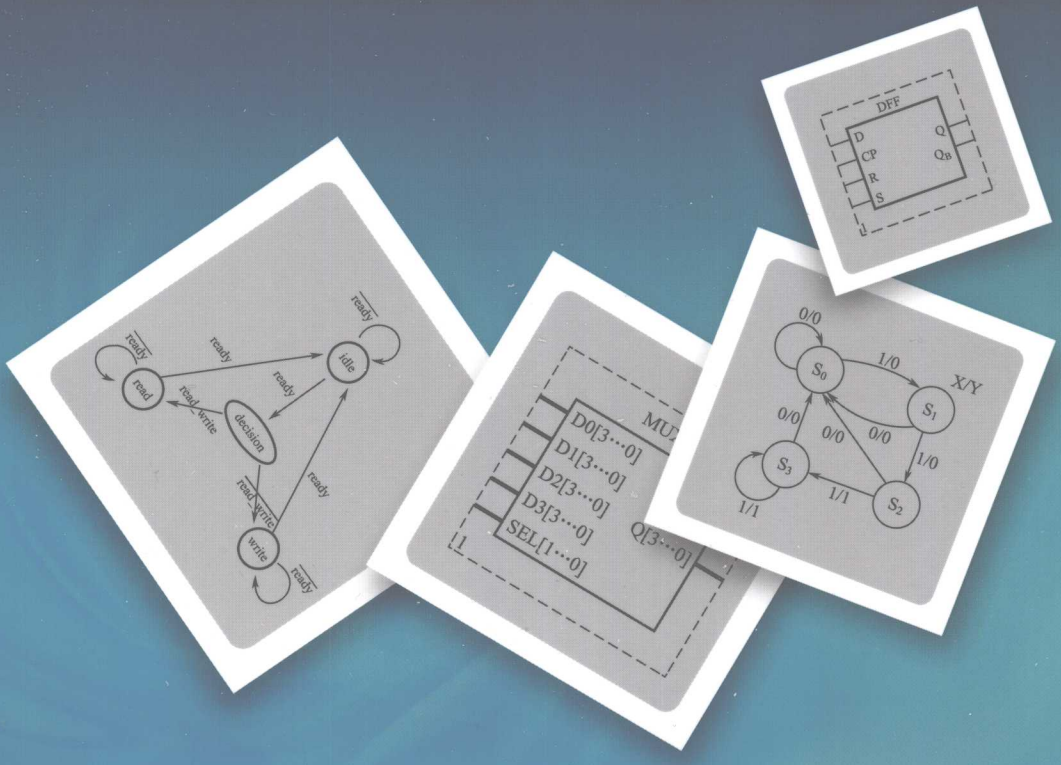
图说VHDL

数字电路设计

王振红　编著

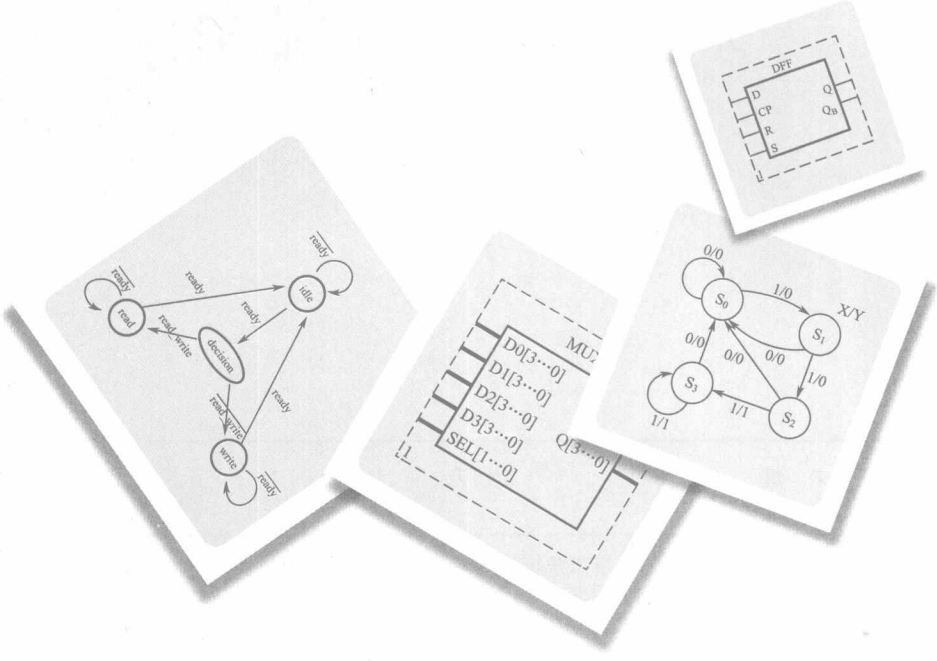


化学工业出版社

图说VHDL

数字电路设计

王振红　编著



化学工业出版社

·北　京·

图书在版编目（CIP）数据

图说VHDL数字电路设计/王振红编著. —北京：化学工业出版社，2008.12
ISBN 978-7-122-03734-3

Ⅰ.图… Ⅱ.王… Ⅲ.①硬件描述语言，VHDL-程序设计-图解②数字电路-电路设计-图解 Ⅳ.TP312-64 TN79-64

中国版本图书馆CIP数据核字（2008）第142513号

责任编辑：宋 辉　　装帧设计：尹琳琳
责任校对：蒋 宇

出版发行：化学工业出版社（北京市东城区青年湖南街13号 邮政编码100011）
印　　装：化学工业出版社印刷厂
720mm×1000mm 1/16 印张9 1/2 字数174千字 2009年1月北京第1版第1次印刷

购书咨询：010-64518888（传真：010-64519686） 售后服务：010-64518899
网　　址：http://www.cip.com.cn
凡购买本书，如有缺损质量问题，本社销售中心负责调换。

定　　价：22.00元

前　言

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的设计和发展，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，即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、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，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、由简到繁地宣传和推广VHDL，以提高电子设计领域人员的设计能力。

本书的43个设计实例由浅入深，且配有图、注释，所有设计实例从编程、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，并且经过实践检验是正确的。

本书由王振红编著，北方工业大学信息工程学院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。张东彦、宋鹏、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。

编著者

2008年6月于北方工业大学

目 录

第1章 VHDL的数据和表达式

- 1.1 VHDL程序的特点
- 1.2 VHDL程序的基本结构
- 1.3 VHDL的数据
- 1.4 VHDL的表达式

VHDL即Very High Speed Integrated Circuit Hardware Description Language（超高速集成电路硬件描述语言），是电子设计的新技术，它符合美国电气和电子工程师协会标准（IEEE标准1076），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。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），通过计算机下载到硬件芯片上，实现电路功能，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。如图1.1所示为硬件芯片可编程器件。

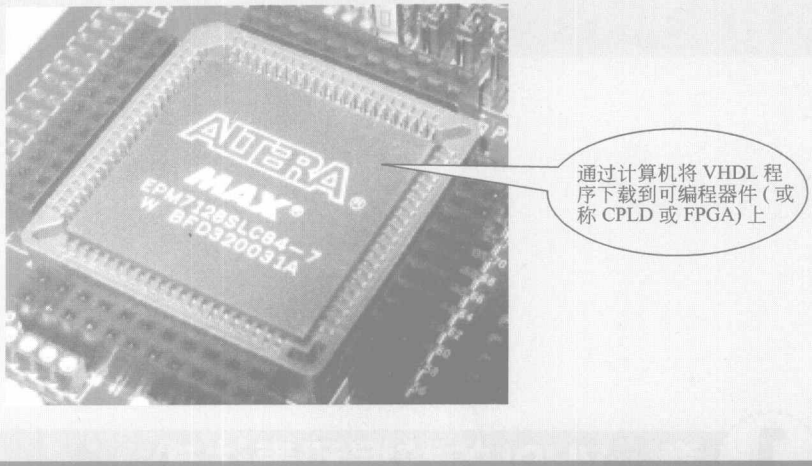


图 1.1 硬件芯片可编程器件

1.1 VHDL 程序的特点

VHDL 程序执行方式与其他语言不同，它不是按顺序一条一条执行每一条语句，而是有并行执行的语句同时也有按顺序执行的语句，来描述在同一时刻中可能发生的事件。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。

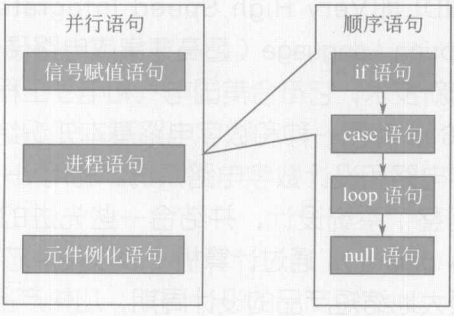


图 1.2 VHDL 程序的特点

为完成VHDL的程序设计，实现数字电路的功能，要求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，二是程序要简捷，以便器件工作速度快、占用器件的资源少。

1.2 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。


```
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、结构体说明三个部分，下面具体介绍这三个部分的语法。

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。

库说明语句一般语法如下：

```
library 库名；
use 库名.程序包名.项目名；
```

库是用VHDL编写的源程序及其通过编译的数据的集合，库由各种程序包组成，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，其中常用的程序包有：

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、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，并定义了由无符号数unsigned、有符号数signed与std_logic_vector数据类型之间的转换函数，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。

用于一般的FPGA/CPLD的开发，IEEE库中的四个程序包std_logic_1164、std_logic_arith、std_logic_signed和std_logic_unsigned已足够使用。

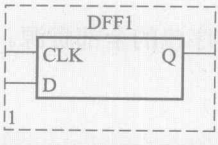
库说明语句作用范围是从一个实体说明开始到它所属的结构体为止。

1.2.2 实体说明

实体的电路图的意义相当于器件，在电路原理图上相当于元器件符号（或元器件管脚图或电路图），程序1.1实体的电路图如图1.3所示。实体说明了器件的输入、输出管脚。它是一个完整的、独立的语言结构，也称为模块，实体给出了设计模块和外部的接口。

实体说明语句语法如下：

```
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；
```

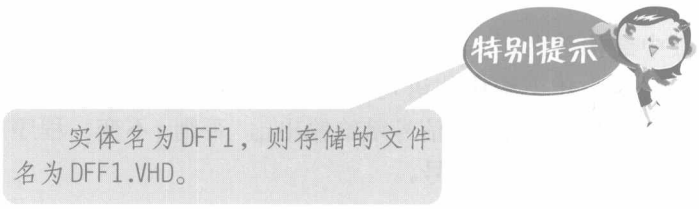


触发器电路符号DFF1，输入管脚CLK，D；输出管脚Q

图1.3 程序1.1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;
```



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，共有三个端口，其名称分别为D，CLK，Q。端口方式D和CLK是输入，Q是输出。端口类型D、CLK、Q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

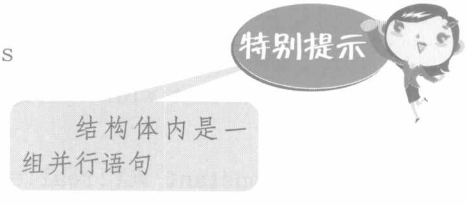
端口类型是预先定义好的数据类型，关于这个内容，在1.3节中有详细介绍。

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自己命名。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。

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。结构体语句部分只有一个进程语句，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1.3 D触发器电路（或元器件）符号。

1.3 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、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 基本标志符

基本标志符有："A"到"Z"，"a"到"z"，"0"到"9"以及下划线"_"，VHDL不区分大小写。标志符必须以字母开头，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志符：

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，共有三种形式的数据对象：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。

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。

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std_logic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。可以随着时间改变值，不像变量赋值立即生效，允许产生延时。信号通常在实体、结构体和程序包中说明，不能在进程中说明，只能在进程中使用。

对信号赋值使用“<=”表示，允许产生延时，这和实际元件的传输延时特性吻合。

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。
```

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或最小值。

```
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。
```

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 <= 赋值，有延时。

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$-(2^{31}-1)$到$+(2^{31}-1)$	integer	+136,-457,3e4(3×10^4)
实数	实数-10^{38}到$+10^{38}$	real 一定有小数点	-1.0，+2.5e23 ($+2.5\times10^{23}$)
位	逻辑0或1	bit 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, μs, ms, s, min, h	20μs、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"

常用的数据类型：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，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。

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。

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志符。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，位矢量的长度将由所列举元素个数决定。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，以便编译过程降低逻辑电路的复杂性和提高芯片资源的利用率。

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 ～ 100;—— percent数据类型名称，它的数据类型是integer整数，range 整数范围是−100 ～ 100。

type current is real range −1.5 ～ 3.0; —— current数据类型名称，它的数据类型是real实数，range 实数范围是 −1.5 ～ 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。

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被定义为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。

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，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 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。

‘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。

1.4 VHDL的表达式

在VHDL中，表达式是通过不同的操作符连接多个操作数来完成算术或逻辑计