IEEE 1588 协议在工业以太网中的实现

关松青，肖昌炎，夏晓荣

(湖南大学电气与信息工程学院，长沙 410082)

摘　要：为满足电力等分布式测控系统对时钟同步的高精度要求，采用 TI 公司支持 IEEE 1588 协议的 Cortex-M3 核 ARM 芯片作为微处理器，设计并实现一种基于中断优先级的前/后台框架的精密时钟网络服务器。由于选用轻量级 TCP/IP 协议栈(LwIP)和简化同步协议设计底层通信软件，能减轻处理器负担，实时性能得到改善。测试实验结果表明，该设备同步精度高、稳定性好、成本低，具有较好的应用前景。

关键词：IEEE 1588 标准；时钟同步；网络授时；LwIP 协议

Implementation of IEEE 1588 Protocol over Industrial Ethernet

GUAN Song-qing, XIAO Chang-yan, XIA Xiao-rong

(College of Electrical and Information Engineering, Hunan University, Changsha 410082, China)

【Abstract】An IEEE 1588 server, which is based on the foreground/background system with the priorities of interrupts, is designed and implemented to meet the high precision requirement of time synchronization in distributed power systems by using an ARM®Cortex™-M3 based microcontroller, which supports IEEE 1588 protocol. The burden of microcontroller is reduced and the real-time performance is further improved as a result of adopting the lightweight TCP/IP stack and simplifying the protocol's complexity to program the bottom communication software. It is shown in test experiments that this server characterizes high clock-synchronization accuracy, excellent stability and low cost, and it has a bright prosperity to be applied into practice.

【Key words】IEEE 1588 standard; clock-synchronization; network timing; LwIP protocol

DOI: 10.3969/j.issn.1000-3428.2011.06.082

1　概述

电力、电信、测试与测量等分布式系统的发展对时钟同步精度的要求越来越高。目前主要有 2 种分布式系统同步方法：(1)每个节点都拥有自己的参考时钟(如 GPS)，它虽然具有较高同步精度，但需独特接口与连接，成本较高；(2)通过节点间时间差来同步，如 NTP(网络时间协议)，设备简单，但其精度只能达到毫秒级。本文研究的 IEEE 1588 PTP (Precision Time Protocol)也属于后者，主要用于解决测量和工业控制网络中精密时钟同步问题[1]。IEEE 1588 协议在 NTP 基础上拓展，由于在时钟同步报文收发过程中引入了硬件时间戳技术，从根本上消除了因软件因素引入的时钟同步偏差，因此将同步精度提升至亚微秒甚至是纳秒级[2]。本文以支持 IEEE 1588 协议的 Cortex-M3 内核 ARM 微处理器为核心，提出一种工业以太网高精度时钟同步实现方法。

2　IEEE 1588 同步原理简介

IEEE 1588 支持存在不同精度、分辨率和稳定度时钟的复杂系统同步[3]。系统中分别设置主时钟和从时钟，工作于主从模式，借助时钟同步报文完成时钟校准。

IEEE 1588 时钟同步由时钟偏移测量和网络延迟测量 2 个阶段组成[1]。要注意的是：测量阶段只是逻辑上的区分，在实际同步过程中是作为一个整体进行。在时钟偏移测量阶段，主时钟相继发送 2 个时钟报文给从时钟，从时钟分别记录发送和接收时刻的 2 个时间戳 Tm_n 和 Ts_n。在网络延迟测量阶段，从时钟发送请求报文和接收相应的响应报文，再次分别记录发送和接收时刻的 2 个时间戳 Ts_{n+1} 和 Tm_{n+1}。至此，从时钟根据 Tm_n、Ts_n、Ts_{n+1} 和 Tm_{n+1} 计算出时钟偏差和网络延迟并以此修正本地时钟，实现时钟同步。

3　IEEE 1588 时钟同步系统的设计与实现

硬件设计采用 TI 公司生产的 LM3S-8962 芯片作为处理器，以 GPS/北斗时钟为参考时钟源，依托工业以太网[4]，通过 UDP/IP 通信完成系统时钟同步。

3.1　系统硬件结构

LM3S-8962 是一款专门针对工业控制领域的 MCU，中断延迟为纳秒级，特别适合实时测量和控制，此外它集成以太网的 MAC 和 PHY 层，支持 IEEE 1588-2002[5]。系统时钟选用 GPS 或北斗卫星时间源，外围电路则由于该 MCU 集成度很高而相对简化。系统的硬件架构如图 1 所示。

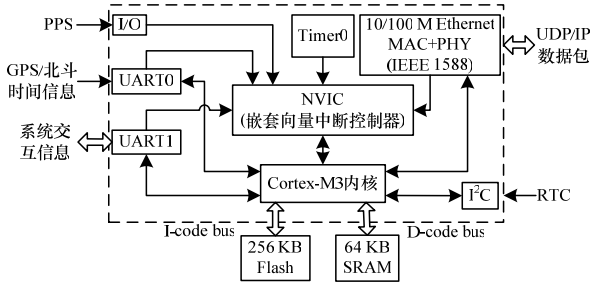

图 1　系统硬件架构

图 1 中串口 UART0 接收 GPS/北斗模块的时间信息；PPS 向系统提供整秒同步脉冲；串口 UART1 输出当前系统信息和接收用户配置参数，主要用于系统监控和调试；以太网接

口用于 UDP/IP 报文的收发；RTC 为系统提供掉电不丢失的时间信息。

3.2 系统功能设计

在设计 IEEE 1588 同步系统的过程中，引入了一个 32 位全局变量，它记录着自 1970 年 1 月 1 日 0 时 0 分 0 秒以来积累的所有秒的个数，为整个系统提供时间依据，是系统中各个功能模块实现的基础。系统功能模块分配与设计如图 2 所示。

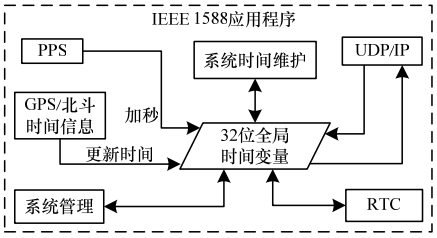

图 2 系统功能模块分配与设计原理

从图 2 可以看出，系统共分为 6 大功能模块：

(1)PPS 加秒模块。提取 GPS/北斗模块提供的高精度整秒脉冲，并对系统时间进行加秒和延时补偿操作，它在所有任务中优先级最高，响应时间最短且可预测。

(2)接收 GPS/北斗卫星时间信息模块。接收来自卫星的串口时间并更新系统时间，使之与卫星时间保持一般的同步。它与 PPS 加秒模块配合运行。在系统运行中，它们被设定为只在系统初始运行阶段和运行中的周期时间段触发作用，以避免每个整秒时刻都进行同步而造成系统时间的不连续。

(3)系统时间维护模块。更新系统时间。它由系统内部定时器 0 提供计时依据，定时器 0 每秒产生 1 次中断。该模块接受卫星时钟源(UTC)的校准，当系统断电重启后且失去 UTC 时钟源时，也可接受 RTC 的校准。在正常状态下，UTC 时钟定期校准 RTC。

(4)RTC 模块。在系统断电后且卫星时钟源失效的情况下，读取由后备电池供电维持的时间并更新系统时间，这时的时间精度取决于 RTC 模块晶振的质量。

(5)系统管理模块。实现人机交互操作，包括设置系统运行和调试参数、控制其他模块的运行并显示系统信息。

(6)IEEE 1588 协议处理模块。是系统设计的主体，它主要完成主从时钟的同步。IEEE 1588 协议的运行依赖于 UDP/IP。考虑到系统要求的高度实时性，本设计采用 LwIP 协议栈并通过回调函数的方法实现 IEEE 1588 协议的通信[6]。LwIP 是一个轻量级协议栈，专为资源有限的嵌入式系统而设计，它占用代码非常少，运行速度快。此外，要注意：LM3S-8962 内部定时器 3 有 2 个 16 位的定时器，它们分别通过硬件捕获事件并行记录在 MAC 层与 PHY 层之间 UDP 报文的接收和发送时间戳，时间戳的硬件特性是实现 IEEE 1588 的关键，在系统开发过程中务必使能微控制器相应的时间戳功能。

为了保证任意时刻系统状态和响应时间的可预测性，提高时钟精度和稳定性，在系统功能的程序设计中，与时间紧密相关的模块用中断服务程序(前台)实现并赋予相应中断优先级，而与时间关联度较差的模块则采用后台方式运行。对于前台程序，根据对系统时间影响的程度给每个使用中断方式的模块进行中断优先级分配，具体从高到低的优先级设置如下：PPS 加秒模块→系统时间维护模块→IEEE 1588 协议处理模块。此外，在软件设计中考虑到系统实时响应的性能，采用无实时操作系统的应用程序代码直接在 CPU 上运行的形式，使系统的时钟同步直接与底层硬件相关，以进一步提高系统响应速度，进而提高同步精度。根据各模块的功能设置和同步系统的设计要求，系统的程序流程如图 3 所示。

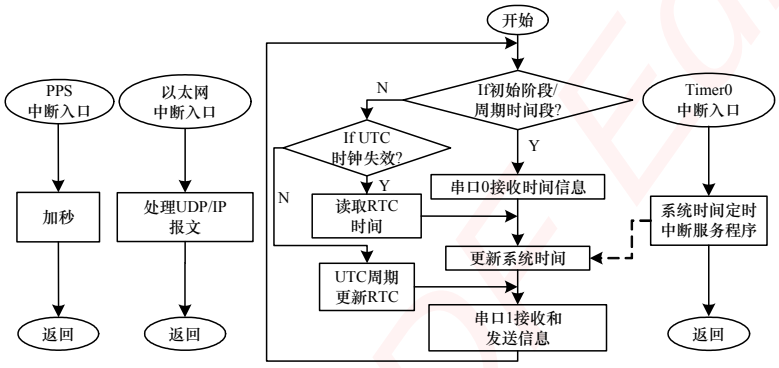

图 3 系统程序流程

开始时，在卫星时钟源正常的情况下，系统会周期性地用 UTC 校正 RTC 的时间，并通过串口 1 与用户进行信息交互，然后重复该过程；而在系统启动的初始阶段或用户设定的周期时间段，系统使能 PPS 中断并通过串 0 接收时间信息以更新系统时间，接着与用户交互操作，然后返回继续运行。在一切正常就绪以后，系统会响应外部的中断(包括 PPS 中断、Timer0 中断和以太网中断)并进入中断处理程序。

4 时钟同步性能测试与分析

为了验证所设计系统的时钟同步效果，本文对系统进行了实验室模拟测试。测试条件为：MCU 运行频率为 50 MHz，片内 64 KB SRAM，最小时钟周期为 20 ns，以太网带宽为 100 Mb/s。测试过程如下：在 UTC(卫星时钟)正常状态下，首先是主时钟与卫星时钟同步，成功同步后，主时钟就是所期望的 UTC 时间，接着主时钟发送同步报文给从时钟，成功同步从时钟后，整个网络处于正常运行状态，这时主从时钟同时在整秒的时刻向测试比对设备发送一个适度宽度的脉冲，测试比对设备通过外部的 2 只捕获引脚并行地记录脉冲到来的时间，并计算两者之差 θn，然后通过串口将 θn 发送到上位机(PC 机)进行相关处理与输出显示，从而可验证出时钟同步的效果。测试时间超过一周，取其中近 10 h 的测试数据。所得的测试结果如图 4 所示。

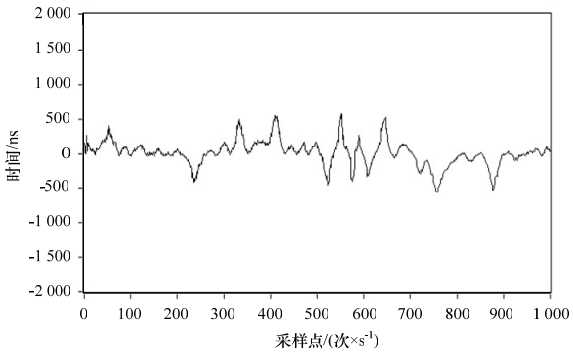

图 4 IEEE 1588 同步测试曲线

由图 4 可看出系统的时钟同步精度在 0.6 μs 以内，且运行状态稳定，充分体现了本设计的合理性和正确性。另外，经分析发现，除了协议本身在软硬件方面影响时钟同步精度之外，系统晶振的精度和稳定性、线路延迟的不稳定性及系统运行的时钟频率等也是影响因素。 (下转第 241 页)

的平均利用率。

任务到达频率对任务的调度影响很大，本文根据文献[7]，任务延迟因子=D/E，其中，D 为任务到达的平均间隔时间；E 为任务的平均执行时间，每组生成 1 000 个任务。任务执行时间分布在[1,20]内，则 E≈10，松弛时间分布在[1,10]内。设定延迟因子为 0.042，则 D=0.42，即每组任务到达时间皆分布在[0,420]内。

4.2 实验结果及分析

图 2~图 4 分别为调度成功率、芯片利用率及性能对比。

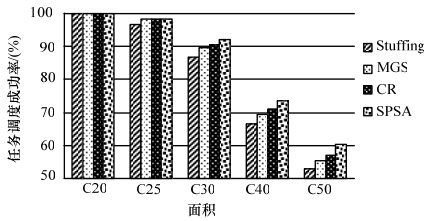

图 2　任务调度成功率对比

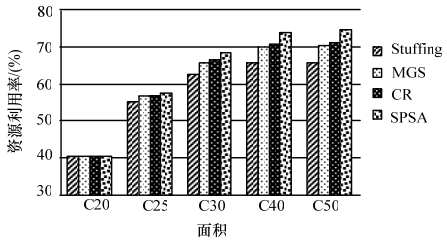

图 3　资源利用率对比

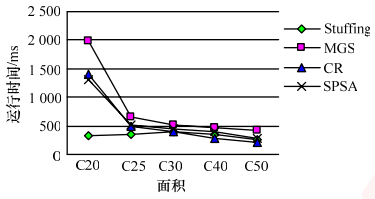

图 4　各算法运行时间对比

从实验结果来看，SPSA 算法在任务调度率和芯片利用率上较 CR、MGS 算法有明显的提高。分析其原因，Stuffing 和 MGS 采用的资源管理算法不具备完全可识别性。Stuffing 算法采用 FirstFit 放置策略，调度质量最差。而 MGS 采用基于任务顶点的放置策略，忽略了一些靠边情况，调度质量不如 CR 算法；同时，由于每次操作都需更新相关矩阵[5]，其效率也不如 CR 和 SPSA。CR 算法是现有较好的一个算法，采用基于任务放置区域周边被占用 RCU 数量的放置策略，但忽略了芯片边界的靠边情况，效果不如 SPSA 算法。

SPSA 直接采用了文献[6]中的资源管理算法，采用最大邻接边值的放置策略，从而最大程度地降低了当前调度对后续任务的影响；并加入亚可抢占机制，一旦任务不能成功调度，算法将预留任务和新任务进行离线调度，进一步提高调度质量。在执行时间上，SPSA 算法虽然经常要对预留任务进行重新调度，但由于采用了效率更好的资源管理算法和任务放置策略，总体调度时间较 CR 算法并没有明显的增加，并优于 MGS 算法。

5　结束语

本文提出一种抢占式任务调度算法。在前人研究成果的基础上，完善了基于邻接边的放置策略，使得硬件任务布置更加紧凑。实验表明了算法的有效性。后续工作包括继续改进离线任务调度算法，在 FPGA 器件上构建可操作的硬件任务调度框架，建立实测平台，进一步验证算法的有效性。

参考文献

[1] 季爱明, 谢满德. 二维阵列型可重构计算的层次参数模型[J]. 计算机工程, 2008, 34(18): 274-277.

[2] Steiger C, Walder H, Platzner M. Operating Systems for Reconfigurable Embedded Platforms: Online Scheduling of Real-time Tasks[J]. IEEE Transactions on Computers, 2004, 53(11): 1393-1407.

[3] Septién J, Mozos D, Mecha H, et al. Perimeter Quadrature-based Metric for Estimating FPGA Fragmentation in 2D HW Multitasking[C]//Proc. of IPDPS'08. Miami, USA: IEEE Press, 2008: 1-8.

[4] 周学功, 梁　楹, 黄勋章, 等. 可重构系统中的实时任务在线调度与放置算法[J]. 计算机学报, 2007, 30(11): 1901-1909.

[5] 齐　骥, 李　曦, 于海晨, 等. 一种面向动态可重构计算的调度算法[J]. 计算机研究与发展, 2007, 44(8): 1439-1447.

[6] Lu Yi, Marconi T, Gaydadjiev G, et al. An Efficient Algorithm for Free Resources Management on the FPGA[C]//Proc. of DATE'08. New York, USA: ACM Press, 2008.

[7] Steiger C, Walder H, Platzner M, et al. Online Scheduling and Placement of Real-time Tasks to Partially Reconfigurable Devices[C]//Proc. of RTSS'03. Cancun, Mexico: [s. n.], 2003.

(上接第 238 页)

5　结束语

本文介绍了一种 IEEE 1588 时钟同步系统，给出了具体的软、硬件设计框图，通过实验室模拟测试验证了系统的同步性能，并分析了影响时钟同步精度的主要因素。实验结果表明：设计的同步精度达到了亚微秒，可满足相关领域高精度时钟同步的要求。本开发针对电力系统，也适合其他分布式测控领域，目前已通过初步样机测试，为产品化和应用推广奠定了基础。

参考文献

[1] Institute of Electrical and Electronics Engineers. IEEE Std 1588-2002 IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems[S]. 2002.

[2] John C, Michael C, Joe W. IEEE 1588-2002 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems[C]//Proc. of ISA'02. [S. l.]: IEEE Press, 2002: 99-105.

[3] Eidson J. Measurement, Control, and Communication Using IEEE 1588[M]. New York, USA: Springer-Verlag, 2006.

[4] Holler R, Sauter T, Kero N. Embedded SynUTC and IEEE 1588 Clock Synchronization for Industrial Ethernet[C]//Proc. of IEEE Conf. on Emerging Technol. Factory Autom.. Lissabon, Portugal: [s. n.], 2003: 422-426.

[5] ARM Limited. Cortex-M3 Technical Reference Manual[EB/OL]. (2006-01-05). http://www.arm.com.

[6] Dunkels A. The lwIP TCP/IP Stack[EB/OL]. (2009-10-15). http://savannah.nongnu.org/projects/lwip/.

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. 基于磁盘异或引擎的 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μC-OSⅡ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 NiosⅡ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