[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
*G06F 13/12 (2006.01)*
*G06F 9/445 (2006.01)*

# [12] 发明专利申请公布说明书

[21] 申请号 200580017641.7

[43] 公开日 2007 年 5 月 9 日

[11] 公开号 CN 1961300A

[22] 申请日 2005.6.21
[21] 申请号 200580017641.7
[30] 优先权
[32] 2004. 6.30 [33] US [31] 10/882,073
[86] 国际申请 PCT/US2005/022226 2005.6.21
[87] 国际公布 WO2006/012196 英 2006.2.2
[85] 进入国家阶段日期 2006.11.30
[71] 申请人 英特尔公司
地址 美国加利福尼亚
[72] 发明人 什里坎特·M·沙阿
舍唐·J·拉瓦尔
[74] 专利代理机构 永新专利商标代理有限公司
代理人 王 英

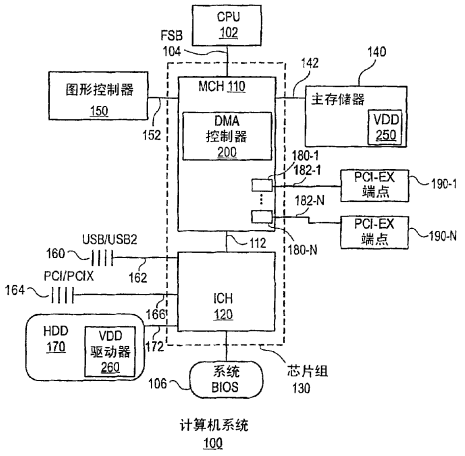
权利要求书 7 页 说明书 14 页 附图 6 页

[54] 发明名称

使用集成 DMA 引擎进行高性能易失性磁盘驱动器存储器访问的装置和方法

[57] 摘要

一种方法和装置，用于使用集成直接存储器访问(DMA)引擎进行高性能易失性磁盘驱动器(VDD)存储器访问。 在一个实施例中，所述方法包括检测对在易失性系统存储器之内实现的 VDD 存储器的数据访问请求。 一旦检测到数据访问请求，VDD 驱动器可以发出 DMA 数据请求以执行来自 VDD 的数据访问请求。 因此，在一个实施例中，向/从在易失性系统存储器的所分配部分之内实现的 VDD 存储器传输数据的工作被分担给 DMA 引擎，例如存储器控制器中心(MCH)中的集成 DMA 引擎。 还描述了其它实施例。



1．一种方法，包括：

检测对在易失性系统存储器中实现的易失性磁盘驱动器(VDD)存储器的数据访问请求；以及

发出直接存储器访问(DMA)数据请求以执行来自所述 VDD 存储器的所述数据访问请求。

2．如权利要求 1 所述的方法，其中，在检测所述数据访问请求之前，所述方法还包括：

由一进程请求对存储在所述 VDD 存储器之内的文件的读/写访问；

通过一操作系统(OS)检测对存储在所述 VDD 存储器之内的所述文件的进程读/写访问请求；以及

将所述读/写访问请求传递到 VDD 驱动器。

3．如权利要求 1 所述的方法，其中，在检测所述数据访问请求之前，所述方法还包括：

保留所述易失性系统存储器的一部分；以及

在初始化过程中，分配系统存储器的所述保留部分以用作所述 VDD 存储器。

4．如权利要求 1 所述的方法，其中，检测所述数据访问请求包括：

从操作系统接收由一进程发出的、对存储在所述 VDD 存储器中的文件的读/写访问请求。

5．如权利要求 1 所述的方法，其中，发出所述 DMA 数据请求包括：

根据所述数据访问请求生成至少一个 DMA 链描述符；以及

发出一 DMA 起动命令以执行对存储在所述 VDD 存储器中的文件的所述数据访问请求。

6．如权利要求 5 所述的方法，其中生成所述至少一个 DMA 链描述符包括：

识别所述 VDD 存储器之内的、所述数据访问请求所指向的文件；

计算所述易失性系统存储器之内的所述文件的物理开始地址；

根据存储在所述 VDD 存储器之内的所述文件之内的所述数据访问请求的偏移量计算 DMA 源地址；以及

根据所述数据访问请求计算 DMA 目的地址。

7．如权利要求 5 所述的方法，其中，发出所述 DMA 起动命令包括：

用本地系统存储器之内的所述至少一个DMA链描述符的地址更新 DMA 引擎后续地址寄存器；以及

设置 DMA 引擎起动 DMA 位，以使所述 DMA 引擎执行对存储在所述 VDD 存储器之内的所述文件的所述数据访问请求。

8．如权利要求 1 所述的方法，还包括：

通过DMA引擎取得位于包含在后续描述符地址寄存器中的地址的 DMA 链描述符；

根据所取得的 DMA 链描述符执行 DMA 传输，以便根据所述数据访问请求将DMA数据传输到存储在所述VDD存储器之内的文件；以及

如果完成所述 DMA 传输，则发出一中断给虚拟磁盘驱动器。

9．如权利要求 1 所述的方法，还包括：

如果所述 DMA 引擎完成所述数据访问请求的 DMA 传输，则从 DMA 引擎接收中断；以及

通知发出所述数据访问请求的应用程序已经完成了所述文件操

作。

10．如权利要求 1 所述的方法，其中，所述 VDD 存储器的尺寸大于四千兆字节。

11．一种制品，包括其上存储有指令的机器可读介质，所述指令可以用于编程系统以执行包括下列步骤的方法：

检测对在易失性系统存储器中实现的易失性磁盘驱动器(VDD)存储器的数据访问请求；以及

发出直接存储器访问(DMA)数据请求以执行来自所述 VDD 存储器的所述数据访问请求。

12．如权利要求 11 所述的制品，其中发出所述 DMA 数据请求包括：

根据所述数据访问请求生成至少一个 DMA 链描述符；以及

发出一 DMA 起动命令以执行对存储在所述 VDD 存储器中的文件的所述数据访问请求。

13．如权利要求 12 所述的制品，其中，生成所述至少一个 DMA 链描述符包括：

识别所述 VDD 存储器之内的、所述数据访问请求所指向的文件；

计算所述易失性系统存储器之内的所述文件的物理开始地址；

根据存储在所述 VDD 存储器之内的所述文件之内的所述数据访问请求的偏移量计算 DMA 源地址；以及

根据所述数据访问请求计算 DMA 目的地址。

14．如权利要求 12 所述的制品，其中，发出所述 DMA 起动命令包括：

用本地系统存储器之内的所述至少一个 DMA 链描述符的地址更新 DMA 引擎后续地址寄存器；以及

设置 DMA 引擎起动 DMA 位，以使所述 DMA 引擎执行对存储在所述 VDD 存储器之内的所述文件的所述数据访问请求。

15．如权利要求 11 所述的制品，还包括：

如果所述 DMA 引擎完成所述数据访问请求的 DMA 传输，则从 DMA 引擎接收中断；以及

通知发出所述数据访问请求的应用程序已经完成了所述文件操作。

16．一种方法，包括：

通过一操作系统(OS)检测对存储在易失性磁盘驱动器(VDD)存储器之内的文件的进程读/写访问请求，其中，所述易失性磁盘驱动器(VDD)存储器在易失性系统存储器之内实现；

将所述读/写访问请求传递到 VDD 驱动器；以及

通过所述 VDD 驱动器发出直接存储器访问(DMA)数据请求以从存储在所述 VDD 存储器中的所述文件执行所述读/写访问请求。

17．如权利要求 16 所述的方法，其中，发出所述 DMA 数据请求包括：

根据所述读/写访问请求生成至少一个 DMA 链描述符；以及

发出 DMA 起动命令以执行对存储在所述 VDD 存储器中的所述文件的所述读/写访问请求。

18．如权利要求 17 所述的方法，其中，生成所述至少一个 DMA 链描述符包括：

识别所述虚拟磁盘驱动器之内的、所述数据访问请求所指向的文件；

计算所述易失性系统存储器内文件的物理开始地址；

根据存储在所述 VDD 存储器之内的所述文件之内的所述读/写访问请求的偏移量计算 DMA 源地址；以及

根据所述读/写访问请求计算 DMA 目的地址。

19. 如权利要求 17 所述的方法，其中，生成所述至少一个 DMA 链描述符包括：

识别所述 VDD 存储器之内的、所述数据访问请求所指向的文件；以及

计算所述易失性系统存储器之内的所述文件的物理开始地址。

20. 如权利要求 16 所述的方法，还包括：

通过 DMA 引擎取得位于包含在后续描述符地址寄存器中的地址的 DMA 链描述符；

根据所取得的 DMA 链描述符执行 DMA 传输，以便根据所述读/写访问请求从存储在所述 VDD 存储器之内的文件传输 DMA 数据；以及

如果完成所述 DMA 传输，则发出一中断给所述虚拟磁盘驱动器。

21. 一种制品，包括其上存储有指令的机器可读介质，所述指令可以用于编程系统以执行包括下列步骤的方法：

通过一操作系统(OS)检测对存储在易失性磁盘驱动器(VDD)存储器之内的文件的进程读/写访问请求，其中，所述易失性磁盘驱动器(VDD)存储器在易失性系统存储器之内实现；

将所述读/写访问请求传递到 VDD 驱动器；以及

通过所述 VDD 驱动器发出直接存储器访问(DMA)数据请求以从存储在所述 VDD 存储器中的所述文件执行所述读/写访问请求。

22. 如权利要求 21 所述的制品，其中，发出所述 DMA 数据请求包括：

根据所述读/写访问请求生成至少一个 DMA 链描述符；以及

发出 DMA 起动命令以执行对存储在所述 VDD 存储器中的所述文件的所述读/写访问请求。

23．如权利要求 22 所述的制品，其中，生成所述至少一个 DMA 链描述符包括：

识别所述虚拟磁盘驱动器之内的、所述数据访问请求所指向的文件；

计算所述易失性系统存储器之内的所述文件的物理开始地址；

根据存储在所述 VDD 存储器之内的所述文件之内的所述读/写访问请求的偏移量计算 DMA 源地址；以及

根据所述读/写访问请求计算 DMA 目的地址。

24．如权利要求 22 所述的制品，其中，生成所述至少一个 DMA 链描述符包括：

识别所述 VDD 存储器之内的、所述数据访问请求所指向的文件；以及

计算所述易失性系统存储器之内的所述文件的物理开始地址。

25．如权利要求 21 所述的制品，还包括：

通过DMA引擎取得位于包含在后续描述符地址寄存器中的地址的 DMA 链描述符；

根据所取得的 DMA 链描述符执行 DMA 传输，以便根据所述读/写访问请求从存储在所述 VDD 存储器之内的所述文件传输 DMA 数据；以及

如果完成所述DMA传输，则发出一中断给所述虚拟磁盘驱动器。

26．一种系统，包括：

易失性系统存储器，其具有为提供易失性磁盘驱动器(VDD)存储器所分配的部分；

芯片组，其连接到所述易失性系统存储器，所述芯片组包括直接存储器访问(DMA)引擎以执行到所述易失性系统存储器的 DMA 传输；以及

非易失性系统存储器，其连接到所述芯片组，所述非易失性系统存储器包括 VDD 驱动器以检测对所述虚拟磁盘驱动器的数据访问请求以及向所述芯片组发出 DMA 请求，以使所述 DMA 引擎执行对所述 VDD 存储器的所述数据访问请求。

27. 如权利要求 26 所述的系统，其中，所述芯片组还包括具有集成 DMA 引擎的存储器控制器中心。

28. 如权利要求 26 所述的系统，其中，所述芯片组还包括：

输入/输出控制器中心。

29. 如权利要求 26 所述的系统，其中，所述芯片组还包括：

DMA 控制器，其包括至少一个后续描述符地址寄存器以存储由所述 VDD 驱动器编程的 DMA 链描述符的物理地址，以执行对所述 VDD 存储器的所述数据访问请求。

30. 如权利要求 29 所述的系统，其中，所述 DMA 控制器还包括：

通道控制寄存器，其包含一 DMA 起动位以指示所述 DMA 引擎装载所述 DMA 链描述符并执行对所述 VDD 存储器的所述数据访问请求。

# 使用集成 DMA 引擎进行高性能易失性磁盘驱动器存储器访问的装置和方法

## 技术领域

本发明的一个或多个实施例一般涉及集成电路和计算机系统设计领域。特别地，本发明的一个或多个实施例涉及使用集成直接存储器访问(DMA)引擎进行高性能易失性磁盘驱动器存储器访问的装置和方法。

## 背景技术

RAMDISK 是一种存储器管理技术，其中在程序控制下，将一部分易失性系统存储器用作大容量存储设备，这里称作“易失性磁盘驱动器(VDD)存储器”。对 RAMDISK 的操作系统(OS)文件操作比对传统硬盘的 OS 操作要快很多。因此，通过将最常用的程序和文件放置在为 RAMDISK 分配的系统存储器的一部分之内，RAMDISK 可以显著地提高系统性能。尽管一旦关闭系统，RAMDISK 就不会保留所存储的信息，但是 RAMDISK 存储器管理技术对于以常规 OS 文件格式存储的数据的临时存储和缓存是非常有用的。典型地，将 RAMDISK 用于 web/文件服务器以便缓存海量媒体存储池中的常用文件。

传统上，用于实现 RAMDISK 的驱动器完全以软件实现。因此，当程序或进程想要读/写 RAMDISK 中的文件时，将调用 RAMDISK 驱动器。这种驱动器首先识别所请求的文件是如何映射到保留给 RAMDISK 的系统存储器位置的。一旦识别出，驱动器将向 RAMDISK 的存储器传输来自请求进程的存储器的数据，或从 RAMDISK 的存储器向请求进程的存储器传输数据。通常，这是在 CPU 的帮助下执行的，其中驱动器指示 CPU 从系统存储器的 RAMDISK 部分请求数据。

然而，需要 CPU(中央处理单元)为 RAMDISK 驱动器既管理数据传输又管理文件管理功能。尽管 RAMDISK 存储器管理技术比使用传

统存储驱动器快许多，但是 RAMDISK 驱动器会利用大量 CPU 周期以在 RAMDISK 位置和应用程序位置之间传输数据。此外，为超出 4G 字节(GB)的存储器(32-位地址)实现 RAMDISK，需要 RAMDISK 驱动器使用 CPU 的页地址扩展(PAE)模式。然而，PAE 模式自身不仅对于 RAMDISK 驱动器，而且对于所有运行于系统之内的进程都会引入额外的性能损失。

## 附图说明

本发明的各种实施例将通过附图中的示例说明，而不是通过限制的方式说明，其中：

图 1 是说明根据一个实施例的计算机系统的框图，该计算机系统包含具有集成 DMA 引擎的 DMA 控制器以提供对虚拟磁盘驱动器(VDD)的高性能访问；

图 2 是根据一个实施例进一步说明图 1 的 DMA 控制器的框图；

图 3 是根据一个实施例说明 DMA 寄存器和链描述符映射的框图；

图 4 是根据一个实施例说明 DMA 描述符集合链表的框图，该链表用于使用 DMA 访问，对虚拟磁盘驱动器之内的数据块进行访问；

图 5 是根据一个实施例说明对 VDD 访问请求的检测以及使用图 1 的 DMA 控制器通过直接存储器访问的 VDD 访问请求的后续操作的框图；

图 6 是说明用于使用已公开的技术对设计进行仿真、模拟以及制造的各种设计表示或格式的框图。

## 具体实施方式

将描述使用集成直接存储器访问(DMA)引擎进行高性能易失性磁盘驱动(VDD)存储器访问的方法和装置。在一个实施例中，方法包括检测对易失性系统存储器中实现的 VDD 存储器的数据访问请求。一旦检测到数据访问请求，VDD 驱动器会发出 DMA 数据请求以执行来自 VDD 的数据访问请求。因此，在一个实施例中，将数据向在

易失性系统存储器的所分配的部分之内实现的 VDD 存储器传输以及从其传输数据的工作被分担给 DMA 引擎，例如存储器控制中心(MCH)之内的集成 DMA 引擎。

在以下描述中，提出了很多诸如逻辑实现、信号和总线的型号和名称、系统部件的类型和相互关系以及逻辑划分/集成选择的具体细节以便于更彻底的理解。但是，本领域的技术人员应当知道，本发明可以不用这些具体细节来实现。在其它实例中，没有详细显示控制结构以及门级电路以避免使本发明难以理解。使用所包含的描述，本领域的那些普通技术人员将能够在不需大量试验的情况下实现正确的逻辑电路。

在以下描述中，将某些术语用于描述本发明的特征。例如，术语“逻辑”表示用于实现一个或多个功能的硬件和/或软件。比如，“硬件”的例子包括但不限于集成电路、有限状态机乃至组合逻辑。集成电路可以采取诸如微处理器、专用集成电路、数字信号处理器、微控制器等等的处理器的形式。

“软件”的例子包括应用程序、小程序、例程乃至指令序列形式的可执行代码。在一个实施例中，一件产品包括机器或其上存储有指令的计算机可读介质，该指令对计算机(或其它电子设备)进行编程以执行根据一个实施例的处理。计算机或机器可读介质包括但不局限于：可编程电子电路、包含易失性存储器(例如，随机访问存储器，等)和/或非易失性存储器(例如，任何种类的只读存储器“ROM”、闪存)在内的半导体存储设备、软盘、光盘(例如，高密度盘或者数字视频光盘“DVD”)、硬盘、磁带等等。

系统

图 1 是说明根据一个实施例的包含直接存储器访问(DMA)控制器 200 以提供对 VDD 存储器 250 的高性能访问的计算机系统 100 的框图。典型地，计算机系统 100 包括用于在处理器(CPU)102 和芯片组 130 之间传输信息的处理器系统总线(前端总线(FSB))104。如这里所述，术语“芯片组”用于总体描述与 CPU 102 连接以执行所希望的系统功能的各种设备。

典型地，芯片组 130 可以包括连接到图形控制器 150 的存储器控制中心 110(MCH)。在一可选实施例中，图形控制器 150 集成到 MCH 中，因此在一个实施例中，MCH 110 用作集成图形 MCH(GMCH)。典型地，MCH 110 还通过系统存储器总线 142 连接到主存储器 140。在一个实施例中，主存储器 140 可以包括但不限于随机访问存储器(RAM)、动态 RAM(DRAM)、静态 RAM(SRAM)、同步 DRAM(SDRAM)、双数据率(DDR)SDRAM(DDR-SDRAM)、Rambus DRAM(RDRAM)或者任何能够支持高速临时数据存储的设备。

在一个实施例中，MCH 110 集成在 CPU 102 之内以使得能够在 CPU 102 和主存储器 140 之间进行直接连接。在一个实施例中，MCH 110 可以包括外设部件互连(PCI)PCI-Express(PCI-Ex)根端口 180(180-1，...180-N)，以便通过 PCI-Ex 连接 182(182-1，...，182-N)将 PCI-Ex 终端 190(190-1，...190-N)连接到 MCH 110。典型地，PCI-Ex 连接 182 可以提供点到点连接，比如由“PCI Express Base Specification 1.0a”(勘误 2003 年 10 月 7 日)定义，以允许外部终端设备 190(190-1，...，190-N)和 MCH 110 之间的双向通信。

作为进一步的说明，芯片组 130 包括输入输出(I/O)控制器中心(ICH)120。典型地，ICH 120 可以包括通用串行总线(USB)连接或互连 162 以将一个或多个 USB 槽 160 连接到 ICH 120。另外，串行高级技术附加装置(SATA)172 可以将硬盘驱动设备(HDD)170 连接到 ICH 120。在一个实施例中，ICH 120 可以包括传统的 PCI/PCIx 连接 166 以将一个或多个 PCI/PCIx 槽 164 连接到 ICH 120。在一个实施例中，系统 BIOS 106 初始化计算机系统 100。

再次参考图 1，在一个实施例中，计算机系统 100 包括 VDD 驱动器 260。在操作中，VDD 驱动器 260 负责预留和分配一部分易失性系统存储器 140，以用作 VDD 存储器 250。在一个实施例中，主存储器 140 的分配给 VDD 存储器 250 的部分是对于计算机系统 100 的核心操作系统(OS)隐蔽的非可交换存储器。如这里所述，VDD 存储器 250 或者可以称为 RAMDISK，它是一种存储器管理技术，用于在程序控制下分配易失性系统存储器的一部分以用作大容量存储设

备。

在一个实施例中，VDD 驱动器 260 利用 DMA 控制器 200 执行 VDD 存储器之内的文件操作。传统上，使用已编程的传输来执行由设备驱动器(例如与连接到芯片组(130)的外围设备相关的软件)进行的主存储器访问，其中 CPU 发出总线事务以为外围设备起动从存储器读数据的操作或向存储器写数据的操作。相反地，DMA 是由先进的体系结构提供的能力，它允许在不涉及 CPU 的情况下将数据在外围设备和主存储器之间直接传输。因此，系统的 CPU 免于参与数据传输，因此加快了整个计算机的操作。在一个实施例中，如图 2 所示，DMA 控制器 200 提供计算机系统 100 之内的 DMA 能力。

在一个实施例中，DMA 控制器包括集成 DMA 引擎 210，用于提供多个 DMA 通道 214(214-1，...214-N)。在一个实施例中，DMA 控制器 200 提供四个 DMA 通道，每个都可以独立地用于在系统存储器 140 之内或从系统存储器 140 向各种连接到芯片组 130 的外围设备传输数据。在一个实施例中，如以下进一步详细描述的，可以通过写入主存储器 140 的链描述符和存储器映射内部寄存器组的组合从 CPU 102 访问 DMA 通道编程接口，如图 3 和 4 所示。

在一个实施例中，DMA 通道 214 之间的仲裁以两阶段出现。典型地，每个 DMA 通道 214 具有独立的对于 DMA 控制器 200 内部的仲裁器(DMA 仲裁器 212)的总线请求/许可对。作为进一步说明，DMA 控制器 200 具有对于 MCH 110 中的主仲裁器(MCH 仲裁器 220)的单个请求/许可对。在一个实施例中，DMA 仲裁器 212 使用严格轮询策略以准许 DMA 通道和请求设备对主存储器 140 的访问。在一个实施例中，DMA 仲裁器 212 可以包括在任何给定时间对于一个 DMA 通道可选的高优先级指示。因此，一组竞争的 DMA 通道在正常操作过程中实现均衡的带宽性能。

再次参考图 1，如这里所述，每个存在于连接到芯片组 130 的总线上的外围设备或 I/O 卡在这里被称为“总线代理”。一般地将总线代理分成均衡代理和优先代理，从而当与均衡代理竞争总线占用权时，优先代理被给予占用权。需要这样的仲裁，因为通常不允许总线

代理同时驱动总线发出事务。与计算机系统 100 的总线代理相关的设备驱动器可以将事务请求 232 发出给 MCH。根据优先级策略，MCH 仲裁器 220 为 DMA 通道 214 仲裁以发出 DMA 传输请求，如图 2 所示。

如这里所述，术语“事务”被定义为与单个总线访问请求相关的总线活动。通常，事务可以从总线仲裁以及传送事务地址的信号的断言(assertion)开始。如 Intel(注册商标)体系结构(IA)规范所定义的，一个事务可以包括多个阶段，每个阶段使用特定的信号集合以传送特定类型的信息。多个阶段中至少包括仲裁阶段(针对总线占用权)、请求阶段、应答阶段以及数据传输阶段。

实现计算机系统(比如计算机系统 100)内的 DMA 访问需要将利用 DMA 访问的设备指定为总线主控。总线主控是微处理器或独立的 I/O 控制器(“设备驱动器”)中的程序，用于引导系统总线或输入输出(I/O)路径上的通信。操作中，总线主控设备驱动器请求 OS 分配一部分主存储器 140，其为被指定或使得能够进行 DMA 以发出 DMA 数据请求。

再次参考图 1，在一个实施例中，VDD 驱动器 260 控制一个或多个被给予总线主控状态的 DMA 通道，其使得 VDD 驱动器 260 能够向 DMA 控制器 200 发出 DMA 访问请求。在操作中，在 DMA 仲裁器 212 的仲裁以及随后 MCH 仲裁器 220 的仲裁之后，将授权 DMA 通道 214 访问以向 VDD 存储器 250 发出 DMA 访问请求。因此，在一个实施例中，当应用程序或进程请求执行文件操作时，比如对 VDD 存储器 250 中存储的文件的读/写访问，那么 OS 可以将该调用传递到 VDD 驱动器 260。在一个实施例中，为引导 DMA 引擎 210 执行所请求的文件操作，VDD 驱动器 260 编写链描述符，如参照图 3 所示。

在一个实施例中，DMA 控制器 200 对所有的每通道寄存器组使用存储器映射配置寄存器。在一个实施例中，与 DMA 控制器 200 相关的存储器映射寄存器空间由 32-位存储器-基址寄存器(BAR)来标识，它由例如 VDD 驱动器 260 使用以访问对编程 DMA 通道以起动 DMA 传输所需要的 DMA 配置寄存器 270。在一个实施例中，如图 3

所示，每个 DMA 通道具有 12 个 32-位存储器映射寄存器用于它的独立操作。

在一个实施例中，当正常操作过程中从本地存储器中取出一新的描述符时，DMA 寄存器 270 中的八个自动地从链描述符(比如链描述符 290)中的它们的对应的域加载。存储器 290 中的对应的描述符域的格式与为 DMA 通道专用寄存器 270 定义的格式相同。当 DMA 控制器 200 处于常规模式时，如由 DMA 模式控制寄存器的 DMA 模式位所定义的，可以对通道控制 272、通道状态 274、后续描述符地址 282-1 以及后续描述符高位地址 282-2 寄存器进行读/写访问。其余的寄存器都是只读的，并且无论何时 DMA 通道从本地系统存储器读取链描述符，它们都被自动加载由链描述符 290 定义的新值。

表 1

| 位 | 说明 |
|---|---|
| 31:8 | 保留 |
| 7：6 | 保留 |
| 5：4 | 保留 |
| 3 | 通道恢复。如果设置，那么当通道空闲时(CSR 中的通道活动位是有效的)或者当通道完成其后续描述符地址为空的 DMA 传输时，通过重读位于本地系统存储器中的在当前描述符地址寄存器中的地址处的当前描述符设置，使通道恢复链接(chaining)。当出现以下情况时，这个位由硬件清除：<br>● 通道完成DMA传输并且后续地址寄存器不为零。在这种情况下，通道继续链表中的下一个描述符设置。<br>● 通道空闲或者通道完成当前描述符设置的执行并且后续描述符地址寄存器为零。在这种情况下，通道仅仅清除“通道恢复”位，并保持空闲。<br>一旦设置，软件不能清除这个位。当在 CSR 中设置停止或者终止位时，硬件阻止设置这个位。 |

| | |
|---|---|
| | 在试图恢复当前描述符设置之前，软件必须清除 CSR 停止和终止位。如果设置了 CSR 的链结束位，那么当恢复当前描述符设置时，DMA 通道清除链结束位。 |
| 2 | 停止 **DMA**。当设置时，使当前 DMA 传输停止。这使得通道不在源端请求总线。将队列中的任何数据清空到目的端，并且清除 CCR（所有位 3：0）和 CSR（通道活动位）中的所有相关位。一旦设置，该位不能由软件清除。应用软件必须小心地设置该位，因为任何 DMA 一旦停止就不能从那一点重新开始。该位具有比挂起 DMA 位更高的优先级。在清除通道活动位并且设置了 DMA 停止位之后，由硬件清除该位。 |
| 1 | 挂起 **DMA**。当设置时，允许当前 DMA 描述符设置结束，但挂起通道链接。通道持续在源端向总线请求当前描述符。当将用于该描述符设置的队列中的数据被清空到目的地端时，设置 CSR 中的 DMA 挂起位。清除该位并设置“通道恢复”将使用由后续描述符地址寄存器指示的描述符设置重新起动 DMA 传输，并清除 CSR 中的 DMA 挂起位。该位对通道活动位没有影响。 |
| 0 | 起动 **DMA**。当设置时，使通道能够进行 DMA 传输。一旦设置，该位不能由软件清除。当 DMA 传输结束或者由软件停止 DMA 或者 DMA 传输遇到不可恢复的错误时，由硬件清除该位。当在 CSR 中设置停止位或者终止位时，硬件阻止设置该位。在使用一个新描述符设置起动 DMA 通道之前，DMA 通道必须是空闲的并且软件必须清除 CSR。 |

再次参考图 3，存储器映射 DMA 寄存器 270 包括通道控制寄存器(CCR)272，如进一步参照以上提供的表 1 所说明的。在一个实施例中，CCR272 为 DMA 通道指定整个操作环境。在一个实施例中，在初始化系统存储器中的链描述符并以存储器中第一个 DMA 链描述符的位置更新后续地址寄存器 282(282-1，282-2)之后，设备驱动器初

始化 CCR 207。在一个实施例中，当 DMA 通道活动时可以写 CCR 272 以修改 DMA 通道操作(例如，停止、挂起等)。

表 2

| 位 | 说明 |
|---|---|
| 31:7 | 保留 |
| 6 | 保留 |
| 5 | 通道活动。当设置时，表示通道在使用中并且活动地执行 DMA 数据传输。当清除时，表示通道不活动并且可用于由应用软件进行 DMA 传输配置。当软件通过设置 CCR 的起动 DMA 位初始化 DMA 传输时由硬件设置通道活动标志，并且响应的 DMA 通道从本地系统存储器中装载描述符设置。 |
| 4 | **DMA** 异常终止（**DABRT**）。当设置时，表示用于该通道的当前 DMA 传输遇到不可恢复的错误。如果设置了 DCR 的 DMA 异常终止中断允许位，这标志着对处理器接口的中断。应用软件可以使用该位轮询是否不能够中断。细节在 DMA FERR/NERR 寄存器中。 |
| 3 | **DMA** 停止（**DSTP**）。当设置时，表示由应用软件通过设置 CCR 中的停止 DMA 位来停止用于该通道的当前 DMA 传输。如果设置了 DCR 的 DMA 停止中断允许位，那么这标志对处理器接口的中断。应用软件可以使用该位轮询是否不能够中断。 |
| 2 | **DMA** 挂起（**DSUS**）。当设置时，表示由应用软件通过设置 CCR 中的挂起 DMA 位要求挂起该通道。如果设置了 DCR 的 DMA 挂起中断允许位，这标志对处理器接口的中断。应用软件可以使用该位轮询是否不能够中断。 |
| 1 | 传输结束（**EOT**）。当设置时，表示通道成功完成了至少一个描述符的无差错 DMA 传输。如果设置了 DCR 的传输结束中断允许位，这标志对处理器接口的中断。 |

| | |
|---|---|
| | 应用软件可以使用该位轮询是否不能够中断。 |
| 0 | 链结束（**EOC**）。当设置时，表示通道成功完成了链表中的最后一个描述符的无差错 DMA 传输。如果设置了 DCR 的链结束中断允许位，这标志对处理器接口的中断。应用软件可以使用该位轮询是否不能够中断。 |

在一个实施例中，通道状态寄存器(CSR)274 包含表示 DMA 通道状态的标志。在一个实施例中，寄存器 274 由 VDD 驱动器 260 读取以得到当前 DMA 通道状态并确定中断源。如表 2 所示，包含在 CSR 274 之内的各种标志包括通道活动位，其表示通道正在使用；DMA 终止(DMABRT)位，其表示终止 DMA 传输。同样，CSR 274 标志还包括 DMA 停止(DSTP)位，当设置它时，表示停止这个通道的当前 DMA 传输。

在一个实施例中，设置 DMA 挂起(DSUS)位表示挂起 DMA 请求，而传输结束(EOT)位表示通道已为至少一个描述符成功完成 DMA；链结束(EOC)标志表示对包括最后一个描述符的所有描述符无错误的进行 DMA 传输。如这里所述，术语“断言”(assert)、“设置”(set)、“取消断言”(deassert)等术语可以表示数据信号，它们是低态有效或者高态有效信号。所以当与信号关联时，可交替地使用这些术语请求或者暗示高态有效或低态有效信号。

当前描述符地址寄存器(CDAR)276(276-1，276-2)包含本地系统存储器中的当前链描述符的地址的低位和高位。在一个实施例中，当接通电源或系统复位时将 CDAR 276 清零，并且当开始新的块传输时将自动用后续描述符地址寄存器(NDAR)中的值装载。在一个实施例中，CDAR 276 在正常操作过程中是只读的，并且可以通过软件轮询以便随着它穿过 DMA 描述符链来监控 DMA 通道的进程。其余的 DMA 寄存器 270 对应于 DMA 链描述符 290，可以在起动 DMA 传输之前由 VDD 驱动器 260 对其编程。

在一个实施例中，VDD 驱动器 260 通过在本地系统存储器中构建一个或多个链描述符来初始化 DMA 通道。如图 3 和 4 所示，描述符可以包括表示 DMA 数据的初始位置的源地址 292、表示 DMA 请

求将 DMA 数据移动或传输到的位置的目的地址 294、表示将要传输的字节的数量的传输计数 298、以及表示描述符链中的后续描述符的地址的后续描述符地址 296。将描述符链的最后描述符的后续描述符地址 296 设置为零。

在操作中，在由应用程序或进程发出文件操作请求之后，VDD 驱动器 260 从 OS 接收文件操作请求。根据这种请求，VDD 驱动器 260 对链描述符 290 的各个部分编程，包括源地址 292 和目的地址 294 以及传输计数。在一个实施例中，以双字(Dword)为单位提供传输计数 298，并且可以称为 Dword 计数 298。一旦对链描述符进行了编程，VDD 驱动器 260 就更新 DMA 寄存器 270 的后续描述符地址寄存器 286(NDAR)。

在一个实施例中，由 DMA 控制器 200 使用 NDAR 286 以便定位由 VDD 驱动器 260 编程的链描述符。因此，一旦用存储器中的第一个链描述符 290 的地址填充 NDAR 286，则 VDD 驱动器 260 就可以设置通道控制寄存器 272(见表 1)之内的 DMA 起动位。如图 4 所示，VDD 驱动器 260 可以对 VDD 存储器 250 设置一串块传输；但是，如果将来自应用程序或进程的文件请求限制为从 VDD 250 的单个块传输，那么 VDD 驱动器 260 将 DMA 链描述符 290-1 的后续描述符地址域设置为空(零)值。

再次参考图 3，DMA 寄存器 270 的源地址寄存器(SAR)278 和目的地址寄存器(DAR)280 域还包括高位地址寄存器域(278-2 和 280-2)。在一个实施例中，提供高位地址寄存器域(278-2 和 280-2)以允许 DMA 访问请求的 36-位或 64 千兆字节寻址范围。因此，与对建立在易失性系统存储器 140 之内的 VDD 存储器 250 或 RAMDISK 的传统访问不同，DMA 控制器 200 允许对数据传输进行 32 位地址界限以外的寻址，否则，对超过四千兆字节(GB)边界的 VDD 存储器 250 的访问请求需要使用 CPU 102 的页地址扩展(PAE)能力。

图 5 根据一个实施例说明了方法 300 的概要，其用于利用 DMA 控制器 200 之内的集成 DMA 引擎 210 来执行 VDD 存储器 250 之内的文件操作。典型地，进程 302(比如应用程序或运行程序)可以向包

含在 VDD 存储器 250 之内的文件请求文件操作或者读/写请求。在转移 310 处，进程 302 发出 VDD 访问请求。当由例如 OS 304 检测到这种访问时，在转移 320 处，OS 304 通过将 VDD 访问请求传递给 VDD 驱动器 260 从而将该调用传递给 VDD 驱动器 260。

在一个实施例中，VDD 驱动器 260 识别对其检测到文件操作或者读/写请求的文件。一旦检测到，VDD 驱动器 260 将文件名以及在该文件中该访问请求被引导到的偏移量转换为主存储器 140 之内的物理地址。在一个实施例中，根据主存储器 140 之内的映射到 VDD 存储器 250 的物理地址范围执行转换。一旦确定该文件的物理地址，VDD 驱动器 260 就可以通过填充 DMA 链描述符来编程 DMA 源和目的地址以及传输长度，例如，如图 3 和 4 所示，从而发出 DMA 读/写(R/W)访问请求到 DMA 控制器 200，如转移 330 所示。

如图 4 所示，如果多个数据块涉及文件操作或者对 VDD 250 的数据访问请求，那么 VDD 驱动器 260 可以为一串 VDD 块传输编程。随后，填充 DMA 控制器 200 的 NDAR 282 以便 DMA 控制器 200 可以装载 DMA 寄存器 270 之内的描述符链，如图 3 所示。一旦由例如 VDD 驱动器 260 设置了通道控制寄存器 272 的 DMA 起动位，DMA 控制器 200 就执行这种动作。随后，如图 2 所示，DMA 引擎 210 通过执行 DMA R/W，根据至少一个由 VDD 存储器 250 编程的 DMA 链描述符来执行数据访问请求或者文件操作，如转移 340 处所示。

在一个实施例中，一旦 DMA 引擎 210 完成了传输，在转移 350 处，DMA 引擎可以向 VDD 驱动器 260 发出一个中断，以通知 VDD 驱动器文件操作完成以及已将数据传输到 VDD 250 或从 VDD 250 传输到分配给进程 302 的存储器。典型地，一旦 VDD 驱动器 260 在转移 350 接收到完成 DMA 传输的通知，VDD 驱动器 260 可以通知进程 302 所请求的文件操作或者向/从 VDD 存储器 250 的读/写访问请求已完成。在一个实施例中，在转移 360 处，VDD 驱动器 260 通知进程 302 已完成的 DMA 传输。

因此，在一个实施例中，VDD 驱动器 260 的生成将访问 VDD 存储器 250 之内的文件数据的操作转交给 DMA 控制器的集成 DMA 引

擎，例如参照图 1 和图 2 所说明的。因此，通过将数据传输工作从 CPU 102 分担到 DMA 控制器 200，提高了总体的系统性能，从而使得 CPU 102 避免执行传统的数据传输以便执行对 VDD 存储器 250 的文件操作。此外，不同于请求使用 CPU 的页地址扩展(PAE)模式以寻址超过 32 位寻址边界的范围的传统的 RAMDISK 驱动器，将 VDD 存储器数据传输分担给 DMA 引擎 210，不仅避免了通过使用 PAE 模式而通常导致的对于 VDD 驱动器 260 的性能损失，而且避免了对于所有系统进程的性能损失。

图 6 是说明使用公开的技术对设计进行仿真、模拟以及制造的各种表示或格式的框图。表示设计的数据会以许多方式表示该设计。首先，由于在仿真中有用，硬件可以用硬件描述语言来表示，或者用另一个功能描述语言表示，其必需提供关于期望所设计的硬件如何执行的计算机化的模型。可以将硬件模型 410 存储在存储介质 400 中，比如计算机存储器，因此，可以使用模拟软件 420 模拟该模型，其中模拟软件将特定的一套测试 430 应用到硬件模型以确定它是否真正按照计划来运行。在一些实施例中，没有将仿真软件记录、收集或者包含在介质中。

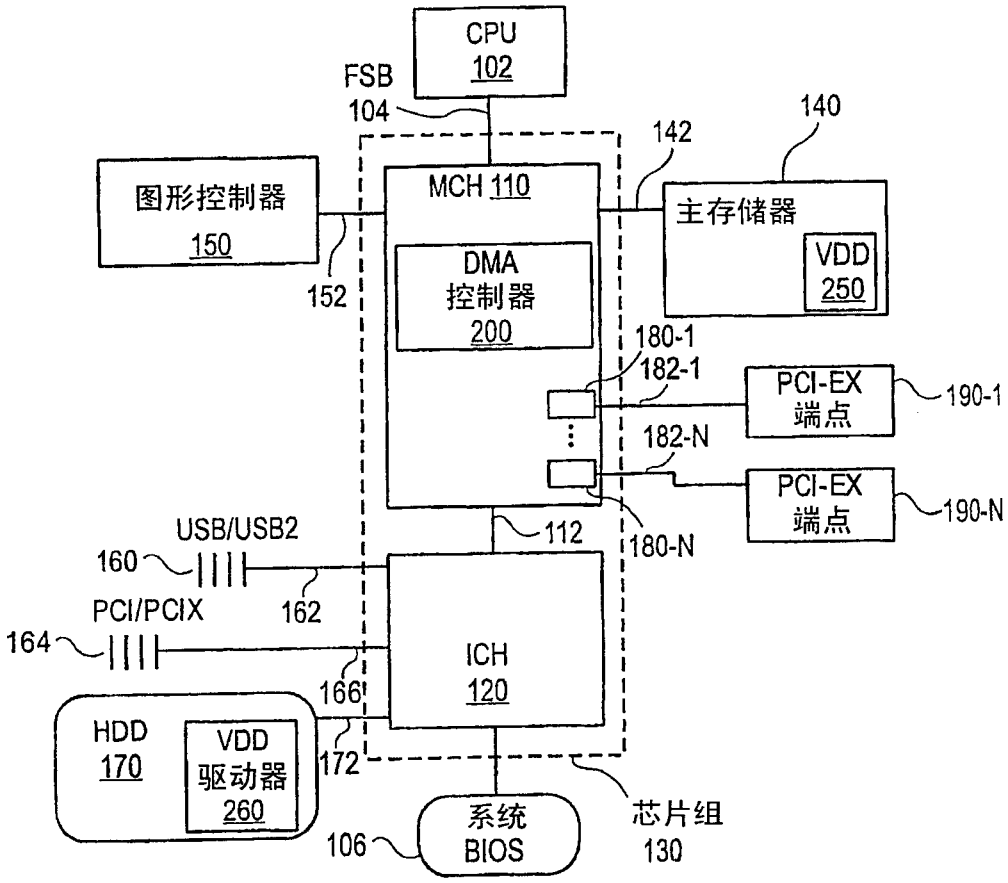
在设计的任何表示中，可以将数据存储在任何形式的机器可读介质中。调制或生成光波或电波 460 以传送这种信息，存储器 450 或者磁或光存储器 440，比如光盘，可以是机器可读介质。任何这些介质都可以携带设计信息。因此，术语“携带”(例如，机器可读数据介质携带信息)覆盖了存储在存储装置的信息或者编码或调制为载波的信息。描述设计或设计的细节的位的集合是(当体现为机器可读介质，例如载体或存储介质时)一物品，其可以封闭在其自身内外，或者由他人用于进一步设计或者制造。

替代实施例

应当理解，对于其它实施例，可以使用不同的系统配置。例如，虽然对于其它实施例系统 100 包括单个 CPU 102，但是多处理器系统(其中，一个或多个处理器可以在结构和操作上与如上所述的 CPU 102 类似)也会受益于对各种实施例的 VDD 存储器的集成 DMA 存储器控

制器访问。此外，不同类型的系统或者不同类型的计算机系统，例如服务器、工作站、桌面计算机系统、游戏系统、嵌入式计算机系统、刀片服务器等等，可以用于其它实施例。

已经公开了实施例和最佳方式，可以对已公开的实施例做出变形或改变，同时仍然属于由以下权利要求所定义的本发明的实施例的范围。


CPU
102
FSB
104
140
142
图形控制器
150
MCH 110
主存储器
VDD
250
DMA
控制器
200
152
180-1
182-1
PCI-EX
端点
190-1
182-N
PCI-EX
端点
190-N
180-N
112
USB/USB2
160
162
PCI/PCIX
164
ICH
120
166
HDD
170
VDD
驱动器
260
172
系统
BIOS
106
芯片组
130

计算机系统
100

图1

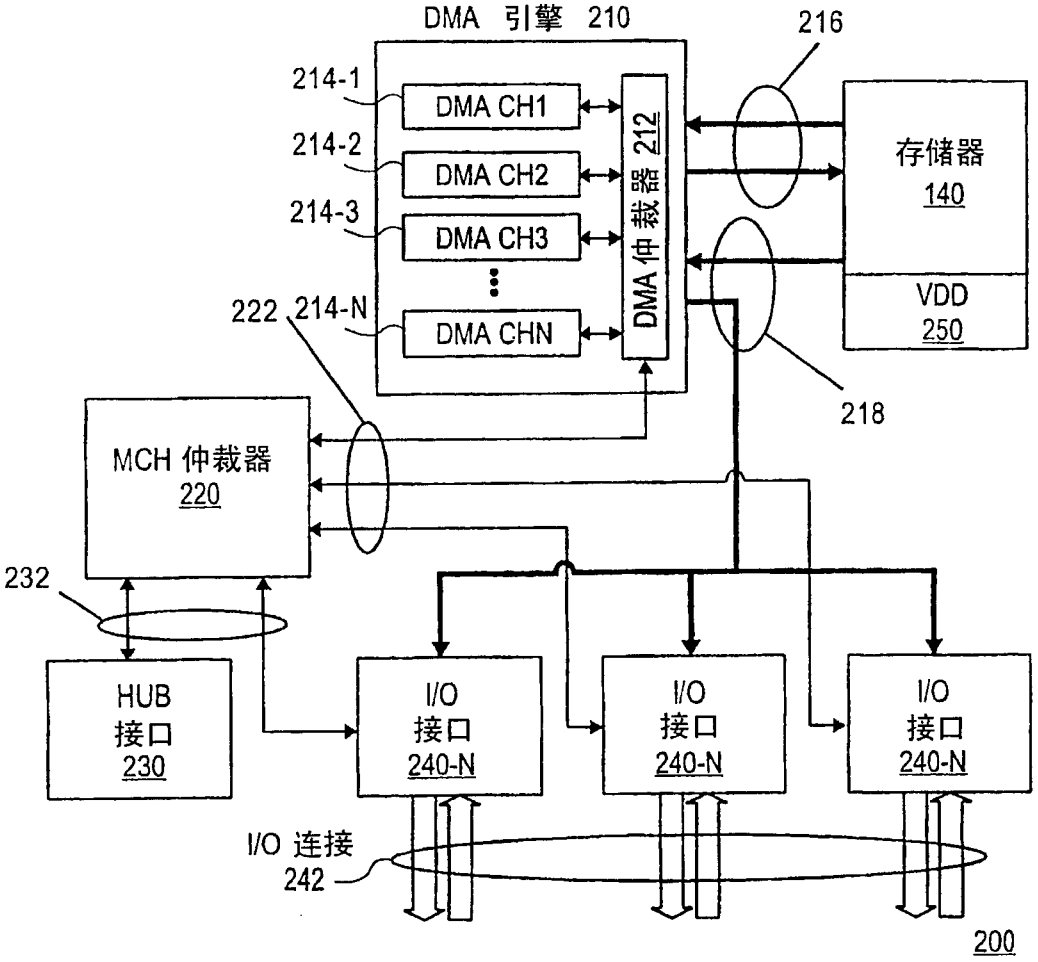

DMA 引擎 210
214-1
DMA CH1
214-2
DMA CH2
214-3
DMA CH3
214-N
DMA CHN
DMA 仲裁器 212
216
存储器
140
VDD
250
218
222
MCH 仲裁器
220
232
HUB
接口
230
I/O
接口
240-N
I/O
接口
240-N
I/O
接口
240-N
I/O 连接
242
200

图2

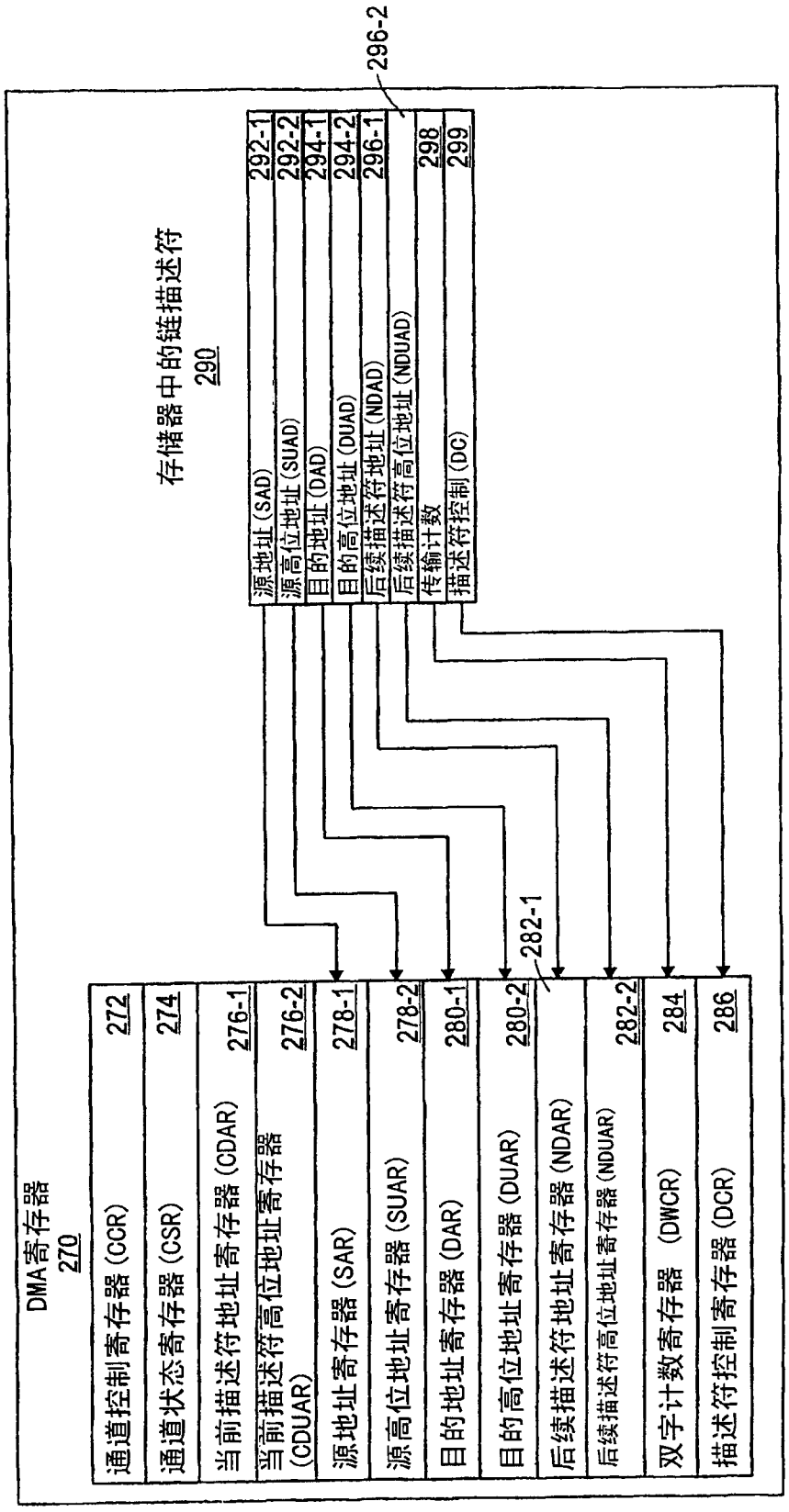

DMA寄存器
270
通道控制寄存器 (CCR) 272
通道状态寄存器 (CSR) 274
当前描述符地址寄存器 (CDAR) 276-1
当前描述符高位地址寄存器 (CDUAR) 276-2
源地址寄存器 (SAR) 278-1
源高位地址寄存器 (SUAR) 278-2
目的地址寄存器 (DAR) 280-1
目的高位地址寄存器 (DUAR) 280-2
后续描述符地址寄存器 (NDAR) 282-1
后续描述符高位地址寄存器 (NDUAR) 282-2
双字计数寄存器 (DWCR) 284
描述符控制寄存器 (DCR) 286
存储器中的链描述符
290
296-2
源地址 (SAD) 292-1
源高位地址 (SUAD) 292-2
目的地址 (DAD) 294-1
目的高位地址 (DUAD) 294-2
后续描述符地址 (NDAD) 296-1
后续描述符高位地址 (NDUAD)
传输计数 298
描述符控制 (DC) 299

图3

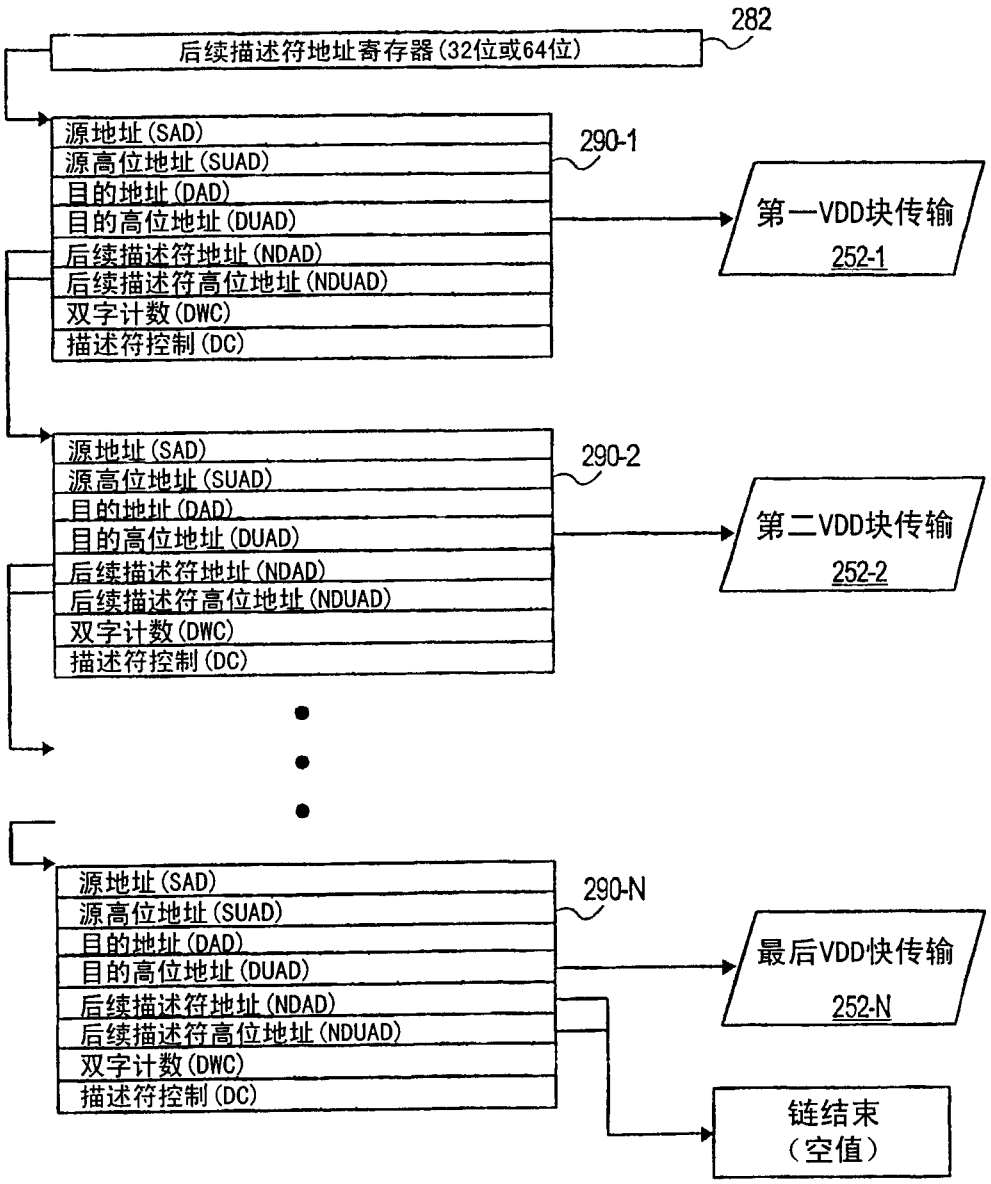

282
后续描述符地址寄存器(32位或64位)
源地址(SAD)
源高位地址(SUAD)
目的地址(DAD)
目的高位地址(DUAD)
后续描述符地址(NDAD)
后续描述符高位地址(NDUAD)
双字计数(DWC)
描述符控制(DC)
290-1
第一VDD块传输
252-1
源地址(SAD)
源高位地址(SUAD)
目的地址(DAD)
目的高位地址(DUAD)
后续描述符地址(NDAD)
后续描述符高位地址(NDUAD)
双字计数(DWC)
描述符控制(DC)
290-2
第二VDD块传输
252-2
源地址(SAD)
源高位地址(SUAD)
目的地址(DAD)
目的高位地址(DUAD)
后续描述符地址(NDAD)
后续描述符高位地址(NDUAD)
双字计数(DWC)
描述符控制(DC)
290-N
最后VDD快传输
252-N
链结束
(空值)

图4

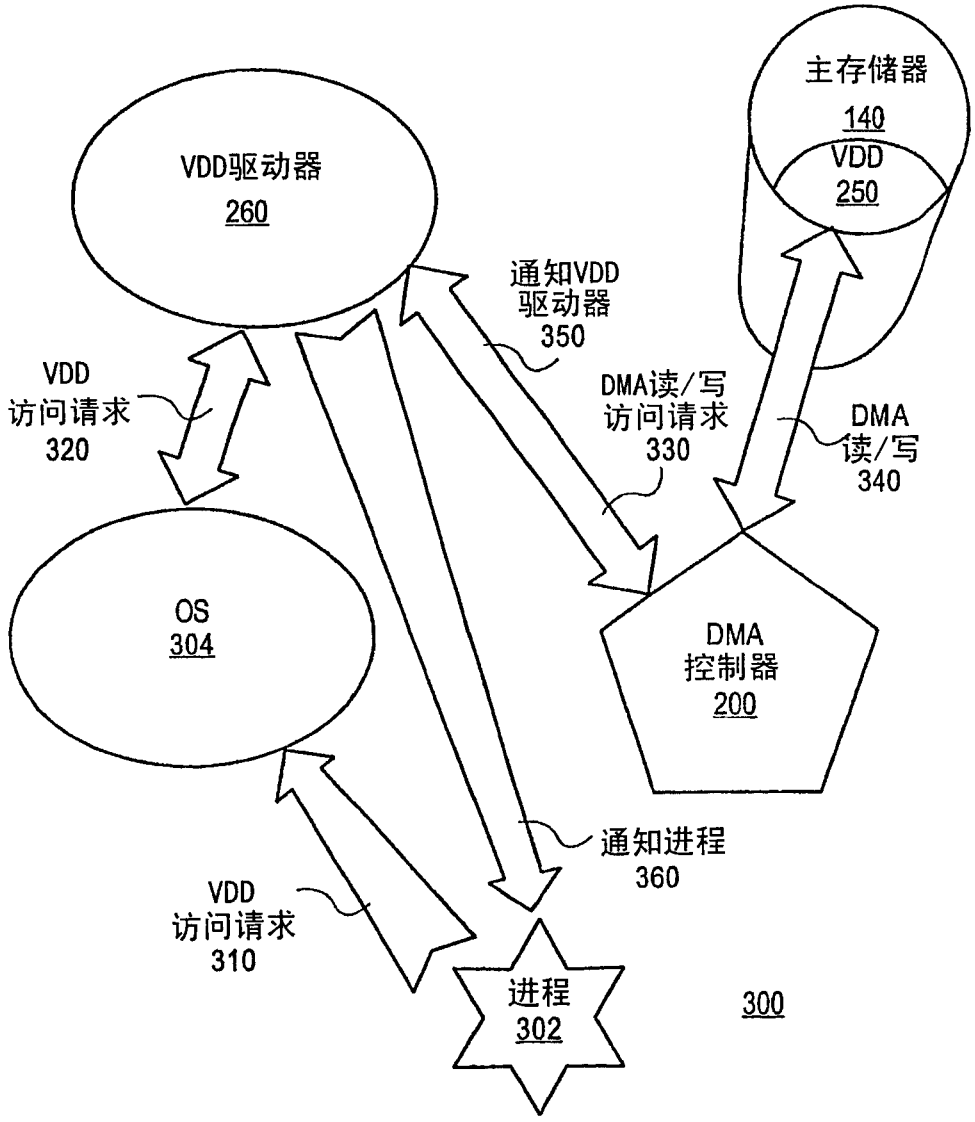

主存储器
140
VDD
250
VDD驱动器
260
通知VDD
驱动器
350
VDD
访问请求
320
DMA读/写
访问请求
330
DMA
读/写
340
OS
304
DMA
控制器
200
通知进程
360
VDD
访问请求
310
进程
302
300

图5

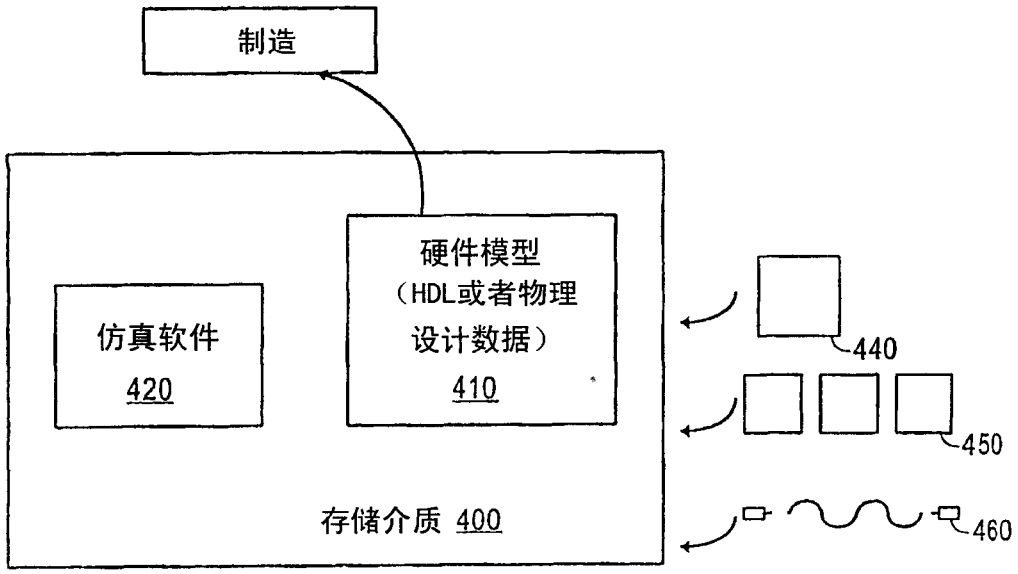



图6