R-RS：一种面向 E 级计算的内存可靠性增强技术

高剑刚 石 嵩 郑 方
（国家并行计算机工程技术研究中心 北京 100190）

摘 要 存储器可靠性问题是构建 E 级计算系统的关键挑战之一. 存储器故障占计算机系统硬件故障的 40%以上，随着存储器数量增加、存储器密度扩展和接口速率提升，E 级计算机中存储器和访存传输通路的可靠性问题将会愈发严峻，传统的 SEC-DED 汉明码的纠检错能力难以满足 E 级系统高可靠性的需求. RS 码是一种纠错能力很强的多项式编码，可实现 Chipkill 技术，然而，可纠多符号错的 RS 码的译码电路复杂，直接应用于存储器领域较为困难. 本文提出了一种基于 RS 码和重传机制的内存可靠性增强技术——R-RS(Retransmission-RS)，通过精心挑选本原多项式和校验矩阵设计了具有低硬件实现开销的 RS 编码，并通过精细化电路设计实现了并行高效低延迟译码，提出了基于窗口保序的重传机制对传输链路上的偶发故障所致错误进行重传，R-RS 可纠正 4 个 8 位符号错，能够有效应对传输链路和存储器内部的随机单比特错、突发错以及传输链路偶发错误. R-RS 的冗余存储开销为 12.5%，性能开销是额外的 1 拍译码延迟，其面积仅占整个存储控制器的 3.5%，与同类别的 E-ECC 方案相比，其纠正双颗粒、三颗粒突发错的能力分别提升了 83.3%和 109.5%，而其误纠概率降低了 97.8%，利用存储器实际出错模型参数进行仿真，结果显示 R-RS 的平均纠错能力相较于 E-ECC 提高了 31%；R-RS 的重传功能在实际系统中使访存失效率降低了 42.1%. R-RS 应用在新一代神威 E 级计算机系统后，使系统的平均无故障运行时间增加了 35.3 倍，表明 R-RS 是一种有效的面向 E 级计算的内存可靠性解决方案.

关键词 存储器；E 级计算；可靠性；Chipkill；RS 码；重传
中图法分类号 TP311 **DOI 号** 10.11897/SP.J.1016.2023.00260

R-RS: A Memory Reliability Improvement Technology for Exascale Computing

GAO Jian-Gang SHI Song ZHENG Fang
(*National Research Center of Parallel Computer Engineering and Technology, Beijing* 100190)

Abstract Memory reliability is one of the key challenges for building Exascale supercomputers. The faults of memory account for more than 40% of the hardware faults in computer systems, and with aggressive storage density scaling in memory, lowering the supply voltage and the increasing of the memory interface rate and the number of memory devices, the reliability of memory device and transmission paths becomes more and more serious in exascale computing systems. The error-correcting capability of traditional SEC-DED(Single Error Correct/Double Error Detect) hamming code can detect and correct random single-bit error, but is not able to correct burst errors, which occurred commonly in supercomputer systems, so it cannot meet the requirements of Exascale supercomputers for high reliability. Reed-Solomon code is a class of symbol-based polynomial code with strong ability to simultaneously correct random errors and burst errors, which is widely used in hard disk protection and communication systems. However, the complicated decoding circuit of RS codes which can correct multi-symbol errors prevents itself from being applied to memory-related architectures. In this paper, R-RS(Retransmission Reed-Solomon), a memory reliability enhancement technology based on RS code and retransmission mechanism, is proposed.

收稿日期：2021-11-30；在线发布日期：2022-09-30.**高剑刚**(通信作者)，硕士，研究员，主要研究方向为计算机体系结构、高性能互联网络. E-mail：1370112205@139.com. **石 嵩**(共同第一作者)，硕士，助理研究员，主要研究方向为计算机体系结构. E-mail：shisong1990@163.com. **郑 方**，博士，研究员，主要研究方向为计算机体系结构.

《计算机学报》

By carefully selecting primitive polynomial and parity check matrix, a RS code which has the lowest hardware implementation overhead is designed, and the efficient and low delay parallel decoding is realized through refined circuit design. A retransmission based on window order-preserving is proposed to automatically retransmit the errors caused by the occasional faults in the transmission link. The R-RS is able to correct up to four 8-bit symbol errors, and can effectively deal with random single-bit errors or burst errors happened in transmission links or memory devices. We utilize Design Compiler to synthesize the proposed design in 28 nm technology. The R-RS has 12.5% storage overhead, and the performance overhead of the RS is 2-cycle latency for RS decoding. The area of the R-RS only accounts for 3.5% of the entire memory controller block. Compared with a recent RS code based memory protection counterpart, E-ECC technology, the correctable probability for 2-device burst errors and 3-device burst errors is improved by 83.3% and 109.5%, respectively, while the mistaken correction rate is reduced by 97.8%. Using an in-field memory fault model, we constructed an simulation platform in C language and simulated the correcting ability of SEC-DED, E-ECC, and R-RS, the results demonstrated that the average error correcting ability of R-RS is increased by 31% over E-ECC, and has an order of magnitude advantage over SEC-DED. The retransmission function of R-RS reduces the system error rate caused by memory system failure by 42.1% in real supercomputer system. The R-RS was successfully applied in the new generation of Sunway Exascale supercomputer system, and increased the mean time between errors of the whole system by 35.3 times, indicating that R-RS is an effective memory reliability solution for exascale supercomputer system.

Keywords　memory; exascale computing; reliability; Chipkill; Reed-Solomon code; retransimission

《计算机学报》

1　引　言

存储器(memory)可靠性问题是构建 E 级计算系统的关键挑战之一. 当前计算机的内存广泛采用 DRAM 技术，DRAM 芯片是计算机系统中数量最多的芯片. 研究表明，超过 40%的硬件故障与存储器有关[1]. 在 E 级计算机系统中，由于使用的存储器数量巨大，且存储器接口速率越来越高、存储器工艺缩放以及运行电压降低等众多原因，内存的可靠性问题更加严重. 如在新一代神威 E 级计算机系统中，使用了上千万个 DDR4 存储器颗粒，即使单个存储器颗粒的平均无故障时间达到 100 年，可以算出无保护措施下该 E 级系统存储分系统的平均无故障时间仅有约 315 s，这远不能满足中大型课题运算的时间要求，因此，在规模巨大的 E 级计算机系统中，提升访存容错能力是保证系统稳定可靠运行的关键.

为了解决存储器的可靠性问题，诸多技术如检查点技术[2]、清洗[3]、备份和重映射机制[4]、退回(retirement)机制[5]和 ECC 技术，其中 ECC 编码技术在可靠性、容量开销和性能之间有较好的平衡效果，是主流的存储器可靠性技术，工业界也普遍采用 ECC(Error Correction Code)技术来容忍存储器故障. 目前服务器领域采用的主流 ECC 技术是能够检双错、纠单错的(72，64) SEC-DED(Single Error Correct/Double Error Detect)汉明码[6]. SEC-DED 有较强的纠随机单比特错的能力，但是无法纠正超级计算机中较常出现的存储器颗粒故障导致的突发错误. 因此很多超级计算机采用纠突发错能力更强的 Chipkill 编码技术[7]. Chipkill 指可以纠正单个颗粒失效所致错误的编码技术，研究表明，相比于 SEC-DED，Chipkill 技术可以大幅降低计算机节点的失效率[8]. 然而，现有的 Chipkill 技术应用于 E 级计算机时还存在一些不足，表现为需要额外的读写操作从而造成性能损失[9-11]，或者纠随机错的能力不够[12]，或者误纠概率较高[13]，或者容量开销大[10]，难以满足 E 级计算机的性能和可靠性要求. Sridharan 等人[14]的存储器现场研究指出，E 级系统需要比 Chipkill 编码更强的可靠性方案.

RS(Reed-Solomon)码是一种基于符号的多项式编码，具有同时纠随机错和突发错能力较强的特点，广泛应用于硬盘阵列[15]和通信系统[16]中. RS 码可用于实现 Chipkill 技术，其纠错能力取决于符号

位宽和可纠错符号数量.在符号位宽与可纠符号数量乘积一定的情况下,符号位宽越大,纠随机错的能力越弱;反之,符号位宽越小,纠随机错的能力越强.RS 码等 BCH(Bose-Chaudhuri-Hocquenghem)码的硬件实现通常采用串行译码方式,延迟会非常大,如 Chen 等人[17]的 CARE 架构中,(573,512,6)BCH 码的译码延迟约为 30 拍,这将极大影响访存性能.因此,将 RS 码应用到存储控制器的关键问题是如何降低 RS 码的译码延迟.

此外,随着存储器接口访问速率的提升,命令、地址传输线的错误已不可忽视,如 DDR4、HBM 等接口协议已新增命令地址校验功能以应对该问题.Chipkill 等编码技术只能保护访存数据通路和存储器中的数据内容,不能保护命令和地址的传输.当命令和地址信号在传输过程中由于电磁干扰等原因出现偶发故障时,可能会导致多个 DRAM 芯片的数据出现错误,无法通过数据编码技术来解决.针对这种传输过程中的瞬态失效,通信领域使用包重发机制[18]来应对传输错误.然而存储器协议里缺乏通信领域丰富的握手机制,无法直接应用通信领域的包重发机制,需要自定义保序规则和保证性能,使得存储控制器中设计可靠低代价的重传机制具有挑战.

本文提出了一种基于 RS 码和重传机制的存储器可靠性增强技术——R-RS(Retransmission-RS),既能够纠正在存储器或传输链路上发生的数据错误,又能纠正传输链路发生的命令地址错误,以应对 E 级系统对存储器可靠性的需求.R-RS 设计了一种最利于硬件实现的以字节(8 位)为符号,将 64 符号(512 位)编码为 72 符号(576 位)的 RS 编码,可以纠正 4 符号(32 位)数据错误;精细化设计了高效的并行低延迟译码电路,能在 2 个存控时钟周期内完成译码;设计了基于窗口保序的重传机制,解决信号传输中的偶发失效问题.相比于通常的存储器保护技术,R-RS 创新提出了适用于存储器访问的重传机制,其编码的纠错能力优于同类的编码方案 E-ECC,应对双颗粒、三颗粒突发错的情形,纠错能力分别提升了 83.3%和 109.5%.R-RS 已经应用在了新一代神威 E 级计算机系统上,与使用传统 SEC-DED 编码相比,开启 R-RS 功能后,整个系统的平均无故障时间提高了 35.3 倍以上.

2 相关工作

存储器可靠性领域在过去十余年里有大量的研究工作[19],和本文相关性较高的是 Chipkill 编码技术.

SEC-DED 汉明码[6]最早被用于 DRAM 数据保护,其可以纠单错、检双错.随着高性能计算的发展,存储器的需求越来越大,对内存可靠性的要求也越来越高,SEC-DED 的纠错能力已不能满足诸如数据中心、超级计算机等高性能计算场景.在这种背景下,产生了能纠正单个存储器芯片错误的容错需求.IBM 最早提出了 Chipkill 的概念[20],其思想是将多个 SEC-DED 码组交织在一起(对于 x4 DRAM,需要 4 个码组,72 个 DRAM 颗粒),每个 DRAM 的每一位都划分到不同的码组.因此,Chipkill 具有完全纠正单个 DRAM 错误的能力.但是该方案一次需要访问 72 个 DRAM 颗粒,会带来明显的功耗和带宽开销.

为了减少 Chipkill 需要访问的 DRAM 数量和功耗开销,后续的内存采用了纠错能力更强的基于符号的编码方案[21].为了减少编码所需的 rank 数目,V-ECC[9]、LOT-ECC[10]、Multi-ECC[11]等采用了双层(Two-Tiered)ECC 编码方案.这类编码的核心思想是将编码信息分为两层,其中一部分校验码和数据放在一起,用于检错;另一部分校验码放在存储空间的其他位置,用于纠错,通过将两层编码结合以实现强大的纠检错能力.正常访问时,该方案只需读取数据和 tier-1 校验码,因此使用较少的 DRAM 就可以实现.然而,这类编码方案会浪费更多的存储空间,并且当数据写入时会产生较为严重的性能损失.

为了平衡编码的性能、纠错能力和面积开销,ARCC[22]和 Bamboo-ECC[23]等实现了纠错能力可调节的 ECC 方案.ARCC 在没有检测到错误时,只需激活一个 rank;当检测到错误时,需要打开两个 rank 以获得纠错能力.因此,ARCC 需要调整 LLC(Last Level Cache)的结构.Bamboo-ECC 以 8 位为符号,适用于 x4 DRAM,其纠错能力可从纠单比特错(存储开销为 3.1%)扩展到纠双芯片错(存储开销为 25%).

为了进一步提高纠错和检错能力,Yeleswarapu 等人[24]提出了基于 19 个 x4 DRAM 颗粒(16 个数据颗粒+3 个校验颗粒)的 SSCMSD 方案,其中 2 个校验颗粒用于 Chipkill 纠错,额外的 1 个校验颗粒用于 Hash 检错,从而实现可纠 1 个颗粒错、检多个颗粒错的能力.该方案需要的容量开销为 18.75%,高于主流的 12.5%.

在工业界,IBM ProteXion[25]通过冗余位转向(Redundant Bit Steering)技术将故障位重定向到备

《计算机学报》

份位，从而实现对故障位的恢复. Intel DDDC(Double Device Data Correction)①采用了芯片备份来提高可靠性，通过在每个 rank 中保留一个备份芯片，当存储器芯片被标记故障后，使用备份芯片代替故障芯片，从而实现纠正双芯片错.

RS 码因具有纠随机错和突发错的能力较强、误纠概率较低的特点，近年来也应用于 DRAM 存储器. Wang 等人使用 32 位符号的 RS 码实现了基于 18 个 x8 DRAM 颗粒的 Chipkill 编码[12]，其只能纠正单个颗粒的错误，无法纠正多个颗粒出现的随机错误. Chen 等人提出了基于 8 位符号的(36,32)RS 码的 E-ECC 编码[13]，实现了针对 x4、x8、x16 DRAM 的解决方案，可以纠 2 个随机符号错，但其误纠概率较高.

3　DRAM 特征分析

3.1　DRAM 内存结构

DRAM 是一种广泛用于计算机系统的存储器，DDR(Double Data Rate)系列的 DRAM 芯片一般有三种位宽：4 位(x4)、8 位(x8)和 16 位(x16). 多个 DRAM 芯片组成 DRAM 模组，商用 DRAM 模组的位宽通常为 64(带 ECC 校验的模组位宽为 72). 图 1 展示了一个由 9 个 x8 DRAM 构成的 72 位模组.

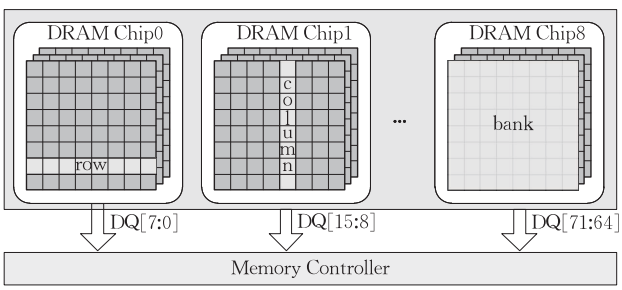


图 1　DRAM 通道示意图

如图 1 所示，每个 DRAM 芯片由多个 DRAM 阵列构成，称之为体(bank). 每个体由行(row)和列(column)组成. 在计算机系统中，这样一组 64 位(72 位)数据位宽的 DRAM 称之为一个通道(channel). DRAM 通道通过存储控制器连接 CPU. 在高性能计算领域，两个 64 位的通道可共用一个存储控制器，以锁步(lock-step)的模式运行①[26]，形成一个 128 位(144 位)的逻辑通道，以获得更大的访存粒度或者实现更可靠的纠错编码. 本文的 R-RS 即是建立在 144 位通道上.

为了实现高速传输，DDR 系列的 DRAM 采用了突发读写技术，即一次读(写)命令读出(写入)多拍数据. DDR3[27]和 DDR4[28]的突发长度均为 8，即一次读出 8 拍数据. 对于 144 位的通道，存储控制器一拍处理 576 位数据. 因此，一次访存操作可能出现连续的多位错，称之为突发错，从图 1 及访存行为可以看出，行故障、体故障、整芯片故障将会导致突发错，而单比特故障、列故障将导致随机错.

3.2　访存错误特征

DRAM 的访存错误可以从不同的角度进行分类. 首先，从访存过程来看，可以把访存错误分为存储器故障所导致的错误和传输链路故障所导致的错误. 对于存储器故障，根据故障的持续时间可以分为瞬时故障和永久故障；瞬时故障是可以复原的故障，而永久故障是存储器的永久性损伤，不可恢复. 根据故障的范围，可以分为单比特故障、行故障、列故障、芯片故障等. 本文将存储器错误分为单比特随机错和突发错两类，其中单比特错由单比特故障和列故障引起，其他故障将导致读数据发生突发错. 传输链路故障包括命令地址线故障和数据线故障，命令地址线故障可导致多个 DRAM 芯片的访问出现错误，而数据线故障只会导致相应 DRAM 芯片的访问出现错误.

Sridaran 等人[8]以及 Bautista-Gomez 等人[29]分析了 DRAM 的故障模型，其中 Sridaran 等人指出在 DRAM 的故障模式中，单比特故障占 49.7%，同时该研究指出与使用 SEC-DED 相比，使用 Chipkill 可以大幅降低由 DRAM 错误引起的节点失效率(仅为 SEC-DED 的 1/42). Bautista-Gomez 等人研究了无 ECC 保护的 DRAM 错误特征，分析了错误特征和其他因素对存储器故障的影响，具体包括：(1) DRAM 存储器错误具有很强的空间相关性和时间相关性；(2) 大多数字内的多比特错的错误位并不连续；(3) 温度和太阳位置等因素会影响 DRAM 存储器的错误率，多比特错发生在中午(太阳位置最高)的概率最大. 这些故障模型对 DRAM 的可靠性技术设计具有较好的指导意义，上述 2 个研究都说明了多比特错并不少见，Bautista-Gomez 等人的研究指出错误发生具有很强的空间相关性和时间相关性，说明需要编码同时具有较强的纠随机错误和突发错误的能力，以及，重传机制有助于应对错误的时间相关性. 本文根据 Sridaran 等人指出的失效模式建立模型以分析 R-RS 的纠错能力.

① Intel. Intel Xeon Processor E7 Family: Reliability, Availability and Serviceability: Advanced data integrity and resiliency support for mission-critical deployment. http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/xeone7-family-ras-server-paper.pdf. 2020-06-05

《计算机学报》

4 R-RS 可靠性增强技术

根据在历代神威超级计算机系统上观察到的错误特征和高性能计算对访存可靠性的需求，梳理出E级计算对访存可靠性的需求如下：

(1)可靠性技术主要针对运行过程中动态出现的错误，而非存储器的永久性故障；

(2)可靠性技术的引入不能影响正常访存请求的性能，尤其是造成带宽效率的损失；

(3)可靠性技术要兼顾突发错误(burst error)类型和单比特随机错误类型；

(4)可靠性技术要能同时应对传输链路上的失效和存储器颗粒的失效.

根据上述观察结果，SEC-DED由于纠检错能力弱，而检查点技术、备份和地址重映射机制、清洗和传统双层Chipkill技术由于带宽损失严重不适合E级计算这种高性能计算应用场景.因此，提出R-RS技术——基于RS码和重传机制的存储器可靠性增强技术以实现既能够纠正在存储器或传输链路上发生的数据错误，又能纠正传输链路发生的命令地址错误.该技术主要应对运行过程出现的动态错误，纠检错能力强，且不会对正常访存请求的带宽效率造成损失，以满足E级计算对内存可靠性的需求.

本文提出的R-RS技术适用于由x4或x8 DRAM芯片组成的以锁步模式运行的144位的DRAM内存通道，为便于描述，下文以x8位宽为例.

4.1 整体架构

R-RS的整体结构如图2所示，存储控制器中的5个白色方框是基本的存储控制器模块，事务处理模块负责访存请求的调度分发处理，命令生成模块负责DRAM协议管理，数据写入和数据读出分别负责数据的写入和读出通路，响应生成模块负责响应的生成.R-RS主要包含三个模块：重传模块、RS编码模块和RS译码模块.重传模块位于事务处理和命令生成模块之间，RS编码模块位于数据写入模块之前，RS译码模块位于数据读出模块之后.R-RS的整体流程为，事务处理模块发出访存请求后，访存请求进入重传模块进行缓存和保序管理，重传模块将请求转发给命令生成模块，产生DRAM命令序列发送给DRAM，然后，如果该请求是写请求则相应数据会在RS编码模块进行编码后经数据写入通路发送到DRAM，若DRAM在指定时间内没有返回校验错误则数据写入模块发送写完成标志给重传模块对缓存条目进行释放，写请求完成，若DRAM返回了校验错则重传模块进行重传；如果是读请求则相应数据从DRAM返回后会被RS译码模块译码，若译码没有错误或者错误可纠正则生成响应返回并通知重传模块释放对应请求条目，否则重传模块重传该请求.从图中可见，R-RS和右边的逻辑构成重传环路，实现了自动重传功能，对上层完全透明.

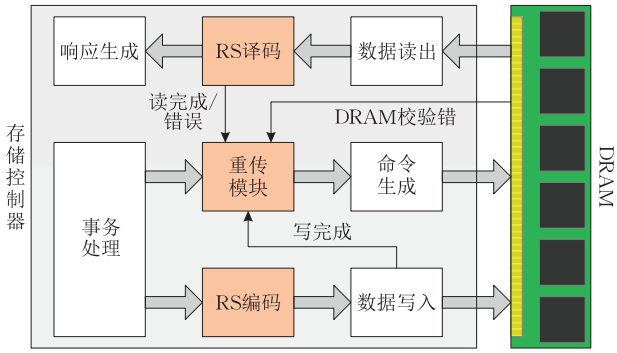


图2 R-RS结构框图

4.2 可纠4符号错的RS码

对于x8颗粒组成的144位通道系统，共18个颗粒(18×8=144)，其中16个颗粒作为数据颗粒，2个颗粒作为冗余颗粒存放校验位，存储控制器一拍处理576位(18×8×4)数据(每个颗粒32位数据).为了兼顾随机错和突发错，我们选取字节(8位)作为符号，设计一种将64符号编码成72符号的(72，64)RS编码，冗余8个符号(64位)，根据编码理论[30]，该RS码的最小距离为9，所以可纠正最多4个符号的错误，4个符号错可随机分布，自然地，可以纠正4个符号错发生在同一个颗粒的情形，即实现了Chipkill的功能.

在存储控制器中实现RS码的最大难点在于降低译码延迟.传统通信领域使用串行译码电路，但是该电路的延迟达数十拍，会在很大程度上增加访存操作的延迟.并行译码电路可以降低延迟，但是逻辑量巨大、设计困难，特别是纠3符号错以上的并行译码电路.为了解决这个问题，本文通过精心设计最利于硬件实现的RS编码，并且精细化电路设计，实现了低延迟的并行RS码译码方案.

4.2.1 RS码构造

RS码通常用多项式来描述，也可以用等价的二进制矩阵来描述[12].本文使用二进制矩阵来描述.

RS码的生成矩阵$\boldsymbol{G}$和校验矩阵$\boldsymbol{H}$分别如式(1)和式(2)所示，二者的关系满足$\boldsymbol{GH}^{\mathrm{T}}=0$.

$$\boldsymbol{G}=\begin{bmatrix} & \boldsymbol{X}_{0,0} & \boldsymbol{X}_{0,1} & \cdots & \boldsymbol{X}_{0,7} \\ & \boldsymbol{X}_{1,0} & \boldsymbol{X}_{1,1} & \cdots & \boldsymbol{X}_{1,7} \\ \boldsymbol{I}_{512} & \vdots & \vdots & & \vdots \\ & \boldsymbol{X}_{62,0} & \boldsymbol{X}_{62,1} & \cdots & \boldsymbol{X}_{62,7} \\ & \boldsymbol{X}_{63,0} & \boldsymbol{X}_{63,1} & \cdots & \boldsymbol{X}_{63,7} \end{bmatrix} \tag{1}$$

$$\boldsymbol{H}=\begin{bmatrix} \boldsymbol{I}_8 & \boldsymbol{I}_8 & \boldsymbol{I}_8 & \cdots & \boldsymbol{I}_8 & \boldsymbol{I}_8 \\ \boldsymbol{A}^{a_0} & \boldsymbol{A}^{a_1} & \boldsymbol{A}^{a_2} & \cdots & \boldsymbol{A}^{a_{70}} & \boldsymbol{A}^{a_{71}} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ \boldsymbol{A}^{7a_0} & \boldsymbol{A}^{7a_1} & \boldsymbol{A}^{7a_2} & \cdots & \boldsymbol{A}^{7a_{70}} & \boldsymbol{A}^{7a_{71}} \end{bmatrix} \tag{2}$$

其中矩阵 $\boldsymbol{A}$ 是一个 8×8 的二进制矩阵，可以通过本原多项式 $g(x)=x^8+\sum_{i=0}^{7}g_ix^i$，$g_i\in\{0,1\}$ 计算得出，$\{0,\boldsymbol{A}^1,\boldsymbol{A}^2,\boldsymbol{A}^3,\cdots,\boldsymbol{A}^{2^8-1}=\boldsymbol{I}_8\}$ 构成 GF(2^8)的性质.

$\boldsymbol{G}$ 中第 i 行($0\leq i\leq 63$)的 8 个待定矩阵($\boldsymbol{X}_{i,0}$，$\boldsymbol{X}_{i,1}$，…，$\boldsymbol{X}_{i,7}$)，都是上述 GF(2^8)中 8×8 的矩阵，计算方法如式(3)所示.

$$\begin{cases} \boldsymbol{I}_8=\boldsymbol{X}_{i,0}+\boldsymbol{X}_{i,1}+\cdots+\boldsymbol{X}_{i,7} \\ (\boldsymbol{A}^{a_i})^{\mathrm{T}}=\boldsymbol{X}_{i,0}(\boldsymbol{A}^{a_{64}})^{\mathrm{T}}+\boldsymbol{X}_{i,1}(\boldsymbol{A}^{a_{65}})^{\mathrm{T}}+\cdots+\boldsymbol{X}_{i,7}(\boldsymbol{A}^{a_{71}})^{\mathrm{T}} \\ \quad\vdots \\ (\boldsymbol{A}^{7a_i})^{\mathrm{T}}=\boldsymbol{X}_{i,0}(\boldsymbol{A}^{7a_{64}})^{\mathrm{T}}+\boldsymbol{X}_{i,1}(\boldsymbol{A}^{7a_{65}})^{\mathrm{T}}+\cdots+\boldsymbol{X}_{i,7}(\boldsymbol{A}^{7a_{71}})^{\mathrm{T}} \end{cases} \tag{3}$$

RS 编码方案需要确定本原多项式、生成矩阵和校验矩阵，对于硬件设计来说，应当选择使得硬件逻辑级数最少的方案以降低编码、译码延迟，Neuberger 等人[31]曾指出构造 RS 码时应选取生成矩阵中“1”的数量尽可能少的 RS 码，这是因为生成矩阵中“1”代表实际电路的门单元. 实际上，译码的逻辑量远大于编码的逻辑量，对于并行译码来说，应优先降低校验矩阵中“1”的数量. 因此，上述 3 个构件中，关键是确定本原多项式和校验矩阵，其中本原多项式决定了多项式乘法的操作数量，进而影响了编码和译码逻辑的复杂度，而校验矩阵中 1 的个数同样影响译码的开销和延迟.

对于本原多项式的寻找，首先根据算法 1 找出所有阶数为 8 的本原多项式，然后计算出这些本原多项式所构成的伽罗华域下乘法操作中生成各次项系数时参与异或操作的数量，其结果如表 1 所示，从表中可以看出本原多项式 $g(x)=x^8+x^5+x^3+x^2+1$ 对应的各次项系数异或操作的最大值最小，因此选择该本原多项式作为本编码使用的本原多项式.

算法 1.　本原多项式搜索算法.

输入：无

输出：本原多项式 $g(x)$的集合 GS

1. $g(x)\leftarrow x^8+1$;　　//记录 $g(x)$
2. $GS\leftarrow\{\}$;　　//记录 GS
3. FOR $i\in[0,2^7-1]$ DO　//搜索的空间：$g[7:1]$
4. 　$g(x)$的系数$\leftarrow${1'b1，i 的二进制表示，1'b1};
5. 　使用 $g(x)$的系数构造矩阵 $\boldsymbol{A}$;
6. 　$original\leftarrow 1$;　　//是否是本原多项式
7. 　FOR $j\in\{2^8-1$ 除了 1 与自身以外的因数$\}$ DO
8. 　　$\boldsymbol{B}\leftarrow\boldsymbol{A}^{j+1}$;
9. 　　IF $\boldsymbol{B}==\boldsymbol{A}$ THEN
10. 　　$original\leftarrow 0$;
11. 　　BREAK;
12. 　　ENDIF
13. 　END
14. 　IF ($\boldsymbol{B}==\boldsymbol{A}^{2^8}$) AND $original$ THEN
15. 　　$GS\leftarrow GS\cup\{g(x)\}$;
16. 　ENDIF
17. END
18. RETURN GS;

表 1　不同本原多项式的伽罗华域下乘法操作的异或操作数量

多项式	乘法操作中参与异或操作数量								
	x^7	x^6	x^5	x^4	x^3	x^2	x	1	最大值
$g(x)=x^8+x^4+x^3+x^2+1$	17	19	21	24	24	20	11	14	24
$g(x)=x^8+x^5+x^3+x+1$	19	23	28	15	16	20	24	15	28
$g(x)=x^8+x^5+x^3+x^2+1$	16	18	21	19	22	19	12	14	22
$g(x)=x^8+x^6+x^3+x^2+1$	21	23	19	23	28	13	15	18	28
$g(x)=x^8+x^6+x^4+x^3+x^2+x+1$	18	20	17	18	21	26	27	15	27
$g(x)=x^8+x^6+x^5+x+1$	23	26	25	14	16	19	22	20	26
$g(x)=x^8+x^6+x^5+x^2+1$	26	31	21	12	14	16	17	21	31
$g(x)=x^8+x^6+x^5+x^3+1$	27	32	20	17	20	14	18	22	32
$g(x)=x^8+x^6+x^5+x^4+1$	20	23	29	25	11	13	15	17	29
$g(x)=x^8+x^7+x^2+x+1$	33	11	13	16	20	24	10	28	33
$g(x)=x^8+x^7+x^3+x^2+1$	30	12	14	16	19	19	22	26	30
$g(x)=x^8+x^7+x^5+x^3+1$	28	21	23	23	26	16	20	24	28
$g(x)=x^8+x^7+x^6+x+1$	26	30	10	12	14	17	21	22	30
$g(x)=x^8+x^7+x^6+x^3+x^2+x+1$	28	26	14	17	21	12	19	23	28
$g(x)=x^8+x^7+x^6+x^5+x^2+x+1$	25	19	28	13	15	18	17	20	28
$g(x)=x^8+x^7+x^6+x^5+x^4+x^2+1$	18	21	25	30	23	27	14	16	30

得到本原多项式后，接着确定校验矩阵，原则是力求选出“1”的数量最少的矩阵，以减少生成伴随式 $\boldsymbol{s}$ 过程中异或操作的数量. 其方法是利用算法 2 从 0，1，…，254 中选择 72 个值作为式(2)中的 a_0，a_1，…，a_{71}，即 a_j 的选取原则是使式(2)中第 j 列中“1”的数量之和最少.

《计算机学报》

算法 2. 校验矩阵元素挑选算法.

输入:$\boldsymbol{A}$

输出:$a_0,a_1,\cdots,a_{71}$构成的数组 $count\_vec[71:0]$

1. DECLARE $count\_vec[255]$;
2. FOR $i\in[0,254]$ DO
3. $count\leftarrow 0$;
4. FOR $j\in[0,7]$ DO
5. $count\leftarrow count+\boldsymbol{A}^{ij}$ 中"1"的数量;
6. END
7. $count\_vec[i]\leftarrow count$;
8. END
9. $sort(count\_vec)$; //升序排序
10. RETURN $count\_vec[71:0]$;

最后根据 $\boldsymbol{H}$ 矩阵和式(3)计算出生成矩阵 $\boldsymbol{G}$. 通过该方法设计出的 RS 码的并行译码电路具有最少的逻辑门级数,从而具有最低的译码延迟.

4.2.2 RS 码高效译码电路

在通信领域里,RS 采用串行的迭代译码算法,延迟达到了数十拍,本文在电路层面通过精细化电路设计,实现了并行低延迟译码.

设编码前的数据 $\boldsymbol{u}=(u_0,u_1,u_2,\cdots,u_{511})$,编码后的数据 $\boldsymbol{v}=(v_0,v_1,v_2,\cdots,v_{575})$,编码过程可以表示为

$$\boldsymbol{v}=\boldsymbol{u}\boldsymbol{G} \tag{4}$$

伴随式 $\boldsymbol{s}=(s_0,s_1,s_2,\cdots,s_{63})$的生成过程为

$$\boldsymbol{s}=\boldsymbol{v}\boldsymbol{H}^{\mathrm{T}} \tag{5}$$

存储颗粒出错时会引入错误:$\boldsymbol{v}^*=\boldsymbol{v}+\boldsymbol{e}$,因此 $\boldsymbol{s}=\boldsymbol{v}^*\boldsymbol{H}^{\mathrm{T}}=(\boldsymbol{v}+\boldsymbol{e})\boldsymbol{H}^{\mathrm{T}}=\boldsymbol{v}\boldsymbol{H}^{\mathrm{T}}+\boldsymbol{e}\boldsymbol{H}^{\mathrm{T}}=\boldsymbol{u}\boldsymbol{G}\boldsymbol{H}^{\mathrm{T}}+\boldsymbol{e}\boldsymbol{H}^{\mathrm{T}}=\boldsymbol{e}\boldsymbol{H}^{\mathrm{T}}$.

(1)无错的判断

对于无错的情况,不需要进行纠错,可以根据$\boldsymbol{s}=0$ 判断.

(2)单符号错

可以推导出单符号错的伴随式满足式(6),因此根据式(6)中 7 个等式的成立情况进行判断,错误图样即 $\boldsymbol{s}_0$.

$$\begin{cases}\boldsymbol{s}_0(\boldsymbol{A}^{a_i})^{\mathrm{T}}=\boldsymbol{s}_1\\ \boldsymbol{s}_1(\boldsymbol{A}^{a_i})^{\mathrm{T}}=\boldsymbol{s}_2\\ \quad\vdots\\ \boldsymbol{s}_6(\boldsymbol{A}^{a_i})^{\mathrm{T}}=\boldsymbol{s}_7\end{cases} \tag{6}$$

译码过程中,使用钱搜索算法[32]找出错误位,即对于第 i 个符号($0\le i\le 71$),事先计算出$(\boldsymbol{A}^{a_i})^{\mathrm{T}}$,再计算 $\boldsymbol{s}_0(\boldsymbol{A}^{a_i})^{\mathrm{T}}+\boldsymbol{s}_1$的值.如果第 i 个符号的该值为 0,则说明单符号错误指针 $single\_sig\_err\_ptr[i]=1$,否则 $single\_sig\_err\_ptr[i]=0$.若 $single_sig_err_ptr[71:0]$中有一位为 1,则说明确实发生了单符号错.单符号错译码结构示意图如图 3 所示.

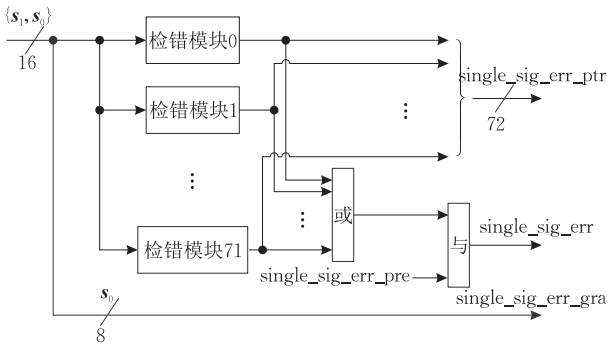


图 3 单符号错译码结构示意图

(3)双符号错至四符号错的检测与纠正

双符号错至四符号错的检纠方法采用错误位置多项式实现.本文以二符号错为例进行描述,三符号错和四符号错可用类似方法实现.

双符号错的伴随式可记为

$$\begin{cases}\boldsymbol{s}_0=\boldsymbol{e}_i+\boldsymbol{e}_j\\ \boldsymbol{s}_1=\boldsymbol{e}_i(\boldsymbol{A}^{a_i})^{\mathrm{T}}+\boldsymbol{e}_j(\boldsymbol{A}^{a_j})^{\mathrm{T}}\\ \boldsymbol{s}_2=\boldsymbol{e}_i(\boldsymbol{A}^{2a_i})^{\mathrm{T}}+\boldsymbol{e}_j(\boldsymbol{A}^{2a_j})^{\mathrm{T}}\\ \quad\vdots\\ \boldsymbol{s}_7=\boldsymbol{e}_i(\boldsymbol{A}^{7a_i})^{\mathrm{T}}+\boldsymbol{e}_j(\boldsymbol{A}^{7a_j})^{\mathrm{T}}\end{cases} \tag{7}$$

定义错误位置多项式 $\sigma(x)$为

$$\sigma(x)=(1-x(\boldsymbol{A}^{a_i})^{\mathrm{T}})(1-x(\boldsymbol{A}^{a_j})^{\mathrm{T}}) \tag{8}$$

$\sigma(x)$的根即为错误位置的逆,记

$$\begin{aligned}\boldsymbol{\sigma}_{2,1}&=(\boldsymbol{A}^{a_i}+\boldsymbol{A}^{a_j})^{\mathrm{T}},\\ \boldsymbol{\sigma}_{2,2}&=(\boldsymbol{A}^{a_i+a_j})^{\mathrm{T}}\end{aligned} \tag{9}$$

式(8)可化解为

$$\sigma(x)=1+\boldsymbol{\sigma}_{2,1}x+\boldsymbol{\sigma}_{2,2}x^2 \tag{10}$$

由式(7)可推导出

$$\begin{cases}\boldsymbol{s}_1\boldsymbol{\sigma}_{2,1}+\boldsymbol{s}_0\boldsymbol{\sigma}_{2,2}=\boldsymbol{s}_2\\ \boldsymbol{s}_2\boldsymbol{\sigma}_{2,1}+\boldsymbol{s}_1\boldsymbol{\sigma}_{2,2}=\boldsymbol{s}_3\end{cases} \tag{11}$$

可解得

$$\begin{cases}\boldsymbol{\sigma}_{2,1}=\dfrac{\boldsymbol{s}_1\boldsymbol{s}_3+\boldsymbol{s}_2\boldsymbol{s}_2}{\boldsymbol{s}_0\boldsymbol{s}_2+\boldsymbol{s}_1\boldsymbol{s}_1}\\ \boldsymbol{\sigma}_{2,2}=\dfrac{\boldsymbol{s}_0\boldsymbol{s}_3+\boldsymbol{s}_1\boldsymbol{s}_2}{\boldsymbol{s}_0\boldsymbol{s}_2+\boldsymbol{s}_1\boldsymbol{s}_1}\end{cases} \tag{12}$$

令

$$\begin{aligned}\hat{\boldsymbol{\sigma}}_{2,0}&=\boldsymbol{s}_0\boldsymbol{s}_2+\boldsymbol{s}_1\boldsymbol{s}_1,\\ \hat{\boldsymbol{\sigma}}_{2,1}&=\boldsymbol{s}_1\boldsymbol{s}_3+\boldsymbol{s}_2\boldsymbol{s}_2,\\ \hat{\boldsymbol{\sigma}}_{2,2}&=\boldsymbol{s}_0\boldsymbol{s}_3+\boldsymbol{s}_1\boldsymbol{s}_2\end{aligned} \tag{13}$$

则

$$\begin{cases}\boldsymbol{\sigma}_{2,1}=\dfrac{\hat{\boldsymbol{\sigma}}_{2,1}}{\hat{\boldsymbol{\sigma}}_{2,0}}\\ \boldsymbol{\sigma}_{2,2}=\dfrac{\hat{\boldsymbol{\sigma}}_{2,2}}{\hat{\boldsymbol{\sigma}}_{2,0}}\end{cases} \tag{14}$$

如果 $\hat{\boldsymbol{\sigma}}_{2,0}\neq 0$,则求解方程 $1+\boldsymbol{\sigma}_{2,1}x+\boldsymbol{\sigma}_{2,2}x^2=0$,可以等价于求解 $\hat{\boldsymbol{\sigma}}_{2,0}+\hat{\boldsymbol{\sigma}}_{2,1}x+\hat{\boldsymbol{\sigma}}_{2,2}x^2=0$.相反,如

《计算机学报》

果 $\hat{\boldsymbol{\sigma}}_{2,0}=0$，则说明没有双符号错.

随后，采用钱搜索算法，对于第 i 个符号（$0\leq i\leq 71$），先分别计算出 $(\boldsymbol{A}^{255-a_i})^{\mathrm{T}}$ 和 $(\boldsymbol{A}^{2\times(255-a_i)})^{\mathrm{T}}$ 的值，再计算 $\hat{\boldsymbol{\sigma}}_{2,0}+\hat{\boldsymbol{\sigma}}_{2,1}(\boldsymbol{A}^{255-a_i})^{\mathrm{T}}+\hat{\boldsymbol{\sigma}}_{2,2}(\boldsymbol{A}^{2\times(255-a_i)})^{\mathrm{T}}$ 的值. 如果第 i 个符号的该值为 0，则双符号错误指针 two_sig_err_ptr[i]=1，否则 two_sig_err_ptr[i]=0. 如果 two_sig_err_ptr[71:0]中恰有两位为 1，其余位为 0，则说明确实发生了双符号错，然后可根据式(7)求出错误图样，双符号错译码结构示意图如图 4 所示.

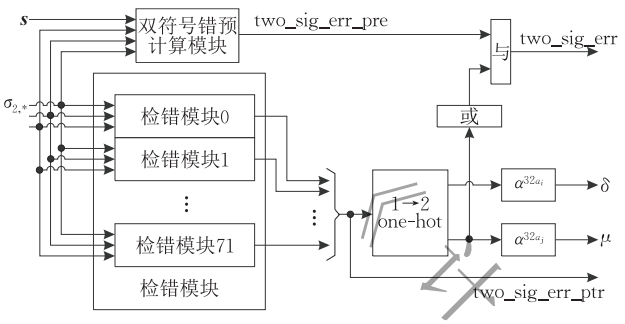


图 4　双符号错译码结构示意图

整个译码模块基本框图如图 5 所示.

从译码模块框图可以看出，译码模块主要包括伴随式计算、错误位置多项式系数计算、单符号错～四符号错处理单元，多项式乘法和多项式求逆是这些模块的基本组成单元. 在硬件电路实现这些部件时，进行针对性的精细化电路优化设计：

(1) 错误位置多项式系数计算单元是译码模块中最复杂的部件，为了控制电路延迟和面积，需尽可能减少多项式乘法单元的数量，因此，在电路设计上，本文并不是对每个系数单独进行计算，而是优化构建一个计算网络直接算出所有系数的结果，计算网络重复利用了中间结果，从而降低电路开销、节省电路面积.

(2) 多项式求逆不采用常规的欧几里得扩展算法，而是采用查表方式实现以达到较小的计算延迟.

(3) 合理划分站台以满足频率需求，根据逻辑功能和复杂性，在错误位置多项式系数计算单元和符号错判等单元后插入站台，使站台前后的逻辑延迟尽可能均衡.

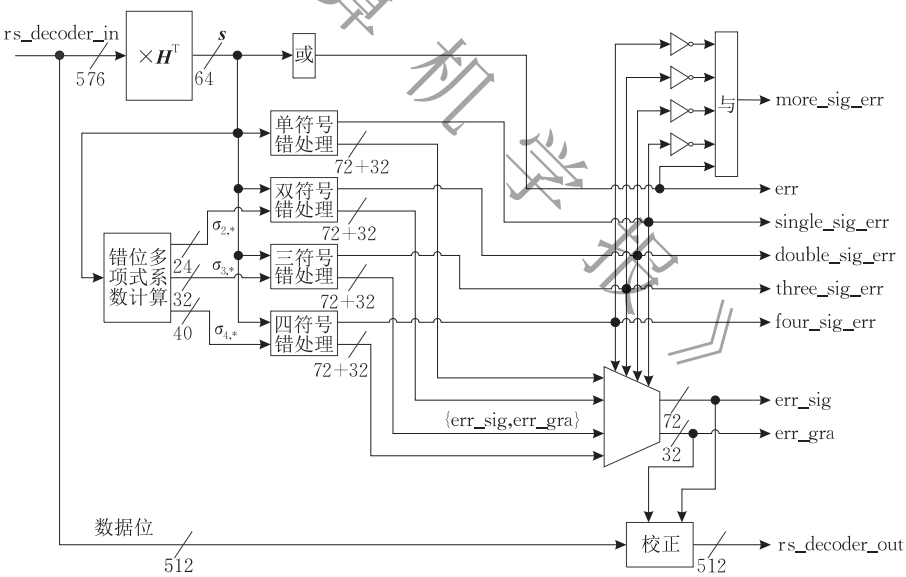


图 5　RS 码译码模块逻辑框图

通过精细化电路优化设计，最终能在 2 拍内完成译码，而传统的串行译码需要数十拍.

4.3　基于窗口保序的重传技术

RS 码强大的纠错能力可以较好地保证内存中数据存储和数据传输的可靠性. 然而，随着 DRAM 接口的速率逐渐增高，命令地址信号的可靠性问题也愈发严峻. 这可以从新一代 SDRAM 接口规范如 DDR4、HBM[33] 等都增加了命令地址信号的校验机制，并反馈校验错标志给存储控制器得到佐证. 对于命令地址信号，如果出现链路的瞬时故障，可能会影响到多个 DRAM 芯片. 在这种情况下，读出数据的错误很可能超出 RS 码的纠错能力. 针对这种情况，本文在 RS 码的基础上设计了一种基于窗口保序的重传技术，以提供更完备的访存保护机制.

设计高效的重传技术需要考虑以下几个问题：(1) 重传模块在存储控制器中的位置；(2) 重传模块的重传机制；(3) 访存事务的优先级和保序处理. 对于 DRAM 存储器，上层的读写访存事务会在存储

计算机学报

控制器中根据相应的 DRAM 协议转换成 DRAM 命令，然后与 DRAM 进行交互，通常一个访存事务需要被拆分成多个 DRAM 命令来实现. 因此，有两种请求重传的位置选择：一种是在事务层面进行重传，另一种是在命令层面进行重传. 在命令层面重传具有重传时间较短的优点，但是需要记录详细的 DRAM 的状态和一段时间内的命令序列，不利于实现. 重传过程只在出现错误时才需要，重传过程的时间开销对系统性能的影响非常小，因此，本文采用在事务层面进行重传. 事务级重传只需要关注读(写)事务从发出到完成期间有没有发生错误，若没有发生错误，则正常释放，否则进行重传. 为了减少重传模块中缓冲的大小，重传模块的位置应尽可能靠近将事务拆分成命令的位置.

重传模块需要重传的情形主要有两种：(1) RS 码译码模块检测到 RS 码不可纠正的错误(例如传输链路瞬态故障所致的多颗粒数据错)；(2) 目前 DDR4 等主流 DRAM 存储器检测到命令地址校验错或者写 CRC 校验错. 因此，重传模块需包含针对读数据校验失败、写数据校验失败以及命令地址信号校验失败的重传逻辑. 命令地址信号校验功能是针对 DRAM 命令进行校验的，采用时间窗口的方法将其与事务关联起来，当监测到命令地址校验失败时，所有正在飞行的事务(已经转换成 DRAM 命令但是还没有完成的事务)都需要进行重传. 当监测到 DRAM 返回写数据 CRC 校验错时，所有正在飞行的写事务都需要进行重传；对于读事务，由于其响应信息是精确的，只需要重传出错的读事务即可.

重传模块功能的正确性依赖于保序的正确性和优先级的处理，涉及的问题包括：

(1) 相同地址的读后写(RAW)、写后读(WAR)、写后写(WAW)请求需要保序；

(2) 读写事务同时完成时的释放处理；

(3) 一种类型事务报错与另一种类型事务完成同时出现的处理；

(4) 多个错误源同时报错的重传处理.

针对这一问题，本文提出基于窗口保序的重传机制. 首先，在重传模块中设置飞行缓冲用以缓存飞行中的请求，新发出的事务添加到缓冲中，完成的事务从缓冲释放. 对于申请发送的正常事务，须判断其是否与缓冲中的事务存在 RAW、WAR 和 WAW 三类地址冲突. 如果满足(1)重传模块中没有正在申请重传的事务，(2)不存在这三类地址冲突，(3)后续资源可用，则可以发送该正常事务，并且将其添加到缓冲中. 其次，由于飞行读事务和写事务是乱序完成的，存在两种事务同时完成或者一种事务报错而另一种事务完成的情形. 为了处理这些并发情形，可以通过在重传模块中将读写事务的飞行缓冲拆分成读飞行缓冲和写飞行缓冲来解决. 对于多个错误源同时报错的情况，由于读事务重传是单个事务重传，因此可以采用读重传优先处理的策略. 最后，在飞行缓冲中的每个事务都设置一个重传次数计数器，用于记录已重传的次数，以避免无限次重传. 重传模块的结构如图 6 所示. 访存请求的重传流程如图 7 所示.

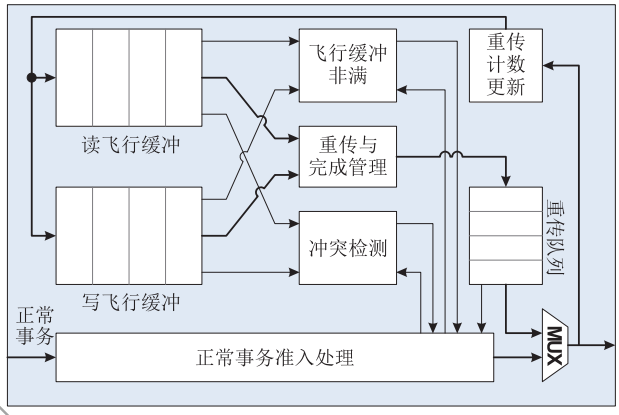


图 6 重传模块结构图

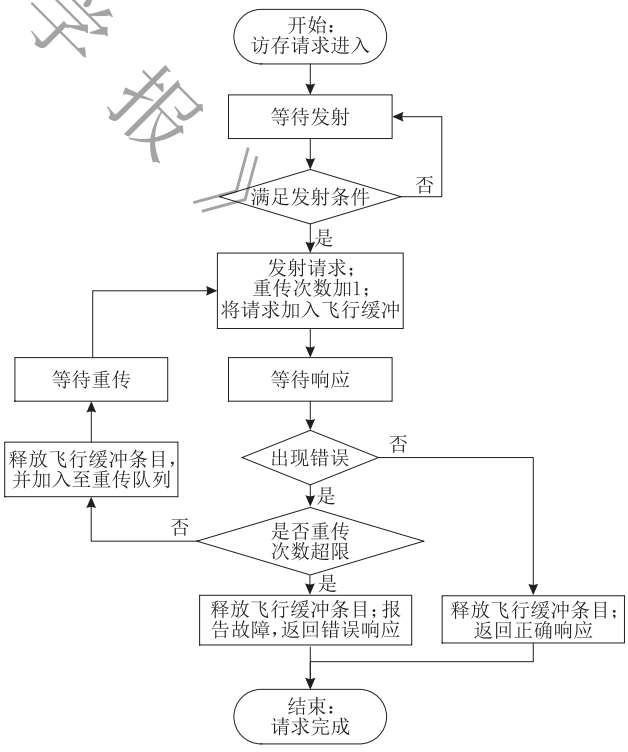


图 7 重传流程

《计算机学报》

5　评估与分析

本文使用 Verilog 语言实现了整个 R-RS 方案，使用 Synopsys 公司的 Design Compiler 工具在 28 nm 工艺单元库下进行综合并进行面积和时序评估，利用 Cadence IES15 工具对代码进行了带宽效率评估. 同时，用 C 语言搭建了由随机数生成模块、编码模块、错误注入模块、译码模块和结果比较模块构成的纠错能力模拟仿真环境，结构如图 8 所示，该环境可以高效真实分析出各种编码的纠错能力.

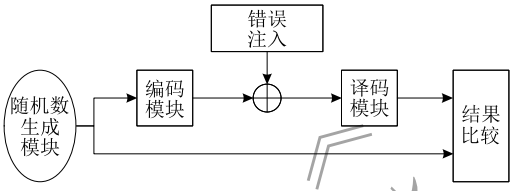


图 8　纠错模拟仿真环境

本节从可靠性、性能、面积开销对 R-RS 进行评估分析，选择 SEC-DED、E-ECC[13]、SSCMSD[24]①作为比较对象.

5.1　可靠性分析

5.1.1　编码纠错能力分析

根据编码理论，SEC-DED 只能纠正单比特错误，SSCMSD 可以纠正 18 个符号的任意 1 个符号错，E-ECC 可以纠正 36 个符号中的任意 2 个符号错，R-RS 可以纠正 72 个符号中的任意 4 个符号错，利用排列组合的知识，可以计算出 SEC-DED、E-ECC 和 R-RS 应对不同颗粒出错的纠错概率如表 2 所示，其中双颗粒突发错和三颗粒突发错是指 2 个或 3 个颗粒出现随机突发错，即每个颗粒可随机出现 1～4 个符号错. 从表中可以看出，E-ECC 和 R-RS 都能 100%纠正单颗粒突发错，即均实现了 Chipkill 功能，但是，对于双颗粒突发错和三颗粒突发错情形，本文的 R-RS 的纠错能力远高于 E-ECC，分别是其 1.833 倍和 2.095 倍.

表 2　不同编码方案的纠错能力

错误类型	SEC-DED/%	E-ECC/%	SSCMSD/%	R-RS/%
单比特随机错	100	100.0	100.0	100.0
单颗粒突发错	0	100.0	100.0	100.0
双颗粒突发错	0	40.0	22.2	58.7
三颗粒突发错	0	5.0	2.8	10.4

5.1.2　编码误纠概率分析

对于 RS 码，对于超出纠错能力的错误，可能出现误纠的情形，对于可纠 n 符号错的编码，分析出现 $n+1$ 个符号错时被误纠的概率.

对于 R-RS，最多可以纠 4 个符号错，5 符号错被误纠为 4 符号错的概率为

$$p_{5\to4}=\frac{C_{72}^{9}(2^8-1)C_9^5}{C_{72}^{5}(2^8-1)^5}=1.81\times10^{-4} \tag{15}$$

对于 E-ECC，最多可以纠 2 个符号错，3 符号错被误纠为 2 符号错的概率为

$$p_{3\to2}=\frac{C_{36}^{5}(2^8-1)C_5^3}{C_{36}^{3}(2^8-1)^3}=8.12\times10^{-3} \tag{16}$$

对于 SSCMSD，最多可以纠 1 个符号错，2 符号错被误纠为 1 符号错的概率为

$$p_{2\to1}=\frac{C_{18}^{3}(2^8-1)C_3^2}{C_{18}^{2}(2^8-1)^2}=6.27\times10^{-2} \tag{17}$$

但是 SSCMSD 方案中使用了一个 Hash 算法来检测纠错的正确性，因此其误纠概率还需要乘以 Hash 误检的概率，结果约为 1.46×10^{-11}，三种编码的误纠概率如表 3 所示. 由表可见，R-RS 误纠概率仅为 E-ECC 的 2.2%.

表 3　不同编码方案的误纠概率

编码方案	SSCMSD	E-ECC	R-RS
误纠概率	1.46×10^{-11}	6.27×10^{-2}	1.81×10^{-4}

5.1.3　R-RS 的编码纠错效果仿真结果

Sridaran 等人[8]研究了单芯片 DRAM 存储器不同错误类型的分布，如表 4 所示. 从表中可以看出，当一个存储器芯片出错时，接近一半(49.7%)的错是单比特错，其他是各种类型的多比特错. 本文提出的 R-RS 基于单 rank，多 rank 错误类型可当单 rank 看待. 据此建立如下的错误模型：假定单个芯片出错概率为 α，按照表 4 中给出的各种类型错误的分布，随机生成 10 000 000 种 576 位数据的错误模式(包含各种失效模式).

表 4　失效模式分布

失效模式	比例/%
单比特	49.7
单字	2.5
单列	10.6
单行	12.7
单体	16.3
多体	2.5
多排(rank)	5.5

出错芯片有 1/2 的概率为单比特错，出错符号数量的分布满足均匀分布. 根据该错误模型，随机生成.

① 由于原文中针对的是 x4 颗粒，不能直接比较，本文中将其类推到 x8 颗粒进行比较.

《计算机学报》

本文选择三种不同的 α 进行仿真，分别为 0.01，0.001，0.000 01. 三种纠错码的成功纠正的比例如表 5 所示. 从该表中可以得到四点结论：(1) 对于每个 α 值，E-ECC 和 R-RS 纠错成功的比例都显著高与 SEC-DED；(2) 在三种情况下，R-RS 成功的比例都高于 E-ECC；(3) SEC-DED 成功的比例并没有随着 α 的降低而显著提升；(4) 当 $\alpha=0.01$ 时，E-ECC 和 R-RS 成功的比例高于 90%，当 $\alpha=0.0001$ 时，E-ECC 和 R-RS 成功的比例接近 100%. 这体现出 E-ECC 和 R-RS 在纠各种类型错方面的综合能力显著优于 SEC-DED.

表 5 不同编码方案的访存成功率

α	SEC-DED	E-ECC	R-RS
0.01	0.459 124	0.915 015	0.934 352
0.001	0.495 632	0.991 288	0.993 377
0.0001	0.499 652	0.999 114	0.999 333

根据表 5 中的数据，可以计算出三种编码方案的失效率，如表 6 所示. 可以看出，在三种错误概率下，R-RS 的失效率都低于 E-ECC. 当 α 等于 0.01、0.001 和 0.0001 时，R-RS 的纠错能力分别是 E-ECC 的 1.29 倍、1.32 倍和 1.33 倍，分别是 SEC-DED 的 8.24 倍、76.15 倍和 750.15 倍. 这体现出 R-RS 在纠错能力方面的优势.

表 6 不同编码方案的失效率

α	失效率		
	SEC-DED	E-ECC	R-RS
0.01	5.41E-3	8.50E-4	6.56E-4
0.001	5.04E-4	8.71E-6	6.62E-6
0.0001	5.00E-5	8.86E-8	6.67E-8

5.1.4 实际运行效果

本文提出的 R-RS 技术在某国产众核处理器中实现并成功流片，已应用在了新一代神威 E 级计算机系统中，新一代神威 E 级计算机使用了上千万 3200Mbps DDR4 x8 颗粒. 在该系统中，(1) 通过读取重传相关的状态寄存器，我们观察到在 2 个月内有 38 个节点发生了访存重传，其中有 22 个节点出现了系统故障，另外 16 个节点通过重传得到了修复，系统正常运行，重传功能使得系统因访存导致的失效率降低了 42.1%；(2) 我们对比了使用 R-RS 功能和使用传统 SEC-DED 功能（处理器的存储控制器中实现了 2 种编码，可支持系统启动时配置使用不同的编码方案）的运行效果，其中使用 SEC-DED 的平均无故障运行时间约为 1.9 h，使用 R-RS 后，平均无故障运行时间提高到了约 69 h，可靠性提升了约 35.3 倍.

5.2 性能分析

5.2.1 重传对性能的影响

重传模块位于请求通路，它不会带来额外的处理节拍，不过重传机制其会阻塞访问相同地址的访存请求直到前序相同地址请求完成，所以对于只访问同一个地址的访存模式，访存延迟会增加约 14 个存控时钟周期，不过这种访存模式在实际中几乎不会出现，对于实际系统中常见的连续地址或者随机地址访存模式，重传机制不会影响正常事务的带宽效率. 对于读请求错误重传，一次重传将增加约 17 个存控时钟周期；对于写数据 CRC 错和命令地址校验错导致的重传，将会增加数百纳秒的重传延迟.

此外，我们评估了重传缓冲深度对性能的影响，由于随机地址访存本身的带宽效率有随机性，为了精准衡量重传缓冲深度的影响，本文选取连续地址读访存模式进行实验. 利用 verilog 编写的 R-RS 存储控制器代码＋Micron 的 DDR4 verilog 模型，在 Cadence IES15 工具上仿真运行，DDR4 配置速率为 3200 Mbps. 不同缓冲深度下带宽效率如图 9 所示，其中“无”表示没有重传缓冲，从图中可以看出缓冲深度较小时，带宽效率较低，并随缓冲深度增加而近似对数增长，当缓冲深度达到 32 时，对于带宽效率几乎没有影响，因此本文确定的缓冲深度为 32.

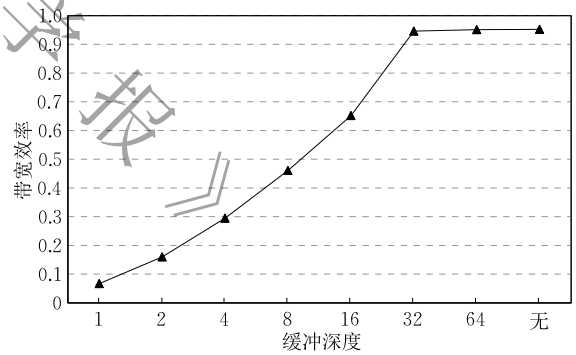


图 9 缓冲深度对带宽效率的影响

5.2.2 性能比较

硬件视角上，访存的性能指标有 2 个：访存带宽和访存延迟，从访存带宽来看，SEC-DED、E-ECC、SCCMSD 和 R-RS 均没有使用双层编码，校验信息和数据信息伴随传输，没有额外的读写操作，因此，存储数据的 DRAM 颗粒上的带宽的效率没有损失.

在访存延迟上，各方案的额外延迟如表 7 所示.

表 7 不同编码方案的延迟

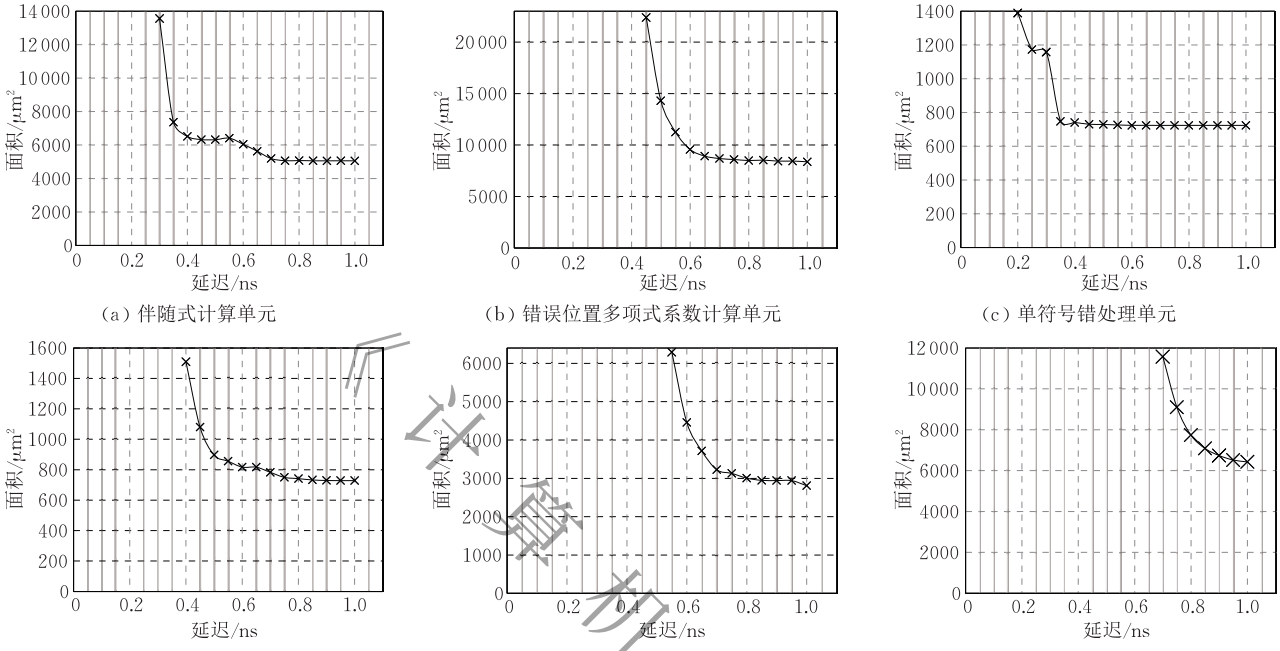
编码方案	SEC-DED	E-ECC	SSCMSD	R-RS
延迟（节拍）	0	1	2	1

R-RS 方案仅引入 1 拍的额外延迟.

《计算机学报》

5.3　综合结果与开销分析

RS 并行译码模块电路的可实现性是整个 R-RS 方案的关键因素，因此，本文对译码模块主要单元的面积延迟进行了仔细评估分析，图 10 显示了译码模块各主要单元的综合结果，从图中可以看出，错误位置多项式系数计算单元是译码模块中面积最大的单元，其面积达到了 8000 μm^2 以上，充分说明对其进行精细化优化的重要性，从图 10(d)～图 10(f)可以看出，随着所纠符号数的增加，其对应纠错单元的面积急剧增加，延迟也急剧增加，四符号错误图样生成单元的延迟是整个译码模块中最长的，在电路实现中可通过适量的低阈值电压单元进行优化.



(a) 伴随式计算单元　(b) 错误位置多项式系数计算单元　(c) 单符号错处理单元

(d) 双符号错错误图样生成单元　(e) 三符号错错误图样生成单元　(f) 四符号错错误图样生成单元

图 10　译码模块不同单元综合结果

本文分析了不同分段处理的面积和延迟开销，如表 8 所示.

表 8　R-RS 面积和延迟开销

处理方式	面积/μm^2	延迟/ns
1 级流水	58545.1	2.5
2 级流水	74599.3	1.1
3 级流水	73198.4	0.8

从表中的数据可以看出，译码逻辑分为 2 级流水处理时，可以满足 DDR4 等主流存储器高速传输的需求，具有可实现性. 采取 2 级流水的 R-RS 面积为 74599.3 μm^2，仅占整个存储控制器面积的 3.5%.

各编码方案的容量开销如表 9 所示.

表 9　不同编码方案的容量开销

编码方案	SEC-DED/%	E-ECC/%	SSCMSD/%	R-RS/%
延迟(节拍)	12.5	12.5	18.75	12.5

从表中可以看出，R-RS 编码的容量开销仅为 12.5%，容量开销小.

6　总　结

存储器的可靠性是构建 E 级计算机系统最关键的挑战之一. 传统的 SEC-DED 编码无法应对 E 级计算机系统的可靠性需求，而传统的 Chipkill 编码方案亦具有性能上的不足如需要额外的读写操作而难以适应 E 级计算机系统的要求，且这些已有的技术均没有考虑对命令、地址传输的保护.

本文提出了 R-RS 技术来应对 E 级计算机系统的内存可靠性挑战，通过精心挑选用于编码的本原多项式和校验矩阵设计了最利于硬件实现的可以纠 4 个 8 位符号错的 RS 编码，并精细化电路设计实现了高效的并行低延迟译码，同时设计了基于窗口保序的重传机制在存储控制器中首次实现了错误重传技术以保护命令、地址的传输.

R-RS 具有纠错能力强、误纠概率小、开销小的优势. R-RS 可以在数据传输链路、存储器等访存通路的各个层面对访存操作进行保护，应用在新一代

《计算机学报》

神威 E 级计算机系统后，使系统的平均无故障时间增长了 35.3 倍，表明 R-RS 是一种有效的 E 级计算的内存可靠性解决方案.

参 考 文 献

[1] Li S, Chen K, Hsieh M Y, et al. System implications of memory reliability in exascale computing//Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. Seattle, USA, 2011: 1-11

[2] Chen L, Zhang Z. MemGuard: A low cost and energy efficient design to support and enhance memory system reliability// Proceedings of the 2014 International Symposium on Computer Architecture. Minneapolis, USA, 2014: 49-60

[3] Qureshi M K, Kim D-H, Khan S, et al. AVATAR : A variable-retention-time (VRT) aware refresh for dram systems// Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). Rio de Janeiro, Brazil, 2015: 427-437

[4] Patil A, Nagarajan V, Balasubramonian R, et al. Dvé: Improving DRAM reliability and performance on-demand via coherent replication//Proceedings of the 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA). Valencia, Spain, 2021: 526-539

[5] Kim D W, Erez M. Stay alive, don't give up: DUE and SDC reduction with memory repair//Proceedings of the International Conference on Cluster Computing (CLUSTER). Chicago, USA, 2015: 588-594

[6] Hamming R W. Error detecting and error correcting codes. Bell System Technology Journal, 1950, 29(2): 147-160

[7] Nair P J, Sridharan V, Qureshi M. XED: Exposing on-die error detection information for strong memory reliability// Proceedings of the 43rd International Symposium on Computer Architecture. Seoul, Korea, 2016: 1-13

[8] Sridaran V, Liberty D. A study of DRAM failures in the field//Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. Salt Lake City, USA, 2012: 1-11

[9] Yoon D H, Erez M. Virtualized ECC: Flexible reliability in main memory. IEEE Micro, 2011, 31(1): 11-19

[10] Udipi A N, Muralimanohar N, Balsubramonian R, et al. LOT-ECC: Localized and tiered reliability mechanisms for commodity memory systems//Proceedings of the 39th International Symposium on Computer Architecture. Portland, USA, 2012: 285-296

[11] Jian X, Duwe H, Sartori J, et al. Low-power, low-storage-overhead Chipkill correct via multi-line error correction// Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. Denver, USA, 2013: 1-12

[12] Wang Di, Xu Yong. A new Chipkill code for DDR3 interface. Computer Engineering and Science, 2014, 36(12): 2386-2393 (in Chinese)
(王谛，许勇. 一种面向 DDR3 接口的新型 Chipkill 编码. 计算机工程与科学，2014，36(12)：2386-2393)

[13] Chen H M, Jeloka S, Arunkumar A, et al. Using low cost erasure and error correction schemes to improve reliability. IEEE Transactions on Computers, 2016, 65(12): 3766-3779

[14] Sridharan V, De Bardeleben N, Blanchard S, et al, Memory errors in modern systems: The good, the bad, and the ugly// Proceedings of the 20th International Conference on Architectural Support for Programming Languages and Operating Systems. Istanbul, Turkey, 2015: 297-310

[15] Vasudevan V, Sheshadri V, Ramachandran S, et al. Design and complexity analysis of Reed Solomon code algorithm for advanced RAID system in quaternary domain//Proceedings of the IEEE Computer Society Annual Symposium on VLSI. Chennai, India, 2011: 307-312

[16] Zhang W, Wang J, Zhang X. Low-power design of Reed-Solomon encoders//Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS). Beijing, China, 2013: 1560-1563

[17] Chen J, Jiang X W, Zhang Y, et al. CARE: Coordinated augmentation for elastic resilience on DRAM errors in data centers//Proceedings of the 2021 IEEE International Symposium on High Performance Computer Architecture. Seoul, Korea, 2021: 533-544

[18] Chen C M, Lin C W, Chen Y C. Packet scheduling for video vtreaming over wireless with content-aware packet retry limit// Proceedings of the 2006 IEEE Workshop on Multimedia Signal Processing. Victoria, Canada, 2006: 409-414

[19] Mittal S, Inukonda M S. A survey of techniques for improving error-resilience of DRAM. Journal of Systems Architecture, 2018, 91: 11-40

[20] Dell T J. A White Paper on the Benefits of Chipkill-Corret ECC for PC Server Main Memory. New York, USA: IBM Microelectronics Division, 1997

[21] Li S, Yoon D H, Chen K, et al. MAGE: Adaptive granularity and ECC for resilient and power efficient memory systems// Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. Salt Lake City, USA, 2012: 1-11

[22] Jian X, Kumar R. Adaptive reliability Chipkill correct (ARCC) //Proceedings of the IEEE 19th International Symposium on High Performance Computer Architecture. Shenzhen, China, 2013: 270-281

[23] Kim J, Sullivan M, Erez M. Bamboo ECC: Strong, safe, and flexible codes for reliable computer memory//Proceedings of IEEE 21st International Symposium on High Performance Computer Architecture. Burlingame, USA, 2015: 101-112

[24] Yeleswarapu R, Somani A K. SSCMSD- Single-symbol correction multi-symbol detection for dram subsystem//Proceedings of the 2018 IEEE 23rd Pacific Rim International Symposium on Dependable Computing (PRDC). Taipei, China, 2018: 15-24

[25] Krutov I. Reliability, Availability and Serviceability Features of the IBM eX5 Portfolio. New York, USA: IBM, 2012

《计算机学报》

[26] Jacob B, Ng S W, Wang D T. Memory Systems Cache, DRAM, Disk. Burlington, USA: Morgan Kaufmann, 2007: 410-416

[27] JEDEC. JESD79-3A. DDR3 SDRAM Specification. JEDEC Solid State Technology Association, Arlington, USA, 2007

[28] JEDEC. JESD79-4B. DDR4 SDRAM Specification. JEDEC Solid State Technology Association, Arlington, USA, 2017

[29] Bautista-Gomez L, Zyulkyarov F, Unsal O, et al. Unprotected computing: A large-scale study of DRAM raw error rate on a supercomputer//Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. Salt Lake City, USA, 2016: 1-11

[30] Lin S, Costello D J. Error Control Coding. 2nd Edition. New York: Pearson Education, 2004

[31] Neuberger G, de Lima Kastensmidt F G, Reis R. An automatic technique for optimizing Reed-Solomon codes to improve fault tolerance in memories. IEEE Design & Test, 2005, 22(1): 50-58

[32] Chien R. Cyclic decoding procedures for Bose-Chaudhuri-Hocquenghem codes. IEEE Transactions on Information Theory, 1964, 10(4): 357-363

[33] JEDEC. JESD235A. High Bandwidth Memory(HBM) DRAM. JEDEC Solid State Technology Association, Arlington, USA, 2015

GAO Jian-Gang, M. S., researcher. His research interests include computer architecture and high-performance interconnection network.

SHI Song, M. S., research assistant. His research interests focus on computer architecture.

ZHENG Fang, Ph. D., researcher. His research interests focus on computer architecture.

《计算机学报》

Background

Exascale computer is an important milestone in high performance computing. Many countries and organizations, including U. S., Japan, E. U. and China, have proposed their development plan of exascale computing. Reliability especially the memory reliability is one of the toughest challenges in constructing exascale systems due to the facts that (1) the memory capacity explosively increase, (2) the memory interface rates improved greatly, and (3) new technologies such as 3D stacking and lower supply voltage are adopted.

In order to solve the reliability problem of memory, the industry adopts the error-correction code (ECC) technology to tolerate memory faults. Currently, the mainstream ECC technology used in the server field is single error correction double error detection (SEC-DED) hamming code which can detect doubles error and correct single error. SEC-DED has a strong ability to correct random single-bit errors, but it cannot correct burst errors caused by memory faults, which often occur in supercomputers. Therefore, many supercomputers adopt Chipkill coding technology with stronger error-correcting ability. Compared with SEC-DED, Chipkill technology can significantly reduce the failure rate of computing nodes. However, current Chipkill technology still has some shortcomings when applied in exascale computing, such as performance loss caused by additional read and write operations, or insufficient ability to correct random errors, or cannot protect the transmission of commands and addresses. In addition, with the increase of memory interface rate, the error in commands and addresses lines are becoming much severer. For example, the command/address parity check function is adopted by DDR4, HBM and other high performance memory interface specifications. Traditional memory protection schemes in memory controller don't deal with it.

This paper proposed a memory reliability enhancement technology based on RS code and Retransmission mechanism, R-RS(Retransmission-RS) to meet the reliability requirements of exascale system. It can not only correct errors happened in data transmission links or in memory devices, but can also recovery from error happened in command/address transmission links, which greatly improve the reliability of memory access. Compared with a counterpart memory protection scheme, E-ECC, its error correction ability is improved by 83.3% and 109.5% respectively in the case of 2-device burst errors and 3-device burst errors. R-RS has been applied in the new generation of Sunway exascale supercomputer system and increased the mean time between errors of the whole system by 35.3 times.

Our team has been working on the high-performance computing and processors design. We have developed several designs for active and passive fault tolerant techniques on home-grown heterogeneous many-core processor architecture. We have developed several designs for active and passive fault tolerant techniques on home-grown heterogeneous many-core processor architecture. Such as independent and coordinated lightweight error recovery technique.