[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 1/32 (2006.01)
G06F 1/26 (2006.01)
G06F 9/38 (2006.01)

[12] 发明专利说明书

专利号 ZL 01803287.7

[45] 授权公告日 2008 年 11 月 26 日

[11] 授权公告号 CN 100437433C

[22] 申请日 2001.8.17 [21] 申请号 01803287.7

[30] 优先权

[32] 2000.8.23 [33] US [31] 09/645,468

[86] 国际申请 PCT/EP2001/009509 2001.8.17

[87] 国际公布 WO2002/017064 英 2002.2.28

[85] 进入国家阶段日期 2002.6.24

[73] 专利权人 NXP 股份有限公司

地址 荷兰艾恩德霍芬

[72] 发明人 D·恩沃伊 L·戈夫

B·塞克斯顿

[56] 参考文献

WO0002118A1 2000.1.30

CN1171159A 1998.1.21

US005953741A 1999.9.14

审查员 尹剑峰

[74] 专利代理机构 中科专利商标代理有限责任公司

代理人 王波波

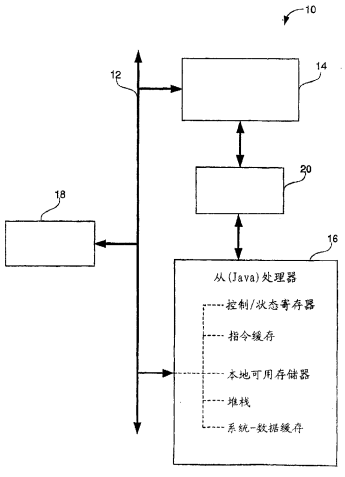
权利要求书 3 页 说明书 14 页 附图 5 页

[54] 发明名称

Java 加速器环境中功率管理的系统和方法

[57] 摘要

一种提供了基于 Java 加速器硬件能力管理的能力管理方法被公开。最初，响应初始化 Java 应用，由主处理器产生 Java 模式信号。然后，主处理器的能力减少；响应 Java 模式信号，Java 处理器的能力增加。在 Java 应用终止之后，由 Java 处理器产生 Java 完成信号，然后发送信号给系统交还控制给主处理器。



1. 一种为基于硬件的Java加速器提供功率管理的功率管理方法，所述功率管理方法包括以下操作:

响应Java应用的启动，由主处理器（14）提供Java模式信号（68);

响应所述Java模式信号，减少给所述主处理器（14）的功率;

响应所述Java模式信号，增加给Java处理器（16）的功率;

当所述Java应用的执行终止时，由所述Java处理器(16)产生一Java完成信号（84);

所述功率管理方法还包括以下操作:

响应所述Java完成信号（84)，增加给所述主处理器（68）的功率;和

响应所述Java完成信号（84)，减少给所述Java处理器（16）的功率;

其中使用斜坡电路处理来增加/减少给处理器的功率。

2. 权利要求1中所述的功率管理方法，其中功率包括电压和频率。

3. 权利要求1中所述的功率管理方法，其中所述斜坡电路处理包括如下操作:

（a）从目标功率寄存器获得目标功率值（104);

（b）比较（102）所述目标功率值（104）和来自当前功率寄存器的当前功率值（106);

（c）在所述当前功率值（106）小于所述目标功率值（104）时，增加所述当前功率值（106);

（d）重复操作（a）-（c)，直至所述当前功率值与所述目标功率值之间的差值处于一个预先定义的阈值之内为止。

4. 权利要求3中所述的功率管理方法，其中按照从功率偏移量表（108）中获取的量来增加所述当前功率值（106)。

5. 权利要求1中所述的功率管理方法，其中所述斜坡电路处理包括以下操作:

（a）从目标功率寄存器获得目标功率值（104）;

（b）比较（102）所述目标功率值（104）和来自当前功率寄存器的当前功率值（106）:

（c）在所述当前功率值（106）大于所述目标功率值（104）时，减少所述当前功率值（106）;

（d）重复操作（a）-（c），直至在所述当前功率值与所述目标功率值之间的差值处于一个预先定义的阈值之内为止。

6. 权利要求 5 中所述的功率管理方法，其中按照从功率偏移量表（108）中获取的量来减少所述当前功率值（104）。

7. 一种为基于硬件的 Java 加速器提供功率管理的功率管理系统，所述功率管理系统包括:

与功率产生电路连接的主处理器（14），所述主处理器（14）具有 Java 模式信号端口，能够响应 Java 应用的启动，提供 Java 模式信号（68）;

与功率产生电路连接的 Java 处理器（16），所述 Java 处理器（16）具有 Java 完成信号端口，能够当所述 Java 应用的执行终止时，提供 Java 完成信号（84）;

其中响应接收到 Java 模式信号（68），所述功率产生电路减少给主处理器（14）的功率并且增加给 Java 处理器（16）的功率；响应接收到 Java 完成信号（84），所述功率产生电路增加给主处理器（14）的功率并且减少给 Java 处理器（16）的功率;

连接于所述主处理器（14）的第一组可编程电压和频率产生器（62）;

连接于 Java 处理器（16）的第二组可编程电压和频率产生器（78）;

其中第一组可编程电压和频率产生器（62）能够调节所述主处理器（14）的电压和频率；第二组可编程电压和频率产生器（78）能够调节所述 Java 处理器（16）的电压和频率;

连接于第一组可编程电压和频率产生器（62）的主斜坡电路（64），所述主斜坡电路（64）能够把渐增的电压和频率变化提供给第一组可编程电压和频率产生器（62）;

连接于第二组可编程电压和频率产生器（78）的 Java 斜坡电路（80），所述 Java 斜坡电路（80）能够把渐增的电压和频率变化提供给第二组可

编程电压和频率产生器（78）;

由此所述主斜坡电路（64）和 Java 斜坡电路（80）就抑止了提供给主处理器（14）和 Java 处理器（16）的电压和频率变化。

8. 权利要求 7 中所述的功率管理系统，其中所述主斜坡电路（64）和所述 Java 斜坡电路（80）根据它们相关的处理器（14; 16）的目标功率值（104）和当前功率值之间的差值从功率偏移量表来确定要改变当前电压和频率的量。

9. 权利要求 8 中所述的功率管理系统，其中为所述主处理器（14）存储常态操作电压和频率值（58）和低功率电压和频率值（60），而为所述 Java 处理器（16）存储另一常态操作电压和频率值（72）和另一低功率电压和频率值（74）。

10. 权利要求 9 中所述的功率管理系统，其中所述主斜坡电路（64）提供在由所述主处理器（14）的所述常态操作电压和频率值（58）和低功率电压和频率值（60）所定义的范围值之内的渐增的电压和频率的变化;

而所述 Java 斜坡电路（80）提供在由所述 Java 处理器（16）的所述另一常态操作电压和频率值（72）和另一低功率电压和频率值（74）所定义的范围值之内的渐增的电压和频率变化。

Java加速器环境中功率管理的系统和方法

技术领域

本发明一般涉及硬件加速Java执行,更具体而言是包含主/从Java加速环境的功率管理。

背景技术

本申请涉及美国专利申请No.09/565,679(代理人卷号No.VTI1P333),提交于2000年5月4日,题目为:“一种具有Java堆栈、算术逻辑单元并使用相同多堆栈点及方法的智能子系统体系结构”,在此全文引入作为参考。

本申请同时还涉及美国专利申请No.09/670,496(代理人卷号No.PHILP337),题目为:“校验Java队列低上层边界的系统和方法”,在此全文引入作为参考。当今世界的计算机程序设计提供了许多高水平的编程语言。例如,Java就在相对较短的时间中获得了广泛的应用,这在很大程度上归功于无处不在的因特网的成功。Java的流行,至少在部分上是因为它平台的独立性、面向对象和有生命力的本性。而且,Java剔除了许多必须由包括存储器管理和跨平台接口的应用程序员所执行的乏味和易错的任务。这样,Java程序员能够更好的将精力集中于设计和函数问题。

使用Java处理器来执行Java应用,可以仅仅通过Java虚拟机(JVM)的形式由软件来象征性地实现。但是,因为伴随着执行软件中的程序,而要产生的问题是:JVM操作缓慢的问题使人烦扰。

传统上,两种方法被用来提高Java编译的性能,它们被称为本地Java执行和部分硬件编译Java指令。本地Java执行方法使用硬件来构建真正的Java处理器。但是,这种方法面对着一个严峻的缺点就是:它抹杀了Java“独立于机器之外”的概念,从而排斥了大量的软件,它们不再能够在这样的Java处理器上执行。

部分硬件编译Java指令方法使用硬件辅助来提高编译过程。这一结构通常被称为Java加速器。基本上,它在性能上近似于执行0等待状态存储器之外的汇编语言编译器。

一种 Java 加速器通过采用主/从结构来实现。其中：主处理器执行除 Java 指令之外的所有指令;从处理器仅仅执行 Java 指令。但是，使用该方法的传统系统充斥着主处理器和从处理器之间功率管理的问题。当两个处理器均用在小型、由电池供电、移动设备时这个问题尤为突出，那里的节能就尤为重要。

历史上，伴随着计算设备功率管理的问题来源于尝试着监测系统或系统内的各个部分的有意义的工作。通常，当监测设备的功率能够被关闭时，常规的功率管理系统协同空闲等待记时器，来监测输入和输出信号（I/O）。

在其它的传统功率管理系统中，处理器的接口频率是变化的，因此来减少计算机系统的功率消耗。此外，其它的传统功率管理系统使用被称为时钟遏制的方法来降低微处理器的功率消耗和计算机系统的功率消耗。在这种方法中，当微处理器被认为没有进行它的有意义工作时，微处理器的输入将改变，这就有效地通知微处理器停止使用内部时钟。通过改变微处理器的输入，微处理器能够减缓遏制的恢复，这样微处理器就不会被烧毁，而且同时减少计算机系统的功率消耗。尽管近期现有的功率管理技术判定程序已经通过使它们成为操作系统（OS）的中心而得到了提高，但是许多传统的功率管理判定仅仅是猜测。

尽管上述方法或许能够达到减少功率消耗的结果，但却没有一种方法综合地节减系统中每一个处理器的操作电压，从而能进一步减少系统的功率消耗。能够既控制主处理器又控制 Java 处理器的操作电压和操作频率将允许优化功率，平衡计算机系统的性能。

从发展的观点来说，需要提供一种提高 Java 加速器功率管理的系统和方法。功率管理系统能够控制主处理器和 Java 处理器操作电压和操作频率。而且，功率管理系统在操作状态改变时能够保证没有功率供应和数字电路应力发生；并确保在处理器之间存储器更新时，有充沛的时间可被利用。

发明内容

概括地说，本发明通过提供提高 Java 加速器的功率管理系统满足这些需要。本发明的功率管理系统能够控制主处理器和 Java 处理器操作电压和操作频率，因此允许优化计算机系统中的功率和性能。在一

种实施方案中，公开一种为基于 Java 加速器硬件提供功率管理的功率管理方法。最初，响应 Java 应用的启动，由主处理器提供 Java 模式信号。随后，响应 Java 模式信号，给该主处理器的功率被减少，给 Java 处理器的功率被增加。当 Java 应用的执行被终止时，从该 Java 处理器产生一个 Java 完成信号，从而发送信号给该系统将控制权交还给该主处理器。

在另一种实施方案中，公开一种为基于 Java 加速器硬件提供功率管理的功率管理方法。功率管理系统包括与功率产生电路连接的主处理器。主处理器包括能提供 Java 模式信号的 Java 模式信号端口。功率管理系统还包括与功率产生电路连接的 Java 处理器。Java 处理器包括能提供 Java 完成信号的 Java 完成信号端口。在使用中，响应接收 Java 模式信号，该功率产生电路减少给主处理器的功率，并且增加给 Java 处理器的功率。此外，响应接收 Java 完成信号，该功率产生电路增加给主处理器的功率，并减少给 Java 处理器的功率。

在本发明的进一步实施方案中公开一种抑制提供给处理器变化的斜坡(ramp)电路方法。斜坡电路方法从获得一个处理器所要设定的、代表希望的频率或电压的目标值开始。随后，目标值和当前值相比较而获得一个差值。与目标值类似，当前值是处理器当前操作的当前频率或电压。当差值在预定义的阈值之外时，调整当前值。这里的差值可以是值的范围或者是一个单一的值，例如：0。以上操作然后被重复，直至差值在预定义的阈值之内。

有利的是，本发明通过控制主处理器和 Java 处理器的操作频率和操作电压，允许优化计算机系统的功率和性能。而且，当频率和电压被控制时，电压在频率增加之前而增加，频率在电压减少之前被减少。这就保证了系统不会在当前电压下，以过快的频率运行，从而，节约了功率。

最后，对本领域的技术人员来说，本发明中的功率管理系统在操作状态改变时能够保证没有功率供应和数字电路应力发生；并确保在处理器之间存储器更新时，有充沛的时间可被利用。本发明的其它优点在下面的详细描述中将会是显而易见的，结合附图，本发明的原则将通过举例的方式予以说明。

附图说明

本发明，包含着更多的优点，最好通过参照下列描述、并结合附图予以更好的理解。附图中：

图 1 是说明结合本发明一种实施方案的，Java 硬件加速器系统方块图。

图 2 是说明结合本发明一种实施方案的，功率管理系统方块图。

图 3 是说明结合本发明一种实施方案的，可效仿的斜坡电路方块图。

图 4 是说明结合本发明一种实施方案的，为基于 Java 加速器硬件提供功率管理程序的流程图。

图 5 是说明结合本发明一种实施方案的，抑制提供给处理器变化的斜坡电路程序的流程图。

具体实施方式

本发明公开的是一种 Java 加速器功率管理系统。概括地说，本发明的功率管理系统在 Java 加速器中，通过 Java 程序被启动时减少主处理器的功率，增加 Java 处理器的功率来管理功率。当 Java 应用被终止时，该过程被逆转。

在下面的描述中，许多细节被提出以提供一个对本发明的透彻了解。然而，对本领域的技术人员来说，本发明可以无需特定部分或全部所有的细节而进行实践。另一方面，为了不必要地混淆本发明，众所周知的程序步骤没有进行详述。

图 1 是说明结合本发明一种实施方案的，Java 硬件加速器系统 10 的方块图。Java 硬件加速器系统 10 包括与主处理器 14、从（Java）处理器 16 和系统存储器 18 连接的系统总线 12。此外，Java 硬件加速器系统 10 还包括在主处理器 14 与 Java 处理器 16 之间连接的总线仲裁 20。

Java 硬件加速器系统 10 是双处理器系统，其中：Java 处理器 16 来执行很多 Java 指令，主处理器 14 来执行包括复杂 Java 指令的所有其它指令。如同 Java 是天生的“机器中的机器”典范，Java 处理器 16 是从处理器，作为“内部机器”。Java 处理器 16 被“外部机器”，即主处理器 14 所启动和控制。

主处理器 14 可以被嵌入任何处理器，包括当前的工业标准装置，例如：X86、ARM 和 MIPS 装置。Java 处理器 16 优选的是：包含指令和

数据缓存和内部 SRAM 的基于堆栈的处理器。而且，Java 处理器 16 优选的包括基于寄存器堆栈，正如在涉及到的美国专利申请 No. 09/565,679(代理人卷号 No. VTI1P333)，提交于 2000 年 5 月 4 日，题目为："一种具有 Java 堆栈、算术逻辑单元和使用相类似的多堆栈点及方法的智能子系统体系"，它在此全文引入作为参考。

一般来讲，主处理器 14 支持操作系统、非 Java 应用和复杂的 Java 指令。在一种实施方案中，额外的专门 ARM，或其等价物处理器（未示出）被用来处理复杂的 Java 指令。

Java 处理器 16 通常至少能够执行下列 Java 指令：加载、储存、计算、跳转、堆栈和分区。优选的，Java 处理器 16 内建的支持所有的计算指令，包括整数和浮点。如前面所述，主处理器 14 执行更复杂的指令，例如：程序调用。当复杂指令被解码时，Java 处理器 16 产生一个中断，所有的在数据缓存中标记行被写入系统存储器 18 中。然后，Java 处理器 16 停止执行。

优选的，Java 处理器 16 包括用于快速访问当前被执行的程序指令缓存。本地数据被用来支持留驻内部 SRAM 的程序，它是典型的单口，并且可被主处理器 14 和 Java 处理器 16 所访问。当 Java 处理器 16 执行时，主处理器 14 锁定所有的本地数据区和堆栈。此外，数据缓存包括快速访问的非本地变量，如：队列。

图 2 是说明结合本发明一种实施方案的，功率管理系统 50 的方块图。功率管理系统 50 包括主处理器端口 52 和 Java 处理器部分 54。主处理器端口 52 包括与一组主常态操作电压和频率寄存器 58 和一组主低功率电压和频率寄存器 60 连接的主开关寄存器 56。此外，功率管理系统 50 包括与主斜坡电路 64 和主处理器 14 连接的一组可编程电压和频率发生器 62。主处理器 14 包括用于在 Java 应用启动时提供 Java 模式信号 68 的 Java 模式信号端口。

Java 处理器部分 54 包括与一组 Java 常态操作电压和频率寄存器 72、一组 Java 低功率电压和频率寄存器 74 和空闲延时记时器连接的 Java 开关寄存器 70。此外，功率管理系统 50 包括与 Java 斜坡电路 80 和 Java 处理器 16 连接的一组可编程电压和频率发生器 78。Java 处理器 16 包括当 Java 应用完成时，用来提供 Java 完成信号 84 的 Java 完成信号端口。

每个编程电压和频率发生器 62 和 78 为由主斜坡电路 64 和 Java 斜坡电路 80 提供的频率变化提供真正的建立。在操作中，每个可编程控制器电压和频率发生器 62 和 78 用低频时钟信号锁定，允许合适的斜坡电路为每个时钟提供新的功率。

每个处理器 14 和 16 基本上有两种操作模式，常态模式和低功率操作模式。在常态操作模式中，主处理器 14 和 Java 处理器 16 在通常被看作主处理器 14 或 Java 处理器 16 嵌入处理器通常工作状态的电压和频率下操作。在低功率操作模式中，主处理器 14 或 Java 处理器 16 在低功率电压和频率下操作，它们低于主处理器 14 或 Java 处理器 16 嵌入的处理器通常工作状态。

当处理器 14/16 从运行在有意义工作、使用常态操作，切换至处理器 14/16 没有运行在有意义工作、低功率操作模式时，本发明提供了增强的功率节约。功率的节约通过当每个处理器 14/16 不被使用时，降低操作电压和操作频率，从而得到明显的增强。

开始时，主处理器的常态操作电压和常态操作频率值被储存在一组常态操作电压和频率寄存器 58 中。主处理器低功率操作电压值和主处理器的低频率值被储存在一组主低功率电压和频率寄存器 72 中。Java 处理器低功率操作电压和 Java 处理器低操作频率值则被储存在一组 Java 低电压和频率寄存器 74 中。

这样，几组主和 Java 常态操作电压和频率寄存器 58 和 72 各自定义主处理器 14 和 Java 处理器 16 的常态操作电压和频率。相类似，几组主和 Java 低功率操作电压和频率寄存器 60 和 74 各自定义主处理器 14 和 Java 处理器 16 的低功率操作电压和频率。

在通常的操作中，主处理器 14 根据它的常态操作模式，即被主操作电压和频率寄存器所定义的模式进行操作，同时 Java 处理器 16 根据被 Java 低功率和频率寄存器 74 它的低功率操作模式进行操作。如同先前讨论的那样，主处理器 14 通常执行所有非 Java 指令，以及 Java 处理器不能执行的复杂的 Java 指令，如：程序调用。当初化一个 Java 应用时，主处理器产生一个 Java 模式信号 68，通知功率管理系统 50 一个 Java 应用将要被 Java 处理器 16 所执行。主寄存器开关 56 和 Java 寄存器开关 70 接收 Java 模式信号 68，命令它们去切换主处理器 14 和 Java 处理器 16 操作模式。

当接收到 Java 模式信号 68，主寄存器开关 56 根据存储在主低功率电压和频率寄存器 60 中的值，为主处理器 14 设定主目标电压和频率值。与之相类似的，Java 寄存器开关 70 根据存储在 Java 常态操作电压和频率寄存器 72 中的值，为 Java 处理器 16 设定 Java 目标电压和频率值。

此时，主斜坡电路 64 使用主目标电压和频率值，逐步地减少主处理器当前的电压和频率到主目标电压和频率值。在相类似的方法中，Java 斜坡电路使用 Java 目标电压和频率值，逐步地提高 Java 处理器当前的电压和频率到 Java 目标电压和频率值。

主斜坡电路 64 和 Java 斜坡电路 80 各自提供小的电压和频率增量随着时间的变化而增加或减少，直到主处理器和 Java 处理器当前的电压和频率达到它们各自的目标电压和频率值。当每次电压和频率减少之后，主斜坡电路 64 与主处理器 14 连接，为一组可编程的电压和频率发生器 62 提供新的电压和频率减量值。该组可编程的电压和频率发生器 62 然后为主处理器 14 的电压和频率设定新的电压和频率减量值。

相类似的，当每次增加之后，Java 斜坡 80 电路与 Java 处理器 16 连接，为一组可编程的电压和频率发生器 78 提供新的电压和频率增量值。该编程的电压和频率发生器 78 然后为 Java 处理器 16 的电压和频率设定新的电压和频率增量值。在此方法中，当一个 Java 应用被启动时，功率在主处理器 14 中被减少，在 Java 处理器 16 中增加，其结果就是功率实质上在系统 50 中被恢复了。

然后，Java 处理器 16 开始执行 Java 应用。Java 处理器 16 将逐步地执行 Java 应用，直至其完成，或者直至遇到一个 Java 处理器 16 不能完成的复杂的 Java 指令。在此时，Java 处理器 16 产生一个 Java 完成信号 84 或者中断，它将被主寄存器开关 56 和 Java 寄存器开关 70 接收，通知电源管理系统 50，Java 应用的执行被终止了。

当接收到 Java 完成信号 84，主寄存器开关 56 根据存储在主常态操作电压和频率寄存器 58 中的值来设定主目标电压和频率值。相类似的，Java 寄存器开关 70 根据存储在 Java 低功率操作电压和频率寄存器 74 中的值，为 Java 处理器 16 设定 Java 目标电压和频率值。

在此时，主斜坡电路 64 使用主目标电压和频率值，逐步地增加主处理器 14 当前电压和频率到主目标电压和频率值。在相类似的方法

中，Java 斜坡电路 80 使用 Java 目标电压和频率值，逐步地减少 Java 处理器 14 当前电压和频率到 Java 目标电压和频率值。

正如上面所提及的，主斜坡电路 64 和 Java 斜坡电路 80 各自提供小的电压和频率增量随着时间的变化而增加或减少，直到主处理器和 Java 处理器当前电压和频率达到它们各自的目标电压和频率值。当每次电压和频率增加之后，主斜坡电路 64 与主处理器 14 连接，为一组可编程的电压和频率发生器 62 提供新的电压和频率增量值。该组可编程的电压和频率发生器 62 系列然后为主处理器 14 的电压和频率设定新的电压和频率增量值。

相类似的，当每次减少之后，Java 斜坡电路 80 与 Java 处理器 16 连接，为一组可编程的电压和频率发生器 78 提供新的电压和频率减量值。该组可编程的电压和频率发生器 78 然后为 Java 处理器 16 的电压和频率设定新的电压和频率增量值。这样，当 Java 应用的执行被终止时，功率在主处理器 14 中增加，在 Java 处理器 16 中被减少。此时，主处理器 14 重新获得系统的控制，继续执行计算机的指令。

在 Java 执行中，Java 处理器 16 在本地缓存中存储用于执行各种 Java 指令的信息。当 Java 执行被终止时，这个信息通过系统总线使得主处理器 14 来说是可以获取它。为了在 Java 处理器 16 被关闭来降低功率操作模式之前提供这样的信息，本发明使用一个空闲延迟计时器 76 连接在 Java 寄存器开关 70 和 Java 斜坡电路 80 之间完成此项任务。

空闲延迟计时器 76 通过设定 Java 目标电压和频率值到存储在 Java 低操作电压和频率寄存器 74 中低值，来延迟 Java 电压斜坡电路。延迟的长度是 Java 处理器 16 缓存大小的函数，它被设定为：在 Java 处理器 16 的功率被减少之前，缓存中的信息能同时被传输到系统存储器。要说明的是，空闲延迟计时器 76 仅仅在 Java 处理器 16 产生一个 Java 完成信号 84 之后才起作用。

图 3 是说明结合本发明一种实施方案的，可效仿的斜坡电路 100 的方块图。可仿效的斜坡电路 100 使用的是频率斜坡电路。但是，斜坡电路 100 同样也可以使用电压斜坡电路给寄存器提供电压值而不是频率值。

斜坡电路 100 包括与目标频率寄存器 104、当前频率寄存器 106

和偏移量表 108 连接的比较器 102。此外，斜坡电路 100 还包括与偏移量表 108、当前频率寄存器 106 连接的偏移量频率寄存器 110。最后，斜坡电路 100 包括一个频率状态机器 112，来提供对斜坡电路 100 的控制。应当指出的是，当前频率寄存器 106 同样与可编程频率发生器 114 连接。

当功率模式在常态和低功率模式之间进行切换时，电压和频率的变化使得主处理器和 Java 处理器优选地被抑制，以避免给功率供应和数字逻辑增加应力。

如同其后更加详尽的说明，斜坡电路 100 提供给处理器电压和频率一个小的增量变化，确保这些变化的正确的优先权。在一般情况下，当处理器的功率由低功率模式切换到常态模式时，电压将在频率增加前增加。与之相反，当处理器的功率由常态功率模式到低功率模式进行切换时，频率将在电压减少前减少。

在操作中，斜坡电路 100 由频率状态机器 112 提供相对低的时钟信号所封锁。如同前面所讨论的，斜坡电路 100 从一组常态操作或者低功率频率寄存器获得一个目标频率，这取决于当前操作模式和存储在频率寄存器 104 中的值。相应处理器当前操作频率存储在当前频率寄存器 106 中。

在各个时钟信号中，比较器 102 比较存储在目标频率寄存器 104 与存储在当前频率寄存器 106 中的值，而获得差值。这个差值是目标频率寄存器 104 与存储在当前频率寄存器 106 中的值的差。

如果差值在预定义的阈值之外，表格查找将使用差值和偏移表 108。表格查找的结果将被存储在频率偏移寄存器 110 中。预定义的阈值可以是一定范围的可接受的频率值、信号值或 0。设定阈值为 0 将使在差值不为 0 时，其结果在阈值之外。任何差值在阈值的范围之外都会使得其差值在阈值之外。

其后，存储在频率偏移寄存器 110 中的值将在存储在当前频率寄存器 106 中值基础上增加或减少，其结果存储在当前频率寄存器中。优选地，偏移表被设定为提供频率偏移，其结果是频率变化是平滑的线性频率增长。可选择的，频率偏移寄存器 110 可以被忽略，偏移量将直接从当前频率寄存器 114 和偏移表 108 中增加或减去。新的频率值存储在当前频率寄存器 106 中，然后由可编程频率发生器 114 编程，

来设定相应处理器到新的当前频率值。

斜坡电路 100 继续上述操作，直至差值达到预定义的阈值之内。此时，相应的处理器将在目标频率下操作，开始执行适当的指令。在其它的实施方案中，上述操作在阈值达到后将继续。在这个实施方案中，0 值将被存储在偏移寄存器 110 中，其结果是当前的频率值不会有太多的变化。当一个新的目标频率被存储在目标频率寄存器中时，斜坡电路 100 自动进行操作来提升频率到新的目标值。

如上面所提到的，应当指出，斜坡电路 100 可以作为电压斜坡电路来操作，即通过在斜坡电路 100 中改变频率值来改变电压值、将电路 100 连接到一个可编程电压发生器。

图 4 是说明结合本发明一种实施方案的，为基于 Java 加速器硬件提供功率管理程序 200 的流程图。在 202 的初始操作中，预处理操作被执行。预处理操作包括：初始化 Java 程序的计数和存储指针，其它的预处理操作对本领域的技术人员来讲中是浅显的。

在 Java 模式操作 204 中，响应启动 Java 应用，由主处理器提供一个 Java 模式信号。在常态的操作中，主处理器根据它的主常态操作电压和频率寄存器所定义的操作模式来进行操作；而 Java 处理器则根据由 Java 低功率操作电压和频率寄存器所定义的，低功率操作模式来进行操作。主处理器执行所有非 Java 指令，包括 Java 处理器不能执行的复杂 Java 指令，如程序调用。启动 Java 应用时，主处理器产生一个 Java 模式信号，通知功率管理系统，Java 应用将被 Java 处理器所执行。然后，主寄存器开关和 Java 寄存器开关接收 Java 模式信号，命令它们来在主和 Java 处理器间切换操作模式。

接下来，在主功率降低操作 206 中，响应 Java 模式信号，主处理器的功率被减少。接收到 Java 模式信号，主寄存器开关参照存储在主低功率电压和频率寄存器中的值来设定主处理器的主目标电压和频率值。然后，主斜坡电路参照存储在主低功率电压和频率寄存器中的值，使用主目标电压和频率值来逐步地减少当前的电压和频率至主目标电压和频率值。

如同前面所讨论的那样，主斜坡电路提供小的电压和频率增量随着时间的变化来增加或减少，直到当前的电压和频率达到它们各自的目标电压和频率值。当每次电压和频率减少之后，与主处理器连接的

主斜坡电路为该组可编程的电压和频率发生器提供新的电压和频率减量值。该组可编程的电压和频率发生器然后为主处理器的电压和频率设定新的电压和频率减量值。

在 Java 功率增加操作 208 中，响应 Java 模式信号，Java 处理器的功率增加。当接收到 Java 模式信号，Java 寄存器参照存储在 Java 常态电压和频率寄存器中的值为 Java 处理器开关设定 Java 目标电压和频率值。Java 斜坡电路则使用 Java 目标电压和频率值逐步增加 Java 处理器的当前电压和频率到相应的 Java 目标电压和频率值。

当每次电压和频率增加之后，与 Java 处理器连接的 Java 斜坡电路为该组可编程的电压和频率发生器提供新的电压和频率增量值。该组可编程的电压和频率发生器然后为 Java 处理器的电压和频率设定新的电压和频率增量值。当达到它的通常操作电压和频率时，Java 处理器开始执行 Java 应用，并且逐步地继续执行 Java 应用，直至其完成，或者直至遇到一个 Java 处理器不能完成的复杂的 Java 指令。

在 Java 完成信号操作 210 中，当 Java 应用执行被终止时，Java 处理器将产生一个 Java 完成信号。此时，Java 处理器产生一个由主寄存器开关和 Java 寄存器开关接收的 Java 完成信号或中断。Java 完成信号通知功率管理系统，Java 应用的执行被终止了。

然后，响应 Java 完成信号，在 Java 完成操作 212 中，主处理器的功率被增加，Java 处理器的功率被减少。接收到 Java 完成信号，主寄存开关参照由主常态操作电压和频率寄存器的值设定主目标电压和频率值。相类似的，Java 寄存器开关参照由 Java 低操作电压和频率寄存器的值设定 Java 目标电压和频率值。

此时，主斜坡电路使用主目标电压和频率值，逐步地增加主处理器的当前电压和频率值至主目标电压和频率值。在相类似的方法中，Java 斜坡电路使用 Java 目标电压和频率值，逐步地减小 Java 处理器当前电压和频率到 Java 目标电压和频率值。

正如上面所提及的，主斜坡电路和 Java 斜坡电路各自提供小的电压和频率增量随着时间的变化而增加或减少，直到主处理器和 Java 处理器当前的电压和频率达到它们各自的目标电压和频率值。当每次电压和频率增加之后，主斜坡电路与主处理器连接，为一组可编程的电压和频率发生器提供新的电压和频率增量值。该组可编程的电压和

频率发生器然后为主处理器的电压和频率设定新的电压和频率增量值。

相类似的，当每次减少之后，Java 斜坡电路与 Java 处理器连接，为一组可编程的电压和频率发生器提供新的电压和频率减量值。该组可编程的电压和频率发生器然后为 Java 处理器的电压和频率设定新的电压和频率减量值。在此方法中，当 Java 应用的执行被终止时，功率在主处理器中被增加，在 Java 处理器中被减少。此时，主处理器重新获得系统的控制，继续执行计算机的指令。

当 Java 执行被终止时，存储在本地存储器缓冲区中的信息通过系统总线使得主处理器可以获取它。为了在 Java 处理器被关闭来降低功率操作模式之前提供这样的信息，本发明使用一个空闲延迟记时器连接在 Java 寄存器开关和 Java 斜坡电路之间协同完成此项任务。

空闲延迟记时器通过设定 Java 目标电压和频率值到由 Java 低操作电压和频率寄存器中的低值，来延迟 Java 电压斜坡电路。延迟的长度是 Java 处理器缓存大小的函数，它被设定为：在 Java 处理器的功率被减少之前，缓存中的信息被传输到系统存储器中去。要说明的是，空闲延迟记时器被优选地设为：仅仅在 Java 处理器产生一个 Java 完成信号之后才起作用。

后处理操作然后在操作 214 中被执行。后处理操作包括：Java 方法调用的执行，其它的后处理操作对本领域的技术人员来说是浅显的。有利地是，既能控制主处理器和 Java 处理器的操作频率又能控制其操作电压将在计算机系统中优化功率和性能。

图 5 是说明结合本发明一种实施方案的，抑制提供给处理器变化的斜坡电路程序 300 的流程图。尽管斜坡电路程序 300 被描述为一个电压斜坡电路程序，斜坡电路程序同样可以被用于提升处理器频率或电压。通过用频率值代替电压值，斜坡电路程序 300 可以被用作频率斜坡电路程序。在初始操作 302 中，预处理被执行。预处理操作包括：初始化常态操作、低功率电压和频率寄存器，其它的预处理操作对本领域的技术人员来说是浅显的。

在目标值操作 304 中，目标值从电压和频率寄存器中被获取。目标电压是从常态操作或者低电压寄存器中获取得，它取决于当前功率操作模式，并被存储在目标电压寄存器中。相应处理器的当前操作电

压值存储在当前电压寄存器中。

接下来，在比较器 306 中，目标值和当前值进行比较，而获得差值。在每个时钟信号中，比较器比较存储在目标频率寄存器和存储在当前频率寄存器中的值，而获得差值。这个差值是存储在目标频率寄存器和存储在当前频率寄存器中的值的差。

随后，一个是否该差值在预定义的阈值之外的判定将在操作 308 中被执行。预定的阈值可以是一定范围的可接受的频率值、信号值或 0。设定阈值为 0 将使得当差值不为 0 时，其结果在阈值之外。设定阈值为单一值表示，则该值得范围在 0 和该单一值之间。一旦该差值在阈值的范围之外，均判断差值超出阈值。如果差值超出了预定义的阈值，斜坡电路进程 300 继续执行查表操作 310。否则，斜坡电路操作 300 将在 312 操作中完成。

差值在查表操作 310 中进行偏移表查表操作。偏移表提供包括多个电压值。通过在偏移表中查找差值，可以获得一个偏移电压值。偏移表中的偏移值然后被储存在偏移电压寄存器中。优选地，偏移表被设定为提供电压偏移，其结果是电压变化是平滑的线性电压增长。

在调整操作 314 中，当前的值基于在操作 310 中获得的偏移电压被调整。存储在偏移电压寄存器中的值被从存储在当前电压寄存器中增加或者减去，结果存储在当前电压寄存器中。可选择的，电压偏移寄存器可以被忽略，偏移量将直接从当前电压寄存器和偏移表中增加或减去。

在电压编程操作 316 中，新的当前电压值被提供给可编程的电压发生器。存储在当前电压寄存器中的新的电压值，被用来编写可编程电压发生器，设定所涉及处理器至新的当前电压值。斜坡电路程序 300 然后和操作 304 进行其它比较，这里的当前电压寄存器包括新的调整过的当前电压值。

在操作 312 中，当前的电压值包含在预定的阈值内，斜坡电路程序 300 就完成了。此时，所涉及的处理器将在目标电压下操作，可以开始执行合适的指令。

在其它的实施方案中，当预定的阈值达到之后，斜坡电路进程 300 继续和操作 304 的其它比较。在这个实施方案中，0 值被存储在偏移寄存器中，其结果是当前的电压不会发生任何的变化。当新的目标电

压被储存在目标电压寄存器中时，斜坡电路自动地继续操作，提升当前的电压至新的目标电压。

斜坡电路 300 确保电压和频率变化的正确优先权。在一般情况下，当处理器的功率由低功率模式到常态模式进行切换时，电压将在频率增加前增加。与之相反，当处理器的功率由常态功率模式到低功率模式进行切换时，频率将在电压减小前减小。

有利的是，斜坡电路 300 在操作状态改变时能够保证没有功率供应和数字电路应力发生；并确保在处理器之间存储器更新时，有充沛的时间可被利用。

尽管前面发明细节的详细描述其目在于理解的澄清，然而显然会在附属的权利要求范围中会有一定的变化和调整。相应的，本实施方案可以被看作是说明而不是限制，并且发明也并不局限于给定的细节，但是，范围内的调整和其等价物或许会包含在附属的权利要求中。

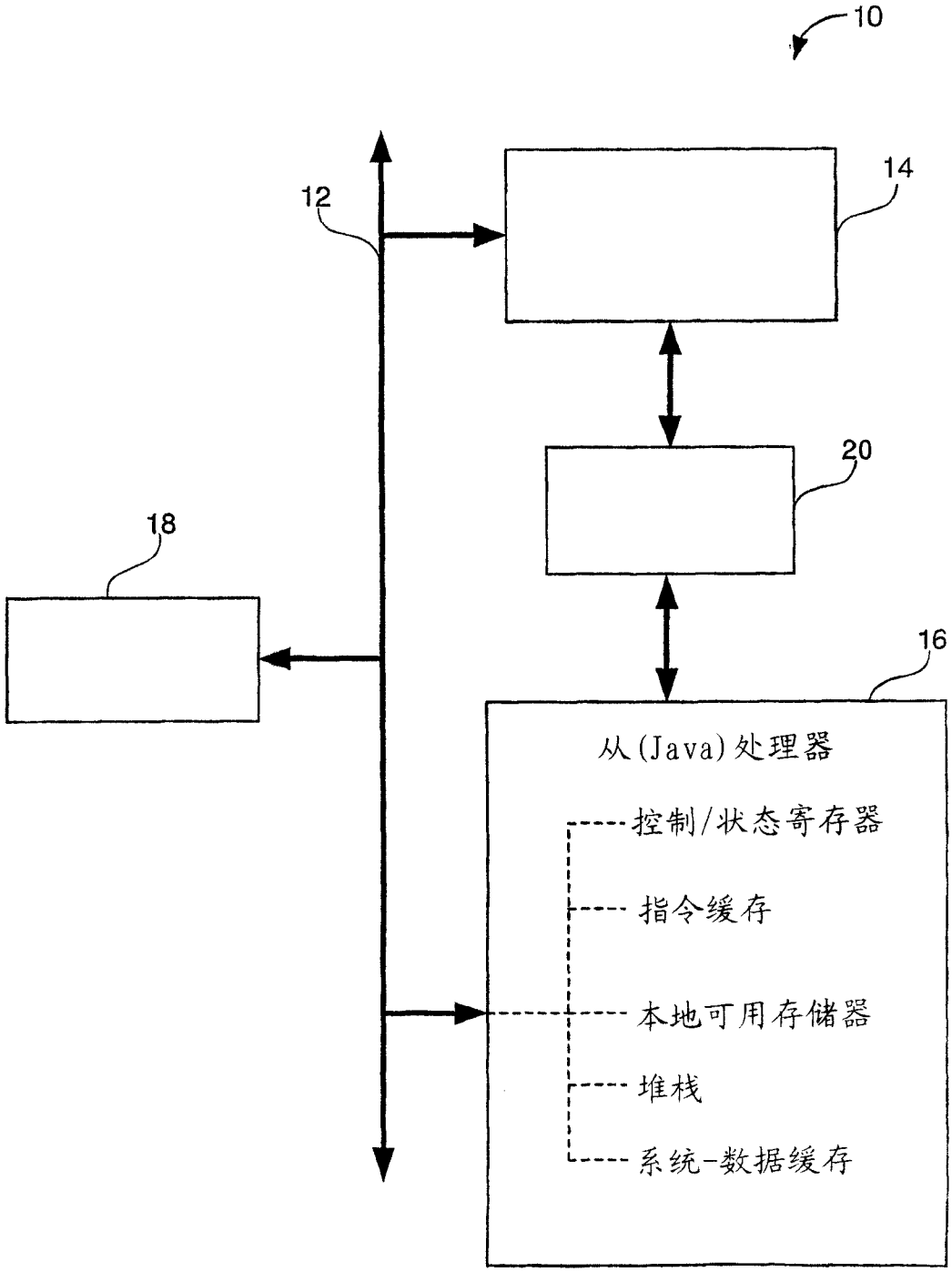


图 1

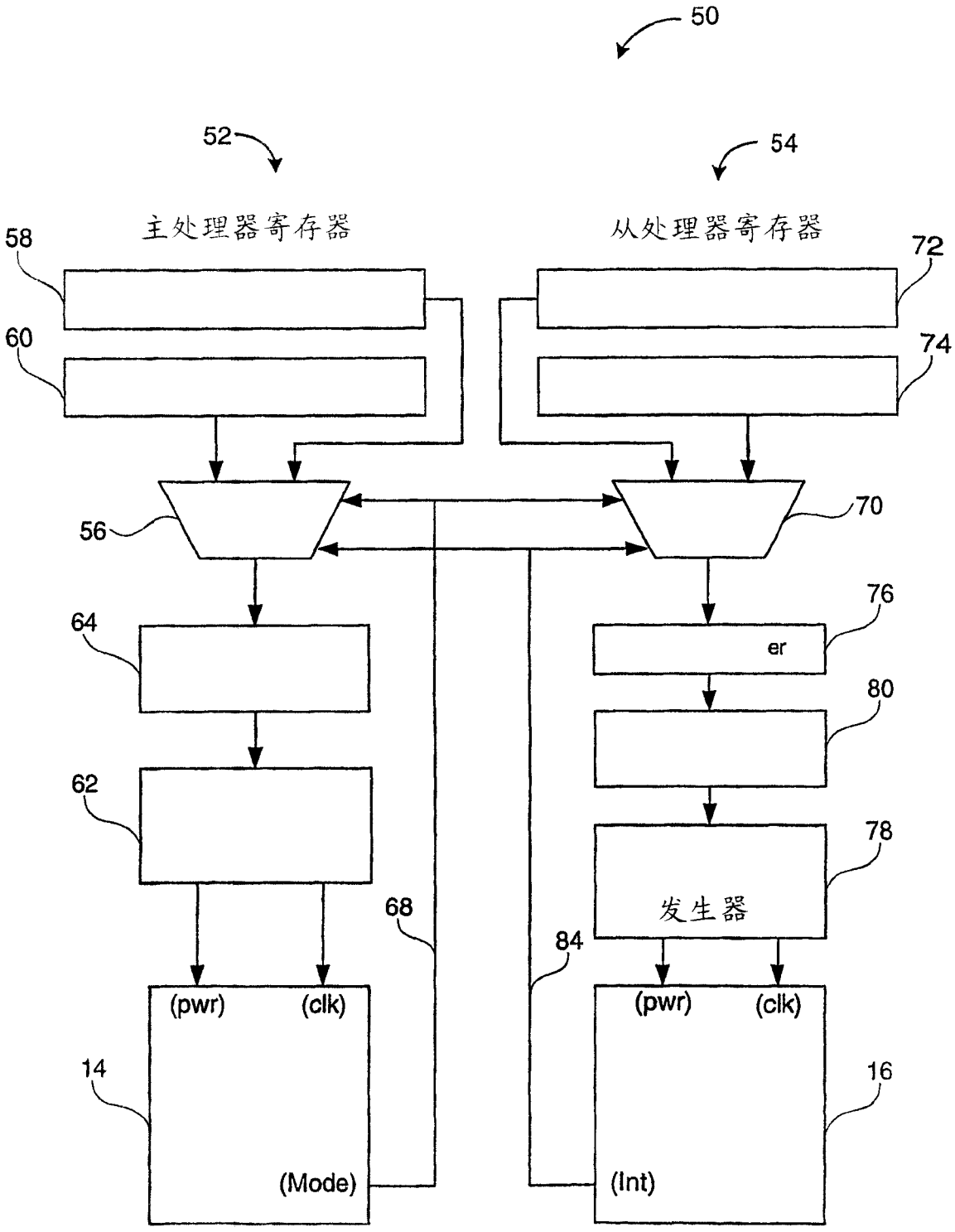

50
52
54
主处理器寄存器
从处理器寄存器
58
60
72
74
56
70
64
76
er
80
62
78
发生器
68
84
(pwr)
(clk)
(pwr)
(clk)
14
16
(Mode)
(Int)

图 2

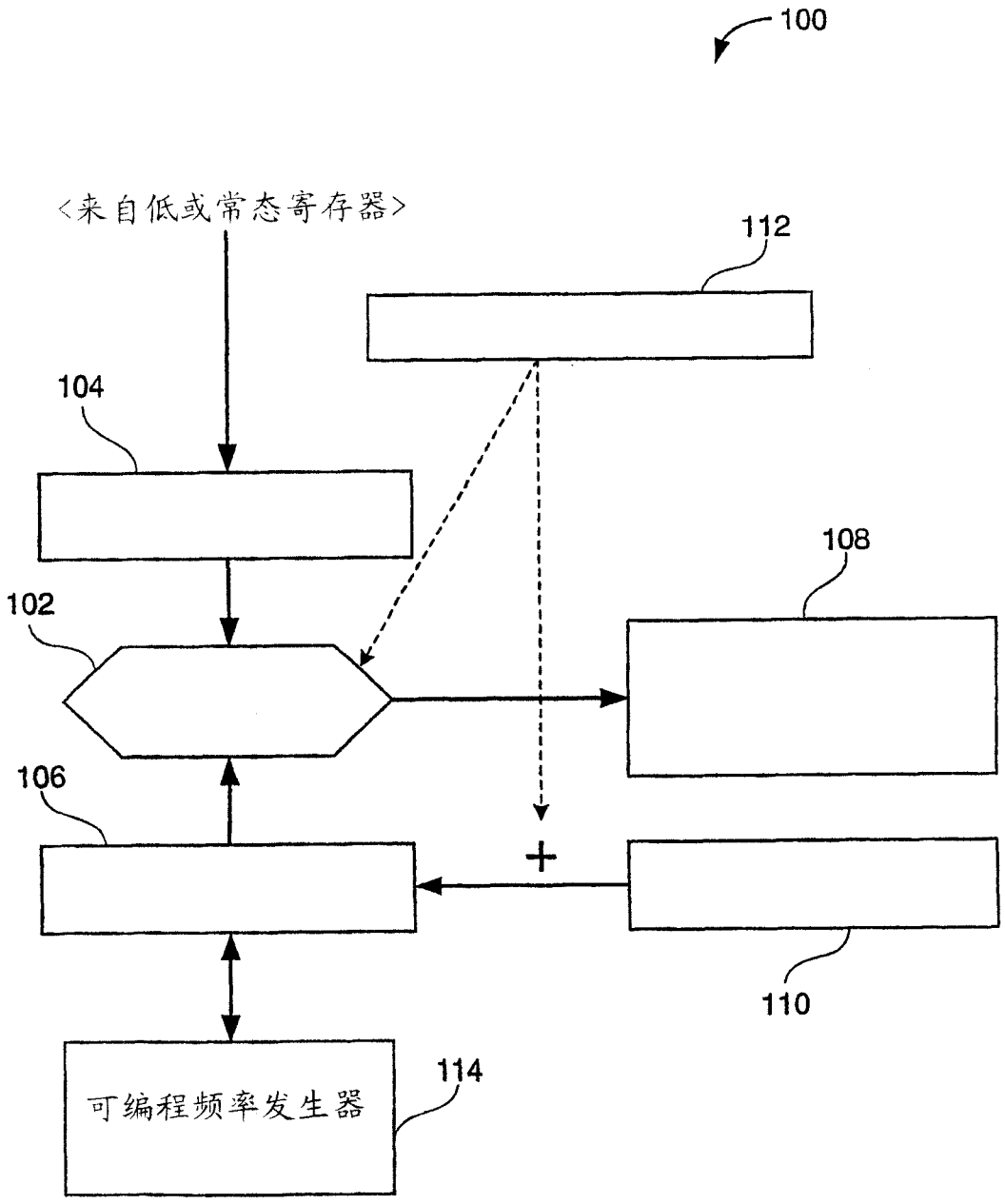


图 3

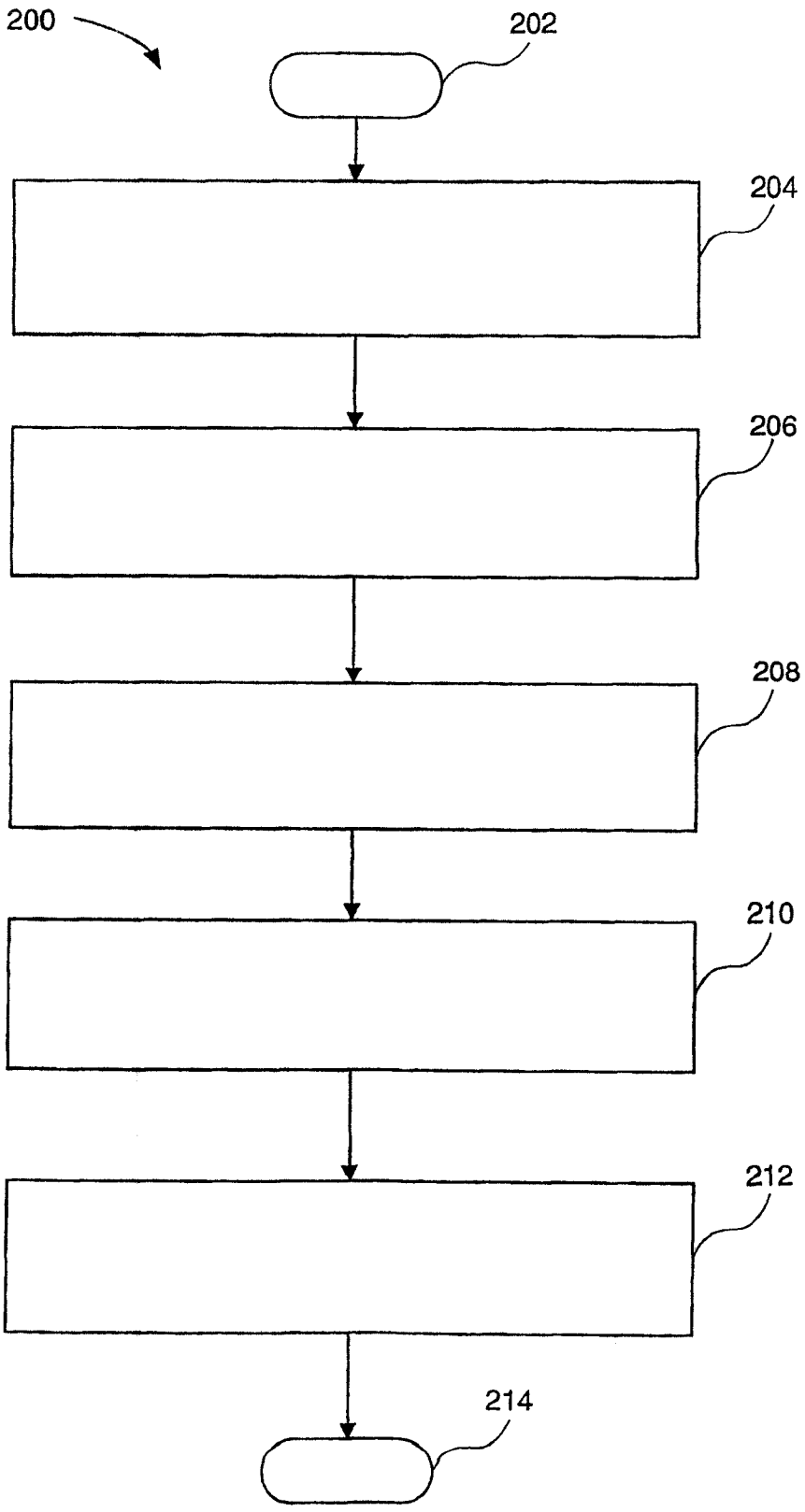


图 4

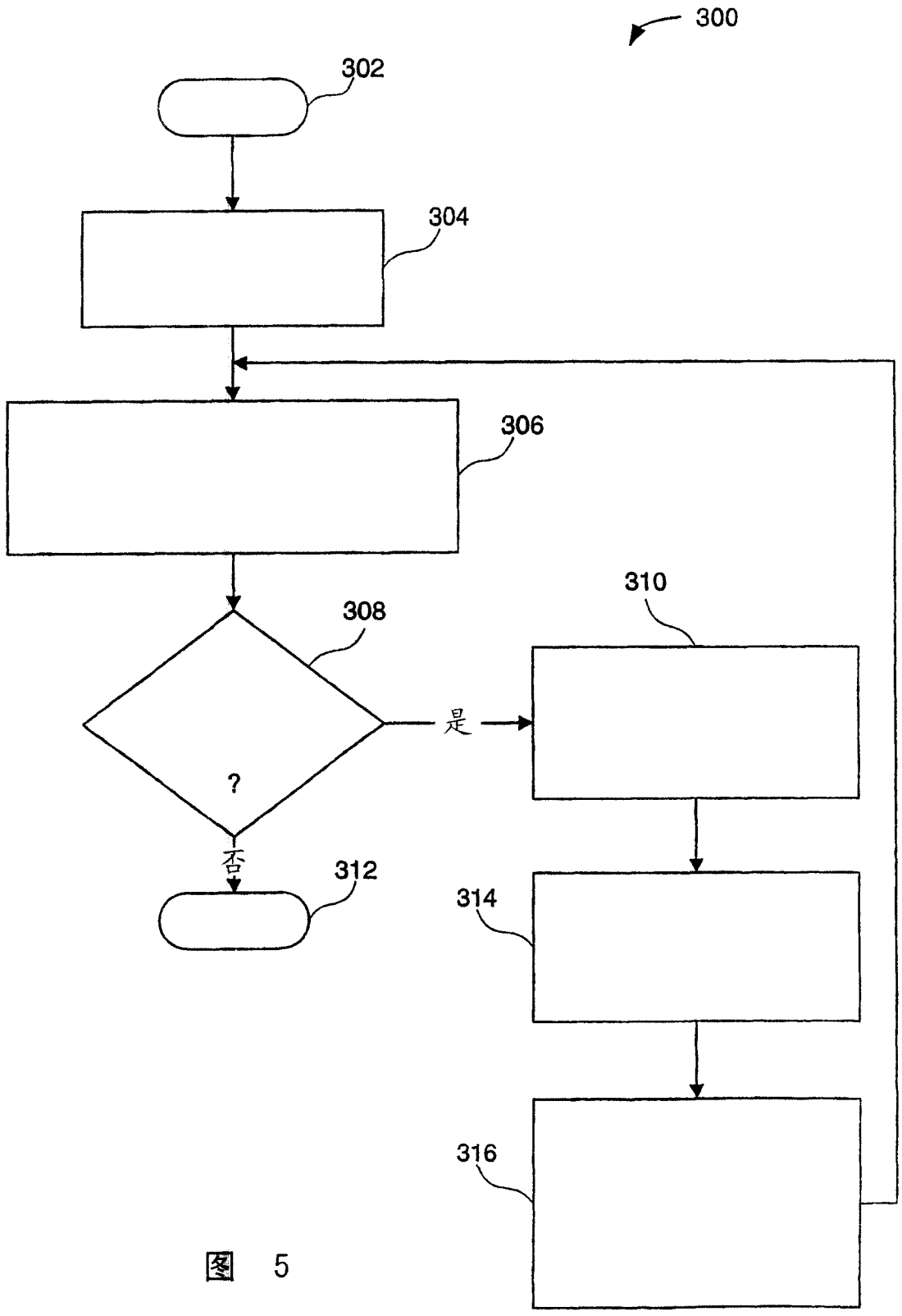


图 5