第3章 VHDL基础

硬件描述语言(Hardware Description Language,HDL)是一种用于数字系统设计的高级语言,具有很强的电路描述和建模能力,大大简化了硬件设计任务,提高设计的效率和可靠性。以 HDL 语言设计,以 CPLD/FPGA 为硬件实现载体,以 EDA 软件为开发环境的现代数字系统设计方法已经被广泛采用。

本章将介绍常用硬件描述语言 VHDL 的基本知识,包括 EDA、VHDL 简介,基于 VHDL 的数字系统设计流程; VHDL 程序的基本结构,数据对象、数据类型、运算符和表达式; 顺序语句,并行语句; VHDL 库和程序包等。

3.1 硬件描述语言 VHDL 介绍

3.1.1 EDA 技术及发展

EDA 是电子设计自动化(Electronic Design Automation)的缩写,是 20 世纪 90 年代初从计算机辅助设计(CAD)、计算机辅助制造(CAM)、计算机辅助测试(CAT)和计算机辅助工程(CAE)等概念基础上发展而来的新兴电子设计技术。

EDA 技术以大规模可编程逻辑器件为设计的载体,依赖功能强大的计算机,在 EDA 开发软件平台上用软件的方法设计电子系统。采用硬件描述语言描述系统逻辑,生成系统设计文件,软件自动完成逻辑编译、逻辑化简、逻辑分割、逻辑综合及优化、布局布线、逻辑仿真测试,直至实现电子系统的功能。再针对指定的目标芯片适配编译、逻辑映射、编程下载等,最终完成对电子系统硬件功能的实现。

伴随着大规模集成电路制造技术、可编程逻辑器件、计算机辅助工程,以及电子系统设计技术的发展,EDA 技术的发展经过了三个主要阶段。

1. 计算机辅助设计阶段(CAD)

20 世纪 70 年代以后,利用计算机的图形编辑、分析和存储能力,辅助设计工程师进行 IC 版图设计、PCB 布局布线等工作,取代了人工劳动。CAD 设计技术初见雏形。

2. 计算机辅助工程阶段(CAE)

20 世纪 80 年代出现的 EDA 工具除了具备图形绘制功能以外,还增加了结构设计和电

路设计功能，代替了部分设计师的工作。在逻辑设计、逻辑仿真分析、布尔方程综合优化、自动布局布线等方面承担了重要的工作。

3. 电子系统设计自动化阶段(EDA)

进入20世纪90年代，出现了以高级语言描述、系统级仿真和综合技术为特征的新一代EDA工具。设计工程师采用结构化、自顶向下的设计方法，先对整个电子系统进行系统级设计和功能模块划分，采用硬件描述语言HDL对各个功能模块描述，再用EDA工具对设计进行行为描述和结构综合，系统仿真和测试验证，自动布局布线，最后编程下载到CPLD/FPGA中。采用这种设计方法后，大大提高了复杂电子系统设计能力，提高了设计效率，缩短了设计周期。

EDA技术进入21世纪后，随着现代半导体精密加工技术发展，基于大规模或超大规模集成电路技术的定制或半定制ASIC(Application Specific IC)器件大量涌现，SOC(System On Chip)技术成熟并得到应用。嵌入式处理器软核成熟，在单片CPLD/FPGA上能够实现功能完备的数字系统，使得SOPC(System On Programmable Chip)进入实用阶段。现代电子技术全面融入到EDA技术中，数字系统建模理论、现代电子系统设计理论、软件无线电技术、数字信号处理技术、图像处理技术的应用，使得传统的电子系统设计理念发生重大变化。在设计和仿真中支持标准硬件描述语言的EDA软件，以及系统级硬件描述语言的出现使复杂电子系统的设计变得简单。

3.1.2 VHDL语言简介

硬件描述语言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，是EDA技术的重要组成部分。

在HDL语言出现前的高级语言，如C、Pascal、Fortran、Basic等，只适合用于描述过程和算法，无法用于硬件行为描述。随着EDA技术的发展，需要专用的硬件描述语言作为EDA工具的工作语言，于是各个EDA工具软件开发商都推出了相应的HDL语言。但是，这些语言一般各自面向特定的设计领域与层次，因此急需一种面向设计的多领域、多层次并得到普遍认同的标准硬件描述语言。

20世纪70年代末和80年代初，面对各个电子系统承包商技术线路不一致，使得产品不兼容，采用各自的设计语言，信息交换和维护困难，设计不能重复利用等情况，由美国国防部牵头，来自IBM、Texas Instruments和Intermetrics公司的专家组成VHDL(Very High Speed Integrated Circuit HDL)工作组，提出了新的硬件描述语言版本和开发环境。IEEE标准化组织进一步发展，经过反复的修改与扩充，在1987年宣布了VHDL语言标准版本，即IEEE STD 1076—1987标准。1993年，VHDL—87标准被重新修订，更新为IEEE STD 1076—1993标准。现在公布的最新版本是IEEE STD 1076—2002。

1995年，我国国家技术监督局制定的《CAD通用技术规范》推荐VHDL作为我国电子设计自动化硬件描述语言国家标准。从此，VHDL语言在我国迅速普及，成为广大硬件工程师必须掌握的一项技术。

VHDL语言能够成为标准化的硬件描述语言并获得广泛应用，是因为有其他硬件描述语言不具备的优点：

- 较强的系统级和电路描述能力。VHDL 语言可用于描述系统级电路，采用多层次、模块化、自顶向下与自底向上或混合方式描述系统功能。也可以采用行为描述、寄存器传输描述或结构描述，或三者混合描述门级电路，同时，VHDL 语言还支持同步、异步和随机电路设计，这是其他硬件描述语言难以比拟的。
- 与具体器件无关，可移植性强。VHDL 语言是标准化硬件描述语言，同一个设计描述可以被不同的工具实现。设计人员采用 VHDL 语言设计硬件电路时，可以不针对某种具体的器件，也不需要熟悉器件的内部结构，设计人员只需集中精力进行系统设计和优化。设计完成后，再选用不同结构的器件实现其功能。
- 基于库的设计方式，便于复用。VHDL 语言采用基于库(Library)的设计方式，使得设计不用从门电路一步步进行，可以直接复用以前设计中存放在库中的模块和元件，提高了设计效率。
- 语法规范、易于共享。VHDL 的语法规范，可读性极强。用 VHDL 编写的代码文件既是程序，也是文档；既可以作为设计人员之间交流的内容，又可以作为签约双方的合同文本。另一方面，作为一种工业标准，VHDL 易于共享，适合大规模协作开发。

常见的 HDL 语言还有 Verilog HDL、ABEL、AHDL、SystemVerilog 和 SystemC 等，其中 VHDL 和 Verilog HDL 是现在 EDA 设计中使用最多的两种 HDL 语言。

3.1.3 VHDL 语言设计开发流程

以 CPLD/FPGA 为硬件载体，采用 VHDL 语言的 EDA 软件进行数字系统设计的完整流程包括设计方案制定、设计输入、逻辑综合、布局布线、仿真测试、编程下载等。其他硬件描述语言的设计过程也是类似。设计流程图如图 3-1 所示。

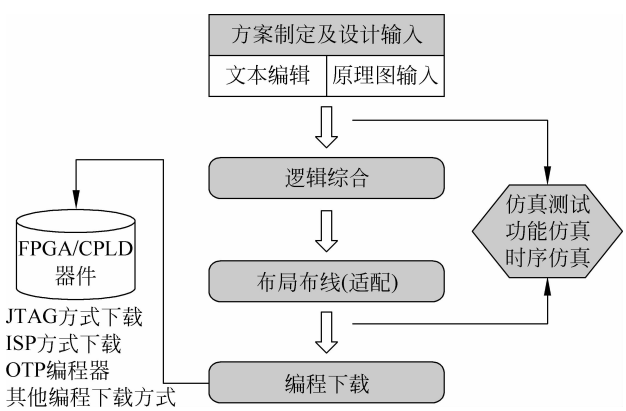


图 3-1 设计流程图

1. 设计方案制定

采用自顶向下、模块化设计的设计方式，确定整个系统的设计方案，划分系统的各个逻辑模块，确定各个模块的功能，以及采用的设计方式。

2. 设计输入

利用 EDA 软件中的文本编辑器将系统功能或结构用 VHDL 语言描述出来，保存为

VHDL 文件格式，为后面的综合优化做准备。

现代大多数 EDA 软件除了可以使用 HDL 语言设计输入以外，通常还支持类似传统电子系统设计的原理图输入方式。原理图输入方式中使用的逻辑模块或符号，可以使用 EDA 软件库中预制的功能模块，也可以使用 VHDL 语言设计的模块或原件。

实际上，图形输入方式除了原理图输入外还有状态图输入和波形输入等常用方式。

采用模块化设计方式，完成各个功能模块设计后，将各个模块组合在一起，即完成对整个系统的设计。

3. 逻辑综合

所谓综合就是将较高层次的抽象描述转化为低层次描述的过程，是将软件设计转化为硬件电路的关键步骤。在完成设计输入后，根据硬件结构和约束条件进行编译、优化、综合，最终得到门级甚至更低层次的电路描述网表文件。网表文件就将软件描述和给定的硬件结构形成对应逻辑连接关系。

4. 布局布线（适配）

布局是指将网表文件中的逻辑连接关系合理地配置到目标器件内部的硬件结构上，通常需要在速度优先还是面积最优间选择。布线就是根据布局的拓扑结构，利用目标器件内部资源，合理地连接各个单元。适配后产生的仿真文件可用于精确的时序仿真，同时生成用于编程下载的文件。

5. 仿真测试

仿真是 EDA 设计过程中的重要步骤，通常 EDA 软件中会提供仿真工具，也可以使用第三方的专业仿真工具。根据不同的实施阶段，分为功能仿真和时序仿真。

功能仿真：在采用不同方式完成设计输入后，即可进行逻辑功能的仿真测试，以了解功能是否满足设计要求。这个阶段的仿真测试不涉及具体的硬件结构、特性。

时序仿真：又称后仿真，是最接近硬件真实运行的仿真。利用布局布线后生成的包含硬件特性参数的仿真文件，对系统和各个模块进行时序仿真，分析其时序关系和延迟信息。

6. 编程下载

将适配后生成的下载或配置文件，通过编程器或下载线缆下载到目标器件中。一般将对 CPLD 的下载称为编程，对 FPGA 的下载称为配置。最后将整个系统进行统一的测试，验证设计在目标系统上的实际工作情况。

3.2 VHDL 程序的基本结构

VHDL 程序是由库（library）、程序包（package）、实体（entity declaration）、结构体（architecture body）、配置（configuration）五部分组成。设计实体结构图如图 3-2 所示，其中设计实体必须有实体和结构体，其他部分根据设计需要来添加。

实体是 VHDL 程序的基本单元，类似原理图设计中的一个元件符号。其中实体说明部

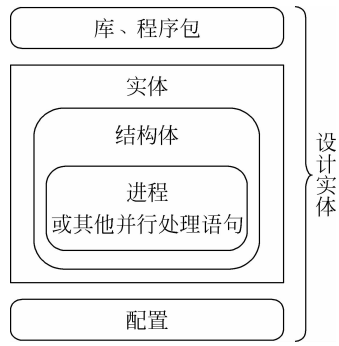


图 3-2 设计实体结构图

分规定了其与外界通信的引脚或接口信号。在实体内部有一个或多个结构体，用来描述设计的逻辑结构或功能。

【例 3-1】 VHDL 程序基本结构。

```
实体 端口说明:
        LIBRARY  IEEE;                                  -- 库说明部分
        USE  IEEE.STD_LOGIC_1164.ALL;                   -- 程序包说明部分
        ENTITY  nand2  IS                               -- 实体说明部分
          PORT ( a,b: IN    STD_LOGIC;
                 y: OUT   STD_LOGIC);
        END  nand2;
结构体 进程:
        ARCHITECTURE  arch_name  OF  nand2  IS          -- 结构体描述部分
        BEGIN
        PROCESS (a , b)
              VARIABLE  comb : STD_LOGIC_VECTOR ( 1 DOWNTO 0 );
          BEGIN
              comb : = a & b;
              CASE comb IS
                 WHEN "00" = > y < = '1';
                 WHEN "01" = > y < = '1';
                 WHEN "10" = > y < = '1';
                 WHEN "11" = > y < = '0';
                 WHEN OTHERS = > y < = '0';
              END  CASE ;
          END  PROCESS ;
        END  arch_name ;
```

3.2.1 实体说明

实体说明部分的一般结构：

```
ENTITY 实体名 IS
    [GENERIC (类属表); ]
    [PORT (端口表); ]
END [ENTITY] 实体名;
```

1. 实体名

实体说明部分以“ENTITY 实体名 IS”开始，以“END［ENTITY］实体名”结束。其中实体名由设计者自定义，一般根据所设计实体的功能来取名，ENTITY 是 VHDL 语法规定中的保留关键字。大多数 EDA 软件中的编译器和适配器是不区分 VHDL 语言大小写的，但为了保持良好的设计风格和便于阅读，通常将 VHDL 语言的标识符和保留关键字以大写表示，设计者自定义符号小写表示，如实体名、结构体名、变量名等。

2. 类属说明

类属参数用来在不同层次的设计模块间传递信息和参数，比如数组长度、位矢量长度、端口宽度、器件延时时间等。这些参数都要求是整数类型。

类属说明的一般格式如下：

```
  GENERIC (参数 1: 参数类型 [ : = 静态表达式];
           参数 2: 参数类型 [ : = 静态表达式];
           ⋮
           参数 n: 参数类型 [: = 静态表达式]);
ENTITY entity_name IS
   GENERIC (n: INTEGER : = 100 );
   PORT (A: IN STD_LOGIC_VECTOR ( n - 1 DOWNTO 0 );
         Y: OUT STD_LOGIC);
END entity_name;
```

3. 端口说明

端口说明是对设计实体和外部接口的描述，是设计实体和外部通信的通道，对应电路图上的引脚。一个端口就是一个数据对象，包括端口名、数据类型、通信模式。端口说明的一般格式如下：

```
PORT (端口名 1: 通信模式 数据类型;
      端口名 2: 通信模式 数据类型;
      ⋮
      端口名 n: 通信模式 数据类型;
      );
```

通信模式说明数据、信号通过端口的流动方向，主要有四种。

① IN：定义端口为单向只读模式。数据或信号从外部流向实体内部，或者从该端口读取外部数据。

② OUT：定义端口为单向输出模式。数据或信号只能从该端口流出，或者向该端口赋值。

③ BUFFER：定义端口为缓冲模式。该模式和输出模式类似，区别在于缓冲模式允许实体内部应用该端口信号即允许内部反馈，输出模式则不能用于内部反馈。缓冲模式的端口只能连接设计实体内部信号源，或者是其他实体的缓冲模式端口。

④ INOUT：定义端口为输入输出双向模式。在某些设计实体中，例如双向总线、RAM

数据口、单片机的I/O口等，数据是双向的，既可以流入实体内部，也可以从实体流出，这时需设计为双向模式。实体内部的信号和外部输入实体的信号都可以经过双向模式端口，也允许引入内部反馈，所以双向模式是一个完备的端口模式。

在VHDL语言中，任何一个数据对象都要对其数据类型做明确的说明，端口也是一样。EDA工具软件支持的数据类型较多，在3.3节中将详细介绍相关数据类型。

3.2.2 结构体描述

结构体具体描述了设计实体行为，定义了实体的逻辑功能或内部电路结构关系，规定了该实体的数据流程，建立了实体输出与输入之间的关系。

结构体的一般格式如下：

```
ARCHITECTURE  结构体名  OF  实体名  IS
   [定义语句]内部信号,常数,数据类型,函数定义;        -- 说明语句
BEGIN
   [进程语句];                                      -- 功能描述语句
   [并行处理语句];
END [ARCHITECTURE] [结构体名];
```

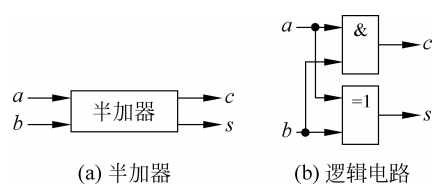
“说明语句”用来说明和定义结构体内部使用的信号、常数、数据类型、函数、过程、元件调用声明等，这是结构体中必需的。

“功能描述语句”描述结构体的行为、功能、电路连接关系等，可以是并行语句、顺序语句或者它们的混合。其中并行语句是结构体描述的主要语句，并行语句间是并行的，没有顺序关系。进程语句是典型的并行语句，进程间是并行的，但进程内部的语句是有顺序的。

结构体功能可以用三种方式进行描述，即行为描述法、数据流描述法、结构描述法。

1. 行为描述法

行为描述表示输入与输出间转换的关系，是对设计实体按算法的路径来描述。采用进程语句，顺序描述设计实体的行为。这种描述方式通常是对整体设计功能的定义，不是对单一器件进行描述，是一种高层次的描述方法，如图3-3(a)所示的半加器。



(a) 半加器　　(b) 逻辑电路

图3-3 半加器及其逻辑电路

对半加器的行为描述：

```
ARCHITECTURE alg_ha OF half_adder IS
BEGIN
    PROCESS(a,b)
    BEGIN
        IF a = '0' AND b = '0' THEN
           c <= '0';
           s <= '0';
        ELSIF a = '1' AND b = '1' THEN
           c <= '1';
           s <= '0';
        ELSE
```

```
            c <= '0';
            s <= '1';
        END IF;
    END PROCESS
END alg_ha;
```

2. 数据流描述法

数据流描述法描述了数据流程的运动路径、运动方向和运动结果，采用进程语句顺序描述数据流在控制流作用下被加工、处理、存储的全过程。

由半加器的真值表可推导信号间逻辑关系，用逻辑表达式描述如下：

$$s = a \oplus b$$

$$c = a \cdot b$$

基于上述逻辑表达式的数据流描述为：

```
ARCHITECTURE dataflow_ha OF half_adder IS
BEGIN
    s <= a XOR b;
    c <= a AND b;
END dataflow_ha;
```

可见，结构体内的两条信号赋值语句之间是并行关系，每一赋值语句均相当于一个省略了“说明”的进程，描述了信号从输入到输出的路径。而行为描述中进程内的信号赋值语句是顺序语句。

3. 结构化描述法

结构化描述给出了实体内部结构组织、所包含的模块或元件及其互连关系。

结构化描述通常用于层次化结构设计。对于一个复杂的电子系统，将其分解成许多子系统，子系统再分解成各个功能模块。多层次设计的每个层次都可以作为一个元件，再构成一个模块或构成一个系统，每个元件分别仿真，然后再整体调试。

图 3-3(a)所示的半加器可以用图 3-3(b)所示的逻辑电路加以实现。对该电路结构采用结构化描述法的程序如下：

```
ARCHITECTURE struct_ha OF half_adder IS
  COMPONENT and_gate
      PORT (a1,a2: IN BIT;
            a3: OUT BIT );
  END COMPONENT;
  COMPONENT xor_gate
    PORT (x1,x2: IN BIT;
          x3: OUT BIT );
    END COMPONENT;
BEGIN
    g1: and_gate PORT MAP (a,b,c );
    g2: xor_gate PORT MAP (a,b,s );
END struct_ha;
```

其中 COMPONENT 为元件说明语句，说明元件的名称及端口特性。本结构体有两条并行的元件例化语句，被例化的元件为 and_gate 和 xor_gate。PORT MAP 为端口映射，指明元件之间以及元件与实体端口之间的连接关系。

3.3 数据对象、数据类型、运算符和表达式

在 VHDL 语言中可以赋值的客体叫做数据对象。每一种数据对象代表的物理含义和使用规则、允许赋值的数据类型、可以参与的运算等都有严格的规定。

3.3.1 数据对象

VHDL 语言的基本数据对象有三种：常量、变量和信号。变量、常量和其他高级语言中相应类型类似，信号则是硬件描述语言中特有的，它带有硬件特征。从硬件电路的角度来看，信号和变量相当于电路之间的连线或连线上的信号值，常量则相当于电源(VCC)、地(GND)等。

1. 常量

常量(Constant)是设计者在实体中给某一常量名定义数据类型和赋值，在程序中试图多次给常量赋值是错误的。常量定义的一般格式如下：

```
CONSTANT  常量名 : 数据类型 : = 表达式 ;
```

其中表达式的数据类型必须和定义的常量数据类型一致。

常量定义一般包含在实体、结构体、程序包、进程、函数、过程等设计单元中。

例如：

```
CONSTANT VCC : REAL : = 3.3 ;
```

该例子中，常量 VCC 被赋值为实型类型数据，在程序中该常量的值将不能再改变，并保持到程序结束。

```
CONSTANT ABUS : BIT_VECTOR : = "11000101" ;
```

常量 ABUS 的数据类型是 BIT_VECTOR，被赋初值为"11000101"，在程序中被作为某一器件的固定地址。

2. 变量

变量(Variable)是个局部量，作为一个临时的数据存储单元，只能在进程、函数、过程等结构中使用，不能将信息带出它定义所在的当前结构。变量赋值是立即生效的，不存在延时。变量定义的一般格式如下：

```
VARIABLE 变量名 : 数据类型 : = 表达式;
```

其中表达式的数据类型必须和定义的变量数据类型一致。

例如：

```
VARIABLE a : STD_LOGIC : = '1';
                                        -- 定义标准逻辑位类型变量 a,初始值为'1'
VARIABLE count: INTEGER RANGE 0 TO 255 ;
                                        -- 定义整数类型变量 count,取值范围为 0～255
```

在变量定义语句中可以给出和变量相同数据类型的初始值,但这不是必需的。由于硬件电路上电后的随机性,很多综合器并不支持初始值设定,这样可以在程序中通过赋值语句来赋予变量一个值。变量赋值的方式如下:

```
变量名 : = 表达式;
```

变量在赋值时不能产生附加延时。例如,tmp1、tmp2 是变量,那么下面产生延时的方式是不合法的:

```
tmp1 : = tmp2 AFTER 10 ns;
```

3. 信号

信号(Signal)是硬件系统描述中的基本数据类型,类似电路内部的连接线,实现实体和实体间、元件和元件间的连接。信号具有全局性特征,不但可以在一个设计实体内部各个单元间传递数据,还可以作为实体中并行语句模块间的信息通道,无须注明信息的流动方向。信号通常在实体、结构体、包集合中定义说明。注意,不允许在进程和过程的顺序语句中定义信号。信号定义的格式如下:

```
SIGNAL 信号名 : 数据类型 : = 表达式;
```

例如:

```
SIGNAL bus_enable : BIT : = '1';        -- 定义 BIT 类型信号,初始值为'1'
SIGNAL data_bus : STD_LOGIC_VECTOR [ 7 DOWNTO 0 ] ;
                                        -- 定义 8 位宽度的数据总线
```

在给出信号的完整定义后,就可对信号赋值。信号赋值语句如下:

```
信号名 <= 表达式 AFTER 时间量;
```

“AFTER 时间量”表示数据信号的传入需延时给定的时间量,这与实际器件的硬件特征是吻合的。

3.3.2 数据类型

VHDL 语言对参与运算的各个量的数据类型有严格要求,相同类型的量之间才能互相传递。VHDL 语言要求设计实体中的常量、变量、信号都要指定数据类型,而且数据类型相同,而位长不同时也不能直接代入。这样就使得 VHDL 编译或综合工具能很容易地找出设计中的错误。

1. VHDL 的标准数据类型

VHDL 语言的标准数据类型共有十种,如表 3-1 所示。