[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
G06F 1/32

[12] 发明专利申请公开说明书

[21] 申请号 02818841.1

[43] 公开日 2005 年 4 月 27 日

[11] 公开号 CN 1610872A

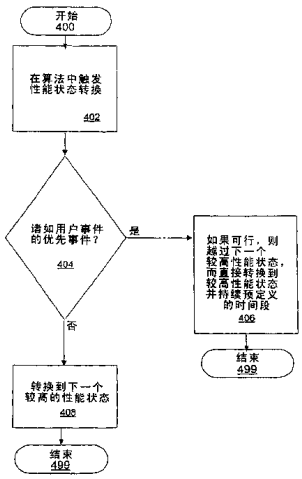
[22] 申请日 2002.9.4 [21] 申请号 02818841.1
[30] 优先权
[32] 2001. 9.25 [33] US [31] 09/965,283
[86] 国际申请 PCT/US2002/028537 2002.9.4
[87] 国际公布 WO2003/027818 英 2003.4.3
[85] 进入国家阶段日期 2004.3.25
[71] 申请人 英特尔公司
地址 美国加利福尼亚州
[72] 发明人 兰迪·斯担利
[74] 专利代理机构 北京东方亿思知识产权代理有限责任公司
代理人 王 怡

权利要求书 4 页 说明书 12 页 附图 4 页

[54] 发明名称 提供用户优先级模式的方法和装置

[57] 摘要

基于检测到用户事件，将具有多个性能状态的集成电路直接从第一性能状态转换到第三性能状态的方法、装置和系统。 该集成电路具有多个性能状态，包括第一性能状态、高于第一性能状态的第二性能状态和高于第二性能状态的第三性能状态。



ISSN1008-4274

1. 一种方法，包括：

在计算系统中检测用户事件，所述计算系统包括具有多个性能状态的集成电路，所述多个性能状态包括第一性能状态、高于所述第一性能状态的第二性能状态和高于所述第二性能状态的第三性能状态，所述计算系统具有电源，所述电源包括电池；以及

基于检测到所述用户事件，直接将所述集成电路从所述第一性能状态转换到所述第三性能状态。

2. 如权利要求 1 所述的方法，其中所述用户事件由编程环境定义，所述计算系统在所述编程环境中工作。

3. 如权利要求 1 所述的方法，其中直接转换包括没有延迟地转换。

4. 如权利要求 1 所述的方法，还包括：

根据没有故障地在所述第三性能状态下工作的热因素，使所述集成电路在所述第三性能状态下工作一段预定义的时间。

5. 如权利要求 4 所述的方法，其中所述计算系统包括膝上计算机。

6. 如权利要求 1 所述的方法，其中所述计算系统包括个人数字助理。

7. 一种装置，包括：

计算机可读介质；

具有多个性能状态的第一集成电路，所述多个性能状态包括第一性能状态、高于所述第一性能状态的第二性能状态和高于所述第二性能状态的第三性能状态，所述第一集成电路耦合到所述计算机可读介质；和

存储在所述计算机可读介质中的程序，所述程序用于管理所述第一集成电路中的功率消耗，与所述程序相关联的指令用于基于检测到用户事件，而直接将所述第一集成电路从所述第一性能状态转换到所述第三性能状态。

8. 如权利要求 7 所述的装置，其中所述第一性能状态包括第一电压电平和第一工作频率。

9. 如权利要求 7 所述的装置，其中所述第三性能状态包括第二集成电

路与所述第一集成电路共同处理指令。

10. 如权利要求 7 所述的装置，还包括：

频率调整逻辑，用于改变所述第一集成电路的工作频率，所述频率调整逻辑从所述程序接收信号。

11. 如权利要求 7 所述的装置，还包括：

电压调整逻辑，用于改变所述第一集成电路的工作电压，所述电源调整逻辑从所述程序接收信号。

12. 如权利要求 7 所述的装置，其中所述指令驻留在基本输入输出系统中。

13. 如权利要求 7 所述的装置，其中所述指令驻留在操作系统中。

14. 如权利要求 7 所述的装置，其中所述指令驻留在应用软件中。

15. 如权利要求 7 所述的装置，其中所述第一集成电路包括芯片组。

16. 如权利要求 7 所述的装置，其中所述第一集成电路包括处理单元。

17. 如权利要求 12 所述的装置，其中所述基本输入输出系统从操作系统接收关于所述用户事件已经发生的通知信号。

18. 如权利要求 11 所述的装置，其中所述程序包括与降低状态转换算法分离的升高状态转换算法。

19. 如权利要求 7 所述的装置，其中所述程序基于非用户事件的发生，将所述第一集成电路转换到下一个较高的性能状态，所述非用户事件将所述第一集成电路的利用率提高到预设的阈值以上。

20. 一种提供了指令的机器可读介质，当所述指令被机器执行时，使得所述机器执行以下操作：

在计算系统中检测用户事件；所述计算系统包括具有多个性能状态的集成电路，所述多个性能状态包括第一性能状态、高于所述第一性能状态的第二性能状态和高于所述第二性能状态的第三性能状态；以及

基于检测到所述用户事件，直接将所述集成电路从所述第一性能状态转换到所述第三性能状态。

21. 如权利要求 20 所述的机器可读介质，还包括指令，当所述指令被

机器执行时，使得所述机器还执行以下操作：

改变所述集成电路的工作频率来改变所述集成电路的性能状态。

22. 如权利要求 20 所述的机器可读介质，还包括指令，当所述指令被机器执行时，使得所述机器还执行以下操作：

改变所述集成电路的工作电压电平来改变所述集成电路的性能状态。

23. 如权利要求 20 所述的机器可读介质，还包括指令，当所述指令被机器执行时，使得所述机器还执行以下操作：

使所述集成电路在所述第三性能状态下工作一段短暂的时间。

24. 一种装置，包括：

用于在计算系统中检测用户事件的装置；所述计算系统包括具有多个性能状态的集成电路，所述多个性能状态包括第一性能状态、高于所述第一性能状态的第二性能状态和高于所述第二性能状态的第三性能状态；和

用于基于检测到所述用户事件，直接将所述集成电路从所述第一性能状态转换到所述第三性能状态的装置。

25. 如权利要求 24 所述的装置，还包括：

用于改变所述集成电路的工作频率来改变所述集成电路的性能状态的装置。

26. 如权利要求 24 所述的装置，还包括：

用于改变所述集成电路的工作电压电平来改变所述集成电路的性能状态的装置。

27. 一种提供指令的机器可读介质，当所述指令被机器执行时，使得所述机器执行以下操作：

在由电池供电的计算系统中检测用户事件，所述计算系统包括具有多个性能状态的集成电路，所述多个性能状态包括第一性能状态和最高性能状态；以及

基于检测到所述用户事件，直接将所述集成电路从所述第一性能状态转换到所述最高性能状态。

28. 如权利要求 27 所述的机器可读介质，还包括指令，当所述指令被机器执行时，使得所述机器还执行以下操作：

通过改变管理处理负载的处理器的数量来改变所述集成电路的性能状态。

29. 如权利要求 27 所述的机器可读介质，还包括指令，当所述指令被机器执行时，使得所述机器还执行以下操作：

通过改变所述集成电路的工作频率级别来改变所述集成电路的性能状态。

提供用户优先级模式的方法和装置

版权声明

技术领域

本发明一般地涉及对系统功率管理机制的资源分配的加强。更具体地说，本发明涉及响应于优先事件，移动到高性能状态。

背景技术

设计者试图降低计算机系统所消耗的功率，尤其是在移动电子设备方面。移动电子设备包括电池操作计算机系统，例如笔记本计算机、小型笔记本计算机（sub-notebook computer）和个人数字助理。通过降低这些或其他的电池操作设备所消耗的功率，用户可以享受电池充电之间的长时间的系统使用和操作。因此，主要考虑电池操作计算机系统可获得的优势，系统制造商已经向研究与开发辅助降低这些移动电子设备内的处理器所消耗的功率的技术，投入了可观的资源。

众所周知，诸如中央处理单元（CPU）的处理器所消耗的功率与供给处理器的电压的平方乘以处理器的工作频率近似成比例，$P \sim (V^2 \times f)$。对于这个关系，可见降低频率或电压将使得处理器所消耗的功率降低。但是，降低处理器工作的频率降低了处理器可以处理数据的速度。因此，降低处理器工作的频率既降低了处理器所消耗的功率，也降低了处理器的性能状态。

一般地，CPU 功率管理系统使用算法来控制处理器的性能状态。算法在降低处理器的功率消耗和对处理器的处理需求之间进行平衡。通常，算法在将处理器转换到处理器的最高性能状态之前产生一定时间的滞后或阻

碍。

例如，算法可以被编程为将处理器的处理工作量分散到三个时段，而不是提高处理速率以在一个时段中完成主要的处理负载，并在下面两个时段中不充分地利用处理器的能力。

一般地，现代移动 CPU 功率管理系统基于 CPU 利用率的简短历史来转换到较高的性能状态。CPU 利用率的历史通常由到当前为止的小的有限时间窗口来定义。一般的移动系统在很大一部分时间是处于空闲性能状态，这使得即使处理最密集的任务要达到最高性能状态也会产生一个时间窗口的一部分的延迟。实际上，有限时间窗口作为加权平均而阻碍了处理器向最高性能状态的转换。

例如，在开始处理密集任务时，由于处理器在该时段期间处于空闲的性能状态，有限时间窗口的百分之九十被以 70%的容量使用的 CPU 占用，而由于处理器上处理密集任务的需求，有限时间窗口的百分之十被以 100%的容量使用的 CPU 占用。因此，算法确定在这整个有限时间窗口期间，处理器的平均利用百分比是容量的 73%，而不是其当前对容量 100%的利用。算法将百分之九十的有限时间窗口乘以 70%的利用率，再加上百分之十的有限时间窗口乘以 100%的利用率来计算，得到 73%的利用率。例如，如果启动点（trip point）被设置为 85%，则将需要再经过 40%的时间窗口才会开始向下一个较高的性能状态转换。也许到下一个有限时间窗口时，处理器的利用百分比将反映其当前的 100%的需求，并且算法将升高处理器的性能状态。

附图说明

本发明参考以下附图，其中：

图 1 是示例性多性能状态计算机系统的方框图，该计算机系统可以一起使用来在检测到用户事件时，将集成电路从第一性能状态转换到较高性能状态；

图 2 图示了具有多个性能状态的集成电路的一个实施例的各种性能状态的图；

图 3 图示了集成电路的一个实施例的各种性能状态的图，该集成电路具有多个性能状态，包括在短暂的时间段内工作在较高的性能状态下的性能状态；以及

图 4 图示了转换集成电路性能状态的算法的一个实施例的流程图。

虽然本发明可以进行各种改进并采用其他形式，但是已经通过视图中的示例示出了其具体实施例。本发明应该被理解为不限于所公开的特定形式，而相反，本发明覆盖落在本发明的精神和范围之内的所有改进、等价物和替换物。

具体实施方式

在下面对本发明的详细描述中给出了大量具体细节，例如具体的数据信号、命名的组件框、性能级别、算法的存储和操作位置等的例子，以便于充分理解本发明。但是，对于本领域的技术人员很清楚，没有这些具体细节也可以实施本发明。在另外一些例子里，没有对公知的组件或方法进行详细的描述，而只是以方框图来描述，以避免不必要地混淆本发明。具体的细节可以变化，而仍被认为在本发明的精神和范围之内。术语耦合被定义为直接或间接连接的含义。

一般地，描述了方法、装置和系统，它们允许用户事件来触发集成电路从第一性能状态到较高性能状态的直接转换。在一个实施例中，第一性能状态是供给处理器的第一电压电平、以及对于该处理器或者计算机系统中其他类似集成电路的第一工作时钟频率。在一个实施例中，较高性能状态是供给处理器的第二较高的电压电平和对于该处理器的第二较高工作时钟频率。在一个实施例中，较高性能状态增加第二处理器以应付处理负载。在一个实施例中，降低处理器的时钟频率和/或电压电平降低了处理器所消耗的功率。在一个实施例中，功率管理算法控制供给处理器的时钟频率和电压。

图 1 是示例性多性能状态计算机系统的方框图，该计算机系统可以一起使用来在检测到用户事件时，将集成电路从第一性能状态转换到较高性能状态。在一个实施例中，计算机系统 100 包括用于传递信息的通信机制

或总线 111，以及用于处理信息的、与总线 111 耦合的、诸如处理器 112 的集成电路组件。处理器 112 可以包括微处理器，但是不限于微处理器，例如 Pentium™、PowerPC™、Alpha™等。

计算机系统 100 还包括耦合到总线 111 的随机访问存储器（RAM）或其他动态存储设备（称为主存储器），用于存储信息和要被处理器 112 执行的指令。主存储器 114 也可以在处理器 112 执行指令时，用于存储临时变量或其他中间信息。

计算机系统 100 还包括耦合到总线 111 的只读存储器（ROM）和/或其他静态存储设备 106，用于存储静态信息和用于处理器 112 的指令，还包括大容量存储器 107，例如磁盘或光盘以及其相应的盘驱动器。大容量存储器 107 耦合到总线 111，用于存储信息和指令。

计算机系统 100 还可以耦合到显示设备 121，例如阴极射线管（CRT）或液晶显示器（LCD），其耦合到总线 111 以向计算机用户显示信息。包括字母数字和其他键的字母数字输入设备（键盘）122 也可以耦合到总线 111，用于向处理器 112 传递信息和命令选择。另外的用户输入设备是耦合到总线 111 的光标控制设备 123，例如鼠标、轨迹球、轨迹板、记录针或光标定向键，用于向处理器 112 传递方向信息和目录选择，并用于控制显示设备 112 上的光标运动。

可以耦合到总线 111 的另一种设备是硬拷贝设备 124，它可以用于在诸如纸、胶片或类似类型的介质上，打印指令、数据或其他信息。另外，诸如扬声器和/或麦克风（未示出）的声音记录和回放设备，可以可选地耦合到总线上，用于与计算机系统 100 音频接口。另一个可以耦合到总线 111 的设备是有线/无线通信接口 125，用于与电话通信。

在一个实施例中，处理器 112 使用由频率调整逻辑 134 提供的时钟频率来协调处理器 112 中指令的执行。对于一个实施例，频率调整逻辑 134 包括这样的电路系统，其在将时钟信号发送到处理器 112 的内部执行单元之前，对时钟频率倍频、三倍频或将其乘以一个整数或有理数值。对于一个实施例，处理器 112 自身包括这样的电路系统，其在将时钟信号发送到处理器 112 的内部执行单元之前，对时钟频率倍频、三倍频或将其乘以一

个整数或有理数值。例如，处理器 112 的工作频率在低性能状态下可以是 225 兆赫兹，而在高性能状态下可以甚至是初始频率的八倍或 2000 兆赫兹。在一个实施例中，时钟耦合并控制处理器 112 的工作频率。

在一个实施例中，处理器 112 使用由电压调整逻辑 130 供给的电压来为其操作供电。电压调整逻辑 130 耦合到诸如电池操作计算机中的电池的电源 132，并且生成供给处理器 112 的供电电压。在一个实施例中，电压调整器耦合到处理器以确定处理器的工作电压。例如，电压调整逻辑可以供应 1.8 伏的电压电平和较高的 3.3 伏电压电平。

在一个实施例中，功率管理算法 136 响应于某个预定条件，例如预设的处理器利用百分比，而发信号通知频率调整逻辑 134 降低供给处理器 112 的工作频率。这样，预定条件降低了处理器 112 的工作频率。一旦频率降低，频率调整逻辑 134 直接与电压调整逻辑 130 通信，告诉电压调整逻辑 130 降低供给处理器 112 的电压。电压调整逻辑 130 照做，并且处理器 112 继续在此较低的功率性能状态下工作，降低了电源 130 的功率耗损。

在一个实施例中，功率管理算法 136 响应于某个优先事件，例如检测到用户事件，而发信号通知频率调整逻辑 134 将频率提高到较高的工作频率，以使处理器又可以全速工作。但是，在提高频率之前，频率调整逻辑 134 直接与电压调整逻辑 130 通信，来信号通知电压调整逻辑 130 将供电电压提高到较高的电压电平。电压调整逻辑 130 响应该请求，并且在完成时直接向频率调整逻辑 134 传回信息，告知供电电压已经提高。一旦从电压调整逻辑 130 接收到该信息，频率调整逻辑 134 就接着将频率提高回较高的值，以允许处理器 112 在较高的性能状态下工作。在一个实施例中，功率管理算法 136 被分成两个分离的算法。第一算法将处理器 112 从较高的性能状态降低而转换到较低的性能状态，第二算法将处理器 112 从较低的性能状态转换到较高的性能状态。

注意，可以存在根据本发明的其他实施例的各种其他配置和实现。

例如，在一个实施例中，处理器 112 可以是能够工作在可变工作频率和电压电平下的单个处理器。在一个实施例中，处理器 112 可以是合作工

作来响应处理负载的两个或多个处理器。在一个实施例中，处理器 112 可以是具有多个性能模式的芯片组。

例如，功率管理算法 136 可以是基于软件的或基于硬件的，例如可以在处理器之间共享的仲裁逻辑，或者是基于软件和基于硬件的结合。

例如，在一个实施例中，功率管理算法 136 检测到用户事件。一检测到该用户事件，功率管理算法 136 就触发多性能状态计算机系统 100 来转换到较高的性能状态。处理器 112 具有多个性能状态，包括第一性能状态、高于第一性能状态的第二性能状态和高于第二性能状态的第三性能状态。基于检测到用户事件，算法直接将处理器 112 从第一性能状态转换到第三性能状态。注意，第三性能状态可以是，也可以不是处理器 112 能达到的最高性能状态。

例如，在一个实施例中，优先事件可以是对最高性能状态的直接请求，该请求是由知道功率管理算法 136 的应用直接或间接地发起或模拟善意用户事件而提出的。

图 2 图示了具有多个性能状态的集成电路的一个实施例的各种性能状态的图。上面的图图示了响应于优先事件 207 的发生以及响应于非优先事件 209 的发生时，相对于时间 203 的 CPU 平均利用百分比。下面的图图示了响应于优先事件 211 的发生以及响应于非优先事件 213 的发生时，相对于同样时间线的处理器 205 的性能级别。虚线表示非优先事件 209 的示例性的 CPU 的平均利用百分比，以及响应于那些非优先事件 213 的性能级别转换。实线表示优先事件 207 的示例性的 CPU 的平均利用百分比，以及响应于优先事件 211 的性能级别转换。控制处理器性能状态的算法可以将诸如用户事件的某些事件分优先级，使得优先事件 202 的发生可以触发来立即进入诸如处理器的多性能状态集成电路中的较高性能状态。在实施例中，直接转换是响应于优先事件 202 而转换到最高性能状态 204。

在一个实施例中，响应于由非优先事件所启动来转换到较高性能状态的触发，控制处理器性能状态的算法利用在一个时间窗口（历史窗口 208）中 CPU 利用的实时历史数据，来决定转换到下一个较高的性能状态或从下一个较高的性能状态转换，所述时间窗口例如过去的一百微秒。相

反，当诸如用户事件的优先事件 202 被触发时，控制处理器性能状态的算法立即在定义的时间段 206 内处于较高的性能状态，例如最高性能状态 204。定义的时间段 206 的持续时间可以包括这样的考虑，例如较高的性能状态是可维持的还是短时间的。

注意，响应于诸如软件发起事件的非优先事件，控制处理器性能状态的算法将向下一个较高性能状态的转换阻碍一段时间。在一个实施例中，控制处理器性能状态的算法使用这样一个要求，即在历史窗口 208 的有限时间段内 CPU 201 的平均利用百分比大于预定的设置，例如 85%的利用率的设置，然后控制处理器性能状态的算法逐渐地将处理器转换到较快的性能状态。

例如，在超过为 CPU 的平均利用百分比设置的触发量后，算法可以将处理器从诸如睡眠性能状态的最低性能状态，转换到下一个较高的状态即空闲性能状态 212。经过第一短暂时间段 214，算法检查历史窗口 206 的时间段内的 CPU 利用率 201。历史窗口 208 表示了沿着时间线最近的有限宽度时间的记录，例如一百毫秒，其反映了 CPU 的平均利用百分比 201。因为对更高 CPU 利用率 201 的要求已经在历史窗口 206 的检查时段中持续了更大百分比的时间，并且因为 CPU 自身已经在较高的处理或性能状态下运行了历史窗口 206 的一部分检查时间，所以 CPU 的平均利用百分比已在改变。

注意，在一个实施例中，睡眠状态是最低的性能状态，空闲状态是下一个较高的性能状态，活动状态是仅次于最高的性能状态，并且下一个较高的性能状态是最高性能状态。在该示例中描述了四个性能状态。但是在任何实施例中可以存在两个或更多的性能状态。

当预设的 CPU 平均利用百分比仍超过诸如 95%的预设阈值时，接着算法将处理器从空闲性能状态 212 转换到下一个较高的状态，例如活动性能状态 216。经过第二短暂时间段 218，算法检查 CPU 在历史窗口 206 的时间段内的 CPU 利用率。

但是，如果预设的 CPU 的平均利用百分比仍超过触发阈值，则算法可以将处理器从活动性能状态 216 转换到下一个较高的状态，例如最高性能

状态 204。这样，在该示例中，第三时间段 220 等于在处理器转换到最高性能状态 204 之前经过的所有的转换时间段。算法可以在时间段使用这种处理功率随时间的逐步增加，来平衡对于提高性能的要求和降低电源的功率耗损的努力。

注意，即使对 CPU 利用率的要求立即增加到百分之百，也可能不超过诸如 95%CPU 利用率的触发阈值，直到在历史窗口 206 中整个宽度的时间段内的平均值等于或超过 95%。这样，在算法触发处理器向较高状态的转换之前，发生时间加权的延迟。

相反，当诸如用户事件的优先事件 202 被触发时，算法随后立即将处理器转换到诸如活动性能状态 216 的较高性能状态或最高性能状态 204。检测到优先事件 202 强迫算法越过与当前的性能级别紧邻的较高的性能级别，而直接转换到较高的性能状态。在一个实施例中，算法越过多少个性能级别取决于诸如温度考虑和可用电池功率的情况。在一个实施例中，如果处理器只有两个性能级别，则算法将处理器转换到最高性能级别。注意，对于该实施例，可以避免确定 CPU 利用率百分比的计算开销。

当处理器响应于优先事件而转换到较高的性能状态时，因为处理负载可能还未增加但是处理器的性能容量增加了，所以实际的 CPU 平均利用百分比可能会降低。例如，在优先事件 202 发生之前，处理器可以工作在 50%的 CPU 利用率下。算法可以将处理器转换到最高性能状态 204。但是，处理负载可能未增加，这样就将实际的 CPU 利用百分比 201 从例如 50%降低到 48%的 CPU 利用率 201。在一个实施例中，因为优先事件是异步的，并且经常在系统为更多的工作“准备好”（空闲）时发生，所以对于它们发生更快的转换。

在一个实施例中，用户可以对软件发起的事件增加处理时间。例如，作为用户在键盘上按下一个键的结果，Excel 图表和图形将更快地计算和重新绘画。用户事件的发生使得算法立即向较高的性能级别转换，而不论实际的 CPU 平均利用百分比如何。这样，在一个实施例中，即使在软件发起的处理任务期间，例如 Excel 电子数据表的计算，用户可以通过简单地按下一个键来增加那些式子的处理速率。

计算系统可以察觉到诸如击键、鼠标移动、操纵杆输入、鼠标点击、经由麦克风的人类命令等的用户事件。在一个实施例中，处理器工作的编程环境定义了用户事件。在此实施例中，操作环境可以定义什么事件构成用户事件以及如何检测该事件。

在一个实施例中，控制处理器性能状态的算法可以驻留在操作软件中。在一个实施例中，控制处理器性能状态的算法可以驻留在基本输入输出系统（BIOS）中。BIOS 可以是个人计算机中基本的例程集合，其存储在芯片上，并提供操作系统和计算机硬件之间的接口。在该实施例中，取决于编程语言和系统体系结构，BIOS 可以直接察觉到用户事件已经发生，或者可以例如从操作系统接收关于用户事件刚发生的通知。在一个实施例中，控制处理器性能状态的算法可以作为应用程序、可执行程序模块或其他类似的程序而驻留。在该实施例中，应用程序、可执行程序模块或其他类似的程序可以直接检测诸如用户事件的优先事件何时发生，或者从操作系统接收关于优先事件发生的通知。

在一个实施例中，在非优先事件中触发转换的阈值可以是某个 CPU 平均利用百分比，例如 85%、基于空闲与活动比率的转换或者其他预定义的阈值。

在一个实施例中，控制处理器性能状态的算法可以被分成两个分离的算法。第一算法控制向较高性能状态的转换。类似地，第二算法控制向较低性能状态的转换。在一个实施例中，处理器可以在长时间可维持的时间段内在较高的性能状态下运行。在该实施例中，最高性能状态与系统的热消散能力组合成这样，使得或者长时间的运行将不会热损坏系统内的任何组件，或者做好了对过负荷模式（override mode）的准备。在一个实施例中，由于对热的考虑，处理器可以在瞬间或短暂的时间段内运行在较高的性能状态。

图 3 图示了集成电路的一个实施例的各种性能状态的图，该集成电路具有多个性能状态，包括在短暂的时间段内工作在较高的性能状态下的性能状态。在一个实施例中，由于组件过热，该处理器的较高性能状态无法在可能不导致系统故障的情况下来长时间维持。该图图示了处理器的工作

频率随时间的变化。示出了三个性能状态，例如在 500MHz 的空闲性能状态 302、在 1000MHz 的热最大性能状态 304 和在 2000MHz 被称为峰值虚拟 MHz（peak virtual MHz）的实际最高性能状态 306。在这段时间，在图上发生了多个用户事件 308 以及非用户事件 310。

由诸如应用程序的非用户事件 310 发起的处理请求，只可以将处理器转换到热最大性能状态 304。处理器和其他系统组件可以在热最大性能状态 304 下工作可维持的时间段，而不会由于热因素引起故障。在实际最高性能状态 306 和热最大性能状态 304 之间存在热保护带 312。持久地工作在超过热最大性能状态 304 的工作频率和电压下可能损坏诸如集成组件的热敏感组件。

但是，用户事件 308 的发生或重复相对于处理时间发生得很缓慢。在用户按下第一键，并接着按下第二键的时段中，处理器可能会响应几百或上千个非用户事件处理请求 310。通常，第一用户事件 316 和第二用户事件 318 之间存在相对很大的滞后时间。在用户事件 308 之间的很大的时间滞后，一般允许计算系统的热耗散能力去除，由响应于第一用户事件 316 转换到实际最高性能状态 306 而产生的多余的热，该转换在处理器响应于第二用户事件 318 而转换到实际最高性能状态 306 之前。

但是，可以增加一些设计安全预防措施。在一个实施例中，转换到实际最高性能状态 306 在很短的时间段内，这保证没有组件由于热因素而发生故障。此外，在处理器转换到实际最高性能状态 306 之后，接着发生在预定义时间段内阻止转换到实际最高性能状态 306，此时间段例如热隙（thermal gap）320。如果在热隙段 320 期间发生诸如鼠标点击 326 的用户事件，则算法将处理器转换到热最大性能状态 304。例如，如果诸如击键 324 和鼠标点击 326 的两个用户事件实质上同时发生，则击键 234 将处理器转换到实际最高性能状态 306，而发生在热隙 320 期间的鼠标点击 326，只触发向热最大性能状态 304 的转换。在一个实施例中，热隙段 320 被隔开这么长，即等于在由快速用户生成的用户事件 308 之间观察到的时间延迟。

图 4 图示了转换集成电路性能状态的算法的一个实施例的流程图。在

一个实施例中，嵌入计算机可读介质的程序运行下面的步骤。

在框 402，算法检测这样的触发，即将诸如处理器的集成电路转换到较高性能状态。该触发可以是检测到诸如用户事件或类似事件的优先事件。该触发也可以是诸如 CPU 的平均利用百分比的预设阈值。如果该触发是非优先事件，则响应于该触发时可以存在延迟，来平衡处理器性能和功率消耗的因素。

在框 404，算法确定处理事件是否是诸如用户事件的优先事件，该优先事件使得算法立即将处理器转换到较高的性能状态。

在框 406，如果算法确定处理器事件是优先事件，则如果可行的话，算法立即越过下一个较高性能状态，而将处理器转换到较高的性能状态一段预定的时间段。这样，如果有第一性能状态、高于第一性能状态的第二性能状态、高于第二性能状态的第三性能状态以及高于第三性能状态的第四性能状态，则算法可以转换到第三状态或第四状态，以带给用户可能的对性能的显著提高。在一个实施例中，如果只有两个性能状态，则算法将处理器转换到最高性能状态。在一个实施例中，如果存在两个更多性能状态，则算法将处理器转换到最高性能状态。在一个实施例中，在经过预定的时间段之后，功率管理算法随后基于 CPU 利用率，而将处理器转换到适合的性能状态。

在一个实施例中，根据没有故障地在最高性能状态下工作的热因素，最高性能状态可以维持很长的时间段。在一个实施例中，根据没有故障地在最高性能状态下工作的热因素，最高性能状态可以维持短暂的时间段。

在框 408，如果算法确定处理事件不是优先事件，则发送将处理器转换到下一个较高性能状态。

在一个实施例中，用于帮助算法的软件可以嵌入机器可读介质。机器可读介质包括任何以机器（例如，计算机）可读的形式提供（例如，存储和/传输）信息的机构。例如，机器可读介质包括只读存储器（ROM）；随机访问存储器（RAM）；磁盘存储介质；光存储介质；闪存设备；DVD；电、光、声或其他形式的传播信号（例如载波、红外信号、数字信号）；EPROM、EEPROM、FLASH、磁或光卡；或任何类型的适于存储

电子指令的介质。较慢的介质可以用较快的更实用的介质来高速缓存。

上面详细描述的某些部分展示为对于计算机存储器内数据比特的操作的符号表示和算法。这些算法描述和表示是那些数据处理领域的技术人员所使用的方法，以最有效地将他们的工作传达给本领域其他技术人员。这里算法一般被认为是导致所需结果的自相容步骤。尽管不是必须的，但通常这些量采取电或磁信号的形式，其能够被存储、传递、结合、比较或者操纵。将这些信号称为比特、值、元素、符号、字符、项（term）、数字等等，这已被证明有时是方便的，主要是为了共同使用。

但是，应该意识到，所有这些和类似的术语都与适合的物理量相关联，并且只是赋予这些量的方便的标记。除非特别声明，否则从上面的讨论很清楚，应该意识到在整个说明书中，利用诸如“处理”或“计算”或“运算”或“确定”或“显示”等的讨论，是指计算机系统或类似电子计算设备的动作和处理，所述计算机系统或类似电子计算设备将表示为计算机系统中的寄存器和存储器中的物理（电子）量的数据，操纵并转换为类似地表示为计算机系统存储器或寄存器或其他这样的信息存储、传输或显示设备中的物理量的其他数据。

虽然已经示出了本发明的一些具体实施例，但是本发明不限于这些实施例。例如，大多数由电子硬件组件运行的功能可以由软件模拟来复制。这样，软件程序可以发出命令来模仿用户事件命令代码。即使实际的用户事件还没发生，检测用户事件命令的算法也直接将处理器转换到较高的性能状态。在一个实施例中，优先事件可以是软件命令，其被发出来使得处理器直接转换到较高的性能状态。本发明应被理解为不限于这里描述的具体的实施例，而只由所附权利要求的范围来限制。

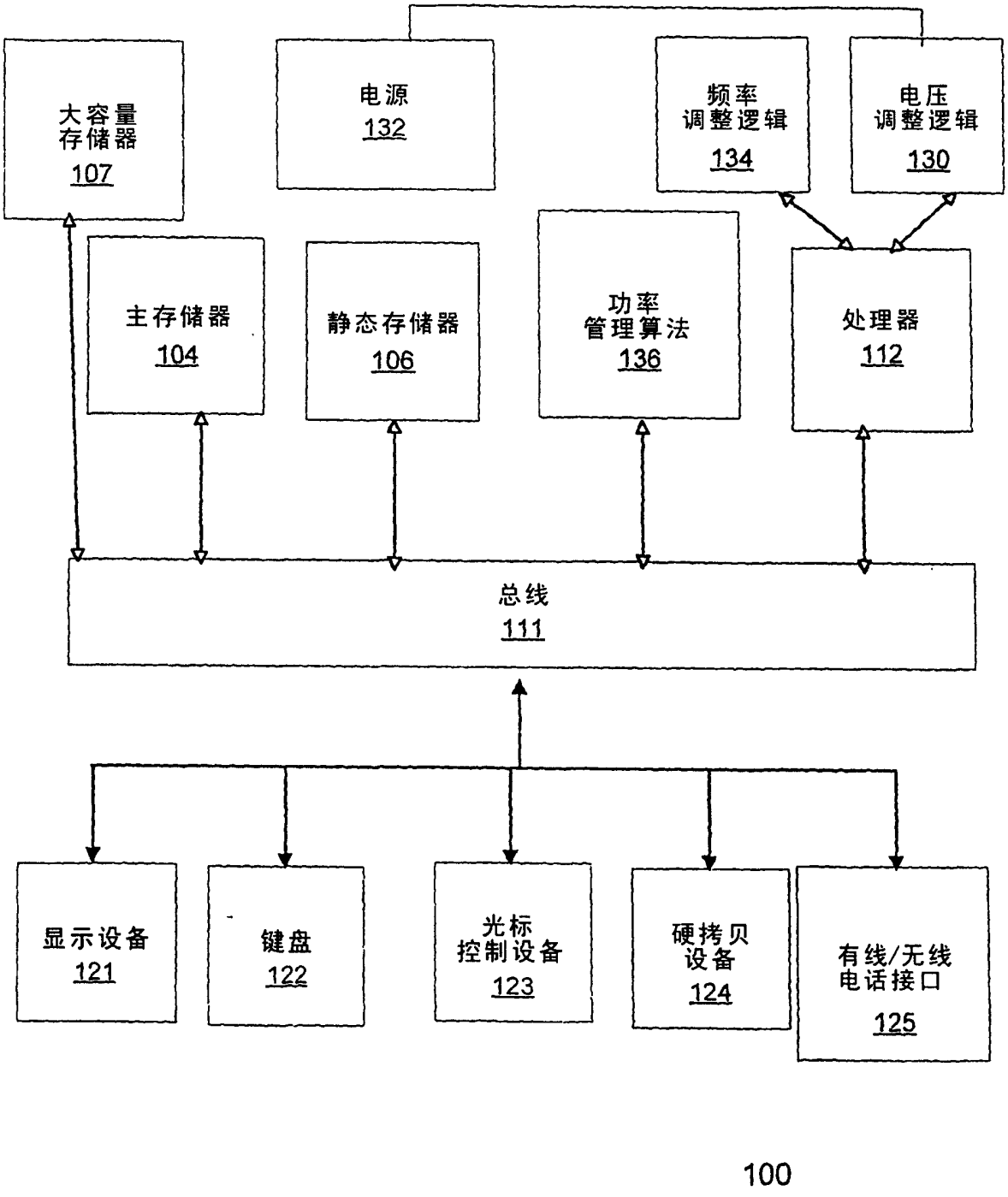

大容量
存储器
107
电源
132
频率
调整逻辑
134
电压
调整逻辑
130
主存储器
104
静态存储器
106
功率
管理算法
136
处理器
112
总线
111
显示设备
121
键盘
122
光标
控制设备
123
硬拷贝
设备
124
有线/无线
电话接口
125
100

图1

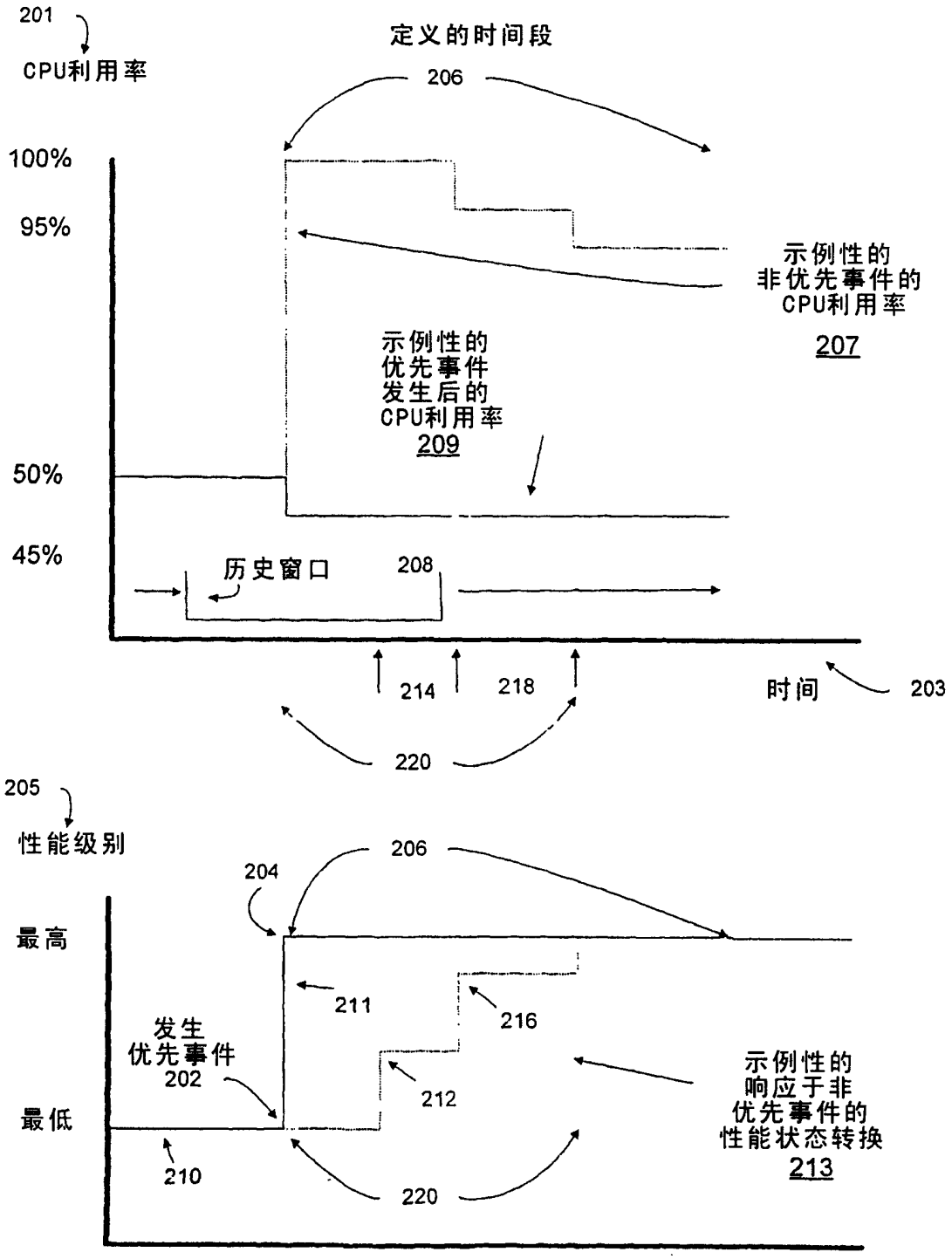

201
CPU利用率
定义的时间段
206
100%
95%
示例性的
非优先事件的
CPU利用率
207
示例性的
优先事件
发生后的
CPU利用率
209
50%
45%
历史窗口
208
214
218
时间
203
220
205
性能级别
206
204
最高
211
216
发生
优先事件
202
212
示例性的
响应于非
优先事件的
性能状态转换
213
最低
210
220

图2

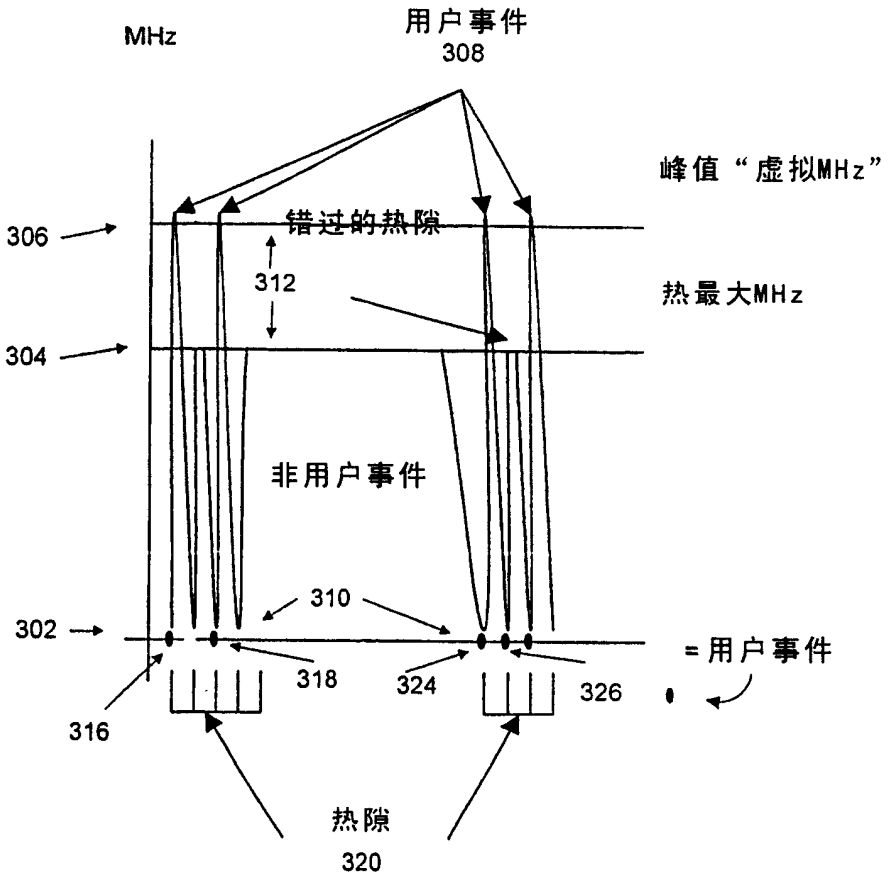

MHz
用户事件
308
峰值“虚拟MHz”
306
错过的热隙
312
热最大MHz
304
非用户事件
310
302
=用户事件
318
324
326
316
热隙
320

图3

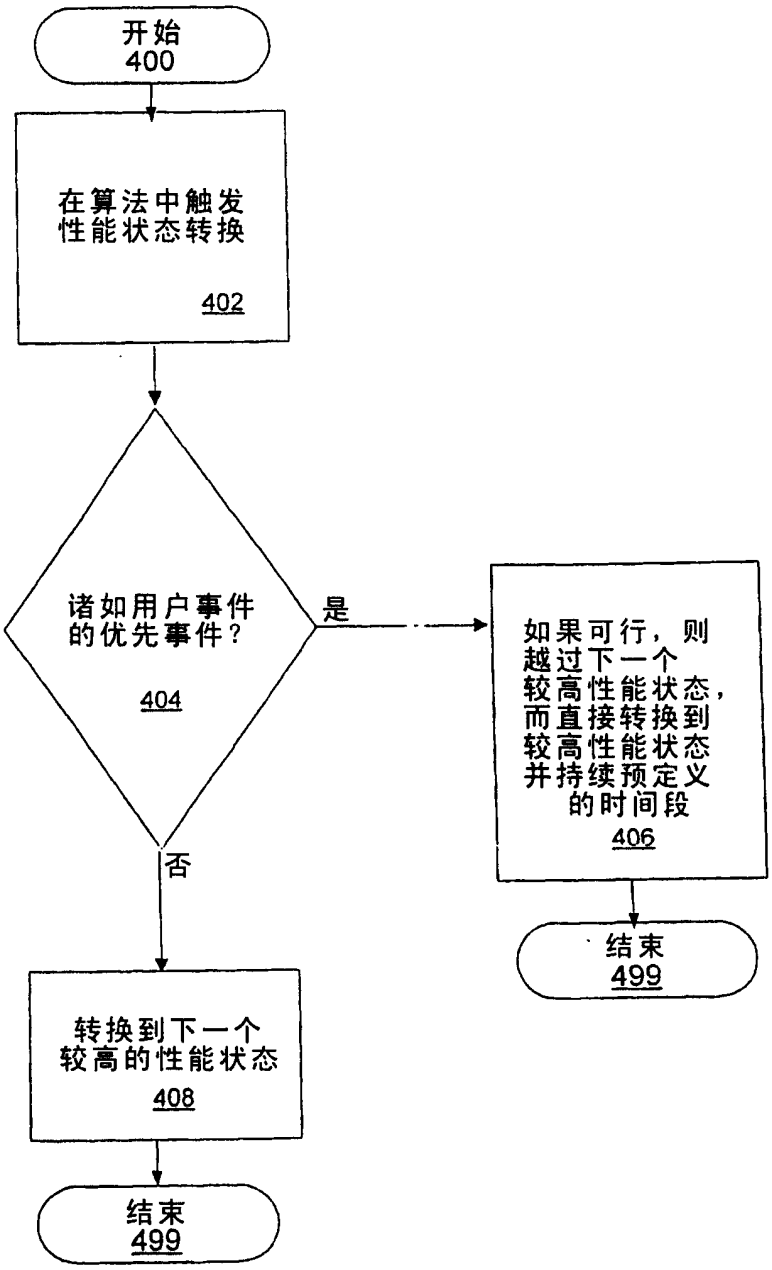

开始
400
在算法中触发
性能状态转换
402
诸如用户事件
的优先事件？
404
是
否
如果可行，则
越过下一个
较高性能状态，
而直接转换到
较高性能状态
并持续预定义
的时间段
406
结束
499
转换到下一个
较高的性能状态
408
结束
499

图4