[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 12/02 (2006.01)
G11C 16/06 (2006.01)

[12] 发明专利申请公布说明书

[21] 申请号 200610142359.8

[43] 公开日 2007 年 3 月 7 日

[11] 公开号 CN 1924831A

[22] 申请日 2002.1.7
[21] 申请号 200610142359.8
分案原申请号 02803882.7
[30] 优先权
[32] 2001. 1.19 [33] US [31] 09/766,436
[71] 申请人 三因迪斯克公司
地址 美国加利福尼亚州
[72] 发明人 凯文·M·康利
[74] 专利代理机构 中国国际贸易促进委员会专利商标事务所
代理人 康建峰

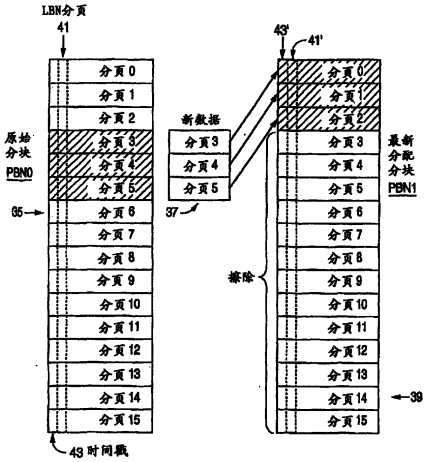
权利要求书 2 页 说明书 16 页 附图 9 页

[54] 发明名称

非易失性存储器系统及操作非易失性存储器系统的方法

[57] 摘要

通过在同一或另一分块的未用分页中对新数据进行编程，更新非易失性存储分块的部分分页中的数据。 为了防止必须将未变数据分页拷贝到新分块中，或者将标志编程到替换数据分页中，用与新数据所替换的数据分页相同的逻辑地址标识新数据分页，并且添加时间戳来表示写入每个分页的时间。当读取数据时，使用最新数据分页，并且忽略较旧替换数据分页。 该投术还应用于包括来自若干不同存储阵列单元的各自一个分块的元分块，其中，将所有分页更新引导到其中一个单元内的单个未用分块。



1. 一种操作非易失性存储器系统的方法，其中非易失性存储器系统具有存储元件阵列，存储元件阵列组织为至少两个子阵列，其中，各个子阵列又分为多个非重叠存储元件分块，而各个分块包含可以一起擦除的最小存储元件组，所述方法包括:

链接来自所述至少两个子阵列的各个子阵列的至少一个分块作为元分块的组成分块;

同时擦除一起作为一个单元的元分块的组成分块；以及

通过仅在指定的子阵列的一个中将更替数据编程到至少一个组成分块而不管正被更新的数据存储在哪个子阵列中，更新存储在元分块的至少一个组成分块内的数据。

2. 如权利要求 1 所述的方法，其中存储器系统的各个分块分为多个存储元件的分页，其中，各个分页包含可以一起编程的存储元件，并且其中更新数据包括更新少于元分块的所述至少一个组成分块内的所有分页中的数据。

3. 如权利要求 1 所述的方法，其中存储更替数据包括:

用对存储器系统的相同逻辑地址标识更新的数据和正被更新的数据；以及

通过利用将原始和更替数据编程到存储器的相对时间，区分更新的数据和正被更新的数据。

4. 一种非易失性存储器系统，包括:

至少两个非易失性存储元件的子阵列，各个子阵列又分为多个存储元件的分块，其中各个分块的存储单元一起擦除;

不同子阵列内的分块的链接，作为至少一个元分块的组成分块;

编程机构，通过仅在指定的子阵列的一个中将更替数据写入至少一个组成分块而不管正被更新的数据存储在哪个子阵列中，更新存储在所述至少一个元分块的至少一个组成分块内的数据;

寻址机构，对具有相同地址的正被更新的数据和更替数据进行逻辑

寻址；以及

读取机构，区分具有相同逻辑地址的正被更新的数据和更替数据。

5. 如权利要求 4 所述的存储器系统，其中

编程机构根据写入数据的相对时间指示，对正被更新的数据和更替数据进行写入；以及

读取机构至少部分根据所述相对时间指示区分正被更新的数据和更替数据。

6. 如权利要求 5 所述的存储器系统，还包括擦除机构，用于同时擦除所述至少一个元分块的所有组成分块内的存储单元。

7. 如权利要求 4 所述的存储器系统，还包括擦除机构，用于同时擦除所述至少一个元分块的所有组成分块内的存储单元。

8. 如权利要求 4 所述的存储器系统，其中各个分块独立分为多个存储单元的分页，并且编程机构对一个分页内的存储单元一起编程。

非易失性存储器系统及操作非易失性存储器系统的方法

本发明专利申请为 2002 年 1 月 7 日提出的发明名称为“非易失性存储器中的部分分块数据编程和读取操作”、申请号为 02803882.7 的发明专利申请的分案申请。

技术领域

本发明涉及半导体非易失性数据存储系统架构及其操作方法的领域，并且应用于具有对基于快闪(flash)电可擦除可编程只读存储器(EEPROM)的数据存储系统。

背景技术

快闪 EEPROM 装置的一般应用是作为电子装置的大容量数据存储子系统。这些子系统通常被实现为可以插入到多个主机系统的可移动式存储卡或者主机系统内的非可移动式嵌入存储器。在这两种实现中，子系统包括一个或多个快闪装置，并且经常还包括一个子系统控制器。

快闪 EEPROM 装置由一个或多个晶体管分区(cell)阵列组成，其中，每个分区能够非易失性地存储一个或多个数据位。因此，闪存不需要电源来保持在其中编程的数据。然而，一旦编程，在分区可以用新数据值进行重新编程之前，它必须被擦除。这些分区阵列分成多组，以支持高效实现读取、编程和擦除功能。用于大容量存储的典型闪存架构将大的分区组编排为可擦除分块(block)，其中，分块包含一次可擦除的最小数目的分区(擦除单元)。

在一种商业形式中，各分块包含足够的分区来存储一个扇区的用户数据以及与用户数据和/或存储它的分块相关的一些开销数据。在一类这种存储系统中，包括在一个扇区中的用户数据量为标准 512 字节，但是也可以具有某种其它尺寸。由于为了使各个分区分块可单独擦除而需

要在集成电路芯片上对这些分块进行相互隔离，因此另一类闪存使分块显著增大，从而减小这种隔离所需的空间。但是由于还希望以小得多的扇区处理用户数据，因此经常还将每个大分块分为可单独寻址的分页，它是用于对用户数据进行读取和编程的基本单元(编程和/或读取单元)。每个分页通常存储一个扇区的用户数据，但是每个分页也可以存储部分扇区或多个扇区。在此所用的"扇区"是指作为单元传输到主机和从主机传输的用户数据量。

大分块系统中的子系统控制器执行多种功能，包括存储子系统从主机接收的逻辑地址(LBA)与存储分区阵列内的物理分块号(PBN)和分页地址之间的转换。该转换经常涉及使用逻辑分块号(LBN)和逻辑分页的中间术语。控制器还通过它经由接口总线发给闪存装置的一系列命令，管理低层快闪电路操作。控制器所执行的另一功能是通过各种手段如通过使用纠错码(ECC)维护存储到子系统的数据的完整性。

在理想情况下，分块的所有分页中的数据经常通过将更新数据写入到未被分配的擦除分块内的分页来一起更新，并且使用新地址更新逻辑-到-物理分块号表。然后，可以擦除原始分块。然而，更典型的是，必须更新给定分块内的部分分页中的数据。存储在给定分块的其余分页中的数据保持不变。在每个分块所存储数据扇区数越高的系统中，这一事件的概率就越高。目前用来完成这种部分分块更新的一种技术是将所要更新的分页数据写入到未用擦除分块内具有对应编号的分页中，然后将未变分页从原始分块拷贝到新分块的分页中。然后，可以擦除原始分块，并且将其加到以后可以对数据进行编程的未用分块库存中。另一种技术类似地将更新分页写入到新分块，但是通过改变原始分块内正被更新的分页的标志来表示它们包含过时数据，而不需要将其它数据分页拷贝到新分块中。然后，当读取数据时，组合从新分块的分页中读取的更新数据与从原始分块内未标为过时的分页中读取的未变数据。

发明内容

根据本发明的一个方面，提供一种操作非易失性存储器系统的方

法，其中非易失性存储器系统具有存储元件阵列，存储元件阵列组织为至少两个子阵列，其中，各个子阵列又分为多个非重叠存储元件分块，而各个分块包含可以一起擦除的最小存储元件组，所述方法包括：链接来自所述至少两个子阵列的各个子阵列的至少一个分块作为元分块的组成分块；同时擦除一起作为一个单元的元分块的组成分块；以及通过仅在指定的子阵列的一个中将更替数据编程到至少一个组成分块而不管正被更新的数据存储在哪个子阵列中，更新存储在元分块的至少一个组成分块内的数据。

根据本发明的另一方面，提供一种非易失性存储器系统，包括：至少两个非易失性存储元件的子阵列，各个子阵列又分为多个存储元件的分块，其中各个分块的存储单元一起擦除；不同子阵列内的分块的链接，作为至少一个元分块的组成分块；编程机构，通过仅在指定的子阵列的一个中将更替数据写入至少一个组成分块而不管正被更新的数据存储在哪个子阵列中，更新存储在所述至少一个元分块的至少一个组成分块内的数据；寻址机构，对具有相同地址的正被更新的数据和更替数据进行逻辑寻址；以及读取机构，区分具有相同逻辑地址的正被更新的数据和更替数据。

根据本发明原理的一方面，简短且概括地说，当更新分块内部分分页的数据时，避免将未变数据从原始分块拷贝到新分块以及更新原始分块内的标志的需要。这通过采用共同逻辑地址维护替换数据分页和更新数据分页来完成。然后，数据的原始和更新分页通过对它们进行编程的相对次序来区分。在读取期间，组合存储在具有相同逻辑地址的分页中的最新数据与未变数据分页，而忽略更新分页原始版本中的数据。更新数据可以写入到与原始数据不同的分块内的分页或者可以在同一分块内获得的未用分页。在一个特定实施例中，对各个数据分页存储某种形式的时间戳，从而允许确定具有相同逻辑地址的分页的写入相对次序。在另一个特定实现中，在分块内以特定次序对分页进行编程的系统中，对各个数据分块存储某种形式的时间戳，并且通过分页在分块内的物理位置来确定分块内分页的最新副本。

这些技术既不需要将未变数据从原始分块拷贝到新分块，又不需要改变原始分块内其数据已被更新的分页中的标志或其它数据。由于不需要改变替换分页中的标志或其它数据，因此消除由于这种写入操作而干扰同一分块的相邻分页中的先前写入数据的潜在可能性。另外，避免额外编程操作所带来的性能恶化。

可以结合上述技术使用的另一操作特征记录各个存储分区分块内各个数据分页的逻辑偏移，从而不需要使用与替换数据相同的物理分页偏移来存储更新数据。这就允许更高效使用新分块的分页，并且甚至允许将更新数据存储在与替换数据相同的分块内的任何擦除分页中。

本发明的另一原理方面将位于不同存储阵列单元(也称作“子阵列”)中的两个或更多分块分组在一起，以作为单个操作的一部分一起进行编程和读取。这种多分块组在此称作“元分块(metablock)”。其组成分块可以均位于单存储器集成电路芯片上，或者在使用多个这种芯片的系统中位于两个或更多不同芯片上。当更新这些分块中的一个分块的部分分页的数据时，通常需要使用同一单元中的另一分块。实际上，可以对元分块中的每个分块单独采用上述或其它技术。因此，当更新元分块中的多个分块的分页内的数据时，需要使用多个另外分块内的分页。例如，如果元分块由四个不同存储单元的四个分块形成，则可能将要另外使用最大四个分块来存储原始分块的更新分页，其中每个单元均有一个另外分块。对于原始元分块的每个分块，可能都需要各自单元中的一个更新分块。另外，根据本发明，来自元分块中多个分块的分页的更新数据可以仅存储在其中一个单元内公共分块的分页中。这就大大减少存储更新数据所需的未用擦除分块，从而更高效利用可用来存储数据的存储分区分块。当存储系统频繁更新来自元分块的单个分页时，该技术尤其有用。

本发明的另外方面、特性和优点包括在下面对示例性实施例的描述中，其中，该描述应结合附图一起阅读。

附图说明

图 1 是具有存储控制逻辑、数据和地址寄存器的典型现有技术快闪

EEPROM 存储阵列的方框图；

图 2 示出具有系统控制器的采用多个图 1 的存储器的架构；

图 3 是示出图 2 的存储系统的典型拷贝操作的时序图；

图 4 示出更新多页分块的部分分页中的数据的现有过程；

图 5A 和 5B 分别是图 4 的原始和新分块的逻辑与物理分块地址对应表；

图 6 示出更新多页分块的部分分页中的数据的另一现有过程；

图 7A 和 7B 分别是图 6 的原始和新分块的逻辑与物理分页地址对应表；

图 8 示出更新多页分块的部分分页中的数据的改进过程的例子；

图 9 是图 8 的新分块的逻辑与物理分页号对应表；

图 10 提供图 8 所示的分页内的数据布局的一个例子；

图 11 示出图 8 的例子的进一步发展；

图 12 是图 11 的新分块的逻辑与物理分页号对应表；

图 13 示出在图 11 的分块中读取更新数据的一种方式；

图 14 是将数据编程到如图 8 和 9 所示组织的存储系统中的过程的流程图；

图 15 示出现有多单元存储器，其中，来自各个单元的分块链接在一起形成元分块；以及

图 16 示出当更新数据量远远小于元分块的数据存储容量时在图 12 的多单元存储器中更新元分块数据的改进方法。

具体实施方式

【现有大分块管理技术的描述】

图 1 示出典型闪存装置内部架构。主要部件包括：输入/输出(I/O)总线 411 和控制信号 412，用来通过接口连接到外部控制器；存储控制电路 450，用来通过用于命令、地址和状态信号的寄存器控制内部存储操作。还包括一个或多个快闪 EEPROM 分区阵列 400，其中，每个阵列均具有其自己的行解码器(XDEC)401 和列解码器(YDEC)402，一组感测

放大器和程序控制电路(SA/PROG)454 和数据寄存器 404。目前，存储分区通常包括一个或多个导电浮动栅作为存储元件，但是也可以改为使用其它长期电子电荷存储元件。存储分区阵列(memory cell array)可以采用为每个存储元件定义的两个电荷电平工作，从而使用每个元件存储一个数据位。可选地，可以为每个存储元件定义两个以上的存储状态，在这种情况下，在每个元件中存储多个数据位。

如果需要，提供多个阵列 400 以及相关 X 解码器、Y 解码器、程序/验证电路、数据寄存器等，例如美国专利 5,890,192 所述，该专利发布于 1999 年 3 月 30 日，并且受让给 Sandisk 公司，即本申请的受让人，在此将其引作参考。相关存储系统特性在 Kevin Conley 等人于 2000 年 2 月 17 日提交的共同未决专利申请 09/505,555 序列号中有描述，在此将其引作参考。

外部接口 I/O 总线 411 和控制信号 412 可以包括如下方面:

CS-芯片选择	用来激活闪存接口
RS-读取选通	用来表示 I/O 总线正用来从存储阵列传输数据
WS-写入选通	用来表示 I/O 总线正用来将数据传输到存储阵列
AS-地址选通	表示 I/O 总线正用来传输地址信息
AD[7:0]-地址/数据总线	该 I/O 总线用来在控制器与存储控制 450 的闪存命令、地址和数据寄存器之间传输数据

该接口是仅作为例子给出的，并且可以使用其它信号结构来提供相同的功能。图 1 仅示出一个具有相关组件的闪存阵列 400，但是多个这种阵列可以存在于单个闪存芯片上，这些阵列共用公共接口和存储控制电路，但具有单独的 XDEC、YDEC、SA/PROG 和 DATA REG(数据寄存器)电路，从而允许并行读取和编程操作。

数据凭藉数据寄存器与 I/O 总线 AD[7:0]411 的连接，通过数据寄存器 404 从存储阵列传输到外部控制器。数据寄存器 404 还连接到感测放大器/编程电路 454。连接到各感测放大器/编程电路元件的数据寄存器元件数可以依赖于存储在存储分区的每个存储元件中的位数，其中，快闪 EEPROM 分区均包含一个或多个浮动栅作为存储元件。如果存储分区在多状态模式下工作，则每个存储元件可以存储多位如 2 或 4。可选地，存储分区可以在二进制模式下工作，从而每个存储元件存储一个数据位。

行解码器 401 对阵列 400 的行地址进行解码，从而选择所要访问的物理分页。行解码器 401 通过内部行地址线 419 从存储控制逻辑 450 接收行地址。列解码器 402 通过内部列地址线 429 从存储控制逻辑 450 接收列地址。

图 2 示出典型非易失性数据存储系统的架构，在这种情况下，采用闪存分区作为存储介质。在一种形式中，该系统封装在可移动卡内，其中，该卡具有沿着一侧延展的电气连接器，以当插入到主机插座中时提供主机接口。可选地，图 2 的系统可以采用永久性安装的嵌入电路或其它形式，嵌入到主机系统中。系统采用单个控制器 301 来执行高层主机和存储控制功能。闪存介质由一个或多个闪存装置组成，其中，每个这种装置经常是在其自己的集成电路芯片上形成的。系统控制器和闪存通过总线 302 连接，其中，总线 302 允许控制器 301 载入命令、地址，并且将数据传输到闪存阵列和从其传输数据。控制器 301 与主机系统(未示出)通过接口连接，通过它，将用户数据传输到闪存阵列和从其传输用户数据。在图 2 的系统包括在卡中的情况下，主机接口包括卡和主机设备上的匹配插头和插座组合。

控制器 301 从主机接收命令，以读取或写入以特定逻辑地址开始的一个或多个扇区的用户数据。该地址可以或者可以不与存储分区的物理分块的边界对齐。

如上所述，在具有分为多页的大容量存储分区分块的一些现有技术系统中，来自未被更新分块的数据需要从原始分块拷贝到新分块，该新

分块还包含正由主机写入的新更改数据。该技术如图 4 所示，其中，包括存储器的大量分块中的两个分块。图中示出一个分块 11(PBN0)分为 8 个分页，其中每个分页存储一个扇区的用户数据。包含在各个分页内的开销数据字段包括字段 13，它包含分块 11 的 LBN。逻辑分块内逻辑分页的次序相对于物理分块内的对应物理分页是固定的。从未用擦除分块库存中选择类似构造的第二分块 15(PBN1)。原始分块 11 的分页 3-5 内的数据正被新分块 15 的三个分页更新。将新数据写入到新分块 15 的对应分页 3-5 中，并且将来自分块 11 的分页 0-2、6 和 7 的用户数据拷贝到新分块 15 的对应分页中。新分块 15 的所有分页最好采用单一编程操作顺序来编码。在对分块 15 编程之后，可以擦除原始分块 11，并且将其放入库存以作以后使用。分块 11 和 15 之间的数据拷贝涉及从原始分块中的一个或多个分页中读取数据，然后将相同数据顺序编程到最新分配分块内的分页，这将大大降低写入性能和存储系统的可用寿命。

参照图 5A 和 5B，部分表示出参照图 4 所述的数据更新之前(图 5A)和之后(图 5B)逻辑分块到原始和新物理分块 11 和 15 的映射。在本例中，在数据更新之前，原始分块 11 将 LBN 0 的分页 0-7 存储到 PBN0 的对应分页 0-7 中。在数据更新之后，新分块 15 将 LBN0 的分页 0-7 存储在 PBN1 的对应分页 0-7 中。接收从 LBN0 读取数据的请求因而引导到物理分块 15，而不是物理分块 11。在典型控制器操作中，采用图 5A 和 5B 所示形式的表根据从物理分页读取的 LBN 字段 13 以及当读取数据字段 13 时所寻址的 PBN 的知识来构建。该表通常存储在控制器的易失性存储器中以便于访问，但是在任何时候典型地仅存储整个系统的完整表的一部分。该表的一部分通常就是在涉及包括在表部分中的分块的读取或编程操作之前形成的。

在其它现有技术系统中，对分页中的用户数据记录标志，并且这些标志用来表示原始分块内正被最新写入数据替换的数据分页无效。只有新数据才写入到最新分配分块。因此，写入操作没有涉及到但包含在与替换数据相同的物理分块中的那些分页数据不需要拷贝到新分块中。该操作如图 6 所示，其中，原始分块 21(PBN0)内的数据分页 3-5 再次正被

更新。数据 23 的更新分页 3-5 写入到新分块 25 的对应分页中。作为同一操作的一部分，在分页 3-5 中各自写入旧/新标志 27，以表示那些分页的数据旧，而其余分页 0-2、6 和 7 的标志 27 保持设为“新”。类似地，将新 PBN1 各自写入到分块 21 内分页 3-5 的另一开销数据字段中，以表示更新数据位于何处。LBN 和分页存储在各个物理分页内的字段 31 中。

图 7A 和 7B 是数据更新完成之前(图 7A)和之后(图 7B)的数据 LBN/分页和 PBN/分页之间的对应表。LBN 的未变分页 0-2、6 和 7 保持映射到 PBN0 中，而更新分页 3-5 示出为驻留在 PBN1 中。图 7B 的表 7B 由存储控制器通过在数据更新之后读取分块 PBN0 内分页的开销数据字段 27、29 和 31 来构建。由于在原始分块 PBN0 的分页 3-5 中均将标志 27 设为“旧”，因此该分块将不再出现在这些分页的表中。而是新分块号 PBN1 出现，它是从更新分页的开销字段 29’读取的。当从 LBN0 读取数据时，读取存储在图 7B 的右列所列出的分页中的用户数据，然后以所示次序进行组合，以传输到主机。

各种标志典型地位于与其它相关开销数据如 LBN 和 ECC 相同的物理分页中。因此，为在数据已被替换的分页中对旧/新标志 27 等进行编程需要分页支持多编程循环。也就是，存储阵列必须具有可以在擦除之间的至少两个阶段对其分页进行编程的能力。此外，分块必须支持当分块内具有更高偏移或地址的其它分页已经被编程时对分页进行编程的能力。然而，一些闪存的限制由于指定只能以物理顺序方式对分块中的分页进行编程而不能使用这种标志。而且，分页支持有限数目的编程循环，并且在某些情况下不允许对已编程分页进行附加编程。

需要一种在不从现有分块拷贝未变数据或者不将标志编程到先前已编程的分页的情况下可以写入对存储在现有分块中的数据进行部分替换的数据的机制。

【本发明示例性实施例的描述】

存在很多不同类型的快闪 EEPROM，其中每种快闪 EEPROM 均具有其自己的限制，要操作在小量集成电路区域上形成的高性能存储系

统，则必须绕开这些限制。某些不支持将任何数据写入到已经被编程的分页中，从而如上所述在包含替换数据的分页中更新标志是不可能的。其它允许写入这种标志，但是在数据正被替换的分页中这样做会干扰同一分块内保持最新的其它分页数据。

已发现存在问题的一种示例性存储系统是“与非(NAND)”类型，其中，存储分区列作为位线和公共电势之间的串联电路串来形成。每个字线跨越由每个这种串中的一个分区形成的一行存储分区。当在多状态模式下工作以在每个这种分区中存储多个数据位时，这种存储器尤其易受这种存储状态干扰的影响。该操作将存储分区晶体管阈值电压范围的可用窗口分为窄小非重叠电压电平范围，其中每个范围随着电平数以及相应的存储在每个分区中的位数的增加而变窄。例如，如果使用四个阈值范围，则在每个分区的存储元件中存储两位数据。并且由于这四个阈值电压范围均必然较小，因此分区状态受到对同一分块内的其它分区进行编程的干扰的机会随着多状态操作而加大。在这种情况下，不能容忍如参照图 6、7A 和 7B 所述的旧/新或其它标志写入。

上面参照图 4-7B 所述的各种现有存储管理技术的一个公共特性是在系统内将逻辑分块号(LBN)和分页偏移映射到最多两个物理分块号(PBN)。一个分块是原始分块，而另一个包含更新分页数据。数据写入到分块中与其逻辑地址(LBA)的低位相对应的分页位置。该映射在各种存储系统中是典型的。在下述技术中，包含更新数据的分页也分配有与其数据已被替换的分页相同的 LBN 和分页偏移。然而不是将包含原始数据的分页标记为被替换，而是存储控制器通过如下方式区分包含替换数据的分页与包含最新更改版本的分页：(1)例如使用计数器记录写入具有相同逻辑地址的分页的次序；以及/或者(2)根据物理分页地址，其中，当在分块内按照从最低分页地址到最高分页地址的次序写入分页时，较高物理地址包含最新数据副本。因此，当对数据进行读取访问时，如果存在多个包含具有相同逻辑地址的替换数据的分页，则使用最新分页中的数据，而忽略替换数据。

参照图 8 和 9 对该技术的第一特定实现进行描述。本例的情形与参

照图 4-7B 所述的现有技术相同，即部分重写分块 35 内的数据，但是现在示出每个分块包含 16 个分页。类似于前面所述，分块 35(PBN0)的分页 3-5 中的新数据 37 写入到先前已被擦除的新分块 39(PBN1)的三个分页中。写入到包含更新数据的 PBN1 分页中的 LBN 和分页偏移开销数据字段 41 与初始分块 PBN0 内的替换数据分页相同。根据字段 41 和 41'内的数据形成的图 9 的表示出这一情形。第一列中的逻辑 LBN 和分页偏移映射到第二列中的第一物理分块(PBN0)，并且对于已被更新的分页，还映射到第三列中的第二物理分块(PBN1)。各自写入到新分块 PBN1 内三个更新数据分页中的 LBN 和逻辑分页偏移 41'均与写入到原始分块 PBN0 的对应逻辑分页中的 LBN 和逻辑分页偏移 41 相同。

为了确定具有相同 LBN 和分页偏移的两个分页中哪个分页包含更新数据，每个分页包含另一开销字段 43 来表示其编程时间，该时间至少相对于对具有相同逻辑地址的其它分页进行编程的时间。这就允许控制器确定当从存储器读取数据时分配有相同逻辑地址的数据分页的相对年龄。

存在可以写入包含某种形式的时间戳的字段 43 的若干方式。最直接简单的方式是在该字段中记录对其相关分页的数据进行编程的时间，即系统中实时时钟的输出。具有相同逻辑地址的较晚编程分页因而在字段 43 中记录有较晚时间。但是当在系统中不能得到这种实时时钟时，可以使用其它技术。一种特定技术是存储模 N 计数器作为字段 43 的值。计数器的范围应比被设计为以相同逻辑分页号存储的分页数大 1。当更新原始分块 PBN0 的特定分页数据时，例如，控制器首先读取其数据正被更新的分页的字段 43 中所存储的计数，对计数增加某个量如 1，然后将增加之后的计数作为字段 43'写入在新分块 PBN1 中。当达到计数 N+1 时，计数器回到 0。由于具有相同 LBN 的分块数小于 N，因此存储计数值总是存在不连续点。因而在规格化到不连续点的情况下便于处理回零。

当请求读取数据时，控制器通过比较具有相同 LBA 和分页偏移的分页的字段 43 和 43'中的计数，容易区分新的和替换分页的数据。然

后，根据读取数据文件最新版本的需要，将被标识为新的分页中的数据与未被更新的原始分页组合在一起以形成数据文件的最新版本。

需要注意的是，在图 8 的例子中，新数据分页 37 存储在新分块 PBN1 的前三个分页 0-2 中，而不是原始分块 PBN0 内它们所替换的相同分页 3-5 中。通过记录各个逻辑分页号，在新分块中存储更新数据的分页偏移不需要一定与包含替换数据的旧分块相同。更新数据分页也可以写入到与正被替换数据的分页相同的分块的擦除分页。

这样，所述技术对新数据可以写入到哪个物理分页中没有任何约束。但是，实现这些技术的存储系统可能存在某些约束。例如，一种与非系统要求以顺序方式对分块内的分页进行编程。这意味着在新分块 25 中对中间分页 3-5 进行编程(图 6)将浪费分页 0-2，以后就不能对这些分页进行编程。通过在这种限制系统中将新数据 37 存储在新分块 39 的起始可用分页 3-7 中(图 8)，其余分页 3-7 可供以后使用来存储其它数据。实际上，如果在存储新数据 37 的三个分页的时候分块 39 在其分页 0-4 中存储有其它数据，则新数据可以存储在其余未用分页 5-7 中。这就最大利用这种系统的可用存储容量。

存储在图 8 的分块内的各个分页中的数据的结构例子如图 10 所示。最大部分是用户数据 45。根据用户数据算出的纠错码(ECC)47 也存储在分页中。开销数据 49 包括 LBN 和分页标记 41(逻辑分页偏移)、时间戳 43 和根据开销数据算出的 ECC 51，它们也存储在分页中。通过包括与用户数据 ECC 47 相独立的覆盖开销数据的 ECC 50，在不需要传输存储在分页中的所有数据的情况下开销 49 可以以与用户数据相独立的方式读取并且评价为有效。然而，可选地，在单独读取开销数据 49 不是频繁事件的情况下，分页中的所有数据可以由单个 ECC 覆盖，从而减小分页中 ECC 的总位数。

本发明技术的第二特定实现也可以参照图 8 来描述。在本例中，时间戳仅用来确定存储在分块中的数据的相对时长(age)，而具有相同 LBN 和分页号的数据中的最新分页通过它们的相对物理位置来确定。时间戳 43 因而不需要作为各分页的一部分来存储。而是，可以作为分

块的一部分或者在非易失性存储器内的其它地方，为每个分块记录单个时间戳，并且每次将数据分页写入到分块中时均进行更新。然后，以包含具有相同 LBN 的数据分页的最新更改分块的最后分页开始，以下降物理地址的次序从分页读取数据。

在图 8 中，例如，首先在新分块 PBN1 中从最后(分页 15)到第一(分页 0)读取分页，然后，以相同的反向次序读取原始分块 PBN0 的分页。一旦逻辑分页 3、4 和 5 已从新分块 PBN1 读取，则在读取过程期间可以忽略原始分块 PBN0 内由相同逻辑分页号标识的那些分页中的替换数据。具体地说，在本例中，一旦控制器确定旧分块 PBN0 内物理分页 3、4 和 5 的 LBN/分页 41 与已经从新分块 PBN1 读取的分页相同，则在读取期间跳过这些分页。该过程可以提高读取速度，并且减少需要为每个分页存储的开销位 49。此外，当采用该反向分页读取技术时，在读取操作期间由控制器使用的图 9 的表可以简化为图 5A 和 5B 的形式。只需要知道包含共同逻辑分块数据的那些物理分块的标识和对物理分块进行编程的相对时间，从而执行此高效读取过程。

图 11 示出通过对最初写入在分块 PBN0 中的数据进行第二次更新来扩展图 8 的例子。逻辑分页 5、6、7 和 8 的新数据 51 与它们的 LBN 和分页号一起写入到新分块 PBN1 的各自物理分页 3、4、5 和 6。注意，在本例中逻辑分页 5 的数据正在被第二次更新。在以新分块 PBN1 的最后分页开始的读取操作期间，首先依次读取感兴趣数据的最新写入逻辑分页 8、7、6 和 5。然后，需要注意，PBN1 的物理分页 2 中的 LBN/分页开销字段与从物理分页 3 中所读取的 LBN/分页开销字段相同，从而不读取分页 2 的用户数据。然后，读取物理分页 1 和 0。下一步，以物理分页 15 开始，读取原始分块 PBN0 的分页。在读取物理分页 15-9 之后，控制器将注意到各分页 8-3 的 LBN/分页字段与已经读取其数据的分页相匹配，从而不需要从这些分页中读取旧数据。因此提高读取过程的效率。最后，读取物理分页 2-0 的原始数据，因为该数据未被更新。

需要注意，由于以从分页 0 开始的次序将数据写入在擦除分块的物理分页位置，因此以反向次序读取分页的本例子从替换数据分页选出新

数据分页。然而，该技术不限于与具有这种特定编程约束的存储系统一起使用。只要在给定分块内对分页进行编程的次序是已知的，就可以采用写入数据的反向次序读取这些分页中的数据。需要首先读取具有与较早编程的其它分页相同的 LBN 的最新编程分页，并且这些分页就是最新编程分页。首先读取更新分页的最新版本，从而其后可以容易地识别替换版本。

图 12 示出图 11 的例子的逻辑数据与物理分页地址之间的对应表。虽然存在两次数据更新，但是它们均用第二分块 PBN1 的单列表示。逻辑分页 5 所对应的以 PBN1 表示的物理分页根据对该分页的第二次更新而简单改变。如果更新涉及第三分块，则为那个另一分块增加另一列。当不使用反向分页读取技术时，可以通过第一种实现来使用通过从已写入具有共同 LBN 的数据的分块内的各分页中读取开销数据而构造的图 12 的表。当使用上述反向分页读取技术时，图 12 的表只需被构建为标识 LBN 和包含具有该 LBN 的数据的所有 PBN 之间的对应关系。

组织正从其中一个或多个分页已被更新的物理分块中读取的数据分页的高效方式如图 13 所示。在控制器的易失性存储器中提供足够的空间来一次对至少若干数据分页进行缓冲，并且最好是对完整的数据分块进行缓冲。这如图 13 所示。与存储在非易失性存储分块中的量相等的十六个数据分页存储在控制器存储器中。由于分页最通常的是无序读取的，因此各个数据分页存储在它相对于其它分页的适当位置。例如，在图 11 的反向分页读取操作中，首先读取逻辑分页 8，从而它存储在控制器存储器的位置 8，如圆圈中的“1”所示。下一分页是逻辑分页 7 等等，直到读取主机所需的所有数据分页，并且将其存储在控制器存储器中。然后，将整个分页数据集合传输到主机，而无需在缓冲存储器中操纵数据次序。数据分页通过将它们写入到控制器存储器中的适当位置而已经得到组织。

图 14 的流程图示出一种对利用参照图 8 和 9 所述技术的非易失性存储系统进行编程的方法。如方框 52 所示，从主机系统接收所要更新的现有文件的分页数据。首先通过步骤 53 判定所要存储的更新数据的

分页数是否等于或大于系统分块的存储容量，为简单起见在上述例子中示出 16 个分页作为分块容量。如果是，则在步骤 55 对一个或多个未用擦除分块进行寻址，并且在步骤 57，将新数据分页写入到寻址分块。典型地，一个或多个数据分块的更新将导致一个或多个分块存储已被新数据替换的数据。如果是，如步骤 59 所示，标识具有替换数据的那些分块以进行擦除。为了提高性能，最好在背景中或者当不发生主机请求编程或读取操作时，发生擦除操作。在被擦除之后，分块返回给未用擦除分块的库存以作进一步的使用。可选地，分块的擦除可以推迟到需要它们进行编程操作的时候。

另一方面，如果在步骤 53 判定新数据的分页比利用分块的完全存储容量的分页少，则下一步骤 61 判定在具有以其它数据编程的一些分页的分块中是否有足够的未用分页。如果是，在步骤 63 对该分块进行寻址。如果否，则在步骤 65 对完全未用的擦除分块进行寻址。在任一种情况下，在步骤 67，将新数据编程到寻址分块的未用分页中。作为该编程过程的一部分，以上述方式将 LBN 和分页偏移写入到字段 41 中，并且将时间戳写入到更新数据的各分页的字段 43(图 8)中。

该编程过程的一个期望特性是使仅存储替换数据的任何分块可用于以后编程。因此，在步骤 69 询问数据更新过程是否导致仅保留有替换数据的整个分块。如果是，则在步骤 71 将该分块排入擦除队列，然后过程完成。如果否，则忽略步骤 71，并且完成数据更新。

【元分块操作】

为了通过减少编程时间来提高性能，一个目标是在不引起其它恶化的情况下以合理的方式尽可能多地对多个分区进行并行编程。一种实现将存储阵列分为大体上独立的子阵列或单元，如图 15 的多个单元 80-83，其中每个单元又如图所示分为很多分块。因而将数据分页同时编程到多个单元中。另一结构还组合来自多个存储芯片的这些单元中的一个或多个。这些多个芯片可以连接到单个总线(如图 2 所示)，或者连接到多个独立总线以提高数据吞吐量。对此的扩展是链接来自不同单元的分块以一起编程、读取和擦除，一个例子如图 15 所示。例如，来自各单

元 80-83 的分块 85-88 可以作为元分块一起操作。对于上述存储器实施例，每个分块作为存储阵列的最小可擦除组典型地分为多个分页，一个分页包含可以在分块内一起编程的最小数目的分区。因此，图 15 所示的元分块编程操作通常将包括将数据同时编程到形成元分块的各分块 85-88 的至少一个分页中，对此进行重复，直到元分块满或者输入数据全部已被编程。其它元分块由来自阵列单元的不同分块形成，其中，各单元均有一个分块。

在操作这种存储器的过程中，如同其它，需要经常更新少于整个分块的数据分页。这可以对于元分块的各个分块以相同于上面参照图 4 或 6 所述的方式来完成，但是最好使用参照图 8 所述的改进技术。当使用上述三种技术中的任一种来更新元分块中一个分块的数据时，还使用相同单元内的另一存储分块。此外，数据更新可能要求为元分块的两个或更多分块中的一个或多个分页写入新数据。因而，即使仅更新一些分页中的数据，这还会需要另外使用最大四个分块 90-93，其中，四个单元均包括一个另外分块，以更新存储在元分块中的数据文件。

为了减少这种部分分块更新所需的分块，根据本发明的另一方面，如图 16 所示，只要分块 80 中还存在未用分页，就使用存储单元 80 中的另一分块 90 来更新所述元分块的任何分块内的数据分页。例如，如果一次更新分块 86 的三个分页和分块 88 的两个分页中的数据，则新数据的所有五个分页都写入到分块 90 中。这可以节省使用一个存储分块，从而有效地将可用擦除分块的数目增加一个分块。这将帮助避免或者至少推迟擦除分块库存耗尽的时候。如果正在更新来自四个分块 85-88 中的每个分块的一个或多个分页，则在单个分块 90 中对所有新数据分页进行编程，从而避免为进行更新而额外占用三个存储分块。如果新数据的分页数超过未用分块的容量，则分块 90 不能接受的分页写入到可以处于相同单元 80 或其它单元 81-83 中的一个单元内的另一未用分块。

虽然本发明是针对各种示例性实施例来描述的，但是应该理解本发明在所附权利要求的全部范围内受到保护。

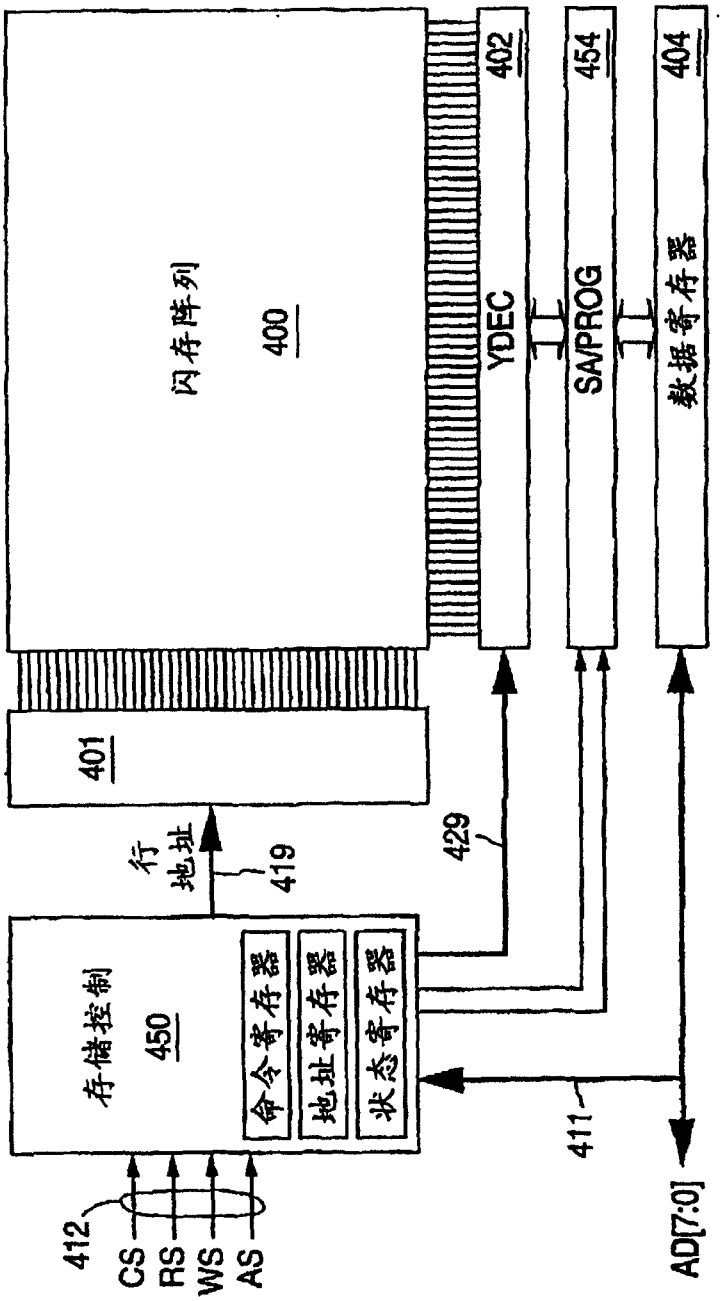

闪存阵列
400
402
YDEC
454
SA/PROG
404
数据寄存器
401
行地址
419
429
存储控制
450
命令寄存器
地址寄存器
状态寄存器
411
412
CS
RS
WS
AS
AD[7:0]

图1
（现有技术）

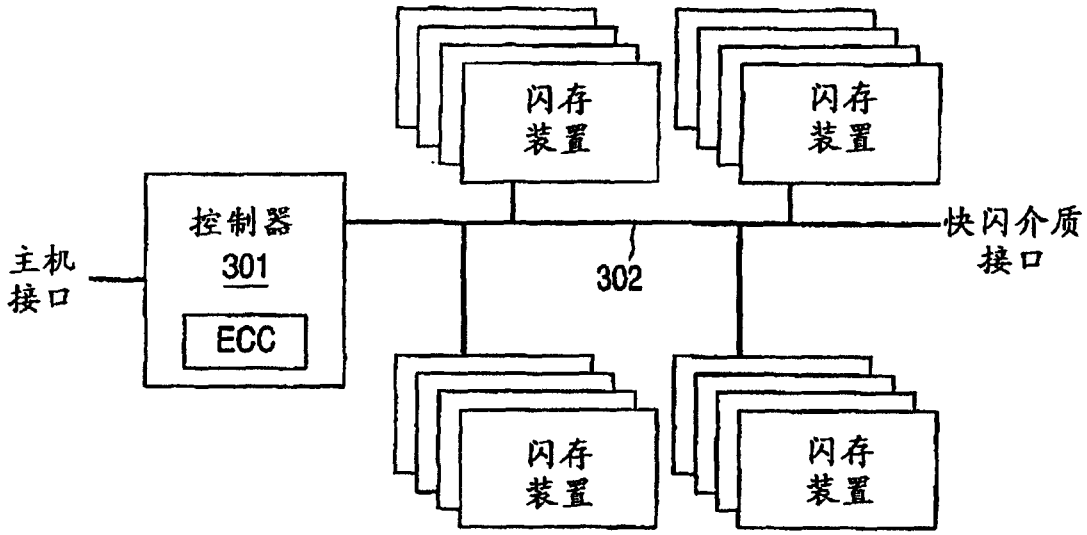


图2
（现有技术）

LBN	PBN
0	0
⋮	⋮

原始分块11

图5A

LBN	PBN
0	1
⋮	⋮

使用新分块15

图5B

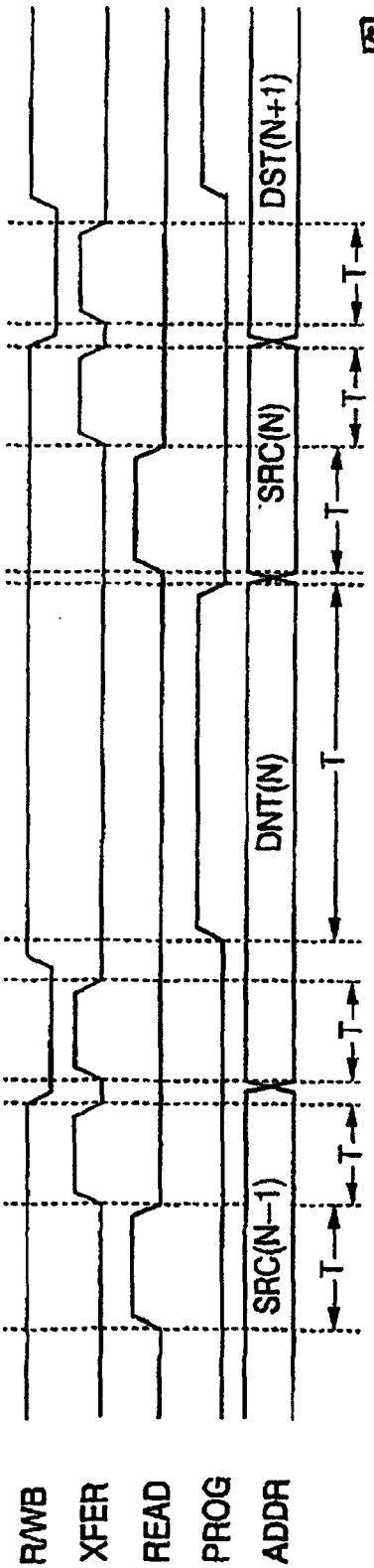


图3
（现有技术）

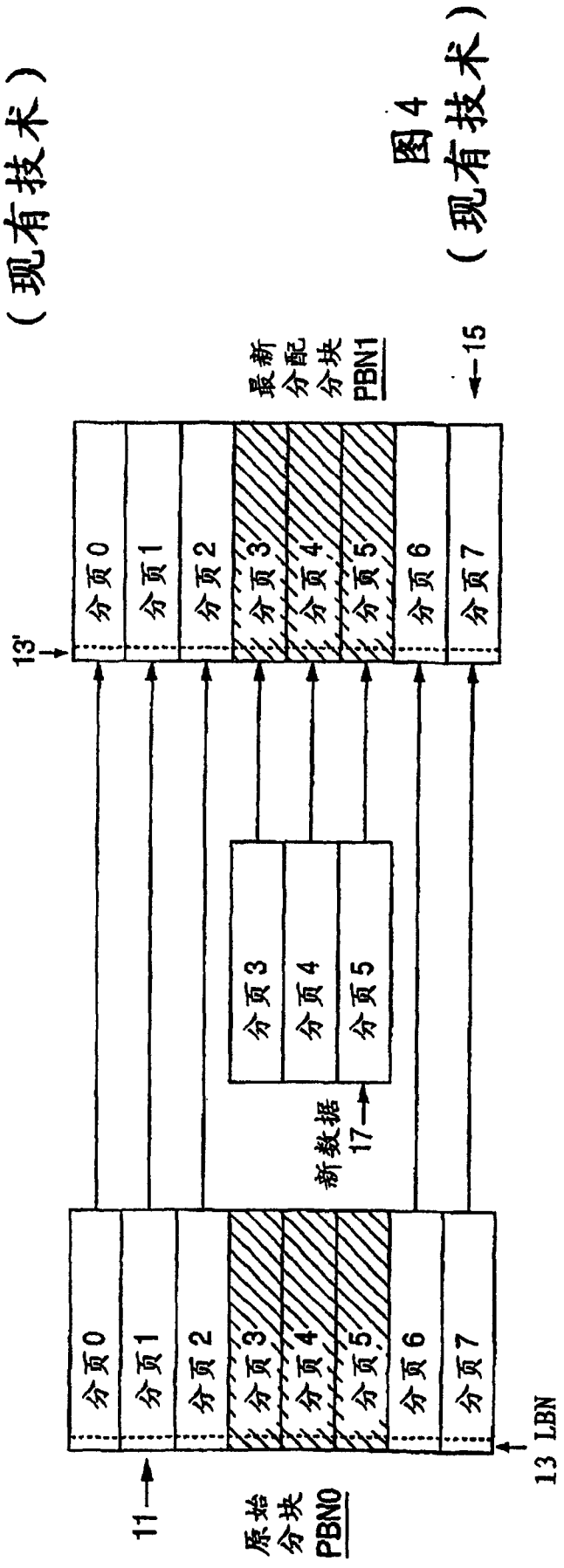


图4
（现有技术）

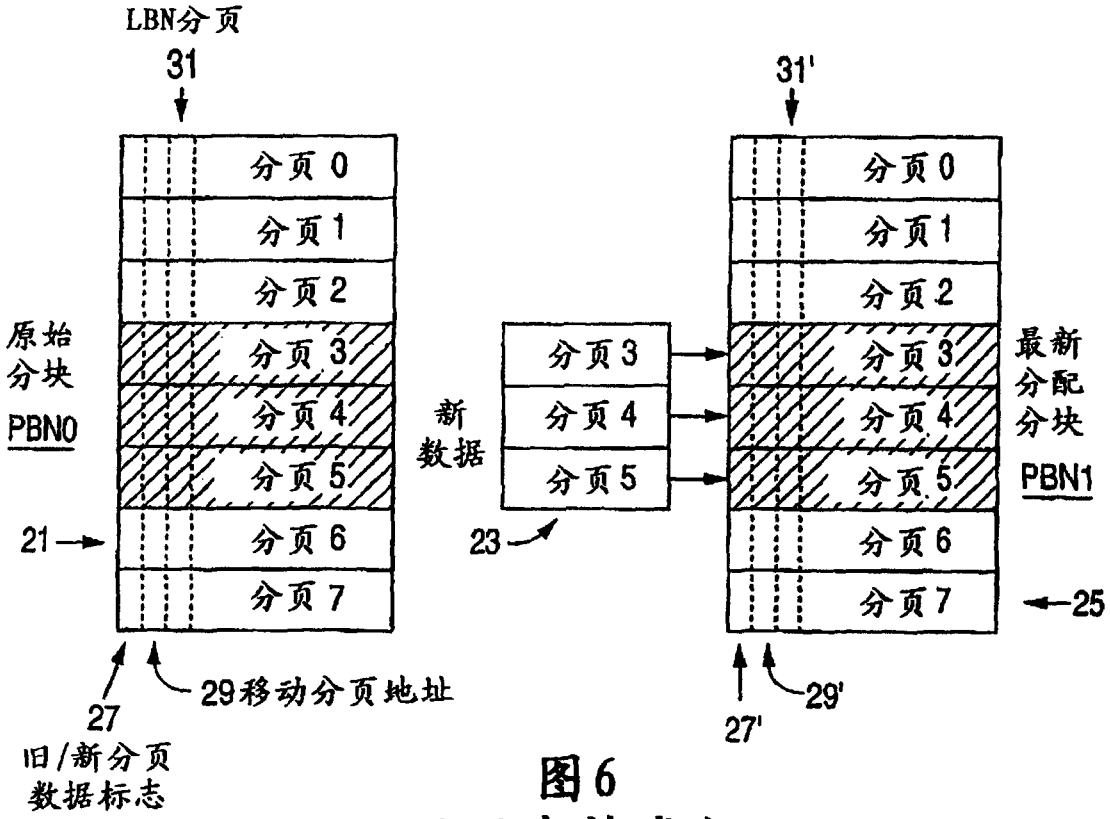


图6
（现有技术）

LBN	分页	PBN	分页
0	0	0	0
0	1	0	1
0	2	0	2
0	3	0	3
0	4	0	4
0	5	0	5
0	6	0	6
0	7	0	7
:	:	:	:

原始分块

图7A

LBN	分页	PBN	分页
0	0	0	0
0	1	0	1
0	2	0	2
0	3	1	3
0	4	1	4
0	5	1	5
0	6	0	6
0	7	0	7
:	:	:	:

使用新分块

图7B

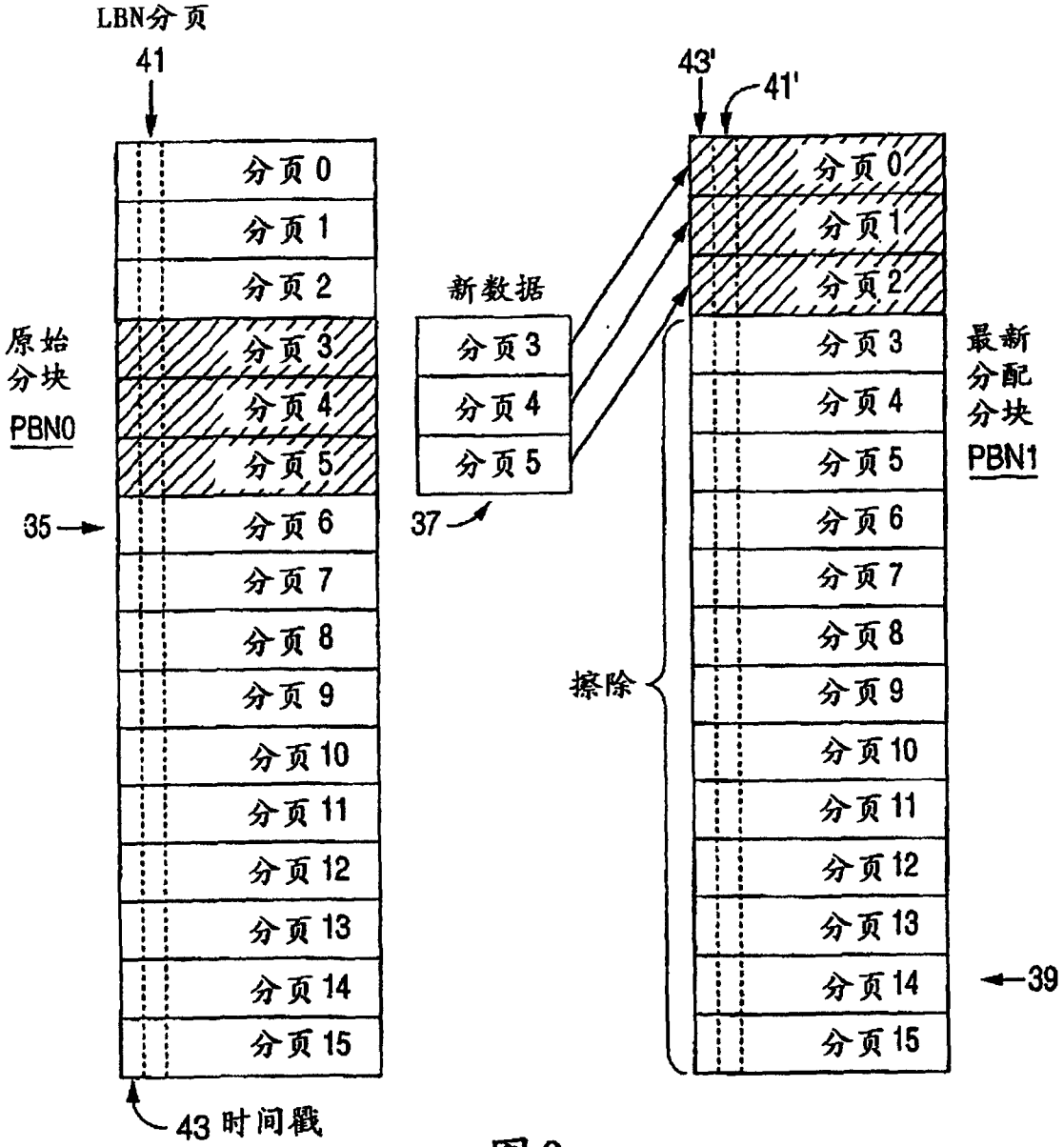


图 8

LBN	分页	PBN0	分页	PBN1	分页
0	0	0	0		
0	1	0	1		
0	2	0	2		
0	3	0	3	1	0
0	4	0	4	1	1
0	5	0	5	1	2
0	6	0	6		
0	7	0	7		

图 9

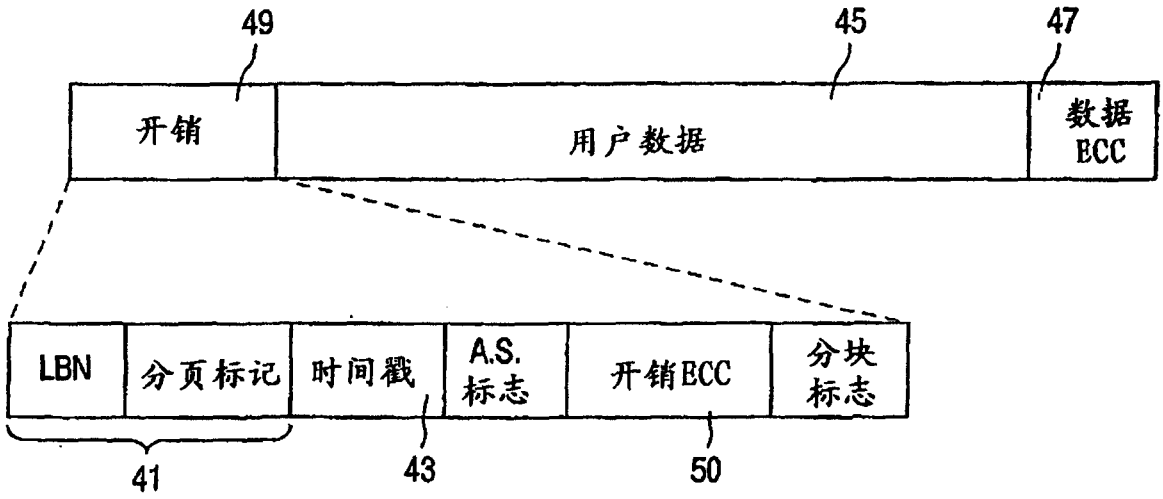


图10

LBN	分页	PBN0	分页	PBN1	分页
0	0	0	0		
0	1	0	1		
0	2	0	2		
0	3	0	3	1	0
0	4	0	4	1	1
0	5	0	5	1	3
0	6	0	6	1	4
0	7	0	7	1	5
0	8	0	8	1	6
0	9	0	9		
:	:	:	:	:	

图12

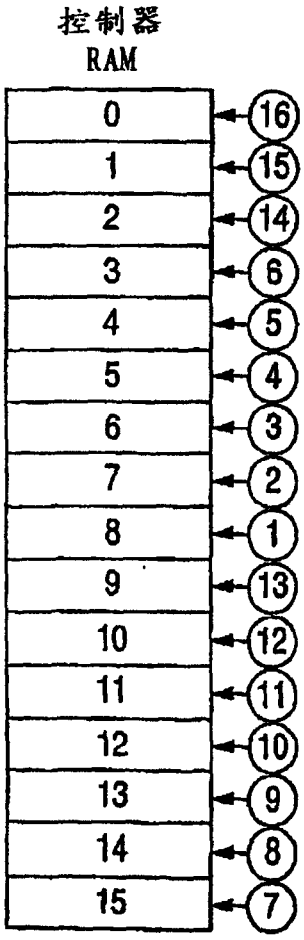


图13

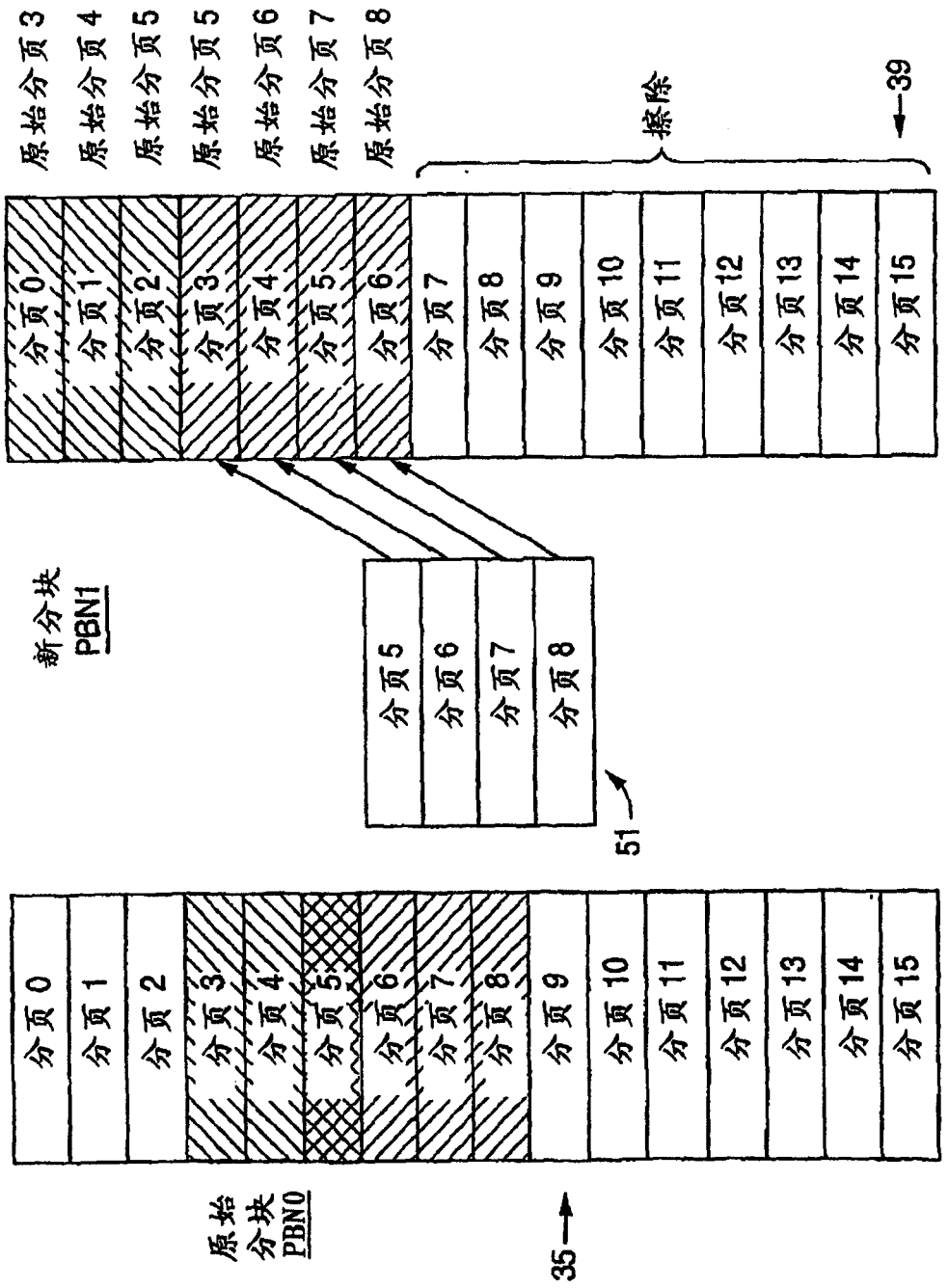

原始分页3
原始分页4
原始分页5
原始分页5
原始分页6
原始分页7
原始分页8
擦除
39
分页0
分页1
分页2
分页3
分页4
分页5
分页6
分页7
分页8
分页9
分页10
分页11
分页12
分页13
分页14
分页15
新分块
PBN1
分页5
分页6
分页7
分页8
51
原始
分块
PBN0
35

图11

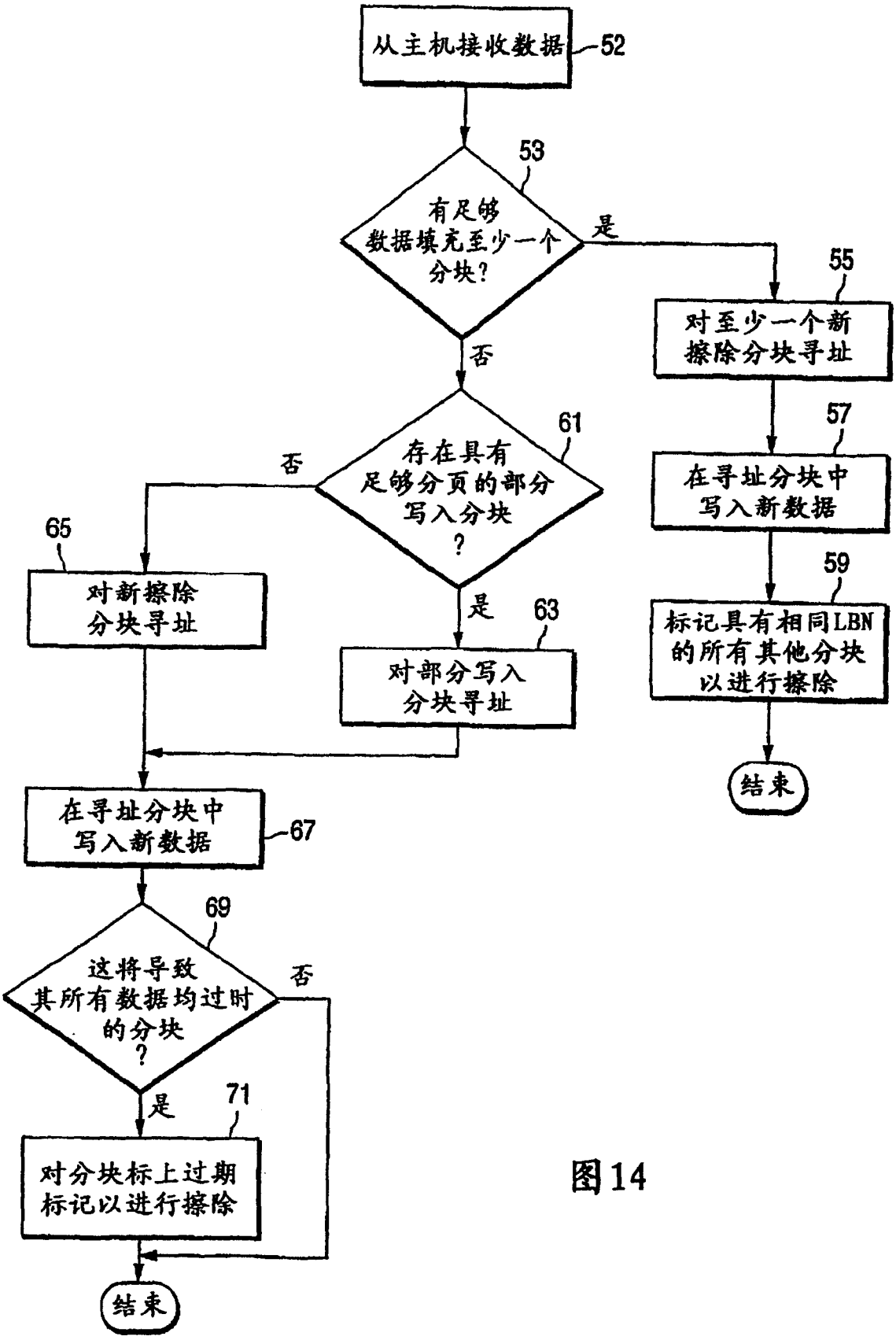

从主机接收数据
52
53
有足够数据填充至少一个分块?
是
否
55
对至少一个新擦除分块寻址
57
在寻址分块中写入新数据
59
标记具有相同LBN的所有其他分块以进行擦除
结束
61
存在具有足够分页的部分写入分块?
否
是
63
对部分写入分块寻址
65
对新擦除分块寻址
在寻址分块中写入新数据
67
69
这将导致其所有数据均过时的分块?
否
是
71
对分块标上过期标记以进行擦除
结束

图14

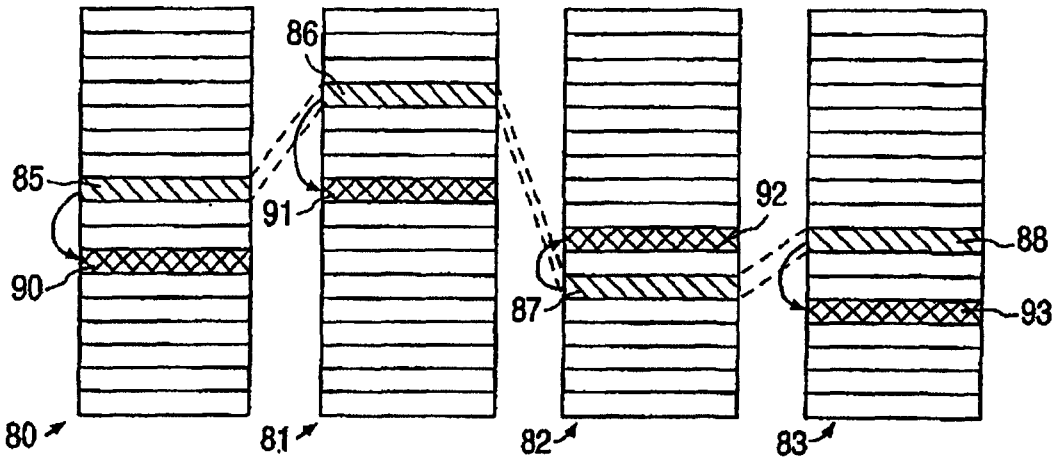


图15

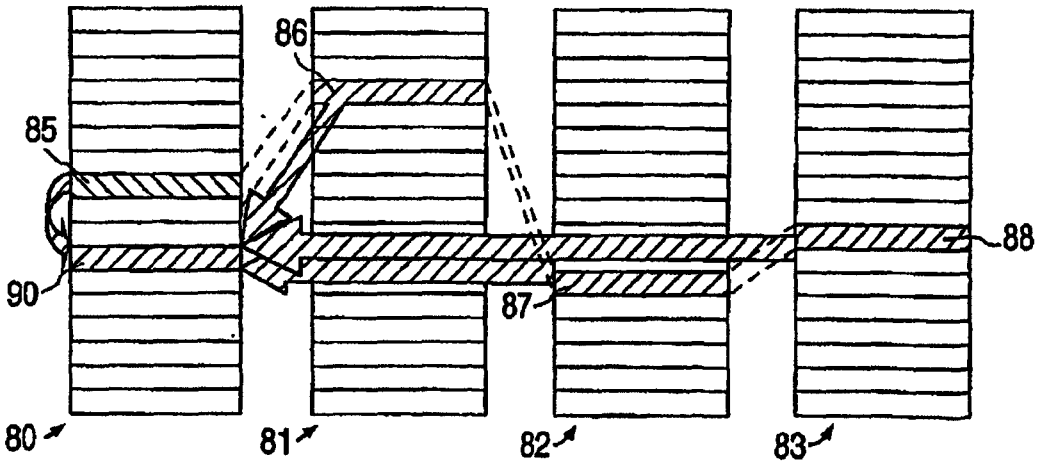


图16