

学习资源
见书中
学习说明

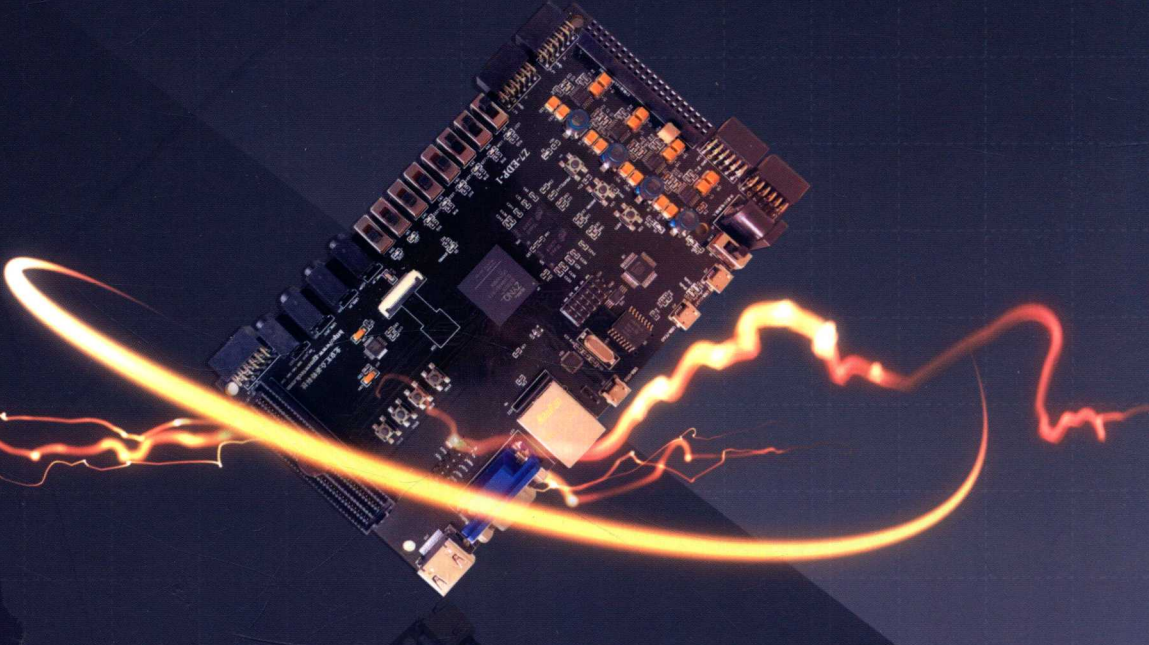
电子信息 · *Ei* 精品

电子系统EDA新技术丛书

Xilinx Zynq-7000 嵌入式系统设计与实现

基于 Arm Cortex-A9 双核处理器和
Vivado 的设计方法 (第二版)

◎ 何 宾 编著

- 
- ★ 将Arm架构知识引入嵌入式系统设计原理的经典著作
 - ★ 将Vivado HLS高层次综合工具引入嵌入式系统设计的经典著作
 - ★ 将Petalinux工具引入Zynq-7000嵌入式系统设计的经典著作
 - ★ 将Python语言引入嵌入式系统应用开发的经典著作



中国工信出版集团



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
<http://www.phei.com.cn>

■ 学习资源

本书以Xilinx公司的Zynq-7000 SoC器件、Vivado集成开发环境，以及PetaLinux工具为主线，全面系统地介绍了嵌入式系统设计的完整设计流程。作者以本书为核心，构建了公开教学视频、设计案例代码、教学课件、QQ交流群等学习资源，以方便广大读者与作者交流互动。

■ 本书特色

- ▶ **知识全面** 本书内容涵盖 Arm 架构基础、Arm Cortex-A9 双核处理器的架构、汇编指令集、片上存储器系统、GPIO、中断、定时器、DMA 和外设等关键知识点。以 Xilinx 的 Vivado 集成开发环境和 PetaLinux 工具为设计平台，全面系统地说明了在 Xilinx Zynq-7000 SoC 平台上实现嵌入式系统设计的方法，对裸机环境和 Linux 环境下的嵌入式实现流程进行了详细说明。在本书中详细介绍了使用 Xilinx HLS 高级综合工具构建硬件加速器的方法。此外，本书还将流行的 Python 语言引入 Zynq-7000 SoC 嵌入式应用开发中，使得读者可以选择使用 C++ 和 Python 语言进行嵌入式应用开发。
- ▶ **内容先进** 在编写本书内容时，参考了 Arm 公司大学计划提供的 Cortex-A9 单核 / 双核处理器的教学资料，以及 Xilinx 公司大学计划提供的 Zynq-7000 SoC 嵌入式设计教学资料，力图全面反映全球新的嵌入式系统设计理论和实现方法。
- ▶ **实例丰富** 以 Xilinx 公司的 Vivado 集成开发环境为平台，通过大量的设计实例，详细说明了 Cortex-A9 嵌入式系统的设计和实现方法。全书实例近40个，可以满足嵌入式系统教学和自学的需求。
- ▶ **软硬融合** 在编写本书的过程中，特别强调软件与硬件协同设计、协同仿真和协同调试的嵌入式系统设计新方法。同时，也突出体现了以 IP 核为中心的系统级软件与硬件相融合的设计思想。

■ 作者简介

何宾 著名的嵌入式技术和EDA技术专家，长期从事电子设计自动化方面的教学和科研工作，与全球多家知名的半导体厂商和EDA工具厂商大学计划保持紧密合作。目前已经出版嵌入式和EDA方面的著作近60部，内容涵盖电路仿真、电路设计、可编程逻辑器件、数字信号处理、单片机、嵌入式系统、片上可编程系统等。典型的代表作有《Xilinx FPGA设计权威指南》、《Altium Designer 13.0电路设计、仿真与验证权威指南》、《Xilinx FPGA数字设计：从门级到行为级的双重HDL描述》、《Xilinx FPGA数字信号处理权威指南：从HDL到模型和IC的描述》、《Altium Designer 15.0电路仿真、设计、验证与工艺实现权威指南》、《STC单片机C语言程序设计》、《Cypress WICED物联网开发指南：从传感器、无线接入到云端的设计与实现》，以及《模拟电子系统设计指南（基础篇）：从半导体、分立元件到ADI集成电路的分析与实现》。



责任编辑：张 迪
封面设计：徐海燕

ISBN 978-7-121-37471-5



9 787121 374715 >

定价：179.00 元

电子系统 EDA 新技术丛书

Xilinx Zynq - 7000 嵌入式系统设计与实现

基于 Arm Cortex - A9 双核处理器和 Vivado 的设计方法

(第二版)

何 宾 编著

贵州师范学院内部使用

电子工业出版社

Publishing House of Electronics Industry

北京 · BEIJING

内 容 简 介

本书是在作者已经出版的《Xilinx Zynq - 7000 嵌入式系统设计与实现：基于 ARM Cortex - A9 双核处理器和 Vivado 的设计方法》一书的基础上进行修订而成的。本书修订后的内容增加到 30 章。修订后，本书的一大特色就是加入了 Arm 架构及分类、使用 PetaLinux 工具在 Zynq - 7000 SoC 上搭建 Ubuntu 操作系统，以及在 Ubuntu 操作系统环境下搭建 Python 语言开发环境，并使用 Python 语言开发应用程序的内容。本书修订后，进一步降低了读者学习 Arm Cortex - A9 嵌入式系统的门槛，并引入了在 Zynq - 7000 SoC 上搭建 Ubuntu 操作系统的新方法。此外，将流行的 Python 语言引入到 Arm 嵌入式系统中，进一步拓宽了在 Arm 嵌入式系统上开发应用程序的方法。

本书可以作为学习 Arm Cortex - A9 处理器嵌入式开发、Xilinx Zynq - 7000 SoC 嵌入式开发，以及在 Arm 嵌入式系统上使用 Python 语言开发嵌入式应用程序的教材、工程参考用书。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目 (CIP) 数据

Xilinx Zynq-7000 嵌入式系统设计与实现：基于 Arm Cortex-A9 双核处理器和 Vivado 的设计方法/何宾编著. —2 版. —北京：电子工业出版社，2019.11

(电子系统 EDA 新技术丛书)

ISBN 978-7-121-37471-5

I. ①X… II. ①何… III. ①可编程序逻辑器件—系统设计 IV. ①TP332.1

中国版本图书馆 CIP 数据核字 (2019) 第 209107 号

责任编辑：张迪 (zhangdi@phei.com.cn)

印 刷：三河市良远印务有限公司

装 订：三河市良远印务有限公司

出版发行：电子工业出版社

北京市海淀区万寿路 173 信箱 邮编：100036

开 本：787×1092 1/16 印张：47.5 字数：1216 千字

版 次：2016 年 7 月第 1 版

2019 年 11 月第 2 版

印 次：2019 年 11 月第 1 次印刷

定 价：179.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：(010) 88254888, 88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：(010) 88254469, zhangdi@phei.com.cn。

学习说明

Study Shows

本书提供的教学视频、教学课件、设计文件、硬件原理图、使用说明下载地址
北京汇众新特科技有限公司技术支持网址：

<http://www.edawiki.com>

注意：所有教学课件及工程文件仅限购买本书读者学习使用，不得以任何方式传播！

本书作者联络方式

电子邮件：hb@gpnewtech.com

购买本书配套 Z7-EDP-1 硬件开发平台事宜由北京汇众新特科技有限公司负责

公司官网：<http://www.gpnewtech.com>

市场及服务支持热线：010-83139176，010-83139076

何宾老师的微信公众号



前 言

本书是在作者已经出版的《Xilinx Zynq - 7000 嵌入式系统设计与实现：基于 ARM Cortex - A9 双核处理器和 Vivado 的设计方法》一书的基础上修订而成的。主要修订内容包括：

(1) 删除原书第一章 Zynq - 7000 SoC 的 Vivado 设计流程一节的内容，增加了 Arm 架构及分类一节的内容，系统介绍了 Arm 架构与不同处理器 IP 之间的关系，使得读者能更加系统地了解并掌握 Arm 架构与 Cortex -M、Cortex-R 和 Cortex-A 框架下不同处理器 IP 之间的对应关系。

(2) 本书增加了第 30 章，该章主要介绍了 PetaLinux 2018.2 工具，以及通过该工具在 Xilinx SoC 器件上构建 Ubuntu 操作系统环境的方法。需要指出，与本书前面使用传统的方法搭建 Ubuntu 操作系统环境相比，通过 Xilinx 公司自己的 PetaLinux 工具，使得在 Xilinx Zynq - 7000 SoC，以及在 UltraScale MPSoC 上搭建 Ubuntu 操作系统环境更加便捷高效，显著降低了构建操作系统运行环境的难度，更加有利于初学者的入门学习。

(3) 作为本书的亮点之一，在使用 PetaLinux 工具构建嵌入式操作系统运行环境的基础上，构建了嵌入式 Python 语言开发环境，并在所构建的环境下使用 Python 语言开发了应用程序。为了比较 Python 语言在 Arm Cortex - A9 嵌入式处理器上的运行性能，同时在 PC/笔记本电脑上也构建了 Python 语言开发环境，并使用 Python 语言开发了应用程序。因此，实现了 Zynq - 7000 SoC 作为服务器，以及 PC/笔记本电脑作为客户端的网络交互和数据传输功能。

(4) 修改并更正了书中的一些错误。

考虑到 Vivado 集成开发环境的主要功能和开发界面没有明显的不同。因此，书中所有案例仍然保留原来的开发版本。但是，增加的一章内容使用了较新的 PetaLinux 2018.2 环境。

总体来说，通过本书这一次的修订，使得介绍嵌入式系统设计的内容更加条理化，读者更容易系统学习基于 Arm Cortex - A9 处理器的嵌入式系统硬件和软件的设计方法，进一步降低了学习 Arm 嵌入式系统和 Xilinx Zynq - 7000 SoC 的难度。

由于作者水平有限，书中难免有不足之处，恳请读者批评指正，以帮助读者今后进一步提高图书的编写质量。同时，也要感谢电子工业出版社的张迪编辑为本书的出版所做出的辛勤工作，还要感谢作者曾经带过并且已经毕业的研究生张艳辉，协助作者修订了本书，使得作者可以将更加精彩的内容奉献给广大读者。

编著者

2019.09 于北京

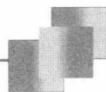
目 录

第1章	Zynq - 7000 SoC 设计导论	1
1.1	全可编程片上系统基础知识	1
1.1.1	全可编程片上系统的演进	1
1.1.2	SoC 与 MCU 和 CPU 的比较	3
1.1.3	全可编程 SoC 诞生的背景	4
1.1.4	可编程 SoC 系统技术特点	5
1.1.5	全可编程片上系统中的处理器类型	5
1.2	Arm 架构及分类	6
1.2.1	M - Profile	7
1.2.2	R - Profile	9
1.2.3	A - Profile	10
1.3	Zynq - 7000 SoC 功能和结构	11
1.3.1	Zynq - 7000 SoC 产品分类及资源	12
1.3.2	Zynq - 7000 SoC 的功能	12
1.3.3	Zynq - 7000 SoC 处理系统的构成	14
1.3.4	Zynq - 7000 SoC 可编程逻辑的构成	19
1.3.5	Zynq - 7000 SoC 内的互联结构	20
1.3.6	Zynq - 7000 SoC 的供电引脚	22
1.3.7	Zynq - 7000 SoC 内 MIO 到 EMIO 的连接	23
1.3.8	Zynq - 7000 SoC 内为 PL 分配的信号	28
1.4	Zynq - 7000 SoC 在嵌入式系统中的优势	30
1.4.1	使用 PL 实现软件算法	30
1.4.2	降低功耗	32
1.4.3	实时减负	33
1.4.4	可重配置计算	34
第2章	AMBA 规范	35
2.1	AMBA 规范及发展	35
2.1.1	AMBA 1	36

2.1.2	AMBA 2	36
2.1.3	AMBA 3	36
2.1.4	AMBA 4	37
2.1.5	AMBA 5	38
2.2	AMBA APB 规范	40
2.2.1	AMBA APB 写传输	40
2.2.2	AMBA APB 读传输	42
2.2.3	AMBA APB 错误响应	43
2.2.4	操作状态	44
2.2.5	AMBA 3 APB 信号	44
2.3	AMBA AHB 规范	45
2.3.1	AMBA AHB 结构	45
2.3.2	AMBA AHB 操作	46
2.3.3	AMBA AHB 传输类型	48
2.3.4	AMBA AHB 猝发操作	50
2.3.5	AMBA AHB 传输控制信号	53
2.3.6	AMBA AHB 地址译码	54
2.3.7	AMBA AHB 从设备传输响应	55
2.3.8	AMBA AHB 数据总线	58
2.3.9	AMBA AHB 传输仲裁	59
2.3.10	AMBA AHB 分割传输	64
2.3.11	AMBA AHB 复位	67
2.3.12	关于 AHB 数据总线的位宽	67
2.3.13	AMBA AHB 接口设备	68
2.4	AMBA AXI4 规范	69
2.4.1	AMBA AXI4 概述	69
2.4.2	AMBA AXI4 功能	70
2.4.3	AMBA AXI4 互联结构	78
2.4.4	AXI4 - Lite 功能	79
2.4.5	AXI4 - Stream 功能	80

第 3 章 Zynq - 7000 系统公共资源及特性 83

3.1	时钟子系统	83
3.1.1	时钟子系统架构	83
3.1.2	CPU 时钟域	84
3.1.3	时钟编程实例	86
3.1.4	时钟子系统内的生成电路结构	87



3.2 复位子系统	91
3.2.1 复位子系统结构和层次	92
3.2.2 复位流程	93
3.2.3 复位的结果	94

第4章 Zynq 调试和测试子系统 95

4.1 JTAG 和 DAP 子系统	95
4.1.1 JTAG 和 DAP 子系统功能	97
4.1.2 JTAG 和 DAP 子系统 I/O 信号	99
4.1.3 编程模型	99
4.1.4 Arm DAP 控制器	101
4.1.5 跟踪端口接口单元 (TPIU)	102
4.1.6 Xilinx TAP 控制器	102
4.2 CoreSight 系统结构及功能	103
4.2.1 CoreSight 结构概述	103
4.2.2 CoreSight 系统功能	104

第5章 Cortex - A9 处理器及指令集 107

5.1 应用处理单元概述	107
5.1.1 基本功能	107
5.1.2 系统级视图	108
5.2 Cortex - A9 处理器结构	110
5.2.1 处理器模式	111
5.2.2 寄存器	113
5.2.3 流水线	118
5.2.4 分支预测	118
5.2.5 指令和数据对齐	119
5.2.6 跟踪和调试	121
5.3 Cortex - A9 处理器指令集	122
5.3.1 指令集基础	122
5.3.2 数据处理操作	125
5.3.3 存储器指令	130
5.3.4 分支	131
5.3.5 饱和算术	133
5.3.6 杂项指令	134

第6章	Cortex - A9 片上存储器系统结构和功能	138
6.1	L1 高速缓存	138
6.1.1	高速缓存背景	138
6.1.2	高速缓存的优势和问题	139
6.1.3	存储器层次	140
6.1.4	高速缓存结构	140
6.1.5	缓存策略	145
6.1.6	写和取缓冲区	147
6.1.7	缓存性能和命中速度	147
6.1.8	无效和清除缓存	147
6.1.9	一致性点和统一性点	149
6.1.10	Zynq - 7000 中 Cortex - A9 L1 高速缓存的特性	151
6.2	存储器顺序	153
6.2.1	普通、设备和强顺序存储器模型	154
6.2.2	存储器属性	155
6.2.3	存储器屏障	155
6.3	存储器管理单元	159
6.3.1	MMU 功能描述	160
6.3.2	虚拟存储器	161
6.3.3	转换表	162
6.3.4	页表入口域的描述	165
6.3.5	TLB 构成	167
6.3.6	存储器访问顺序	169
6.4	侦听控制单元	170
6.4.1	地址过滤	171
6.4.2	SCU 主设备端口	171
6.5	L2 高速缓存	171
6.5.1	互斥 L2 - L1 高速缓存配置	173
6.5.2	高速缓存替换策略	174
6.5.3	高速缓存锁定	174
6.5.4	使能/禁止 L2 高速缓存控制器	176
6.5.5	RAM 访问延迟控制	176
6.5.6	保存缓冲区操作	176
6.5.7	在 Cortex - A9 和 L2 控制器之间的优化	177
6.5.8	预取操作	178
6.5.9	编程模型	179



6.6	片上存储器	180
6.6.1	片上存储器概述	180
6.6.2	片上存储器功能	181
6.7	系统地址分配	186
6.7.1	地址映射	186
6.7.2	系统总线主设备	188
6.7.3	I/O 外设	188
6.7.4	SMC 存储器	188
6.7.5	SLCR 寄存器	188
6.7.6	杂项 PS 寄存器	189
6.7.7	CPU 私有寄存器	189

第 7 章 Zynq - 7000 SoC 的 Vivado 基本设计流程

7.1	创建新的工程	190
7.2	使用 IP 集成器创建处理器系统	192
7.3	生成顶层 HDL 并导出设计到 SDK	197
7.4	创建应用测试程序	199
7.5	设计验证	202
7.5.1	验证前的硬件平台准备	202
7.5.2	设计验证的具体实现	203
7.6	SDK 调试工具的使用	205
7.6.1	打开前面的设计工程	205
7.6.2	导入工程到 SDK	205
7.6.3	建立新的存储器测试工程	205
7.6.4	运行存储器测试工程	206
7.6.5	调试存储器测试工程	207
7.7	SDK 性能分析工具	209

第 8 章 Arm GPIO 的原理和控制实现

8.1	GPIO 模块原理	213
8.1.1	GPIO 接口及功能	214
8.1.2	GPIO 编程流程	217
8.1.3	I/O 接口	218
8.1.4	部分寄存器说明	218
8.1.5	底层读/写函数说明	220
8.1.6	GPIO 的 API 函数说明	220

8.2	Vivado 环境下 MIO 读/写控制的实现	221
8.2.1	调用底层读/写函数编写 GPIO 应用程序	221
8.2.2	调用 API 函数编写控制 GPIO 应用程序	224
8.3	Vivado 环境下 EMIO 读/写控制的实现	226
8.3.1	调用底层读/写函数编写 GPIO 应用程序	227
8.3.2	调用 API 函数编写控制 GPIO 应用程序	232

第 9 章 Cortex - A9 异常与中断原理及实现 236

9.1	异常原理	236
9.1.1	异常类型	237
9.1.2	异常处理	241
9.1.3	其他异常句柄	242
9.1.4	Linux 异常程序流	243
9.2	中断原理	244
9.2.1	外部中断请求	244
9.2.2	Zynq - 7000 SoC 内的中断环境	247
9.2.3	中断控制器的功能	248
9.3	Vivado 环境下中断系统的实现	252
9.3.1	Cortex - A9 处理器中断及异常初始化流程	252
9.3.2	Cortex - A9 GPIO 控制器初始化流程	252
9.3.3	导出硬件设计到 SDK	253
9.3.4	创建新的应用工程	253
9.3.5	运行应用工程	256

第 10 章 Cortex - A9 定时器原理及实现 257

10.1	定时器系统架构	257
10.1.1	CPU 私有定时器和看门狗定时器	257
10.1.2	全局定时器/计数器	258
10.1.3	系统级看门狗定时器	259
10.1.4	3 重定时器/计数器	261
10.1.5	I/O 信号	264
10.2	Vivado 环境下定时器的控制实现	264
10.2.1	打开前面的设计工程	265
10.2.2	创建 SDK 软件工程	265
10.2.3	运行软件应用工程	267

第 11 章	Cortex - A9 DMA 控制器原理及实现	268
11.1	DMA 控制器架构	268
11.2	DMA 控制器功能	271
11.2.1	考虑 AXI 交易的因素	272
11.2.2	DMA 管理器	273
11.2.3	多通道数据 FIFO (MFIFO)	274
11.2.4	存储器—存储器交易	274
11.2.5	PL 外设 AXI 交易	274
11.2.6	PL 外设请求接口	275
11.2.7	PL 外设长度管理	276
11.2.8	DMAC 长度管理	277
11.2.9	事件和中断	278
11.2.10	异常终止	278
11.2.11	安全性	280
11.2.12	IP 配置选项	282
11.3	DMA 控制器编程指南	282
11.3.1	启动控制器	282
11.3.2	执行 DMA 传输	282
11.3.3	中断服务例程	282
11.3.4	寄存器描述	283
11.4	DMA 引擎编程指南	284
11.4.1	写微代码编程用于 AXI 交易的 CCRx	284
11.4.2	存储器到存储器传输	284
11.4.3	PL 外设 DMA 传输长度管理	287
11.4.4	使用一个事件重新启动 DMA 通道	289
11.4.5	中断一个处理器	289
11.4.6	指令集参考	290
11.5	编程限制	291
11.6	系统功能之控制器复位配置	292
11.7	I/O 接口	293
11.7.1	AXI 主接口	293
11.7.2	外设请求接口	293
11.8	Vivado 环境下 DMA 传输的实现	294
11.8.1	DMA 控制器初始化流程	295
11.8.2	中断控制器初始化流程	295
11.8.3	中断服务句柄处理流程	296

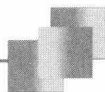
11.8.4	导出硬件设计到 SDK	296
11.8.5	创建新的应用工程	297
11.8.6	运行软件应用工程	303

第12章 Cortex - A9 安全性扩展 305

12.1	TrustZone 硬件架构	305
12.1.1	多核系统的安全性扩展	307
12.1.2	普通世界和安全世界的交互	307
12.2	Zynq - 7000 APU 内的 TrustZone	308
12.2.1	CPU 安全过渡	309
12.2.2	CP15 寄存器访问控制	310
12.2.3	MMU 安全性	310
12.2.4	L1 缓存安全性	311
12.2.5	安全异常控制	311
12.2.6	CPU 调试 TrustZone 访问控制	311
12.2.7	SCU 寄存器访问控制	312
12.2.8	L2 缓存中的 TrustZone 支持	312

第13章 Cortex - A9 NEON 原理及实现 313

13.1	SIMD	313
13.2	NEON 架构	315
13.2.1	与 VFP 的共性	315
13.2.2	数据类型	316
13.2.3	NEON 寄存器	316
13.2.4	NEON 指令集	318
13.3	NEON C 编译器和汇编器	319
13.3.1	向量化	319
13.3.2	检测 NEON	319
13.4	NEON 优化库	320
13.5	SDK 工具提供的优化选项	321
13.6	使用 NEON 内联函数	324
13.6.1	NEON 数据类型	325
13.6.2	NEON 内联函数	325
13.7	优化 NEON 汇编器代码	327
13.8	提高存储器访问效率	328
13.9	自动向量化实现	329



13.9.1	导出硬件设计到 SDK	329
13.9.2	创建新的应用工程	330
13.9.3	运行软件应用工程	331
13.10	NEON 汇编代码实现	331
13.10.1	导出硬件设计到 SDK	331
13.10.2	创建新的应用工程	332
13.10.3	运行软件应用工程	333
第14章	Cortex - A9 外设模块结构及功能	334
14.1	DDR 存储器控制器	334
14.1.1	DDR 存储器控制器接口及功能	335
14.1.2	AXI 存储器接口	337
14.1.3	DDR 核和交易调度器	338
14.1.4	DDRC 仲裁	338
14.1.5	DDR 存储器控制器 PHY	340
14.1.6	DDR 初始化和标定	340
14.1.7	纠错码	341
14.2	静态存储器控制器	342
14.2.1	静态存储器控制器接口及功能	343
14.2.2	静态存储器控制器和存储器的信号连接	344
14.3	四 - SPI Flash 控制器	345
14.3.1	四 - SPI Flash 控制器功能	347
14.3.2	四 - SPI Flash 控制器反馈时钟	349
14.3.3	四 - SPI Flash 控制器接口	349
14.4	SD/SDIO 外设控制器	351
14.4.1	SD/SDIO 控制器功能	352
14.4.2	SD/SDIO 控制器传输协议	353
14.4.3	SD/SDIO 控制器端口信号连接	356
14.5	USB 主机、设备和 OTG 控制器	356
14.5.1	USB 控制器接口及功能	358
14.5.2	USB 主机操作模式	361
14.5.3	USB 设备操作模式	363
14.5.4	USB OTG 操作模式	365
14.6	吉比特以太网控制器	365
14.6.1	吉比特以太网控制器接口及功能	367
14.6.2	吉比特以太网控制器接口编程向导	368
14.6.3	吉比特以太网控制器接口信号连接	372



14.7	SPI 控制器	373
14.7.1	SPI 控制器的接口及功能	374
14.7.2	SPI 控制器时钟设置规则	376
14.8	CAN 控制器	376
14.8.1	CAN 控制器接口及功能	377
14.8.2	CAN 控制器操作模式	379
14.8.3	CAN 控制器消息保存	380
14.8.4	CAN 控制器接收过滤器	381
14.8.5	CAN 控制器编程模型	382
14.9	UART 控制器	383
14.10	I ² C 控制器	387
14.10.1	I ² C 速度控制逻辑	388
14.10.2	I ² C 控制器的功能和工作模式	388
14.11	XADC 转换器接口	390
14.11.1	XADC 转换器接口及功能	391
14.11.2	XADC 命令格式	392
14.11.3	供电传感器报警	392
14.12	PCI - E 接口	393

第15章 Zynq - 7000 内的可编程逻辑资源

15.1	可编程逻辑资源概述	395
15.2	可编程逻辑资源功能	396
15.2.1	CLB、Slice 和 LUT	396
15.2.2	时钟管理	396
15.2.3	块 RAM	398
15.2.4	数字信号处理	398
15.2.5	输入/输出	399
15.2.6	低功耗串行收发器	400
15.2.7	PCI - E 模块	401
15.2.8	XADC (模拟 - 数字转换器)	402
15.2.9	配置	402

第16章 Zynq - 7000 内的互联结构

16.1	系统互联架构	404
16.1.1	互联模块及功能	404
16.1.2	数据路径	406



16.1.3	时钟域	407
16.1.4	连接性	408
16.1.5	AXI ID	409
16.1.6	寄存器概述	409
16.2	服务质量	410
16.2.1	基本仲裁	410
16.2.2	高级 QoS	410
16.2.3	DDR 端口仲裁	411
16.3	AXI_HP 接口	411
16.3.1	AXI_HP 接口结构及特点	411
16.3.2	接口数据宽度	415
16.3.3	交易类型	416
16.3.4	命令交替和重新排序	416
16.3.5	性能优化总结	416
16.4	AXI_ACP 接口	417
16.5	AXI_GP 接口	418
16.6	AXI 信号总结	418
16.7	PL 接口选择	422
16.7.1	使用通用主设备端口的 Cortex - A9	423
16.7.2	通过通用主设备的 PS DMA 控制器 (DMAC)	423
16.7.3	通过高性能接口的 PL DMA	426
16.7.4	通过 AXI ACP 的 PL DMA	426
16.7.5	通过通用 AXI 从 (GP) 的 PL DMA	426
第 17 章	Zynq - 7000 SoC 内定制简单 AXI - Lite IP	429
17.1	设计原理	429
17.2	定制 AXI - Lite IP	429
17.2.1	创建定制 IP 模板	429
17.2.2	修改定制 IP 设计模板	432
17.2.3	使用 IP 封装器封装外设	436
17.3	打开并添加 IP 到设计中	440
17.3.1	打开工程和修改设置	440
17.3.2	添加定制 IP 到设计	442
17.3.3	添加 XDC 约束文件	445
17.4	导出硬件到 SDK	446
17.5	建立和验证软件应用工程	446
17.5.1	建立应用工程	447