

[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 11/07 (2006.01)



[12] 发明专利说明书

专利号 ZL 03819900.9

[45] 授权公告日 2009年3月11日

[11] 授权公告号 CN 100468351C

[22] 申请日 2003.6.13 [21] 申请号 03819900.9
[30] 优先权
[32] 2002.6.29 [33] US [31] 10/184,774
[86] 国际申请 PCT/US2003/018667 2003.6.13
[87] 国际公布 WO2004/003749 英 2004.1.8
[85] 进入国家阶段日期 2005.2.22
[73] 专利权人 英特尔公司
地址 美国加利福尼亚州
[72] 发明人 S·贝内特 A·安德森
S·耶亚辛 A·卡吉 G·奈格尔
R·乌尔利希 X·邹 M·科祖赫
[56] 参考文献
US4812967A 1989.3.14
CN85101171A 1987.2.11
GB1495729A 1977.12.21

US6237061B1 2001.5.22
MICROPROCESSING AND MICROPROGRAMMING. SHANG RONG TSAI ET AL, 81.96, ON THE ARCHITECTURAL SUPPORT FOR LOGICAL MACHINE SYSTEMS. 1988
审查员 姜玲玲
[74] 专利代理机构 中国专利代理(香港)有限公司
代理人 杨凯 王勇

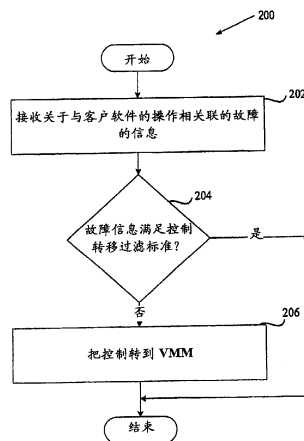
权利要求书3页 说明书20页 附图9页

[54] 发明名称

对在虚拟机体系结构中客户软件操作期间出现的故障的控制

[57] 摘要

在一个实施例中,接收关于与客户软件的操作相关联的故障的故障信息。此外,确定故障信息是否满足一个或多个过滤标准。如果该确定为肯定的,则控制仍保留在客户软件中而不转移到虚拟机监控器(VMM)。



1. 一种用于在客户软件的操作期间控制出现的故障的方法，所述方法包括：

接收关于与所述客户软件的所述操作相关联的故障的故障信息；

确定所述故障信息是否满足一个或多个过滤标准的全体；

如果所述故障信息满足所述过滤标准的全体，则禁止把控制转移到虚拟机监控器，其中禁止把控制转移到虚拟机监控器包括把所述故障传递给所述客户软件，以及

如果所述故障信息不满足所述过滤标准中的至少一个，则把控制转移到所述虚拟机监控器。

2. 如权利要求1所述的方法，其特征在于：

所述过滤标准是一个或多个控制转移过滤标准；以及

禁止把控制转移到所述虚拟机监控器还包括如果所述故障信息满足所述控制转移过滤标准的全体，则把所述故障传递给所述客户软件。

3. 如权利要求1所述的方法，其特征在于，所述故障信息为故障标识符、与所述故障相关联的误码以及与所述故障相关联的一个或多个附加数据值其中的至少一项。

4. 如权利要求1所述的方法，其特征在于，所述故障表示中断、异常和平台事件其中的任一项。

5. 如权利要求1所述的方法，其特征在于，所述故障信息包括多个值，所述多个值中的每个值指示特定条件的出现。

6. 如权利要求5所述的方法，其特征在于，所述故障信息的值的一个或多个组合要求向所述虚拟机监控器的转移。

7. 如权利要求1所述的方法，其特征在于，所述过滤标准采用在一个或多个指定字段中存储的数据来定义。

8. 如权利要求7所述的方法，其特征在于，所述一个或多个指定

字段包括匹配字段和掩码字段。

9. 如权利要求 8 所述的方法，其特征在于，确定所述故障信息是否满足所述过滤标准的全体还包括：

采用按位进行的“与”算子将所述故障信息与所述掩码字段中存储的数据组合；以及

确定所述组合的结果是否与所述匹配字段中存储的数据匹配。

10. 如权利要求 9 所述的方法，其特征在于，确定所述故障信息是否满足所述过滤标准的全体还包括：

确定所述组合的结果是否与所述匹配字段中存储的数据匹配；以及

识别重定向图中相应位的值，所述相应位的值指示所述故障是否要求控制转移到所述虚拟机监控器。

11. 如权利要求 9 所述的方法，其特征在于，确定所述故障信息是否满足所述过滤标准的全体还包括：

确定所述组合的结果与所述匹配字段中存储的数据不匹配；以及
识别重定向图中相应位的值；以及

对所述相应位的值取反，所述相应位的取反值指示所述故障是否要求控制转移到所述虚拟机监控器。

12. 一种用于在客户软件的操作期间控制出现的故障的系统，所述系统包括：

用于接收关于与所述客户软件的所述操作相关联的故障的故障信息的部件；

用于确定所述故障信息是否满足一个或多个过滤标准的全体的部件；

用于如果所述故障信息满足所述过滤标准的全体，则禁止把控制转移到虚拟机监控器的部件，其中禁止把控制转移到虚拟机监控器包括把所述故障传递给所述客户软件；以及

用于如果所述故障信息不满足所述过滤标准中的至少一个，则把

控制转移到所述虚拟机监控器的部件。

13. 如权利要求 12 所述的系统，其特征在于：

所述过滤标准是一个或多个控制转移过滤标准；以及

所述用于禁止把控制转移到虚拟机监控器的部件包括用于如果所述故障信息满足所述控制转移过滤标准的全体，则把所述故障传递给所述客户软件的部件。

14. 如权利要求 13 所述的系统，其特征在于，所述控制转移过滤标准采用在一个或多个指定字段中存储的数据来定义，所述一个或多个指定字段包括匹配字段和掩码字段。

15. 如权利要求 14 所述的系统，其特征在于，所述用于确定所述故障信息是否满足一个或多个过滤标准的全部的部件包括：

用于采用按位进行的“与”算子将所述故障信息与所述掩码字段中存储的数据组合的部件；以及

用于确定所述组合的结果是否与所述匹配字段中存储的数据匹配的部件。

对在虚拟机体系结构中客户软件操作期间出现的故障的控制

发明领域

一般来说,本发明涉及虚拟机,更具体来说,涉及处理与虚拟机体系结构中客户软件的操作相关联的故障。

发明背景

传统的虚拟机监控器(VMM)通常在计算机上运行,并为其它软件提供一个或多个虚拟机的抽象。各虚拟机可用作自含式平台,运行它自己的“客户操作系统”(即VMM接纳的操作系统(OS))和其它软件,这些统称为客户软件。客户软件预期工作起来就象它在专用计算机而不是虚拟机上运行一样。也就是说,客户软件预期控制各种事件并有权访问硬件资源。硬件资源可包括处理器驻留资源(例如控制寄存器)、驻留在存储器中的资源(例如描述符表)以及驻留在基础硬件平台上的资源(例如输入-输出装置)。事件可包括中断、异常、平台事件(例如初始化(INIT)或系统管理中断(SMI))等。

在虚拟机环境中,VMM应当能够具有对这些事件和硬件资源的最终控制,以便提供在虚拟机上运行的客户软件的适当操作,以及用于从在虚拟机上运行的客户软件及在它们之间保护。为了实现这个目的,当客户软件访问受保护资源时或者其它事件(诸如中断或异常)出现时,VMM通常接收控制。

例如,当页面错误(即与地址转换操作相关联的异常)在客户软件的操作期间出现时,控制被转移到VMM,VMM则确定客户软件或VMM本身是否需要处理页面错误。如果页面错误需要由客户软件处理,则控制重新转移到客户软件。由于不需要由VMM处理的页面错误(以及其它异常和中断)相当频繁地发生,因此存在与VMM和客户

软件之间的这种控制转移相关联的大量性能成本。

附图简介

在附图的各图中通过举例而不是限制的方式来说明本发明，附图中相似的参考标号表示类似的元件，其中：

图 1 说明本发明可在其中工作的虚拟机环境的一个实施例；

图 2 是因故障而利用控制转移过滤标准过滤 VM 出口的过程的一个实施例的流程图；

图 3-5 说明因故障而利用控制转移过滤标准过滤 VM 出口的过程的示范实施例；

图 6 是因故障而利用故障过滤标准和控制转移过滤标准过滤故障和 VM 出口的过程的一个实施例的流程图；

图 7-8 说明利用故障过滤标准过滤故障的过程的示范实施例；以及

图 9 是定义故障过滤标准和控制转移过滤标准的过程的一个实施例的流程图。

实施例描述

描述处理与虚拟机体系结构中的客户软件的操作相关联的故障的方法和装置。为了说明，以下描述中提出了大量具体细节，以便透彻地理解本发明。然而，本领域的技术人员十分清楚，即使没有这些具体细节，也可以实施本发明。

以下详细描述某些部分是以对计算机系统的寄存器或存储器中的数据位进行操作的算法和符号表示来提供的。这些算法描述和表示是数据处理领域的技术人员用来向本领域的其它技术人员最有效地传达其工作实质的方式。算法在这里以及一般被认为是产生所希望结果的独立运算序列。运算是物理量的那些要求的物理处理。这些量通常但不一定采取电或磁信号的形式，它们能够被存储、传送、组合、

比较以及以其它方式处理。主要为了普通使用的原因，将这些信号称作位、值、元素、符号、字符、项目、数字等，已经证明有时非常方便。

但应当记住，所有这些及类似的术语要与适当物理量相关联，并且只是应用于这些量的便捷标记。若没有明确说明，从以下论述中显而易见，在本发明中，采用诸如“处理”或“计算”或“运算”或“确定”等术语的论述可表示计算机系统或类似电子计算装置的动作和过程，所述计算机系统或类似电子计算装置处理表示为计算机系统的寄存器和存储器中的物理(电子)量的数据并将其转换为类似地表示为计算机系统存储器或寄存器或者其它这类信息存储、传送或显示装置中的物理量的其它数据。

在实施例的以下详细描述中，参照附图，附图以图解说明的方式示出可实施本发明的具体实施例。附图中，相似的标号描述若干视图中基本相似的元件。对这些实施例进行充分详细的描述，使本领域的技术人员能够实施本发明。可采用其它实施例，并且可进行结构、逻辑以及电气变更，而没有背离本发明的范围。此外，大家理解，本发明的各种实施例虽然有所不同，但不一定是相互排斥的。例如，在一个实施例中描述的特定功能、结构或特征可包含在其它实施例中。因此，以下详细描述不是限制性的，本发明的范围仅由所附权利要求以及这些权利要求涵盖的全部等效范围来限定。

图 1 说明本发明可在其中工作的虚拟机环境 100 的一个实施例。在此实施例中，裸平台硬件 116 包括计算平台，它可以例如能够运行标准操作系统(OS)或虚拟机监控器(VMM)、如 VMM 112。VMM 112 虽然通常以软件实现，但可模拟裸机接口并将其输出到较高层软件。这种较高层软件可包括标准或实时 OS，可以是具有有限操作系统功能性的严重剥离的操作环境，可以不包括传统的 OS 工具，等等。或者，例如 VMM 112 可以在另一个 VMM 之中或之上运行。VMM 及其典型特征和功能性是本领域的技术人员熟知的，并且可例如以软

件、固件或通过各种技术的组合来实现。

平台硬件 116 可以属于个人计算机(PC)、大型计算机、手持装置、便携计算机、机顶盒或其它任何计算系统。平台硬件 116 包括处理器 118、存储器 120 以及可能的其它未示出的平台硬件(例如输入-输出装置)。

处理器 118 可以是能够运行软件的任何类型的处理器, 诸如微处理器、数字信号处理器、微控制器等等。处理器 118 可包括用于完成本发明的方法实施例运行的微码、可编程逻辑或硬编码逻辑。

存储器 120 可以是硬盘、软盘、随机存取存储器(RAM)、只读存储器(ROM)、闪速存储器、以上装置的任何组合、或者处理器 118 可读的其它任何类型的机器媒体。存储器 120 可存储用于完成本发明的方法实施例运行的指令或数据。

VMM 112 为其它软件(即“客户”软件)提供一个或多个虚拟机(VM)的抽象。VMM 112 可向各种客户提供相同或不同的抽象。图 1 表示两个这样的 VM、即 102 和 114, 但 VMM 112 可支持多于或少于 2 个的 VM。在各 VM 上运行的客户软件可包括诸如客户 OS 104 或 106 等客户 OS 以及各种客户软件应用程序 108 和 110。客户 OS 和软件应用程序在本文中统称客户软件 103 和 115。客户软件 103 和 105 预期访问客户软件正在其中运行的 VM 102 和 114 中的物理资源(例如处理器寄存器、存储器和 I/O 装置)。VMM 112 便于对客户软件所需资源的访问, 同时保持对平台硬件 116 内资源的最终控制。

此外, 客户软件 103 和 115 预期处理各种事件, 诸如异常、中断以及平台事件(例如初始化(INIT)和系统管理中断(SMI))。这些事件中的一部分是“特许的”, 因为它们必须由 VMM 112 来处理, 以便确保 VM 102、114 和客户软件 103、115 的正确工作, 以及用于从客户软件 103 和 115 及其之间保护。对于特许事件, VMM 112 便于客户软件所需的功能性, 同时保持对这些特许事件的最终控制。便于客户软件的功能性的动作可包括 VMM 112 的一部分上的大量活动。

VMM 112 的活动及其特性不应当限制本发明的范围。

除了特许事件之外，还存在大量“非特许事件”，它们在客户软件的操作期间出现，但不需要由 VMM 112 处理，因而不要求控制向 VMM 112 转移。在一个实施例中，提供一种过滤机制来区分特许事件与非特许事件。通过这种过滤机制，利用一个或多个过滤标准来评估与当前事件(例如异常)相关联的信息，从而确定对当前事件的控制是属于客户软件还是被转移到 VMM 112。这些过滤标准在本文中称作控制转移过滤标准。与当前事件相关联的信息在本文中称作故障信息。

在另一个实施例中，提供一种过滤机制以识别可能被忽视的某些非特许事件，因为在当前事件时存在的其它因素表明客户软件可完成期望的操作，而不损害 VMM 112 或其它 VM 的安全性及正确操作。在此实施例中，利用一个或多个过滤标准来评估与当前事件相关联的故障信息，从而确定当前事件是否可被忽视(即，是否允许客户软件完成期望的操作)。这些过滤标准在本文中称作故障过滤标准。

在一个实施例中，过滤标准(控制转移过滤标准和/或故障过滤标准)采用在虚拟机控制结构(VMCS)122 中的一个或多个指定字段中存储的数据来定义。不同的 VM 可采用来自不同 VMCS 存储映象的数据，但图 1 中仅示出一个这样的 VMCS。VMCS 122 可驻留在存储器 120 中，并且可由处理器 118 维护。应当指出，其它任何数据结构(例如芯片内高速缓存、文件、查找表等)可用来存储 VMCS 122 或者与过滤机制相关联的字段，而不失一般性。下面更详细地描述过滤机制的各种实施例。

在一个实施例中，如果过滤机制确定当前事件必须由 VMM 112 处理，则控制被转移到 VMM 112。然后，VMM 112 可处理该事件，以及把控制重新转移到客户软件。在一个实施例中，控制从 VMM 112 到客户软件的转移通过执行特殊指令来实现。控制从 VMM 到客户软件的转移在本文中称作 VM 入口，控制从客户软件到 VMM 的转移在

本文中称作 VM 出口，VM 出口的可能原因(例如特许的异常和中断以及特许的平台事件)在本文中称作虚拟化事件。

在一个实施例中，当 VM 出口出现时，控制在 VMCS 122 中描述的特定入口点(例如指令指针值)被传递给 VMM 112。在另一个实施例中，控制在通过重定向结构(例如 Intel[®]Pentium[®]IV 的处理器指令集体系结构(ISA)中的中断描述符表(在本文中称作 IA-32 ISA))引导之后被传递到 VMM 112。或者，本领域已知的其它任何机制都可用来把控制从客户软件转移到 VMM 112。

包括异常、中断和平台事件的特许和非特许事件在本文中称作故障。故障可由处理器上指令的执行来产生。例如，访问存储器的操作可因调页和分段保护机制而产生各种故障。各故障与故障信息相关联。故障信息可表征为动态、静态或半动态故障信息。动态故障信息在故障时或接近故障时产生。动态故障信息的实例包括误码，它由异常产生，并直接取决于出错操作的特性或者准备通过导致页面错误的写存储器操作写入存储器的数据值。

静态或半静态故障信息可能具有相同的值，而与故障的定时无关。半静态故障信息的一个实例是各种控制寄存器中很少改变的位的设定，例如 IA-32 ISA 中 CR0 寄存器中的高速缓存禁用(CD)或写保护(WP)位。静态故障信息的一个实例是处理器实现的版本(例如通过 IA-32 ISA 中的 CPUID 指令来报告那样)。

一般来说，故障信息可包括故障标识符、相关故障误码、与故障相关联的附加数据值或者这些数据项的任何组合。故障标识符可以是用来区分这个特定故障与其它故障的值。误码可包含多个值，其中每个值指示特定条件的出现。附加数据值可表示连接到出错指令或故障触发条件的其它任何数据。另外，附加数据值可表示在故障产生期间计算的数据。附加数据值的一些实例包括在故障时被请求写入特定位置的数据、在故障时被访问的地址、导致故障的指令的地址、在故障时存储器的状态等。

可参照 IA-32 ISA 中的页面错误来说明故障信息的一个实例。在 IA-32 ISA 中，页面错误由等于 14 的故障标识符来标识。因此，当页面错误出现时，通过经由条目 14 处的中断描述符表(IDT)引导到故障处理器，控制被传递到故障处理器。(产生页面错误的)被访问地址被存储在控制寄存器(CR2)中。另外，在把控制传递到故障处理器之前，页面错误产生误码，它被推入堆栈，供故障处理器使用。误码包含四位，它通知故障处理器什么条件导致页面错误。具体来说，误码中的位 0 表明错误是否由用于地址转换的页面表中的非当前页面引起，误码中的位 1 表明错误访问是否为写入，误码中的位 2 表明访问是否在处理器处于用户模式时被发起，以及误码中的位 3 表明错误是否由页面目录中设置为 1 的保留位引起。

附加数据值可与页面错误相关联。相关附加数据值的实例包括导致页面错误的被访问地址(CR2)、出错指令的地址、客户软件在出现页面错误时尝试写入页面表分级结构中的不可写页面的数据值、物理和客户线性存储器中的页面表的地址等等。

现在将描述利用控制转移过滤标准的过滤机制的一些实施例。控制转移过滤标准可手动或以编程方式来定义。控制转移过滤标准随特定故障、误码形式、ISA、附加数据值的存在、VMM 的特性及其它因素而改变。控制转移过滤标准的复杂度和表达标准所需的元素数量取决于误码中包含的值的数量、附加数据值的数量、以及需要(和不需要)引起向 VMM 的转换的误码值和/或附加数据值的可能组合的数量。例如，当需要考虑大量这类组合时，控制转移过滤标准可能要求对误码和/或其它故障信息及一组预定数据执行若干运算(例如算术和/或布尔逻辑运算)。

在一些实施例中，预定数据存储在 VMCS 的指定字段中，如上所述。在其它实施例中，预定数据可被硬编码(例如在计算机程序、可编程逻辑、微码或处理器的硬编码逻辑中)。

图 2 是因故障而利用控制转移过滤标准过滤 VM 出口的过程 200

的一个实施例的流程图。该过程可通过可包括硬件(例如电路、专用逻辑、可编程逻辑、微码等)、软件(例如运行于通用计算机系统或专用机器上)或者两者的组合的处理逻辑来执行。

参照图 2, 过程 200 从处理逻辑接收故障信息(处理框 202)开始。故障信息可涉及在允许操作继续进行到完成的情况下已出现或将出现的故障。故障信息包括标识该故障的故障标识符。在一些实施例中, 故障信息还包括与该故障相关联的误码和/或附加数据值。

在判定框 204, 确定故障信息是否满足一个或多个控制转移过滤标准。如果在判定框 204 进行的确定为否定, 即, 故障信息不满足控制转移过滤标准, 则控制转到 VMM(处理框 206), 它处理该故障, 然后可把控制重新转移到客户 OS。否则, 如果在判定框 204 进行的确定为肯定, 即, 故障信息满足控制转移过滤标准, 则控制仍属于客户软件。然后, 可允许客户软件处理该故障。

在备选实施例中, 如果故障信息不满足控制转移过滤标准, 则控制被转到 VMM(如果故障信息满足控制转移过滤标准, 则控制仍属于客户软件)。

如上所述, 控制转移过滤标准的复杂度取决于各种因素。在一个实施例中, 控制转移过滤标准可能只要求将故障信息与某个值进行比较(例如, 如果误码大于 10, 则产生 VM 出口, 或者如果误码等于 0×4 , 则产生 VM 出口, 等等)。在另一个实施例中, 控制转移过滤标准可能要求将故障信息与若干值进行比较。例如, 需要引起到 VMM 的转换的误码值可事先确定, 以及当前误码可与这些预定值进行比较以识别匹配。预定值可存储在指定字段中(例如在图 1 的 VMCS 122 中), 或者被硬编码在计算机程序、可编程逻辑、微码或处理器的硬编码逻辑中。在其它实施例中, 可能要求过滤机制对故障信息和一组预定值执行一个或多个布尔逻辑和/或算术运算。

图 3-5 说明因故障而利用控制转移过滤标准过滤 VM 出口的过程的示范实施例。这些过程可通过可包括硬件(例如电路、专用逻辑、

可编程逻辑、微码等)、软件(例如运行于通用计算机系统或专用机器上)或者两者的组合的处理逻辑来执行。

参照图 3, 过程 300 从处理逻辑在客户软件操作期间检测故障的出现(处理框 301)开始。在处理框 302, 处理逻辑接收标识故障并且可包含与故障有关的其它信息的故障信息。在判定框 303, 处理逻辑确定故障信息是否包含与故障相关联的误码。如果该确定为肯定, 则处理逻辑确定(在处理框 304)以下表达式的值(真或假):

$$EC \text{ AND } MASK == MATCH \quad (1)$$

其中 EC 为误码, AND 为逐位布尔逻辑算子, “==” 为逐位比较算子, MASK 为在第一字段、本文称作掩码字段中存储的预定数据, 以及 MATCH 为在第二字段、本文称作匹配字段中存储的预定数据。

各掩码字段和匹配字段的大小取决于相应误码的大小。在一些实施例中, 掩码字段和匹配字段的宽度与误码的大小相同。在其它实施例中, 掩码字段和匹配字段的宽度可以与误码的大小不同。例如, 掩码字段和匹配字段可小于误码, 并且可以仅把某些位映射到表达式 1 中。

此外, 在图 3 所示的一个实施例中, 处理逻辑还采用重定向图中的相应控制位来进一步确定是否产生 VM 出口。重定向图表示由 VMM 维护的一组控制位, 用于配置哪些虚拟化事件将产生 VM 出口。被咨询的控制位对应于导致当前故障的虚拟化事件。例如, 虚拟化事件可以是页面错误、外部中断或客户软件对调试寄存器的访问, 其中的每个在重定向图中具有相关位。在一个实施例中, 重定向图包含在图 1 的 VMCS 122 中。在一个实施例中, 重定向图是 VMCS 122 中的单字段。在其它实施例中, 在 VMCS 122 中要求多个字段来详细描述重定向图。

如果在判定框 303 作出的确定为否定(即故障信息没有包含与此故障相关联的误码), 或者处理逻辑在判定框 304 确定表达式 1 中的等式成立(即经由逐位 AND 算子与掩码字段中存储的数据组合的误

码匹配在匹配字段中存储的数据), 则在一个实施例中, 重定向图中的相应控制位的值被用来确定是否产生 VM 出口(处理框 306)。例如, 如果控制位被置位, 则处理逻辑将产生 VM 出口(处理框 313); 否则, 处理逻辑将把故障传递给客户软件(处理框 312)。

如果表达式 1 中的等式不成立, 则处理逻辑对重定向位图中的控制位的值取反(处理框 308), 并使用这个取反值来确定是否产生 VM 出口(处理框 310)。如果取反值被置位, 则处理逻辑产生 VM 出口(处理框 314); 否则, 故障将被引导到客户软件(处理框 312)。在另一个实施例中, 如果如判定框 304 中所确定的那样, 等式不成立, 则处理逻辑可使用控制位的实际值, 如果等式成立, 则使用控制位的取反值。

在图 3 的过程 300 中掩码字段和匹配字段与控制位的使用可采用与 IA-32 ISA 页面错误相关联的误码(本文中称作页面错误误码或 PFEC)的各种值作为实例来说明。以下实例假定在表达式 1 的等式不成立时存在控制位的取反(如图 3 所示)。

如上所述, PFEC 包括四位。为了实现 PFEC 值的所有可能组合所需的结果, 掩码字段、匹配字段和控制位中的每个应当具有特定设定。例如, 为了对所有页面错误产生 VM 出口, 控制位可被设置为 1, 掩码字段的值将设置为 0×0 , 以及匹配字段的值将设置为 0×0 。或者, 为了得到同样的性能, 控制位可被设置为 0, 掩码字段设置为 0×0 , 以及匹配字段设置为 $0 \times F$ (注意, 有匹配、掩码字段及控制位值的许多设定都提供相同的功能性)。在另一个实例中, 为了对于因监督程序写入当前页面而导致的页面错误产生 VM 出口, 控制位将设置为 1, 掩码字段的值将设置为 $0 \times F$, 以及匹配字段的值将设置为 0×3 。这些值确保只对产生等于 0×3 的误码的页面故障出现 VM 出口。在又一个实例中, 为了对于因非当前页面或保留位违反而导致的页面错误产生 VM 出口, 控制位将设置为 0, 掩码字段的值将设置为 0×9 (即只有位 0 和 3 被设置为 1), 以及匹配字段的值将设置为 0×1 。这将对

指示当前页面(即位 0 被设置为 1)以及无保留位违反(即位 3 被清除为 0)的那些之外的所有页面错误产生 VM 出口。

在另一个实施例(未示出)中,没有使用控制位。也就是说,处理逻辑根据表达式 1 的结果来确定是否把控制转到 VMM(即产生 VM 出口)。如果表达式 1 中评估的等式成立(即经由逐位 AND 算子与掩码字段中存储的数据组合的误码匹配在匹配字段中存储的数据),则满足控制转移过滤标准,以及与误码相关联的故障将由客户操作系统处理。否则(即经由 AND 算子与掩码字段中存储的数据组合的误码不匹配在匹配字段中存储的数据),则不满足控制转移过滤标准,并产生 VM 出口,把控制转到 VMM。

图 3 所示实施例中使用的控制位的取反增加了实现误码值的各种组合的预期结果(即 VM 出口结果或无 VM 出口结果)时的灵活性。例如,下表说明包含两位的误码的值的各种组合的两种可能的预期结果:

误码位		预期结果 1	预期结果 2
0	0	出口	无出口
0	1	无出口	出口
1	0	出口	无出口
1	1	出口	无出口

如果掩码字段位的值等于(1 1),匹配字段的值等于(0 1),以及控制位等于 0,则图 3 所示的过程 300 的实施例可得到预期结果 1。如果掩码字段位的值等于(1 1),匹配字段的值等于(0 1),以及控制位被设置为 1,则可得到预期结果 2。

应当指出,预期结果 2 不要求使用控制位(即预期结果 2 仅要求掩码=(1 1)以及匹配=(0 1))。但是,如果不使用控制位的取反,或者更多字段没有包含在确定是否产生 VM 出口的过程中,则无法得到预期结果 1。

采用四个指定字段用于确定是否产生 VM 出口的控制转移过滤过程的一个实施例如图 4 所示。参照图 4, 过程 400 可通过可包括硬件(例如电路、专用逻辑、可编程逻辑、微码等)、软件(例如运行于通用计算机系统或专用机器上)或者两者的组合的处理逻辑来执行。

过程 400 从处理逻辑在客户软件操作期间检测故障的出现(处理框 401)开始。在图 4 所示实施例中, 假定所有故障产生误码, 在处理框 402 处理逻辑接收这些误码。随后, 处理逻辑采用逐位 AND 算子将误码与第一掩码字段中存储的数据组合(处理框 404)以及与第二掩码字段中存储的数据组合(处理框 406)。即, 第一组合的结果 $INT1=EC \text{ AND } MASK1$ 以及第二组合的结果 $INT2=EC \text{ AND } MASK2$ 。

此外, 在判定框 408, 处理逻辑确定第一组合 $INT1$ 是否匹配第一匹配字段中存储的数据(MATCH1), 或者第二组合 $INT2$ 是否匹配第二匹配字段中存储的数据(MATCH2)。如果找到任何匹配, 则处理逻辑产生 VM 出口(处理框 410)。或者, 如果没有发现任何匹配, 则故障被插入客户操作系统(处理框 409)。

控制转移过滤标准可采用更复杂的算术或布尔逻辑和/或附加字段, 以便为预期结果提供更大的灵活性, 如图 5 所示。参照图 5, 过程 500 可通过可包括硬件(例如电路、专用逻辑、可编程逻辑、微码等)、软件(例如运行于通用计算机系统或专用机器上)或者两者的组合的处理逻辑来执行。

过程 500 从处理逻辑在客户软件操作期间检测故障的出现(处理框 501)开始。在处理框 502, 处理逻辑接收故障信息。如处理框 503 所确定的那样, 如果故障信息包含误码, 则处理逻辑采用逐位 AND 算子将误码与第一掩码字段中存储的数据组合(处理框 504)以及与第二掩码字段中存储的数据组合(处理框 506)。即, 第一组合的结果 $INT1=EC \text{ AND } MASK1$ 以及第二组合的结果 $INT2=EC \text{ AND } MASK2$ 。

此外, 在判定框 508, 处理逻辑确定在第一组合 $INT1$ 中是否设

置了任何位，或者第二组合 INT2 是否匹配在匹配字段中存储的数据 (MATCH)。在一个实施例中(图 5 中未示出)，如果任一个确定为肯定，则处理逻辑产生 VM 出口。否则，故障被插入客户 OS。

在另一个实施例中(图 5 中所示)，与 VM 出口有关的确定还取决于重定向图中的相应控制位。明确地说，如果在判定框 508 进行的确定为肯定(或者如果在判定框 503 进行的确定为否定，即故障信息没有包含误码)，则处理逻辑还在判定框 510 确定控制位是否指定 VM 出口(例如控制位被设置为 1)。如果控制位指定了 VM 出口，则处理逻辑产生 VM 出口(处理框 512)。如果控制位没有指定 VM 出口(例如控制位被清零)，则故障被插入客户 OS(处理框 518)。

否则，如果在判定框 508 作出的确定为否定，则处理逻辑对控制位的值取反(处理框 514)。如果取反值指定了 VM 出口(判定框 516)，则处理逻辑产生 VM 出口(处理框 512)。如果取反值没有指定 VM 出口，则故障被插入客户 OS(处理框 518)。

虽然图 3-5 所示的实施例采用某些算子(即逐位 AND、比较等)，但各种各样的其它算子也可与过滤标准配合使用，而不失一般性。另外，以上结合图 3-5 所述的那些之外的各种过滤标准也可用来处理在客户软件操作期间出现的故障。此外，上述附加数据值可用来代替与过滤过程中的故障相关联的误码，或者与它们一起使用。

在一些实施例中，附加过滤标准(本文中称作故障过滤标准)可在控制转移过滤标准之前应用于故障信息。在其它实施例中，故障过滤标准在控制转移过滤标准之后应用于故障信息。在另一些实施例中，故障过滤标准而不是控制转移过滤标准被应用于故障信息。故障过滤标准用来确定在客户软件操作期间出现的出错条件是否可被忽略。出错条件表示若允许操作完成、则通常会导致故障的事件。这种出错条件的一个实例是客户软件写入页面表分级结构中的不可写页面的尝试。在一些情况下，出错条件可以被忽略，因为在出错条件时存在的其它因素表明出错条件不会例如损害 VMM 或其它虚拟机的性能、安

全性或正确操作。下面结合图 7 和图 8 更详细地描述这些情况的实例。

图 6 是因故障而利用故障过滤标准和控制转移过滤标准过滤故障和 VM 出口的过程 600 的一个实施例的流程图。该过程可通过可包括硬件(例如电路、专用逻辑、可编程逻辑、微码等)、软件(例如运行于通用计算机系统或专用机器上)或者两者的组合的处理逻辑来执行。

过程 600 从处理逻辑在客户软件操作期间检测出错条件的出现(处理框 602)以及分析与出错条件有关的故障信息(处理框 604)开始。这种故障信息可包括故障标识符、误码、出错地址、待写入的数据、页面表地址等。

随后,处理逻辑确定故障信息是否满足故障过滤标准(判定框 604)。故障过滤标准要求利用与客户软件的工作有关的规则,评估处理器状态、存储器状态和/或故障信息的一个或多个元素。例如,故障过滤标准可要求将客户软件尝试写入存储单元的数据与当前存储在此存储单元的数据进行比较。故障过滤标准可要求单独检验处理器状态、存储器状态和/或故障信息的一个元素,或者多次检验各种元素。故障过滤标准的复杂度可随特定出错条件、与出错条件有关的操作规则、ISA、VMM 的特性、应用及其它因素而改变。下面结合图 7 和图 8 更详细地论述示范过滤标准。

如果满足故障过滤标准,则处理逻辑忽略出错条件,以及允许客户软件继续进行,好像没有检测到出错条件一样(处理框 608)。因此,客户软件发起的操作的行为经修改以允许其完成,而不管出错条件的检测如何。

如果不满足故障过滤标准,则处理逻辑利用控制转移过滤标准确定处理该故障的适当实体(判定框 610)。如果故障信息满足控制转移过滤标准,则处理逻辑把故障传递到客户软件,以及允许客户软件处理该故障(处理框 612)。如果故障信息不满足控制转移过滤标准,则

处理逻辑把控制转到 VMM(处理框 614)。

在其它实施例中,如果不满足故障过滤标准,则处理逻辑不使用控制转移过滤标准,而始终把控制转到 VMM。

在其它实施例中,处理逻辑首先利用控制转移过滤标准来确定处理故障的适当实体。然后,如果适当实体为客户软件,则处理逻辑还确定是否满足故障过滤标准。如果满足故障过滤标准,则允许客户忽略故障条件;如果不满足故障过滤标准,则该故障被传递给客户,如上所述。

以下论述假定页面表结构在 IA-32 ISA 中,但应当指出,类似的页面表结构存在于各种 ISA 中,本发明不限于使用 IA-32 ISA。另外,可进行各种简化以便于以下论述。例如,没有包含物理地址扩展(PAE)和页面大小扩展(PSE)标志的作用,没有说明基于环的保护机制,没有测试存储器存取的对准要求等等。

如本领域熟知的那样,IA-32 ISA 中的页面表由页面目录条目(PDE)和页面表条目(PTE)组成。每个 PDE 和 PTE 包含控制存储器页面的可存取性的各种位字段。例如,“P”位把页面标记为当前(1)或非当前(0),“R/W”位表明页面是只读(0)还是可读写(1),“U/S”位表明页面是否要求监督程序特权,页面帧编号(PFN)是否包含逻辑页面所在的物理地址的一部分,等等。CR3 是 ISA 中的控制寄存器,它包含页面目录的基物理地址(和附加标志)。页面目录基物理地址等于 $CR3[31:12] \ll 12$ (即,页面目录基地址的低 12 位为 0,高 20 位处于 CR3 的高 20 位中)。

为了防止一个虚拟机中的错误或恶意代码损害在 VMM 或另一个虚拟机上运行的代码,需要把客户软件的访问限制为基础物理机器的物理存储资源。在提供了由客户 OS 和 VMM 分别管理的独立页面表分级结构的一些体系结构中,一种把客户软件的访问限制为物理存储资源的方法包括客户页面表分级结构的 VMM 修改许可,从而防止客户软件对受保护页面(例如属于 VMM 或其它 VM 的页面)的读或写

操作。另外，客户页面表经修改以防止对属于存储客户页面表的 VM 的一些页面的写操作。一旦已经进行这些许可修改，则客户软件改变页面表的尝试将导致页面错误事件，这是 VMM 可通过 VM 出口观察的。VMM 可检查所尝试的访问，并确定要采取的适当动作。例如，它可允许、不允许或修改所尝试的访问。例如，可能不允许客户软件映射属于 VMM 或另一个 VM 的物理页面。由于客户软件对页面表的修改频繁出现，因此与退出到 VMM 相关的、用于筛选每次尝试的客户页面表改变的开销可能强加重大的性能负担。

为了简化页面表的管理，若干操作系统、例如 Microsoft Windows XP 采用递归页面目录。换言之，页面目录条目(PDE)将设置为把页面目录页引用为页面表页。这种把同一个页面用作页面目录和页面表的用法使所有页面表页在通过自指的 PDE 访问的线性地址空间的 4MB 区域中是可访问的。给定自指 PDE 的使用，则可计算用来映射任何特定线性地址的 PTE 或 PDE 的线性地址。在采用自指 PDE 的操作系统中，典型的页面表编辑是通过这种自映射区域进行的(即，对页面表的编辑是利用自映射 PDE 通过写入采用处于 4MB 区域的线性地址的存储器进行的)。不采用这种自映射 PDE 的页面表编辑在数量上极少。

其中公共页面用作页面目录页以及页面表页的页面表在本文中称作自映射页面表。本领域的技术人员非常清楚，本发明还适用于其它方法，其中页面表条目以组织方式映射到线性地址，无论这种映射是在页面表分级结构的多层上结构再用的结果，还是通过 OS 惯例。

在本发明的一个实施例中，过滤机制用来标识不要求退出到 VMM 的页面映射编辑。在此实施例中，用于过滤标准的附加字段被添加到 VMCS。

在一个实施例中，客户操作系统所用的页面表是自映射的，即，页面表分级结构中的页面目录条目(PDE)之一重新指向页面目录(PD)页的基部。如上所述，自映射页面表的概念是本领域都熟知的，以及

由操作系统、例如 Microsoft 的 Windows XP 使用。在一个实施例中，通过确保页面目录基部线性地址(本文中称作 PTBASE)满足一组条件来支持自映射。这组条件可检验 PTBASE 值为 4MB 对准的、即 $PTBASE[21:0]=0$ ，以及页面表包含适当的自映射条目、即单元 $((CR3[31:12] \ll 12) + (PTBASE[31:22] \ll 2))$ 中条目中的 PFN 等于 $CR3[31:12]$ 。在一个实施例中，PTBASE 值存储在 VMCS 中，由过滤机制使用，如以下所述。

假定自映射页面表正在使用，则可确定任何给定地址是否处于页面表中(即，是否 $ADDR[31:22]=PTBASE[31:22]$ ，其中 ADDR 为被访问的地址)。另外，可确定该地址是否处于 PDE 中(即地址是否处于页面表中，并且 $ADDR[21:12]=PTBASE[31:22]$)或者处于 PTE 中(即该地址是否处于页面表中，但不在 PDE 中)。

图 7 和图 8 说明利用故障过滤标准过滤页面错误的过程的两个示范实施例。该过程可通过可包括硬件(例如电路、专用逻辑、可编程逻辑、微码等)、软件(例如运行于通用计算机系统或专用机器上)或者两者的组合的处理逻辑来执行。在图 7 和图 8 所示的实施例中，VMM 通过确保映射页面表自身的客户页面表条目(客户 PTE)具有只读访问来保护物理存储器。因此，当客户软件尝试改变页面表条目时，页面错误条件出现。这些实施例可与采用自映射页面表的客户 OS 配合使用，如上所述，或者与没有采用这个页面表结构的客户 OS 配合使用，但采用自映射页面表的客户 OS 的使用提供更大的性能提高。

在图 7 所示的实施例中，说明了故障过滤标准的集合，它允许完成对 PTE 的写入，在客户操作系统没有尝试把“P”位设置为 1 时没有传递故障或产生 VM 出口，同时仍然保护物理存储空间以及制止为客户软件提供对页面表的完全控制。这个规则的理论基础在于，标记了非当前的 PTE (即“P”位等于 0)无法映射物理存储页面。因此，由客户软件改变的 PTE 无法映射物理存储页面，因此无法干扰 VMM 或者在另一个 VM 上运行的软件的操作。因此，不需要通知 VMM 页

面表修改, 以及将允许页面表写入继续进行, 而没有导致页面错误或 VM 出口。

参照图 7, 处理逻辑从在客户软件操作期间检测页面错误条件(处理框 702)以及接收与页面错误条件有关的页面错误信息(处理框 704)开始。

随后, 一组故障过滤标准被应用于页面错误信息。明确地说, 处理逻辑确定出错访问是否为写操作(判定框 706)、写入 PTE(判定框 708), 以及不是把“P”位设置为 1 的尝试(即 $DATA.P=0$, 其中 DATA 为客户正试图写入的值)(判定框 710)。如果这些确定为肯定的, 则允许客户操作系统的访问完成(即允许写入以修改存储器), 而没有产生故障或 VM 出口(处理框 714)。如果处理框 706、708 和 710 中的确定的任一个为否定, 则控制被转移到处理框 712 以确定 VM 出口是否要被产生, 或者故障将被引导到客户软件, 如结合图 2-5 更详细描述。

在另一个实施例中(附图中未示出), 写入必须是对 PTE 进行的条件可被取消, 或者采用例如只要当前位未被设置就允许对 PTE 或 PDE 写入的测试来代替。

参照图 8, 说明更复杂的故障过滤标准集合, 它包括以上结合图 7 所述的故障过滤标准以及其它一些标准。故障过滤标准的这个集合允许客户操作系统修改 PTE 中除 PFN、读/写以及当前位之外的位, 而不要求页面错误或产生 VM 出口。另外, 允许读/写和当前位的某些修改, 而不要求页面错误或产生 VM 出口。

处理逻辑从在客户软件操作期间检测页面错误条件(处理框 802)以及接收与页面错误条件有关的页面故障信息(处理框 804)开始。随后, 处理逻辑确定出错访问是否为写操作(判定框 806)、写入 PTE(判定框 808), 以及不是把“P”位设置为 1 的尝试(即 $DATA.P=0$)(判定框 810)。如果这些确定均为肯定, 则允许客户操作系统的访问完成(即允许写入以修改存储器), 而没有产生故障或 VM 出口(处理框 814)。如果处理框 806 或 808 中的确定为否定, 则控制进入处理框 812, 确

定是否要求 VM 出口，如参照图 2-5 所述。

如果处理框 810 中的确定为否定，则应用附加故障过滤规则。明确地说，处理逻辑确定 PTE 是否已经在 PTE 中标记为当前(即， $ADDR \rightarrow P = 1$ ，其中 ADDR 为客户软件正尝试写入的地址，以及 $ADDR \rightarrow P$ 表示在存储器中位于 ADDR 的数据被解释为 PTE 时的 P 位)(判定框 816)，PFN 是否没有改变(即 $ADDR \rightarrow PFN = DATA.PFN$)(判定框 818)，以及客户是否正尝试把 PTE 映射的页面标记为只读($DATA.R/W = 0$)或者由 PTE 映射的页面是否在存储器中的 PTE 中已经标记为可写($ADDR \rightarrow R/W = 1$)(判定框 820)。如果这些确定为肯定，则允许客户操作系统的访问完成(即允许写入以修改存储器)，而没有产生故障或 VM 出口(处理框 814)。如果处理框 816、818 和 810 中的任何确定为否定，则控制被转移到处理框 812 以确定是否要求 VM 出口。

应当指出，图 7 和图 8 所示的过滤机制将滤除利用自映射 PDE 尝试修改页面表的访问，如上所述。没有采用自映射 PDE 的修改页面表的尝试不会被故障过滤标准滤除(即，在框 708 和 808 中对于写入是否对 PTE 进行的确定将失败)。然后，控制转移过滤标准用来确定是否需要 VM 出口，如图 2-5 所示。从安全性或正确操作的观点来看，这不会产生问题，因为 VMM 能够确定是否在实际编辑 PTE。控制转移过滤机制将配置成对所有写入页面错误产生 VM 出口，以及评估被写入的地址和数据，以确定是否正尝试页面表编辑。

除以上参照图 7 和图 8 所述的那些之外的各种故障过滤标准可以用来过滤故障，而不失一般性。过滤的形式可由各种 VMM 控制字段(例如 PTBASE)来控制，在一个实施例中，这些字段可驻留在 VMCS 中。或者，过滤可由单个启用位来控制，可以被硬编码到处理器实现中，等等。

图 9 是定义控制转移过滤标准和/或故障过滤标准的过程 900 的一个实施例的框图。过程 900 可手动或自动执行。

参照图 9，过程 900 从识别故障信息的哪些组合(例如误码值、故障标识符等)不要求到 VMM 的转换或者应当被允许完成、忽略故障(处理框 902)开始。然后，可用于 VM 出口和/或故障过滤标准的字段最大数量以及字段语义被识别(处理框 904)，以及可用算子(例如布尔、算术等)被识别(处理框 906)。此外，根据在处理框 902-906 所识别的信息，创建过滤标准(处理框 908)。过滤标准可包括一个或多个预定值以及对预定值和误码执行的一个或多个布尔逻辑和/或算术运算。

这样，已经描述了处理在客户软件操作期间出现的故障的方法及装置。应当理解，以上描述只是说明性而不是限制性的。通过阅读和理解以上说明，本领域的技术人员将会十分清楚其它许多实施例。因此，本发明的范围应当参照所附权利要求以及这些权利要求涵盖的完整等效范围来确定。

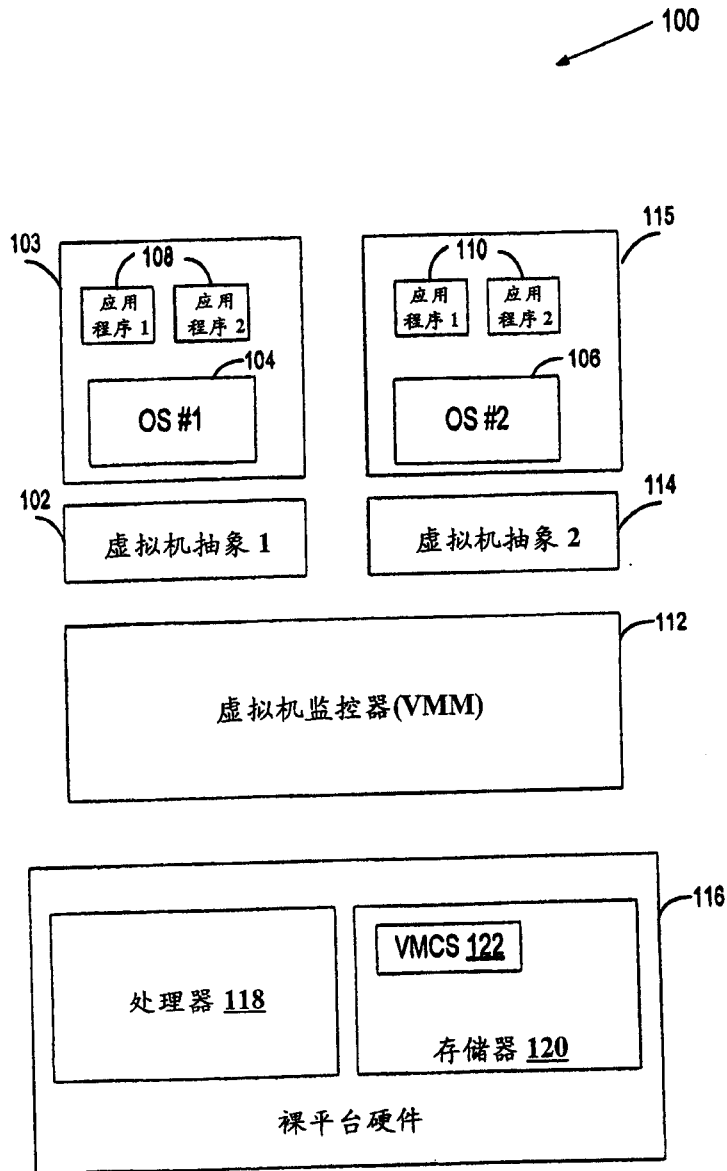


图 1

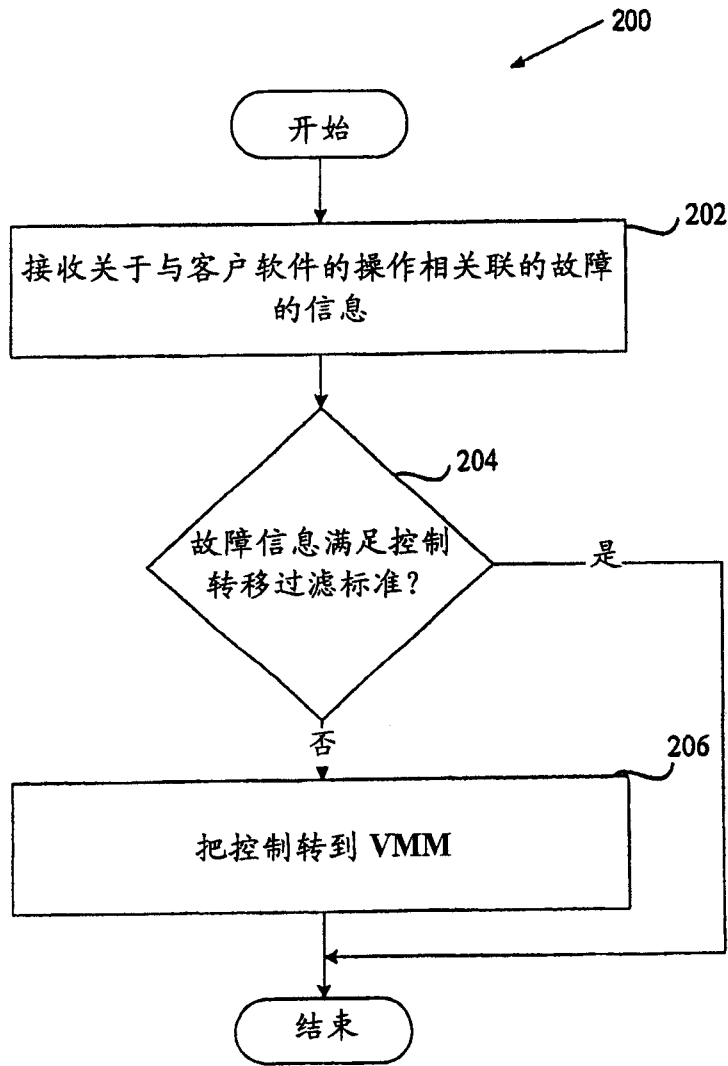


图 2

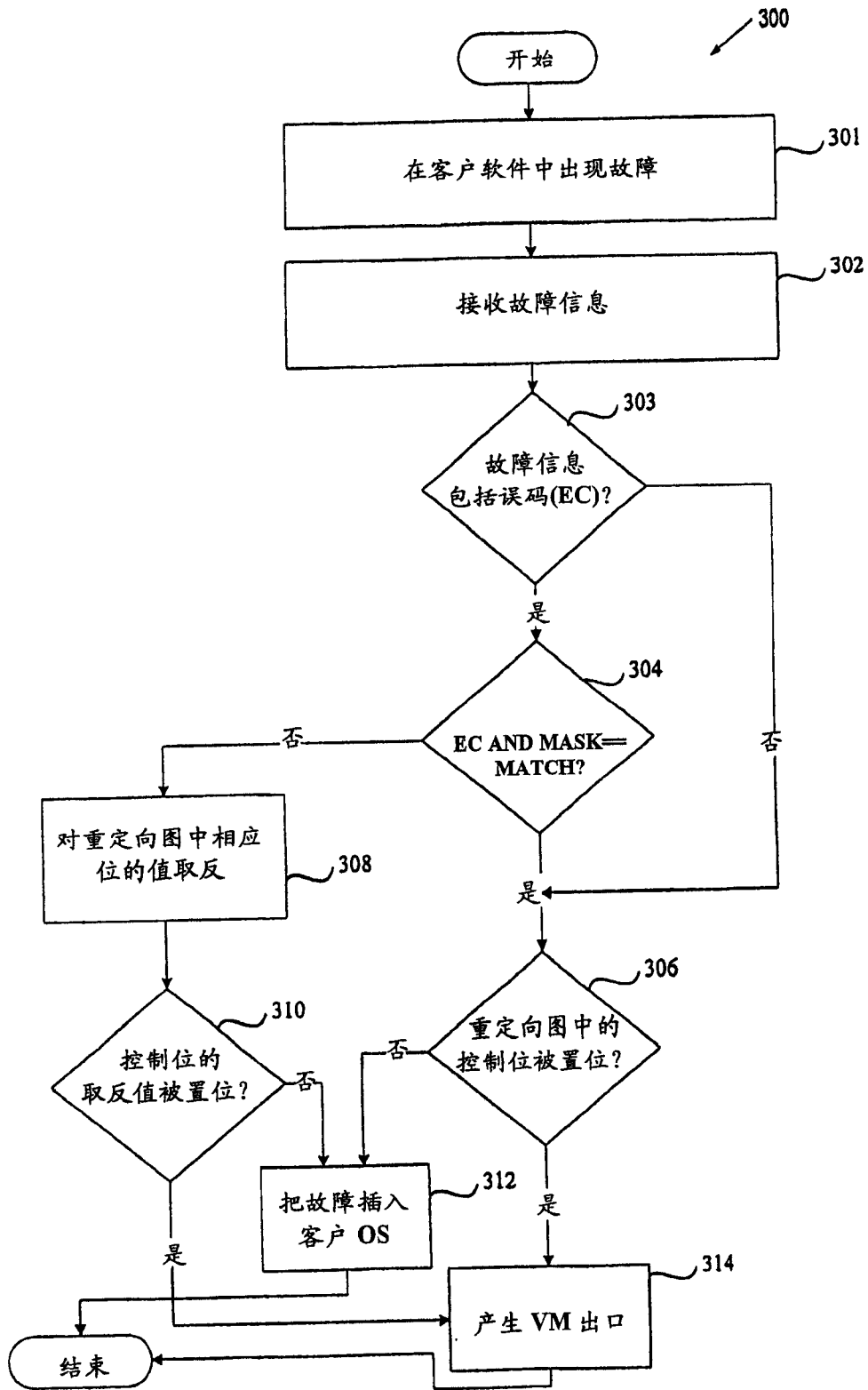


图 3

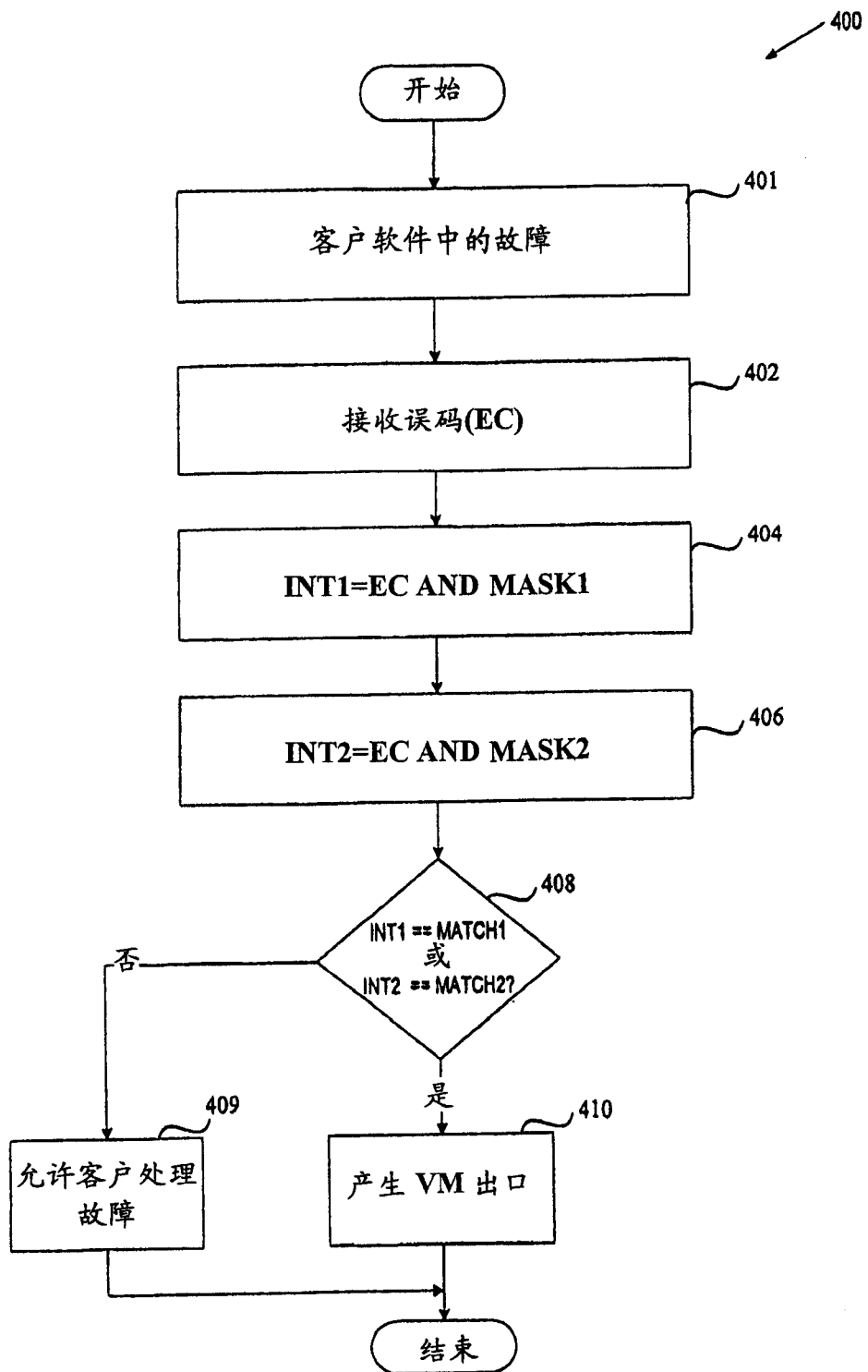


图 4

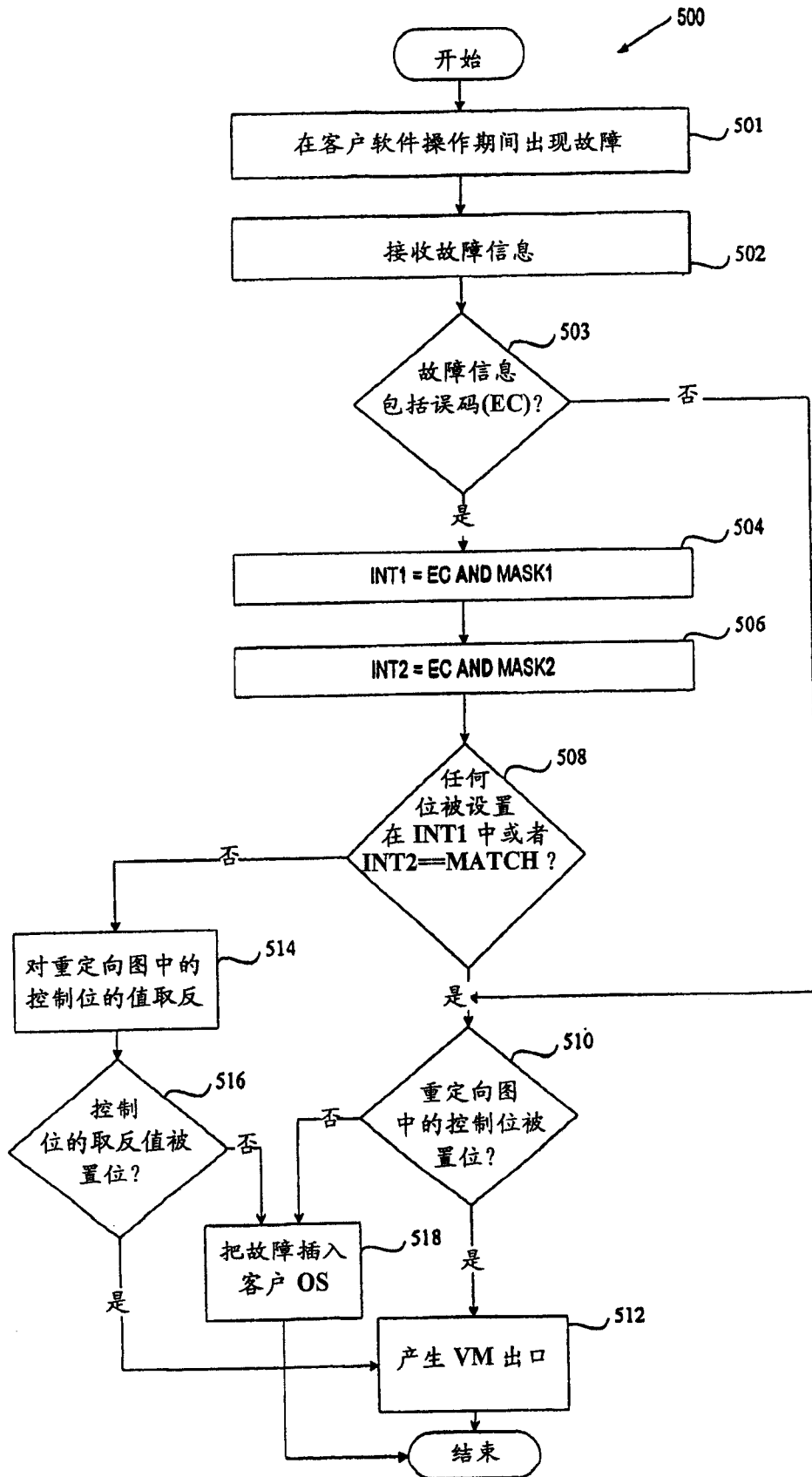


图 5

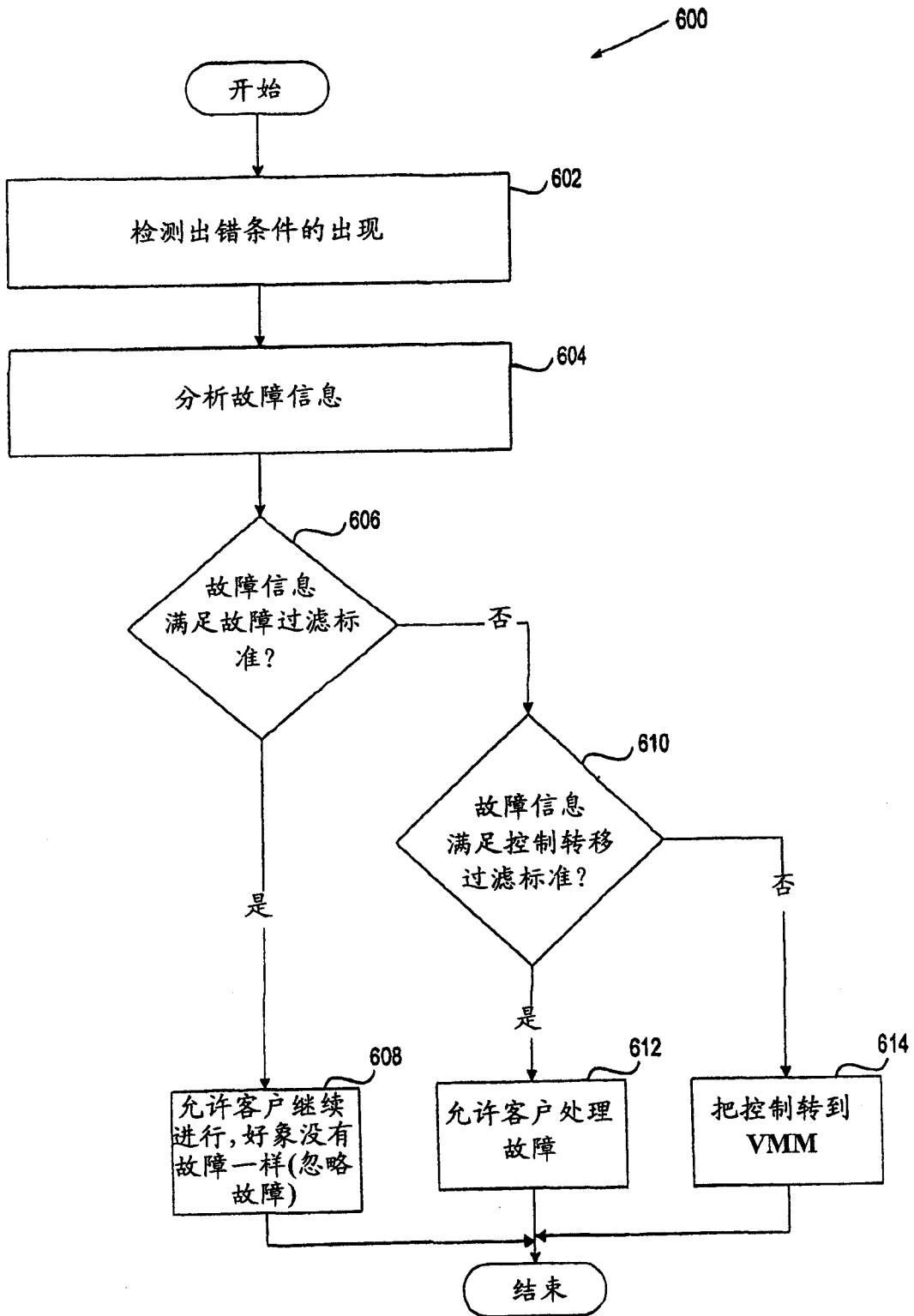


图 6

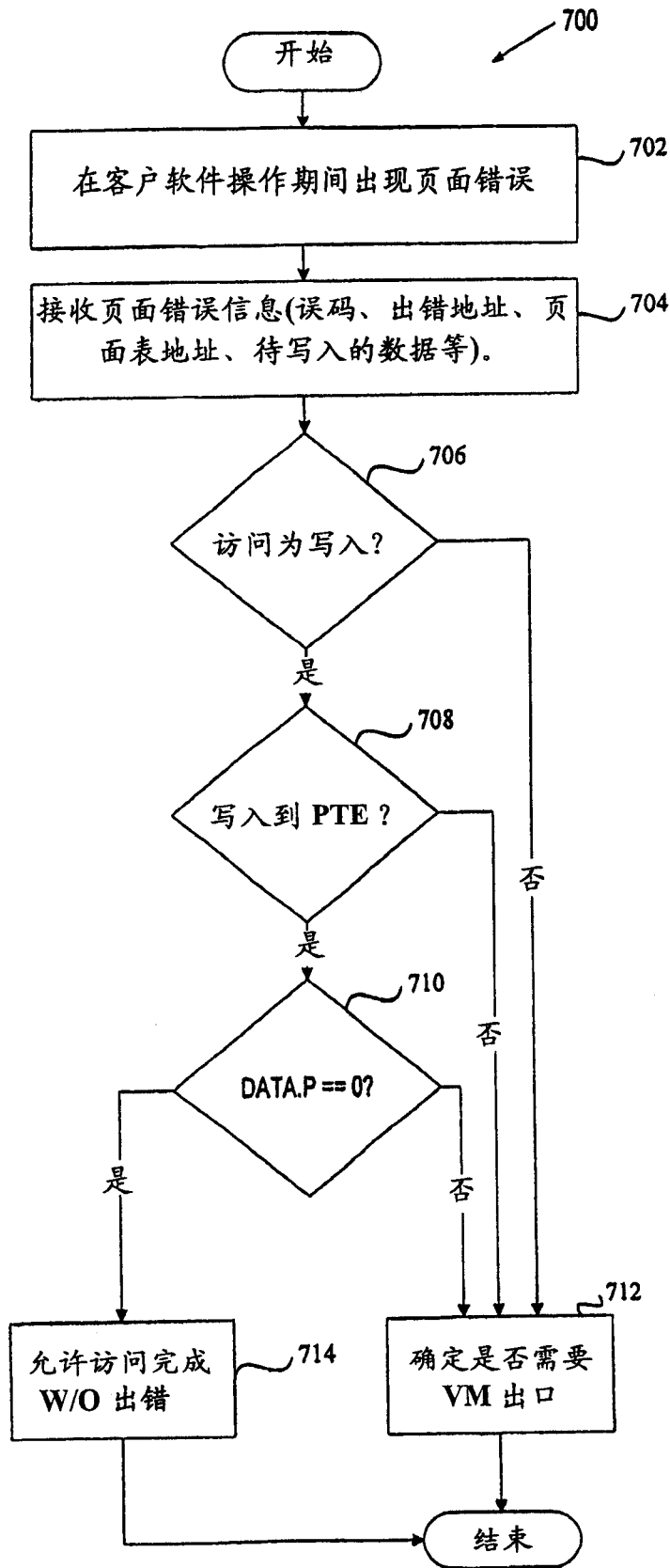


图 7

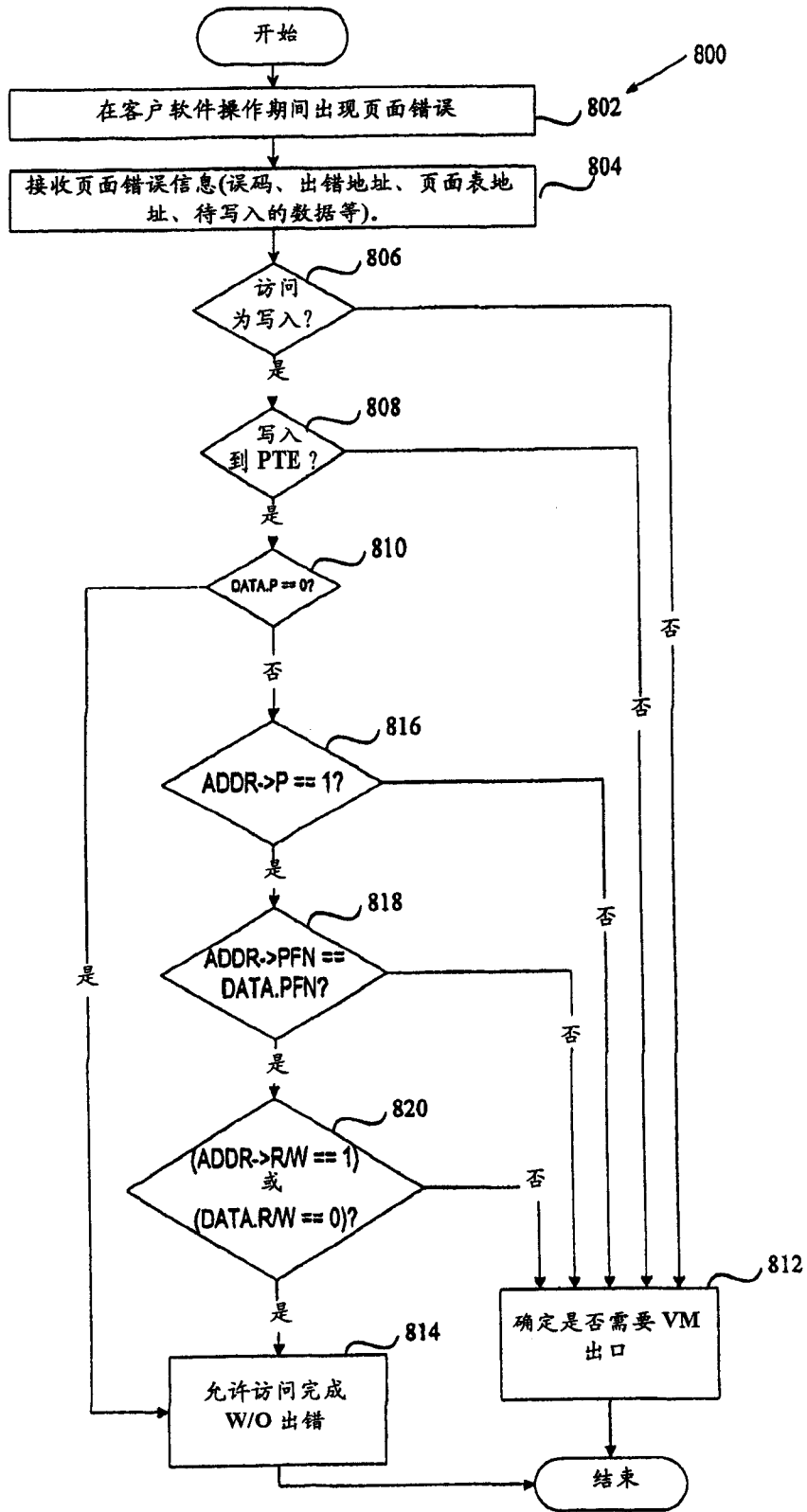


图 8

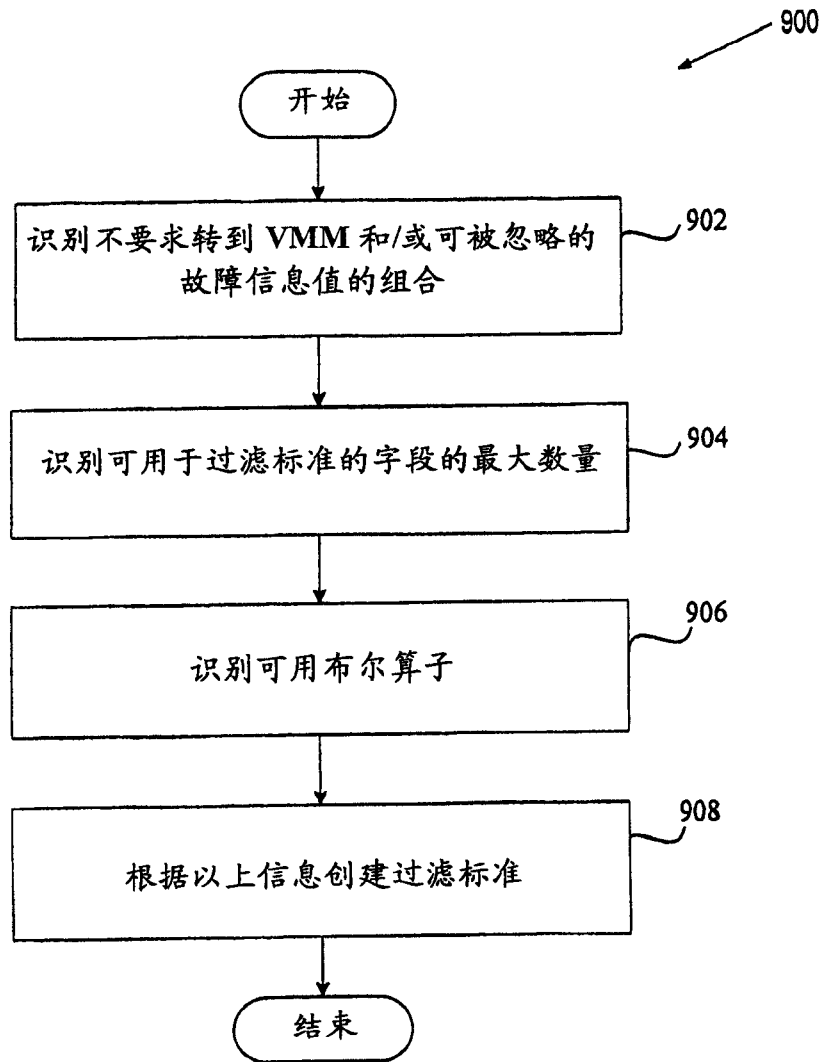


图 9