高等工科院校
计算机专业系列教材

彭建朝　编著

数字电路的逻辑分析与设计

SHUZI DIANLU De
LUOJI FENXI YU SHEJI

北京工业大学出版社

数字电路的逻辑分析与设计

彭建朝　编著

北京工业大学出版社

内 容 简 介

本书结合现代数字系统设计技术的发展，系统地介绍了数字电路逻辑分析与设计的基本理论、基本方法以及基于硬件描述语言 Verilog HDL 的建模技术等。

全书共有 8 章。第 1 章～第 3 章讲述数字逻辑的理论基础，包括数制、码制、逻辑代数基础以及硬件描述语言基础等；第 4 章介绍了组合电路中常用逻辑功能电路的设计思想、分析方法、Verilog HDL 建模方法以及 MSI 器件的应用；第 5 章～第 8 章在分析锁存器、触发器工作原理和逻辑特性的基础上，讨论了同步时序电路的分析方法与设计技术，特别是同步时序电路的 Verilog HDL 建模技术。

本书可作为计算机科学与技术、自动控制、电子信息等专业的本科生教材，也可作为数字系统设计相关技术人员学习 Verilog HDL 建模方法的参考书。

图书在版编目（CIP）数据

数字电路的逻辑分析与设计/彭建朝编著. —北京：北京工业大学出版社，2007.9

ISBN 978-7-5639-1822-5

Ⅰ.数… Ⅱ.彭… Ⅲ.数子电路-逻辑设计-高等学校-教材 Ⅳ.TN79

中国版本图书馆 CIP 数据核字（2007）第 117577 号

数字电路的逻辑分析与设计

彭建朝 编著

*

北京工业大学出版社出版发行

邮编：100022 电话：（010）67392308

各地新华书店经销

徐水宏远印刷厂印刷

*

2007 年 9 月第 1 版 2007 年 9 月第 1 次印刷

787mm×1092mm 16 开本 18 印张 443 千字

印数：1～3 000 册

ISBN 978-7-5639-1822-5/G·890

定价：28.00 元

前　言

目前，在数字系统的工程设计中，已广泛采用基于硬件描述语言（HDL）、电子设计自动化（EDA）平台和可编程逻辑器件（PLD）的现代数字系统设计技术，而基于特定功能集成电路器件“搭积木”式的传统设计技术正在成为历史。因此，以数字电路的分析、设计和应用为主要教学内容的“数字逻辑”课程，必须改革陈旧的以中、小规模集成电路为核心的教学内容、教学方法和实验手段，引入基于硬件描述语言的“自主”芯片设计方法和以“自主”芯片为核心的数字电路设计技术，强化基于 EDA 平台的实验环节，才能适应现代数字系统设计技术发展的要求，并为后续课程（数字系统设计、计算机组成原理、微机接口技术、计算机体系结构以及嵌入式工程方法等）的学习打下坚实的基础。

《数字电路的逻辑分析与设计》是教学改革、“精品课程建设”中数字逻辑课程的配套教材。具有以下突出特点：

1. 在确保基本知识、基本理论、基本分析方法和基本设计方法的前提下，以逻辑分析和逻辑设计为主线，对传统的教学内容进行了优化、精简。例如，删除了逻辑函数化简中的 Q－M 法；淡化了多输出逻辑函数的化简；删除了不完全给定同步时序电路的状态化简；淡化了状态分配以及基于 JK 触发器设计的激励表法、卡诺图分区法。另外，由于硬件描述语言的模块化设计方法可以将异步时序电路的设计转化为同步时序电路设计，因此本书没有涉及异步时序电路的分析与设计等内容。

2. 在组合电路、同步时序电路的分析与设计中，合理渗透现代数字系统设计技术的新概念、新理论、新方法和新技术，并全面引入硬件描述语言——Verilog HDL 建模技术。本书专门用一章的篇幅讲述面向可综合逻辑电路的 Verilog HDL 基础知识、语法规范以及门级描述、数据流描述、行为描述等建模方法。在讲述数字电路的基本分析方法的基础上，讨论了如何对以中、小规模逻辑器件构成的组合电路、同步时序电路进行逻辑抽象并转化为 Verilog HDL 模型；在讲述采用中、小规模逻辑器件构造组合电路、同步时序电路的基本设计技术的同时，更侧重讨论基于 Verilog HDL 的建模技术和方法。其目的在于，吸收传统分析方法和设计技术的精华，摆脱传统逻辑器件的束缚，适应现代数字系统设计的要求，培养创新意识和“自主”芯片的设计能力。

3. 数字逻辑课程具有较强的实践性特征。为此，本书中所有 Verilog HDL 模型均在 EDA 设计平台上进行了仿真验证，并给出了相当数量的仿真波形，以加深对设计描述的理解。另外，通过与“数字逻辑实验”课的紧密结合，可在理论学习的基础上，进一步培养学生动手能力，建立电路物理实现的真实感受，激发数字系统硬件设计与研究的兴趣。

4. 本书的每章都有本章小结和思考题。在对本章知识点进行总结归纳的基础上，通过思考题达到发散性思维和自主性学习的目的。

本书共分 8 章。

第 1、2 章是数字逻辑的基础部分，讨论了数字系统设计必备的数制与码制、逻辑代数基础等。主要内容有十进制、二进制、八进制和十六进制数之间的转换；带符号二进制数的

机内表示（原码、反码、补码）；常用 BCD 码；可靠性编码；逻辑代数的基本概念、基本定理和规则；逻辑函数的基本表达形式以及逻辑函数的卡诺图化简法等。

第 3 章讲述了面向可综合逻辑设计的 Verilog HDL 基础知识。包括模块的概念、基本语法、行为语句以及三种基本描述方式——门级结构描述、数据流描述和行为描述。其中的行为描述是讨论的重点。关于 Verilog HDL 的其他建模技术放在数字系统设计课程中讲述。

第 4 章是组合电路的逻辑分析与设计。在组合电路的逻辑分析中，除介绍基于逻辑门、常用逻辑器件的组合电路的分析方法外，进一步讨论了如何对电路功能进行逻辑抽象并转换为 Verilog HDL 模型；在组合电路的逻辑设计中，讲述了加法器、比较器、编码器、译码器、数据选择器、数据分配器、奇偶校验器等常用组合电路的设计思想、设计方法、MSI 器件以及 Verilog HDL 建模方法。最后讨论了组合电路中的险象问题。

第 5 章介绍时序电路中的双稳态元件——锁存器与触发器。从发现问题、分析问题和解决问题的角度，重点讨论 *RS* 锁存器、*D* 锁存器、*JK* 锁存器、主从 *JK* 触发器、边沿 *JK* 触发器、边沿 *D* 触发器的外部逻辑功能、触发方式以及 Verilog HDL 模型，而对内部电路结构仅作简单介绍。

第 6 章为同步时序电路的分析。通过时序电路的结构模型，建立相关的概念；给出同步时序电路的分析方法，并对基于触发器的同步时序电路进行实例分析，同时讨论如何建立与其对应的 Verilog HDL 模型；还研究了时序电路中的“挂起”现象及消除方法。

第 7 章讲述了计数器、寄存器、移位寄存器、移位寄存器型计数器、节拍分配器和序列信号发生器等典型同步时序电路的设计方法、Verilog HDL 模型以及有关 MSI 时序器件的应用。

第 8 章重点讨论一般同步时序电路设计中原始状态图的建立、状态的化简以及状态的分配，并给出了完整的设计实例。

每章的最后都有一定数量的习题，以便加深对基本知识、基本理论、基本分析方法、基本设计方法和 Verilog HDL 建模技术的理解。习题数量较大，有些习题具有一定难度，为教学提供了一定的选择余地。

近年来，在计算机学院领导的支持下，数字逻辑课程的教育教学改革稳步进行、不断深入，不但为课程组创造了充分的研究、实验条件，而且在实验中心建立了先进的 EDA 实验室。系统结构系的诸位教授亲自组织、指导数字逻辑课程建设的各个环节，包括课程大纲的修订与完善、课程内容的优化、教材的编写以及与“数字逻辑实验”课程、后续课程的衔接等。

在本书的编写过程中，得到了课程组许向众、宗德忠、游周密、贾熹滨、孙丽君和魏坚华等各位教师的大力支持，他们的教学实践与经验为作者提供了极大的帮助。许向众高级工程师对全书进行了通阅，并提出了宝贵的意见和建议。在此一并表示衷心的感谢。

还要感谢实验中心韩德强主任对数字逻辑课程建设的全力支持与配合。

限于作者的水平与经验，书中的疏漏之处敬请广大读者批评指正。

彭建朝

2007 年 4 月于北京工业大学

目　　录

第1章 数制和码制

【本章内容】 本章主要介绍数字系统中数的基本表示方法。首先讨论现实世界中不同的进位计数制及其相互之间的转换；然后讨论带符号二进制数在计算机中的表示方法——原码、反码和补码。另外，本章还介绍了数字系统中几种常用的编码，包括十进制数字符号的常用编码、可靠性编码以及字符编码。

1.1 进位计数制

所谓进位计数制，就是按照某种进位方式实现计数的一种规则，简称进位制。

在现实世界中，人们最熟悉、普遍采用的进位制是十进制。在十进制中，采用了0、1、2、3、4、5、6、7、8、9十个数字符号和一个小数点符号“.”；遵循“逢十进一”的计数规则，这里的“十”是进位制的基数；数字符号在不同的位置代表着不同的数值，即不同的数位具有不同的权值，称为位权。例如十进制数845.13可以表示为如下形式：

$$(845.13)_{10}=8\times10^{2}+4\times10^{1}+5\times10^{0}+1\times10^{-1}+3\times10^{-2}$$

上式左侧表示形式称为位置表示法（位置计数法、并列表示法）；右侧表示形式称为按权展开式（多项式表示法），其中10^{2}、10^{1}、10^{0}、10^{-1}、10^{-2}即为数字符号所在位置的位权。

对于任意一个十进制数N，其位置表示法为

$$(N)_{10}=(K_{n-1}K_{n-2}\cdots K_{1}K_{0}.K_{-1}K_{-2}\cdots K_{-m})_{10} \tag{1.1}$$

对于任意一个十进制数N，其多项式表示法为

$$\begin{aligned}(N)_{10}=&K_{n-1}10^{n-1}+K_{n-2}10^{n-2}+\cdots+K_{1}10^{1}+K_{0}10^{0}+\\&K_{-1}10^{-1}+K_{-2}10^{-2}+\cdots+K_{-m}10^{-m}=\\&\sum_{i=-m}^{n-1}K_{i}10^{i}\end{aligned} \tag{1.2}$$

在式（1.1）和式（1.2）中，n表示整数部分的位数；m表示小数部分的位数；K_i表示十进制中的任意一个数字符号，$0\leqslant K_i\leqslant 9$；10表示基数；$10^i$表示第$i$位的位权。

由上面对十进制数的讨论，可以推断出任意R进制数的共有规律：

（1）具有R个数字符号：0、1、2、…、$R-1$；

（2）具有小数点符号“.”；

(3) 遵循“逢 R 进一”的计数规则，“R”——进位制的基数；

(4) 不同的数位具有不同的权值——位权 R^i。

因此，任意 R 进制的数可表示为

$$(N)_R=(K_{n-1}K_{n-2}\cdots K_1K_0.K_{-1}K_{-2}\cdots K_{-m})_R=\sum_{i=-m}^{n-1}K_iR^i \tag{1.3}$$

式 (1.3) 中，N 为所要表示的数；R 表示进位制的基数；n 表示整数部分的位数；m 表示小数部分的位数；K_i 表示 R 进制中的任意一个数字符号，$0\leqslant K_i\leqslant R-1$；$R^i$ 表示第 i 位的位权。

例如：二进制 ($R=2$)，只有 0 和 1 两个数字符号，逢二进一；八进制 ($R=8$)，有 8 个数字符号，即 0、1、2、3、4、5、6、7，逢八进一；而十六进制 ($R=16$) 中的数字符号有 0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F (注意：数字符号只能占一位，所以，十六进制中用 A～F 表示 10～15)，逢十六进一。

表 1.1 是几种常用进位制下数的对照表。

表 1.1　不同进位制下数的对照

十进制	二进制	八进制	十六进制	十进制	二进制	八进制	十六进制
0	0	0	0	9	1001	11	9
1	1	1	1	10	1010	12	A
2	10	2	2	11	1011	13	B
3	11	3	3	12	1100	14	C
4	100	4	4	13	1101	15	D
5	101	5	5	14	1110	16	E
6	110	6	6	15	1111	17	F
7	111	7	7	16	10000	20	10
8	1000	10	8				

下面是几种进位制下数的按权展开式：

$$(254.29)_{10}=2\times10^2+5\times10^1+4\times10^0+2\times10^{-1}+9\times10^{-2}$$

$$(1011.1101)_2=1\times2^3+0\times2^2+1\times2^1+1\times2^0+1\times2^{-1}+1\times2^{-2}+0\times2^{-3}+1\times2^{-4}$$

$$(107.6)_8=1\times8^2+0\times8^1+7\times8^0+6\times8^{-1}$$

$$(10A.B9)_{16}=1\times16^2+0\times16^1+A\times16^0+B\times16^{-1}+9\times16^{-2}$$

有时也用 D (Decimal，十进制)、B (Binary，二进制)、O (Octal，八进制)、H (Hexdecimal，十六进制) 作为下标或后缀来区分不同进位制的数。

例如：

$$(125)_{10}=(125)_D=125D$$

$$(10100)_2=(10100)_B=10100B$$

$$(307)_8=(307)_O=307O$$

$$(F0CA)_{16}=(F0CA)_H=F0CAH$$

在数字系统（机器世界）中，采用二进制来表示数并进行运算。这是因为二进制只有 0

和1两个数字符号，很容易用物理状态来表示；二进制的运算规则简单，便于进行算术运算；不仅可以节省设备，还便于用逻辑代数进行分析研究。

二进制的运算规则如下：

加法规则　0+0=0　　0+1=1　　1+0=1　　1+1=0（进位为1）

减法规则　0−0=0　　0−1=1（借位为1）　　1−0=1　　1−1=0

乘法规则　0×0=0　　0×1=0　　1×0=0　　1×1=1

除法规则　0÷1=0　　1÷1=1

当一个二进制数的位数很多时，书写和阅读不方便，容易出错。因此，常采用八进制或十六进制作为二进制的缩写形式。这就引出了不同进位制数之间的相互转换问题。

1.2　常用进位制之间的转换

将一个数从一种进位制表示形式转换成另一种进位制表示形式，称为数制转换。相互转换的原则是，转换前后两个数的整数部分和小数部分必定分别相等。下面介绍常用的十进制数、二进制数、八进制数和十六进制数之间的转换。

1.2.1　其他进制向十进制的转换

将二进制数、八进制数、十六进制数转换成十进制数的方法是：按权展开，对多项式进行算术求和，结果用十进制的位权表示法给出。

【例1.1】　$(1011.1101)_2=(\ ?\ )_{10}$

解　$(1011.1101)_2=1\times2^3+1\times2^1+1\times2^0+1\times2^{-1}+1\times2^{-2}+1\times2^{-4}=$

$(8+2+1+1/2+1/4+1/16)_{10}=$

$(11+0.5+0.25+0.062\ 5)_{10}=$

$(11.812\ 5)_{10}$

【例1.2】　$(10A.B9)_{16}=(\ ?\ )_{10}$

解　$(10A.B9)_{16}=1\times16^2+A\times16^0+B\times16^{-1}+9\times16^{-2}=$

$(256+10+11/16+9/256)_{10}\approx$

$(266+0.687\ 5+0.035\ 156)_{10}=$

$(266.722\ 656)_{10}$

【例1.3】　$(25.67)_8=(\ ?\ )_{10}$

解　$(25.67)_8=2\times8^1+5\times8^0+6\times8^{-1}+7\times8^{-2}=$

$(16+5+6/8+7/64)_{10}\approx$

$(21.859\ 4)_{10}$

1.2.2　十进制向其他进制的转换

按照若两数相等，则它们的整数部分和小数部分分别相等的转换思路，用除以基数取余

数的方法对整数部分进行转换；用乘以基数取整数的方法对小数部分进行转换。

整数部分的转换采用“除基取余”法。即用目标数制的基数（R）去除十进制数，第一次相除所得的余数是目的数的最低位（K_0），将所得的商再除以该基数，所得的余数为目的数的次低位（K_1），重复上述过程，直到商为“0”，所得的余数为目的数的最高位（K_{n-1}）。

小数部分的转换采用“乘基取整”法。即用该小数乘以目标数制的基数（R），第一次相乘结果的整数部分为目的数的最高位（K_{-1}），将其小数部分再乘以该基数，所得结果的整数部分为目的数的次高位（K_{-2}），重复上述过程，直到乘积的小数部分为“0”或满足要求的精度为止（即根据设备字长限制，取有限位的近似值），所得结果的整数部分为目的数的最低位（K_{-m}）。

下面以十进制数转换成二进制数为例进行说明。

【例 1.4】 $(28.44)_{10}=(\ ?\ )_2$，精确到小数点后 5 位。

解 对整数部分和小数部分分别转换。

整数部分：除以 2 取余

$$
\begin{aligned}
28/2&=14\cdots\cdots\text{余}0 \quad \text{低位 } K_0\\
14/2&=7\cdots\cdots\ 0\\
7/2&=3\cdots\cdots\ 1\\
3/2&=1\cdots\cdots\ 1\\
1/2&=0\cdots\cdots\ 1 \quad \text{高位 } K_4
\end{aligned}
$$

即 $$(28)_{10}=(11100)_2$$

小数部分：乘 2 取整

$$
\begin{aligned}
0.44\times2&=0.88\cdots\cdots\text{取整}0 \quad \text{高位 } K_{-1}\\
0.88\times2&=1.76\cdots\cdots\ 1\\
0.76\times2&=1.52\cdots\cdots\ 1\\
0.52\times2&=1.04\cdots\cdots\ 1\\
0.04\times2&=0.08\cdots\cdots\ 0 \quad \text{低位 } K_{-5}
\end{aligned}
$$

即 $$(0.44)_{10}\approx(0.01110)_2$$

答：$(28.44)_{10}\approx(11100.01110)_2$。

由例 1.4 可看出，整数部分总是能进行准确的转换，但对于小数部分不一定能准确转换，即存在一个转换精度问题。在进行小数转换时，究竟需要多少位才能保证原来的精度呢?

设 α 进制小数有 i 位，转换成 β 进制后，保证相同精度需要 j 位，此时应有

$$\alpha^{-i}=\beta^{-j}$$

等式两边同取对数，取正值，可得

$$i\lg\alpha=j\lg\beta$$

则

$$j=i\lg\alpha/\lg\beta$$

所以应取 j 为满足下列不等式的最小整数

$$j\geqslant i\lg\alpha/\lg\beta \tag{1.4}$$

例如，将$(0.4071)_{10}$转换成八进制数时，$\alpha=10$，$i=4$，$\beta=8$，带入式（1.4）得

$$j\geqslant 4\lg(10)/\lg(8)=4.428$$

所以，应取 $j=5$。

1.2.3 二进制与八进制之间的转换

1 位八进制的 8 个数字符号与 3 位二进制的 8 种不同组合存在对应关系：

八进制数	0	1	2	3	4	5	6	7
二进制数	000	001	010	011	100	101	110	111

即：3 位二进制数可表示一个八进制数字符号或一个八进制数字符号可用 3 位二进制数表示。因此，二进制与八进制之间的转换很方便。

将二进制数转换成八进制数的方法是分组对应转换。以小数点为界，将二进制数的整数部分从低位开始，小数部分从高位开始，每 3 位分成一组，最后一组不足 3 位时，分别在整数的最高位前面和小数的最低位后面补“0”。然后，将每组的 3 位二进制数转换成对应的八进制数字符号。

【例 1.5】 $(11010111.0100111)_2=(\quad ?\quad)_8$

解

二进制数	011	010	111	.	010	011	100
八进制数	3	2	7	.	2	3	4

所以，$(11010111.0100111)_2=(\ 327.234\ )_8$

将八进制数转换成二进制数的方法是按位对应转换。将八进制数中的每一位数字符号表示成对应的 3 位二进制数，去掉首尾的“0”。

【例 1.6】 $(\ 125.654\ )_8=(\ ?\ )_2$

解

八进制数	1	2	5	.	6	5	4
	↓	↓	↓		↓	↓	↓
二进制数	001	010	101	.	110	101	100

所以，$(125.654)_8=(\ 1010101.1101011)_2$

1.2.4 二进制与十六进制之间的转换

1 位十六进制的 16 个数字符号与 4 位二进制的 16 种不同组合存在对应关系：

十六进制数	0	1	2	…	A	B	C	D	E	F
二进制数	0000	0001	0010	…	1010	1011	1100	1101	1110	1111

即：4 位二进制数可表示一个十六进制数字符号或一个十六进制数字符号可用 4 位二进制数表示。因此，二进制与十六进制之间的转换也很方便。

将二进制数转换成十六进制数的方法是分组对应转换。以小数点为界，将二进制数的整数部分从低位开始，小数部分从高位开始，每 4 位分成一组，最后一组不足 4 位时，分别在整数的最高位前面和小数的最低位后面补“0”。然后，将每组的 4 位二进制数转换成对应的十六进制数字符号。

【例 1.7】 $(11010111.0100111)_B=(\quad ?\quad)_H$

解

二进制数	1101	0111	.	0100	1110
十六进制数	D	7	.	4	E

所以，$(11010111.0100111)_B=(\ D7.4E\ )_H$

将十六进制数转换成二进制数的方法是按位对应转换。将十六进制数中的每一位数字符号表示成对应的4位二进制数，去掉首尾的“0”。

【例1.8】 $(3A5.6)_H=(\ ?\ )_B$

解 十六进制数　3　A　5　.　6

二进制数　0011　1010　0101　.　0110

所以，$(3A5.6)_H=(\ 1110100101.011)_B$

八进制与十六进制之间的转换，通常以十进制或二进制作为中间的过渡进制，进行转换。例如：

$$(A0.D)_{16}=(10100000.1101)_2=(240.64)_8$$

1.3 带符号二进制数的代码表示

前面讨论现实世界中常用进制的数及其转换时没有涉及符号，在机器世界中，是采用一组两种不同的状态（状态0和状态1）来表示二进制数的，且需考虑其符号。那么，一个带符号的二进制数在数字系统中是如何表示的呢？

1.3.1 真值与机器数

在现实世界中，通常用“+”、“−”符号表示数的正、负。带“+”、“−”符号的二进制数，称为真值。例如，$x_1=+0.101101$，$x_2=-100110$。可以看出，真值（带符号的二进制数）由符号部分和数值部分组成，真值X一般可表示成

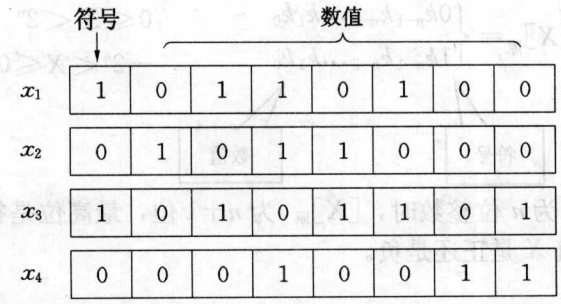
$$X=\pm k_{n-1}k_{n-2}\cdots k_1k_0.k_{-1}k_{-2}\cdots k_{-m}$$

在数字系统中，存储信息的最小单位是“位”（bit），其“0”状态和“1”状态正好可以表示二进制的两个数字符号0和1；8位（8 bit）构成一个字节（Byte），可以用来存储二进制数值（单字节数）；若将真值的“+”、“−”符号也用“0”、“1”状态来编码表示，则可以将带符号的二进制数存放到机器中，这时，所形成的代码称做机器数或机器码。显然，机器数也是由符号和数值两部分构成的，在符号部分用“0”表示“+”，用“1”表示“−”。但由于设备的原因，机器数有字长的限制。在下面的示例中，均以一个字节作为字长（1位符号位，7位数值位）进行讨论。

设带符号的二进制数分别为

$$x_1=-0.01101=-0.0110100$$
$$x_2=+0.1011=+0.1011000$$
$$x_3=-101101=-0101101$$
$$x_4=+10011=+0010011$$

它们在数字系统中的表示如下：



数的小数点是隐含表示的，整数时，小数点默认在数值位的最后面；小数时，小数点默认在数值位的最前面、符号位的后面。

根据数值部分在机器内存放形式的不同，机器数有多种表示形式，主要有原码、反码和补码。

1.3.2 原码

原码（True Form）又称符号—数值表示法。其符号位用状态“0”表示“+”，用状态“1”表示“−”，而数值部分与真值的数值部分相同。

当真值 X 为小数（$\pm 0.k_{-1}k_{-2}\cdots k_{-m}$）时，它的原码表示为

$$[X]_{原}=\begin{cases}0.k_{-1}k_{-2}\cdots k_{-m} & 0\leqslant X<1\\ 1.k_{-1}k_{-2}\cdots k_{-m} & -1<X\leqslant 0\end{cases}$$

符号　书写　数值

这表明，当真值 X 为 m 位小数时，$[X]_{原}$ 为 $m+1$ 位，最高位是符号位，根据符号位是 0 还是 1，便可确定真值 X 是正还是负。这里的小数点在机器中是不存在的（只是默认位置），只是在书写时为了区别于整数。

例如：

$$x_1=+0.1101=+0.1101000\quad [x_1]_{原}=0.1101000$$
$$x_2=-0.0011=-0.0011000\quad [x_2]_{原}=1.0011000$$

下面给出真值 X 为小数时，其原码的数学描述形式。

设真值 $X=\pm 0.k_{-1}k_{-2}\cdots k_{-m}\quad (-1<X<1)$，则

$$[X]_{原}=\begin{cases}X & 0\leqslant X<1\\ 1-X & -1<X\leqslant 0\end{cases}\tag{1.5}$$

根据式（1.5）也可计算出真值 X 为小数的原码，例如：

$x_1=+0.1101000\quad [x_1]_{原}=0.1101000$

$x_2=-0.0011000\quad [x_2]_{原}=1-(-0.0011000)=$

$1+0.0011000=$

1.0011000

当真值 X 为整数（$\pm k_{n-1}k_{n-2}\cdots k_0$）时，它的原码表示为