(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 116738510 A
(43)申请公布日 2023.09.12

(21)申请号 202310220244.X
(22)申请日 2023.03.08
(30)优先权数据
63/318,534 2022.03.10 US
17/874,962 2022.07.27 US
(71)申请人 三星电子株式会社
地址 韩国京畿道
(72)发明人 C-C·C-J·A·吴 S·J·文卡塔 Y·D·金
(74)专利代理机构 北京市柳沈律师事务所 11105
专利代理师 邵亚丽
(51)Int.Cl.
G06F 21/79(2013.01)
G06F 21/60(2013.01)
G06F 21/62(2013.01)

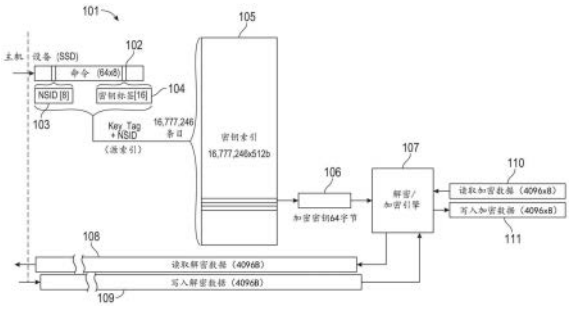
权利要求书4页 说明书18页 附图15页

(54)发明名称

有效地获得存储在地址空间中的信息的系统和方法

(57)摘要

提供了一种存储器设备的方法。该方法包括:由存储器设备的控制器通过用散列函数对包括名称空间标识(NSID)和密钥标签的源索引进行散列来生成散列索引;由控制器访问对应于散列索引的第一存储地址;由控制器将对应于第一存储地址的标签部分与源索引进行比较以识别是否存在冲突;以及响应于识别是否存在冲突,由控制器从第一存储地址获得信息。



CN 116738510 A

1.一种存储器设备的方法,包括:

由所述存储器设备的控制器通过用散列函数对包括名称空间标识NSID和密钥标签的源索引进行散列来生成第一预定数量的比特的散列索引;

由所述控制器访问对应于所述散列索引的第一存储地址;

由所述控制器将对应于所述第一存储地址的标签部分与所述源索引进行比较以识别是否存在冲突;以及

响应于识别是否存在所述冲突,由所述控制器从所述第一存储地址获得信息。

2.根据权利要求1所述的方法,其中,所述第一存储地址被包括在第一存储器中,并且

其中,由所述控制器将对应于所述第一存储地址的所述标签部分与所述源索引进行比较以识别是否存在所述冲突还包括:

响应于识别出存在所述冲突,由所述控制器将所述第一存储器中的所述第一存储地址链接到所述第一存储器中的空的第二存储地址。

3.根据权利要求1所述的方法,其中,所述第一存储地址被包括在第一存储器中,

其中,由所述控制器将对应于所述第一存储地址的所述标签部分与所述源索引进行比较以识别是否存在所述冲突还包括:

响应于识别出存在所述冲突,由所述控制器将所述第一存储器中的所述第一存储地址链接到第二存储器中的空的第二存储地址。

4.根据权利要求1所述的方法,其中,指向所述第一存储地址的重定向索引被包括在第一类型的存储器中,其中所述第一类型的存储器包括比第二类型的存储器更多的条目,并且

其中,所述散列索引附加地包括被配置为指向所述第一类型的存储器中的所述重定向索引的比特的一部分,其中所述重定向索引指向所述第二类型的存储器中的所述第一存储地址。

5.根据权利要求4所述的方法,还包括:

由所述控制器通过至少对在所述第二类型的存储器上访问的最新条目进行计数来确定最频繁使用MFU条目;以及

由所述控制器调整所述重定向索引中的条目以指向所述第二类型的存储器上的所述MFU条目,并将第一链接列表的行尾TOL分配为指向第二链接列表的行首HOL条目。

6.根据权利要求1所述的方法,其中,所述散列函数是双散列函数,并且

其中,由所述控制器通过用所述散列函数对包括所述NSID和所述密钥标签的所述源索引进行散列来生成所述第一预定数量的比特的所述散列索引包括:

由所述控制器通过使用第一双散列函数对所述源索引进行散列来生成第一散列索引,

由所述控制器将对应于所述第一存储地址的标签部分与所述源索引进行比较以识别是否存在所述冲突,以及

响应于识别出存在所述冲突,由所述控制器通过使用第二双散列函数对所述源索引进行散列来生成第二散列索引。

7.根据权利要求1所述的方法,其中,所述散列函数是双散列函数,并且

其中,由所述控制器通过用所述散列函数对包括所述NSID和所述密钥标签的所述源索引进行散列来生成所述第一预定数量的比特的所述散列索引包括:

并发地由所述控制器通过使用第一双散列函数对所述源索引进行散列来生成第一散列索引并通过使用第二双散列函数对所述源索引进行散列来生成第二散列索引。

8.根据权利要求1所述的方法,其中,所述散列函数是双散列函数,并且

其中,由所述控制器通过用所述散列函数对包括所述NSID和所述密钥标签的所述源索引进行散列来生成所述第一预定数量的比特的所述散列索引包括:

由所述控制器基于预定条件确定根据顺序双散列模式还是并发双散列模式来生成所述散列索引,

其中,所述预定条件基于先前吞吐量持续时间、作为动态选择的先前查找动态随机存取存储器DRAM事务计数、静态选择或基于与阈值相比的先前选择计数的累积的渐进动态选择中的至少一个。

9.根据权利要求1所述的方法,其中,指向所述第一存储地址的行首HOL链接列表索引被包括在静态随机存取存储器SRAM中,其中所述SRAM包括与动态随机存取存储器DRAM中的条目数量相比至少1.5x的条目数量,并且

其中,所述散列索引除了所述第一预定数量的比特之外还附加地包括第二预定数量的比特,并且指向HOL链接列表中指向DRAM中的所述第一存储地址的条目。

10.一种系统,包括:

存储器;和

控制器,被配置为:

通过用散列函数对包括名称空间标识NSID和密钥标签的源索引进行散列来生成第一预定数量的比特的散列索引;

访问对应于所述散列索引的第一存储地址;

将对应于所述第一存储地址的标签部分与所述源索引进行比较以识别是否存在冲突;以及

响应于识别是否存在所述冲突,从所述第一存储地址获得信息。

11.根据权利要求10所述的系统,其中,所述第一存储地址被包括在第一存储器中,并且

其中,将对应于所述第一存储地址的所述标签部分与所述源索引进行比较以识别是否存在所述冲突还包括:

响应于识别出存在所述冲突,由所述控制器将所述第一存储器中的所述第一存储地址链接到所述第一存储器中的空的第二存储地址。

12.根据权利要求10所述的系统,其中,所述第一存储地址被包括在第一存储器中,

其中,将对应于所述第一存储地址的所述标签部分与所述源索引进行比较以识别是否存在所述冲突还包括:

响应于识别出存在所述冲突,由所述控制器将所述第一存储器中的所述第一存储地址链接到第二存储器中的空的第二存储地址。

13.根据权利要求10所述的系统,其中,指向所述第一存储地址的重定向索引被包括在第一类型的存储器中,其中所述第一类型的存储器包括比第二类型的存储器更多的条目,并且

其中,所述散列索引附加地包括被配置为指向所述第一类型的存储器中的所述重定向

索引的比特的一部分,其中所述重定向索引指向所述第二类型的存储器中的所述第一存储地址。

14.根据权利要求13所述的系统,其中,所述控制器还被配置为:

通过至少对在所述第二类型的存储器上访问的最新条目进行计数来确定最频繁使用MFU条目;以及

调整所述重定向索引中的条目以指向所述第二类型的存储器上的所述MFU条目,并将第一链接列表的行尾TOL分配为指向第二链接列表的行首HOL条目。

15.根据权利要求10所述的系统,其中,所述散列函数是双散列函数,并且

其中,通过用所述散列函数对包括所述NSID和所述密钥标签的所述源索引进行散列来生成所述第一预定数量的比特的所述散列索引包括:

由所述控制器通过使用第一双散列函数对所述源索引进行散列来生成第一散列索引,

由所述控制器将对应于所述第一存储地址的标签部分与所述源索引进行比较以识别是否存在所述冲突,以及

响应于识别出存在所述冲突,由所述控制器通过使用第二双散列函数对所述源索引进行散列来生成第二散列索引。

16.根据权利要求10所述的系统,其中,所述散列函数是双散列函数,并且

其中,通过用所述散列函数对包括所述NSID和所述密钥标签的所述源索引进行散列来生成所述第一预定数量的比特的所述散列索引包括:

并发地由所述控制器通过使用第一双散列函数对所述源索引进行散列来生成第一散列索引并通过使用第二双散列函数对所述源索引进行散列来生成第二散列索引。

17.根据权利要求10所述的系统,其中,所述散列函数是双散列函数,并且

其中,通过用所述散列函数对包括所述NSID和所述密钥标签的所述源索引进行散列来生成所述第一预定数量的比特的所述散列索引包括:

由所述控制器基于预定条件确定根据顺序双散列模式还是并发双散列模式来生成所述散列索引,

其中,所述预定条件基于先前吞吐量持续时间、作为动态选择的先前查找动态随机存取存储器DRAM事务计数、静态选择或基于与阈值相比的先前选择计数的累积的渐进动态选择中的至少一个。

18.根据权利要求10所述的系统,其中,指向所述第一存储地址的行首HOL链接列表索引被包括在静态随机存取存储器SRAM中,其中所述SRAM包括与动态随机存取存储器DRAM中的条目数量相比至少1.5x的条目数量,并且

其中,所述散列索引除了所述第一预定数量的比特之外还附加地包括第二预定数量的比特,并且指向DRAM中的所述第一存储地址。

19.一种存储设备,包括:

控制器;和

存储介质,

其中,所述控制器被配置为:

通过用散列函数对包括名称空间标识NSID和密钥标签的源索引进行散列来生成散列索引;

访问对应于所述散列索引的第一存储地址；

将对应于所述第一存储地址的标签部分与所述源索引进行比较以识别是否存在冲突；以及

响应于识别是否存在所述冲突，从所述第一存储地址获得信息。

20.根据权利要求19所述的存储设备，其中，所述第一存储地址被包括在第一存储器中，并且

其中，将对应于所述第一存储地址的所述标签部分与所述源索引进行比较以识别是否存在所述冲突还包括：

响应于识别出存在所述冲突，由所述控制器将所述第一存储器中的所述第一存储地址链接到所述第一存储器中的空的第二存储地址。

有效地获得存储在地址空间中的信息的系统和方法

[0001] 相关申请的交叉引用

[0002] 本申请基于2022年3月10日提交的序列号为63/318,534的美国临时专利申请并要求其优先权,该申请的全部内容通过引用并入本文。

技术领域

[0003] 本公开总体上涉及存储器设备和加密密钥系统。

背景技术

[0004] 文件是应用用于管理用户数据的一种类型的数据结构。因此,数据的高效处理、存储、安全和一般管理对于信息技术(IT)系统非常重要。应用使用并依赖于文件系统、操作系统(OS)和其他此类系统软件来进行文件管理和访问相关操作。

[0005] 用于现代IT基础设施的存储设备(例如,诸如固态驱动器(SSD)的持久数据存储设备)越来越受欢迎,因为各种应用(例如,物联网(IOT)、社交网络、自动驾驶车辆等)正在生成大量数据。基于NAND闪存介质的SSD存储设备也是IT基础设施中的组件。

[0006] 当应用需要数据时,将从存储设备获取存储文件的所需数据部分。由于存储设备可以提供高性能持久性存储,一些系统性能瓶颈已转向系统软件层。此类操作的文件读取延迟是应用(例如,游戏和在线购物应用)性能和最终用户体验的重要因素。

发明内容

[0007] 做出本公开以至少解决上述缺点并至少提供下述优点。

[0008] 根据本公开的一方面,提供了一种存储器设备的方法。该方法包括:由存储器设备的控制器通过用散列函数对包括名称空间标识(NSID)和密钥标签的源索引进行散列来生成散列索引;由控制器访问对应于散列索引的第一存储地址;由控制器将对应于第一存储地址的标签部分与源索引进行比较以识别是否存在冲突;以及响应于识别是否存在冲突,由控制器从第一存储地址获得信息。

[0009] 根据本公开的另一方面,一种系统包括存储器和控制器,该控制器被配置为:通过用散列函数对包括NSID和密钥标签的源索引进行散列来生成散列索引;访问对应于散列索引的第一存储地址;将对应于第一存储地址的标签部分与源索引进行比较以识别是否存在冲突;以及响应于识别是否存在冲突,从第一存储地址获得信息。

[0010] 根据本公开的另一方面,一种存储设备包括控制器和存储介质。该控制器被配置为:通过用散列函数对包括NSID和密钥标签的源索引进行散列来生成散列索引;访问对应于散列索引的第一存储地址;将对应于第一存储地址的标签部分与源索引进行比较以识别是否存在冲突;以及响应于识别是否存在冲突,从第一存储地址获得信息。

附图说明

[0011] 结合附图进行的以下详细描述将使本公开的某些实施例的上述和其他方面、特征

和优点更加明显,其中:

[0012] 图1是示出根据实施例的每输入输出密钥(Key Per Input Output,KPIO)系统的框图;

[0013] 图2是示出根据实施例的散列冲突解决的框图;

[0014] 图3是示出根据实施例的降低冲突发生的似然性的散列条目扩展的框图;

[0015] 图4是示出根据实施例的请求条目的重定向表的框图;

[0016] 图5是根据实施例的添加/删除/重新排序/遍历命令的比较图2、图3和图4的实施例的存储器访问属性的表;

[0017] 图6是示出根据实施例的用于将行首(Head Of Line,HOL)有效地重新排序到最频繁使用(MFU)条目的环形链接列表的框图;

[0018] 图7是示出根据实施例的双散列机制的框图;

[0019] 图8是示出根据实施例的使用并发多散列探测(probe)的双重散列机制的框图;

[0020] 图9是示出根据实施例的能够降低冲突可能性的双散列机制的框图;

[0021] 图10是示出根据实施例的五种多散列索引探测类型和对应的应用场景的表;

[0022] 图11是根据实施例的添加/删除/重新排序/遍历命令的比较图2、图3、图4、图7、图8和图9的实施例的存储器访问属性的表;

[0023] 图12是示出根据实施例的在不等待完成第一查找请求的情况下执行第二查找请求的时序图;

[0024] 图13是示出根据实施例的用于执行本申请中描述的实施例的结构组件的存储器系统;

[0025] 图14是示出根据实施例的访问存储条目的流程图;并且

[0026] 图15示出了根据实施例的网络环境中的电子设备。

具体实施方式

[0027] 以下,参考附图详细描述本公开的实施例。应当注意,尽管在不同的附图中示出了相同的元件,但是相同的元件将由相同的附图标记表示。在以下描述中,仅提供诸如详细配置和组件的具体细节以帮助总体理解本公开的实施例。因此,本领域技术人员应当明白,在不脱离本公开的范围的情况下,可以对本文描述的实施例进行各种改变和修改。此外,为了清楚和简洁,省略了对公知功能和构造的描述。下面描述的术语是考虑到本公开中的功能而定义的术语,并且可以根据用户、用户的意图或习惯而不同。因此,应根据本说明书中的内容确定术语的定义。

[0028] 本公开可以具有各种修改和各种实施例,其中以下参考附图详细描述实施例。然而,应当理解,本公开不限于实施例,而是包括本公开范围内的所有修改、等同和替代。

[0029] 尽管包括序数(诸如第一、第二等)的术语可用于描述各种元件,但结构元件不受这些术语的限制。这些术语用于区分一个元件和另一元件。例如,在不脱离本公开的范围的情况下,第一结构元件可以被称为第二结构元件。类似地,第二结构元件也可以称为第一结构元件。如本文所用,术语“和/或”包括一个或多个相关项目的任何和所有组合。

[0030] 本文使用的术语仅用于描述本公开的各种实施例,但不旨在限制本公开。除非上下文另有明确规定,否则单数形式应包括复数形式。在本公开中,应当理解,术语“包括”或

“具有”表示特征、数字、步骤、操作、结构元件、部件或其组合的存在，并且不排除添加一个或多个其他特征、数字，步骤、操作，结构元件、部件或它们的组合的存在或可能性。

[0031] 除非有不同的定义，否则本文使用的所有术语具有与本公开所属领域的技术人员所理解的含义相同的含义。诸如在一般使用的词典中定义的那些术语的含义应被解释为与相关领域的上下文含义相同，并且除非在本公开中明确定义，否则不应被解释成具有理想或过度正式的含义。

[0032] 根据实施例的电子设备可以是利用存储设备的各种类型的电子设备之一。电子设备可以包括例如便携式通信设备(例如，智能电话)、计算机、便携式多媒体设备、便携式医疗设备、相机、可穿戴设备或家用电器。根据本公开的一个实施例，电子设备不限于上述那些。

[0033] 本公开中使用的术语不旨在限制本公开，而是旨在包括对应实施例的各种变化、等同物或替换。关于附图的描述，可以使用类似的附图标记来指代类似的或相关的元件。除非相关上下文另有明确说明，否则与一个项目相对应的名词的单数形式可能包括一个或多个事物。如本文所使用的，诸如“A或B”、“A和B中的至少一个”、“A或B中的至少一个”，“A、B或C”，“A、B和C中的至少一个”以及“A、B或C中的至少一个”等短语中的每一个都可以包括在这些短语中的相应短语中一起列举的项目的所有可能组合。如本文所使用的，诸如“第1”、“第2”、“第一”和“第二”的术语可用于将对应的组件与另一组件区分开来，但不旨在在其他方面(例如，重要性或顺序)限制组件。意图是，如果一个元件(例如，第一元件)被称为“与另一元件(例如，第二元件)耦合”、“耦合到另一元件”、“与另一元件连接”或“连接到另一元件”，无论是否使用术语“可操作地”或“可通信地”，则这表明元件可以直接(例如，有线)、无线或通过第三元件与另一元件耦合。

[0034] 如本文所使用的，术语“模块”可以包括以硬件、软件、固件或其组合实现的单元，并且可以与其他术语互换使用，例如，“逻辑”、“逻辑块”、“部件”和“电路”。模块可以是单个集成组件，或者是适合于执行一个或多个功能的最小单元或部分。例如，根据一个实施例，模块可以以专用集成电路(ASIC)的形式实现。

[0035] 对于安全的数据存储，应保护用户可访问的数据，这可能需要加密和解密。存储器存储系统可以允许主机设备为每个输入/输出(IO)命令选择加密或解密密钥，称为每IO密钥(KPIO)。

[0036] 自加密驱动器(SED)可以对用户可访问的数据执行连续加密。这是使用存储设备生成/保存在持久介质中的少量密钥以接口速度完成的。KPIO可以利用大量加密密钥来管理并安全地下载到非易失性存储器子系统中。用户数据的加密可以按每个命令进行(每个命令可以请求使用不同的密钥)。

[0037] 为了有效地存储加密数据，可以使用散列函数。散列函数采用任意长的字节字符串，并产生较短的固定大小结果。例如，第一密钥类型可以被输入到散列函数以输出较短的固定大小结果(例如，散列值)。散列值可以是存储在存储器中的特定元素的索引。散列表可用于将对应于第一密钥类型和值对的散列值存储在可通过其索引访问的列表中。散列表中的值(与第一密钥类型的散列值配对)可以是用于KPIO的第二密钥类型。因此，当散列表中的值是用于KPIO的第二密钥类型时，这些值可以用于加密/解密数据以执行读取/写入命令。此外，散列表中的值(与第一密钥类型的散列值配对)可以包括一个或多个数据对象。

[0038] 存储器存储系统通常需要大量可能的密钥用于存储,但是在存储器读取/写入操作期间可以使用其中的相对小的一部分。因此,存储器存储可能是低效的,因为可能分配大于所需的存储器空间用于存储大量密钥,从而增加了此类设计的成本和功耗。此外,低效的设计也可能导致数据的读取和写入缓慢。

[0039] 提出了一种在大的源地址空间中有效地分配存储器空间和查找信息(加密或解密密钥)的方法。该方法可以扩展查找表大小以提高吞吐量,而不必增加存储器大小。进而,这可以使查找迭代最小化,并能够利用每秒SSD输入输出操作(IOPS)来缩放加密密钥查找,存储器的访问时间基本上是固定的。因此,本公开可以以最小的成本提供加密密钥处理的改进的产品性能。

[0040] 此外,本公开提出了将密钥打包到较小的存储空间中并以唯一的顺序存储的存储。可以将原始访问索引转换为新访问索引以优化存储空间。提出了一种散列函数访问-索引转换方法,有利地降低了存储信息的成本。然而,一个挑战是提供一种提供高访问性能同时适应低存储器访问时间的散列冲突解决方案。

[0041] 图1是示出根据实施例的每输入输出密钥(KPIO)系统的框图。在整个图1中,一些组件包括与每个组件相对应的比特或字节的数量(例如,命令(64x8)和/或加密密钥64字节)。对应于每个组件的比特或字节的数量是示例性的,并且被提供以帮助理解附图。具体地,每个组件的大小可用于说明目的,以帮助理解组件之间的物理关系。每个组件的实际比特数、字节数或条目数可能会有所不同,这取决于系统的需要。

[0042] KPIO系统101可以是{密钥,值}系统。{密钥,值}系统可以将数据存储为密钥-值对的集合,其中密钥用作值的唯一标识符。值可以由固定数量的比特(例如,512比特)组成,并且可以表示用于加密或解密数据的安全密钥(例如,私钥)(注意,“{密钥,值}”对中包括的密钥可以是与“值”字段中包括的安全密钥不同的密钥类型)。高级加密标准(AES)解密密钥可以存储在易失性只读存储器中,诸如动态只读存储器(DRAM)。代表安全密钥的值(例如,512比特安全密钥)可以通过名称空间标识符(NSID)+密钥标签来寻址。

[0043] 参考图1,主机可以向存储设备(例如,SSD(例如,闪存驱动器))发送读取或写入命令102。命令102可以包括NSID+密钥标签。NSID 103可以识别要读取或写入的存储器。NSID 103可以由8比特或更多比特组成。密钥标签104可以是用于索引特定NSID的加密/解密密钥的密钥索引。系统可以由多个NSID组成,不同NSID中的密钥标签可以重叠。NSID 103和密钥标签104可以被组合以形成数据结构105(例如,密钥索引、非重叠{密钥,值}索引、加密索引、解密密钥索引、查找表或数组)。数据结构105可以是AES索引。数据结构105的大小可以由16,777,246(条目)x512比特组成,并且可以用于检索安全密钥106(例如,AES加密密钥或加密密钥)。安全密钥106可以是64字节(512比特)。

[0044] 对此,数据结构105是大的。数据结构105包括大约1600万个可能的条目。然而,系统可以使用64000个有效条目。因此,总存储器大小可以是1千兆字节(GB),总有效存储器大小可以为4兆字节(MB)。由于有效条目的数量相对于存储器的总大小很小,因此存储器是稀疏的。具有预定义的固定/最大数量的有效密钥(例如,64000)是KPIO加密的特性,使用有限存储器大小和在大且稀疏的源地址空间中的最大查找次数,从而增加了依靠高效吞吐量计算来快速查找和检索有效密钥的重要性。

[0045] 可以检索安全密钥106并将其发送到解密/加密引擎107。解密数据(例如,未加密

数据)可以被读取108,或者解密数据可以被从主机写入109到SSD,并且安全密钥106可以被解密/加密引擎107用来读取加密数据110或写入加密数据111。加密数据110或111可以是4096x8比特,并存储在memflash中。例如,加密数据110或111可以存储在远程服务器(例如,云服务器)中。

[0046] 直接地址查找表(LUT)可用于在KPIO系统中的密钥索引105中存储密钥值。直接地址LUT将每个命令102寻址到密钥索引105中的特定地址空间,然而AES加密密钥索引105的大小必须非常大,因为密钥索引105的大小的固定部分可以包括有效条目。

[0047] 使用内容可寻址存储器(CAM)查找可以是在KPIO系统中有效获取或检索密钥值的解决方案。有利地,CAM查找减少了密钥索引105的存储器存储大小并实现了快速吞吐量。不幸的是,CAM使用大量的逻辑门来输入或检索密钥值,这很昂贵。例如,对于64000个有效条目,可能需要64000x24比特比较器来执行逻辑功能,以将源索引重定向到密钥索引105中的有效密钥地址空间。

[0048] 此外,散列寻址可用于将源索引命令散列到有限大小,以识别散列密钥索引的密钥存储地址(例如,将索引命令从24比特减少到16比特)。散列寻址通过使用散列密钥索引有利地减小了存储器的存储器大小,因为散列索引可以以减小的长度被紧凑地存储。不幸的是,由于多个源密钥(例如,要输入到散列函数中的来自一个或多个源的密钥)可以链接到单个散列索引,因此从同一散列索引检索有效值可能需要冲突解决操作,这可能会降低吞吐量。

[0049] 散列冲突解决操作可以涉及在使用散列索引来标识主存储器中的存储索引时执行比较操作。如果比较操作将对应于主存储器中的存储地址的散列索引与源索引(例如,NSID+密钥标签)匹配,则可以使用链接列表来指定辅存储器地址空间,以将对应于主要存储器中的存储地址的散列索引与源索引相关联。通过这种方式,可以使用链接列表将多个条目与同一散列索引相关联。因此,主存储器的大小可以减小到较小的大小,因为可以只包括有效的条目。然而,当发生冲突并指定了链接列表时,必须扩展存储器大小并将其指定为第二存储器。此外,由于比较操作,可能会发生冲突,访问存储器可能需要多次操作才能从链接列表中获取数据。

[0050] 外围组件互连快速(PCIe)链接可用于访问存储器。PCIe4x4标准可使用200万(M)IOPS或500纳秒(ns)进行一次查找(例如,读取或写入)。PCIe4x6标准可以使用8M IOPS或125ns进行一次查找。DRAM存取可以是大约50-70ns。因此,使用PCIe4x6标准,每个命令只能执行1-2次查找,这大大提高了查找速度。

[0051] 因此,本申请提出了使用少量逻辑门有效地将密钥存储在密钥值索引中的解决方案(例如,通过提高每总存储器大小存储更多有效密钥条目的比率)。此外,本申请提出的解决方案能够执行快速吞吐量,从而最小化用于访问存储在DRAM中的密钥索引的查找次数。

[0052] 此外,从主机设备访问SSD并组成密钥(例如,从图1的命令102)的时间到识别有效加密密钥(例如,识别图1中的安全密钥106)的时间的延迟应该理想地减少和/或最小化。本申请中提出的解决方案实现了这一点。

[0053] 此外,存储在存储器索引(例如,图1的密钥索引105)中的{密钥,值}对可以被存储以有效地允许调整,诸如添加、插入、删除和/或重新排序{密钥,值}对的查找顺序,以便最小化操作的数量。

[0054] 图2是示出根据实施例的散列冲突解决的框图。在整个图2中，一些组件包括与每个组件相对应的比特或字节的数量(例如，NSID[8]和/或标签65336(条目)x24比特)。对应于每个组件的比特或字节的数量是示例性的，并且被提供以帮助理解附图。具体地，每个组件的大小可用于说明目的，以帮助理解组件之间的物理关系。每个组件的实际比特数、字节数或条目数可能会有所不同，这取决于系统的需要。

[0055] 使用链接列表的散列冲突解决可以将单独的链式链接列表与散列表合并，以节省主存储器中的未使用槽的存储器空间。散列冲突解决可以应用于KPIO系统，使得源索引(NSID和密钥标签)被散列并用于标识与存储在DRAM中的加密或解密密钥相对应的存储地址，如下面在图2中所描述的。

[0056] 图2中描述的操作可以由存储在存储器中的控制器、处理器或计算机实现的指令执行。

[0057] 参考图2，包括NSID和密钥标签的源索引可以从24比特散列到16比特，以生成散列索引。散列索引表示主存储器的存储地址(例如，散列表)。在步骤201，访问1的标签部分与NSID的散列索引和表示主存储器的存储地址的密钥标签进行比较(例如，由控制器)。访问1的标签部分可以表示DRAM条目中的访问1的本地地址，并与包括NSID和密钥标签的源索引进行比较。

[0058] 如图2所示，通过步骤201的比较确定存在冲突(失配1)。因此，第一指针被分配为将访问1链接到主存储器(DRAM)中的另一存储地址，访问2。

[0059] 在步骤202，将存储地址的标签部分(访问2)与NSID和密钥标签进行比较。如图2所示，步骤202的比较确定存在另一冲突(失配2)，并且第二指针被分配为将访问2链接到主存储器(DRAM)中的另一存储地址，访问3。

[0060] 在步骤203，将存储地址的标签部分(访问3)与NSID和密钥标签进行比较。如图2所示，步骤203的比较确定不存在冲突(匹配3)，并且可以访问加密密钥。

[0061] 当存储在主存储器中的散列索引与源索引(例如，NSID+密钥标签)匹配时发生冲突时，可以执行图2的“折叠”操作(例如，创建到同一存储器内的散列索引的链接列表)。如果发生冲突，则将主存储器中的多个存储器空间配置为同一散列索引。

[0062] 有三种类型的条目可以存储在主存储器中：列表头(HOL)条目、空条目和占用第二加条目。HOL条目是与散列索引(例如，链接列表的开始)相关联的第一存储地址。空条目是与散列索引无关并且否则未使用的存储地址。占用第二加条目是与已经与另一存储地址(例如，与相同散列索引相关联的第二或第三条目)相关联的散列索引相关联的存储地址。

[0063] 如上所述，辅存储器的链接列表功能被“折叠”到主存储器中，从而减小了存储器的总大小。然而，如果添加与占用链接列表条目(例如，占用第二链接列表条目)冲突的条目，则将需要重新配置占用链接列表，因此添加、插入和删除操作可能导致对存储器的访问变慢。在重新配置链接列表链时，添加、插入和删除操作可能会导致主存储器处于动态状态，从而限制其可访问性并导致停机。因此，降低冲突发生的似然性是可取的。当添加或插入与HOL冲突的条目时，不需要重新配置，因为该条目可以被分配给空地址，该空地址可以链接为链接列表的行尾(Tail Of Line，TOL)。

[0064] 图3是示出根据实施例的降低冲突发生的似然性的散列条目扩展的框图。在整个图3中，一些组件包括与每个组件相对应的比特或字节的数量(例如，NSID[8]和/或加密密

钥512b)。对应于每个组件的比特或字节的数量是示例性的,并且被提供以帮助理解附图。具体地,每个组件的大小可用于说明目的,以帮助理解组件之间的物理关系。每个组件的实际比特数或字节数可能会有所不同,这取决于系统的需要。

[0065] 图3中描述的操作可以由存储在存储器中的控制器、处理器或计算机实现的指令执行。

[0066] 参考图3,包括NSID和密钥标签的条目可以从24比特散列到18比特,以生成散列索引。在这种情况下,散列函数具有18比特,这比图2中提供的实施例中呈现的散列函数多2比特。此外,与图2中提供的实施例相比,图3中的主存储器扩展了四倍。此外,还提供了辅存储器。如下所述,将散列函数增加2比特以及将主存储器增加4倍将冲突发生的可能性降低了1/4。

[0067] 散列索引表示主存储器的存储地址(例如,散列表)。在步骤301,将访问1的标签部分与NSID和密钥标签进行比较。访问1的标签部分可以表示访问1条目的本地地址,并与包括NSID和密钥标签的源索引进行比较。

[0068] 如图3所示,通过步骤301的比较,确定存在冲突(失配1)。因此,第一指针被分配为将访问1链接到辅存储器中的另一存储地址,访问2。

[0069] 由于主存储器扩展了四倍,因此存储器的利用率降低了1/4,从而将负载因子降低了1/4,并减少了冲突发生的机会。

[0070] 此外,在步骤302中,将存储地址的标签部分(访问2)与NSID和密钥标签进行比较。步骤302的比较确定存在另一冲突(失配2),并且第二指针被分配为将访问2链接到辅存储器中的另一存储地址,访问3。

[0071] 在步骤303中,将存储地址的标签部分(访问3)与NSID和密钥标签进行比较。如图2所示,步骤303的比较确定不存在冲突(匹配3),并且可以访问加密密钥。

[0072] 图4是示出根据实施例的请求条目的重定向表的框图。在整个图4中,一些组件包括与每个组件相对应的比特或字节的数量(例如,NSID[8]和/或加密密钥512b)。对应于每个组件的比特或字节的数量是示例性的,并且被提供以帮助理解附图。具体地,每个组件的大小可用于说明目的,以帮助理解组件之间的物理关系。每个组件的实际比特数或字节数可能会有所不同,这取决于系统的需要。

[0073] 图4中描述的操作可以由存储在存储器中的控制器、处理器或计算机实现的指令执行。

[0074] 参考图4,主存储器由65536x4比特散列索引重定向表(散列重定向表)组成,这显著减小了主存储器的大小,因为条目的存储(例如,存储加密/解密密钥)和标签组件没有存储在主存储器上。在这种情况下,主存储器可以由静态只读存储器(SRAM)组成,并且位于具有比访问DRAM更快的读取/写入速度的IC(例如,ASIC)上。DRAM可以位于集成电路(IC)芯片之外,并且可以相对于SRAM的大小较大。因此,散列重定向表可以称为“片上”散列重定向表,其可以被快速访问以获得重定向地址。

[0075] 散列重定向表是重定向到辅存储器(例如,DRAM)的地址的重定向索引。散列重定向表被配置为使得散列重定向表的条目指向辅存储器中的对应第一条目(包括对应的AES存储和标签)。散列重定向表的输入可以是18比特,并且输出是16比特。

[0076] 由于在该配置中每个条目的AES存储组件和标签组件存储在辅存储器中而不是主

存储器中,因此存储在主存储器中的信息(例如,散列重定向表)的大小被减小。

[0077] 因此,当使用散列重定向表时,控制器(例如,处理器)使用18比特散列源索引来标识重定向索引,以指向辅存储器中对应条目的第一条目(例如,使用HOL指针)。如图4所示,在步骤401,访问1的标签部分与NSID和密钥标签进行比较,并由HOL指针指向。访问1的标签部分可以表示访问1条目的本地地址(例如,DRAM中的访问1条目的存储器地址),并与包括NSID和密钥标签的源索引进行比较。

[0078] 如图4所示,通过步骤401的比较确定存在冲突(失配1)。因此,第一指针被分配为将访问1链接到辅存储器中的另一条目,访问2。

[0079] 在步骤402中,将存储地址的标签部分(访问2)与NSID和密钥标签进行比较。如图4所示,步骤402的比较确定不存在冲突(匹配2),并且可以访问加密密钥(或解密密钥)。

[0080] 访问存储器地址的过程可以称为执行“跳跃(hop)”或“跳转(jump)”。每次使用比较功能(例如,步骤401和步骤402)时,执行跳跃或跳转。因此,由于散列重定向表用于快速识别指向辅存储器中的对应条目的HOL指针,因此可以通过经由片上散列重定向表快速访问重定向索引,并随后访问两个条目(两个跳跃)来获得加密(或解密)密钥,从而减少存储器访问时间。

[0081] 如上所述,本公开提供了一种可以减少访问数据所需的跳跃数量的配置。由于PCIe4x6标准可以使用8M IOPS或125ns进行一次查找,因此减少跳跃数量改进了访问AES加密密钥的吞吐量时间,这改进了总体处理时间。

[0082] 跳跃(例如,跳转)数量与执行查找所遇到的冲突数量密切相关。也就是说,每个冲突都可能导致执行跳跃。此外,取决于存储器类型(例如,DRAM与SRAM),每个跳跃可能需要不同的时间量(例如,典型的KPIO操作应该在125ns或更少的时间内执行,如果DRAM访问是50-70ns,则执行KPIO查找可能需要2个或更少跳转)。

[0083] 此外,如以上提供的示例中所述,散列重定向表的输入是18比特,因为向散列索引添加了2个附加的比特。16至18比特扩展可将冲突的可能性降低1/4。此外,可以使用1比特扩展,以便散列索引从16比特扩展到17比特,这可以将冲突的可能性降低1/2。此外,非整数比特宽扩展是可能的(例如,将64k条目的16比特散列索引扩展为198k条目的16.5比特散列索引),这可以将冲突的可能性降低1/3。

[0084] 其他散列索引扩展大小也是可能的。例如,散列索引可以是8比特,并扩展2个附加的比特,以将冲突的可能性降低1/4。实际上,可以使用任何数量的比特,只要可以扩展附加的比特以使用散列重定向表来降低冲突的可能性。增加散列索引的大小可以有利地用于降低在查找期间发生冲突的可能性。

[0085] 尽管散列索引扩展降低了在查找期间发生冲突的似然性,但扩展存储器大小是必要的,以适应散列索引的更多比特数。然而,存储器相对较小的扩展(例如,从16比特长度扩展到18比特长度)的好处是冲突发生的似然性降低了1/4,这对于在2个跳转或更少时间内执行KPIO查找非常有价值(例如,对于PCIE-G6x4,当可以在125ns或更少时间访问存储器时)。

[0086] 此外,在上述示例中,重定向表的长度仅扩展了2比特,特别是不包括条目的内容(密钥值),这将导致存储器扩展大小更大。

[0087] 图5是根据实施例的添加/删除/重新排序/遍历命令的比较图2、图3和图4的实施

例的存储器访问属性的表。为了便于描述，图2-图4中描述的实施例可以称为“链式散列”实施例。

[0088] SRAM比其他RAM类型(诸如DRAM)相对更快。它还消耗更少的电力。DRAM是一种RAM，允许你将数据的每比特存储在特定IC内的单独电容器中。因此，与SRAM相比，SRAM具有更低的访问时间，并且更快，而DRAM具有更高的访问时间并且更慢。SRAM可以是片上存储器的形式，但DRAM具有片外存储器的特性。DRAM.1可以是指DRAM的第一存储器地址，DRAM.2可以是指SRAM的第二存储器地址。

[0089] 参考图5，考虑到合并RAM方法(例如，图2)，为了获得加密密钥，访问DRAM.1，并且有50％的冲突机会。因此，执行两个跳转的平均存储器访问时间为1.5x(1x+.5x)。

[0090] RAM方法的条目大小可以是65,536(64B+2B+3B)。

[0091] 考虑到存储器扩展方法(例如，图3)，为了获得加密密钥，访问DRAM.1，并且由于主存储器被扩展了四倍，因此有12.5％的冲突机会。因此，执行两个跳转的平均存储器访问时间可以是1.125x(1x+0.125x)。

[0092] 扩展方法的条目大小可以是(4+1)x65,536(64B+2B+3B)。

[0093] 考虑到存储器扩展和重定向方法(例如，图4)，为了获得加密或解密密钥，访问SRAM。当访问SRAM时执行第一跳跃。与DRAM相比，SRAM的访问速度非常快。例如，SRAM访问时间可能需要2ns-5ns，DRAM可能需要50ns-70ns。假设标准化SRAM值为DRAM的.04x。然而，更快的SRAM访问时间是可能的。当访问DRAM.1和DRAM.2时，执行第二跳跃和第三跳跃。因此，执行一个重定向跳跃和两个DRAM跳跃的平均存储器访问时间可以是1.165x。

[0094] 存储器扩展和重定向方法的条目大小可以是65,536(64B+2B+3B)(对于扩展部分(例如，辅存储器))+4x65,536(2B)(对于重定向部分(例如，主存储器))。

[0095] 对于图2-图4所描述的每个方法，添加和删除HOL操作分别包括添加到行尾(TOL)以及删除和重新链接链接列表。

[0096] 现在将描述MFU条目。如果AES存储中有三个条目，并且其中一个条目比其他条目使用得更频繁，则最小化后续查找次数可能是有利的，对条目进行重新排序，使得MFU条目作为HOL先于其他条目排序(例如，MFU条目应该是访问1而不是访问2或访问3)。

[0097] 对于合并RAM方法和扩展方法，将MFU重新排序到HOL需要将MFU条目复制到HOL并改变链接列表中条目的顺序。对于扩展和重定向方法，可以通过使用环形链接列表更快地完成将MFU重新排序到HOL，如下所述。

[0098] 图6是示出根据实施例的用于将行首(HOL)有效地重新排序到MFU条目的环形链接列表的框图。在整个图6中，一些组件包括与每个组件相对应的比特或字节的数量(例如，NSID[8]和/或加密密钥512b)。对应于每个组件的比特或字节的数量是示例性的，并且被提供以帮助理解附图。具体地，每个组件的大小可用于说明目的，以帮助理解组件之间的物理关系。每个组件的实际比特数或字节数可能会有所不同，这取决于系统的需要。

[0099] 图6中描述的操作可以由存储在存储器中的控制器、处理器或计算机实现的指令执行。例如，可以使用现场可编程门阵列(FPGA)、专用集成电路(ASIC)、通用计算机或远程处理系统(例如，云计算系统)来实现由控制器执行的指令。

[0100] 图6所示的实施例类似于图4的实施例，但附加地包括能够将HOL旋转到链接列表的MFU条目的环形链接列表。链接列表可以是封闭链接列表，使得最后一个条目指向HOL。

[0101] 参考图6,使用环形链接列表来标识MFU条目,以将由散列重定向表标识的地址的指针改变为链接列表的MFU条目(例如,NEW.HOL.Next_pointer被改变为MFU)。使用环形链接列表大大降低了冲突的可能性,因为MFU条目是将被访问的链接列表中最可能的条目。因此,使用MFU条目减少了获得加密或解密密钥所需的时间。

[0102] MFU单元可用于对访问链接列表中的条目的次数进行计数。例如,MFU单元可以基于滚动(滑动窗口)计数最后16次查找以确定MFU条目。此外,MFU可以生成每个链接列表的直方图和每个条目的计数(Bin)(例如,每个Bin中的最后访问中的16或32个)。可以使用散列重定向表来存储和/或排序由MFU单元获得的信息。也就是说,可以在DRAM或SRAM中存储和/或排序由MFU获得的信息。

[0103] 使用环形列表将HOL指向链接列表的MFU条目有利地不需要读取/写入操作,不改变链接列表顺序,因此不需要以访问AES存储的形式中断。因此,通过使用环形列表实现的益处可能是降低了冲突的可能性,从而产生了用于从AES存储快速获得加密密钥的更快方法。

[0104] 图7是示出根据实施例的双散列机制的框图。在整个图7中,一些组件包括与每个组件相对应的比特或字节的数量(例如,NSID[8]和/或加密密钥512b)。对应于每个组件的比特或字节的数量是示例性的,并且被提供以帮助理解附图。具体地,每个组件的大小可用于说明目的,以帮助理解组件之间的物理关系。每个组件的实际比特数或字节数可能会有所不同,这取决于系统的需要。

[0105] 图7中描述的操作可以由存储在存储器中的控制器、处理器或计算机实现的指令执行。

[0106] 参考图7,双散列机制可以能够解决一个或多个冲突,并且在不使用链接列表的情况下访问AES加密密钥。双散列机制可以应用于KPIO系统。

[0107] 双散列机制可以使用两个或更多个散列函数。在图7中,示出了散列值H1、H2、H3、H4和H5。每个散列值可以是两个基线散列函数(例如,H(K)和H'(K))的组合,并且每个散列值都可以生成与主存储器中的五个随机分配地址相对应的分离的和非相关的散列索引。

[0108] 源NSID和密钥标签索引可以使用从24到16比特的双散列值H1进行散列,以生成对应于主存储器中的存储地址(例如,对应于访问1的存储地址)的散列索引H1_index。在步骤701中,系统将访问1的标签部分与源NSID和密钥标签索引进行比较。如图7所示,访问1的标签部分被确定为与NSID和密钥标签对冲突(失配1)。当在步骤701中确定冲突时,系统用第二散列值H2对源NSID和密钥标签索引进行散列,以生成对应于主存储器中的存储地址(例如,对应于访问2的存储地址)的散列H2_index。由于H2是与H1不同的散列值,因此散列索引指示的存储地址将导致不同的位置。在步骤702中,系统将访问2的标签部分与源NSID和密钥标签索引进行比较。如图7所示,访问2的标签部分被确定为与NSID和密钥标签对冲突(失配2)。当在步骤702中确定冲突时,系统用第三散列值H3对源NSID和密钥标签索引进行散列,以生成与主存储器中的不同存储地址相对应的散列H3_index。在步骤703中,系统将访问3的标签部分与源NSID和密钥标签索引进行比较,如图7所示,该比较确定存储地址对应于加密密钥(匹配3)。

[0109] 关于图2-图6,通过存储器识别的链接列表用于解决冲突。如上关于图7所述,使用双散列函数来解决冲突。此外,对于图7的双散列函数,主存储器中的每个存储地址是HOL

(因为链接列表中不包括任何地址)。

[0110] 图8是示出根据实施例的使用并发多散列探测的双重散列机制的框图。在整个图8中,一些组件包括与每个组件相对应的比特或字节的数量(例如,NSID[8]和/或加密密钥512b)。对应于每个组件的比特或字节的数量是示例性的,并且被提供以帮助理解附图。具体地,每个组件的大小可用于说明目的,以帮助理解组件之间的物理关系。每个组件的实际比特数或字节数可能会有所不同,这取决于系统的需要。

[0111] 图8中描述的操作可以由存储在存储器中的控制器、处理器或计算机实现的指令执行。

[0112] 参考图8,图8的并发多散列探测双散列机制基于图7的双散列机制,但附加地应用源索引的并发散列。具体地,当NSID+密钥标签源索引被散列时,而不是使用第一散列函数H1进行第一散列,并且如果有使用第二散列函数H2的冲突散列,则并发多散列探测同时使用散列函数H1和散列函数H2对NSID+键标签源索引进行散列。

[0113] 也就是说,如图8所示,使用散列函数H1、H2、H3、H4和H5同时对NSID+密钥标签源索引进行散列,从而分别同时输出H1_index、H2_index和H3_index、H4_index以及H5_index。H1_index、H2_index和H3_index,H4_index以及H5_index用于同时识别和访问存储器中的存储地址(对应于访问1、访问2、访问3、访问4和访问5的AES存储地址)。对应于访问1、访问2和访问3的标签本地索引被同时访问并且在步骤801、802和803中与NSID+密钥标签源索引同时比较,以确定匹配,从而输出对应于主存储器中找到匹配的地址的AES加密密钥。

[0114] 因此,步骤801、802和803类似于图7中的步骤701、702和703,不同之处在于同时执行步骤801、801和803中执行的比较。

[0115] 图9是示出根据实施例的能够降低冲突可能性的双散列机制的框图。在整个图9中,一些组件包括与每个组件相对应的比特或字节的数量(例如,NSID[8]和/或加密密钥512b)。对应于每个组件的比特或字节的数量是示例性的,并且被提供以帮助理解附图。具体地,每个组件的大小可用于说明目的,以帮助理解组件之间的物理关系。每个组件的实际比特数或字节数可能会有所不同,这取决于系统的需要。

[0116] 图9中描述的操作可以由存储在存储器中的控制器、处理器或计算机实现的指令执行。

[0117] 图9中提出的配置降低了冲突发生的似然性,从而最小化了对NSID+密钥标签源索引进行散列所需的双散列函数的平均数量。执行更少的双散列函数是优选的,因为它减少了识别与NSID+密钥标签源索引的匹配以检索AES加密密钥所需的处理量。

[0118] 参考图9,NSID+密钥标签源索引用双散列函数H1、H2和H3进行散列以输出18比特索引。18比特索引被映射到HOL链接列表表。如图9所示,提供了四个HOL重定向表,从而将冲突的可能性降低四倍。可以在SRAM(例如,片上)上提供HOL链接列表表。因此,可以使用HOL链接列表表快速指向AES存储中的条目,降低冲突的可能性(并且增加匹配的可能性)。此外,探测的数量不限于上述示例。例如,可以使用两个或更多个探测来执行多探测实现。

[0119] 图10是示出根据实施例的五种多散列索引探测类型和对应的应用场景的表。图10中提到的五种多散列索引探测类型仅仅是示例性的,并且可以应用附加的探测类型。此外,应用场景是类似地示例性的,并且可以应用附加的应用场景。

[0120] 参考图10,第一类型的多散列索引探测可以是顺序多散列探测(例如,类似于图7

所示的实施例),其可以由DRAM控制器使用,该DRAM控制器针对每个读取命令执行页丢失(page-miss)或行开始(start-of-row)。

[0121] 另一种类型的多散列索引探测可以是并发发布探测模式(例如,类似于图8中所示的实施例),其可以由DRAM控制器使用,该DRAM控制器针对每个读取或写入命令背对背执行页命中访问。

[0122] 另一种类型的多散列索引探测可以是程序可选择的顺序或并发探测模式。在这种情况下,可以基于指令(例如,来自DRAM控制器的指令)选择顺序或并发探测类型。

[0123] 另一种类型的多散列索引探测可以是动态自适应顺序和并发模式。在这种情况下,探测模式可以取决于最后数据条目或预定数量的最后数据条目(例如,基于最后16个条目的滑动窗口来应用计算)。例如,如果使用顺序探测访问DRAM中的条目导致访问条目的多个冲突,则当访问下一个条目时,探测模式可以从顺序切换到并发,使得可以更有效地访问数据条目。

[0124] 另一种类型的多散列索引探测可以是渐进动态自适应模式。在这种情况下,探测模式可以在顺序和并发之间切换,并且基于最后数据条目或预定数量的最后数据条目来改变应用于源索引的双散列函数的数量(例如,基于最后16个条目的滑动窗口来应用计算)。例如,如果使用顺序散列并且访问数据条目导致冲突,则下一个条目的探测模式可以从顺序切换到并发,并且应用的双散列函数的数量可以增加1。

[0125] 图11是根据实施例的添加/删除/重新排序/遍历命令的比较图2、图3、图4、图7、图8和图9的实施例的存储器访问属性的表。

[0126] 参考图11,该表结合了先前在图5中呈现的关于图2、图3和图4中呈现的实施例的表的发现。因此,可以参考图5的上述描述来描述关于图2、图3和图4的发现。

[0127] 此外,图11中呈现的表呈现了顺序双散列方法(例如,图7)、并发双散列方法以及扩展重定向表双散列方法(例如,图9)的属性。

[0128] 考虑到顺序双散列方法(例如,图7),为了获得加密密钥,执行第一双散列并且访问DRAM.1。有50%的冲突机会。如果存在冲突,则执行第二双散列,并访问DRAM.2。相对于DRAM.1,DRAM.2的平均访问时间为.5x。因此,执行两个双散列函数的平均存储器访问时间为1.5x(1x+.5x)。

[0129] 顺序双散列方法的条目大小可以是65,536(64B+3B)。

[0130] 考虑到并发双散列方法(例如,图8),为了获得加密密钥,第一双散列与至少一个其他双散列同时执行。因此,如果对源索引同时执行两个双散列函数,则访问DRAM.1和DRAM.2。DRAM.1的平均访问时间为1x,DRAM.2的近似平均访问时间为～0x。因此,执行两个双散列函数的平均存储器访问时间为～1x。虽然平均存储器访问时间似乎很短,但很难预测从存储器中获得加密或解密密钥所需的并发双散列的数量。因此,可能需要执行大量并发双散列以确保访问加密密钥,这是不希望的。

[0131] 并发双散列方法的条目大小可以是(4+1)x65,536(64B+3B)。在图8所示的实施例中,在H1处发生冲突的总体似然性为.1(10%),假设单个散列的冲突可能性为.5(50%)。在H1处发生冲突的似然性是基于五个散列函数H1-H5来确定的,这五个散列函数同时分割了它们之间发生冲突的可能性(通过将单个散列的冲突的可能性(.5)除以散列函数的数量(5)来确定.1总体冲突可能性)。因此,在图8所示的实施例中,在H1处发生冲突的总体似然

性为.1(.1＝0.5/5)。

[0132] 考虑到扩展重定向表双散列方法(例如,图9),为了获得加密密钥,访问SRAM。当访问SRAM时,执行第一双散列函数。与DRAM相比,SRAM的访问速度非常快。例如,SRAM访问时间可能需要2ns-5ns,DRAM可能需要50ns-70ns。假设标准化SRAM值为DRAM的.1x。然而,更快的SRAM访问时间是可能的。当访问DRAM.1时,执行第二双散列函数。因此,执行两个双散列函数的平均存储器访问时间为1.1x。此外,扩展重定向表具有降低负载因子的效果,因此降低了冲突的可能性,从而提高了识别与加密密钥的匹配的可能性。在图9的情况下,重定向表进一步将H1处发生冲突的总体似然性降低了4倍。因此,假设单个散列的冲突的可能性为.5(50%),则图9在H1处的总体冲突可能性为0.025或2.5%(0.025＝(0.5/5)/4),因为五个散列函数H1-H5同时分割了单个散列的冲突的可能性(0.5/5),并且重定向表进一步将在H1处发生冲突的总体似然性降低了4倍(/4)。

[0133] 扩展重定向表双散列方法的条目大小可以是65,536(64B+3B)(对于存储部分(例如,辅存储器))+4x65,536(2B)(对于扩展重定向部分(例如,主存储器))。

[0134] 对于对应于图7-图9的每个方法,添加和删除操作分别包括开放散列索引和使条目无效。开放散列索引和使条目无效(或使重定向条目无效,如扩展重定向表双散列方法的情况)的添加和删除操作比添加到TOL并删除和重新链接的添加和移除操作复杂得多,与对应于图2-图4的链式散列方法相对应。

[0135] 对于对应于图7-图9的每个方法,重新排序MFU条目需要交换条目关联,这比将MFU复制到HOL(可以在链接列表方法中使用)更复杂,因为交换条目关联需要移动存储在交换的每个地址中的实际数据(例如,将512b加密密钥从一个地址移动到另一地址)。

[0136] 图12是示出根据实施例的在不等待完成第一查找请求的情况下执行第二查找请求的时序图。时序图中提出的解决方案可以由控制器实现,并且因为可以同时执行多个查找请求,所以改进了访问DRAM的总体延迟。

[0137] 参考图12,示出了125ns周期,并且每125ns输入CMD。125ns周期值是示例性的,可以使用其他值。为了使用链接列表执行查找请求,在1201对源密钥进行散列,并在1202访问重定向表。命令1201和1202可以在静态存储器中执行,因此,延迟小。

[0138] 接下来,在1203访问HOL DRAM。与SRAM相比,访问DRAM所需的时间较长,因此在1203访问HOL DRAM时引起SoC转发延迟。在1204,检查HOL密钥,并且在1205,输出HOL的冲突确定。当输出HOL时,结果可能是丢失(冲突)或匹配。输出HOL,因此引起SoC响应等待。

[0139] 当1205处的HOL输出是匹配时,则不需要进一步的命令来识别加密密钥。如果1205处的HOL输出是丢失(冲突),则在引起SoC转发延迟延时之后,可以在1206处访问DRAM的第二存储地址。在1207检查第二存储地址的密钥,并且在1208输出冲突确定。如图12所示,1208处的输出是匹配,因此不需要进一步的命令来识别加密密钥。

[0140] 值得注意的是,在1209处对第二源索引进行散列以开始识别第二加密密钥之前,图12的信令配置不等待匹配被识别(例如,在1208处)。换句话说,图12的信令配置在1209指示对第二源索引进行散列,即使1205处的输出是丢失并导致冲突。因此,信令配置能够同时获得两个或更多个加密密钥。

[0141] 图12示出了由于在1201处执行单独散列而执行顺序散列的情况。然而,信令配置也适用于并发散列,并且可以在1201处同时执行两个或更多个散列。如果同时执行两个或

更多个散列，则SoC转发延迟时间和SoC响应延迟时间将比在1201处执行单独散列的情况下更多。此外，尽管图12示出了为访问存储在DRAM中的地址而执行的查找请求，但是可以使用其他存储器类型。例如，可以执行本申请中描述的查找请求以访问存储在SRAM、闪存、远程服务器、片上存储器设备或附加存储器类型上的信息。

[0142] 图13是示出根据实施例的用于执行本申请中描述的实施例的结构组件的存储器系统。

[0143] 图1-图12中描述的操作可以由存储在存储器中的控制器、处理器或计算机实现的指令执行。例如，可以使用FPGA、ASIC、通用计算机或远程处理系统(例如，云计算系统)来实现由控制器执行的指令。

[0144] 参考图13，示出了能够执行本申请中描述的实施例的存储器系统1300。存储器系统1300包括主机1301、IC 1302(例如，存储器缓冲芯片)和DRAM(例如，非易失性存储器)1303。虽然DRAM被示出为与IC 1302分离，但是DRAM可以被包括在IC 1302上。

[0145] IC 1302包括主机接口1304、SRAM(例如，易失性存储器)1305、存储介质(例如，闪存)1306和控制器1307。主机接口1304可以将信息从主机1301通信传送到IC 1302或从IC 1302通信传送到主机1301。与DRAM 1303相比，SRAM 1305的大小可以相对较小。然而，SRAM 1305可以具有比DRAM 1303更快的读取/写入速度。此外，存储介质1306可以存储要发送到主机1301和/或DRAM 1303或从主机1301和/或DRAM 1303发送的数据。

[0146] 本申请中描述的实施例提供了特定配置，以改进对存储信息的访问，特别是对KPIO系统的存储信息的访问。图13所示的存储系统提供了用于实现本申请的实施例的结构。然而，应注意，本申请的实施例不应限于图13的结构，因为本领域普通技术人员将认识到，可以应用其他存储器存储系统来实现本申请的实施例。

[0147] 图14是示出根据实施例的访问存储条目的流程图。

[0148] 图14中描述的步骤可以由存储在存储器中的控制器、处理器或计算机实现的指令执行。

[0149] 参考图14，在步骤1401，对源索引进行散列以生成散列索引。散列索引可以是预定数量的比特(例如，16比特或18比特)，并且可以是包括NSID和密钥标签的源索引的散列。

[0150] 在步骤1402中，访问对应于散列索引的第一存储地址。第一存储地址可以被包括在主存储器或辅存储器中。主存储器可以是SRAM或DRAM。辅存储器可能是DRAM(而不是SRAM)。

[0151] 在步骤1403中，将对应于第一存储地址的标签部分与源索引进行比较以识别是否存在冲突。标签部分可以是与源索引相同数量的比特(例如，两者都可以是24比特)。

[0152] 在步骤1404中，响应于在步骤1403中识别出不存在冲突，从第一存储地址获得信息。获得的信息可以是预定大小(例如，512比特)的加密或解密密钥

[0153] 如果确实存在冲突，则可能出现多个不同的选项(例如，折叠链接列表、辅存储器链接列表、具有链接列表的重新路由表、具有链接列表的MFU、顺序双散列冲突解决、并发双散列冲突解决、具有双散列冲突解决的HOL重新路由表等)。

[0154] 图15示出了根据实施例的网络环境中的电子设备。

[0155] 参考图15，网络环境1500中的电子设备1501(例如，包括GPS功能的移动终端)可以经由第一网络1598(例如，短距离无线通信网络)与电子设备1502通信，或者经由第二网络

1599(例如,长距离无线通信网)与电子设备1504或服务器1508通信。电子设备1501可以经由服务器1508与电子设备1504通信。电子设备1501可以包括处理器1520、存储器1530、输入设备1550、声音输出设备1555、显示设备1560、音频模块1570、传感器模块1576、接口1577、触觉模块1579、相机模块1580、电力管理模块1588、电池1589、通信模块1590、订户识别模块(SIM)1596或天线模块1597(包括GNSS天线)。在一个实施例中,可以从电子设备1501中省略至少一个组件(例如,显示设备1560或相机模块1580),或者可以将一个或多个其他组件添加到电子设备1501。在一个实施例中,一些组件可以实现为单个IC。例如,传感器模块1576(例如,指纹传感器、虹膜传感器或照度传感器)可嵌入显示设备1560(例如,显示器)中。

[0156] 处理器1520可以执行软件(例如,程序1540)以控制与处理器1520耦合的电子设备1501的至少一个其他组件(例如,硬件或软件组件),并且可以执行各种数据处理或计算。作为数据处理或计算的至少一部分,处理器1520可以将从另一组件(例如,传感器模块1576或通信模块1590)接收的命令或数据加载到易失性存储器1532中,处理存储在易失性存储器1532中的命令或数据,并将结果数据存储在非易失性存储器1534中。处理器1520可以包括主处理器1521(例如,中央处理单元(CPU)或应用处理器(AP))和辅处理器1523(例如,图形处理单元(GPU)、图像信号处理器(ISP)、传感器集线器处理器或通信处理器(CP)),其可独立于主处理器1521或与主处理器1521结合操作。附加地或可选地,辅处理器1523可以适于消耗比主处理器1521更少的功率,或者执行特定功能。辅处理器1523可以被实现为与主处理器1521分离或是其一部分。

[0157] 当主处理器1521处于非活动(例如,睡眠)状态时,辅处理器1523可以控制与电子设备1501的组件中的至少一个组件(例如,显示设备1560、传感器模块1576或通信模块1590)相关的功能或状态中的至少一些,或者在主处理器1521处于活动状态(例如执行应用)时与主处理器1522一起执行。根据一个实施例,辅处理器1523(例如,图像信号处理器或通信处理器)可以实现为在功能上与辅处理器1523相关的另一组件(例如,相机模块1580或通信模块1590)的一部分。

[0158] 存储器1530可以存储由电子设备1501的至少一个组件(例如,处理器1520或传感器模块1576)使用的各种数据。各种数据可以包括例如软件(例如,程序1540)以及与之相关的命令的输入数据或输出数据。存储器1530可以包括易失性存储器1532或非易失性存储器1534。

[0159] 程序1540可以作为软件存储在存储器1530中,并且可以包括例如操作系统(OS)1542、中间件1544或应用1546。

[0160] 输入设备1550可以从电子设备1501的外部(例如,用户)接收将由电子设备1502的另一组件(例如,处理器1520)使用的命令或数据。输入设备1550可以包括,例如,麦克风、鼠标或键盘。

[0161] 声音输出设备1555可以向电子设备1501的外部输出声音信号。声音输出设备1555可以包括例如扬声器或接收器。扬声器可用于一般用途,诸如播放多媒体或录音,接收器可用于接收来电。接收器可以被实现为与扬声器分离或是扬声器的一部分。

[0162] 显示设备1560可以向电子设备1501的外部(例如,用户)可视地提供信息。显示设备1560可以包括例如显示器、全息设备或投影仪以及控制电路,以控制显示器、全息设备和投影仪中的对应一个。根据一个实施例,显示设备1560可以包括适于检测触摸的触摸电路,

或者适于测量触摸引起的力的强度的传感器电路(例如,压力传感器)。

[0163] 音频模块1570可以将声音转换成电信号,反之亦然。根据一个实施例,音频模块1570可以经由输入设备1550获得声音,或者经由声音输出设备1555或与电子设备1501直接(例如,有线)或无线耦合的外部电子设备1502的耳机输出声音。

[0164] 传感器模块1576可以检测电子设备1501的操作状态(例如,功率或温度)或电子设备1502外部的环境状态(例如用户的状态),然后生成与检测到的状态相对应的电信号或数据值。传感器模块1576可以包括例如手势传感器、陀螺仪传感器、大气压力传感器、磁性传感器、加速度传感器、抓握传感器、接近传感器、颜色传感器、红外(IR)传感器、生物测定传感器、温度传感器、湿度传感器或照度传感器。

[0165] 接口1577可以支持电子设备1501直接(例如,有线)或无线地与外部电子设备1502耦合的一个或多个指定协议。根据一个实施例,接口1577可以包括例如高清晰度多媒体接口(HDMI)、通用串行总线(USB)接口、安全数字(SD)卡接口或音频接口。

[0166] 连接端子1578可以包括电子设备1501可以经由其与外部电子设备1502物理连接的连接器。根据一个实施例,连接端子1578可以包括例如HDMI连接器、USB连接器、SD卡连接器或音频连接器(例如,耳机连接器)。

[0167] 触觉模块1579可将电信号转换为机械刺激(例如,振动或运动)或电刺激,其可由用户经由触感或动觉来识别。根据一个实施例,触觉模块1579可以包括例如马达、压电元件或电刺激器。

[0168] 相机模块1580可以捕获静止图像或运动图像。根据一个实施例,相机模块1580可以包括一个或多个镜头、图像传感器、图像信号处理器或闪光灯。

[0169] 电力管理模块1588可以管理供应给电子设备1501的电力。电力管理模块158可以被实现为例如电力管理集成电路(PMIC)的至少一部分。

[0170] 电池1589可向电子设备1501的至少一个组件供电。根据一个实施例,电池1589可以包括例如不可再充电的原电池、可再充电电池或燃料电池。

[0171] 通信模块1590可以支持在电子设备1501和外部电子设备(例如,电子设备1502、电子设备1504或服务器1508)之间建立直接(例如,有线)通信信道或无线通信信道,并通过所建立的通信信道执行通信。通信模块1590可以包括一个或多个通信处理器,其可独立于处理器1520(例如,应用处理器)操作并且支持直接(例如,有线)通信或无线通信。根据一个实施例,通信模块1590可以包括无线通信模块1592(例如,蜂窝通信模块、短程无线通信模块或全球导航卫星系统(GNSS)通信模块)或有线通信模块1594(例如,局域网(LAN)通信模块或电力线通信(PLC)模块)。这些通信模块中的对应一个可以经由第一网络1598(例如,短距离通信网络,诸如蓝牙TM、无线保真(Wi-Fi)直连或者红外数据协会(IrDA))或第二网络1599(例如,长距离通信网络,诸如蜂窝网络、互联网或计算机网络(例如LAN或广域网(WAN))的标准与外部电子设备进行通信。这些各种类型的通信模块可以实现为单个组件(例如,单个IC),或者可以实现为彼此分离的多个组件(例如,多个IC)。无线通信模块1592可以使用存储在订户识别模块1596中的订户信息(例如,国际移动订户身份(IMSI))来识别和认证通信网络(诸如第一网络1598或第二网络1599)中的电子设备1501。

[0172] 天线模块1597可向电子设备1501的外部(例如,外部电子设备)发送或接收信号或电力。根据一个实施例,天线模块1597可包括一个或多个天线,并且可从中选择适合于通信

网络(诸如第一网络1598或第二网络1599)中使用的通信方案的至少一个天线,例如,通过通信模块1590(例如,无线通信模块1592)。然后,可以经由所选择的至少一个天线在通信模块1590和外部电子设备之间发送或接收信号或电力。

[0173] 上述组件中的至少一些可以相互耦合,并且经由外围设备间通信方案(例如,总线、通用输入和输出(GPIO)、串行外围设备接口(SPI)或移动工业处理器接口(MIPI))在其间之间传送信号(例如,命令或数据)。

[0174] 根据一个实施例,可通过与第二网络1599耦合的服务器1508在电子设备1501和外部电子设备1504之间发送或接收命令或数据。电子设备1502和1504中的每一个可以是与电子设备1501相同类型或不同类型的设备。将要在电子设备1500处执行的全部或部分操作可以在外部电子设备1503、1504或服务器1508中的一个或多个处执行。例如,如果电子设备1501应自动或者响应于来自用户或另一设备的请求执行功能或服务,代替执行该功能或服务或者除了执行该功能或服务之外,电子设备1501可以请求一个或多个外部电子设备执行功能或服务的至少一部分。接收请求的一个或多个外部电子设备可执行所请求的功能或服务的至少一部分,或与请求相关的附加功能或附加服务,并将执行的结果传送给电子设备1501。电子设备1501可在对结果进行或不进行进一步处理的情况下提供结果,作为对请求的答复的至少一部分。为此,例如,可以使用云计算、分布式计算或客户端-服务器计算技术。

[0175] 一个实施例可以被实现为软件(例如,程序1540),该软件包括存储在机器(例如,电子设备1501)可读的存储介质(例如,内部存储器1536或外部存储器1538)中的一个或多个指令。例如,电子设备1501的处理器可以调用存储在存储介质中的一个或多个指令中的至少一个,并在处理器的控制下使用或不使用一个或多个其他组件来执行该指令。因此,可以操作机器以根据所调用的至少一个指令执行至少一个功能。一个或多个指令可以包括由编译器生成的代码或由解释器执行的代码。可以以非暂时性存储介质的形式提供机器可读存储介质。术语“非暂时性”表示存储介质是有形设备,不包括信号(例如,电磁波),但该术语不区分数据半永久存储在存储介质中的位置和数据临时存储在存储媒体中的位置。

[0176] 根据一个实施例,可以在计算机程序产品中包括并提供本公开的方法。计算机程序产品可以作为产品在卖方和买方之间进行交易。计算机程序产品可以以机器可读存储介质(例如,光盘只读存储器(CD-ROM))的形式分发,或者通过应用商店(例如,Play Store™)在线分发(例如,下载或上传),或者直接在两个用户设备(例如,智能电话)之间分发。如果在线分发,则计算机程序产品的至少一部分可以临时生成或至少临时存储在机器可读存储介质中,诸如制造商服务器的存储器、应用商店的服务器或中继服务器。

[0177] 根据一个实施例,上述组件的每个组件(例如,模块或程序)可以包括单个实体或多个实体。可以省略一个或多个上述组件,或者可以添加一个或多个其他组件。可选地或附加地,多个组件(例如,模块或程序)可以集成到单个组件中。在这种情况下,集成组件仍然可以以与在集成之前由多个组件中的对应组件执行的功能相同或相似的方式执行多个组件的每个组件的一个或多个功能。由模块、程序或另一组件执行的操作可以顺序地、并行地、重复地或启发式地执行,或者可以以不同的顺序执行或省略一个或多个操作,或者可以添加一个或多个其他操作。

[0178] 尽管已经在本公开的详细描述中描述了本公开的某些实施例,但是可以在不脱离

本公开的范围的情况下以各种形式修改本公开。因此，本公开的范围不应仅基于所描述的实施例来确定，而是基于所附权利要求及其等同物来确定。

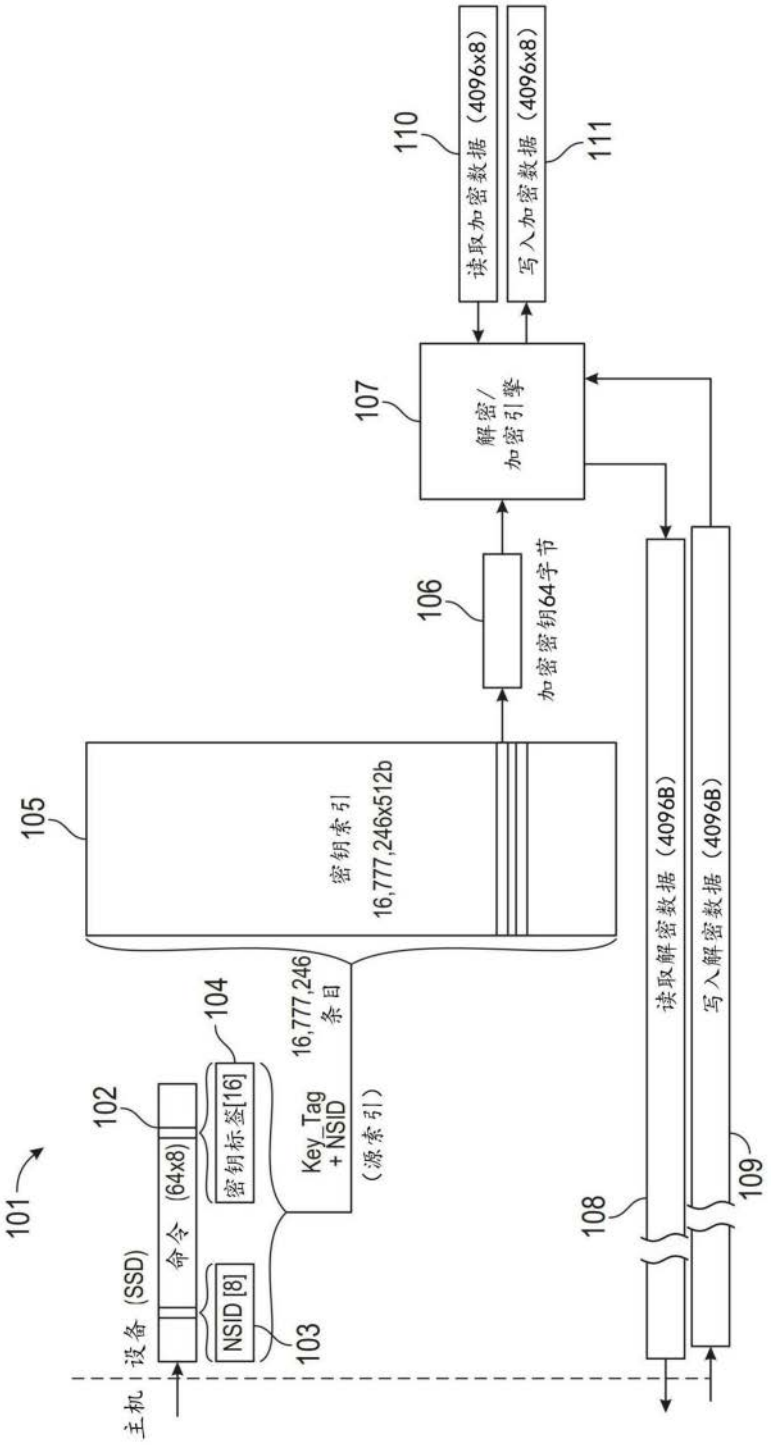

101
主机
设备 (SSD)
命令 (64x8)
NSID [8]
密钥标签[16]
Key_Tag + NSID (源索引)
16,777,246 条目
密钥索引 16,777,246x512b
加密密钥64字节
解密/加密引擎
读取加密数据 (4096x8)
写入加密数据 (4096x8)
读取解密数据 (4096B)
写入解密数据 (4096B)
102
103
104
105
106
107
108
109
110
111

图1

使用链接列表的示例散列冲突解决
将单独的链式链接列表与散列表合并
节省第1存储器中的未使用槽的存储器空间

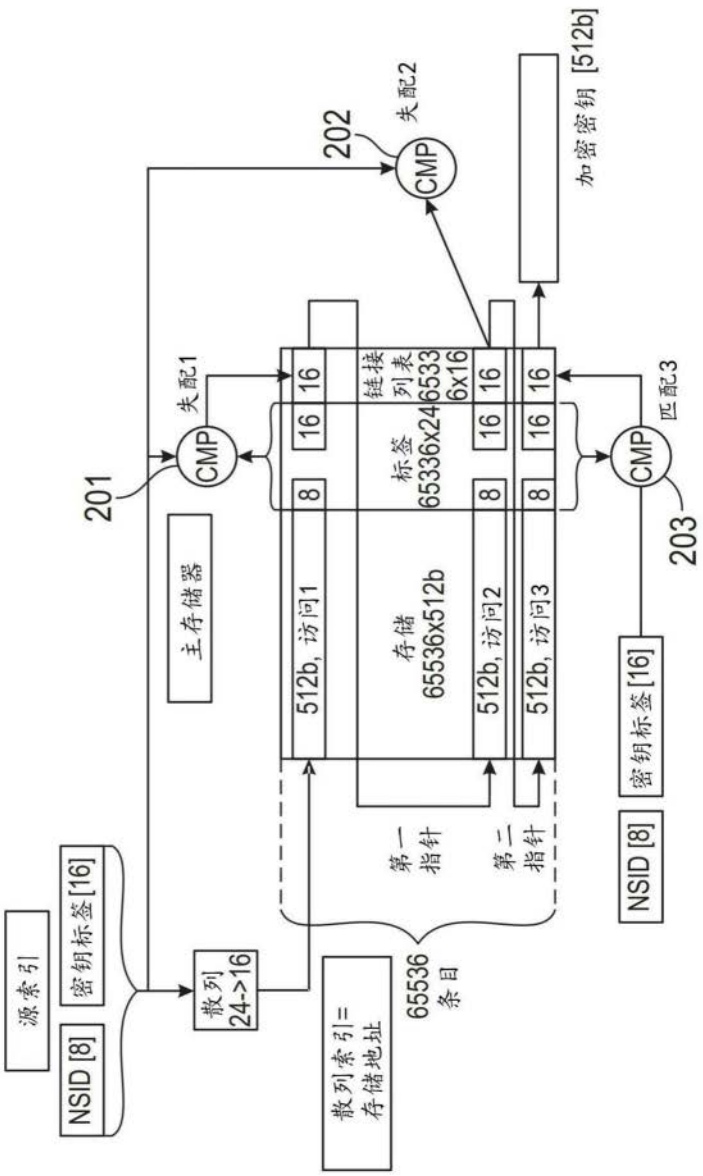


图2

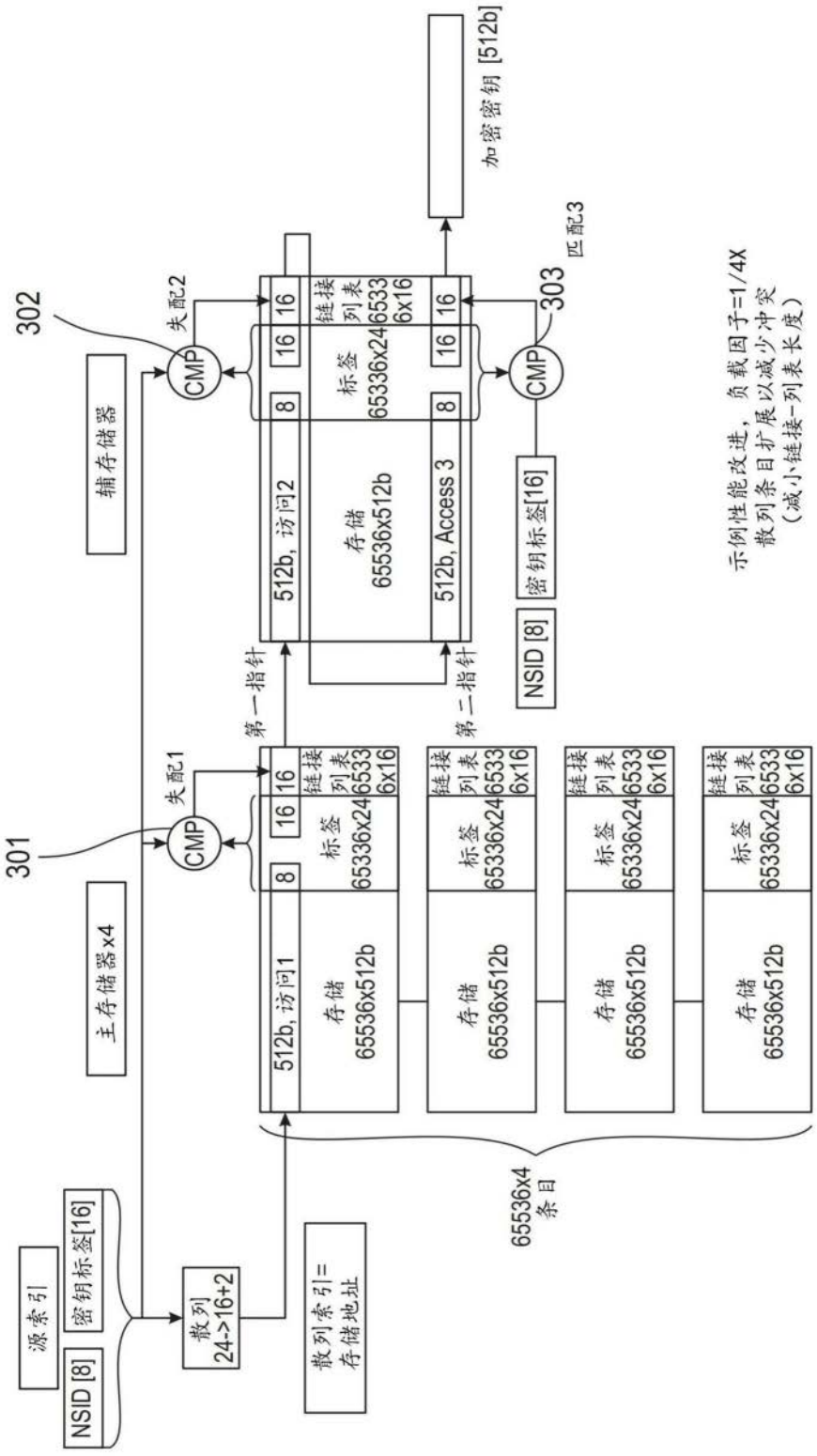

源索引
NSID [8]
密钥标签[16]
散列
24->16+2
散列索引=
存储地址
主存储器x4
辅存储器
301
302
303
失配1
失配2
匹配3
CMP
第一指针
第二指针
512b, 访问1
512b, 访问2
512b, Access 3
存储
65536x512b
8
16
标签
65336x24
链接
列表
6533
6x16
密钥标签[16]
NSID [8]
加密密钥 [512b]
65536x4
条目
示例性能改进，负载因子=1/4X
散列条目扩展以减少冲突
(减小链接-列表长度)

图3

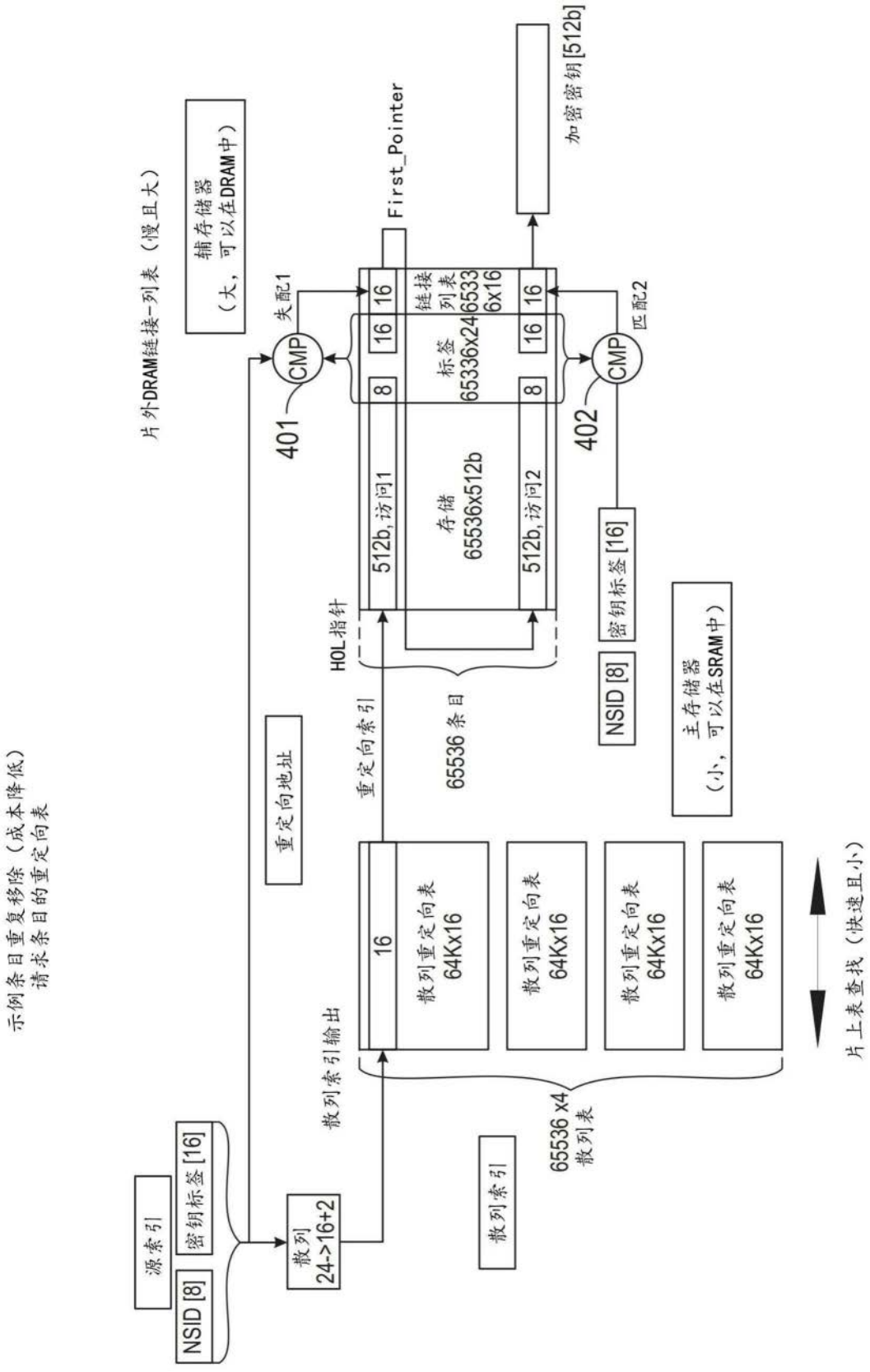

示例条目重复移除（成本降低）
请求条目的重定向表
片外DRAM链接-列表（慢且大）
辅存储器
（大，可以在DRAM中）
源索引
NSID [8]
密钥标签 [16]
散列
24->16+2
散列索引输出
重定向地址
失配1
CMP
401
First_Pointer
加密密钥[512b]
链接
列表
6533
6x16
标签
65336x24
16
8
512b,访问1
存储
65536x512b
512b,访问2
402
匹配2
HOL指针
重定向索引
65536 条目
NSID [8]
密钥标签 [16]
主存储器
（小，可以在SRAM中）
散列索引
65536 x4
散列表
散列重定向表
64Kx16
片上表查找（快速且小）

图4

添加/删除/重新排序/遍历 #

#	方法	条目 #	添加	删除HOL	重新排序MFU	标准化为DRAM.1的存储器访问查找次数		
						SRAM	DRAM.1	DRAM.2
图2	LL散列+合并RAM	65,536	添加到TOL	删除并重新链接	将MFU复制到HOL LL改变		1	0.5
图3	LL散列+扩展	(4+1)x 65,536	添加到TOL	删除并重新链接	将MFU复制到HOL LL改变		1	0.125
图4	LL散列+扩展+重定向	65,536	添加到TOL	删除并重新链接	LL改变	.04	1	0.125
		4x 65,536						

注释1:假设Load_Factor=1x处50%的散列冲突
注释2:假设Load_Factor=1/4x处12.5%的散列冲突
注释3:标准化SRAM访问到DRAM访问：1/25x-1/14x
- SRAM:2ns-5ns vs DRAM 50ns-70ns

图5

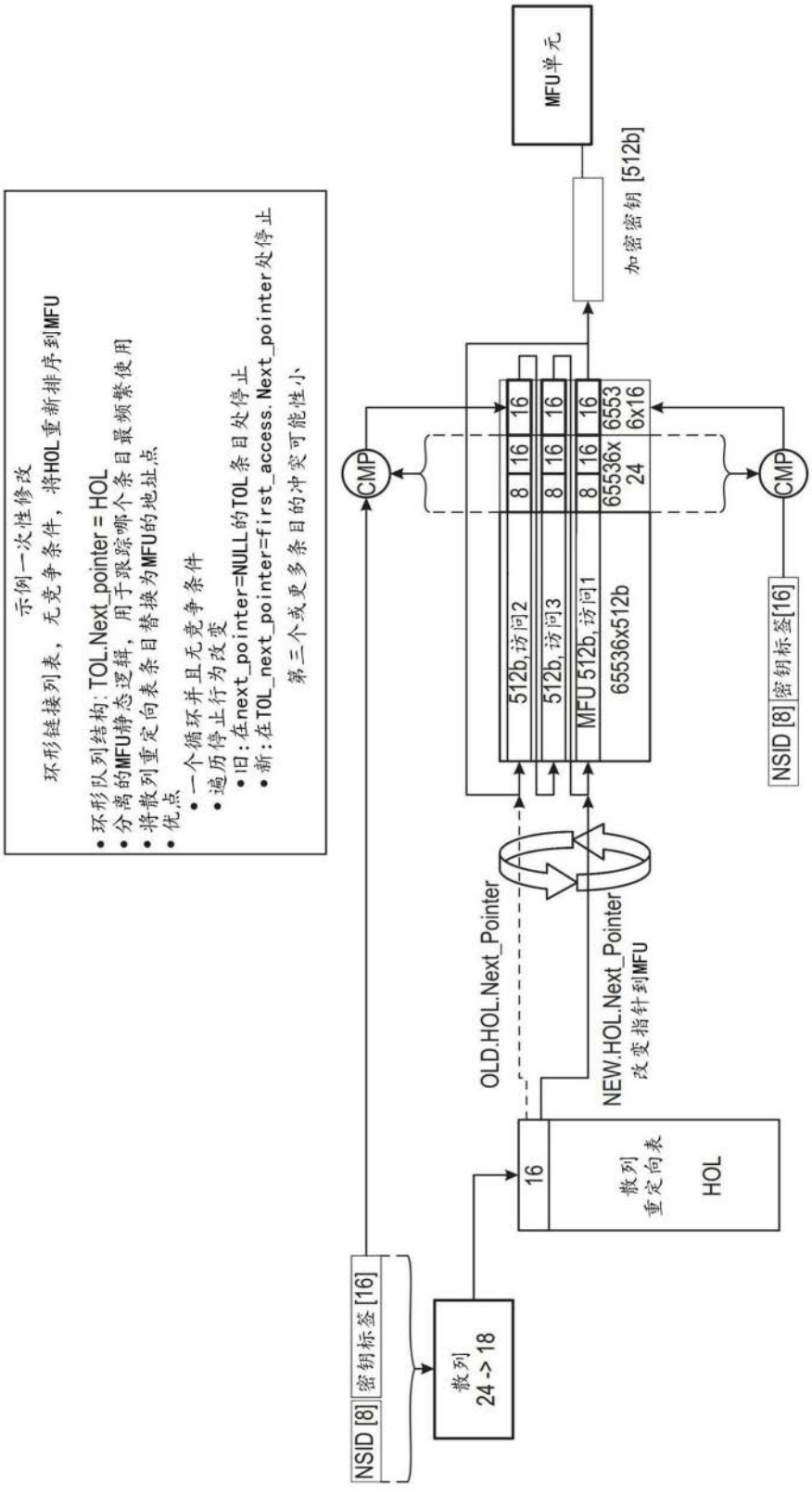

示例一次性修改
环形链接列表，无竞争条件，将HOL重新排序到MFU
• 环形队列结构：TOL.Next_pointer = HOL
• 分离的MFU静态逻辑，用于跟踪哪个条目最频繁使用
• 将散列重定向表条目替换为MFU的地址点
• 优点
• 一个循环并且无竞争条件
• 遍历停止行为改变
• 旧：在next_pointer=NULL的TOL条目处停止
• 新：在TOL_next_pointer=first_access.Next_pointer处停止
第三个或更多条目的冲突可能性小
MFU单元
加密密钥 [512b]
CMP
CMP
8 16 16
512b,访问2
8 16 16
512b,访问3
8 16 16
MFU 512b,访问1
65536x 6553
24 6x16
65536x512b
NSID [8] 密钥标签[16]
OLD.HOL.Next_Pointer
NEW.HOL.Next_Pointer
改变指针到MFU
16
散列
重定向表
HOL
NSID [8] 密钥标签 [16]
散列
24 -> 18

图6

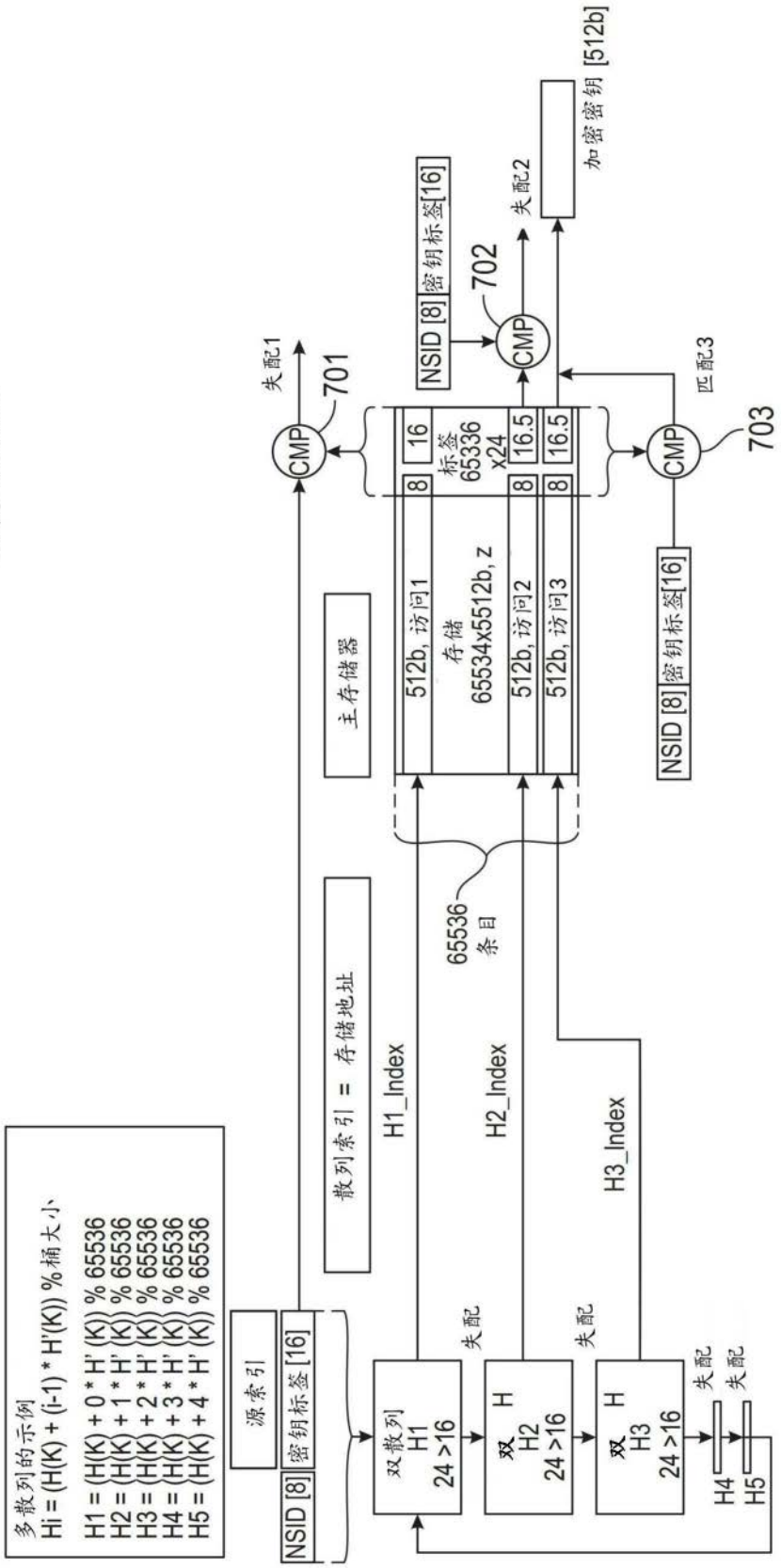

示例双+散列机制无链接列表
顺序多散列探测
多散列的示例
Hi = (H(K) + (i-1) * H'(K)) % 桶大小
H1 = (H(K) + 0 * H'(K)) % 65536
H2 = (H(K) + 1 * H'(K)) % 65536
H3 = (H(K) + 2 * H'(K)) % 65536
H4 = (H(K) + 3 * H'(K)) % 65536
H5 = (H(K) + 4 * H'(K)) % 65536
源索引
NSID [8]
密钥标签 [16]
散列索引 = 存储地址
主存储器
双散列 H1 24 >16
双 H H2 24 >16
双 H H3 24 >16
失配
失配
失配
失配
H4
H5
H1_Index
H2_Index
H3_Index
65536 条目
512b, 访问1
512b, 访问2
512b, 访问3
存储 65534x5512b, z
8
16
标签 65336 x24
16.5
16.5
8
8
CMP
701
失配1
NSID [8]
密钥标签[16]
702
CMP
失配2
加密密钥 [512b]
NSID [8]
密钥标签[16]
CMP
匹配3
703

图7

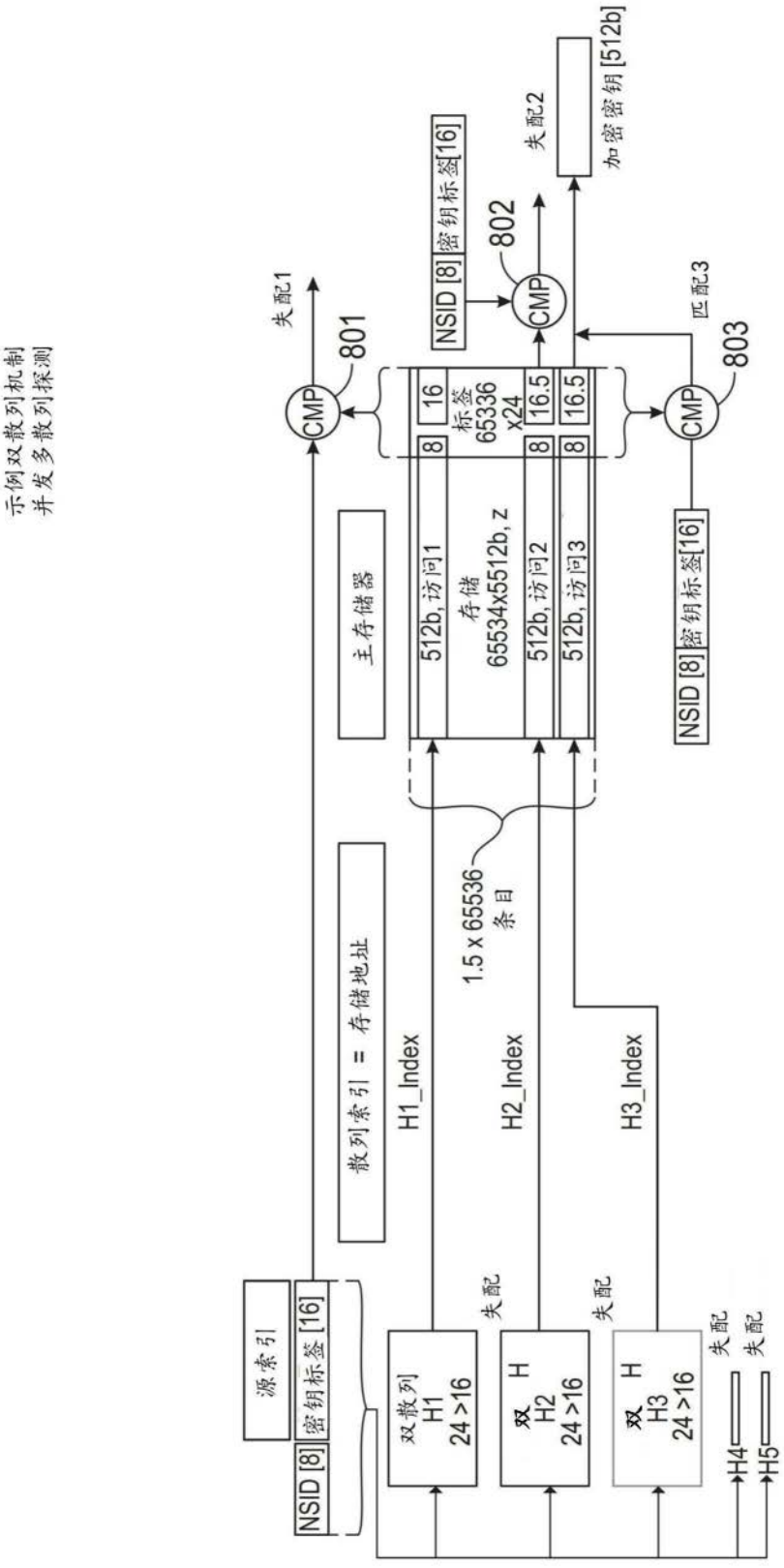

示例双散列机制
并发多散列探测
源索引
NSID [8]
密钥标签 [16]
双散列
H1
24 >16
双 H
H2
24 >16
双 H
H3
24 >16
失配
失配
失配
失配
H4
H5
散列索引 = 存储地址
H1_Index
H2_Index
H3_Index
1.5 x 65536
条目
主存储器
512b, 访问1
512b, 访问2
512b, 访问3
8
16
存储
65534x5512b, z
标签
65336
x24
16.5
16.5
8
8
CMP
801
失配1
NSID [8]
密钥标签[16]
802
失配2
加密密钥 [512b]
匹配3
803
NSID [8]
密钥标签[16]

图8

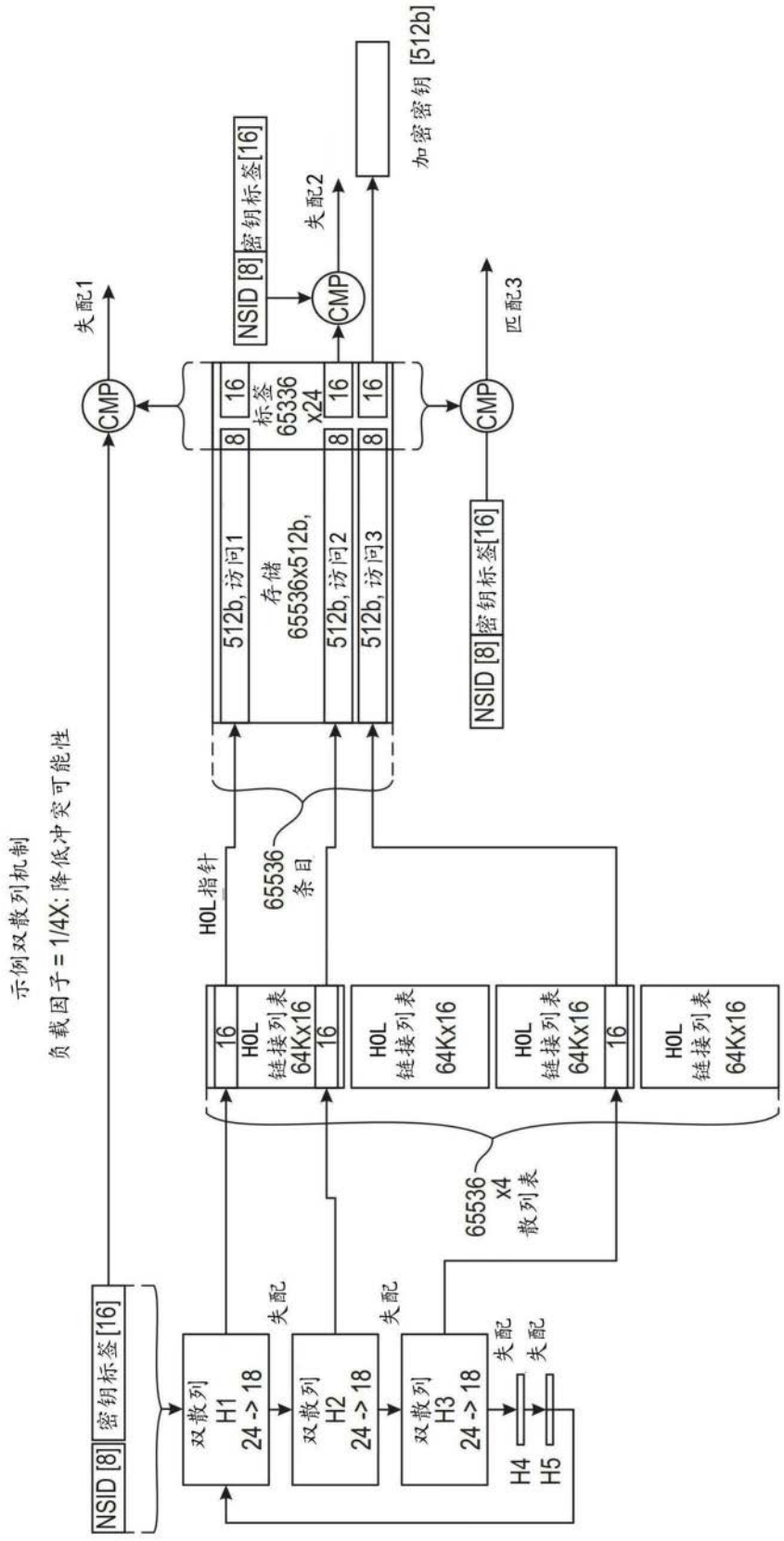

示例双散列机制
负载因子 = 1/4X: 降低冲突可能性
NSID [8]
密钥标签[16]
双散列 H1 24 -> 18
双散列 H2 24 -> 18
双散列 H3 24 -> 18
H4
H5
失配
失配
失配
失配
65536 x4 散列表
HOL 链接列表 64Kx16
16
HOL指针
65536 条目
512b, 访问1
512b, 访问2
512b, 访问3
8
存储 65536x512b,
标签 65336 x24
CMP
失配1
失配2
匹配3
NSID [8]
密钥标签[16]
加密密钥 [512b]

图9

#	发布多散列索引探测的5个选项	应用场景
图7	顺序发布H1、H2、H3…探测模式	DRAM控制器针对每个读取执行页丢失或行开始
图8	并发发布H1、H2、H3…探测模式	DRAM控制器针对3个读取背对背执行页命中访问
混合1	程序可选择顺序或并发探测模式	程序根据DRAMC选择#1或#2
混合2	程序自适应顺序和并发模式	按顺序开始。如果当前探测冲突，则下一个探测并发发布。 如果当前探测没有冲突，则下一个探测是顺序发布。
混合3	渐进动态自适应模式	按顺序开始。如果发现冲突，将下一个探测改变为并发发布，但递增探测数量 - 从0到1、从1到2或从2到3 - 在3处饱和（或最大并发计数） 如果发现没有冲突，改变下一个探测以减少并发发布或探测的顺序数量 - 从3到2、从2到1或从1到0（顺序发布）

图10

添加/删除/重新排序/遍历#

#	方法	条目#	添加	删除	重新排序MFU	表+第1+第2+遍历#存储器访问		
						SRAM	DRAM.1	DRAM.2
图2	LL散列+合并	65,536	添加到TOL	删除并重新链接	将MFU复制到HOL LL改变		1	0.5
图3	LL散列+扩展	4x 65,536	添加到TOL	删除并重新链接	将MFU复制到HOL LL改变		1	0.125
图4	LL散列+扩展+重定向	65,536	添加到TOL	删除并重新链接	LL改变	.04	1	0.125
		4x 65,536						
图7	双散列+顺序	65,536	开放散列索引	使条目无效	交换条目关联		1	0.5
图8	双散列+并行	65,536	开放散列索引	使条目无效	交换条目关联		1	~0
图9	双散列+扩展+重定向+并行	65,536				.04	1	~0
		4x 65,536	开放散列索引	使重定向条目无效	交换条目关联			

注释 1: 假设Load_Factor=1x处50%的散列冲突
注释 2: 假设Load_Factor=1/4x处12.5%的散列冲突
注释 3: 标准化SRAM访问到DRAM访问: 1/25x-1/14x
- SRAM:2ns-5ns vs DRAM 50ns-70ns

图11

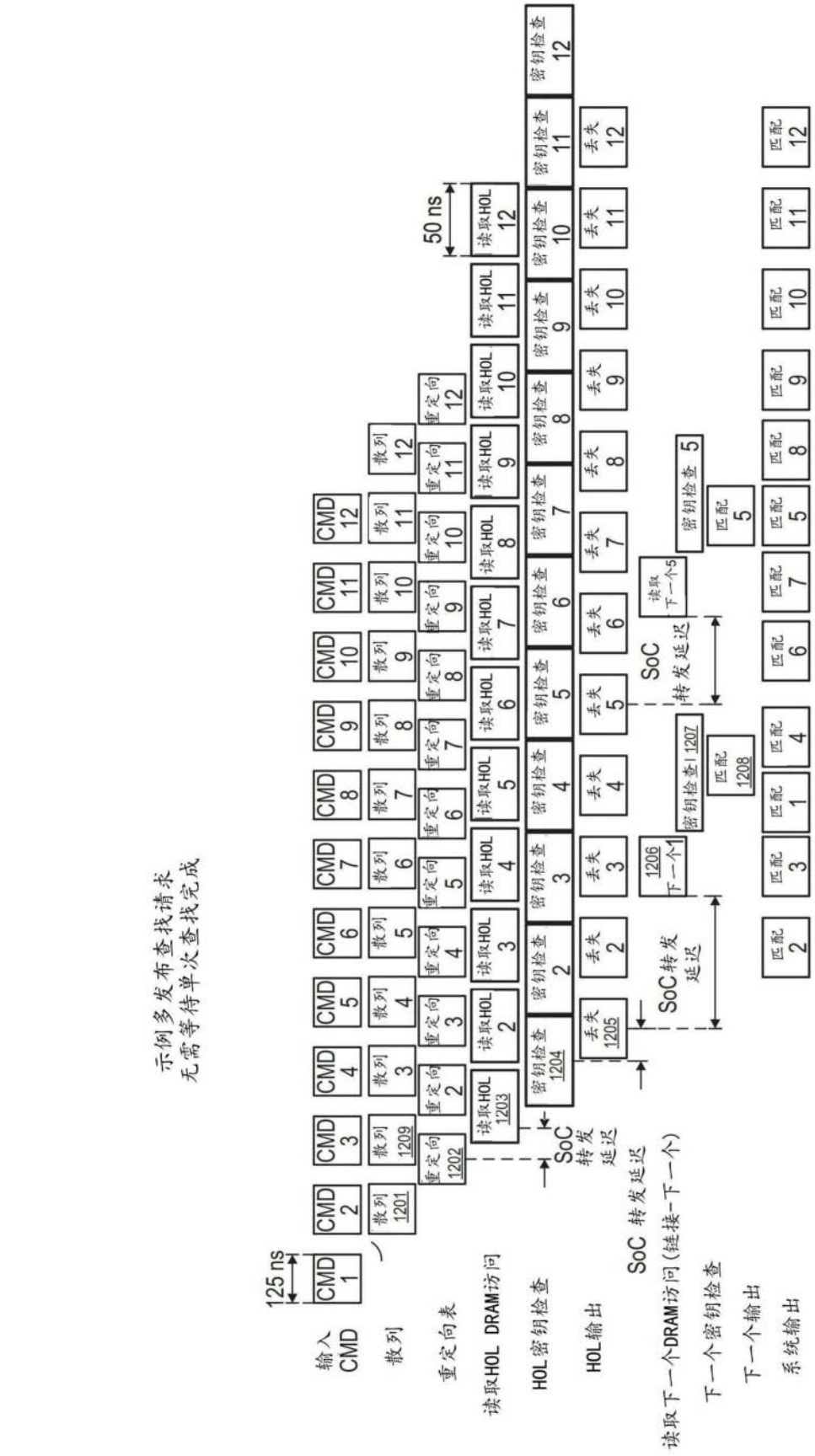

示例多发布查找请求
无需等待单次查找完成
125 ns
50 ns
输入CMD
散列
重定向表
读取HOL DRAM访问
HOL密钥检查
HOL输出
SoC 转发延迟
读取下一个DRAM访问(链接-下一个)
下一个密钥检查
下一个输出
系统输出
SoC转发延迟
SoC转发延迟
SoC转发延迟
1201
1202
1203
1204
1205
1206
1207
1208
1209

图12

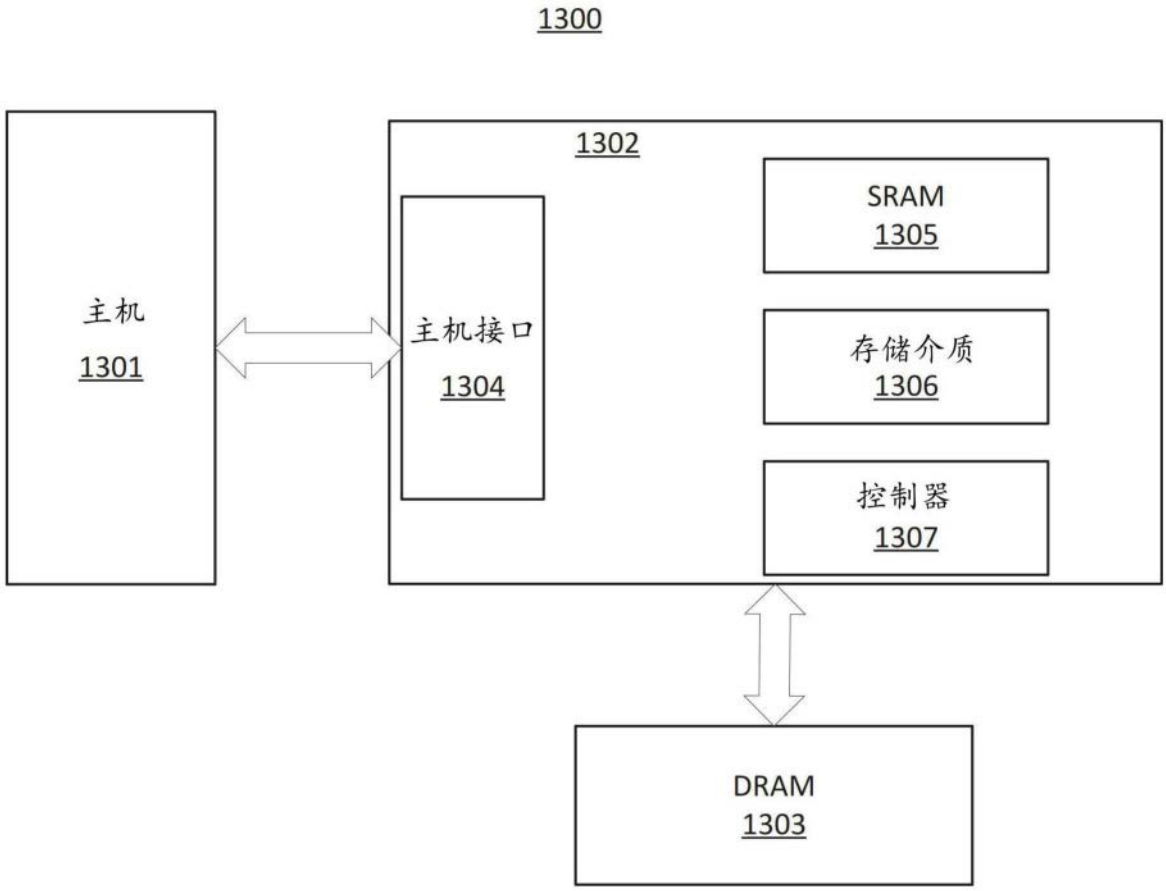

1300
主机
1301
1302
主机接口
1304
SRAM
1305
存储介质
1306
控制器
1307
DRAM
1303

图13

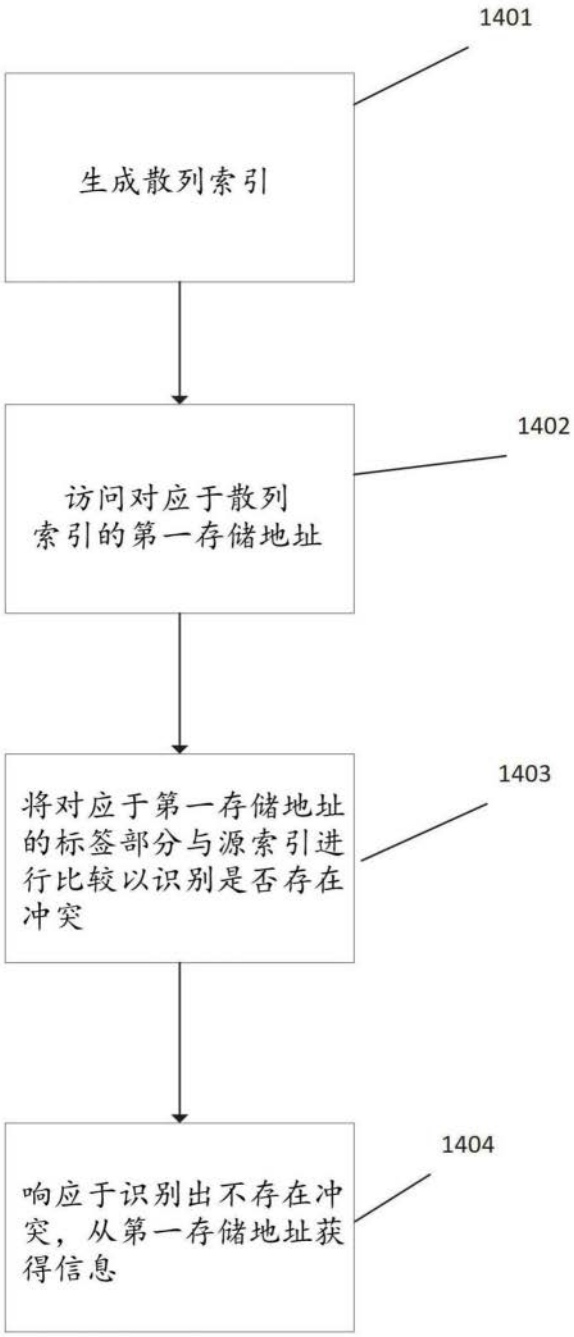

1401
生成散列索引
1402
访问对应于散列索引的第一存储地址
1403
将对应于第一存储地址的标签部分与源索引进行比较以识别是否存在冲突
1404
响应于识别出不存在冲突，从第一存储地址获得信息

图14

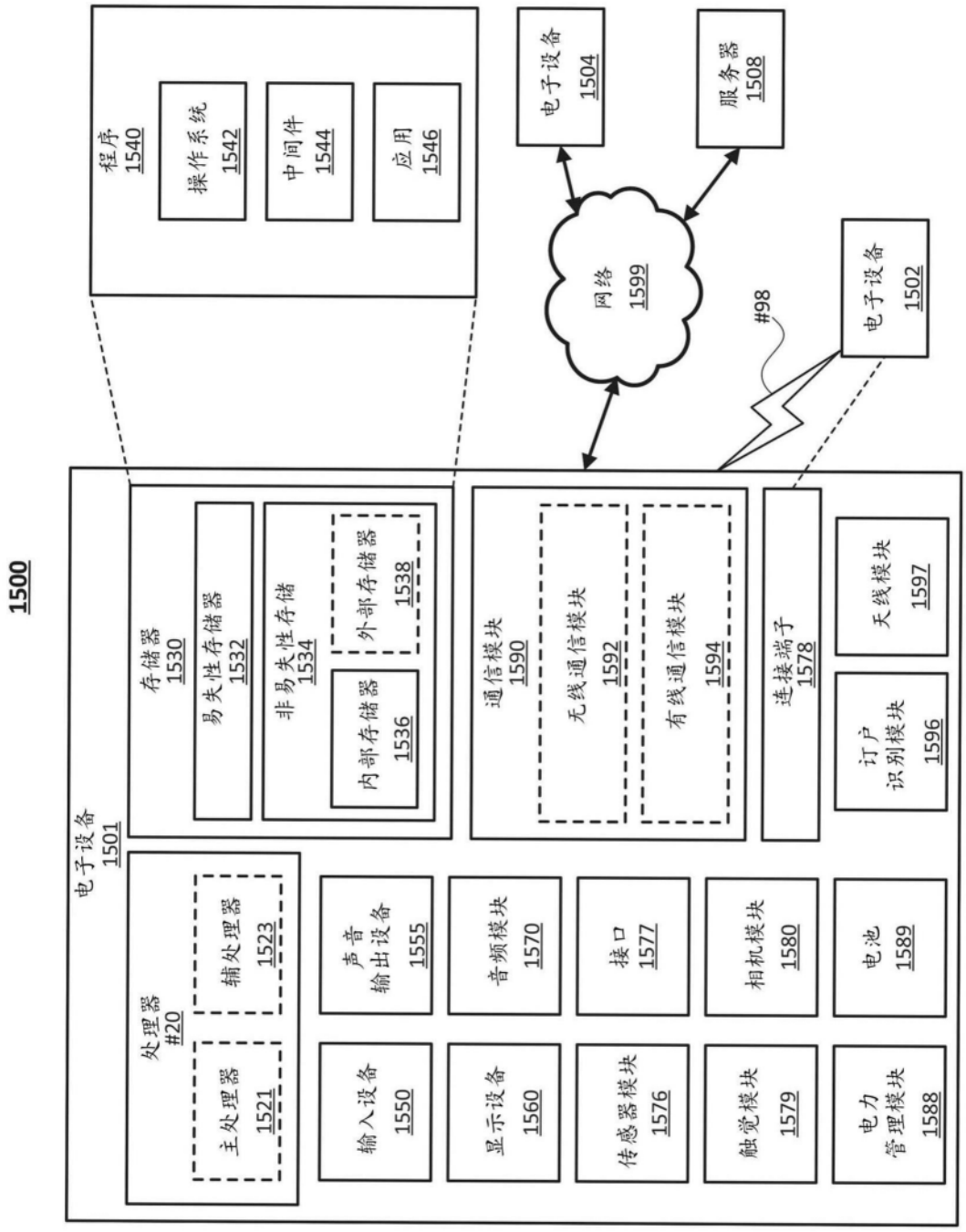

1500
电子设备 1501
处理器 #20
主处理器 1521
辅处理器 1523
存储器 1530
易失性存储器 1532
非易失性存储 1534
内部存储器 1536
外部存储器 1538
输入设备 1550
显示设备 1560
声音输出设备 1555
音频模块 1570
传感器模块 1576
接口 1577
触觉模块 1579
相机模块 1580
电力管理模块 1588
电池 1589
通信模块 1590
无线通信模块 1592
有线通信模块 1594
连接端子 1578
订户识别模块 1596
天线模块 1597
程序 1540
操作系统 1542
中间件 1544
应用 1546
网络 1599
电子设备 1504
服务器 1508
#98
电子设备 1502

图15