

(12) 发明专利申请

(10) 申请公布号 CN 102033715 A

(43) 申请公布日 2011.04.27

(21) 申请号 201010543977. X

(22) 申请日 2010.09.26

(30) 优先权数据

12/571382 2009.09.30 US

(71) 申请人 英特尔公司

地址 美国加利福尼亚州

(72) 发明人 G·库马 D·纳加拉

V·R·弗雷塔格 E·德拉诺

G·S·阿维里尔

(74) 专利代理机构 中国专利代理(香港)有限公

司 72001

代理人 叶晓勇 王洪斌

(51) Int. Cl.

G06F 3/06 (2006.01)

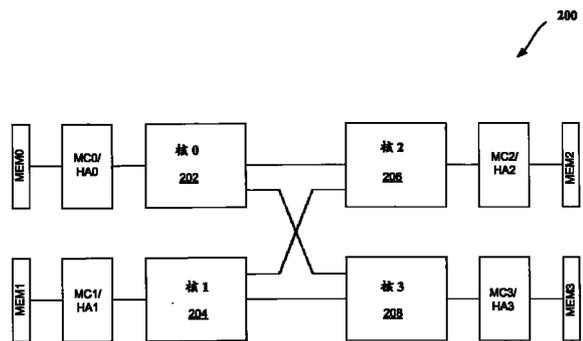
权利要求书 2 页 说明书 11 页 附图 7 页

(54) 发明名称

在本地代理的存储器镜像和迁移

(57) 摘要

描述了涉及在本地代理 (HA) 的存储器镜像和迁移的方法和设备。在一个实施例中,本地代理可以在从属代理镜像其数据。在一些实施例中,目录中的位可以指示高速缓存行的状态。还公开了其它实施例。



1. 一种设备,包括:
第一代理,将数据存储耦合至所述第一代理的第一存储器中;以及
第一逻辑,在第二代理对所述第一存储器中的所存储数据进行镜像,其中将以存储器控制器粒度复制所镜像数据。
2. 如权利要求1所述的设备,其中所述第一逻辑将基于显式写回、隐式写回或读故障恢复对所存储数据进行镜像。
3. 如权利要求1所述的设备,其中所述第二代理耦合至第二存储器以存储所镜像数据。
4. 如权利要求1所述的设备,还包括:包含所述第一存储器和第二存储器的存储器。
5. 如权利要求1所述的设备,其中所述第一代理和第二代理将维持目录,所述目录存储与各高速缓存行被高速缓存在哪个代理且以什么状态进行高速缓存有关的信息。
6. 如权利要求5所述的设备,其中对应于请求的一个或多个侦听仅被发送给被所述目录指示为包括与所述请求对应的数据的高速缓存副本的一个或多个本地代理。
7. 如权利要求5所述的设备,其中用于所述目录的条目的位将指示那个条目是否已被修改。
8. 如权利要求5所述的设备,其中用于所述目录的条目的位将指示发送了冲突响应的高速缓存代理是否涉及最近的冲突。
9. 如权利要求8所述的设备,其中所述最近的冲突将指示所述高速缓存代理具有冲突数据的最新副本。
10. 如权利要求1所述的设备,其中所述第一代理将包含所述第一逻辑。
11. 如权利要求1所述的设备,还包括:耦合至所述第一代理的目录高速缓存,存储与耦合至所述第一代理的多个高速缓存代理对应的数据。
12. 如权利要求11所述的设备,其中所述第一代理将响应于从所述多个高速缓存代理的一个或多个中接收的一个或多个侦听响应而更新所述目录高速缓存。
13. 如权利要求11所述的设备,其中所述第一代理将向被所述目录高速缓存识别为具有对应于目标地址的数据的副本的多个高速缓存代理中的一个或多个发送一个或多个侦听。
14. 如权利要求1所述的设备,其中所述第一代理是本地代理并且所述第二代理是从属代理。
15. 如权利要求1所述的设备,还包括:串行链路,耦合所述第一代理和第二代理。
16. 如权利要求1所述的设备,其中所述第一代理和第二代理在同一集成电路管芯上。
17. 如权利要求1所述的设备,其中所述第一逻辑将把所存储数据迁移到所述第二代理。
18. 一种方法,包括:
将数据存储耦合至第一代理的第一存储器中;以及
在第二代理对所述第一存储器中的所存储数据进行镜像,其中将以存储器控制器粒度复制所镜像数据。
19. 如权利要求18所述的方法,其中对所存储数据的镜像将基于显式写回、隐式写回或读故障恢复执行。

20. 如权利要求 18 所述的方法,还包括将来自所述第一存储器的数据存储在耦合至所述第二代理的第二存储器中。

21. 如权利要求 18 所述的方法,还包括维持目录,所述目录存储与各高速缓存行被高速缓存在哪个代理且以什么状态进行高速缓存有关的信息。

22. 如权利要求 21 所述的方法,还包括将对应于请求的一个或多个侦听仅发送给被所述目录指示为包括与所述请求对应的数据的高速缓存副本的一个或多个本地代理。

23. 如权利要求 21 所述的方法,其中用于所述目录的条目的位将指示那个条目是否已被修改。

24. 如权利要求 21 所述的方法,其中用于所述目录的条目的位将指示发送了冲突响应的高速缓存代理是否涉及最近的冲突。

25. 一种系统,包括:

存储目录的存储器,所述目录存储与每个高速缓存行被高速缓存在哪个代理且以什么状态进行高速缓存有关的信息;以及

将数据存储在所述存储器中的第一代理,其中所述第一代理将包括在第二代理对所述存储器中的所存储数据进行镜像的第一逻辑,

其中将以存储器控制器粒度复制所镜像数据。

26. 如权利要求 25 所述的系统,其中所述第一逻辑将基于显式写回、隐式写回或读故障恢复对所存储数据进行镜像。

27. 如权利要求 25 所述的系统,其中所述第二代理耦合至所述存储器以存储所镜像数据。

28. 如权利要求 25 所述的系统,其中对应于请求的一个或多个侦听仅被发送给被所述目录指示为包括与所述请求对应的数据的高速缓存副本的一个或多个本地代理。

29. 如权利要求 25 所述的系统,其中用于所述目录的条目的位将指示那个条目是否已被修改。

30. 如权利要求 25 所述的系统,其中用于所述目录的条目的位将指示发送了冲突响应的高速缓存代理是否涉及最近的冲突。

在本地代理的存储器镜像和迁移

技术领域

[0001] 本公开内容一般涉及电子学领域。更具体而言,本发明的实施例涉及在本地代理(HA :home agent)的存储器镜像(mirroring)和迁移(migration)。

背景技术

[0002] 计算机系统中的高速缓冲存储器可使用侦听(snoopy)总线或基于目录的协议来保持一致。在任一情况下,存储器地址与系统中的特定存储单元(location)相关联。这个存储单元(location)通常被称为存储器地址的“本地节点(homenode)”。

[0003] 在基于目录的协议中,处理/高速缓存代理可向本地节点发送请求以访问(access)与相应“本地代理”相关联的存储器地址。相应地,这种计算机系统的性能可直接依赖于如何有效地管理本地代理数据和/或存储器。

发明内容

[0004] 本发明一方面涉及一种设备,包括:第一代理,将数据存储在与耦合至所述第一代理的第一存储器中;以及第一逻辑,在第二代理对所述第一存储器中的所存储数据进行镜像,其中将以存储器控制器粒度复制所镜像数据。

[0005] 本发明另一方面涉及一种方法,包括:将数据存储在与耦合至第一代理的第一存储器中;以及在第二代理对所述第一存储器中的所存储数据进行镜像,其中将以存储器控制器粒度复制所镜像数据。

[0006] 本发明再一方面涉及一种系统,包括:存储目录的存储器,所述目录存储与每个高速缓存行被高速缓存在哪个代理且以什么状态进行高速缓存有关的信息;以及将数据存储在与所述存储器中的第一代理,其中所述第一代理将包括在第二代理对所述存储器中的所存储数据进行镜像的第一逻辑,其中将以存储器控制器粒度复制所镜像数据。

附图说明

[0007] 参照附图提供详细说明。在附图中,参考标号的最左边数字识别参考标号首次出现的附图。相同参考标号在不同附图中的使用指示相似或相同项目。

[0008] 图1-2和图8-9示出可用于实现本文所讨论各种实施例的计算系统的实施例的框图。

[0009] 图3-7示出根据一些实施例的流程图。

具体实施方式

[0010] 在下面的描述中,提出多种具体细节以提供对各种实施例的透彻理解。然而,没有这些具体细节也可实现一些实施例。在其他情况下,没有详细描述众所周知的方法、过程、部件和电路以免混淆特定实施例。

[0011] 本文所讨论的一些实施例一般涉及在本地代理的存储器镜像和/或迁移。在实

施例中,迁移和 / 或镜像功能性可通过互连网络 (例如,本文如参照图 1 所述的网络结构) 无缝实现。此外,存储器可在存储器控制器粒度 (granularity) (例如,相对于在存储器装置级粒度进行复制的存储器装置级热备 (sparing) (例如双列直插存储器模块 (DIMM :Dual In-line Memory Module) 或动态随机存取存储器 (DRAM) 级)) 进行复制。此外,对从属 (slave) HA 的读取可以在从主要 (primary) 存储器控制器接收到 UNCORR 响应 (指示不可校正错误) 时进行转发,从而增加正常运行时间 (up time)。此外,在实施例中,镜像和 / 或迁移可以对操作系统是透明的。

[0012] 通常,计算系统中的高速缓冲存储器可利用侦听总线或基于目录的协议来保持一致。在任一情况下,系统存储器地址可与系统中的特定存储单元 (location) 相关联。这个存储单元 (location) 通常被称为存储器地址的“本地节点”。在基于目录的协议中,处理 / 高速缓存代理可向本地节点发送请求以访问与“本地代理”相关联的存储器地址。另外,在分布式高速缓存一致性协议中,高速缓存代理可向控制对相应存储空间的一致访问 (access) 的本地代理发送请求。而本地代理负责确保将所请求数据的最新副本 (copy) 从拥有所请求数据的存储器或高速缓存代理返回给请求方。例如,如果请求针对专有副本,则本地代理可还负责使在其他高速缓存代理的数据副本无效。为达到这些目的,本地代理通常可侦听每一个高速缓存代理或依靠目录 (例如,图 1 的目录高速缓存 122 或存储在如图 1 中存储器 120 的存储器中的目录的副本) 来追踪数据可驻留 (reside) 的高速缓存代理的集合。在实施例中,目录高速缓存 122 可包括存储在存储器 120 中的目录的全部或部分副本。

[0013] 根据一个实施例,在基于目录的高速缓存一致性协议中,监视或管理存储器的一个或多个代理 (被称为本地代理或 HA) 可共同维持部分追踪在系统中 (例如在快速路径接口 (QPI :Quick Path Interface) 系统中) 每个高速缓存行 (line) 被高速缓存在哪里以及以什么状态进行高速缓存的目录。想要获取高速缓存行的高速缓存代理 (CA :caching agent) 将其请求发送给 HA, HA 查找目录并仅向目录指示可能已经高速缓存了那个行的副本的那些 CA 发送侦听。如果目录知晓没有高速缓存副本存在,或者仅共享副本存在并且该请求针对另一共享副本,则根本不需要发送侦听并且从存储器满足该请求。因此,目录可消除 (或至少减小) 对每个请求侦听所有 CA 的需要,并减小侦听的带宽要求。系统允许 CA 请求在高速缓存分级结构中已被修改的高速缓存行。这样的 CA 请求在本文被称为 buriedM 请求。在一个实施例中,提供在追踪器中附加有单个位 (DEFUNCT 位) 的 buriedM 流程 (flow) (参见例如图 6)。此外,由于对例如冲突列表的数据结构重排序以支持 OOO (Out Of Order, 无序) 完成会在硬件中变得非常昂贵 (例如,由于与重排序关联的等待时间和 / 或硬件利用),实施例允许 HA 在不必要在 HA 对冲突列表重排序的情况下支持 buriedM。

[0014] 另外,如上文所讨论的,目录消除 (或至少减小) 对每个请求侦听所有 CA 的需要并减小侦听的带宽要求。此外,HA 可将来自不同 CA 的冲突请求接收到同一高速缓存行。在实施例中,串行化这样的冲突请求并确保以公平顺序服务所有 CA 是 HA 的职责。而且,在源 - 侦听一致性协议中,HA 可通过保持冲突高速缓存行的冲突方列表来解析冲突。然后,它以 FIFO (First-In, First-Out, 先入先出) 方式服务冲突方。这个列表通常随系统中高速缓存代理的数量改变大小 (scale) 并且可在大型无粘接 (glueless) (例如 8 个插槽 (socket) 及以上) 配置中变得异乎寻常的大。然而,在基于本地 - 侦听目录的一致性协议中,按照一

个实施例（例如，如参照图 7 所讨论的），HA 可采用该协议的定义来将这个列表限制到仅单个位。

[0015] 本文所述的：

[0016] ●“主代理 (primary)”指的是主要本地代理，例如迁移 / 镜像操作的源；

[0017] ●“从代理 (slave)”或“镜像代理 (mirror)”指的是从属 / 镜像本地代理，例如迁移 / 镜像操作的目标；

[0018] ●“迁移 (migration)”指的是在从代理建立一致数据副本的过程。在主代理的所有写入（隐式写回 (IWB)、显式写回 (EWB)）也将被发送到从代理（作为 NcWr [Pt1]）；

[0019] ●“镜像 (mirroring)”指的是从代理拥有一致数据副本（例如，继续发送 IWB/EWB 给从代理，此外，为数据而将在主要存储器控制器导致未校正响应的读取发送给从代理）；

[0020] ●“读故障恢复 (read failover)”（在镜像期间）指的是当在主代理的读取遇到未校正错误时从主代理发送读取到从代理；

[0021] ●“硬故障恢复 (hard failover)”（在镜像期间）指的是由于主要存储器控制器被确定为死机 (dead)（因为对主代理的写入失败，或者 N 次读取遇到未校正错误，并且 N 超过所编程阈值）而无条件地向从代理发送读取；

[0022] ●“HA 故障恢复 (HA failover)”（迁移后）指的是对地址解码器重新编程以指向从属 HA 以供存储器引导 (reference)，从而使其为新主要 HA；

[0023] ●“主代理_死机 (primary_dead)”指的是主要存储器控制器被认定为死机 / 不可用；以及

[0024] ●“从代理_死机 (slave_dead)”指的是从属存储器控制器被认定为死机 / 不可用。

[0025] 各种计算系统可用于实现这里讨论的实施例，例如参照图 1 和图 8-9 所讨论的系统。更具体而言，图 1 示出根据本发明实施例的计算系统 100 的框图。系统 100 可包括一个或多个代理 102-1 到 102-M（本文统称为“多个代理 102”或更一般地称为“代理 102”）。在实施例中，代理 102 中的一个或多个可以是例如参照图 8-9 所讨论计算系统的计算系统的部件中的任何部件。

[0026] 如图 1 所示，代理 102 可经由网络结构 104 进行通信。在一个实施例中，网络结构 104 可包括允许各个代理（例如计算装置）通信数据的计算机网络。在实施例中，网络结构 104 可包括一个或多个通过串行（例如点对点）链路和 / 或共享通信网络进行通信的互连（或互连网络）。例如，一些实施例可促进在允许与全缓冲双列直插存储器模块 (Fully Buffered Dual in-line memory modules, FBD) 通信的链路上的部件调试或验证，例如，其中 FBD 链路是将存储器模块耦合到主机 (host) 控制器装置（例如处理器或存储器集线器）的串行链路。可从 FBD 通道主机 (channel host) 传送调试信息，使得可由通道流量踪迹捕获工具（例如一个或多个逻辑分析器）沿该通道观察调试信息。

[0027] 在一个实施例中，系统 100 可支持分层协议方案，该协议方案包括物理层、链路层、路由层、传输层和 / 或协议层。对于点对点或共享网络，结构 104 还可促进数据（例如具有分组的形式）从一种协议（例如高速缓存处理器或高速缓存察觉存储器控制器）到另一种协议的传送。另外，在一些实施例中，网络结构 104 可提供遵循一个或多个高速缓存一致性协议的通信。

[0028] 此外,如图 1 中箭头方向所示,代理 102 可通过网络结构 104 传送和 / 或接收数据。因此,一些代理可利用单向链路而其他代理可利用双向链路来通信。例如,一个或多个代理 (如代理 102-M) 可传送数据 (例如通过单向链路 106),其他代理 (例如代理 102-2) 可接收数据 (例如通过单向链路 108),而一些代理 (例如代理 102-1) 可以既传送又接收数据 (例如通过双向链路 110)。

[0029] 此外,代理 102 中的至少一个可以是本地代理,并且代理 102 中的一个或多个可以是本文将进一步讨论的请求代理或高速缓存代理。如图所示,至少一个代理 (仅一个示为代理 102-1) 可包括或访问一个或多个逻辑 (或引擎) 111 以便将数据镜像、迁移数据、解析 buriedM 和 / 或解析冲突,如本文例如参照图 3-7 所述。此外,在实施例中,代理 102 中的一个或多个 (仅一个示为代理 102-1) 可访问诸如存储器 120 的存储器 (其可专用于该代理或与其他代理共享)。另外,代理 102 中的一个或多个 (仅一个示为代理 102-1) 可维持一个或多个存储装置 (仅一个示为代理 102-1,例如目录高速缓存 122,例如实现为表、队列、缓冲器、链接列表等) 中的条目 (entry) 以追踪与系统中代理 102-1 (作为本地代理) 和 / 或其他代理 (例如包括 CA) 所存储 / 维持的项目有关的信息。在一些实施例中,代理 102 中的每个或至少一个可耦合到在与该代理相同的管芯 (die) 上或者可由该代理以别的方式访问的存储器 120 和 / 或相应目录高速缓存 122。

[0030] 图 2 是根据实施例的计算系统的框图。系统 200 可包括多个插槽 202-208 (示出四个,但是一些实施例可以具有更多或更少插槽)。在实施例中,每个插槽可包括处理器。此外,每个插槽可通过例如参照图 9 讨论的点对点 (PtP) 链路耦合到其他插槽。如针对图 1 参照网络结构 104 所讨论的,每个插槽可耦合到系统存储器的局部 (local) 部分,系统存储器例如由多个可包括动态随机存取存储器 (DRAM) 的双列直插存储器模块 (DIMM) 形成。

[0031] 如图 2 所示,每个插槽可耦合到存储器控制器 (MC) / 本地代理 (HA) (例如 MC0/HA0 到 MC3/HA3)。存储器控制器可耦合到可以是系统存储器 (例如图 9 的存储器 912) 一部分的相应局部存储器 (标记为 MEM0 到 MEM3)。在一些实施例中,存储器控制器 (MC) / 本地代理 (HA) (例如 MC0/HA0 到 MC3/HA3) 可以与图 1 中代理 102-1 (例如包括逻辑 111 等) 相同或相似,并且标记为 MEM0 到 MEM3 的存储器可以与图 1 中存储器 120 相同或相似。此外,在一个实施例中,MEM0 到 MEM3 可配置成将数据镜像,例如作为主数据 (master) 或从数据 (slave)。此外,在一些实施例中,系统 200 的一个或多个部件可包含在同一集成电路管芯上。

[0032] 因此,例如图 2 所示的实现可针对采用镜像的插槽无粘接 (glueless) 配置。例如,指派给存储器控制器 (例如 MC0/HA0) 的数据可通过 PtP 链接被镜像到另一存储器控制器 (例如 MC3/HA3)。此外,与存储器控制器 MC3/HA3 关联的目录可在复制到镜像代理时以未知 (U) 状态初始化。在对这个控制器的故障恢复 (例如,由于对这个存储器控制器的在线服务 - 调用 (call)) 时,可从 U 状态重建目录。

[0033] 在实施例中,软件提取层 (例如 BIOS (Basic Input/Output System, 基本输入 / 输出系统)) 迁移流程 (flow) 可包括例如使用信号量环路 (semaphore loop) 将源存储器的所有行标记为 “M” (表示相应行的修改 / 迁移)。

[0034] 例如,下面的伪码可用于各 64B 高速缓存 - 行 :

[0035] //r2 是该 64B 高速缓存行的最低 8 字节的地址

[0036] ld8 r1 < -[r2]

[0037] mov ar.ccv < -r1

[0038] // 比较 ar.ccv 和 [r2], 如果成功则将 r1 写回到 [r2], 如果不成功则行已被另一代理修改

[0039] cmpxchg r8 < -[r2], r1, ar.ccv

[0040] ; ;

[0041] Fcr2

[0042] 高速缓存行的刷新 (上述伪码中的 Fcr2) 将使数据被写入 HA。如果在 HA 中启用迁移, 数据将被复制到从代理。

[0043] 参照图 3-7 所讨论的操作可由参照图 1、2、8 或 9 所讨论的部件执行, 其中图 3 示出按照实施例的显式写回操作的迁移流程 (flow) 的流程图。如图所示, CA0 发送显式写回 (WbM2i/WbIdata, 例如指示 CA0 在高速缓存行上的在先存储操作之后正在驱逐 (evict) 那个高速缓存行) 给主要 HA。在该主要 HA 启用迁移。该主要 HA 将数据写入 (Write) 到主要存储器 (其中目录是 I), 此外, 在迁移模式中, 主要 HA 发送 NcWr (例如 Non-cohWrite- 指示数据的副本正被发送到从代理) 给从属 HA。从属 HA 将该写入提交给从属存储器 (目录为 U) 并将完成 (Cmp) 发送给主要 HA。在主要 HA 接收到来自从属 HA 的 Cmp 之后, 它将完成 (Cmp) 发送给最初请求方 CA0。

[0044] 图 4 示出按照实施例的隐式写回操作的迁移流程 (flow) 的流程图。如图所示, 主要 HA 代表来自 CA1 的 RdInvOwn (例如, 指示对所有权的读取) 将 SnpInvOwn (例如, 指示使当前所有者无效的侦听) 发送给 CA0。SnpInvOwn 又导致从 CA0 的隐式写回 (RspIWb/WbIdata, 例如指示 CA0 正在写回无效 (dirty) 数据)。主要 HA 将写入提交给主要存储器 (目录为 E@CA1)。此外, 主要 HA 发送 NcWr 给从属 HA。从属 HA 将写入提交给从属存储器 (目录为 U) 并且发送 Cmp (例如指示完成) 给主要 HA。在接收到来自从属 HA 的 Cmp 时, 主要 HA 发送 DataE_Cmp 给 CA1 以完成请求。

[0045] 图 5 示出按照实施例的读故障恢复操作的流程图。就镜像时的读故障恢复而言, 当主代理 - 读取 (primary-read) 返回不可校正信号 (UNCORR) 时, 将侦听广播给参与向量 (假设目录高速缓存中有遗漏 (miss)) 中的所有代理, 并解析侦听响应。此外, 对镜像代理的读取 (假设没有来自侦听的无效写回) 可被弹回 (bounce), 并且将数据直接从镜像代理 (mirror agent) 提供给请求方。

[0046] 在实施例中, 读故障恢复可逐行进行, 而不是在第一次遇到不可校正错误时失去对整个主要存储器的访问。实施例实现可编程阈值计数器 (对不可校正错误的数量进行计数), 并对于读操作仅在这个阈值已被超过之后禁用整个存储器控制器。在写操作的单个不可校正错误之后或者如果主要存储器控制器发信号通知链路, 整个存储器控制器可被禁用 (例如由于硬故障恢复)。在从代理的校正数据可不被主动写回到主代理。然而, 存储器控制器可将不可校正错误的地址记入日志并发信号通知不可恢复错误。逻辑 (例如逻辑 111) 也可获取该行的所有权并将它刷新到主代理而没有对该数据的任何改变。

[0047] 对于在主代理的目录更新, 不可校正行可能已在主代理如受损 (poisoned) 那样被写回。在这样的情况下, 该行的后续读取不会接收到来自存储器控制器的 UNCORR 响应 (假设软错误引起 UNCORR), 而是返回受损行。受损行会使 HA 将读取也弹回到从属 HA。

[0048] 就 HA 故障恢复（迁移后）而言，按照一个实施例，在将高速缓存代理地址解码器指向原（old）从代理（新的主代理）之后的流程如下。首先，在新的主要目录中命中 U（未知）状态的任何行将广播侦听。无论读取何时出现 HA 将发布 SnpInv*（并在该目录中以 E 状态标记该行），且该目录指示 FBD 目录（例如其可被存储在图 1 的目录高速缓存 122 和 / 或存储器 120 中）中的 U。注意，原从属（新的主要）目录在迁移 / 写入 - 镜像过程期间可以作为 U 写入。

[0049] 在实施例中，不要求运行对通过地址空间的所有权环路的刷新或读取以去除目录中的 U 状态，在某种程度上，因为在对 U 状态行的首次存取（access）时，HA 可以 E 状态标记该行（RdCode/RdData/RdCur 请求将广播 SnpInvItoE 并且在请求方将该行标记为 E）。失效请求（RdInvOwn, InvI2E）也可以在请求方将该行标记为 E。对 U 状态行的 EvictCln（例如，指示驱逐清除行命令）将继续把该行保留在 U 状态。

[0050] 参照图 5, CA0 发送 RdData 给主要 HA。来自主要 HA 的相应读取导致 UNCORR 响应。主要 HA 将该读取重定向到从属 HA（经由 NCRd），此外，它将主要 HA 中的目录写入 E@CA0。注意，可在主要存储器中设置损坏（poison）位。在接收到 NcRd 时，从属 HA 读取从属存储器，并且将 DataE_Cmp 直接发送回最初请求方 CA0（注意，从属目录继续保持在 U 状态）。

[0051] 在实施例中，就镜像 / 迁移流程而言，对故障恢复到从属本地代理的读取采用下列流程。主要本地代理向具有下列参数的从属本地代理发布 NcRdPt1（例如，指示 Non-coherentReadPartial 命令，其是用于发送故障恢复读取的特殊命令）。主要本地代理将这个分组作为“发出就不管（fire and forget）”分组（例如，其中不预期来自主要本地代理的响应）进行发送。

[0052] • RHNID- 引起读故障 - 恢复（fail-over）的事务的请求方的 NID（网络标识符）。这将被缩写为 Orig_Req_RHNID

[0053] • RTID- 引起读故障恢复的事务的 RTID（事务标识符）。这将被缩写为 Orig_RTID

[0054] • DNID- 从属本地 NID。这将被缩写为 Slave_NID

[0055] • NCRdPt1 中的长度字段 -' 00-dataNC, 01-dataI, 10-dataE, 11-dataS- 向从代理指示数据应该如何返回

[0056] 从属本地代理可发布 DataC* 或 DataNC 分组给具有下列参数的最初请求方。注意，从代理可不把响应发送回主要本地代理：

[0057] • RHNID-Primary_Home_agent NID（来自从属 HA 中的 CSR）

[0058] • RTID-NcRdPt1 的 RTID(Orig_RTID)

[0059] • DNID-NcRdPt1 的 RHNID。这可与 Orig_Requester_RHNID 相同

[0060] 这个方案的一个优点在于：用于镜像 - 故障恢复的主要本地代理流程（flow）大为简化，因为它不需要接收来自从代理的数据并将该数据向前转发到最初请求方。此外，对从代理的欺骗读取（spoofed Read）从从代理的角度来看类似于正常读取，因此对充当从代理的本地代理只要求小的改变。

[0061] 在一个实施例中，主要代理不接收对其发布的事务的响应（NcRdPt1）。通常，在此情况下它应该接收 DataNC。作为代替，这种行为依赖于主要和次要本地代理之间关于次要本地代理将满足最初请求而不满足从主要本地代理（primary）到次要本地代理（secondary）的表面（apparent）请求的“协定（agreement）”（其可以在初始化时进行）。

此外,主代理(primary)可使用 NcRdPt1 的较低地址位(其在某些实施例中对于故障-恢复读取不是必需的)的长度把信息隧道传送(tunnel)给从代理(slave)。从代理可又使用这种经隧道传送信息来确定如上所述的正确响应类型。

[0062] 此外,从主要本地代理(primary)到次要本地代理(secondary)的请求仍然是不同的“全局唯一 ID(globally unique ID)”,因为 DNID 是唯一的(即从主要本地代理(primary)到次要本地代理(secondary)的 DNID 不同于在最初请求/事务中的 DNID)。最初请求高速缓存代理不受任何这种改变影响。

[0063] 在一些实施例中,写入(Write)镜像流程如下进行。主要本地代理发布 NcWr[Pt1]给从属本地代理以提交写入数据(来自隐式/显式写回,或 ncwr):

[0064] • RHNID-最初请求方 RHNTD

[0065] • RTID-最初请求方 RTID

[0066] • DNID-从属本地节点 DNID

[0067] 从属本地代理(例如,在提交写入给从属存储器控制器后)可向主要本地代理以 Cmp 分组响应。在一个实施例中,这个分组具有下列字段:(1)RHNID-从属本地代理 NID;(2)RTID-来自 NcWr[pt1]的 RTID。在此情况下,它是最初请求方 RTID;(3)DNID-主要配置-代理(ubox)的 NID([5:3]-主要 HA 插槽位,[2:0]-ubox nid(0x2));和/或(4)RSNID-来自 NcWr[pt1]的 RHNID。

[0068] 图 6 示出按照实施例的 buriedM 操作的流程图。在一些实施例中,HA 实现将多个请求以 FIFO 方式服务给同一高速缓存行。参照图 6,CA1 发送 RdInvOwn 到地址 A。由于 HA 首先看到这个请求,它致力于首先为其服务。在 CA0 的高速缓存分级结构(如图 6 中“M”所指示的)中修改所涉及的地址。但 CA0 同样已经向同一地址发出请求。在 HA 的追踪器数据结构中使 CA0 的请求下沉(sink)(或将其丢弃)。在 HA 的目录可反映该地址为 E@CA0 的事实。因此,HA 发送 SnpInvOwn 给 CA0(代表 CA1 的请求)。当 CA0 接收到该响应时,它响应 RspCnfltOwn(例如,指示 CA0 已经接收到对在 CA0 的高速缓存分级结构中被标记为已修改(Modified)的行的冲突侦听)。这是因为该行在其高速缓存分级结构中是 buriedM。

[0069] 如上所讨论的,HA 已致力于服务 CA1,它仍然需要侦听出来自 CA0 的高速缓存的该行的最新副本。HA 能这样做的一种方式首先是完成 CA0 的无序(000)。HA 完成 CA0 的请求 000,并在 CA0 的追踪器条目中设置指示当前请求是 DEFUNCT 的位。它然后完成与 CA0 的最近冲突握手,从而从 CA0 的高速缓存提取该行的最新副本。将该行的最新副本转发给 CA1,然后 HA 最终完成 CA1 的请求。当 CA0 的最初请求在 HA 得到(come up for)服务时,HA 检测到该请求已被标记 DEFUNCT,这暗示该请求已完成 000。因此,来自 CA0 的请求只是被 HA 丢弃。

[0070] 在一些实施例中,可以修改上述流程,使行(当 HA 完成 CA0 的请求 000 时)CA0 可在它接收到其 000 完成响应并完成其冲突握手之后发送另一请求来重新使用同一追踪器-索引。在这种情形下,HA 可再次将 CA0 的追踪器条目标记为 ACTIVE。稍后,当 CA0 的(较新)请求得到服务时,它通常由 HA 完成。

[0071] 相应地,在一个实施例中,提供在追踪器中增加有单个位(DEFUNCT 位)的 buriedM 流程(flow)。重排序例如冲突列表的数据结构以支持 000(无序)完成在硬件中非常昂贵。实施例允许 HA 支持 buriedM 而不必在 HA 重排序冲突列表。

[0072] 图 7 示出按照实施例的冲突处理操作的流程图。在实施例中, HA 将为系统中的所有 CA 请求预先分配空间。这种预先分配的数据结构被定义为“追踪器 (Tracker)”。在源-侦听协议中, 各追踪器条目可以具有冲突追踪器条目向量。例如, 对于 512 条目追踪器 (9 位追踪器索引), 各条目将具有 $9 \times 16 = 144$ 位宽的冲突列表-假定在系统中有 16 个 CA, 在 8-插槽无粘接 (glueless) 配置中的每个插槽具有两个逻辑 CA, 这是典型的 8-插槽无粘接 (glueless) 配置。用于所有冲突列表的总存储量 (total storage) 为 512×144 (73728 位)。相比之下, 在实施例中, 在 HA (基于本地-侦听目录的 HA), 冲突仅由追踪器条目中的单个位来追踪。

[0073] 在一些实施例中, 这样的方案可考虑两种类型的冲突。早期 (early) 冲突是其中冲突 CA 还没有高速缓存行的副本的冲突类型。在实施例中, HA 忽略这种冲突, 并且将冲突侦听响应作为正常侦听响应对待。第二种类型的冲突被称为最近的 (或实际的) 冲突。在这种冲突中, 提供冲突响应的 CA 实际上具有该行的最新副本。在这种情况下, 发送第二 (继续 (follow-up)) 侦听以便提取该行的最新副本是 HA 的职责。当 HA 接收到最近的冲突侦听响应时, 它可以在响应发送方的追踪器条目中设置位。该立指示 CA (其发送了冲突响应) 涉及到最近冲突中。另外, CA (其发送冲突响应) 可采用“AckCnflt”消息来继续该响应。当 HA 接收到对具有所设置最近冲突位的追踪器索引的 AckCnflt 响应时, 它可以发送 CmpFwd (侦听类型) 响应。这允许 HA 从发送了最近冲突响应的 CA 中提取该行的最新副本。

[0074] 参照图 7, HA 发送 DataE_Cmp 响应给 CA1 以完成其请求。它把该高速缓存行的目录更新到 Exclusive@CA1。接下来, HA 代表来自 CA2 的 RdInvOwn 请求侦听 CA1 (在实施例中 HA 实现基于目录的本地侦听协议)。对 CA1 的侦听在系统结构中传递 DataCE_Cmp。CA1 以 RspCnflt 响应, 因为从它的观点来看其请求还没有完成。当 HA 接收到来自 CA1 的 RspCnflt 时, 它查看对应追踪器条目, 并注意到 CA1 的请求已被完成, 它在 CA1 的追踪器条目中设置位, 并且等待 AckCnflt (其在一些实施例中是需要的)。

[0075] 当来自 CA1 的 AckCnflt 到达时, HA 注意到在 CA1 的追踪器条目中的最近冲突位被设置。因此, 它发送 CmpFwd* 侦听给 CA1, 要求它转发该高速缓存行给下一冲突方 (CA2)。接收到没有设置最近冲突位的 AckCnflt 会使 HA 发送 Cmp 响应, 而不是侦听 (CmpFwd) 响应。接着, HA 等待对侦听的响应 (CmpFwd)。当接收到侦听响应时, CA2 的请求被最终完成。

[0076] 可见, 与源-侦听协议的数据结构要求不同, 在本地-侦听协议中, HA 可仅仅通过冲突方的追踪器条目中的附加位来解析冲突。相应地, 可以使用非常有限的存储量 (storage) (例如一位每追踪器条目) 来解析冲突。这又允许按比例放大 (scale up) 的实现以解决 (address) 更大的市场部分。此外, 由于只需要一附加位每追踪器条目来解析冲突, 这种实施例使设计例如相比源-侦听对应物更加可缩放 (scalable)。

[0077] 图 8 示出计算系统 800 的实施例的框图。图 1 的代理 102 中的一个或多个可包括计算系统 800 的一个或多个部件。同时, 如图 8 所示, 系统 800 的各种部件可包括目录高速缓存 (例如, 诸如图 1 的目录高速缓存 122) 和 / 或逻辑 (诸如图 1 的逻辑 111)。然而, 目录高速缓存和 / 或逻辑可被提供在遍及系统 800 的位置 (location), 包括或不包括那些已示出的。计算系统 800 可包括一个或多个耦合至互连网络 (或总线) 804 的中央处理单元 (CPU) 802 (其在本文中可统称为“多个处理器 802”或更一般地称为“处理器 802”)。处理器

802 可以是任何类型的处理器,如通用处理器、网络处理器(其可处理通过计算机网络 805 通信的数据)等(包括简化指令集计算机(RISC)处理器或复杂指令集计算机(CISC))。另外,处理器 802 可以具有单核或多核设计。具有多核设计的处理器 802 可在同一集成电路(IC)管芯上集成不同类型的处理器核。同时,具有多核设计的处理器 802 可以实现为对称或非对称多处理器。

[0078] 处理器 802 可包括一个或多个高速缓存(例如,不同于图示的目录高速缓存 122),这些高速缓存在各种实施例中可以是专用的和/或共享的。一般而言,高速缓存存储与存储在其它地方或较早计算的最初数据对应的数据。为了减小存储器访问(access)的等待时间,一旦数据被存储在高速缓存中,可通过存取高速缓存副本而不是重取或重算该最初数据来进行未来的使用。高速缓存可以是存储由系统 800 中一个或多个部件使用的电子数据(例如,包括指令)的任何类型的高速缓存,比如一级(L1)高速缓存、二级(L2)高速缓存、三级(L3)高速缓存、中间级高速缓存、末级高速缓存(LLC)等。另外,这种高速缓存可位于各种位置(location)(例如,在包括图 1、2、8 或 9 中系统的本文所讨论计算系统的其它部件内部)。

[0079] 芯片组 806 还可以耦合至互连网络 804。此外,芯片组 806 可包括图形存储器控制集线器(GMCH)808。GMCH 808 可包括耦合至存储器 812 的存储器控制器 810。存储器 812 可存储例如包括由处理器 802 或与计算系统 800 中部件通信的任何其它装置执行的指令序列的数据。同时,在本发明的一个实施例中,存储器 812 可包括一个或多个易失存储(或存储器)装置,例如随机存取存储器(RAM)、动态 RAM(DRAM)、同步 DRAM(SDRAM)、静态 RAM(SRAM)等。也可以使用诸如硬盘的非易失存储器。例如多个处理器和/或多个系统存储器的附加装置可以耦合至互连网络 804。

[0080] GMCH 808 还可包括耦合至显示装置 816 的图形接口 814(例如,在实施例中通过图形加速器)。在一个实施例中,图形接口 814 可以通过加速图形端口(AGP)耦合至显示装置 816。在本发明的实施例中,显示装置 816(例如平板显示器)可以通过例如信号转换器耦合至图形接口 814,信号转换器将存储在例如视频存储器或系统存储器(例如存储器 812)的存储装置中的图像数字表示变换成由显示器 816 解释并显示的显示言号。

[0081] 如图 8 所示,集线器接口 818 可将 GMCH 808 耦合至输入/输出控制集线器(ICH)820。ICH 820 可提供到耦合至计算系统 800 的输入/输出(I/O)装置的接口。ICH 820 可以通过例如可遵照 PCIe 规范的外围部件互连(PCI)桥的外围桥(或控制器)824、通用串行总线(USB)控制器等耦合至总线 822。桥 824 可在处理器 802 和外围装置之间提供数据通路。可以使用其它类型的拓扑。同时,多个总线可以例如通过多个桥或控制器耦合至 ICH 820。此外,总线 822 可包括其它类型和配置的总线系统。另外,在本发明的各种实施例中,耦合至 ICH 820 的其它外设可包括集成驱动电子电路(IDE)或小型计算机系统接口(SCSI)硬驱动、USB 端口、键盘、鼠标、并行端口、串行端口、软盘驱动、数字输出支持(例如数字视频接口(DVI))等。

[0082] 总线 822 可以耦合至音频装置 826、一个或多个盘驱动 828 和网络适配器 830(其在实施例中可以是 NIC)。在一个实施例中,耦合至总线 822 的网络适配器 830 或其它装置可与芯片组 806 通信。同时,在本发明的一些实施例中,各种部件(例如网络适配器 830)可以耦合至 GMCH 808。此外,可以合并处理器 802 和 GMCH 808 以形成单个芯片。在实施例

中,可以在 CPU 802 的一个或多个中提供存储器控制器 810。此外,在实施例中, GMCH 808 和 ICH 820 可以被合并到外围控制集线器 (PCH)。

[0083] 另外,计算系统 800 可包括易失和 / 或非易失存储器 (或存储装置)。例如,非易失存储器可包括如下中的一个或多个:只读存储器 (ROM)、可编程 ROM (PROM)、可擦 PROM (EPROM)、电 EPROM (EEPROM)、盘驱动 (例如 828)、软盘、密致盘 ROM (CD-ROM)、数字通用盘 (DVD)、闪速存储器、磁光盘或能够存储电子数据 (例如包括指令) 的其它类型的非易失机器可读介质。

[0084] 在实施例中,存储器 812 可包括如下中的一个或多个:操作系统 (O/S) 832、应用 834、目录 801 和 / 或装置驱动器 836。存储器 812 也可以包括专用于存储器映射 I/O (MMIO: Memory Mapped I/O) 操作的区域。存储在存储器 812 中的程序和 / 或数据可以作为存储器管理操作的一部分被交换 (swap) 到盘驱动 828 中。应用 834 可执行 (例如在处理器 802 上),以便与耦合至网络 805 的一个或多个计算装置通信一个或多个分组。在实施例中,分组可以是可由一个或多个从至少一个发送方传送到至少一个接收方 (例如通过如网络 805 的网络) 的电信号进行编码的一个或多个符号和 / 或值的序列。例如,每个分组可以具有报头,报头包括可在路由和 / 或处理该分组时使用的各种信息,例如源地址、目的地址、分组类型等。每个分组也可以具有包括原始数据 (或内容) 的有效载荷,分组通过计算机网络 (例如网络 805) 在各种计算装置之间转送。

[0085] 在实施例中,应用 834 可利用 O/S 832 来例如通过装置驱动器 836 与系统 800 的各种部件通信。因此,装置驱动器 836 可包括网络适配器 830 专用命令以便例如通过芯片组 806 在 O/S 832 和网络适配器 830 或者耦合至系统 800 的其它 I/O 装置之间提供通信接口。

[0086] 在实施例中,O/S 832 可包括网络协议栈。协议栈一般是指可被执行以处理通过网络 805 发送的分组的过或程序集,其中这些分组可符合指定协议。例如,TCP/IP (传输控制协议 / 因特网协议) 分组可以使用 TCP/IP 栈进行处理。装置驱动器 836 可指出存储器 812 中将例如通过协议栈进行处理的缓冲器。

[0087] 网络 805 可包括任何类型的计算机网络。网络适配器 830 还可以包括直接存储器存取 (DMA) 引擎,该引擎将分组写入分配到可用描述符 (例如被存储在存储器 812 中) 的缓冲器 (例如被存储在存储器 812 中),以便通过网络 805 发送和 / 或接收数据。另外,网络适配器 830 可包括网络适配器控制器,网络适配器控制器可包括执行适配器相关操作的逻辑 (例如一个或多个可编程处理器)。在实施例中,适配器控制器可以是 MAC (媒体访问控制) 部件。网络适配器 830 还可以包括存储器,例如任何类型的易失 / 非易失存储器 (例如,包括一个或多个高速缓存和 / 或参照存储器 812 所讨论的其它存储器类型)。

[0088] 图 9 示出按照本发明的实施例以点对点 (PtP) 配置进行设置的计算系统 900。特别地,图 9 示出其中处理器、存储器和输入 / 输出装置通过多个点对点接口互连的系统。参照图 1-8 所讨论的操作可以由系统 900 的一个或多个部件执行。

[0089] 如图 9 所示,系统 900 可包括若干处理器,为清楚起见只示出其中的两个处理器 902 和 904。处理器 902 和 904 可各自包括局部存储器控制器集线器 (GMCH) 906 和 908 以实现与存储器 910 和 912 的通信。存储器 910 和 / 或 912 可存储各种数据,例如参照图 9 的存储器 912 所讨论的那些数据。如图 9 所示,处理器 902 和 904 (或系统 900 中例如芯片

组 920、I/O 装置 943 等的其它部件)也可以包括一个或多个高速缓存,例如参照图 1-8 所讨论的那些高速缓存。

[0090] 在实施例中,处理器 902 和 904 可以是参照图 9 所讨论的处理器 902 中之一。处理器 902 和 904 可分别使用 PtP 接口电路 916 和 918 通过点对点 (PtP) 接口 914 交换 (exchange) 数据。同时,处理器 902 和 904 可各自通过单独的 PtP 接口 922 和 924 使用点对点接口电路 926、928、930 和 932 来与芯片组 920 交换数据。芯片组 920 还可以通过高性能图形接口 936 例如使用 PtP 接口电路 937 来与高性能图形电路 934 交换数据。

[0091] 在至少一个实施例中,可以在处理器 902、904 和 / 或芯片组 920 的一个或多个中提供目录高速缓存和 / 或逻辑。然而,本发明的其它实施例可以存在于图 9 的系统 900 内的其它电路、逻辑单元或装置中。此外,本发明的其它实施例可以遍布于图 9 示出的若干电路、逻辑单元或装置。例如,系统 900 的各种部件可包括目录高速缓存 (例如,诸如图 1 的目录高速缓存 122) 和 / 或逻辑 (诸如图 1 的逻辑 111)。然而,该目录高速缓存和 / 或逻辑可提供在遍及系统 900 的位置,包括或不包括那些所示出的位置。

[0092] 芯片组 920 可以使用 PtP 接口电路 941 与总线 940 通信。总线 940 可以具有一个或多个与之通信的装置,例如总线桥 942 和 I/O 装置 943。通过总线 944,总线桥 942 可以与例如键盘 / 鼠标 945、通信装置 946 (例如调制解调器、网络接口装置或可以与计算机网络 905 通信的其它通信装置)、音频 I/O 装置和 / 或数据存储装置 948 的其它装置通信。数据存储装置 948 可以存储可由处理器 902 和 / 或 904 执行的代码 949。

[0093] 在本发明的各种实施例中,本文中例如参照图 1-9 所讨论的操作可以被实现为硬件 (例如电路)、软件、固件、微代码或它们的组合,其可被提供为例如包括具有其上存储的指令 (或软件程序) 的机器可读或计算机可读介质的计算机程序产品,指令 (或软件程序) 用于将计算机编程成执行本文讨论的过程。此外,术语“逻辑”可以示例方式包括软件、硬件或者软件和硬件的组合。机器可读介质可以包括存储装置,例如参照图 1-9 讨论的那些存储装置。另外,这种计算机可读介质可以作为计算机程序产品被下载,其中程序可以经由通信链路 (例如总线、调制解调器或网络连接) 通过在载波或其它传播介质中提供的数据信号从远程计算机 (例如服务器) 传送到请求计算机 (例如客户端)。

[0094] 说明书中对“一个实施例”或“实施例”的引用表示,结合该实施例描述的特定特征、结构或者特性可被包含在至少一个实现中。短语“在一个实施例中”在说明书中各个地方的出现可以是或可以不是都指向同一实施例。

[0095] 同时,在说明书和权利要求书中,可以使用术语“耦合”和“连接”以及它们的派生词。在本发明的一些实施例中,可以使用“连接”来表示两个或更多要素彼此处于直接的物理接触或电接触。“耦合”可表示两个或更多要素处于直接的物理接触或电接触。然而,“耦合”也可以表示两个或更多要素彼此不处于直接接触,但仍可以彼此合作或交互。

[0096] 由此,虽然已经用专用于结构特征和 / 或方法动作的语言描述了本发明的实施例,但是要理解,要求保护的主体可不被限制到所描述的具体特征或动作。而是将这些具体特征和动作作为实现要求保护的主题的范例形式进于公开。

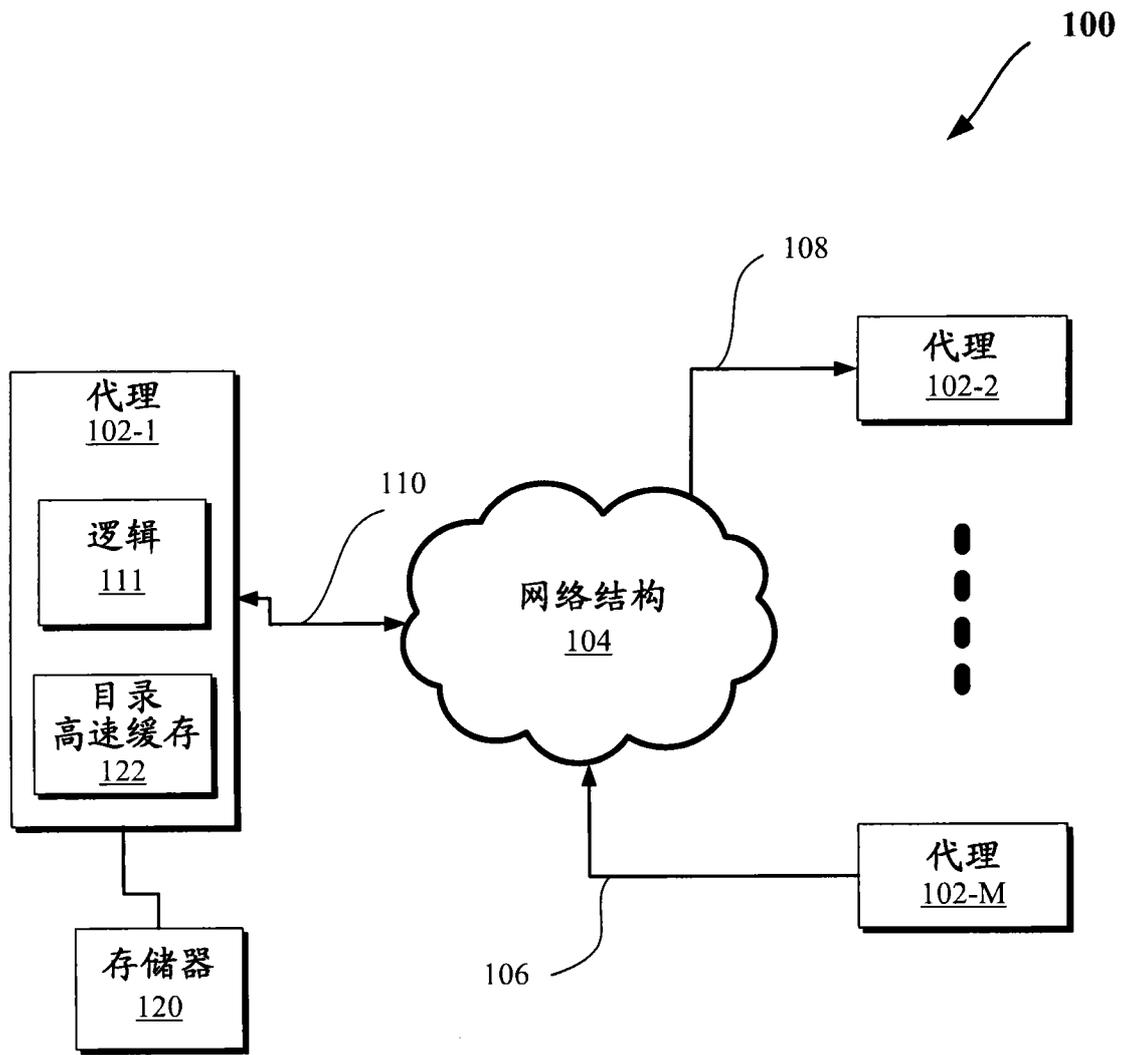


图 1

200

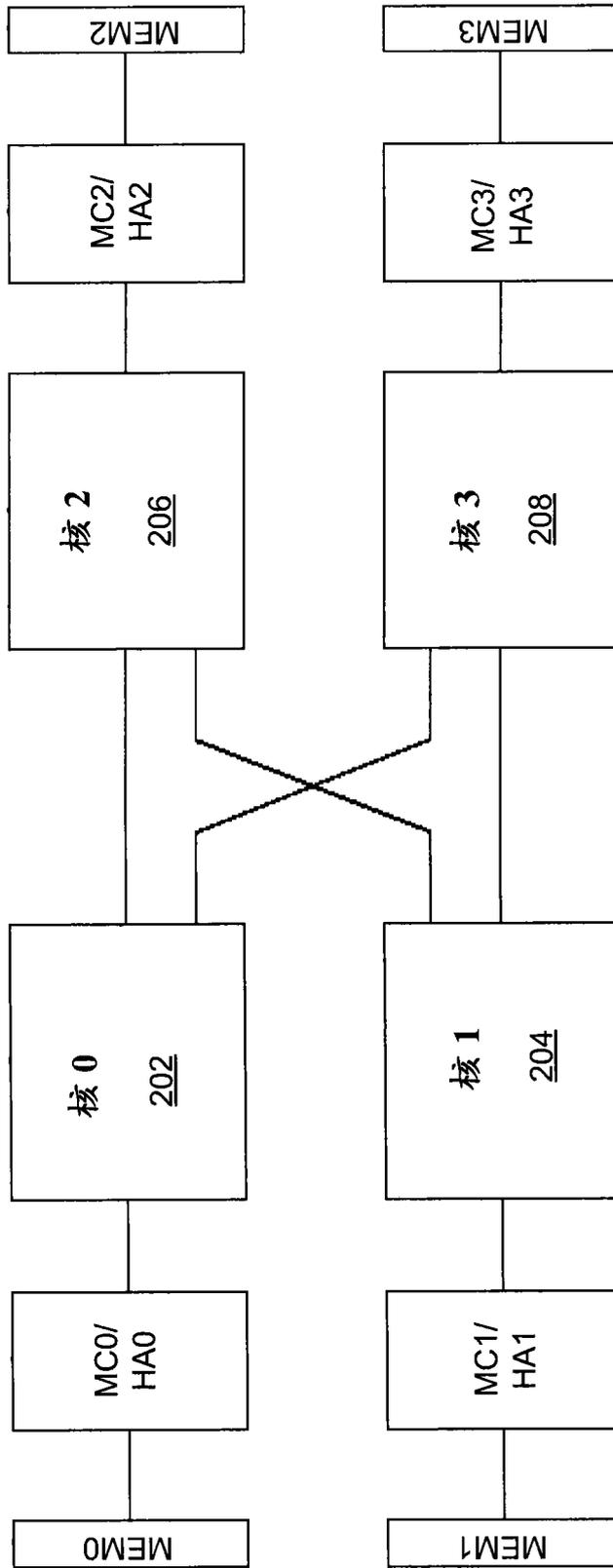


图 2

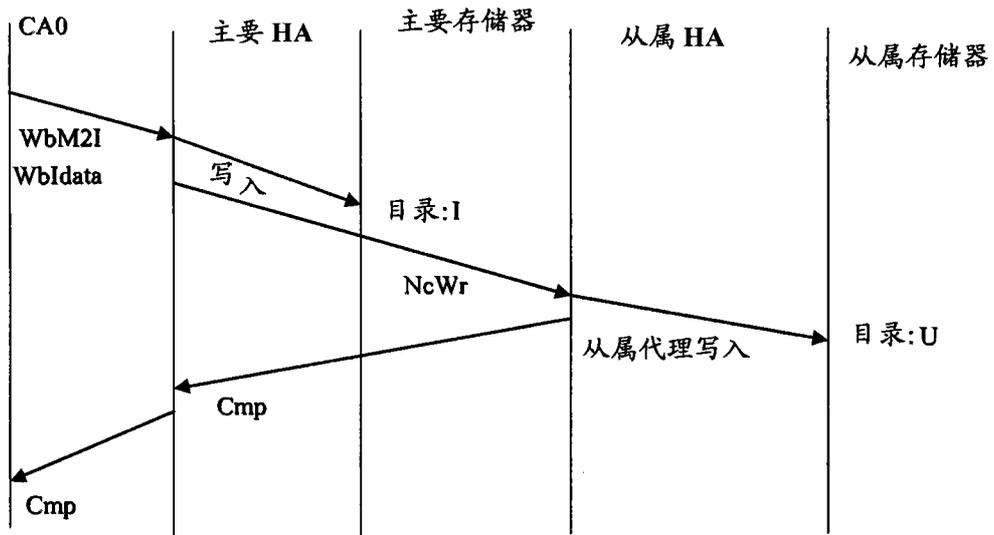


图 3

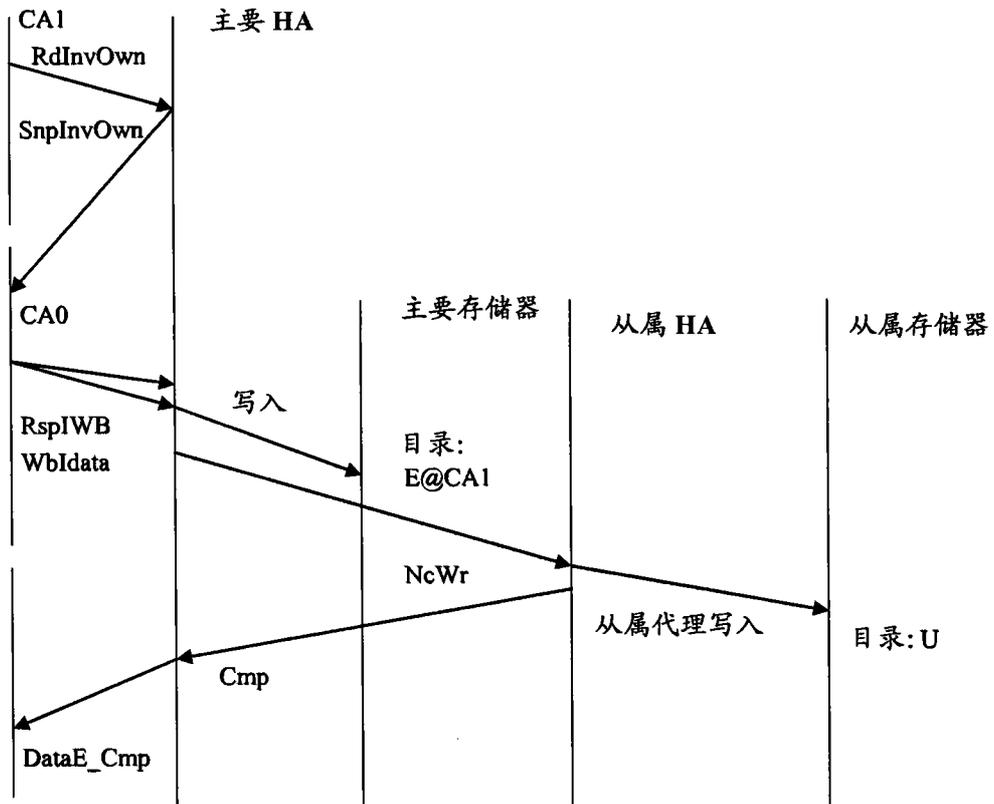


图 4

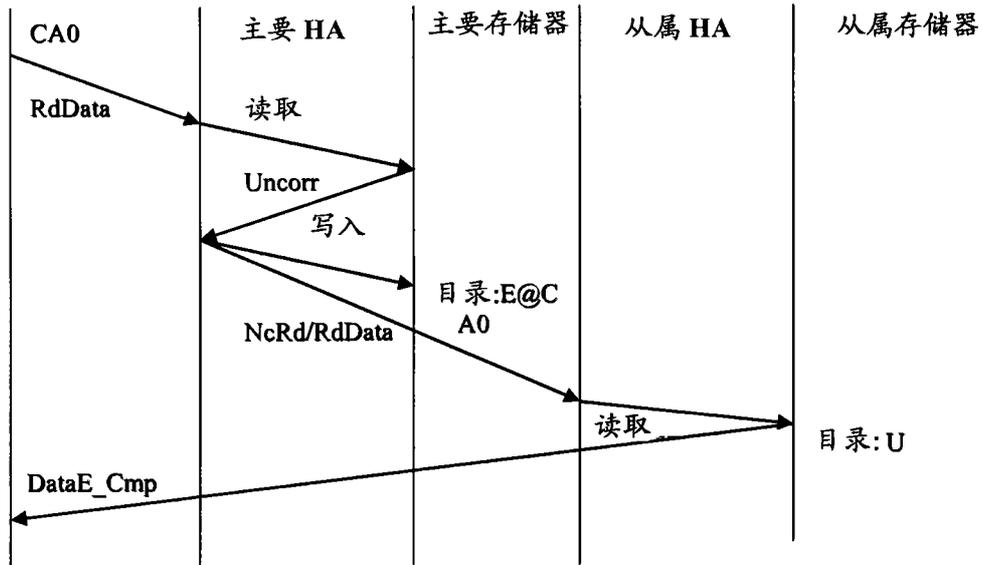


图 5

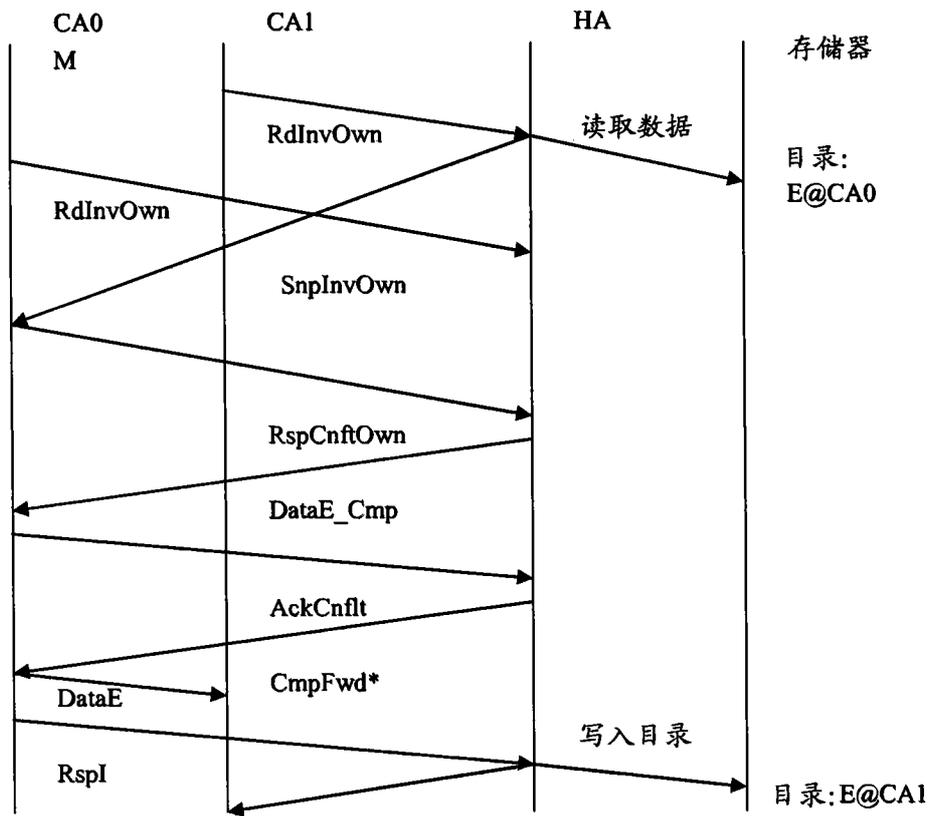


图 6

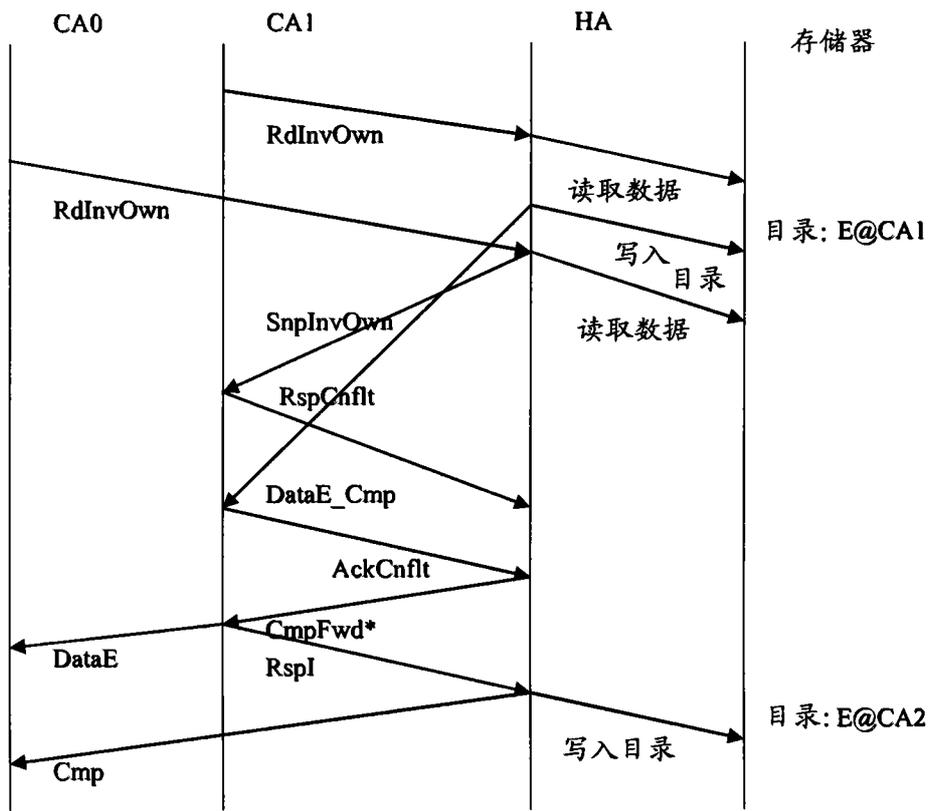


图 7

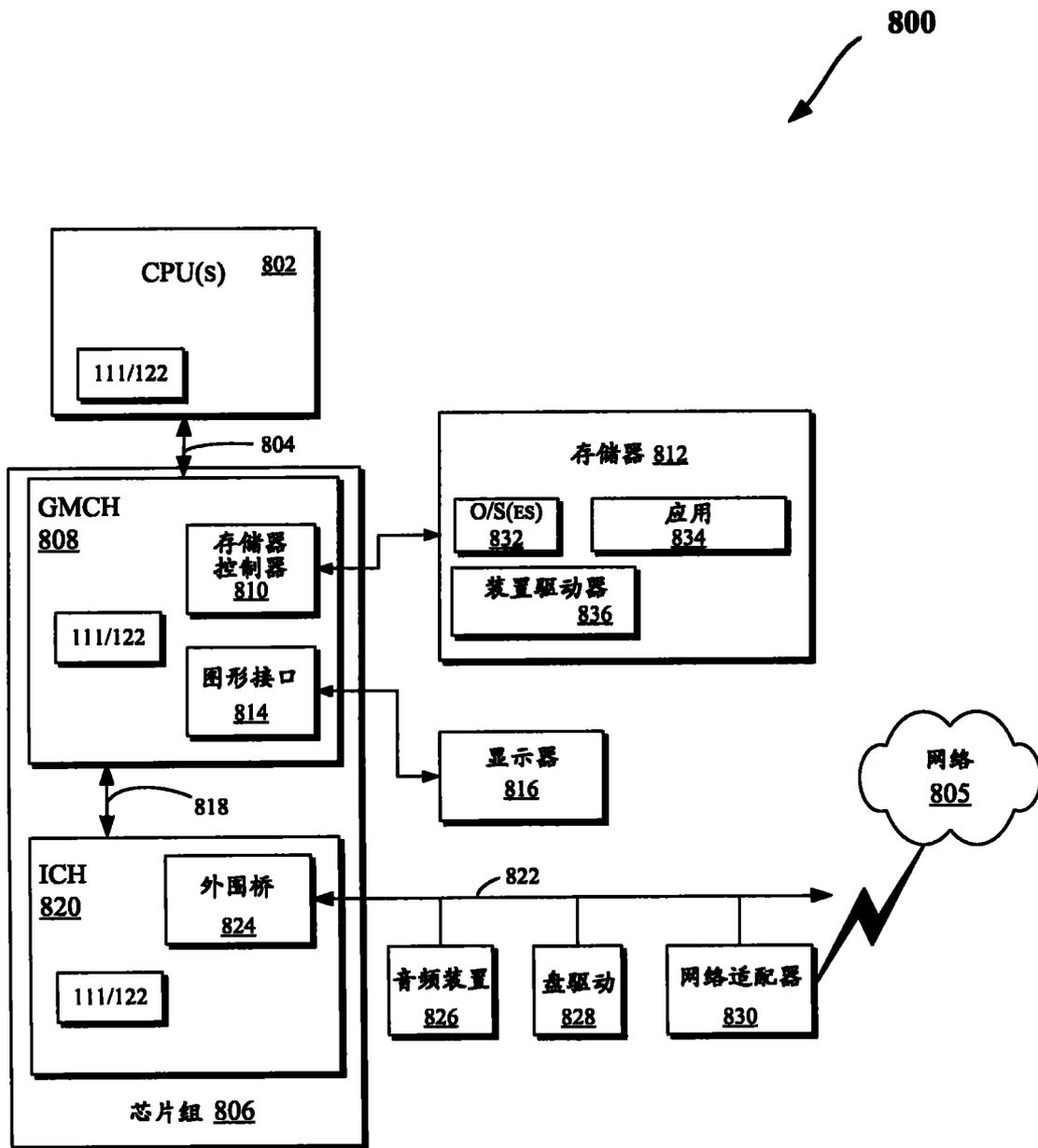


图 8

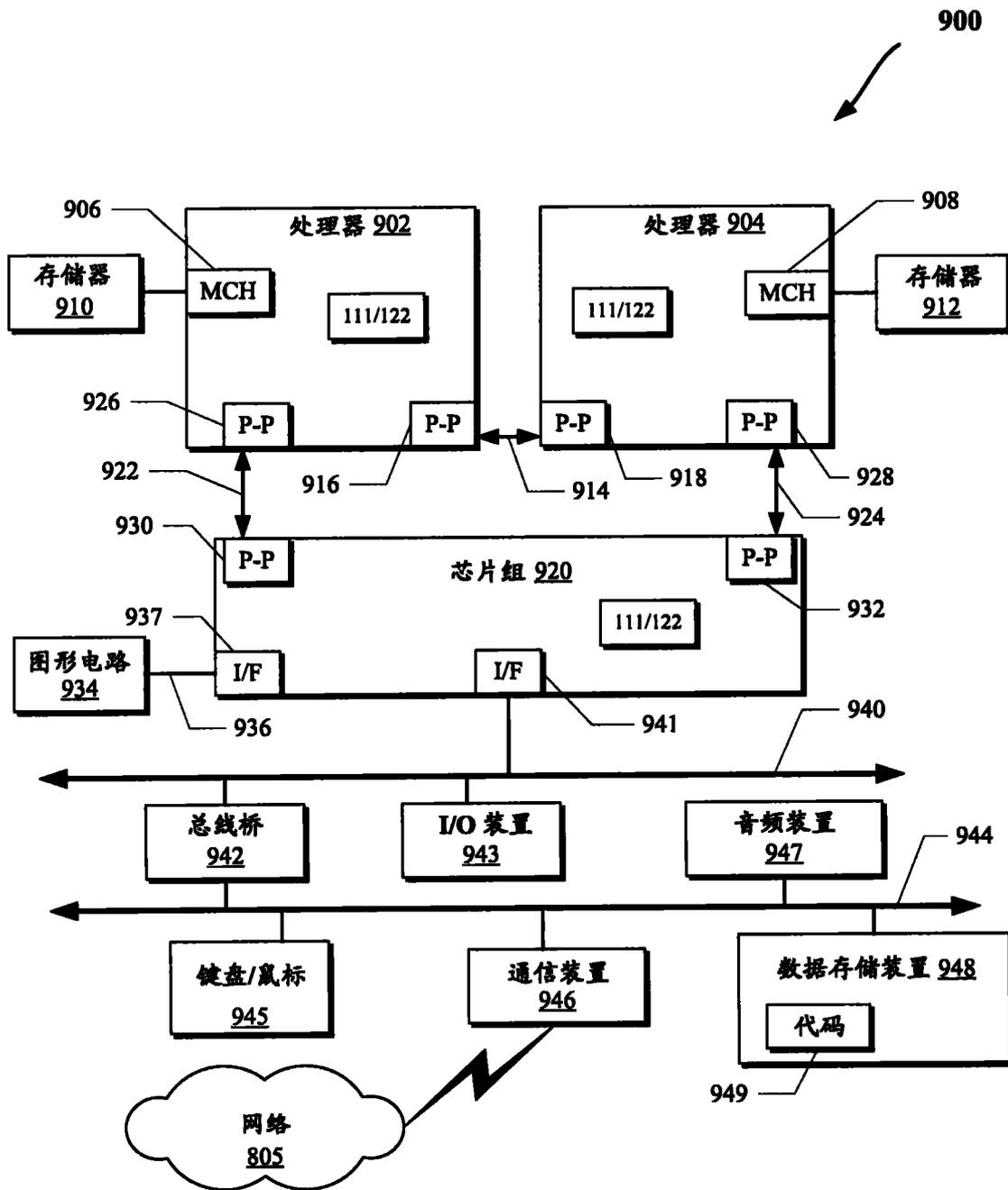


图 9