



(12) 发明专利

(10) 授权公告号 CN 101861569 B

(45) 授权公告日 2014.03.19

(21) 申请号 200880109465.3

(51) Int. Cl.

(22) 申请日 2008.07.24

G06F 11/16 (2006.01)

(30) 优先权数据

60/935,044 2007.07.24 US

12/138,717 2008.06.13 US

(85) PCT国际申请进入国家阶段日

2010.03.24

(56) 对比文件

GB 2425380 A, 2006.10.25,

US 6615366 B1, 2003.09.02,

US 5226152 A, 1993.07.06,

WO 2006045786 A1, 2006.05.04,

CN 1834678 A, 2006.09.20,

(86) PCT国际申请的申请数据

PCT/US2008/071023 2008.07.24

审查员 张雯

(87) PCT国际申请的公布数据

W02009/015276 EN 2009.01.29

(73) 专利权人 通用电气航空系统有限责任公司

地址 美国密执安州

(72) 发明人 J·R·普勒伊特 G·R·赛克斯

T·D·斯库特

(74) 专利代理机构 中国专利代理(香港)有限公司

72001

代理人 朱海煜 徐予红

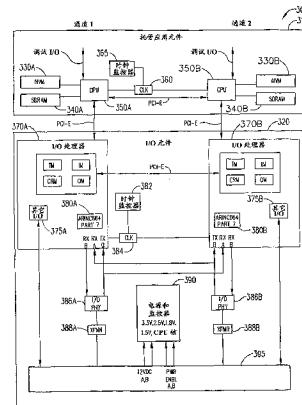
权利要求书1页 说明书10页 附图8页

(54) 发明名称

高集成度和高可用性计算机处理模块

(57) 摘要

一种高集成度 N-通道计算机处理模块 (Module)，其中 N 是大于或等于 2 的整数。该模块包括：每个处理通道一个托管应用元件和 I/O 元件；时间管理单元 (TM)，配置成为由在这 N 个处理通道中的每个处理通道上运行的软件所做的请求确定等效时间值，而不管这 N 个处理通道中的每个处理通道实际上何时接收到请求并对请求采取行动；以及临界区域管理单元 (CRM)，配置成使得能够在所有这 N 个处理通道中识别相应通道内的临界区域并使它们同步。



1. 一种高集成度 N- 通道计算机处理模块 (Module) 系统,N 是大于或等于 2 的整数,所述系统包括 :

每个处理通道内的一个托管应用元件和 I/O 元件 ;以及

时间管理单元 (TM), 配置成为由在所述 N 个处理通道中的每个处理通道上运行的软件所做的请求确定等效时间值, 而不管所述 N 个处理通道中的每个处理通道实际上何时接收到所述请求并对所述请求采取行动 ;以及

临界区域管理单元 (CRM), 配置成使得能够在所有所述 N 个处理通道中识别相应通道内的临界区域并使它们同步。

2. 如权利要求 1 所述的系统, 还包括 :

数据输入管理 (IM) 单元, 配置成确保每个相应通道接收与所述 N 个处理通道中的所有其它处理通道完全相同的一组高集成度数据, 否则输出错误情况 ;以及

数据输出管理 (OM) 单元, 配置成确定相应通道是否输出与所述 N 个处理通道中的所有其它处理通道完全相同的一组高集成度数据, 否则输出错误情况。

3. 如权利要求 1 所述的系统, 其中由所述临界区域管理单元识别的所述临界区域对应于与当前运行的执行线程分开的任何其它执行线程无法抢先的软件内的区域。

4. 如权利要求 1 所述的系统, 其中所述时间管理单元包括 1- 深度缓冲器。

5. 如权利要求 1 所述的系统, 其中所述时间管理单元包括 M- 深度缓冲器, M 是大于或等于 2 的整数。

6. 如权利要求 1 所述的系统, 其中高集成度数据和正常集成度数据都在所述 N 个处理通道上流动, 并且其中仅所述高集成度数据由所述高集成度 N- 通道计算机处理模块进行操作。

7. 如权利要求 1 所述的系统, 其中所述时间管理单元实现为有限状态机。

8. 如权利要求 1 所述的系统, 其中所述临界区域管理单元实现为有限状态机。

9. 一种高集成度 N- 通道计算机处理模块 (Module) 系统,N 是大于或等于 2 的整数,所述系统包括 :

每个处理通道内的一个托管应用元件和 I/O 元件 ;以及

时间管理单元 (TM), 实现为有限状态机, 并配置成为由在所述 N 个处理通道中的每个处理通道上运行的软件所做的请求确定等效时间值, 而不管所述 N 个处理通道中的每个处理通道实际上何时接收到所述请求并对所述请求采取行动 ;

临界区域管理单元 (CRM), 实现为有限状态机, 并配置成使得能够在所有所述 N 个处理通道中识别相应通道内的临界区域并使它们同步 ;

数据输入管理 (IM) 单元, 配置成确保每个相应通道接收与所述 N 个处理通道中的所有其它处理通道完全相同的一组高集成度数据, 否则输出错误情况 ;以及

数据输出管理 (OM) 单元, 配置成确定相应通道是否输出与所述 N 个处理通道中的所有其它处理通道完全相同的一组高集成度数据, 否则输出错误情况 ;

其中高集成度数据和正常集成度数据都在所述 N 个处理通道上流动, 并且其中仅所述高集成度数据由所述高集成度 N- 通道计算机处理模块进行操作。

高集成度和高可用性计算机处理模块

[0001] 相关申请的交叉引用

[0002] 本申请要求 2007 年 7 月 24 日提交的题为“High Integrity and High Availability Computer Processing Module and Method”的临时申请序列号 60/935044 的优先权。

技术领域

[0003] 本文描述的技术涉及用于实现源处理的高集成度和高可用性的计算机处理模块 (Module)，其对托管在该模块上的软件应用 (托管应用) 施加了最小的设计约束，使得它们仍可在典型正常集成度的计算机处理模块上运行。

背景技术

[0004] 计算机处理模块 (Module) 可在源处提供高集成度和高可用性以确保以一定精度检测和隔离故障，并将假警报减至最少。高集成度模块对于飞行器甚至更加重要，其中未及时而准确检测和隔离的故障可导致操作困难。在源处提供高集成度的模块中的故障的正确检测和隔离有时称为在模块或系统内建立故障包容区 (FCZ)、以使得故障不能够传播到发生故障的 FCZ 的外部的能力。再者，重要的是，高集成度模块还应具有非常低的假警报概率，因为每个假警报都可能导致功能的暂时丧失或浪费计算机资源来校正事实上不存在的误报问题。

[0005] 用于在源模块实现高集成度的常规设计需要昂贵的定制电路，以便在模块上的两个或更多微处理器之间实现指令级锁步处理。常规的指令级锁步处理方法为所有托管应用提供高集成度，但可能难以 (或不可能) 用现有技术的微处理器来实现，现有技术的微处理器用于实现需要具有不同时钟恢复电路的多个锁相环 (PLL) 的嵌入式存储器控制器和输入 / 输出支持。

[0006] 需要在模块的源设计处实现高集成度，该模块对托管应用施加最小的设计约束 (即，相同托管应用也可在典型的正常集成度模块上运行)，并且能够利用高速微处理器 (例如，集成处理器)。

发明内容

[0007] 本发明的一方面涉及一种高集成度 (high integrity) N-通道计算机处理模块 (Module)，其中 N 是大于或等于 2 的整数。该模块包括：每个处理通道一个托管 (hosted) 应用元件和 I/O 元件；时间管理单元 (TM)，配置成为在这 N 个处理通道中的每个处理通道上运行的软件所做的请求确定等效 (equivalent) 时间值，而不管这 N 个处理通道中的每个处理通道实际上何时接收到请求并对请求采取行动；以及临界区域管理单元 (CRM)，配置成使得能够在所有这 N 个处理通道中识别相应通道 (lane) 内的临界区域并使它们同步。

附图说明

- [0008] 下面将参考附图描述示范实施例，其中相似的标号描绘相似的元件，并且其中：
- [0009] 图 1 示出期望缓解、以便为托管应用排除故障情况的第一种情形；
- [0010] 图 2 示出期望缓解、以便为托管应用排除故障情况的第二种情形；
- [0011] 图 3 是时间管理 (TM)、临界区域管理 (CRM)、数据输入管理 (IM) 和数据输出管理 (OM) 单元的逻辑框图；
- [0012] 图 4 是示出根据示范实施例的高集成度松散同步的计算机处理模块 (Module) 的框图；
- [0013] 图 5 是示出根据示范实施例的时间管理单元的细节的框图；
- [0014] 图 6 是示出根据示范实施例的临界区域管理单元的细节的框图；
- [0015] 图 7 示出通过利用根据示范实施例的系统和方法排除了潜在故障情况的（图 1 的）第一种情形；以及
- [0016] 图 8 示出通过利用根据示范实施例的系统和方法排除了潜在故障情况的（图 2 的）第二种情形。

具体实施方式

[0017] 在以下描述中，为了说明的目的，阐述了大量具体细节，以便全面理解本文描述的技术。然而，对于本领域技术人员显而易见的是，在没有这些具体细节的情况下也可实现示范实施例。在其它情况下，以示图形式示出结构和装置以便于描述示范实施例。

[0018] 下面参考附图描述示范实施例。这些附图说明用于实现本文描述的模块、方法和计算机程序产品的具体实施例的某些细节。然而，不应将附图解释为施加了可能存在于附图中的任何限制。该方法和计算机程序产品可提供在任何机器可读介质上以便实现它们的操作。可使用现有计算机处理器或由为这个目的或另一目的而结合的专用计算机处理器或由硬连线系统来实现这些实施例。

[0019] 如上面所提到的，本文描述的实施例包括计算机程序产品，计算机程序产品包括用于承载或其上存储有机器可执行指令或数据结构的机器可读介质。这种机器可读介质可以是任何可用介质，它可由通用或专用计算机或具有处理器的其它机器访问。作为示例，这种机器可读介质可包括 RAM、ROM、EPROM、EEPROM、CD-ROM 或其它光盘存储设备、磁盘存储设备或其它磁储存装置、或可用于承载或存储机器可执行指令或数据结构形式的期望程序代码并可由通用或专用计算机或具有处理器的其它机器访问的任何其它介质。当通过网络或另一通信连接（硬连线、无线、或硬连线或无线的组合）将信息传送或提供到机器时，机器适当地将该连接视为机器可读介质。因此，任何这种连接都可适当地称为机器可读介质。以上的组合也包含在机器可读介质的范围内。机器可执行指令包括例如使通用计算机、专用计算机或专用处理机器执行某种功能或某组功能的指令和数据。

[0020] 将在方法步骤的通常上下文中描述实施例，在一个实施例中，方法步骤可由包含例如由联网环境中的机器执行的程序模块形式的机器可执行指令（如程序代码）的程序产品来实现。一般来说，程序模块包括执行特定任务或实现特定抽象数据类型的例行程序、程序、对象、组件、数据结构等。机器可执行指令、关联的数据结构和程序模块代表用于执行本文公开的方法的步骤的程序代码的示例。这种可执行指令或关联的数据结构的特定序列代表用于实现在这些步骤中描述的功能的对应动作的示例。

[0021] 可在联网环境中使用到具有处理器的一个或多个远程计算机的逻辑连接来实现实施例。逻辑连接可包括在此作为示例而非限制性给出的局域网 (LAN) 和广域网 (WAN)。这种联网环境在办公室范围或企业范围的计算机网络、内联网和互联网中是常见的，并且可以使用各种各样的不同通信协议。本领域的技术人员将认识到，这种网络计算环境通常将涵盖许多类型的计算机系统配置，包括个人计算机、手持装置、多处理器系统、基于微处理器的或可编程消费型电子装置、网络 PC、小型计算机、大型计算机等等。

[0022] 也可在分布式计算环境中实现实施例，在分布式计算环境中，由通过通信网络链接（通过硬连线链路、无线链路、或通过硬连线或无线链路的组合）的本地和远程处理装置来执行任务。在分布式计算环境中，程序模块可位于本地和远程存储器存储装置中。

[0023] 用于实现示范实施例的整体或部分的示范系统可包括计算机形式的通用计算装置，它包括处理单元、系统存储器和用于将包括系统存储器在内的各种系统组件耦合到处理单元的系统总线。系统存储器可包括只读存储器 (ROM) 和随机存取存储器 (RAM)。计算机还可包括用于从磁硬盘读取并向其写入的磁硬盘驱动器、用于从可移动磁盘读取或向其写入的磁盘驱动器、以及用于从可移动光盘（如 CD-ROM 或其它光介质）读取或向其写入的光盘驱动器。这些驱动器及其关联的机器可读介质为计算机提供对机器可执行指令、数据结构、程序模块和其它数据的非易失性存储。

[0024] 下面将详细描述第一实施例，它对应于用于在包括计算机处理模块 (Module) 的系统的源处提供高集成度的松散同步的方法。

[0025] 源计算的高集成度当前需要在指令级以锁步方式运行的至少两个处理通道、或一个处理通道和一个监控器。对于在源处理模块的双通道高集成度，要解决的问题可与有限状态机相比拟。也就是说，如果在模块的每个处理通道上运行的软件接收相同的输入（数据、中断、时间等）并且能够对数据执行相同“量”的处理，之后再发送输出或之后再接收新输入，则每个通道将在无故障的情况下产生同样的输出。应注意，这个实施例主要是针对其中每个处理通道具有同样的微处理器的模块进行描述的。然而，这个实施例也适用于在这 N 个通道中的一个或多个通道上具有不一样处理器的模块。在这种情况下，期望每个处理通道将产生同样在规定范围内的输出（例如，可能由于微处理器的浮点单元的差异引起）。

[0026] 有限状态机类比的含义如下。当在模块上运行的软件接收输入时，两个通道上的输入必须是同样的，并且两个通道必须在它们处于完全相同的状态时接收这些输入。输入应视为是明确请求的输入（例如，ARINC653 端口数据、时间戳等）或是由于外部事件（硬件中断、虚拟中断等）而接收的输入。要对由于例如优先级抢先行为而使软件改变其执行线程（状态）的输入给予特别注意。当在模块上运行的软件发送输出时，来自两个通道的数据必须进行比较，之后才能输出。为了确保输出数据比较不会失败（因为不适当的状态同步），负责产生输出数据的软件部分必须在两个通道中都达到相同状态之后才可以比较输出，然后随后传送输出。

[0027] 图 1 和图 2 中所示的情形提供了必须缓解以便（通过模块设计）排除故障情况的两种潜在故障情形的图示。之所以选择这些特定情形是因为相信，可缓解这些故障情况的模块设计具有能够处理（或可扩展成处理）输入数据等效性的更一般设计约束和控制在模块的 N 个通道上运行的软件的同步化的高概率。

[0028] 现在转到图 1，对于双通道高集成度模块描述第一类型的潜在故障情况。在该模块

中,通道 1 和 2 正在松散地同步运行,但是没有增加本文描述的 TM 和 CRM 单元。在这种情况下,松散同步意味着,通道 1 可以在从通道 2 前面或后面的少于一个指令到通道 2 前面或后面的任何数量指令的任何地方。对于图 1 所示的示例,通道 1 在通道 2 的“前面”。这个示例中所用的布尔值的初始条件是“假”。

[0029] 在步骤 1 中,当发生计时器中断时,通道 1 中的进程 1 刚好完成将布尔值设成“真”。通道 2 中的进程 1 未完全有机会将布尔值设成“真”(由此布尔值仍是“假”)。

[0030] 在步骤 2 中,中断使通道 1 和通道 2 中的托管应用切换到进程 2(由于优先级抢先)。

[0031] 在步骤 3 中,通道 1 中的进程 2 和通道 2 中的进程 2 读取布尔值,并发送包含布尔值的状态的输出。通道 1 输出“真”,而通道 2 输出“假”。

[0032] 在步骤 4 中,数据输出管理 (OM) 单元检测到这两个通道之间的失配 (mis-compare)。如果模块提供了这两个计算通道之间的适当同步,则这是本来可以预防的故障类型 (因此增大可用性)。

[0033] 现在转到图 2,对于双通道高集成度模块描述第二类型的潜在故障情况。在该系统中,通道 1 和 2 正在松散地同步运行,但是没有本文描述的 TM 和 CRM 单元。在这种情况下,松散同步意味着,通道 1 可以在从通道 2 前面或后面的少于一个指令到通道 2 前面或后面的任何数量指令的任何地方。对于图 2 所示的示例,通道 1 在通道 2 的“前面”。

[0034] 在步骤 1 中,当发生计时器中断时,通道 1 中的进程 1(低优先级背景进程)刚好完成端口 FOO 上的输出事务。通道 2 中的进程 1 尚未完成相同的输出事务。

[0035] 在步骤 2 中,背景进程 (进程 1) 不再运行,因为它处于低优先级。而高优先级进程 (进程 2) 在两个通道中运行,并接收使进程 1 重新开始的输入数据。因此,通道 2 中的进程 1 未曾发送其输出。

[0036] 在步骤 3 中,最终 (在某个有界时限内),数据输出管理单元报告由于通道 2 未曾曾在端口 FOO 上发送输出的事实而引起的故障。如果模块提供了这两个计算通道之间的适当同步,则这是本来可以预防的故障类型 (因此增大可用性)。

[0037] 第一实施例中所用的体系结构方法是,模块的硬件和软件组件一起工作,以确保在执行 I/O 处理之前 (以及同时) 使每个处理通道的软件状态同步。应注意,“软件”是指托管应用软件和模块的软件组件。还应注意,术语“同步”表示,每个通道都已完成相同的一组临界区域,并且都在采集相同输入的相同临界区域内,或者都在发送相同输出的相同临界区域内。来自这 N 个通道中的每个通道的 I/O 输出要进行比较,并且必须在被输出之前通过这个比较。

[0038] 该体系结构方法的最高级属性如下。该体系结构鲁棒地支持:代表支持虚拟化 (例如,如 ARINC 规范 653 所规定) 的模块的时间和 / 或空间分区环境;以及其中模块只支持单个托管应用的环境。该体系结构支持模块的这 N 个处理通道上的同样或不一样的处理器 (2 个或更多个)。该体系结构松散地同步,由此使计算状态同步。该体系结构以可能的最大程度从托管应用提取冗余管理 (同步和比较)。这使得托管应用提供者能够对他们的软件使用常规设计标准 (它们不需要加入特殊高集成度特征),并将使得它们能够在典型的正常集成度模块上运行相同的托管应用软件。该体系结构是参数化的,以使得可静态地配置提供高集成度和可用性的单元。这使得一些托管应用 (或到 / 来自那些托管应用的数

据)能够配置为正常集成度。该体系结构确保及时检测故障以缓解由于错误输出而引起的功能危害。

[0039] 为了实现这种方法,根据第一实施例的系统和方法提供包含如下项的机构(或元件):数据输入管理(IM)、时间管理(TM)、临界区域管理(CRM)和数据输出管理(OM)。图3示出这些元件如何与模块和托管应用软件相关的逻辑框图。下面将详细描述这些元件中的每个元件。

[0040] 在第一实施例的一个可能实现中,IM、TM、CRM和OM机构构建到经由高速总线(例如,PCI-Express或专用总线)连接到托管应用处理器元件的I/O元件中。利用两个I/O元件(它们之间具有通信信道)以便支持高集成度要求。此外,托管应用元件上的软件在规定的同步点与这些机构交互。

[0041] 图4示出根据第一实施例如何在双通道高集成度模块中实现这种功能性的框图。本领域的技术人员将认识到,第一实施例具有包括以下实现在内的许多其它可能的实现。模块由两个处理通道组成,每个处理通道包括高度集成的双(或多)核微处理器和相关联的时钟、存储器装置、I/O装置等,其中托管应用元件310的功能性经由模块硬件和软件组件利用一个或多个微处理器核(以及相关联的时钟、存储器、I/O装置等)实现,并且I/O元件320的功能性经由模块硬件和软件组件利用每个通道上的一个或多个嵌入式微处理器核(以及相关联的存储器、I/O装置等)实现。模块由两个处理通道组成,每个处理通道包括单核微处理器和相关联的时钟、存储器装置、I/O装置等,其中每个通道的托管应用元件310和I/O元件320的所有功能性都经由每个通道上的微处理器核以及相关联的存储器、I/O装置等所提供的模块硬件和软件组件实现。

[0042] 如图4中所提供的示例所示,根据第一实施例的高集成度松散同步模块300包括两个通道,即通道1和通道2,由此可在N-通道模块中利用第一实施例,其中N是大于或等于2的正整数。模块300还包括托管应用元件310,它对于每个通道具有处理器CPU 350A、350B(在图4所示的示例中,存在两个处理器CPU,一个350A用于通道1,而一个350B用于通道2)。每个处理器CPU 350A、350B访问非易失性存储器(NVM)330A、330B和同步动态随机存取存储器(SDRAM)340A、340B,由此为每个处理器CPU提供时钟电路。图4示出向每个处理器CPU 350A、350B提供时钟信号的一个时钟电路360,由此还提供时钟监控器365以确保在任何时候向每个通道的处理器CPU 350A、350B提供稳定的时钟信号。本领域的技术人员将认识到,可以用在每个通道上运行的独立时钟来代替托管应用元件310上的时钟360和时钟监控器365,并且可以用在每个通道上运行的独立时钟来代替I/O元件320上的时钟384和时钟监控器382,这仍在本文描述的实施例的精神和范围内。

[0043] 托管应用元件310在每个相应通道中通过PCI-E总线以通信方式连接到I/O元件320。此外,托管应用元件310的每个通道通过PCI-E总线连接到托管应用元件310的另一通道。本领域的技术人员将认识到,可利用其它类型的总线、交换网络或存储器装置来提供托管应用元件310内和托管应用元件310与I/O元件320之间的这种通信连接,这仍在本文描述的实施例的精神和范围内。

[0044] I/O元件320包括通道1I/O处理器370A和通道2I/O处理器370B,由此这些I/O处理器370A、370B通过PCI-E总线彼此通信地连接。本领域的技术人员将认识到,可利用其它类型的总线、交换网络或存储器装置来提供每个通道的I/O处理器370A、370B之间的

这种通信连接,这仍在本文描述的实施例的精神和范围内。

[0045] 每个 I/O 处理器 370A、370B 包括数据输入管理元件 (IM)、时间管理元件 (TM)、临界区域管理元件 (CRM) 和数据输出管理元件 (OM)。每个 I/O 处理器 370A、370B 还包括其它 I/O 元件 375A、375B 和 ARINC 664Part7 元件 380A、380B,其中这些元件对于飞行器计算机处理领域的技术人员是公知的,并且为了简洁起见,将不作进一步描述。本领域的技术人员将认识到,可利用其它类型的 I/O 数据总线 (而不是 ARINC 664 Part 7) 来为模块提供这种通信连接,这仍在本文描述的实施例的精神和范围内。

[0046] 图 4 中还示出时钟单元 384 和时钟监控器 382,它们用于向多通道模块的每个通道中的每个 I/O 处理器 370A、370B 提供稳定的时钟信号。本领域的技术人员将认识到,可以用在每个通道上运行的独立时钟来代替 I/O 元件 320 上的时钟 384 和时钟监控器 382,这仍在本文描述的实施例的精神和范围内。

[0047] 图 4 还示出每个通道的 I/O PHY 单元 386A、386B、每个通道的 XFMR 单元 388A、388B 以及用于提供功率信号并对多通道模块的每个通道中的组件执行监控的电源和监控器单元 390。接口单元 395 向模块 300 的各种组件提供电力信号连接 (例如,12V DC、PWR ENBL)。作为示例,可以从飞行器的引擎 (当飞行器引擎打开时) 或从电池或发电机 (当飞行器引擎关闭时) 向接口单元 395 (并因此向高集成度模块 300 的各种组件) 提供电力。本领域的技术人员将认识到,电源和监控器 390 可实现为独立的 (每通道一个) 或模块的单个电源和监控器,这仍在本文描述的实施例的精神和范围内。

[0048] 下面提供对 IM、TM、CRM 和 OM 机构的概述。

[0049] IM 确保运行所有计算通道的软件接收完全相同的一组高集成度输入数据。如果不能为每个通道提供相同的一组数据,则 IM 将丢弃该数据,阻止任一通道接收数据,并报告错误情况。

[0050] 可能会有大量数据流被视为是正常集成度。也就是说,可能会有大量数据流入到不需要双通道 I/O 接口 (以及执行交叉通道数据确认的相关联开销) 的模块中、或者从该模块的托管应用中流出。第一实施例使得能够将正常集成度数据流从一个正常集成度源提供给两个计算通道。这种优化可经由将每个数据流 (例如,送往托管应用或从托管应用发送的每个 ARINC664 Part 7 虚拟链路) 指定为正常或高集成度的配置参数来实现。

[0051] 在商用飞机上使用的第一实施例的一个可能实现中,需要在多个通道上提供输入数据等效值的服务的示例是 :ARINC653 Part 1 I/O API 调用 (例如,采样和排队端口);ARINC653 Part 2 I/O API 调用 (例如,文件系统和服务接入点);OS I/O API 调用 (例如,POSIX 进程间通信);以及其它 (例如,平台特有的)API 调用。

[0052] TM 确保所有计算通道接收相同请求的等效时间值,即便请求在时间上有偏差 (由于计算通道之间的松散同步引起)。在这点上,时间是到托管应用的特殊类型的输入数据,因为其值由模块产生 / 控制,而不是由另一个托管应用或由模块外部的 LRU 产生。图 5 示出根据第一实施例的 TM 400 的框图以及它传送到多通道模块的通道和从这些通道接收的信号。

[0053] 实质上, TM 确保每个计算通道获得与另一通道所做的请求对应的相同准确时间。1- 深度缓冲器 (例如,只存储一个时间条目的缓冲器) 保存在两个通道发出时间请求时将传递到这两个通道的时间值。如果一个计算通道在很长一段时间 (最可能是由于另一通道

中的错误引起)“等待”另一通道发出时间请求,则使用那个通道的看门狗计时器机构(未示出)来检测并响应这种错误情况。

[0054] 根据第一实施例的 TM 可在模块中经由硬件 / 软件逻辑来实现(例如,在 I/O 元件上的 FPGA 中结合用于控制对 FPGA 的访问的模块软件来实现)。为了提供有效的同步时间, TM 可在“用户”模式进行访问(使得不需要系统调用)。

[0055] 在商用飞机上使用的第一实施例的一个可能实现中,当托管应用进行如下 API 调用时调用 TM :可应用 ARINC653 Part 1 和 Part 2 API 调用(例如 Get_Time);可应用 POSIX API 调用(例如计时器 API);以及其它(例如,平台特有的)API 调用。

[0056] 当平台软件对系统时间有需要时,调用 TM。图 5 中所示的 TM 包括时间缓冲器。TM 接收来自每个通道的请求时间信号,并向每个通道输出时间数据。通过时间硬件单元向 TM 提供当前时间。

[0057] 在第一实施例的备选实现中,时间缓冲器可实现为 N- 深度缓冲器(例如,能够存储 N 个时间值的缓冲器),而不是 1- 深度缓冲器。如果确定在计算通道之间存在大量偏差 / 漂移的可能,并且如果期望将同步点(对应于一个通道必须等待另一通道跟上的点)的数量减至最少,则这可提供性能优化。

[0058] 图 6 示出根据第一实施例的 CRM 500 的框图以及它传送到多通道模块的通道和从这些通道接收的信号。CRM 使得能够在计算通道中识别多个通道内的临界区域并使它们同步。这些临界区域实质上是在相同处理上下文内不能由任何其它执行线程抢先的软件内的区域。由托管应用和模块软件生成的某些时期将与 CRM 交互,以便在所有计算通道中适当同步。CRM 确保所有通道以同步方式进入和退出模块 CR 状态。

[0059] 在图 6 的框图中可见,CRM 逻辑对于 2- 通道模块需要三组输入事件:进入或退出临界区域的通道 1 请求,进入或退出临界区域的通道 2 请求,以及模块中断。每个通道可通过在该通道上运行的软件或通过该通道上的硬件(例如,硬件中断)生成进入临界区域的请求。每个通道可通过在该通道上运行的软件或通过该通道上的硬件生成退出临界区域的请求。对于 2- 通道模块,CRM 具有单个输出事件,即串行化临界事件。串行化临界事件包括计时器中断和临界区域状态改变事件的串行化。所有计算通道都将基于串行化临界事件执行相同的状态转变。对于 N- 通道处理模块,其中 N 是大于或等于 2 的整数,CRM 支持进入或退出临界区域的 N 个输入请求、模块中断和输出到所有 N 个通道的 1 个串行化临界事件。对于本领域技术人员显而易见的是,CRM 可基于模块的实现来对附加临界事件进行串行化。对于本领域技术人员还显而易见的是,CRM 可扩展为支持多级临界区域,以便支持像多级操作系统这样的东西(例如,用户模式、监管模式)。

[0060] CRM 可实现为硬件逻辑(例如,现场可编程门阵列)和 / 或软件逻辑的组合。

[0061] 一般而言,在如下情况(经由进入 / 退出 CR 的请求和模块中断)调用根据第一实施例的 CRM :无论何时操控可作为到不同于当前运行的线程(或进程)的执行线程的输入的数据(CRM 确保在所有计算通道上的原子性);无论何时对软件输入或从软件输出数据(包括时间);无论何时软件试图改变其执行线程;当执行线程修改需要通过模块重启维持的数据;无论何时发生生成模块中断的事件。

[0062] 图 7 示出 CRM 如何结合 I/O 处理器的其它机构来缓解图 1 所示的情形的示例。

[0063] 在图 7 的系统中,通道 1 和 2 正在松散地同步运行,其中包括增加了本文描述的 OM

和 CRM 单元。在这种情况下,松散同步意味着,通道 1 可以在从通道 2 前面或后面的少于一个指令到通道 2 前面或后面的任何数量指令的任何地方。对于图 7 中所示的示例,通道 1 在通道 2 的“前面”。

[0064] 在步骤 1 中,在将全局布尔值设成“真”之前,通道 1 中的进程 1 调用 ARINC 653 锁定 - 抢先 (Lock-Preemption) API。对锁定 - 抢先的调用生成进入临界区域 (CR) 的请求。然而,不允许通道 1 继续进行到“锁定 - 抢先”状态,直到通道 2 也调用了生成进入临界区域 (CR) 的请求的 ARINC653 锁定 - 抢先 API,之后 CRM 向两个通道发送串行化临界事件。

[0065] 在步骤 2 中,当发生计时器中断 (图 6 中所示的模块中断) 时,生成进入 CR 的请求。CRM 不能允许计时器中断引起任一通道中的上下文切换,因为它不能生成另一个串行化临界事件,直到每个通道都已经生成退出 CR 的请求。

[0066] 在步骤 3 中,在将来的某一时间点,通道 1 解除对抢先的锁定,并且通道 2 对抢先进行锁定和解除锁定 (这生成退出 CR 的请求)。在这个时间点,两个通道都成功更新了全局数据,并且优先级抢先 (这启动两个通道中的进程 2) 现在可经由 CRM 传递下一串行化临界事件而发生。

[0067] 在步骤 4 中,两个通道中的进程 2 读取布尔值并发送输出 (真)。数据输出管理 (OM) 单元验证两个通道的输出相等。在图 7 中可见,CRM 缓解了图 1 中所示的情形。

[0068] 图 8 示出 CRM 如何结合 OM 来缓解图 2 所示的情形的示例。

[0069] 在图 8 的系统中,具有两个进程 (进程 1 和进程 2) 的相同软件正以松散同步的方式在通道 1 和通道 2 上运行。在这种情况下,松散同步意味着,通道 1 可以在从通道 2 前面或后面的少于一个指令到通道 2 前面或后面的任何数量指令的任何地方。对于图 8 中所示的示例,通道 1 在通道 2 的“前面”。

[0070] 在步骤 1 中,通道 1 中的进程 1 (低优先级背景进程) 向 CRM 发送进入临界区域的请求,以使得它可在端口 FOO 上开始输出事务,并且 CRM 允许通道 1 开始其输出事务。通道 2 中的进程 1 也已经向 CRM 发送进入临界区域的请求,并且也已经在端口 FOO 上开始输出事务,但是它在通道 1 的“后面”。通道 1 上的处理处于 FOO 已经从该通道输出但是 FOO 尚未从通道 2 输出的那个点。由于在模块中引入了 CRM,所以 CRM 将不允许通道 1 退出临界区域,直到通道 2 中的进程 1 也已经完成了相同的输出事务,并请求退出临界区域。

[0071] 在步骤 2 中,发生计时器中断,同时通道 1 正等待退出临界区域,并且通道 2 仍在临界区域中执行其输出事务。

[0072] 在步骤 3 中,一旦两个通道都已经完成了它们的 I/O 事务,并且已经发送了退出临界区域的请求,就可传递串行化中断,并且两个通道中的进程 2 都开始运行。在这点之后,进程 2 可安全地重启进程 1 (在两个通道上)。在图 8 中可见,增加 CRM 缓解了在图 2 所示的情形中发生的故障情况。

[0073] OM 确认在所有计算通道上从软件输出的高集成度数据流。如果在输出数据流中检测到错误,则 OM 将阻止数据输出,并将提供错误指示。

[0074] 应注意,可能会有大量数据被视为是正常集成度。也就是说,可能会有大量数据 (以及整个软件应用) 不需要双通道 I/O 元件 (以及执行交叉通道比较的相关联开销)。根据第一实施例的系统和方法使得能够从其中一个计算通道输出正常集成度的数据 (并忽略来自另一个计算通道的输出)。在第一实施例的一个可能实现中,配置参数将特定数据或

整个托管应用指定为正常或高集成度。

[0075] 根据第一实施例的方法和系统支持在源处对于高集成度和可用性的要求。此外，因为对于在平台上运行的软件的状态提取了同步点，所以第一实施例可扩展为支持不一样的处理器。

[0076] 第一实施例的性能受到可在 I/O 平面上合理同步和验证的数据量的限制。如果这是一个问题，则可通过利用正常集成度与高集成度数据和软件应用之间的区别（在系统中）来优化性能。

[0077] CRM、TM、IM 和 OM 单元的设计和实现不依赖于定制硬件能力（定制 FPGA、ASIC）、或当前的和 / 或可能过时的微处理器能力的属性。因此，根据第一实施例构建的模块将呈现如下示范性有利属性：它们能够利用包括嵌入式存储器控制器、具有不同时钟恢复电路的多个锁相环 (PLL) 等的现有技术的微处理器（这将允许模块性能容易地提高（经由微处理器升级），而无需对提供 CRM、TM、IM 和 OM 的模块组件进行大量重新设计）；同步时期的频率（即，开销）应远小于指令级锁步体系结构。因此，同步机构对于需要访问它们的软件来说都是可直接访问的（而不需要附加的系统调用）。因此，由于同步引起的附加开销在每个时期都应在几个指令的数量级。

[0078] 还提供根据第一实施例的系统和方法的其它优点。性能改进应该直接与硬件性能改进成比例。也就是说，不需要可能对处理器与存储器子系统之间的接口施加许多约束的特殊硬件。整个托管应用 (DO-178B 级 B、C、D、E) 能够被识别为正常集成度。当这样做时，将对与这个托管应用相关联的所有数据和控制禁用 IM、TM、CRM 和 OM 元件，所有事务都将只在一个计算通道上进行，并且另一个计算通道可在这个时间期间处于空闲状态。这不仅有利于性能，而且如果不活动计算通道中的处理器可在正常集成度时间窗口期间进入“休眠”模式，则它还可导致功耗（发热）下降。

[0079] 这个第一实施例使得系统集成器能够通过利用不活动计算通道中的空闲时间来运行不同的托管应用而利用正常集成度托管应用的概念。这可导致具有大量正常集成度的托管应用的系统的性能改进。

[0080] 根据第一实施例的系统和方法适合于能够运行双独立计算通道，因此有效地在正常集成度模式使模块性能加倍。

[0081] 根据第一实施例的系统和方法支持模块的不同计算通道上的不一样的处理器。在这种情况下，有可能的是（例如），不一样处理器的浮点单元可提供不同的舍入 / 截断行为，由此导致从不一样的计算通道输出稍微不同的数据。因此，近似数据比较（与准确数据相比）可用于某些类别的输出数据流，以便支持不一样的处理器。

[0082] 具有采用 IM、TM、CRM 和 OM 的机构的软件应用交互可构建到任何操作系统 API 中（即，将不需要“特殊”API）。因此，根据第一实施例的系统和方法被认为对软件应用开发者只施加了最小约束。

[0083] 预期，对系统集成器（和 / 或工具）的仅有影响将是，I/O 配置数据将具有用于将数据流和托管应用识别为高集成度或正常集成度的（可选）属性。

[0084] 书面描述利用示例来公开包括最佳模式的本发明，并且还使得本领域的技术人员能够获得和使用本发明。本发明的可授予专利的范围由权利要求限定，并且可包括本领域技术人员可想到的其它示例。如果这些其它实例具有与权利要求的字面语言没什么不同

的结构元素,或者如果这些其它实例包括与权利要求的字面语言无实质差别的等效结构元素,则它们要在权利要求的范围内。

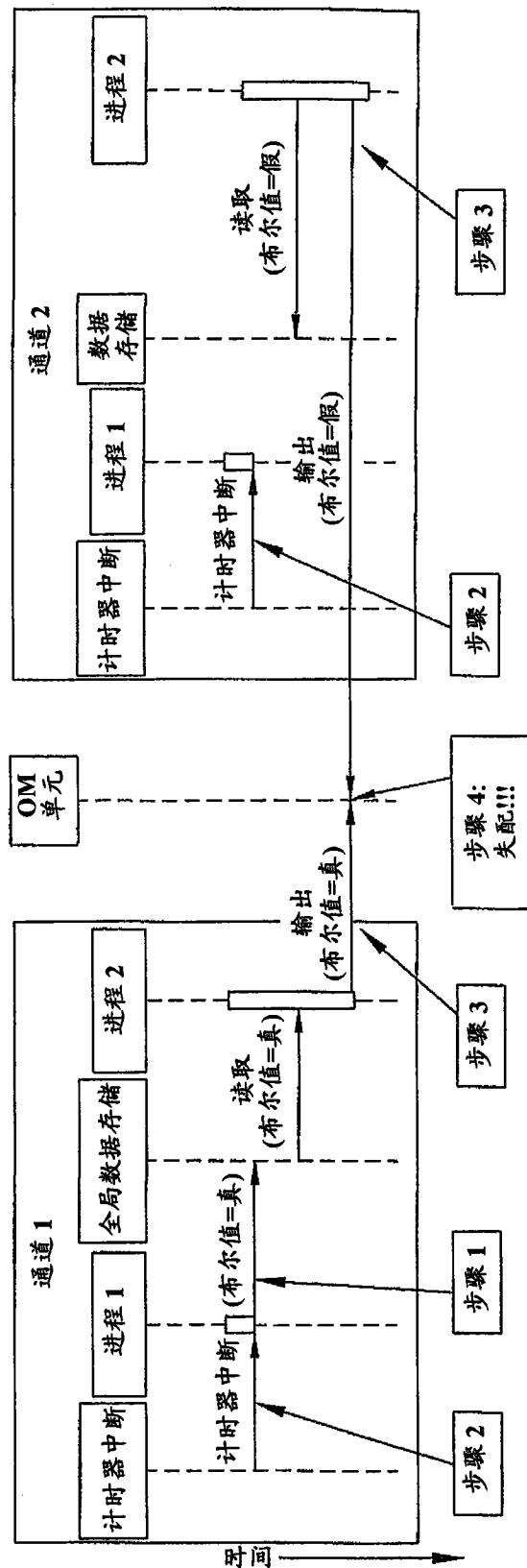


图 1

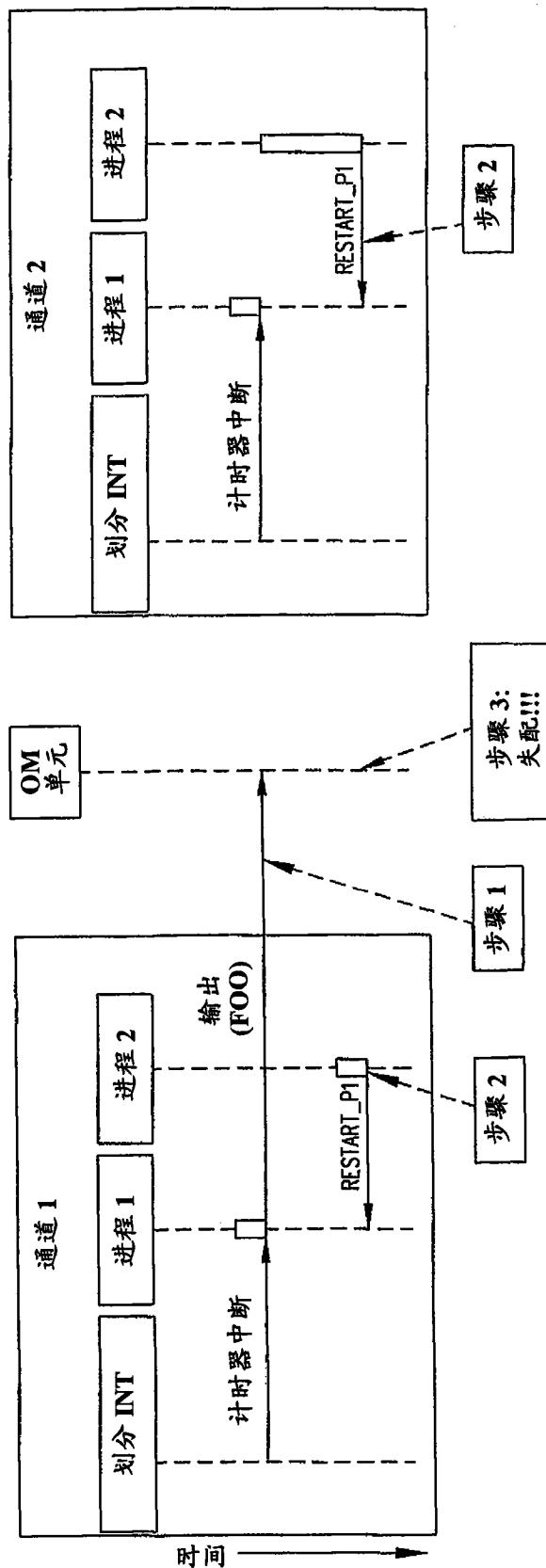


图 2

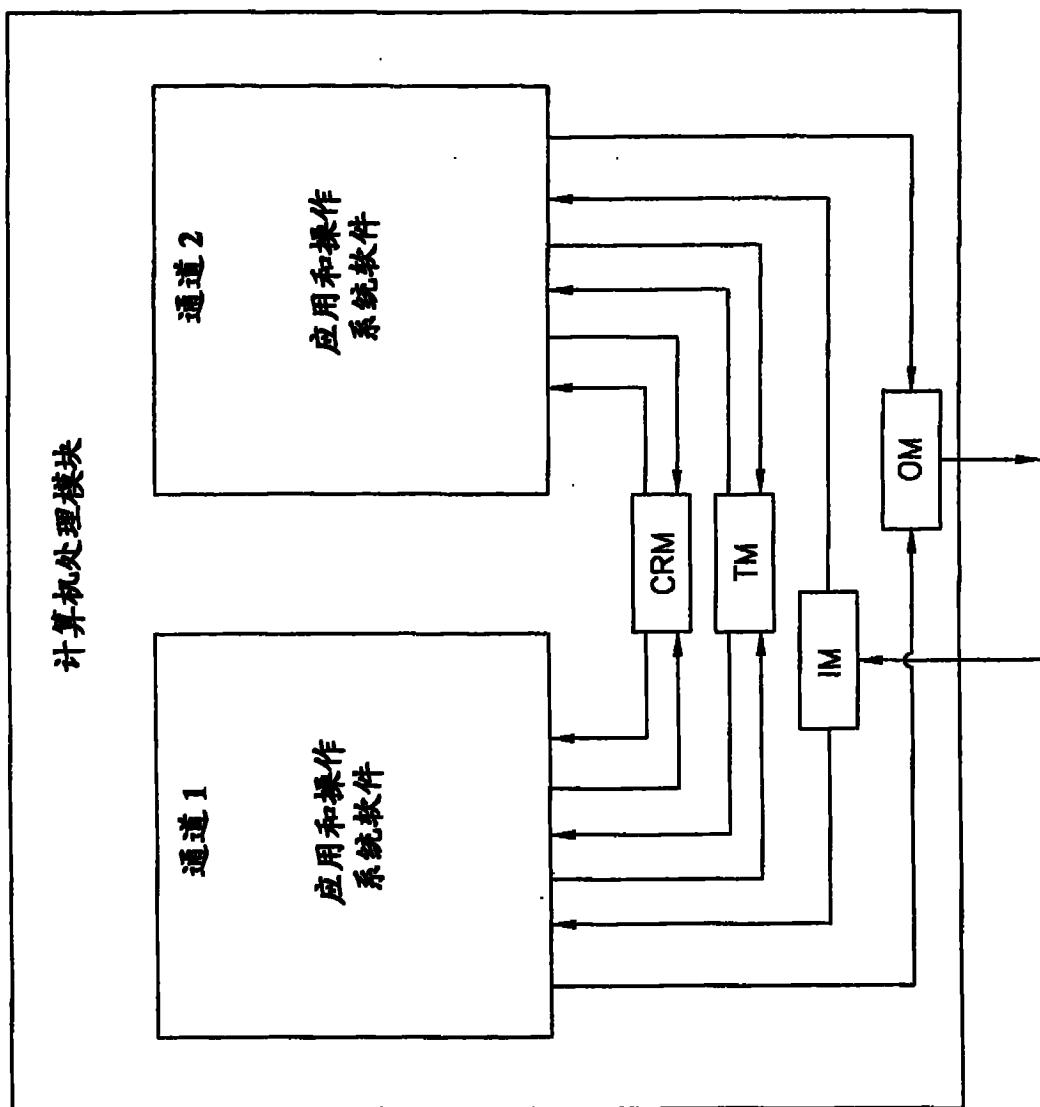


图 3

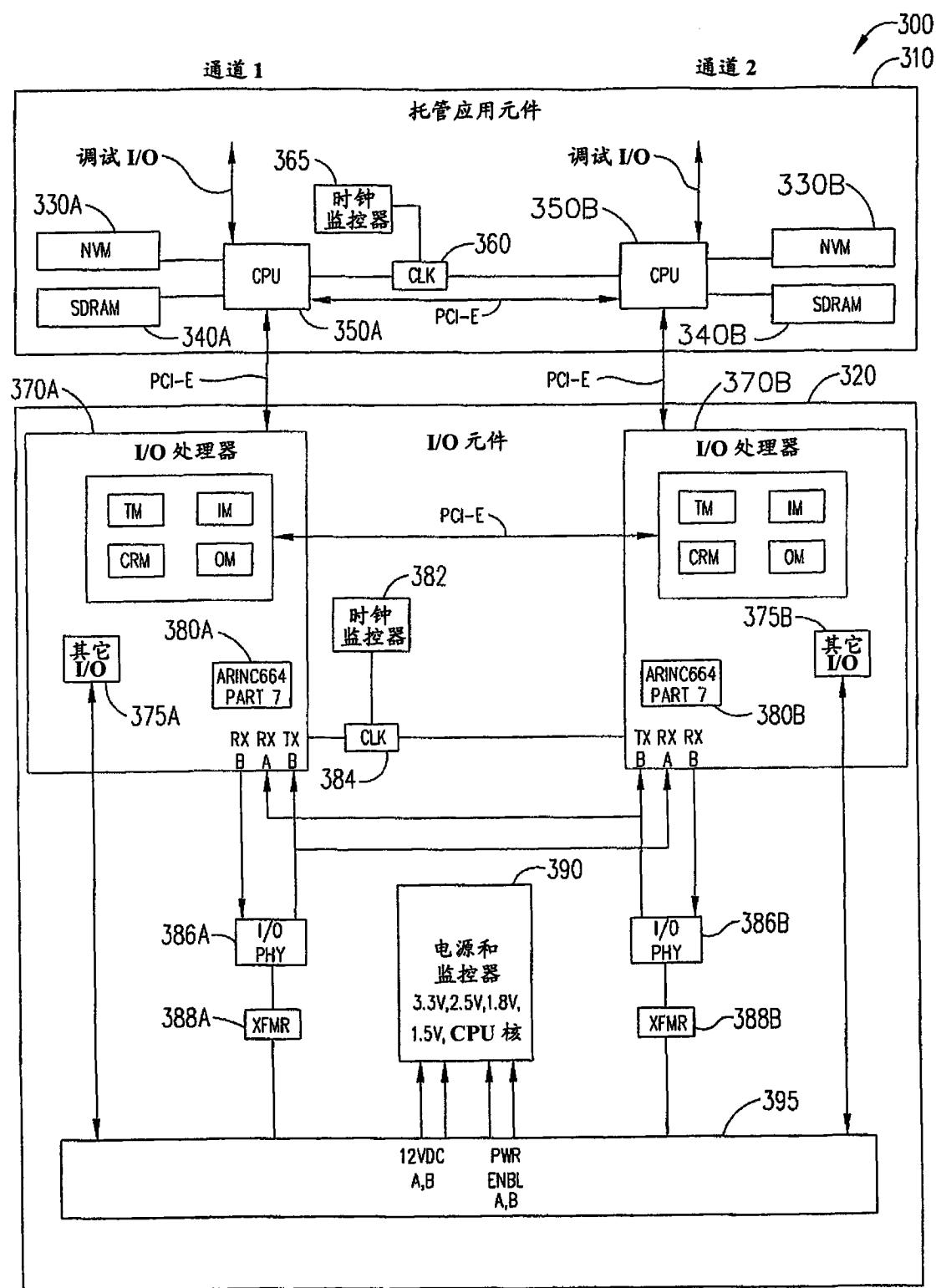


图 4

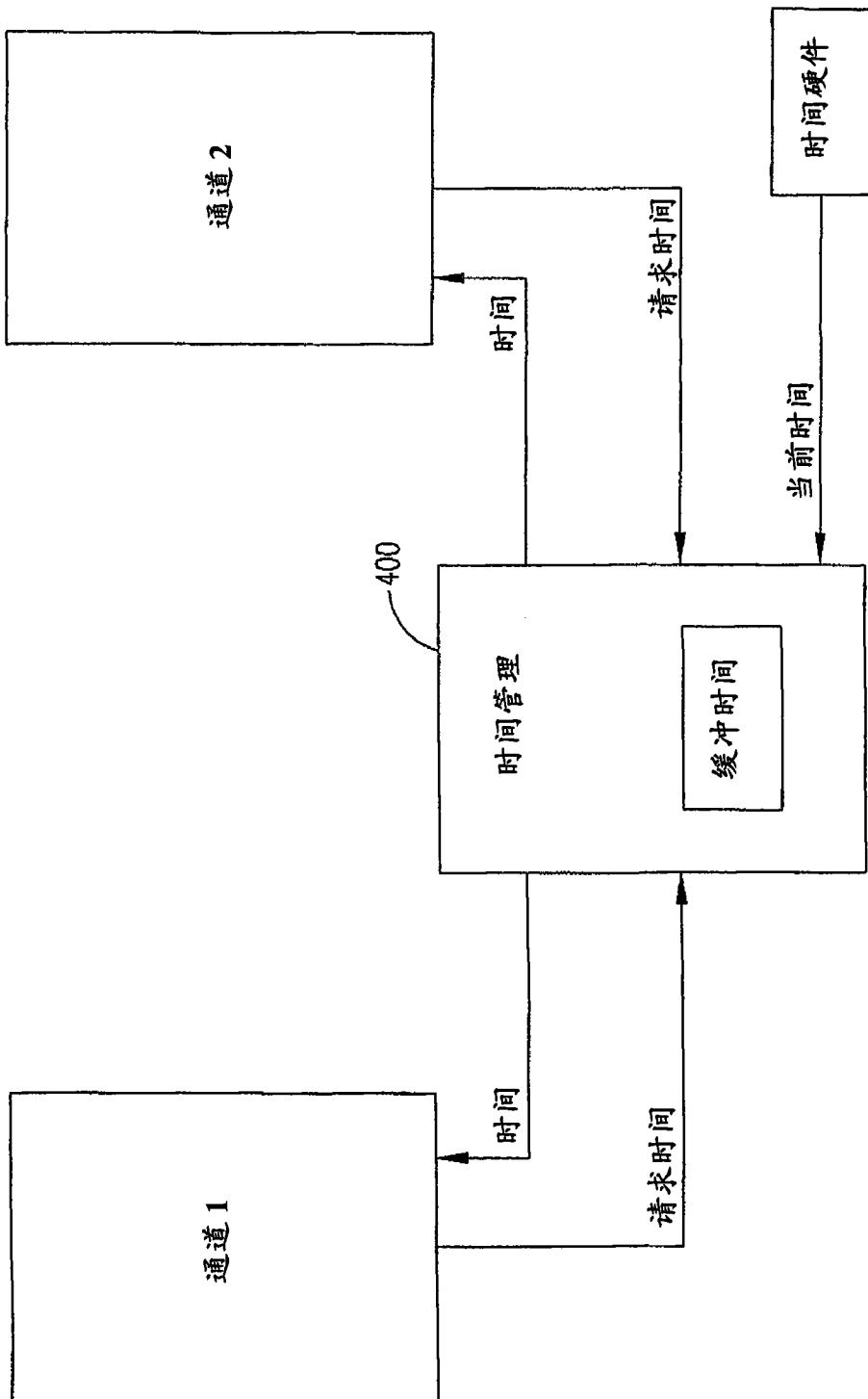


图 5

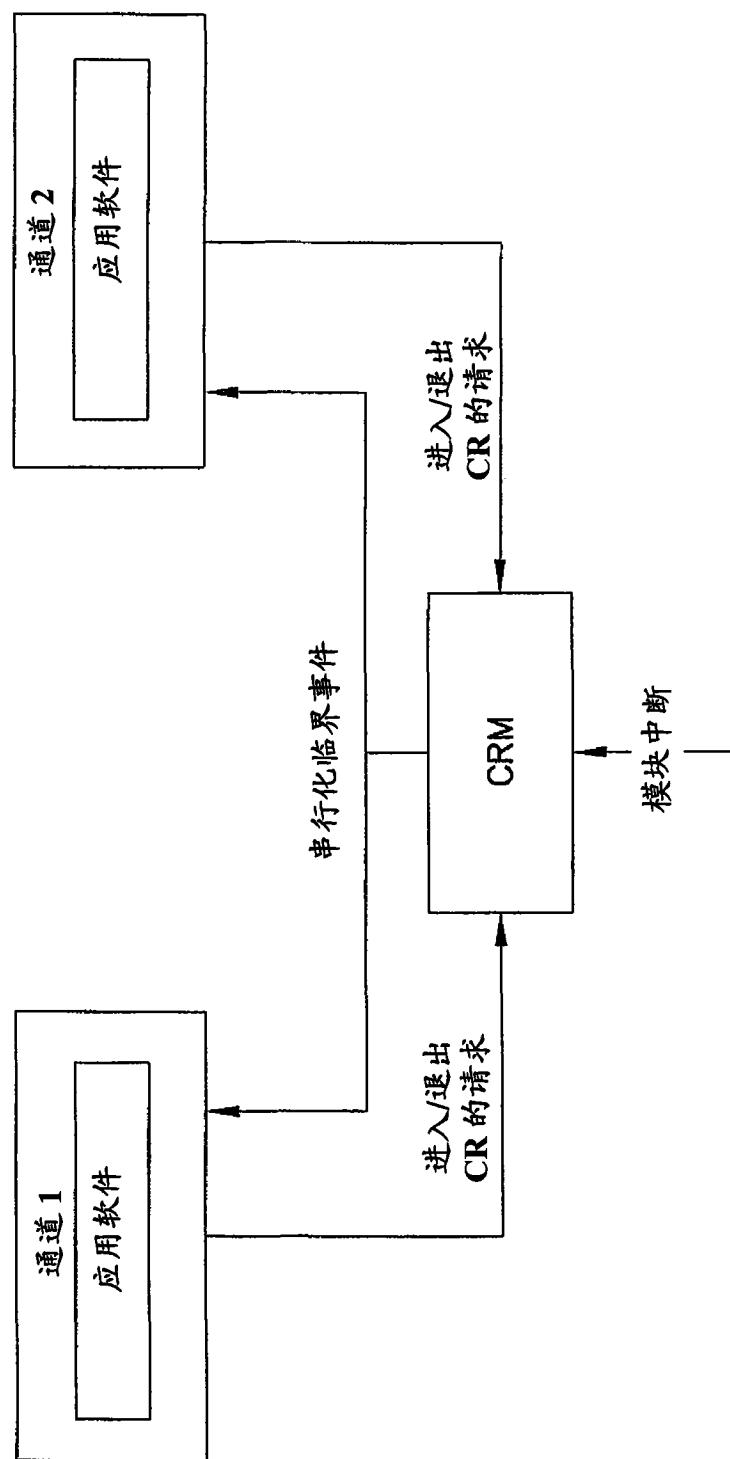


图 6

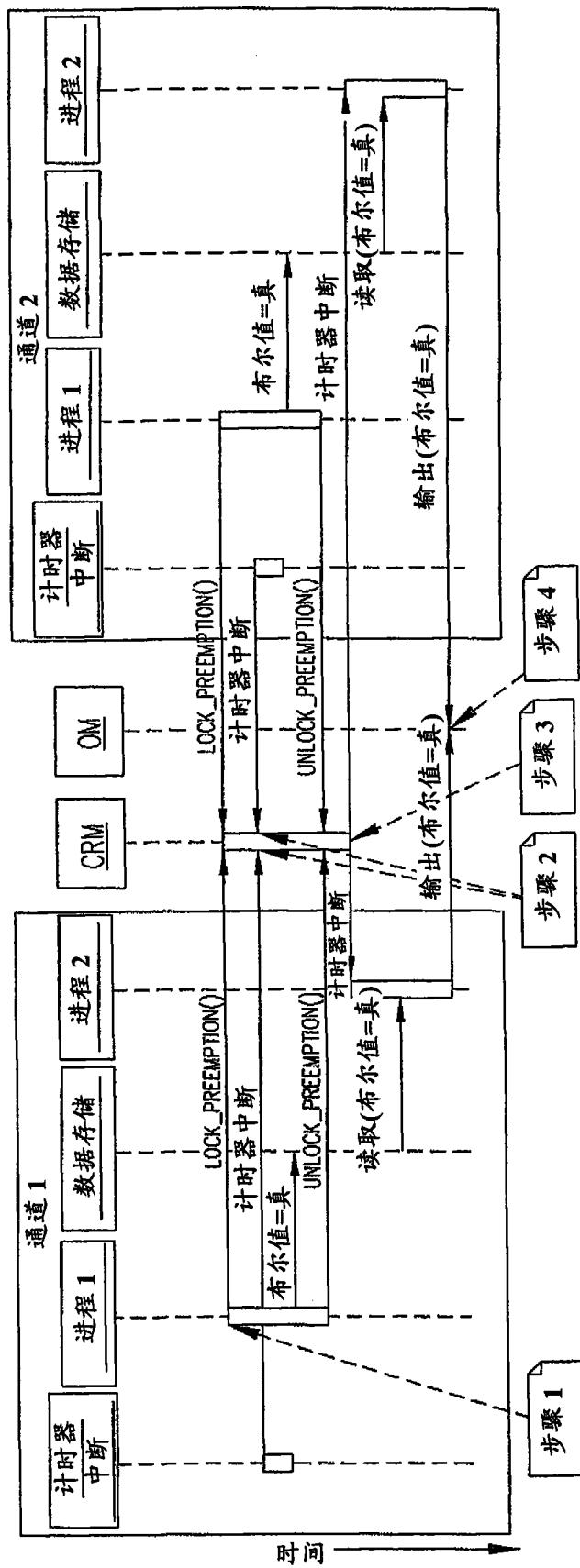


图 7

