VHDL 语言程序

目录（CONTENTS）

一、 基本概念；
二、 VHDL 程序的基本结构；
三、 VHDL 的描述方法；
四、 VHDL 的常用语句；
五、学习 VHDL 时可能遇到的问题。
六、常见词汇中英文对照表。

## 一、基本概念：

1. VHDL 超高速集成电路硬件描述语言。
   V-Very High Speed Integrated Circuit；
   HDL-Hardware Description Language。
2. IEEE（I-triple E）美国电气和电子工程师协会。
3. 仿真与综合：
   HDL 有两种用途：系统仿真和硬件实现。 如果程序只用于仿真，那么几乎所有的语法和编程方法都可以使用。 但如果我们的程序是用于硬件实现（例如：用于 FPGA 设计），那么我们就必须保证程序“可综合”（程序的功能可以用硬件电路实现）。 不可综合的 HDL 语句在软件综合时将被忽略或者报错。 我们应当牢记一点：“所有的 HDL 描述都可以用于仿真，但不是所有的 HDL 描述都能用硬件实现。”
4. HDL 开发流程：
   用 VHDL/VerilogHD 语言开发 PLD/FPGA 的完整流程为：
   1° **文本编辑**：用任何文本编辑器都可以进行，也可以用专用的 HDL 编辑环境。通常 VHDL 文件保存为.vhd 文件，Verilog 文件保存为.v 文件。
   2° **功能仿真**：将文件调入 HDL 仿真软件进行功能仿真，检查逻辑功能是否正确（也叫前仿真，对简单的设计可以跳过这一步，只在布线完成以后，进行时序仿真）
   3° **逻辑综合**：将源文件调入逻辑综合软件进行综合，即把语言综合成最简的布尔表达式和信号的连接关系。逻辑综合软件会生成.edf（edif）的 EDA 工业标准文件。
   4° **布局布线**：将.edf 文件调入 PLD 厂家提供的软件中进行布线，即把设计好的逻辑安放到 PLD /FPGA 内。
   5° **时序仿真**：需要利用在布局布线中获得的精确参数，用仿真软件验证电路的时序，也叫后仿真。
   6° **编程下载**：确认仿真无误后，将文件下载到芯片中。
   通常以上过程可以都在 PLD/FPGA 厂家提供的开发工具（如 MAXPLUSII，Foundation，ISE）中完成，但许多集成的 PLD 开发软件只支持 VHDL/Verilog 的子集，可能造成少数语法不能编译，如果采用专用 HDL 工具分开执行，效果会更好。
5. 逻辑描述层次：
   一般的硬件描述语言可以在三个层次上进行电路描述，其层次由高到低依次可分为行为级、RTL 级和门电路级。VHDL 语言是一种高级描述语言，适用于行为级和 RTL 级的描述，最适于描述电路的行为；Verilog 语言和 ABEL 语言是一种较低级的描述语言，适用于 RTL 级和门电路级的描述，而 ABEL 最适于描述门级电路。

## 二、VHDL 程序的基本结构：

一段完整的 VHDL 程序包括：实体（Entity）、结构体（Architecture）、配置（Configuration）、库（Library）和程序包（Package）。其中实体、结构体是必不可少的，程序包和库可以使用默认设置而不明确地显现在程序中。

1. 实体（Entity）：

VHDL 的基本单元，相当于一个“黑盒”，用来描述所设计的单元的外部接口，而不能描述内部功能。

重要性：一个设计文件必须有一个或更多实体。

格式：ENTITY 实体名 IS

[GENERIC (类属参数说明); ][1]

[PORT (端口说明);]

[实体说明部分;]

*END 实体名; --IEEE Standard 1076-1987 写法；

*END ENTITY 实体名; --IEEE Standard 1076-1993 写法；

END 语句有两种写法，对应两种标准。注 [1] ：{}及[]内的内容不是必须的。

(1) 类属参数说明：

类属参数说明是实体的可选项，可以没有。它的功能类似 C++中的“宏”（如#define PI 3.14159;），即在设计中多次用到此处定义的参数。这些参数的值如需改变，只要在类属参数说明处更改即可修改整个程序中用到这些参数的地方。类属为所说明的环境提供了一种静态信息通道，类属的值可以由设计实体外部提供。可用来定义端口宽度，实体中的元件数目以及器件延迟时间等。

位置：必须放在端口说明语句之前。

格式：GENERIC(参数名 1: 类型名[: =缺省值 1] {;

参数名 2: 类型名[: =缺省值 2];

……

参数名 n: 类型名[: =缺省值 n]} ); --注意分号位置；

例 2-1：ENTITY mck IS

GENERIC(width : INTEGER := 16);

PORT……

END mck;

(2) 端口说明：

实体说明中每个输入输出信号称为端口，端口对应于实体生成器件图形的一个引脚，它的功能是为设计实体和其外部环境通信的动态信息提供通道。

格式：PORT(端口名 1 {,端口名,……} :端口模式 数据类型;

端口名 2 {,端口名,……} :端口模式 数据类型;

……

端口名 n {,端口名,……} :端口模式 数据类型);

端口模式有 IN、OUT、INOUT、BUFFER、LINKAGE 五种，用来说明数据或者信号通过端口所在实体的具体方向。

| 端口模式 | 说明 |
|---|---|
| IN | 输入端口，只允许信号流入。 |
| OUT | 输出端口，只允许信号流出。 |

| | |
|---|---|
| INOUT | 双向端口，信号既可以流入，也可以流出，应尽量避免使用。 |
| BUFFER | 缓冲端口，类似输出端口，区别是允许实体内部使用该信号以实现内部反馈。 |
| LINKAGE | 无方向端口，可连接任意模式的信号。 |

数据类型有 BOOLEAN、BIT、BIT_VECTOR、INTEGER、STD_LOGIC、STD_LOGIC_VECTOR 等，用来说明通过端口的数据或者信号的类型。

| 数据类型 | 说明 |
|---|---|
| BOOLEAN | 布尔型（取值 true, false） |
| BIT | 位型（取值 0，1） |
| BIT_VECTOR | 位矢量型（n 位 BIT 的组合） |
| INTEGER | 整型，可作循环指针或常数，通常不作 I/O |
| STD_LOGIC | 工业标准逻辑类型（取值 0、1、X、Z，由 STD_LOGIC1164 程序包定义） |
| STD_LOGIC_VECTOR | 工业标准逻辑向量类型（为 STD_LOGIC 的组合） |

例 2-2：
```
ENTITY first IS
PORT ( h1,h2,h3 : IN STD_LOGIC;
       p      : OUT STD_LOGIC;
       addr   : OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
END first;
```

(3) 实体说明部分：

可选项，用来定义设计实体接口中的一些公共信息，例如数据类型的定义以及被动进程语句等。

例：
```
ENTITY ram IS
    GENERIC ……
    PORT ……
    TYPE instruction IS ARRAY (0 TO 7) OF instruction;
END ram;
```

注意：

1° VHDL 程序里用到的名称都应是合法的标识符。

2° 空格仅用来增强可读性，不影响功能。如“: =”也可写成“:=”。

3° VHDL 不区分大小写，字符或字符串除外。区分全角/半角字符，应都用半角字符。

4° VHDL 语句以分号（半角）“;”结尾。

5° VHDL 中用“--”（双减号）表示注释，“--”之后的内容均为注释，一般显示为绿色。

**2.** 结构体（Architecture）：

实体的重要组成部分，描述了设计的实体所要实现的功能。指明了设计实体中的行为和内部器件的连接关系以及数据流程，是对实体功能的具体描述，因此要跟在实体后面，而不能单独存在。结构体内部的语句是并行处理的。每个实体可以有多个结构体（需要用 Configuration 语句来配置），每个结构体对应着实体不同结构和算法实现方案，其间的各个结构体的地位是同等的，它们完整地实现了实体的行为。但同一结构体不能为不同的实体所拥有。

重要性：一个实体必须有一个或更多结构体来描述其功能。

格式：ARCHITECTURE 结构体名 OF 实体名 IS
[定义语句]; --内部信号、常数、数据类型、函数等的定义；
BEGIN
[并行处理语句];
*END 结构体名; --IEEE 1076-1987;
*END ARCHITECTURE 结构体名; --IEEE 1076-1993;

结构体常采用的三种方式进行描述：

1° ARCHITECTURE Behavioral OF XXX IS 使用结构体的行为命名。对应***行为级描述***。采用进程语句顺序描述实体行为，抽象程度高，大量使用逻辑运算、算术运算和关系运算，为高层次描述，不需关注电路组织和门级实现。

2° ARCHITECTURE Dataflow OF XXX IS 使用结构体的数据流命名。对应***数据流级描述***。采用进程语句描述数据流在控制流下被加工、处理、存储的全过程。它描述电路的行为，隐含地表示电路的结构。寄存器明确，常用语法：CASE-WHEN（条件信号赋值语句）和 WITH-SELECT-WHEN（选择信号赋值语句）。

3° ARCHITECTURE Structural OF XXX IS 使用结构体的结构命名。对应***结构级描述***。采用并行处理语句描述实体的结构组织和元件间的连接关系，描述了电路的结构，不明显地描述行为。寄存器和组合逻辑均明确。

这三种方式的区别仅体现在所使用的语句上，结构体的语法是一样的。对应不同描述方式，结构体命名时也尽量与描述方式相对应。另外，三种描述方式可混合使用。

结构体中的定义语句是对结构体的功能描述语句中将要用到的信号(SIGNAL)、数据类型(TYPE)、常数(CONSTANT)、元件(COMPONENT)、函数(FUNCTION)和过程(PROCEDURE)等加以说明的语句。由于是内部信号，故不必定义信号模式和方向。但在一个结构体中说明和定义的数据类型、常数、元件、函数和过程只能用于这个结构体中，若希望其能用于其他的实体或结构体中，则需要将其作为程序包来处理。

并行语句是结构体功能的主要部分。这些语句具体描述结构体的行为和连接关系。语句之间没有顺序关系，并行执行。如果用模块化结构描述则各模块间是并行的，内部的执行顺序要视其内部描述方式而定。如果用进程描述，则各个进程之间是并行的，进程内部是顺序的。

3. 子结构：

结构体的组成部分，含三种方式：进程（Process）、模块（Block）和子程序（Subprogram）。

(1) 进程（Process）：

格式：[进程名:]PROCESS(敏感信号 1,敏感信号 2,……)
变量说明
BEGIN
……
END PROCESS [进程名];

1° END PROCESS 后的进程名可省略。

2° 与 Block 中不同的是 Process 中的语句是按书写顺序执行的，而不是并行执行的。

3° 进程只有两种运行状态：执行状态和等待状态。

4° PROCESS 后括号中的信号量通常称为敏感信号。只要其中有一个信号发生了变化，都将启动 PROCESS 语句。即只要有敏感信号变化，PROCESS 进程就会执行一遍。

5° PROCESS 也可以不包含敏感信号。它的启动还可以由 WAIT 语句等待一个触发条件的成立。

格式为：PROCESS
变量说明
BEGIN
WAIT 语句
……
END PROCESS;

值得注意的是，进程一定要有敏感信号表或者 WAIT 语句，否则会进入死循环状态。

6° 同一结构体中的进程是并行执行的（进程内部是串行执行的），也可以通过信号互相通信。如 P1 进程的执行使 P2 进程的敏感信号发生变化而启动 P2 进程，P2 反过来又会触发 P1 进程工作，如此循环。

(2) 模块（Block）：

格式：块结构名:
BLOCK
BEGIN
……
END BLOCK 块结构名;

例：
```
ARCHITECTURE BLK_CON OF CON IS
BEGIN
  BLK1:        --模块开始；
  BLOCK
  BEGIN
    Q<=D;
  END BLOCK BLK1;   --模块结束；
END ARCHITECTURE BLK_CON;
```

1° BLOCK 模块中的各个语句是并发执行的与书写顺序无关。结构体内各语句也是并发执行的。一个结构体可包含多个模块（Block）。

2° 并发执行的语句分为两种：无条件并发执行和有条件并发执行。有条件并发执行的 BLOCK 被称为 GUARDED BLOCK（卫式 BLOCK）。

格式：BLOCK (条件布尔表达式)

如：
```
BLK2
  BLOCK (CLK='1')
  BEGIN
    P<=GUARDED F;
  END BLOCK BLK2;
```

卫式 BLOCK 模块中的语句都有保留字 GUARDED，表明只有条件布尔表达式成立时，才执行该语句。

(3) 子程序（Subprogram）：

能被主程序调用的某一特定功能的程序段。子程序被调用时，首先要初始化。执行后将结果返回到主程序。

VHDL 语言有两种子程序：过程（Procedure）和函数（Function）。

Ⅰ. 过程（Procedure）：

格式：PROCEDURE 过程名(参数 1;参数 2;……)IS

[定义语句]; --定义变量等；
BEGIN
[顺序处理语句];
END 过程名;

其中，参数可以是输入也可以是输出（但必须指明信号方向），都列在括号里，以分号间隔。一般 IN 作为常数处理，OUT 和 INOUT 作为变量进行拷贝，过程语句被主程序调用后，将变量 OUT 和 INOUT 拷贝到调用者的信号和变量中。如果需要在过程中把 IN 和 INOUT 作为信号使用，则应该用定义语句特别指明。

例：
```
PROCEDURE VEC_CHG (TC: IN STD_LOGIC_VECTOR;
                   TQ: OUT BOOLEAN;
                   Q: INOUT INTEGER)IS
BEGIN
    Q := 0;
    ……
END VEC_CHG;
```

过程（Procedure）调用时，先传值给输入参数（TC），然后按顺序执行过程中的语句，执行结束后，将输出值拷贝给调用者 OUT 或 INOUT 所定义的变量或信号（TQ 和 Q）中。子程序调用格式（借上例），如：VEC_CHG(TC,TQ,Q); Z 值的变化触发过程执行，结果存于 TQ 和 Q 中，结构体中的其他语句可直接使用该结果。

Ⅱ.函数（Function）

格式：
```
FUNCTION 函数名 (参数 1;参数 2;……) RETURN 数据类型名 IS
[定义语句];
BEGIN
[顺序处理语句];
RETURN [返回变量名];
END 函数名;
```

参数 1，参数 2 等均为输入信号（分号间隔），所以不用指定信号方向。函数的输入由调用者拷贝到括号里的参数中。

通常各种功能的函数程序都被集中在程序包（Package）中。

例：
```
PACKAGE FUNC IS          --在程序包 FUNC 中定义函数 MAX；
    FUNCTION MAX (A : STD_LOGIC;
                  B : STD_LOGIC_VECTOR)RETURN STD_LOGIC IS
    VARIABLE TEMP : STD_LOGIC(A'RANGE);
    BEGIN
      ……
    RETURN TEMP;
    END MAX;
END FUNC;
```

函数调用：如 PEAK<=MAX(DATA,PEAK);

注意：过程（Procedure）和函数（Function）的区别：

函数有返回值，过程没有；函数参数限定为 IN，过程可以为 IN、OUT 或 INOUT；函数参数为变量或信号，过程为变量、信号或常量；函数默认参数类型为常量，而过程中 IN 默认为常量，

INOUT 和 OUT 默认为变量；过程中运行使用等待语句和顺序信号赋值语句，函数中不允许。

4. 配置（Configuration）：

VHDL 程序的基本组成部分，用于描述层与层之间的连接关系和实体与结构体之间的连接关系。在实体与结构体之间的连接关系的配置说明中，设计者可以利用配置语句为实体选择不同的结构体。VHDL 程序设计中，配置的功能就是把元件安装到设计实体中，可以把配置看作是设计的零件清单。

配置可分三类：默认配置、元件配置和结构体配置。配置通常跟在结构体后面。

共同格式：CONFIGURATION 配置名 OF 实体名 IS

```
    FOR 选配结构体名
    END FOR;                --[说明语句];
END 配置名;
```

1° 默认配置：

一个实体带多个结构体时，可以把实体说明和某一特定结构体通过默认配置组成不同的实体—结构体对。但需要注意的是，默认配置属于显式配置，因此当一个结构体中含有任何元件和块语句时将不能采用默认配置，否则 EDA 工具编译时会报错。

格式：CONFIGURATION 配置名 OF 实体名 IS

```
    FOR 选配结构体名;
    END FOR;
    END 配置名;
```

可见，默认配置的说明语句只包含为实体配置的结构体名。如不同进制的计数器，可以只定义一个实体，而带不同的结构体以构成所需进制的计数器。

2° 元件配置：

层次化设计中，为提高效率，常需引用库中的元件。编写 VHDL 程序时，与元件引用密切相关的语句是例化元件说明语句（COMPONENT）和例化元件映射语句（PORTMAP）。一般这两个语句仅说明元件的端口信息，而不会给出元件属于哪个设计库里的哪一个设计实体，为了避免混淆，可采用元件配置语句来引用结构体中的元件进行配置。

元件配置语句又包含两种形式，即低层次的配置和实体—结构体的配置。

前者格式：CONFIGURATION 配置名 OF 实体名 IS

```
        FOR 配置结构体名;
          FOR 例化标号 : 元件名 USE CONFIGURATION 库名.元件配置名;
          END FOR;
          FOR ……
          END FOR;
          ……
        END FOR;
    END 配置名;
```

对于后者，只需将划线部分改成“库名.实体名(结构体名)”即可。即为设计实体中引用的所有元件直接指定其设计实体的结构体。

3° 结构体配置：

结构体配置是对结构体中所用元件进行配置。与元件配置不同，结构体配置的配置说明不需要和元件所在的结构体分开。

格式也有两种：

FOR 例化标号 ： USE CONFIGURATION 库名.元件配置名;

FOR 例化标号 ： USE ENTITY 库名.实体名(结构体名);

5. 库（Library）和程序包（Package）：

(1) 库（Library）用于存储和放置可编译的的设计单元的集合，它存放实体说明、结构体、配置说明、程序包标题和程序包体，可通过其目录进行查询和调用。库实际上就是一种存储预先完成的程序包、数据集合体和元件的“仓库”，以方便设计者共享已经编译成功的设计成果。

VHDL 库分两类：设计库和资源库。每当综合器在较高层次的 VHDL 源文件中遇到库语言时，就将随库指定的源文件读入，并参与综合。因此，必须在设计实体前使用库语句和 USE 语句。

当前在 VHDL 中使用的库的种类有：STD 库、IEEE 库、ASIC 库、WORK 库和用户定义库。常用库简介：

1° IEEE 库是被 IEEE 正式认可的标准化库，也最为重要和常用。包含的程序包有 STD_LOGIC_1164、 STD_LOGIC_ARITH、STD_LOGIC_SIGNED、STD_LOGIC_UN-SIGNED 等。

其中，STD_LOGIC_1164 是 IEEE 库中最常用的程序包，它定义了使用最多、最广的、满足工业标准的两个数据类型 STD_LOGIC 和 STD_LOGIC_VECTOR。但由于符合 IEEE 库标准的程序包并非符合 VHDL 语言标准，故在 VHDL 设计实体前 要显式说明所用的 IEEE 库中的程序包。

STD_LOGIC_ARITH 在 STD_LOGIC_1164 的基础上扩展了三个数据类型 UNSIGNED、SIGNED 和 SMALL_INT，并为其定义了相关的算术运算符和转换函数。

2° STD 库是 VHDL 的标准库，是所有设计单元所共享、默认的库。其中包含 STANDARD 和 TEXTIO 两个程序包。其中使 STANDARD 程序包时用时不需按标准格式说明，即不需显式说明即可在程序的任意位置调用。它定义了最基本的数据类型（ Bit, bit_vector, Boolean, Integer, Real, Time）。

TEXTIO 程序包主要供仿真器使用。可以用文本编辑器建立一个数据文件，文件中包含仿真时需要的数据，然后仿真时用 TEXTIO 程序包中的子程序存取这些数据。综合器中，此程序包被忽略。在使用本程序包之前，需加语句 USE STD.TEXTIO. ALL。

3° ASIC 矢量库是为各个 EDA 厂商和公司提供的面向 IC 设计的特色工具库和元件库，在该库中存放着与逻辑门一一对应的实体。使用该库需要进行说明。

4° WORK 库是 VHDL 的现行工装库，用于保存当前的设计单元，是用户的临时仓库，用户设计的成品、半成品、模块等都放在其中。使用该库毋须声明。

5° 用户定义库为设计者自己开发的设计单元的集合库。使用时需要说明库名。

(2) 程序包（Package）用来单纯地罗列 VHDL 语言中所要用到的信号定义、常数定义、数据类型、元件语句、函数定义和过程定义等。它是可编译、可调用的设计单元，也是库结构中的一个层次。程序包由两部分构成，程序包标题（Package Header）和程序包体（Package Body）。程序包标题为程序包定义的接口，声明其中的信号、常数、数据类型、元件、子程序等，声明方式与实体说明中的端口定义类似。程序包体规定程序的实际功能以及存放说明中的子程序，其声明方式与结构体中语句方法相同。即程序包标题列出了所有项的名称，而程序包体给出了所有项的细节。程序包体可缺省，因为程序包标题中也允许使用数据赋值和有实质性的操作语句。

格式：PACKAGE 程序包名 IS

[说明语句];

END 程序包名;
PACKAGE BODY 程序包名 IS
[说明语句];
END 程序包名;

标题和体分开说明使调整和重新编译变得简单。

库和程序包描述语句总是放在 VHDL 程序的最前面：

格式：LIBRARY 库名;
USE 库名.程序包名.项目名;

例：LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

库是相互独立的，不能嵌套。

1° 库及程序包的说明总是放在实体单元前面，默认库及程序包可不作说明。

2° 用关健字 library 说明要使用的库，用关健字 use 说明要使用的库中的程序包。库及程序包的作用范围：仅限于所说明的设计实体。每一个设计实体都必须有自己完整的库及程序包说明语句。

## 三、VHDL 的描述方法：

1. 标识符：

标识符 IEEE Standard 1076-1987 中有关标识符的语法规范被 IEEE Standard 1076-1993 完全接受并加以扩展。前者称为短标识符，后者称为扩展标识符。扩展标识符允许使用保留字，允许包含图形符号和空格，允许多下划线相连，区分大小写等是短标识符所不具备的。由于目前仍有许多 VHDL 工具不支持扩展标识符，故建议使用短标识符。

2. 词法：

(1) 注释：

以“--”（双减号）开头，到行尾截至，允许使用特殊字符，仅用以增强程序可读性。

(2) 语句：

除某些特定框架外，一般语句均以分号（半角）结尾。

(3) 数字：

标量，包含实数和整数。根据进制不同，可分为十进制数和基数字两类。

实数在使用时，必须带小数点，如 0.0，22.32，8.0 等，不可综合。

1° 十进制数的定义格式为：

整数[.小数][指数]

整数可表示为“整数_整数”，数字之间的下划线不影响数值，但数字之间不能有其它字符或空格。

指数可表示为“E±整数”，只有十进制数才允许指数为负值。

2° 以基表示的数，定义格式如下：

基#基于基的整数[.基于基的整数]#[指数]

基是一个整数，最小为 2，最大为 16。

基于基的整数可表示为“扩展数字_扩展数字”，扩展数字可以为十六进制中的 0~F。下划线及字母大小写均不影响数值。

字符“#”不能省略。例：2#1111_1111#，16#F_34#e8。

(4) 字符：

仅包含一个字符的词法单元，格式: ‘ASCII 字符’。

(5) 字符串：

包含若干个字符，格式：“ASCII 字符序列”。

(6) 位串：

用来表示位矢量，由进制标识符和数字字符串组成，格式为：

基数说明符(B/O/X) “数字字符串”

位串长度以二进制数位为基准。

B，O，X 分别表示 2，8，16 进制数。数字字符串中可以有下划线以增强可读性。

如：B “0010_1101”，X “FA0”(位串长度为 12，而非 3)。

3. 数据对象：

Data Object，可赋值的客体，主要有三种类型：常量（Constant）、变量（Variable）和信号（Signal）。

| 类型 | 作用范围 | 使用场合 |
|---|---|---|
| 常量 | 全局量 | 实体说明 结构体描述 程序包标题 进程说明 过程说明 函数调用说明 |
| 变量 | 局部量 | 进程说明 过程说明 函数调用说明 |
| 信号 | 全局量 | 实体说明 结构体描述 程序包标题 |

(1) 常量：

固定值，相当于硬件电路中的恒定电平。通常被放在程序的开始，数据类型也同时在说明语句中指出。

格式：CONSTANT 常量名 : 数据类型: = 表达式;

例：CONSTANT VCC : REAL : = 5.0;

注意：常量只能初始化，不能多次赋值。定义在程序包中的常量可由所在的任何实体和结构体调用，定义在实体内或进程内的常量仅能在实体或进程内使用。VHDL 为强数据类型语言，数据必须有类型，且类型一致或兼容才能赋值。

(2) 变量：

局部量，相当于电路连接线上的信号值。变量最终是要赋值给信号的，不能作为进程的敏感信号参数。一定要注意，变量仅能在进程（Process）说明、过程（Procedure）说明和函数（Function）调用说明中使用。

1° 说明语句格式：

VARIABLE 变量名 : 数据类型 约束条件 :=表达式;

例：VARIABLE M,N:INTEGER;

VARIABLE NUM:INTEGER RANGE 0 TO 127 : = 20;

第二句定义了一个范围为 0~127 的整数。

2° 变量赋值格式：变量名 : = 表达式;

赋值符号“: =”是立即生效的，没有赋值延时，也不能附加延时。

(3) 信号：

全局量，可用于进程之间的通信，与硬件的“连线”相对应。除无数据流动方向外，其他同端口概念。用于实体、结构体和程序包。

1° 说明语句格式：

SIGNAL 信号名 : 数据类型 约束条件 := 表达式;

2° 信号赋值语句格式：

信号名 <= 表达式[AFTER[时间表达式]];

其中，“<=”表示信号的代入赋值，是信号之间的传递。时间表达式用于指定延迟时间，缺省时取默认值。

(4) 文件：

文件用于存放大量数据，在仿真测试时，测试输入的激励数据和测试的结果都可以存放在文件中。文件不能像变量或信号那样通过赋值更新内容，但可以作为参数向子程序传递，通过子程序对文件进行读和写操作。

格式：FILE 文件名 : 文件类型 OPEN [打开模式] IS “物理文件名”;

文件类型，如 TEXT（文本文件）等。打开模式有 READ_MODE 和 WRITE_MODE，表示打开文件用于读或写。物理文件名为实际文件的路径，如果为当前工作目录，则可省略路径而只写文件名。

(5) 变量与信号的区别：

| 项目 | 信号 | 变量 |
|---|---|---|
| 赋值符号 | <= | := |
| 赋值后的变化 | 经延迟后称为当前值 | 立即变成当前值 |
| 作用范围 | 全局 | 局部 |

就是说，信号可以设置延时量，而变量则不能；变量只能作为局部的信息载体，而信号则可作为模块间的信息载体。变量的设置有时只是一种过渡，最后的信息传输和界面间的通信都靠信号来完成。

(6) 信号的属性函数：

用来得到信号的行为信息和功能信息。定义属性的一般格式为：

项目名’属性表示符; --不要漏掉单引号；

信号的属性函数有：

1° signal’ DELAYED[(time)]

延时函数。信号表达式在 time 表达式成立时得到。

2° signal’ STABLE[(time)]，

稳定函数。如果 signal 在 time 时间内保持稳定，没有事件发生，函数返回布尔型变量 true，否则返回 false。

3° signal’ QUIET[(time)]

安静函数。如果 signal 在 time 时间内没有事项要处理，则返回 true，否则返回 false。

4° signal’ TRANSACTION

处理函数。建立一个 BIT 型的信号，当 signal 每发生一次变化时，该 BIT 信号翻转一次。

5° signal’ EVENT

事件函数。如果在当前模拟周期内，该信号发生某个事件（信号值发生了变化），函数返回 true，否则返回 false。

6° signal’ ACTIVE

活跃函数。在当前模拟周期内，如果信号发生了变化，时间做了处理，则返回 true，否则返回 false。

7° signal’ LAST_EVENT

返回信号最后一次改变到现在时刻所经历的时间。