循序渐进实例化学习Verilog HDL

XUNXUJIANJIN SHILIHUA XUEXI VERILOG HDL

主　编　李　明　张　黎　刘亚秋
主　审　宋文龙

東北林業大學出版社

循序渐进实例化学习 Verilog HDL

主　编　李　明　张　黎　刘亚秋
主　审　宋文龙

東北林業大學出版社
·哈尔滨·

图书在版编目（CIP）数据

循序渐进实例化学习 Verilog HDL / 李明，张黎，刘亚秋主编. —哈尔滨 ：东北林业大学出版社，2014.8
（东北林业大学优秀教材丛书）
ISBN 978-7-5674-0498-4

Ⅰ. ①循… Ⅱ. ①李…②张…③刘… Ⅲ. ①VHDL 语言-程序设计-高等学校-教材 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字（2014）第 207791 号

责任编辑：任兴华
封面设计：乔鑫鑫
出版发行：东北林业大学出版社
（哈尔滨市香坊区哈平六道街 6 号　邮编：150040）
印　　装：哈尔滨市石桥印务有限公司
开　　本：787mm×960mm　1/16
印　　张：15.5
字　　数：276 千字
版　　次：2014 年 8 月第 1 版
印　　次：2014 年 8 月第 1 次印刷
定　　价：35.00 元

前　言

作者从事 EDA 教学和 Verilog HDL 教学很长时间了，用过很多相关的教材。市场上出现的很多书籍大多数是面向开发设计人员的，真正面向高校的、循序渐进地结合 FPGA 讲述 Verilog HDL 的优秀书籍并不多。因此，作者根据近 10 年的教学经验，站在初学者角度，结合初学者能接受的程度，编写了适合初学者学习的 Verilog HDL 教材。

本书适合具有数字电路技术基础和 C 语言课程基础的、大二下学期或者大三上学期的本科生，因为这两门课程构成了 RTL 级 Verilog HDL 学习的基础。Verilog HDL 主要是用于描述数字电路的，所以也能起到对数字电路知识的巩固作用。但是它绝不仅仅是数字电路的单纯表示，它在数字电路基础上有一种层次性的提高。从基本的数字电路分析提升到数字电路系统的设计。书中不仅安排了循序渐进的基本电路模块设计，而且在此基础上逐步进入 CPU 设计等系统级的电路设计。

本书不像一般市面上的书籍那样对 Verilog HDL 的语法用单独的一章来讲解，而是将语法贯穿于整个实例化的过程中，让读者能够在充分了解 Verilog HDL 描述电路功能的同时轻松掌握 Verilog HDL 语法。另外，在有些较模糊的地方使用了 RTL 电路对比方式，更容易使读者掌握并熟练使用 Verilog HDL。由于 Verilog HDL 主要用来做 IC（集成电路）设计，容易出现很多“生”的名词，本书中尽可能避免其集中出现，以免让读者感到“茫然”，待读者入门之后再参考其他相关书籍进行学习。

本书共分为 10 个部分。第 1 章主要讲述 PLD 的基本结构；第 2~4 章为 Verilog HDL 的相关知识点，通过实例引入方式讲授；第 5 章中涉及一些常见的实用电路；第 6 章进行一个简单的 CPU 设计，从原理到设计实现均详细讲解，是本书的特色；附录 1 为推荐的 HDL 编程风格，尤其是现阶段很多本科生的编程风格很不好，需要改正；附录 2 为 Quartus Ⅱ 软件仿真使用说明，便于初学者对书中的实例进行软件仿真验证；附录 3 是基于 DE2-115 的实验入门教程，初学者可以根据此教程一步一步地在开发板上实现设计的电路；附录 4 中对 Quartus Ⅱ 中引脚分配的方法进行了详细说明，为后续大型的工程项目奠定软件操作基础。李明编写了第 1 章、第 4~5 章和附录 1~4；张黎编写了第 2，3，6 章。全书由宋文龙教授审校。

本书的特点：①需要用到的基础知识少，具有数字电子技术基础，会使用 Windows 软件和操作即可。②语法简单，易于掌握，内容实用，语法的学习蕴含于实例中，这样才会彻底掌握语法的精髓。③循序渐进，实例化。④应用性强，其中的一些应用设计可以直接应用在工程上，很多实例稍加修改也可以应用。

使用本书要了解 EDA 技术的两个特点：①数字化（硬件载体为“可编程逻辑”器件）。一般来说，模拟电路能做到的，数字电路也能做到（除了微弱信号和高频段信号），很多模拟电路做不到的设计，数字电路也能做到。当然，实际的电路设计应该从成本、功能、功耗等各方面考虑，不一定要全部使用“数字”的。②软件化：在早期（20 世纪 70~80 年代）的电路设计中，工程师具有严格的分工，即硬件工程师只负责硬件设计，软件工程师只负责程序的编写。20 世纪 90 年代左右提出一个名词“软硬不分家”，强调软硬件协同设计，软件编程的责任也部分地落到硬件工程师头上。而现阶段，随着可编程逻辑器件的出现，更符合硬件设计的应该是“硬件设计软件化”！以美国为首的各大厂商推出了可编程逻辑器件（PLD）之后，硬件的设计几乎就变为软件的编程，用代码描述、鼠标轻松点击几下，一个新的电路系统就完成了！

学习本书的要点：一定要能勾画出 Verilog HDL 语言所能生成的电路的构成形式，而不是理解硬件描述语言本身的语法。第一节课内容不宜过多，不要大篇幅介绍各个 EDA 公司及其软件，那样会使学生产生恐惧和抵触心理，担心内容太多而学不过来。后期教学过程中会发现只用其中的一款软件入门就可以了，入门后有精力的同学可以再去研究其他的工具和 EDA 相关内涵。

本书适合作为电子信息工程专业（系）本科生的教材，可以作为没有 HDL 基础的计算机组成原理的教材或参考书，也可以作为广大电子爱好者学习的 Verilog HDL 入门教材。

由于作者水平有限，书中难免有错误和不足之处，恳请读者批评指正。作者邮箱：liming_ nefu@ nefu. edu. cn。

李 明

2014 年 4 月

目　录

1 EDA 技术简介

本章主要讲授 EDA 技术的基本概念和可编程逻辑器件的工作原理。

本章内容：EDA 技术简介

可编程逻辑器件简介

可编程逻辑器件基本的工作原理

EDA 是电子设计自动化（Electronic Design Automation）的缩写。狭义上讲，电子设计自动化特指以计算机为工具，设计者用硬件描述语言作为设计的输入文件，然后由计算机自动地完成逻辑编译、化简、分割、综合、优化、布局、布线和仿真，直至完成特定目标芯片的适配编译、逻辑映射和编程下载等工作。简而言之，设计者只需要编写硬件描述语言，借助于 EDA 技术就可以在芯片上实现所需的逻辑功能电路。广义上说，使用计算机完成电子设计的辅助工作都可以称之为电子设计自动化，不仅包括狭义的范畴，还包含计算机辅助设计，如 PCB 工具 Protel（最新版本名称为 Altium Designer 等），计算机辅助电路仿真工具软件 Multisim 和 Proteus 以及一些计算机辅助制造等都属于电子设计自动化的范畴。

本书讨论的 EDA 技术特指狭义上的概念。EDA 技术的体现形式是在可编程逻辑器件（Programmable Logic Device，PLD）上实现设计者需要的逻辑功能电路。PLD 为一种新兴的芯片，内部逻辑不固定，可以被编程实现各种所需要的逻辑电路，可以被编程为小规模的与非门电路，也可以被编程为一个大规模的复杂的 CPU。与可编程逻辑器件相对的芯片被称为专用集成电路（ASIC），专用集成电路的内部功能不能被编程修改，例如，74LS00，无论何时它都是与非门逻辑功能芯片，除非芯片坏掉。

1.1 可编程逻辑器件概述

可编程逻辑器件（PLD）起源于 20 世纪 70 年代，是在专用集成电路（ASIC）的基础上发展起来的一种新型逻辑器件，是当今数字系统设计的主要硬件平台，其主要特点就是完全由用户通过软件进行配置和编程，从而完成某种特定的功能，且可以反复擦写。在修改和升级 PLD 时，不需额外地改变 PCB 电路板，只是在计算机上修改和更新程序，使硬件设计工作成为软件

开发工作，缩短了系统设计的周期，提高了实现的灵活性并降低了成本，因此获得了广大硬件工程师的青睐，并且形成了规模巨大的 PLD 产业。

目前常见的 PLD 产品有：编程只读存储器（Programmable Read Only Memory，PROM），现场可编程逻辑阵列（Field Programmable Logic Array，FPLA），可编程阵列逻辑（Programmable Array Logic，PAL），通用阵列逻辑（Generic Array Logic，GAL），可擦除的可编程逻辑器件（Erasable Programmable Logic Array，EPLA），复杂可编程逻辑器件（Complex Programmable Logic Device，CPLD）和现场可编程门阵列（Field Programmable Gate Array，FPGA）等类型。PLD 器件从规模上又可以细分为简单 PLD（SPLD）、复杂 PLD（CPLD）以及 FPGA。它们内部结构的实现方法各不相同。

随着制造技术和手段的不断升级和更新，现阶段的 PLD 应用主要集中在 CPLD 和 FPGA 上，其他的产品均已渐渐淡出市场，或成为历史。

1.2 可编程逻辑器件的发展历史

可编程逻辑器件的发展可以划分为 4 个阶段，即从 20 世纪 70 年代初到 70 年代中为第一阶段，20 世纪 70 年代中到 80 年代中为第二阶段，20 世纪 80 年代到 90 年代末为第三阶段，20 世纪 90 年代末到目前为第四阶段。

第一阶段的可编程器件只有简单的可编程只读存储器（PROM）、紫外线可擦除只读存储器（EPROM）和电可擦只读存储器（EEPROM）3 种，由于结构的限制，它们只能完成简单的数字逻辑功能。

第二阶段出现了结构上稍微复杂的可编程阵列逻辑（PAL）和通用阵列逻辑（GAL）器件，正式被称为 PLD，能够完成各种逻辑运算功能。典型的 PLD 由“与”、“非”阵列组成，用“与或”表达式来实现任意组合逻辑，所以 PLD 能以乘积的形式完成大量的逻辑组合。

第三阶段 Xilinx 公司和 Altera 公司分别推出了与标准门阵列类似的 FPGA 和类似于 PAL 结构的扩展性 CPLD，提高了逻辑运算的速度，具有体系结构和逻辑单元灵活、集成度高以及适用范围宽等特点，兼容了 PLD 和通用门阵列的优点，能够实现超大规模的电路，编程方式也很灵活，成为产品原型设计和中小规模（一般小于 10 000）产品生产的首选。这一阶段，CPLD，FPGA 器件在制造工艺和产品性能上都获得长足的发展，达到了 0.18 工艺和系统门数百万门的规模。

第四阶段出现了片上系统（SOC）和可编程片上系统（SOPC）技术，是 PLD 和 ASIC 技术融合的结果，涵盖了实时化数字信号处理技术、高速数

据收发器、复杂计算以及嵌入式系统设计技术的全部内容。Xilinx 公司和 Altera 公司也推出了相应的 SoCFPGA 产品，制造工艺达到 28nm，系统门数也超过千万门。并且，这一阶段的逻辑器件内嵌了硬核高速乘法器、Gbits 差分串行接口、时钟频率可以超过 500MHz 的 PowerPC 微处理器、软核 MicroBlaze，Picoblaze，Nios 以及 Nios Ⅱ，不仅实现了软件需求和硬件设计的完美结合，还实现了高速与灵活性的完美结合，使其既超越了 ASIC 器件的性能和规模，也超越了传统意义上 FPGA 的概念，使 PLD 的应用范围从单片扩展到系统级。目前，基于 PLD 片上可编程的概念仍在进一步向前发展。

1.3 可编程逻辑器件的开发工具

基于高复杂度 PLD 器件的开发，在很大程度上要依靠电子设计自动化（EDA）来完成。PLD 的 EDA 工具以计算机软件为主，将典型的单元电路封装起来形成固定模块并形成标准的硬件开发语言（如 Verilog HDL 语言）供设计人员使用。设计人员考虑如何将可组装的软件库和软件包搭建出满足需求的功能模块甚至完整的系统。PLD 开发软件需要自动地完成逻辑编译、化简、分割、综合及优化、布局布线、仿真以及对于特定目标芯片的适配编译和编程下载等工作。典型的 EDA 工具中必须包含两个特殊的软件包，即综合器和适配器。综合器的功能就是将设计者在 EDA 平台上完成的针对某个系统项目的 HDL、原理图或状态图形描述，针对给定的硬件系统组件，进行编译、优化、转换和综合。

随着开发规模的级数性增长，就必须减短 PLD 开发软件的编译时间、并提高其编译性能以及提供丰富的知识产权（IP）核资源供设计人员调用。此外，PLD 开发界面的友好性以及操作的复杂程度也是评价其性能的重要因素。目前在 PLD 产业领域中，各个芯片提供商的 PLD 开发工具已成为影响其成败的核心成分。只有全面做到芯片技术领先、文档完整和 PLD 开发软件优秀，芯片提供商才能获得客户的认可。

Xilinx 公司的 ISE、Altera 公司的 Quartus Ⅱ（详见附录）等是业界公认的优秀集成 PLD 开发软件。此外综合软件 Synplify 和仿真软件 ModelSim 等诸多第三方开发软件也满足上述要求。还有一些 EDA 软件，如 Altium Designer，不仅具有上述软件的功能，还可以将一些传统的老式的处理器，如 8051，pic165cx 和 z80 等单片机以“软”处理器（soft-core）的形式直接提供给用户，用户可以将这些“处理器”下载到 FPGA 内部，使处理器和硬件外设同时集成到一个芯片内部，实现另一种形式的片上系统。

Altera © SoC 使用宽带干线互联，在 FPGA 架构中集成了基于 ARM 的硬核处理器系统（HPS），包括处理器、外设和存储器接口。它实现了高性能和低功耗特性，以及可编程逻辑的灵活性。这些基于可定制 SoC 非常适合于在一片 FPGA 中集成分立处理器和数字信号处理（DSP）功能，降低系统功耗和成本，减小电路板面积。

通过处理器和 FPGA 之间的宽带互联，增强系统性能。使用 Altera 独特的 FPGA 自适应调试功能开发 ARM 兼容软件，前所未有地提高了目标可视化、控制能力和效能。

HPS 包括一个双核 ARM © Cortex™-A9 MPCore™处理器、丰富的外设，以及共享 FPGA 中逻辑的多端口存储器控制器，提供灵活的可编程逻辑，降低了硬核 IP 的成本，迅速实现定制 ARM 处理器，没有 ASIC 那样昂贵的设计、验证和流片（NRE）成本。

1.4 可编程逻辑器件的逻辑定制原理

现阶段，常用的可编程逻辑器件主要有两类：CPLD 和 FPGA，它们的结构不同、工作原理不同。

1.4.1 CPLD 的基本结构和工作原理

CPLD，即复杂可编程逻辑器件。内部由各种逻辑功能部件（如逻辑门、D 触发器）和可编程开关构成。为了能更清楚地展示 CPLD 内部逻辑构成，用几种符号代替常见的门电路。

（1）缓冲器。

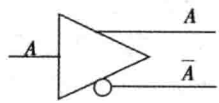


图 1-1 缓冲器的表示形式

在图 1-1 中，左侧的 A 表示输入，右侧上方的 A 为输出，它的逻辑电平和输入相同，故采用了同一个符号表示；右侧下方 $\bar{A}$ 为输入 A 的逻辑反相，即若输入 A 为高，则 $\bar{A}$ 为低。反之，若输入 A 为低电平，则 $\bar{A}$ 为高电平。圆圈“O”表示“非”的含义。

（2）与门、或门的表示。

如图 1-2 所示，左侧部分为数字电路中“门”的表示符号，右侧为本书中的表示符号。

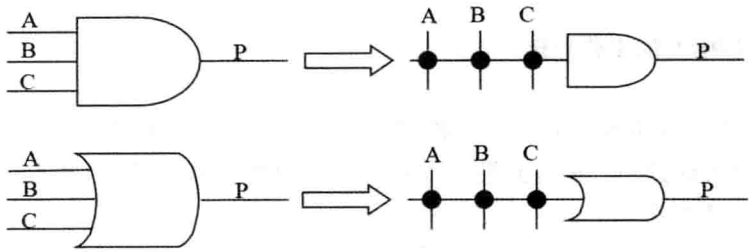


图 1-2 与、或门在本书中的表示形式

为了使电路看起来更加清晰明了，本章中采用的门电路符号均用右侧的符号表示相应的门，其中黑点表示连接。还有可编程连接和不连接的表示形式，见（3）。

（3）连接点。

图 1-3 交叉线的连接形式

在图 1-3 中，“点”表示固定连接，“叉”表示可编程开关的连接，只交叉没有任何表示的为无连接。以基本的逻辑关系 $F = A \cdot \bar{B} \cdot C + \bar{A} \cdot B \cdot C$ 为例，如图1-4 所示。将图中的可编程连接点通过 EDA 工具连接即可实现该表达式的功能。

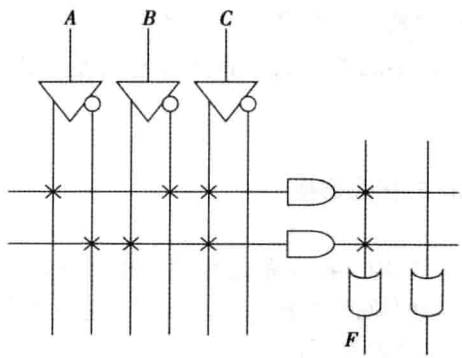


图 1-4 基本的逻辑电路编程原理

若想实现时序电路，由于 CPLD 的内部具有少量的 D 触发器，因此将组合逻辑电路和 D 触发器编程连接在一起即可实现。

1.4.2 FPGA 的工作原理

FPGA 内部由大量的查找表构成，一般来说 4 输入的查找表最为常用。所谓 4 输入查找表，本质上是一个具有 4 根地址线输入的 RAM 存储器。原理如图 1-5 所示。

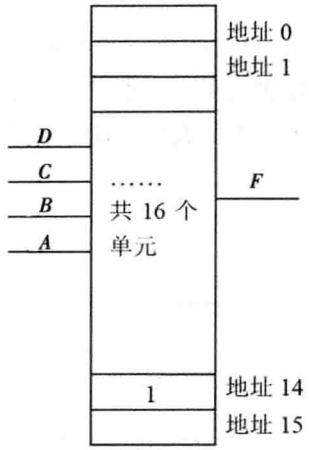


图 1-5 查找表结构的 FPGA 工作原理

在这 16 个可编程的存储空间中，若想实现 $F = D \cdot C \cdot B \cdot \bar{A}$ 功能，在地址 14 的空间中置入逻辑 1 即可。这个置入过程由 EDA 软件完成。若想实现更多输入的电路逻辑，可以使用多个查找表组合完成。若要实现时序电路，可以将 FPGA 内部的查找表配置成边沿触发形式（即 D 触发器）即可。

1.4.3 FPGA/CPLD 的在线编程

现在的 FPGA/CPLD 均支持在系统编程功能，所谓在系统编程，使之对器件的逻辑功能进行随时修改的能力，即重构。它可以在产品设计、生产过程的任一环节，甚至是产品交付以后。

几乎所有的新型芯片，包括 FPGA/CPLD，均在芯片内部集成了 IEEE

1149.1 的 JTAG 协议接口。通过该接口，使用特定的下载线就可以将电脑开发环境中编译出的结果烧录到芯片的内部，实现芯片功能的重定制，即对芯片的编程烧写。

不同的厂家的编程接口电路连接形式有所不同，即便是同一厂家不同的芯片也可能不同，在设计电路过程中，需要参考具体芯片的官方手册。

2 Verilog HDL 入门

本章主要讲授 Verilog HDL 的基础语法以及与数字电路之间的关系。

本章内容：基本 Verilog HDL 的语法构成
基本的组合电路设计及语法组成
基本的时序电路设计及语法构成
Verilog HDL 部分基本语句
阻塞赋值与非阻塞赋值
边沿检测电路的描述

Verilog HDL 是一种硬件描述语言，它是用来描述硬件电路的，而现阶段 Verilog HDL 主要描述的是数字电路。需要注意的是，“描述”二字不是“编程”语言，所以从这个角度讲硬件描述语言与数字电路之间会有着不可分割的联系。以下主要通过 HDL 与数字电路的联系来讲述基本的 Verilog HDL 语法。

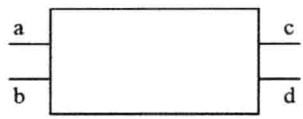
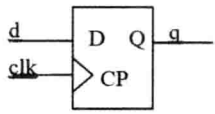
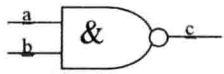
2.1 Verilog HDL 简介及基本语法

Verilog HDL 是 1983 年有 GDA（Gateway Design Automation）公司的 Phil Moorby 首创的。它是可以在 CPLD/FPGA 上设计数字电路系统的一种语言，其系统级的描述能力强，能够描述出逻辑功能相当复杂甚至是处理器级的电路系统，是目前应用最广泛的硬件描述语言之一。

2.1.1 Verilog HDL 基本语法

既然 Verilog HDL 被称为硬件描述语言，那么在 Verilog HDL 语言中，Verilog HDL 所描述的电路逻辑在数字电路中就应该有相应的符号与其相对应。表 2-1 是常见的一些数字电路符号和 Verilog HDL 之间的对应关系表。

表 2-1 数字电路符号和 Verilog HDL 之间的对应关系

比较项	数字电路的表示符号	Verilog HDL 的描述
电路符号及名称		modulefunc (a, b, c, d); (module 关键字后面的 func 称为模块名) …… endmodule
端口	上图中的 a，b，c，d 称为该电路 func 的端口。	在模块名后面的括号中都是端口，端口之间用“,”隔开
端口的方向	假设图中 a，b 为输入端口，c，d 为输出端口。	input a; 用 input 声明的端口为输入端口 input b; output c; 用 output 声明的端口为输出端口 output d; 此外，用 inout 声明的端口为双向端口
端口的类型	在数字电路中，通常具有两类数据类型，能保持数值的寄存器型（如触发器）和不能保持数值的连线型（即导线）。 	wire a; 用 wire 声明的端口为连线型 reg c; 用 reg 声明的端口为寄存器型。通常，输入一般都为 wire 型，而输出可以为 wire 型也可以为 reg 型。 wire d; wire clk; reg q;
功能实现	由内部硬件电路实现，此处以与非门为例： 	使用 always 或者 assign 或者例化语句实现如： ① always c = ! (a & b); ② assign c = ! (a & b); ③NAND2 func (c, a, b); 这里要求掌握①②的用法，③的用法后面章节将详细讲述 被赋值对象为 wire 型时，使用 assign 语句进行赋值 被赋值对象为 reg 型时，使用 always 语句进行赋值
边沿	数字电路中有上升沿和下降沿	posedge 信号，表示信号的上升沿 negedge 信号，表示信号的下降沿 posedge 和 negedge 关键字都是配合 always 使用的。用法稍后讲述

续表 2-1

比较项	数字电路的表示符号	Verilog HDL 的描述
总线	假设有 8 位数据总线 D0~D7	用 D［7：0］或者 D［0：7］或者 D［8：1］或者 D［1：8］等表示。也可以用 D［4：-3］之类的表示方式
其他	高电平 低电平 高阻态 不确定态	1 0 Z 或 z X 或 x

2.1.2 Verilog HDL 结构构成

Verilog HDL 描述的一个基本的电路被称作模块，模块的结构如下：

module <模块名>（ <端口列表>）；

<端口方向定义和类型>

<内部功能的实现>

endmodule

一个完整的 Verilog HDL 通常由 4 部分构成：

（1）模块声明部分。关键字 module 和 endmodule 组成。紧接着 module 后面的是模块名称和端口名列表，注意，端口名列表的括号后面有个分号。

（2）端口方向定义。指明所有的端口是输入还是输出或者是双向（双向端口定义的关键字为 inout，后面会学习到）的。

（3）端口（和内部变量）类型。一般来说，所有的输入都是 wire 型，输出端一般要根据是组合电路还是时序电路分别定义成 wire 型和 reg 型。

（4）功能实现部分。这部分通常是设计的核心，大型设计的重点就在这里，通常由 always，assign 和元件例化（后面会学习到）构成。

任何一个电路的 Verilog HDL 描述都由这 4 部分构成，无论设计的逻辑电路是多么的简单或者多么的复杂。以下通过一些基本的 Verilog HDL 实例进行详细说明。

例 2-1：一个简单的组合逻辑电路的 Verilog HDL 实现—用 Verilog HDL 实现一个与门的设计。

代码注释：

文件名 Myandgate2. v