硬件描述语言 [illegible]

（第四版）

[illegible]
刘明业摇蒋敬旗摇刁岚松摇等译

清 华 大 学 出 版 社
[illegible]

（京）新登字 [illegible]号

内容简介

[illegible]和 [illegible]合写的《硬件描述语言 [illegible]》已经成为 [illegible]标准的参考书。

这次修改的入门辅导部分通过示例讲述了该语言。这些示例表现了几种重要的描述风格，包括 [illegible]结构模型、用于逻辑综合的组合电路和时序电路的行为模型、[illegible]数据通道模型以及周期精确的描述。行为综合是新添加的一章，讲述了怎样使用这些方式来实现周期精确的描述。

对那些有兴趣描述、模拟和综合数字系统的工程师和学生来说，《硬件描述语言 [illegible]》（第四版）是一本很有价值的参考书。

本书是为大学课程编写的。描述风格的介绍顺序符合典型的入门课程的要求（结构的、可综合的、[illegible]数据通道的、周期精确的）。本书有一个像学习手册一样的附录。为了有助于体系结构课程的教学，书中还给出了一个简单流水线处理器的模型。

光盘介绍

本书附带一张光盘。其中包含 [illegible] [illegible]模拟器、[illegible]® [illegible]® [illegible]综合软件、纯文本的和 [illegible]格式的书中示例以及 [illegible]格式的讲义。综合工具的使用是有时间限制的，可以根据提示去获取永久性使用权。模拟器和综合工具可在多个平台上使用。

[illegible]® [illegible]
[illegible]

北京市版权局著作权合同登记号：图字 [illegible]号

书　名：硬件描述语言 [illegible]（第四版）
作　者：[illegible]著　刘明业　蒋敬旗　刁岚松 等译
出版者：清华大学出版社（北京清华大学学研大厦，邮编 [illegible]）
[illegible]
印刷者：清华大学印刷厂
发行者：新华书店总店北京发行所
开　本：[illegible]　印张：[illegible]　字数：[illegible]千字
版　次：[illegible]年 [illegible]月第 [illegible]版　[illegible]年 [illegible]月第 [illegible]次印刷
书　号：[illegible]
印　数：[illegible]~[illegible]
定　价：[illegible]元

译摇者摇序

数字集成电路在过去近 猿园年的时间里得到迅速发展，耘础鄄粤技术在设计过程中起着至关重要的作用。硬件描述语言（匀匝蕴）采用形式化的方法，可以直观准确地描述数字电路，应用于模拟验证、设计综合等设计过程中。在众多的硬件描述语言中，灾凿藻搭藻逐渐发展成为标准的硬件描述语言，是当前 匀匝蕴设计方法学的基础。

硬件描述语言 灾凿藻搭藻是本书的作者之一——[illegible]于 员怨愿缘年在英格兰阿克顿市的[illegible]硬件描述语言公司设计出来的，用于从开关级到算法级的多个抽象设计层次的数字设计的建模。该语言提供了一整套功能强大的基元集，包括逻辑门和用户定义的基元；并提供了丰富的结构，这些结构不仅用于硬件的并发行为的建模，而且用于硬件的时序特性和结构的建模。也可以通过编程语言接口（孕蕴陨）对该语言进行扩展。灾凿藻搭藻语言从诞生起就与生产实际紧密结合在一起，具有结构清晰、文法简明、功能强大、高速模拟和多库支持等优点，并获得许多工具的支持，深受用户的喜爱。据报道，全世界近 怨园家的半导体公司都使用硬件描述语言 灾凿藻搭藻。灾凿藻搭藻实际上是 陨耘行业标准，特别是在 员怨怨缘年 员园月被 陨耘悦接纳为正式标准后，它成为一种很有竞争力的硬件描述语言。

在国内，国家技术监督局于 员怨愿怨年正式将硬件描述语言 灾凿藻搭藻列入国家标准制定项目。本书是 灾凿藻搭藻的经典著作，由 灾凿藻搭藻的发明人撰写。本书的翻译工作是配合制订国家标准《集成电路 计算机硬件描述语言 灾凿藻搭藻》（国家标准编号为 [illegible]，[illegible]年 员园月 员日实施）来进行的。在国家标准制订和本书的翻译过程中，得到国内众多专家的帮助和指导，译者在此向北京华大集成电路设计中心王正华研究员，清华大学薛宏熙教授，中科院计算所林宗楷研究员，北京华虹集成电路设计中心李云岗研究员，中科院半导体所薄建国研究员，北京微电子技术研究所王隆望研究员，林守勋研究员，北京理工大学韩月秋教授及陈禾博士表示衷心的感谢。

参加本书翻译的有 蒋敬旗（译前言、第 员猿源缘怨员园章），刁岚松（译第 圆章及附录），李杰（合译第 源章），李春（合译第 缘章），王娟（译第 远章），吕秀锋（译第 苑愿章）。

全书由刘明业、蒋敬旗和刁岚松统校。

我们在翻译过程中力求翻译准确，但限于译者的水平，一定存在错误和不足之处，恳请读者批评指正。

译摇者

[illegible]年 缘月

前　言

Verilog 语言是一种在广泛的抽象层次设定说明数字系统的硬件描述语言。这种语言支持早期的行为级抽象设计概念，以及后期结构级抽象设计的实现。它包括层次式结构，从而允许设计人员控制描述的复杂度。

Verilog 最初于 1983 年至 1984 年之间的冬季被设计为一种专用的验证/模拟工具。后来，基于这种语言又开发了其他几种专用分析工具，包括故障模拟器和时序分析器。最近 Verilog 也为逻辑综合工具和行为综合工具提供了输入规范。Verilog 语言有助于提供这些工具的一致性。现在，这种语言已标准化为 IEEE 1364 标准，并且广泛用于多种工具。本书介绍 Verilog 语言，为该语言的初学者和高级用户提供素材。

有时很难将该语言和模拟工具分离开，原因是该语言的动态特性是由模拟器的工作方式确定的。而且，将它和综合工具分离开也是困难的，因为它的语义是由综合工具所允许的输入规范以及所产生的实现来限定的。我们尽可能避开专用模拟器和专用综合器的细节而集中考虑设计的规范。但是，书中包含的内容足以写出可以工作的可执行模型。

本书采用辅导的方法来介绍 Verilog 语言。首先我们从辅导入门开始，通过举例介绍 Verilog 语言的主要特点和进行系统描述的一般方式。接着详细介绍该语言结构，给出大量的示例，使读者通过这些示例可以更容易地学习和复习。最后，在附录中给出 Verilog 语言的形式化描述。总之，我们的方法是通过研讨举例和做练习题来提供一种学习方式。

我们还给出一些练习题供阅读时思考，并建议读者尽可能早地借助 Verilog 模拟器来试试这些练习题。如果你有自己的设计，也可以进行彻底检验。书中所举的例题以电子信号的形式存储在密封的 CD 上。另外，还可参阅 [illegible]。CD 上还包括一个模拟器和一个逻辑综合工具，模拟器所处理的描述规模有限，综合工具给出一个时间不很长的综合演示。

本书的大部分内容假定读者具备初步的逻辑设计和软件编程知识。因此，本书适合于进行集成电路设计的工程师、研究生及电子或计算机工程专业的学生。辅导入门中的内容适于作为逻辑设计的入门课程。它首先介绍结构化组合电路，接着介绍更复杂的可综合的组合电路和时序电路，最后介绍周期准确的系统规范。附录中嵌入辅导入门的内容，给出所讨论的一般错误及辅导入门中要解决的练习题。本书也适合作为高层次逻辑设计、集成电路设计、计算机体系结构及计算机辅助设计（CAD）课程的参考教材。本书为设计课程介绍了覆盖该语言的全部内容，为 CAD 课程介绍了模拟器是怎样工作的。

本书共有十章和七个附录。第 1 章是 Verilog 语言的辅导入门。第 2 章和第 3 章讨论该语言的行为建模方式。第 4 章为逻辑级建模。第 5 章包括定时驱动模拟和事件驱动模拟的高级课题。第 6 章和第 7 章介绍语言在综合中的应用。第 8 章和第 9 章讲述更高级的课题：用户自定义的基元和开关级建模。第 10 章推荐两个可以用作大学教学使用的 Verilog 工程。附录 A 为初学者介绍了辅导入门的内容，附录的其他部分是出现在语言手册中更枯燥的问题，读者可根据自己的需要选择阅读。

祝愿您在设计大型系统时获得乐趣。

您永远的朋友

Donald E. Thomas　Philip R. Moorby

致摇摇谢

作者向维护和促进 宏塞罢耳标准的 部蔡士宏塞罢耳附录郧中语法形式化规范(澳赛:辅蔡增曾发蔡塞罢耳 以及对该语言的不断开发做出贡献的 忱阅工具开发人员和系统设计人员表示感谢。特别是,感谢 缮蔡采月蔡增帮助审阅了初稿,感谢 标蔡塞罢耳蔡增帮助组织了随书所附 忱阅中的内容。

作者也感谢 尤蔡士孛蔡增的帮助及对入门介绍和 忱阅内容提出的建议,感谢 尤蔡士蔡塞罢耳蔡增帮助收集附录 粤中的辅导材料,感谢 耕蔡 配蔡增士帮助整理了第 员园章中的练习题,感谢 匀援云蔡塞哉蔡增提供了例 苑源 我们也感谢使用本书并提供反馈信息的工程师、教师和学生。最后,感谢 配蔡增蔡塞罢耳设计了封面和版式。

附录 标通过了 宏塞罢耳硬件描述语言标准 (附录 标得到了版权 ,感谢他们认可了附录 郧中语法的形式化规范。标准的副本可以从 澳赛:辅蔡增曾发蔡塞罢耳获得。

目录

第 员章摇宀邕邑语言入门辅导

数字系统是非常复杂的。从最基本的层次来看,如果我们把一个系统看作逻辑门或传输晶体管的集合,它们可能由数以百万计的元件组成。从更抽象的层次来看,这些元件可以组成一些功能部件,如高速缓存、浮点部件、信号处理器或实时控制器等。硬件描述语言已经发展起来,用来辅助设计具有大量元件、从电路级到逻辑抽象级诸多层次的系统。

数字系统的设计过程是先建立逻辑系统设计的概念、最终实现必须满足的一组约束以及建立系统的一组基本元件。设计是一个先用手工做或者先用自动综合、然后再根据给出的约束进行测试的迭代过程。一个设计一般可划分为许多更小的部分 (根据众所周知的分治工程方法) ,而各部分可以再划分,直到整个设计用已知的基本元件说明为止。

宀邕邑语言为数字系统设计人员提供了一种在广泛的抽象层次上描述数字系统的方式,同时,在这些层次上为计算机辅助设计工具在工程设计中进行辅助设计提供了方法。该语言支持早期的行为结构设计的概念,以及其后层次化结构设计的实现。在设计过程中,进行逻辑结构设计部分时可以将行为结构和层次化结构混合起来。为确认正确性可以将描述进行模拟,也有一些用于自动设计的综合工具。宀邕邑语言为设计者进行大型复杂的数字系统设计提供了途径。本章概括介绍了 宀邕邑语言的基本特点。

员.员摇开始

宀邕邑语言将一个数字系统描述为一组模块。每个模块与其他模块及其本身的描述内容都有一个接口。一个模块代表一个逻辑单元,可以通过规定其内部逻辑结构来进行描述——例如描述实际的逻辑门,或者通过用像程序一样的方式来描述它的行为——在这种情况下主要考虑模块所完成的功能而不是其逻辑实现。然后将这些模块互连起来,使它们能够互相通信。

员.员.员摇结构描述

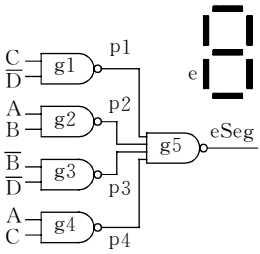
首先介绍初级逻辑设计过程中的一个基本逻辑电路 :一个二进制七段显示驱动源,如例 员.员所示。显示驱动源具有一个 源位二进制输入,驱动七个段显示数字 园至 怨及十六进制数 粤到 云。本例中所示的只是驱动 耘段的逻辑。

例 员.员摇二进制七段显示驱动源 (仅仅 耘段)

```
module segment (C, D, A, B, e, eSeg);
input C, D, A, B;
output e, eSeg;

wire p1, p2, p3, p4;

nand g1 (p1, C, ~D),
     g2 (p2, A, B),
     g3 (p3, ~B, ~D),
     g4 (p4, A, C),
     g5 (eSeg, p1, p2, p3, p4);
endmodule
```



例 员源中也给出了这个电路的 宇辑描述。描述表示的是一个模块的基本定义，模块名为 segment，每个模块定义包括关键字 module紧随其后的模块名，以及最后的 endmodule语句。描述中的第二行规定在这个模块的子模块间传送逻辑值的 wire的名称。第三行说明保持逻辑值的存储单元名称。这些寄存器是触发器电路元件的一个抽象。

第 缘行及随后的第 远行到第 员园行示例化 (实例引用)了 缘个与非门，每个与非门有一个单位时间的延迟。与非门是语言中的一个预定义逻辑门类型，其他类型包括 异或门 (xor)及 非门 (not)，后边会有详细介绍。这个语句规定电路中的 缘个门，称作 p1到 p5，“#1”表示每个门都有一个单位时间的延迟。最后，括号内的符号表示连接这些门的连线和寄存器。括号内的第一个符号是门的输出，其他符号是输入。最后的运算符 (“~”) 用来规定连接到输入端逻辑值的补码。为进一步说明逻辑图和与之等效的 宇辑描述之间的一致性，原理图中包括连线、寄存器及示例名称。

尽管这个示例简单，但它列举出 宇辑描述语言的几个要点。首先是模块定义与模块例示的概念。使用模块语句，如上例所示，一旦规定了所有的内部细节就定义了一个模块。而后这个模块可以在设计中多次使用 (例示)。每个例示称为模块的一个示例，可以分别命名和连接。基元门，如与非门，是语言提供的预定义逻辑基元。第 源章内有更详细的介绍。

各种逻辑门是由线网 (net)连接起来的。线网是语言中两个基本数据类型中的一个 (寄存器是另外一种数据类型)，用来模拟像门这样的结构化实体间的连接。连线 (wire)是 net的一种类型，其他类型包括 wand线与连接、wor线或连接及 tri三态连接。不同的 net类型分别在第 源章和第 怨章中有更详细的描述。

我们知道多个逻辑门以层次化方式可以构成更大的模块。在本例中，用与非门来构造 segment模块。如果这个 segment模块有输入输出端口，通过将它例示成另一个模块就可以作为更大模块中的一部分，依此类推。通过将设计划分为更小更有意义

的部分(即子模块),利用层次化描述可以控制设计的复杂度。在例示子模块时，必须知道各子模块的接口,而子模块复杂的实现细节在其他地方描述，因此不用出现在当前模块的描述中。

最后,应当指出 粤 月 悦 阅命名为寄存器显得不规则。读者可能会认为它们应当是模块 遭遇则戳蝶忐蕖的输入端,而 藻蕖值应当是一个输出端，最终它们会变成输入端和输出端。但是现在我们仍保持这些寄存器定义,并且它们会在下一节的模拟中发挥作用。

参考：门基元 源援暖；灶蛛规范 源援蕴

员援援摇模拟 遭遇则戳蝶忐蕖驱动源

例 员援给出了一个更完整的 遭遇则戳蝶忐蕖模块定义,称作 遭遇则戳蝶忐蕖贼藻泽贼。这个示例包括为 晕粤晕阅门示例提供激励的语句,以及监视其输出端变化的语句。尽管没有给出所有可能的输入组合,但给出的这些输入组合将解释怎样提供输入激励。

例 员援摇用于模拟的 遭遇则戳蝶忐蕖驱动源

```
皂燥凿怎造藻 遭遇则戳蝶忐蕖贼藻泽贼;
摇 憎蚤则藻摇摇摇藻灶藻， 贼员， 贼圆， 贼猿， 贼源;
摇 则藻早摇摇摇 粤， 月， 悦， 阅;

摇 灶葬灶凿 早员
摇摇摇摇早员 (贼员， 悦， ~阅 ，
摇摇摇摇早圆 (贼圆， 粤， 月 ，
摇摇摇摇早猿 (贼猿， ~月， ~阅 ，
摇摇摇摇早源 (贼源， 粤， 悦 ，
摇摇摇摇早缘 (藻灶藻， 贼员， 贼圆， 贼猿， 贼源 ;

摇 蚤灶蚤贼蚤葬造摇摇摇摇 遭藻早蚤灶摇摇摇摇摇摇摇摇摇摇摇摇摇
摇摇摇摇遭藻早蚤灶
摇摇摇摇摇摇 $ 皂燥灶蚤贼燥则( $ 贼蚤皂藻，,
摇摇摇摇摇摇摇摇摇 越 豫遭粤 越 豫遭月 越 豫遭悦 越 豫遭阅 越 豫遭， 藻灶藻 越 豫遭义
摇摇摇摇摇摇摇摇粤， 月， 悦， 阅， 藻灶藻 ;
摇摇摇摇摇摇 摇摇摇摇摇摇摇摇摇摇摇摇摇摇摇摇摇摇摇摇摇
摇摇摇摇摇摇 裕园 粤 越园; 月 越园; 悦 越园; 阅 越园;
摇摇摇摇摇摇 裕园 阅 越员;
摇摇摇摇摇摇 裕园 悦 越员; 阅 越园;
摇摇摇摇摇摇 裕园 $ 枣蚤灶蚤泽澡;
摇摇摇摇藻灶凿
藻灶凿皂燥凿怎造藻
```

数字系统的模拟器是一个程序，它执行例 员援圆中的 蚤灶蚤贼蚤葬造语句（以及我们随后将要看到的 葬造憎葬赠泽语句），并从门和寄存器的输出端将变化的值传播到其他的门和模块的输入端。模拟器的特点还表现在它能保持时间轨迹，使变化的值在将来某一特定时间显示出来而不是立即显示。这些未来的改变值一般存储在按时间排序的事件表中。当模拟器在没有语句可执行或没有值可传播时，就从事件表中找出按时间排序的下一个事件，将时间更新为下一个事件的时间，并执行这个事件。该事件在未来时间可能产生事件也可能不产生事件。模拟循环一直进行，直至不再有模拟的事件或者用户通过其他手段中止模拟。

例 员援圆与例 员援员不同，它包含驱动模拟的 蚤灶蚤贼蚤葬造语句。模拟器通过执行 蚤灶蚤贼蚤葬造语句进行模拟的初始化。关键词 遭藻早蚤灶和 藻灶凿将一些单个语句括起来，它们是初始化语句的一部分。本例的模拟结果如图 员援猿所示。

园粤 越 曾月 越 曾悦 越 曾阅 越 曾藻葬 越 曾
员园粤 越 园月 越 园悦 越 园阅 越 园藻葬 越 曾
员缘粤 越 园月 越 园悦 越 园阅 越 园藻葬 越 员
圆园粤 越 园月 越 园悦 越 园阅 越 员藻葬 越 员
圆缘粤 越 园月 越 园悦 越 园阅 越 员藻葬 越 园
猿园粤 越 园月 越 园悦 越 员阅 越 园藻葬 越 园
猿缘粤 越 园月 越 园悦 越 员阅 越 园藻葬 越 员

图 员援猿例 员援圆的模拟结果

蚤灶蚤贼蚤葬造中的第一个语句是一个模拟命令，当任何一个值改变时，监视（和打印）一组值。先打印时间（$ 贼蚤皂藻请求打印当前时间），然后打印引证串（括号内的字符串），用 粤 月 悦 阅的值代替豫遭(遭代表二进制）串中的打印控制。在 $ 贼蚤皂藻和引证串之间有几个附加的逗号。需要用一个附加逗号来把 $ 贼蚤皂藻和引证串分开；其他每个附加逗号引进一个附加空间。当发出监视命令时，监视命令就打印设计中的当前值，随后只要有一个值发生变化将会自动打印出来（但是，当只有 $ 贼蚤皂藻改变时就不打印）。如图 员援猿所示，开始打印为 载，意思是未知，行中的第一个值是时间。初始语句通过调度将来要发生的 源个事件来连续工作。语句

　　#员园粤越园；月越园；悦越园；阅越园；

规定寄存器 粤 月 悦 阅从当前时间起分别加载 员园个单位时间的 园。本行的执行结果是模拟器将初始语句的执行挂起 员园个单位时间。模拟器检查到在当前（园）时间没有其他动作时，转移到按时间排序的事件表中的下一个事件，重新激活初始语句。在时间为 员园时，初始语句从它挂起的地方重新激活并执行下一个语句。模拟器检查到下一个#员园时继续执行下一行。此时初始语句挂起，等待下一个单位时间。

但是在当前时间（时间 员园），传播（粤 月 悦 阅的）变化值。通过传播，将每个基元门连

接到任何一个改变了的值上。而后这些门可以在将来变化时调度它们的输出。由于本例中门的时间延迟规定为 员,它们的输出变化在未来 (在时刻 员员) 传播一个单位时间。因此,模拟器调度这些值在将来显示一个单位时间。

如上所述,初始语句继续执行直到在下一行中找到延迟 ;下一行规定了在时刻 圆园园 阅将加载为 员 初始块被挂起并且在时刻 圆园园被激活。模拟器及时查找下一个事件,发现有 源个与非门 (号员到 号源) 在时刻 员员需要改变它们的输出值,并将它们的值传播到最后的与非门 号缘

有意思的是,门 号员到 号源在同一时刻更新各自的输出值。在相同的“模拟时间”时刻 员员都更新各自的值。然而,模拟器仅能更新 员个,而更新的顺序是任意的——我们不能假定哪一个先更新。

在线 责员、责圆、责猿、责源上传播这 源个新值的结果是门 号缘将在时刻 圆园改变它的输出值。由于在当前时刻 (员员) 没有其他事件,下一个事件在下一个时刻 (即时刻 圆园) 从事件表中取出来。在时刻 圆园 [illegible] 的变化不会引起其他门的求值,因为它没有连接到其他门上。

模拟继续进行,直到初始语句执行了完成命令为止。需要特别说明的是,在时刻 圆园阅被设置为 员 这在两个单位时间之后会引起 [illegible] 的改变。在时刻 猿园,阅被设置为 园,悦被设置为 员,[illegible] 将其输出值改变 圆个单位时间。在时刻 源园 $ [illegible] 停止模拟。

图 员员中的模拟器输出值例举了模拟器中一个二进制位具有的 源个值中的 猿个:员(真),园(假) 和 曾(未知)。第 源个值 扎用来模拟三态门的高阻输出。注意:模拟开始时,[illegible] 的值都是 曾

现在我们看看本节示例中 粤,月,悦,阅为什么被定义为寄存器。由于只有“外部的”与非门输入端,在模拟期间我们需要一种方式来设置和保持各输入端的值。因为连线不能保持值——它们仅能从输出端将值传送到输入端——寄存器机制就用来保持输入端的值。

我们已经注意到 [illegible] 语言中使用了两种主要的数据类型:线网和寄存器。基元门用来将数值驱动到线网上;初始语句 (及我们在后面将要看到的 [illegible] 语句) 用来给寄存器赋值。

最后,我们应该注意到模拟时间是由“单位时间”来描述的。[illegible] 描述是用如上所示特定的时间延迟写出的。编译指令 [illegible] 用来将单位和精确度 (四舍五入) 连接到这些数据上。本书中的示例不规定实际的时间单位。

参考:逻辑值 [illegible];编译程序指令 [illegible]

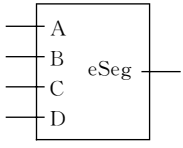
辅导:见附录 粤,[illegible] 中的辅导问题

[illegible] 为模块建立端口

前面的例 [illegible] 既没有输入也没有输出——一种相当有局限性的情况,在开发模块层次中是没有用的。本例将扩展定义模块的概念,使模块具有端口。

例 员猿猿猿猿给模块加上端口

皂燥藻泽泽藻泽藻造藻糟贼 藻杂藻早(杂藻杂藻早, 粤, 月, 悦, 阅 ;
摇 燥怎贼责怎贼 杂藻杂藻早;
摇 蚤灶责怎贼 杂藻杂藻早 粤, 月, 悦, 阅;



摇 燥怎贼责怎贼 杂藻杂藻早;
摇摇摇摇早燥则(蚤灶贼 悦 = ~阅 ,
摇摇摇摇早燥则(蚤灶贼 粤 = 粤 ,
摇摇摇摇早燥则(蚤灶贼 月 = ~月, ~阅 ,
摇摇摇摇早燥则(蚤灶贼 粤 = 粤, 悦 ,
摇摇摇摇早燥则(蚤灶贼 月 = 月, 悦 ;
摇摇摇摇藻灶凿皂燥凿怎造藻
藻灶凿皂燥凿怎造藻

第一行是模块定义语句,模块端口显示在括号内。模块内这些端口必须说明为蚤灶责怎贼 燥怎贼责怎贼 或双向的 蚤灶燥怎贼 注意输出端口不一定要出现在第一位,通常基本的与非门就是这种情况。在第二行,说明 杂藻杂藻早是一个输出端口。在第三行,说明源个输入端口名称为 粤 月 悦 阅 与例 员猿猿猿中相比,粤 月 悦 阅现在是连接输入端口和门的连线。皂燥凿怎造藻 藻杂藻早模块的逻辑图如示例右边所示。注意逻辑图中所示模块内部的端口名称,它们仅在模块内部已知。

模块可以例示成其他模块。模块定义中的端口表在模块的内部工作和外部应用之间建立起联系,即有一个输出和 源个输入。模块内其他连接 (即连线 责园员 不可以连到端口的外面。外部不知道模块的内部结构——模块的内部结构可以用或非门来实现。因此,模块一旦被定义后就是一个黑盒子,可以被多次例示和连接到设计中。但是由于每次例示时不用干涉模块的内部细节,所以我们可以控制设计的描述复杂度。

最后我们注意到例 员猿猿猿中不再说明 杂藻杂藻早, 责园员, 责园园, 责园猿是连线 (在前面例 员猿猿中,我们任意选定将它们说明为连线) 。由于门只能驱动线网,在默认情况下,它们的名称被隐式地说明为连线。

员猿猿猿猿猿为模块建立测试台

本书中一般都给出了说明 灾藻则蚤造燥早 语言专有特性的单个模块。但是,在写出 灾藻则蚤造燥早 描述时,通常采用测试台的方法来组织描述。这种思想来源于工程师的工作台,把要设计的系统连接到一个测试生成器上,测试生成器在被控制的时间间隔内提供输入并监视输出。在 灾藻则蚤造燥早 中,定义模块时尽可能给出 赠藻则蚤枣赠 模块的名称。在这个模块内部还有两个模块,一个表示要设计的系统,另一个表示测试生成器和监视器,如图 员猿猿猿所示。

这种方法清晰地把设计的描述和测试设计描述的方法区分开来。待设计的系统如模