手把手教你学系列丛书

阿　东　编著

手把手

教你学FPGA

北京航空航天大学出版社
BEIHANG UNIVERSITY PRESS

手把手教你学系列丛书

手把手教你学 FPGA

阿　东　编著

北京航空航天大学出版社

内 容 简 介

本书主要讲解 FPGA 的程序设计，以一款热销的 FPGA 开发板为例，介绍学习 FPGA 和使用 Verilog，以及 FPGA 开发板的硬件配置，重点是第 3 章的 16 个典型实例程序，由简单到复杂，最后是 FPGA 的设计心得。

本书适合电子、通信、自动化等相关专业的本科生以及从事 FPGA 开发/IC 设计/PCB 等相关职业的初学者阅读参考。

图书在版编目(CIP)数据

手把手教你学 FPGA / 阿东编著. -- 北京 : 北京航空航天大学出版社，2017.2

ISBN 978-7-5124-1049-7

Ⅰ. ①手… Ⅱ. ①阿… Ⅲ. ①可编程序逻辑器件—研究 Ⅳ. ①TP332.1

中国版本图书馆 CIP 数据核字(2017)第 002529 号

手把手教你学 FPGA

阿 东 编著

责任编辑 张冀青

*

北京航空航天大学出版社出版发行

北京市海淀区学院路 37 号(邮编 100191) http://www.buaapress.com.cn

发行部电话：(010)82317024 传真：(010)82328026

读者信箱：emsbook@buaacm.com.cn 邮购电话：(010)82316936

北京市同江印刷有限公司印装 各地书店经销

*

开本：710×1 000 1/16 印张：12 字数：256 千字

2017 年 3 月第 1 版 2017 年 3 月第 1 次印刷 印数：3 000 册

ISBN 978-7-5124-1049-7 定价：35.00 元

前言

本书写作的目的是让更多的同学踏入 FPGA 的世界，消除 FPGA 带给大家的一种很难学习的感觉。

很多同学对于学习单片机、ARM、FPGA、DSP 非常困惑，不清楚自己要学习什么，学习哪个更有前途。

当前，学习单片机和 ARM 的人最多，而学习 FPGA 和 DSP 的人相对少一些；单片机和 ARM 的市场应用也是最多的，基本上涉及各个领域，而 DSP 一般应用在算法领域，FPGA 一般应用在高性能处理、实时要求高的领域，比如高速接口、报文转发、图像处理、视频传输等，还可以应用在芯片前期验证。

如果您对算法不是很敏感，那么可以排除 DSP，笔者就对算法不是很敏感。如果您只是想找个工作，对于是单片机还是 FPGA 没有要求，那么可以好好学习一下单片机和 ARM。当然，对 ARM 的深入学习也是比较困难的，特别是要达到操作系统的级别。FPGA 本身的特性决定了其适合做高速和大容量处理，而且 FPGA 工程的方案设计相对而言比较复杂和系统化，所以学习之后，可以使你的系统观念更强，几年之后，与做单片机相比，其差异也会越来越大，成为系统工程师也就指日可待了。这也是学习 FPGA 能给您带来的一个好处，当您系统观念越来越强的时候，承担的工作也就越来越重要，待遇自然也不用多说了。

学习 FPGA 还有一个好处就是可以转型做芯片设计，因为做芯片设计是很多人心中的一个梦想，想当年笔者就是怀揣着这个梦想走下来的。

笔者在 FPGA 领域里面已经奋斗了 8 年，希望自己的一些经验，能够让学习它的人少走一些弯路。其实，FPGA 本身只是一个器件，无论是 ALTERA 的 FPGA 还是 XILINX 的 FPGA，目的都是学会使用 FPGA，而使用 FPGA 是比较简单的。

FPGA 的难点是逻辑设计。逻辑设计是需要长时间积累和锻炼的，不是一蹴而就的。初期最好有一套入门级的 FPGA 开发板和配套书籍，根据配套书中的实例一个一个学习，踏踏实实，多去思考，才能真正掌握逻辑设计的精髓。简单实际的例子学会了，再设计复杂点的小系统，其中的原理都是相通的。比如学会了写 100 行代码，然后再试着写 1000 行代码，到了会写 1000 行代码的时候，上万行甚至几十万行的代码也就不是什么难事了。本书就是基于这个思路进行编写的，由简单的流水灯开始，逐渐向复杂的 SDRAM 控制器进发。

现在的逻辑芯片设计，一般都是几十万行甚至上百万行的代码，大多是由几十人

组成一个团队进行开发，设计和验证的工作由不同的同事分开进行，这就需要彼此默契配合和沟通。因此，同学们在学习的时候也要多讨论和交流，养成团队开发的好习惯。

另外，编程规范也需要同学们多重视，一个好的风格的代码可以节省自己和他人大量的时间，尤其是在项目开发里面，因为你的代码是需要其他人检视的。好的风格的代码一般具有清晰的注释、对齐的格式，命名能体现信号含义，逻辑清晰易懂，组合逻辑不能太复杂等。

逻辑设计能力的提高和修行差不多，都是苦差事，但是吃得苦中苦，方为人上人。从助理工程师、普通工程师、资深工程师、系统工程师到架构师，是一个漫长的修练过程。

本书是初次出版，肯定还存在这样那样的问题，希望大家反馈和指正（沟通 QQ：1530384236），一起让这本书完善起来，让更多的同学加入到学习 FPGA 的世界！

阿　东

2016 年 10 月 8 日

目 录

第1章

FPGA 概述

FPGA 的出现可以说是划时代的事件。FPGA 改变了单板硬件的设计方式，赋予 RTL 设计极大的灵活性，让 ASIC 前期物理验证有了保证，让各种高速接口有了载体，让规范协议的变化不再和硬件强相关，等等。

1.1 为什么要学习 FPGA

很多在校的学生对于学单片机、ARM、FPGA、DSP 非常困惑，不清楚自己要学习什么。当前学习单片机和 ARM 的人是最多的，而学习 FPGA 和 DSP 的人相对少一些；单片机和 ARM 的市场应用也是最多的，基本上涉及各个领域，而 DSP 一般应用在算法领域，FPGA 一般应用在高性能处理、实时要求高的领域，比如高速接口、报文转发、图像处理、视频传输等，还可以应用在芯片前期验证。

如果你对算法不是很熟悉，那么可以排除 DSP。如果你只是想找个工作，对于是单片机还是 FPGA 没有要求，那么可以好好学习一下单片机和 ARM。当然，对 ARM 的深入学习也还是比较困难的，特别是要达到操作系统级别。FPGA 本身的特性决定了它适合做高速和大容量处理，而且 FPGA 工程的方案设计相对比较复杂和系统化，所以学习它之后，你的系统观念会更强，几年之后，与做单片机的差异也会越来越大，很可能那时你就是系统工程师了。这也是学习 FPGA 所能带来的一个好处，当你的系统观念越来越强时，承担的工作也就越来越重要，待遇自然也会越来越好。

学习 FPGA 还有一个好处就是可以转型做芯片设计，因为做芯片设计是很多人心中的一个梦想，想当年笔者就是怀揣着这个梦想走下来的。

1.2 学习 FPGA 的几个疑点

1.2.1 选择 VHDL 还是 Verilog

VHDL 和 Verilog 都是用于数字电子系统设计的硬件描述语言，而且两种语言都是 IEEE 的标准。

VHDL 诞生于 1982 年。在 1987 年底，VHDL 被 IEEE 和美国国防部确认为标准硬件描述语言。自 IEEE 公布了 VHDL 的标准版本——IEEE - 1076(简称 87 版)之后，各 EDA 公司相继推出了自己的 VHDL 设计环境，或宣布自己的设计工具可以和 VHDL 接口。此后，VHDL 在电子设计领域被广泛接受，并逐步取代了原有的非标准的硬件描述语言。而 Verilog HDL 是由 GDA(Gateway Design Automation)公司的 Phil Moorby 在 1983 年末首创的，最初只设计了一个仿真与验证工具，之后又陆续开发了相关的故障模拟与时序分析工具。1985 年，GDA 推出它的第三个商用仿真器 Verilog - XL，获得了巨大的成功，从而 Verilog HDL 得到迅速推广应用。1989 年，CADENCE 公司收购了 GDA 公司，Verilog HDL 成为了该公司的独家专利。1990 年，CADENCE 公司公开发表了 Verilog HDL，并成立了 LVI 组织，以促进 Verilog HDL 成为 IEEE 标准，即 IEEE Standard 1995。

2001 年，IEEE 发布了 Verilog—2001 标准。Verilog—2001 相对于 Verilog—1995 做了很多有用的改进，给编程人员带来很大的帮助。

很多初学者对于学习 FPGA 是使用 VHDL 还是 Verilog 没有方向，很多学校目前还是在教 VHDL，真是误人子弟。之前，一位初学者学习了一个月的 VHDL，然后才问我该学习 VHDL 还是 Verilog。在此特别强调一下，现在基本上所有的芯片和设计都采用 Verilog，本人也经历了几个大型项目，全部采用 Verilog，包括 ARM 的芯片和 IP 都是采用 Verilog，使用 VHDL 的人非常非常少，只有部分学校还在教学和少数研究所在用。Verilog 已经占据了完全主导的地位。

它们的设计思想都是一样的，为什么还要学习大公司都不用的东西呢？

1.2.2 NIOS 重要还是 Verilog 重要

NIOS 嵌入式处理器是 Altera 公司推出的采用哈佛结构、具有 32 位指令集的第二代片上可编程的软核处理器，其最大优势和特点是模块化的硬件结构，以及由此带来的灵活性和可裁剪性。

因此很多初学者可能觉得 NIOS 很重要，但是很遗憾地告诉大家，FPGA 的主流和优势设计是使用 Verilog。为什么不是 NIOS 呢？因为 NIOS 使用 C 语言开发，需

要在FPGA宝贵的逻辑资源内建一个CPU,速度可能还不及一般的ARM芯片。另外,功耗、成本都比ARM高很多,易用性也比ARM差很多。在一些ARM满足不了的场景使用NIOS倒是不错,比如系统需要3个以上串口等多个相关外设,或者单板上已经有大容量的FPGA(FPGA资源不使用也浪费)。

FPGA为什么还那么流行呢?因为FPGA是逻辑器件,内部都是由可以编程的寄存器、组合逻辑构成,使用硬件语言编程实现,可以一个时钟周期执行一次动作,甚至0个时钟周期都可以执行操作(完全组合逻辑实现),而单片机/ARM需要CPU取指、译码、执行,自然FPGA的速度非常快,这一点是ARM替代不了的。

1.3 FPGA简介

FPGA(Field-Programmable Gate Array),即现场可编程门阵列,它是在PAL(Programmable Array Logic,可编程阵列逻辑)、GAL(Generic Array Logic,通用可编程阵列逻辑)、CPLD(Complex Programmable Logic Device,复杂的可编程逻辑器件)等可编程器件的基础上进一步发展的产物。它是作为专用集成电路(ASIC)领域中的一种半定制电路而出现的,既解决了定制电路的不足,又克服了原有可编程器件门电路数有限的缺点。

可以说,FPGA是划时代的发明,解决了硬件逻辑可以任意编程而不需要改动硬件的问题,提高了硬件高速处理的能力,同时给ASIC前端验证提供了非常重要的手段。

图1.1是FPGA的结构原理图,内部包含了很多个物理单元。

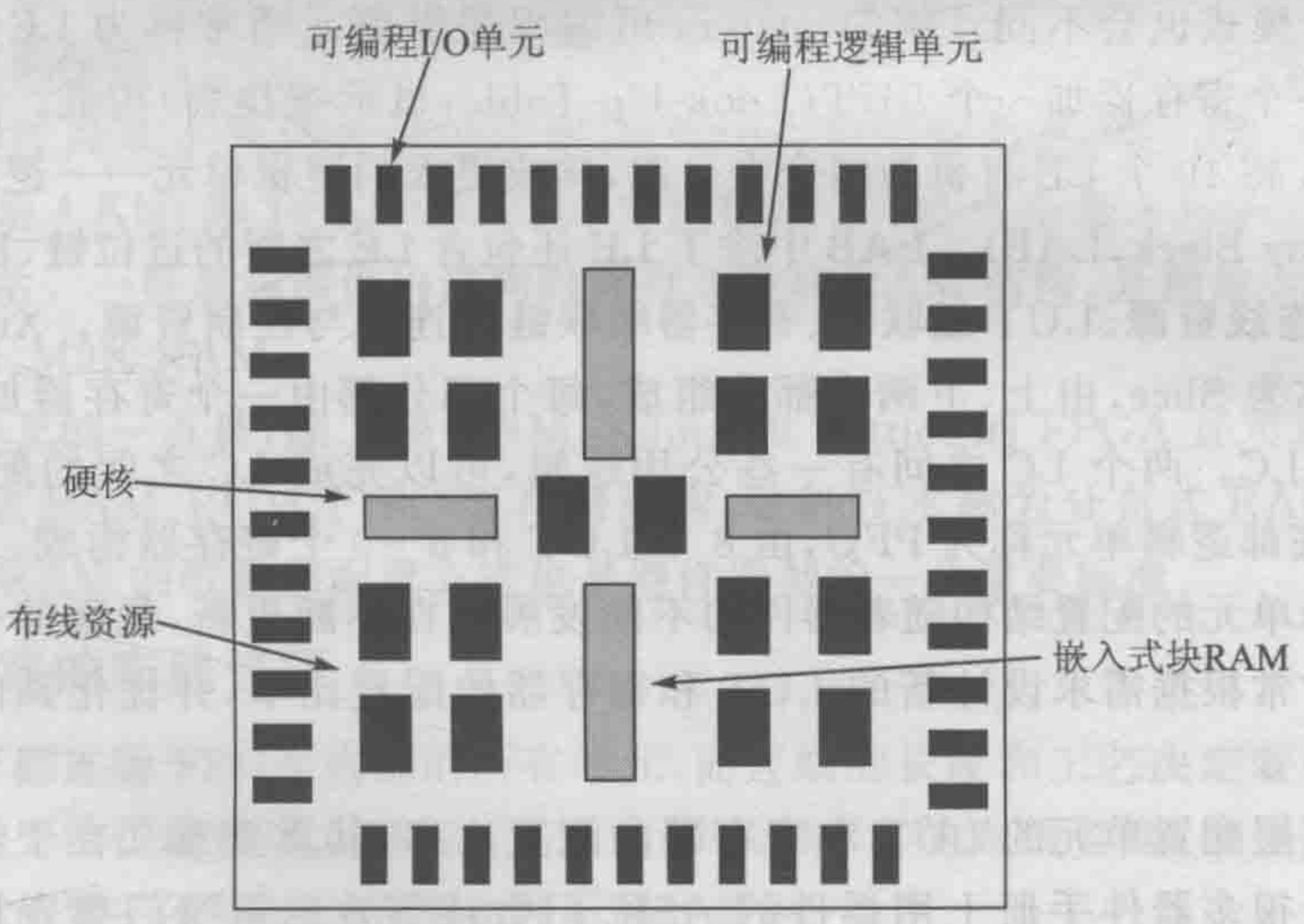

图1.1 FPGA结构原理图

下面介绍每个单元的基本概念。

1. 可编程输入/输出单元

输入/输出单元简称 I/O 单元。它们是芯片与外界电路的接口部分，用于完成不同电气特性下对输入/输出信号的驱动与匹配需求。为了使 FPGA 有更灵活的应用，目前大多数 FPGA 的 I/O 单元被设计为可编程模式，即通过软件的灵活配置，可以适配不同的电气标准与 I/O 物理特性；可以调整匹配阻抗特性，上下拉电阻；可以调整驱动电流的大小等。

可编程 I/O 单元支持的电气标准因工艺不同而异，因为不同器件商、不同器件族的 FPGA 支持的 I/O 标准不同。一般来说，常见的电气标准有 LVTTL、LVCMOS、SSTL、HSTL、LVDS、LVPECL、PCI 等。值得一提的是，随着 ASIC 工艺的飞速发展，目前可编程 I/O 接口支持的最高频率越来越高，一些高端 FPGA 通过 DDR 寄存器技术，甚至可以支持高达 2 Gbit/s 的数据速率。

2. 基本可编程逻辑单元

基本可编程逻辑单元是可编程逻辑的主体，可以根据设计灵活地改变其内部连接与配置，完成不同的逻辑功能。FPGA 一般是基于 SRAM 工艺的，其基本可编程逻辑单元几乎都是由查找表和寄存器组成的。FPGA 内部查找表一般分为 4 个输入，查找表完成纯组合逻辑功能。FPGA 内部寄存器结构相当灵活，可以配置为带同步/异步复位或置位、时钟使能的触发器，也可以配置成锁存器。FPGA 一般依赖寄存器完成同步时序逻辑设计。比较经典的基本可编程逻辑单元的配置是一个寄存器加一个查找表，但是不同的厂商，寄存器与查找表也有一定的差异，而且寄存器与查找表的组合模式也会不同。例如，Altera 可编程逻辑单元通常称为 LE(Logic Element)，由一个寄存器加一个 LUT(Look-Up-Table，显示查找表)构成。Altera 的大多数 FPGA 将 10 个 LE 有机地组合在一起，构成更大的逻辑单元——逻辑阵列模块(Logic Array Black，LAB)。LAB 中除了 LE 还包含 LE 之间的进位链、LAB 控制信号、局部互连线资源、LUT 级联链、寄存器级联链等连线与控制资源。Xilinx 可编程逻辑单元称为 Slice，由上、下两个部分组成，每个部分都由一个寄存器加一个 LUT 组成，称为 LC。两个 LC 之间有一些公用逻辑，可以完成 LC 之间的配合与级联。Lattice 的底部逻辑单元称为 PFU，由 8 个 LUT 和 8～9 个寄存器构成。当然，这些可编程逻辑单元的配置结构随着器件的不断发展也在不断更新，最新的一些可编程逻辑器件常常根据需求设计新的 LUT 和寄存器的配置比率，并优化其内部的连接构造。

学习底层配置单元的 LUT 和寄存器的配置比率，其重要意义在于器件选型和规模估算。很多器件手册上用器件的 ASIC 门数或等效的系统门数表示器件的规模。但是，由于目前 FPGA 内部除了基本可编程逻辑单元外，还包含丰富的嵌入式 RAM、PLL 或 DLL，以及专用的 Hard IP Core(如 PCIE、Serdes 硬核)等。这些功能

模块也会等效出一定规模的系统门，所以用系统门权衡基本可编程逻辑单元的数量是不准确的，常常让设计者混淆。比较简单而科学的方法是用器件的寄存器或LUT的数量衡量。例如，Xilinx的Spartan系列XC3S1000有15 360个LUT，而Lattice的EC系列LFEC15E也有15 360个LUT，所以这两款FPGA的可编程逻辑单元数量相当，属于同一规模的产品。Altera的Cyclone Ⅳ器件族EP4CE6的LUT数量是6 000个，就比前面提到的两款FPGA规模略小。需要说明的是，器件选型是一个综合性的问题，需要综合考虑设计的需求、成本、规模、速度等级、时钟资源、I/O特性、封装、专用功能模块等诸多因素。

3. 嵌入式块RAM

目前，大多数FPGA都有内嵌的块RAM。FPGA内部嵌入可编程RAM模块，大大地拓展了FPGA的应用范围和使用灵活性。FPGA内嵌的块RAM一般可配置为单端口RAM、双端口RAM、伪双口RAM、内容可寻址存储器(Coment-Addressable Memory，CAM)、FIFO等常用存储结构。RAM的概念和功能，读者应该非常熟悉，在此不再赘述。FPGA中其实并没有专用的ROM硬件资源，实现ROM的思路是对RAM赋初值。所谓CAM，也称内容地址存储器，CAM这种存储器在其每个存储单元都包含了一个比较的逻辑，写入CAM的数据会与其内部存储的每一个数据进行比较，并返回与端口数据相同的所有内部数据的地址。概括地讲，RAM是一种根据地址读、写数据的存储单元，而CAM恰恰相反，它返回的是与端口数据相同的所有内部地址。CAM的应用十分广泛，比如在路由器中的交换表等。FIFO是先进先出队列的存储结构。FPGA内部实现的RAM、ROM、CAM、FIFO等存储结构都可以基于嵌入式块RAM单元，并根据需求自动生成相应的粘合逻辑以完成地址和片选等控制逻辑。

不同器件商或不同器件族的内嵌块RAM，其结构也不同。Xilinx常见的块RAM大小是4 Kbit和18 Kbit，Lattice常用的块RAM大小是9 Kbit，Altera的块RAM最灵活。一些高端器件内部同时含有3种块RAM结构，分别是M512 RAM、M4K RAM、M9K RAM。

需要补充的一点是，除了块RAM，Xilinx和Lattice的FPGA还可以灵活地将LUT配置成RAM、ROM、FIFO等存储结构，这种技术称为分布式RAM。根据设计需求，块RAM的数量和配置方式也是器件选型的一个重要标准。

4. 丰富的布线资源

布线资源连通FPGA内部的所有单元，而连线的长度和工艺决定着信号在连线上的驱动能力和传输速度。FPGA芯片内部有着丰富的布线资源，根据工艺、长度、宽度和分布位置的不同而划分为4种不同的类别。

第一类是全局布线资源，用于芯片内部全局时钟和全局复位/置位的布线；

第二类是长线资源，用于完成芯片Bank间的高速信号和第二全局时钟信号的

布线；

第三类是短线资源，用于完成基本逻辑单元之间的逻辑互连和布线；

第四类是分布式的布线资源，用于专有时钟、复位等控制信号线。

在实际中设计者不需要直接选择布线资源，布局布线器可自动地根据输入逻辑网表的拓扑结构和约束条件选择布线资源来连通各个模块单元。从本质上讲，布线资源的使用方法与设计的结果有密切、直接的关系。

5. 底层嵌入功能单元

底层嵌入功能单元的概念比较笼统，本文指的是那些通用程度较高的嵌入式功能模块，比如 PLL、DLL、DSP、CPU 等。随着 FPGA 的发展，这些模块被越来越多地嵌入到 FPGA 的内部，以满足不同场合的需求。

目前，大多数 FPGA 厂商都在 FPGA 内部集成了 DLL 或者 PLL 硬件电路，用于完成时钟的高精度、低抖动的倍频、分频、占空比调整、相移等功能。高端 FPGA 产品集成的 DLL 和 PLL 资源越来越丰富，功能越来越复杂，精度越来越高。Altera 芯片集成的是 PLL，Xilinx 集成的是 DLL，Lattice 的新型 FPGA 同时集成了 PLL 与 DLL 以适应不同的需求。Altera 芯片的 PLL 模块分为增强型 PLL 和快速 PLL 等，Xilinx 芯片的 DLL 模块名称为 CLKDLL。在高端 FPGA 中，CLKDLL 的增强型模块为 DCM。这些时钟模块的生成和配置方法一般分为两种。一种是在 HDL 代码和原理图中直接例化，另一种是在 IP 核生成器中配置相关参数，自动生成 IP。Altera的 IP 核生成器叫做 Mega wizard，Xilinx 的 IP 核生成器叫做 Core Generator，Lattice 的 IP 核生成器叫做 Module/IP manager。另外，可以通过在综合、实现步骤的约束文件中编写约束属性来完成时钟模块的约束。

越来越多的高端 FPGA 产品将包含 DSP 或 CPU 等软处理核，从而 FPGA 将由传统的硬件设计手段逐步过渡到系统级设计平台。例如 Altera 的 Stratix、Stratix Ⅱ、Stratix Ⅳ等器件族内部集成了 DSP Core，配合同样的逻辑资源，还可实现 ARM、MIPS、NIOS Ⅱ等嵌入式处理系统；Xilinx 的 Virtes Ⅱ和 Virtex Ⅱ pro 系列 FPGA 内部集成了 Power PC450 的 CPU Core 和 MicroBlaze RISC 处理器 Core；Lattice 的 ECP 系列 FPGA 内部集成了系统 DSP Core 模块。这些 CPU 或 DSP 处理模块的硬件主要由一些加、乘、快速进位链，Pipelining 和 Mux 等结构组成，用逻辑资源和块 RAM 实现软核部分，组成了功能强大的软运算中心。这种 CPU 或 DSP 比较适合实现 FIR 滤波器、编码解码、FFT 等运算密集型应用。FPGA 内部嵌入 CPU 或 DSP 等处理器，使 FPGA 在一定程度上具备了实现软硬件联合系统的能力，FPGA 正逐步成为 SPOC 的高效设计平台。Altera 的系统级开发工具有 SOPC Buider、DSP Builder，以及专用硬件结构与软硬件协同处理模块等；Xilinx 的系统开发工具有 EDK 和 Platform Studio；Lattice 的嵌入式 DSP 开发工具有 Matlab 的 Simulink。

6. 内嵌专用硬核

这里的内嵌专用硬核与前面的底层嵌入单元是有区分的，这里讲的内嵌专用硬核主要指那些通用性相对较弱，不是所有FPGA器件都包含的硬核。我们称FPGA和CPLD为同样逻辑器件，是区别于专用集成电路而言的。其实，FPGA内部也有两个阵营：一方面是通用性强，目标市场范围很广，价格适中的FPGA；另一方面是针对性较强，目标市场明确，价格较高的FPGA。前者主要指低成本FPGA，后者主要面向某些高端通信市场的可编程逻辑器件。

1.4 Verilog简介

Verilog语法标准规定的内容还是很多的，可以单独写一本书了，但是本书项目里面真正用到的不多，本节不准备描述所有的语法，只选择实际项目中用到的几个语法。

阿东也不建议大家一上来就拼命学习各种语法，知道常用的几个语法就可以了，关键还在于学习实践Verilog的设计思想。如果项目里需要用到复杂的语法，打开相关的语法书看一下就知道了。

下面通过流水灯实验来讲解基本的语法，代码如下：

```
module LED(
                    //input
        input       sys_clk,  //system clock
        input       sys_rst_n,  //system reset,low is active
                    //output
        output      reg[7:0]LED
        );

//reg define
reg  [ 2:0]  count;
reg  [24:0]  counter;

//*************************************************************
//* *                    Main Program
//*************************************************************

//count for add counter,0.5 s/20 ns = 25 000 000,need 25 bit cnt
always @(posedge sys_clk or negedge sys_rst_n) begin
    if ( sys_rst_n == 1'b0 )
        counter <= 25'b0;
    else
        counter <= counter + 25'b1;
```

```
    end

    always @(posedge sys_clk or negedge sys_rst_n) begin
        if ( sys_rst_n == 1'b0 )
            count <= 3'b0;
        else if( counter == 25'd0 )
            count <= count + 3'b1;
        else;
    end

    //ctrl the LED pipiline display when count is equal to 0,1..
    always @(posedge sys_clk or negedge sys_rst_n) begin
            if ( sys_rst_n == 1'b0 )
                LED <= 8'b0;
            else begin
                case ( count )          //通过控制 I/O 口的高低电平实现发光二极管的亮灭
                    0: LED = 8'b00000001;
                    1: LED = 8'b00000010;
                    2: LED = 8'b00000100;
                    3: LED = 8'b00001000;
                    4: LED = 8'b00010000;
                    5: LED = 8'b00100000;
                    6: LED = 8'b01000000;
                    7: LED = 8'b10000000;
                    default: LED = 8'b00000000;
                endcase
            end
    end

    endmodule
    //end of rtl code
```

1.4.1 端口定义

模块的端口可以是输入端口、输出端口或双向端口。缺省的端口类型为线网类型(即 wire 类型)。输入端口默认为 wire 类型,不需要定义,输出端口或双向端口可声明为 wire/reg 类型,使用 reg 必须显式声明,使用 wire 也强烈建议显式声明。

Verilog 2001 语法中的端口定义包括端口名、信号的输入/输出、wire/reg 类型、位宽定义,极大地减少了端口声明占用的代码行数。当前 Verilog 2001 语法已经非常普及。