USB3.0 物理层中弹性缓冲的设计与实现

朱小明，王小力，程　曾
（西安交通大学 电信学院，陕西 西安 710049）

摘　要：弹性缓冲用于在不同的时钟域中同步数据以保持数据的完整性，在USB、PCIE等高速串行总线的物理层中普遍应用.通过分析弹性缓冲的作用机制，根据USB3.0的协议要求，采用具有写指针屏蔽、指针跳跃、断点保存与握手、输出控制等具有创新功能的异步FIFO来设计弹性缓冲，很好实现了时钟频率补偿的目的.所设计的弹性缓冲采用并行10位数据，读写时钟可达到500 MHz的频率.该研究结论可用于满足USB3.0协议的弹性缓冲等高速弹性缓冲的场合，具有一定的工程应用价值.

关键词：弹性缓冲；常半满；USB3.0；异步FIFO
中图分类号：TP393　　**文献标识码**：A　　**文章编号**：1000－7180(2012)06－0117－05

Design and Implementation of Elastic Buffer for USB3.0 PHY

ZHU Xiao-ming，WANG Xiao-li，CHENG Zeng
(School of Electronic and Information Engineering，Xi′an Jiaotong University，Xi′an 710049，China)

Abstract：Elastic Buffers which are applied in USB3.0 and PCIE physical layer are used to ensure data integrity when bridging two different clock domains. By the research of elastic buffer mechanism and USB3.0 specification，an elastic buffer is designed by using asynchronous FIFO which has innovative function including write pointer mask，breakpoint restored，pointer jump，handshake methods and output control，and the frequency delta is managed perfectly by this elastic buffer. The width of input and output data is 10 bits，and the write clock and read clock can achieve 500 MHz. The research result can be applied to elastic buffer of USB3.0 and other system where high speed elastic buffer is needed，and is valuable to engineering application to some extent.

Key words：elastic buffer；normal half full；USB3.0；asynchronous FIFO

1　引言

通用串行总线3.0(Universal Serial Bus 3.0，USB3.0)是由Intel、Microsoft、NEC等公司于2008年11月公布的标准[1].USB是近些年来应用在PC领域的新型接口技术[2].相对于USB2.0，USB3.0依然属于高速，串行，源同步的传输协议.在发送端采用差分对的方式发送串行数据，在接收端利用时钟和数据恢复电路(Clock and Data Recovery，CDR)从串行数据中恢复出串行数据和时钟.由于USB3.0的收发两端支持独立的参考时钟源[3]，所以接收端恢复出的时钟频率和其本地时钟频率存在差别.为了补偿两个时钟的频率差别，需要一个弹性缓冲(Elastic buffer)，将CDR恢复的时钟域数据有效地同步到本地时钟域中.

弹性缓冲由Maurice Karnaugh在电话网络中传输脉冲编码调制(Pulse Code Modulation，PCM)信号中提出来的[4].弹性缓冲通过插入和删除某一特殊的符号来实现时钟补偿的目的，并能够稳定地在两个时钟域间同步数据，因此，弹性缓冲在很多技术应用中被采用，例如USB、PCIE、以太网等协议中都采用了弹性缓冲技术来同步数据.本文依据USB3.0协议，设计并实现了适用于USB3.0物理层的弹性缓冲.

收稿日期：2011－09－28；**修回日期**：2011－11－09

2 弹性缓冲的作用域及其容量

2.1 弹性缓冲的作用域

USB3.0的数据传输采用全双工(Full duplex),物理层有两个差分信号对,一对为发送差分信号对,另一对为接收差分信号对.由于弹性缓冲只用于接收端,如图1给出物理层的接收端的框图,它包括差分接收、时钟数据恢复电路、串并转换、8B10B解码和解扰电路等.

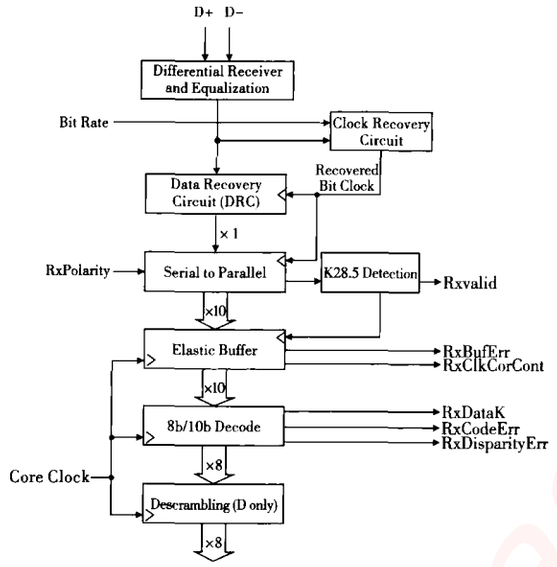


图1 USB3.0物理层接收部分结构

如图1所示,差分信号由差分接收端接收,经过CDR恢复出时钟和数据,然后数据经过串并转换变为10比特的数据,通过序列检测器检测出包的起始标志符K28.5,当检测到K28.5时,使符号锁定(symbol lock)信号有效,当symbol lock有效时,弹性缓冲开始写入数据.弹性缓冲的写时钟为CDR恢复的时钟,称为接收时钟,而其读时钟为接收端的系统时钟.通过弹性缓冲将接收时钟域的数据有效的同步到系统时钟域,并且经过8B10B译码和解扰器将数据传递给系统.

2.2 弹性缓冲容量

为了补偿接收时钟和系统时钟的频率差别,在发送端平均每发送354个字节插入一个SKP序列对(SKP ordered set),一个SKP ordered set是由2个SKP(K28.1)构成,当接收端弹性缓冲检测到SKP ordered set时根据此时弹性缓冲中数据的个数来删除或插入SKP ordered set.由于USB3.0的发送和接收端采用独立的参考时钟,所以,在接收端通过CDR恢复出来的接收时钟和本地的系统时钟会有一个时钟偏差.为了匹配两端的频率差别,需要一个弹性缓冲,通过插入和删除SKP来达到速率匹配.

本文设计的参考时钟精度-300×10^{-6}到$+300\times10^{-6}$,扩频时钟(Spread Spectrum Clocking)的时钟变化为$-5\ 000\times10^{-6}$到0,因此,发送端和接收端的时钟偏差为$-5\ 300\times10^{-6}$到300×10^{-6},最大的偏差为$5\ 600\times10^{-6}$.不管参考时钟频率如何,两者之间的偏差为$5\ 600*0.000\ 001=0.0056$T.在USB3.0中最大数据包长度为1056字节,并且协议要求在一个数据包中不能插入SKP.最差的情况下两个时钟发送数据的最大数据偏差为:$(T*1056*10-T(1-0.0056)*1056*10)/T=59$比特$=8$字节,这里乘10是考虑了8b/10b编码,因此弹性缓冲容量最小为8字节.

弹性缓冲深度为其可以存放的数据个数,通过计算可知,弹性缓冲的深度应该为8.本文设计采用常半满[5](Normal half full)模式来设计弹性缓冲,其容量为16,在通常情况下弹性缓冲中有8个数据,剩下的8个为缓冲空间,故称常半满.常半满模式首先要向弹性缓冲中写入8个符号(symbol,在USB3.0中一个8比特位宽或者一个10比特位宽的数据为一个symbol),达到半满,然后读使能有效,常半满模式只有在symbol队列中出现SKP对时,才能添加或者删除SKP对.

3 弹性缓冲的原理和结构

3.1 弹性缓冲的原理

弹性缓冲实质上是一个异步FIFO,其写时钟为接收时钟,读时钟为系统时钟,数据位宽为10比特,FIFO深度为16,当symbol lock有效时,在接收时钟域,向FIFO中写入8个来自串并转换器的数据.当读写时钟一样快慢时,FIFO中有8个有效数据,维持在半满状态.

当读时钟快于写时钟,读出的数据多于写入的数据,经过一段时间后导致FIFO中的数据数量少于8,甚至有可能被读空.如图2所示,当输入数据出现SKP时,FIFO中有效数据为4,比常态少了4.此时,弹性缓冲应该添加4个SKP,使得FIFO维持半满状态,以此来调节时钟.此时写指针向前跳跃4个间隔,并保存跳跃点的位置,当读指针读到跳跃区间时,通过输出控制,完成SKP添加,也就是说,并没有将新插入的SKP确实写入FIFO中,跳跃区间

内的地址数据为无效的数据，并且添加的 SKP 必须满足运行差异参数(Running Disparity)[6]的要求，否则会导致 8B10B 译码时出错.

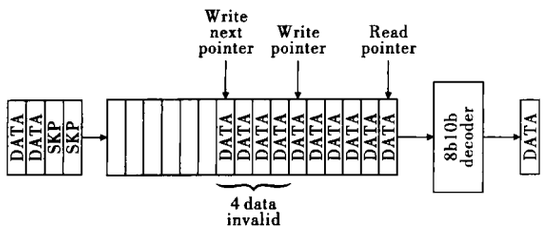


图 2　常半满模式中读快于写

当读时钟慢于写时钟，写入的数据多于读出的数据，经过一段时间后将导致 FIFO 中的数据数量多于 8，甚至有可能被写满. 如图 3 所示，在出现 SKP 窗口时，FIFO 中有效数据为 10，比常态多 2. 此时，弹性缓冲应该删除 2 个 SKP，使得 FIFO 维持半满，以此来调节时钟. 此时写指针应该暂停 2 个时钟周期，完成 SKP 删除.

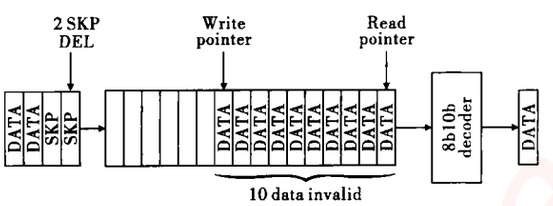


图 3　常半满模式中写快于读

在弹性缓冲读写控制的过程中，写先于读. 当 FIFO 中数据达到 8 个后，读写同时有效. 当写结束，即一个包接收完毕，但是读不一定结束，直至读到空，即所有数据已经同步到本地系统时钟域中，表示此次读写次任务结束. 这种流程控制就保持了系统数据的完整性.

3.2　弹性缓冲的结构设计

本文设计的常半满模式的弹性缓冲，其结构如图 4 所示，其结构可以分为输入检测模块，二进制和格雷码读写指针产生模块，格雷码读写指针在两个时钟域间同步模块，读写控制模块，判断弹性缓冲中有效数据的阈值监测模块以及输出控制模块. 本文采用同步比较来判断弹性缓冲中的有效数据个数，避免了异步比较的毛刺和不稳定，采用了一种新的满空标志产生方法，采用写指针屏蔽，写指针跳跃和读写指针握手的方式实现 SKP 的删除和插入.

3.2.1　输入检测模块设计

输入检测模块的功能是检测输入数据是否为

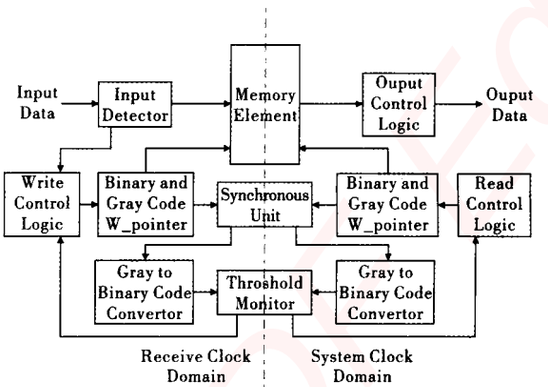


图 4　弹性缓冲的结构

SKP，并根据此时弹性缓冲中有效数据个数，选择是否删除输入的 SKP 或插入新的 SKP. 为了使新插入的 SKP 满足 8B10B 编码要求，还应保存输入 SKP 的运行差异参数.

3.2.2　读写指针的产生和同步模块设计

在判断弹性缓冲中有效数据个数和产生满空标志时，由于异步 FIFO 的读写是由异步时钟控制的[7]，不能将两个时钟域的指针直接比较，这样会产生较大的亚稳态. 本文设计中利用二进制码和格雷码指针产生模块，产生二进制码和格雷码，其中用二进制码作为存储单元的地址输入. 将格雷码的读指针，用写时钟采样两次，即经过两个写时钟控制的寄存器同步到写时钟域中，由于格雷码每次只变化一位，故经过同步后，可将亚稳态的可能性降到最低，然后将写指针和同步到写时钟域的读指针做比较产生 FIFO 满的标志. 同样，将写指针同步到读时钟域中产生 FIFO 空的标志. 本文采用新的满空标志产生方法，只将读写指针地址的次高位到最低位作为存储单元的地址，当读写地址相同时弹性缓冲为空，当读写指针最高位不同而其余所有位相同时弹性缓冲为满[8].

3.2.3　阈值监测模块设计

阈值监测模块的功能是用来判断弹性缓冲的有效数据个数，直接用格雷码无法判断缓冲中的数据，因此，需将同步到写时钟域中的格雷码读指针转化为二进制码与二进制码的写指针比较来判断弹性缓冲中的有效数据. 由于二进制码的读指针由同步到写时钟域中的格雷码的读指针转换而来，此时不会产生亚稳态的问题. 当弹性缓冲中的数据少于等于 6 时，添加 SKP，add_skp 信号有效，当弹性缓冲中的数据大于等于 10 时删除 SKP，dele_skp 信号有效.

3.2.4 读写指针控制模块和输出控制模块设计

当 dele_skp 信号有效时，并且输入检测模块检测到输入数据为 SKP 时，写指针控制模块使写使能无效，暂停写指针，或称指针屏蔽(pointer mask)，则完成 SKP 的删除.

当 add_skp 信号有效时，并且输入检测模块检测到输入的数据为 SKP 时，根据弹性缓冲的数据个数，插入新的 SKP. SKP 添加通过断点保存，写指针跳跃与握手和输出控制模块来实现.

在 SKP 窗口出现和添加阀值标志有效时，写指针此时计算 FIFO 中的有效数据个数. 根据 FIFO 中有效数据的个数与 8 的差距，来决定 write_addr_b 所指向的下一个指针点，在写指针模块的控制下完成写指针的跳跃，即为写指针跳跃(write pointer jump). 并且在写时钟域把当前的写指针和下一个指针点保存到 current_write_addr_b 和 next_write_addr_b 中. 同时在写时钟域给出 inform_rp 信号，由于读指针总是落后于写指针，当读指针读到断点的起始地址时，inser_skp_en 信号有效，并反馈 ack 信号，当写时钟域检测到 ack 信号有效时撤销 inform_rp 信号，输出控制模块检测到 inser_skp_en 信号有效后，根据此时输入的 SKP 的 RD 的值插入新的 SKP，输出控制模块保证插入所有的 SKP 均满足 8B10B 编码的要求；当读指针读到断点的结束地址时，inser_skp_en 信号变为无效，同时撤销 ack 信号，完成 SKP 的插入，使 FIFO 维持半满状态.

读指针控制模块的功能是控制读使能信号，当 symbol lock 信号有效时，此时写时钟开始向 FIFO 中写入数据，读使能需要等到 FIFO 写入 8 个数据后才能有效，当 symbol lock 无效后，即一个数据包已经接收完毕，写使能无效，但此时读使能不一定立刻无效. 直到 FIFO 被读空为止，所有的数据都已经同步到本地，读使能才无效.

4 弹性缓冲设计的仿真

本文采用 Verilog HDL 设计弹性缓冲，用 NC-Verilog 做功能仿真验证.

本设计采用台积电 TSMC 65 nm 的工艺库用 Design Compiler 进行综合，并且给予较为保守的时序约束，读写的时钟频率均可达到 500 MHz. 用 Encounter 进行后端的布局和布线，得到精确的时序信息，然后采用 Prime Time 进行静态时序分析，最终采用 NC-Verilog 做时序仿真，其仿真结果如图 5 和图 6 所示.

仿真结果表明，本文设计的弹性缓冲可以正确地写入和读出数据，能够正确删除和插入 SKP，插入的 SKP 满足 8B10B 编码的要求. 能够正确地产生弹性缓冲满空标志，并可在两个时钟域间同步数据. 设计完全符合 USB3.0 协议的要求，验证了本文设计的可靠性.

5 结束语

本文系统分析了弹性缓冲的作用原理和结构模块. 根据 USB3.0 协议要求，采用异步时钟 FIFO 来设计弹性缓冲，弹性缓冲采用并行 10 位数据，读写可以达到500 MHz的频率. 经过仿真验证，设计完

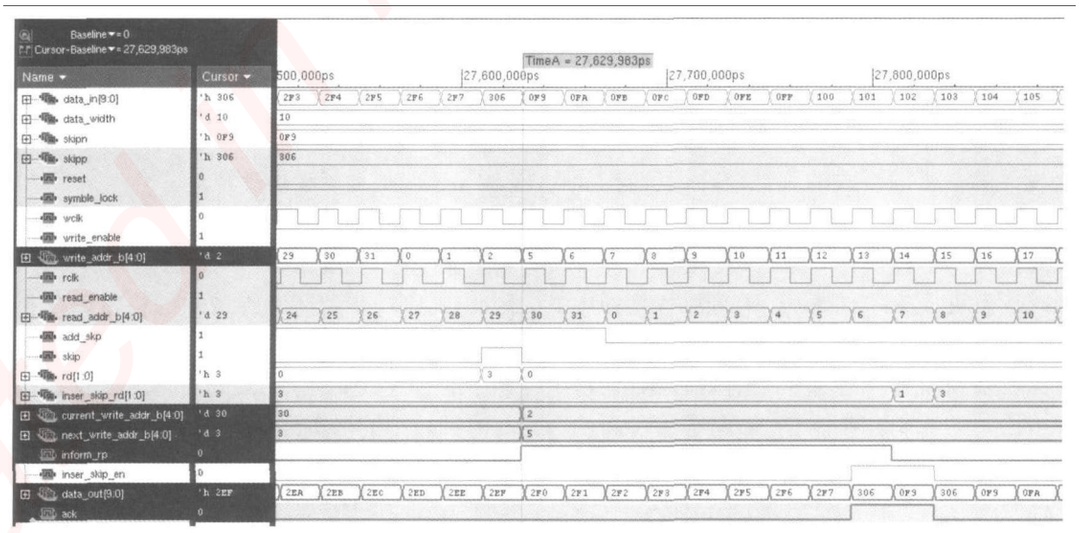


图 5 SKP 对的添加

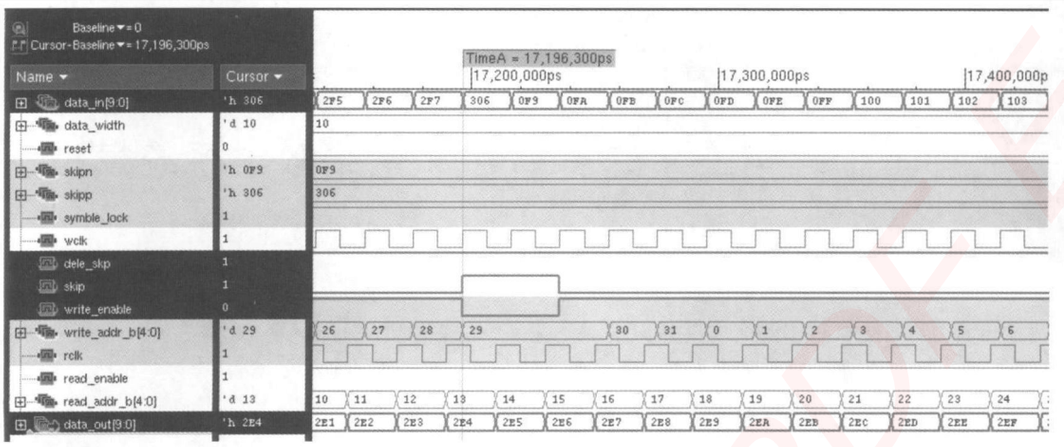


图 6　SKP 对的删除

全满足 USB3.0 的协议要求，实现了时钟频率补偿的技术要求. 将本文设计的结构稍加修改也可作为 PCIE 物理层的时钟频率补偿，并可满足其它需要高速弹性缓冲的技术应用，本文研究结论具有一定工程应用价值.

参考文献：

[1] Wikipedia, USB[EB/OL]. [2011-09-28]. http://en.wikipedia.org.

[2] 彭琰，曾云. 基于 HID 类 USB 外设功能控制器的 ASIC 设计[J]. 微电子学与计算机，2009(4)：15-18.

[3] Universal Serial Bus 3.0 Specification[S]. 美国：USB 组织，2008.

[4] Joe Winkles. Elastic Buffer Implementations in PCI Express Devices [M]. Mindshare Inc, 2003.

[5] PHY Interface for PCI Express and USB 3.0 Version3.0[S]. 美国：英特尔，2009.

[6] 张民，许进，黄学田. 光以太网[M]. 北京：北京邮电大学出版社，2003.

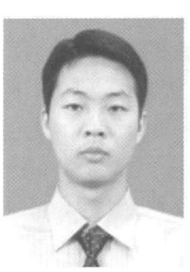
[7] 刘洪波，龙娟，郝晓莉，等. 异步 FIFO 状态判断的研究与设计[J]. 微电子学与计算机，2007(24)：81-84.

[8] Cummings C E. Simulation and synthesis techniques for asynchronous FIFO design [R]. SNUG, 2002.

作者简介：

朱小明　男，(1985－)，硕士研究生. 研究方向为计算机高速串行接口技术、视频编解码与通信.

王小力　男，(1956－)，教授，博士生导师. 研究方向为高性能 VLSI 与优化设计、集成化芯片系统设计与基础研究、光纤通讯与光波分复用技术.



程　曾　男，(1987－)，硕士研究生. 研究方向为 SOC 低功耗设计与验证.

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
51. IEEE 1588 协议在工业以太网中的实现
52. 基于 USB30 的设备自定义请求实现方法

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
33. 基于 VxWorks 实时控制系统中文交互界面开发平台

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
39. 基于实时 Linux 计算机联锁系统实时性分析与改进

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
21. DCOM 协议在网络冗余环境下的应用

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
19. 基于 PowerPC 的控制器研究与设计

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
27. Uboot 在 S3C2440 上的移植
28. 基于 ARM11 的嵌入式无线视频终端的设计

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
24. 基于 X86 平台的嵌入式 BIOS 可配置设计

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