(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 103098032 A
(43)申请公布日 2013.05.08

(21)申请号 201180038696.1
(22)申请日 2011.07.27
(30)优先权数据
2010-184845 2010.08.20 JP
(85)PCT申请进入国家阶段日
2013.02.05
(86)PCT申请的申请数据
PCT/JP2011/067132 2011.07.27
(87)PCT申请的公布数据
WO2012/023397 JA 2012.02.23
(71)申请人 国际商业机器公司
地址 美国纽约阿芒克
(72)发明人 松村郁生 清水周一
(74)专利代理机构 北京市金杜律师事务所 11256
代理人 酆迅
(51)Int.Cl.
G06F 11/28(2006.01)

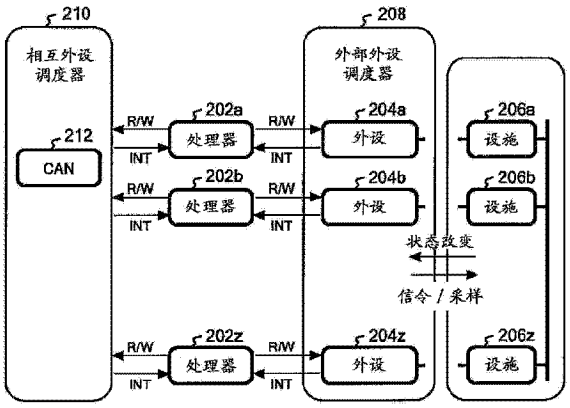
权利要求书3页 说明书19页 附图21页

(54)发明名称

仿真方法、系统和程序

(57)摘要

提供一种包括4层的仿真系统，这4层是作为ECU模拟器的处理器模拟器、设施仿真器、外部外设调度器和相互外设调度器。外部外设调度器使得设施仿真器仅被提前执行处理器模拟器的响应延迟时间（备选地为直到下一事件之前的时间）并且发送将提前执行处理器模拟器直至在设施仿真器已经实际停止时的时间这样的通知。相互外设调度器通知处理器模拟器仅提前执行处理器模拟器之间的响应延迟时间（备选地为直到下一事件之前的时间）这样的通知。处理器模拟器保守地使处理前进到直至在已经通知的时间之中的最早时间。



CN 103098032 A

1. 一种执行仿真的计算机实现的仿真系统，包括：

多个处理器模拟器，其在所述计算机上运行；

多个设施仿真器，其在所述计算机上运行；

外部外设调度器，其在所述计算机上运行并且首先运行所述设施仿真器直至所述处理器模拟器的反应延迟时间，并且通过所述外设模拟器提供用于在所述处理器模拟器之前运行直至所述仿真器停止时间的通知；以及

相互外设调度器，其在所述计算机上运行并且提供用于仅提前运行所述处理器模拟器与所述处理器模拟器的通信延迟时间的通知，其中所述处理器模拟器执行保守处理直至所述通知时间。

2. 根据权利要求 1 所述的仿真系统，其中所述计算机是多任务系统，并且将所述多个处理器模拟器、所述多个设施仿真器、所述外部外设调度器以及所述相互外设调度器作为单独的线程或者进程被执行。

3. 根据权利要求 2 所述的仿真系统，其中所述计算机是多核或者多处理器系统，并且将所述多个处理器模拟器、所述多个设施仿真器、所述外部外设调度器以及所述相互外设调度器执行作为向与所述多处理器系统不同的单独处理器或者核所分配的单独线程或者进程。

4. 根据权利要求 3 所述的仿真系统，其中向相同的核或者处理器分配所述设施仿真器和与所述设施仿真器通信的所述外设模拟器。

5. 根据权利要求 1 所述的仿真系统，其中所述多个设施仿真器具有自发确定的状态的主动设施仿真器和在所述处理器模拟器侧上确定定时的被动设施仿真器，并且所述外部外设调度器具有与所述主动设施仿真器通信的用于主动设施的外部外设调度器以及与所述被动设施仿真器通信的用于被动设施的外部外设调度器。

6. 根据权利要求 1 所述的仿真系统，还包括 CAN 模拟器，其中所述 CAN 模拟器由所述相互外设调度器调度。

7. 根据权利要求 1 所述的仿真系统，其中所述处理器模拟器中的每个处理器模拟器与具有队列的内部外设调度器结合操作，并且基于用于所述队列中的写入事件的时间来避免由所述多个处理器模拟器的共享式存储器访问死锁。

8. 一种执行仿真的计算机实现的仿真方法，包括步骤：

在所述计算机上执行多个处理器模拟器；

在所述计算机上执行多个设施仿真器；

在所述计算机上执行外部外设，并且首先运行所述设施仿真器直至所述处理器模拟器的反应延迟时间，并且通过所述外设模拟器提供用于在所述处理器模拟器之前运行直至所述仿真器停止时间的通知；以及

执行相互外设调度器，所述相互外设调度器在所述计算机上运行并且提供用于仅提前运行所述处理器模拟器与所述处理器模拟器的通信延迟时间的通知，其中：

所述处理器模拟器保守地进行直至所述通知时间。

9. 根据权利要求 8 所述的仿真方法，其中所述计算机是多任务系统，并且作为单独的线程或者进程来执行所述多个处理器模拟器、所述多个设施仿真器、所述外部外设调度器和所述相互外设调度器作为单独的线程或者进程。

10. 根据权利要求 9 所述的仿真方法，其中所述计算机是多核或者多处理器系统，并且根据所述多个多处理器系统向所述单独的处理器或者核分配所述多个处理器模拟器、所述多个设施仿真器、所述外部外设调度器以及相互外设调度器作为单独的线程或者进程。

11. 根据权利要求 10 所述的仿真方法，其中向相同的核或者处理器分配所述设施仿真器和与所述设施仿真器通信的所述外设模拟器。

12. 根据权利要求 8 所述的仿真方法，其中所述多个设施仿真器具有自发确定的状态的主动设施仿真器和在所述处理器模拟器侧上确定定时的被动设施仿真器，并且所述外部外设调度器具有与所述主动设施仿真器通信的用于主动设施的外部外设调度器以及与所述被动设施仿真器通信的用于被动设施的外部外设调度器。

13. 根据权利要求 8 所述的仿真方法，还包括执行 CAN 模拟器的步骤，其中所述 CAN 模拟器由所述相互外设调度器调度。

14. 根据权利要求 8 所述的仿真方法，其中所述处理器模拟器中的每个处理器模拟器与具有队列的内部外设调度器结合操作，并且基于用于所述队列中的写入事件的时间来避免由所述多个处理器模拟器的共享式存储器访问死锁。

15. 一种执行仿真的计算机执行的仿真程序，所述程序使得所述计算机执行以下步骤：

执行多个处理器模拟器；

执行多个设施仿真器；

执行外部外设并且首先运行所述设施仿真器直至所述处理器模拟器的反应延迟时间，并且通过所述外设模拟器提供用于在所述处理器模拟器之前运行直至所述仿真器停止时间的通知；以及

执行相互外设调度器，所述相互外设调度器在所述计算机上运行并且提供用于仅提前运行所述处理器模拟器与所述处理器模拟器的通信延迟时间的通知；

其中所述处理器模拟器执行保守处理直至所述通知时间。

16. 根据权利要求 15 所述的仿真程序，其中所述计算机是多任务系统，并且作为单独的线程或者进程来执行所述多个处理器模拟器、所述多个设施仿真器、所述外部外设调度器和所述相互外设调度器。

17. 根据权利要求 16 所述的仿真程序，其中所述计算机是多核或者多处理器系统，并且根据所述多个多处理器系统向所述单独的处理器或者核分配所述多个处理器模拟器、所述多个设施仿真器、所述外部外设调度器和所述相互外设调度器作为单独的线程或者进程。

18. 根据权利要求 17 所述的仿真程序，其中向相同核或者处理器分配所述设施仿真器和与所述设施仿真器通信的所述外设模拟器。

19. 根据权利要求 15 所述的仿真程序，其中所述多个设施仿真器具有自发确定的状态的主动设施仿真器和在所述处理器模拟器侧上确定定时的被动设施仿真器，并且所述外部外设调度器具有与所述主动设施仿真器通信的用于主动设施的外部外设调度器以及与所述被动设施仿真器通信的用于被动设施的外部外设调度器。

20. 根据权利要求 15 所述的仿真程序，还包括执行 CAN 模拟器的步骤，其中所述 CAN 模拟器由所述相互外设调度器调度。

21. 根据权利要求 15 所述的仿真程序，其中所述处理器模拟器中的每个处理器模拟器与具有队列的内部外设调度器结合操作，并且基于用于所述队列中的正确事件的时间来避免由所述多个处理器模拟器的共享式存储器访问死锁。

仿真方法、系统和程序

技术领域

[0001] 本发明涉及诸如汽车之类的物理系统的仿真并且更具体地涉及一种软件实现的仿真系统。

背景技术

[0002] 在 20 世纪早期，汽车由包括动力引擎、加速器、方向盘、发送系统和悬架的机械部件制成，但是几乎使用除了引擎火花塞和前灯之外的已知电气设备。

[0003] 然而自从 20 世纪 70 年代起由于空气污染和石油危机而一直需要控制引擎效率，因此将 ECU 用于控制引擎。一般而言，ECU 具有执行对例如来自传感器的输入信号的 A/D 转换的输入接口、根据所建立的逻辑来处理数字输入信号的逻辑操作器（微计算机）和将处理结果转换成致动器操作信号的输出接口。

[0004] 如今，现代汽车不仅包含机械部件，而且电子部件和软件占用大量份额从而不仅延伸至引擎和发送控制系统、防抱死制动系统 (ABS)、电子稳定性控制 (ESC) 和动力转向而且甚至延伸至风挡水刮器控制和监视系统等的安全性。用于这些部件的开发成本据说为总成本的从 25 至 40％并且占据混合车辆的 70％。

[0005] 附带提一点，汽车包含诸如引擎之类的动力设备、动力传送设备、用于转向等的驱动设备、制动设备和用于其它主体系统（设施 (plant)）的其它机械构建，并且这些设施的操作基于传感器输入（速度等）和来自人类的输入（加速度等）由 30 至 70 个或者更多电子控制单元 (ECU) 的程序动态地确定。

[0006] ECU 基本上控制每个单独设施的操作。例如引擎控制单元使用软件以确定用于引擎的燃料注入和激发的数量和定时。对于诸如具有“运动”模式的车辆之类的高级车辆，可以通过使用软件根据模式增加或者减少注入的燃料数量。另外，可以通过自动地向引擎给油门（使引擎空转）来使引擎的旋转速度与用于减档的定时匹配。在这一情况下，引擎 ECU 和发送 ECU 必须以协作方式操作。伴随着集成车辆位置稳定设备 (ESC：电子稳定性控制) 用于防止车辆的横向滑移等，与诸如刹车等制动设备协调是必需的，并且因此 ECU 软件很复杂。注意，这一类型的“干预”功能主要是软件并且可以容易被削减。

[0007] 为了充分实现设施的功能并且稳定地操作，必须在设计和开发 ECU 软件的过程中充分地执行对操作参数的调谐 (tune) 和测试。一般而言，在原型由实际车辆制成之后反复地执行调节和测试，但是由于成本和定时约束，存在对如下方法的强烈需求，该方法在原型化之前在计算设备中虚拟地实现控制器和设施，因此可以快速和准确地执行操作。这一 ECU 仿真包括以如下顺序使用来逼近原型的 4 个类型、即：(1) 循环中的模型仿真 (MILS)，该仿真使用诸如状态机在逻辑上表达控制器的操作之类的表达式格式；(2) 循环中的软件仿真 (SILS)，在该仿真中，向逻辑操作引入诸如数据精确性之类的一些硬件控制；(3) 循环中的处理器仿真 (PILS) 和循环中的虚拟硬件仿真 (V-HILS)，在这些仿真中，完全地封装软件以模拟 ECU 处理器；以及 (4) 循环中的硬件仿真，在该仿真中，ECU 板被完全地封装并且与实时设施仿真连接。

[0008] 主要在试验和错误阶段期间使用 MILS 和 SILS 以便实现设施的基本性能。然而操作不同于在 ECU 中实际包括的软件，因此对产品验证应用的使用是不可能的。另一方面，V-HILS 使用完成的 ECU 软件并且作为一种用于解决在软件中发现的非预期动作（缺陷）的方法是极有前途的，但是已经实现具有可再现性的操作尚无前例。总是执行 HILS 以便确认完成的 ECU 板的最终行为，但是即使发现故障仍然未保证可再现性，因此不能用于调试目的。

[0009] 不能使用 HILS 来再现操作的原因不是因为 HILS 的配置不完整而是因为所有的 ECU 配置由诸如 CAN 等之类的网络人工地连接在一起。一般而言，网络提供在模块之间的松散耦合，因此数据到达顺序将基于模块操作定时的略微差异而变化，并且因而整个系统的行为将变化。因此，即使实际车辆被原型化，仍然不能预计操作的可再现性。这是调试并行分布式系统极为困难的相同原因。

[0010] 利用这一类型的 HILS 配置，或者换而言之，利用其中在设施仿真器的 ECU 板被松散地链接的配置，即使加速每个部件仍然不能实现操作一致性。有必要按照通信顺序实现一致性以便实现操作的可再现性。预计 V-HILS 具体解决这一问题。

[0011] 根据传统概念，典型 V-HILS 配置包括多个 ECU 模拟器、多个设施仿真器和调度所有操作的全局调度器。

[0012] ECU 模拟器包括处理器模拟器和外设模拟器。另一方面，设施仿真器包括刹车仿真器和引擎仿真器等。

[0013] 这时，处理器模拟器例如在 80MHz 使用相对高分辨率的时钟来操作。另一方面，设施仿真器是物理机制的仿真器、因此例如在 10kHz 的相对低分辨率操作。一般而言，可以在高速度对更低分辨率进行仿真，因此设施仿真器经常更快。

[0014] 设施仿真器未必在具有固定长度的处理步进 (step) 时间重复值的计算，并且通常需要抑制计算差值的影响并且具有以非连续改变点的定时为基础的可变步进。在任何情况下，在每个步进中从控制器接收指令信号，并且向每个传感器输出内部状态。注意指令信号通常是用于表达开关的接通或者关断条件的脉冲。

[0015] 外设模拟器使用 ECU 模拟器的 I/O 接口来相互地连接到设施仿真器和处理器模拟器。通常（平均）可以在近似 10MHz 的分辨率抑制操作。这增加设施仿真器的速度、但是减少处理器模拟器的速度。外设模拟器向设施仿真器发送脉冲信号。另外，读取来自设施仿真器的内部状态为量化数据。

[0016] 外设模拟器接收读取写入 (R/W) 请求并且与处理器模拟器发送和接收数据并且发送中断 (INT)。具体而言，利用在处理器之间相互地连接的诸如 CAN（控制器区域网络）之类的网络的功能，从处理器 (W) 接收发送数据，并且通过总线执行在外设之间的通信，并且当接收数据时，向处理器发送中断 (INT)，并且基于请求 (R) 读取从处理器接收的数据。

[0017] 从一侧来看，外设是系统的中心并且相互地连接于设施与处理器之间以及处理器之间。如果仅有充分时间分辨率来相互地区分通过外设传递的信号的顺序，则可以恰当地再现顺序以求在设施与处理器之间的协同效应。然而如果在下一信号之前的时间决定精确程度（计算速率等），则更细微的时间分辨率将更有利。换而言之，数据误差的大小取决于时间分辨率。

[0018] 除了数据误差的问题之外也存在开销的问题。换而言之，就提供固定同步间隔的

方法而言，如果缩短同步间隔则可以实现更恰当操作，但是反言之，同步处理所需要的开销将增加，因此用于所有过程的时间将增加。

[0019] 因此，固定并且在最大限度内减少同步间隔的方式不能是用于实际地解决数据误差和开销这两个方面的方法。

[0020] 因此就 V-HILS 而言，同步问题是关键的，并且在被组织时，以下三种方法是可设想的。

[0021] (1) 时间同步 :利用这一方法，处理器模拟器在相互地通信之时执行仿真。存在在执行速度与时间精确度之间的权衡，并且存在针对时间的量化误差。另外如上文描述的那样，利用这一方法，同步通信成本在处理器模拟器之间更高，因此未实现实用分辨率。

[0022] (2) 乐观事件优化 :利用这一方法，每个处理器和模拟器通过发送或者接收具有时间信息的事件来与另一方同步。然而，推测 (speculate) 如果在保持“飞行”的可能性之时执行仿真则，因此如果该推测引起无序状态，则不能适当地回滚 (roll back) 处理器和模拟器。同步的成本比用时间同步方法更小，但是在预备回滚时相继地记录状态的成本太高并且不切实际。

[0023] (3) 保守事件同步 :利用这一方法，每个处理器和模拟器通过发送或者接收具有时间信息的事件来与另一方同步。然而仿真“保守地”进行，因此因果关系未冲突，因此将保持相互事件并且死锁可能出现。为了避免这一类型的死锁，在高频率发送特殊空值消息等以便执行时间发送，因此避免的成本高。

[0024] 作为用于 V-HILS 的常规技术，日本待审专利申请 2007-11720 公开了待解决的问题是在适应具有复杂配置的系统之时实现系统仿真器的配置的灵活变化并且系统仿真器包括 3 个类型、即 :对 CPU 操作进行仿真的指令集仿真器、对总线操作进行仿真的总线仿真器和对外设操作进行仿真的外设仿真器而在每个仿真器之间提供可以参考和改变相互状态的接口。然而这一常规技术未提示一种用于优化在外设与 CPU 之间的同步的技术。

[0025] 因此，本发明人开发一种用于保守事件同步的机制并且提供一种如在日本专利申请 2009-238954 的说明书中公开的使用外设调度器的仿真技术。利用这一技术，外设调度器通过取消 (OFF) 所有外设模拟器完成标志来开始并行操作。另外，外设调度器基于设置的每个单独外设模拟器的处理断点的定时来找到被调度处面临最早处理断点的外设模拟器。这被称为外设 P。如果该处理断点的时间为 T，则外设调度器继续执行每个处理器模拟器和每个设施仿真器直至时间达到时间 T。因此，外设调度器等待设置外设 P 的完成标志。响应于设置外设 P 的完成标志，外设调度器在外设 P、处理器模拟器和设施仿真器之间同步数据。

[0026] 然而近来已经有对如下仿真的需求，该仿真更接近地逼近实际设备的总体操作、比如同时模拟具有多核主机计算机的多个 ECU。因此，执行速度将针对多个 ECU 的主机中的每个主机而变化，因此在它们之间需要显式同步处理，但是在 ECU 之间执行有规律中断，因此同步多个 ECU 并不容易。在日本专利申请 2009-238954 的说明书中公开的技术仅适用于单个 ECU，因此不修改具有多个 ECU 的系统的仿真就不能应用这一技术。

[0027] 背景文献

[0028] 专利文献

[0029] 专利文献 1 日本未审专利申请 2007-11720

[0030] 专利文献 2 日本专利申请 2009-238954

发明内容

[0031] 本发明要解决的问题

[0032] 因此，本发明的目的是提供一种可以针对如下系统执行具有充分时间精确度和最少数据误差的快速仿真，该系统具有相互地提供异步数据通知（中断）的多个 ECU 和设施。这里，异步数据通知是指向控制器通知设施的有规律状态改变或者在处理器之间的异步 CAN 通信。

[0033] 本发明的另一目的是链接三个类型的处理器模拟器、外设模拟器和设施仿真器以便仿真整个系统，该系统具有 ECU 和受到其控制的设施。

[0034] 用于解决问题的手段

[0035] 利用本发明，将外设划分成三个类型、即相互外设、外部外设和内部外设，并且提供每个类型唯一的调度器。这三个类型的调度器配合并且高效地实现准确的链接的操作。

[0036] 相互外设是处理在处理器之间的通信的中间层并且具有唯一相互外设调度器。相互外设具有如下功能，该功能执行对在处理器之间的异步通信的高速处理。

[0037] 外部外设是设施与处理器之间通信的中间层并且具有唯一外部外设调度器。外部外设具有如下功能，该功能从设施接收异步通知并且向处理器传送而且向设施发送来自处理器的指令。

[0038] 内部外设是仅连接到具体处理器的外设层并且例如对应于看门狗定时器和存储器等。未与设施和其它处理器交换信号。内部外设具有内部外设调度器（或者处理器调度器）。

[0039] 相互外设调度器、外部外设调度器和内部外设调度器用协同例程形式并行操作。

[0040] 注意设施仿真器优选地由连续值仿真建模系统、比如 MATLAB®/Simulink® 创建。外设模拟器是事务型模拟器并且优选地由 SystemC/TLM 创建。处理器模拟器使用指令集仿真器 (ISS) 来执行模拟。

[0041] 外部外设调度器选择处理器模拟器的“反应延迟时间”或者下一事件出现的时间中的较早时间，并且执行 (perform) 设施仿真器的提前执行 (advanced execution) 直至该时间。另外，向所有处理器调度器（或者内部外设调度器）通知设施仿真器将执行提前执行直至停止时间。注意，反应延迟时间表示从接受来自设施的状态改变通知直至为对应设施计算出指令并且那些实际指令在设施中生效为止的最少时间差。

[0042] 相互外设调度器选择在处理器模拟器之间的“通信延迟时间”和下一事件出现的时间之间的较早时间并且向所有处理器调度器（或者内部外设调度器）提供通知以便执行提前执行直至该时间。注意，通信延迟时间是指从处理器发送数据到另一处理器接收数据为止所需要的最少时间。

[0043] 处理器调度器（或者内部外设调度器）继续处理直至通知的目标同步时间的最早定时。以这一方式，可以通过保守地处理直至两个类型的外设调度器提供的同步时间来低成本避免在处理器之间的死锁而未执行在处理器之间的时间调整。

[0044] 每个外设层接收最慢进展的处理器的定时作为事件，并且执行自处理直至该时间。另外，基于更新的定时确定下一同步时间通知。

[0045] 以这一方式，仿真按照设施、处理器和外设的顺序进行，并且实现总体上在不同类型之间链接的处理。

[0046] 本发明的效果

[0047] 根据本发明，使用处理器模拟器的反应延迟时间和在处理器模拟器之间的通信延迟时间，并且执行处理器仿真器的推测执行，因此无用于避免死锁的开销并且使用保守事件同步有可能迅速仿真。另外，同步方法基于可以表达任意时间的事件而不是通过按照固定时间间隔同步，因此与时间有关的误差最少的准确仿真是可能的。

附图说明

[0048] 图 1 是图示了用于实现本发明的计算机硬件的示例的框图。

[0049] 图 2 是图示了用于实现本发明的功能块的实施例的图。

[0050] 图 3 是描述处理器、外设和设施中的延迟的图。

[0051] 图 4 是图示了处理器模拟功能的图。

[0052] 图 5 图示了外设模拟操作的示例。

[0053] 图 6 图示了设施模拟操作的示例。

[0054] 图 7 是图示了内部外设调度器的组成单元的图。

[0055] 图 8 是图示了内部外设调度器的过程流程图的图。

[0056] 图 9 是图示了内部外设调度器的过程流程图的图。

[0057] 图 10 是图示了内部外设调度器的过程流程图的图。

[0058] 图 11 是图示了相互外设调度器的组成单元的图。

[0059] 图 12 是图示了相互外设调度器的过程流程图的图。

[0060] 图 13 是外部外设调度器的组成单元的图。

[0061] 图 14 是图示了外部外设调度器的过程流程图的图。

[0062] 图 15 是图示了用于相互外设调度器的 TQ 以及对应 IQ 和 OQ 的事件的时间进入和离开的示例的图。

[0063] 图 16 是图示了逻辑过程的配置的图。

[0064] 图 17 是图示了连接的逻辑过程组的图。

[0065] 图 18 是图示了在空值消息防止死锁时的情形的图。

[0066] 图 19 是用于描述处理器的通信延迟 ΔT_c 的图。

[0067] 图 20 是用于描述处理器的反应延迟 ΔTr 的图。

[0068] 图 21 是用于描述共享式存储器访问的仿真的图。

[0069] 图 22 是用于描述以与处理器通信的高频率在外设上使用与 ISS 相同的核的模拟的图。

[0070] 图 23 是用于描述在通向 CAN 设备的端口期间停止的操作的仿真的图。

[0071] 图 24 是用于描述用于 CAN 设备重置的仿真的图。

[0072] 图 25 是图示了用于实现本发明的功能块的另一实施例的图。

[0073] 图 26 是用于被动地使用被动设施仿真器的外部外设调度器的操作的流程图的图。

具体实施方式

[0074] 下文在参照附图之时描述本发明实施例的配置和过程。在以下描述中，除非另有明示，贯穿附图向相同单元分配相同标号。注意将这里描述的配置和过程描述作为示例，并且应当理解未旨在于使本发明的技术范围限于这些实施例。

[0075] 首先在参照图 1 之时描述用于执行本发明的计算机硬件。在图 1 中，多个 CPU 1 104a、CPU 2 104b、CPU 3 104c、...CPUn 104n 由主机总线 102 连接在一起。主机总线 102 也连接到主存储器 106 以便执行 CPU 1 104a、CPU 2 104b、CPU 3 104c、...CPUn104 的操作。

[0076] 另一方面，八个键盘 110、鼠标 112、显示器 114 和硬盘驱动 116 连接到 I/O 总线 108。I/O 总线 108 通过 I/O 桥接器 118 连接到主机总线 102。键盘 110 和鼠标 112 用于诸如输入命令和点击菜单等之类的操作。显示器 114 用于使用 GUI 如必需的那样显示用于执行过程的菜单。

[0077] IBM® System X 是有利地用于这一目的的硬件 48 计算机系统的示例。这时，CPU 1 104a、CPU 2 104b、CPU 3 104c 例如是 Intel® Xeon®，并且操作系统是 Windows(商标)Server 2003。操作系统优选地具有多任务功能。操作系统存储于硬盘驱动 116 中并且在计算机系统启动期间从硬盘驱动 116 向主存储器 106 读取。

[0078] 为了执行本发明，优选地使用多处理器系统。这里，多处理器系统一般指示使用如下处理器的系统，该处理器具有多个核，这些核具有可以独立地执行计算的处理器功能，因此应当理解可以使用多核单处理器系统、单核多处理器系统或者多核多处理器系统。

[0079] 注意可以用于执行本发明的计算机系统的硬件不限于 IBM® System X，并且可以使用可以运行本发明的仿真程序的任何计算机系统。操作系统不限于 Windows(R)，并且可以使用任何操作系统、比如 Linux® 或者 Mac OS(R)。另外，为了增加仿真程序的操作速度，操作系统可以是基于 Power(商标)6 的诸如 IBM(R)System P 的 AIX(商标)之类的计算机系统。

[0080] 硬盘驱动 116 存储诸如处理器模拟器、外设模拟器、设施仿真器、内部外设调度器、相互外设调度器、外部外设调度器和 CAN 模拟器等程序，并且这些程序中的每个程序在计算机系统启动期间向主存储器 106 中加载并且向 CPU1 至 CPUn 之一分配而且作为单独线程或者进程来执行。因此，图 1 中所示计算机系统优选地具有用于向诸如处理器模拟器、外设模拟器、设施仿真器、相互外设调度器、外部外设调度器、CAN 模拟器等线程中的每个线程分配的充分数目的 CPU。

[0081] 图 2 是图示了在作为单独线程或者进程来操作并且向单独 CPU1 至 CPUn 分配的诸如处理器模拟器、外设模拟器、设施仿真器、相互外设调度器、外部外设调度器、CAN 模拟器等处理程序之间的协作关系的功能框图。

[0082] 这一配置是包含与 ECU、外设模拟器和设施仿真器对应的处理器模拟器的仿真系统。将外设模拟器划分成相互外设和外部外设。注意虽然在图中未图示，但是内部外设应当解释成包含于处理器模拟器块中。总体上，可以在附图的横向方向上获得四层仿真系统。

[0083] 在图 2 中，处理器模拟器 202a、202b、...202z 具有中心功能，该中心功能作为仿真系统中的 ECU 功能的控制器。处理器模拟器 202a、202b、...202z 下载软件代码并且使用指令集仿真器 (ISS) 来执行模拟。

[0084] 外设模拟器 204a、204b、...204z 是事务型模拟器并且优选地由 SystemC/TLM 创建。

[0085] 设施仿真器 206a、206b、...206z 优选地由连续值仿真建模系统、比如 MATLAB®/Simulink® 创建。设施仿真器 206a、206b、...206z 中的每个设施仿真器对应于汽车中的机械设备、比如动力设备如引擎等、动力传送设备如发送系统(transmission)、驱动设备如方向盘和制动设备等。

[0086] 处理器模拟器 202a、202b、...202z 通常例如在 80MHz 使用相对高分辨率的时钟来操作。另一方面，设施仿真器 206a、206b、...206z 是物理机制的仿真器、并且因此例如在 10kHz 的相对低分辨率操作。

[0087] 因此，不能直接地连接处理器模拟器 202a、202b、...202z 和设施仿真器 206a、206b、...206z，因此在它们之间提供外设模拟器 204a、204b、...204z，并且因此外设模拟器发挥作用，从而将来自设施仿真器的状态改变和传感器值转换成事件并且向处理器模拟器传送，或者反言之，将来自处理器模拟器的指令（信令）转换成事件并且向设施仿真器传送。

[0088] 内部外设调度器执行对连接到它的处理器模拟器和外设模拟器的调度。连接到处理器模拟器的外设模拟器处理与诸如 ROM、RAM 和看门狗定时器等其它处理器和设施不直接地有关的外设（内部外设）功能。内部外设调度器具有如下重要功能，这些功能参考 ECU 的存储器映射规范划分从 ISS 到内部外设调度器、相互外设调度器和外部外设调度器的加载 / 存储命令。注意在下文中，也将内部外设调度器描述为处理器调度器，但是应当注意两个名称指示相同功能。

[0089] 外部外设调度器 208 是如下调度器，该调度器处理连接到设施仿真器 206a、206b、...206z 的外设模拟器 204a、204b、...204z。外部外设调度器 208 执行调度以便仅在处理器模拟器反应延迟时间期间（或者在下一事件之前的时间）执行设施仿真器的提前执行。设施仿真器在指定的时间停止，或者如果需要通知更新内部状态则在更早时间停止。另外，提供对执行处理器模拟器的提前执行的通知直至实际设施仿真器 206a、206b、...206z 停止时间。

[0090] 外部外设调度器 208 向设施仿真器 206a、206b、...206z 提供控制器请求的用于信令或者采样的指令。信令是其中控制器致动设施的事件，并且示例是指示开关的接通 / 关断信号的操作。另外，采样是如下事件，其中控制器提供用于在监视（感测）设施的状态时读取值的指令，并且示例是如下操作，其中控制器读取电池单元（设施）的电压值。这里，控制器是向处理器模拟器加载的软件并且是创建用于设施的控制信号的逻辑。无论是否定期地执行用于模拟的过程，用于信令和采样的定时都由控制器确定。

[0091] 另一方面，外部外设调度器 208 从设施仿真器 206a、206b、...206z 接收状态改变事件的通知。这是向控制器通知设施已经改变状态的事件，并且设施确定其定时。这可能有规律地或者无规律地出现。

[0092] 相互外设调度器 210 是如下调度器，该调度器提供在处理器模拟器 202a、202b、...202z 之间连接的相互外设功能。相互外设调度器 210 提供通知以便仅在处理器模拟器 202a、202b、...202z 通信延迟时间（或者在下一事件之前的时间）执行处理器模拟器 202a、202b、...202z 的提前执行。处理器模拟器 202a、202b、...202z 保守地处理直至通

知时间中的最早时间。这时，在处理器模拟器之间的通信由 CAN（控制器区域网络）模拟器 212 执行。使用图 21 来描述如下仿真，其中共享式存储器用于在 ECU 的处理器之间通信。

[0093] 如图 2 中所示，执行从外部外设调度器 208 和相互外设调度器 210 到处理器模拟器 202a、202b、...202z 的事件为中断 (INT)，执行从处理器模拟器 202a、202b、...202z 到外部外设调度器 208 和相互外设调度器 210 的事件为读取写入 (R/W)。

[0094] 参照图 3，如下文所示描述使用在处理器模拟器（或处理器调度器）、相互外设调度器、外部外设调度器、以及设施仿真器中的延迟的提前执行。

[0095] • 首先如图 3(1) 中所示，外部外设调度器 208 引起设施仿真器 206 的提前执行。实际提前时间是 $T_{设施}$。

[0096] • 提前时间的上限是在后继信令时或者更接近 ΔT_R。这里，ΔT_R 是从来自设施仿真器的输入时间直至由此引起的处理器模拟器的反应向设施仿真器传送并且变得有效为止的延迟的最小值，但是在图 20 中提供更具体描述。

[0097] •接着如图 3(2) 中所示，外部外设调度器 208 提供用于提前执行处理器模拟器 202 直至设施仿真器 206 实际地在停止位置 ($T_{设施}$) 的通知。

[0098] • 接着如图 3(3) 中所示，相互外设调度器 210 向处理器模拟器 202 提供用于执行提前执行直至在处理器模拟器 202 之间的通信延迟定时和在下一事件之前的时间中的较早时间的通知。该图图示了其中通信延迟时间更早的示例。

[0099] • 在图 19 中进一步具体描述通信延迟时间 ΔT_c。

[0100] • 接着如图 3(4) 中所示，处理器模拟器 202 继续保守处理直至两个通知时间中的最早定时，但是处理未必继续直至这一时间而无中断，并且例如，如果存在共享式存储器读取 (R)，则将暂时地停止这一时间。该图图示了在过程进行至时间 (3) 之前的时间 (4) 暂时停止的情形。如果未访问共享式存储器，则处理将立刻进行至时间 (3)。

[0101] • 如图 3(5) 中所示，相互外设调度器和外部外设调度器本身继续处理直至从处理器接收的最早事件。

[0102] 接着参照图 4 描述处理器的模拟功能。如上文描述的那样，实现处理器模拟为 ISS（指令集仿真器）。处理器模拟以指定的时间为目标来执行、基于指令（命令语言）输出对访问 (R/W) 的请求或者接收从外设生成的任何有规律中断。

[0103] 具体在这一实施例中，下文记录去往 ISS 的指令和来自 ISS 的请求以便支持描述。

[0104] 去往 ISS 的通知和指令

[0105] advance(t)：向 ISS 指示执行最多直至时间 t。然而如果在达到这一时间之前有必要通信，则 ISS 将中断处理以输出该请求。

[0106] resume(v)：向 ISS 发送与读取请求对应的数据以重启执行。

[0107] interrupt(i)：向 ISS 通知中断 i 已经出现。

[0108] 来自 ISS 的请求和通知

[0109] load(t,a)：请求在时间 t 读取在地址 a 的数据 v。以后过程被中断直至接收恢复通知。

[0110] store(t,a,v)：请求在时间 t 在地址 a 写入数据。

[0111] complete()：如果过程前进至 advance 指示的时间 t，则将进行完成通知。

[0112] 图 4 图示了处理器使用这些命令的模拟操作的示例。在图 4 中，首先，处理器接收

命令 advance(t_g)。处理器在时间 t_1 输出 load(t_1,a_1)。作为响应,处理器接收 resume(v_1)。处理器在时间 t_2 输出 store(t_2,a_2,v_2)。

[0113] 当达到时间 t_g 时,处理器根据先前接收的 advance(t_g) 输出 complete()。接着,处理器可以基于设施的反应接收 interrupt(i)。在这一情况下,将接收下一 advance() 指令。

[0114] 接着,在参照图 5 之时描述外设模拟的操作。如上文描述的那样,优选地使用 SystemC 来构造外设模拟器。外设模拟根据在任务队列中加载的事件执行事务的执行。

[0115] 事件可以是寄存器等的读取请求或者写入请求。外设模拟器按照从任务队列的顶部起的顺序取回这一事件、解译事件、作为事务来执行、向新事件转换、然后向任务队列重新插入。向一个或者多个事件转换是可能的,并且也可能的是未生成事件。事件解析和执行依赖于每个外设的功能。通过重复过程来执行外设模拟直至完成事件。换而言之,重复 pop(移除)、execution(执行)和 push(插入)。

[0116] 利用这一实施例,仿真未接近外设、但是实际上是如下仿真,该仿真链接处理器和设施,因此跨越多个外设模拟器的任务队列切换事件。在这一情况下,外设调度器向适当任务队列添加事件。换而言之,重复 pop、execute、push 或者 export(插入或者传送)、import(导入)。然而导入根据周期是可选的(无)。

[0117] 设施仿真器的输入和输出是事件 <port_update, t, p, v>。这里,端口 p 在时间 t 提供值 y 的通知。例如以这一方式表达去往设施的指令(信令)和来自设施的状态改变通知。

[0118] 从处理器发出的请求包括 <read, t, r> 事件和 <write, t, r, v> 事件,并且这些是请求在时间 t 读取寄存器 r 中的值的事件和请求在时间 t 向寄存器 r 写入值 v 的事件。反言之,处理器接收事件 <register_update, t, r, v>,并且这与请求 <read, t, r> 一致地动作并且提供在时间 t 在寄存器 r 中的值 v 的通知。另外,<interrupt, t, i> 也是接收事件,并且这是如下事件,该事件提供在时间 t 生成中断 I 的通知。

[0119] 具体在这一实施例中,下文记录与外设任务有关的操作以便支持描述。

[0120] insert(e[t]) :向任务队列添加在时间 t 生成的事件 e[t]。

[0121] next() :k :取回任务队列的顶部(弹出)、激发(执行)并且如果必要则返回新事件作为响应。

[0122] Peak()t :返回在任务队列的顶部的事件的时间 t。

[0123] next 返回的新事件包括诸如定时器激发等之类的接近外设的内部事件,但是具体而言,将向其它外设传送的事件是 3 个事件 <interrupt, t, i>、<register_update, t, r, v> 和 <port_update, t, p, v>。

[0124] 自然地,外设的操作不限于这些,但是暂时描述与本发明的操作有关的外设。

[0125] 图 5 图示了外设模拟操作的示例。在图 5(1) 中,首先,任务队列操作命令 insert(e_1[t_1]) 从调度器到达外设,因此 FIFO 添加事件 e_1。如果任务队列为空,或者如果另一事件时间晚于 t_1,则将向任务队列的顶部返回 e_1。这里,e_1 是寄存器读取请求。

[0126] 接着,当操作命令 next() 到达时,在任务队列的顶部的事件被解析、作为事务来执行,并且输出新事件 <register_update, t1, r1, v1>。这一示例是来自输出用于读取寄存器的值的请求的处理器直至获得响应的一系列过程流。将向外部发送事件,因此未向原有

任务队列执行插入（推送）而实际上向处理器的调度器传送（导出）事件。

[0127] 如图 5(2) 中所示，当向寄存器的写入请求 insert(e_2[t_2]) 和 next() 命令到达外设时，类似地，如果在任务队列中挑选的 e_2 在顶部，则将在时间 t_2 弹出 e_2，并且例如将向调度器返回新事件 <port_update, t_2, p_2, v_2>。这一示例是其中处理器向设施发送信令指令的情况。将新事件转换成设施仿真器可以在传送之前解析的形式。

[0128] 图 5(3) 的示例示范如下情形，其中从设施提供状态改变的通知并且使用中断信号向处理器传送该通知。当调度器在向任务队列添加状态改变通知 (insert(e_3[t_3])) 之后取回（弹出）和执行（执行）时，执行向新事件的转换 <interrupt, t, i> 并且向处理器传送（导出）。

[0129] 接着，在参照图 6 之时描述设施仿真的操作。如上文描述的那样，设施仿真优选地由连续值仿真建模系统、比如MTLAB®/Simulink®实现。设施先根据在指定的时间的内部状态（矢量）和外部输入（矢量）计算输出（矢量）。接着输出内部状态的改变部分（矢量），但是这时基于指定的误差范围、设施唯一的最大步进和过零等调整内部状态的改变部分，然后确定下一步进时间。

[0130] 具体在这一实施例中，下文记录设施的操作以便支持描述。

[0131] 去往设施的指令

[0132] Advance(t) :指示 ISS 执行最多直至时间 t。然而如果不能延伸在时间 t 之前的步进，则执行将在该时间之前停止。

[0133] Input(u) :来自设施的、对设施更新输入矢量 u 的通知。

[0134] complete() :不迟于在 advance 中指定的时间完成执行的通知。

[0135] output(t, y) :在时间 t 更新从设施输出的输出矢量 y 的通知。

[0136] 自然地，设施的操作不限于这些，但是暂时描述与本发明的操作有关的设施。

[0137] 注意为了参考，呈现对Simulink®的 S-Function 回调例程的响应。

[0138] mdlGetTimeOfNextVarHit() 调用 complete()

[0139] mdlGetTimeOfNextVarHit() 返回 advance(t)

[0140] mdlOutputs() 调用 output(t, y)

[0141] mdlOutputs() 返回 input(u)

[0142] 图 6 图示了设施仿真操作的示例。在图 6 中，首先命令 advance(t_g) 到达设施。设施在仿真时间 t_1 输出 output(t_1, y_1)。

[0143] 接着，input(u_1) 到达设施。设施在仿真时间 t_2 输出 output(t_2, y_2)。

[0144] 接着，input(u_g) 到达设施。当在仿真时间 t_g 到达时，设施输出 complete()。

[0145] 这一方式，将 advance 和 complete 以及 output 和 input 一起配对。

[0146] 利用本发明，内部外设调度器、相互外设调度器和外部外设调度器执行主要作用。下文首先描述保守事件同步仿真及其术语的概述准备描述三个调度器的操作。

[0147] 首先，图 7 是图示了与内部外设调度器的输入队列和输出队列有关的过程的图。如该图中所示，输入队列包括 IP（来自处理器的输入队列）或者换而言之从其它处理器接收事件（共享式存储器的更新通知）的队列以及 IQ 或者换而言之从相互外设或者外部外设接收消息的队列。P_i 和 P_j 分别代表第 i 个和第 j 个处理器。存储于 IQ 中的数据是 CAN 接收、中断的事件消息或者空值消息。

[0148] 另一方面，输出队列包括 OP（去往处理器的输出队列）或者换而言之向其它处理器发送事件（共享式存储器的更新通知）的队列以及 OQ 或者换而言之向相互外设或者外部外设发送消息的队列。存储于 OQ 中的数据是 CAN 发送事件消息或者进度空值消息。

[0149] 当处理器一起交换事件时，使用 IP 和 OP 作为用于每个连接的处理器的集合。

[0150] 接着，在参照图 8 至图 10 的流程图之时描述内部外设调度器过程。图 8 的流程图的初始值对于所有 IQ/IP 而言为空、对于所有 OQ 而言为时间零进度事件并且对于所有 OP 而言为空，而且 T :＝ 0。

[0151] 另外，到达 IQ/IP 的第 k 个事件的时间 T_k 满足以下条件 :

[0152] $T_k \leqslant T_{k+i}$

[0153] 内部外设调度器在步骤 802 中等待队列输入 e、在步骤 804 中确定是否所有 IQ 被填充（一个或者多个消息已经到达），并且如果填充则在步骤 806 中将所有 IQ 中的最早时间设置成 T_0 并且在步骤 808 中调用子例程“parallel”并且执行与其它处理器并行进行的过程。后文在参照图 9 的流程图之时描述子例程“parallel”。

[0154] 在步骤 810 中，内部外设调度器确定在 IQ 中是否 $T_0 = T$ 或者换而言之在 IQ 中是否存在满足条件 $T_0 = T$ 的事件 e。另外，如果满足条件，则内部外设调度器在步骤 812 中弹出和处理来自 IQ 的 e，

[0155] 然后流程返回到步骤 810。

[0156] 在步骤 810 中，如果在 IQ 中无满足 $T_e = T$ 的事件 e，则过程返回到步骤 804 的判决。

[0157] 接着，在参照图 9 的流程图之时具体描述步骤 808 的“并行”处理。在步骤 902 中，内部外设调度器确定过程是否为“读取”，并且如果是这样，则在步骤 904 中调用子例程“fill”。

[0158] 后文在参照图 10 的流程图之时描述子例程“fill”。在步骤 904 之后，流程前进到步骤 906。如果过程在步骤 902 中不是“读取”，则流程严格地前进到步骤 906。

[0159] 步骤 906 基于公式 $T1 := \min(T_0, T_{top})$ 在 T_1 中插入值。如果 T_{top} 是在 IP 中的顶部时间，如果无 IP（无执行共享式存储器通信的过程），或者如果 IP 为空，则 $T_{top} = \infty$。

[0160] 在步骤 908 中，内部外设调度器以 T_1 为目标执行处理器。换而言之，时间前进至相互外设调度器或者外部外设调度器指示的时间。

[0161] 在步骤 910 中，内部外设调度器在时间 T 前进至该数量之时执行处理器，并且在步骤 912 中执行 OQ/OP 的更新和快闪 (flash)。

[0162] 在步骤 914 中，内部外设调度器确定是否 $T = T_1$，并且如果不是，则在步骤 916 中关于来自 ISS 的最新通知是否为“读取”进行判决，并且如果是这样，则流程返回至步骤 904。

[0163] 在步骤 916 中，如果确定来自 ISS 的最新通知不是“读取”，则在步骤 918 中执行向共享式存储器写入、CAN 通信或者空值消息发送，并且流程返回至步骤 908。

[0164] 在步骤 914 中，如果 $T = T_1$，则流程前进至步骤 920，并且关于在 IP 中是否 $T_e = T$ 或者换而言之是否满足条件 $T_e = T$ 的事件在 IP 中进行判决。

[0165] 在步骤 920 中，如果确定具有事件 e 的事件的时间不在 IP 中，则关于是否 $T = T_0$ 进行确定，并且如果是这样，则终止过程。

[0166] 在步骤 924 中，如果确定 $T = T_0$ 为假，则在步骤 926 中将执行读取共享式存储器

或者接收空值消息，并且流程将返回至步骤 902 的确定。

[0167] 接着，在参照图 10 的流程图之时具体描述步骤 904 的“填充”处理。在图 10 中，内部外设调度器在 IP 队列中创建暂时条目列表。在创建暂时列表之后，移除并且向暂时列表添加 IP 中的具有比当前时间更旧的时间戳的所有消息。

[0168] 在步骤 1004 中，内部外设调度器确定是否填充所有 IP。如果是这样，则在步骤 1006 中按照时间戳顺序示出暂时列表中的（从其它处理器向共享式存储器写入的）所有事件。当处理暂时列表时，终止填充子例程。

[0169] 在步骤 1004 中，如果确定未填充所有 IP，则流程前进至步骤 1008，并且内部外设调度器在此等待来自 IP 的事件 e。

[0170] 在步骤 1010 中，关于是否 T ＞ Te 进行确定，并且如果是这样，则在步骤 1014 中向暂时列表添加和更新 e，并且流程返回至步骤 1008。

[0171] 如果在步骤 1010 中确定 T ＞ T0 为假，则内部外设调度器在步骤 1012 中向 IP 添加 e，并且流程返回至步骤 1004。

[0172] 图 11 是图示了与相互外设调度器的队列有关的过程的图。如该图中所示，来自 P_i、P_j 的事件到达输入队列 IQ 并且向 CAN 模拟器发送。

[0173] 另一方面，当事件从 CAN 模拟器到达输出队列 OQ 时，处理器将取回事件。

[0174] 另外，建立任务队列 TQ，并且这带来 CAN 模拟器的激发。应当注意为了区别于输入队列和输出队列，向任务队列添加的事件被称为任务。

[0175] 接着，将在参照图 12 的流程图之时描述相互外设调度器过程。这时的初始值使得所有 IQ 为空、所有 OQ 为空、TQ 为空并且 T :＝ 0。在步骤 1202 中，相互外设调度器等待更新 IQ，在步骤 1204 中，关于是否所有 IQ 被填充进行确定，并且如果不是这样，则流程返回至步骤 1202，并且继续等待 IQ 的更新。

[0176] 另外，在步骤 1204 中，如果相互外设调度器确定所有 IQ 被填充，则在步骤 1206 中，在所有 IQ 中的最接近时间视为 T_0。

[0177] 在步骤 1208 中，相互外设调度器确定在 IQ 中是否 Te ＝ T_0 或者换而言之在 IQ 中是否存在 T_e ＝ T_0 的任务 e。如果是这样，则相互外设调度器在步骤 1210 中弹出事件 e 并且在步骤 1212 中使用 insert(e_1[t])、insert(e_2[t]) 来更新 TQ。这里，事件 e_1 是 <read，t，r> 或者换而言之是在时间 t 读取寄存器 r 的事件，并且事件 e_2 是 <write，t，r，v> 或者换而言之在时间 t 将寄存器 r 中的值更新成 v 的事件。

[0178] 在步骤 1212，流程返回至步骤 1208，并且如果相互外设调度器确定在 IQ 中不存在 Te ＝ T_0 的任务 e，则在步骤 1214 中设置条件 T :＝ T0，并且在步骤 1216 中，peak() 用来确定在 TQ 中是否 T_k ＝ T 或者换而言之在 TQ 中是否存在 T_k ＝ T 的任务 k。

[0179] 在步骤 1216 中，关于在 TQ 中是否有 T_k ＝ T 的任务 k 进行确定，并且相互外设调度器在步骤 1218 中弹出 k 并且在步骤 1220 中使用 next() 来执行任务。这里，当接收 CAN 数据时生成作为通知向处理器提供的事件 <interrupt，t，i>、<register_update，t，r，v>。

[0180] 在步骤 1222 中，相互外设调度器更新所有 OQ，并且流程返回值步骤 1216。

[0181] 在步骤 1216 中，如果确定在 TQ 中无 T_k ＝ T 的任务 k，则流程进行步骤 1224，并且这里相互外设调度器确定是否已经更新 OQ 队列。

[0182] 如果尚未更新 OQ 队列，则相互外设调度器在步骤 1226 中向所有 OQ 添加用于时间

$\min(T+\Delta T_C, T_{top})$ 的空值消息、在步骤 1228 中快闪 OQ，然后过程返回至步骤 1204。注意这里 ΔT_C 是处理器的通信延迟时间。

[0183] 如果已经更新 OQ，则相互外设调度器在步骤 1228 中快闪 OQ，并且然后流程返回至步骤 1204。

[0184] 为了参考，图 15 是图示了用于相互外设调度器的 TQ 以及对应 IQ 和 OQ 的事件的时间进入和离开的示例的图。虚线箭头示出了访客时间的进度。

[0185] 在这一示例中，从处理器调度器（内部外设调度器）到达 IQ 的消息的事件按照时间顺序是 CAN 放弃、CAN 发送、空值（进度消息事件）和 CAN 重置。除了空值之外的上述消息都是向 CAN 设备发送命令（用于恰当执行寄存器）的处理器生成的事件。在这些消息之中，CAN 放弃是发送如下命令的事件，该命令用于如果仲裁失败则取消发送待命条件并且向处理器发送中断通知。CAN 发送是发送如下命令的事件，该命令用于发送已经预先设置的数据。图 15 图示了如下示例，其中来自两个不同处理器的 CAN 发送事件连续地到达。CAN 重置是发送如下命令的事件，该命令从总线分离 CAN 设备并且恢复初始条件。

[0186] 在这一示例中，向处理器调度器发送的消息的事件按照时间顺序是空值（空值消息事件）、放弃通知和 CAN 接收。CAN 接收是如下事件，其中 CAN 设备通过中断向处理器通知已经接收 CAN 数据。放弃通知是如下事件，其中 CAN 设备通过中断向处理器通知仲裁已经失败。

[0187] 当 CAN 发送事件出现（到达 IQ）时，先前发送发起的数据目前可能在发送中。在图 15 的示例中，当第二 CAN 发送事件到达时，与先前已经开始的传送完成结合向 TQ 的顶部添加（第一）CAN 接收事件。在 CAN 接收事件之后立即执行仲裁以便发送后续数据。在这一幅图的示例中，CAN 接收事件和仲裁事件定时相同以求简化。在空值事件到达 IQ 时的时间戳与用于 CAN 接收和仲裁的访客时间相同。这里，本发明的相互外设调度器包括如下过程，该过程向处理器调度器通知（向处理器调度器发送空值消息）执行提前执行直至在 TQ 中的顶部时间。在图 15 中的 IQ 到达的空值是进度消息，其中处理器调度器提供处理器在这一过程的通知定时之前到达的通知。另外，图中的 OQ 初始地发送的空值消息向处理器调度器指示这一过程向处理器调度器通知提前执行。

[0188] 当执行仲裁时，接受从两个处理器请求的 CAN 发送中的一个（利用仲裁）而将另一个置于保持（仲裁失败）。在该图中，去往进行第一发送请求的处理器的 CAN 接收事件是向执行第二发送请求的处理器提供放弃通知的事件，但是示出了向 TQ 添加二者。与 CAN 设备的模拟进度结合按顺序处理两个添加的事件，并且通过 OQ 向处理器调度器通知对应事件（放弃通知、CAN 接收）。

[0189] 图 13 是图示了与外部外设调度器的队列有关的过程的图。如该图中所示，来自 P_i、P_j 的消息到达输入队列 IQ。

[0190] 向输出队列 OQ 发送中断通知的事件消息和空值消息，并且这些取回 P_i、P_j。

[0191] 按照时间戳对任务队列 TQ 排序。

[0192] 另外，外部外设调度器保持连接到设施的 I/O 端口的当前信号值。

[0193] 接着，在参照图 14 的流程图之时描述外部外设调度器过程。这时的初始值使得所有 IQ 为空、所有 OQ 为空、TQ 为空、PV（端口值）是 ECU 规范的初始值、$T := 0$ 并且 $T_R := \Delta T_R$。

[0194] 在步骤 1402 中，外部外设调度器等待 IQ 的更新，在步骤 1404 中关于是否填充所有 IQ 进行确定，并且如果不是，则流程返回至步骤 1402，并且继续等待 IQ 的更新。

[0195] 另外，在步骤 1404，如果外部外设调度器确定所有 IQ 都被填充，则在步骤 1406 中，在所有 IQ 中的最近时间视为 T_0。这里，$T \leqslant Te \leqslant T_p$，并且 T_p 是当前设施时间。

[0196] 在步骤 1408 中，外部外设调度器确定在 IQ 中是否 $T_e = T_0$ 或者换而言之在 IQ 中是否有 $T_e = T_0$ 的任务 e。如果是这样，则外部外设调度器在步骤 1410 中弹出事件 e 并且在步骤 1412 中使用 insert(e_1[t])、insert(e_2[t]) 来更新 TQ。这里，事件 e_1 是 <read，t，r> 或者换而言之是在时间 t 读取寄存器 r 的事件，并且事件 e_2 是 <write，t，f，v> 或者换而言之在时间 t 将寄存器 r 中的值更新成 v 的事件。

[0197] 在步骤 1412 之后，流程返回至步骤 1408，并且如果外部外设调度器确定在 IQ 中无 $Te = T_0$ 的任务 e，则在步骤 1414 中设置条件 T :＝ T0，并且在步骤 1416 中，关于是否 $T = T_p$ 进行确定。

[0198] 如果在步骤 1416 中确定 $T = T_p$，则外部外设调度器将在步骤 1418 中调用 insert(e_4[T]) 并且将向 TQ 添加在时间 T 激发的任务 <plant_go>。这里，e_4 是在时间 $\min\{T_{top}, T+\Delta T_R\}$ 执行设施的事件。T_{top} 是 TQ 的顶部时间，并且如果 TQ 为空，则 $T_{top} = \infty$。换而言之，如果所有处理器跟踪设施，则这将是设施的主要执行的过程。

[0199] 在步骤 1418 之后，流程前进至步骤 1420 的确定步骤。如果在步骤 1416 中确定 T ＝ Tp 为假，则流程直接前进至步骤 1420。

[0200] 在步骤 1420 中，外部外设调度器调用 peek() 并且确定在 IQ 中是否存在 $T_k = T$ 的事件 k。如果没有，则流程向步骤 1404 返回。

[0201] 在步骤 1420 中，关于在 TQ 中是否有 $T_k = T$ 的事件 k 进行确定，并且外部外设调度器在步骤 1422 中弹出 k 并且在步骤 1424 中确定设施 k 是否为设施执行的。

[0202] 如果外部外设调度器在步骤 1420 中确定 k 为执行的设施，则在步骤 1426 中调用 advance()，并且执行设施。

[0203] 在后续步骤 1428 中，外部外设调度器执行 TQ 的更新。具体而言，接收 output(t，y)，执行 pusg(e3[t])，并且关于 PV 进行 input(u) 的通知。另外发送空值消息。这一事件使用 <port_update，p，v> 来更新端口 p 值 v。

[0204] 在下一步骤 1430 中，外部外设调度器更新 Tp，并且在下一步骤 1432 中确定过程是否完成。如果确定过程完成，则流程返回至步骤 1420。如果确定过程未完成，则流程在步骤 1434 中返回任务之后返回至步骤 1420。

[0205] 返回至步骤 1424，如果确定 k 并非设施执行的，则外部外设调度器前进至步骤 1436、调用 next() 并且执行任务。具体而言，这是指执行 <interrupt>、<register value>、<register_update> 和 <port_update> 并且发送空值消息。

[0206] 在后续步骤 1438 中，外部外设调度器执行 OQ 的更新。这里，如果向 OQ 添加除了空值之外的消息，则向 TQ 添加在时间 $T+\Delta T_R$ 激发的任务。换而言之，将执行 insert(e_4[$T+\Delta T_R$])，但是具体而言，将执行设施直至时间 $\min\{T_{top}, T+\Delta T_R\}$。这里，$T_{top}$ 是 TQ 的顶部时间，并且如果 T 为空，则 $T_{top} = \infty$。

[0207] 在下一步骤 1440 中，外部外设调度器快闪所有 OQ，并且流程返回至步骤 1420。

[0208] 接着，下文描述当执行保守事件同步时出现死锁和需要空值消息以避免这一死

锁。注意这不是新发现而是常规已知的发现。

[0209] 如果拆分仿真对象并且连接每个区域的仿真器（或者模拟器）以执行整个对象的仿真，则包括通信接口的每个区域的仿真单元称为逻辑过程。图 16 图示了逻辑过程的配置。

[0210] 如图 16 中所示，在保守事件同步仿真期间，逻辑过程 (LPk) 包括用于消息接收的输入队列 IQ、用于消息发送的输出队列 OQ、记录调度的仿真任务的任务队列 TQ4 和指示当前事件的学者 (scholar) 值。IQ 和 OQ 是 FIFO，并且按时间对 TQ 排序。

[0211] 图 17 是图示了如下情形的图，其中通过通信通道连接多个逻辑过程 (LP1 至 LP4)。各种逻辑过程具有用于每个通信通道的 IQ 和 DQ 以便提供这一通信。

[0212] 逻辑过程交换具有时间信息（时间戳）的消息。用于由 IQ/OQ 发送 / 接收的消息的时间戳每当执行发送 / 接收时稳定地增加。因此，如果逻辑过程从任何其它逻辑过程接收消息，则仿真可以进行而不向接收的消息的最小时间戳回滚。交换的消息包括事件消息和空值消息。

[0213] 与在时间 t 的时间 e 有关的事件消息是陈述“从发送侧向接收侧生成的下一事件的时间是 t 并且生成的事件是 e”的消息。

[0214] 时间 t 的空值消息是陈述“从发送侧向接收侧生成的下一事件的时间将在 t 之后”的消息并且对接收侧上的实际仿真区域无影响。空值消息的目的如下文描述的那样是防止死锁。利用保守事件同步，停止过程直至从所有其它逻辑过程接收具有比当前时间戳更大的时间戳的消息以便防止回滚。然而如图 18(a) 中所示，如果等待消息接收的其它逻辑过程在相同时间等待从该逻辑过程接收，则两个逻辑过程将等待彼此的消息并且处理不能重启。因此，执行“提前读取”以确定从当前时间戳往回有多久尚未生成去往外界的事件并且如图 18(b) 中所示通过发送空值消息来防止死锁。

[0215] 下文示出用于逻辑过程的过程的概述。

[0216] (1) 等待直至消息到达所有 IQ

[0217] (2) 弹出 IQ 中的最早时间的消息 ($t = T_0$)

[0218] (3) 在向 T_0 转发当前时间之时执行仿真（弹出任务队列并且执行）

[0219] (a) 向任务队列添加将来事件

[0220] (b) 向 OQ 序列地添加针对外界而生成的事件

[0221] (c) 向其中未添加事件的那些 OQ(t ＝当前时间 + 提前自由时间）添加空值消息

[0222] 处理在 (2) 中弹出的消息并返回至 (1)。

[0223] 注意利用本发明，其中未执行“提前读取”的空值消息（具有与时间戳的发送相同的当前时间的否消息）称为进度消息。进度消息的目的是避免具有微小提前读取宽度的空值消息所引起的频繁同步。利用本发明，内部外设调度器发送进度消息。这是因为驱动内部外设调度器的 ISS 仅能预测具有微小宽度的另一逻辑过程的下一事件。

[0224] 在参照图 19 之时描述处理器通信延迟 ΔT_c。换而言之，如该图中所示，从向相互外设写入直至向接收侧 / 发送例处理器生成中断为止的延迟的最小值是处理器通信延迟 ΔT_c。

[0225] 附带提一点，当通过具有 1.0Mbps 的最大波特率的 CAN 通信时，数据帧的最大延迟是 44μs（帧最小 bid 计数是 44），并且这是用于 ΔT_c 的参考值。

[0226] 接着，在参照图 20 之时描述处理器反应延迟 ΔT_R。换而言之，如该图中所示，处理器反应延迟 ΔT 是用于从设施输入事件直至向设施传送对应处理器反应并且已经变成有效为止的延迟的最小值。图 20 中的 T_R 是处理器有可能在 TQ 为空时到达设施的最少时间。

[0227] 作为前提条件，在从接收设施的状态改变所引起的中断起的 T_R 内生成去往设施的确定初始事件生成时间 $T_{反应}$ 的寄存器中断。如果这一中断时间由 $T_{触发}$ 表达，则建立以下公式。

[0228] $T_{触发} << T_R << T_{反应}$

[0229] 即使使用从处理器确定操作（时间 $T_{触发}$）直至致动器操作（时间 $T_{反应}$）为止的时间来提前执行设施作为提前读取，仍然不会损失时间精确度，并且基于使用这一发现来使用处理器反应延迟 ΔT_R。

[0230] 接着举例说明处理器反应延迟的示例。作为第一示例，利用引擎燃料注入控制，如果曲柄的每 15° 从设施生成旋转极（状态改变），则将使用定时器来执行燃料注入和激发指令以便使定时准确。当生成前述旋转脉冲中的一个或者多个旋转脉冲时计算用于定时器的时间设置，因此在注入和激发之前将存在一个或者多个脉冲间隔，或者换而言之，将在 10,000RPM 的最大旋转速度有为 250μs 的脉冲间隔或者更多的充分延迟。这是反应延迟。

[0231] 作为另一示例，利用如下控制，其中处理器关于来自设施的输入立即返回指令，反应延迟将极小。然而利用这一类型的控制，经常关于与处理器相比缓慢的物理现象（示例：去往风扇的 PWM 信号的指令，其中温度是受控变量；用于节流打开的指令，其中车辆速度是受控变量）执行循环任务。用于温度、速度和转矩的控制周期的近似值分别为 1 秒、1ms 和 0.1ms，并且无需在比这些更小的时间单位向设施提供指令。因此，使用取决于设施的时间单位大小的时间作为反应延迟。

[0232] 接着，在参照图 21 之时描述共享式存储器访问的仿真。在图 21 中，处理器 Pr1、Pr2、Pr3 访问共享式存储器。因此，处理器 Pr1 和 Pr2 领先，并且处理器 Pr3 跟随处理器 Pr1 和 Pr2。因此，处理器 Pr3 基于 IQ 的内容理解这是在最晚时间并且根据以往知道第二处理器的定时。因此在跟踪第二处理器之时执行存储和加载。最终，停止在加载之前立即出现，并且通过在已经按照时间顺序反应已经到达 IQ 的所有以往存储操作之后执行加载来实现重启。

[0233] 因此，所有处理器必须在到达“最旧时间”之后重启，因此避免死锁。反言之，利用其中以这一方式循环地执行 R/W 的逻辑过程，死锁不会出现。因此，仿真在维持读取和写入的完整性之时在是可能的而无死锁。

[0234] 图 22 图示了向主机计算机的核分配本发明的 ISS、外设模拟器、设施仿真器和三个调度器的功能的示例。可以通过使用相同核 2204 模拟与处理器频繁通信的内部外设调度器和 ISS 来在高频率执行仿真。

[0235] 内部外设调度器通过参考 ECU 的存储器映射规范来划分从 ISS 到内部外设（在核中的 SystemC）、相互外设和外部外设的加载 / 存储命令。

[0236] 注意，该图中的表达式如下文所示。

[0237] <ISS>

[0238] int：中断

[0239] rs：恢复

[0240] a :提前

[0241] l :加载

[0242] s :存储

[0243] c :完成

[0244] <SystemC>

[0245] r :读取

[0246] w :写入

[0247] ru :寄存器更新

[0248] int :中断

[0249] pu :端口更新

[0250] < 设施 >

[0251] a :提前

[0252] i :输入

[0253] c :完成

[0254] o :输出

[0255] < 通信通道 >

[0256] r :读取

[0257] w :写入

[0258] ru :寄存器更新

[0259] int :中断

[0260] 接着，使用图 23 来描述如下事实 :在 CAN 设备调制已经失败之后向处理器提供放弃通知的操作可以由相互外设调度器保守地模拟（而未引起回滚）。图 23 图示了其中 2 个处理器 1、2 产生向 CAN 设备的 CAN 帧发送请求的示例。处理器 1 请求发送的帧具有比处理器 2 已经请求发送的帧更高的优先级。处理器 2 向 CAN 设备发送命令，从而如果仲裁失败，则将取消发送待命状态并且将产生中断通知（放弃集合）。另外，在该图的竖直方向上的虚线箭头示出了访客时间的进度。

[0261] 在图 23 的示例中，CAN 设备模拟器（表示为 SystemC 等）在两个处理器朝着 CAN 设备（相互外设）执行放弃集合和发送集合之后开始仲裁过程。CAN 设备的模拟器向 TQ 添加开始仲裁的事件，并且基于这一时间同步所有处处理器与访客时间（仲裁开始）。在这一示例中，放弃集合、2 个发送集合和仲裁对应于到达 IQ 的 CAN 放弃的仲裁事件、到达 IQ 的 2 个 CAN 发送、和向 TQ 添加的仲裁事件。另外，在图 23 中，与处理器时间同步的相互外设调度器的过程是接收进度消息并且在 TQ 的顶部时间发送空值消息而且是关于图 15 描述的相同过程。

[0262] 另外，图 23 的示例图示了 CAN 设备模拟器在仲裁开始时向 TQ 添加失败通知时间的事件。当相互外设调度器向下一处理器提供提前执行的通知时，发送的空值消息的时间戳不会是 ΔT_c，但是实际上将添加失败通知的时间。因此，处理器 2 不会传递失败通知时间（需要回滚的开销）。

[0263] 接着，使用图 24 来描述如下事实 :已经请求发送高优先级 CAN 帧的处理器重置 CAN 设备可以由相互外设调度器保守地模拟（而未引起回滚）。这通过在 CAN 设备模拟器中

校验在失败通知之前出现重置而变得可能。图 24 中所示操作与图 23 相同直至仲裁开始。

[0264] 图 24 的示例举例说明针对具有高优先级的 CAN 帧发送请求的处理器将在访客时间中的仲裁开始与仲裁完成（对帧的成功或者失败的确认）之间重置 CAN 设备。如图 23 的示例中所示，CAN 设备的模拟器在开始仲裁的时刻向 TQ 添加用于向处理器 2 提供放弃通知的事件。

[0265] 在图 24 的示例中，当处理用于提供添加的放弃通知的事件时，CAN 设备的模拟器在提供放弃通知之前校验是否产生重置通知并且确定是否提供放弃通知。

[0266] 通过执行这一确定，如果已经产生重置通知，则 CAN 设备的模拟器重新计算哪个 CAN 帧已经赢得仲裁。如果已经被调度成在仲裁开始时输掉的帧作为重新计算的结果而赢得仲裁，则不会生成放弃，因此不会向处理器通知放弃。图 24 的示例图示了不会由于第二处理器 2 请求发送的（被调度成在仲裁开始时输掉的）CAN 帧作为重新计算的结果赢得仲裁而产生调度的放弃通知。

[0267] 如果已经被调度在仲裁开始时输掉的帧作为重新仲裁的结果而输掉仲裁，则将产生放弃，因此将向调度器通知如调度的放弃。另外，如果在前述确定过程中未产生重置通知，则无法知道 CAN 帧何时赢得仲裁。因此，如果未进行重置通知，则将向处理器通知在仲裁开始时调度的放弃。

[0268] 因此，即使未通过重置来生成放弃，处理器仍然不会无意中接收放弃通知（需要回滚的开销），因为 CAN 设备的模拟器将在通知输掉仲裁之前校验当前重置。

[0269] 接着，下文在参照图 25 之时描述本发明的另一实施例。在图 25 中，处理器模拟器 2502a、...2502m、2504a、...2504n 具有与图 2 中所示处理器模拟器 202a、...202z 实质上相同的功能。

[0270] 外设模拟器 2506a、...2506m、2508a、...2508n 也具有与图 2 中所示外设模拟器 204a、...204z 实质上相同的功能。

[0271] 图 25 的配置与图 2 的配置不同在于将设施仿真器划分成主动设施仿真器 2510a、...2510m 和被动设施仿真器 2512a、...2512n。这里，动态设施仿真器是具有如下功能的诸如引擎仿真器之类的设施仿真器，该功能自治地确定状态改变的定时等。例如引擎是每 15° 使用脉冲来提供曲柄角度信息的设施，但是控制器 (ECU) 侧不能确定定时。

[0272] 另一方面，被动设施仿真器是其中控制器可以确定定时的设施仿真器，并且示例包括在指定的定时监视（采样）电池电压的设施。被动设施的其它示例包括诸如温度传感器、压强传感器、转速传感器等传感器型设施。如果通过采样来检测状态，则也将刹车和悬架分类为被动设施。

[0273] 因此，提供外部外设调度器——调度连接到主动设施仿真器的外设模拟器的外部外设调度器 2514a 和调度连接到被动设施模拟器的外部外设调度器 2514b——以便适应这些主动设施仿真器和被动设施仿真器。关于连接到主动设施仿真器的外设模拟器，对应处理器模拟器一方面提供信号更新的通知并且也工作以接受状态改变，但是对于连接到被动设施仿真器的外设模拟器，对应处理器模拟器作为规则仅提供信号更新通知和采样提前通知。

[0274] 分离主动设施仿真器和被动设施仿真器将基本上对在处理器模拟器之间的通信无影响，因此相互外设调度器 2516 的操作与图 2 的相互外设调度器 210 实质上相同，并且

CAN 模拟器 2518 也与图 2 的 CAN 模拟器 212 实质上相同。

[0275] 附带提一点，用于主动设施仿真器的外部外设调度器 2514a 与图 2 的外部外设调度器 208 在功能上实质上相同，但是用于被动设施仿真器的外部外设调度器 2514b 无需计算使设施相对于处理器提前的时间宽度的过程。

[0276] 因此，在参照图 26 的流程图之时描述外部外设调度器 2514b 的操作。这时的初始值使得所有 IQ 为空、所有 OQ 为空、TQ 为空、PV（端口值）是 ECU 规范的初始值、T :＝ 0。用于被动设施仿真器的外部外设调度器仅需重复直至用于每个提前处理器时间通知的时间的过程。

[0277] 在步骤 2602 中，外部外设调度器 2514b 等待 IQ 的更新，在步骤 2604，关于是否填充所有 IQ 进行确定，并且如果不是，则流程向步骤 2602 返回，并且继续等待 IQ 的更新。

[0278] 另外，在步骤 2604 中，如果外部外设调度器 2514b 确定所有 IQ 都被填充，则在步骤 1406 中，在所有 IQ 中的最接近时间视为 T_0。

[0279] 在步骤 2608 中，外部外设调度器 2514b 确定在 IQ 中是否 $T_e = T_0$ 或者换而言之在 IQ 中是否存在 $T_e = T_0$ 的任务 e。如果是这样，则外部外设调度器 2514b 在步骤 2610 中弹出任务 e 并且在步骤 2612 中更新 TQ。

[0280] 在步骤 2612 之后，流程返回至步骤 2608，并且如果外部外设调度器 2514b 确定在 IQ 中无 $T_e = T_0$ 的任务 e，则在步骤 2614 中设置条件 T :＝ T0。

[0281] 接着，流程前进至步骤 2616，其中外部外设调度器 2514b 执行设施直至 T。这时，设施是被动设施，因此外部外设调度器 2514b 未接收状态改变的通知。

[0282] 接着，外部外设调度器 2514b 执行 TQ entry(T)，并且流程返回至步骤 2604。

[0283] 注意描述了如下情况，其中 CAN 是相互外设调度器操纵的汽车 LAN 协议，但是本发明的仿真系统不限于 CAN 并且可以使用诸如 LIN 或者 FlexRay 等任何任意汽车 LAN 协议。

[0284] 上文关于用于汽车的多个仿真系统描述本发明的具体实施例，本领域技术人员应当理解本发明不限于这些具体实施例并且可以被应用于诸如用于飞机的仿真系统等通用机电控制的系统的仿真。

[0285] 另外，本发明不限于具体计算机架构和平台并且可以被应用于可以执行多任务的任何平台。

[0286] 符号描述

[0287]	102	主机总线
[0288]	104a、...104n	CPU
[0289]	106	主存储器
[0290]	116	硬盘驱动
[0291]	202、2502、2504	处理器模拟器
[0292]	204、2506、2508	外设模拟器
[0293]	206、2510、2512	设施模拟器
[0294]	208、2514a、2514b	外部外设调度器
[0295]	210、2516	相互外设调度器
[0296]	212、2518	CAN 模拟器

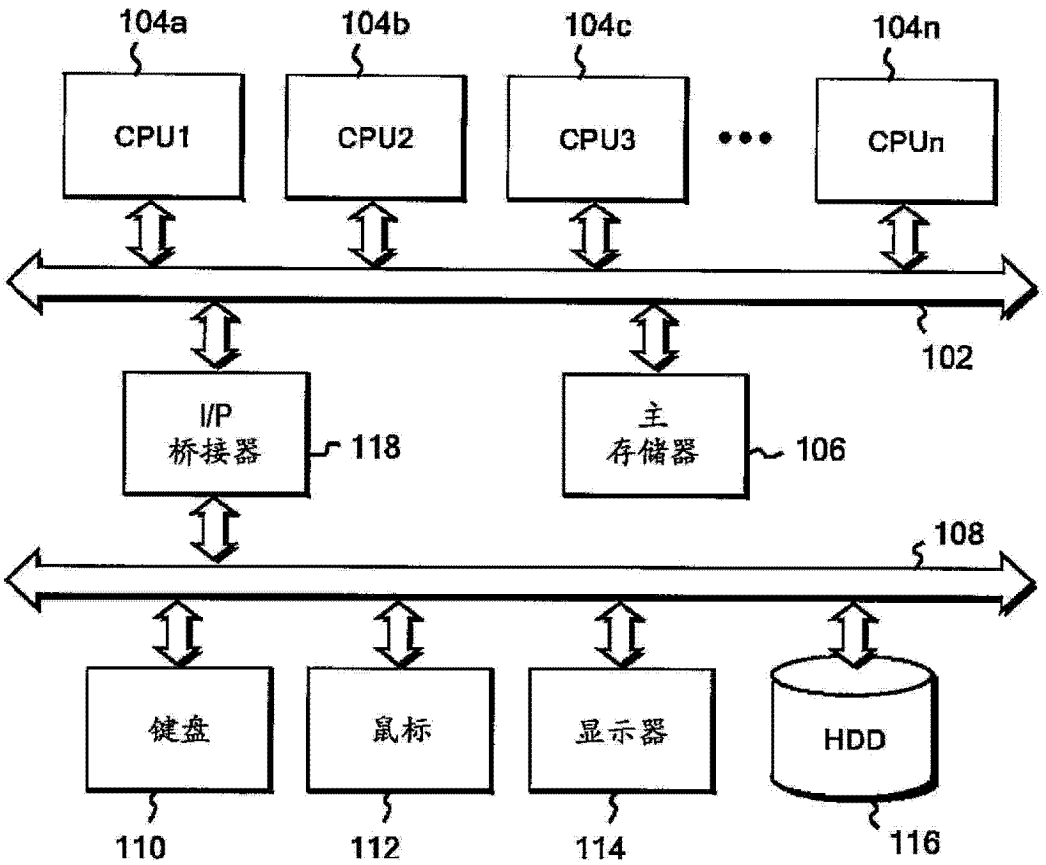

104a
104b
104c
104n
CPU1
CPU2
CPU3
CPUn
102
I/P
桥接器
118
主
存储器
106
108
键盘
鼠标
显示器
HDD
110
112
114
116

图 1

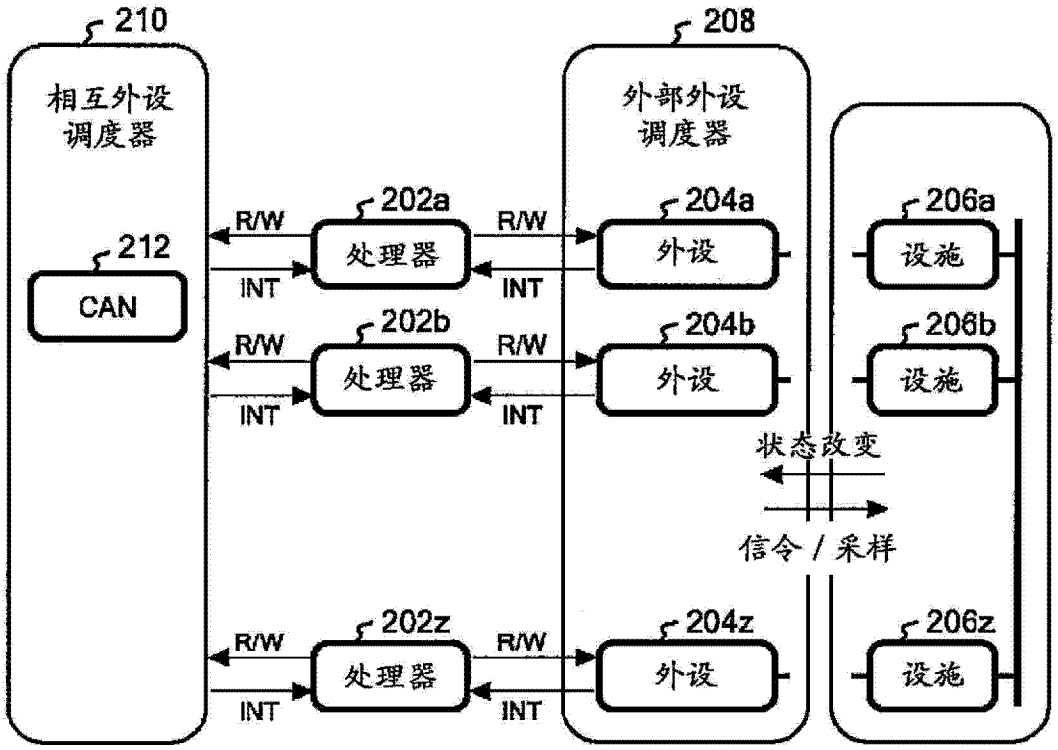

210
相互外设
调度器
212
CAN
208
外部外设
调度器
202a
处理器
204a
外设
206a
设施
R/W
INT
202b
处理器
204b
外设
206b
设施
状态改变
信令 / 采样
202z
处理器
204z
外设
206z
设施

图 2

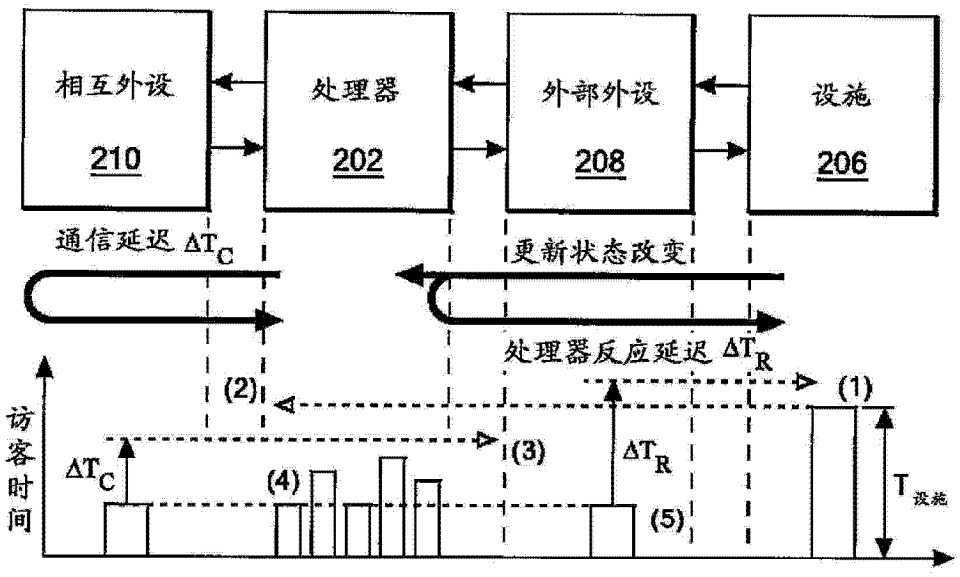

相互外设
210
处理器
202
外部外设
208
设施
206
通信延迟 ΔTC
更新状态改变
处理器反应延迟 ΔTR
访客时间
(1)
(2)
(3)
(4)
(5)
ΔTC
ΔTR
T设施

图 3

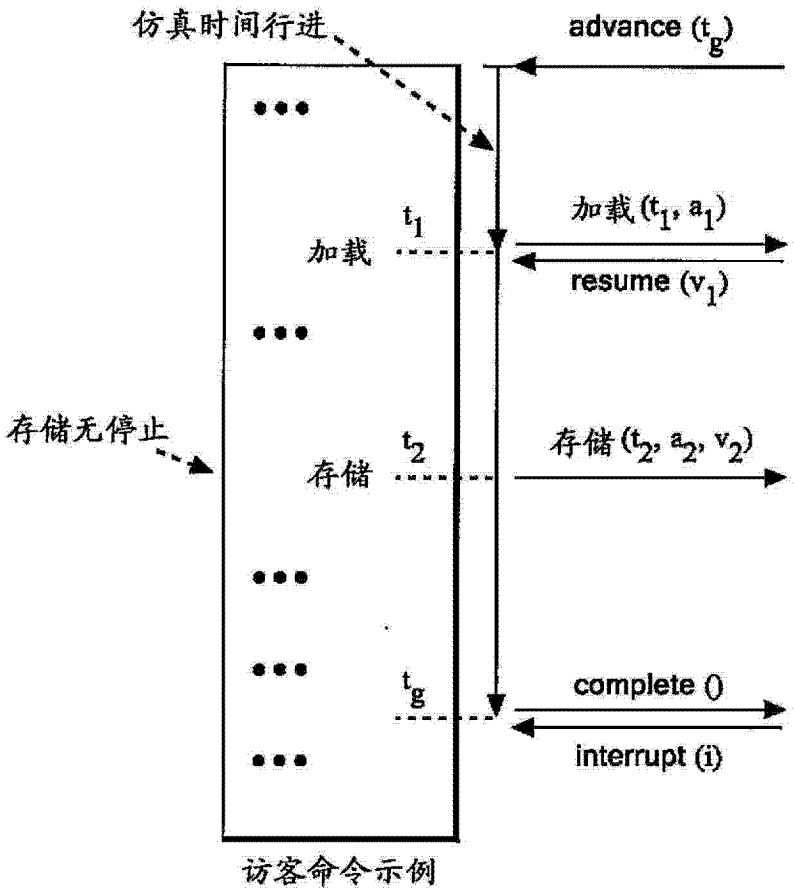
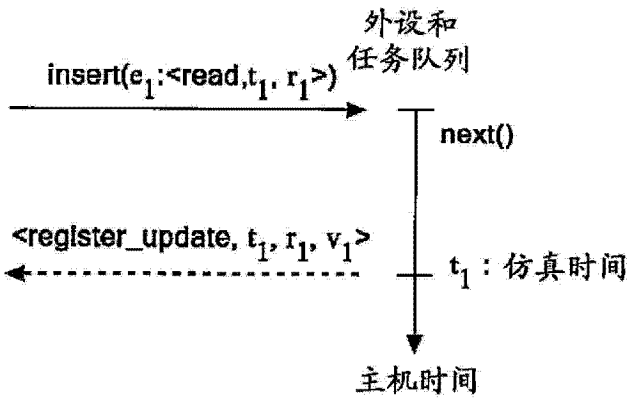
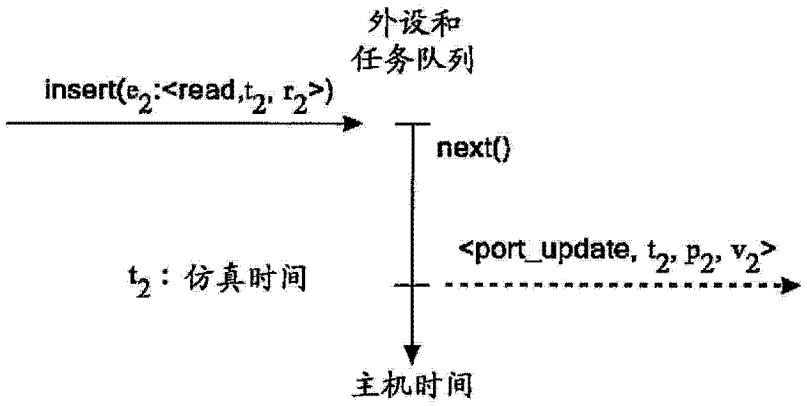

仿真时间行进
advance (t_g)
加载
t_1
加载 (t_1, a_1)
resume (v_1)
存储无停止
存储
t_2
存储 (t_2, a_2, v_2)
t_g
complete ()
interrupt (i)
访客命令示例

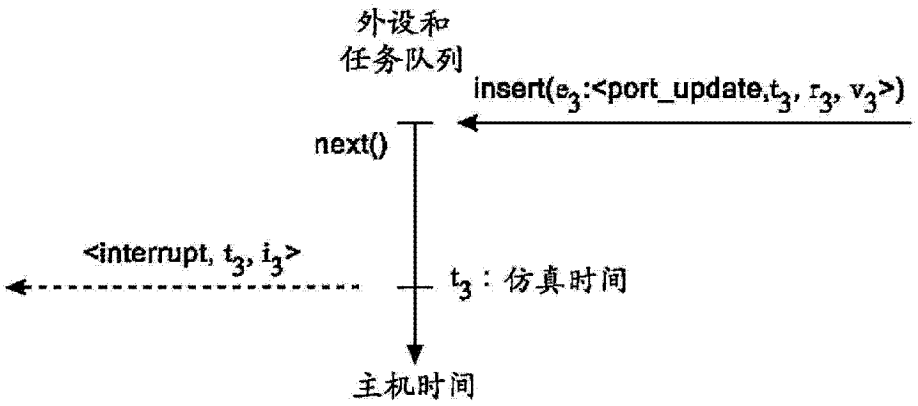
图 4



(1) 来自处理器的查询



(2) 从处理器到设施的信令



(3) 从设施到处理器的状态改变通知

图 5

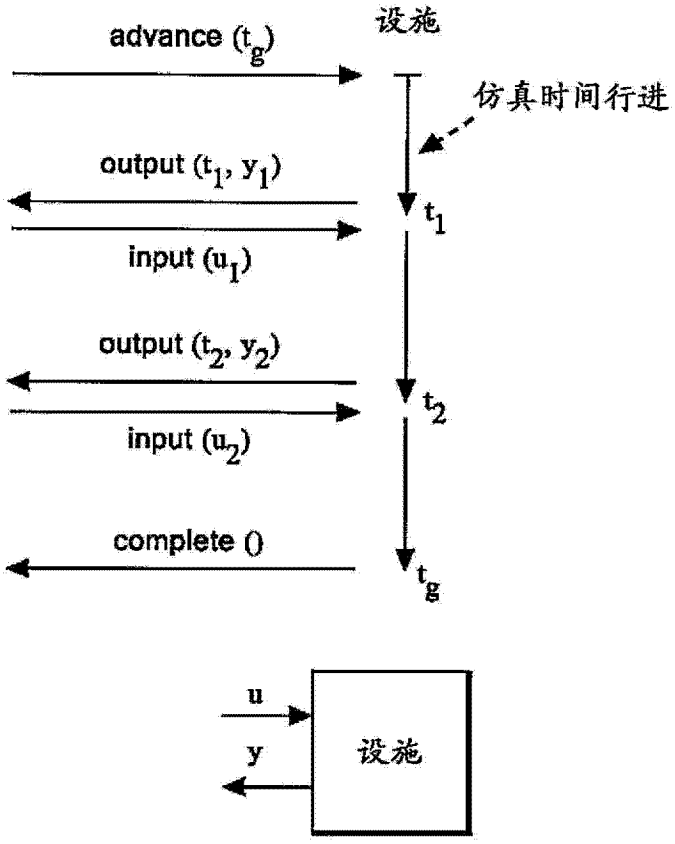

advance (t_g)
设施
仿真时间行进
output (t_1, y_1)
t_1
input (u_1)
output (t_2, y_2)
t_2
input (u_2)
complete ()
t_g
u
y
设施

图 6

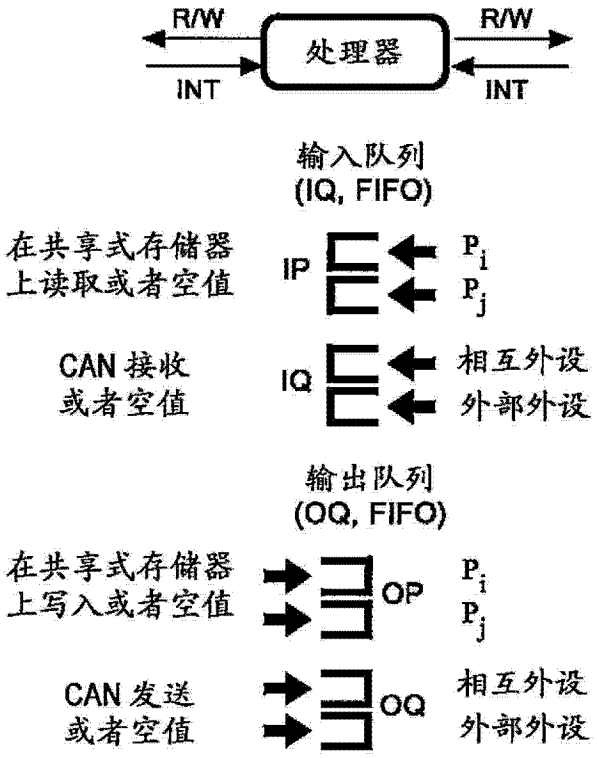

R/W
R/W
处理器
INT
INT
输入队列
(IQ, FIFO)
在共享式存储器
上读取或者空值
IP
P_i
P_j
CAN 接收
或者空值
IQ
相互外设
外部外设
输出队列
(OQ, FIFO)
在共享式存储器
上写入或者空值
OP
P_i
P_j
CAN 发送
或者空值
OQ
相互外设
外部外设

图 7

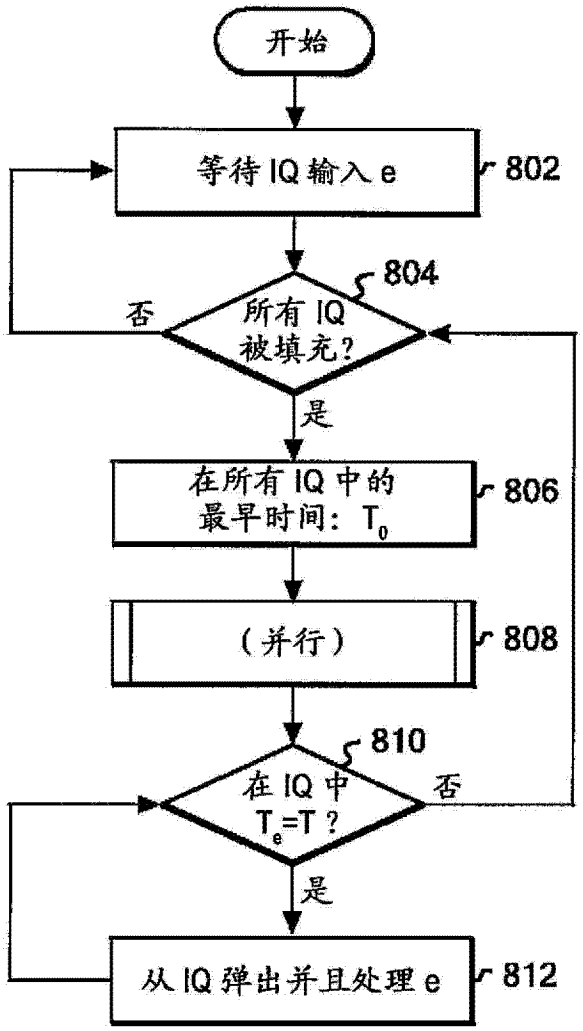

开始
等待 IQ 输入 e
802
804
所有 IQ 被填充?
否
是
在所有 IQ 中的最早时间: T_0
806
(并行)
808
810
在 IQ 中 T_e=T ?
否
是
从 IQ 弹出并且处理 e
812

图 8

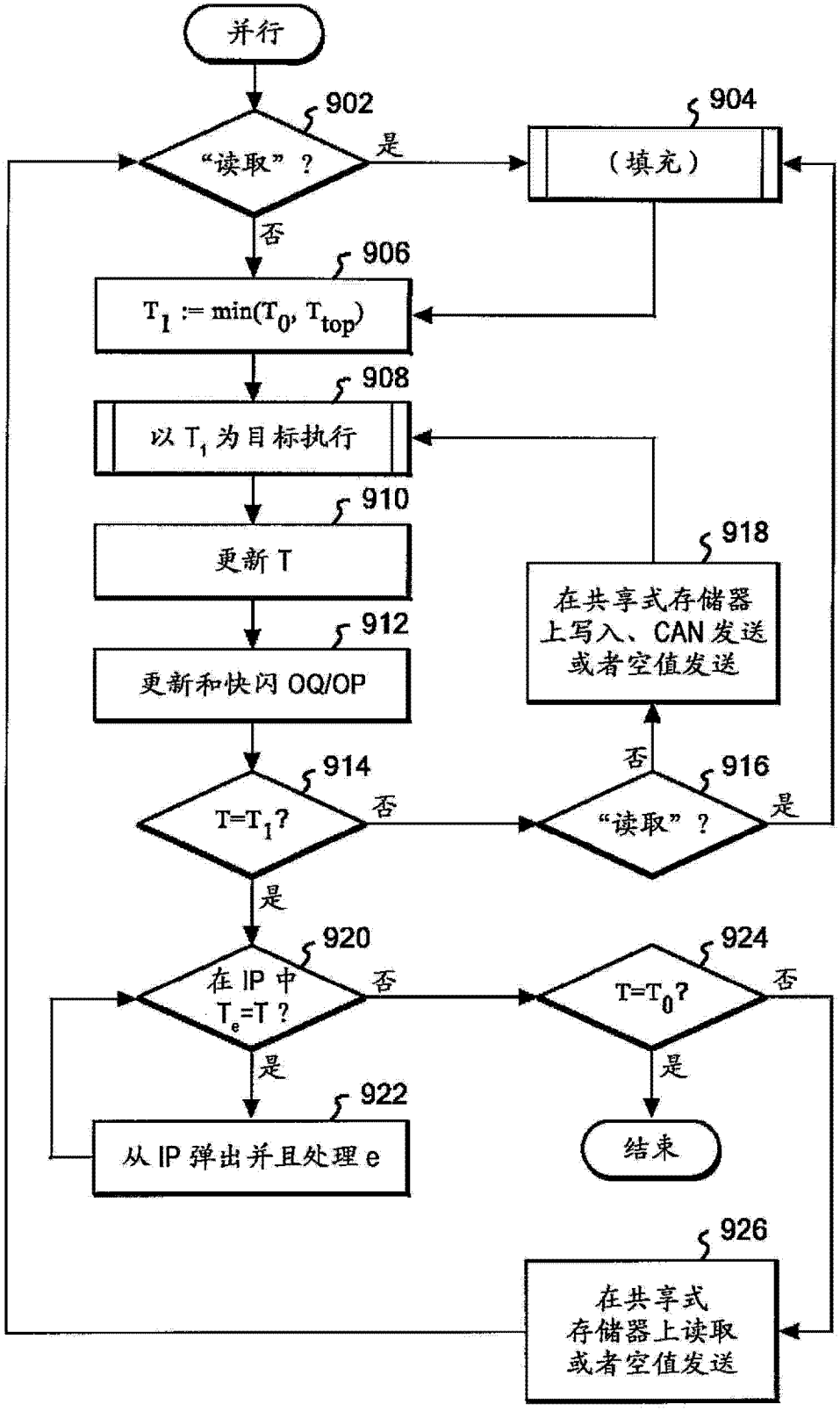

并行
902
“读取”？
是
904
（填充）
否
906
$T_1 := \min(T_0, T_{top})$
908
以 T_1 为目标执行
910
更新 T
912
更新和快闪 OQ/OP
914
$T=T_1$?
否
916
“读取”？
是
否
918
在共享式存储器
上写入、CAN 发送
或者空值发送
是
920
在 IP 中
$T_e=T$?
否
924
$T=T_0$?
否
是
922
从 IP 弹出并且处理 e
是
结束
926
在共享式
存储器上读取
或者空值发送

图 9

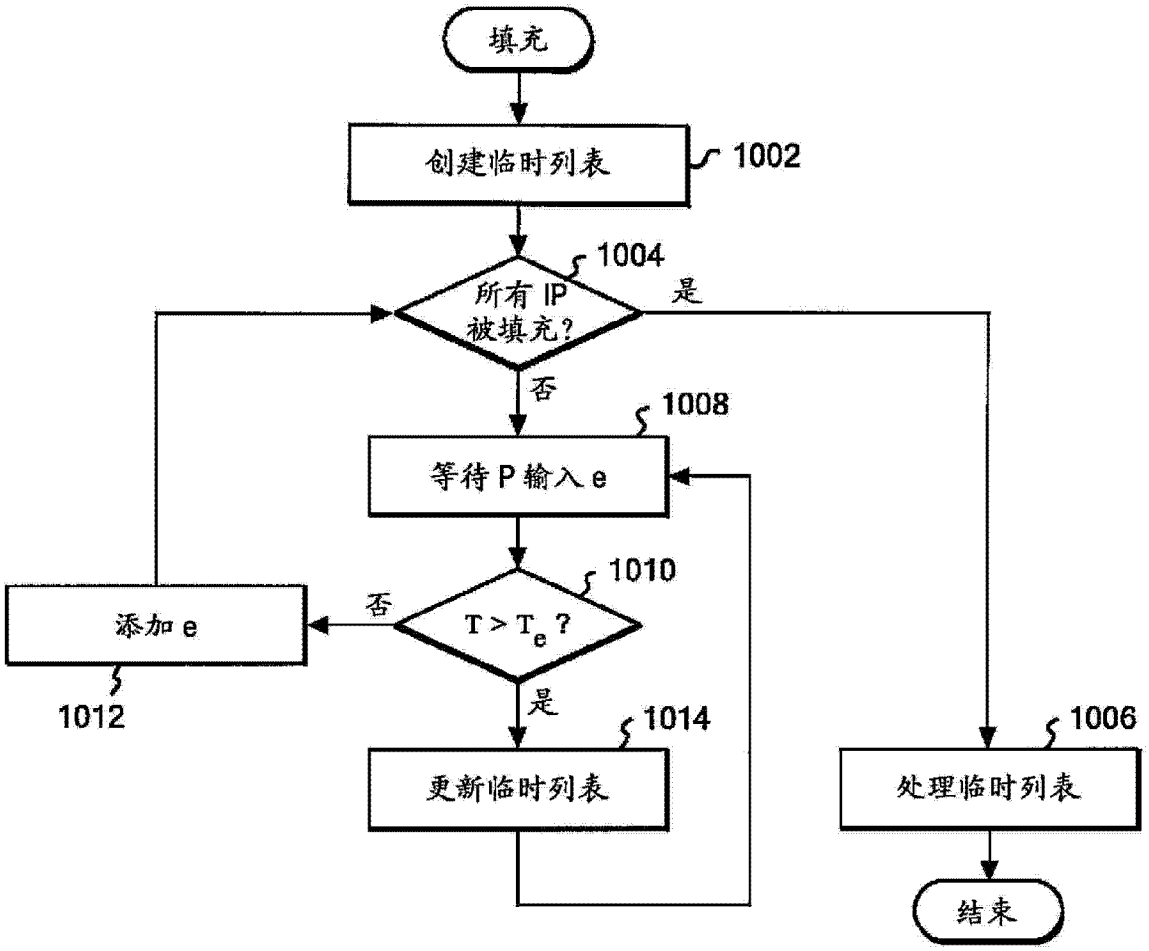

填充
创建临时列表
1002
1004
所有 IP 被填充?
是
否
1008
等待 P 输入 e
1010
$T > T_e$?
否
添加 e
1012
是
1014
更新临时列表
1006
处理临时列表
结束

图 10

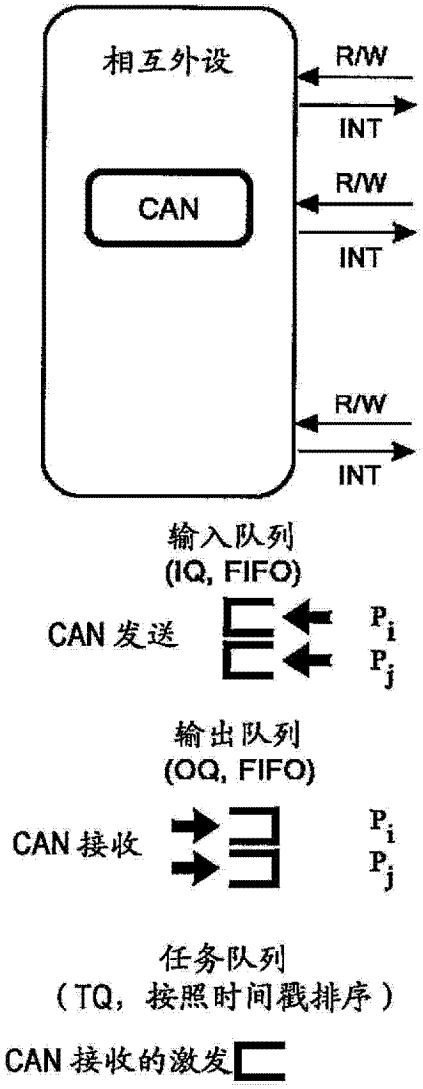

相互外设
CAN
R/W
INT
R/W
INT
R/W
INT
输入队列
(IQ, FIFO)
CAN 发送
P_i
P_j
输出队列
(OQ, FIFO)
CAN 接收
P_i
P_j
任务队列
（TQ，按照时间戳排序）
CAN 接收的激发

图 11

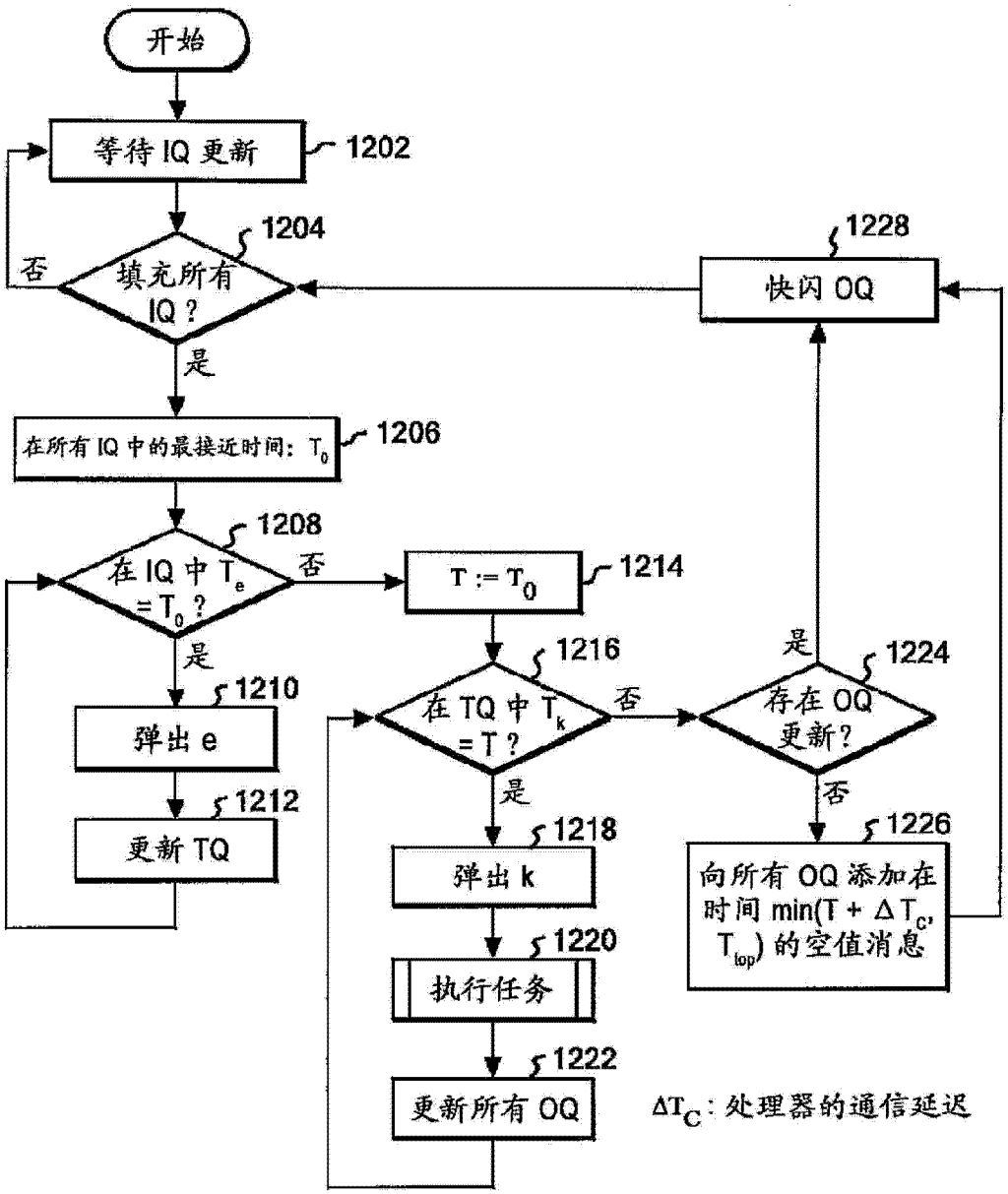

开始
等待 IQ 更新
1202
1204
填充所有 IQ ?
否
是
在所有 IQ 中的最接近时间: T_0
1206
1208
在 IQ 中 T_e = T_0 ?
否
是
1210
弹出 e
1212
更新 TQ
T := T_0
1214
1216
在 TQ 中 T_k = T ?
否
是
1218
弹出 k
1220
执行任务
1222
更新所有 OQ
1228
快闪 OQ
1224
存在 OQ 更新?
是
否
1226
向所有 OQ 添加在时间 min(T + ΔT_C, T_{top}) 的空值消息
ΔT_C : 处理器的通信延迟

图 12

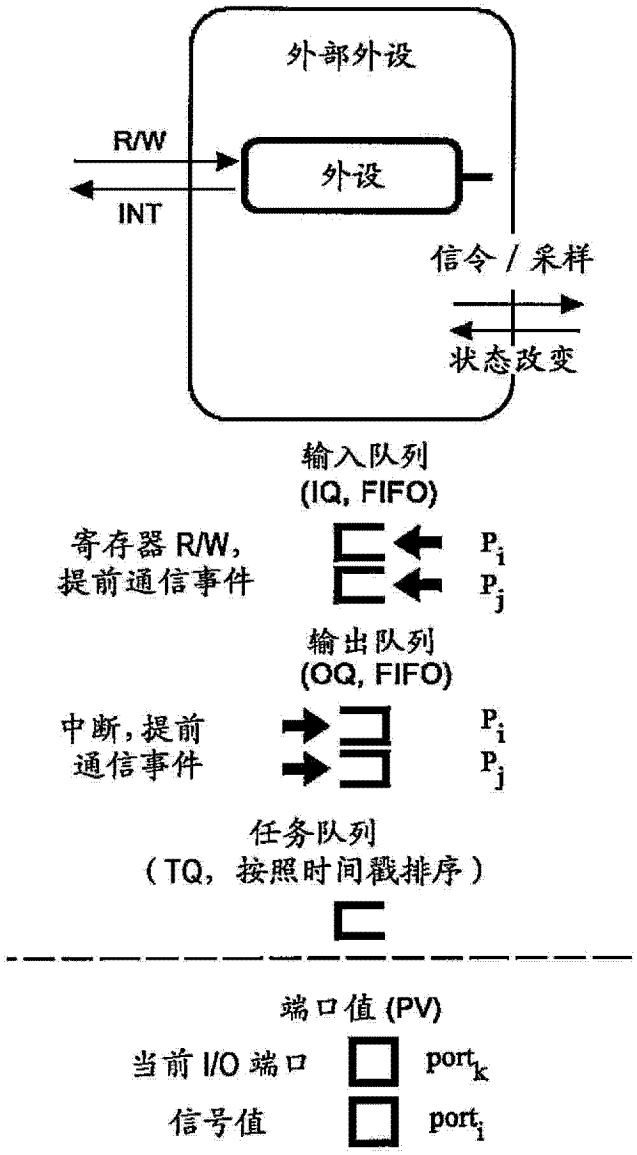

外部外设
R/W
INT
外设
信令 / 采样
状态改变
输入队列
(IQ, FIFO)
寄存器 R/W,
提前通信事件
P_i
P_j
输出队列
(OQ, FIFO)
中断,提前
通信事件
P_i
P_j
任务队列
(TQ, 按照时间戳排序)
端口值 (PV)
当前 I/O 端口
$port_k$
信号值
$port_i$

图 13

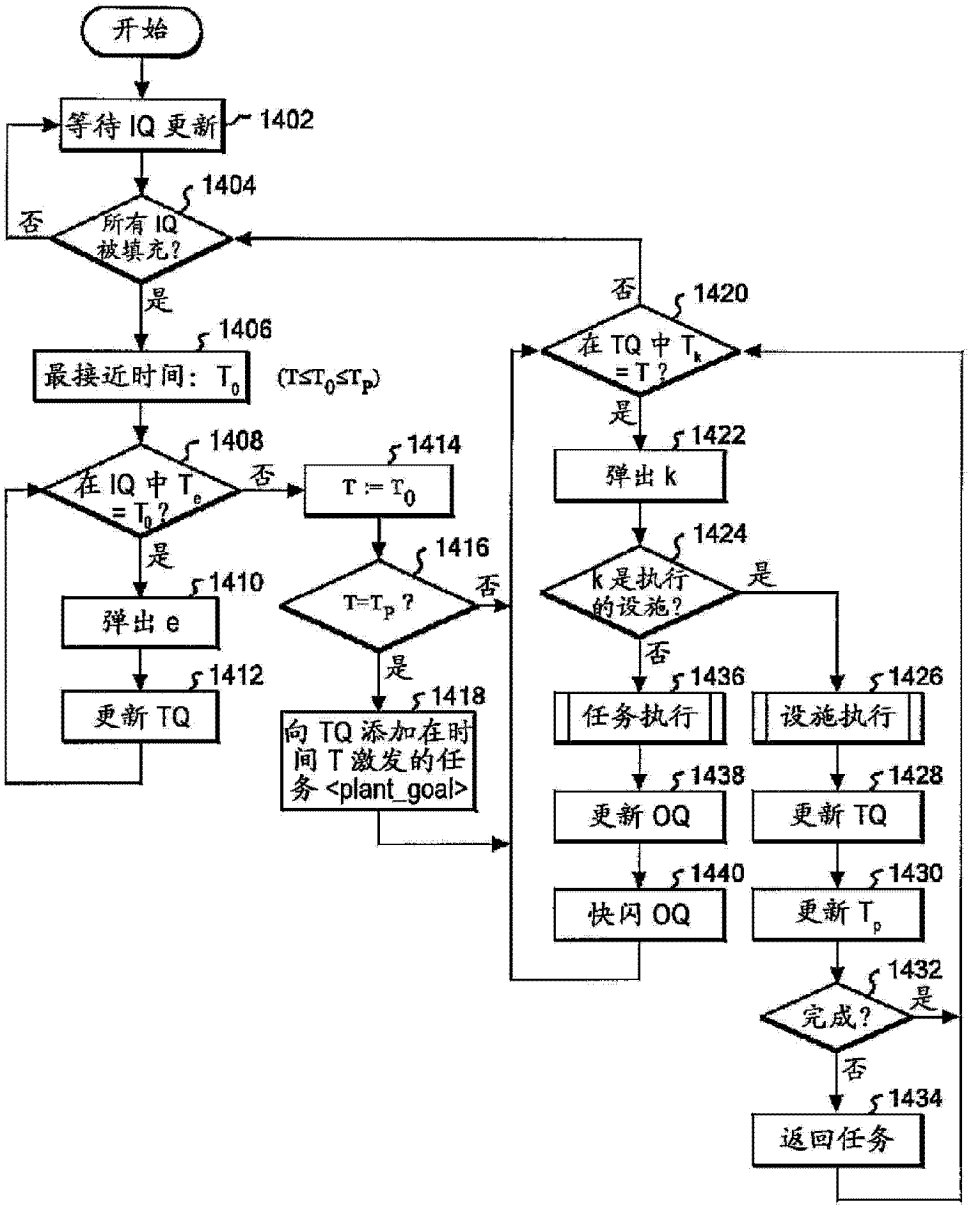

开始
等待 IQ 更新
1402
1404
所有 IQ 被填充?
否
是
1406
最接近时间: T_0
($T \le T_0 \le T_p$)
1408
在 IQ 中 $T_e = T_0$?
否
是
1410
弹出 e
1412
更新 TQ
1414
$T := T_0$
1416
$T = T_p$?
否
是
1418
向 TQ 添加在时间 T 激发的任务 <plant_goal>
1420
在 TQ 中 $T_k = T$?
否
是
1422
弹出 k
1424
k 是执行的设施?
是
否
1436
任务执行
1438
更新 OQ
1440
快闪 OQ
1426
设施执行
1428
更新 TQ
1430
更新 T_p
1432
完成?
是
否
1434
返回任务

图 14

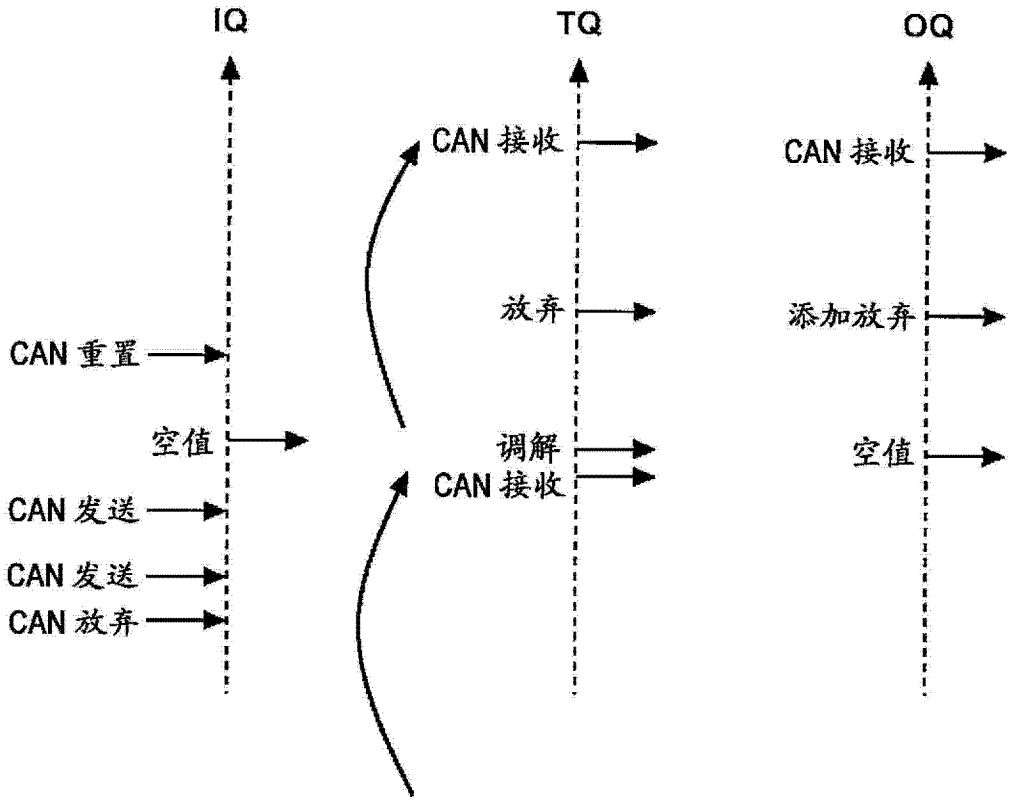

IQ
TQ
OQ
CAN 接收
CAN 接收
CAN 重置
放弃
添加放弃
空值
调解
空值
CAN 接收
CAN 发送
CAN 发送
CAN 放弃

图 15

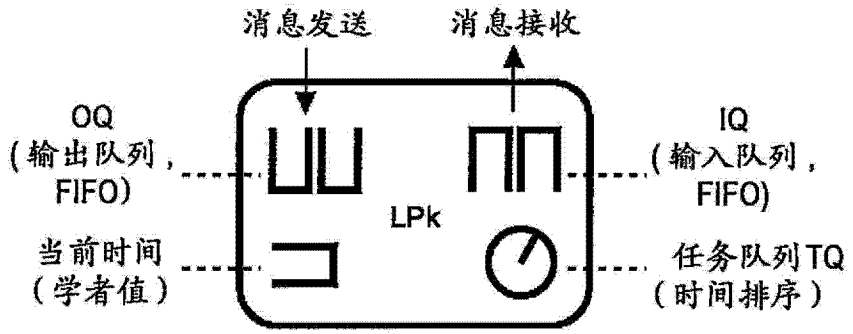

消息发送
消息接收
OQ
(输出队列，
FIFO)
IQ
(输入队列，
FIFO)
LPk
当前时间
(学者值)
任务队列TQ
(时间排序)

图 16

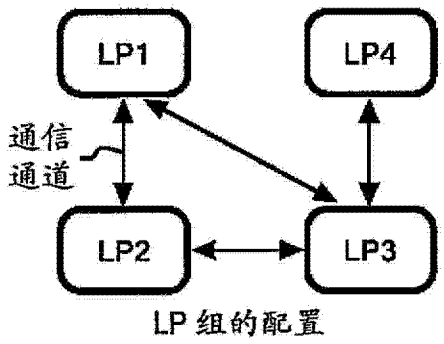

LP1
LP4
通信
通道
LP2
LP3
LP 组的配置

图 17

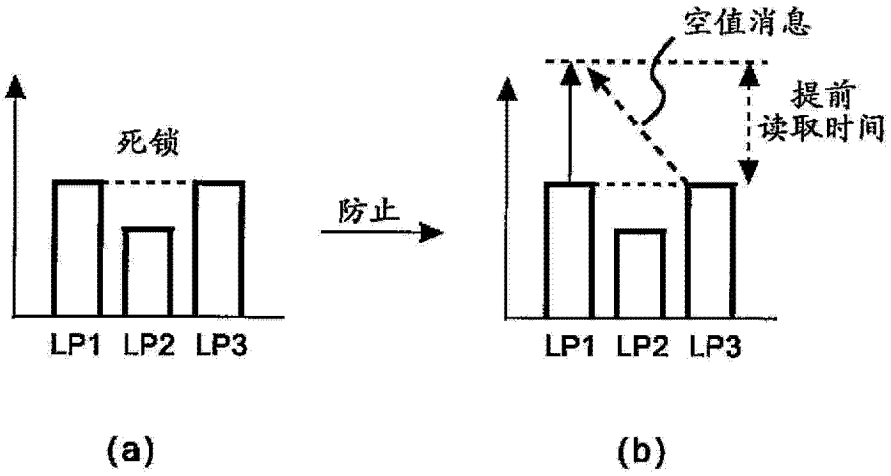

死锁
防止
空值消息
提前
读取时间
LP1 LP2 LP3
LP1 LP2 LP3
(a)
(b)

图 18

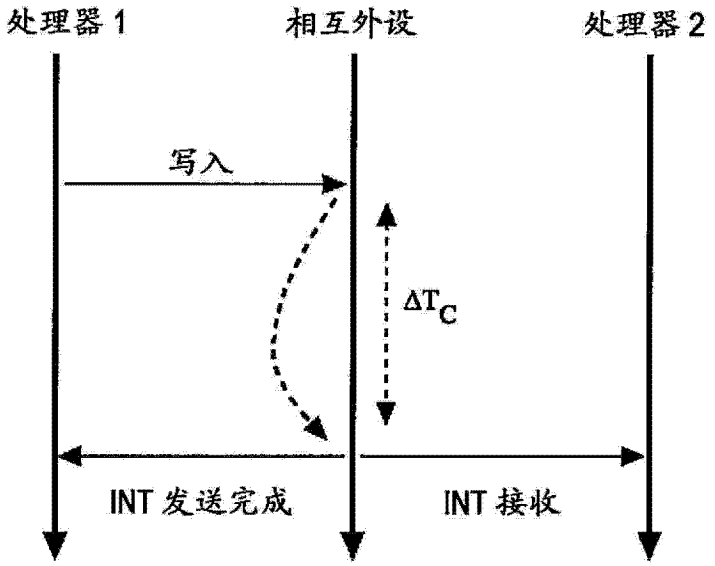

处理器 1
相互外设
处理器 2
写入
ΔT_C
INT 发送完成
INT 接收
处理器通信延迟 ΔT_C

图 19

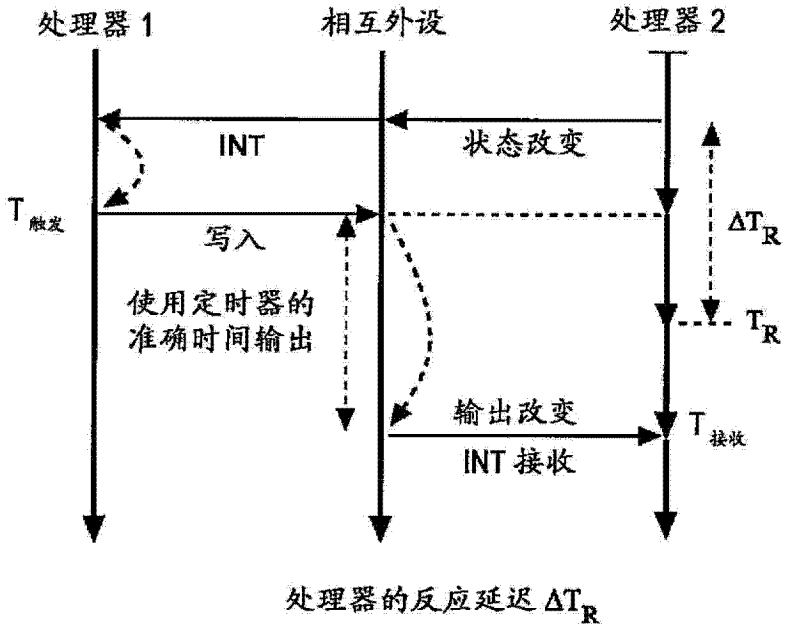

处理器 1
相互外设
处理器 2
INT
状态改变
$T_{触发}$
写入
ΔT_R
使用定时器的
准确时间输出
T_R
输出改变
$T_{接收}$
INT 接收
处理器的反应延迟 ΔT_R

图 20

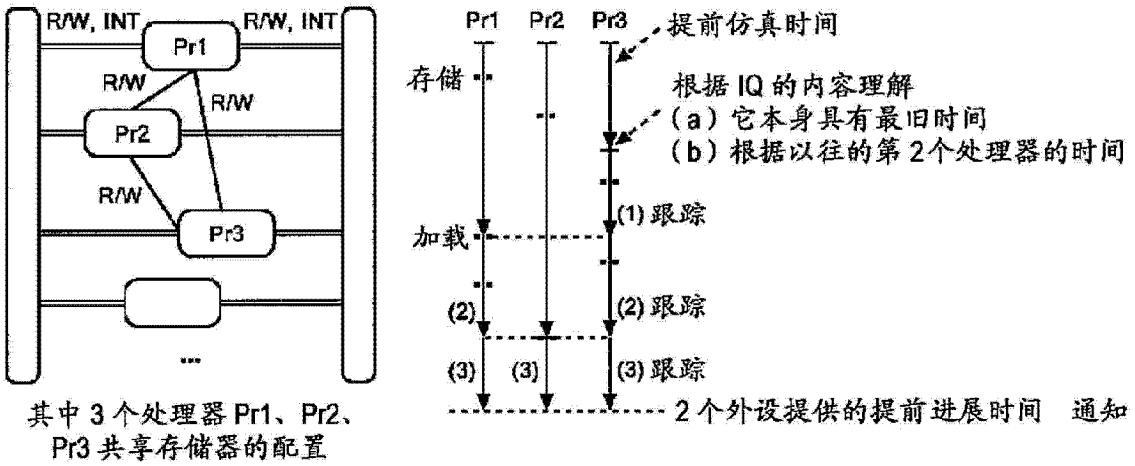

R/W, INT
R/W, INT
Pr1
R/W
R/W
Pr2
R/W
Pr3
...
其中 3 个处理器 Pr1、Pr2、Pr3 共享存储器的配置
Pr1
Pr2
Pr3
提前仿真时间
存储
根据 IQ 的内容理解
(a) 它本身具有最旧时间
(b) 根据以往的第 2 个处理器的时间
(1) 跟踪
加载
(2)
(2) 跟踪
(3)
(3)
(3) 跟踪
2 个外设提供的提前进展时间
通知

图 21

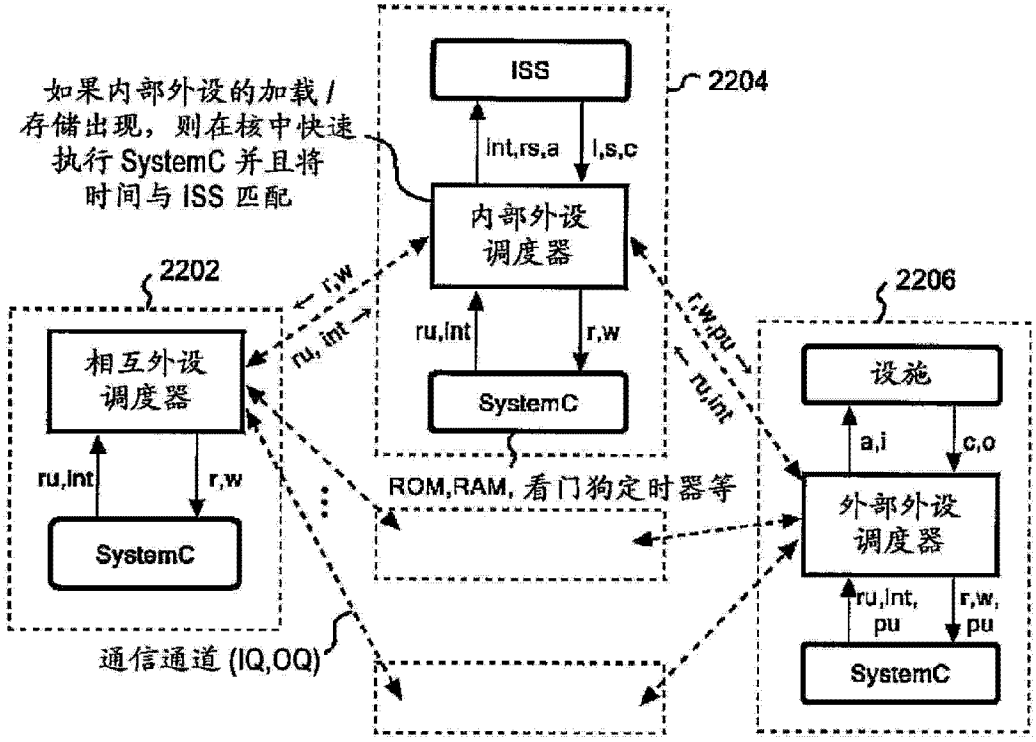

如果内部外设的加载 / 存储出现，则在核中快速执行 SystemC 并且将时间与 ISS 匹配
ISS
2204
Int,rs,a
I,s,c
内部外设调度器
2202
r,w
ru, int
ru,Int
r,w
相互外设调度器
ru,Int
r,w
SystemC
SystemC
ROM,RAM, 看门狗定时器等
r,w,pu
ru,Int
2206
设施
a,i
c,o
外部外设调度器
ru,Int, pu
r,w, pu
SystemC
通信通道 (IQ,OQ)

图 22

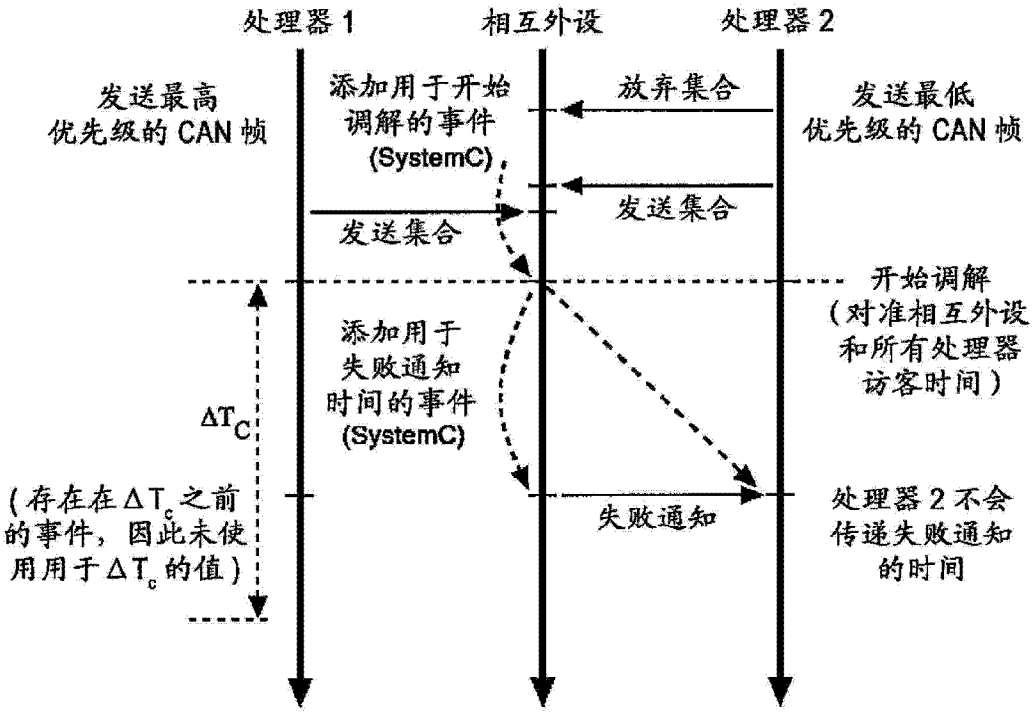

处理器 1
相互外设
处理器 2
发送最高
优先级的 CAN 帧
添加用于开始
调解的事件
(SystemC)
放弃集合
发送最低
优先级的 CAN 帧
发送集合
发送集合
开始调解
(对准相互外设
和所有处理器
访客时间)
添加用于
失败通知
时间的事件
(SystemC)
ΔT_C
(存在在 ΔT_c 之前
的事件，因此未使
用用于 ΔT_c 的值)
失败通知
处理器 2 不会
传递失败通知
的时间

图 23

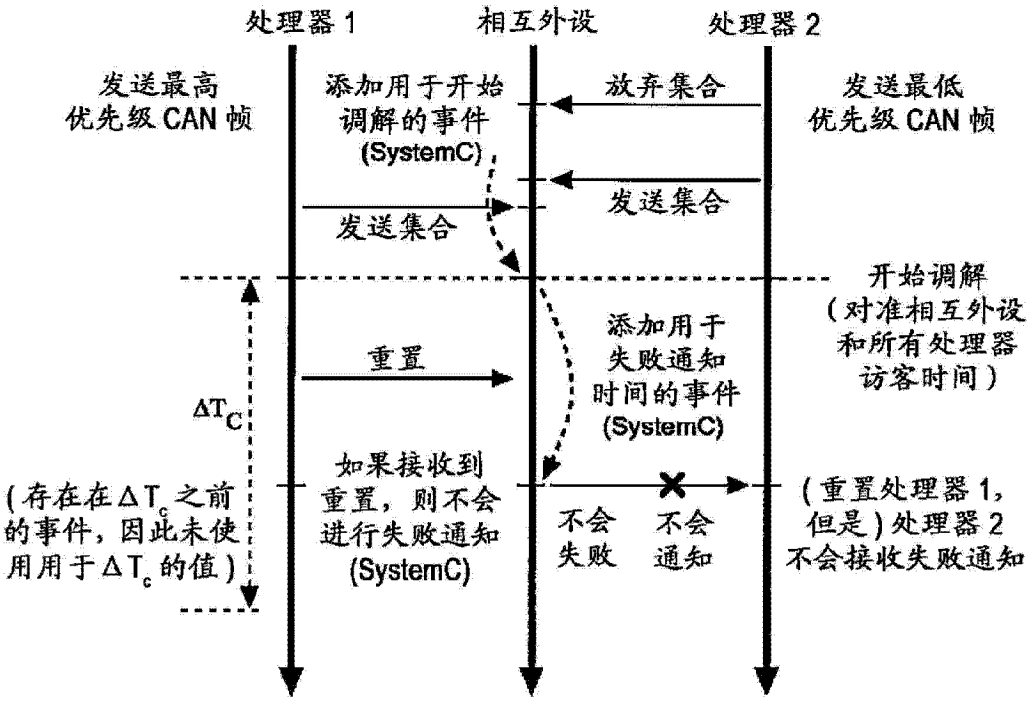

处理器 1
相互外设
处理器 2
发送最高
优先级 CAN 帧
添加用于开始
调解的事件
(SystemC)
放弃集合
发送最低
优先级 CAN 帧
发送集合
发送集合
开始调解
(对准相互外设
和所有处理器
访客时间)
添加用于
失败通知
时间的事件
(SystemC)
重置
ΔT_C
(存在在 ΔT_c 之前
的事件，因此未使
用用于 ΔT_c 的值)
如果接收到
重置，则不会
进行失败通知
(SystemC)
不会
失败
不会
通知
(重置处理器 1，
但是)处理器 2
不会接收失败通知

图 24

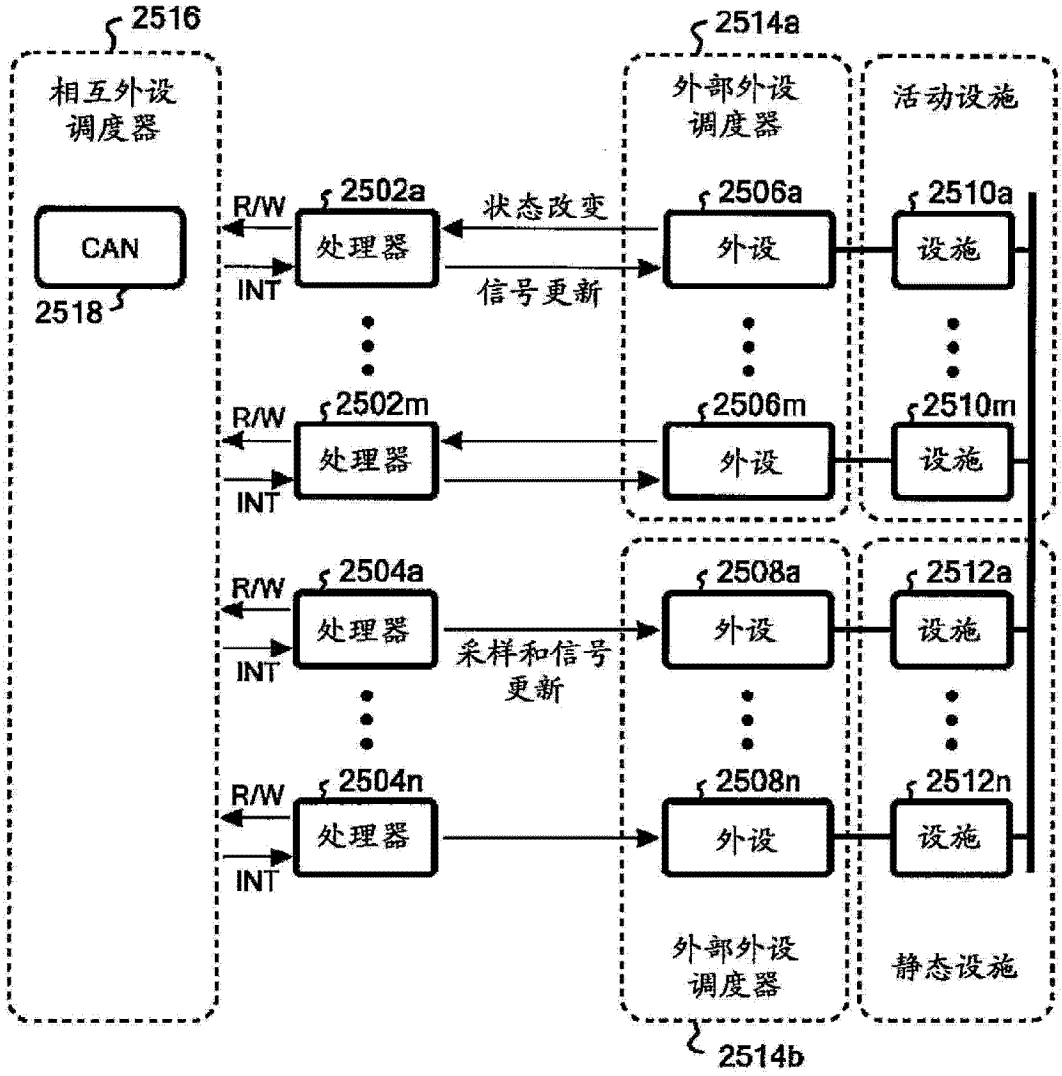

2516
相互外设
调度器
CAN
2518
R/W
INT
2502a
处理器
状态改变
信号更新
2506a
外设
2510a
设施
2502m
处理器
2506m
外设
2510m
设施
2514a
外部外设
调度器
活动设施
2504a
处理器
采样和信号
更新
2508a
外设
2512a
设施
2504n
处理器
2508n
外设
2512n
设施
外部外设
调度器
静态设施
2514b

图 25

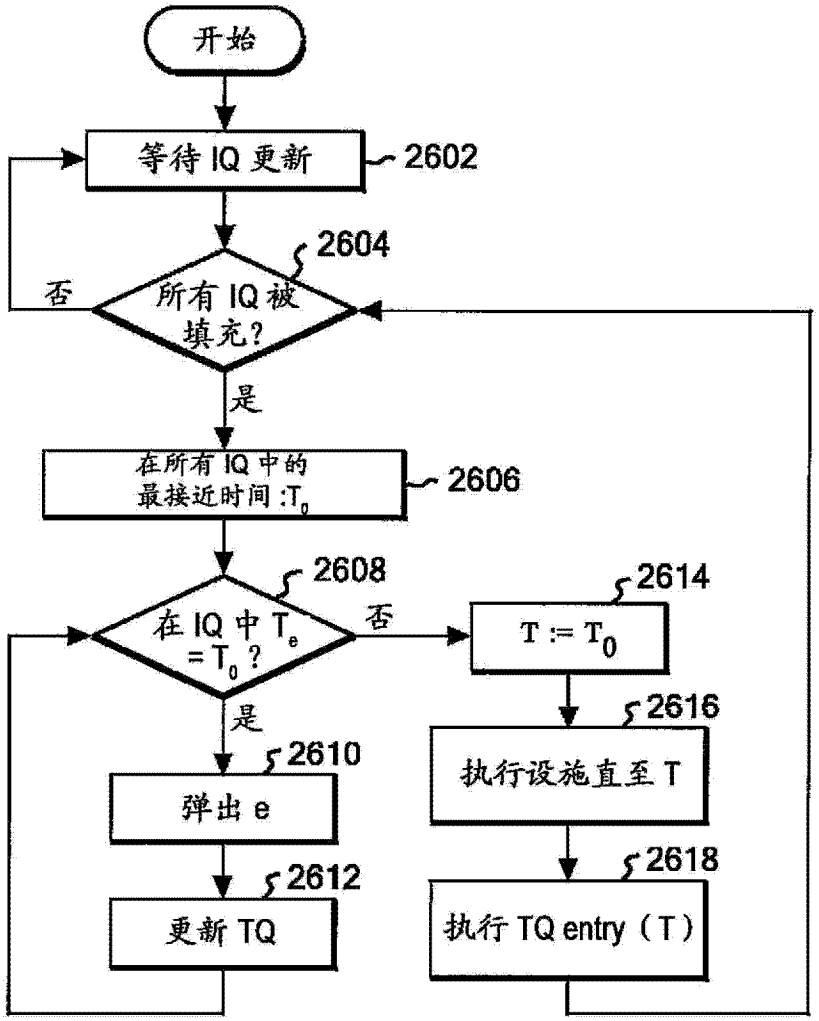

开始
等待 IQ 更新
2602
2604
所有 IQ 被填充?
否
是
在所有 IQ 中的最接近时间 :T_0
2606
2608
在 IQ 中 T_e = T_0 ?
否
是
2610
弹出 e
2612
更新 TQ
2614
T := T_0
2616
执行设施直至 T
2618
执行 TQ entry（T）

图 26