(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 105190579 B
(45)授权公告日 2018.04.06

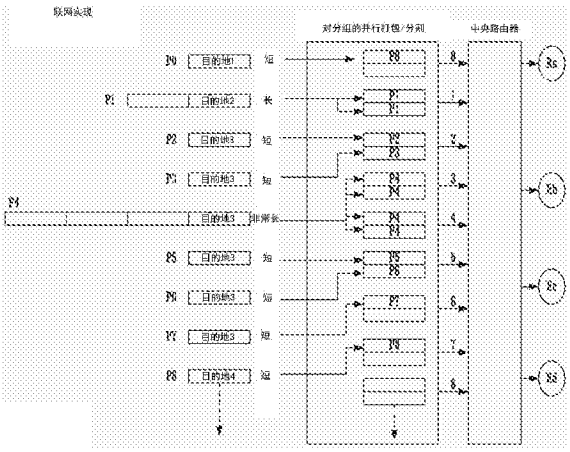
(21)申请号 201480024465.9
(22)申请日 2014.03.13
(65)同一申请的已公布的文献号
申请公布号 CN 105190579 A
(43)申请公布日 2015.12.23
(30)优先权数据
61/852,389 2013.03.15 US
(85)PCT国际申请进入国家阶段日
2015.10.30
(86)PCT国际申请的申请数据
PCT/US2014/026427 2014.03.13
(87)PCT国际申请的公布数据
WO2014/151773 EN 2014.09.25
(73)专利权人 英特尔公司
地址 美国加利福尼亚州
(72)发明人 穆罕默德·阿布达拉
(74)专利代理机构 上海专利商标事务所有限公司 31100
代理人 黄嵩泉
(51)Int.Cl.
G06F 13/14(2006.01)
G06F 13/38(2006.01)
(56)对比文件
US 2004/0202178 A1,2004.10.14,
US 2004/0202178 A1,2004.10.14,
US 7521961 B1,2009.04.21,
US 5303362 A,1994.04.12,
US 2011047354 A1,2011.02.24,
US 2011314444 A1,2011.12.22,
US 2006013207 A1,2006.01.19,
US 5937202 A,1999.08.10,
EP 0935200 A1,1999.08.11,
审查员 李思彤
权利要求书3页 说明书11页 附图21页

(54)发明名称

一种用于实现线路速度互连结构的方法

(57)摘要

一种用于线路速度互连处理的方法。该方法包括从输入通信路径接收初始输入，通过利用第一阶段互连并行处理器，执行对所述初始输入的预分类，以创建中间输入，以及通过利用第二阶段互连并行处理器，执行对所述中间输入的最终组合和分割，以创建结果输出。该方法还包括以线路速度将所述结果输出传输出所述第二阶段。



CN 105190579 B

1.一种用于线路速度互连处理的方法,包括:

从输入通信路径接收初始输入;

通过利用第一阶段互连并行处理器,执行对所述初始输入的预分类,以创建中间输入,其中所述第一阶段互连处理器通过以与在将被检查以用于配对的所述初始输入中间识别候选者并行地对所述初始输入执行预分类和预分簇处理来工作;

通过利用第二阶段互连并行处理器,执行对所述中间输入的最终组合和分割,以创建结果输出;以及

以线路速度将所述结果输出传输出所述第二阶段。

2.根据权利要求1所述的方法,其中所述第二阶段互连处理器通过并行地对所述中间输入执行位置混排、配对和分割从而以线路速度创建所述结果输出来工作。

3.根据权利要求1所述的方法,其中所述线路速度互连处理被实现在联网的体系架构中,其中所述初始输入包括联网的分组。

4.根据权利要求1所述的方法,其中所述线路速度互连处理被实现在高速缓存访问体系架构中,其中所述初始输入包括对高速缓存线的数据的访问请求。

5.根据权利要求1所述的方法,其中所述线路速度互连处理被实现在仲裁体系架构中,其中所述初始输入包括利用输出带宽的流,并且其中所述仲裁体系架构并行地利用频率和/或时间复用来在所述输入流中间进行仲裁以创建结果输出流。

6.根据权利要求1所述的方法,其中所述线路速度互连处理被实现在计算机指令体系架构解码器中,其中所述初始输入包括计算机指令,所述计算机指令将被并行地组合或分割成机器指令。

7.根据权利要求1所述的方法,其中所述线路速度互连处理被实现在DRAM访问体系架构中,其中所述初始输入包括对DRAM页的访问,所述对DRAM页的访问将被并行地配对或分割成对DRAM页的优化的结果访问。

8.一种非暂态计算机可读存储器,具有计算机可读代码,当计算机系统执行所述计算机可读代码时,导致所述计算机系统实现用于线路速度互连处理的方法,包括:

从输入通信路径接收初始输入;

通过利用第一阶段互连并行处理器,执行对所述初始输入的预分类,以创建中间输入,其中所述第一阶段互连处理器通过以与在将被检查以用于配对的所述初始输入中间识别候选者并行地对所述初始输入执行预分类和预分簇处理来工作;

通过利用第二阶段互连并行处理器,执行对所述中间输入的最终组合和分割,以创建结果输出;以及

以线路速度将所述结果输出传输出所述第二阶段。

9.根据权利要求8所述的计算机可读存储器,其中所述第二阶段互连处理器通过并行地对所述中间输入执行位置混排、配对和分割从而以线路速度创建所述结果输出来工作。

10.根据权利要求8所述的计算机可读存储器,其中所述线路速度互连处理被实现在联网的体系架构中,其中所述初始输入包括联网的分组。

11.根据权利要求8所述的计算机可读存储器,其中所述线路速度互连处理被实现在高速缓存访问体系架构中,其中所述初始输入包括对高速缓存线的数据的访问请求。

12.根据权利要求8所述的计算机可读存储器,其中所述线路速度互连处理被实现在仲

裁体系架构中，其中所述初始输入包括利用输出带宽的流，并且其中所述仲裁体系架构并行地利用频率和/或时间复用来在所述输入流中间进行仲裁以创建结果输出流。

13.根据权利要求8所述的计算机可读存储器，其中所述线路速度互连处理被实现在计算机指令体系架构解码器中，其中所述初始输入包括计算机指令，所述计算机指令将被并行地组合或分割成机器指令。

14.根据权利要求8所述的计算机可读存储器，其中所述线路速度互连处理被实现在DRAM访问体系架构中，其中所述初始输入包括对DRAM页的访问，所述对DRAM页的访问将被并行地配对或分割成对DRAM页的优化的结果访问。

15.一种计算机系统，包括：

系统存储器；

中央处理单元，所述中央处理单元与所述系统存储器耦合，其中所述中央处理单元执行计算机可读代码，并且使得所述计算机系统实现用于线路速度互连处理的方法，包括：

从输入通信路径接收初始输入；

通过利用第一阶段互连并行处理器，执行对所述初始输入的预分类，以创建中间输入，其中所述第一阶段互连处理器通过以与在将被检查以用于配对的所述初始输入中间识别候选者并行地对所述初始输入执行预分类和预分簇处理来工作；

通过利用第二阶段互连并行处理器，执行对所述中间输入的最终组合和分割，以创建结果输出；以及

以线路速度将所述结果输出传输出所述第二阶段。

16.根据权利要求15所述的计算机系统，其中所述第二阶段互连处理器通过并行地对所述中间输入执行位置混排、配对和分割从而以线路速度创建所述结果输出来工作。

17.根据权利要求15所述的计算机系统，其中所述线路速度互连处理被实现在联网的体系架构中，其中所述初始输入包括联网的分组。

18.根据权利要求15所述的计算机系统，其中所述线路速度互连处理被实现在高速缓存访问体系架构中，其中所述初始输入包括对高速缓存线的数据的访问请求。

19.根据权利要求15所述的计算机系统，其中所述线路速度互连处理被实现在仲裁体系架构中，其中所述初始输入包括利用输出带宽的流，并且其中所述仲裁体系架构并行地利用频率和/或时间复用来在所述输入流中间进行仲裁以创建结果输出流。

20.根据权利要求15所述的计算机系统，其中所述线路速度互连处理被实现在计算机指令体系架构解码器中，其中所述初始输入包括计算机指令，所述计算机指令将被并行地组合或分割成机器指令。

21.根据权利要求15所述的计算机系统，其中所述线路速度互连处理被实现在DRAM访问体系架构中，其中所述初始输入包括对DRAM页的访问，所述对DRAM页的访问将被并行地配对或分割成对DRAM页的优化的结果访问。

22.一种用于线路速度互连处理的设备，包括：

用于从输入通信路径接收初始输入的装置；

用于通过利用第一阶段互连并行处理器来执行对所述初始输入的预分类以创建中间输入的装置，其中所述第一阶段互连处理器通过以与在将被检查以用于配对的所述初始输入中间识别候选者并行地对所述初始输入执行预分类和预分簇处理来工作；

用于通过利用第二阶段互连并行处理器来执行对所述中间输入的最终组合和分割以创建结果输出的装置;以及

用于以线路速度将所述结果输出传输出所述第二阶段的装置。

23.根据权利要求22所述的设备,其中所述第二阶段互连处理器通过并行地对所述中间输入执行位置混排、配对和分割从而以线路速度创建所述结果输出来工作。

24.根据权利要求22所述的设备,其中所述线路速度互连处理被实现在联网的体系架构中,其中所述初始输入包括联网的分组。

25.根据权利要求22所述的设备,其中所述线路速度互连处理被实现在高速缓存访问体系架构中,其中所述初始输入包括对高速缓存线的数据的访问请求。

26.根据权利要求22所述的设备,其中所述线路速度互连处理被实现在仲裁体系架构中,其中所述初始输入包括利用输出带宽的流,并且其中所述仲裁体系架构并行地利用频率和/或时间复用来在所述输入流中间进行仲裁以创建结果输出流。

27.根据权利要求22所述的设备,其中所述线路速度互连处理被实现在计算机指令体系架构解码器中,其中所述初始输入包括计算机指令,所述计算机指令将被并行地组合或分割成机器指令。

28.根据权利要求22所述的设备,其中所述线路速度互连处理被实现在DRAM访问体系架构中,其中所述初始输入包括对DRAM页的访问,所述对DRAM页的访问将被并行地配对或分割成对DRAM页的优化的结果访问。

一种用于实现线路速度互连结构的方法

[0001] 相关申请的交叉引用

[0002] 本申请要求2013年3月15日递交的、Mohammad A.Abdallah的名为“A METHOD FOR IMPLEMENTING A LINE SPEED INTERCONNECT STRUCTURE”、专利申请号为61/852389的共同未决共同转让的美国临时专利申请的权益，其全部内容被结合于此。

技术领域

[0003] 本申请一般涉及数字计算机系统，更具体地，涉及用于选择包括指令序列的指令的系统和方法。

背景技术

[0004] 要求处理器处理多个或依赖性的或完全独立的任务。这种处理器的内部状态通常包括寄存器，该寄存器可以保持程序执行的每个特定瞬间的不同值。在程序执行的每个瞬间，内部状态图像被称作处理器的体系架构状态。

[0005] 当代码执行被切换以运行另一函数(例如，另一线程、进程或程序)时，则不得不保存机器/处理器的状态，使得新的函数可以利用内部寄存器来构建其新的状态。一旦新的函数被终止，则可丢弃其状态，并且将恢复之前的上下文的状态，并且执行继续。这种切换处理被称为上下文切换，并且通常包括数十或成百个周期，特别是针对利用大数量的寄存器(例如，64，128，256)和/或乱序执行的现代体系架构而言。

[0006] 在线程感知的硬件体系架构中，对硬件来说支持针对有限数量的硬件支持的线程的多个上下文状态是正常的。在该情形中，针对每个所支持的线程，硬件复制所有体系架构状态元件。这消除了当执行新线程时进行上下文切换的需要。但是，这仍具有多个缺陷，即针对每个硬件中所支持的额外线程，复制所有体系架构状态元件(即，寄存器)的区域、功率和复杂度。另外，如果软件线程的数量超过硬件明确支持的线程的数量，则必须仍执行上下文切换。

[0007] 这变得很平常，因为基于要求大量线程的精确粒度，并行化是需要的。带有复制的上下文状态硬件存储器的硬件线程感知的体系架构对非线程化软件代码无益，并且仅减少了线程化的软件的上下文切换次数。但是，那些线程通常被构建用于粗粒度并行化，并且导致用于初始化和同步的繁重的软件开销，使得诸如函数调用和循环并行执行之类的细粒度并行化没有高效的线程初始化/自动生成。这种所描述的开销伴随有针对非明确/容易并行化/线程化的软件代码利用现有技术的编译器或用户并行化技术来进行对这种代码的自动并行化的困难。

发明内容

[0008] 在一个实施例中，本发明被实现为一种用于线路速度互连处理的方法。该方法包括从输入通信路径接收初始输入，通过利用第一阶段互连并行处理器，执行对所述初始输入的预分类，以创建中间输入，以及通过利用第二阶段互连并行处理器，执行对所述中间输

入的最终组合和分割，以创建结果输出。该方法还包括以线路速度将所述结果输出传输出所述第二阶段。

[0009] 前述是概要，并且因此必须包括简化、概括，以及省略细节；因此，本领域技术人员将理解，该概要是阐释性的，并且不意欲以任何方式限制。鉴于以下给出的非限制性详细的描述，完全由权利要求所限定的本发明的其他方面、创造性特征和优势将变得明显。

附图说明

[0010] 本发明由示例示出，而非限制，在附图的示图中，相似的参考标号指相似的元件。

[0011] 图1示出了根据本发明的一个实施例的线路速度互连结构的概略图。

[0012] 图2示出了根据本发明的一个实施例的对线路速度互连结构的联网实现的概述。

[0013] 图3示出了根据本发明的一个实施例的由线路速度互连结构所使用的处理的某些术语。

[0014] 图4示出了根据本发明的一个实施例的如何使用FB值和NP值的第二示图。

[0015] 图5示出了根据本发明的一个实施例的示出了初始位置/分组号码、新的分组中的新的位置号码，以及结果输出分组号码之间的关系的另一示图。

[0016] 图6示出了根据本发明的一个实施例的上述模2评估方程的操作的更加详细的示图。

[0017] 图7描述了根据本发明的一个实施例的高速缓存访问实现的操作，其中高速缓存的访问被重新定位并布置到针对存储器层级的高速缓存线的大小对齐的新的发出的高速缓存访问中。

[0018] 图8示出了根据本发明的一个实施例的、根据高速缓存端口的重新映射处理的示图。

[0019] 图9示出了根据本发明的一个实施例的、根据高速缓存端口的两阶段重新映射处理的示图。

[0020] 图10描述了根据本发明的一个实施例的计算机指令实现的操作，其中计算机指令被重新定位并融合或分成新的发出的计算机指令。

[0021] 图11描述了根据本发明的一个实施例的仲裁器实现的操作，其中不同的源针对多个发出的目的地被仲裁。

[0022] 图12示出了根据本发明的一个实施例的仲裁器实现的另一示图，其中不同的源针对多个发出的目的地被仲裁。

[0023] 图13示出了根据本发明的一个实施例的仲裁器实现的另一示图，其中不同的源针对多个发出的目的地被仲裁。

[0024] 图14示出了根据本发明的一个实施例的对示出了模2评估方程的电子表格的图形描绘。

[0025] 图15示出了根据本发明的一个实施例的对累积和评估处理的操作。

[0026] 图16示出了根据本发明的一个实施例的、描绘了用于并行执行累积和评估的电路的示图。

[0027] 图17示出了根据本发明的一个实施例的执行对累积和的评估的电路的示图。

[0028] 图18示出了根据本发明的一个实施例的执行对累积和的评估的第二电路的示图。

[0029] 图19示出了并行加法器实现的示例性体系架构。

[0030] 图20示出了根据本发明的一个实施例的、描绘了并行进位保存加法器的示图。

[0031] 图21示出了根据本发明的一个实施例的阶段优化高速度并行加法器的实施例。

具体实施方式

[0032] 虽然已经与一个实施例一起描述了本发明，但是，本发明并不意欲限制为此处所给出的具体形式。相反，意欲覆盖这种替换例、修改例和等同例，只要能合理地被包括在所附权利要求所限定的本发明的范围内即可。

[0033] 在以下的详细的描述中，已经给出了众多具体的细节，诸如具体的方法顺序、结构、元件和连接。但是，应理解，无需利用这些以及其他具体细节来实践本发明的实施例。在其他情况中，已经省略了已知的结构、元素或连接，或者未以特定细节来描述，以便避免不必要地模糊本描述。

[0034] 本说明书中对“一个实施例”或“实施例”的引用意欲指示与实施例一起描述的特定特征、结构或特点被包括在本发明中的至少一个实施例中。本说明书中各处所出现的短语“在一个实施例中”并不必须都指相同的实施例，单独的或可替换的实施例也并非与其他实施例互斥。另外，描述了各种特征，其可有某些实施例而非其他实施例展现。类似地，描述了各种要求，其可能是某些实施例而非其他实施例的要求。

[0035] 之后的详细描述的某些部分以流程、步骤、逻辑块、处理、以及对计算机存储器内的数据比特的操作的其他符号表示的方式呈现。这些描述和表示是数据处理领域的技术人员用来最有效地将他们的工作传达给其他本领域技术人员的手段。此处的流程、计算机执行的步骤、逻辑块、处理等一般被看作为导致所希望的结果的步骤或指令的自洽序列。步骤是要求对物理量进行的物理操纵。通常，虽然不是必须的，这些量以计算机可读存储介质的电或磁信号的形式存在，并且能够在计算机系统中被存储、转移、组合、比较，以及否则操纵。主要出于通用原因，将这些信号称为比特、值、元素、符号、字符、术语、数字等有时已经被证实为很方便。

[0036] 但是，应牢记，所有这些和类似的术语要与合适的物理量相关联，并且仅仅是应用到这些量的方便的标签。除非特别声明，否则从以下的讨论中很显然，应理解，遍及本发明，利用诸如“处理”、或“访问”、或“写入”、或“存储”、或“复制”等之类的术语的讨论指计算机系统或类似电子计算设备的动作和处理，该电子计算设备操纵并将计算机系统的寄存器和存储器以及其他计算机可读介质中表示为物理(电子)量的数据转化成计算机系统存储器或寄存器或其他这样的信息存储、传输或显示设备中同样被表示为物理量的其他数据。

[0037] 本发明的实施例实现了一种线路速度互连结构，以用于要求极其低的延迟的应用中。存在许多联网操作影响这种应用，其无法忍受延迟。为了将进来的分组转发给不同的目的地，要求非常高速度的电路。以下的示图示出了这种高速度电路的若干实施例，并且示出了这种结构在联网环境切换分组中的使用，在管理至不同高速缓存线的访问以及至不同开放DRAM页的访问的处理环境中的使用，以及在任何合并和/或分割宏指令为对应的合并和/或分割微指令的处理环境中的使用。

[0038] 在许多实现中，元素以并行请求或分组进来，并且其经历一个或多个通用的动作。例如，它们中的两个或更多个可被组合、合并或分组，以形成统一/均一的请求或分组。其他

动作可将请求或分组分割成或分段成两个或更多个请求或分组。另一个示例是可变长度联网分组。相同的概念可被应用至固定大小的分组，其支持多个固定大小。

[0039] 以下的示图示出了这种线路速度互连结构的不同实施例。应当注意，术语“目的地”并不必意味着最终目的地。

[0040] 图1示出了根据本发明的一个实施例的线路速度互连结构的概略图。图1概略图示出了线路速度互连如何包括了两个阶段。

[0041] 在图1实施例中，第一阶段接收初始输入，并且在初始输入上执行预分类/分簇。第一阶段的输出被传递给第二阶段，其执行最终的重新映射。第二阶段的输出随后被传递至线路速度处。预分类阶段通过在将被检查以用于配对的初始输入中识别潜在的候选者来工作，例如利用目的地ID的第一匹配等等。第二阶段随后执行位置混排(shuffl ing)、配对或分割。线路速度通过能够并行组合并且分割输入以及创建结果输出来完成。

[0042] 图1的结构可被应用在多种不同的实现中，其将在后续示图中描述。实现包括联网体系架构、高速缓存访问体系架构、DRAM访问体系架构，以及仲裁体系架构，以及计算机指令组合/分割体系架构。遍及这些实现，此处图1中所图示的整体结构提供了将输入组合/分割成结果输出的线路速度。

[0043] 图2示出了根据本发明的一个实施例的线路速度互连结构的联网实现的概述。

[0044] 如以上所描述的，存在许多应用，其中输入请求、访问、数据、指令或分组的集合经历了将这些输入重新映射/复用或混排至不同的输出；对此一个非常常见的理由是，如果两个或更多个这些输入合并、组合或分组在一起，另一个原因是，当这些输入被分割、划分、分段或广播时，这两个原因可导致将输入完全重新映射至任意接口、互连的输出，导致对数据或请求或访问的混排、指令编码、路由或任意复用。在许多这些应用和实现中，线路速度/管线速度被维持有这种重新映射功能也是很重要的。

[0045] 图2示出了联网实现，其利用线路速度互连结构的低延迟优势，其中联网分组经历重新映射/组合/混排/分割处理，以接受进来的分组(示出为图2的左手边中的分组P0至P8)，并且将其处理成发出的分组(例如，示出为指向中央路由器的箭头0至8)。中央路由器随后将分组发送至其目的地，示出为Ra至Rd。

[0046] 因此，图2示出了一种处理，其中进来的分组的目的地和大小确定了它们如何被分割/组合/混排/重新排序。在一个实施例中，目标是将去往相同的下一个目的地(跨过相同的瞬态路径)的两个短分组配对。配对的理想条件将是这样的实例：两个分组都是短分组，并且它们去往相同的目的地。另外，在一个实施例中，目标是接受不同大小的进来的分组，并且执行分割/组合/混排/重新排序，以创建发送至中央路由器的统一大小的发出的分组。每个发出的分组示出为具有上半部和下半部(例如，偶数的或奇数的)，以示出多个短分组是如何被组合的。

[0047] 如图2中所示，P0去往目的地1并且是短分组，其被布置在发出的分组0中。下一个分组P1是中间大小的分组，并且去往不同于目的地P0的目的地。P1被排序成发出的分组1的上半部和下半部。P2和P3都是去往相同目的地的短分组，在该情形中为目的地3。因此，P2和P3被组合成发出的分组2。P4是大分组，并且示出了其中大分组被分割成两个或更多个发出的分组的示例，此处示出为P4被分割成占据发出的分组3和4。P5和P6都是去往相同目的地的短分组，并且因此被组合成发出的分组5。P7和P8是去往不同目的地的短分组，并且因此

不能被组合。因此，它们中的每一个被指派至其自身各自的发出的分组6和7。并且因此，该处理针对所接收的所有进来的分组继续执行。如图2中所图示的，该处理被并行实现，并且具有非常小的延迟(例如，线路速度)。

[0048] 图3示出了根据本发明的一个实施例的由线路速度互连结构所使用的处理的某些术语。在本实施例中，术语FB＝0指示短分组，术语FB＝1是指长分组，并且术语FB＝3指示非常长的分组。术语NP＝0指示分组具有相同的目的地。如上所述，在一个实施例中，目标是将去往相同下一目的地(跨越相同的瞬态路径)的两个短分组配对。配对的理想条件将是这样的实例：分组都是短分组，并且它们去往相同的目的地。因此，图3的表中示出了如何使用FB值和NP值，以快速地评估进来的分组是否可以通过它们自身配对、分割，或发送成发出的分组。通过这种方式，FB可被想作是桶/块/分组是否是完整的，并且NP可描述它们是否去往相同的目的地(例如，由虚线框301中的对所示出的)。图4中所示出的术语“sdx”指去往目的“x”的短分组，并且术语“ldx”指去往目的地“x”的长分组。

[0049] 图4示出了第二示图，该示图示出了根据本发明的一个实施例，如何使用FB值和NP值。图4示出了其中使用了评估方程的用于处理所有进来的分组并确定将进来的分组分割/组合/重新排序成发出的分组的方式。另外，并行执行该评估。

[0050] 如以上所描述的，目标是将去往相同下一目的地(例如，跨越相同的瞬态路径)的两个短分组配对。针对两个分组的条件应当是FB＝0(例如，二者都是短分组)，并且后一个分组应当具有NP＝0(例如，后一个分组具有与更早的分组相同的目的地，并且因此可被配对)。评估方程可被写作M5＝G5+OR(MOD(G5,2)I5)*(MOD(E5+MOD(SUM(M$2:M4),2),2))，其中MOD指模2运算。这在框401中示出，其示出了用于评估元素E5的新位置的示例。框402示出了用于评估元素E6的新位置的另一示例。6上面的星号指示6是分割的第一部分的位置。分割的第二部分的位置通过将分割的第一部分的位置加2而本地生成。

[0051] 以上方程中的OR试图找出NP或FB是否被设置。如果其中一个被设置，则评估处理将创建气泡(bubble)，并且该气泡将处于上半部分或下半部分(例如，偶数的或奇数的)的位置处。方程将当前位置与当前位置之前的所有气泡累积相加。方程的操作示出在图4中所描绘的表中，并且还由图4的下半部分示出，其中进来的分组I0至I7被描绘为它们如何被处理成发出的分组0至7。

[0052] 图5示出了根据本发明的一个实施例的示出了初始位置/分组号码、新的分组中的新的位置号码，以及结果输出分组号码之间的关系的另一示图。进来的分组可以是短的，其被表示为“S”，长的，其被表示为“L”，或非常长的，其被表示为“VL”。在本示例中，目的地都是相同的，被表示为术语“D1”。因此，图5是对初始分组位置如何被变成结果发出分组的新的分组位置的阐释。某些分组被组合(例如，分组0和分组1)，并且，某些分组被分割(例如，分组2和分组6)。具体地，图5示出了非常大的分组(例如，分组6)如何被分割以占据输出分组。对初始分组的处理用上述模2评估方程并行执行。

[0053] 图6示出了根据本发明的一个实施例的上述模2评估方程的操作的更加详细的示图。

[0054] 如上所述，在许多实现中，元素作为并行请求或分组进来，并且其经历一个或多个通用的动作。它们中的两个或更多个可被组合、合并或分组，以形成统一/均一的请求或分组。其他动作可以是将请求或分组分割成或分段成两个或更多个请求或分组。第一个示例

是可变长度联网分组。相同的概念可被应用于固定大小的分组，其中支持多个固定大小。并且另外，该评估的之后的分组应当具有NP＝0(之后的分组具有与之前的分组相同的目的地，并且因此可被配对)。

[0055] 在许多实现中，元素作为并行请求或分组进来，并且其经历一个或多个通用的动作。它们中的两个或更多个可被组合、合并或分组，以形成统一/均一的请求或分组。其他动作可以是将请求或分组分割成或分段成两个或更多个请求或分组。第一个示例是可变长度联网分组。相同的概念可被应用于固定大小的分组，其中支持多个固定大小。

[0056] 图6实施例描述了路由体系架构实现的操作，其中可变大小的初始分组被重新定位并布置成新的发出的统一大小的分组。在本实现中，发出的分组具有偶数位置和奇数位置。根据上述模2函数，可变大小的初始分组被布置到偶数位置和奇数位置。在本实施例中，“NP”指示符当针对初始分组被设置为0时，指示这两个分组可被组合或合并在一起，因为它们去往相同的中间/瞬态目的地。当“NP”被设置为1时，则它们不能被组合或合并，因为它们去往不同的目的地。通过相应地调整取模函数，针对分组多于两个，分组/组合可被促进。

[0057] 但是，当FB指示符被设置为1时，大的分组需要被分段成两个更小的分组(分组要求完整的块/桶:统一的奇数/偶数槽位(s lot))。另一个FB被设置为1的理由是，每当共享相同中间/瞬态目的地的初始合并的两个分组需要被分割成带有不同的最终目的地的两个不同的分组时。FB可被设置为3，如果分组需要被分割成4片。

[0058] 左侧的图6的示图示出了初始分组号/位置、值FB和NP，以及目的地。图6的右侧，示出了跨越位置的气泡以及新的位置。公式首先计算由气泡所导致的从初始位置的累积布置。然后，通过将初始位置与累积布置的累积和相加来计算新的位置。该累积和属性由虚线椭圆示出，其示出了每个后续位置是如何由所有之前的位置的累积和所确定的。另外，虚线箭头示出了初始位置和NP值如何作为因子转化为评估方程。

[0059] 图7描述了根据本发明的一个实施例的高速缓存访问实现的操作，其中高速缓存的访问被重新定位并布置到针对存储器层级的高速缓存线的大小对齐的新的发出的高速缓存访问中。

[0060] 如上所述，在许多实现中，元素作为并行请求进来，并且其经历一个或多个处理动作。在一个情形中，两个或更多个请求可被组合、合并或分组，以形成统一/均一的请求或分组。其他情形可以是将请求分割成或分段成两个或更多个请求。

[0061] 图7实施例描述了存储器/高速缓存体系架构，其中在存在试图访问其他高速缓存线的其他存储器请求的情况下，不同的存储器请求访问相同的高速缓存线。在该情形中，希望合并访问相同高速缓存线的两个或更多个请求。

[0062] 例如，请求可被重新排序，使得至相同高速缓存线的请求被合并，以便只有一个请求争取该高速缓存线，并且，该高速缓存线恢复。不存在对相同高速缓存线的多个请求和多个返回。非对准的存储器请求(例如，请求跨越两个不同高速缓存线的数据)是分割的示例，其中该请求被分割成针对包括非对准的请求数据的两个不同的高速缓存线的两个请求。

[0063] 在本实施例中，访问相同的高速缓存线的两个请求被组合成相同的统一的请求(奇数-偶数槽位)。根据上述模2评估函数，初始存储器请求被布置到偶数和奇数位置中。在本实施例中，“NP”指示符当针对初始请求被设置为0时，并且指示这两个请求可被组合或合并在一起，因为它们正访问相同的高速缓存线。当“NP”被设置为1时，则它们不能被组合或

合并在一起，因为它们正访问不同的高速缓存线。通过相应地调整取模函数，针对分组多于两个，分组/组合可被促进。

[0064] 但是，当FB指示符被设置为1时，非对准的存储器请求需要被分割成两个不同的高速缓存线访问。FB可被设置为3，如果访问需要被分割成4个高速缓存线访问，例如要求访问多于一个高速缓存线的特殊的存储器请求(例如，字符串、缓冲器复制、I/O请求等等)。如上所述，评估公式首先计算由气泡所导致的从初始位置的累积布置。然后，通过将初始位置与累积布置的累积和相加来计算新的位置。

[0065] 应当注意，在本存储器/高速缓存实现(不同于联网情况)中，当两个高速缓存线请求被组合时，它们变为单个请求，并且它们物理上不占据块/桶的奇数/偶数槽位。但是，奇数和偶数槽位代表高速缓存线中的两个不同的地址。

[0066] 应当注意，在本实现中，每个奇数偶数槽位代表可被独立承载到至高速缓存系统/存储器系统的不同的端口或总线的访问。

[0067] 另外，应当注意，在本存储器/高速缓存实现中，气泡的概念是无法利用偶数槽位和奇数槽位二者来访问给定的高速缓存线。

[0068] 在另一实现中，由图7所描述的相同的存储器访问概念可被应用于DRAM控制器，其中多个请求被组合在一起，以访问存储器系统中的相同的开放的DRAM页。在这种实施例中，DRAM页被看作是类似于存储器层级的高速缓存线。该类似特别应用于DRAM页被打开或关闭的方式。在本实现中，至DRAM的请求被重新排序，使得至相同DRAM页的请求被移动到一起，使得当DRAM页打开时，它们可以访问该DRAM页。与当DRAM页关闭时访问DRAM页相比，当该页是打开时，至DRAM的请求更快。打开关闭的DRAM页花费一定量的延迟。因此，请求被重新排序至相同的总线/端口/信道/DRAM存储器条，以获得对相同DRAM条/DRAM页的多个访问的益处。

[0069] 图8示出了根据本发明的一个实施例的、根据高速缓存端口的重新映射处理的示图。图8的左手边示出了一系列初始高速缓存访问请求，示出为“A”至“F”。本发明的评估处理对初始高速缓存请求执行预分簇/过滤/混排处理。图8中所图示的Mrs作为初始位置，其随后经历了至新位置的映射。这些新位置对应于高速缓存存储器的端口或条。图8示出了上述相同的结构可如何利用NP值来确定访问是否应当被组合(例如，LD(A)和LD(B))，并且可利用FB值来确定访问是否应当被分割(例如，LD(F)，其需要访问两个不同的场地(court)/条)。在非对准的访问的情形中，其请求跨越多于一个高速缓存线的数据，访问一般被分割。

[0070] 应当注意，在一个实施例中，应当实现这样的端口：其中整个高速缓存是一个统一的多端口条，或者高速缓存可被分割，其中高速缓存线被跨越多个条分割，使得每个条具有一个或一些端口，在这种情形中，经配对的访问被检查，以去往高速缓存线的相同条。

[0071] 图9示出了根据本发明的一个实施例的、根据高速缓存端口的两阶段重新映射处理的示图。例如，虽然图8仅描绘了单个阶段，但是，图9，出于阐释的目的，描绘了根据本发明的一个实施例的两个阶段的处理。如之前对图1的讨论所描述的，通过两个阶段的处理，第一阶段执行对初始输入的预分簇/移位/混排/分组，在该情形中，是对高速缓存的访问。第二阶段执行对高速缓存的访问的最终映射。

[0072] 在图9的实施例中，为了找到去往相同的高速缓存线的载入请求，使用比较逻辑来扫描进来的载入请求组并找到第一匹配。例如，当LD(A)被检查时，比较逻辑找到去往相同

高速缓存线的LD(D)。这导致LD(D)被混排或实质上被关联至靠近LD(A)。相同的事发生在LD(B)上，其中比较逻辑找到LD(E)。如所示，这可被实现为比较并找到第一匹配。

[0073] 随着载入请求被混排和重新排序，如果不存在将要转移至第二阶段的对应的载入请求，则预分簇阶段中的初始位置和结果位置可被无效。从预分簇阶段到重新映射阶段的载入请求的表示可被称为虚拟请求。

[0074] 当访问相同的高速缓存线(例如，NP＝0)时，重新映射的结果包括多个合并的载入请求。重新映射的不同结果包括非对准的访问分割(例如，FB＝3)，其中非对准的访问被分割成两个高速缓存线访问。

[0075] 一旦重新排序的虚拟请求由第二阶段接收，它们被重新映射(例如，根据NP变量或FB变量)成统一合并的发出的载入请求，其中某些虚拟请求已经被组合并且某些虚拟请求已经被分割。

[0076] 应当注意，对最终映射阶段的实现是与之前的示图中所描述的示例相同的。如所示，无效的虚拟请求被自动删除，或者否则被过滤掉。值得注意的是，在无效请求已经被过滤掉之后，NP位被设置并检查。

[0077] 图10描述了根据本发明的一个实施例的计算机指令实现的操作，其中计算机指令被重新定位并融合或分成新的发出的计算机指令。图10示出了以上示图中的结构脚本如何被用于计算机指令实现中，其中进来的指令，指令0至指令8，可被融合或分割，以创建新的发出的指令。通过并行将指令融合和分割成最终的指令/微指令，这被在图10中示出。块实现了对指令的并行打包/分割，诸如操作融合或操作分割成微操作。以类似于以上描述的实施例的方式，NP值可确定哪些指令被融合，并且FB值可确定哪些指令被分割。以类似于以上描述的实施例的方式，结果指令具有上半部分和下半部分(例如，或者奇数的/偶数的)。

[0078] 图11描述了根据本发明的一个实施例的仲裁器实现的操作，其中不同的源针对多个发出的目的地被仲裁。图11示出了一种实现，其中以上实施例中所描述的结构被用于执行去往多个目的地的不同源之间的仲裁。例如，来自多个源的某些请求可被组合至相同的目的地。某些其他请求可被广播至多个目的地。以类似于以上描述的实施例的方式，NP值可确定哪些请求被组合，并且FB值可确定哪些请求被分割。

[0079] 例如，在一个实施例中，NP＝0导致合并操作(组合)。这通常在分解(paring)/唤醒通信(例如，掉电/睡眠，重置)中看到。一个示例是打印机或其他类型的外围设备。FB＝1导致分割操作(例如，分割成两个或更多个)。这通常在广播通信中看到。

[0080] 在图11的实施例中，NP＝0导致合并操作(组合)。这通常在分解/唤醒通信中看到。一个示例是到打印机或其他类型的外围设备的请求。在本实施例中，FB＝1导致分割操作(例如，分割成两个或更多个)。这通常在广播通信或掉电/睡眠、重置中看到。

[0081] 图12示出了根据本发明的一个实施例的仲裁器实现的另一示图，其中不同的源针对多个发出的目的地被仲裁。图12示图示出了明确的两个阶段的实现。第一阶段以上述方式进行预分类和预分簇。第二阶段以上述方式执行分割中的最终分类组合。在图12的实施例中，如所示，第一阶段和第二阶段之间所示出的输入是具有不同带宽要求的广播流，这些广播流可被组合成单个高速度输出广播，例如，其实现了更快的时间复用或频率复用输出。

[0082] 图13示出了根据本发明的一个实施例的仲裁器实现的另一示图，其中不同的源针对多个发出的目的地被仲裁。在图13的示出中，输出被视为不同的更快的接口或总线或信

道。图13的左侧的输入可包括更低的频率/更低的带宽输入。如所示，仲裁器通过将这些输入组合或分割成带有更高频率的更高额的输出来工作。例如通过利用时间复用或频率复用，输出可具有更高的带宽。输入包括利用输出带宽的流。仲裁器利用频率和/或时间复用来在输入流中间进行仲裁，以创建结果的输出流。配对暗示着请求被连续排序，使得请求获得最佳的DRAM访问时间。图13的右手边示出了不同的目的地。例如，某些目的地是去往许多不同最终目的地的广播站。其他目的地是单点目的地。

[0083] 图14示出了根据本发明的一个实施例的对示出了模2评估方程的电子表格的图形描绘。电子表格的上部分示出了串行模式评估处理(例如，串行模式FB/NP)。电子表格的下部分示出了并行模式评估处理(例如，并行模式FB/NP)。应当注意，电子表格示出了图6中所描述的评估处理期间所示出的方程。公式首先计算由气泡所导致的从初始位置的累积布置。然后，通过将初始位置与累积布置的累积和相加来计算新的位置。在串行处理中，如图14的上部分所示，该累积和出现在逐周期的处理中。在并行处理中，如图6的描述中所描述的，累积和被并行计算，如图14的下部分所示。

[0084] 图15示出了根据本发明的一个实施例的对累积和评估处理的操作。图15示出了模2项方程可以如何通过展开递归项并归零重复项来简化的示例。这利用了模2运算的属性。图14的前三行示出了项M2、M3和M4。图14更加详细地示出了这些项。此处图15示出了从图14复制的取模项。由于取模项被展开，重复项可被归零。这在图15的第四行示出了，其中M4的重复项被归零。该属性导致需要递归求和的项的数量的减少，因此简化了实现该公式的电路的并行化。

[0085] 图16示出了根据本发明的一个实施例的、描绘了用于并行执行累积和评估的电路的示图。图16示出了如何利用AND门来实现相乘项，以及如何利用XOR门来实现态度项(attitude term)。因此，图16示出了执行模2相加和相乘二者的逻辑电路。

[0086] 该示图示出了阶段优化高速度重新映射逻辑结构，其被用于解决单个时钟周期中所描绘的取模函数。该重新映射逻辑特别适于任何其中多个一比特输入需要快速相加以产生一比特或两比特输出的应用。重新映射逻辑结构的目的是并行并以线路速度对遇到的分组进行映射或打包或分割。为了这样做，结构利用两个观察的优势。第一观察是当如示图中所示的展开递归和方程时，重复和元素将在模2运算下归零。第二观察是示图中所示的递归和方程中的相加元素和相乘元素如何在模2算数下表现。在这种条件下，相加元素变成XOR函数，并且相乘元素变成AND函数。这允许递归和方程如示图中所示的被映射到逻辑电路。括号内的元素是乘法，并且因此由AND门操作。括号外的元素是加法，并且因此由XOR门操作。现在，方程变为在空间上完全展开，而非被串行化。

[0087] 为了找到重新映射的新位置，这些逻辑函数的输出被用于馈送并行加法器，并且该并行加法器产生虚拟请求至新位置的最终重新映射。通过这种方式，电路执行上述两个动作，其中两个或更多个那些输入(例如，虚拟请求)可被组合、合并或分组，以形成合并的请求。另一动作可以是将虚拟请求或分组分割或分段成两个或更多个请求。

[0088] 通过这种方式，电路进行串行化相加处理，其中每个虚拟请求的布置依赖于每个在先的虚拟请求的布置，并且并行实现模2函数，以产生重新映射的输出请求。

[0089] 图17示出了根据本发明的一个实施例的执行对累积和的评估的电路的示图。图17示出了评估方程项是如何被评估电路的硬件组件所处理的。该示图示出了阶段优化高速度

加法器电路(例如,并行加法器1700),其被用来解决单个时钟周期中所描绘的取模函数。该电路特别适于任何其中多个一比特输入需要快速相加以产生一比特或两比特输出的应用。在以下的图21中进一步描述了并行加法器1700的细节。

[0090] 如上所述,线路速度对这种无法忍受延迟的应用具有重大影响。一个示例来自联网应用,其要求将进来的分组转发至不同的目的地,因此要求非常高速度的电路。以上的示图示出了这种高速度电路的若干实施例,并且示出了这种结构在交换/路由分组的联网环境中的应用,在管理至不同高速缓存线的访问以及至不同打开的DRAM页的访问的处理环境中的应用,以及任何将宏指令合并/融合以及分割/解码成对应的合并和/或分割的微指令的处理环境中的应用。

[0091] 另外,如上所述,许多实现、输入元素作为并行请求或分组进来,并且它们经历一个或两个通用的动作。两个或更多个那些输入可被组合、合并或分组,以形成统一/均一的请求或分组。另一动作可以是将请求或分组分割或分段成两个或更多个请求或分组。第一示例是可变长度联网分组。相同的概念可被应用于固定大小的分组,其中支持多个固定大小。

[0092] 执行这种功能的图17电路可被构建成串行化取模和(例如,其示出为模2功能,如所给出的合并2个元素以及分割成2个或4个的示例),但是,应当注意,该功能也可以层级的方式应用,用于在多个步骤中合并多于2个的情形。通过这种方式,图17示出了新颖的惩罚减少机制,其中许多电路组件被减少成并行的AND-XOR门,其具有与其相关联的并行加法器,以获得线路速度。

[0093] 图18示出了根据本发明的一个实施例的执行对累积和的评估的第二电路的示图。图18与图17类似,但是,图18示出了与并行一比特加法器一起使用的两比特加法器。图18的示图示出了用于采用初始位置并从其计算最终位置的方程是如何并行计算的。如上所述,方程的元素可被分解成相乘项和相加项。如所示,相乘项由AND门解决。如所示,相加项由XOR门解决。图18示出了解决针对初始位置5(例如,n(5))的方程的电路的示例。每个初始位置(例如,在本示例中初始位置1至10)将具有对应的方程,该对应的方程具有类似于图18中所示的对应的电路。这些方程中的每一个的结果是高速度并行加法器的输入,诸如加法器1700。

[0094] 图19示出了并行加法器实现的示例性体系架构。图19示出了如何利用4:2进位保存加法器来实现并行加法器。

[0095] 图20示出了根据本发明的一个实施例的、描绘了并行进位保存加法器的示图。如图20中所示,传统的并行4:2进位保存加法器被用于执行对并行输入的求和,诸如处理器调度中的就绪位,其被用于选择分派的指令,或用于将选择用于分派的指令映射到对应的分派端口,或例如被用于对有效位求和,以用于计数有效的稀疏条目并向其指派分配写入端口。

[0096] 图21示出了根据本发明的一个实施例的阶段优化高速度并行加法器的实施例。图21的实施例可被用于实现图17中所示的并行加法器1700。图21的上部分示出了输入的到达时间,并且还示出了三个4:1加法器和一个2:1加法器。图21的下部分示出了如何实现4:1加法器。图21的中间部分示出了使用两个耦合的3:2进位保存加法器以接收其上的三个4:1加法器和一个2:1加法器的输出。与传统的并行加法器相比,该布置省去了一个4:2进位保存

加法器阶段，因此使得图21的实施例更快。该实现利用了并行输入是每个一位输入的事实，这允许电路相应地被优化。

[0097] 在图21中，加法器可被用于执行对一位输入的求和，诸如处理器调度中的就绪位，其被用于选择分派的指令，或用于将选择用于分派的指令映射到对应的分派端口，或例如被用于对有效位求和，以用于计数有效的稀疏条目并向其指派分配写入端口。

[0098] 前述描述，出于说明的目的，已经参照具体实施例进行了描述。但是，所示出的以上讨论并不意欲穷尽式的或将发明限制到所公开的确切的形式。鉴于以上教导，许多修改例和变化例是可能的。实施例被选择并描述，以便最好地说明发明的原理及其实际应用，因此使得本领域技术人员能够最好地利用本发明和各种实施例，该实施例带有各种修改例，如可适于所构想的特定使用。

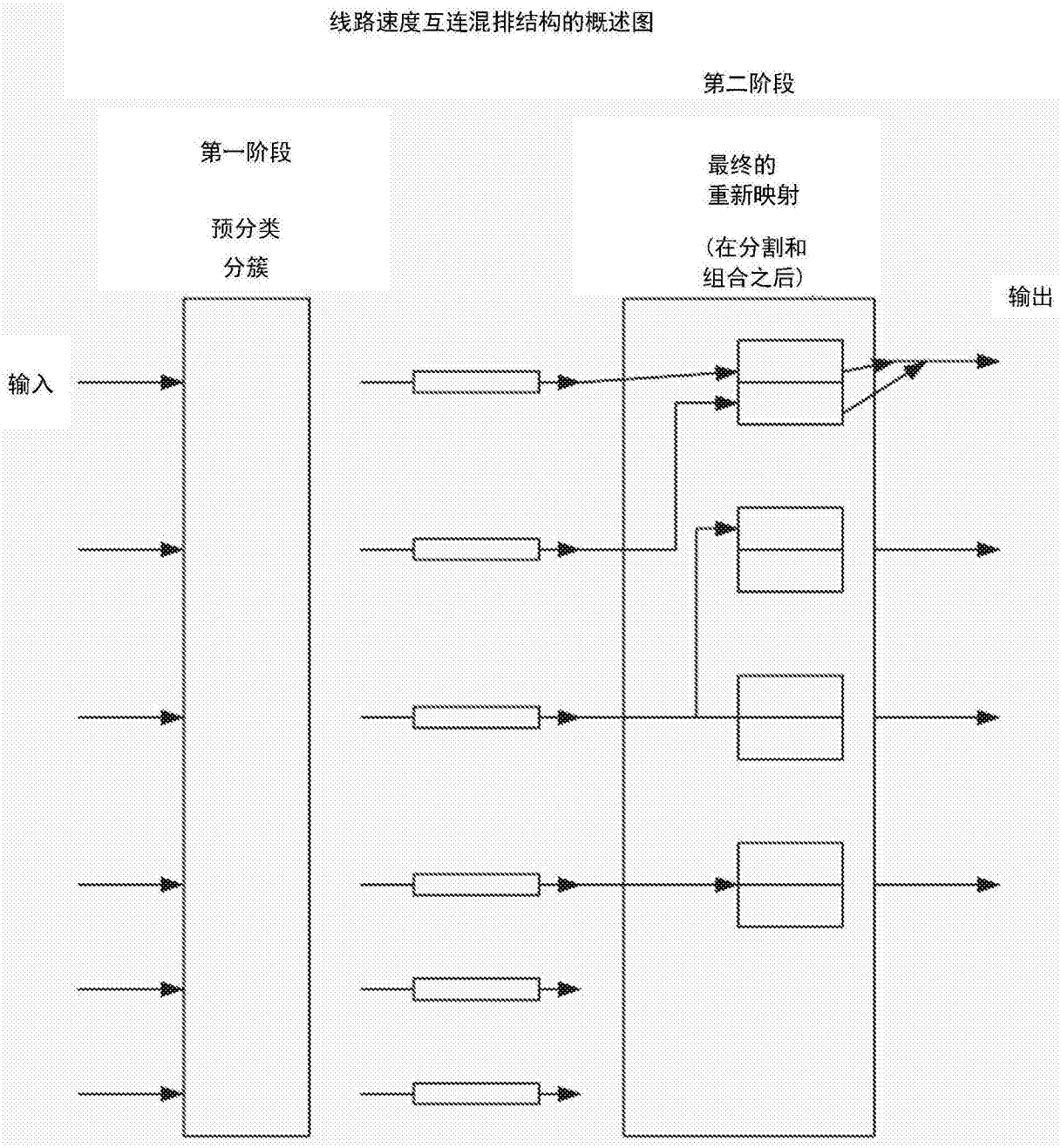


图1

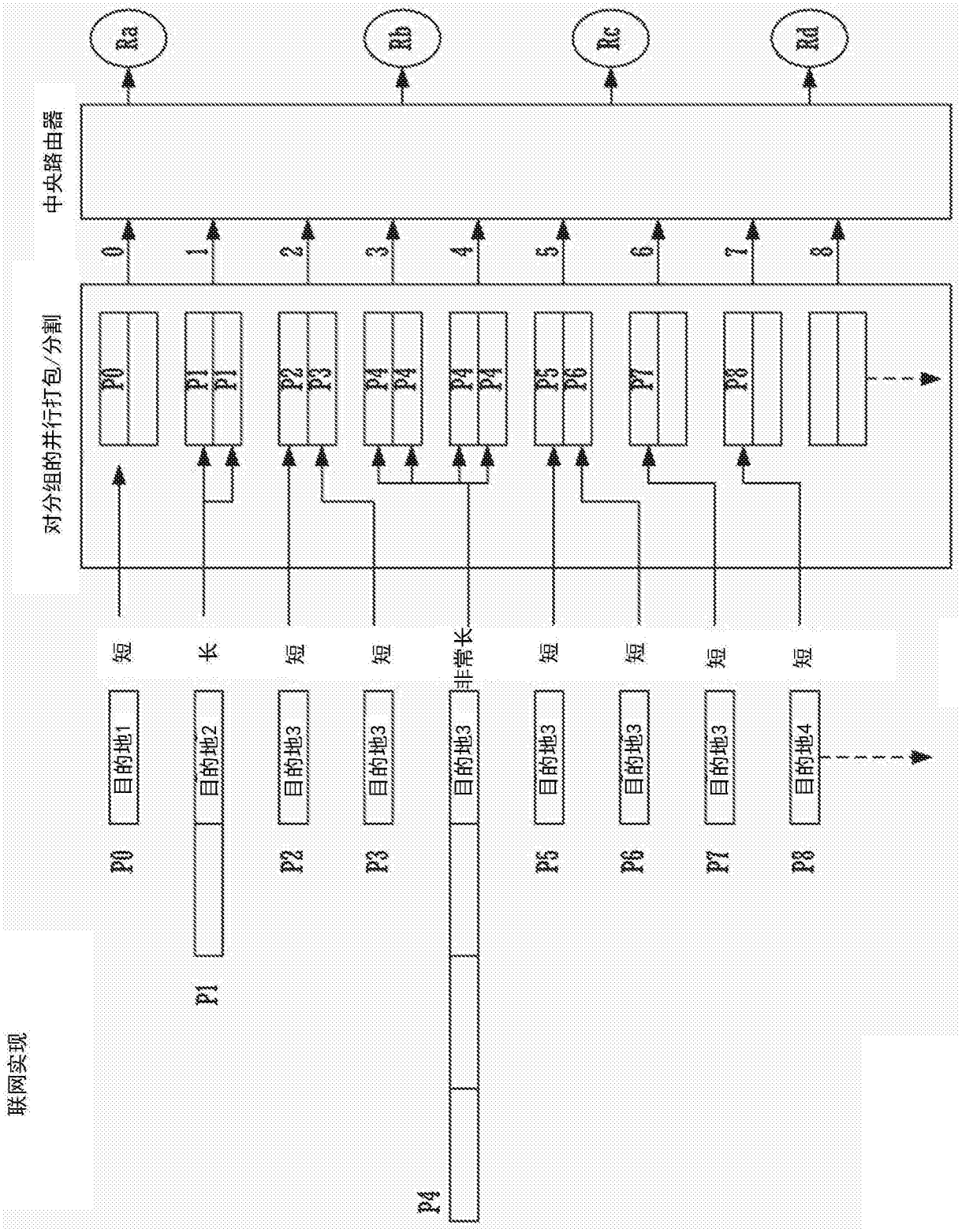

Ra
Rb
Rc
Rd
中央路由器
0
1
2
3
4
5
6
7
8
对分组的并行打包/分割
P0
P1
P1
P2
P3
P4
P4
P4
P4
P5
P6
P7
P8
短
长
短
短
非常长
短
短
短
短
目的地1
目的地2
目的地3
目的地3
目的地3
目的地3
目的地3
目的地3
目的地4
P0
P1
P2
P3
P4
P5
P6
P7
P8
联网实现

图2

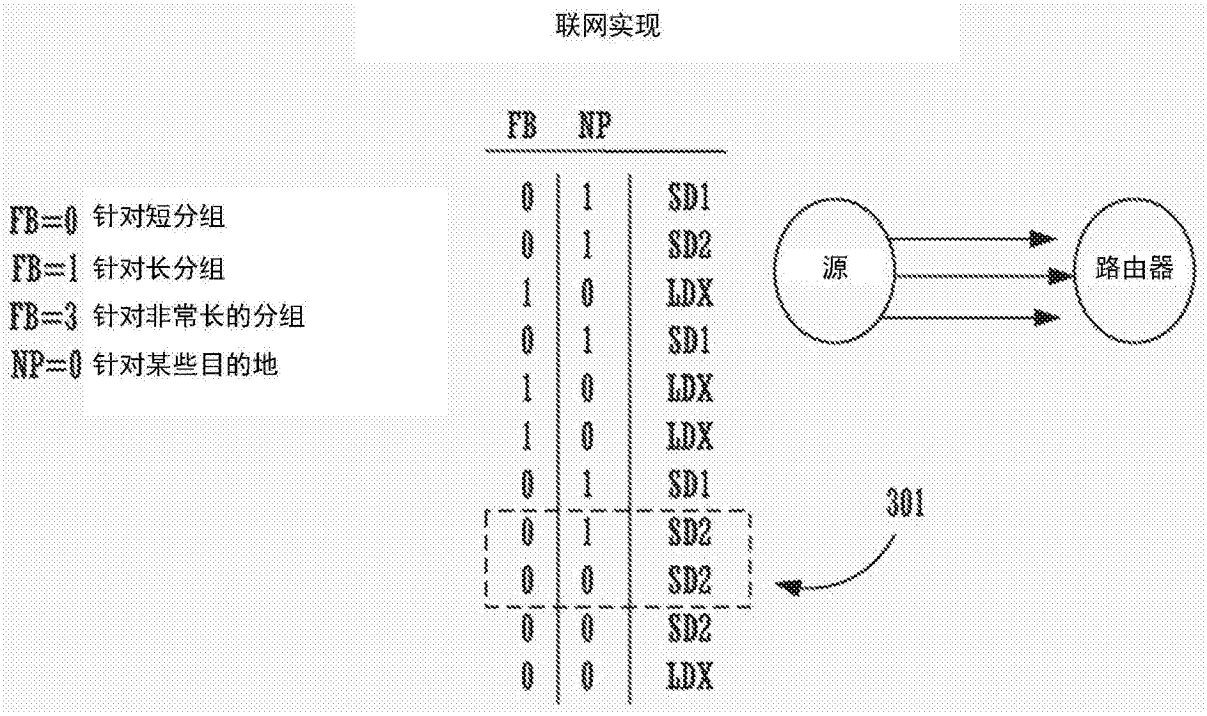

联网实现
FB=0 针对短分组
FB=1 针对长分组
FB=3 针对非常长的分组
NP=0 针对某些目的地
FB NP
0 1 SD1
0 1 SD2
1 0 LDX
0 1 SD1
1 0 LDX
1 0 LDX
0 1 SD1
0 1 SD2
0 0 SD2
0 0 SD2
0 0 LDX
源
路由器
301

图3

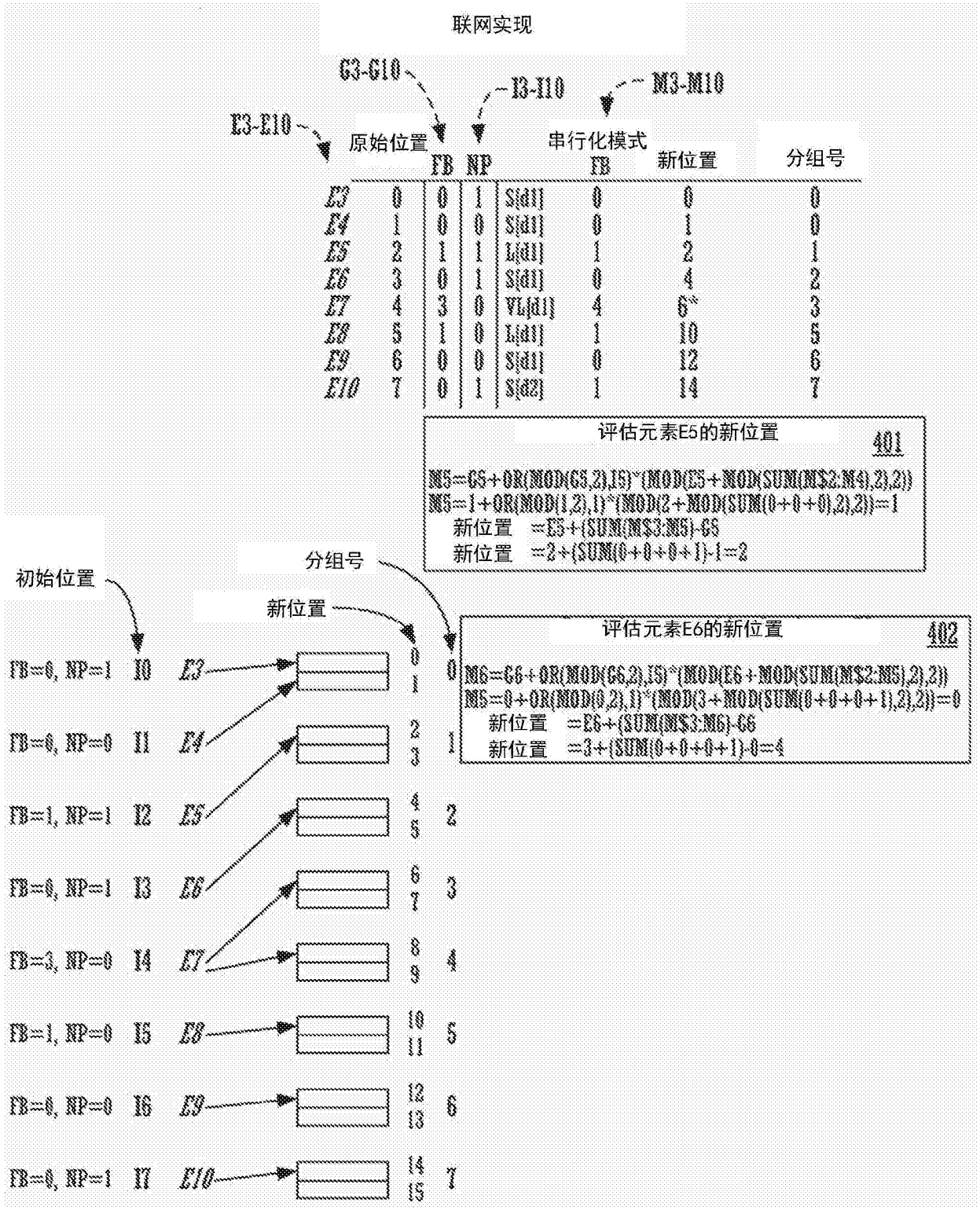

联网实现
G3-G10
I3-I10
M3-M10
E3-E10
原始位置
串行化模式
FB NP FB 新位置 分组号
E3 0 0 1 S[d1] 0 0 0
E4 1 0 0 S[d1] 0 1 0
E5 2 1 1 L[d1] 1 2 1
E6 3 0 1 S[d1] 0 4 2
E7 4 3 0 VL[d1] 4 6* 3
E8 5 1 0 L[d1] 1 10 5
E9 6 0 0 S[d1] 0 12 6
E10 7 0 1 S[d2] 1 14 7
评估元素E5的新位置
401
M5=G5+OR(MOD(G5,2),I5)*(MOD(G5+MOD(SUM(M$2:M4),2),2))
M5=1+OR(MOD(1,2),1)*(MOD(2+MOD(SUM(0+0+0),2),2))=1
新位置 =E5+(SUM(M$3:M5)-G5
新位置 =2+(SUM(0+0+0+1)-1=2
初始位置
分组号
新位置
评估元素E6的新位置
402
M6=G6+OR(MOD(G6,2),I5)*(MOD(E6+MOD(SUM(M$2:M5),2),2))
M6=0+OR(MOD(0,2),1)*(MOD(3+MOD(SUM(0+0+0+1),2),2))=0
新位置 =E6+(SUM(M$3:M6)-G6
新位置 =3+(SUM(0+0+0+1)-0=4
FB=0, NP=1 I0 E3
FB=0, NP=0 I1 E4
FB=1, NP=1 I2 E5
FB=0, NP=1 I3 E6
FB=3, NP=0 I4 E7
FB=1, NP=0 I5 E8
FB=0, NP=0 I6 E9
FB=0, NP=1 I7 E10
0 1 0
2 3 1
4 5 2
6 7 3
8 9 4
10 11 5
12 13 6
14 15 7

图4

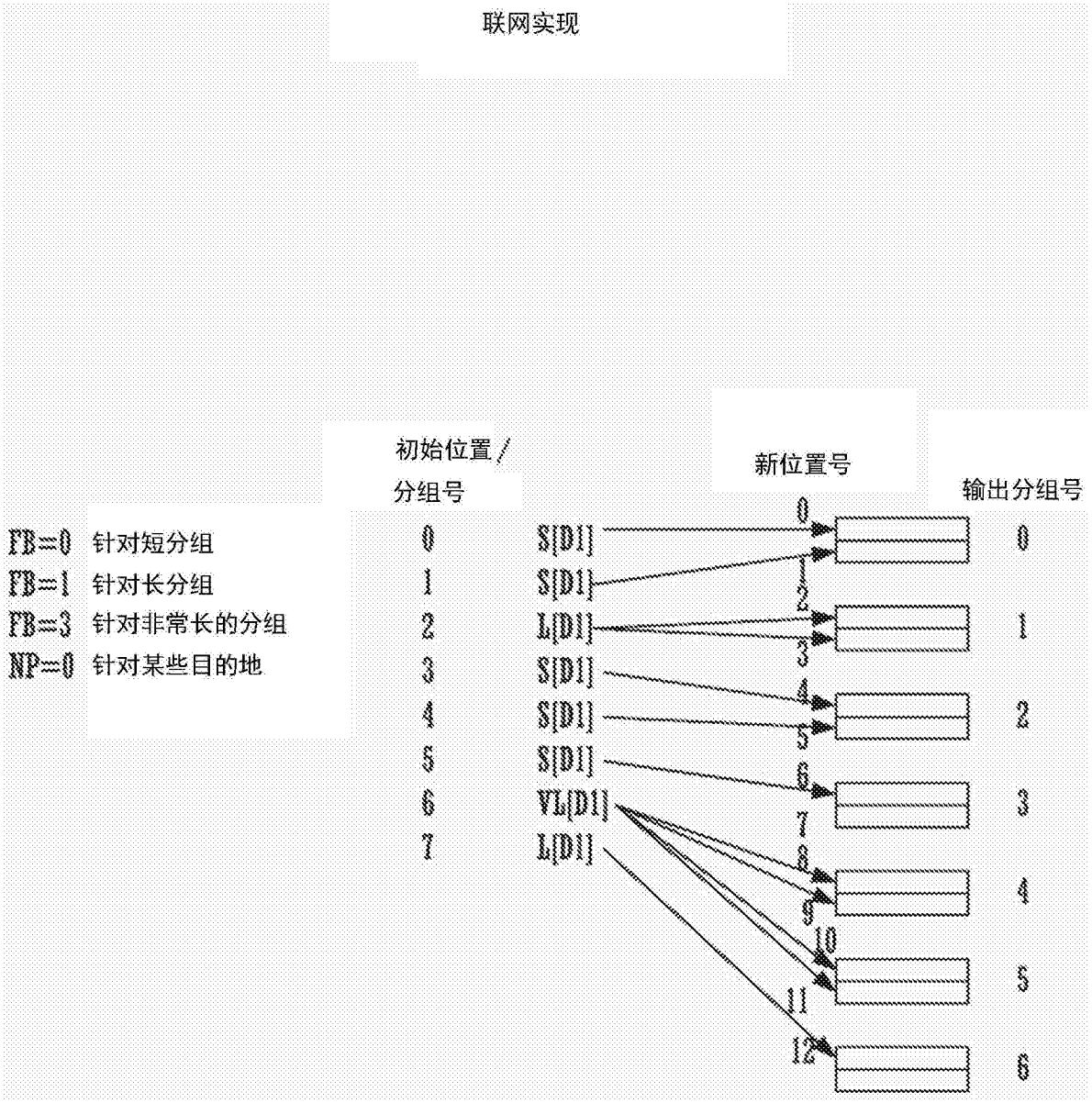

联网实现
初始位置/
分组号
新位置号
输出分组号
FB=0 针对短分组
FB=1 针对长分组
FB=3 针对非常长的分组
NP=0 针对某些目的地
0 S[D1]
1 S[D1]
2 L[D1]
3 S[D1]
4 S[D1]
5 S[D1]
6 VL[D1]
7 L[D1]
0
1
2
3
4
5
6
7
8
9
10
11
12
0
1
2
3
4
5
6

图5

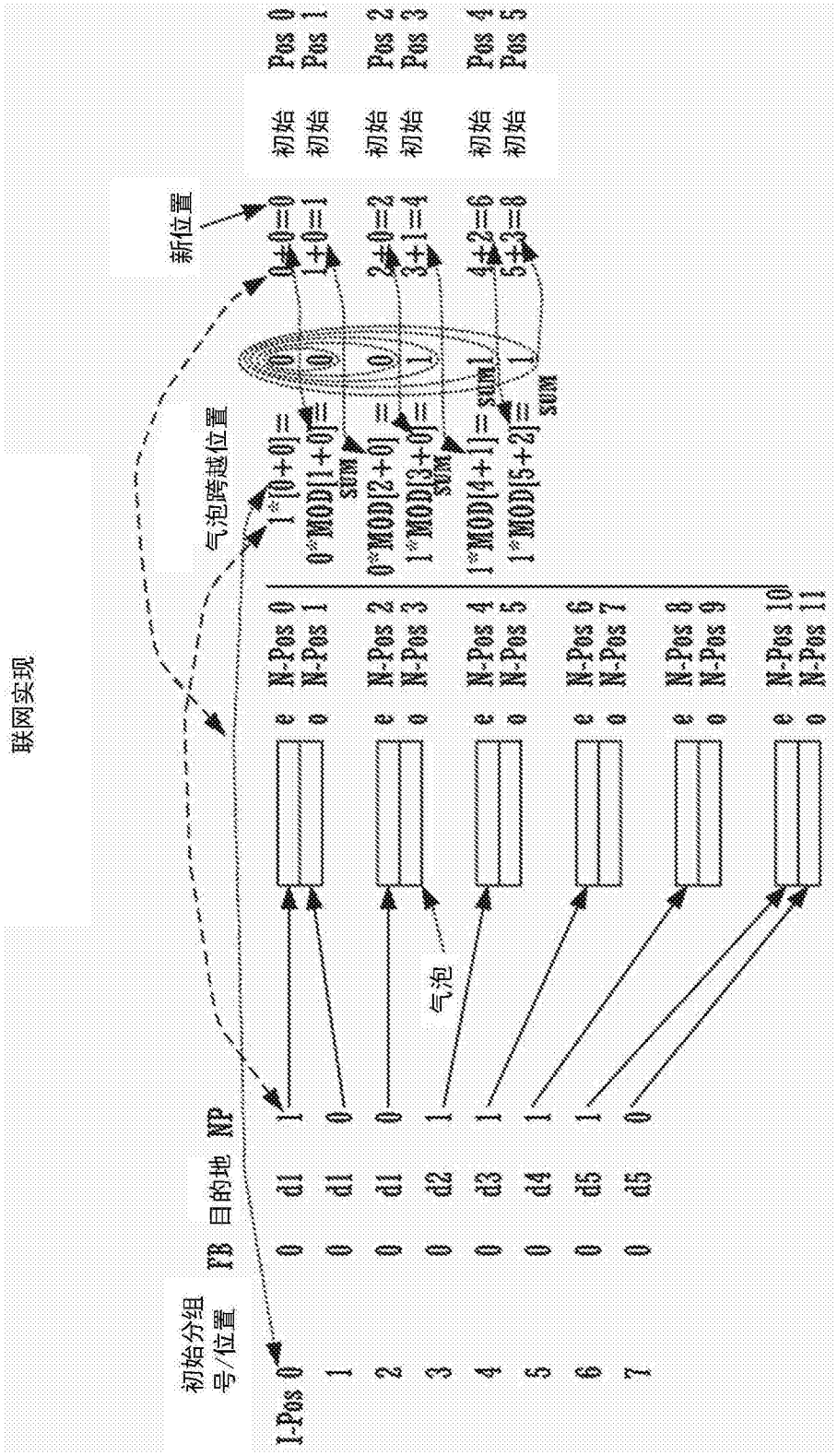

联网实现
初始分组号/位置
I-Pos
FB
目的地
NP
新位置
气泡跨越位置
气泡
初始
Pos 0
Pos 1
Pos 2
Pos 3
Pos 4
Pos 5
N-Pos 0
N-Pos 1
N-Pos 2
N-Pos 3
N-Pos 4
N-Pos 5
N-Pos 6
N-Pos 7
N-Pos 8
N-Pos 9
N-Pos 10
N-Pos 11
sum

图6

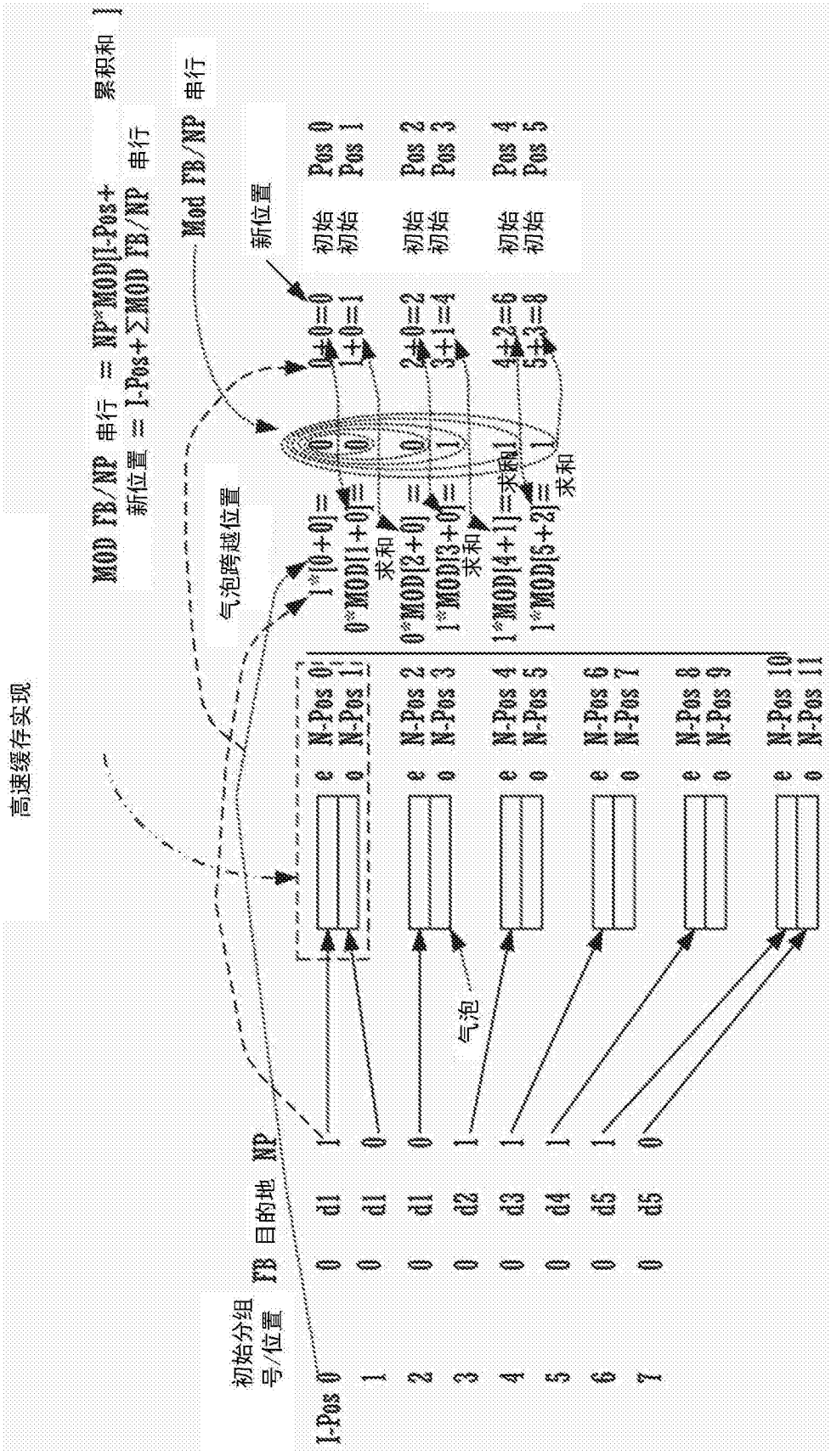

高速缓存实现
初始分组号/位置
FB 目的地 MP
l-Pos
气泡
气泡跨越位置
新位置
求和
Mod FB/MP 串行
MOD FB/MP 串行 = MP*MOD[l-Pos+
新位置 = l-Pos+ΣMOD FB/MP 串行
累积和]
N-Pos 0
N-Pos 1
N-Pos 2
N-Pos 3
N-Pos 4
N-Pos 5
N-Pos 6
N-Pos 7
N-Pos 8
N-Pos 9
N-Pos 10
N-Pos 11
Pos 0
Pos 1
Pos 2
Pos 3
Pos 4
Pos 5
初始
初始

图7

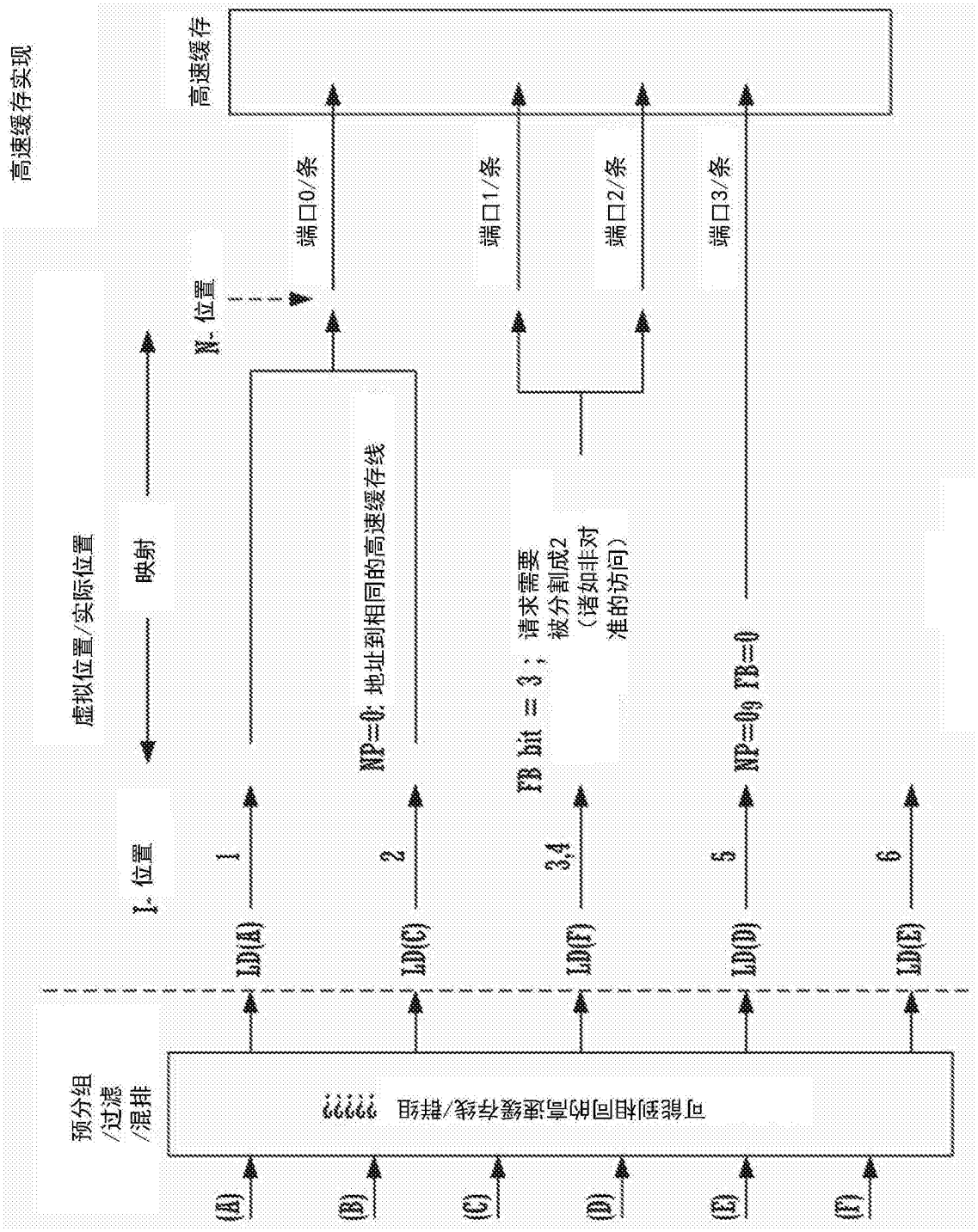

高速缓存实现
虚拟位置/实际位置
预分组/过滤/混排
可能到相同的高速缓存线/群组 ?????
(A)
(B)
(C)
(D)
(E)
(F)
LD(A)
LD(C)
LD(F)
LD(D)
LD(E)
1. 位置
映射
1
2
3,4
5
6
WP=0; 地址到相同的高速缓存线
FB bit = 3；请求需要被分割成2（诸如非对准的访问）
WP=0; FB=0
X. 位置
端口0/条
端口1/条
端口2/条
端口3/条
高速缓存

图8

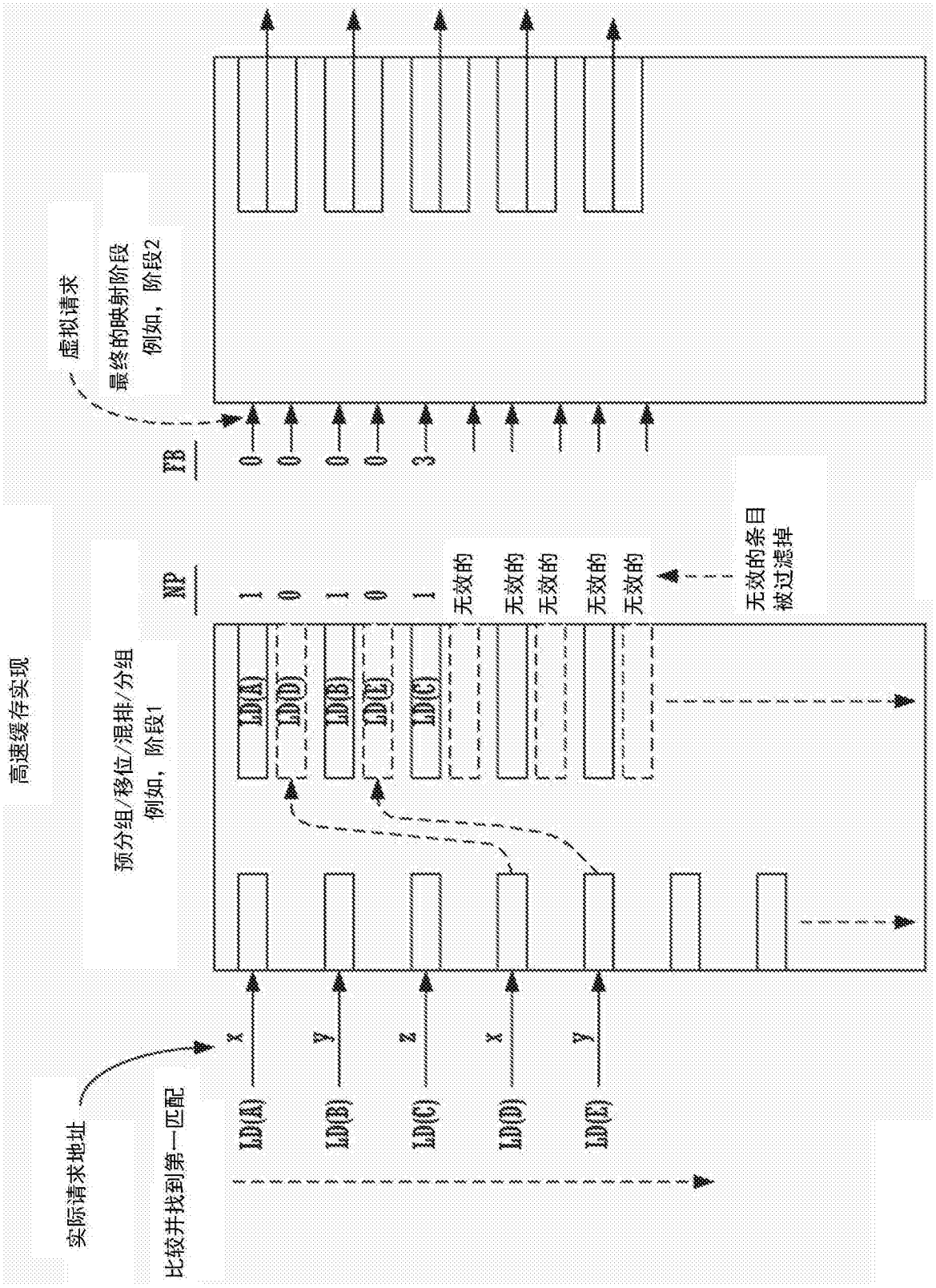

高速缓存实现
实际请求地址
比较并找到第一匹配
预分组/移位/混排/分组
例如，阶段1
虚拟请求
最终的映射阶段
例如，阶段2
XP
TB
LD(A)
LD(B)
LD(C)
LD(D)
LD(E)
x
y
z
x
y
0
0
0
0
3
1
0
1
0
1
无效的
无效的
无效的
无效的
无效的
无效的条目
被过滤掉

图9

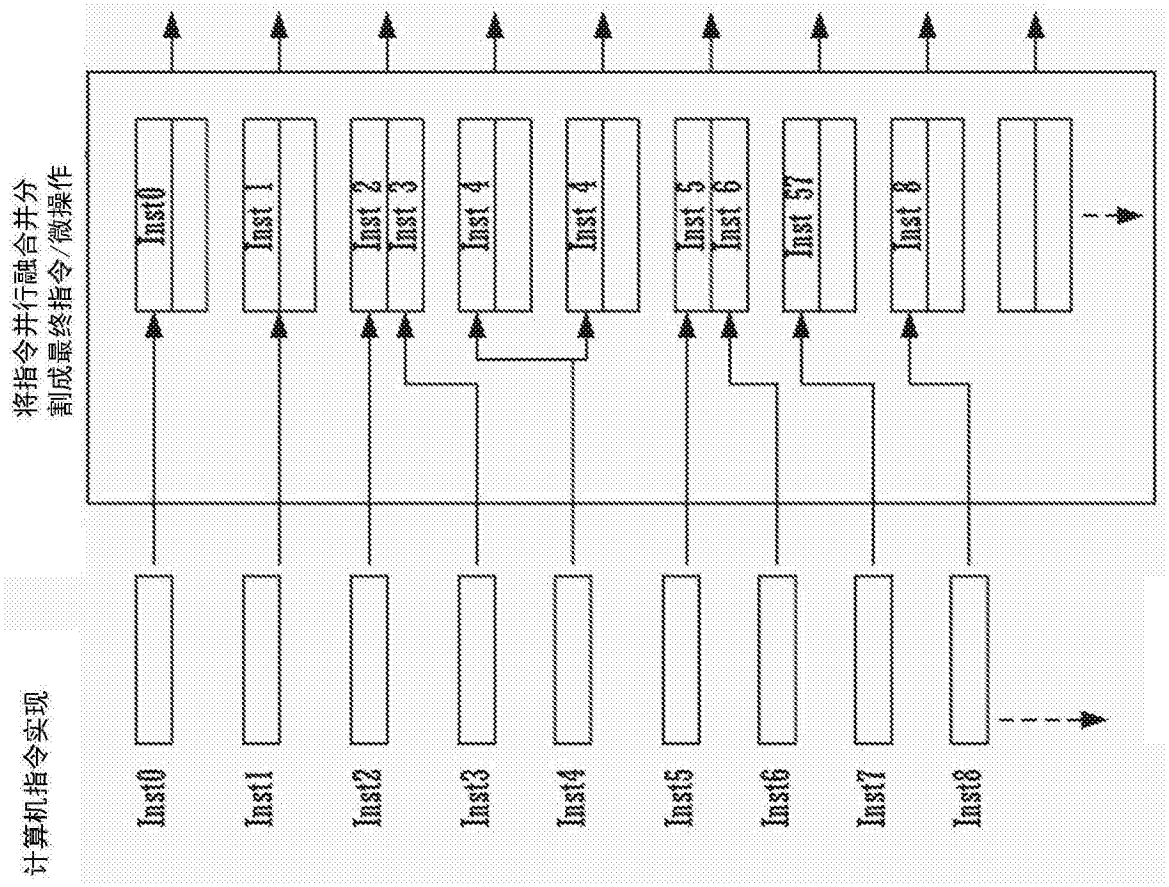

将指令并行融合并分割成最终指令/微操作
计算机指令实现
Inst0
Inst 1
Inst 2
Inst 3
Inst 4
Inst 4
Inst 5
Inst 6
Inst 57
Inst 8
Inst0
Inst1
Inst2
Inst3
Inst4
Inst5
Inst6
Inst7
Inst8

图10

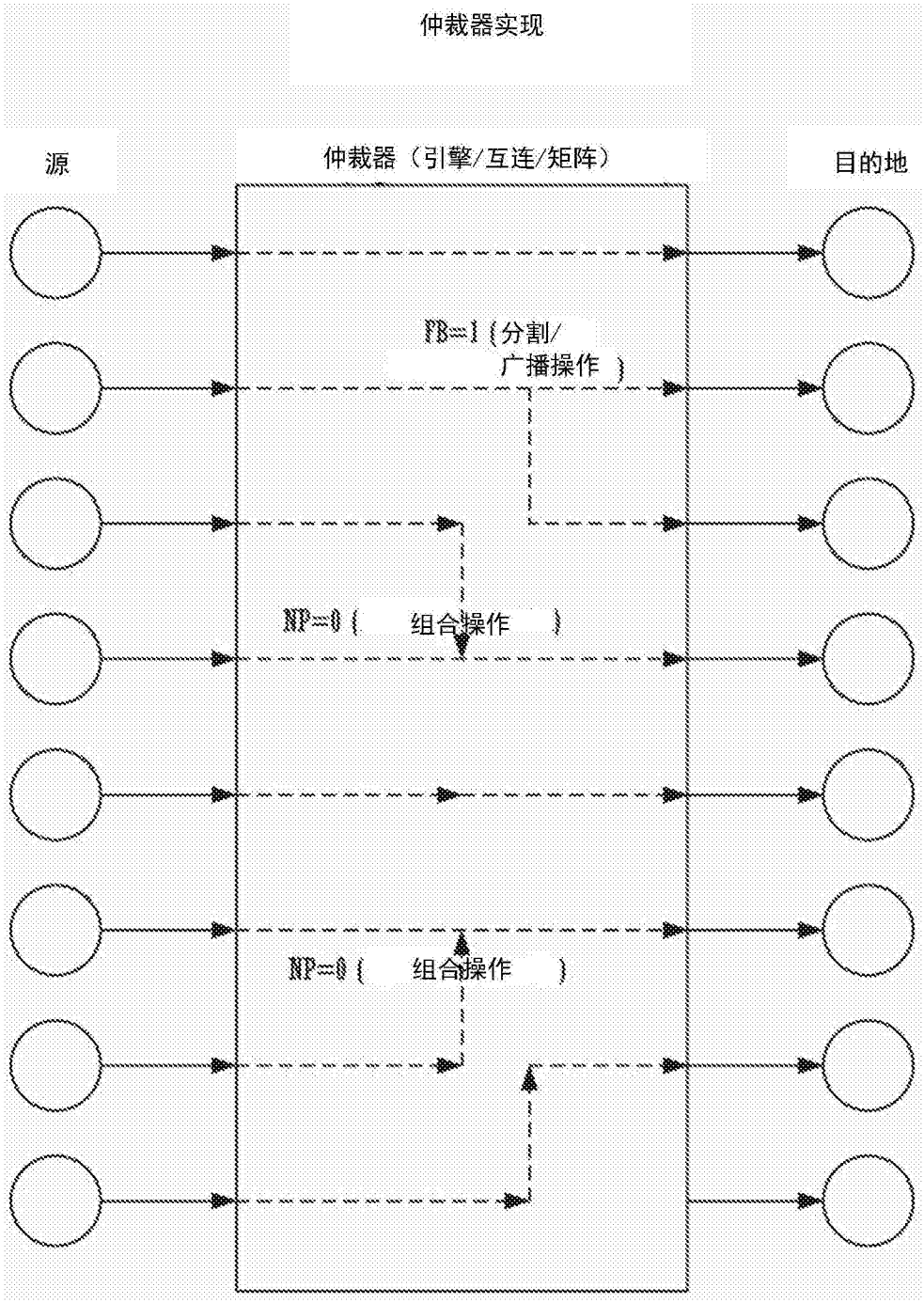


图11

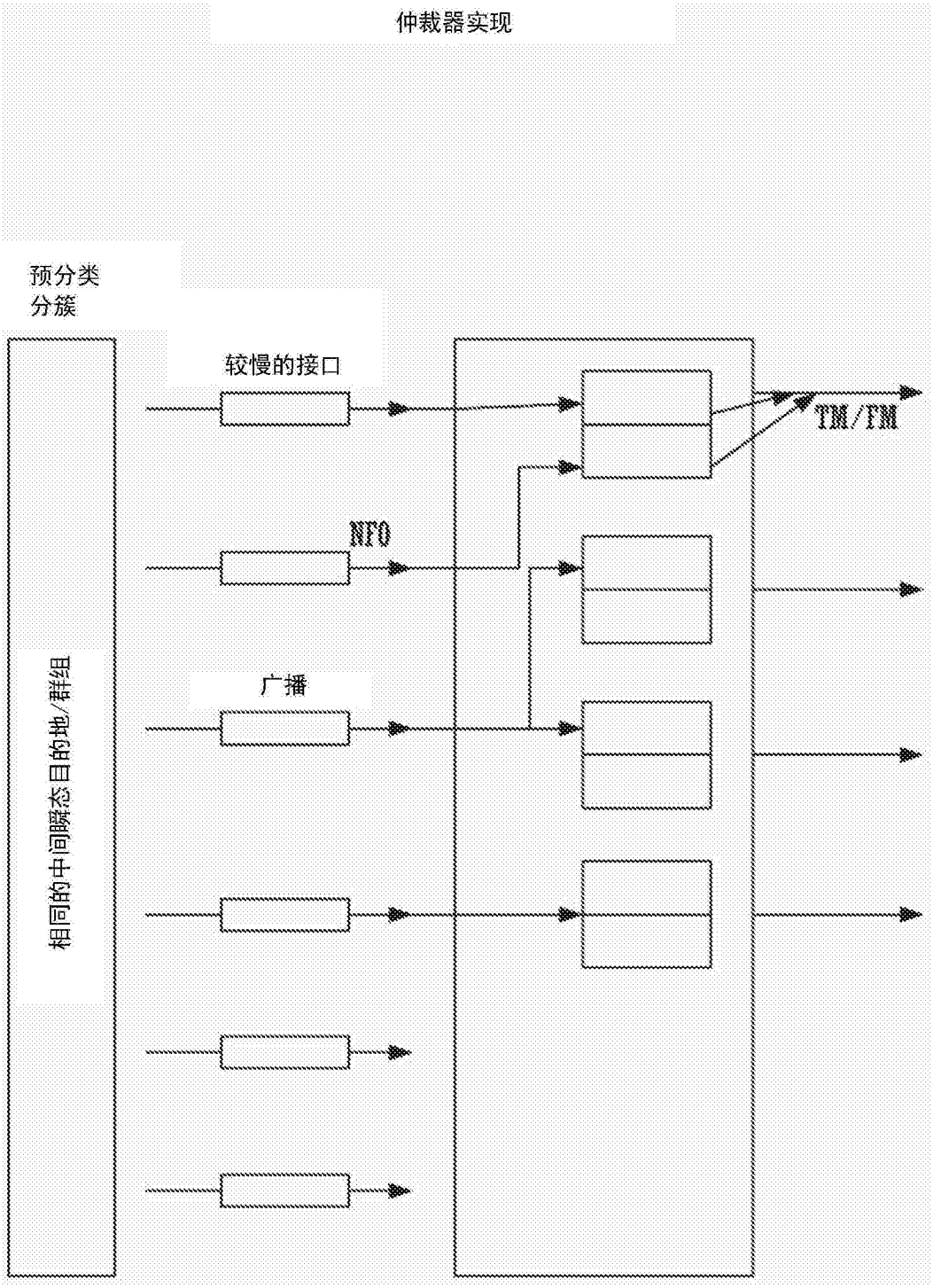


图12

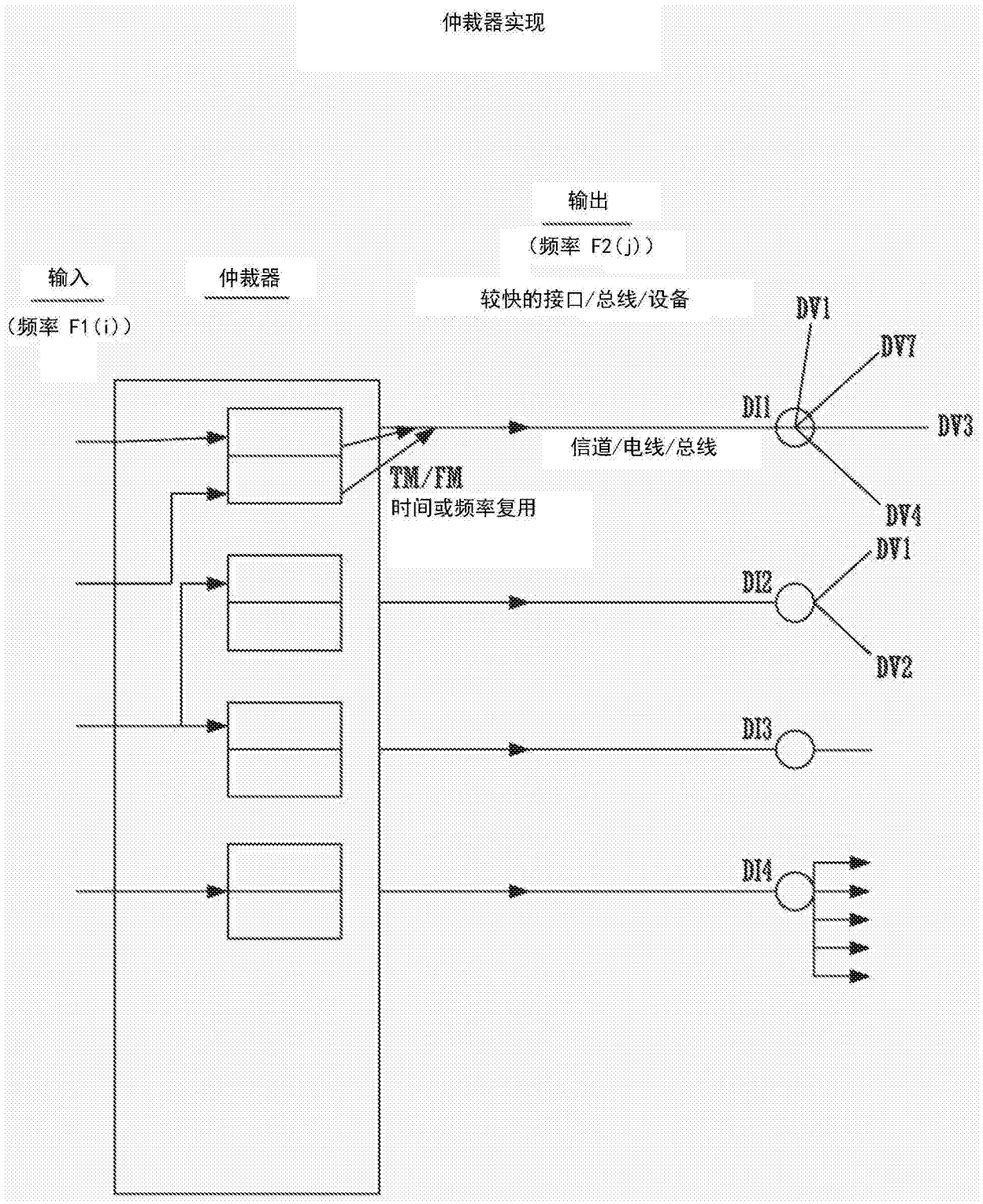

仲裁器实现
输出
（频率 F2(j)）
输入
（频率 F1(i)）
仲裁器
较快的接口/总线/设备
DV1
DV7
DI1
DV3
信道/电线/总线
TM/FM
时间或频率复用
DV4
DV1
DI2
DV2
DI3
DI4

图13

并行线路速度分组&分割

K-索引	D	E	F	G	I	J	K	M	N			串行模式 PB/BP（递归）	瞬态位置
V-索引	实例开始	初始位置	PB或BP	PB	BP	PB或BP	模式位置	串行模式 PB/BP	并行模式 PB/BP	新的瞬态位置	并行瞬态位置+1	&##	
3	1	0	1	0	0	0	0	0	0	0	1	0	G3+或(MOD(G3,2),IG)*(MOD(E3+MOD(SUM(M$2:M2),2),2))	G3+SUM(M$2:M3)-G3
4	1	1	1	0	0	0	1	0	0	1	2	0	G4+或(MOD(G4,2),IG)*(MOD(E4+MOD(SUM(M$2:M3),2),2))	G4+SUM(M$2:M4)-G4
5	1	2	1	0	0	0	0	0	0	2	3	1	G5+或(MOD(G5,2),IG)*(MOD(E5+MOD(SUM(M$2:M4),2),2))	G5+SUM(M$2:M5)-G5
6	1	3	0	0	1	1	1	1	1	4	5	2	G6+或(MOD(G6,2),IG)*(MOD(E6+MOD(SUM(M$2:M5),2),2))	G6+SUM(M$2:M6)-G6
7	1	4	0	1	1	1	0	2	2	6	7	3	G7+或(MOD(G7,2),IG)*(MOD(E7+MOD(SUM(M$2:M6),2),2))	
8	1	5	1	0	0	0	1	0	0	8	9	4	G8+或(MOD(G8,2),IG)*(MOD(E8+MOD(SUM(M$2:M7),2),2))	
9	1	6	0	3	0	1	0	4	4	10	11	5	G9+或(MOD(G9,2),IG)*(MOD(E9+MOD(SUM(M$2:M8),2),2))	
10	1	7	0	0	1	1	1	0	0	14	15	7	G10+或(MOD(G10,2),IG)*(MOD(E10+MOD(SUM(M$2:M9),2),2))	

E-索引	D	E	F	G	I	J	K	M	N				并行模式 PB/BP	并行瞬态位置
V-索引	实例开始	初始位置	PB或BP	PB	BP	PB或BP	模式位置	串行模式 PB/BP	并行模式 PB/BP	新的瞬态位置	并行瞬态位置+1	&##		
3	1	0	1	0	0	0	0	0	0	0	1	0	G3+IF(MOD(E3,2)	SUM(M$3:M3)-G3
4	1	1	1	0	0	0	1	0	0	1	2	0	G4+IF(MOD(E4+G3+(IF(E3,2)	SUM(M$3:M4)-G4
5	1	2	1	0	0	0	0	0	0	2	3	1	G5+IF(MOD(E5+G4+(IF(E4+G3)+(IF(E3,2)	SUM(M$3:M5)-G5
6	1	3	0	0	1	1	1	1	1	4	5	2	G6+IF(MOD(E6+G5+(IF(E5+G4)+(IF(E4+G3)+(IF(E3,2)	SUM(M$3:M6)-G6
7	1	4	0	1	1	1	0	2	2	6	7	3	G7+IF(MOD(E7+G6+(IF(E6+G5)+(IF(E5+G4)+(IF(E4+G3)+(IF(E3,2)	
8	1	5	1	0	0	0	1	0	0	8	9	4	G8+IF(MOD(E8+G7+(IF(E7+G6)+(IF(E6+G5)+(IF(E5+G4)+(IF(E4+G3)+(IF(E3,2)	
9	1	6	0	3	0	1	0	4	4	10	11	5	G9+IF(MOD(E9+G8+(IF(E8+G7)+(IF(E7+G6)+(IF(E6+G5)+(IF(E5+G4)+(IF(E4+G3)+(IF(E3,2)	
10	1	7	0	0	1	1	1	0	0	14	15	7	G10+IF(MOD(E10+G9+(IF(E9+G8)+(IF(E8+G7)+(IF(E7+G6)+(IF(E6+G5)+(IF(E5+G4)+(IF(E4+G3)+(IF(E3,2)	

图14

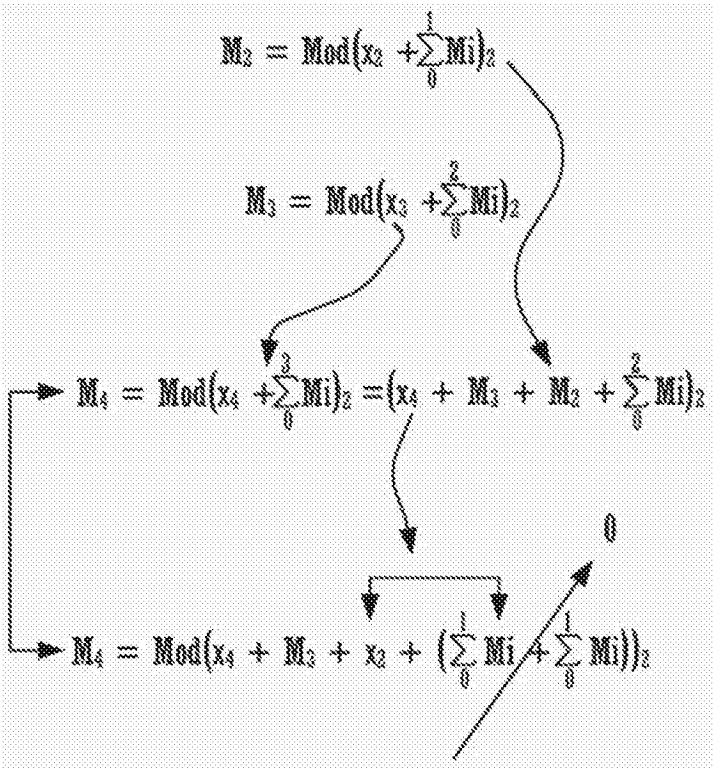

M2 = Mod(x2 + ∑ 0 1 Mi)2
M3 = Mod(x3 + ∑ 0 2 Mi)2
M4 = Mod(x4 + ∑ 0 3 Mi)2 =(x4 + M3 + M2 + ∑ 0 2 Mi)2
0
M4 = Mod(x4 + M3 + x2 + (∑ 0 1 Mi + ∑ 0 1 Mi))2

图15

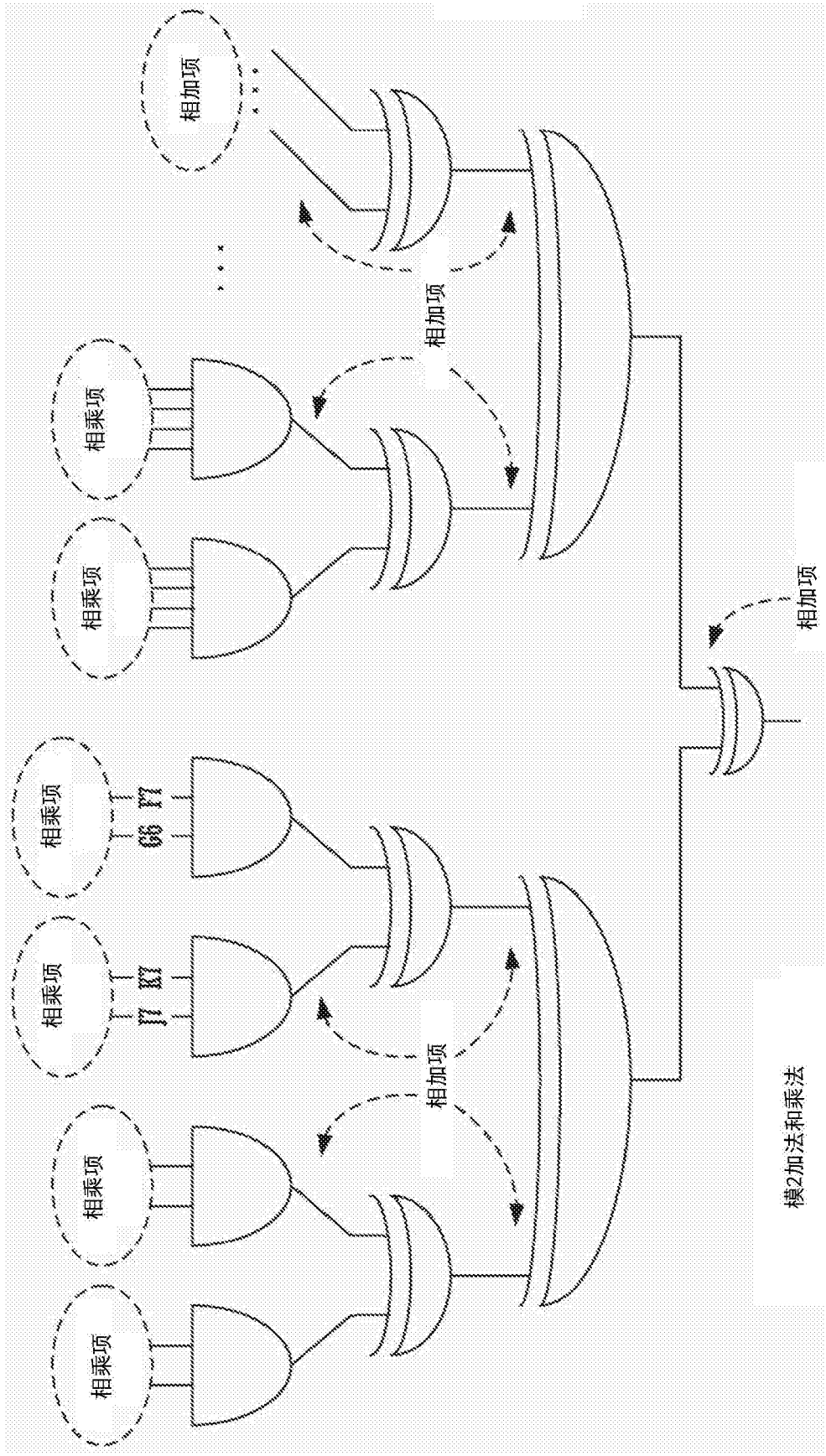


图16

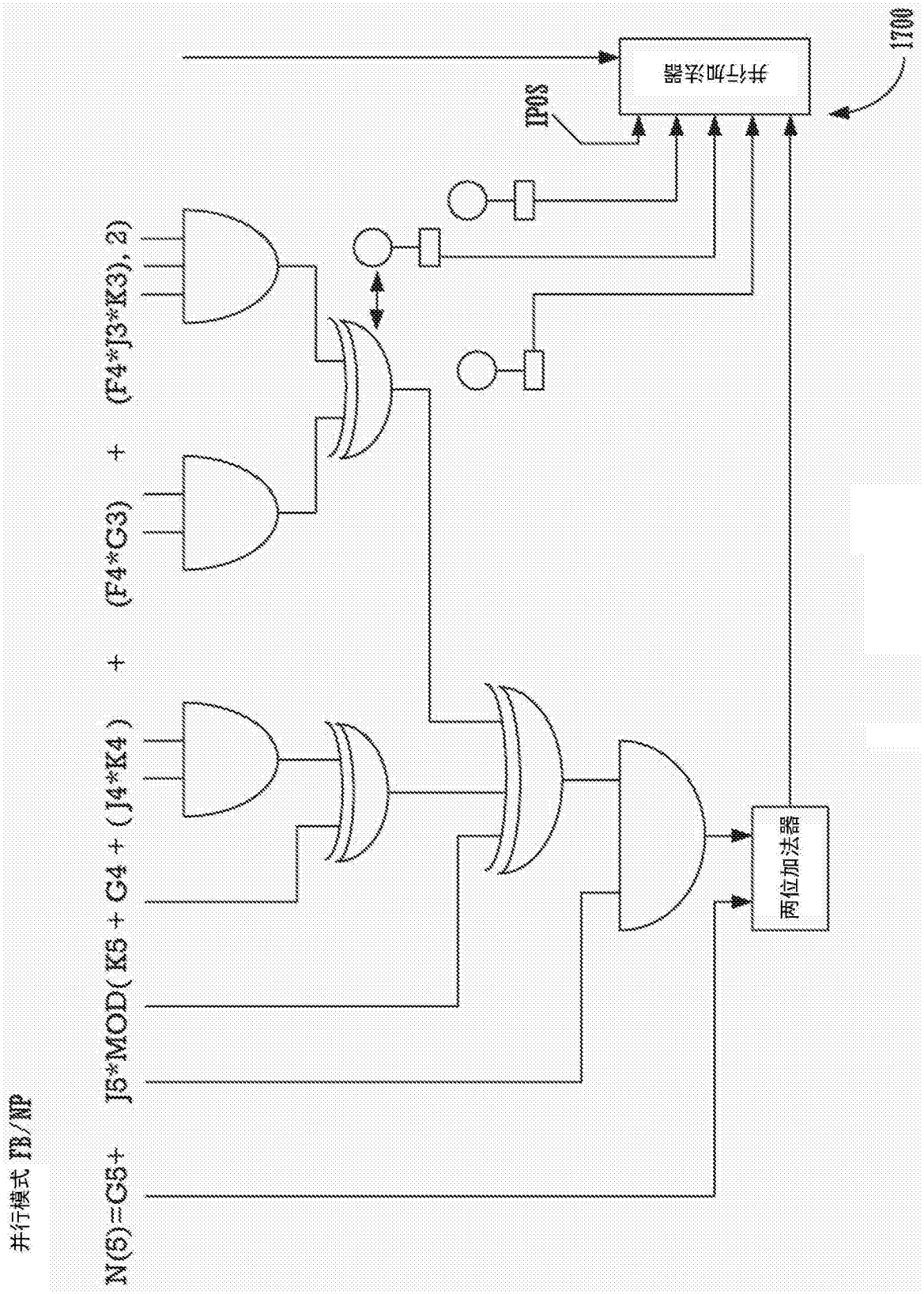

并行模式 FB/MP
N(6)=G5+ J5*MOD(K5 + G4 + (J4*K4) + (F4*G3) + (F4*J3*K3), 2)
两位加法器
1ms
并行加法器
1100

图17

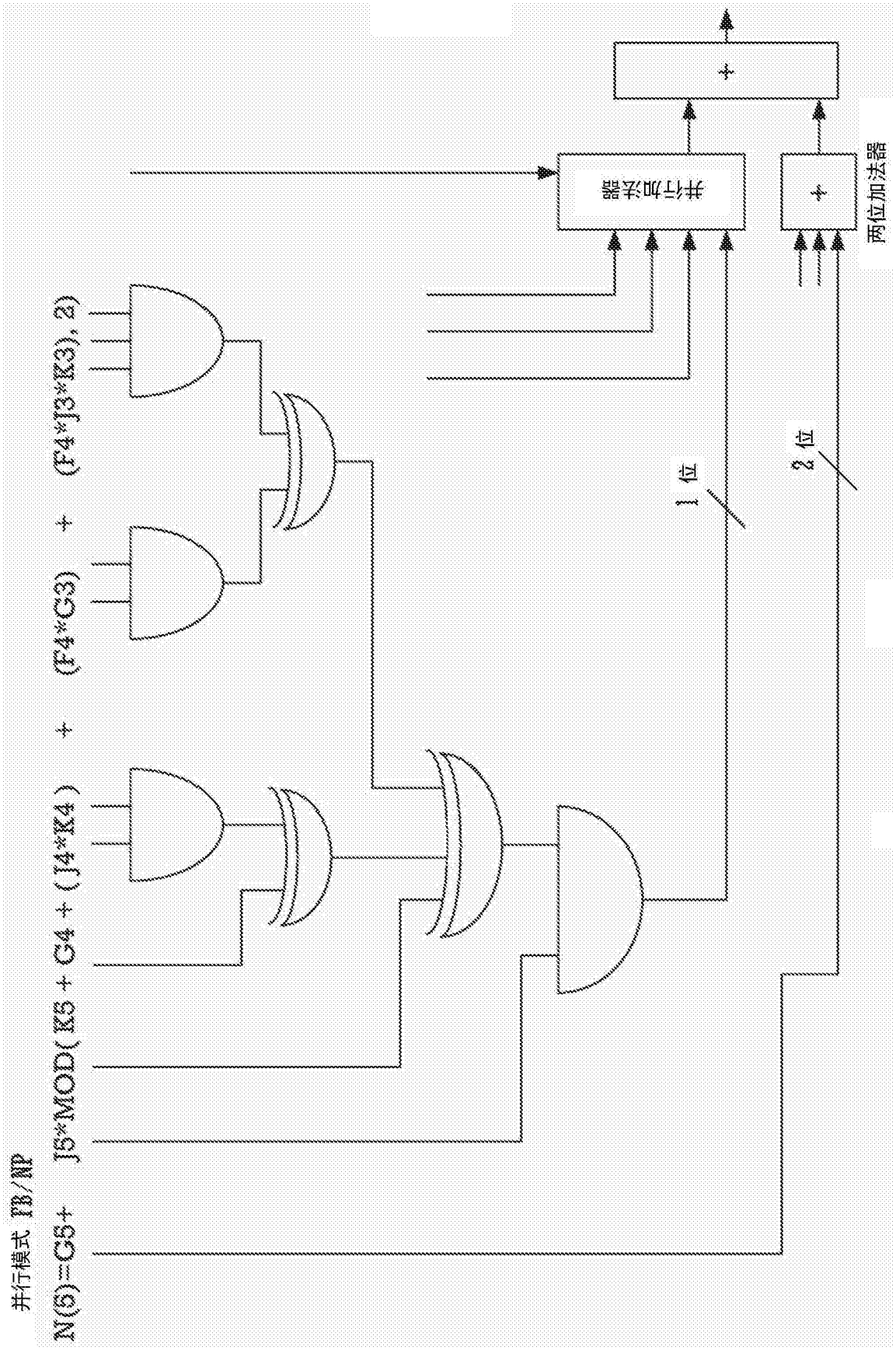

并行模式 FB/MP
N(5)=G5+ J5*MOD(K5 + G4 + (J4*K4) + (F4*G3) + (F4*J3*K3),2)
两位加法器
并行加法器
1 位
2 位

图18

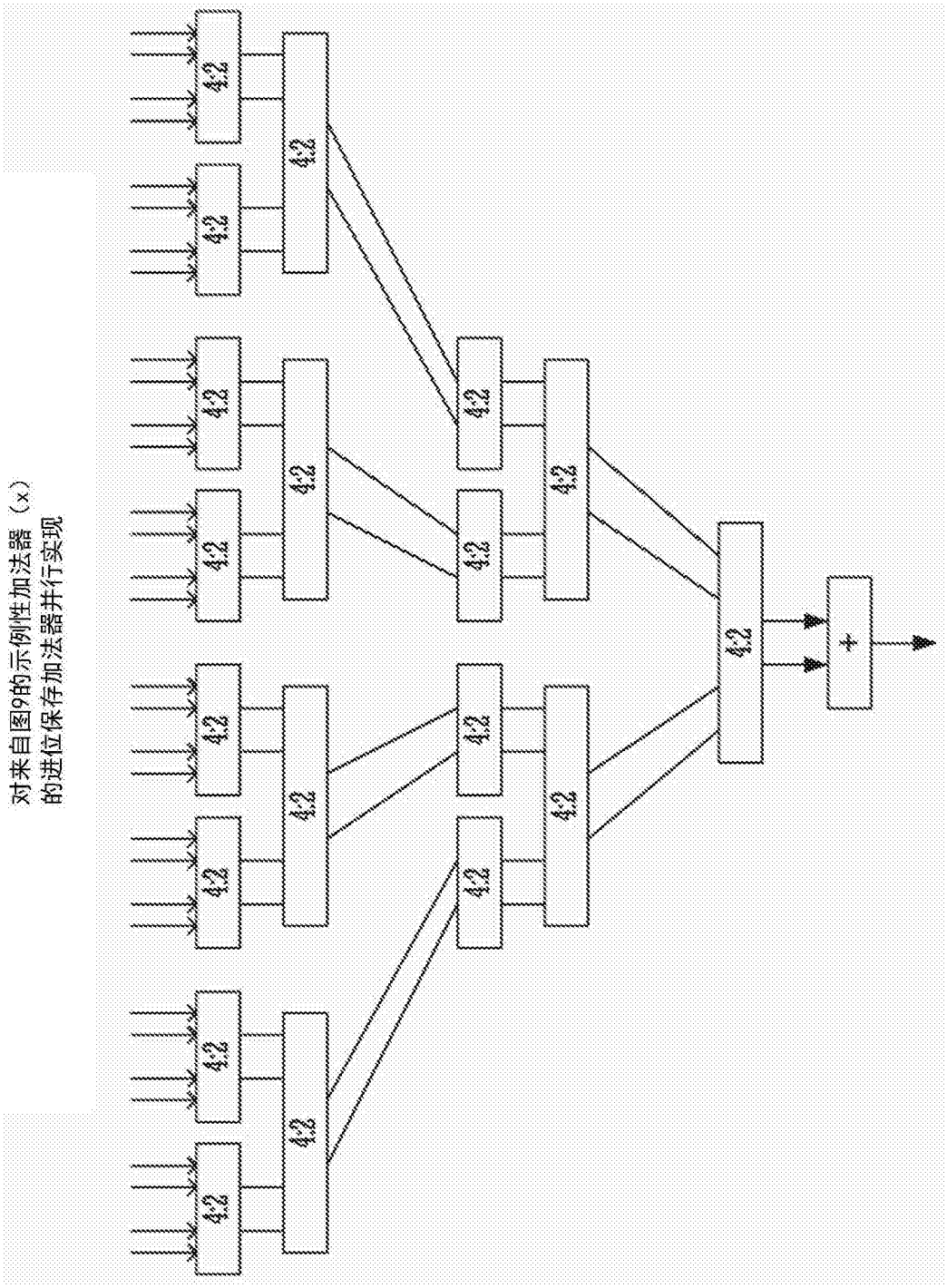


图19

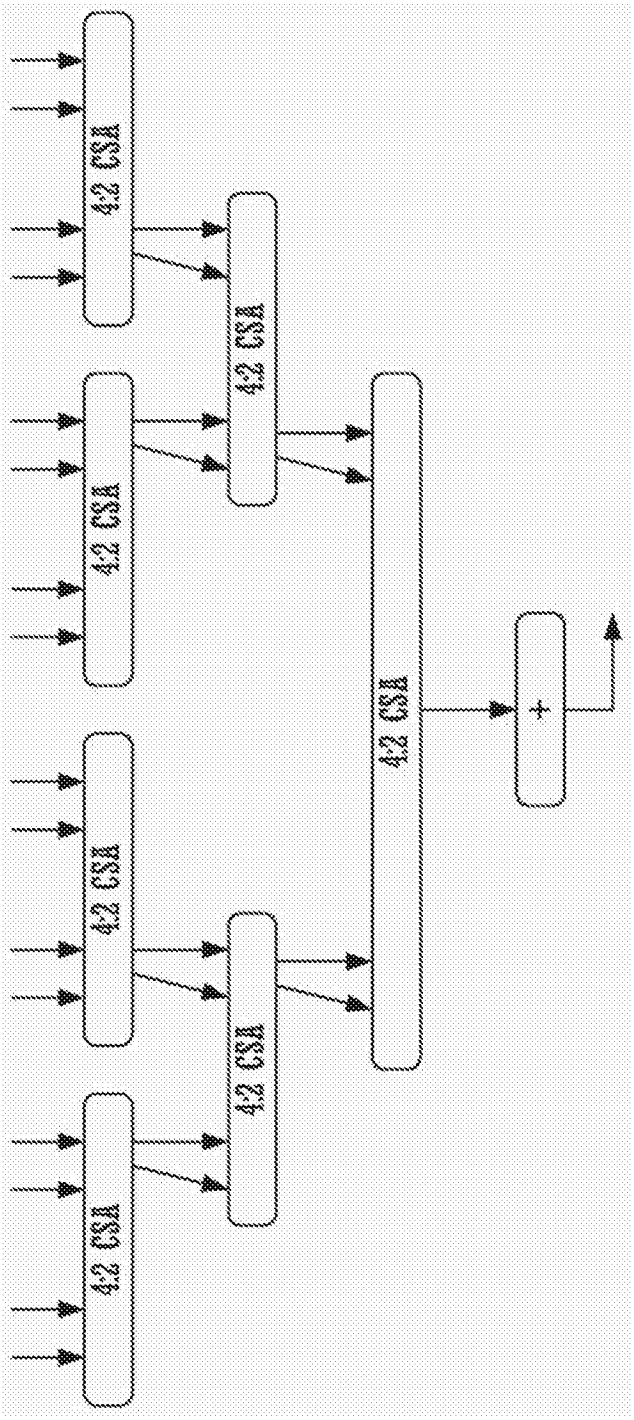


图20

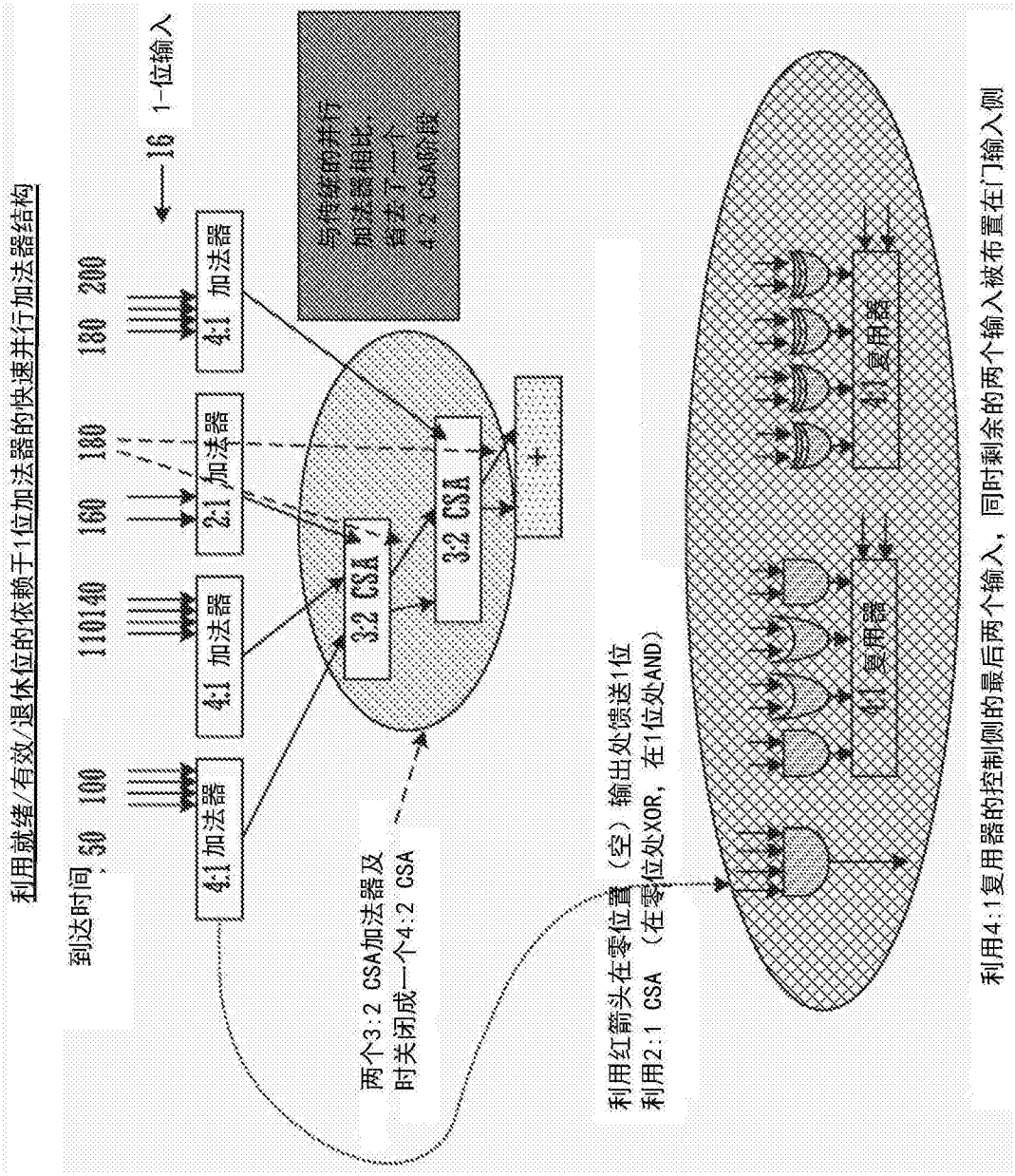

利用就绪/有效/退休位的依赖于1位加法器的快速并行加法器结构
到达时间，50 100 110140 160 180 180 200
16 1-位输入
4:1 加法器
4:1 加法器
2:1 加法器
4:1 加法器
3:2 CSA
3:2 CSA
与传统的并行
加法器相比，
省去了一个
4:2 CSA阶段
两个3:2 CSA加法器及
时关闭成一个4:2 CSA
利用红箭头在零位置（空）输出处馈送1位
利用2:1 CSA （在零位处XOR，在1位处AND）
4:1 复用器
4:1 复用器
利用4:1复用器的控制侧的最后两个输入，同时剩余的两个输入被布置在门输入侧

图21