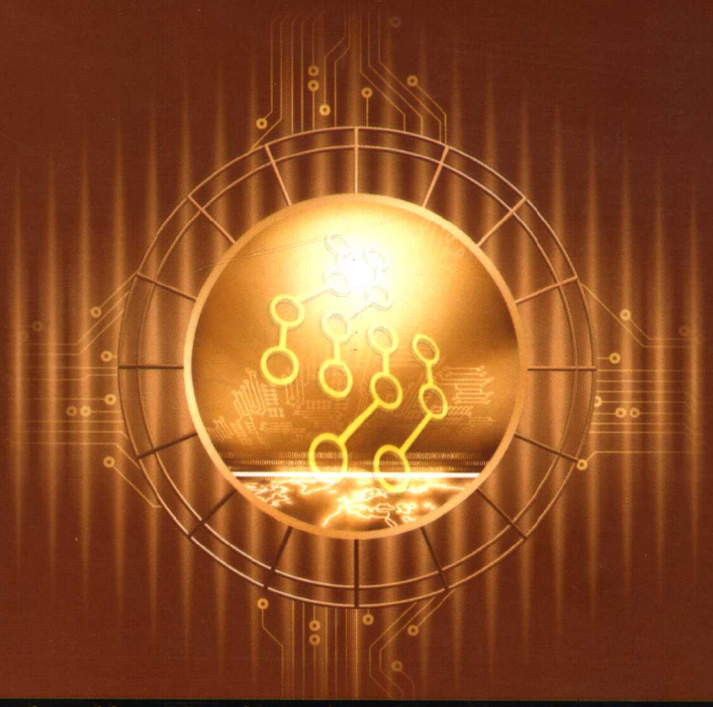

21世纪高等院校电气信息类系列教材

VHDL数字电路设计与应用实践教程

第2版

王振红　主编

机械工业出版社
CHINA MACHINE PRESS

21世纪高等院校电气信息类系列教材

VHDL数字电路设计与应用实践教程

第2版

王振红　主编

机械工业出版社

本书共分 8 章。第 1 章介绍了 VHDL 语言中常用的数据、运算符、顺序描述语句和并行描述语句、子程序等基本概念和应用。第 2 章介绍了 MAX+plus II 软件应用方法。第 3～7 章内容包括门电路、组合逻辑电路、触发器、时序逻辑电路及存储器，针对其中的各种功能芯片以及一些例题，讲解了基于 VHDL 及可编程逻辑器件的实现方法。第 8 章为数字电路系统设计课题。

本书可作为高等院校电类专业学生学习 VHDL 及可编程逻辑器件的实训教材，也可供有关工程技术人员参考使用。

图书在版编目（CIP）数据

VHDL 数字电路设计与应用实践教程/王振红主编. －2 版. －北京：机械工业出版社，2005.11

（21 世纪高等院校电气信息类系列教材）

ISBN 7-111-12115-5

Ⅰ. V… Ⅱ. 王… Ⅲ. 硬件描述语言，VHDL－程序设计－高等学校－教材 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2005）第 131934 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）

策　　划：胡毓坚

责任编辑：时　静

责任印制：陶　湛

北京诚信伟业印刷有限公司印刷

2006 年 1 月第 2 版 · 第 1 次印刷

787mm×1092mm　1/16 · 17.5 印张 · 431 千字

5001－10000 册

定价：25.00 元

出 版 说 明

随着科学技术的不断进步，整个国家自动化水平和信息化水平的长足发展，社会对电气信息类人才的需求日益迫切、要求也更加严格。在教育部颁布的“普通高等学校本科专业目录”中，电气信息类（Electrical and Information Science and Technology）包括电气工程及其自动化、自动化、电子信息工程、通信工程、计算机科学与技术、电子科学与技术、生物医学工程等子专业。这些子专业的人才培养对社会需求、经济发展都有着非常重要的意义。

在电气信息类专业及学科迅速发展的同时，也给高等教育工作带来了许多新课题和新任务。在此情况下，只有将新知识、新技术、新领域逐渐融合到教学、实践环节中去，才能培养出优秀的科技人才。为了配合高等院校教学的需要，机械工业出版社组织了这套“21 世纪高等院校电气信息类系列教材”。

本套教材是在对电气信息类专业教育情况和教材情况调研与分析的基础上组织编写的，期间，与高等院校相关课程的主讲教师进行了广泛的交流和探讨，旨在构建体系完善、内容全面新颖、适合教学的专业教材。

本套教材涵盖多层面专业课程，定位准确，注重理论与实践、教学与教辅的结合，在语言描述上力求准确、清晰，适合各高等院校电气信息类专业学生使用。

机械工业出版社

前　言

现在，电子技术的发展非常迅猛，高新技术日新月异。特别是专用集成电路（ASIC）设计技术的日趋进步和完善，推动了数字电路系统设计方法的发展，使它从单纯的 ASIC 设计走向了系统设计和单片系统设计。传统的电子技术设计方法，是从单元电路入手到整体电路的设计、“固定功能集成电路+连线”的设计。这种自下而上的设计，已不能够满足市场的需要。根据系统的功能和行为要求，利用计算机辅助设计自上而下地逐层完成相应的描述，并与大规模可编程器件相结合，使设计出的电路系统速度更快、体积更小、重量更轻、功耗更小、稳定性更高，从而大大提高了产品的竞争能力。

电子设计自动化（EDA）工具给电子设计带来了巨大变革，特别是硬件描述语言的出现和发展，解决了用传统的电路原理设计大系统工程时的诸多不便，成为电子电路设计人员的最得力的助手。其实，早在 20 世纪 80 年代后期，各个 ASIC 研制和生产厂商为了缩短产品开发周期，提高产品在市场上的竞争力，就相继开发了用于各自目的的硬件描述语言，如 ABEL、AHDL 等。但是由于没有统一的标准，这些语言的普及受到了限制。1987 年 12 月，IEEE 对美国国防部开发的超高速集成电路硬件描述语言（Very High Speed Integrate Circuit Hardware Description Language，VHDL）进行了标准化的工作，得到广大用户的一致欢迎。自此以后，VHDL 成了数字电路系统设计的“世界语”。各个 CAD 厂商都努力使自己的电子设计软件与 VHDL 兼容，各高等院校纷纷开设了 VHDL 设计课程，国内也有越来越多的设计人员开始学习和使用 VHDL 进行电路系统的设计。

近年来，可编程逻辑器件的开发生产和销售规模以惊人的速度增长。发展集成电路事业已成为我国在新世纪的重要发展目标。编写本书的目的，是通过大量的设计实例，由浅入深、由简到繁地宣传和推广 VHDL，以提高电子设计领域人员的设计能力。

本书第 1 章介绍了 VHDL 语言的基础知识，第 2 章介绍了 MAX+plus II 软件应用方法。第 3～7 章是基于 VHDL 的功能芯片设计，第 8 章是数字电路系统设计课题。所用的设计手段，包括 ALTERA 公司的 MAX+plus II 软件工具、VHDL 编程语言、EPM7128SLC84-15 和 EPF10K10LC84-4 可编程器件。本次再版对其中的 VHDL 编程语言和 MAX+plus II 软件工具做了较详细的介绍，并又补充了一些应用实例。目的是让此书自成体系，成为 VHDL 电子电路设计实用教材。

本书所例举的课题，有些是北方工业大学参加全国大学生电子竞赛训练的题目，有些是毕业设计或课程设计的题目。每个题目从编程、编译、仿真、布局布线和适配，直至配置/下载和硬件测试，都运用了 VHDL 设计方法。

参加本书程序调试的有王仲、赵新建、郑静静、郝晨羲、孙可佩。北方工业大学信息工程学院院长张常年教授担任本书的主审，在认真审阅的同时提出了许多宝贵意见。赵红怡、曹淑琴、刘红、范锦宏、王玉花、张东彦、康晓麓、赵徐森、刘淑敏、周燕平、吴晓林等对本书的编写工作给予了很多关心和支持。在此对他们表示衷心的感谢。

由于作者水平有限，书中难免存在错误和不妥之处，敬请读者批评指正。读者的反馈信息可通过电子邮件发送至：wzh_writer@sohu.com。

作者

目　录

第1章　VHDL语言的基础知识

VHDL即Very High Speed Integrated Circuit Hardware Description Language，是符合美国电气和电子工程师协会标准（IEEE 标准 1076）的超高速集成电路硬件描述语言，它可以用一种形式化的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（例如下一章所介绍的MAXPLUS II），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，在当今高速发展的信息时代，可以更好地把握商机。

1.1　VHDL编程思想

VHDL 适应实际电路系统的工作方式，以并行和顺序的多种语句方式来描述在同一时刻中所有可能发生的事件。因此可以认为，VHDL 具有描述由相关和不相关的多维时空组合的复合体系统的功能。因此，要求系统设计人员摆脱一维的思维模式，以多维并发的思路来完成VHDL的程序设计。

一个成功的 VHDL 工程设计，其评判的标准包括：是否完成功能要求、满足速度要求、并考虑其可靠性以及资源的占用情况。在具体的工程设计中，必须清楚软件程序和硬件构成之间的联系，在考虑语句能够实现的功能的同时，要考虑实现这些功能可能付出的硬件代价。在编程过程中某个不恰当的语句、算法或可省去的操作都可能带来硬件资源的浪费，因此，在保证完成功能的条件下，应该合理而有效地利用 VHDL 语言所提供的各种语法条件，尽量地优化算法，从而节约硬件资源。

1.2　VHDL语言程序的基本结构

先看下面的一个关于D触发器设计的完整程序，电路如图1-1所示。

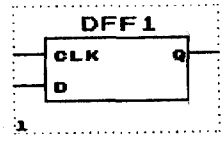

图1-1　D触发器

【程序1.1】

```
library ieee;
use ieee.std_logic_1164.all;                --库说明
entity dff1 is
```

```
    port(clk,d:in std_logic;
         q:out std_logic);
end dff1;                                    --实体说明
architecture rtl of dff1 is
begin
    process(clk)
    begin
         if(clk'event and clk='1')then
           q<=d;
         end if;
    end process;
end rtl;                                     --结构体说明
```

从程序 1.1 的描述层次上可以看出，一个完整的 VHDL 语言程序通常包括库说明、实体说明、结构体说明 3 个部分，下面具体介绍这 3 个部分的语法。

1.2.1 库说明

在程序 1.1 中，库说明语句：

```
library ieee;
use ieee.std_logic_1164.all;
```

用到了 ieee 库以及 ieee 库中的 std_logic_1164 程序包的全部资源。

库说明语句的一般语法如下：

```
library 库名;
use 库名.程序包名.项目名;
```

库是用 VHDL 语言编写的源程序及其通过编译的数据的集合，它由各种程序包组成，程序包提供了各种数据类型、函数的定义以及各种类型转换函数及运算等，以供给设计者使用。VHDL 提供 5 个库，分别为 IEEE 库、STD 库、VITAL 库、自定义库和 WORK 库。

STD 库、WORK 库属预定义库，其他类型的库统称资源库。预定义库的特点是：描述时无需用库语句 library 指定库名，只要求 use 语句指定所使用库中的程序包名；资源库则要用库语句指定库名。

（1）IEEE 库。IEEE 库是按国际 IEEE 组织制定的工业标准进行编写的标准资源库，库的内容很丰富，是最常用的资源库，其中常用的程序包有：

std_logic_1164 程序包：常用数据类型（其中有 std_logic 及 std_logic_vector 数据类型）和函数的定义、各种类型转换函数及逻辑运算。

std_logic_arith 程序包：它在 std_logic_1164 程序包的基础上定义了无符号数 unsigned、有符号数 signed 数据类型并为其定义了相应的算术运算、比较，无符号数 unsigned、有符号数 signed 及整数 integer 之间转换函数。

std_logic_unsigned 和 std_logic_signed 程序包：

这些程序包定义了可用于 integer 数据类型和 std_logic 及 std_logic_vector 数据类型混合

运算的运算符，并定义了一个由 std_logic_vector 型到 integer 型的转换函数。其中，std_logic_signed 中定义的运算符是有符号数运算符。

一般基于 FPGA/CPLD 的开发，IEEE 库中的 4 个程序包 std_logic_1164、std_logic_arith、std_logic_signed 和 std_logic_unsigned 已足够使用。

（2）STD 库。STD 库是标准库，主要包含两个程序包：

standard 标准程序包：定义了基本数据类型、子类型和函数以及各种类型的转换函数等。实际应用中已隐性地打开，不需用 use 语句另作说明。

textio 文本程序包：定义了支持文本文件操作的许多类型和子程序等。在使用 textio 程序包之前，需要先写上 use 语句 use std.textio.all。

（3）VITAL 库。使用 VITAL 库可以提高门级时序仿真的精度，一般在 VHDL 语言程序进行时序仿真时使用。主要包含两个程序包：

VITAL_timing 程序包：时序仿真程序包。

VITAL_primitives 程序包：基本单元程序包。

（4）WORK 库。WORK 库是现行的工作库，设计人员设计的 VHDL 语言程序的编译结果不需要任何说明，都将存放在 WORK 库中。WORK 库可以是设计者个人使用，也可提供给设计组多人使用。

（5）自定义库。设计者自己定义的库。

库说明语句的作用范围是从一个实体说明开始到它所属的结构体为止。

1.2.2 实体说明

实体的电路意义相当于器件，在电路原理图上相当于元件符号，它是一个完整的、独立的语言模块，并给出了设计模块和外部的接口。

实体说明语句的语法如下：

```
entity  实体名  is
    port（端口名称 1：端口方式 1  端口类型 1；
          端口名称 2：端口方式 2  端口类型 2;.....）;
end  实体名;
```

在程序 1.1 中实体说明语句为

```
entity dff1 is
    port(clk,d:in std_logic;
         q:out std_logic);
end dff1;
```

实体名为 dff1。在程序设计过程中，要求实体名与存储的文件名一致。实体名为 dff1，则存储的文件名为 dff1.vhd。其端口名称是这个设计模块与外部接口的桥梁，共有 3 个端口，其名称分别为 d、clk、q。d 和 clk 的端口方式是输入，q 是输出。d、clk、q 的端口类型都是 std_logic 数据类型。

端口方式有 5 种，如表 1-1 所示。

表 1-1 端口方式

in	输入型	信号从该端口进入实体
out	输出型	信号从实体内部经该端口输出
inout	输入输出型	信号既可从该端口输入也可输出
buffer	缓冲型	与 out 类似但在结构体内部可作反馈
linkage		无指定方向，可与任何方向的信号连接

端口类型是预先定义好的数据类型，关于这个内容，在 1.3 中有详细介绍。

1.2.3 结构体说明

结构体是整个 VHDL 语言中至关重要的一个组成部分，这个部分会给出模块的具体实现，指定输入与输出之间的行为。结构体说明语句的语法如下：

```
architecture 结构体名称 of 实体名 is
        结构体说明部分;
begin
        结构体并行语句部分;
end 结构体名称;
```

结构体名称：本结构体的命名。通常根据结构体描述方式命名，把结构体名称命名为 behavioral（行为）、dataflow（数据流）、structural（结构）。

实体名：实体的命名，也就是所存储文件的命名。

结构体说明部分：对结构体内部所使用的信号、常数、数据类型和函数进行定义。

结构体并行语句部分：具体地确定各个输入、输出之间的关系，描述了结构体的行为，是一组并行处理语句，也就是说结构体中的语句的执行是不以书写语句顺序为准的。

结构体对实体的输入输出关系可以用 3 种方式进行描述，即行为描述（基本设计单元的数学模型描述）、寄存器传输描述（数据流描述）和结构描述（逻辑元器件连接描述）。不同的描述方式，只体现在描述语句上，而结构体的框架是完全一样的。

在程序 1.1 中结构体名 rtl，实体名 dff1，结构体并行语句部分从 process 开始，到 end process 为止。结构体语句部分有一个进程语句，进程语句描述了当时钟 clk 上升沿到时，将输入 d 信号赋给输出 q，否则 q 不变。程序 1.1 产生了图 1-1 D 触发器电路符号。

1.3 VHDL 语言中的数据

VHDL 语言和其他软件编程语言一样，也有严格的标识符、数据对象、数据类型定义。准确、熟练掌握基本的数据定义，对初学者是非常必要的。

1.3.1 基本标识符

基本标识符有："A" 到 "Z"，"a" 到 "z"，"0" 到 "9" 以及下划线 "_"。VHDL 不区分大小写。标识符必须以字母开头，不能以下划线为结尾，不能出现连续的两个或多个下划线。以下是一些有效的基本标识符：

DRIVE_BUS、addr_bus、decoder_38、RAM18

1.3.2 数据对象

数据对象也可认为是数值的载体，共有 3 种形式的对象：常量（constant）、变量（variable）、信号（signal）。

（1）常量是设计者给某一常数名赋予固定值的量，一旦赋值就不会发生变化。一般格式为

constant 常数名：数据类型:=表达式；

常量声明的例子如下：

```
constant width:integer:=8;          --常数名 width，数据类型是 integer（整数），赋初始值为 8
constant VCC:real:=3.3;             --常数名 VCC，数据类型是 real（实数），赋初始值为 3.3
```

（2）变量是可以改变值的量，可以在进程和子程序中说明，可以是任意数据类型。变量的赋值是立即生效。一般格式为

variable 变量名：数据类型 := 初始值或表达式；

对变量赋值用“:=”表示。变量声明的示例如下：

```
variable temp:std_logic:='0';       --变量名 temp，数据类型是 std_iogic（标准逻辑位），赋初始值 0
variable a,b:bit_vector(0 to 7);    --变量名 a,b，数据类型是 bit_vector(0 to 7)（位矢量）
b:="10101010";                      --位矢量赋值
a(3 to 6):=('1','1','0','1');       --段赋值
a(0 to 5):=b(2 to 7);
a(7):='0';                          --位赋值
```

（3）信号是电子电路内部硬件连接的抽象，可以将结构体中分离的并行语句连接起来，并且能通过端口与其他的模块连接。信号可以随着时间改变值，不像变量赋值那样立即生效，允许产生延时。信号通常在实体、结构体和程序包中说明，但不能在进程中说明，只能在进程中使用。

对信号赋值可以使用“<=”，允许产生延时（一般用δ表示延时间），这和实际元件的传输延时特性吻合。

一般格式为

signal 信号名：数据类型 ：=初始值；

信号声明的示例如下：

signal a: bit:='0'; --信号名 a，数据类型是位，赋初始值为 0。赋值符“:=”给信号赋初值时会立即生效，而不产生延迟。但信号一般用代入符“<=”赋值，会产生一定延迟才生效。信号可以赋初值也可以不赋初值，没有赋初值，则取默认值，即指定数据类型的最左值

```
signal INIT:bit_vector(7 downto 0);     --定义信号 INIT 是位矢量
signal c:integer range 0 to 15;         --定义信号 c 的数据类型是整数，整数范围 0～15
signal y,x: real;                       --定义信号 y、x 数据类型是实数
y<=x;                                   --经过δ延时后将 x 值赋给 y
```

1.3.3 数据类型

1．标准数据类型

标准数据类型共有 10 种，见表 1-2。

表 1-2 标准数据类型

数据类型	含义	备注	例子
整数	整数$-(2^{31}-1)$～$+(2^{31}-1)$	integer	+136，−457
实数	实数-10^{38}～$+10^{38}$	real 一定有小数点	−1.0，+2.5e23
位	逻辑 0 或 1	bit	'1'，'0'
位矢量	位矢量	bit_vector 双引号括起来的一组数	"00101"
布尔量	逻辑假或真	boolean 只有真（true）和假（false）	
字符	ASCII 字符	character 用单引号括起来	'a','b','1'
时间	整数和时间单位	time fs,ps,ns,us,ms,sec,min,hr	20us、32ns
错误等级	VHDL 程序在编译、仿真、综合过程的工作状态	severitylevel note,warning,error,failure	note,warning 可以忽略，error,failure 不可以忽略
自然数，正整数	整数的子集	natural,positive	
字符串	字符矢量	string 双引号括起来的字符序列	"START"

2．用户自定义的数据类型

VHDL 允许用户自定义数据类型，一般书写格式为

type 数据类型名 is 数据类型定义；

VHDL 常用的用户自定义类型包括枚举类型，整数类型，数组类型，子类型等。

（1）枚举类型。把数据类型中的各个元素都列举出来，方便、直观，提高了程序可阅读性。书写格式为

type 数据类型名称 is（元素 1，元素 2，……）；

其中，数据类型名称和元素都是一个标识符。例如：

```
type color is (blue,green,yellow,red);
```

数据类型名称是 color，（元素 1，元素 2，……）是（blue,green,yellow,red）。枚举类型中所列举的元素在程序编译过程通常是自动编码，编码顺序是默认的，左边第一个元素编码为 0，以后的依次加 1。编码过程中自动将每一个元素转变成位矢量，位矢量的长度由所列举元素个数决定。如上例的 4 个元素，位矢量的长度为 2，编码默认值为：blue="00"； green="01"；yellow="10"； red="11"。在信号定义时就可以使用这种数据类型，例如：

```
type color is (blue,green,yellow,red);
signal p: color;          --信号 p 是 color 数据类型
```

（2）整数类型，实数类型。自定义的整数类型、实数类型是标准数据类型的整数、实

数的子类型，是根据特殊需要自定义的数据类型，以便降低逻辑综合的复杂性和提高芯片资源的利用率。书写格式为

type 数据类型名称 is integer range 整数范围;
type 数据类型名称 is real range 实数范围;

例如：

type percent is integer range –100 to 100;
type current is real range –1.5 to 3.0;

（3）数组（array）类型。数组是将相同类型的数据即数组元素集合在一起所形成的一个新的数据类型。数组类型分限定数组和非限定数组两种，书写格式为

type 数组类型名 is array 范围 of 数组元素的数据类型;
type 数组类型名 is array (range <>) of 数组元素的数据类型;

其中范围是用整数指明数组的上下界，是一个限定数组，例如：

type stb is array (7 downto 0) of bit;
variable addend: stb; --变量 addend 定义为 stb 数组

stb 是数组类型名。(7 downto 0)是数组的上下界，数组有 8 个元素，数组的下标排序是 7、6、5、4、3、2、1、0，各元素的排序是 stb(7)、stb(6)、stb(5)、stb(4)、stb(3)、stb(2)、stb(1)、stb(0)，每一个元素的数据类型是 bit。范围由“range <>”指定，这是一个没有范围限制的数组，是一个非限定数组。在这种情况下，具体范围由信号说明语句来确定。例如：

type bit_vector is array(integer range <>)of bit;-- bit_vector 是一个非限定数组，每一个元素的数据类型是 bit
variable my_vector:bit_vector(5 downt to –5);-- 变量 my_vector 定义为 bit_vector 数组，数组的下标排序是 5、4、3、2、1、0、-1、-2、-3、-4、-5

（4）子类型。子类型说明语句就是对已经存在的基本数据类型作一些范围限制便形成了一种新的数据类型。子类型 subtype 的语句格式如下所示：

subtype 子类型名 is 基本数据类型 range 约束范围

用子类型说明语句有利于提高综合的优化效率，这是因为综合器可以根据子类型的约束范围，有效地推知参与综合的寄存器的最合适的数目，节省资源。例如：

type nat is integer range 0 to 999; --自定义整数类型 nat 是整数，范围 0～999
subtype a_nat is nat ragne 0 to 255;--子类型 a_nat 是 nat 类型，范围 0～255

3. IEEE 标准数据类型“std-logic”和“std-logic-vector”

在 IEEE 库的程序包 std_logic_1164 中定义了两个非常重要的数据类型,即标准逻辑位 std-logic 数据类型和标准逻辑矢量 std-logic-vector 数据类型。

std-logic 定义了 9 种不同的值，增加了不定状态“X”、高阻状态“Z”。不定状态方便了系统仿真，高阻状态方便了双向总线的描述。