

一种基于 NAND Flash 的多通道存储系统 编址方式研究与实现

赵 倩, 唐 磊

(西安微电子技术研究所, 陕西 西安 710054)

摘 要: 针对 NAND Flash 存储器的工作原理和物理特性, 大容量固态存储系统可以引入多通道并行技术来支持高速数据传输. 文章针对闪存阵列存储系统提出了一种新的编址方式—FVLA(Flash Virtual Layer Address), 该编址方式能很好地体现闪存系统的驱动顺序, 而且具有灵活的可配置性. 利用该编址方式设计出一种基于超级页的地址映射方法, 通过仿真证明了该映射方式能很好地执行并行操作并大幅降低了映射表所占用的存储空间.

关键词: NAND Flash; 多通道; FVLA; 超级页; 地址映射

中图分类号: TP334.5

文献标识码: A

文章编号: 1000—7180(2018)09—0032—05

Research and Implementation of Addressing Method Based on NAND Flash Multichannel Storage System

ZHAO Qian, TANG Lei

(Xi'an Microelectronics Technology Institute, Xi'an 710054, China)

Abstract: In view of the working principle and physical characteristics of NAND Flash devices, mass solid state memory system can introduce multi-channel parallel technology to support high-speed data transmission. This paper proposes a new addressing method for Flash Array Storage System—FVLA (Flash Virtual Layer Address), which can well reflect the operation sequence of flash memory system and can be configured flexibly. This paper uses the addressing mode to design an address mapping method based on superpage. The simulation results show that the mapping method can perform parallel operations and greatly reduce the memory space occupied by the mapping table.

Key words: NAND Flash; multichannel; FVLA; superpage; address mapping

1 引言

固态存储设备与传统机械磁盘相比, 具有体积小、存储密度高、随机存储和读写能力强、抗震抗摔、功耗低等特点^[1]. 基于这诸多优点, 大容量固态存储器已取代硬盘成为非常重要的存储设备, 目前基于 NAND Flash 存储器的闪存阵列存储设备已广泛应用于航空航天、国防、金融、消费电子产品等各行各业中.

随着“数据”在电子系统中重要性的不断提高, 对闪存阵列存储设备的容量和数据传输速率提出了

更高的要求. 由于 NAND Flash 存储器的单片存储容量有限以及慢写快读的特点, 各大厂商一方面将多个 NAND Flash 芯片组合封装在一个器件内以提高单片存储容量, 同时充分利用 NAND Flash 的操作特性开发各种并行操作机制以显著提高数据传输速率.

Flash 存储器由于其先擦后写等的操作限制, 在系统应用中需要采用闪存转换技术 FTL(Flash Translation Layer), 通过在上层文件系统与底层 Flash 存储器之间使用 FTL 来隐藏 Flash 存储器的操作限制, 进而使对以 Flash 存储器构成的闪存阵

列的 IO 读写操作像访问普通的磁盘一样. FTL 包括地址映射^[2] (address mapping)、垃圾回收^[3] (garbage collection)和磨损均衡^[4] (wear-leveling), 而垃圾回收和磨损均衡的设计都依赖于地址映射的设计. 本文在研究大容量固态存储系统多通道并行性的基础上, 针对闪存阵列提出一种新的编址方式—FVLA, 并用该编址方式设计出一种基于超级页的地址映射算法, 通过在自主搭建的仿真平台上进行 4×4 结构实验, 证明了该映射方式能很好的执行并行操作并极大地减少映射表所占用的存储空间.

2 NAND Flash 存储器介绍

目前, 对 NAND Flash 存储器的应用研究主要分为两个方面, 即存储系统体系结构中多级并行性方面的研究和地址管理软件算法的研究, 这里分别进行介绍.

2.1 并行性

N. Agrawal^[5] 和 L. M. Grupp^[6] 在其文章中提到, NAND Flash 存储器具有多个层次的组织结构, 即器件 (component)、芯片 (die)、分组 (plane)、块 (block) 和页 (page). 由于闪存存储系统多采用多通道结构, 故而闪存存储系统存在四个层次的并行性即通道级、器件级、芯片级和分組级. 这里主要分析一下通道级的并行性.

在 NAND Flash 存储器中, 读写操作均分成三个阶段, 即命令地址传输阶段、数据传输阶段、访问存储介质阶段^[7]. 一般情况下, 命令地址传输需要的时间为纳秒级, 但数据传输和访问存储介质需要的时间是微秒级. 事实上, 访问存储介质所需要的时间是数据传输的好几倍, 而且在访问存储介质期间不能对 NAND Flash 存储器进行任何操作, 这就是闪存存储系统并行性的基础. 闪存存储系统具有多个独立的通道, 每个通道有自己独立的通道控制器, 故而通道之间可以互不干扰地并发工作. 图 1 是通道间并行操作的示意图.

2.2 地址管理算法

基于 NAND Flash 存储器的闪存存储系统通过 FTL 访问底层存储器件, FTL 包括地址映射 (address mapping)、垃圾回收 (garbage collection) 和磨损均衡 (wear-leveling), 而垃圾回收和磨损均衡的设计都依赖于地址映射的设计. FTL 地址映射算法主要包括页级地址映射、块级地址映射和混合级地址映射.

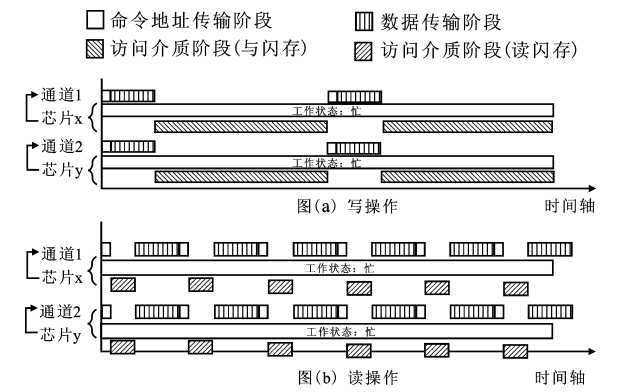


图 1 通道之间的并行示意图

页级地址映射算法是一种细粒度的地址映射算法, 逻辑页的数据可以存放在任何位置的物理页中, 每一个逻辑页有一个映射关系. 如果某文件包含 n 个逻辑页单元, 按照页级地址映射算法, 其对应的映射表应该有 n 行, 如图 2 所示. 通常页级映射表占用的空间较大, 但其性能最好. 块级映射算法是一种粗粒度的映射算法, 它以物理块为基本的映射单元, 逻辑页的数据只能存放在某个物理块中特定偏移的物理页中, 故而它的映射表占用的空间少, 但其性能较差. 混合映射算法将物理块分为数据块和日志块, 数据块采用块级地址映射, 日志块采用页级地址映射, 其性能介于页级地址映射和块级地址映射之间. 目前, 在追求高性能的闪存系统中, 大部分设计依然采用基于页级映射的映射算法.

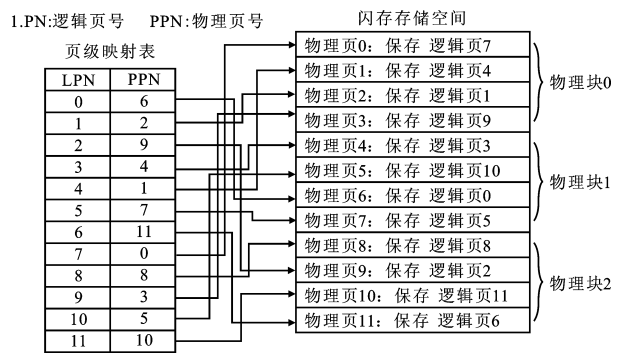


图 2 页级映射结构示意图

3 系统整体设计及分析

大容量闪存系统通常采用多通道并行的体系结构, $p \times q$ 描述的是多通道控制器的物理结构, p 表示的是通道控制器数目, q 表示每个通道所连接的 NAND Flash 存储芯片个数. 每一个通道有一个独立的闪存控制器, 它包含一套完整的 NAND 接口 IO、NAND 命令产生器及软件接口、DMA 控制器和纠错单元, 同一通

道下的各个存储芯片拥有独立的片选信号线和独立的工作状态信号线,所有通道公用一组数据总线.在这种结构下,每个存储芯片可以独立工作,它们通过分时复用来争用共有的数据总线进行数据传输.

本系统采用 4×4 结构,如图 3 所示,本系统拥有 4 个通道,每个通道包含 4 片独立的 NAND Flash 存储器件,每个存储器件包含一个芯片,每个芯片包含一个分组,每个分组包含 1 024 个物理块,每个物理块包含 128 个物理页,每个物理页包含 2K 字节的数据区和 64 字节的空余区.为了提高数据的传输速率,应该让各通道和各器件更多地处于工作状态,故而这里采用通道级和器件级两级并行.由于同一行上的 4 块器件使用相同的片选信号和控制信号,而使用不同的数据通道,故而实现了位扩展,这 4 片并行的 Flash 元件相同地址的页可以组成一个大小为 8 K 字节超级页,可同时进行数据传输.同一列上的四个器件公用相同的控制信号和数据通道,而使用不同的片选信号,故而可以分时复用数据信号来加大数据传输速度.

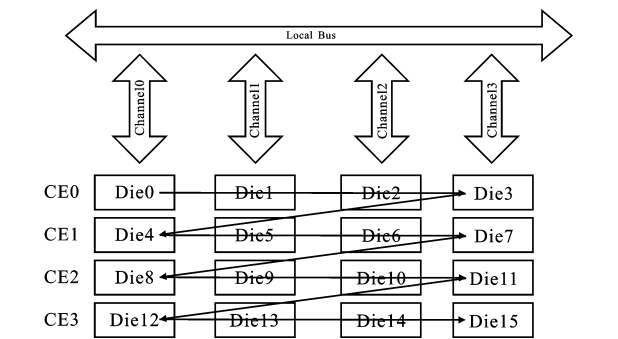


图 3 闪存阵列 4×4 多通道控制器结构

在此闪存阵列底层系统架构基础上,本文提出一种新的编址方式即闪存虚拟层编址—FVLA(Flash Virtual Layer Address),闪存虚拟层(FVL)位于 FTL 和 Flash 驱动层之间.它抽象了闪存阵列的底层结构,使 FTL 层看到的是线性的存储单元,而这种线性单元正是大规模闪存系统在驱动过程中的实际顺序. FVLA 是一个 32 位的逻辑地址,其每一位都有自己代表的含义,具体说明如表 1 所示.

表 1 FVLA 组成示意图

FVLA(32 位)	10 位	10 位	8 位	2 位	2 位
reserved	1 024 块	128 页	4 CE	4 通道	

FVLA 地址可以解析出确定一个物理页位置所需要的各个参数.根据通道和片选信号可以确定具体的闪存器件和芯片,根据块号和页号可以进一步定位到具体的物理页,故而这个地址是可以唯一

地表示闪存阵列里的任意一个物理页的.当通道数、片选数、物理块和物理页的数目变化时,FVLA 可以根据具体的需要进行动态调整.FTL 层看到的是线性空间,每一个 FVLA 地址在 4×4 结构的闪存阵列里的具体位置示例图如图 4 所示,当 $FVLA=0$ 时,表示该页所在的位置是 channel0,CE0,blk0,page0;当 $FVLA=1$ 时,表示该页所在的位置是 channel1,CE0,blk0,page0... 同一行的相同物理位置的 4 页构成一个超级页(superpage),如图所示,假如对 superpage0 进行操作之后,然后就要利用 CE 级并行操作 superpage1,接下来操作 superpage2,superpage3... 采用二维“Z”字型驱动模型来获得最大并行化的闪存访问带宽.

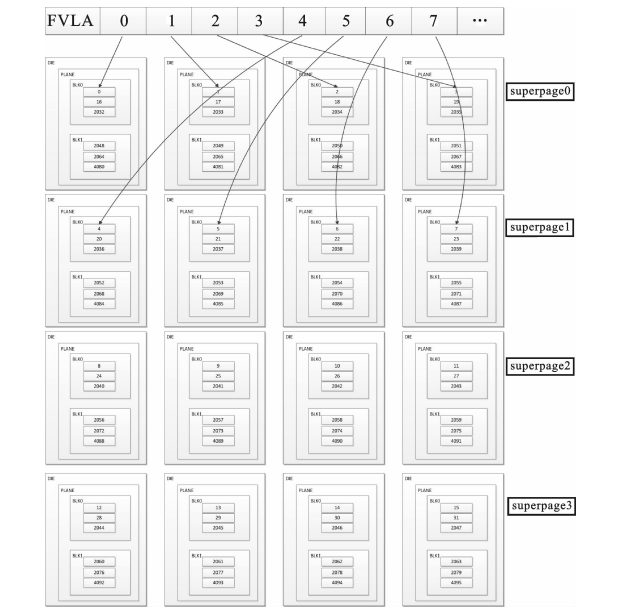


图 4 FVLA 在 4×4 闪存阵列中的位置示例图

由于闪存系统每次处理的数据都是以超级页为基本单位进行的,故而地址映射机制也应该采用基于超级页的地址映射方式.假如写入的文件大小为 $n\times 8\text{ k}$ 字节,需要 n 个超级页来存储数据,对应的一种地址映射表如表 2 所示,在闪存系统最开始使用的时候就可以采用这种映射方式,随着对文件的删除等操作,映射表会动态更新.

表 2 4×4 闪存阵列中 FVLA 地址映射表示例表

LPN	PPN(FVLA)
0	0
1	4
2	8
...	...
n	$4n$

4 仿真验证

在 Ubuntu 操作系统下,利用 CodeBlocks 集成开发环境建立的 NAND Flash 仿真模型对 FVLA 进行系统验证,利用 C 语言提供的库函数,自主编写了一些结构和函数,实现有关闪存存储的核心部分,包括:以页为单位进行数据读写,以块为单位进行数据擦除,数据异地更新等.这部分主要完成对 NAND Flash 操作特性的模拟,弱化 NAND Flash 内部的逻辑时序,建立具有多通道的闪存阵列仿真模型,下面给出对闪存阵列的配置描述:

```
uint8_tNandPre_SetSysParam()
{
    gHW_NandFlashParam. mSectorsPerPage= SECTOR_
CNT_PER_PAGE;
    gHW_NandFlashParam. mBytesPerPage= gHW_Nand-
FlashParam. mSectorsPerPage
    * BYTE_PER_MDSECTOR;
    gHW_NandFlashParam. mBytesPerSpare = BYTE _
CNT_PER_SPARE;
    gHW_NandFlashParam. mPagesPerBlock = PAGE _
CNT_PER_BLK;
    gHW_NandFlashParam. mPlanesPerChip = PLANE _
CNT_PER_DIE;
    gHW_NandFlashParam. mBlocksPerPlane= BLK_CNT
_PER_PLANE;
    gHW_NandFlashParam. mBlocksPerChip= BLK_CNT
_PER_D;
    gHW_NandFlashParam. mSizePerPage=
    gHW_NandFlashParam. mBytesPerPage +gHW_Nand-
FlashParam. mBytesPerSpare;
    gHW _ LayoutParam. mChannelsPerArray = CHL
_CNT;
    gHW_LayoutParam. mChipsPerChannel = CE_CNT_
PER_CHL;
    return 0;
}
```

该配置给出了闪存系统的通道数,片选数目,每个器件包含的芯片数目,每个芯片包含的分组数,每个分组包含的物理块数,每个块的物理页数,每一页的数据区长度和冗余区长度,这些就能非常完整地描述基于 NAND Flash 的闪存系统参数,并且仿真中这些参数可以根据需要自主配置,增加灵活性.

图 5 是 FVLA 地址映射算法的读写执行流程图. 首先需要在内存里开辟一块存储空间 mapT-ableSaddr 专门用来存放地址映射表,初始化地址映

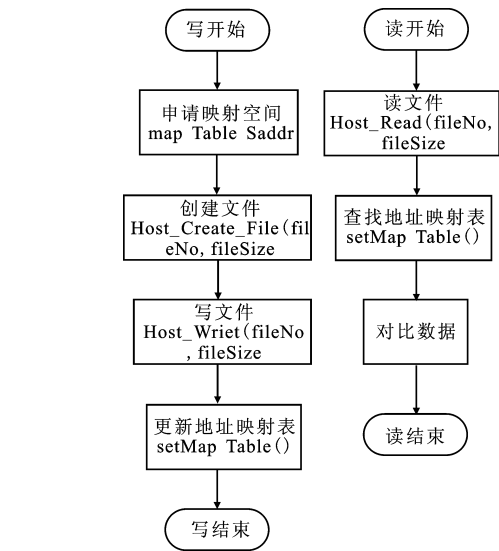


图 5 FVLA 地址映射算法的读写执行流程图

射表内容均为空闲可用. 当有文件写入闪存阵列时,就需要更新地址映射表 setMapTable(),这时地址映射表里就会保存 FVLA 地址,实际应用中驱动层得到 FVLA 地址后,就会去配置地址寄存器,以将数据写入物理的闪存阵列,因为仿真系统中没有地址寄存器的概念,故驱动层通过解析 FVLA 地址,将数据写入对应的超级物理页中;当需要读取文件时就会从地址映射表里读取文件的 FVLA 位置信息即 searchMapTable(),根据 FVLA 地址,驱动层就去闪存阵列的相应位置以超级页为单位读出文件,这里写入的数据量可以自主进行控制,可以选择全零或者全一,亦或递增递减数列等,在已知写入的具体数据时,就可以与根据超级页映射表读出来的数据进行比较,以验证该映射算法的正确性;当有文件删除操作时,也需要更新地址映射表,释放的逻辑页地址空间和对应的物理页地址空间均设置为空闲可用,以便为下一次的文件写入留出空间. 这里首先介绍几个文件操作的函数:Host_Create_File(fileNo,fileSize)这个函数是创建一个文件号为 fileNo,文件大小为 fileSize 的文件;Host_Write(fileNo,fileSize),此函数是向文件号为 fileNo 的文件写入大小为 fileSize 的数据;Host_Read(fileNo,fileSize)是读取文件号为 fileNo,文件大小为 fileSize 的文件. 以下是在自主搭建的仿真平台上采用 4×4 结构仿真的具体例子:

```
Host_Create_File(0,3*8); //创建一个文件号为 0,
文件大小为 3*8 k 的数据
Host_Write(0,3*8); //写入文件 0
setMapTable(lbaBase, pVal) //更新映射表,把 FVLA
```

地址写入映射表

```
{
    pMapTable=mapTableSaddr + sizeof(MapTABLE) *
lbaBase;
    pMapTable->sPage=pVal->sPage;
}
```

仿真系统实时调试可以发现,从初始化闪存模型到文件 0 第一次写入后,映射表发生变化,有三个超级页的映射表从最开始的空闲可用变成了 0,4,8,成功更新了映射表,文件实际写入的 FVLA 地址是 0-11 这 12 个小页也就是 3 个超级页. Host_Read(0,3*8); //读出文件 0

```
searchMapTable(lbase) //查找映射表
{pMapTable=(PMapTable)(mapTableSaddr
+sizeof (MAP_TABLE) * lbaBase);
returnpMapTable->sPage; //读出 FVLA
}
```

驱动层拿到 FVLA=0,FVLA=4,FVLA=8 地址后会将 FVLA 地址从 0-11 这三个 superpage 的数据都从闪存阵列中读出来,通过将读出的数据与写入的数据比较得出的结论是读出的数据和写入的数据完全一致,该映射方法是完全可行的.

5 结束语

本文在研究了基于 NAND Flash 的闪存系统多级并行性及地址映射算法的基础上,提出一种新的编址方式—FVLA,并用该编址方式设计了一种基于超级页的地址映射方法. 通过在自主搭建的 Flash 仿真平台上进行 4×4 结构的实验,证明了该算法的可行性,其能够有效地进行并行操作并极大的减少映射表所占用的存储空间.

参考文献:

[1] Yan Hua, Yao Qian. An efficient file-aware garbage

collection algorithm for NAND Flash-based consumer [J]. IEEE Transactions on Consumer Electronics, 2014,60(4):623-627.

[2] Gupta, Kim Y, Urgaonkar B. DFTL: a flash translation layer employing demand-based selective of page-level address mapping[C]//Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systemd(AS-PLOS09). Washington, DC, 2009: 9-22.

[3] Shin J, Xia Z, Xu N, et al. FTL design exploration in reconfigurable high-performance SSD for server applications [C]//Proceedings of the 23rd International Conference on Supercomputing (ICS'09). Yorktown, NY, 2009: 102-114 .

[4] Park C, Cheon W, Kang J, et al. A reconfigurable FTL architecture for NAND flash-based applications [J]. ACM Transactions on Embedded Computing Systems,2008, 7(2): 60-71.

[5] Agrawal N, Prabhakaran V, Wobber T, et al. Design Tradeoffs for SSD Performance[C]//Proceedings of the 2008 USENIX Annual Technical Conference (USENIX'08). San Diego, CA, 2008: 89-101.

[6] Grupp L M, Caulfield A M, Coburn J, et al. Characterizing flash memory: anomalies, observations, and applications[C]//Proceedings of the 42nd International Symposium on Microarchitecture(MICRO09). New York, NY, 2009: 20-32.

[7] 胡洋. 高性能固态硬盘的多级并行性及算法研究[D]. 武汉: 华中科技大学,2012.

作者简介:

赵 倩 女,(1993-),硕士研究生. 研究方向为嵌入式系统结构. E-mail:18792580584@163. com.

唐 磊 男,(1973-),研究员,硕士生导师. 研究方向为空间计算机系统结构、微系统.