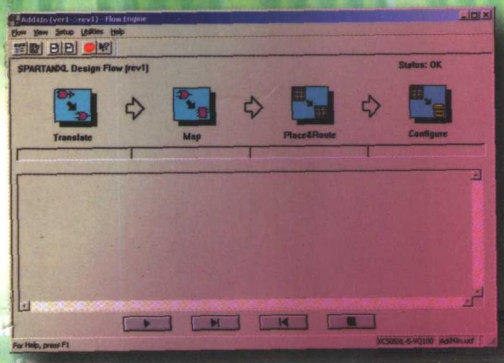
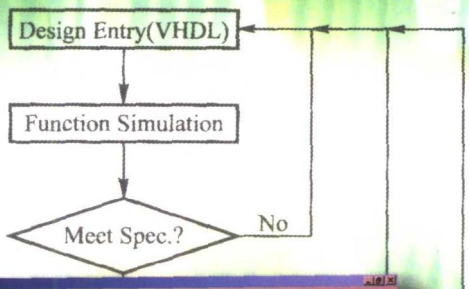
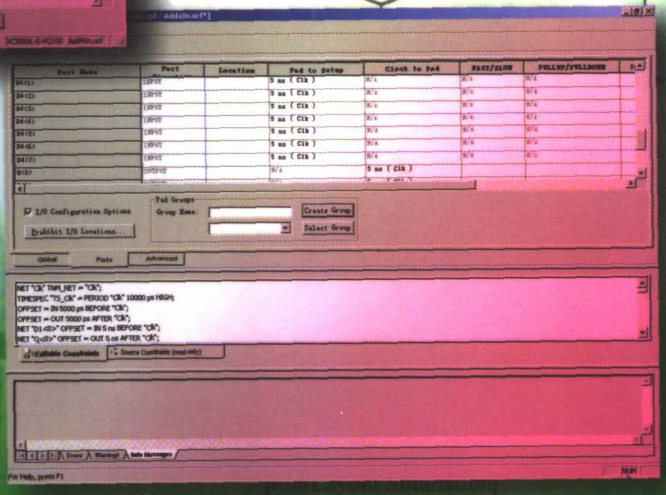
VHDL & FPGA Desgin

VHDL与FPGA设计

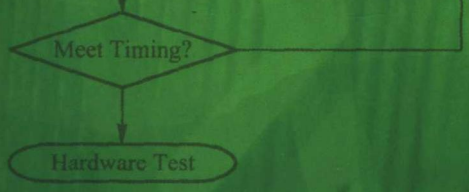
胡振华 编著



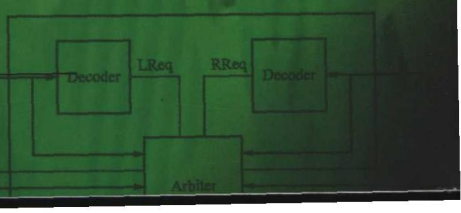




```
ENTITY E IS
5   PORT(D : IN STD_LOGIC;
         CLK : IN STD_LOGIC;
         RST : IN STD_LOGIC;
         OE : IN STD_LOGIC;
         Q : OUT STD_LOGIC);
10  END E;
ARCHITECTURE A OF E IS
    SIGNAL Q_INT : STD_LOGIC;
BEGIN
15  PROCESS(RST,CLK)
    BEGIN
      IF RST = '0' THEN
        Q_INT <= '0';
      ELSIF CLK = '1' AND CLK'EVENT THEN
        Q_INT <= D;
      END IF;
    END PROCESS;
    Q <= Q_INT WHEN OE = '0' ELSE
      'Z';
```

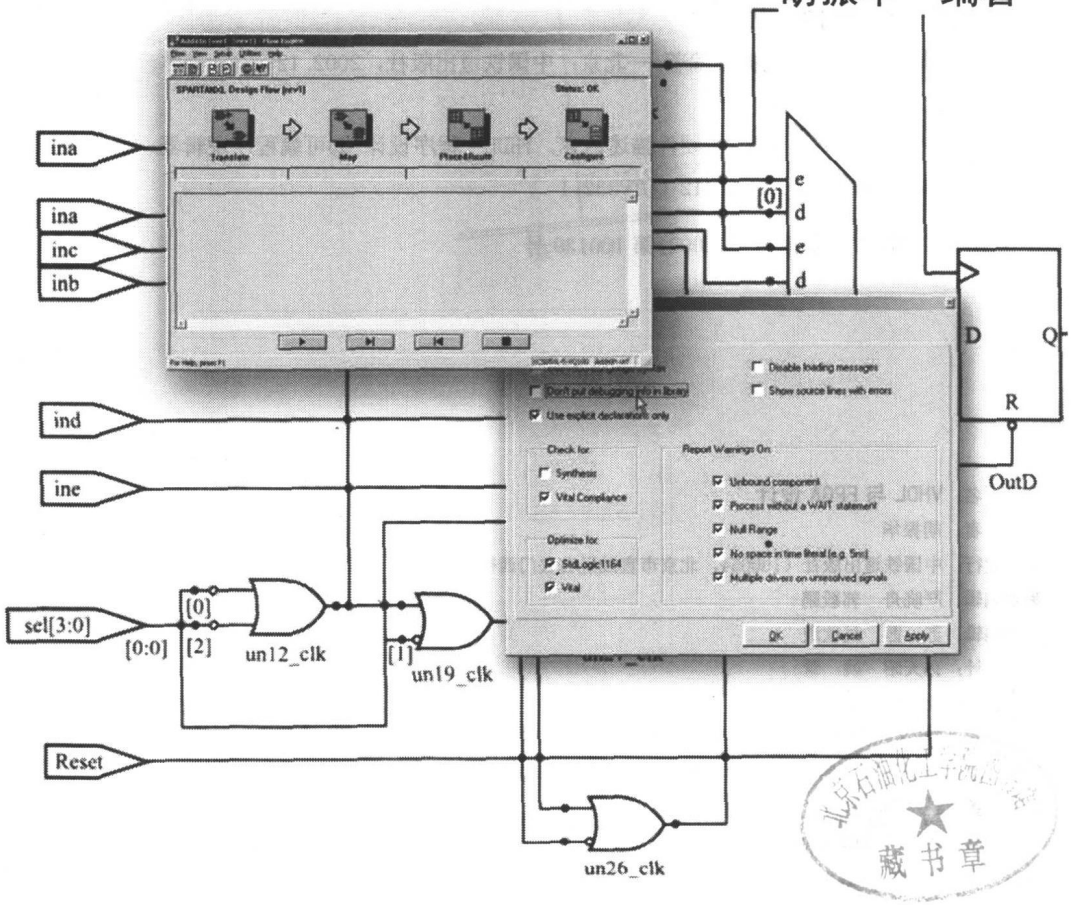


中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE



VHDL与FPGA设计

胡振华　编著



中国铁道出版社

2003・北京

（京）新登字 063 号

北京市版权局著作权合同登记号：01-2002-3959 号

图书在版编目（CIP）数据

VHDL 与 FPGA 设计/胡振华编著. —北京：中国铁道出版社，2002.12

ISBN 7-113-05046-8

Ⅰ.V… Ⅱ.胡… Ⅲ.①硬件描述语言，VHDL—程序设计 ②可编程序逻辑器件—基本知识 Ⅳ.①TP312 ②TP332.1

中国版本图书馆 CIP 数据核字(2002)第 100139 号

书　　名：VHDL 与 FPGA 设计
作　　者：胡振华
出版发行：中国铁道出版社（100054，北京市宣武区右安门西街 8 号）
策划编辑：严晓舟　郭毅鹏
责任编辑：苏　茜　吴秋淑
封面设计：孙天昭　姚　蕾
印　　刷：北京兴顺印刷厂
开　　本：787×1092　1/16　印张：20.5　字数：481 千
版　　本：2003 年 1 月第 1 版　　2003 年 1 月第 1 次印刷
印　　数：1～5000 册
书　　号：ISBN 7-113-05046-8/TP・837
定　　价：32.00 元

出版说明

VHDL 语言是一种非常流行的硬件描述语言。而 FPGA 设计已成为电子设计的主要手段。随着集成电路产业以及通信产业在我国的迅速发展，学习 VHDL 语言和 FPGA 设计是大中专在校生和工程技术人员的迫切需求。虽然国内关于VHDL语言的图书很多，但是真正把VHDL和 FPGA 设计捆绑在一起，通过大量实例来讲述的图书却是很少。

对于读者来说，单单学 VHDL 语言，只能是局限在纸面上，不能和实际应用结合起来。学习起来体会不深刻，相对难度也大。本书做到了既有理论又有实践。真正将 VHDL 语言和 FPGA 设计两者结合在一起去讲，即深入浅出地讲述了 VHDL 程序设计方法，又结合 Modelsim、FPGA Express 和 Xilinx Foundation 软件详细讲述了 FPGA 设计的功能仿真、综合、布局布线和时序仿真。

本书的另一个特色是从 FPGA 设计的角度出发，剖析了 VHDL 语法的特点以及它们的正确使用方法，将初学者在运用 VHDL 语言进行 FPGA 设计中会遇到的疑惑，一一点拨清楚。并结合作者的多年 FPGA 设计经验，讲述了许多 EDA 设计思想，并贯穿全书始终。

出于上述原因，中国铁道出版社计算机图书项目中心引进了该书的版权，本书由苏国彬、杨柏松和高海茹翻译，最后由苏国彬统稿。陈贤淑、陈晓娟、廖康良参与了本书的编排工作。

由于时间仓促，书中错误在所难免，恳请广大读者批评指正。

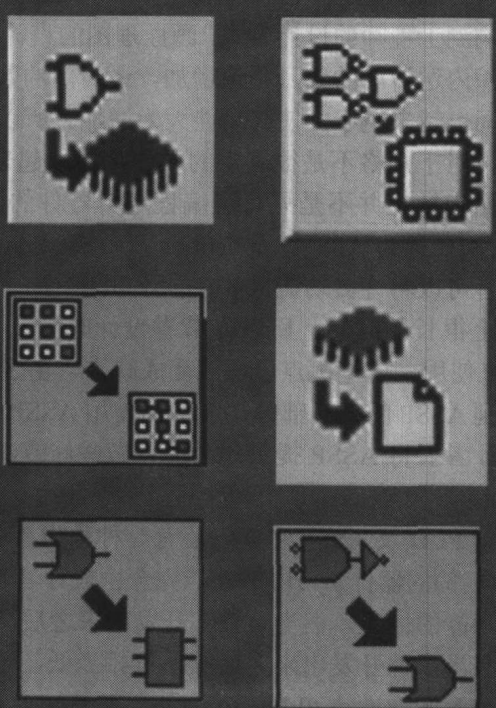
中国铁道出版社

2002 年 12 月

目　录

CHAPTER 0

前言



0-1 VHDL 的发展

VHDL 是 VHSIC(Very High Speed Integrated Circuit) Hardware Description Language 的缩写，最早是美国国防部为描述电子电路所开发的一种语言，因此早期的 VHDL 是用在文件上的。它可以很容易地描述一个系统的功能，也可以详细描述一个器件的状态。在 70 年代，大多数的数字逻辑电路还是以分散的 TTL (Transistor-Transistor Logic)器件所组合的。就当时的情况而言，工程师们必须了解各种逻辑器件的特性，再将逻辑器件组合成电路图。一个简单的逻辑电路，也许需要数十个分散的 TTL 器件组合成一块电路板。加上当时的器件都是以 DIP(Dual Inline Package)的方式封装，一块电路板满是一排排的器件，看起来是壮观无比。

80 年代后期出现了可编程逻辑器件(PLD，Programmable Logic Device)，其设计的切入点(Design Entry)有两种，一种是采取画原理图的方式，这就像是原先将分散的 TTL 逻辑组合成电路板一样，只是将电路板变成了一颗 IC。另一种设计方式是以简单的方程式来表现逻辑架构，经过编译器产生刻录文件，再写到可编程逻辑器件中。这种器件的最大好处是可以将数种不同的逻辑放在一颗 IC 中。假设一个设计中使用到一个 AND gate（与门）、一个 OR gate（或门）和一个 Inverter（反相器），要是使用分散的 TTL 器件，电路板上就要使用三个 IC。但如果使用可编程逻辑器件，只要一颗 IC 就能将这三种功能放进去了。不过由于其容量并不是很大，因此所能容纳的功能也不是很多，所以电路板上的 IC 数目仍然很多。

随着技术的发展，90 年代初期出现了 FPGA(Field Programmable Gate Array)，这是一种比较复杂的可编程逻辑器件。但当时的 FPGA 其逻辑门数目(gate count)并不是很多，因此并没有影响到工程师的设计方式，画原理图的设计方式仍是最流行的。随着半导体制程的演进，单位面积内逻辑门的数目急速增加。逻辑门的数目增多，半导体线径的宽度变细，逻辑门的延迟时间也越来越小，使其功能大为增加。所以 FPGA 使用的范围也就大了许多。加上其弹性极大，对于规格不是很确定的设计，使用起来相当方便。

当然市场上并不是只有可编程逻辑器件的出现，许多大厂乐于将固定功能设计成一颗 IC，这种 IC 称为 ASSP(Application Specific Standard Product，专用标准产品)。ASSP 的好处是设计者可以将许多功能都放在一颗 IC 里面，一般设计者要将这么复杂的功能设计出来，可能得花上很长的时间。所以在节省设计时间及降低风险的情况下，许多设计者会直接选择 ASSP 来使用。但是优点也可能变成缺点，由于 ASSP 的功能都已经确定了，如果有些设计中只用到 ASSP 的一小部分功能但仍使用 ASSP 做设计的话，则只有将整颗 IC 放在电路板上。或是设计者要将 ASSP 提供的功能再做些修改，由于 ASSP 缺乏弹性，在某些设计中也会无法应用。

除了 PLD 及 ASSP 之外，还有一种称为 ASIC(Application Specific Integrated Circuit，专用集成电路)的器件。其内部功能完全由设计者自行决定，因此不会出现使用 ASSP 做设计浪费许多功能的情况。但当 ASIC 开发出来之后，就和 ASSP 一样，仅能完全满足设计者的要求。而且当器件开发出来之后其单价相当低，这是 ASIC 的最大优点。不过其仍有缺点，由于其设计时间与设计 PLD 相当，还要额外加上光罩制作等过程，所以其设计的时间最长。而

且开发出来的 ASIC 同样没有弹性。如果规格修改或是功能上需要改进，只好再花大把的钱重新开发了。虽然上面三种产品各有其市场，但是上市时间一直是一个重要因素，而且设计的复杂度越来越大。因此设计 PLD 或 ASIC 时，传统的画原理图方式也就越来越不能满足设计者的需求。加上逻辑合成器的日趋成熟，高级的硬件描述语言(如 VHDL 及 Verilog HDL)的设计方式已逐渐取代画原理图的方式了。

0-2 VHDL 的优点

使用硬件描述语言从事设计工作有以下几点好处：

1. 以硬件描述语言从事设计工作，不需要考虑线路的布局问题，设计之初只需要一个好用的仿真软件即可，因此可降低设计时的复杂度。
2. 硬件描述语言比较接近算法的推演，而画原理图的设计方式需由算法转换成硬件概念，再将硬件概念转变成一个个硬件单元，因此硬件描述语言可减少设计周期，加快商品的上市时间。譬如一个补码的转换电路，相信学过基本数字逻辑设计的人都知道，只要将待转换数值的每个位反相，再将结果加 1 就可以了。但对于一个对数字设计不是很有经验的设计者，要以逻辑电路完成补码转换电路，相信必定要一段相当长的时间才行。
3. 硬件描述语言并没有固定的目标器件，硬件描述语言可以用来做 ASIC，也可以用来做 FPGA 或 CPLD（Complex Programmable Logic Device，复杂可编程逻辑器件）。但使用画原理图的设计方式，其使用的器件种类（或厂商）不同，就必须使用不同的 library（库）来设计。因此在选择目标器件方面，硬件描述语言的设计会有较大的弹性。

或许 VHDL 的出现比 Verilog HDL 晚，使得在 ASIC 的发展上仍是以 Verilog HDL 为主流。VHDL 也因此采取较开放的方式，并在 1987 年通过成为 IEEE(International Electrical & Electronic Engineer)的标准 IEEE-1076。

0-3 所须具备的概念

有人曾说，即使没有硬件概念的人也可以用 VHDL 来做设计，但我个人并不认为如此。没有硬件概念的人在做 VHDL 的设计时，往往会把它当做一般的软件语言来做设计，结果就是写出了软件式的 VHDL。当然，将 VHDL 用做文件的描述本来也就是其功能之一。但是如果要达到的目标是做出实际的成品，如 FPGA、PLD 或 ASIC，如果没有硬件逻辑概念所做的设计，往往会迁就语言本身的语法及自己所要达到的目标，而写出不合“逻辑”的程序。

至于说要不要有软件的设计经验，或是了解某种软件语言的语法呢？我想这并不是绝对的，但对某种软件语言的语法有些许的了解也是件好事。像在 C++的语法中有重载(Overload)的特性，也就是不论 function 或 operator，其 function name 或 operator 可以相同，但代表的却是不完全相同的 function 或 operator，只要它所定义的 argument 有所不同就可以了。譬如你要一个计算最大值的 function，在 C++语言中可以将其定义为：

```
int Max(int a, int b);
double Max(double c, double d);
```

当程序中调用 function Max 时，到底会使用到以上两个 function 中的哪一个呢？我们先看看以下的例子：

```
int x,y,o;
o = Max(x,y);
```

由于定义的 function 名称都是 Max，compiler 怎么会知道要调用哪一个呢？在上面的例子中，x、y、o 都是整数，因此在：o=Max(x,y);语句中会调用到的 function 就是之前所定义的 int Max(int a, int b)，而非 double Max(double c, double d)。

在 VHDL 中也有同样的作用，以 IEEE library 中的 NUMERIC_STD package 而言，其中的 function "+" 就有六种不同的接法，分别是：

1. 左右边都接 UNSIGNED
2. 左右边都接 SIGNED
3. 左边接 UNSIGNED，右边接 Natural
4. 左边接 Natural，右边接 UNSIGNED
5. 左边接 SIGNED，右边接 Integer
6. 左边接 Integer，右边接 SIGNED

等六种。当你使用到 function "+" 时，只要你的数据类型符合以上六种情况之一，compiler 都能为你找到合适的 function。

所以说有些软件的概念对于硬件描述语言的了解也是好的。但在另一方面有了软件的概念却不见得是件好事，因为硬件描述语言有大半的用途是在制造硬件，它不像软件一样会依照顺序来执行。当电路板一上电后，器件内的所有信号都会开始动作，这绝不像软件一样照着程序的顺序来执行。如果你仍然按照软件的思考方式，你可能会写出一个很像软件的硬件描述语言程序，那么在做成硬件之后也许会产生一个令你觉得可怕的设计结果。所以说在做硬件设计时是该软的时候要软，该硬的时候一定要硬。

0-4 "SRAM Base" vs. "Anti-Fuse"

FPGA 可分为两大类别：SRAM Base 及 Anti-Fuse。SRAM Base 的 FPGA 本身就像是一个 SRAM，在开机后其必须经过一个称为 configuration(配置)的过程。Configuration 可将设计加载到电路板上的 FPGA 中，其方式不外乎是通过各家厂商提供的特殊 download cable（下载电缆），或是在 FPGA 旁边加上一颗 RPOM。但在关机之后，FPGA 的内容就消失，直到下次开机重新 configuration 后，FPGA 才会恢复正常功能，像 Xilinx、Altera 及 Lucent 等。另一种称为 Anti-Fuse 的 FPGA，其本身则像一个 fuse-array，将设计载入 FPGA 的过程一般称为刻录，因为它是将 FPGA 内的 fuse 烧断。因此，当执行过这个动作之后，FPGA 的功能就固定了，即使掉电后重新再上电，其功能仍然存在。

由上面的讲述可知：在设计阶段应该尽量选用 SRAM Base 的 FPGA。虽然在设计阶段我们做了 function simulation 及 timing simulation，但是设计者用来做仿真的 testbench 不见得能

包含电路上所有的情况。如果有例外情况发生，而使用的又是 Anti-Fuse 的 FPGA，那么设计者的 FPGA 就报销了。但如果使用 SRAM Base 的 FPGA，只要重新 configuration 一次就好了。即使已经使用 PROM 来做 configuration，重刻一颗 PROM 的代价应该会比重刻一颗 FPGA 低得多。

然而 SRAM Base 的 FPGA 并不是没有缺点，由于每次开机时都要做 configuration，因此会占用一些时间。这些时间对人而言可能很快，但对电子电路而言可能就很长了。而且在做 configuration 时，所有的 IO 都处在状态不明的情况下，若有些信号需要在一上电时就有确定的逻辑状态，譬如说 Reset 信号，那就不适合用 SRAM Base 的 FPGA 来产生了。

0-5 本书的内容

在本章中我们介绍了 VHDL 发展的背景及器件演进的过程，并介绍使用 VHDL 的优点。我们并不是要回溯到真空管或晶体管的时代，只是从基本的标准 TTL 逻辑谈起，看看当时的设计方式是怎么样的。在简单的可编程逻辑器件出现后，设计方式的改变是什么。到目前复杂的可编程逻辑门阵列出现，与特殊应用标准器件及特殊应用集成电路各占有不同的领域。虽然本书的重点是在可编程逻辑器件的设计，但是特殊应用集成电路的设计流程也是相似的。因此在了解整个设计流程后，相信要开发可编程逻辑器件或是专用集成电路都不会有太大的问题。

第 1 章是一个概述，其主要目的是将可编程逻辑器件的设计及验证工作做一简单的介绍。

第 2 章是将 VHDL 程序的架构做一声明，由于接下来就会接触到仿真的工作，因此在本章一开始会介绍本书所使用到的仿真器。之后我们会介绍一个简单的基本架构，其中包含 library 及 use 的使用，对 entity 的定义和 architecture 的设计也都做了一番声明。最后介绍的是 package 的使用，其使用的方式相当广，像是衍生的数据类型、子程序和常数值，都能在 package 内定义。

第 3 章介绍的是 VHDL 中的数据类型。首先介绍的是 VHDL 的 standard package，其中包含了标量类型和枚举类型。之后介绍的是较复杂的数据类型，包括数组，记录和文件类型。如果 VHDL 提供的基本类别不够使用，还可以定义自己想要的数据类型。

第 4 章的主题是运算符，首先我们要讲的是在 IEEE 1076-1987 与 IEEE 1076-1993 所定义的运算符的差异。接下来将这些运算符分成七大类，分别是逻辑运算符、关系运算符、移位运算符、加减运算符、符号运算符、乘法运算符和其余无法分类的运算符。运算符的使用相当方便，许多不同数据类型的信号也可以用相同的运算符做处理，这就是运算符重载所带来的便利，本章中也会针对这一点进行声明。

第 5 章和第 6 章是在介绍数字逻辑中的两大类——组合逻辑(combinational logic)和时序逻辑(Sequential logic)。在这两章的一开始将介绍基本的逻辑语法，接下来结合几个实际的例子进行声明，因为结合例子来分析语法比较容易理解。

第 7 章要介绍的是函数(function)和过程(procedure)的定义和用法。一开始将比较 function 和 procedure 的基本差异，接下来会说明 function 的用途和定义方式。Function 有个重要的用途，就是做数据类型的转换，本章会对此进行声明。接下来还介绍重载函数(overload function)

的使用，最后再讲解 procedure 的定义和使用方法。

第 8 章的内容是属性(attribute)和配置(configuration)。属性的使用有的是设计上所必须的，有些只是使设计有更大的弹性或是更方便阅读。配置的使用像是一种动态的连结，当 entity 定义完毕之后，设计者可以有不同的架构（architecture），而在做仿真前再将特定的 architecture 连结到 entity 上。配置也可以用在 generic 的参数传递上，这些都会在第 8 章中介绍到。

从第 2 章到第 8 章所介绍的都是 VHDL 语法的一部分。从第 9 章开始已经与基本的语法没有太大的关联，而只是些技巧上的声明。

第 9 章的内容是介绍层次式的设计。随着设计复杂度的增加，程序的设计已经不再只是简单地将所有的设计放在一个程序中。在设计初期先将整个设计分成小的部分，当小的方块设计完毕验证后再组合成完整的设计。

第 10 章介绍的是功能仿真(function simulation)，这可能是设计过程中最花时间、重复次数最多的地方。首先介绍的是 ModelSim 仿真器的环境介绍，当然仿真时会要求设计者输入某些信号的波形，有些信号是设计者必须观察的，所以要介绍宏命令的使用。如何将这些命令组合成一个需要的输入波形送入设计以得到验证，是本章的重点之一。当有大批数据要输入设计时，宏的使用已经无法满足设计者的需求，我们会介绍 testbench（测试平台）的建立及使用。这能使得大批数据送入设计后，设计者不需要再花许多时间来对比仿真器的波形就可以知道其结果是否正确了，而且用计算机进行对比的结果比用肉眼进行对比的可信度要高得多。

第 11 章所要介绍的是合成(synthesis)，这是将设计者的 VHDL 程序转换成特定器件架构的动作。逻辑合成的过程是完全自动的，合成器根据设计者所拟定的约束条件(constraint)，将 RTL(Register Transfer Level，寄存器传输级)的 VHDL 代码转换成逻辑门级(gate level)的布线。首先介绍的是合成器的使用，本书所介绍的合成器是 Synopsys 公司的 FPGA Express。接着介绍如何在做完合成后验证其结果，并会枚举一些语法上正确，但是不能做逻辑合成的例子。之前也曾提过要引导合成器达到设计者的需求，就要为合成器定义 constraint，所以本章也会介绍如何定义 constraint。在今天，设计往往不是一两位设计者就能完成的，有许多厂商已经将一些成熟的设计提供出来，使得 PLD 的设计者能采用这些现成的设计，很快地完成其工作，这些成熟的设计我们称为知识产权(Intellectual Property)，简称 IP。在本章中也会介绍如何将使用 IP 的设计拿来做逻辑合成。最后会介绍层次式设计的逻辑合成，相信在读完这一章后，对逻辑合成应该没有太大的问题了。

第 12 章所要介绍的是将逻辑合成的结果对特定的目标器件做布局及布线(place & route)。首先会介绍到 place & route 工具的使用，接着会声明在 place & route 工具中定义 constraint 的方法。当 place & route 的工作做完之后，可以从一些相关的报告中得知设计是否符合要求，所以本章也会介绍相关报告的解读方法。最后介绍层次式设计的 place & route。

第 13 章所要介绍的是时序验证的仿真(timing simulation)。本章以前一章做完 place & route 工作的设计为例，设定 place & route 工具即可产生逻辑门级的 VHDL 程序。在做时序仿真时先将之前做功能仿真的宏作为输入信号，在发现问题之后再做宏的修正。除了针对 timing 出问题时的查找方法做介绍外，我们还对 SDF(Standard Delay Format)文件的内容做了声明。这个文件记载了逻辑门级的 VHDL 程序中，每个器件的延迟时间。当了解 SDF 文件中各个参数的意义后，才能使设计者更顺利地查找时序上的问题。本章还介绍了如何修改输

入信号的时序，以得到正确的输出信号，并根据修改后的时序制定出设计的规格。最后介绍的是如何利用 VHDL 中的 textio 进行大批数据的比较，使用该种方法即可验证时序仿真的结果是否与功能仿真的结果一致。

第 14 章的主题是状态机(state machine)的设计。本章以一个实际的设计为例子，声明如何建立一个状态转移图，再根据状态转移图进行设计。介绍了当设计不完整，必需要加入新的状态时，如何修改程序的方法。当程序编写完成后，会将整个设计流程再执行一次，一则确保设计的正确性，再则也能令读者复习介绍过的章节内容。

第 15 章介绍的是并行处理。在今天一切都讲究速度，系统的时钟(clock)速度也一再攀升，这时许多动作已经不能在一个时钟周期内完成了，就需要并行处理了。本章以一个乘法器为例，并介绍三种不同的设计方式，分别是

1. 仅在输出端加上触发器(flip-flop)
2. 在输入及输出端都加上触发器
3. 除在输入及输出端都加上触发器外，内部再加上一层触发器

读者在看完这些设计修改的方法，就能了解并行处理的意义了。除此之外我们还介绍了将已有的设计转成并行处理的方法及所要注意的事项。最后剖析了并行处理的缺点，设计者可以由此了解并行处理所要付出的代价。

0-6 使用工具

本书中所使用的工具如下：

1. Simulator：Mentor Graphic ModelSim SE PLUS 5.5c
2. Synthesizer：Synopsys FPGA Express
3. Place & Router：Xilinx Foundation 2.1i