

VHDL

芯片设计

刘绍汉 林灶生 刘新民 编著

逻辑电路的设计已经进入以软件、语言方式实现电路的时代。VHDL语言具有功能强大，可以灵活运用优势，让你轻松地规划硬件控制电路！



清华大学出版社

VHDL 芯片设计

刘绍汉 林灶生 刘新民 编著

清华大学出版社

北 京

内 容 简 介

本书由浅入深、循序渐进地介绍了 Xilinx 公司提供的 VHDL 语言,从逻辑电路设计的发展过程、VHDL 语言中各部分的特性、VHDL 语言的属性……到层级式模式模块化的电路设计、过程设计及程序包的设计和建立,全面探讨了整个 VHDL 语言的特性和设计技巧。

书中附有许多程序范例,所有程序范例皆验证无误,并已经收录到配书的压缩文档中。本书适合于理工学院电子系、电机系学习芯片设计课程的学生及 VHDL 初学者,或已经有基础概念且有志于开发数字集成电路芯片的中级读者使用。

本书繁体字版书名为《VHDL 晶片设计》,由台湾全华科技图书股份有限公司出版,版权属刘绍汉、林灶生、刘新民所有。本书简体字中文版由台湾全华科技图书股份有限公司授权清华大学出版社独家出版,仅限于中国大陆地区出版发行。未经本书原版出版者和本书出版者书面许可,任何单位和个人不得以任何形式或任何手段复制或传播本书的部分或全部内容。

北京市版权局著作权合同登记号 图字:01-2004-3981

版权所有,翻印必究。举报电话:010-62782989 13901104297 13801310933

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

本书防伪标签采用清华大学核研院专有核径迹膜防伪技术,用户可通过在图案表面涂抹清水,图案消失,水干后图案复现;或将表面膜揭下,放在白纸上用彩笔涂抹,图案在白纸上再现的方法识别真伪。

图书在版编目(CIP)数据

VHDL 芯片设计/刘绍汉,林灶生,刘新民编著.—北京:清华大学出版社,2004.11

ISBN 7-302-09729-1

I. V… II. ①刘…②林…③刘… III. 硬件描述语言, VHDL—程序设计 *IV. TP312

中国版本图书馆 CIP 数据核字(2004)第 105325 号

出 版 者:清华大学出版社 地 址:北京清华大学学研大厦
http://www.tup.com.cn 邮 编:100084
社 总 机:010-62770175 客 户 服 务:010-62776969

责任编辑:宋延清

封面设计:陈刘源

印 刷 者:北京季蜂印刷有限公司

装 订 者:三河市新茂装订有限公司

发 行 者:新华书店总店北京发行所

开 本:185×260 印 张:29.5 字 数:715 千字

版 次:2004 年 11 月第 1 版 2004 年 11 月第 1 次印刷

书 号:ISBN 7-302-09729-1/TP·6736

印 数:1~4000

定 价:42.00 元

本书如存在文字不清、漏印以及缺页、倒页、脱页等印装质量问题,请与清华大学出版社出版部联系调换。联系电话:(010)62770175-3103 或(010)62795704

再版前言

曾几何时，电子爱好者们对由二极管、三极管、电容器、电阻器等分立元件构成的各种电子线路如痴如狂，他们对照着电路图，挥舞着手中的电烙铁焊接一个个管脚，然后悠哉乐哉地徜徉于越来越庞大复杂的功能实现中。但是那个时代迅速地过去了。

功能各异和复杂多变的集成电路的出现，让一般电子爱好者们始料不及。面对着数量庞大的集成电路元件，人们不禁慨叹：到底我应当选用哪些元件？

最为聪明的电子工程师会觉得与其被动地选择，莫如跃进到一种较高的层次，“让我来随心所欲地设计自己所需要的集成元件！”

事实证明，这样的想法不仅完全有必要，而且在新技术的不断推动下，变得十分可行。如今运用 VHDL 编程语言，智慧高超的芯片开发设计人员可以对各种复杂的数字逻辑轻而易举地进行自定义的组合，产生出由自己设计的超大规模集成电路元件。

VHDL 语言是如此的清晰明确，其蕴含的智慧和创造力是那样的令人惊奇和神往；面对类似微处理器那样复杂的逻辑设计，不少人想必尚不知道它们竟然是用编程方法实现的。可以毫不夸张地说，硬件编程语言是开启高科技大门的一把钥匙。而 VHDL 是一种优秀的硬件编程语言。只要你对硬件不觉得陌生，只要你对一般程序设计稍微有了解，只要你个人的创造欲望并没有被其他的创造者所征服，那么学习 VHDL 语言便是顺理成章的事情。

本书原名为《VHDL 晶片设计》，由台湾全华科技图书股份有限公司出版，在业界获得广泛好评，并数次再版。该书语言简练、精粹、通俗易懂，并且对关键的编程概念运用英文反复强调，非常有利于读者对设计技能的掌握；全书紧密围绕程序架构的三种风格和形式阐述设计思想，行文中穿插了大量的带有解题过程的例题，并配合合成的硬件电路图，生动形象地揭示了具体程序架构的含义。

本书由清华大学出版社聘请有关专业技术人士，根据普通话语言习惯及国标专业词汇进行了必要的改版。但是为了避免程序错误，对全部程序代码均采用原版图片方式排版。并对本书全部配图都进行了严格的文字处理和校对。

编辑

2004年9月

作者序

如今电子产品的更新速度越来越快，功能要求越来越高。传统的逻辑电路设计方式已经无法满足这些需要。为了在极短的时间内设计出一个功能强大的控制电路，对于逻辑电路的设计过程，我们早已跳过传统逻辑门级别即 Gate Level 的层次，直接进入到了以软件或语言方式来实现电路的时代。也就是说，工程师们如今可以选择专业厂商所提供的超大规模元件，并以软件或编程语言方式，将它规划成自己所需要的硬件控制电路。在市面上提供上述服务的厂商有 Xilinx 和 Altera 等等。各厂商所提供的设计方式有绘图方式 Schematics、有限状态机器 FSM 和硬件描述语言方式等；其中弹性较大、功能较强且被大众所广泛使用的为硬件描述语言 VHDL。本书的主要目的就是介绍 Xilinx 公司所提供的 VHDL 语言的特性和使用方法（其特性与 Altera 相似，只是语法不同而已）。我们将整个 VHDL 语言依照其特性和前后顺序总共分成 9 个章节，各个章节的内容分别为：

- 第1章 介绍逻辑电路设计的发展过程，使读者了解整个历史过程，从早期的 SSI → MSI → LSI → VLSI，从传统的逻辑门级别 → PLD → FPGA 芯片设计的概念和原理。
- 第2章 介绍 VHDL 语言的程序结构及保留字，使读者了解设计一个 VHDL 语言程序时，每个单元的特性，以及哪些单元必须存在，哪些单元可以不要，哪些单元可以使用，哪些单元不可以使用（系统在使用）等。
- 第3章 介绍数据对象和数据类型，使读者了解系统提供了哪些数据对象和数据类型，如何定义属于自己的数据类型，以及这些数据对象及数据类型的特性和使用方式等。
- 第4章 介绍属性、并发性和顺序性，使读者了解系统提供了哪些属性函数，了解 VHDL 语言所提供电路架构的描述方式，以及了解何为并发性描述和顺序性描述等。
- 第5章 介绍数据流的叙述，使读者了解 Data Flow 的描述方式，以及用于数据流描述的并发性语句“<=”、when...else 和 when...select...when...等命令的特性和语法。（利用很多程序来说明。）
- 第6章 介绍行为描述，使读者了解 Behavior 的描述方式，以及用于行为描述的顺序性语句 process、if、case...is...when、loop 和 wait 等命令的特性和用法。（配合很多程序来说明。）
- 第7章 介绍层级式、模块化及参数化的电路设计。使读者了解层级式和模块化、参数化的设计方式，并熟悉系统所提供的模块化结构、“方块”、“元件”、“参数化元件”、“参数化重复元件”的特性和使用方法。（配合很多程序及屏幕画面来说明。）

- 第8章 介绍函数、过程和程序包，使读者了解“函数”、“过程”和“程序包”的设计方式、建立方式和使用方式。（配合很多程序及屏幕画面来说明。）
- 第9章 介绍状态机、计数器和移位寄存器的设计，使读者了解如何去设计 Moore Machine 及 Mealy Machine、Counter 和 Shift Register 等常见的电路。

在介绍完整个 VHDL 语言的特性和设计技巧之后，在附录 B 中，我们将一个完整的 VHDL 程序从开始设计到整个电路实现以及硬件电路在 IC 内部的分布状况等，以屏幕画面的方式逐一详细地介绍一遍。读者只要依照本书的指引，一个画面一个画面地完成即可（无须求助于人）。另外在附录 C 内我们以实际的例子介绍了很多命令文件的指令，看完了这些命令，读者在建立命令文件时，即可随心所欲地撰写出自己所需要的功能模拟程序。本书附带一张 CD 光盘（改版注：本光盘现已取消，其全部内容已制成配书压缩文档，读者请从清华网站下载：<http://www.wenyuan.com.cn>），凡是书中出现过的程序及硬件电路文件等，皆被刻录于其中。读者可以依照书内所指定的文件名称，找出其程序的源文件以及合成之后的硬件电路文件，并将他们显示在屏幕上（依照附录的操作流程）。

本书比较适合初学者，如果您是初学者且逻辑设计的功底不是很好时，可以阅读我们所撰写的《最新逻辑设计》一书，该书对于逻辑设计的发展和概念都有详细的陈述；如果您是这方面的高手，或者比较习惯于使用 Verilog 时，我们正在撰写这方面较为深入的实例书籍，敬请期待。

刘绍汉

林灶生

谨呈

2003 年 6 月

如何使用配书压缩文档

由于示例文件压缩后只有 5MB 左右，所以本书的清华版已取消配书光盘，读者请到清华网站 <http://www.wenyuan.com.cn> 下载配书压缩文档。解压缩后存在如下两个目录：

Book1: 本书在 Xilinx 系统下开发出来的所有项目文件。

Book2: 本书所有的 VHDL 程序文件。

而每个文件或者项目名称与书内每个程序标题的名称完全相同，读者在阅读本书内容时，可以很容易地在压缩文档内找到需要的文件。至于使用方式，可按如下的介绍操作。

1. 如果您也是使用 Xilinx 系统，那么，
 - (1) 将目录 Book1 的所有内容复制到硬盘中。
 - (2) 在 DOS 系统下进入到 Book1 目录内下达 `Attrib *.*-r/s` 命令（也可在 Windows 系统下右击文件后从属性对话框直接修改），将其内部所有的只读文件修改成非只读文件。
 - (3) 完成上述操作后，我们即可在 Xilinx 系统内将经过合成的硬件电路直接显示在屏幕上。
2. 如果您不是使用 Xilinx 系统，那么，

由于系统不同，因此我们直接提供本书所有程序的 VHDL 文件（全部存放在 Book2 目录中）。读者可以将其内容任意打印出来，或在您使用的系统中稍加修改后即可使用。

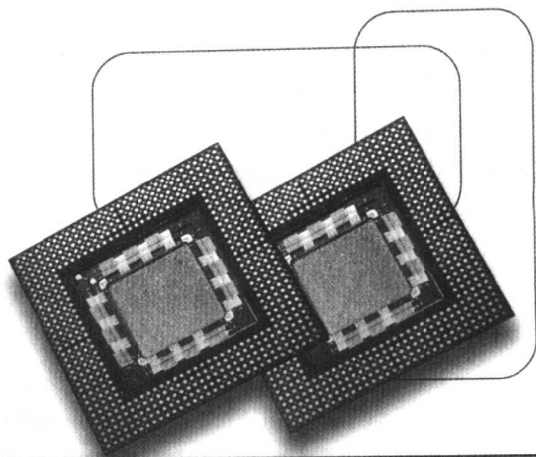
目 录

第 1 章 集成电路设计的发展	1	2.8 完整的 VHDL 程序的要素	28
1.1 使用小规模集成电路元件进行设计	2	2.9 标识符 (Identifier) 及大小写	29
1.2 使用中规模集成电路元件进行设计	5	第 3 章 数据对象和数据类型	31
1.3 使用大规模和超大规模集成 电路元件进行设计	7	3.1 数据类型 (Data type)	32
1.3.1 PROM (可编程 ROM)	7	3.2 预定义数据类型	33
1.3.2 PLA (可编程逻辑阵列)	9	3.2.1 位 (Bit)	33
1.3.3 PAL (可编程阵列逻辑)	10	3.2.2 字符 (Character)	33
1.3.4 PEEL (可编程电可清除 逻辑阵列)	11	3.2.3 位向量 (Bit_Vector)	33
1.3.5 FPGA (现场可 编程门阵列)	12	3.2.4 字符串 (String)	34
1.4 FPGA 元件的规划方式	14	3.2.5 整型 (Integer)	34
1.4.1 电路图设计方式	15	3.2.6 实型 (Real)	34
1.4.2 FSM (有限态自动机)	15	3.2.7 自然数 (Natural)	35
1.4.3 Waveform (信号波形 输入方式)	16	3.2.8 正数 (Positive)	35
1.4.4 VHDL (硬件描述语言)	16	3.2.9 布尔 (Boolean)	35
1.5 什么是 VHDL	17	3.2.10 时间 (Time)	36
第 2 章 VHDL 语言的程序结构 及保留字	19	3.2.11 预定义数据类型的使用	36
2.1 VHDL 语言的程序结构	20	3.3 自定义数据类型	40
2.2 注释 (Comment) 栏	21	3.3.1 枚举 (Enumeration)	40
2.3 库程序 (library)	22	3.3.2 子数据类型 (Sub-Type)	41
2.4 实体 (entity)	22	3.3.3 数组 (Array)	41
2.5 端口 (port)	23	3.3.4 记录 (Record)	43
2.5.1 输入 (in)	23	3.3.5 物理量 (Physical)	45
2.5.2 输出 (out)	24	3.4 IEEE Std_logic_1164 程序包	46
2.5.3 输入输出双向 (inout)	24	3.5 数据对象 (Data object)	49
2.5.4 缓冲器 (Buffer)	24	3.5.1 常量 (Constant)	49
2.6 架构 (architecture)	27	3.5.2 信号 (Signal)	50
2.7 架构的描述方式	28	3.5.3 变量 (Variable)	52
		3.5.4 别名 (Alias)	52
		3.6 运算符 (Operator)	55
		3.6.1 逻辑运算 (Logical Operation)	55
		3.6.2 关系运算 (Relational Operation)	56

3.6.3	算术运算	57	Declaration)	86
3.6.4	数值移位运算	58	4.7.5	元件连线映射 (Mapping) 87
3.6.5	其他运算	60	4.7.6	名称映射 (Mapping
3.6.6	运算符 (Operator) 的		By Name)	87
	优先顺序	61	4.7.7	位置映射 (Mapping
3.7	常用符号	62	By Position)	88
3.7.1	对象连接 (Concatenation)		第 5 章	以数据流风格来描述架构 95
	“&”	63	5.1	并发性语句 (Concurrent
3.7.2	信号设置 “<= ”	65	statement)	96
3.7.3	设置初值或变量内容 “:= ”	66	5.1.1	程序 A 96
第 4 章	属性、并发性、顺序性	69	5.1.2	程序 B 96
4.1	属性 (Attributes)	70	5.2	直接式信号设定 <= 97
4.2	数值属性 (Value Attributes)	70	5.3	条件式信号设定 when...else 108
4.2.1	数值类型属性 (Value		5.3.1	单行语句条件式信号设定 108
	Type Attribute)	70	5.3.2	多行语句 (Multiple statement)
4.2.2	数值数组属性 (Value		条件式信号设定	118
	Array Attribute)	71	5.4	选择性信号设定 with...select...
4.2.3	数值块属性 (Value		when	132
	Block Attribute)	72	第 6 章	以行为风格来描述架构 151
4.3	函数属性 (Function Attributes)	73	6.1	process 语句 152
4.3.1	函数类型属性 (Function		6.2	if 语句 153
	Type Attribute)	74	6.2.1	if...then...end if 153
4.3.2	函数数组属性 (Function		6.2.2	if...then...else...end if 155
	Array Attribute)	75	6.2.3	if...then...elsif...end if 156
4.3.3	函数信号属性 (Function		6.2.4	if...then...elsif...else...
	Signal Attribute)	76	end if	159
4.4	信号属性 (Signal Attributes)	79	6.2.5	嵌套式 if 语句 160
4.5	类型属性 (Type Attributes)	81	6.3	case...is...when 语句 170
4.6	范围属性 (Range Attributes)	81	6.4	循环语句 (Loop) 188
4.7	电路架构的描述 (Architecture		6.4.1	for...loop...end loop 188
	Description)	82	6.4.2	while...loop...end loop 189
4.7.1	数据流描述 (Data Flow		6.4.3	loop...end loop 191
	Description)	83	6.4.4	next 191
4.7.2	行为描述 (Behavior		6.4.5	exit 192
	Description)	84	6.4.6	null 192
4.7.3	结构描述 (Structure		6.5	等待语句 (wait) 209
	Description)	86	6.5.1	wait until 条件式 209
4.7.4	元件声明 (Component		6.5.2	wait on 信号 219

6.5.3	wait for 时间	220	8.2	过程 (procedure)	315
6.5.4	assert	220	8.2.1	过程的声明	315
6.6	结论	222	8.2.2	过程的主体 (即过程的定义)	317
第 7 章	以结构风格来描述架构	229	8.3	程序包 (Package)	333
7.1	方块 (Block)	231	8.3.1	程序包的声明 (Package Declaration)	334
7.2	元件 (Component)	248	8.3.2	程序包的主体 (Package Body)	335
7.2.1	元件的设计和声明	249	第 9 章	状态机、计数器、移位寄存器	350
7.2.2	元件连线映射 (Mapping)	250	9.1	Moore 状态机	351
7.2.3	名称映射 (Mapping By name)	250	9.2	Mealy 状态机	361
7.2.4	位置映射 (Mapping By position)	250	9.3	计数器 (Counter)	378
7.3	参数化元件	273	9.3.1	无规则计数器	378
7.4	参数化重复性元件	287	9.3.2	有规则计数器	384
7.4.1	重复性生成语句 (for... generate)	288	9.4	移位寄存器 (Shift Register)	396
7.4.2	条件式生成语句 (if... generate)	288	附录 A	VHDL 的保留字	407
第 8 章	函数、过程和程序包	299	附录 B	完整的系统操作流程	409
8.1	函数 (function)	300	附录 C	命令文件内常用的命令	441
8.1.1	函数的声明	300	附录 D	国内外电气图形符号对照表	457
8.1.2	函数的主体 (即函数的定义)	302			

第 1 章



集成电路设计的发展

- ◎1.1 使用小规模集成电路元件进行设计
- ◎1.2 使用中规模集成电路元件进行设计
- ◎1.3 使用大规模和超大规模集成电路元件进行设计
- ◎1.4 FPGA 元件的规划方式
- ◎1.5 什么是 VHDL

如果您是从事数字电路设计这个领域的前辈，相信您会认同和赞叹科技进步的神速，并对未来数字电路的开发工具抱有无限的期待和好奇之心。

整个数字电路的发展从第一阶段开始，利用晶体管、电阻、二极管等电子元件设计成各式各样的逻辑门，如 NOT、AND、OR、NAND、NOR、XOR、EX-NOR……等小规模集成电路（Small Scale Integrated Circuit, SSI）；第二阶段再以这些基本逻辑门配合卡诺图（Karnough-Map）化简，设计出译码器（Decoder）、分用器（Demultiplexer）、复用器（Multiplexer）、加法器（Adder）、触发器（Flip Flop）、移位寄存器（Shift Register）、计数器（Counter）等中规模集成电路（Medium Scale Integrated Circuit, MSI）；第三阶段再由这些 MSI 慢慢地扩大，成为大规模和超大规模集成电路 LSI、VLSI……等。随着市场需求的快速变化，除了芯片功能的复杂化和多元化之外，产品的研发周期必须大幅度缩短，以保持其竞争力，正所谓“Time to market, time to money（早上市，早赚钱）”。因此早期（即前三个阶段）数字电路的设计方式已经无法满足市场的需求，目前在数字控制电路上所要求的功能，大都通过可编程逻辑设备（Programmable Logic Device, PLD）、现场可编程门阵列（Field Programmable Gate Array, FPGA）、微控制器（Micro Controller）、微处理器（Micro Processor），以及专用 IC（Application Specific Integrated Circuit, ASIC）等芯片来规划完成。

下面我们就概括性地谈谈整个数字电路设计的发展过程，读者可以从中了解各种概念之间的关联性。

1.1 使用小规模集成电路元件进行设计

使用普通逻辑门所构成的小规模集成电路（SSI）来设计数字控制电路，是我们在最为早期所使用的电路设计方式。其设计步骤通常为：

- (1) 描述所要设计的控制电路。
- (2) 画出方框图并决定输入和输出。
- (3) 将输入和输出转换成真值表。
- (4) 用卡诺图将电路简化。
- (5) 得到控制电路。

下面我们就以几个大家熟悉的电路为例来说明，并以此与后面将要使用的 VHDL 语言的设计方式做个比较。

【例题 1.1】 设计一个 2 对 1 的复用器（Multiplexer）。步骤如下：

(1) 描述：

当 $S=0$ 时， $F=A$ 。

当 $S=1$ 时， $F=B$ 。

(2) 方框图（图 1.1）：

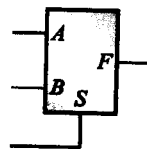


图 1.1 方框图

(3) 真值表 (表 1.1) :

表 1.1 真值表

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

(4) 卡诺图化简 (图 1.2) :

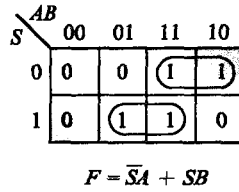


图 1.2 卡诺图化简

(5) 电路 (图 1.3) :

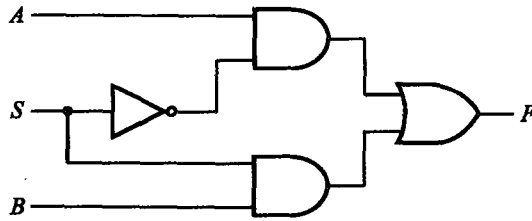


图 1.3 电路

【例题 1.2】 设计一个 2 对 4 高电平有效的译码器。

(1) 描述:

$AB = "00"$, $Y_0 = 1$, 其余为 0。

$AB = "01"$, $Y_1 = 1$, 其余为 0。

$AB = "10"$, $Y_2 = 1$, 其余为 0。

$AB = "11"$, $Y_3 = 1$, 其余为 0。

(2) 方框图 (图 1.4) :

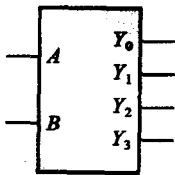


图 1.4 方框图

(3) 真值表 (表 1.2) :

表 1.2 真值表

A	B	Y_0	Y_1	Y_2	Y_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

(4) 卡诺图化简 (此处不需要) :

$Y_0 = \bar{A}\bar{B}$ $Y_1 = \bar{A}B$

$Y_2 = A\bar{B}$ $Y_3 = AB$

(5) 电路 (图 1.5) :

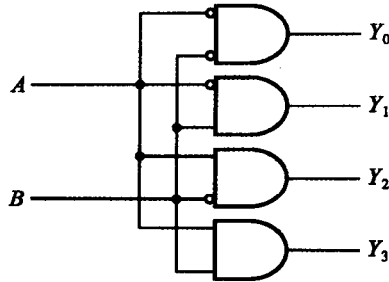


图 1.5 电路

【例题 1.3】 设计 1Bit 的全加器 (Full Adder) 电路。

(1) 描述:

$$\begin{array}{r} C_i \\ X_i \\ + Y_i \\ \hline C_{i+1} S_i \end{array}$$

(2) 方框图 (图 1.6) :

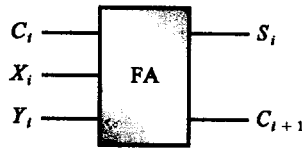


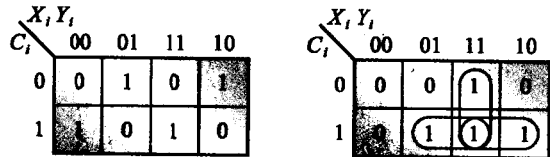
图 1.6 方框图

(3) 真值表 (表 1.3) :

表 1.3 真值表

C_i	X_i	Y_i	S_i	C_{i+1}
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

(4) 卡诺图化简 (图 1.7) :



$$\begin{aligned} S_i &= \overline{C_i}(\overline{X_i}Y_i + X_i\overline{Y_i}) + C_i(\overline{X_i}\overline{Y_i} + X_iY_i) \\ &= \overline{C_i}(X_i \oplus Y_i) + C_i(\overline{X_i}\overline{Y_i}) \\ &= C_i \oplus X_i \oplus Y_i \end{aligned} \quad C_{i+1} = C_i X_i + C_i Y_i + X_i Y_i$$

图 1.7 卡诺图化简

(5) 电路 (参见图 1.8)。

从上述三个例题中我们可以发现, 这种设计方式的缺点为:

- 设计过程较为麻烦。
- 所有元件皆为普通逻辑门 (属于小规模集成电路, 如 SN74XXX 系列的 IC)。
- 当电路功能复杂时, 由于 IC 的数量很多, 造成体积庞大、耗电量高、稳定度低、成本高、速度慢 (延迟时间长) 等不良后果。

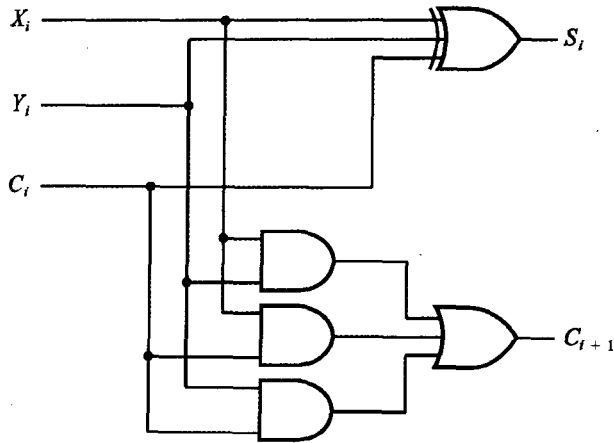


图 1.8 电路

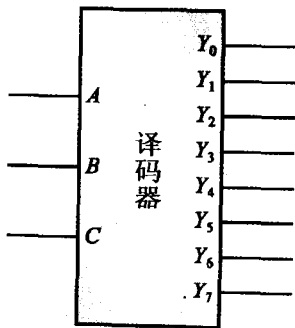
1.2 使用中规模集成电路元件进行设计

由上面所得到的结论可知，由 SSI 组合出来的电路，其体积较为庞大且电路较不稳定。另外从例题 1.2 中我们又发现，一个译码器的每一个输出皆为其输入所对应的布尔积 (Minterm)，因此我们只要在一个译码器 (中型 ICMSI) 的输出端加上一个多输入的 OR 门，即可实现我们所需要的任何组合控制电路。

下面我们就举例来说明。

【例题 1.4】 利用 3 对 8 的译码器设计一个 1 Bit 的全加器。

(1) 3 对 8 译码器，其每一个输出皆对应其输入的布尔积 (图 1.9)，即：



$$\begin{aligned}
 Y_0 &= \overline{A}\overline{B}\overline{C} & Y_1 &= \overline{A}\overline{B}C \\
 Y_2 &= \overline{A}B\overline{C} & Y_3 &= \overline{A}BC \\
 Y_4 &= A\overline{B}\overline{C} & Y_5 &= A\overline{B}C \\
 Y_6 &= AB\overline{C} & Y_7 &= ABC
 \end{aligned}$$

图 1.9 方框图

(2) 上面的例题 1.3 中，一个全加器 FA 的和 \$S_i\$ 及进位 \$C_{i+1}\$ 未经化简的布尔代数分别为：

$$\begin{aligned}
 S_i &= \overline{C_i}\overline{X_i}Y_i + \overline{C_i}X_i\overline{Y_i} + C_i\overline{X_i}\overline{Y_i} + C_iX_iY_i \\
 C_{i+1} &= \overline{C_i}X_iY_i + C_i\overline{X_i}Y_i + C_iX_i\overline{Y_i} + C_iX_iY_i
 \end{aligned}$$

- (3) 综合上面两个式子，我们可以得到如图 1.10 所示的等效电路（详细的设计原理请自行参阅逻辑设计的书籍，本书的重点并非在此，故不做详细的叙述）。

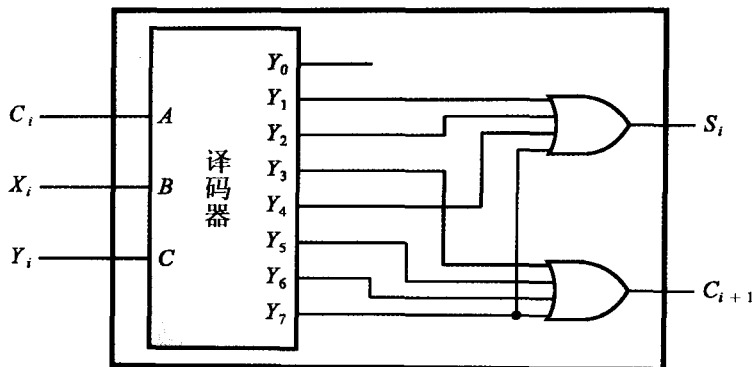


图 1.10 1 Bit 全加器

从上述的电路中我们可以发现，以一个译码器来设计组合逻辑时，我们只要将原来组合电路输出为 1 的布尔积（即译码器的输出）作或运算即可，这种原理我们也可以应用在复用器 Multiplexer 上。

【例题 1.5】 用一个 4 对 1 的复用器（中型 IC MSI）来设计 1 Bit 全加器的和 S_i 。

- (1) 一个 4 对 1 复用器的方框图及动作状况如图 1.11 所示：

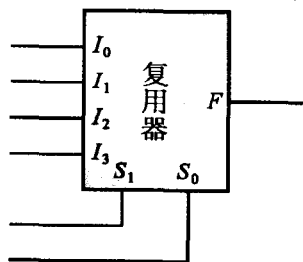


图 1.11 方框图

当： $S_1 S_0 = "00"$ 时， $F = I_0$
 $S_1 S_0 = "01"$ 时， $F = I_1$
 $S_1 S_0 = "10"$ 时， $F = I_2$
 $S_1 S_0 = "11"$ 时， $F = I_3$

- (2) 全加器输出端和 S_i 的真值表如表 1.4 所示：

表 1.4 真值表

	S_1	S_0	I	F	
	C_i	X_i	Y_i	S_i	
0	0	0	0	0	$\Rightarrow Y_i$
	0	0	1	1	
1	0	1	0	1	$\Rightarrow \overline{Y_i}$
	0	1	1	0	
2	1	0	0	1	$\Rightarrow Y_i$
	1	0	1	0	
3	1	1	0	0	$\Rightarrow \overline{Y_i}$
	1	1	1	1	

- (3) 将复用器的选择线 S_1 、 S_0 当成全加器 C_i 、 X_i 的输入，且输入线 $I_0 \sim I_3$ 分别接到 Y_i 、 \bar{Y}_i 、 \bar{Y}_i 、 Y_i 。其引脚情况如图 1.12 所示。

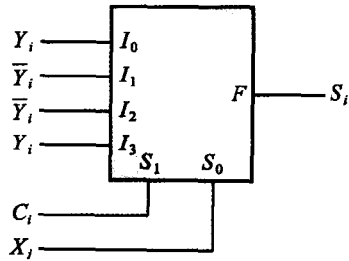


图 1.12 引脚图

- 从步骤 (1) 中得知，当 $S_1 S_0 = "00"$ 时， $F = I_0$ 。此时在上图中 $C_i X_i = "00"$ 时， $S_i = Y_i$ ，即若 $Y_i = 0$ 则 $S_i = 0$ ；若 $Y_i = 1$ 则 $S_i = 1$ （符合步骤 (2) 中全加器真值表中的“0”部分）。
- 从步骤 (1) 中又得知，当 $S_1 S_0 = "01"$ 时， $F = I_1$ 。此时在上图中 $C_i X_i = "01"$ 时， $S_i = \bar{Y}_i$ ，即若 $Y_i = 0$ 则 $S_i = 1$ ；若 $Y_i = 1$ 则 $S_i = 0$ （符合步骤 (2) 中全加器真值表中的“1”部分）。
- 依此类推我们可以得知，步骤 (3) 的电路动作状况跟全加器的和 S_i 完全相同。

从上述两个例题得到印证，我们可以利用译码器或复用器这类中规模集成电路，来实现组合电路的设计。

1.3 使用大规模和超大规模集成电路元件进行设计

从 1.2 节我们可以得到一个很重要的结论，一个元件只要其输出端拥有输入端所对应的全部布尔积 (Minterm)（如 2 对 4 的译码器，其两个输入所对应的 4 个布尔积皆在其输出端），只要利用或门，即可将其规划成任何一种组合逻辑电路。基于这种观念，可编程逻辑设备 (Programmable Logic Device, PLD) 元件即应运而生。硬件工程师可以利用软件工具将手上的 PLD 元件规划成自己所需要的硬件控制电路。当然，这些元件的特性也会随着科技的进步而得到改善。有关 PLD 的产品，从早期的 PROM、PLA、PAL、GAL 到 PEEL，不论是执行速度，还是烧录方式和烧录次数，皆有长足的进步，但不管它的材料如何改变，其所遵循的原理皆相同。下面我们就来简单地介绍这些元件的内部架构。

1.3.1 PROM (可编程 ROM)

可编程只读存储器 (Programmable Read Only Memory, PROM) 是最早出现的元件，其原理就是我们前面所叙述的，它是利用每个 ROM 内部的译码器电路，并在后面加入一个可以规划的 OR 电路，其架构如图 1.13 所示。