第 34 卷 第 5 期 计 算 机 工 程 2008 年 3 月
Vol.34 No.5 Computer Engineering March 2008
·开发研究与设计技术· 文章编号：1000—3428(2008)05—0257—03 文献标识码：A 中图分类号：TP311

YAFFS 文件系统的研究与改进

孙 丰[1,2]，张福新[1]
(1. 中国科学院计算技术研究所，北京 100080；2. 中国科学院研究生院，北京 100080)

摘 要：通过对 NAND Flash 硬件特点和 YAFFS 文件系统的分析与研究，在遵循 NAND Flash “只写一次”限制的基础上，提出改进删除页操作的新策略，给出改进的实现过程，重点涉及实现中的难点及相应的解决策略。在一款龙芯开发板上的测试表明，该改进策略能够把改写、删除和截短文件等文件系统基础操作的性能提高约 40%，具有很高的实用价值。

关键词：NAND 闪存；YAFFS 文件系统；删除策略

Research and Improvement of YAFFS File System

SUN Feng[1,2], ZHANG Fu-xin[1]
(1. Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080;
2. Graduate School, Chinese Academy of Sciences, Beijing 100080)

【Abstract】This paper researches some hardware characteristics of the NAND Flash and the YAFFS Flash memory log file system. On the basis of following the restriction of “write once” on the NAND Flash, it puts forward a new strategy to improve on deleting a paper, and provides an implementation method, focussing on difficulties in realization and relevant solutions. Test on FPGA platform integrated with GODSON-1 IP core indicates that the new strategy can raise the performance of the file system’s basic operations such as rewriting, deleting or resizing a file by nearly 40%, so it has high practical value.

【Key words】NAND Flash memory; Yet Another Flash File System(YAFFS); deletion strategy

YAFFS(Yet Another Flash File System)类似于JFFS/JFFS2，是专门为NAND闪存设计的嵌入式文件系统，适用于大容量的存储设备。它是日志结构的文件系统，提供了损耗平衡和掉电保护，可以有效地避免意外掉电对文件系统一致性和完整性的影响[1]。

1 NAND Flash 硬件特点

NAND闪存的存储单元为页和块。一般来说，128 MB以下容量芯片的一页大小为 528 B，依次分为 2 个 256 B的主数据区，最后是 16 B的备用空间；一个块由若干页组成，通常为 32 页；一个存储设备又由若干块组成。与其他存储器相比，NAND闪存具有以下特点：不是完全可靠的，每块芯片出厂时都有一定比例的坏块存在；各个存储单元是不可直接改写的，在每次改写操作之前需要先擦除；擦除操作以块为单位进行，而读写操作通常以页为单位进行；各块的擦除次数有限，一般为 10 万~100 万次；使用复杂的I/O口串行存取数据[2]。

2 YAFFS 简介

YAFFS充分考虑了NAND闪存的特点，根据NAND闪存以页面为单位存取的特点，将文件组织成固定大小(512 B)的数据页。每个文件都有一个页面专门存放文件头，文件头保存了文件的模式、所有者id、组id、长度、文件名等信息。利用NAND闪存提供的每个页面 16 B的备用空间来存放ECC(Error Correction Code)和文件系统的组织信息。备用空间中 6 个字节被用作页面数据的ECC，2 个字节分别用作块状态字和数据状态字，其余的 8 个字节用来存放文件系统的组织信息[3]。

YAFFS在文件进行改写时总是先写入新的数据页，然后将旧的数据页从文件中删除。删除的旧数据页只有被擦除后才能成为干净页，重新被使用。YAFFS中用数据结构来描述每个擦除块的状态。该数据结构记录了块状态，用一个 32 b的位图表示块内各个页面的使用情况。在YAFFS中，有且仅有一个块处于“当前分配”状态。新页面从当前进行分配的块中顺序进行分配，若当前块已满，则顺序寻找下一个空闲块[3]。

3 YAFFS 删除页策略的分析与改进

3.1 删除页策略分析

YAFFS 删除页函数主要涉及如下步骤：(1)置备用空间中的数据状态字标志为 0，表示该页为删除页；(2)调用底层函数写删除页的备用空间，将数据状态字标志写到 Flash 物理介质上；(3)修改内存中文件页管理数据结构 32 b 的位图，置对应删除页的位值为 0；(4)判断该删除页所在的块是否为脏块，如果是则调用块擦除函数进行块擦除操作。

文件系统删一页数据就是对该删除页的备用空间中数据状态字作标记 0，即要写该页的备用空间，这就需要访问 Flash 物理介质，消耗比较长的时间。而删除页操作关联到一些其他函数操作，如改写文件、删除文件、截短文件操作。改写文件时，需要删除旧页，写入新页。删除文件时，需要将该文件的所有页删除。截短文件时，需要根据文件新的长度(小于原来长度)，将超过长度的部分删除。

作者简介：孙 丰(1981－)，女，硕士研究生，主研方向：嵌入式系统；张福新，副研究员、博士

收稿日期：2007-03-20 **E-mail：**chunxiangmo@163.com

因此，如果能优化删除页的操作，就能同时提高改写文件、删除文件、截短文件函数操作的性能，减少这些操作所花费的时间。而且减少的时间是跟文件的大小成比例的，文件越大，减少的时间越多。

3.2 改进方法

通过上面的分析可知，删除页被擦除之前，写了一次该页的数据区域和 2 次备用空间。写新数据时，将数据写进数据区域，将文件组织信息写入备用空间；在删除该页时，又写了一次该页的备用空间。

如果彻底遵循 NAND Flash“只写一次”的限制，所有页只允许写一次数据空间和备用空间。当删除旧页时，不需要再写该页的备用空间，真正做到“只写一次”，这样可以大大提高改写文件、删除文件、截短文件操作的性能。

原新旧页的区别是依据备用空间中的数据状态字标记的，现在为了区分新旧页，需要在备用空间中增加一个新的域，叫做周期使用计数值(cyclesequence)。在分配新块时，是按块序递增分配的，同一块内又按页序递增分配；当一轮分配结束时，又重新从块 0 分配，这时 cyclesequence 值就增 1，即 cyclesequence 又表示周期使用增长数。当写一页数据时，就要写入此时的周期使用计数值，删一页时，就无须再写该页备用空间了。新旧页的区别就在于这个周期使用计数值和该页在 Flash 中的自然顺序。若 cyclesequence 值大，则为新页；若 cyclesequence 相同，则比较页在 Flash 中的物理地址，地址大的为新页。

周期使用计数值位数的确定：考虑到各方面的因素，采用 32 位计数比较合适。可以大概计算一下cyclesequence的使用寿命：假定 1 s可分配 8 个物理块，cyclesequence的最大值假定就为 2^{32}，其他暂不考虑，则连续不停地进行分配，cyclesequence 可以使用的时间计算如下：$2^{32}\div2^{3}\div60\div60\div24\div356>17$ 年，因此，采用 32 位计数是可行的。

3.3 具体实现

3.3.1 数据结构的修改

在一页备用空间的文件组织信息结构 yaffs_Tags 中增加新的成员：周期使用计数值 cyclesequence。yaffs_Tags 结构修改后如图 1 所示。

bits	Content
20	ChunkID，该page在一个文件内的索引号，因此，文件大小被限制在2^{20} Page即512 MB
2	2 bit serial number
10	ByteCount该page内的有效字节数
18	ObjectID对象ID号，用来唯一标识一个文件
2	2 bit unused
12	ECC,Yaffs_Tags本身的ECC校验和
32	cyclesequence周期使用计数值

图 1 yaffs_Tags 结构

为了在删除文件、写文件、读文件等操作中修改及维护 cyclesequence 域的值，实现新的删除页策略，除了在 yaffs_Tags 结构中增加一个成员 cyclesequence 外，还要在设备对象 yaffs_Device 、文件对象 yaffs_Object 和块对象 yaffs_BlockInfo 数据结构中同样增加 cyclesequence 成员。

3.3.2 函数的修改

(1)修改删除页操作，无须写删除页的备用空间，只要修改一下内存的文件页管理数据结构，并判断该删除页所在的块是否为脏块，是否需要调用块擦除函数。还有一些函数涉及到周期使用计数值 cyclesequence 的操作或维护，需要添加相应的代码。

(2)修改删除文件函数。对被删除的文件，要写新文件头，在新文件头中将被删除文件的父目录置为 unlinked。若该文件在此次断电之前没有被垃圾回收，那么在下次初始化扫描时，如该文件头的父目录为 unlinked，表示该文件为已删除文件，防止了删除不掉的情况。

(3)修改截短文件函数。要重写文件头，在新的页上写新的文件头，标识已经变化的文件长度。在初始扫描时，可根据新文件头中的文件长度判断页数据是否可以加入文件中，超过文件长度的数据就可以不加入文件。

(4)依据周期使用计数值 cyclesequence 来区分新旧页，再加上垃圾回收操作可能引起数据存储位置的变动，使得在初始化扫描期间很难进行准确的判断。会出现回收了文件头，但文件的数据页还没有回收的情况，在初始扫描中，会发现一个文件没有文件头。因此，要修改垃圾回收函数，使文件的数据页先回收，等文件数据页全部回收了，再回收文件头。

(5)修改文件系统初始化扫描函数，扫描整个 Flash 介质时，按 cyclesequence 值从大到小的顺序进行块扫描，在块中也是按从大到小的页序进行页扫描。这样做是符合日志文件记录数据方式的。最新修改的数据总是记录在当前 cyclesequence 值最大的块中，因此，随着 cyclesequence 值由大到小，块上记录的数据也是从新到旧。

3.4 性能测试与评价

3.4.1 测试环境介绍

测试目标板采用的是“龙芯 1 ”SoC的FPGA原型验证板，已在目标板上成功地移植了μC/OS-II嵌入式操作系统。该SoC集成了“龙芯 1 ”RISC CPU以及其他 12 种IP核，其中，NAND Flash采用的是三星公司 32 MB的K9F5608U0C- PCB0，这款器件数据组织如下：器件总容量是 32 MB的主数据区域加 1 MB的备用空间，共由 2 048 个块组成[4]，1 个块又由 32 页组成，一页数据容量为 528 B，分为 512 B的主数据区域和 16 B的备用空间[4]。

首先将 YAFFS 文件系统移植到开发板上，在移植成功的基础上，对 YAFFS 删除页策略进行优化改进，将原 YAFFS 和改进后的 YAFFS 进行性能测试比较。

3.4.2 改进前后的性能比较

采用了新的删除页策略后，测试改写文件和截短文件这 2 个操作，将改进前后所花的时间进行对比。测试结果如下：

(1)测试改写文件操作。改进前后改写操作花费时间对比如图 2 所示。

从图 2 中可以看出，改写操作在改进后的 YAFFS 中性能得到了提高，花费的时间明显减少了。该操作花费的时间与文件大小成正比，随着文件长度的增加，减少的时间更多。当文件大小是 28 MB 时，原改写操作需要 370 s，而改进后的改写操作只要 231 s，整整减少了 139 s。减少的原因就在于 YAFFS 中改写操作涉及到了写新页和删除旧页，而在改进后的 YAFFS 只要写新页，无须删除旧页，因此，改写操作在改进后的 YAFFS 中性能得到了提高。经大量测试数据可计算出此操作的平均优化比例为 38.1%。

(2)测试截短文件操作。测试的文件假定都从原来的大小截短为 0。改进前后截短操作花费时间的对比如图 3 所示。

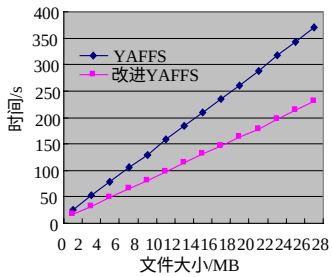

图 2　改进前后改写操作花费时间的对比

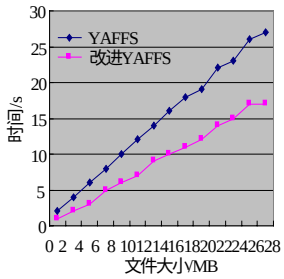

图 3　改进前后截短操作花费时间的对比

从图 3 中可以看到，在改进后的 YAFFS 中截短文件操作比在原来的 YAFFS 中花费的时间减少了，性能得到了提高。因为对超过文件长度的数据页都要调用删除页函数进行删除，所以新的删除页策略减少了截短操作的花费时间。虽然在改进后的 YAFFS 中，截短文件函数还要在最后增加一个写新文件头的操作，标识文件的新长度，但总的来说，截短文件函数的操作性能还是得到了很大的提高。从图中可以看出缩短的时间与文件大小大致成正比，当文件大小为 28 MB，将其截短为 0 时，消耗的时间就能减少 10 s。大量测试数据计算出此操作的平均优化比例为 40.1%。

4　结束语

本文通过对 YAFFS 文件系统管理策略的分析，提出了新的删除页策略，从而提高了 YAFFS 文件系统中改写文件、删除文件和截短文件操作的性能。改进后的 YAFFS 文件系统已成功移植到“龙芯 1”SoC 目标板上，整合进μC/OS-II 操作系统中，通过了一系列性能测试，取得了满意的结果。

参考文献

[1] Rosenblum M, Ousterhout J K. The Design and Implementation of a Log-structured File System[J]. ACM Transactions on Computer Systems, 1992, 10(1): 26-56.

[2] 李红燕, 王　力. 日志结构文件系统技术的研究[J]. 计算机应用研究, 2003, 20(1): 73-76.

[3] 毛勇强, 黄光明. YAFFS 文件系统在嵌入式 Linux 上的实现[J]. 电子设计应用, 2006, (1): 96-98.

[4] 32M x 8 Bit, 16M x 16 Bit NAND Flash Memory(Revision 2.5)[Z]. Samsung Electronics Co., 2003.

（上接第 247 页）

Interface 页面下设置组件的名称为 MY_LCD，组件类型为 avalon_slave，其他的按照默认参数设置即可。

(3)将上述的目录及文件定义路径在 SOPC Builder 的 IP 库文件夹里，设定目录名与该 IP 模块的名称均为 MY_LCD。完成后在 SOPC Builder 里就可以在模块选择栏内就能找到自己所建立的构件的 IP 模块。

4　结束语

本文对基于 Nios II 平台下的 LCD 驱动 IP 核的设计进行初步的研究，经过测试，此驱动适用于各型号的 12 英寸的液晶屏。对于其他尺寸的液晶屏只需要修改屏幕相关参数即可。

本文创新点在于将 LCD 的驱动 IP 核进行紧耦合的封装，构成一个模块化的独立组件，结合 Nios II 强大的可扩展性，使得该 IP 核驱动可以应用于不同型号的液晶屏，能够有效地控制成本和减少开发时间，有利于模块的重复利用。

参考文献

[1] 姚胜兴. 基于嵌入式系统的彩色液晶显示驱动控制[J]. 微计算机信息, 2007, 23(1): 57-59.

[2] Altera Inc. Nios II Processor Reference Handbook ver 6.1[EB/OL]. (2006-12-01). www.altera.com/literature/hb/nios2/n2cpu_nii5v1.pdf.

[3] Hall T S, Hamblen J O. System On a Programmable Chip Development Platforms in the Classroom[J]. IEEE Transactions on Education, 2004, 47(4): 502-507.

[4] 沈振乾, 孔　冰, 常海峰. 基于 NIOS 嵌入式软核处理器的 LCD 控制方法研究[J]. 天津工业大学学报, 2006, 25(4): 94-96.

[5] Altera Inc. Nios Development Kit, Cyclone Edition Getting Started User Guide[EB/OL]. (2003-05-01). http://www.altera.com.cn/products/devkits/altera/kit-niosii-2S60.html.

（上接第 256 页）

里只是以 Sigmoid 函数为例介绍了这种方法，但任何单值非线性函数，甚至任何单值函数的定点数实现都可以用此方法。例如乘法器可以将 2 个乘数并列作为函数的输入、乘积作为输出，便可列出函数的真值表，并用此方法设计出门级电路，加法器也是如此，对于乘累加计算部件还可以将此方法与分布式算法配合使用，对其中的查找表进行化简及优化实现。此方法概念清晰、简单易行，实现的电路结构规整，当定点数位较少时速度优势明显，结果表明这种方法具有很强的实用性。目前，随着可编程逻辑器件的普及，越来越多的研究人员已致力于将先进信号处理或控制算法应用数字 VLSI 实现，希望本文提出的 LMN 方法能为这些算法中复杂(非线性)函数的实现提供一些启发和参考。

参考文献

[1] Ferreira P, Ribeiro P, Antunes A, et al. Artificial Neural Networks Processor: A Hardware Implementation Using A FPGA[C]//Proc. of the 4th International Conference on Field-programmable Logic and Applications. Leuven, Belgium: [s. n.], 2004.

[2] Lu Chun, Shi Bingxue, Chen Lu. A General-purpose Neural Network with On-chip BP Learning[C]//Proceedings of IEEE International Symposium on Circuits and Systems. Greece: [s. n.], 2002: 520-523.

[3] Sammut K M, Jones S R. Implementing Nonlinear Activation Function in Neural Network Emulators[J]. Electronic Letters, 1991, 27(12): 1037-1038.

[4] Amin H, Curtis K M, Hayes-gill B R. Piecewise Linear Approximation Applied to Nonlinear Function of a Neural Network [J]. IEE Proceedings: Circuits, Devices and Systems, 1997, 144(6): 313-317.

[5] Tocci R J, Widmer N S, Moss G L. 数字系统原理与应用[M]. 北京: 机械工业出版社, 2006.

[6] Zhang M, Vassiliadis S, Delgago-Frias J G. Sigmoid Generators for Neural Computing Using Piecewise Approximations[J]. IEEE Trans. on Computer, 1996, 45(9): 1045-1049.