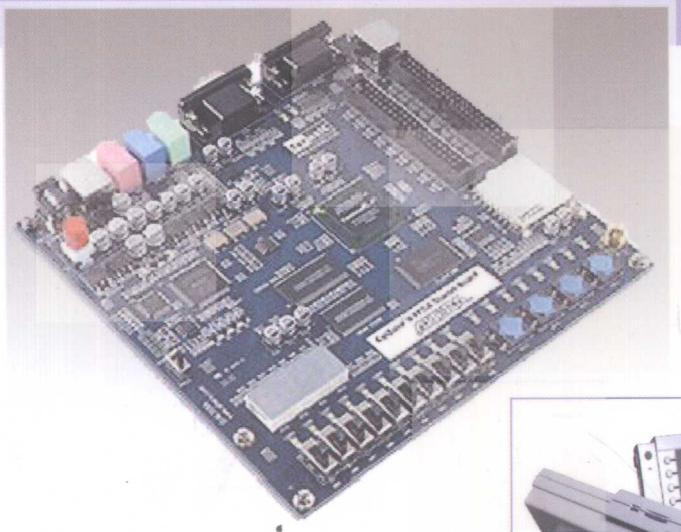
应用型特色

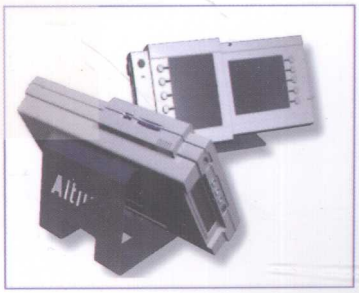
高等学校应用型特色规划教材

北京市精品课程

FPGA开发与应用

FPGA KAIFA YU YINGYONG





王振红 主编

免费赠送电子课件

- 易学、易懂、易上手，覆盖了模拟电子技术基础、数字电子技术基础、FPGA基本知识，采用了大量综合性电子电路小系统设计实例。
- 从兴趣到提高再到创新，不断循环往复，使学生的实践创新能力不断得到提高。
- 设计实例由浅入深，经过实验检验，可以作为电子设计竞赛赛前训练题目，也可以作为电子电路课程设计参考题目。

清华大学出版社

高等学校应用型特色规划教材

FPGA 开发与应用

王振红　主编

清华大学出版社
北　京

内 容 简 介

本书第 1 章～第 5 章介绍了 FPGA 及其硬件描述语言 VHDL 的特点，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、时钟信号描述、状态机等基本概念和应用。第 6 章介绍了 MAX+plus II 软件应用方法。第 7 章与清华大学阎石主编的《数字电子技术基础》(第 4 版)同步，为 FPGA 数字电路设计实例，针对门电路、组合逻辑电路、触发器、时序逻辑电路及存储器等各种功能芯片以及一些例题，讲解了基于 VHDL 及 FPGA 的实现方法。第 8 章介绍了 FPGA 应用系统设计实例，设计实例由浅入深，并配有相关的图及注释。这些设计实例可以作为电子设计竞赛的赛前训练题目，也可以作为电子电路课程设计的参考题目。

本书可作为大专院校电类学生学习 VHDL 及 FPGA 的实训教科书，也可供有关工程技术人员参考使用。

图书在版编目(CIP)数据

FPGA 开发与应用/王振红主编. --北京：清华大学出版社，2010.9
(高等学校应用型特色规划教材)
ISBN 978-7-302-23656-6

Ⅰ. ①F…　Ⅱ. ①王…　Ⅲ. ①可编程序逻辑器件—系统开发　Ⅳ. ①TP332.1

中国版本图书馆 CIP 数据核字(2010)第 160438 号

责任编辑：李春明　郑期彤
装帧设计：杨玉兰
责任校对：李玉萍
责任印制：何　芊
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
社　总　机：010-62770175　　**邮　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：北京四季青印刷厂
装 订 者：三河市新茂装订有限公司
经　　销：全国新华书店
开　　本：185×260　**印　张**：25.75　**字　数**：623 千字
版　　次：2010 年 9 月第 1 版　　**印　　次**：2010 年 9 月第1 次印刷
印　　数：1～4000
定　　价：39.00 元

产品编号：038937-01

前　言

近年来，FPGA 的开发生产和销售规模以惊人的速度增长。发展集成电路事业是我国制定的新世纪的重要发展目标，也是经济全球化新形势下的科技挑战。编写本书的目的是通过大量的设计实例，由浅入深、由简到繁地提高电子设计人员 FPGA 的应用和设计能力。

如何提高学生实践创新能力？首先要使学生对实践感兴趣，在综合性、设计性实验中学习、制作。其次要增加题目数量，做多了，能力自然就会提高。之后，学生应根据生产和生活实际的需要实现创新作品。从兴趣到提高再到创新，不断循环往复，学生的实践创新能力就会不断提高。

本书就是通过电子小系统实践提高学生对实践的兴趣，使学生由单元电路实践顺利过渡到电子小系统实践，减少学生实践的难度。

本书特色体现在：学生易学、易懂、易上手，覆盖了模拟电子技术基础、数字电子技术基础及 FPGA 基本知识，包含了多个综合电子电路小系统设计实例。这些设计实例既有硬件电路，又有软件程序，使用的是 VHDL 和 MAX+plus II 软件。书中列举的每个综合电子电路小系统内容详尽，并且经过实验检验，是北方工业大学参加全国大学生电子设计竞赛的赛前训练题目，也是电子电路课程设计的参考题目。

本书由北方工业大学信息工程学院王振红主编。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。在本书编写过程中，张东彦、宋鹏、曲洪权、王恩成、曹淑琴、周燕平、康晓麓、赵徐森、刘淑敏、吴晓林、韩宇龙、胜智勇等同志给予了很多关心和支持，在此对他们表示衷心的感谢。

由于编者水平有限，如果书中存在错误和不妥之处，敬请读者批评指正。反馈信息可通过电子邮件发送至wzh_writer@sohu.com。

编　著

目　录

第 1 章　FPGA 及其硬件描述语言 VHDL

1.1　FPGA 简介

FPGA(Field Programmable Gate Array，现场可编程门阵列)是在 PAL(Programmable Array Logic，可编程阵列逻辑)、GAL(Generic Array Logic，通用阵列逻辑)、CPLD(Complex Programmable Logic Device，复杂可编程逻辑器件)基础上进一步发展的产物。它具有高集成度，是几万到几百万逻辑门、触发器的集成，便于实现高速的大规模数字电路系统。FPGA 采用 CMOS 工艺，实现了低功耗要求。另外，FPGA 掉电后不能保存数据，因而需要配置 EPROM 芯片，只要将程序存放在 EPROM 中，上电后程序会自动加载到 FPGA 上，因此，FPGA 能够反复使用。FPGA 的编程无须专用的 FPGA 编程器，只需用通用的 EPROM 编程器即可。当需要修改 FPGA 的功能时，只需换一片 EPROM 即可。这样，同一片 FPGA，不同的编程数据，可以产生不同的电路功能。因此，FPGA 的使用非常灵活。

用户可对 FPGA 内部的逻辑模块和 I/O 模块重新配置，以实现用户的逻辑，因而 FPGA 也被用于模拟 CPU。用户可以将 FPGA 的编程数据放在 EPROM 芯片中，也可以在线对 FPGA 进行编程，实现系统在线重构。根据这一特性，用户可以利用 FPGA 构建一个根据工程任务不同而实时定制的 CPU，这是当今研究的热门领域。

1.2　VHDL 程序的特点

FPGA 的硬件描述语言 VHDL(Very High Speed Integrated Circuit Hardware Description Language，超高速集成电路硬件描述语言)符合美国电气和电子工程师协会标准(IEEE 标准 1076)，它使用户能够利用一种和数字电路基本知识结合较密切的语言来描述数字电路和设计数字电路系统。用户可以利用 VHDL 进行分块单元电路设计和整个系统设计，并结合一些先进的 EDA 工具软件(例如 MAX+plus II)，通过计算机将 VHDL 程序下载到硬件芯片上，以实现电路功能，如图 1.1 所示。在当今高速发展的信息时代，这种设计方法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，从而更好地把握商机。

为适应实际数字电路的工作方式，VHDL 以并行和顺序等多种语句方式来描述在同一时刻所有可能发生的事件。因此 VHDL 程序的执行方式与其他语言不同，它不是按顺序执行每一条语句，而是并行执行与顺序执行共存。这要求数字电路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。VHDL 程序的特点如图 1.2 所示，它通常由一组并行语句构成，有些并行语句里又包含了顺序语句。

为完成 VHDL 的程序设计，实现数字电路的功能，工程设计人员要懂得每一种 VHDL

语句格式、内容和各种 VHDL 语句执行的顺序，以及 VHDL 语句中的各种变量、数据类型和表达式。一个成功的 VHDL 程序设计，一是要完成数字电路功能，二是程序要简洁，以便器件工作速度快，占用的资源少。

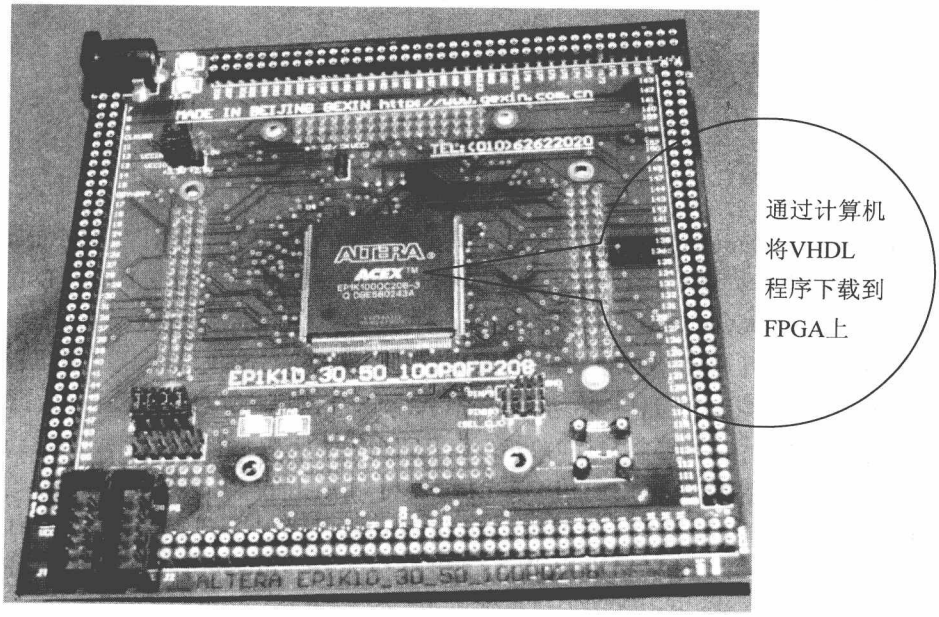


图 1.1　硬件芯片 FPGA

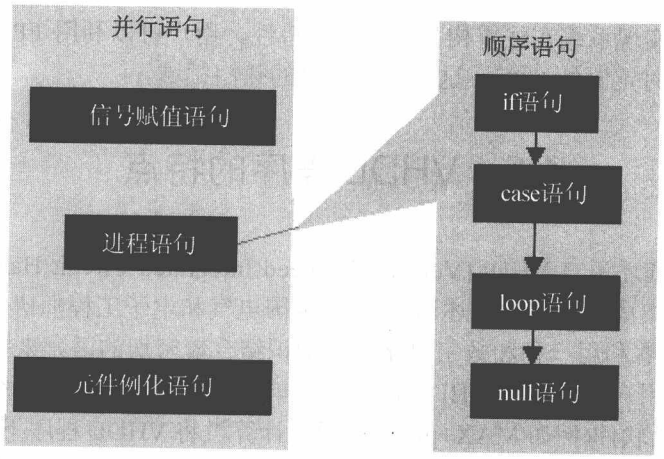


图 1.2　VHDL 程序的特点

1.3　VHDL 程序的基本结构

先看一个关于 D 触发器设计的完整程序。

【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程序通常包括库说明、实体说明和结构体说明3个部分。下面具体介绍这3个部分的语法。

1.3.1 库说明

在程序1.1中，库说明语句为：

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

其中用到了IEEE库以及IEEE库中的std_logic_1164程序包的全部资源。

库说明语句的语法如下：

```
library 库名;
use 库名.程序包名.项目名;
```

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。

库是用VHDL编写的源程序及其通过编译的数据的集合。库是由各种程序包组成的，程序包提供了各种数据类型以及各种类型转换函数及运算等，以供给设计者使用。VHDL提供了IEEE库和其他的库。

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。其中常用的程序包如下。

- std_logic_1164程序包：包括常用数据类型(其中有std_logic及std_logic_vector数据类型)和各种数据类型转换函数及其算术、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)和有符号数(signed)数据类型，并为其定义了相应的算术运算、关系运算，以及无符号数(unsigned)、有符号数(signed)及整数(integer)之间的转换函数。
- 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unsigned、signed与std_logic_vector数据类型之间的转换函数，以及std_logic_vector数据类型与integer数据类型之间的转换函数。其中，

std_logic_signed 中定义的运算符是有符号数运算符。

对于一般的 FPGA/CPLD 的开发，IEEE 库中的这 4 个程序包(std_logic_1164、std_logic_arith、std_logic_signed 和 std_logic_unsigned)已足够使用。

1.3.2 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器件符号(或元器件管脚图、电路图)。程序 1.1 描述的实体的电路图如图 1.3 所示。实体说明是一个完整、独立的语言结构，也称为模块，其中会给出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;
```

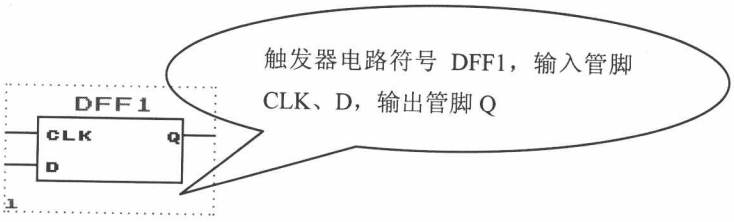


图 1.3 程序 1.1 描述的实体的电路图

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;
```

在程序设计过程中，要求实体名与存储的文件名一致。如程序 1.1 中，实体名为 DFF1，则存储的文件名为 DFF1.VHD。端口名称是设计模块与外部接口的桥梁。程序 1.1 中描述的实体共有 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。

VHDL 中的端口方式共有 5 种，如表 1.1 所示。

表 1.1 端口方式

端口方式	类 型	说 明
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，也可从该端口输出

续表

端口方式	类　型	说　明
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，在 1.4 节中有详细介绍。

1.3.3　结构体说明

结构体是整个 VHDL 中至关重要的一个组成部分，它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;
```

特别提示：结构体内是一组并行语句。

- 结构体名称：本结构体的名称，通常根据结构体的描述方式进行命名。
- 实体名：实体的名称，也就是所存储文件的名称。
- 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。

在程序 1.1 中，结构体名为 rtl；实体名为 dff1；结构体并行语句部分从 process 开始，到 end process 为止。结构体语句部分只有一个进程语句，其描述的结构体行为是：当时钟 clk 上升沿到来时，将输入 D 信号赋给输出 Q，否则 Q 不变。程序 1.1 产生了如图 1.3 所示的 D 触发器电路(或元器件)符号。

1.4　VHDL 的数据

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

1.4.1　基本标识符

VHDL 的基本标识符有：A～Z、a～z、0～9 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
```

1.4.2 数据对象

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

1. 常量

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;
```

对常量赋值用“：=”表示。常量声明的例子如下：

```
constant width:integer:=8; --常数名 width，数据类型 integer，整数赋值 8
constant VCC:real:=3.3;    --常数名 VCC，数据类型 real，实数赋值 3.3
```

2. 变量

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的。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_iogic:='0';     --变量名 temp，数据类型 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 bit_vector，(0 to 7)是位矢量
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

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进行连接。信号的值可以随着时间改变，且不像变量赋值那样立即生效。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。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，赋初值为 0
```

用赋值符“:=”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，产生一定延迟后才生效。信号可以赋初值，也可以不赋初值。没有赋初值时，信号取默认值，即指定数据类型的最左值或最小值。

定义信号及给信号赋值的例子如下：

```
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 数据类型是整数，范围为 0～15
signal y,x: real;  --定义信号 y、x 数据类型是实数
y<=x;                --经过 δ 时间的延时后将 x 值赋给 y
```

特别提示： 变量只能在进程语句中说明和使用；信号、常量可以在结构体和进程语句中使用。变量、常量用“:=” 赋值，无延时；信号用“:=”赋初值时，无延时，用“<=”赋值时，有延时。

1.4.3 数据类型

1. 标准数据类型

标准数据类型共有 10 种，如表 1.2 所示。

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、3e4(3×10^4)
实数	实数-10^{38}～$+10^{38}$	real，一定有小数点	-1.0、+2.5e23($+2.5\times10^{23}$)
位	逻辑 0 或 1	bit，用单引号括起来	‘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、综合过程中的工作状态	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”

特别提示： 常用的数据类型为整数、实数、位、位矢量。整数可以用十进制、二进制、八进制、十六进制表示。例如，十进制 125，用二进制写为 2#11111111#，八进制写为 8#377#，十六进制写为 16#FF#。布尔量用于关系运算，如 $a>b$ 成立，即表示 $a>b$ 的结果为真(true)。

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、实数类型、数组类型、子类型等。

1) 枚举类型

枚举类型是把数据类型中的各个元素都列举出来，其优点为方便、直观，可提高程序的可阅读性。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中通常是自动编码，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，位矢量的长度将由所列举元素的个数决定。如上例 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，其目的是在编译过程中降低逻辑电路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;-- percent 为数据类型名称，它的数据
                                              类型是 integer，范围是-100～100
type current is real range -1.5 to 3.0; -- current 为数据类型名称，它的数据类
                                           型是 real，范围是-1.5～3.0
```

3) 数组类型

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，表示是限定数组。例如：

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

上例中，stb 是数组类型名，(7 downto 0)是数组的上下界。该数组有 8 个元素，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)，各元素的数据类型是 bit。

用“range <>”定义的数组是没有范围限制的数组，即为非限定数组。在这种情况下，数组的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，各元素的数据类型是 bit
variable my_vector:bit_vector(5 downto -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做一些范围限制后形成的一种新的数据类型。子类型(subtype)的语句格式为：

```
subtype 子类型名 is 基本数据类型 range 约束范围;
```

用子类型说明语句的好处在于编译过程中可以根据子类型的约束范围，有效地推知参与的寄存器的最合适的数目，以节省芯片资源。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 range 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

ieee 库的 std_logic_1164 程序包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，其中包括了不定状态‘X’和高阻状态‘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。

std_logic 的 9 种值如下。

- ‘U’——初始值；
- ‘X’——不定，未知；
- ‘0’——0；
- ‘1’——1；
- ‘Z’——高阻；
- ‘W’——弱信号不定，未知；
- ‘L’——弱信号 0；
- ‘H’——弱信号 1；
- ‘-’——不可能情况。