[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
G06F 9/38

[12] 发明专利说明书

[21] ZL 专利号 00818156.X

[45] 授权公告日 2005 年 7 月 13 日

[11] 授权公告号 CN 1210649C

[22] 申请日 2000.8.16 [21] 申请号 00818156.X

[30] 优先权

[32] 2000. 1. 3 [33] US [31] 09/476,322

[32] 2000. 1. 3 [33] US [31] 09/476,570

[32] 2000. 1. 3 [33] US [31] 09/476,578

[32] 2000. 1. 3 [33] US [31] 09/476,204

[86] 国际申请 PCT/US2000/022458 2000.8.16

[87] 国际公布 WO2001/050253 英 2001.7.12

[85] 进入国家阶段日期 2002.7.3

[71] 专利权人 先进微装置公司

地址 美国加利福尼亚州

[72] 发明人 J·B·凯勒 R·W·哈达德
S·G·迈耶

审查员 韩鲜萍

[74] 专利代理机构 北京纪凯知识产权代理有限公司

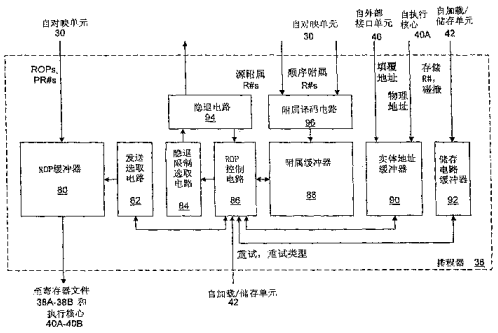
代理人 戈 泊 程 伟

权利要求书 3 页 说明书 48 页 附图 14 页

[54] 发明名称 能够发送及重新发送附属链接的排程器、包括该排程器的处理器以及排程方法

[57] 摘要

排程器发送指令运算以供执行，但是亦保留该指令运算。 若特定的指令运算后来发现需要非臆测执行，该特定的指令运算仍然储存在排程器内。接着判定该特定的指令运算已经变成非臆测的(透过在该特定的指令运算之前的指令运算的发送及执行)，该特定的指令运算可以由该排程器重新发送。 相较于由该管线排除的该特定的指令运算及较新的指令运算并且重新提取该特定的指令运算，可以减少非臆测执行的指令运算的不正确排程的代价。 此外，该排程器对于每个已经发送的指令运算可维持该附属指示。 若该特定的指令运算为重新发送，附属于该特定的指令运算(直接地或非直接地)的指令运算可以透过该附属指示来确认。 该排程器亦重新发送该附属指令运算。 在程序顺序内接续该特定的指令运算但不附属于该特定的指令运算的指令运算将不重新发送。 因此，对于为非臆测执行的指令运算的不正确排程的代价可以比该特定的指令运算及所有较新的指令运算的排除及重新提取该特定的指令运算更为降低。



ISSN1008-4274

1.一种排程器（36），包括：

经配置以储存第一指令运算的缓冲器（80）；

连接至该缓冲器（80）的电路（86），其中该电路（86）配置为保留在该缓冲器（80）内的该第一指令运算为被从所述缓冲器选择而用于发送的该第一指令运算，以及其中该电路（86）被配置为若该第一指令运算为不正确执行时，启动该第一指令运算的重新发送；其特征在于：

该排程器还包括连接至该电路（86）的地址缓冲器（90），其中该第一指令运算为第一内存运算，并且其中该地址缓冲器（90）经配置以储存由该第一内存运算所存取的第一地址，该地址缓冲器经连接以接收该第一地址，并且其中接续该第一地址运算，该地址缓冲器（90）经连接以接收储存内存运算的第二地址，并且其中该地址缓冲器（90）经配置以比较该第二地址与储存在该地址缓冲器（90）内的地址，并且其中若该储存内存运算更新由该第一指令运算所存取的至少一个字节及该储存内存运算在程序顺序中是位于该第一内存运算之前，则该第一指令运算为不正确执行。

2.如权利要求 1 项的排程器，其中该排程经连接以接收来自执行单元（42）而显示该第一指令运算为不正确执行的第一信号，并且其中该电路（86）经配置以重新发送该第一指令运算以响应于该第一信号，并且其中第一信号更显示该第一指令运算为非臆测地执行，并且该电路（86）经配置以延迟重新发送该第一指令运算直到该第一指令运算为非臆测的。

3.如权利要求 1 项的排程器，其中该排程经连接以接收来自执行单元（42）而显示该第一指令运算为不正确执行的第一信号，其中该电路(86)被配置为启动该第一指令运算的重新发送以响应于该第一信号。

4.如权利要求 1 项的排程器，其中该地址缓冲器（90）经连接以接收表示数据将传送至数据闪存（44）的填覆地址，并且其中若该第一

地址显示该数据将传送至该数据闪存内的数据，则该电路(86)被配置为启动该第一内存运算的重新发送。

5.如权利要求 1 项的排程器还包括经连接以接收对应于储存内存运算的储存标记的标记缓冲器（92），其中该第一指令运算为第一内存运算，其中该储存内存运算更新由该第一内存运算所存取的至少一个字节并且该储存内存运算在程序顺序中是位于该第一内存运算之前，其中该标记缓冲器（92）经配置以储存该储存标记于对应于该第一内存运算的项目内以响应于执行该第一内存运算。

6.如权利要求 5 项的排程器，其中该标记缓冲器（92）还经连接以接收对应于执行储存的第二储存标记，并且其中该标记缓冲器（92）经配置以比较该第二储存标记与该储存标记，并且其中该电路经配置以重新发送该第一内存运算以响应于该储存标记等于该第二储存标记。

7.一种处理器（10），包括：

如权利要求 1 至 6 项的任何一项的排程器（36）；以及

连接至该排程器的执行单元（42）；该执行单元经配置以执行该第一指令运算。

8.一种用于发送及重新发送附属链接的方法，所述方法包括：

由排程器（36）发送第一指令运算以供执行；

接续该发送而在该排程器（36）内保留该第一指令运算；

由该排程器（36）重新发送该第一指令运算以供执行以响应于该第一指令运算为不正确执行。其特征为：

该第一指令运为第一内存运算；该方法还包括：

在地址缓冲器（90）内储存该第一指令运算的第一地址；

接续发送该第一指令运算而发送在程序顺序中位于该第一指令运算之前的储存内存运算；

比较在该地址缓冲器（90）内的该第一地址与对应于该储存内存运算的储存地址；以及

若该比较显示该储存内存运算更新由该第一内存运算所存取的至

少一个字节，则侦测该第一指令运算为不正确执行。

9.如权利要求 8 所述的方法，其中该第一指令运算为第一内存运算，该方法还包括：

在地址缓冲器（90）内储存该第一指令运算的第一地址；

比较表示数据将传送至数据闪存（44）的填覆地址；以及

若该第一地址显示在该数据内的数据将传送至该数据闪存时，则重新发送该第一内存运算。

10.如权利要求 8 所述的方法，其中该第一指令运算为第一内存运算，该方法还包括：

储存对应于储存内存运算的储存标记于对应于该第一内存运算的标记缓冲器（92）的一个项目内，该储存内存以编程顺序运算更新由该第一内存运算所存取的至少一个字节；

比较对应于执行储存的第二储存标记与在该标记缓冲器（92）内的该储存标记；以及

重新发送该第一内存运算以响应于显示相等的该比较。

能够发送及重新发送附属链接的排程器、包括该排程器的处理器以及排程方法

技术领域

本发明涉及处理器的领域，尤其涉及在处理器内的指令排程机制。

背景技术

超纯量（superscalar）处理器尝试通过在每个时钟周期发送及执行多重指令以及通过使用与设计相符的最高可能时钟频率以达到高效能。一种用于增加每个时钟周期所执行的指令数目的方法为脱序（out of order）执行。在脱序执行中，指令可以在不同于程序序列（或“程序顺序”）所记载的顺序下执行。在程序序列中的彼此相近的某些指令可具有禁止它们同时执行的附属，同时在程序序列内的后续的指令可不具有在先前指令上的附属。因此，脱序执行可通过增加同时间（在平均上）所执行的指令数目来增加该超纯量处理器的效能。另一种关于脱序执行的方法为臆测（speculative）执行，其中指令是接续其它可导致程序执行进入不同于含有该臆测的指令的路径来执行。例如，若该指令是接续可能造成例外的特定的指令，指令是可以臆测的。若指令是接续一个仍然尚未执行的预测条件的分支指令，该指令亦是可臆测的。同样地，指令可以脱序或臆测地排程、发送等等。

遗憾的是，用于脱序或臆测执行的排程指令对于该处理器呈现额外的硬件复杂度。该术语“排程”一般意指选择一个用于执行的指令。通常，该处理器尝试尽可能迅速地安排指令以最大化该平均指令执行速率（例如通过执行脱序的指令以处理用于各种指令形式的附属及硬件可利用性）。这些复杂度可能限制该处理器可以运算的时钟频率。尤其，在指令之间的附属必须由该排程硬件所关联。通常，于此所使用的术语“附属”意指第一指令及后续的第二指令之间以程序顺序的关联，该顺序要求该第一指令的执行早于该第二指令的执行。附属的变化是可以定义的。例如，若该第二指令的来源操作数（source operand）

是该第一指令的目标操作数（destination operand），来源操作数附属将发生。

通常，指令可能具有一个或一个以上的来源操作数及一个或一个以上的目标操作数。该来源操作数是依据指令定义以产生一个或一个一以的结果（该结果为目标操作数）的待处理的输入值。来源及目标操作数可以是储存在内存位置内而外接至该处理器的记忆操作数，或者可以是储存在寄存器储存位置而包含于该处理器内的暂存操作数。由该处理器所使用的指令集架构定义许多架构寄存器（architected registers）。这些寄存器经定义通过该指令集架构而存在，并且指令可以经由编码以使用该架构寄存器而成为来源及目标操作数。一个指令通过在该指令的运算元内的寄存器数目（或寄存器地址）来记载特定的寄存器而成为一个来源及目标操作数。该寄存器数目在该架构寄存器中唯一确认该所选择的寄存器。来源操作数通过来源寄存器数目所确认并且目标操作数通过目标寄存器数目所确认。

除了操作数属性之外，一种或一种以上形式的顺序附属可以由处理器来强制。顺序附属例如可以用于简化所使用的硬件或产生正确的程序执行。通过强迫某些指令相对于某些其它指令而按顺序执行，用于处理该指令的脱序执行的结果的硬件可以省略。例如，更新含有一般处理器运算状态的特殊寄存器的指令可能影响大量并未明显存取该特殊寄存器的后续的指令的执行。通常，顺序附属可能由微小架构至微小架构而改变。

当该排程机制与附属有关联时，极为可能需要在脱序及／或臆测排程指令上尝试最大化该效能增益的实现。然而，愈积极的排程机制（意即可能阻碍特定指令受到排程的较少的指令），愈有可能使不正确地执行指令产生。用于不正确执行指令的回复机制通常已经排除（purge）了该不正确执行指令及所有来自处理器管线（pipeline）的后续的指令并且重新提取（refetch）该不正确执行指令（及后续的指令）。通常，对于简单的硬件该排除及重新提取将延迟自不正确指令的发现（例如直到该不正确地执行指令成为最旧的指令而丢弃）。每个时钟周期的平均指令数量的实际执行因为该不正确执行及该后续排除的事件而减少。对于时常遭遇不正确执行的积极的排程机制，可归因于这

些回复机制的效能降低可能具有相当份量。因此，通过积极臆测或脱序排程而让用于保留效能增益的不正确臆测执行的回复的机制成为可能是需要的。

美国专利案号 5,987,594 说明一处理器，该处理器使用接收编码指令及发送该指令以供执行的指令排程单元以执行编码指令。当数据回传至该闪存并且指令附于该内存运算时，该处理器重新发送在闪存内失误的内存运算。此外，该处理器通过侦测“写入后读取（read after write）”尝试（hazards）及传递该写入数据给该读取指令可执行内存运算的脱序执行。

发明内容

上文所列举的问题可全部由本文所描述的排程器所解决。该排程器发送用于执行的指令运算，但是亦保留该指令运算。若特定的指令运算后来发现为不正确地执行，该特定指运算可以由该排程器重新发送。好处是相较于自管线排除该特定指令操较新的指令运算以及提取该特定指令运算，可以减少执行指令运算的不正确排程的代价。效能可以因为对于该不正确执行的减少而增加。再者，由于对于不正确执行的代价是减少的，该排程器可使用更多积极的排程机制。

此外，该排程器对于每个已经发送的指令运算可以维持附属指示。若该特定的指令运算是重新发送的，附属该特定指令运算（直接地或非直接地）的指令运算可以经由附属指示来确认。该排程器亦重新发送该附属指令运算。以程序顺序接续于特定指令运算但是并未附属该特定指令运算的指令运算将不再重新发送。因此，对于指令运算的不正确排程的代价可以比该特定指令及所有后来的指令运算的排除以及提取该特定指令运算更为减少。效能因此更为增加。

广义地说，排程器是经过充分考虑的。该排程器包括经由配置用于储存第一指令运算的指令缓冲器、连接至该指令缓冲器的发送选取（pick）电路以及控制电路。该发送选取电路经由配置以选择来自该指令缓冲器而用于发送的第一指令运算。连接至该发送选取电路的控制电路经由配置以维持该第一指令运算的第一执行状态。该控制电路经由配置以改变该第一执行状态成为响应于该发送选取电路选择用于发

送的该第一指令运算的执行状态。此外，该控制电路经由配置以改变该第一执行状态成为响应于指示该第一指令运算是不正确地执行的第一信号的不执行状态。

此外，处理器经过考量包括排程器及执行单元。该排程器经由配置以储存第一指令运算及发送用于执行的指令运算。该排程器经由配置以维持对应于该第一指令运算的第一执行状态，并且经由配置以改

变该第一执行状态成为响应于发送该第一指令运算的执行状态。连接至该排程器以接收该第一指令运算以响应由该排程器所发送的第一指令运算，该执行单元经由配置以执行该第一指令运算。该控制电路经由配置以该第一执行状态成为响应于指示该第一指令运算是不正确地执行的第一信号的不执行状态。再者，计算机系统经过考量包含该处理器及输入／输出（I／O）组件，该组件经由配置而对该I／O组件所连接的该计算机系统与另一部计算机系统之间做沟通。

再者，本发明所提供的方法是经过充分考虑的。第一指令运算由排程器发送至执行单元。该第一指令运算接续该发送而保留于排程器内。第一信号经接收为该第一指令运算执行不正确。该第一指令运算重新发送以响应接收的该第一信号。

再者，本发明所提供的处理器是经过充分考虑的。该处理器包括排程器及执行单元。该排程器经由配置以储存第一指令运算及发送用于执行的该第一指令运算。该排程器接续发送而经由配置以保留该第一指令运算，并且经连接以接收指示该第一指令运算是不正确地执行的第一信号。响应于该第一信号，该排程器经由配置以重新发送该指令运算而响应该第一信号。连接至该排程器以接收该第一指令运算以响应由该排程器所发送的第一指令运算，其中该执行单元经由配置以执行该第一指令运算。

附图说明

在读下列详细说明及参考附图后，本发明的其它目的及优点将变得显而易见，其中：

图1为处理器的一项实施例的方块图。

图2为可以由图1所显示的该处理器的一项实施例所使用的例示性管线图。

图3为图1所显示的说明该对映单元、排程器执行核心及加载／储存单元的一项实施例的较详细的方块图。

图4为图1及图3所显示的排程器的一项实施例的方块图。

图5为附属向量的一项实施例的方块图。

图6为附属缓冲器的一项实施例的方块图。

图 7 为图 6 所显示的部分附属缓冲器的一项实施例的较详细的方块图。

图 8 为关于在该排程器的一项实施例内的一个指令运算的状态机图。

图 9 为在该排程器的一项实施例内的说明用于每个指令运算所储存的状态讯息的方块图。

图 10 为说明附属链接的取消的时序图。

图 11 为说明来自该排程的一项实施例的指令运算的发送及重新发送的时序图。

图 12 为说明来自该排程的一项实施例的指令运算的发送及非臆测的重新发送的时序图。

图 13 结合用于运算该项目的例示性逻辑，为图 4 所显示的在实体地址缓冲器的一项实施例内的例示性项目的示意图。

图 14 结合用于运算该项目的例示性逻辑，为图 4 所显示的在储存确认缓冲器的一项实施例内的例示性项目的示意图。

图 15 为响应储存地址符合该加载地址的重试加载的一项实施例的时序图，及后续附属运算的取消。

图 16 为图 1 所显示的包含该处理器的计算机系统的第一项实施例的区块图。

图 17 为图 1 所显示的包含该处理器的计算机系统的第二项实施例的区块图。

由于本发明易于各种修正及替代形式，本发明的特定的实施例通过图式中的范例来显示并且将于此详细描述。然而，应该了解的是对于本发明的图式及详细说明并非意在限定本发明于该所揭露的特定形式，而相反地，是意在含括所有落在由附加的申请专利范围所定义的本发明的精神及范围内的修正、等同及替代。

具体实施方式

处理器概要

翻页至图 1，该图显示处理器 10 的一项实施例的方块图。其它实施例是可能的及经过考量的。在图 1 的实施例中，处理器 10 包含路线

预测器 12、指令闪存（I-闪存）14、对准单元 16、分支预测 / 提取 PC 产生单元 18、多个译码单元 24A-24D、预测器失误译码单元 26、微码（microcode）单元 28、对映单元 30、隐退队列（retire queue）32、架构改名文件（architectural renames file）34、未来文件 20、排程器 36、整数寄存器文件 38A、浮点暂存器文件 38B、整数执行核心 40A、浮点执行核心 40B、加载 / 储存单元 42、数据闪存（D-闪存）44、外部接口单元 46 及 PC 筒仓（silo）48。路线预测器 12 连接至预测器失误译码单元 26、分支预测 / 提取 PC 产生单元 18、PC 筒仓 48 及对准单元 16。I-闪存 14 连接至对准单元 16 及分支预测 / 提取 PC 产生单元 18，该分支预测 / 提取 PC 产生单元 18 更连接至 PC 筒仓 48。对准单元 16 更连接至预测器失误译码单元 26 及译码单元 24A-24D。译码单元 24A-24D 更连接至对映单元 30，并且译码单元 24D 连接至微码单元 28。对映单元 30 连接至隐退队列 32（该隐退队列连接至架构改名文件 34）、未来文件 20、排程器 36 及 PC 筒仓 48。架构改名文件 34 连接至未来文件 20。排程器 36 连接至寄存器文件 38A-38B，该寄存器文件更连接至个别的执行核心 40A-40B。执行核心 40A-40B 更连接至加载 / 储存单元 42 及排程器 36。执行核心 40A 更连接至 D-闪存 44。加载 / 储存单元 42 连接至排程器 36、D-闪存 44 及外部接口单元 46。D-闪存 44 连接至寄存器文件 38。外部接口单元 46 连接至外部接口单元 52 及 I-闪存 14。于此由图式标号伴随着字母所参照的组件将通过单独的图式标号来集合参照。例如，译码单元 24A-24D 将集合参照为译码单元 24。

在图 1 的实施例中，处理器 10 使用可变位组长度、复杂指令集计算（complex instruction set computing，CSIC）指令集架构。例如，处理器 10 可使用该 x86 指令集架构（也意指为 IA-32）。其它实施例可使用其它指令集架构，包含固定长度指令集架构及精简指令集计算（reduced instruction set computing，RISC）指令集架构。显示于图 1 的某些特征在此类架构下可以省略。此外，若有需要，任何上述的实施例可使用 64 位架构。

分支预测 / 提取 PC 产生单元 18 经由配置以提供提取地址（提取 PC）给 I-闪存 14、路线预测器 12 及 PC 筒仓 48。分支预测 / 提取 PC 产生单元 18 可包含用于辅助提取地址的产生的适当的分支预测机制。

响应于该提取地址，路线预测器 12 提供对应于多个指令的对准数据给对准单元 16，并且接续由该提供的指令数据所确认的指令可以提供用于提取指令的下一个提取地址。该下一个提取地址可以提供给分支预测 / 提取 PC 产生单元 18 或者依所需可以直接提供给 I-闪存 14。分支预测 / 提取 PC 产生单元 18 可接收来 PC 筒仓 48 的捕捉地址（若捕捉受到侦测）并且该捕捉地址可包括由分支预测 / 提取 PC 产生单元 18 所产生的提取 PC。此外，该提取 PC 可使用该分支预测数据以及由路线预测器 12 的数据来产生。通常，路线预测器 12 储存对应于由处理器 10 预先臆测所提取的指令的数据。在一项实施例中，路线预测器 12 包含 2K 项目，每个项目找寻（locating）一个或一个以上之于此意指为指令的路线的指令群组。该指令的路线可以通过透过放置于排程器 36 内的处理器 10 的指令处理管线来同时处理。

I-闪存 14 为用于储存指令字节的高速高速缓存。依据一项实施例 I-闪存可包括，例如，128 千字节（Kbyte），使用 46 元组闪存路线的四向集合联合组织。然而，任何 I-闪存构造将是合适的（包含直接对映构造）。

对准单元 16 接收来自路线预测器 12 的指令对准数据及对应于来自 I-闪存 14 的提取地址的指令字节。对准单元 16 依据该提供的指令运算数据选择指令字节进入每个译码单元 24A-24D。更明确地说，路线预测器 12 提供对应于每个译码单元 24A-24D 的指令指针。该指令指针找寻用于传送至该对应的译码单元 24A-24D 的该提取指令字节内的指令。在一项实施例中，某些指令可以传送至超过一个译码单元 24A-24D。因此，虽然其它实施例可包含较多或较少译码单元 24 以提供较多或较少的指令于路线内，在该实施例的显示中，来自路线预测器 12 的指令的路线可包含约四个指令。

译码单元 24A-24D 译码所提供的指令，并且每个译码单元 24A-24D 对应于该指令产生确认一个或一个以上的指令运算（或 ROPs）的数据。在一项实施例中，每个译码单元 24A-24D 对个指令可产生约二个指令运算。如同这里所使用的，指令运算（或 ROP）为执行核心 40A-40B 内的执行单元经由配置以执行为单一实体的运算。简单的指令可符合单一指令运算，而较多复杂指令可符合多重指令运算。某些

较多复杂指令可以于微码单元 28 内实现作为微码程序（在本实施例中是透过译码单元 24D 从只读存储器提取）。再者，其它实施例对于每个指令可使用单一指令运算（例如在此类实施例中，指令及指令运算可以同步）。

PC 筒仓 48 储存该提取地址及用于每个指令提取的指令数据，并且当排除时负责重新指向指令提取（诸如由处理器 10 所使用的指令集架构所定义的指令捕捉）。PC 筒仓 48 可包含用于储存提取地址的环状（circular）缓冲器及对应于多重指令路线的指令数据，而该指令在处理器 10 内可能尚待解决。响应于指令路线的隐退，PC 筒仓 48 可舍弃该相对应的项目。响应于排除，PC 筒仓 48 可提供捕捉地址给分支预测 / 提取 PC 产生单元 18。隐退及排除数据可以由排程器 36 所提供。在一项实施例中，对映单元 30 指派一系列数目（R#）给每个指令以确认在处理器 10 内指令待解决的顺序。排程器 36 可回复 R#s 给 PC 筒仓 48 以确认指令运算经验排除或隐退指令运算。

当在路线预测器 12 上侦测失误时，对准单元 16 从 I-闪存 14 至预测器失误译码单元 26 递送对相对应的指令字节。预测器失误译码单元 26 译码该指令，强制任何限制于指令的路线上，如同处理器 10 所设计（例如指令运算的最大数目、指令的最大数目、终止分支指令等等）。当终止路线时，预测器失误译码单元 26 提供该数据给路线预测器 12 以供储存。需注意的是预测器失误译码单元 26 可以经由配置以发送当经过译码的指令。此外，预测器失误译码单元 26 可译码指令数据的路线并且提供该数据给路线预测器 12 以供储存。接着，该失误提取地址可以于路线预测器 12 内重新尝试并且碰撞（hit）可以受到侦测。

除了当路线预测器 12 的失误而译码指令外，若由路线预测器 12 所提供的该指令数据为无效的，预测器失误译码单元 26 可以经由配置以译码指令。在一项实施例中，处理器 10 并未尝试保持与 I-闪存 14 内的指令耦合的数据于路线预测器 12 上（例如当指令在 I-闪存 14 内受到置换或无效时，可以不需主动地使该相对应的指令数据成为无效）。当侦测到无效指令数据时，译码单元 24A-24D 可确认所提供的指令数据，并且可发送信号给预测器失误译码单元 26。依据一项特定的实施例，该下列的指令运算由处理器 10 所支持：整数（包含计算、

逻辑、调换／循环及分支运算）、浮点（包含多媒体运算）及加载／储存。

该译码的指令运算及来源与目标寄存器数目将提供给对映单元30。对映单元30经由配置通过指派实体寄存器数目（PR#s）给每个指令运算的每个目标寄存器操作数及来源寄存器操作数以执行寄存器改名。该实体寄存器数目确认在寄存器文件38A-38B内的寄存器。对映单元30通过提供更新指派给该指令运算的来源操作数的每个实体寄存器数目的指令运算的R#s以额外地提供用于每个指令运算的附属的指示。基于该对应的逻辑寄存器数目，对应单元30以指派给每个目标寄存器（以及该对应的指令运算的R#）的实体寄存器数目来更新未来文件。此外，对映单元30储存该目标寄存器的逻辑寄存器数目、指定的实体寄存器数目及在隐退队列内的该先前指定的实体寄存器数目。当指令隐退时（由排程器36所指示的对映单元30），隐退队列32更新架构改名文件34并且释放任何将不再使用的寄存器。因此，在架构寄存器文件34内的该实体寄存器数目确认该实体寄存器储存处理器10的该托付架构状态，同时未来文件20显示处理器10的臆测状态。换言之，架构改名文件34储存对应于每个逻辑寄存器的实体寄存器数目，而显示用于每个逻辑寄存器的托付寄存器状态。未来文件20储存对应于每个逻辑寄存器的实体寄存器数目，而显示用于每个逻辑寄存器的该臆测的寄存器状态。

依据由对映单元30所指定的R#s，该指令运算路线、来源实体寄存器数目及目标实体寄存器数目储存至排程器36内。再者，对于特定指令运算的属性可以视为在储存于该排程器内的其它指令运算上的属性。在一项实施例中，指令运算仍存在于排程器36内直到隐退。

排程器36储存每个指令运算直到该属性显示指令运算已经符合为止。响应于用于执行的排程特定指令运算，排程器36可判断在那一个时钟周期特定的指令运算将更新寄存器文件38A-38B。在执行核心40A-40B内的不同的执行单元可使用不同数目的管线行程（并且因此为不同的延迟）。再者，某些指令可能在管线内比其它指令经历更多的延迟。因此，倒数计时将产生用以量测该特定指令运算的延迟（时钟周期的数目）。排程器36等候该特定数目的时钟周期（直到该更新

将早于或与该附属指令运算读取该寄存器文件时同时发生），并且接着视该特定指令运算可以排程时指示指令运算。需要注意的是一旦本身附属已经符合时（意即在该排程队列内相对于本身顺序的脱序），排程器 36 可排程指令。整数及加载 / 储存指令运算依据由寄存器文件 38A 的来源实体寄存器数目读取来源操作数并且传送至执行核心 40A 以供执行。执行核心 40A 执行该指令运算并且更新指派给在寄存器文件 38A 内的该目标的实体寄存器。此外，执行核心 40A 视该指令运算（如果有）至排程器 36 而回报该指令运算的 R#及排除数据。寄存器 38B 及执行核心 40B 对于浮点指令运算可以类似的方式运算（并且可提供用于浮点储存的储存数据给加载 / 储存单元 42）。需注意的是若操作数所依赖的运算为同时完成的，用于附属运算的操作数可以直接绕过而至该附属运算。

在一项实施例中，执行核心 40A 例如可包含二个整数单元、分支单元及二个地址产生单元（具有相对应的转换后备缓冲器（translation lookaside buffers），或 TLBs）。执行核心 40B 可包含浮点 / 多媒体乘法器、浮点 / 多媒体加法器及用于传送数据加载 / 储存单元 42 的储存数据单元。执行单元的其它配置是可能的，包含复合浮点 / 整数执行核心。

加载 / 储存单元 42 提供接口给 D-闪存 44 以执行内存运算及排程对于失误 D-闪存 44 的内存运算的填入运算。加载内存运算可以通过执行核心 40A 执行地址产生及传送数据给寄存器文件 38A-38B（由 D-闪存 44 或在加载 / 储存单元 42 内的储存队列）来完成。当通过执行核心 40A 的地址产生时，储存地址可以引入 D-闪存 44（直接透过在执行核心 40A 及 D-闪存 44 之间的连接）。该储存地址为分配储存队列登入。依据设计选择，该储存资料可以同时地提供，或可以接续地提供。当该储存指令的隐退时，该数据储存于 D-闪存 44 内（虽然在 D-闪存 44 的隐退与更新之间可能有某些延迟）。此外，加载 / 储存单元 42 可包含失误 D-闪存 44 用于后续闪存填入（通过外部接口单元 46）及重新尝试该失误加载 / 储存运算的用于储存加载 / 储存地址的加载 / 储存缓冲器。加载 / 储存单元 42 经由配置以处理加载 / 储存内存附属。

D-闪存 44 为通过处理器 10 所存取用于储存数据的高速高速缓存。

而 D-闪存 44 可包括任何适当的构造(包含直接对映及集合联合构造)，D-闪存 44 可包括 128 千字节，具有 64 字节路线之二向集合联合闪存。外部接口单元 46 经由配置透过外部接口 52 以与其它组件沟通。再者，外部接口单元 46 亦可执行由处理器 10 所产生的非闪存的读取及写入。

接着翻页至图 2，该图为例示性管线图，说明一组可以由处理器 10 所显示的一项实施例所使用的例示性管线行程。其它实施例可使用不同的管线，管线包含比图 2 所显示的管线更多或更少的管线行程。该图 2 所显示的行程由垂直线所界定。每个行程为在处理器 10 中的用于时钟储存组件（例如寄存器、闩锁、开关（flip）及类似组件）的时钟信号的其中一个时钟周期。

如图 2 的说明，该例示性管线包含 CAM0 行程、CAM1 行、路线预测（LP）行程、指令闪存（IC）行程、对准（AL）行程、译码（DEC）行程、对映 1（M1）行程、对映 2（M2）行程、写入排程（WR SC）行程、读取排程（RD SC）行程、寄存器文件读取（RF RD）行程、执行（EX）行程、寄存器文件写入（RF WR）行程及隐退（RET）行程。某些指令在该执行状态内使用多重时钟周期。例如，内存运算、浮点运算及整数乘法运算以分解形式说明于图 2 中。内存运算包含地址产生（AGU）行程、转换（TLB）行程、数据闪存 1（DC1）行程及数据闪存 2（DC2）行程。同样地，浮点运算包含约四个浮点执行（FEX1-FEX4）行程，以及整数乘法包含约四个（IM1-IM4）行程。

在该 CAM0 及 CAM1 行程期间，路线预测器 12 对分支预测 / 提取 PC 产生单元 18 所提供的提取地址与预测器内所储存的路线的地址做比较。此外，该提取地址在该 CAM0 及 CAM1 行程期间经转换由虚拟地址（例如在 x86 架构内的路线地址）成为实体地址。响应于在该 CAM0 及 CAM1 行程期间所侦测到的碰撞，该对应的路数据在该路线预测器行程期间由该路线预测器所读取。而且，I 闪存 14 在该路线预测器行程期间初始化读取（使用该实体地址）。该读取在该指令闪存行程期间完成。

需要注意的是，当图 2 所说明的管线使用二个时钟周期以侦测提取地址而在路线预测器 12 内的碰撞时，其它实施例可使用单一时钟周期（及行程）以执行此运算。再者，在一项实施例中，路线预测器 12

提供对于碰撞的用于 I-闪存 14 及下一个进入路线预测器 12 的下一个提取地址，并且因此该 CAM0 及 CAM1 行程可以略过来自路线预测器 12 的先前碰撞的提取。

由 I-闪存 14 所提供的指令字节在该对准行程期间通过对准单元 16 的对准以译码单元 24A-24D 以响应来自路线预测器 12 所对应的路线数据。需要注意的是某些指令可能需对准超过一个译码单元 24A-24D。译码单元 24A-24D 译码该提供的指令，在该译码行程期间确认对应于该指令的 ROPs 以及操作数数据。对映单元 30 在该对映 1 行程期间产生自于该提供数据的 ROPs，并且执行寄存器改名（更新未来文件 20）。在该对映 2 行程期间，该 ROPs 及指定的改名记录于隐退队列 32 内。再者，每个 ROP 所附属的 ROPs 将会判定。每個 ROP 可以为附属早先记录于未来文件内的 ROPs 的寄存器，并且亦可显示其它附属形式（例如在先前序列指令的附属）。

该产生的 ROPs 在该写入排程行程期间写入排程器 36 内。直到此行程，由数据的特定路线所找寻的该 ROPs 以一个单位流入（flow through）该管线。需注意的是由于该 ROPs 可以由该微码 ROM 于多重时钟下读取，包括微码程序的 ROPs 对于先前所提的陈述可以是一个例外。然而，接续写入至排程器 36，该 ROPs 在不同时间内可以单独流入该剩余的行程。通常，特定的 ROP 留在此行程直到由排程器 36 选择以供执行（例如在该特定 ROP 所附属的 ROPs 已经选择用于执行后，如上文所述）。因此，特定的 ROP 在该写入排程器写入行程及该读取排程器行程之间可以经历一个或一个以上的时钟周期延迟。在该读取排程器行程期间，参与在排程器 36 内的选择逻辑的该特定的 ROP，经选择以供执行，以及由排程器 36 读取。该特定的 ROP 接着在该寄存器文件读取行程中由其中一个寄存器文件 38A-38B（视 ROP 形式而定）执行读取寄存器文件运算。

该特定的 ROP 及操作数提供给该对应的执行核心 40A 或 40B，并且该指令运算在该执行期间是在操作数上执行。如上文所述，某些 ROPs 具有数个执行的管线行程。例如，内存指令运算（例如加载及储存）透过地址产生行程（其中由该内存指令运算所存取的内存位置的数据地址将产生）、转换行程（其中由该地址产生行程提供的该虚拟

数据地址将转换）及一对数据闪存行程（其中 D-闪存 44 将存取）来执行。浮点运算可使用约四个时钟周期来执行，并且整数乘法器可类似地使用约约四个时钟周期来执行。

当完成该执行程行或多个行程时，该特定的 ROP 在该寄存器文件写入行程期间更新本身指定的实体寄存器。最后，在每个之前的 ROP 隐退后（在该隐退行程中），该特定的 ROP 将隐退。再者，对于特定 ROP，在该寄存器文件写入行程及该隐退行程之间的一个或一个以上的时钟周期可能经过。再者，如同此项技艺中所习知的，特定的 ROP 可以在任何行程中因为管线延迟（stall）条件而延迟。

排程器

今翻页至图 3，该图显示对映单元 30、未来文件 20、排程器 36、整数执行核心 40A 及加载 / 储存单元 42 的说明的一项实施例的方块图。特定的例示性内连接以及该单元而非排程器 36 的一项实施例的特定内部细节说明于图 3 中。其它实施例是可能的并且经过考量的。在图 3 的实施例中，对映单元 30 连接至译码单元 24A-24D、未来文件 20 及排程器 36。排程器 36 更连接至外部接口单元 46、整数执行核心 40A 及加载 / 储存单元 42。在图 3 的实施例中，对映单元 30 包含目标改名电路 60、内部路线附属检测电路 62、顺序附属电路 64、一组顺序附属寄存器 66A-66N 及多任务器（mux）68。目标改名电路 60、内部路线附属检测电路 62 及顺序附属电路 64 连接至来自译码单元 24A-24N 的接收指令运算。目标改名电路 60 连接至多任务器 68 及排程器 36。内部路线附属检测电路 62 连接至多任务器 68，该多任务器更连接至未来文件 20。未来文件 20 连接至对应于由对映单元 30 所接收的指令运算的接收来源操作数确认器。顺序附属电路 64 连接至顺序附属寄存器 66A-66N 及排程器 36。加载 / 储存单元 42 包含经由连接以接收来自整数执行核心 40A 的实体地址的储存队列 70。整数执行核心 40A 包含连接至转换后备缓冲器（TLB）40AB 的地址产生单元 40AA。

通常，对映单元 30 接收来自译码单元 24A-24D 的指令运算。对映单元 30 对于每个指令运算执行寄存器改名并且判定在排程器 26 内等待（或同时发送至排程器 36）的位在旧的指令运算上的每个指令运算

的附属性。对映单元 30 提供指令运算及寄存器改名给排程器 36 以供储存（及后来用于执行的发送）。因此，对映单元 30 提供用于每个指令运算的附属性的指示（如图 3 所显示的该来源操作数附属及该顺序附属）。尤其，对映单元 30 通过 R#确认该旧的指令运算（该数目确认在排程器内部的指令运算）。指定给该操作数的实体寄存器的 PR# 提供给排程器 36 以供具有该指令运算的发送，但是并未用在判定该附属。排程器 36 储存该指令操作数及对应的附属，并且排程该指令运算以响应该符合的对应的附属。该排程指令运算将发送至经由配置以执行该指令运算的具有执行资源的执行核心 40A-40B。

某些指令运算当发送时可能未完全执行。例如，在所显示的一项实施例中，内存运算可能未完全执行。若指令运算未完全执行，该指令运算将通过包含在该指令运算的执行内的单元所“隐退”。重试指令运算包含发送该指令运算将隐退的信号给排程器 36。排程器 36 保留发送的指令运算，并且若该发送指令运算是隐退的，则排程器 36 重新发送该指令运算。尤其在一项实施例中，排程器 36 维持用于每个指令运算的执行状态。响应于先前发送指令运算的重试，排程器 36 重置该指令运算的执行状态成为“未执行”状态。接着，该指令运算可以重新发送。此外，程器 36 保留每个发送指令运算的附属。任何直接或非直接附属于该隐退的指令运算的指令运算也将回传至该未执行的状态。需要注意的是一群组的指令运算，其中该第一群组指令运算附属于特定指令运算并且其中每个于该群组内的其它指令运算附属于其中一个其它指令运算及透过其它指令运算于此是意指为“附属链接”的非直接附属于该特定的指令运算。重该执行状态而成为非执行的以响应该指令运算的重试或该指令运算是直接或非直接附属的另一个指令运算于此亦意指为“取消”该指令运算。

通过允许重试（以及重新发送以响应该重试）指令运算，排程器 36 可积极地排程指令运算以供执行并且可于较晚的时间通过重新发送该不正确地排程指令运算从不正确的排程来回复。用于不正确排程的代价实质上可以低于排除该不正确地排程指令运算及所有后来的指令运算以及在该不正确地排程指令运算上开始重新提取。

对映单元 30 使用目标改名电路 60、内部路线附属检测电路 62、

未来文件 20 及顺序附属电路 64 以判定用于每个指令运算的附属。目标改名电路 60 接收用于每个指令运算的指示，该指示为是否该指令运算具有寄存器目标操作数以及若该指令运算确实具有寄存器目标操作数时是否具有目标寄存器数目。若该指令运算具有寄存器目标操作数，目标改名电路 60 指定空的实体寄存器数目给该指令运算。该指定的 PR#s 具有该指令运算提供给排程器 36。此外，目标改名电路 60 提供每指令操作数的 R#s 及 PR#s 给工器 68。

对于每个来源操作数寄存器数目，未来文件 20 提供最近具有相对应的架构寄存器作为目标操作数的指令运算的 PR#及 R#。尤其，未来文件 20 可包括用于每个架构寄存器的项目的对照表（以及在实施例中使用微码，每个微码临时暂存）。该来源操作数寄存器数目用于选择由该指令运算的来源操作数所详载的寄存器的项目。每个项目储存该最旧的指令运算的 R#（在指令运算的目前的路线之前）以更新指定给该最旧的指令运算的目标的实体寄存器的寄存器及 PR#。此外，未来文件 20 在每个进入包含有效位（V）。该有效位表示是否记录于该寄存器的 R#是有效的（意即是否该对应的指令运算在排程器 36 内依然有效）。该有效位设定为对应于该 R#至排程器 36 的指令运的发送，并且当该指令运算隐退时重置。当该项目经选择为该来源操作数附属时，该有效位将提供给排程器 36。若该有效位清除时，排程器 36 并未记录用于该来源操作数的附属，并且若该有效位设置时，排程器 36 确实记录附属。

内部线路附属检测电路 62 接收每个指令运算的来源及目标寄存器数目并且由对映单元 30 所接收的指令运算的内部路线内执行附属检测。内部线路附属检测电路 62 比较该路线内的每个旧的指令运算的目标寄存器数目与该路线内的特定指令操作的来源寄存器数目。若发现其中一个来源操作数匹配时，内部线路附属检测电路 62 忽略来自对应于该来源操作数的未来文件 20 的 R#及 PR#与由目标改名电路 60 所提供的该对应的 R#及 PR#。若未发现匹配时，来自未来文件 20 的 R#及 PR#对于该来源操作数提供该正确的寄存器改名及附属 R#。内部线路附属检测电路 62 产生多任务选择路线给多任务器 68 以选择该适当的 R#及 PR#给每个指令运算的每个指令操作数。需要注意的是多任务器

68 对于选择该来源操作数附属可表示任何适当的选择电路。例如，多任务器 68 对于在该路线内的每个可能的指令运算的每个可能的来源操作数可表示个别的多任务。

外部路线附属检测电路更可以比较在该路线内用于每个指令运算的目标寄存器数目以判定在该路线内的最旧的指令运算以更新在该路线内的一个或一个以上的指令运算的目标操作数的每个架构寄存器。未来文件 20 在对应于该路线的目标操作数的项目中，接着可以通过目标改名电路 60 所指定的 R#s 与 PR#s 来更新。该更新路径为了图式简简便并未显示于图 3 中

顺序附属电路 64 追踪可以依据某些指令运算的顺序附属。例如，在使用该 x86 指令集架构的一项实施例中，顺序附属定义为：(i)部分加载，该加载造成用于每个后续内存运算的顺序附属；以及(ii)浮动控制字符更新，该更新造成用于每个后续浮点指令运算的顺序附属。通常，任何对后续指令运算的产生连续阻碍（serialization barrier）的指令运算导致从该连续指令运算至后续影响的指令运算的顺序附属。“连续阻碍”是脱序或臆测执行在该程序顺序附近处受到禁止的一种阻碍。某些指令集架构具有该指令的唯一功能在于提供该连续阻碍的指令。

该上文提及的顺序附属可以使用顺序附属寄存器 66A-66N 来追踪。响应于产生顺序附属的指令运算的顺序附属电路 64 储存该指令运算的 R#于其中一个顺序附属寄存器 66A-66N。其中一个顺序附属寄存器 66A-66N 可以提供给由处理器 10 所侦测的每个顺序附属。此外，有效位可以包含内并且可以经过设置以响应于当该对应的指令运算的重新进入时记录 R#及重置（类似于在末来文件 20 中的有效位）。响应于经定义而透过特定的顺序附属以顺序附属的指令运算，顺序附属电路 64 提供该对应的 R#成为其中一个该顺序附属给该指令运算。

除了上述特定情况外，顺序附属电路 64 可使用对照表以追踪先前在较旧的内存运算及后续发现以附属该较旧的储存记忆运算（由该加载所存取的该内存操作数）之前所排程的加载内存运算的发生。当该附属在执行期间受到侦测时，该对照表可包括由该加载内存操作数的提取地址所标示以及使用该较旧的储存内存运算的提取地址所训练的第一表格。第二表格由该储存内存运算的提取地址所标示，并且在以

该储存内存运算的R#的储存内存运算的发送时来更新。若该加载内存运算在对表中为碰撞时，该对应的R#将以顺序附属提供给该加载内存运算。

如上文所提及，排程器36排程及发送指令运算给适合的执行核心以响应侦测该指令运算的每个附属是满足的。尤其，内存运算是发送给在执行核心40A内的地址产生单元40AA。地址产生单元40AA接收来自整数寄存器文件38A所读取的寄存器操作数并且产生对应于该内存运算的内存运算的地址。该地址为虚拟地址，该地址透过由处理器10所使用的指令集架构所详载的地址转移技术转换成用于存取内存（及D-闪存44）的实体地址。TLB40AB是用于先前转换结果的闪存，允许碰撞的虚拟地址的快速转换成对应的实体地址并且透过该转换机制允许指定给该对应的内存位置的各种属性的快速判定。AGU40AA及TLB40AB的组合提供实体地址给加载/储存单元42（及以平行方式提供给D-闪存44及排程器36）。

加载/储存单元42判定是否该内存运算成功地完成执行或是隐退。若侦测到重试状况，加载/储存单元42维持该重试信号给排程器36并且透过该重试形式信号提供用于重试的理由。在一项实施例中，内存运算可以因下列的理由而重试：

(i)该内存运算为失误D-闪存44的加载内存运算；

(ii)该内存运算在该满的加载/储存单元42内需要缓冲器（意即通过外部接口单元46从主要内存用于储存待存取失误地址的失误缓冲器）；

(ii)该内存在D-闪存44内经历一组与另一个内存运算同时存取D-闪存的冲突；(iv)该内存运算是储存内存运算并且需要自我修正码（SMC）检测。

(v)该内存运算是在储存队列70内碰撞一个或一个以上储存内存运算的加载内存运算（意即该一个或一个以上储存内存运算供给至少一个由该加载内存运算所存取的内存操作数的字节）并且储存队列70无法传递相对应的数据；(vi)该记忆体操作是非臆测地执行。

理由(i)编码为个别的重试形式，对于该形式在排程及重新发送加载地址运算前，排程器36等待由外部介单元46所提供的匹配填覆地

址。外部接口单元 46 提供该填覆地址以显示正由该填覆地址至 D-闪存 44 来提供以供储存（并且因此该对应的加载内存运算在 D-闪存 44 内可能碰撞）。排程器 36 记录该加载内存运算（由执行核心 40A 所提供）的实体地址以供与该填覆地址比较。理由(ii)、(iii)及(v)可以编为单一重试形式，对于该形式排程器 36 通过重新排程该对应的内存运算而无任何特定的等待需求来响应。理由(iv)编码为重试形式并且在该 SMC 检测已经完成之后，排程器 36 可排程该对应的储存内存运算以供重新发送。理由(vi)编码为重试形式并且在该对应的内存运算变成非臆测时，排程器 36 排程用于重新发送的该内存运算。依据一项实施例，若该内存运算存取横跨页边界的内存操作数，内存运算将执行非臆测的（意即至少内存操作数的其中一个字节储存在由该第一地址转换所转换的第一页及至少内存运算的其中一个其它字节储存在由不同于第一地址转换的该第二地址转换所转换的第二页），该转换显示该内存运算的内存形式非臆测的，或该内存运算在该 TLB 内失误。用于非臆测地执行的该第一及最后理由为设计选择以简化该硬件，并且该中间的理由是由处理器 10 所使用的指令集架构所批准。

需注意的是，虽然上文描述意指非臆测地重新发送特定内存运算，其它指令运算可以非臆测地重新发送。例如，经过排除的任何指令运算（例如由该架构所详载的捕捉或错误或用于由处理器 10 所实现的该特定的微架构所定义的微架构排除）可以非臆测地重新发送。在此方式中，关于该排除的数据在该非臆测执行期间可以捕捉。因此，使用以储存及追踪排除数据的硬件数量可以最小化。

储存队列 70 视加载内存运算而提供额外的数据，该内存运算通过该碰撞及储存 R#信号在该储存队列内碰撞储存内存运算。该碰撞及储存 R#经由提供而不管是否该加载内存运算的重试发生。该碰撞信号显示在该储存队列内的碰撞受到侦测，并且该储存 R#为通过该加载所碰撞的储存的 R#。若由该加载所碰撞的储存接着重新执行时（并且接收不同的地址），此数据可以用于造成该加载内存运算的重试。该储存 R#的使用将于下文做更详细的描述。需要注意的是，当该储存 R#在此例子中使用时，任何确认该储存的识别器皆可使用。例如，可以提供确认由该加载所碰撞的在储存队列 70 内的储存队列进入的储存队列数

目。此类实施例是经过深思考量的。

如上文所必须注意的，储存队列 70 在储存队列 70 的加载内存运算碰撞储存内存运算的所有的例子中可能不能传送数据。例如，各种加载内存操作数的字节可以由在储存队列 70 内的不同的储存所提供。然而，储存队列 70 可限定个别储存的数目，来自该储存的特定的加载内存操作数的字节可以传送。例如，若储存队列 70 能够由约两个储存内存运算传送数据，对于该特定的加载记忆体的不同的字节的在三个或更多储存内存运算的碰撞避免该特定的加载内存操作数的所有的字节的传送。此外，储存队列 70 的某些实施例在接收该储存数据前可接收该储存内存操作数地址。若该储存数据无法获得，即使碰撞可能侦测到，储存队列 70 不能传送该储存数据。

需要注意的是于此意指为比其它指令运算“较旧”或“较新”的指令运算。若在程序顺序中该第一指令运算早于该第二指令运算，则第一指令运算“较旧”于第二指令运算。另一方面，若在程序顺序中该第一指令运算接续该第二指令运算，则第一指令运算“较新”于第二指令运算。如同在此所使用的，该术语“发送”意指传送指令运算给执行单元以供执行。该术语“重新发送”意指先前发送（并且发现为不正确执行，不论是直接透过重试或非直接透过由排程器 36 所记录的附属以供指令运算）的指令运算的发送。再者，该术语“记忆运算”于此是用于意指具有内存运算的指令运算。加载内存运算具有来源操作数作为来源操作数（以及寄存器目标操作数）并且详载来自该内存来源操作数的数据的转移而至该寄存器目标操作数。储存内存运算具有寄存器来源操作数及内存目标操作数，并且详载来自该寄存器来源操作数的数据的转移而至该内存目标操作数。需要注意的是，虽然图 3 说明地址产生单元 40AA 及相对应的 TLB40AB，各种实施例可包含何数目的地址产生单元及 TLBs。加载 / 储存单元 42 可提供个别的重试信号，重试形式信号、碰撞信号及储存 R#s 给对应于每个 AGU 的内存运算。

翻页至图 4，该图显示排程器 36 的一项实施例的方块图。其它实施例是可能的并且经过考量的。如图 4 所示，排程器 36 包含指令运算（ROP）缓冲器 80，发送选取电路 82、隐退限制选取电路 84、ROP

控制电路 86、附属缓冲器 88、实体地址缓冲器 90、储存电路缓冲器 92、隐退电路 94 及附属译码电路 96。ROP 缓冲器经由连接以接收指令运算（包含如同实时或位移数据的数据等等）及来自对映单元 30 的指定 PR#s 并且经由连接以提供发送指令运算及 PR#s 给寄存器文件 38A-38B 及执行核心 40A-40B。ROP 缓冲器 80 更经由连接至发送选取电路 82，该电路连接至 ROP 控制电路 86。隐退限制选取电路 84 连接至隐退电路 94 及 ROP 控制电路 86，该电路连接至隐退电路 94、附属缓冲器 88、实体地址缓冲器 90 及储存 R#缓冲器 92。ROP 控制电路 86 更经由连接以接收来自加载 / 储存单元 42 的重试及重试形式信号。附属译码电路 96 经由连接以接收来自对映单元 30 的来源附属 R#s 及顺序附属 R#s 并且连接至附属缓冲器 88。实体地址缓冲器 90 经由连接以接收来自外部接口单元 46 的填覆地址及来自执行核心 40A 的一个或一个以上的地址。储存 R#缓冲器 92 经由连接以接收来自加载 / 储存单元 42 的一个或一个以上的碰撞信号及一个或一个以上的储存 R#s。

附属译码电路 96 接收该 R#s 识别指令运算，在该指令运算上，写入至排程器 36 的每个指令运算是附属的并且译码该 R#s 项目用于该对应的指令运算的附属指示。如上文所提，若 R#显示为有效的（例如来自未来文件 20），则基于该 R#的附属并未显示。如同相对于提供的译码电路 96，对映单元 30 对于每个指令运算可直接产生附属指示（例如通过提供诸如图 5 所显示的附属向量给每个指令运算）。通常，附属指示指定给第一指令运算及第二指令运算，并且确认在该第二指指令运算上的第一指令运算的附属（或该指令运算的欠缺）。例如，每个附属指示可包括一个位表示当设置时的位在该第二指令运算上的第一指令运算的附属及表示当清除时的位在该第二指令运算上的第一指令运算的附属的欠缺。该位的设置及清除意义在一项实施例中可以保留，并且该附属指示的其它编码是可能的。

附属译码电路 96 提供该附属指示给附属缓冲器 88 以供储存。附属缓冲器 88 包括多重附属项目，每个该项目指定给在 ROP 缓冲器 80 内的两个项目。该附属项目储存该附属指示，该指示确认储存于两个在 ROP 缓冲器 80 内的其中一个的第一指令运算的附属或欠缺附属，而该缓冲器位在储存于该两个项目的另一个项目的第二指令运算上。

若该附属指示显示附属，则该第一指令运算并未准许排程直到该第二指令运算满足该附属。

ROP 控制电路 86 在监测于附属缓冲器 88 内的附属指示及该附属的符合，并且确认准许排程的指令运算。ROP 控制电路 86 确认该准许指令运算给发送选取电路 82，该电路扫描该准许指令运算以选择用于发送给该执行核心 40A-40B 的指令运算。选择指令运算由 ROP 缓冲器 80 所读取以响应发送选取电路 82，并且提供给寄存器文件 38A-38B 及执行核心 40A-40B 以供执行。通常，发送选取电路 82 经由配置以选择在每个执行核心 40A-40B 内的每个执行单元的指令运算（若该形式的指令运算准许用于排程）。该选择的指令运算为准许用于排程的形式的最旧的指令运算。在一项实施例中，发送选取电路 82 每周期二次扫描该准许指令运算以允许给定形式的二个指令运算的选择。该第二扫描选取用于发送给特定形式的第二执行单元的第二指令运算（意即二个地址产生单元及二个 ALUs 在一项实施例中提供给执行核心 40A）。在该第二扫描中，在该第一扫描期间所选择的指令运算将遮蔽（masked off）（意即显现不合格）以便该对应形式的第二最旧的指令运算可以选择。

在一项特定的实现中，发送选取电路 82 可包括用于每个指令形式的独立选取电路。每个选取电路以平行于其它取电路的运算可扫描用于该对应形式的指令运算。每个指令形式可使用来自其它指令形式的不同的执行资源（例如执行单元），允许该选取电路的独立运算。

发送选取电路 82 回报（给 ROP 控制电路 86）那一个指令运算已经选择以供发送。该选择指令运算意指作为排程，并且一旦指令运算由 ROP 缓冲器 80 读取时，该指令运算将发送（或重新发送）。ROP 控制电路 86 对每个指令运算维持执行状态。该执行状态可广泛地定义以含“不执行”状态、“执行”状态及“已实行（done）”状态。每个这些状态依据设计选择可包括多重状态，如同图 8 所显示的于例示性状态机的说明。指令运算一旦储存至排程器 36 内将视为不执行，直到该指令运算发送。该指令运算的执行状态改变为执行以响应于该发送，并且当完成执行时接着改变成已实行状态。若该指令运算为隐退（例如透过来自加载 / 储存单元 42 的该重试信号）或者若该指令运算

所附属（直接地或非直接地）的另一个指令运算为不实行时，该指令运算的执行状态在任何时刻可以改变成该不执行状态（或可以为不实行（undone））。若该特定指令运算具有不执行的执行状态并且若该特定指令运算的每个附属已经符合时，ROP 控制电路 86 通常可以确认特定指令运算为准许用于排程。

由于指令运算的执行状态改变为不执行以响应对于该指令运算的重试，该指令运算可变成准许排程及重新发送以响应该重试。然而，某些重试形式可能详载该指令运算未重新排程直到后续事件的发生（例如在加载内存运算的例子中所提供的填覆地址失误或该指令运算变成不臆测）。在此类例子中，ROP 控制电路 86 可改变该退隐 ROP 的执行状态成为不执行但是对于排程可以不发送该指令运算为准许的信号直到该后续事件发生时。

由于该附属指令并未由附属缓冲器 88 删除以响应于发送该对应的指令运算，当该附属变成符合时在附属链接内的指令运算可以臆测地发送。若该特定的指令运算为不实行时，在特定指运算上的其它指令运算的附属将重新分类为不符合，并且因此那些其它指令运算亦变成不实行。在此方式中，臆测地发送附属链接为不实行及重新发送以响应在该链接内的该第一指令运算的重试。

除了在加载内存运算的执行期间重试回报外，加载内存运算亦可因为接续该加载内存运算的较旧的储存内存运算发送而重试。实体地址缓冲器 90 提供用于检测这些重试流程。通常，加载内存运算并未显示（透过在附属缓冲器 88 内的该附属指示）为附属于较旧的储存内存运算。相反地，加载内存运算经由排程而不管较旧的储存内存运算（在实施例中使用上文所述的该顺序附属机制的排除）。然而，若该较旧的储存内存运算更新由该加载内存运算所存取的内存操作数的至少其中一个字节，加载内存运算可以附属于较旧的储存内存运算是可能的。为了侦测这些流程，实体地址缓冲器 90 储存由该加载（由执行核心 40A 所接收）所存取的实体地址。实体地址缓冲器 90 包含相同于 ROP 缓冲器 80 的项目的数目，每个项目能够储存用于加载内存运算的实体地址数据并且指定给在 ROP 缓冲器 80 内的对应的项目。对应于执行加载内存运算的项目使用该加载内存运算的实体地址来更新。

在储存内存运算的执行期间，通过该储存内存运算所新的实体地址由执行核心 40A 所提供。实体地址缓冲器 90 比较该储存地址与对应于较新的加载内存运算的在实体地址缓冲器 90 内的实体地址。换句话说，该址比较为较新于该执行储存内存运算的对应于指令运算的在实体地址缓冲器 90 内的该项目做遮蔽。若在加载地址上的储存地址的碰撞受到侦测，该对应加载内存运算为不实行（实体地址缓冲器 90 发送信号给 ROP 控制电路 86 指示该对应加载内存运算已经碰撞，并且 ROP 控制电路 86 改变该对应加载内存运算的执行状态成为不执行）。该对应的加载内存运算于后来重新发送。在该重新发送后的执行期间，该加载内存运算将碰撞在储存队列 70 中的该较旧的储存内存运算（并且该储存数据将传送或该加载内存运算隐退）或者该较旧的储存内存运算将必须更新该闪存及 / 或主要内存。在两者的例子中，该加载内存运算在重新发送后接收该正确的内存操作数并且成功地完成执行。需要注意的是在实施例中，若加载内存运算因为较旧的储存碰撞在实体地址缓冲器 90 内的对应的实体地址而为不实行时，该加载内存运算可以训练至在顺序附属电路 64 内的对照表中。

虽然实体地址缓冲器 90 在该加载内存运算附属的旧的储存内存运算之前提供来自加载内存运算的不正确排程的用于回复的机制，可能造成该加载内存运算变成不实行的该另一个问题可能存在。即使附属在该储存内存运算之后的该加载内存运算受到排程并且该储存数据由在加载 / 储存单元 42 内的该储存队列所传送，该储存内存运算本身可能变成不实行。该储存内存运算（使用于形成该储存内存运算的内存操作数的地址）的地址操作数在该重新发送期间（意即接收不正确的地址操作数可能是该重新发送的原因）可以是不同的，并且因此该储存地址在该重新发送执行期间可能不会碰撞实体地址缓冲器 90 并且造成该加载内存运算变成不实行。排程器 36 具备有储存 R#缓冲器 92 以处理此种可能性。

响应于侦测加载内存运算的碰撞于储存队列 70 内的储存，加载 / 储存单元 42 提供碰撞信号给排程器 36 及由该加载内存运算所碰撞的储存内存运的储存 R#。类似于实体地址缓冲器 90，储存 R#缓冲器 92 包含与 ROP 缓冲器 80 相同的项目数目。每个该项目指定给在 ROP 缓

冲器 80 内的对应的项目。若该碰撞信号维持用于执行加载内存运算，储存 R#缓冲器 92 储存由加载 / 储存单元 42 所提供的该储存 R#。

加载 / 储存单元 42 亦提供执行储存的 R#以储存 R#缓冲器 92。该储存 R#与储存于储存 R#缓冲器 92 的该 R#做比较。若侦测到符合时，储存 R#缓冲器 92 发信号给 ROP 控制电路 86 告知该对应的加载内存运算为未实行。ROP 控制电路 86 改变该对应的加载内存运算的执行状态成为不执行以响应该信号。接着，该加载内存运算重新排程及重新发送。需要注意的是若有需要，该储存 R#在该储存内存运算的执行期间可以由执行核心 40A 所提供。

除了侦测该储存以加载如上文所描述的附属，实体地址缓冲器 90 可以用于其它目的。例如，实体地址缓冲器 90 可以用于判定何时失误 D-闪存 44 的加载内存运算是欲重新发送。该加载内存运算是接续将由外部接口单元 46 所提供的该对应的数据来重新发送。因此，外部接口单元 46 提供填覆地址确认将提供给 D-闪存 44 的填覆数据。实体地址缓冲器 90 比较该填覆地址与储存在该缓冲器内的地址并且发送任何符合信号给 ROP 控制电路 86。在响应中，ROP 控制电路 86 记录用于加载内存运算的数据已经受到提供并且该加载内存运算可以重新排程（假设该加载内存运算的其它附属是符合的）。

外部接口单元 46 更可提供对应于在外部接口所接收的探测（probes）的探测地址。通常，若处理器 10 具有该闪存时钟的复制时，探测是用于维持在计算机系统内的闪存一致性以及记载由另一个组件所取得的闪存时钟及提供该闪存时钟的适当的闪存状态。若该探测地址碰撞在实体住址缓冲器 90 内的加载实体地址时，该对应的加载可能需要重新排程以维持一致性及由处理器 10 所使用的指令集架构所详载的该内存顺序原则。例如，该 x86 指令集架构详载清楚的内存顺序。因此，若先前内存运算存在于排程器 36 内并且具有不执行，由探测所碰撞的臆测的加载可能需要重新排程。

如同上文所描述，ROP 缓冲器 80 储存该指令运算并且发送该指令运算给寄存器文件 38A-38B 及响应于该发送选取电路 82 的执行核心 40A-40B。ROP 缓冲器 80 包括多个项目，每一个项目能够储存一个指令运算。指定给特定的指令运算的项目由该指令运算的 R#所确认。因

此，每个在 ROP 缓冲器 80 内的项目具有：(i)对应的第一指定集附属项目于附属缓冲器 88 内，该缓冲器储存指令运算的附属指示于排程器 36 内的其它指令运算上的该项目内；(ii)对应的第二指定集附属项目，该项目储存其它指令运算的附属指示于该项目的指令运算上的排程器 36 内；(iii)对应的实体地址缓冲器项目；以及(iv)对应的储存 R#缓冲器项目。总括来说，对应于给定的 R#的排程器 36 的各种缓冲器内的项目于此是意指为“排程器项目”。

隐退限制选取电路 84 及隐退电路 94 联合隐退来自排程器 36 的指令运算。ROP 控制电路 86 指示隐退限制选取电那一个指令运算具有已实行的执行状态。隐退限制选取电路 84 扫描来自排程器 36 的头部的指示（意即在排程器 36 内的最旧的指令）给具有未实行的执行状态的第一指令运算或者已经扫描及皆处于已实行状态的指令运算的预定的最大数目。隐退限制选取电路 84 因此判定可以隐退的该最新的指令运算。隐退限制选取电路 84 与可以隐退的最新的指令运算沟通，并且隐退电路 94 判定有多少指令运算实际上是隐退的。隐退电路 94 播送待隐退的最后的指令运算的 R#，并且与将隐退的指令运算的 ROP 控制电路 86 通沟。对于每个隐退指令运算，ROP 控制电路 86 使在 ROP 缓冲器 80、实体地址缓冲器 90 及储存 R#缓冲器 92 内部的该对应的项目成为无效。此外，对于每个隐退指令运算，ROP 控制电路 86 清除在该隐退指令运算上的指令运算的指示附属的附属缓冲器 88 内的每个附属项目。

如同于此所使用的，该术语“缓冲器”意指内存经由配置以储存数据的项目。该缓冲器可包含一个或一个以上的项目，每个该项目为包含充分储存以储存其中一个用于该缓冲器所设计的数据项的在该内存内的储存位置。

需要注意的是，虽然实体地址缓冲器 90 及储存 R#缓冲器 92 经由描述为具有相同于 ROP 缓冲器 80 的项目的数目，其它实施例可使用具有较少项目的缓冲器。每个在该缓冲器 90 或 92 的项目例如可包含确认在 ROP 缓冲器 80 内储存该对应的加载内存运算的该项目的标记（tag）。更需要注意的是，如同先前所提及的，该储存队列数目可以代替该储存 R#来使用以侦测那一个加载内存运算经侦测为碰撞的储存

内存运算的重新发送。

今翻页至图 5，该图显示附属向量 100 的一项实施例的方块图。附属向量 100 包含多个附属指示 102A-102N。每个附属指示 102A-102N 显示对应于在排程器 36 内的其中一个其它指令运算上的附属向量 100 的指令运算的附属（缺乏该附属）。该指令运算因此可以附属于任意数目的其它指令运算上。再者，由于附属为依据该指令运算而非附属的形式来记录，该附属可以以任何理由来产生（例如为了处理器 10 的简化设计）。如同上文所提及的，附属向量 100 可以通过对映单元 30 所提供的译码附属 R#s 及在附属向量 100 内设置对应的附属指示以显示附属以及设置该剩余附属指示以显示无附属而来产生。另外，对映单元 30 可提供图 5 所显示的形式的附属向量给排程器 36 以供储存。

翻页至图 6，该图显示附属缓冲器 88 的一项实施例的方块图。其实施例是可能的并且经过考量的。在图 6 的实施例中，附属缓冲器 88 包含含有附属项目 104A-104L 的多个附属项目。确认储存在排程器 36 的特定项目的特定指令运算的附属的该附属项目（意即在 ROP 缓冲器 80 内及在实体地址缓冲器 90 与储存 R#缓冲器 92 内的对应的项目）经由安排为附属项目之列及栏。每个附属项目之列储存详载在特定的排程器项目内的特定的运算的附属的附属指示。例如，确认在排程器项目 0 的指令运算的附属的附属项目是记录于附属项目 104A-104G 内（以及未显示的在该列内的中间媒介项目（intermediate entries））。显示于附属附属项目 104A-104G 的该例示性附属指示说明在排程器项目 N-2（附属项目 104F）的指令运算上的排程器项目 0 内的该指令运算的附属。再者，附属项目的每个栏详载在特定指令运算上的每个其它指令运算的附属。例如，在排程器项目 0 内的指令运算上的每个其它指令的附属是记录于附属项目 104H-104L 内。所显示的该例示性附属指示说明在排程器项目 0 内（附属项目 104I）的指令运算上的排程器项目 2 内的该指令运算的附属。

附属缓冲器 88 经由连接以接收一组输入信号（从 Block(0)至 Block(N-1)）。每个 Block 信号对应于于其中一个排程器项目。当信号植入时，该 Block 信号显示储存在该对应的排程器进内的指令运算具有未符合于该指令运算上的附属。当信号不植入时，该 Block 信号显

示在指令运上的该附属已经符合。通常，该信号在写入该对应的指令运算至排程器时将植入并且在该对应的指令运算执行期间不植入。若该指令运算是隐退或是其它变成不实行时，该 Block 信号将不植入直到该对应的指令运算为重新执行。该 Block 信号依据该对应的指令运算的执行状态而通过 ROP 控制电路 86 来植入及不植入。每个 Block 信号传递给记录在对应的指令运算上的其它指令运算的附属的附属项目。例如，Block(0)递送给附属项目 104H-104L。当该禁止信号为植入时，该对应的附属经由考量为符合。例如，当 Block(0)为植入时，在该排程器项目 0 的指令运算上的排程项目 2 内的指令运算的附属是符合的。

附属缓冲器 88 更提供多个输出信号（Not_Blocked(0)至 Not_Blocked(N-1)）。每个 Not_Blocked 信号对应于其中一个排程器项目。当植入该 Not_Blocked 信号表示储存于该对应的排程器项目的指令运算的附属已经符合。当不植入时，该 Not_Blocked 信号表示储存于该对应的排程器项目的指令运算的附属已经不符合。通常，该 Not_Blocked 信号是不植入的直到对应于该对应指令运算的附属的最后 Block 号为不植入，并且接着该 Not_Blocked 信号为植入的。该 Not_Blocked 信号是植入的对于指令运算是准许排程的，至少是对于该指令运算的附属（意即其它条件，诸如详载在后续事件上等待的重试形式，可以禁止排程）。每个 Not_Blocked 信号是递送给记录该对指令运算的附属的附属项目。例如，Not_Blocked 是递送给附属项目 104A-104G。每个该 Not_Blocked 信号可为预充电以植入并且接着由一个或一个以上的附属项目反植入的 wire-OR 路线，对于该路线该对应的 Block 信号是植入的并且该附属指示表示附属。

通过记录依据在该排程器内的指令运算的位置的附属（例如通过 R#）而相对于依据资源或附属理由，附属缓冲器 88 可以较容易在高频率下实现及运算。在附属缓冲器内的布线可以极为正规的（意即无该附属缓冲器的面积在布线上是阻塞的并且该布线甚少重叠）。该正规性容易实现并且对于高频率运算具有贡献（例如允许密集的附属缓冲器 88 的实现）。

需要注意的是如同图 6 所显示的来自上左至下右的在对角上的附

属项目将表示在本身上面的指令运算的附属。这些附属项目可能不实行（如同该表示那些项目的虚框的说明）。于此所使用的，该术语“植入”意指提供逻辑真值的信号或位。若其传送表示特定条件的值时，信号或位石以可以植入。当其传送逻辑零值，或者反过来，当其传送逻辑壹值时，信号或位可以定义成为植入，并且当该相反逻辑值传送时，该号或位可以定义为非植入。

今翻页至图 7，该图显示用于说明的更详细的部分附属缓冲器 88 及 ROP 控制电路 86 的一项实施例方块图。其它实施例是可能的且经过考量的。依据图 7 的实施例，ROP 控制电路 86 包括多个独立电路，每个该电路对应在排程器 36 内的项目。例如，在该排程器内的 entry(i)表示于图 7 中。ROP 控制 circuit(i)86A 说明用于追踪储存在 entry(i)内的指令运算的执行状态。因此，该图显示储存在 entry(i)内的指令运算的储存附属的数个附属项目 104M-104N。尤其，该图显示表示储存在 entry(i)的指令运算的附属于储存在 entry(j)的指令运算上（附属项目 104M）及储存在 entry(j+1)的指令运算上（附属项目 104N）。该 Block(i)及 Not_Blocked(i)信号以及该 Block(j)及 Block(j+1)信号皆显示于图中。ROP 控制 circuit(i)86A 经由连接以提供该 Block(i)信号并且经由连接以接收该 NOT_Block(i)信号。此外，ROP 控制 circuit(i)86A 经由连接以接收来自实体地址缓冲器 90 的 retry_PA(i)信号及 fill_hit(i)信号、来自外部接口单元 46 的填覆 / 探测信号、来自储存 R#缓冲器 92 的 retry_stq(i)信号、来自加载 / 储存单元 42 的重试信号及重试形式信号、来自执行核心 40A-40B 的 almost_done 信号及来自发送选取电路 82 的 pick(i)信号。再者，ROP 控制 circuit(i)86A 经由连接以提供 request(i)信号给发送选取电路 82。

当写入该指令运算至 entry(i)时，ROP 控制 circuit(i)86A 开始监测储存在 entry(i)的指令运算的附属。直到该指令运算已经满足在该指令运算上的其它指令运算的附属，ROP 控制 circuit(i)86A 植入该 Block(i)信号（该信号递送给记录在该指令运算上的其它指令运算的附属的该附属项目）。当该指令运算的执行状态是在不执行状态时，并且当该执行状态是在执行状态但是并未足够接近以完成执行以具有符合的附属时，该指令运算未具有符合的附属。因此，ROP 控制 circuit(i)86A

监测该 Not_Blocked(i)信号以判定何时该指令运算的附属已经符合。

储存在其它指令运算上的指令运算的附属指示的每个附属项目 104 经由连接以植入该 Not_Blocked(i)信号以表示该指令信号是禁止的。例如，该附属项目 104M 连接至 AND 栅极 106A 及晶体管 108A 并且附属项目 104N 连接至 AND 栅极 106B 及晶体管 108B。若该附属指示储存及该附属项目显示附属以及该对应的 Block 信号是植入的，该 AND 栅极驱动不植入该 Not_Blocked(i)信号的该对应的晶体管。另一方面，若该附属指示显示无附属或该 Block 信号为不植入，该 AND 栅极不驱动该对应的晶体管并且该晶体管不植入该 Not_Blocked(i)信号。因此，在该 entry(i)上的指令运算并未附属的该指令运算并未阻碍该指令运算的发送。在该 entry(i)上的指令运算是附属的该指令运算阻碍该指令运算的发送，直到该附属是符合的（由该对应的 Block 信号的不植入所表示）。

响应于该 Not_Blocked(i)信号为植入的，ROP 控制 circuit(i)86A 植入该 request(i)信号给发送选取电路 82。发送选取电路 82 扫描该 request(i)信号以及来自对应于其它项目的其它控制电路的类似信号。一旦发送选取电路 82 排程在 entry(i)内的该指令运算以供发送，发送选取电路 82 植入该选取(i)信号。响应于该 pick(i)信号，ROP 控制 circuit(i)改变该植入状态成为执行。如同上文必须注意，在本实施例中，排程器 36 记录该执行运算的隐藏（latency）及计算来自该指令运算的发送的时钟周期以判定附属符合的时刻点。其它实施例例如可接收来自执行单元的完成信号，或者使用任何其它额外的机制用于判定何时附属将符合。因此，在本实施例中，某些指令运算具有可变的隐藏或具有比欲计算较长的隐藏。对于此类指令运，执行核心 40A-40B 可提供 almost_done 信号。当该执核心判定可变的隐藏指令运算已经达到预定的来自完成的时钟周期的数目时，该 almost_done 信号将植入。该 almost_done 信号可以由 ROP 控制 circuit(i)86A 使用以开始计算周期，在该指令运算已经完成执行的时刻点达到该预定的数目。

若该指令运算为内存运算，ROP 控制 circuit(i)86A 在重试情况提供给储存在 entry(i)内的指令运算的时钟周期期间取样来自加载 / 储存单元 42 的该重试信号。响应于植入重试信号，ROP 控制 circuit(i)86 改

变该执行状态成为不执行并且重新植入该 Block(i)信号。在此方式中，该指令运算回传至预发状态并且在具有指令运算的附属链接内的后续的指令运算亦回传至预发送状态（透过该对应的 NOT_Blocked 信号的不植入）。此外，若该重试信号为植入时，ROP 控制 circuit(i)86A 取样该重试形式信号。若该试形式在该指令运算为重新发送前需要后续事件的发生，ROP 控制 circuit(i)86A 记录该待搜寻事件并且禁止请求重新发送（通过不植入该 request(i)信号）直到该后续事件发生。

除了在执行期间的隐退外，加载内存运算可以因为执行储存内存运算的实体地址碰撞该加载内存运算的实体地址（储存在实体地址缓冲器 90 内）或该执行储存内存算的 RR#碰撞记录用于该加载内存运算的储存 R#而隐退。实体地址缓冲器 90 植入 retry_PA(i)信号以将先前情况通联给 ROP 控制 circuit(i)86A（并且可包含用于每个其它项目的类似的信号）。储存 R#缓冲器 92 植入 retry_stq(i)信号以通联该后来的情况（并且可包含用于每个其它项目的类似的信号）。响应于每个信号的植入，ROP 控制 circuit(i)86A 改变该执行状态成不执行并且重新植入该 Block(i)信号。假设该 Not_Blocked(i)信号为植入，ROP 控制 circuit(i)86A 可植入该 request(i)信号以请求该指令运算的重新排程及重新发送。

除了该重试、retry_PA(i)及 retry_stq(i)信号外，若该 Not_Blocked(i)信号为不植入时，该指令运算的执行状态可以回传而不执行。当指令运算在该链接之一开始为不实行时，此机制用于取消附属链接的该已实行状态以造成在该附属链接内的指令运算的重新发送。因此，若该 Not_Blocked(i)信号为不植入时，ROP 控制 circuit(i)86A 改变该执行状态成为不执行并且重新植入该 Blick(i)信号（该信号后续可能造成其它 Not_Blocked 信号不植入，更进而取消该附属链接）。

实体地址缓冲器 90 提供额外的信号给 ROP 控制 circuit(i)86A 以指示是否由外部接口单元 46 所提供的地址碰撞在实体地址缓冲器 90 内的加载的实体地址，如同在图 7 所显示的 fill_hit(i)。实体地址缓冲器 90 植入该 fill_hit(i)信号以指示由外部接口单元 46 所提供的地址碰撞在实体地址缓冲器 90 内指定给 entry(i)的实体地址。外部接口单元 46 亦提供填覆／探测信号以指示所提供的地址的形式。若该填覆／探测信

号指示填覆，则该 fill_hit(i)的植入是指示包含该加载内存运算的实体地址的用于该闪存路线的填覆数据正在受到提供。若该加载内存运算因为在先前发送期间侦测闪存失误而由排程所禁止，该载入内存运算可以臆测用于排程并且 ROP 控制 circuit(i)86A 可植入该 request(i)信号以响应该填覆地址碰撞。上文所提及的实施例亦提供来自外部接口单元 46 的地址以完成探测。若该 fill_hit(i)信号为植入并且来自外部接口单元 46 的该填覆 / 探测信号表示探测时，则可能需要正确的动作的探测碰撞将受到侦测。在一项实施例中，用于探测的 fill_hit(i)的植入可造成 ROP 控制 circuit(i)86A 改变执行状态成为不执行。其它实施例可尝试更多详尽的机制以确保内存顺序而不会过度重新发送指令运算。例如，ROP 控制 circuit(i)86A 可通过该探测住址记录碰撞。若较旧的加载内存运算于后续由该排程器所隐退，则 ROP 控制 circuit(i)86A 可改变该执行状态为不执行。其它的选择亦是可能的。

翻页至图 8，该图显示可以由 ROP 控制 circuit(i)86A 的一项实施例所使用的例示性的状态机。其它控制电路可使用类似的状态机。其它实施例是可能的并且经过考量的。在图 8 的实施例中该状态机包含无效状态 110、禁止状态 112、请求状态 114、执行可变（ExecV）状态 118、Exec6 状态 120、Exec5 状态 122、Exec4 状态 124、Exec3 状态 126、Exec2 状态 128、Exec1 状态 130 及已执行状态 132。

当该对应的项目未储存于指令运算时，该状态机在该无效状态 110 下开始。响应于写入至该对应项目的指令运算，该状态机转移至禁止状 112 或请求状态 114。若该指令运算具有一个或一个以上的未符合的附属，禁止状态 112 将受到选择。易言之，若该 Not_Blocked(i)信号为不植入时，禁止状态 112 将受到选择，并且若该 Not_Blocked(i)信号为植入时，禁止状态 114 将受到选择。在其它实施例中，指令运算可以写入至具有预定的等待事件的排程器内，该事件阻碍来自待排程的指令运算即使所有附属是符合的（在指令运算已经回传至该不执行状态后，以一种类似可禁止重新排程的事件）。此类指令运算可造成对该禁止状态 112 的转移即使该 Not_Blocked(i)信号为植入的。

该状态机仍留在禁止状态 112 内直到该指令运算变成不禁止。虽然在本实施例中来自无效状态 110 至禁止状态 112 或请求状态 114 的

转移可以基于该Not_Blocked(i)信号，来自禁止状态112至请求状态114的转移考量重试情况的影响，在该指令运算是臆测用于排程之前，该情况详载后续事件的发生。在图 8 的区块 134 中包含用于用在上文描述的实施例的图 8 内的箭号上的禁止转移项目的判断式。尤其，若该Not_Blocked(i)信号是不植入的，指令运算是禁止的，或者先前发送将造成判定该指令运算为非臆测地执行（blocked_non_spec）并且仍然为臆测的，或者先前发送造成闪存失误（blocked_until_fill）并且填覆数据仍然未提供。依所需的其它实施例可包含禁止重新排程的额外的事件。一旦该指令运算不禁止时，该状态机由禁止状态 112 转移至请求状态 114。

当该状态机是在请求状态 114 内时，ROP 控制 circuit(i)86A 植入该 request(i)信号。若该指令运算再度变成禁止而在请求状态 114 时，该状态机转移至禁止状态 112。该状态机从请求状态转移至其中一个状态 118-128（基于该指令运算的隐藏）以响应该 pick(i)信号的植入。在一项实施例中该状态转移以响应该 pick(i)信号可对应于图 2 的管线的读取排程器行程。

本实施例支持二至六个时钟周期的隐藏，以及大于六个时钟周期的可变隐藏。该状态机仍保留在 ExecV 状态 118 内直到该 almost_done 信号由执行核心 40A-40B 所植入，并且接着转移至该 Exec6 状态 120。若该指令运算并未不实行，如图 8 所显示，每个 exec6 状态 120 从 exec2 状态 128 转移至下一个在该隐藏链接内的较低的状态。从 Exec1 状态 130，若该指令运算并未不实行时，该状态机转移至该已实行状态 132。最后，若该指令运算在隐退之前并未不实行，该状态机从已实行状态 132 转移至无效状态 110。

在图式中为了明确起见，该 pick(i)信号显示为将选取节点 116，从该节点中其中一项状态 118-128 将项目。选取节点 116 仅用于减少在图式中的混乱状态，并且并非意在表示分离的状态。

在本实施例中，该指令运算的隐藏为了图 8 的状态机的目的在该指令运算已经符合在该指令运算的附属之前为时钟周期的数目。此种隐藏在该指令运算回传执行状况数据之前可以终止（意即是否该指令运算经历了排除）。然而该状态机具有在指令运算受到排程与该指令

运算由寄存器文件 38A-38B 读取操作数之间的管线延迟以表示附属在该附属实际上为透过该寄存器文件的更新的实体附合之前是符合的。因此，若在本实施例中该指令运算已经达到该 Exec2 状态 128 时，该 Block(i)信号是不植入，并且若该状态机是在 Execl 状态 130、已实行状态 132 或无效状态 134（见区块 134）内，该 Block(i)信号保留不植入。该 Block(i)信号对于其它状态是植入的。

在任何已经排程（pick(i)植入）后的位置，该指令运算可变成不实行并且回传成不执行状态。此种操作通过每个状态 118-132 显示基于“取消”判断式（区块 134）转移成中央点位置（central point）136 而说明于图 8 中，从该位置转移至禁止状态 112 或请求状态 114 是基于说明于区块 134 内的禁止判断式而执行。中央点位置 136 仅用于减少在图式中的混乱，而非表示分离的状态。对于每个显示转移至中央点位置 136 的状态，若该取消判断式为真且该禁止判断式为真，则至禁止状态 112 的转移将执行，并且若该取消判断式为真且该禁止判断式为假，则至请求状态 114 的转移将执行。

在本实施例中，若该指令运算是直接隐退或者若该 Not_Blocked(i)信号变成不执入，则指令运算变成“不实行”（意即回传不执行的执行状态）。当在 entry(i)内的表示指令运算的 retry_this_op 值是隐退的，在区块 134 内的该取消判断式说明该重试条件。当若该 retry_PA(i)信号或 retry_stq(i)信号是植入的而判断式可能为真时，或者若该指令运算在执行期间是隐退时（例如来自加载 / 储存单元 42 的该重试信号），区块 138 更显示说明该 retry_this_op 值。当该执行运算是在 Excel 状态 130 时，该 retry_this_op 判断式更说明该重试信号的取样。在本实施例中，当该对应的指令运算是在 Excel 状态 130 时，重试状况将由加载 / 储存单元 42 回报。依据设计选择，其它实施例在指令运算的执行期间可回报在不同位置的情况。再者，依据设计选择，重试指令运算而非内存运算的实施例在该指令运算的执行内的其它位置可取样该重试信号。

如上文所提及的，指令运算的执行状态可广义地包含不执行、执行及已实行状态。对于图 8 的实施例，该不执行状态可包括禁止状态 112 或请求状态 114。该执行状态可包括执行状态 118-130。该已实行

状态可包括已实行状态 132。需要注意的是执行状态 118-130 的数目是特定实行的并且可以依据设计选择而改变。再者，在附属是符合的指令运算的执行内的位置可以依据设计选择而改变。该改变部分可以基于管线行程的数目，在该行程之间该附属指令运算是排程的并且位在该附属符合的特定行程，诸如操作数或顺序附属，是需要符合的。在本实施例中，该特定行程为该寄存器文件读取行程。

今翻页至图 9，该图显示可通过 ROP 控制 circuit(i)86A 所使用的寄存器 140 以储存图 8 的状态机的状态及所需要的附加状态。其它实施例是可能的并且经过考量的。在图 9 的实施例中，寄存器 140 可储存状态 142、blocked_non_spec 指示 144、blocked_until_fill 指示 146 及其它数据 148。

状态 142 储存在图 8 中所说明的状态机的电流状态。该状态可以在以何适当的方式在状态 142 内做编码。寄存器 142 依据在图 8 所说明的状态转移在每个时钟周期内更新。

若该重试形式显示该指令运算是将要非臆测地执行，则 Blocked_non_spec 指示 144 可以设定以指示禁止以响应在指令运算的执行期间接收来自加载 / 诸存单元 42 的重试信号。Blocked_non_spec 指示 144 可以用于在图 8 的区块 134 所显示的禁止判别式内。尤其，当该 Blocked_non_spec 指示 144 显示禁止时，该指令运算是由请求排程所禁止直到该指令运算变成非臆测的。响应于变成非臆测的指令运算，该 Blocked_non_spec 指示可以设定以显示不禁止并且该指令运算可以排程。在一项特定的实施例中，若在排程器 36 内的每个较旧的指令运算具有已实行的执行状态，则该指令运算变成非臆测的。

若该重试形式显示该指令运算失算 D-闪存 44，Block_until_fill 指示 146 可以设定以指示禁止以响应在指令运算的执行期间接收来自加载 / 诸存单元 42 的重试信号。该 Block_until_fill 指示 146 可以用于在图 8 的区块 134 所显示的禁止判别式内。尤其，当该 Block_until_fill 指示 146 显示禁止时，该指令运算是由请求排程所禁止直到该对应的填覆数据受到提供。响应于当受到提供而显示的填覆数据，该 Block_until_fill 指示可以设定以显示不禁止并且该指令运算可以排程。

其它数据可依所需在其它数据范畴 148 内可以记录。例如，特定

的实施例可以禁止重试储存运算直到 SMC 检测可以实行。其它数据范畴 148 可以记录该需求以等待该 SMC 检测并且可记录该 SMC 检测的完成。任何其它数据皆可记录。再者，无其它数据在内记录的实施例是经过考量的。

今翻页至图 10，该图显示依据排程器 36 的一项实施例的说明取消附属链接的例子的时序图。时钟周期通过垂直虚线界定范围，对于每个时钟周期在界定该时钟周期的垂直虚线之间的上端具有识别器。对于每个指令运算的状态（如同 ROP 控制电路 86 所记录）亦显示于图 10 中（接续于该文字“状态”及在括号中的该对应的指令运算的 R#），具有“已实行”表示已实行状态 132 及“blkd”表示禁止状态 112。图 10 包含说明二个附属链接的区块 150。该第一附属链接开始于指令运算 I0，指定 10 的 R#，并且更包含指定运算 I1、I2 及 I3。指令运算 I1 附属于 I0 并且具有具有 15 的 R#。指令运算 I2 附属于 I1 并且具有 23 的 R#。指令运算 I3 附属于 I2 并且具有 34 的 R#。指令运算 I4 是在第二附属链接内由 I0 所起始，并且因此附属于 I0。指令运算 I4 具有 45 的 R#。I1 及 I4 直接附属于 I0，而 I2 及 I3 是非直接附属于 I0。对于每个指令运算的 Block 及 Not_Blocked 信号说明于图 10 中（具有指令运算的 R#提供于括号中）。某些造成其它事件的事件是通过箭头从该事件至该结果事件来说明。例如，Not_Blocked(10)的不植入造成 State(10)以变成禁止，由箭头从该 Not_Blocked(10)的不植入至该 State(10)的禁止状态来说明。

在时钟周期 clk0 期间，每个指令运算是在该已实行状态内。因此，该对应的 Block 信号是不植入的并且该 Not_Blocked 信号是植入的。在该时钟周期 clk1 期间，该 Not_Blocked(10)信号是不植入的（因为在该一个或一个以上的指令运算上的 I0 附属变成不实行）。回戚于 Not_Blocked(10)的不植入，用于 I0（State(10)）的状态机回传至禁止状态，并且因此该 Block(10)信号在时钟周期 clk2 内是重新植入。响应于该 Block(10)的植入及 I0 及 I4 的附属于 I0，该 Not_Blocked(15)及 Not_Blocked(45)信号不植入（时钟周期 clk2）。该 Not_Blocked(15)及 Not_Blocked(45)信号的不植入依序导致 I1 及 I4 的取消（State(15)及 State(45)在时钟周期 clk3 内变成该禁止状态）。接着，I2 及 I3 因为它

们分别直接附属于 I1 及 I2 并且因此通过它们非直接附属于 I0 而取消。通过时钟周期 clk5 的结束，在说明的例子中的该附属链接已经不实行并且对应于每个指令运算（I0 至 I4）的执行状态是在不执行状态。接着，该指令运算可接收它们附属的符合并且可依据在该附属链接重新发出内的指令运算的依序重新发出以及符合在该附属链接内的其它指令运算的附属。

需要注意的是，虽然该 Block 及 Not_Blocked 信号是显示于图 10（及第 11、12 及 15 图下方），如同在特定的时钟周期期间的植入或不植入，该 Block 信号在该时钟周期的第一部分期间可以是不致发的以允许该 Not_Blocked 信号的预充电，并且接着该 Block 信号在该时钟周期的第二部分期间可以脉冲（并且 Not_Blocked 信号可以放电或仍然预充电，依据记录的附属）。再者，第 10、11、12 及 15 图的时序图说明基于所说明的 Block 信号的转移的 Not_Blocked 信号的转移。因此，在所说明的例子中，该说明的附属链接的附属是该最后附属而符合于在该附属链接内的每个指令运算。若其它附属仍然不符合时，该 Not_Blocked 信号将保留不植入直到该其它附属的符合。同样地，该时序图说明指令运算将实时地排程以响应对于在时序图内的简化的请求。然而，若该相同形式的其它较新的指令运算是请求排程时，该排程可以通过一个或一个以上的时钟周期来延迟。

今翻页至图 11，该图为说明在附属链接内的例示性指令运算的发送及重新发送的时序图，该重新发送因为在该附属链接内的第一指令运算的重试而发生。时钟周期通过垂直虚线所界定，具有用于在该垂直虚线界定该时钟周期之间的上端的每个时钟周期的识别器。区块 152 说明该例示性附属链接，该链接为来自图 10 的例子的从 I0 至 I2 的相同的指令运算及附属。该图说明用于每个指令运算的该 Block 及 Not_Blocked 信号以及每个指令运算的状态（由 ROP 控制电路 86 所记录），类似于图 10 的例子。说明于图 11 的该状态包含该禁止及已实行状态，由类似于图 10 的图 11 的“blkd”及“done”来表示。再者，请求状态 114、Exec2 状态 128 及 Exec1 状态 130 分别说明为“rqst”、“ex2”及“ex1”。又类似于图 10，造成其它事件的某些事件通过从该造成事件至该结果事件的箭头来说明。在此例子中，指令运算 I0 及

I11 两者皆是隐藏 2。

时钟周期 clk0 在时钟状态内说明每个指令运算 I0-I2，在变成准许发送前等待附属符合。每个该 Not_Blocked 信号是不植入的，并且每个 block 信号是植入的。在时钟周期 clk1 期间，Not_Blocked(10)植入。响应于 Not_Blocked(10)的植入，状态(10)在时钟周期 clk2 期改变成该请求状态。I0 经选择用于发送并且因此在时钟周期 clk4 内 state(10)转移至该 Exec2 状态。

响应于 state(10)的 Exec2 状态，Block(10)在时钟周期 clk4 期间是不植入的（接着依序造成 Not_Block(15)是植入的）。State(15)在时钟周期 clk4 内转移至该请求状态以响应 Not_blocked(15)的植入，并且转移至在时钟周期 clk5 内的 Exec2 状态以响应受到的选取。

在 state(10)的 exec1 状态期间（时钟周期 clk4），ROP 控制电路 86 侦测 I0 的重试（透过该重试（R#10）信号说明于图 10 内）。该重试造成 I0 的取消，并且因此 State(10)在时钟周期 clk5 内转移成不执行。尤其，由于该 Not_Blocked(10)信号在时钟周期 clk4 期间是执入的，State(10)转移至该请求状态。响应于 State(10)回传至不执行状态，该 Block(10)信号是重新植入（并且因此该 Not_Blocked(15)是不植入的。Not_Blocked(10)的不植入造成 State(15)的回传至不执行状态（时钟周期 clk6））。

在此例子中的 I0 的重试是允许 I0 的实时重新发送的重试形式。因此，状态（10）是在时钟周期 clk5 的请求状态内。I0 是用于执行的选取，并且因此 State(10)在时钟周期 clk6、clk7 及 clk8 内分别转移至该 Exec2、Exec1 及已实行状态。在 I0 的执行期间，重试并未发生。然而，需要注意的是该重试在特定的指令运算成功地完成之前可以多次发生。一旦 State(10)在 I0（时钟周期 clk6）的重新执行期间达到该 exec2 状态，该 Block(10)信号是不植入的并且该 Not_Blocked(15)信号是植入的。响应于该 Not_Blocked(15)信号的植入，State(15)转移至该请求状态（时钟周期 clk7）并且接着转移至 exec2 状态以响应用于发送的选择（时钟周期 clk8）。State(15)在时钟周期 clk9 及 clk10 内分别转移至该 exec1 及已实行状态。

响应于达到该 Exec2 状态（时钟周期 clk8）的 State(15)，该 Block(15)

信号是不植入的。该 Not_Block(23)信号在时钟周期 clk8 期间植入以响应 Block(15)的不植入，并且因此 State(23)转移至在时钟周期 clk9 内的请求状态。I2 的发送在后来的时钟周期（未显示）期间可能发生。

今翻页至图 12，该图为说明指令运算的重试的时序图，该指令运算具有需续事在该指令运算执行的排程前发生的重试理由。尤其，图 12 说明欲非臆测执行的指令运算的重试。时钟周期由垂直虚线所界定，具有用于在该垂直虚线界定该时钟周期之间的上端的每个时钟周期的识别器。区块 152 说明该例示性附属链接，该链接为来自图 11 的从 I0 至 I2 的相同的指令运算及附属。该图说明用于每个指令运算的该 Block 及 Not_Blocked 信号以及每个指令运算的状态（由 ROP 控制电路 86 所记录），类似于图 11 的例子。说明于图 12 的该状态是以类似于图 11 的方式来表示。又类似于图 11，造成其它事件的某些事件通过从该事件至该结果事件的箭头来说明。在此例子中，指令运算 I0 是隐藏 2。

时钟周期从 clk0 至 clk6 是类似于图 11 的从 clk0 至 clk6 的该对应的时钟周期，具有在时钟周期 clk4 内的 I0 的重试指示为重试的排除，因为 I0 是欲非臆测地执行。因此，I0 是非臆测地发送并且本身的非臆测特性在发送后而发现。排程器 36 通过取消 I0 来解决此种情况（以及本身的附属指令运算 I1 及 I2）并且在允许重新发送之前等待 I0 变成非臆测的。尤其，响应于该重试形式成为“等待非臆测的”，ROP 控制电路 86 可设定对应于 I0 的该 Blocked_non_spec 指示。因此，即使该 Not_Block(10)信号是植入的，ROP 控制电路 86 是由 I0 的请求排程来禁止直到 I0 变成非臆测的。时钟周期的某些数目可以经过，并且接着 ROP 控制电路 86 可判定 I0 是非臆测的（例如，在图 12 中的时钟周期 clkn，由在图 12 的该 non-spec(R#10)信号的植入来说明）。如上文所提及，依据一项实施例，当每个在排程器 36 内的先前的指令运算（在程序顺序中）是在该已实行状态内，指令运算可以是非臆测的。

响应于 I0 变成非臆测的，State(10)转移至该请求状态（时钟周期 clkn+1）。接着，I0 经选择用于发送（在时钟周期 clkn+2 内的 state(10)的 Exec2 状态）及执行。当它们的附属 I0 是符合时，附属指令运算 I1 及 I2 后续可以执行。

令翻页至图 13，该图显示可以由实体地址缓冲器 90 的一项实施例

所使用的例示性实体地址缓冲项目 160。此外，该图显示例示性组合逻辑电路 172。电路 172 可用以产生该 fill_hit(i)及 retry_PA(i)信号。其它实施例是可能的并且经过考量的。尤其，任何适当的组合逻辑电路皆可以使用，包含任何显示于图 13 的该逻辑的 Boolean 等式。再者，依据设计选择，储存于项目 160 的数据可以改变形式及内容。在图 13 的实施例中，项目 160 包含有效位 162、第一加载 PA 栏区 164、第一字节屏蔽栏区 166、第二加载 PA 栏区 168 及第二字节屏蔽栏区 170。

通常，若项目 160 指定给在指令缓冲器项目内的指令运算是加载内存运算，项目 160 是由加载内存运算（该“加载内存操作数”）存取的内存操作数的实体地址数据所更新，并且该合法的位 162 是设置的。在本实施例中，该数据是由包含该加载内存操作数（第一加载 PA 栏区 164）的第一字节的四方字符（quadword）的实体地址所表示并且字节屏蔽指示那一个在该四方字符内的字节是部分该加载内存操作数（第一字节屏蔽栏区 166）。该字节屏蔽包括用于在该四方位内的每个字节的位。若该位是设置时，该对应的字节为部分该加载内存操作数。若该位是清除时对应的字节并非部分该加载的内存操作数。

加载内存操作数在内存内可以意地对准。因此，该加载内存操作数的一个或一个以上的字节可以在其中一个四方字符内并且该加载内存操作数一个或一个以上的字节可以在下一个连续的四方字符内。因此，项目 160 提供第二加载 PA 栏区 168 及第二字节屏蔽栏区 170。第二加载 PA 栏区 168 储存该下一个连续的四方字符的实体地址至第一加载 PA 栏区 168。在本实施例中，该实体地址的页内部分储存在第二加载 PA 栏区 168 内。由于交错页边界的加载内存运算在本实施例中是非臆测的，所以必须仅储存该下一个续连续字符的该页内部分（由于若页是交错的，该加载内存运算将非臆测地重新发送并且因此无较旧的储存将会接续该加载内存运算的重新发送而发送）。其它实施例可储存该下一个连续的四方字符或任何所需的适当部分的全部。再者，虽然本实施例储存在四方字符分布特性（granulrity）上的地址，其它实施例可使用任何适当的分布特性（例如八方字符（octword）、双字符等等）。第二字节屏蔽栏区 170，类似第一字节屏蔽栏区 166，表示那一个在下一个连续的四方字符内的字节为部分该加载内存操作数。

执行核心 40A 在储存内存运算的执行期间提供该储存实体地址及对应的字节屏蔽。电路 172 比较对应的部分储存实体地址与储存在第一加载 PA 栏区 164 及第二加载 PA 栏区 168 的值。因此，对应的储存字节屏蔽将会提供。在电路 172 内接收接收储存及加载字节屏蔽的该 AND 栅极表示判断是否在该加载字节屏蔽内的至少一个位及在该储存字节屏蔽内的少一个对应的位的逻辑是设置的，表示该加载内存操作数的至少一个字节是由该储存内存所更新。例如，用于每个位的 AND 栅极、该栅极为 OR 的输出可以使用。若项目 160 是有效的，该实体地址部分符合及在该对应的四方字符内的至少一个字节是部分该加载内存操作数并且是由储内存运算所更新，接着该 retry_PA(i)信号可以产生。需要注意的是若该储存内存运算在程序顺序中并未早于该加载内存运算（未显示于图 3 中），该 retry_PA(i)信号亦可以屏蔽。

需要注意的是储存内存操作数在内存中亦可以任意地对准。因此，该储存内存操作数的一个或一个以上的字节可以在一个四方字符内并且该储存内存操作数之一或一个以上的字节可以在下一个连续的四方字符内。因此，该储存 PA+1（类似于该加载 PA+1）可以与该储存加载 Pas 进行比较以侦测储存 PA 碰撞该加载 PA。该下列的公式可表示该 Retry_PA(i)信号（其中若至少其中一个在 A(n:0)内的位是设置的并且在 B(n:0)内的对应的位是设置的，则该 MATCH(A(n:0), B(n:0))函式回传二位 1）：

Retry_PA(i)=V&Load_PA(39:12)==Store_PA(39:12)&((Load_PA(11:3) == Store_PA(11:3) & MATCH(Store_Byte_Mask(7:0), Load_Byte_Mask(7:0)))||

Load_PA(11:3)+1 == Store_PA(11:3) & MATCH(Store_Byte_Mask(6:0), Load_Byte_Mask(14:8)))||

Load_PA(11:3) == Store_PA(11:3)+1 & MATCH(Store_Byte_Mask(14:8), Load_Byte_Mask(6:0)))||

Load_PA(11:3)+1 == Store_PA(11:3)+1 & MATCH(Store_Byte_Mask(14:8), Load_Byte_Mask(14:8))))

更需要注意的是对于内存运算在下一个连续的四方字符内具有一个有效字节，由于该内存运算在该第一四方字符内具有至少一个有效字节（字节 7，由屏蔽位 7 所表示），该最后四项（比较 Load_PA(11:3)+1

及 Store_PA(11:3)+1）为多余的并且在本实施例中可以消去。因，此在该第四项内的符合只有在该第一项内的符合亦遇到时才会遇到（比较 Load_PA(11:3)及 Store_PA(11:3)）。

此外，项目 160 与由外部接口单元 46 所提供的填覆 / 探测地址做比较。在该说明的实施例中，在该填覆内提供的闪存路线的地址提供给实体地址缓冲器 90 以供比较。第一加载 PA 栏区 164 及第二加载 PA 栏区 168 的对应的部分可以与该填覆地址比较。若侦测至符合时，该 fill_hit(i)信号可以植入。在其它实施例中，该闪存路线可以提供给 D-闪存 44 作为多个封包。所提供的确认该闪存路线及封包的地址部分可此实施例中可以做比较。

仍然更需要注意的是若对应于该 store_PA 的储存内存运算是较新于对应该项目 160 的加载内存运算，该 Retry_PA(i)信号可以屏蔽。

今翻页至图 14，该图显示可以由储存 R#缓冲器 92 的一项实施例所使用的例示性储存 R#缓冲器项目 180。此外，该图显示例示性组合逻辑电路 190。电路 190 可以用于产生该 retry_stq(i)信号。其它实施例是可能的并且经过考量。尤其，任何适合的组合逻辑电路可以使用，包含任何第 14 所显示的逻辑的 Boolean 等式。再者，储存在项目 180 的数据依据设计选择可以在形式及内容上做变化。在图 14 的实施例中，项目 180 包含有效位 182 与 186 以及储存 R#栏区 184 与 188。

通常，若项目 180 所指定的位在指令缓冲器项目的指令运算为加载内存运算，项目 180 将以通过该加载内存运算所碰撞的该储存在储存队列 70 内的储存 R#来更新。本实施例提供由约二个储存内存运算至加载内存运算的传递，并且因此二个储存 R#栏区 184 及 188 经由提供以记录每个传递储存的 R#。若侦测到对应的传送储存，个别的有效位 182 及 186 将设置。其它实施例可仅由其中一个储存传送，并且项目 180 可仅记录一个储存储存 R#。其它实施例仍然可以由超过二个储存来传送，并且项目 180 可以记录储存 R#s 的对应的数目。

当储存内存运算执行时，加载 / 储存单元 42 提供该储存内存运算的 R#给储存 R#缓冲器 92。该 R#与储存在项目 180 内的 R#s 做比较，并且若侦测到符合时（并且该应的有效位为设置），电路 190 植入该 retry_stq(i)信号，如上文所提及，在另一个选择中，储存队列数目可以

储存在缓冲器 92 内并且储存队列数目可以提供以供比较。

翻页至图 15，该图为说明加载内存运算透过在实体地址缓冲器 90 内的碰撞的重试的时序图。加载内存运算透过在储存 R#缓冲器 92 内的碰撞的重试可以是相似。时钟周期由垂直虚线所界定，具有用于在该垂直虚线界定该时钟周期之间的上端的每个时钟周期的识别器。区块 192 说明该例示性附属链接，该链接为来自图 11 的例子的从 I0 至 I2 的相同的指令运算及附属（除了 I0 现为加载内存运算）。该图说明用于每个指令运算的该Block及Not_Blocked信号以及每个指令运算的状态（由 ROP 控制电路 86 所记录），类似于图 11 的例子。说明于图 12 的该状态以类似于图 11 的方式表示。此外，该 Exec4 及 Exec3 状态分别以“ex4”及“ex3”来说明。又类似于图 11，造成其它事件的某些事件通过从该造成事件至该结果事件的箭头来说明。在此例子中，加载内存运算 I0 是隐藏 4。

在时钟周期 clk0 上，每个指令运算 I0-I2 已经发送及执行，并且因此是在该已实行状态。对应的 Block 信号为不植入的并且 Not_Blocked 信号为植入的。然而，在实体地址缓冲器 90 上的碰撞是侦测为 R#10（retry_PA(10)在时钟周期 clk0 期间是植入的）。在响应上，State(10)在时钟周期 clk1 内转移至该请求状态。此外，该 Block(10)信号是植入的，并且 I1 及 I2 在时钟周期 clk2 至 clk3 期间接着是取消。

该加载内存运算 I0 经选择用于执行并且在时钟周期 clk2 至 clk6 内通过该执行状态至该已实行状态。响应于在时钟周期 clk4 内的 I0 达到该 Exec2 状态，该 Block(10)信号为不植入（并且因此该 Not_Blocked(15)信号变成植入）。如图 15 所显示，指令运算 I1 及 I2 因此重新排程及重新发送。

图 15 说明加载指令运算可以在该加载指令运算附属于储存内存运算之前发送及执行。接着，该储存内存运算可以发送并且该附属受到侦测。当附属的侦测时，该附属由来自排程器 36 的重新发送该加载内存运算（以及本身的附属链接）而响应。具有该 retry_stq(10)信号植入的类似的时序图说明在先前不正确执行及后续重新发送的储存内存运算上的加载内存运算的错误附属的侦测。再者，排程器 36 处理由重新发送该加载内存运算及本身的附属链接的情况。正确的运算可以提供

具有最少效能退化，并且因此积极的臆测执行可以执行并且较高的效能可以达到。

计算机系统

今翻页至图 16，该图为显示包含透过总线桥接器（bus bridge）202 连接至各种系统组件的处理器 10 的计算机系统 200 的一项实施例的方块图。其它实施例是可能的并且经过考量。在该显示的系统中，主存储器 204 透过内存总线 206 连接至总线桥接器 202，并且图形控制器 208 透过 AGP 总线 210 连接至总线桥接器 202。最后，多个 PCI 组件 212A-212B 透过 PCI 总线 214 连接至总线桥接器 202。第二总线桥接器 216 更可以透过 EISA / ISA 总线 220 提供搭配的电气接口给一个或一个以上的 EISA 或 ISA 组件 218。处理器 10 透过 CPU 总线 224 连接至总线桥接器 202 及连接至选择 L2 闪存 228。同时，CPU 总线 224 及该接口至 L2 闪存 228 可包括外部接口 52。

总线桥接器 202 在处理器 10、主存储器 204、图形控制器 208 及连接至 PCI 总线 214 的组件之间提供接口。当运算由其中一个连接至总线桥接器 202 的组件所接收时，总线桥接器 202 辨识该运算的目标（意即特定的组件，或者在 PCI 总线 214 的例子中，该目标在 PCI 总线 214 上）。总线桥接器 202 递送该运算至该目标组件。总线桥接器 202 通常传送来自由该来源组件或总线所使用的协议运算至由该目标组件或总线所使用的协议。

除了提供接口给用于 PCI 总线 214 的 ISA / EISA 总线，第二总线桥接器 216 更可依所需加入额外的功能。输入 / 输出控制器（未显示），不论来自外部或与第二总线桥接器 216 整合，亦可包含在计算机系统 200 内以提供算支持给需要的键盘及鼠标 222 以及各种序列及并行端口。外部闪存单元（未显示）在其它实施例中更可以连接至在处理器 10 及总线桥接器 202 之间的 CPU 总线 224。此外，该外部闪存可以连接至总线桥接器 202 并且用于外部闪存的闪存控制逻辑可以整合至总线桥接器 202。L2 闪存 228 更显示于处理器 10 的背面配置上。需要注意的是 L2 闪存 228 可以从处理器 10 分离，整合至具有处理器 10 的卡匣（例如槽 1 或槽 A）内，或甚至整合至具有处理器 10 的半导体基板

上。

主存储器 204 为应用程序储存在内的内存并且处理器主要由该内存来执行。适当的主存储器 204 包括 DRAM（动态随机存取内存）。例如，复数组 SDRAM（同步 DRAM）或 Rambus DRAM（RDRAM）可能是适合的。

PCI 组件 212A-212B 显示各种外围组件，例如诸如网络适配卡、影像加速卡、声卡、硬或软盘或插槽控制器、SCSI（小计算机系统接口（Small Computer System Interface））转接器及电话卡（telephony cards）。同样地，ISA 组件 218 显示各种外围组件的形式，诸如调制解调器、声卡及各种诸如 GPIB 或字段总线接口（field bus interface）卡的数据撷取卡。

图形控制器 208 经由提供以控制在显示器 226 上的文字及图形的显示。图形控制器 208 可包括一般在此项技艺中习知的典型的图像加速卡以显示可以有效至／由主存储器 204 转移的三维数据构造。图形控制器 208 因此可以为 AGP 总线 210 的主导者，其中该控制器可以请求及接收而存取在总线桥接器 202 内的目标接口藉以获得而存取主存储器 204。开放的图形总线搭配来自主存储器 204 的数据的快速检索。对于某些运算，图形控制器 208 更可以经由配置以在 AGP 总线 210 上产生 PCI 协议处理。总线桥接器 202 的 AGP 接口可因此包含支持 AGP 协议处理以及 PCI 协议目标与起始处理两者的功能。显示器 226 为影像或文字可以表示在其上的任何电子式显示器。适当的显示器 226 包含阴极射线管（cathode ray tube，“CRT”）、液晶显示器（liquid crystal display，“LCD”）等等。

需要注意的是，虽然该 AGP、PCI 及 ISA 或 EISA 总线已经使用作为在上述说明中的例子，任何总线架构皆可以依需求而取代。更需要注意的是计算机系统 200 可以是包含额外的处理器的多任务处理的计算机系统（例如处理器 10a 显示为计算机系统 200 的选择性的组件）。处理器 10a 可以类似于处理器 10。尤其，处理器 10a 可以是处理器 10 的相等的复制。处理器 10a 可以透过独立总线（如同在图 16 所显示）连接至总线桥接器 202 或者可与处理器 10 分享 CPU 总线 224。再者，处理器 10a 可以连接至类似于 L2 闪存 228 的选择性的 L2 闪存 228a。

今翻页至图 17，该图显示计算机系统 300 的另一项实施例。其它的实施例是可能的并且经过考量。在图 17 的实施例中，计算机系统 300 包含数个处理节点 312A、312B、312C 及 312D。每个处理节点透过包含在每个个别的处理节点 312A-312D 内的内存控制器 316A-316D 连接至个别的内存 314A-314D。此外，处理节点 312A-312D 包含用于在处理节点 312A-312D 之间通信的接口逻辑。例如，处理节点 312A 包含用于与处理节点 312B 通信的接口逻辑 318A、用于与处理节点 312C 通信的接口逻辑 318B 及用于与另一个处理节点（未显示）通信的第三接口逻辑 318C。同样地，处理节点 312B 包含接口逻辑 318D、318E 及 318F；处理节点 312C 包含接口逻辑 318G、318H 及 318I；并且处理节点 312D 包含接口逻辑 318J、318K 及 318L。处理节点 312D 透过接口逻辑 318L 经连接以与多个输入 / 输出组件通信（例如以雏菊花环（daisy chain）状配置的组件 320A-320B）。其它处理节点可与其它 I / O 组件以类似的方式通信。

处理器节点 312A-312D 实现用于内部处理节点通信的封包型（packet-based）连接。在本实施例中，该连接以数组单一方向的路线来实现（例如路线 324A 是用于从处理节点 312A 至处理节点 312B 传输封包并且例如路线 324B 是用于从处理节点 312B 至处理节点 312A 传输封包）。其它数组路线 324C-324H 是用于在其它其它处理节点之间传输封包，如同在图 17 中的说明。通常每组路线 324 可包含一个或一个以上的数据路线、对应于该数据路线的一个或一个以上的时钟路线及表示待传送的封装形式的一个或一个以上的控制路线。该连接可以以闪存耦合（coherent）方式操作而用于在处理节点之间通信或以非耦合方式运算而用于在处理节点及 I / O 组件之间通信（或为总线桥接器至习知结构的 I / O 总线，诸如 PCI 总线或 ISA 总线）。再者，该连接可以使用雏菊花环构造依所显示在 I / O 组件之间以非耦合方式运算。需要注意的是从其中一个处理节点至另一个处理节点的待传输的封包可能通过一个或一个以上的中间媒介的节点。例如，由处理节点 312A 至处理节点 312D 所传输的封包可能通过处理节点 312B 或处理节点 312C，如同图 17 所显示。任何适当的处理递送算法皆可以使用。计算机系统 300 的其它实施例可包含比图 17 中的实施例更多或更少的

处理节点。

通常，该封包可以以一个或一个以上的位时间在节点之间的路线324上传输。位时间可以是在对应的时钟路线上的时钟信号的上升或下降缘。该封包可包含用于初始化处理的命令封包、用于维持闪存一致性的探测封包及来自响应探测及命令的响应封包。

除了内存控制器及接口逻辑之外，处理器节点312A-312D可包含一个或一个以上的处理器。广义地说，处理器节点包括至少一个处理器并且可以选择性地包含用于与内存及其它所需逻辑通信的内存控制器。尤其，处理节点312A-312D可包括处理器10。外部接口单元46可包含在节点内的该接口逻辑318以及该内存控制器316。

内存314A-314D可包括任何适当的内存组件。例如，内存341A-314D可包括一个或一个以上的RAMBUS DRAMs（RDRAMs）、同步的DRAMs（SDRAMs）、静态的RAM等等。计算机系统300的地址空间是分开于内存314A-314D里。每个处理节点312A-312D可包含用于判定那一个地址是对映于那一个内存314A-314D的内存对映，并且因此对映于那一个处理节点312A-312D对于特定的地址内存请求应该来递送。在一项实施例中，对于在计算机系统300内的地址的一致的位置为连接至对应于该地址的内存储存字节的内存控制器316A-316D。换句话说，该内存控制器316A-316D负责确认在闪存耦合形式的每个内存存取该对应的内存控器314A-314D的发生。内存控制器316A-316D可包括用于与内存314A-314D接口接合的控制电路。此外，内存控制器316A-316D可包含用于排列内存请求的请求队列。

通常，接口逻辑318A-318L可包括用于由该连接接收封包及当连接时用于缓冲待传输的封包的各种缓冲器。计算机系统300可使用任何适当的流程控制机制以供传输封包。例如，在一项实施例中，每个接口逻辑318储存接口逻辑所连接的该连接的另一端的接收器内的缓冲器的每个形式的数目的计量。该接口逻辑并未传输封包除非该接收接口逻辑具有空的缓冲器以储存该封包。当接收缓冲器通过向前递送封包而清除时，该接收接口逻辑传送讯息给发送接口逻辑以显示该缓冲器已经清除。此类机制可以视为“配给型（coupon-based）”系统。

I/O组件320A-320B可以是任何适当的I/O组件。例如，I/O

组件 320A-320B 可包含网络适配卡、影像加速卡、声卡、硬式或软式磁盘驱动器或磁盘驱动器控制器、SCSI（小型计算机系统接口）转接器及电话卡、调制解调器、声卡及诸如 GPIB 或字段总线适配卡的各种数据撷取卡。

一旦完全了解上文的揭露后，对于熟习此项技艺的人士而言各种变化及修正将变得显而易见。本发明的权利要求应当视为意在含括所有此类的变化及修正。

产业可应用性

本发明适用于处理器及计算机系统的领域。

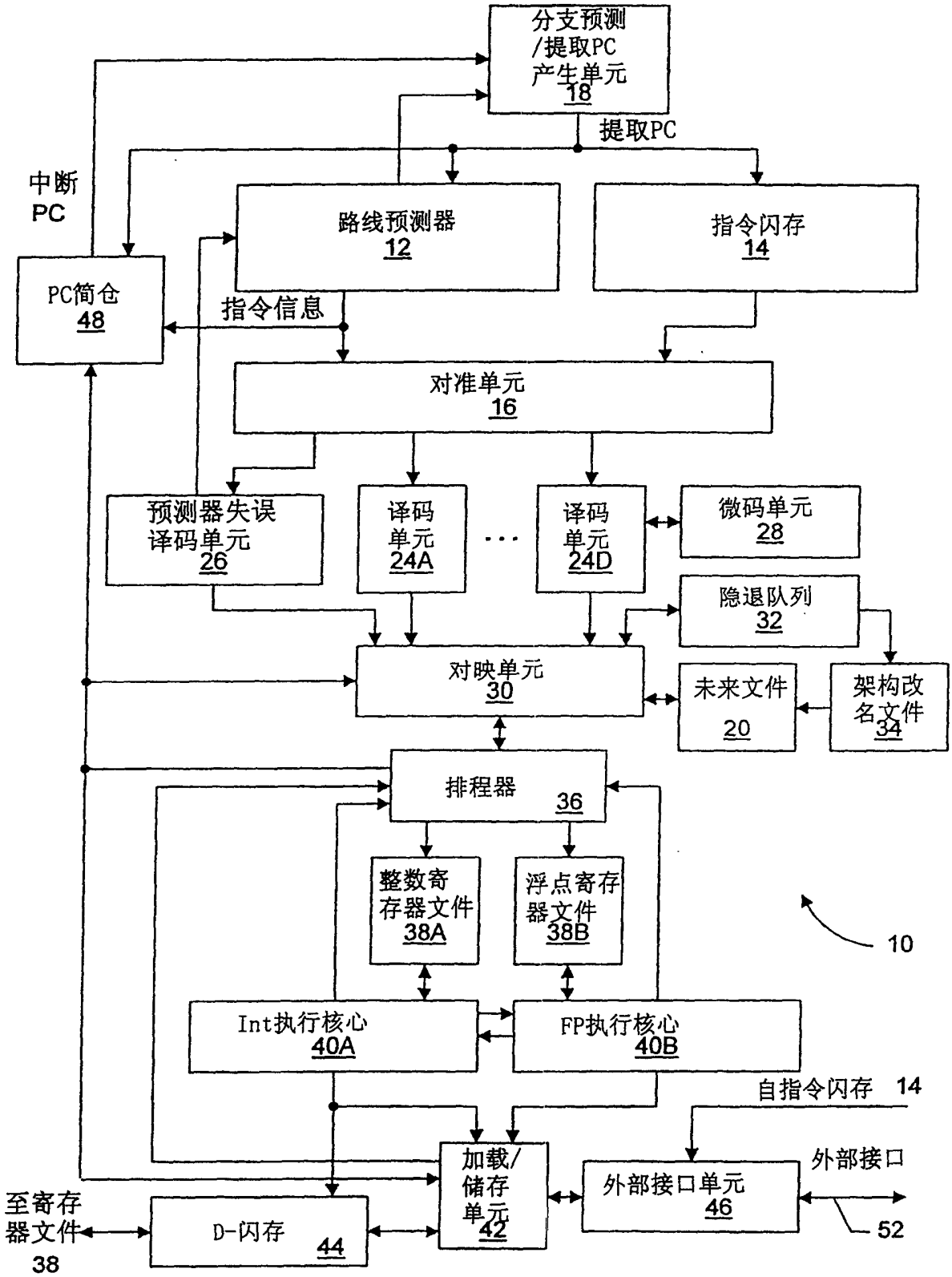

分支预测
/提取PC
产生单元
18
提取PC
中断
PC
路线预测器
12
指令闪存
14
PC简仓
48
指令信息
对准单元
16
预测器失误
译码单元
26
译码
单元
24A
译码
单元
24D
微码单元
28
隐退队列
32
对映单元
30
未来文件
20
架构改
名文件
34
排程器
36
整数寄
存器文件
38A
浮点寄存
器文件
38B
10
Int执行核心
40A
FP执行核心
40B
自指令闪存 14
加载/
储存
单元
42
外部接口单元
46
外部接口
52
至寄存
器文件
38
D-闪存
44

图1

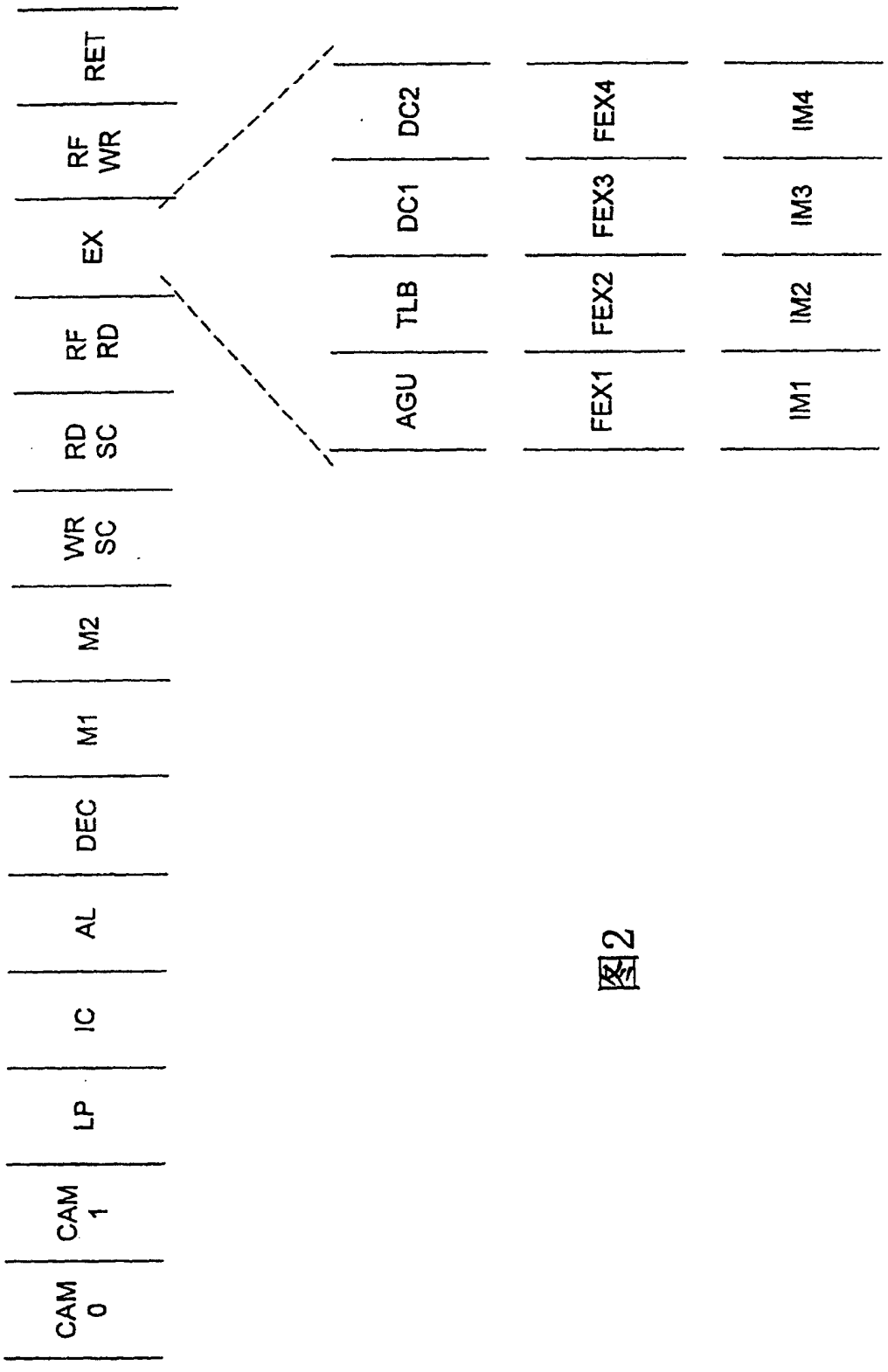

CAM 0
CAM 1
LP
IC
AL
DEC
M1
M2
WR SC
RD SC
RF RD
EX
RF WR
RET
AGU
TLB
DC1
DC2
FEX1
FEX2
FEX3
FEX4
IM1
IM2
IM3
IM4

图2

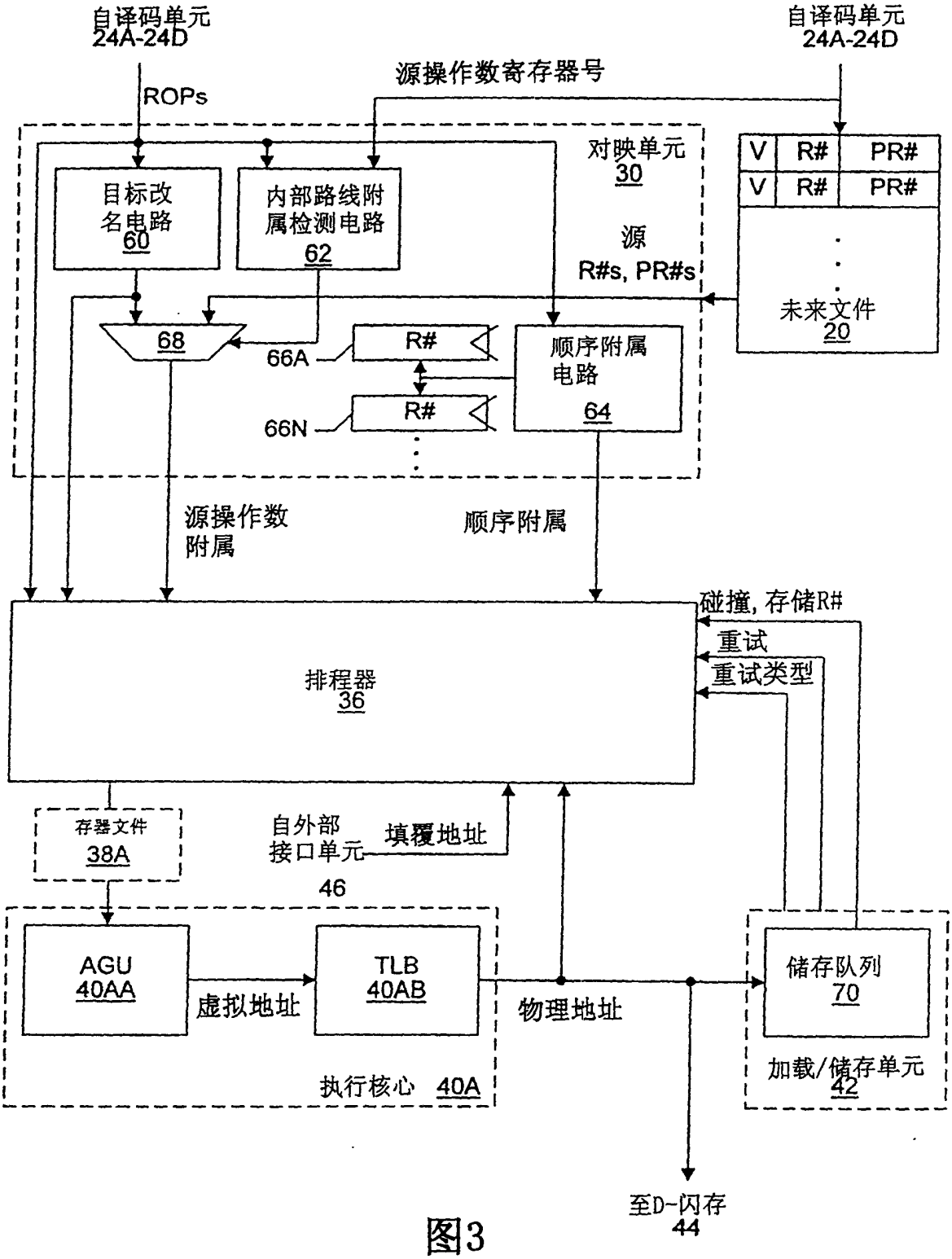

自译码单元
24A-24D
ROPs
源操作数寄存器号
自译码单元
24A-24D
对映单元
30
目标改
名电路
60
内部路线附
属检测电路
62
源
R#s, PR#s
V
R#
PR#
V
R#
PR#
未来文件
20
68
66A
R#
顺序附属
电路
64
66N
R#
源操作数
附属
顺序附属
碰撞, 存储R#
重试
重试类型
排程器
36
存器文件
38A
自外部
接口单元
填覆地址
46
AGU
40AA
虚拟地址
TLB
40AB
物理地址
储存队列
70
加载/储存单元
42
执行核心 40A
至D-闪存
44

图3

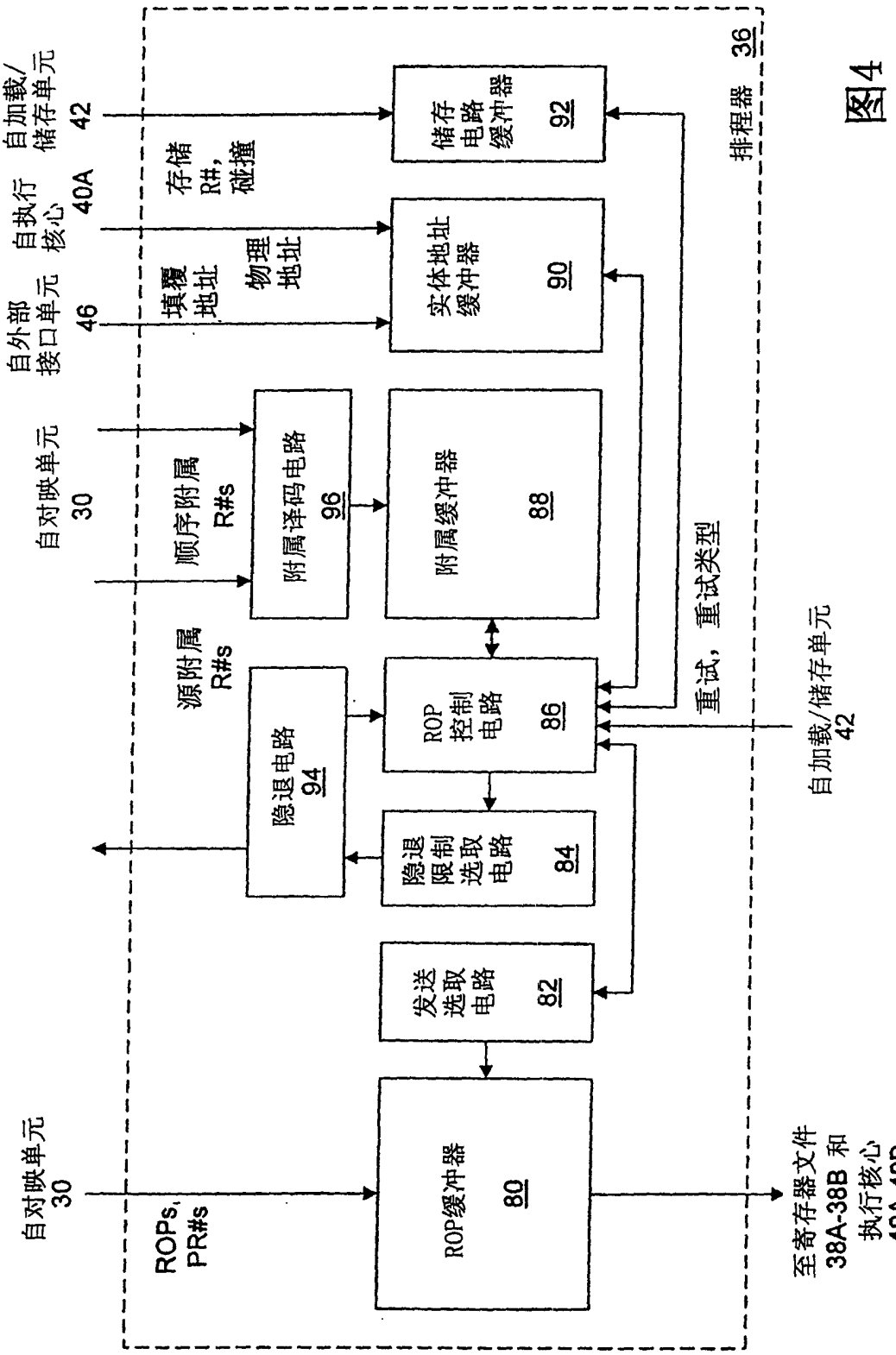

自对映单元
30
ROPs,
PR#s
ROP缓冲器
80
发送
选取
电路
82
隐退
限制
选取
电路
84
隐退电路
94
源附属
R#s
ROP
控制
电路
86
顺序附属
R#s
附属译码电路
96
附属缓冲器
88
自对映单元
30
自外部
接口单元
46
填覆
地址
物理
地址
实体地址
缓冲器
90
自执行
核心
40A
存储
R#,
碰撞
储存
电路
缓冲器
92
自加载/
储存单元
42
排程器
36
重试，重试类型
自加载/储存单元
42
至寄存器文件
38A-38B 和
执行核心
40A-40B

图4

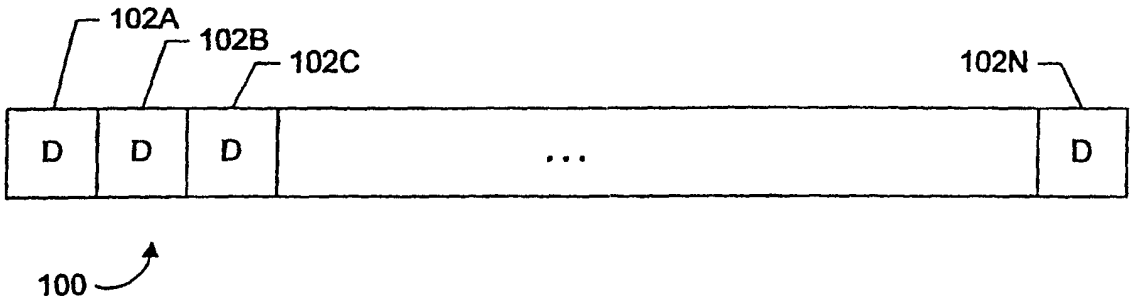

102A
102B
102C
102N
D
D
D
...
D
100

图5

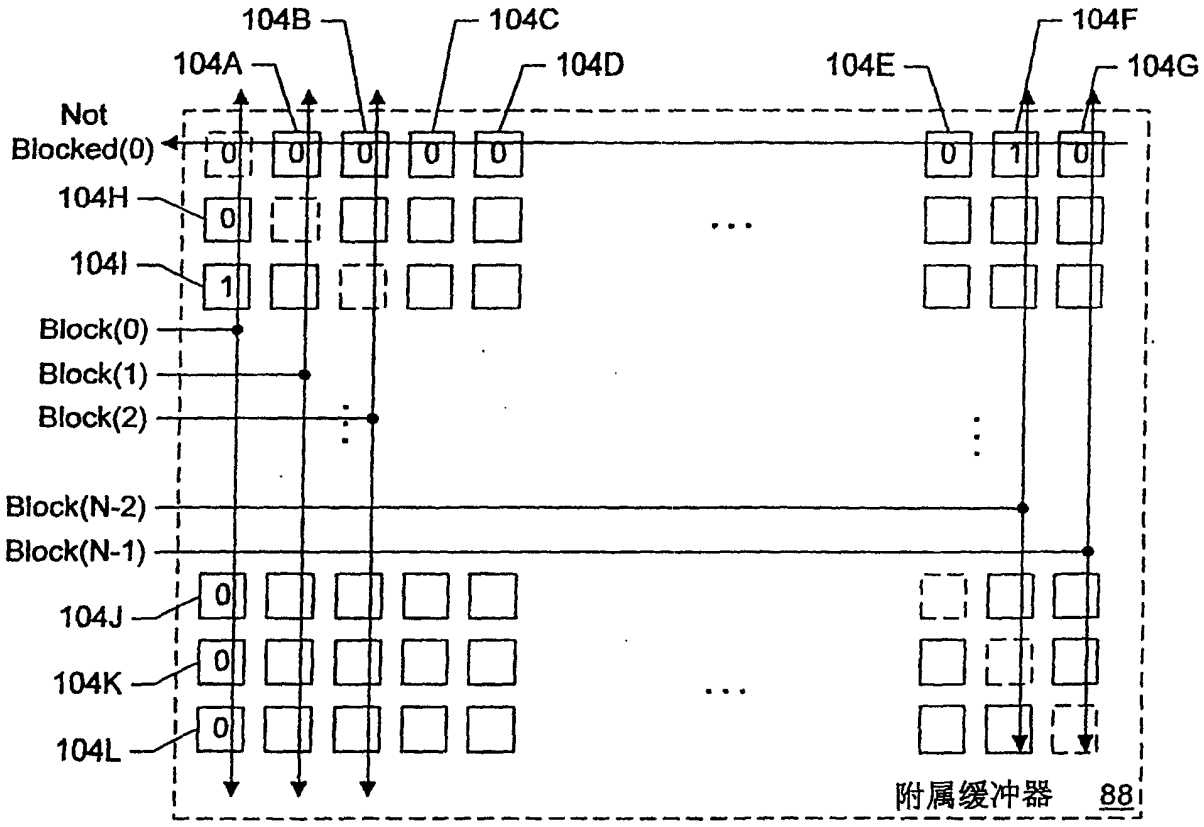

104B
104C
104F
104A
104D
104E
104G
Not
Blocked(0)
104H
104I
Block(0)
Block(1)
Block(2)
Block(N-2)
Block(N-1)
104J
104K
104L
附属缓冲器 88

图6

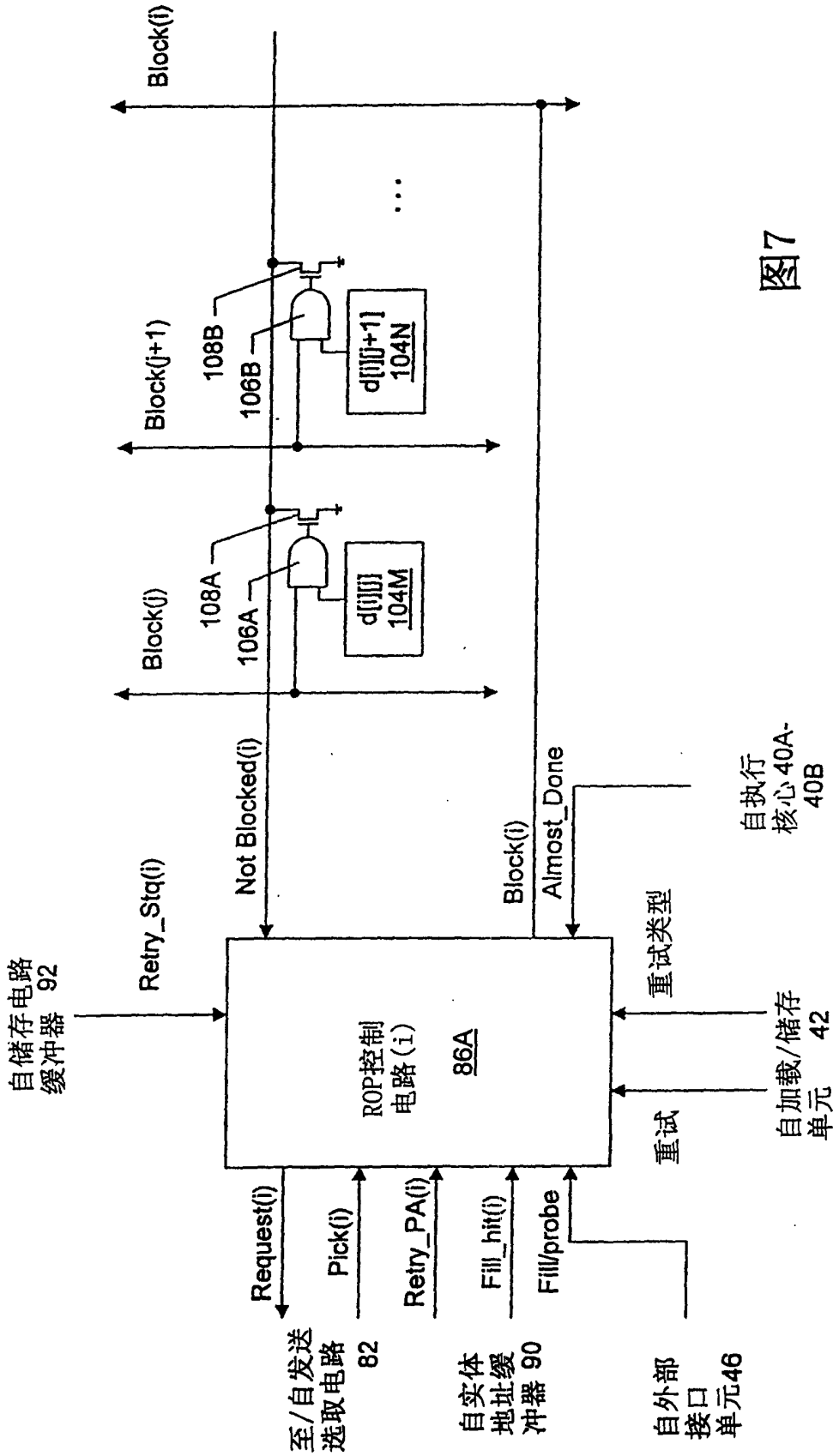

Block(i)
Block(j+1)
108B
106B
d[[j]j+1]
104N
Block(j)
108A
106A
d[[j]]j]
104M
Not Blocked(i)
Block(i)
Almost_Done
Retry_Stq(i)
自储存电路
缓冲器 92
ROP控制
电路(i)
86A
重试类型
自执行
核心40A-
40B
自加载/储存
单元 42
重试
Request(i)
Pick(i)
Retry_PA(i)
Fill_hit(i)
Fill/probe
至/自发送
选取电路
82
自实体
地址缓
冲器 90
自外部
接口
单元46

图7

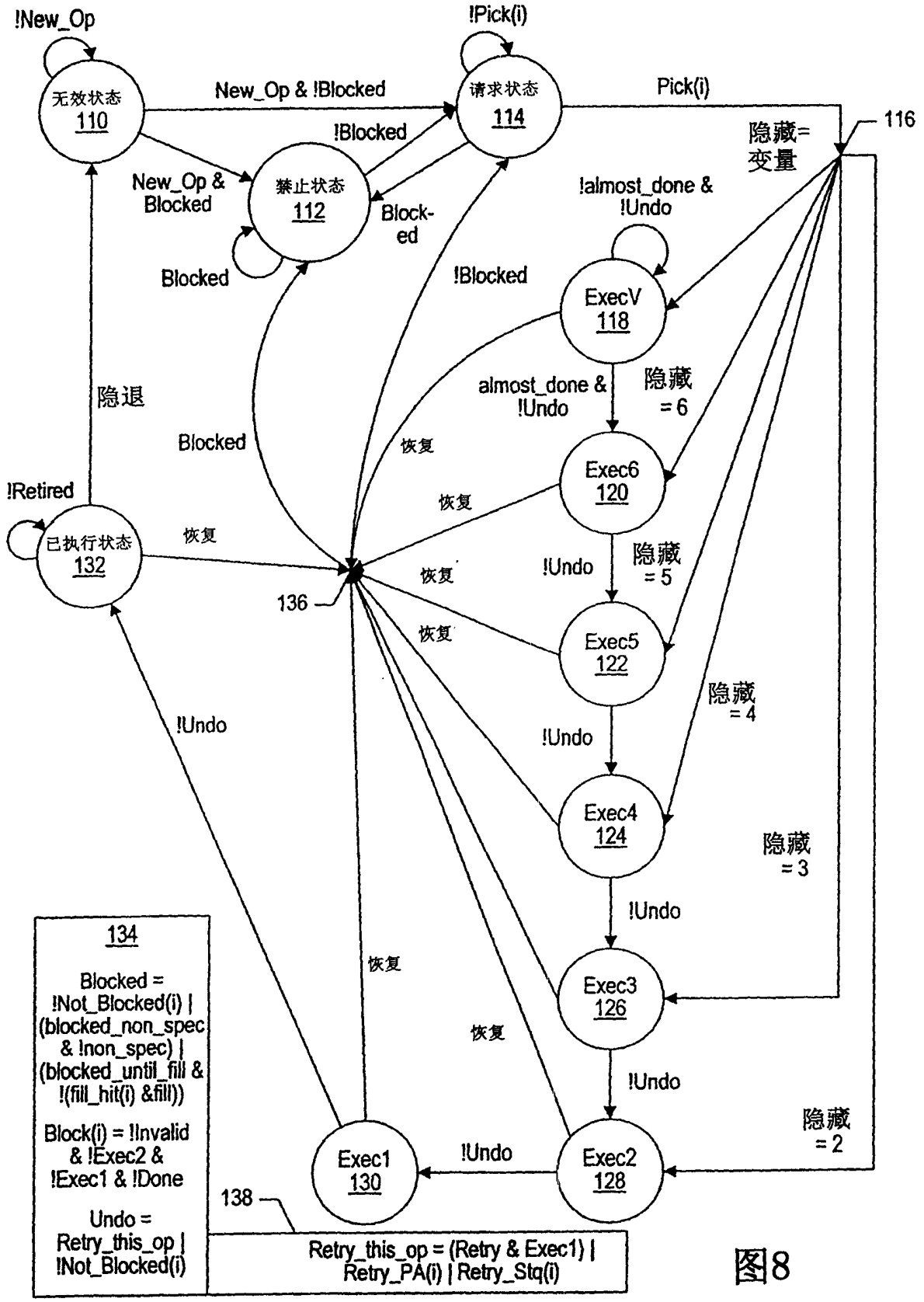

!New_Op
无效状态
110
New_Op & !Blocked
!Pick(i)
请求状态
114
Pick(i)
116
隐藏=
变量
!Blocked
New_Op &
Blocked
禁止状态
112
Block-
ed
Blocked
!Blocked
!almost_done &
!Undo
ExecV
118
隐退
almost_done &
!Undo
隐藏
= 6
Blocked
恢复
Exec6
120
!Retired
恢复
恢复
已执行状态
132
恢复
!Undo
隐藏
= 5
136
恢复
Exec5
122
恢复
!Undo
隐藏
= 4
!Undo
Exec4
124
隐藏
= 3
!Undo
134
Blocked =
!Not_Blocked(i) |
(blocked_non_spec
& !non_spec) |
(blocked_until_fill &
!(fill_hit(i) &fill))
Block(i) = !Invalid
& !Exec2 &
!Exec1 & !Done
Undo =
Retry_this_op |
!Not_Blocked(i)
恢复
Exec3
126
恢复
!Undo
隐藏
= 2
Exec1
130
!Undo
Exec2
128
138
Retry_this_op = (Retry & Exec1) |
Retry_PA(i) | Retry_Stq(i)

图8

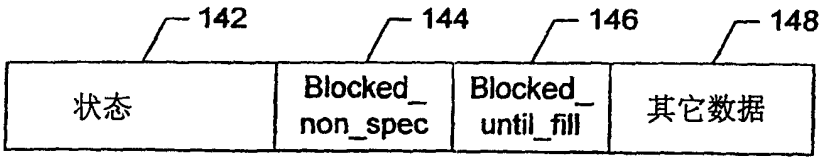
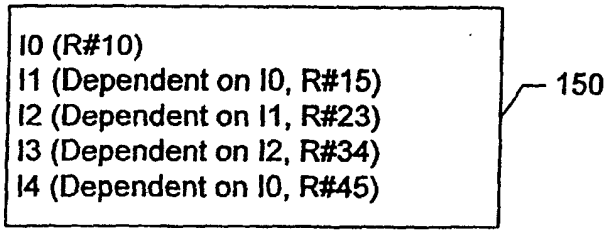

142
144
146
148
状态
Blocked_ non_spec
Blocked_ until_fill
其它数据
140

图9


I0 (R#10)
I1 (Dependent on I0, R#15)
I2 (Dependent on I1, R#23)
I3 (Dependent on I2, R#34)
I4 (Dependent on I0, R#45)
150

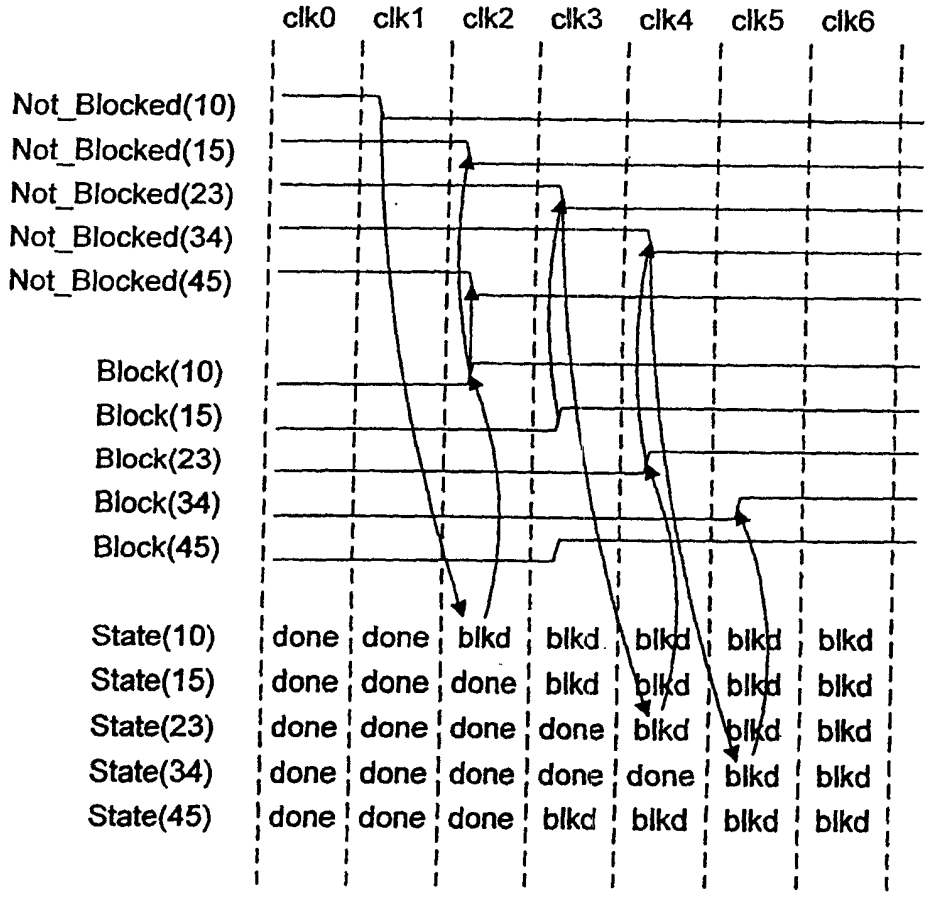

clk0 clk1 clk2 clk3 clk4 clk5 clk6
Not_Blocked(10)
Not_Blocked(15)
Not_Blocked(23)
Not_Blocked(34)
Not_Blocked(45)
Block(10)
Block(15)
Block(23)
Block(34)
Block(45)
State(10) done done blkd blkd blkd blkd blkd
State(15) done done done blkd blkd blkd blkd
State(23) done done done done blkd blkd blkd
State(34) done done done done done blkd blkd
State(45) done done done blkd blkd blkd blkd

图10

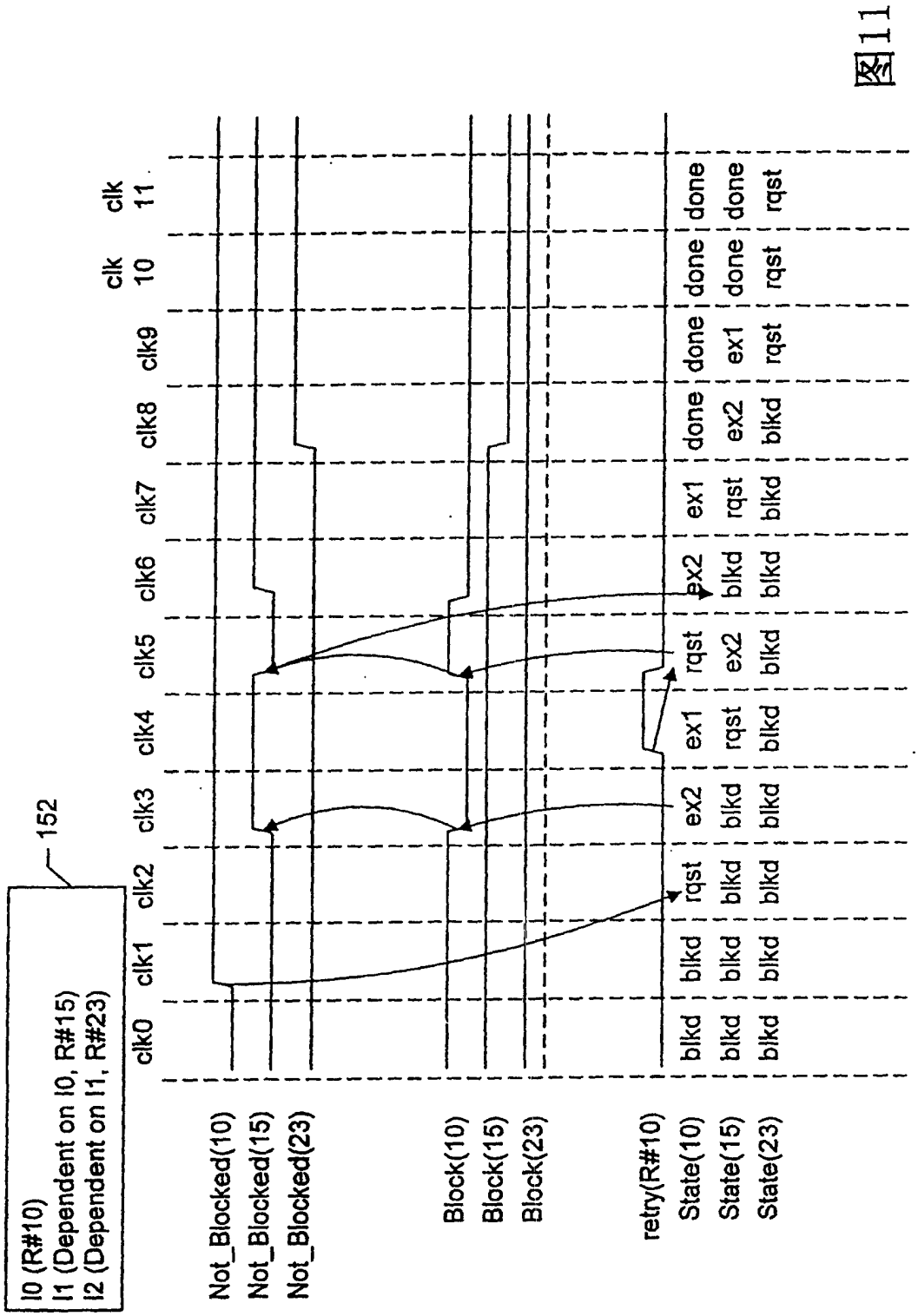

I0 (R#10)
I1 (Dependent on I0, R#15)
I2 (Dependent on I1, R#23)
152
clk0 clk1 clk2 clk3 clk4 clk5 clk6 clk7 clk8 clk9 clk 10 clk 11
Not_Blocked(10)
Not_Blocked(15)
Not_Blocked(23)
Block(10)
Block(15)
Block(23)
retry(R#10)
State(10)
State(15)
State(23)
blkd rqst ex2 ex1 rqst ex2 ex1 done done done done
blkd blkd blkd blkd blkd blkd blkd blkd ex2 ex1 done done
blkd blkd blkd blkd blkd blkd blkd blkd blkd rqst rqst rqst

图11

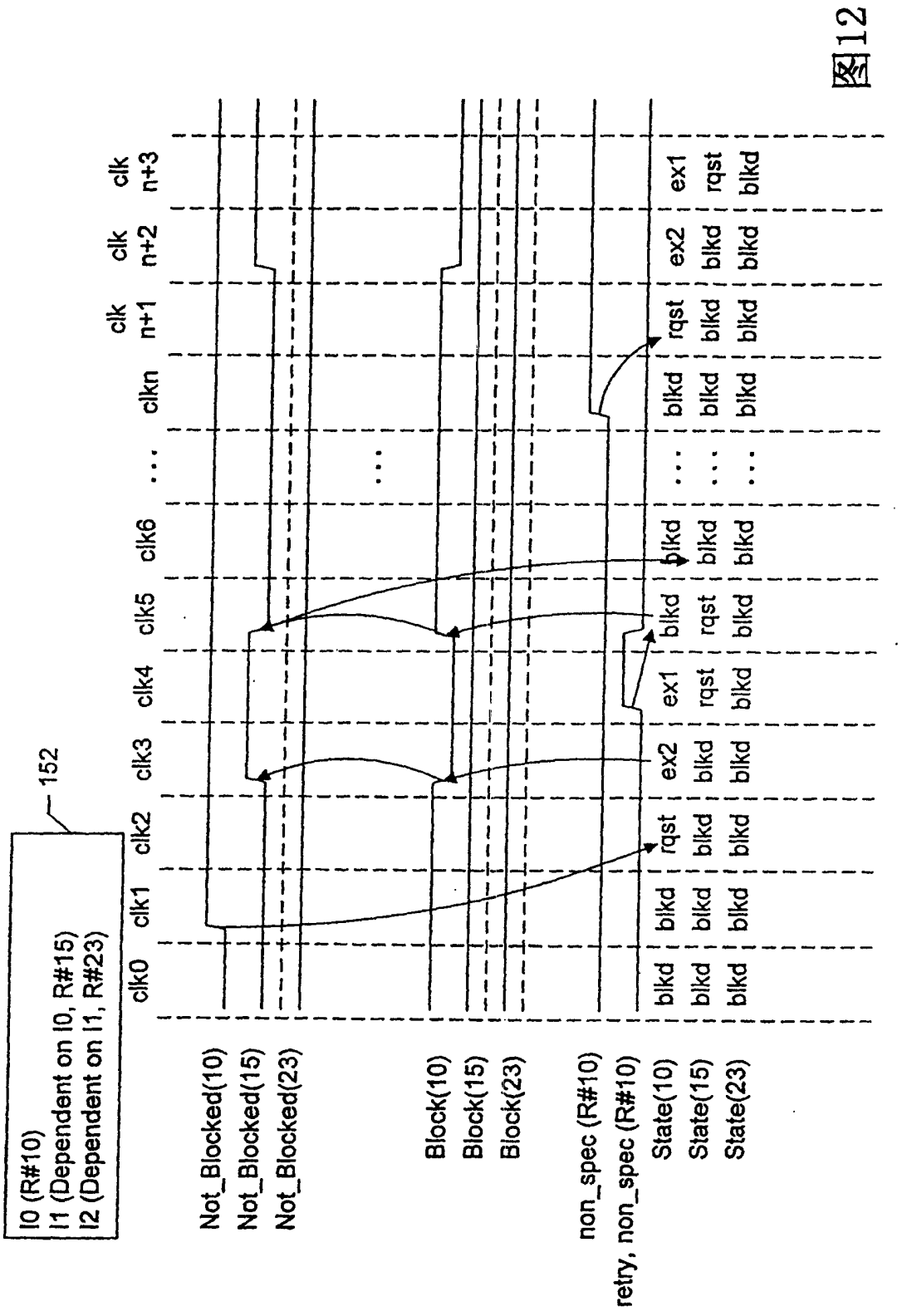

I0 (R#10)
I1 (Dependent on I0, R#15)
I2 (Dependent on I1, R#23)
152
clk0
clk1
clk2
clk3
clk4
clk5
clk6
...
clkn
clk n+1
clk n+2
clk n+3
Not_Blocked(10)
Not_Blocked(15)
Not_Blocked(23)
Block(10)
Block(15)
Block(23)
non_spec (R#10)
retry, non_spec (R#10)
State(10)
State(15)
State(23)
blkd
rqst
ex1
ex2

图12

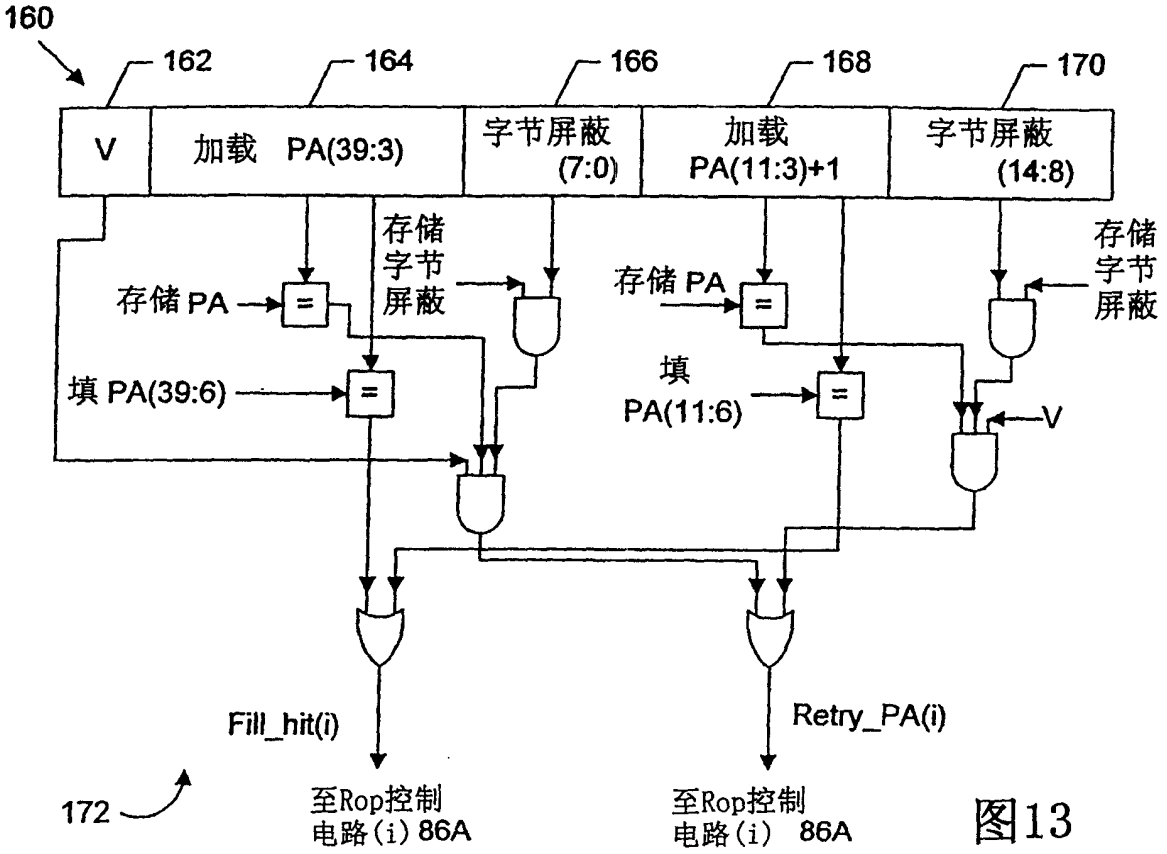

160
162
164
166
168
170
V
加载 PA(39:3)
字节屏蔽 (7:0)
加载 PA(11:3)+1
字节屏蔽 (14:8)
存储 字节 屏蔽
存储 PA
=
填 PA(39:6)
填 PA(11:6)
V
Fill_hit(i)
Retry_PA(i)
172
至Rop控制 电路(i) 86A
至Rop控制 电路(i) 86A

图13

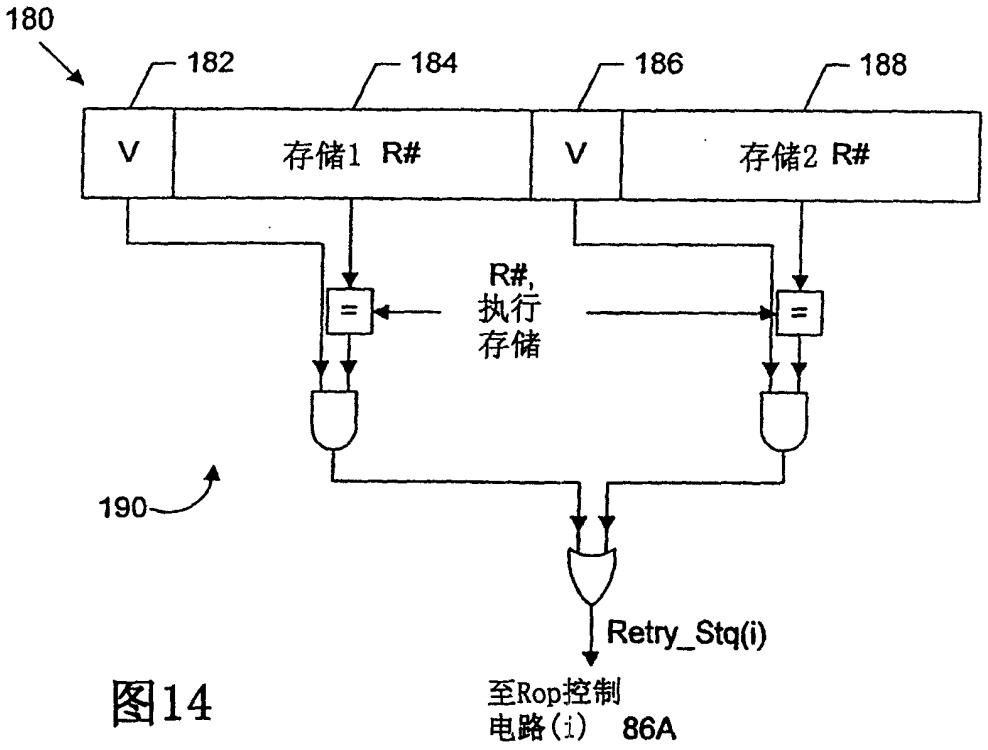

180
182
184
186
188
V
存储1 R#
V
存储2 R#
R#, 执行 存储
=
190
Retry_Stq(i)
至Rop控制 电路(i) 86A

图14

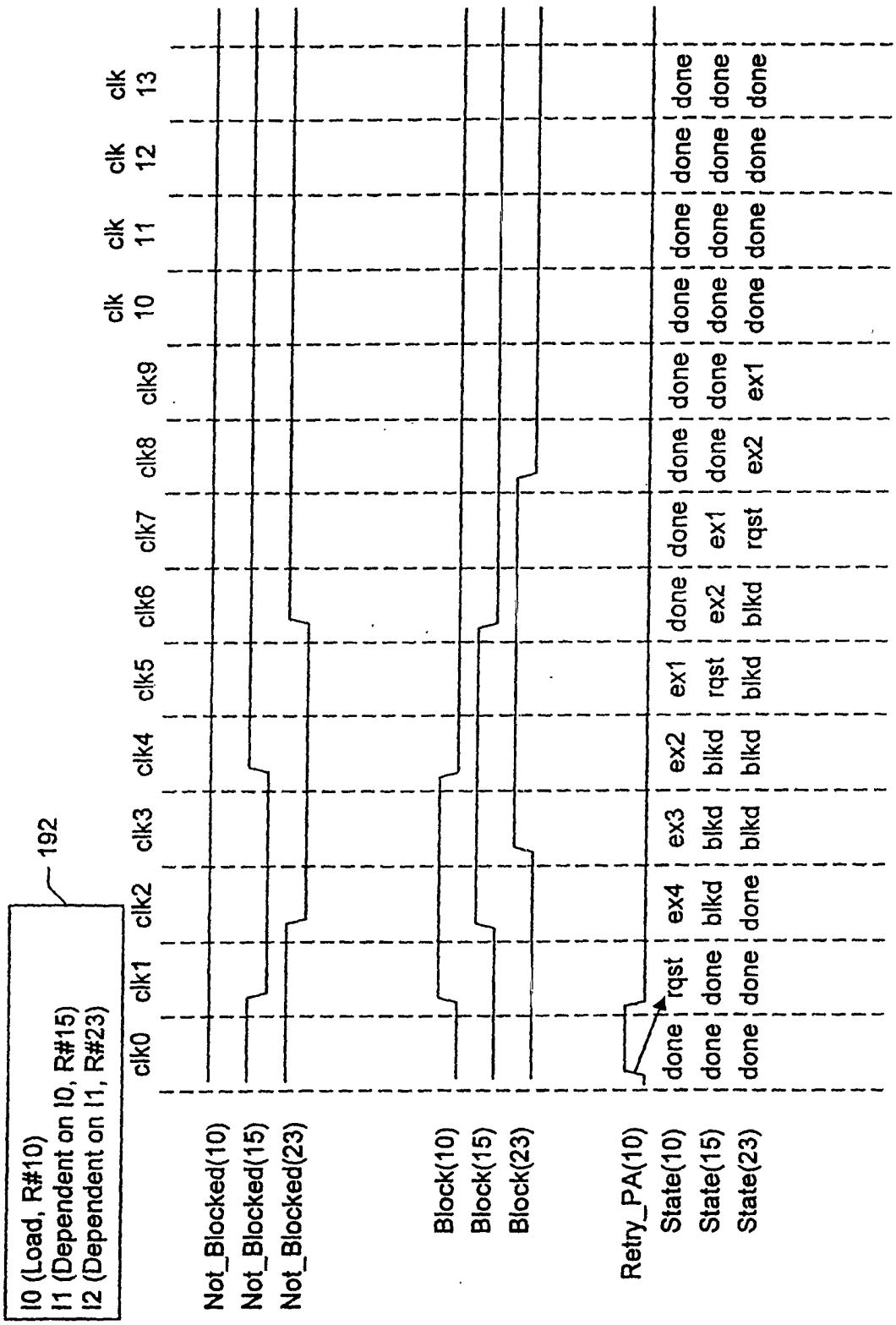

I0 (Load, R#10)
I1 (Dependent on I0, R#15)
I2 (Dependent on I1, R#23)
192
clk0 clk1 clk2 clk3 clk4 clk5 clk6 clk7 clk8 clk9 clk 10 clk 11 clk 12 clk 13
Not_Blocked(10)
Not_Blocked(15)
Not_Blocked(23)
Block(10)
Block(15)
Block(23)
Retry_PA(10)
State(10)
State(15)
State(23)
done rqst ex4 ex3 ex2 ex1 done done done done done done
done done blkd blkd blkd rqst ex2 ex1 done done done done
done done done blkd blkd blkd blkd rqst ex2 ex1 done done

图15

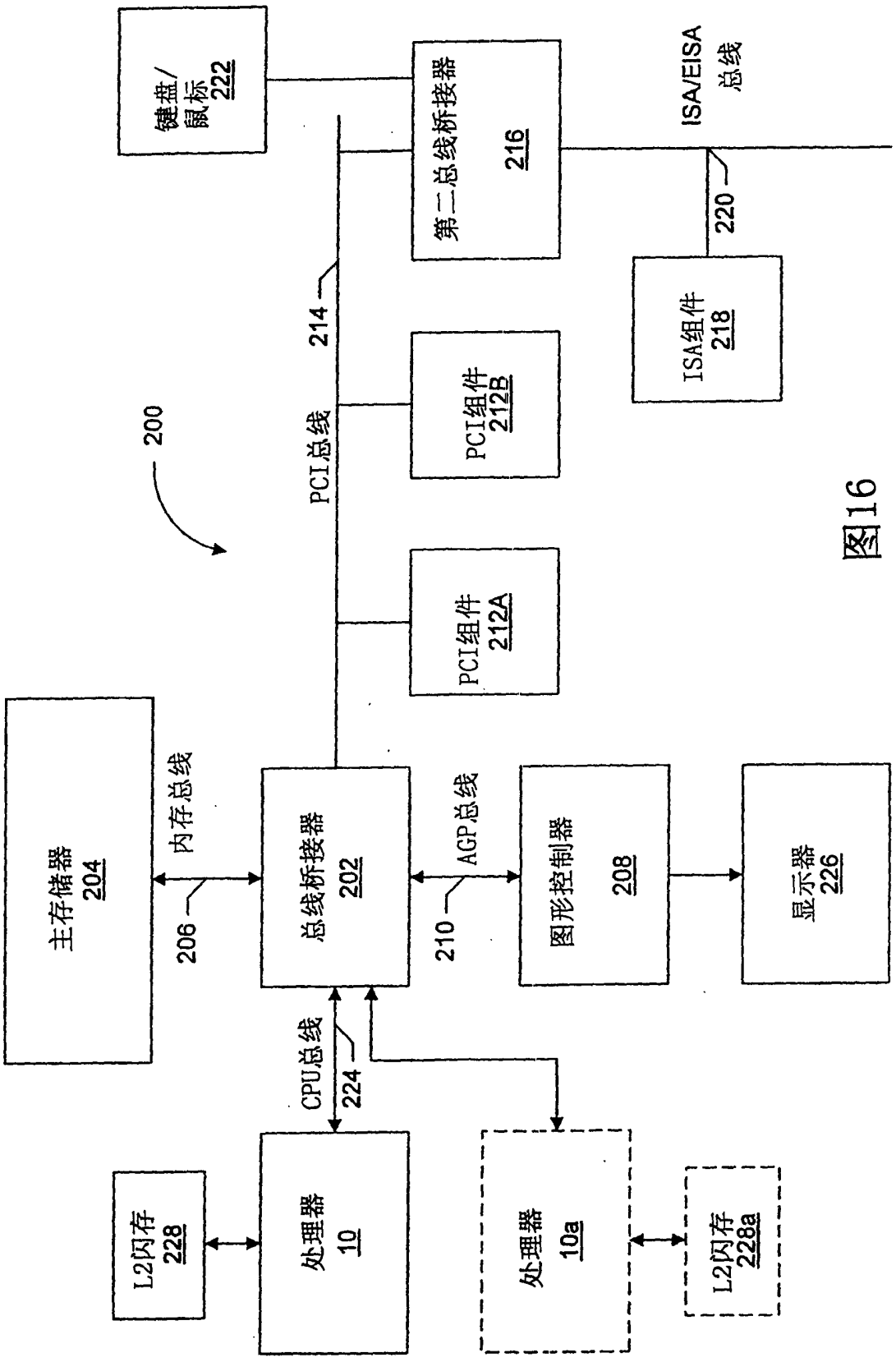

键盘/鼠标 222
第二总线桥接器 216
ISA/EISA 总线
220
ISA组件 218
214
PCI总线
PCI组件 212B
PCI组件 212A
200
主存储器 204
内存总线
206
总线桥接器 202
AGP总线
210
图形控制器 208
显示器 226
CPU总线
224
L2闪存 228
处理器 10
处理器 10a
L2闪存 228a

图16

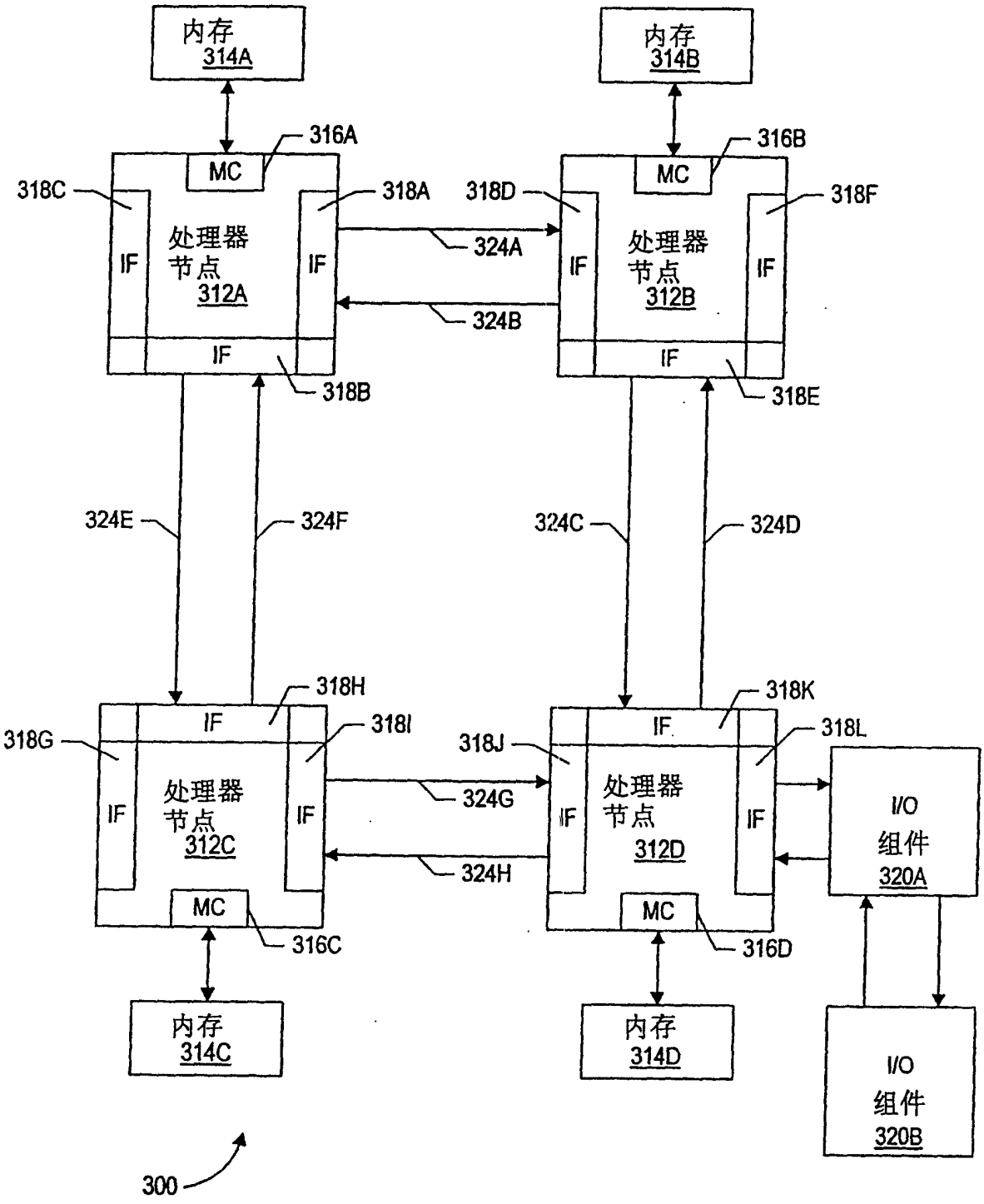

内存
314A
内存
314B
316A
316B
MC
MC
318C
318A
318D
318F
处理器
节点
312A
处理器
节点
312B
IF
IF
IF
IF
IF
IF
324A
324B
318B
318E
324E
324F
324C
324D
318H
318K
318G
318I
318J
318L
处理器
节点
312C
处理器
节点
312D
324G
324H
I/O
组件
320A
MC
MC
316C
316D
内存
314C
内存
314D
I/O
组件
320B
300

图17