

A GUIDE TO FPGA/VHDL TECHNOLOGY AND ITS APPLICATION

FPGA/VHDL快速工程实践 入门与提高

杨 恒 卢飞成 编著

 北京航空航天大学出版社

<http://www.buaapress.com.cn>



FPGA/VHDL 快速工程实践 入门与提高

A GUIDE TO FPGA/VHDL TECHNOLOGY AND ITS APPLICATION

杨 恒 卢飞成 编著

北京航空航天大学出版社

《FPGA/VHDL 快速工程实践入门与提高》

编委会

主编 杨 恒 卢飞斌

编委 王 辉 唐会华 曹新华

前 言

在高新技术日新月异的今天,科学技术已经成为整个社会发展的源动力,计算机和电子领域的发展更是令人目不暇接。本书结合作者自身的实际工程经验和科研成果,介绍了近年来电子、计算机及信息领域一个非常有发展潜力的热点——FPGA/VHDL 技术。本书包括 5 章及 4 个附录。第 1 章介绍了 FPGA/VHDL 的一些常识性的知识;第 2 章叙述了 VHDL 语言的基本知识;第 3 章介绍了 MAX+PLUS II 集成开发软件的使用(该软件可以从 www.altera.com 网站免费下载);第 4 章以 FPGA 的几个典型应用——LED 数码管显示的电子钟设计、发光二极管表示的交通灯实验以及液晶 LCD 显示为例,较为详细地介绍了 FPGA/VHDL 的软硬件应用;第 5 章介绍了 FPGA/VHDL 相对高级的应用——PCI/ISA 总线接口。本书所提供的源代码和程序大都经过直接验证,可以直接应用于读者的设计中。若本书作为培训教材,建议将本书第 5 章作为具有较强自学能力的学员选学部分。对于需要快速应用 FPGA 技术的读者,可以跳过第 2 章 VHDL 编程基础,直接进入第 3、4 章的学习。

本书是集体智慧的结晶,由香港理工大学杨恒博士与青岛飞洋学院卢飞成院长编著。参加本书编写工作的还有王辉先生、唐会华先生、曹新华先生和申向军先生。本书在成稿过程中,青岛飞洋学院王立祥副院长和计算机系的邵兆娜主任也做了重要的工作。

本书可以独立作为大专院校或工程技术人员学习参考之用,也可以与深圳市英特数码科技有限公司(www.edtyang.com)研发提供的配套学习套件(含开发板、液晶模块、下载线缆和光盘)结合使用。

作者向以下关心过本书或在学术上给予过帮助的朋友表示感谢,他们是:
香港理工大学(Hong Kong Polytechnic University) Prof. H. Y. TAM;
新加坡南洋理工大学(Nanyang Technological University, Singapore) Prof. Chris Koh, Prof. G. B. HUANG;

美国 Ethentica by Security First Corp. 副总裁 Catherine WANG;
西北工业大学陈明教授,博士生导师,蒋东方博士;

北京航空航天大学何立民教授；

山东科技大学贺国平教授、陈新华教授、徐庆莘教授和李克周先生；

深圳文鼎公司钟世雄先生；

深圳广宁实业公司赵广玉先生；

深圳信息职业技术学院管建福副教授、黄奇工程师。

本书在 VHDL 和 MAX+PLUS II 的基本知识方面参考了附录的有关资料，
在此谨向作者表示感谢。

由于作者水平有限，错误和不当之处在所难免，敬请各位读者不吝赐教。

杨 恒 卢飞成

E-mail: Hyang999@sina.com.cn

2003 年 1 月于香港

目 录

第1章 绪 论

1.1	FPGA/CPLD 概述	1
1.2	FPGA/CPLD 的开发软件	2
1.3	FPGA/CPLD 的分类和使用	3

第2章 VHDL 编程基础

2.1	VHDL 语言程序的基本结构	5
2.1.1	设计实体	5
2.1.2	实体说明	5
2.1.3	结构体	7
2.2	VHDL 语言的数据类型及运算操作符	10
2.2.1	标识符	10
2.2.2	对象及其分类	11
2.2.3	数据类型	12
2.2.4	类型转换	16
2.2.5	词法单元	16
2.2.6	运算操作符	17
2.2.7	运算操作符的优先级	19
2.3	VHDL 语法基础	21
2.3.1	并行语句	21
2.3.2	顺序语句	32
2.3.3	延时语句	39
2.4	属性的描述和定义	39
2.4.1	数值类属性函数	40
2.4.2	函数类属性函数	41
2.4.3	信号类属性函数	42
2.4.4	数据类型类属性函数	43

2.4.5 数据区间类属性函数.....	44
2.5 VHDL 程序设计基础	44
2.5.1 库.....	44
2.5.2 程序包.....	47
2.5.3 配 置.....	48
2.6 基本逻辑电路设计实例.....	50
2.6.1 组合逻辑电路设计.....	50
2.6.2 时序电路设计.....	57

第3章 MAX+PLUS II 使用指南

3.1 MAX+PLUS II 的特点	64
3.2 MAX+PLUS II 的组成	65
3.3 MAX+PLUS II 的使用	65
3.3.1 图形文件的建立.....	66
3.3.2 文本文件的建立.....	70
3.3.3 建立顶层设计文件.....	71
3.3.4 工程文件的编译.....	73
3.3.5 模拟仿真.....	74
3.3.6 定时分析.....	77
3.3.7 器件编程.....	80

第4章 FLEX 6000 系列器件简介及应用实例

4.1 FLEX 6000 系列器件简介	83
4.1.1 特 点.....	83
4.1.2 概 述.....	85
4.1.3 功能描述.....	86
4.1.4 输出配置.....	98
4.1.5 JTAG 边界扫描	99
4.2 交通信号灯控制逻辑设计.....	99
4.2.1 系统要求分析.....	99
4.2.2 控制逻辑描述	100
4.3 电子钟的设计实例	102
4.3.1 功能要求和结构	102
4.3.2 控制芯片的设计	103

4.4 字符型 LCD(KS 0066)接口的设计	116
4.4.1 原理介绍	116
4.4.2 接口的 VHDL 描述	125

第 5 章 应用 FPGA 设计 PC 机 ISA 和 PCI 总线接口

5.1 应用 FPGA 设计 PC 机的 ISA 总线接口	134
5.1.1 ISA 总线概述	134
5.1.2 基于 Altera FLEX 6000 的 ISA 接口设计	138
5.2 应用 FPGA 设计 PC 机的 PCI 总线接口	148
5.2.1 PCI 总线概述	148
5.2.2 PCI 总线命令	153
5.2.3 PCI 总线访问地址解码	154
5.2.4 PCI 配置空间操作	154
5.2.5 基于 Altera FLEX 10K 的 PCI 接口设计	158

附录 A Altera 公司 FPGA/CPLD 系列器件纵览

附录 B MAX+PLUS II Windows 2000 驱动配置指南

附录 C FPGA/VHDL 快速学习开发工具系统 CHICAGO 1.0

附录 D FPGA/VHDL 快速学习开发工具系统 CHICAGO 2.0

参考文献

第 1 章 绪 论

1.1 FPGA/CPLD 概述

FPGA(现场可编程门阵列)与 CPLD(复杂可编程逻辑器件)都是可编程逻辑器件,它们是在 PAL 和 GAL 等逻辑器件的基础之上发展起来的。与以往的 PAL 和 GAL 等相比较, FPGA/CPLD 的规模比较大,可以替代几十甚至几千块通用 IC 芯片。这样的 FPGA/CPLD 实际上就是一个子系统部件。这种芯片受到世界范围内电子工程设计人员的广泛关注和普遍欢迎。经过了十几年的发展,许多公司都开发出了多种可编程逻辑器件,比较典型的的就是 Xilinx 公司的 FPGA 器件系列和 Altera 公司的 CPLD 器件系列。

尽管 FPGA 和 CPLD 与其他类型 PLD 的结构各有其特点和长处,但概括起来,它们是由三大部分组成的:二维的逻辑块阵列,构成了 PLD 器件的逻辑组成核心;输入/输出块,连接逻辑块的互连资源;连线资源,由各种长度的连线线段组成,其中也有一些可编程的连接开关,用于逻辑块之间、逻辑块与输入/输出块之间的连接。图 1.1 所示为典型的 PLD 的框图。

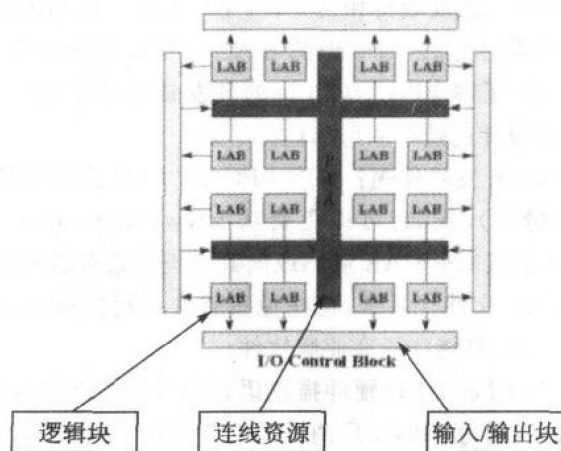


图 1.1 典型的 PLD 的框图

对用户而言, CPLD 与 FPGA 的内部结构稍有不同,但用法一样,所以,多数情况下不加以区分。

FPGA/CPLD 芯片都是特殊的 ASIC 芯片,除了具有 ASIC 的特点之外,还具有以下几个

优点:

① 随着超大规模集成电路 VLSI(Very Large Scale IC)工艺的不断提高,单一芯片内部可以容纳上百万个晶体管。FPGA/CPLD 芯片的规模也越来越大,其单片逻辑门数已达到上百万门,所能实现的功能越来越强,同时还可以实现系统集成。

② FPGA/CPLD 芯片在出厂之前 100%都做过测试,不需要设计人员承担投片风险和费用。设计人员只需在自己的实验室里就可以通过相关的软硬件环境来完成芯片的最终功能设计。所以, FPGA/CPLD 的资金投入少,节省了许多潜在的花费。

③ 用户可以反复地编程、擦除、使用,或者在外围电路不动的情况下,用不同软件就可实现不同的功能;因此,用 FPGA/CPLD 试制样片,能以最快的速度占领市场。FPGA/CPLD 软件包中有各种输入工具、仿真工具、版图设计工具及编程器等全线产品,使电路设计人员在很短的时间内就可完成电路的输入、编译、优化、仿真,直至最后芯片的制作。当电路有少量改动时,更能显示出 FPGA/CPLD 的优势。电路设计人员使用 FPGA/CPLD 进行电路设计时,不需要具备专门的 IC(集成电路)深层次的知识。FPGA/CPLD 软件易学易用,可以使设计人员集中精力进行电路设计,快速将产品推向市场。

1.2 FPGA/CPLD 的开发软件

由于 FPGA/CPLD 软件已经发展得相当完善,用户甚至可以不用详细了解 FPGA/CPLD 的内部结构,也能用自己熟悉的方法,如原理图输入或 HDL 语言来完成相当优秀的 FPGA/CPLD 设计。对初学者来说,了解 FPGA/CPLD 的开发软件和开发流程,了解 FPGA/CPLD 的内部结构,将有助于提高设计的效率和可靠性。

如何获得 FPGA/CPLD 开发软件呢?许多 FPGA/CPLD 公司都提供免费试用版或演示版(当然商业版大都是收费的),例如:可以免费从 www.altera.com 上下载 Altera 公司的 MAX+PLUS II (Baseline 版或 E+MAX 版),或向其代理商索取这套软件。以上免费软件都需要在网上注册申请 License 文件。通常,这些免费软件已经能够满足一般设计的需要。当然,要想软件功能更强大一些,只能购买商业版软件。

如果使用 VHDL 或 Verilog HDL 硬件描述语言来开发 FPGA/CPLD,通常还需要使用一些专业的 HDL 开发软件,因为 FPGA 厂商提供的软件的 HDL 综合能力一般都不是很强,需要其他软件来配合使用。

对于 FPGA/CPLD 产品,一般分为:基于乘积项(product-term)技术、EEPROM(或 Flash)工艺的中小规模 FPGA/CPLD;基于查找表(look-up table)技术、SRAM 工艺的大规模 FPGA/CPLD。EEPROM 工艺的 FPGA/CPLD 密度小,多用于 1 000 门以下的小规模设计,适合做复杂的组合逻辑,如译码。SRAM 工艺的 FPGA/CPLD 密度高、触发器多,多用于 10 000 门以上的大规模设计,适合做复杂的时序逻辑,如数字信号处理和各种算法。

1.3 FPGA/CPLD 的分类和使用

在 FPGA/CPLD 开发软件中完成设计以后,软件会产生一个最终的编程文件(如 .pof)。如何将编程文件烧到 PLD 芯片中去呢?

1. 基于乘积项技术

对于基于乘积项(product-term)技术、EEPROM(或 Flash)工艺的 PLD(如 Altera 公司的 MAX 系列、Lattice 公司的大部分产品及 Xilinx 公司的 XC 9500 系列),由厂家提供编程电缆,如 Altera 公司的 ByteBlaster。电缆的一端接在计算机的并行打印口上,另一端接在 PCB 板上的一个 10 芯插头上,如图 1.2 所示。FPGA/CPLD 芯片有 4 个引脚(编程脚)与插头相连。

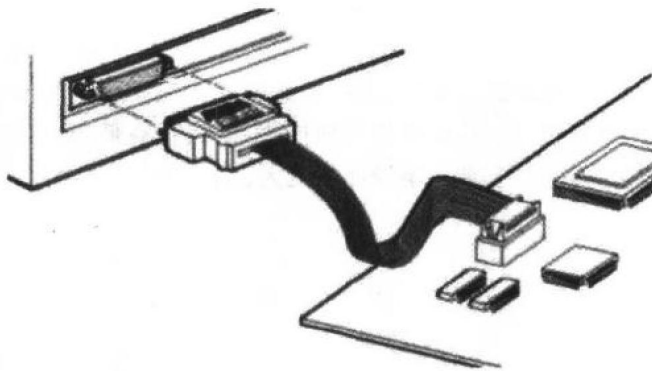


图 1.2 电缆连接示意图

编程电缆向系统板上的器件提供配置或编程数据,这就是所谓的在线可编程 ISP(如图 1.3 所示)。ByteBlaster 使用户能够独立地配置 FPGA/CPLD 器件,而不需要编程器或任何其他编程硬件。编程电缆可以向代理商购买,也可以根据厂家提供的编程电缆的原理图自己制作,成本仅需一二十元。早期的 FPGA/CPLD 是不支持 ISP 的,它们需要用编程器烧写;目前的 FPGA/CPLD 都可以用 ISP 在线编程,也可用编程器编程。这种 FPGA/CPLD 可以加密,并且很难解密。

2. 基于查找表技术

对于基于查找表(look-up table)技术、SRAM 工艺的 FPGA/CPLD(如 Altera 公司的所有 FLEX, ACEX, APEX 系列和 Xilinx 公司的 Spartan, Virtex),由于 SRAM 工艺的特点,掉电后数据会消失,因此调试期间可以用下载电缆配置 FPGA/CPLD 器件,调试完成后,需要将数据固化在一个专用的 EEPROM 中(用通用编程器烧写)。上电时,由这片配置 EEPROM 先对 FPGA/CPLD 加载数据,十几个毫秒后, FPGA/CPLD 即可正常工作(亦可由 CPU 配置 FPGA/CPLD)。但 SRAM 工艺的 FPGA/CPLD 一般不可以加密。



图 1.3 在线可编程过程

3. 反熔丝技术

还有一种反熔丝(anti-fuse)技术的 FPGA/CPLD, 如 Actel, Quicklogic 及 Lucent 公司的部分产品就采用这种工艺。用法与 EEPROM 的 FPGA/CPLD 一样, 但这种 FPGA/CPLD 不能重复擦写, 所以初期开发过程比较麻烦, 费用也比较高。但反熔丝技术也有许多优点: 布线能力更强, 系统速度更快, 功耗更低, 同时抗辐射能力强, 可耐高低温, 可以加密。所以它在一些有特殊要求的领域中运用较多, 如军事及航空航天。

第 2 章 VHDL 编程基础

2.1 VHDL 语言程序的基本结构

一个完整的 VHDL 语言程序通常包含实体(entity)、结构体(architecture)、配置(configuration)、包集合(package)和库(library)5 个部分。实体用于描述所设计的系统的外部接口信号;结构体用于描述系统内部的结构和行为;配置用于从库中选取所需单元来组成系统设计的不同版本;包集合存放各设计模块能共享的数据类型、常数和子程序等;库存放已经编译的实体、结构体、包集合和配置。

2.1.1 设计实体

设计实体是 VHDL 语言设计的基本单元,是最重要的系统抽象。它可以代表整个电子系统、一块电路板、一个芯片或一个门电路,既可以代表像微处理器那样复杂的电路,也可以代表像单个逻辑门那样简单的电路。

设计实体由实体说明和结构体两部分组成。实体说明部分规定了设计单元的输入/输出接口信号或引脚,而结构体部分定义了设计单元的具体构造和操作。

2.1.2 实体说明

设计实体是一个初级设计单元,可以单独编译并且可以被并入设计库。它给实体命名并给实体定义一个接口。接口信息用于与其他模块通信。实体说明是一个器件的外部视图,即从外部看到的器件的外貌,包括该器件的端口。实体说明也可以定义参数,并把参数从外部传入模块内部。

实体说明的一般格式如下:

```
ENTITY 实体名 IS
    [ GENERIC (类属参数说明); ]
    [ PORT (端口说明); ]
    实体说明部分;
[BEGIN
    实体语句部分; ]
END [实体名];
```

实体说明以“ENTITY 实体名 IS”开始至“END [实体名]”结束。这里大写字母表示实体说明的框架,是不可缺少和省略的部分。对 VHDL 而言,大写或小写都一样,不加区分,如:

```
ENTITY deco IS  
END deco;
```

1. 类属参数说明

类属和端口说明是实体说明的头,说明用于设计实体和其外部环境通信的对象、通信的格式约定和通信通道的大小。

类属为设计实体与外部环境通信的静态信息提供通道,用来规定端口的大小、实体中子元件的数目、实体的定时特性等等。类属参数说明必须放在端口说明之前,一般书写格式为:

GENERIC([CONSTANT] 名字表: [IN] 子类型标识 [:= 静态表达式],...);

如: GENETRIC(m; TIME; = 3ns)

2. 端口说明

端口为设计实体与外部环境的动态通信提供通道,是对基本设计实体与外部接口的描述。每个端口必须有一个名字、一种通信模式和一种数据类型。名字是该端口的标识符;模式说明数据通过该端口的流动方向;类型说明流过该端口的数据类型。其一般书写格式为:

PORT ([SIGNAL] 端口名: [模式] 子类型标识 [BUS] [:= 静态表达式],...);

因为关键字 PORT 后必须是信号类,故可略去关键字 SIGNAL。如果该端口和多个输出端相连,则要使用关键字 BUS。

(1) 端口名

端口名是赋予每个外部引脚的名称,通常用一个或几个英文字母,或用英文字母加数字命名。名称的含义最好与惯例一致,如 D 开头的端口名表示数据, A 开头的端口名表示地址。

(2) 端口模式

端口模式用来说明数据、信号通过该端口的方向。VHDL 语言提供了如下端口模式:

IN——流入实体;

OUT——流出实体;

INOUT——双向端口:既可流入,又可流出;

BUFFER——也是双向端口,但限定该端口只能有一个源;

LINKAGE——不指定方向,无论哪个方向都可连接。

(3) 数据类型

数据类型端口说明除了定义端口标识名称和端口外,还要标明出入端口的数据类型。在 VHDL 语言中有 10 种数据类型。VHDL 语言的 IEEE 1706—93 标准规定,EDA 综合工具提供的数据类型为布尔型(Boolean)、位型(Bit)、位矢量型(Bit_Vector)和整数型(Integer)。

例 2-1 只有端口说明的实体说明。

```

ENTITY deco IS
    PORT( clock: IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
          q:  OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
    );
END deco;

```

例 2-2 有类属和端口说明的实体说明。

```

ENTITY and2 IS
    GENERIC(rise,fall: TIME);
    PORT(a,b: IN BIT;
          c: OUT BIT
    );
END and2;

ARCHITECTURE behav OF and2 IS
    SIGNAL internal: BIT;
BEGIN
    internal<= a AND b;
    c<=internal AFTER (rise) WHEN internal='1' ELSE
        internal AFTER (fall);
END behav;

```

2.1.3 结构体

结构体是次级设计单元,在其对应的初级设计单元实体说明被编译并被并入设计库之后,它就可以单独地被编译,并且被并入该设计库之中。结构体描述一个设计的结构或行为,把一个设计的输入与输出之间的关系建立起来。一个设计实体可以有多个结构体,分别代表该器件的不同实现方案。用 VHDL 语言描述结构体功能有 3 种方法:

- ① **行为描述法** 采用进程语句顺序描述设计实体的行为。
- ② **数据流描述法** 采用进程语句顺序描述数据流在控制流作用下被加工、处理和存储的全过程。
- ③ **结构描述法** 采用并行处理语句描述设计实体内的结构组织和元件互联关系。

结构体的一般书写格式为:

```

ARCHITECTURE 结构体名 OF 实体名 IS
    ——定义语句

```

```
BEGIN
    —— 并行处理语句
END [结构体名];
```

如:

```
ARCHITECTURE ver3 OF deco IS
BEGIN

    WITH clock SELECT

        q<= "00111111" WHEN "0000",
            "00000110" WHEN "0001",
            "01011011" WHEN "0010",
            "01001111" WHEN "0011",
            "01100110" WHEN "0100",
            "01101101" WHEN "0101",
            "01111101" WHEN "0110",
            "00100111" WHEN "0111",
            "01111111" WHEN "1000",
            "01101111" WHEN "1001",
            "00000000" WHEN OTHERS;

END ver3;
```

1. 结构体名称的命名

结构体的名称可自由命名,但通常把结构体的名称命名为 behavioral(行为)、dataflow(数据流)或者 structural(结构)。这 3 个名称实际上是 3 种结构体描述方式的名称。当设计者采用某一种描述方式来描述结构体时,该结构体的结构名称就命名为那个名称。这样,使得阅读 VHDL 语言程序的人能直接了解设计者采用的描述方式。命名格式如下:

```
ARCHITECTURE behavioral OF mux IS    —— 用结构体的行为命名;
ARCHITECTURE dataflow OF mux IS      —— 用结构体的数据流命名;
ARCHITECTURE structural OF mux IS    —— 用结构体的结构命名。
```

2. 定义语句

定义语句位于 ARCHITECTURE 和 BEGIN 之间,用于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。须注意的是,这是结构体内部,而不是实体内部,实体中可能有几个结构体。实体说明中定义 IO 信号为外部信号,而结构体定义的信号为内部信号。

结构体的信号定义与实体的端口说明一样,应有信号名称和数据类型定义,但不需要定义信号模式,不用说明信号方向。

例 2-3 计数器的 VHDL 描述。

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;

ENTITY cnt10 IS
PORT (clock: IN STD_ULOGIC;
      q: OUT STD_ULOGIC_VECTOR(3 DOWNTO 0);
      ca: OUT STD_ULOGIC);
END cnt10;

ARCHITECTURE rtl OF cnt10 IS
    SIGNAL qs:STD_LOGIC_VECTOR(3 DOWNTO 0);
BEGIN
    PROCESS(clock)
        VARIABLE q10:INTEGER:=0;
    BEGIN
        IF (clock'EVENT AND clock='1') THEN
            IF(q10 =9) THEN
                q10:=0;
                ca<='1';
            ELSE
                q10:=q10+1;
                ca<='0';
            END IF;
        END IF;
        qs<=CONV_STD_LOGIC_VECTOR(q10,4);
        q<=TO_STDULOGICVECTOR(qs);
    END PROCESS;
END rtl;
```