4 微电子学讲座

树下行三
〔日〕浅田邦博 著
唐津 修

超大规模集成电路设计 Ⅱ

逻辑与测试

科学出版社

微电子学讲座 4

超大规模集成电路设计 II

逻辑与测试

〔日〕树下行三　浅田邦博　唐津　修 著

裴武焕 译

洪先龙 校

科 学 出 版 社

1991

（京）新登字 092 号

内 容 简 介

本书是微电子学讲座第四卷．本丛书第三、四卷系统地介绍了超大规模集成电路的设计方法．本卷重点介绍计算机辅助设计技术在逻辑设计及测试中的应用．

本书共六章．第一章是全书的基础部分，主要介绍布尔代数．第二至六章分别介绍逻辑设计基础、描述语言、逻辑模拟及超大规模集成电路的测试(包括测试系列的生成及易测性设计)．

本书内容新、实用性强，既可作为大专院校微电子学、电子工程、计算机应用专业的教学参考书或教学用书，也可供从事大规模集成电路设计、计算机辅助设计工作的工程技术人员、科研人员参考．

樹下行三　浅田邦博　唐津　修 著

岩波講座マイクロエレクトロニクス 4

VLSI の設計 II

論理とテスト

岩波書店，1985

微电子学讲座 4

超大规模集成电路设计 II

[illegible]行三　浅田邦博　唐津　修 著

[illegible]

[illegible]版

[illegible]

[illegible]印刷

新华书店北京发行所发行　各地新华书店经售

*

1991 年 12 月第 一 版　开本：850×1168　1/32

1991 年 12 月第一次印刷　印张：9 1/4

印数：0001—1 700　字数：239 000

ISBN 7-03-002457-5/TN · 104

定价：10.60 元

原 编 者 的 话

随着大规模集成（LSI）电路技术的不断发展，迎来了超大规模集成（VLSI）电路时代．现在，一个大规模的系统已经能够集成在一个 VLSI 芯片内．因而，面向 VLSI 的系统设计技术以及把这些 VLSI 组合起来构成巨大系统的技术，在目前均被看作是微电子学的一部分．

本讲座中所介绍的微电子学就是这种广义的微电子学，VLSI 的发展将给今后的社会带来巨大的影响．因而，本讲座的目的就是要集中介绍设计制作 VLSI 及由 VLSI 构成计算机、通信、机械电子学等信息系统所必需的理论和技术．

然而，微电子学是 LSI 技术诞生不久后出现的发展十分迅速的领域，至于它将以何种形态向前发展，其可能性是很多的．所以，若想把它汇总成一个完整的学科系统还有许多困难．因此，本讲座的目的是：

（1）向关心微电子学的初学者和准备应用微电子技术的人们介绍有关微电子学的基础知识．

（2）为在计算机、通信、机械电子学领域工作的技术人员、研究人员提供实践指南．

（3）把掌握微电子学所必需的基础知识和技术明确化．

（4）把微电子学作为一门学科进行系统介绍，并使其成为迈向理想教学用书的里程碑．

希望本讲座能成为学生、技术人员、研究人员的良师益友．

元冈　达　管野卓雄　渡边　诚

渊　一博　石井威望

前　言

本书是微电子学讲座的第四卷，重点阐述超大规模集成电路的计算机辅助设计（CAD）中有关逻辑设计及测试的内容。在VLSI电子设备的CAD中，逻辑设计及测试的CAD比第三卷中所介绍的电路、版图设计的CAD内容新，而且即将得到实际应用。从这个意义上说，与其说本书说明了各种成熟的方法，倒不如说它介绍了这一领域的现状和今后的发展方向。因此，对上述内容感兴趣的技术人员及研究人员，如果能从本书中受到一些启发，也就算达到了我们编写本书的目的。

以往的系统设计及逻辑设计，绝大部分要靠人工去完成，所以，被认为是最能发挥设计者主观能动性的工作。可是，VLSI设计要求人们在短期内准确无误地完成大规模的逻辑设计。因而，迫切要求实现逻辑设计的自动化。

第一章是全书的基础部分。主要叙述布尔代数。众所周知，布尔代数是逻辑设计的数学基础。这部分没有什么新内容，只是考虑到本书的完整性，简要地介绍了一些布尔代数的基本内容。

第二章是逻辑设计部分。逻辑设计方法如同布尔代数一样，早已问世。随着不同时期所用元件的差异，设计目标也有所变化。本书从VLSI逻辑设计的观点出发，介绍了许多面向PLA（Programmable Logic Array）和VLSI系统设计的最新方法。

第三章介绍了描述语言。如果说第一、二章是辑逻设计的算法篇，那么这一章便是逻辑设计的表现方法篇。在逻辑设计的CAD中，如何利用计算机进行逻辑设计，很大程度上将取决于描述方法。如同程序语言一样，描述方法是反映人们的思考方法的，所以究竟用哪一种描述方法，需视所设计的课题而定。目前提出的描述方法较多，在此仅从VLSI的逻辑设计描述语言的角度着

重说明结构描述语言和功能描述语言，并分别以 HSL 和它的扩展语言 HSL-FX 作为例子.

第四章是逻辑模拟部分. 逻辑模拟是验证逻辑设计是否正确的重要手段. 变更集成电路的设计所需费用很大. 目前，在还没有实现设计的全部自动化，而仅仅以采用计算机辅助设计为主要手段的情况下，逻辑模拟便是验证逻辑功能的重要途径.

第五章及第六章是有关测试部分. 过去是在制造产品之后对成品进行测试的，在设计阶段还谈不上测试. 然而，VLSI 中作为测试对象的元件和设备的规模越来越大，数目也越来越多，而且又不能直接对其内部进行测试，因此给测试工作带来了很大困难.

这里所说的测试，是指检验装置是否能按设计要求正确地进行工作. 随着电路的 VLSI 化，集成电路测试的困难在于求测试码模式及用该测试码模式进行测试都需要很长的时间. 这两者之间又是互相牵制的，减小一方势必增大另一方.

第五章是故障测试系列的生成部分. 重点介绍有关测试的基本概念及目前常用的测试码模式的生成方法. 由于 VLSI 功能的日趋复杂和元件数目的不断增加，到目前为止还没有找出更为实用而且效率较高的一种测试方法. 因此，在设计阶段就应预先考虑其测试问题，并把测试这一内容作为设计的重要内容. 目前，在 CAD 中测试所占的比例越来越大.

第六章从测试的难度出发，阐述了一些容易进行测试的设计方法(以下简称易测性设计). 其中有些虽已达到实用的阶段，但仍有必要进行深入的探讨. 在此，仅介绍一些基本的方案和现状以及它的发展动向. 从逻辑设计的 CAD 角度来看，这一问题的重要性也是显而易见的.

以上就是本书的写作思想及内容概要. 但是，急速发展的 VLSI 时代使我们很难准确地介绍 VLSI 设计的现状. 仅从本书开始写作到定稿这一段时间，就已产生了许多新方案. 这就是 VLSI 的现状. 所以不能说本书全部包括了最近的新方案，但是可以认为，VLSI 时代所应包括的问题，即逻辑设计和测试的基

本方向及内容已写入了本书中．相信本书作为开拓新时代的微电子学的一部分，会作出自己应有的贡献。

各章的执笔者如下：

第一章	布尔代数	浅田邦博
第二章	逻辑设计基础	浅田邦博
第三章	描述语言	唐津　修
第四章	逻辑模拟	树下行三
第五章	故障测试序列的生成	树下行三
第六章	易测试性设计	树下行三

最后，对在我们编写及出版本书过程中给于鼓励和帮助的岩波书店编辑部的各位表示衷心的感谢。

树下行三

浅田邦博

唐津　修

1985年3月

目　　录

第一章 布尔代数

在数字系统中，以“0”和“1”两个状态的组合来表示、处理数量。因此，在设计构成数字系统的大规模集成电路（LSI）时，首先必须了解数的表示方法，其次要了解其处理方法。布尔代数提供了由“1”和“0”两个符号所组成的二值系统的数学基础。数的运算将用布尔代数的函数（逻辑函数）形式加以描述，LSI 的设计最终也将用逻辑函数决定其结构形式。关于在 LSI 中如何实现用逻辑函数表示的运算，将在下一章叙述。在研究逻辑函数的阶段中，最重要的是找出等效变换为最实用的电路形式的问题。基于这种考虑。本章主要介绍数的表示法、布尔代数的基础知识和逻辑函数的简化等内容。

1.1 数的表示

1.1.1 模拟量和数字量

在电子线路中，两点间的电压和电流是主要的状态量。它们都是连续量，在数学中用实数表示连续量。若把电压和电流表示成实数，则可利用电子线路的一些固有特性，对它们进行计算。不仅加减运算，微积分也很容易计算，若用非线性电子线路，还可以进行乘除运算。若用电压值和电流值来代表实数，我们就可以解微积分方程，这种方法被称作模拟计算方法，电压值和电流值这样的连续变化的量被称之为模拟量。在电子线路中，模拟计算方法容易实现，速度也较快，所以曾被广泛使用过。而目前，除一些小型设备外，这种方法已不大被使用了。这主要是由于在实际过程中用模拟量表示数值不够稳定。目前对于复杂的数据处理，要求反复进行几万次以上的运算操作，所以不倾向于用模拟量表示。为了

说明这个问题，我们不仿举一个用数值表示电压的例子：若想进行几万次以上的运算，就应当在一定时间里存贮该数值．然而，目前模拟电压保持电路以 10^{-4} 的精度长时间保持是很困难的，并且，由于电路中元件的精度和热稳定性等因素，在一次运算中就很容易产生 10^{-4} 程度的运算误差．由此可见，对于这种经几万次以上的运算和反复存贮所产生的结果，其可靠性究竟如何就可想而知了．

为克服模拟计算方法的这些缺点，目前，在数字系统中采用了数字表示法．要用数字表示电子线路中的电压和电流，须对电压和电流设若干个阈值，并分成有限个区间．在一个区间里，将把电压和电流看成是处于一种状态．在该区间里，位于中心处的电压、电流值有时不容易受干扰．这种数字表示方法的特点是，虽然外界的干扰和某些误差可能会使这一数值有所变动，但均可以用某些方法使其恢复．在模拟方法中是做不到这一点的．这种恢复操作的可能性就使电压、电流的数值可以长时间地保持稳定，并且，在运算中产生误差的可能性也极大地降低了．

目前的数字系统，几乎都采用二值数值表示法．在电子线路中，用设阈值的办法很容易实现这一点．上面所说的二值状态通常用 1 和 0 表示，1 和 0 究竟分别代表哪一状态则是任意的．一般用 1 来表示高电平和大电流的状态，并把这种逻辑称为正逻辑．

前面我们已经指出，电子线路的电压和电流是连续变化的物理量．但正如量子力学所说明的，在特殊的情况下，也可将其看做是离散值．正在研究的一种未来的高速器件——约瑟夫森器件就是一例．在磁量子型线路中，将利用离散电流值的性质来构成逻辑电路．在这些情况下，一般都用数字法．但数字法的缺点是用一个量只能表示有限个状态(一般是二值状态)．

1.1.2 二进制

在数字系统中，用电压或电流等物理量只能表示两个状态，因此用增加位数的办法来表示“所必需的数的范围和精度”．日常生活中用 0—9 十个数符表示数的方法称作十进制．与此相对应，只

用 0 和 1 两个数符表示数的方法被称作二进制．用二进制表示某一正实数 R 时，可用下式展开：

$$R = \sum_{i=-\infty}^{M-1} b_i \times 2^i \quad (\text{其中 } b_i = 1 \text{ 或 } 0) \tag{1.1}$$

(这里所说的"正"包括 0)由(1.1)式可以得到一个无穷的二进制数列 $(b_{M-1}b_{M-2}\cdots b_0 b_{-1}\cdots)$，实际的位数由所需要的精度来决定．和十进制数一样，b_{M-1} 到 b_0 的 M 位称为整数部分，b_{-1} 以下的部分称为小数部分，小数点一般在 b_0 和 b_{-1} 之间，这一点是大家所公认的，不需加以说明．在二进制中，若设整数部分为 M 位，小数部分为 N 位，则所表示的数的范围为 0 至 $(2^M - 2^{-N})$，其精度为 2^{-N}．

由式 (1.1) 知，欲将某一正实数转换成二进制数 $(b_{M-1}b_{M-2}\cdots b_0 b_{-1}\cdots)$，其方法如下：将原来的实数分成整数部分和小数部分，将其整数部分依次除 2，取其余数，得到 $b_0 b_1 \cdots b_{M-1}$；将其小数部分依次乘 2，取其积的整数部分，得到 $b_{-1}b_{-2}\cdots$．其转换过程如图 1.1 所示．在这种转换中，即使是一个有限位的十进制数，在二进制中也可能成为无限循环的小数．所以用有限位的二进制数表示上述十进制数时，就可能出现因转换而产生的误差，这一点必须注意．

正的实数有时可能是正的整数，这时由式 (1.1) 可知，当 i 为负数时 b_i 等于零，所以可省略．$b_{M-1}\cdots b_0$ 的 M 位所能表示的整数

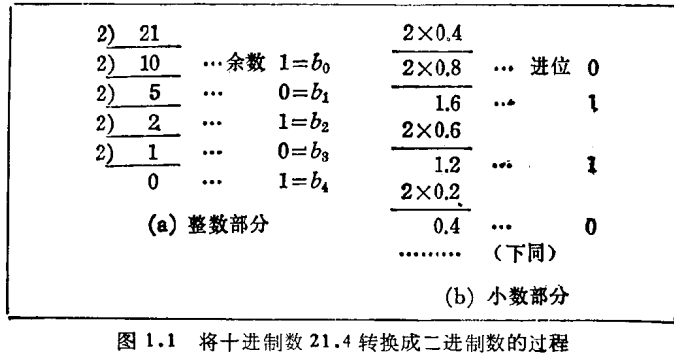

图 1.1 将十进制数 21.4 转换成二进制数的过程
(结果为10101.01100110…)

的范围为 0 至 $(2^M - 1)$.

1.1.3 负数的表示

表示负数的方法有几种. 一种是在前述的正数的二进制表示中多设一位"符号位". 对于符号位一般用 0 表示正,用 1 表示负. 反之亦然. 这种方法虽然简单,但要注意当数为 0 时,符号位可能有两种情况.

第二种方法是补偿法 (offset). 在负数上加一个绝对值比该负数大的正数, 把负数变成正的二进制数. 比如对 $(2^{M-1} - 1)$ 至 -2^{M-1} 范围内的数,加上 2^{M-1}, 即可变成 $(2^M - 1)$ 至 0 范围内的正数.

第三种是补码表示法. 补码加它的原码是一个恒定值, 称为模. 令 P 为一恒定值, 若正数 R 的补码为 C, 则 $C = P - R$. 为了不使 $P - R$ 和 R 是相同数, 要适当规定 P 值. 如 R 是

表 1.1 数的表示(四位二进制整数)

十进制数	有符号位数	补偿法	模 1 补码	模 2 补码
−8	* * * *	0 0 0 0	* * * *	1 0 0 0
−7	1 1 1 1	0 0 0 1	1 0 0 0	1 0 0 1
−6	1 1 1 0	0 0 1 0	1 0 0 1	1 0 1 0
−5	1 1 0 1	0 0 1 1	1 0 1 0	1 0 1 1
−4	1 1 0 0	0 1 0 0	1 0 1 1	1 1 0 0
−3	1 0 1 1	0 1 0 1	1 1 0 0	1 1 0 1
−2	1 0 1 0	0 1 1 0	1 1 0 1	1 1 1 0
−1	1 0 0 1	0 1 1 1	1 1 1 0	1 1 1 1
0	0 0 0 0	1 0 0 0	0 0 0 0	0 0 0 0
1	0 0 0 1	1 0 0 1	0 0 0 1	0 0 0 1
2	0 0 1 0	1 0 1 0	0 0 1 0	0 0 1 0
3	0 0 1 1	1 0 1 1	0 0 1 1	0 0 1 1
4	0 1 0 0	1 1 0 0	0 1 0 0	0 1 0 0
5	0 1 0 1	1 1 0 1	0 1 0 1	0 1 0 1
6	0 1 1 0	1 1 1 0	0 1 1 0	0 1 1 0
7	0 1 1 1	1 1 1 1	0 1 1 1	0 1 1 1

$(2^{M-1}-2^{-N})$至 0 范围内的一个数时，P 的数要大于 (2^M-2^{-N}). 一般 P 等于 (2^M-2^{-N}) 或者 2^M. 前者称为模 1 的补码，后者称为模 2 的补码. 模 1 的补码也称反码. 反码就是把原码的 0 和 1 反过来. 而模 2 的补码显然可以由反码末位加 1 (即对 N 位来说加 2^{-N}) 求出. 另外，对反码来说，0 的补码是 (2^M-2^{-N}). 与用符号位的情况一样，0 的表示有冗余位. 而对模 2 的补码来说，0 的补码为 2^M，有 $(M+1)$ 位整数部分. 若第 $(M+1)$ 位的溢出部分可以忽略，则 0 的表示可以单意地确定.

上述三种方法中，整数位为 M 位(有符号位时包括符号位)，小数部为 N 位，在补偿法和模 2 补码表示法中，数的范围为 $[2^{M-1}-2^{-N}$ 至 $-2^{M-1}]$，在其它方法中，数的范围为 $[2^{M-1}-2^{-N}$ 至 $-(2^{M-1}-2^{-N})]$. 表 1.1 为 $M=4, N=0$ 时的各种表示法.

1.1.4 二进制数的运算

二进制数的加法运算与十进制数的加法运算基本相同. 但由于二进制数只有两个数符，即 0 和 1，所以它的运算规则只有 4 种(表 1.2). 图 1.2 是 30.5 与 10.25 的二进制数运算例子.

表 1.2 二进制数加法运算规则

a	b	和	进 位
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

```
  11110.10 ……十进制数 30.5
+)1010.01 ……十进制数 10.25
-----------
 101000.11 ……十进制数 40.75
```

图 1.2 30.5+10.25 的二进制数运算例子

减法运算用模 2 的补码比较方便，它将减法运算变成加法运算. 由模 2 补码定义可知，$a-b$ 的减法运算可写为 $a+(2^M-$

$b) = 2^M + (a - b)$. 如整数部分为M位，则2^M自然溢出可以忽略不计，所以结果就是 $a - b$. 由此可看出补码的优点.

二进制数的乘除法运算也比十进制数的乘除法运算简单得多. 设R为乘数，S为被乘数，则其乘积 $T = R \times S$:

$$T = R \times S = \left(\sum_{i=-N}^{M-1} r_i \cdot 2^i\right) \times \left(\sum_{j=-N}^{M-1} s_j \cdot 2^j\right)$$

$$= \sum_{i=-N}^{M-1} r_i \left(\sum_{j=-N}^{M-1} s_j \cdot 2^{j+i}\right) \tag{1.2}$$

对每一个 i, 被乘数S乘以 2^i，当 r_i 为1时只进行加法运算，最后得到结果（2^i倍可用移位的办法得以实现）. 其中$\{s_i\}$, $\{r_i\}$是S和R的二进制表示. 由式(1.2)可知，乘积T的整数部分可能为$2M$位，小数部分为$2N$位，所以要注意可能产生的溢出问题. 这里

```
    11110.1  ……十进制数 30.5
×)  11.01  ……十进制数 3.25
-----------
   11110 1
  111101
 111101
-----------
1100011.001 ……十进制数 99.125
```

图 1.3 30.5 × 3.25 的二进制数乘法运算的例子

```
          1001.011…
       ------------
11.01)11110.1
      1101
      ----
       10010
       1101
       ----
        10100
        1101
        ----
         1110
         1101
         ----
          ⋮
```

图 1.4 30.5÷3.25 的二进制数除法运算的例子

我们设 R 和 S 为正数，如是负数时进行绝对值相乘，其符号有必要单独进行运算，所以要设符号位．除法运算也和十进制数的除法运算相同．图 1.3 和图 1.4 是乘除运算的例子．

1.1.5 浮点表示法

前面所说的二进制数的小数点位置是固定的，所以称为定点表示法．定点表示法的缺点是数的表示范围小，如整数部分为 M 位时，其表示范围为 2^{M-1} 至 -2^{M-1}．为克服这一缺点，采用了浮点表示法．把一个数分成两部分，一部分称为尾数部分，一部分称为指数部分，也称阶码部分．用阶码部分表示小数点的位置，原来的数 R 则为尾数部分，表示小数点位置的数 E 称为指数部分．因此，数的大小为 $R \times B^E$．其中 B 可以是任意自然数，但为了 '$\times B^E$' 乘法运算的方便(用移位法)，一般用 2 或16等 2 的整数倍．

1.1.6 其它的二进制数的表示法

用 0 和 1 两个数符表示二进制数的方法还有一些．原则上，能表示 L 位的二进制数值的种类有 2^L 种．与数值相对应的方法有 $2^L!$ 种．但是在数字系统中并非所有这些方法都很方便．一般来说，在数字系统中进行二进制运算时，较方便的方法是上面所说的几种．

相反，数字系统与外界进行数字交换时，又很难说比较方便的只是上述那几种表示方法．比如表 1.3 中的格雷码就是数字系统与外界进行数字交换时较方便的一种方法，它的特征是两个相邻数的二进制表示中只有一位不同．所以当数值发生变化时，若由外界输入该数值，不确定的位只有一个，故可保证产生的误差最小．如果用一般的二进制数表示法，有可能在所有位产生变化时进行输入，所以产生的误差可能最大．

人们在社会上所用的是十进制表示法，而人与数字系统之间还要进行数据的交换，所以必须进行二进制表示与十进制表示的转换，这就是表 1.3 中所示的二-十进制表示法，即 BCD 码．这

表 1.3 数的各种表示方法

十进制数	二进制数	格雷码	二-十进制数 (BCD 码)	余 3 BCD 码
0	0	0	0 0 0 0	0 0 1 1 0 0 1 1
1	1	1	0 0 0 1	0 0 1 1 0 1 0 0
2	1 0	1 1	0 0 1 0	0 0 1 1 0 1 0 1
3	1 1	1 0	0 0 1 1	0 0 1 1 0 1 1 0
4	1 0 0	1 1 0	0 1 0 0	0 0 1 1 0 1 1 1
5	1 0 1	1 1 1	0 1 0 1	0 0 1 1 1 0 0 0
6	1 1 0	1 0 1	0 1 1 0	0 0 1 1 1 0 0 1
7	1 1 1	1 0 0	0 1 1 1	0 0 1 1 1 0 1 0
8	1 0 0 0	1 1 0 0	1 0 0 0	0 0 1 1 1 0 1 1
9	1 0 0 1	1 1 0 1	1 0 0 1	0 0 1 1 1 1 0 0
10	1 0 1 0	1 1 1 1	0 0 0 1 0 0 0 0	0 1 0 0 0 0 1 1
11	1 0 1 1	1 1 1 0	0 0 0 1 0 0 0 1	0 1 0 0 0 1 0 0
12	1 1 0 0	1 0 1 0	0 0 0 1 0 0 1 0	0 1 0 0 0 1 0 1
13	1 1 0 1	1 0 1 1	0 0 0 1 0 0 1 1	0 1 0 0 0 1 1 0
14	1 1 1 0	1 0 0 1	0 0 0 1 0 1 0 0	0 1 0 0 0 1 1 1
15	1 1 1 1	1 0 0 0	0 0 0 1 0 1 0 1	0 1 0 0 1 0 0 0
16	1 0 0 0 0	1 1 0 0 0	0 0 0 1 0 1 1 0	0 1 0 0 1 0 0 1
17	1 0 0 0 1	1 1 0 0 1	0 0 0 1 0 1 1 1	0 1 0 0 1 0 1 0
18	1 0 0 1 0	1 1 0 1 1	0 0 0 1 1 0 0 0	0 1 0 0 1 0 1 1
19	1 0 0 1 1	1 1 0 1 0	0 0 0 1 1 0 0 1	0 1 0 0 1 1 0 0
20	1 0 1 0 0	1 1 1 1 0	0 0 1 0 0 0 0 0	0 1 0 1 0 0 1 1
⋮	⋮	⋮	⋮	⋮

时也可能产生不必要的转换误差. 它用四位二进制数表示一位(digit)十进制数，其特点是与十进制数之间的转换比较容易，但四则运算却比纯二进制数麻烦. 比如，BCD 码如果完全按十进制数加法运算规则运算，则在 10 以上的各位将产生进位操作，所以还要减去 10. 四位二进制数可表示数 0 到 15，所以减 10 的操作利用补码规则可变成加 6 的操作. 因此可采用在 BCD 码上预先加 3 的码，这称为余 3 BCD 码. 在余 3 BCD 码中，进位处理完全与一般的二进制相同.