

内存纠错方法、内存条、内存控制器和处理器

本申请要求于 2023 年 3 月 31 日提交的申请号 202310363637.6、发明名称为“内存联合纠错方法、装置、芯片、设备和存储介质”的中国专利申请的优先权，以及，于 2023 年 6 月 25 日提交的申请号 202310751941.8、发明名称为“内存纠错方法、内存条、内存控制器和处理器”的中国专利申请的优先权，其全部内容通过引用结合在本申请中。

技术领域

本申请涉及存储技术领域，特别涉及一种内存纠错方法、内存条、内存控制器和处理器。

背景技术

在存储技术中，内存系统包括内存控制器和双倍速率同步动态随机存储器（double data rate synchronous dynamic random access memory, DDR SDRAM）。其中，双倍速率同步动态随机存储器也即是内存条（memory module），可以称为内存模块或者内存。该内存条包括多个内存颗粒，内存颗粒又称动态随机存取存储器（dynamic random access memory, DRAM），其中，一部分内存颗粒用于存储数据，称为数据颗粒（data device），另一部分内存颗粒用于存储数据的纠错码（error correcting code, ECC），也即是冗余信息，称为纠错码颗粒（ECC device），该纠错码用于校验内存颗粒中所存储的数据是否发生错误；内存控制器能够基于纠错码对码字内错误进行纠错，也即是内存控制器能够实现系统级纠错。其中，内存颗粒内部的存储阵列存储有多组码字，每组码字包括数据和校验码；内存颗粒还包括片上纠错引擎，该片上纠错引擎能够基于校验码对与该校验码属于同一码字的数据进行检错，若数据中单个比特发生错误，则片上纠错引擎能够对该错误进行纠正；若数据中多个比特发生错误，则可能会被片上纠错引擎当成单比特错误纠正，也可能片上纠错引擎无法检出该错误，形成静默错误，也即是内存颗粒中的片上纠错引擎能够实现颗粒级纠错。

相关技术中，在读取数据时，先通过片上纠错引擎对内存颗粒进行检错，片上纠错引擎一旦检出错误，立即上报给内存控制器，以辅助内存控制器定位到发生错误的内存颗粒，进而内存控制器基于 ECC device 中的冗余信息，对发生错误的内存颗粒进行数据恢复，再对数据进行重读。然而，上述方法中，针对多个比特发生错误的情况，片上纠错引擎有较大概率无法检出，从而导致内存控制器无法对该内存颗粒进行纠错，带来数据静默错误风险，影响内存数据的可靠性。

发明内容

本申请实施例提供了一种内存系统中的内存纠错方法、内存条、内存控制器和处理器，能够降低数据静默错误风险、提高内存系统的纠错能力，提高内存条中数据的可靠性。该技术方案如下。

第一方面，提供了一种内存纠错方法，该方法包括：

内存控制器在对从内存条中获取的数据纠错失败的情况下，确定内存条中发生错误的内存颗粒，进而基于发生错误的内存颗粒，对数据再次进行纠错。

其中，内存控制器对多个内存颗粒中的数据进行检测的结果以及后续步骤包括三种情况，第一种情况：未检测出错误，内存控制器直接将数据返回给处理器；第二种情况：检测出的错误数量在内存控制器所采用的纠错算法的纠错能力之内，也即是纠错成功，内存控制器通过该纠错算法对数据进行纠错，将纠错后的数据返回给处理器；第三种情况：检测出的错误数量超出了该纠错算法的纠错能力，也即是纠错失败，内存控制器确定内存条中发生错误的内存颗粒，进而基于发生错误的内存颗粒，对数据进行纠错。

其中，内存控制器通过内存颗粒的纠错状态信息确定内存条中发生错误的内存颗粒。内存控制器可以将纠错状态为不可纠错错误的内存颗粒确定为发生错误的内存颗粒；也可以将纠错状态为可纠正错误和不可纠正错误的内存颗粒均确定为发生错误的内存颗粒。

上述方法中，基于内存控制器的系统级纠错和基于片上纠错引擎的颗粒级纠错相互耦合，通过内存控制器对数据进行纠错，与仅依靠片上纠错引擎进行纠错相比，能够对片上纠错引擎未检测出的静默错误和误纠的错误进行纠正，因此能够降低内存颗粒中的数据静默错误风险和数据误纠风险；另外，内存

控制器在第一次纠错失败的情况下,利用片上纠错引擎的纠错结果,确定发生错误的内存颗粒,进而基于发生错误的内存颗粒再次对数据进行纠错,由于内存控制器在第二次纠错时,用于纠错的已知信息除了从内存条中获取的数据,还包括发生错误的内存颗粒的信息,与仅依靠内存控制器进行纠错相比,能够提高内存控制器的纠错能力,从而能够提高内存系统的纠错能力和内存条中数据的可靠性。

可选地,内存控制器基于内存颗粒的第一寄存器中所记录的内存颗粒的纠错状态信息,确定内存条中发生错误的颗粒。

其中,第一寄存器为内存条中原有的未定义功能的保留寄存器,该第一寄存器的取值能够用于表示内存颗粒的纠错状态。

上述方法中,将内存颗粒的纠错状态信息写入内存颗粒的第一寄存器中,内存控制器能够通过读取该第一寄存器中的纠错状态信息来定位到发生错误的颗粒,从而使得颗粒级的纠错和系统级的纠错能够相互配合,实现了对冗余资源的充分利用,有利于在固定冗余配置下提高内存系统的纠错能力。

可选地,内存控制器在对从内存条中获取到的数据纠错失败的情况下,对内存条的读写进程进行反压,并在反压成功后再次获取内存条中的数据,在两次获取到的数据一致的情况下,从多个内存颗粒的第一寄存器中,读取多个内存颗粒的纠错状态信息。

其中,反压是指从中央处理器(central processing unit, CPU)源头或访存路径抑制内存访问的产生。其中,内存控制器判断读取操作对应的内存地址是否为直接内存访问地址,在该内存地址为除直接内存访问(direct memory access, DMA)地址之外的内存地址,内存控制器才能够对内存条的读取进程进行反压,从而防止其他读写进程对待纠正的数据进行改写,进而避免数据不一致。

上述方法中,内存控制器对内存条的读取进程进行反压,从而防止其他读写进程对待纠正的数据进行改写,进而避免数据不一致;此外,通过再次从内存条中获取数据,以确定数据在反压生效前是否被改写过,在未被改写过的情况下,才继续后续数据纠错流程,能够保证数据纠错流程的有效性,进而保证数据的一致性。

可选地,在发生错误的内存颗粒的数量小于或等于内存条中纠错码颗粒的数量的情况下,内存控制器才能基于发生错误的内存颗粒,对该数据进行纠错。

其中,在确定发生错误的内存颗粒的情况下,内存控制器的纠错能力能够满足对目标数量的发生错误的内存颗粒进行纠错,其中,目标数量为内存条中纠错码颗粒的数量。发生错误的内存颗粒的数量小于或等于内存条中纠错码颗粒的数量,说明内存条中冗余的内存颗粒的数量大于或等于发生错误的内存颗粒的数量,也即是发生错误的内存颗粒的数量在内存控制器的纠错能力之内,内存控制器能够基于内存条中未发生错误的内存颗粒中的数据,对发生错误的内存颗粒中的数据进行纠错。

可选地,在对数据进行纠错之后,内存控制器将纠错后的数据回写至内存颗粒中,对该读取操作对应的内存地址进行重读,对重读到的数据进行校验,若校验通过,则内存控制器将重读到的数据返回给处理器,若校验不通过,则内存控制器向处理器上报错误。

其中,内存控制器将纠错后的数据回写至读取操作对应的内存地址中,也即是对各个内存颗粒中该内存地址对应的位置上的所有数据进行替换。

上述方法中,内存控制器将纠错后的数据回写至内存颗粒中,从而使得在下一次对同一内存地址进行读取操作时,能够读取到正确的数据,有利于提高内存条中数据的可靠性;此外,内存控制器对重读到的数据进行校验,在校验通过的情况下才返回数据,能够保证数据的一致性,提高内存条中数据的可靠性。

可选地,内存颗粒的不同的纠错状态由第一寄存器的目标位上的不同的取值来表示,第一寄存器的目标位的取值所表示的内存颗粒的纠错状态为无错误、可纠正错误和不可纠正错误中任一项。例如,目标位的取值为00B、01B和10B分别表示内存颗粒的纠错状态为无错误、可纠正错误和不可纠正错误。

可选地,内存颗粒的纠错状态由寄存器的目标位的占用状态来表示。例如,目标位的占用状态为空,也即是目标位无取值,表示内存颗粒的纠错状态为无错误或可纠正错误,目标位的占用状态不为空,也即是目标位有取值,表示内存颗粒的纠错状态为不可纠正错误。

可选地,发生错误的内存颗粒是指纠错状态为可纠正错误的内存颗粒和纠错状态为不可纠正错误的内存颗粒中的至少一种内存颗粒。

其中,内存控制器可以将纠错状态为不可纠正错误的内存颗粒确定为发生错误的内存颗粒;也可以将纠错状态为可纠正错误和不可纠正错误的内存颗粒均确定为发生错误的内存颗粒。

上述方法中，内存控制器可以将纠错状态为可纠正错误和不可纠正错误的内存颗粒均确定为发生错误的内存颗粒，由于片上纠错引擎在对数据进行纠错时可能会出现误纠的情况，也即是多比特错误当成单比特错误进行了纠正，因此，将纠错状态为可纠正错误的内存颗粒也确定为发生错误的内存颗粒，进而内存控制器不仅对片上纠错引擎不可纠的数据进行纠正，还能够对片上纠错引擎误纠的数据进行纠正，有利于进一步提高内存条中数据的可靠性。

可选地，内存颗粒包括片上纠错引擎，该片上纠错引擎能够对从内存颗粒中读取到的数据进行纠错，将该内存颗粒的纠错状态信息写入该内存颗粒的第一寄存器中，并将纠错后的数据输出至内存控制器。

上述方法中，片上纠错引擎将内存颗粒的纠错状态信息写入内存颗粒的第一寄存器中，使得内存控制器能够通过读取该第一寄存器中的纠错状态信息来定位到发生错误的内存颗粒，从而使得颗粒级的纠错和系统级的纠错能够相互配合，实现了对冗余资源的充分利用，有利于在固定冗余配置下提高内存系统的纠错能力。

第二方面，提供了一种内存纠错方法，由内存条执行，该内存条包括多个内存颗粒，每个内存颗粒包括第一寄存器，该方法包括：

对从内存颗粒中读取到的数据进行纠错，得到内存颗粒的纠错状态信息，将内存颗粒的纠错状态信息写入该内存颗粒的第一寄存器中。

第三方面，提供了一种内存控制器，该内存控制器包括至少一个功能模块，该至少一个功能模块用于执行如前述第一方面或第一方面中任一种可能的实现方式所提供的内存纠错方法。

第四方面，提供了一种内存条，该内存条包括多个内存颗粒，每个内存颗粒包括第一寄存器，该内存条用于执行如前述第二方面所提供的内存纠错方法。

第五方面，提供了一种内存控制器，该内存控制器用于执行如前述第一方面或第一方面中任一种可能的实现方式所提供的内存纠错方法。

第六方面，提供了一种处理器，该处理器包括内存控制器和计算核心，该处理器用于执行如前述第一方面或第一方面中任一种可能的实现方式所提供的内存纠错方法，该计算核心用于对内存颗粒中的数据执行计算操作。

第七方面，提供了一种计算设备，该计算设备包括内存控制器和内存条，该内存条用于暂存数据，该内存控制器用于执行如前述第一方面或第一方面中任一种可能的实现方式所提供的内存纠错方法。

第八方面，提供了一种计算机可读存储介质，该计算机可读存储介质用于存储至少一段程序代码，该至少一段程序代码用于执行如前述第一方面或第一方面中任一种可能的实现方式所提供的内存纠错方法。

第九方面，提供了一种包含至少一段程序代码的计算机程序产品，当该至少一段程序代码被计算设备运行时，使得该计算设备执行如前述第一方面或第一方面中任一种可能的实现方式所提供的内存纠错方法。

附图说明

- 图1是本申请实施例提供的一种内存条的示意图；
- 图2是本申请实施例提供的一种包括片上纠错引擎的内存颗粒的示意图；
- 图3是本申请实施例提供的一种内存系统的示意图；
- 图4是本申请实施例提供的一种内存纠错方法的流程图；
- 图5是本申请实施例提供的一种内存纠错方法的流程示意图；
- 图6是本申请实施例提供的一种内存纠错装置的结构框图；

图7是本申请实施例提供的一种计算设备的结构示意图。

具体实施方式

为使本申请的目的、技术方案和优点更加清楚，下面将结合附图对本申请实施方式作进一步地详细描述。

为了便于理解本申请的技术方案，下面先对本申请实施例中涉及的若干技术术语进行介绍。

1.内存条：冯诺依曼计算机系统包括五大部件：运算、控制、存储、输入和输出，其中存储包括主存（内存）和辅存（固态硬盘、机械硬盘等）。其中，内存条主要充当工作存储器（working memory），用于存储计算机运行所需要的指令和数据，是冯诺依曼计算机系统中不可或缺的一部分。其中，双倍速率同步动态随机存储器（double data rate synchronous dynamic random access memory, DDR SDRAM）是内存条的一种，该内存条的应用形态包括颗粒表贴和双列直插内存模组（dual in-line memory module, DIMM）。内存模组中的连接于同一个片选信号的内存条也称为一个内存列（rank），图1是本申请实施例提供的一种内存条的示意图，如图1所示，该内存条包括动态随机存取存储器（dynamic random access memory, DRAM）、寄存时钟驱动器（registering clock driver, RCD）、串行检测集线器（serial presence detect with hub, SPD Hub）、电源管理芯片（power management ic, PMIC）、温度传感器（temperature sensor, TS）、印制电路板（printed circuit board, PCB）和其他阻容件。内存模组包括两个通道（channel A和channel B），在面向高可靠性应用场景的企业级内存中，每个通道的位宽为40比特（bit），其中，数据位宽为32bit，纠错码（error correcting code, ECC）位宽为8bit。内存条又称为内存模块（memory module），内存条除了DRAM，还可以是随机存取存储器（random access memory, RAM）和可变电阻式存储器（resistive random access memory, RRAM）。

2.内存颗粒（device）：内存中的一个DRAM即是一个内存颗粒，内存颗粒是内存条中的存储介质，如图1所示，一个内存条中包括多个内存颗粒。根据内存颗粒中存放的内容，内存颗粒又分为数据颗粒（data device）和纠错码颗粒（ECC device）。根据每个内存颗粒对外的数据线路（lane）的数量，将内存颗粒分为x4内存颗粒和x8内存颗粒，x4和x8分别代表每个内存颗粒对外的数据线路的数量是4和8。

其中，第5代双倍速率同步动态随机存储存储器（double data rate synchronous dynamic random access memory, DDR5 SDRAM）的x8内存条中，内存模组的每个通道内的一个内存列（rank）包括4个数据颗粒（data device）和1个纠错码颗粒（ECC device）。响应于一次内存读取操作，每个内存颗粒向内存控制器输出128bit数据，该128bit数据分布于两个码字，每个内存颗粒为一个码字贡献64bit数据，由于每个内存颗粒有8个对外的数据线路，每个内存颗粒向内存控制器输出的数据也即是8×8的数据块。

其中，第5代双倍速率同步动态随机存储存储器（double data rate synchronous dynamic random access memory, DDR5 SDRAM）的x4内存条中，内存模组的每个通道内的一个内存列（rank）包括8个数据颗粒（data device）和2个纠错码颗粒（ECC device）。响应于一次内存读取操作，每个内存颗粒向内存控制器输出64bit数据，该64bit数据分布于两个码字，每个内存颗粒为一个码字贡献32bit数据，由于每个内存颗粒有4个对外的数据线路，每个内存颗粒向内存控制器输出的数据也即是4×8的数据块。

3.存储阵列（memory array）：内存颗粒中用于存储数据的阵列，由多个存储单元构成。存储阵列包括多个库组（bank group），每个库组包括多个库（bank）。

4.片上纠错码（on-die ECC, OD ECC）引擎：又称为in-DRAM ECC引擎，是实施于内存颗粒内部的纠错引擎，能够对从内存颗粒中读取到的数据进行纠错，从而提高内存颗粒良率。片上纠错引擎可以存在于内存颗粒的内部数据通路中的输入输出引脚（IO pin）或全局输入输出（global io, GIO）上，也可以存在于内存颗粒的内部数据通路中的局部输入输出（local io, LIO）上。图2是本申请实施例提供的一种包括片上纠错引擎的内存颗粒的示意图，如图2所示，图2包括内存颗粒和双倍速率同步动态随机存储器控制器（double data rate controller, DDRC），该双倍速率同步动态随机存储器控制器也即是内存控制器。该内存颗粒包括8个库组（bank group0-7）和片上纠错引擎，每个库组包括4个库（bank0-3）。图2中（a）所示为片上纠错引擎存在于内存颗粒的全局输入输出上的情况，当对内存条的读取操作对应的内存地址对应于bank group7中的bank3时，片上纠错引擎读取将该内存地址中的数据，片上纠错引擎对所读取到的数据进行纠错，并将纠错后的数据返回给内存控制器。图2中（b）所示为片上纠错引擎存在于内存颗粒的局部输入输出的情况，也即是每个bank中的数据由一个片上纠错引擎进行纠错。

其中, 片上纠错引擎对所读取到的数据进行检测的结果包括三种情况: 第一种情况: 未检测到错误, 相对应地, 该内存颗粒的纠错状态为无错误; 第二种情况: 检测到单比特错误, 片上纠错引擎能够对该单比特错误进行纠正, 相对应地, 该内存颗粒的纠错状态为可纠正错误; 第三种情况: 检测到多个比特发生错误, 片上纠错引擎无法对多比特错误进行纠正, 相应地, 该内存颗粒的纠错状态为不可纠正错误。

5. 内存控制器: 内存控制器位于 CPU 中, 是计算机系统内部控制内存并且负责内存条与 CPU 之间数据交换的重要组成部分。以内存控制器为双倍速率同步动态随机存储器控制器 (double data rate controller, DDRC) 为例, 该内存控制器包括内存控制器纠错引擎 (DDRC ECC engine)、动态内存控制器 (dynamic memory controller, DMC)、微控制单元 (micro controller unit, MCU)、位模式发生器 (bit patter generator, BPG) 和物理层 (physical layer, PHY)。

6. 里德-所罗门码 (reed-solomon codes, RS codes) 算法: 该 RS 码算法码字内包括 k 个数据元素 (symbol) 和 $2t$ 个纠错码元素 (ECC symbol), 其中, k 和 t 为大于 0 的正整数。一个 symbol 为通过两个引脚 (pin) 输出的数据, 该 RS 码算法对 t 个 symbol 以内的错误 100% 可纠, 超过 t 个 symbol 的错误则超出了该 RS 码算法的纠错能力。例如, DDR5 SDRAM x8 内存条包括 4 个数据颗粒和 1 个纠错码颗粒, 其中, 每个内存颗粒对外的数据线路为 8, 则每个内存颗粒输入的数据可以被划分为 4 个 symbol, 5 个内存颗粒输出至内存控制器中的数据被划分为 $4 \times 4 = 16$ 个数据 symbol 和 $1 \times 4 = 4$ 个纠错码 symbol, 也即是 $k=16$, $t=2$, 该 RS 码算法能够纠正 DDR5 SDRAM x8 内存条中 2 个 symbol 的错误, 而 2 个 symbol 以上的错误, 则超出了 RS 码算法的纠错能力。

下面介绍本申请实施例的实施环境。

图 3 是本申请实施例提供的一种内存系统的示意图, 如图 3 所示, 该内存系统包括内存条 301 和内存控制器 302, 该内存条 301 包括多个内存颗粒, 内存条 301 和内存控制器 302 之间通过 DDR 总线进行通信。

其中, 该内存条 301 可以是任意带有片上纠错引擎的内存条, 例如, DDR5、DDR6 或低功耗双倍数据速率内存 (Low Power Double Data Rate SDRAM, LPDDR) 等等, 该内存条 301 的形态可以是双列直插内存模组, 也可以是颗粒表贴, 该内存 301 中的内存颗粒可以是 x4 内存颗粒, 也可以是 x8 内存颗粒, 本申请实施例对此不做限定。内存 301 中的内存颗粒包括数据颗粒和纠错码颗粒, 每个内存颗粒存储有多组码字, 每组码字包括数据和校验码。示例性地, 内存颗粒中的片上纠错引擎基于校验码对与该校验码属于同一码字的数据进行纠错, 并将内存颗粒的纠错状态信息写入寄存器中, 以及将纠错后的数据输出至内存控制器 302。

其中, 该内存控制器 302 可以是任意具有系统级纠错功能的内存控制器, 例如, 使用 RS 码进行纠错的 DDRC, 本申请实施例对此不做限定。示例性地, 该内存控制器 302 能够获取内存条 301 中的数据, 并对获取到的数据进行纠错, 进而将纠错后的数据回写至内存条 301 中。

本申请实施例对内存条中原有的未定义功能的保留寄存器进行了定义, 以使寄存器 (也即是第一寄存器) 的取值能够用于表示内存颗粒的纠错状态。在一些实施例中, 内存颗粒的纠错状态由第一寄存器的目标位的取值来表示, 例如: 目标位的取值为 00B、01B 和 10B 分别表示内存颗粒的纠错状态为无错误、可纠正错误和不可纠正错误。在另一些实施例中, 内存颗粒的纠错状态由寄存器的目标位的占用状态来表示, 例如, 目标位的占用状态为空, 也即是目标位无取值, 表示内存颗粒的纠错状态为无错误或可纠正错误, 目标位的占用状态不为空, 也即是目标位有取值, 表示内存颗粒的纠错状态为不可纠正错误。需要说明的是, 上述对内存颗粒的纠错状态的表示方式的说明仅是示例性地, 可以根据实际需求设置内存颗粒的纠错状态的表示方式, 本申请实施例对此不做限定。

下面以 DDR5 SDRAM 内存为例, 对内存颗粒的纠错状态由第一寄存器的目标位的取值来表示这一表示方式进行举例说明。

DDR5 SDRAM 内存包括 256 个 8bit 的模式寄存器 (mode register, MR): MR0-MR255, 其中, MR41、MR49、MR70-102、MR117、MR119、MR125、MR127、MR135、MR143、MR155、MR159、MR167、MR175、MR183、MR191、MR199、MR207、MR215、MR223、MR231、MR239、MR247、MR255 完全未被定义, 还有很多模式寄存器具有两比特及以上寄存器空间未被定义, 例如 MR9 的操作数 2 至操作数 6 (Operand[6:2], OP[6:2]) 的取值均可用于表示纠错状态。在一些实施例中, 第一寄存器 MRx 具有两比

特及以上空间未被定义，取其中两比特空间，例如，OP[u]、OP[v]，OP[u]、OP[v]也即是目标位，则用OP[u]、OP[v]的不同取值表示不同的纠错状态，如下表 1 所示。其中，片上纠错引擎通过对第一寄存器中的OP[u]、OP[v]赋值的方式，将纠错状态信息写入第一寄存器中。如表 2 所示，若片上纠错引擎未检测出错误，则将 MRx.uv（第一寄存器 MRx 中的 OP[u]、OP[v]）赋值 00B；若片上纠错引擎检测到可纠正错误，则将MRx.uv 赋值 01B；若片上纠错引擎检测到不可纠正错误，则将 MRx.uv 赋值 10B。需要说明的是，表 1 所示的表示方式和表 2 所示的各个取值所表示的纠错状态仅是示例性的，可以根据实际需求设置内存颗粒的纠错状态的表示方式，本申请实施例对此不做限定。

表 1

OP[7]	...	OP[u]	OP[v]	...	OP[0]
On-die ECC error checking status (纠错状态信息)					

表 2

Function (功能)	Register Type (寄存器类型)	Operand (操作数)	Data (数值)
On-die ECC error checking status (纠错状态信息)	R (可读)	OP[u]、OP[v]	00B:No error reported (无错误) 01B:CE (可纠正错误) 10B:UCE (不可纠正错误) 11B:Reserved (保留位)

其中，内存控制器可以通过带内模式寄存器读取命令(mode register read, MRR)对第一寄存器进行读取，也可以通过带外方式对第一寄存器进行读取，本申请实施例对此不做限定。

上面介绍了本申请实施例中第一寄存器的定义方式，下面介绍本申请实施例提供的一种内存纠错方法。该方法可应用于 DDR5 SDRAM x8 内存条和 DDR5 SDRAM x4 内存条，下面分别以上述两种内存条为例进行介绍。需要说明的是，上述两种类型的内存条仅是示例性的，该方法还可以应用于其他类型的内存条，本申请实施例对此不做限定。

下面以 DDR5 SDRAM x8 内存条为例进行介绍。图 4 是本申请实施例提供的一种内存纠错方法的流程图，如图 4 所示，该方法包括下述步骤 401 至步骤 408。

401、响应于对内存条的读取操作，内存颗粒中的片上纠错引擎获取内存颗粒中的数据，对所获取到的数据进行纠错，将纠错状态信息写入该内存颗粒的第一寄存器中，将纠错后的数据输出至内存控制器。

其中，内存颗粒存储有数据和校验码，其中，校验码用于校验数据是否发生错误。片上纠错引擎获取内存颗粒中的数据，对所获取到的数据进行纠错的过程包括：响应于对内存条的读取操作，从该读取操作对应的内存地址读取码字，所读取到的码字包括数据和数据的校验码；基于所读取到的校验码，对所读取到的码字进行纠错。

其中，片上纠错引擎对数据进行检测的结果包括三种情况：第一种情况：未检测到错误；第二种情况：检测到数据发生单比特错误，片上纠错引擎对该单比特错误进行纠正；第三种情况：检测到数据发生两比特及以上的错误，片上纠错引擎无法纠正。与纠错结果相对应地，内存颗粒的纠错状态包括无错误、可纠正错误(correctable error, CE)和不可纠正错误(uncorrectable error, UCE)。该纠错状态信息用于指示内存颗粒中的数据的纠错状态。

其中，内存颗粒通过数据总线将纠错后的数据输出至内存控制器，其中，DDR5 SDRAM 的 x8 内存条中，响应于一次内存读取操作，每个内存颗粒向内存控制器输出 128bit 数据，该 128bit 数据分布于两个码字，每个内存颗粒向一个码字贡献 64bit 数据，由于每个内存颗粒有 8 个对外的数据线路，每个内存颗粒向内存控制器输出的数据也即是 8×8 的数据块。

上述方法中，片上纠错引擎将内存颗粒的纠错状态信息写入内存颗粒的第一寄存器中，使得内存控制器能够通过读取该第一寄存器中的纠错状态信息来定位到发生错误的内存颗粒，从而使得颗粒级的纠错和系统级的纠错能够相互配合，实现了对冗余资源的充分利用，有利于在固定冗余配置下提高内存系统的纠错能力。

需要说明的是，响应于对内存条的读取操作，内存条中的多个内存颗粒将纠错状态信息写入各自的第一寄存器，并同步向内存控制器输出纠错后的数据，内存颗粒输出的数据为存储阵列中的数据（也即

是所读取到的数据)和片上纠错引擎对数据的纠错结果的叠加。

需要说明的是,上述步骤401中片上纠错引擎获取内存颗粒中的数据、对所获取到的数据进行纠错、将纠错状态信息写入该内存颗粒的第一寄存器中的步骤为可选步骤,在一些实施例中,若内存颗粒发生错误,则内存颗粒直接向内存控制器上报错误,进而内存控制器在纠错失败后能够基于上报错误的内存颗粒确定发生错误的内存颗粒,而内存颗粒无需将纠错状态信息写入第一寄存器,第一寄存器也无需从第一寄存器中读取纠错状态信息以确定发生错误的内存颗粒,能够节约纠错的时间成本和寄存器资源。

402、内存控制器获取多个内存颗粒输出的数据,对多个内存颗粒中的数据进行纠错。

其中,内存控制器对多个内存颗粒中的数据进行检测,若内存控制器检测到的错误数量大于目标阈值,说明内存控制器纠错失败。

在一些实施例中,内存控制器通过RS码算法对多个内存颗粒中的数据进行检测。该目标阈值为内存控制器所采用的纠错算法所能纠正的最大错误数量,例如,DDR5 SDRAM x8内存条包括4个数据颗粒和1个纠错码颗粒,其中,每个内存颗粒对外的数据线路为8,则每个内存颗粒输入的数据可以被划分为4个symbol,5个内存颗粒输出至内存控制器中的数据被划分为 $4 \times 4 = 16$ 个数据symbol和 $1 \times 4 = 4$ 个纠错码symbol,也即是 $k=16$, $t=2$,该RS码算法能够纠正DDR5 SDRAM x8内存中2个symbol的错误,而2个symbol以上的错误,则超出了RS码算法的纠错能力。

其中,内存控制器对多个内存颗粒中的数据进行检测的结果以及后续步骤包括三种情况,第一种情况:未检测出错误,内存控制器直接将数据返回给CPU;第二种情况:检测出的错误数量在内存控制器所采用的纠错算法的纠错能力之内,也即是纠错成功,内存控制器通过该纠错算法对数据进行纠错,将纠错后的数据返回给CPU;第三种情况:检测出的错误数量超出了该纠错算法的纠错能力,也即是纠错失败,内存控制器确定内存中发生错误的内存颗粒,进而基于发生错误的内存颗粒,对数据进行纠错。

下面对上述第三种情况下,内存控制器基于发生错误的内存颗粒,对数据进行纠错的过程进行介绍,该过程包括下述步骤403至步骤406。

403、若纠错失败,则内存控制器对内存条的读写进程进行反压。

其中,反压是指从CPU源头或访存路径抑制内存访问的产生。其中,内存控制器判断读取操作对应的内存地址是否为直接内存访问地址,在该内存地址为除直接内存访问(direct memory access, DMA)地址之外的内存地址,内存控制器才能够对内存条的读取进程进行反压,从而防止其他读写进程对待纠正的数据进行改写,进而避免数据不一致。

在一些实施例中,处理器包括多个内核(core),内存控制器对内存条的读写进程进行反压的过程包括:若任一处理器内核的内存读取操作所访问到的数据的错误数量超过目标阈值,则该内核通过软件触发中断(software generated interrupt, SGI)发起停核广播;处理器中的其他内核收到该停核广播后置停核标志位,以停止内存访问,并等待被唤醒;发起停核广播的内核在预设时间后检测其他内核是否停核完成,若各个内核均停核完成,则内存控制器进入后续数据恢复流程;若任一内核未停核完成,则内存控制器记录停核失败,并唤醒其他内核,返回错误状态。上述方法中,通过各个处理器内核停核的方式对内存条的读取进程进行反压,能够从访问源防止其他读写进程对待纠正的数据进行改写,从而能够有效地保证数据的一致性。

在另一些实施例中,内存控制器包括动态内存控制器和物理层(physical layer, PHY),该动态内存控制器包括调度器,该调度器用于对内存条的读写命令进行调度。内存控制器通过反压内存控制器中的命令调度队列,来切断调度器与PHY之间的通路,从而从调度队列防止其他读写进程对待纠正的数据进行改写,反压更精细化,处理器内核不停止工作,对上层业务的影响较小。

需要说明的是,该步骤403是以读取操作对应的内存地址为除DMA地址之外的内存地址为例进行说明的,在一些实施例中,读取操作对应的内存地址为DMA地址,则内存控制器放弃本次纠错,返回错误状态。

需要说明的是,该步骤403中对内存条的读写进程进行反压的过程为可选步骤,在一些实施例中,不执行该步骤403中对内存条的读写进程进行反压的过程,本申请实施例对此不做限定。

404、内存控制器再次获取内存条中的数据,若两次获取到的数据一致,则从多个内存颗粒的第一寄存器中,读取多个内存颗粒中数据的纠错状态信息。

其中,内存控制器获取内存条中的数据的过程与上述步骤402同理,不再赘述。若内存控制器两次获取到的数据一致,则说明读取操作对应的内存地址中的数据在反压生效前没有被改写,内存控制器继续

后续数据纠错流程；若内存控制器两次获取到的数据不一致，则说明读取操作对应的内存地址中的数据在反压生效前被改写过，内存控制器结束数据纠错流程，返回错误状态。上述方法中，通过再次从内存条中获取数据，以确定数据在反压生效前是否被改写过，在未被改写过的情况下，才继续后续数据纠错流程，能够保证数据纠错流程的有效性，进而保证数据的一致性。

其中，内存控制器可以通过带内模式寄存器读取命令对第一寄存器进行读取，也可以通过带外方式对第一寄存器进行读取，本申请实施例对此不做限定。

需要说明的是，该步骤 404 为可选步骤，在一些实施例中，若不执行上述步骤 403 中对内存条的读写进程进行反压的过程，则不执行该步骤 404，本申请实施例对此不做限定。

405、若发生错误的内存颗粒的数量小于或等于内存条中纠错码颗粒的数量，则内存控制器基于发生错误的内存颗粒，对数据进行纠错。

其中，内存控制器基于从第一寄存器中读取到的多个内存颗粒的纠错状态信息，确定发生错误的内存颗粒，若任一内存颗粒的纠错状态信息指示该内存颗粒发生错误，则内存控制器确定该内存颗粒为发生错误的内存颗粒。上述方法中，内存控制器基于第一寄存器中的纠错状态信息确定发生错误的内存颗粒，进而基于发生错误的内存颗粒对数据进行纠错，颗粒级的纠错和系统级的纠错能够相互配合，实现了对冗余资源的充分利用，有利于在固定冗余配置下提高内存系统的纠错能力。

在一些实施例中，内存控制器将纠错状态为不可纠错误的内存颗粒确定为发生错误的内存颗粒；在另一些实施例中，内存控制器将纠错状态为可纠正错误和不可纠正错误的内存颗粒均确定为发生错误的内存颗粒，由于片上纠错引擎在对数据进行纠错时可能会出现误纠的情况，也即是多比特错误当成单比特错误进行了纠正，因此，将纠错状态为可纠正错误的内存颗粒也确定为发生错误的内存颗粒，进而内存控制器不仅对片上纠错引擎不可纠的数据进行纠正，还能够对片上纠错引擎误纠的数据进行纠正，有利于进一步提高内存条中数据的可靠性。

其中，内存控制器的纠错能力能够满足对目标数量的发生错误的内存颗粒进行纠错，其中，目标数量为内存条中纠错码颗粒的数量。发生错误的内存颗粒的数量小于或等于内存条中纠错码颗粒的数量，说明内存中冗余的内存颗粒的数量大于或等于发生错误的内存颗粒的数量，也即是发生错误的内存颗粒的数量在内存控制器的纠错能力之内，内存控制器能够基于内存中未发生错误的内存颗粒中的数据，对发生错误的内存颗粒中的数据进行纠错。例如，DDR5 SDRAM x8 内存条包括 4 个数据颗粒和 1 个纠错码颗粒，也即是内存中冗余的内存颗粒的数量为 1，当且仅当发生错误的内存颗粒的数量小于或等于 1 时，内存控制器能够对发生错误的内存颗粒中的数据进行纠错，当发生错误的内存颗粒的数量大于 1 时，内存控制器结束数据纠错流程，向 CPU 返回不可纠错误。

在一些实施例中，内存控制器采用纠删码 (erasure code, EC) 算法对发生错误的内存颗粒中的数据进行纠错。以 DDR5 SDRAM x8 内存条为例，该数据纠错过程包括下述步骤 405A 至步骤 405C。

405A、将 4 个数据颗粒中的数据分别记为 D1、D2、D3、D4，将 1 个纠错码颗粒 (冗余颗粒) 中的数据记为 C1，其中，D1、D2、D3、D4 和 C1 均为 8×8 的数据块。根据 RS 码算法也即是 EC 码的性质，则存在一个矩阵 H 用于生成码字，其中 H 可见元素均为 8×8 的分块矩阵，该过程可以由下述公式 (1) 表示。

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ B1 & B2 & B3 & B4 \end{bmatrix} \begin{bmatrix} D1 \\ D2 \\ D3 \\ D4 \end{bmatrix} \quad (1)$$

405B、对于任一内存颗粒发生错误的情况，将该内存颗粒对应的矩阵行从矩阵 H 中去掉，矩阵依旧是满秩矩阵 H' ，该满秩矩阵存在逆矩阵 H'^{-1} 。例如，内存颗粒 D1 发生错误，则将该 D1 对应的矩阵行从矩阵 H 中去掉，该过程可以由下述公式 (2) 表示。

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ B1 & B2 & B3 & B4 \end{bmatrix} \begin{bmatrix} D1 \\ D2 \\ D3 \\ D4 \end{bmatrix} = \begin{bmatrix} D2 \\ D3 \\ D4 \\ C1 \end{bmatrix} \quad (2)$$

405C、将去掉 D1 后的数据左乘逆矩阵 H'^{-1} ，得到纠错后的数据，该过程可以由下述公式 (3) 表示。

$$\begin{bmatrix} D1 \\ D2 \\ D3 \\ D4 \end{bmatrix} = H'^{-1} \begin{bmatrix} D2 \\ D3 \\ D4 \\ C1 \end{bmatrix} \quad (3)$$

需要说明的是，上述步骤 403 至步骤 405 是若纠错失败，则确定内存条中发生错误的内存颗粒，基于发生错误的内存颗粒，对数据进行纠错的一种实现方式，在一下实施例中，基于其他方式实现该过程，本申请实施例对此不做限定。

406、内存控制器将纠错后的数据回写至内存颗粒中。

其中，内存控制器将纠错后的数据回写至读取操作对应的内存地址中，也即是对各个内存颗粒中该内存地址对应的位置上的所有数据进行替换。在一些实施例中，内存控制器将纠错后的数据中对应于发生错误的内存颗粒的数据回写至该内存颗粒中该内存地址对应的位置上，也即是仅对发生错误的内存颗粒中该内存地址对应的位置上的数据进行替换，本申请实施例对此不做限定。

上述方法中，内存控制器将纠错后的数据回写至内存颗粒中，从而使得在下一次对同一内存地址进行读取操作时，能够读取到正确的数据，有利于提高内存条中数据的可靠性。

需要说明的是，该步骤 406 为可选步骤，在一些实施例中，不执行该步骤 406，内存控制器直接将纠错后的数据返回给 CPU，以缩短对内存条的读取操作的应答时间，提高应答效率，本申请实施例对此不做限定。

407、内存控制器解除对内存条的读写进程的反压。

其中，该步骤 407 与上述步骤 403 相对应，若内存控制器通过处理器内核停核的方式对内存条的读写进程进行反压，则内存控制器通过唤醒处理器内核来解除反压；若内存控制器通过反压命令调度队列来对内存条的读写进程进行反压，则内存控制器解除对命令调度队列的反压，使得调度器与物理接口协议之间的通路连通，从而解除对内存条的读写进程的反压。

需要说明的是，该步骤 407 为可选步骤，在一些实施例中，若不执行上述步骤 403，则不执行该步骤 407，本申请实施例对此不做限定。

408、内存控制器对该读取操作对应的内存地址进行重读，对重读到的数据进行校验，若校验通过，则内存控制器将重读到的数据返回给处理器，若校验不通过，则内存控制器向处理器上报不可纠错错误。

其中，内存控制器对该读取操作对应的内存地址进行重读的过程与上述步骤 402 中内存控制器获取内存条中的数据的过程同理，内存控制器对重读到的数据进行校验的过程与上述步骤 402 中内存控制器对数据进行检测的过程同理，不再赘述。

上述步骤 408 中，内存控制器对重读到的数据进行校验，在校验通过的情况下才返回数据，能够保证数据的一致性，提高内存中数据的可靠性。

需要说明的是，该步骤 408 为可选步骤，在一些实施例中，若不执行上述步骤 406，则不执行该步骤 408，本申请实施例对此不做限定。

下面通过图 5 对上述步骤 401 至步骤 408 所示的流程进行举例说明。图 5 是本申请实施例提供的一种内存纠错方法的流程示意图，如图 5 所示，图 5 包括内存控制器和内存条，其中，内存条为 DDR5 SDRAMx8 内存条，内存控制器为 DDRC。内存条中一个通道中的一个内存 rank 包括 4 个数据颗粒和 1 个纠错码颗粒，每个数据颗粒对外的数据线路的数量为 8，内存颗粒包括片上纠错引擎，每个内存颗粒对应于一个第一寄存器。内存控制器包括 RS 码 DDRC ECC engine、DMC、MCU、BPG 和 PHY。

图 5 中步骤 1（对应于上述步骤 401）：响应于对内存条的读取操作，片上纠错引擎从内存颗粒的存储阵列中读取数据并做校验，基于校验结果，将纠错状态信息写入 MRx 寄存器中。图 5 中步骤 2（对应于上述步骤 402）：各个内存颗粒中的数据通过数据总线返回给内存控制器中的 DDRC ECC engine，该 DDRC

ECC engine 采用 RS 码算法对内存颗粒中的数据进行检测, 若 RS 码 DDRC ECC engine 没有检测到错误, 则内存控制器直接将数据返回给 CPU, 流程提前结束; 若 RS 码 DDRC ECC engine 检测到的发生错误的 symbol 的数量在 t 个以内, 则内存控制器通过 RS 码 DDRC ECC engine 将数据纠正并返回给 CPU, 流程提前结束; 若 RS 码 DDRC ECC engine 检测到的发生错误的 symbol 的数量大于 t , 则超出了 RS 码 DDRC ECC engine 的纠错能力上限。图 5 中步骤 3 (对应于上述步骤 403): 若任一 CPU core 访问到的内存数据的错误超出 RS 码 DDRC ECC engine 的纠错能力上限, 则 RS 码 DDRC ECC engine 触发同步异常 (synchronous external abort, SEA), MCU 进入基本输入输出系统 (basic input output system, BIOS) 处理流程。MCU 首先判断该地址是否属于 DMA 地址, 执行对应操作: 如果属于 DMA 地址则放弃本次纠错, 返回错误状态, 执行原处理流程; 如果不属于 DMA 地址, 则继续执行后续纠错流程。图 5 中步骤 4 (对应于上述步骤 403) DMC 对内存条的读写进程进行反压, 防止数据被其他读写进程改写造成数据不一致。图 5 中步骤 5 (对应于上述步骤 404): MCU 通过 BPG 对内存颗粒 (DRAM) 发起读取, 将读取数据与 RS 码 DDRC ECC engine 记录的错误数据进行比对, 如一致, 则继续处理流程, 否则, 表明反压生效前数据被改写过, 提前结束流程, 并返回错误状态。MCU 直接通过 MRR 命令读取各个内存颗粒的第一寄存器 (MRx.uv), 记录各个内存颗粒的片上纠错引擎得到的纠错状态信息。图 5 中步骤 6 (对应于上述步骤 405 和步骤 406): 当且仅当某个颗粒的纠错状态为 10B, 其余均为 00B 或 01B 时, MCU 利用 EC 算法进行数据纠错; 其余场景流程提前结束, 返回不可纠。EC 算法纠错完成后, MCU 通过 BPG, 将纠正后的数据回写到内存颗粒, 并修改中断返回向量, 触发系统对读取操作对应的内存地址的重读, 也即是对该缓存行 (cacheline) 的重读。图 5 中步骤 7 (对应于上述步骤 407 和步骤 408) MCU 解除反压, 反压解除后, 内存控制器发起对发生错误的内存颗粒中的数据的重读, 并利用 RS 码 DDRC ECC engine 对重读到的数据进行校验, 若校验通过则返回数据, 否则上报不可纠错。

需要说明的是, 上述步骤 401 至步骤 408 是以 DDR5 SDRAM x8 内存条为例进行说明的, 本申请实施例提供的内存纠错方法还可应用于 DDR5 SDRAM x4 内存条中, DDR5 SDRAM x4 内存条中的内存纠错方法与上述步骤 401 至 407 所示的流程同理, 不同之处在于: DDR5 SDRAM x4 内存条中, 响应于一次内存读取操作, 每个内存颗粒向内存控制器输出 64bit 数据, 该 64bit 数据分布于两个码字, 每个内存颗粒为一个码字贡献 32bit 数据, 由于每个内存颗粒有 4 个对外的数据线路, 每个内存颗粒向内存控制器输出的数据也即是 4×8 的数据块; 此外, DDR5 SDRAM x4 内存条包括 8 个数据颗粒和 2 个纠错码颗粒, 也即是内存条中冗余的内存颗粒的数量为 2, 当且仅当发生错误的内存颗粒的数量小于或等于 2 时, 内存控制器能够对发生错误的内存颗粒中的数据进行纠正, 当发生错误的内存颗粒的数量大于 2 时, 内存控制器结束数据纠错流程, 向 CPU 返回不可纠错。在 DDR5 SDRAM x4 内存条中, 内存控制器对数据进行纠错过程包括下述步骤 A 至步骤 C。

步骤 A、将 8 个数据颗粒中的数据分别记为 $D1$ 、 $D2$...、 $D8$, 将 2 个纠错码颗粒 (冗余颗粒) 中的数据记为 $C1$ 和 $C2$, 其中, $D1$ - $D8$ 、 $C1$ 和 $C2$ 均为 4×8 的数据块。根据 RS 码算法也即是 EC 码的性质, 则存在一个矩阵 H 用于生成码字, 其中 H 可见元素均为 4×4 的分块矩阵, 该过程可以由下述 (4) 表示。

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ B1 & B2 & B3 & B4 & B5 & B6 & B7 & B8 \\ B9 & B10 & B11 & B12 & B13 & B14 & B15 & B16 \end{bmatrix} \begin{bmatrix} D1 \\ D2 \\ D3 \\ D4 \\ D5 \\ D6 \\ D7 \\ D8 \end{bmatrix} = \begin{bmatrix} D1 \\ D2 \\ D3 \\ D4 \\ D5 \\ D6 \\ D7 \\ D8 \\ C1 \\ C2 \end{bmatrix} \quad (4)$$

步骤 B、对于任意两个内存颗粒发生错误的情况, 将这两个内存颗粒对应的矩阵行从矩阵 H 中去掉, 矩阵依旧是满秩矩阵 H' , 该满秩矩阵 H' 存在逆矩阵 H'^{-1} 。例如, 内存颗粒 $D1$ 和 $C1$ 发生错误, 则将 $D1$ 和 $C1$ 对应的矩阵行从矩阵 H 中去掉, 该过程可以由下述公式 (5) 表示。

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ B9 & B10 & B11 & B12 & B13 & B14 & B15 & B16 \end{bmatrix} \begin{bmatrix} D1 \\ D2 \\ D3 \\ D4 \\ D5 \\ D6 \\ D7 \\ D8 \end{bmatrix} = \begin{bmatrix} D2 \\ D3 \\ D4 \\ D5 \\ D6 \\ D7 \\ D8 \\ C2 \end{bmatrix} \quad (5)$$

步骤 C、将所读取到的数据左乘逆矩阵 H'^{-1} ，将得到纠错后的数据，该过程可以由下述公式（6）表示。

$$\begin{bmatrix} D1 \\ D2 \\ D3 \\ D4 \\ D5 \\ D6 \\ D7 \\ D8 \end{bmatrix} = H'^{-1} \begin{bmatrix} D2 \\ D3 \\ D4 \\ D5 \\ D6 \\ D7 \\ D8 \\ C2 \end{bmatrix} \quad (6)$$

DDR5 SDRAM x4 内存条中的内存纠错方法的其他步骤与 DDR5 SDRAM x8 内存条中的内存纠错方法中的步骤同理，同理之处不再赘述。

上述方法中，片上纠错引擎对数据进行第一次纠错，并将内存颗粒的纠错状态信息写入第一寄存器中，使得内存控制器能够通过读取该第一寄存器中的纠错状态信息定位到发生错误的颗粒，颗粒级的纠错和系统级的纠错能够相互配合，实现了对冗余资源的充分利用，有利于在固定冗余配置下提高内存系统的纠错能力：内存控制器获取到数据后，对数据进行第二次纠错，与仅依靠片上纠错引擎进行纠错相比，能够对片上纠错引擎未检测出的静默错误以及误纠的错误进行纠正，从而降低内存颗粒中的数据静默错误风险和数据误纠风险；若内存控制器纠错失败，则内存控制器从内存条中的寄存器中读取内存颗粒的纠错状态信息，基于纠错状态信息，确定发生错误的内存颗粒，进而对数据进行第三次纠错，由于利用了颗粒级纠错的结果，与仅依靠内存控制器进行纠错相比，能够提高内存控制器的纠错能力，从而提高了内存系统的纠错能力和内存条中数据的可靠性。对于 x4 内存，该方法可实现对两个内存颗粒纠错（dual-chipkill），对于 x8 内存条，该方法可实现对单个内存颗粒纠错（chipkill）。

需要说明的是，上述实施例是针对内存条的应用形态中的模组形态为例进行说明的，在一些实施例中，本申请实施例提供的内存纠错方法可以应用于颗粒表贴形态的内存条中。在另一些实施例中，本申请实施例提供的内存纠错方法对于内存颗粒包括片上纠错引擎，且内存控制器使用 RS 码进行纠错的场景也适用，例如 DDR6 或 LPDDR4 做边带 ECC（sideband ECC）的场景，本申请实施例不局限于上述示出的具体场景。

图 6 是本申请实施例提供的一种内存控制器，该内存控制器包括数据获取模块 601 和数据纠错模块 602。

数据获取模块 601，用于响应于对内存条的读取操作，从内存条中获取数据；

数据纠错模块 602，用于对该数据进行纠错，若纠错失败，则确定内存条中发生错误的内存颗粒，基于该发生错误的内存颗粒，对该数据进行纠错。

可选地，数据纠错模块 602 包括：

读取单元，用于对数据进行纠错，若纠错失败，则从多个内存颗粒的第一寄存器中，读取多个内存颗粒的纠错状态信息；

确定单元，用于若任一内存颗粒的纠错状态信息指示该内存颗粒发生错误，则确定该内存颗粒为发生错误的内存颗粒。

可选地，该读取单元用于：

对该数据进行纠错，若纠错失败，则内存控制器对内存条的读写进程进行反压；

再次获取内存中的数据，若两次获取到的数据一致，则从多个内存颗粒的第一寄存器中，读取多个内存颗粒的纠错状态信息。

可选地，该数据纠错模块包括：

纠错单元，用于若发生错误的内存颗粒的数量小于或等于内存条中纠错码颗粒的数量，则基于发生错误的内存颗粒，对该数据进行纠错。

可选地，该内存控制器还包括：

回写模块，用于将纠错后的数据回写至内存颗粒中；

重读模块，用于对该读取操作对应的内存地址进行重读；

校验模块，用于对重读到的数据进行校验，若校验通过，则将重读到的数据返回给处理器，若校验不通过，则内存控制器向处理器上报错误。

可选地，内存颗粒的纠错状态由第一寄存器的目标位的取值来表示，第一寄存器的目标位的取值所表示的内存颗粒的纠错状态为无错误、可纠正错误和不可纠正错误中任一项。

可选地，内存颗粒的纠错状态由第一寄存器的目标位的占用状态来表示，第一寄存器的目标位的占用状态为空表示内存颗粒的纠错状态为无错误或可纠正错误，第一寄存器的目标位的占用状态不为空表示内存颗粒的纠错状态为不可纠正错误。

可选地，发生错误的内存颗粒是指纠错状态为可纠正错误的内存颗粒和纠错状态为不可纠正错误的内存颗粒中的至少一种内存颗粒。

其中，数据获取模块 601 和数据纠错模块 602 均可以通过软件实现，或者可以通过硬件实现。示例性的，接下来以数据获取模块 601 为例，介绍数据获取模块 601 的实现方式。类似的，数据纠错模块 602 的实现方式可以参考数据获取模块 601 的实现方式。

模块作为软件功能单元的一种举例，数据获取模块 601 可以包括运行在计算实例上的代码。其中，计算实例可以包括物理主机（计算设备）、虚拟机、容器中的至少一种。进一步地，上述计算实例可以是一台或者多台。例如，数据获取模块 601 可以包括运行在多个主机/虚拟机/容器上的代码。需要说明的是，用于运行该代码的多个主机/虚拟机/容器可以分布在相同的区域（region）中，也可以分布在不同的 region 中。进一步地，用于运行该代码的多个主机/虚拟机/容器可以分布在相同的可用区（availability zone, AZ）中，也可以分布在不同的 AZ 中，每个 AZ 包括一个数据中心或多个地理位置相近的数据中心。其中，通常一个 region 可以包括多个 AZ。

同样，用于运行该代码的多个主机/虚拟机/容器可以分布在同一个虚拟私有云（virtual private cloud, VPC）中，也可以分布在多个 VPC 中。其中，通常一个 VPC 设置在一个 region 内，同一 region 内两个 VPC 之间，以及不同 region 的 VPC 之间跨区通信需在每个 VPC 内设置通信网关，经通信网关实现 VPC 之间的互连。

模块作为硬件功能单元的一种举例，数据获取模块 601 可以包括至少一个计算设备，如服务器等。或者，数据获取模块 601 也可以是利用专用集成电路（application-specific integrated circuit, ASIC）实现或可编程逻辑器件（programmable logic device, PLD）实现的设备等。其中，上述 PLD 可以是复杂程序逻辑器件（complex programmable logical device, CPLD）、现场可编程门阵列（field-programmable gate array, FPGA）、通用阵列逻辑（generic array logic, GAL）或其任意组合实现。

数据获取模块 601 包括的多个计算设备可以分布在相同的 region 中，也可以分布在不同的 region 中。数据获取模块 601 包括的多个计算设备可以分布在相同的 AZ 中，也可以分布在不同的 AZ 中。同样，数据纠错模块 602 包括的多个计算设备可以分布在同一个 VPC 中，也可以分布在多个 VPC 中。其中，该多个计算设备可以是服务器、ASIC、PLD、CPLD、FPGA 和 GAL 等计算设备的任意组合。

需要说明的是，在其他实施例中，上述模块负责实现的步骤可根据需要指定，通过上述模块分别实现上述内存纠错方法中不同的步骤来实现上述装置的全部功能。也即是，上述实施例提供的内存纠错装置在实现内存纠错方法时仅以上述各功能模块的划分进行举例说明，实际应用中，可以根据需要而将上述功能分配由不同的功能模块完成，即将装置的内部结构划分成不同的功能模块，以完成以上描述的全

部或者部分功能。另外，上述实施例提供的装置与相应方法实施例属于同一构思，其具体实现过程详见方法实施例，这里不再赘述。

本申请还提供一种计算设备，该计算设备包括内存控制器和内存条，该内存条用于暂存数据，该内存控制器用于执行上述方法实施例所提供的内存纠错方法。

图7是本申请实施例提供的一种计算设备的结构示意图，如图7所示，计算设备700包括：总线701、处理器702、存储器703和通信接口704。处理器702、存储器703和通信接口704之间通过总线701通信。计算设备700可以是服务器或终端设备。应理解，本申请不限定计算设备700中的处理器、存储器的个数。

总线701可以是外设部件互连标准(peripheral component interconnect, PCI)总线或扩展工业标准结构(extended industry standard architecture, EISA)总线等。总线可以分为地址总线、数据总线、控制总线等。为便于表示，图7中仅用一条线表示，但并不表示仅有一根总线或一种类型的总线。总线701可包括在计算设备700各个部件(例如，存储器703、处理器702、通信接口704)之间传送信息的通路。

处理器702可以包括中央处理器(central processing unit, CPU)、图形处理器(graphics processing unit, GPU)、微处理器(micro processor, MP)或者数字信号处理器(digital signal processor, DSP)等处理器中的任意一种或多种。

存储器703可以包括易失性存储器(volatile memory, VM)，例如随机存取存储器(random access memory, RAM)。存储器703还可以包括非易失性存储器(non-volatile memory, NVM)，例如只读存储器(read-only memory, ROM)，快闪存储器，机械硬盘(hard disk drive, HDD)或固态硬盘(solid state drive, SSD)。

存储器703中存储有可执行的程序代码，处理器702执行该可执行的程序代码以分别实现前述数据获取模块601和数据纠错模块602的功能，从而实现内存纠错方法。也即，存储器703上存有用于执行内存纠错方法的指令。

通信接口704使用例如但不限于网络接口卡、收发器一类的收发模块，来实现计算设备700与其他设备或通信网络之间的通信。

本申请实施例提供了一种处理器，该处理器包括内存控制器和计算核心，该处理器用于执行如上述实施例所提供的内存纠错方法，该计算核心用于对内存颗粒中的数据执行计算操作。

本申请实施了提供了一种包含指令的计算机程序产品，该计算机程序产品可以是包含指令的，能够运行在计算设备上或被储存在任何可用介质中的软件或程序产品。当该计算机程序产品在至少一个计算设备上运行时，使得至少一个计算设备如上述实施例所提供的内存纠错方法。

本申请实施例提供了一种计算机可读存储介质，该计算机可读存储介质可以是计算设备能够存储的任何可用介质或者是包含一个或多个可用介质的数据中心等数据存储设备。该可用介质可以是磁性介质，(例如，软盘、硬盘、磁带)、光介质(例如，DVD)、或者半导体介质(例如固态硬盘)等。该计算机可读存储介质包括指令，当该指令由计算设备集群执行时，该计算设备集群执行如上述实施例所提供的内存纠错方法。

需要说明的是，本申请所涉及的信息(包括但不限于用户设备信息、用户个人信息等)、数据(包括但不限于用于分析的数据、存储的数据、展示的数据等)以及信号，均为经用户授权或者经过各方充分授权的，且相关数据的收集、使用和处理需要遵守相关国家和地区的相关法律法规和标准。例如，本申请中涉及到的数据都是在充分授权的情况下获取的。

本领域普通技术人员可以意识到，结合本文中所公开的实施例中描述的各方法步骤和单元，能够以电子硬件、计算机软件或者二者的结合来实现，为了清楚地说明硬件和软件的可互换性，在上述说明中已经按照功能一般性地描述了各实施例的步骤及组成。这些功能究竟以硬件还是软件方式来执行，取决于技术方案的特定应用和设计约束条件。本领域普通技术人员可以对每个特定的应用来使用不同方法来

实现所描述的功能，但是这种实现不应认为超出本申请的范围。

所属领域的技术人员可以清楚地了解到，为了描述的方便和简洁，上述描述的系统、装置和单元的具体工作过程，可以参见前述方法实施例中的对应过程，在此不再赘述。

在本申请所提供的几个实施例中，应该理解到，所揭露的系统、装置和方法，可以通过其它的方式实现。例如，以上所描述的装置实施例仅是示意性的，例如，该单元的划分，仅仅为一种逻辑功能划分，实际实现时可以有另外的划分方式，例如多个单元或组件可以结合或者可以集成到另一个系统，或一些特征可以忽略，或不执行。另外，所显示或讨论的相互之间的耦合或直接耦合或通信连接可以是通过一些接口、装置或单元的间接耦合或通信连接，也可以是电的，机械的或其它的形式连接。

该作为分离部件说明的单元可以是或者也可以不是物理上分开的，作为单元显示的部件可以是或者也可以不是物理单元，即可以位于一个地方，或者也可以分布到多个网络单元上。可以根据实际的需要选择其中的部分或者全部单元来实现本申请实施例方案的目的。

另外，在本申请各个实施例中的各单元可以集成在一个处理单元中，也可以是各个单元单独物理存在，也可以是两个或两个以上单元集成在一个单元中。上述集成的单元既可以采用硬件的形式实现，也可以采用软件单元的形式实现。

该集成的单元如果以软件功能单元的形式实现并作为独立的产品销售或使用，可以存储在一个计算机可读存储介质中。基于这样的理解，本申请的技术方案本质上或者说对现有技术做出贡献的部分，或者该技术方案的全部或部分可以以软件产品的形式体现出来，该计算机软件产品存储在一个存储介质中，包括若干指令用以使得一台计算设备（可以是个人计算机，服务器，或者计算设备等）执行本申请各个实施例中方法的全部或部分步骤。而前述的存储介质包括：U 盘、移动硬盘、只读存储器（read-only memory, ROM）、随机存取存储器（random access memory, RAM）、磁碟或者光盘等各种可以存储程序代码的介质。

本申请中术语“第一”“第二”等字样用于对作用和功能基本相同的相同项或相似项进行区分，应理解，“第一”、“第二”、“第 n”之间不具有逻辑或时序上的依赖关系，也不对数量和执行顺序进行限定。还应理解，尽管以下描述使用术语第一、第二等来描述各种元素，但这些元素不应受术语的限制。这些术语只是用于将一元素与另一元素区别分开。例如，在不脱离各种示例的范围的情况下，第一内存颗粒可以被第二内存颗粒，并且类似地，第二内存颗粒可以被第一内存颗粒。第一内存颗粒和第二内存颗粒都可以是内存颗粒，并且在某些情况下，可以是单独且不同的内存颗粒。

本申请中术语“至少一个”的含义是指一个或多个，本申请中术语“多个”的含义是指两个或两个以上，例如，多个第一内存颗粒是指两个或两个以上的第一内存颗粒。本文中术语“系统”和“网络”经常可互换使用。

还应理解，术语“如果”可被解释为意指“当...时”（“when”或“upon”）或“响应于确定”或“响应于检测到”。类似地，根据上下文，短语“如果确定...”或“如果检测到[所陈述的条件或事件]”可被解释为意指“在确定...时”或“响应于确定...”或“在检测到[所陈述的条件或事件]时”或“响应于检测到[所陈述的条件或事件]”。

以上描述，仅为本申请的具体实施方式，但本申请的保护范围并不局限于此，任何熟悉本技术领域的人员在本申请揭露的技术范围内，可轻易想到各种等效的修改或替换，这些修改或替换都应涵盖在本申请的保护范围之内。因此，本申请的保护范围应以权利要求的保护范围为准。

在上述实施例中，可以全部或部分地通过软件、硬件、固件或者其任意组合来实现。当使用软件实现时，可以全部或部分地以计算机程序产品的形式实现。该计算机程序产品包括一个或多个计算机程序指令。在计算机上加载和执行该计算机程序指令时，全部或部分地产生按照本申请实施例中的流程或功能。该计算机可以是通用计算机、专用计算机、计算机网络、或者其他可编程装置。

该计算机指令可以存储在计算机可读存储介质中，或者从一个计算机可读存储介质向另一个计算机可读存储介质传输，例如，该计算机程序指令可以从一个网站站点、计算机、服务器或数据中心通过有线或无线方式向另一个网站站点、计算机、服务器或数据中心进行传输。该计算机可读存储介质可以是计算机能够存取的任何可用介质或者是包含一个或多个可用介质集成的服务器、数据中心等数据存储设备。该可用介质可以是磁性介质（例如软盘、硬盘、磁带）、光介质（例如，数字视频光盘（digital video disc, DVD）、或者半导体介质（例如固态硬盘）等。

本领域普通技术人员可以理解实现上述实施例的全部或部分步骤可以通过硬件来完成，也可以通过

程序来指令相关的硬件完成，该程序可以存储于一种计算机可读存储介质中，上述提到的存储介质可以是只读存储器，磁盘或光盘等。

以上实施例仅用以说明本申请的技术方案，而非对其限制；尽管参照前述实施例对本申请进行了详细的说明，本领域的普通技术人员应当理解：其依然可以对前述各实施例所记载的技术方案进行修改，或者对其中部分技术特征进行等同替换；而这些修改或者替换，并不使相应技术方案的本质脱离本申请各实施例技术方案的范围。

权 利 要 求 书

1.一种内存纠错方法，其特征在于，所述方法包括：

响应于对内存条的读取操作，内存控制器从所述内存条中获取数据；

所述内存控制器对所述数据进行纠错，若纠错失败，则确定所述内存条中发生错误的内存颗粒，基于所述发生错误的内存颗粒，对所述数据进行纠错。

2.根据权利要求1所述的方法，其特征在于，所述内存控制器对所述数据进行纠错，若纠错失败，则确定所述内存条中发生错误的内存颗粒，包括：

所述内存控制器对所述数据进行纠错，若纠错失败，则从所述多个内存颗粒的第一寄存器中，读取所述多个内存颗粒的纠错状态信息；

若任一所述内存颗粒的纠错状态信息指示所述内存颗粒发生错误，则所述内存控制器确定所述内存颗粒为发生错误的内存颗粒。

3.根据权利要求2所述的方法，其特征在于，所述内存控制器对所述数据进行纠错，若纠错失败，则从所述多个内存颗粒的第一寄存器中，读取所述多个内存颗粒的纠错状态信息，包括：

所述内存控制器对所述数据进行纠错，若纠错失败，则所述内存控制器对所述内存条的读写进程进行反压；

所述内存控制器再次获取所述内存条中的数据，若两次获取到的数据一致，则从所述多个内存颗粒的第一寄存器中，读取所述多个内存颗粒的纠错状态信息。

4.根据权利要求1至3中任一项所述的方法，其特征在于，所述基于所述发生错误的内存颗粒，对所述数据进行纠错，包括：

若发生错误的内存颗粒的数量小于或等于所述内存条中纠错码颗粒的数量，则基于所述发生错误的内存颗粒，对所述数据进行纠错。

5.根据权利要求1至4中任一项所述的方法，其特征在于，在所述基于所述发生错误的内存颗粒，对所述数据进行纠错之后，所述方法还包括：

所述内存控制器将纠错后的数据回写至所述内存颗粒中；

所述内存控制器对所述读取操作对应的内存地址进行重读，对重读到的数据进行校验；

若校验通过，则所述内存控制器将所述重读到的数据返回给处理器；

若校验不通过，则所述内存控制器向处理器上报错误。

6.根据权利要求2至5中任一项所述的方法，其特征在于，所述内存颗粒的纠错状态由所述第一寄存器的目标位的取值来表示，所述第一寄存器的目标位的取值所表示的所述内存颗粒的纠错状态为无错误、可纠正错误和不可纠正错误中任一项。

7.根据权利要求2至5中任一项所述的方法，其特征在于，所述内存颗粒的纠错状态由所述第一寄存器的目标位的占用状态来表示，所述第一寄存器的目标位的占用状态为空表示所述内存颗粒的纠错状态为无错误或可纠正错误，所述第一寄存器的目标位的占用状态不为空表示所述内存颗粒的纠错状态为不可纠正错误。

8.根据权利要求1至7中任一项所述的方法，其特征在于，所述发生错误的内存颗粒是指纠错状态为可纠正错误的内存颗粒和纠错状态为不可纠正错误的内存颗粒中的至少一种内存颗粒。

9.根据权利要求2至8中任一项所述的方法，其特征在于，所述内存颗粒包括片上纠错引擎，所述方法还包括：

响应于对所述内存条的读取操作，所述片上纠错引擎获取所述内存颗粒中的数据；

所述片上纠错引擎对所获取到的数据进行纠错，将所述内存颗粒的纠错状态信息写入所述内存颗粒的第一寄存器中，将纠错后的数据输出至所述内存控制器。

10.一种内存纠错方法，其特征在于，由内存条执行，所述内存条包括多个内存颗粒，每个所述内存颗粒包括第一寄存器，所述方法包括：

对所述内存颗粒中的数据进行纠错，得到所述内存颗粒的纠错状态信息；

将所述纠错状态信息写入所述内存颗粒的第一寄存器中。

11.一种内存控制器，其特征在于，所述内存控制器包括：

数据获取模块，用于响应于对内存条的读取操作，从所述内存条中获取数据；

数据纠错模块，用于对所述数据进行纠错，若纠错失败，则确定所述内存条中发生错误的内存颗粒，基于所述发生错误的内存颗粒，对所述数据进行纠错。

12.根据权利要求 11 所述的内存控制器，其特征在于，所述数据纠错模块包括：

读取单元，用于对所述数据进行纠错，若纠错失败，则从所述多个内存颗粒的第一寄存器中，读取所述多个内存颗粒的纠错状态信息；

确定单元，用于若任一所述内存颗粒的纠错状态信息指示所述内存颗粒发生错误，则确定所述内存颗粒为发生错误的内存颗粒。

13.根据权利要求 12 所述的内存控制器，其特征在于，所述读取单元用于：

对所述数据进行纠错，若纠错失败，则对所述内存条的读写进程进行反压；

再次获取所述内存条中的数据，若两次获取到的数据一致，则从所述多个内存颗粒的第一寄存器中，读取所述多个内存颗粒的纠错状态信息。

14.根据权利要求 11 至 13 中任一项所述的内存控制器，其特征在于，所述数据纠错模块包括：

纠错单元，用于若发生错误的内存颗粒的数量小于或等于所述内存条中纠错码颗粒的数量，则基于所述发生错误的内存颗粒，对所述数据进行纠错。

15.根据权利要求 11 至 14 中任一项所述的内存控制器，其特征在于，所述装置还包括：

回写模块，用于将纠错后的数据回写至所述内存颗粒中；

重读模块，用于对所述读取操作对应的内存地址进行重读；

校验模块，用于对重读到的数据进行校验，若校验通过，则将所述重读到的数据返回给处理器，若校验不通过，则向处理器上报错误。

16.根据权利要求 12 至 15 中任一项所述的内存控制器，其特征在于，所述内存颗粒的纠错状态信息为所述第一寄存器的目标位的取值，所述第一寄存器的目标位的取值所指示的所述内存颗粒的纠错状态为无错误、可纠正错误和不可纠正错误中任一项。

17.根据权利要求 12 至 15 中任一项所述的内存控制器，其特征在于，所述内存颗粒的纠错状态信息为所述第一寄存器的目标位的占用状态，所述第一寄存器的目标位的占用状态为空指示所述内存颗粒的纠错状态为无错误或可纠正错误，所述第一寄存器的目标位的占用状态不为空指示所述内存颗粒的纠错状态为不可纠正错误。

18.根据权利要求 11 至 17 中任一项所述的内存控制器，其特征在于，所述发生错误的内存颗粒是指纠错状态为可纠正错误的内存颗粒和纠错状态为不可纠正错误的内存颗粒中的至少一种内存颗粒。

19.一种内存条，其特征在于，所述内存条包括多个内存颗粒，每个所述内存颗粒包括第一寄存器，所述内存条用于执行如权利要求 10 所述的内存纠错方法。

20.一种内存控制器，其特征在于，所述内存控制器用于执行如权利要求 1 至权利要求 9 中任一项所述的内存纠错方法。

21.一种处理器，其特征在于，所述处理器包括内存控制器和计算核心，所述处理器用于执行如权利要求 1 至权利要求 9 中任一项所述的内存纠错方法，所述计算核心用于对内存颗粒中的数据执行计算操作。

22.一种计算设备，其特征在于，包括内存控制器和内存条，所述内存条用于暂存数据，所述内存控制器用于执行如权利要求 1 至权利要求 9 中任一项所述的内存纠错方法。

23.一种计算机可读存储介质，其特征在于，所述计算机可读存储介质用于存储至少一段程序代码，所述至少一段程序代码用于执行如权利要求 1 至权利要求 9 中任一项所述的内存纠错方法。