[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
G06F 3/06

# [12] 发明专利申请公开说明书

[21] 申请号 02823207.0

[43] 公开日 2005 年 3 月 9 日

[11] 公开号 CN 1592882A

[22] 申请日 2002.12.17 [21] 申请号 02823207.0
[30] 优先权
[32] 2001.12.21 [33] US [31] 10/028,065
[86] 国际申请 PCT/US2002/040357 2002.12.17
[87] 国际公布 WO2003/060729 英 2003.7.24
[85] 进入国家阶段日期 2004.5.21
[71] 申请人 英特尔公司
地址 美国加利福尼亚州
[72] 发明人 J·卡瓦罗 S·伊普利托
[74] 专利代理机构 上海专利商标事务所
代理人 谢喜堂

权利要求书 3 页 说明书 6 页 附图 5 页

[54] 发明名称 顺序数据传输检测

[57] 摘要

一种检测顺序数据传输请求的方法，包括检测第一数据传输请求是否跨越边界地址，且如果是，则确定第一数据传输请求是否可指示为可与随后的数据传输请求组合。 该方法还可包括确定先前的数据传输请求是否可指示为可组合的请求，且如果已指示为可组合的，则确定新数据传输请求是编址为与先前的数据传输请求相邻。

ISSN1008-4274

1．一种检测顺序数据传输请求的方法，其特征在于，它包括：

确定第一数据传输请求是否跨越边界地址，如果是，则：

确定是否可指示所述第一数据传输请求为可与随后的数据传输请求组合的。

2．如权利要求1所述的方法，其特征在于，它还包括：

确定先前的数据传输请求是否已被指示为可组合的，如果已指示为可组合的，则：

确定新的数据传输请求被定址到为与所述先前的数据传输请求相邻。

3．如权利要求2所述的方法，其特征在于，所述确定新的数据传输请求被定址为相邻包括：

确定所述新数据传输请求如同先前数据请求被定址到特定的最小块数之内。

4．如权利要求2所述的方法，其特征在于，所述特定边界地址跨越的最小数目在指示数据传输请求可组合之前确定。

5．如权利要求2所述的方法，其特征在于，它还包括：

定义边界块长度；以及

在指示第一数据传输请求可组合之前，确定第一数据传输请求跨越等于边界块长度的倍数的地址。

6．如权利要求5所述的方法，其特征在于，它还包括：

将第一跟踪地址设定为等于边界长度的倍数；

确定第二数据传输请求跨越第一跟踪地址；以及

指示第二数据传输请求可与随后的数据传输请求组合。

7．如权利要求5所述的方法，其特征在于，所述边界块长度包括一个为2的幂的数，其中，所述确定第一数据传输请求是否跨越边界地址包括：

确定边界长度的最高有效位是否等于第一数据传输请求地址的最高有效位。

8．如权利要求6所述的方法，其特征在于，它还包括：

跟踪至少两个单独的顺序流用于顺序处理。

9．如权利要求8所述的方法，其特征在于，所述跟踪进一步包括：

为每一跟踪的顺序流储存跟踪地址和相应的跟踪地址计数器值。

10．如权利要求9所述的方法，其特征在于，它还包括：

对每一确定与先前数据传输请求相邻的数据传输请求，将其跟踪地址计数器加1；以及

当跟踪地址计数器中所对应的一个大于一特定的最大值时，指示被跟踪的顺序流之一可以被释放为组合的 I/O 传输。

11．如权利要求 10 所述的方法，其特征在于，它还包括：

为每一确定不与先前数据传输请求相邻的数据传输，将其跟踪地址计数器减一。

12．一种制品，包括存储机器可执行指令的机器可读媒质，机器可执行指令用于检测顺序的数据传输请求，所述指令使机器：

确定第一数据传输请求是否跨越边界地址，并且，如果是，则：

确定第一数据传输请求是否可被指示为可与随后的数据传输请求组合的。

13．如权利要求 12 所述的制品，其特征在于，它进一步包括使机器进行以下步骤的指令：

确定先前的数据传输请求是否已被指示为可组合，如果已被指示为可组合的，则：

确定新的数据传输请求是否被定址为与先前数据传输请求相邻。

14．如权利要求 13 所述的制品，其特征在于，所述确定新数据传输请求被定址为相邻包括：

确定所述新数据传输请求如同先前的数据传输请求被定址在特定最小块数的范围之内。

15．如权利要求 13 所述的制品，其特征在于，所述特定边界地址跨越的最小数目在指示数据传输请求可组合之前确定。

16．如权利要求 13 所述的制品，其特征在于，它还包括指令使机器：

定义边界块长度；以及

在指示第一数据传输请求可组合之前，确定第一数据传输请求跨越等于边界块长度的倍数的地址。

17．如权利要求 16 所述的制品，其特征在于，它还包括指令使机器：

将第一跟踪地址设定为等于边界块长度；

确定第二数据传输请求跨越第一跟踪地址；以及

指示第二数据传输请求可与随后的数据传输请求组合。

18．如权利要求 16 所述的制品，其特征在于，所述边界长度包括一个为 2 的幂的数，其中，确定第一数据传输请求是否跨越边界地址包括：

确定所述边界块长度的最高有效位是否等于所述第一数据传输请求地址的最高有效位。

19．如权利要求 16 所述的制品，其特征在于，至少两个单独的顺序流被跟踪用于顺序处理，其中，所述物品进一步包括指令使机器：

为每一被跟踪的顺序流储存跟踪地址和相应的跟踪地址计数器值。

20．如权利要求19所述的制品，其特征在于，它还包括指令使机器：

对每一确定与先前数据传输请求相邻的数据传输请求，将其跟踪地址计数器之一加1；以及

当所对应的跟踪地址计数器之一大于特定最大值时，指示所跟踪的一个顺序流可被释放为组合I/O传输。

21．如权利要求20所述的制品，其特征在于，它还包括指令使机器：

对每一确定不与先前数据传输请求相邻的数据传输请求，将其跟踪地址计数器减1。

# 顺序数据传输检测

本发明涉及顺序数据传输的检测。

计算机处理系统通常连接至一个或多个输入/输出（“I/O”）装置。通常，处理器以一列数据传输的形式向I/O装置发送并从其接收数据项目。顺序数据流是指两个或多个被定址到I/O装置上的相邻位置的数据传输请求，通常是以升序排列。处理数据传输的效率影响处理系统的整体性能。

## 附图描述

图1是可实现顺序流检测进程的第一实施例计算机硬件的框图。

图2A－2C是表示顺序数据流检测进程的流程图。

图3是可实现顺序数据流检测进程的第二实施例的计算机硬件的框图。

## 说明

参考图1，示例性计算机处理系统100包括具有处理器104和用于储存机器可执行指令108的存储器106的个人计算机102。系统100还包括I/O控制器110，通过I/O总线111连接至I/O装置，如外置存储器112和盘D1－D5。

数据传输请求（以下称为“REQ”）通常由计算机102在例如执行程序时产生。如果需要读取或写入的数据被定址到I/O装置，REQ就被发送至I/O控制器110（或由处理器104执行的I/O子进程）。REQ由I/O处理器110处理作为向合适的I/O装置的I/O传输。每一I/O传输可包括处理单个数据传输请求，或者，在特定情况下I/O传输可包括共同处理多个REQ，下文将会解释。

每一由I/O处理器110执行的I/O传输可要求数个操作。例如，这类操作包括为I/O总线111判优以及在I/O总线111上发送命令或地址。其它操作可包括将I/O总线111上的地址发送至I/O装置、根据传送的数据计算奇偶校验、传送奇偶校验信息以及接收表示数据被发送到I/O装置或被其接收的确认。

虽然数据的传送代表着整个I/O传输所需要的时间的主要部分，I/O传输所涉及的许多时间是由建立事务的系统开销所消耗的。如果检测到顺序数据流，个别的REQ可处理为组合I/O传输。因此，减少了处理所有个别REQ所需要的时间，并且也减少了完成多个I/O传输所用的处理器执行周期数。

参考图2A－2C，显示了用于检测可组合成到I/O装置的输入/输出（I/O）

传输中的顺序数据传输请求（对相邻地址的读写）的进程 10。

以下显示了示例性顺序数据流“S1”。该例顺序流包括三个独立的 READ(读)命令：第一 REQ（Read1），指定了起始地址 10 和块长度 5；第二 REQ（Read 2），指定了起始地址 15 和块长度 20；第三 REQ（Read 3），指定了地址 35 和块长度 10。

S1：READ（10，5）；块 10－14

READ（15，20）；块 15－34

READ（35，10）；块 35－44

顺序流 S1 可以作为组合 I/O 传输被处理，组合 I/O 传输即：在地址 10 开始的块长为 35 的 I/O READ，如下所示。

I/O READ（10，35）；组合 I/O 传输，读地址 10－44。

仅当进程 10 确定（14）有至少一个“已知顺序流”（即，已检测到并被指示为已知顺序流的先前的顺序数据流），或者当进程 10 确定（16）一个新 REQ 跨越了预定义的“块边界地址”时，进程 10 才分析新 REQ。进程 10 包括子进程 50，当执行该子进程 50 时，可以检测并指示“已知顺序流”。当进程 10 确定（16）包括在一个新 REQ 的地址范围内的块地址包括了块边界地址时，就执行子进程 50，下文将会解释。

进程 10 包括对进程 10 期间中所使用的变量进行初始化（11）。进程 10 将全局数组结构初始化（11），全局数组结构包括至少一个位置跟踪器地址和相应的位置跟踪器计数器。进程 10 中使用的位置跟踪器和相应跟踪器计数器的总数可多于一个，因此进程 10 将总计位置跟踪器变量初始化等于“I”（11）（I 是进程 10 中使用的位置跟踪器的地址数）。进程 10 也初始化（11）块边界长度以及可编程变量“Max Counter(最大计数)”（由进程 10 的程序员或用户设定）。Max Counter(最大计数)为每一跟踪器计数器值定义了所检测的顺序流可达到的上限。定义 Max Counter 是必须的，因为对于每一检测到的顺序“命中”，每一跟踪器计数器递增，然后对于每一顺序“遗漏”则减 1，下文将会解释。已检测顺序流继续被跟踪，直到相应的跟踪器计数器值递减到一个相对低的数字为止。因此，例如，设定 Max Counter 等于 100 次顺序命中，即为每一跟踪器计数器值定义了上限，并且减少了对不再具有顺序命中的顺序流将跟踪器计数器递减所需要的时间。

仍参考图 2A，进程 10 接收（12）一个新 REQ，确定（14）是否指示了任何已知顺序流，如果指示了已知顺序流，则跳（20）至顺序流子进程 20（见图 2B）。如果没有指示已知顺序流，进程 10 就确定（16）该新 REQ 是否跨越块边界地址。如果新 REQ 跨越了块边界地址，则进程 10 就跳（50）至块边界子进程 50（见图 2C）。如果该新 REQ 没有跨越块边界地址，则进程 10 退出（18），并且该新 REQ 被正常处理，即，该新 REQ 不标记为顺序处理。

已知顺序流标志在子进程 50 的执行过程中被置位（以及清零）。因此，子进程 20 仅在子进程 50 至少执行了一次并且已对已知顺序流标志置位后才执行。因此，子进程 50 会在子进程 20 之前描述。

子进程 50（见图 2C）包括对于各位置跟踪器（I）重复 I 次的循环（52）。每次通过循环（52），子进程 50 就确定（54）新 REQ 是否包括“跨越”位置跟踪器地址（I）的块。更详细地说，每一位置跟踪器地址（I）最初被设定为块边界长度的偶数倍。所述确定（54）包括确定新 REQ 地址范围内的块是否也包括位置跟踪器（I）的块边界地址。例如，READ（80，22）包括块 80 至块 101 的地址范围，因此跨越了位置跟踪器地址（I）100。如果子进程 50 确定（54）该新 REQ 跨越位置跟踪器（I）地址，子进程 50 就指示（60）一个顺序命中，并将位置跟踪器计数器（I）加 1（62）。子进程 50 然后确定（63）位置跟踪器计数器（I）是否大于 Min Seq Count(最小顺序计数)。Min Seq Count(最小顺序计数)是一个可编程变量，用于定义在指示(68)已知顺序流之前必须检测的最小的顺序命中数目。如果子进程 50 确定（63）位置跟踪器计数器（I）大于 Min Seq Count(最小顺序计数)，循环（52）就将位置跟踪器地址（I）设定（65）为等于新 REQ 寻址的最后那个块加 1（以指向顺序数据流的下一相邻地址），并且标记（66）用于顺序处理的新 REQ，指示（68）已知顺序流正在被跟踪，并返回循环（52）的开始。如果子进程 50 确定位置跟踪器计数器（I）不比 Min Seq Count(最小顺序计数)大，则子进程 50 将位置跟踪器（I）的地址设定（64）为等于下一边界地址（位置跟踪器地址（I）加到块边界长度），并返回以重复循环（52）。如果在循环（52）中子进程 50 确定（54）新 REQ 地址未跨越位置跟踪器（I）地址，则进程 50 将位置跟踪器计数器（I）减 1（56）并返回至循环（52）的开始。

循环（52）完成后，子进程 50 确定（70）在循环（52）过程中是否指示了 Seq Hit(顺序命中)（通过动作（60））。如果指示过 Seq Hit(顺序命中)，则子进程 50 退出（72）。如果子进程 50 确定（70）未指示过 Seq Hit(顺序命中)，则子进程 50 确定（74）小于 Min Seq Count(最小顺序计数)的最低的位置跟踪器计数器（I）值，并将相应位置跟踪器地址（I）设定（78）为等于新 REQ 跨越的边界地址加上边界地址长度。子进程 50 退出（72）。

子进程 20（见图 2B）包括对每一位置跟踪器（I）重复 I 次的循环（21）。子进程 20 采用可编程变量“N”来确定（23）新 REQ 地址和位置跟踪器地址之间的最大间距，所述地址仍被标记为顺序流的一部分。更详细地来说，变量 N 允许一个轻微“出序”的新 REQ 地址变成顺序流的一部分。例如，等于 10 的变量 N 在与当前等于 60 的位置跟踪器地址比较时就允许将出序地址 70 标记为进行顺序处理。

每次通过循环（21），子进程 20 确定（21）位置跟踪器（I）是否大于

Min Seq Count(最小顺序计数)，如果的确确定位置跟踪器（I）大于 Min Seq Count(最小顺序计数)，子进程 20 就确定（23）新 REQ 地址是否处于位置跟踪器地址（I）的 N 个块的范围之内，如果是，则进程 10 标记（24）该新 REQ 成顺序处理(即，作为顺序流的一部分)。当位置跟踪器计数器小于等于 Max Counter(最大计数)时，子进程 20 就将位置跟踪器计数器（I）加 1（26）。子进程 20 将位置跟踪器地址（I）设定（28）为等于新 REQ 寻址的最后那个块加 1，即，设定位置跟踪器地址（I）指向位置跟踪器地址（I）所跟踪的顺序流中下一顺序地址。子进程 20 返回至循环（21）的开始，并重复确定（22）。如果子进程 20 确定位置跟踪器计数器（I）不比 Min Seq Count(最小顺序计数)大，或者子进程 20 确定（23）新 REQ 地址不处于位置跟踪器地址（I）的 N 块范围之内，则子进程 20 将位置跟踪器计数器（I）减 1（34）并且返回循环（21）的开始。

在完成循环（21）之后，子进程 20 确定（36）新 REQ 是否已标记为用于顺序处理（动作 24），如果新 REQ 已标记为用于顺序处理，则进程 20 退出（40）。如果新 REQ 未标记为用于顺序处理，进程 20 就为已知顺序流更新（38）标志，即，确定位置跟踪器计数器（I）中是否有任何大于指定最小数的值，例如，大于 Min Seq Count(最小顺序计数)变量。在更新了（38）已知顺序流标志之后，子进程 20 跳（39）至子进程 50。

通常的情况是，计算机处理器所产生的大多数 REQ 是随机 REQ（被定址到非顺序地址的 REQ）。在这种情况下，执行进程 10 来检测顺序流有效地减少了用于分析新 REQ 的处理周期数，因为仅当进程 10 确定（14）已指示了已知顺序流时，或者当进程 10 确定（16）新 REQ 跨越块边界地址时，进程 10 才分析新 REQ。否则，进程 10 退出(18)，因此大多数新 REQ 作为个别 I/O 传输来处理，而无需执行任何顺序流检测。

在进程 10 的一个实施例中，定义了一个相对较大的块边界长度，以减少进程 10 所分析的新 REQ 数，因此减少了用于分析新 REQ 的处理器执行时间。更详细地来说，通过定义相对较大的块边界长度，随机 REQ 不太有可能跨越边界块地址，因为非边界地址比边界地址多得多。

在进程 10 的更进一步的实施例中，包括定义块边界长度为 2 的幂，使得块边界跨越确定（16）（或确定（50））可以通过对边界地址和新 REQ 地址的最高有效位执行逻辑位比较来完成。更详细地来说，考虑以下示例，其中块边界长度和边界地址大小以十六进制数来表示：采用块边界长度为 100，以及初始块边界地址 100，接收到新 REQ“READ（FF，2）”。READ（FF，2）覆盖了从 0FF 到 101 的地址范围，即，具有起始块 0FF 以及终止地址块 101。进程 10 将起始地址 0FF 的最高有效位（MSB）与边界地址 100 的 MSB 进行比较（如采用“AND(与)”函数）。在这一情况下，第一比较结果是“不等于”。

下一步，进程 10 将终止地址 101 的 MSB 与边界地址 100 的 MSB 进行比较。在这一情况下，第二比较结果是“等于”。由于第一比较结果是等于并且第二比较结果是不等于，保证了地址 100 包含在 READ（FF，2）之内，并且该新 REQ 跨越了边界地址 100。

测试边界地址跨越的常规技术需要执行至少一次除法操作。然而，每一除法操作的执行需要多个处理器周期来完成。例如，采用除操作来确定先前描述的 REQ“READ（FF，2）的边界地址跨越，将如下进行：READ 起始地址 0FF 除以边界地址 100（0FF/100），第一次除法结果为 0。下一步，READ 终止地址 101 除以边界地址 100（101/100），第二次除法结果是 1。由于第一除结果（0）不等于第二除结果（1），因此边界地址被该新 REQ 跨越，然而，这一测试边界跨越的技术需要执行至少两个除操作。

通过比较，如上所述，进程 10 通过执行逻辑位比较（如“AND(与)”函数，或“EQUAL”/“NOT EQUAL(等于/不等于)”函数）来确定边界跨越，比传统执行除法函数的方法能够更简单并且更快地执行。

可能会有这一情况，随机 REQ 被产生并被与顺序 REQ 内部混合。如果计算机处理器从第一个程序的执行切换到第二个程序的执行时可能会产生这一情况。同时，可能以内部混合方式产生多个顺序流。在一个实施例中，进程 10 采用多个位置跟踪器及相关的位置跟踪器计数器来检测多个顺序数据流。

参考图 3，计算机 102 所产生的 REQ 可以被定址为“逻辑”地址，所述逻辑地址然后由 I/O 控制器 110 映射到一个或多个输入/输出装置 D1－D5 中的物理地址或地址范围。将逻辑地址空间映射到物理地址空间和物理装置有时候采用“RAID”映射进程来实现，“RAID”是指“廉价磁盘冗余阵列”，即，根据特定的 RAID 进程将一个逻辑地址空间映射到多个物理装置。有多个可用的不同 RAID 映射进程。

图 3 显示了一个示例性 RAID 映射进程，将一个数据“条”132 或 134 分别分区为多个数据“条”132a－132e 和 134a－134e。在该示例中，每个数据条 132a－132e 以及 134a－134e 被映射到不同的物理装置，分别为盘 D1－D5。而且，每一条可以包括多个相邻的数据子块，如，数据条 134a 包括数据子块 140－143，数据条 134b 包括数据子块 150－153。因此，如果系统 100 采用 RAID 映射进程，则到一数据条内的相邻子块的个别数据传输将作为组合 I/O 传输被处理。而且，到“跨越”数据条边界但仍处于单个数据条范围内的相邻子块的数据传输将作为组合 I/O 传输被处理（作为示例，子块 143 与同一数据条 134 内的子块 150 相邻）。作为结果，提高了 I/O 控制器 110 和系统 100 的性能，因为 I/O 控制器 110 需要执行更少的 I/O 传输。

在某些情况下，进程 10 包括确定组合 I/O 传输是否是“最优化尺寸”的。

“最优化尺寸”是指基于I/O控制器110所使用的地址映射模式的组合I/O传输的最大尺寸。例如，在某些RAID映射模式中，“最优化尺寸”组合的I/O读将允许组合仅定址单个数据条范围内的子块的数据传输。这一情况出现是由于跨越条边界的组合的读I/O传输对每一条要求单独的I/O传输，因此，组合跨越条边界的数据传输将不会必要地改进系统性能。作为另一示例，在某些RAID映射处理中，“最优化尺寸”的组合写I/O传输将组合定制整个数据条范围内的相邻子块的数据传输。这是由于特定的RAID映射处理包括基于包含在一条内的所有数据条的纠错码（ECC）计算。因此，通过组合跨越一条内多个条边界的任何相邻数据传输，使ECC计算变得更快，并且避免了在执行ECC计算时需要读那些组合的数据条。

进程10未局限于与图1和图4的硬件和软件来使用。它可以适用于任何计算或处理环境。进程10可以以硬件、软件或者两者的组合来实现。进程10可以用在可编程计算机或其它机器上执行的计算机程序来实现，每一机器包括处理器、处理器可读存储介质（包括易失性和非易失性存储器和/或存储组件）、至少一个输入装置以及一个或多个输出装置。每一这类程序可以用高级程序语言或面向对象编程语言来实现，来与计算机系统通信。然而，这些程序可以用汇编语言或者机器语言来实现。语言可以是已编译的或已解释的语言。

I/O控制器110可以作为计算机102的一部分被包括，即，作为同一集成处理器的部分或者作为同一计算机机箱的部分，并可以共享处理器104和存储器106。机器可执行指令也可以从ROM中执行，也可以从ROM移至存储器106。进程10可以在REQ发送至I/O控制器110之前由计算机102执行。

本发明未局限于以上描述的特定实施例。如，提到了向多个物理装置的逻辑到物理地址映射。然而，单个物理装置，如磁带驱动器，可以是逻辑到物理映射的，即，将磁带介质上的不同物理位置映射到逻辑地址空间。而且，即使没有使用地址映射模式也可以使用进程10。我们也提到了“读”和“写”命令作为数据传输请求的示例。然而，“读”和“写”命令可以更复杂，如“带有校验的写”，等等。我们提到了“顺序”的一个定义为包括直接在REQ引用的另一数据条之前或之后的数据条。然而，该“顺序”的定义可以延伸至包括数据尺寸和配置的更大范围。作为示例，我们也提到了边界块尺寸100。然而，可以使用更大（或小）的块尺寸。同时，在执行进程10的过程中可以使用单个位置跟踪器地址和关联位置跟踪器计数器。

其它此处未描述的实施例也包括在一下权利要求书的范围之内。

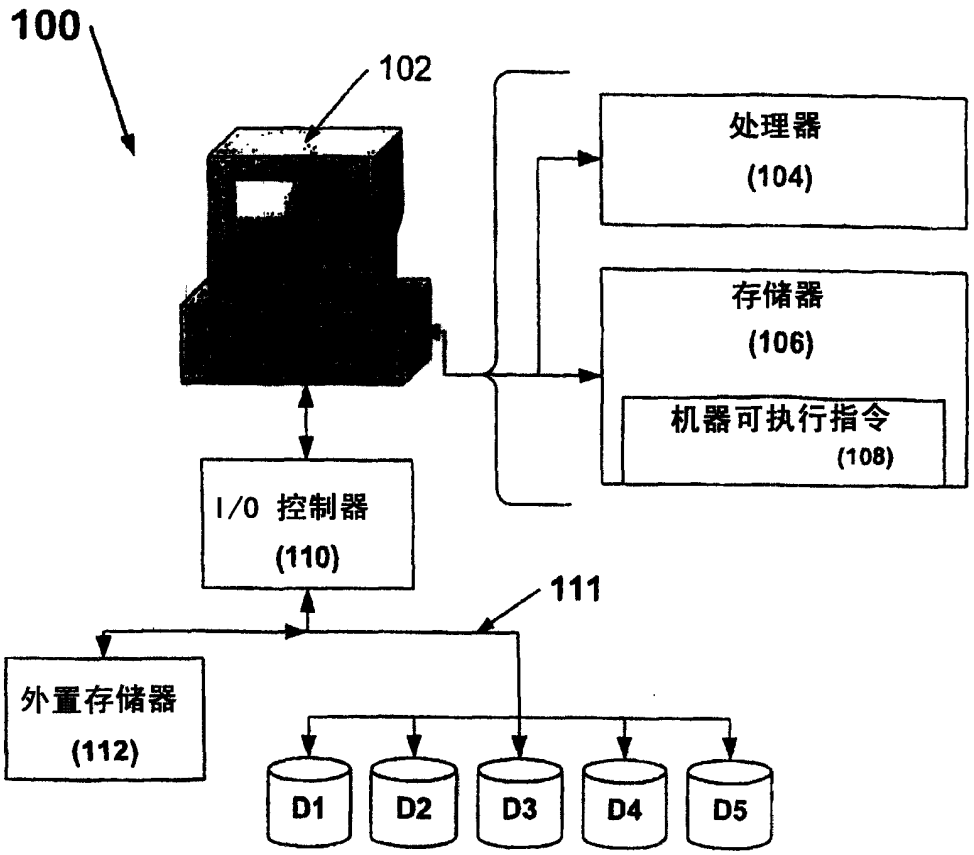
100
102
处理器
(104)
存储器
(106)
机器可执行指令
(108)
I/O 控制器
(110)
111
外置存储器
(112)
D1
D2
D3
D4
D5

图 1

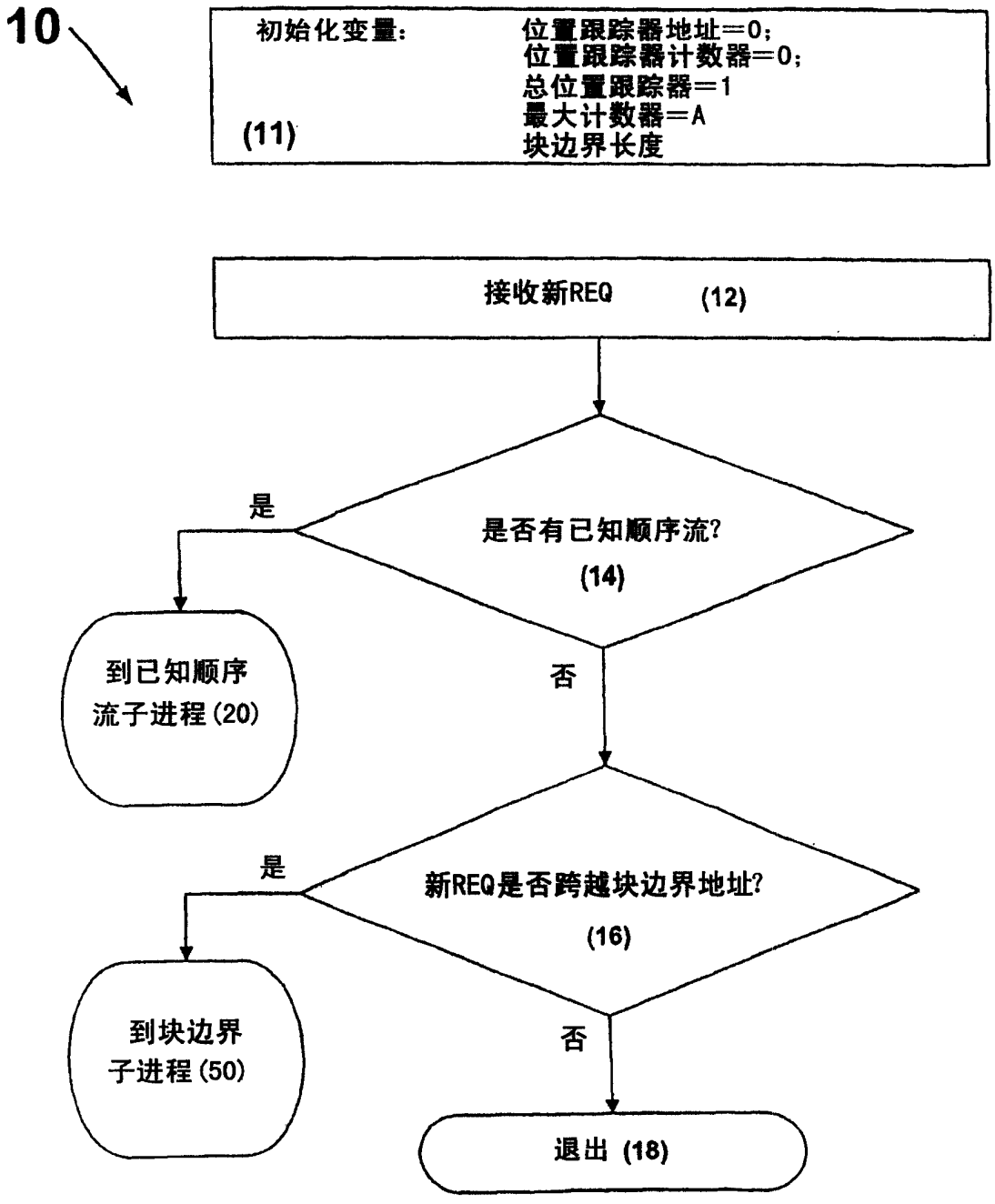
10
初始化变量：
位置跟踪器地址＝0；
位置跟踪器计数器＝0；
总位置跟踪器＝1
最大计数器＝A
块边界长度
(11)
接收新REQ (12)
是否有已知顺序流?
(14)
是
否
到已知顺序
流子进程(20)
新REQ是否跨越块边界地址?
(16)
是
否
到块边界
子进程(50)
退出 (18)

图 2A

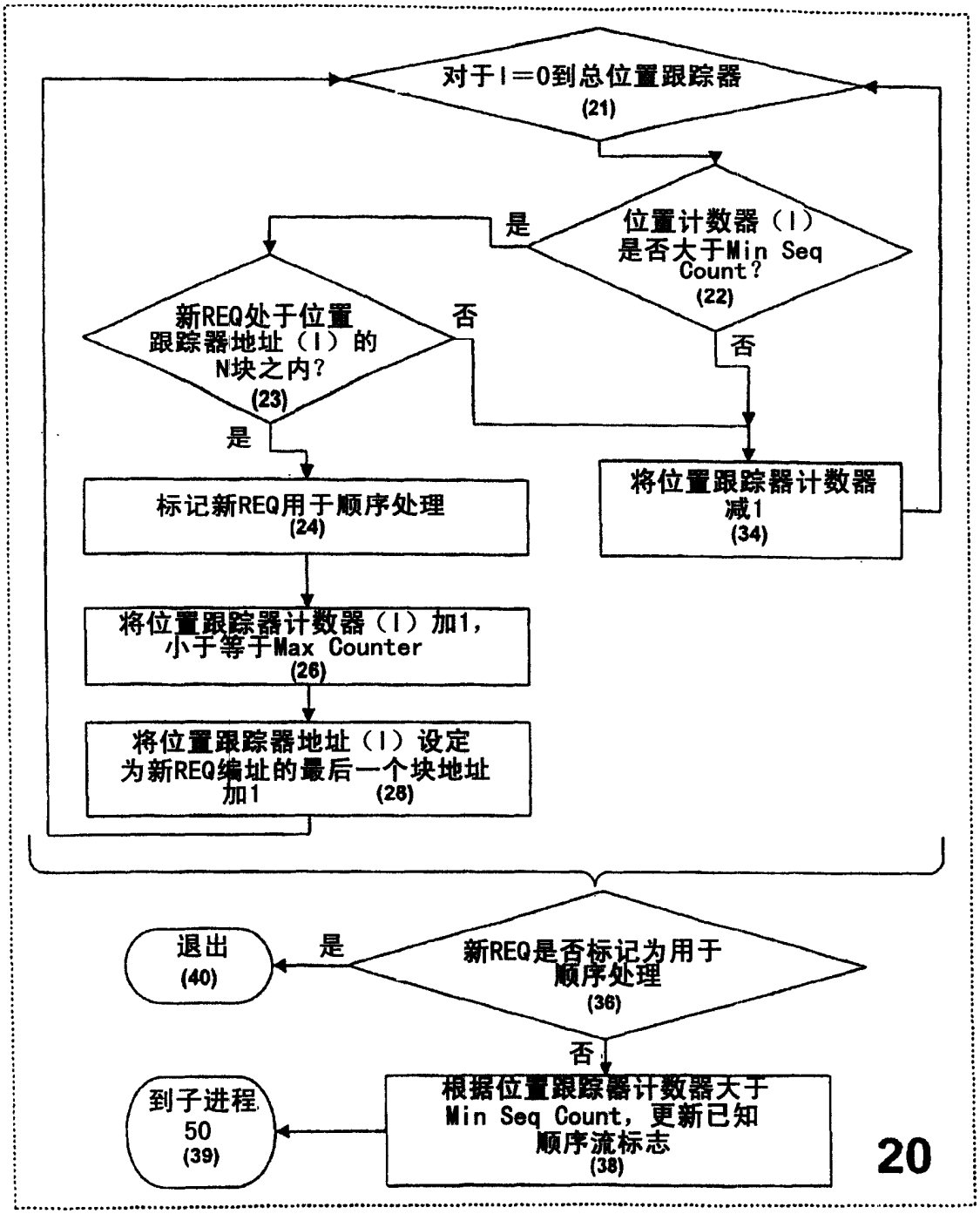
对于I＝0到总位置跟踪器
(21)
位置计数器（I）是否大于Min Seq Count?
(22)
是
否
新REQ处于位置跟踪器地址（I）的N块之内?
(23)
否
是
标记新REQ用于顺序处理
(24)
将位置跟踪器计数器减1
(34)
将位置跟踪器计数器（I）加1，小于等于Max Counter
(26)
将位置跟踪器地址（I）设定为新REQ编址的最后一个块地址加1
(28)
新REQ是否标记为用于顺序处理
(36)
是
退出
(40)
否
根据位置跟踪器计数器大于Min Seq Count，更新已知顺序流标志
(38)
到子进程50
(39)
20

图 2B

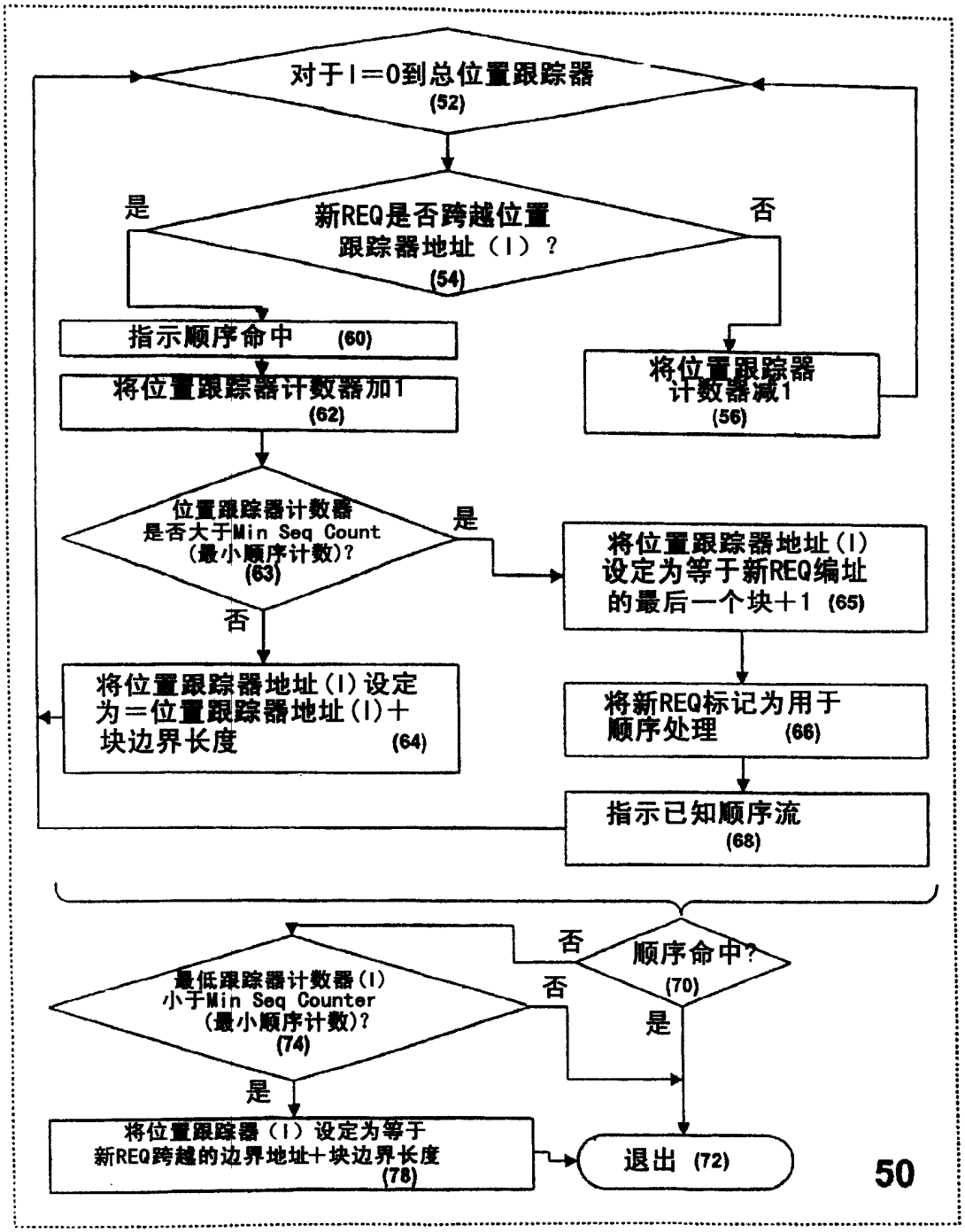
对于I＝0到总位置跟踪器
(52)
是
新REQ是否跨越位置
跟踪器地址（I）？
(54)
否
指示顺序命中 (60)
将位置跟踪器计数器加1
(62)
将位置跟踪器
计数器减1
(56)
位置跟踪器计数器
是否大于Min Seq Count
(最小顺序计数)?
(63)
是
否
将位置跟踪器地址(I)
设定为等于新REQ编址
的最后一个块＋1 (65)
将位置跟踪器地址(I)设定
为＝位置跟踪器地址(I)＋
块边界长度 (64)
将新REQ标记为用于
顺序处理 (66)
指示已知顺序流
(68)
否
顺序命中?
(70)
最低跟踪器计数器(I)
小于Min Seq Counter
(最小顺序计数)?
(74)
否
是
是
将位置跟踪器（I）设定为等于
新REQ跨越的边界地址＋块边界长度
(76)
退出 (72)
50

图 2C

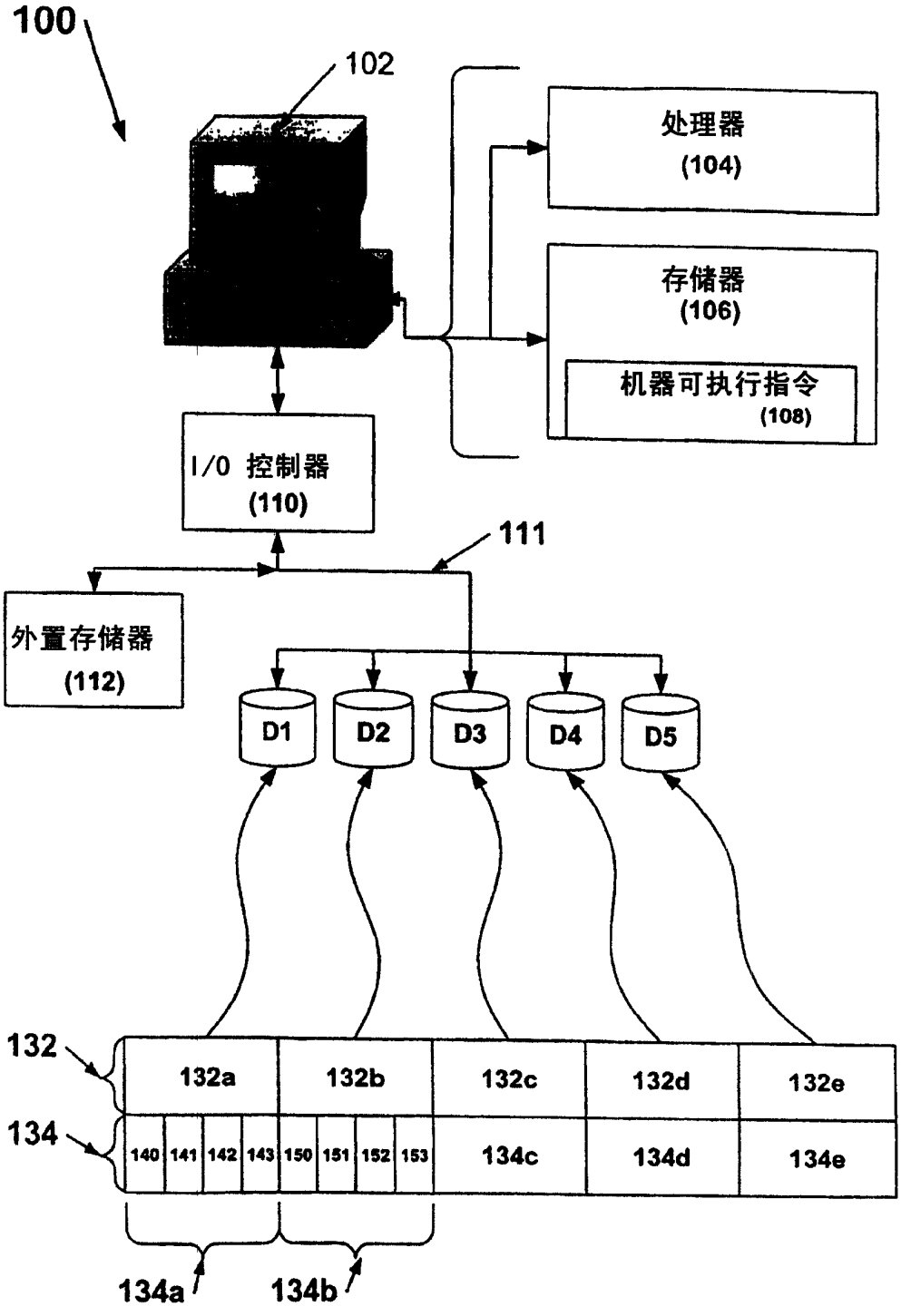
100
102
处理器
(104)
存储器
(106)
机器可执行指令
(108)
I/O 控制器
(110)
111
外置存储器
(112)
D1
D2
D3
D4
D5
132
132a
132b
132c
132d
132e
134
140
141
142
143
150
151
152
153
134c
134d
134e
134a
134b

图 3