VHDL

入门与应用

陈雪松　滕立中　编著

人民邮电出版社
www.pptph.com.cn

前　言

VHDL 语言是符合美国电气和电子工程师协会标准（IEEE 标准 1076）的超大规模集成电路硬件描述语言，它可以用一种形式化方法来描述数字电路和设计数字逻辑系统。利用 VHDL 进行自顶向下的电路设计，并结合一些先进的 EDA 工具软件，可以极大地缩短产品的设计周期，加快产品进入市场的步伐，特别适用于当今高速发展的信息产业。

当前，VHDL 语言在国内的硬件设计中已得到了较为广泛的应用，几乎是每个硬件设计工程师必备的工具。同时我们应当看到，由于受到国内 ASIC 设计工艺的限制，大部分集成电路的设计还处于半定制 ASIC 阶段，即对 CPLD/FPGA 的设计阶段。目前虽然有了一些 VHDL 方面的参考书籍，但大都基于 VHDL 语言的语法本身，而没有详细介绍如何使用 VHDL 进行具体的 FPGA/CPLD 设计，对相关知识介绍的也比较少，而这一部分往往也是使用 VHDL 进行开发的人员最需了解的地方。基于此，我们编写了这本书，详细介绍了使用 VHDL 进行中小规模 FPGA 设计的方法，并将介绍的重点放在了实际应用上。同时本书还具有如下几个特点：

- 从事硬件开发的人员一般也熟悉部分传统程序设计语言，如 C 语言。实际上，VHDL 作为一种语言，与这些传统程序设计语言有非常大的相似性。本书在介绍 VHDL 语言语法时，与传统程序设计语言进行了对比，以便利用传统程序设计语言更快地熟悉并掌握 VHDL 语言。
- IEEE 的 VHDL 语言标准中定义了较完备的 VHDL 语法集，但许多 VHDL 语法（如文件类型）并不能被多数厂商的编译综合软件所支持。同时许多特定的语法是用于 ASIC 设计的，并不为 FPGA 开发人员所使用，开发 FPGA 所需使用的往往只是此标准的一个子集。因而本书在介绍语言时，只介绍了一些基本的、必需的语法，而省略掉了一些不常用的语法，这使本书更为简练易懂。
- 本书以 VHDL 为立足点，但又不仅仅满足于介绍 VHDL 的语法，同时还提供了一些看似简单，但又非常典型的程序设计实例。由于 VHDL 是一种硬件描述语言，加强对 VHDL 语言硬件特性的介绍是本书的一大重点。同时，对于 VHDL 的初学者而言，语法固然重要，但同样要注重练习的过程。特别是对于这种硬件描述语言，不同的写法，虽然都能编译通过，完成同样的功能，但会具有不同的综合效果，反映到硬件上，就是占用了不同的资源，具有不同的性能。基于此情况，作者加入了一些对基本数字电路单元，如移位寄存器，编译码器等用 VHDL 语言描述的一般写法。通过它们，读者可以知道如何正确使用 VHDL 语言来高效地设计电路，并在此过程中加深理解 VHDL 语言的特性。同时，这些特例也可以为读者所用，融合在具体的设计中。
- 本书还主要以 Altera 和 Xilinx 的器件为例，介绍了 FPGA 内部的基本结构，同时也介绍了相关的一些用于 PC 机的 EDA 工具软件。学习完本书后，读者将能够使用常

用 EDA 工具软件编辑 VHDL 文件，使用仿真工具仿真所设计的电路系统，并学会使用综合工具以及下载工具，最终达到能够独立设计硬件电路系统的目的。

本书第 1 章介绍了 EDA 工具和 VHDL 的基础知识；第 2、3 章详细介绍了 VHDL 语言；第 4 章给出了一些设计实例；第 5 和第 6 章分别介绍了较为常用的两个 FPGA 开发工具软件包 MAX+plus II 和 Quartus；第 7 章介绍了 FPGA 开发的一些高级技巧。在本书的后记中，还对微电子技术、可编程器件及 EDA 工具等的发展趋势作了简要的介绍。

本书主要面向使用 VHDL 语言进行 FPGA/CPLD 器件设计的硬件设计人员，同时也面向对此感兴趣的科研人员、系统设计者、电子电路设计爱好者及相关专业的大学生。本书特别适合在 PC 机上进行一些中小规模器件设计的人员参阅，可以消除读者对芯片设计产生的高深莫测的印象。通过本书的学习，可以使读者掌握半定制 ASIC 设计的基本过程，最终达到能独立设计一般硬件电路的目的。书中所有的程序都在 MAX+plus II 和 Quartus 下编译通过，并完成了前仿真。书中部分例子中的硬件电路图由 Synopsys 的 Design Compiler 生成。

本书第 1、2、3、5、6 章由陈雪松编写，第 4、7 章由滕立中编写。在编写过程中，得到了人民邮电出版社编辑、老师的大力支持，另外还得到了宋挥师等同志的帮助和鼓励，在此对他们表示感谢。

书中若有不妥之处，恳请广大读者多提宝贵意见和建议。

作　者

2000 年 10 月

目　录

第 1 章　绪论

1.1 EDA 工具的历史及现状

20 世纪 90 年代，个人微机、移动电话、光纤通信、Internet 以及高速数据传输设备迅猛发展。电子设备生产商为了取得商业竞争上的主动权，迫切需要功能强、品质优、成本低、功耗小、集成度高的电子产品以满足不断增长的社会需求。为此，生产商必须用尽可能少的 IC 器件，尽可能小的 PCB 板来研制高集成度的复杂系统。设计复杂程度的增加孕育了对现代电子设计方法和测试方法的普遍需求。

Internet 技术的出现，标志着人类社会已进入信息时代。信息社会与大规模集成电路（VLSI）的广泛应用息息相关，而大规模集成电路又是以硅处理技术为基础的。因此，也有人称当今社会进入了硅石时代。

信息社会的标志性产品是电子产品。现代电子产品的性能越来越高，复杂度越来越大，更新步伐也越来越快。实现这种进步的主要原因就是微电子技术和电子设计技术的发展。前者以微细加工技术为代表，目前已进入超深亚微米阶段，可以在几平方厘米的芯片上集成几千万个晶体管；后者的核心就是电子设计自动化 EDA（Electronic Design Automatic）技术。EDA 是指以计算机为工作平台，融合了应用电子技术、计算机技术、智能化技术的最新成果而开发出的电子 CAD 通用软件包，它根据硬件描述语言 HDL 完成的设计文件，自动完成逻辑编译、化简、分割、综合、优化、布局布线及仿真，直至完成对于特定目标芯片的适配编译、逻辑映射和编程下载等工作。目前 EDA 主要辅助进行三个方面的设计工作：IC 设计、电子电路设计和 PCB 设计。没有 EDA 技术的支持，想要完成超大规模集成电路的设计制造是不可想象的；反过来，生产制造技术的不断进步又必将对 EDA 技术提出新的要求。

回顾近 30 年来电子设计技术的发展历程，可将 EDA 技术的发展分为三个阶段。

70 年代为 CAD 阶段。这一阶段人们开始利用计算机取代手工劳动，辅助进行 IC 版图编辑，PCB 布局布线，产生了计算机辅助设计的概念。

80 年代为 CAE 阶段。与 CAD 相比，除了纯粹的图形绘制功能外，又增加了电路功能设计和结构设计，并且通过电气连接网表将两者结合在一起。这就是计算机辅助工程的概念。CAE 的主要功能是：原理图输入、逻辑仿真、电路综合、电路时延后仿真、自动布局布线及 PCB 后分析等。

90 年代为 ESDA 阶段。尽管 CAD/CAE 技术取得了巨大的成功，但并没有完全把设计人员从繁重的设计工作中解放出来。整个设计过程中自动化和智能化程度还不高、各种 EDA 软件的界面千差万别，学习使用困难，并且互不兼容，直接影响到设计环节间的衔接。基于以上不足，人们开始追求贯彻整个设计过程的自动化，这就是 ESDA，即电子系统设计自动化。

设计一个完善的电子系统是一项非常复杂的任务。在现代电子系统设计领域，EDA 已经

成为重要手段，无论是设计逻辑芯片还是数字系统，由于其复杂程度都在不断增加，仅仅依靠手工进行数字系统设计已经不能满足要求，今后所有设计工作都将借助 EDA 工具进行。EDA 工具的使用，简化了设计的实现过程，使设计人员能够将注意力集中在系统的优化和完善方面，从而设计出性能价格比优越的电子产品。

过去的十多年，出现了许多不同种类的设计工具。而一个工具要被广泛地使用，一般都要支持设计原理图、功能仿真、时序同步分析以及最终的物理电路实现。其中，EDA 工具立下了汗马功劳。从 PCB（Printed Circuit Board）电路板的设计，到可编程逻辑器件 FPGA/CPLD（Field Programmable Gate Array/Complex Programmable Logic Device）的编程，以及各种功能仿真，都离不开不同功能的电子设计软件的应用。

设计电路板时，我们经常使用 Protel、Cadence、Mentor 等工具软件；对 FPGA/CPLD 编程时，离不了 MAX+plus II、Quartus 等工具软件。Synopsys、Cadence 等公司更是提供了功能强大的系统仿真软件和综合器。人们正是通过这些软件，设计出了许多复杂的电子产品。

本书主要针对 FPGA/CPLD，介绍如何应用 VHDL。

1.2 硬件描述语言 HDL

硬件描述语言 HDL（Hardware Description Language）是一种用形式化方法来描述数字电路和设计数字逻辑系统的语言，主要用来描述离散电子系统的结构和行为。硬件描述语言自从 1962 年诞生以来，已逐步发展成为可以用于描述复杂设计的语言。与软件描述语言（Software Description Language）的发展相似，硬件描述语言经历了从机器码（晶体管和焊接）到汇编语言（网表）再到高级语言（硬件描述语言）的一系列过程。

HDL 的设计方式一般有三种：自顶向下设计、自底向上设计和平坦式设计。其中广泛使用的是自顶向下的设计方式。这种设计处理方式要求将设计划分成不同的功能块，每个功能块具有专门定义的输入和输出，并执行专门的逻辑功能。

在一个大型项目（许多设计者或设计小组同时工作）中，采用自顶向下设计方式最为有效。设计的模块化，可以使整个设计、维护和修改变得非常简单。当系统结构和主要模块及其连接确定以后，子项目的工作可以分别独立进行。

另外，HDL 还支持混合层次的描述。所谓混合层次的描述是指结构或网表结构可与行为或算法描述相混合。利用这个功能，使用者可以在抽象的高层次上描述系统结构，然后逐步使设计精练，最后达到详细的元件级或门级实现。当然，也可将 HDL 的设计描述读入编译器（EDA 工具）中，然后利用编译器自动地将该设计综合成门级实现。

进入 80 年代后期，硬件描述语言向着标准化的方向发展。VHDL 和 Verilog HDL 语言是目前业内主要使用的两种 HDL 语言。VHDL 在 1987 年成为 IEEE 标准，Verilog HDL 则在 1995 年才正式成为 IEEE 标准。下面我们来看一下这两种设计语言的异同。

VHDL 和 Verilog HDL 都可用于数字电子系统设计。设计者可以用它们来进行各种级别的逻辑设计，也可以进行数字逻辑系统的仿真验证、时序分析、逻辑综合。从功能上讲，两者是相通的，只是在语法上有些区别。掌握了 VHDL 语言，理解 Verilog HDL 语言也不会十分费力。

VHDL 和 Verilog HDL 作为描述硬件电路设计的语言，其共同的特点是能形式化地抽象表示电路的结构和行为，支持逻辑设计中层次与领域的描述，可借用高级语言的精巧结构来简化电路的描述，具有电路仿真与验证机制以保证设计的正确性，支持电路描述由高层到低层的综合转换，硬件描述与实现工艺无关（有关工艺参数可通过语言提供的属性包括进去），便于文档管理，以及易于理解和设计重用。

在国内一些科研单位和企业的研发中，广泛使用的是 VHDL 语言。因此，本书主要介绍这种硬件描述语言。

1.3 VHDL 简述

作为一门硬件描述语言，VHDL 有它自身的特点。下面简要介绍 VHDL 语言出现的历史背景，VHDL 语言的使用给硬件设计带来的巨大影响，VHDL 语言的基本设计结构。

1.3.1 VHDL 的诞生

1980 年，美国国防部开始实施超高速集成电路 VHSIC（Very High Speed Integrated Circuit）开发项目。在开发进程中，出现了一个越来越明显的需求，就是一个可以描述集成电路的结构和功能的标准语言。因而在此过程中开发了 VHSIC HDL（VHDL），并且成了 IEEE 的标准（IEEE Standard 1076，1987 年批准），它也是美国国防部的标准（MIL-STD-454L）。

编写 VHDL 语言的代码与编写其他计算机程序语言的代码有很大的不同，必须清醒地认识到 VHDL 是硬件编程语言，编写的 VHDL 代码必须能够综合到可用可编程器件实现。如果懂得 EDA 工具中仿真软件和综合软件的大致工作过程，将有助于编写优秀的 VHDL 代码。

VHDL 可以满足设计进程中的多种需求。首先，它允许对设计进行结构化描述，也就是可以将一个设计分成多个不同的功能块及其间的相互连接关系来考虑；其次，它允许使用易读的程序设计语言形式来规定设计功能；再次，它可以在一个设计投入生产以前对其进行仿真，所以用户可以方便地比较不同的设计并测试其正确性，从而省略了生成硬件原形这一耗时耗力的过程。

1.3.2 VHDL 语言特点

VHDL 的设计方法相对于传统的门级设计方法有几个优点。

- VHDL 具有功能强大的语言结构，可用明确的代码描述复杂的控制逻辑设计，并且具有多层次的设计描述功能，支持设计库和可重复使用的元件的生成，是一种设计、仿真和综合的标准硬件描述语言。
- VHDL 语言可读性强，易于修改和发现错误。与大多数高级软件语言一样，HDL 一般都提供很强的类型校验。比如需要 4 bit 宽的信号类型的元件不能与 3 或 5 bit 宽的信号元件相连接。在编译 HDL 描述时，类型不匹配会导致出错。若一个变量的范围定义为 1 到 15，对该变量赋以 0 值将会导致出错。类型使用错误是描述中

的主要错误，类型校验可以检测出 HDL 描述中的这类错误，甚至在设计生成以前就会发现，从而尽可能减少对设计日程计划的影响。

- 可以使用仿真器对 VHDL 源代码进行功能仿真。对于大型设计，采用 VHDL 仿真软件对其进行仿真可以节省时间，在设计的早期阶段就可以检测到设计中的错误，从而予以修正。因为大型设计的综合、优化、配置往往要花费几个小时，综合之前对源代码仿真，可以大大减少设计重复和修正错误的次数和时间。另外，较高层上的设计仿真，只要在门级实现之前，允许对结构和设计进行检验。
- VHDL 允许设计者不依赖于器件。同一个设计描述，可以采用多种不同器件结构来实现其功能。若需对设计进行资源利用和性能方面的优化，也并不是要求设计者非常熟悉器件的结构，从而可以集中精力从事设计构思。
- HDL 描述实现了设计与工艺无关。HDL 描述比网表或原理描述更易读，更易于理解。因为初始 HDL 设计描述是与工艺无关的，在以后的设计中我们可以通过对其重用生成另一不同工艺下的设计，而不必从初始工艺转换过来。
- 可移植性。因为 VHDL 语言是一个标准语言，VHDL 的设计描述可以被不同的 EDA 工具支持，可以从一个仿真工具移植到另一个仿真工具，从一个综合工具移植到另一个综合工具，从一个工作平台移植到另一个工作平台来执行。
- 上市时间快，成本低。VHDL 语言描述快捷，修改方便。可编程逻辑器件的应用则将产品设计的前期风险投资降至最低，而且设计复制速度快，简便易行。VHDL 和可编程逻辑的结合作为一种强有力的设计方式，为设计者的产品上市带来创纪录的速度，同时节省了人力，可以将技术人员从繁琐的电路描述中解放出来。
- ASIC 移植。如果 VHDL 语言设计被综合到 FPGA 中，则可以使设计的产品以最快的速度上市。当产量达到相当数量时，采用 VHDL 很容易转成 ASIC 设计，仅需更换不同的库重新进行综合。另外，由于工艺技术的改进，需要采用更先进的工艺时，仍可采用原来的 VHDL 代码。

1.3.3 VHDL 设计简述

VHDL 描述了数字电路设计的行为、功能、输入以及输出。在语法上，VHDL 与现代编程语言相似，但是它包含了许多与硬件有特殊关系的结构。

VHDL 将实体（元件、电路或系统）分成外部的可见部分（实体名和连接）和内部的隐藏部分（实体算法和实现）。当定义了一个实体的外部接口之后，其他实体可以利用该实体。内部和外部的概念对系统设计的 VHDL 来说是十分重要的。相对于其他实体，定义一个实体要依据其连接和行为。我们可以尝试一个实体的别的实现（结构体），而不用改变设计的其余部分。

当定义了一个设计的实体以后，在其他的设计中如果需要这个实体可以对其进行重用。同样也可以开发一个实体库，用在许多设计或一系列设计中。

VHDL 的硬件模型，如图 1.1 所示。

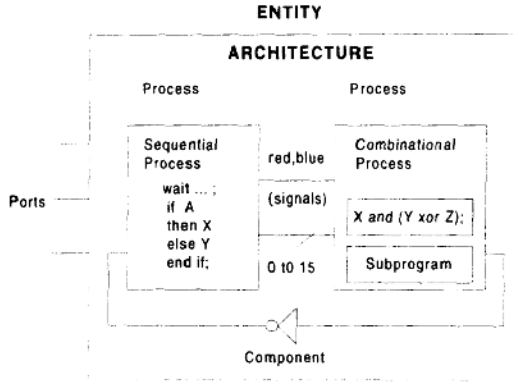

图 1.1 VHDL 硬件模型

一个 VHDL 实体（设计）可以有多个输入和输出，或有双向端口与邻近系统相连。一个实体往往包括相互连接的多个进程和组件，并且都并行运行。

每个实体都可以对应一个或多个结构体，结构体由信号赋值语句、进程语句、组件例化语句等组成。进程中可以定义和调用（实例化）子程序（子模块）。不同的进程之间通过信号交互信息。VHDL 里的进程可以生成由触发器和锁存器组成的时序电路，也可以生成仅由逻辑门组成的组合电路。

信号必须有一个源（驱动），可以有一个或多个宿（接收），并且具有一定的类型，如用户定义的类型“color”、“number between 0 and 15”等。

1.3.4 VHDL 的结构体描述

VHDL 提供了多种结构集。通过 VHDL，用户可以使用不同程度的抽象方式来描述具有不同复杂程度的离散电子系统，如系统级的、板级的、芯片级的以及模块级的。VHDL 语言的结构主要分为三种，即行为(Behavioral)级描述、数据流(Dataflow)级描述和结构(Structural)级描述。

- 行为级描述。
  通过一组串行的 VHDL 进程，反映设计的功能和算法。
- 数据流级描述。
  这种描述将数据看成从设计的输入端流到输出端，对它的操作定义为用并行语句表示的数据形式的改变。
- 结构级描述。
  这是最接近于实际硬件的一种描述。它是将设计看成多个功能块的相互连接，并且主要通过功能块的实例化来表示。

1.4 一个简单实例

这一节，我们来看一个用 VHDL 实现的 2 bit 计数器。通过例 1.1，可以使读者初步感觉一下 VHDL 语言的设计方法。

我们先来定义一个实体及其端口。

【例 1.1】

```
entity count2 is
generic (prop_delay : Time := 10 ns) ;
port (clock : in bit;
q1, q0 : out bit) ;
end count2;
```

这个定义指出实体 count2 有一个输入和两个输出且均为 bit 型，即可以表示 0 或 1 的类型。同时还定义了一个常数属性 prop_delay，利用它来控制实体的操作，本例中它给出了传输延时。在未对其赋值的情况下，取缺省值为 10ns。

实体的实现在结构体中描述，可以存在多个结构体，每个都从不同的方面描述了实体。例 1.2 是一个计数器的行为级描述。

【例 1.2】

```
architecture behaviour of count2 is
begin
    count_up : process (clock)
    variable count_value : natural := 0;
    begin
        if clock = '1' then
            count_value := (count_value + 1) mod 4;
            q0 <= bit'val (count_value mod 2) after prop_delay;
            q1 <= bit'val (count_value / 2) after prop_delay;
        end if;
    end process count_up;
end behaviour;
```

在这个描述中，结构体 behaviour 通过一个名为 count_up 的进程来实现。这个进程有一个敏感信号 clock。进程其实就是一组代码，只要它的敏感信号发生变化，就得到执行。进程还定义了一个状态变量 count_value，保存计数器的当前状态。在仿真的最初阶段，变量初始化为 0。当 clock 输入从 0 变为 1 时，状态变量加 1，同时输出的两个端口反映出状态变量的新值。信号赋值采用了一个常数属性 prop_delay，用它来决定时钟信号变化多久后输出更新。当程序执行到进程体末尾时，进程被悬挂起来，直到 clock 端口信号的又一次变化来触发它。

2 bit 计数器也可描述成包含两个 T 触发器和一个反向器的电路，如图 1.2 所示。

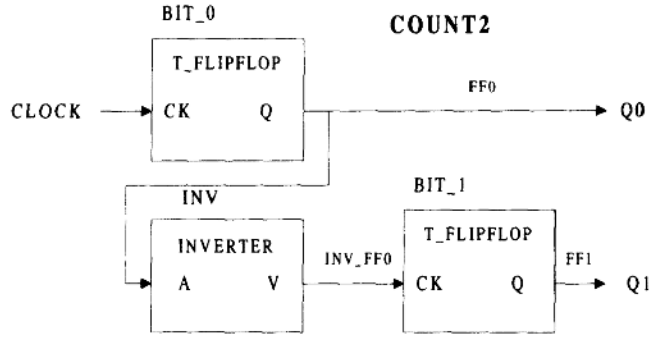

图 1.2 count2 的结构图

这种结构化的描述可用 VHDL 语言表示如下：

【例 1.3】

```
architecture structure of count2 is
component t_flipflop
port (ck : in bit; q : out bit) ;
end component;

component inverter
port (a : in bit; y : out bit) ;
end component;

signal ff0, ff1, inv_ff0 : bit;
begin
    bit_0 : t_flipflop port map (ck => clock, q => ff0) ;
    inv : inverter port map (a => ff0, y => inv_ff0) ;
    bit_1 : t_flipflop port map (ck => inv_ff0, q => ff1) ;
    q0 <= ff0;
    q1 <= ff1;
end structure;
```

在这个结构体中声明了两个组件 t_flipflop 和 inverter，同时声明的还有 3 个内部信号，接着例化了每个组件，并把实例的端口和内部信号映射到实体的端口上。比如，bit_0 是 t_flipflop 的一个实例，它的 ck 端口与实体 count2 的 clock 端口相连，它的端口 q 与内部信号 ff0 相连。最后两个并行赋值语句把内部信号直接引到了外部。

读者通过这个例子可以看到，用 VHDL 语言描述电路并不复杂，特别是在行为级描述时，可以使用一些普通软件编程语言的方法，但同时它也具有硬件方面的一些特性，描述与具体结构息息相关。在接下来的学习中，我们将详细学习一下 VHDL 语言的语法，同时也将逐步接触到多种描述的方法。

1.5 小结

本章简要讲述了 EDA 工具的一般概念及其发展历史，同时也简要介绍了一种广泛使用的硬件描述语言 VHDL 的诞生、发展和特点，最后给出了用 VHDL 语言实现且采用两种不同的描述方式的同一个设计实例。所有这些，都为读者后面的学习打下了基础。

第 2 章　VHDL 语言基础

在本章及第 3 章中，我们将详细介绍 VHDL 语言的语法基础。本章介绍 VHDL 语言的基础特性，包括程序基本结构、VHDL 中的数据对象与数据类型以及表达式；第 3 章介绍 VHDL 语言的基本描述语句，包括并行语句与串行语句两部分；在第 4 章介绍用 VHDL 语言进行设计的一些实例。这三章的内容是本书的基础，需要读者认真学习，并在学习中不断实践练习。

这里需要提醒读者的是，VHDL 作为一种程序设计语言，与软件编程语言有非常相似的地方，读者在学习中会发现 VHDL 有许多与 C 语言或 PASCAL 语言相似的表达式、运算操作符甚至程序结构。同样，VHDL 语言里也有子程序、函数、过程等类似软件语言的概念，这些都为学习 VHDL 语言提供了有利的条件。但同样值得注意的是，VHDL 语言是硬件设计描述语言，因此，并不像一般软件设计语言，经编译后由 CPU 执行，而是要经编译，综合成可实现的硬件结构，下载到 FPGA 芯片中或制成 ASIC 器件，即最终转化为硬件实现。因此我们在编写程序的过程中，就得时刻注意语句的硬件实现方式。

2.1 VHDL 程序设计基本结构

VHDL 语言是硬件设计描述语言，用它既可以设计基本的门电路，也可以设计数字电子系统。在 VHDL 中，对某个数字系统的硬件抽象称为实体（Entity）。实体既可以单独存在，也可以作为另一个更大实体的一部分。当一个实体成为另一个实体的一部分时，我们就把这个实体称为组件（Component）。因此，组件跟实体并没有本质上的区别，只是在系统中的具体层次不同。

描述一个实体的对外特性及其内部功能，是设计的主要任务。具体描述一个实体，或者说一个 VHDL 程序设计的基本结构，主要包括 4 个方面。

- 实体（Entity）声明。
- 结构体（Architecture）。
- 配置（Configuration）声明。
- 程序包（Package）。

接下来我们将对这 4 个方面进行详细介绍。

2.1.1 实体声明

实体声明定义了一个设计模块的输入和输出端口，即模块对外的特性。也就是说，实体声明给出了设计模块与外部的接口，如果是顶层模块的话，就给出了芯片外部引脚定义。

个设计可以包括多个的实体，处于最高层的实体模块称为顶层模块，而处于底层的各个实体，都将作为一个个组件，例化到高一层的实体中去。具体的例化过程，我们将在以后的章节中讨论。

实体声明语法如下：

```
entity 实体名称 is [generic（类属声明）;]
        [port（端口声明）;]
end [实体名称];
```

由实体声明语法结构可知，实体声明主要包括端口声明和类属声明两个方面。注意，在实体声明中不能使用类属声明和端口声明以外的任何声明。实体声明，作为一个设计的对外特性的具体描述，提供了与其他设计（实体或组件）的接口。所有这些功能是通过定义实体的特征（类属和端口）来完成的。

2.1.1.1 端口声明

端口声明确定了输入和输出端口的数目和类型。

语法如下：

```
port（
 端口名称:端口方式 端口类型
 {;端口名称:端口方式 端口类型}
 ）;
```

其中，端口方式可以是下面 4 种方式：

1. in 输入型，表示这一端口为只读型。
2. out 输出型，表示只能在实体内部对其赋值。
3. inout 输入输出型，既可读也可赋值。可读的值是该端口的输入值，而不是内部赋给端口的值。
4. buffer 缓冲型，与 out 相似但可读。读的值即内部赋的值。它只能有一个驱动的源。

端口类型是预先定义好的数据类型，关于数据类型，我们将在 2.2.3 节中详细叙述。

例 2.1 给出了一个与非门的实体声明，在使用它与其他实体连接之前，必须指出该实体输入和输出端口的数目和类型。

【例 2.1】

```
entity NAND2 is
port (A, B : in BIT;          --两个输入 A 和 B
     Z : out BIT              --一个输出 Z = (A and B)
     ) ;
end NAND2;
```

注意，VHDL 语言中用两个或两个以上连写的“-”符号表示其后面的内容为注释，如“--”。