第 40 卷 第 2 期 计 算 机 工 程 2014 年 2 月
Vol.40 No.2 Computer Engineering February 2014

·开发研究与工程应用· 文章编号：1000—3428(2014)02—0300—03 文献标识码：A 中图分类号：TP316

# 一种混合映射闪存转换层的设计与实现

郁志平[1,2]，刘 伟[1,2]，彭 虎[3]，张耀辉[1,2]

(1. 中国科学院苏州纳米技术与纳米仿生研究所系统集成部，江苏 苏州 215123；2. 中国科学院大学，北京 100049；
3. 记忆科技研究院，江苏 苏州 215123)

**摘 要：**使用 NAND Flash 作为存储媒介的存储设备常需要闪存转换层(FTL)对 NAND 进行管理。页映射是一种常见的映射方式，但需要很大的内存存放页映射表，在嵌入式环境下这一条件往往无法满足。针对该问题，提出一种基于超级块的混合映射 FTL，包括坏块管理、地址翻译、垃圾回收、上电恢复，使用的 SRAM 空间不到 128 KB，远小于页映射，同时不需要存储映射表，程序在固态硬盘开发板上成功运行，实现固态硬盘基本读写功能。测试结果表明，该混合映射 FTL 方案具有较好的顺序读写性能。

**关键词：**固态硬盘；NAND 闪存；闪存转换层；混合映射；超级块；垃圾回收

# Design and Implementation of a Hybrid Mapping Flash Translation Layer

YU Zhi-ping[1,2], LIU Wei[1,2], PENG Hu[3], ZHANG Yao-hui[1,2]

(1. System Integration Division, Suzhou Institute of Nano-tech and Nanobionics, Chinese Academy of Sciences, Suzhou 215123, China;
2. University of Chinese Academy of Sciences, Beijing 100049, China;
3. Memory Science and Technology Research Institute, Suzhou 215123, China)

**【Abstract】**Flash Translation Layer(FTL) is always needed when NAND Flash is used in a memory device. Page level translation is the most popular, but needs a large RAM to store mapping table. This pager employs a superblock based hybrid mapping, and realizes a basic FTL on the Solid State Disk(SSD) development board. The test result shows that the hybrid FTL has good sequential read and write performance, and only needs 128 KB SRAM, which is far less than page level mapping. It also does not need to store mapping table. The program runs successfully in solid state disk development board, to achieve the basic functions of solid state disk read and write. Test results show that, the hybrid mapping FTL scheme has better sequential read and write performance.

**【Key words】**Solid State Disk(SSD); NAND Flash; Flash Translation Layer(FTL); hybrid mapping; superblock; garbage collection

**DOI:** 10.3969/j.issn.1000-3428.2014.02.065

## 1 概述

NAND 闪存(NAND Flash)具有体积小、重量轻、功耗低、抗震性强，并且性能高等优点，目前已经得到广泛应用。NAND 的容量越来越大，而平均每字节的价格却在下降[1]，基于 NAND 的固态硬盘(Solid State Disk, SSD)向传统机械磁盘发起了挑战[2]。由于 NAND 写之前需要先做擦除，闪存转换层(Flash Translation Layer, FTL)正是基于此特性，将上层应用与底层 NAND 衔接起来，隐藏了 NAND 需要先擦后写的限制。

本文采用一种混合映射的方式，实现一套基本的 FTL 方案，并且在固态硬盘开发板上运行。

## 2 研究背景

### 2.1 NAND Flash 特性

NAND 最小读写单位是页，擦除单位是块，块由页组成[3]。一个页分为主区和备用区，主区可以用来存放用户数据，备用区可以用来存放 ECC 编码[4-5]和其他管理数据。备用区和主区可一起进行读写操作，所以经常将一些管理信息放在备用区中，以保证在掉电的情况下用户数据与管理信息的一致性。在实际运用中，可能会使用多块 NAND 颗粒，NAND 控制器一般支持多通道并行处理 NAND 命令[6]。

### 2.2 页映射、块映射与混合映射

页映射是一种细颗粒度的地址翻译策略，它在内存中

**作者简介：**郁志平(1988－)，男，硕士研究生，主研方向：嵌入式系统；刘 伟，博士研究生；彭 虎，工程师；张耀辉，研究员、博士生导师

**收稿日期：**2013-01-22 **修回日期：**2013-03-13 **E-mail：**zpyu2011@sinano.ac.cn

维护了完整的页映射表，每个逻辑页可以映射到 NAND 的任何一个物理页内。页映射有非常高效的块利用率和垃圾回收的效率，其读写性能也是最好的。然而页映射需要很大的内存空间来存放完整的映射表，而在内存空间受限的系统中是无法满足这种要求的[7]。相反，块映射是另一种极端情况，其映射颗粒度非常大，为一个块的大小，一个逻辑块可以映射到 NAND 的任何一个物理块上，但是块内页的写入位置完全绑定。块映射具有较低的块利用率和垃圾回收效率，同时读写性能也相对较差，然而块映射需要的内存空间很小。于是，一种介于两者之间的混合映射被提出来，其目的在于通过结合块映射和页映射，在内存开销和性能之间有一个很好的折衷[8-9]。

### 2.3 闪存转换层

FTL 的主要功能是将 NAND 抽象成一个块设备[10]，一个基本的 FTL 算法包含以下 4 个方面：

(1)地址翻译：负责将逻辑地址转换为 NAND 的物理地址，也就是某个页的地址。

(2)垃圾回收：NAND 是不能被复写的，当 NAND 的块被耗尽时，必须先将若干个源块中的有效页搬移到一个空闲块中，再将那些源块擦除掉，这样就会有富余的块来维持继续写入。垃圾回收的效率很大程度上影响固态硬盘的性能。

(3)坏块管理：NAND 的一个块在使用过程中，因为擦除次数变大的原因，会变成坏块，就不能拿来使用了。另外，NAND 出厂时本身就有少量的坏块，这些坏块的信息必须被记录下来，保证不能使用这些块，否则数据很容易出错。

(4)掉电恢复：硬盘在使用过程中会断电，存在内存中的映射表和其他管理信息都会丢失，上电过程中需要重建这些信息，保证这些信息和上次断电前是一致的。

## 3 基于 Superblock 的混合映射 FTL

在页映射中，内存大部分空间用来存放页映射表，每执行一次写操作，需要更新页映射表，这就面临着掉电映射表丢失的问题，所以需要经常将页映射表存储到 NAND 上，以减少上电恢复时重建页映射表的时间。以此为出发点，考虑将映射信息存放在 NAND 的备用区中，每进行一次写，新的映射信息更新在备用区中，由于位于主区的用户数据和位于备用区的映射信息是同时写入 NAND 的，因此有很好的一致性，并且完全不需要额外再存储映射表。基于此想法，采用一种基于超级块的混合映射方法[11]，设计并实现了一套基本的 FTL 方案。

### 3.1 地址映射

整个逻辑地址空间会被划分为 $n$ 个逻辑超级块，一个逻辑块包含 $m$ 个逻辑页，每个逻辑块最多会映射到 $s$ 个物理块上，超级块内部采用页映射，充分利用每个物理块，超级块外部采用块映射，尽量减少内存的开销。映射信息存放在 2 个地方：

(1)内存中维护一套完整的超级块映射表，用作超级块的逻辑地址到最新物理地址的索引，因为逻辑块会映射到多个物理块上，超级块映射表中保存的是最新的写入地址——最新写入的块号和页号。

(2)超级块内部的页映射信息存放在 NAND 的备用区中，当需要查询页映射信息时，需要读取 NAND 的备用区，该映射信息本身无需内存开销。

由于 NAND 的备用区空间有限，无法存放逻辑块内部的所有页映射信息，一般采用分组的方式，将 $m$ 个逻辑页的映射信息分成 $t$ 组，每组包含 $p$ 个逻辑页。相应的，NAND 的备用区中的映射信息由 2 个部分组成：(1)组映射表，用以查询一个组的映射信息所在的位置；(2)组内页映射表，存放一个组内成员的页映射信息。

如图 1 所示，一个主机的逻辑页地址会被划分为 3 段，用以索引不同区域的映射信息。当进行读取操作时，需要分 3 步完成：

(1)逻辑块号用来查询位于 SRAM 中的超级块映射表，找到该逻辑块的最新写入地址，并且将其备用区的映射信息读取出来。

(2)根据组号索引第 1 步读到的备用区的组映射表，找到该组最新映射信息的地址，将其备用区的映射信息读取出来。

(3)根据组内偏移索引第 2 步读到的备用区的组内页映射表，找到该逻辑页对应的物理地址，读取该页，得到用户数据。

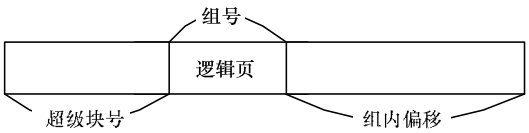


**图 1 逻辑页分段索引**

当进行写入操作时：首先，根据超级块映射表得到该超级块最新的地址——块号和页号，递增得到了下一个写入地址，若已写满则申请新的空闲块；其次，读取该组最新的映射信息，其操作过程和读操作类似，更新组内页映射表和组映射表，将更新后的映射信息和用户数据一同写入到 NAND 上，同时更新超级块映射表。

综上，基于超级块的混合映射通过超级块映射表记录逻辑块的最新写入地址，每一个超级块的最后写入页的备用区中的组映射表是最新的，通过该组映射表就可以找到组内页映射表所在的地址，由该组内页映射表得到一个逻辑页对应的物理地址。所以，一次读或写操作会额外需要最多 2 次备用区的读取。

### 3.2 Cache 机制

读写 SRAM 中表花费的时间很短，而读取 NAND 花费的时间很长。因为一次读或写操作额外需要最多 2 次备用区的读取，所以可以在 SRAM 中为备用区的映射信息建立

多个 cache[12]，以达到提升性能的作用。

备用区中的映射信息有 2 类，即组映射表和组内页映射表。组映射表和组内页映射表的大小和超级块的大小以及 $t$、$p$ 的值有关系。在超级块大小固定的情况下，分组越少，即 $t$ 越小，组映射表也就越小，而 $p$ 和组内页映射表也就越大。组映射表 cache 命中可以直接得到组内页映射表所在的地址，减少一次备用区的读，组内页映射表 cache 命中可以直接得到逻辑页的物理地址，又可以减少一次备用区的读，当组内页映射表和组映射表同时命中时，通过 3 次 SRAM 内的查表(包括第 1 次查询超级块映射表)，就可以得到一个逻辑页的完整映射信息。读操作时，cache 是只读的；写入时，需要对 cache 做更新，然后写入到备用区中。少量的 cache 可以大大提升顺序读写的性能，因为 cache 命中的效率很高，而随机读写时，cache 命中的概率与 cache 的个数有关。

### 3.3 块管理

NAND 的块一共有 3 种状态：(1)空闲块，擦除了但还未写的块；(2)数据块，含有有效数据的块；(3)坏块，不能用来存储数据的块。为了表征每一个块的状态，在 SRAM 中有一个空闲块表，空闲块表用 1 bit 来表征块的状态，即空闲块和非空闲块，非空闲块包含了数据块和坏块。空闲块表主要为写入时分配空闲块的提供依据，同时也作为垃圾回收触发一个条件。另外，在 NAND 的特定几个块上，存有坏块表，坏块表也用 1 bit 来表征块的状态，即坏块和非坏块。空闲块表不需要保存到 NAND 上，它是在上电的过程中建立起来的。坏块表不需要存放在 SRAM 中，在上电的过程中会读取坏块表，作为建立空闲块表的一个步骤，在使用过程中，新出现的坏块会被更新到 NAND 上。

### 3.4 垃圾回收

一个超级块最多会映射到 $s$ 个物理块上，也就是说凡是属于该超级块的逻辑页最终只能写入到分配的 $s$ 个物理块内，属于不同超级块的逻辑页是不可能落在同一个物理块内的。当块耗尽时，需要通过垃圾回收，挪出富余的块来维持继续写入，在基于超级块的混合映射下，块耗尽有 2 层含义：(1)总的空闲块不足；(2)某个超级块已经分配了最多的 $s$ 个块，并且写到了最后一个页。相应的对应 2 种垃圾回收触发条件，即总的空闲块不足或某个超级块占用的物理块达到规定的上限，任一个条件满足都会触发垃圾回收，这种是强制性的垃圾回收。

### 3.5 上电恢复

由于存放在 SRAM 中的信息，比如映射表掉电后会丢失，重新上电需要重建映射表等信息。对于基于超级块的混合映射方式，上电的主要任务是重建超级块映射表和空闲块表，在设计中，不需要将表存储到 NAND 上，而是通过以下 3 个步骤建立：(1)从 NAND 上读取坏块表，并更新到空闲块表中；(2)读取 NAND 备用区，通过组映射表确定数据块的块号，而读取空块会得到特殊的返回状态，据此更新空闲块表，此时得到最新的空闲块表；(3)由第(2)步中读取的组映射表，可以确定分配给每个超级块的物理块块号，按二分查找法读取最后分配的那个物理块，得到最后一个写入的页，通过这种方式就能得到每个超级块最后写入的物理块号和页号，从而得到最新的超级块映射表。

### 3.6 SRAM 空间的使用

SRAM 中存放的有超级块映射表、空块表、组映射表 cache 和组内页映射表 cache，数据读写缓冲区以及其他数据结构。在设计中，共设定 2 032 个超级块，NAND 物理块个数为 8 000，超级块映射表不超过 8 KB(一个条目 4 Byte)，空块表为 1 KB(1 bit 表示一个块的状态)，数据缓冲区为 64 KB，cache 结构占用 2 KB 左右，加上程序中其他数据变量，总共使用的空间小于 128 KB。

## 4 实验结果与分析

本文实验平台是一块固态硬盘开发板，该开发板容量为 32 GB，NAND 控制器具有 4 个通道，可并行处理 NAND 命令，SRAM 大小为 128 KB，具有额外的存储空间用于存放代码，采用高速的 SATA 接口和主机进行通信。通过主机应用程序 CrystalDiskMark 对硬盘做性能测试。

表 1 为顺序读写性能，和页映射相比，该性能有所下降，顺序读写时，主机逻辑地址按序递增，每隔一段时间需要建立一次 cache，这就带来性能的损耗，但由于 cache 的利用率非常高，所以下降有限；当 cache 个数为 1 和 20 时，顺序读写性能一样，这和预期是一致的，因为顺序读写时，新建立的一个 cache 会一直命中，直到主机逻辑地址空间超过这个 cache 的范围，则会建立下一个 cache，cache 的利用率是最高的。

**表 1 顺序读写性能** ($MB \cdot s^{-1}$)

| 映射方式 | 顺序读 | 顺序写 |
|---|---|---|
| 页映射 | 320 | 52 |
| 混合映射，cache 个数=1 | 190 | 28 |
| 混合映射，cache 个数=20 | 190 | 28 |

表 2 为 4 KB 随机读写性能(非 NCQ 命令)和 cache 个数的关系，可以看到 cache 个数越大，随机读写性能越高，但是增长很缓慢，这是因为 cache 个数越多，随机读写命中 cache 的概率也就越高，所以性能有所提升，但是由于超级块的个数为 2 032，内部分成 16 组，SRAM 空间为 128 KB，cache 个数相对还很小，因此性能提升缓慢。

**表 2 随机读写性能和 cache 个数的关系** ($MB \cdot s^{-1}$)

| cache 个数 | 4 KB 随机读 | 4 KB 随机写 |
|---|---|---|
| 1 | 5.4 | 2.0 |
| 5 | 5.4 | 2.2 |
| 20 | 6.6 | 2.2 |
| 40 | 8.0 | 2.4 |

(下转第 307 页)