中国科学: 信息科学 2016 年 第 46 卷 第 6 期: 743–762
SCIENTIA SINICA Informationis
《中国科学》杂志社
SCIENCE CHINA PRESS
论 文

三值光学计算机双空间存储器的结构和理论

欧阳山[①②], 彭俊杰[①*], 金翊[①②], 沈云付[①], 刘学民[①], 韩越兴[①], 李卫民[①]

① 上海大学计算机工程与科学学院, 上海 200444
② 上海大学高性能计算中心, 上海 200444
* 通信作者. E-mail: jjie.peng@shu.edu.cn

收稿日期: 2015–02–27; 接受日期: 2015–06–15; 网络出版日期: 2016–05–27
国家自然科学基金青年基金项目 (批准号: 61103054)、上海市自然科学基金项目 (批准号: 15ZR1415400, 13ZR1416000)、上海市教育委员会科研创新项目 (批准号: 13ZZ074, 13YZ005) 和上海高校青年教师培养资助计划项目 (批准号: ZZSD13035) 资助

摘要 三值光学处理器的数据位数高达数千位, 每一位的计算功能都可以按用户要求实时重构, 每一位都可分配给不同的任务使用. 因此, 会有大量数据在三值光学计算机的存储器和光学处理器之间频繁传送. 针对这一新问题, 利用固态盘兼有非易失性和随机访问性的特点, 作者研发了双空间存储系统和内存空间推移技术. 本文详细介绍了双空间存储器的理论、构造、管理和使用方法, 以及内存空间推移技术的硬件结构、技术原理和推移命令. 讨论了利用在双空间存储器中设置的不可闭窗, 实现开机后无等待地接续停机前的工作、启动程序无等待和提高系统安全性的新方法. 文中描述了在 8086 系统中设置双空间存储器的结构和实施内存空间推移技术的仿真实验, 实验结果证实了这个新理论的正确性和相关技术的有效性. 双空间存储器理论和内存空间推移技术不仅满足了三值光学计算机对存储器的需求, 而且为构造基于固态盘的计算机新体系结构提供了理论和技术基础.

关键词 固态盘 双空间存储器 内存推移技术 不可闭窗 三值光学计算机

1 引言

经过 15 年努力, 三值光学计算机走过了从原始思想到完整实物系统雏形的艰难历程. 随着三值光学处理器 [1~12]、三值光学计算机底层监控软件雏形 [13,14] 和编程技术雏形的建立 [15,16], 这种计算机的存储系统成为新的研究对象. 由于三值光学计算机具有数据位数众多、按位可重构、按位可分配等应用特色 [17,18], 它不仅需要容量大且数据通道宽的随机存储器, 而且需要在非易失存储器和处理器之间快速馈送大量数据. 然而在电子计算机中, 随机存储器 (内存) 的容量受到 CPU 地址线根数的限制, 数据通道的位数受到 CPU 数据总线宽度的限制, 非易失存储器 (外存) 和处理器之间馈送数据又受到依靠内存来中转的限制, 这 3 种限制导致电子计算机的存储系统不能满足三值光学计算机的需求. 因此, 如何利用现代存储器件来研发三值光学计算机的存储系统成为一项重要任务.

近年来, 各种无机械部件的存储器不断涌现, 如: 闪存 (flash)、相变存储器 (phase change memory, PCM)、磁阻存储器 (magnetoresistive RAM, MRAM)、铁电存储器 (ferroelectric RAM, FRAM) 和阻变

引用格式: 欧阳山, 彭俊杰, 金翊, 等. 三值光学计算机双空间存储器的结构和理论. 中国科学: 信息科学, 2016, 46: 743–762, doi: 10.1360/N112015-00036

存储器 (resistive RAM, RRAM) 等 [19~23], 不仅出现了物理特性和读写性能各异的新品种, 而且将廉价且成熟的闪存和动态随机存储器结合形成了非易失随机存储芯片 (non-volatile RAM, NVRAM)[1]. 构造 NVRAM 的初衷是将非易失的闪存作为动态存储器的掉电备份. 目前这种芯片被用在超级计算机或长期工作服务器的内存中, 当系统电源出现突发故障时, 这种芯片依靠自带的高性能电容将动态存储器的内容拷贝进闪存, 而当系统电源恢复后, 再从闪存中恢复动态存储器的内容, 以避免当前任务因系统临时掉电而异常终止[2]. 2014 年 AgigA 公司已经生产出随机访问性能到达 DDR4 水平的双列直插式内存条 NVDIMM (non-volatile dual-inline-memory-modules) [3], 目前中国无锡云动科技发展公司也推出了国产 NVDIMM 内存条. 然而, 也可以将 NVRAM 中的动态存储器看作闪存的读写缓冲器, 本文将用这一观点看待 NVRAM, 并藉此阐述双空间存储器理论. 总之, 闪存和动态存储器结合, 使 NVRAM 芯片既有足够好的读写速度和读写次数, 又有非易失性, 成为性能卓越的复合型固态盘.

集可随机访问和非易失两大特点于一身的固态盘, 既能用作内存储器, 又能用作外存储器, 它提供了 CPU 随机访问非易失存储器的物质条件. 如闪存既用作嵌入式系统的内存 —— 只读存储器, 又用作常见的外存储器 —— U 盘. 近年来, 对固态盘的研究不仅涉及各种存储器件的物理原理, 而且涉及到固态盘的多种应用, 其研究热点是用固态盘对现有的存储系统进行多方面的优化, 以提高整体性能, 而未见到用固态盘构造全新存储系统的研究. 本文第 7 节概括地介绍了当前的几个典型研究, 并简略地说明了它们与本文所述双空间存储系统的关系.

2013 年, 作者同时利用固态盘的非易失性和随机可访问性, 构造了一种双空间存储系统, 其中以字或字节为存储单位的空间由 CPU 通过内存空间随机访问, 称为字空间; 而以数千个字节为单位的空间由 CPU 通过外存储器管理系统顺序访问, 称为块空间 [24]. 将双空间存储器用于三值光学计算机时, 众位数的三值光学处理器可以直接和非易失存储单元随机交换数据, 从而加快非易失存储器和处理器之间的数据馈送速度. 本文介绍这种存储系统的基本理论、基本结构、基本使用方法和实验情况, 并阐述这种双空间存储系统对电子计算机系统结构的影响. 但本文不讨论固态盘本身的任何技术和问题.

2 三值光学计算机存储系统的基本特性和现状

三值光学处理器的数据位数众多、每一位的计算功能均能随时重构成二值逻辑运算器或三值逻辑运算器或无进位加 (减) 法器之一、每一位均可分配给不同的任务使用, 使得这种计算机特别适合处理大批量数据或简单结构型数据或多任务共享 [17]. 三值光学计算机的这些特征要求它的存储系统要有 3 个基本特性:

(1) 面向处理器的数据宽度与处理器的数据位数相同, 目前要达到数千位, 将来要达到数万位;

(2) 有足够大的随机访问存储器, 达到数百 GB 或 TB 量级;

(3) 在非易失存储器和处理器之间高速交换数据.

显然, 以电子计算机中常用的存储系统为基础来开发三值光学计算机需要的这种新存储系统是最可行的研究途径. 2003 年以来, 三值光学计算机研究者在致力于构建众位数处理器 [13] 的同时, 也在不断考虑如何构建三值光学计算机的存储系统, 至 2009 年前后给出了利用电子计算机的存储系统来

1) Non-Volatile RAM. http://www.agigatech.com/Technology/Non-Volatile RAM.

2) Agiga Tech to Demonstrate NVDIMM Technology at 2014 Flash Memory Summit. http://www.agigatech.com/Press Room/Press Releases.

3) Agiga RAM DDR4 NVDIMM. http://www.agigatech.com/Products/DDR4.

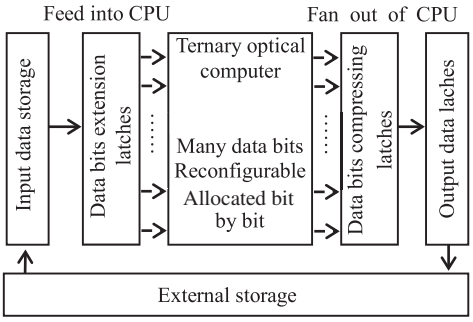

图 1 三值光学计算机现用存储系统

Figure 1 Present storage system of ternary optical computer

构造三值光学计算机存储系统的基本思想 [14, 25], 这个思想的基本依据是: 鉴于三值光学处理器有别于电子计算机, 其主要特征为数据位数众多、按位可重构和按位可分配, 而其萌发期和初期的运行速度会远远低于半导体存储器的读写速度, 于是在这个时期可以用顺序读写多个半导体存储单元的方式来模拟一次读写很多位的三值光学计算机存储器. 因此在三值光学计算机的各实验系统中一直用电子计算机的存储系统来替代三值光学计算机的存储系统, 文献 [13] 在描述三值光学计算机众多数据位的实现技术时, 刻画出了三值光学计算机现用的存储系统概貌, 如图 1 所示.

文献 [13] 指出: 三值光学计算机的数据位在输入数据存储器和输出数据存储器中的物理形态都是 "多个字节的半导体存储单元", 但在三值光学处理器中是 "数千个三态信号光束或数千个液晶像素位". 为了将数据位在 "多个字节" 形式和 "数千个位" 形式之间转换, 三值光学处理器的数据输入端设置了由多个常规锁存器组成的位数扩展锁存器, 输出端设置了位数压缩锁存器. 在位数扩展锁存器中, 每个常规锁存器输入端的数据宽度都与馈入 CPU (电子计算机) 的数据宽度一致, 它们各自接收来自输入数据存储器 (馈入 CPU 的内存) 的一个数据馈入, 于是整个位数扩展锁存器连续接收了多个数据馈入; 而所有常规锁存器的输出端依次排序, 在同一个信号的控制下同步输出, 形成数千位数据宽度的位数扩展锁存器输出端. 这个输出端能够满足三值光学处理器的数据宽度要求. 位数压缩锁存器的工作原理与位数扩展锁存器正好相反, 这里不再赘述. 在输入数据存储器和位数扩展锁存器之间、位数压缩锁存器和输出数据存储器之间传送数据时, 虽然 "传送过程仍然是以字节为单位, 但对三值光学计算机而言, 这是以 n 位为单位的一次数据传送, 只是采用了以电子计算机数据宽度为单位的多次相继传送方式而已".

显然, 当三值光学处理器速度较慢时, 用一个馈入 (馈出) CPU 将多个内存单元的数据连续送入 (读出) 位数扩展 (压缩) 锁存器即可. 当三值光学处理器的速度提高后, 只需设置多对位数扩展锁存器和位数压缩锁存器, 每个位数扩展 (压缩) 锁存器都有一个馈入 (馈出) CPU 为其送入 (读出) 数据, 而各对位数扩展锁存器和位数压缩锁存器轮流与三值光学处理器连接, 并行交换千位数据即可. 正如文献 [25] 附图中所述的 "输入存储阵列 1−n" 和 "输出存储阵列 1−n" 所示. 位数扩展锁存器和位数压缩锁存器解决了输入/输出数据存储器和三值光学处理器之间的数据宽度转换问题, 只需将位数扩展锁存器和位数压缩锁存器作为未来存储系统面向三值光学处理器的接口, 就实现了三值光学计算机存储器的第一个基本特性.

然而, 馈入 (馈出) CPU 的内存容量决定于 CPU 内存地址线的根数, 这限制了图 1 中输入 (输出) 数据存储器的容量, 从而限制了三值光学计算机随机访问存储器的容量; 同时, 数据进入位数扩展锁存器要先经过从外存拷贝进输入存储器 (馈入 CPU 的内存) 的过程, 而退出位数压缩锁存器又要经过

从输出存储器 (馈出 CPU 的内存) 拷贝进外存的过程, 这两个过程严重阻碍了三值光学处理器和非易失性存储器交换数据的速度. 因此, 图 1 所示的存储器结构不能实现三值光学计算机存储器的第 2 个和第 3 个基本特性. 为此, 我们再次考察计算机存储系统的基本需求, 来寻找解决这两个问题的思路.

从计算机原理看, 处理器需要能按地址总线上随机给出的地址值, 对相应的存储单元进行读操作和写操作 —— 随机访问性; 同时计算机系统又需要所保存的信息在掉电时不丢失 —— 非易失性. 20 世纪 30 年代, 没有找到同时具有可随机访问性和非易失性的合适材料或器件, 于是早期的计算机先用磁泡, 后用半导体存储器来满足处理器对随机访问性的要求, 形成了内存储器; 而先用磁带, 后用磁盘做外存, 来满足计算机系统对非易失性的要求, 形成了外存储器; 在处理器使用数据前, 将数据从外存拷贝进内存; 在处理器用完数据后, 再将修改过的数据从内存拷贝回外存. 由此引发了内存重定位和虚拟内存等著名的内/外存映射技术, 同时也形成了内存墙. 早期存储器件的单一特性导致电子计算机的存储系统至今仍分为内存储器和外存储器.

近 30 年, 固态盘技术和相关材料发展迅速, 尤其是闪存和动态存储器结合而成的非易失性动态存储器提供了同时具有可随机访问性和非易失性的存储器件, 用这种器件就可以将内存空间和外存空间同时构建在一个存储实体上, 形成双空间存储系统. 2013 年作者建立了双空间存储器结构和内存空间在双空间存储器上的推移技术, 从理论和实验上证明双空间存储器能够实现三值光学计算机存储器的第 2 个和第 3 个基本特性, 本文重点讨论这一理论.

3 双空间存储器和内存空间推移技术的基本思想

构造双空间存储器的物质基础是同时具有随机访问性和非易失性的存储器件, 但对这个器件的工作原理、材质、物理本性和器件结构并无要求. 下文中统称具有随机访问性和非易失性的存储器件为随机固态盘. 显然, NVRAM 就是一种随机固态盘, 而且是目前构造双空间存储器的较好选择之一.

构造双空间存储器的基本思想可以叙述为: 用随机固态盘的非易失性构造块空间, 而用其随机访问性构造字空间. 块空间以包含许多字节的数据块为访问单位, 如果选取块空间的块大小与外存空间的块大小成整倍数, 则块空间就可以直接用作外存空间, 有关外存空间的各种技术和管理策略都在块空间上有效. 字空间则以字或字节为访问单位, 如果选取字空间访问单位与处理器内存空间的数据总线宽度一致, 再使用某种技术将内存空间地址映射到大很多的字空间地址的某个区段上, 处理器就能够透过内存空间而直接访问双空间存储器字空间的这个区段. 由于字空间和块空间落实于同一个存储实体 —— 双空间存储器, 于是处理器就在事实上透过内存空间, 在字空间的区段中直接访问了块空间 (外存) 上的数据, 因此双空间存储器能够避免数据在内存和外存之间的拷贝过程.

块空间可以直接用作外存储器, 因此双空间存储器理论在块空间方面能够全面沿用当前的外存储器技术. 虽然字空间的容量远远大于内存空间的容量, 但这两个空间的存储器构造技术却完全相同, 只需增加存储器的地址线数目并相应地增加存储体内的地址译码电路, 就可沿用构造内存储器的技术构造出字空间存储器. 但是, 字空间的地址线数目多于内存空间的地址线数目, 必须开发出一项新技术, 将内存空间自动映射到字空间的一个区段, 而且能够很方便地将这种映射在整个字空间上移动, 才能实现处理器透过内存空间直接访问整个字空间的目标. 作者从 2013 年 6 月至今已经成功研发出了这一技术, 称其为 “内存空间推移技术”, 简称 “内存推移”. 这项技术也是本文讨论的主要问题之一.

如果用双空间存储系统制作三值光学计算机的存储器, 则图 1 成为图 2. 在这个新存储系统中, 块空间直接用于外存空间, 而字空间采用内存推移技术同馈入 (馈出) CPU 的内存空间对接. 于是, 通过块空间 (即外存空间) 存入双空间存储器上的数据, 可以通过字空间被随机访问, 从而使非易失存储器

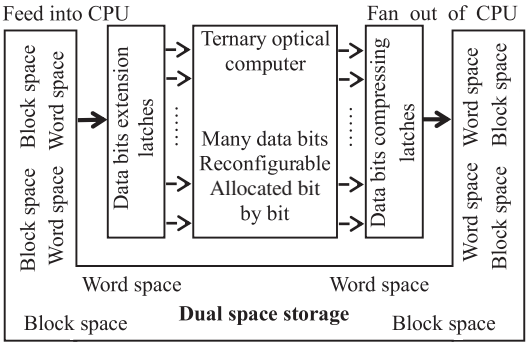

图 2 基于双空间存储系统的三值光学计算机存储系统

Figure 2 Storage system of ternary optical computer based on dual space storage

上的数据与三值光学处理器之间直接进行馈入和馈出, 提高数据的整体馈送速度.

从双空间存储器的概念可以看出: 虽然双空间存储系统的需求来自三值光学计算机存储器的需要, 但双空间存储系统却完全不依赖于三值光学处理器, 而只与字空间和块空间的地址对应关系有关、与字空间地址和内存空间地址的映射有关. 字空间和块空间的地址对应关系决定于双空间存储器的构造, 与处理器无关; 字空间和内存空间的地址映射只涉及内存地址 (即物理地址) 和字空间地址的变换, 这种变换发生在处理器输出物理地址之后, 因此与处理器如何产生物理地址无关, 于是与虚拟内存技术、内存分段管理技术和内存分页管理技术无关, 事实上采用双空间存储器后就没有了数据在内存与外存之间的拷贝过程, 因此所有内存重定位技术都不再需要, 这为简化 CPU 结构和内存管理技术奠定了基础.

双空间存储器与处理器无关, 因此也可以将这种存储器用于电子计算机, 从而构造出基于固态盘的新型电子计算机体系结构. 在这种新结构的电子计算机中, 不需要对 CPU 做任何改动 (注: 如果重新设计 CPU 可以删去内存管理机制, 但考虑到程序还需要分段, 故段寄存器还应保留), 只需要依据内存推移技术构造新的主板和存储器就可消除数据在内存和外存之间的拷贝过程、消除开机和调用程序时的等待时间、为系统安全增加新的保护手段等, 本文也将对双空间存储器带给电子计算机的影响做适度讨论.

鉴于双空间存储器概念、理论和技术都与处理器无关, 下文中将不再区分三值光学处理器或电子处理器, 而统一用 CPU 来代表处理器.

4 构造双空间存储系统的关键问题和解决技术

构造双空间存储系统的关键在于建立字空间存储器的技术、字空间地址和块空间地址的转换技术、字空间管理技术和字空间使用技术, 而字空间使用技术的核心就是内存空间在字空间上的映射和移动技术 —— 内存推移技术. 于是, 构建双空间存储系统必须解决的关键问题可以概括为:

(1) 字空间地址线和块空间地址线的对接问题;

(2) 字空间地址线和内存空间地址线的对接问题;

(3) 内存空间在字空间上的移动问题;

(4) 对字空间使用状况的管理问题.

结合图 3 所示的双空间存储器和内存推移技术的硬件结构, 可以概述本文所述理论和技术解决这

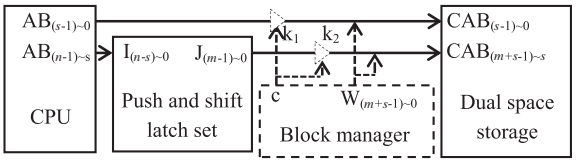

图 3　双空间存储器地址线配置示意

Figure 3　Schematic of address bus of dual space storage

4 个关键问题的技术要点为:

解决问题 (1) 的要点在于, 图 3 中合理配置了双空间存储器字空间地址线和块空间地址线的对应关系; 解决问题 (2) 的要点在于, 图 3 中增设了一个推移锁存器组, 其中的每个推移锁存器都能将内存地址线的数条高序号线自动变换为字空间地址线中较多的高序号线; 解决问题 (3) 的要点在于, 通过修改推移锁存器的值, 可以把内存空间在整个字空间上任意移动, 称这种移动为推移操作; 解决问题 (4) 的要点在于, 该项技术仿照文件管理表和内存管理表技术, 分别设立了记录双空间存储器各部分使用状况的窗壁管理表和记录内存空间与字空间对应关系的窗框管理表. 下文逐一讨论解决这 4 个关键问题的理论和技术细节. 从图 3 可以看到双空间存储系统与当前的存储系统在概念上有下列不同:

(1) 当前的存储系统分为两个存储实体, 一个是容量较小但可快速随机访问的内存储器, 另一个是容量很大但不能随机访问的外存储器; 而图 3 中仅有一个存储实体, 它是由随机固态盘构成的容量很大且可快速随机访问的双空间存储器.

(2) 当前存储系统的内存和外存之间由内存重定位技术管理数据在二者之间的拷贝操作; 而图 3 中完全没有数据拷贝操作, 故不需要这类技术.

(3) 当前存储系统的块管理器 (外存管理器) 输出的地址线连接到外存储器上, 与内存储器无关联, 而 CPU 输出的内存地址线连接到内存储器上, 与外存储器无关联; 但图 3 中的块管理器输出的地址线则与 CPU 输出的内存地址线共同接在双空间存储器的地址线上.

(4) 当前存储系统中, CPU 输出的内存地址线一一对应地连接到内存地址线; 而图 3 中 CPU 输出的内存地址线仅有低位部分的地址线一一对应地连接到双空间存储器的低位地址线, 而 CPU 输出地址线的高位部分则接入推移锁存器组, 再由推移锁存器组输出更多的双空间存储器的高位地址线.

4.1　字空间地址线和块空间地址线的对接

双空间存储系统的块空间和字空间分别与目前的外存空间和内存空间对应, 而在当前的电子计算机中, 块管理器是外存管理器的核心部件, 常采用直接存储器传送 (DMA) 方式产生块内的顺序地址, 并将其送上外存储器的块内地址线 (外存储器的低序号地址线), 又往往将 DMA 管理器安排在 I/O 空间, CPU 通过 I/O 空间给 DMA 控制器送入本次传送的外存起始地址和传送的数据量, 由此将外存置于 I/O 空间中, 其特征是 CPU 不产生外存的块内地址、而外存的地址线与 CPU 的内存空间地址线没有连接.

然而双空间存储器的字空间和块空间落实在同一个存储实体上, 因此这两个空间的地址线必然有连接关系, 这种连接关系决定了字空间地址值和块空间地址值的互换规律. 依靠地址值的互换规律, 由外存空间 (块地址) 存储在这个存储器上的数据, 可以通过将内存空间地址映射到字空间地址后进行随机读写, 这就是双空间存储器能够消除数据在内存和外存之间拷贝的根源. 显然, 最简单的连接方式是这两个空间的地址线共用一组物理连线, 这时字空间地址值就与块空间地址值完全相同, 统称其为双空间存储器地址线和地址值. 图 3 中的双空间存储器块空间地址线和字空间地址线正是采用了这

种连接结构, 故它只有一组地址线 $\mathrm{CAB}_{(m+s-1)\sim 0}$, 称其为双空间地址线. 于是图 3 中的双空间存储器的每一个地址都是一个存储单元的唯一地址, 既是该单元的块空间地址, 也是该单元的字空间地址.

图 3 中双空间存储器的块空间地址线和字空间地址线采用共线结构, 简化了双空间存储器的制作, 但这组地址线必须分时与内存空间地址线或外存空间地址线对接才能适合计算机系统使用. 为此, 图 3 中在逻辑上给块管理器增设了输出信号 c, 它控制三态门 k_1 和 k_2 的通断, 当系统使用双空间存储器的块空间时, CPU 启动块管理器, 这时 c 信号就关断 k_1 和 k_2, 使 CPU 的内存地址总线 $\mathrm{AB}_{(n-1)\sim 0}$ 与双空间地址线 $\mathrm{CAB}_{(m+s-1)\sim 0}$ 断开, 同时由块管理器输出块地址 $\mathrm{W}_{(m+s-1)\sim 0}$ 到双空间地址线上, 这时双空间存储器就以块为单位受块管理器的控制, 与外存无异, 因此双空间存储器成为当前的外存. 当系统通过内存空间地址使用双空间存储器时, CPU 停止块管理器工作, 则 c 和 $\mathrm{W}_{(m+s-1)\sim 0}$ 均无输出, 这时 c=0, 使 k_1 和 k_2 的导通, 于是 CPU 的内存空间地址线通过推移锁存器变换后与双空间地址线 $\mathrm{CAB}_{(m+s-1)\sim 0}$ 相连, 这时 CPU 能够通过内存地址以字为单位随机访问双空间存储器, 双空间存储器成为当前的内存空间. 由此可见, k_1 和 k_2 在 c 的控制下实现 CPU 内存地址线及推移锁存器的输出线和块管理器输出的地址线分时与双空间地址线 $\mathrm{CAB}_{(m+s-1)\sim 0}$ 连接, 在逻辑上使双空间存储器的块空间地址线与字空间地址线分离. 二者在逻辑上的分离保证了当前使用的外存技术都可以直接用于块空间.

再考虑到 CPU 都有分时使用内存和外存的机制, 图 3 中的 c, k_1 和 k_2 的逻辑功能就可以由 CPU 的这一机制来承担, 因此实际系统中往往不必设置 c, k_1 和 k_2.

4.2 双空间地址线和内存空间地址线的对接

双空间存储器的容量远远大于 CPU 的内存容量, 意味着双空间地址线的数目多于 CPU 内存地址线的数目. 因此, 必须将内存地址线自动与双空间地址线的一部分相接, 并且能够方便地自动改变二者的连接关系, 才能实现 CPU 对双空间存储器字单元的随机访问. 为此, 图 3 中设置了一组推移锁存器, 并把 CPU 输出的内存地址线分成两组, 低编号的 s 条内存地址线 $\mathrm{AB}_{(s-1)\sim 0}$ 与双空间地址线的低 s 条 $\mathrm{CAB}_{(s-1)\sim 0}$ 一对一直接连接, 但高编号的 $n-s$ 条内存地址线 $\mathrm{AB}_{(n-1)\sim s}$ 接入推移锁存器组的锁存器选择端 $\mathrm{I}_{(n-s)\sim 0}$, 用于在 2^{n-s} 个推移锁存器中选择一个, 被选中的推移锁存器会输出 m 根信号线 $\mathrm{J}_{(m-1)\sim 0}$, 这 m 根信号线与双空间地址线高 m 条 $\mathrm{CAB}_{(m+s-1)\sim s}$ 一对一连接, 成为双空间地址线的高 m 根.

当 CPU 输出一个 n 位的内存地址后, 其低 s 位由 $\mathrm{AB}_{(s-1)\sim 0}$ 线进入 $\mathrm{CAB}_{(s-1)\sim 0}$, 形成双空间地址的低 s 位; 而该内存地址的高 $n-s$ 位则由 $\mathrm{AB}_{(n-1)\sim s}$ 线进入推移锁存器组的输入端 $\mathrm{I}_{(n-s)\sim 0}$ 线, 该 $n-s$ 位内存高位地址将选择一个推移锁存器, 被选中的推移锁存器将其保存的 m 位数据从推移锁存器组的输出端 $\mathrm{J}_{(m-1)\sim 0}$ 送上双空间存储器的高 m 位地址线 $\mathrm{CAB}_{(m+s-1)\sim s}$, 形成双空间地址的高 m 位. 于是推移锁存器中的 m 位数据和 CPU 送出的内存地址中的低 s 位拼接成双空间存储器的当前地址值. 当 m 大于 $n-s$ 时, 双空间地址线的数目就比内存空间地址线多出 $m-n+s$ 条, 于是双空间存储器的容量就是内存空间容量的 2^{m-n+s} 倍.

例 1 奔腾处理器有 32 位内存地址线 $\mathrm{AB}_{31\sim 0}$, 若将其低序号的 20 位地址线 $\mathrm{AB}_{19\sim 0}$ 与双空间地址线 $\mathrm{CAB}_{19\sim 0}$ 相连, 而将 $\mathrm{AB}_{31\sim 20}$ 接入推移锁存器组的输入端 $\mathrm{I}_{11\sim 0}$, 即知系统有 4096 个推移锁存器, 再设每个推移锁存器有 32 位, 则推移锁存器组的输出端有 32 根线, 记为 $\mathrm{J}_{31\sim 0}$, $\mathrm{J}_{31\sim 0}$ 连接双空间地址线的 32 根高序号线 $\mathrm{CAB}_{51\sim 20}$, 于是双空间存储器有 52 根地址线, 其容量为 2^{52} 字节, 即达到 4 PB, 为 CPU 内存容量的 2^{20} 倍.

4.3 内存空间在字空间上的推移

根据局部性原则, CPU 几乎总是连续访问内存空间的一段相近地址, 即内存地址线高序号部分 $AB_{(n-1)\sim s}$ 的值很少改变, 因此很少改换当前选择的推移锁存器, 于是双空间存储器的高序号地址线的值也会保持不变, 这意味着 CPU 几乎总是通过地址线 $AB_{(s-1)\sim 0}$ 直接访问双空间地址线 $CAB_{(s-1)\sim 0}$ 所对应的区域, 故称 $AB_{(s-1)\sim 0}$ 对应的内存空间区域为映射窗, 称 $CAB_{(s-1)\sim 0}$ 对应的双空间存储器区域为窗壁. 另一方面, 每一个推移锁存器保存的值 (也是其输出的值) 给出了双空间地址的一个高序号部分 $CAB_{(m+s-1)\sim s}$, 显然, $CAB_{(m+s-1)\sim s}$ 的每个值都是且仅是一个窗壁中所有双空间地址的高位部分, 于是也用这个值作为该窗壁的编号, 即推移锁存器的值指定了一个窗壁, 给这个推移锁存器的值后面补 s 个 0 就得到这个窗壁的首地址, 称这个地址为窗位. 为便于讨论, 我们可以想象每个推移锁存器对应有一个与窗壁等大的虚拟空间, 称这个虚拟空间为窗框, 每个推移锁存器的值指出了它的窗框目前所在的窗壁.

窗壁、窗框和映射窗的容量都相同, 称这个容量为窗幅, 三者的关系为: 窗壁是双空间存储器的区域, 双空间存储器的每一个存储单元都在且仅在一个窗壁中; 窗框是推移锁存器隐含的虚拟存储空间, 窗框与推移锁存器一一对应, 所以窗框的数量远远少于窗壁的数量, 而所有窗框容量的总和等于内存空间的容量; 映射窗是内存空间的区域, 系统中仅有一个. 每个推移锁存器的值将它对应的窗框定位在一个窗壁上, CPU 使用高序号内存地址线选中一个推移锁存器时, 映射窗就自动与对应的窗框重合, 于是 CPU 透过映射窗随机访问该窗框所在的窗壁.

例 2 在例 1 设定的系统中, 有 4096 个窗框, 4294967296 个窗壁, 窗幅 1 MB.

由于 CPU 给出一个内存地址时, 该地址的高序号部分一定会选择一个推移锁存器, 所以 CPU 不可能访问没有窗框的窗壁. 鉴于此, 称放置了窗框的窗壁为开窗, 而没有放置窗框的窗壁为闭窗, 窗框不能被移走的窗壁为不可闭窗. 显然, 任何时候 CPU 都能立刻访问不可闭窗, 因此不可闭窗中的内容相当于目前常驻内存的内容.

当 CPU 输出的内存地址改变了高序号部分时, 就将映射窗移动到另一个窗框上, 因此, 映射窗会随着内存地址高位部分的改变在各个窗框间自动移动, 不需要额外的操作, CPU 感知不到这种移动.

只有修改一个推移锁存器的值, 才能将对应的窗框移动到另一个窗壁上. 因此将推移锁存器的写入地址称为推移指针或推移向量. 显然, 给推移向量赋值的指令就是推移命令. 虽然推移命令很简单, 但必须知道推移向量的具体值, 推移命令才有意义. 若对用户隐去推移向量的具体值, 并将用户移动窗框的请求都由主板制造商提供的底层软件代理执行, 则用户就不能无意或恶意地移动窗框, 从而为保护系统安全提供了新手段.

由于推移锁存器的位数与双空间地址线中高序号线的数目一样多, 因此每一个推移锁存器都能将对应的窗框定位在任一个窗壁上, 从而使映射窗可以被对应到双空间存储器的任何位置, 解决了构建双空间存储系统的第 3 个关键问题 —— 内存空间在大很多的字空间上的移动问题.

4.4 双空间存储器管理

双空间存储器字空间与内存空间的对接是通过窗框将映射窗和窗壁对应起来而实现的, 于是记录好各窗壁与窗框的对应关系、该窗壁是否不可闭、窗壁内数据的属性等, 即可完成对字空间的管理.

双空间存储器块空间与目前的外存储器对应, 由于窗幅和外存块的大小都是 2 的幂, 所以二者一定成整数倍, 一般说窗幅大于外存块. 因此, 将窗壁细分成多个块, 则对双空间存储器块空间的管理就同目前对外存空间的管理完全一致. 于是兼顾窗壁管理和块管理即可实现对双空间存储器的管理.

表 1 窗壁管理表 (窗幅: 1 MB)

Table 1 Window entity management table (window size: 1 MB)

1	2	3	4	5				6	7			
				Block 0				⋯	Block 255			
Window entity No.	Window position (Starting address of window entity)	User visible	Closable	Window frame No.	File name	Succeeding window entity &block	File attribute	⋯	Window frame No.	File name	Succeeding window entity &block	File attribute
0	00000000 00 000	1	1	***	***	***	***	⋯	***	***	***	***
1	00000001 00 000	1	1	***	***	***	***	⋯	***	***	***	***
2	00000002 00 000	1	1	***	***	***	***	⋯	***	***	***	***
3	00000003 00 000	1	1	***	***	***	***	⋯	***	***	***	***
4	00000004 00 000	1	1	***	***	***	***	⋯	***	***	***	***
5	00000005 00 000	1	1	***	***	***	***	⋯	***	***	***	***
6	00000006 00 000	1	1	***	***	***	***	⋯	***	***	***	***
7	00000007 00 000	1	1	***	***	***	***	⋯	***	***	***	***
8	00000008 00 000	1	1	***	***	***	***	⋯	***	***	***	***
9	00000009 00 000	1	1	***	***	***	***	⋯	***	***	***	***
10	0000000A 00 000	1	1	***	***	***	***	⋯	***	***	***	***
11	0000000B 00 000	1	1	***	***	***	***	⋯	***	***	***	***
12	0000000C 00 000	1	1	***	***	***	***	⋯	***	***	***	***
⋯⋯	⋯⋯	1	1	***	***	***	***	⋯	***	***	***	***
2147483647	0FFFFFFF 00 000	1	0	FFE	Operating system	***	0	⋯	7FE	Operating system	***	0
2147483648	10000000 00 000	0	*	Hidden	***	***	***	⋯	Hidden	***	***	***
2147483649	10000020 00 000	0	*	Hidden	***	***	***	⋯	Hidden	***	***	***
⋯⋯	⋯⋯	0	*	Hidden	***	***	***	⋯	Hidden	***	***	***
4294967294	0FFFFFFFE 00 000	0	*	Hidden	***	***	***	⋯	Hidden	***	***	***
4294967295	0FFFFFFFF 00 000	0	0	4095	Underlying software	***	0	⋯	2047	Underlying software	***	0

例 3 对于例 1 描述的系统, 假设其外存空间的块大小为 4 KB, 则每个窗壁包含 256 个块. 于是, 可以用表 1 所示的数据结构作为管理双空间存储器的基础.

表 1 中, "***" 表示该内容由操作系统根据实际情况写入. 表 1 第一列为窗壁编号: 理论上该系统有 4 PB 容量, 分为 4294967296 个窗壁, 故表 1 相应有 4294967296 行, 但实际系统中往往用不了如此大的存储容量, 这时可以少装一些存储芯片, 例如只安装 2 PB 的存储芯片, 则 2147483648 号以后的窗壁不可用. 如果再使图 3 中的块管理器没有最高序号地址线 W_{51}, 则用户无法以外存方式访问双空间存储器的这一部分空间, 于是在表 1 中将这部分窗壁用灰色标示. 但是推移锁存器仍有 J_{31} 线, 于是通过内存空间可以访问这些窗壁, 再考虑到推移窗框的操作只能通过主板制造商提供的底层控制软件来执行, 因此可以禁止用户将窗框推移到这部分窗壁, 即这部分窗框对用户不可见. 于是, 主板制造商可以把最重要的软件放在这个区域, 如把初始化程序、中断向量表、推移向量表和窗壁管理表等放

在这个区域, 则用户不能对这些重要程序和数据进行恶意操作或误操作, 从而进一步提高了系统的安全性.

第 2 列是各窗壁的首地址 —— 窗位. 表 1 中双空间存储器的地址被分成了三段, 最低的 12 位是块内地址, 中间的 8 位是块号, 最高的 32 位是窗壁号.

第 3 列记录该窗壁是否用户可见. “1” 表示用户可见, 即底层软件将接受用户访问该窗壁的请求, 将一个窗框推移到该窗壁. “0” 表示用户不可见, 即底层软件将拒绝用户访问该窗壁的请求.

第 4 列记录该窗壁是否可以关闭. “1” 表示可闭, 即可以把其上的窗框移走. “0” 表示不可闭, 即不能把该窗壁的窗框移走, 这一点既可以用硬件实现, 如对应的推移锁存器为不可写入, 也可以用软件实现, 如底层软件将不移动不可闭窗上的窗框. 该列的 “*” 表示 0 或 1 均可.

第 5 列是 0 号块的登记表. 其中的 “窗框号” 列记录本块目前所占用窗框的编号, 用 *** 表示, 其块号就是该块起始地址在该窗框中的偏移量; “文件名” 列记录放置在本块上的文件的名称; “后继窗壁和块” 列记录本块所存储文件的相邻后继部分所在的窗壁号和块号, 这是将一个文件链接起来的指针; “文件属性” 列记录本块数据的属性, 如 “00” 为可执行文件, “01” 为可修改文件, “10” 为只读文件, “11” 为读写文件. 显然, 该列的位数越多, 能记录的文件属性就越多.

第 6 列表示 1 号块到 254 号块的登记表.

第 7 列表示 255 号块的登记表.

显然, 一个窗壁上的不同块可以占用不同的窗框, 即一个窗壁上可以安置多个窗框, 这只要给几个推移锁存器赋相同的值就可以了. 对于单任务系统这时浪费了窗框, 但对于多任务或多 CPU 系统, 利用这一点能够实现共享存储器结构, 关于这一方面的研究将另文讨论.

表 1 的第 5~7 列与目前的外存管理表很相似, 这是因为双空间存储器的块空间本身就起到目前外存的作用; 其中的 “窗壁号” 对应于目前的 “内存页框号”, 因为二者都是指示本块目前在随机访问存储空间中的映射关系, 但内存页框号指示本块已经完整地拷贝进内存的那个页上, 而 “窗框号” 指示可以通过那个窗框来直接随机访问本块, 这时没有发生任何数据拷贝.

与外存管理表相比, 在本质上表 1 仅多了第 3 和 4 列, 这两列指示双空间存储器字空间面向用户随机访问的特征.

4.5 窗框管理

推移操作是将一个窗框移动到一个窗壁上, 该操作的第一步是尽快找到暂时不用的窗框, 为此需要构建一个窗框管理表.

例 4 对于例 1 所述系统, 其窗框管理表如表 2 所示.

表 2 中, “***” 表示该内容由操作系统根据实际情况写入. 表 2 第 1 列是窗框编号; 第 2 列是该窗框当前位于的窗壁号; 第 3 列指示该窗框是否可移动, “1” 为可移动, “0” 为不可移动. 第 4 列指示已经有多长时间没有使用该窗框, 本列的值是构造窗框选择策略的基础. 第 5~7 列的含义同表 1.

表 2 与目前的 “内存页管理表” 或 “内存段管理表” 几乎相同, 源于它们都是对随机访问存储空间的管理方式, 但 “内存页管理表” 或 “内存段管理表” 管理的是实实在在的内存储器的一部分存储单元, 而表 2 管理的是推移锁存器对应的一个虚拟窗框, 或说表 2 本质上是管理推移锁存器.

表 2 中没有 “被修改” 指示位 (俗称脏位), 原因是对字空间存储单元的修改就是对双空间存储器单元的修改, 也就是对块空间存储单元的修改, 在这里没有 “将被修改的内容拷回非易失存储器” 的问题.

表 2 窗框管理表

Table 2 Window frame management table

1	2	3	4	5			6	7		
Window frame No.	Windows entity No.	Movable	Un-used time	Block 0			.....	Block FF		
				File name	Succeeding window entity & block	File attribute	.....	File name	Succeeding window entity & block	File attribute
0	***	1	***	***	***	***	.....	***	***	***
1	***	1	***	***	***	***	.....	***	***	***
2	***	1	***	***	***	***	.....	***	***	***
3	***	1	***	***	***	***	.....	***	***	***
.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....
FFE	0FFF FFFF	0	0	Operating system	***	00	.....	Operating system	***	00
FFF	0FFFF FFFF	0	0	Underlying software	***	00	.....	Underlying software	***	00

4.6 双空间存储器的使用

使用双空间存储器相当于综合使用内存和外存, 可以分成 3 种情况.

1. 启动计算机. 这个操作从给计算机上电或重启开始, 到计算机能够接受前台指令结束. 第 1 步是 CPU 在厂家设定的首指令位置找到第一条指令, 第 2 步是在首指令的引导下启动初始化程序, 第 3 步是由初始化程序启动操作系统.

在双空间存储器系统中, 首指令、初始化程序和操作系统都位于不可闭窗中, 如表 1 中的 4294967295 窗壁和 2147483647 窗壁, 它们都随时能够被 CPU 访问, 只要将首指令放置在正确位置, 即可顺利启动计算机, 其后的启动初始化程序和启动操作系统与当前的计算机完全相同, 但没有把操作系统拷入内存的过程, 故启动过程快很多. 细节请见第 5.2 小节.

2. 安装新软件. 这个操作主要是将新软件和数据安装到非易失性存储器, 并在文件管理表中进行登记, 目前这一操作都是在外存上实施. 在双空间存储系统中这一操作是在块空间中实施. 首先查表 1 找到空闲块, 然后将软件安装或拷贝到这个块中, 修改表 1 中该块的指示位并修改在不可闭窗中的注册表, 最后将该软件第一块所在的窗壁号和块号登记到文件管理表中. 鉴于文件管理表与本文的主要内容关系较远, 且与当前系统的文件管理表没有显著差别, 故文中没有专门讨论. 显然, 新系统和当前系统对于这个操作几乎没有差别.

3. 运行已安装好的程序. 这是日常操作, 目前的实施过程大致为:

(1) 查文件管理表, 找到该程序在外存的位置 (块号);

(2) 查内存管理表, 找到空闲的内存页;

(3) 将程序和数据拷进内存, 修改内存管理表、外存管理表和注册表, 完成内存地址重定位;

(4) 在内存中运行该程序;

(5) 若对内存中的区域有修改, 则适时将全部内容拷回外存.

在双空间存储器上的实施过程大致为:

(1) 查文件管理表, 找到该程序在块空间的位置 (窗壁号和块号);

(2) 查窗框管理表, 找到空闲的窗框;

(3) 给对应的推移锁存器赋值, 修改窗框管理表、窗壁管理表和注册表;

(4) 在字空间中运行该程序.

对比这两个系统的操作过程可以看到, 第 (1) 和 (2) 两项在两个系统中的操作相当, 因为这两个操作是查各种资源管理表, 不涉及存储器操作; 第 (3) 项在新系统中用给 32 位的推移锁存器赋值替代了将程序和数据拷贝进内存, 显然会大幅提高系统的运行效率; 第 (4) 项在两个系统中的操作也相当, 因为都是在可随机访问存储器中运行程序; 第 (5) 项在新系统中不存在, 因为所有修改都是对双空间存储器单元的直接更新.

由上述分析可以看到, 双空间存储器能避免程序运行前后其在内存和外存之间的拷贝过程, 从而显著提高系统的工作效率.

5 不可闭窗的重要性

在目前的计算机系统中, 总是将关乎系统效率、系统安全和保证系统正常运行的关键数据和程序驻留在内存中, 以便 CPU 能够及时访问这些数据和程序, 如当前注册表、内存管理表、各种资源管理表、操作系统内部指令对应的程序模块、防火墙软件、防病毒软件等. 而在双空间存储器中, 处理器随时可以访问不可闭窗中的数据、运行不可闭窗中的程序, 因此把上述关键数据和程序安置在不可闭窗中就能起到 “驻留内存” 的效用. 故不可闭窗的数量虽然很少但非常重要.

5.1 构造不可闭窗的方法

只需限定某个推移锁存器的值不能被更改, 则对应的窗框就不能被移动, 则该窗框所在的当前窗壁就被构造成一个不可闭窗. 使推移锁存器不能被修改的方法有两种, 一是推移锁存器本身为不可修改锁存器, 二是在窗框管理表中设定该窗框为不可移动. 前一种方法是硬件设置, 只有制造商能够实现. 后一种方法是软件实现, 当一个软件的级别很高时, 它可以修改窗框管理表, 将某个窗框设定为不可移动, 从而使该窗框的当前窗壁成为不可闭窗.

例 5 对于例 1 描述的系统, 其不可闭窗的设置及应用如图 4 所示.

5.2 安置系统的第一条指令

任何计算机系统复位或上电后都将到厂家设定的一个内存地址去取第一条机器指令, 并由此开始工作. 例如, Intel 公司将 IA32 架构 CPU (如 Pentium 系列) 系统复位后执行的第一条指令设定于 0FFFFFFF0H 内存地址 [26], 该地址的 $A_{31\sim20}$ 位全为 1, 于是系统复位后 CPU 取第一条指令时必定选择第 4095 号推移锁存器, 即系统复位后映射窗必定位于第 4095 号窗框, 另一方面, 第一条指令地址的 $A_{19\sim0}$ 位为 0FFFF0H, 于是必须将第一条指令安置于 4095 号窗框所在窗壁的 0FFFF0H 偏移地址处. 这只需要将第 4095 号推移锁存器设定为不能被修改, 从而使 4095 号窗框成为不可推移窗框, 它所在的窗壁就成为不可闭窗, 然后在这个不可闭窗的 0FFFF0H 偏移地址处存放好第一条指令即可. 如图 4(a) 中, 设定 4095 号推移锁存器的值为 0FFFFFFFFH, 并不能被改变, 则 4095 号窗框被定位在 4294967295 号窗壁, 该窗壁的起始地址为 FFFF FFFF 00 000H, 于是该窗壁的 0FFFF0H 偏移地址的双空间存储器地址为 0FFFF FFFF 00 000H+0FFFF0H=0FFFF FFFF FF FF0H, 于是在双空间存储器的这个地址处放置 CPU 复位后要执行的第一条指令, 见图 4(b).

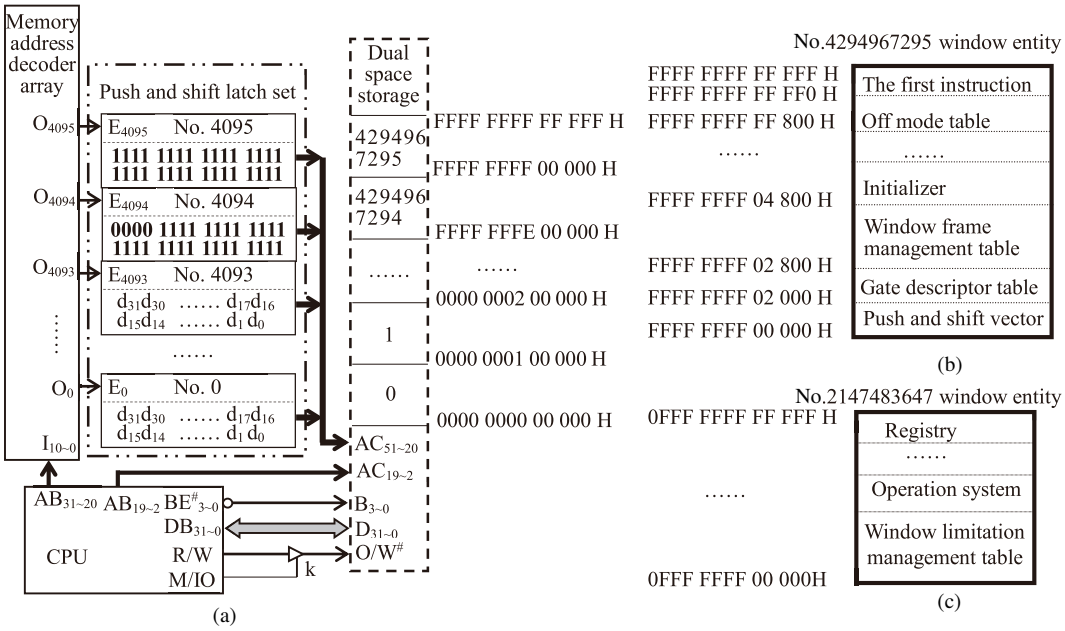

图 4 不可闭窗及其使用示意图

Figure 4 Un-closable window and its application

当系统复位后, 第一个时钟脉冲就驱动 CPU 输出 0FFFFFFF0H 内存地址, 并到这个内存地址取指令, 这个内存地址将选择 4095 号推移锁存器, 从而通过 4095 号窗框访问 4294967295 号窗壁, 在该窗壁中偏移量为 0FFFF0H 处取得第一条指令, 系统由此启动.

5.3 系统初始化程序的设置

初始化程序是系统进行自我检查和自我配置的最底层软件, 因此往往是系统启动后自动执行的第一个程序. 与此呼应, 系统的第一条指令总是跳转到初始化程序的入口. 考虑到此时任何用户都无法干涉系统的运行, 初始化程序也必须由厂家设置在一个不可闭窗中. 在图 4 示意的系统中, 初始化程序被安排在双空间存储器中从 0FFFF FFFF 04 800H 地址开始的位置, 这个位置在 4095 号映射窗中的偏移地址为 0FFFF FFFF 04 800H-0FFFF FFFF 00 000H=4800H, 于是, 第一条指令应该是一条以 4800H 为转移距离的段内转移指令, 这个设置符合 IA32 架构的要求.

在图 4 所示意的安排下, 系统复位后执行内存地址 0FFFF FFF0H 处第一条指令的操作被推移系统引导为执行双空间存储器 0FFFF FFFF FF FFF0H 处的指令, 该指令又将 CPU 转移到映射窗内偏移地址为 4800H 处的实地址方式初始化程序的入口处, 随后 CPU 执行初始化程序. 从 CPU 的角度看, 整个过程与目前的情况并无差别.

5.4 中断向量表的设置

中断向量表是频繁被 CPU 访问的数据结构, 由 CPU 制造商确定其具体结构和在内存空间中的具体位置, 例如 Intel 公司将 8086 系统的中断向量表设定在 00000~003FFH 内存地址. 但在例 1 所述的系统中, Intel 公司将保护模式下的中断向量表以门描述符表的形式给出, 并将门描述符表在内存中的具体地址由底层软件 (如操作系统或初始化程序) 通过内存重定位技术确定, 并将其基地址送入

门描述符表寄存器 IDTR [27], 以便 CPU 随时访问该表. 图 4(b) 中假设制造商将门描述符表设置在 4294967295 号窗壁的 2000H 处, 则制造商编制的初始化程序中就要给 IDTR 赋值 0FFF0 2000H, 即 CPU 认为门描述符表在内存空间的基地址为 0FFF0 2000H. 系统进入工作状态后, 一旦发生中断操作, CPU 会将 IDTR 寄存器的值加上中断类型码乘 8 生成该中断的门描述符入口的内存地址, 由于 IDTR 给出门描述符表基地址的高 12 位 $A_{31\sim20}$ 全为 1, 这个地址会自动选择 4095 号推移矢量, 于是这个内存空间地址通过 4095 号窗框落实到 4294967295 号窗壁, 又由于 IDTR 给出的基地址的低 20 位为 02000H, 即门描述符表入口地址在映射窗内的偏移地址为 02000H, 于是门描述符表在内存空间的基地址透过映射窗落实到 4294967295 号窗壁中 02000H 偏移地址处, 这时由 “中断类型码乘 8” 生成的门描述符表中的偏移地址落实为 4294967295 号窗壁中从 0FFFF FFFF 02 000H 地址开始的门描述符表内的偏移地址, 于是 CPU 给出的门描述符入口内存地址被自动落实到双空间存储器的相应地址, 并在这个地址上取得正确的门描述符. 对 CPU 而言, 整个过程与在 IA32 架构内存中的过程没有任何差异.

5.5 推移向量表和窗框管理表的设置

用户对窗框的使用和移动都必须由制造商提供的底层软件代为实施, 因此推移向量表和窗框管理表只可能被底层软件使用, 只需将二者在双空间存储器的具体位置与底层软件中的相关命令相一致即可. 图 4(b) 中分别将二者设置在 4294967295 号窗壁的 4800H 偏移地址处和 2800H 偏移地址处.

5.6 操作系统、注册表和窗壁管理表的设置

操作系统是用户经常使用的软件, 而注册表和窗壁管理表是操作系统管控系统资源的基本数据结构, 因此三者应该安置在一个 (或几个) 不可闭窗中. 当安装操作系统时, 安装程序向系统底层软件申请 “在外存空间” 上的安装位置, 底层软件将初始化程序退出时指向的一个不可闭窗的双空间地址告知安装程序, 安装程序将其视为 “外存空间根目录” 的地址, 按照目前的安装过程, 将操作系统安装在这个 “外存空间上”, 则操作系统就被安装到了指定的不可闭窗上. 图 4(c) 给出了将操作系统安装在不可闭窗 2147483647 号窗壁的情况, 4094 号推移锁存器指向这个窗壁.

操作系统安装好之后, 它再为自身设置注册表区和窗壁管理表区, 如图 4(c) 所示.

目前的计算机中操作系统安装在外存上, 但它的一些功能模块和重要数据结构, 如拷贝命令和注册表等, 经常要被使用, 为加快对这些模块和数据结构的访问, 往往将它们驻留在内存中. 于是在开机时总要等待操作系统将驻留内存的模块调入内存, 而且要更新注册表, 这就延长了每次开机的等待时间. 同时, 这些驻留内存的模块占用了大量的内存, 导致用户可用内存减少, 引发更多的内存调度, 降低了系统的整体效率. 在双空间存储器的计算机系统中, 操作系统、注册表和窗壁管理表都安装在不可闭窗中, 由于访问不可闭窗的时间与访问内存的时间相同, 于是对操作系统任何模块的访问速度都与访问目前驻留内存的模块相同. 另一方面, 由于双空间存储器具有非易失性, 安装在其上的任何程序和数据表都不会因掉电而丢失, 因此, 上电时不再需要启动操作系统的过程, 这将使用户体验到上电后系统会迅速启动.

显然, 同样的技术可用于安装其他应用程序, 如安装 office 程序等.

5.7 关机状态记录表和开机立即工作

利用双空间存储器的非易失性和可随机访问性, 可以在不可闭窗中增设一个关机状态表, 如图

4(b) 中所示. 当系统关机时, 底层控制软件将 CPU 的当前状态完整地保存在关机状态表中; 下次开机时, 初始化程序用关机状态表中保存的数据来恢复 CPU 的状态. 再考虑到操作系统、注册表、窗壁管理表、窗框管理表都在不可闭窗中, 可以立刻被访问, 而且各个推移锁存器保留了上次关机时的值, 以致上次运行中的程序、上次运行时访问的数据、文件管理表和其他管理表所对应的窗框都没有被移动. 于是, 当 CPU 的状态被恢复后, 计算机就会立即接续上次关机时的任务, 用户将完全没有开机等待的过程, 也不需要再次启动各个应用程序.

6 双空间存储器实验

鉴于块空间可以直接用作外存, 因此本文将实验集中在验证内存空间在双空间存储器上的推移技术, 实验的详细情况见参考文献 [28].

本实验以硬件仿真软件 Proteus 8.0 为实验平台. Proteus 8.0 有两个支撑本实验的关键性能: 一是它能够对 8086CPU 的工作过程进行仿真; 二是它能对目标系统中各种存储芯片的数据进行实时仿真 [29]. 利用前一点, 实验中能够仿真 8086CPU 对内存空间的访问; 利用后一点, 实验中能观察双空间存储器每个芯片的内容, 并以此来判断对内存空间的访问是否落实到双空间存储器的指定位置, 从而判断推移技术是否正确和可行.

6.1 实验目标系统设计

设计实验的目标系统为: 8086CPU 小模式结构、推移锁存器组包含 16 个 8 位的锁存器、16 MB 的双空间存储器字空间, 鉴于 Proteus 8.0 不能模拟非易失性随机存储器, 本实验中用 RAM 构成双空间存储器的字空间, 这不影响构造容量远大于内存空间的双空间存储器字空间和验证内存推移技术. 在这个目标系统中, 若能按照推移技术的操作过程使 8086CPU 透过它的 1 MB 内存地址及时地随机访问到 16 MB 的字空间的任何单元, 则推移技术和相关理论的正确性被验证.

系统地址线连接设计: 内存地址线的高 4 根 ($AB_{19\sim16}$) 接入推移锁存器组的输入端 ($I_{3\sim0}$), 推移锁存器的 8 根输出线 ($J_{7\sim0}$) 接入双空间存储器的高位地址线 ($CAB_{23\sim16}$); 内存地址线的其余 16 根 ($AB_{15\sim0}$) 直接接入双空间存储器的低 16 根地址线 ($CAB_{15\sim0}$), 故窗幅为 64 KB.

首指令与初始化程序设置: 8086 首指令的地址为 0FFFF0H, 其高 4 位为 1111, 于是 $AB_{19\sim16}$ 将选中第 15 号推移锁存器. 因此, 本实验将第 15 号推移锁存器设定为不可修改并赋值 0FFH, 于是 255 号窗壁成为不可闭窗. 首指令地址的低 16 位为 0FFF0H, 于是将首指令放置在 255 号窗壁内偏移量为 0FFF0H 处. 再将初始化程序设置在 255 号窗壁中偏移量为 0000H 处, 于是首指令是跳转到 F0000H 的无条件转移指令, 其汇编指令为: jmp CS:0000, 其中 CS 已被赋值 0F000H.

中断向量表设置: 8086 的中断矢量表必须位于内存的 00000H 到 003FFH 区域, 且必须能够随时被 CPU 随机访问. 中断向量表地址的高 4 位 ($AB_{19\sim16}$) 为 0000B, 因此访问中断向量表时总是选中第 0 号推移锁存器. 鉴于此, 本实验给 0 号推移锁存器设定固定值 0FEH, 从而将 254 号窗壁设定为不可闭窗, 同时将中断向量表设置在 254 号窗壁上偏移地址为 00000H 到 03FFH 的区域.

推移向量表和推移指令的设计: 本实验设计 16 个推移锁存器的写入端地址如表 3, 并将该表设置在 254 号窗壁, 称为推移向量表. 同时设置对推移锁存器写入的数据由数据线 $D_{7\sim0}$ 传送. 因此实验系统的推移命令采用如下形式: MOV DS, 0400H; MOV [推移锁存器号$\times$ 2], 00**H, 其中 * 号为一位十六进制数.

表 3 推移锁存器写入端地址

Table 3 Writing entrance address of push-and-shift latches

Push and shift laches No.	Writing entrance address1	Writing entrance address2
0	000400	000401
1	000402	000403
2	000404	000405
3	000406	000407
4	000408	000409
5	00040A	00040B
6	00040C	00040D
7	00040E	00040F
8	000410	000411
9	000412	000413
A	000414	000415
B	000416	000417
C	000418	000419
D	00041A	00041B
E	00041C	00041D
F	00041E	00041F

6.2 实验内容

本实验设计了下列 8 项实验内容来验证构建双空间存储器的可行性和内存推移技术的有效性与正确性, 实验的具体操作和各项结果请见参考文献 [28], 对这 8 项内容进行了多次实验, 实验结果均相同, 且证明了双空间存储器理论正确性和内存空间推移技术的有效性.

(1) 绘制目标系统电路.

(2) 设置目标系统的初始化程序和首指令.

(3) 模拟 8086CPU 启动过程.

(4) 执行目标系统初始化程序, 并设置推移矢量表模拟实验.

(5) 透过映射窗访问双空间存储器字空间实验.

(6) 修改推移锁存器实验.

(7) 移动窗框实验.

(8) 中断系统实验.

6.3 实验结论

基于 Proteus 8.0 硬件仿真系统完成的实验证实了构造双空间存储器的可行性, 证实了将内存空间在双空间存储器上推移的理论及核心技术. 在这个实验中, 8086CPU 的 1 MB 内存空间依靠任一个推移锁存器均能被分时映射到 16 MB 的双空间存储器的任意位置, 从而使 8086CPU 对其 1 MB 内存空间的随机访问自动落实为对 16 MB 双空间存储器指定位置的随机实时访问. 实验中通过设置不可闭窗, 保证了 CPU 的正确启动、执行初始化程序和执行中断命令. 实验中对推移锁存器进行了方便灵活的改写, 并以此实现了将内存空间在双空间存储器上的推移操作.

7 内存推移原理和技术与相近技术的不同点

7.1 与内存管理技术的不同点

分段技术和分页技术是内存管理的两个基本技术, 二者结合又产生了段页式管理技术. 这 3 种技术的共同点是相加基地址值和偏移地址值生成内存的物理地址, 前者由机器指令给出的两个逻辑地址生成线性地址或物理地址, 后者由线性地址和页基址生成物理地址, 这些操作都在 CPU 内部完成, CPU 将其生成的物理地址送上内存地址总线. 这些技术都不能扩大随机访问存储空间的容量.

而推移技术则是将物理地址自动映射到大很多的双空间存储器地址, 这个过程在 CPU 外部自动完成, 与 CPU 无关, 没有地址相加运算. 两类技术的共同点是都涉及到内存地址和随机访问存储空间.

7.2 与内存空间扩大技术的对比

虚拟存储器是目前常用的一项内存空间扩大技术, 其核心是采用内存重定位技术将位于外存上的程序和数据分批调入较小的内存中运行, 从而在逻辑上用外存储器的一部分来延展程序的运行空间. 它没有增加系统的随机访问存储器空间, 更没有消除数据在内存和外存之间的拷贝过程, 这是它不及双空间存储器的地方.

与内存推移技术比较接近者当属在 8086 时代曾出现过的扩充内存技术 (expanded memory specification, EMS). 扩充内存技术是在 8086 的 1 MB 内存空间中指定一块 64 KB 范围, 并将其分为 16 KB 容量的 4 个页; 同时将最大为 32 MB 的扩充内存专用板的存储器也按 16 KB 分页, 用额外的指令可将扩充内存板上的页与内存空间的页建立映射, 每次映射 4 个页, 以此 CPU 可分时访问全部 EMS 存储器. 两种技术的相同之处在于: 都是利用锁存器把内存空间的一个较小范围映射到了一个较大的存储空间. 但二者有更多的明显不同: (1) 扩充内存技术仅仅把一小部分内存空间用于映射, 大部分内存空间仍然用于内存储器; 而推移技术将全部内存空间映射到了双空间存储器上, 完全没有内存储器. (2) 扩充内存技术的映射过程比较复杂, 必须安装专门的软件, 内存推移技术只需要简单的推移命令即可完成映射. (3) 访问扩充内存需要两条指令, 是 “用时间换空间” 的典型技术, 而访问双空间存储器不需要增加时间, CPU 根本不知道内存地址已经透过映射窗落实到了双空间存储器上. (4) 扩充内存不具有非易失性, 因此也不存在不可闭窗. 随着 80286 提供了扩展内存, 扩充内存技术就不再被应用.

7.3 与固态盘应用技术的对比

各种固态存储器, 如相变存储器 (PCM)、铁电存储器 (FRAM)、磁阻存储器 (MRAM)、阻变存储器 (RRAM) 和闪存 (flash), 以及结合固态盘和动态随机存储器形成的非易失随机存储芯片 (NVRAM) 都有非易失性和随机访问性. 因此, 它们都能够成为构建双空间存储器的物理实体, 但它们本身都不是双空间存储器, 它们本身也不存在内存空间推移技术.

目前常用固态盘作为外部存储器, 如 U 盘, 也常用固态盘作为磁盘阵列的 I/O 缓存区, 如各种混合存储技术 [30, 31], 在这种场合固态盘与内存没有关系, 因此与双空间存储器没有关系, 也不存在内存推移技术.

目前固态盘也被用作为系统意外掉电时保护内存数据的备份存储器, 即 NVDIMM 内存条. 但这种芯片本身并不涉及双空间存储器结构, 也不存在内存推移技术.

目前固态盘应用的一个研究热点是存储级内存技术 (storage-class memory, SCM) [32]. 这项技术的核心是研究如何用固态盘或多种固态盘组合来构造内存储器, 目标是用固态盘芯片来改善内存的性

能. 但 IBM 公司也试图用这种方式在 “外存和内存之间插入一个存储层次”. 显然这类研究包含有 “使内存具有非易失性和用动态存储器来改善固态盘的写特性” 两个方面, 在这两点上它与双空间存储器理论相同. 但 SCM 没有 “双空间存储器” 概念和构造技术, 更没有内存空间推移技术, 可以认为 SCM 仅仅涉及到双空间存储系统的一小部分.

8 结论

用具有非易失性和随机访问性的存储元件能够建立双空间存储器, 在双空间存储器中消除了数据/程序在内存与外存之间的拷贝过程, 可以消除开机和启动程序的等待过程, 利用推移锁存器能够将内存空间完整地分时映射到很大的双空间存储器的各个区域, 从而使 CPU 可以自动的随时随机访问双空间存储器的任何单元. 对 CPU 而言, 访问双空间存储器的速度和方式与访问内存完全相同, CPU 完全感觉不到对内存的访问被内存推移系统落实到了双空间存储器上, 因此 CPU 对内存管理的现有技术不受影响, 而未来的 CPU 可以省掉内存管理技术. 本文对双空间存储器理论和内存空间推移技术做了详细讨论, 并给出了在经典 CPU —— 8086 系统中实现双空间存储器和内存空间推移技术的仿真实验, 实验结果证实了这项理论的正确性和核心技术的有效性.

双空间存储器理论和内存空间推移技术不仅解决了三值光学计算机的存储器问题, 而且为构造基于固态盘的计算机新体系结构提供了理论和技术基础.

参考文献

1 Yan J Y, Jin Y, Zuo K Z. Decrease-radix design principle for carrying/borrowing free multi-valued and application in ternary optical computer. Sci China Ser F-Inf Sci, 2008, 51: 1415–1426

2 Yan J Y, Jin Y, Zuo K Z. Carry-free n-value operator. China Patent, ZL200710041144.1 [严军勇, 金翊, 左开中. 无进位无借位 n 值运算器. 中国专利, ZL200710041144.1]

3 Wang X C, Peng J J, Ouyang S. Control method for the optical components of a dynamically reconfigurable optical platform. Appl Optics, 2011, 50: 662–670

4 Jin Y, Wang H J, Ouyang S, et al. Principles, structures, and implementation of reconfigurable ternary optical processors. Sci China Inf Sci, 2011, 54: 2236–2246

5 Song K, Jin Y, Ouyang S, et al. Reconfigurable ternary optical processor with double rotator structure. Optics Precis Eng, 2012, 20: 1890–1898 [宋凯, 金翊, 欧阳山, 等. 双旋光器结构的可重构三值光学处理器. 光学精密工程, 2012, 20: 1890–1898]

6 Wang H J, Jin Y, Ouyang S. Design and implementation of a 1-bit reconfigurable ternary optical processor. Chinese J Comput, 2014, 37: 1500–1507 [王宏健, 金翊, 欧阳山. 一位可重构三值光学处理器的设计和实现. 计算机学报, 2014, 37: 1500–1507]

7 Jin Y, Ouyang S, Peng J J, et al. Reconfigurable ternary optical computer. China Patent, ZL201010584129.3 [金翊, 欧阳山, 彭俊杰, 等. 可重构的三值光学处理器. 中国发明专利, ZL201010584129.3]

8 Jin Y, Shen Y F, Peng J J, et al. Principles and construction of MSD adder in ternary optical computer. Sci China Inf Sci, 2010, 53: 2159–2168

9 Song K, Yan L P. Design and implementation of the one-step MSD adder of optical computer. Appl Optics, 2012, 51: 917–926

10 Shen Y F, Pan L, Jin Y, et al. One-step binary MSD adder for ternary optical computer. Sci Sin Inform, 2012, 42: 869–881 [沈云付, 潘磊, 金翊, 等. 三值光学计算机一种限制输入一步式 MSD 加法器. 中国科学: 信息科学, 2012, 42: 869–881]

11 Peng J J, Liu Y P, Jin Y, et al. Carry-free adder based on ternary optical computer. China Patent, ZL201010518342.4 [彭俊杰, 刘艳萍, 金翊, 等. 三值光学计算机的无进位加法器. 中国发明专利, ZL201010518342.4]

12 Peng J J, Shen R, Jin Y, et al. Design and implementation of modified signed-digit adder. IEEE Trans Comput, 2014, 63: 1134–1143

13 Jin Y, Ouyang S, Song K, et al. Management of many data bits in ternary optical computers. Sci Sin Inform, 2013, 43: 361–373 [金翊, 欧阳山, 宋凯, 等. 三值光学处理器的数据位管理理论和技术. 中国科学: 信息科学, 2013, 43: 361–373]

14 Jin Y. Management strategy of data bits in ternary optical computer, J Shanghai Univ (Natural Science), 2007, 13: 519–523 [金翊. 三值光计算机高数据宽度的管理策略. 上海大学学报 (自然科学版), 2007, 13: 519–523]

15 Zhang Q, Jin Y, Song K, et al. MPI programming based on ternary optical computer in supercomputer. J Shanghai Univ (Natural Science). 2014, 20: 180–189 [张茜, 金翊, 宋凯, 等. 三值光学计算机 MPI 编程技术在超算集群中的使用. 上海大学学报 (自然科学版), 2014, 20: 180–189]

16 Gao H, Jin Y, Song K. Extension of C language in ternary optical computer. J Shanghai Univ (Natural Science), 2013, 19: 280–285 [高桓, 金翊, 宋凯. 针对三值光学计算机的 C 语言扩展. 上海大学学报 (自然科学版), 2013, 19: 280–285]

17 Jin Y, Xu Q, Ouyang S, et al. Structured data computer—application characteristics of ternary optical computer. Sci Sin Inform, 2016, 46: 311–324 [金翊, 徐群, 欧阳山, 等. 结构量计算机 —— 三值光学计算机的应用特点. 中国科学: 信息科学, 2016, 46: 311–324]

18 Jin Y. Draw near optical computer. J Shanghai Univ (Natural Science), 2011, 17: 401–411 [金翊. 走近光学计算机. 上海大学学报 (自然科学版), 2011, 17: 401–411]

19 Lu Y Y, Shu J W. Survey on flash-based storage systems. J Comput Res Dev, 2013, 50: 49–59 [陆游游, 舒继武. 闪存存储系统综述. 计算机研究与发展, 2013, 50: 49–59]

20 Fan C F, Yang Y, Zhang S M, et al. Review of patent technology related to phase change memory. Metallic Functional Materials, 2013, 20: 54–59 [范崇飞, 杨燕, 张思秘, 等. 相变存储器专利技术现状和趋势分析. 金属功能材料, 2013, 20: 54–59]

21 Hao J H, Gao H. Micromagnetic simulation of magnetization reversal on the annular free layer with nick in magnetic random access memory. Acta Phys Sin, 2013, 62: 057502 [郝建红, 高辉. 磁存储器环形带切口结构自由层磁化反转的微磁模型. 物理学报, 2013, 62: 057502]

22 Zhai Y H, Li W, LI P, et al. Research progress of radiation hardened ferroelectric random access memory. Materials Review A: Review Article, 2012, 26: 34–38 [翟亚红, 李威, 李平, 等. 抗辐射铁电存储器的研究进展. 材料导报 A : 综述篇, 2012, 26: 34–38]

23 Duan S K, Hu X M, Wang L D, et al. Memristor-based RRAM with applications. Sci Sin Inform, 2012, 42: 754–769 [段书凯, 胡小方, 王丽丹, 等. 忆阻器阻变随机存取存储器及其在信息存储中的应用. 中国科学: 信息科学, 2012, 42: 754–769]

24 Jin Y, Ouyang S, Shen Y F, et al. A new computer architecture and its read/write method. China Patent, 201410199434.9 [金翊, 欧阳山, 沈云付, 等. 一种计算机系统和数据读写方法. 中国发明专利, 201410199434.9]

25 Jin Y, Wang X C, Peng J J, et al. Conceptual structure of ternary optical computer and high performance computer merger. High Perform Comput Technol, 2010. 1–4 [金翊, 王先超, 彭俊杰, 等. 三值光学计算机与高性能计算机系统融合的概念结构. 高性能计算技术, 2010. 1–4]

26 Intel. Intel 64 and IA-32 Architectures Software Developer's Manual, Volume 3A, System Programming Guide, Part 1. 9.1.4 First instruction Executed, Vol.3A:9-6

27 Shi X F, Jin Y, Feng P, et al. 32 Bits Microcomputer Principle, Interfacing and Application. 2nd Ed. Xi'an: Northwestern Polytechnical University Press, 2001. 135–139 [史新福, 金翊, 冯萍, 等. 32 位微型计算机原理接口技术及其应用. 第二版. 西安: 西北工业大学出版社, 2001. 135–139]

28 Zhan H J, Jin Y, Ouyang S, et al. Experimentation of moving memory space on double-space storage. J Shanghai Univ (Natural Science), accepted [展豪君, 金翊, 欧阳山, 等. 内存空间在双空间存储器上的推移技术实验研究. 上海大学学报 (自然科学版), 录用]

29 Gu H, Liang X Y. Microcomputer Principle & Interfacing: Based on 8086 and Proteus Simulation. Beijing: Publishing House of Electronics Industry, 2011. 177–288 [顾晖, 梁惺彦. 微机原理与接口技术: 基于 8086 和 Proteus 仿真. 北京: 电子工业出版社, 2011. 177–288]

30 Zhu Q, Li X Y. A review on hybrid storage. Microcomput Appl, 2013, 29: 33–37 [祝青, 李小勇. 混合存储综述. 微型电脑应用, 2013, 29: 33–37]

31 Li H Y. SMARC: exploit SLC and MLC cells complementarily and effectively within one architecture. Appl Res Comput, 2013, 30: 2443–2446 [李红艳. SMARC: 一种结合 SLC 和 MLC 的混合固态盘架构. 计算机应用研究, 2013, 30: 2443–2446]

32 Freitas R, Wilcke W, Kurdi B. Storage class memory, technology and use. Tutorial of 6th USENIX Conference on File and Storage Technologies (FAST'08), 2008

Structure and theory of dual-space storage for ternary optical computer

Shan OUYANG[1,2], Junjie PENG[1*], Yi JIN[1,2], Yunfu SHEN[1], Xuemin LIU[1], Yuexing HAN[1] & Weimin LI[1]

1 *School of Computer Engineering and Science, Shanghai University, Shanghai* 200444*, China*;
2 *High Performance Computing Center, Shanghai University, Shanghai* 200444*, China*
*E-mail: jjie.peng@shu.edu.cn

Abstract A ternary optical processor can have thousands of data bits, each of which can be independently assigned to a different task and reconstructed in real-time according to user demand at runtime. Consequently, significant amounts of data are frequently transferred between storage and processor in a ternary optical computer. In this study, a dual-space storage (DSS) system and a new technique to push memory space (PMS) on the DSS was developed to rectify this issue. The developed methods exploit the non-volatility and random access of solid state disks. This paper introduces the theory, architecture, management, and usage of DSS in detail, and also describes the hardware structure, technical principles, and push commands of PMS. Several new methods based on the unclosable windows in DSS (such as jobs resuming as soon as the computer is powered on, elimination of the wait time to launch a program, and improved system security) are also discussed. The results of simulation of DSS and PMS in an 8086 system verify the efficacy of the new theory and the related technologies. Both DSS and PMS not only meet the memory requirements of the ternary optical computer, but also provide a theoretical and technical foundation for constructing a new computer architecture based on solid state disks.

Keywords solid state disk, dual-space storage, push and shift technique for memory, unclosable window, ternary optical computer

Shan OUYANG was born in 1979. He received a Ph.D. degree from the School of Computer Engineering and Science, Shanghai University in 2012. Currently, he is an engineer at Shanghai University. His research interests include ternary optical computer and embedded systems.

Junjie PENG was born in 1977. He received a Ph.D. degree in computer application technology from Harbin Institute of Technology, Harbin in 2005. Currently, he is an associate professor at the School of Computer Science at Shanghai University. His research interests include architecture of new computer systems, and storage and energy-saving in cloud computing. He is the chair of CCF YOCSEF Shanghai (2015-2016).

Yi JIN was born in 1957. He received a Ph.D. degree in computer science from Northwestern Polytechnic University, Xi'an, in 2003. Currently, he is a professor and senior researcher at Shanghai University. His research interests include optical computer and computer architecture. Dr. JIN is a senior member of China Computer Federation.

Yunfu SHEN was born in 1960. Currently, he is an associate professor at Shanghai University. His research interests include software formalization, model checking, computing technology, and ternary optical computer and its reliability.