



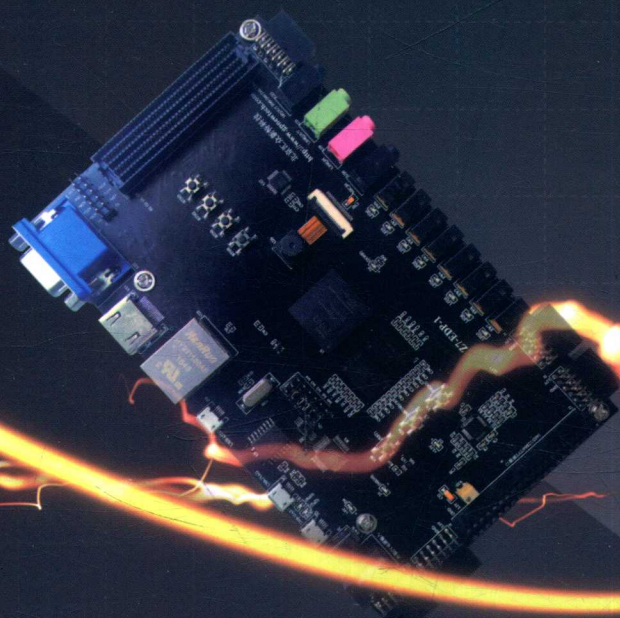
Xilinx 大学合作计划推荐用书
ARM 大学合作计划推荐用书

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（立体化教程）

◎ 何 宾 张艳辉 编著



- ★ 详细介绍在 Vivado 2015.4 集成开发环境下嵌入式系统设计方法的经典著作
- ★ 深度融合软硬件协同设计、协同仿真和协同调试的经典著作
- ★ 将 Vivado HLS 高层次综合工具引入嵌入系统设计的经典著作

 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>

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

何 宾 张艳辉 编著

電子工業出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

本书是在作者已经出版的《Xilinx All Programmable Zynq - 7000 SoC 设计指南》一书的基础上进行大幅度修订而成的。本书的一大特色就是更加突出 ARM Cortex - A9 双核处理器的使用。此外,在修订本书时采用了 Xilinx 最新的 Vivado2015.4 集成开发环境。通过本书的修订,能反映最新的 ARM 嵌入式设计技术和实现方法,同时也能更加凸显采用异构架构的 Zynq - 7000 SoC 在高性能数据处理尤其是在图像处理中的巨大优势。

本书可以作为学习 ARM Cortex - A9 处理器嵌入式开发,以及 Xilinx Zynq - 7000 SoC 嵌入式开发的教材、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,张艳辉编著. —北京:电子工业出版社,2016.7

(电子系统 EDA 新技术丛书)

ISBN 978 - 7 - 121 - 28995 - 8

I. ①X… II. ①何… ②张… III. ①可程序逻辑器件 - 系统设计 IV. ①TP332.1

中国版本图书馆 CIP 数据核字 (2016) 第 124535 号

策划编辑:王敬栋 (wangjd@phei.com.cn)

责任编辑:王敬栋

印 刷:北京京科印刷有限公司

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 印张:46.5 字数:1190 千字

版 次:2016 年 7 月第 1 版

印 次:2016 年 7 月第 1 次印刷

印 数:3 000 册 定价:128.00 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 - 88254451

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

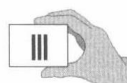
电子邮件: hb@gpnewtech.com

购买硬件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 All Programmable Zynq - 7000 SoC 设计指南》一书的基础上进行大幅度修订而成的。在本书修订的过程中，汲取了广大读者的参考意见。修订后本书的特色主要包含：

(1) 本书大幅度增加了 ARM Cortex - A9 基本知识的讲解，包括指令集、处理器架构等方面的内容，降低读者阅读本书的入门门槛；

(2) 本书大幅度增加了 ARM Cortex - A9 应用和设计实例，包括 MIO 和 EMIO、定时器、中断、DMA、NEON 等，帮助读者进一步全面掌握 ARM Cortex - A9 处理器的使用方法；

(3) 本书开发环境使用了 Xilinx 最新的 Vivado2015.4 集成开发环境，使得本书能反映出 Vivado 集成开发环境的最新特性；

(4) 本书大幅度增加了 Ubuntu 操作系统在嵌入式系统中的应用实例，使得读者系统掌握 Ubuntu 构建、驱动程序开发，以及 Qt 应用程序开发的全部流程。

在修订本书的过程中，有更多新的体会，在此愿意与广大读者分享。

(1) Xilinx 推出的 Zynq - 7000 SoC 器件是异构架构的典型代表，即把专用的嵌入式 ARM Cortex - A9 双核处理器和通用的大规模现场可编程门阵列 FPGA 结合在单个芯片中，将专用处理器的串行执行和 FPGA 的并行执行完美结合，为解决未来大数据处理、人工智能等复杂高性能算法处理铺平了道路。

(2) 新的设计工具的推出，比如 Vivado HLS、SDSoC 等，更加注重嵌入式系统的系统级建模，对未来设计方法将产生深远的影响。通过 HLS 工具，用户只需要编写 C 语言代码，就可以让工具自动转换和生成 HDL 代码，显著地提高了设计效率，缩短了开发周期。

(3) 随着异构架构和片上系统技术的不断发展，协同设计、协同仿真和协同调试将成为未来嵌入式系统开发者必须具备的品质。所谓的协同，要求设计者必须同时掌握软件和硬件知识，这与传统上软件和硬件分离的设计方法有着本质的区别。

(4) 特别值得一提的是，在 2016 年初 Xilinx 最新量产的 Zynq UltraScale + MPSoC 采用了台积电的 16nm 工艺，将 ARM 的 Cortex - A53 四核处理器、ARM 的 Cortex - R5 双核处理器、ARM 的 Mali - 400 MP2 GPU 及超大规模 FPGA 资源集成在单芯片中，为未来高性能数据中心提供强大的支持，进一步改善了数据中心的处理能力，可以预见越来越多的搜索引擎公司、电商平台及互联网企业等都会使用这种平台加速海量数据的处理。

(5) 由于半导体技术的不断发展，使得电子系统从传统的 PCB 板级进化到了芯片级，这对嵌入式系统的小型化、低功耗和可靠性的改善都提供了强大的保障，这也是未来嵌入式系统发展的潮流。

本书在修订的过程中，突出体现 ARM 的嵌入式和 Xilinx 最新的 Vivado 设计工具，所涵盖的内容是作者所出版图书中最多的，全书内容达到 29 章之多。在编排本书内容时，分成



下面几大部分。

(1) 设计导论是本书中最基础的内容，目的要介绍 Zynq - 7000 SoC 的架构、优势、设计方法，以及 Vivado2015.4 集成开发环境流程。

(2) ARM AMBA 规范是读者理解和掌握 ARM Cortex - A9 架构必须要知道的基本知识，掌握这些知识对于读者能顺利学习本书后续章节非常关键。

(3) 系统介绍 Zynq - 7000 SoC 内 PS 所有功能部件的原理及使用方法，内容包括：Cortex - A9 处理器架构、Cortex - A9 指令集、片上存储器系统、设计流程、MIO/EMIO 操作、中断和异常、定时器、DMA、安全性扩展、NEON、外设模块。

(4) 系统介绍 Zynq - 7000 SoC 内 PL 的资源及 Zynq - 7000 SoC 内的互联结构，使得读者可以清楚地了解并掌握在异构架构下，Cortex - A9 专用处理器与 PL 内定制外设实现满足不同要求的连接方法。

(5) 系统介绍在 Zynq - 7000 SoC 内 PL 通过 GP、HP 和 ACP，构建不同定制外设，满足不同数据处理和传输性能要求的方法。

(6) 系统介绍基于 Ubuntu 操作系统构建嵌入式系统的方法，包括：Linux 开发环境的构建、Zynq - 7000 SoC 内 Ubuntu 硬件运行环境的构建、Zynq - 7000 SoC 内 Ubuntu 软件运行环境的构建、驱动程序的开发以及基于 Ubuntu 构建图形处理系统。

在编写本书的过程中，特别感谢 Xilinx 公司大中华区大学计划经理陆家华、ARM 公司亚太区大学计划经理陈炜、Xilinx 公司亚太区传媒经理张俊伟给予的大力支持和帮助，也感谢 Xilinx 公司各位热心的技术支持给予的无私帮助。集宁师范学院物理系聂阳老师参与编写本书的第 1 ~ 3 章，作者的研究生张艳辉帮助作者设计和验证本书所有章节的设计案例，研究生李宝隆编写了本书第 4 ~ 5 章的内容，本科生汤宗美参与整理并编写了本书所有配套的教学资源。在本书出版的过程中，得到电子工业出版社领导和编辑的大力支持和帮助，在此也一并向他们表示感谢。

何宾

2016 年 5 月于北京

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 Zynq-7000 SoC 功能和结构	6
1.2.1 Zynq-7000 SoC 产品分类及资源	6
1.2.2 Zynq-7000 SoC 的功能	7
1.2.3 Zynq-7000 SoC 处理系统 PS 的构成	8
1.2.4 Zynq-7000 SoC 可编程逻辑 PL 的构成	13
1.2.5 Zynq-7000 SoC 内的互联结构	14
1.2.6 Zynq-7000 SoC 的供电引脚	16
1.2.7 Zynq-7000 SoC 内 MIO 到 EMIO 的连接	17
1.2.8 Zynq-7000 SoC 内为 PL 分配的信号	22
1.3 Zynq-7000 SoC 在嵌入式系统中的优势	24
1.3.1 使用 PL 实现软件算法	24
1.3.2 降低功耗	26
1.3.3 实时减负	27
1.3.4 可重配置计算	28
1.4 Zynq-7000 SoC 的 Vivado 设计流程	28
1.4.1 Vivado 的 IP 设计和系统级设计集成	29
1.4.2 使用 RTL 或网表的设计流程	29
1.4.3 IP subsystem 设计	30
1.4.4 嵌入式处理器硬件设计	30
1.4.5 使用模型和高级综合的 DSP 设计	31
1.4.6 脱离上下文的设计流程	31
1.4.7 I/O 引脚规划和布局	31
1.4.8 设计分析和验证	32
1.4.9 器件编程和硬件验证	32
1.4.10 部分可重配置	32
第 2 章 AMBA 协议规范	33
2.1 AMBA 规范概述	33
2.2 AMBA APB 规范	34

2.2.1	AMBA APB 写传输	34
2.2.2	AMBA APB 读传输	36
2.2.3	AMBA APB 错误响应	37
2.2.4	操作状态	38
2.2.5	AMBA3 APB 信号	38
2.3	AMBA AHB 规范	39
2.3.1	AMBA AHB 结构	39
2.3.2	AMBA AHB 操作	40
2.3.3	AMBA AHB 传输类型	42
2.3.4	AMBA AHB 猝发操作	44
2.3.5	AMBA AHB 传输控制信号	47
2.3.6	AMBA AHB 地址译码	48
2.3.7	AMBA AHB 从设备传输响应	49
2.3.8	AMBA AHB 数据总线	52
2.3.9	AMBA AHB 传输仲裁	53
2.3.10	AMBA AHB 分割传输	58
2.3.11	AMBA AHB 复位	61
2.3.12	关于 AHB 数据总线的位宽	61
2.3.13	AMBA AHB 接口设备	62
2.4	AMBA AXI4 规范	64
2.4.1	AMBA AXI4 概述	64
2.4.2	AMBA AXI4 功能	64
2.4.3	AMBA AXI4 互联结构	72
2.4.4	AXI4 - Lite 功能	74
2.4.5	AXI4 - Stream 功能	75

第3章 Zynq -7000 系统公共资源及特性

3.1	时钟子系统	78
3.1.1	时钟系统架构	78
3.1.2	CPU 时钟域	79
3.1.3	时钟编程实例	81
3.1.4	时钟系统内生成电路结构	82
3.2	复位子系统	86
3.2.1	复位系统结构和层次	87
3.2.2	复位流程	88
3.2.3	复位的结果	89

第4章 Zynq 调试和测试子系统

4.1	JTAG 和 DAP 子系统	90
4.1.1	JTAG 和 DAP 系统功能	92
4.1.2	JTAG 和 DAP 系统 I/O 信号	94
4.1.3	编程模型	94

4.1.4	ARM DAP 控制器	96
4.1.5	跟踪端口接口单元 TPIU	97
4.1.6	Xilinx TAP 控制器	97
4.2	CoreSight 系统结构及功能	98
4.2.1	CoreSight 结构概述	98
4.2.2	CoreSight 系统功能	99
第5章	Cortex - A9 处理器及指令集	102
5.1	应用处理单元概述	102
5.1.1	基本功能	102
5.1.2	系统级视图	103
5.2	ARM 处理器架构发展	105
5.3	Cortex - A9 中央处理器结构	106
5.3.1	处理器模式	107
5.3.2	寄存器	109
5.3.3	流水线	115
5.3.4	分支预测	115
5.3.5	指令和数据对齐	116
5.3.6	跟踪和调试	118
5.4	Cortex - A9 处理器指令集	119
5.4.1	指令集基础	119
5.4.2	数据处理操作	123
5.4.3	存储器指令	127
5.4.4	分支	129
5.4.5	饱和算术	130
5.4.6	杂项指令	132
第6章	Cortex - A9 片上存储器系统结构和功能	137
6.1	L1 高速缓存	137
6.1.1	高速缓存背景	137
6.1.2	高速缓存的优势和问题	138
6.1.3	存储器层次	139
6.1.4	高速缓存结构	139
6.1.5	缓存策略	144
6.1.6	写和取缓冲区	146
6.1.7	缓存性能和命中速度	146
6.1.8	无效和清除缓存	147
6.1.9	一致性和统一性点	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	152
6.2.1	普通、设备和强顺序存储器模型	153
6.2.2	存储器属性	154

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	159
6.3.2	虚拟存储器	160
6.3.3	转换表	161
6.3.4	页表入口域的描述	164
6.3.5	TLB 构成	166
6.3.6	存储器访问顺序	168
6.4	侦听控制单元	169
6.4.1	地址过滤	169
6.4.2	SCU 主设备端口	170
6.5	L2 高速缓存	170
6.5.1	互斥 L2 - L1 高速缓存配置	172
6.5.2	高速缓存替换策略	173
6.5.3	高速缓存锁定	173
6.5.4	使能/禁止 L2 高速缓存控制器	175
6.5.5	RAM 访问延迟控制	175
6.5.6	保存缓冲区操作	175
6.5.7	在 Cortex - A9 和 L2 控制器之间的优化	176
6.5.8	预取操作	177
6.5.9	编程模型	177
6.6	片上存储器	178
6.6.1	片上存储器概述	178
6.6.2	片上存储器功能	180
6.7	系统地址分配	185
6.7.1	地址映射	185
6.7.2	系统总线主设备	187
6.7.3	I/O 外设	187
6.7.4	SMC 存储器	187
6.7.5	SLCR 寄存器	187
6.7.6	杂项 PS 寄存器	188
6.7.7	CPU 私有总线寄存器	188

第 7 章	Zynq - 7000 SoC 的 Vivado 基本设计流程	189
7.1	创建新的工程	189
7.2	使用 IP 集成器创建处理器系统	191
7.3	生成顶层 HDL 并导出设计到 SDK	196
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	204



7.6.1	打开前面的设计工程	204
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	213
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	227
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	249
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	253
9.3.2	Cortex - A9 GPIO 控制器初始化流程	253
9.3.3	导出硬件设计到 SDK	253
9.3.4	创建新的应用工程	254
9.3.5	运行应用工程	257
第10章	Cortex - A9 定时器原理及实现	258
10.1	定时器系统架构	258
10.1.1	CPU 私有定时器和看门狗定时器	259

10.1.2	全局定时器/计数器	259
10.1.3	系统看门狗定时器	260
10.1.4	三重定时器/计数器	262
10.1.5	I/O 信号	265
10.2	Vivado 环境下定时器的控制实现	265
10.2.1	打开前面的设计工程	266
10.2.2	创建 SDK 软件工程	266
10.2.3	运行软件应用工程	268

第 11 章	Cortex - A9 DMA 控制器原理及实现	270
11.1	DMA 控制器架构	270
11.2	DMA 控制器功能	273
11.2.1	考虑 AXI 交易的因素	274
11.2.2	DMA 管理器	275
11.2.3	多通道数据 FIFO (MFIFO)	276
11.2.4	存储器—存储器交易	276
11.2.5	PL 外设 AXI 交易	276
11.2.6	PL 外设请求接口	277
11.2.7	PL 外设长度管理	278
11.2.8	DMAC 长度管理	279
11.2.9	事件和中断	280
11.2.10	异常终止	281
11.2.11	安全性	282
11.2.12	IP 配置选项	284
11.3	DMA 控制器编程指南	284
11.3.1	启动控制器	284
11.3.2	执行 DMA 传输	284
11.3.3	中断服务例程	285
11.3.4	寄存器描述	285
11.4	DMA 引擎编程指南	286
11.4.1	写微码编程用于 AXI 交易的 CCRx	286
11.4.2	存储器到存储器传输	286
11.4.3	PL 外设 DMA 传输长度管理	289
11.4.4	使用一个事件重新启动 DMA 通道	291
11.4.5	中断一个处理器	292
11.4.6	指令集参考	292
11.5	编程限制	293
11.6	系统功能之控制器复位配置	295
11.7	I/O 接口	295
11.7.1	AXI 主接口	295
11.7.2	外设请求接口	296
11.8	Vivado 环境下 DMA 传输的实现	296

11.8.1	DMA 控制器初始化流程	297
11.8.2	中断控制器初始化流程	297
11.8.3	中断服务句柄处理流程	298
11.8.4	导出硬件设计到 SDK	298
11.8.5	创建新的应用工程	299
11.8.6	运行软件应用工程	306

第12章 Cortex - A9 安全性扩展

12.1	TrustZone 硬件架构	307
12.1.1	多核系统的安全性扩展	309
12.1.2	普通世界和安全世界的交互	309
12.2	Zynq - 7000 APU 内的 TrustZone	310
12.2.1	CPU 安全过渡	311
12.2.2	CP15 寄存器访问控制	312
12.2.3	MMU 安全性	313
12.2.4	L1 缓存安全性	313
12.2.5	安全异常控制	313
12.2.6	CPU 调试 TrustZone 访问控制	313
12.2.7	SCU 寄存器访问控制	314
12.2.8	L2 缓存中的 TrustZone 支持	314

第13章 Cortex - A9 NEON 原理及实现

13.1	SIMD	315
13.2	NEON 架构	317
13.2.1	与 VFP 的共性	317
13.2.2	数据类型	318
13.2.3	NEON 寄存器	318
13.2.4	NEON 指令集	320
13.3	NEON C 编译器和汇编器	321
13.3.1	向量化	321
13.3.2	检测 NEON	321
13.4	NEON 优化库	322
13.5	SDK 工具提供的优化选项	323
13.6	使用 NEON 内联函数	326
13.6.1	NEON 数据类型	327
13.6.2	NEON 内联函数	328
13.7	优化 NEON 汇编器代码	329
13.8	提高存储器访问效率	331
13.9	自动向量化实现	332
13.9.1	导出硬件设计到 SDK	332
13.9.2	创建新的应用工程	332
13.9.3	运行软件应用工程	333

13.10	NEON 汇编代码实现	334
13.10.1	导出硬件设计到 SDK	334
13.10.2	创建新的应用工程	334
13.10.3	运行软件应用工程	335

第14章	Cortex – A9 外设模块结构及功能	337
14.1	DDR 存储器控制器	337
14.1.1	DDR 存储器控制器接口及功能	338
14.1.2	AXI 存储器端口接口	340
14.1.3	DDR 核和交易调度器	341
14.1.4	DDRC 仲裁	341
14.1.5	DDR 控制器 PHY	342
14.1.6	DDR 初始化和标定	343
14.1.7	纠错码	344
14.2	静态存储器控制器	344
14.2.1	静态存储器控制器接口及功能	345
14.2.2	静态存储器控制器和存储器的信号连接	347
14.3	四 – SPI Flash 控制器	348
14.3.1	四 – SPI Flash 控制器功能	350
14.3.2	四 – SPI Flash 控制器反馈时钟	352
14.3.3	四 – SPI Flash 控制器接口	352
14.4	SD/SDIO 外设控制器	354
14.4.1	SD/SDIO 控制器功能	355
14.4.2	SD/SDIO 控制器传输协议	356
14.4.3	SD/SDIO 控制器接口信号连接	359
14.5	USB 主机、设备和 OTG 控制器	359
14.5.1	USB 控制器接口及功能	361
14.5.2	USB 主机操作模式	364
14.5.3	USB 设备操作模式	366
14.5.4	USB OTG 操作模式	368
14.6	吉比特以太网控制器	368
14.6.1	吉比特以太网控制器接口及功能	370
14.6.2	吉比特以太网控制器接口编程向导	371
14.6.3	吉比特以太网控制器接口信号连接	375
14.7	SPI 控制器	376
14.7.1	SPI 控制器的接口及功能	377
14.7.2	SPI 控制器时钟设置规则	379
14.8	CAN 控制器	379
14.8.1	CAN 控制器接口及功能	380
14.8.2	CAN 控制器操作模式	382
14.8.3	CAN 控制器消息保存	383
14.8.4	CAN 控制器接收过滤器	384

14.8.5	CAN 控制器编程模型	384
14.9	UART 控制器	386
14.9.1	UART 控制器接口及功能	387
14.10	I ² C 控制器	389
14.10.1	I ² C 速度控制逻辑	390
14.10.2	I ² C 控制器的功能和工作模式	391
14.11	ADC 转换器接口	393
14.11.1	ADC 转换器接口及功能	394
14.11.2	ADC 命令格式	394
14.11.3	供电传感器报警	395
14.12	PCI-E 接口	395
第15章	Zynq-7000 内的可编程逻辑资源	397
15.1	可编程逻辑资源概述	397
15.2	可编程逻辑资源功能	398
15.2.1	CLB、Slice 和 LUT	398
15.2.2	时钟管理	398
15.2.3	块 RAM	399
15.2.4	数字信号处理 - DSP Slice	400
15.2.5	输入/输出	401
15.2.6	低功耗串行收发器	402
15.2.7	PCI-E 模块	403
15.2.8	XADC (模拟 - 数字转换器)	403
15.2.9	配置	404
第16章	Zynq-7000 内的互联结构	405
16.1	系统互联架构	405
16.1.1	互联模块及功能	405
16.1.2	数据路径	407
16.1.3	时钟域	408
16.1.4	连接性	409
16.1.5	AXI ID	410
16.1.6	寄存器概述	410
16.2	服务质量	411
16.2.1	基本仲裁	411
16.2.2	高级 QoS	411
16.2.3	DDR 端口仲裁	412
16.3	AXI_HP 接口	412
16.3.1	AXI_HP 接口结构及特点	412
16.3.2	接口数据宽度	416
16.3.3	交易类型	417
16.3.4	命令交替和重新排序	417

16.3.5 性能优化总结	417
16.4 AXI_ACP 接口	418
16.5 AXI_GP 接口	419
16.6 AXI 信号总结	419
16.7 PL 接口选择	423
16.7.1 使用通用主设备端口的 Cortex - A9	424
16.7.2 通过通用主设备的 PS DMA 控制器 (DMAC)	424
16.7.3 通过高性能接口的 PL DMA	424
16.7.4 通过 AXI ACP 的 PL DMA	424
16.7.5 通过通用 AXI 从 (GP) 的 PL DMA	429

第17章 Zynq -7000 SoC 内定制简单 AXI - Lite IP

17.1 设计原理	430
17.2 定制 AXI - Lite IP	430
17.2.1 创建定制 IP 模板	430
17.2.2 修改定制 IP 设计模板	433
17.2.3 使用 IP 封装器封装外设	438
17.3 打开并添加 IP 到设计中	442
17.3.1 打开工程和修改设置	442
17.3.2 添加定制 IP 到设计	444
17.3.3 添加 xdc 约束文件	447
17.4 导出硬件到 SDK	448
17.5 建立和验证软件应用工程	448
17.5.1 建立应用工程	449
17.5.2 下载硬件比特流文件到 FPGA	451
17.5.3 运行应用工程	452

第18章 Zynq -7000 SoC 内定制复杂 AXI LITE IP

18.1 设计原理	453
18.1.1 VGA IP 核的设计原理	453
18.1.2 移位寄存器 IP 核的设计原理	454
18.2 定制 VGA IP 核	456
18.2.1 创建定制 VGA IP 模板	456
18.2.2 修改定制 VGA IP 模板	457
18.2.3 使用 IP 封装器封装 VGA IP	461
18.3 定制移位寄存器 IP 核	463
18.3.1 创建 SHIFTER IP 模板	463
18.3.2 修改定制 SHIFTER IP 模板	464
18.3.3 使用 IP 封装器封装 SHIFTER IP	466
18.4 打开并添加 IP 到设计中	467
18.4.1 打开工程和修改设置	467
18.4.2 添加定制 IP 到设计	469



18.4.3	添加 xdc 约束文件	472
18.5	导出硬件到 SDK	474
18.6	建立和验证软件工程	475
18.6.1	建立应用工程	475
18.6.2	下载硬件比特流文件到 FPGA	480
18.6.3	运行应用工程	480
第19章	Zynq-7000 AXI HP 数据传输原理及实现	482
19.1	设计原理	482
19.2	构建硬件系统	483
19.2.1	打开工程和修改设置	483
19.2.2	添加并连接 AXI DMA IP 核	484
19.2.3	添加并连接 FIFO IP 核	486
19.2.4	连接 DMA 中断到 PS	489
19.2.5	验证和建立设计	491
19.3	建立和验证软件工程	491
19.3.1	导出硬件到 SDK	492
19.3.2	创建软件应用工程	492
19.3.3	下载硬件比特流文件到 FPGA	503
19.3.4	运行应用工程	503
第20章	Zynq-7000 ACP 数据传输原理及实现	504
20.1	设计原理	504
20.2	打开前面的设计工程	504
20.3	配置 PS 端口	504
20.4	添加并连接 IP 到设计	505
20.4.1	添加 IP 到设计	506
20.4.2	系统连接	506
20.4.3	分配地址空间	507
20.5	使用 SDK 设计和实现应用工程	508
20.5.1	创建新的软件应用工程	509
20.5.2	导入应用程序	509
20.5.3	下载硬件比特流文件到 FPGA	512
20.5.4	运行应用工程	513
第21章	Zynq-7000 软件和硬件协同调试原理及实现	514
21.1	设计目标	514
21.2	ILA 核原理	515
21.2.1	ILA 触发器输入逻辑	515
21.2.2	多触发器端口的使用	515
21.2.3	使用触发器和存储限定条件	515
21.2.4	ILA 触发器输出逻辑	517

