高级可编程逻辑设计语言

abel

·陆凌雷 译

北京四通办公设备有限公司

abel

高级可编程逻辑设计语言

陆凌霄 译

王北文 校

北京四通办公设备有限公司

BEIJING STONE OFFICE EQUIPMENT TECHNOLOGY CO., LTD. (SOTEC)

前 言

可编程逻辑器件(*Programmble logic devices*,简称 *PLD*)是 *80* 年代蓬勃发展起来的专用集成电路(*ASIC*)的一个重要分支。*PLD* 器件的应用为硬件电路设计带来了革命性的变化。它使逻辑电路的设计和修改更加灵活,实现起来更加方便。用户可以用很少的器件完成很强的功能,简化了硬件电路,降低了成本,提高了可靠性和保密性。特别是 *PLD* 器件的全部编程工作可由用户自己完成,不必依赖于器件生产厂,这样就缩短了开发周期,加强了产品的市场竞争能力,所以在国内外的计算机硬件、工业控制、智能仪表、数字电路系统、家用电器中都得到广泛应用。

随着 *PLD* 器件的应用,出现了一批用于 *PLD* 编程的各种高级逻辑设计软件,借助这种软件,用户可以使用自然语言的逻辑表达式如布尔方程式、真值表和状态图等去解决组合变换、时序变换、状态变换、自锁、互锁控制、反馈、三态门控制、寄存器输出控制等一系列问题。这种软件不但可以对用户给出的逻辑设计进行语法检查、逻辑化简、自动生成用于烧制 *PLD* 器件的熔丝点阵,而且还具有模拟仿真功能,即将用户的设计要求和所选择的具体器件的性能结合在一起,以检查用户的目的是否能切实地实现。无疑,这种逻辑设计软件为促进 *PLD* 器件的应用发挥了巨大的作用。*DATA I/O* 的高级逻辑设计语言 *ABEL* 就是当前国际上流行的一种 *PLD* 辅助设计软件。

为了使广大电路设计人员学习和掌握 *PLD* 器件编程技术,提高逻辑电路的设计和技术水平,使 *PLD* 器件得到更广泛的应用,我们将《高级可编程逻辑设计语言 *ABEL*》一书推荐给大家。

本书由三部分组成

《 $ABEL^{TM}$ 语言参考手册 》介绍了 *ABEL* 语言的语法规则。

《 $ABEL^{TM}$ 用户指南 》结合具体实例介绍了如何用 *ABEL* 语言处理软件把逻辑描述转换成编程器装入文件,详尽描述了所有处理程序的任选项,解释了输出文件,描述了向编程器传送编程器装入文件的全过程。

《 $ABEL^{TM}$ 应用指南 》为使用户使用好 *ABEL* 提供了一些资料和实例,给出了一些典型的可编程逻辑设计的设计说明书,设计方法和完整的源文件。

我们建议读者先阅读《 $ABEL^{TM}$ 用户指南 》和《 $ABEL^{TM}$ 应用指南 》,在对 *ABEL* 应用方法有了一个初步了解后,再仔细学习语言参考手册,学习用正确的 *ABEL* 语言描述自己的逻辑设计。

本书由陆凌雷译,王北文校,由于时间仓促,我们的水平有限,错误在所难免,敬请读者原谅。

希望本书的出版为推广 *PLD* 器件的应用起到积极的作用。

译者

1989.12

总 目 录

ABEL™ 语言参考手册

ABEL™ 用 户 指 南

ABEL™ 应 用 指 南

一九九一年 九月 . 修 订 .

abel™ 语言参考手册

abel™ Language Reference

目 录

第一节 引 言

该语言参考手册详细地描述了 ABEL™ 语言的元素与结构。它既可作为对 ABEL™ 语言的完整的规格说明书以供不熟悉 ABEL™ 语言的设计人员使用，亦可作为一部简明的参考材料提供给比较有经验的设计人员。

如表 1 — 1 所示，这本手册主要由下述四个主要章节组成：

表 1— 1 该语言参考手册的章节及内容

章节	标 题	说 明
1.	引言	对本手册的介绍。
2.	元素	描述了ABEL™ 语言的基本元素，譬如说：关键字，常量，表达式，集合和等式等。
3.	结构	解释应如何把不同元素组织成不同的结构，以形成一个 ABEL™ 的逻辑描述。
4.	命令	讨论影响源文件处理的命令。

1.1 其它ABEL™ 参考手册

这本语言参考手册是三本描述 ABEL™ 语言的手册中的一本。其它两本参考手册是：

ABEL™ 用户指南

该参考手册阐述了如何使用 ABEL™ 语言处理软件来把逻辑描述转换成编程设计文件。其中描述了 SETUP 过程，给出了一个使用语言处理软件的示例，详尽地描述了所有的处理程序的选择项，解释了输出文件，并描述了装入编程器装入文件的过程。

ABEL™ 应用指南

这本参考手册为帮助设计者使用 ABEL™ 提供了一些资料和示例，并为典型的可编程序逻辑设计提供了规格说明书、设计方法及完整的源文件，并提供了一些高级的设计，也给出了关于使用 ABEL™ 的建议与告诫。

1.2 符号约定

在本手册中用于定义及语法描述中的符号约定如表 1— 2 所示。

用在文本中的斜体的一个例子。

在表 1 — 2 中，字 ellipsis（省略号）代表在一行中的三个圆点。

用在文本中的引号的例子：

若关键字按"Mod ule"键入，那么，它将被解释为二个标识符，Mod 及 ule。

用于语法描述中的大写字母、小写字母、方括号及省略号的例子。

表 1— 2 符号约定

符 号	用 法
斜体字	表示在例子、插图、清单及表格中按名字引用的项.
引号(")	当斜体的参考名引用中含有空格时, 用来括起斜体项.
大写字母	在语法描述和图表中, 表明大写的单词或字母必须要全部键入. 该项既可用大写字母, 亦可用小写字母键入, 表明这是一个关键字.
小写字母	在语法描述和图表中, 表示要用户提供一个变量或名字来取代其中的小写单词或小写字母. 用户提供的变量既可用大写字母键入, 亦可用小写字母键入.
方括号[ ]	用于包围按需要既可以提供, 也可省去的候选项.
省略号(…)	表示前面的项可按需要而重复出现.
所有其它的标点符号	撇号, 感叹号, 括号, 引号, 逗号, 必须严格地按所示的键入.

MODULE modname[(arg[, arg] …)]

这表明: 必须键入关键字 MODULE, 既可以用大写亦可用小写字母键入, 必须键入一个模块名来取代 modname. 可提供 0 个或多个参数来取代 arg. 一旦提供了参数, 则必须用括号将他们括起来. 下面便是一个合法的用户项:

module ADDER(Carryin, add1,ADD2)

借助于语法描述给出了 ABEL™ 的语法定义. 一个语法描述包含了一个完整而又合法的关于语言结构的语法定义及关于组成结构的各个部分的定义. 一个语法描述的第一行(粗体表示的)定义了上面所列出的关于符号约定的语法. 第一行以后的行定义了语法的元素. 语法描述包含在方框中, 图 1— 1 表示了一个语法描述的实例.

图 1— 1 语法描述实例

device__id	[, device__id] … DEVICE real__device
device__id	在模块中用于表示某一器件的一个标识符
real__device	用于描述device__id 所表示的真实器件的工业部件号的字符串.

第二节 语 言 元 素

本节中描述了 ABEL™ 设计语言的各种元素. 这些语言元素可根据在第 3 节中描述的结构组合而生成 ABEL™ 逻辑描述. 在后面我们将描述每一个元素. 从简入手, 逐步加深.

2.1 基本语法

在 ABEL™ 源文件中每行必须满足下述语法规则和限制:

1. 一行最多可达 131 个字符长.

2. 可用换行符(十六进制 0A),竖表符(十六进制 0B),或用换表符(十六进制 0C)来结束一行。在一行中的回车被忽略,因此,适合于通常的行结束序列,譬如说,回车/换行。在多数计算机中,只要单纯地按下回车或进入键便结束输入行。

3. 关键字、标识符及数之间至少要用一个空格符来隔开。对该规则的例外情况是在标识符表中用逗号隔开,在表达式中标识符或数用操作符隔开,在有些地方用括号来隔开。

4. 空格不得包含在关键字、数、操作符或标识符之中,空格符可出现在字符串,注释、块及实际参数之中。

例如,若关键字 MODULE 按"MOD ULE"键入,那么,它将被解释为二个标识符 MOD 和 ULE。类似地,如果你想键入"102 05"来代表 10205,那么,它将被解释成二个数 102 和 05。

5. 关键字(作为语言的一个部分而定义的,且有特定用途的单词)既可用大写方式键入,亦可用小写方式键入,其效果完全一样。

6. 标识符(用户所提供的名字和标号)既可用大写方式或小写方法键入,亦可用混合方式键入,但与大小写方式有关:标识符 output 按全部小写方式键入,则与仅一个大写"O"的标识符 Output 不同。

2.2 合法的 ASCII 字符

所有在普通键盘上的大写字母、小写字母及其它字符均为合法的。表 2—1 表示了所有合法的字符。

表 2—1 合法 ASCII 字符

a – z	lower – case alphabet
A – Z	upper – case alphabet
0 – 9	digits
space	
tab	
! @ # $ % ^ & * ( ) –	
_ = + [{] } ; : ' "	
` ~ \ | , < > . / ?	

2.3 标识符

标识符是用来标识器件、器件插头或结点、集合、输入或输出信号、常量、宏、及哑参数等的名字。将在第二节的后面来定义所有这些项,除了各标识符所描述的东西不同外,对所有标识符的规则和限制都是相同的。

限制标识符的规则是:

1. 标识符必须以一个字母或下划线开始。
2. 标识符最多可长达31个字符，任何长于31个字符者均认为出错并由语言处理软件打出出错标志。
3. 除了第一个字符外(见规则 1)，标识符可包含大写字母、小写字母，数字及下划线符号。
4. 空格不得出现于标识符中，为了分开二个单词可使用下划线。
5. 标识符与大小写有关：大写字母和小写字母是不同的。

某些合法的标识符有：

HELLO
hello
__K5input
P__h
This__is__a__very__long__identifier

请注意使用下划线符来分开单词，亦请注意小写字母与大写字母之不一样，因此 hello 是一个不同于 HELLO 的标识符，为了使你的源文件易读，可使用不同的大小写。

一些非法标识符有：

7__	不以字母或下划线符开头。
$4	不以字母或下划线符开头。
HEL. LO	含有圆点。
b6 kj	含有空格。

请注意，上述非法标识符中的最后一个，被语言处理软件视为二个标识符 b6 和 kj。

2.3.1 保留标识符

关键字是保留标识符，是ABEL™ 设计语言的一个部分，这意味着不能用它们作为器件、插头、结点、常量、集合、宏或信号的名字。一旦使用了关键字，则它只代表该关键字之功能，一旦把关键字用于不恰当的上下文中，则语言处理软件将会给出出错标志。表 2— 2 给出了这些保留标识符。

表 2—2 保留标识符

CASE	FUSES	PIN
DEVICE	GOTO	STATE
ELSE	IF	STATE_DIAGRAM
ENABLE	IN	TEST_VECTORS
END	ISTYPE	THEN
ENDCASE	LIBRARY	TITLE
ENDWITH	MACRO	TRUTH_TABLE
EQUATIONS	MODULE	WITH
FLAG	NODE	

2.3.2 选择标识符

正确地选择标识符可使源文件易于阅读并便于理解，尤其是在多于一个人工共同开发同一项目或一个设计人员要接替其它人以前的工作时，更显得重要。

下面给出一些有助于使逻辑描述具有自解释性的建议。因此可减少额外的文档说明。

- 选择与其功能相匹配的标识符。例如，对于用作加法器的进位的管脚可用名字carry_in来命名，对于一个简单的或门，其二个输入脚可用IN1和IN2来标识，其输出可标识为OR。
- 避免相似标识符的数目太大，例如，不要把16位加法器的输出命名为

  ADDER_OUTPUT_BIT_1, ADDER_OUTPUT_BIT_2, 等等。

  这种名字的编组将使得源文件难读。
- 在你的标识符中使用下划线符来分开其中的单词。

  THIS_IS_AN_DENTIFIER 远要比 THISISANIENTIFIER 容易读得多。
- 混合使用大小写字母，也可使你的源文件易于阅读。例如: CarryIn。

2.4 字符串

字符串是用单引号(apostrophes)包围起来的ASCII 字符序列。字符串可用于TITLE、MODULE及FLAG语句以及管脚，结点及属性说明中，字符串允许有空格。

合法字符串有:

'Hello'

'Text with a space in front'

' '

'The preceding line is an empty string'

'Punctuation ? is even allowed!!'

通过在前面加上一个斜杠符"\"，单引号亦可出现于字符串之中。

'It\'s easy to use ABEL'就是字符串" It ' s easy to use ABEL "。

斜杠符通过相应地使用二个而可加入字符串中。

'He\\ she can use backslashes in a string' 便是字符串:

"He \\ she can use backslashes in a string"

注意: 后单引号(')亦可作为字符串的分隔符，并可与前单引号(')交换地使用。

2.5 注释

注释是使源文件易于理解的又一方法。注释解释了源代码本身并不能立即明了的意义，注释不影响代码的意义。

注释以双引号'"'开头，以另一个双引号或行结束为止(无论先遇到哪个都行)，注释文字跟在开始的双引号之后。

注释不得嵌入关键字中。

合法的注释(粗体部分):

MODULE Basic_ logic ; "**gives the module a name**
TITLE 'ABEL design example: simple gates'; "**title**
"**declaration section**
IC4 device 'P10L8'; "**dedare IC4 to be a P10L8**
IC5 "**decoder PAL**" device 'P10H8';
注意: 单引号内的信息不是解释而是语句的一个部分。

2.6 数

ABEL™ 中的所有数值运算的精度为 32 位。因此，合法数值应在 0 到 $2^{32}-1$ 之间，数可按五种形式中的任何一种来描述，四种最常用的形式是根据不同的基数来描述数，第五种形式是使用字母来描述数值。

当选择了 4 个基中的一个而不是缺省基来表示一个数时，所使用的基用在数字前的一个符号来说明，表 2— 3 给出了 ABEL™ 支持的 4 个基及对应的符号，基符号既可用大写亦可用小写键入。

当说明了一个数且前面没有基符号时，该数被认为是缺省基数系。一般的缺省基是基为 10。因此，除非通过前面用符号指出了所使用的基，否则该数用 10 进制形式描述。

对于特定的应用，缺省基可被改变，详尽信息请参见第 4 节中的"@ RADIX"。

表 2— 3 不同基的数的表示

Base Name	Base	Symbol
binary	2	^b
octal	8	^o
decimal	10	^d
hexadecimal	16	^h

下面给出一些合法的数字说明的例子，缺省基为 10。

说明	十进制值
75	75
^h 75	117
^b 101	5
^ o 17	15
^h0 f	15

长音符号"^"必须作为一个字符从键盘上键入。当在某些其它的通用软件中，它并不表示一个控制键序列。

数也可以用字母字符来说明，在这种情况，用字母的ASCII 数字码作为数值，例如，字符"a"为十进制的 97，在 ASCII 编码中为十六进制的 61，若"a"被说明为一个数，则可用作十进制数值 97。

字母字符序列先被转换成其二进制ASCII 值，然后再把这些值连结一起以构成数(通常较

大)。

下面给出用字符说明的数:

说明	十六进制值	十进制值
'a'	^h 61	97
'b'	^h 62	98
'abc'	^h 616263	6382203

2.7 特殊常量值

在ABEL™逻辑描述中可使用值不变的常量。常量值可用于赋值语句,真值表和测试向量中,有时可把常量赋予标识符,然后,该标识符将在整个模块中代表该常量值。(参见3.5节的"说明语句"及3.2节的"模块语句")。

这些常数值既可是数值的,亦可是非数值的特殊常数值。这些特殊值列于表2—4中。

表2—4 特殊常数值

值	描述
.C.	时钟输入(低—高—低变换)
.F.	浮点输入或输出信号
.K.	时钟输入(高—低—高变换)
.P.	寄存器预装入
.SVn.	n=2到9,驱动输入到过压2到9
.X.	无关项
.Z.	测试高阻抗的输入或输出

一旦使用某个特定的常量,则该常量必须按表2—4所示的用圆点括起来键入。圆点表示正在使用某个特定的常量。假如没有圆点,则c.便是以C为名的一个标识符。特殊常量既可用大写字母亦可用小写字母键入。

2.8 运算符、表达式及等式

象常量及信号名字一类的项可引入表达式中。表达式对其所包含的各项进行组合,比较或执行等运算以产生出一个结果。所要完成的运算由表达式内的运算符指定(例如,加法和逻辑AND等)。

ABEL™运算符分为四个基本类型:逻辑运算、算术运算、关系运算及赋值运算。下面将分别讨论它们,然后再描述如何把它们组合成表达式,并总括所有运算符及相应之规则,最后,解释等式如何使用表达式。

2.8.1 逻辑运算符

逻辑运算符用于布尔表达式中。ABEL™中包含了在多数逻辑设计中使用的标准逻辑运算符;表2—5列出了所有这些运算符。

2— 5 逻辑运算符

运算符	示　例	描述
!	!A	非：1 的补
&	A&B	AND
#	A#B	OR
$	A $ B	XOR: 异或
!$	A! $ B	XNOR: 异或非

涉及多于1位的操作数的逻辑运算按位进行运算，因此，2#4 等于 6，对于替换运算符，请参见本语言参考手册中的 @　ALTERNATE 指令一节。

2.8.2 算术运算符

算术运算符定义了表达式中各项间的算术关系。移位运算符也属此类，因为每左移一位等价于乘2，而右移一位等价于除 2 。表 2— 6 列出了这些算术运算符。

表 2— 6　算术运算符

运算符	示 例	描　述
−	−A	2 的补
−	A−B	减法
+	A+B	加法
*	A * B	乘法
/	A / B	无符号整除
%	A % B	求模: / 的余数
<<	A<<B	将A 左移 B 位
>>	A>>B	将A 右移 B 位

注意减号根据其使用的不同有不同的意义，当用于一个操作数时，意味着形成该操作数对2的补，当减号出现在二个操作数的中间时，则将第二个操作数对 2 的补加上第一个操作数。

除法为无符号整除: 除的结果为正整数，可使用求模运算符"%"来得到除法的余数。

移位运算符完成无符号逻辑移位，右移时从左部加 0，反之则从右部加 0。

2.8.3 关系运算符

关系运算符用于比较表达式中的两项。用关系运算符构成的表达式的结果为布尔值真或假。 表 2— 7 中列出了这些关系运算符。

表 2— 7 关系运算符

运算符	示例	描述
= =	A = = B	等于
! =	A! = B	不等于
<	A < B	小于
< =	A < = B	小于或等于
>	A > B	大于
> =	A > = B	大于或等于

所有关系运算都是无符号的，例如表达式 −1 > 4 为真。因为 1 的补为 1111，在无符号二进数中为 15，而 15 又大于 4。为说明这个例子，我们假定采用的是 4 位表示。在实际应用中，−1 作为 1 的补为 32 位全部置 1。

下面是表达式中的关系运算符的一些示例：

表达式	值
2 = = 3	false
2 ! = 3	true
3 < 5	true
−1 > 2	true

逻辑值 TRUE 和 FALSE 在内部是用数表示的。

逻辑 t r u e 值为 −1，即以补的形式：32 位全置 1。逻辑 false 值为 0，按 2 的补的形式为 32 位全置 0。这意味着：产生 true 和 false 值的表达式(关系表达式)可被用于一个数或一个数值表达式可以使用的任何地方，并根据逻辑结果的值用 −1 或 0 来取代它们。

例如：

A = D $ (B = = C)

表示：若 B 等于 C 则 A 等于 D 的补。

若 B 不等于 C 则 A 等于 D。

2.8.4 赋值运算符

赋值运算符是用于等式而不是表达式中的一种特殊的运算符。等式将表达式的值赋予输出信号。关于等式的更全面的论述请参见第 2.8.6 节。

有两种赋值运算符：不带时钟的运算符和带有时钟的运算符。不带时钟的或立即赋值一旦求出了等式的值后无须任何延迟便执行赋值。带有时钟的赋值则要等待与输出相联系的时钟的下一个时钟脉冲的到来。表 2— 8 列出了这两个赋值运算符。

表 2— 8 赋值运算符

运算符	描 述
=	不带时钟的赋值(组合输出)
:=	带时钟的赋值(寄存器输出)

2.8.5 表达式

表达式是标识符和运算符的组合,计算时产生一个结果. 任何逻辑运算符、算术运算符或关系运算符均可用在表达式中.

表达式根据它所涉及的特定运算符来进行计算. 一些运算符优先于其它运算符,它们的运算将首先进行. 已给每个运算符一个优先权以决定其计算顺序. 优先权 1 为最高优先权,而优先权 4 为最低.

表 2—9 综述了逻辑运算符,算术运算符和关系运算符,并根据其优先权按组给出.

当具有同等优先权的运算符出现在表达式中时,则按在表达式中从左到右的方式执行. 括号可以象在通常数学中的方式用来改变计算的次序,并且先执行最内层括号中的运算.

下面将给出一些合法表达式的例子,请注意运算的次序及括号的使用对计算结果的影响.

表达式	结果	注 释
2*3/2	3	同优先权的运算符
2 * 3 / 2	3	空格无关
2*(3/2)	2	
2+3*2	14	
2#4 $ 2	4	
2#(4 $ 2)	6	
2==^HA	0	false
14==^HE	−1	true