(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 102063286 B
(45)授权公告日 2015.09.16

(21)申请号 201010528139.5
(22)申请日 2010.10.19
(30)优先权数据
0918297.3 2009.10.19 GB
(73)专利权人 瑞士优北罗股份有限公司
地址 瑞士塔尔维尔
(72)发明人 M·奥里奇 J·韦尔东克
(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 11258
代理人 李晓冬
(51)Int.Cl.
G06F 9/30(2006.01)
(56)对比文件
US 6757810 B1,2004.06.29, 全文.
JP 平4-3226 A,1992.01.08, 说明书第2页右栏第30行-第6页左栏第12行和附图1-4.
CN 100538640 C,2009.09.09, 全文.
审查员 陈敏

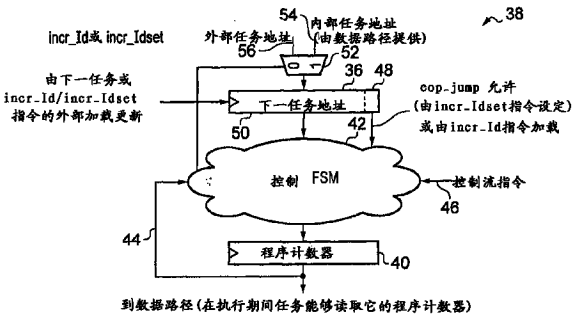
权利要求书2页 说明书8页 附图5页

(54)发明名称

程序流控制

(57)摘要

本发明涉及程序流控制,更具体地,涉及数据处理设备。其中,所述数据处理设备包括具有指令解码器18的数据引擎6,用于产生一个或多个控制信号24,所述一个或多个控制信号24用于控制处理电路系统20以执行由所述被解码的程序指令指定的数据处理操作。所述指令解码器18响应标记指令而从流控制寄存器38读取可编程的流控制值。所述可编程的流控制值指定在当前的程序指令序列的执行完成时将被采取的动作。所述被采取的动作可以是在目标程序指令序列开始处跳转到目标程序指令或者进入等待新的处理任务将被发起的空闲状态。



CN 102063286 B

1. 一种用于处理数据的设备，所述设备包括：

指令解码器电路系统，所述指令解码器电路系统响应程序指令以产生一个或多个控制信号；

处理电路系统，所述处理电路系统响应所述一个或多个控制信号以执行由所述程序指令指定的数据处理操作；以及

流控制寄存器，所述流控制寄存器被配置为存储可编程的流控制值，其中所述流控制值包括跳转允许域；其中

所述指令解码器电路系统响应指示正被执行的当前程序指令序列的末尾的标记指令而从所述流控制寄存器读取所述可编程的流控制值并且依据存储在所述跳转允许域中的值产生所述一个或多个控制信号以在所述当前程序指令序列完成之后触发下列各项之一：

(i) 从目标程序指令开始处理目标程序指令序列；以及

(ii) 进入等待新的处理任务将被发起的空闲状态。

2. 根据权利要求 1 所述的设备，其特征在于，其中所述可编程的流控制值指定所述目标程序指令的存储器地址。

3. 根据权利要求 1 所述的设备，其特征在于，所述可编程的流控制值包括指定所述目标程序指令的存储器地址的目标地址域。

4. 根据权利要求 1 所述的设备，其特征在于，其中所述跳转允许域是跳转允许位。

5. 根据权利要求 1 所述的设备，其特征在于，其中所述标记指令是程序指令的末尾，其标识所述当前程序指令序列的最后的程序指令。

6. 根据权利要求 1 所述的设备，其特征在于，其中所述目标程序指令序列是用于控制任务调度的目标调度程序。

7. 根据权利要求 1 所述的设备，其特征在于，其中所述设备包括耦合到数据引擎的主机处理器并且所述指令解码器电路系统、所述处理电路系统以及所述流控制寄存器是所述数据引擎的一部分。

8. 根据权利要求 7 所述的设备，其特征在于，其中所述目标程序指令序列是用于控制任务调度的目标调度程序，并且其中所述目标调度程序由所述数据引擎执行并且所述当前程序指令序列被调用供所述主机处理器和所述目标调度程序之一执行。

9. 根据权利要求 8 所述的设备，其特征在于，其中所述主机处理器使用由所述主机处理器执行的主机调度程序来调用所述当前程序指令序列。

10. 根据权利要求 9 所述的设备，其特征在于，其中：

(i) 在执行所述目标调度程序的所述数据引擎调用所述当前程序指令序列时，所述可编程的流控制值被编程为在所述当前程序指令序列完成之后触发返回所述目标调度程序的执行的值；以及

(ii) 在执行所述主机调度程序的所述主机处理器调用所述当前程序指令序列时，所述可编程的流控制值被编程为在所述当前程序指令序列完成之后触发所述数据引擎进入所述空闲状态的值。

11. 根据权利要求 7 所述的设备，其特征在于，其中响应于由所述指令解码器电路系统用所述数据引擎解码的加载程序指令，所述可编程的流控制值被存储在所述流控制寄存器

中。

12. 根据权利要求 11 所述的设备，其中所述指令解码器电路系统响应：

(i) 所述加载程序指令的第一形式，其中所述跳转允许域被强加为指定所述指令解码器电路系统生成所述一个或多个控制信号以在所述当前程序指令序列完成之后触发从所述目标程序指令开始处理所述目标程序指令序列的值，所述可编程的流控制值中包括的目标地址域指定所述目标程序指令的存储器地址；以及

(ii) 所述加载程序指令的第二形式，其中所述跳转允许域被强加为指定所述指令解码器电路系统生成所述一个或多个控制信号以在所述当前程序指令序列完成之后触发从所述目标程序指令开始处理所述目标程序指令序列的值和指定所述空闲状态在所述当前程序指令序列的执行已完成时被进入的值中的一个，其中所述可编程的流控制值中包括的目标地址域指定所述目标程序指令的存储器地址。

13. 一种用于处理数据的设备，所述设备包括：

指令解码器装置，所述指令解码器装置响应用于产生一个或多个控制信号的程序指令；

处理装置，所述处理装置响应所述一个或多个控制信号用于执行由所述程序指令指定的数据处理操作；以及

流控制寄存器装置，所述流控制寄存器装置用于存储可编程的流控制值，其中所述可编程的流控制值包括跳转允许域；其中

所述指令解码器装置响应指示正被执行的当前程序指令序列的末尾的标记指令而从所述流控制寄存器装置读取所述可编程的流控制值并且依据所述跳转允许域中存储的值产生所述一个或多个控制信号以在所述当前程序指令序列完成之后触发下列各项之一：

(i) 从目标程序指令开始处理目标程序指令序列；以及

(ii) 进入等待新的处理任务将被发起的空闲状态。

14. 一种处理数据的方法，所述方法包括下列步骤：

响应程序指令，产生一个或多个控制信号；

响应于所述一个或多个控制信号，执行由所述程序指令指定的数据处理操作；以及

存储可编程的流控制值，其中所述可编程的流控制值包括跳转允许域；其中

响应于指示正被执行的当前程序指令序列的末尾的标记指令，读取所述可编程的流控制值并且依据所述跳转允许域中的值产生所述一个或多个控制信号，以在所述当前程序指令序列完成之后触发下列各项之一：

(i) 从目标程序指令开始处理目标程序指令序列；以及

(ii) 进入等待新的处理任务将被发起的空闲状态。

程序流控制

技术领域

[0001] 本发明涉及数据处理系统领域。更具体地，本发明涉及数据处理系统内的程序流的控制。

背景技术

[0002] 向数据处理系统提供指令解码器电路系统是已知的，所述指令解码器电路系统响应程序指令以产生一个或多个控制信号，用于控制处理电路系统以执行由所述程序指令指定的数据处理操作。组织程序的方便的方法是将它们划分成当需要时能够被调用的例程或者函数。作为例子，代码可以被编写以及被编译以执行函数，诸如在一组数据值上计算傅立叶变换。这个函数可以从程序流内的多个不同的点被调用。也可能的是：在包括多于一组用于执行程序指令的硬件的系统中，一组硬件可以执行一组指令而另一组硬件执行另一组指令。在这种环境中，一组硬件可以调用将被另一组硬件执行的函数，诸如傅立叶变换。在这样的环境中，形成用于执行特定函数的专用硬件是可能的。

[0003] 用于执行特定函数的专用硬件的例子是数据引擎。数据引擎可以被提供以执行计算密集的任务，举例来说诸如媒体处理。数据引擎典型地包括被设计用于有效地执行与所述数据引擎相关联的任务的专用硬件。这样的数据引擎可以与通用处理器相结合而工作，所述通用处理器调度将由所述数据引擎执行的任务。在一些实例中，可能的是数据引擎自身使用在数据引擎上执行的调度程序来调度其自己的任务以供该数据引擎执行。

[0004] 用于执行特定函数的程序的开发和测试是昂贵并且费时的。程序需要少量的内存存储器 (memory storage) 是期望的。

发明内容

[0005] 从一个方面来看，本发明提供了用于处理数据的设备，所述设备包括：

[0006] 指令解码器电路系统，所述指令解码器电路系统响应程序指令以产生一个或多个控制信号；

[0007] 处理电路系统，所述处理电路系统响应所述一个或多个控制信号以执行由所述程序指令指定的数据处理操作；以及

[0008] 流控制寄存器，所述流控制寄存器被配置为存储可编程的流控制值；其中

[0009] 所述指令解码器响应指示正被执行的当前程序指令序列的末尾的标记指令以从所述流控制寄存器读取所述可编程的流控制值并且依据所述可编程的流控制值产生所述一个或多个控制信号以在所述当前程序指令序列完成之后触发下列各项之一：

[0010] (i) 从目标程序指令开始处理目标程序指令序列；以及

[0011] (ii) 进入等待新的处理任务将被发起的空闲状态。

[0012] 本发明的技术辨识正被执行的当前程序指令序列可能已经通过不同机制被调用，其中取决于当前程序指令序列如何被调用，不同的行为在当前程序指令序列的执行完成时是恰当的。本发明的技术使用存储在流控制寄存器内的程序流控制值来确定什么行为应当

在当前程序指令序列完成执行时发生，而不是取决于它们如何被调用来提供不同版本的当前程序指令序列。因此，所述当前程序指令序列不需要被修改或者以不同的形式被提供，并且所述当前程序指令序列还可以支持在完成时对不同行为的触发。更具体地，期望在完成时支持的所述不同行为触发从目标程序指令开始处理目标程序指令序列或者进入空闲状态，例如等待新的处理任务被发起。这些行为支持在其中当前程序指令序列根据目标调度程序被调用的系统，所述目标调度程序依据执行当前程序指令序列的相同机制执行并且支持在其中所述当前程序指令序列根据与执行当前程序指令序列的机制分开的机制而被调用的机制，以便在所述当前程序指令序列完成时随后空闲状态应当被进入。

[0013] 由于所述可编程的流控制值中的一些值而被执行的目标程序指令可以是在固定的位置处（例如硬连线的向量）的固定的目标程序指令。然而，在可编程的流控制值指定程序指令的存储器地址时更大的灵活性被实现。因此，依据用于所述可编程的流控制值的值集合，所述目标程序指令可以被改变。

[0014] 所述可编程的流控制值可以包括跳转允许域，所述跳转允许域存储指定所述可编程的流控制值是否包括目标地址域的值，所述目标地址域指定目标程序指令的存储器地址，或者可替代地，存储指定在当前程序指令序列的执行已完成时所述空闲状态应当被进入的值。这个跳转允许域可以是跳转允许位，其被用于在当前程序指令序列完成时可替代的行为之间的切换。

[0015] 所所述标记指令可以是程序指令的末尾，所述指令解码器响应于所述标记指令而触发可替代的例程末尾行为，所述程序指令的末尾标识所述当前程序指令序列的最后的程序指令。这个程序指令的末尾可以是所述序列中的最后的指令或者可以位于当前程序指令序列的最后的指令之前的预先确定数量的指令处。后一种行为在当前程序指令序列在流水线执行机制上被执行时是有用的，以便通过确保如果到目标程序指令序列的分支被要求，则这可以在当前程序指令序列的最终指令之前足够的时间中被标识使得目标程序指令可以被取来以开始继续的执行来促进流水线的有效使用。

[0016] 其执行可以以一个行为而被触发的目标程序指令序列可以采取各种不同的形式。作为例子，可能的将是：把多个函数与被用于触发下一个函数的开始的一个函数的末尾链接在一起。可替代的机制是目标程序指令序列是用于控制任务调度并且由正在执行当前程序指令序列的相同机制执行的目标调度程序，即所述机制使用目标调度程序来调度其自己的任务。

[0017] 利用上述技术的设备可以具有各种不同形式并且适用于各种不同的多处理环境。然而，所述技术良好适用的一个环境是在所述设备包括耦合于数据引擎的主机处理器并且所述指令解码器电路系统、所述处理电路系统和所述流控制寄存器是所述数据引擎的一部分的时候。

[0018] 在这个环境内，所述目标调度程序可以由所述数据引擎执行并且所述当前程序指令序列可以被所述主机处理器调用或者被所述目标调度程序调用。将所述流控制寄存器用于控制在当前程序指令序列的完成时的行为允许相同的当前程序指令序列独立于其如何被调用以及在该当前程序指令序列完成时如何行为是恰当的而被使用。

[0019] 所述主机处理器还可以利用主机调度程序来调用所述当前程序指令序列供所述数据引擎执行。

[0020] 在一些实施例中，所述流控制值可以被编程以便当所述目标调度程序调用所述当前程序指令序列时，则返回所述目标调度程序的执行在所述当前程序指令序列的完成之后被触发并且当后调度程序调用所述当前程序指令序列时，则所述数据引擎在所述当前程序指令序列完成之后被触发以进入空闲状态。

[0021] 响应于在所述数据引擎内被解码和执行的加载程序指令，所述可编程的流控制值可以被加载。

[0022] 所述加载程序指令可以具有在其中所述跳转允许域被强加为指定所述可编程的流控制值包括目标地址域的值的一种形式，所述目标地址域指定将被触发供在所述当前程序指令序列完成时执行的目标程序指令的存储器地址，以及在其中所述跳转允许域可以被设定为在所述当前程序指令序列的执行已完成时给出上述行为或者强迫进入所述空闲状态的值的第二形式。所述第一形式不需要是现有的。

[0023] 从另一个方面来看，本发明提供了用于处理数据的设备，所述设备包括：

[0024] 指令解码器装置电路系统，所述指令解码器装置电路系统响应用于产生一个或多个控制信号的程序指令；

[0025] 处理装置，所述处理装置响应所述一个或多个控制信号用于执行由所述程序指令指定的数据处理操作；以及

[0026] 流控制寄存器装置，所述流控制寄存器装置用于存储可编程的流控制值；其中

[0027] 所述指令解码器装置响应指示正被执行的当前程序指令序列的末尾的标记指令以从所述流控制寄存器装置读取所述可编程的流控制值并且依据所述可编程的流控制值产生所述一个或多个控制信号以在所述当前程序指令序列完成之后触发下列各项之一：

[0028] (i) 从目标程序指令开始处理目标程序指令序列；以及

[0029] (ii) 进入等待新的处理任务将被发起的空闲状态。

[0030] 从另外的方面来看，本发明提供了处理数据的方法，所述方法包括下列步骤：

[0031] 响应程序指令，产生一个或多个控制信号；

[0032] 响应于所述一个或多个控制信号，执行由所述程序指令指定的数据处理操作；以及

[0033] 存储可编程的流控制值；其中

[0034] 响应于指示正被执行的当前程序指令序列的末尾的标记指令，读取所述可编程的流控制值并且，依据所述可编程的流控制值，产生所述一个或多个控制信号以在所述当前程序指令序列完成之后触发下列各项之一：

[0035] (i) 从目标程序指令开始处理目标程序指令序列；以及

[0036] (ii) 进入等待新的处理任务将被发起的空闲状态。

[0037] 根据将结合附图而被阅读的下面对说明性的实施例的详细描述，本发明的上述以及其他目的、特征和优点将是显而易见的。

附图说明

[0038] 图 1 示意性地示出了包括主机处理器和数据引擎的片上系统集成电路；

[0039] 图 2 示意性地示出了供在图 1 的数据引擎内使用的程序流控制机制；

[0040] 图 3 示意性地示出了存储可编程的流控制值的流控制寄存器；

[0041] 图 4 示意性地示出了存储相应的程序指令的主机处理器存储器和数据引擎存储器；

[0042] 图 5 示意性地示出了由所述数据引擎执行的并且包括以程序指令的末尾形式的标记指令的当前程序指令序列；

[0043] 图 6 示意性地示出了在主机处理器调用 (invoke) 函数在所述数据引擎中的执行的操作流；以及

[0044] 图 7 是示意性地示出通过在数据引擎上执行的目标调度器的函数调度的流程图。

具体实施方式

[0045] 图 1 示意性地示出了包括经由系统总线 8 连接的主机处理器 4 和数据引擎 6 的片上系统集成电路 2。所述主机处理器 4 包括处理器核 10、主机存储器 12 和系统接口 14。所述数据引擎 6 包括处理核 16，该处理核 16 包括指令解码器 18 和处理电路系统 20。指令紧密耦合的存储器 22 存储供所述指令解码器 18 解码以产生控制所述处理电路系统 20 的控制信号 24 的程序指令序列。数据紧密耦合的存储器 26 存储由所述处理电路系统 20 操纵的输入和输出数据值，所述处理电路系统 20 在来自所述指令紧密耦合的存储器 22、由所述指令解码器 18 解码的程序指令的控制下。系统接口 28 将所述数据引擎 6 耦合到所述系统总线 8。直接存储器访问单元 30 允许所述主机处理器 4 经由判别器 32 和 34 从指令紧密耦合的存储器 22 和数据紧密耦合的存储器 26 读取数据值并且将数据值写入所述指令紧密耦合的存储器 22 和所述数据紧密耦合的存储器 26 中。

[0046] 在操作中，所述主机处理器 4 典型地将执行由存储在所述主机存储器 12 内的程序指令所指示的控制和其他高级处理函数。存储在所述主机存储器 12 内的程序指令可以包括被用于在数据引擎 6 上调用处理操作的主机调度程序。所述数据引擎 6 典型地执行较低等级的计算密集的处理操作，诸如媒体处理、压缩、解密等等。

[0047] 被调用供在所述数据引擎 6 上执行的函数（当前程序指令序列）的正常操作是这个当前程序指令序列将执行直到在其末尾遇到程序末尾 (EOP) 指令（标记指令），因此所述数据引擎 6 将进入等待下一个任务在其上将被调用的空闲状态。也可能的是所述数据引擎 6 可以包括其自己的以被存储在所述指令紧密耦合的存储器 22 内的目标调度程序的形式的调度机制。这个目标调度程序可以调用将在所述数据引擎 6 上被执行的函数，因而允许所述数据引擎“自调度”。在执行通过由所述数据引擎 6 自身所执行的目标调度程序在所述数据引擎 6 上被调用的函数的末尾处，程序流可以返回那个目标调度程序，或者至少返回另一程序（例如被链接的函数），而不是所述数据引擎 6 进入空闲状态，如对于从所述主机处理器 4 被调用的函数的情况那样。

[0048] 执行被调用的函数的当前程序指令序列可以是从库中取得的公共的编程指令序列，该库被共享用于从所述主机处理器 4 和从所述数据引擎 6 两者被调用的函数。这是为了避免不得不提供当前程序指令序列（库函数）的分开的形式以依据所述主机处理器 4 和所述数据引擎 6 中的哪一个正引起当前程序指令序列的执行而被使用。这节省了编码工作强度和程序存储资源。尽管如此，在被调用的函数完成时不同的动作被触发是必要的。使用所述数据引擎 6 内的流控制寄存器 36 来控制这种不同的行为。

[0049] 所述流控制寄存器 36 存储在正被执行的当前程序指令序列内遇到程序指令末尾

（标记指令）时由所述指令解码器 18 读取的可编程的流控制值。所述可编程的流控制值指定哪种类型的行为将在所述当前程序指令序列完成执行时被触发。被触发的一种类型的行为在执行被完成时将所述数据引擎 6 置于空闲状态中（低功率状态）。所述可编程的流控制值的不同的值被用于在所述当前程序指令序列的执行完成时触发到目标程序指令的跳转，该目标程序指令由所述可编程的流控制值内的目标地址指定。这个目标地址和所述目标程序指令可以在所述数据引擎 6 是“自调度”时处于所述数据引擎 6 的目标调度程序内。因此，在所述当前程序指令序列完成时控制被返回到所述目标调度程序。也可能的是所述可编程的程序控制值可以存储对应于到另一函数的跳转的地址，该另一函数被与所述当前程序指令序列链接。

[0050] 图 2 示意性地示出所述数据引擎 6 内的程序流控制机制 38。程序计数器寄存器 40 存储所述指令紧密耦合的存储器 22 内的存储器地址，将由所述指令解码器 18 解码的下一个指令从该存储器被获取。所述指令解码器 18 产生用于控制来自被解码的程序指令的程序流的控制信号（一个或多个控制信号）24。由解码器在并联的处理单元（例如 ALU、AGU 等等）中产生的其他控制信号被用于控制所述处理电路系统 20 以执行对应于被解码的程序指令的数据处理操作。所述处理操作可以是简单的，诸如加载、存储、移位等等，或者更复杂的操作，诸如乘法、乘法累积操作、过滤操作等等，这取决于所述数据引擎 6 所要求的处理的性质。

[0051] 由控制流状态机 42 向所述程序计数器寄存器 40 加载程序计数器值。所述控制流状态机 42 可以向所述程序计数器寄存器 40 加载从各种不同的源取得的程序计数器值。当所述当前程序指令序列正被顺序地执行时，则所述控制流状态机 42 内的反馈路径 44 和增量电路被用于稳定地增加所述程序计数器值并单步调试 (step through) 正被执行的当前程序指令序列。当控制流指令在所述当前程序指令序列（例如分支指令）内被指令解码器 18 解码时，则那些控制流指令的目标地址经由控制流指令路径 46 被提供给所述控制流状态机 42 并且被加载到所述程序计数器寄存器 40 中以便在执行点执行所述跳转。

[0052] 所述流控制寄存器 36 还向所述控制流状态机 42 提供输入。所述流控制寄存器 36 存储由跳转允许标志 48 和下一任务地址 50 形成的可编程的流控制值。在程序末尾 (EOP) 指令在所述当前程序指令序列内被遇到时，这可以通过所述控制流指令路径 46 上的信号而被发信号通知所述控制流状态机 42。程序指令的末尾控制所述控制流状态机 42 从所述流控制寄存器 36 读取所述可编程的流控制值。如果所述跳转允许位 48 没有被设定，则所述控制流状态机 42 将触发所述数据引擎 6 在所述当前程序指令序列完成时进入空闲状态（例如在所述程序指令末尾的解码之后的预先确定的固定数量的处理周期）。这种行为对应于在所述主机处理器 4 已引起所述当前程序指令序列的执行时的需要。

[0053] 如果所述跳转允许标志 48 被设定，则所述控制流状态机 42 将在遇到所述程序指令末尾时读取所述下一任务地址 50（所述指令紧密耦合的存储器 22 内的目标程序指令的存储器地址）并且触发所述目标程序指令的存储器地址被加载到所述程序计数器寄存器 40 中（以匹配于流水线深度的延迟），以便所述目标程序指令将在所述当前程序指令序列完成时被获取并且被执行。

[0054] 在这个例子中，所述跳转允许标志 48 是单个位。更一般地，所述跳转允许标志可以被看作是跳转允许域，其可以包括一个或多个位。

[0055] 所述流控制寄存器 36 通过加载指令被加载以所述可编程的流控制值 48、50，其因而可以具有两个不同的形式。在第一形式 incr_ldset 中，所述跳转标志 48 被强加为设定值（″ 1″ ）并且所述目标程序指令的存储器被加载到所述可编程的流控制值 36 的下一任务地址 50 部分。第二形式的加载指令 incr_ld 以相似的方式作用，除了所述跳转允许标志没有被强加为设定值并且代替地可以被指定为被设定或者没有被设定。响应于所述加载指令而被加载到所述流控制寄存器 36 中的这些可编程的流控制值经由来自执行所述加载指令的数据引擎 6 内的多路复用器 52 和内部数据路径 54 而被提供给所述流控制寄存器 36。当处理是由所述主机处理器 4 引起时，则所述流控制寄存器 36 内的可编程的流控制值可以经由所述多路复用器 52 和外部任务地址路径 56 而被加载，由此由所述主机处理器 44 设定的可编程的流控制值被加载到所述数据引擎 6 的流控制机制 38 中。

[0056] 图 3 示意性地示出了流控制寄存器 36。所述流控制寄存器 36 存储所述下一任务地址 50 和所述跳转允许标志 48。所述第一形式的加载指令 incr_ldset 将所述目标地址加载到所述下一任务地址域 50 中并且将所述跳转允许标志 48 强加为值″ 1″，从而在所述当前程序指令序列完成时在所述数据引擎 6 内强迫程序流的跳转。所述第二形式的加载指令 incr_ld 也将所述目标地址加载到所述下一任务地址域 50 中，但是在这种情况下允许设定值或者未被设定的值被加载到所述跳转允许标志 48 中。

[0057] 图 4 示意性地示出了存储在所述主机存储器 12 内的程序以及指令紧密耦合的存储器 22。所述主机存储器 12 可以存储用于执行与在所述数据引擎 6 上调用函数的动作不相关的数据处理操作的主机通用程序 58。所述主机存储器 12 也存储被用于调度函数供所述数据引擎 6 执行的主机调度程序 60。

[0058] 被调度供所述数据引擎 6 执行的函数对应于被存储在所述指令紧密耦合的存储器 22 内的程序指令序列 62、64。这些程序指令序列 62、64 在它们正在执行以及在它们完成之前各自对应于如在上文中所提到的当前程序指令序列。所述指令紧密耦合的存储器 22 典型地将以函数库的形式存储这些函数中的许多，每个函数对应于用于控制所述数据引擎 6 以执行对应于期望的功能的处理操作的程序指令序列。也被存储在所述指令紧密耦合的存储器 22 内的是目标调度程序 66，其可以在被所述数据引擎 6 执行的程序指令序列被所述数据引擎 6 自身调用时被所述数据引擎 6 用于“自调度”。

[0059] 在图 4 的左手侧，相应的程序指令序列 62、64 的调用在其是由引起执行的主机处理器 4 执行的主机调度程序 40 时被示出。当以这种方式被调用的程序指令序列完成时，则所述数据指令被置于空闲状态，如由连接到点“DE 空闲”的线所示出的那样。由所述数据引擎 6 执行的目标调度程序 66 在其被发起时也从所述主机调度程序 60 被调用。此后，所述目标调度程序 66 被用于调度程序指令序列 62、64 供所述数据引擎 6 在“自调度”的基础上执行直到所述目标调度程序 66 自身完成。

[0060] 被调用的程序指令序列 62、64 可以是以在内联函数的库内提供的内联函数的形式。典型地使用带有运算对象的调入指令（例如 call[intrinsic0_pointer]）来调用这样的内联函数，所述运算对象是指向期望的内联函数的指针。

[0061] 在所述示出的指令紧密耦合的存储器 22 的右手侧示出的是指示通过在所述数据引擎 6 上执行的目标调度程序调用不同程序指令序列 62、64 的线。在这些程序指令序列 62、64 中的每一个完成时，程序执行点的返回被设置到所述目标调度程序 66 中以便这个目

标调度程序可以继续其调度操作。

[0062] 图5示意性地示出了被执行的当前程序指令序列62。这个当前程序指令序列包括指令I0、I1等等，其由所述数据引擎6按次序执行。更具体地，所述指令解码器18解码这些指令I0、I1等等并且产生控制所述处理电路系统20以执行所述指定的处理操作的一个或多个对应的控制信号24。在从所述当前程序指令序列62内的最后的指令起的预先确定的数量的指令处设置有以程序指令末尾EOP形式的标记指令。这个程序指令末尾指示在下三个程序指令 I_n+2、I_n+3 和 I_n+4 已被执行之后，所述数据引擎6可以例如触发两种类型的进一步动作之一。第一类型的动作是分出来执行所述目标调度程序66，从而执行因此在所述数据引擎6内继续。从所述流控制寄存器36读取的可编程的流控制值指定在所述目标调度程序66内将跳转到的所述目标程序指令的地址。可以被触发的第二行动过程是所述数据引擎进入空闲状态。这种空闲状态进入在所述可编程的流控制值内的跳转允许位48没有被设定时发生。

[0063] 图6示意性地示出了将由所述主机处理器4在所述数据引擎6上执行的函数的调用。所述主机处理器4执行主机调度程序60。这个主机调度程序60随后调用数据引擎函数并且发起由所述数据引擎6进行的处理。这种处理的调用可以通过经由图2所示的多路复用器52和外部任务地址路径将下一任务地址值加载到所述流控制寄存器36中来实现。因此，将要被调用的当前程序指令序列的开始地址交给所述数据引擎6。在数据引擎6上已调用所需要的处理之后，所述主机处理器4继续进一步的主机处理并且可以被控制（被编程）或者可以不被控制（不被编程）用于以响应所述数据引擎6完成被调用的任务的方式操作。

[0064] 在图6的数据引擎侧，所述数据引擎6是空闲的直到由所述主机处理器4调用的函数被发起。所述数据引擎6随后执行所述被调用的函数。朝着对应于所述被调用的函数的所述当前程序指令序列的末尾，程序指令的末尾会被遇到并且被执行。所述程序指令末尾的执行导致对所述跳转允许位48是否在存储在所述流控制寄存器36内的可编程的流控制值内被设定的检测。由于正被数据引擎6所执行的函数被所述主机处理器4调用，所预期的是所述数据引擎6在所述函数的执行完成之后将进入它的空闲状态并且所述跳转允许位48将不被设定。因此，在跳转允许位的被尝试的检测之后，这将被确定为没有被设定并且所述数据引擎6将进入所述空闲状态。

[0065] 图7是示意性地示出通过由所述数据引擎6本身所执行的目标调度程序66对程序指令序列62、64的执行进行调度的流程图。在步骤68处，所述目标调度程序66的执行被发起。所述目标调度程序66的执行本身可以由所述主机调度程序60引起，如图4中所示出的。

[0066] 在步骤70处，所述目标调度程序66等待直到有数据引擎函数调用。当数据引擎函数准备好被调用时，则步骤72开始有关的数据引擎函数（当前程序指令序列）的执行。朝着这个数据引擎函数的末尾，程序指令的末尾将在步骤74处被执行。所述程序指令末尾的执行触发在步骤76处从所述流控制寄存器36读取所述可编程的流控制值。步骤78确定所述跳转允许位48是否被设定。如果所述跳转允许位没有被设定，则步骤80将所述数据引擎切换到空闲状态。所述目标调度器68可以执行向所述流控制值36加载被设定为"0"的跳转允许位48的值，以便在下一程序指令序列62、64完成时所述数据引擎6将进入空闲

状态。然而，如果所述目标调度程序 66 要在当前程序指令序列 66、64 完成之后保持活动并且收回控制，则所述跳转允许位将被设定并且在步骤 78 处的确定之后，处理将前进到步骤 82，在步骤 82 处将执行到在下一任务地址域 58 内所指定的目标地址的跳转。在所示的例子中，这个跳转是回到所述目标调度程序 66 和步骤 70，在所述步骤 70 中所述目标调度程序 66 等待下一数据引擎函数调用。也可能的是函数可以被链接在一起并且在步骤 82 处被执行的跳转可以是到另一函数的开始。

[0067] 尽管已经参考附图在本文中详细地描述了本发明的说明性实施例，应当理解的是本发明不限于那些精确的实施例，并且本领域的技术人员可以在其中实现各种变化和修改而不背离由随附的权利要求所限定的本发明的范围和精神。

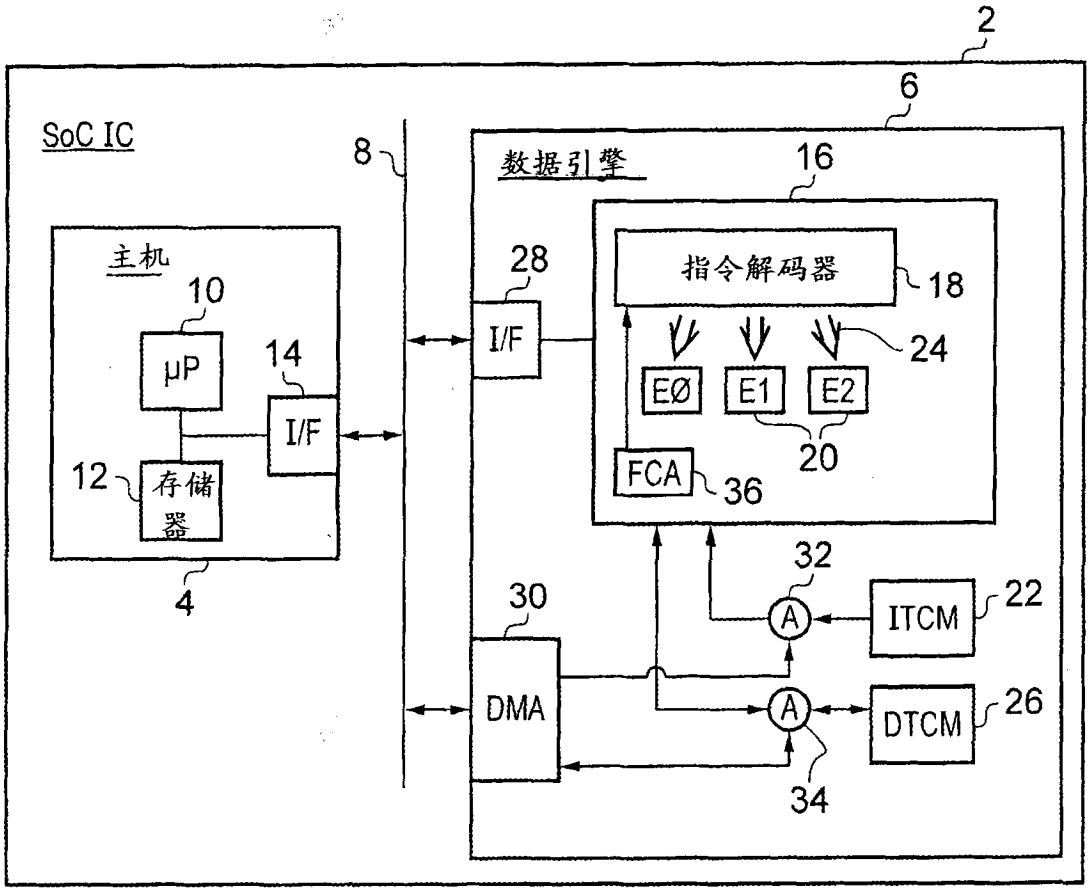

2
6
SoC IC
8
数据引擎
16
主机
10
μP
14
I/F
12
存储器
4
28
I/F
指令解码器
18
24
EØ
E1
E2
20
FCA
36
32
A
ITCM
22
30
DMA
A
DTCM
26
34

图 1

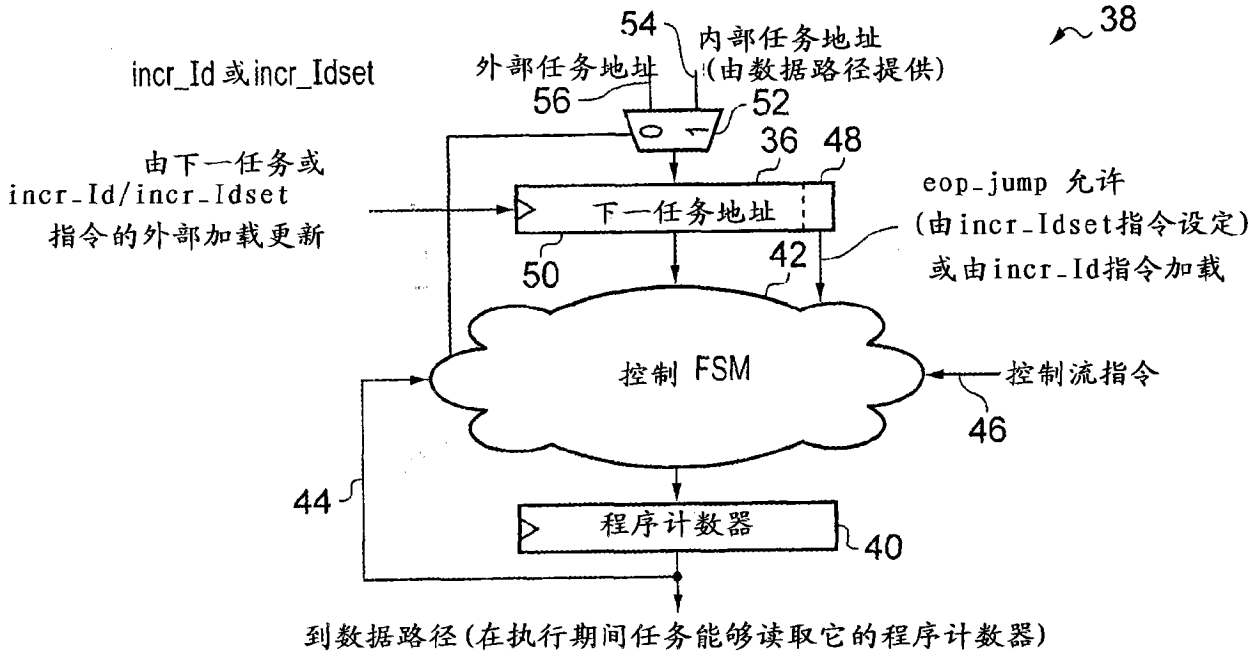

38
54
内部任务地址
(由数据路径提供)
外部任务地址
incr_Id 或 incr_Idset
56
52
36
48
由下一任务或
incr-Id/incr-Idset
指令的外部加载更新
下一任务地址
eop-jump 允许
(由incr-Idset指令设定)
或由incr-Id指令加载
50
42
控制 FSM
控制流指令
46
44
程序计数器
40
到数据路径(在执行期间任务能够读取它的程序计数器)

图 2

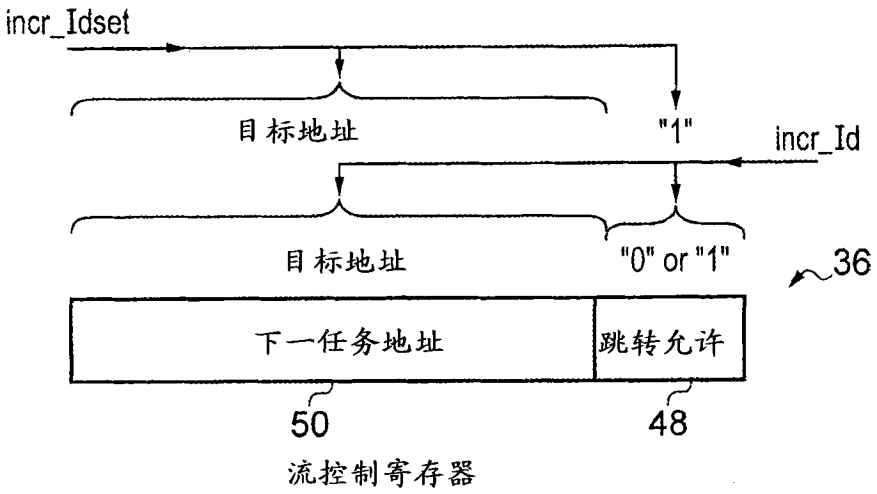

incr_Idset
目标地址
"1"
incr_Id
目标地址
"0" or "1"
36
下一任务地址
跳转允许
50
48
流控制寄存器

图 3

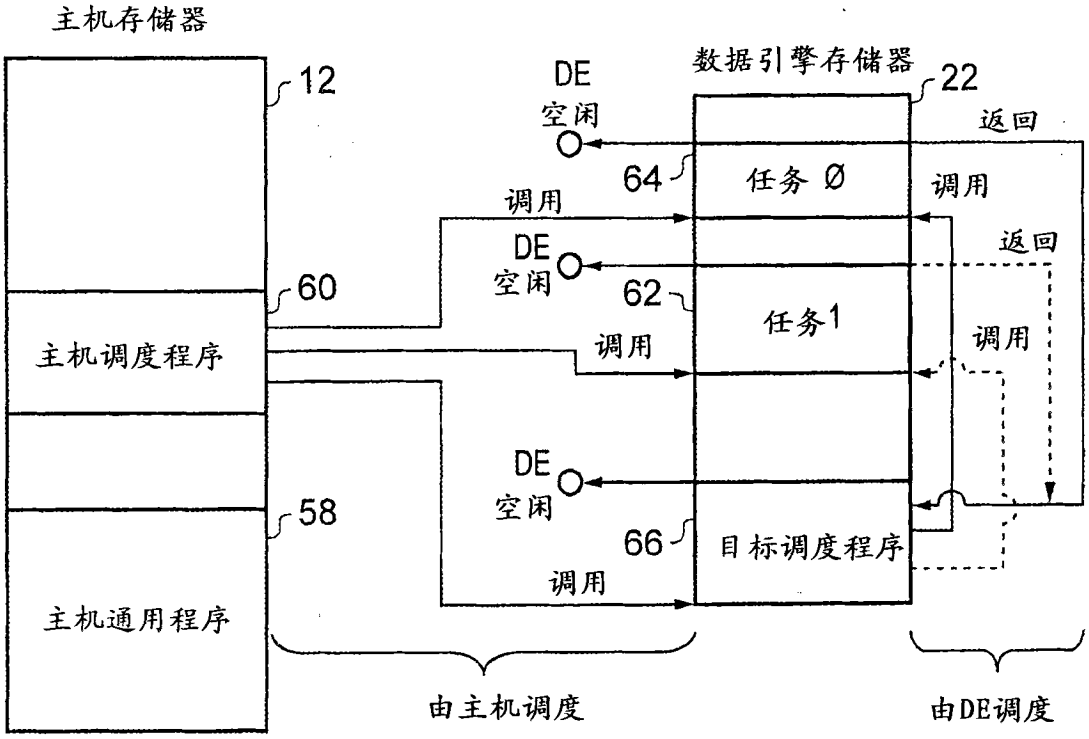

主机存储器
12
60
主机调度程序
58
主机通用程序
数据引擎存储器
22
DE
空闲
64
任务 Ø
返回
调用
调用
DE
空闲
62
任务1
返回
调用
调用
DE
空闲
66
目标调度程序
调用
由主机调度
由DE调度

图 4

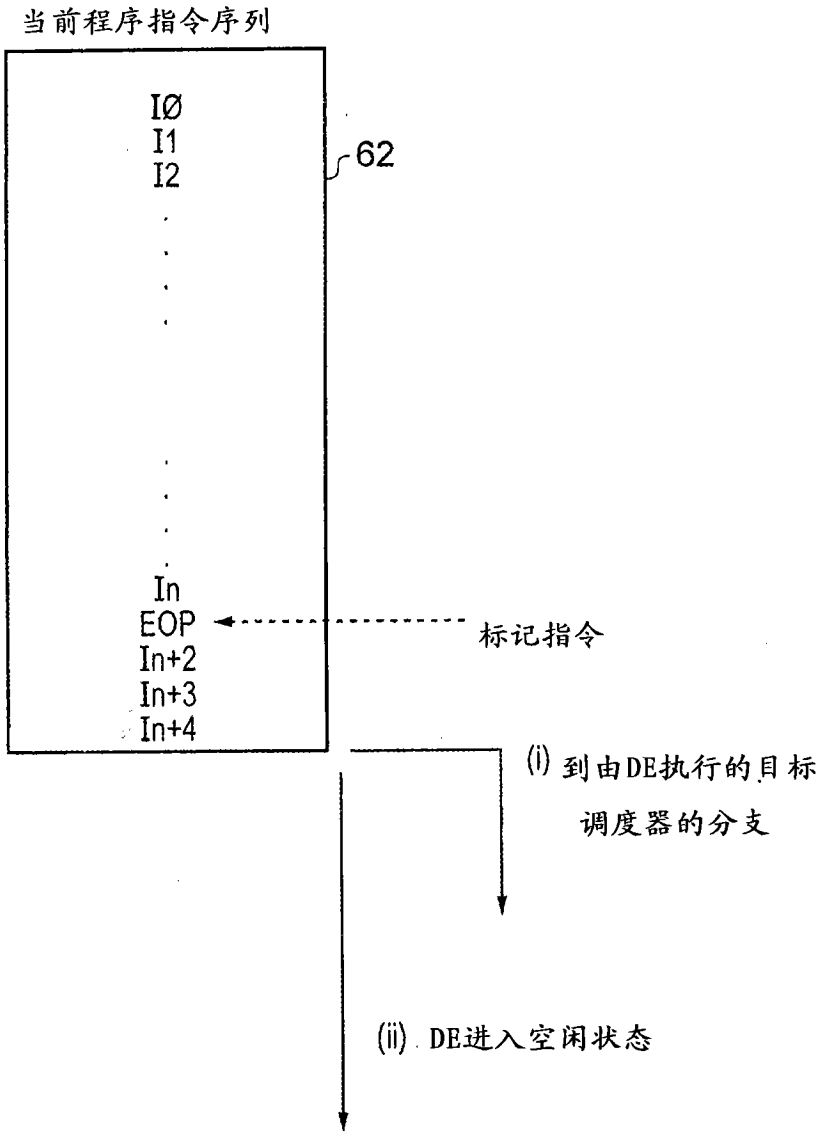

当前程序指令序列
IØ
I1
I2
62
In
EOP
In+2
In+3
In+4
标记指令
(i) 到由DE执行的目标调度器的分支
(ii) DE进入空闲状态

图 5

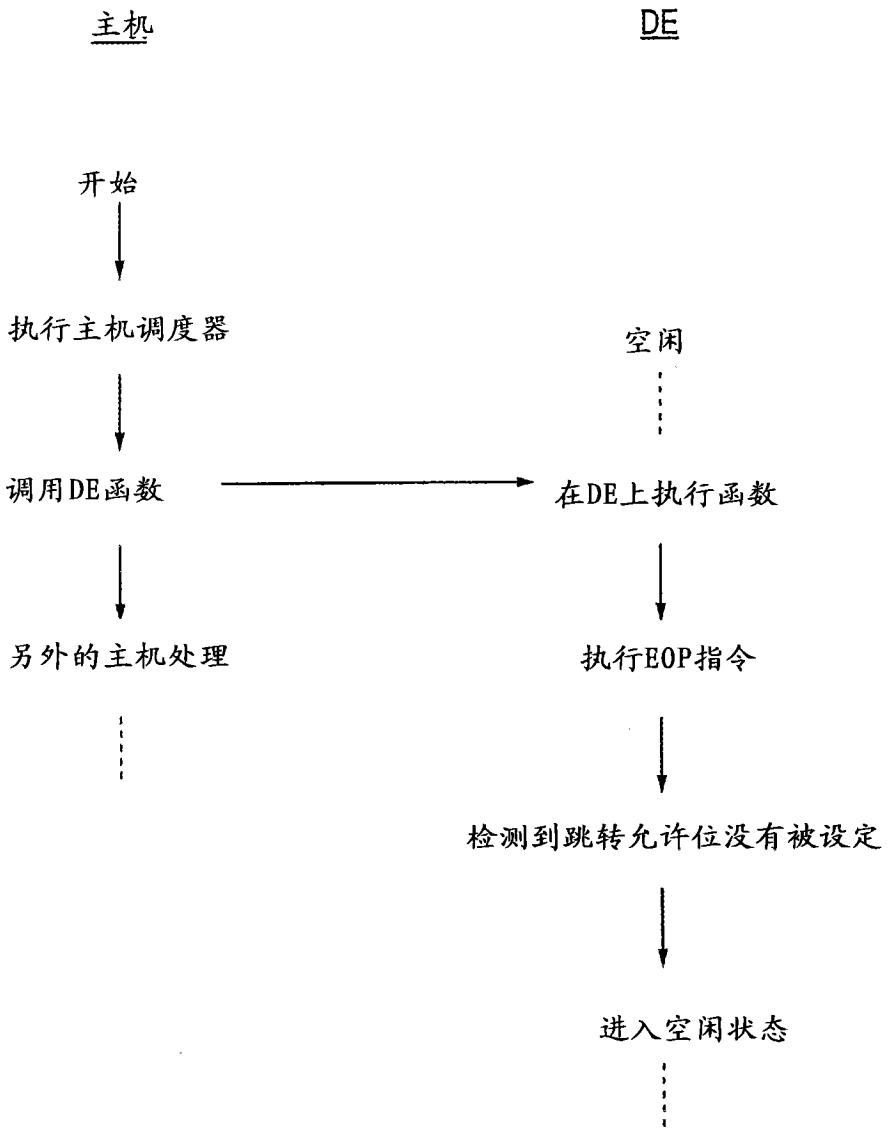

主机
DE
开始
执行主机调度器
调用DE函数
另外的主机处理
空闲
在DE上执行函数
执行EOP指令
检测到跳转允许位没有被设定
进入空闲状态

图 6

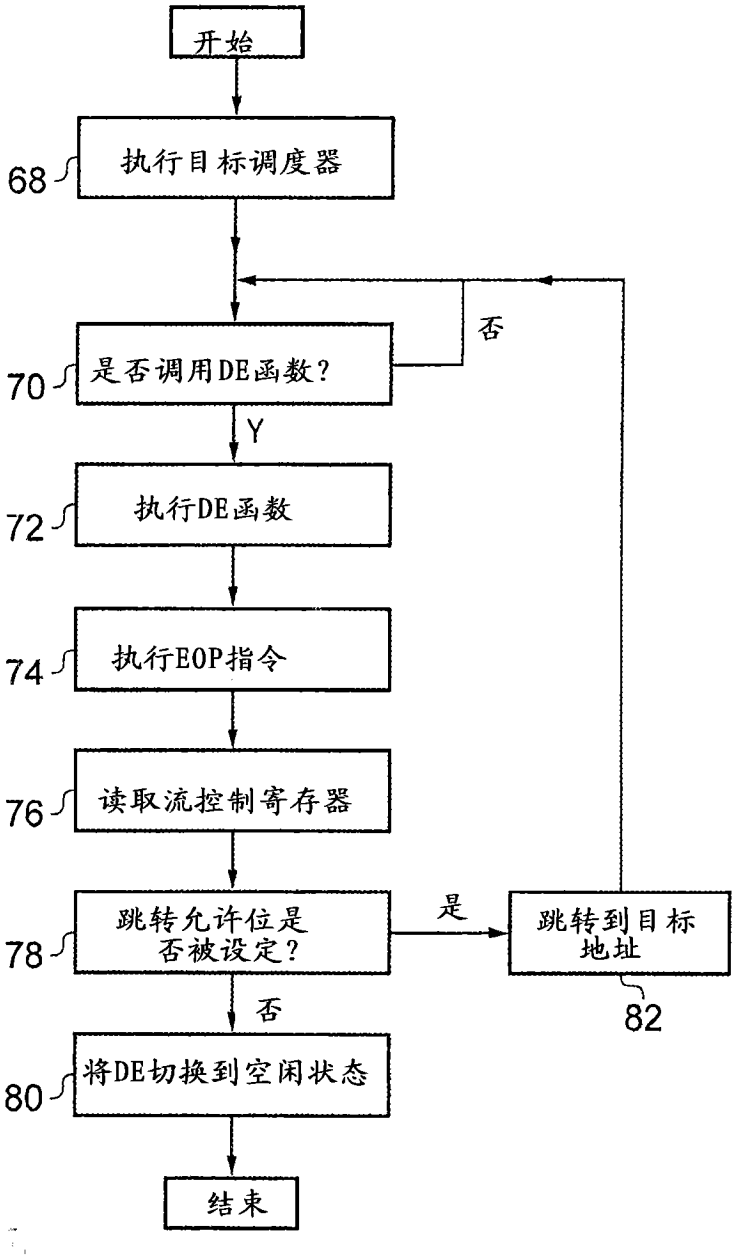

开始
68
执行目标调度器
70
是否调用DE函数?
否
Y
72
执行DE函数
74
执行EOP指令
76
读取流控制寄存器
78
跳转允许位是否被设定?
是
跳转到目标地址
82
否
80
将DE切换到空闲状态
结束

图 7