[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 12/00 (2006.01)
G06F 13/00 (2006.01)

[12] 发明专利说明书

专利号 ZL 03809269.7

[45] 授权公告日 2007 年 1 月 24 日

[11] 授权公告号 CN 1296832C

[22] 申请日 2003.4.7 [21] 申请号 03809269.7

[30] 优先权

[32] 2002.4.25 [33] US [31] 10/063,466

[86] 国际申请 PCT/US2003/010746 2003.4.7

[87] 国际公布 WO2003/091883 英 2003.11.6

[85] 进入国家阶段日期 2004.10.25

[73] 专利权人 国际商业机器公司

地址 美国纽约州

[72] 发明人 布赖恩·L·吉 黄重礼 切幡年明 宗藤正治

[56] 参考文献

US6178479B1 2001.1.23 G06F12/00

US6081872A 2000.6.27 G06F12/00

CN1206150A 1999.1.27 G06F12/12

US6161208A 2000.12.12 G11C11/00

US4725945A 1988.2.16 G11C7/00

US5544306A 1996.8.6 G06F12/00

US6018763A 2000.1.25 G06F15/167

US6205076B1 2001.3.20 G11C7/00

审查员 郑宗玉

[74] 专利代理机构 北京市柳沈律师事务所

代理人 邸万奎 黄小临

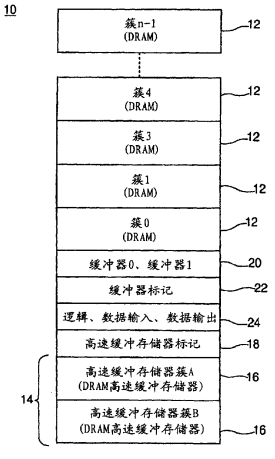
权利要求书 7 页 说明书 19 页 附图 14 页

[54] 发明名称

用破坏读取高速缓冲存储器来缓冲的破坏读取随机访问存储器系统

[57] 摘要

公开了一种存储器存储系统(10)。示范实施例中，存储器存储系统包括多个存储器存储簇(12)以及与其通信的高速缓冲存储器(14)。多个存储器存储簇(12)以及高速缓冲存储器(14)还包括破坏读取存储器存储元件。



1、一种存储器存储系统，包括:

多个存储器存储簇；以及

与所述多个存储器存储簇通信的高速缓冲存储器;

其中所述多个存储器存储簇以及所述高速缓冲存储器还包括破坏读取存储器存储元件;

与所述多个存储器存储簇和所述高速缓冲存储器通信的线缓冲器结构。

2、如权利要求 1 所述的存储器存储系统，其中，所述破坏读取存储器存储元件包括动态随机访问存储器存储单元。

3、如权利要求 1 所述的存储器存储系统，其中，所述高速缓冲存储器包括所述多个存储器存储簇的一部分。

4、如权利要求 1 所述的存储器存储系统，还包括:

与所述线缓冲器结构相关的缓冲器标记；以及

与所述高速缓冲存储器相关的高速缓冲存储器标记。

5、如权利要求 4 所述的存储器存储系统，其中，所述线缓冲器结构包括一对缓冲器。

6、如权利要求 5 所述的存储器存储系统，其中:

所述多个存储器存储簇还包括数目为 n 的 DRAM 存储簇，所述 n 个 DRAM 存储簇的每一个具有与其相关的数目为 m 的字线；以及

所述高速缓冲存储器还包括一对 DRAM 高速缓冲存储器簇，所述一对 DRAM 高速缓冲存储器簇的每一个具有与其相关的所述数目为 m 的字线。

7、如权利要求 6 所述的存储器存储系统，其中，所述一对缓冲器中的每个均能够在其中存储数据页，所述数据页包括从选择的 DRAM 存储簇的所选字线、或选择的 DRAM 高速缓冲存储器簇的所选字线破坏读取的数据位。

8、如权利要求 7 所述的存储器存储系统，其中，可将原始包含在所述数目的 DRAM 存储簇的一个中的数据存储在所述一对 DRAM 高速缓冲存储器簇的一个中。

9、如权利要求 8 所述的存储器存储系统，其中，将与所述数目的 DRAM 存储簇原始相关的所述一对 DRAM 高速缓冲存储器簇中包含的任意数据通

过直接映射方案存储在其中；

其中，将与所述数目的 DRAM 存储簇中的一个之内的给定字线地址相关的数据存储在所述一对 DRAM 高速缓冲存储器簇的一个之内的所述给定字线地址内。

10、如权利要求 7 所述的存储器存储系统，其中，所述一对缓冲器中的每个包括电平敏感锁存器。

11、一种动态随机访问存储器 DRAM 系统，包括：

n 个 DRAM 存储簇，所述 n 个 DRAM 存储簇的每一个具有与其相关的 m 个字线；

高速缓冲存储器，所述高速缓冲存储器包括第一 DRAM 高速缓冲存储器簇和第二 DRAM 高速缓冲存储器簇，所述第一 DRAM 高速缓冲存储器簇和所述第二DRAM高速缓冲存储器簇均具有与其相关的所述数目 m的字线；

线缓冲器结构，所述线缓冲器结构包括能够存储从所述 DRAM 存储簇和所述第一和第二 DRAM 高速缓冲存储器簇读取的数据的一对缓冲器；和

控制算法，用于控制在所述 DRAM 存储簇、所述一对缓冲器和所述 DRAM 高速缓冲存储器簇之间传送数据；

其中从所述 DRAM 存储簇和所述 DRAM 高速缓冲存储器簇读取的数据被破坏地读取。

12、如权利要求 11 所述的 DRAM 系统，还包括：

高速缓冲存储器标记，用于存储包含在所述 DRAM 高速缓冲存储器簇的给定字线内的数据的 DRAM 存储簇地址信息；以及

缓冲器标记，用于存储包含在所述一对缓冲器内的数据的 DRAM 存储簇地址信息。

13、如权利要求 12 所述的 DRAM 系统，其中所述缓冲器标记还包括：

第一标志，其指明所述一对缓冲器内是否存在有效数据；以及

第二标志，其指明随机请求的数据是否包含在所述一对缓冲器中的任一个中。

14、如权利要求 11 所述的 DRAM 系统，还包括：

数据输入总线，所述数据输入总线用于通过所述缓冲器结构，将外部始发数据接收到 DRAM 系统；

数据输出总线，所述数据输出总线用于将内部存储的数据传送出 DRAM

系统;

读取二级数据线，用于将来自所述 DRAM 存储簇的数据传送到所述一对缓冲器中的一个;

高速缓冲存储器读取二级数据线，用于将来自所述 DRAM 高速缓冲存储器簇的数据传送到所述一对缓冲器中的另一个;

写入二级数据线，用于将来自所述一对缓冲器中的任一个的数据传送到所述 DRAM 存储簇；以及

高速缓冲存储器写入二级数据线，用于将来自所述一对缓冲器中的任一个的数据传送到所述 DRAM 高速缓冲存储器簇。

15、如权利要求 14 所述的 DRAM 系统，其中，所述一对缓冲器还包括电平敏感存储锁存器。

16、如权利要求 14 所述的 DRAM 系统，其中，所述一对缓冲器还包括边缘触发锁存器。

17、如权利要求 14 所述的 DRAM 系统，其中，所述系统的控制算法包括:

定义的一组可允许状态，用于存在于所述 DRAM 存储簇、所述一对缓冲器、以及所述 DRAM 高速缓冲存储器簇内的有效数据;

用于在对存储于所述 DRAM 存储簇内的数据的第一次随机访问请求之前初始化 DRAM 系统的步骤，其中所述用于初始化的步骤遵循所述定义的一组可允许状态；以及

在所述对数据的第一次随机访问请求之后、以及在其后对数据的任意随后的随机访问请求之后，用于确保请求的数据在所述线缓冲器结构中成为可用的步骤，所述用于确保的步骤也遵循所述定义的一组可允许状态。

18、如权利要求 17 所述的 DRAM 系统，其中:

所述用于确保请求的数据在所述缓冲器结构中成为可用的步骤还包括:在所述 DRAM 存储簇和线缓冲器结构之间、以及在所述 DRAM 高速缓冲存储器簇和所述线缓冲器结构之间运行的、确定的一系列数据传送操作;

其中，所述确定的一系列数据传送操作为路径无关的，其特征在于，所述确定的一系列数据传送操作不取决于存储在所述 DRAM 存储簇、所述线缓冲器结构、以及所述高速缓冲存储器中的数据的先前位置。

19、一种用于控制数据在动态随机访问存储器 DRAM 系统内的移动的

方法，所述系统具有数目为 n 的 DRAM 存储簇、线缓冲器结构以及高速缓冲存储器，所述方法包括:

定义的一组可允许状态，用于存在于存储簇、缓冲器结构、以及高速缓冲存储器内的有效数据;

在对存储于 DRAM 存储簇内的数据的第一次随机访问请求之前，初始化 DRAM 系统，其中所述初始化遵循所述定义的一组可允许状态; 以及

在对数据的所述第一次随机访问请求之后、以及在其后对数据的任意随后的随机访问请求之后，确保请求的数据在所述缓冲器结构中成为可用;

其中，在所述确保请求的数据在所述缓冲器结构中成为可用之后，支持所述一组可允许状态。

20、如权利要求 19 所述的方法，其中:

所述确保请求的数据在所述缓冲器结构中成为可用还包括: 在 DRAM 存储簇和缓冲器之间、以及在高速缓冲存储器和缓冲器之间运行确定的一系列数据传送操作;

其中，所述确定的一系列数据传送操作是路径无关的，其特征在于，所述确定的一系列数据传送操作不取决于存储在 DRAM 存储簇、缓冲器结构、以及高速缓冲存储器中的数据的先前位置。

21、如权利要求 20 所述的方法，其中:

n 个 DRAM 存储簇中的每一个具有数目 m 的与其相关的字线以及与其相关的存储簇地址;

高速缓冲存储器包括第一高速缓冲存储器簇和第二高速缓冲存储器簇，所述第一和第二高速缓冲存储器簇也具有与其相关的 m 个字线; 以及

缓冲器结构包括: 第一缓冲器和第二缓冲器，所述第一和第二缓冲器每一个都能够在其中存储数据字，所述数据字包括来自于特定字线、从 DRAM 存储簇中的一个、或所述第一和第二高速缓冲存储器簇中的一个读取的数据。

22、如权利要求 20 所述的方法，其中所述定义的一组可允许状态还包括在给定的时钟周期的开始确保:

所述缓冲器结构包括第一和第二缓冲器，所述第一和第二缓冲器的每一个在其中包含有效数据，所述第一缓冲器中的所述有效数据具有与所述第二缓冲器中的所述有效数据相同的字线地址 i，其中 i 为 0 和 m 之间的任意数;

在具有等于 i 的字线地址的所述第一或第二高速缓冲存储器簇中，没有有效数据；以及

对于除了 i 之外的所有其它字线地址，有一个数据字存储在所述第一高速缓冲存储器簇或所述第二高速缓冲存储器簇中的对应字线中。

23、如权利要求 22 所述的方法，其中所述确保请求的数据在所述缓冲器结构中成为可用包括：直到所述给定的时钟周期的结束，确保：

对应于存储簇地址 X、以及字线地址 j 的请求的数据被包含于所述第一和第二缓冲器中的一个中，其中 X 为 0 和 n-1 之间的任意数，j 为 0 和 m 之间的任意数；

所述第一和第二缓冲器中的另一个包含：具有字线地址 j、但不同于 X 的存储簇地址的数据；

在具有等于 j 的字线地址的所述第一或第二高速缓冲存储器簇中，没有有效数据；以及

对于除了 j 之外的所有其它字线地址，有一个数据字存储在所述第一高速缓冲存储器簇或所述第二高速缓冲存储器簇中的对应字线中。

24、如权利要求 23 所述的方法，其中所述确保请求的数据在所述缓冲器结构中成为可用还包括：

确定是否最初在对应的 DRAM 簇、所述第一和第二缓冲器中的一个、或所述第一和第二高速缓冲存储器簇中的一个中找到请求的数据；

其中所述运行确定的一系列数据传送操作取决于请求的数据的初始位置。

25、一种用于控制数据在破坏读取动态随机访问存储器 DRAM 系统内移动的方法，所述系统具有数目 n 的 DRAM 存储簇、线缓冲器结构以及破坏读取 DRAM 高速缓冲存储器，所述方法包括：

定义的一组可允许状态，用于存在于存储簇、缓冲器结构、以及高速缓冲存储器内的有效数据；

在对存储于 DRAM 存储簇内的数据的第一次随机访问请求之前，初始化 DRAM 系统，其中所述初始化遵循所述定义的一组可允许状态；以及

在对数据的所述第一次随机访问请求之后、以及在其后对数据的任意随后的随机访问请求之后，确保请求的数据在所述缓冲器结构中成为可用；

其中在所述确保请求的数据在所述缓冲器结构中成为可用之后，支持所

述一组可允许状态。

26、如权利要求 25 所述的方法，其中：

所述确保请求的数据在所述缓冲器结构中成为可用还包括：在 DRAM 存储簇和缓冲之间、以及在高速缓冲存储器和缓冲器之间运行确定的一系列数据传送操作；

其中所述确定的一系列数据传送操作与路径无关，其特征在于，所述确定的一系列数据传送操作不取决于存储在 DRAM 存储簇、缓冲器结构、以及高速缓冲存储器中的数据的先前位置。

27、如权利要求 26 所述的方法，其中：

n 个 DRAM 存储簇中的每一个具有数目 m 的与其相关的字线以及与其相关的存储簇地址；

高速缓冲存储器包括第一高速缓冲存储器簇和第二高速缓冲存储器簇，所述第一和第二高速缓冲存储器簇还具有与其相关的 m 个字线；以及

缓冲器结构包括：第一缓冲器和第二缓冲器，所述第一和第二缓冲器中的每一个能够在其中存储数据字，所述数据字包括来自于特定字线、从 DRAM 存储簇中的一个或所述第一和第二高速缓冲存储器簇中的一个读取的数据。

28、如权利要求 27 所述的方法，其中所述定义的一组可允许状态还包括在给定的时钟周期的开始确保：

所述第一和第二缓冲器中的每一个在其中包含有效数据，所述第一缓冲器中的所述有效数据具有与所述第二缓冲器中的所述有效数据相同的字线地址 i，其中 i 为 0 和 m 之间的任意数；

在具有等于 i 的字线地址的所述第一或第二高速缓冲存储器簇中没有有效数据；以及

对于除了 i 之外的所有其它字线地址，有一个数据字存储在所述第一高速缓冲存储器簇或所述第二高速缓冲存储器簇中的对应字线中。

29、如权利要求 28 所述的方法，其中所述确保请求的数据在所述缓冲器结构中成为可用包括：直到所述给定的时钟周期的结束，确保：

对应于存储簇地址 X 以及字线地址 j 的请求的数据被包含于所述第一和第二缓冲器中的一个中，其中 X 为 0 和 n-1 之间的任意数，j 为 0 和 m 之间的任意数；

所述第一和第二缓冲器中的另一个包含：具有字线地址 j、但不同于 X 的存储簇地址的数据；

在具有等于 j 的字线地址的所述第一或所述第二高速缓冲存储器簇中没有有效数据；以及

对于除了 j 之外的所有其它字线地址，有一个数据字存储在所述第一高速缓冲存储器簇或所述第二高速缓冲存储器簇中的对应字线中。

30、如权利要求 29 所述的方法，其中所述确保请求的数据在所述缓冲器结构中成为可用还包括：

确定是否最初在对应的 DRAM 簇、所述第一和第二缓冲器中的一个、或所述第一和第二高速缓冲存储器簇中的一个中找到请求的数据；

其中所述运行确定的一系列数据传送操作取决于请求数据的初始位置。

31、如权利要求 27 所述的方法，其中所述定义的一组可允许状态还包括：在给定的时钟周期的开始确保：

如果所述第一和第二缓冲器的每一个在其中包含有效数据，则所述第一缓冲器中的所述有效数据与所述第二缓冲器中的所述有效数据有相同的字线地址 i，其中 i 为 0 到 m 之间的任意数；

如果所述第一和第二缓冲器在其中均包含有效数据、或者如果所述第一和第二缓冲器中的一个在其中包含字线地址为 i 的有效数据，则在具有等于 i 的字线地址的所述第一或所述第二高速缓冲存储器簇中没有有效数据；和

如果所述第一和第二缓冲器的任一个或两个缓冲器均包含字线地址为 i 的有效数据，则对于除了 i 之外的所有其它字线地址，有一个数据字存储在所述第一高速缓冲存储器簇或所述第二高速缓冲存储器簇中的对应字线中。

32、如权利要求 31 所述的方法，其中所述确保请求的数据在所述缓冲器结构中成为可用包括：直到所述给定的时钟周期的结束，确保：

对应于存储簇地址 X 以及字线地址 j 的请求的数据被包含于所述第一和第二缓冲器中的一个中，其中 X 为 0 和 n-1 之间的任意数，j 为 0 和 m 之间的任意数；

在具有等于 j 的字线地址的所述第一或所述第二高速缓冲存储器簇中没有有效数据；以及

对于除了 j 之外的所有其它字线地址，有一个数据字存储在所述第一高速缓冲存储器簇或所述第二高速缓冲存储器簇中的对应字线中。

用破坏读取高速缓冲存储器来缓冲的破坏读取随机访问存储器系统

技术领域

本发明一般涉及集成电路存储器装置，并尤其涉及被破坏读取存储器(destructive-read memory)所高速高速缓冲存储器的破坏读取存储器的随机访问存储器系统。

背景技术

亚微米(sub-micron)CMOS 技术的发展已促成了微处理器速度的显著增加。(增加速度)约每三年四倍，微处理器速度已超过了 1Ghz。连同微处理器技术中的这些进步，已带来更先进的软件和多媒体应用程序，其随之需要更大的存储器，用于其中的应用程序。因此，存在对具有更高密度和性能的更大的动态随机访问存储器(DRAM)的日益增加的要求。

这些年中，受到需要更大存储器容量的系统需求的驱使，DRAM 结构已经得到了发展。然而，以其随机访问时间(tRAC)和其随机访问周期时间(tRC)为特征的 DRAM 的速度没有以类似方式提高。因此，由于 CPU 的时钟速度随着时间而稳定增长，所以，在 DRAM 和 CPU 之间，有不断加大的速度差距。

通常，DRAM 阵列的随机访问周期时间(tRC)由阵列时间常数来确定，该常数表示完成全部随机访问操作的时间量。这样的操作包括：字线激活、位线上的信号发展、位线检测、信号写回、字线去激活(deactivation)和位线预充电。由于在传统 DRAM 结构中顺序执行这些操作，所以，增加 DRAM 的传输速度(或带宽)成为问题。

为某些应用改进 DRAM 系统的行访问周期时间的一种方法是：实现存储在 DRAM 单元中的数据的破坏读取，并随后将破坏读取的数据暂时存储到与同一本地存储器阵列的读出放大器相连接的缓冲单元中。(例如，参看授予 Wakayama 等人的美国专利第 6205076 和 6333883 号)此方法中，连接到普通读出放大器块的本地存储器阵列中的不同字线可被顺序地破坏读取多次，该

次数被设置为 1 加上每个读出放大器的缓冲单元的数目。然而，由于用于每个本地 DRAM 阵列的缓冲单元和相关控制逻辑需要较大的面积，因而此方法中可实际实现的缓冲单元的数目较小。此外，只要缓冲单元的数目小于原始单元阵列中的字线的数目，此系统仅为有限数目的数据访问情况改进访问周期时间，而不是总的应用中所需要的随机访问周期时间。

改进 DRAM 系统的随机访问周期时间的更实用的方法是：实现存储在 DRAM 单元中的数据的破坏读取，并随后将破坏读取的数据暂时存储到主存储器阵列之外的基于 SRAM 的高速缓冲存储器中。基于 SRAM 的高速缓冲存储器具有至少与一个单簇(bank)DRAM 阵列相同数目的字线(这里叙述的术语“簇”表示共享相同的读出放大器的存储器单元的阵列)。在 2001 年 4 月 26 日提交、并共同转让给了本申请的受让人的美国专利申请第 09/843504 号、标题为“A Destructive Read Architecture for Dynamic Random Access Memories (用于动态随机访问存储器的破坏读取结构)”的文献中描述了此技术。此技术中，随后调度延迟写回操作，用于在后面时间将数据恢复到适当的 DRAM 存储器位置。延迟写回操作的调度取决于基于 SRAM 的高速缓冲存储器内的空间的可用性。尽管这样的方法对于减少随机访问周期时间有效，但基于 SRAM 的高速缓冲存储器的使用可能占用不理想数量的芯片固定资源(real estate)，并导致为在 DRAM 和高速缓冲存储器之间传输数据的更复杂的互连线路。因此，在尤其关心芯片面积的场合，期望减少随机访问周期时间，而不会由使用基于 SRAM 的高速缓冲存储器而占用相对较大的装置面积。

发明内容

通过一种存储器存储系统，可克服或减小现有技术的上面讨论的和其它缺点和缺陷。在示范实施例中，存储器存储系统包括多个存储器存储簇和与其通信的高速缓冲存储器。多个存储器存储簇以及高速缓冲存储器均还包括破坏读取存储器存储元件。

在优选实施例中，破坏读取存储器存储元件是动态随机访问存储器(DRAM)存储单元。另外，该系统还包括与多个存储器存储簇和高速缓冲存储器通信的线缓冲器结构。缓冲器标记与线缓冲器结构相关，而高速缓冲存储器标记与高速缓冲存储器相关。线缓冲器结构包括一对寄存器阵列，其每一个能够存储整页字线数据，如同在 DRAM 阵列中。换句话说，线缓冲器结构

按照数据 I/O 包括两个寄存器。

多个存储器存储簇还包括数目为 n 的 DRAM 存储簇，其每一个具有与其相关的数目为 m 的字线。高速缓冲存储器还包括一对 DRAM 高速缓冲存储器簇，其每一个具有与其相关的相同数目 m 的字线。此外，每个缓冲其均能够在其中存储数据页，数据页包括从选择的 DRAM 存储簇的所选字线、或选择的 DRAM 高速缓冲存储器簇的所选字线破坏地读取的数据位。可将原始包含在 DRAM 存储簇中的一个内的数据存储在 DRAM 高速缓冲存储器簇中的一个中。将与 DRAM 存储簇原始相关的、包含在所述一对 DRAM 高速缓冲存储器簇中的任意数据通过直接映射方案存储在其中，其中将与 DRAM 存储簇中的一个之内的给定字线地址相关的数据存储在 DRAM 高速缓冲存储器簇中的一个之内的那个字线地址内。

另一方面，是一种用于控制数据在动态随机访问存储器(DRAM)系统内移动的方法，该系统具有数目 n 的 DRAM 存储簇、线缓冲器结构和高速缓冲存储器。在示范实施例中，该方法包括：为存在于存储簇、缓冲器结构、以及高速缓冲存储器内的有效数据定义一组可允许的状态。在对存储于 DRAM 存储簇内的数据的第一次随机访问请求之前初始化 DRAM 系统，其中该初始化遵循定义的一组可允许状态。在对数据的第一次随机访问请求之后，以及在其后对数据的任意随后的随机访问请求之后，在缓冲器结构中使请求的数据可用，并且维持所述一组可允许状态。

附图说明

参照示范附图，其中在几幅图中，相同的元素被相同地标记。

图 1 为根据本发明的实施例的破坏读取动态随机访问存储器(DRAM)的示意方框图；

图 2 为图示包括在 DRAM 系统中的高速缓冲存储器标记的结构的表；

图 3 为图示包括在 DRAM 系统中的缓冲器标记的结构的表格；

图 4(a)为图示包括在 DRAM 系统中的缓冲器结构的一个实施例的示意图；

图 4(b)为图示用于图 4(a)中示出的数据线的示范连接的示意图；

图 5(a)为图示 DRAM 系统配置下允许的可能的数据传输操作的例子的表；

图 5(b)为图示图 4 中示出的缓冲器结构的操作的时序图;

图 5(c)为图 4 中示出的缓冲器结构的替换实施例的示意图;

图 5(d)为图示与图 5(c)的缓冲器结构相关的流水线(pipeline)方案的表;

图 6(a)到 6(f)为与 DRAM 系统相结合使用的强读式(strong form)算法下，表示各种数据传输操作的状态图。

图 7 为图示强读式算法中使用的初始化程序的状态图;

图 8 为图示强读式算法中的可选数据传输操作的状态图;

图 9 为图示强读式算法下可允许状态的状态表;

图 10(a)到 10(d)为图示替换地可与 DRAM 系统相结合使用的普通式(general form)算法下的可允许状态的状态表; 以及

图 11(a)到 11(f)为在与 DRAM 系统相结合使用的普通式算法下，表示各种数据传输操作的状态图。

具体实施方式

这里公开了一种基于也被破坏读取存储器所高速缓冲存储的破坏读取存储器的随机访问存储器系统。破坏读取存储器描述一种存储器结构，其在执行读操作之后，丢失其数据，并由此，执行随后的写回操作，以将该数据恢复到存储器单元中。如果读取 DRAM 内的数据而没有紧接着写回到其中，那么该数据之后将不再驻留在该单元中。如上所述，改进随机访问周期时间的一个方法为: 以破坏读取模式操作存储器阵列，所述破坏读取模式与使用 SRAM 数据高速缓冲存储器的延迟写回的调度相结合。而且如前所述，然而，现有 SRAM 装置占用更多的装置固有资源，并且，与具有单个访问晶体管和存储电容器的 DRAM 单元相比，每单元通常包括四个或更多晶体管。因此，本发明的实施例允许相同的破坏读取 DRAM 簇也起到高速缓冲存储器的作用，由此在将要讨论的其它优点中节省装置固有资源。

简要地说，在当前实施例中，通过从多个 DRAM 簇读取而破坏的数据现在被也以破坏读取模式工作的双簇 DRAM 数据高速缓冲存储器所高速缓冲存储(例如，写入)。除了 DRAM 簇和双簇(dual-bank)高速缓冲存储器之外，还包括一对寄存器线缓冲器，每个用于单页数据的存储。高速缓冲存储器标记为高速缓冲存储器的每个字线存储簇信息，以及标志，以指明双簇高速缓冲存储器中的特定簇当前是否具有有效数据。还有缓冲器标记，对于每个缓

冲器来说，包含指明其中是否存在有效数据的标志，以及与该数据相关的簇和行信息。指明两个缓冲器中的哪一个的另一个标志可包含从先前周期随机请求的数据。

作为将更详细描述的，基于“可允许状态的规则”的概念，可设计一个或更多路径无关算法，用来确定将在对下一个时钟周期的准备中实现的数据传输操作。数据传输操作(即移动)将仅取决于 DRAM 簇、高速缓冲存储器和缓冲器中的数据的当前状态，而不是当前状态之前的历史。为了便于理解，将下面的详细描述组织为两个主要部分：(1)系统架构；以及(2)为该架构实现的调度算法。

I.系统架构

首先参照图 1，其中示出了破坏读取动态随机访问存储器(DRAM)系统 10 的示意方框图。DRAM 系统 10 包括：n 个 DRAM 存储簇 12(分别指定为 BANK 0 到 BANK n-1)；破坏读取 DRAM 高速缓冲存储器 14，其包括双高速缓冲存储器簇 16；高速缓冲存储器标记 18；一对寄存器线缓冲器 20(分别指定为缓冲器 0 和缓冲器 1)；缓冲器标记 22；以及相关逻辑电路 24。应当注意，这里使用的术语“簇(bank)”或“BANK”表示共享一组普通的读出放大器的存储器单元阵列。

相关逻辑电路 24 还可包括：元件中的接收器/数据、元件外的 OCD/数据、以及其它逻辑元件。不象现有的高速缓冲存储器，两个 DRAM 高速缓冲存储器簇 16 可与 n 个正常 DRAM 簇 12 相同(在配置以及性能上)。因此，还可将所述系统 10 视为具有 n+2 簇架构，其中，n 个 DRAM 簇用于传统的存储器存储，而 2 个 DRAM 簇用作高速缓冲存储器。下文中，用于高速缓冲存储器 14 的两个 DRAM 簇 16 将被称为“高速缓冲存储器簇”，其分别被指定为 CBANK A 和 CBANK B。

每个 DRAM 簇 12(BANK 0 到 n-1)和高速缓冲存储器簇 16(CBANK A、B)有相同数目的字线和位线。优选实施例中，相同的阵列支持电路(例如，字线驱动器和读出放大器配置)可支持 DRAM 簇 12 和高速缓冲存储器簇 16。另一方面，只要每个高速缓冲存储器簇 16(CBANK A、B)包含至少与 DRAM 簇 12(BANK 0 到 n-1)相同数目的字线和位线，便可为每个 DRAM 簇 12(BANK 0 到 n-1)和每个高速缓冲存储器簇 16(CBANK A、B)使用不同的阵列配置。

两个高速缓冲存储器簇 16 均共享直接映射高速缓冲存储器方案中的高

速缓冲存储器标记 18。可将与 DRAM 簇 12 中的一个的具体字线地址(例如，字线 0)相关的数据存储在两个高速缓冲存储器簇 16 中的一个(A 或 B，但不是所有两个)的那个具体字线中。这允许高速缓冲存储器 14 在将新数据写入到另一个 DRAM 簇 12 时，从 DRAM 簇中的一个读取数据。图 2 中示出了高速缓冲存储器标记 18 的结构。可以看出，对于每个字线(编号 0 到 m)，高速缓冲存储器标记 18 存储 DRAM 簇地址信息(此例子中示出为 3 位编码的簇地址)，以及在高速缓冲存储器簇 CBANK A 和 CBANK B 的字线中存在数据的指示。“A”和“B”标志指明 CBANK A、CBANK B 中是否存在有效数据，或者都不存在。图示的例子中，CBANK A 的字线 0 包含来自(DRAM)BANK 0、字线 0 的有效数据。而且，CBANK B 的字线 2 包含来自 BANK 7、字线 2 的有效数据，而 CBANK A 的字线 3 包含来自 BANK 3、字线 3 的有效数据。

两个线缓冲器 20 中的每个均能够存储单页数据(即字)。线缓冲器 20 可由寄存器阵列形成，并分别具有独立输入和输出端口。图 3 图示了缓冲器标记 22 的结构，其包括每个缓冲器 20 的簇地址、行地址(此例子中的 8 位编码的地址)、有效标志和请求标志。然而，如后面将描述的，在优选实施例中，与每个缓冲器 20 相关的行地址应当相同，因此，其在两个缓冲器之间共享。有效标志指明缓冲器中的数据是否有效。请求标志指明缓冲器 20(缓冲器 0 或缓冲器 1)是否包含先前为具体簇所请求的数据、以及用于读取到 data_out(数据输出)(OCD)或从 data_in(数据输入)(接收器)写入的行地址。

现在参照图 4，其中示出了图示用于每个 data_in/out 引脚的两个缓冲器 20(缓冲器 0 和缓冲器 1)的结构的示意图。缓冲器 20 用于处理 DRAM 簇(BANK 0 到 n-1)和高速缓冲存储器 14(CBANK A、B)之间的数据传输的流量，同时避免数据线上任何潜在的数据争用。将 data_in/data_out 总线 30(包括 data_in 线 30a 和 data_out 线 30b)通过由 read(读)/write(写)信号 gw0/ gr0 以及 gw1/ gr1 控制的多个对应传输门 32 分别连接到缓冲器 0 和缓冲器 1。data_in/data_out 总线 30 提供 DRAM 系统 10 和从该系统读取或写入到该系统的任意外部装置(用户)之间的外部接口。

读取二级数据线(RSDL)34 为单向信号总线，其将 DRAM 簇阵列中的读出放大器(SA)或二级读出放大器(SSA)(未示出)的输出通过由信号 r0 控制的传输门 36 连接到缓冲器 0。换句话说，将从 DRAM 簇 12 中的一个读取到缓冲器 20 的数据通过 RSDL 34 发送到缓冲器 0。类似地，CRSDL(高速缓冲存储

器读取二级数据线)38 为单向信号总线，其将与高速缓冲存储器 14 相关的读出放大器(SA)或二级读出放大器(SSA)(未示出)的输出通过由信号 r1 控制的传输门 40 连接到缓冲器 1。换句话说，将从高速缓冲存储器簇 16 中的一个读取到缓冲器 20 的数据通过 CRSDL 38 发送到缓冲器 1。

另外，写入二级数据线(WSDL)42 为单向信号总线，其将来自缓冲器 0 或缓冲器 1 的输出数据连接回到 DRAM 簇 12。这通过由信号 w00 和 w10 控制的多路复用传输门 44 来完成。对应地，高速缓冲存储器写入二级数据线(CWSDL)46 为单向信号总线，其将来自缓冲器 0 或缓冲器 1 的输出数据连接到高速缓冲存储器簇 16。这通过由信号 w01 和 w11 控制的多路复用传输门 48 来完成。

图 4(b)为图示用于 n 个 DRAM 簇、两个 DRAM 高速缓冲存储器簇和两个缓冲器之间的数据线 WSDL、RSDL、CWSDL 和 CRSDL 的示范连接的示意图。通过例子的方式，假定数据线的宽度为 128 位。

尽管通过使用图 4(a)中示出的电平敏感锁存器来实现缓冲器结构，但也可实现基于边缘触发锁存器和流水线标记比较的替换方案，如后面将描述的。此外，应当注意，当前实施例下，定义图 4(a)的缓冲器结构，使得：总是将从 DRAM 簇 12 输入的数据存储在缓冲器 0 中；总是将从高速缓冲存储器簇 16 输入的数据存储在缓冲器 1 中；以及从缓冲器 0 和缓冲器 1 输出的数据可转到 DRAM 簇 12 或高速缓冲存储器簇 16。然而，应当理解，可反转该结构，以便可将从 DRAM 簇 12 或高速缓冲存储器簇 16 输入到缓冲器中的数据存储在任一缓冲器中，而在具体缓冲之外写入的数据将仅转到 DRAM 簇或高速缓冲存储器簇 16。

为了对 DRAM 系统 10 的操作的理解，下面，通过假定将命令和地址信号的 1/4 时钟建立时间用于标记比较，来讨论单时钟周期操作。单周期操作意味着，每个 DRAM 簇(包括高速缓冲存储器簇)可在一个时钟周期内完成读或写操作。每个时钟周期中，将会有不多于一个的来自 DRAM 簇的读操作、不多于一个的来自高速缓冲存储器簇的读操作、不多于一个的到 DRAM 簇的写操作、不多于一个的到高速缓冲存储器簇的写操作。因此，每个周期期间可在 DRAM 簇和高速缓冲存储器簇之间最多发生四个独立的读或写操作，同时仍允许与两个缓冲器成功通信。通过每个数据传输操作，能够通过两个缓冲器中的一个进行通信。

图 5(a)为图示当前系统配置下允许的四个可能数据传输操作中的每个的例子的表格。例如，在对存储在 BANK 2、字线 4(缩写为 bank2_wl4)中的数据的随机访问请求期间，可发生下面的操作:

(1)将先前在缓冲器 0(例如，来自 bank0_wl2)中的数据移回到 BANK 0;

(2)将该数据从 bank2_wl4 移出到缓冲器 0;

(3)将先前在缓冲器 1(例如，来自 bank3_wl2)中的数据移到高速缓冲存储器簇(例如，CBANK A)；以及

(4)将数据从 CBANK B 中的字线 4(例如，bank0_wl4)移到缓冲器 1。

由于至少有两个 DRAM 簇、两个缓冲器、以及两个高速缓冲存储器簇，所以能够使所有四个上面的数据传输操作同步。

如将在后面更详细说明的，通常，基于请求的命令(如果有的话)和数据的现有状态，来确定上面的时钟周期期间的系列操作。将定义一系列允许状态，并且，将实现支持允许状态的规则的算法。上面图 5(a)中的例子的上下文中，在紧挨在 bank2_wl4 中的数据请求之前，缓冲器 0 最初包含: 先前从与 DRAM 簇 0 中的第二字线(bank0_wl2)相关的单元读取的数据位。另外，缓冲器 1 最初包含: 先前从与 DRAM 簇 3 中的第二字线(bank3_wl2)相关的单元读取的数据位。高速缓冲存储器标记 18 保持与高速缓冲存储器簇中的每个字线相关的簇地址信息，以及 A 或 B 簇中是否有任何有效数据。具体地，上面例子假定高速缓冲存储器 B 最初包含: 先前从 DRAM 簇 0 中的第四字线(bank0_wl4)读取的数据位。从先前时钟周期获知所有这些初始状态。

当在建立时间期间接收了新命令(请求 bank2_wl4)时，完成标记比较。标记比较确定请求的数据是否为缓冲器命中(hit)、高速缓冲存储器命中、或缓冲器及高速缓冲存储器未中(miss)(即 DRAM 簇命中)。此例子中，请求的数据既不在缓冲器中也不在高速缓冲存储器中，并由此，比较结果被视为 DRAM 簇命中。换句话说，请求的数据实际在其指定的 DRAM 簇位置中。除了定位请求的数据之外，标记比较还检查看: 在与请求中相同的字线地址(例如，wl4)的任一高速缓冲存储器簇中是否有任何有效数据。此例子的结果是来自 DRAM 簇 0、字线 4(bank0_wl4)的有效数据，其被发现在高速缓冲存储器簇 B 中。

由于当前系统使用直接映射调度，所以，为了将来的调度，不应将来自 bank0_wl4 的数据位存储在任一高速缓冲存储器簇中。因此，来自 bank0_wl4

的数据位将被传送到存储器 1。同时，请求的数据 bank2_wl4 需要被传送到缓冲器 0，以随后被用户检索。然而，必须首先将最初在缓冲器 0(bank0_wl2)中的数据返回到其在 DRAM 簇中的位置(即 BANK 0，字线 2)，其中 DRAM 簇与请求中的簇不同。由于两个缓冲器均包含有效数据，它们中的一个将与 DRAM 簇号码相关，其中所述 DRAM 簇号码不是与请求的 DRAM 簇相同的号码。首先检查缓冲器 0，并且通过标记比较确定其与和请求中相同的 DRAM 簇号码不相关，于是将缓冲器 0 中的数据送回 DRAM 簇 0。其它缓冲器中的数据，即来自缓冲器 1 中的 bank3_wl2 的数据位将被传送到高速缓冲存储器簇 A。

当前系统的基本数据传送原则为：将具有相同字线号码的最多两个数据页存储在两个缓冲器中，作为一对。一个用于请求的数据页，如果有必要的话，另一个用于传送高速缓冲存储器之外的有效数据页(具有对应于与请求的数据相同的字线地址的具体字线地址)，以便避免将来周期中的数据溢出。只要遵循此配对规则，便可完全保持数据传输的完整性，而不丢失任何簇数据。

现在参照图 5(b)，其中示出了图示图 4(a)中示出的缓冲器结构的操作的时序图。再次，有假定的对应于新请求的命令、地址和数据的建立时间。由于相关逻辑相对简单，所以少量时间(例如，0.13 微米技术中大约 0.5ns)可能就足够了。另外，可为内部 DRAM 操作实现延迟时钟。在建立时间期间，将地址信息(add)用于标记比较，而将命令(cmd)用于查看是否存在对数据传送的请求。命令还被流水线输送到下一个时钟，使得随后如同在读(写)等待时间 1 操作中一样，执行从(或到)数据缓冲器的读(或写)命令。此电平敏感锁存器方案中，图 4(a)中的“w”门 44 在时钟周期的前一半期间接通，而在时钟周期的后一半期间关断，由此，允许数据被发送到 WSDL 42，并被锁存一个时钟。“r”门 36、40 在时钟周期的后半段接通，由于假定宏读取等待时间(macro read latency)小于一个时钟周期，所以允许来自 DRAM 簇的有效数据进入。随后，数据在下一个周期的上升沿被锁存到寄存器缓冲器中。如果接收了读命令，则将数据从缓冲器(与请求的地址相关)读取到 data_out 线 30b。如果接收了写命令，则将数据从 data_in 线 30a 写入到与请求的地址相关的缓冲器中。

应当注意，无论请求为读、写、或带有位屏蔽的写，对于提出的随机访问存储器系统来说，内部操作将首先把与请求的字线相关的数据页带到缓冲器中的一个中，其中执行读(将数据向外拷贝到 data_out30b)或写(将数据页更

新为 data_in30a 上的输入)。除了 data_in 和 data_out 线以及控制门 gw0、gw1、gr0、gr1 上的操作，用于 DRAM 簇、高速缓冲存储器、以及缓冲器的调度算法和数据移动对于读和写请求来说是等同的。

图 5(c)中示出了基于正时钟边缘触发锁存器 52 的替换的缓冲器方案 50。这里，将一个时钟周期用于标记读取和比较，以和 ASIC 方法更为一致。使用一个时钟标记比较，图 5(d)图示了与图 5(c)的时钟边缘触发设计相一致的时序流程。此实施例中的读取等待时间为两个时钟。应当注意，如图 5 所示，两个连续的命令(请求 1 和请求 2)顺序堆积并以无缝流水线形式运行。在说明书的下个部分将示出：通过实现为 DRAM 系统 10 特别设计的路径无关算法，用于任意随机顺序的无缝堆积成为可能。

II.调度算法

为了成功地使用具有破坏读取高速缓冲存储器的破坏读取 DRAM 阵列的上面描述的架构，适当的调度方案应当在适当位置，以便将系统维持在跟随任意新随机访问请求之后的可允许状态。一般方法是，首先定义可允许状态，初始化该系统以遵循可允许状态(即初始化)，并随后确保在任意给出的数据传送操作之后，保持可允许状态(即系统连续性)。

可允许状态的规则(强读式)

优选实施例中，定义了以在所有两个缓冲器中保持有效数据的对称算法为特征的可允许状态的“强读式”规则，其中所述数据具有相同的字线地址，但来自不同的 DRAM 簇。因此，在每个时钟周期的上升沿，将满足下面的规则：

规则#1－具有存储在两个缓冲器中的每个的数据字，其具有彼此共同的字线地址。缓冲器中的两个数据字的一个为对应于来自先前的随机访问请求的簇地址和字线地址的特定数据。

此规则的例子可为：缓冲器 0 包含从 DRAM 簇 2、字线 3(如同从先前周期请求)读取的数据，而缓冲器 1 包含先前从高速缓冲存储器(高速缓冲存储器簇 A 或高速缓冲存储器簇 B)的字线 3 读取、并与簇 4、字线 3 相关的数据。

规则#2－高速缓冲存储器中没有与上面的字线地址(即，与缓冲器中的数据相关的具体字线地址)当前相关的有效数据。

继续上面的例子，高速缓冲存储器簇 A 或高速缓冲存储器簇 B 均不会有存储在字线地址 3 上的有效数据。也就是说，在高速缓冲存储器标记的字线

3 上，A=0 且 B=0。

规则#3－对于除了对应于缓冲器的字线地址的字线地址之外的每个字线地址，具有存储在一个且仅有一个的高速缓冲存储器簇中的一个且仅有一个有效数据字。

因此，上面的例子中，对于除了字线地址 3 之外的每个字线地址，(A=1 且 B=0)或(A=0 且 B=1)。

应当注意，在规则#1 下，与请求的簇和字线地址(用于读或写)相关的数据页将在下一个时钟周期到达缓冲器，用于适当的读和写操作。给定上面概述的可允许状态的规则，对于任意随机访问请求(读/写)，有可能运行预定的程序，其中借助该程序，提出的系统将被初始化，并随后在可允许状态保持任意时钟周期。

初始化

强读式算法的第一部分通过初始化程序开始。随着系统启动，缓冲器标记 22(来自先前讨论的图 3)如下设置：(1)将用于缓冲器 0 和缓冲器 1 的有效标志设为“1”；(2)行地址对应于字线 0；(3)用于缓冲器 0 的簇地址为簇 1；(4)用于缓冲器 1 的簇地址为簇 0；以及(5)由于没有先前请求，所以用于所述两个缓冲器的请求标志为 0。

另外，随着系统启动，除了字线 0，用于高速缓冲存储器标记 18 的高速缓冲存储器簇 A 的每个标志被初始化为 A=1，而高速缓冲存储器标记 18 中的所有簇地址都被设置到簇 0。将用于高速缓冲存储器簇 B 的每个标志初始化为 B=0。如上所述，通过设置缓冲器和高速缓冲存储器标记，缓冲器 0 对应于与 bank(簇)1_woedline(字线)0 相关的有效数据字，而缓冲器 1 对应于与 bank0_wordline0 相关的有效数据字。最后，除了字线 0 不对应有效数据，双簇高速缓冲存储器中的所有其它字线具有与高速缓冲存储器簇 A 中的簇 0 相关的有效数据。因此，初始满足了上面陈述的强读式规则。

连续性

跟随初始化之后，将假定紧挨在时钟周期的上升沿稍前，作出随机读或写请求。在时钟周期的上升沿，随机访问请求(读或写)将被表示为 Xj，其中“X”为簇地址，而“j”为字线号码(地址)。术语 Di 将表示最初存储在缓冲器 0 中的数据页，其中“D”为簇地址，而“i”为字线号码。术语 Qi 将表示最初存储在缓冲器 1 中的数据页，其中“Q”为簇地址，而“i”为字线号码。

应当注意，根据上述规则#1，所有情况中 D≠Q，并且字线号码(i)对于缓冲器 0 和缓冲器 1 是相同的。

而且，在规则#2 和规则#3 的情况下，对于任意给定的字线号码 k≠i，有且仅有一个与存储在高速缓冲存储器中的字线 k 相关的有效数据页。由此，术语 C(k)被表示为用于高速缓冲存储器标记中的字线 k 的对应簇地址。因此，对于任意给定的请求 Xj，将在相关的 DRAM 簇、两个缓冲器中的一个、或高速缓冲存储器中找到数据。下面，示出为三个一般可能情况而运行的结果数据传送操作:

情况 1－缓冲器命中

此情况中，j=i，且 X=D 或 X=Q。也就是说，请求的数据已经存储在缓冲器 0 或缓冲器 1 中。由于已经满足可允许状态的规则，此时钟周期中不实现进一步的数据传送。由图 6(a)的状态图表中缺少变化可反映出此情况。

情况 2－高速缓冲存储器命中

如果请求的数据 Xj 被包含在高速缓冲存储器中，则 j≠i(在规则#2 下)。也就是说，请求的数据的字线号码不对应于缓冲器中的数据的字线。此外，由于单页数据不能对应于两个簇地址，因而 X≠D 或 X≠Q，或者 X≠D 且 X≠Q。

如果 X≠D，则用于 Dj 的数据既不在缓冲器中(从上面段落可知)也不在高速缓冲存储器中(在规则#3 下)，因此，用于 Dj 的数据在对应的 DRAM 簇中。随后实现下面的步骤，以遵循上面用于可允许状态的规则:

Xj 从一个高速缓冲存储器簇(A 或 B)移到缓冲器 1;

Di 从缓冲器 0 移到另一个高速缓冲存储器簇(B 或 A);

Qi 从缓冲器 1 移到 DRAM 簇 Q;

Dj 从 DRAM 簇 D 移到缓冲器 0。

图 6(b)中演示了此系列数据移动。另一方面，如果 X=D，则 X≠Q 必定为真，并且在对应的 DRAM 簇中找到用于 Qj 的数据。因此，随后实现下面的步骤:

Xj 从一个高速缓冲存储器簇(A 或 B)移到缓冲器 1;

Qi 从缓冲器 1 移到另一个高速缓冲存储器簇(B 或 A);

Di 从缓冲器 0 移到 DRAM 簇 D;

Qj 从 DRAM 簇 Q 移到缓冲器 0。

图 6(c)中演示了此系列数据移动。

情况 3a－缓冲器未中，高速缓冲存储器未中，j=i

如果请求的数据 Xj 既不在缓冲器中也不在高速缓冲存储器中，则其(Xj)在对应的 DRAM 簇中。由于 j=i，则 X≠Q 也为真。因此，可在两个步骤中执行遵循的操作，如图 6(d)所示：

Xj 从 DRAM 簇 X 移到缓冲器 0;

Di 从缓冲器 0 移到 DRAM 簇 D。

情况 3b－缓冲器未中，高速缓冲存储器未中，j≠i，X≠D

此情况中，请求的数据再次位于对应的 DRAM 簇中。然而，请求的数据的字线地址不同于缓冲器中的数据的字线地址。在可允许状态的规则下，存在存储于一个高速缓冲存储器簇中的用于行地址 j 的有效的 Cj。由于 X≠D，则请求的数据的簇地址不同于缓冲器 0 中的数据的簇地址，并实现下面的步骤：

Xj 从一个 DRAM 簇 X 移到缓冲器 0;

Cj 从一个高速缓冲存储器簇(A 或 B)移到缓冲器 1;

Qi 从缓冲器 1 移到另一个高速缓冲存储器簇(B 或 A);

Di 从缓冲器 0 移到 DRAM 簇 D。

图 6(e)中演示了此系列数据移动。

情况 3c－缓冲器未中，高速缓冲存储器未中，j≠i，X=D，X≠Q

此情况与上面的情况 3b 的唯一不同在于：请求的数据的簇地址现在和包含在缓冲器 0 中的数据的簇地址相同(即 X=D)。然而，请求的数据的簇地址不同于包含在缓冲器 1 中的数据的簇地址(即 X≠D)必定为真。因此，如图 6(f)所示，实现下面的步骤：

Xj 从一个 DRAM 簇 X 移到缓冲器 0;

Cj 从一个高速缓冲存储器簇(A 或 B)移到缓冲器 1;

Di 从缓冲器 0 移到另一个高速缓冲存储器簇(B 或 A);

Qi 从缓冲器 1 移到 DRAM 簇 Q。

初始化程序的替换实施例通过不将任何数据长时间存储在缓冲器中，可对帮助系统减少软错误率(SER)起到作用。这样的实施例中，随着系统启动，用于高速缓冲存储器标记 18 的高速缓冲存储器簇 A 的每个标志被初始化为 A=1，同时高速缓冲存储器标记 18 中的所有簇地址被设为相同的地址(例如

000)，并且用于标记缓冲器 22 的有效标志均被设为“0”。

现在，参照图 7，其中也示出了用于上述初始化程序的状态图，其中，对于第一请求的高速缓冲存储器命中(Ci)，将数据从高速缓冲存储器移到缓冲器 1；对于高速缓冲存储器和缓冲器未中(Di)，将数据从 DRAM 簇 D 移到缓冲器 0。

最后，图 8 为可选数据移动操作的状态图，其中如果在给定时钟周期期间没有接收请求，可执行该操作。由于软错误率(SER)的可能性的缘故，如果没有来自用户(外部装置)的请求，可能不期望保留存储在缓冲中的数据。此情况中，缓冲器 0 中的数据 Di 返回到 DRAM 簇 D，而将缓冲器 1 中的数据 Qi 发送到高速缓冲存储器簇中的一个。

上述可允许状态的“强读式”规则下的数据移动算法的优点在于，通过总是使两个缓冲器均包含有效数据，可在一个时钟周期期间将请求的数据传送到缓冲器中的一个，同时仍将系统保持在可允许状态。如可从上面概述的各种可能性看出，最多仅有四个数据传送操作，并且数据传送逻辑相对容易实现。然而，可将强读式规则概括为其系统实现中的性能、功率以及逻辑门的数目之间的折衷。因此，还提出了“普通式”算法。

简单地说，“普通式”允许缓冲器中有更多可允许状态，由此减少需要的数据传送操作的数目。接下来，这使得装置中的功率耗散较少。另一方面，折衷为：将额外的逻辑用于处理可允许状态中的增加。通过比较的方式，图 9 为图示强读式算法下可允许状态的表格，而图 10(a)-(d)图示了普通式算法下的可允许状态。可以看出，除了包含有效数据的缓冲器 0 和 1 之外，任一缓冲器或两者也可以为空。可将用于普通式的可允许状态的规则总结如下：

可允许状态的规则(普通式)

规则#1 - 可将两个或更少的数据页置于两个缓冲器中。如果每个缓冲碰巧包含有效数据，则每个缓冲器中的数据具有相同的字线地址。然而，如果在先前周期期间做出随机访问请求，则缓冲器中的一个必须包含对应于先前随机访问请求的数据。

规则#2 - 如果任一缓冲器或所有两个缓冲器其中包含任何有效数据页(与具体字线地址相关)，则不存在具有存储在高速缓冲存储器中的相同字线地址的有效数据。

规则#3 - 对于除了规则#2 中描述的一个具体字线地址之外的所有字线地

址，最多有一个与存储在一个且仅为一个的高速缓冲存储器簇中的字线地址相关的有效数据字。也就是说，对于除了和有效标志一起存储在缓冲器标记(图 3)中的字线地址之外的每个字线地址，A 和 B(图 2)不能同时等于 1。

在上述普通式规则下，实现了低功率方法，以与强读式方法相比减少需要的移动数目。例如，在早先讨论的高速缓冲存储器命中情况中(情况 2)，在可允许状态的强读式规则下从 DRAM 簇到缓冲器的数据传送在可允许状态的普通式规则下是不必要的。另外，可允许状态的普通式规则下，某些有效数据字(例如，Di、Qi 以及 Cj)，即早先描述的一些数据移动的开始点，在随机访问请求期间可能不以初始系统状态出现。因此，不再需要到其它缓冲器、和从其它缓冲器的对称移动。

通过普通式规则，为每个特定周期确定需要的数据移动的最小数目。应当注意，在最初仅有一个缓冲器包含有效数据页的情况中，可将该页发送到高速缓冲存储器或 DRAM 簇。然而，在优选实施例中，选择的操作是将数据移到 DRAM 簇。如果改为将数据移到高速缓冲存储器，则随后的具有相同字线地址的 DRAM 簇命中将导致该数据必须从高速缓冲存储器移回到缓冲器。由于 DRAM 簇命中(缓冲器和高速缓冲存储器未中)为基于随机访问请求的统计上的最可能事件，所以其遵循：只要有可能减少移动操作的数目，就应当将数据从缓冲器移到 DRAM 簇。如果 n 表示系统中的 DRAM 簇的总数，而 m 表示每个 DRAM 簇的字线数目，则请求期间的缓冲命中的可能性小于 2/(n*m)，而高速缓冲存储器命中的可能性小于 1/n。相反地，DRAM 簇命中(缓冲器和高速缓冲存储器未中)的可能性约为(n-1)/n。因此，对于随机访问操作，n 和 m 的值越大，则 DRAM 簇命中的可能性越大。可实现普通式下的初始化程序，作为更传统的系统。例如，可通过将用于高速缓冲存储器和缓冲器的所有有效标志设为“0”，而将所有有效数据置入常规的 DRAM 簇。

下面的方法概述了由普通式规则管理的数据传送操作。如果没有接收到随机访问请求，则将 Di 或 Qi 中的一个(如果缓冲器 0 或缓冲器 1 中存在其中的任一个)移回到各自的 DRAM 簇(DRAM 簇 D 或 DRAM 簇 Q)。如果接收了随机访问请求 Xj，则最初在缓冲器 0 中可能有有效 Di、在缓冲器 1 中可能有有效 Qi、以及在高速缓冲存储器中可能有有效 Cj，或所述三者的任意组合，如普通式规则中所概述的。可能的情况如下：

情况 1－缓冲器命中

如同通过强读式规则，j=i，以及 X=D 或 X=Q。也就是说，请求的数据已经通过限定存储在缓冲器 0 或缓冲器 1 中。由于已经满足用于可允许状态的普通式规则，此时钟周期中不实现进一步的数据传送。由图 11(a)的状态图表中缺少变化可反映出此情况。

情况 2－高速缓冲存储器命中

期望将请求的数据 Xj 从一个高速缓冲存储器簇中的其当前位置移到缓冲器 1。如果在任一或全部两个缓冲器中的有任何有效数据，则只要可能，数据便将被优选地移出到对应的 DRAM 簇。和两个缓冲器的状态无关:

Xj 从一个高速缓冲存储器簇(A 或 B)移到缓冲器 1;

现在，如果两个缓冲器均包含有效数据，则进一步的操作为:

Di 从缓冲器 0 移到另一个高速缓冲存储器簇(B 或 A); 以及

Qi 从缓冲器 1 移到 DRAM 簇 Q。

另外，如果两个缓冲器中仅有一个包含有效数据，那么:

Di 从缓冲器 0 移到另一个高速缓冲存储器簇(A 或 B); 或者

Qi 从缓冲器 1 移到另一个高速缓冲存储器簇(B 或 A)。

本质上，如果两个缓冲器最初均不包含有效数据，则除了将 Xj 从高速缓冲存储器移到缓冲器 1 之外，不执行附加操作。图 11(b)和 11(c)演示了上面的系列数据移动。

情况 3a－缓冲未中，高速缓冲存储器未中，j=i，至少一个缓冲器具有有效数据

如果请求的数据既不在缓冲器中也不在高速缓冲存储器中，则其(Xj)在对应的 DRAM 簇中。假定最初至少一个缓冲器具有有效数据，并进一步假定 j=i，如果存在 Di 或 Qi，则 X≠D 及 X≠Q 也为真。因此，可在两个移动中执行遵循的操作，如图 11(d)所示，其中假定存在 Di:

Xj 从 DRAM 簇 X 移到缓冲器 0;

Di 从缓冲器 0 移到 DRAM 簇 D。

情况 3b－缓冲器未中，高速缓冲存储器未中，j≠i，至少一个缓冲器具有有效数据

在此情况中，请求的数据再次位于对应的 DRAM 簇中。然而，请求的数据的字线地址不同于一个或所有两个缓冲器中的数据的字线地址。在可允许状态的普通规则下，可能存在存储于一个高速缓冲存储器簇中的用于行地址

j 的有效 Cj。将首先假定最初 Cj、Di 和 Qi 每个都存在。同样，X≠D 或 X≠Q，或者 X≠D 且 X≠Q 必定为真。如果 X≠D，则实现下面的步骤:

Xj 从 DRAM 簇 X 移到缓冲器 0;

Cj 从一个高速缓冲存储器簇(A 或 B)移到缓冲器 1;

Qi 从缓冲器 1 移到另一个高速缓冲存储器簇(B 或 A);

Di 从缓冲器 0 移到 DRAM 簇 D。

然而，如果 X=D，则 X≠Q，则实现下面的步骤:

Xj 从 DRAM 簇 X 移到缓冲器 0;

Cj 从一个高速缓冲存储器簇(A 或 B)移到缓冲器 1;

Di 从缓冲器 0 移到另一个高速缓冲存储器簇(B 或 A);

Qi 从缓冲器 1 移到 DRAM 簇 Q。

如果 Cj 存在，并且 Di 和 Qi 中仅有一个存在，则由于不存在对应的移动，所以不将这样的移动作为开始点。最终结果仍将遵循普通式规则。

接下来，将假定 Cj 不存在，但 Di 和 Qi 均存在。那么，如果 X≠D，则实现下面的步骤:

Xj 从 DRAM 簇 X 移到缓冲器 0;

Qi 从缓冲器 1 移到任一高速缓冲存储器簇(A 或 B);

Di 从缓冲器 0 移到 DRAM 簇 D。

另外，如果 X=D，则 X≠Q，则实现下面的步骤:

Xj 从 DRAM 簇 X 移到缓冲器 0;

Di 从缓冲器 0 移到任一高速缓冲存储器簇(A 或 B);

Qi 从缓冲器 1 移到 DRAM 簇 Q。

现在，如果 Cj 不存在，并且，缓冲器中仅有一个有效数据页(Di 存在或 Qi 存在，但不都存在)，并且如果 X 不对应于缓冲器数据(X≠D 或 X≠Q)，则实现下面的步骤:

Xj 从 DRAM 簇移到缓冲器 0;

有效缓冲器数据移到对应的 DRAM 簇。

否则，如果缓冲器中的一个有效数据页对应于 X(X=D 或 X=Q)，则实现下面的步骤:

Xj 从 DRAM 簇移到缓冲器 0;

有效缓冲器数据移到高速缓冲存储器簇(B 或 A)。

最后，如果 Cj、Di 或 Qi 均不存在，则执行的操作仅为：将 Xj 移入缓冲器 0。

图 11(e)和 11(f)演示了该系列数据移动。

因此，已经示出了如何将破坏读取的、基于 DRAM 的高速缓冲存储器与破坏读取 DRAM 阵列相结合使用，以减少对该阵列的随机访问周期时间。在其它优点中，本系统提供了可观的面积节约、处理集成中的兼容、以及与诸如使用基于 SRAM 的高速缓冲存储器的系统的其它系统相比减少的软错误。

系统架构的一个具体的关键点包括双簇高速缓冲存储器结构，其中可运行同步的读和写访问操作。另外，该架构还包括两个缓冲器，其被用于对数据传送重定向。高速缓冲存储器标记和缓冲器标记包含与以当前状态存储的数据页的相关的所有信息，由此代表足够的信息，以此做出对下一个时钟周期的数据移动的确定性判定。因此，不需要将历史数据存储在任何标记中。

通过定义可允许状态的概念(如通过强读式规则和普通式规则所示例的)，可设计路径无关算法，以便所有未来的数据移动仅取决于当前状态，而不是当前状态之前的历史。可将任意顺序的连续操作堆积在一起，并由此可无缝地执行所有随机访问。此外，请求的数据在有限数目周期内到达最终状态(也就是说，如果使用建立时间，则请求的数据在一个时钟周期内到达缓冲器，或者如果将一个时钟进程(pipe) 用于标记比较，则请求的数据在两个时钟周期内到达缓冲器)。给定路径无关的属性，以及在有限周期期间完成随机存储请求的事实，则仅存在有限数目的测试情况。因此，DRAM 高速缓冲存储器系统可通过测试台(test bench)设计来完整地校验。

如前所述，强读式规则下的可允许状态为普通式规则下的可允许状态的子集。因此，与强读式规则相结合使用的“对称算法”将通常包括较简单的逻辑，但导致更高的功率消耗。“低功率”算法具有较小的功率耗散，但通常具有更多的逻辑组件，同时有更多与其相关的标记比较时间。然而，应当注意，只要保持路径无关，本发明的实施例还可考虑可允许状态的其它可能规则和相关算法。

进一步考虑到，对于由破坏读取存储器高速缓冲的当前的破坏读取随机访问存储器系统，用作高速缓冲存储器的 DRAM 簇的数目可多于两个。缓冲器数目也可为多于两个。任意附加的高速缓冲存储器簇和缓冲器可与替换架构相结合、或以与核心不同的诸如多循环等待时间的操作配置来使用。对于

使用两倍快速的周期时间的高速缓冲存储器的系统，高速缓冲存储器簇的数目还可减少到一个。如果在其它地方，如在本地 DRAM 阵列中或在全局数据再驱动器(re-driver)提供了锁存功能，则缓冲器可被替换为多路复用器。在不太关心芯片面积的场合，也可将上面的架构和/或算法应用到基于 SRAM 高速缓冲存储器的系统。为使操作在高速缓冲存储器等待时间上有更多余量(margin)、或为了可能的更好的冗余处理、或为了其它性能或定时问题，还可将上面的架构和/或算法应用到基于单端口或双端口 SRAM 高速缓冲存储器的系统。

尽管已通过参照优选实施例描述了本发明，但本领域的技术人员应当理解，可做出各种变化，并且可将其中的元件替换为等价物，而不背离本发明的范围。另外，可做出许多修改，以使具体情况或材料适合于本发明的教导，而不背离其本质范围。因此，有意使本发明不限于作为考虑到实现本发明的最佳模式而公开的具体的实施例，而本发明将包括所有落入所附权利要求的范围中的实施例。

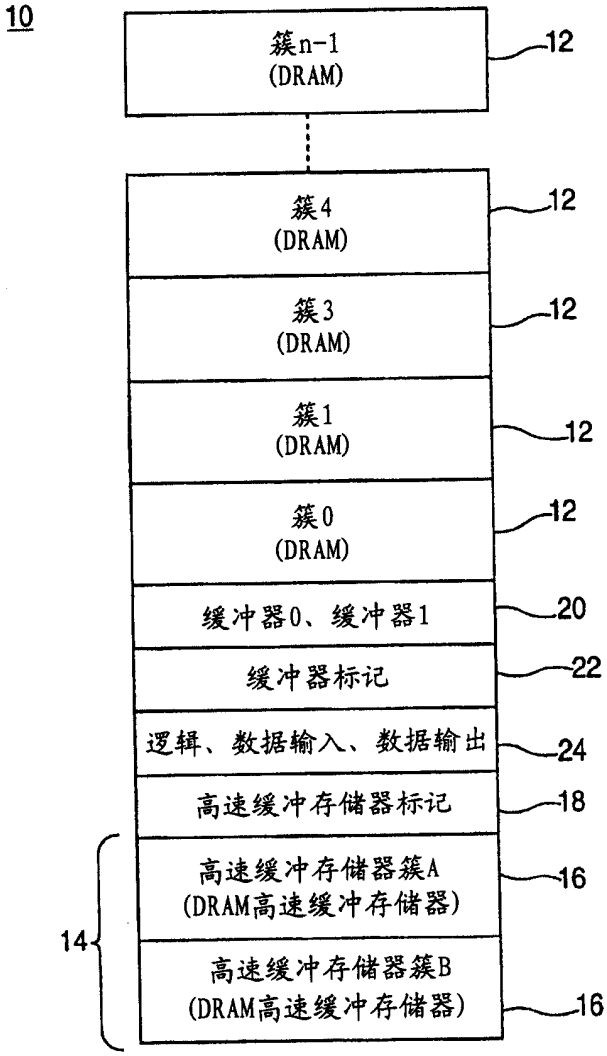

10
簇n-1
(DRAM)
12
簇4
(DRAM)
12
簇3
(DRAM)
12
簇1
(DRAM)
12
簇0
(DRAM)
12
缓冲器0、缓冲器1
20
缓冲器标记
22
逻辑、数据输入、数据输出
24
高速缓冲存储器标记
18
高速缓冲存储器簇A
(DRAM高速缓冲存储器)
16
14
高速缓冲存储器簇B
(DRAM高速缓冲存储器)
16

图 1

18

	簇地址	A	B
字线0	000	1	0
1	010	0	0
2	111	0	1
3	010	1	0
...	...	...	...
m	100	0	1

图 2

22

	簇地址	有效	请求标志	行地址
缓冲0	011	1	1	00101001
缓冲1	101	1	0	

图 3

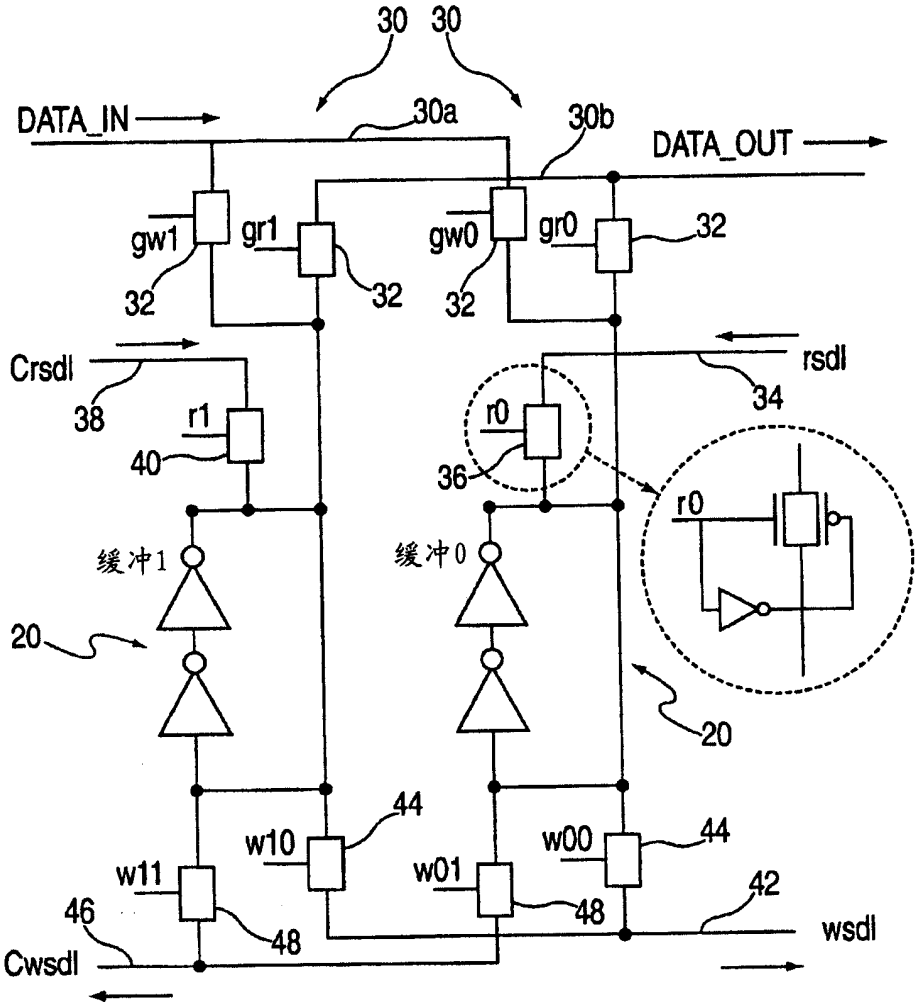

30
30
DATA_IN
30a
30b
DATA_OUT
gw1
gr1
gw0
gr0
32
32
32
32
Crsdl
rsdl
38
34
r1
r0
40
36
r0
缓冲1
缓冲0
20
20
44
44
w10
w00
w11
w01
42
46
48
48
wsdl
Cwsdl

图 4a

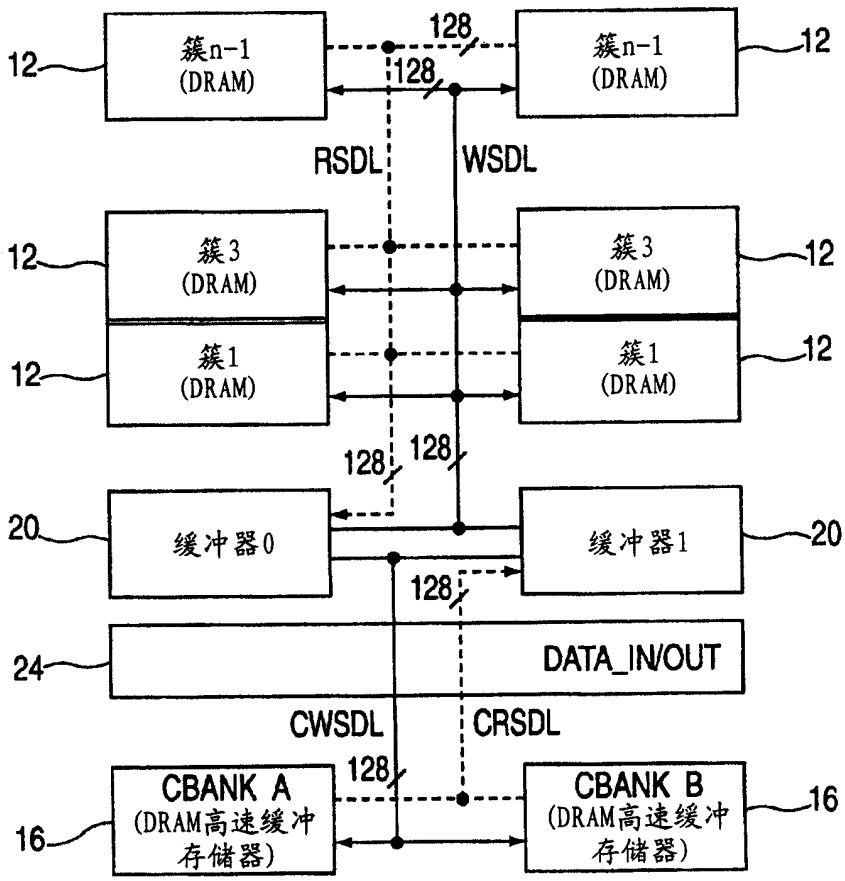

簇n-1
(DRAM)
128
簇n-1
(DRAM)
12
RSDL
WSDL
簇3
(DRAM)
簇1
(DRAM)
20
缓冲器0
缓冲器1
24
DATA_IN/OUT
CWSDL
CRSDL
CBANK A
(DRAM高速缓冲存储器)
CBANK B
(DRAM高速缓冲存储器)
16

图 4b

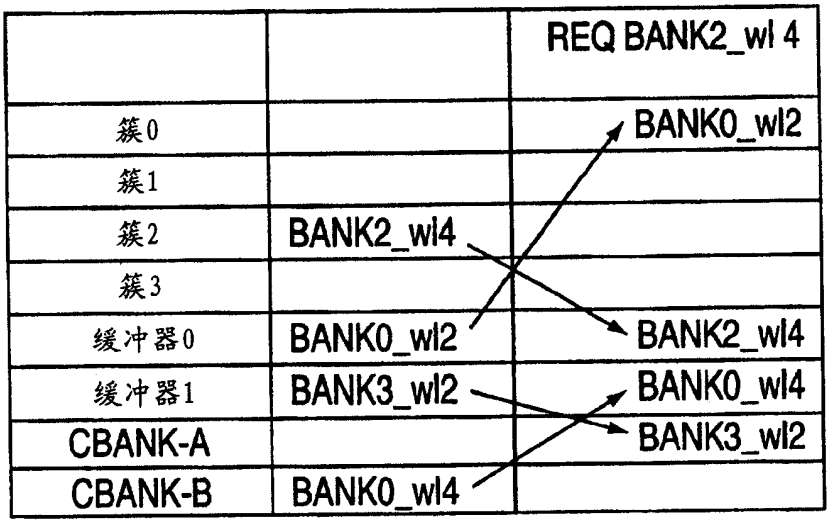

REQ BANK2_wl 4
簇0
簇1
簇2
簇3
缓冲器0
缓冲器1
CBANK-A
CBANK-B
BANK2_wl4
BANK0_wl2
BANK3_wl2
BANK0_wl4
BANK0_wl2
BANK2_wl4
BANK0_wl4
BANK3_wl2

图 5a

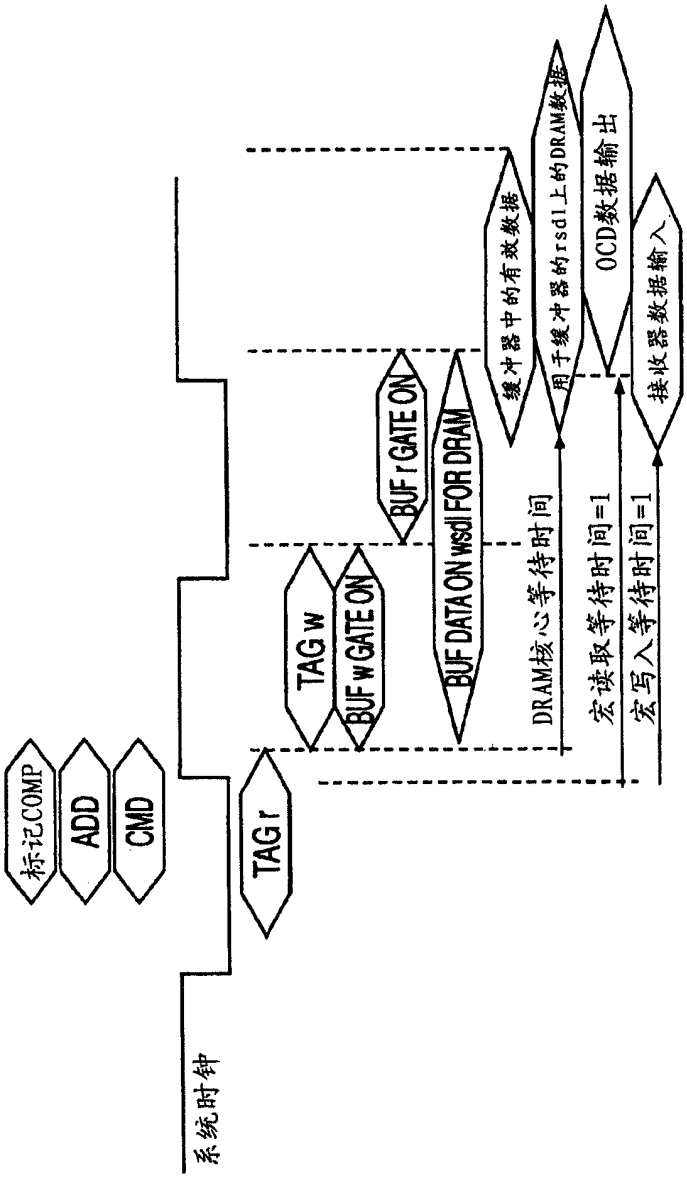

系统时钟
标记COMP
ADD
CMD
TAG r
TAG w
BUF w GATE ON
BUF r GATE ON
BUF DATA ON wsdl FOR DRAM
缓冲器中的有效数据
用于缓冲器的rsdl上的DRAM数据
DRAM核心等待时间
宏读取等待时间=1
宏写入等待时间=1
OCD数据输出
接收器数据输入

图 5b

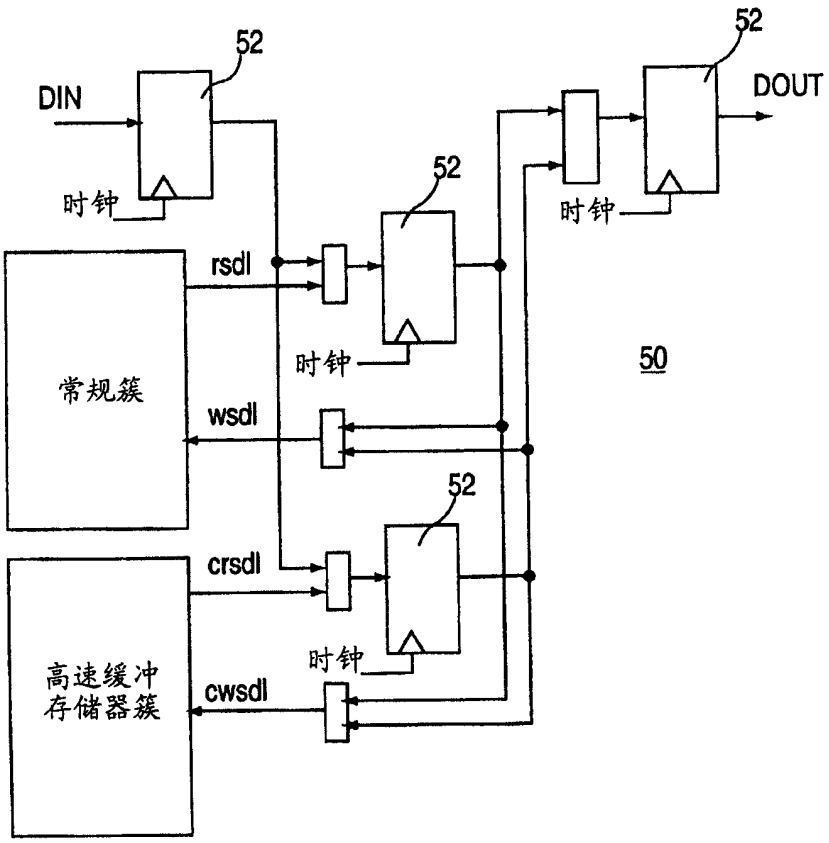

52
DIN
时钟
52
52
DOUT
时钟
rsdl
常规簇
时钟
50
wsdl
52
crsdl
高速缓冲
存储器簇
时钟
cwsdl

图 5c

时钟1	时钟2	时钟3	时钟4
请求1	请求2		
标记r1	标记r2		
标记w1	标记w2		
	CBANK w 1	CBANK w 2	
	CBANK r 1	CBANK r 2	
	簇w1	簇w2	
	簇r1	簇r2	
	DIN 1	DIN 2	
		DOUT 1	DOUT 2

图 5d

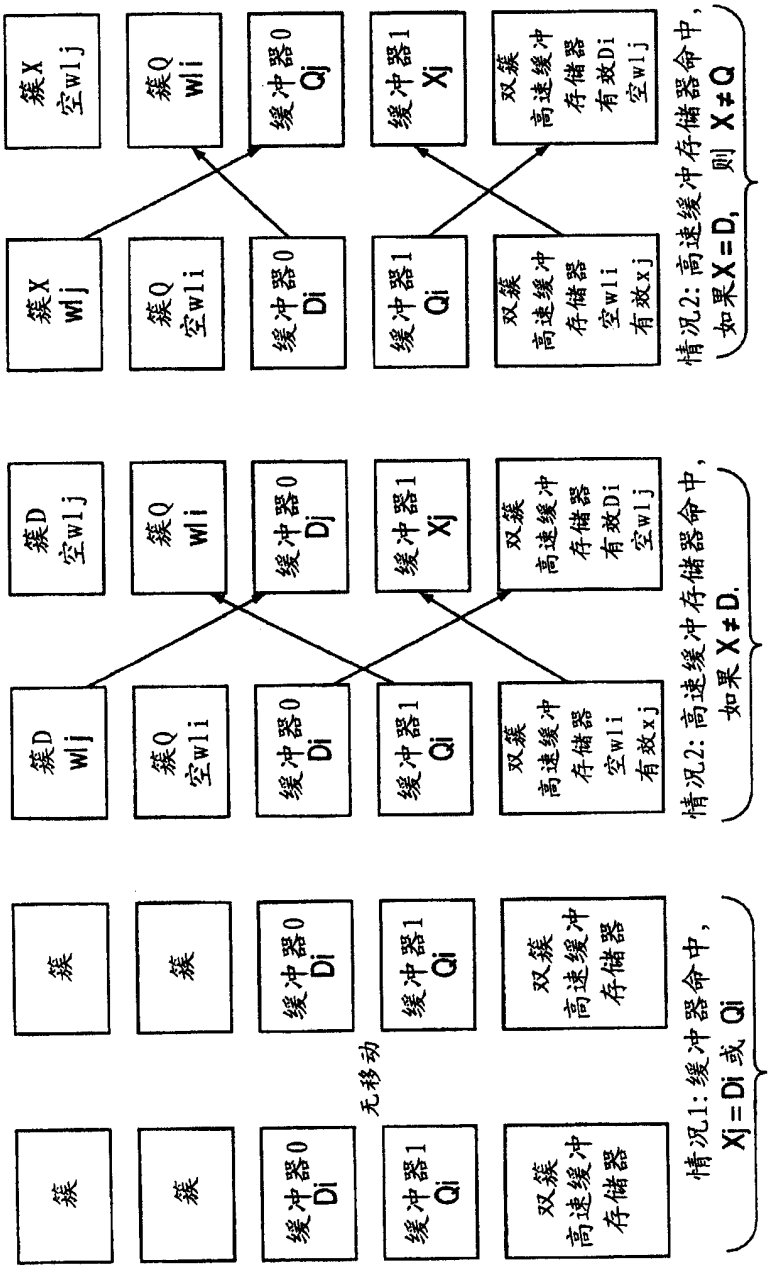

簇
簇
缓冲器0 Di
缓冲器1 Qi
双簇 高速缓冲存储器
无移动
情况1: 缓冲器命中, Xj=Di 或 Qi
簇D wlj
簇Q 空wli
缓冲器0 Di
缓冲器1 Qi
双簇 高速缓冲存储器 空wli 有效xj
簇D 空wlj
簇Q wli
缓冲器0 Dj
缓冲器1 Xj
双簇 高速缓冲存储器 有效Di 空wlj
情况2: 高速缓冲存储器命中, 如果X≠D.
簇X wlj
簇Q 空wli
缓冲器0 Di
缓冲器1 Qi
双簇 高速缓冲存储器 空wli 有效xj
簇X 空wlj
簇Q wli
缓冲器0 Qj
缓冲器1 Xj
双簇 高速缓冲存储器 有效Di 空wlj
情况2: 高速缓冲存储器命中, 如果X=D, 则 X≠Q

图 6a 图 6b 图 6c

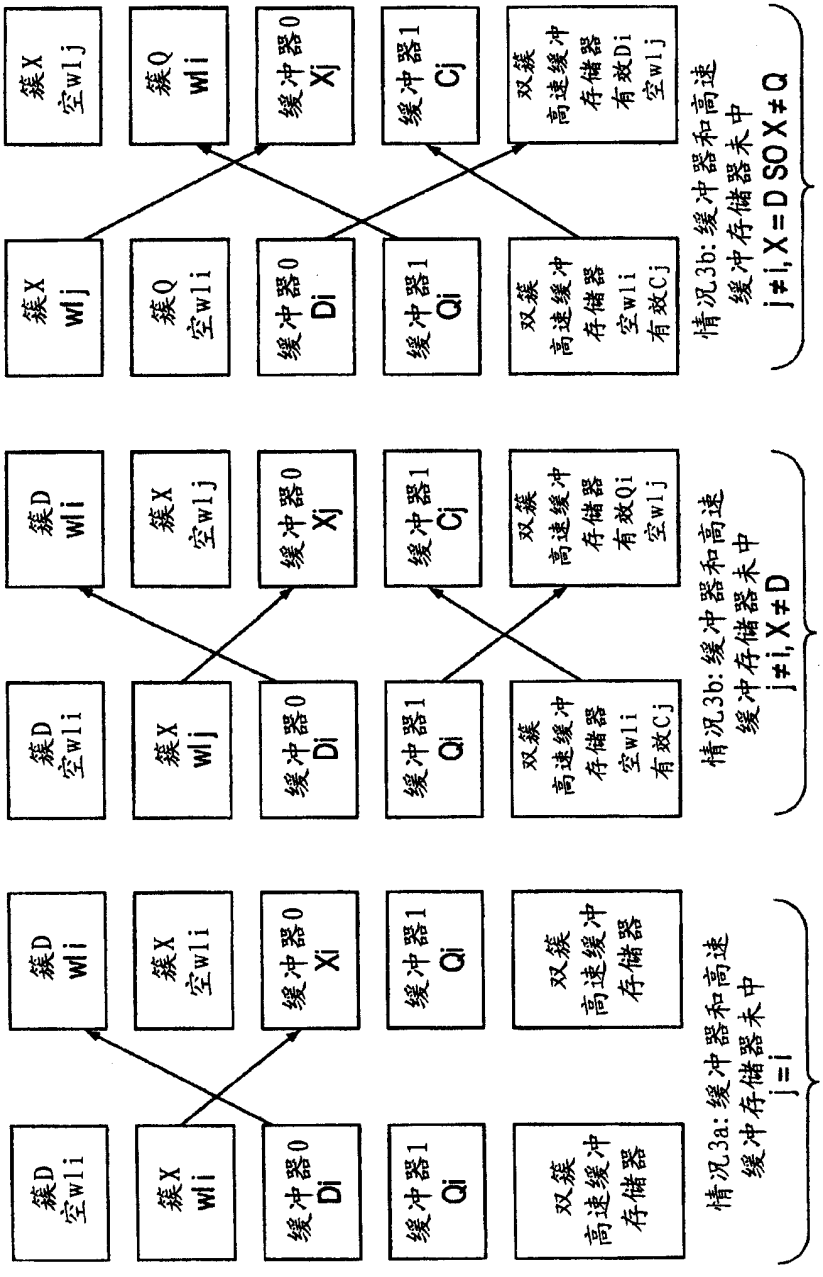

簇D 空w1i
簇X wli
缓冲器0 Di
缓冲器1 Qi
双簇 高速缓冲 存储器
情况3a:缓冲器和高速 缓冲存储器未中 j=i
簇D wli
簇X 空w1i
缓冲器0 Xi
缓冲器1 Qi
双簇 高速缓冲 存储器
簇D 空w1i
簇X wlj
缓冲器0 Di
缓冲器1 Qi
双簇 高速缓冲 存储器 空w1i 有效Cj
情况3b:缓冲器和高速 缓冲存储器未中 j≠i,X≠D
簇D wli
簇X 空w1j
缓冲器0 Xj
缓冲器1 Cj
双簇 高速缓冲 存储器 有效Qi 空w1j
簇X wlj
簇Q 空w1i
缓冲器0 Di
缓冲器1 Qi
双簇 高速缓冲 存储器 空w1i 有效Cj
簇X 空w1j
簇Q wli
缓冲器0 Xj
缓冲器1 Cj
双簇 高速缓冲 存储器 有效Di 空w1j
情况3b:缓冲器和高速 缓冲存储器未中 j≠i,X=DSOX≠Q

图 6d 图 6e 图 6f

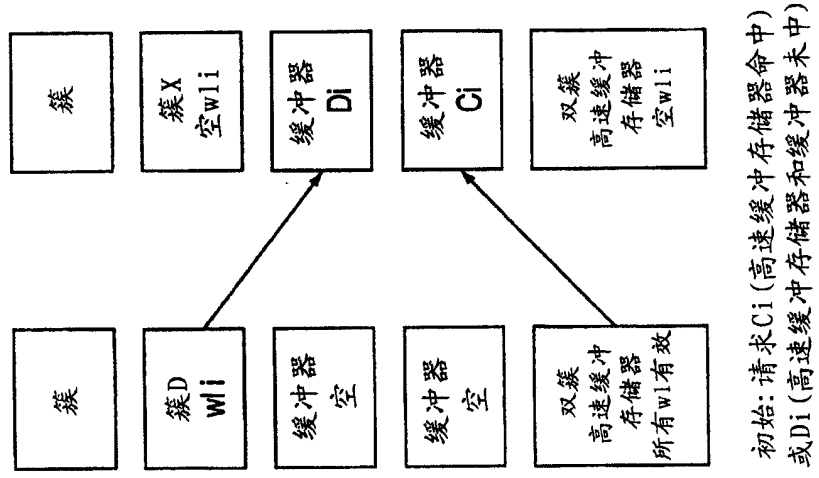

簇
簇D wli
缓冲器 空
缓冲器 空
双簇 高速缓冲 存储器 有效Qi
簇D 空wli
簇
缓冲器 Di
缓冲器 Qi
双簇 高速缓冲 存储器 空wli
恢复情况:无请求
簇
簇X 空wli
缓冲器 Di
缓冲器 Ci
双簇 高速缓冲 存储器 空wli
簇
簇D wli
缓冲器 空
缓冲器 空
双簇 高速缓冲 存储器 所有wli有效
初始:请求Ci(高速缓冲存储器命中)
或Di(高速缓冲存储器和缓冲器未中)

图 8

图 7

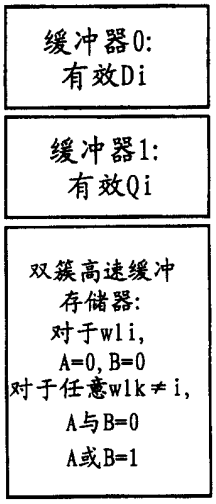


图 9

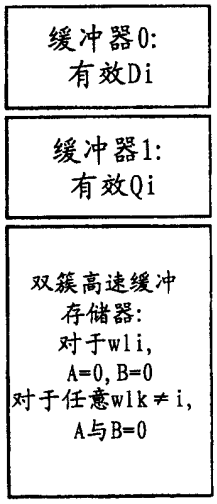


图 10a

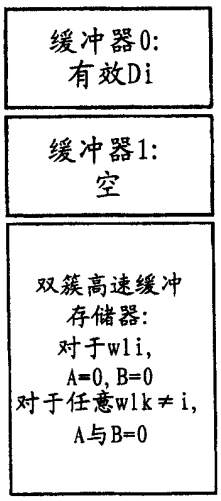


图 10b

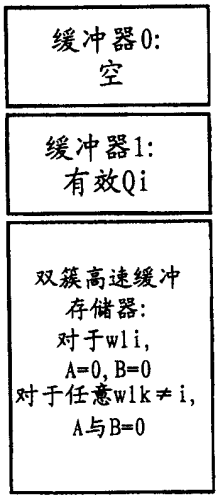


图 10c

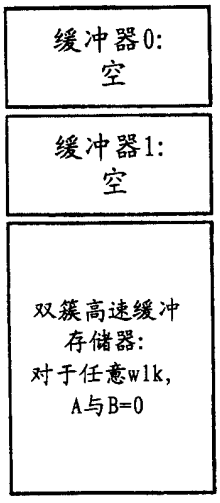


图 10d

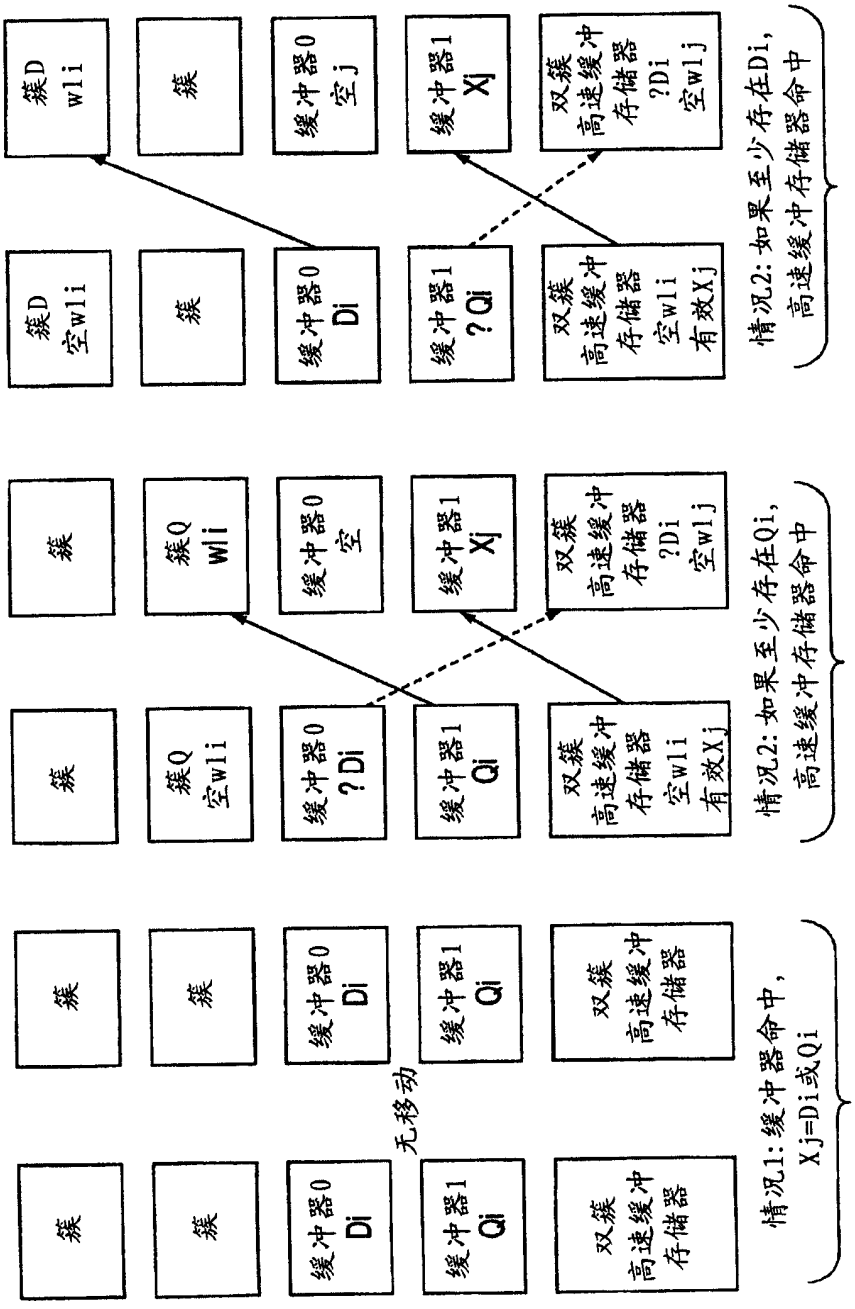


图 11a 图 11b 图 11c

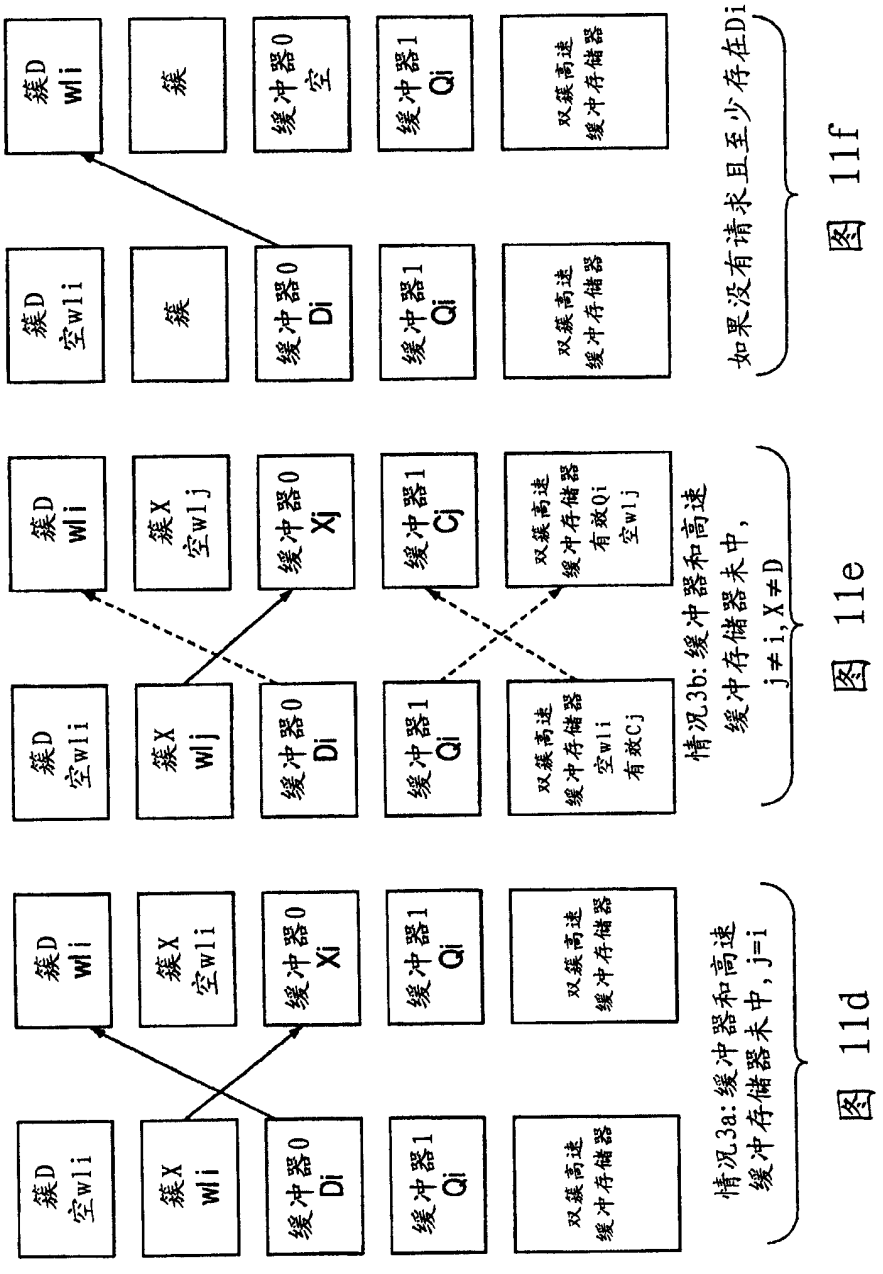


图 11d 图 11e 图 11f