

基于 PowerPC 的控制器研究与设计

买培培,苏 涛,张晓曦
(西安电子科技大学,西安 710071,China)

摘要:利用嵌入在 Xilinx FPGA 中的 PowerPC 技术,采用嵌入式可编程片上系统的设计思想,设计了基于现场可编程门阵列(FPGA)的控制器,实现在 FPGA 中定制完整的实时信号处理设备,方便了信号处理的设计。文中设计的控制器完成对快速傅里叶变换(FFT)、先进先出(FIFO)等模块的各控制信号的控制及对数据收发的控制。通过在总线上挂接自定义 IP,增强了片上微机功能,使得设计更具灵活性。通过编制特定的软件代码,利用 PowerPC 的架构可以方便地控制 FFT 模块的处理点数。最后对该控制器进行了仿真,并在 XUP VirtexII pro 实验板上对其功能进行了具体的实现。

关键词:硬核;可编程片上系统;快速傅立叶变换;信号处理

中图分类号: TP332

文献标识码: A

文章编号: CN32-1413(2009)05-0102-05

Study and Design of A Controller Based on PowerPC

MAI Pei-pei, SU Tao, ZHANG Xiao-xi
(Xidian University, Xi'an 710071, China)

Abstract: This paper uses the PowerPC technique embedded in Xilinx field-programmable gate array (FPGA) and the design idea of embedded system on programmable chip (SOPC) to design the controller based on FPGA, which implements to customize the whole real-time signal processing device in FPGA and facilitates the design of signal processing. The controller designed in this paper accomplishes the control to each control signal of fast Fourier transform (FFT) model, first in first out (FIFO) model, etc. and the control to the data receiving and transmission. The function of the microcomputer on chip is strengthened by connecting the custom IP in the bus, which makes the design more flexible. The quantity of management points of FFT model can be conveniently controlled through programming the special software codes by using the structure of PowerPC. Finally, this paper simulates the controller, and concretely realizes the function on the XUP Virtex pro test board.

Key words: hard core; system on programmable chip; fast Fourier transform; signal processing

0 引言

实时信号处理设备由控制器、信号处理部件、存储器组成。传统的实时信号处理设备要么是利用微控制单元(MCU)或数字信号处理器(DSP)等来实现,要么是结合 FPGA 在信号处理方面具有最多吞吐率的特性,利用逻辑单元在 FPGA 中搭建控制模块来实现。采用 MCU 或 DSP 设计时,器件的引入

势必增加系统设计的复杂度及电路板设计的难度。在 FPGA 中进行信号处理虽然可以获得最多的吞吐率,但是采用逻辑单元搭建控制模块通用性差,且参数不易改变;在具体的应用中采用上述 2 种方法还容易出现与其它设备接口瓶颈等问题。然而可编程片上系统(SOPC)技术的出现正好解决了实时信号处理设备设计过程中存在的问题。PowerPC 是 Xilinx 公司推出的基于 SOPC 技术的嵌入式处理

器,基于它的信号处理设备的设计既可以充分发挥 FPGA 在信号处理方面的优势,又结合了 MCU 或 DSP 在设计控制器中的优越性,将设计集成在 FPGA 芯片中,降低了设计的复杂度,同时增加了设计的灵活性。

1 SOPC 技术及 PowerPC 硬核介绍

所谓 SOPC 即可编程的片上系统,在一片系统芯片上通过编程来实现一整套嵌入式系统的搭建。SOPC 技术尽可能在单片 FPGA 上集成大规模的完整的电子系统,包括处理器系统、外设控制器系统、存储器、DSP 系统、通信系统和普通的数字电路逻辑等,从而使得电子电路系统在功能、规模、可靠性、体积、功耗、性能指标、上市周期及其硬件升级等多方面达到综合上的最优化。SOPC 设计技术是一门全新的综合性电子设计技术,涉及面很广,它将普通的电子设计自动化(EDA)技术、计算机系统、嵌入式系统、工业自动化控制系统、DSP 及无线电等融为一体,涵盖了嵌入式系统设计技术的全部内容;同时它结合了片上系统(SOC)和可编程器件(PLD)、FPGA 各自的优点,集成了 CPU、DSP、存储器、外围 I/O 及可编程逻辑,用户可以利用 SOPC 平台自行设计各种高速高性能的 DSP 处理器或特定功能的 CPU 处理器,从而使得电子系统设计进入了一个全新的模式。

Xilinx 公司结合其高端芯片 Virtex-II Pro 和 Virtex 4 推出了全新的嵌入式开发系统,利用先进的 IP 植入技术,实现以硬核 PowerPC405 或软核 Microblaze 32 位处理器为核心的 SOPC 系统。由于本系统中使用的是基于 Virtex-II pro 的 PowerPC 硬核处理器,下面分别对 Virtex-II pro FPGA 和 PowerPC 硬核做一介绍。

Virtex-II Pro FPGA 在 Virtex-II 基础上嵌入了最多可达 2 个高性能精简指令集(RISC)技术,运行频率高达 400 MHz 的 PowerPC405 处理器,其中还包括了先进的主动互连,以解决高性能系统结构所面临的挑战;同时它还嵌入了多个 3.125 Gbps 速率的 Rocket 串行收发器,为千兆以太网等高速串行接口标准提供了解决方案;另外它还利用 IP 核植入技术嵌入了 18 kB 的双口 RAM、18 × 18 乘加器和数字时钟管理单元。

PowerPC 具有 5 阶段流水线;32 × 32bit 通用寄存器;增强的指令和数据片上存储器控制器,直接

与嵌入式块 RAM 接口;支持 JTAG 调试和跟踪;新的辅助处理器单元(APU)控制器,使 CPU 管道直接与 FPGA 架构接口;同时支持用户定义的指令。另外 PowerPC405 处理器采用 IBM CoreConnect 总线技术,该总线结构由处理器局部总线(PLB)、片上外设总线(OPB)和设备控制寄存器总线(DCR)组成。PLB 总线为片内的高速数据通道,通常连接高速外设、直接存储器访问(DMA)控制器等;OPB 总线通常用于连接速率较低的片上外设,如串口、键盘等;PLB 总线和 OPB 总线通过总线桥接器相连。DCR 总线用于实现 PowerPC 的通用寄存器与逻辑设备控制寄存器的数据通信。

嵌入式系统设计涉及了软件和硬件的开发以及两者的综合设计,开发过程较为复杂。Xilinx 为了简化基于 FPGA 的嵌入式开发流程,提供了功能强大、操作简单的工具集 ISE 和 EDK。ISE 是 Xilinx 公司 FPGA 逻辑设计的基础。在这个环境下,设计者可以进行约束文件的编写、时序分析、逻辑布局布线及器件编程等。以 PowerPC 为核心的片上系统的开发则是通过嵌入式开发套件(EDK)来实现。该套件自带了许多工具和 IP,主要包括 Xilinx 平台工作室 XPS 和软件开发套件 SDK,可方便地规划、设计并生成整个片上系统的硬件和软件结构。

控制器的硬件搭建在 EDK 中实现,软件代码在 SDK 中调试,系统的最终设计则是在 ISE 中,结合 EDK 中的工程和 SDK 中的软件代码及其它逻辑模块完成的。图 1 给出了基于 PowerPC 控制器的开发流程。

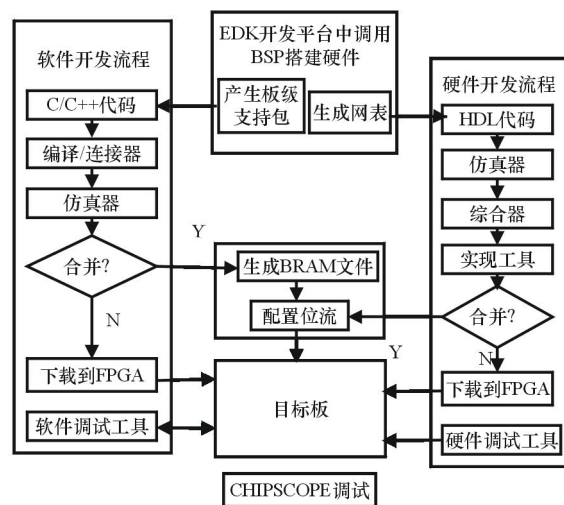


图 1 系统开发流程

2 硬件设计

本系统的主要功能是对 FFT 模块进行控制,其中包括为 FFT 模块提供各个控制信号及待处理的数据,同时将所设计的系统应用于具体的硬件电路中,利用 Chipscope 工具将真实的仿真结果加以捕捉,通过比较确认系统的正确性及可行性。文中使用的 FFT 模块来自于 ISE 中提供的 FFT 核,该核采用 Cooley-Tukey 算法计算 FFT。

系统的具体任务是首先由 PC 机向嵌入模块发出状态转移指令,此时在控制器的作用下,FFT 模块或由接收并处理数据状态转到输出数据状态,或由输出数据或空闲状态转为接收数据、处理数据状态。同时,为了可以实时控制 FFT 的处理点数,需要提供控制其处理点数的 $nfft[4:0]$ 信号及使能 $nfft$ 的 $nfft_we$ 信号。为方便验证,本设计中的数据由控制器向 FFT 模块提供,因此涉及到数据缓存的问题,文中采用 FIFO 模块作为数据缓存单元, FIFO 的读、写控制信号由控制器提供(这里的 FIFO 模块为调用的 FIFO 核)。

整个硬件实现框图如图 2 所示。FIFO1 用于缓存控制器提供给 FFT 模块的数据, FIFO2 缓存 FFT 模块输出的数据。待状态转移指令发出后,控制器使能 FIFO1 的写信号并向 FIFO1 发送数据;当数据达到所要求的数量时,将 FIFO1 的写信号置低,即使得写信号无效,同时使能 FIFO1 的读信号,并给 $nfft[4:0]$ 信号置数。使能 $start$ 信号(FFT 模块的使能信号),此时 FFT 模块接收并处理数据;待数据处理完毕,即 $done$ 信号有效后,通过控制器使能 FIFO2 模块的写信号,将 FFT 模块处理的结果缓存下来。之后使能 FIFO2 的读信号,将 FIFO2 中缓存的数据导入 PowerPC 中,利用终端、网口等工具将这些数据显示出来。各信号恢复默认值,等待下一次状态转移指令的发出。

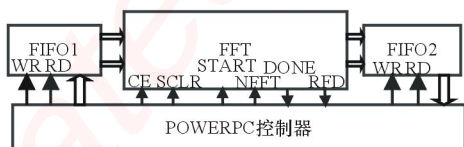


图 2 硬件实现框图

由功能要求可知,控制模块需要提供多个信号,由于 PowerPC 中采用的是 IBM 总线架构,若调用 EDK 提供的 GPIO 模块实现上述功能,则容易出现

时序混乱等问题,因此需要挂接用户自定义 IP。结合具体的应用,设计中定义了 3 个用户 IP,分别为 DATA_IN, DATA_OUT, FFT_IO。其中用户 IP DATA_IN 用于接收并存储 FFT 输出数据, DATA_OUT 向 FFT 模块提供输入数据及 FIFO 的控制信号, FFT_IO 模块则用于提供 FFT 模块的主要控制信号。

本设计中的自定义 IP 挂接在 OPB 总线上,按照 FFT_IO 所需提供的功能及端口信息,在代码段添加相应的代码。图 3 为 FFT 模块控制信号起始段时序图,从图中可看出 FFT 模块被使能后, ce 信号恒为高, $sclr$ 信号保持为低。因此对 FFT 模块控制时,最重要的 3 个信号为 $nfft$, $nfft_we$ 和 $start$ 。

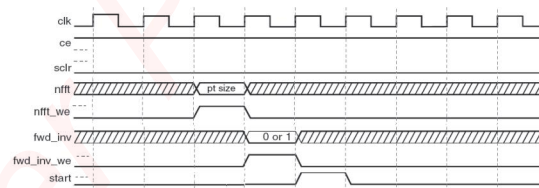


图 3 FFT 控制信号起始段时序图

图 4 为设计结果在实验板上的运行结果。从图中可以看出设计结果和 FFT 时序相一致,设计符合要求。

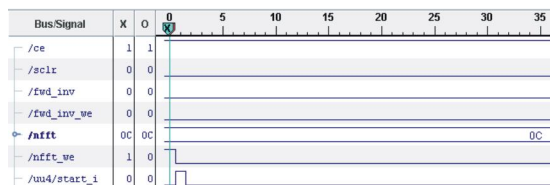


图 4 控制信号在实验板上的运行结果

DATA_OUT 模块在实验板上的运行结果如图 5 所示,由上文所介绍的 DATA_OUT 模块的功能,在信号 wr 有效时,控制器向 FIFO 发送数据。由于软件代码采用 C 语言编写,因此一条指令的执行需要占用若干个周期,这就是 wr 信号每隔 60 多个周期 1 次有效的原因。DATA_IN 模块设计和 DATA_OUT 相类似,这里就不一一列出。

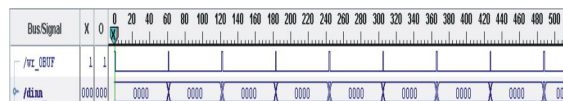


图 5 DATA_OUT 模块在实验板上的运行结果

硬件代码编写完成,并在实验板上调试通过后,将 EDK 中的工程利用“export to project navigator”工具导入到 ISE 中,作为一个独立的模块,结合已

调用的 FFT 核和 FIFO 核及其它逻辑单元完成最终的设计,如图 6 所示,图中只列出了设计中采用的主要模块。

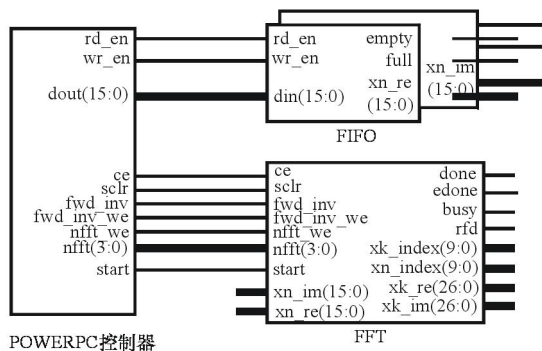


图 6 硬件连接框图

3 系统软件设计

系统软件设计,首先需要定制软件平台,选择相应的驱动程序版本、库、内核、操作系统等信息;然后调用“Generate Libraries and BSPs”工具产生相应的软件库和板级支持包,其中包括所添加外设的驱动、地址映射等信息;之后按照用户及 IP 的调用顺序,编写软件代码。

按照设计要求,控制器首先使能 wr_en 信号,向 FIFO1 发数,数据存满或存够指定量的数据后,关闭 FIFO1 写通道,停止向 FIFO1 发送数据,同时设置 FFT 处理点数 nfft,使能 FFT 模块,并打开 FIFO1 读通道。待 FFT 模块处理完毕,即 done 信号有效后,打开 FIFO2 写通道,使其接收处理所得的结果,同时 FFT 转入空闲状态;FIFO2 中的数据也可以读回到控制器模块中进行检验。图 7 中给出软件代码的设计流程。在编写软件代码时可以调用

每个设备相应的驱动程序来完成读写操作。

软件设计完成后,在 ISE 中选择“Update Bitstream with Processor Data”将软件目标程序 *.elf 和已生成的硬件目标程序 top.bit 合并(这里 top 为 ISE 中的顶层文件),产生新的内存初始化代码 top_download.bit,该文件中包括了嵌入式系统设计的全部信息。将 top_download.bit 作为配置文件下载到 FPGA,完成片上系统的搭建工作。

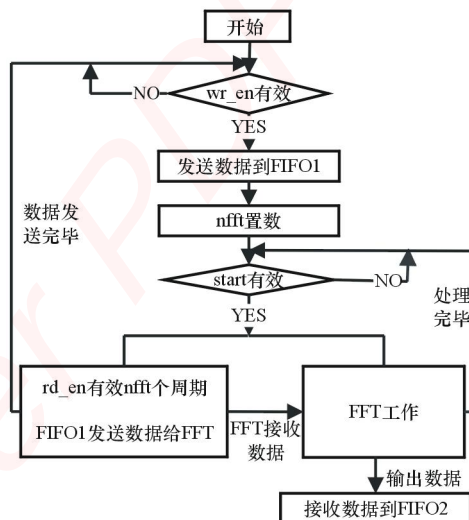
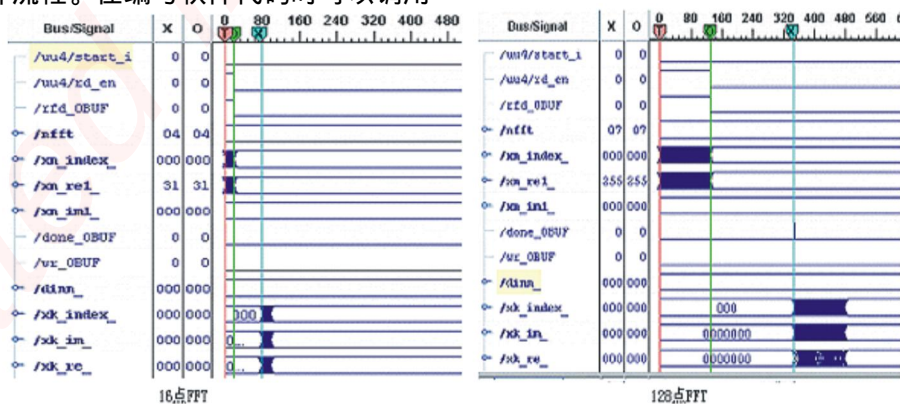


图 7 软件代码设计流程

4 控制器的实验结果

文中通过插入 Chipscope 内核对数据进行捕捉及检测。图 8 为测试结果,由于篇幅的限制,文中只列出了 16 点 FFT,而在实际应用中不同点数的 FFT 均可实现,其可实现的最大点数是由调用 FFT 核时设置的数据深度决定的。



xn_re, xn_im 输入数据的实部和虚部 start_i FFT 模块使能信号 xn_index FFT 已接收的数据个数 nfft FFT 处理点数控制信号 xk_index FFT 已输出的数据个数 xk_re, xk_im 处理结果的实部和虚部 标尺 0 与标尺 1 间为数据输入端,标尺 X 与标尺 0 间为数据处理段,标尺 X 右侧为结果输出段

图 8 测试结果

例如,本文中设置的数据深度为 2 048,则 FFT 模块处理点数的变化范围为 8~2 048。经过分析及验证,其结果完全正确,而且可以方便地控制 FFT 处理点数,完成设计要求。

表 1 统计了系统所占用的资源。文中所采用的设计平台是 XUP VirtexII pro,从表中可以看出本设计所占用资源较少,因此具有确实的可行性。

表 1 系统占用资源

逻辑使用资源	已使用	资源总量	使用量
Slice Flip Flops 个数	8 479	27 392	30 %
4 输入 LUTs	5 944	27 392	21 %
Slices	6 726	13 696	49 %
接口 IOBs	164	556	29 %
硬核 PPC405s	2	2	100 %
存储器 Block RAMs	64	136	47 %
乘法器 MUL T18X18s	22	136	16 %
GCL Ks	4	16	25 %
数字时钟管理器 DCMs	1	8	12 %

5 结束语

经验证本系统所设计的控制器工作稳定可靠、具有可移植性及简单方便的用户接口。通过该控制器的简单修改就可以方便地控制 FFT 模块的处理点数,也可以用于控制其它模块,因此极大地降低了信号处理板的设计难度,并可充分利用芯片的资源,提高系统的集成度,降低设计成本及功耗。

本文的特点在于利用嵌入在 Xilinx FPGA 中的 PowerPC 实现对信号处理模块的控制,实现在 FPGA 中对实时信号处理设备的设计,增加了设计的实用性和灵活性;采用在总线上挂接用户自定义外设技术,完成了设计要求,增强了片上微机功能。

参考文献

[1] 董代洁,郭怀理,曹春雨.基于 FPGA 的可编程 SOC 设计[M]. 北京:北京航空航天大学出版社,2006.

[2] 薛小刚,葛毅敏. Xilinx ISE 9. x FPGA/ CPLD 设计指南[M]. 北京:人民邮电出版社,2007.

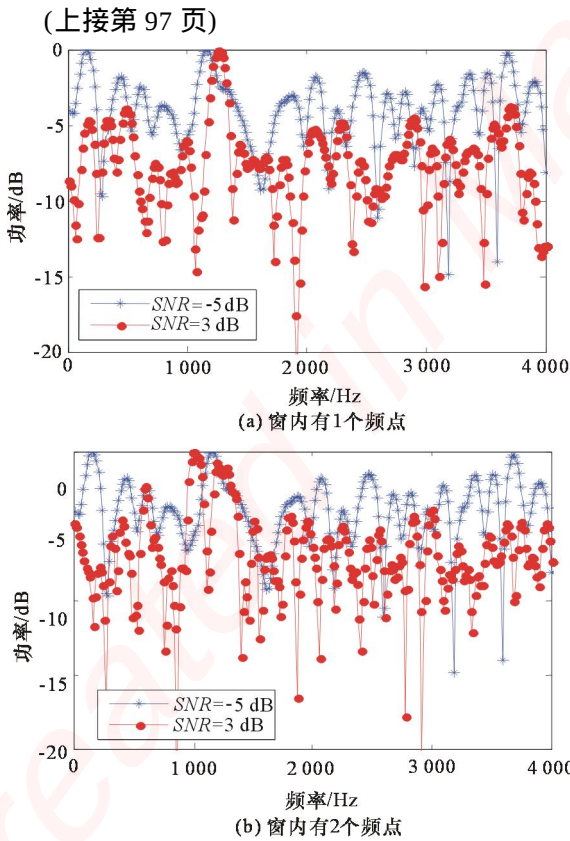
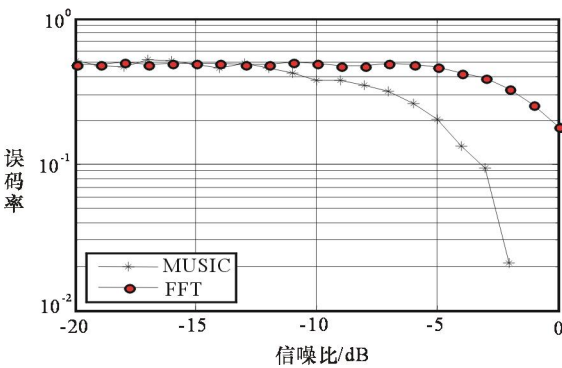


图 6 直接进行 FFT



3 结束语

通信信号在大气中传播的过程中会受到各种因素的影响,包括信道和干扰信号带来的影响。这些因素都会对信号的解调带来许多的困难,提升系统误码率。因此,对短波 ALE 系统解调方法的研究依然具有重要的现实意义。

参考文献

[1] 张曲,李晓峰. ALE Modem 8FSK 信号解调研究[J]. 四川工业学报,2001,20(1):8-13.

[2] 郭仕剑,王宝顺,贺志国,等. MATLAB 7. X——数字信号处理[M]. 北京:人民邮电出版社,2006.

嵌入式资源免费下载

总线协议:

1. [基于 PCIe 驱动程序的数据传输卡 DMA 传输](#)
2. [基于 PCIe 总线协议的设备驱动开发](#)
3. [CANopen 协议介绍](#)
4. [基于 PXI 总线 RS422 数据通信卡 WDM 驱动程序设计](#)
5. [FPGA 实现 PCIe 总线 DMA 设计](#)
6. [PCI Express 协议实现与验证](#)
7. [VPX 总线技术及其实现](#)
8. [基于 Xilinx FPGA 的 PCIE 接口实现](#)
9. [基于 PCI 总线的 GPS 授时卡设计](#)
10. [基于 CPCI 标准的 6U 信号处理平台的设计](#)
11. [USB30 电路保护](#)
12. [USB30 协议分析与框架设计](#)
13. [USB 30 中的 CRC 校验原理及实现](#)
14. [基于 CPLD 的 UART 设计](#)
15. [IPMI 在 VPX 系统中的应用与设计](#)
16. [基于 CPCI 总线的 PMC 载板设计](#)
17. [基于 VPX 总线的工件台运动控制系统研究与开发](#)
18. [PCI Express 流控机制的研究与实现](#)
19. [UART16C554 的设计](#)
20. [基于 VPX 的高性能计算机设计](#)
21. [基于 CAN 总线技术的嵌入式网关设计](#)
22. [Visual C 串行通讯控件使用方法与技巧的研究](#)
23. [IEEE1588 精密时钟同步关键技术研究](#)
24. [GPS 信号发生器射频模块的一种实现方案](#)
25. [基于 CPCI 接口的视频采集卡的设计](#)
26. [基于 VPX 的 3U 信号处理平台的设计](#)
27. [基于 PCI Express 总线 1394b 网络传输系统 WDM 驱动设计](#)
28. [AT89C52 单片机与 ARINC429 航空总线接口设计](#)
29. [基于 CPCI 总线多 DSP 系统的高速主机接口设计](#)
30. [总线协议中的 CRC 及其在 SATA 通信技术中的应用](#)
31. [基于 FPGA 的 SATA 硬盘加解密控制器设计](#)
32. [Modbus 协议在串口通讯中的研究及应用](#)
33. [高可用的磁盘阵列 Cache 的设计和实现](#)
34. [RAID 阵列中高速 Cache 管理的优化](#)

35. [一种新的基于 RAID 的 CACHE 技术研究与实现](#)
36. [基于 PCIE-104 总线的高速数据接口设计](#)
37. [基于 VPX 标准的 RapidIO 交换和 Flash 存储模块设计](#)
38. [北斗卫星系统在海洋工程中的应用](#)
39. [北斗卫星系统在远洋船舶上应用的研究](#)
40. [基于 CPCI 总线的红外实时信号处理系统](#)
41. [硬件实现 RAID 与软件实现 RAID 的比较](#)
42. [基于 PCI Express 总线系统的热插拔设计](#)
43. [基于 RAID5 的磁盘阵列 Cache 的研究与实现](#)
44. [基于 PCI 总线的 MPEG2 码流播放卡驱动程序开发](#)
45. [基于磁盘阵列引擎的 RAID5 小写性能优化](#)
46. [基于 IEEE1588 的时钟同步技术研究](#)
47. [基于 Davinci 平台的 SD 卡读写优化](#)
48. [基于 PCI 总线的图像处理及传输系统的设计](#)
49. [串口和以太网通信技术在油液在线监测系统中的应用](#)
50. [USB30 数据传输协议分析及实现](#)

VxWorks:

1. [基于 VxWorks 的多任务程序设计](#)
2. [基于 VxWorks 的数据采集存储装置设计](#)
3. [Flash 文件系统分析及其在 VxWorks 中的实现](#)
4. [VxWorks 多任务编程中的异常研究](#)
5. [VxWorks 应用技巧两例](#)
6. [一种基于 VxWorks 的飞行仿真实时管理系统](#)
7. [在 VxWorks 系统中使用 TrueType 字库](#)
8. [基于 FreeType 的 VxWorks 中文显示方案](#)
9. [基于 Tilcon 的 VxWorks 简单动画开发](#)
10. [基于 Tilcon 的某武器显控系统界面设计](#)
11. [基于 Tilcon 的综合导航信息处理装置界面设计](#)
12. [VxWorks 的内存配置和管理](#)
13. [基于 VxWorks 系统的 PCI 配置与应用](#)
14. [基于 MPC8270 的 VxWorks BSP 的移植](#)
15. [Bootrom 功能改进经验谈](#)
16. [基于 VxWorks 嵌入式系统的中文平台研究与实现](#)
17. [VxBus 的 A429 接口驱动](#)
18. [基于 VxBus 和 MPC8569E 千兆网驱动开发和实现](#)
19. [一种基于 vxBus 的 PPC 与 FPGA 高速互联的驱动设计方法](#)
20. [基于 VxBus 的设备驱动开发](#)

21. [基于 VxBus 的驱动程序架构分析](#)
22. [基于 VxBus 的高速数据采集卡驱动程序开发](#)
23. [Vxworks 下的冗余 CAN 通讯模块设计](#)
24. [WindML 工业平台下开发 S1d13506 驱动及显示功能的实现](#)
25. [WindML 中 Mesa 的应用](#)
26. [VxWorks 下图形用户界面开发中双缓冲技术应用](#)
27. [VxWorks 上的一种 GUI 系统的设计与实现](#)
28. [VxWorks 环境下 socket 的实现](#)
29. [VxWorks 的 WindML 图形界面程序的框架分析](#)
30. [VxWorks 实时操作系统及其在 PC104 下以太网编程的应用](#)
31. [实时操作系统任务调度策略的研究与设计](#)
32. [军事指挥系统中 VxWorks 下汉字显示技术](#)

Linux:

1. [Linux 程序设计第三版及源代码](#)
2. [NAND FLASH 文件系统的设计与实现](#)
3. [多通道串行通信设备的 Linux 驱动程序实现](#)
4. [Zsh 开发指南-数组](#)
5. [常用 GDB 命令中文速览](#)
6. [嵌入式 C 进阶之道](#)
7. [Linux 串口编程实例](#)
8. [基于 Yocto Project 的嵌入式应用设计](#)
9. [Android 应用的反编译](#)
10. [基于 Android 行为的加密应用系统研究](#)
11. [嵌入式 Linux 系统移植步步通](#)
12. [嵌入式 CC++ 语言精华文章集锦](#)
13. [基于 Linux 的高性能服务器端的设计与研究](#)
14. [S3C6410 移植 Android 内核](#)
15. [Android 开发指南中文版](#)
16. [图解 Linux 操作系统架构设计与实现原理（第二版）](#)
17. [如何在 Ubuntu 和 Linux Mint 下轻松升级 Linux 内核](#)
18. [Android 简单 mp3 播放器源码](#)
19. [嵌入式 Linux 系统实时性的研究](#)
20. [Android 嵌入式系统架构及内核浅析](#)
21. [基于嵌入式 Linux 操作系统内核实时性的改进方法研究](#)
22. [Linux TCP IP 协议详解](#)
23. [Linux 桌面环境下内存去重技术的研究与实现](#)
24. [掌握 Android 7.0 新增特性 Quick Settings](#)

25. [Android 应用逆向分析方法研究](#)
26. [Android 操作系统的课程教学](#)
27. [Android 智能手机操作系统的研究](#)
28. [Android 英文朗读功能的实现](#)
29. [基于 Yocto 订制嵌入式 Linux 发行版](#)
30. [基于嵌入式 Linux 的网络设备驱动设计与实现](#)
31. [如何高效学习嵌入式](#)
32. [基于 Android 平台的 GPS 定位系统的设计与实现](#)
33. [Linux ARM 下的 USB 驱动开发](#)
34. [Linux 下基于 I2C 协议的 RTC 驱动开发](#)
35. [嵌入式下 Linux 系统设备驱动程序的开发](#)
36. [基于嵌入式 Linux 的 SD 卡驱动程序的设计与实现](#)
37. [Linux 系统中进程调度策略](#)
38. [嵌入式 Linux 实时性方法](#)

Windows CE:

1. [Windows CE.NET 下 YAFFS 文件系统 NAND Flash 驱动程序设计](#)
2. [Windows CE 的 CAN 总线驱动程序设计](#)
3. [基于 Windows CE.NET 的 ADC 驱动程序实现与应用的研究](#)
4. [基于 Windows CE.NET 平台的串行通信实现](#)
5. [基于 Windows CE.NET 下的 GPRS 模块的研究与开发](#)
6. [win2k 下 NTFS 分区用 ntldr 加载进 dos 源代码](#)
7. [Windows 下的 USB 设备驱动程序开发](#)
8. [WinCE 的大容量程控数据传输解决方案设计](#)
9. [WinCE6.0 安装开发详解](#)
10. [DOS 下仿 Windows 的自带计算器程序 C 源码](#)
11. [G726 局域网语音通话程序和源代码](#)
12. [WinCE 主板加载第三方驱动程序的方法](#)
13. [WinCE 下的注册表编辑程序和源代码](#)
14. [WinCE 串口通信源代码](#)
15. [WINCE 的 SD 卡程序\[可实现读写的源码\]](#)
16. [基于 WinCE 的 BootLoader 研究](#)
17. [Windows CE 环境下无线网卡的自动安装](#)
18. [基于 Windows CE 的可视电话的研究与实现](#)
19. [基于 WinCE 的嵌入式图像采集系统设计](#)
20. [基于 ARM 与 WinCE 的掌纹鉴别系统](#)

PowerPC:

1. [Freescale MPC8536 开发板原理图](#)
2. [基于 MPC8548E 的固件设计](#)
3. [基于 MPC8548E 的嵌入式数据处理系统设计](#)
4. [基于 PowerPC 嵌入式网络通信平台的实现](#)
5. [PowerPC 在车辆显控系统中的应用](#)
6. [基于 PowerPC 的单板计算机的设计](#)
7. [用 PowerPC860 实现 FPGA 配置](#)
8. [基于 MPC8247 嵌入式电力交换系统的设计与实现](#)
9. [基于设备树的 MPC8247 嵌入式 Linux 系统开发](#)
10. [基于 MPC8313E 嵌入式系统 UBoot 的移植](#)
11. [基于 PowerPC 处理器 SMP 系统的 UBoot 移植](#)
12. [基于 PowerPC 双核处理器嵌入式系统 UBoot 移植](#)
13. [基于 PowerPC 的雷达通用处理机设计](#)
14. [PowerPC 平台引导加载程序的移植](#)
15. [基于 PowerPC 嵌入式内核的多串口通信扩展设计](#)
16. [基于 PowerPC 的多网口系统抗干扰设计](#)
17. [基于 MPC860T 与 VxWorks 的图形界面设计](#)
18. [基于 MPC8260 处理器的 PPMC 系统](#)

ARM:

1. [基于 DiskOnChip 2000 的驱动程序设计及应用](#)
2. [基于 ARM 体系的 PC-104 总线设计](#)
3. [基于 ARM 的嵌入式系统中断处理机制研究](#)
4. [设计 ARM 的中断处理](#)
5. [基于 ARM 的数据采集系统并行总线的驱动设计](#)
6. [S3C2410 下的 TFT LCD 驱动源码](#)
7. [STM32 SD 卡移植 FATFS 文件系统源码](#)
8. [STM32 ADC 多通道源码](#)
9. [ARM Linux 在 EP7312 上的移植](#)
10. [ARM 经典 300 问](#)
11. [基于 S5PV210 的频谱监测设备嵌入式系统设计与实现](#)
12. [Uboot 中 start.S 源码的指令级的详尽解析](#)

13. [基于 ARM9 的嵌入式 Zigbee 网关设计与实现](#)
14. [基于 S3C6410 处理器的嵌入式 Linux 系统移植](#)
15. [CortexA8 平台的 \$\mu\$ C-OS II 及 LwIP 协议栈的移植与实现](#)
16. [基于 ARM 的嵌入式 Linux 无线网卡设备驱动设计](#)
17. [ARM S3C2440 Linux ADC 驱动](#)
18. [ARM S3C2440 Linux 触摸屏驱动](#)
19. [Linux 和 Cortex-A8 的视频处理及数字微波传输系统设计](#)
20. [Nand Flash 启动模式下的 Uboot 移植](#)
21. [基于 ARM 处理器的 UART 设计](#)
22. [ARM CortexM3 处理器故障的分析与处理](#)
23. [ARM 微处理器启动和调试浅析](#)
24. [基于 ARM 系统下映像文件的执行与中断运行机制的实现](#)
25. [中断调用方式的 ARM 二次开发接口设计](#)
26. [ARM11 嵌入式系统 Linux 下 LCD 的驱动设计](#)

Hardware:

1. [DSP 电源的典型设计](#)
2. [高频脉冲电源设计](#)
3. [电源的综合保护设计](#)
4. [任意波形电源的设计](#)
5. [高速 PCB 信号完整性分析及应用](#)
6. [DM642 高速图像采集系统的电磁干扰设计](#)
7. [使用 COMExpress Nano 工控板实现 IP 调度设备](#)
8. [基于 COM Express 架构的数据记录仪的设计与实现](#)
9. [基于 COM Express 的信号系统逻辑运算单元设计](#)
10. [基于 COM Express 的回波预处理模块设计](#)
11. [基于 X86 平台的简单多任务内核的分析与实现](#)
12. [基于 UEFI Shell 的 PreOS Application 的开发与研究](#)
13. [基于 UEFI 固件的恶意代码防范技术研究](#)
14. [MIPS 架构计算机平台的支持固件研究](#)
15. [基于 UEFI 固件的攻击验证技术研究](#)
16. [基于 UEFI 的 Application 和 Driver 的分析与开发](#)
17. [基于 UEFI 的可信 BIOS 研究与实现](#)
18. [基于 UEFI 的国产计算机平台 BIOS 研究](#)
19. [基于 UEFI 的安全模块设计分析](#)
20. [基于 FPGA Nios II 的等精度频率计设计](#)
21. [基于 FPGA 的 SOPC 设计](#)
22. [基于 SOPC 基本信号产生器的设计与实现](#)

23. [基于龙芯平台的 PMON 研究与开发](#)

Programming:

1. [计算机软件基础数据结构 - 算法](#)
2. [高级数据结构对算法的优化](#)
3. [零基础学算法](#)
4. [Linux 环境下基于 TCP 的 Socket 编程浅析](#)
5. [Linux 环境下基于 UDP 的 socket 编程浅析](#)
6. [基于 Socket 的网络编程技术及其实现](#)