大容量 NAND Flash文件系统中的地址映射算法研究

时　正,陈香兰,纪金松,龚育昌

(中国科学技术大学 计算机科学与技术系, 安徽 合肥 230027)

E-mail: shizheng@mail.ustc.edu.cn

摘　要: 随着 Flash芯片容量的日益增长,如何设计低空间复杂度的 Flash管理算法已经成为 RAM 空间受限的嵌入式存储系统的一个关键问题. 本文根据文件在 Flash介质上连续存放的特性,引入区段的概念,提出基于极大映射区段的地址映射算法,并使用简单的数据结构和高效的算法来维护映射中的区段,大大减小了 Flash文件系统的 RAM 用量. 最后,通过不同应用负载的实验验证了该算法的有效性. 在部分应用负载下,可降低高达 95%的 RAM 使用量.

关 键 词: NAND Flash; 文件系统; 地址映射算法; 嵌入式系统

中图分类号: TP315　　**文献标识码**: A　　**文 章 编 号**: 1000-1220(2010)01-0155-05

Space Efficient Mapping Management for Large-scale NAND Flash File System

SHI Zheng, CHEN Xiang-lan, JI Jin-song, GONG Yu-chang

(Department of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China)

Abstract: Flash memory is prevalent in embedded storage systems nowadays, but the RAM requirements for address mapping increases as the flash chip's volume increases. To design a space efficient management is a critical issue for the resource-limited embedded environment. In this paper a maximum segment based address mapping policy is introduced, according to the sequential programming characteristic of Flash, and the stress of RAM usage can be relieved while keeping the I/O efficiency. A simple but efficient algorithm is proposed which is evaluated under various workloads. Experiments show that the RAM reduction could reach 95%.

Key words: NAND flash; file system; space management; embedded system

1 引　言

得益于 Flash存储芯片所具有的非易失、低功耗、存储密度高等优点,近年来,基于 NAND Flash(简称 Flash)芯片的存储系统在航天设备以及手持设备 (MP3、Digital Camera)等嵌入式环境中得到了广泛应用[1, 2]. 然而,受到 Flash"擦后写"、擦写速度慢、具有擦写寿命和顺序写入[3]等限制,大容量 Flash存储系统一般需针对上述特性设计专用的文件系统 (Flash File System,简称 FFS).

常见的 FFS主要有两类. 其一是通过 Flash转换层 (Flash Translation Layer,简称 FTL)[4],把 Flash芯片模拟成块设备,隐藏芯片的擦除操作,只提供和磁盘相同的读、写操作. 在 FTL 上则使用传统的基于磁盘的传统文件系统,如 FAT等. 其二是直接在 Flash上实现数据管理,称为 Native FFS,例如 JFFS/UBIFS[5]、Yaffs[6, 7]等.

地址映射是 FFS中存储管理的核心问题. FTL 方案中的地址映射包括:

(1) 虚拟扇区到实际物理页面地址间的映射;

(2) 传统文件系统中维护的目录项到文件信息以及文件偏移到虚拟扇区地址的映射.

由于 Native FFS直接维护了数据文件偏移到 Flash物理页面地址的映射,没有软件层次的限制,因而地址映射的管理更加灵活[8]. 目前,上述映射关系大多在 RAM 中存储和维护. 由于传统的映射算法普遍忽略了实际文件的大小和分布等特点,导致用于维护映射关系的 RAM 的用量普遍偏大. 然而,受到上市时间、产品体积以及价格等诸多因素的限制,Flash存储系统的 RAM 容量往往受到严格限制,因此如何减少 Flash对 RAM 的使用得到广泛的关注.

目前已有若干系统致力于降低 FFS中映射关系的 RAM 占用量. 如:在基于 FTL 的实现方式中, [9]提出了映射粒度可以自适应调整的管理方法,但调整范围仅局限于块级别映射和页面级别映射的切换;在基于 Native FFS的实现方式中, UBIFS把映射信息保存在 Flash上,只需要在 RAM 中保存近期被访问文件的映射关系,但映射信息在 Flash上的更新带来额外的开销; [8]提出将映射信息以 μ-tree结构存储在 Flash上,降低更新操作的开销. 但这些映射算法的映射粒度仍然是单个页面,同样没有充分考虑文件大小和存放方式的特点, RAM 使用量仍然偏大.

针对上述问题,本文在对 Native FFS的存储算法以及文件大小和存储分布等特点进行深入地分析后,引入了区段的概念,提出了基于极大映射区段的地址映射算法,并使用简单的数据结构和高效的算法来维护映射中的区段,既高效地维

收稿日期: 2008-07-25　基金项目:安徽省自然科学基金项目 (070412030)资助;电子信息产业发展基金项目 (财建[2008]329 工信部运[2008]97) 资助.　作者简介:时　正,男, 1982年生,博士研究生,研究方向为嵌入式操作系统;陈香兰,女, 1977年生,博士,研究方向为分布式操作系统.

护了映射关系,又大大减小了 Native FFS 系统的 RAM 占用量.最后通过实验验证了该算法的有效性.

2 区段和映射区段的基本概念

2.1 Native FFS的文件映射的区段连续性特点

Native FFS采用了 LFS (Log-structured File System)[10]的日志结构化思想.整个 Flash被组织成一个循环使用的日志,新数据的写入都采用异地更新方式,追加到日志末尾.这种顺序追加的空间分配策略一方面刚好适应了 Flash介质的顺序写入特点,另一方面使得文件内容被连续的存放在相邻的 Flash页面中.本文称这种特性为文件地址映射的区段连续性特点.

2.2 区段和区段映射的定义

文件的一个区段是指该文件的一段连续内容,使用 (laddr, len)表示.其中, laddr为文件内容的逻辑 page地址; len为文件内容长度.

Flash的一个区段是指 Flash上一段地址连续的页面,使用 (paddr, len)表示.其中, paddr为 Flash 页面物理地址; len为这段存储空间的长度.

如果文件的一个区段被存放在 Flash的一个区段上,则可以使用一个三元组表示该文件区段的地址映射关系,称之为映射区段 (Mapping Segment):

$$ms = (laddr, paddr, len).$$

其中, laddr为文件内容的逻辑 page地址; paddr为实际存储位置的 Flash页面物理地址; len为区段的长度,即地址范围是 [ms. laddr, ms. laddr + ms. len) 的文件内容被连续存放在地址为 [ms. paddr, ms. paddr + ms. len)的 Flash页面中.

给定两个映射区段 (laddr1, paddr1, len1)和 (laddr2, paddr2, len2),不妨设 paddr1 < paddr2.

(1) 若 paddr1 + len = paddr2,则称这两个映射区段物理相邻.

(2) 若这两个映射属于同一个文件,并且 laddr1 + len = laddr2,则称这两个映射区段逻辑相邻.

(3) 若上述两个条件同时成立,则称这两个映射区段绝对相邻.两个绝对相邻的映射区段可以合并为一个映射区段 (laddr1, paddr1, len1 + len2).

对于依次逻辑相邻的三个映射区段 ms1、ms2和 ms3,若 ms1与 ms2、ms2与 ms3都不物理相邻,则称 ms2为一个极大映射区段 (简称 MMS).

2.3 基于极大映射区段的优化思想及其可行性

Flash写操作的最小单位是页.为便于讨论,下面假设地址总是页对齐, len的单位为页, len≥1.

在以页为单位的映射算法中,映射区段长度始终为 1.此时,要存储的映射区段个数与页的多少有关:页越多,占用的 RAM 空间越多;反之,页越少,占用的 RAM 空间越少.若能将找到绝对相邻的映射区段,并将其合并,则可以减少要存储的映射区段个数,绝对相邻区段越多,合并进行得越多,则 RAM 开销越少.鉴于此,本文下面将提出基于极大映射区段的映射算法 (简称 MMSA).该算法中, RAM 中存储的任意一个映射区段都是一个 MMS.

为说明 MMSA 算法的可行性,本文对不同应用负载 (详见第 4节)下的 MMS的大小进行了调研,调研结果表明实际应用中 MMS的平均大小相对较大,例如对于存储在 Flash上的 4～5Mbyte大小的 MP3文件,其平均 MMS大小约为 100个 page.由此可见,使用 MMSA 是有意义的.

3 基于极大映射区段的地址映射算法 MMSA

使用合适的数据结构来存储极大映射区段 MMS是算法关键之一.第 4节的实验表明,虽然文件的 MMS个数与其大小、Flash芯片当前的碎片程度相关,但在各种不同的负载下, MMS个数的均值均不大,使用简单的数据结构即可满足要求.当然,如果区段数较多,也可以使用哈希桶等进行优化.本文使用有序双向链表 (Mapping List)组织文件的 MMS.令 next(ms)表示 ms在链表中的后继映射关系,则

$$ms.\ laddr < ext(ms).\ laddr.$$

在 Native FFS的各种文件操作中,最重要的地址映射操作是查找和更新,下面将着重描述这两个算法.

3.1 映射关系的查找

Native FFS中读文件以页面为单位进行,使用 MMSA 后,逻辑地址到物理地址的映射只需在区段的基础上进行简单的加减运算即可.算法 1给出了获取文件存储位置的方法,该算法首先直接在链表上进行顺序查找,找出对应的区段,然后再根据区段的信息进行地址映射关系计算.

算法 1. 获取文件逻辑页面的物理页面地址

输入:文件内容的逻辑地址 la,文件映射关系链表 ml

输出:实际存储在 Flash上的页面地址

```
1. for each ms in ml
2.     if ms. laddr ≤ la < ms. laddr + length then
3.         pa = ms. paddr + la - ms. laddr;
4.         return pa
5.     end if
6. end for
```

3.2 映射关系的更新

文件的写操作会引发映射关系的变化,具体表现为映射关系链表上 ms的增加 (A)、删除 (D)、拆分 (S)与修改 (M)4种操作,其中拆分可以看成是另外 3个操作的组合.上述操作本身都比较简单,仅需简单运算.

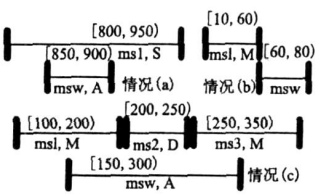

图 1 写操作引发的区段映射关系链表操作

Fig. 1 Mapping List operations caused by writes

由于 MMS存在互相覆盖等复杂情况,因此需要对写操作所涉及的 MMS进行判断,以便区分不同的情况并进行不

同的处理.如上页图 1所示,假设需要处理的新映射关系是 msw,写操作可能引发的区段变化主要有 3种情形.

情况(a). 区段 ms和 msw 存在交叠 (intersects(ms1, ms2)),则:

∃ laddr, msi. laddr≤addr < i. laddr+msi. len, i=1或 2.

特别地,如果 ms1⊆ms2,则:

ms2. laddr≤ms1. laddr,且 ms1. laddr + ms1. len ≤ms2. laddr + ms2. len.

算法将根据区段交叠的情况,再做删除(D),划分(S)或修改操作(M).

例如,在图 1情况(a)中,msw为[850, 900),与原有区段 ms[800, 950)存在交叠,且[850, 900) ⊂ [800, 950),所以需要对 ms进行拆分操作(S).

情况(b). 考虑映射关系 msi和 msj,如果 msi和 msj绝对相邻(记作 aadjacent(msi, msj)),算法将仅修改已存在于 RAM 中的区段.例如在图 1情况(b)中,ms1为[10, 60),msw为[60, 80),根据绝对相邻的定义,仅需修改 ms1为[10, 80).

情况(c). 对于映射关系 ms,定义 ms_left是 ms的最左近邻区段映射关系 leftadjacent(ms),当且仅当

ms_left. laddr < ms. laddr≤next(ms_left). laddr.

对于如图 1情况(c)所示的情况,ms_left就是 ms1,处理 msw时需要:修改 ms1、ms3的区段信息,删除 ms2,并将 msw 插入到 ms_left之后.

综上,映射关系的更新算法如算法 2所示.

文件的删除(delete)以及截断(truncate)操作,还包括对映射关系链表中相应元素的删除,这些操作就是链表元素查找和删除操作的组合,算法 2中的拆分操作(split)和区段修改操作(modify)也仅仅是链表元素属性的修改,都非常简单,不赘述.

算法 2 映射关系的更新

输入:新映射关系 msw,文件映射关系链表 ml

```
1.  for each ms in ml
2.    if intersects(ms, msw) then
3.        if ms⊆msw then
4.        delete ms;
5.        else if msw⊂ms then
6.        split ms;
7.        else
8.        modify ms;
9.      end if
10.   end if
11.   if aadjacent(ms, msw) then
12.       modify ms;
13.       return;
14.   end if
15.     if ms = leftadjacent(msw) then
16.       ms_left = ms;
17.   end if
18.  end for
19.  insert msw after ms_left;
20.  return;
```

3.3 时间复杂度分析及算法优化

使用 MMSA时,文件的读操作只涉及到算法 1,而写操作可能执行算法 2的部分分支.从这两个算法的描述中可以看出,文件的读写操作的复杂度均是 O(n),其中 n是映射关系链表的长度.链表的长度直接影响了该映射算法的效率,第 4节的各次实验指出 n的大小在各种应用负载下都与基于索引树结构的树高 h具有可比性,在表现最差的 MP3型负载下,n=7h(参见表 2),因此可以根据文件访问的空间局部性原理,对有序双向链表做出如下优化:记上次访问的映射关系为 cached_ms,算法 1和算法 2在遍历映射关系链表之前先查看 cached_ms是否满足本次操作的条件,如果不满足,则以 cached_ms为遍历起点,选择恰当的方向进行链表的遍历操作,从而减小 O(n)复杂度带来的开销.

4 实验结果及分析

为验证算法的有效性,本文在 Yaffs的基础上实现了基于极大映射区段的地址空间映射算法(Rlist),并针对常见 4种文件应用负载[8, 11]进行测试,各种负载的特性参见表 1. Yaffs原有的地址映射算法使用索引树结构(Tnode),每个叶子节点保存 16个文件逻辑页面的物理 Flash page地址,占用 64Bytes.而 MMSA 中的每个链表项需要 20Bytes.实验的硬件平台是 Intel Core 2 Duo 2.3GHz, 4G DDR2 400MHz SDRAM. Flash芯片采用 Linux NANDSIM 进行模拟[3].

表 1 各种应用负载特性

Table 1 Characteristic of applications

负载名	描述	文件大小
小文件密集型	Linux source tree的 copy以及 kernel compile	< 20k 66%, 20k ~ 800K 34%
DC(Digital Camera)	以写入、删除部分的递增方式填满 Flash,再全部删除	1MB~2MB
MP3	填满 Flash,删除 50%的文件,再填满,每次删除和填满过程为一次事务,执行指定个数次事务	4MB~5MB
大小文件混合型 XP trace	3小时的 windows XP系统中网络浏览和多媒体应用引发的 IO操作	参见图 4

为了能较好的衡量和比较算法的效果,定义映射关系管理算法的空间占用指标 α——维护文件 1个 page的地址映射所需要的 RAM 大小. Tnode管理算法的空间占用指标 α是固定的,为 64Bytes/16 = 4.而 Rlist管理算法的 α与文件大小以及存储的分散程度有关,分散程度可以用每个文件的所有映射区段长度的平均值 ms_ave_len代表,则 α = 20/ms_ave_len.

下面在 MP3(大文件)型、小文件型、XP trace(大小文件混合)型这 3种不同应用负载下分别对 Tnode映射算法和 Rlist映射算法进行对比分析.

4.1 MP3型应用负载下的 RAM 使用情况

Tnode和 Rlist两种映射算法在 MP3型应用负载下的 RAM 使用情况对比参见图 2,横坐标是每次实验的 Flash容量以及事务个数,一次事务模拟了用户批量更换 MP3播放器

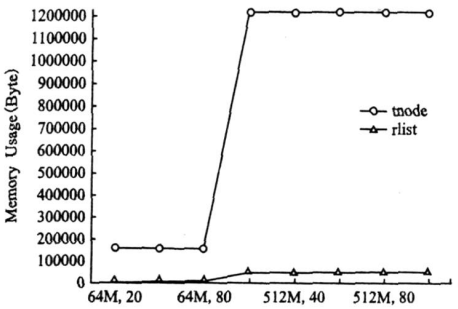

图 2 MP3内存使用测试数据

Fig. 2 RAM usage testing result of MP3 workloads

中文件的过程;纵坐标已给出了每次实验中映射关系占用的 RAM 大小,其中带圆形符号的线是使用 Tnode的映射算法,而带三角的曲线则是使用极大映射区段的 Rlist算法. 从图 2中可以看出在 Flash容量为 512M 的情况下, RAM 使用量降低了 95%;在 64M 的容量下虽然降低的百分比没有那么高,但也显示出相同的趋势 ——Rlis算法大大降低了内存使用量.

表 2 MP3文件在各种测试下的分散程度

Table 2 ms_ave_len under different MP3 file workloads

事务次数	20	40	60	80	100
Tnode高度			3		
ms_ave_len	19	19	19	20	20

表 2给出了 MP3型负载下各次实验中的文件分散程度. 从表中可以看出,随着事务次数变大, ms_ave_len并没有明显变化,也就是空间占用指标 α并没有明显变化. 这与 Yaffs 的垃圾块挑选策略和页面分配策略相关[6].

4.2 小文件型应用负载下的 RAM 使用情况

Tnode和 Rlist两种不同的地址映射算法在小文件型应用负载下的 RAM使用情况对比参见,图中横坐标是编译内

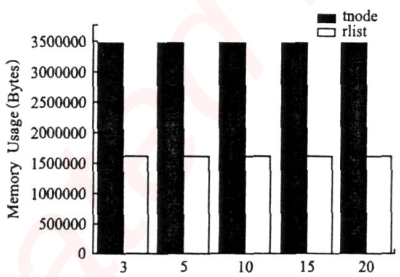

图 3 小文件密集型负载内存使用比较

Fig. 3 RAM usage comparison of small-files workloads

核的次数,纵坐标是内存占用量. 因为单个文件占用的 Flash 页面总个数较小,所以 Yaffs + Rlist的空间占用指标 α上升,且在多事务的情况下又产生了碎片因素,所以 RAM 的用量只比 Yaffs + Tnode算法减小了 50%.

4.3 XP trace应用负载下的 RAM 使用情况

XP trace应用负载是一个大小文件混合型负载,共涉及 1295个普通文件和 240个目录 (大小分布情况见图 4). 在该应用负载下, Tnode算法的 RAM 使用量为 2, 240, 000Bytes而 Rlist算法的 RAM 使用量则为 90, 000Bytes,缩减了 76%,介于小文件密集型和大文件密集型之间.

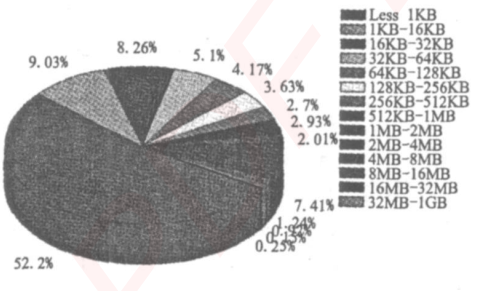

图 4 XP trace负载中的文件大小分布图

Fig. 4 Distribution of file size in XP trace workload

4.4 4种不同负载的 RAM 使用比较

图 5总结了 4种应用负载下的 RAM 用量,可以看出在 DC负载下 RAM用量的优化达到了与 MP3负载相同的效果.

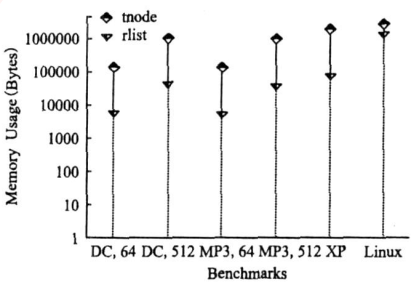

图 5 内存使用比较图

Fig. 5 RAM usage comparison between different workloads

表 3~表 5给出了查找、更新和删除这三种操作在 4种不同负载和 2种不同映射算法下的时间比较 (1个 cycle即一个时钟周期数),其中 Rlis算法又分为使用 3.3节优化策略

表 3 4种负载的查找操作时间比较

Table 3 Search time comparison between four different workloads

单位 (cycle)	Linux	DC	MP3	XP
Tnode	18157	136	139	27398
Rlist + cache	19455	79	78	27295
Rlist no cache	25743	93	147	27384

的 Rlist + cache和不使用该策略的 Rlist no cache 可以看出,除了 Linux负载的更新操作之外, Rlist + cache都要优于或等同于原有的 Tnode算法以及 Rlist no cache,这是因为 3.3节中的缓存算法符合文件访问的空间局部性原理. 而 Linux负载的更新操作可以通过与 Flash的写操作并行化来消除其相对于 Tnode的劣势[8].

综上,使用 MMSA 能够有效的降低映射所需的 RAM 使用量,并在多种不同的负载下均能快速的支持查找、更新、删除等常见操作;该算法能较大的提高大容量 NAND Flash 文件系统中的地址映射效率和速率.

表 4 4种负载的更新操作时间比较

Table 4 Update time comparison between four different workloads

单位 (cycle)	Linux	DC	MP3	XP
Tnode	272	253	261	298
Rlist + cache	1833	205	210	250
Rlist no cache	18454	255	351	408

表 5 4种负载的删除操作时间比较

Table 5 Deletion time comparison between four different workloads

单位 (cycle)	Linux	DC	MP3	XP
Tnode	6156	64231	191247	750
Rlist + cache	2195	1500	3710	296
Rlist no cache	2209	1552	3658	293

5 总结和展望

已有的 Flash 文件系统中存储管理的核心问题是地址映射. 虽然已经有若干系统致力于降低 Flash 文件系统中映射关系的 RAM 占用量,但这些映射算法仍未充分考虑文件大小和存放方式的特点, RAM 使用量仍然偏大. 本文在详细分析了 Native FFS 的存储算法和文件大小特性的基础上,根据文件在 Flash 介质上连续存放的特性,提出了基于极大映射区段的 FFS 地址映射算法. 通过引入极大映射区段的概念并使用简单的数据结构和高效的算法来维护映射中的区段,高效地维护映射关系的同时,大大减小了 Native FFS 系统的 RAM 用量. 各种不同的文件负载下的实验表明,在不影响系统执行效率的前提下,MMSA 能大大降低 RAM 占用量,特别在大文件负载下减少了高达 95%的 RAM 占用量. 考虑文件访问的空间局部型原理之后的优化算法,使该算法能够快速的支持查找、更新、删除操作. 该算法能较大的提高大容量 NAND Flash 文件系统中的地址映射效率和速率. 进一步工作将考虑将区段映射的索引结构高效地保存在 Flash 上,针对不同应用程序的文件访问模式设计更合理的地址映射算法.

References:

[1] Bez R, Camerlenghi E, Modelli A, et al. Introduction to flash memory[C]. Proceedings of the IEEE, 2003, 91: 489-502.

[2] Toshiba Corp., NAND vs. NOR flash memory technology overview [EB/OL]. http://www.toshiba.com/taec/components/Generic/Memory_Resources/NANDvsNOR.pdf 2006.

[3] Electronics S [EB/OL]. http://www.samsung.com/global/business/semiconductor/products/flash/Products_NANDFlash.html, 2008.

[4] Atsuo K, Shingo N, Hiroshi M. A flash-memory based file system [C]. Proceedings of the USENIX 1995 Technical Conference, USENIX Association, New Orleans, Louisiana, January 16-20, 1995.

[5] Hunter A. A brief introduction to the design of UBIFS [EB/OL]. http://www.linux-mtd.infradead.org/doc/ubifs_whitepaper.pdf, 2008.

[6] Aleph One Corp. Yet another flash file system [EB/OL]. http://www.aleph1.co.uk/yaffs, 2008.

[7] Peng Fu-shi, Cai Yong. On NAND flash file system initialisation [J]. Journal of Zhengzhou University of Light Industry (Natural Science Edition), 2007, 22: 5.

[8] Dongwon K, Dawoon J, Jeong-Uk K, et al. μ-tree: an ordered index structure for NAND flash memory[C]. Proceedings of the 7th ACM & IEEE International Conference on Embedded Software, ACM, Salzburg, Austria, 2007, 144-153.

[9] Chin-Hsien W, Tei-wei K. An adaptive two-level management for the flash translation layer in embedded systems[C]. Proceedings of the 2006 IEEE/ACM International Conference on Computer-aided design, ACM, San Jose, California, 2006, 601-606.

[10] Mendel R, John K O. The design and implementation of a log-structured file system[C]. ACM TOCS, 1992, 10(1): 26-52.

[11] Jo H, Kang J U, Park S Y, et al. FAB: flash-aware buffer management policy for portable media players[J]. IEEE Trans. on Consumer Electronics, 2006, 52: 485-493.

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
53. IEEE1588 协议在网络测控系统中的应用
54. USB30 物理层中弹性缓冲的设计与实现
55. USB30 的高速信息传输瓶颈研究
56. 基于 IPv6 的 UDP 通信的实现
57. 一种基于 IPv6 的流媒体传送方案研究与实现
58. 基于 IPv4-IPv6 双栈的 MODBUS-TCP 协议实现
59. RS485CAN 网关设计与实现
60. MVB 周期信息的实时调度
61. RS485 和 PROFINET 网关设计
62. 基于 IPv6 的 Socket 通信的实现
63. MVB 网络重复器的设计
64. 一种新型 MVB 通信板的探究
65. 具有 MVB 接口的输入输出设备的分析
66. 基于 STM32 的 GSM 模块综合应用
67. 基于 ARM7 的 MVB CAN 网关设计
68. 机车车辆的 MVB CAN 总线网关设计
69. 智能变电站冗余网络中 IEEE1588 协议的应用
70. CAN 总线的浅析 CANopen 协议
71. 基于 CANopen 协议实现多电机系统实时控制
72. 以太网时钟同步协议的研究
73. 基于 CANopen 的列车通信网络实现研究
74. 基于 SJA1000 的 CAN 总线智能控制系统设计
75. 基于 CANopen 的运动控制单元的设计
76. 基于 STM32F107VC 的 IEEE 1588 精密时钟同步分析与实现

77. 分布式控制系统精确时钟同步技术
78. 基于 IEEE 1588 的时钟同步技术在分布式系统中应用
79. 基于 SJA1000 的 CAN 总线通讯模块的实现
80. 嵌入式设备的精确时钟同步技术的研究与实现
81. 基于 SJA1000 的 CAN 网桥设计
82. 基于 CAN 总线分布式温室监控系统的设计与实现
83. 基于 DSP 的 CANopen 通讯协议的实现
84. 基于 PCI9656 控制芯片的高速网卡 DMA 设计
85. 基于以太网及串口的数据采集模块设计
86. MVB1 类设备控制器的 FPGA 设计
87. MVB 接口彩色液晶显示诊断单元的显示应用软件设计
88. IPv6 新型套接字的网络编程剖析
89. 基于规则的 IPv4 源程序到 IPv6 源程序的移植方法
90. MVB 网络接口单元的 SOC 解决方案
91. 基于 IPSec 协议的 IPv6 安全研究
92. 具有 VME 总线的车载安全计算机 MVB 通信板卡
93. SD 卡的传输协议和读写程序
94. 基于 SCTP 的 TLS 应用
95. 基于 IPv6 的静态路由实验设计
96. 基于 MVB 的地铁列车司机显示系统研究
97. 基于参数优化批处理的 TLS 协议

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
34. 基于 VxWorks 操作系统的 WindML 图形操控界面实现方法
35. 基于 GPU FPGA 芯片原型的 VxWorks 下驱动软件开发
36. VxWorks 下的多串口卡设计
37. VxWorks 内存管理机制的研究
38. T9 输入法在 Tilcon 下的实现
39. 基于 VxWorks 的 WindML 图形界面开发方法
40. 基于 Tilcon 的 IO 控制板可视化测试软件的设计和实现
41. 基于 VxWorks 的通信服务器实时多任务软件设计
42. 基于 VXWORKS 的 RS485MVB 网关的设计与实现
43. 实时操作系统 VxWorks 在微机保护中的应用
44. 基于 VxWorks 的多任务程序设计及通信管理
45. 基于 Tilcon 的 VxWorks 图形界面开发技术
46. 嵌入式图形系统 Tilcon 及应用研究
47. 基于 VxWorks 的数据采集与重演软件的图形界面的设计与实现
48. 基于嵌入式的 Tilcon 用户图形界面设计与开发
49. 基于 Tilcon 的交互式多页面的设计
50. 基于 Tilcon 的嵌入式系统人机界面开发技术
51. 基于 Tilcon 的指控系统多任务人机交互软件设计
52. 基于 Tilcon 航海标绘台界面设计
53. 基于 Tornado 和 Tilcon 的嵌入式 GIS 图形编辑软件的开发
54. VxWorks 环境下内存文件系统的应用
55. VxWorks 下的多重定时器设计
56. Freescale 的 MPC8641D 的 VxWorks BSP

57. VxWorks 实验五[时间片轮转调度]
58. 解决 VmWare 下下载大型工程.out 出现 WTX Error 0x100de 的问题
59. 基于 VxWorks 系统的 MiniGUI 图形界面开发
60. VxWorks BSP 开发中的 PCI 配置方法
61. VxWorks 在 S3C2410 上的 BSP 设计
62. VxWorks 操作系统中 PCI 总线驱动程序的设计与实现
63. VxWorks 概述
64. 基于 AT91RM9200 的 VxWorks END 网络驱动开发
65. 基于 EBD9200 的 VxWorks BSP 设计和实现
66. 基于 VxWorks 的 BSP 技术分析
67. ARM LPC2210 的 VxWorks BSP 源码
68. 基于 LPC2210 的 VxWorks BSP 移植
69. 基于 VxWorks 平台的 SCTP 协议软件设计实现
70. VxWorks 快速启动的实现方法[上电到应用程序 1 秒]

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
40. 基于嵌入式 Linux 下的 USB30 驱动程序开发方法研究
41. Android 手机应用开发之音乐资源播放器
42. Linux 下以太网的 IPv6 隧道技术的实现
43. Research and design of mobile learning platform based on Android
44. 基于 linux 和 Qt 的串口通信调试器调的设计及应用
45. 在 Linux 平台上基于 QT 的动态图像采集系统的设计
46. 基于 Android 平台的医护查房系统的研究与设计
47. 基于 Android 平台的软件自动化监控工具的设计开发
48. 基于 Android 的视频软硬解码及渲染的对比研究与实现
49. 基于 Android 移动设备的加速度传感器技术研究
50. 基于 Android 系统振动测试仪研究
51. 基于缓存竞争优化的 Linux 进程调度策略
52. Linux 基于 W83697 和 W83977 的 UART 串口驱动开发文档
53. 基于 AT91RM9200 的嵌入式 Linux 系统的移植与实现
54. 路由信息协议在 Linux 平台上的实现
55. Linux 下 IPv6 高级路由器的实现

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
22. Windows XP Embedded 在变电站通信管理机中的应用
23. XPE 在多功能显控台上的开发与应用
24. 基于 Windows XP Embedded 的 LKJ2000 仿真系统设计与实现
25. 虚拟仪器的 Windows XP Embedded 操作系统开发
26. 基于 EVC 的嵌入式导航电子地图设计
27. 基于 XPEmbedded 的警务区 SMS 指挥平台的设计与实现
28. 基于 XPE 的数字残币兑换工具开发
29. Windows CENET 下 ADC 驱动开发设计
30. Windows CE 下 USB 设备流驱动开发与设计
31. Windows 驱动程序设计
32. 基于 Windows CE 的 GPS 应用
33. 基于 Windows CE 下大像素图像分块显示算法的研究
34. 基于 Windows CE 的数控软件开发与实现

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
20. 基于 PowerPC 的模拟量输入接口扩展
21. 基于 PowerPC 的车载通信系统设计
22. 基于 PowerPC 的嵌入式系统中通用 IO 口的扩展方法
23. 基于 PowerPC440GP 型微控制器的嵌入式系统设计与研究
24. 基于双 PowerPC 7447A 处理器的嵌入式系统硬件设计
25. 基于 PowerPC603e 通用处理模块的设计与实现
26. 嵌入式微机 MPC555 驻留片内监控器的开发与实现
27. 基于 PowerPC 和 DSP 的电能质量在线监测装置的研制
28. 基于 PowerPC 架构多核处理器嵌入式系统硬件设计
29. 基于 PowerPC 的多屏系统设计
30. 基于 PowerPC 的嵌入式 SMP 系统设计
31. 基于 MPC850 的多功能通信管理器
32. 基于 MPC8640D 处理系统的技术研究
33. 基于双核 MPC8641D 处理器的计算机模块设计
34. 基于 MPC8641D 处理器的对称多处理技术研究

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
29. 基于 S3C6410 的 Uboot 分析与移植
30. 基于 ARM 嵌入式系统的高保真无损音乐播放器设计
31. UBoot 在 Mini6410 上的移植
32. 基于 ARM11 的嵌入式 Linux NAND FLASH 模拟 U 盘挂载分析与实现
33. 基于 ARM11 的电源完整性分析
34. 基于 ARM S3C6410 的 uboot 分析与移植
35. 基于 S5PC100 移动视频监控终端的设计与实现
36. UBoot 在 AT91RM9200 上的移植简析
37. 基于工控级 AT91RM9200 开发板的 UBoot 移植分析

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
25. 基于龙芯 2F 架构的 PMON 分析与优化
26. CPU 与 GPU 之间接口电路的设计与实现
27. 基于龙芯 1A 平台的 PMON 源码编译和启动分析
28. 基于 PC104 工控机的嵌入式直流监控装置的设计
29. GPGPU 技术研究与发展
30. GPU 实现的高速 FIR 数字滤波算法
31. 一种基于 CPUGPU 异构计算的混合编程模型
32. 面向 OpenCL 模型的 GPU 性能优化
33. 基于 GPU 的 FDTD 算法
34. 基于 GPU 的瑕疵检测
35. 基于 GPU 通用计算的分析与研究
36. 面向 OpenCL 架构的 GPGPU 量化性能模型
37. 基于 OpenCL 的图像积分图算法优化研究
38. 基于 OpenCL 的均值平移算法在多个众核平台的性能优化研究
39. 基于 OpenCL 的异构系统并行编程
40. 嵌入式系统中热备份双机切换技术研究
41. EFI-Tiano 环境下的 AES 算法应用模型
42. EFI 及其安全性研究
43. 基于 UEFI Shell 的 PreOS Application 的开发与研究

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
7. 数据结构考题 - 第 1 章 绪论
8. 数据结构考题 - 第 2 章 线性表
9. 数据结构考题 - 第 2 章 线性表 - 答案
10. 基于小波变换与偏微分方程的图像分解及边缘检测
11. 基于图像能量的布匹瑕疵检测方法
12. 基于 OpenCL 的拉普拉斯图像增强算法优化研究
13. 异构平台上基于 OpenCL 的 FFT 实现与优化
14. 数据结构考题 - 第 4 章 串
15. 数据结构考题 - 第 4 章 串答案
16. 用 IPv6 编程接口实现有连接通信的方法

FPGA / CPLD:

1. 一种基于并行处理器的快速车道线检测系统及 FPGA 实现
2. 基于 FPGA 和 DSP 的 DBF 实现
3. 高速浮点运算单元的 FPGA 实现
4. DLMS 算法的脉动阵结构设计及 FPGA 实现
5. 一种基于 FPGA 的 3DES 加密算法实现
6. 可编程 FIR 滤波器的 FPGA 实现
7. 基于 FPGA 的 AES 加密算法的高速实现
8. 基于 FPGA 的精确时钟同步方法
9. 应用分布式算法在 FPGA 平台实现 FIR 低通滤波器
10. 流水线技术在用 FPGA 实现高速 DSP 运算中的应用
11. 基于 FPGA 的 CAN 总线通信节点设计
12. 基于 FPGA 的高速时钟数据恢复电路的实现
13. 基于 FPGA 的高阶高速 FIR 滤波器设计与实现
14. 基于 FPGA 高效实现 FIR 滤波器的研究
15. FPGA 的 VHDL 设计策略
16. 用 FPGA 实现串口通信的设计
17. GPIB 接口的 FPGA 实现