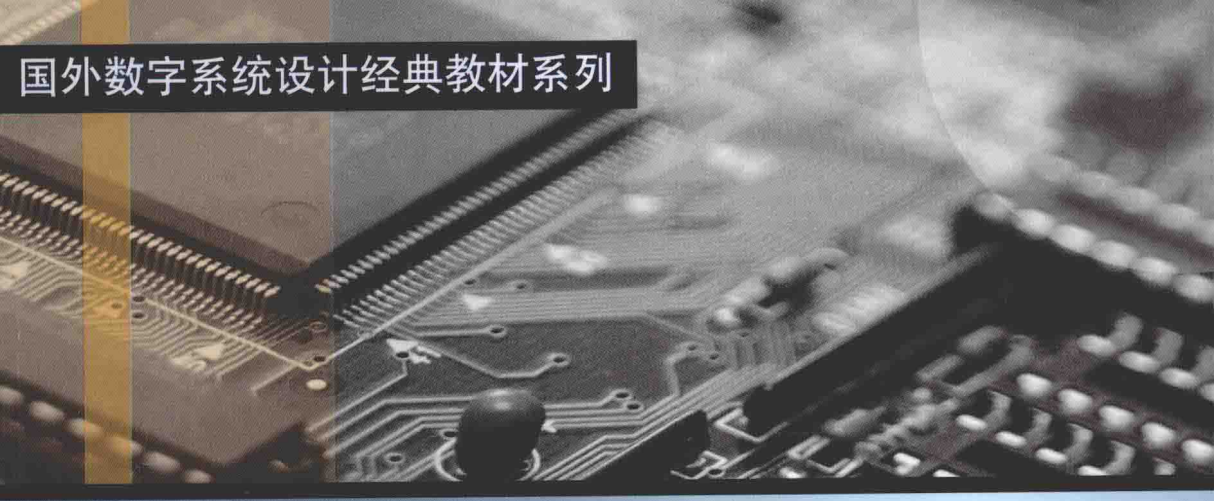

国外数字系统设计经典教材系列

VHDL for Logic Synthesis, Third Edition

[美] Andrew Rushton 著

用于逻辑综合的VHDL

（第3版）

刘雷波 陈英杰 译

夏宇闻 审校

北京航空航天大学出版社
BEIHANG UNIVERSITY PRESS

国外数字系统设计经典教材系列

用于逻辑综合的 VHDL

(第3版)

VHDL for Logic Synthesis, Third Edition

[美]Andrew Rushton 著
刘雷波　陈英杰　译
夏宇闻　审校

北京航空航天大学出版社

内 容 简 介

本书侧重于介绍面向逻辑综合的 VHDL 程序的编写方法，全面介绍了可综合的 VHDL 语法条款。但是，考虑到测试工作的重要性，本书也介绍了一部分最为实用的与编写测试平台有关的 VHDL 语法。

本书的读者对象是数字系统设计工程师和正在学习逻辑综合技术的硕士研究生。

图书在版编目(CIP)数据

用于逻辑综合的 VHDL：第 3 版 /（美）拉什顿（Rushton，A.）著；刘雷波，陈英杰译. -- 北京：北京航空航天大学出版社，2014.1

ISBN 978-7-5124-1366-5

Ⅰ. ①用… Ⅱ. ①拉… ②刘… ③陈… Ⅲ. ①VHDL 语言—程序设计 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字(2014)第 009850 号

用于逻辑综合的 VHDL(第 3 版)

VHDL for Logic Synthesis, Third Edition

[美]Andrew Rushton 著

刘雷波　陈英杰　译

夏宇闻　审校

责任编辑　卫晓娜

*

北京航空航天大学出版社出版发行

北京市海淀区学院路 37 号(邮编 100191)　http://www.buaapress.com.cn

发行部电话：(010)82317024　传真：(010)82328026

读者信箱：emsbook@gmail.com　邮购电话：(010)82316936

涿州市新华印刷有限公司印装　各地书店经销

*

开本：710 mm×1 000 mm　1/16　印张：29.25　字数：623 千字

2014 年 1 月第 1 版　2014 年 1 月第 1 次印刷　印数：3 000 册

ISBN 978-7-5124-1366-5　定价：89.00 元

译者序

硬件工程师在用 VHDL 语言进行电路设计的时候，都非常关心自己所编写的程序是否是可综合的，即是否能够被 EDA 工具自动综合成所希望的逻辑电路。市面上的其他针对 VHDL 语言的专业书籍通常将该硬件编程语言的所有细节都介绍得非常详细。这让一部分没有经验的硬件工程师很难判断出哪些内容对设计出一个优秀的逻辑电路有直接的帮助，哪些语法要素跟逻辑综合相关，哪些无关，从而变得无所适从。本书侧重于介绍面向逻辑综合的 VHDL 程序的编写方法，全面介绍了可综合的 VHDL 语法条款。但是，考虑到测试工作的重要性，本书也介绍了一部分最为实用的与编写测试平台有关的 VHDL 语法。这部分内容通常是不可综合的。

本书的翻译过程中，我们根据自己的硬件设计经验，在不改变作者本意的前提下，用中文尽量把原书中文字的具体含义表达清楚。但是，考虑到译者水平有限，难免会出现不恰当，甚至错误的方，希望读者谅解并指正。

参与本书的翻译工作的，除本人以外，还有清华大学的在读硕士研究生任彧、清华大学的硬件工程师陈英杰，以及北京航空航天大学的夏宇闻教授。非常感谢以上各位对本书翻译工作的大力支持。最后，我还非常感谢我的爱人在本书翻译出版过程中给我的支持和鼓励。

刘雷波
清华大学
2013 年 10 月 26 日

序　言

当初我学习 VHDL 时，找不到有关逻辑综合的书籍，于是萌生了编写本书的想法。另外，还发现大多数 VHDL 书籍都有一个共同的弱点，即它们以一种无差别的方式来描述语法的所有细节，并让读者自己分辨哪些语法与逻辑综合有关，可用于逻辑综合。通过这种途径来判断可综合子集是极其困难的。

本书从硬件设计师的视角，全面介绍了逻辑综合工作者必须知道的 VHDL 语法要点。用硬件专业术语，对 VHDL 每款语法做出相应的解释，并画出 VHDL 语句的硬件对照图。由于本书只介绍可综合的语法条款，所以不要将可综合与不可综合的语法条款混为一谈。

本书只介绍可综合语法，但测试平台这一章例外。硬件工程师通常只使用可综合的语法来生成相应的逻辑，但他们也不得不编写测试平台，以验证自己完成的设计是正确的，因为测试平台并非一定要能被综合成电路，所以在设计过程中，整个语法体系都变得有用（但未必每条语法都有用）。所以，在测试平台那一章还介绍了与编写测试平台有关的和有用的那部分（不可综合的）语法。

编写本书是十分有必要的，原因是 VHDL 是一个非常庞大和笨拙的语言体系。这种语言在初创时曾遭受到设计委员会的百般挑剔和磨难，从而造成学习难度大的后遗症，而且语言体系中存在许多无用的语法条款。根据我自己的经验：VHDL 极难实现其初创时设定的目标。我不是 VHDL 的拥护者，但我认识到它仍然可能是用于逻辑综合的最好的硬件描述语言。我在学习 VHDL 语言以及编写可综合代码过程中获得了不少知识和经验，希望能与读者分享，以避免许多易犯的错误。

我对 VHDL 持有这样的观点，是因为我的职业生涯始于电子工程师，专门从事数字系统设计多年，并在 1983 年和 1987 年分别获得英国南安普顿（Southampton）大学电子系的理学学士学位和博士学位。后来进入软件工程行业，从事电子设计自动化行业内的软件开发，但这项工作需要用到硬件背景。自从 1988 年以来，我一直在研究和学习 VHDL，并使用 VHDL 从事电子设计自动化软件设计的专业工作。

刚参加工作时,我的任务是逻辑综合系统的改进,起先是为 Plessey Research Roke Manor 公司工作,它现在是西门子公司英国分公司的一个部门。接着,在 1992 年,当时的项目经理是 Jim Douglas,他买下我们已成功开发的综合技术的控股权,组建一家新公司,成为公司的 CEO,这次收购得到了 MTI 合作伙伴的风险投资支持。从此, TransEDA 有限公司诞生了。他带走了该项目的主要工程师,因此我成为 TransEDA 公司的创始成员之一,是新公司的研发经理,继续从事逻辑综合项目的开发。我们的目标是将公司内部的逻辑综合工具发展成商业标准,并以 TransGate 品牌出售。我首批任务之一是帮助开发 VHDL 语言的前端工具,来替代当时现成的专有语言的前端工具。对当时取得的成果非常自豪,TransGate 对 VHDL 语言的支持非常全面,可与市场上最好的逻辑综合工具竞争,性能远优于大多数工具。

TransGate 第一次发布时,期望工程师们能很容易地掌握 VHDL 的使用,所以把研发重点放在综合算法的纯技巧方面。然而,反馈回来的信息表明,用户在应用 VHDL 语言编写可综合的逻辑代码时遇到了许多问题,学习进展十分缓慢,这是由于工程师们需要接受一个全新的硬件设计规范造成的。意识到这个问题后,我于 1992 年开办了一个新的培训课程,以公开课或者现场讲课的方式提供。该课程的名称为'用于硬件设计的 VHDL'。这个课程的内容是根据我所了解的综合器是如何解释 VHDL 代码的知识编写的,其中有参与解决客户实际问题的经验,这是公司客户支持计划的一部分。本书的第一版源自那个培训课程,由 McGraw－Hill 于 1995 年出版。许多段落和一些实例直接取自那个课程。然而,一本书包括的内容可以远远超过 3 天培训课程所能涵盖的范围,所以本书涵盖了更多的资料,远比以前培训课程提供的内容详实得多。

在编写第 1 版时,国际标准化组织正努力定义一套标准化的算术程序包和用于逻辑综合的 VHDL 常用子集和详解。当时该项工作的标准化尚未完成,尽管如此,在有关 VHDL 逻辑综合的某些方面已有了广泛的共识,这些共识为编写本书提供了有用的资料。

重返 TransEDA 公司,我们发现逻辑综合市场空缺不仅早已被占领,而且是被信誉卓著的公司全面占领,而我们在销售自己的综合工具中却没有取得什么进展。幸亏我们另辟蹊径,开发了代码覆盖工具,并在这个市场中为自己开创了一片天地。我成为开发 VHDL 覆盖系统的首席系统设计师。这个项目涉及与许多客户的合作,通过这个项目,我获得了大规模可综合的 VHDL 设计的丰富经验,这些设计是由上百个设计师以许多不同风格完成的。

公司方面的变化对本书的第 2 版(1998 年,由 John Wiley 和 Sons 出版社出版)有非常大的影响。3 年过去了,标准委员会终于批准了综合程序包的标准。而且,由于工作关系,我接触到许多其他设计师的工作,对综合器的使用和它在设计周期中的位置有了一个更宽阔的视野,这使得本书比第 1 版更好地面向编写可综合代码的读者;第 1 版确实把太多的精力放在解释综合器的工作原理上了。我认为,本版重点的

改变，尽管改变并不大，却显著地改进了读者阅读本书的感觉。

1999 年，我离开了 TransEDA 公司，自从我离开后，该公司就破产了，不得不解散开发团队。然而，代码覆盖技术和公司名已被其他公司收购，所以 TransEDA 仍然在销售 VHDLCover(一种 VHDL 代码覆盖检查工具)，但目前以 VN－Cover 的名义销售。

离开 TransEDA 之后，我加入了南安普顿(Southampton)大学，并成为该大学派生出的 Leaf－Mould Enterprises(LME)公司的创始人。成立 LME 公司的宗旨是基于我以前所在的电子与计算机科学系的研究课题，期望能开发出一套有商业价值的可把 VHDL 行为描述自动转换为逻辑电路的综合系统。由我负责设计 VHDL 库管理器、编译器和汇编器，以及编写汇编器生成并发的汇编代码，实现汇编代码执行相应的操作，并把 VHDL 行为模型综合成逻辑网表。不幸的是，由于资金问题，LME 公司于 2001 年倒闭。

从那以后，我成为一名自由职业者，在多个领域工作，担任过程序设计师、网络应用设计师、系统工程师和顾问。

本书第 2 版的发行已有 12 年了，观察综合技术领域中发生的变化是很有趣的。主要的变化是设计师们开始尝试系统级综合，系统级综合使用类 C 语言，如 SystemVerilog、SystemC 和 Handel－C。然而，这一点十分清楚：随着综合工具向更抽象的高层次行为发展，对于那些需要对设计的具体电路进行人为干预的设计者而言，用 VHDL 的逻辑综合仍然十分有用。现在，无论对 ASIC 和 FPGA 设计而言，可用的逻辑综合工具实在太多了。

然而，大部分时间里，VHDL 本身几乎没有发生改变，只在 2000 年和 2002 年，这个语言有一些小的改动。然后，在 2008 年，VHDL 发布了重大更新声明，这些更新涉及的范围很广，甚至涉及随 VHDL 语言发布的已递交的预定义的程序包范畴。这些更新中的许多地方影响到综合。所以，需要编写本书的第 3 版来反映这些改变的时机已经到来。我已对整本书做了全面更新，以反映当前的形势，目前完全符合 VHDL—2008 标准要求的仿真或综合的商业工具还尚未面世，但其中一些与综合有关的特性逐渐成熟，已被纳入综合工具，或者作为可下载的附件，供用户使用。

Andrew Rushton，伦敦，2010

目　录

第1章 引 言

本章重点介绍 VHDL 在数字系统设计中的使用方式、创建 VHDL 的历史原因，以及有关 VHDL 语言维护和升级的国际(合作)项目。

1.1 VHDL 设计周期

从概念上说，VHDL 旨在支持硬件设计周期中的每个阶段。读者可以从 VHDL 语言参考手册(IEEE - 1076，2008)(以下简称 LRM)的序言中清楚地看到这一点，该手册对 VHDL 做了如下定义(引文来自 LRM)：

VHDL 是一种专门用于电子系统设计每个阶段的正规的表示方法。因为它具有机器可读性和人类可读性，所以 VHDL 支持硬件设计的开发、验证、综合和测试；支持硬件设计数据的交流；支持硬件的维护、修改和采购。

关键词是'每个阶段'。这就意味着，VHDL 旨在覆盖从系统规范的制订到网表产生的整个设计周期中的所有阶段，所以 VHDL 语言体系相当庞大而且冗长。然而，学习 VHDL 未必困难。读者可以将 VHDL 看作是一种适用于设计周期中一个或多个设计阶段，包含了多阶段子语言的混合体，每个阶段都有各自适用且能有效覆盖该阶段描述需求的子语言，而该阶段的子语言又是 VHDL 语言体系的一个子集。假如读者能对 VHDL 各阶段子集的内容做一个明确的说明，则每个子集的学习都会变得相对容易。

理想的设计过程通常使用 3 个子集—— 因为有 3 个阶段需要使用 VHDL。这 3 个阶段分别是：系统建模(规范阶段)，寄存器传输级(RTL)建模(设计阶段)和网表(实现阶段)。除了这些基于 VHDL 的阶段外，还有一个初始需求描述阶段，它通常使用通俗易懂的人类语言。因此，每个设计都有 3 个阶段的转换：从需求到规范，从规范到设计，从设计到实现。前两个阶段由设计师完成，最后一个阶段目前主要由综合器

来完成。图 1.1 说明了这个理想的设计周期。

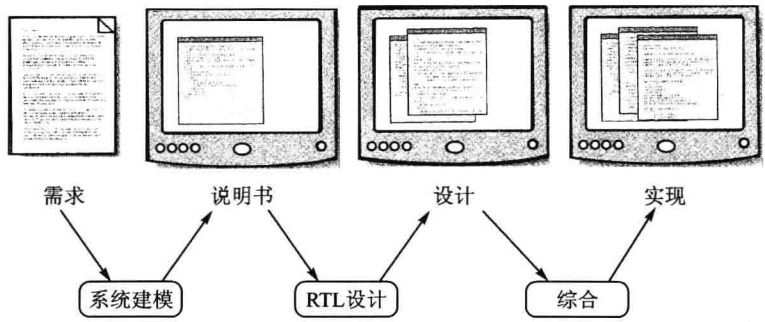

图 1.1 基于 VHDL 的硬件设计周期

通常,系统模型是一个 VHDL 模型,该模型表示的是想要实现的算法,并非具体硬件的实现,目的是创建一个仿真模型,用来替代正规的设计说明书。通过在仿真器中运行该仿真模型,以检查设计的功能是否已经完全满足用户的需求。

系统模型随后被转换成寄存器传输级(RTL)设计,以便为综合成具体电路做准备。转换的目标是实现特定的硬件,但在这个阶段,实现的只是粗粒度级的硬件结构块。该阶段需要确定以时钟周期为基准的时序细节。此外,还需要在块级层次指定实现中要用到的特定硬件资源。

设计周期的最后阶段是对 RTL 设计进行综合,产生出一个网表,这个网表应能满足对实现电路的面积约束和时序需求。当然,实际情况可能并非如此,通常还需要进行修改,这将对设计过程的起始阶段产生影响。以上这几个阶段是使用 VHDL,通过逻辑综合设计电路过程中最基本的、理想的步骤。

1.2 VHDL 的起源

VHDL 源自美国国防部。美国国防部以前采购的一些设计是用私有产权的硬件描述语言编写的,这样,不仅不能将设计资料转给二级承包商生产,而且还不能保证在该硬件的生命周期内,描述该硬件的语言是否还能继续存在。解决的办法是采用一种有发展前景的单一的、标准化的硬件描述语言。该语言的细则起源于 20 世纪 80 年代初期提出的超高速集成电路计划(英文缩写为 VHSIC),属于其中的一部分工作。为此,该语言后来被命名为 VHSIC 硬件描述语言(英文缩写为 VHDL)。

如果该语言仅仅满足美国国防部采购的需求,它很可能只是一个对国防部承包商有利益的晦涩语言。然而,更大的电子工程团体认识到该语言开发的重要性,尤其该语言标准化的重要性,因此于 1986 年将该结构性语言移交给 IEEE 管理,从而使其进入公众领域。IEEE 随即开始该语言的标准化工作,并于 1987 年推出 IEEE 标

准1076。该标准的浓缩版保存在VHDL语言参考手册(LRM)中。

1.3 标准化过程

标准化过程的一部分工作就是规定该语言定期升级的标准化方式。VHDL语言标准规定每5年更新一次。实际上,标准的更新却是不定期的,根据需要决定的,不是固定的每5年更新一次。多年来,这个语言已经发生了许多变化,所以有时需要辨别版本。本书通过标记IEEE标准批准年份来辨别版本。例如,原始标准被标记为IEEE标准1076,由于该标准是1987年被批准的,所以它通常被称为VHDL-1987。通常根据标准被批准的年份来标记修订后的标准版本。

下面列出了不同版本对综合特性的影响:

- VHDL-1987:原始标准。
- VHDL-1993:添加了扩展标识符、异或和移位运算符、元件的直接实例化;
  改进了编写测试平台用的输入/输出(I/O);
  VHDL综合子集中的绝大部分条款都基于VHDL-1993;
- VHDL-2000(轻微修订):没有增加与综合相关的条款。
- VHDL-2002(轻微修订):没有增加与综合相关的条款。
- VHDL-2008:添加了定点程序包和浮点程序包;
  添加了类属类型和类属程序包,能使用类属来定义可复用的程序包和子程序。增强了条件语句,输出端口的读取;改进了编写测试平台用的I/O;
  VHDL标准的统一。

如上所述,只有3个版本的VHDL与综合相关:VHDL-d1987,VHDL-1993和VHDL-2008。VHDL-1993是为综合添加有用条款的最终修订本。所以,VHDL-2008是15年里第一次有显著变化的版本。VHDL-2008(Ashenden和Lewis,2008)中添加了许多特色条款,其中大部分与综合有某种程度的关联。

然而,综合工具的供应商历来对新添加的语法条款采取较保守的态度。这是因为综合的重点是综合后生成电路的质量和综合优化产生的效果,而不是支持更多的语法特色。这意味着,VHDL-2008中新添加的重要语法特色将需要过许多年后才有可能被综合工具实现,而其中许多语法特色将永远不可能实现。实际上,综合用户仍在使用VHDL-1993,并在可预见的未来将继续使用VHDL-1993。

因此本书主要基于VHDL-1993来编写。最近关于综合器功能扩展的讨论主要集中在VHDL-2008中新添加的定点程序包和浮点程序包,这两个程序包已作为VHDL-1993兼容的程序包,可以直接用于综合。目前的综合器尚未能支持VHDL-2008标准中其余新的特色条款。

1.4 VHDL 标准的统一

VHDL-2008 把定义语言不同部分的多个标准及其环境标准归纳为一个统一的标准。这是 VHDL-2008 标准的最大改变之一。

标准化过程的管理属于 VHDL 分析和标准化组(英文缩写 VASG)的工作范畴,VASG 是 IEEE 标准化组织机构的组成部分。除了以语言本身为主体的标准化工作外,还有若干个致力于 VHDL 使用方式标准化的工作小组。过去这些工作组已经发布了他们各自的标准。例如,有一个工作组致力于 VHDL 的模拟建模(VHDL-AMS-VHDL 模拟混合信号-标准 1076.1);一个工作组致力于标准的可综合数值程序包(VHDL 综合程序包-标准 1076.3(1997));一个工作组致力于加速门级仿真(VITAL-面向 ASIC 库的 VHDL 基准-标准 1076.4);还有一个工作组致力于可综合成逻辑的 VHDL 标准的解释(VHDL 综合可移植性-标准 1076.6)。另外,几乎所有综合工具一致采用的 9 值逻辑类型(即 std_logic)已发展成一个完全不同的 IEEE 标准(VHDL 多值逻辑程序包-标准 1164)。

在语言发展的初期,对 VHDL 的不同应用领域分别进行标准化是高效的,因为它允许子工作组独立于 VHDL 总体标准化过程开展工作,还意味着当条件成熟时,他们能发布各自的标准,而不用等到下一版 VHDL 标准的正式发布。然而,当工作组的工作变得成熟、稳定和经常化时,分别标准化就会产生问题。例如,VHDL 新标准的发布很有可能造成子工作组的标准出现滞后的现象,如果使新标准与以前的 VHDL 标准版本兼容,则又会造成新标准缺少新特色的缺憾。

所以,在 VHDL-2008 中,那些针对综合的工作组标准已经部分融入了 VHDL 标准本身。标准 1076 现在包含了标准逻辑类型(1164)、标准数值类型(1076.3)和部分标准综合解释(1076.6)。对于用户而言,并没有什么不同,但是确实已将语言的这些部分确认为 VHDL 不可或缺的一部分,并确保未来它们将与语言同步发展。

正如读者可能想到的那样,这些材料使得 VHDL 语言参考手册(IEEE-1076,2008)(的范围和规模)变得相当得庞大。

1.5 可移植性

由于可综合的 RTL 设计方法与半导体制造工艺技术之间无直接的联系,所以用可综合的 RTL 设计方法所完成的设计项目享有很长的寿命。同一个设计可以映射到不同的工艺,采用不同的制造技术实现,甚至在原设计代码编写完成多年后,还可对原设计稍加修改,用最新的工艺实现。因此,为设计做长期支持规划以及使用一

种安全常见的 VHDL 风格编写代码是明智的做法，而不是借助于某综合工具特有的技巧来进行综合。可以预见，在未来数年内常见风格的代码可获得一般综合工具的广泛支持，而那些靠特定技巧的综合可能不会继续获得支持。

换用不同的 EDA 工具，或者由于选用不同的制造工艺技术，工程师们不得不换用新的综合工具，这些都是公司内很正常的变动。所以，采用可综合 VHDL 中可移植子集编写代码是很好的做法，该子集适用于多种不同的综合工具。

综合器在处理 VHDL 代码时，根据它对一组 VHDL 模板所做的解释，把代码转换成相应的电路网表，这是造成可移植子集问题的原因。历来每个综合器供应商都发展出自己的一组模板。这就意味着，每个综合工具支持一个略有差异的 VHDL 子集。然而，这些子集之间总是存在许多重叠部分，本书试图找到它们的共同点。

IEEE 设计自动化标准委员会已经为 VHDL(IEEE－1076.6，2004)指定了一个综合标准，该标准似乎是一个超集，而不是由商业工具支持的子集，这样做把这件事搞得更复杂了。因此，遵守这个标准并不意味着，使用任意指定的综合工具就可对设计进行综合处理。而且，看来由任何单一综合工具实现符合该综合标准的每个细则似乎是不可能的。

建议使用所有综合工具都支持的公共子集来编写可综合代码。因此，本书侧重于 VHDL 中可综合的公共子集，并避免过多地涉及某些工具所特有的晦涩的 VHDL 语法条款，即使那些晦涩的语法条款已包含在综合标准之中，也尽可能不要使用。