

[19] 中华人民共和国国家知识产权局



[12] 发明专利申请公开说明书

[21] 申请号 200480018443.8

[51] Int. Cl.

G06F 9/30 (2006.01)

G06F 9/315 (2006.01)

[43] 公开日 2006 年 8 月 2 日

[11] 公开号 CN 1813241A

[22] 申请日 2004.6.24

[74] 专利代理机构 上海专利商标事务所有限公司

[21] 申请号 200480018443.8

代理人 李 玲

[30] 优先权

[32] 2003.6.30 [33] US [31] 10/611,344

[86] 国际申请 PCT/US2004/020601 2004.6.24

[87] 国际公布 WO2005/006183 英 2005.1.20

[85] 进入国家阶段日期 2005.12.29

[71] 申请人 英特尔公司

地址 美国加利福尼亚州

[72] 发明人 小 W · 梅西 E · 德贝斯

P · 鲁塞尔 H · 恩古彦

权利要求书 7 页 说明书 37 页 附图 30 页

[54] 发明名称

混洗数据的方法和装置

[57] 摘要

用于混洗数据的方法、装置和程序。一个实施例的方法包括接收具有一组 L 个数据元素的第一操作数和具有一组 L 个控制元素的第二操作数。对于每个控制元素，如果其排空到零字段未设置，则来自由个别控制元素指定的第一操作数数据元素的数据被混洗到相关联的结果数据元素位置，而如果其排空到零字段未设置，则将零置入相关联的结果数据元素位置。

1.一种方法，包括：

接收具有一组 L 个数据元素的第一操作数和具有一组 L 个控制元素的第二
5 操作数；以及

对于每个控制元素，如果未设置其排空到零字段，将由所述控制元素指定的来自第一操作数数据元素的数据混洗到相关的结果数据元素位置，而如果其排空到零字段未设置，则将零置入所述的相关结果数据元素位置。

2.如权利要求 1 所述的方法，其特征在于，所述 L 个控制元素中的每一个
10 都占据所述第二操作数中的特殊位置并与结果中类似定位的数据元素位置相
关联。

3.如权利要求 2 所述的方法，其特征在于，所述 L 个数据元素中的每一个都占
据所述第一操作数中的特殊位置。

4.如权利要求 3 所述的方法，其特征在于，所述控制元素将通过数据元素
15 位置编号指定第一操作数数据元素。

5.如权利要求 4 所述的方法，其特征在于，所述控制元素中的每一个都包
括：

排空到零字段，所述排空到零字段用于指示与该控制元素相关联的数据元
素位置是否用零值填充；以及

20 选择字段，所述选择字段用于指示混洗来自哪个第一操作数数据元素的数
据。

6.如权利要求 5 所述的方法，其特征在于，所述控制元素中的每一个都进
一步包含源选择字段。

7.如权利要求 2 所述的方法，其特征在于，还包括输出结果数据块，该数
25 据块包括响应于所述第二操作数的所述控制元素从所述第一操作数混洗的数
据。

8.如权利要求 1 所述的方法，其特征在于，所述数据元素中的每一个都包
括一字节的数据。

9.如权利要求 8 所述的方法，其特征在于，所述控制元素中的每一个都是

一字节宽。

10.如权利要求 9 所述的方法，其特征在于，L 是 8 且其中所述第一操作数、所述第二操作数和所述结果分别由 64 比特宽的紧缩数据构成。

11.如权利要求 9 所述的方法，其特征在于，L 是 16 且其中所述第一操作数、所述第二操作数和所述结果分别由 128 比特宽的紧缩数据构成。
5

12.一种装置，包括：

执行单元，用于执行包括含一组 L 个数据元素的第一操作数和含一组 L 个控制元素的第二操作数的混洗指令，所述混洗指令用于使得所述执行单元：

对于每个单独的控制元素，确定其排空到零字段是否被设置，且如果是
10 真则将零置入相关的结果数据元素位置，否则将来自由所述单个控制元素指定的第一操作数数据元素的数据混洗到所述相关联的结果数据元素位置。

13.如权利要求 12 所述的装置，其特征在于，所述 L 个控制元素中的每一个都占据所述第二操作数中的一位置并与结果中类似定位的数据元素位置相
15 关联。

14.如权利要求 13 所述的装置，其特征在于，每个单个控制元素将通过数据元素位置编号指定一第一操作数数据元素。

15.如权利要求 14 所述的装置，其特征在于，所述控制元素中的每一个都包括：

20 排空到零字段，所述排空到零字段用于指示与该控制元素相关联的数据元素位置是否用零值填充；以及

选择字段，所述选择字段用于指示混洗来自哪个第一操作数数据元素的数据。

16.如权利要求 15 所述的装置，其特征在于，所述控制元素中的每一个都
25 进一步包括源选择字段。

17.如权利要求 16 所述的装置，其特征在于，所述混洗指令将进一步使得所述执行单元生成具有 L 个数据元素位置的结果，这 L 个数据元素位置已基于所述一组 L 个控制元素被填充。

18.如权利要求 12 所述的装置，其特征在于，所述数据元素中的每一个都

包括一字节数据且每个所述控制元素都是字节宽。

19.如权利要求 18 所述的装置，其特征在于，L 是 8 且其中所述第一操作数、所述第二操作数和所述结果分别由 64 比特宽的紧缩数据构成。

20.如权利要求 18 所述的装置，其特征在于，L 是 16 且其中所述第一操作数、所述第二操作数和所述结果分别由 128 比特宽的紧缩数据构成。

21.一种包含存储表示预定功能的数据的机器可读媒介的制品，包括：

接收具有一组 L 个数据元素的第一操作数和具有一组 L 个控制元素的第二操作数；以及

对于每个控制元素，如果未设置其排空到零字段，则将来自由所述控制元 10 素指定的第一操作数数据元素的数据混洗到相关联的结果数据元素位置，如果其排空到零字段未设置，则将零置入所述相关联的结果数据元素位置。

22.如权利要求 21 所述的制品，其特征在于，所述机器可读媒介存储的所述数据表示集成电路设计，在制造时响应于单个指令执行所述预定功能。

23.如权利要求 22 所述的制品，其特征在于，所述预定功能还包括生成具有 L 个数据元素位置的结果，这 L 个数据元素位置已根据所述一组 L 个控制元 15 素被填充。

24.如权利要求 23 所述的制品，其特征在于，所述 L 个控制元素中的每一个都与结果中的类似定位的数据元素位置相关联。

25.如权利要求 24 所述的制品，其特征在于，每个单个控制元素将通过一 20 数据元素位置编号指定第一操作数数据元素。

26.如权利要求 25 所述的制品，其特征在于，所述数据元素中的每一个都包括一字节数据。

27.如权利要求 26 所述的制品，其特征在于，每个所述控制元素都包括：

排空到零字段，所述排空到零字段用于指示与该控制元素相关联的数据元 25 素位置是否用零值填充；以及

选择字段，所述选择字段用于指示混洗来自哪个第一操作数数据元素的数据。

28.如权利要求 27 所述的制品，其特征在于，所述控制元素中的每一个都进一步包括源选择字段。

29.如权利要求 21 所述的制品，其特征在于，由所述机器可读媒介存储的所述数据表示计算机指令，在由机器执行时使得所述机器执行所述预定功能。

30.一种方法，包括：

接收具有一组 L 个数据元素的第一操作数；

5 接收具有一组 L 个掩码的第二操作数，其中所述 L 个掩码中的每一个都占据所述第二操作数中的一特殊位置并与结果中的类似定位的数据元素位置相关联，所述 L 个掩码中的每一个包括一排空到零字段；

对于每个掩码，确定是否设置了其排空到零字段，且如果是真，则将零置入相关联的结果数据元素位置；以及

10 如果其排空到零字段未设置，则将来自由所述掩码指定的第一操作数数据元素的数据混洗到所述相关联的结果数据元素位置。

31.如权利要求 30 所述的方法，其特征在于，所述 L 个掩码中的每一个都占据所述第二操作数中的一特殊位置并与所述结果中的类似定位的数据元素位置相关联。

15 32.如权利要求 31 所述的方法，其特征在于，所述 L 个掩码中的每一个都包括：

排空到零字段，所述排空到零字段用于指示与该控制元素相关联的数据元素位置是否用零值填充；以及

20 选择字段，所述选择字段用于指示混洗来自哪个第一操作数数据元素的数据。

33.如权利要求 32 所述的方法，其特征在于，所述掩码中的每一个都进一步包括源选择字段。

34.如权利要求 33 所述的方法，其特征在于，所述第一操作数、所述第二操作数和所述结果分别由 64 比特宽的紧缩数据构成。

25 35.如权利要求 33 所述的方法，其特征在于，所述第一操作数、所述第二操作数和所述结果分别由 128 比特宽的紧缩数据构成。

36.一种方法，包括：

接收具有一组 L 个数据元素的第一操作数；

接收具有一组 L 个混洗掩码的第二操作数，所述 L 个混洗掩码中的每一个

都与结果中的类似定位的数据元素位置相关联；

对于每个单个混洗掩码，确定其排空到零字段是否被设置，且如果是真，则将零置入相关联的结果数据元素，否则将来自由所述单个混洗掩码指定的第一操作数数据元素的数据混洗到所述相关联的结果数据元素位置。

5 37.如权利要求 36 所述的方法，其特征在于，所述 L 个混洗掩码的每一个包括：

排空到零字段，所述排空到零字段用于指示与该控制元素相关联的数据元素位置是否用零值填充；以及

10 选择字段，所述选择字段用于指示混洗来自哪个第一操作数数据元素的数据。

38.如权利要求 37 所述的方法，其特征在于，所述掩码中的每一个还包括源选择字段。

39.一种装置，包括：

第一存储器位置，用于存储多个源数据元素；

15 第二存储器位置，用于存储多个控制元素，所述控制元素中的每一个对应于一结果数据元素位置，且所述控制元素中的每一个都包括排空到零字段和选择字段；

控制逻辑，它耦合到所述第二存储器位置，所述控制逻辑响应于所述控制元素的值生成多个选择信号和多个排空到零信号；

20 第一多个多路复用器，它耦合到所述第一存储器位置和所述多个选择信号，所述第一多个多路复用器中的每一个响应于与一特定结果数据元素位置相对应的选择信号混洗用于所述特定结果数据元素位置的数据元素；以及

25 第二多个多路复用器，它耦合到所述第一多个多路复用器和所述多个排空到零信号，所述第二多个多路复用器中的每一个与一特定结果数据元素位置相关联，如果其排空到零信号是有效的则所述第二多个多路复用器中的每一个输出零否则输出为该特定结果数据元素位置混洗的数据元素。

40.如权利要求 39 所述的装置，其特征在于，所述多个源数据元素是第一紧缩数据操作数。

41.如权利要求 40 所述的装置，其特征在于，所述多个控制元素是第二紧

缩数据操作数。

42.如权利要求 40 所述的装置，其特征在于，所述第一和第二存储器位置是单指令多数据寄存器。

43.如权利要求 42 所述的装置，其特征在于，

5 所述第一紧缩操作数是 64 比特长且所述源数据元素中的每一个都是一字节宽；以及

所述第二紧缩操作数是 64 比特长且所述控制元素中的每一个是一字节宽。

44.如权利要求 42 所述的装置，其特征在于，

所述第一紧缩操作数是 128 比特长且所述源数据元素中的每一个都是一字节宽；以及

所述第二紧缩操作数是 128 比特长且所述控制元素中的每一个是一字节宽。

45.一种装置，包括：

控制逻辑，用于接收一组 L 个混洗掩码，其中每个混洗掩码与一唯一的结果数据元素位置相关联，所述控制逻辑用于为每个结果数据元素位置提供选择信号和排空到零信号；

一组 L 个多路复用器，它们耦合到所述控制逻辑，其中每个多路复用器也与一唯一的结果数据元素位置相关联，如果其相关联的排空到零信号有效则每个多路复用器输出零，且如果其相关联的排空到零信号无效则输出基于其相关联的选择信号从一组 M 个数据元素混洗的数据。

46.如权利要求 45 所述的装置，其特征在于，还包括具有 L 个唯一数据元素位置的寄存器，每个数据元素位置保持来自其相关联的多路复用器的输出。

47.如权利要求 46 所述的装置，其特征在于，L 是 16，M 是 16。

48.一种系统，包括：

25 存储器，用于存储数据和指令；

处理器，它耦合到总线上的所述存储器，所述处理器可操作用于执行混洗操作，所述处理器包括：

总线单元，用于接收来自所述存储器的指令，所述指令引起基于来自第二操作数的一组 L 个混洗控制元素的对来自第一操作数的 L 个数据元素中的

至少一个的数据混洗；

执行单元，它耦合到所述总线单元，所述执行单元执行所述指令，所述指令使得所述执行单元：

对每个混洗控制元素，如果其排空到零字段未设置，则将来自由所述混洗控制元素指定的第一操作数数据元素的数据混洗到相关联的结果数据元素位置，且如果其排空到零字段未设置，则将零置入所述相关联的结果数据元素位置。

49.如权利要求 48 所述的系统，其特征在于，每个混洗控制元素都包括：

排空到零字段，所述排空到零字段指示与该混洗控制元素相关联的数据元
素位置是否将用零值填充；以及

选择字段，所述选择字段用于指示混洗来自哪个第一操作数数据元素的数据。

50.如权利要求 49 所述的系统，其特征在于，每个混洗控制元素都进一步包括源选择字段。

51.如权利要求 48 所述的系统，其特征在于，所述指令是具有排空到零能力的紧缩字节混洗指令。

52.如权利要求 48 所述的系统，其特征在于，每个数据元素都是字节宽，每个混洗命令元素是字节宽，且 L 是 8。

53.如权利要求 48 所述的系统，其特征在于，所述第一操作数是 64 比特长
20 且所述第二操作数是 64 比特长。

混淆数据的方法和装置

5 本申请是 2001 年 10 月 29 日提交的标题为 “An Apparatus and Method For Efficient Filtering And Convolution Of Content Data” 的美国专利申请 No.09/952891 的延续。

本申请涉及以下内容：2003 年 6 月 30 日提交的标题为 “Method And Apparatus For Parallel Table Lookup Using SIMD Instructions” 的共同待批的美国专利申请
10 No. ；以及 2003 年 6 月 30 日提交的标题为 “Method And Apparatus For Rearranging Data Between Multiple Registers” 的共同待批的美国专利申请 No. 。

技术领域

本发明一般涉及微处理器和计算机系统领域。本发明尤其涉及用于混淆数据
15 的方法和装置。

发明背景

计算机系统已在当今社会中变得日益普遍。在许多职业中，计算机的处理能力提升了工人的效率和产量。随着购买和拥有计算机的成本持续下降，越来越多的
20 消费者能利用更新、更快的机器。此外，许多人因为很自由而乐于使用笔记本计算机。移动计算机允许用户在离开办公室或旅行时方便地随他们一起携带数据和工作。这种情况是营销人员、公司职员以及甚至学生们所熟悉的。

随着处理器技术的进步，还产生了更新的软件代码以便在具有这些处理器的
机器上运行。用户通常期望和需要他们计算机的更高的性能，而不管使用的软件类
25 型。一个这种问题会由于处理器内实际执行的指令和操作的种类而产生。基于所需的操作复杂性和/或电路类型，某些类型的操作需要更长的时间来完成。这提供了优化某些复杂操作在处理器内部执行的方式的机会。

在超过 10 年的时间里，媒体应用驱动着微处理器的发展。事实上，近些年的多数计算升级都是由媒体应用驱动的。这些升级主要出现于消费者方面，尽管在用

于娱乐增强教育和通信用途的企业方面也看到了显著的进步。然而，未来的媒体应用将需要更高的计算要求。结果，明天的个人计算体验在视听效果方面将更加丰富且更易于使用，更重要地是计算将与通信合并在一起。

因此，图像显示以及统称为内容的音频和视频数据的回放已变成当前计算装置的日益流行的应用。⁵ 过滤和卷积操作是对诸如图像音频和视频数据的内容数据执行的最普通操作中的一些。这些操作是计算密集的，但用诸如单指令多数据(SIMD)寄存器的各种数据存储装置提供可通过有效实现加以利用的高水平的数据并行。许多当前架构还需要不必要的数据类型变化，他们最小化指令处理量并显著增加排序用于算术操作的数据所需的时钟周期数。

10

附图说明

附图中作为示例而非限制地说明本发明，其中相同标号表示相似的元件，其中：

图 1A 是根据本发明一实施例的由包含执行用于混洗数据的指令的执行单元¹⁵ 的处理器形成的计算机系统的框图；

图 1B 是根据本发明可选实施例的另一示例性计算机系统的框图；

图 1C 是根据本发明另一可选实施例的又一示例性计算机系统的框图；

图 2 是根据本发明的包括执行数据混洗操作的逻辑电路的一实施例的处理器的微架构的框图；

20 图 3A-C 示出了根据本发明各实施例的混洗掩码；

图 4A 示出了根据本发明一实施例的多媒体寄存器中的各种紧缩数据类型表示；

图 4B 示出了根据一可选实施例的紧缩数据类型；

图 4C 示出了用于混洗指令的操作编码(操作码)格式的一实施例；

25 图 4D 示出了可选的操作编码格式；

图 4E 示出了另一可选的操作编码格式；

图 5 是根据本发明的基于混洗掩码对数据操作数进行混洗操作的逻辑的一实施例的框图；

图 6 是根据本发明的用于进行数据混洗操作的电路的一实施例的框图；

图 7 示出了根据本发明一实施例的对字节宽数据元素的数据混洗操作；
图 8 示出了根据本发明另一实施例的对字宽数据元素的数据混洗操作的操作；
图 9 是示出混洗数据的方法的一实施例的流程图；
图 10A-H 示出了使用 SIMD 指令的并行表查找算法的操作；
5 图 11 是示出了使用 SIMD 指令进行表查找的方法的一实施例的流程图；
图 12 是示出执行表查找的方法的另一实施例的流程图；
图 13A-C 示出了多个寄存器之间重新排列数据的算法；
图 14 是示出多个寄存器之间重新排列数据的方法的一实施例的流程图；
图 15A-K 示出了用于在多个寄存器之间混洗数据以生成交错数据的算法；以
10 及
图 16 是示出用于在多个寄存器之间混洗数据以生成交错数据的方法的一实施
例的流程图。

具体实施方式

15 揭示了混洗数据的方法和装置。还描述了用于使用 SIMD 指令进行并行表查
找的方法和装置。还揭示了用于在多个寄存器之间重新排列数据的方法和装置。这
里所述的实施例在微处理器的环境中进行描述，但不限于此。虽然参考处理器描述
了以下实施例，但其它实施例也可应用于其它类型的集成电路和逻辑装置。本发明
的相同技术和教导可方便地应用于其它类型的电路或半导体装置，它们也可从较高
20 的管道处理量和改进性能中受益。本发明的教导可应用于执行数据处理的任何处理
器或机器。但，本发明不限于执行 256 比特、128 比特、64 比特、32 比特或 16 比
特数据操作的处理器或机器，而是可以应用于需要数据混洗的任何处理器和机器。

在以下描述中，为了说明目的阐述了许多特定细节，以便提供本发明的透彻
理解。但，本领域的普通技术人员将理解，这些特定细节是实施本发明所非必要的。
25 另一方面，对公知的电气结构和电路则不进行具体的细节说明，以使本发明更加清
楚。此外，出于说明目的，以下描述提供了一些示例且附图示出了各种示例。但，
这些示例不应被理解为是限制性的，因为它们仅旨在于提供本发明的示例而非提供
本发明的所有可能实现的详尽列表。

在一实施例中，本发明的方法体现于机器可执行指令中。这些指令可用于使

用这些指令编程的通用或专用处理器执行本发明的步骤。或者，本发明的步骤可由包含用于上述执行步骤的硬连线逻辑的专门的硬件组件或者编程的计算机组件和定制硬件组件的任何组合执行。

虽然以下示例描述了执行单元和逻辑电路环境下的指令处理和分发，但本发明的其它实施例可借助软件完成。本发明可作为包括存储了指令的机器或计算机可读媒介的计算机程序产品或软件提供，所述指令用于编程计算机(或其它电子装置)以执行根据本发明的过程。这种软件可存储在系统中的存储器内。类似地，代码可经由网络或借助其它计算机可读媒体分发。因此，机器可读媒介可包括用于以机器(例如，计算机)可读形式存储或发送信息的任何机制，但不限于，软盘、光盘、
10 紧致盘、只读存储器(CD-ROM)和磁光盘、只读存储器(ROM)、随机存取存储器(RAM)、可擦可编程只读存储器(EPROM)、电可擦可编程只读存储器(EEPROM)、磁或光卡、闪存、因特网上传输、电、光、声或其它形式的传播信号(例如，载波、红外线信号、数字信号等)等等。

因此，计算机可读媒介包括适于以机器(例如，计算机)可读形式存储或发送电子指令或信息的任何类型的媒体/机器可读媒介。此外，本发明还可作为计算机程序产品下载。这样，程序可从远程计算机(例如，服务器)被传送到请求计算机(例如，客户机)。程序的传送可作为经由通信链接(例如，调制解调器、网络连接等)的载波或其它传播媒介中的电、光、声或其它形式的数据信号。

此外，根据本发明的集成电路设计的实施例可按电子形式作为磁带或其它机器可读媒体上的数据库进行通信或传送。例如，一实施例中处理器的集成电路设计的电子形式可经由工厂处理或制造以获得计算机组件。在另一实例中，电子形式的集成电路设计可由机器处理以模拟计算机组件。因此，一些实施例中的电路布局设计和/或处理器设计可经由机器可读媒体分配或包含于其上用于制成电路或用于集成电路的模拟，在由机器处理时模拟处理器。在其它实施例中，机器可读媒体还能
25 存储表示根据本发明的预定功能的数据。

在现代处理器中，许多不同的执行单元被用于处理和执行各种代码和指令。不是所有的指令都被同等地创建，因为某些较快完成而另一些花费较多的时钟周期数。指令的处理越快，处理器的总体性能越好。因此，使许多指令尽可能快地执行是有利的。但是，存在某些较复杂且需要更多执行时间和处理器资源的指令。例如，

有浮点指令、加载/存储操作、数据移动等。

由于越来越多的计算机系统用于因特网和多媒体应用中，随时间的流逝已引入附加的处理器支持。例如，单指令多数据（SIMD）整数/浮点指令和流 SIMD 扩展（SSE）是减少执行特殊程序任务所需的总指令数的指令。通过并行地对多个数据元素进行操作，这些指令能使软件执行加速。结果，可在较宽范围的应用中实现性能增益，包括视频、语音和图像/照片处理。微处理器和相似类型的逻辑电路中 SIMD 指令的实现通常涉及几个问题。此外，SIMD 操作的复杂性常导致对附加电路的需要，以便正确地处理和操作数据。

本发明的一些实施例提供了一种方式来将具有排空到零能力的紧缩字节混洗指令作为利用 SIMD 相关硬件的算法实现。对于一个实施例，该算法基于根据每个数据元素位置的控制掩码值将数据从特殊寄存器或存储器位置混洗的概念。紧缩字节混洗的实施例可用于减少重排数据的许多不同应用中所需的指令数量。紧缩字节混洗指令还可用于具有非对准加载的任何应用。本混洗指令的实施例可用于过滤，以为有效的乘法—累积操作排列数据。类似地，紧缩混洗指令可用于视频和加密应用，用于排序数据和小查找表。该指令可用于混合来自两个或更多寄存器的数据。因此，根据本发明的具有排空到零能力算法的紧缩混洗的实施例可在处理器中实现以有效支持 SIMD 操作，而不会严重危害整体性能。

本发明的实施例提供了具有排空到零能力的紧缩数据混洗指令（PSHUFB），用于有效地排序和排列任何大小的数据。在一个实施例中，以字节间隔大小在寄存器中混洗或重排数据。在混洗操作期间通过维持较大数据内的字节的相对位置，字节混洗操作排序大于字节的数据大小。此外，字节混洗操作可改变 SIMD 寄存器中数据的相对位置并也可复制数据。该 PSHUFB 指令根据第二源寄存器中的混洗控制字节的内容混洗来自第一源寄存器的字节。虽然该指令置换数据，但在本实施例的该混洗操作期间保持混洗掩码不受影响并不受改变。用于该实现的记忆存储器是“PSHUFB 寄存器 1、寄存器 2/存储器”，其中第一和第二操作数是 SIMD 寄存器。但，第二操作数的寄存器也可用存储器位置替换。第一操作数包括用于混洗的源数据。对于该实施例，用于第一操作数的寄存器也是目的地寄存器。除了改变它们的位置，根据本发明的实施例还包括将选定字节设定到零的能力。

第二操作数包括一组混洗控制掩码字节以指定混洗模式。用于选择源数据元

素的比特数是源操作数中数据元素数量的 \log_2 。例如，在一个 128 比特寄存器实施例中的字节数是 16。16 的 \log_2 是 4。因此，需要四个比特或四位字节。以下代码中的[3:0]指数表示 4 个比特。如果置位了混洗控制字节的最高有效比特（MSB），本实施例中的比特 7，则将常数零写入结果字节。如果第二操作数的字节 I 的最低 5 有效四位字节（掩码设置）包含整数 J，则混洗指令使得第一源寄存器的第 J 个字节被复制到目的地寄存器的第 I 字节位置。以下是用于 128 比特操作数的紧缩字节混洗操作的一个实施例的示例性伪码：

```
For i = 0 to 15 {
    if(SRC2[(i*8)+7] == 1)
        DEST[(i*8)+7...(i*8)+0] ← 0
    else
        index[3:0] ← SRC2[(i*8)+3 ... SRC2(i*8)+0]
        DEST[(i*8)+7...(i*8)+0] ← SRC1/DEST[(index*8+7)... (index*8+0)]
}
```

类似地，这是用于 64 比特操作数的紧缩字节混洗操作另一个实施例的示例性 10 伪码：

```
For i = 0 to 7 {
    if(SRC2[(i * 8)+7] == 1)
        DEST[(i*8)+7...(i*8)+0] ← 0
    else
        index[2:0] ← SRC2[(i*8)+2 ... SRC2(i*8)+0]
        DEST[(i*8)+7...(i*8)+0] ← SRC1/DEST[(index*8+7)... (index*8+0)]
}
```

注意，该 64 比特寄存器实施例中，当 64 比特寄存器中存在 8 个字节时使用掩码的较低 3 比特。8 的 \log_2 是 3。以上代码中的[2:0]指数表示 3 个比特。在可选实施例中，掩码中的比特数可改变以符合源数据中可用的数据元素的数量。例如， 15 需要具有较低 5 个比特的掩码来选择 256 比特寄存器中的数据元素。

当前，在 SIMD 寄存器中重排数据稍许困难和冗长。某些算法需要的用于排列用于算术操作的数据的指令超过了用于执行这些操作的指令的实际数量。通过实现根据本发明的紧缩字节混洗指令的实施例，实现数据重排所需的指令数量会急剧减少。例如，紧缩字节混洗指令的一个实施例可将一字节数据广播到 128 比特寄存 20 器的所有位置。在寄存器中广播数据常在过滤应用中使用，其中单个数据项被乘以许多系数。在没有该指令的情况下，数据字节必须从其源中过滤并被移位到最低字

节位置。随后，该单个字节必须首先作为字节被复制，随后这两个字节被再次复制以形成双字，且该双字被复制以最终形成四字。所有这些操作都可用单个紧缩混洗指令来替代。

类似地，诸如大 endian 和小 endian 格式之间变化的 128 比特寄存器中的所有 5 字节的反转可用单个紧缩混洗指令方便地进行。然而如果不使用紧缩混洗指令，即使这些相当简单的模式也需要许多指令，复杂或随机模式需要甚至更低效的指令例程。对于 SIMD 寄存器中重排随机字节的最直接的正向解法是将它们写入缓冲器，随后使用整数字节读取/写入以便将它们重排并将它们读回 SIMD 寄存器。所有这些数据处理都需要冗长的代码序列，而单个紧缩混洗指令则足够了。通过减少所需的 10 指令数量，大大地减少了生成相同结果所需的时钟周期数。本发明的实施例还使用混洗指令来用 SIMD 指令访问表中的多个值。甚至在表是寄存器大小的两倍的情况下，根据本发明的算法也允许以比如同整数操作的每个指令一数据元素更快的速度访问数据元素。

图 1A 是根据本发明一实施例的由处理器形成的示例性计算机系统的框图，该 15 处理器包括执行单元以便执行用于混洗数据的指令。根据本发明，诸如这里所述的实施例中，系统 100 包括诸如处理器 102 的组件，用于利用包含逻辑的执行单元执行混洗数据的算法。系统 100 表示基于可从 Intel Corporation of Santa Clara, California 获得的 PENTIUM®III, PENTIUM®4, Celeron®, Xeon™, Itanium®, XScale™ 和/或 StrongARM™ 的微处理器的处理系统，尽管其它系统（包括具有其 20 它微处理器的 PC、工程工作站、置顶盒等）也可使用。在一个实施例中，样品系统 100 可执行可从 Redmond, Washington 的 Microsoft Corporation 获得的 WINDOW™ 操作系统的版本，尽管其它操作系统（例如，UNIX 和 Linux）、嵌入软件和/或图形用户界面也是可以使用的。因此，本发明不限于任何特殊的硬件电路和软件的组合。

25 本改进不限于计算机系统。本发明的可选实施例可用于其它装置中，诸如手持设备和嵌入应用。手持设备的某些示例包括蜂窝电话、因特网协议装置、数字照相机、个人数字助理（PDA）和手持 PC。嵌入应用可包括微控制器、数字信号处理器（DSP）、单片系统、网络计算机（NetPC）、置顶盒、网络集线器、广域网（WAN）开关或对操作数进行整数混洗操作的任何其它系统。此外，实现某些架

构以使指令能对数个数据同时进行操作，从而提升多媒体应用的效率。随着数据的类型和量的增加，计算机及其处理器必须得到提升以按更有效的方法操作数据。

图 1A 是用根据本发明的处理器 102 形成的计算机系统 100 的框图，处理器 102 包括用于执行数据混洗算法的一个或多个执行单元 108。在单个处理器的台式或服务器系统的环境中描述本实施例，但可选实施例也可包含在多处理器系统中。
5 系统 100 是集线器架构的示例。计算机系统 100 包括处理数据信号的处理器 102。处理器 102 可以是复杂指令集计算机 (CISC) 微处理器，精简指令集计算 (RISC) 微处理器、超长指令字 (VLIW) 微处理器、实现指令集组合的处理器或诸如数字信号处理器的任何其它处理器装置。处理器 102 耦合到处理器总线 110，它能在处理器 102 和系统 100 的其它组件之间传送数据信号。系统 100 的元件执行它们为本领域技术人员所熟知的常规功能。
10

在一个实施例中，处理器 102 包括一级 (L1) 内部高速缓存存储器 104。根据该架构，处理器 102 可具有单个内部高速缓存或多级内部高速缓存。或者，在另一实施例中，高速缓存存储器可位于处理器 102 外部。根据特殊实现和需要，其它实施例还可包括内部和外部高速缓存的组合。寄存器堆 106 可在各种寄存器中存储不同类型的数据，这些寄存器包括整数寄存器、浮点寄存器、状态寄存器和指令指针寄存器。
15

包括执行整数和浮点运算的逻辑的执行单元 108 也在处理器 102 中。处理器 102 还包括存储特定宏指令的微码的微码 (ucode) ROM。对于这种实施例，执行
20 单元 108 包括处理紧缩指令集 109 的逻辑。在一个实施例中，紧缩指令集 109 包括用于组织数据的紧缩混洗指令。通过在通用处理器 102 的指令集中包含紧缩指令集 109 连同执行这些指令的相关电路，可利用通用处理器 102 中的紧缩数据执行许多多媒体应用所使用的操作。因此，可通过将处理器数据总线的总宽度用于对紧缩数据进行操作而加速和更有效地执行许多多媒体应用。这能消除对在处理器的数据总
25 线上传递较小单元数据以便每次对一个数据单元执行一个或多个操作的需要。

执行单元 108 的可选实施例也可用于微控制器、嵌入处理器、图形装置、DSP 和其它类型的逻辑电路中。系统 100 包括存储器 120。存储器 120 可以是动态随机存取存储器 (DRAM) 装置，静态随机存取存储器 (SRAM) 装置、闪存装置或其它存储器装置。存储器 120 可存储由可通过处理器 102 执行的数据信号表示的指令

和/或数据。

系统逻辑芯片 116 耦合到处理器总线 110 和存储器 120。所示实施例中的系统逻辑芯片 116 是存储器控制器集线器 (MCH)。处理器 102 可经由处理器总线 110 与 MCH116 通信。MCH116 将较高的带宽存储器路径 118 提供给存储器 120，用于指令和数据存储并用于图形命令、数据和纹理的存储。MCH116 在处理器 102、存储器 120 和系统 100 中的其它组件之间引导数据信号并在处理器总线 110、存储器 120 和系统 I/O122 之间桥接数据信号。在某些实施例中，系统逻辑芯片 116 可提供图形端口，用于耦合到图形控制器 112。MCH116 通过存储器接口 118 与存储器 120 耦合。图形卡 112 通过加速图形端口 (AGP) 互连 114 与 MCH116 耦合。

系统 100 使用专有集线器接口总线 122 将 MCH116 耦合到 I/O 控制器集线器 (ICH)130。ICH130 经由本地 I/O 总线提供到某些 I/O 装置的直接连接。本地 I/O 总线是用于将外围设备连接到存储器 120、芯片组和处理器 102 的高速 I/O 总线。某些示例是音频控制器、固件集线器 (快闪 BIOS) 128、无线收发器 126、数据存储装置 124、包含用户输入和键盘接口的传统 I/O 控制器、诸如通用串行总线 (USB) 的串行扩展端口以及网络控制器 134。数据存储装置 124 可以包括硬盘驱动器、软盘驱动器、CD-ROM 装置、闪存装置或其它海量存储装置。

对于系统的另一实施例，用混洗指令执行算法的执行单元可与单片系统一同使用。单片系统的一个实施例由处理器和存储器构成。一个这种系统的存储器是闪存。闪存可与处理器和其它系统组件位于同一管芯上。此外，诸如存储器控制器或图形控制器的其它逻辑块也可位于单片系统上。

图 1B 示出了实现本发明原理的数据处理系统 140 的可选实施例。数据处理系统 140 的一个实施例是具有 Intel XScaleTM 技术的 Intel®Personal Internet Client Architecture (Intel®PCA) 应用处理器 (如 developer.intel.com 处所描述的)。本领域的熟练技术人员易于理解，这里所述的实施例可与可选处理系统一起使用而不背离本发明的范围。

计算机系统 140 包括能执行包括混洗的 SIMD 操作的处理核心 159。对于一个实施例，处理核心 159 表示任何类型的架构的处理单元，包括但不限于 CISC、RISC 或 VLIW 型架构。处理核心 159 还适合于以一种或多种处理技术制造，并通过在其可读媒体上充分详细地表示，适合于促进所述制造。

处理核心 159 包括执行单元 142、一组寄存器堆 145 和解码器 144。处理核心 159 还包括附加电路（未示出），这对于本发明的理解是非必要的。执行单元 142 用于执行处理核心 159 接收的指令。除了识别典型的处理器指令，执行单元 142 可识别用于执行对紧缩数据格式的操作的紧缩指令集 143 中的指令。紧缩指令集 143 包括用于支持混洗操作的指令，还可包括其它紧缩指令。执行单元 142 通过内部总线耦合到寄存器堆 145。寄存器堆 145 表示用于存储信息（包括数据）的处理核心 159 上的存储区域。如前所述，可以理解，用于存储紧缩数据的存储区域是非关键的。执行单元 142 与解码器 144 耦合。解码器 144 用于将处理核心 159 接收的指令解码成控制信号和/或微码入口点。响应于这些控制信号和/或微码入口点，执行单元 142 执行合适的操作。

处理核心 159 与总线 141 耦合，以便与各种其它系统装置通信，包括但不限于同步动态随机存取存储器（SDRAM）控制 146、静态随机存取存储器（SRAM）控制 147、猝发闪存接口 148、个人计算机存储器卡国际协会（PCMCIA）/紧致闪存（CF）卡控制 149、液晶显示器（LCD）控制 150、直接存储器存取（DMA）控制器 151 和可选总线主接口 152。在一个实施例中，数据处理系统 140 还可包括 I/O 桥路 154，用于经由 I/O 总线 153 与各种 I/O 装置通信。这种 I/O 装置可包括但不限于，通用异步接收器/发送器（UART）155、通用串行总线（USB）156、蓝牙无线 UART157 和 I/O 扩展接口 158。

数据处理系统 140 的一个实施例提供用于移动、网络和/或无线通信以及能执行包含混洗操作的 SIMD 操作的处理核心 159。处理核心 159 可用包含离散变换、压缩/解压缩技术以及调制/解调（MODEM）函数的各种音频、视频、成像和通信算法编程，所述离散变换诸如 Walsh-Hadamard 变换、快速傅里叶变换（FFT）、离散余弦变换（DCT）和它们各自的逆变换；所述压缩/解压缩技术诸如色空间变换、视频编码运动估计或视频解码运动补偿；所述调制/解调函数诸如脉冲编码调制（PCM）。

图 1C 示出了能执行 SIMD 混洗操作的数据处理系统的又一个可选实施例。根据一个可选实施例，数据处理系统 160 可包括主处理器 166、SIMD 协处理器 161、高速缓存存储器 167 和输入/输出系统 168。输入/输出系统 168 可任选地耦合到无线接口 169。SIMD 协处理器 161 能执行包含数据混洗的 SIMD 操作。处理核心 170

能适合于以一种或多种处理技术制造，并通过在机器可读媒体上的充分详细的表示，适合于促进包含处理核心 170 的全部或部分数据处理系统 160 的制造。

对于一个实施例，SIMD 协处理器 161 包括执行单元 162 和一组寄存器堆 164。主处理器 165 的一个实施例包括用于识别包含由执行单元 162 执行的 SIMD 混洗指令的指令集 163 的指令的解码器 165。对于可选实施例，SIMD 协处理器 161 还包括解码指令集 163 的指令的至少部分解码器 165B。处理核心 170 还包括对于本发明的理解非必要的附加电路（未示出）。

操作中，主处理器 166 执行数据处理指令流，它控制包括与高速缓存存储器 167 和输入/输出系统 168 的交互作用的普通型数据处理操作。SIMD 协处理器指令被嵌入数据处理指令流内。主处理器 166 的解码器 165 识别作为应由附加的 SIMD 协处理器 161 执行的类型的这些 SIMD 协处理器指令。因此，主处理器 166 在协处理器总线 166 上发出这些 SIMD 协处理器指令（或者表示 SIMD 协处理器指令的控制信号），且它们由任何附加的 SIMD 协处理器接收。在这种情况下，SIMD 协处理器 161 将接受和执行打算送给它的任何接收到的 SIMD 协处理器指令。

经由无线接口 169 接收用于由 SIMD 协处理器指令处理的数据。对于一个示例，语音通信可按数字信号形式接收，它可通过 SIMD 协处理器指令处理以再生表示语音通信的数字音频采样。对于另一示例，压缩的音频和/或视频可按数字比特流形式接收，它们通过 SIMD 协处理器指令处理以再生数字音频采样和/或运动视频帧。对于处理核心 170 的一个实施例，主处理器 166 和 SIMD 协处理器 161 被集成入包含执行单元 162、一组寄存器堆 164 和用于识别包含 SIMD 混洗指令的指令集 163 的指令的解码器 165 的单个处理核心 170。

图 2 是根据本发明的包含用于执行混洗操作的逻辑电路的一个实施例的处理器 200 微架构的框图。如同在以上的讨论中，混洗操作也可称作紧缩数据混洗操作和紧缩混洗指令。对于混洗指令的一个实施例，该指令可用字节颗粒度混洗紧缩数据。该指令还可称作 PSHUFB 或紧缩混洗字节。在其它实施例中，混洗指令也可被执行以对具有字、双字、四倍字等大小的数据元素进行操作。按次序的前端 201 是处理器 200 的一部分，它取出要执行的宏指令并在处理器管道中预备以后使用。该实施例的前端 201 包括数个单元。指令预取器 226 从存储器中取出宏指令并将它们馈送给指令解码器 228，该指令解码器将宏指令解码成机器知道如何执行的称作

微指令或微操作（也称作 micro op 或 uops）的基元（primitives）。跟踪高速缓存 230 获得被解码的微操作并将它们组合成微操作队列 234 中的程序有序的序列或踪迹，用于执行。当跟踪高速缓存 230 遇到复杂的宏指令时，微码 ROM232 提供完成操作所需的微操作。

5 许多宏指令被转换成单个微操作，而其它宏指令需要数个微操作来完成全部操作。在该实施例中，如果需要超过 4 个微操作来完成一个宏指令，则解码器 228 访问微码 ROM232 来实施该宏指令。对于一个实施例，一个紧缩混洗指令可被解码成应用于在指令解码器 228 处处理的较少数量的微操作。在另一实施例中，若需要一定数量的微操作来完成操作，则用于紧缩数据混洗算法的指令可存储在微码 10 ROM232 内。跟踪高速缓存 230 引用入口点可编程逻辑阵列（PLA）来确定正确的微指令指针，用于读取微码 ROM232 中用于混洗算法的微码序列。在微码 ROM232 为当前宏指令完成排序微操作后，机器的前端 201 继续从跟踪高速缓存 230 取出微操作。

某些 SIMD 和其它多媒体类型的指令被认为是复杂指令。多数浮点相关的指令也是复杂指令。这样，当指令解码器 228 遇到复杂宏指令时，在合适位置存取微码 ROM232 以检索用于该宏指令的微码序列。执行宏指令所需的各种微操作被传递给无序执行引擎 203，用于在合适的整数和浮点执行单元处执行。

在无序执行引擎 203 处准备微指令用于执行。无序执行逻辑具有一些缓冲器，用于平滑和重排微指令流以便在它们沿管道下降和被调度用于执行时优化性能。分配器逻辑分配每个微操作执行所需的机器缓冲器和资源。寄存器重命名逻辑将逻辑寄存器重命名为寄存器堆中的条目。在指令调度器（存储器调度器，快速调度器 202，慢速/普通浮点调度器 204，和简单浮点调度器 206）前，分配器还为两个微操作队列（一个用于存储器操作，一个用于非存储器操作）之一中的每个微操作分配条目。微操作调度器 202、204 和 206 基于它们依赖的输入寄存器操作数源的预备状态以及微操作完成操作所需的执行资源的可用性来确定微操作何时准备执行。本实施例的快速调度器 202 可在每半个主时钟周期上进行调度，而其它调度器只能每个主处理器时钟周期调度一次。调度器对分派端口进行裁决，以调度用于执行的微操作。

寄存器堆 208 和 210 位于调度器 202、204、206 和执行块 211 中的执行单元

212、214、216、218、220、222、224 之间。存在分别用于整数和浮点运算的分开的寄存器堆 208、210。本实施例中的每个寄存器堆 208、210 还包括旁路网络，它将未被写入寄存器堆的刚完成的结果旁路或传递到新依赖微操作。整数寄存器堆 208 和浮点寄存器堆 210 还能彼此通信数据。对于一个实施例，整数寄存器堆 208
5 被分成两个分开的寄存器堆，其中一个用于数据的低位 32 比特而另一个用于数据的高位 32 比特。由于浮点指令通常具有宽度从 64 到 128 比特的操作数，一个实施例的浮点寄存器堆 210 具有 128 比特宽的条目。

执行块 211 包含实际执行指令的执行单元 212、214、216、218、220、222、
224。该部分包括寄存器堆 208、210，它们存储微指令执行所需的整数和浮点数据
10 操作数值。本实施例的处理器 200 由一些执行单元构成：地址生成单元（AGU）
212、AGU214、快速 ALU216、快速 ALU218、慢速 ALU220、浮点 ALU222、浮
点移动单元 224。对于该实施例，浮点执行块 222、224 执行浮点、MMX、SIMD
和 SSE 操作。本实施例的浮点 ALU222 包括浮点除法器，以执行除法、平方根和
余数微操作。对于本发明的实施例，涉及浮点值的任何动作都随浮点硬件出现。例
15 如，整数格式和浮点格式之间的转换涉及浮点寄存器堆。类似地，浮点除法运算发
生于浮点除法器处。另一方面，用整数硬件资源处理非浮点数和整数类型。简单的
非常频繁的 ALU 操作转到高速 ALU 执行单元 216、218。该实施例的快速 ALU216、
218 能以半个时钟周期的有效等待时间执行快速操作。对于一个实施例，多数复杂的
整数操作转到慢速 ALU220，因为慢速 ALU220 包括用于长等待时间类型操作的
20 整数执行硬件，诸如乘法器、移位、标记逻辑和分支处理。存储器加载/存储操作
由 AGU212、214 执行。对于该实施例，整数 ALU216、218、220 在对 64 比特数
据操作数执行整数操作的环境下描述。在可选实施例中，可实现 ALU216、218、
220 以支持包括 16、32、128、256 等的各种数据比特。类似地，可实现浮点单元
222、224 以支持具有各种宽度的比特的操作数范围。对于一个实施例，浮点单元
25 222、224 可联系 SIMD 和多媒体指令对 128 比特宽的紧缩数据操作数进行操作。

在该实施例中，在父加载完成执行前，微操作调度器 202、204、206 分派依
赖性操作。在处理器 200 中推测性调度和执行微操作时，处理器 200 还包括处理存
储器未命中的逻辑。如果数据加载在数据高速缓存中未命中，则管道中可能有依赖
性操作在飞行中（in flight），它们将暂时错误的数据留给调度器。重放机制跟踪

并重新执行使用错误数据的指令。仅依赖性操作需要被重放并允许一些非依赖性的操作完成。处理器的一个实施例的调度器和重放机制还被设计成获取用于混淆操作的指令序列。

术语“寄存器”这里用于表示用作标识操作数的宏指令的一部分的板载处理器存储位置。换句话说，这里引用的寄存器是可从处理器外部看见的（从程序员的观点）。但，实施例的寄存器不应在意义上限制于特殊类型电路。相反，实施例的寄存器只需要能存储和提供数据，并执行这里所述的功能。这里描述的寄存器可利用任何数量的不同技术由处理器内的电路实现，诸如专用物理寄存器、使用寄存器重命名的动态分配的物理寄存器、专用和动态分配的物理寄存器的组合等等。在一个实施例中，整数寄存器存储 32 比特的整数数据。一个实施例的寄存器堆还包括用于紧缩数据的 8 个多媒体 SIMD 寄存器。对于以下的讨论，寄存器被理解为被设计保持紧缩数据的数据寄存器，诸如来自 Santa Clara, California 的 Intel Corporation 的具有 MMX 技术的微处理器中的 64 比特宽 MMXTM 寄存器（某些情况下也称作‘mm’寄存器）。在整数和浮点形式中可用的这些 MMX 寄存器可与附随 SIMD 和 SSE 指令的紧缩数据元件一起操作。类似地，与 SSE2 技术有关的 128 比特宽 XMM 寄存器也可用于保持这种紧缩数据操作数。在该实施例中，在存储紧缩数据和整数数据过程中，寄存器不需要在两种数据类型之间区分。

在以下附图的示例中，描述了许多数据操作数。为简单起见，从字母 A 起按字母顺序标注初始源数据段，其中 A 位于最低地址处且 Z 将位于最高地址处。因此，A 初始地在地址 0 处，B 在地址 1 处，C 在地址 3 处，等等。概念上，如对于一个实施例在紧缩字节混洗中，混洗操作需要从第一操作数混洗数据段并将一个或多个源数据元素重排入由第二操作数中的一组掩码指定的模式（pattern）中。因此，混洗可旋转或将数据元素的一部分或全部完全重排成任何期望的顺序。此外，任何特殊数据元素或一些数据元素可在结果中被复制或广播。根据本发明的混洗指令的实施例包括排空到零功能，其中用于每个特殊数据元素的掩码能使得该数据元素位置在结果中被清零。

图 3A-C 示出了根据本发明各种实施例的混洗掩码。本例中示出了由多个个别数据元素 311、312、313、314 构成的紧缩数据操作数 310。在包含一组掩码以指示用于另一操作数的相应紧缩数据元素的混洗模式的紧缩数据操作数环境中描述

本例的紧缩操作数 310。因此，紧缩操作数 310 的数据元素 311、312、313、314 中的每一个中的掩码都指定结果的相应数据元素位置中的内容。例如，数据元素 311 在最左面的数据元素位置中。数据元素 311 中的掩码将指定哪个数据应被混洗或设置于混洗操作结果的最左面的数据元素位置中。类似地，数据元素 312 是次最 5 左面的数据元素位置。数据元素 312 中的掩码将指定哪个数据应被设置于结果的次最左面的数据元素位置。对于该实施例，包含混洗掩码的紧缩操作数中的每个数据元素都具有与紧缩结果中的数据元素位置的一对一对应。

图 3A 中，数据元素 312 被用于描述一个实施例的示例混洗掩码的内容。用于一实施例的混洗掩码 318 由三部分构成：‘设定到零标记’字段 315、‘保留’字 10 段 316 和‘选择比特’字段 317。‘设定到零标记’字段 315 指示本掩码指定的结果数据元素位置是否应被清零，或换句话说，是否应用零（0）值替换。在一个实施例中，‘设定到零标记’字段是支配的，其中如果‘设定到零标记’字段 315 被设定，则忽略掩码 318 中的其余字段且结果数据元素位置用‘0’填充。‘保留’字 15 段 316 包括一个或多个比特，可以或不可以在可选实施例中使用或者可以保留用于将来或特殊的使用。该混洗掩码 318 的‘选择比特’字段 317 为紧缩结果中的相应数据元素位置指定数据源。

对于紧缩数据混洗指令的一个实施例，一个操作数由一组掩码构成且另一操作数由一组紧缩数据元素构成。这两种操作数的大小是相同的。根据操作数中数据元素的数量，需要变动数量的选择比特从用于紧缩结果中的设置的第二紧缩数据操 20 作数中选择个别数据元素。例如，在具有紧缩字节的 128 个比特的源操作数的情况下，由于 16 个字节的数据元素可用于选择，所以需要至少 4 个选择比特。基于掩码的选择比特所指示的值，来自源数据操作数的合适数据元素被置于该掩码的相应数据元素位置。例如，数据元素 312 的掩码 318 对应于次最左面数据元素位置。如果该掩码 318 的选择比特 317 包含值‘X’，则来自源数据操作数中数据元素位置‘X’ 25 的数据元素被混洗入结果中的次最左面数据元素位置。但如果‘设定到零标记’字段 315 被设定，则结果中的次最左面数据元素位置用‘0’代替且忽略选择比特 317 的指定。

图 3B 示出了用字节尺寸的数据元素和 128 比特宽的紧缩操作数操作的一个实施例的掩码 328 的结构。对于该实施例，因为存在 16 个可能的数据元素选择，‘设

定到零’字段 325 由比特 7 构成且‘选择’字段 327 由比特 3 到 0 构成。在该实施例中不使用比特 6 到 4 且它们留在‘保留’字段 326 中。在另一实施例中，‘选择’字段 327 中使用的比特数量可以按需要增加或减少，以适应源数据操作数中可用的可能数据元素选择的数量。

- 5 图 3C 示出了用于与字节大小数据元素和 128 比特宽的紧缩操作数一起操作但具有多个数据元素源的另一实施例的掩码 338 的结构。在该实施例中，掩码 338 由‘设定到零’字段 335、‘源 (src) 选择’字段 336 和‘选择’字段 337 构成。
‘设定到零’字段 335 和‘选择’字段 337 类似于以上描述运作。本实施例的‘源选择’字段 336 指示应从哪个数据源获得由选择比特指定的数据操作数。例如，相
10 同一组掩码可与诸如多个多媒体寄存器的多个数据源一起使用。每个源多媒体寄存器都被分配一数值且‘源选择’字段 336 中的值指向这些源寄存器之一。根据‘源选择’字段 336 的内容，从合适的数据源中选择所选数据元素，用于置于紧缩结果中的相应数据元素位置。

- 图 4A 示出了根据本发明实施例的多媒体寄存器中的各种紧缩数据类型表示。
15 图 4A 示出了用于 128 比特宽的操作数的紧缩字节 410、紧缩字 420 和紧缩双字
(dword) 430 的数据类型。该示例的紧缩字节格式 410 是 128 比特长并包含十六个紧缩字节数据元素。这里字节被定义为 8 比特数据。每个字节数据元素的信息被存入用于字节 0 的比特 7 到比特 0、用于字节 1 的比特 15 到比特 8、用于字节 2 的比特 23 到比特 16 并最后用于字节 15 的比特 120 到比特 127。因此，寄存器中
20 使用所有可用的比特。该存储结构增加了处理器的存储效率。此外，在 16 个数据元素被访问的情况下，现在可以并行地对十六各数据元素进行一个操作。

通常，数据元素是与其它相同长度的数据元素一起被存入操作数（单个寄存器或存储器位置）的单件数据。在关于 SSE2 技术的紧缩数据序列中，操作数(XMM 寄存器或存储器位置)中存储的数据元素数量是 128 比特除以各数据元素的比特长度。类似地，关于 MMX 和 SSE 技术的紧缩数据序列中，操作数(MMX 寄存器或存储器位置)中存储的数据元素数量是 64 比特除以各数据元素的比特长度。该示例的紧缩字格式 420 是 128 比特长并包含 8 个紧缩字数据元素。每个紧缩字包含 16 比特的信息。图 4A 的紧缩双字格式 430 是 128 比特长并包含 4 个紧缩双字数据元素。每个紧缩双字数据元素都包含 32 比特的信息。紧缩四倍字是 128 比特长并

包含两个紧缩四倍字数据元素。

图 4B 示出了可选的寄存器中数据存储格式。每个紧缩数据可包括超过一个独立的数据元素。示出了三种紧缩数据格式；紧缩半 441、紧缩单 442 和紧缩双 443。
5 紧缩半 441、紧缩单 442 和紧缩双 443 的一个实施例包含定点数据元素。对于可选实施例，紧缩半 441、紧缩单 442 和紧缩双 443 中的一个或多个可包含浮点数据元素。紧缩半 441 的一个可选实施例是 128 比特长，包含 8 个 16 比特数据元素。紧缩单 442 的一个实施例是 128 比特长并包含 4 个 32 比特数据元素。紧缩双 443 的一个实施例是 128 比特长并包含 2 个 64 比特数据元素。可以理解，这种紧缩数据格式可进一步扩展到其它寄存器长度，例如扩展到 96 比特、160 比特、192 比特、
10 224 比特、256 比特或以上。

图 4C 示出了操作编码（操作码）格式 460 的一实施例，它具有 32 个或更多比特，且寄存器/存储器操作数寻址模式与万维网 (www.intel.com/design/litcentr) 处的可从 Intel Corporation, Santa Clara, CA 获得的“IA-32 Intel Architecture Software Developer’s Manual Volume 2: Instruction Set Reference” 中所描述的操作码格式类型一致。混洗操作的类型可通过一个或多个字段 461 和 462 被编码。可标识直至每个指令两个操作数位置，包括直至两个源操作数标识符 464 和 465。对于混洗指令的一个实施例，目的地操作数标识符 466 与源操作数标识符 464 相同。对于可选实施例，目的地操作数标识符 466 与源操作数标识符 465 相同。因此，对于混洗操作的实施例，由源操作数标识符 464 和 465 标识的源操作数之一被混洗操作的结果重写。对于混洗操作的一个实施例，操作数标识符 464 和 465 可用于标识 64 比特的源和目的地操作数。
15
20

图 4D 示出了具有 40 个以上比特的另一可选操作编码（操作码）格式 470。操作码格式 470 与操作码格式 460 一致并包括任选前缀字节 478。混洗操作类型可由字段 478、471 和 472 中的一个或多个编码。高达每个指令两个操作数位置可由源操作数标识符 474 和 475 以及通过前缀字节 478 标识。对于混洗指令的一个实施例，前缀字节 478 可用于标识 128 比特的源和目的地操作数。对于混洗操作的一个实施例，目的地操作数标识符 476 与源操作数标识符 474 相同。对于可选实施例，目的地操作数标识符 476 和源操作数标识符 475 相同。因此，对于混洗操作的实施例，由源操作数标识符 474 和 475 标识的源操作数之一由混洗操作的结果重写。操
25

作码格式 460 和 470 允许部分由 MOD 字段 463 和 473 以及由任选的比例指标基和位移字节指定的寄存器到寄存器、存储器到寄存器、寄存器接存储器、寄存器接寄存器、寄存器接邻接（immediate）、寄存器到存储器寻址。

接着转到图 4E，在一些可选实施例中，通过协处理器数据处理（CDP）指令 5 进行 64 比特的单指令多数据（SIMD）算术操作。操作编码（操作码）格式 480 描述了具有 CDP 操作码字段 482 和 489 的一个这种 CDP 指令。对于混洗操作的可选实施例，CDP 指令的类型可由字段 483、484、487 和 488 中的一个或多个编码。可标识达每个指令三个操作数位置，包括达两个源操作数标识符 485 和 490 以及一个目的地操作数标识符 486。协处理器的一个实施例可在 8、16、32 和 64 比特值 10 上操作。对于一个实施例，混洗操作在定点或整数数据元素上操作。在一些实施例中，利用条件字段 481，可有条件地执行混洗指令。对于某些混洗指令，源数据大小可由字段 483 编码。在混洗指令的一些实施例中，可对 SIMD 字段进行零（Z）、负（N）、进位（C）和溢出（V）检测。对于一些指令，可通过字段 484 编码饱和类型。

15 图 5 是根据本发明基于混洗掩码对数据操作数进行混洗操作的逻辑的一个实施例的框图。具有本实施例的设定到零能力的用于混洗操作的指令（PSHUFBD）以两段信息第一（掩码）操作数 510 和第二（数据）操作数 520 开始。对于以下讨论，掩码（MASK）、数据（DATA）和结果（RESULTANT）一般称作操作数或数据块，但非限制于此，并还包括寄存器、寄存器文件和存储器位置。在一个实施例中，
20 混洗 PSHUFBD 指令被解码为一个微操作。在可选实施例中，该指令可解码为各种数量的微操作，以便对数据操作数进行混洗操作。对于该示例，操作数 510、520 是具有字节宽数据元素的源寄存器/存储器中存储的 128 比特宽的信息段。在一个实施例中，操作数 510 和 520 保存于 128 比特长的 SIMD 寄存器中，诸如 128 比特的 SSE2 XMM 寄存器。但是，操作数 510 和 520 中的一个或两者也可从存储器位置处加载。对于一个实施例，结果（RESULTANT）540 也是 MMX 或 XMM 数据
25 寄存器。此外，RESULTANT540 也可以是与源操作数之一相同的寄存器或存储器位置。根据特殊实现，操作数和寄存器可以是其它宽度，诸如 32、64 和 256 比特，并具有字、双字或四字大小的数据元素。该示例中的第一操作数 510 由一组 16 个掩码构成（按十六进制格式）：0x0E、0x0A、0x09、0x8F、0x02、0x0E、0x06、

0x06、0x06、0xF0、0x04、0x08、0x08、0x06、0x0D 和 0x00。每个个别掩码将指定结果 540 中其相应数据元素位置的内容。

第二操作数 520 由十六个数据区段构成: P,O,N,M,L,K,J,I,H,G,F,E,D,C,B,A。

第二操作数 520 中的每个数据区段也用十六进制的数据元素位置值标记。这里的数
5 据区段是等长的且各自由单个字节(8 比特)数据构成。如果每个数据元素是字(16
比特)、双字(32 比特)或四字(64 比特), 则 128 比特的操作数将分别具有 8
个字宽、4 个双字宽或 2 各四字宽的数据元素。但是, 本发明的其它实施例可以其
它大小的操作数和数据区段运作。本发明的实施例不限于特殊长度的数据操作数、
数据区段或移位计数, 并可针对每个实现适当调整尺寸。

10 操作数 510 和 520 可驻留在寄存器或存储器位置或者寄存器文件或混合中。
将数据操作数 510 和 520 与混洗指令一起发送到处理器中的执行单元的混洗逻辑
530。在混洗指令达到执行单元时, 该指令应在处理器管道中较早地被解码。因此,
混洗指令可以是微操作(uop)或某些其它解码格式的形式。对于该实施例, 在混
洗逻辑 530 处接收两个数据操作数 510 和 520。混洗逻辑 530 基于掩码操作数 510
15 中的值从源数据操作数 520 中选择数据元素并将所选数据元素排列/混洗到结果
540 中的合适位置。混洗逻辑 530 还将所指定的结果 540 中的给定数据元素位置清
零。这里, 结果 540 由十六个数据区段构成: O,K,J,'0',C,O,G,G,F,'0',E,I,I,G,N 和 A。

20 这里用一些数据元素描述混洗逻辑 530 的操作。对于掩码操作数 510 中最左
边的数据元素位置的混洗掩码是 0x0E。混洗逻辑 530 翻译以上图 3A-C 中描述的
掩码的各字段。在这种情况下, 不设置“设定到零”字段。包括较低 4 个比特或四
位字节的选择字段具有‘E’的十六进制值。混洗逻辑 530 将数据操作数 520 的数
据元素位置‘0xE’中的数据 O 混洗到结果 540 的最左边的数据元素位置。类似地,
掩码操作数 510 中的次最左边的数据元素位置处的掩码是 0x0A。混洗逻辑 530 翻
译用于该位置的掩码。该选择字段具有‘A’的十六进制值。混洗逻辑 530 将数据
25 操作数 520 的数据元素位置‘0xA’中的数据 K 复制到结果 540 的次最左边的数据
元素位置。

本实施例的混洗逻辑 530 也支持混洗指令的排空到零功能。从掩码操作数 510
的左边起第四个数据元素位置处的混洗掩码是 0x8F。混洗逻辑 510 识别出设置了
‘设定到零’字段, 如掩码的比特 8 处的‘1’所指示的。作为响应, 排空到零的

指令胜过选择字段，且混洗逻辑 510 忽略该掩码的选择字段中的十六进制值 ‘F’。将 ‘0’ 置入从结果 540 中的左面起相应的第四数据元素位置。对于该实施例，混洗逻辑 530 为每个掩码评估 ‘设定到零’ 和选择字段并不在意掩码中那些字段之外存在的其它比特，诸如保留比特或源选择字段。对掩码操作数 510 中的整组掩码重 5 复混洗掩码的该处理和数据混洗。对于一个实施例，掩码都被并行处理。在另一实施例中，每次可一起处理掩码组和数据元素的某些部分。

采用本混洗指令的实施例，操作数中的数据元素可按各种方式被重排。此外，来自特殊数据元素的某些数据可在多个数据元素位置处被重复或甚至传播到每个位置。例如，第四和第五掩码两者都具有 0x08 的十六进制值。结果，数据操作数 10 520 的数据元素位置 0x8 处的数据 I 被混洗入从结果 540 的右侧起的第四和第五个数据元素位置。采用设定到零功能，混洗指令的实施例可迫使结果 540 中的任何数据元素位置为 ‘0’ 。

根据特殊实现，每个混洗掩码都可用于指明结果中单个数据元素位置的内容。如在本示例中，每个单个字节宽的混洗掩码对应于结果 540 中字节宽的数据元素位 15 置。在另一实施例中，多个掩码的组合可一同被用于指示数据元素块。例如，两个字节宽的掩码可一起用于指明字宽的数据元素。混洗掩码不限于为字节宽，而是可以是具体实现所需的任何其它尺寸。类似地，数据元素和数据元素位置可拥有字节之外的其它间隔尺寸。

图 6 是根据本发明的用于执行数据混洗操作的电路 600 的一个实施例的框图。该实施例的电路包括多路复用结构，以便基于第二操作数的解码混洗掩码从第一源操作数中选择正确的结果字节。这里的源数据操作数由上紧缩数据元素和下紧缩数据元素构成。该实施例的多路复用结构与用于实现其它紧缩指令的其它多路复用结构相比相对比较简单。结果，本实施例的多路复用结构不引入任何新的临界计时路径。本实施例的电路 600 包括混洗掩码块、用于保持来自源操作数的下/上紧缩数据元 25 素的块、用于数据元素的初始选择的第一多个 8 至 1 (8: 1) 复用器、用于上和下数据元素的选择的其它多个 3 至 1 (3: 1) 复用器、复用器选择&零逻辑以及许多控制信号。为了简便，图 6 中示出了有限数量的 8: 1 和 3: 1 复用器并用点表示。但是，它们的功能类似于所示出的那些并通过以下描述加以理解。

在本示例的混洗操作期间，该混洗处理电路 600 处接收两个操作数：具有一

组紧缩数据元素的第一操作数和具有一组混洗掩码的第二操作数。混洗掩码被传递到混洗掩码块 602。这组混洗掩码在复用器选择和零逻辑块 604 处被解码以生成各种选择信号（选择 A606，选择 B608，选择 C610）和设定到零信号（零）611。这些信号用于控制将结果 632 接合在一起过程中复用器的操作。

5 对于该示例，掩码操作数和数据操作数都是 128 比特长，并各自用 16 字节大小的数据区段紧缩。在这种情况下，各信号上示出的值 N 是 16。在该实施例中，数据元素被分成一组下和上紧缩数据元素，每个都具有 8 个数据元素。这允许在数据元素选择期间使用较小的 8: 1 复用器而非 16: 1 复用器。这些下和上组的紧缩数据元素分别保存在下和上存储区域 612 和 622 处。从下数据组开始，八个数据元
10 素中的每一个经由诸如路由线 614 的一组线路被发送到第一组 16 个单个的 8: 1 复用器 618A-D。用 N 个选择 A 信号 606 之一控制 16 个 8: 1 复用器 618A-D 中的每一个。根据其选择 A606 的值，该复用器将输出 8 个下数据元素 614 之一用于进一步处理。由于可能将任何下数据元素混洗入十六个结果数据元素位置中的任一个，所以存在用于该组下紧缩数据元素的 16 个 8: 1 复用器。十六个 8: 1 复用器
15 中的每一个都用于十六个结果数据元素位置之一。类似地，十六个 8: 1 复用器提供用于上紧缩数据元素。8 个上数据元素被发送给第二组 16 个 8: 1 复用器 624A-D 中的每一个。用 N 个选择 B 信号 608 之一控制十六个 8: 1 复用器 624A-D 中的每一个。基于其选择 B608 的值，该 8: 1 复用器将输出 8 个上数据元素 616 之一用于进一步处理。

20 十六个 3: 1 复用器 628A-D 中的每一个都对应于结果 632 中的数据元素位置。来自十六个下数据复用器 618A-D 的十六个输出 620A-D 被路由到一组十六个 3: 1 上/下选择复用器 628A-D，作为来自上数据复用器 624A-D 的输出 626A-D。这些 3: 1 选择复用器 628D-D 中的每一个都从复用器选择&零逻辑 604 接收其自己的选择 C610 和零 611 信号。对于该 3: 1 复用器的选择 C610 上的值将指示复用器是否将输出来自下数据组或来自上数据组的选择数据操作数。到每个 3: 1 复用器的控制信号零 611 将指示该复用器是否应迫使其输出到零（‘0’）。对于该实施例，控制信号零 611 代替选择 C610 上的选择并在结果 632 中迫使对数据元素位置的输出到 ‘0’ 。

例如，3: 1 复用器 628A 接收来自 8: 1 复用器 618A 的选择的下数据元素 620A

以及来自 8: 1 复用器 624A 的选择的上数据元素 626A，用于该数据元素位置。选择 C610 控制其输出 630A 哪个数据元素将混洗入结果 632 中它管理的数据元素位置。但是，如果到复用器 628A 的信号零 611 是有效的，指示该数据元素位置的混洗掩码声明‘0’是期望的，则复用器输出 630A 是‘0’且不使用数据元素输入 620A、
5 626A。混洗操作的结果 632 由来自 16 个 3: 1 复用器 628A-D 的输出 630A-D 构成，其中每个输出都对应于特殊的数据元素位置并且是数据元素或‘0’。在该示例中，每个 3: 1 复用器输出都是字节宽且结果是由 16 个紧缩字节数据构成的数据块。

图 7 示出了根据本发明一个实施例的对字节宽数据元素的数据混洗操作。这是指令“PSHUFB 数据，掩码”的示例。注意，设定用于掩码 701 的字节位置 0x6 和 0xC 的混洗掩码的最高有效位，以使用于这些位置的结果 741 中的结果为零。
10 在该示例中，考虑到指定地址的一组掩码 701 其中来自源操作数 721 的各数据元素将存入目的地寄存器 741，源数据被组织为目的地数据存储装置 721，它在一个实施例中也是源数据存储装置 721。该示例中，如结果 741 所作的，两个源操作数（掩码 701 和数据 721）各自由十六个紧缩数据元素构成。在本实施例中，所涉及的每
15 个数据元素是 8 比特或字节宽的。因此，掩码 701、数据 721 和结果 741 数据块分别是 128 比特长。此外，这些数据块可驻留在存储器或寄存器中。对于一个实施例，掩码的排列基于期望的数据处理操作，例如可包括过滤操作或卷积操作。

如图 7 所示，掩码操作数 701 包括数据元素，它具有混洗掩码：0x0E 702、0x0A
20 703、0x09 704、0x8F 705、0x02 706、0x0E 707、0x06 708、0x06 709、0x05 710、
0xF0 711、0x04 712、0x08 713、0x08 714、0x06 715、0x0D 716 和 0x00 717。类似地，数据操作数 721 包括源数据元素：P722、O723、N724、M725、L726、K727、
J728、I729、H730、G731、F732、E733、D734、C735、B736、A737。在图 7 的数据区段的表示中，数据元素位置也在数据下被标注为十六进制值。因此，用掩码
25 701 和数据 721 执行紧缩混洗操作。使用该组混洗掩码 701，可并行执行数据 721 的处理。

在评估每个数据元素混洗掩码时，来自指定数据元素或‘0’的合适数据被混洗到该特殊混洗掩码的相应数据元素位置。例如，最右面的混洗掩码 717 具有值 0x00，它被解码成指定来自源数据操作数的位置 0x00 的数据。作为响应，来自数据位置 0x0 的数据 A 被复制到结果 741 的最右面的位置。类似地，从右面起的第

二个混洗掩码 716 具有值 0x0D，它被解码为 0xD。因此，来自数据位置 0xD 的数据 N 被复制到结果 741 中从右面起的第二个位置。

结果 741 中从左面起的第四个数据元素位置是 ‘0’。这属于该数据元素位置的混洗掩码中的 0x8F 值。在该实施例中，混洗掩码字节的比特 7 是 ‘设定到零’
5 或 ‘排空到零’ 指示器。如果该字段被设定，则结果中的相应数据元素位置用 ‘0’
值填充代替来自第二数据操作数 721 的数据。类似地，结果 741 中从右面起的第七
个位置具有值 ‘0’。这是由于掩码 701 中用于该数据元素位置的 0xF0 的混洗掩
码值。注意，在某些实施例中不是混洗掩码中的所有比特都被使用。在该实施例中，
混洗掩码的下四位字节或 4 个比特足以选择源数据操作数 721 中的十六个可能的数
10 据元素中的任一个。当比特 7 是 ‘设定到零’ 字段时，在某些实施例中 3 个其它比
特保持不使用并可被保留或忽略。对于该实施例，‘设定到零’ 字段控制和超过数
据元素选择，如混洗掩码的下四位字节中所指示的。在这两种实例中，从左面起的
第四个数据元素位置和从右面起的第七个位置，其中在比特 7 处设置 ‘排空到零’
标记的 0x80 的混洗掩码值也可使得相应结果数据元素位置用 ‘0’ 填充。

15 如图 7 所示，箭头示出了掩码 701 中每个混洗掩码的数据元素混洗。根据特
殊一组混洗掩码，一个或多个源数据元素不会出现于结果 741 中。在某些实例中，
一个或多个 ‘0’ 也出现于结果 741 中的各数据元素位置处。如果混洗掩码被配置
成广播一个或特殊一组数据元素，则用于这些数据元素的数据会在结果中作为选择
模式被重复。本发明的实施例不限于任何特殊的排列或混洗模式。

20 如上所述，在本实施例中源数据寄存器也用作目的地数据存储寄存器，从而
减少了所需寄存器的数量。尽管因此将源数据 721 重写，但混洗掩码 701 的组不会
被改变并可用于将来的参考。源数据存储装置内的被重写数据可从存储器或其它寄
存器重新加载。在另一实施例中，多个寄存器可用作源数据存储装置，其中它们各
自的数据在目的地数据存储装置中按期望组织。

25 图 8 示出了根据本发明另一实施例的对字宽数据元素的数据混洗操作。本实
例的一般讨论一定程度上类似于图 7 的讨论。但在这种情况下，数据操作数 821
和结果 831 的数据元素是字长度。对于该实施例，数据元素字作为数据元素字节对
处理，而掩码操作数 801 中的混洗掩码是字节大小。因此，一对混洗掩码字节被用
于定义每个数据元素字位置。但对于另一实施例，混洗掩码也可具有字间隔尺寸并

描述结果中字大小的数据元素位置。

本示例的掩码操作数 801 包括若干字节宽的数据元素，并具有混洗掩码: 0x03 802、0x02 803、0x0F 804、0x0E 805、0x83 806、0x82 807、0x0D 808、0x0C 809、0x05 810、0x04 811、0x0B 812、0x0A 813、0x0D 814、0x0C 815、0x01 816 和 0x00 5 817。数据操作数 821 包括源数据元素: H822、G823、F824、E835、D836、C827、B828、A829。在图 8 的数据区段的表示中，数据元素位置也标注于数据下作为十六进制值。如图 8 所示，数据操作数 821 中的每个字大小数据元素都具有数据位置地址，它占据两个字节大小位置。例如，数据 H822 占据字节大小数据元素位置 0xF 和 0xE。

10 用掩码 801 和数据 821 进行紧缩混洗操作。图 8 的箭头示出了掩码 801 中每个混洗掩码的数据元素的混洗。在评估每个数据元素混洗掩码时，来自数据操作数 821 的指定数据元素位置的合适数据或者 ‘0’ 被混洗到用于该特殊混洗掩码的结果 831 中的相应数据元素位置。在该实施例中，字节大小混洗掩码成对操作，以指定字大小的数据元素。例如，掩码操作数 801 中的两个最左面的混洗掩码 0x03 802、15 0x02 803 一同对应于结果 831 的最左面的字宽数据元素位置 832。在混洗操作期间，数据元素字节位置 0x03 和 0x02 的两个数据字节或单个数据字(在这种情况下是数据 B828)被排列成结果 831 中的两个最左面的字节大小数据元素位置 832。

此外，混洗掩码也可配置成迫使结果中的一字大小数据元件为 ‘0’，如用结果 831 中的第三个字大小数据元素位置 834 的混洗掩码 0x83 806 和 0x82 807 示出的。混洗掩码 0x83 806 和 0x82 807 设置它们的 ‘设定到零’ 字段。尽管这里使两个混洗掩码字节成对，但也可实现不同的成对以便例如将四个字节排列在一起作为四字或 8 个字节排列在一起以形成一双四字。类似地，成对不限于连续的混洗掩码或特殊字节。在另一实施例中，字大小混洗掩码可用于指定字大小数据元素。

25 图 9 是示出混洗数据方法的一个实施例的流程图。长度值 L 这里一般用于表示操作数和数据块的宽度。根据特殊实施例，L 可用于按数据区段的数目指定宽度，比特、字节、字等。在框 910 处，第一长度 L 的紧缩数据操作数被接收用于供混洗操作使用。在框 920 处接收指定混洗模式的长度 L 的一组 M 长度混洗掩码。在该示例中，L 是 128 比特而 M 是 8 比特或一字节。在另一实施例中，L 和 M 也可是其它值，如分别为 256 和 16。在框 930 处，执行混洗操作，其中来自数据操作

数的数据元素根据混洗模式被混洗排入结果中。

将在对每个数据元素位置发生了什么方面进一步描述该实施例的框 930 处的混洗细节。对于一个实施例，对所有紧缩结果数据元素位置的混洗被并行处理。在另一实施例中，每次可一起处理某些部分的掩码。在框 932 处，进行检查以确定是 5 否设定了零标记。该零标记表示每个混洗掩码的设定/排空到零字段。如果在框 932 处该零标记被确定为设置，则与该特殊混洗掩码相对应的结果数据元素位置处的项目被设定为‘0’。如果在框 932 处发现该零标记未被设定，则将来自混洗掩码指定的源数据元素的数据排入与该混洗掩码相对应的结果的目的地数据元素位置。

当前，使用整数指令的表查找需要大量指令。如果整数操作被用于访问用于 10 用 SIMD 指令实现的算法的数据，则每次查找就需要更多数量的指令。但通过使用紧缩字节混洗指令的实施例，急剧减少指令计数和执行时间。例如，如果表大小是 16 个字节或更少，则在用单个指令的表查找期间，可访问十六个数据字节。如果表大小在 17 和 32 字节之间，可使用 11 个 SIMD 指令查找表数据。如果表大小在 33 到 64 字节之间，则需要 23 个 SIMD 指令。

15 存在某些使用数据并行的应用，由于它们查找表的使用而不用 SIMD 指令实现。视频压缩方法 H.26L 的量化和分块算法是使用可装入 128 比特寄存器的小查找表的算法示例。在某些情况下，这些算法使用的查找表较小。如果表能装入单个寄存器，则可用一个紧缩混洗指令完成表查找操作。但如果表的存储空间要求超过单个寄存器的大小，则紧缩混洗指令的实施例仍可通过不同算法工作。用于处理过 20 大表的一个方法实施例将表划分成诸部分，每部分等于一个寄存器的容量，且用混洗指令访问这些表部分中的每一个。混洗指令使用相同的混洗控制序列来访问表的每个部分。结果，在这些情况下可用紧缩字节混洗指令实现并行表查找，因此允许 SIMD 指令的使用来改善算法性能。本发明的实施例可帮助改善性能并减少使用小查找表的算法所需的存储器访问数量。其它实施例还允许使用 SIMD 指令的多个查 25 找表元素的访问。根据本发明的紧缩字节混洗指令允许有效的 SIMD 指令实现，代替使用小查找表的算法的较无效的整数植入。本发明的该实施例显示了如何从需要大于单个寄存器的存储器空间的一表访问数据。在该示例中，寄存器包含表的不同区段。

图 10A-H 示出了使用 SIMD 指令的并行表查找算法的操作。图 10A-H 所述的

示例包括来自多个表的数据的查找，其中如一组掩码中指定的某些选择的数据元素从这多个表混洗入结果数据的合并块。在紧缩操作环境下说明以下的讨论，特别是以上揭示的紧缩混洗指令。该示例的混洗操作重写寄存器中的源表数据。如果该表将在查找操作后被重复使用，则表数据应在执行操作前被复制到另一寄存器，从而
5 不需要另一次加载。在可选实施例中，混洗操作利用三个分开的寄存器或存储器位置：两个源和一个目的地。可选实施例中的目的地是与任一源操作数不同的寄存器或存储器位置。因此，源表数据不被撤消并可被重复使用。在该示例中，表数据被
10 处理为来自较大表的不同部分。例如，低表数据 1021 来自表的低地址区域而高表数据 1051 来自表的高地址区域。本发明的实施例不限制表数据能够源自哪里。数据块 1021 和 1051 可相邻、远离或甚至重叠。类似地，表数据也可来自不同的数据表或不同的存储器源。也可想象，这种表查找和数据合并可在来自多个表的数据上进行。例如，代替来自于相同表的不同部分，低表数据 1021 可来自于第一个表而高表数据 1051 可来自于第二个表。

图 10A 示出了基于一组混洗掩码的来自表的第一组数据元素的紧缩数据混洗。该第一组数据元素被组成为叫做低表数据 1021 的操作数。在该示例中，掩码 1001 和低表数据 1021 分别由 16 个元素构成。掩码 1001 和低表数据 1021 的混洗操作生成临时结果 A1041。混洗控制掩码的较低部分选择寄存器中的数据元素。选择数据元素所需的比特数是寄存器数据元素数量的 \log_2 。例如，如果寄存器容量是 128 比特且数据类型是字节，则寄存器数据元素的数量是 16。在这种情况下，需要
15 四个比特来选择数据元素。图 10B 示出了基于图 10A 的相同一组混洗掩码的来自表的第二组数据元素的紧缩数据混洗。该第二组数据元素被组成为叫做高表数据 1051 的操作数。本示例中，高表数据 1051 也由 16 个元素构成。掩码 1001 和高表数据 1051 的混洗操作产生临时结果 B1042。
20

因为相同的一组掩码 1001 与低表数据 1021 和高表数据 1051 一起使用，所以
25 它们各自的结果 1041 和 1042 表现为具有类似设置的数据，但来自于不同的源数据。例如，结果 1041 和 1042 两者的最左面的数据位置具有来自其各自数据源 1021、1051 的数据元素 0xE1023、1053 的数据。图 10C 示出了涉及选择过滤器 1043 和一组混洗掩码 MASK（掩码）1001 的逻辑紧缩 AND 操作。这种情况中的选择过滤器是区分掩码 1001 中的哪个混洗掩码相关于第一表数据 1021 和哪个相关于第二表

数据 1051 的过滤器。该实施例的混洗掩码使用源选择字段，SRC 选择 336，如图 3C 中之前所讨论的。混洗控制字节的较低比特用于选择寄存器中的数据元素位置，且排除最高有效比特之外的几个较高比特用于选择表的区段。对于该实施例，与用于选择数据的那些比特紧接上方和紧邻的比特用来选择表的部分。选择过滤器 5 1043 将 0x10 应用于掩码 1001 中的所有混洗掩码，分出来自混洗掩码的源选择字段。紧缩 AND 操作产生表选择掩码 1044，它指示最终结果中的哪个数据元素位置应来自第一数据组 1021 或第二数据组 1051。

选择表部分的比特数等于表部分数量的 \log_2 。例如，在具有 16 字节寄存器的从 17 到 32 字节的表大小的情况下，最低的 4 个比特选择数据且第五个比特选择表 10 部分。这里，当存在两个数据源 1021 和 1051 时，源选择使用每个混洗掩码的第二四位字节的最低比特，即比特 4，来指定数据源。具有 0 和 15 之间的指数的表部分用图 10A 中的紧缩混洗指令访问。具有 16 和 31 之间的指数的表部分用图 10B 中的紧缩混洗指令访问。选择表部分的字段与图 10C 中的混洗控制字节/指数分离。在使用较大量数据源的实现中，源选择字段会需要附加比特。在三十二字节的 15 表的情况下，混洗控制字节 0x00 到 0x0F 将选择第一表部分中的表元素零到十五，且混洗控制字节 0x10 到 0x1F 将选择第二表部分中的表元素十六到三十一。例如，考虑一混洗控制字节指定 0x19。0x19 的比特表示是 0001 1001。较低的四个比特 1001 选择第九个字节（从 0 计数）且被设定为 1 的第五个比特选择两个表中的第二个表。等于 0 的第五个比特将选择第一个表。

20 通过选择第五位是零的混洗控制字节，对于图 10D 中的本实施例，用紧缩比较相等操作计算用于选择从具有指数 0 到 15 的第一表部分访问到的数据值的掩码。图 10D 示出了低过滤器 1045 和表选择掩码 1044 的紧缩“比较相等操作”。用于第一表部分的图 10D 中产生的低表选择掩码通过另一紧缩混洗操作选择从第一表部分访问的数据元素。该实例中的低过滤器 1045 是用于抽出或突出来自第一数据 25 组 1021 的由混洗掩码指示的数据元素位置的掩码。如果本实施例中源选择字段是‘0’，则数据源将是低表数据 1021。比较相等操作产生低表选择掩码 1046，其中 0xFF 值用于具有源选择值‘0’的数据元素位置。

通过选择第五位是 1 的混洗控制字节，用图 10E 中的紧缩比较相等操作计算用于选择从具有指数 16 到 31 的第二表部分访问到的数据值的掩码。图 10E 示出

了作用于高过滤器 407 和表选择掩码 1044 的类似的比较相等操作。图 10E 中产生的用于第二表部分的高表选择掩码通过紧缩混洗操作选择从第二表部分访问到的数据元素。高过滤器 1047 是用于抽出来自第二数据组 1051 的由混洗掩码的源选择字段指示的数据元素位置的掩码。如果本实施例中源选择字段是 ‘1’，则数据源 5 将是高表数据 1051。比较相等操作产生高表选择掩码 1048，其中 0xFF 值用于具有源选择值 ‘1’ 的数据元素位置。

选自两个表部分的数据元素在图 10F 处被合并。图 10F 处，示出了对低表选择掩码 1046 和临时结果 A1041 的紧缩 AND 操作。该紧缩 AND 操作从经由基于源选择字段的掩码 1046 的第一数据组 1021 中滤出选择的混洗数据元素。例如，用 10 于最左面的数据元素位置的混洗掩码 1002 中的源选择字段具有值 ‘0’，如表选择掩码 1044 中示出的。因此，低表选择掩码 1046 在该位置中具有 0xFF 值。这里图 10F 中 0xFF 和最左面的数据元素位置中的数据的与操作使得数据 O 转移到选择的低表数据 1049。另一方面，用于从左面起的第三数据元素位置的混洗掩码 1004 中的源选择字段具有值 ‘1’，指示该数据来自于第一数据组 1021 之外的源。因此， 15 低表选择掩码 1046 在该位置中具有 0x00 值。这里的与操作不会将数据 J 传递到选择的低表数据 1049 且该位置保持为空，0x00。

对高表选择掩码 1048 和临时结果 B1042 的类似的紧缩 AND 操作在图 10G 中示出。该紧缩 AND 操作滤出从经掩码 1048 的第二数据组 1051 中选择的混洗数据元素。不同于图 10F 中所述的紧缩 AND 操作，掩码 1048 允许来自第二数据组的 20 源选择字段所指定的数据传递到选择的高表数据 1050 同时保持其它数据元素位置为空。

图 10H 示出了来自第一数据组和第二数据组的选择数据的合并。对选择的低表数据 1049 和选择的高表数据 1050 执行紧缩逻辑 OR 操作，以获得合并的选择表数据 1070，这是本示例中并行表查找算法的期望结果。在可选实施例中，将选择的低表数据 1049 和选择的高表数据 1050 加在一起的紧缩加法操作也可产生合并的选择表数据 1070。如图 10H 所示，在该实施例中，选择的低表数据 1049 或选择的高表数据 1050 具有用于给定数据位置的 0x00 值。这是因为没有 0x00 值的其它操作数将包含从合适源选择的期望的表数据。这里，结果 1070 中最左面的数据元素位置是 O，它是来自第一数据组 1021 的混洗数据 1041。类似地，结果 1070 中从

左面起的第三个数据元素位置是 Z，它是来自第二数据组 1051 的混洗数据 1042。

本实施例中用于在过大表中查找数据的方法一般可通过以下操作概括。首先，将表数据复制或加载入寄存器。通过紧缩混洗操作访问来自每个表部分的表值。标识表部分的源选择字段从混洗掩码中提取。用表部分数对源选择字段进行比较是否相等（compare-if-equal）操作以确定哪些表部分是用于混洗数据元素的合适源。比较是否相等操作提供掩码，以进一步滤出用于每个表部分的期望的混洗数据元素。来自合适表部分的期望的数据元素被合并在一起，形成最终的表查找结果。

图 11 是示出使用 SIMD 指令执行表查找的方法的一个实施例的流程图。这里描述的流程通常遵循图 10A-H 的方法，但不限于此。这些操作中的某些也可按不同顺序或使用各种类型的 SIMD 指令执行。在框 1102 处，接收指定混洗模式的一组混洗掩码。这些混洗掩码还包括源字段，以指示从哪个表或源混洗数据元素以获得期望的结果。在框 1104 处，加载表的第一部分的数据元素或第一数据组。第一部分数据元素在框 1106 处根据框 1102 的混洗模式被混洗。在框 1108 处加载用于第二部分表的数据元素或第二数据组。在框 1110 处根据框 1102 的混洗模式混洗第二部分数据元素。在框 1112 处，从混洗掩码中滤出表选择。本实施例的表选择包含指定假定数据元素源自哪里的源选择字段。在框 1114 处，为来自表的第一部分的混洗数据生成表选择掩码。在框 1116 处，为来自表的第二部分的混洗数据生成表选择掩码。这些表选择掩码将从合适的表数据源中滤出用于特定数据元素位置的期望的混洗数据元素。

框 1118 处，根据用于第一表部分的框 1114 的表选择掩码，从第一表部分的混洗数据中选择出数据元素。框 1120 处，根据用于第二表部分的框 1116 的表选择掩码，从第二表部分的混洗数据中选择数据元素。在框 1118 处从第一表部分和在框 1120 处从第二表部分选择的混洗数据元素在框 1122 处被合并在一起，以获得合并的表数据。一个实施例的合并的表数据包括从第一表数据和第二表数据混洗的数据元素。对于另一实施例，合并的表数据可包括从超过两个表源或存储器区域查找的数据。

图 12 是示出执行表查找的方法的另一实施例的流程图。在框 1202 处，加载具有多个数据元素的表。在框 1204 处，确定该表是否装入单个寄存器。如果该表装入单个寄存器，则在框 1216 处用混洗操作执行表查找。如果数据不装入单个寄

存器，则在框 1206 处对于表的每个相关部分用混洗操作执行表查找。执行逻辑紧缩的 AND 操作，以获得选择表部分或数据源的比特或字段。框 1210 处的“比较是否相等”查找形成一掩码，以便从要查找的表的相关部分中选择表数据。在框 1212 处，逻辑 AND 操作用于从表部分中查找和选择数据项。逻辑 OR 操作在框 5 1214 处合并选择的数据，以获得期望的表查找数据。

紧缩混洗指令的一个实施例被实现为用于用排空到零能力在多个寄存器之间重排数据的算法。混合操作的目的是在单个 SIMD 寄存器中按选择的结构合并两个或多个 SIMD 寄存器的内容，在该选择的结构中结果中数据的位置与源操作数中它们的原始位置不同。选择的数据元素首先被移到期望的结果位置且未选择的数据元素被设定为零。一个寄存器的选择的数据元素要移到的位置在另一个寄存器中被设定为零。结果，单独一个结果寄存器会在给定的数据元素位置中包含非零数据项。以下的一般指令序列可用于混合来自两个操作数的数据：

10 紧缩字节混洗数据 A，掩码 A；
15 紧缩字节混洗数据 B，掩码 B；
紧缩逻辑 OR 结果 A，结果 B。

操作数数据 A 和数据 B 包含要被重排或设定零的元素。操作数掩码 A 和掩码 B 包含指定数据元素要移动到哪里以及哪些数据元素要设定为零的混洗控制字节。对于该实施例，未由掩码 A 设定为零的目的地位置中的数据元素由掩码 B 设定为零，且未由掩码 B 设定为零的目的地位置由掩码 A 设定为零。图 13A-C 示出了用于在多个寄存器之间重排数据的算法。在该示例中，来自两个数据源或寄存器 1304、1310 的数据元素被一同混洗入交错数据块 1314。该示例的包含掩码 1302、1308、源数据 1304、1310 以及结果 1306、1312、1314 的数据块分别是 128 比特长并由十六个字节大小的数据元素构成。但，可选实施例可包括具有各种大小的数据元素的其它长度的数据块。

25 图 13A 示出了第一掩码 MASK（掩码） A1302 的作用于第一源数据操作数 DATA（数据） A 1304 第一紧缩数据混洗操作。对于该示例，期望的交错结果 1314 将包括来自第一数据源 1304 的一个数据元素和来自第二数据源 1310 的另一数据元素的交错模式。在该示例中，数据 A1304 的第五个字节与数据 B1310 的第十二个字节交错。在该实施例中，掩码 A1302 包括“0x80”和“0x05”的重复模式。该

实施例中的 0x80 值设置了设定到零字段，其中相关的数据元素位置用 ‘0’ 填充。0x05 值阐明了用于该混洗掩码的有关数据元素位置用来自数据 A1304 的数据元素 0x5 的数据 F1 排列。本质上，掩码 A1302 中的混洗模式在每隔开一个结果数据元素的位置处排列和重复数据 F1。这里，数据 F1 是要从数据 A1304 混洗的单块数据。在可选实施例中，可以混洗来自各种数量的源数据元素的数据。因此，实施例不限于包括单块数据的模式或任何特定模式。用于掩码模式的排列组合具有所有类型的可能性。图 13A 的箭头示出了经掩码 A1302 的混洗掩码的数据元素的混洗。因此，该混洗操作的结果 A1306 由经掩码模式 1302 的 ‘0’ 和 F1 的模式构成。

图 13B 示出了包含第二掩码 MASK（掩码） B1308 以及第二源数据操作数 DATA(数据) B1310 的第二紧缩数据混洗操作。掩码 B1308 包括“0x0C”和“0x80”的重复模式。0x80 值使得该混洗掩码的有关数据位置接收 ‘0’。0xC0 值使得与该混洗掩码相对应的结果数据元素位置用来自数据 B1310 的数据元素 0xC 的数据 M2 排列。掩码 B1308 的混洗模式将数据 M2 排列到所有其它结果数据元素位置。图 13B 中的箭头示出了掩码 B1308 中经该组混洗掩码的数据元素的混洗。因此，该混洗操作的结果 B1312 由经掩码模式 1308 的 ‘0’ 和 M2 的模式构成。

图 13C 示出了混洗数据、结果 A1306 和结果 B1312 的合并，以实现交错结果 1314。该合并用紧缩逻辑 OR 操作实现。结果 A1306 和结果 B1312 中的 ‘0’ 值模式允许 M2 和 F1 数据值 1314 的交错。例如，在最左面的数据元素位置处，‘0’ 和 M2 的逻辑 OR 导致结果 1314 的最左面数据元素位置中的 M2。类似地，在最右面的数据元素位置处，F1 和 ‘0’ 的逻辑 OR 导致结果 1314 的最右面数据元素位置中的 F1。因此，来自多个寄存器或存储器位置的数据可被重排成期望模式。

图 14 是示出在多个寄存器之间重排数据的方法的一个实施例的流程图。在框 1402 处从第一寄存器或存储器位置加载数据。在框 1404 处，基于第一组混洗掩码混洗该第一寄存器数据。在框 1406 处，从第二寄存器或存储器位置加载数据。该第二寄存器数据在框 1408 处根据第二组混洗掩码被混洗。来自第一和第二寄存器混洗的数据在框 1410 处以逻辑 OR 合并，以便用来自第一和第二寄存器的数据得出交错数据块。

图 15A-K 示出了用于在多个寄存器之间混洗数据以生成交错数据的算法。这是交错平面色数据的应用示例。常在分开的色平面中处理图像数据，随后这些平面

稍后被交错用于进行显示。下述算法显示红平面、绿平面和蓝平面数据的交错，如诸如位图的图像格式所使用的。大量色空间和交错模式都是可能的。这样，该方法能方便地扩展到其它色空间和格式。该示例实现了常用的图像处理数据格式过程，其中红（R）平面、绿（G）平面和蓝（B）平面数据被交错成 RGB 格式。该示例
5 显示了根据本发明的排空到零能力如何显著地减少存储器访问。

来自三个源的数据按交错方式组合在一起。更具体地，数据与象素颜色数据相关。例如，每个象素的颜色数据可包括来自红（R）、绿（G）和蓝（B）源的信息。通过组合颜色信息，可以评估红/绿/蓝（RGB）数据以提供特定象素期望的颜色。这里，红数据保存于操作数数据 A1512 中，绿数据于数据操作数数据 B1514
10 中，且蓝数据于数据 C1516 中。该结构可存在于图形或存储器系统中，其中用于每个分开颜色的数据被存储在一起或者在流数据中分别采集。为了在再造或显示期望图像时使用该信息，象素数据必须被排列成 RGB 模式，其中用于特定象素的所有数据被组合在一起。

对于该实施例，具有预定模式的一组掩码用于将 RGB 数据交错在一起。图 15A
15 示出了一组掩码：具有第一模式的掩码 A1502，具有第二模式的掩码 B1504，以及具有第三模式的掩码 C1506。来自每个寄存器的数据将隔开三个字节，以使它可与来自两个其它寄存器的数据相交错。具有十六进制值 0x80 的控制字节设定了最高有效比特，以使相应的字节通过紧缩字节混洗指令被排空到零。在这些掩码的每一个中，每个第三混洗掩码被启用以选择用于混洗的数据元素，同时两个插入混洗掩
20 码具有值 0x80。该 0x80 值指示用于那些相应数据元素位置的掩码中的设定到零字段被置 1。因此，‘0’将被置于与该掩码相关的数据元素位置中。在该示例中，掩码模式将基本分出每个颜色的数据元素，以实现交错。例如，在将掩码 A1502 应用于混洗操作中的数据操作数时，掩码 A1502 使得六个数据元素（0x0、0x1、
0x2、0x3、0x4、0x5）以每个数据元素之间隔开两个数据元素空间进行混洗。类似
25 地，掩码 B1504 将在 0x0、0x1、0x2、0x3、0x4 处隔开数据元素进行混洗。掩码 C1506 将在 0x0、0x1、0x2、0x3、0x4 处隔开数据元素进行混洗。

注意，在该实现中，每个特定重叠数据元素位置的混洗掩码具有两个设定到零字段和指定数据元素的一个混洗掩码。例如，参考三组掩码 1502、1504、1506 的最右面的数据元素位置，混洗掩码值为 0x00、0x80 和 0x80，分别用于掩码 A1502、

掩码 B1504 和掩码 C1506。因此，只有用于掩码 A1502 的混洗掩码 0x00 指明用于该位置的数据。本实施例中的掩码被形成模式，以使混洗数据能被方便地合并以形成交错 RGB 数据块。

图 15B 示出了要交错的数据块：数据 A1512、数据 B1514 和数据 C1516。对于该实施例，每组数据都具有数据条目，它具有用于十六个象素位置的颜色信息。这里，附随数据元素中的每个颜色字母的下标概念表示该象素号。例如，R0 是用于象素 0 的红数据而 G15 是用于象素 15 的绿数据。所示的每个数据元素处的十六进制值是该数据元素位置的号码。颜色数据（数据 A1512、数据 B1514、数据 C1516）可被复制入其它寄存器，以使数据不会由混洗操作重写并可被重复使用而不用其它加载操作。在该实施例的示例中，需要使用掩码 1502、1504 和 1506 的三次传递（pass）来完成象素数据交错。对于可选实现和其它数据量，传递和混洗操作的次数可按需要变化。

图 15C 示出了结果数据块，MASKED DATA（掩码数据）A1522，用于使用第一混洗模式 MASK（掩码）A1502 的对红象素数据 DATA（数据）A1512 的紧缩混洗操作。响应于掩码 A1502，红象素数据被排入所有第三数据元素位置。类似地，图 15D 示出了结果数据块，MASKED DATA（掩码数据）B1524，用于使用第二混洗模式 MASK（掩码）B1504 的对绿象素数据 DATA（数据）B1514 的紧缩混洗操作。图 15E 示出了结果数据块，MASKED DATA（掩码数据）C1526，用于使用第三混洗模式 MASK（掩码）C1506 的对蓝象素数据 DATA（数据）C1516 的混洗操作。对于该实施例的掩码模式，来自这些混洗操作的结果数据块提供交错的数据元素以使数据元素中的一个具有数据而两个具有‘0’。例如，这些结果 1522、1524、1526 的最左面的数据元素位置分别包含 R5、‘0’ 和‘0’。在下一个数据元素位置处，呈现了用于另一 RGB 颜色的象素数据。因此，当一起合并时，达到组合的 RGB 类型。

在该实施例中，用于红数据和绿数据的以上混洗数据以紧缩逻辑 OR 操作被首先合并在一起。图 15F 示出了用于掩码数据 A1522 和掩码数据 B1524 的紧缩逻辑 OR 的结果数据，即交错的 A&B 数据 1530。混洗的蓝数据现在以另一紧缩逻辑 OR 操作与交错的红和绿数据合并在一起。图 15G 示出了来自于掩码数据 C1526 和掩码数据 A&B1530 的紧缩逻辑 OR 的新结果，即交错的 A,B&C 数据 1532。因

此，图 15G 的结果数据块包含了头五个象素和第六个象素的一部分的交错 RGB 数据。本实施例的算法的后续迭代将为十六个象素的其余部分产生交错 RGB 数据。

在这点上，数据 A1512、数据 B1514 和数据 C1516 中的数据的三分之一已被交错。两种方法可用于处理这些寄存器中的其余数据。另一组混洗控制字节可用于安排要交错的数据或者数据 A1512、数据 B1514 和数据 C1516 中的数据可被右移，以便可再次使用混洗掩码 1502、1504 和 1506。在这里示出的实现中，将数据移位以避免需要加载附加混洗控制字节的存储器访问。在没有这些移位操作的情况下，在该实施例中需要九组控制字节代替三组(掩码 A1502、掩码 B1504、掩码 C1506)。该实施例也可在有限数量的寄存器可用且存储器访问较长的架构中应用。

在其中有较多寄存器可用的可选实施例中，保持寄存器中的所有或大量掩码组以使移位操作是不必要的会是更加有效的。此外，在具有许多寄存器和执行单元的架构中，所有混洗操作可并行执行而不必等待移位产生。例如，具有九个混洗单元和九个掩码组的乱序处理器可并行执行九个混洗操作。在以上实施例中，在再应用掩码前必须将数据移位。

根据已为该特定颜色处理的数据元素的数目，数据 A1512、数据 B1514 和数据 C1516 的原始颜色数据中的数据元素被移位。在该示例中，以上已为红色处理了用于六个象素的数据，从而用于红数据操作数数据 A1512 的数据元素被向右移 6 个数据元素位置。类似地，已为绿和蓝两者处理了用于 5 个象素的数据，从而用于绿数据操作数数据 B1514 和用于蓝数据操作数数据 C1516 的数据元素被分别右移 5 个数据元素位置。被移位的源数据在图 15H 中分别示作为用于颜色红、绿、蓝的数据 A'1546、数据 B'1542 和数据 C'1544。

用该移位数据重复以上用图 15A-G 讨论的混洗和逻辑 OR 操作。分别与掩码 A1502、掩码 B1504 和掩码 C1506 一起作用于数据 B'1542、数据 C'1544 和数据 A'1546 的后续紧缩混洗操作结合对三个紧缩混洗结果的紧缩逻辑 OR 操作提供用于另四个象素的交错 RGB 数据和另两个的一部分。该结果数据，即交错 A'、B' 和 C' 数据 1548 在图 15I 中示出。注意，最右面的两个数据元素涉及第六个象素，它已将其红数据 R5 用第一交错数据组 1532 排列。按第二传递的处理结果，再次使原始象素颜色数据移位合适数量的位置。这里，已为红和蓝处理了用于五个附加象素的数据，从而用于红数据操作数数据 A'1546 和用于蓝数据操作数数据 C'1544

的数据元素被右移 5 个数据元素位置。已为绿处理了用于六个象素的数据，从而用于绿数据操作数数据 B'1542 的数据元素被右移六个位置。图 15J 示出了用于该第三次传递的移位数据。将以上紧缩混洗和逻辑 OR 的重复应用于 DATA C''1552、DATA A''1554 和 DATA B''1556。图 15K 将用于最后的 16 个象素的最终交错 RGB 数据示作为交错的 A'',B'' DATA1558。具有 B10 的最右面的数据元素相关于第十一个象素，它已使其绿数据 G10 和红数据 R10 与第二交错数据组 1548 一起排列。因此，经由具有一组掩码模式的一系列紧缩混洗和紧缩逻辑 OR 操作，来自多个源 1512、1514 和 1516 的数据按期望的方式一起合并和重排，以便进一步使用或处理这些结果 1532、1548、1558。

10 图 16 是示出在多个寄存器之间混洗数据以生成交错数据的一个实施例的流程图。例如，本方法的实施例可应用于交错象素数据的生成，如图 15A-K 中所讨论的。虽然在三个数据源或数据平面的环境下描述本实施例，但其它实施例可用两个或更多个数据平面进行操作。这些数据平面可包括用于一个或多个图像帧的颜色数据。在框 1602 处，加载用于第一、第二和第三平面的帧数据。在该示例中，用于 15 多个象素的 RGB 颜色数据可作为来自三个不同平面的单独颜色获得。第一平面中的数据用于颜色红，第二平面中的数据用于绿色，且第三平面中的数据用于蓝色。在框 1604 中，加载具有混洗控制模式 (M1、M2 和 M3) 的一组掩码。这些混洗控制模式确定混洗模式和数据排列，以便一起交错颜色。根据该实现，可采用任何数量的混洗模式来生成期望的数据排列。

20 在框 1606 处，为每个数据平面选择合适的控制模式。在该实施例中，基于期望模式数据是什么顺序以及当前执行哪个迭代来选择混洗模式。框 1608 处，来自第一数据组 (红) 的帧数据用第一混洗控制模式进行混洗，以获得混洗的红数据。在框 1610 处，用第二混洗控制模式混洗第二数据组 (绿)，以获得混洗的绿数据。在框 1612 处，用第三混洗控制模式混洗第三数据组 (蓝)，以实现混洗的蓝数据。 25 虽然本实施例中三个掩码和它们的混洗控制模式彼此不同，但在每次迭代期间可在超过一个单独数据组上使用一掩码及其混洗模式。此外，某些掩码会比其它的更常使用。

框 1614 处，用于三个数据组的框 1608、1610、1612 的混洗数据被合并在一起，形成该传递的交错结果。例如，第一传递的结果会看起来像图 15G 的交错数

据 1532，其中每个象素的 RGB 数据被一起组成为一组。框 1616 处，进行检查以确定寄存器中是否加载了更多帧数据用于进行混洗。如果否，则在框 1620 处进行检查以确定是否存在更多的来自三个数据平面用于被交错的数据。如果否，则进行所述方法。如果在框 1620 中存在更多的平面数据，则该过程返回到框 1602，加载
5 用于混洗的更多帧数据。

如果框 1616 处的确定是真，则颜色数据的每个平面中的帧数据被移位预定计数，该计数对应于在最后传递期间哪个掩码模式被应用于该特殊颜色的数据组。例如，与来自图 15G 的第一传递示例相一致，第一、第二、第三平面中的红、绿、蓝数据分别被移位 6、5 和 5 个位置。根据该实现，每次传递时为每个颜色数据选择的混洗模式可以是不同的或者重新使用相同的一个。对于一个实施例在第二次传递期间，来自第一迭代的三个掩码被旋转，以使第一平面数据现在与第三掩码配对，第二平面数据与第一掩码配对，且第三平面数据与第三掩码配对。该掩码旋转允许从一次传递到下一次的交错 RGB 数据的合适的连续性，如图 15G 和 15I 所示出的。如同在第一次传递中，混洗和合并继续。如果期望三分之个或更多的迭代，本实施
10 例的混洗掩码模式在不同数据平面之间继续旋转以生成更多交错 RGB 数据。
15

根据本发明使用紧缩混洗指令的算法实施例也可改善具有当前硬件资源的处理器和系统性能。但随着技术继续进步，本发明的实施例在与更大量的硬件资源和更快、更有效的逻辑电路组合时可对改善性能具有更深的影响。因此，具有字节间隔尺寸和排空到零选项的紧缩混洗指令的一个有效实施例可具有跨九代处理器的
20 不同和更大的影响。在现代处理器架构中简单添加更多资源不会确保更好的性能改善。通过维持类似于并行表查找和紧缩字节混洗指令（PSHUFB）的一个实施例的应用的效率，更大的性能改善是可能的。

虽然以上示例在 128 比特宽的硬件/寄存器/操作数环境中作出一般性描述以简化讨论，但其它实施例采用 64 或 128 比特宽的硬件/寄存器/操作数来执行紧缩混
25 洗操作、并行表查找和多寄存器数据重排。此外，本发明的实施例不限于特定硬件或技术类型，诸如 MMX/SSE/SSE2 技术，并可与其它 SIMD 实现和其它图形数据处理技术一同使用。

在以上说明书中，参考其示例性实施例描述了本发明。但显然，可对其进行各种修改和变化而不背离本发明的较宽精神和范围，如所附权利要求书中所阐述

的。因此，说明书和附图被认为是说明性而非限制性的。

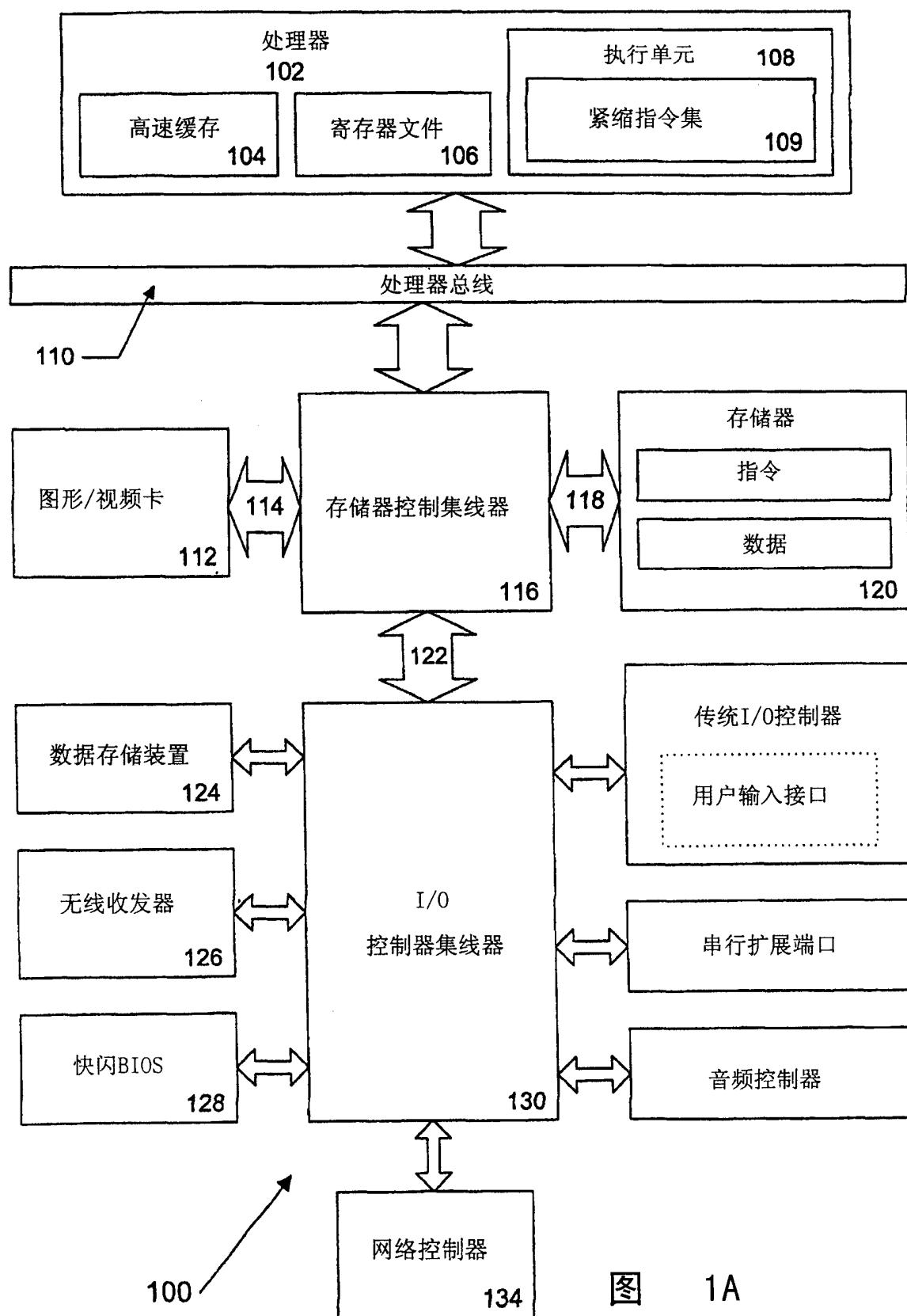


图 1A

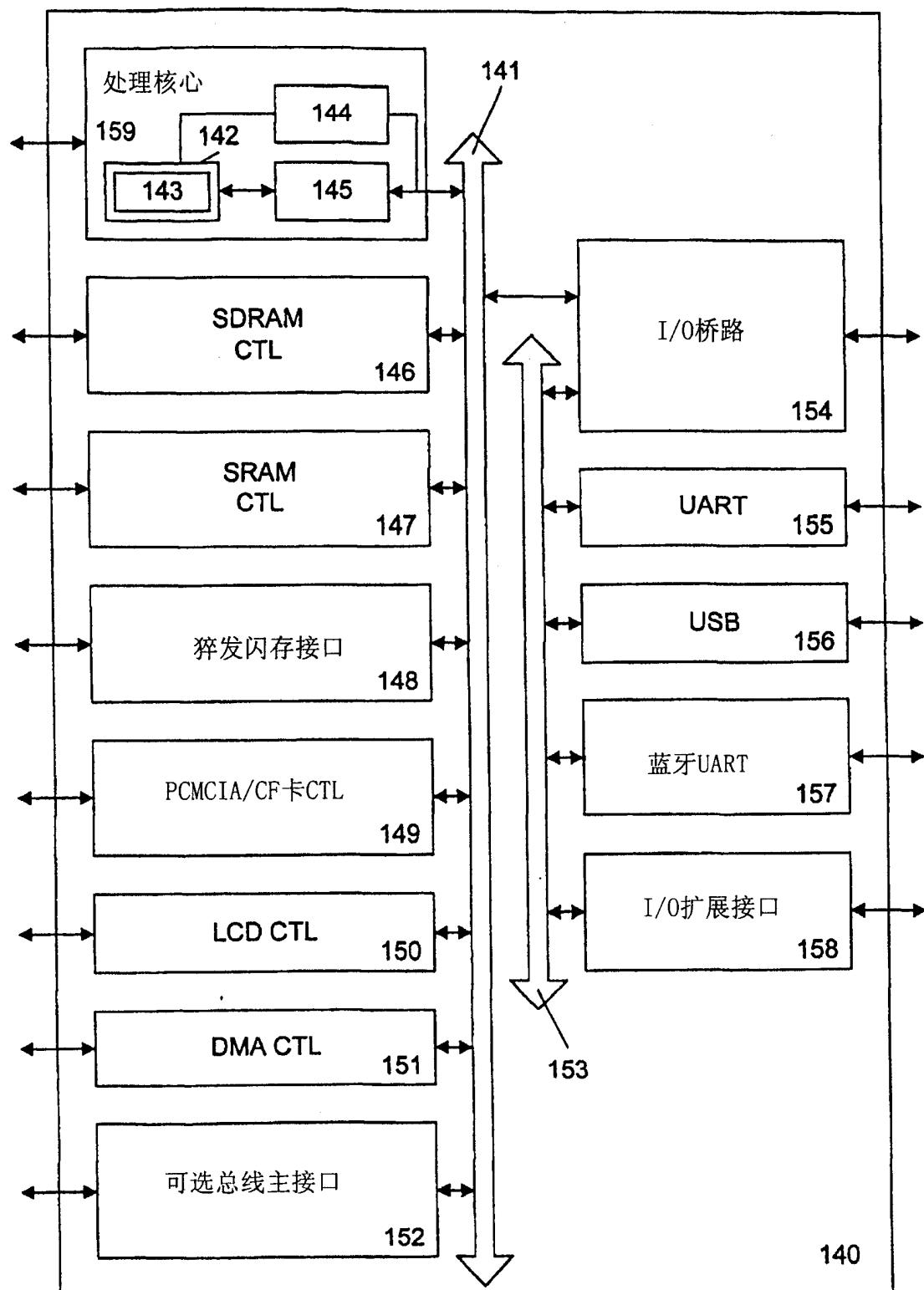


图 1B

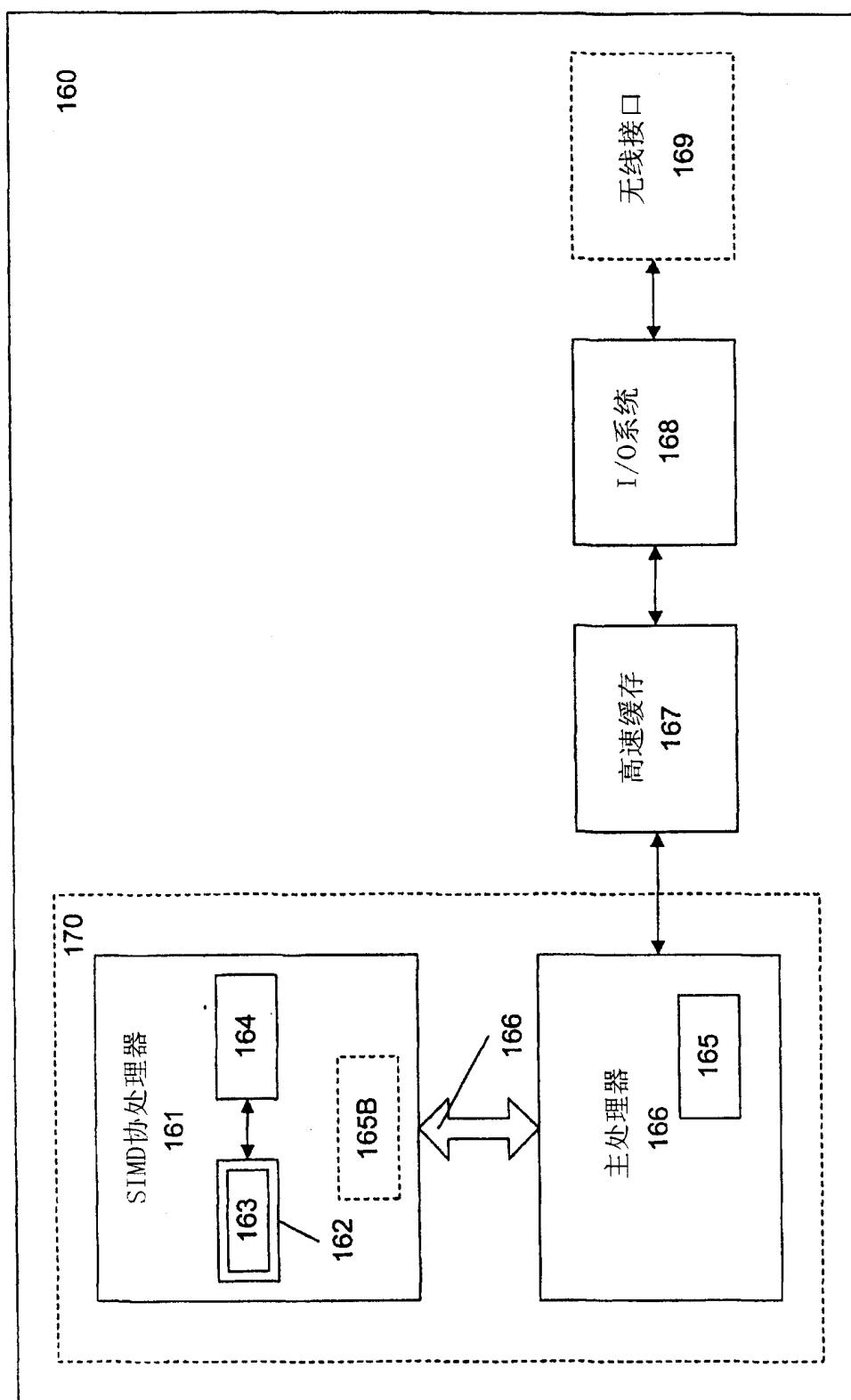


图 1C

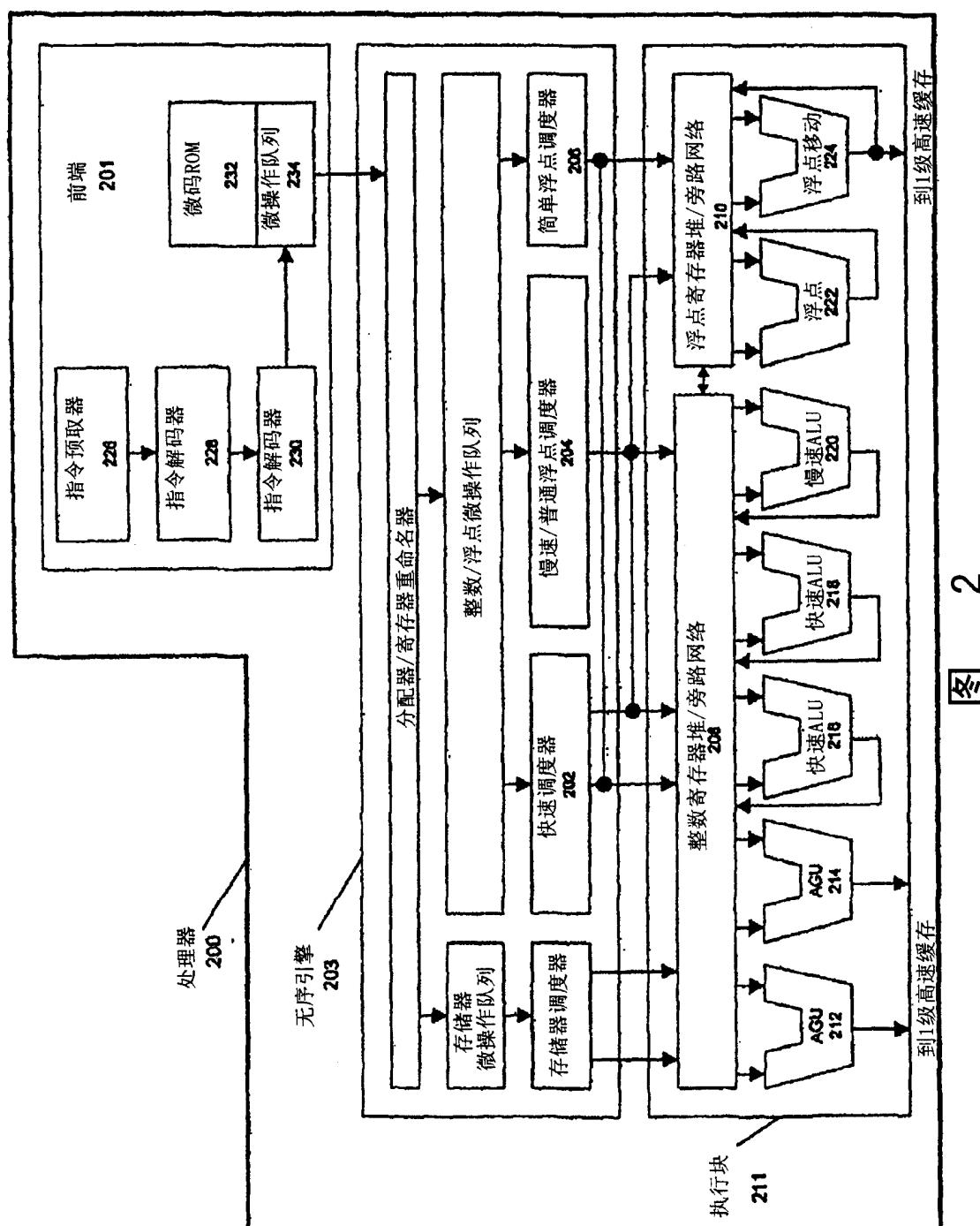
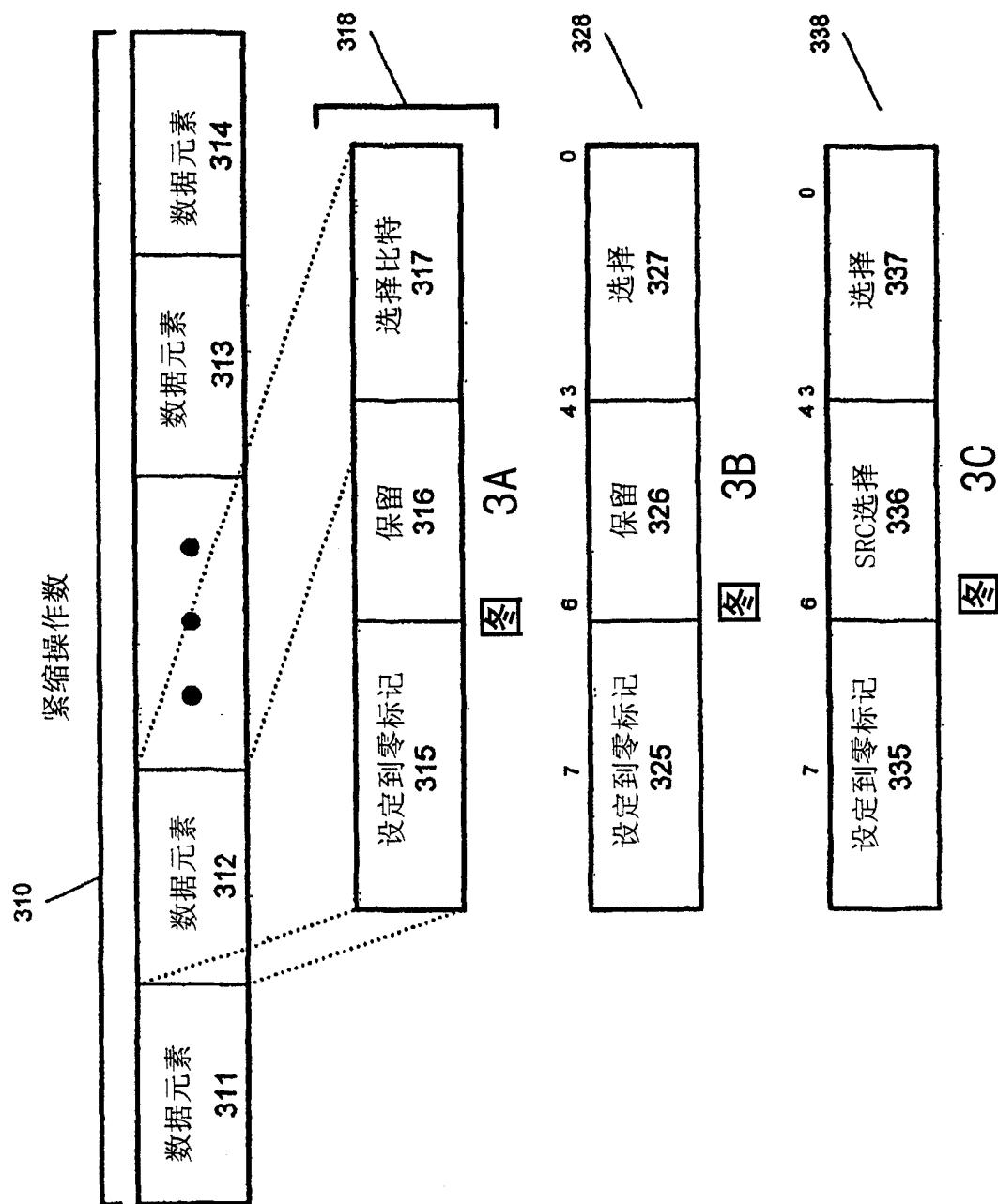
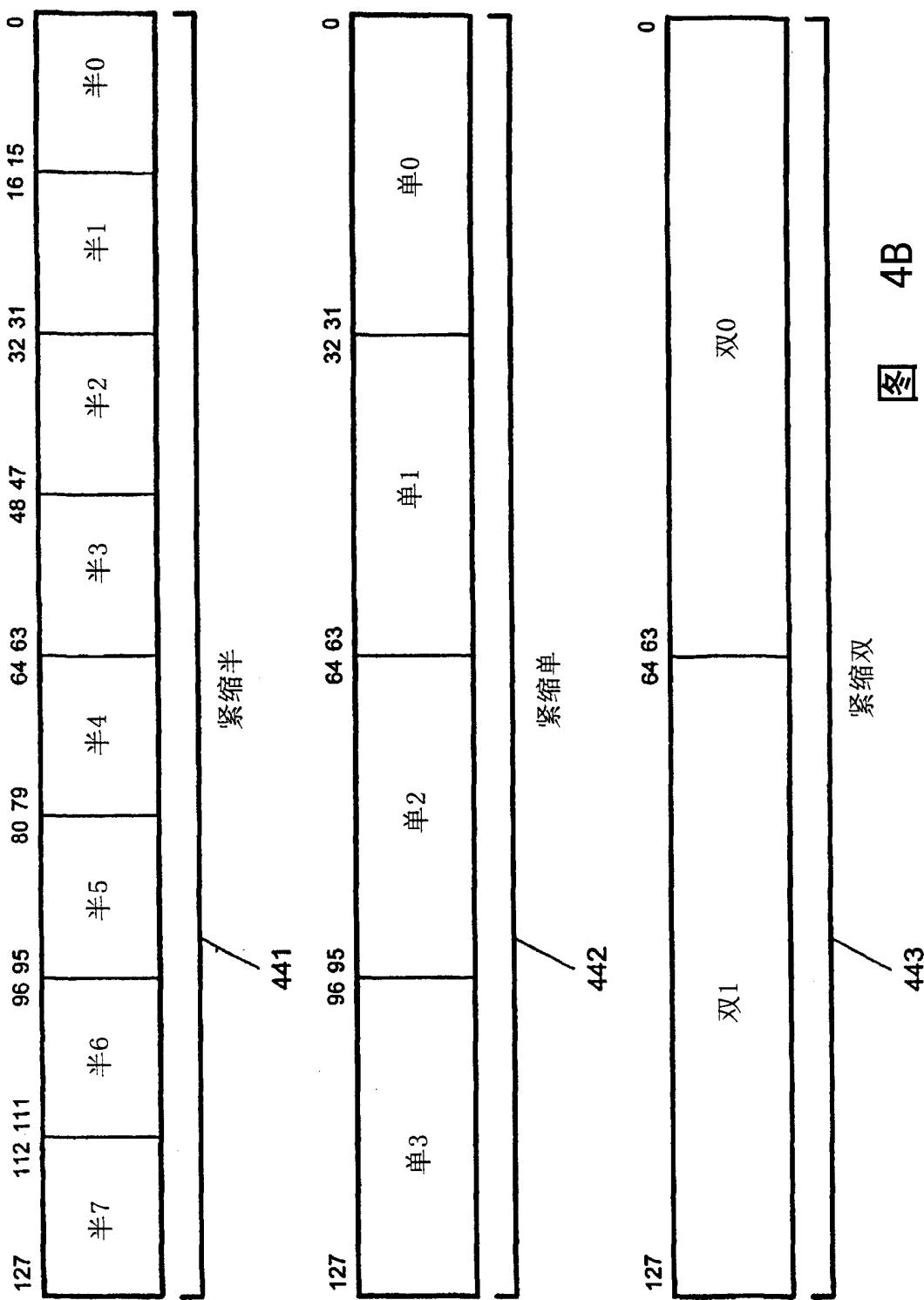


图 2



127	120 119	112 111	104 103	96 95	32 31	24 23	16 15	8 7	0
字节 15	字节 14	字节 13	字节 12	字节 11	字节 3	字节 2	字节 1	字节 0	
127	112 111	96 95			32 31	16 15	0		
字 7	字 6				字 1		字 0		
127	96 95								
双字 3							双字 0		
127	32 31								
420									
410									
127									
紧缩字节									
430									
紧缩双字									

图 4A



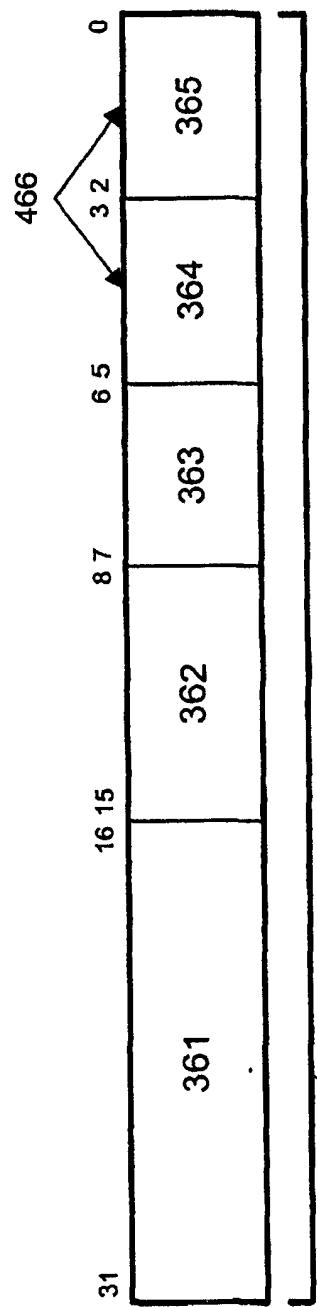


图 4C

460

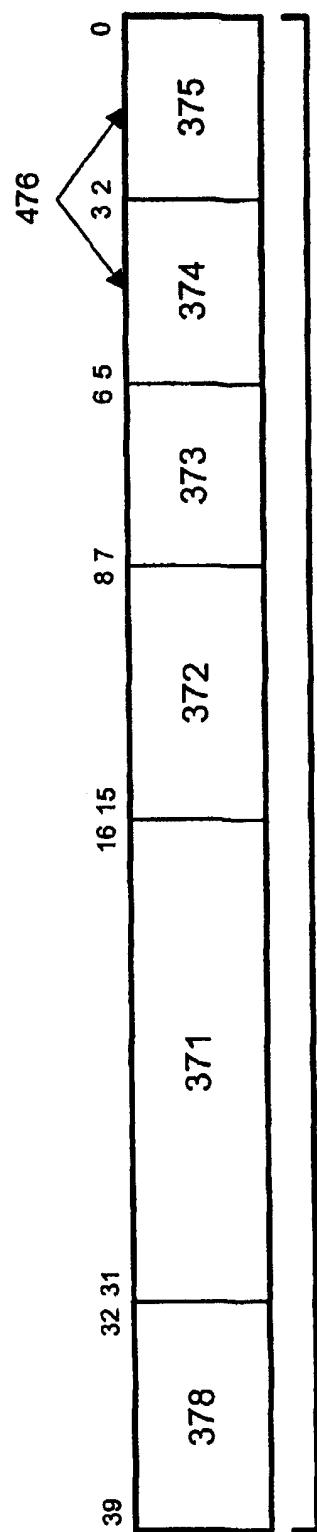
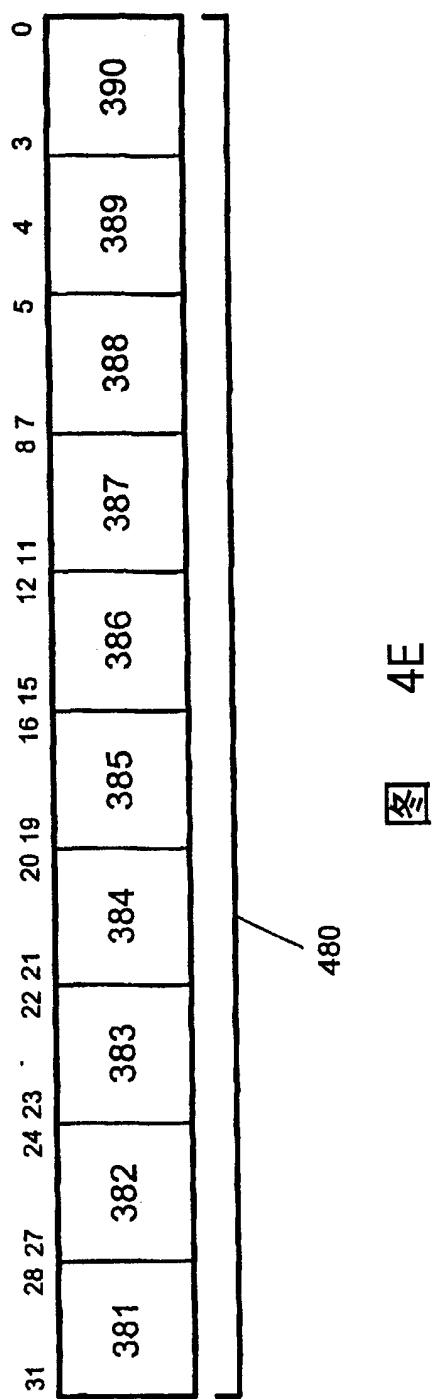
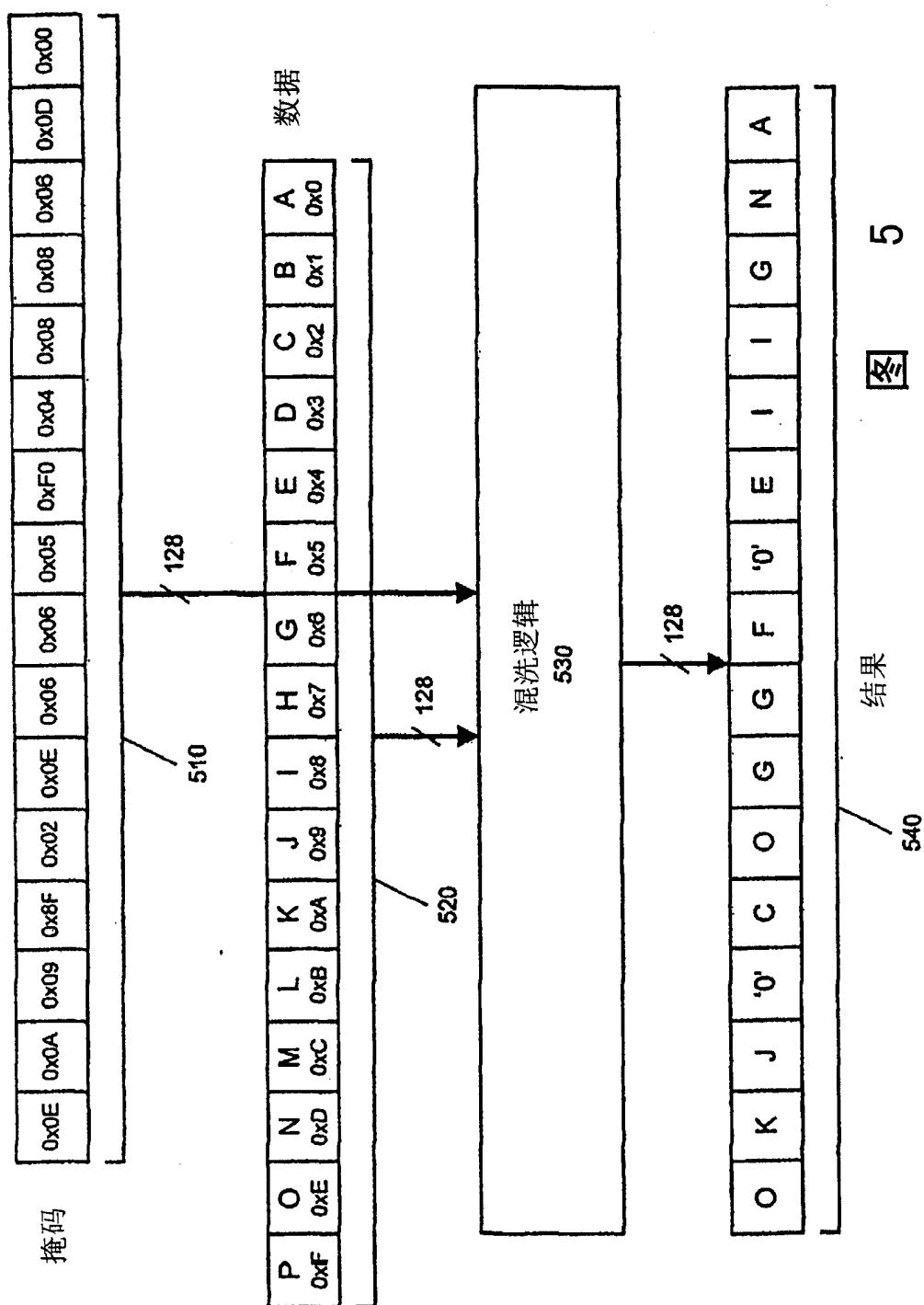
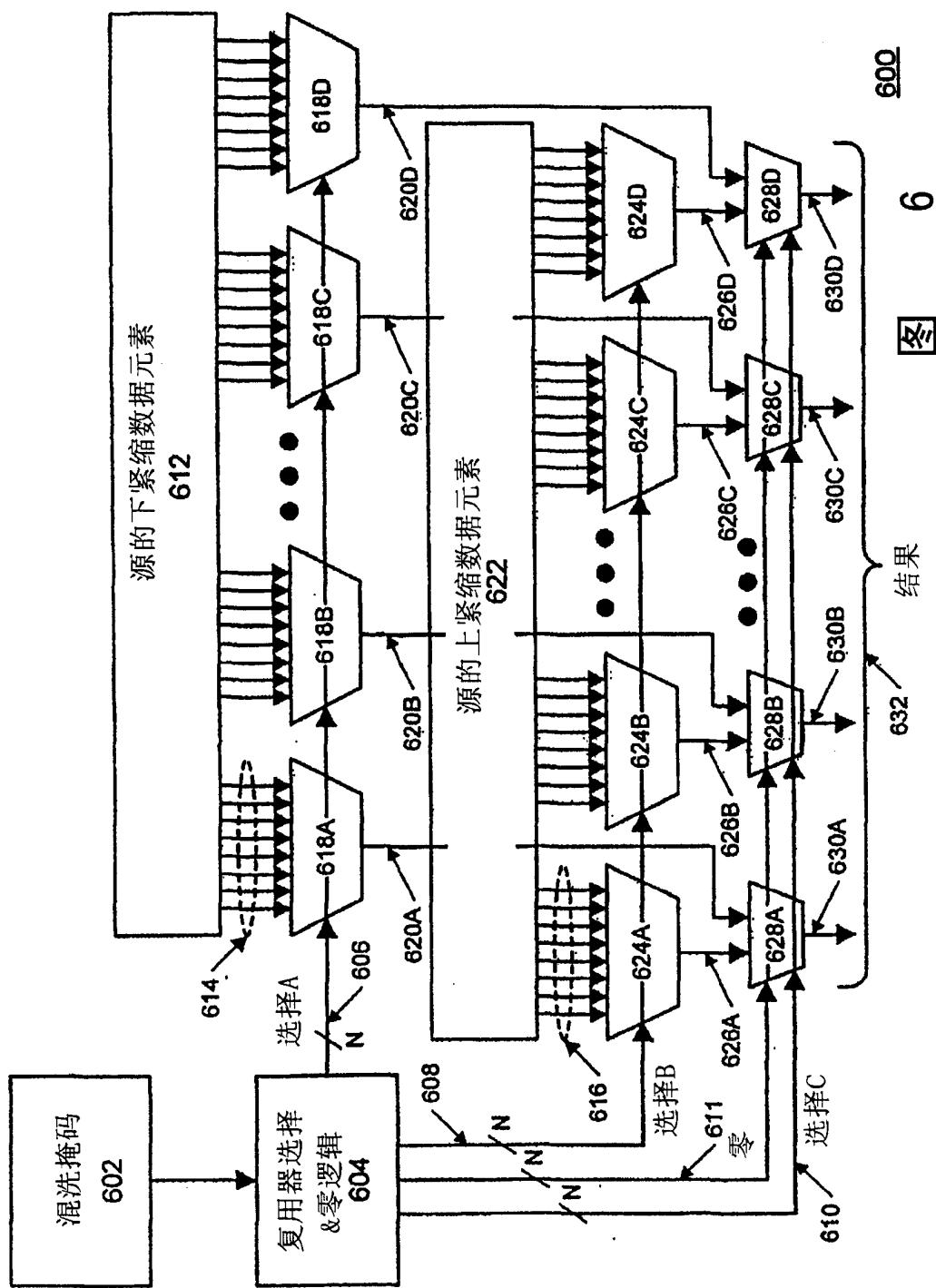


图 4D

470







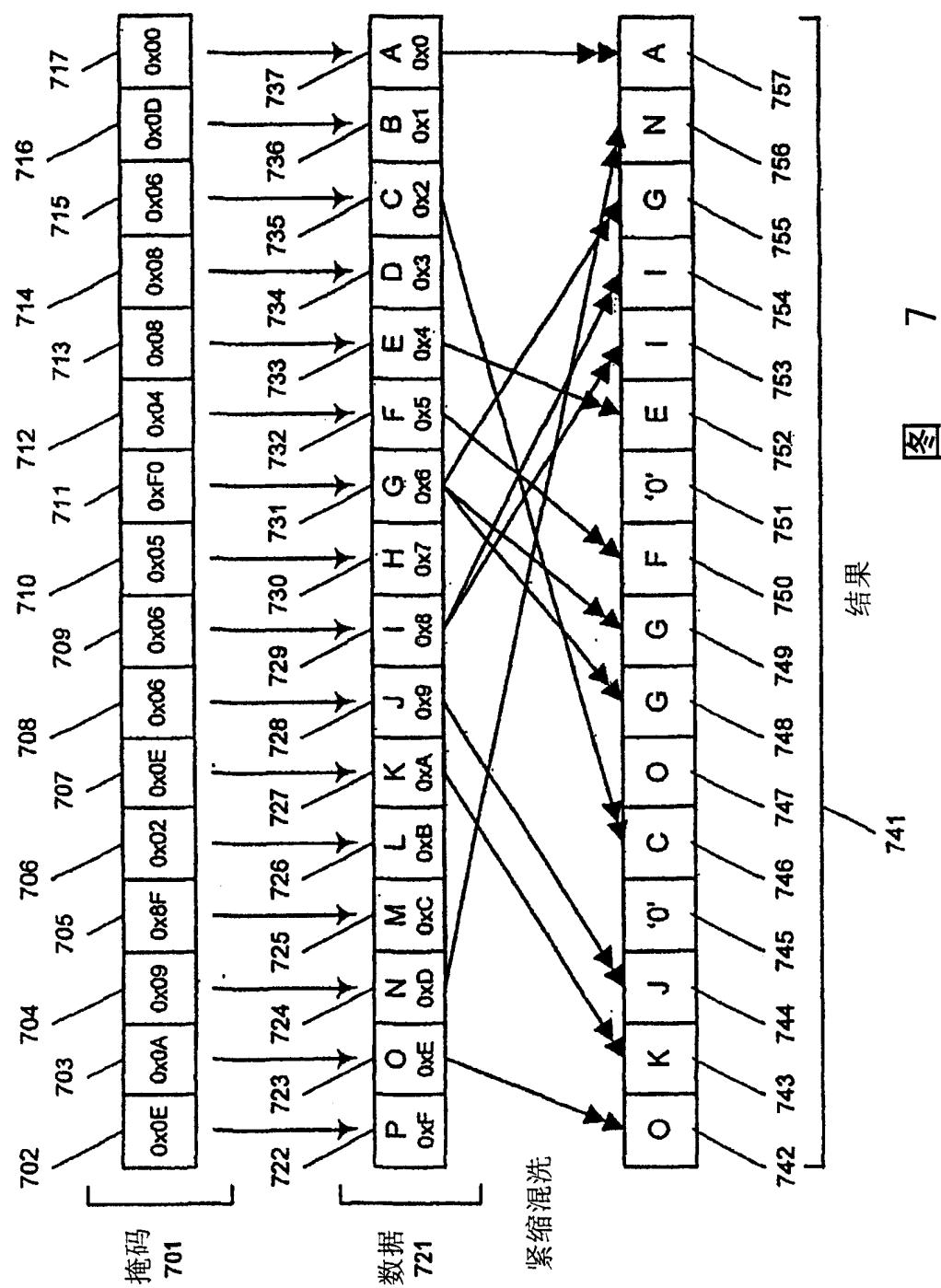
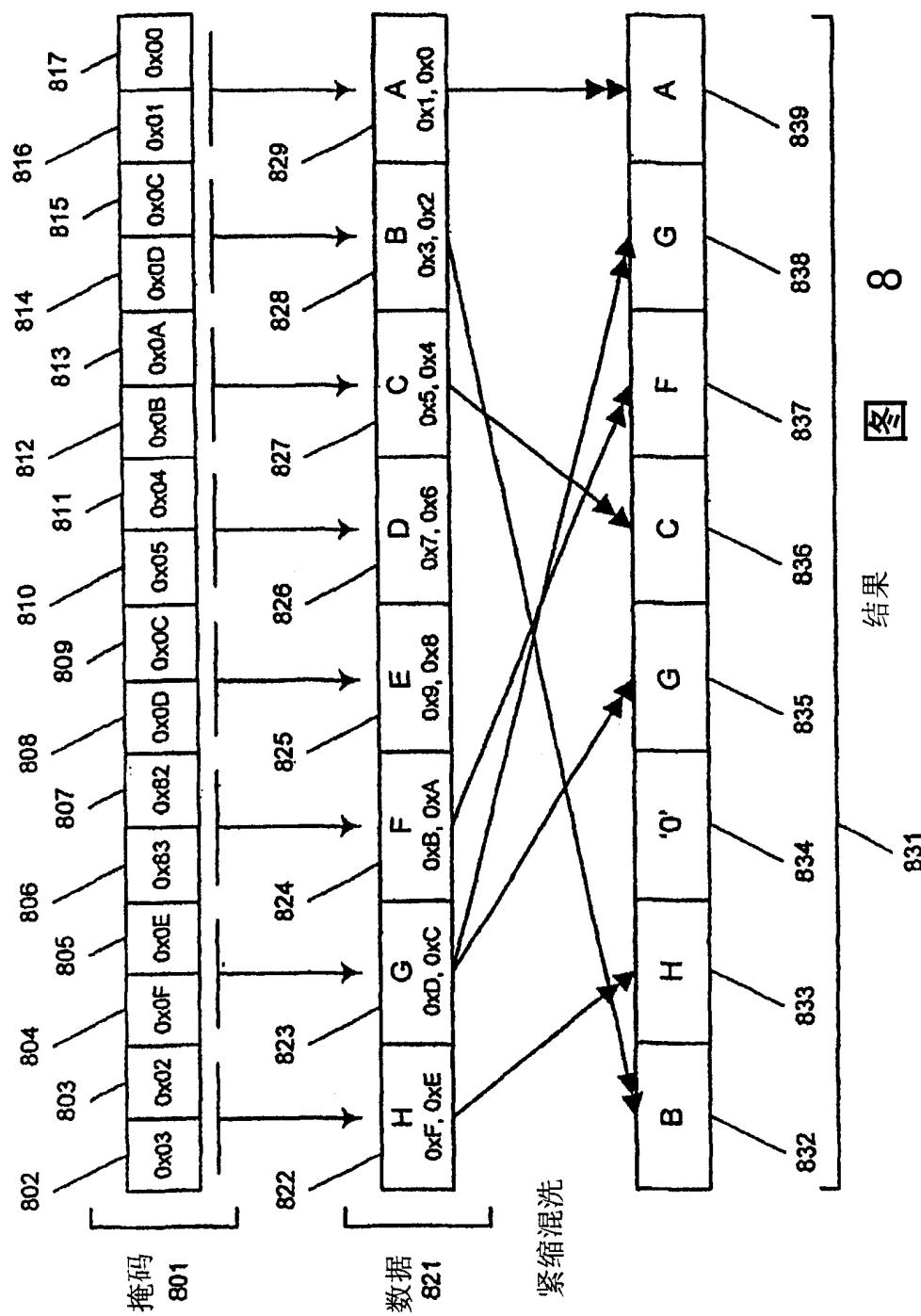
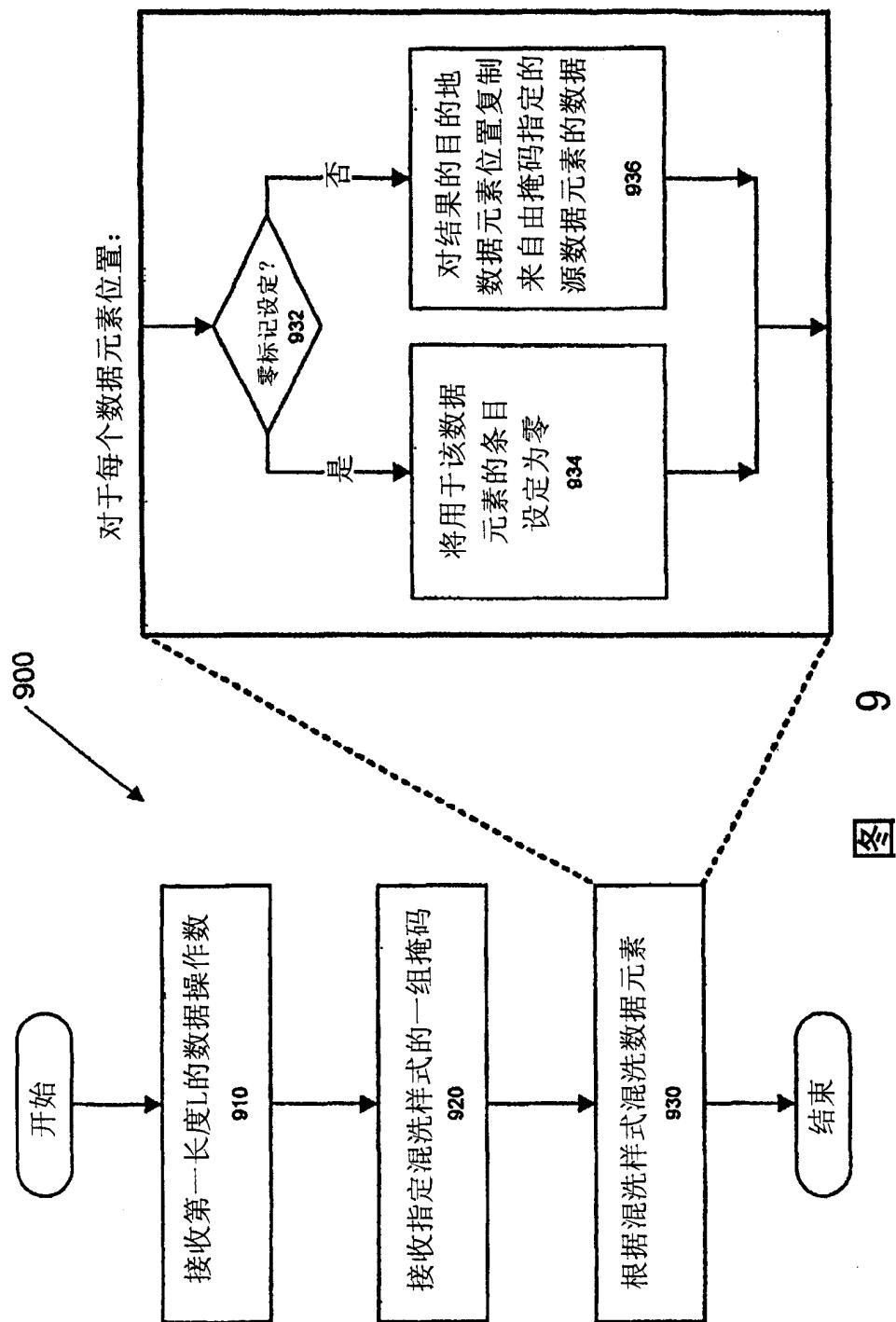
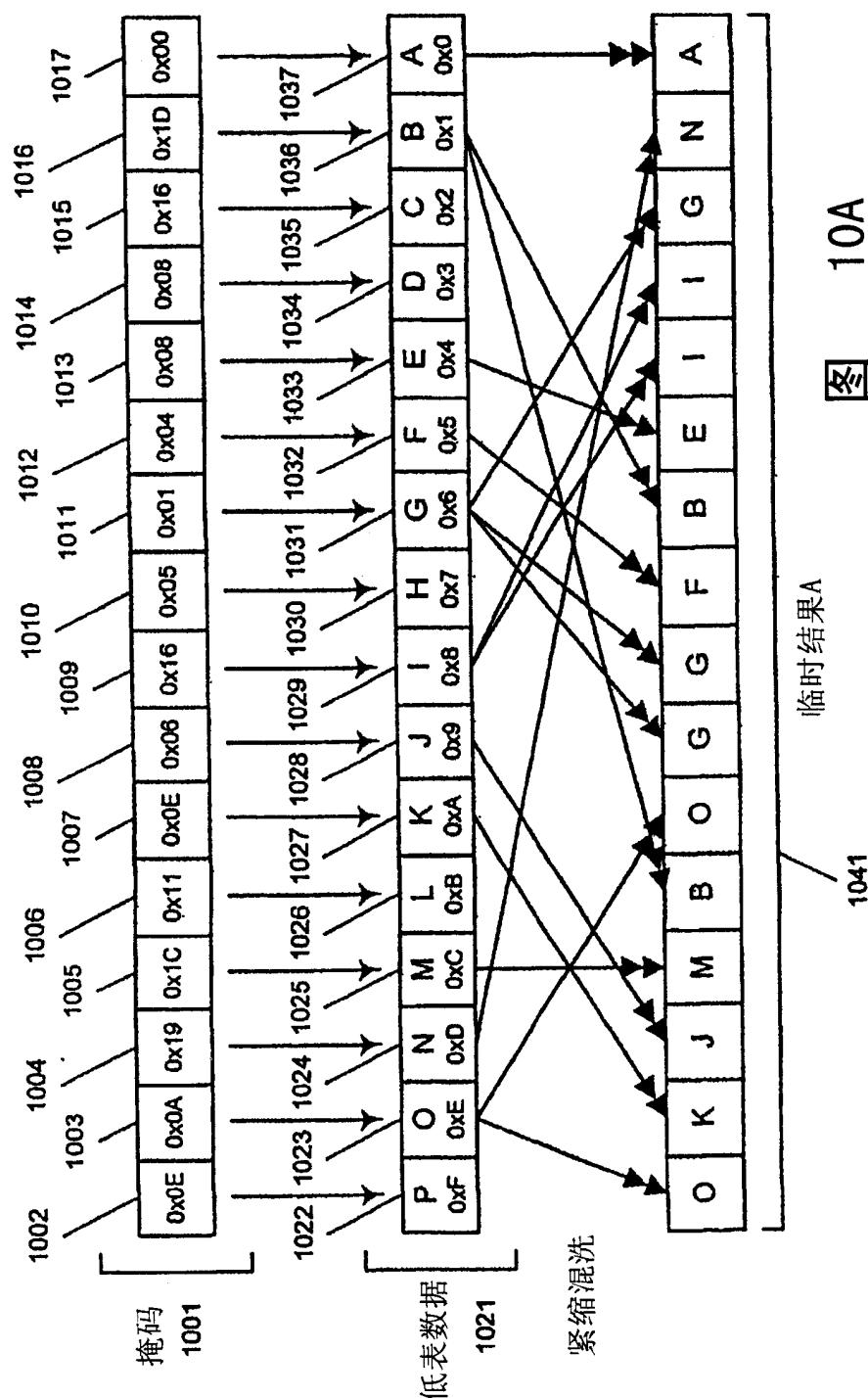


图 7







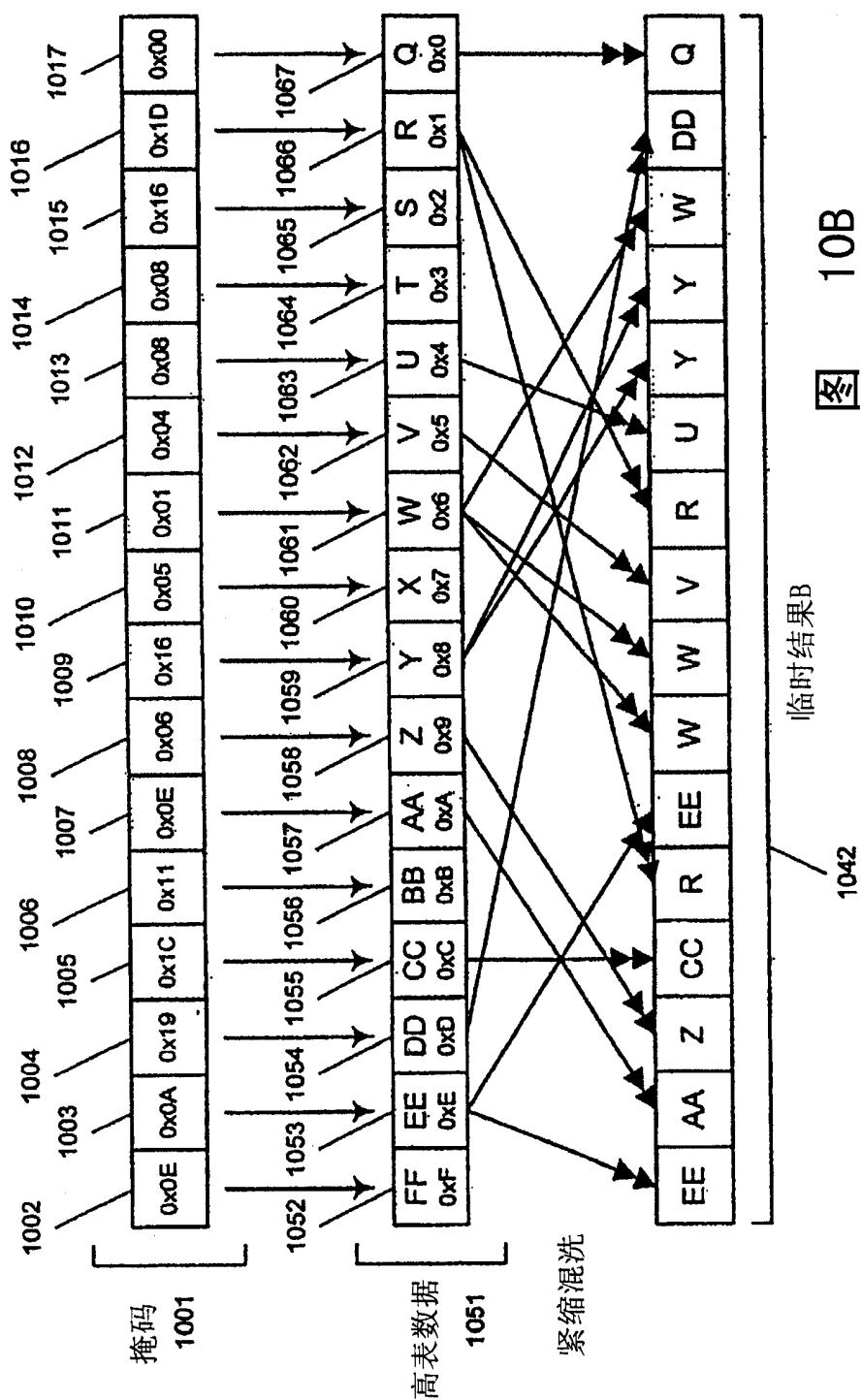
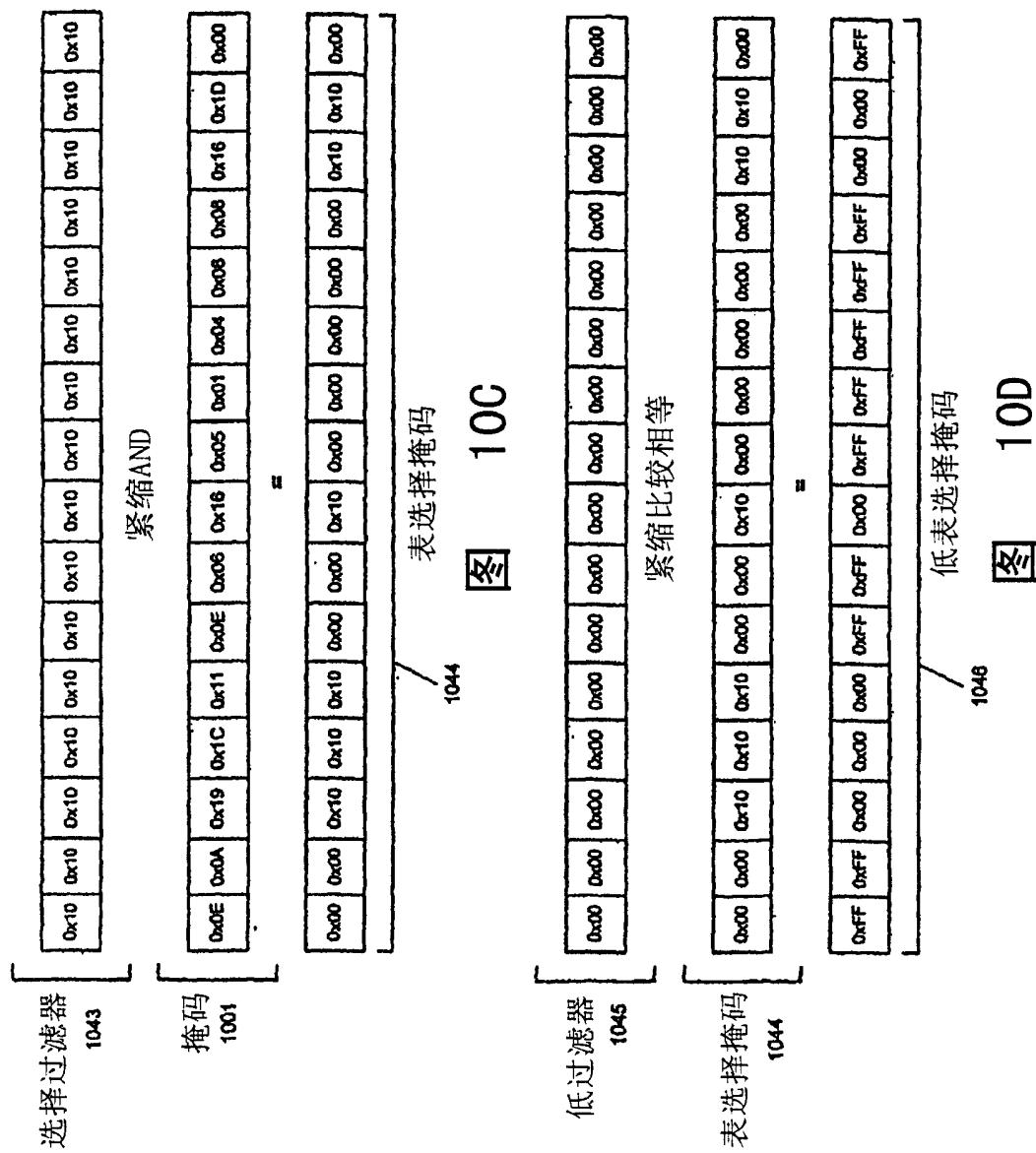
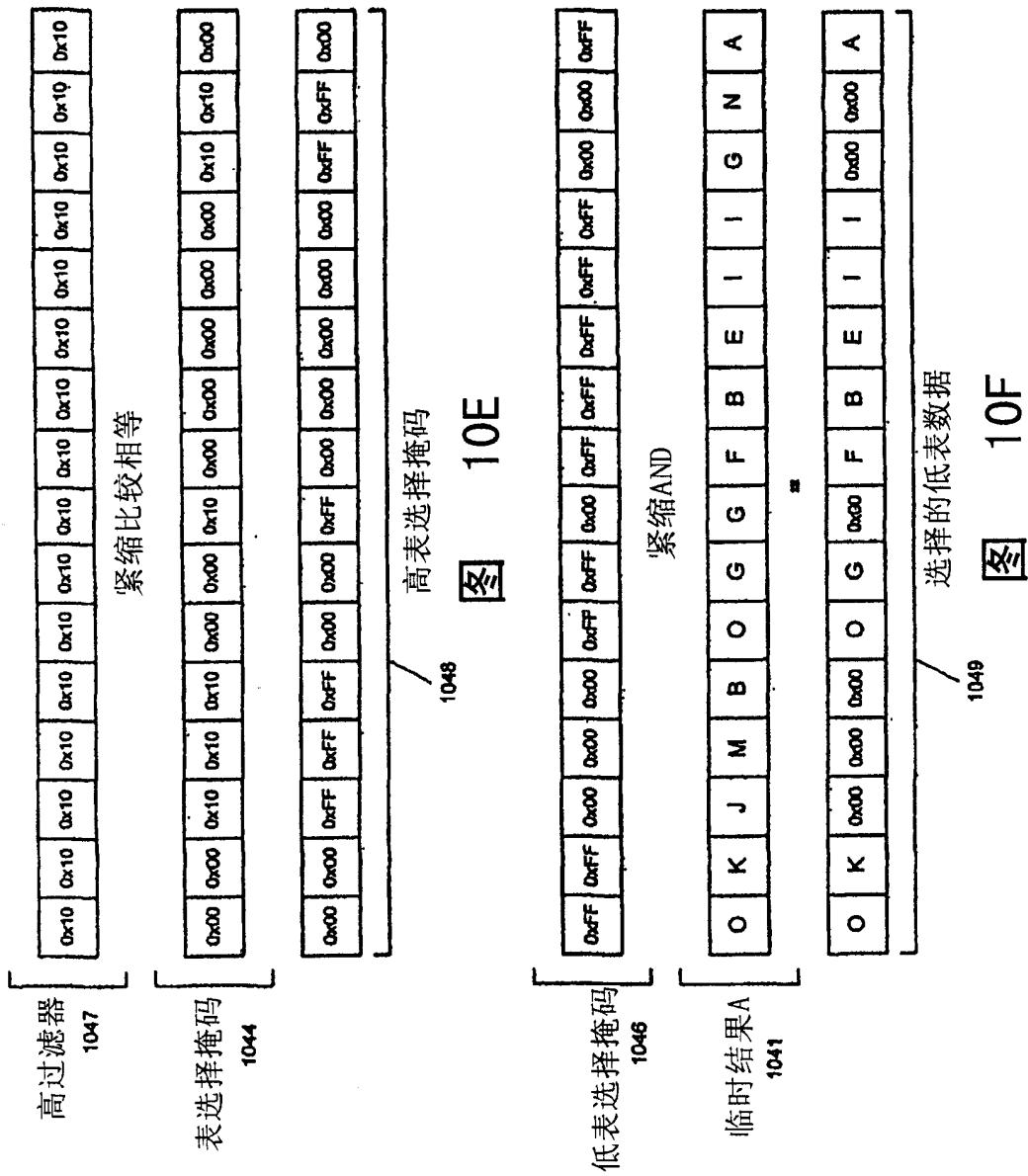
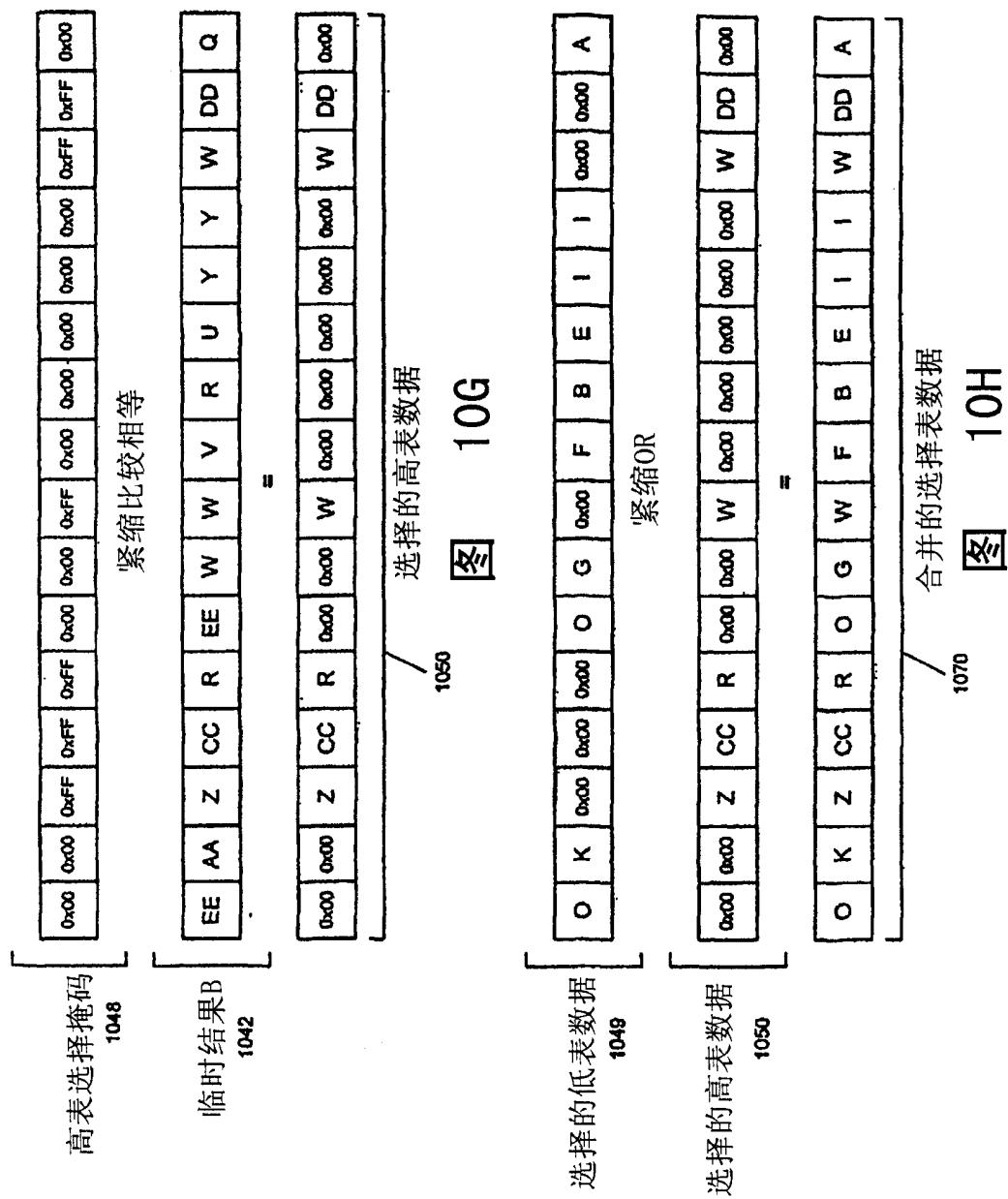
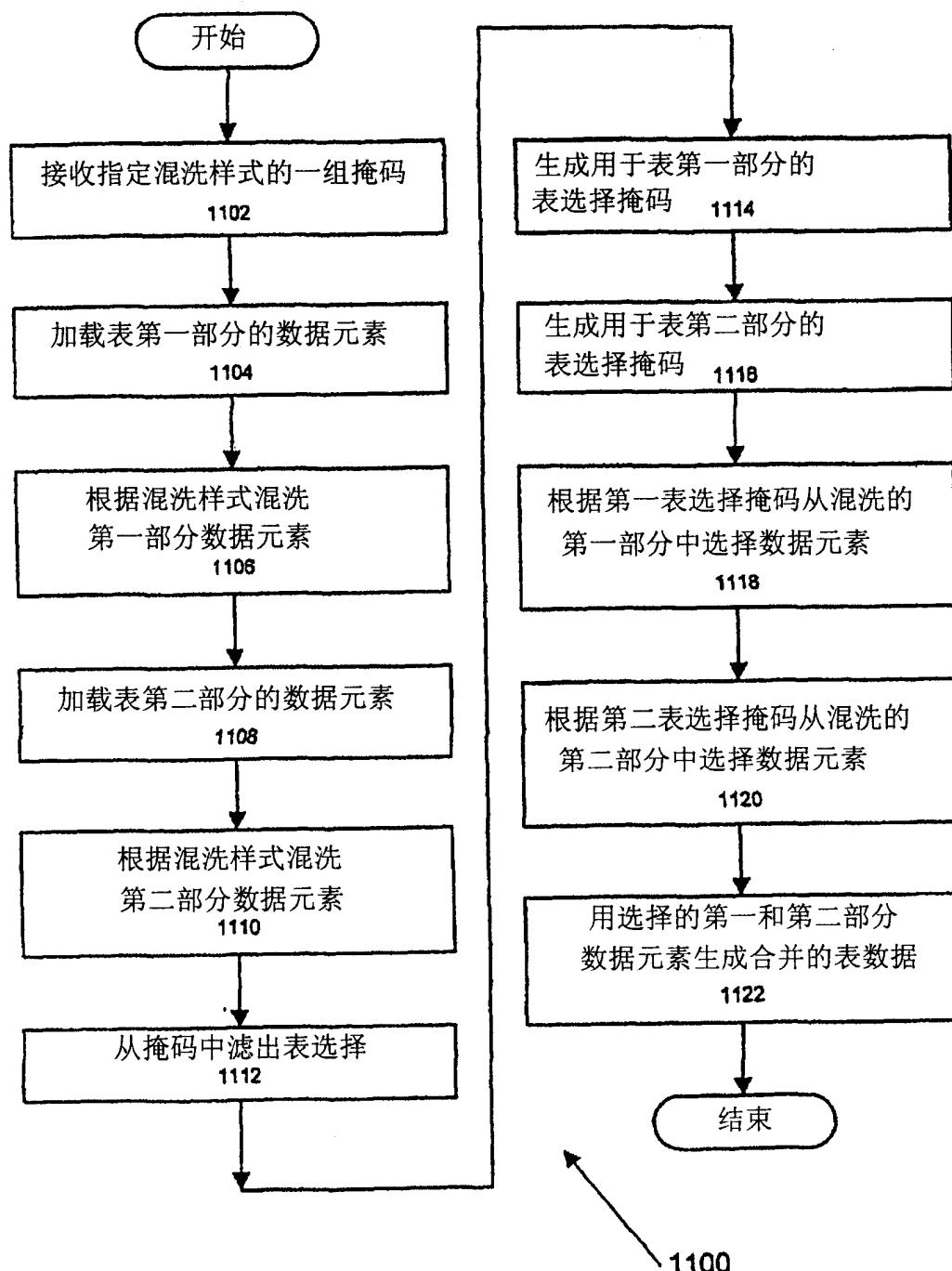


图 10B









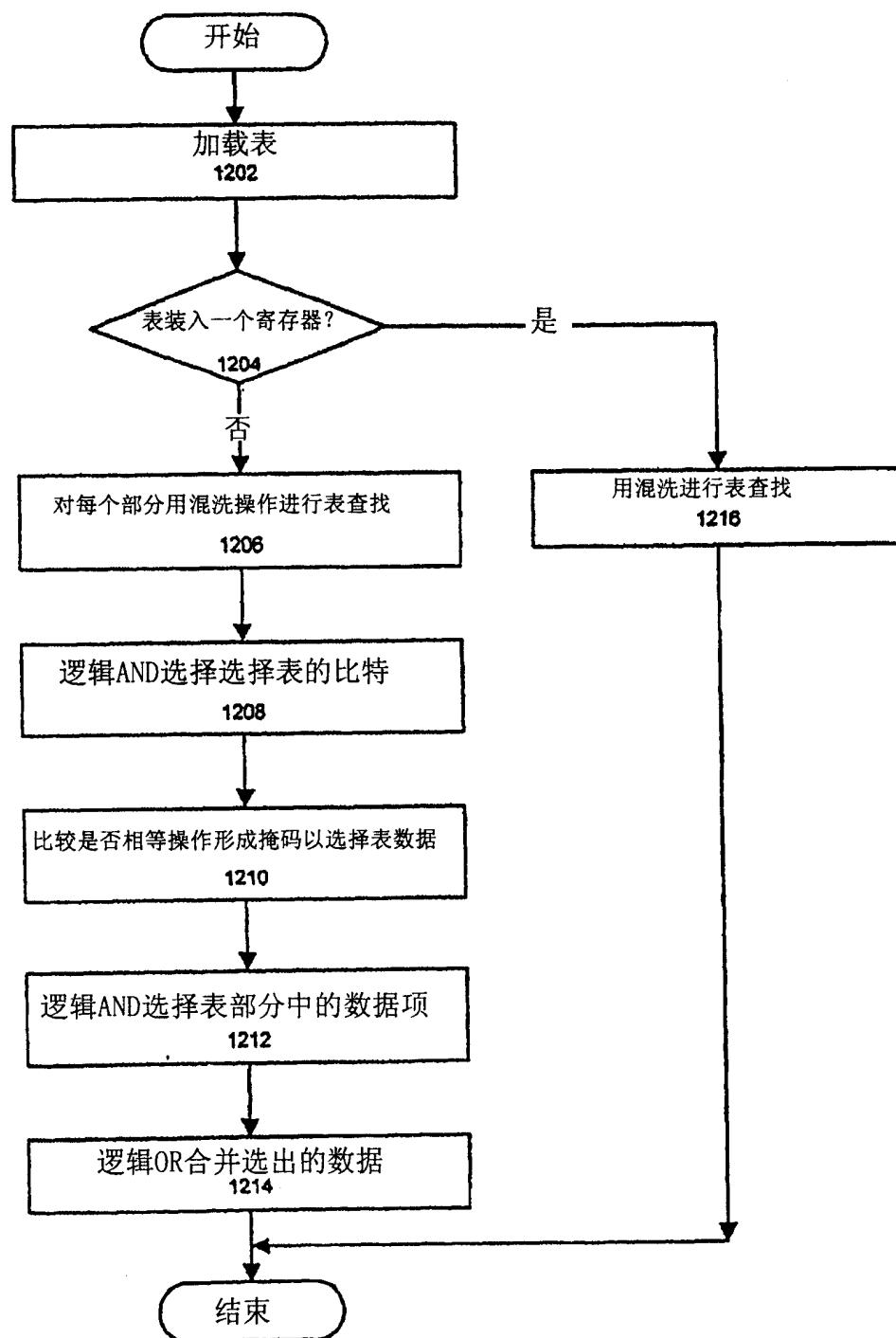


图 12

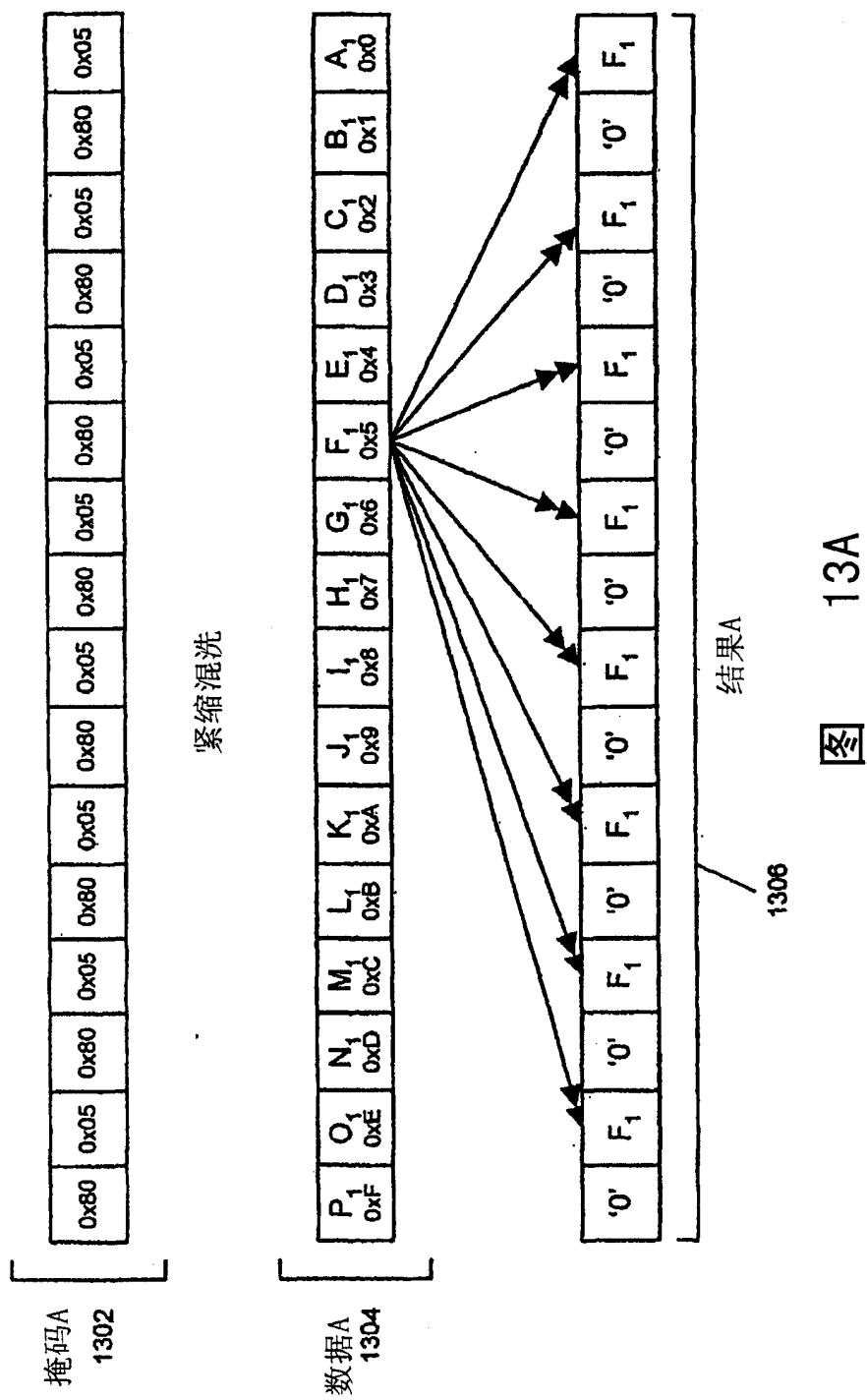
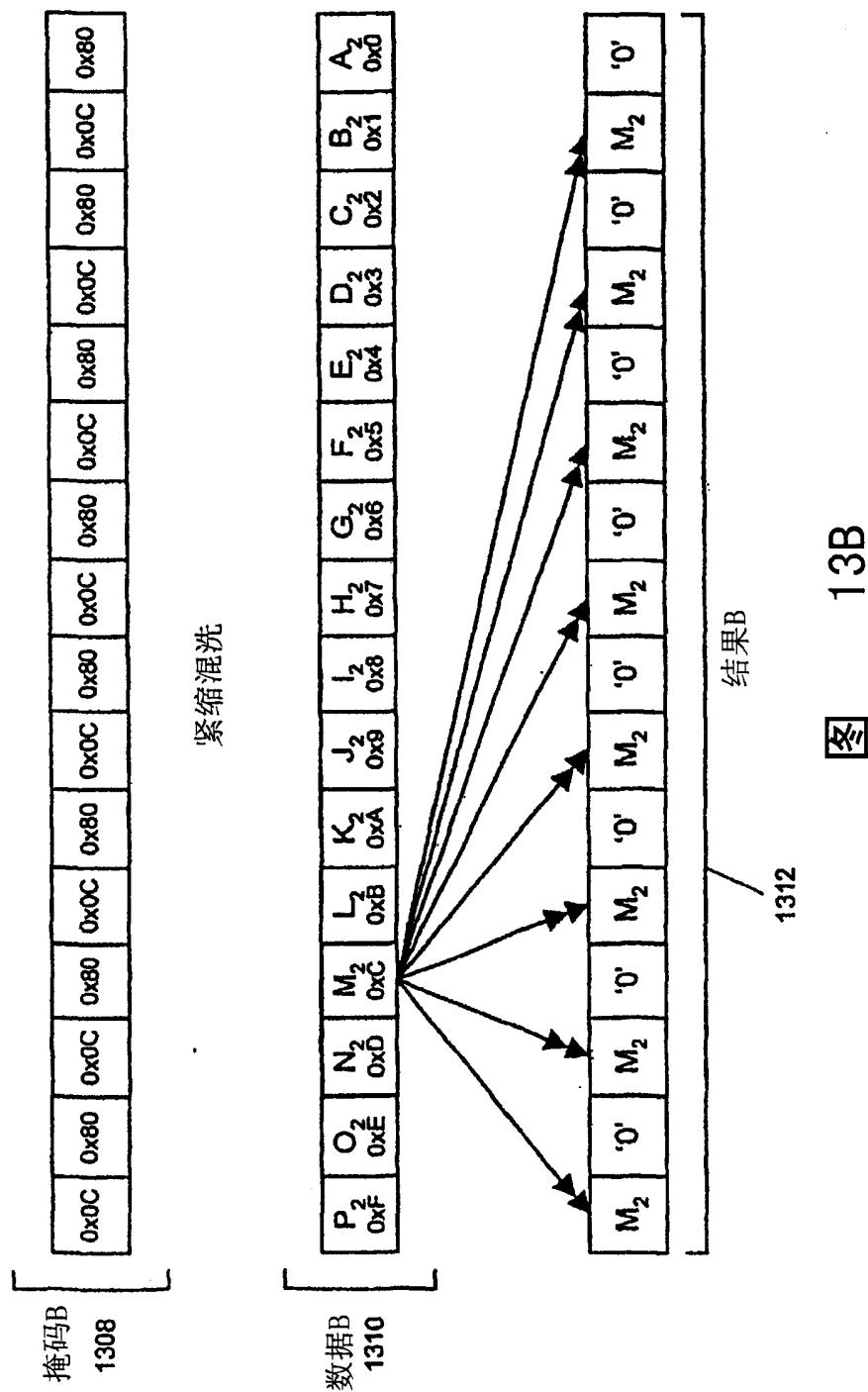


图 13A



结果A	1306	紧缩OR	<table border="1"> <tr><td>'0'</td><td>F₁</td><td>'0'</td><td>F₁</td><td>'0'</td><td>F₁</td><td>'0'</td><td>F₁</td><td>'0'</td><td>F₁</td><td>'0'</td><td>F₁</td><td>'0'</td><td>F₁</td><td>'0'</td><td>F₁</td></tr> </table>	'0'	F ₁														
'0'	F ₁	'0'	F ₁	'0'	F ₁	'0'	F ₁	'0'	F ₁	'0'	F ₁	'0'	F ₁	'0'	F ₁				
结果B	1308	=	<table border="1"> <tr><td>M₂</td><td>'0'</td><td>M₂</td><td>'0'</td><td>M₂</td><td>'0'</td><td>M₂</td><td>'0'</td><td>M₂</td><td>'0'</td><td>M₂</td><td>'0'</td><td>M₂</td><td>'0'</td><td>M₂</td><td>'0'</td></tr> </table>	M ₂	'0'														
M ₂	'0'	M ₂	'0'	M ₂	'0'	M ₂	'0'	M ₂	'0'	M ₂	'0'	M ₂	'0'	M ₂	'0'				
	1314	交错结果	<table border="1"> <tr><td>M₂</td><td>F₁</td><td>M₂</td><td>F₁</td><td>M₂</td><td>F₁</td><td>M₂</td><td>F₁</td><td>M₂</td><td>F₁</td><td>M₂</td><td>F₁</td><td>M₂</td><td>F₁</td><td>M₂</td><td>F₁</td></tr> </table>	M ₂	F ₁														
M ₂	F ₁	M ₂	F ₁	M ₂	F ₁	M ₂	F ₁	M ₂	F ₁	M ₂	F ₁	M ₂	F ₁	M ₂	F ₁				

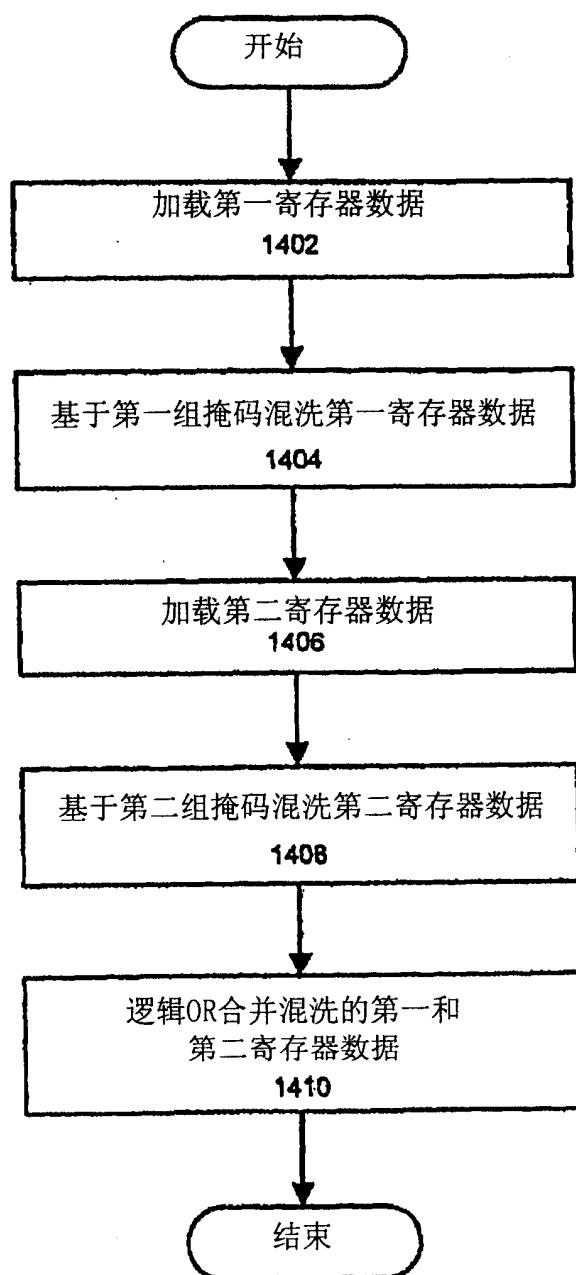


图 14

掩码A 1502	[<table border="1"><tr><td>0x05</td><td>0x80</td><td>0x04</td><td>0x80</td><td>0x03</td><td>0x80</td><td>0x02</td><td>0x80</td><td>0x01</td><td>0x80</td><td>0x00</td><td>0x80</td><td>0x00</td></tr></table>]	0x05	0x80	0x04	0x80	0x03	0x80	0x02	0x80	0x01	0x80	0x00	0x80	0x00
0x05	0x80	0x04	0x80	0x03	0x80	0x02	0x80	0x01	0x80	0x00	0x80	0x00		
掩码B 1504	[<table border="1"><tr><td>0x80</td><td>0x04</td><td>0x80</td><td>0x03</td><td>0x80</td><td>0x02</td><td>0x80</td><td>0x01</td><td>0x80</td><td>0x00</td><td>0x80</td><td>0x00</td><td>0x80</td></tr></table>]	0x80	0x04	0x80	0x03	0x80	0x02	0x80	0x01	0x80	0x00	0x80	0x00	0x80
0x80	0x04	0x80	0x03	0x80	0x02	0x80	0x01	0x80	0x00	0x80	0x00	0x80		
掩码C 1506	[<table border="1"><tr><td>0x00</td><td>0x04</td><td>0x80</td><td>0x03</td><td>0x80</td><td>0x02</td><td>0x80</td><td>0x01</td><td>0x80</td><td>0x00</td><td>0x80</td><td>0x00</td><td>0x80</td></tr></table>]	0x00	0x04	0x80	0x03	0x80	0x02	0x80	0x01	0x80	0x00	0x80	0x00	0x80
0x00	0x04	0x80	0x03	0x80	0x02	0x80	0x01	0x80	0x00	0x80	0x00	0x80		

冬 15A

数据A 1512	[<table border="1"><tr><td>R₁₅ 0xF</td><td>R₁₄ 0xE</td><td>R₁₃ 0xD</td><td>R₁₂ 0xC</td><td>R₁₁ 0xB</td><td>R₁₀ 0xA</td><td>R₉ 0x9</td><td>R₈ 0x8</td><td>R₇ 0x7</td><td>R₆ 0x6</td><td>R₅ 0x5</td><td>R₄ 0x4</td><td>R₃ 0x3</td><td>R₂ 0x2</td><td>R₁ 0x1</td><td>R₀ 0x0</td></tr></table>]	R ₁₅ 0xF	R ₁₄ 0xE	R ₁₃ 0xD	R ₁₂ 0xC	R ₁₁ 0xB	R ₁₀ 0xA	R ₉ 0x9	R ₈ 0x8	R ₇ 0x7	R ₆ 0x6	R ₅ 0x5	R ₄ 0x4	R ₃ 0x3	R ₂ 0x2	R ₁ 0x1	R ₀ 0x0
R ₁₅ 0xF	R ₁₄ 0xE	R ₁₃ 0xD	R ₁₂ 0xC	R ₁₁ 0xB	R ₁₀ 0xA	R ₉ 0x9	R ₈ 0x8	R ₇ 0x7	R ₆ 0x6	R ₅ 0x5	R ₄ 0x4	R ₃ 0x3	R ₂ 0x2	R ₁ 0x1	R ₀ 0x0		
数据B 1514	[<table border="1"><tr><td>G₁₅ 0xF</td><td>G₁₄ 0xE</td><td>G₁₃ 0xD</td><td>G₁₂ 0xC</td><td>G₁₁ 0xB</td><td>G₁₀ 0xA</td><td>G₉ 0x9</td><td>G₈ 0x8</td><td>G₇ 0x7</td><td>G₆ 0x6</td><td>G₅ 0x5</td><td>G₄ 0x4</td><td>G₃ 0x3</td><td>G₂ 0x2</td><td>G₁ 0x1</td><td>G₀ 0x0</td></tr></table>]	G ₁₅ 0xF	G ₁₄ 0xE	G ₁₃ 0xD	G ₁₂ 0xC	G ₁₁ 0xB	G ₁₀ 0xA	G ₉ 0x9	G ₈ 0x8	G ₇ 0x7	G ₆ 0x6	G ₅ 0x5	G ₄ 0x4	G ₃ 0x3	G ₂ 0x2	G ₁ 0x1	G ₀ 0x0
G ₁₅ 0xF	G ₁₄ 0xE	G ₁₃ 0xD	G ₁₂ 0xC	G ₁₁ 0xB	G ₁₀ 0xA	G ₉ 0x9	G ₈ 0x8	G ₇ 0x7	G ₆ 0x6	G ₅ 0x5	G ₄ 0x4	G ₃ 0x3	G ₂ 0x2	G ₁ 0x1	G ₀ 0x0		
数据C 1516	[<table border="1"><tr><td>B₁₆ 0xF</td><td>B₁₄ 0xE</td><td>B₁₃ 0xD</td><td>B₁₂ 0xC</td><td>B₁₁ 0xB</td><td>B₁₀ 0xA</td><td>B₉ 0x9</td><td>B₈ 0x8</td><td>B₇ 0x7</td><td>B₆ 0x6</td><td>B₅ 0x5</td><td>B₄ 0x4</td><td>B₃ 0x3</td><td>B₂ 0x2</td><td>B₁ 0x1</td><td>B₀ 0x0</td></tr></table>]	B ₁₆ 0xF	B ₁₄ 0xE	B ₁₃ 0xD	B ₁₂ 0xC	B ₁₁ 0xB	B ₁₀ 0xA	B ₉ 0x9	B ₈ 0x8	B ₇ 0x7	B ₆ 0x6	B ₅ 0x5	B ₄ 0x4	B ₃ 0x3	B ₂ 0x2	B ₁ 0x1	B ₀ 0x0
B ₁₆ 0xF	B ₁₄ 0xE	B ₁₃ 0xD	B ₁₂ 0xC	B ₁₁ 0xB	B ₁₀ 0xA	B ₉ 0x9	B ₈ 0x8	B ₇ 0x7	B ₆ 0x6	B ₅ 0x5	B ₄ 0x4	B ₃ 0x3	B ₂ 0x2	B ₁ 0x1	B ₀ 0x0		

冬 15B

掩码数据A 1522	R ₅	·Φ	Ψ	R ₄	·Φ	Ψ	R ₃	·Φ	Ψ	R ₂	·Φ	Ψ	R ₁	·Φ	Ψ	R ₀
---------------	----------------	----	---	----------------	----	---	----------------	----	---	----------------	----	---	----------------	----	---	----------------

15C

掩码数据B 1524	Φ	·Φ	G ₄	Φ	·Φ	G ₃	Φ	·Φ	G ₂	Φ	·Φ	G ₁	Φ	·Φ	G ₀	Ψ
---------------	---	----	----------------	---	----	----------------	---	----	----------------	---	----	----------------	---	----	----------------	---

15D

掩码数据C 1526	Φ	B ₄	·Φ	Φ	B ₃	·Φ	Φ	B ₂	·Φ	Φ	B ₁	·Φ	Φ	B ₀	·Φ	Ψ
---------------	---	----------------	----	---	----------------	----	---	----------------	----	---	----------------	----	---	----------------	----	---

15E

交错A&B数据 1530	R ₅	Φ	G ₄	R ₄	Φ	G ₃	R ₃	Φ	G ₂	R ₂	Φ	G ₁	R ₁	Φ	G ₀	R ₀
-----------------	----------------	---	----------------	----------------	---	----------------	----------------	---	----------------	----------------	---	----------------	----------------	---	----------------	----------------

15F

交错A、B&C数据 1532	R ₆	B ₄	G ₄	R ₄	B ₃	G ₃	R ₃	B ₂	G ₂	R ₂	B ₁	G ₁	R ₁	B ₀	G ₀	R ₀
-------------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------

15G

'0'	'0'	'0'	'0'	'0'	G ₁₅ 0xF	G ₁₄ 0xE	G ₁₃ 0xD	G ₁₂ 0xC	G ₁₁ 0xB	G ₁₀ 0xA	G ₉ 0x9	G ₈ 0x8	G ₇ 0x7	G ₆ 0x6	G ₅ 0x5
-----	-----	-----	-----	-----	------------------------	------------------------	------------------------	------------------------	------------------------	------------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----------------------

数据B'

1542

'0'	'0'	'0'	'0'	'0'	B ₁₅ 0xF	B ₁₄ 0xE	B ₁₃ 0xD	B ₁₂ 0xC	B ₁₁ 0xB	B ₁₀ 0xA	B ₉ 0x9	B ₈ 0x8	B ₇ 0x7	B ₆ 0x6	B ₅ 0x5
-----	-----	-----	-----	-----	------------------------	------------------------	------------------------	------------------------	------------------------	------------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----------------------

数据C'

1544

'0'	'0'	'0'	'0'	'0'	R ₁₅ 0xF	R ₁₄ 0xE	R ₁₃ 0xD	R ₁₂ 0xC	R ₁₁ 0xB	R ₁₀ 0xA	R ₉ 0x9	R ₈ 0x8	R ₇ 0x7	R ₆ 0x6	R ₅ 0x5
-----	-----	-----	-----	-----	------------------------	------------------------	------------------------	------------------------	------------------------	------------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----------------------

数据A'

1546

图 15H

G ₁₀	R ₁₀	B ₉	G ₉	R ₉	B ₈	G ₈	R ₈	B ₇	G ₇	R ₇	B ₆	G ₆	R ₆	B ₅	G ₅
-----------------	-----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------

交错A'、B' &C' 数据

1548

图 15I

'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

数据C',
1552

'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

数据A',
1554

'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

数据B',
1556

15J

B ₁₅	G ₁₅	R ₁₅	B ₁₄	G ₁₄	R ₁₄	B ₁₃	G ₁₃	R ₁₃	B ₁₂	G ₁₂	R ₁₂	B ₁₁	G ₁₁	R ₁₁	B ₁₀
-----------------	-----------------	-----------------	-----------------	-----------------	-----------------	-----------------	-----------------	-----------------	-----------------	-----------------	-----------------	-----------------	-----------------	-----------------	-----------------

交错A'、B'、C'，数据

15H

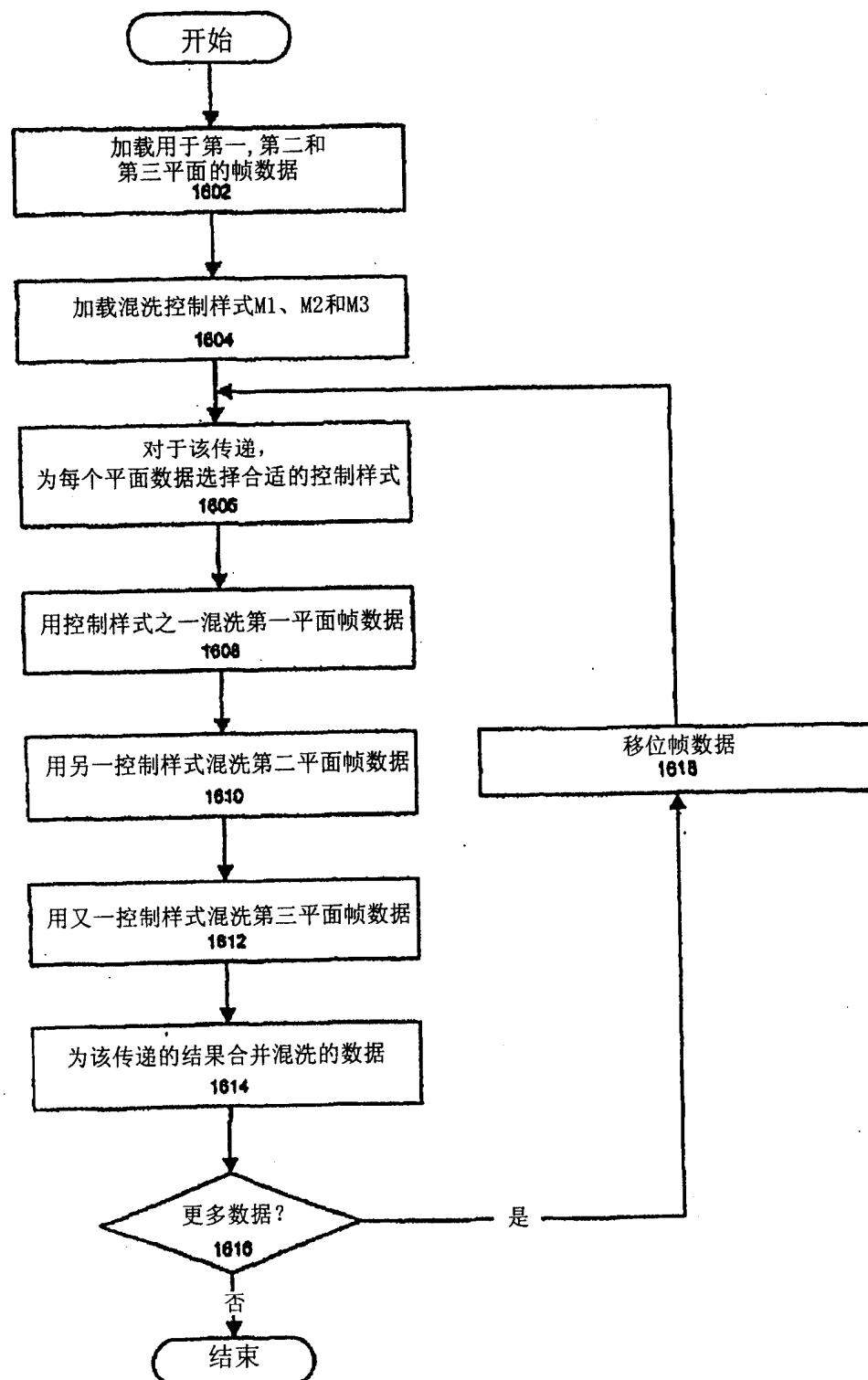


图 16