_{i\max}$ 。

将取样所得信号转换为数字信号往往需要一定的时间, 为了给后续的量化编码电路提供一个稳定值, 取样电路的输出还须保持一段时间。一般取样与保持过程都是同时完成的。取样-保持电路的原理图及输出波形分别如图 10.2.2(a)、(b) 所示。取样-保持电路由输入放大器 A_1 、输出放大器 A_2 , 保持电容 C_H 和开关驱动电路组成。电路中要求 $A_{v1} \cdot A_{v2} = 1$, 且 A_1 具有较高的输入阻抗, 以减小对输入信号源的影响。 A_2 选用有较高输入阻抗和低输出阻抗的运放, 这样不仅 C_H 上所存电荷不易泄漏, 而且电路还具有较高的带负载能力。

$t_0 \sim t_1$ 时段, 开关 S 闭合, 电路处于取样阶段, 电容器 C_H 充电, 由于 $A_{v1} \cdot A_{v2} = 1$, 因此 $v_o = v_i$;

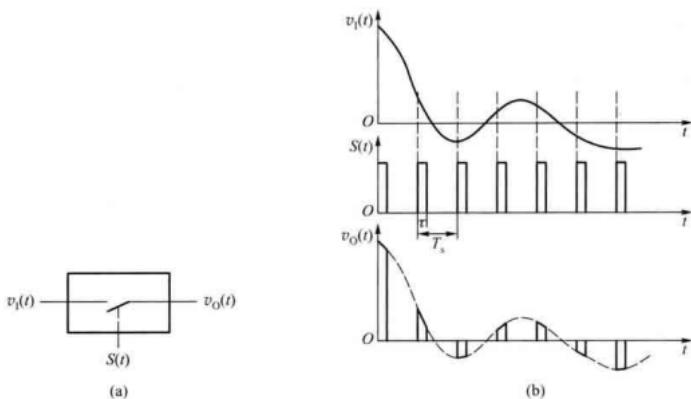


图 10.2.1 取样过程
(a) 取样电路示意图 (b) 各信号波形图

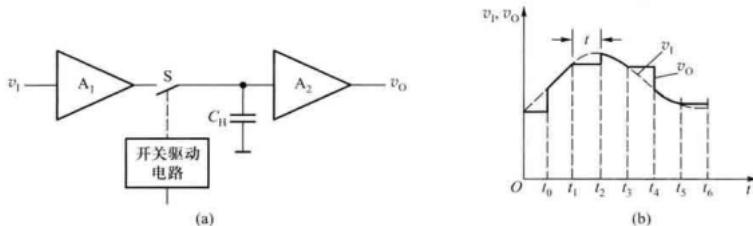


图 10.2.2 取样-保持电路
(a) 原理图 (b) 波形图

$t_1 \sim t_2$ 时段为保持阶段, 此期间 S 断开, 若 A_2 的输入阻抗足够大, 且 S 为较理想的开关, 可认为 C_H 几乎没有放电回路, 输出电压保持 v_o 不变。

2. 量化与编码

以上分析结果表明, 经取样和保持电路后的输出信号, 在数值上还是连续变化的模拟量, 至此, 只实现了对输入信号在时间上的离散。要转换成数字量, 还要实现数值上的离散, 将取样电压表示为一最小数量单位的整数倍, 这一转换过程称为量化。量化所取的最小数量单位称为量化单位, 用 Δ 表示。量化单位 Δ 是数字信号最低有效位为 1 时, 所对应的模拟量, 即 1LSB。由于取样电压是连续的, 它的值不一定都能被 Δ 整除, 所以, 在量化过程中, 不可避免地存在误差, 此误差称为量化误差, 用 ϵ 表示。 ϵ 属于原理误差, 它是无法消除的。A/D 转换器的位数越多,

1LSB 所对应的 Δ 值越小,量化误差的绝对值也越小。

量化的方法,一般有舍尾取整法和四舍五入法两种。舍尾取整的处理方法是:如输入电压 v_i 在两个相邻的量化值之间时,在 $(n-1)\Delta < v_i < n\Delta$ 时,取 v_i 的量化值为 $(n-1)\Delta$ 。四舍五入的处理方法是:当 v_i 的尾数不足 $\Delta/2$ 时,舍去尾数取整数;当 v_i 的尾数大于或等于 $\Delta/2$ 时,则其量化单位在原数上加一个 Δ 。

例如要将 0~1 V 的模拟电压转换为 3 位二进制码,取 $\Delta = (1/8)$ V,采用舍尾取整法,凡数值在 0 V ~ $(1/8)$ V 之间的模拟量,都当作 0Δ ,并用二进制码 000 表示;凡数值在 $(1/8)$ V ~ $(2/8)$ V 之间的模拟量,都当作 1Δ ,并用二进制码 001 表示。

而采用四舍五入量化的方式,则取量化单位 $\Delta = (2/15)$ V,凡数值在 0 V ~ $(1/15)$ V 之间的模拟电压都当作 0Δ ,并用二进制数 000 表示;而数值在 $(1/15)$ V ~ $(3/15)$ V 之间的模拟电压都当作 1Δ ,用二进制数 001 表示。不难看出,舍尾取整的量化方法,最大量化误差 $|\epsilon_{max}| = 1LSB$,而四舍五入量化方法 $|\epsilon_{max}| = 1SB/2$,由于后者量化误差小,为大多数 A/D 转换器所采用。

按上述两种方法划分量化电平的示意图分别如图 10.2.3(a)、(b)所示。

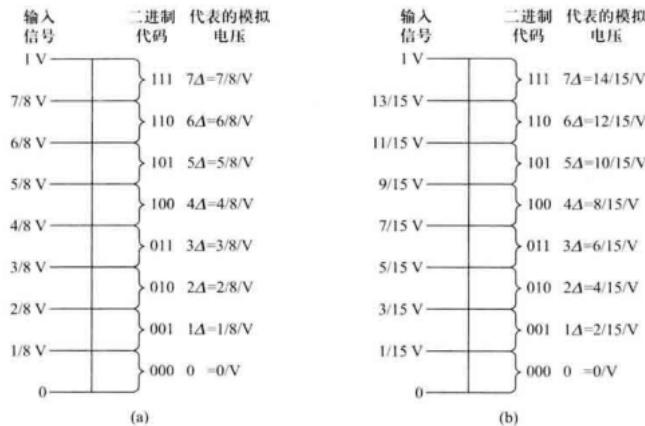


图 10.2.3 划分量化电平的两种方法

(a) 舍尾取整法 (b) 四舍五入法

将量化后的结果用二进制码或其他代码表示出来的过程称为编码。经编码输出的代码就是 A/D 转换器的转换结果。

A/D 转换器按其工作原理的不同分为直接 A/D 转换器和间接 A/D 转换器两种。直接 A/D 转换器将模拟信号直接转换为数字信号,这类 A/D 转换器具有较快的转换速度,典型电路有并行比较型 A/D 转换器,逐次比较型 A/D 转换器。而间接 A/D 转换器则是先将模拟信号转换成

某一中间量(时间或频率),然后再将中间量转换为数字量输出。此类 A/D 转换器的速度较慢,典型电路有双积分型 A/D 转换器、电压频率转换型 A/D 转换器等。

10.2.2 并行比较型 A/D 转换器

3 位并行比较型 A/D 转换器原理电路如图 10.2.4 所示。它由电阻分压器、电压比较器、寄存器及优先编码器组成。优先编码器输入信号 I_7 的优先级最高, I_1 最低。分压器将基准电压分为 $(1/15) V_{REF}$ 、 $(3/15) V_{REF}$ 、 \cdots 、 $(13/15) V_{REF}$ 不同电压值, 分别作为比较器 $C_1 \sim C_7$ 的参考电压。输入电压为 v_i 的大小决定各比较器的输出状态, 例如, 当 $0 \leq v_i < (1/15) V_{REF}$ 时, $C_7 \sim C_1$ 的输出状态都为 0; 当 $(3/15) V_{REF} \leq v_i < (5/15) V_{REF}$ 时, 比较器 C_6 和 C_7 的输出 $C_{06} = C_{07} = 1$, 其余各比较器的状态均为 0。比较器的输出状态由 D 触发器存储, 经优先编码器编码后得到数字量输出。

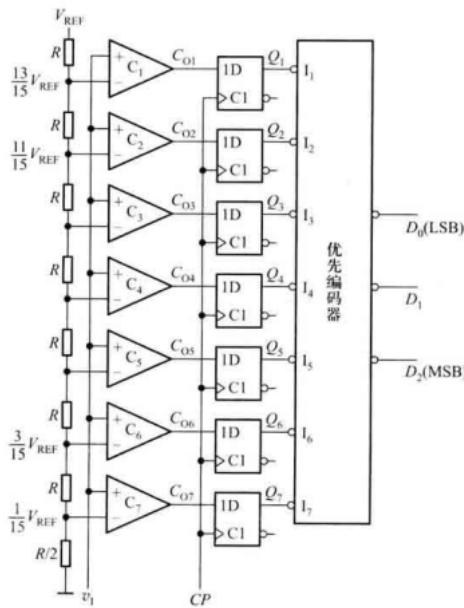


图 10.2.4 3 位并行 A/D 转换器

设 v_i 变化范围是 $0 \sim V_{REF}$, 输出 3 位数字量为 $D_2 D_1 D_0$, 3 位并行比较型 A/D 转换器的输入、输出关系如表 10.2.1 所示。

表 10.2.1 3 位并行 A/D 转换器输入与输出关系对照表

模拟输入	比较器输出状态							数字输出		
	C_{01}	C_{02}	C_{03}	C_{04}	C_{05}	C_{06}	C_{07}	D_2	D_1	D_0
$0 \leq v_i < V_{REF}/15$	0	0	0	0	0	0	0	0	0	0
$V_{REF}/15 \leq v_i < 3 V_{REF}/15$	0	0	0	0	0	0	1	0	0	1
$3 V_{REF}/15 \leq v_i < 5 V_{REF}/15$	0	0	0	0	0	1	1	0	1	0
$5 V_{REF}/15 \leq v_i < 7 V_{REF}/15$	0	0	0	0	1	1	1	0	1	1
$7 V_{REF}/15 \leq v_i < 9 V_{REF}/15$	0	0	0	1	1	1	1	1	0	0
$9 V_{REF}/15 \leq v_i < 11 V_{REF}/15$	0	0	1	1	1	1	1	1	0	1
$11 V_{REF}/15 \leq v_i < 13 V_{REF}/15$	0	1	1	1	1	1	1	1	1	0
$13 V_{REF}/15 \leq v_i < V_{REF}$	1	1	1	1	1	1	1	1	1	1

在并行 A/D 转换器中, 输入电压 v_i 同时加到所有比较器的输入端, 从 v_i 加入, 到稳定输出数字量, 所经历的时间为比较器、D 触发器和编码器延迟时间的总和。如不考虑各器件的延迟, 可认为输出数字量是与 v_i 输入时刻同时获得的。所以并行 A/D 转换器具有最短的转换时间。但也可以看到, 随着分辨率的提高, 元件数目几乎按几何级数增加, 如一个 n 位的转换器, 就要用到 $2^n - 1$ 个比较器和触发器。随着位数的增加, 电路复杂程度急剧增加。分辨率很高的并行 A/D 转换器, 对集成电路工艺指标要求是很高的。

例 10.2.1 在图 10.2.4 中, 已知 $V_{REF} = 6$ V, 输入模拟电压 $v_i = 3.4$ V, 试确定 3 位并行比较型 A/D 转换器的输出数码。

解: 根据并行比较型 A/D 转换器的工作原理可知, 输入到比较器 $C_1 \sim C_7$ 的参考电压分别为 $(1/15) V_{REF} \sim (13/15) V_{REF}$ 。将 $V_{REF} = 6$ V 代入, 求得图 10.2.1 中 $(7/15) V_{REF} = 2.8$ V, $(9/15) V_{REF} = 3.6$ V。

输入模拟电压 $v_i = 3.4$ V, 即 $(7/15) V_{REF} < v_i < (9/15) V_{REF}$, 对照表 10.2.1, 可知输出数码为 100。

10.2.3 逐次比较型 A/D 转换器

1. 转换原理

直接 A/D 转换器中, 逐次比较型 A/D 转换器是采用较多的一种。它的转换过程与用天平称物重相似。天平称重过程是, 从最重的砝码开始试放, 与被称物体进行比较, 若物体重于砝码, 则该砝码保留, 否则移去。再加上第二个次重砝码……照此进行, 一直加到最小一个砝码。将所有留下的砝码重量相加, 就得到物体重量。仿照这一思路, 逐次比较型 A/D 转换器, 就是将输入模拟信号与不同的权值电压做多次比较, 使转换所得的数字量在数值上逐次逼近输入模拟量。

8位逐次比较型A/D转换器框图如图10.2.5所示。它由控制逻辑电路、数据寄存器、移位寄存器、D/A转换器及电压比较器组成。

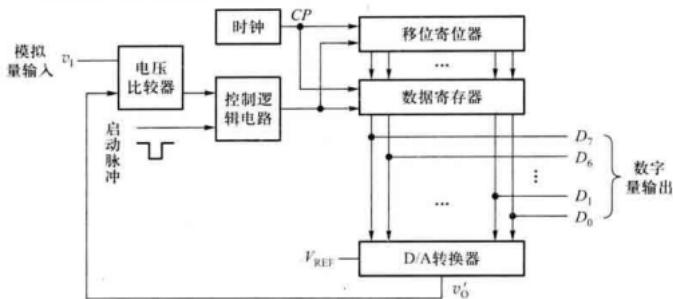


图 10.2.5 逐次比较型 A/D 转换器框图

电路启动后,第一个CP将移位寄存器置为**10000000**,该数字经数据寄存器送入D/A转换器。输入模拟电压 v_i 首先与**10000000**所对应的模拟电压 $V_{REF}/2$ 相比较[根据式(10.1.7)计算可得],如 $v_i \geq V_{REF}/2$,比较器输出为**1**,若 $v_i < V_{REF}/2$,则为**0**,此结果存于数据寄存器的 D_7 位;第二个CP使移位寄存器为**01000000**。如最高位已存**1**,数据寄存器将**11000000**送入D/A转换器,其输出电压 $v'_0 = \frac{3}{4}V_{REF}$, v_i 再与 $\frac{3}{4}V_{REF}$ 相比较,如 $v_i \geq \frac{3}{4}V_{REF}$ 则次高位(D_6)存**1**,否则 $D_6 = 0$;如最高位为**0**,数据寄存器将**01000000**送入D/A转换器,其输出电压 $v'_0 = V_{REF}/4$, v_i 与 v'_0 比较,如 $v_i \geq V_{REF}/4$,则数据寄存器的 D_6 位存**1**,否则存**0**……。以此类推,经逐次比较得到输出数字量。

设图10.2.5所示电路为8位逐次比较型A/D转换器,输入模拟量 $v_i = 6.84\text{ V}$,D/A转换器的基准电压 $V_{REF} = -10\text{ V}$ 。

根据逐次比较D/A转换器的工作原理,可画出在转换过程中CP、启动脉冲、 $D_7 \sim D_0$ 及D/A转换器输出电压 v'_0 的波形,如图10.2.6所示。

由图10.2.6可见,启动脉冲低电平到来后转换开始。第一个CP,数据寄存器将 $D_7 \sim D_0 = \mathbf{10000000}$ 送入D/A转换器,其输出电压 $v'_0 = 5\text{ V}$, v_i 与 v'_0 比较, $v_i > v'_0$, D_7 在下个CP到来时存**1**;第二个CP到来时,寄存器输出 $D_7 \sim D_0 = \mathbf{11000000}$, v'_0 为 7.5 V , v_i 再与 7.5 V 比较,因为 $v_i < 7.5\text{ V}$,所以 D_6 在下个CP到来时存**0**;输入第三个CP时, $D_7 \sim D_0 = \mathbf{10100000}$, $v'_0 = 6.25\text{ V}$; v_i 再与 v'_0 比较,……如此重复比较下去,经8个时钟周期,转换结束。由 v'_0 的波形可见,在转换过程中,输出数字量对应的模拟电压 v'_0 逐次逼近 v_i 值,最后的转换结果 $D_7 \sim D_0 = \mathbf{10101111}$ 。该数字量所对应的模拟电压为 6.8359375 V ,与实际输入的模拟电压 6.84 V 的相对误差仅为 0.06% 。

2. 转换电路

4位逐次比较型A/D转换器的逻辑电路如图10.2.7所示。图中,移位寄存器可进行并入/

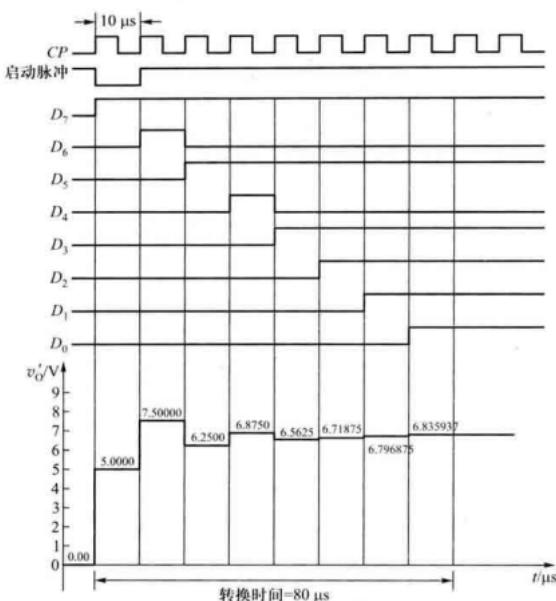


图 10.2.6 八位逐次比较型 A/D 转换器波形图

并出或串入/串出操作, F 为其并行置数端, 高电平有效, S 为高位串行输入端。5 个 D 触发器组成数据寄存器, 输出数字量为 $D_3 \sim D_0 = Q_4 \sim Q_1$ 。

在启动脉冲上升沿 $FF_0 \sim FF_4$ 被清零, Q_5 置 1, G_2 门开启, 时钟 CP 脉冲进入移位寄存器。

在第一个 CP 脉冲作用下, 移位寄存器被置数 $Q_4 Q_3 Q_2 Q_1 Q_0 = 01111$ 。 Q_5 的低电平使数据寄存器的最高位置 1, 即 $Q_4 Q_3 Q_2 Q_1 = 1000$ 。D/A 转换器将数字量 1000 转换为模拟电压 v'_0 送入比较器 C 与输入模拟电压 v_i 比较, 若输入电压 $v_i > v'_0$, 则比较器 C 输出 v_c 为 1, 否则为 0。比较结果送 $FF_4 \sim FF_1$ 的数据输入端 $D_4 \sim D_1$ 。

第二个 CP 脉冲到来后, 移位寄存器的 Q_5 变 1, 同时最高位向低位移动一位。 Q_5 由 0 变 1, 这个正跳变作为有效触发信号加到 FF_4 的 CP 端, 使第一次比较的结果保存于 Q_4 。此时, 由于其他触发器无脉冲正跳沿, 它们保持原状态不变。 Q_5 变 1 后建立了新的 D/A 转换器的数据, 输入电压再与此时的 v'_0 相比较, 比较结果在第三个时钟脉冲作用下存于 $Q_3 \dots$ 。如此进行, 直到 Q_5 由 1 变 0, 使 Q_5 由 1 变 0 后将 G_2 封锁, 转换完毕。于是电路的输出端 $D_3 D_2 D_1 D_0$ 得到与输入电压 v_i 成正比的数字量。

由以上分析可见, 逐次比较型 A/D 转换器完成一次转换所需时间与其位数 n 和时钟脉冲频

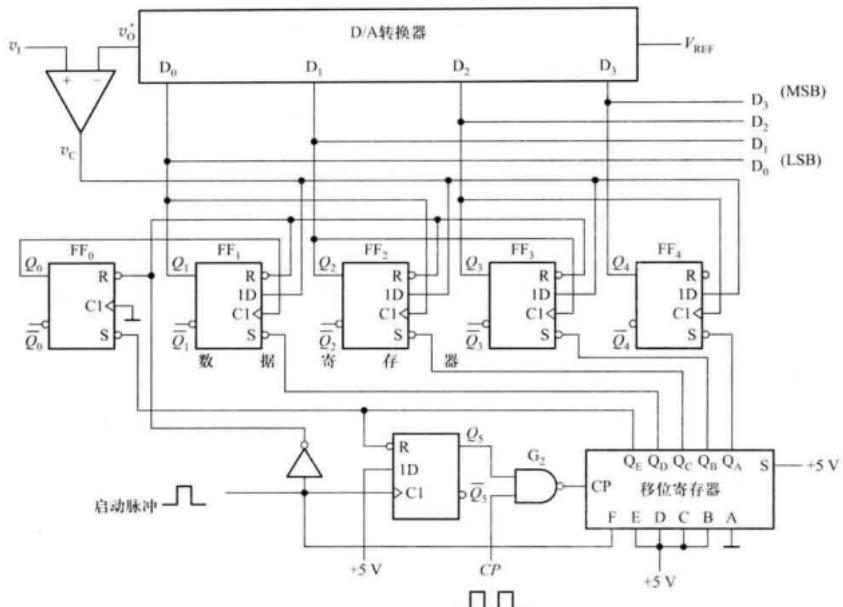


图 10.2.7 4 位逐次比较型 A/D 转换器的逻辑电路

率有关,位数越少,时钟频率越高,转换所需时间越短。这种 A/D 转换器具有转换速度快、精度高的特点。

10.2.4 双积分式 A/D 转换器

双积分式 A/D 转换器是一种间接 A/D 转换器。它采用对输入模拟电压和参考电压分别进行两次积分,将输入电压平均值变成与之成正比的时间间隔,然后利用时钟脉冲和计数器测出此时间间隔,进而在输出端得到与模拟量相应的数字量。由于该转换电路是对输入电压的平均值进行变换,所以它具有很强的抗工频干扰能力,在数字测量中得到广泛应用。

双积分式 A/D 转换器的原理电路如图 10.2.8 所示,它由积分器(由集成运放 A 组成),过零比较器(C),时钟脉冲控制门(G)和计数器(FF₀ ~ FF_{n-1})等几部分组成。

下面以输入正极性的直流电压 v_I ($v_I < V_{REF}$) 为例,说明电路的基本工作原理。

电路的工作过程分为以下几个阶段进行。

(1) 准备阶段

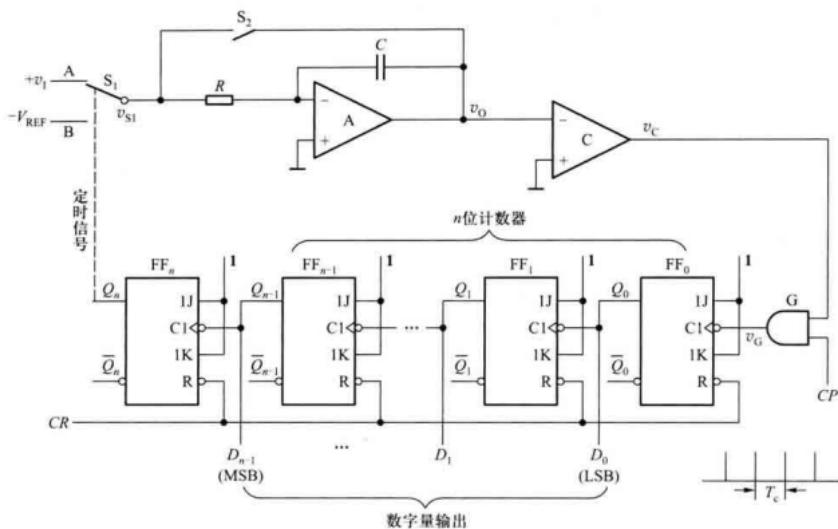


图 10.2.8 双积分 A/D 转换器

转换开始前, CR 信号将计数器清零, 开关 S_2 闭合, 使积分电容完全放电完毕。启动脉冲到来时转换开始, S_2 断开, 同时 S_1 与 v_i 接通。

(2) 第一次积分阶段

设 $t=0$ 时, 被测电压 v_i 加到积分器的输入端, 电路进入第一次积分阶段, 积分器的输出电压以与 v_i 大小成正比的斜率从 0 V 开始下降, 其波形如图 10.2.9(e) 的①段。积分器的输出电压 v_o 为

$$v_o = -\frac{1}{\tau} \int_0^t v_i dt \quad (10.2.2)$$

式中, $\tau = RC$ 。由于此时 $v_o < 0$, 比较器输出为高电平, 所以, 时钟控制门 G 被打开, 计数器在 CP 作用下从 0 开始计数。经 2^n 个时钟脉冲后, 计数器输出的进位脉冲使 $Q_n = 1$, 开关 S_1 由 A 点转接到 B 点, 第一次积分结束。第一次积分时间为

$$t = T_i = 2^n T_c \quad (10.2.3)$$

令 V_i 为输入电压在 T_i 时间间隔内的平均值, 则由式(10.2.2)可得第一次积分结束时积分器的输出电压为 v_p

$$v_p = -\frac{T_i}{\tau} V_i = -\frac{2^n T_c}{\tau} V_i \quad (10.2.4)$$

(3) 第二次积分阶段

当 $t = t_1$ 时, S_1 转接到 B 点将与 v_i 极性相反的基准电压 $-V_{REF}$ 加到积分器的输入端, 积分器进入第二次积分阶段, v_o 以 v_p 为初始值向反方向积分。当 $t = t_2$ 时, 积分器输出电压 $v_o = 0$, 比较器的输出电压 $v_c = 0$, 时钟脉冲控制门 G 被关闭, 计数停止。在第二次积分阶段结束后, 控制电路又使开关 S_2 闭合, 电容 C 放电, 电路为下一次转换做好准备。第二次积分阶段结束时 v_o 的表达式可写为

$$v_o(t_2) = v_p - \frac{1}{\tau} \int_{t_1}^{t_2} (-V_{REF}) dt = 0 \quad (10.2.5)$$

设 $T_2 = t_2 - t_1$, 于是有

$$\frac{V_{REF} T_2}{\tau} = \frac{2^n T_e}{\tau} V_i$$

设在此期间计数器所累计的时钟脉冲个数为 λ , 则

$$T_2 = \lambda T_e \quad (10.2.6)$$

$$T_2 = \frac{2^n T_e}{V_{REF}} V_i \quad (10.2.7)$$

可见, T_2 与 V_i 成正比, 也就是说电路已经将输入电压的平均值转换成了中间变量时间间隔 T_2 。

$$\lambda = \frac{T_2}{T_e} = \frac{2^n}{V_{REF}} V_i \quad (10.2.8)$$

式(10.2.8)表明, 在计数器中所计的数据 $\lambda (\lambda = Q_{n-1} \cdots Q_1 Q_0)$ 与在取样时间 T_1 内输入电压的平均值 V_i 成正比。只要 $V_i < V_{REF}$, T_2 期间计数器不会产生溢出问题, 转换器就能正常地将输入模拟电压转换为数字量, 并从计数器的输出读取转换的结果。如果取 $V_{REF} = 2^n V$, 则 $\lambda = V_i$, 计数器所计的数在数值上就等于被测电压。

电路的工作波形如图 10.2.9 所示。

由于双积分 A/D 转换器在 T_1 时间内取的是输入电压的平均值, 因此具有很强的抗工频干扰的能力。特别是对周期等于 T_1 整数倍的对称干扰信号(即在 T_1 期间平均值为零的干扰信号), 理论上有无穷大的抑制能力。在工业系统中经常碰到的是工频干扰近似于对称干扰, 若选定 T_1 是等于工频周期的倍数, 如 20 ms 或 40 ms 等, 即使工频干扰幅度大于被测直流信号, 仍能得到

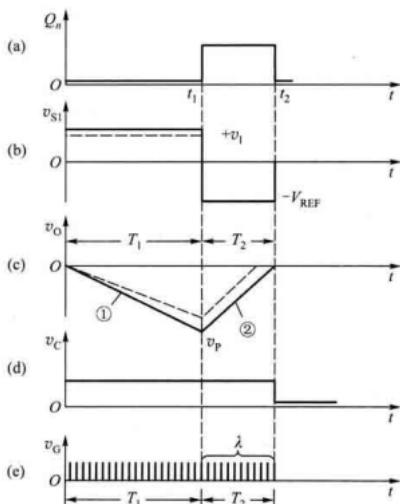


图 10.2.9 双积分 A/D 转换器的工作波形

良好的测量结果。另一方面,由于在转换过程中,前后两次积分所采用的同一积分器。因此,在两次积分期间(一般在几十至数百毫秒之间), R 、 C 和脉冲源等元器件参数的变化对转换精度的影响均可以忽略。

10.2.5 A/D 转换器的主要技术指标

A/D 转换器的主要技术指标有转换精度、转换速度等。选择 A/D 转换器时除考虑这两项技术指标外,还应注意满足其输入电压的范围、输出数字的编码、工作温度范围和电压稳定度等方面的要求。

1. 分辨率

A/D 转换器的分辨率用输出二进制(或十进制)数的位数表示。它说明 A/D 转换器对输入信号的分辨能力。从理论上讲, n 位输出的 A/D 转换器能区分 2^n 个输入模拟电压信号的不同等级,能区分输入电压的最小值为满量程输入的 $1/2^n$ 。在最大输入电压一定时,输出位数越多,量化单位越小,分辨率越高。例如 A/D 转换器输出为 8 位二进制数,输入信号最大值为 5 V,那么这个转换器应能区分出输入信号的最小电压为 19.53 mV。

2. 转换时间

转换时间是指 A/D 转换器从转换控制信号到来开始,到输出端得到稳定的数字信号所经过的时间。A/D 转换器的转换时间与转换电路的类型有关。不同类型的转换器转换速度相差甚远。其中并行比较 A/D 转换器的转换速度最高,逐次比较型 A/D 转换器次之,间接 A/D 转换器的速度最慢。在实际应用中应从系统数据总的位数、精度要求、输入模拟信号的范围及输入信号极性等方面综合考虑 A/D 转换器的选用。

10.2.6 集成 A/D 转换器及其应用

1. 集成 A/D 转换器 ADC0809

在单片集成 A/D 转换器中,逐次比较型使用较多,下面我们以 ADC0809 介绍集成 A/D 转换器及其应用。ADC0809 是 AD 公司采用 CMOS 工艺生产的一种 8 位逐次比较型 A/D 转换器。其内部结构框图如图 10.2.10 所示。ADC0809 的转换时间为 100 μ s,输入电压范围为 0 ~ 5 V,片内有 8 通道模拟开关,可接入 8 个模拟量输入。由于芯片内有输出数据寄存器,输出的数字量可直接与在计算机 CPU^①的数据总线相接,而无需附加接口电路。

$IN_0 \sim IN_7$:8 路模拟信号输入端;

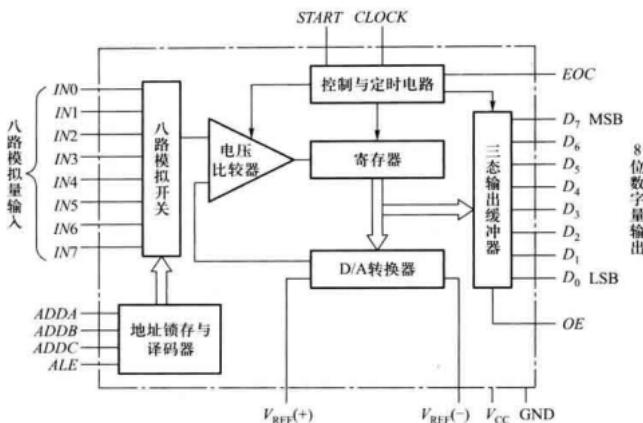
$D_7 \cdots D_0$:8 位数字信号输出端;

CLOCK:时钟信号输入端;

ADDA、ADDB、ADD_C:地址码输入端,不同的地址码选择不同通道的模拟量输入;

ALE:地址码锁存输入端,当输入地址码稳定后,ALE 的上升沿将地址信号锁存于地址锁存器内;

① CPU 系 Central Processing Unit 的缩写。



10.2.10 ADC0809 内部结构框图

$V_{REF}(+)$ 、 $V_{REF}(-)$: 分别为参考电压的正、负输入端。一般情况下 $V_{REF}(+)$ 接 V_{CC} , $V_{REF}(-)$ 接 GND;

START: 启动信号输入端。该信号的上升沿到来时片内寄存器被复位,在其下降沿开始 A/D 转换;

EOC: 转换结束信号输出端。当 A/D 转换结束时 EOC 变为高电平,并将转换结果送入三态输出缓冲器, EOC 可以作为向 CPU 发出的中断请求信号。

OE: 输出允许控制输入端。当 $OE=1$ 时,三态输出缓冲器的数据送到数据总线。

在使用时应注意以下几点:

(1) 转换时序

ADC0809 控制信号的时序图如图 10.2.11 所示,该图描述了各信号之间时序关系。

ALE 信号在地址信号有效后加入,在其上升沿将地址信号锁存于地址锁存与译码器,选择输入通道,在通道信号有效后经 t_i 时间,在 START 的下降沿电路开始 A/D 转换。经 t_c 时间转换结束,EOC 的高电平将结果存于三态输出缓冲器,当 OE 的高电平到来后的 t_c 时间,数字信号送出。

(2) 参考电压的调节

在使用 A/D 转换器时,为保证其转换精度,要求输入电压满量程使用。如输入电压动态范围较小则可调节参考电压 V_{REF} 以保证小信号输入时 ADC0809 芯片 8 位的转换精度。

(3) 接地

模数、数模转换电路中要特别注意地线的正确连接,否则会产生严重的干扰,影响转换结果。

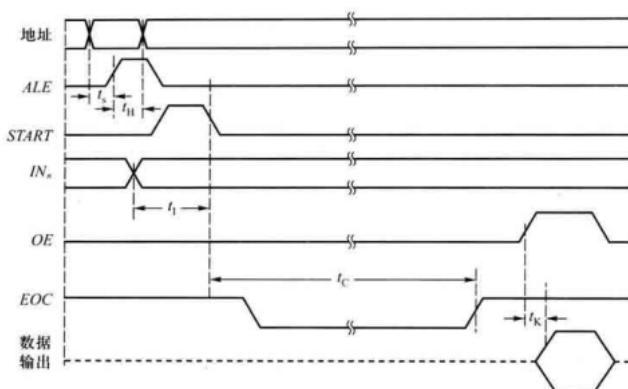


图 10.2.11 ADC0809 控制信号的定时图

的准确性。A/D、D/A 及取样保持芯片上都提供了独立的模拟地 (AGND) 和数字地 (DGND) 的引脚。在线路设计中, 必须将所有器件的模拟地和数字地分别相连, 然后将模拟地与数字地仅在一点上相连接。地线的正确连接方法如图 10.2.12 所示。

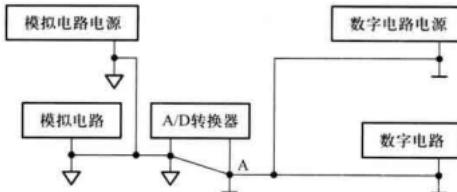


图 10.2.12 正确的地线连接

2. ADC0809 的典型应用

下面以数据采集系统为例介绍 ADC0809 的典型应用。

在现代过程控制及各种智能仪器和仪表中, 为采集被控 (被测) 对象数据达到由计算机进行实时控制、检测的目的, 常用微处理器和 A/D 转换器组成数据采集系统。单通道微机化数据采集系统的示意图如图 10.2.13 所示。

系统由微处理器、存储器和 A/D 转换器组成, 系统信号采用总线传送方式, 它们之间的信号通过数据总线 (DBUS) 和控制总线 (CBUS) 连接。

现以程序查询方式为例说明 ADC0809 在数据采集系统中的应用。采集数据时, 微处理器先执行一条传送指令, 在该指令执行过程中微处理器在控制总线上产生写信号, 其低电平信号启动

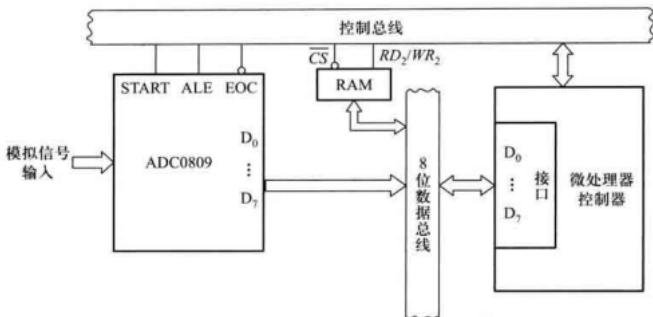


图 10.2.13 单通道微机化数据采集系统示意图

A/D 转换器工作, ADC0809 经 $100 \mu s$ 后将输入模拟信号转换为数字信号存于输出锁存器, EOC 信号经反相器产生中断请求信号 INTR, 通知微处理器取数。当微处理器响应中断请求转入数据采集子程序后, 立即执行输入指令, 指令产生读信号给 ADC0809, 将数据取出并存入存储器中。整个数据采集过程中, 由微处理器有序地执行若干指令完成。

复习思考题

10.2.1 实现模数转换一般要经过哪四个过程? 按工作原理不同分类, A/D 转换器可分为哪两种?

10.2.2 在图 10.2.4 所示并行比较 A/D 转换电路中, 若输入电压 v_i 为负电压, 试问电路能否正常进行 A/D 转换? 为什么? 如不能正常工作需要如何改进电路?

10.2.3 已知在图 10.2.4 所示并行 A/D 转换器中 $V_{REF} = 10 V$, $v_i = 9 V$, 试问输出数字量 $D_2 D_1 D_0 = ?$

10.2.4 在图 10.2.7 所示逐次比较型 A/D 转换器中, 完成一次 A/D 转换所需时间为多少? 转换时间与哪些因素有关?

10.2.5 试问双积分 A/D 转换器输出数字量与下述哪些参数有关? a) 积分时间常数; b) 时钟脉冲频率; c) 输入信号电压; d) 计数器位数; e) 运放的零漂; f) $|V_{REF}|$ 。

10.2.6 比较并行比较型 A/D 转换器、逐次比较型 A/D 转换器、双积分式 A/D 转换器的优、缺点, 试问应如何根据实际系统要求合理选用?

小结

- 倒 T 形电阻网络 D/A 转换器具有如下特点: 电阻网络仅有 R 和 $2R$ 两种阻值; 各 $2R$ 支路电流 I_i 与相应 D_i 数码状态无关, 是一定值; 由于支路电流流向运放反相端时不存在传输时间, 因而具有较高的转换速度。

● 不同结构的 A/D 转换器有各自的特点，在要求转换速度高的场合，可选用并行 A/D 转换器；在要求精度高的情况，可以采用双积分 A/D 转换器，当然也可选用高分辨率的其他形式 A/D 转换器，但成本会增加。由于逐次比较型 A/D 转换器在一定程度上兼顾了以上两种转换器的优点，因此得到普遍应用。

● A/D 转换器和 D/A 转换器的主要技术参数是转换精度和转换速度。目前，A/D 与 D/A 转换器正向着高速度、高分辨率和易于与微型计算机接口方向发展。

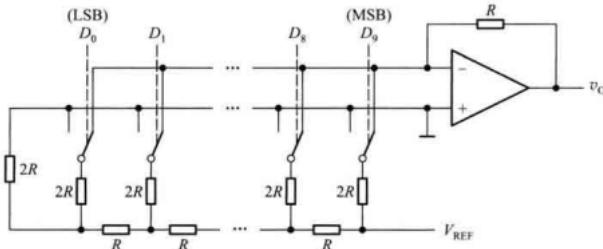
习 题

10.1 D/A 转换器

10.1.1 10 位倒 T 形电阻网络 D/A 转换器如图题 10.1.1 所示。

(1) 试求输出电压的取值范围；

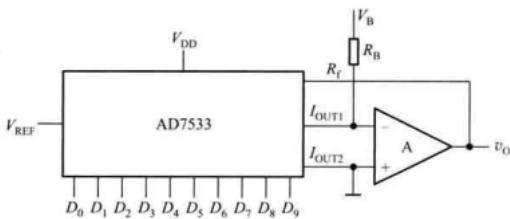
(2) 若要求电路输入数字量为 200H 时输出电压 $V_o = 5 \text{ V}$ ，试问 V_{REF} 应取何值？



图题 10.1.1

10.1.2 在图 10.1.7 中所示 4 位权电流 D/A 转换器中，已知 $V_{REF} = 6 \text{ V}$, $R_i = 48 \text{ k}\Omega$ ，当输入 $D_3 D_2 D_1 D_0 = 1100$ 时， $v_o = 1.5 \text{ V}$ ，试确定 R_i 的值。

10.1.3 在图 10.1.4 所示的倒 T 形电阻网络 D/A 转换器中，设 $R_f = R$ ，外接参考电压 $V_{REF} = -10 \text{ V}$ ，为保证



图题 10.1.4

V_{REF} 偏离标准值所引起的误差小于 $LSB/2$, 试计算 V_{REF} 的相对稳定性应取多少?

10.1.4 由 AD7533 组成双极性输出 D/A 转换器如图题 10.1.4 所示。

(1) 根据电路写出输出电压 v_o 的表达式;

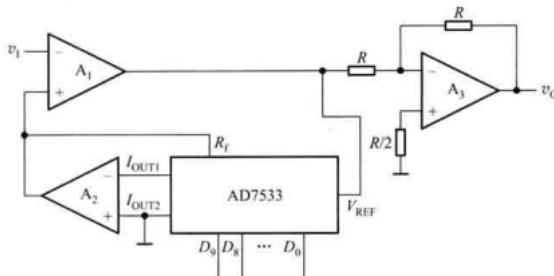
(2) 试问实现输入为 2 的补码时的双极性输出电路中, V_B 、 R_B 、 V_{REF} 和片内的 R 应满足什么关系?

10.1.5 可编程放大器(数控可变增益放大器)电路如图题 10.1.5 所示。

(1) 推导电路电压放大倍数 $A_v = v_o/v_i$ 的表达式;

(2) 当输入编码为(001H)和(3FF)时, 电压放大倍数 A_v 分别为多少?

(3) 试问当输入编码为(000H)时, 运放 A_1 处于什么状态?



图题 10.1.5

10.1.6 试用 D/A 转换器 AD7533 和计数器 74161 组成如图题 10.1.6 所示的阶梯波形发生器, 要求画出完整的逻辑图。

10.2 A/D 转换器

10.2.1 在图 10.2.4 所示平行比较型 A/D 转换器中, $V_{REF} = 7$ V, 试问电路的最小量化单位 Δ 等于多少? 当 $v_i = 2.4$ V 时, 输出数字量 $D_7D_6D_5 = ?$

10.2.2 在图 10.2.5 所示逐次比较 A/D 转换器中, 若 $n = 10$, 已知时钟频率为 1 MHz, 则完成一次转换所需时间是多少? 如要求数字量小于 100 μ s, 问时钟频率应选多大?

10.2.3 在图 10.2.7 逐次比较 A/D 转换器中, 设 $V_{REF} = 10$ V, $v_i = 8.26$ V, 试画出在时钟脉冲作用下, v_o 的波形并写出转换结果。

10.2.4 计数型 A/D 转换器如图题 10.2.4 所示。试分析其工作原理。

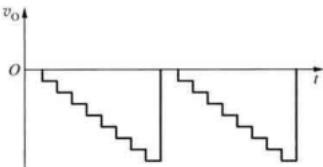
10.2.5 某双积分 A/D 转换器中, 计数器为十进制计数器, 其最大计数容量为 $(3000)_B$ 。已知计数时钟频率 $f_{CP} = 30$ kHz, 积分器中 $R = 100$ k Ω , $C = 1 \mu$ F, 输入电压 v_i 的变化范围为 0~5 V, 试求:

(1) 第一次积分时间 T_1 ;

(2) 求积分器的最大输出电压 $|V_{max}|$;

(3) 当 $V_{REF} = 10$ V, 第二次积分计数器计数值 $\lambda = (2500)_{10}$ 时, 输入电压的平均值 v_i 为多少?

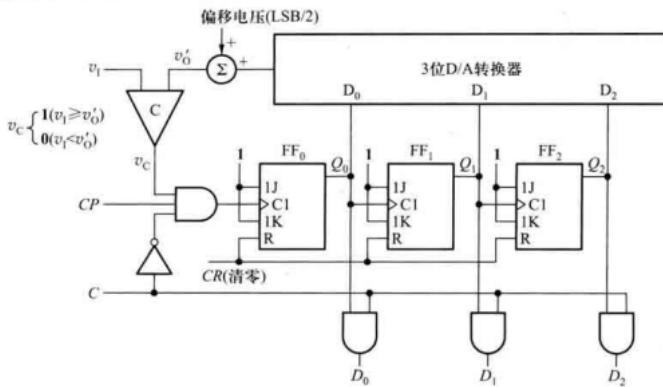
10.2.6 在图 10.2.8 所示双积分 A/D 转换器中, 设时钟脉冲频率为 f_{CP} , 其分辨率为 n 位, 写出最低的取样



图题 10.1.6

频率表达式。

10.2.7 在双积分 A/D 转换器中输入电压 v_i 和参考电压 V_{REF} 在极性和数值上应满足什么关系？如 $|v_i| > |V_{REF}|$ ，电路能完成模数转换吗？为什么？



图题 10.2.4

10.2.8 在应用 A/D 转换过程中应注意哪些主要问题？如某人用满度值为 10 V 的 8 位 A/D 转换器对输入信号幅值为 0.5 V 的电压进行模数转换，你认为这样使用正确吗？为什么？

* 11 >>>

数字系统设计基础

引
言

前面几章讨论了组合逻辑电路和时序逻辑电路的分析和设计方法。这些分析和设计方法是建立在真值表、卡诺图和状态表的基础上，针对功能相对单一的基本逻辑单元电路进行的。如果将这些基本逻辑部件组成规模更大、功能更复杂的数字电路时，采用经典的逻辑电路设计方法进行描述和设计，则设计过程比较复杂，需要采用新的方法，即分层次设计。

本章首先介绍数字系统的基本概念，数字系统的设计方法，以及数字系统的实现方法。然后介绍常用的设计工具，即算法状态机 ASM 图，并列举简单的实例说明数字系统的设计过程和实现方法。通过讲述这些引导性的内容以期初学者获得数字系统设计的基础知识和设计技巧。

11.1 数字系统概述

11.1.1 数字系统的组成

前面几章介绍的数据选择器、比较器、加法器、计数器等电路，都是只能实现某种单一的特定功能，因此称为功能部件级电路。由若干这样的数字电路和逻辑部件构成的，按一定顺序处理和传输数字信号的设备，称为数字系统。电子计算机、数字照相机、数字电视等就是常见的数字系统。当数字系统比较复杂，内部状态数目很大时，若采用经典的状态图、状态表等方法进行设计时，设计过程过于复杂，不易于完成设计任务。通常采用层次化设计方法，即先将一个规模很大的数字系统划分为几个子系统，再将每个子系统分解为几个模块，每个模块再分解为几个功能块，最终将每个功能块用各种门和触发器等组成。

数字系统从结构上可以划分为数据处理单元和控制单元两部分，如图 11.1.1 所示。因此，数字系统中的二进制信息也划分成数据信息和控制信息两大类。



图 11.1.1 数字系统框图

数据处理单元接受控制单元发来的控制信号,对输入的数据进行算术运算、逻辑运算、移位操作等处理,然后输出数据,并将处理过程中产生的状态信息反馈到控制单元,数据处理单元也称为数据通路(Datapath)。

控制单元根据外部输入信号及数据处理单元提供的状态信息,决定下一步要完成的操作,并向数据处理单元发出控制信号以控制其完成该操作。通常以是否有控制单元作为区别功能部件和数字系统的标志,凡是包含控制单元且能按顺序进行操作的系统,不论规模大小,一律称为数字系统,否则只能算是一个子系统部件,不能叫做一个独立的数字系统。例如,大容量存储器尽管电路规模很大,但也不能称为数字系统。

11.1.2 数字系统的设计方法

数字系统的设计方法可分为两大类,即自下而上的设计方法和自上而下的设计方法。现分别介绍如下:

1. 自下而上的设计方法

数字系统自下而上的设计是一种试探法。设计者根据自己的经验将规模大、功能复杂的数字系统按逻辑功能划分成若干子模块,一直分到这些子模块可以用经典的方法和标准的逻辑功能部件进行设计,最后将整个系统安装、调试达到设计要求。

自下而上的设计方法通常按照下列步骤进行:

(1) 分析设计要求,确定系统总体方案

设计任务常用文字描述整个数字系统的逻辑要求,当系统较大或逻辑关系较复杂时,从理解文字描述的含义到抽象出逻辑表述并非容易之事。因此,必须仔细全面分析系统的任务,以防出现疏漏和偏差。

(2) 划分逻辑单元,确定系统的初始结构,建立总体逻辑图

逻辑单元的划分可以采用由粗到细的方法,先将系统分为处理单元和控制单元,明确处理任务或控制功能,如果某一部分的规模仍比较大,则需进一步划分。划分后的各部分逻辑功能清楚明了,规模大小合适。

(3) 选择逻辑功能部件,构成逻辑电路

将划分的逻辑单元进一步分解成若干相对独立的模块,以便选用通用集成电路芯片实现。

(4) 将各个功能部件组装成数字系统

连接各个模块,绘制总体电路图,此时要考虑各模块之间的配合问题,例如时序协调、负载匹配电路启动等。

(5) 制作 PCB,进行物理实现

物理实现是指用实际的器件实现数字系统,并对实际电路进行测试看是否满足要求。

自下而上设计方法具有如下特点:

(1) 这种设计方法没有明显的规律可循,主要依靠设计者的实践经验和熟练的设计技巧,用逐步试探的方法最后设计出一个完整的数字系统。

(2) 系统的各项性能指标只有在系统构成后才能分析测试。如果系统设计存在比较大的问题,也有可能要重新设计,使得设计周期加长、资源浪费也较大。

尽管自下而上的硬件设计方法,在实际运用的一定范围内解决了不少问题,而且目前个别设计还在使用这种方法。但随着计算机技术及电子技术的发展,这种设计方法日益显得陈旧,取而代之的是自上而下的设计方法。

2. 自上而下的设计方法

自上而下的设计方法是针对数字系统层次化结构的特点,将系统的设计分层次、分模块进行。通常将整个系统从逻辑上划分成控制器和处理器两大部分。如果控制器和处理器仍比较复杂,可以在控制器和处理器内部多重地进行逻辑划分,分解成几个子模块进行逻辑设计,给出实现系统的硬件和软件描述,最后得到所要求的数字系统。

自上而下的设计方法一般要遵循下列几个步骤:

(1) 明确所要设计系统的功能,进行逻辑抽象

设计题目通常是比较简单的文字叙述,没有细节说明,设计者必须对题目消化、理解,逐步明确并抽象出系统要完成的逻辑功能。

(2) 确定实现系统功能的算法,画出系统方框图

算法设计实质是将系统要实现的复杂功能进行分解,分成若干子功能模块,并确定各功能模块的操作顺序和相互联系,画出系统的方框图。确定数字系统的算法是设计的关键步骤,也是最困难、最具有创造性的一步。但由于数字系统逻辑功能的多样性,至今仍没有一种通用合理的方法导出算法。

(3) 设计数据处理单元

首先明确数据处理单元的基本运算和操作,它们可以是算术和逻辑运算、数据的存储、变换和传送等。选用通用集成电路芯片实现其功能,或用硬件描述语言描述其逻辑功能,然后用可编程逻辑器件实现。

(4) 设计控制单元

根据数据处理单元进行的操作及操作顺序,确定控制单元的逻辑功能。在绝大多数数字系统中,控制单元是同步时序电路,因此,用设计同步时序电路的方法设计控制单元电路。

(5) 仿真验证

将整个系统连在一起,通过 EDA 软件在计算机上进行仿真验证,当验证所设计的系统正确

后,再进行实际电路的安装与测试。

由于数字系统自上而下的分层次设计方法,使得在不同阶段对电路或系统进行描述的方法不同。本章重点介绍控制单元的设计工具:算法状态机。

11.1.3 数字系统的实现

随着集成电路技术的发展和计算机的应用,数字系统的实现方法也经历了由分立元件、小规模、中规模到大规模、超大规模,直至今天的专用集成电路(ASIC)。数字系统的实现大致有以下几种方法:

- (1) 采用通用的集成逻辑器件组成。
- (2) 采用单片微处理器作为核心实现。
- (3) 采用可编程逻辑器件 PLD。
- (4) 设计功能完整的数字系统芯片。

第一种是传统的方法,曾经得到广泛应用,目前在设计较为简单电路时还在使用。第二种方法所用器件少,使用灵活,也得到广泛应用,但工作速度相对后面两种方法低些。第三种方法设计的系统体积小、功耗低、可靠性高、易于进行修改等,成为当今实现数字系统设计的首选方案。现在的 ASIC 芯片规模已经达到几百万个元件。可编程逻辑器件 FPGA 或 CPLD 属于 ASIC 电路的一类。一个复杂的数字系统只要一片或几片 ASIC 即可实现。制作 ASIC 的方法大体可分为两种,一种是掩膜方法,即由半导体厂家制造;另一种是现场可编程方法,用户通过计算机和开发工具,将所设计的电路或系统“编程”到芯片上。第四种方法是将一个完整的系统集成在一个芯片上,称为芯片系统(SOC)。微电子制造工艺的进步和 EDA 软件技术的发展,使得数字系统的设计效率和复杂程度都在不断提高。这里主要介绍采用通用集成逻辑器件和可编程逻辑器件 PLD 实现数字系统。

通常可编程逻辑器件的集成软件开发系统,支持电路图输入方式、硬件描述语言输入或两种输入的混合方式。所谓硬件描述语言,就是利用某一种语言描述硬件电路的功能、信号连接关系及定时关系。它能比电路原理图更有效地表示硬件电路的特性。硬件描述语言在硬件设计领域的作用与 C 或 C++ 在软件设计领域的作用类似。软件语言在某一时刻只需执行一条语句,而硬件描述语言可能同时要执行几条语句,因为实际系统中许多操作是并行的,这是它与软件语言的最大区别之一。

硬件描述语言有很多种类,较早使用 ABEL,现在比较流行使用 Verilog HDL 和 VHDL。比较而言,ABEL 是一种用来描述相对简单的数字系统,而 Verilog HDL 和 VHDL 则是用来描述更复杂的数字系统,这里主要介绍 Verilog HDL 语言的应用。

复习思考题

11.1.1 从结构上划分,数字系统由哪几部分组成?因此,所传输的二进制信息分为哪几类?

11.1.2 数字系统的设计方法有哪几种？各有什么特点？

11.2 算法状态机

在时序电路的分析和设计过程中，通常使用状态图或状态表描述只有几个输入和输出的电路，称为有限状态机(Finite State Machine，简称FSM)。对于数字系统这样较大型状态机，其输入和输出信号的数目较多。若仍然采用状态图或状态表等有限状态机的设计方法，则设计过程过于复杂，甚至不能完成整个系统的设计。因此，使用另一种不同的描述方法，即算法状态机(Algorithmic State Machine，简称ASM)图。ASM图是描述数字系统控制单元的工作流程图，主要用来描述控制单元的时序操作特性，说明控制条件及控制单元状态转换过程。其结构形式类似于计算机中的程序流程图。应用ASM图设计控制单元，可以很容易将语言描述的设计问题变成具有严格操作顺序和操作时间的时序流程图的描述，只要描述逻辑设计问题的时序流程图一旦形成，状态函数和输出函数就容易获得，从而得出相应的硬件电路。

11.2.1 ASM图形符号

ASM图中有三种基本的符号，即状态框、判断框和输出框。

1. 状态框

状态框用于表示控制单元的一个状态，如图11.2.1(a)所示。框内标出在此状态下处理器执行的操作和控制器输出等于1的逻辑变量，即穆尔型输出。状态的名称置于状态框左上角，分配给状态的二进制代码位于状态框的右上角，图11.2.1(b)为状态框实例。状态框的名称是 S_1 ，其代码是010，框内规定的处理器操作是 $A \leftarrow A+1$ ，即计数器的内容必须加1。输出信号Z只取决于该状态的状态变量值，也可以只标出有效的信号名，例如只写Z就表示其为1，而不必写 $Z=1$ 。图中的箭头表示控制单元状态的流向，在时钟脉冲触发沿的触发下，控制单元进入状态 S_1 ，在下一个时钟脉冲触发沿的触发下，控制单元离开状态 S_1 ，因此一个状态框占用一个时钟脉冲周期。

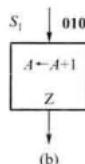


图 11.2.1 状态框与实例
(a) 状态框 (b) 状态框实例



图 11.2.2 判断框

2. 判断框

判断框表示根据输入条件的测试对状态转换分支做出的判断，如图11.2.2所示。它有一个

入口和多个出口，框内填判断条件，如果条件是真，选择一个出口，若条件是假，选择另一个出口。

判断框的入口来自某一个状态框，在该状态占用的一个时钟周期内，根据判断框中的条件，以决定下一个时钟脉冲触发沿来到时，该状态从判断框的哪个出口出去，因此，判断框不占用时间。

3. 条件输出框

条件输出框如图 11.2.3(a) 所示，它描述了对判断框中判断条件的响应。条件框的入口必定与判断框的输出相连。列在条件框内的处理器(R)操作或米利型输出是在给定的状态下，满足判断条件才发生的。米利型输出取决于状态变量以及输入信号的值。

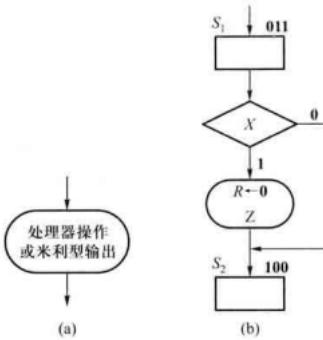


图 11.2.3 条件输出框与实例

(a) 条件输出框 (b) 实例

在图 11.2.3(b) 的例子中，当系统处于状态 S_1 时，若条件 $X=1$ ，则 R 被清 0，同时 $Z=1$ ；否则 R 保持不变；不论 X 为何值，系统的下一个状态都是 S_2 。米利型输出是输入的函数，其输出写在条件输出框内，而不是写在状态框内。

有时系统的输出有多个，既有穆尔型又有米利型。穆尔型输出写在状态框内，米利型输出写在条件输出框内。

4. ASM 图的定时关系

一个 ASM 图由多个 ASM 模块组成。每个模块包含一个状态框、若干个判别框和条件输出框。在同一个 ASM 模块中，状态框和条件输出框内定义的处理器的操作都是由同一个系统时钟信号控制下完成的，同时系统控制器在下一个时钟信号到达时进入下一个状态。例如在图 11.2.4(a) 中虚线内为一个 ASM 模块。当系统控制器进入 S_0 状态后，计数器的内容加 1。若这时 $E=0, F=0$ ，则下一个时钟信号到达时，控制器转入 S_1 状态。若这时 $E=0, F=1$ ，下一个时钟信号到达时，控制器转入 S_2 状态。如果 $E=1$ ，则 R 清零，下一个时钟信号到达时，控制器转入 S_3 状态。

图 11.2.4(b)给出了 ASM 图的各种操作及状态转换的时间图。假设系统中所有触发器都是上升沿触发的,在第一个时钟脉冲上升沿来到时,系统转换到 S_0 状态,随后根据条件由判断框输出 1(真)或 0(假),以便在下一个时钟脉冲上升沿到达时,系统的状态由 S_0 转换到 S_1 、 S_2 和 S_3 中的一个。

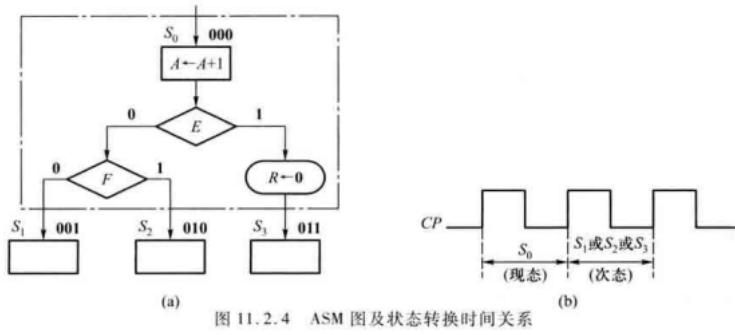


图 11.2.4 ASM 图及状态转换时间关系

(a) ASM 图 (b) 状态转换时间图

从表面上来看 ASM 图与程序流程图很相似,但实际上有很大的差异。程序流程图只表示事件发生的先后顺序,没有时间概念,而 ASM 图则不同,它表示事件的精确时间间隔顺序。

11.2.2 ASM 图与状态图

ASM 图可以比较清楚地描述数字系统的逻辑关系,不仅是人工设计数字系统的一个必要环节,也是用 EDA 软件设计时经常采用的系统逻辑描述方法。但这种描述形式与系统控制器硬件的对应关系不是十分明显。在时序电路的分析和设计中采用的状态图,可以清楚地反映出逻辑电路应提供多少个状态值,各个状态之间的转换必须符合什么条件。根据状态图可以设计出满足数字系统逻辑关系的控制电路。根据 ASM 图可以方便地画出对应的状态图。

一个 ASM 模块所表示的内容对应状态图中一个状态所表示的内容。因此,可以很容易将图 11.2.4(a)所示的 ASM 图,转换成状态图,如图 11.2.5 所示,其中 E 和 F 为状态转换条件。与 ASM 图不同,状态图无法表示处理器操作。

从 6.3.2 节的例 6.3.2 中 110 序列编码检测器设计过程可见,设计任务的第一步,也是最困难的一步,是根据简单的文字描述抽象出系统要完成的逻辑功能,得到状态图。当逻辑关系较复杂或数字系统较大时,逻辑抽象并非易事。ASM 是状态转换图按时钟信号顺序展开的另一种形式,能够更加直观地表示出时序电路的运行过程,并且同时表示控制器和处理器的操作。因此,当进行复杂电路或数字系统设计时,首先画出 ASM 图,然后转换为状态图进行控制。

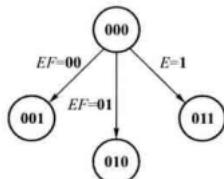


图 11.2.5 图 11.2.4 所示
ASM 的状态图

器的设计。

下面以设计**110**序列编码检测器为例(6.3.2节的例6.3.2)说明如何画米利型电路和穆尔型电路的ASM图,以及它们与米利型和穆尔型状态图的对应关系。

为了记忆电路输入端A输入3位编码的顺序,电路需要有3个状态: S_0 、 S_1 和 S_2 。 S_0 为初始状态,表示输入端A尚未输入过1; S_1 表示A端输入了一个1; S_2 表示A端输入了两个或两个以上的1。

(1) 在初始状态 S_0 ,当 $A=0$ 时,电路保持在 S_0 状态。当 $A=1$ 时,电路转到 S_1 状态,同时输出 $Y=0$ 。

(2) 在 S_1 状态,当 $A=0$ 时,电路回到初态 S_0 ,同时 $Y=0$ 。当 $A=1$ 时,电路转到 S_2 状态,表明已连续收到两个1,同时 $Y=0$ 。

(3) 在 S_2 状态,当 $A=0$ 时,表明电路收到**110**序列编码,此时 $Y=1$,同时电路回到初态 S_0 。若 $A=1$,电路保持在 S_2 状态。

由此得出米利型ASM图如图11.2.6所示。米利型输出取决于状态变量以及输入信号的值。电路的输出是用条件输出框表示,在给定的状态下,满足判断条件才发生的。

由米利型ASM图很容易得到状态图,如图11.2.6(b)所示。ASM图中的每一个状态框对应着状态图中表示状态的一个圆圈。判断框中的输入条件对应着状态转移线上的输入信号。条件输出框中给出的输出表示在该状态下,并且满足输入条件的输出,对应着状态图中相应状态转移线上标明的输出信号。

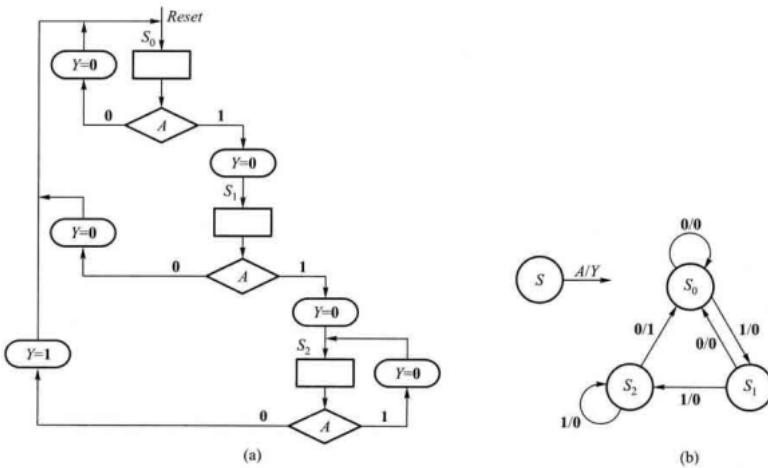


图11.2.6 110序列编码检测的米利型状态机ASM图及状态图

(a) ASM图 (b) 状态图

将上述分析过程与例 6.3.2 的 110 序列编码检测器设计对比可见, ASM 图更容易将文字描述抽象为系统要完成的逻辑功能, 然后得到状态图。

如果将 110 序列编码检测器设计成穆尔型电路, 其 ASM 图作图过程与米利型电路类似, 两者的区别在于, 穆尔型电路的输出仅是状态变量的函数, 因此相应的 ASM 图中没有条件输出框, 穆尔型输出标记在状态框内, 如图 11.2.7 所示。由穆尔型 ASM 图很容易得到状态图, 如图 11.2.7(b) 所示。穆尔型电路的状态图中, 电路输出标记在表示状态的圆圈内, 表明输出只是状态的函数。

与米利型电路 ASM 图相比, 穆尔型电路多了一个 S_4 状态, 即连续输入 110 序列编码的状态。这是因为米利型电路在连续输入两个 1 的 S_2 状态下, 如果 $A=0$ 时, 表明电路收到 110 序列编码, 不需先改变状态, 即刻输出 $Y=1$ 。电路可以在 S_2 的下一个状态回到初态 S_0 。而穆尔型电路的输出仅是状态变量的函数, 在 S_2 状态下, 如果 $A=0$ 时, 电路首先进入 S_3 状态, 在此状态下 $Y=1$ 。

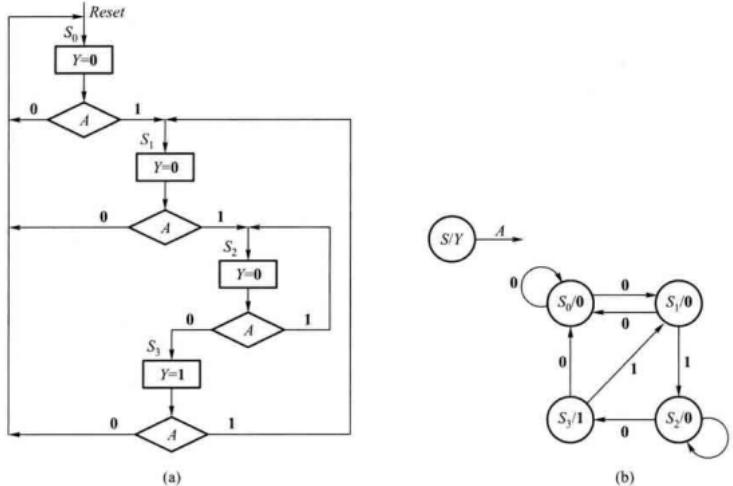


图 11.2.7 110 序列编码检测的穆尔型状态机 ASM 图及状态图

(a) ASM 图 (b) 状态图

需要指出, 通常在设计控制器电路或比较复杂的电路时, 电路的输出不止一个, 可以既有米利型输出, 又有穆尔型输出。电路的 ASM 图既包含状态输出框, 也包含条件输出框。

复习思考题

11.2.1 ASM 图中有哪几种基本符号? 各符号的含义是什么?

11.2.2 ASM 图中条件分支与条件输出有什么区别, 条件输出与状态输出有什么区别?

11.2.3 ASM 图与程序流图的根本区别是什么?

11.2.4 米利型和穆尔型电路的 ASM 图最大的区别是什么?

11.3 交通信号灯控制系统

交通信号灯自问世以来, 经历了机电定时控制, 继电器控制, 到现在计算机控制的发展过程。随着经济发展、城市人口增加、汽车的普及, 交通问题成为各个国家面临的共同问题。许多国家投入巨大人力和物力研究智能交通系统, 即运用计算机、通信、人工智能、传感器等领域先进的技术, 随时对各个交通路口信号进行调整以及对外界进行信息发布, 使整个交通系统的通行能力达到最大。

作为数字系统设计入门, 下面以最简单的交通灯控制系统为例, 说明数字系统设计过程, 并说明 ASM 图法在数字系统设计过程中的应用。数字系统的实现方法很多, 这里我们只介绍两种方法, 一种是用前面介绍的组合和时序电路的基本模块实现, 另一种是用可编程逻辑器件来实现。

11.3.1 交通信号灯控制系统 ASM 图

设计一个主干道和支干道十字路口的交通信号灯控制电路, 其技术要求如下:

(1) 一般情况下, 保持主干道畅通, 主干道绿灯亮、支干道红灯亮, 并且主干道绿灯亮的时间不得少于 60 s。

(2) 当主干道绿灯亮超过 60 s, 且支干道有车时, 主干道红灯亮、支干道绿灯亮, 但支干道绿灯亮的时间不得超过 30 s。

(3) 每次主干道或支干道绿灯变红灯时, 黄灯先亮 5 s。

设计过程如下:

(1) 明确系统的功能, 进行逻辑抽象

图 11.3.1 表示位于主干道和支干道的十字路口交通灯系统, 支干道两边安装传感器 S , 要求优先保证主干道的畅通。平时处于主干道绿灯、支干道红灯的状态。当支干道有车时, 传感器发出信号 $S=1$, 主干道绿灯先转换成黄灯、再变成红灯, 支干道由红灯变成绿灯。如果支干道继续有车通过时, 则传感器继续有信号, 使支干道保持绿灯亮, 但支干道绿灯持续亮的时间不得超过 30 s, 否则支干道绿灯先转换成黄灯再变成红灯, 同时主干道由红灯变成绿灯。主干道每次通行时间不得短于 60 s, 在此期间, 即使支干道 S 有信号, 也不能中断主干道的绿灯亮。

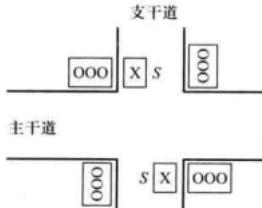


图 11.3.1 交通灯示意图

(2) 确定系统方案并画出 ASM 图。

系统由控制单元和处理单元组成, 控制单元接收外部系统时钟和传感器信号。处理单元由定时器和译码显示器组成。定时器能向控制单元发出 60 s 、 30 s 或 5 s 定时信号, 译码显示电路在控制单元的控制下, 改变交通灯信号。根据题目要求画出系统框图如图 11.3.2 所示, 其中:

HG, HY, HR 分别表示主干道绿、黄、红三色灯。

FG, FY, FR 分别表示支干道绿、黄、红三色灯。

交通灯系统工作主要有三个不同的时间段, T_L 、 T_S 和 T_Y

T_L : 主干道绿灯亮的最短时间, 不少于 60 s 。

T_S : 支干道绿灯亮的最长时间, 不多于 30 s 。

T_Y : 主干道或支干道黄灯亮的时间为 5 s 。

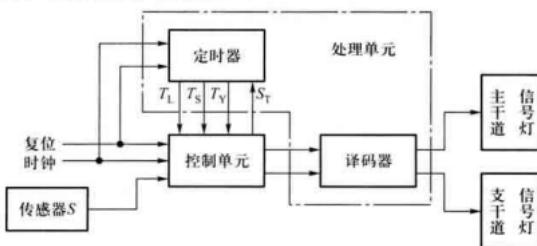


图 11.3.2 交通灯系统框图

因此, 用定时器分别产生三个持续的时间段后, 向控制单元发出时间已到信号, 控制单元根据定时器及传感器的信号, 决定是否进行状态转换。如果肯定, 则控制单元发出状态转换信号 S_T , 定时器开始清零, 准备重新计时。

交通灯控制单元的控制过程分为四个阶段, 对应的输出有四种状态, 分别用 S_0, S_1, S_2 和 S_3 表示:

S_0 状态: 主干道绿灯亮支干道红灯亮, 此时若支干道有车等待通过, 而且主干道绿灯已亮足规定的时间 T_L , 控制器发出状态转换信号 S_T , 输出从状态 S_0 转换到 S_1 。

S_1 状态: 主干道黄灯亮, 支干道红灯亮, 进入此状态, 黄灯亮足规定的时间 T_Y 时, 控制器发出状态转换信号 S_T , 输出从状态 S_1 转换到 S_2 。

S_2 状态: 支干道绿灯亮, 主干道红灯亮, 若此时支干道继续有车, 则继续保持此状态, 但支干道绿灯亮的时间不得超过 T_S , 否则控制单元发出状态转换信号 S_T , 使输出转换到 S_3 状态。若此时支干道没有车, 则控制单元立即发出状态转换信号 S_T , 使输出转换到 S_3 状态。

S_3 状态: 支干道黄灯亮, 主干道红灯亮, 此时状态与 S_1 状态持续的时间相同, 均为 T_Y , 时间到时, 控制器发出 S_T 信号, 输出从状态 S_3 回到 S_0 状态。

对上述 S_0, S_1, S_2 和 S_3 四种状态按照格雷码进行编码分别为 **00, 01, 11** 和 **10**, 由此得到交通灯控制单元的 ASM 图如图 11.3.3 所示。异步复位信号 *Reset* 使控制单元直接进入主干道绿灯

亮、支干道红灯亮的初始状态,用 S_0 状态框表示。当 S_0 状态持续时间 T_L 大于等于 60 s, 并且支干道有车等待通过使传感器 $S = 1$ 。两个状态转换条件同时满足使 $T_L \cdot S = 1$, 即满足判断框中的 $T_L \cdot S = 1$ 条件, 控制单元发出状态转换信号 S_T , 由条件输出框表示, 同时系统从状态 S_0 转到主干道黄灯亮、支干道红灯亮的 S_1 状态。

在 S_1 状态, 黄灯亮足规定的时间 T_Y , 即状态转换的条件 $T_Y = 1$ 时, 系统转到 S_2 状态。

在 S_2 状态, 有两个条件控制系统状态转换, 一个是支干道绿灯亮足规定的时间 T_S , 另一个是否支干道没有车 $S = 0$ 。只要满足两个条件之一, 即 $T_S + \bar{S} = 1$, 系统立即转换到 S_3 状态。

以此类推得出 11.3.3 所示的 ASM 图的图。

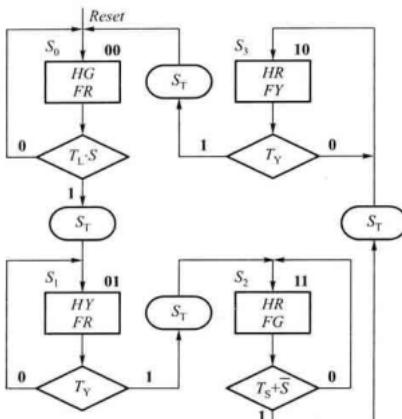


图 11.3.3 交通灯控制单元 ASM 图

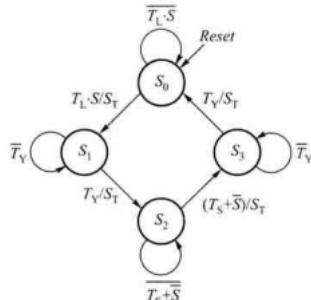


图 11.3.4 交通灯控制单元状态图

根据交通灯控制单元的 ASM 图, 得出其状态图如图 11.3.4 所示。ASM 图中的状态框与状态图中的状态相对应, 判断框中的条件是状态转换的输入条件, 条件输出框与控制单元状态转换的输出相对应。状态图是描述状态之间的转换, 例如在 S_0 状态, 如果条件 $T_L \cdot S = 1$ 时, 系统状态转移到 S_1 , 同时输出状态转换信号 S_T 。如果 $T_L \cdot S = 0$, 则系统保持在 S_0 状态。

图中未画出四种状态下主干道和支干道交通灯的输出情况, 它们是四种状态的组合输出。 S_T 是控制单元的输出信号, 用来控制定时器的工作。

数字系统的实现可以采用典型的组合和时序逻辑电路模块实现, 也可以用可编程逻辑器件实现。实现的方法不同, 则后续设计方案也不同。

11.3.2 用典型电路基本模块实现交通灯控制系统

通过分析交通灯控制系统的要求可知, 系统主要由传感器、时钟脉冲产生器、定时器、控制单

元及译码器构成,传感器 S 在有车辆通过时发出一个高电平信号。时钟脉冲产生器由石英晶体组成(见第 9 章)。下面介绍用典型的组合和时序电路模块设计实现交通灯控制系统。

1. 设计控制单元电路

根据图 11.3.4 所示的交通灯控制单元电路的状态图,用 6.3 节介绍的同步时序电路的设计方法即可设计出电路。其中状态分配方案有:自然二进制码、格雷码和“一对一”。“一对一”是指一个触发器对应一个状态。这里主要介绍采用格雷码进行状态分配的方法设计交通灯控制单元。

根据状态图 11.3.4 可知,交通灯控制单元有 4 种状态,选用两个触发器 FF_1, FF_0 的 4 种输出组合对其进行格雷码的编码,可以列出状态转换表,如表 11.3.1,控制单元状态转换的条件为 $T_L \cdot S, T_Y$ 和 $T_S + \bar{S}$ 。 $Reset$ 信号将控制单元置为初始状态 $Q_1^n Q_0^n = 00$ 以后,如果 $T_L \cdot S = 0$,则控制单元保持在 00 状态;如果 $T_L \cdot S = 1$,则控制器转换到 $Q_1^{n+1} Q_0^{n+1} = 01$ 状态。这两种情况与条件 T_Y 和 $T_S + \bar{S}$ 无关,所以用无关项“ \times ”表示。其余情况以此类推,同时表中列出状态转换信号 S_T 。

表 11.3.1 控制单元状态转换表

现态 符号	现态 $Q_1^n Q_0^n$	输入			输出	
		$T_L \cdot S$	T_Y	$T_S + \bar{S}$	$Q_1^{n+1} Q_0^{n+1}$	S_T
S_0	0 0	0	\times	\times	0 0	0
S_0	0 0	1	\times	\times	0 1	1
S_1	0 1	\times	0	\times	0 1	0
S_1	0 1	\times	1	\times	1 1	1
S_2	1 1	\times	\times	0	1 1	0
S_2	1 1	\times	\times	1	1 0	1
S_3	1 0	\times	0	\times	1 0	0
S_3	1 0	\times	1	\times	0 0	1

在表 11.3.1 中,将 Q_1^{n+1}, Q_0^{n+1} 和 S_T 为 1 的项所对应的输入和状态转换条件变量相与,其中“1”用原变量表示,“0”用反变量表示,然后将各与项相或,可以推出状态方程和转换信号方程如下:

$$Q_1^{n+1} = \overline{Q_1^n} Q_0^n T_Y + Q_1^n \overline{Q_0^n} \overline{T_Y} + Q_1^n Q_0^n$$

$$Q_0^{n+1} = \overline{Q_1^n} \overline{Q_0^n} T_L \cdot S + \overline{Q_1^n} Q_0^n + Q_1^n Q_0^n \overline{T_S + \bar{S}}$$

$$S_T = \overline{Q_1} \overline{Q_0} T_L + S \overline{Q_1} Q_0 T_Y + Q_1 \overline{Q_0} T_Y + Q_1 Q_0 (T_S + \overline{S})$$

FF_1 和 FF_0 选用 D 触发器, 每个触发器的输入函数用数据选择器选择, 所以数据选择器的数目等于触发器的数目。考虑到控制信号 S_T 与输入变量的关系, 也需要用一个数据选择器, 所以共选用三个数据选择器, 并将触发器的现态值加到数据选择器的选择变量端, 数据选择器的输入端信号可以根据状态方程和转换信号方程得出。交通灯控制单元的逻辑图如图 11.3.5 所示。

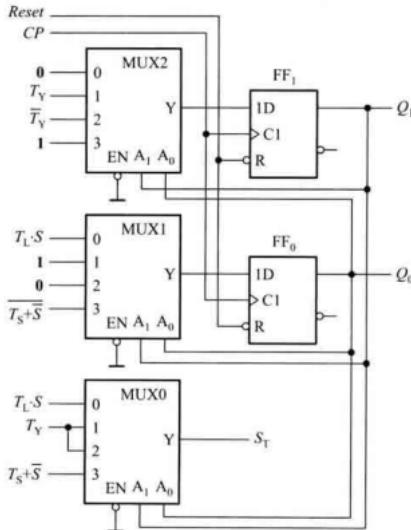


图 11.3.5 交通灯控制单元电路

2. 设计处理单元电路

(1) 定时器

定时器由计数器构成。该计数器与系统时钟(由时钟脉冲产生器提供)同步, 时钟脉冲上升沿到来时, 在控制信号 S_T 作用下, 计数器从零开始计数, 并向控制单元提供模 M5、M30 和 M60 信号, 即 T_Y 、 T_S 和 T_L 三个不同的持续时间信号。计数器工作在 M60 状态, 并且当计数到 4 和 29 时, 也给出 T_Y 、 T_S 信号, 但此时系统处于 S_0 状态, 控制单元只检测 T_L 信号而不响应 T_Y 、 T_S 信号。

当系统处于 S_0 状态, 为满足主干道绿灯亮, 支干道红灯亮的持续时间 $T_L \geq 60$ s, 所以要将 M60 的输出端反馈到计数器的使能端 EN, 使它计到 59 时停止计数, 并保持在 $M=60$ 的状态直到支干道有车要通过时, 才转换到 S_1 状态。

要求计数器在状态转换信号 S_T 作用下, 首先清零, 然后开始计数。定时器框图如图 11.3.6 所示。

计数器具有高电平有效使能端 EN , 低电平有效同步清零端 CLR 和进位输出端 C_o 。当计数到 $Q_5Q_4Q_3Q_2Q_1Q_0 = 111011$, 即 $M=60$ 时, $C_o = 1$, 将其反相后接入使能端, 就可以保持在 $M=60$ 状态。控制单元发出的 S_t 信号是高电平有效, 所以经反相后接至计数器清零端, 具体电路可使用中规模芯片自行设计。

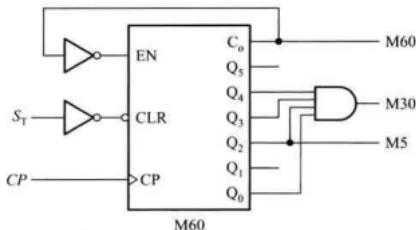


图 11.3.6 定时器框图

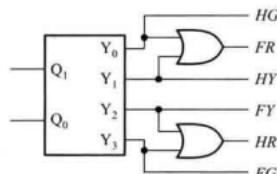


图 11.3.7 信号灯的译码电路

(2) 译码器

系统的输出是在 Q_1Q_0 驱动下的六个信号灯, 各状态与信号灯的关系由表 11.3.2 给出。因此, 得到灯光信号与控制单元状态变量的关系为

$$\begin{aligned} HG &= \overline{Q}_1 \overline{Q}_0 & HY &= \overline{Q}_1 Q_0 & HR &= Q_1 \\ FG &= Q_1 Q_0 & FY &= Q_1 \overline{Q}_0 & FR &= \overline{Q}_1 \end{aligned}$$

实现上述关系的译码电路如图 11.3.7 所示。

表 11.3.2 信号灯与控制器状态编码表

状态	HG	HY	HR	FG	FY	FR
S_0	1	0	0	0	0	1
S_1	0	1	0	0	0	1
S_2	0	0	1	1	0	0
S_3	0	0	1	0	1	0

D 触发器选用 74HC74, 多路选择器选用 74HC153, 译码器选用 74HC139 外加反相器 (74HC04) 构成输出高电平有效电路, 计数器选用 74HC163, 就可构成系统电路。系统逻辑图略。系统工作波形如图 11.3.8 所示。

11.3.3 用可编程逻辑器件实现交通灯控制系统

当采用可编程逻辑器件实现数字系统时, 一个重要任务是用 HDL 描述数字系统。前面几章我们介绍了组合电路、时序电路, 以及选择器、计数器和寄存器等一些标准器件的 Verilog HDL

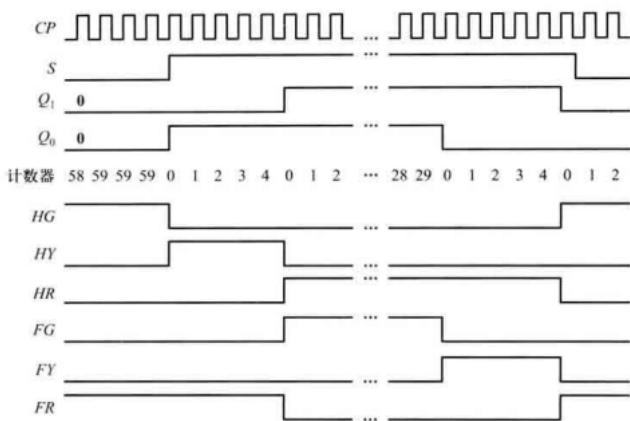


图 11.3.8 交通灯系统工作波形

描述。这里将介绍如何借助于 ASM 图、状态图,用 Verilog HDL 描述一个完整的数字系统。数字系统的 HDL 描述可以在结构级或行为级进行,行为级又可分为寄存器级或算法级。因此,对数字系统的描述可以分为结构级描述、寄存器传输级(RTL)描述和基于算法的行为级描述。

结构级描述是最底层、最详细的描述。它是根据具体物理元件以及它们之间的连接描述系统。这些元件包括门、触发器及选择器、计数器等标准部件。描述过程首先是将系统划分为多个不同功能的模块,然后用 HDL 描述每个功能模块,最后将所有这些底层模块组合起来构成顶层模块,即完成了整个系统的设计。

寄存器传输级描述是根据寄存器要完成的操作,以及操作的顺序来描述系统。这种类型的描述是用过程语句说明各种操作的关系,不涉及具体硬件电路结构。但是,寄存器传输级描述隐含了寄存器的硬件电路结构,可以用标准部件实现系统。

基于算法的行为级描述是最抽象的。用类似于编程语言中的过程算法形式描述系统的功能,不涉及任何的硬件电路实现。因此,这一层设计的某些描述不能被开发软件综合成具体结构形式。这种行为级描述适合于复杂数字系统的仿真,用来证明设计是否正确。

下面我们将通过实例介绍数字系统的结构级和 RTL 级描述。

1. 寄存器传输级描述

交通灯控制系统的传输级 HDL 描述程序如下,分为 4 部分。第一部分声明了系统的输入、输出、所用的寄存器。其输入信号为时钟 CLK、传感器 S 和复位信号 RESET。输出信号有主干道和支干道信号灯 HG、HY、HR 及 FR、FY、FG。第二部分描述控制单元工作的时序关系。控制寄存器有 4 个状态,由两个 D 触发器构成,用二进制数表示,NextState 表示 D 触发器的输出,Cur-

rentState 表示 D 触发器的输入。第三部分和第四部分描述了处理单元中寄存器的传输操作和输出。

控制单元的 HDL 描述是根据图 11.3.4 所示的状态图编写的。用两个 **always** 语句描述其时序转换过程。第一个 **always** 语句说明两个操作过程：异步复位信号 RESET 使系统进入初态 S0，系统状态转换是在时钟 CLK 的上升沿进行的。第二个 **always** 语句是由 **case** 多路分支语句构成的一个组合逻辑电路，说明由现态到次态的转换条件。例如，现态是 S0，如果 $T1 \cdot S = 1$ ，则下一个 CLK 的上升沿转到 S1 状态。如果 $T1 \cdot S = 0$ ，则保持在 S0 状态。

第三部分定时器的 HDL 编写，可以根据 ASM 图。其工作过程由 RESET 和 St 控制。RESET 信号使定时器清零后开始计数。St 是由控制单元发出的信号，使定时器清零后重新计数。在 S0、S2、S1（或 S3）状态下，定时器分别给出 3 个定时信号 T1、Ts 和 Ty，以供控制单元决定是否进行状态转换。

第四部分输出电路的 HDL 根据图 11.3.7 或图 11.3.8 编写。由 **case** 语句说明在不同的状态下，两组输出信号（HG、HY、HR 或 FR、FY、FG）中均有一个为 1。

```
module Traffic_RTL (
    /* ====== 声明输入/输出端口, 参见图 11.3.2 ===== */
    input CLK, S, RESET,
    output reg HG, HY, HR, FG, FY, FR //输出信号是寄存器类型
);
//声明系统内部的信号变量及其类型, 见图 11.3.2
reg [3:0] TimerH, TimerL; //用于定时器的寄存器
wire T1, Ts, Ty;
reg St;
reg [1:0] CurrentState, NextState; //控制寄存器
//对状态进行编码
parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b11, S3 = 2'b10;
/* ====== 控制单元状态转换的描述, 参见图 11.3.4 ===== */
always @ (posedge CLK, negedge RESET)
begin; statereg
    if (~RESET)
        CurrentState <= S0; //初始 S0 状态
    else CurrentState <= NextState;
end //statereg
always @ (S or CurrentState or T1 or Ts or Ty )
begin; fsm
    case (CurrentState)
        S0: begin
```

```

        NextState = (Tl && S) ? S1 : S0;
        St = (Tl && S) ? 1:0;
    end
S1: begin
        NextState = (Ty) ? S2 : S1;
        St = (Ty) ? 1:0;
    end
S2: begin
        NextState = (Ts || ~S) ? S3 : S2;
        St = (Ts || ~S) ? 1:0;
    end
S3: begin
        NextState = (Ty) ? S0 : S3;
        St = (Ty) ? 1:0;
    end
endcase
end //fsm
/* =====定时器工作的描述===== */
always @ (posedge CLK or negedge RESET)
begin;counter
if (~RESET)
    |TimerH, TimerL| <= 8'b0;
else if (St)
    |TimerH, TimerL| <= 8'b0;
else if ((TimerH==5) & (TimerL== 9))
    begin |TimerH, TimerL| <= |TimerH, TimerL|; end
else if (TimerL == 9)
    begin TimerH <= TimerH + 1; TimerL <= 0; end
else
    begin TimerH <= TimerH; TimerL <= TimerL + 1; end
end //counter
assign Ty = (TimerH==0) & (TimerL==4);
assign Ts = (TimerH==2) & (TimerL==9);
assign Tl = (TimerH==5) & (TimerL==9);
/* =====输出译码电路的描述===== */
always @ (CurrentState)
begin
    case (CurrentState)

```

```

S0;begin
    [HG, HY, HR] = 3'b100; //主干道绿灯亮
    [FG, FY, FR] = 3'b001; //支干道红灯亮
end
S1;begin
    [HG, HY, HR] = 3'b010; //主干道黄灯亮
    [FG, FY, FR] = 3'b001; //支干道红灯亮
end
S2;begin
    [HG, HY, HR] = 3'b001; //主干道红灯亮
    [FG, FY, FR] = 3'b100; //支干道绿灯亮
end
S3;begin
    [HG, HY, HR] = 3'b001; //主干道红灯亮
    [FG, FY, FR] = 3'b010; //支干道黄灯亮
end
endcase
end
endmodule

```

2. 结构级描述

前面介绍了用 HDL 中的过程语句,对交通灯控制系统的功能进行寄存器传输级描述。使用开发软件中的逻辑综合工具对其进行编译,就可以获得等效的门级电路描述。我们也可以根据构成电路的元器件,以及它们之间的连接关系,直接对交通灯控制系统进行结构级描述。

根据交通灯控制系统结构图 11.3.2 可知,电路由三部分组成:控制单元、定时器和输出译码电路。按照数字系统自上而下分模块、分层次的设计方法,整个系统 HDL 描述分为三个层次,如图 11.3.9 所示。顶层系统的 HDL 描述是由控制单元、定时器和输出译码电路三部分的 HDL 描述组成。而控制单元又由底层的两个 D 触发器和 3 个 4 选 1 数据选择器构成。

下面是交通灯控制系统结构级 HDL 描述。它由 6 个模块构成,可以分为 4 部分:

(1) 第一个顶层模块调用 3 个设计块。调用时端口采用了两种连接方式,U0、U1 是位置对应的调用方式,即调用时端口的排列顺序必须跟下层模块定义时端口的排列顺序一致;U2 是端口名称对应的调用方式,端口顺序可任意排列,圆括号内部是顶层模块使用的端口名称,圆括号外部是下层模块使用的端口名称。

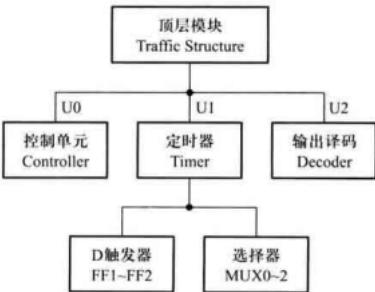


图 11.3.9 交通灯控制系统层次结构框图

- (2) 后面 3 个模块描述了控制单元、D 触发器和数据选择器。
- (3) 第五个模块描述了定时器。
- (4) 最后一个模块描述了输出电路。

与寄存器传输级类似，顶层模块定义了系统的输入和输出。由于在底层说明了输出信号的类型，这里可以省略。顶层模块由后面 3 个模块构成，它们的输入输出端口中，有些是系统的输入输出端口，有些则是系统内其他模块产生的内部信号。例如控制模块 U0 的输入 Tl、Ts、Ty，是定时器 U1 的输出。

控制模块是根据图 11.3.5 描述的。其中两个 D 触发器的输入 Y2、Y1 和输出 Q1、Q0 均定义为 **wire** 类型，这是因为触发器现态 Q1、Q0 作为控制单元的输出是组合型的，而 Y2、Y1 是后面调用的组合电路数据选择器的输出。控制模块调用了 3 个选择器模块和 2 个 D 触发器模块。然后描述 D 触发器和数据选择器模块。数据选择器的输出 Y 定义为 **reg** 类型，是因为 **always** 块中，被赋值的信号必须是 **reg** 型。

最后两个模块是具有同步清零的定时器和输出译码电路。其中 M60、M30、M5 和 HG、HY、HR、FG、FY、FR 定义为 **wire** 类型，是因为后面的 **assign** 语句要求被赋值的信号必须是 **wire** 型。

```
module Traffic_Structure (
/* ====== 定义输入/输出端口, 参见图 11.3.2 ===== */
  input CLK, S, RESET,
  output HG, HY, HR, FG, FY, FR
);
// 定义系统内部的信号变量及其类型
  wire [3:0] TimerH, TimerL; // 用于定时器的寄存器
  wire Tl, Ts, Ty;
  wire St;
  wire Q1, Q0; // 控制寄存器
Controller U0(St, Q1, Q0, S, Tl, Ts, Ty, RESET, CLK);
Timer U1(.M60(Tl), .M30(Ts), .M5(Ty), .CLR(RESET), .St(St), .CP(CLK), .TimerH(TimerH), .TimerL(TimerL));
Decoder U2( HG, HY, HR, FG, FY, FR, Q1, Q0);
endmodule
```

/ ===== 控制单元的描述, 参见图 11.3.5 ===== */*

```
module Controller (
  input S, Tl, Ts, Ty, CLR, CP,
  output St, Q1, Q0
);
  supply1 Vec;
  supply0 GND;
```

```
Mux4x1 MUX2( Y2,GND,Ty,~Ty,Vcc,| Q1,Q0|,GND);
Mux4x1 MUX1( Y1,(Tl&S),Vec,GND,~(Ts|~S),| Q1,Q0|,GND);
Mux4x1 MUX0( St,(Tl&S),Ty,Ty,(Ts|~S),| Q1,Q0|,GND);
UDFF FF1(Q1,Y2,CLR,CP);
UDFF FF0(Q0,Y1,CLR,CP);
endmodule
```

/ =====D 触发器的描述===== */*

```
module UDFF (
  input D,CLR,CP,
  output reg Q
);
  always @ ( posedge CP, negedge CLR)
    if (~CLR)
      Q <= 1'b0;
    else Q <= D;
endmodule
```

/ =====数据选择器的描述===== */*

```
module Mux4x1 (
  input I0,I1,I2,I3,EN,
  input [1:0] SEL,
  output reg Y
);
  always@ (I0, I1, I2, I3, SEL, EN)
    if (EN) Y = 1'b0;
    else
      case (SEL)
        2'b00: Y=I0;
        2'b01: Y=I1;
        2'b10: Y=I2;
        2'b11: Y=I3;
      endcase
endmodule
```

/ =====定时器模块的描述===== */*

```
module Timer (
  input CLR, St, CP,
```

```

output M60, M30, M5,
output reg [3:0] TimerH, TimerL,
);
always @ (posedge CP, negedge CLR)
begin; counter
if (~CLR)
    |TimerH, TimerL| <= 8'h00;
else if (St)
    |TimerH, TimerL| <= 8'h00;
else if ( |TimerH, TimerL| == 8'h59)
    begin |TimerH, TimerL| <= |TimerH, TimerL|; end
else if (TimerL == 9)
    begin TimerH <= TimerH + 1'bl; TimerL <= 4'b0000; end
else
    begin TimerH <= TimerH; TimerL <= TimerL + 1'bl; end
end //counter
assign M5 = (TimerH==0)&(TimerL==4);
assign M30 = (TimerH==2)&(TimerL==9);
assign M60 = (TimerH==5)&(TimerL==9);
endmodule

```

/* =====输出译码电路的描述,参见图 11.3.7===== */

```

module Decoder (
input Q1, Q0,
output HG, HY, HR, FG, FY, FR
);
assign HG = ~Q1 & ~Q0;
assign HY = ~Q1 & Q0;
assign HR = Q1;
assign FG = Q1 & Q0;
assign FY = Q1 & ~Q0;
assign FR = ~Q1;
endmodule

```

比较上述交通灯控制系统寄存器传输级和结构级的 Verilog 描述可知,寄存器传输级描述简单方便,不需要知道电路的具体结构。随着数字系统复杂性的增加,寄存器传输级描述更表现出优越性。

3. 用 FPGA 器件实现交通灯控制系统

用 Altera 公司的 FLEX10K 系列器件 EPF10K10LC84-4 实现上述设计的过程如下:

(1) 在 Quartus II 9.1 软件中建立一个新的工程项目, 输入上述 HDL 文件, 对设计项目进行编译。

(2) 新建一个仿真波形文件, 给出输入、输出信号的激励波形, 对设计项目进行时序仿真, 得到如图 11.3.10 所示的波形。

分析波形图可知, 在 AB 段开始时, 低电平有效的 RESET 信号使系统直接进入主干道绿灯亮、支干道红灯亮的初始状态。BC 段表示支干道有车, $S=1$, 并且主干道绿灯需亮足 60 s, 系统转换到 CD 段, 主干道黄灯亮、支干道红灯亮, 持续 5 s 后转换到 DE 段, 此时主干道红灯亮、支干道绿灯亮。由于支干道一直有车, DE 段持续 30 s 后, 系统才进入主干道红灯亮、支干道黄灯亮的 EF 段, 5 s 后又回到主干道绿灯亮、支干道红灯亮的状态, 如图中 FG 段。在主干道红灯亮、支干道绿灯亮的 HI 段, 当支干道没有车 $S=0$ 时, 系统立刻转换到下一个状态, 并准备进入主干道绿灯亮、支干道红灯亮的状态。仿真结果完全符合设计要求。

(3) 选定目标器件 EPF10K10LC84-4, 将输入、输出信号分配到器件相应的引脚上, 如表 11.3.3 所示。然后重新编译设计项目, 生成下载文件(文件后缀为.pof 或.sof)。

(4) 将下载文件写入到目标器件中, 就是一块专用的交通灯控制电路。

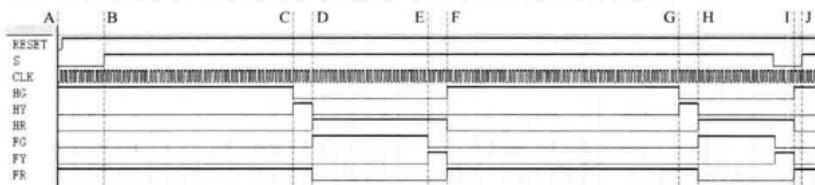


图 11.3.10 交通灯控制系统的仿真波形图

表 11.3.3 用 EPF10K10LC84-4 实现交通灯控制系统引脚分配表

引脚名称	器件对应的引脚号	引脚功能说明	引脚名称	器件对应的引脚号	引脚功能说明
CLK	3	系统时钟	FG	72	支路绿灯输出
RESET	8	系统复位	HG	79	主路绿灯输出
S	9	传感器输入信号	HY	80	主路黄灯输出
FR	70	支路红灯输出	HR	81	主路红灯输出
FY	71	支路黄灯输出			

复习思考题

11.3.1 用 Verilog HDL 描述数字系统时, 可以分为几种形式? 各种描述形式的特点是什么?

11.3.2 数字系统中, 通常借助于什么编写控制单元的 Verilog HDL 描述?

11.3.3 比较交通灯控制系统的结构级描述和寄存器传输级描述,你得出什么结论?

11.4 数字密码锁

电子技术的发展使许多智能密码锁问世,如刷卡式、指纹类等。这类密码锁适应于保密要求高,仅供个人使用的场合,而且 IC 卡有易丢失的特点,加上成本一般较高,在一定程度上限制了这类产品的使用。数字密码锁适用范围广,特别适合人员变动大,开锁钥匙不止一人拥有的场合。

作为数字系统设计入门,下面以简单的数字密码锁为例,说明数字系统的设计过程,以及 ASM 图法在数字系统设计过程中的应用。并且分别介绍用组合和时序电路的基本模块和可编程逻辑器件来实现数字密码锁。

11.4.1 数字密码锁的 ASM 图

设计一个 8 位串行数字密码锁,该锁只有当依次收到的 8 位串行码与规定的二进制数码一致时,才能被打开。

设计过程如下:

(1) 明确系统的功能,进行逻辑抽象

数字密码锁内部由 8 个拨动开关设置了 8 位二进制密码,分别用 $D_7, D_6, D_5, D_4, D_3, D_2, D_1, D_0$ 表示。开锁时的串行输入数码由开关 BIT 产生,可以为 0 或 1,如图 11.4.1 所示,为了使系统能 1 位 1 位地依次读取由 BIT 开关送来的串行数码,设置了 1 个按钮开关 $READ$,送入数码时,首先用 BIT 开关设置 1 位数码,然后按下 $READ$ 开关,这样就将 BIT 开关产生的当前数码读入系统。为了标识串行数码输入开始和结束,设置了 $RESET$ 和 TRY 按钮开关, $RESET$ 信号使系统进入初始状态,准备接受新的串行数码,当送 8 位数码与开锁密码一致时,按下 TRY 产生开锁信号,系统便输出 $OPEN$ 信号打开锁,否则数字锁不开,并输出错误信号 $ERROR$ 。

(2) 将系统划分为控制单元和处理单元

将数字密码锁划分为控制单元和处理单元两部分如图 11.4.2 所示,控制单元发出控制信号使处理单元接受 BIT 产生的数码,并将其与对应的密码位相比较,比较结果 B 送回到控制单元作为控制器状态转换条件之一。为累计输入数码的位数,需要一个计数器。控制单元发出的清零信号 CLR 使计数器清零,并使数码的比较从低位开始,同时计数器开始累计输入数码的次数并与密码的位数相比较,两者相等则输出一个控制信号 M 到控制单元。

图 11.4.3 给出了处理单元逻辑结构图。由 8 位拨动开关设置的数码作为 8 选 1 多路选择

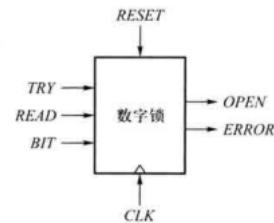


图 11.4.1 数字锁框图

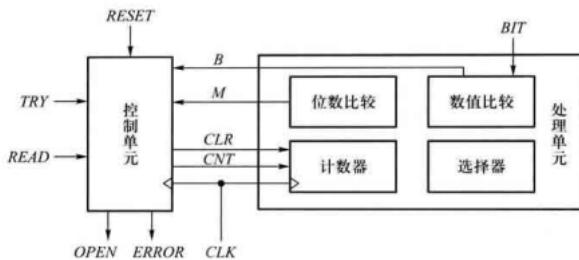


图 11.4.2 数字锁系统结构图

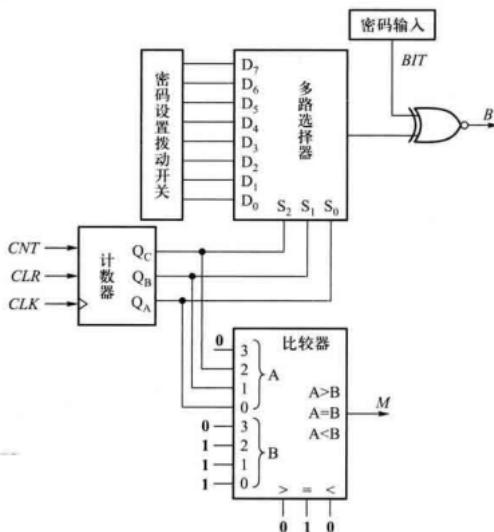


图 11.4.3 处理单元逻辑结构图

器的数据输入，3 位二进制计数器的输出作为多路选择器的选择数据输入。多路选择器的输出与 BIT 开关产生的数码相比较，两者相同时输出 B 为 1，不同时为 0。复位后，控制单元发出 CLR 命令使计数器清零，在控制信号 CNT 的作用下，多路选择器的输入数据从低位到高位逐位被选择出来。控制单元根据处理电路反馈回来的 B 状态信息，获得各次比较结果。开锁密码位数的确定由比较器完成，当输入数码的位数为 8 位时，比较器输出 M 为 1，否则为 0。

(3) 确定控制单元的算法,画出 ASM 图

根据上述分析得出控制单元的 ASM 图,如图 11.4.4 所示,其中 S_0 为初始状态, S_1 为接收数码状态, S_2 为准备开锁状态, S_3 状态表示每正确接收一次数码,计数器 C 加 1, S_4 为开锁状态, S_5 为错误状态。

异步复位信号 $RESET$ 使得控制单元进入初始状态,在该状态时,控制单元发出 CLR 命令使计数器 C 清零。下一个时钟到来时,系统无条件地转到接收数码状态 S_1 。

在 S_1 状态下若开锁信号 TRY 为 1 时,控制单元进入错误状态 S_5 , TRY 为 0 时,等待接收数码。 $READ$ 为 1 时读取数码,若本次送的数码 BIT 与开锁密码相应位的数值不相等,比较器输出 B 为 0,控制单元进入错误状态 S_5 ,若两者数值相等,比较器输出 B 为 1,然后判断位数比较器输出 M 的结果, M 为 0 时控制器进入 S_3 状态。

在 S_3 状态,控制单元输出 CNT 命令使计数器 C 加 1,然后转换到接收数据状态 S_1 ,这样循环下去直到正确接收 8 次数码后,位数比较器 M 输出为 1,系统转入 S_2 状态。

S_2 为准备开锁状态,在 S_2 状态下再读数时,即 $READ$ 为 1 时,控制单元进入错误状态 S_5 ,否则等待开锁信号 TRY , TRY 为 1 时,控制单元由 S_2 状态进入开锁状态 S_4 ,并输出 $OPEN$ 开锁命令。

在上述过程中,任何一次送入的数码与开锁密码的数值不一致,或者 $READ$ 开关和 TRY 开关使用的顺序与规定的不符,都将使控制单元进入错误状态 S_5 ,同时控制单元发出错误信息 $ERROR$ 。

根据图 11.4.4 所示的数字密码锁控制单元的 ASM 图,得出状态图如图 11.4.5 所示。ASM 图中的状态框与状态图中的状态相对应,判断框中的条件是状态转换的输入条件,状态框中的输出与状态图中的输出相对应。例如在接收数码 S_1 状态,如果条件 $\overline{TRY} \cdot \overline{READ} \cdot \overline{B} \cdot \overline{M}$ 为 1,系统转到 S_3 状态等待下一个时钟到来时,再回到 S_1 状态接收下一个数码。控制单元有多个输出,状态图中标出输出变量的,在该状态下为 1,其余情况下各个输出均为 0。如在 S_3 状态, CNT 为 1,其余输出均为 0。

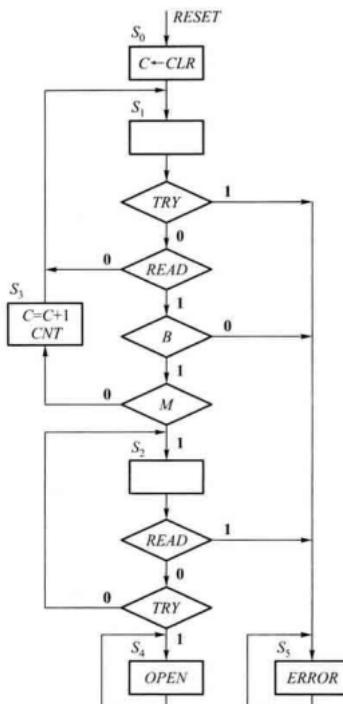


图 11.4.4 控制单元 ASM 图

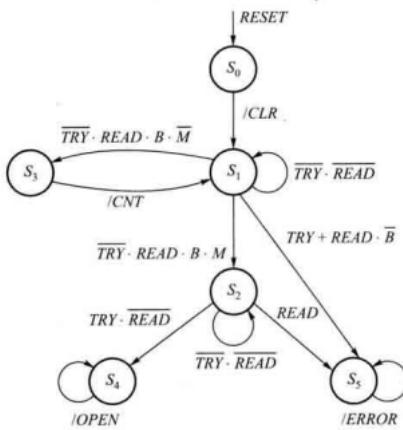


图 11.4.5 控制单元状态图

11.4.2 用典型电路基本模块实现数字密码锁

数字密码锁系统主要由控制单元及位数比较器、数值比较器、计数器和选择器构成的处理单元组成。下面介绍用典型的组合和时序电路模块设计实现数字密码锁系统。

对于数字密码锁控制单元的设计,根据图 11.4.4 所示的 ASM 图,采用“一对一”的编码方案,用六个 D 触发器 $FF_0 \sim FF_5$ 的输出分别表示 $S_0 \sim S_5$ 六种状态,开锁过程中的每一时刻,只能有一个状态为 1,其余状态为 0。根据图 11.4.5 所示的状态图,可以列出控制单元的状态转换表如表 11.4.1 所示。以次态是 S_1 状态为例说明列表过程。

表 11.4.1 控制器状态转换表

输入		输出
现态 Q_i^*	转换条件	次态 Q_i^{*+1}
S_0	1	
S_1	$\overline{TRY} \cdot \overline{READ}$	S_1
S_3	1	
S_1	$TRY \cdot READ \cdot B \cdot M$	
S_2	$TRY \cdot READ$	S_2
S_1	$TRY \cdot READ \cdot B \cdot \overline{M}$	S_3

续表

输入		输出
现态 Q_i^n	转换条件	次态 Q_i^{n+1}
S_2	$\overline{READ} \cdot TRY$	
S_4	1	S_4
S_1	TRY	
S_1	$\overline{TRY} \cdot READ \cdot \overline{B}$	
S_3	$READ$	S_5
S_5	1	

在图 11.4.5 中有三个箭头 (S_0 状态、 S_3 状态和 S_1 状态本身) 最终直接指向 S_1 状态框, 说明控制单元在 S_0 、 S_3 和 S_1 状态下, 根据不同转换条件的作用, 都可以转换到 S_1 状态。例如控制单元现态为 S_0 , 此时 FF_0 为 **1**, 其余触发器均为 **0**。在下一个时钟脉冲来到时, 不需要其他任何条件, 控制单元状态将转换到 S_1 状态, 所以转换条件为 **1**, 此时 FF_1 为 **1**, 其余触发器为 **0**; 控制单元在 S_1 状态时, 则 S_1 为 **1**, 当下一个时钟脉冲来到时, TRY 和 $READ$ 均为 **0**, 则控制单元保持在 S_1 状态, 这样要求 FF_1 的输入为 $S_1 \cdot \overline{TRY} \cdot \overline{READ}$, 所以转换条件为 $TRY \cdot \overline{READ}$ 。其余以此类推得出表 11.4.1。表中没有列出 S_0 为次态的情况, 因为 S_0 是直接由复位信号建立的, 另外为方便起见, 没有区别现态和次态的符号。由表 11.4.1 可以写出各触发器的输入逻辑关系式:

$$FF_0: D_0 = 0 \quad (RESET 异步置 S_0 = 1)$$

$$FF_1: D_1(S_1) = S_0 + S_1 \cdot \overline{TRY} \cdot \overline{READ} + S_3$$

$$FF_2: D_2(S_2) = S_1 \cdot \overline{TRY} \cdot READ \cdot B \cdot M + S_2 \cdot \overline{TRY} \cdot \overline{READ}$$

$$FF_3: D_3(S_3) = S_1 \cdot \overline{TRY} \cdot READ \cdot B \cdot \overline{M}$$

$$FF_4: D_4(S_4) = S_2 \cdot \overline{READ} \cdot TRY + S_5$$

$$FF_5: D_5(S_5) = S_1 \cdot TRY + S_1 \cdot \overline{TRY} \cdot READ \cdot \overline{B} + S_2 \cdot READ + S_3$$

$$= S_1 \cdot TRY + S_1 \cdot READ \cdot \overline{B} + S_2 \cdot READ + S_5$$

由图 11.4.4 所示的 ASM 图可以写出控制单元输出信号的逻辑表达式为

$$CLR = \overline{S}_0 \quad (\text{低电平有效})$$

$$CNT = S_3$$

$$OPEN = S_4$$

$$ERROR = S_5$$

根据上述逻辑方程得出数字锁的控制单元电路如图 11.4.6 所示。

图 11.4.3 给出处理单元电路的结构图。如果 D 触发器选用 74HC74。多路选择器选用

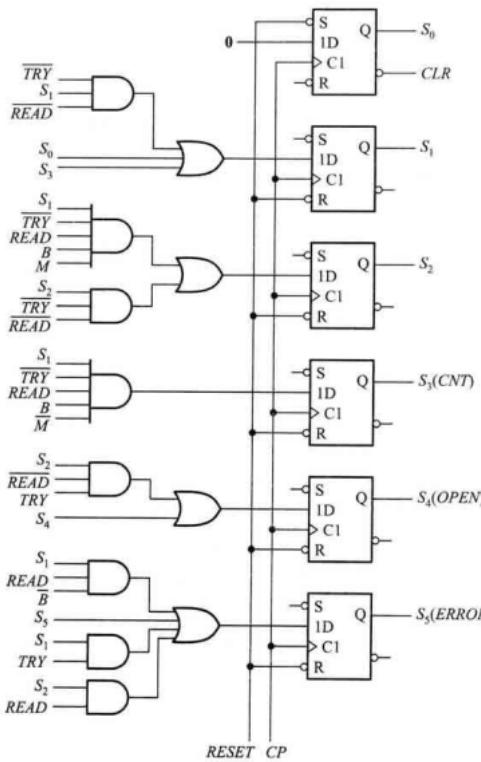


图 11.4.6 数字锁控制器电路

74HC151，比较器选用 74HC85，计数器选用 74HC163，外加必要的门电路，将控制单元和处理单元组合起来就可以构成数字密码锁系统，数字锁系统电路图略。

比较交通灯控制系统和数字密码锁两种控制单元的硬件实现方法可知，触发器状态组合方法所用的触发器少，但组合电路相对复杂。一个触发器对应一个状态所用的触发器较多，但可以简化组合电路。对于寄存器数量多，而逻辑门相对缺乏的 FPGA 器件，采用后一种方法可以提高器件资源的利用率，提高电路的速度和可靠性。

11.4.3 用可编程逻辑器件实现数字密码锁

1. 寄存器传输级描述

数字密码锁寄存器传输级的 HDL 描述如下, 它分为 3 部分, 第一部分声明了系统的输入 nRESET、TRY、READ、BIT、CLK, 输出 OPEN、ERROR, 所用的寄存器以及系统内部信号变量。第二部分说明了控制单元工作的时序关系。第三部分是处理单元的操作。

根据图 11.4.4 所示的数字密码锁控制器的 ASM 图, 可以用 HDL 描述。控制单元的 HDL 描述用两个 **always** 语句描述其时序过程。第一个 **always** 语句说明异步复位信号 nRESET 使系统进入初态 S0, 并且系统状态转换是在时钟 CLK 的上升沿进行的。第二个 **always** 语句是由 **case** 多路分支语句构成, 说明控制单元的 5 种状态, 以及状态之间的转换条件。

第三部分处理单元的 HDL 编写, 可以根据图 11.4.3 所示的结构图进行。其中第一个 **always** 语句说明计数器在控制信号 CNT 的作用下进行计数, 其输出作为选择器的选择信号。当计数器输出为 111, 即 8 位密码全部输入时, 比较器输出 M 为 1, 由控制单元判断是否开锁。第二个 **always** 语句说明选择器的工作。当选择器的输出与所输入的密码相同时 B 为 1, 否则为 0。

```
module Lock_RTL(
    /* ===== 定义输入/输出端口, 参见图 11.4.2 ===== */
    input  nRESET, TRY, READ, BIT, CLK,
    output reg OPEN, ERROR
);
//对状态进行编码
parameter S0 = 6' b000001, S1 = 6' b000010, S2 = 6' b000100, S3 = 6' b001000, S4 = 6' b010000, S5 = 6'b100000;
parameter Password = 8'b0 1011001;
//定义系统内部的信号变量及其类型, 见图 11.4.2
reg [5:0] CurrentState, NextState; //控制寄存器
reg nCLR, CNT; //CNT=1,计数器工作;CNT=0,暂停计数
wire B, M; //数据处理器输出
reg [2:0] Q; //计数器输出
wire A2,A1,A0; //数据选择器控制信号
wire [7:0] D; //数据选择器输入信号
reg Mux_out; //数据选择器输出信号
/* ===== 控制单元状态转换的描述, 参见图 11.4.4 ===== */
always @ (posedge CLK, negedge nRESET)
begin: statereg
    if (~nRESET)
        begin
            CurrentState <= S0;
            nCLR <= 1'b0;
            end
    else

```

```

begin  CurrentState <= NextState; nCLR <= 1'b1; end
end //statereg
always @ ( BIT, READ, TRY, B , M , CurrentState )
begin: fsm
  OPEN = 1'b0;
  ERROR = 1'b0;
  CNT = 1'b0;
  case ( CurrentState )
    S0: NextState = S1;
    S1: begin
      CNT = ( ~TRY & READ == 1'b1 );
      if (TRY) NextState = S5;
      else if ( ~READ) NextState = S1;
      else if ( ~B ) NextState = S5;
      else if ( M ) NextState = S2;
      else NextState = S3;
    end
    S2:begin
      if (READ) NextState = S5;
      else if ( ~TRY) NextState = S2;
      else NextState = S4;
    end
    S3: NextState = S1;
    S4:begin
      OPEN = 1'b1;
      NextState = S4;
    end
    S5:begin
      ERROR = 1'b1;
      NextState = S5;
    end
    default: NextState = S0;
  endcase
end
/* =====处理单元的描述,参见图 11.4.3===== */
always @ ( posedge CLK, negedge nCLR )
begin:counter
if ( ~nCLR) Q <= 3'b000;

```

```

else if ( CNT==1'b1 )
    Q <= Q + 1'b1;
else
    Q <= Q;
end
assign M = (Q==3'b111); //比较器输出信号
assign [A2,A1,A0] = Q;
assign D = Password; //设置密码
always @ (A2, A1, A0, D)
begin:Muxultiplexer
    case ([A2,A1,A0])
        3'd0: Mux_out = D[0];
        3'd1: Mux_out = D[1];
        3'd2: Mux_out = D[2];
        3'd3: Mux_out = D[3];
        3'd4: Mux_out = D[4];
        3'd5: Mux_out = D[5];
        3'd6: Mux_out = D[6];
        3'd7: Mux_out = D[7];
    endcase
end
assign B = (Mux_out ~^ BIT); //异或运算
endmodule

```

2. 时序仿真

在 Quartus II 9.1 软件中建立一个新的工程项目, 输入上述 HDL 文件, 对设计项目进行编译。然后建立一个仿真波形文件, 给出输入、输出信号的激励波形, 对设计项目进行时序仿真, 得到如图 11.4.7 所示的波形。

若系统内设置的密码为 **01011001**。分析波形图可知, nRESET 信号使系统首先进入初始状态。BIT 开关用于产生 1 位数码, READ 开关将 BIT 产生的当前数码读入系统, 并与系统内所设置密码的相应位进行比较, 比较的顺序是从低位到高位。当读入 8 位数码与开锁密码一致时, 按下开锁的 TRY 信号, 系统将产生高电平开锁信号 OPEN, 如图 11.4.7(a) 所示。如果开锁过程中任何一次送入的数码与设置的密码数值不一致, 如图 11.4.7(b) 中输入的第 2 位数码为 **1**, 与设置的 **0** 不符, 系统发出错误信息 ERROR。或者 TRY 信号使用不当, 也会产生 ERROR 信号。

为便于分析开锁过程中系统状态之间的转换, 图中给出了中间变量的波形图, 具体含义这里不再赘述, 读者可自行分析。

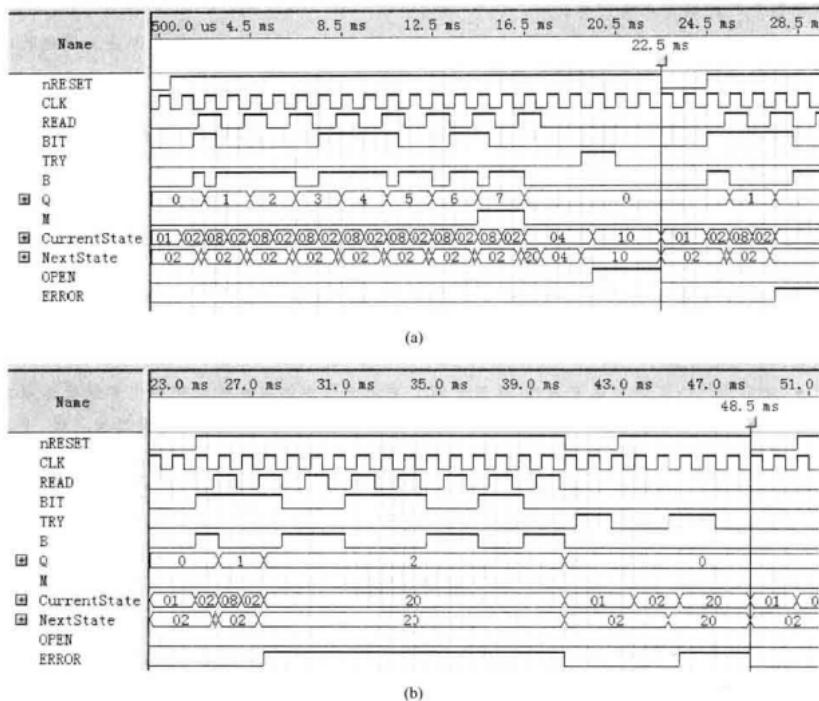


图 11.4.7 数字密码锁的仿真波形图

(a) 正确开锁过程仿真波形图 (b) 错误开锁过程仿真波形

复习思考题

11.4.1 用 Verilog HDL 描述数字系统时, 可以分为几种形式? 各种描述形式的特点是什么?

11.4.2 数字系统中, 通常借助于什么编写控制单元的 Verilog HDL 描述?

11.4.3 比较交通灯控制系统的结构级描述和寄存器传输级描述, 你得出什么结论?

小 结

- 数字系统从结构上分为数据处理单元和控制单元。因此, 数字系统中的二进制信息也分

成数据信息和控制信息两大类。

- 数字系统的设计方法有两种，自下而上的设计方法和自上而下的设计方法。现代数字系统的设计常采用自上而下的设计方法。
- 数字系统自下而上的设计是设计者根据经验将数字系统按逻辑功能划分成若干子模块，一直分到这些子模块可以用经典的方法和标准的逻辑功能部件进行设计，最后将整个系统安装、调试达到设计要求。
- 自上而下的设计方法是将系统的设计分层次、分模块进行。通常将系统分为控制器和处理器两部分。如果控制器和处理器仍比较复杂，可以在它们内部多重地分解成几个子模块进行逻辑设计，给出实现系统的硬件和软件描述，最后得到所要求的数字系统。
- ASM 图是数字系统自上而下设计方法中经常采用的设计工具，它是指数字系统控制算法的流程图，可以将语言描述的设计问题变成时序流程图的描述，从而获得状态函数或输出函数，得出相应的硬件电路。
- 数字系统的实现方法有多种，这里介绍了采用典型电路基本模块和采用可编程逻辑器件两种方法。采用典型电路基本模块是传统的、成熟的实现方法。采用可编程逻辑器件时，用户可以将设计的数字系统，用图形输入方式或硬件描述语言方式或两者的混合方式，输入到开发系统，生成编程文件，将编程文件写到芯片中。Verilog HDL 是常用的硬件描述语言，采用 Verilog HDL 语言可以简化数字系统的实现过程。

习 题

11.1 数字系统概述

11.1.1 数字电路与数字系统的根本区别是什么？大容量存储器属于数字电路还是数字系统？

11.1.2 数字系统自上而下的设计步骤是什么？

11.1.3 数字系统的实现有哪几种方法？各种方法的特点是什么？

11.2 算法状态机

11.2.1 初始状态为 T_0 的数字系统，有两个控制信号 X 和 Y ，当 $XY=10$ 时，寄存器 R 加 1，系统转到第二个状态 T_1 。如果 $XY=01$ 时，寄存器 R 清零，同时系统从 T_0 转到第三个状态 T_2 。试画出该数字系统的 ASM 图。

11.2.2 一个数字系统的数据处理单元由触发器 E 和 F 、4 位二进制计数器 A 以及必要的门电路组成。计数器的各位为 A_4, A_3, A_2, A_1 。系统开始处于初始状态，当信号 $S=0$ 时，系统保持在初始状态；当 $S=1$ 时，计数器 A 和触发器 F 清零。从下一个时钟脉冲开始，计数器进行加 1 计数，直到系统操作停止。 A_4 和 A_3 的值决定了系统的操作顺序。

当 $A_3=0$ 时，触发器 E 清零，计数器继续计数。

当 $A_3=1$ 时，触发器 E 置 1，并检测 $A_4, A_4=0$ 时，继续计数； $A_4=1$ 时，触发器 F 置 1，并停止计数，回到系统初始状态。

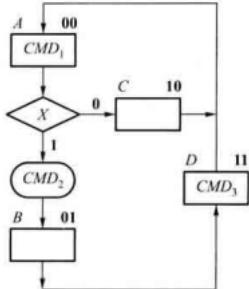
(1) 试画出该系统的 ASM 图；

(2) 画出该系统控制单元的状态图，并用 D 触发器及必要的门电路设计控制单元。

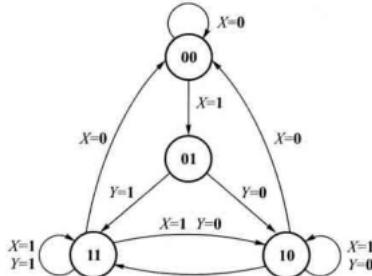
11.2.3 试画出一个序列检测器电路的 ASM 图及状态图。该电路的功能是：当输入的串行数据为 **0101**（从左到右）时，输出为 **1**，否则输出为 **0**。要求考虑序列重叠性，当输入为 **010101**，相当于出现两个 **0101** 序列。

11.2.4 用一个触发器一个状态方法实现图题 11.2.4 所示的 ASM 图。

11.2.5 一个数字系统控制单元的状态图如图题 11.2.5 所示，试画出等效的 ASM 图（状态框是空的），并用 D 触发器和数据选择器实现控制单元电路。

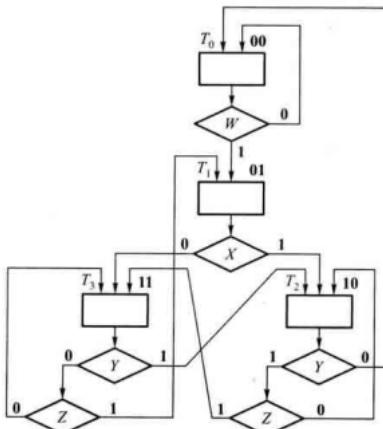


图题 11.2.4



图题 11.2.5

11.2.6 数字系统的 ASM 图如图题 11.2.6 所示，试设计系统的控制单元电路。



图题 11.2.6

11.3 交通信号灯控制系统

11.3.1 设计一个可以控制计数功能的计数器，当 $X=0$ 时，计数器以二进制数序列 **000, 001, 010, 011, 100,**

101,110,111 计数并重复。当 $X=1$ 时,计数器以格雷码序列 **000,001,011,010,110,111,101,100** 计数并重复,要求:

- (1) 画出计数器的状态图和 ASM 图;
- (2) 用 Verilog HDL 语言描述系统的工作过程。

11.3.2 设计一个用移位相加实现的乘法器,乘数与被乘数均为同步输入的 4 位无符号二进制数。要求:

- (1) 确定乘法器算法,画出乘法器系统方案框图;
- (2) 画出系统控制器的 ASM 图。用一个触发器对应一个状态的方法设计控制电路;
- (3) 用 Verilog HDL 语言描述系统的工作过程。

11.4 数字密码锁

11.4.1 设计一个简单的 3 位二进制数字密码锁,控制房门的打开,当接收到的串行输入数码与原设定的密码相同时,发出开锁信号,锁被打开。要求:

- (1) 画出系统的 ASM 图;
- (2) 画出系统的状态转换图;
- (3) 用 Verilog HDL 语言描述其工作过程。

提示:数字锁系统的示意图如图题 11.4.1 所示,由于开锁过程比较简单,所以没有划分控制单元和处理单元。 L_0, L_1, L_2 用于从低位到高位设置密码,另外三个输入端 Re, En, Ki 中, Re 是复位端,当 $Re=1$ 时,系统复位进入初始状态; En 是控制开关, Ki 是数据输入端,当按一下 En 键使 $En=1$ 时,从 Ki 端输入一个数码,再按一下 En 键,再送一个数码,直到将三个数码送完为止。每输入一个数据都要同原设定的密码比较,依次从低位到高位进行。若相等则准备接收下一位数码,若不相等,系统应进入错误状态。输入数码的位数也是开锁的条件。当输入数码的位数和位值与相应密码都相等时,系统发出开锁信号 $Un=1$,锁被打开,否则 $Er=1$ 表示开锁过程错误。为保密起见,中间错误状态不显示,并且不能返回初态,直到三个数码全部送完为止。

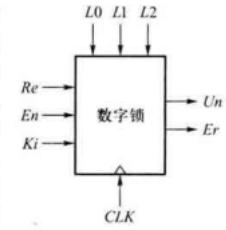
11.4.2 设计一个体育比赛中常用的数字跑表。它是通过两个按键来控制计时开始和停止,一个是清零控制按键 Reset(简称 R 键),另一个是 Start/Stop 控制按键(简称 S 键),其工作过程如下:

开始时 R 键使跑表为零初始状态。在 R 键无效的时候,按一下 S 键则计时器开始计时,在此计时状态下,按一下 S 键暂停计时,再按一下 S 键则继续计时,并且这一过程可由 S 键控制重复进行。如果在暂停状态按一下 R 键,跑表被清零。

如果在计时状态下,按一下 R 键则暂停计时,再按一下 R 键则继续计时,并且这一过程也可由 R 键控制重复进行。当按 R 键使计时暂停时,再按 S 键不起作用。

要求跑表的计时范围为 $0.01 \text{ s} \sim 59.99 \text{ s}$,计时精度为 10 ms ;跑表的输出能够直接驱动共阳极 7 段数码管显示。输入信号的频率为 100 Hz 。

- (1) 画出跑表的结构框图;
- (2) 画出控制单元的 ASM 图及状态图;
- (3) 用 Verilog HDL 描述跑表的功能。



图题 11.4.1

附录 A EDA 工具 Quartus II 9.0 简介

Quartus II 软件是由 Altera 公司开发的一个 EDA 工具, 它完全取代了该公司早期的 MAX+Plus II 软件。Quartus II 集成了设计输入、逻辑综合、布局布线、仿真验证、时序分析、器件编程等开发 FPGA 和 CPLD 器件所需要的多个软件工具。

为了支持教育事业和培养潜在的用户, Altera 公司设立了大学计划项目, 为大学生免费提供 Quartus II 的网络版软件。读者可以登陆公司网站下载该软件, 其网址为: <http://www.altera.com/> 或 <http://www.altera.com.cn/>。网络版软件的功能会受到部分限制, 只支持部分器件, 但不需要许可证文件就可以免费使用, 方便初学者使用该软件。

随着 Altera 公司器件集成度的提高、器件结构和性能的改进, Quartus II 软件也在不断地改进和更新之中, 到目前为止已推出若干个版本。本附录使用 2009 年推出的 Quartus II 9.0 版本, 并将其安装在运行微软公司的 Windows XP 操作系统的计算机上。由于 Quartus II 软件规模大, 功能强, 本附录旨在帮助读者迅速入门, 不能详细介绍其使用。读者可以查阅该软件的在线帮助 (Help) 文档进一步获得许多高级功能的使用方法。

A.1 Quartus II 9.0 软件主界面

启动装有 Quartus II 9.0 软件的计算机后, 用鼠标左键单击桌面左下角“开始”菜单项目中的“程序 | Altera | Quartus II 9.0 | Quartus II 9.0 (32-Bit)”命令后, Quartus II 开始运行, 屏幕上出现图 A.1.1 所示的主界面。该主界面由多个窗口组成, 用户可以通过菜单 View | Utility Windows 命令选取 Quartus II 的组成窗口, 从而改变主界面的形式。

主界面的顶部是标题栏, 标题栏下面有许多菜单, 通过这些菜单可以选用 Quartus II 提供的绝大多数命令。许多常用命令以图标形式显示在快捷工具栏上, 选择 Tools | Customize | Toolbars 命令可以定制工具栏。当鼠标光标放置到某个图标上, 便显示出与该图标关联命令的名字。

通过 Quartus II 主窗口的 Help 菜单可以访问在线帮助文档, 该帮助文档能回答用户在使用该软件时可能遇到的大多数问题。

另外, 安装 Quartus II 软件时, 在安装子目录(例如, C:\altera\90\qdesigns)下面有几个设计项目示例, 供用户参考。

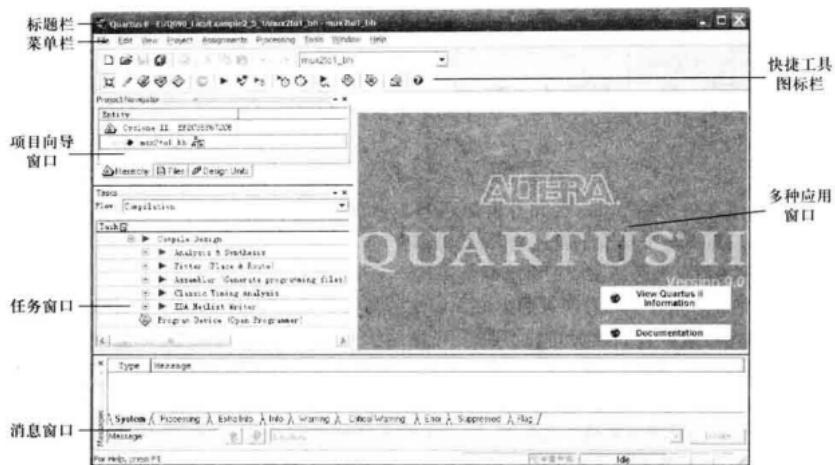


图 A.1.1 Quartus II 主界面

A.2 Quartus II 的设计流程

8.3 节简单介绍了可编程逻辑器件的开发过程,本节将结合图 A.2.1 来介绍使用 Quartus II 软件的设计流程。大体上可以分为以下 6 个步骤:

(1) 创建一个新项目,并为此项目指定一个工作目录,然后指定一个目标器件。

在用 Quartus II 进行设计时,将每个逻辑电路或者子电路称为项目(project)。当软件对项目进行编译处理时,将产生的一系列文件(例如,电路网表文件、编程文件、报告文件等)。因此需要创建一个目录用于放置设计文件以及设计过程产生的一些中间文件。我们创建目录 E:\QIH90_Lab,以便保存本教材的设计文件。建议每个项目使用一个目录。

注意,目录的位置可以任意选择,但不能将设计文件直接放在根目录下,目录或者文件的名字不能使用汉字,最好使用英文字母或下划线开头,后面跟字母或数字。

(2) 设计输入。Quartus II 可以使用的设计输入文件有:电路原理图、Verilog HDL 以及其他硬件描述语言文件,例如 VHDL 和 AHDL(Altera 公司专用的硬件描述语言),也可以选用状态

机文件和 EDIF(Electronic Design Interface Format, 电子设计接口格式)文件^①等。

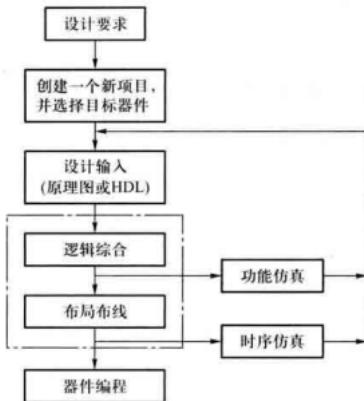


图 A.2.1 Quartus II 设计流程

(3) 逻辑综合。就是把原始描述(原理图或 HDL 代码)转换成面向某个具体的 FPGA 器件的电路网表文件,即用目标芯片中的逻辑元件来实现设计的逻辑,供后面的布局布线软件使用。Quartus II 软件内部的集成综合工具支持 Verilog-1995 和 Verilog-2001 的 IEEE 标准,还支持 VHDL 1987 和 VHDL 1993 标准。

(4) 布局布线。根据事先设定的约束条件(例如,器件型号、指定的输入/输出引脚、电路工作频率等),将逻辑综合器生成的网表文件输入到布局布线器,然后用目标芯片中某具体位置的逻辑资源(元件、连线)去实现设计的逻辑,完成逻辑元件、引脚的布局以及连线工作。同时生成一系列中间文件(例如,供时序仿真用的电路网表文件、报告文件等)和编程数据文件(.sof 和 .pof)。

在 Quartus II 软件中,将逻辑综合、布局布线等软件集成在一起,称为编译工具。在 Quartus II 主界面,使用菜单 Processing | Compiler Tool 命令,弹出图 A.2.2 所示的编译器窗口,该窗口包含了对设计文件进行处理的四个模块。

Analysis & Synthesis(分析和综合)模块对设计文件进行语法检查、设计规则检查和逻辑综合。综合过程分两步:第一步是将 HDL 语言翻译成逻辑表达式。第二步是进行工艺技术映射,即用目标芯片中的逻辑元件来实现每个逻辑表达式。

Fitter(电路适配器)模块的功能是用目标芯片中某具体位置的逻辑资源(元件、连线)去实现

^① EDIF 文件是由第三方综合工具生成的表示电路逻辑结构的标准格式文件。该 EDIF 标准为 EDA 工具之间交换信息提供了一个便利的机制。

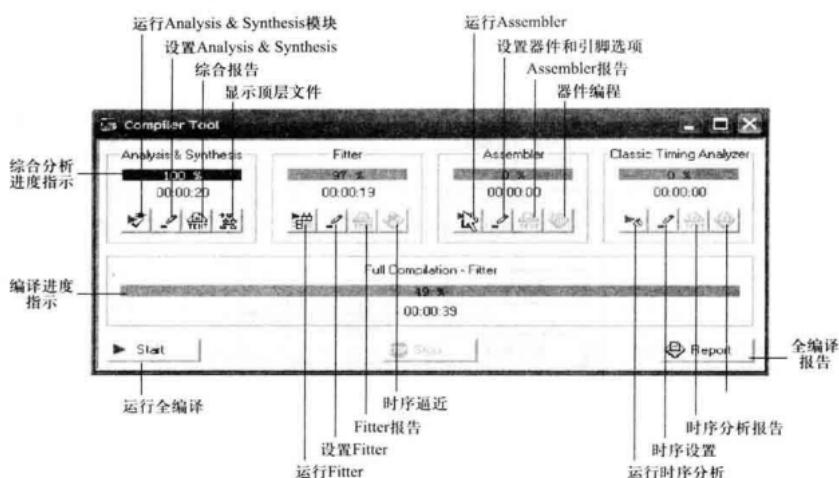


图 A.2.2 Quartus II 的编译器窗口

设计的逻辑,完成布局布线的工作。

Assembler(装配)模块产生多种形式的编程数据文件(包括.pof,.sof等)。

Classic Timing Analyzer(经典的时序分析^①)用于分析逻辑设计的性能,并指导电路适配器工作,以满足设计项目中定时要求。默认情况下,该模块作为全程编译的一部分将会自动运行,并分析和报告器件内部逻辑电路各路径的定时信息。

启动编译器运行的方法多种多样。例如,在图 A.2.2 上,分别点击 4 个模块左下角的按钮,将运行相应的模块。如果点击最左下角的 Start 按钮,将按图中顺序运行四个模块(全程编译)。当设计项目较大,全程编译时间很长时,根据设计流程中的某一特定步骤,只运行对应的模块。例如,需要功能仿真时,只运行 Analysis & Synthesis 就可以了,没有必要进行全程编译。

第二种启动编译器运行的方法是:在 Quartus II 主界面,选择 Processing | Start 菜单下面的命令,该菜单下面许多命令的功能与图 A.2.2 中的相同。

第三种启动编译器运行的方法是:在 Quartus II 主界面,选择菜单 Processing | Start Compilation 命令,或点击工具栏上的▶快捷图标,启动全程编译运行,该命令与点击图 A.2.2 中的 Start 按钮等价。

(5) 仿真验证。仿真的目的是验证设计的电路能否达到预期的要求。Quartus II 软件支持

^① 在 Quartus II 10.0 及以后的版本中,去掉了 Classic Timing Analyzer 工具,只有 TimeQuest Timing Analyzer(静态时序分析器)。

功能仿真和时序仿真两种方式。

功能仿真(functional simulation)就是假设逻辑单元电路和互相连接的导线是理想的,电路中没有任何信号的传播延迟,从功能上验证设计的电路是否达到预期要求。仿真结果一般为输出波形和文本形式的报告文件,从波形中可以观察到各个节点信号的变化情况。但波形只能反映功能,不能反映定时关系。在进行功能仿真之前,需要完成3项准备工作:

- 对设计文件进行部分编译(分析和综合);
- 产生功能仿真所需要的网表文件;
- 建立输入信号的激励波形文件。

时序仿真(timing simulation)是在布局布线完成后,根据信号传输的实际延迟时间进行的逻辑功能测试,并分析逻辑设计在目标器件中最差情况下的时序关系,它和器件的实际工作情况基本一致,因此时序仿真对整个设计项目的时序关系以及性能评估是非常必要的。

(6) 器件编程。将编译得到的编程数据文件下载到目标器件中,使该可编程器件能够完成预定的功能,成为一个专用的集成电路芯片。

编程数据是在计算机上编程软件的控制下,由下载电缆传到FPGA器件的编程接口,然后再对器件内部的逻辑单元进行配置。常用的下载电缆有:USB-Blaster、ByteBlaster II 和 Ethernet Blaster等,USB-Blaster 使用计算机的 USB 口,ByteBlaster II 使用计算机的并行口,Ethernet Blaster 使用计算机的以太网口,在使用之前,都需要安装驱动程序。下面以安装 USB-Blaster 驱动程序为例,介绍其安装过程。

如果没有安装 USB-Blaster 驱动程序,当连接好 USB-Blaster 下载电缆并接通电源时,Windows 系统将会弹出如图 A.2.3 所示的对话框,选择“从列表或指定位置安装”,单击“下一步”按钮。



图 A.2.3 安装 USB-Blaster 驱动程序向导

弹出如图 A.2.4 所示的对话框,单击“浏览”按钮,选择驱动程序所在的子目录(位于 Quartus II 软件的安装目录下,例如,C:\altera\90\quartus\drivers\usb-blaster),再单击“下一步”按钮,即可完成硬件驱动程序的安装。

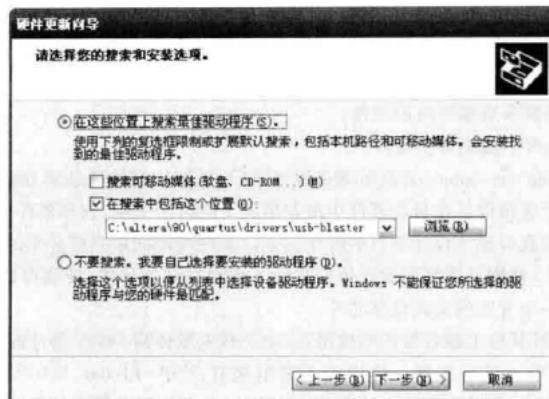


图 A.2.4 指定驱动程序所在的子目录

A.3 设计与仿真的过程

下面以 Verilog HDL 语言设计一个 2 选 1 数据选择器为例,介绍使用 Quartus II 软件进行设计输入与仿真验证的过程。介绍时,首先说明每个步骤涉及的基本知识,然后给出操作步骤。受篇幅所限,原理图(或称为图形块)输入方式不作介绍。

A.3.1 建立新的设计项目

在 Quartus II 中,创建一个新设计项目(design project)的方法有两种:(1)利用项目向导(wizard),先创建一个新项目,然后为此项目准备设计输入文件;(2)先输入设计文件,然后在保存文件时,将其指定为一个新项目。这里介绍第一种方法。

创建一个新的项目大致要经过设定工作目录(本例为 E:\QII90_Lab\Example2_5_1)和项目名称、添加文件到本项目、选择器件型号、指定所需的第 3 方工具等几个步骤,具体操作如下:

① 在 Quartus II 的主界面,选择主菜单 File | New Project Wizard 进入向导启动界面,点击 Next,出现图 A.3.1 所示的窗口。按照提示输入设计项目的工作目录、项目名称以及顶层文件名称。点击 Next 按钮进入第 2 页面。

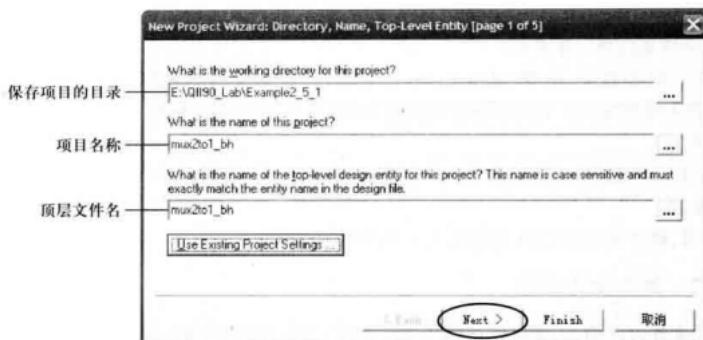


图 A.3.1 指定设计项目的目录和名字

注意,Quartus II 建议将项目名 mux2to1_bh 作为该项目顶层文件的名字。但是用户也可以另外再起一个不同的名字,只要忽略软件提出的建议即可。

② 第 2 页面用于将已经存在的文件添加到当前工程项目中。本项目不添加文件,直接单击“Next”按钮进入第 3 页。

③ 第 3 页面如图 A.3.2 所示,选择将要使用的目标器件。首先要选择使用的目标器件所属

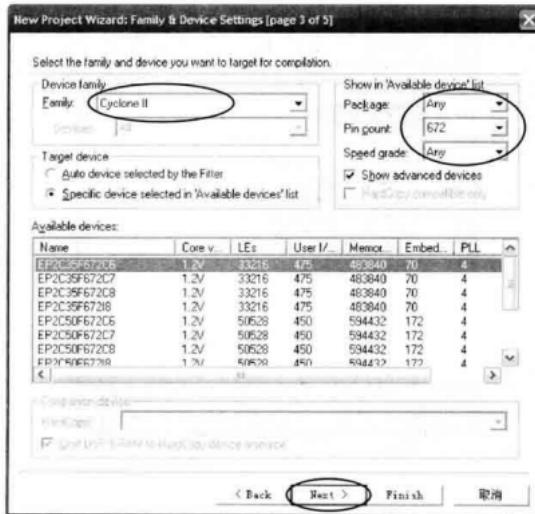


图 A.3.2 指定器件系列及型号

系列,此处选定 Cyclone II 系列的 EP2C35F672C6 器件,然后单击“Next”按钮进入第 4 页面。否则,编译器会自动选择一个芯片。

芯片命名规则如下:EP2C 是指 Cyclone II 系列,35 表明该芯片中逻辑单元的个数,编号 F672 表明其采用细线 672 引脚球格阵列封装,C6 是指芯片的速度等级。

④ 第 4 页面用于选择使用第三方软件工具^①,此例只使用 Quartus II 软件包集成的工具,不选用其他工具。点击 Next,进入最后一个页面,该页面显示设计项目的摘要(Summary)。检查全部参数设置,若有误,可单击“Back”按钮返回,重新设置。若无误,则单击“Finish”,返回到 Quartus II 主界面,此时 mux2to1_bh 被指定为一个新项目。

A.3.2 输入设计文件

对设计文件进行输入,需要经过选择文件类型、打开编辑器窗口、再输入文件(键入 HDL 代码或画原理图)等步骤,具体操作如下:

① 在 Quartus II 主界面,选择 File | New... 命令,出现如图 A.3.3 所示的窗口,在 Design Files(设计文件)栏目下,选择 Verilog HDL File,点击 OK,打开文本编辑器。

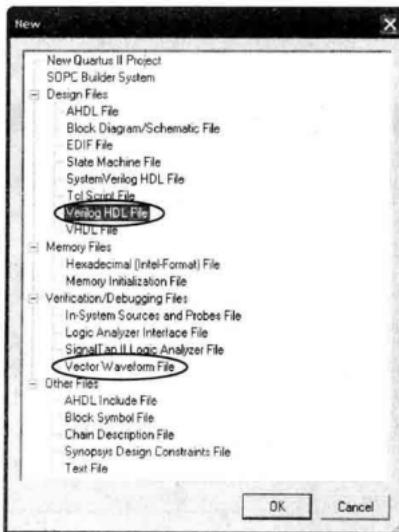


图 A.3.3 选择设计文件的类型

^① Quartus II 10.0 及以后版本的软件,去掉了内嵌的仿真工具,必须使用如 ModelSim 等第三方的仿真工具。

② 保存文件。选择 File | Save As, 弹出一个对话框, 在“保存类型”下拉列表框内选择 Verilog HDL File, 在“文件名”列表框内键入 mux2to1_bh.v, 并单击最底下 Add file to current project (添加文件到当前项目) 前面的方框使其出现对勾, 保存该文件。

③ 键入 HDL 代码。以例 A.5.1 中的 Verilog 代码为例, 在文本编辑器窗口键入代码, 用 File | Save 或者快捷键 Ctrl-s 保存该文件, 如图 A.3.4 所示。

```

1 module mux2to1_bh( //Verilog 2001,2005 module port syntax
2   input D0,D1,S, // Input Ports
3   output reg Y; // Output Ports
4 );
5
6   always@(*)
7     begin
8       if(S)
9         Y = D1;
10      else
11        Y = D0;
12    end
13  endmodule

```

图 A.3.4 在文本编辑器中键入 Verilog 代码

④ 选择命令 Processing | Analyze Current File, 对编辑好的 HDL 程序进行语法检查。

A.3.3 编译设计文件

Quartus II 编译器主要完成设计项目的检查、逻辑综合、布局布线等任务, 为项目的时序仿真生成含有延时信息的电路网表文件, 并生成最终的编程数据文件。其操作步骤是: 在 Quartus II 主界面, 选择菜单 Processing | Start Compilation 命令, 或点击工具栏上的▶ 快捷图标, 启动全程编译运行。

在编译进行过程中, Quartus II 主界面左边的 Task(任务)窗口中将显示整个编译进程、各个模块编译进程的进度以及所用的时间; Messages(消息)窗口将显示编译过程中的消息以及设计中出现的错误等。若输入的 Verilog 代码存在错误, 则显示每个错误。双击错误消息, 在文本编辑器中高亮显示相应的出错语句。用同样的方法, 也可以找到编译过程的警告消息对应的 Verilog 源代码行。当选定某信息, 按 F1 键, 得到关于该错误或者警告消息的更多信息。否则, 显示编译成功的消息。

编译完成后, 会自动出现图 A.3.5 所示的编译报告窗口, 选择左边窗口中要查看的条目, 相应的报告内容会在右边窗口显示出来。

若编译报告窗口没有打开, 则在编译工具窗口中点击 Report 图标, 即可打开编译报告。另外, 还可以用 Processing | Compilation Report 打开编译报告。

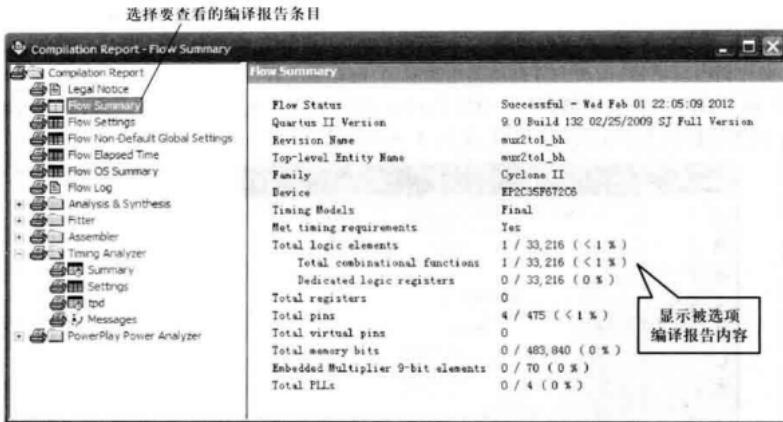


图 A.3.5 编译报告窗口

A.3.4 设计项目的仿真验证

在对电路进行仿真之前,需要做好以下准备工作:

(1) 准备好电路网表文件。这里分两种情况:如果进行功能仿真,则使用命令 Processing | Generate Functional Simulation Netlist,生成功能仿真网表文件。如果进行时序仿真,则需要对整个设计进行全程编译,生成时序仿真网表文件。

(2) 准备好测试向量文件。用 Quartus II 波形编辑器(Vector/Waveform Editor)建立输入信号的激励波形(即测试向量),并以波形文件(后缀名为.vwf)形式保存。

具体操作如下:

① 打开波形编辑器,建立一个新的测试向量波形文件。

在 Quartus II 主界面,选择 File | New 命令,弹出前面图 A.3.3 所示的选择文件类型对话框。选择 Vector Waveform File(向量波形文件),单击 OK 按钮,波形编辑器窗口如图 A.3.6 所示。用 mx2to1_bh.vwf 作为文件名保存该文件。

设置仿真时间 0~100 ns。选择命令 Edit | End Time,在弹出的对话框中,将默认的仿真时间 1 μs 改成 100 ns。选择 View | Fit in Window,可在该窗口中显示整个仿真过程。

设置栅格尺寸。栅格就是图中的垂直参照虚线。选择 Edit | Grid Size(栅格尺寸),在弹出的对话框中输入 5.0 ns。注意,栅格的尺寸必须小于仿真文件的时间长度。

② 在测试向量文件中,添加输入、输出节点(信号)名。

在波形编辑器窗口中,选择 Edit | Insert | Insert Node or Bus...,或者用鼠标左键双击左边

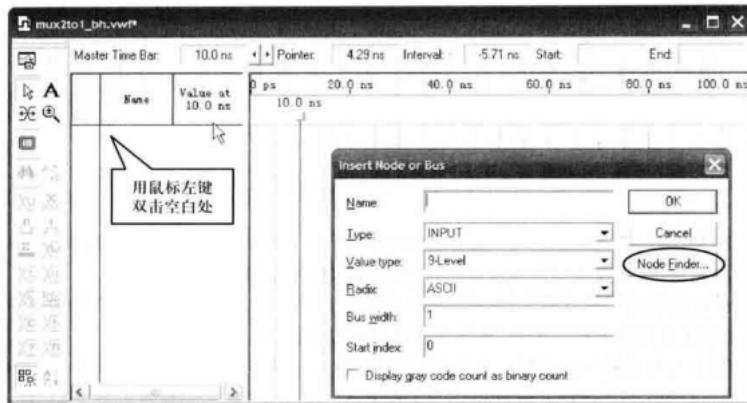


图 A.3.6 波形编辑器窗口

Name 列的空白处，打开如图 A.3.6 所示的 Insert Node or Bus(插入节点或总线)对话框。

单击 Node Finder... (节点寻找)按钮，打开如图 A.3.7 所示的窗口。在 Filter(过滤器)栏目内选择 Pins : all。点击 List 按钮，在窗口的左边显示节点 D0、D1、S 和 Y。选择 D0，接着点击 \geq 按钮，将 D0 添加到右边的列表框中。对 D1、S 和 Y，按照同样方法处理。或者，直接单击 \gg 按钮，将所有节点添加到右边的列表框中。点击 OK 关闭节点寻找实用程序。接着点击图 A.3.6 中的 OK，回到波形编辑器窗口。

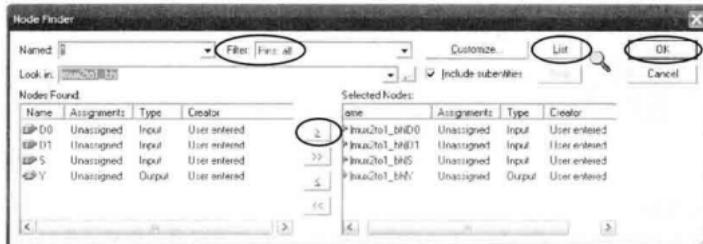


图 A.3.7 节点寻找实用程序

③ 绘制输入信号(节点)波形，即指定输入节点的逻辑电平变化。

波形编辑器提供了绘图工具栏。选择 Tools | Customize Waveform Editor 打开工具栏，然后单击 Waveform Editor 前面的方框使其出现对勾，再单击“确定”按钮，启动工具栏。工具栏各按钮的功能如图 A.3.8 所示，使用这些工具绘制波形。



图 A.3.8 绘图快捷工具栏按钮的功能

对输入 D0、D1 和 S 施加所有可能的 8 种逻辑值，输出信号的逻辑值将由仿真器自动生成。当电路比较复杂，输入取值数目非常多时，可以选取具有代表性的输入值。

输入任意信号波形的方法是：在波形的起点按下鼠标左键不放，并拖动到需要编辑的区域末尾，再单击快捷工具栏上相应按钮，即可完成输入波形编辑。

按照图 A.3.9 编辑输入波形。开始时，所有的输入都是 0。D0 的值是每隔 10 ns 变化一次，在 10~20 ns 区域，按下鼠标左键不放并拖动高亮显示该区域，单击左侧高电平按钮 L ，此时该区域变为 1。用同样的方法，将 D0、D1 和 S 相应的区域设置为 1。

最后使用默认的文件名 (mux2to1_bh.vwf) 保存该文件。

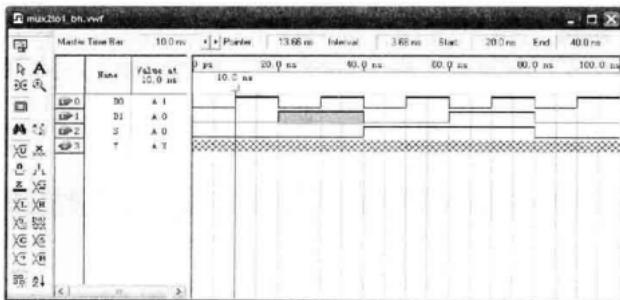


图 A.3.9 完整的测试向量波形

④ 执行仿真。

选择命令 Processing | Simulator Tool，弹出如图 A.3.10 所示设置仿真模式的对话框。

首先进行功能仿真。在 Simulation mode(仿真模式)栏目中选择 Functional，在 Simulation input 栏目内选择测试向量文件 mux2to1_bh.vwf，再单击右边的 Generate Functional Simulation Netlist 按钮，产生功能仿真网表。最后单击左下角的 Start 按钮，仿真器开始运行。同时，状态窗口显示仿真进度以及所用时间。

仿真结束后，仿真器将根据输入测试向量产生输出节点的波形。单击右下角的 Report 按钮，得到的仿真报告如图 A.3.11 所示。



图 A.3.10 指定仿真模式

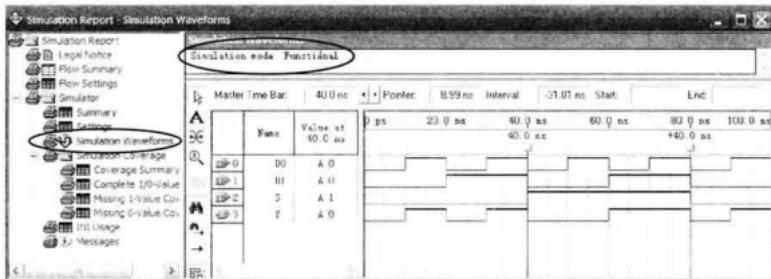


图 A.3.11 功能仿真波形图

接着进行时序仿真。在仿真开始前,必须对整个设计进行全程编译,产生时序仿真网表文件。在图 A.3.10 中的 Simulation mode(仿真模式)栏目中选择 Timing,其他选项及操作与功能仿真相同,得到的仿真报告如图 A.3.12 所示。与功能仿真波形图进行比较,输出信号 Y 的波形有些不同,即相对于输入有延迟,这是由于芯片内部逻辑元件和连接导线的延迟特性引起的。

在波形窗口中,选择命令 Edit | Time bar(时间游标条),可以插入多根时间游标的垂直线。选择命令 View | Snap to Transition(捕捉到跳变沿),然后用鼠标拖拉游标条就可以准确地对齐任何波形的跳变沿。单击 Master Time Bar(主时间游标条)垂直线顶点并拖拉到 Y 值最初跳变

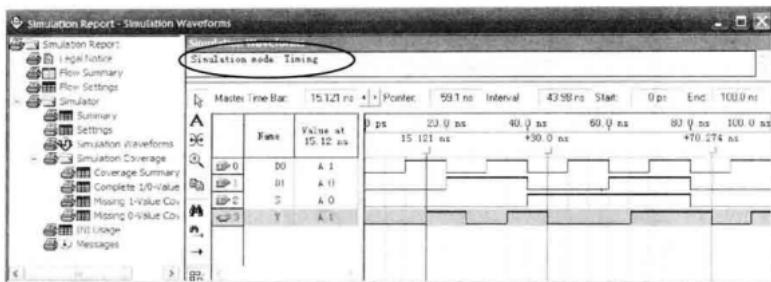


图 A.3.12 时序仿真波形图

为高电平 1 的跳变沿处,此时游标条顶部显示 15.121 ns,表示输入 D0 在 10ns 时刻发生变化,经过 5.121ns 器件延时,输出 Y 值才产生变化。

另一种启动仿真器运行的方法是:完成仿真器的设置后,在 Quartus II 主界面,选择 Processing | Start Simulation 命令,或者单击仿真快捷图标 ,即可运行仿真器。

A.4 引脚分配与器件编程

在选定目标器件、完成设计项目的分析和综合并得到正确的仿真结果以后,接着要进行引脚分配与器件编程等物理实现方面。这里用 DE2 教学开发板^①上的开关、发光二极管(LED)或者数码显示器等外围资源验证设计的正确性。

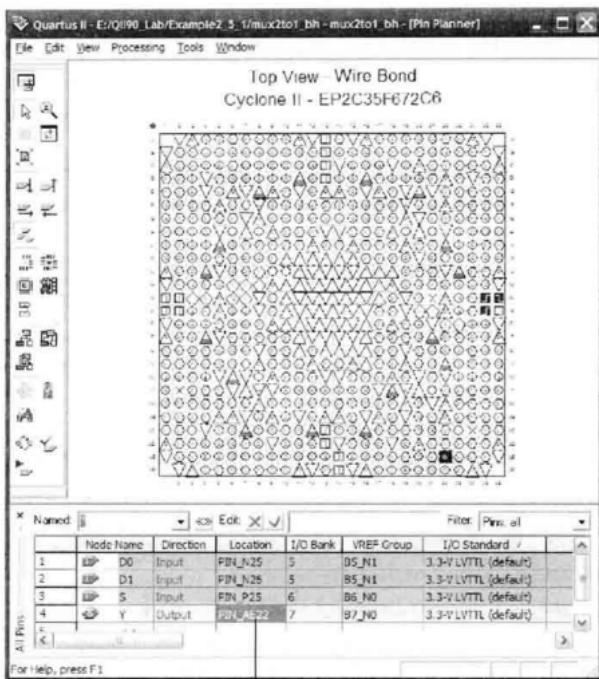
A.4.1 引脚分配

对设计文件中的输入、输出端口指定具体器件的引脚号码,称为引脚分配或引脚锁定。DE2 开发板上 FPGA 芯片的型号为 EP2C35F672C6。因此必须按照 DE2 说明书给设计文件中的输入、输出端口分配引脚。

使用引脚布局工具(Pin Planner tool)可以看到封装的引脚编号与布局。在 Quartus II 主界面,选择 Assignments | Pin Planner(引脚布局器),打开如图 A.4.1 所示 EP2C35F672C6 芯片顶视封装示意图。

EP2C35F672C6 芯片有 672 个引脚,用行和列标记,行用字母表示,列用数字表示。例如,最上面一行第 5 列的引脚称为 A5,最下面一行第 5 列的引脚称为 AF5。

^① DE2 是友晶科技有限公司生产的 FPGA 教学开发板,该公司网址:www.terasic.com.cn。



用鼠标左键双击该单元

图 A.4.1 芯片布局器上显示的引脚

使用 DE2 开发板上 3 只乒乓开关 (SW[0]、SW[1] 和 SW[2]) 和 1 只发光二极管 (LEDG[0]) 来实际测试我们的设计。电路端口与器件引脚对应关系如表 A.4.1 所示。

表 A.4.1 电路端口与器件引脚的对应关系

电路端口名	器件引脚编号	该引脚与 DE2 板上相连的元件名称
D0	PIN_N25	乒乓开关 SW[0]
D1	PIN_N26	乒乓开关 SW[1]
S	PIN_P25	乒乓开关 SW[2]
Y	PIN_AE22	绿色发光二极管 LEDG[0]

图 A.4.1 下部的表格列出了项目的输入和输出端口,为了连接输入信号 D0, 双击该表 Location 列, 从显示的清单中选择引脚 PIN_N25。重复该过程完成所有引脚分配。删除已分配引脚的方法是: 选择该引脚, 按一下键盘上的 Delete 键。

另外, 选择菜单 Assignments | Assignments Editor 命令, 或选择菜单 Assignments | Pins 命令, 也可以分配引脚。

引脚分配完毕后, 必须对设计项目再次进行全程编译。适配器(Fitter)将用户指定的引脚分配给相应的端口, 而其他未指定引脚的端口, 则由软件自动分配引脚。

上述方法适合于电路端口数较少的设计, 当电路端口较多时, 可以采用下述方法:

- (1) 用普通文本编辑器(例如记事本)创建一个以逗号分隔的文本文件, 如图 A.4.2 所示。

图 A.4.2 引脚分配文本文件

- (2) 将文件以 mux2to1_bh.csv 名字保存。文件名后缀.csv (comma separated value) 的含义是数据之间以逗号分隔。

- (3) 在 Quartus II 主界面, 选择命令 Assignments | Import Assignments, 出现图 A.4.3 所示窗口, 在该窗口中指定需要输入的文件名(本例是 mux2to1_bh.csv), 则完成了引脚分配。

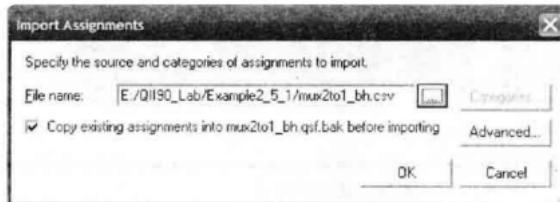


图 A.4.3 输入引脚分配文件

A.4.2 对目标器件编程

Quartus II 软件支持多种编程模式: 被动串行模式 PS(Passive Serial mode)、JTAG 模式、主动串行模式 AS(Active Serial mode) 和套接字内编程模式(In-Socket Programming mode)。在调试阶段, 一般采用 JTAG 模式编程, 将编程数据从计算机上直接下载到 FPGA 芯片的 SRAM 中。这种方法下载速度快, 便于调试, 只要电源持续供电, FPGA 将一直保留这次配置信息, 但断电后配置信息就会立即丢失。当设计成功后, 多采用 AS 模式编程, 该模式将编程数据下载到 FPGA 专用串行配置器件(例如, EPCS1、EPCS4、EPCS16)中, 断电后配置数据不会丢失, 在系统上电时, 由配置器件自动地对 FPGA 器件进行配置。

对 DE2 板上的 EP2C35F672C6 芯片进行编程之前, 需要完成以下准备工作:

to, location
D0, PIN_N25
D1, PIN_N26
S, PIN_P25
Y, PIN_AE22

(1) 用一条电缆连接 DE2 板最上边靠左的 USB 接口与计算机 USB 接口。

(2) 用专用电源适配器给 DE2 板提供直流电源(9V)。

(3) 安装 USB-Blaster 驱动程序(参考 A.2 节)。

1. 使用 JTAG 编程模式, 对 FPGA 器件编程

其操作步骤如下:

① 将 DE2 板上的 RUN/PROG 开关设置在 RUN 上。

② 选择 Tools | Programmer 命令, 出现如图 A.4.4 所示编程器窗口。此时, 编程数据文件名 mux2to1_bh.sof 及目标器件等信息显示在文件列表中。否则, 选择菜单 Edit | Add File... 命令, 添加该文件, 并单击 Program/Configure 下面的小方框, 选中编程操作。



图 A.4.4 编程器窗口

③ 指定编程硬件和编程模式。在图 A.4.4 中, 在 Mode 下拉列表框中选择 JTAG。单击左边的 Hardware Setup(硬件设置)按钮, 在弹出的窗口(如图 A.4.5 所示)中选择 USB-Blaster, 单击 Add Hardware 按钮, 再单击 Close, 返回编程器窗口。

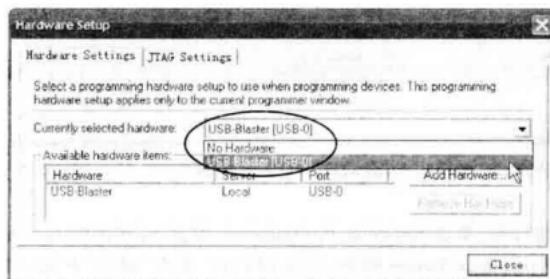


图 A.4.5 编程硬件设置窗口

④ 单击图 A.4.4 窗口中的 Start(启动)按钮。开始编程, 编程结束时有提示信息出现。若有错误报告, 表明编程失败, 则需要检查硬件连接及电源等。

2. 实际测试电路功能

完成配置数据下载后, 需要测试电路。将代表电路输入的三个开关 SW[0]、SW[1] 和 SW[2] 的状态分别设置 8 种取值之一, 观察代表电路输出的发光二极管 LEDG[0] 状态, 看是否满足 2 选 1 数据选择器的逻辑功能。

若电路工作不正常, 需要确认引脚分配是否正确。若用户想要对设计电路做一些修改, 则首先关闭编程器窗口, 然后修改 Verilog 文件, 重新全程编译, 产生新的编程数据文件, 对开发板重新编程。

3. 使用 AS 编程模式, 对配置器件 EPCS16 进行编程

其操作步骤如下:

① 在 Quartus II 主界面, 选择命令 Assignments → Settings, 在左边 Category(类别)栏中选择 Device, 打开设置器件窗口, 如图 A.4.6 所示。

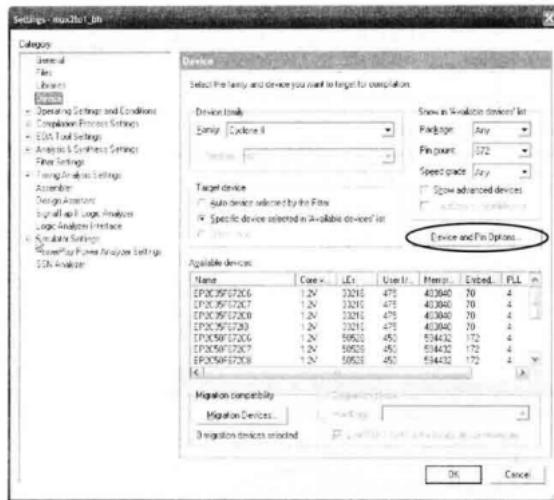


图 A.4.6 设置器件窗口

② 在 Settings 窗口中, 单击 Device & Pin Options... 按钮, 并切换到 Configuration 页面, 如图 A.4.7 所示。在 Configuration Device 框中, 选择 EPCS16, 单击“确定”按钮, 再单击 OK, 返回到 Quartus II 主界面, 重新全程编译整个项目。

③ 将 DE2 板上的 RUN/PROG 开关设置在 PROG 上。



图 A.4.7 配置器件 EPCS16 选择窗口

④ 选择 Tools | Programmer 命令, 出现如图 A.4.4 所示的编程器窗口。在 Mode 下拉列表框中选择 Active Serial Programming, 弹出一个是否清除现有器件的提示信息, 选择“是”, 清除当前器件。

⑤ 选择编程器窗口的菜单 Edit | Add File... 命令, 添加文件 mux2to1_bh.pof。然后选中 Program/Configure 编程操作。

⑥ 在编程器窗口, 单击 Start 按钮进行编程。

编程结束后, 断开电源, 并将 DE2 板上的 RUN/PROG 开关设置在 RUN 上, 重新上电, 测试电路的功能。

附录 B 电气简图用图形符号——二进制逻辑单元^①(GB/T 4728.12—1996)简介

《电气简图用图形符号——二进制逻辑单元》(GB/T 4728.12—1996)是由国家标准局颁布的用于绘制二进制逻辑单元电路的符号标准。该标准是绘制电路图、功能图、概略图等功能性简图的依据,是电气技术的工程语言。

B.1 二进制逻辑单元图形符号的组成

二进制逻辑单元图形符号是由一个方框(或几个方框的组合)和标注其上的一个或多个限定性符号组成,如图B.1.1所示。方框的*大小和长宽比可以是任意的。图中**为总限定符号,用来说明该逻辑单元所完成的逻辑功能。框内的表示与输入和输出有关的限定符号。标注在方框外的字母和其他字符不是符号的组成部分,仅用于区分多个输入和输出。通常输入线画在方框的左边或上边;输出线画在方框的右边或下边,以使信息流方向是从左到右,或从上到下。

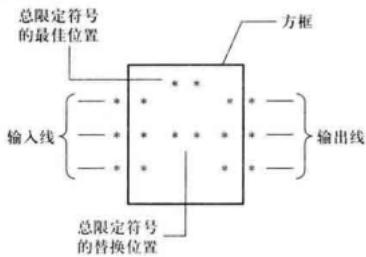


图 B.1.1 符号的组成

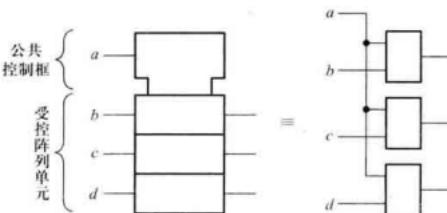


图 B.1.2 公共控制框

除上述基本单元框外,为减小图形所占的篇幅,还可以使用公共控制框和公共输出单元框。图B.1.2所示为公共控制框举例。图中a不加任何限定符号时,表示它同时加到公共控制框下面每个受控的阵列单元上。

① 应为二值逻辑单元。

图 B.1.3 所示为公共输出单元框举例。图中公共输出单元的输出与阵列中每一个单元都有连接。此外，公共输出单元也可以有其他外输入，如图中输入信号 a 。公共输出单元的逻辑功能需另外标注限定符号加以说明。

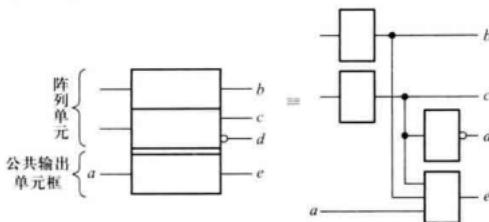


图 B.1.3 公共输出框

为了缩小图形所占的篇幅，可以将相邻的单元框组合在一起。组合的方式有邻接法和镶嵌法。在有邻接单元或镶嵌单元的符号中，如果单元框之间的公共线是沿着信息流方向，就表明这些单元之间无逻辑连接；如果单元框之间的公共线是垂直于信息流方向，则表明单元之间至少有一种逻辑连接，如图 B.1.4(a), (b) 所示。

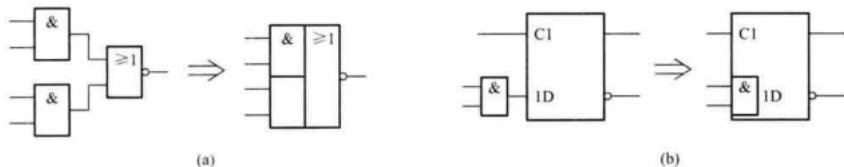


图 B.1.4 邻接法与相嵌法

(a) 邻接法 (b) 嵌入法

B.2 限定性符号

限定性符号标注在逻辑单元图形符号的方框中，用以表明该单元的逻辑功能，或者是输入或输出的物理特性或逻辑特性。限定性符号大体可以分为总限定符号，以及与输入、输出有关的限定符号等等。

1. 总限定符号

总限定符号用来表示逻辑单元总的逻辑功能。这里的逻辑功能是指符号框内部输入、输出之间的逻辑关系。表 B.2.1 列出了部分总限定符号及其所表达的逻辑功能。

表 B.2.1 部分总限定符号

符号	说明	符号	说明
&	与	CPG	先行(超前)进位
≥ 1	或	MUX	多路选择
I	缓冲	DX 或 DMUX	多路分配
=1	异或	X/Y	编码、译码或代码转换
=	逻辑恒等(所有输入状态相同时，输出才为 1 状态)	I=0	触发器的初始状态为 0
2k	偶数(输入 I 的个数为偶数时，输出为 1 状态)	I=1	触发器的初始状态为 1
2k+1	奇数(输入 I 的个数为奇数时，输出为 1 状态)	上升沿	不可重复触发的单稳态电路
▷	缓冲/驱动	下降沿	可重复触发的单稳态电路
•J	迟滞特性		非稳态电路
Σ	加法运算	SRGm	m 位的移位寄存
P-Q	减法运算	CTRm	循环长度为 2^m 的计数
且	乘法运算	ROM * *	只读存储
COMP	数值比较	PROM * *	可编程只读存储
ALU	算术逻辑单元	RAM * *	随机存储

注: * 号用表明单元逻辑功能的总限定符号代替; ** 号用“地址×位数”来代替。

2. 与输入、输出和其他连接有关的定性符号

与输入、输出有关的限定符号用以说明相应输入端或输出端所具有的逻辑功能或物理特性，如表 B.2.2 所示。

表 B.2.2 与输入、输出有关的限定符号

符号	说明	符号	说明
	逻辑非, 在输入端		T 输入
	逻辑非, 在输出端		移位输入, 从左到右或从顶到底
	低有效输入极性指示符		移位输入, 从右到左或从底到顶
	低有效输出极性指示符		加计数输入
	动态输入		减计数输入
	延迟输出		数值比较器的“大于”输入
	具有迟滞特性的输入		数值比较器的“小于”输入
	开路输出 (例如集电极开路、发射极开路、漏极开路、源极开路)		数值比较器的“等于”输入
	三态输出		数值比较器的“大于”输出
	使能输入		数值比较器的“小于”输出
	D 输入		数值比较器的“等于”输出
	J 输入		运算单元的借位输入
	K 输入		运算单元的借位输出
	R 输入		运算单元的进位输入
	S 输入		运算单元的进位输出

相邻单元的方框可以采用如表 B.2.3 所示的内部连接符号表示。

表 B.2.3 内部连接符号

符号	说明	符号	说明
	内部连接		具有逻辑非和动态特性的内部连接
	具有逻辑非的内部连接		内部输入(虚拟输入)
	具有动态特性的内部连接		内部输出(虚拟输出)

表 B.2.4 给出了非逻辑连接和信息流指示符号。其中,非逻辑连接符号表示逻辑图中,如果信号线不是逻辑信号的连接线,用信号线上加“×”表示。

原则上,信息流的方向是从左到右、从上到下。如果不符此规则或信息流方向不明显时,应在信号线上标出指示信息流方向的箭头,如表 B.2.4 所示。

表 B.2.4 非逻辑连接和信息流指示符

符号	说明
	非逻辑连接,在左边示出
	单向信息流
	双向信息流

B.3 关联标注法

有时只使用限定性符号不能充分说明逻辑单元的各输入之间、各输出之间或输入与输出之间的关系,为此,采用关联标注法加以补充说明。

关联标注法引用“影响”和“受影响”两个术语,用以表示信号之间内在的“影响”和“受影响”的关系。例如,在第 5 章锁存器和触发器中涉及关联标注法,如图 B.3.1 所示 D 锁存器符号。其中输入 D 受到输入 E 的影响。当 $E=1$ 时,输出 $Q=D$;当 $E=0$ 时,输入 D 不影响输出 Q 。在符号框中用 D 前面加 1,即 1D 表示受 $C1$ 的影响。

关联标记的标注方法:

- (1) 用一个表示内在关系的字母和后面加标识序号来表示影响其他输入(或输出)。
- (2) 用相同的标识符号来标记每一个受该“影响输入”影响的输入(或输出)。
- (3) 若一个输入(或输出)受两个以上“影响输入”的影响时,则所有“影响输入”的标识序

号都应出现在“受影响输入”的标记中，并以逗号隔开。

这里列出了 10 种关联类型，如表 B.3.1 所示，其中：

与关联、或关联和非关联用来注明输入和（或）输出之间的逻辑关系。

互连关联是用来表示一个输入或输出把其逻辑状态强加到另一个或多个输入和（或）输出上。

控制关联用来标识时序单元的定时输入或时钟输入，并指出受它控制的输入。

置位关联和复位关联用来规定当 R 输入和 S 输入均处在它们的内部 1 状态时，基本触发器的内部逻辑状态。

使能关联用来标识使能输入并指出由它控制的输入和/或输出（例如，哪些输出呈现高阻抗状态）。

方式关联用来标识选择单元操作方式的输入，并指出取决于该方式的输入和/或输出。

地址关联用来标识存储器的地址输入。

表 B.3.1 关联类型

关联类型	关联符号	对“受影响输入”或“受影响输出”的影响	
		当“影响输入（出）”=1 时	当“影响输入（出）”=0 时
控制	C	允许动作	禁止动作
使能	EN	允许动作	禁止“受影响输入”动作、置开路或三态输出于外部高阻抗状态，置其他输出于 0 状态
方式	M	允许动作（已选方式）	禁止动作（未选方式）
复位	R	“受影响输出”复位	不起作用
置位	S	“受影响输出”置位	不起作用
与	G	允许动作	置 0 状态
或	V	置 1 状态	允许动作
非	N	求补状态	不起作用
互连	Z	置 1 状态	置 0 状态
地址	A	允许动作（已选地址）	禁止动作（未选地址）

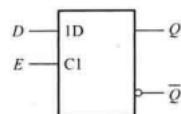


图 B.3.1 D 锁存器的关联标注法

附录 C 常用逻辑符号对照表

名 称	国标符号	IEEE 标准符号	曾用逻辑符号
与门			
或门			
非门			
与非门			
或非门			
与或非门			
异或门			
同或门			
漏极(集电极)开路与非门			
三态输出非门			
带施密特触发特性的与门			

续表

名 称	国标符号	IEEE 标准符号	曾用逻辑符号
传输门			
半加器			
全加器			
SR 锁存器 (触发器)			
逻辑门控 SR 锁存器			
上升沿触发 D 触发器			
下降沿触发 JK 触发器			
正脉冲触发 SR 触发器			
负脉冲触发 JK 触发器			

部分习题答案

第1章

1.1.2 二进制数为 **010110100**

1.1.4 $T = 10\text{ms}$, $f = 100\text{Hz}$, $q = 10\%$

1.2.1 (1) **00101100** (2) 2ms (3) 0.25ms

1.2.2 (1) $(11)_b$ (2) $(23)_b$ (3) $(45)_b$ (4) $(75)_b$

1.2.3 (1) $(0.625)_b$ (2) $(0.1875)_b$
(3) $(0.65625)_b$ (4) $(0.609375)_b$

1.2.4 (1) $(2.25)_b$ (2) $(6.625)_b$
(3) $(14.5625)_b$ (4) $(23.40625)_b$

1.2.5 (1) $(51)_o$ (29)_h (2) $(13.24)_o$ (B.5)_h
(3) $(32.62)_o$ (1A.C8)_h (4) $(67.35)_o$ (37.74)_h

1.2.6 (1) $(101\ 011)_h = (53)_o = (2B)_h$ (2) $(111\ 011)_h = (73)_o = (3B)_h$
(3) $(1\ 111\ 111)_h = (177)_o = (7F)_h$ (4) $(11\ 101\ 010)_h = (352)_o = (EA)_h$

1.2.7 (1) $(0.11100110)_h = (0.714)_o = (0.E6)_h$
(2) $(0.01010111)_h = (0.256)_o = (0.57)_h$
(3) $(0.00000111)_h = (0.016)_o = (0.07)_h$
(4) $(0.10111000)_h = (0.560)_o = (0.B8)_h$

1.2.8 (1) $(100.\ 1101)_h = (4.64)_o = (4.D)_h$
(2) $(1111.\ 0100)_h = (17.2)_o = (F.4)_h$
(3) $(11111110.\ 0110)_h = (376.3)_o = (FE.6)_h$
(4) $(1111101010.\ 0111)_h = (1752.34)_o = (3EA.7)_h$

1.2.9 (1) $(0111)_h$
(2) $(1101\ 0100)_h$
(3) $(10\ 0011\ 1111.\ 0100\ 0101)_h$
(4) $(1010\ 0000\ 0100\ 0000.\ 0101\ 0001)_h$

1.2.10 (1) $(19)_b$
(2) $(259.125)_b$
(3) $(163.046875)_b$
(4) $(42077.0458984375)_b$

1.2.11 $(10E)_h > (165)_o > (1101101)_h > (74)_h$

1.3.1 (1) $A_{\text{IG}} = A_{\text{IS}} = A_{\text{H}} = \text{1110}$
(2) $A_{\text{IG}} = A_{\text{IS}} = A_{\text{H}} = \text{10110}$

$$(3) A_{10} = 11110, A_{12} = 10001, A_{14} = 10010$$

$$(4) A_{10} = 110110, A_{12} = 101001, A_{14} = 101010$$

$$1.3.2 (1) (11)_b \quad (2) (23)_b \quad (3) (-24)_b \quad (4) (-39)_b$$

$$1.3.3 (1) 00001011 \quad (2) 01000100 \quad (3) 11101000 \quad (4) 10100110$$

$$1.3.4 (1) 00010101 \quad (2) 00001000 \quad (3) 11001010 \quad (4) 10100110$$

$$1.4.1 (1) (0100\ 0011)_{BCD}$$

$$(2) (0001\ 0010\ 0111)_{BCD}$$

$$(3) (0010\ 0101\ 0100.\ 0010\ 0101)_{BCD}$$

$$(4) (0010.\ 0111\ 0001\ 1000)_{BCD}$$

$$1.4.2 (1) (151)_b, (97)_b$$

$$(2) (2195)_b, (893)_b$$

$$(3) (329)_b, (149)_b$$

$$(4) (132.56640625)_b, (84.91)_b$$

$$1.4.3 (1) 111 \quad (2) 1110 \quad (3) 10111 \quad (4) 111011$$

$$1.4.4 (1) 100 \quad (2) 1010 \quad (3) 11011 \quad (4) 100101$$

$$1.4.5 (1) (2B)_H \quad (2) (40)_H \quad (3) 79, 6F, 75 \quad (4) 34, 33_0$$

$$1.6.1 L = \overline{A}\overline{B}\overline{C} + \overline{ABC} + \overline{ABC}$$

$$1.6.3 L = \overline{AB} \odot \overline{A+B+C}$$

$$1.6.4 L = \overline{ABC} + \overline{ABC} + ABC$$

第2章

$$2.1.3 (1) \bar{L} = \overline{AB} + \overline{AB}, \quad L' = \overline{AB} + \overline{AB}$$

$$(2) \bar{L} = (\overline{A} + \overline{B})(C + D), \quad L' = (A + B)\overline{CD}$$

$$(3) \bar{L} = (A + B)(\overline{ABC} + \overline{D}), \quad L' = (\overline{A} + \overline{B}) \cdot (\overline{A + B + C} + D) = (\overline{A} + \overline{B})(\overline{ABC} + D)$$

$$2.2.1 (1) \overline{ABC}\overline{D} + \overline{AB}\overline{CD} + \overline{ABC}\overline{D} + \overline{ABCD} + \overline{ABC}\overline{D} + \overline{ABC}\overline{D} + \overline{ABC}\overline{D} + \overline{ABC}\overline{D}$$

$$(2) AC + BC + D$$

$$(3) \overline{AC} + \overline{AD} + \overline{CD} + \overline{B}$$

$$2.2.2 \sum m(2, 4, 5, 6)$$

$$2.2.3 (1) \overline{ABC}\overline{D} + A\overline{B}\overline{CD} + \overline{ABC}\overline{D} + \overline{ABC}\overline{D} + ABC\overline{D} + ABCD$$

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

$$(3) \overline{ABD}(C + \overline{C}) = \overline{ABC}\overline{D} + \overline{ABC}\overline{D}$$

$$(4) \sum m(0, 1, 3, 6, 7)$$

$$2.2.4 (1), (2), (3) 均成立 \quad (4) 不成立$$

$$2.2.5 \prod M(0, 2, 5, 6, 7)$$

$$2.2.6 (1) AB \quad (2) \overline{ABC} + A\overline{B}\overline{C} + A\overline{BC}$$

2.2.7 (1) $(A+B)(\bar{A}+\bar{B})$

(2) $\prod M(0, 3, 4, 7, 8, 9, 10, 11, 12, 13, 14, 15)$

2.3.1 (1) 0 (2) $\bar{A}\bar{B}$ (3) 1 (4) 1 (5) $AB+\bar{B}\bar{C}+\bar{B}\bar{D}$ (6) BC

2.3.5 $L = \overline{\overline{ABC}}\overline{\overline{A}\overline{B}\overline{C}}\overline{ABD}$

2.4.2 $B\bar{C}\bar{D} + \bar{B}\bar{C}\bar{D} + \bar{B}\bar{C}\bar{D}$

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

(2) $A\bar{C}D+\bar{A}\bar{B}C+ABC$

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

(4) $\bar{C}\bar{D}+\bar{B}\bar{D}$

(5) $\bar{C}\bar{D}+\bar{B}\bar{C}+C\bar{D}$

(6) $\bar{A}+D$

(7) $\bar{A}+C+D$

2.4.4 (1) $(A+\bar{B})(\bar{C}+D)$ (2) $(\bar{A}+B)(B+C)(\bar{A}+C+D)(A+\bar{C}+D)$

(3) $B(\bar{A}+D)$ (4) $(B+D)(\bar{A}+\bar{B})(\bar{C}+\bar{D})$

2.5.1 (1)、(3)、(4)和(5)正确；

(2) 错误，因为标识符不能以数字开头。

2.5.3 在 Verilog 程序中，默认的数据类型为 1-bit 的 wire 类型。

2.5.4 (1) $8'b1010_1010$

(2) $\&m = 1'b0, !m = 1'b1, ^m = 1'b0, ~m = 1'b1$ 。

2.5.5 在第 1, 2, 3 行末尾添加漏掉了的分号，第 4 行将“`reg E;`”改为“`wire E;`”，第 5 行将“`and G1(A, B, E);`”改为“`and G1(E, A, B);`”，第 6 行将“`NOT (Y, C);`”改为“`not (Y, C);`”，第 7 行将“`OR (X, E, Y);`”改为“`or (X, E, Y);`”，第 8 行将末尾的分号去掉。

第 3 章

3.2.1 (a) 饱和导通； (b) 截止； (c) 截止； (d) 饱和导通；

3.2.2 (1) 240 μs , 4.2 kHz

(2) $t_r = 5$ ns, $t_f = 6$ ns

(3) $t_{pH} = 8$ ns, $t_{pL} = 7$ ns。

3.2.4 $L = \overline{A(BC+D)}$

3.2.5 $L = AB + \bar{A}\bar{B} = A \odot B$

3.2.8 $L = \bar{A}B + A\bar{B} = A \oplus B$,

3.3.4 $R_{p(min)} = 1.2$ k Ω , $R_{p(max)} = 33.3$ k Ω

3.3.5 $L = \overline{\overline{AB} \cdot \overline{BC} \cdot \overline{D} \cdot E}$ $R_{p(min)} = 1.3$ k Ω , $R_{p(max)} = 17.1$ k Ω

3.3.6 (1) $L = (\bar{AB} + \bar{A}\bar{B})\overline{BC} = \bar{A}\bar{B} + \bar{A}B\bar{C}$ (2) $R_{p(min)} = 1.2$ k Ω , $R_{p(max)} = 60$ k Ω

3.3.10 逻辑门 C

3.3.11 1.3 mW

3.3.12 56%

3.3.13 逻辑门 C_o

3.3.14 (1) 20 (2) 80

3.4.1 $L = \overline{(BC+DE)A+(A+G)EF}$

3.4.2 $L = A \odot B$

3.4.4 $L = \overline{A+B}$

3.5.1 $\frac{R_b}{R_s} \leq 90$

3.5.4 $R_{p(\min)} \approx 0.75 \text{ k}\Omega, R_{p(\max)} \approx 7.7 \text{ k}\Omega$

3.5.5 $R_{p(\min)} \approx 1.1 \text{ k}\Omega, R_{p(\max)} \approx 7.7 \text{ k}\Omega$

3.6.1 $V_{NH} = 0.125 \text{ V}, V_{NL} = 0.155 \text{ V}$

3.8.3 $V_{NH} = 0.4 \text{ V}, V_{NL} = 0.4 \text{ V}$

3.8.4 $V_{NH} = 0.7 \text{ V}, V_{NL} = 0.3 \text{ V}, V_{NH} = 0 \text{ V}, V_{NL} = 0.4 \text{ V}$

3.8.5 $V_{NH} = 1.23 \text{ V}, V_{NL} = 0.23 \text{ V}, V_{NH} = 0.83 \text{ V}, V_{NL} = 0.23 \text{ V}$

3.8.9 $R_{p(\min)} \approx 0.56 \text{ k}\Omega, R_{p(\max)} \approx 4.9 \text{ k}\Omega$

3.8.13 (a) $L = AB + \overline{C}$ (b) $L = CD$

第 4 章

4.1.2 $L = A \odot B$

4.1.3 $W = \overline{CB} + \overline{DC} \overline{A} + \overline{DC} \overline{A} + \overline{CBA}$

$X = D\overline{B} + \overline{B}\overline{A} + CBA + \overline{D}\overline{CA}$

$Y = \overline{D}\overline{C}\overline{B} + \overline{DCB} + \overline{DC}\overline{A}$

$Z = CA + \overline{DB}\overline{A} + DBA + \overline{DCB}$

4.1.4 $L = (A \oplus B) \oplus (C \oplus D)$, 奇校验电路

4.1.6 $S = A \oplus B \oplus C_i, C_o = AB + (A \oplus B)C_i$, 1 位数全加器

4.1.7 $S_0 = A_0 \oplus B_0, C_0 = A_0 B_0$

$S_1 = A_1 \oplus B_1 \oplus C_0, C_1 = A_1 B_1 + (A_1 \oplus B_1)C_0$, 2 位数全加器

4.3.1 (1) 可能 (2) 可能 (3) 不可能 (4) 可能

4.3.2 可能

4.4.1 $Y_2 Y_1 Y_0 = 000$

4.4.14 0, 1, 6, 9, 4_o

4.4.17 $L = I_0 \bar{S}_1 \bar{S}_0 + I_1 \bar{S}_1 S_0 + I_2 S_1 \bar{S}_0 + I_3 S_1 S_0$

当 $I_3 = 0, I_2 = I_1 = I_0 = 1$ 时, $L = \bar{S}_1 + S_1 \cdot \bar{S}_0$

4.4.18 当 $I_0 = 0, I_1 = I_2 = I_3 = 1, F = S_1 + S_0$

4.4.20 (a) $L = A \oplus B \oplus C$

(b) $L = AB + BC + AC$

4.4.37 当 $MN = 00$ 时, $S = I$; 当 $MN = 01$ 时, $S = I+2$; 当 $MN = 10$ 时, $S = I+3$; 当 $MN = 11$ 时, $S = I+5$ 。

4.4.38 当 $A > B$ 时, $S = A - B$; 当 $A < B$ 时, $S = B - A$ 。

4.5.1 $L_0 = \overline{AB} + AC + \overline{ABC}$, $L_1 = \overline{AC} + BC$

4.5.3 $L_1 = \overline{BC} + \overline{AC} + \overline{ABC}$, $L_2 = \overline{A}\overline{B} + AB + BC$,

$L_3 = \overline{BC} + \overline{AC} + \overline{A}\overline{BC} + ABC = L_1 + ABC$, $L_4 = C + \overline{AB}$

4.6.2 (1) $AltB = \overline{A[1]} \cdot B[1] + A[1] \cdot \overline{A[0]} \cdot B[0] + \overline{A[0]} \cdot B[1] \cdot B[0]$

$AeqB = \overline{A[1]} \oplus B[1] + \overline{A[0]} \oplus B[0]$, $AgtB = \overline{AltB + AeqB}$

(2) $Y1 = A \cdot B$, $Y2 = A + B$

4.6.6 实现的功能是:两路带有三态门的 2 选 1 数据选择器。

4.6.10 (1) 在层次化设计时,上层模块调用下层模块时,上层模块中的输出变量必须定义成 wire 类型,因此 reg Cout; 改为 wire Cout; reg [4:0] temp; 改为 wire [4:0] temp;

(2) 在数据类型说明部分,未定义输出变量 Sum 的类型,因此需增加一行: wire [3:0] Sum;

(3) 上层模块调用下层模块时,只能用结构化的描述方式将模块之间的连接关系表示出来,不能采用行为描述方式中过程块的方式进行调用。因此,要删除 always @ (A or B or Cin)、begin、end 语句,将 temp[0] = Cin; 和 Cout = temp[4]; 语句改为连续赋值语句,即改为 assign temp[0] = Cin; 和 assign Cout = temp[4]; 就可以了,其他语句不用改。

第 5 章

5.3.1 ①省去一个非门;②D 端少驱动一个门;③单纯用与非门实现,在某些类型的集成电路中,与非门速度高于或非门。

5.4.1 带直接置1、置0功能的主从结构 D 触发器

5.4.3 维持阻塞结构的 JK 触发器。

5.5.8 ϕ_2 落后 ϕ_1 一个 \overline{CP} 周期

5.5.11 $T = \overline{D}\overline{Q} + D\overline{Q}$

5.5.12 $T = J\overline{Q} + KQ$

5.6.1 带复位功能的 D 锁存器。

第 6 章

6.1.5 Z 序列为: 0, 1, 1, 0, 1, 0

6.1.6 输出序列为 1, 0, 1, 0, 1, 0, 1。

6.1.7 串行加法器, Q_1 为和, Q_0 为进位信号

6.3.2 激励方程组: $D_2 = \overline{Q}_0$, $D_1 = \overline{Q}_2$, $D_0 = \overline{Q}_1$

6.3.4 激励方程组: $D_1 = \overline{A}Q_0$, $D_0 = A\overline{Q}_1$; 输出方程: $Y = AQ_1$

6.4.3 $M=7$ 异步计数器

6.5.1 四相时序脉冲产生电路

6.5.4 $M=6$

6.5.8 $M=5$

6.5.9 $M=7$

6.5.12 $t_{COMpd(max)} = 5t_{Gpd}$

6.5.13 $M=10$

6.5.14 $M=11$

6.5.15 $M=11$

6.5.16 $M=11$

6.5.18 $M=174$

6.5.19 $M=174$

6.7.2 带并行数据输入的 n 位通用移位寄存器。

第 7 章

7.1.1 (1) 64K,16 根,1 根 (2) 1M,18 根,4 根

(3) 1M,18 根,1 根 (4) 1M,17 根,8 根

7.1.2 (1) 7FFH (2) 3FFFH (3) 3FFFFH

7.1.3 (1) 26×6 位 (2) 28×10 位

7.2.2 0095H,0096H,0097H

7.2.3 6.4 μ s, 约 1 ms, 0.64%

7.2.4 10 根

第 8 章

8.1.1 16 个,36 个,80 个

8.2.1

$$\begin{aligned} Y &= AL_1 + \bar{A}L_0 \\ &= A(\bar{B}\bar{C}\bar{D}\bar{E} + \bar{B}\bar{C}\bar{D}\bar{E} + \bar{B}\bar{C}\bar{D}\bar{E} + \bar{B}\bar{C}\bar{D}\bar{E} + \bar{B}\bar{C}\bar{D}\bar{E}) \\ &\quad + \bar{A}(\bar{B}\bar{C}\bar{D}\bar{E} + \bar{B}\bar{C}\bar{D}\bar{E} + \bar{B}\bar{C}\bar{D}\bar{E} + \bar{B}\bar{C}\bar{D}\bar{E} + \bar{B}\bar{C}\bar{D}\bar{E}) \end{aligned}$$

第 9 章

9.1.2 $t_s = 69 \mu$ s

9.1.3 $t_s \approx 0.7 RC$

9.1.4 $t_s \approx (3.57 \sim 17.57) \text{ ms}$

9.2.2 ΔV_T 变动范围为 0.7 ~ 1.4 V

9.2.3 $t_s = 0.21 \text{ ms}$

$$9.3.1 T = t_1 + t_2 = RC \ln \frac{(V_{DD} + V_{TH})(2V_{DD} - V_{TH})}{V_{TH}(V_{DD} - V_{TH})}$$

$$9.3.3 \quad f = \frac{1}{T} \quad T = R_1 C_1 \ln \frac{V_{DD} - V_{T+}}{V_{DD} - V_{T-}} + R_2 C_2 \ln \frac{V_{T+}}{V_{T-}}$$

$$9.4.2 \quad f = \frac{1}{t_{PL} + t_{PL}} = \frac{1.43}{(R_1 + 2R_{DS})C}$$

$$9.4.3 \quad t_s = \frac{2R_s(R_1 + R_2)C}{3R_2}$$

$$9.4.7 \quad R_1 = 910 \text{ k}\Omega \quad R_2 = 0.61 \text{ k}\Omega$$

$$9.4.8 \quad t_s = R_s C_s \ln 3 = 1.1 R_s C_s$$

第 10 章

$$10.1.1 \quad (1) (-V_{REF} \sim 0) \text{ V} \quad (2) V_{REF} = -10 \text{ V}$$

$$10.1.2 \quad R_f = 16 \text{ k}\Omega$$

$$10.1.3 \quad 3.33\%$$

$$10.1.4 \quad (1) v_o = - \left(\frac{V_{REF}}{2^{10} \cdot R} \sum_{i=0}^9 D_i \cdot 2^i + \frac{V_B}{R_B} \right) R_f \quad (2) \frac{|V_B|}{R_B} = \frac{|V_{REF}|}{2R}$$

$$10.2.1 \quad \Delta = 2V_{REF}/15; D_2D_1D_0 = 011.$$

$$10.2.3 \quad D_3D_2D_1D_0 = 1101.$$

$$10.2.5 \quad (1) T_i = 100 \text{ ms} \quad (2) 5 \text{ V} \quad (3) -8.3 \text{ V}$$

$$10.2.6 \quad f_{min} = \frac{1}{T_{max}} = f_{CP}/2^{n+1}$$

索引(汉英对照)

二 画

- 二值数字逻辑 (Binary digital logic) 7
二进制 (Binary) 13
 ~ 数 (number) 13
 ~ 权 (weights) 13
二进制编码的十进制 (Binary-Coded-Decimals, BCD) 25
二-十进制转换 (Binary to decimal conversion) 15
二级管 (Diode) 3
 发光 ~ (Light-emitting ~) 129
 肖特基势垒 ~ (Schottky Barrier ~ , SBD) 114
十进制数 (Decimal number) 12
十-二进制转换 (Decimal to binary conversion) 15
十六进制数 (Hexadecimal number) 18
八进制数 (Octal number) 8
七段显示器 (Seven-Segment display) 173
二的补码 (Two's complement) 22

三 画

- 三极管 (Bipolar Junction Transistor, BJT) 111
 肖特基 ~ (Schottky ~) 114
三极管-三极管逻辑 (Transistor-Transistor Logic, TTL) 3, 111
下降沿 (Fall edge) 11
下降时间 (Fall time) 10
上拉电阻 (Pull-up resistor) 96
上升沿 (Rise edge) 11
上升时间 (Rise time) 10
门 (Gate) 30
 ~ 阵列 (array) 198
与 (AND) ~ 30

- 或 (OR) ~ 31
非 (NOT) ~ 31, 86
与或非 (AND-OR-INVERT) ~ 92
异或 (Exclusive OR) ~ 32, 92
同或 (Exclusive NOR) ~ 32, 92
与非 (NAND) ~ 32, 91
或非 (NOR) ~ 32, 91
三态 (three state) ~ 96
漏极开路 (open drain) ~ 96
集电极开路 (open collector) ~ 115
门级元件 (Gate Level Primitives) 67

四 画

- 专用集成电路 (Application Specific Integrated Circuit, ASIC) 3
无关项 (Don't care terms) 59
开启电压 (Threshold voltage) 82
开关特性 (Switching characteristics) 81
开关时间 (Switching time) 114
互补 MOS 门 (Complementary MOS gate, CMOS) 86
 增强型 ~ (Enhancement-mode ~) 81
 耗尽型 ~ (Depletion-mode ~) 81
卡诺图 (Karnaugh map) 54
分配器 (Demultiplexer) 177
分辨率 (Resolution) 456, 470
分频 (Frequency division) 316
反相器 (Inverter) 86
 反演规则 (Complementary operation theorem) 44
 反码 (One's complement) 22
计数器 (counter) 316
 模 $M \sim (\text{modulo}-M \sim)$
 异步 ~ (asynchronous ~) 316

- 同步 ~ (synchronous ~) 317
 二进制 ~ (binary ~) 316
 BCD ~ (BCD ~) 324
 递增 ~ (up ~) 316
 递减 ~ (down ~) 318
 同步串行 ~ (synchronous serial ~) 320
 同步并行 ~ (synchronous parallel ~) 321
 纹波 ~ (ripple ~) 316
 环形 ~ (ring ~) 329
 扭环形 ~ (twisted-ring ~, Johnson ~) 329
 双稳态 (Bistable) 231
 双向移位寄存器 (Bidirectional shift register) 313
 双列直插式封装 (Dual In-line Package, DIP) 79
 双积分 (斜) 模数转换器 (Dual-slope analog to digital converter) 467
 比特率 (Bit rate) 9
- ## 五 画
- 可编程逻辑器件 (Programmable Logic Device, PLD) 197
 可编程阵列逻辑 (Programmable Array Logic, PAL) 202
 布尔代数 (Boolean algebra) 41
 正逻辑 (Positive logic) 120
 功能表 (Function table) 164
 只读存储器 (Read Only Memory, ROM) 364
 可编程 ~ (Programmable ~, PROM) 364, 202
 光可擦除可编程 ~ (Erasable Programmable ~, EPROM) 200, 364
 电可擦除可编程 ~ (Electrical Erasable Programmable ~, E²PROM) 201, 364
 电子设计自动化 (Electronic Design Automatic, EDA) 5
 电平 (Level) 7
 有效 ~ 101, 122
 高 ~ 有效 (Active high) 122
 低 ~ 有效 (Active low) 122
 占空比 (Pulse duration ratio) 10
- 印刷电路板 (Printed Circuit Board, PCB) 60
 片上系统 (System on Chip, SoC) 3
 加法器 (Adder) 190
 半 (half) ~ 190
 全 (full) ~ 190
 串行进位 (Serial carry) ~ 192
 半导体存储器 (Semiconductor memory) 364
 - 读/写控制 (~ read-write control) 373
 延时-功耗积 (Time delay-power dissipation product) 105
- ## 六 画
- 权 (Weight) 12
 存储器 (Memory) 364
 只读 (Read-Only ~, ROM) 364
 读/写 (Read-Write ~, RWM) 364
 随机存取 (Random Access ~, RAM) 364
 静态 (static) ~ 364
 动态 (dynamic) ~ 364
 快闪存储器 (flash ~) 364
 存储阵列 (Memory array) 365
 动态 ~ (dynamic ~) 364
 静态 ~ (static ~) 364
 存储单元 (Memory cell) 365
 动态 (dynamic) ~ 381
 静态 (static) ~ 375
 存储时间 (Storage time) 370
 动态随机存取存储器 (Dynamic Random Access Memory, DRAM) 364
 同步 ~ (Synchronous ~, SDRAM) 381
 地址 (address) 365
 - 单元 (~ cell) 365
 在系统可编程 (In System Programmability, ISP) 391
 再生 (刷新) (Regenerate, Refresh) 381
 列选择线 (Column—Select line) 373
 同或门 (见门)
 回差电压 (Backlash voltage) 420
 异或门 (见门)

异步输入(asynchronous input) 274
 传输延迟(propagation delay) 104
 自校正(self-correcting) 388
 传输延迟时间(Propagation delay time) 104
 传输特性(Transfer characteristic) 87
 传输门(Transmission Gate, TG) 92
 行选择线(Row—Select line) 375
 字(Word) 365
 ~长(~length) 365

七 画

运算电路(Arithmetic circuit) 190
 取样-保持电路(Sample-hold circuit) 460
 时钟(clock) 246
 ~脉冲(~pulse) 246
 ~偏移(~skew) 303
 时序可编程逻辑器件(Sequential Programmable Logic Device, SPLD) 331
 时序图(timing sequence diagram) 11, 281
 时序逻辑电路(sequential logic circuit) 273
 同步(~synchronous ~) 274
 异步(~asynchronous ~) 274
 米利型(~Mealy ~) 275
 穆尔型(~Moor ~) 275
 余3码(Excess three code) 25
 位(Bit) 9
 运算符(Operator) 64
 译码器(Decoder) 166
 ~二进制(binary) ~ 166
 ~二-十进制(BCD) ~ 172
 ~七段显示(~Seven-segment display decoder) 173
 ~线(line)
 状态(state) 273
 ~变量(~variable) 274
 ~名(~name) 279
 ~化简(~reduction) 289
 ~分配(~assignment) 290

现态(present ~) 254
 次态(next ~) 254
 初始~(initial ~) 256
 无效~(unused ~) 293
 等价~(equivalent ~) 294
 状态表(state table) 279
 状态机(state machine) 274
 有限~(finite ~) 274
 时钟同步~(clocked synchronous ~) 275
 状态图(state diagram) 279

八 画

现场可编程门阵列(Field Programmable Gate Array, FPGA) 202, 391
 奇偶校验(Odd-even check) 149
 拉电流(Draw-off current) 89
 转换表(transition table) 278
 转换方程(transition equation) 277
 或门(见门)
 或非门(见门)
 非门(见门)
 时序图(Timing diagram) 11, 281
 金属-氧化物-半导体场效应管(Metal-Oxide-Semiconductor Field-effect Transistor, MOSFET) 81
 和之积(Products of Sum, POS) 45
 单稳态触发器(Monostable multivibrator) 410
 定时器(Timer) 431
 定时图(timing diagram) 11, 234
 定时参数(timing parameters) 244
 线与(Wire-AND) 96
 组合逻辑电路(Combinational logic circuit) 148
 参考电压(Reference voltage) 117
 建立时间(Setup time) 244, 458

九 画

查找表(look-up table) 183, 395
 相邻项(Adjacencies) 54
 恢复时间(Recovery time) 412

复位(reset) 233

复杂可编程逻辑器件(Complex Programmable Logic Device,CPLD) 202,391

保持时间(hold time) 244

保持时间约束(hold-time margin)

施密特触发器(Schmitt trigger) 419

总线(Bus) 102

宽~(Widebus) 132

十 画

真值表(Truth table) 33

逐次逼近A/D转换器(见模数转换器)

射极耦合逻辑(见逻辑)

乘积项(Product Term,PT) 34

积之和(Sum of Products,SOP) 45

特性表(characteristic table) 254

特性方程(characteristic equation) 255

扇出(Fan out) 106

扇入(Fan in) 106

竞争-冒险(Race and hazard) 158

读/写控制(Read-write control) 374

高阻态(High impedance state) 101

高电平输入电流(High level input current) 106

通用阵列逻辑(Generic Array Logic,GAL) 331

预置(preset) 291

十一 画

逻辑(Logic) 7

~电平(level) 7

~函数(function) 33

~变量(variables) 29

~门(gate) 78

~表达式(expression) 29

~图(diagram) 34

正~(positive) 120

负~(negative) 120

三极管-三极管~(Transistor-Transistor~,TTL)

4,111

电流开关型~(Current-mode~,CML) 119

小~(Little logic) 131

~仿真(Logic Simulation) 60

~综合(Logic Synthesis) 60

射极耦合~(Emitter-Coupled~,ECL) 117

清零(clear) 322

减法器(Subtractor)

寄存器(register) 310

移位~(shift~) 311

双向移位~(bidirectional shift~) 313

随机存取存储器(Random Access Memory, RAM)

373

双倍数据传输率~(Double Data Rate,DDR~)

380

四倍数据传输率~(Quad Data Rate,QDR~)

380

单倍数据传输率~(Single Data Rate,SDR~)

380

十二 画

硬件描述语言(Hardware Description Language,HDL)

60

超前进位(Look~ahead carry) 192

~产生器(generator) 193

最大项(Maxterm) 47

量化(Quantification) 461

最小项(Miniterm) 46

集成电路(Integrated circuit) 3

小规模~(Small scale integrated circuit) 4

中规模~(Medium scale integrated circuit) 4

大规模~(Large scale integrated circuit) 4

超大规模~(Very large scale integrated circuit)

4

甚大规模~(Ultra large scale integrated circuit)

4

集成度(Integration) 4

锁存器(latch) 232

SR~(SR~) 232

- 门控 SR ~ (gated SR ~) 237
 D ~ (D ~) 239
 编码(encoding) 161
 状态 ~ (state ~) 290
 一对一一 (one-hot ~) 300
 编码器(Encoder) 161
 优先 (priority) ~ 162
- ### 十 三 画
- 输出方程 (output equation) 274
 输出逻辑宏单元 (Output Logic Macro Cell, OLMC) 331
 置位 (set) 233
 触发 (triggering) 245
 触发器 (flip-flop) 245
 主从 ~ (master-slave ~) 245
 边沿触发的 ~ (edge-triggered ~) 245
 D ~ (D ~) 246
 JK ~ (JK ~) 251
 T ~ (T ~) 257
 T^{*} ~ (T^{*} ~) 258
 SR ~ (SR ~) 258
 数字电路 (Digital circuit) 3
 数码显示器 (Digital display) 173
 数字比较器 (Digital comparator) 186
 数模转换器 (Digital to Analog Converter, DAC) 445
 权电阻 (weighted resistor) ~ 446
 权电流 (weighted current) ~ 450
 权电容 (weighted capacitor) 452
 倒 T 形 (inverted T) ~ 447

数据选择器/多路复用器 (Data selector/Multiplexer)
 179, 184

十 四 画

- 静态随机存取存储器 (Static Random Access Memory, SRAM) 373
 同步 ~ (Synchronous ~, SSRAM) 378
 异步 ~ (Asynchronous ~, ASRAM) 378
 双口 ~ (dual-port ~) 377
 先进先出 ~ (First-In First-Out, FIFO ~) 377
 模拟开关 (Analog switch) 92
 漏极开路门 (见门)
 模数转换器 (Analog to Digital Converter, ADC) 460
 双积分 (dual slope) ~ 467
 逐次比较 (successive comparation) ~ 464
 算法状态机 (Algorithmic State Machine, ASM) 481
 算法状态机图 (Algorithmic State Machine diagram, ASM diagram) 481

十 五 画

- 噪声容限 (Noise margin) 103
 高电平 ~ (High-state ~) 103
 低电平 ~ (Low-state ~) 103
 摩根定理 (De Morgan's theorem) 42

十 六 画

- 激励表 (excitation table) 295
 激励方程 (excitation equation) 290

电子技术基础 模拟部分 第六版 康华光

电子技术基础 模拟部分 第六版 电子教案

电子技术基础 模拟部分 第六版 学习辅导与习题解答

电子技术基础 数字部分 第六版 康华光

电子技术基础 数字部分 第六版 电子教案

电子技术基础 数字部分 第六版 学习辅导与习题解答

华中科技大学电子技术基础课程基本教程系列

模拟电子技术基础 第三版 张林 陈大钦

模拟电子技术基础 第三版 习题解答

模拟电子技术基础 第三版 电子教案

数字电子技术基础 第三版 罗杰 彭容修

数字电子技术基础 第三版 习题解答

数字电子技术基础 第三版 电子教案

ISBN 978-7-04-038004-0

9 787040 380040 >

定价 53.00 元