

本科生毕业设计

|  |
| --- |
| 面向近似应用的固态硬盘数据布局方案的设计及实现 |

|  |  |
| --- | --- |
| 院 系 | 计算机科学与技术 |
| 专业班级 | CS1610 |
| 姓 名 | 覃映超 |
| 学 号 | U201614795 |
| 指导教师 | 崔金华 |

2020年06月10日

**学位论文原创性声明**

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包括任何其他个人或集体已经发表或撰写的成果作品。本人完全意识到本声明的法律后果由本人承担。

作者签名： 年 月 日

**学位论文版权使用授权书**

本学位论文作者完全了解学校有关保障、使用学位论文的规定，同意学校保留并向有关学位论文管理部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权省级优秀学士论文评选机构将本学位论文的全部或部分内容编入有关数据进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于 1、保密囗，在 年解密后适用本授权书

2、不保密囗 。

（请在以上相应方框内打“√”）

作者签名： 年 月 日

导师签名： 年 月 日

摘 要

基于近似存储的闪存系统有望消除存储系统的能耗高、I/O瓶颈等难题，其发展进步将带来计算机体系结构以及系统软件层的革新。但直接在闪存存储系统中应用现有近似存储技术时，没有充分考虑闪存系统中闪存芯片固有工艺偏差导致数据访问性能低下等现象。

为了解决闪存系统中不可避免的闪存工艺偏差的现象，提出了近似数据布局方案。首先，提出了近似存储技术，该技术使用近似写操作来存储应用程序中的近似数据，通过增加编程电压可以提高近似写操作的速度。使用近似读操作从固态盘中读取近似数据，通过不受错误编码纠正的读取操作可以提高读取速度。然后，提出了基于工艺偏差的数据布局方案，该方案通过将近似数据分配到可靠性较低的块中，精确数据分配到可靠性较高的块中。实验结果表明，与传统的存储方案相比，该方案将闪存访问的平均读写响应时间分别提高了55.72%和38.93%，将IOPS与吞吐量分别平均提高了35.41%和35.67%。

**关键词**：近似存储，工艺偏差，数据分配

Abstract

The flash memory system based on approximate storage is expected to eliminate the problems of high energy consumption and I/O bottlenecks in the storage system. Its development and progress will bring innovations to computer architecture and system software layer. However, when the existing approximate storage technology is directly applied to the flash memory storage system, the phenomenon of poor data access performance caused by the inherent process variation of the flash memory chip in the flash memory system is not fully considered.

In order to solve the inevitable flash process variation in flash memory system, an approximate data layout scheme is proposed. First of all, the approximate storage technology is proposed, which uses the approximate write operation to store the approximate data in the application program. By increasing the programming voltage, the speed of the approximate write operation can be improved. The approximate read operation is used to read the approximate data from the solid-state disk, and the read speed can be improved by the read operation which is not protected by error code correction(ECC). Then, a data allocation scheme based on process variation is proposed, which allocates approximate data to flash blocks with low reliability and precise data to flash blocks with high reliability. The experimental results show that compared with the traditional storage scheme, this scheme improves the average read/write response time of flash memory access by 55.72% and 38.93% respectively, and improves the IOPS and throughput by 35.41% and 35.67% respectively.

**Keywords:** Approximate Storage，Process Variation，Data Allocation

目 录

[摘 要 I](#_Toc40001008)

[Abstract II](#_Toc40001009)

[1 绪 论 1](#_Toc40001010)

[1.1 课题背景 1](#_Toc40001011)

[1.2 国内外研究现状 1](#_Toc40001012)

[1.3 研究目的和主要内容 3](#_Toc40001013)

[1.4 论文结构 3](#_Toc40001014)

[1.5 课题来源 4](#_Toc40001015)

[2 固态盘背景概述 5](#_Toc40001016)

[2.1 闪存单元 5](#_Toc40001017)

[2.2 增量步进脉冲编程 5](#_Toc40001018)

[2.3 闪存芯片逻辑结构 6](#_Toc40001019)

[2.4 闪存转换层 7](#_Toc40001020)

[2.5 数据分配 8](#_Toc40001021)

[2.6 闪存工艺偏差 9](#_Toc40001022)

[2.7 固态盘模拟器 10](#_Toc40001023)

[2.8 本章小结 10](#_Toc40001024)

[3 近似存储背景概述 11](#_Toc40001025)

[3.1 近似计算 11](#_Toc40001026)

[3.2 近似数据识别 11](#_Toc40001027)

[3.3 近似写入 12](#_Toc40001028)

[3.4 近似读取 13](#_Toc40001029)

[3.5 本章小结 14](#_Toc40001030)

[4 数据布局方案设计 15](#_Toc40001031)

[4.1 块的可靠性特征 15](#_Toc40001032)

[4.2 写入速度 16](#_Toc40001033)

[4.3 读取速度 17](#_Toc40001034)

[4.4 数据分配 17](#_Toc40001035)

[4.5 设计中考虑的制约因素 18](#_Toc40001036)

[4.6 成本估算 18](#_Toc40001037)

[4.7 本章小结 18](#_Toc40001038)

[5 数据布局方案实现 19](#_Toc40001039)

[5.1 块的可靠性量化 19](#_Toc40001040)

[5.2 块的速度设置 19](#_Toc40001041)

[5.3 近似存储 20](#_Toc40001042)

[5.4 数据分配 22](#_Toc40001043)

[5.5 本章小结 26](#_Toc40001044)

[6 性能测试与分析 27](#_Toc40001045)

[6.1 测试环境 27](#_Toc40001046)

[6.2 性能测试 28](#_Toc40001047)

[6.3 本章小结 31](#_Toc40001048)

[7 总结与展望 32](#_Toc40001049)

[7.1 全文总结 32](#_Toc40001050)

[7.2 未来工作展望 32](#_Toc40001051)

[致 谢 33](#_Toc40001052)

[参考文献 34](#_Toc40001053)

# 绪 论

本章首先简述了课题的背景和近似存储需求的产生，然后介绍了国内外在近似存储和闪存工艺偏差方面的相关研究工作，最后对本文的主要研究内容以及目的作了说明。

## 课题背景

传统的机械硬盘主要由机械零件组成，其结构比较复杂。与之相反，基于闪存的固态盘中不包含机械零件，而是由几个主要组件构成：闪存、控制器、连接器、DRAM缓冲区等电子装置，避免了传统机械硬盘在寻道和旋转上的延时，因此固态盘的读写性能很高。与机械硬盘相比，固态盘的结构简单并且没有机械零件，其故障率大大低于机械硬盘。并且在过去的二十年中，随着闪存制造工艺水平的提升和存储密度的增加，固态盘容量一直在稳步提升，因此在服务器、移动设备、客户端设备等场景中，正在逐渐取代传统机械硬盘成为主流的存储介质。

如今越来越多的应用程序对存储的数据的精度要求没那么高了，因此在为了确保数据能够精确存储的操作上所花费的时间和空间对于这些应用程序来说是浪费的。例如在线观看视频时，视频文件将会被缓存起来，当缓存已满时，视频数据将被写入闪存。由于下列两个原因，缓存的视频数据可以承受较高错误率：（1）这些数据大多数是编码的多媒体数据，具有很高的容错性，因为这些数据出现少量错误时对视频的影响通常并不明显；（2）即使它们已损坏，系统也可以从Internet重新下载。还有在计算机视觉，机器学习和传感器数据分析等领域中的应用程序也可以看到对计算精度较小的妥协，以换取效率的提高。基于这种背景，这些应用的请求数据在存储层近似存储能够提高固态盘的性能，能效和容量。

## 国内外研究现状

### 近似计算

近似存储建立在三大类相关工作的基础上：近似计算，优化对存储单元的访问以及容忍固态存储器中的故障。

近似计算是一个研究领域，旨在使用硬件和软件技术来优化容错程序的执行。程序员使用编程语言的功能，分析或程序逻辑来控制近似执行。许多有关近似计算的工作都集中在计算本身（优化算法或处理器逻辑），还有一些工作利用了非易失性固态存储器（如PCM和Flash）属性的技术：损耗和多级单元[2]。并针对这两种属性提出了两种机制，以此提高固态存储器的性能，寿命和密度。第一种通过减少或消除保护带，使得用于写入多级单元的编程脉冲的数量大大减少，从而加快写入速度。第二种机制通过将近似数据分配到已用尽其硬件纠错资源的块上，从而减轻了磨损故障带来的影响并延长了存储器的寿命。

### 闪存工艺偏差

工艺偏差在先前的工作中已得到广泛的研究。最新的工作主要是利用工艺偏差来提高写入性能[1]、读取性能[12]或延长使用寿命[3][20][21][23]。Shiet等人[1]试图通过在更可靠的块上使用更高的编程速度来提高写入性能，从而在数据的写入速度和原始误码率之间进行权衡。李等人[12]试图将读取热数据分配到可靠性较高的块中，以此提高读取速度。崔等人[24]提出了一种基于读取数据的热度和保留期限的I/O调度方案，以减少冲突。其他[3][20][21][23]，提出了基于工艺偏差的磨损均衡方案。由于高可靠性块可以承受更多的P/E周期，因此他们建议为高可靠性的块分配更多的写请求，以延长使用寿命。Di[22]等人进一步研究了块之间的保留时间差异，并利用它来减少NAND闪存上刷新操作的频率。

### 数据分配方案

固态盘中最关键的固件算法是闪存转换层（FTL），它主要职责之一是管理上层请求数据的逻辑地址到固态盘中的物理地址的映射。目前有大量的研究致力于设计出更高效的闪存转换层策略来实现合理的数据布局方案。针对与本文研究相近的国内外研究现状进行调查分析，主要分为两大类：

1. 利用了闪存工艺偏差的数据分配方案：将读取的热数据分配给可靠性高的块，读取的冷数据分配给可靠性低的块，从而提高了读取性能[12]。将写入的热数据分配给可靠性高的块，写入的冷数据分配给可靠性低的块，从而提高了写入性能[1]。
2. 利用了近似数据的数据分配方案：在一个数据页中同时存储近似数据和精确数据，页面的纠错功能将全部用于纠正精确数据中的错误，而近似数据将在没有错误编码纠正（ECC）保护的情况下进行存储[7]。采用模式引导页面分配的设计，该设计优先考虑在物理块中将近似数据与精确数据交替存储，以减少编程干扰[5]。

## 研究目的和主要内容

在本次课题的研究中有2个重要部分，分别是近似存储和数据分配。其中近似存储是基础部分，而数据分配方案则要在近似存储的基础上，加以考虑工艺偏差给不同的闪存块之间带来的可靠性差异的影响，进而利用块的可靠性差异来设计数据分配方案，达到提升闪存存储系统性能的目的。

本课题将从以下几个方面进行研究：⑴利用近似数据来设计近似存储方案。⑵利用近似数据和闪存工艺偏差的特性来设计数据布局方案。⑶将设计的近似存储方案和数据布局方案在固态盘模拟器中实现，并验证和评估本文提出的方案。

## 论文结构

本文的主要内容如下：

第一章首先简述了课题的背景和近似存储需求的产生，然后简述了国内外在近似存储、闪存工艺偏差和数据分配方面的相关研究工作，最后对本文的主要研究内容以及目的作了说明。

第二章主要介绍了固态盘的背景知识，首先是固态盘的闪存单元结构，其中包括单级单元、多级单元；然后是闪存单元的编程方式，主要是阐述了增量步进脉冲编程方案；接着是闪存芯片的逻辑结构，其中包括闪存的组织方式以及固态盘基本操作的粒度；再接着就是闪存转换层，主要阐述了其两个主要职责：地址映射和垃圾回收；最后是闪存工艺偏差给固态盘带来的影响。

第三章主要介绍了近似存储的背景知识，首先是近似计算，主要阐述了其概念以及与近似存储的关系；然后是近似数据的识别，主要阐述了近似数据识别的意义以及当前识别近似数据的方案；接着是近似写入，主要阐述了两种近似写入方案，包括增大编程步长以及降低阈值电压，并对比了两种方案优势；最后是近似读取，主要阐述了近似读取的方法。

第四章主要是数据布局方案的设计，首先阐述了速度检测方案，包括块的可靠性特征，块的写入速度，读取速度的检测。然后阐述了数据分配的基本思想。最后就是设计中考虑的制约因素和成本估算。

第五章主要将设计的布局方案在固态盘模拟器中实现。首先是块的可靠性量化，包括初始化以及更新块的可靠性变量。接着是实现近似存储，其中包括近似写入和近似读取。然后就是数据分配方案，其中包括物理块分配和管理的实现，以及垃圾回收的改进。

第六章说明了对本文实现的数据布局方案的测试过程，包括对读取写入平均响应时间，IOPS和吞吐量的测试，以及对测试结果的分析。

第七章对本文所做的工作进行总结，然后基于本文的工作提出了更为完善的发展方向。

## 课题来源

本课题为湖北省自然科学基金项目“基于近似存储的闪存系统关键技术研究”中的一个基础课题。该项目紧密结合大数据时代环境下存储系统所遭遇的难题，利用近似存储技术对不精确的数据具有内在容忍度的优势，深入研究基于近似存储的闪存系统中关键技术。利用近似存储的优势，构建一个稳定高效的闪存存储系统。通过解决关键技术问题，更好地发挥基于近似存储的闪存存储系统的性能、寿命和容量优势，有利于促进存储设备主控算法的国产化发展，提高我国在大数据存储系统方面的技术水平。

本毕设课题作为该项目中的一个研究内容，主要致力于研究闪存存储系统中的工艺偏差的特征，设计并实现一种工艺偏差感知的近似数据布局方针，提升面向近似应用的闪存固态硬盘的I/O性能。

# 固态盘背景概述

如今固态硬盘已经普遍配备在移动设备，服务器等领域中。为了提高硬盘容量，闪存单元从单级单元逐渐发展为多级单元，此外，采用三维结构的闪存通过垂直堆叠闪存单元进一步提高了存储容量。但是闪存发展的不利因素是可靠性降低，以及工艺偏差造成的可靠性差异。

## 闪存单元

NAND闪存将特定数量的电荷存储在浮栅或电荷陷阱中用于表示存储的数据。在单级单元（SLC）设备中，每个存储单元存储一位信息。多级单元（MLC）设备中，每个单元存储两位信息。随着位密度和技术规模的发展，三级单元（TLC）、四级单元（QLC）的固态盘正逐渐投入使用。每个单元存储更多的位可以增加固态盘的容量，而不会增加芯片大小，但是会由于使正确存储和读取位变得困难而降低了可靠性。

如果在闪存单元中存储n位，就需要种不同的电压状态。图2‑1显示了多级单元（MLC）闪存在一个闪存单元中存储两位信息的例子，其中定义了四个电压状态，通过在编程写入操作中将电荷注入到浮栅中，每个闪存单元将转变为四个电压状态之一。两个相邻电压状态之间的缝隙称之为保护带。

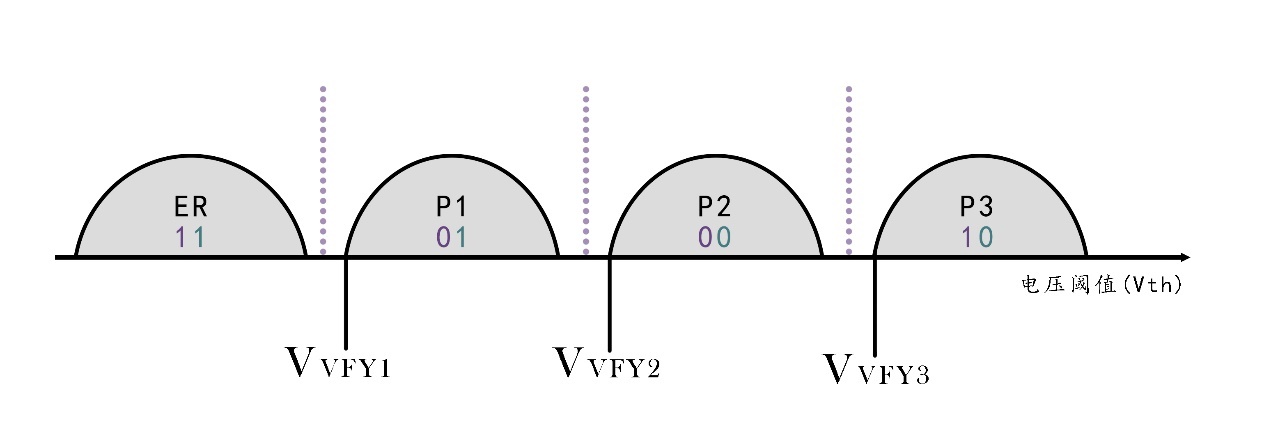


图2‑1 闪存单元（MLC）的电压分布

## 增量步进脉冲编程

目前，闪存编程方式大部分采取的都是增量步进脉冲编程（ISPP）策略。典型的增量步进脉冲编程方案由多个步骤组成，每个步骤的编程电压增量通常称为ΔVpp，如图 2‑2所示。在每个编程步骤之后，执行验证操作以检查单元电压是否高于目标电压电平（目标Vth），并确定编程过程是否可以终止。

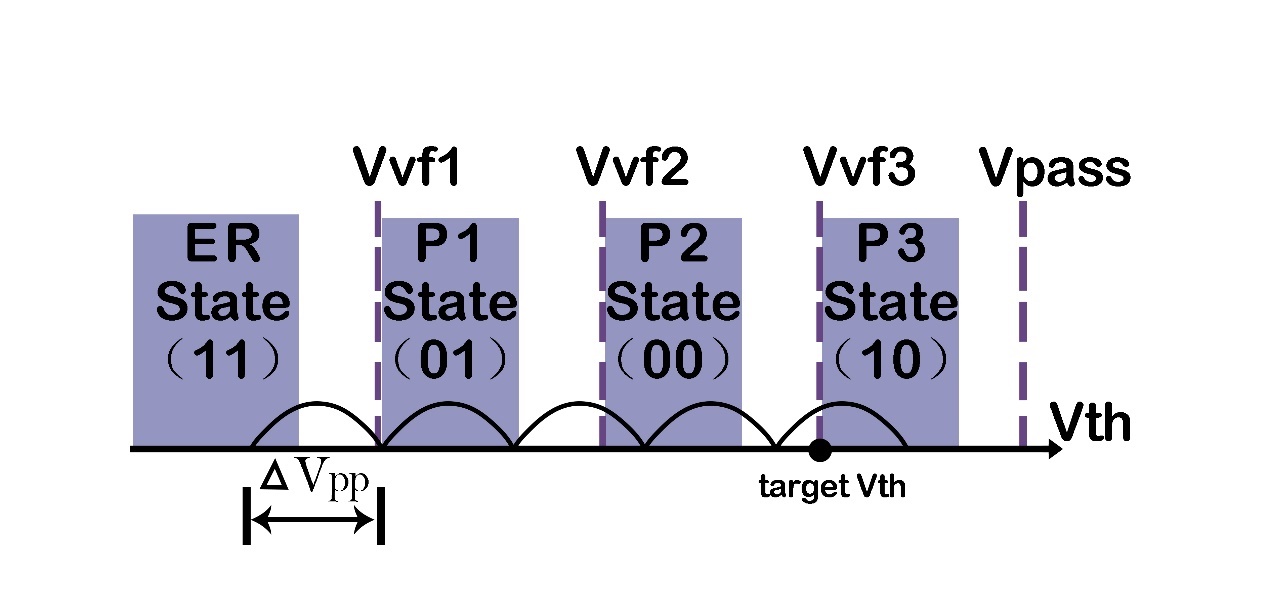


图 2‑2增量步进脉冲

研究表明，闪存的可靠性取决于每个状态的电压分布以及相邻电压状态之间的保护带。编程后单元电压可能会由于干扰和电荷泄露而产生波动，少数闪存单元的电压可能会偏离其电压状态的范围，从而导致原始位错误（RBE），因此在将数据从闪存中读取出来时，需要经过错误编码纠正（ECC）修复。闪存页面包含的位错误越多，读取期间进行纠错的过程越长，读取性能就越低。选择较小的ΔVpp有助于减小电压状态范围，使得保护带范围变大，即数据更可靠，因此读取速度也更快。但是，完成写入操作所需要的步骤增多，写入时间增加。

## 闪存芯片逻辑结构

图 2‑3显示了如何在固态盘内组织NAND闪存的示例。闪存存储体（flash memory）包含多个闪存芯片（chip），其中每个闪存芯片中包含多个晶圆（die），晶圆也是可以单独执行命令和返回状态的最小单位。如今的固态盘通常具有4-16个闪存芯片，每个芯片可以具有多达16个晶圆。每个闪存芯片都连接到一个或多个物理内存通道（channel），并且这些内存通道不在芯片之间共享。闪存晶圆之间相互独立，并且包含1-4个分组（plane）。每个分组中包含数百到数千个闪存块（block）。每个块中又包含着数十个页面（page）。

块中的数据以页面为单位进行写入，在NAND闪存中页面大小通常为4KB-16KB。所有读取和写入操作均以页面的粒度执行，而擦除操作则是以块的粒度执行。

每个闪存通道与固态盘控制器之间都有自己的数据和控制连接，数据和命令都可以通过通道发送。

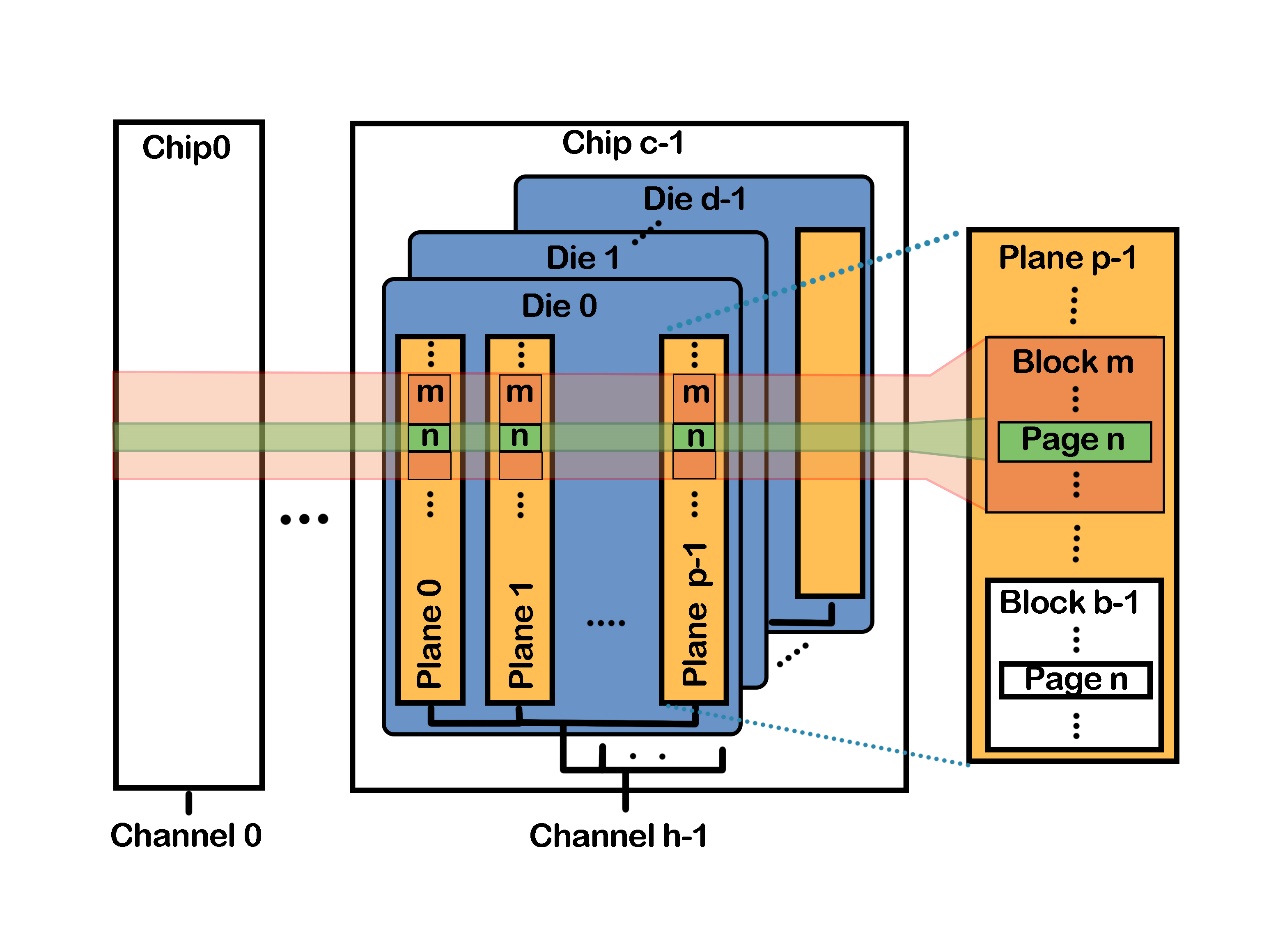


图 2‑3闪存芯片结构图

## 闪存转换层

闪存转换层（FTL）是基于NAND闪存的固态盘中的关键组件，闪存转换层的主要职责之一是管理每一页数据的逻辑地址（即主机使用的地址空间）到底层闪存中的物理地址（即数据所在实际位置的地址空间）的映射，如图 2‑4所示。通过在地址空间之间提供这种间接方式，闪存转换层可以在不通知主机的情况下将逻辑地址重新映射到其他物理地址（即将数据移动到其他物理地址）。每当主机写入页面数据或移动页面数据以进行固态盘基本维护操作时（例如垃圾回收，见下文），只是在物理块的页面中将旧数据所在的页面标记为无效，然后将新数据写入闪存块中当前可以写入的页面。

随着时间的流逝，页面无效会导致块内产生碎片，即块中的大多数页面变为无效。闪存转换层定期执行垃圾回收操作，垃圾回收标识每个高度碎片化的闪存块并擦除整个块（将所有剩余的有效页面迁移到新的块之后，目的是用有效页面完全填充新块）。垃圾回收操作通常首先选择有效页面最少的块，在擦除完整个块后，会将该块重新标记为空闲。当前正在操作的可用块已满时，闪存转换层会在空闲的块中选择一个新的块，用于执行写入操作。

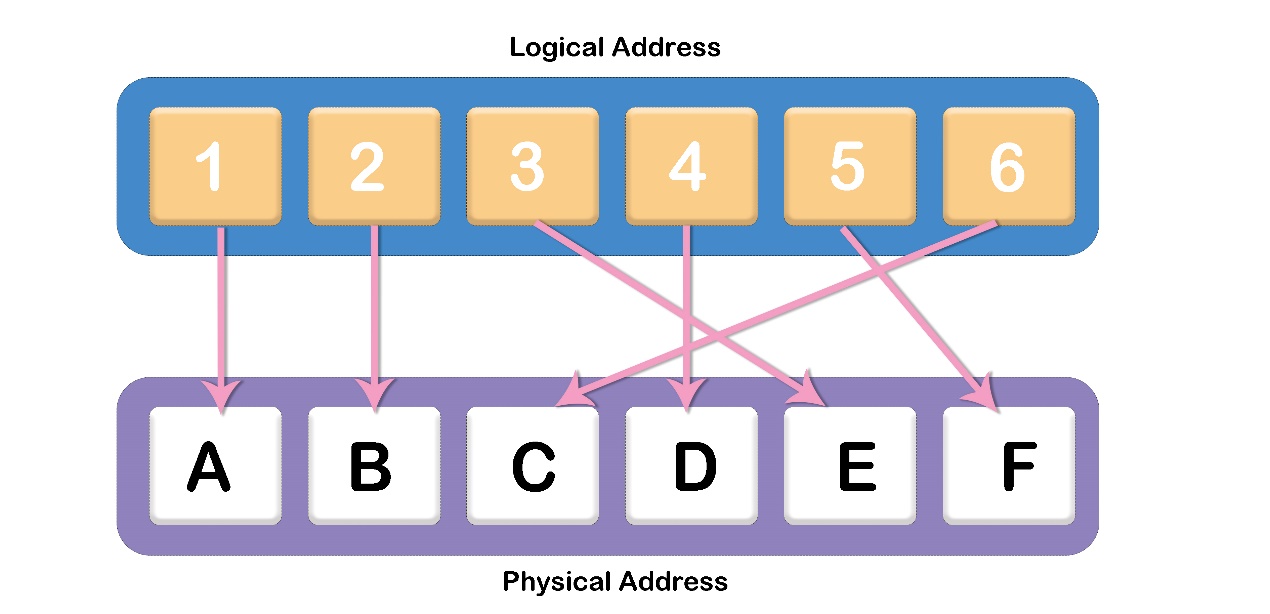


图 2‑4地址映射

## 数据分配

数据分配是为了给数据的逻辑地址分配相应的物理地址，目前常用的分配方式有静态分配和动态分配两种。其中动态分配时，物理地址对应的通道号、芯片号、晶圆号、分组号都是按照轮询令牌标志token进行分配，即物理地址信息根据token给出。令牌标志token的更新操作如图 2‑5所示，按照替换的顺序，轮询分配到每个die的缓冲区，以充分利用晶圆（die）间并行性。而channel的token主要用于保证每次从不同的channel开始查询，查询状态为空闲的channel用于为数据分配channel号，以充分利用通道（channel）间并行性。

通过动态或者静态的分配方式获得通道号、芯片号、晶圆号、分组号后，还需要为数据分配相应的闪存块以及页面编号。通常每个分组都维护有一个活动块，即可以直接用于分配的块编号，在每个闪存块中，页面是按照顺序写入。

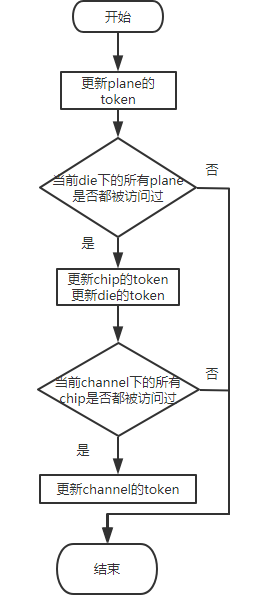


图 2‑5 token更新

## 闪存工艺偏差

工艺偏差是晶体管属性自然发生的差异[8][9][10]。随着技术的发展和位密度的提高，闪存的工艺差异已经被放大，特别是对于不同块所支持的可用P/E周期数[3][4]。许多工作表明，一个块的可用P/E周期数表示该块的强度。Pan等[3]指出闪存块的误码率（BER）遵循对数高斯分布，最终反映了闪存块可用P/E周期数的分布。Woo等[4]进一步评估了块的可用P/E循环数远高于上述数字，并且块之间存在着显著差异。基于这些观察，可以得出结论，工艺偏差会导致闪存块强度明显变化。随着闪存存储容量需求的急剧增加，三维闪存（3D NAND）通过在第三个维度上堆叠闪存芯片来提升存储容量获得了广泛地研究。但是，工艺偏差在闪存块中仍然广泛存在。在3DNAND中，闪存块中的闪存单元彼此垂直堆叠，并且由于制造工艺的原因，3DNAND单元的大小在不同层之间会有很大差异，这称为层间的工艺偏差。

## 固态盘模拟器

本文在实现数据布局方案时，采用的是3Dsim模拟器。下面简单介绍在本文中提到的有关模拟器的数据结构。

在模拟器中，无论是外部请求还是内部产生的子请求，均由操作类结构体完成。与本文内容相关的操作类数据结构如表 2‑1所示，

表 2‑1操作类数据结构

|  |  |  |
| --- | --- | --- |
| 命名 | 意义 | 主要成员 |
| request | 外部请求 | 操作类型、LSN、大小、到达时间 |
| sub\_request | 子请求 | 类型、LPN |
| local | 物理地址 | 通道号、芯片号、晶圆号、分组号、块号、页号 |

## 本章小结

本章介绍了关于闪存的一些背景知识，包括闪存单元、增量步进脉冲、闪存工艺偏差、固态盘模拟器。在介绍闪存单元时，阐述了多级单元的概念，固态盘存储数据的方法；介绍增量步进脉冲时，主要阐述了不同的ISPP方案对性能以及可靠性的影响；介绍闪存芯片逻辑结构时，主要阐述了一个闪存芯片的构成要素；介绍闪存转换层时，主要阐述了闪存转换层的地址映射和垃圾回收功能；介绍闪存工艺偏差时，主要阐述了由于制造工艺的原因导致闪存块之间存在明显的可靠性差异。最后还简单说明本文在实现时采用的模拟器，以及在本文中提及的数据结构。

# 近似存储背景概述

近似存储技术提供了一种途径，用于缓解存储领域中面临的空间开销大、能耗高、访问性能低等挑战。在本节中，主要介绍了近似存储的背景知识。

## 近似计算

近似计算是一种新兴的计算技术，它利用了许多现代应用程序（例如音频、视频和图像处理应用程序）中固有的错误恢复能力[13]。例如，在渲染视频流时出现少量错误通常并不明显；还有在高分辨率图像中具有少量错误像素通常不会影响图像分类。为了确保这些应用程序的整体服务质量（QoS），在程序员[13]或者编译器[14]的帮助下，数据通常分为关键数据和非关键数据。虽然非关键数据中可能会出现错误，但关键数据仍保持精确。将近似计算扩展到固态存储器中就可以构建近似存储。

## 近似数据识别

识别近似数据是近似存储的前提[27]，应用程序的数据并不是都能够近似处理，所以需要标识近似数据，然后对近似数据执行近似存储操作。应用程序的近似数据通常具有容错性，因为具有容错能力的近似数据在执行近似操作后其质量降低对原应用程序的影响不大。对于近似应用而言，通常近似的数据占全部数据的比重更大，近似存储不仅能够提升固态盘的读写性能，还可以提高应用的整体性能，进而提高用户体验。

目前，用于区别近似数据和精确数据的方法有故障注入法和程序分析法。其中程序分析法，主要依靠程序员或近似计算感知的编译器（例如Enerj[2]）对近似数据进行注释。而故障注入法则是向应用程序中输入错误的数据，观察输入错误的数据后对应用运行时的影响，输入错误的数据后，不会造成应用运行错误或者崩溃的数据类型可以判定为近似数据。故障注入法需要的时间很长，但是适用性很强，具有一般性。

## 近似写入

### 增大编程步长

根据闪存的写入速度与编程步长（ΔVpp）之间的关系，近似写入的一种简单实现方式是使用比传统存储模式更大的ΔVpp，从而减少了写入步骤的数量，并提高了写入性能。如图 3‑1所示，采用更大的ΔVpp会缩小多级单元电压分布状态之间的保护带，但会导致原始误码率（RBER）显著增加。

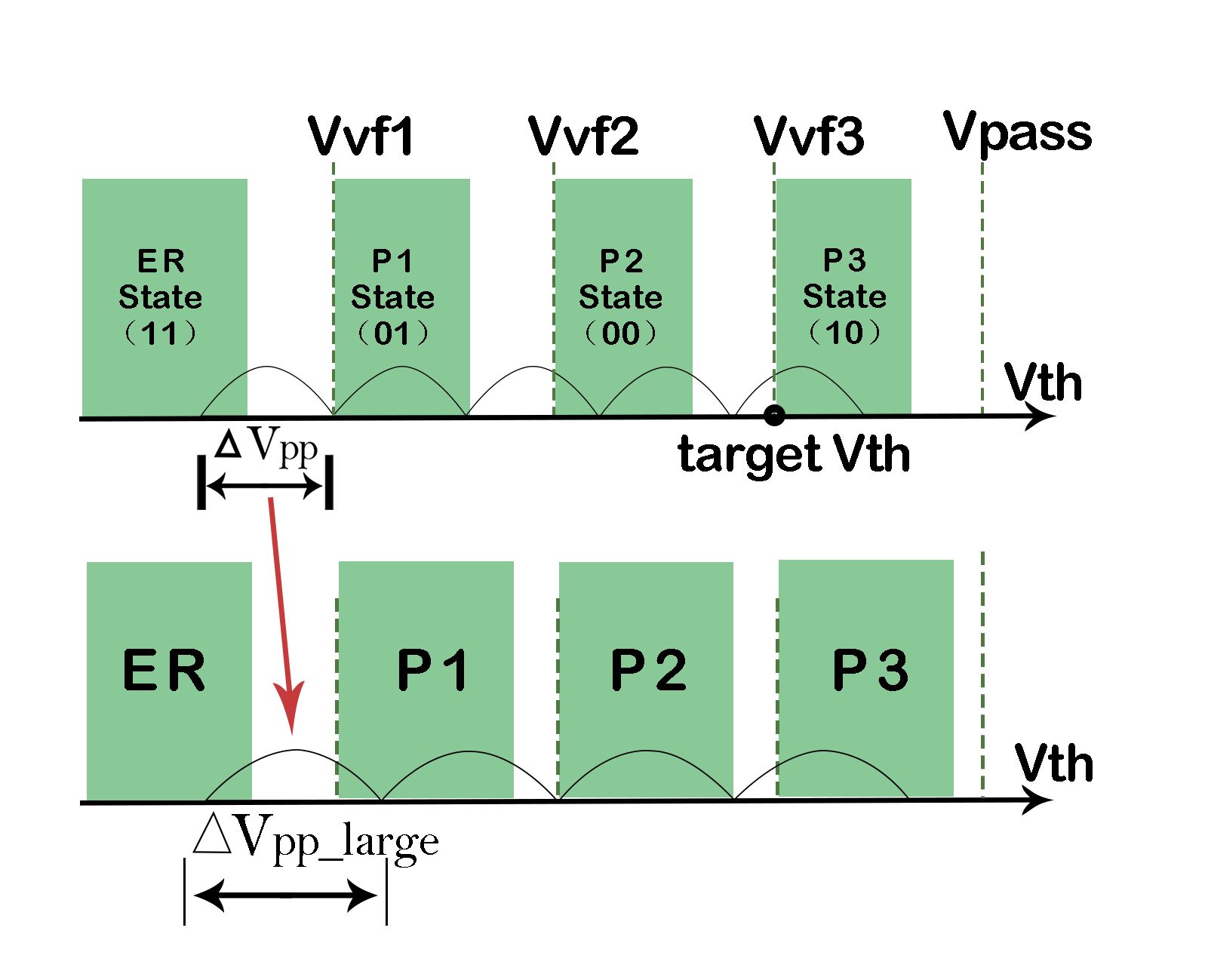


图 3‑1 使用更大编程电压的ISPP方案

### 降低阈值电压

实现近似写入的另一种方法是，通过降低最大阈值电压Vmax\_th，可以降低目标电压电平Vth\_small和缩小保护带，并使用与传统方案相同的编程增量电压ΔVpp，如图 3‑2所示。

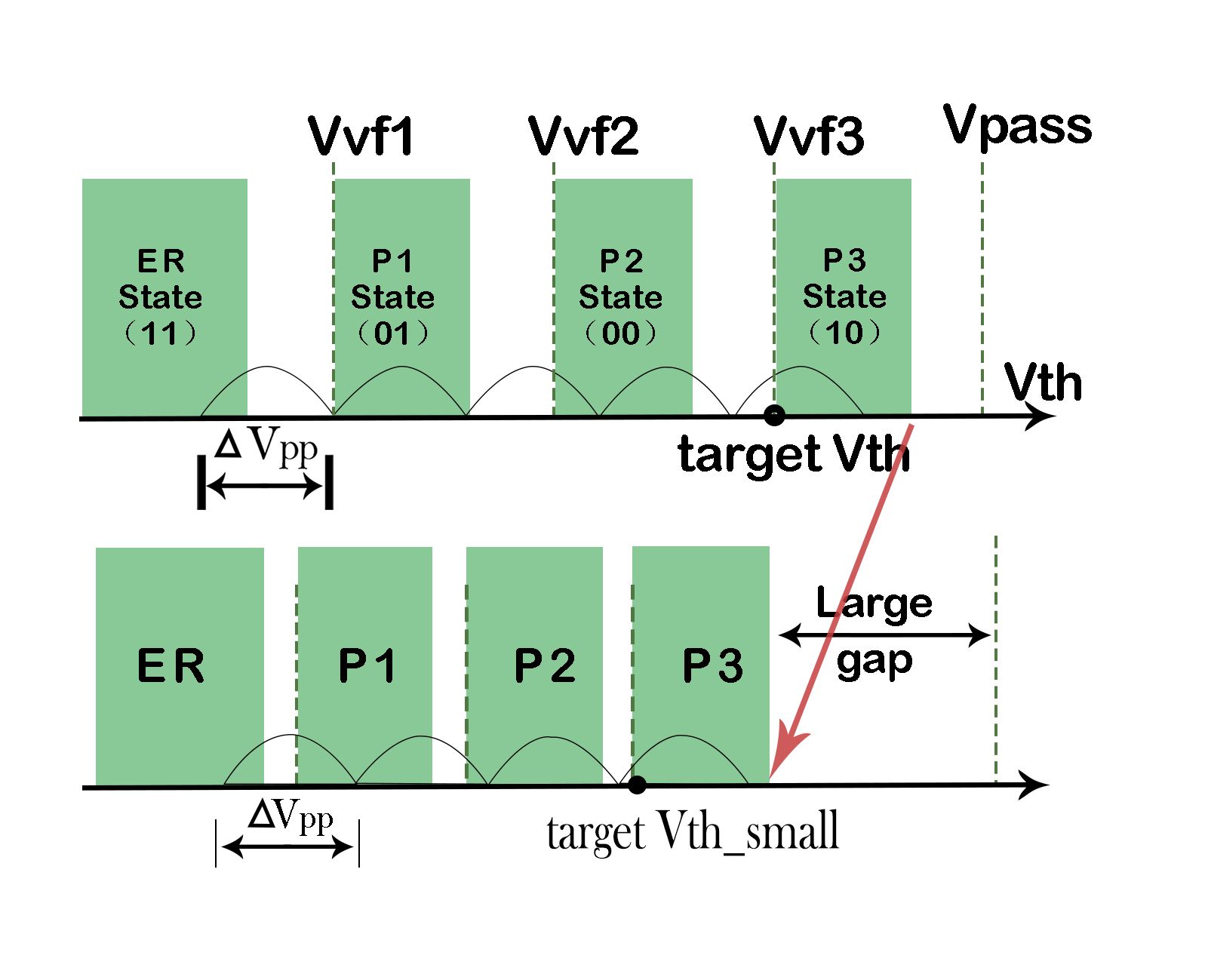


图 3‑2降低阈值电压的ISPP方案

基于降低阈值电压Vmax\_th的近似写入方案与使用较大ΔVpp的方案相比，降低Vmax\_th有助于减少编程时对物理块的干扰；有助于减轻近似页面的闪存单元的压力，研究表明，闪存的磨损效果取决于执行擦除操作时的擦除电压[15][16]，当阈值电压降低时，可以按比例降低擦除电压，从而可以改善P/E循环。

降低阈值电压的方案也吸引了其他研究工作的关注，这些工作已用于提高传统精确存储的性能、寿命中[16][17]。但是他们需要保守地降低阈值电压，使得闪存单元的原始误码率不超过闪存纠错功能的接受范围。在面向近似应用的近似写入情况下，可以更大幅度地降低阈值电压，在这种情况下，其原始误码率可能会超出闪存的纠错能力。

## 近似读取

读取速度主要受到错误代码纠正（ECC）的影响，闪存页面包含的位错误越多，即RBER更高，读取期间进行纠正的过程耗时就越长，读取速度就越慢。而应用程序对近似数据具有容错性。因此实现近似读取的简单方式就是，利用应用对近似数据的容错性，在不使用ECC纠错保护的情况下执行读取操作。

## 本章小结

本章主要介绍了近似存储的背景知识。首先是近似计算的概念以及近似存储于近似计算的关系。然后介绍了近似数据识别的方法，其中包含了程序分析法和故障输入法。接着介绍了近似写入的方法，其中包含有增大编程步长的方案和降低阈值电压的方案，还对比了两种方案的优势。最后还介绍了近似读取的方法。

# 数据布局方案设计

工艺偏差已经广泛的存在于闪存中，并对不同闪存块之间的可靠性造成很大的差异。在本章中，提出了一种利用近似数据以及闪存工艺偏差来提升存储系统性能的方法。首先提出了一种块速度检测方案，以确定块的写入速度和读取速度。然后，提出了一种数据分配方案以进一步利用块之间的可靠性差异来提高性能。

## 块的可靠性特征

闪存块的可靠性主要体现在可用编程擦除（P/E）周期数和块的误码率（RBER），而块的误码率又和很多因素相关。NAND闪存中的误码率问题是由于各种干扰造成的，例如NAND闪存的写入、读取、空闲或者闪存工艺偏差。这种干扰会在读取的参考电压上使得电压分布与原始电压分布产生偏移，如我们通过比较图 4‑1中原始分布和偏移分布的相对位置所示。由于这种偏移，一些单元被误读为与编程状态不同的状态。这种现象导致许多原始位错误。这些错误包括P/E循环错误，单元间编程干扰错误，编程错误，读取干扰错误，保留错误和工艺偏差错误。

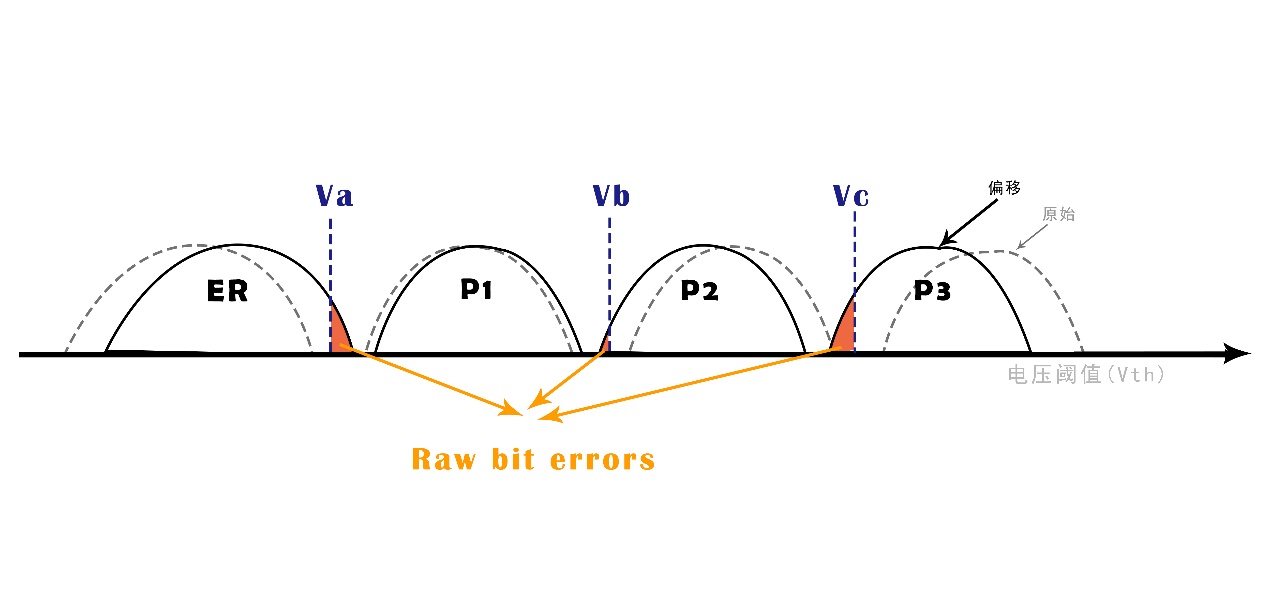


图 4‑1阈值电压偏移

在制造过程中，某些存储单元器件的参数，例如氧化层厚度，栅极宽度和长度，都呈现出很大的差异，从而导致闪存块之间的可靠性差异很大[26]。随着技术的持续发展和闪存位密度的提高，这种差异尤为严重。最新的工作已经在实际的闪存芯片上测试了工艺偏差，并提出闪存块之间的可靠性差异可能超过10倍[3][21][23]。闪存之间[20]、页面之间[25]和块之间[1][21][23]都存在差异。在本课题中，主要关注块间可靠性差异。由于工艺偏差在闪存块之间引入了很大的可靠性差异，因此必须根据其自身的可靠性来设计单个块的写入速度和读取速度。

## 写入速度

在本节中，首先讨论增量步进脉冲编程（ISPP）方案的ΔVpp与闪存块的强度之间的关系。然后提出了本文的写速度检测方案。

### ΔVpp与闪存块可靠性之间的关系

闪存中的数据通过ISPP方案以编程步长ΔVpp编程到闪存单元中，其中ΔVpp与写入操作的速度成正比。以前的研究表明，闪存是容易出错的设备，并且错误是由许多原因引起的。因此ΔVpp的设置要谨慎，以确保状态之间的保护带足够大。在当前的闪存中，ΔVpp是根据最坏情况设置的-可靠性最低的块达到其最坏情况的磨损状态，即报废阶段[18]。Pan等[19]建议根据已用完的状态为块设置ΔVpp，使用可用P/E周期数表示。当P/E周期足够大时，可以设置较大的ΔVpp，并且随着P/E周期的减少而降低。但是他们没有考虑每个块的具体可靠性。闪存工艺偏差在闪存块之间变得非常重要，因为这会导致闪存块的可靠性差异[4][8][9][10]。P/E循环引起的对可靠性较高块的磨损要小于对可靠性较低块的磨损，并且在存储期间，可靠性较高的块的电子泄露远小于可靠性较低的块。

基于上述观察，对于可靠性较高的块，可以通过缩小保护带来提高ΔVpp。随着闪存块逐渐磨损，ΔVpp相应降低，以保持闪存的可靠性。面临的挑战是在其当前磨损状态下检测块的合适写入速度。

### 写速度检测

在这项工作中，本文根据块的可靠性来确定该块的写速度。基本思想是将闪存块按照可靠性分类，共有N个类，将编程步长按照大小也分为N个类。在执行写入操作时，根据分配的块的可靠性来确定编程步长，如图 4‑2所示。当块的可靠性较高时，使用相对较大写入速度。随着P/E循环，闪存块的可用P/E周期数在不断减少，其可靠性也在慢慢降低，此时就会降低写入速度。

手机屏幕的截图

描述已自动生成

图 4‑2可靠性与编程步长对应

## 读取速度

块的读取速度主要受到ECC纠正耗时的影响，纠正过程耗时越长读取速度越慢。闪存页面包含的位错误越多，读取期间进行纠错的过程越长，读取性能就越低。因此块的读取速度与原始误码率（RBER）有关，当块的可靠性较高时，其RBER较低，读取速度较快；反之，当块的可靠性降低时，其RBER变大，读取速度降低。

## 数据分配

在本节中，提出了一种利用近似数据和闪存工艺偏差的数据分配方案，其基本思想是使近似数据分配到可靠性较低的块，精确数据分配到可靠性较高的块。如图 4‑3所示。

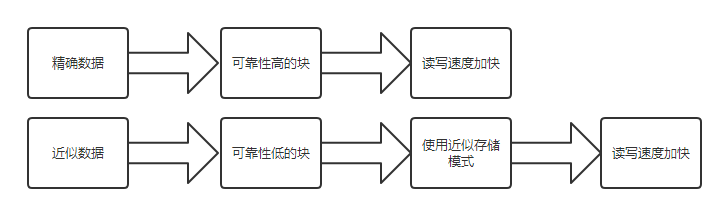


图 4‑3数据分配对应图

对于精确数据而言，一方面，存储在可靠性较高的块中，可以使用更大的编程电压ΔVpp，从而加快写入速度；另一方面由于数据存储在可靠性较高的块中，其误码率（RBER）较低，读取操作耗时较少，使得读取速度更快。对于近似数据而言，近似数据使用近似存储模式，可以利用应用程序对近似数据的容错性，使得其输出质量可以接受。这样一来闪存整体性能都得到了提升。

## 设计中考虑的制约因素

近似应用中存在大量的近似数据，闪存工艺偏差造成闪存块之间存在明显的可靠性差异。在考虑了上述的两种因素后，本文设计的数据布局方案中，物理页面读写比传统方案中的访问等待时间更短，因此消耗的能源更少；对近似页面进行编程以及对存储精确数据的高可靠性的块中的页面进行编程时，本文采用增大了编程步长ΔVpp的方法，使得每次写入需要的步骤数减少，从而使每次写入所消耗的能量少于传统方案所消耗的能量。

## 成本估算

这一节主要是估算本文设计所花的成本，选用的是Putnam模型，其方程为：L = Ck \* K1/3 \* td4/3

其中L源代码行数约为500，K整个开发过程所花费的工作量约为4/12，Ck技术状态参数约为8000，从上述方程加以变换，可以得到估算开发时间： td = [L3/(Ck3\*K)]1/4，将参数代入后计算的td=0.17。

## 本章小结

在本章中描述了本文数据布局方案的具体设计。首先阐述了速度检测方案，包括块的可靠性特征，块的写入速度，读取速度的检测。然后数据分配的具体设计进行了详细分析。其中在数据分配时，优先将近似数据分配到可靠性较低的块中，将精确数据分配到可靠性较高的块中。

# 数据布局方案实现

本章将对上文设计的数据布局方案在模拟器中实现。首先介绍了如何量化块的可靠性；接着为存取精确数据设置块的读写速度；再接着就是近似存储的实现；最后就是数据分配方案的实现。

## 块的可靠性量化

在模拟器中实现时，需要在块的结构体中添加一个变量blk\_reliability用于量化表示块的可靠性，以此确定块的读取/写入速度。其数值越低代表可靠性越高，并在模拟器初始化阶段，生成符合对数高斯分布的随机数初始化变量值。

由于少量P/E周期可能不会在闪存块上造成明显的可靠性下降，因此可以在每个特定数量的P/E周期之后进行闪存块的可靠性评估。如果闪存块的可靠性与上次测量获得的可靠性不同，就需要更新变量blk\_reliability的值。

## 块的速度设置

在这一节中，块的速度设置是针对于精确数据的，因为对于近似数据，采用的都是近似存储模式。

### 写入速度

在模拟器中，写入速度主要体现在数据从分组的寄存器写到目标物理页所消耗的时间tprog上。在传统模式下，tprog值是固定不变的，而在本文的实现中，其值与可靠性变量blk\_reliability的值成反比，即表示可靠性越高，变量blk\_reliability的值越小，写入耗时越短，写入速度越快。因此在执行写入操作之前，需要根据数据分配到的物理地址信息，获取数据所在的块的可靠性数值，再根据这个数值确定此次写入操作所支持的最大速度。

### 读取速度

在模拟器中，读取速度主要体现在数据从目标物理页中读到分组的寄存器所消耗的时间tR上。在传统模式下，tR的值固定不变，也就是假设了读介质阶段与ECC纠正阶段的耗时对于所有的物理块来说都是相同的。而在本文的设计中，tR与变量blk\_reliability的值成反比，即表示可靠性越高，变量blk\_reliability的值越小，读取耗时越短，读取速度越快。因此在执行读取操作时，需要通过数据的逻辑地址在地址映射表中找到其对应的物理地址，并将物理地址转换成用通道号、芯片号、晶圆号、分组号、块号、页号表示，然后获取数据所在块的可靠性数值，再根据这个数值确定从该块读取数据的读取速度。

## 近似存储

### 近似数据标记

本文在实现基于闪存的近似存储设计时，做出了一个假设，即标记近似数据与精确数据的标签跟随每一个IO请求一起传递到数据接口模块。

在固态盘模拟器中，负载请求文件既trace文件，用于记载特定的应用的所有的请求的抵达时间，请求的类型，请求的逻辑地址，请求的大小。3Dsim根据这个请求文件，可以模拟出这个应用在真实固态盘中的运行情况。在实现近似存储时，负载踪迹文件还需要添加一个参数，用于标识当前的I/O请求是否可以使用近似存储模式。也就是说，传给固态盘的I/O请求至少包括五项：

（抵达时间，请求的类型，请求的逻辑地址，请求的大小，近似标签）

并在闪存的页面信息中添加一位，用于表示当前页面所存储的数据类型。

在数据接口模块，从trace文件中读取应用请求，根据请求的抵达时间判断是不是会发生阻塞，并生成对应的请求结构体；根据请求的近似标签为其对应的请求结构体中的变量approxFlag进行赋值，该变量主要用于为下一阶段生成的子请求结构中的approxFlag赋值。并添加到请求队列上，等待后续处理。

### 近似写入

近似写入操作是针对于近似数据的，当请求中的数据是近似数据时，则执行近似写入操作。

写入操作位于闪存介质模块，在执行写入操作之前需要从写子请求队列中找到符合条件的子请求，然后为该子请求分配物理地址，最后再根据子请求结构中的近似标签决定使用的写入方式。如图 5‑1所示。

图片包含 游戏机

描述已自动生成

图 5‑1近似写入流程

### 近似读取

本文采用的近似读取方式是利用近似数据的容错性来实现，即当应用程序需要读取的是近似数据时，可以在不使用ECC纠错的情况下执行读取操作，进而提高读取性能。

读取操作位于闪存介质模块，但在执行读取操作前，需要从读子请求队列中查找符合条件的子请求，获取该子请求访问的物理地址信息，再根据子请求中的近似标签执行相应的读取操作。如图 5‑2所示。

手机屏幕的截图

描述已自动生成

图 5‑2近似读取流程

## 数据分配

在为数据分配物理地址时，本文采用与传统模式相同的条带化动态分配方式，如2.5所述。但与传统模式不同的是，在为数据分配物理块时的策略不同。

### 物理块分配

通过动态分配方式获取了channel,chip,die和plane之后，还需要为子请求分配合适的块。与传统模式不同的是，本文在plane结构体中添加两个变量reliable\_block,unreliable\_block，用以记录当前plane中可以直接分配使用的可靠性较高的块和可靠性较低的块的编号。而在每一个块的结构内都有一个free\_page\_num用于记录当前块的可用page数量。因此，在分配物理块号和物理页号时，首先需要根据子请求中的近似标签，确定子请求中的数据类型，如果是近似数据则分配可靠性较低的块；如果是精确数据则分配可靠性较高的块。然后根据块的free\_page\_num信息，判断当前块是否还有可用的page，如果有则在块中按照page的顺序进行写入操作，否则重新找到一个合适的块，并更新reliable\_block，unreliable\_block的值。如图 5‑3所示。

图片包含 游戏机

描述已自动生成

图 5‑3 分配物理块号

### 物理块管理

在闪存初始化，以及块的空闲页面不足时，需要查找合适的块为reliable\_block,unreliable\_block两个变量赋值。对于reliable\_block，需要在plane指向的block数组中查找可靠性最高的且free\_page\_num大于0的块，如图 5‑4所示；对于unreliable\_block，则需要在plane指向的block数组中查找可靠性最低的且free\_page\_num大于0的块，如图 5‑5所示。

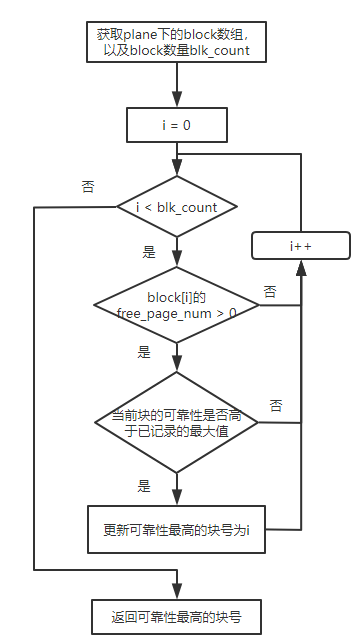


图 5‑4 查找可靠性较高的块

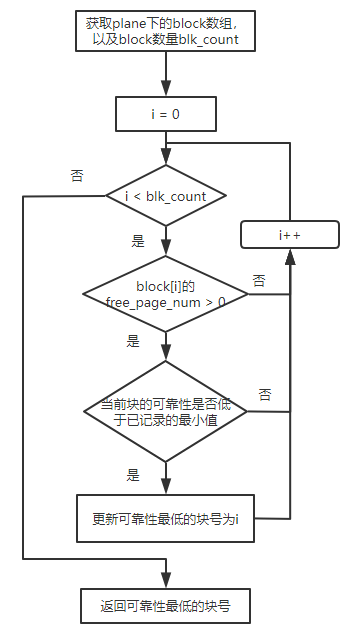


图 5‑5查找可靠性较低的块

### 垃圾回收

在上文中，对本文设计的新的数据分配方式进行了实现，相应的，需要修改垃圾回收机制，使其能够适应于新的数据分配方式。垃圾回收机制采用贪心垃圾回收算法，垃圾回收操作是在每一个晶圆下进行的，贪心算法是选择无效页最多的闪存块执行垃圾回收操作。

在本文设计的数据分配方式中，每个分组需要维持两个变量reliable\_block和unreliable\_block，表示当前可直接用于给近似数据或精确数据分配使用的块，也就是当前正在操作的块。由于闪存是不能够进行复写操作的，也就是说，如果请求需要重复写入相同的位置时，需要先将旧数据读取出来，然后重新分配空闲页，接着在SSD的buffer中修改数据，将修改后的数据写入新分配的空闲页，同时修改地址映射信息，并将旧数据所在的页面置为无效。在某一时刻，应用程序频繁执行更新写操作时，会使得当前正在操作的块中的无效页远高于其他块，从而变成垃圾回收算法选择擦除的对象，导致地址映射表中的逻辑页号与物理页号的映射关系出错。

因此，在垃圾回收的贪心查找阶段，需要避免将正在操作的两个块考虑进来，如图 5‑6所示。

手机屏幕的截图

描述已自动生成

图 5‑6垃圾回收贪心查找

查找过程如下：

1. 初始化erase\_block数组，数组元素个数为每个晶圆die下的分组plane数量，用于保存每个分组plane下无效页最多的块编号。
2. 从第一个分组plane开始，获取plane中的活动块编号。初始化变量block等于-1,invalid\_page等于0，分别用于保存block数组中无效页最多的block编号，及其无效页数量。
3. 在plane指向的block数组中查找编号不为活动块编号并且块的无效页大于invalid\_page的块编号，并更新invalid\_page和block的值。
4. 访问完block数组后，将block的值赋值到erase\_block中。
5. 重复以上操作，直到访问完当前晶圆下所有的分组。

## 本章小结

本章主要介绍了上文设计的数据布局方案在模拟器中实现。首先介绍了如何量化块的可靠性，以及更新其值；接着为存取精确数据设置块的读写速度；再接着就是近似存储的实现，包括标记近似数据、近似写入和近似读取；最后就是数据分配方案的实现，包括物理块分配、物理块管理和修改垃圾回收。

# 性能测试与分析

本章将对上文实现的利用近似数据以及闪存工艺偏差的数据布局方案进行测试，包括测试读取写入平均响应时间，IOPS以及吞吐量。读/写平均响应时间是评估本文的方案的性能的重要指标，此外还基于顺序读/写，随机读/写，混合随机读取的负载踪迹文件，评估本文实现的方案下的固态盘的IOPS以及吞吐量。正态分布用于模拟trace文件中近似请求的分布，对数正态分布用于模拟闪存工艺偏差带来的闪存块可靠性差异。

## 测试环境

测试平台配置为：CPU Intel i7 2.6GHz，12GB内存，编译器为Visual Studio 2019，固态盘模拟器模拟256GB的SSD。具体测试环境见表 6‑1 测试平台配置以及表 6‑2 固态盘基本参数配置。

表 6‑1 测试平台配置

|  |  |
| --- | --- |
| 配 置 | 参数说明 |
| CPU | Intel(R) Core(TM) i7-6700HQ CPU @2.6GHz |
| 内存 | 12GB |
| 操作系统 | Windows 10专业版 |
| 编译器 | Visual Studio 2019 |

表 6‑2 固态盘基本参数配置

|  |  |  |
| --- | --- | --- |
| 配 置 | 参数 | 参数说明 |
| channel | 2 | 固态盘2个通道 |
| chip | 4 | 每个通道下有4个chip |
| die | 4 | 每个chip下有4个die |
| plane | 4 | 每个die下有4个plane |
| block | 2048 | 每个plane下有2048个块 |
| page | 64 | 每个块下有64个page |
| page大小 | 16KB | page的大小为16KB |

用于测试的trace数据集来自3Dsim的wiki[11]，其中每个trace的写入请求中近似数据与精确数据的占比如图 6‑1所示，读取请求中的近似数据与精确数据的占比如图 6‑2所示。

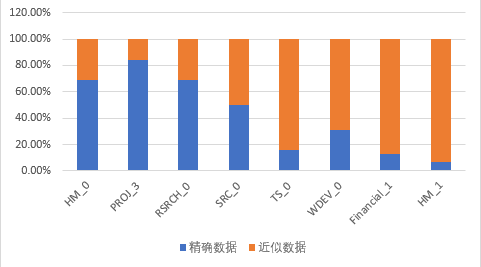


图 6‑1写入请求中近似数据与精确数据的占比

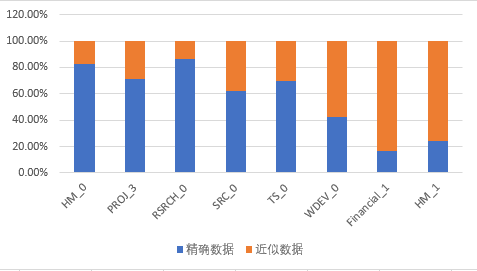


图 6‑2读取请求中近似数据与精确数据的占比

## 性能测试

在实验中，本文实现并比较了以下方案。

1. 传统方案（Traditional）。这是实现传统存储模式的方案，对块中数据进行编程时，使用传统的精确写操作对所有页面进行编程。它没有利用现代应用程序中的容错能力。
2. 近似存储方案（Approxmate）。这是仅仅实现了近似存储的方案。它使用一个较大的ΔVPP进行编程操作，是传统方案的1.25倍。
3. 考虑了闪存工艺偏差的方案（PV）。这是在近似存储方案的基础上，加上工艺偏差对块的可靠性的影响的方案。近似数据的读/写速度按照近似存储方案实行，精确数据的读/写速度则与块的可靠性有关。
4. 基于闪存工艺偏差和近似数据的数据分配方案（PV+Allocation）。这个方案是在2)3)方案的基础上加入数据分配方案。

### 读取平均响应时间比较

图 6‑3对比了不同方案的读取平均响应时间。将结果规范化，即以传统方案的结果为基准，计算不同方案的结果相对于传统方案的比例。从图中可以看出，在测试的数据集中，三个方案都提高了读取性能，其中PV+Allocation的方案效果更明显，与传统方案相比将读取平均响应时间降低了55.72%。读取性能的提高来自：⑴读取近似数据时，可以在不受ECC的纠正情况读取，减少了读取等待时间。⑵读取精确数据时，由于精确数据存放在可靠性较高的块中，其误码率（RBER）较低，读取等待时间较短。

图 6‑3 规范化的读取性能比较

### 写入平均响应时间比较

图 6‑4比较了不同方案的写响应时间，并将结果规范化。从图中可以看出，在测试的数据集中，三个方案都提高了写入性能，其中PV+Allocation的方案效果更明显，与传统方案相比将写入平均响应时间降低了38.93%。写入性能的提高来自：⑴写入近似数据时采取近似写入模式，减少了写入操作的耗时。⑵写入精确数据时，精确数据被分配到可靠性较高的块中，可以使用更大的ΔVpp，减少ISPP步长，从而减少了写入时间。

图 6‑4规范化的写入性能比较

### IOPS以及吞吐量比较

在这一项测试中，本文测试了四种方案在16KB随机读/写,顺序读/写，混合随机读写情况的下IOPS，以及吞吐量。

IOPS是指单位时间内系统能处理的I/O请求数量，数据吞吐量(Throughput)是指单位时间内可以成功传输的数据数量。

图 6‑5与图 6‑6为四种方案的IOPS和吞吐量比较图，从图中可以看到三种方案与传统方案相比，都能够提高IOPS和吞吐量，其中PV+Allocation方案的效果更明显，与传统方案相比将IOPS与吞吐量分别平均提高了35.41%和35.67%。

由于写入操作比读取操作更耗时，因此写入操作的IOPS以及吞吐量比读取操作低。在利用了近似存储和闪存工艺偏差的数据布局设计中，本文优先将高可靠性的块分配给精确数据，使得固态盘对精确数据进行读写操作的延迟大大降低；近似数据则优先分配到低可靠性的块中，使得固态盘中能有更多高可靠性的块的为精确数据服务，而近似数据利用近似的特征，即使其存储在低可靠性的块中，仍可以使用较高的读取写入速度。因此固态盘的整体读写延迟都大大降低，使得IOPS和吞吐量得到提高。

图 6‑5 IOPS比较

图 6‑6吞吐量比较

从总体性能评测结果看，本文最终设计的数据布局方案，基本达到了设计要求。

## 本章小结

本章给出了数据布局方案的测试过程，依次对读取平均响应时间、写入平均响应时间、IOPS、吞吐量进行了测试，并详细说明了测试的过程。最后对测试结果进行了分析。测试结果表明本文设计的方案能够提升固态盘的性能。

# 总结与展望

## 全文总结

在本文中提出了一种数据布局方案，这是一种面向近似应用的固态盘数据布局方案。本文通过增大ΔVpp，实现了近似写入；通过使近似数据在不受ECC纠错保护的情况进行读取，从而实现了近似读取。并基于近似数据和闪存工艺偏差，在本文中设计了数据分配方式，该设计优先考虑将近似数据分配到可靠性较低的块中，将精确数据分配到可靠性较高的块中，使得固态盘能够充分利用近似数据以及闪存工艺偏差的特征。将尽可能多的高可靠性块用于为精确数据服务，使得精确数据能够在保证输出质量的情况下，提高读写性能。而近似数据利用近似的特征，即使其存储在低可靠性的块中，仍可以使用较高的读取写入速度。从而，固态盘的整体性能都得到提升。然后，还修改了垃圾回收方案，使其适应于本文设计的数据分配方案。实验结果表明，本文的数据布局方案将闪存访问的读写响应时间分别平均提高了55.72%和38.93%，将IOPS与吞吐量分别平均提高了35.41%和35.67%。

## 未来工作展望

在未来的工作中，还应该考虑磨损均衡，当前的数据分配方案优先将数据分配到可靠性高与可靠性低的块，会对磨损均衡方案造成一定的影响。为了解决这个问题，可以对近似数据以及精确数据占用可靠性高的块和可靠性低的块的数量进行限制，减少对磨损均衡方案的影响，从而在增强性能的同时，延长闪存的寿命。基于三维闪存的固态盘将会大量投入使用，但闪存工艺偏差仍然广泛存在，因此考虑工艺偏差对三维闪存的影响，以及利用闪存工艺偏差来提高三维闪存的性能，是一个值得研究的方向，使得固态盘的性能和寿命都得到提升，从而让更多人可以使用更完善的固态盘。

致 谢

论文完成之际，首先要感谢我的导师崔金华老师，在老师人生中最重要的时刻之一，她仍能坚持辅导我的毕业设计工作。崔老师给我提供了一个很适合的课题，并在第一时间将所有资料发给我，让我能够充分了解课题的背景知识。在我遇到问题时，总是能及时的给出建议，并引导我想出解决方案。除此之外，崔老师还为我提供了论文以及答辩在内的各种指导，让我能顺利的完成这些工作。

同时我要感谢我的室友朱博抡，在我选题纠结的时候，给了我建议，让我能够顺利选到合适的题目。

我还要感谢我的另外两个室友龚艺涛和秦浩，我们三个选的都是崔老师的课题，平时互相监督，互相交流，在快乐中完成毕设。

我还要感谢我的学校和学院，让我度过了四年值得回忆的时光。

最后，我还要感谢这段疫情期间的所有抗疫英雄们，正是因为有了他们的无私奉献，我才可以放心的坐在家里完成毕业设计。

参考文献

1. L.Shi, Y. Di, M. Zhao, C.-J. Xue, K. Wu, and E. Sha. Exploiting Process Variation for Write Performance Improvement on NAND Flash Memory Storage Systems[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2016, 24(1):1-4.
2. A. Sampson, J. Nelson, K. Strauss and L. Ceze, "Approximate storage in solid-state memories[C]," 2013 46th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Davis, CA, 2013, pp. 25-36.
3. Y. Pan, G. Dong, and T. Zhang. Error rate-based wear-leveling for nand flash memory at highly scaled technology nodes. IEEE Transactions on Very Large Scale Integration (VLSI) Systems[J], 21(7):1350–1354, July 2013.
4. Y.-J. Woo and J.-S. Kim. Diversifying wear index for mlc nand flash memory to extend the lifetime of ssds[C]. In EMSOFT 2013, pages 1–10, Sept 2013.
5. J. Cui, Y. Zhang, L. Shi, C. Xue, W. Wu，and J. Yang. ApproxFTL: On the Performance and Lifetime Improvement of 3D NAND Flash based SSD[J]，IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems， 2018, 37 (19) :1957-1970.
6. J. Cui，Y. Zhang，W. Wu，J. Yang, Y. Wang, and J. Huang. DLV: Exploiting Device Level Latency Variations for Performance Improvement on Flash Memory Storage Systems[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems， 2018，37(8):1546-1559.
7. Q. Li, L. Shi, J. Yang, Y. Zhang and C. J. Xue, "Leveraging Approximate Data for Robust FIash Storage[C]," 2019 56th ACM/IEEE Design Automation Conference (DAC), Las Vegas, NV, USA, 2019, pp. 1-6.
8. M.-F. Chang and S.-J. Shen. A process variation tolerant embedded split-gate flash memory using pre-stable current sensing scheme[J]. IEEE Journal of Solid-State Circuits, 44(3):987–994, March 2009.
9. J. Chen and L. E. Cleveland. System for providing tight program/erase speeds that are insensitive to process variations[C], Aug. 11 1998. US Patent 5,793,249.
10. M. Meterelliyoz, A. Goel, J. Kulkarni, and K. Roy. Accurate characterization of random process variations using a robust low-voltage highsensitivity sensor featuring replica-bias circuit[C]. In ISSCC 2010, pages 186–187, Feb 2010.
11. zuoerfeng. 3Dsim wiki[DB/OL].https://github.com/zuoerfeng/3Dsim/wiki，2019-4-7
12. Q. Li et al., "Process Variation Aware Read Performance Improvement for LDPC-Based nand Flash Memory[C]," in IEEE Transactions on Reliability, vol. 69, no. 1, pp. 310-321, March 2020.
13. Song Liu, Karthik Pattabiraman, Thomas Moscibroda, and Benjamin GZorn. Flikker: saving dram refresh-power through critical data partitioning[J].ACM SIGPLAN Notices, 47(4):213–224, 2012.
14. Adrian Sampson,Werner Dietl,Emily Fortuna,Danushen Gnanapra-gasam,Luis Ceze,and Dan Grossman.Enerj: Approximate data typesfor safe and general low-power computation[C]. InACM SIGPLAN Notices,volume 46, pages 164–174. ACM, 2011
15. Jaeyong Jeong, Sangwook Shane Hahn, Sungjin Lee, and Jihong Kim.Improving nand endurance by dynamic program and erase scaling[C]. InHotStorage, pages 1–5, 2013.
16. Liang Shi, Kaijie Wu, Mengying Zhao, Chun Jason Xue, Duo Liu, andEdwin H-M Sha. Retention trimming for lifetime improvement of flashmemory storage systems[J].IEEE Transactions on Computer-Aided Designof Integrated Circuits and Systems, 35(1):58–71, 2016.
17. Dae-Seok Byeon, Sung-Soo Lee, Young-Ho Lim, Jin-Sung Park, Wook-Kee Han Pan-Suk Kwak, Dong-Hwan Kim, Dong-Hyuk Chae, Seung-Hyun Moon,Seung-Jae Lee,et al.An 8gb multi-level nand flashmemory with 63nm sti cmos process technology[C].InSolid-StateCircuits Conference,2005.Digestof Technical Papers. ISSCC. 2005IEEE International, pages 46–47. IEEE, 2005.
18. R. Micheloni,L.Crippa,and A.Marelli.Inside nand flash memories[C].InSpringer Dordrecht Heidelberg London New York, 2010
19. Y.Pan,G.Dong,and T. Zhang.Exploiting memory device wear-out dynamics to improve NAND flash memory system performance[C].InFAST 2011.
20. R. A. Pletka and S. Tomi ́c, “Health-binning: Maximizing the performanceand the endurance of consumer-level NAND flash[C],” inProc. ACM Int.Syst. Storage Conf., 2016, pp. 31–40.
21. Y.-J. Woo and J.-S. Kim, “Diversifying wear index for MLC NANDflash memory to extend the lifetime of SSDs[C],” inProc. ACM Int. Conf.Embedded Softw., 2013, pp. 1–10.
22. Y.Di, L. Shi, K. Wu, and C. J. Xue, “Exploiting process variation forretention induced refresh minimization on flash memory[C],” inProc. IEEEDes. Autom. Test Eur., 2016, pp. 391–396.
23. M. C. Yang, Y. H. Chang, C. W. Tsao, and P. C. Huang, “New ERA:New efficient reliability-aware wear leveling for endurance enhancementof flash storage devices[C],” inProc. ACM/EDAC/IEEE Des. Autom. Conf.,2013, pp. 1–6.
24. J. Cui, W. Wu, X. Zhang, J. Huang, and Y. Wang, “Exploiting latencyvariation for access conflict reduction of NAND flash memory[C],” inProc.IEEE Symp. Mass Storage Syst. Technol., 2016, pp. 1–7.
25. D. Wei, L. Deng, L. Qiao, and P. Zhang, “PEVA: A page endurancevariance aware strategy for the lifetime extension of NAND flash[C],”TVLSI ’16, vol. 24, no. 5, 2016.
26. A. Spessot, A. Calderoni, P. Fantini, A. S. Spinelli, C. M. Compagnoni,F. Farina, A. L. Lacaita, and A. Marmiroli, “Variability effects on the VTdistribution of nanoscale NAND Flash memories[C],” inIRPS ’10, 2010.
27. 吴宇,杨涓,刘人萍,任津廷,陈咸彰,石亮,刘铎. 近似存储技术综述[J]. 计算机研究与发展, 2018, 55(9): 2002-2015.
28. 胡洋．高性能固态盘的多级并行性及算法研究［Ｍ］．武汉：华中科技大学，2012.

装订毕业论文时应将任务书原件装订到最后



**本科生毕业设计任务书**

|  |  |
| --- | --- |
| 题 目 | 面向近似应用的固态硬盘数据布局方案的设计及实现 |
|  |  |

（任务起止日期：2020年1月1日～2020年6月10日）

|  |  |
| --- | --- |
| 院 系 | 计算机科学与技术 |
| 专业班级 | CS1610 |
| 姓 名 | 覃映超 |
| 学 号 | U201614795 |
| 指导教师 | 崔金华 |

教研室（系、所）负责人 2020年1月6日 审查

院（系）负责人 2020年1月9日 批准

|  |
| --- |
| 课题内容 |
|  |
| 课题任务要求 |
|  |
| 主要参考文献 |
|  |
| 同组设计者 |
| 无 |

**成 绩 评 定**

**指导教师评定意见**

装订论文时应将成绩评定页放最后，导师评语放前面！

一、对毕业设计（论文）的学术评语（应具体、确切、实事求是）

|  |
| --- |
|  |

二、对毕业设计评分

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 评分项目  (分值) | 调研论证  (10分) | 外文翻译  (5分) | 设计(论文)撰写质量  (10分) | 学习态度  (10分) | 基本理论和基本技能  (50分) | 创 新  (15分) | 合 计  (100分) |
| 得分 |  |  |  |  |  |  |  |

指导教师签字： \_\_\_\_ **崔金华** \_\_\_\_\_ 年 月 日

**答辩小组评定意见**

答辩小组意见和综合评定页放最后，双面！

一、评语（根据学生答辩情况及其设计（论文）质量综合评价）

|  |
| --- |
|  |

二、评分

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 评分项目  (分值) | 答 辩 情 况 | | 论 文 质 量 | | 合 计  (100分) |
| 答辩情况  (15分) | 回答问题情况  (25分) | 规范要求与文字表达  (20分) | 学术水平  (40分) |
| 得分 |  |  |  |  |  |

答辩小组长签字：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 年 月 日

**毕业答辩及成绩评定说明**

1. 毕业答辩
2. 答辩前，答辩小组应详细审阅每个学生的毕业设计（论文），为答辩做好准备。
3. 严肃认真组织答辩，开好答辩会。
4. 指导教师应参加所指导学生的答辩会，但评定其成绩时宜回避。
5. 答辩中要做好记录以供成绩评定时参考。
6. 成绩评定
7. 答辩前每个学生都要将自己的毕业设计（论文）在指定时间内交给指导教师，由指导教师审阅，写出评语并预评分。
8. 答辩工作结束后，答辩小组应举行专门会议进行讨论，在参考指导教师预评结果的基础上，结合学生毕业设计（论文）质量和学生答辩情况，综合评定每个学生的成绩。
9. 院（系）对专业答辩小组提出的优秀和不及格的毕业设计（论文），要组织院（系）级答辩，最终确定成绩，并向学生公布。
10. 各专业学生的最后成绩应符合正态分布规律。
11. 请用蓝、黑钢笔手写或五号宋体字编辑，签名须手写，A4纸双面打印。

**毕业设计（论文）成绩评定**

|  |
| --- |
| 班号：**CS1610** 学生姓名：**覃映超**  综合成绩：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_分（折合等级\_\_\_\_\_\_\_\_\_\_\_\_\_\_）  答辩小组长（签名）：\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 年 月 日 |