[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 99815322.2

[45] 授权公告日 2004 年 9 月 8 日

[11] 授权公告号 CN 1165840C

[22] 申请日 1999.12.16 [21] 申请号 99815322.2
[30] 优先权
[32] 1998.12.30 [33] US [31] 09/222805
[86] 国际申请 PCT/US1999/029805 1999.12.16
[87] 国际公布 WO2000/041070 英 2000.7.13
[85] 进入国家阶段日期 2001.7.2
[71] 专利权人 英特尔公司
地址 美国加利福尼亚州
[72] 发明人 A·A·麦钱特 D·J·萨格
审查员 袁丽颖

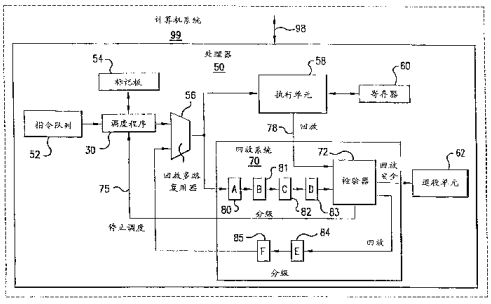
[74] 专利代理机构 中国专利代理(香港)有限公司
代理人 郑立柱 王忠忠

权利要求书 3 页 说明书 8 页 附图 9 页

[54] 发明名称 计算机处理器检验指令的方法和系统及相应计算机系统

[57] 摘要

一种计算机处理器具有接收指令的检验器。该检验器包括包括标记板、用于接收一个外部回放信号的输入端以及与标记板和输入端连接的判定逻辑。判定逻辑以该标记板和该外部回放信号为基础确定指令是否执行正确。



ISSN1008-4274

1.一种计算机处理器，包括：

用于接收指令的检验器，所述的检验器包括：

标记板；

用于接收外部回放信号的输入端；以及

与所述标记板和输入端连接的判定逻辑；

其中所述的判定逻辑在所述标记板和在所述输入端接收的外部回放信号的基础上确定由所述检验器接收的指令是否被正确执行。

2.如权利要求1所述的处理器，其中指令包括至少一个源寄存器和一个目的寄存器,并且所述的标记板包括该源寄存器和该目的寄存器的状态，所述的判定逻辑从所述的标记板中读取源寄存器的状态从而确定执行的指令是否正确。

3.如权利要求1所述的处理器，其中如果指令被确定为被正确执行，则所述的判定逻辑回放该指令。

4.如权利要求1所述的处理器，其中如果指令被确定为被正确执行，所述的判定逻辑调度该指令退役。

5.如权利要求2所述的处理器，其中如果指令被确定为被正确执行，所述的判定逻辑更新目的寄存器的状态。

6.如权利要求1所述的处理器，所述的判定逻辑包括：

一个检验器矩阵引擎。

7.如权利要求6所述的处理器，所述的检验器矩阵引擎包括：

具有多个入口的保持缓冲器用来保持对应的指令；以及

相关性矩阵，具有多个行，每一行对应于所述保持缓冲器的一个入口，并且具有多个列，每一列对应于所述保持缓冲器的一个入口的相关性。

8.如权利要求7所述的处理器，其中所述的判定逻辑在相关性矩阵行内与指令对应的相关性指令的基础上确定执行的指令是否正确。

9.如权利要求8所述的处理器，其中如果指令被确定已被正确执行，所述的一个列被清除。

10.如权利要求8所述的处理器，其中所述的相关性指示包括至少一个设置在相关性矩阵行内与指令对应的位。

11．一种检验具有至少一个源的计算机指令的方法，所述的方法包括：

（a） 通过读出一个标记板的状态确定源是否准备就绪；

（b） 确定外部回放条件是否无效；

（c） 如果源没有准备就绪或外部回放条件有效，回放指令；以及

（d） 如果源准备就绪并且外部回放条件无效，调度该指令退役。

12．如权利要求11所述的方法，其中步骤（b）包括接收一个外部回放信号。

13．如权利要求11所述的方法，其中步骤（a）包括以下步骤：

如果该指令非常近似于一个另外的指令，检验相关性矩阵行中与指令对应的相关性指示。

14．如权利要求13所述的方法，进一步包括以下步骤：

如果源准备就绪并且外部回放条件无效，清除该相关性矩阵的行。

15．一种检验具有至少一个源的计算机指令的系统，所述的系统包括：

（a）确定源是否准备就绪的标记板；

（b）确定外部回放条件是否无效的检验器；

（c）如果源没有准备就绪或外部回放条件有效，回放指令的执行单元；以及

（e）如果源准备就绪并且外部回放条件无效，所述检验器将指令调度到退役过程。

16．一种计算机系统，包括

总线；

与所述总线连接的存储装置；

用于执行计算机指令的处理器，所述的处理器包括：

标记板；

用于接收外部回放信号的输入端；以及

与所述的标记板和所述的输入端连接的判定逻辑；

其中所述的判定逻辑在所述标记板和在所述输入端接收的外部回放信号的基础上确定由所述检验器接收的指令是否被正确执行。

17．如权利要求16所述的计算机系统，其中指令包括至少一个源寄存器和一个目的寄存器，并且所述的标记板包括该源寄存器和目的寄存器的状态，所述的判定逻辑从标记板读取源寄存器的状态从而确定指令执行是否正确。

18．如权利要求16所述的计算机系统，其中如果指令被确定为没有正确执行，所述的判定逻辑回放指令。

19．如权利要求16所述的计算机系统，其中如果指令被确定为正确执行，所述判定逻辑调度指令到退役过程。

20．如权利要求17所述的计算机系统，其中如果指令被确定为正确执行，所述的判定逻辑更新目的寄存器的状态。

21．如权利要求16所述的计算机系统，所述的判定逻辑包括：

一个检验器矩阵引擎。

22．如权利要求21所述的计算机系统，所述的检验矩阵引擎包括：

具有多个入口的用于保持指令的保持缓冲器；以及

相关性矩阵，具有多个行，每行对应于保持缓冲器的一个入口，以及多个列，每列对应于所属保持缓冲器一个入口的相关性。

23．如权利要求22所述的计算机系统，其中所述的判定逻辑在相关性矩阵行内与指令对应的相关性指令的基础上确定指令执行是否正确。

24．如权利要求23所述的计算机系统，其中如果指令被确定为正确地执行，所述的一列被清除。

25．如权利要求23所述的计算机系统，其中所述的相关性指示包括至少一个设置在相关性矩阵行内与指令对应的位。

计算机处理器、检验指令的方法和系统及相应计算机系统

发明领域

本发明涉及计算机处理器，特别是涉及在计算机处理器中的检验指令的检验器。

发明技术背景

大多数计算机处理器的主要功能是执行计算机指令。大多数处理器按照所接收的编程顺序来执行指令。然而，近来一些处理器，例如英特尔公司的奔腾Ⅱ处理器是“无序”（out-of-order）处理器。

无序处理器可以按照任何顺序执行指令，因此对于每一个指令所要求的数据和执行单元是可利用的。计算机系统中的通过机器寄存器而依赖于另外的指令。无序处理器试图积极地寻找输入源通过计算的得到的指令来利用并发性，并且将其调度在编程在后的指令的前面。这样创造了能更有效地使用机器资源并且整体执行更快的机会。

无序处理器也可以通过减少整体延时来提高性能。假定处理器使用的存储器子系统是完备的，这样可以通过推测地调度指令来完成。因此，处理器可以假定所有的高速缓存的存取被命中。这样允许相关的算术和逻辑指令被调度，无需从存储器子系统接收它们被正确执行的确认的整个延时。

推测地预定指令的无序处理器要求一个机制，来重新执行那些没有被正确执行的指令。这类机制的一种是回放系统，该系统被美国专利申请披露，专利申请号为09/106,857，申请日为1998年7月30日。该回放系统必须包括一个检验装置用来确定指令的执行是否正确。

基于以上所述，需要一个用于预测调度指令的计算机处理器的回放系统检验装置。

发明概述

本发明的一个实施例是具有接收指令的检验器的计算机处理器。该检验器包括标记板,接收外部回放信号的输入端，以及与标记板和输入端相连的判定逻辑。判定逻辑在标记板和外部回放信号二者的基础上确定指令执行是否正确。

附图简述

附图1表示具有检验器的回放系统的处理器的方框图。

附图2表示按照本发明的一个实施例的具有标记板的检验器的详细方框图。

附图3表示每个接收指令的检验器的判定逻辑执行步骤的流程图。

附图4表示按照本发明的一个实施例的检验器的详细方框图。

附图5表示通过调度程序以连续的调度周期调度的指令的列表。

附图6A-6J表示在附图5所示的每一个调度周期内检验器矩阵引擎的状态。

详细描述

本发明的一个实施例是可预测调度指令的处理器，该处理器包括一个在回放系统中的检验器。回放系统回放那些被初始调度到一个执行单元时没有正确被执行的指令,同时保护指令的初始调度顺序。检验器确定指令是否正确执行。

附图1表示按照本发明的一个实施例的具有检验器的回放系统的计算机处理器的方框图。处理器50包括在计算机系统99中。处理器50与计算机系统的其它部件相连接，例如通过系统总线98连接的存储器装置（图中未示出）。处理器50是一个无序处理器。

处理器50包括指令队列52。指令对列52向调度程序30提供指令。在这一实施例中，指令为“微操作”。微操作是通过将复杂的指令转换为便于执行简单的、固定长度的指令而产生的。本发明的实施例中的每一指令包括两个逻辑源和一个逻辑目的。源和目的是处理器50中的寄存器。

当资源可用于执行指令并且指令所需要的输入源准备就绪时，调度程序30调度从指令队列52收到一个指令。调度程序30连接到标记板54。标记板54追踪源的准备状态。当一个指令被执行并且其结果（目的）寄存器获得正确的数据时，调度程序30更新标记板54中目的从而作好准备。

具有较佳结构设计的现有的无序处理器在数据实际可用之前更新标记板，而处理器的流水线特性完全是已知的。这就允许处理器利用从调度到实际执行之间的延时。然而，在这些处理器中调度程序一直等待指令正确执行的确认。

相反，处理器50的结构更加先进并且在确认指令正确执行之前更新标记板。这就允许处理器50利用更多的并发性，比传统无序处理器设计更进一步地减少延时，但是需要一种如回放系统70的机制，来重新执行那些由于较高的预测调度没有被正确调度的指令。

调度程序30将指令输出到回放多路复用器56中。多路复用器56的输出连接到执行单元58上。执行单元58执行所接收的指令。执行单元58可以是算术逻辑单元（ALU）,浮点ALU,存储器单元等等。执行单元58与处理器50的寄存器60连接。当执行指令时，执行单元58将数据装入和存储在寄存器60中。

回放系统70

处理器50进一步包括一个回放系统70。回放系统70回放那些被调度程序30调度后没有被正确执行的指令。如执行单元58一样，回放系统70接收从回放多路复用器56输出的指令。回放系统70包括两个分级组。其中的一个分级组包括多个级80-83。另一个分级组包括级84和85。因此，指令通过回放系统70被分级与通过执行单元58形成的级并行。级80-85的数量依据在每个执行通路所期望的级数而变化。

回放系统70进一步包括检验器72。一般来说，按照本发明的检验器72接收指令并且分析哪些指令通过了一系列的标准而哪些没有。在附图1所示的实施例中，检验器72是回放系统70的一部分，检验器72从级83接收指令，并且确定哪些指令正确执行而哪些没有。如果指令执行正确，检验器72宣布指令“回放安全”并且将指令发送至退役单元62（retirement unit），在该退役单元中指令被按照编程顺序退出。退役指令有利于处理器50，因为其释放处理器资源并且允许另外的指令开始执行。如果指令没有正确执行，检验器72通过级84和85将指令发送到回放多路复用器56来回放或重新执行指令。

在将回放指令发送到回放多路复用器56的同时，检验器72发出一个“停调度程序”的信号75给调度程序30。在回放指令到达回放多路复用器56之前的至少一个时钟周期，发送停调度程序信号75。在这一实施例中，停调度程序信号75告知调度程序30在下一个时钟周期不要调度指令。这样就产生一个开放的时隙（open slot）用于从多路回放复用器56输出的回放指令，并且避免在同一钟周期内两个指令被输入到回放多路复用器56中。

检验器72

检验器72的主要功能为分析指令流从而确定哪个指令被正确执行，哪个指令未被正确执行。一个指令执行的不正确可能有许多原因。最主要的原因是源相关性或外部回放条件。源相关性可能出现在一个指令源依赖于另一个指令的结果的情况。外部回放条件的例子包括高速缓存的故障,不正确的数据发送（从

存储缓冲器回到一载入缓冲器），内藏式内存相关性，回写冲突，未知的数据/地址和串行指令。

检验器72利用两组判别来确定指令是否正确执行。第一组是由一个外部代理如存储器子系统或一执行引擎产生的外部回放条件，用于告知检验器72指令未被正确执行。

第二组为指令开始执行时输入源不正确。这一情况发生在由于比较高的预测性和处理器50的高级流水线的特性而使不正确的数据被传送时。当输入源不正确，到指令已被确定为被正确执行时已经有许多相关的指令已经被调度。错误的数据通过寄存器相关性从一个指令的结果传送到另一个指令。数据错误的传送过程与不断扩展的树相似并且严重损害处理器50的性能。

对于第一判别，外部回放条件通过回放信号78由检验器72接收。对于第二判别，在这一实施例中当输入源在指令开始执行时不正确时，检验器72利用一个标记板来确定。

附图2为检验器72的详细方框图，描述了在本发明的一个实施例如何使用标记板。检验器72包括标记板104和判定逻辑101。判定逻辑101在线107上接收指令和外部回放信号78。判定逻辑101在线108上进一步接收来自于标记板104的输入，如果所有输入源在执行时间内都是正确的，用该输入告知判定逻辑101。如果外部回放信号78已经被证实并且源输入就绪（或正确），判定逻辑101确定执行的指令正确，并且从线109输出指令到退役单元62。否则通过线111输出指令到回放多路复用器56中而使指令回放。当指令被确定为正确执行时，在寄存器准备就绪线110上标记板104在适当的时间被设置，表示目的就绪/可用/正确。适当的时间的过程等于指令的延时。

在这一实施例中,标记板104为10位宽并且用来记录哪个寄存器准备就绪。每一位代表一个寄存器,例如，“0”代表没有就绪，而“1”代表寄存器就绪。判定逻辑101可以要求读取每一位，并且通过线106确定指令源的准备状态。源的读出结果(也就是标记板104的每一位的状态)通过线108返回到判定逻辑101。判定逻辑101可以通过线110更新标记板104的寄存器状态。

当寄存器被用来重新使用时（也就是当指令被重新分配时），标记板104通过103被更新，表示目的不可用/没有准备就绪。除非指令执行正确，否则目的不可用。这就是检验器72清除标记板104的位的过程。

附图3表示检验器72的判定逻辑101接收每一次指令的执行步骤的流程图。在步骤120,判定逻辑101确定指令的两个源是否准备就绪。正如所讨论的，判定逻辑101时通过在线108上从标记板104接收各个寄存器的状态来确定这些的。

在步骤120如果两个源没有准备就绪，在步骤124指令被回放，并且发送至回放多路复用器56中。如果在步骤120中两个源准备就绪，在步骤122检验器72确定外部回放信号78是无效的，那么表示不要求回放，因为指令执行正确。

如果在步骤122中，回放信号78是无效的，在步骤124指令被回放，并且发送到回放多路复用器56中。如果在步骤122回放信号78是无效的，在步骤126指令被安全回放并且发送到退役单元62。

如果指令被安全回放，在步骤128中判定逻辑101对标记板104进行写操作表示指令的目的寄存器“准备就绪”。换句话说，判定逻辑101在标记板104代表指令的目的寄存器的位中写入“1”。

在执行附图3的步骤中会产生一个问题，这就是随着新的处理器操作频率越来越快，处理器时钟周期越来越短。这将导致从标记板104的同样位置读和写在非常接近的时间内。例如，如果处理器50的每一个时钟周期是1.0ns，设想这样一种情况，从标记板读和写各需0.5ns。实现这一情况非常困难，但是设想通过构造一个极快的电路来完成。

然而问题依然存在。设想两个指令，指令1和指令2。假设指令2使用指令1的结果。假设在紧接的周期调度指令1和指令2。对于指令1如果判断逻辑101在时间t=0.0,始读标记板104, 在t=0.5结束读取标记板104的过程。接着，判定逻辑101必须花费时间来确定指令1执行是否正确。假设这一过程所用时间为0.25ns。加上写过程所用时间0.5ns。在写完成时，所经过的时间为t=0.5+0.25+0.5=1.25ns。

由于，指令2在指令1之后的一个周期内被调度。因此，指令2必须在t=1.0开始读。这就产生了一个诱发性问题：前一操作的写入还未完成，后一指令的读取已经开始。加上电干扰和与长距离传输信号相关的导线延迟，这就使在0.5ns内完成读和写这一时间要求变得几乎不可能。随着所需的高频率的增加，使得操作处理器50变得不可能。

因此，需要一种机制来提供在同一位置对于紧接的读和写存取时间更快速。检验器72提供的解决方案是将这一问题分为两部分。特别是，上面已经描述了检验器72使用传统的标记板的解决方案,通过大于两个周期来分割相关性。

但是，也可以通过使用一个检验器矩阵引擎来解决在非常接近的时间内指令间的相关性。它是通过确定在接近时间内某个指令是依赖于在这一时间之前的某一指令来实现的。它设置了一个相关性矩阵。当指令经过判定逻辑101时，发出指令执行是否正确的信号。该信号与相关性信息相结合很快地确定一条指令执行的是否正确。因此对于一个小的时间片(time slice)，检验器矩阵引擎提供了一个高速解决方案。

附图4表示按照本发明一个实施例的检验器72的方框图，它包括检验器矩阵引擎100和标记板104。检验器矩阵引擎100在一种高速方式下完成附图3所示的步骤。

检验器矩阵引擎100的操作可以结合一系列被调度的指令来进行更好的描述。附图5表示调度程序30在连续的调度周期内所调度指令的列表。每一个指令（指令1，指令2，等等。）包括两个源和一个目的。因此，例如在调度周期1中，指令1被调度。指令1的源是寄存器10（r10）和寄存器11。指令1的目的是寄存器12。在调度周期2中,指令2被调度。指令2的源是寄存器12和寄存器10，目的是寄存器13。指令2依赖于指令1因为指令2的其中的一个源，寄存器12，由指令1生成(寄存器12是指令1的目的寄存器)。在调度周期3，指令3被调度，诸如此类，经过10个调度周期。但是在调度周期7，9和10没有指令被调度。

附图6A-6J表示在附图5的每一个调度周期内检验器矩阵引擎100的状态。如图所示，例如，在附图6A中，检验器矩阵引擎100包括一个保持缓冲器或目的寄存器210，以及一个相关性矩阵200。保持缓冲器210包括多个入口，或者行,其对应于一条指令。相关性矩阵200包括对应于保持缓冲器210入口的多个行，以及多个列。每一列对应于保持缓冲器210中一个入口的相关性。

在附图6A-6J中，保持缓冲器210和相关性矩阵200包括三个入口。保持缓冲器210进一步包括一个有效列204和一个目的列206。有效列204中的“1”代表一个有效指令处在相应的入口。指令的目的入口写入目的列206。写标记202（“wr”）指向最先存储的指令。保持缓冲器210必须包括用于源的两个端口以窥探目的。如果源与其目的相匹配，那么适当的相关性位被设置。

因此，在附图6A的调度周期1，指令1被写入保持缓冲器210的第一个入口。指令1的目的是寄存器12，指令1的源是寄存器10和寄存器11（被写标记202指向的指令的源表示在保持缓冲器210的底部）。附图6A的第二和第三入口不含

有有效指令，因此在有效列204的这些入口用“0”来表示。指令1依赖于那些指令的目的与指令1的源相匹配的指令（即，寄存器10或寄存器11）。

相关性矩阵200包括位或者元素，它们对应于保持缓冲器210的一个入口的一个指令与保持缓冲器210另一个入口的一个指令的相关性。“D”是其中的一个元素，表示该指令依赖于保持缓冲器210的指令入口，该入口与相关性矩阵的列数对应。例如，参考附图6B，第二个入口的第一列中的“D”表示保持缓冲器210的第二个入口的指令依赖于保持缓冲器210的第一个入口的指令所产生的结果。换句话说，指令2（保持缓冲器210第二个入口的指令）依赖于指令1（保持缓冲器210的的一个入口指令）。因为一个指令不能依赖于自身，沿着相关性矩阵200对角线的每一格被标记为“X”。

在附图6A中（调度周期1），指令1被在入口1写入保持缓冲器210。

在附图6B中（调度周期2），指令2被在第二个入口写入保持缓冲器210。指令2的源（寄存器10和寄存器12）相对于保持缓冲器210的有效目的被匹配。基于匹配来确定相关性并且相关性被相应地更新。在该例中，矩阵200的列1入口2中写入“D”表示指令2依赖于指令1。此外，在调度周期2期间通过指令1的外部回放信号78一个外部回放指示被检验器72接收。

在附图6C（调度周期3）中，指令3在入口3写入保持缓冲器210，并且矩阵200的第一列，第二和第三入口的“D”表示指令2和指令3依赖于指令2。此外，在调度周期3，指令1被检验器72检验，并且由于在调度周期2的外部回放信号，使检验失效（即被回放）。因为这一失效，指令1通过线111被传送到检验器72的回放多路复用器56中。

在附图6D（调度周期4）中，指令4在入口1写入保持缓冲器210并且入口2的指令2被检验器72检验。指令2使检验失效并且被回放，因为在该行中至少有一个“D”对应于矩阵200的入口2，表明指令2所依赖的指令执行不正确。

在附图6E（调度周期5）中，指令5写入保持缓冲器210的入口2并且入口3的指令3被检验器72检验。指令3使检验失效并且被回放，因为在该行中至少有一个“D”对应于矩阵200的入口3，表明指令3所依赖的指令执行不正确。

在附图6F（调度周期6）中，指令6写入保持缓冲器210的入口3并且入口1的指令4被检验器72检验。指令4通过检验并且回放安全，因为在该行中没有一个“D”对应于矩阵200的入口1，表明指令4不依赖于没有正确执行的指令。

此外，在外部回放信号78上设有所接收的给指令4的外部回放状态。指令4通过线109被发送到退役单元62中,并且由于指令4被正确地执行,对应于指令4（即第一列）的矩阵200的所有列被清除。清除这些列意味着删除列中的所有“D”。

在附图6G（调度周期7）中，没有指令被调度，因此“0”被写入到保持缓冲器210中。此外，入口2的指令5被检验器72调度。指令5通过检验并且回放安全因为在行没有“D”对应于矩阵200的入口2，表示指令5不依赖于没有正确执行的指令。此外，在外部回放信号78上设有所接收的给指令4的外部回放状态。指令5被发送到退役单元62中，并且由于指令5被正确地执行，对应于指令5（即第二列）的矩阵200的所有列被清除。

在附图6H（调度周期8）中，指令7写入保持缓冲器210的入口2并且入口3的指令6被检验器72检验。指令6通过检验并且回放安全，因为在该行中没有一个“D”对应于矩阵200的入口3，表明指令6不依赖于没有正确执行的指令。此外，在外部回放信号78上设有所接收的给指令4的外部回放状态。指令6通过线109被发送到退役单元62中,并且由于指令6被正确地执行，对应于指令6（即第三列）的矩阵200的所有列被清除。

在附图6I（调度周期9）中，没有指令被调度因此“0”被在入口3处写入保持缓冲器210中。此外，没有指令被检验器72检验。

最后，在附图6J（调度周期10）中，没有指令被调度因此“0”被写入保持缓冲器210的入口1中。指令7被检验器72检验并且由于在矩阵2的入口3中没有“D”，指令7回放安全并且被发送到退役单元62中。

正如所披露的，检验器72接收指令并且确定指令是否执行正确。检验器72以标记板和外部回放信号为基础作出判断。为了尽快作出判断，检验器72可以包括一个检验矩阵引擎。

本发明的几个实施例在这里已经被详细地说明和描述。然而，只要对本发明进行修改和变化都被上述教导所覆盖并包含在所附的权利要求的范围内，而不脱离本发明的精神和保护范围。

例如，尽管所描述的检验器是回放系统的一部分，检验器可以用在处理器的其它应用如奇偶检验，任何存储器的操作，检验地址的相关性，或任何需要判断相关性的其它应用。

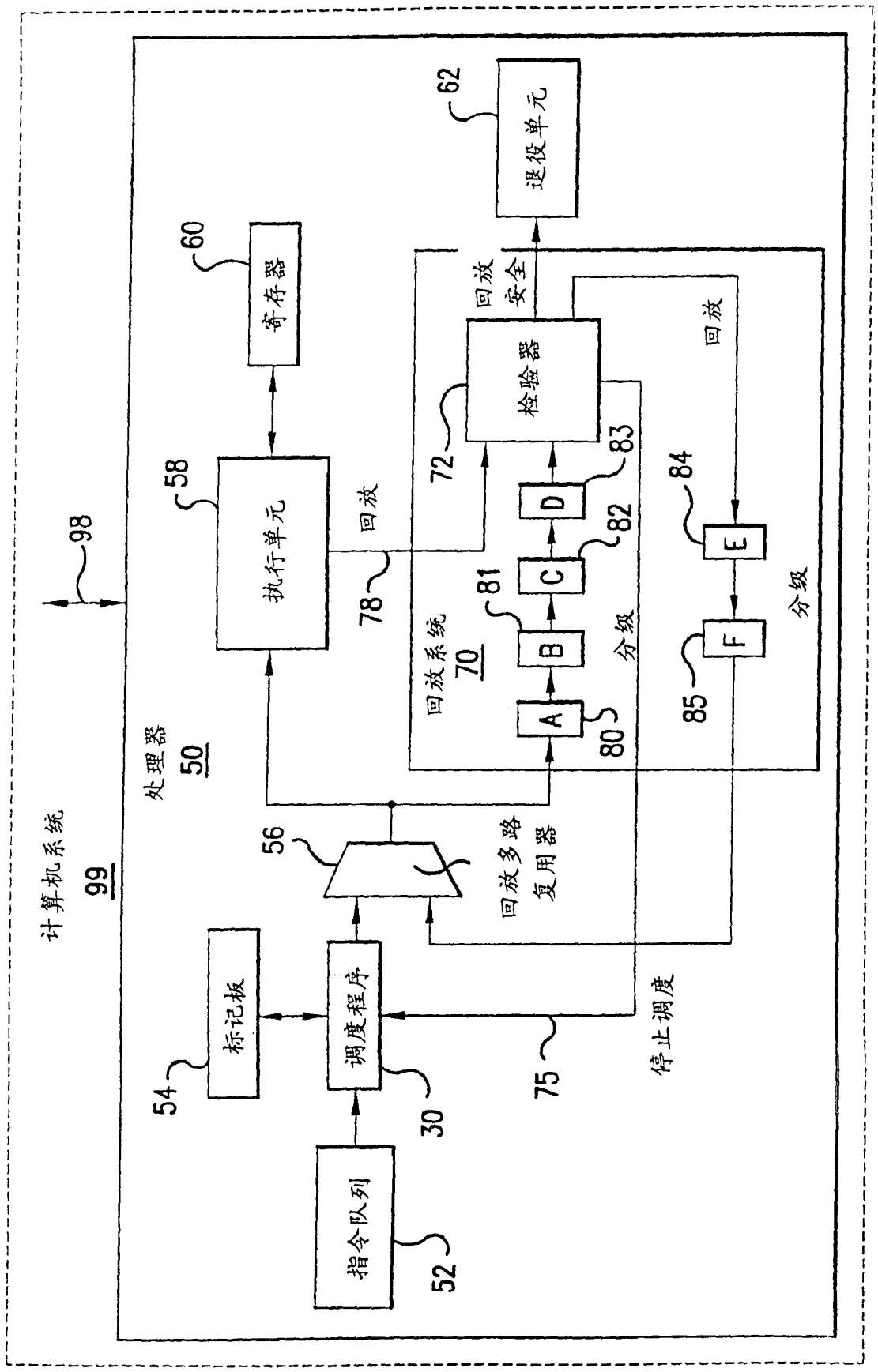

计算机系统
99
处理器
50
98
指令队列
52
标记板
54
调度程序
30
寄存器
60
执行单元
58
回放多路复用器
56
回放系统
70
A
B
C
D
E
F
80
81
82
83
84
85
78
回放
检验器
72
回放安全
退役单元
62
分级
停止调度
75

图 1

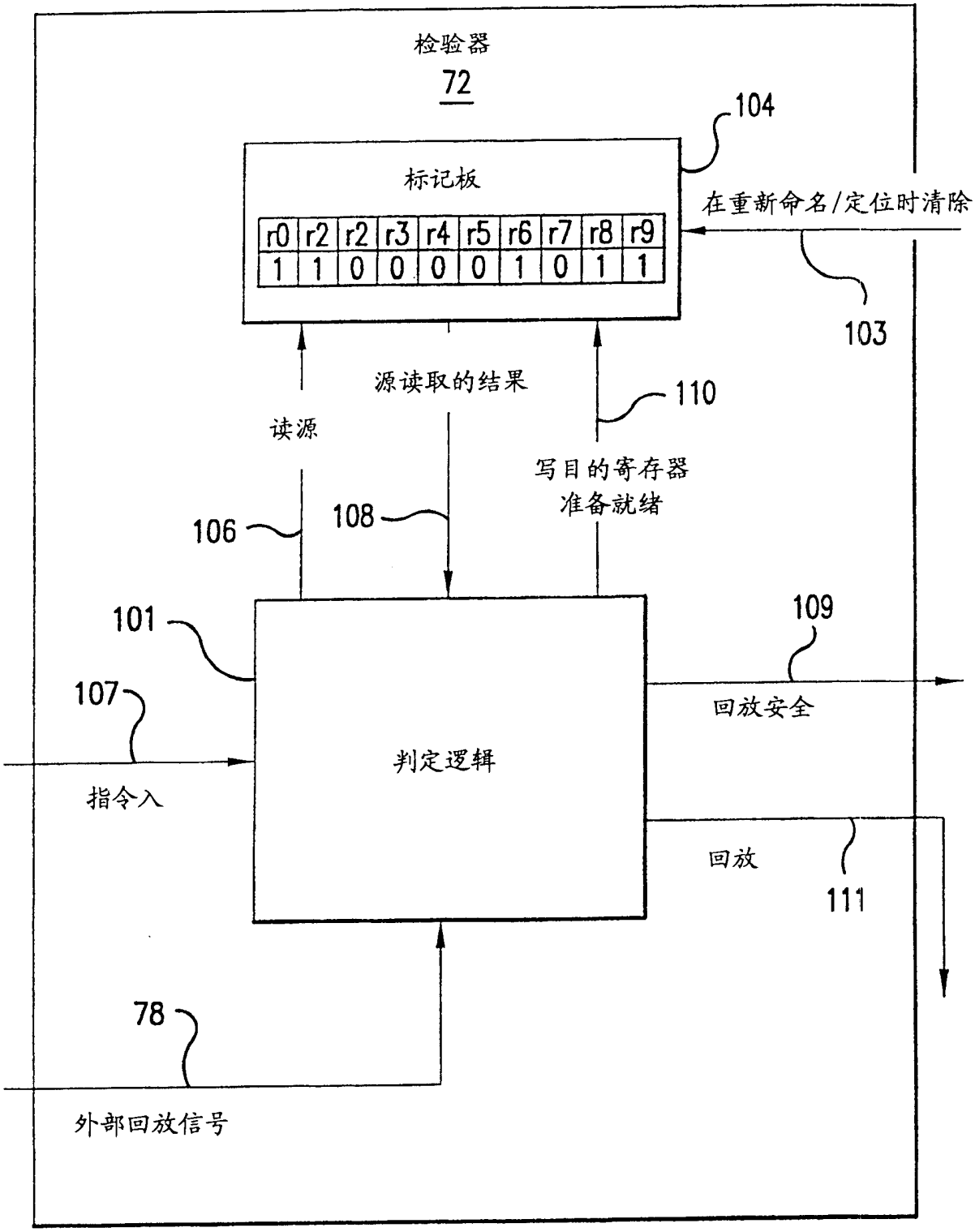

检验器
72
104
标记板
r0 r2 r2 r3 r4 r5 r6 r7 r8 r9
1 1 0 0 0 0 1 0 1 1
在重新命名/定位时清除
103
源读取的结果
110
读源
写目的寄存器
准备就绪
106
108
101
109
107
回放安全
判定逻辑
指令入
回放
111
78
外部回放信号

图 2

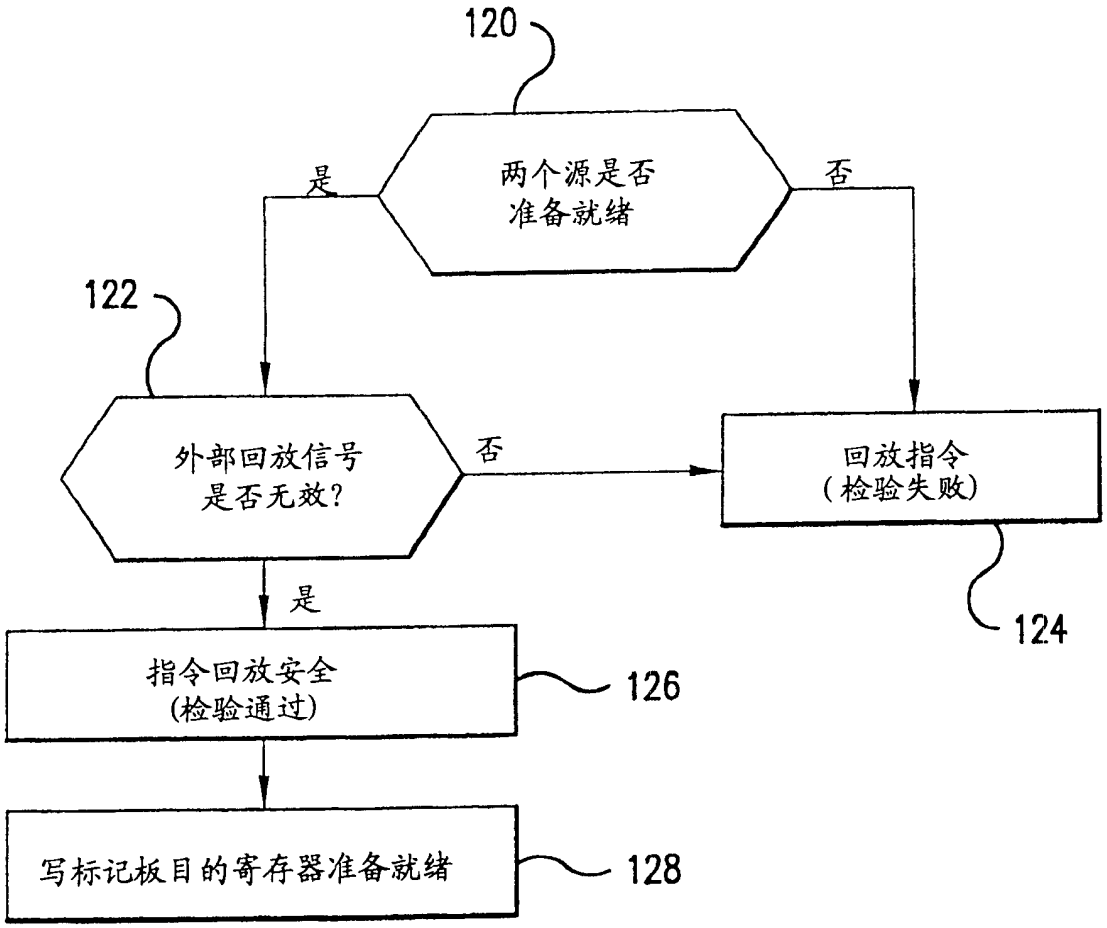

120
两个源是否
准备就绪
是
否
122
外部回放信号
是否无效?
否
回放指令
(检验失败)
124
是
指令回放安全
(检验通过)
126
写标记板目的寄存器准备就绪
128

图 3

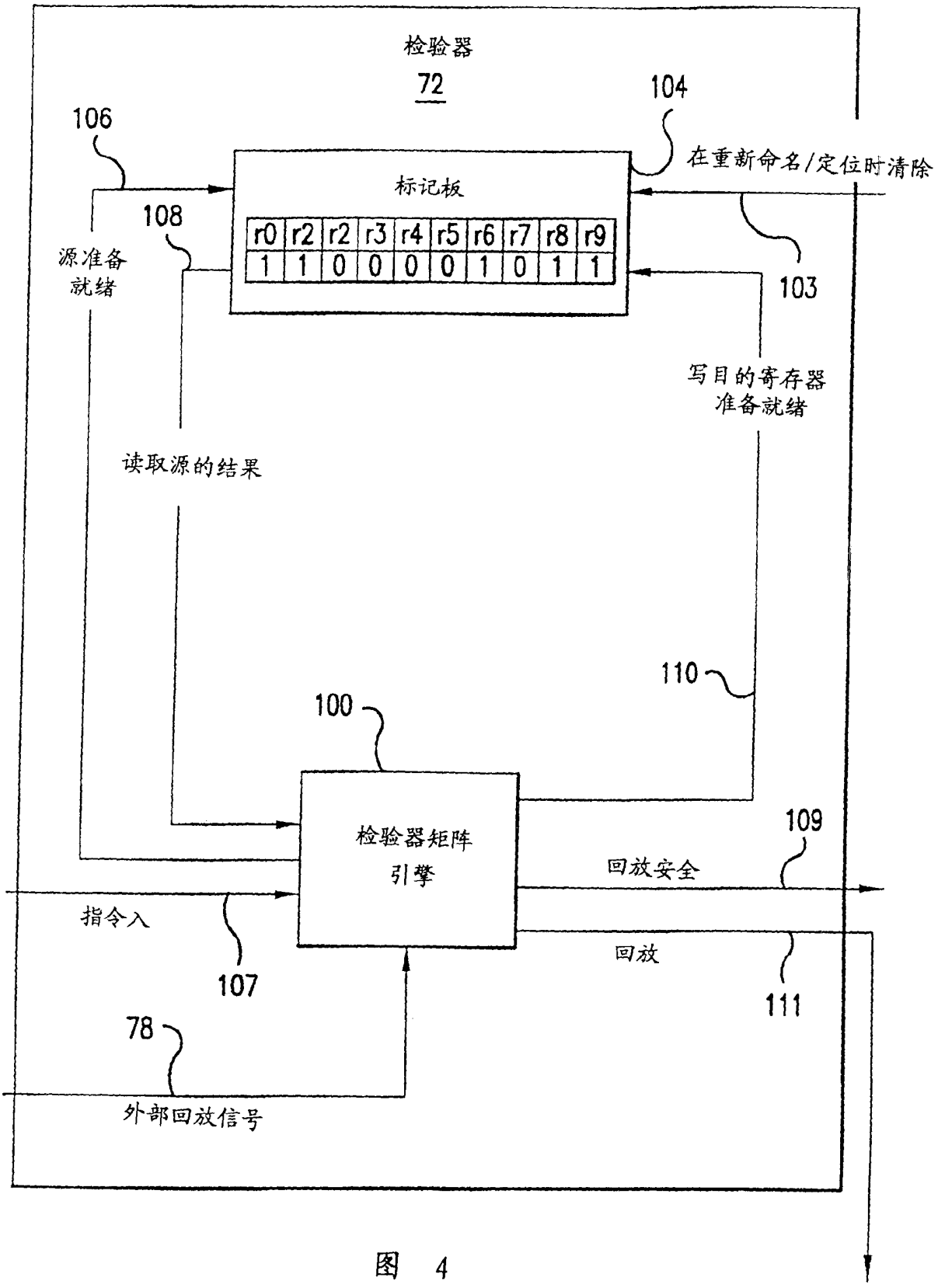
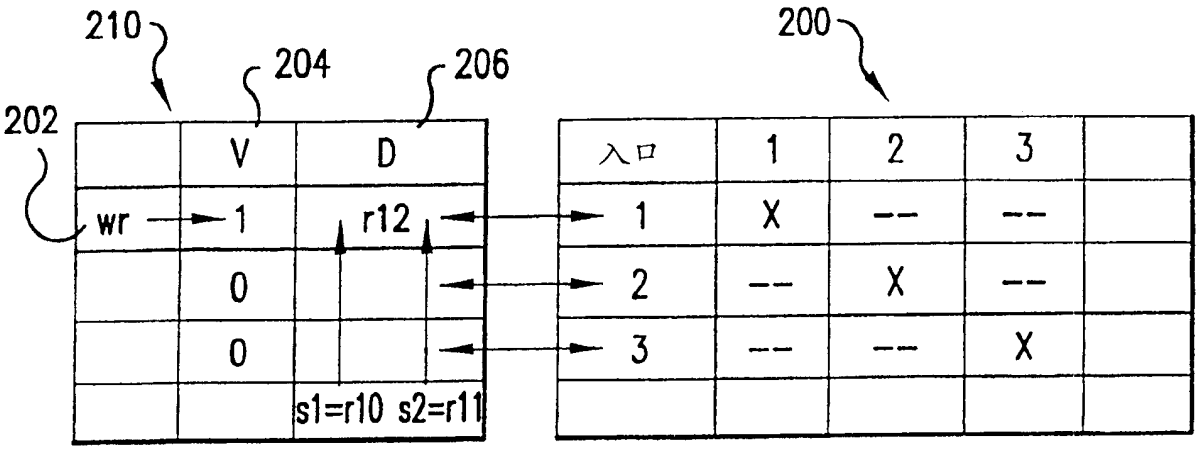

检验器
72
104
106
在重新命名/定位时清除
标记板
r0 r2 r2 r3 r4 r5 r6 r7 r8 r9
1 1 0 0 0 0 1 0 1 1
108
源准备
就绪
103
写目的寄存器
准备就绪
读取源的结果
110
100
检验器矩阵
引擎
109
回放安全
指令入
107
回放
111
78
外部回放信号

图 4

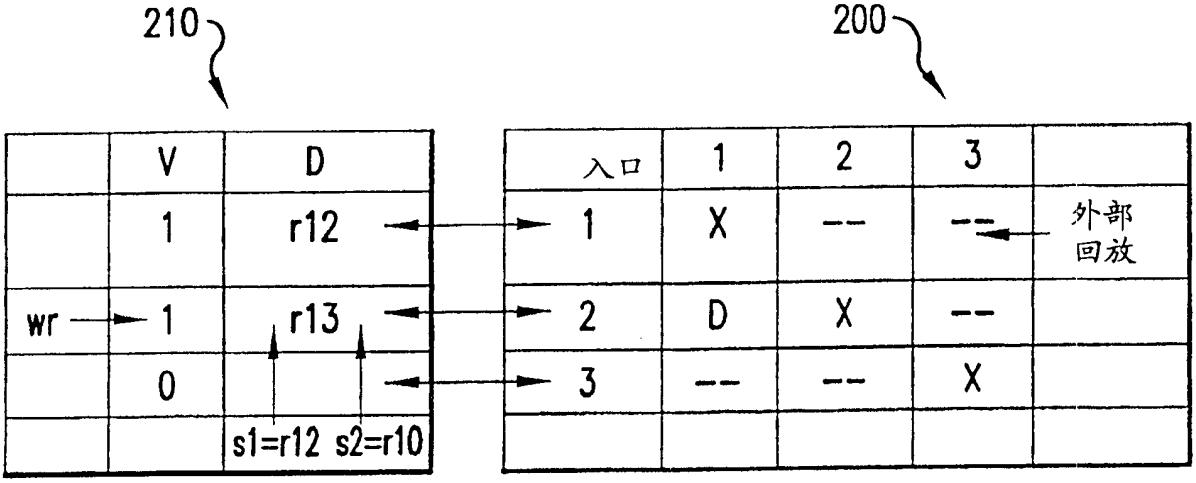
调度周期	指令序号	源 1	源2	目的
1	I1	r10	r11	r12
2	I2	r12	r10	r13
3	I3	r12	r13	r14
4	I4	r8	r9	r15
5	I5	r15	r10	r16
6	I6	r15	r16	r19
7	--	--	--	--
8	I7	r19	r9	r18
9	--	--	--	--
10	--	--	--	--

图 5



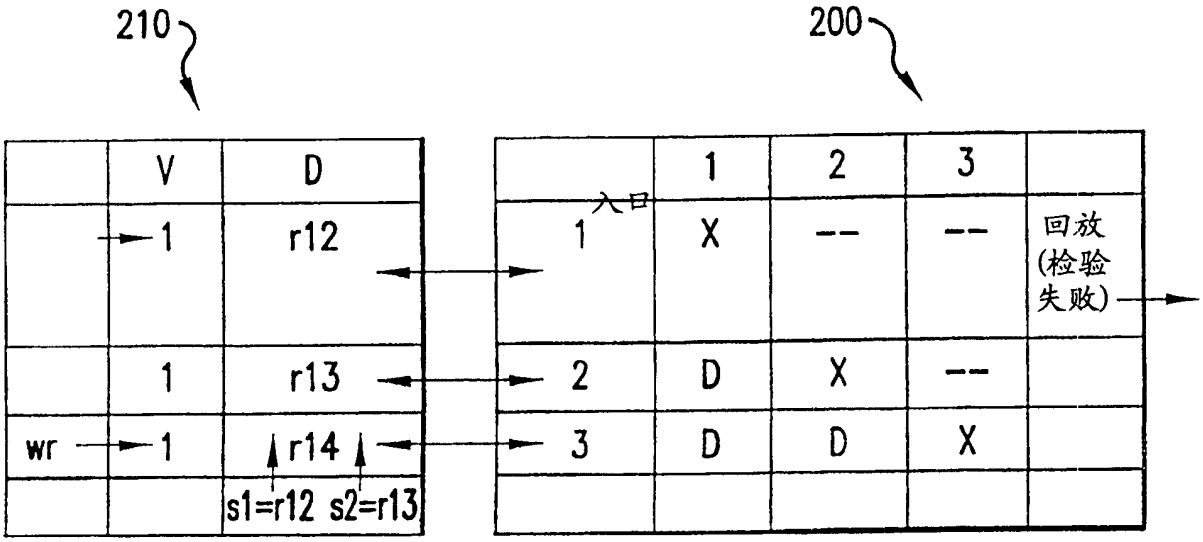
调度周期1

图 6A



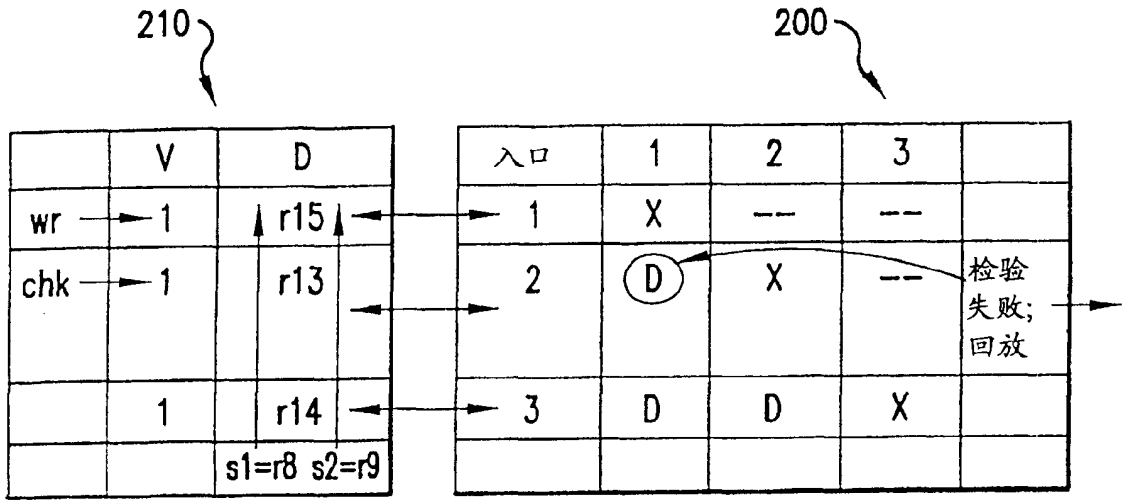
调度周期2

图 6B



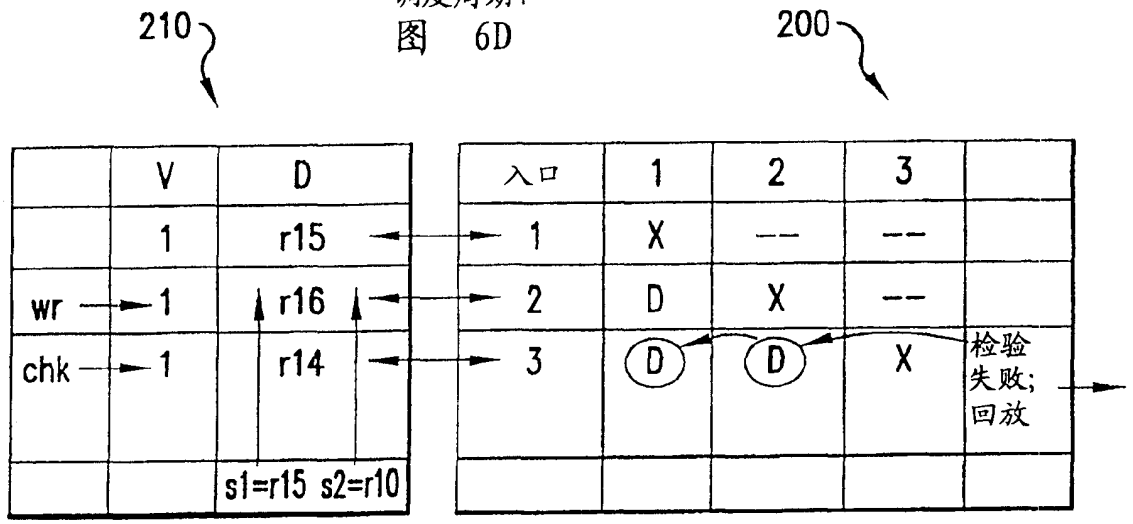
调度周期3

图 6C



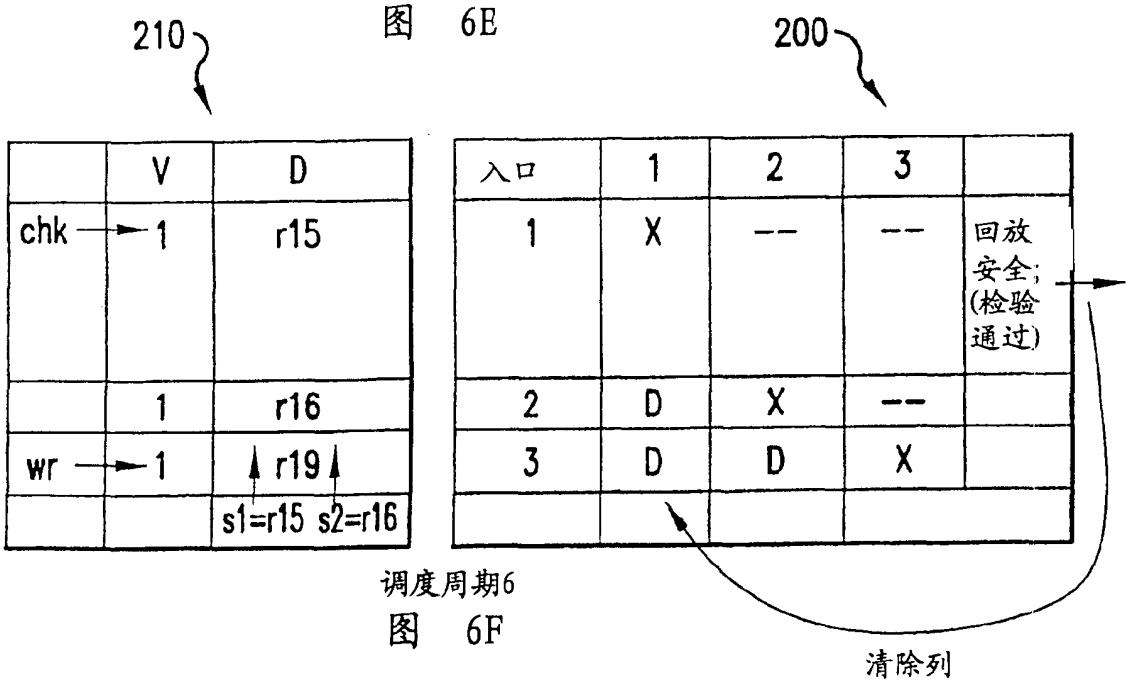
调度周期4

图 6D



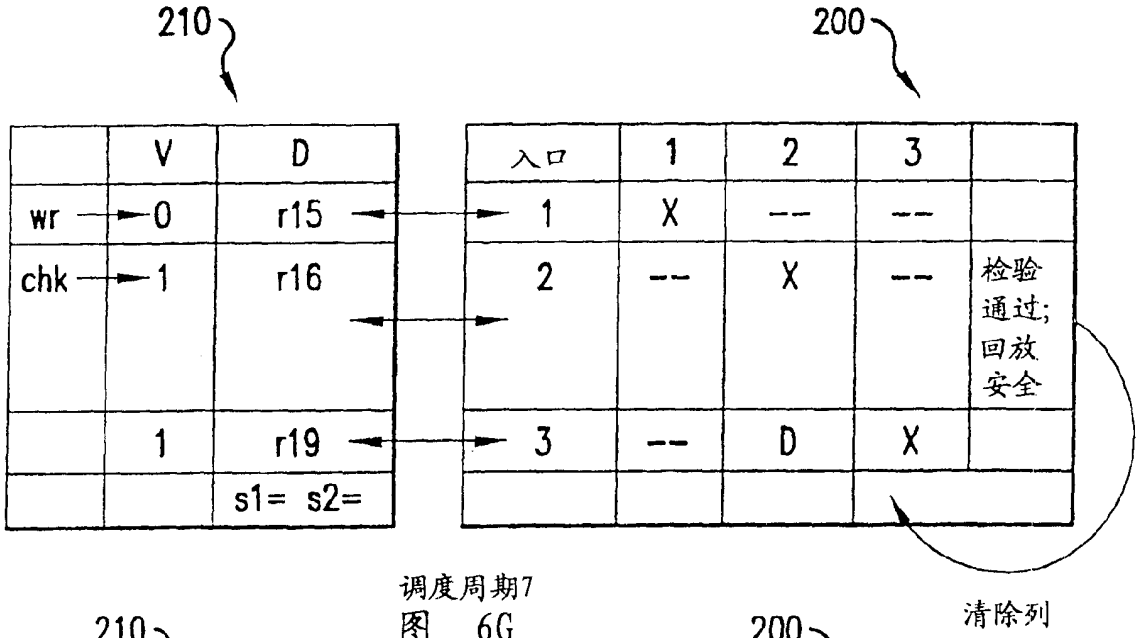
调度周期5

图 6E



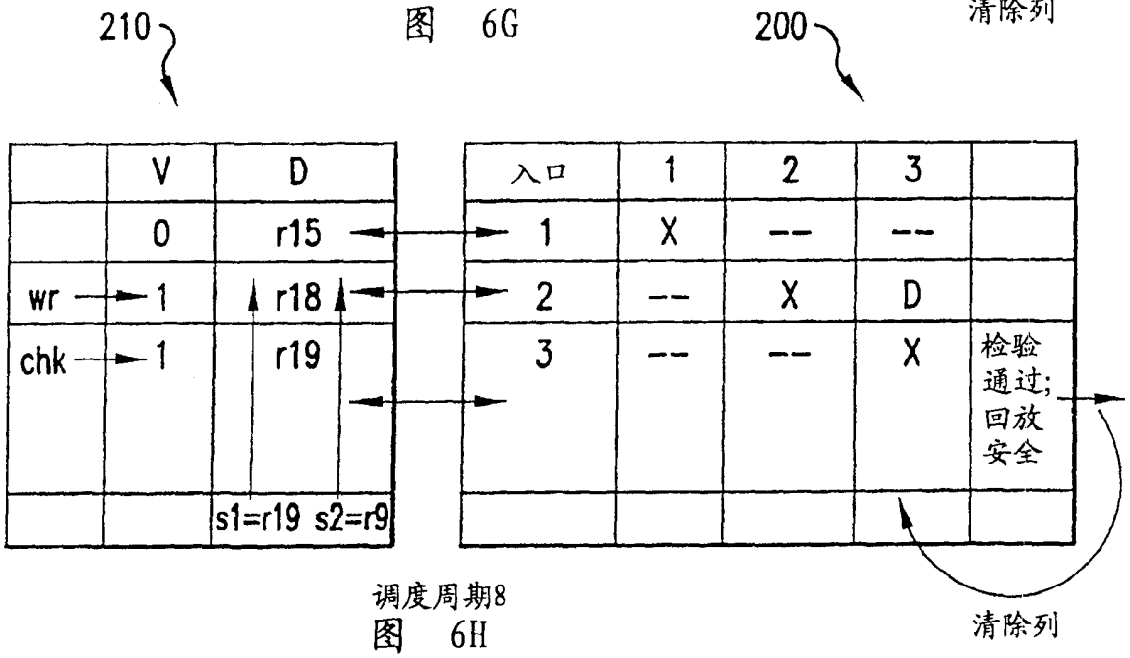
调度周期6

图 6F



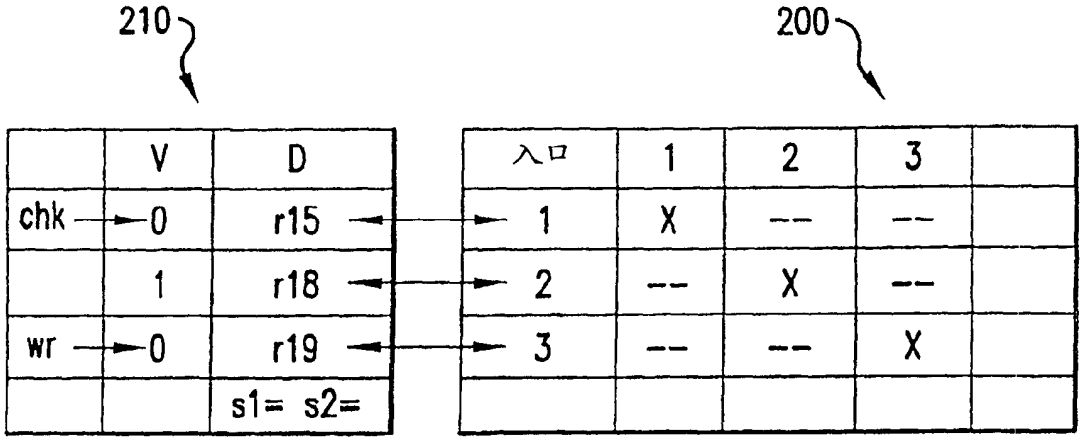
调度周期7

图 6G



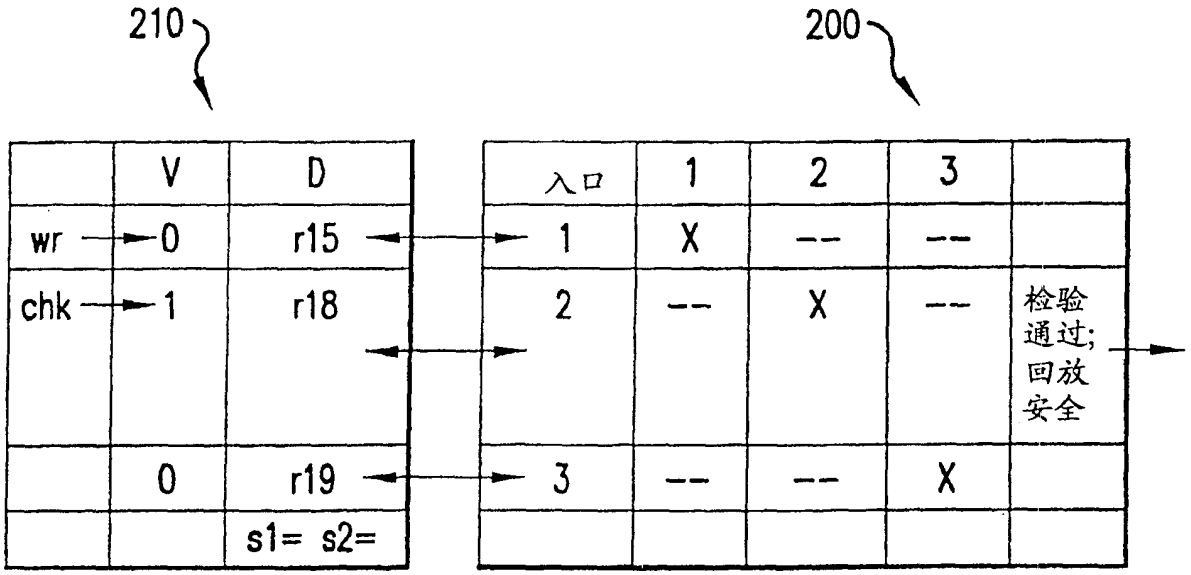
调度周期8

图 6H



调度周期9

图 6I



调度周期10

图 6J