

<<Xilinx FPGA高级设计及应用>>

图书基本信息

书名：<<Xilinx FPGA高级设计及应用>>

13位ISBN编号：9787121166266

10位ISBN编号：7121166267

出版时间：2012-4

出版时间：电子工业出版社

作者：汤琦，蒋军敏 编著

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

## <<Xilinx FPGA高级设计及应用>>

### 内容概要

《Xilinx

FPGA高级设计及应用》用简洁的语言向读者展示了FPGA高质量和可靠性设计中必须掌握的概念、思想和设计方法，如FPGA设计思想、高速数据传输系统设计、高级配置管理、空间应用可靠性设计、分区设计和高效测试系统设计等。

本书来源于工程实际，选取的专题都是实际工程开发中必须面对、难度很高的问题。

作者结合多年的工作经验编写了本书，书中深入剖析了其实现原理和技术细节，并提供了使用方法和实例。

# <<Xilinx FPGA高级设计及应用>>

## 书籍目录

### 第1章 概述

- 1.1 可编程逻辑器件与PLD开发工具
  - 1.1.1 可编程逻辑器件
  - 1.1.2 可编程逻辑器件的发展历史
  - 1.1.3 PLD开发工具
- 1.2 FPGA工作原理与开发流程
- 1.3 可编程技术
  - 1.3.1 SRAM编程技术
  - 1.3.2 Flash / E2PROM编程技术
  - 1.3.3 反熔丝编程技术
  - 1.3.4 编程技术比较
- 1.4 FPGA芯片结构
  - 1.4.1 可编程输入/输出单元 (IOB)
  - 1.4.2 可配置逻辑块 (CLB)
  - 1.4.3 数字时钟管理模块 (DCM)
  - 1.4.4 嵌入块式RAM (BRAM)
  - 1.4.5 布线资源
  - 1.4.6 内嵌功能单元
  - 1.4.7 内嵌专用硬核
  - 1.4.8 软核、硬核及固核的概念
- 1.5 Xilinx公司FPGA介绍

- 1.5.1 Spartan系列

- 1.5.2 Virtex系列

### 第2章 FPGA设计思想

- 2.1 可综合设计思想
  - 2.1.1 VHDL语言概述
  - 2.1.2 设计层次
  - 2.1.3 可综合描述规范
  - 2.1.4 组合逻辑电路可综合设计
  - 2.1.5 时序逻辑电路可综合设计
- 2.2 面积与速度互换思想
  - 2.2.1 利用层次化设计控制设计结构
  - 2.2.2 if语句和case语句控制实现结构
  - 2.2.3 减少关键路径的逻辑级数
  - 2.2.4 流水线Pipelining
  - 2.2.5 串行转并行处理
  - 2.2.6 组合逻辑和时序逻辑分离
- 2.3 时钟设计思想
  - 2.3.1 工作时钟模型
  - 2.3.2 全局时钟
  - 2.3.3 门控时钟
  - 2.3.4 多级逻辑时钟
  - 2.3.5 行波时钟
  - 2.3.6 多时钟系统
  - 2.3.7 Xilinx FPGA中的时钟资源

## <<Xilinx FPGA高级设计及应用>>

- 2.3.8 时序约束
- 2.4 同步设计思想
  - 2.4.1 异步电路和同步电路
  - 2.4.2 一般组合逻辑的同步设计
  - 2.4.3 二次时钟的同步设计
  - 2.4.4 多时钟系统的同步设计
  - 2.4.5 非同源时钟同步化(D触发器使能信号的合理使用)
  - 2.4.6 数据接口同步设计
- 2.5 延时电路设计思想
- 2.6 复位电路设计思想
  - 2.6.1 同步复位
  - 2.6.2 异步复位
  - 2.6.3 触发器组模块的复位
  - 2.6.4 复位电路的同步化方法
- 2.7 抗干扰设计思想
  - 2.7.1 干扰产生的原因
  - 2.7.2 干扰抑制设计方法
  - 2.7.3 基于采样法的串口通信设计
- 2.8 可靠性设计检查单
- 第3章 高速数据传输设计
  - 3.1 概述
    - 3.1.1 高速数据通信的发展现状
    - 3.1.2 几种高速数据通信方案简介
  - 3.2 高速数据传输中的同步技术
    - 3.2.1 同步方法及其特点
    - 3.2.2 同步方式比较及对数据通信的影响
  - 3.3 FPGA对同步技术的支持
    - 3.3.1 动态相位调整技术
    - 3.3.2 基于ChipSync的动态相位调整方法
    - 3.3.3 串行收发器SERDES ( ISERDES和OSERDES )
  - 3.4 应用实例——基于SERDES的多路高速同步传输系统
    - 3.4.1 系统方案
    - 3.4.2 发送模块
    - 3.4.3 接收模块
  - 3.5 基于RocketIO的高速数据传输系统
    - 3.5.1 自同步通信系统架构
    - 3.5.2 RocketIO简介
    - 3.5.3 基于标准协议的可靠通信模型
    - 3.5.4 应用实例——基于自定义协议的即时传输系统
  - 3.6 高速数据通信的加固设计
    - 3.6.1 数据通信加固的概念
    - 3.6.2 交织汉明码原理及其性能分析
    - 3.6.3 基于交织汉明码的高速通信加固设计
  - 3.7 LVDS应用设计
    - 3.7.1 LVDS简介
    - 3.7.2 LVDS系统设计
- 第4章 Xilinx FPGA高级配置管理

## <<Xilinx FPGA高级设计及应用>>

- 4.1 配置模式
  - 4.1.1 配置接口及配置模式
  - 4.1.2 配置引脚
  - 4.1.3 SelectMAP接口
  - 4.1.4 和配置相关的BitGen选项
- 4.2 配置电路
  - 4.2.1 配置时的电源要求
  - 4.2.2 常用配置存储器介绍
  - 4.2.3 主串模式配置电路
  - 4.2.4 主从模式配置电路
  - 4.2.5 SelectMAP模式配置电路
- 4.3 FPGA配置过程
  - 4.3.1 配置数据流加载过程
  - 4.3.2 从串模式配置过程
  - 4.3.3 SelectMAP模式配置数据加载
  - 4.3.4 延时加载
- 4.4 配置命令分析
  - 4.4.1 配置帧
  - 4.4.2 配置数据流类型
  - 4.4.3 配置帧寻址方式
  - 4.4.4 配置寄存器
  - 4.4.5 配置命令执行过程分析
  - 4.4.6 配置数据解析程序
- 4.5 配置内存回读
  - 4.5.1 回读准备设计
  - 4.5.2 回读指令序列
  - 4.5.3 回读数据校验
- 4.6 配置内存重构（刷新）
  - 4.6.1 SelectMAP模式下重构流程
  - 4.6.2 刷新对系统功能的影响
- 4.7 SelectMAP接口下重配置实现代码
- 4.8 配置数据文件格式分析
  - 4.8.1 字节格式
  - 4.8.2 文件格式
- 第5章 空间应用可靠性设计
  - 5.1 集成电路单粒子效应的机理
  - 5.2 单粒子翻转故障模式
  - 5.3 SRAM型FPGA单粒子问题的缓解措施
    - 5.3.1 循环加电
    - 5.3.2 配置管理
    - 5.3.3 Xilinx三模冗余
    - 5.3.4 器件冗余
  - 5.4 三模冗余设计方法介绍
    - 5.4.1 三模冗余原理
    - 5.4.2 TMR Tool工具介绍
    - 5.4.3 TMR Tool设计流程
    - 5.4.4 创建一个ISE工程完成三模冗余前的设计

## <<Xilinx FPGA高级设计及应用>>

5.4.5 创建一个TMR Tool工程产生三模冗余后的网表

5.4.6 创建第二个ISE工程完成三模冗余后的设计

5.4.7 三模冗余技术问题分析

5.5 Half?Latch处理

5.6 异步FIFO处理

5.7 配置、刷新一体化方法

5.7.1 实现电路

5.7.2 工作流程及控制时序

5.7.3 SelectMAP接口Active刷新实现

第6章 分区设计

6.1 为什么使用分区设计

6.2 分区与SmartGuide

6.3 如何使用分区

6.4 用Synplify

6.5 分区保留级别

6.6 分区保留

6.7 对分区进行布局规划

6.8 删除分区

6.9 结论

第7章 高效验证 ( TestBench ) 设计

7.1 为什么要进行仿真验证

7.2 仿真验证程序设计

7.2.1 仿真的三个阶段

7.2.2 仿真的注意事项

7.2.3 仿真程序结构

7.3 使用TestBench对设计进行仿真

7.4 双向总线信号仿真

7.5 基于TEXTIO的交互式仿真

7.5.1 基于TEXTIO的测试程序

7.5.2 TEXTIO

7.5.3 基于TEXTIO的交互式仿真实例

7.6 几种常用的FPGA系数表文件产生方法

7.6.1 滤波器系数表

7.6.2 RAM系数表

7.6.3 I/O文件

第8章 综合实例——数字DBF系统

8.1 系统实现结构

8.1.1 实现分区

8.2 数字下变频 ( DDC )

8.3 数据传输模块

8.4 波束形成的实现

附录A 类型转换

附录B 文件操作

附录C 常用元件的规范化设计示例

附录D FPGA设计流程

参考文献



## 章节摘录

版权页：插图：第1章 概述 1.1 可编程逻辑器件与PLD开发工具 1.1.1 可编程逻辑器件 可编程逻辑器件（Programmable Logic Device, 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，并且可以反复擦写。

在修改和升级PLD时，不需要额外地改变PCB，只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，形成了巨大的PLD产业规模。

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）及FPGA。

它们内部结构的实现方法不相同。

可编程逻辑器件按照颗粒度可以分为3类：小颗粒度（如“门海（Sea of Gates）”架构）、中等颗粒度（如FPGA）和大颗粒度（如CPLD）。

按照编程工艺可以分为4类：熔丝（Fuse）和反熔丝（Antifuse）编程器件、可擦除的可编程只读存储器（EPROM）编程器件、电信号可擦除的可编程只读存储器（E2PROM）编程器件（如CPLD）、SRAM编程器件（如FPGA）。

在工艺分类中，前3类为非易失性器件，编程后配置数据保留在器件上；第4类为易失性器件，掉电后配置数据会丢失，因此在每次上电后需要重新进行数据配置。

1.1.2 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1阶段，从20世纪70年代中到80年代中为第2阶段，从20世纪80年代中到90年代末为第3阶段，从20世纪90年代末到目前为第4阶段。

第1阶段的可编程逻辑器件只有简单的可编程只读存储器、紫外线可擦除只读存储器（EPROM）和电信号可擦除的可编程只读存储器3种。

由于结构的限制，它们只能完成简单的数字逻辑功能。



## <<Xilinx FPGA高级设计及应用>>

### 编辑推荐

《Xilinx FPGA高级设计及应用》可作为从事FPGA设计的工程技术人员、硬件工程师和IC工程师的学习、参考用书，也可作为电子信息、通信工程及相关工科专业的教材。  
FPGA（Field Programmable Gate Array）是1984年由Xilinx公司提出的一类半定制通用器件。用户可以通过对FPGA器件的设计及配置来实现所需的逻辑功能。  
相比传统的IC设计来说，一方面基于FPGA的电路设计具有开发时间短、成本低廉、可靠性得到验证、开发风险小等优点；另一方面针对基于SRAM的FPGA具有可重复、在线或离线配置等特点，为设计的更改、升级和重构提供了基础，使得所设计的电路易于升级维护。

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>