芯片防护 ECC：一种高效、低延迟的方法

Tanj Bennett
Avant-Gray LLC
IEEE 高级会员

电子邮箱：tanj@avant-gray.com

1. 摘要

Chip Guard 是一种新型符号纠错码方法，可适配各种数据突发大小和可靠性等级。文中介绍了一个针对 DDR5 的具体版本。该版本采用 DDR5 常见的配置：8 个数据芯片，外加 2 个用于纠错码和元数据的芯片，每个芯片支持 64 位突发，从而可靠地支持整芯片纠错，并具有较高的健全性（即报告不可纠正故障的能力）。该方法可支持不同数量的元数据位，并在可靠性和健全性之间进行明确的权衡。该方法应能纠正单个芯片的所有有界故障[*1*]；对于单个芯片中的无界故障，纠正失败的概率低于 $1/10^{12}$；对于影响多个芯片的未纠正故障，检测失败的概率低于 $1/10^{12}$。

2. 引言

DDR5内存旨在支持对可靠性要求较高的大容量内存系统。制造测试会筛除初始缺陷，并通过映射备用资源来替换缺陷，或直接淘汰芯片。然而，芯片投入使用后的数年间仍会出现新的缺陷。DDR5芯片在每个芯片中都内置了单比特纠错机制 [*2*]，在芯片的使用寿命期间，这种机制可能会纠正 95% 的错误。其余未纠正错误几乎全是多比特故障。这些故障由字线等结构中的缺陷引起；字线采用所有特征中最精细的分辨率绘制，并与多个存储单元相互作用。

现代服务器可能需要在生产环境中连续运行 5 年或更长时间环境中可能需要连续运行 5 年或更久。每个 CPU插槽可能由数百个 DRAM 芯片支持。现代服务器采用纠错码机制，可以纠正单个 DRAM 芯片中发现的任意或全部错误。对单个 DRAM 中多个比特的纠正，加上其他芯片上 DRAM 内部对任何单比特错误的纠正，即使在大型内存中，也能高度确保设备正常运行。操作系统或虚拟机监控程序最终应使用备用资源替换故障位置，但在此之前，纠错码对于确保应用继续运行，以及让数据能够安全读取并复制到新的备用资源中至关重要。

还有一些其他故障模式可能会翻转来自多个芯片的位。这些故障最可能是瞬态干扰，例如电源中的电源毛刺，或附近源产生的电磁干扰作用于电路和导线。这些罕见且无法纠正的错误不应被悄然放过。

这要求内存容量大且可靠性要求高的系统使用一种纠错码，既能纠正单个 DDR5芯片提供的整个64位符号，又能在出现更严重且无法纠正的错误时，以很高的健全性进行报告。

3. 背景与动机

现代服务器最多可配备 12 个 DDR5通道，每个 DDR5 通道包含两个可独立运行的 DDR5 子通道。每个子通道的每次存储和加载都需要纠错码，因此希望纠错码逻辑电路尽可能小，并尽量降低每次操作的功耗，因为需要使用该功能的多个实例。

纠错码逻辑电路最好具有低延迟，包括需要执行纠正时也是如此。低延迟纠正使设计能够采用固定路径，避免因执行纠正时延迟不同而增加流水线复杂性。

完整的 DDR5 DIMM 设计采用 10 个 DRAM 芯片，提供 512 个数据位和额外的 128 位。这 128 位是宝贵的资源，可用于存储元数据。理想情况下，ECC 方法应允许将其中一些位用于其他用途，同时仍能达到较高的 ECC 可靠性和健全性标准。

现代服务器可以充分利用与每条缓存行关联的元数据。用途包括（但不限于）标记因先前发生的不可纠正错误而失效的内存位置、缓存一致性提示 [*3*], 和能力位 [*4*]。

检测无法纠正的错误非常重要。这些错误可能由多个芯片的电源或信号受到物理干扰导致，也可能由行锤攻击（干扰）等效应引起；随着未来 DRAM 芯片的发展，这类效应可能会变得更加常见。要确保健全性，就必须以极高的概率检测到这些不可纠正故障。

DRAM 是一种非常可靠的技术，但当每个芯片扩展到数十亿位、系统包含数百个芯片时，就会出现错误。其中大多数是由 DDR5 DRAM 内部纠正的单比特错误，但有些错误源于会影响多个比特的故障。纠错码应能纠正任意单个芯片导致的所有故障，并检测超出单个芯片范围的未纠正故障。

服务器中 DDR4 通常使用的纠错码是 Reed-Solomon 码 [*5*] ，但在保证健全性的同时，无法容纳太多架构位。将这种方式扩展到 DDR5，会浪费 128 位的元数据资源。本文介绍了一种新颖的替代方案，结合使用奇偶校验和签名。随着纠错码资源增加，这种方法的效果会更好。有 128 位可用时，与不使用纠错码相比，未纠正的单芯片错误可减少 10^{12} 倍，同时未被检测到的未纠正多芯片错误的概率也可降低 10^{12}, ，并且仍可保留 16 个元数据位用于其他架构用途。也可以采用其他元数据位分配方案，其可靠性各不相同；除非另有说明，本文讨论的是使用 16 个元数据位的配置。

4. 方法

每个 DDR5 子通道使用 10 个芯片，每个芯片在一次传输中存储并返回 64 位（图 1）。其中 8 个芯片包含该次传输的 512 位用户数据。

该方法指定一个芯片作为简单的奇偶校验芯片，使用 64 位。它对前 9 个芯片执行横向奇偶校验，因此，若 B(c,b) 表示芯片 c 的第 b 位，则

$$Parity(b) = XOR(B(c,b) : c \in \{0..8\}) : b \in \{0..63\}$$

奇偶校验芯片是 10th 芯片。

当故障发生在单个芯片上时——这正是需要纠正的故障模型——奇偶校验会给出一个伴随式，其内容与错误位相同，但不会指出是哪个芯片导致了这些错误。

第二项要求是确定哪个芯片发生了故障。失效模型（“9 个芯片中的任意一个芯片上最多纠正 64 位”）为我们提供了大量擦除信息。可能的擦除位置只有 10 个，每个芯片对应一个（包括奇偶校验芯片）。

通过为前 8 个芯片中的数据构造伪随机签名，并将其存储在 9th 芯片中，来确定位置。该芯片有 64 位，因此，如果其中 16 位用于 ECC 以外的特征，就还剩下 48 位用于签名。随后将这 16 个特征位视为数据，并将其纳入构造签名的源位中。

签名函数可以是将源位映射到签名的任意伪随机映射，但某些性质有助于提高效率。每个数据位都有独特的映射。最终签名的构造可以分离进行：如果将任意一组数据位拆分成若干部分，那么

$$Sig(A,B) = Sig(A) \text{ op } Sig(B) = Sig(B) \text{ op } Sig(A)$$

$$Sig(A,B,C) = Sig(A+C) \text{ op } Sig(B)$$

其中，“op”是用于重新组合分别构造的签名的一种运算。

错误检测和纠正过程如下：

- 如果从双列直插式内存模块加载的突发数据在奇偶校验重建中匹配，并且在签名重建中也匹配，则将数据视为正确并直接传递，无需纠正。- 如果签名重建或奇偶校验重建与从 DRAM 读取的内容不匹配，那么

奇偶校验重建中检测到的位翻转（可能没有）将成为纠错综合。利用该伴随式，分别翻转 10 个芯片中对应的位。如果 10 种重建结果中恰有一种与签名匹配，则将翻转后的芯片值视为纠正结果，并作为成功结果传递。该方法依赖于签名的可分离性。

- 如果针对每个芯片重新计算奇偶校验伴随式仍无法修正签名，或者有两个或更多芯片都能修正签名，那么该故障便无法纠正。

值得仔细考虑几种特殊情况：

- 如果加载数据时没有错误（奇偶校验匹配且签名匹配），则数据和元数据位会作为正确值传递。正确的加载绝不会出现假阴性。

- 如果单个数据芯片发生故障，且没有其他故障，那么奇偶校验翻转将纠正该芯片，并且使用纠正后的芯片进行签名计算时，签名会匹配。如果只有该芯片能使签名匹配，则接受此次纠正。将翻转伴随式应用于另一芯片后也得到48位匹配签名的概率极低，但并非为零。在这种情况下，有两种可能的纠正方式，因此纠正失败（检测到故障）。第7节对此有更详细的说明。

- 如果签名显示没有错误，但加载后的奇偶校验不匹配，则故障可能出在奇偶校验芯片上。另一种可能是，数据或元数据中的翻转随机产生了48位零值签名。可以通过运行奇偶校验伴随式，为数据和元数据生成签名，以区分这两种情况。如果除了奇偶校验芯片外没有其他候选项，那么奇偶校验会自行纠正，数据和元数据仍是安全的结果。如果将奇偶校验翻转应用于其他9个芯片之一时，会生成48位零值签名，那么该故障无法纠正（因为它会与奇偶校验芯片产生别名），并会作为此类故障报告（检测到）。

这概括了该方法的基本原理。它优于单符号纠错、多符号检测（SSCMSD）方案 [*6*] ，因为 DDR5 中的片上 ECC 会纠正每个芯片中的单比特错误。现在我们来看看签名构造和增强方面的一些改进。

5. 一种实用且快速的签名

该签名需要接近密码学级随机性，同时易于快速计算。

在一种具体实现中，所选签名是比特排列的异或（“op”是异或）。对于数据芯片 $c \in \{0..7\}$，签名中的 8 位分别用于表示每个数据位：

$$S(c, b) \in P(48, 8) : c \in \{0..7\}, b \in \{0..63\}$$

其中，P(48,8) 是从 48 位中选取 8 位所构成的所有排列的集合。每个 S(c, b) 都是唯一的。

每个元数据位获得 19 位签名：

$$S(8, b) \in P(48, 19) : b \in \{0..15\})$$

所有签名经过异或后，形成完整的 48 位签名。

由于异或的性质，这种映射是可分离的。这一点将用于优化检测和纠正。

这种签名在逻辑电路中速度很快，每个签名位只需经过 7 层深的异或树即可生成。它具有所需的可分离性。10 个芯片中任意一个的签名都可以单独计算，然后将这些签名异或在一起，形成完整签名。检查签名是否已纠正时，只需要奇偶校验综合征，而不需要芯片的底层数据。这意味着纠正阶段只需执行 9 次单芯片纠正计算（奇偶校验芯片不参与签名计算），每次计算都使用相同的奇偶翻转模式，然后将每次计算的结果与签名综合征进行比较。奇偶校验综合征一旦就绪，就可以并行处理所有芯片，总体复杂度与额外计算一个合并签名的复杂度大致相同。

这并非唯一可行的签名方法。例如，我们曾考虑过循环冗余校验。尽管“op”更为复杂，但可分离性依然成立。然而，随机映射与异或相结合，能实现已知最简单的方案。

6. 强化签名

潜在弱点源于这样一种巧合：一组签名可能与另一组错误的签名相同，即发生别名冲突。需要关注的弱点包括：

- 由于芯片间发生别名冲突而无法纠正的单芯片错误 - 由于多个位的签名组合后相互抵消为零，因而未被检测到的不可纠正错误

签名从 P(48, 8) 种可能性中选取，共有 377,348,994 种不同的可能，因此即使只有一位发生别名冲突的风险也很低。通过穷举搜索碰撞并选择新的模式，可以大幅降低别名冲突的风险。对于给定的签名分配，会针对所有物理上最可能出现的单芯片模式进行搜索。例如，一项测试可以检查任意芯片上任意两个 DQ（数据引脚）故障的所有组合。每个 DQ 有 16 个可翻转的数据位，而 x4 DDR 芯片的 4 个引脚有 6 种可能的配对方式。DDR5 供应商设计芯片时，会确保最常见的多位故障原因属于“有界故障”，其影响范围仅限于两个 DQ。因此，与其他故障模式相比，这种模式在物理上更可能出现。它包含 6 x (2^{32})，约 257.6 亿次试验，并在 10 个芯片上重复进行。在模拟中运行 2580 亿次试验只需几天时间。结果要么是所有模式均不存在别名冲突，表明所选模式非常理想；要么是发现别名冲突，此时该方法会用新的排列替换映射中的部分内容，并重新开始检查。

生成很可能表现良好的模式是可行的，因此经过少量试验，就能找到一种模式分配，使 DRAM 芯片上的有界故障始终不会发生别名冲突，并且始终能够纠正。

对于其余可能发生的故障，对每个芯片运行搜索以检查是否存在别名冲突也很有用：搜索范围包括最多 10 位的任意排列（不限定为 2 个数据选通信号），以及它们的逆排列（最多 10 位未翻转）。这种搜索也很可能成功，但可能会导致选择一些新模式并重新开始测试。

穷举搜索证明，剩余的不可纠正故障只能是物理上极不可能发生的无界故障，并且必须包含大量位翻转。

7. 未纠正的单芯片故障

如果证明所有最多有 10 位翻转或未翻转的模式都不存在别名冲突，那么唯一可能发生别名冲突的故障，其翻转位数必须在 11 到 53 位之间。由于这些故障各自会产生 8 个伪随机签名位，因此会在 48 位中至少造成 88 次位翻转。这接近密码学哈希的理想情况：签名中的任意一位都有 50% 的概率发生翻转。要让两块芯片之间出现别名冲突，必须使两块芯片的奇偶校验综合征中出现的相同 11 位或更多位翻转，生成相同的签名综合征。由于其中一块芯片始终是真正的故障源，因此有 9 种其他可能。这意味着，对于任何 11 位或更多位的单芯片错误，无法纠正的概率约为 2^{44} 分之一，不到 10 万亿分之一。

单个芯片的未纠正故障总是能够被检测到，不会悄然发生。这是因为奇偶校验芯片总会回显翻转的数据或特殊位，因此这些故障不可能在未被察觉的情况下发生。

总之，DRAM 芯片极少发生错误。其中，非有界故障错误更是罕见。签名机制仅会有 10^{13} 次中的 1 次无法纠正此类错误，从而极小幅度地增加不可纠正但可检测的内存故障的发生率。

8. 特殊位中的故障

特殊位是指未用于纠错码的元数据位，它们被分配了 19 位签名。这是因为它们的纠正机制较弱。普通数据位纠正可以利用 3 个芯片（数据芯片、奇偶校验芯片和签名芯片），而特殊位仅依赖两个芯片（奇偶校验芯片和签名芯片，后者同时也是特殊数据芯片）。

因此，很容易预测出会导致 9th 和 10th 芯片发生别名冲突的翻转模式。每个特殊位都有一个 20 位模式——包括其自身和 19 个签名位——这些模式的异或组合也会形成别名冲突模式。因此，奇偶校验芯片或签名芯片上有 65,535 种翻转模式彼此无法区分（两个芯片都同样可能是纠正位置），因此结果会被标记为未纠正故障。

尽管这些模式可以快速枚举，但它们仍是伪随机生成的，因此极不可能与任何实际的位翻转物理模式相匹配。采用替代值进行穷举搜索并重试

可确保任何不可纠正错误都不会被任意两个数据选通信号所限制，也不会有任何错误的位翻转/未翻转数少于 11。48 位签名中约有 16.7 万亿种 20 位排列，因此发生混叠或结果为零的概率与常规数据相比，同样降低了约 10^{13} 倍。

9. 多芯片故障

多芯片故障不会被纠正，但几乎总应被检测出来（具有较高健全性），而不应静默发生。

该目标不包括纠正多芯片故障，因为当操作系统或虚拟机监控程序使用备用资源妥善替换已知故障时，DDR5 DIMM 的设计应能使此类故障变得极其罕见；这正是现代服务器中可靠性、可用性和可维护性（RAS）的最佳实践。现存的、可能产生多芯片故障的故障模型包括行锤或干扰，以及电气干扰。这些因素会在一次突发内产生数量和位置均任意的位翻转；对于标准 DDR5 DIMM 和通道，目前尚无实用的纠正方法。因此，我们的目标是检测此类故障，避免它们被静默传播。

未检测到的多芯片错误最可能的模型是：不同芯片上有 N 对对齐的数据位，它们在奇偶校验中相互抵消，同时组合后得到零签名。很容易看出，N = 1 无法成立（因为每个位都有不同的签名）；而对于更大的 N，进行穷举搜索也相对简单。搜索到 N = 5 完全可行（如果找到零值，就用新模式重新生成），因此可以确认最多 5 对位总能被检测到。剩余故障在任何模式下至少会翻转 12 个数据位（N = 6），并影响 96 个签名位。由此达到了万亿分之一的稀有度目标。对简单组合进行穷举搜索，并在极不可能找到零值时重新开始，这种简单方法已经过测试，证明可行。

如果数据位没有配对，则至少需要 3 个芯片（数据、奇偶校验、签名），并且需要更多位（因为签名位翻转必须与奇偶校验位翻转相匹配）。对最多 6 位进行穷举搜索后发现，存在实用的设计，使得少量位翻转不会导致这种别名。

10. 设计目标已达成

所有有界故障都可以纠正。超出有界故障范围的单芯片故障未能得到纠正的概率小于 1 比10^{12}。所有未纠正的单芯片故障，无论多么罕见，都保证能被检测到。

多芯片故障无法纠正，且未检测到而通过的概率低于 10^{12} 分之一。

该设计需要约 5,000 个二输入异或门等效单元用于存储，需要约 11,000 个二输入异或门等效单元用于加载，此外还需要通过锁存来控制内存接口的流水线时序，并通过多路复用选择加载的原始结果或纠正结果。

11. 其他变体

如果所需的特殊位少于 16 位，则可以使用多于 48 位的签名。如果只需要 4 个特殊数据位，则可将每个数据位 10 个签名、每个特殊位 19 个签名映射到一个 60 位签名中，而逻辑电路的规模只需略微增加。这将使 10^{15} 中的质量因子提升至 1。

这种方法可能适用于其他 DRAM 系统。一个特别值得关注的规模是：在具有 64 字节缓存行的 CPU 上，采用“内联”LP-DDR5 [7] 时可用的 64 位往返元数据。奇偶校验签名方法可以支持一个 16 位的有界且对齐的纠正符号，使用 32 位签名，并剩余 16 位用于架构用途。符号纠正失败的概率约为千万分之一，同时健全性（未检测到无界错误）的概率也约为千万分之一。

另一种可能的变体是覆盖 9 芯片 DIMM 通道，以纠正相邻 DQ 中的任意有界故障。这些是按 16 位边界对齐的 32 位序列。每个芯片有 3 个这样的边界，因此需要检查 27 个擦除位置。这种有界故障纠正的可靠性会略低于完整的芯片纠正，其健全性水平可能是百万分之一，而非万亿分之一；不过，节省一颗芯片（9 颗而不是 10 颗）对某些应用而言或许是合理的。

12. 致谢

作者谨此感谢 Rivos Inc. 的 Eric Shiu、Tao Zhang、Greg Lucas、Tom Bielecki 和 Raymond Xiao，在芯片防护算法及其实现的开发与验证过程中给予的指导、见解和支持。

13. 图

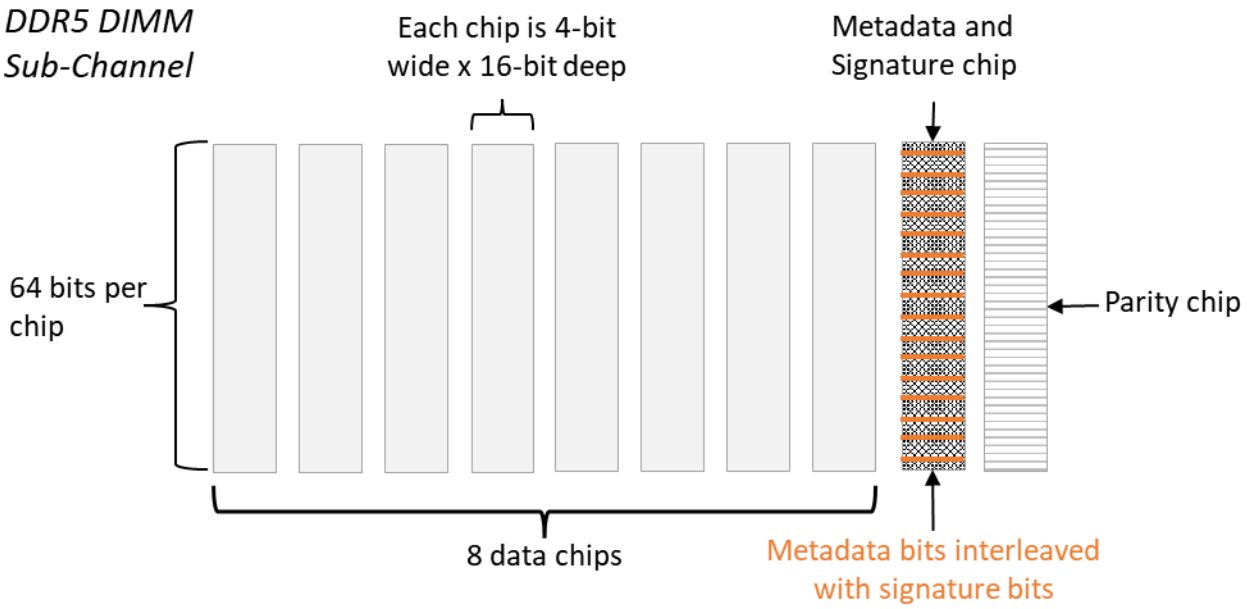

图*1*

14. 参考文献

[1] 通过限定DRAM故障提高内存可靠性：DDR5改进的可靠性功能 https://dl.acm.org/doi/10.1145/3422575.3422803[2] JEDEC发布新的DDR5标准 https://www.jedec.org/news/pressreleases/jedec-publishes-new-ddr5-standard-advancing-next-generation-high-performance[3] Knights Landing和Skylake Xeon处理器中的拓扑与缓存一致性 https://www.ixpug.org/documents/1524216121knl_skx_topology_coherence_2018-03-23.pptx 第7–8页[4] 能力硬件增强型RISC指令（CHERI） https://www.cl.cam.ac.uk/research/security/ctsrd/cheri/[5] Wu, Y.（2015）。里德–所罗[illegible] https://doi.org/10.1109/TCOMM.2015.2445759[6] Yeleswarapu, R. 与 Somani, A. K.（2021）。解决DRAM子系统中的多位/符号错误。PeerJ计算机科学，7，e359。https://doi.org/10.7717/peerj-cs.359[7] 平台上的DRAM纠错码内存，https://docs.nvidia.com/drive/drive_os_5.1.6.1L/nvvib_docs/index.html#page/DRIVE_OS_Linux_SDK_Development_Guide/Syste[illegible]m%20Programming/sys_components_dram_ecc.html