

[19] 中华人民共和国国家知识产权局

[51] Int. Cl.

G06F 9/38 (2006.01)

G06F 9/46 (2006.01)



[12] 发明专利说明书

专利号 ZL 200510007035.9

[45] 授权公告日 2008 年 7 月 2 日

[11] 授权公告号 CN 100399263C

[22] 申请日 2000.4.20

[21] 申请号 200510007035.9

分案原申请号 00809404.7

[30] 优先权

[32] 1999.4.29 [33] US [31] 09/302633

[73] 专利权人 英特尔公司

地址 美国加利福尼亚州

[72] 发明人 S·卡拉法蒂斯 A·B·凯克

R·D·菲施

[56] 参考文献

US5761522A 1998.6.2

US5553291A 1996.9.3

EP0346003A2 1989.12.13

EP0747816A2 1996.12.11

US5742782A 1998.4.21

审查员 覃冬梅

[74] 专利代理机构 中国专利代理(香港)有限公司

代理人 陈景峻

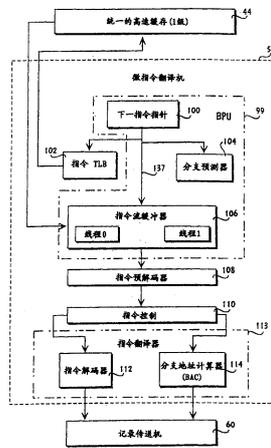
权利要求书 3 页 说明书 25 页 附图 17 页

[54] 发明名称

用于在一个多线程处理器内进行线程切换的方法和装置

[57] 摘要

一种用于在一个多线程处理器内部执行一个线程切换操作的方法。检测用于第一个线程的第一个预定量的指令信息从一个指令流缓存器的发送。响应于对第一个预定量指令信息发送的检测，就指令流缓存器的输出执行一个线程切换操作。因此开始了从指令流缓存器发送用于第二个线程的指令信息。依据用于已经处理(或发送用于处理)的一个特定线程的指令数据的数量，而不是依据一个任意定时机制，在线程之间分配对处理器资源的利用。



1. 一种用于在一个多线程处理器内进行线程切换操作的方法，包括：

在一个多线程处理器内检测第一线程的一个预定量指令信息从一个指令信息源的发送；

检测将被从所述指令信息源发送的所述第一线程的指令信息内的分支指令；以及

对第一线程的预定量的指令信息发送的检测做出响应，并对所述分支指令的检测做出响应，从所述指令信息源开始发送第二线程的指令信息。

2. 如权利要求 1 所述的方法，其中对第一线程的预定量指令信息的发送的检测包括保持对从所述指令信息源发送的、第一线程的指令信息的非连续数量的计数，以及确定何时所述指令信息的非连续数量计数大于第一预定阈值。

3. 如权利要求 2 所述的方法，其中所述检测包括在一个预定阈值和一个计数器的内容之间执行一个比较操作，其中所述计数器保持从指令信息源发送的第一线程的非连续数量指令信息的计数。

4. 如权利要求 1 所述的方法，其中指令信息源被进行分区为第一分区，以便提供第一线程的指令信息，和第二分区，以提供第二线程的指令信息，以及其中开始发送第二线程的指令信息包括操作线程选择逻辑，以便选择从第二分区发送的指令信息。

5. 如权利要求 1 所述的方法，包括在检测所述分支指令之前检测第一线程的另一个预定量指令信息的发送，对另一个预定量的检测包括对来自所述指令信息源的、第一线程的指令信息的非连续数量的发送进行计数，以及确定何时所述指令信息的非连续数量计数大于第四预定阈值。

6. 如权利要求 1 所述的方法，其中所述指令信息源包括一个指令流缓冲器。

7. 如权利要求 1 所述的方法，其中所述指令信息包括宏指令信息，和所述指令信息是从所述指令信息源发送到一个指令解码器。

8. 如权利要求 7 所述的方法，其中所述指令解码器解码从指令信息源发送的指令信息，而不执行在第一线程的指令信息和第二线程的指令信息之间的任何识别。

9. 一种用于在一个多线程处理器内进行线程切换操作的设备，包括：

检测逻辑，检测从在多线程处理器中的一个指令信息源发送的第一线程的一个预定量的指令信息的排序，并且检测将被从所述指令信息源发送的所述第一线程的指令信息内的分支指令；以及

选择逻辑，与所述检测逻辑相连，对所述指令信息源发送的第一线程的预定量的指令信息的排序的检测做出响应，开始对从所述指令信息源发送的第二线程的指令信息排序，并且通过所述检测逻辑检测所述分支指令。

10. 如权利要求 9 所述的设备，其中所述检测逻辑包括一个计数器，保持从指令信息源发送的第一线程的非连续数量指令信息的计数。

11. 如权利要求 10 所述的设备，其中所述检测逻辑包括一个比较器，所述比较器检测第一线程的非连续数量指令信息的计数何时超过第一预定阈值。

12. 如权利要求 9 所述的设备，其中所述指令信息源被分区为第一分区，以提供第一线程的指令信息，和第二分区，以提供第二线程的指令信息，以及其中选择逻辑通过从排序的第二分区选择指令信息开始对第二线程的指令信息排序。

13. 如权利要求 9 所述的设备，其中检测逻辑检测在检测所述分支指令之前第一线程的另一个预定量指令信息是否已经从所述指令信息源排序，并且其中检测逻辑包括一个计数器，对来自所述指令信息源的第一线程的指令信息的非连续数量的排序进行计数。

14. 如权利要求 9 所述的设备，其中所述指令信息源包括一个指

令流缓冲器。

15. 如权利要求 9 所述的设备，其中所述指令信息包括宏指令信息，和所述指令信息是从所述指令信息源排序到一个指令解码器。

16. 如权利要求 15 所述的设备，其中所述指令解码器解码从指令信息源发送的指令信息，而不执行在第一线程的指令信息和第二线程的指令信息之间的任何识别。

17. 一种在一个多线程处理器内执行一个线程切换操作的设备，所述设备包括：

检测装置，检测从在多线程处理器中的一个指令信息源的第一线程的一个预定量的指令信息的发送，并且检测将被从所述指令信息源发送的所述第一线程的指令信息内的分支指令；以及

选择装置，与所述检测装置相连，对来自所述指令信息源的第一线程的预定量的指令信息的发送的检测做出响应，开始从所述指令信息源发送第二线程的指令信息，并且通过所述检测装置检测所述分支指令。

18. 一种多线程处理器，包括：

处理器流水线，包括指令信息源；

检测器，检测从所述指令信息源发送的第一线程的一个预定量的指令信息的排序，并且检测将被从所述指令信息源发送的所述第一线程的指令信息内的分支指令；以及

选择器，与所述检测器相连，对所述指令信息源发送的第一线程的预定量的指令信息的排序的检测做出响应，并且对通过所述检测器检测所述分支指令做出响应，开始对从所述指令信息源发送的第二线程的指令信息排序。

用于在一个多线程处理器内进行线程切换的方法和装置

技术领域

本发明通常涉及多线程处理器领域，尤其是涉及一种用于在一个多线程处理器内执行上下文（或线程）切换的方法及其装置。

背景技术

多线程处理器设计近来已经被认为是一种用于提高处理器性能的日益有吸引力的选择。此外，在一个处理器内的多线程处理为更有效地利用各种处理器资源提供了可能性，尤其是为更加有效地利用在一个处理器内的执行逻辑提供了可能性。特别地，通过将多个线程传送到一个处理器的执行逻辑，原先由于在一个特定线程处理过程中的停止或其它延迟而造成空闲的时钟周期可以被用来服务另外一个线程。在一个特定线程处理过程中的一个停止可以由一个处理器流水线内的许多事件引起。例如，用于一条包含在一个线程内的指令的一个高速缓存故障或一个分支错误预测（即一个长等待时间的操作），通常会导致有关线程处理过程的停止。长等待时间的操作在执行逻辑效率上的消极影响由于近来执行逻辑吞吐量的增加而有所加剧，其中该执行逻辑吞吐量在存储器访问和检索率方面有超前的增长。

鉴于由许多普及的操作系统、诸如 Windows NT 和 UNIX 操作系统提供给这种多线程应用的支持，多线程的计算机应用也变得日益普及。多线程计算机应用在中媒体环境中尤其有效。

依据在有关处理器内使用的线程交错或切换方案，多线程处理器可以被广泛地分为两类（即精细或粗糙设计）。精细的多线程设计在一个处理器内支持多个有效线程，并且通常在一个周期接一个周期的基础上交错两个不同的线程。粗糙的多线程设计通常在发生某些长等待时间的事件、诸如一个高速缓存故障时交错不同线程的指令。在以下文献中讨论了一种粗糙的多线程设计：R.；Johnson, R.； et al.；

“Evaluation of Multithreaded Uniprocessors for Commercial Application Environments”, The 23rd Annual International Symposium on Computer Architecture (第 23 届计算机体系结构国际年会论文集), pp. 203-212, May 1996. 而在以下文献中进一步讨论

了精细和粗糙设计之间的区别: Laudon, j; Gupta, A, "Architectural and Implementation Tradeoffs in the Design of Multiple-Context Processors", Multithreaded Computer Architectures (多线程结构): A Summary of the State of the Art (技术状态概述), edited by R. A. Iannucci et al., (由 Iannucci 等编辑), pp.167-200, Kluwer Academic Publishers, Norwell, Massachusetts, 1994. Laudon 进一步提出了一个交错方案, 它将一个精细设计一个周期接一个周期的切换和一个粗糙设计的完全流水线联锁(或锁定方案)结合起来。为此, Laudon 提出了一条“补偿”指令, 使一个特定线程(或上下文)在一个特定数目的周期内无法使用。这样一条“补偿”指令可以依据预定事件、诸如一个高速缓存故障的发生来发布。这样, Laudon 通过简单地使这些线程中的一个无法使用而避免了必须执行一个实际的线程切换。

发明内容

依据本发明, 提供了一种用于在一个多线程处理器内执行一个线程切换操作的方法, 该方法包括:

在一个多线程处理器内检测用于第一个线程的第一个预定量的指令信息预定量的指令信息从一个指令信息源的发送;

确定用于第二个线程的第二个预定量的指令信息预定量的指令信息是否可用于从该指令信息源的发送; 以及

对用于第一个线程的第一个预定量的指令信息发送的检测做出响应, 如果第二个线程的第二预定量的指令信息可用于从该指令信息源的发送, 则从该指令信息源开始发送用于第二个线程的指令信息。

本发明还提供了用于在一个多线程处理器内执行一个线程切换操作的设备, 该设备包括:

检测逻辑, 用于检测从该多线程处理器的一个指令信息源发送用于第一个线程的第一个预定量的指令信息, 以及检测用于第二个线程的第二个预定量的指令信息可用于从该指令信息源的发送; 以及

选择逻辑, 连接到该检测逻辑, 对该检测逻辑对第一个预定量的指令信息的检测作出响应, 并且当该检测逻辑检测到用于第二个线程的第二个预定量的指令信息可用于从该指令信息源发送时, 由该选择逻辑指示该指令信息源开始发送用于第二个线程的指令信息,

其中该检测逻辑是用来检测用于第二个线程的第二个预定量的指令信息是否可用于从该指令信息源的发送,和如果第二个线程的第二个预定量的指令信息可用于从该指令信息源的发送,则该选择逻辑从该指令信息源开始发送用于第二个线程的指令信息。

本发明还提供了一种多线程处理器,包括:

一个处理器流水线,包括指令信息源;

一个检测器,用于检测从该多线程处理器的一个指令信息源发送用于第一个线程的第一个预定量的指令信息,以及检测用于第二个线程的第二个预定量的指令信息可用于从该指令信息源的发送;以及

一个选择器,连接到该检测器,对该检测器对第一个预定量的指令信息的检测作出响应,并且当该检测器检测到用于第二个线程的第二个预定量的指令信息可用于从该指令信息源发送时,由该选择器指示该指令信息源开始发送用于第二个线程的指令信息,

其中检测器检测用于第二个线程的第二个预定量的指令信息是否可用于从指令信息源的发送,如果第二个线程的第二个预定量的指令信息可用于从该指令信息源的发送,则选择器从该指令信息源开始发送用于第二个线程的指令信息。

通过附图和随后的详细说明,本发明的其它特征将是显而易见的。

附图说明

本发明通过附图进行了举例说明,但是不局限于此,且附图中相似的标记表示相似的单元,其中:

图1的框图说明了在其内部可以实现本发明的一个处理器的一个示范流水线;

图2的框图以一个通用多线程微处理器的形式说明了在其内部可以实现本发明的一个处理器的一个示范实施例;

图3的框图为一个包含在如图2所示的通用微处理器内部的微指令翻译机的示例的体系结构提供了更多细节;

图4的框图说明了一个示范性多线程微处理器的选定部件,而且特别地描述了由于为容纳多个线程而被逻辑分区提供各种提供缓存能力的功能单元;

图5的框图说明了关于依据本发明一个实施例的一个示范性指令

流缓存器的结构和体系结构的更多细节；

图 6 的框图说明了依据本发明一个示范实施例的线程切换控制逻辑的逻辑元件；

图 7 的流程图说明了依据本发明一个示范实施例、用于当在一个多线程处理器内的多个线程空闲时确定一个开始线程的方法；

图 8 的流程图说明了依据本发明示范实施例、用于当从一个指令源发送当前线程的分支指令时在一个多线程处理器内部执行线程切换操作的方法；

图 9 的流程图说明了依据本发明一个示范实施例、用于当发生一个长等待时间的停止时在一个多线程处理器内执行线程切换操作的方法；

图 10 的流程图说明了依据本发明一个示范实施例、用于当发生一个内部流水线清除时在一个多线程处理器内执行线程切换操作的方法；

图 11 的流程图说明了依据本发明示范实施例、用于当出现关于一个特定线程的“无数据流”状态时在一个多线程处理器内执行线程切换操作的方法；

图 12 的流程图说明了依据本发明一个示范实施例、用于当从一个指令流缓存器向一个指令预解码器发送用于一个特定线程的一个预定量指令信息时在一个多线程处理器内执行线程切换操作的方法；

图 13 是一个说明了依据本发明示范实施例、用于当发生一个外部流水线清除时在一个多线程处理器内执行线程切换操作的方法的流程图；

图 14 是一个说明了依据本发明示范实施例、用于当在一个用于当前线程的指令流内检测到一个插入的流程时在一个多线程处理器内部执行线程切换操作的方法的流程图；

图 15A 和 15B 是显示了依据本发明、如图 6 中框图形式所示的线程切换控制逻辑的相应示范实施例的结构简图；

图 16 是依据本发明的示例、用于将一个流程标记（或插入流）插入到一个指令流里的逻辑简图。

具体实施方式

描述了一种用于在一个多线程处理器内进行线程切换的方法和装置。在下面的描述中，为说明起见，阐述了大量具体细节以便提供对本发明的彻底理解。然而，对本领域技术人员来说，显然本发明没有这些具体细节也可以实现。

就该说明书来说，术语“发送”应该包括数据从一个存储单元或功能单元的实际发送或传播，以及准备该数据的实际发送或传播的那些步骤。例如，术语“发送”应该包含要从一个存储单元或缓存器传播的数据以一个特定次序或准备状态的放置。

处理器流水线

图 1 是说明了在其内部可以实现本发明的处理器流水线 10 的一个示范实施例的高级框图。就该说明书来说，术语“处理器”应该指的是任何能够执行一个指令序列（例如，宏指令或微指令）的机器，并且应该包含但不局限于：通用微处理器、专用微处理器、图形控制器、音频控制器、多媒体控制器和微控制器。此外，术语“处理器”还应该指的是复杂指令集计算机(CISC)、精简指令集计算机(RISC)、或超长指令字(VLIW)处理器。该流水线 10 包含多个进程阶段，从一个读取进程阶段 12 开始，在那儿读出指令（例如宏指令）并将其送入该流水线 10 中。例如，一条宏指令可以从一个与该处理器集成或与之密切相关的高速缓存存储器中读出，或是经由一条处理器总线从一个外部主存储器中读出。这些宏指令从该读取进程阶段 12 被传播到一个解码进程阶段 14，在那儿宏指令被译成适于在该处理器内执行的微指令（也称为“微码”）。然后这些微指令被向下传送到一个分配进程阶段 16，在那儿依据可用性和需要将处理器资源分配给不同的微指令。然后这些微指令在一个退出进程阶段 20 中退出之前在一个执行阶段 18 执行。

微处理器体系结构

图 2 的框图以一个通用微处理器 30 的形式说明了在其内部可以实现本发明的一个处理器的示范实施例。该微处理器 30 作为一个多线程(MT)处理器而描述如下，而且因此能同时处理多个指令线程(或上下文)。然而，以下在说明书中提供的多个示教没有特指一个多线程处理器，而且在一个单线程处理器中也可以发现该应用。在一个示例中，该微处理器 30 可以包含一个能够执行 Intel 体系结构指令集

的 Intel (英特尔) 体系结构 (IA) 微处理器。

微处理器 30 包含一个有序前端和一个无序后端。该有序前端包含一个总线接口单元 32, 起到微处理器 30 和一个在其内部可以使用该微处理器 30 的计算机系统的其它元件 (例如主存储器) 之间的管道作用。为此, 该总线接口单元 32 将该微处理器 30 连接到一条处理器总线 (未显示), 经由该总线可以在微处理器 30 接收、并从微处理器 30 传播数据和控制信息。该总线接口单元 32 包含前侧总线 (FSB) 逻辑 34, 由它控制经由该处理器总线的通信。此外该总线接口单元 32 还包含一个总线队列 36, 它为经由该处理器总线的通信提供缓存功能。该总线接口单元 32 从一个存储器执行单元 42 接收总线请求 38, 并向其发送探听或总线返回, 其中该存储器执行单元 42 提供了在微处理器 30 内的一个本地存储器性能。该存储器执行单元 42 包含一个统一的数据和指令高速缓存 44、一个数据翻译后备缓存器 (TLB) 46、和存储器排序逻辑 48。该存储器执行单元 42 从一个微指令翻译机 54 接收指令读取请求 50, 并向其传送未处理的指令 52 (即编码的宏指令), 其中该微指令翻译机 54 将接收的宏指令翻译成为一组相应的微指令。在下面提供了关于该微指令翻译机 54 的更多细节。

解码的指令 (即微指令) 从该微指令翻译机 54 发送到一个记录传送机 60。该记录传送机 60 包含一个记录高速缓存 62、一个记录分支预测器 (BTB) 64、一个微码定序器 66 和一个微码 (uop) 队列 68。该记录传送机 60 起一个微指令高速缓存的作用, 而且是用于一个下游的执行单元 70 的微指令的主要来源。通过在该处理器流水线内提供一个微指令高速缓存功能, 该记录传送机 60、特别是该记录高速缓存 62 允许由微指令翻译机 54 所做的翻译工作被平衡以便提供一个相对高的微指令带宽。在一个示范实施例中, 该记录高速缓存 62 可以包含一个 256 组的 8 路组相关存储器。在本示例中, 术语“记录”可以指在记录高速缓存 62 的条目内存储的微指令序列, 其中每个条目包含指向包含该记录在内的先前和正在处理的微指令的指针。这样, 该记录高速缓存 62 便于高性能的排序, 体现在: 在完成当前访问之前为获得随后一条微指令而要访问的下一个条目的地址是已知的。记录可以被看作是彼此由记录头区别、而且一旦遇到一个间接分支或达

到许多给出的阈值条件中的一个就结束的指令“块”，这些阈值条件诸如在单个记录中可以容纳的条件分支的数目或是可以包含一个记录的全体微指令的最大数目。

该记录高速缓存分支预测器 64 提供了关于在该记录高速缓存 62 内的记录的本地分支预测。该记录高速缓存 62 和微码定序器 66 向微码队列 68 提供微指令，然后从那儿将这些微指令送到一个无序的执行群集。因此，该微处理器 30 可以被看作是具有一个包含总线接口单元 32、存储器执行单元 42、微指令翻译机 54 和记录传送机 60 的有序前端、以及一个无序后端，该无序后端将在下面进行详细描述。

从微码队列 68 发送的微指令被接收到一个无序群集 71 中，其中该无序群集 71 包含一个调度器 72、一个寄存器重命名器 74、一个分配器 76、一个重新排序缓存器 78 以及一个重调队列 80。调度器 72 包含一组预定位置，而且操作用以调度和发送微指令用于由执行单元 70 执行。寄存器重命名器 74 就隐藏的整数和浮点寄存器（可以用来代替八个通用寄存器中的任何一个或八个浮点寄存器中的任何一个，在那儿微处理器 30 执行 Intel 体系结构指令集）执行寄存器重命名功能。分配器 76 依据可用性和需要操作用以向微指令分配执行单元 70 和群集 71 的资源。在没有足够资源可以用来处理一条微指令的情况下，该分配器 76 负责确定一个停止信号 82，该信号通过记录传送机 60 被传送到微指令翻译机 54，如在 58 所示。其源字段由寄存器重命名器 74 进行调整的微指令以严格的程序次序被放置在一个重新排序缓存器 78 中。然后，当在该重新排序缓存器 78 内的微指令已经完成了执行、并准备好退出时，将它们从该重新排序缓存器 78 中删除。重调队列 80 将要重调的微指令传播到执行单元 70。

如图所示，执行单元 70 包含一个浮点执行机 84、一个整数执行机 86、以及一个 0 级数据高速缓存 88。在一个其中微处理器 30 执行 Intel 体系结构指令集的示范实施例中，浮点执行机 84 还可以执行 MMX® 指令。

微指令翻译机

图 3 的框图为微指令翻译机 54 的一个示范实施例的体系结构提

供了更多细节。微指令翻译机 54 实际上操作作为一个记录高速缓存“故障处理器”，体现在：如果发生一个记录高速缓存故障它就操作用于向记录高速缓存 62 传送微指令。为此，如果发生一个记录高速缓存故障，则该微指令翻译机 54 起到提供读取进程阶段 12 和解码进程阶段 14 的作用。微指令翻译工具 54 包含一个下一指令指针 (NIP) 100、一个指令翻译后备缓存器 (TLB) 102、一个分支预测器 104、一个指令流缓存器 106、一个指令预解码器 108、指令控制逻辑 110、一个指令解码器 112、以及一个分支地址计算器 114。下一指令指针 100、TLB 102、分支预测器 104 和指令流缓存器 106 一起构成了一个分支预测单元 (BPU) 99。指令解码器 112 和分支地址计算器 114 一起构成了一个指令翻译 (IX) 单元 113。

下一指令指针 100 向统一高速缓存 44 发布下一指令请求。在其中微处理器 30 包含一个能够处理两个线程的多线程微处理器的这个示例中，下一指令指针 100 可以包含一个多路复用器 (MUX) (未显示)，它在与第一或者第二线程有关的指令指针之间进行选择，用于包含在在此发布的该下一指令请求内。在一个实施例中，假定已经请求了用于两个线程的指令，而且用于两个线程的指令流缓存器 106 的资源没有用完，则下一指令指针 100 将在一个周期接一个周期 (“往复转换工作”) 的基础上交错用于第一和第二线程的下一指令请求。取决于初始请求地址是否位于一个 32 字节或者 64 字节校准行的上半部，下一指令指针请求可以是 16、32 或 64 字节。下一指令指针 100 可以由分支预测器 104、分支地址计算器 114、或记录高速缓存 62 利用作为最高优先权的重定向请求的一个记录高速缓存故障请求进行重定向。

当下一指令指针 100 向统一高速缓存 44 做出一个指令请求时，它产生一个与该指令请求有关的 2 位的“请求标识符”，并且该标识符用作该相关指令请求的一个“标记”。当对一个指令请求做出响应返回数据时，该统一高速缓存 44 和该数据一起返回下列标记或标识符：

1. 由下一指令指针 100 提供的“请求标识符”；
2. 一个用以识别返回组块的三位的“组块标识符”；以及
3. 一个用以识别返回的数据所属的那个线程的“线程标识符”。

下一指令请求从该下一指令指针 100 传播到指令 TLB 102, 在那儿执行一个地址查找操作, 并向该统一高速缓存 44 传送一个物理地址。统一高速缓存 44 向指令流缓存器 106 传送一条相应的宏指令。每个下一指令请求也直接从该下一指令指针 100 传播到指令流缓存器 106, 以便允许该指令流缓存器 106 识别从该统一高速缓存 44 接收的一条宏指令所属的那个线程。然后将来自于第一和第二线程的宏指令从该指令流缓存器 106 发布到指令预解码器 108, 由它执行关于接收的(宏指令的)指令流的多个长度计算和字节标记操作。特别地, 指令预解码器 108 还产生一系列的字节标记矢量, 用于区别在传播到该指令控制逻辑 110 的指令流内的宏指令。然后指令控制逻辑 110 利用这些字节标记矢量以操纵离散的宏指令进入指令解码器 112 中进行解码。此外还从指令控制逻辑 110 传播宏指令到分支地址计算器 114 用于分支地址计算。然后微指令从该指令解码器 112 传送到记录传送机 60。

多线程实现

在如图 2 所示的微处理器 30 的示范例实施例中, 会注意到存在资源的有限重复。为了在一个其中存在功能单元有限重复的处理器内提供多线程性能, 就必须在线程之间实现某些程度的资源共享。应当意识到: 使用的资源共享方案取决于该处理器能同时处理的线程数目。因为在一个处理器内的功能单元通常提供了某些缓存(或存储)功能和传播功能, 所以资源共享的问题可以被看作是包含(1)存储和(2)处理/传播带宽共享元件。例如, 在一个支持同时处理两个线程的处理器中, 在不同功能单元内的缓存资源可以在两个线程之间在逻辑上进行分区和分配。类似地, 由一个用于在两个功能单元之间传播信息的路径提供的带宽必须在这两个线程之间被分开和分配。由于这些资源共享问题可以在一个处理器流水线内的多个存储单元产生, 因此可以根据该特定存储单元的命令和特性, 在这些不同的存储单元使用不同的资源共享方案。应当明白, 考虑到各不相同的功能和操作特性, 不同的资源共享方案可以适于不同的存储单元。

图 4 的框图说明了如图 3 所示的微处理器 30 的选定元件, 而且描述了由于为容纳两个线程(即线程 0 和线程 1)而被逻辑分区从而提供缓存能力的各种功能单元。一个功能单元的缓存(或存储)功能

的用于两个线程的逻辑分区，可以通过分配在一个缓存资源内的第一组预定条目给第一个线程、并分配该缓存资源内的第二组预定条目给第二个线程来实现。特别地，这可以通过提供两对读和写指针来实现，其中第一对读和写指针与第一个线程有关，而第二对读和写指针与第二个线程有关。第一组读和写指针可以局限于一个缓存资源内部的第一个预定数目条目，而第二组读和写指针可以局限于在同一缓存资源内的第二个预定数目条目。在该示范实施例中，指令流缓存器 106、记录高速缓存 62、和一个指令队列 103 中的每一个都提供了存储能力，该能力在第一个和第二个线程之间进行逻辑分区。

以下将更为详细地讨论在一个处理器内功能单元之间的一条路径的带宽分配问题。

指令流缓存器

参见图 3，指令流缓存器 106 传送一个指令流进入单指令解码器 112（即没有解码器重复）中。为了有效利用这个单解码资源，期望确保在指令流缓存器 106 和指令解码器 112 之间的路径带宽，因此该指令解码器 112 的“解码带宽”被以一种有效方式分开并分配。因此本发明提出了一种方法，就从指令流缓存器 106 传送的两个指令流而论，通过该方法实现了线程切换。这种线程切换方法确定用于这两个线程中每一个的指令以何种速率被送到指令解码器 112。本发明寻求实现一种线程切换算法，它试图实现下列线程切换目标：

- 只有当另外一个线程有数据可以向下传送（例如从指令流缓存器 106）时，才执行一个切换线程；
 - 当当前一个线程正在被错误地执行时（例如，当当前一个线程接收一个后端清除时），切换线程；
 - 切换线程以便在执行该线程切换之前保证在下游流水线内的前进进程（例如，通过确保当前线程的至少一个完全指令，将在执行从当前线程到目标线程的线程切换之前，从指令流缓存器 106 发出）；
 - 防止一个线程使其它线程空闲（例如，通过只有一旦用于当前线程的预定数量指令信息已经沿该处理器流水线向下传播、而没有任何其它线程切换机制已经被启用时，才执行一个线程切换操作）；
- 以及

- 分摊线程切换损失（例如，通过在寻找一个可能触发线程切换操作的采取的分支之前发出一个预定最低数量的指令信息）。

下面将参考一个示范实施例对本发明进行描述，该实施例用来在指令流缓存器 106 内部缓存的线程数据之间进行选择、用于沿单一路径向下游向指令解码器 112 传播该数据。然而，容易理解，为了线程切换或交错，本发明的这些示教可以在一个处理器流水线内部的任何位置使用。图 5 的框图显示了关于该指令流缓存器 106 的结构和体系结构的更多细节。特别地，如图所示，该指令流缓存器 106 包含一个具有 4 个条目 120（条目 0-条目 3）的存储阵列，它被逻辑上分区成为用于存储第一个线程（线程 0）指令的第一分区 122、以及用于存储第二个线程（线程 1）指令的第二分区 124。在一个示范实施例中，每一条目 120 都能容纳 8 组块的信息，因此该指令流缓存器 106 为每个线程提供了 16 组块的数据高速缓存。

此外该指令流缓存器 106 还包含一个分配模块 125，用于控制将数据写入到逻辑分区 122 和 124、以及从该逻辑分区 122 和 124 读取数据。

每一分区 122 和 124 都有各自的、包含在该分配模块 125 内部的分配逻辑 126。每个分配逻辑 126 都包含一个写入指针 128 和一个读取指针 130，它们每一个都指向在相关分区 122 或 124 内部的一个条目 120。每个分配逻辑 126 从统一高速缓存 44 接收一个指令流，并依据线程规范将该指令流的组块（即 8 字节）写入到第一分区 122 或第二分区 124 里。特别地，通过检查与每个组块数据有关的“线程标识符”，每个分配逻辑 126 能够确定将从该统一高速缓存 44 接收的数据写入到哪个分区。

每一个分配逻辑 126 从各自的分区 122 或 124 输出数据到一个 MUX 132，由它根据由线程切换控制逻辑 136 确定的一个线程选择信号 134 在分区 122 和 124 的输出之间进行选择。因此，在本发明的该示范实施例中，由线程切换控制逻辑 136 实现线程交错或切换方案。此外，线程切换控制逻辑 136 还经由线路 138 和 140 监控每一个分区 122 或 124 的输出，以便能够确定何时将一个组块数据从分区 122 或 124 中的任何一个发送，并识别该组块是从这些分区中的哪一个发送的。然后，将由 MUX 132 选择用于输出的该组块数据在处理器流水线

内部向下传播到指令预解码器 108，并最终传送到指令解码器 122 用于解码。

图 6 的框图描述了线程切换控制逻辑 136 的逻辑元件。特别地，如图所示，该线程切换控制逻辑 136 包含在空闲上切换的逻辑 150、在分支上切换的逻辑 152、长等待时间停止逻辑 154、内部清除逻辑 156、无数据流逻辑 158、强制线程改变逻辑 160、外部清除逻辑 162、以及插入流逻辑 164。虽然如图 6 所示每一个逻辑 150 - 164 中是独立的，但是在各种逻辑之间可能有重要元件共享，如以下将要描述得那样。逻辑 150 - 164 中的每一个都实现一种特定的功能，可以使到该 MUX 132 的选择信号 134 被确定或不确定，借此以上述方式引起一个线程切换。在每一个逻辑 150 - 164 内部包含的功能将参考在图 7 - 14 中提供的流程图进行描述。参考图 15 - 16 对一个特定示范实施例的细节进行描述。

在空闲上切换的逻辑 (150)

在空闲上切换的逻辑 150 确定在所有线程都空闲的情况下多个线程中的哪一个被选择作为一个开始线程。图 7 的流程图说明了一种依据本发明示例、用于当在一个多线程处理器内的两个线程空闲时确定一个开始线程的方法 200。应当明白该方法 200 还可以在不只支持两个线程的多线程处理器中得到应用。该方法 200 可以在在空闲上切换的逻辑 150 的内部实现。方法 200 在步骤 202 用一个空闲线程状态开始，其中两个线程（即线程 0 和 31）都是空闲的。在步骤 204，根据，仅仅用于示例，在一个流水线清除事件之后的微码中的一个指示（例如，一个流水线“nuke”），由该选择信号 134 的确定或不确立选择第一个线程（例如线程 0）。在判定框 206，就在收到对线程 0 的一条指令的请求之前是否收到对线程 1 的一条指令的请求做出判定。参见图 5，能够看出：线程切换控制逻辑 136 连接用以经由一个线程信号 137 从下一指令指针 100 接收一个指示，其中该指示关于从统一高速缓存 44 读出的下一指令。这样，线程切换控制逻辑 136，特别是在空闲上切换的逻辑 150 能够识别下一指令请求指向的那个线程。再次参考图 7，如果对线程 1 一条指令的请求在对线程 0 一条指令的请求之前接收了，则在步骤 208 执行一个线程切换。相反，则在步骤 210 保持当前线程选择。然后方法 200 在步骤 212 结束。

在分支上切换的逻辑 (152)

在诸如图 2 所示、使用了一个分支预测机制的一个微处理器 30 中, 错误预测分支的可能性当然是存在的。为此, 本发明提出了一种线程切换方案, 一旦在一个特定线程的指令流内部遇到了一条要采取的、由分支预测单元预测的分支指令, 就执行一个线程切换。考虑到相关分支指令被错误预测的可能性和继续分支指令的指令流的推测特性, 这减少了分配给那个特定线程的处理器资源。这样, 以不为它的重要程度而可能包含一条分支指令的另外一个线程为代价, 防止了用于一个可能处理错误的预测分支的线程的推理性指令流太深地渗入到该处理器流水线中。

图 8 的流程图说明了一种依据本发明示范实施例的方法 220, 用于在一个多线程处理器内部, 在当前线程 (例如线程 0) 的一条分支指令从指令流缓存器 106 到指令预解码器 108 排序 (或发送) 时, 执行一个线程切换操作。该方法 220 在步骤 222 开始, 在那儿由分配模块 125 排序一条分支指令以便从该阵列的一个逻辑分区 122 或 124 (例如, 从分配给线程 0 的逻辑分区 122) 发出。线程切换控制逻辑 136, 尤其是在分支上切换的逻辑 152, 从如图 5 中所示的分配模块 125 接收一个分支识别信号 224 (BPsbawbranch)。

该分支识别信号 224 由分配模块 125 确定, 以将已经由分支预测单元 99 预测的一条分支指令识别为正在处理。特别地, 分支预测器 104 将利用多个公知的预测和方法或算法中的任何一个 (例如, 根据一条用于相关分支指令的记录分支历史记录)、就一条特定分支指令是否将被处理或者不被处理做出预测。然后分支预测器 104 将设置与该相关指令有关的一个位、并用该指令在处理器流水线内部和“请求标识符”一起向下传播, 从而使它能够与由分配模块 125 进行的相关分配有关。应当注意到: 该分支识别信号 224 仅仅确定被预测进行处理的分支指令, 而不是那些被预测不进行处理的分支指令。然而, 在本发明的一个替换实施例中, 分支识别信号 224 可以在遇到任何已经对其做出任何预测的分支指令时, 进行确定。

图 15 说明了分支识别信号 224 向一个与门 225 提供输入。

在判定框 226, 就用于当前线程 (线程 0) 的一个预定最低数量的指令信息 (例如, 一个预定最少数量的组块) 是否已经从指令流缓

寄存器 106 发送 (或传送) 做出判定。这结束后, 并参见图 15A, 切换控制逻辑 136 包含一个以一个发送组块增量器 228 的形式的计数器, 它保持对在一个线程切换之前从指令流缓存器 106 的一个当前逻辑分区 122 或 124 发送的组块的计数。该发送组块增量器 228 由一个接收三个输入的与门 230 的输出递增。一个 BPreedy 信号 232 由分配模块 125 确定以指示一个组块准备好从有关的逻辑分区 122 或 124 发送。一个与 ENTRY 条目 (来自线程切换控制逻辑 136) 有关的线程信号 234, 标识准备发送的组块所属的那个线程 (以及相应的逻辑分区 122 或 124)。一个 NOT (ISfall) 信号 236 指示在指令控制逻辑 110 没有遇到停止状态。在确定这些信号 232 - 236 时, 对与门 230 的输出进行确定, 并递增该发送组块增量器 228, 以便记录用于从指令流缓存器 106 发送的一个组块的排序。

在本发明的一个实施例中, 线程切换控制逻辑 136 还包含一个“在分支上切换之前的组块”寄存器 238, 它可经由一条控制寄存器总线 240 进行编程以存储一个表示组块一个预定数量的值, 它是在响应于一条分支指令进行一个线程切换操作之前, 要求已经从指令流缓存器 106 中排序的组块数量。在本发明的一个替换实施例中, 表示该组块的预定数量的值可以是硬布线的。此外, 切换控制逻辑 136 还包含一个比较器 242, 用于就增量器 228 和寄存器 238 的输出执行一个比较操作, 并且如果由增量器 228 指示的值大于在寄存器 238 内存储的值的的话, 确定一个信号给与门 244。对比较器 242 输出信号的确定相应于在图 8 的判定框 226 进行的肯定判定。

与门 230 还提供输入给与门 225, 而且与门 230 的输出确定和分支标识信号 224 的确定一起使与门 225 的输出 (即一个分支发送信号 227) 被确定作为给与门 244 的输入。分支发送信号 227 的确定表示在步骤 222 检测的情况的发生。

在判定框 226 进行肯定判定之后, 该方法 200 继续到判定框 248, 在那儿就用于一个替换线程 (例如线程 1) 的一个预定最低数量的指令信息 (例如一个预定最少数目的组块) 是否待决和可用于从指令流缓存器 106 的逻辑分区 124 发送做出判定。这个判定的有利之处在于: 它防止当没有足够的指令信息已经缓存在指令流缓存器 106 内部用于目标线程时, 一个线程切换的发生, 以便保证到这样一个目标线

程的一个线程切换。特别地，一个线程切换可能比要求再次提供用于当前线程的指令信息耗费更多的时钟周期。例如，一个线程切换操作可能要求六个周期，而可能存在一个高的几率，即用于当前线程的指令信息可以在三个时钟周期内接收到。

再次参看图 15，线程切换控制逻辑 136 的一个示范实施例可以包含一个“切换之前未决的组块”寄存器，它可以允许经由控制寄存器总线 240 进行编程以存储一个表示用于一个目标线程（例如线程 1）的、应该在允许切换到那个目标线程的一个线程切换之前被缓存在一个阵列分区内部（例如分区 124）的组块的预定最少数目值。在本发明的一个替换实施例中，表示组块的这个预定数目的值同样可以是硬布线的。线程切换控制逻辑 136 还包含一个“统计未决组块”增量器 252，它保持对指令流缓存器 106 内存储的、用于一个目标线程（即不是当前线程）的组块数目的统计。增量器 252 由一个“IPD 发送未决组块”信号 254 递增，其中该信号在从统一高速缓存 44 接收的一个组块指令信息被分配给缓存器 106 内的一个条目时由分配模块 125 进行确定。分配模块 125 利用以上讨论的、并与每个响应于下一指令请求读出的组块有关的“线程标识符”，来识别一个特定组块是否用于该目标线程，而不是用于当前线程。比较器 256 比较寄存器 250 和增量器 252 内存储的相应值，并且如果由增量器 202 保持的值大于寄存器 250 内存储的值，则确定一个输出信号给与门 244。对比较器 256 的输出信号的确定相应于在图 8 中的判定框 248 进行的肯定判定。

在判定框 248 进行一个肯定判定之后，该方法 220 继续到判定框 260，在那儿就用于当前线程的一个预定量指令信息是否已经沿着微处理器 30 的流水线进行了排序做出判定。这个判定的有利之处在于：它保证了在一个线程切换操作之前当前线程的前进进程。在使用了 Intel 体系结构（IA）指令集的本发明的一个示范实施例中，就用于当前线程的三（3）个组块的指令信息是否已经排序做出判定，因为这保证了用于当前线程的至少一条完整指令。

在本发明的替换实施例中，由于对一条分支指令排序的检测，本身构成了至少一条完整指令，从而确保了与在判定框 260 的判定一致，因此在判定框 260 做出的判定可以省去。

参见图 15A，线程切换控制逻辑 136 的一个示范实施例可以包含

一个比较器 262，由它确定由该“发送组块”增量器 228 保持的计数值是否大于或等于一个预定的最小值、例如 3。在本发明的替换实施例中，这个预定的最小值可以是可编程的或是硬布线的。如果由增量器 228 保持的值等于或大于该预定的最小值，则比较器 262 确定一个输出信号给与门 244。对比较器 262 的输出信号的确定相应于在如图 8 所示的判定框 260 进行的肯定判定。

在判定框 260 进行一个肯定的判定之后，由判定框 226、248 和 260 表示的条件已经满足了，然后在步骤 264 执行一个线程切换操作。另一方面，如果由判定框 226、248 或 260 给出的这些条件中的任何一个没有得到满足的话，则在步骤 266 保持当前线程选择。然后该方法 220 在步骤 268 结束。

应当明白：由步骤 222 表示的条件、以及判定框 226、248 和 260，表示了给如图 15A 所示的与门 244 的四个输入，而且当这条件全部满足时，与门 244 的输出将被确定并提供一个输入给一个或门 245。或门 245 的输出构成了从线程切换控制逻辑 136 输出的线程选择信号 134。对或门 245 的输出的确定相应于在步骤 264 执行线程切换操作。

虽然本发明的上述实施例被描述成是用一个增量器 228、一个“在分支切换之前的组块”寄存器 238 和比较器 242 实现的，但是容易理解：本发明可以用一个减量器代替增量器 228 来实现，而且每当发生一个线程切换时在该减量器中预先载入寄存器 238 中包含的值。在这个实施例中，该减量器在每次从指令流缓存器 106 的当前逻辑分区 122 或 124 发送一个组块时进行递减。然后该减量器确定一个输出信号（对应于在先前描述的实施例中的比较器 242 的输出信号）以指示在判定框 226 的一个肯定判定。在这个实施例中，该减量器还可以由与门 230 的输出递减。

长等待时间停止逻辑（154）

由于可分配给一个多线程处理器的处理器流水线内部的一个特定线程的一个指令流的资源有限，以及由于分支错误预测和高速缓存故障，在这样一个处理器流水线内部的停止是很平常的。特别地，参见如图 2 所示的微处理器 30，可能会发生由分配器 76 确定没有足够的资源（例如物理寄存器、在线程 0 和线程 1 之间被逻辑上分区在预定位置或重新排序缓存器 78 内部的条目）可用于从队列 68 接收的一

个特定线程的指令（即微指令）。在这种情况下，分配器 76 专门为一个线程确定一个停止信号 82，该信号经由记录传送机 60 传送到微指令翻译机 54。在确定用于一个特定线程的这样一个停止信号 82 时，可能期望执行一个线程切换操作。长等待时间停止逻辑 154 包含用于在一个处理器流水线停止的情况下实现一个线程切换操作的电路。

图 9 的流程图说明了一种依据本发明一个示例方法 280，用于例如由于资源不能利用、分支错误预测、或高速缓存故障引起的一个处理器流水线内部的一个停止而出现长等待时间停止时，在一个多线程处理器内部执行一个线程切换操作。该方法 280 在步骤 282 开始，在那儿检测一个停止条件。如上所述，可以通过决定对如图 2 所示的线程特有的停止信号 82 的确定来检测这个停止条件。做为选择，可以通过决定对其它例如由分配器 76、记录高速缓存 62（两个都称为“末端”停止）、指令解码器 112 或指令控制逻辑 110 确定的停止信号的确定来检测该停止条件。在判定框 284，就继在步骤 282 检测到停止以来一个预定最少数目的时钟周期是否已经过去做出判定。参见图 15，线程切换控制逻辑 136 的一个示例可以包含一个“长等待时间统计”寄存器 286，它经由控制寄存器总线 240 进行编程，以存储一个表示自检测停止以来，在允许到一个目标线程的线程切换之前，已经过去的时钟周期的这个预定最少数目。在本发明的一个替换实施例中，表示时钟周期的这个预定最少数目的值可以是硬布线的。切换控制逻辑 136 还包含一个“长等待时间”增量器 288，由它保持对自在步骤 282 检测停止以来已经过去的时钟周期数目的统计。该增量器 288 由一个“IXstall”或指令翻译停止信号 290 递增，其中该指令翻译停止信号 290 在处理器流水线停止的每个时钟周期进行确定。此外，增量器 288 由一个“TCFBstall”信号 292 的不确定进行复位，其中该“TCFBstall”信号 292 在停止被克服时进行不确定。由一个记录高速缓存填充寄存器（TCFB）（未显示）确定该“TCFBstall”信号 292，其中该寄存器（TCFB）从记录传送机 60 向上传送一个分配停止信号 58 到微指令翻译机 54。比较器 294 比较寄存器 286 和增量器 288 内存储的相应值，并且如果由增量器 288 保持的值大于寄存器 286 内存储的值，则确定一个输出信号给一个与门 296。对比较器 294 的输

出信号的确定相应于在如图 9 所示的判定框 284 进行的一个肯定判定。

如果在判定框 284 判定时钟周期的该预定数目没有过去，则在步骤 285 保持当前线程选择，然后该方法 280 返回到判定框 284。

在判定框 284 进行一个肯定判定之后，该方法 280 继续到判定框 284，在那儿就用于当前线程的一个预定最低数量指令信息是否已经沿着微处理器 30 中的流水线进行了排序做出判定。如上面结合图 8 所示的判定框 260 描述的那样，由“发送组块”增量器 228 和比较器 262 的组合做出这个判定。比较器 262 提供一个输入给与门 296，一旦用于当前线程的一个预定数目的组块已经沿着微处理器流水线排序了，就确定比较器 262 的输出。

与门 296 的输出信号 297 被传送到分支地址计算器 114，然后由它确定一个用于从其已经进行切换的那个线程（即当前线程）的再启动指令指针，而且不确定一个用于当前线程的分支地址清除（BAclear）信号。当发生一个线程切换时，在更一般的级别，要求一个指令指针以识别一个存储单元，其中从该存储单元再启动已经从其发生切换的那个线程。这个指令指针可以是：（1）被预测为 TAKEN（要采取的）的一条分支指令的目标，在这种情况下该指令指针由在分支上切换的逻辑 152 提供；（2）由微码提供；或（3）由分支地址计算器 114 提供（是以上刚讨论的情况）。

在判定框 298 进行了一个肯定判定之后，还在判定框 300 就一个指令翻译后端清除信号 302 是否已经被确定做出判定。如果是这样的话，则在步骤 303 执行一个线程切换。做为选择，在判定框 298 或者 300 进行了否定判定之后，在步骤 304 保持当前线程选择。然后该方法 280 在步骤 306 结束。

内部清除逻辑（156）

与从微指令翻译机 54 的外部（例如从退出逻辑以一个“nuke”清除操作的形式）激活一个外部清除相反，从微指令翻译机 54 本身内部激活一个内部流水线清除操作。由于存在作为与当前一个线程有关的一个条件的结果激活了该清除操作的高可能性，而且一个替换线程可能更好地准备了在流水线内部的处理（例如已经填充了缓存器），所以这种清除操作被期望为一个线程切换操作。

图 10 的流程图说明了一种依据本发明一个示范实施例、用于当发生一个内部流水线清除时在一个多线程处理器内执行线程切换操作的方法 310。该方法 310 在步骤 312 开始，在那儿检测一个清除信号的确定。参见如图 15A 所示的切换控制逻辑 136 的示范实施例，这可以通过检测指令翻译分支地址计算器清除“IXbaclear”信号 302 的确定来执行。特别地，这个信号 302 被提供输入给一个与门 314，该与门 314 还进一步从比较器 256 接收输入。如上所述，当在指令流缓存器 106 内部由“统计未决组块”增量器 252 记录的未决的组块数目大于“切换之前未决的组块”寄存器 250 内存储的值时，对比较器 256 的输出进行确定。因此，比较器 256 的输出被确定以表示对由方法 310 的判定框 316 表示的条件的肯定判定，在这之后在步骤 318 执行一个线程切换操作。特别地，当与门 314 的两个输入都被确定时，接着与门 314 的输出被确定以向或门 245 提供一个确定的输入。另一方面，如果在判定框 316 做出了一个否定判定，则在步骤 320 保持当前线程选择。然后该方法 310 在步骤 322 终止。

无数据流逻辑 (158)

在发生了用于当前线程的一个第一级高速缓存故障的情况下，一个长等待时间的操作通常会导致可能要求相对多的时钟周期来完成。如果满足了由如图 9 所示的流程图指定的用于一个长等待时间操作的条件，则长等待时间逻辑 154 在这种情况下可以触发一个线程切换操作。然而，如果用于一个长等待时间操作的所有条件没有得到满足，则长等待时间逻辑 154 将不会触发一个线程切换操作。在这种情况下，就需要一个替换逻辑。

图 11 的流程图说明了依据本发明一个示例、用于当在一个多线程处理器内出现关于一个特定线程的“无数据流”条件时，在该多线程处理器内执行线程切换操作的方法 330。方法 330 在步骤 332 开始，在那儿检测到缺少用于当前线程的、从统一高速缓存 44 到指令流缓存器 106 的一个指令流。参见图 15A，在一个示范实施例中，这可以通过一个数据流信号 334 的确定（不确定）进行检测，其中该数据流信号 334 由统一高速缓存 44 在一个指令流被提供给指令流缓存器 106 期间的每个时钟周期进行确定。“无数据流”条件也可以作为到指令预先解码器 108 的数据流的一个锁定被检测到，因为这暗示在从统一

高速缓存 44 到指令流缓存器 106 的数据流上的一个锁定。“无数据流”条件也可以作为到指令预先解码器 108 的数据流的一个锁定被检测，是由于这暗示在从统一高速缓存 44 到指令流缓存器 106 的数据流上的一个锁定。

在判定框 336，就没有数据流从统一高速缓存 44 到指令流缓存器 106 的一个预定最小数目的时钟周期是否已经过去做出判定。为此，线程切换控制逻辑 136 如图 15A 所示，包含一个“无数据流”计数器 338，它为数据流信号 334 不确定的每个连续时钟周期加 1，以指示缺少一个用于当前线程、流向指令流缓存器 106 的指令流。计数器 338 可以通过信号 335 的确定进行复位，一旦恢复了从指令流缓存器 106 到指令预先解码器 108 的一个数据流，就对信号 335 进行确定。可以响应于在信号 232 和 236（即，BPreedy 和 NOT ISstall 信号）上执行的一个与操作对该信号 335 进行确定。逻辑 136 进一步包含一个“无数据流周期”寄存器 340，它可经由控制寄存器总线 240 进行编程以存储一个表示在能够检测到一个无数据流条件之前的时钟周期的一个预定最小数目值。在本发明的一个替换实施例中，表示这个时钟周期的预定数目的值可以是硬布线的。计数器 338 和寄存器 340 向一个比较器 342 提供输入，当由计数器 338 保持的值大于在寄存器 340 内存储的值时由该比较器 342 确定一个输出到与门 344。对比较器 342 的输出信号的确定相应于在如图 11 所示的判定框 336 进行的一个肯定判定。

如果在判定框 336 进行了一个否定判定，则方法 330 继续进行到步骤 345，在那儿保持当前线程选择。

继在判定框 336 进行了一个肯定判定之后，该方法 330 继续在判定框 346 和 348 执行判定，它们相当于如图 8 所示的方法 220 在判定框 248 和 260 做出的判定。因此，实现在判定框 346 和 348 的判定的比较器 256 和 262 的输出被输入到与门 344。与门 344 的输出再次输入到或门 245。这提供了一个如果对应于判定框 336、346 和 348 出现了肯定判定、则在方法 330 中的步骤 350 执行线程切换操作的逻辑实现。做为选择，如果在判定框 336、346 或 348 中的任何一个出现了一个否定判定，则在步骤 352 保持当前线程选择。然后方法 330 在步骤 354 结束。

强迫线程改变逻辑 (160)

在某些情况下，可能会遇到一个尤其是线性的指令流（包含一个第一或者第二线程），而且因此没有包含许多分支指令。假定以上或下面讨论的其它条件都不存在，可以设想如果没有事件出现以触发一个线程切换，则这样一个线性线程可能耗费一个不成比例的处理器资源。为了防止这样一个线性线程过分使用处理器资源，本发明提出了继从一个资源、例如指令流缓存器 106 排序或发送一个预定量的、用于单个线程的指令信息之后触发一个线程切换操作的逻辑。

图 12 的流程图说明了一种依据本发明一个示例、用于当用于一个特定线程（例如线程 0）的、一个预定量的指令信息从指令流缓存器 106 到指令预解码器 108 排序（或发送）时，在一个多线程处理器内部执行一个线程切换操作的方法 360。方法 360 在步骤 362 开始，在那儿用于一个特定线程（例如线程 0）的指令信息被选择用于从指令流缓存器 106 排序和发送。这个选择可以由从线程切换控制逻辑 136 输出到 MUX 132 的线程选择信号 134 表示。

在判定框 364，就用于当前线程（例如线程 0）的一个预定最大数量的指令信息是否已经从指令流缓存器 106 排序做出判定。如图 15A 所示的示例线程切换控制逻辑 136 包含一个“资源拱曲”寄存器 366，它可经由控制寄存器总线 240 进行编程以存储一个表示在触发一个线程切换操作之前、用于一个特定线程的、从指令流缓存器 106 进行排序的组块的一个预定最大数目的值。在本发明的一个替换实施例中，表示组块的这个预定最大数目的值可以是硬布线的。该“资源拱曲”寄存器 366 和保持对从缓存器 106 发出的用于特定线程的组块的计数的“发送组块”增量器 228，提供输入到一个比较器 368 里，当增量器 228 的输出值大于在该“资源拱曲”寄存器 366 内的存储值时由该比较器 368 确定一个输出给与门 344。因此，对比较器 368 的输出信号的确定相应于在如图 12 所示的判定框 364 进行的一个肯定判定。

如图所示，与门 344 的输出向或门 245 提供输入，还提供一个“插入 FTC”输出 434。

如果在判定框 364 进行一个否定判定，则方法 360 继续进行到步骤 370，在那儿保持当前线程选择，随后方法 330 循环返回到判定框

364.

继在判定框 364 进行了一个肯定判定之后，该方法 360 继续在判定框 372 和 374 执行判定，它们相当于如图 8 所示的方法 220 在判定框 248 和 260 做出判定。因此，如图所示，实现在判定框 372 和 374 做出判定的比较器 256 和 262 的输出被输入到与门 344。

继在判定框 336、346 和 348 做出肯定判定之后，一个流标记（或“插入流”）（例如 ‘1100’）被插入到从指令流缓存器 106 发送的该指令流中。特别地，由于线程切换操作也许已经在一条不完全的指令上发生了，所以分支地址计算器 114 将被要求重新开始读取当前线程，所以该插入流是必需的。在这种情况下，分支预测单元 113 将该插入流插入到指令流中以指示强迫线程改变。插入流具有一个被确定的 ForceUOPValid。该插入流只是在一个组块已经预定从指令流缓存器 106 发送给指令预先解码器 108 以后才被插入。一旦发送了有关的组块，该插入流就被多路复用到一个 ForceUOPVector 字段（event_info 字段）里。为此，参考图 16，它说明了用于在指令流缓存器 106 内部的一个示范阵列分区 122 内缓存的组块的 event_info 字段。如图所示，“插入 FTC”输出 434 操作一个 MUX 450 以在（1）存储在分区 122 内部的一个条目 120 内的一个组块的 event_info 字段 121 的当前内容以及（2）插入流 452 之间进行选择。在继确定了比较器 368 的输出之后输出 244 被确定的情况下，插入流 452 将被插入到从分区 122 发送的一个组块的 event_info 字段 121 中。

然后响应于对与门 344 的输出的确定，在步骤 376 通过对选择信号 134 的确定执行一个线程切换操作。做为选择，如果在判定框 364、372 或 374 中的任何一个出现了一个否定判定，则在步骤 378 保持当前线程选择。然后该方法 360 在步骤 380 结束。

由强迫线程改变逻辑 160 启动的一个线程切换操作不同于在本说明书中讨论的其它线程切换操作，体现在以下方面：在其上发生线程切换的数据组块没有某些特殊或区别特征，其中这些特殊和区别特征可能会提供一个线程切换操作的某些前兆。特别地，由线程切换控制逻辑 136 内部的其它逻辑 150 - 164 执行的线程切换操作伴随有一个使线程切换操作有利的条件，而且因此实施的硬件不要求警告。没有配备指令控制逻辑 110 以处理一个突然的和意料不到的线程切换操

作。因此，插入流标记提供了一个机制以向指令控制逻辑 110 通知上游发生的线程切换操作。

此外，利用一个强迫线程改变，如同利用在当前说明书中讨论的其它线程切换操作一样，需要一个“重新启动”指令指针。由于由强迫线程改变逻辑 160 强迫了在该指令流中的一个中断，因此期望让微码提供有重新启动指针，这与下一指令指针 100 相反。

外部清除逻辑 (162)

如以上参考内部清除逻辑 156 描述得那样，从微指令翻译机 54 的外部激活一个外部清除。

图 13 的流程图说明了依据本发明示范实施例、用于当发生一个外部流水线清除时在一个多线程处理器内执行一个线程切换操作的方法 400。该方法 400 在步骤 402 开始，在那儿检测对一个外部清除信号的确定。参见如图 15A 所示的线程切换控制逻辑 136 的示范实施例，这可以通过检测一个记录高速缓存/微指令翻译机清除信号“TCmitclear”404 的确定来执行，该信号提供一个直接输入到或门 245。因此，如图 5 所示，信号 404 的确定将导致对从线程切换控制逻辑 136 传送到 MUX 132 的线程选择信号 134 的确定。这在该方法 400 的步骤 406 中反映出来，该方法此后在步骤 408 结束。

插入流逻辑 (164)

在微指令翻译机 54 内，如图 3 所示，当指令翻译后备缓冲器 (ITLB) 102 登记一个页故障时，一个流标记 (也被称为一个“插入流”) 被插入到指令流中以标记这个事件。这样一个页故障导致一个页故障控制器 (未显示) 的一个相对长的走页操作，它可能仅仅例如耗费最多 150 个时钟周期。在一个多线程处理器中，当前线程的一个指令流发生一个页故障时，执行一个线程切换操作以便允许一个替换线程利用由走页操作带来的等待时间，这可能是有利的。如上所述，插入流逻辑 164 提供了一个在出现页故障时实现和执行一个线程切换功能的逻辑示范实施例。

除了页故障之外，可能导致一个插入流的其它事件在本领域内是公知的，或是专门用于一个微处理器体系结构。

图 14 的流程图说明了依据本发明一个示范实施例、用于在检测到用于当前线程的一个指令流内部的一个插入流时在一个多线程处

理器内执行一个线程切换操作的方法 420，其中插入流指示仅仅例如一个关于指令 TLB 102 的页故障。该方法 420 在步骤 422 用选择当前线程（例如线程 0）开始。这个选择可以由从线程切换控制逻辑 136 输出到 MUX 132 的线程选择信号 134 表示。在判定框 424，就在用于当前线程的指令流内的一个指令翻译事件 “Iteventid” 字段 426 是否指定了一个插入流（例如不具有 ‘0000 或 ‘0111 的值）做出判定。特别地，在指令翻译事件字段 426 内出现上述值可能指示缺少一个插入流。参见图 15A，如图所示，字段 426 被输入到一对比较器（或其它逻辑）428 和 430，由它们确定该字段 426 是否包含值 ‘0000 或者 ‘0111。如果是这样的话，则比较器 428 或 430 确定一个输出给或非 NOR 门 432，它的输出被输入到或门 245。

在判定框 436，就用于一个目标线程（例如线程 1）的一个预定最少数量的指令信息是否可以从指令流缓存器 106 发送做出选择性地判定。可以利用或非门 NOR42 的输出和比较器 262 的输出来做出这个判定。

继在判定框 424 和 436 做出了肯定判定之后，然后可以在步骤 438 执行一个线程切换操作。特别地，对 NOR 门 42 的输出的确定可以相当于在步骤 438 执行的线程切换操作，它通过或门 245 传播以导致对选择信号 134 进行确定。做为选择，如果在判定框 424 或 436 中的任何一个做出了一个否定判定，则在步骤 440 保持当前线程选择。然后该方法 420 在步骤 442 终止。

结论

在以上参考图 15 讨论的线程切换控制逻辑 136 的示例中，多个逻辑 150-164 被描述为利用增量器、寄存器和比较器的组合来实现。在本发明的一个替换实施例中，这些逻辑中的某些或全部都可以使用仅仅从一个预定的、可编程的值递减并在达到零值时确定一个信号的一个减量器来实现。这样一个实施例的例子如图 15B 所示。

总之，以上描述的线程切换逻辑 136 的有利之处在于：它提供了多个灵活的机制，响应于发生在其上可能有益于执行这种线程切换操作的事件，实现和执行线程切换操作。此外，通过包含可编程的参数，本发明允许用于各种情况和考虑线程切换机制的修改和很好调整。例如，当执行一个类型的应用程序、诸如多媒体应用程序时，可能期望

使线程切换机制的参数被设置为与执行一个不同类型的应用程序、诸如一个字处理应用程序时不同的值。

线程切换逻辑 136 适于时间分割一条路径（在一个源资源和一个目的地资源之间）的带宽和/或在一个多线程处理器内的处理资源。虽然线程切换逻辑 136 如上所述，被用来从一个指令流缓存器向一个指令预解码器发送用于两个线程的指令信息，但是容易理解，本发明的这些示教能够被用在一个处理器流水线内部从任何源向任何目的地发送用于多个线程的指令信息。

因此，已经描述了用于在一个多线程处理器内进行线程切换的方法和装置。尽管已经参考特定示例对本发明进行了描述，但是显然可以在没有背离本发明的宽阔的精神和范围的情况下，对这些实施例进行各种修改和改变。因此，该说明书和附图是用来作为例证的而不具有限制意义。

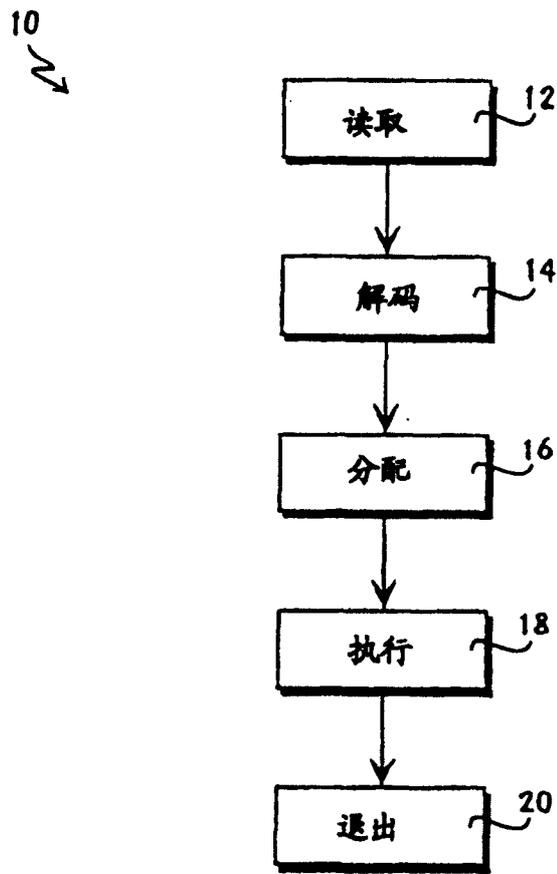


图 1

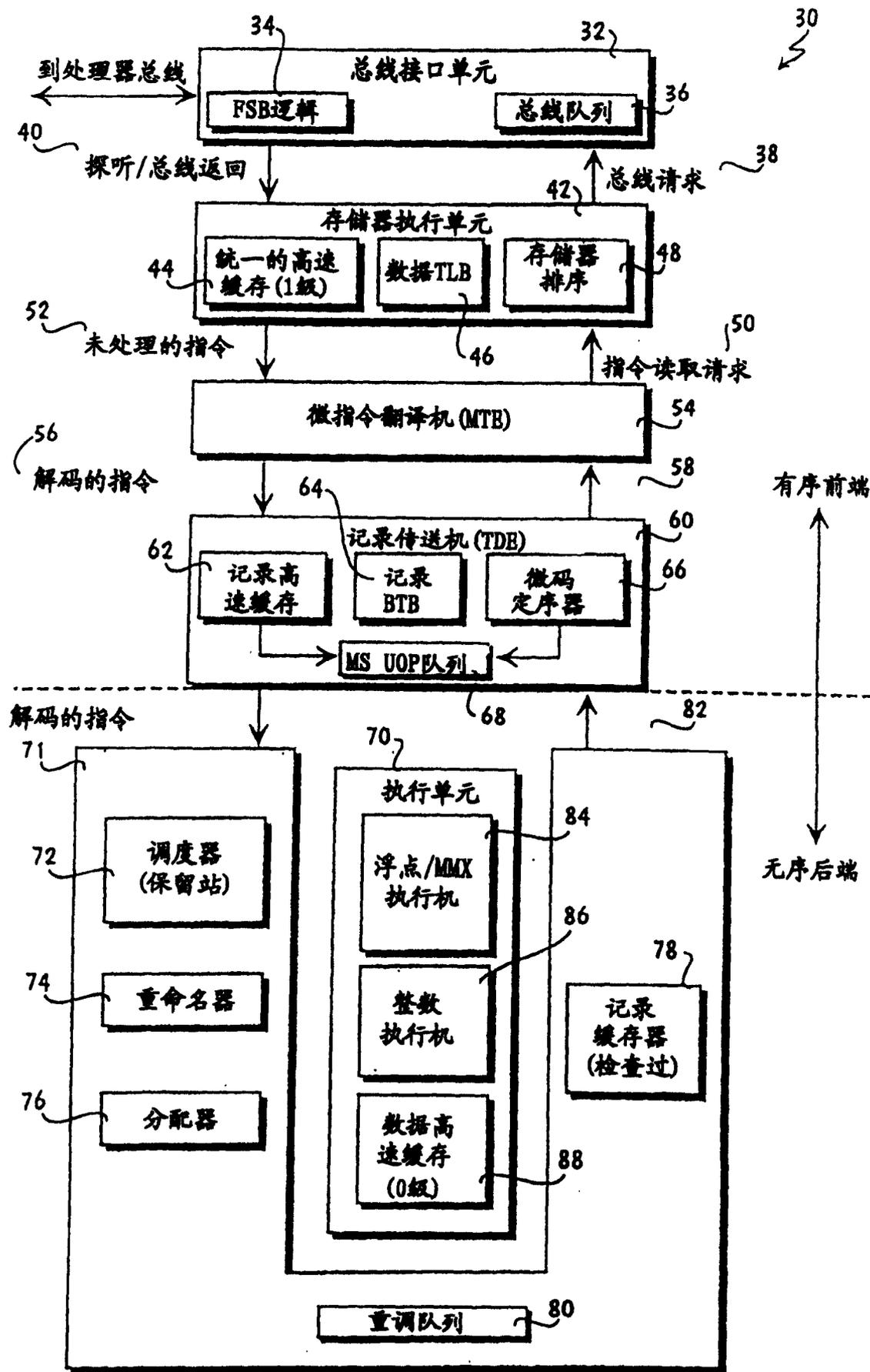


图 2

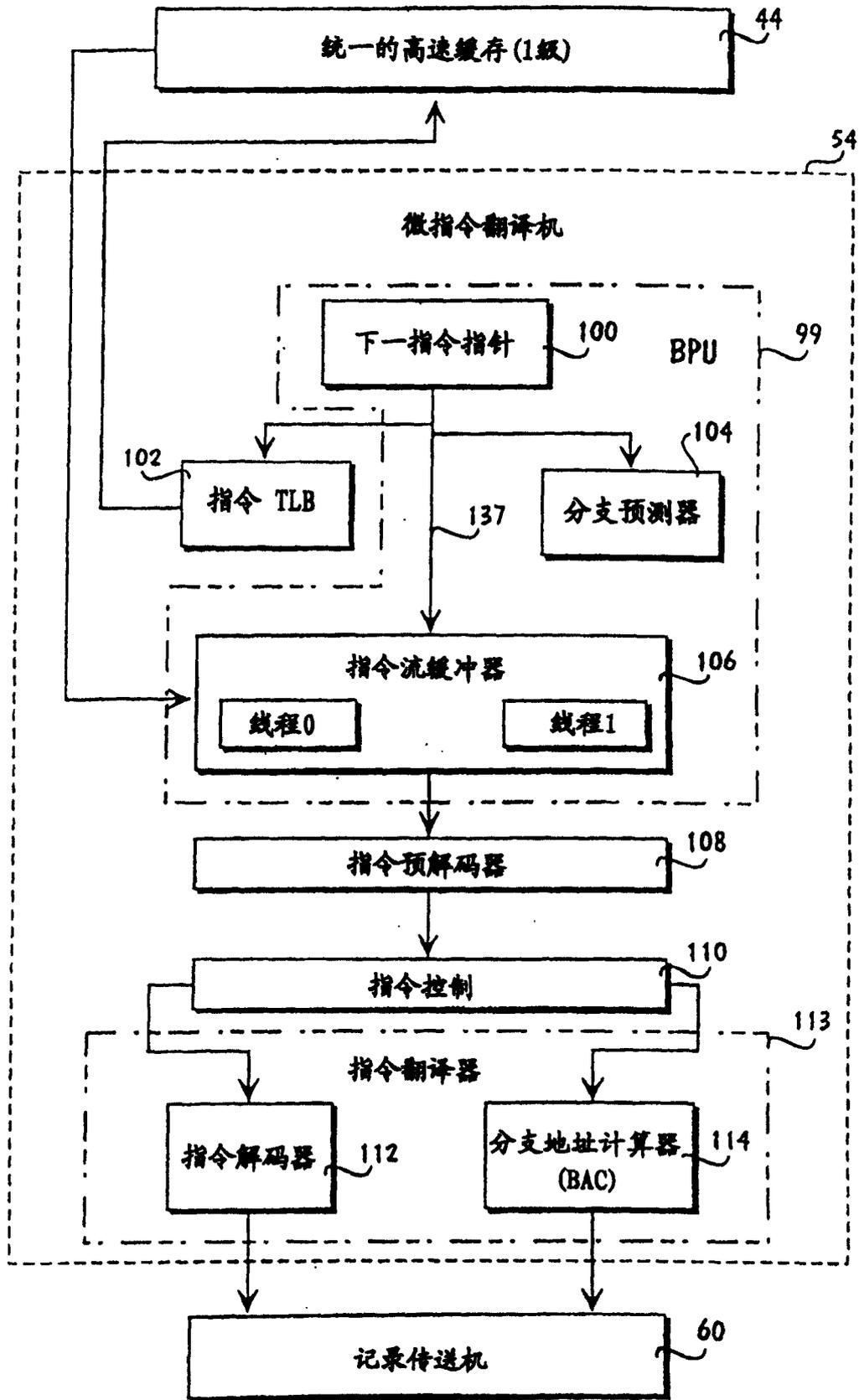


图 3

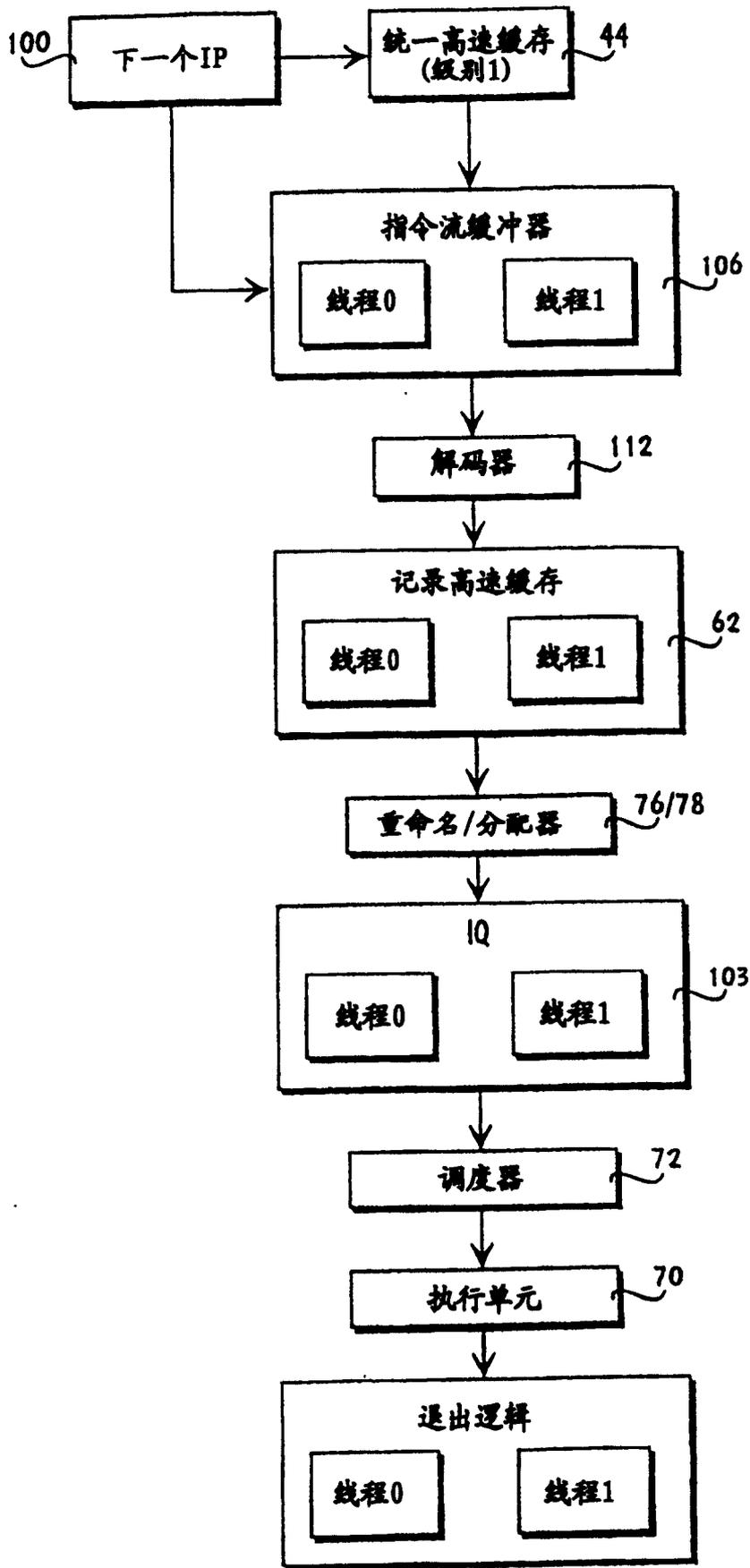


图 4

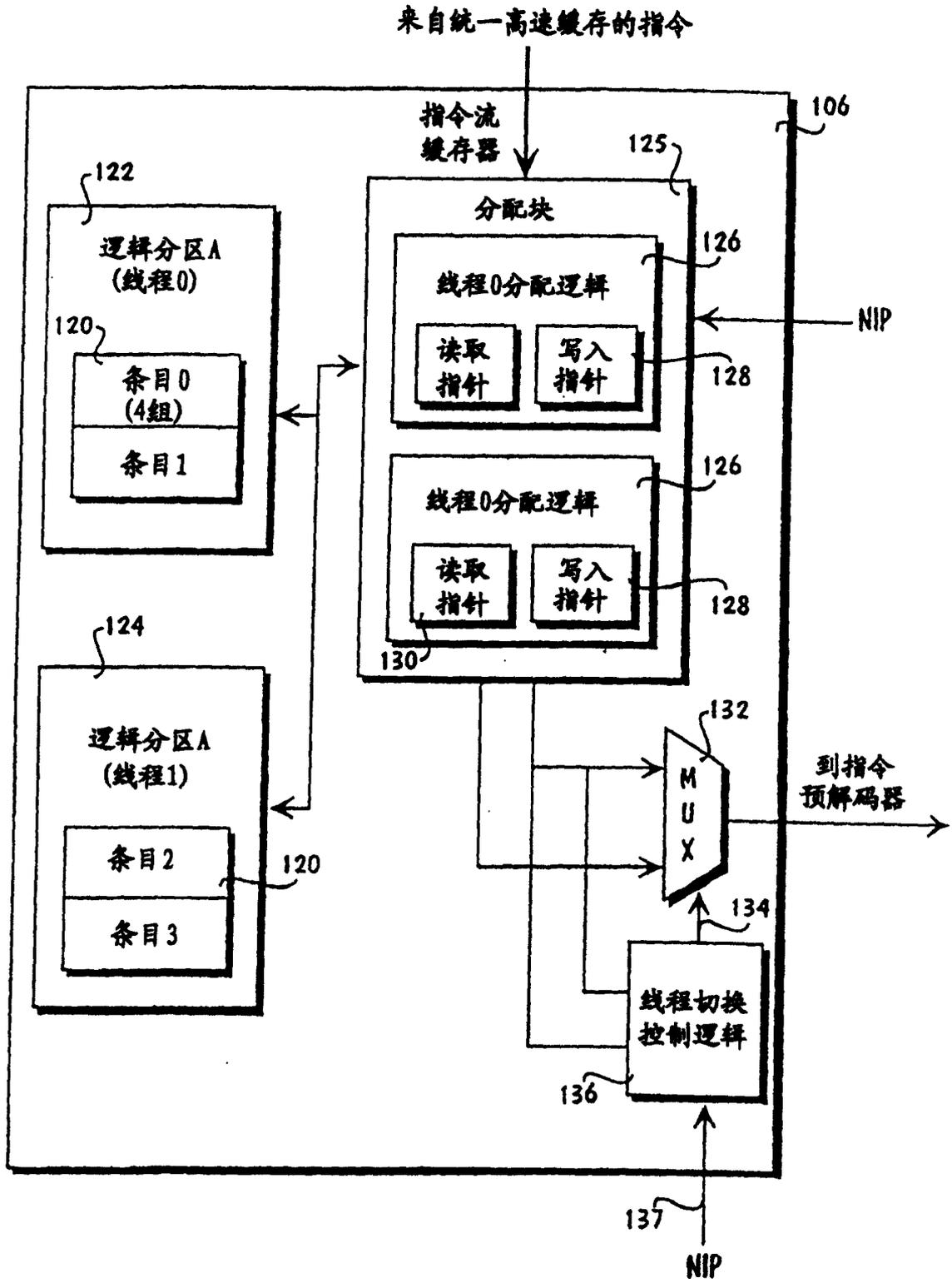


图 5

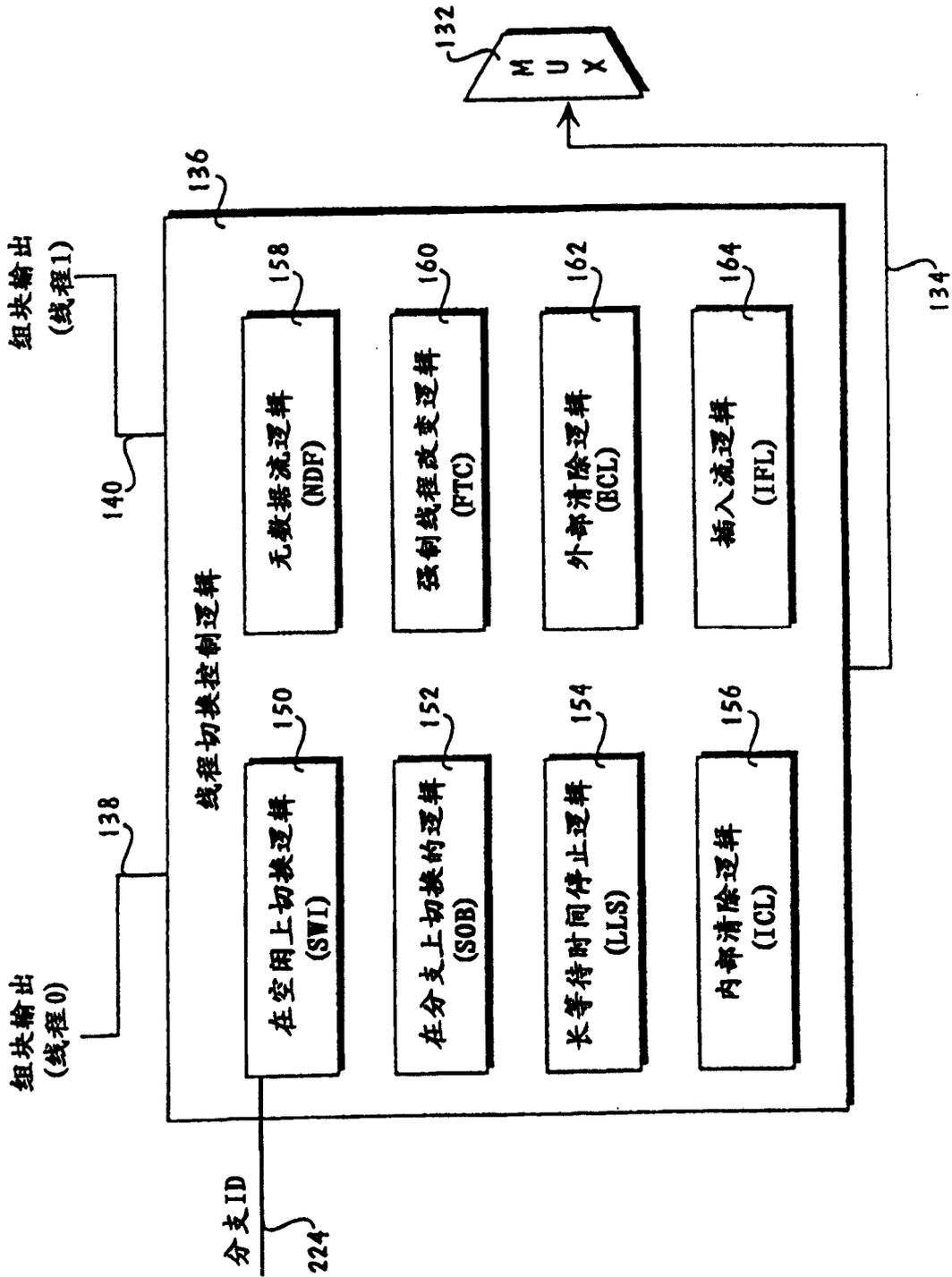


图 6

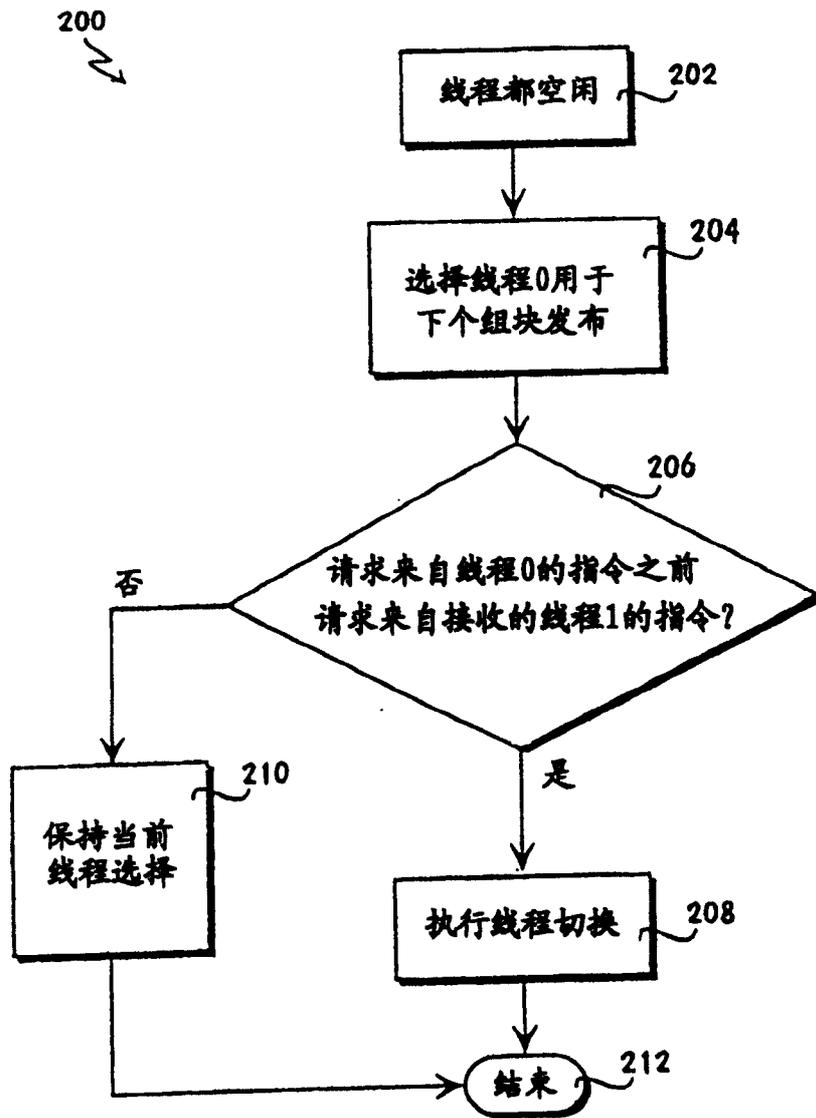


图 7

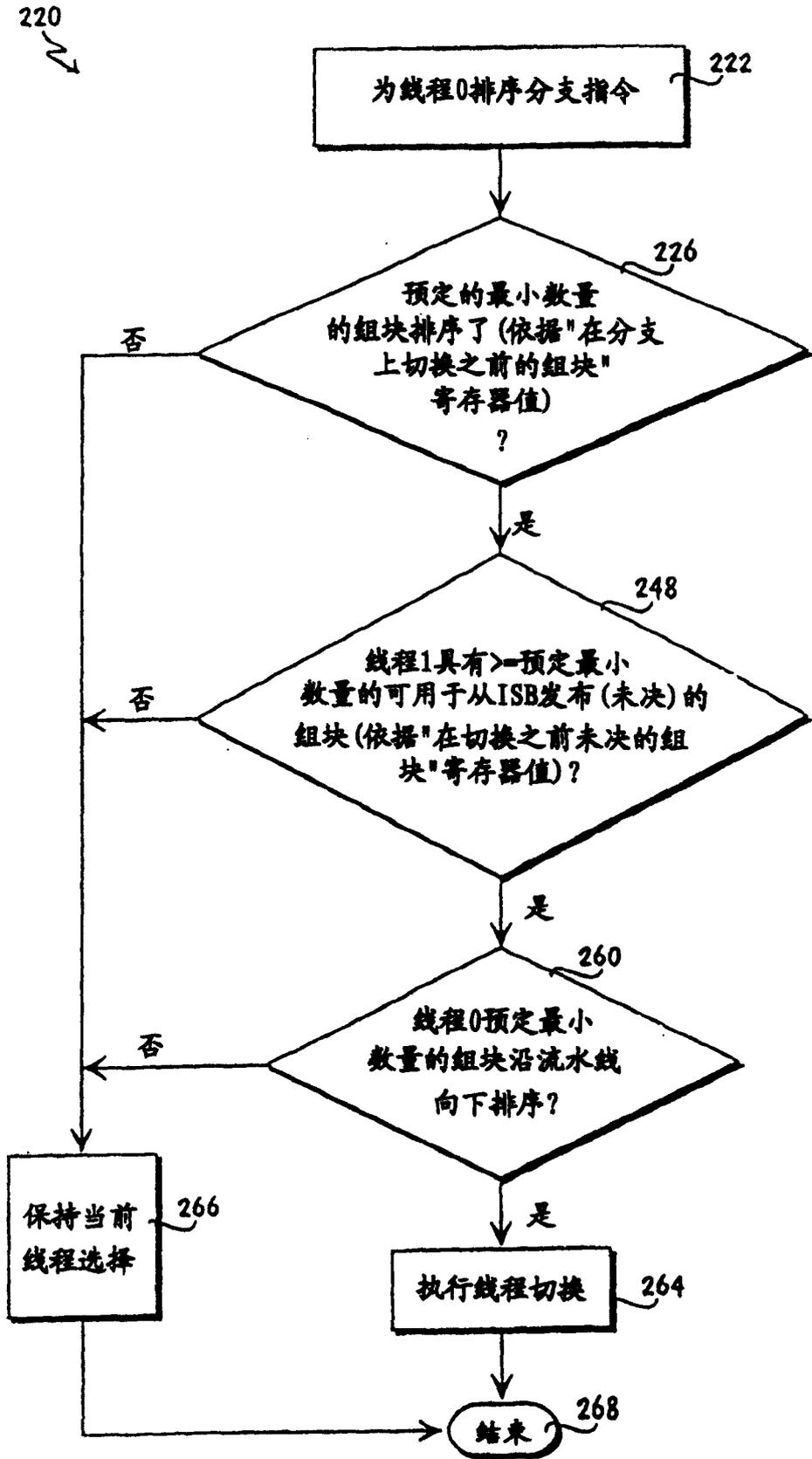


图 8

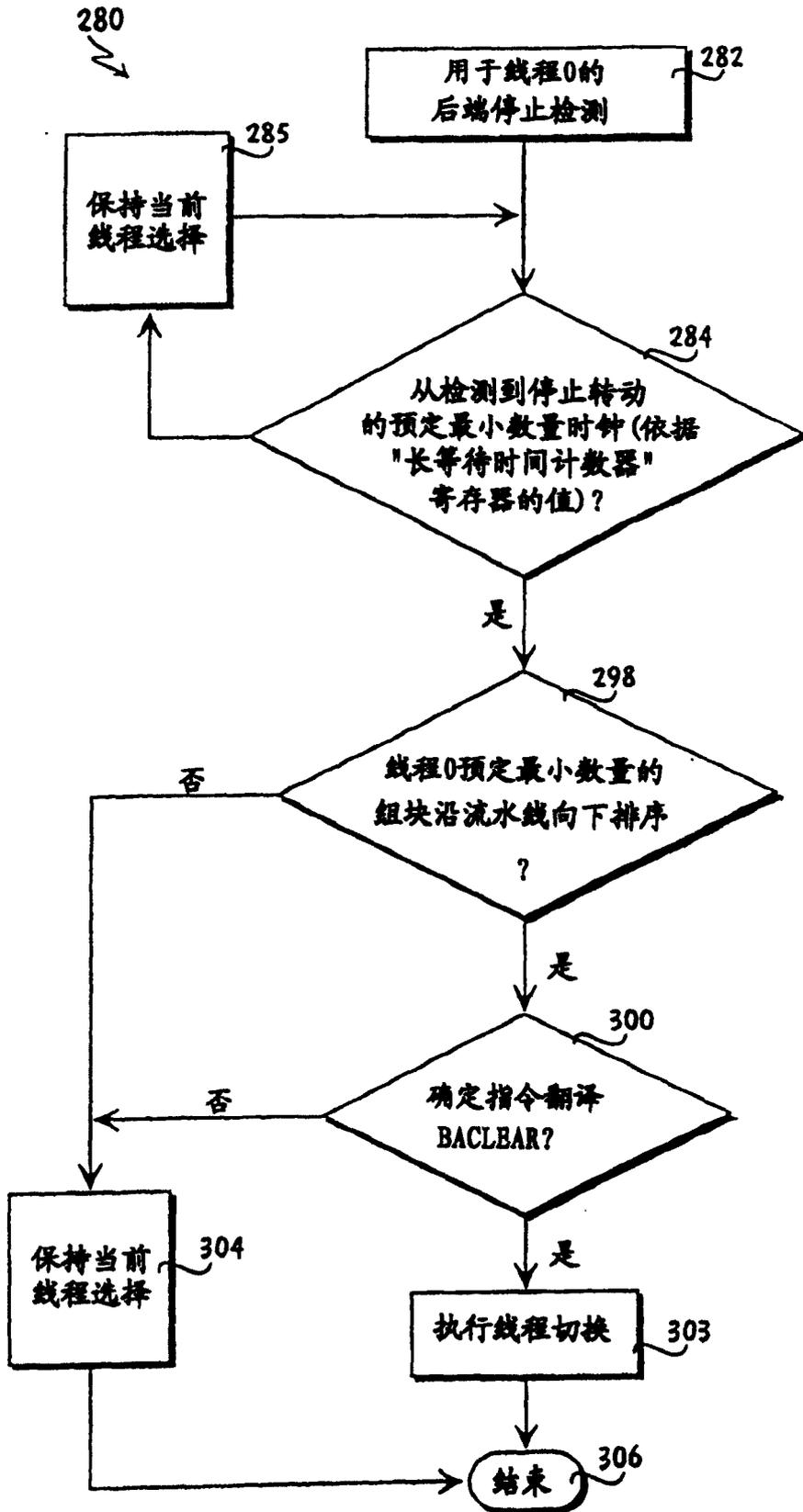


图 9

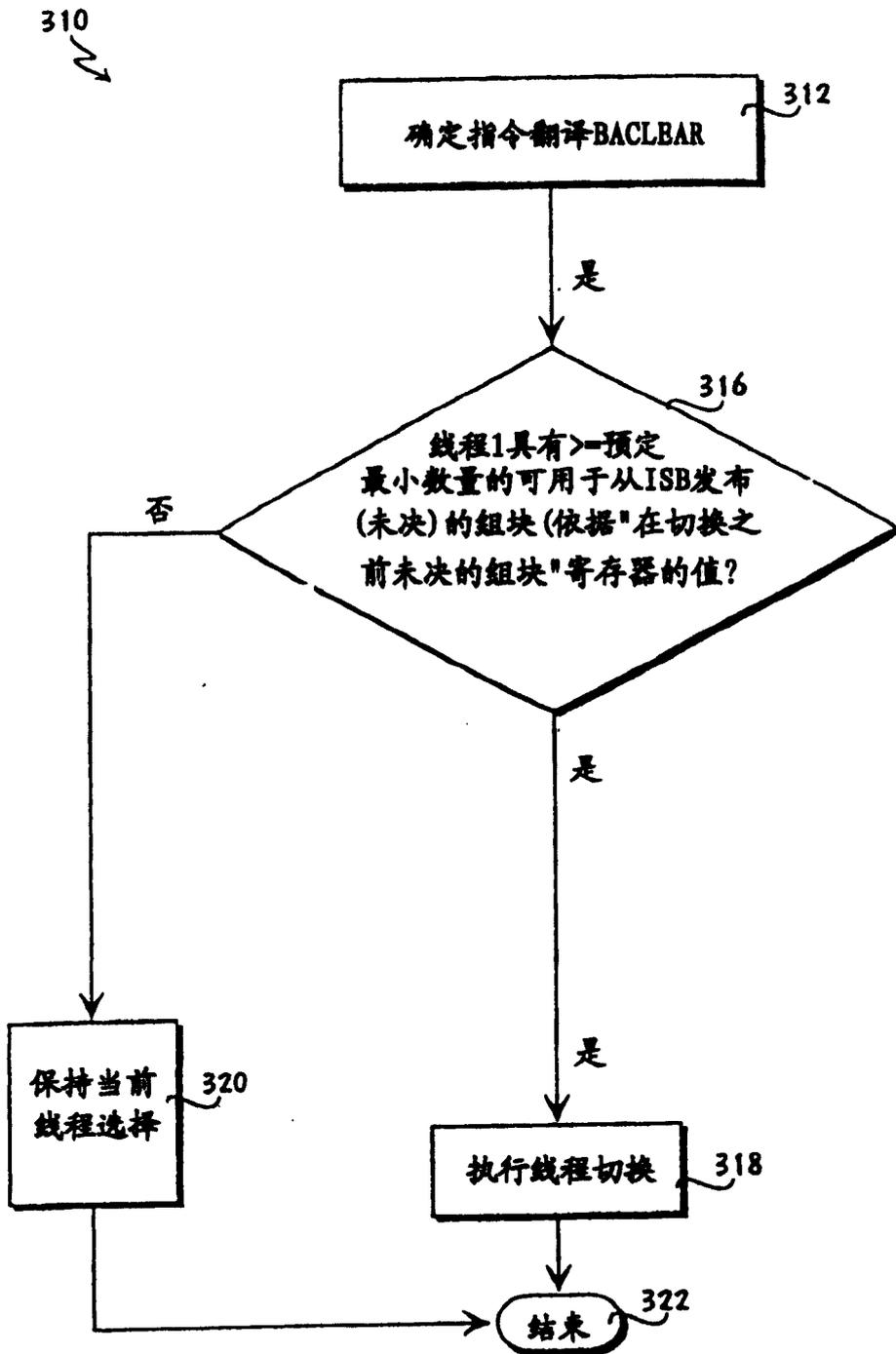


图 10

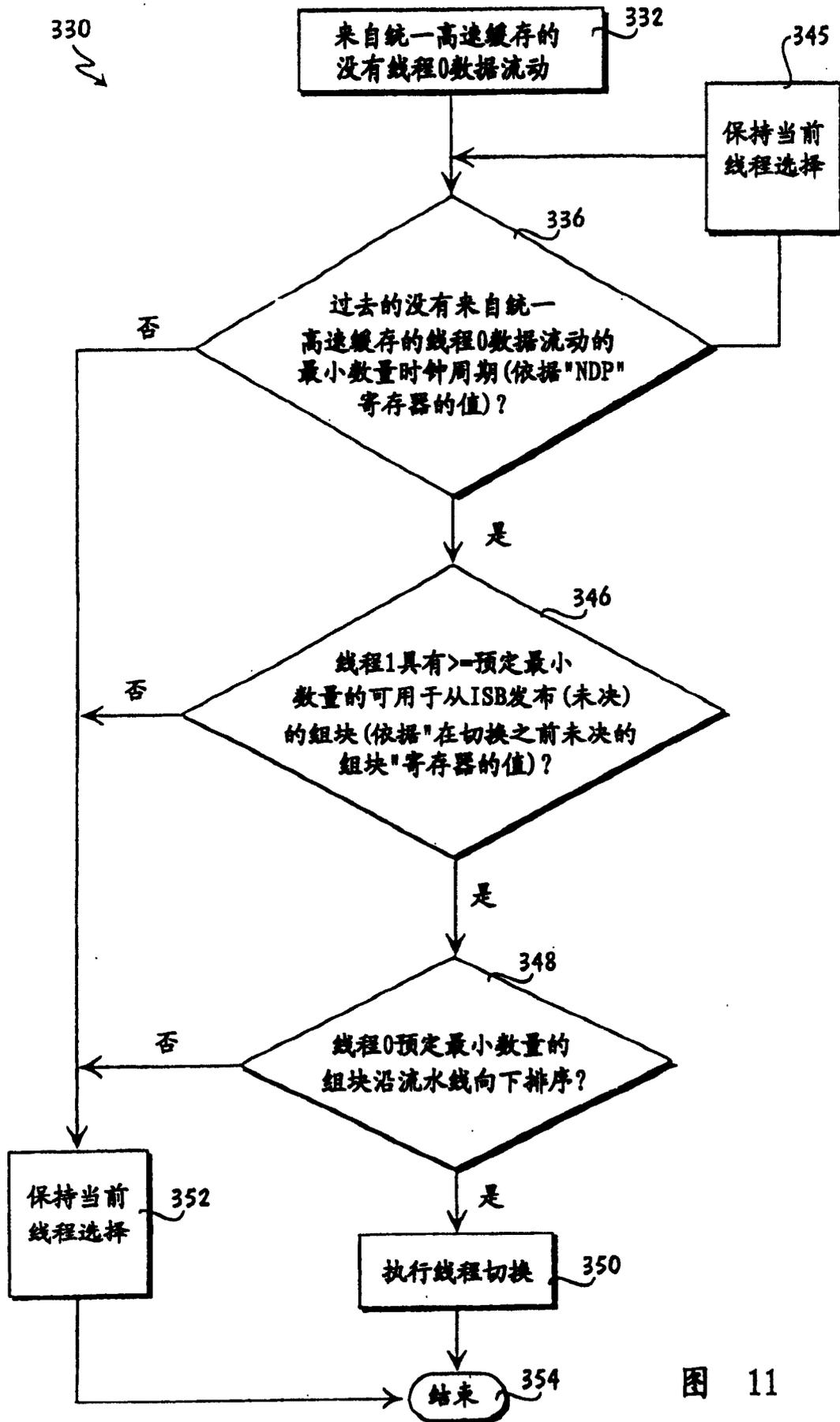


图 11

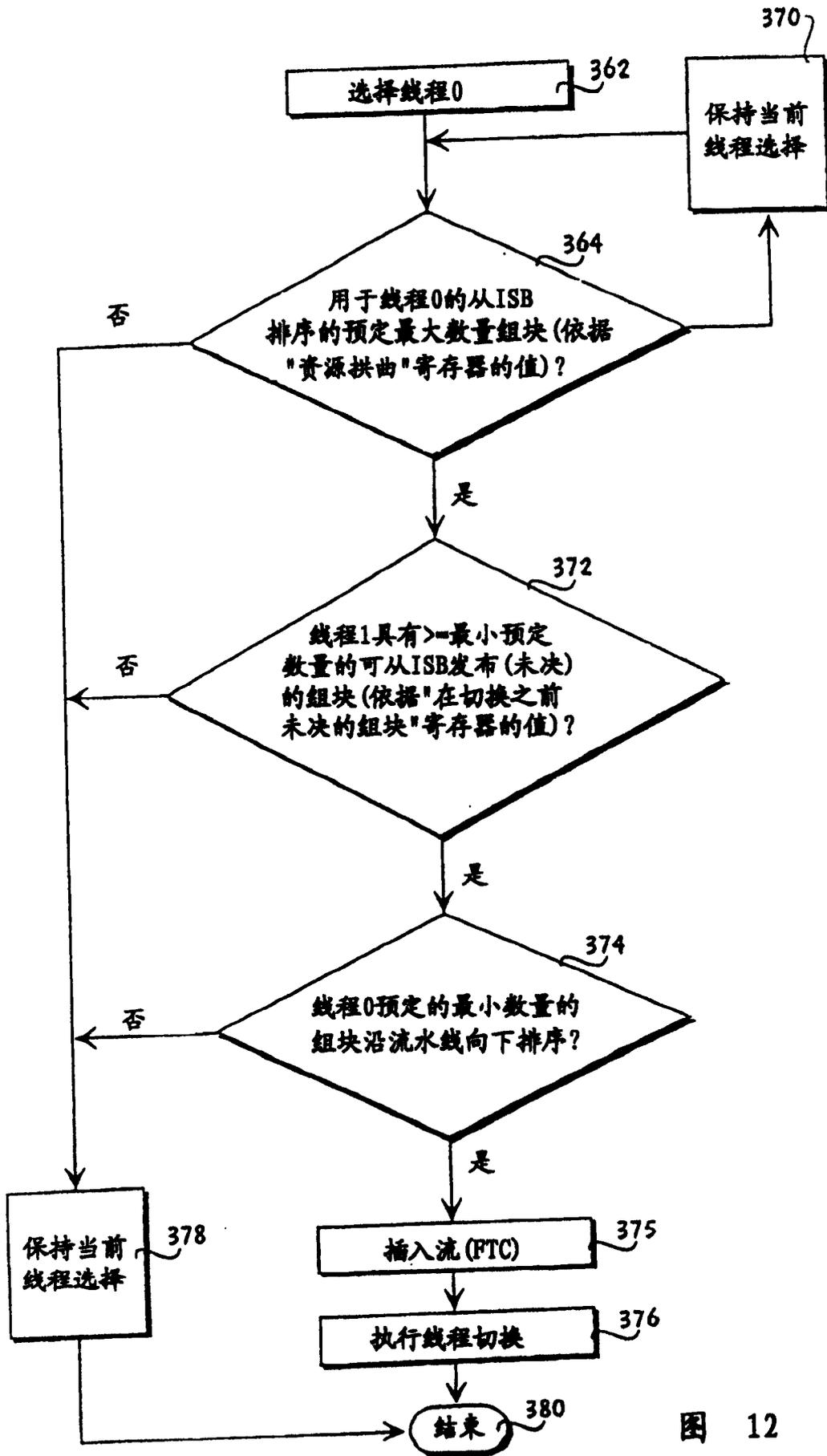


图 12

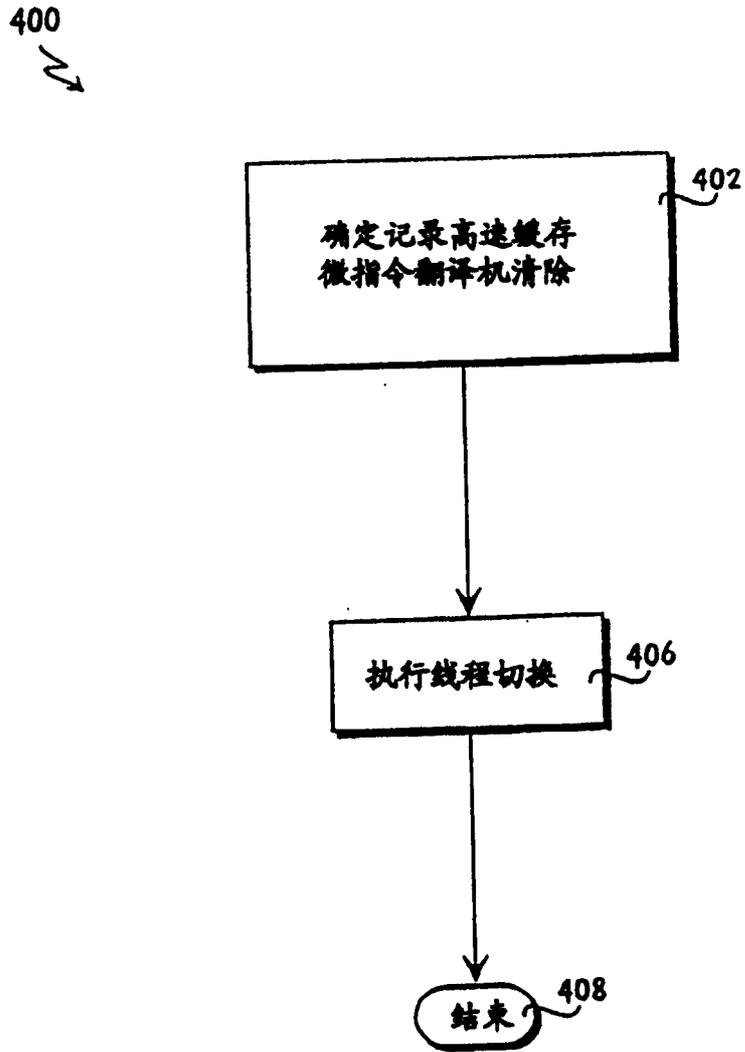


图 13

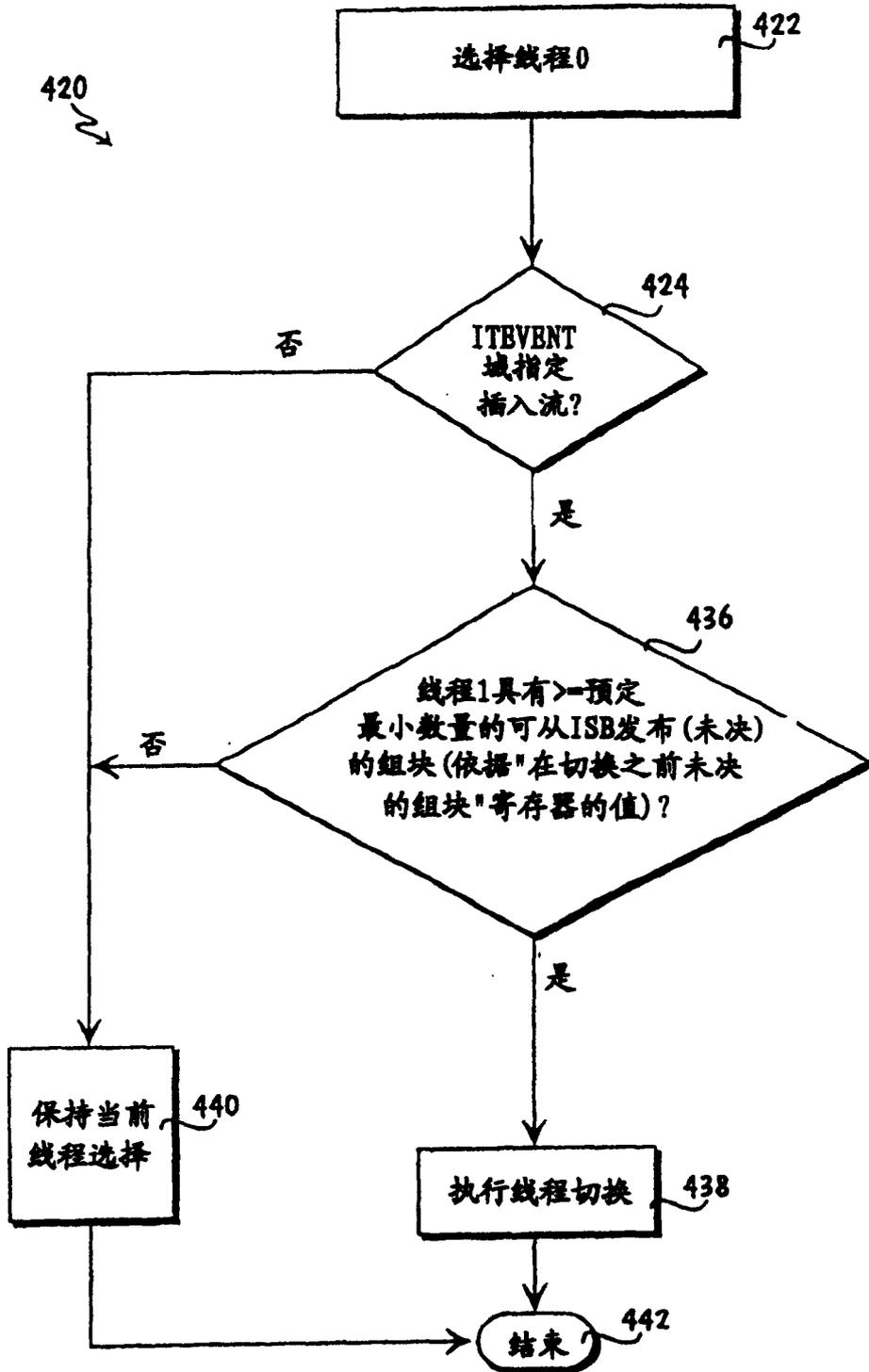


图 14

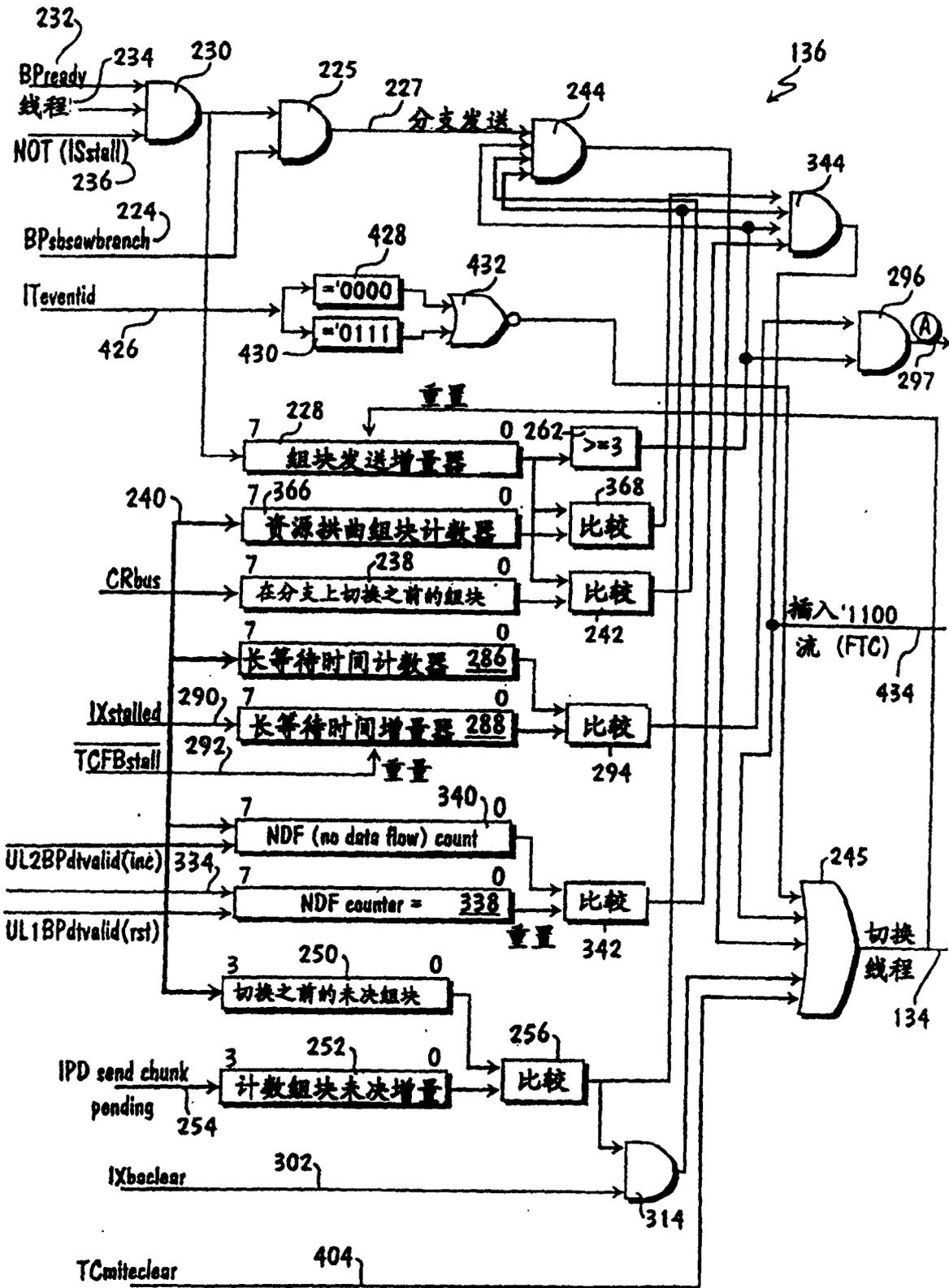


图 15A

Ⓐ BPLLSClearT%hread%M05511

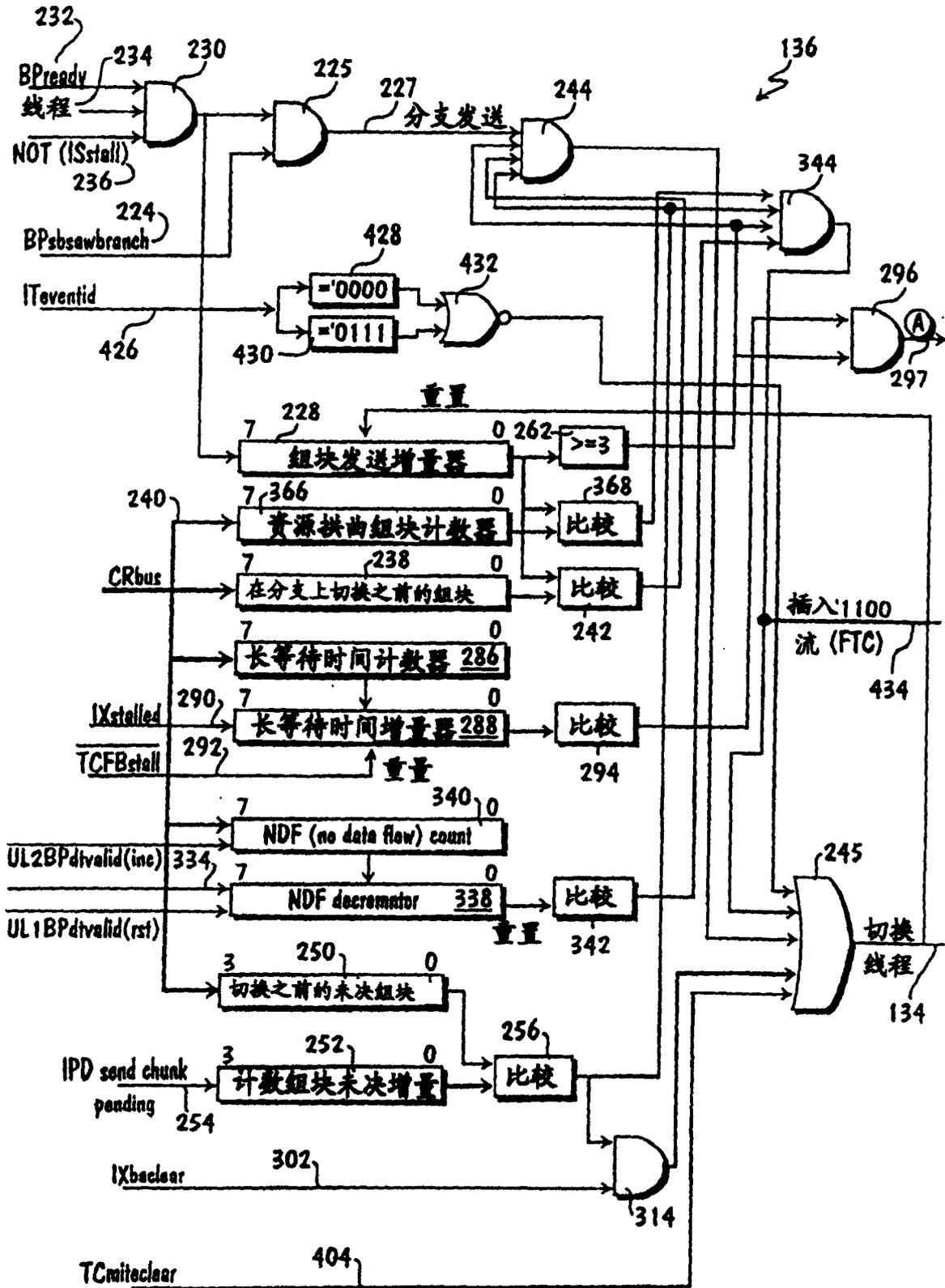


图 15B

Ⓐ BPLLSClearT%threed%MO5511

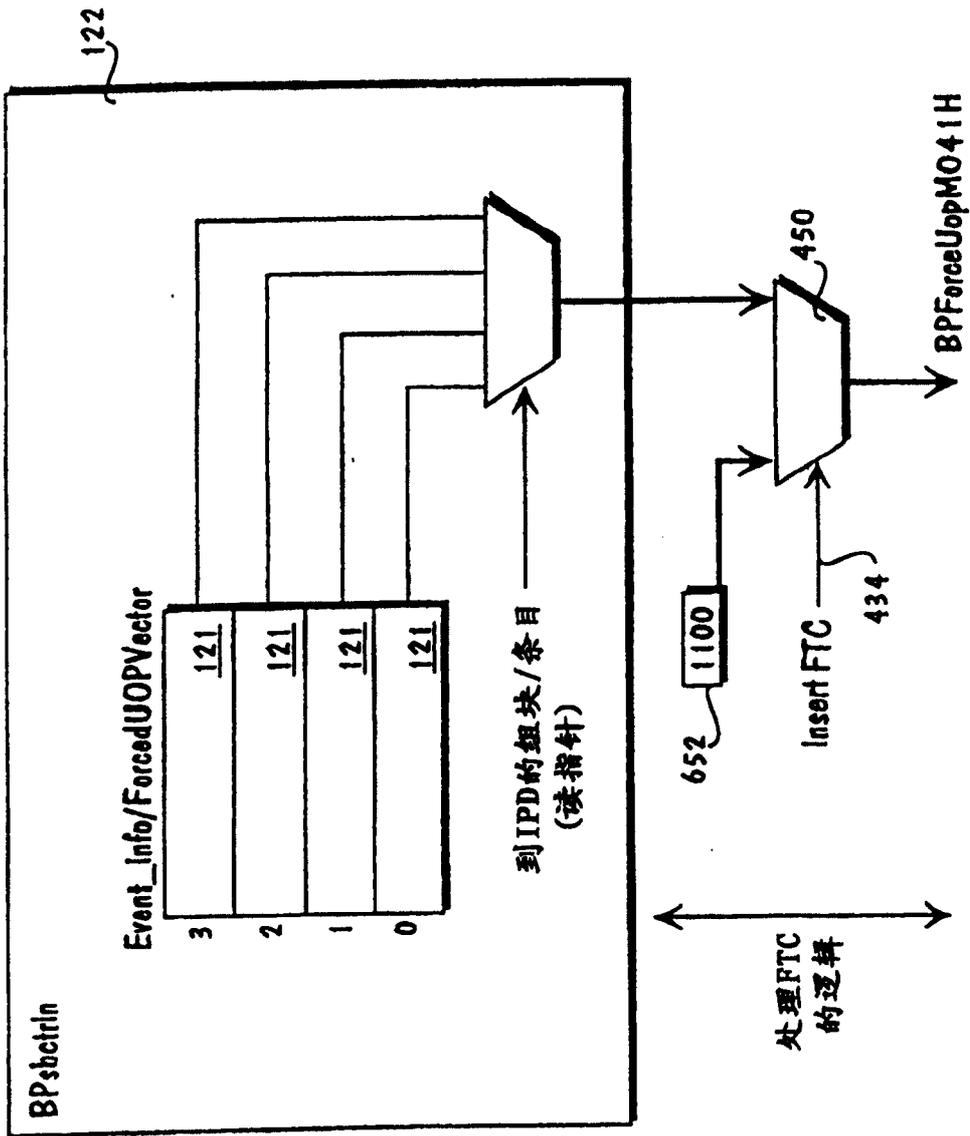


图 16