(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 110968346 A
(43)申请公布日 2020.04.07

(21)申请号 201910913268.7

(22)申请日 2019.09.25

(30)优先权数据
16/146,854 2018.09.28 US

(71)申请人 英特尔公司
地址 美国加利福尼亚州

(72)发明人 布雷特·图尔
亚力山大·F·海涅克
克里斯托弗·J·休斯
罗农·佐哈尔 迈克尔·埃斯皮格
丹·鲍姆 拉阿南·萨德
罗伯特·瓦伦泰恩
马克·J·查尼
埃尔莫斯塔法·乌尔德-艾哈迈德-瓦尔

(74)专利代理机构 北京东方亿思知识产权代理有限责任公司 11258
代理人 姜飞

(51)Int.Cl.
G06F 9/30(2006.01)
G06F 9/34(2006.01)

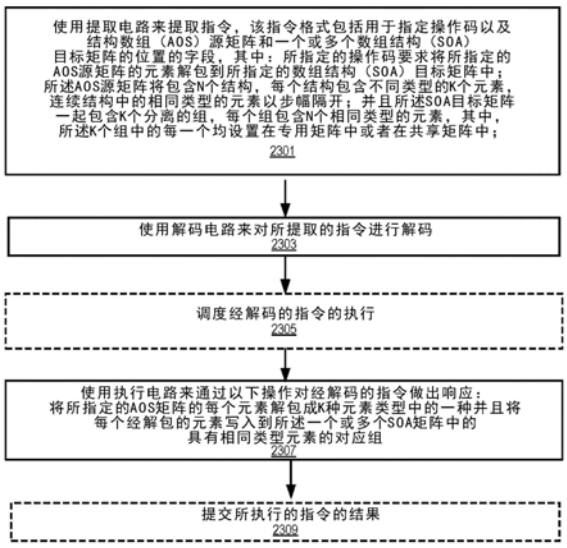
权利要求书3页 说明书40页 附图42页

(54)发明名称

用于执行用于快速元素解包到二维寄存器中的指令的系统

(57)摘要

公开的实施例涉及用于执行用于快速元素解包到二维寄存器中的指令的系统。在一个示例中，一种处理器包括：提取电路，用于提取指令，该指令的格式包括用于指定操作码的字段以及指定结构数组(AOS)源矩阵和一个或多个数组结构(SOA)目标矩阵的位置的字段，其中：所指定的操作码要求将所指定的AOS源矩阵的元素解包到所指定的数组结构(SOA)目标矩阵中，AOS源矩阵用于包含N个结构，每个结构包含不同类型的K个元素，连续结构中的相同类型的元素以步幅隔开，SOA目标矩阵一起包含K个分离的组，每个组包含N个相同类型的元素；解码电路，用于对所提取的指令进行解码；以及执行电路，响应于经解码的指令，所述执行电路用于将所指定的AOS矩阵的每个元素解包成一个或多个SOA矩阵的K种元素类型中的一种。



CN 110968346 A

1.一种处理器,所述处理器包括:

提取电路,所述提取电路用于提取指令,所述指令的格式包括用于指定操作码的字段以及指定结构数组(AOS)源矩阵和一个或多个数组结构(SOA)目标矩阵的位置的字段,其中:

所述操作码要求将所述AOS源矩阵的元素解包到所述数组结构(SOA)目标矩阵中;

所述AOS源矩阵用于包含N个结构,每个结构包含不同类型的K个元素,连续结构中的相同类型的元素以步幅隔开;

所述SOA目标矩阵一起包含K个分离的组,每个组包含N个相同类型的元素,其中,所述K个组中的每个组均设置在专用矩阵中或者在共享矩阵中;

解码电路,所述解码电路用于对所提取的指令进行解码;以及

执行电路,响应于经解码的指令,所述执行电路用于将所述AOS矩阵的每个元素解包成K种元素类型中的一种,并且将每个经解包的元素写入到所述一个或多个SOA矩阵中的具有相同类型元素的所述经解包的元素的对应组中。

2.根据权利要求1所述的处理器,其中,所述AOS矩阵和所述SOA矩阵中的每个矩阵被存储在以下任一个中:向量寄存器的合集、块片寄存器的合集、和存储器位置。

3.根据权利要求1-2中的任一项所述的处理器,其中,所述一个或多个SOA矩阵包括K个矩阵,每个矩阵用于排他地存储所述K个相同类型的组中的一种类型。

4.根据权利要求1-2中的任一项所述的处理器,其中,所述执行电路进一步用于对从所述AOS源矩阵移动到所述SOA目标矩阵的数据的数据格式进行变换。

5.根据权利要求1-2中的任一项所述的处理器,其中,所述指令进一步包括用于指定所述AOS矩阵和所述SOA矩阵的元素的元素大小的字段,所述元素大小包括微字节、半字节、字节、字、双字和四字中的一种。

6.根据权利要求1-2中的任一项所述的处理器,其中,所述指令进一步包括用于指定所述步幅的字段以及用于指定元素组被隔开的辅助步幅的字段。

7.根据权利要求1-2中的任一项所述的处理器,其中,所述一个或多个SOA矩阵包括单个矩阵,所有K组元素被设置在所述单个矩阵中,并且相同类型的元素被分组在一起并且与其他元素类型的组分隔开。

8.一种包括处理器和存储器的系统,所述处理器包括:

提取电路,所述提取电路用于提取指令,所述指令的格式包括用于指定操作码的字段以及指定结构数组(AOS)源矩阵和一个或多个数组结构(SOA)目标矩阵的位置的字段,其中

所述操作码要求将所述AOS源矩阵的元素解包到所述数组结构(SOA)目标矩阵中;

所述AOS源矩阵用于包含N个结构,每个结构包含不同类型的K个元素,连续结构中的相同类型的元素以步幅隔开;

所述SOA目标矩阵一起包含K个分离的组,每个组包含N个相同类型的元素,其中,所述K个组中的每个组均设置在专用矩阵中或者在共享矩阵中;

解码电路,所述解码电路用于对所提取的指令进行解码;以及

执行电路,响应于经解码的指令,所述执行电路用于将所述AOS矩阵的每个元素解包成K种元素类型中的一种,并且将每个经解包的元素写入到所述一个或多个SOA矩阵中的具有相同类型元素的所述经解包的元素的对应组中。

9.根据权利要求8所述的系统,其中所述AOS矩阵和所述SOA矩阵中的每个矩阵被存储在以下任一个中:向量寄存器的合集、块片寄存器的合集、和存储器位置。

10.根据权利要求8-9中的任一项所述的系统,其中,所述一个或多个SOA矩阵包括K个矩阵,每个矩阵用于排他地存储所述K个相同类型的组中的一种类型。

11.根据权利要求8-9中的任一项所述的系统,其中,所述执行电路进一步用于对从所述AOS源矩阵移动到所述SOA目标矩阵的数据的数据格式进行变换。

12.根据权利要求8-9中的任一项所述的系统,其中,所述指令进一步包括用于指定所述AOS矩阵和所述SOA矩阵的元素的元素大小的字段,所述元素大小包括微字节、半字节、字节、字、双字和四字中的一种。

13.根据权利要求8-9中的任一项所述的系统,其中,所述指令进一步包括用于指定所述步幅的字段以及用于指定元素组被隔开的辅助步幅的字段。

14.根据权利要求8-9中的任一项所述的系统,其中,所述一个或多个SOA矩阵包括单个矩阵,所有K组元素被设置在所述单个矩阵中,并且相同类型的元素被分组在一起并且与其他元素类型的组分隔开。

15.一种要由处理器执行的方法,所述方法包括:

使用提取电路来提取指令,所述指令的格式包括用于指定操作码的字段以及指定结构数组(AOS)源矩阵和一个或多个数组结构(SOA)目标矩阵的位置的字段,其中:

所述操作码要求将所述AOS源矩阵的元素解包到所述数组结构(SOA)目标矩阵中;

所述AOS源矩阵用于包含N个结构,每个结构包含不同类型的K个元素,连续结构中的相同类型的元素以步幅隔开;

所述SOA目标矩阵一起包含K个分离的组,每个组包含N个相同类型的元素,其中,所述K个组中的每个组均设置在专用矩阵中或者在共享矩阵中;

使用解码电路来对所提取的指令进行解码;以及

使用执行电路来通过以下操作对经解码的指令做出响应:将所述AOS矩阵的每个元素解包成K种元素类型中的一种,并且将每个经解包的元素写入到所述一个或多个SOA矩阵中的具有相同类型元素的所述经解包的元素的对应组中。

16.根据权利要求15所述的方法,其中,所述AOS矩阵和所述SOA矩阵中的每个矩阵被存储在以下任一个中:向量寄存器的合集、块片寄存器的合集、和存储器位置。

17.根据权利要求15-16中的任一项所述的方法,其中,所述一个或多个SOA矩阵包括K个矩阵,每个矩阵用于排他地存储所述K个相同类型的组中的一种类型。

18.根据权利要求15-16中的任一项所述的方法,其中,所述执行电路进一步用于对从所述AOS源矩阵移动到所述SOA目标矩阵的数据的数据格式进行变换。

19.根据权利要求15-16中的任一项所述的方法,其中,所述指令进一步包括用于指定所述AOS矩阵和所述SOA矩阵的元素的元素大小的字段,所述元素大小包括微字节、半字节、字节、字、双字和四字中的一种。

20.根据权利要求15-16中的任一项所述的方法,其中,所述指令进一步包括用于指定所述步幅的字段以及用于指定元素组被隔开的辅助步幅的字段。

21.根据权利要求15-16中的任一项所述的方法,其中,所述一个或多个SOA矩阵包括单个矩阵,所有K组元素被设置在所述单个矩阵中,并且相同类型的元素被分组在一起并且与

其他元素类型的组分隔开。

22.一种包含代码的机器可读介质,所述代码在被执行时使机器执行根据权利要求15-21中的任一项所述的方法。

用于执行用于快速元素解包到二维寄存器中的指令的系统

技术领域

[0001] 发明领域一般地涉及计算机处理器架构,并且更具体地涉及用于执行用于快速元素解包到二维寄存器中的指令的系统和方法。

背景技术

[0002] 不同类型的数据常常用基于一种关系的逻辑分组加以表示,但是然后有时需要基于不同的关系被“解包”。例如,像素通常被表示为以下3种类型的数字的集合:R(红色)、G(绿色)和B(蓝色)。在某些场合需要跨越多个像素对元素类型中的仅一种(例如R)执行运算。

发明内容

[0003] 根据本申请的一方面,提供了一种处理器,所述处理器包括:提取电路,所述提取电路用于提取指令,所述指令的格式包括用于指定操作码的字段以及指定结构数组(AOS)源矩阵和一个或多个数组结构(SOA)目标矩阵的位置的字段,其中:所述操作码要求将所述AOS源矩阵的元素解包到所述数组结构(SOA)目标矩阵中,所述AOS源矩阵用于包含N个结构,每个结构包含不同类型的K个元素,连续结构中的相同类型的元素以步幅隔开,所述SOA目标矩阵一起包含K个分离的组,每个组包含N个相同类型的元素,其中,所述K个组中的每个组均设置在专用矩阵中或者在共享矩阵中;解码电路,所述解码电路用于对所提取的指令进行解码;以及执行电路,响应于经解码的指令,所述执行电路用于将所述AOS矩阵的每个元素解包成K种元素类型中的一种,并且将每个经解包的元素写入到所述一个或多个SOA矩阵中的具有相同类型元素的所述经解包的元素的对应组中。

[0004] 根据本申请的另一方面,提供了一种包括处理器和存储器的系统,所述处理器包括:提取电路,所述提取电路用于提取指令,所述指令的格式包括用于指定操作码的字段以及指定结构数组(AOS)源矩阵和一个或多个数组结构(SOA)目标矩阵的位置的字段,其中所述操作码要求将所述AOS源矩阵的元素解包到所述数组结构(SOA)目标矩阵中,所述AOS源矩阵用于包含N个结构,每个结构包含不同类型的K个元素,连续结构中的相同类型的元素以步幅隔开,所述SOA目标矩阵一起包含K个分离的组,每个组包含N个相同类型的元素,其中,所述K个组中的每个组均设置在专用矩阵中或者在共享矩阵中;解码电路,所述解码电路用于对所提取的指令进行解码;以及执行电路,响应于经解码的指令,所述执行电路用于将所述AOS矩阵的每个元素解包成K种元素类型中的一种,并且将每个经解包的元素写入到所述一个或多个SOA矩阵中的具有相同类型元素的所述经解包的元素的对应组中。

[0005] 根据本申请的又一方面,提供了一种要由处理器执行的方法,所述方法包括:使用提取电路来提取指令,所述指令的格式包括用于指定操作码的字段以及指定结构数组(AOS)源矩阵和一个或多个数组结构(SOA)目标矩阵的位置的字段,其中,所述操作码要求将所述AOS源矩阵的元素解包到所述数组结构(SOA)目标矩阵中,所述AOS源矩阵用于包含N个结构,每个结构包含不同类型的K个元素,连续结构中的相同类型的元素以步幅隔开,所

述SOA目标矩阵一起包含K个分离的组，每个组包含N个相同类型的元素，其中，所述K个组中的每个组均设置在专用矩阵中或者在共享矩阵中；使用解码电路来对所提取的指令进行解码；以及使用执行电路来通过以下操作对经解码的指令做出响应：将所述AOS矩阵的每个元素解包成K种元素类型中的一种，并且将每个经解包的元素写入到所述一个或多个SOA矩阵中的具有相同类型元素的所述经解包的元素的对应组中。

附图说明

[0006] 通过示例而非限制的方式在附图的各图中图示本发明，在附图中相似的标记指示类似的元素，并且在附图中：

[0007] 图1A图示配置的块片(tile)的实施例；

[0008] 图1B图示配置的块片的实施例；

[0009] 图2图示矩阵存储的若干示例；

[0010] 图3图示利用矩阵(块片)运算加速器的系统的实施例；

[0011] 图4和图5示出使用矩阵运算加速器如何共享存储器的不同实施例；

[0012] 图6图示使用块片(“TMMA”)的矩阵乘法累加运算的实施例；

[0013] 图7图示链式融合乘法累加指令的迭代的执行的子集的实施例；

[0014] 图8图示链式融合乘法累加指令的迭代的执行的子集的实施例；

[0015] 图9图示链式融合乘法累加指令的迭代的执行的子集的实施例；

[0016] 图10图示链式融合乘法累加指令的迭代的执行的子集的实施例；

[0017] 图11图示根据实施例的二次幂大小的SIMD实施方式，其中累加器使用大于乘法器的输入的输入大小；

[0018] 图12图示利用矩阵运算电路的系统的实施例；

[0019] 图13图示使用块片来支持矩阵运算的处理器核流水线的实施例；

[0020] 图14图示使用块片来支持矩阵运算的处理器核流水线的实施例；

[0021] 图15图示以行主要格式和列主要格式表达的矩阵的示例；

[0022] 图16图示矩阵(块片)的使用的示例；

[0023] 图17图示矩阵(块片)的使用的方法的实施例；

[0024] 图18图示根据实施例的对块片的使用的配置的支持；

[0025] 图19图示关于要支持的矩阵(块片)的描述的实施例；

[0026] 图20(A)至图20(D)图示寄存器的示例；

[0027] 图21A至图21C图示根据一些实施例的TileUnpackAOS指令的示例性执行；

[0028] 图22A至图22C图示根据一些实施例的TilePackSOA指令的示例性执行；

[0029] 图23A是图示处理器执行TileUnpackAOS指令的实施例的流程图；

[0030] 图23B是图示处理器执行TilePackSOA指令的实施例的流程图；

[0031] 图24是图示根据一些实施例的TileUnpackAOS/TilePackSOA指令的格式的框图；

[0032] 图25A和图25B是图示根据实施例的通用向量友好指令格式及指令模板的框图；

[0033] 图25A是图示根据实施例的通用向量友好指令格式及A类指令模板的框图；

[0034] 图25B是图示根据实施例的通用向量友好指令格式及B类指令模板的框图；

[0035] 图26A是图示根据实施例的示例性特定向量友好指令格式的框图；

[0036] 图26B是图示根据本发明的一个实施例的特定向量友好指令格式的组成完整操作码字段的字段的框图；

[0037] 图26C是图示根据本发明的一个实施例的特定向量友好指令格式的组成寄存器索引字段的字段的框图；

[0038] 图26D是图示根据本发明的一个实施例的特定向量友好指令格式的组成增强运算字段的字段的框图；

[0039] 图27是根据本发明的一个实施例的寄存器架构的框图；

[0040] 图28A是图示根据实施例的示例性顺序(in-order)流水线和示例性寄存器重命名、乱序(out-of-order)发布/执行流水线两者的框图；

[0041] 图28B是图示根据实施例的要包括在处理器中的顺序架构核和示例性寄存器重命名、乱序发布/执行架构核两者的示例性实施例的框图；

[0042] 图29A-B示出了更具体的示例性顺序核架构的框图，其核将是芯片中的几个逻辑块之一(包括相同类型和/或不同类型的其他核)；

[0043] 图29A是根据实施例的单个处理器核以及其与管芯上互连网络的连接及其二级(L2)缓存的本地子集的框图；

[0044] 图29B是根据实施例的图29A中的处理器核的一部分的展开图；

[0045] 图30是根据实施例的可以具有多于一个核，可以具有集成存储器控制器并且可以具有集成图形的处理器的框图；

[0046] 图31-34是示例性计算机架构的框图；

[0047] 图31示出了根据本发明一个实施例的系统的框图；

[0048] 图32是根据本发明实施例的第一更具体的示例性系统的框图；

[0049] 图33是根据本发明实施例的第二更具体的示例性系统的框图；

[0050] 图34是根据本发明实施例的片上系统(SoC)的框图；及

[0051] 图35是根据实施例的对照使用软件指令转换器将源指令集中的二进制指令转换为目标指令集中的二进制指令的框图。

具体实施例

[0052] 在以下描述中，阐述了许多具体细节。然而，应理解的是，可以在没有这些具体细节的情况下实践实施例。在其他情况下，未详细地示出众所周知的电路、结构和技术，以免模糊对本说明书的理解。

[0053] 在本说明书中对“一个实施例”、“实施例”、“示例实施例”等的引用指示所描述的实施例可以包括特定特征、结构或特性，但是每个实施例可能不一定包括该特定特征、结构或特性。此外，这样的短语不一定指代同一实施例。进一步地，当结合实施例描述特定特征、结构或特性时，认为结合(明确描述或未明确描述的)其他实施例影响这样的特征、结构或特性在本领域的技术人员的知识范围内。

[0054] 在许多主流处理器中，处理矩阵是一项困难和/或指令密集的任务。例如，矩阵的行可被放置到多个压缩数据(例如，SIMD或向量)寄存器中，然后单独地在上面操作。例如，取决于数据大小，加法两个8x2矩阵可能需要加载或聚集到四个压缩数据寄存器中。然后执行与来自每个矩阵的第一行相对应的压缩数据寄存器的第一加法，并且执行与来自每个矩

阵的第二行相对应的压缩数据寄存器的第二加法。然后所得到的压缩数据寄存器被分散回到存储器。虽然对于小矩阵此场景可能是可接受的，但是它对较大矩阵常常是不可接受的。

[0055] 讨论

[0056] 本文描述的是用于在诸如中央处理单元(CPU)、图形处理单元(GPU)和加速器这样的计算机硬件中支持矩阵运算的机制。矩阵运算利用表示诸如寄存器这样的存储器的一个或多个压缩区域的二维(2-D)数据结构。贯穿本说明书，这些2-D数据结构被称为块片。注意的是，矩阵可以比块片小(使用小于一个块片的整体)，或者利用多个块片(矩阵大于任何一个块片的大小)。贯穿说明书，矩阵(块片)语言用于指示使用影响矩阵的块片所执行的运算，该矩阵是否大于任何一个块片通常不是相关的。

[0057] 可以通过不同的运算对每个块片进行操作，所述不同的运算诸如本文详述的那些运算，并且包括但不限于：矩阵(块片)乘法、块片加法、块片减法、块片对角线、块片归零、块片转置、块片点积、块片广播、块片行广播、块片列广播、块片乘法、块片乘法和累加、块片移动等。附加地，对于诸如比例和/或偏差的使用的运算符的支持可以与这些运算一起使用或者支持将来的非数值应用，例如，OpenCL“本地存储器”、数据压缩/解压缩等。本文还描述了用于执行TileUnpackAOS和TilePackSOA指令的指令。

[0058] 存储的各部分(诸如存储器(非易失性和易失性)、寄存器、缓存等)被布置成不同水平和垂直维度的块片。例如，块片可以具有4的水平维度(例如，矩阵的4行)和8的垂直维度(例如，矩阵的8列)。通常，水平维度是与元素大小(例如，2、4、8、16、32、64、128位等)相关的。可以支持多种数据类型(单精度浮点、双精度浮点、整数等)。

[0059] 配置的块片的示例性使用

[0060] 在一些实施例中，可配置块片参数。例如，给定块片可以被配置为提供块片选项。示例性块片选项包括但不限于：块片的行数、块片的列数、块片是否有效、以及块片是否由相等大小的块片的对构成。

[0061] 图1A图示配置的块片的实施例。如所示，应用存储器102的4kB上存储有4个1kB块片：块片t0 104、块片t1 106、块片t2 108和块片t3 110。在此示例中，4个块片不由对构成，并且各自具有按行和列布置的元素。块片t0 104和块片t1 106具有K行和N列的4字节元素(例如，单精度数据)，其中K等于8且N＝32。块片t2 108和块片t3 110具有K行和N/2列的8字节元素(例如，双精度数据)。因为双精度操作数是单精度的宽度的两倍，所以此配置与用于提供块片选项的调色板一致，给至少4个名称提供至少4kB的总存储。在操作中，可使用加载和存储操作来从存储器加载块片并将块片存储到存储器。取决于所使用的指令编码方案，可用应用存储器的量以及可用块片的大小、数量和配置变化。

[0062] 图1B图示配置的块片的实施例。如所示，应用存储器122的4kB上存储有2对1kB块片，第一对是块片t4L 124和块片t4R 126，并且第二对是块片t5L 128和块片t5R 130。如所示，块片对被划分成左块片和右块片。在其他实施例中，块片的对被划分成偶块片和奇块片。在此示例中，4个块片各自具有按行和列布置的元素。块片t4L 124和块片t4R126具有K行和N列的4字节元素(例如，单精度浮点数据)，其中K等于8且N等于32。块片t5L 128和块片t5R 130具有K行和N/2列的8字节元素(例如，双精度浮点数据)。因为双精度操作数是单精度的宽度的两倍，所以此配置与用于提供块片选项的调色板一致，给至少2个名称提供至少4kB的总存储。图1A的四个块片使用4个名称，每个名称为1kB块片命名，然而图1B中的2对块

片可使用2个名称来指定配对的块片。在一些实施例中，块片指令接受配对的块片的名称作为操作数。在操作中，可使用加载和存储操作来从存储器加载块片并将块片存储到存储器。取决于所使用的指令编码方案，可用应用存储器的量以及可用块片的大小、数量和配置变化。

[0063] 在一些实施例中，块片参数是可定义的。例如，“调色板”用于提供块片选项。示例性选项包括但不限于：块片名称数、存储的一行中的字节数、块片中的行数和列数等。例如，可以将块片的最大“高度”(行数)定义为：

[0064] 块片最大行＝架构存储/(调色板名称数*每行的字节数)

[0065] 因此，可编写应用，使得名称的固定使用将能够跨越实施方式利用不同的存储大小。

[0066] 块片的配置使用块片配置(“TILECONFIG”)指令来完成，其中在所选调色板中定义特定块片使用。此声明包括要使用的块片名称数、所请求的每名称(块片)的行数和列数，并且在一些实施例中，包括所请求的每个块片的数据类型。在一些实施例中，在执行TILECONFIG指令期间执行一致性检查以确定它与调色板条目的限制匹配。

[0067] 示例性块片存储类型

[0068] 图2图示矩阵存储的若干示例。在(A)中，块片被存储在存储器中。如所示，每“行”由四个压缩数据元素构成。为了到达下一“行”，使用步幅(stride)值。注意的是，行可以被连续地存储在存储器中。当块片存储未映射底层存储器阵列行宽度时，跨越式(strided)存储器允许一行接一行地访问。

[0069] 块片从存储器加载和存储到存储器通常是从应用存储器到数据的压缩行的跨越式访问。在一些实施例中，在加载运算指令中作为TILE操作数对应用存储器的示例性TILELOAD和TILESTORE指令或其他指令引用是可重新起动的，以每指令处理(多达)2*行的页面错误、未屏蔽浮点异常和/或中断。

[0070] 在(B)中，矩阵被存储在由诸如压缩数据寄存器(单指令、多数据(SIMD)或向量寄存器)这样的多个寄存器组成的块片中。在此示例中，块片被重叠在三个物理寄存器上。通常，使用连续的寄存器，然而，情况不一定是这样。

[0071] 在(C)中，矩阵被存储在对于块片运算中使用的融合多累加(FMA)电路可访问的非寄存器存储中的块片中。此存储可以在FMA内部或者与它相邻。附加地，在下面讨论的一些实施例中，该存储可以用于数据元素而不是整个行或块片。

[0072] 经由CPUID报告用于TMMA架构的支持参数。在一些实施例中，信息的列表包括最大高度和最大SIMD维度。配置TMMA架构需要指定每个块片的维度、每个块片的元素大小和调色板标识符。此配置通过执行TILECONFIG指令来完成。

[0073] TILECONFIG指令的成功执行启用后续TILE运算符。TILERELEASEALL指令清除块片配置并且禁用TILE运算(直到执行下一个TILECONFIG指令)。在一些实施例中，在使用块片的上下文切换中使用XSAVE、XSTORE等。在一些实施例中，在XSAVE中使用2个XCR0位，一个位用于TILECONFIG元数据并且一个位对应于实际块片净荷数据。

[0074] TILECONFIG不仅配置块片使用，而且还设置指示程序在配置了块片的代码的区域中的状态变量。实施方式可以枚举对可与块片区域一起使用的其他指令的限制，诸如不使用现有寄存器组等。

[0075] 退出块片区域通常用TILERELEASEALL指令来完成。它不取任何参数并且很快地使所有块片无效(指示数据不再需要任何保存或恢复)并且清除与位于块片区域中相对应的内部状态。

[0076] 在一些实施例中,块片运算将使超过通过块片配置所指定的维度的任何行和任何列归零。例如,当每行被写入时,块片运算将使超过已配置数量的列的数据归零(将元素的大小考虑进来)。例如,在64字节行以及配置有10行和12列的块片情况下,写入FP32元素的运算将写入前10行中的每一行,其中12*4个字节具有输出/结果数据,并且使剩余每行中的4*4个字节归零。块片运算还完全使前10个配置的行之后的任何行归零。当使用具有64字节行的1K块片时,将有16行,所以在此示例中,最后6行也将被归零。

[0077] 在一些实施例中,上下文恢复指令(例如,XRSTOR)在加载数据时,强制对于块片超过所配置的行的数据将被维持为零。如果没有有效的配置,则所有行都被归零。块片数据的XRSTOR可能在超过所配置的那些列中加载垃圾。XRSTOR应该不可能清除超过所配置的列数,因为没有与块片配置相关联的元素宽度。

[0078] 上下文保存(例如,XSAVE)在将块片写入到存储器时暴露整个TILE存储区域。如果XRSTOR将垃圾数据加载到块片的最右边部分中,则该数据将通过XSAVE保存。XSAVE将针对超过为每个块片所指定的数量的行写入零。

[0079] 在一些实施例中,块片指令是可重新启动的。访问存储器的运算允许在页面故障之后重新启动。处理浮点运算的计算指令还允许未屏蔽浮点异常,其中异常的屏蔽由控制和/或状态寄存器控制。

[0080] 为了支持在这些事件之后重新启动指令,指令将信息存储在下面详述的启动寄存器中。

[0081] 矩阵(块片)运算系统

[0082] 示例性硬件支持

[0083] 图3图示利用矩阵(块片)运算加速器的系统的实施例。在此图示中,主机处理器/处理系统301将命令311(例如,诸如算术或矩阵操作运算这样的矩阵操纵运算、或加载和存储操作)传递到矩阵运算加速器307。然而,这是仅为了讨论目的而示出的。如稍后详述的,此加速器307可以是处理核的一部分。通常,作为块片操作运算符指令的命令311将块片称为寄存器-寄存器(“reg-reg”)或寄存器-存储器(“reg-mem”)格式。诸如TILESTORE、TILELOAD、TILECONFIG等这样的其他命令不对块片执行数据运算。命令可以是用于加速器307处理的解码指令(例如,微运算)或宏指令。

[0084] 在此示例中,相干存储器接口303耦合到主机处理器/处理系统301和矩阵运算加速器307,使得它们可共享存储器。图4和图5示出使用矩阵运算加速器如何共享存储器的不同实施例。如图4中所示,主机处理器401和矩阵运算加速器电路405共享同一存储器403。图5图示主机处理器501和矩阵运算加速器505不共享存储器但是可访问彼此的存储器的实施例。例如,处理器501可正常访问块片存储器507并利用其主机存储器503。类似地,矩阵运算加速器505可访问主机存储器503,但是更典型地使用它自己的存储器507。注意这些存储器可以是不同类型的。

[0085] 在一些实施例中,使用物理寄存器上的覆盖来支持块片。例如,取决于实施例,块片可以利用16个1,024位寄存器、32个512位寄存器等。在一些实施例中,矩阵运算利用表示

诸如寄存器这样的存储器的一个或多个压缩区域的2维(2-D)数据结构。在整个说明书中，这些2-D数据结构被称为块片或块片寄存器。

[0086] 在一些实施例中，矩阵运算加速器307包括耦合到数据缓冲器305的多个FMA 309(在一些实施例中，这些缓冲器305中的一个或多个被存储在如所示的网格的FMA中)。数据缓冲器305缓冲从存储器加载的块片和/或要存储到存储器的块片(例如，使用块片加载或块片存储指令)。数据缓冲器可以是例如多个寄存器。通常，这些FMA被布置为能够读取和写入块片的链式FMA 309的网格。在此示例中，矩阵运算加速器307将使用块片T0、T1和T2来执行矩阵乘法运算。块片中的至少一个被收容在FMA网格309中。在一些实施例中，运算中的所有块片都被存储在FMA网格309中。在其他实施例中，仅子集被存储在FMA网格309中。如所示，T1被收容而T0和T2未被收容。注意的是，A、B和C指代可能占用或者可能未占用块片的整个空间的这些块片的矩阵。

[0087] 图6图示使用块片(“TMMA”)的矩阵乘法累加运算的实施例。

[0088] 矩阵(块片A601)中的行数与包括计算的等待时间的串行(链式)FMA数匹配。实施方式自由在较小高度的网格上再循环，但是计算保持不变。

[0089] 源/目标向量来自N行的块片(块片C 605)并且FMA的网格611执行N个向量矩阵运算，从而产生执行块片的矩阵乘法的完整指令。块片B 603是另一向量源并且向每级中的FMA供应“广播”项。

[0090] 在运算中，在一些实施例中，矩阵B(存储在块片B 603中)的元素遍布FMA的矩形网格上。矩阵B(存储在块片A 601中)使其行的元素被转置以与FMA的矩形网格的列维度相匹配。在网格中的每个FMA处，A和B的元素被相乘并加到传入加数(从图中上方)，并且传出和被传递到FMA的下一行(或最终输出)。

[0091] 单个步骤的等待时间与K(矩阵B的行高度)成比例并且从属TMMA通常具有足够的源-目标行(在单个块片中或跨越块片)以隐藏该等待时间。实施方式还可以跨越时间步进分割SIMD(压缩数据元素)维度M(矩阵A的行高度)，但是这简单地改变乘以K的常数。当程序指定比通过TMACC枚举的最大值小的K时，实施方式自由地用“屏蔽”或“早期输出”实现这个。

[0092] 整个TMMA的等待时间与N*K成比例。重复率与N成比例。每TMMA指令的MAC数是N*K*M。

[0093] 图7图示链式融合乘法累加指令的迭代的执行的子集的实施例。特别地，这图示了目的地的一个压缩数据元素位置的迭代的执行电路。在此实施例中，链式融合乘法累加在有符号源上运算，其中累加器是输入数据大小的2倍。

[0094] 第一有符号源(源1 701)和第二有符号源(源2 703)各自具有四个压缩数据元素。这些压缩数据元素中的每一个均存储诸如浮点数据这样的有符号数据。第三有符号源(源3 709)具有两个压缩数据元素，其中的每一个均存储有符号数据。第一有符号源701和第二有符号源703的大小是第三有符号源(初始值或先前结果)709的大小的一半。例如，第一有符号源701和第二有符号源703可以具有32位压缩数据元素(例如，单精度浮点)，而第三有符号源709可以具有64位压缩数据元素(例如，双精度浮点)。

[0095] 在此图示中，示出了仅第一有符号源701和第二有符号源703的两个最高有效压缩数据元素位置以及第三有符号源709的最高有效压缩数据元素位置。当然，还将处理其他压

缩数据元素位置。

[0096] 如所图示的，压缩数据元素被成对处理。例如，使用乘法器电路705来将第一有符号源701和第二有符号源703的最高有效数据元素位置的数据相乘，并且使用乘法器电路707来将来自第一有符号源701和第二有符号源703的第二最高有效数据元素位置的数据相乘。在一些实施例中，这些乘法器电路705和707被重用于其他压缩数据元素位置。在其他实施例中，使用附加乘法器电路，使得压缩数据元素被并行处理。在一些上下文中，并行执行使用为有符号第三源709的大小的通道来完成。使用加法电路711来将每个乘法的结果相加。

[0097] 乘法的结果的相加的结果被加到来自有符号源3 709的最高有效压缩数据元素位置的数据(使用不同的加法器713或相同的加法器711)。

[0098] 最后，第二加法的结果被存储到压缩数据元素位置中的有符号目的地715中，或者在存在下一次迭代的情况下被传递到下一次迭代，所述压缩数据元素位置对应于从有符号第三源709使用的压缩数据元素位置。在一些实施例中，写入掩码被应用于此存储，使得如果设置了对应的写入掩码(位)，则发生存储，而如果未设置对应的写入掩码(位)，则不会发生存储。

[0099] 图8图示链式融合乘法累加指令的迭代的执行的子集的实施例。特别地，这图示目的地的一个压缩数据元素位置的迭代的执行电路。在此实施例中，链式融合乘法累加在有符号源上运算，其中累加器是输入数据大小的2倍。

[0100] 第一有符号源(源1 801)和第二有符号源(源2 803)各自具有四个压缩数据元素。这些压缩数据元素中的每一个均存储诸如整数数据这样的有符号数据。第三有符号源(源3 809)具有两个压缩数据元素，其中的每一个均存储有符号数据。第一有符号源801和第二有符号源803的大小是第三有符号源809的大小的一半。例如，第一有符号源801和第二有符号源803可以具有32位压缩数据元素(例如，单精度浮点)，第三有符号源809可以具有64位压缩数据元素(例如，双精度浮点)。

[0101] 在此图示中，示出了仅第一有符号源801和第二有符号源803的两个最高有效压缩数据元素位置以及第三有符号源809的最高有效压缩数据元素位置。当然，还将处理其他压缩数据元素位置。

[0102] 如所图示的，压缩数据元素被成对处理。例如，使用乘法器电路805来将第一有符号源801和第二有符号源803的最高有效数据元素位置的数据相乘，并且使用乘法器电路807来将来自第一有符号源801和第二有符号源803的第二最高有效数据元素位置的数据相乘。在一些实施例中，这些乘法器电路805和807被重用于其他压缩数据元素位置。在其他实施例中，使用附加乘法器电路，使得压缩数据元素被并行处理。在一些上下文中，并行执行使用为有符号第三源(初始值或先前迭代结果)809的大小的通道来完成。使用加法/饱和电路813来将每个乘法的结果加到有符号第三源809。

[0103] 当加法产生太大的值时，加法/饱和(累加器)电路813保存操作数的符号。特别地，在多路加与写入到目的地或下一次迭代之间的无限精度结果上发生饱和评估。当累加器813是浮点并且输入项是整数时，乘积和与浮点累加器输入值变成无限精度值(数百位的定点数)，乘法结果和第三输入的相加被执行，并且到实际累加器类型的单个舍入被执行。

[0104] 无符号饱和意味着输出值被限制为该元素宽度的最大无符号数(全1)。有符号饱

和意味着值被限制为在该元素宽度的最小负数和最大正数之间的范围内(例如对于字节，范围是从-128(=-2^7)到127(=2^7-1))。

[0105] 加法和饱和检查的结果被存储到压缩数据元素位置中的有符号结果815中，所述压缩数据元素位置对应于从有符号第三源809使用的压缩数据元素位置，或者在存在下一次迭代的情况下被传递到下一次迭代。在一些实施例中，写入掩码被应用于此存储，使得如果设置了对应的写入掩码(位)，则发生存储，而如果未设置对应的写入掩码(位)，则不会发生存储。

[0106] 图9图示链式融合乘法累加指令的迭代的执行的子集的实施例。特别地，这图示目的地的一个压缩数据元素位置的迭代的执行电路。在此实施例中，链式融合乘法累加在有符号源和无符号源上运算，其中累加器是输入数据大小的4倍。

[0107] 第一有符号源(源1 901)和第二无符号源(源2 903)各自具有四个压缩数据元素。这些压缩数据元素中的每一个均具有诸如浮点数或整数数据这样的数据。第三有符号源(初始值或结果915)具有存储有符号数据的压缩数据元素。第一源901和第二源903的大小是第三有符号源915的四分之一。例如，第一源901和第二源903可以具有16位压缩数据元素(例如，字)并且第三有符号源915可以具有64位压缩数据元素(例如，双精度浮点或64位整数)。

[0108] 在此图示中，示出了第一源901和第二源903的四个最高有效压缩数据元素位置以及第三有符号源915的最高有效压缩数据元素位置。当然，如果存在任何其他压缩数据元素位置，则还将处理其他压缩数据元素位置。

[0109] 如所图示的，压缩数据元素以四元组的形式被处理。例如，使用乘法器电路905来将第一源901和第二源903的最高有效压缩数据元素位置的数据相乘，使用乘法器电路905来将来自第一源901和第二源903的第二最高有效压缩数据元素位置的数据相乘，使用乘法器电路909来将来自第一源901和第二源903的第三最高有效压缩数据元素位置的数据相乘，并且使用乘法器电路911来将来自第一源901和第二源903的最低有效压缩数据元素位置的数据相乘。在一些实施例中，在乘法之前，第一源901的有符号压缩数据元素被符号扩展并且第二源903的无符号压缩数据元素被零扩展。

[0110] 在一些实施例中，这些乘法器电路905-911被重用于其他压缩数据元素位置。在其他实施例中，使用附加乘法器电路，使得压缩数据元素被并行处理。在一些上下文中，并行执行使用为有符号第三源915的大小的通道来完成。使用加法电路913来将每个乘法的结果相加。

[0111] 乘法结果相加的结果被加到来自有符号源3 915的最高有效压缩数据元素位置的数据(使用不同的加法器917或相同的加法器913)。

[0112] 最后，第二加法的结果919被存储到压缩数据元素位置中的有符号目的地中，所述压缩数据元素位置对应于从有符号第三源915使用的压缩数据元素位置，或者被传递到下一次迭代。在一些实施例中，写入掩码被应用于此存储，使得如果设置了对应的写入掩码(位)，则发生存储，而如果未设置对应的写入掩码(位)，则不会发生存储。

[0113] 图10图示链式融合乘法累加指令的迭代的执行的子集的实施例。特别地，这图示目的地的一个压缩数据元素位置的迭代的执行电路。在此实施例中，链式融合乘法累加在有符号源和无符号源上运算，其中累加器是输入数据大小的4倍。

[0114] 第一有符号源1001和第二无符号源1003各自具有四个压缩数据元素。这些压缩数据元素中的每一个均存储诸如浮点数或整数数据这样的数据。第三有符号源1015具有存储有符号数据的压缩数据元素。第一源和第二源的大小是第三有符号源1015(初始或先前结果)的四分之一。例如,第一源和第二源可以具有16位压缩数据元素(例如,字)并且第三有符号源1015(初始或先前结果)可以具有64位压缩数据元素(例如,双精度浮点或64位整数)。

[0115] 在此图示中,示出了第一有符号源1001和第二无符号源1003的四个最高有效压缩数据元素位置以及第三有符号源1015的最高有效压缩数据元素位置。当然,如果存在任何其他压缩数据元素位置,则还将处理其他压缩数据元素位置。

[0116] 如所图示的,压缩数据元素以四元组的形式被处理。例如,使用乘法器电路1005来将第一有符号源1001和第二无符号源1003的最高有效压缩数据元素位置的数据相乘,使用乘法器电路1007来将来自第一有符号源1001和第二无符号源1003的第二最高有效压缩数据元素位置的数据相乘,使用乘法器电路1009来将来自第一有符号源1001和第二无符号源1003的第三最高有效压缩数据元素位置的数据相乘,并且使用乘法器电路1011来将来自第一有符号源1001和第二无符号源1003的最低有效压缩数据元素位置的数据相乘。在一些实施例中,在乘法之前,第一有符号源1001的有符号压缩数据元素被符号扩展,并且第二无符号源1003的无符号压缩数据元素被零扩展。

[0117] 在一些实施例中,这些乘法器电路1005-1011被重用于其他压缩数据元素位置。在其他实施例中,使用附加乘法器电路,以便压缩数据元素被并行处理。在一些上下文中,并行执行使用为有第三有符号源1015(初始或先前结果)的大小的通道来完成。乘法结果相加的结果使用加法器/饱和电路1013被加到来自第三有符号源1015(初始或先前结果)的最高有效压缩数据元素位置的数据。

[0118] 当加法产生对于有符号饱和来说太大或太小的值时,加法/饱和(累加器)电路1013保存操作数的符号。特别地,在多路加与写入到目的地之间的无限精度结果上发生饱和评估。当累加器1013是浮点并且输入项是整数时,乘积和与浮点累加器输入值变成无限精度值(数百位的定点数),乘法结果与第三输入的加法被执行,并且到实际累加器类型的单个舍入被执行。

[0119] 加法和饱和检查的结果1019被存储到压缩数据元素位置中的有符号目的地中,所述压缩数据元素位置对应于从第三有符号源1015(初始或先前结果)使用的压缩数据元素位置,或者被传递到下一次迭代。在一些实施例中,写入掩码被应用于此存储,使得如果设置了对应的写入掩码(位),则发生存储,而如果未设置对应的写入掩码(位),则不会发生存储。

[0120] 图11图示根据实施例的二次幂大小的SIMD实施例,其中累加器使用大于乘法器的输入的输入大小。注意源(对乘法器而言)和累加器值可以是有符号值或无符号值。对于具有2倍输入大小的累加器(换句话说,累加器输入值是源的压缩数据元素大小的两倍),表1101图示不同的配置。对于字节大小的源,累加器使用大小为16位的字或半精度浮点(HPFP)值。对于字大小的源,累加器使用大小为32位的32位整数或单精度浮点(SPFP)值。对于SPFP或32位整数大小的源,累加器使用大小为64位的64位整数或双精度浮点(DPFP)值。

[0121] 对于具有4倍输入大小的累加器(换句话说,累加器输入值是源的压缩数据元素大

小的四倍)，表1103图示不同的配置。对于字节大小的源，累加器使用大小为32位的32位整数或单精度浮点(SPFP)值。对于字大小的源，在一些实施例中累加器使用大小为64位的64位整数或双精度浮点(DPFP)值。

[0122] 对于具有8倍输入大小的累加器(换句话说，累加器输入值是源的压缩数据元素大小的八倍)，表1105图示配置。对于字节大小的源，累加器使用64位整数。

[0123] 如早先示意的，矩阵运算电路可以被包括在核中，或者作为外部加速器被包括。图12图示利用矩阵运算电路的系统的实施例。在此图示中，多个实体与环形互连1245耦合。

[0124] 多个核—核0 1201、核1 1203、核2 1205和核N 1207提供基于非块片的指令支持。在一些实施例中，在核1203中提供矩阵运算电路1251，而在其他实施例中，可在环形互连1245上访问矩阵运算电路1211和1213。

[0125] 附加地，提供一个或多个存储器控制器1223-1225以代表核和/或矩阵运算电路与存储器1233和1231进行通信。

[0126] 图13图示使用块片来支持矩阵运算的处理器核流水线的实施例。分支预测和解码电路1303根据存储在指令存储装置1301中的指令来执行指令的分支预测、指令的解码和/或两者。例如，本文详述的指令可以被存储在指令存储装置中。在一些实施方式中，单独的电路被用于分支预测，并且在一些实施例中，使用微码1305来将至少一些指令解码成一个或多个微运算、微码入口点、微指令、其他指令或其他控制信号。可以使用各种不同的机制来实现分支预测和解码电路1303。适合的机制的示例包括但不限于查找表、硬件实施方式、可编程逻辑阵列(PLA)、微码只读存储器(ROM)等。

[0127] 分支预测和解码电路1303耦合到分配/重命名电路1307，其在一些实施例中耦合到调度器电路1309。在一些实施例中，这些电路通过执行以下步骤中的一个或多个来提供寄存器重命名、寄存器分配和/或调度功能：1)将逻辑操作数值重命名为物理操作数值(例如，在一些实施例中为寄存器别名表)，2)将状态位和标志分配给经解码的指令，以及3)调度经解码的指令以供在指令池外的执行电路上执行(例如，在一些实施例中使用预留站)。

[0128] 调度器电路1309表示任何数量的不同调度器，包括预留站、中央指令窗口等。调度器电路1309耦合到或者包括物理寄存器堆1315。物理寄存器堆1315中的每一个均表示一个或多个物理寄存器堆，其中的不同物理寄存器堆存储一种或多种不同的数据类型，诸如标量整数、标量浮点、压缩整数、压缩浮点、向量整数、向量浮点、状态(例如，作为要执行的下一个指令的地址的指令指针)、块片等。在一个实施例中，物理寄存器堆1315包括向量寄存器电路、写掩码寄存器电路和标量寄存器电路。这些寄存器电路可以提供架构向量寄存器、向量屏蔽寄存器和通用寄存器。物理寄存器堆1315与引退电路1317交叠以图示可以实现寄存器重命名和乱序执行的各种方式(例如，使用重排序缓冲器和引退寄存器堆；使用未来寄存器堆、历史缓冲器和引退寄存器堆；使用寄存器映射和寄存器池；等)。引退电路1317和物理寄存器堆1315耦合到执行电路1311。

[0129] 虽然在乱序执行的上下文中描述寄存器重命名，但是应该理解的是，可以在顺序架构中使用寄存器重命名。虽然所图示的处理器的实施例还可以包括单独的指令和数据缓存单元以及共享L2缓存单元，但是替代实施例可以具有用于指令和数据两者的单个内部缓存，诸如例如，第1级(L1)内部缓存或多级内部缓存。在一些实施例中，系统可以包括内部缓存以及在核和/或处理器外部的外部缓存的组合。可替代地，所有缓存都可以在核和/或处

理器外部。

[0130] 执行电路1311是一个或多个执行电路的集合，包括标量电路1321、向量/SIMD电路1323、和矩阵运算电路1327、以及访问缓存1313的存储器存取电路1325。执行电路对各种类型的数据(例如，标量浮点、压缩整数、压缩浮点、向量整数、向量浮点)执行各种运算(例如，移位、加法、减法、乘法)。虽然一些实施例可以包括专用于具体功能或功能集的许多执行单元，但是其他实施例可以包括仅一个执行单元或全部都执行所有功能的多个执行单元。标量电路1321执行标量运算，向量/SIMD电路1323执行向量/SIMD运算，并且矩阵运算电路1327执行本文详述的矩阵(块片)运算。

[0131] 作为示例，示例性寄存器重命名、乱序发布/执行核架构可以实现如下的流水线：1)指令提取电路执行提取和长度解码级；2)分支和解码电路1303执行解码级；3)分配/重命名电路1307执行分配级和重命名级；4)调度器电路1309执行调度级；5)物理寄存器堆(耦合到或者包括在调度器电路1309及分配/重命名电路1307中)和存储器单元执行寄存器读取/存储器读取级；执行电路1311执行执行级；6)存储器单元和物理寄存器堆单元执行写回/存储器写入级；7)各种单元可能在异常处理级中被涉及；并且8)引退单元和物理寄存器堆单元执行提交级。

[0132] 核可以支持一个或多个指令集(例如，x86指令集(具有已被添加有较新版本的一些扩展)；加利福尼亚州桑尼维尔的MIPS Technologies的MIPS指令集；加利福尼亚州桑尼维尔的ARM Holdings的ARM指令集(具有诸如NEON这样的可选附加扩展)，包括本文描述的指令。在一个实施例中，核1390包括用于支持压缩数据指令集扩展(例如，AVX1、AVX2)的逻辑，从而允许使用压缩数据来执行由许多多媒体应用所使用的运算。

[0133] 应该理解的是，核可以支持多线程处理(执行两个或更多个并行运算或线程集)，并且可以以各种方式这样做，所述各种方式包括时间分片多线程处理、同时多线程处理(其中单个物理核为物理核正在同时地多线程处理的每个线程提供逻辑核)或其组合(例如，时间分片提取和解码以及此后诸如Intel®超线程技术中的同时多线程处理)。

[0134] 图14图示使用块片来支持矩阵运算的处理器核流水线的实施例。分支预测和解码电路1403根据存储在指令存储装置1401中的指令来执行指令的分支预测、指令的解码和/或两者。例如，本文详述的指令可以被存储在指令存储装置中。在一些实施例中，单独的电路被用于分支预测，并且在一些实施例中，使用微码1405来将至少一些指令解码成一个或多个微运算、微码入口点、微指令、其他指令或其他控制信号。可以使用各种不同的机制来实现分支预测和解码电路1403。适合的机制的示例包括但不限于查找表、硬件实施方式、可编程逻辑阵列(PLA)、微码只读存储器(ROM)等。

[0135] 分支预测和解码电路1403耦合到分配/重命名电路1407，其在一些实施例中耦合到调度器电路1409。在一些实施例中，这些电路通过执行以下步骤中的一个或多个来提供寄存器重命名、寄存器分配和/或调度功能：1)将逻辑操作数值重命名为物理操作数值(例如，在一些实施例中为寄存器别名表)，2)将状态位和标志分配给经解码的指令，以及3)调度经解码的指令以供在指令池外的执行电路上执行(例如，在一些实施例中使用预留站)。

[0136] 调度器电路1409表示任何数量的不同调度器，包括预留站、中央指令窗口等。调度器单元调度器电路1409耦合到或者包括物理寄存器堆1415。物理寄存器堆1415中的每一个均表示一个或多个物理寄存器堆，其中的不同物理寄存器堆存储一种或多种不同的数据类

型，诸如标量整数、标量浮点、压缩整数、压缩浮点、向量整数、向量浮点数、状态(例如，作为要执行的下一个指令的地址的指令指针)、块片等。在一个实施例中，物理寄存器堆1415包括向量寄存器电路、写掩码寄存器电路和标量寄存器电路。这些寄存器电路可以提供架构向量寄存器、向量屏蔽寄存器和通用寄存器。物理寄存器堆1415与引退电路1417交叠以图示可以实现寄存器重命名和乱序执行的各种方式(例如，使用重排序缓冲器和引退寄存器堆)；使用未来寄存器堆、历史缓冲器和引退寄存器堆；使用寄存器映射和寄存器池；等)。引退电路1417和物理寄存器堆1415耦合到执行电路1411。

[0137] 虽然在乱序执行的上下文中描述寄存器重命名，但是应该理解的是，可以在顺序架构中使用寄存器重命名。虽然所图示的处理器的实施例还可以包括单独的指令和数据缓存单元以及共享L2缓存单元，但是替代实施例可以具有用于指令和数据两者的单个内部缓存，诸如例如，第1级(L1)内部缓存或多级内部缓存。在一些实施例中，系统可以包括内部缓存以及在核和/或处理器外部的外部缓存的组合。可替代地，所有缓存都可以在核和/或处理器外部。

[0138] 执行电路1411是一个或多个执行电路1427的集合和一个或多个存储器存取电路1425的集合。执行电路1427执行本文详述的矩阵(块片)运算。

[0139] 作为示例，示例性寄存器重命名、乱序发布/执行核架构可以实现如下的流水线：1)指令提取电路执行提取和长度解码级；2)分支和解码电路1403执行解码级；3)分配/重命名电路1407执行分配级和重命名级；4)调度器电路1409执行调度级；5)物理寄存器堆(耦合到或者包括在调度器电路1409和分配/重命名电路1407中)和存储器单元执行寄存器读取/存储器读取级；执行电路1411执行执行级；6)存储器单元和物理寄存器堆单元执行写回/存储器写入级；7)各种单元可能在异常处理级中被涉及；并且8)引退单元和物理寄存器堆单元执行提交级。

[0140] 核可以支持一个或多个指令集(例如，x86指令集(具有已被添加有较新版本的一些扩展)；加利福尼亚州桑尼维尔的MIPS Technologies的MIPS指令集；加利福尼亚州桑尼维尔的ARM Holdings的ARM指令集(具有诸如NEON这样的可选附加扩展)，包括本文描述的指令。在一个实施例中，核1490包括用于支持压缩数据指令集扩展(例如，AVX1、AVX2)的逻辑，从而允许使用压缩数据来执行由许多多媒体应用所使用的运算。

[0141] 应该理解的是，核可以支持多线程处理(执行两个或更多个并行运算或线程集)，并且可以以各种方式这样做，所述各种方式包括时间分片多线程处理、同时多线程处理(其中单个物理核为物理核正在同时地多线程处理的每个线程提供逻辑核)或其组合(例如，时间分片提取和解码以及此后诸如Intel®超线程技术中的同时多线程处理)。

[0142] 布局

[0143] 贯穿本说明书，使用行主要(row major)数据布局来表达数据。列主要(column major)用户应该根据其取向转换项目。图15图示以行主要格式和列主要格式表达的矩阵的示例。如所示，矩阵A是2×3矩阵。当以行主要格式存储此矩阵时，行的数据元素是连续的。当以列主要格式存储此矩阵时，列的数据元素是连续的。矩阵的公知属性是$A^T*B^T=(BA)^T$，其中上标T意指转置。读取列主要数据作为行主要数据导致矩阵看起来像转置矩阵。

[0144] 在一些实施例中，在硬件中利用行主要语义，并且列主要数据是为了交换操作数顺序，其结果是矩阵的转置，但是对于来自存储器的后续列主要读取，它是正确的非转置矩

阵。

[0145] 例如,如果有两个列主矩阵要相乘:

[0146]
a b	g i k	ag+bh ai+bj ak+bl
c d *	h j l =	cg+dh ci+dj ck+dl
e f		eg+fh ei+fj ek+fl
(3x2)	(2x3)	(3x3)

[0147] 输入矩阵将被存储在线性存储器(列主要)中如下:

[0148] a c e b d f

[0149] 和

[0150] g h i j k l。

[0151] 将那些矩阵读取为具有维度2x3和3x2的行主要矩阵,它们将看起来是:

[0152] a c e和g h

[0153] b d f i j

[0154] k l

[0155] 交换顺序和矩阵乘:

[0156]
g h	a c e		ag+bh cg+dh eg+fh
i j	*	b d f =	ai+bj ci+dj ei+fj
k l			ak+bl ck+dl ek+fl

[0157] 转置矩阵输出并且然后可以行主要顺序被存储:

[0158] ag+bh cg+dh eg+fh ai+bjci+dj ei+fj ak+bl ck+dl ek+fl

[0159] 并且用在后续列主要计算中,它是正确的非转置矩阵:

[0160]
ag+bh	ai+bj	ak+bl

[0161]
cg+dh	ci+dj	ck+dl

[0162]
eg+fh	ei+fj	ek+fl

[0163] 示例性使用

[0164] 图16图示矩阵(块片)的使用的示例。在此示例中,矩阵C 1601包括两个块片,矩阵A 1603包括一个块片,并且矩阵B 1605包括两个块片。此图示出用于计算矩阵乘法的算法的内循环的示例。在此示例中,来自矩阵C 1601的两个结果块片tmm0和tmm1用于累加中间结果。来自矩阵A 1603的一个块片(tmm2)被重用两次,因为它与来自矩阵B 1605的两个块片相乘。指针用于从通过箭头指示的方向加载新A矩阵(块片)和两个新B矩阵(块片)。未示出的外循环调整用于C块片的指针。

[0165] 如所示的示例性代码包括块片配置指令的使用并且被执行来配置块片使用,加载块片,循环处理块片,将块片存储到存储器,并且释放块片使用。

[0166] 图17图示矩阵(块片)的使用的实施例。在1701处,配置块片使用。例如,执行TILECONFIG指令以配置块片使用,包括设置每块片的行数和列数。通常,在1703处从存储器加载至少一个矩阵(块片)。使用矩阵(块片)来在1705处执行至少一个矩阵(块片)运算。在

1707处,将至少一个矩阵(块片)存储输出到存储器并且在1709处可发生上下文切换。

[0167] 示例性配置

[0168] 块片配置硬件支持

[0169] 如上面所讨论的,通常需要在使用之前配置块片使用。例如,可能不需要所有行和列的完全使用。在一些实施例中不仅不配置这些行和列节约功率,而且配置可以用于确定运算是否将生成错误。例如,如果M和L不相同,则形式(N×M)*(L×N)的矩阵乘法通常将不工作。

[0170] 在使用利用块片的矩阵之前,在一些实施例中,将配置块片支持。例如,配置每块片的行数和列数、要使用的块片等。TILECONFIG指令是对计算机本身的改进,因为它提供支持以将计算机配置成使用矩阵加速器(作为处理器核的一部分或者作为外部设备)。特别地,TILECONFIG指令的执行使得配置从存储器中被检索并应用于矩阵加速器内的矩阵(块片)设置。

[0171] 块片使用配置

[0172] 图18图示根据实施例的对块片的使用的配置的支持。存储器1801包含要支持的矩阵(块片)的块片描述1803。

[0173] 处理器/核1805的指令执行资源1811将块片描述1803的各方面存储到块片配置1817中。块片配置1817包括用于详述调色板的哪些块片被配置(每个块片中的行数和列数)的调色板表1813和关于矩阵支持在使用中的标记。特别地,指令执行资源1811被配置为使用如通过块片配置1817所指定的块片。指令执行资源1811还可以包括机器特定寄存器或配置寄存器以用于指示块片使用。还设置了诸如在使用中和起始值这样的附加值。块片配置1817利用寄存器1819来存储块片使用和配置信息。

[0174] 图19图示要支持的矩阵(块片)的描述的实施例。这是要在执行STTILECFG指令时存储的描述。在此示例中,每个字段是一字节。在字节[0]中,存储调色板ID 1901。调色板ID用于为调色板表1813编索引,所述调色板表1813按调色板ID存储块片中的字节数以及与如通过配置所定义的此ID相关联的块片的每行字节数。

[0175] 字节1存储要存储在“startRow”寄存器1903中的值并且字节2存储要存储在寄存器startP 1905中的值。为了支持在这些事件之后重新启动指令,指令将信息存储在这些寄存器中。为了支持在诸如上面详述的那些中断事件之后重新启动指令,指令将信息存储在这些寄存器中。startRow值指示应该被用于重新启动的行。startP值指示当使用块片对时用于存储操作的行内的位置,并且在一些实施例中,指示行的下半部(在一对的较低块片中)或行的上半部(在一对的较高块片中)。通常,不需要行(列)中的位置。

[0176] 除了TILECONFIG和STTILECFG之外,成功地执行矩阵(块片)指令会将startRow和startP两者设置为零。

[0177] 无论何时未重新启动中断的矩阵(块片)指令,软件有责任使startRow和startP值归零。例如,未屏蔽的浮点异常处理程序可能决定在软件中完成运算并且将程序计数器值改变为另一指令,通常是下一个指令。在这种情况下,软件异常处理程序必须在恢复程序之前使操作系统呈现给它的异常中的startRow和startP值归零。操作系统将随后使用恢复指令来重新加载那些值。

[0178] 字节3存储块片1907的对(每块片1b)的指示。

[0179] 字节16-17存储用于块片0的行数1913和列数1915，字节18-19存储用于块片1的行数和列数等。换句话说，每个2字节组指定用于块片的行数和列数。如果不使用2个字节的组来指定块片参数，则它们应该具有值零。为多于实施方式所限制或者调色板所限制的块片指定块片参数导致故障。未配置的块片被设置为具有0行、0列的初始状态。

[0180] 最后，存储器中的配置通常以诸如若干连续字节全为零这样的结束划界而结束。

[0181] 示例性块片和块片配置存储

[0182] 图20(A)至图20(D)图示寄存器1819的示例。图20(A)图示多个寄存器1819。如所示，每个块片(TMM0 2001...TMMN 2003)具有单独的寄存器，其中每个寄存器存储用于该特定块片的行和列大小。StartP 2011和StartRow 2013被存储在单独的寄存器中。一个或多个状态寄存器2015被设置(例如，TILES_CONFIGURED＝1)以指示块片被配置以供使用。

[0183] 图20(B)图示多个寄存器1819。如所示，每个块片对于其行和列具有单独的寄存器。例如，TMM0行配置2021、TMM0列配置2023、StartP 2011和StartRow 2013被存储在单独的寄存器中。一个或多个状态寄存器2015被设置(例如，TILES_CONFIGURED＝1)以指示块片被配置以供使用。

[0184] 图20(C)图示单个寄存器1819。如所示，此寄存器存储块片配置(每块片的行和列)2031，StartP 2011和StartRow 2013被存储在作为压缩数据寄存器的单个寄存器中。一个或多个状态寄存器2015被设置(例如，TILES_CONFIGURED＝1)以指示块片被配置以供使用。

[0185] 图20(D)图示多个寄存器1819。如所示，单个寄存器存储块片配置(每块片的行和列)2031。StartP和StartRow被存储在单独的寄存器2011和2013中。一个或多个状态寄存器2015被设置(例如，TILES_CONFIGURED＝1)以指示块片被配置以供使用。

[0186] 设想了其他组合，诸如将起始寄存器组合成其中它们被单独示出的单个寄存器等。

[0187] TILEUNPACKAOS/TILEPACKSOA

[0188] 如上所述，不同类型的数据常常用基于一种关系的逻辑分组加以表示，但是然后有时需要基于不同的关系被“解包”。例如，数字图像通常被表示为结构数组(AOS)，其中结构表示各自具有以下三种类型元素的像素：R(红色)、G(绿色)和B(蓝色)。表示图像的AOS文件通常将散布的所有三种元素类型的所有像素打包在一起。然而，在某些场合，需要跨越多个像素对元素类型中的仅一种(例如R)执行运算。

[0189] 所公开的实施例描述了TileUnpackAOS指令，响应于该指令，处理器对AOS数组的不同元素类型进行解包，并且将类似类型的元素分组到结果数组的不同行、结果数组的不同列中，或者分组到专用于每种类型的元素的不同数组。不同类型的元素被分离并分组在一起的结果矩阵被称为数组结构(SOA)。所公开的实施例还描述了TilePackSOA指令，响应于所述该指令，处理器将对不同的元素类型进行打包，使它们交错在AOS结果数组中。TileUnpackAOS和TilePackSOA指令因此可用于从结构数组(AOS)转换为数组结构(SOA)并且反之亦然。

[0190] 在相同类型的元素上运算的替代方法通常涉及在数据的较小部分(例如RGB的一个组或者可能是单个缓存行)上运算。已使用了“汇集(Gather)”样式指令，但是它们更复杂并且需要更复杂的细节以进行地址计算。替代方法在指令中有较少的数据可用，这减少可针对在大块数据上重复此解包的常见情况所做的微架构优化的量。替代方法还限制组内的

步幅和元素计数,使得它适合最大块大小。

[0191] 所公开的实施例通过以下方式来提供对处理器性能和能力的改进,所述方式为描述指令以用于将使不同类型的元素打包在一起的矩阵解包成替代的分离组织。可将矩阵逻辑上创建为结构数组,像上述的数字图像示例一样,但是所公开的TileUnpackAOS指令允许将类似的元素分组在一起以进行更高效的并行处理。所公开的TileUnpackAOS和TilePackSOA指令可由处理器用来快速地从结构数组(AOS)转换为数组结构(SOA)并且反之亦然。

[0192] 有利地,所公开的TileUnpackAOS和TilePackSOA指令相对于替代或先前的方法有所改进,至少它们利用二维寄存器空间(块片)的益处来为针对这些变换的微架构优化提供更多的机会。它们还提供简单且灵活的使用模型,以允许软件利用这些指令来为向量化和更高效的处理增加机会。

[0193] 如在下面进一步描述的并在附图中图示的,所公开的TileUnpackAOS指令使得能够将相同类型的元素从AOS(结构数组)矩阵高效地解包成SOA(数组结构)矩阵。常见示例是将包含大块RGB像素的数字图像解包成其组成R、G和B,以使得能实现更高的并行性能和改进的向量化。所公开的TilePackSOA指令允许处理器反转此操作,并且将不同类型的元素打包或者交错成结构数组(AOS)。

[0194] 如上面所提及的,所公开的TileUnpackAOS和TilePackSOA指令的有益使用的一个示例是将包含具有三个不同类型的元素('R'、'G'和'B')的像素(结构)数组的数字图像解包成中间数组的不同行或列,并行处理不同类型的元素,然后将中间结果打包回到AOS格式。

[0195] 更多的示例性用途是可能的,诸如用于处理具有采用增加的元素类型'A'(表示Alpha)的RGBA像素的图像。

[0196] 所公开的TileUnpackAOS和TilePackSOA指令的另一示例性用途是用于处理复数数据的矩阵或数组,其中每个元素包含两种元素类型:实数和虚数,并且类似类型的元素被步幅距离2分开。使用所公开的TileUnpackAOS和TilePackSOA允许处理器对复数的矩阵(块片)进行解包,并行对相同类型的元素进行操作,然后将元素打包回到复数数组。

[0197] 公开的TILEUNPACKAOS和TILEPACKSOA的执行

[0198] 在操作中,处理器或类似的计算装置将通过使用提取电路提取指定操作码和以下各项的位置的指令来对TileUnpackAOS和TilePackSOA指令做出响应:包含N个交错结构的结构数组(AOS)矩阵,每个交错结构由具有不同类型的K个元素组成,连续结构中的相同类型的元素以步幅隔开;以及一起包含K个分离的元素组的一个或多个数组结构(SOA)矩阵,每个组包含N个相同类型的元素,其中,K个组中的每个组均设置在专用矩阵中或者被分组在共享矩阵的一部分中。处理器或计算装置将通过以下步骤继续执行:使用解码电路来对所提取的指令进行解码;以及通过使用执行电路进行以下操作来对经解码的解包指令做出响应:当操作码指定从AOS到SOA的转换时,将所指定的AOS矩阵的每个元素分类为K种元素类型中的一种,并且将每个经分类的元素写入到一个或多个SOA矩阵中的相同类型的元素的对应组中;而当操作码指定从SOA到AOS的转换时,从SOA矩阵的K个组中的每个组一次存储一个元素,使每个存储的元素与具有不同类型的元素交错,并且使每个相同类型的元素以步幅隔开。

[0199] TILEUNPACKAOS和TILEPACKSOA指令行为的变化

[0200] 公开的实施例包括对TileUnpackAOS和TilePackSOA指令的若干添加和变化，如在下面所讨论的。

[0201] 在一些实施例中，辅助“步幅”被输入到指令并且允许解包(或打包)算法从存储器中的一个位置加载(或者存储)许多元素，然后向前“跳”到下一个元素块。这在需要对分量进行解包(或者打包)同时跨过较大块的算法中可能是有用的。示例将是需要对图片的每一行的第一像素的R、G和B进行解包的一些情况(步幅将允许向前跳到下一行)。

[0202] 此外，在一些实施例中，TileUnpackAOS和TilePackSOA指令允许在从存储器加载以及寄存器到寄存器移动时解包。

[0203] 另外，在一些实施例中，TileUnpackAOS和TilePackSOA指令允许从寄存器到寄存器或到存储器的打包。

[0204] 另外，在一些实施例中，TileUnpackAOS和TilePackSOA指令对于元素8、16、32和64位的常见大小(但是其他大小是可能的)允许基于元素的解包。

[0205] 另外，在一些实施例中，TileUnpackAOS和TilePackSOA指令允许多于4组元素的打包和解包，但是2、3或4组的打包和解包是最常见的。

[0206] 另外，在一些实施例中，TileUnpackAOS和TilePackSOA指令允许基于元素的解包到单独的块片寄存器、单个块片寄存器的单独的行以及单个块片寄存器的单独的列中(或者从单独的块片寄存器、单个块片寄存器的单独的行以及单个块片寄存器的单独的列打包)。

[0207] 示例性执行

[0208] 图21A图示根据一些实施例的TileUnpackAOS指令的示例性执行。如所示，TileUnpackAOS指令2100包括用于指定操作码2102(例如，TileUnpackAOS)、源位置2104(在这种情况下为AOS矩阵)和目标2106(在这种情况下为SOA矩阵)的字段。通过源位置2104和目标2106所指定的矩阵中的每个矩阵可以在向量寄存器或块片寄存器的合集中，或者在存储器的区域中。另外示出的是源2110矩阵：作为数字图像文件并且包含多个像素的AOS矩阵，每个像素被指定为‘R’、‘G’或‘B’。另外示出的是执行电路2112，其包括解包电路2114和目标矩阵2116。

[0209] 在所图示的系统的上下文中，解码电路类似于至少关于图13、图14及图28A和图28B所图示和描述的解码电路。

[0210] 关于图3至图14进一步图示并描述执行电路。在一些实施例中，执行电路是矩阵运算加速器，诸如被图示和描述为加速器307(图3)的加速器。在一些实施例中，执行电路是矩阵运算电路，诸如矩阵运算电路405(图4)、505(图5)或1213(图12)和1327(图13)。

[0211] 在操作中，提取和解码电路(未示出)用于提取指令2100并对指令2100进行解码，所述指令2100具有用于指定操作码2102以及源矩阵位置2104和目标矩阵位置2106的字段。响应于操作码2102指定从AOS到SOA的转换，执行电路2112将所指定的源(AOS)矩阵2110的每个元素分类为K种元素类型(这里，K＝3)中的一种，并且将每个经分类的元素写入到所指定的目标矩阵2116中的相同类型的元素的对应组中。

[0212] 图21B图示根据一些实施例的TileUnpackAOS指令的执行。如所示，TileUnpackAOS指令2120包括用于指定操作码2122(例如，TileUnpackAOS)、源位置2124(在这种情况下为

AOS矩阵)和目标位置2126(在这种情况下为SOA矩阵)的字段。通过源位置2124和目标位置2126所指定的矩阵中的每个矩阵均可以在向量寄存器或块片寄存器的合集中,或者在存储器的区域中。另外示出的是所指定的源矩阵2130:包含四种类型的元素的AOS矩阵,每个元素被指定为Δ(希腊语Delta)、α(希腊语字母alpha)、μ(希腊语字母mu)或∞(无穷大符号)中的一个,以及指定的目标矩阵2136。这里,与图21A不同,除了解包电路2133之外,执行电路2132被示出为可选地包括变换电路2134和运算电路2135,以执行以下操作中的一者或两者:对元素进行变换(例如从单精度浮点到双精度浮点)或者随着元素经过执行电路2132到指定的目标2136上而对元素执行算术或逻辑运算(例如一元运算,例如INCREMENT)。可选的变换电路2134和运算电路2135预期将改进包括有它们的处理器的效率和性能。

[0213] 关于图3至图14进一步图示和描述执行电路。在一些实施例中,执行电路是矩阵运算加速器,诸如被图示和描述为加速器307(图3)的加速器。在一些实施例中,执行电路是矩阵运算电路,诸如矩阵运算电路405(图4)、505(图5)、1213(图12)或1327(图13)。

[0214] 在操作中,提取和解码电路(未示出)将提取指令2120并对指令2120进行解码,所述指令2120具有用于指定操作码2122、源矩阵位置2124和目标矩阵位置2126的字段。响应于操作码2122指定从AOS到SOA的转换,执行电路2132将所指定的源(AOS)矩阵2130的每个元素分类为K种元素类型(这里,K=4)中的一种,并且将每个经分类的元素写入到所指定的目标矩阵2136中的相同类型的元素的对应组中。

[0215] 在所图示的系统的上下文中,解码电路类似于至少关于图13、图14及图28A和图28B所图示和描述的解码电路。

[0216] 图21C图示根据一些实施例的TileUnpackAOS指令的执行。如所示,TileUnpackAOS指令2140包括用于指定操作码2142(例如,TileUnpackAOS)、源位置2144(在这种情况下为AOS矩阵)和目标位置2146(在这种情况下为SOA矩阵)的字段。通过源位置2144和目标位置2146所指定的矩阵中的每个矩阵均可以在向量寄存器或块片寄存器的合集中,或者在存储器的区域中。另外示出的是所指定的源矩阵2150和所指定的目标矩阵2156,每个矩阵包含四种类型的元素,为了简化描述,所述四种类型的元素与所指定的源2130(图21B)中的四种类型的元素相同。这里,与图21A不同,除了解包电路2153之外,执行电路2152还被示为可选地包括变换电路2154和运算电路2155,以对元素进行变换(例如从单精度浮点到双精度浮点)或者随着元素经过执行电路2152到所指定的目标2156而对元素执行运算(例如,一元运算,例如INCREMENT),或者兼而有之。可选的变换电路2154和操作电路2155预期将改进包括有它们的处理器的效率和性能。

[0217] 关于图3至图14进一步图示和描述执行电路。在一些实施例中,执行电路是矩阵运算加速器,诸如被图示和描述为加速器307(图3)的加速器。在一些实施例中,执行电路是矩阵运算电路,诸如矩阵运算电路405(图4)、505(图5)或1213(图12)和1327(图13)。

[0218] 在操作中,提取和解码电路(未示出)将提取指令2140并对指令2140进行解码,所述指令2140具有用于指定操作码2142、源矩阵位置2144和目标矩阵位置2146的字段。响应于操作码2142指定从AOS到SOA的转换,执行电路2152将所指定的源(AOS)矩阵2150的每个元素分类为K种元素类型(这里,K=4)中的一种,并且将每个经分类的元素写入到K个矩阵中的一个中,每个矩阵保持相同类型的数据。

[0219] 在所图示的系统的上下文中,解码电路类似于至少关于图13、图14及图28A和图

28B所图示和描述的解码电路。

[0220] 图22A图示根据一些实施例的TilePackSOA指令的执行。如所示，TilePackSOA指令2200包括用于指定操作码2202(TilePackSOA)、源位置2204(这里是SOA矩阵)，目标位置2206(这里是AOS矩阵)和步幅(stride)2208的字段。像(图21A的)所指定的AOS源矩阵2110一样，指定的源SOA矩阵2210在这里是数字图像文件并且包含不同类型的多个分离的像素元素，被指定为‘R’、‘G’或‘B’。所指定的源2210和所指定的目标矩阵2216中的每个矩阵可以在向量寄存器或块片寄存器的合集中，或者在存储器的区域中。另外示出的是执行电路2212，其包括打包电路2214。

[0221] 关于图3至图14进一步图示和描述执行电路。在一些实施例中，执行电路是矩阵运算加速器，诸如被图示和描述为加速器307(图3)的加速器。在一些实施例中，执行电路是矩阵运算电路，诸如矩阵运算电路405(图4)、505(图5)或1213(图12)和1327(图13)。

[0222] 在操作中，提取和解码电路(未示出)将提取指令2200并对指令2200进行解码，所述指令2200具有用于指定操作码2202、源矩阵位置2204和目标矩阵位置2206的字段。响应于操作码2202指定从SOA到AOS的转换，执行电路2212将从所指定的源(SOA)矩阵2210的K个组中的每个组一次存储一个元素，在所指定的目标(AOS)矩阵2216中使每个存储的元素与具有不同类型的元素交错，并且使每个相同类型的元素以步幅隔开。

[0223] 在所图示的系统的上下文中，解码电路类似于至少关于图13、图14及图28A和图28B所图示和描述的解码电路。

[0224] 图22B图示根据一些实施例的TilePackSOA指令的执行。如所示，TilePackSOA指令2220包括用于指定操作码2222(TilePackSOA)、源位置2224(这里是SOA矩阵)、目标位置2226(这里是AOS矩阵)和步幅2228的字段。指定的源2230矩阵和指定的目标2236矩阵中的每一个均可以在向量寄存器或块片寄存器的合集中，或者在存储器的区域中。指定的源2230矩阵包含四种类型的元素，为了简化描述，所述四种类型的元素与源2130(图21B)中的四种类型的元素相同。这里，执行电路2232包括分类电路2233，并且与图22A不同，包括可选的变换电路2234和运算电路2235，以对元素进行变换(例如从单精度浮点到双精度浮点)或者随着元素经过执行电路2232到达目标2236而对元素执行运算(例如一元运算，例如INCREMENT)，或者兼而有之。可选的变换电路2234和运算电路2235将用来改进包括有它们的处理器的效率和性能。

[0225] 在操作中，提取和解码电路(未示出)将提取指令2220并对指令2220进行解码，所述指令2220具有用于指定操作码2222、源矩阵位置2224、目标矩阵位置2226和步幅2228的字段。在一些实施例中，如这里一样，执行电路2232将所指定的步幅2228视为源矩阵2224中的元素类型的数量K。响应于操作码2222指定从SOA到AOS的转换，执行电路2232将从所指定的源(SOA)矩阵2230的K个组中的每个组一次存储一个元素，使每个存储的元素与具有不同类型的元素交错，并且使每个相同类型的元素以步幅隔开。

[0226] 在所图示的系统的上下文中，解码电路类似于至少关于图13、图14及图28A和图28B所图示和描述的解码电路。

[0227] 图22C图示根据一些实施例的TilePackSOA指令的执行。如所示，TilePackSOA指令2240包括用于指定操作码2242(TilePackSOA)、源位置2244(这里是SOA矩阵)、目标位置2246(这里是AOS矩阵)和步幅2248的字段。所指定的源2250和目标2256矩阵中的每一个均

可以在向量寄存器或块片寄存器的合集中，或者在存储器的区域中。所指定的源2250矩阵和所指定的目标2256矩阵各自包含四种类型的元素，为了简化描述，所述四种类型的元素与源2130(图21B)中的四种类型的元素相同。这里，执行电路2252包括分类2253电路，并且与图22A不同，包括可选的变换电路2254和运算电路2255，以对元素进行变换(例如从单精度浮点到双精度浮点)或者随着元素经过执行电路2252到达目标2256而对元素执行运算(例如一元运算，例如INCREMENT)，或者兼而有之。可选的变换电路2254和运算电路2255预期将改进包括有它们的处理器的效率和性能。

[0228] 在操作中，提取和解码电路(未示出)将提取指令2240并对指令2240进行解码，所述指令2240具有用于指定操作码2242、源矩阵位置2244、目标矩阵位置2246和步幅2248的字段。在一些实施例中，如这里一样，执行电路2252将所指定的步幅2248视为所指定的源矩阵2250中的元素类型的数量K。响应于操作码2242指定从SOA到AOS的转换，执行电路2252将从所指定的源(SOA)矩阵的K个组中的每个组一次存储一个元素，使每个存储的元素与具有不同类型的元素交错，并且使每个相同类型的元素以步幅隔开。

[0229] 执行的示例性方法

[0230] 图23A是图示处理器执行TileUnpackAOS指令的实施例的流程图。在2301处，处理器将使用提取电路来提取指令，其格式包括用于指定操作码以及结构数组(AOS)源矩阵和一个或多个数组结构(SOA)目标矩阵的位置的字段，其中：所指定的操作码要求将所指定的AOS源矩阵的元素解包到所指定的数组结构(SOA)目标矩阵中；AOS源矩阵将包含N个结构，每个结构包含不同类型的K个元素，其中连续结构中的相同类型的元素以步幅隔开；并且SOA目标矩阵一起包含K个分离的组，每个组包含N个相同类型的元素，其中，K个组中的每个组均设置在专用矩阵中或者在共享矩阵中；

[0231] 在2303处，处理器将使用解码电路来对所提取的指令进行解码。在所图示的系统的上下文中，解码电路类似于至少关于图13、图14及图28A和图28B所图示和描述的解码电路。

[0232] 在一些实施例中，在2305处，处理器将调度经解码的指令的执行。

[0233] 在2307处，处理器将使用执行电路来通过以下操作来对经解码的指令做出响应，所述操作是将所指定的AOS矩阵的每个元素解包成K种元素类型中的一种并且将每个未解包的元素写入到一个或多个SOA矩阵中的相同类型的元素的对应组中。在一些实施例中，执行电路是矩阵运算电路，诸如矩阵运算电路405(图4)、505(图5)或1213(图12)和1327(图13)。

[0234] 在一些实施例中，在2309，处理器将提交所执行的指令的结果。操作2303和2309是可选的，如通过虚线边界所指示的，每个操作可以在不同的时间发生或者根本不发生。

[0235] 图23B是图示处理器执行TilePackSOA指令的实施例的流程图。在2351处，处理器使用提取电路来提取指令，其格式包括用于指定操作码以及一个或多个数组结构(SOA)源矩阵和一个结构数组(AOS)目标矩阵的位置的字段；其中，所指定的操作码将指示从所指定的SOA矩阵到所指定的AOS矩阵的移动；所指定的SOA源矩阵一起包含K个分离的组，每个组包含N个相同类型的元素，其中，K个组中的每一个设置在专用矩阵中或者在共享矩阵中；并且AOS目标矩阵将包含N个结构，每个结构包含不同类型的K个元素，其中连续结构中的相同类型的元素以步幅隔开。

[0236] 在2353处，处理器将使用解码电路来对所提取的指令进行解码。在所图示的系统的上下文中，解码电路类似于至少关于图13、图14及图28A和图28B所图示和描述的解码电路。

[0237] 在2355处，处理器在一些实施例中将调度经解码的指令的执行。

[0238] 在2357处，处理器将使用执行电路通过以下操作来对经解码的指令做出响应，所述操作是将所指定的SOA源矩阵的每个元素分类为K种元素类型中的一种并且将每个经分类的元素写入到所指定的AOS目标矩阵中的其对应位置。在一些实施例中，执行电路是矩阵运算电路，诸如矩阵运算电路405(图4)、505(图5)或1213(图12)和1327(图13)。

[0239] 在2359处，处理器在一些实施例中将提交所执行的指令的结果。

[0240] 示例性指令格式

[0241] 图24是图示根据一些实施例的TileUnpackAOS/TilePackSOA指令的格式的框图。如所示，TileUnpackAOS/TilePackSOA指令2400包括用于指定操作码2402(例如，TileUnpackAOS或TilePackSOA)以及AOS矩阵2404和SOA矩阵2406的字段，取决于操作码，所述AOS矩阵2404和所述SOA矩阵2406中的一个将提供源，并且所述AOS矩阵2404和所述SOA矩阵2406中的另一个将提供指令的目标。当操作码指定TileUnpackAOS时，所指定的AOS矩阵2404是源，而当操作码指定TilePackSOA时，所指定的SOA矩阵2406是源。

[0242] TileUnpackAOS/TilePackSOA指令2400还包括若干可选的参数，包括M(源矩阵的行数)2408、N(源矩阵的列数)2410、K(源矩阵的元素类型数)2412以及元素大小2414，其可以是微字节(crumb)、半字节、字节、字、双字和四字中的任何一种。另外，可选地包括的是：步幅2416，其指定连续元素组中的相同类型的元素之间的距离；以及辅助步幅2418，其指定元素组之间的距离。

[0243] 应当注意的是，在一些实施例中，处理器将对不包含任何可选参数的TileUnpackAOS/TilePackSOA指令2400做出响应。在一些情况下，例如对于源矩阵的维度M和N，处理器假定架构默认值。在一些实施例中，经由操作码2402的前缀或后缀来指定可选行为中的一个或多个(这是操作码用星号示出以表示可选前缀和后缀的原因)。在一些实施例中，经由配置/状态寄存器(例如，XTILECONFIG)指定指令修正符2408、2410、2412、2414、2416和2418中的一个或多个。

[0244] 详细的示例性系统、处理器和仿真

[0245] 本文详述了用于执行上述指令的硬件、软件等的示例。例如，在下面描述的内容详述指令执行的各方面，包括诸如提取、解码、调度、执行、引退等这样的各种流水线级。

[0246] 指令集

[0247] 指令集可以包括一个或多个指令格式。给定指令格式可以定义各种字段(例如，位数、位的位置)以指定要执行的操作(例如，操作码)和将在其上执行该操作的操作数和/或其他数据字段(例如，掩码)等。一些指令格式通过指令模板(或子格式)的定义被进一步细分。例如，给定指令格式的指令模板可以被定义为具有指令格式的字段的不同子集(所包括的字段通常具有相同的顺序，但是至少一些具有不同的位位置，因为包括较少的字段)和/或被定义为具有不同解释的给定字段。因此，ISA的每个指令使用给定的指令格式表示(并且，如果定义，则在该指令格式的给定的一个指令模板中)并且包括用于指定操作和操作数的字段。例如，示例性ADD指令具有特定操作码和指令格式，该指令格式包括指定该操作码

的操作码字段和选择操作数的操作数字段(源1/目的地和源2);并且在指令流中出现该ADD指令将在操作数字段中具有选择特定操作数的特定内容。已发布和/或公布了被称为高级向量扩展(AVX)(AVX1和AVX2)并使用向量扩展(VEX)编码方案的一组SIMD扩展(例如,参见Intel®64和IA-32架构软件开发人员手册,2014年9月;参见Intel®高级向量扩展编程参考,2014年10月)。

[0248] 示例性指令格式

[0249] 本文描述的指令的实施例可以以不同的格式实现。另外,以下详细描述示例性系统、架构和流水线。指令的实施例可以在这样的系统、架构和流水线上执行,但不限于详细说明的那些。

[0250] 通用向量友好指令格式

[0251] 向量友好指令格式是适合于向量指令的指令格式(例如,存在特定于向量操作的某些字段)。虽然描述了通过向量友好指令格式支持向量和标量操作的实施例,但替代实施例仅使用向量友好指令格式的向量操作。

[0252] 图25A-25B是示出根据一些实施例的通用向量友好指令格式及其指令模板的框图。图25A是示出根据一些实施例的通用向量友好指令格式及其A类指令模板的框图;图25B是示出根据一些实施例的通用向量友好指令格式及其B类指令模板的框图。具体地,为通用向量友好指令格式2500定义A类和B类指令模板,两者都包括无存储器访问2505指令模板和存储器访问2520指令模板。在向量友好指令格式的上下文中,术语"通用"是指不与任何特定指令集相捆绑的指令格式。

[0253] 虽然将描述其中向量友好指令格式支持以下内容的实施例:具有32位(4字节)或64位(8字节)数据元素宽度(或大小)的64字节向量操作数长度(或者大小)(因此,64字节向量由16个双字大小的元素组成,或者可替换地由8个四字大小的元素组成);具有16位(2字节)或8位(1字节)数据元素宽度(或大小)的64字节向量操作数长度(或大小);具有32位(4字节)、64位(8字节)、16位(2字节)或8位(1字节)数据元素宽度(或大小)的32字节向量操作数长度(或大小);以及具有32位(4字节)、64位(8字节)、16位(2字节)或8位(1字节)数据元素宽度(或大小)的16字节向量操作数长度(或大小);但是替代实施例可以支持具有更多、更少或不同数据元素宽度(例如,128位(16字节)数据元素宽度)的更多、更少和/或不同的向量操作数大小(例如,256字节向量操作数)。

[0254] 图25A中的A类指令模板包括:1)在无存储器访问2505指令模板内,示出了无存储器访问、全舍入控制类型操作2510指令模板和无存储器访问、数据变换类型操作2515指令模板;2)在存储器访问2520指令模板内,示出了存储器访问、时态(temporal)2525指令模板和存储器访问、非时态2530指令模板。图25B中的B类指令模板包括:1)在无存储器访问2505指令模板内,示出了无存储器访问、写掩码控制、部分舍入控制类型操作2512指令模板和无存储器访问、写掩码控制、vsize类型操作2517指令模板;2)在存储器访问2520指令模板内,示出了存储器访问、写掩码控制2527指令模板。

[0255] 通用向量友好指令格式2500包括以下以图25A-25B所示的顺序列出的以下字段。

[0256] 格式字段2540-该字段中的特定值(指令格式标识符值)唯一地标识向量友好指令格式,因此标识指令在指令流中以向量友好指令格式出现。因此,该字段是可选的,仅具有通用向量友好指令格式的指令集是不需要该字段的。

[0257] 基本操作字段2542-其内容区分不同的基本操作。

[0258] 寄存器索引字段2544-其内容直接或通过地址生成来指定源和目标操作数的位置,无论它们在寄存器中还是在存储器中。这些包括足够数量的位以从PxQ(例如32x512、16x128、32x1024、64x1024)寄存器文件中选择N个寄存器。虽然在一个实施例中,N可以是多达三个源和一个目的地寄存器,但是替代实施例可以支持更多或更少的源和目的地寄存器(例如,可以支持多达两个源,其中这些源中的一个也充当目的地,可以支持多达三个源,其中这些源中的一个也充当目的地,可能支持多达两个源和一个目的地)。

[0259] 修正符字段2546-其内容区分通用向量指令格式中指定存储器访问的指令的出现与不指定存储器访问的指令的出现;即,区分无存储器访问2505指令模板和存储器访问2520指令模板。存储器访问操作读取和/或写入存储器层级结构(在一些情况下使用寄存器中的值指定源和/或目的地地址),而非存储器访问操作不这样做(例如,源和目的地是寄存器)。虽然在一个实施例中,该字段还在三种不同方式之间进行选择以执行存储器地址计算,但替代实施例可以支持更多、更少或不同的方式来执行存储器地址计算。

[0260] 增强操作字段2550-其内容区分除基本操作之外还要执行各种不同操作中的哪一个。该字段是上下文特定的。在本发明的一个实施例中,该字段被分成类字段2568,α字段2552和β字段2554。增强操作字段2550允许在单个指令中而不是2、3或4个指令中执行公共操作组。

[0261] 缩放字段2560-其内容允许缩放索引字段的内容以用于存储器地址生成(例如,用于使用2^{scale}*index+base的地址生成)。

[0262] 位移字段2562A-其内容用作存储器地址生成的一部分(例如,用于使用2^{scale}*index+base+displacement的地址生成)。

[0263] 位移因子字段2562B(注意,直接在位移因子字段2562B上的位移字段2562A的并置指示使用一个或另一个)-其内容用作地址生成的一部分;它指定位移因子,该位移因子将通过存储器访问的大小(N)来缩放-其中N是存储器访问中的字节数(例如,对于使用2^{scale}*index+base+scaled displacement的地址生成)。忽略冗余低阶位,因此,位移因子字段的内容乘以存储器操作数总大小(N),以便生成用于计算有效地址的最终位移。N的值由处理器硬件在运行时基于完整操作码字段2574(稍后描述)和数据操纵字段2554C来确定。位移字段2562A和位移因子字段2562B从以下意义上来说是可选的,例如它们不用于无存储器访问2505指令模板和/或不同实施例可以仅实现这两者中的一个或不实现这两者。

[0264] 数据元素宽度字段2564-其内容区分要使用多个数据元素宽度中的哪一个(在一些实施例中针对所有指令;在其他实施例中仅针对一些指令)。该字段从以下意义上来说是可选的,例如如果仅支持一个数据元素宽度和/或使用操作码的某些方面支持数据元素宽度则不需要该字段。

[0265] 写掩码(write mask)字段2570-其内容基于每个数据元素位置控制目的地向量操作数中的数据元素位置是否反映了基本操作和增强操作的结果。A类指令模板支持合并写掩码,而B类指令模板支持合并和归零写掩码。合并时,向量掩码允许在执行任何操作(由基本操作和增强操作指定)期间保护目的地中的任何元素集免于更新;在另一个实施例中,保留相应掩码位具有0值的目的地的每个元素的旧值。相反,当归零向量掩码允许在执行任何操作(由基本操作和增强操作指定)期间将目的地中的任何元素集归零;在一个实施例中,

当相应的掩码位具有0值时，目的地的元素被设置为0。该功能的一个子集是能够控制正在执行的操作的向量长度(即，从第一个到最后一个被修改的元素的跨度)；但是，修改的元素不必是连续的。因此，写掩码字段2570允许部分向量操作，包括加载、存储、算术、逻辑等。虽然描述了实施例，其中写掩码字段2570的内容选择多个写掩码寄存器中包含要使用的写掩码的一个写掩码寄存器(并且因此写掩码字段2570的内容间接地标识要执行的掩码)，但替代实施例或者附加实施例允许写掩码字段2570的内容直接指定要执行的掩码。

[0266] 立即数(immediate)字段2572-其内容允许指定立即数。该字段从以下意义上来说是可选的，例如它不存在于不支持立即数的通用向量友好格式的实现中并且它不存在于不使用立即数的指令中。

[0267] 类字段2568-其内容区分不同类的指令。参考图25A-B，该字段的内容在A类和B类指令之间进行选择。在图25A-B中，圆角方块用于表示在字段中存在特定值(例如，分别在图25A-B中的类字段2568的A类2568A和B类2568B)。

[0268] A类指令模板

[0269] 在A类的非存储器访问2505指令模板的情况下，α字段2552被解析为RS字段2552A，其内容区分要执行不同的增强操作类型中的哪个(例如，舍入2552A.1和数据变换2552A.2分别被指定用于无存储器访问、舍入型操作2510和无存储器访问、数据变换类型操作2515指令模板)，而β字段2554区分要执行哪个指定类型的操作。在无存储器访问2505指令模板中，不存在缩放字段2560，位移字段2562A和位移因子字段2562B。

[0270] 无存储器访问指令模板-全舍入控制类型操作

[0271] 在无存储器访问全舍入控制类型操作2510指令模板中，β字段2554被解析为舍入控制字段2554A，其内容提供静态舍入。虽然在本发明的所述实施例中，舍入控制字段2554A包括抑制所有浮点异常(SAE)字段2556和舍入操作控制字段2558，但是替代实施例可以支持可以将这些概念编码到同一字段中或仅具有这些概念/字段中的一个或者另一个(例如，可以仅具有舍入操作控制字段2558)。

[0272] SAE字段2556-其内容区分是否禁用异常事件报告；当SAE字段的2556内容表明启用抑制时，给定的指令不会报告任何类型的浮点异常标志，也不会引发任何浮点异常处理程序。

[0273] 舍入操作控制字段2558-其内容区分要执行一组舍入操作中的哪一个(例如，向上舍入，向下舍入，向零舍入和向最近舍入)。因此，舍入操作控制字段2558允许基于每个指令改变舍入模式。在本发明的处理器包括用于指定舍入模式的控制寄存器的一个实施例中，舍入操作控制字段2550的内容覆盖该寄存器值。

[0274] 无存储器访问指令模板-数据变换类型操作

[0275] 在无存储器访问数据变换类型操作2515指令模板中，β字段2554被解析为数据变换字段2554B，其内容区分要执行多个数据变换中的哪一个(例如，无数据变换、调配(swizzle)、广播)。

[0276] 在类A的存储器访问2520指令模板的情况下，α字段2552被解析为逐出提示字段2552B，其内容区分将使用哪一个驱逐提示(在图25A中，时态2552B.1和非时态2552B.2分别被指定用于存储器访问、时态2525指令模板和存储器访问、非时态2530指令模板)，而β字段2554被解析为数据操纵字段2554C，其内容区分要执行多个数据操纵操作(也称为基元)中

的哪一个(例如,无操纵;广播;源的上转换;以及目的地的下转换)。存储器访问2520指令模板包括缩放字段2560,并且可选地包括位移字段2562A或位移因子字段2562B。

[0277] 向量存储器指令利用转换支持执行从存储器的向量加载和到存储器的向量存储。与常规向量指令一样,向量存储器指令以逐个数据元素的方式从/向存储器传输数据,实际传输的元素由被选择作为写掩码的向量掩码的内容决定。

[0278] 存储器访问指令模板-时态

[0279] 时态数据是可能足够快地被重用以从缓存受益的数据。然而,这是一个提示,不同的处理器可以以不同的方式实现它,包括完全忽略提示。

[0280] 存储器访问指令模板-非时态

[0281] 非时态数据是不太可能足够快地被重用以从缓存在第一级缓存中受益的数据,并且应该优先驱逐。然而,这是一个提示,不同的处理器可以以不同的方式实现它,包括完全忽略提示。

[0282] B类指令模板

[0283] 在B类的指令模板的情况下,α字段2552被解析为写掩码控制(Z)字段2552C,其内容区分由写掩码字段2570控制的写掩码应该是合并还是归零。

[0284] 在B类的非存储器访问2505指令模板的情况下,β字段2554的一部分被解析为RL字段2557A,其内容区分要执行不同增强操作类型中的哪一个(例如,舍入2557A.1和向量长度(VSIZE)2557A.2分别被指定用于无存储器访问、写掩码控制、部分舍入控制类型操作2512指令模板和无存储器访问、写掩码控制、VSIZE类型操作2517指令模板),而β字段2554的其余部分区分要执行哪个指定类型的操作。在无存储器访问2505指令模板中,不存在缩放字段2560,位移字段2562A和位移因子字段2562B。

[0285] 在无存储器访问、写掩码控制、部分舍入控制类型操作2510指令模板中,β字段2554的其余部分被解析为舍入操作字段2559A并且禁用异常事件报告(给定指令不报告任何类型的浮点异常标志,不会引发任何浮点异常处理程序)。

[0286] 舍入操作控制字段2559A-正如舍入操作控制字段2558一样,其内容区分要执行的一组舍入操作中的哪一个(例如,向上舍入、向下舍入、向零舍入和向最近舍入)。因此,舍入操作控制字段2559A允许基于每个指令改变舍入模式。在本发明的处理器包括用于指定舍入模式的控制寄存器的一个实施例中,舍入操作控制字段2550的内容覆盖该寄存器值。

[0287] 在无存储器访问、写掩码控制、VSIZE类型操作2517指令模板中,β字段2554的其余部分被解析为向量长度字段2559B,其内容区分要在多个数据向量长度中的哪一个(例如,128、256或512字节)上执行。

[0288] 在B类的存储器访问2520指令模板的情况下,β字段2554的一部分被解析为广播字段2557B,其内容区分是否要执行广播类型数据操纵操作,而β字段2554的其余部分被解析为向量长度字段2559B。存储器访问2520指令模板包括缩放字段2560,并且可选地包括位移字段2562A或位移因子字段2562B。

[0289] 关于通用向量友好指令格式2500,示出了完整操作码字段2574,其包括格式字段2540,基本操作字段2542和数据元素宽度字段2564。尽管示出了完整操作码字段2574包括所有这些字段的一个实施例,但在不支持所有这些字段的实施例中,完整操作码字段2574包括少于所有这些字段。完整操作码字段2574提供操作代码(操作码)。

[0290] 增强操作字段2550、数据元素宽度字段2564和写掩码字段2570允许在通用向量友好指令格式中基于每个指令指定这些特征。

[0291] 写掩码字段和数据元素宽度字段的组合创建分类型的指令，因为它们允许基于不同的数据元素宽度来应用掩码。

[0292] 在A类和B类中找到的各种指令模板在不同情况下是有益的。在一些实施例中，处理器内的不同处理器或不同核可以仅支持A类，仅支持B类或支持这两类。例如，旨在用于通用计算的高性能通用乱序核可以仅支持B类，旨在主要用于图形和/或科学(吞吐量)计算的核可以仅支持A类，而旨在用于两者的核可以支持两者(当然，具有来自两类的模板和指令但不是来自两类的所有模板和指令的某种混合的核也在本发明的范围内)。此外，单个处理器可以包括多个核，所有核都支持相同的类或不同的核支持不同的类。例如，在具有单独图形和通用核的处理器中，旨在主要用于图形和/或科学计算的图形核之一可以仅支持A类，而一个或多个通用核可以是具有乱序执行和寄存器重命名的高性能通用核，旨在用于仅支持B类的通用计算。另一个不具有单独图形核的处理器可以包括支持A类和B类两者的一个或多个通用顺序或乱序核。当然，在不同实施例中，来自一类的特征也可以在另一类中实现。用高级语言编写的程序将被放置(例如，及时编译或静态编译)成各种不同的可执行形式，包括：1)仅具有用于执行的目标处理器所支持的类的指令的形式；或2)具有使用所有类的指令的不同组合编写的备选例程并具有控制流程代码的形式，该控制流程代码基于当前正在执行代码的处理器所支持的指令来选择要执行的例程。

[0293] 示例性特定向量友好指令格式

[0294] 图26A是示出根据实施例的示例性特定向量友好指令格式的框图。图26A示出了特定向量友好指令格式2600，该格式从以下意义上来说是特定的，例如该格式指定字段的位置、大小、解析和顺序、以及这些字段中的一些字段的值。特定向量友好指令格式2600可用于扩展x86指令集，因此一些字段与现有x86指令集及其扩展(例如AVX)中使用的字段类似或相同。此格式与具有扩展的现有x86指令集的前缀编码字段、真实操作码字节字段、MOD R/M字段、SIB字段、位移字段以及立即数字段保持一致。示出了26A中的字段所映射的图25中的字段。

[0295] 应当理解，尽管出于说明性目的，在通用向量友好指令格式2500的上下文中参考特定向量友好指令格式2600描述了实施例，但是本发明不限于特定向量友好指令格式2600，除非声明的情况下。例如，通用向量友好指令格式2500考虑了各种字段的各种可能大小，而特定向量友好指令格式2600被示为具有特定大小的字段。作为具体示例，虽然数据元素宽度字段2564被示为特定向量友好指令格式2600中的一位字段，但是本发明不限于此(即，通用向量友好指令格式2500考虑其他大小的数据元素宽度字段2564)。

[0296] 通用向量友好指令格式2500包括以下按照图26A中所示的顺序列出的以下字段。

[0297] EVEX前缀2602(字节0-3)以四字节形式编码。

[0298] 格式字段2540(EVEX字节0，位[7:0])-第一字节(EVEX字节0)是格式字段2540并且它包含0x62(在本发明的一个实施例中用于区分向量友好指令格式的唯一值)。

[0299] 第二至第四字节(EVEX字节1-3)包括提供特定能力的多个位字段。

[0300] REX字段2605(EVEX字节1，位[7-5])-由EVEX.R位字段(EVEX字节1，位[7]-R)、EVEX.X位字段(EVEX字节1，位[6]-X)和EVEX.X位字段(EVEX字节1，位[5]-B)组成。EVEX.R、

EVEX.X和EVEX.B位字段提供与相应VEX位字段相同的功能，并使用1的补码形式编码，即ZMM0被编码为1111B，ZMM15被编码为0000B。指令的其他字段对寄存器索引的低三位进行编码，如本领域中已知的(rrr、xxx和bbb)，因此可以通过添加EVEX.R、EVEX.X和EVEX.B来形成Rrrr、Xxxx和Bbbb。

[0301] REX'2510A-这是REX'字段2510的第一部分，并且是EVEX.R'位字段(EVEX字节1，位[4]-R')，其用于对扩展32寄存器组的高16或低16位进行编码。在本发明的一个实施例中，该位以及如下所示的其它位以位反转的格式来存储，以与BOUND指令进行区分(在众所周知的x86 32位模式中)，其真实操作码字节为62，但是在MOD R/M字段(下面描述)中不接受MOD字段中的值11；替代实施例不以反转格式存储这个和下面的其他所指示的位。值1用于对低16位寄存器编码。即，R'Rrrr是通过组合EVEX.R'、EVEX.R和来自其他字段的其他RRR形成的。

[0302] 操作码映射字段2615(EVEX字节1，位[3:0]-mmmm)-其内容对隐含的前导操作码字节(0F、0F 38或0F 3)进行编码。

[0303] 数据元素宽度字段2564(EVEX字节2，位[7]-W)-由符号EVEX.W表示。EVEX.W用于定义数据类型的粒度(大小)(32位数据元素或64位数据元素)。

[0304] EVEX.vvvv 2620(EVEX字节2，位[6:3]-vvvv)-EVEX.vvvv的作用可以包括以下内容：1)EVEX.vvvv对第一个源寄存器操作数进行编码，以反转(1的补码)形式指定，并且对有2个或更多源操作数的指令有效；2)EVEX.vvvv对目标寄存器操作数进行编码，以1的补码形式指定用于某些向量移位；或者3)EVEX.vvvv不对任何操作数进行编码，该字段是保留的，并且应该包含1111b。因此，EVEX.vvvv字段2620对以反转(1的补码)形式存储的第一源寄存器指定符的4个低阶位进行编码。取决于指令，使用额外不同的EVEX位字段将指定符大小扩展为32个寄存器。

[0305] EVEX.U 2568类字段(EVEX字节2，位[2]-U)-如果EVEX.U＝0，则表示A类或EVEX.U0；如果EVEX.U＝1，则表示B类或EVEX.U1。

[0306] 前缀编码字段2625(EVEX字节2，位[1:0]-pp)-为基本操作字段提供附加位。除了以EVEX前缀格式提供对传统SSE指令的支持之外，这还具有压缩SIMD前缀的益处(不是要求字节表示SIMD前缀，EVEX前缀仅需要2位)。在一个实施例中，为了支持在传统格式和EVEX前缀格式中使用SIMD前缀(66H，F2H，F3H)的传统SSE指令，将这些传统SIMD前缀编码到SIMD前缀编码字段中；并且在运行时在被提供给解码器的PLA之前扩展为传统SIMD前缀(因此PLA可以执行这些传统指令的传统和EVEX格式而无需修改)。虽然较新的指令可以直接使用EVEX前缀编码字段的内容作为操作码扩展，但是某些实施例以类似的方式扩展以保持一致性，但允许这些传统SIMD前缀指定不同的含义。替代实施例可以重新设计PLA以支持2位SIMD前缀编码，因此不需要扩展。

[0307] α字段2552(EVEX字节3，位[7]-EH；也称为EVEX.EH、EVEX.rs、EVEX.RL、EVEX.写掩码控制、以及EVEX.N；也用α示出)-如前所述，该字段是特定于上下文的。

[0308] β字段2554(EVEX字节3，位[6:4]-SSS，也称为EVEX.s_{2-0}、EVEX.r_{2-0}、EVEX.rr1、EVEX.LL0、EVEX.LLB；也用β示出)-如前所述，该字段是特定于上下文的。

[0309] REX'字段2510-这是REX'字段的剩余部分，并且是EVEX.V'位字段(EVEX字节3，位[3]-V')，其可用于对扩展32寄存器组的高16或低16位编码。该位以位反转格式存储。值1用

于对低16位寄存器编码。即，通过组合EVEX.V'、EVEX.vvvv来形成V'VVVV。

[0310] 写掩码字段2570(EVEX字节3，位[2:0]-kkk)-如前所述，其内容指定写掩码寄存器中的寄存器的索引。在本发明的一个实施例中，特定值EVEX.kkk＝000具有特殊行为，暗示没有写掩码用于特定指令(这可以以各种方式实现，包括使用硬连线到所有的写掩码或绕过掩蔽硬件的硬件)。

[0311] 真实操作码字段2630(字节4)也称为操作码字节。在该字段中指定部分操作码。

[0312] MOD R/M字段2640(字节5)包括MOD字段2642、Reg字段2644和R/M字段2646。如前所述，MOD字段2642的内容区分存储器访问和非存储器访问操作。Reg字段2644的作用可归纳为两种情况：对目的地寄存器操作数或源寄存器操作数编码，或被视为操作码扩展而不用于对任何指令操作数编码。R/M字段2646的作用可以包括以下内容：对引用存储器地址的指令操作数编码，或对目的地寄存器操作数或源寄存器操作数编码。

[0313] 缩放、索引、基准(SIB)字节(字节6)-如前所述，SIB 2650的内容用于存储器地址生成。SIB.xxx 2654和SIB.bbb 2656-这些字段的内容先前已经相关于寄存器索引Xxxx和Bbbb提及了。

[0314] 位移字段2562A(字节7-10)-当MOD字段2642包含10时，字节7-10是位移字段2562A，并且其工作方式与传统的32位位移(disp32)相同并且以字节粒度工作。

[0315] 位移因子字段2562B(字节7)-当MOD字段2642包含01时，字节7是位移因子字段2562B。该字段的位置与传统x86指令集8位位移(disp8)的位置相同，其以字节粒度工作。由于disp8是符号扩展的，它只能解决-128到127个字节之间的偏移量；就64字节缓存行而言，disp8使用8位，只能被设置为4个非常有用的值-128、-64、0和64；因为经常需要更大的范围，所以使用disp32；但是，disp32需要4个字节。与disp8和disp32相比，位移因子字段2562B是disp8的重新解析；当使用位移因子字段2562B时，实际位移由位移因子字段的内容乘以存储器操作数访问的大小(N)来确定。这种类型的位移称为disp8*N。这减少了平均指令长度(用于位移的单个字节，但具有更大的范围)。这种压缩位移假设有效位移是存储器访问的粒度的倍数，因此，不需要对地址偏移的冗余低阶位进行编码。即，位移因子字段2562B代替传统x86指令集8位位移。因此，位移因子字段2562B以与x86指令集8位位移相同的方式被编码(因此ModRM/SIB编码规则没有变化)，唯一的例外是disp8被过加载到disp8*N。即，编码规则或编码长度没有变化，而只是硬件对位移值的解析变化(需要用存储器操作数的大小来缩放位移以获得逐字节地址偏移)。立即数字段2572如前所述操作。

[0316] 完整操作码字段

[0317] 图26B是示出根据本发明一个实施例的构成完整操作码字段2574的特定向量友好指令格式2600的字段的框图。具体地，完整操作码字段2574包括格式字段2540、基本操作字段2542和数据元素宽度(W)字段2564。基本操作字段2542包括前缀编码字段2625、操作码映射字段2615和真实操作码字段2630。

[0318] 寄存器索引字段

[0319] 图26C是示出根据本发明一个实施例的构成寄存器索引字段2544的特定向量友好指令格式2600的字段的框图。具体地，寄存器索引字段2544包括REX字段2605、REX'字段2610、MODR/M.reg字段2644、MODR/M.r/m字段2646、VVVV字段2620、xxx字段2654和bbb字段2656。

[0320] 增强操作字段

[0321] 图26D是示出根据本发明一个实施例的构成增强操作字段2550的特定向量友好指令格式2600的字段的框图。

[0322] 当类(U)字段2568包含0时,它表示EVEX.U0(A类2568A);当类(U)字段2568包含1时,它表示EVEX.U1(B类2568B)。当U=0并且MOD字段2642包含11(表示无存储器访问操作)时,α字段2552(EVEX字节3,位[7]-EH)被解析为rs字段2552A。当rs字段2552A包含1(舍入2552A.1)时,β字段2554(EVEX字节3,位[6:4]-SSS)被解析为舍入控制字段2554A。舍入控制字段2554A包括一位SAE字段2556和两位舍入操作字段2558。当rs字段2552A包含0(数据变换2552A.2)时,β字段2554(EVEX字节3,位[6:4]-SSS)被解析为三位数据变换字段2554B。当U=0并且MOD字段2642包含00、01或10(表示存储器访问操作)时,α字段2552(EVEX字节3,位[7]-EH)被解析为逐出提示(EH)字段2552B,β字段2554(EVEX字节3,位[6:4]-SSS)被解析为三位数据操纵字段2554C。

[0323] 当U=1时,α字段2552(EVEX字节3,位[7]-EH)被解析为写掩码控制(Z)字段2552C。当U=1并且MOD字段2642包含11(表示无存储器访问操作)时,β字段2554的一部分(EVEX字节3,位[4]-S_0)被解析为RL字段2557A;当RL字段2557A包含1(舍入2557A.1)时,β字段2554的其余部分(EVEX字节3,位[6-5]-S_{2-1})被解析为舍入操作字段2559A,而当RL字段2557A包含0(VSIZE 2557.A2)时,β字段2554的其余部分(EVEX字节3,位[6-5]-S_{2-1})被解析为向量长度字段2559B(EVEX字节3,位[6-5]-L_{1-0})。当U=1并且MOD字段2642包含00、01或10(表示存储器访问操作)时,β字段2554(EVEX字节3,位[6:4]-SSS)被解析为向量长度字段2559B(EVEX字节3,位[6-5]-L_{1-0})和广播字段2557B(EVEX字节3,位[4]-B)。

[0324] 示例性寄存器架构

[0325] 图27是根据本发明一个实施例的寄存器架构2700的框图。在所示的实施例中,存在512位宽的32个向量寄存器2710;这些寄存器引用为zmm0到zmm31。低16个zmm寄存器的低阶256位覆盖在寄存器ymm0-16上。低16个zmm寄存器的低阶128位(ymm寄存器的低阶128位)覆盖在寄存器xmm0-15上。特定向量友好指令格式2600对这些覆盖的寄存器文件进行操作,如下表所示。

[0326]

可调节向量长度	类	操作	寄存器
不包括向量长度字段 2559B 的指令模板	A（图 25A；U=0）	2510, 2515, 2525, 2530	zmm 寄存器（向量长度为 64 字节）
	B (图 25B; U=1)	2512	zmm 寄存器（向量长度为 64 字节）
包括向量长度字段 2559B 的指令模板	B (图 25B; U=1)	2517, 2527	zmm, ymm, or xmm 寄存器（向量长度为 64 字节，32 字节或 16 字节），具体取决于向量长度字段 2559B

[0327] 即,向量长度字段2559B在最大长度和一个或多个其他较短长度之间进行选择,其

中每个这样的较短长度是前一长度的一半长度;没有向量长度字段2559B的指令模板对最大向量长度操作。此外,在一个实施例中,特定向量友好指令格式2600的B类指令模板对打包或标量单/双精度浮点数据和打包或标量整数数据进行操作。标量操作是对zmm/ymm/xmm寄存器中的最低阶数据元素位置执行的操作;根据实施例,高阶数据元素位置保持与指令之前相同或归零。

[0328] 写掩码寄存器2715-在所示实施例中,存在8个写掩码寄存器(k0到k7),各自大小为64位。在替换实施例中,写掩码寄存器2715的大小为16位。如前所述,在本发明的一个实施例中,向量掩码寄存器k0不能用作写掩码;当通常表示k0的编码用于写掩码时,它选择0xFFFF的硬连线写掩码,有效地禁用该指令的写掩码。

[0329] 通用寄存器2725-在所示实施例中,有16个64位通用寄存器,它们与现有的x86寻址模式一起用于寻址存储器操作数。这些寄存器由名称RAX,RBX,RCX,RDX,RBP,RSI,RDI,RSP和R8至R15引用。

[0330] 标量浮点堆栈寄存器文件(x87堆栈)2745,其上混叠有MMX打包整数平坦寄存器文件2750-在所示实施例中,x87堆栈是用于使用x87指令集扩展对32/64/80位浮点数据执行标量浮点运算的八元素堆栈;而MMX寄存器用于对64位打包整数数据执行操作,以及保持用于MMX和XMM寄存器之间执行的某些操作的操作数。

[0331] 替代实施例可以使用更宽或更窄的寄存器。另外,替代实施例可以使用更多、更少或不同的寄存器文件和寄存器。

[0332] 示例性核架构、处理器和计算机架构

[0333] 处理器核可以以不同的方式实现,可以被实现用于不同的目的,并且可以在不同的处理器中实现。例如,这种核的实现方式可以包括:1)用于通用计算的通用顺序核;2)用于通用计算的高性能通用乱序核;3)主要用于图形和/或科学(吞吐量)计算的专用核。不同处理器的实现方式可以包括:1)CPU,其包括旨在用于通用计算的一个或多个通用顺序核和/或用于通用计算的一个或多个通用乱序核;2)协处理器,包括主要用于图形和/或科学(吞吐量)的一个或多个专用核。这种不同的处理器导致不同的计算机系统架构,其可以包括:1)在与CPU不同的芯片上的协处理器;2)在与CPU相同的封装中的单独管芯(die)上的协处理器;3)在与CPU相同的管芯上的协处理器(在这种情况下,这种协处理器有时被称为专用逻辑(例如,集成图形和/或科学(吞吐量)逻辑)或被称为专用核);4)片上系统,其可以在同一管芯上包括所描述的CPU(有时被称为(一个或多个)应用核或(一个或多个)应用处理器)、以上描述的协处理器、和附加功能。接下来描述示例性核架构,之后描述示例性处理器和计算机架构。

[0334] 示例性核架构

[0335] 顺序和乱序的核框图

[0336] 图28A是示出根据实施例的示例性顺序流水线和示例性寄存器重命名、乱序发布/执行流水线两者的框图。图28B是示出根据实施例的要被包括在处理器中的顺序架构核和示例性寄存器重命名、乱序发布/执行架构核两者的示例性实施例的框图。图28A-B中的实线框示出了顺序流水线和顺序核,而可选择添加的虚线框示出了寄存器重命名、乱序发布/执行流水线和核。假定顺序方面是乱序方面的子集,将描述乱序方面。

[0337] 在图28A中,处理器流水线2800包括提取(fetch)阶段2802、长度解码阶段2804、解

码阶段2806、分配阶段2808、重命名阶段2810、调度(也被称为调派或发布)阶段2812、寄存器读取/存储器读取阶段2814、执行阶段2816、写回/存储器写入阶段2818、异常处理阶段2822、和提交阶段(commit stage)2824。

[0338] 图28B示出了处理器核2890,其包括耦合到执行引擎单元2850的前端单元2830,并且执行引擎单元2850和前端单元2830两者都耦合到存储器单元2870。核2890可以是精简指令集计算(RISC)核、复杂指令集计算(CISC)核、超长指令字(VLIW)核、或混合或替代的核类型。作为另一种选择,核2890可以是专用核,例如,网络或通信核、压缩引擎、协处理器核、通用计算图形处理单元(GPGPU)核、图形核等。

[0339] 前端单元2830包括耦合到指令缓存单元2834的分支预测单元2832,指令缓存单元2834耦合到指令转换后备缓冲器(TLB)2836,指令转换后备缓冲器2836耦合到指令提取单元2838,指令提取单元2838耦合到解码单元2840。解码单元2840(或解码器)可以解码指令,并且生成作为输出的一个或多个微操作、微代码入口点、微指令、其他指令、或其他控制信号,它们解码自原始指令或以其他方式反映原始指令或导出自原始指令。可以使用各种不同的机制来实现解码单元2840。合适机制的示例包括但不限于查找表、硬件实现方式、可编程逻辑阵列(PLA)、微代码只读存储器(ROM)等。在一个实施例中,核2890包括微代码ROM或存储用于某些宏指令的微代码的其他介质(例如,在解码单元2840中或在前端单元2830内)。解码单元2840耦合到执行引擎单元2850中的重命名/分配器单元2852。

[0340] 执行引擎单元2850包括重命名/分配器单元2852,其耦合到引退(retirement)单元2854和一组一个或多个调度器单元2856。(一个或多个)调度器单元2856表示任意数目的不同调度器,包括,预留站(reservations station)、中央指令窗等。(一个或多个)调度器单元2856耦合到(一个或多个)物理寄存器文件单元2858。每个物理寄存器文件单元2858表示一个或多个物理寄存器文件,这些物理寄存器文件中的不同的物理寄存器文件存储一个或多个不同的数据类型,例如,标量整数、标量浮点、打包整数、打包浮点、向量整数、向量浮点、状态(例如,作为要执行的下一指令的地址的指令指针)等。在一个实施例中,物理寄存器文件单元2858包括向量寄存器单元、写掩码寄存器单元、和标量寄存器单元。这些寄存器单元可以提供架构向量寄存器、向量掩码寄存器、和通用寄存器。(一个或多个)物理寄存器文件单元2858与引退单元2854重叠,以说明寄存器重命名和乱序执行可以被实现的各种方式(例如,使用(一个或多个)重新排序缓冲器和(一个或多个)引退寄存器文件;使用(一个或多个)未来文件、(一个或多个)历史缓冲器、和(一个或多个)引退寄存器文件;使用寄存器图和寄存器池;等等)。引退单元2854和(一个或多个)物理寄存器文件单元2858耦合到(一个或多个)执行集群2860。(一个或多个)执行集群2860包括一组一个或多个执行单元2862和一组一个或多个存储器访问单元2864。执行单元2862可以对各种类型的数据(例如,标量浮点、打包整数、打包浮点、向量整数、向量浮点)执行各种操作(例如,移位、加法、减法、乘法)。虽然一些实施例可以包括专用于特定功能或功能集的多个执行单元,但是其他实施例可以仅包括一个执行单元或者全部执行所有功能的多个执行单元。(一个或多个)调度器单元2856、(一个或多个)物理寄存器文件单元2858、和(一个或多个)执行集群2860被示为可能是多个,因为某些实施例针对某些类型的数据/操作创建单独的流水线(例如,标量整数流水线、标量浮点/打包整数/打包浮点/向量整数/向量浮点流水线、和/或存储器访问流水线,其中每个流水线都有自己的调度器单元、物理寄存器文件单元、和/或执行集群-

并且在单独的存储器访问流水线的情况下,其中仅该流水线的执行集群具有(一个或多个)存储器访问单元2864的某些实施例被实现)。还应理解,在使用单独的流水线的情况下,这些流水线中的一个或多个可以是乱序发布/执行而其余的是顺序发布/执行的。

[0341] 该组存储器访问单元2864耦合到存储器单元2870,存储器单元2870包括耦合到数据缓存单元2874的数据TLB单元2872,其中数据缓存单元2874耦合到2级(L2)缓存单元2876。在一个示例性实施例中,存储器访问单元2864可以包括加载单元、存储地址单元、和存储数据单元,其中的每个单元耦合到存储器单元2870中的数据TLB单元2872。指令缓存单元2834还耦合到存储器单元2870中的2级(L2)缓存单元2876。L2缓存单元2876耦合到一个或多个其他级别的缓存并最终耦合到主存储器。

[0342] 作为示例,示例性寄存器重命名的乱序发布/执行核架构可以按如下方式实现流水线2800:1)指令提取2838执行提取和长度解码阶段2802和2804;2)解码单元2840执行解码阶段2806;3)重命名/分配器单元2852执行分配阶段2808和重命名阶段2810;4)(一个或多个)调度器单元2856执行调度阶段2812;5)(一个或多个)物理寄存器文件单元2858和存储器单元2870执行寄存器读取/存储器读取阶段2814;执行集群2860执行执行阶段2816;6)存储器单元2870和(一个或多个)物理寄存器文件单元2858执行写回/存储器写入阶段2818;7)异常处理阶段2822中可能涉及各个单元;8)引退单元2854和(一个或多个)物理寄存器文件单元2858执行提交阶段2824。

[0343] 核2890可以支持一个或多个指令集(例如,x86指令集(具有已经添加有较新版本的一些扩展);美国加利福尼亚州桑尼维尔市的MIP Technologies的MIPS指令集;美国加利福尼亚州桑尼维尔市的ARM Holdings的ARM指令集(具有可选的附加扩展,例如,NEON)),包括本文所描述的(一个或多个)指令。在一个实施例中,核2890包括支持打包数据指令集扩展(例如,AVX1、AVX2)的逻辑,从而允许要使用打包数据来执行的许多多媒体应用所使用的操作。

[0344] 应理解,核可以支持多线程(执行两个或更多个并行的操作集或线程集),并且可以以各种方式这样做,这些方式包括时间分片多线程、同时多线程(其中,单个物理核为该物理核正在同时进行多线程的每个线程提供逻辑核)、或它们的组合(例如,时间分片的提取和解码以及此后同时的多线程,例如,在Intel®超线程技术中)。

[0345] 虽然在乱序执行的上下文中描述了寄存器重命名,但应理解,寄存器重命名可以用在顺序架构中。虽然所示处理器的实施例还包括单独的指令和数据缓存单元2834/2874以及共享的L2缓存单元2876,但替代实施例可以具有用于指令和数据两者的单个内部缓存,例如,1级(L1)内部缓存、或多级内部缓存。在一些实施例中,系统可以包括内部缓存和外部缓存的组合,其中外部缓存在核和/或处理器外部。替代地,全部缓存可以在核和/或处理器外部。

[0346] 具体示例性顺序核架构

[0347] 图29A-B示出了更具体的示例性顺序核架构的框图,其中核将是芯片中的若干逻辑块(可能包括相同类型和/或不同类型的其他核)中的一个逻辑块。逻辑块通过高带宽互连网络(例如,环形网络)与某固定功能逻辑、存储器I/O接口、和其他必要的I/O逻辑通信,这取决于应用。

[0348] 图29A是根据实施例的单个处理器核以及其与管芯上互连网络2902的连接以及其

在2级(L2)缓存2904的本地子集的框图。在一个实施例中,指令解码器2900支持具有打包数据指令集扩展的x86指令集。L1缓存2906允许低等待时间的访问以将存储器缓存到标量和向量单元中。虽然在一个实施例中(为了简化设计),标量单元2908和向量单元2910使用单独的寄存器组(分别为标量寄存器2912和向量寄存器2914),并且它们之间传输的数据被写入到存储器然后从1级(L1)缓存2906中读回,但是替代实施例可以使用不同的方法(例如,使用单个寄存器组或包括允许数据在两个寄存器文件(file)之间传输而不被写入和读回的通信路径)。

[0349] L2缓存的本地子集2904是全局L2缓存的一部分,全局L2缓存被划分为分开的本地子集,每个处理器核一个本地子集。每个处理器核具有到其自己的L2缓存的本地子集2904的直接访问路径。由处理器核读取的数据被存储在其L2缓存子集2904中并且可以与访问它们自己的本地L2缓存子集的其他处理器核并行地被快速访问。由处理器核写入的数据被存储在其自己的L2缓存子集2904中,并且在需要的情况下被从其他子集冲刷(flushed)。环形网络确保共享数据的一致性。环形网络是双向的,以允许诸如处理器核、L2缓存、和其他逻辑块之类的代理在芯片内彼此通信。每个环形数据路径在每个方向上为1012位宽。

[0350] 图29B是根据实施例的图29A中的处理器核的一部分的展开图。图29B包括L1缓存2906的L1数据缓存2906A部分,以及关于向量单元2910和向量寄存器2914的更多细节。具体地,向量单元2910是16宽的向量处理单元(VPU)(参见16宽的ALU 2928),它执行整数、单精度浮点、和双精度浮点指令中的一个或多个。VPU支持通过调配单元2920对寄存器输入进行调配,使用数字转换单元2922A-B进行数字转换,以及使用复制单元2924对存储器输入进行复制。写掩码寄存器2926允许预测得到向量写入。

[0351] 图30是根据实施例的可具有不止一个核、可具有集成存储器控制器、且可具有集成图形的处理器3000的框图。图30中的实线框示出了具有单核3002A、系统代理3010、和一组一个或多个总线控制器单元3016的处理器3000;但虚线框的可选添加示出了具有以下各项的替代处理器3000:多个核3002A-N、系统代理单元3010中的一组一个或多个集成存储器控制器单元3014、以及专用逻辑3008。

[0352] 因此,处理器3000的不同实现方式可以包括:1)具有专用逻辑3008的CPU(其中专用逻辑是集成图形和/或科学(吞吐量)逻辑(其可以包括一个或多个核)),以及核3002A-N(其是一个或多个通用核(例如,通用顺序核、通用乱序核、或两者的组合);2)具有核3002A-N的协处理器(其中核3002A-N是主要用于图形和/或科学(吞吐量)的大量专用核);3)具有核3002A-N的协处理器(其中核3002A-N是大量通用顺序核)。因此,处理器3000可以是通用处理器、协处理器、或专用处理器,例如,网络或通信处理器、压缩引擎、图形处理器、GPGPU(通用图形处理单元)、高吞吐量的许多集成核(MIC)协处理器(包括30个或更多个核)、嵌入式处理器等等。处理器可以在一个或多个芯片上实现。处理器3000可以是一个或多个衬底的一部分和/或可以通过使用多种工艺技术(例如,BiCMOS、CMOS或NMOS)中的任何一种来在一个或多个衬底上实现。

[0353] 存储器层级包括核内的一个或多个级别的缓存、一组或一个或多个共享缓存单元3006、以及耦合到该组集成存储器控制器单元3014的外部存储器(未示出)。该组共享缓存单元3006可以包括一个或多个中级缓存(例如,2级(L2)、3级(L3)、4级(L4)),或其他级别的缓存、最后级别缓存(LLC)、和/它们的组合。虽然在一个实施例中,基于环的互连单元3012

对专用逻辑3008(集成图形逻辑是专用逻辑的示例并且在本文中也被称为专用逻辑)、该组共享缓存单元3006、以及系统代理单元3010/(一个或多个)集成存储器控制器单元3014进行互连,但替代实施例可以使用任何数目的众所周知的技术来互连这些单元。在一个实施例中,在一个或多个缓存单元3006和核3002A-N之间维持一致性。

[0354] 在一些实施例中,核3002A-N中的一个或多个核能够进行多线程。系统代理3010包括协调和操作核3002A-N的那些组件。系统代理单元3010可以包括例如电源控制单元(PCU)和显示单元。PCU可以是或可以包括调节核3002A-N和专用逻辑3008的功率状态所需的逻辑和组件。显示单元用于驱动一个或多个外部连接的显示器。

[0355] 核3002A-N在架构指令集方面可以是同构的或异构的;也就是说,核3002A-N中的两个或更多个核可能能够执行相同的指令集,而其他核可能能够执行仅该指令集的子集或不同的指令集。

[0356] 示例性计算机架构

[0357] 图31-34是示例性计算机架构的框图。用于膝上型计算机、台式计算机、手持式PC、个人数字助理、工程工作站、服务器、网络设备、网络集线器、交换机、嵌入式处理器、数字信号处理器(DSP)、图形设备、视频游戏设备、机顶盒、微控制器、蜂窝电话、便携式媒体播放器、手持设备、和各种其他电子设备的本领域已知的其他系统设计和配置也是适合的。通常,能够结合本文所公开的处理器和/或其他执行逻辑的各种各样的系统或电子设备通常是合适的。

[0358] 现在参考图31,示出了根据本发明的一个实施例的系统3100的框图。系统3100可以包括一个或多个处理器3110、3115,其耦合到控制器集线器3120。在一个实施例中,控制器集线器3120包括图形存储器控制器集线器(GMCH)3190和输入/输出集线器(IOH)3150(可以在分开的芯片上);GMCH 3190包括耦合到存储器3140和协处理器3145的存储器和图形控制器;IOH 3150将输入/输出(I/O)设备3160耦合到GMCH 3190。替代地,存储器和图形控制器中的一个或两个被集成在处理器内(如本文所描述的),存储器3140和协处理器3145直接耦合到处理器3110,以及包括IOH 3150的单个芯片中的控制器集线器3120。

[0359] 图31中用虚线表示附加处理器3115的可选性质。每个处理器3110、3115可以包括本文所描述的处理核中的一个或多个,并且可以是处理器3000的某个版本。

[0360] 存储器3140可以是例如动态随机存取存储器(DRAM)、相变存储器(PCM)、或这两者的组合。对于至少一个实施例,控制器集线器3120经由多点总线(multi-drop bus)与(一个或多个)处理器3110、3115通信,该多点总线例如是前端总线(FSB)、诸如QuickPath互连(QPI)之类的点对点接口、或类似的连接3195。

[0361] 在一个实施例中,协处理器3145是专用处理器,例如,高吞吐量MIC处理器、网络或通信处理器、压缩引擎、图形处理器、GPGPU、嵌入式处理器等。在一个实施例中,控制器集线器3120可以包括集成图形加速器。

[0362] 在物理资源3110、3115之间在包括架构特性、微架构特性、热特性、功耗特性等的指标度量的范围方面可存在各种差异。

[0363] 在一个实施例中,处理器3110执行控制一般类型的数据处理操作的指令。嵌入在指令内的可以是协处理器指令。处理器3110将这些协处理器指令识别为应该由所附接的协处理器3145执行的类型。因此,处理器3110将这些协处理器指令(或表示协处理器指令的控

制信号)发布到协处理器总线或其它互连上,以到协处理器3145。(一个或多个)协处理器3145接受并执行所接收的协处理器指令。

[0364] 现在参考图32,示出了根据本发明的实施例的第一更具体的示例性系统3200的框图。如图32所示,多处理器系统3200是点对点互连系统,并且包括经由点对点互连3250耦合的第一处理器3270和第二处理器3280。处理器3270和3280中的每一个处理器可以是处理器3000的某个版本。在本发明的一个实施例中,处理器3270和3280分别是处理器3110和3115,而协处理器3238是协处理器3145。在另一个实施例中,处理器3270和3280分别是处理器3110和协处理器3145。

[0365] 处理器3270和3280被示出为分别包括集成存储器控制器(IMC)单元3272和3282。处理器3270还包括作为其总线控制器单元的一部分的点对点(P-P)接口3276和3278;类似地,第二处理器3280包括P-P接口3286和3288。处理器3270、3280可以使用P-P接口电路3278、3288经由点对点(P-P)接口3250来交换信息。如图32所示,IMC3272和3282将处理器耦合到相应的存储器(即,存储器3232和存储器3234),这些存储器可以是本地附接到相应处理器的主存储器的一部分。

[0366] 处理器3270、3280可以各自使用点对点接口电路3276、3294、3286、3298经由各个P-P接口3252、3254来与芯片集3290交换信息。芯片集3290可以可选地经由高性能接口3292来与协处理器3238交换信息。在一个实施例中,协处理器3238是专用处理器,例如,高吞吐量MIC处理器、网络或通信处理器、压缩和/或解压缩引擎、图形处理器、GPGPU、嵌入式处理器等。

[0367] 共享缓存(未示出)可以被包括在任一处理器中,或者在两个处理器外部但经由P-P互连与处理器连接,使得在处理器进入低功率模式的情况下,任一或两个处理器的本地缓存信息可以被存储在共享缓存中。

[0368] 芯片集3290可以经由接口3296耦合到第一总线3216。在一个实施例中,第一总线3216可以是外围组件互连(PCI)总线,或诸如PCI Express总线或另一第三代I/O互连总线之类的总线,但本公开的范围不限于此。

[0369] 如图32所示,各种I/O设备3214可以耦合到第一总线3216,以及将第一总线3216耦合到第二总线3220的总线桥3218。在一个实施例中,一个或多个附加处理器3215(例如,协处理器、高吞吐量MIC处理器、GPGPU、加速器(例如,图形加速器或数字信号处理(DSP)单元)、现场可编程门阵列、或任何其他处理器)耦合到第一总线3216。在一个实施例中,第二总线3220可以是低引脚数(LPC)总线。在一个实施例中,各种设备可以耦合到第二总线3220,包括,例如键盘和/或鼠标3222、通信设备3227、和诸如磁盘驱动器或其他大容量存储设备之类的储存装置3228(其可以包括指令/代码和数据3230)。此外,音频I/O 3224可以耦合到第二总线3220。注意,可能有其他架构。例如,代替图32的点对点架构,系统可以实现多点(multi-drop)总线或其他这样的架构。

[0370] 现在参考图33,示出了根据本发明的实施例的第二更具体的示例性系统3300的框图。图32和33中的相似的元件具有相似的附图标记,并且图32中的某些方面已从图33中省略,以避免模糊图33的其他方面。

[0371] 图33示出了处理器3270、3280可以分别包括集成存储器和I/O控制逻辑(“CL”)3372和3382。因此,CL 3372、3382包括集成存储器控制器单元并包括I/O控制逻辑。图33示

出了不仅存储器3232、3234耦合到CL 3372、3382，而且I/O设备3314也耦合到控制逻辑3372、3382。传统(legacy)I/O设备3315耦合到芯片集3290。

[0372] 现在参考图34，示出了根据本发明的实施例的SoC 3400的框图。图30中的相似的元件具有相似的附图标记。此外，虚线框是更高级SoC上的可选功能。在图34中，(一个或多个)互连单元3402耦合到以下各项：应用处理器3410，其包括一组一个或多个核3002A-N(包括缓存单元3004A-N)和(一个或多个)共享缓存单元3006；系统代理单元3010；(一个或多个)总线控制器单元3016；(一个或多个)集成存储器控制器单元3014；一组或一个或多个协处理器3420，其可以包括集成图形逻辑、图像处理器、音频处理器和视频处理器；静态随机存取存储器(SRAM)单元3430；直接存储器访问(DMA)单元3432；以及显示单元3440，用于耦合到一个或多个外部显示器。在一个实施例中，(一个或多个)协处理器3420包括专用处理器，例如，网络或通信处理器、压缩引擎、GPGPU、高吞吐量MIC处理器、嵌入式处理器等。

[0373] 本文公开的机制的实施例可以以硬件、软件、固件或这些实现方法的组合来实现。本发明的实施例可以实现为在可编程系统上执行的计算机程序或程序代码，该可编程系统包括至少一个处理器，储存系统(包括易失性和非易失性存储器和/或储存元件)，至少一个输入设备及至少一个输出设备。

[0374] 程序代码(例如图32中所示的代码3230)可以应用于输入指令以执行本文描述的功能并生成输出信息。输出信息可以以已知的方式应用于一个或多个输出设备。对于本申请，处理系统包括具有处理器的任何系统，例如，数字信号处理器(DSP)、微控制器、专用集成电路(ASIC)或微处理器。

[0375] 程序代码可以以高级过程或面向对象的编程语言实现，以与处理系统通信。如果需要，程序代码也可以用汇编语言或机器语言实现。实际上，本文描述的机制不限于任何特定编程语言的范围。在任何情况下，该语言可以是经编译或解析的语言。

[0376] 至少一个实施例的一个或多个方面可以通过存储在机器可读介质上的代表性指令来实现，该代表性指令表示处理器内的各种逻辑，当由机器读取时使得机器构造逻辑以执行本文所描述的技术。这种称为“IP核”的表示可以存储在有形机器可读介质上，并提供给各种客户或制造设施，以加载到实际制造逻辑或处理器的制造机器中。

[0377] 这样的机器可读储存介质可以包括但不限于由机器或设备制造或形成的物品的非暂时性有形布置，包括诸如硬盘之类的储存介质，任何其他类型的盘，包括软盘、光盘、光盘只读存储器(CD-ROM)、光盘可擦写(CD-RW)和磁光盘、半导体设备(如只读存储器(ROM))、随机存取存储器(RAM)(如动态随机存取存储器(DRAM)、静态随机存取存储器(SRAM))、可擦除可编程只读存储器(EPROM)、闪存、电可擦除可编程只读存储器(EEPROM)、相变存储器(PCM)、磁卡或光学卡或适用于存储电子指令的任何其他类型的介质。

[0378] 因此，实施例还包括非暂时性实体机器可读介质，其包含指令或包含设计数据，例如硬件描述语言(HDL)，其定义本文描述的结构、电路、装置、处理器和/或系统特征。这些实施例也可以被称为程序产品。

[0379] 仿真(包括二进制转换，代码变形等)

[0380] 在一些情况下，指令转换器可用于将指令从源指令集转换为目标指令集。例如，指令转换器可以转换(例如，使用静态二进制转换，包括动态编译的动态二进制转换)、变形、仿真或以其他方式将指令转换为要由核处理的一个或多个其他指令。指令转换器可以用软

件、硬件、固件或其组合来实现。指令转换器可以在处理器上、处理器外或者部分在处理器上、部分在处理器外。

[0381] 图35是根据实施例的对照使用软件指令转换器将源指令集中的二进制指令转换为目标指令集中的二进制指令的框图。在所示实施例中，指令转换器是软件指令转换器，但替代地，指令转换器可以用软件、固件、硬件、或其各种组合来实现。图35示出了采用高级语言3502的程序可以使用x86编译器3504来编译以生成x86二进制代码3506，其可以由具有至少一个x86指令集核的处理器3516本地执行。具有至少一个x86指令集核的处理器3516表示可以通过进行以下操作来执行与具有至少一个x86指令集核的Intel处理器基本上相同的功能，从而实现与具有至少一个x86指令集核的Intel处理器基本上相同的结果的任何处理器：兼容地执行或以其他方式处理(1)Intel x86指令集核的指令集的大部分或者(2)目标为在具有至少一个x86指令集核的Intel处理器上运行的应用或其他软件的目标代码版本。x86编译器3504表示可操作以生成x86二进制代码3506(例如，目标代码)的编译器，其中二进制代码可以在具有或不具有附加链接处理的情况下在具有至少一个x86指令集核的处理器3516上被执行。类似地，图35示出了采用高级语言3502的程序可以使用替代指令集编译器3508来编译以生成替代指令集二进制代码3510，该二进制代码可以由没有至少一个x86指令集核的处理器3514(例如，具有执行美国加利福尼亚州桑尼维尔市的MIPS Technologies的MIPS指令集和/或执行美国加利福尼亚州桑尼维尔市的ARM Holdings的ARM指令集的核的处理器)本地执行。指令转换器3512用于将x86二进制代码3506转换为可由不具有x86指令集核的处理器3514本地执行的代码。该转换后的代码不太可能与替代指令集二进制代码3510相同，因为很难制造出能够实现它的指令转换器；但是，转换后的代码将完成一般操作，并由来自替代指令集的指令组成。因此，指令转换器3512表示通过仿真、模拟、或任何其他过程来允许不具有x86指令集处理器或核的处理器或其他电子设备执行x86二进制代码3506的软件、固件、硬件、或其组合。

[0382] 另外的示例

[0383] 示例1提供一种示例性处理器，所述示例性处理器包括：提取电路，所述提取电路用于提取指令，该指令的字段指定操作码和以下各项的位置：结构数组(AOS)矩阵，所述AOS矩阵包含N个交错结构，每个交错结构由具有不同类型的K个元素组成，连续结构中的相同类型的元素以步幅隔开；以及一个或多个数组结构(SOA)矩阵，所述一个或多个SOA矩阵一起包含K组元素，每个组包含N个相同类型的元素，其中，所述K个组中的每个组均设置在专用矩阵中或者被分组在共享矩阵的一部分中；解码电路，所述解码电路用于对所提取的指令进行解码；以及执行电路，响应于经解码的指令，所述执行电路用于：当所述操作码指定从AOS到SOA的转换时，将所指定的AOS矩阵的每个元素分类为K种元素类型中的一种，并且将每个经分类的元素写入到所述一个或多个SOA矩阵中的具有相同类型元素的该经分类的元素的对应组中；以及当所述操作码指定从SOA到AOS的转换时，从所述SOA矩阵的K个组中的每个组一次存储一个元素，使每个存储的元素与具有不同类型的元素交错，并且使每个相同类型的元素以步幅隔开。

[0384] 示例2包括根据示例1所述的示例性处理器的实质，其中，所述一个或多个SOA矩阵包括K个矩阵，每个矩阵用于排他地存储一种类型的元素。

[0385] 示例3包括根据示例1所述的示例性处理器的实质，其中，所述执行电路用于对在

所指定的AOS矩阵与所指定的SOA矩阵之间移动的数据的数据格式进行变换。

[0386] 示例4包括根据示例1所述的示例性处理器的实质，其中，所述指令进一步指定所指定的AOS矩阵和所指定的SOA矩阵的元素的元素大小，所述元素大小包括微字节、半字节、字节、字、双字和四字中的一种。

[0387] 示例5包括根据示例1所述的示例性处理器的实质，其中，所述指令进一步指定步幅，所指定的AOS矩阵中的元素组中的相同类型的元素以所述步幅隔开。

[0388] 示例6包括根据示例5所述的示例性处理器的实质，其中，所述指令进一步指定元素组被隔开的辅助步幅。

[0389] 示例7包括根据示例1所述的示例性处理器的实质，其中，所述一个或多个SOA矩阵包括单个矩阵，所有K组元素被设置在所述单个矩阵中，但是相同类型的元素被分组在一起并且与其他元素类型的组分离。

[0390] 示例8包括根据示例7所述的示例性处理器的实质，其中，所述K组元素中的每个组均设置在所述SOA矩阵的一个或多个行中或者在所述SOA矩阵的一个或多个列中。

[0391] 示例9提供一种示例性方法，所述示例性方法包括：使用提取电路来提取指令，所述指令指定操作码和以下各项的位置：结构数组(AOS)矩阵，所述AOS矩阵包含N个交错结构，每个交错结构由具有不同类型的K个元素组成，连续结构中的相同类型的元素以步幅隔开；以及一个或多个数组结构(SOA)矩阵，所述一个或多个SOA矩阵一起包含K组元素，每个组包含N个相同类型的元素；使用解码电路来对所提取的指令进行解码；以及通过使用执行电路进行以下操作来对经解码的指令做出响应：当所述操作码指定从AOS到SOA的转换时，将所指定的AOS矩阵的每个元素分类为K种元素类型中的一种，并且将每个经分类的元素写入到所述一个或多个SOA矩阵中的具有相同类型元素的该经分类的元素的对应组中；以及当所述操作码指定从SOA到AOS的转换时，从所述SOA矩阵的K个组中的每个组一次存储一个元素，使每个存储的元素与具有不同类型的元素交错，并且使每个相同类型的元素以所述步幅隔开。

[0392] 示例10包括根据示例9所述的示例性方法的实质，其中，所述一个或多个SOA矩阵包括K个矩阵，每个矩阵用于排他地存储一种类型的元素。

[0393] 示例11包括根据示例9所述的示例性方法的实质，其中，所述执行电路用于对在所指定的AOS矩阵与所指定的SOA矩阵之间移动的数据的数据格式进行变换。

[0394] 示例12包括根据示例9所述的示例性方法的实质，其中，所述指令进一步指定所指定的AOS矩阵和所指定的SOA矩阵的元素的元素大小，所述元素大小包括微字节、半字节、字节、字、双字和四字中的一种。

[0395] 示例13包括根据示例9所述的示例性方法的实质，其中，所述指令进一步指定所指定的AOS矩阵中的元素组中的相同类型的元素被隔开的步幅。

[0396] 示例14包括根据示例13所述的示例性方法的实质，其中，所述指令进一步指定元素组被隔开的辅助步幅。

[0397] 示例15包括根据示例9所述的示例性方法的实质，其中，所述一个或多个SOA矩阵包括单个矩阵，其中设置了元素的所有K个组，但是同时相同类型的元素被分组在一起并且与其他元素类型的组分离。

[0398] 示例16包括根据示例15所述的示例性方法的实质，其中，所述K组元素中的每个组

设置在所述SOA矩阵的一个或多个行中或者在所述SOA矩阵的一个或多个列中。

[0399] 示例17提供一种示例性系统，所述示例性系统包括存储器和处理器，所述处理器包括：提取电路，所述提取电路用于提取指令，该指令的字段指定操作码和以下各项的位置：结构数组(AOS)矩阵，所述AOS矩阵包含N个交错结构，每个结构由具有不同类型的K个元素组成，连续结构中的相同类型的元素以步幅隔开；以及一个或多个数组结构(SOA)矩阵，所述一个或多个SOA矩阵一起包含K组元素，每个组包含N个相同类型的元素，其中，所述K个组中的每一个均设置在专用矩阵中或者被分组在共享矩阵的一部分中；解码电路，所述解码电路用于对所提取的指令进行解码；以及执行电路，响应于经解码的指令，所述执行电路用于：当所述操作码指定从AOS到SOA的转换时，将所指定的AOS矩阵的每个元素分类为K种元素类型中的一种，并且将每个经分类的元素写入到所述一个或多个SOA矩阵中的具有相同类型元素的该经分类的元素的对应组中；以及当所述操作码指定从SOA到AOS的转换时，从所述SOA矩阵的K个组中的每个组一次存储一个元素，使每个存储的元素与具有不同类型的元素交错，并且使每个相同类型的元素以所述步幅隔开。

[0400] 示例18包括根据示例17所述的示例性系统的实质，其中，所述一个或多个SOA矩阵包括K个矩阵，每个矩阵用于排他地存储一种类型的元素。

[0401] 示例19包括根据示例17所述的示例性系统的实质，其中，所述执行电路用于对在所指定的AOS矩阵与所指定的SOA矩阵之间移动的数据的数据格式进行变换。

[0402] 示例20包括根据示例17所述的示例性系统的实质，其中，所述指令进一步指定所指定的AOS矩阵和所指定的SOA矩阵的元素的元素大小，所述元素大小包括微字节、半字节、字节、字、双字和四字中的一种。

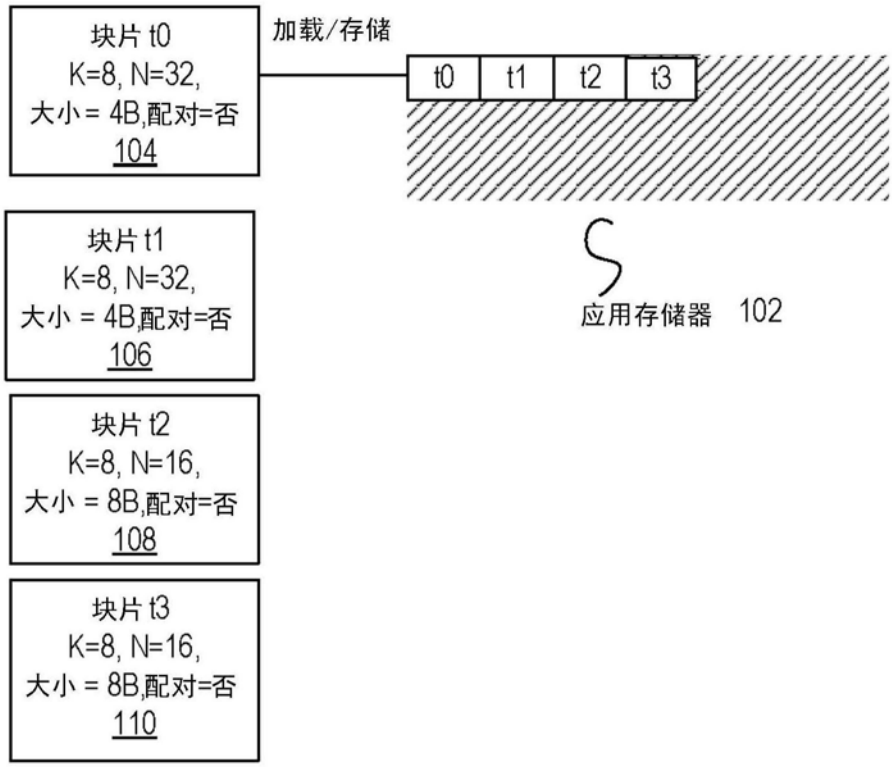

块片 t0
K=8, N=32,
大小 = 4B,配对=否
104
加载/存储
t0
t1
t2
t3
块片 t1
K=8, N=32,
大小 = 4B,配对=否
106
应用存储器 102
块片 t2
K=8, N=16,
大小 = 8B,配对=否
108
块片 t3
K=8, N=16,
大小 = 8B,配对=否
110

图1A

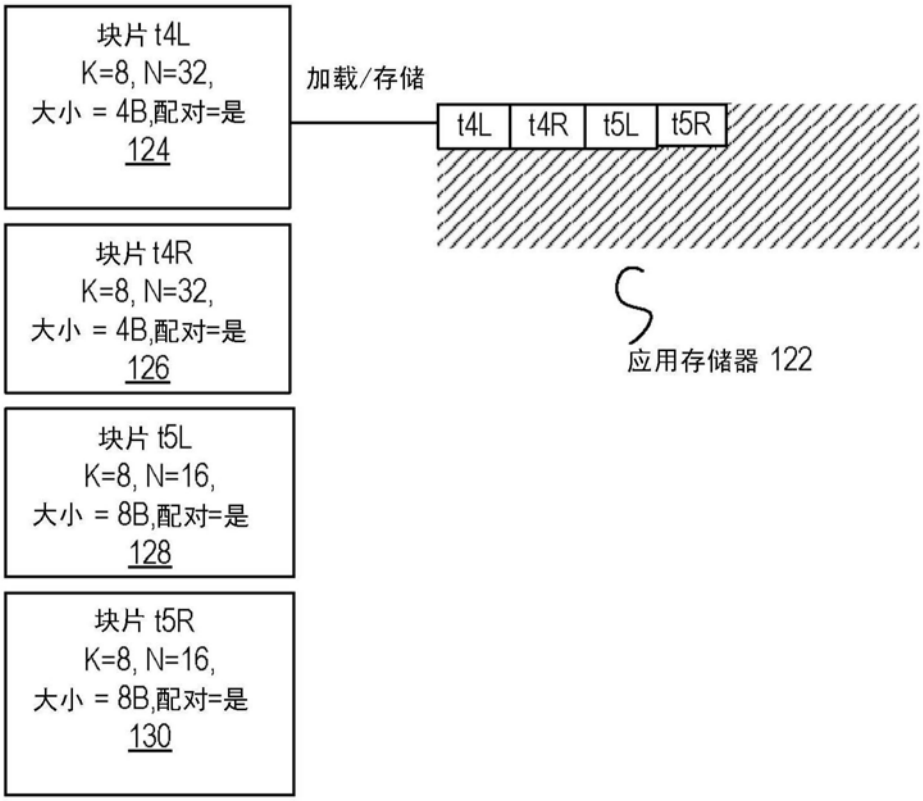

块片 t4L
K=8, N=32,
大小 = 4B,配对=是
124
加载/存储
t4L
t4R
t5L
t5R
块片 t4R
K=8, N=32,
大小 = 4B,配对=是
126
应用存储器 122
块片 t5L
K=8, N=16,
大小 = 8B,配对=是
128
块片 t5R
K=8, N=16,
大小 = 8B,配对=是
130

图1B

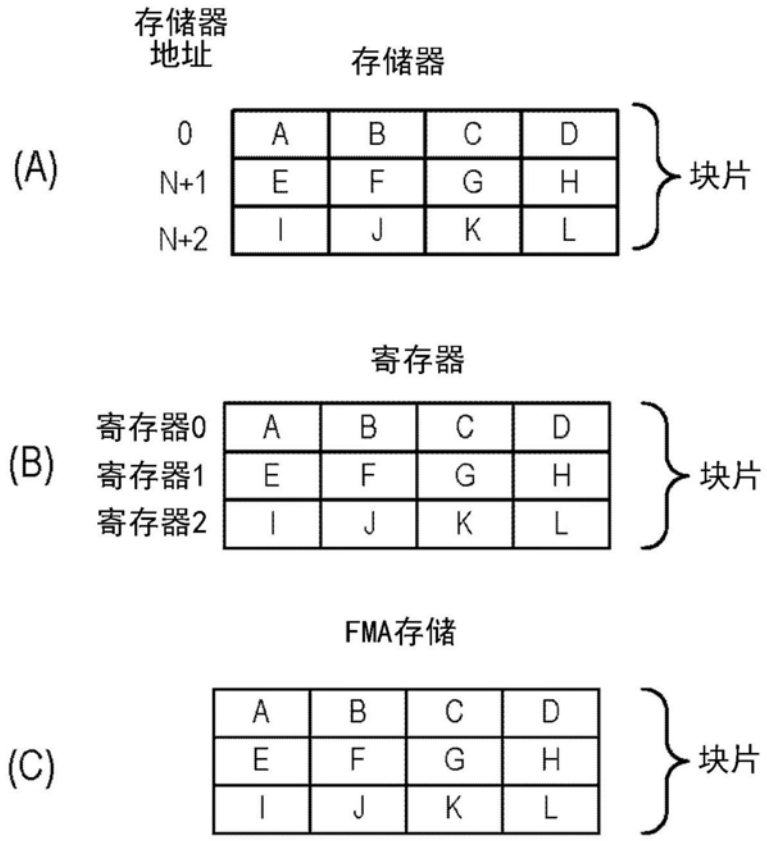

存储器
地址
存储器
(A)
0
N+1
N+2
A
B
C
D
E
F
G
H
I
J
K
L
块片
寄存器
(B)
寄存器0
寄存器1
寄存器2
A
B
C
D
E
F
G
H
I
J
K
L
块片
FMA存储
(C)
A
B
C
D
E
F
G
H
I
J
K
L
块片

图2

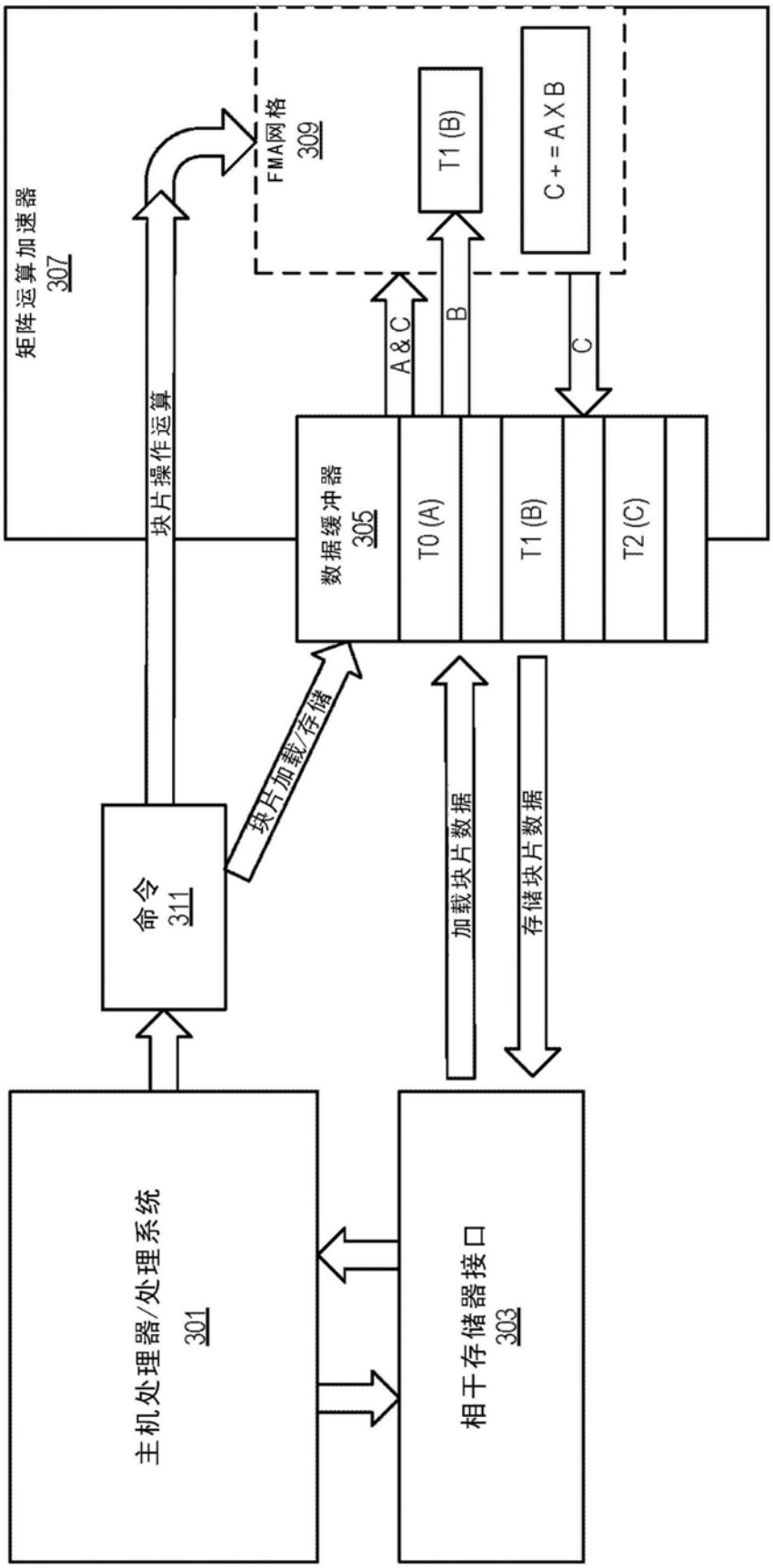

矩阵运算加速器
307
FMA网格
309
T1 (B)
C + = A X B
A & C
B
C
块片操作运算
数据缓冲器
305
T0 (A)
T1 (B)
T2 (C)
块片加载/存储
命令
311
加载块片数据
存储块片数据
主机处理器/处理系统
301
相干存储器接口
303

图3

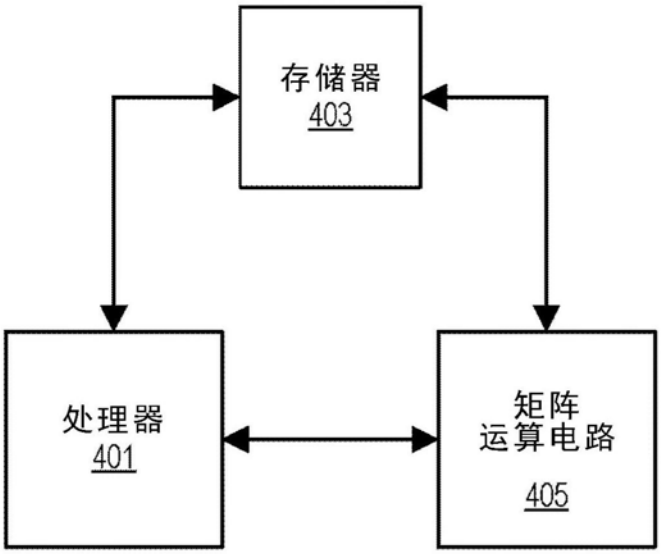

存储器
403
处理器
401
矩阵
运算电路
405

图4

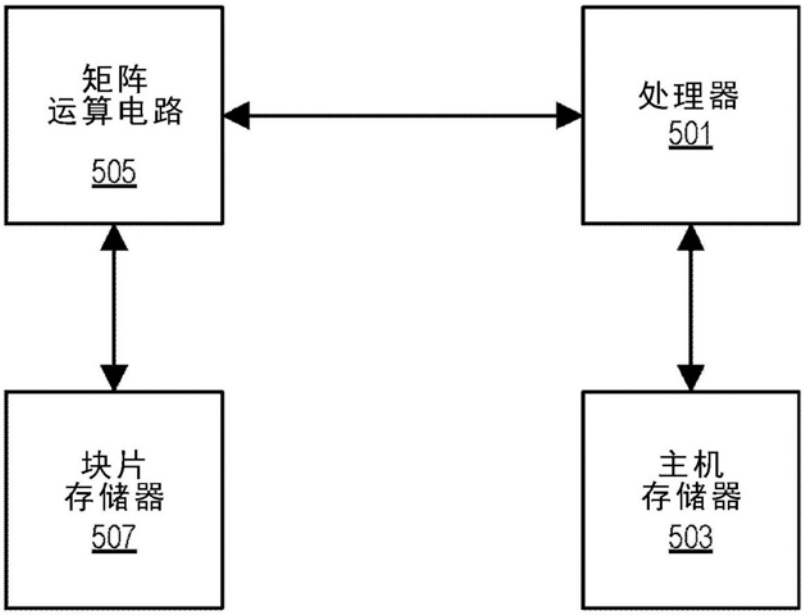

矩阵
运算电路
505
处理器
501
块片
存储器
507
主机
存储器
503

图5

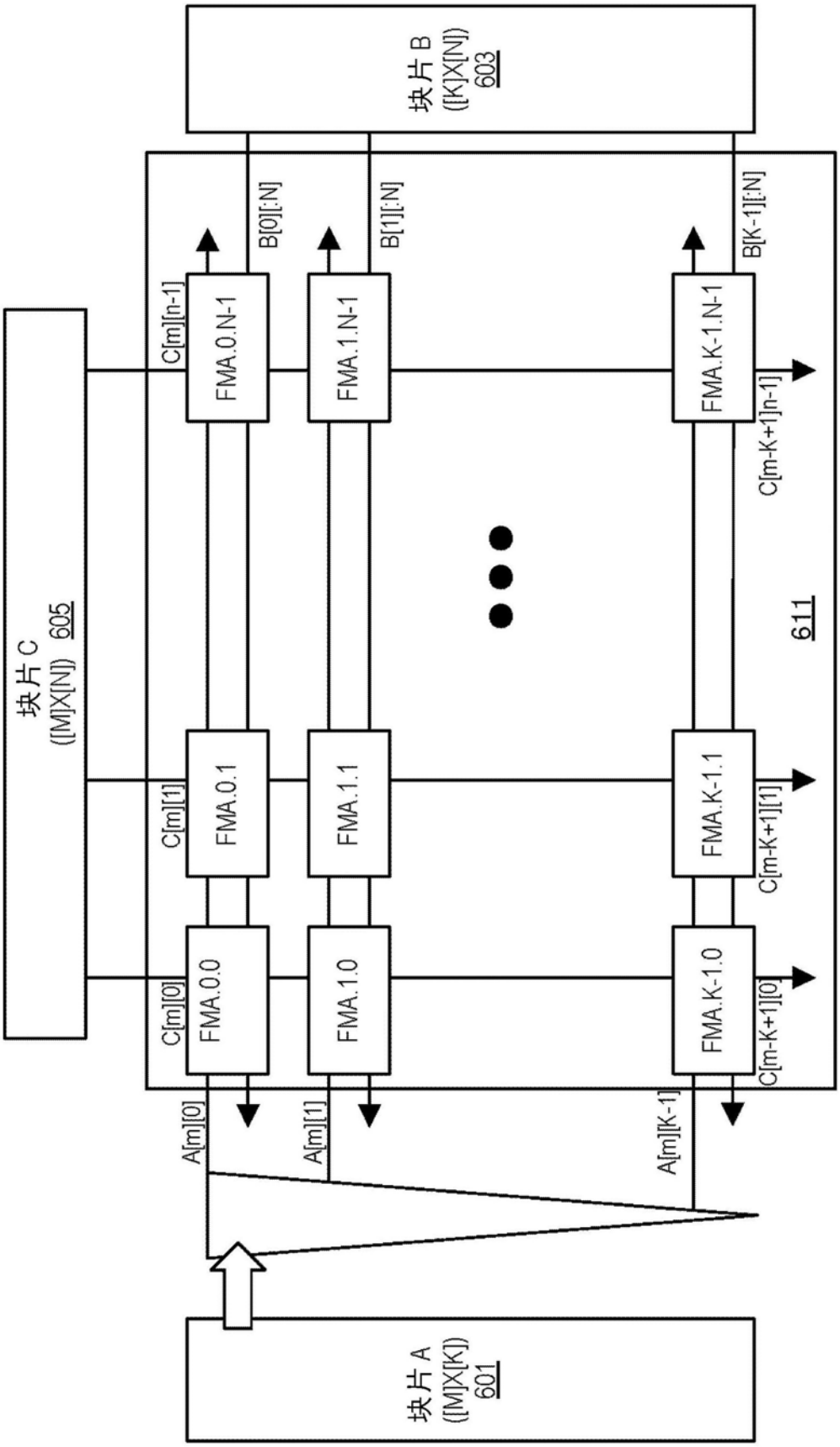

块片 B
([K]X[N])
603
B[0][:N]
B[1][:N]
B[K-1][:N]
C[m][n-1]
FMA.0.N-1
FMA.1.N-1
FMA.K-1.N-1
C[m-K+1][n-1]
块片 C
([M]X[N]) 605
611
C[m][1]
FMA.0.1
FMA.1.1
FMA.K-1.1
C[m-K+1][1]
C[m][0]
FMA.0.0
FMA.1.0
FMA.K-1.0
C[m-K+1][0]
A[m][0]
A[m][1]
A[m][K-1]
块片 A
([M]X[K])
601

图6

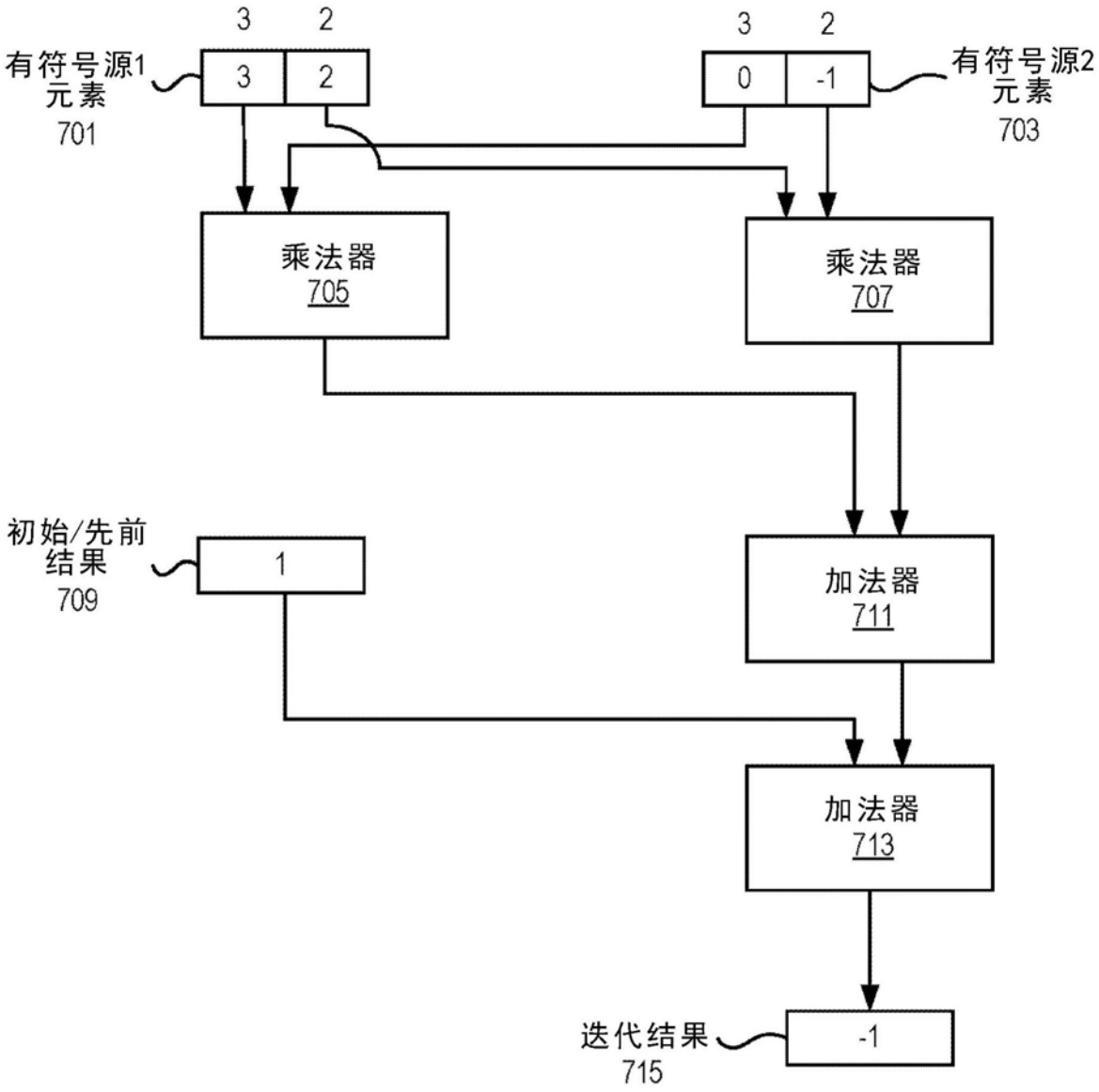

3
2
有符号源1
元素
701
3
2
3
2
0
-1
有符号源2
元素
703
乘法器
705
乘法器
707
初始/先前
结果
709
1
加法器
711
加法器
713
迭代结果
715
-1

图7

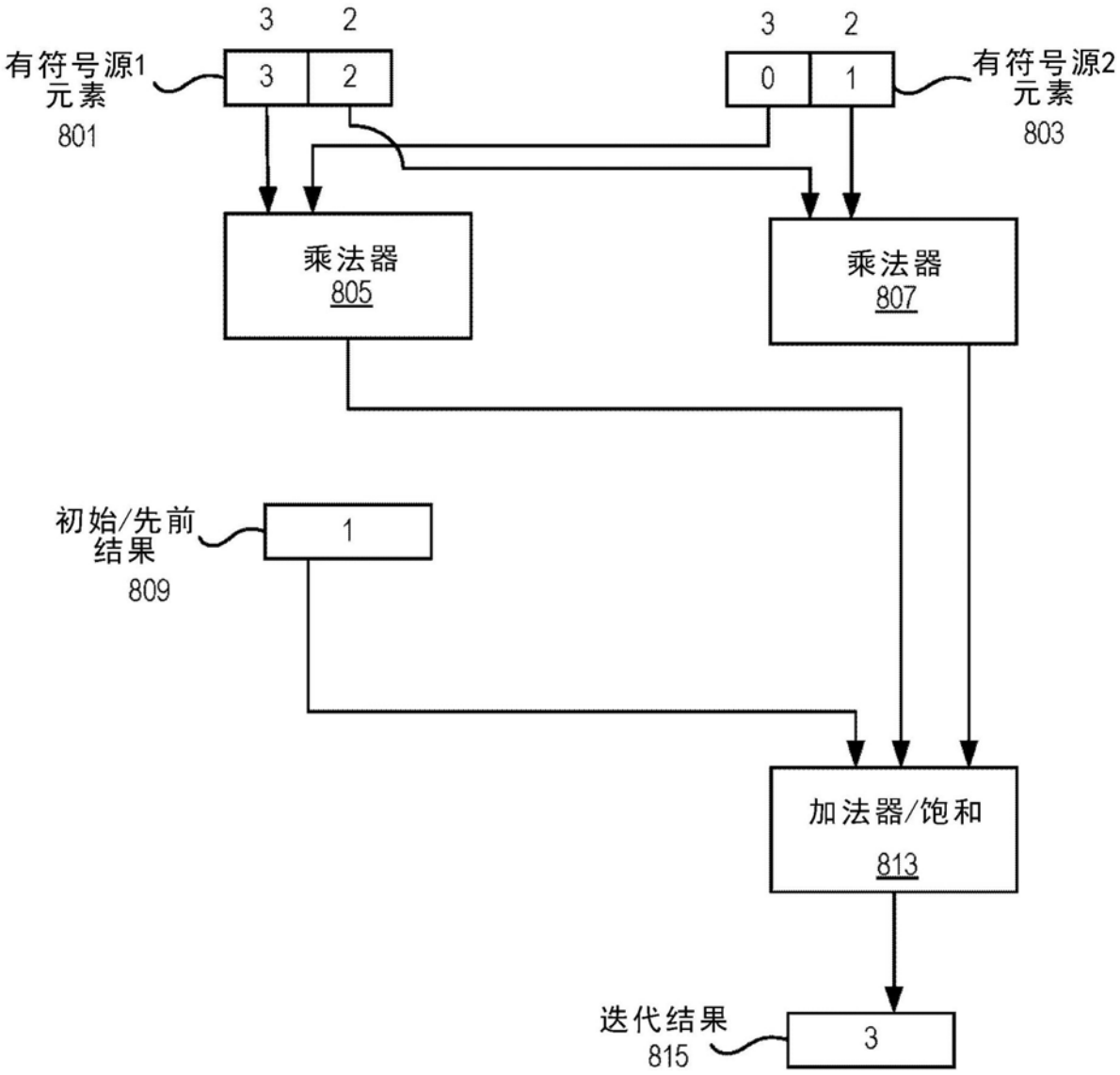

3
2
有符号源1
元素
801
3
2
3
2
0
1
有符号源2
元素
803
乘法器
805
乘法器
807
初始/先前
结果
809
1
加法器/饱和
813
迭代结果
815
3

图8

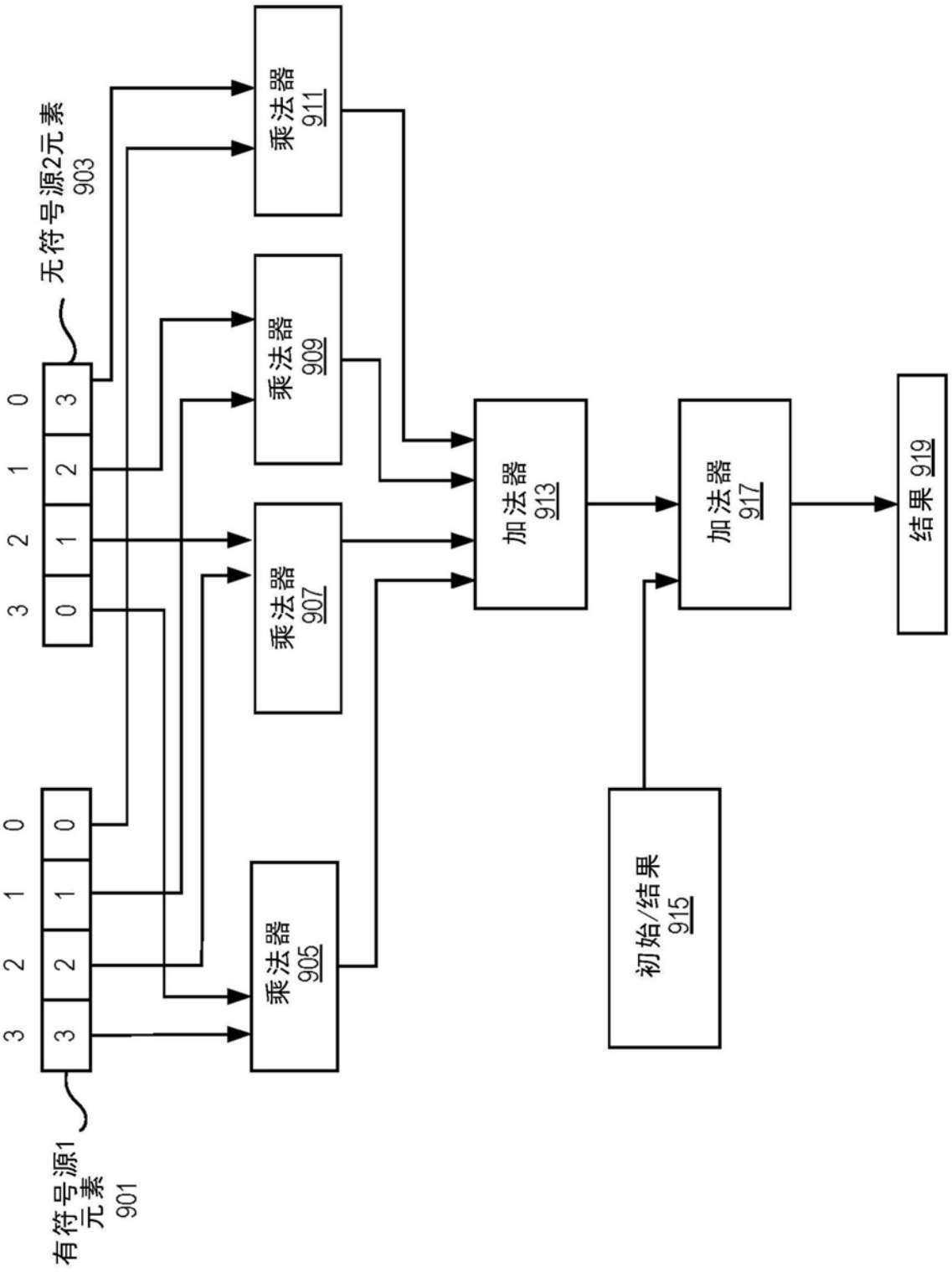

有符号源1元素 901
无符号源2元素 903
3 2 1 0
乘法器 905
乘法器 907
乘法器 909
乘法器 911
加法器 913
初始/结果 915
加法器 917
结果 919

图9

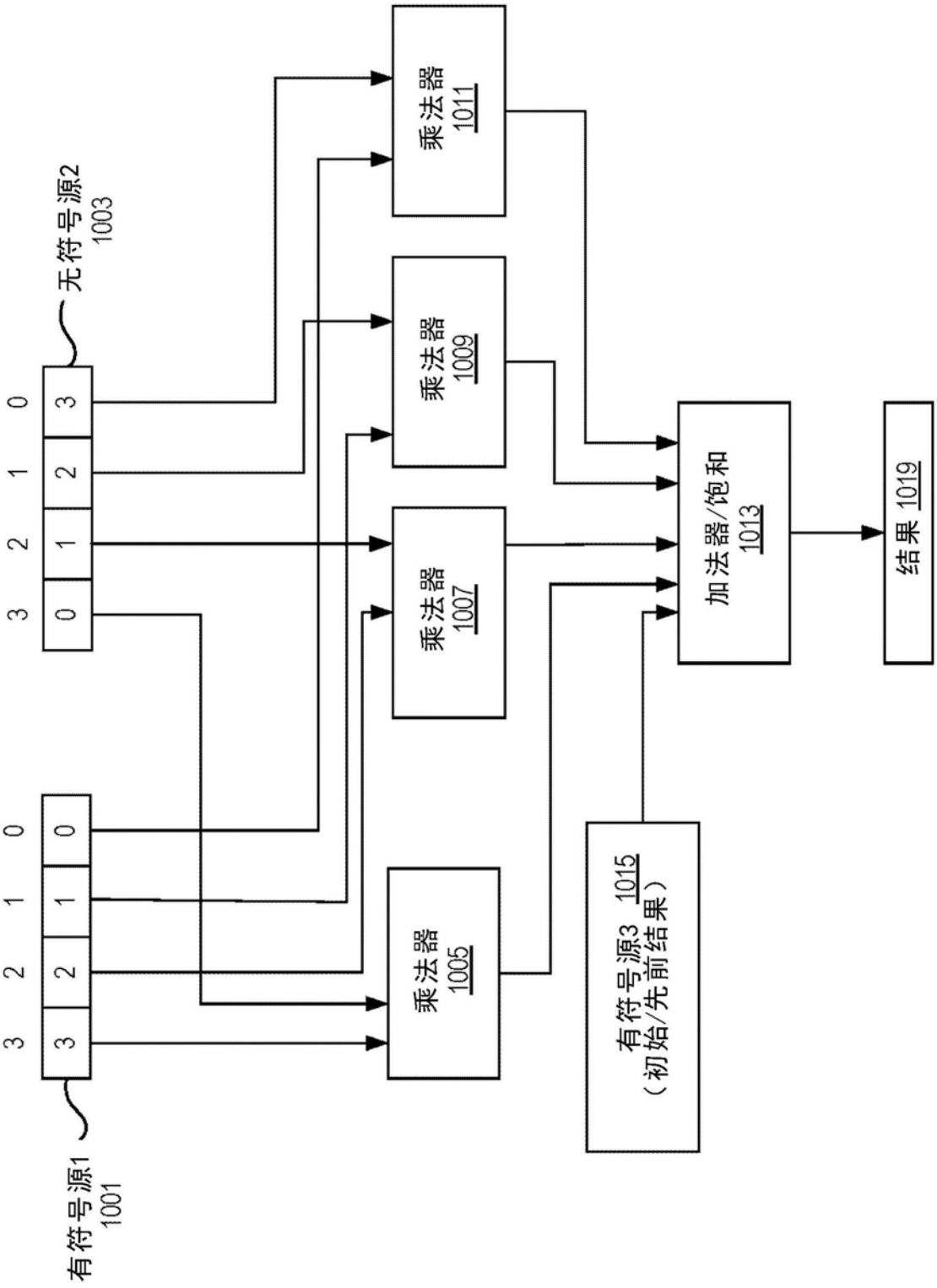

有符号源1
1001
无符号源2
1003
3 2 1 0
3 2 1 0
乘法器
1005
乘法器
1007
乘法器
1009
乘法器
1011
有符号源3 1015
（初始/先前结果）
加法器/饱和
1013
结果 1019

图10

累加器2倍输入大小 1101

源	位	累加器	位
字节	8	字 /HPFP	16
字	16	INT32/SPFP	32
SPFP/INT32	32	INT64/DPFP	64

累加器4倍输入大小 1103

源	位	累加器	位
字节	8	INT32/SPFP	32
字	16	INT64/DPFP	64

累加器8倍输入大小 1105

源	位	累加器	位
字节	8	INT64/DPFP	64

图11

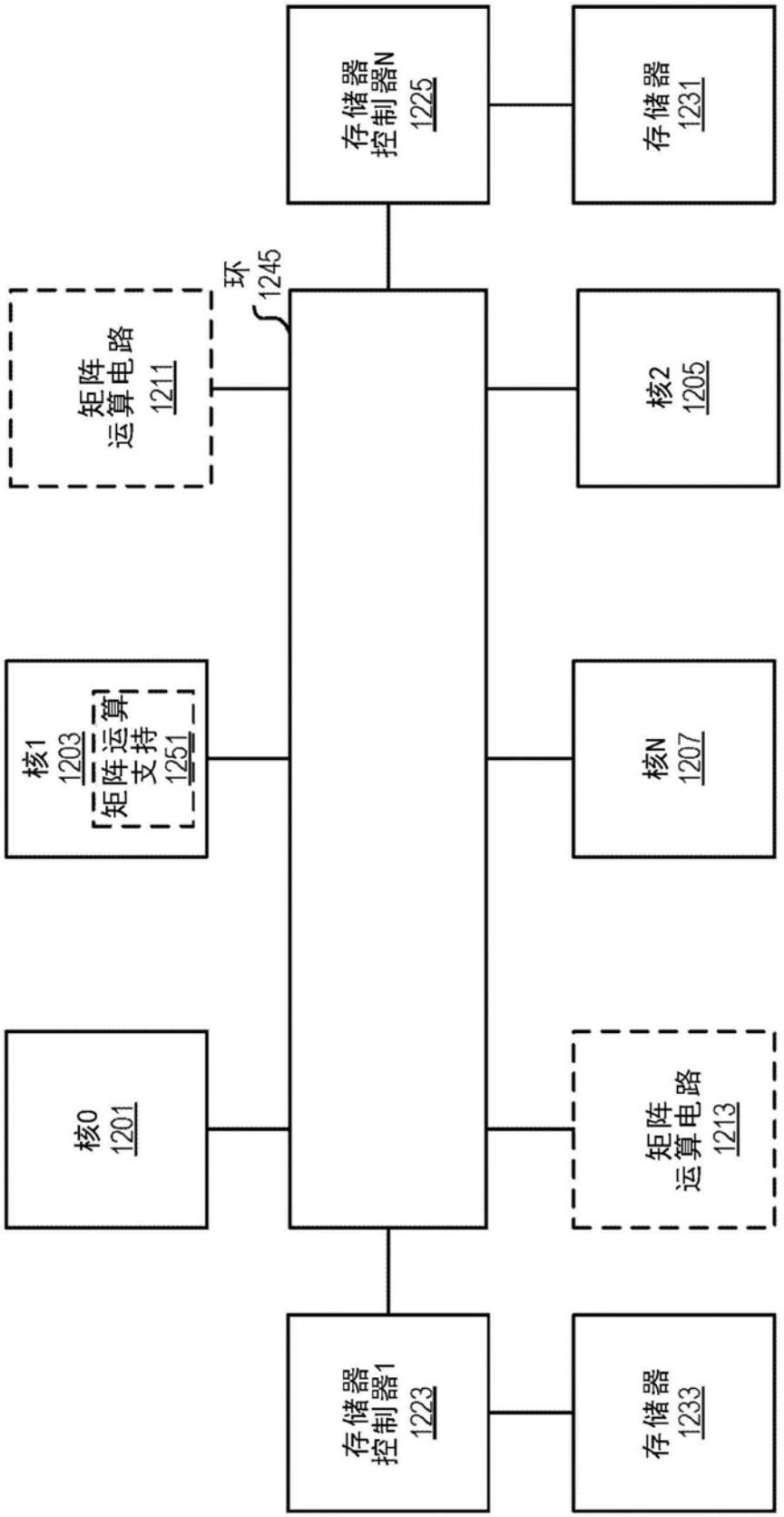

存储器控制器N 1225
存储器 1231
环 1245
矩阵运算电路 1211
核2 1205
核1 1203 [矩阵运算支持 1251]
核N 1207
核0 1201
矩阵运算电路 1213
存储器控制器1 1223
存储器 1233

图12

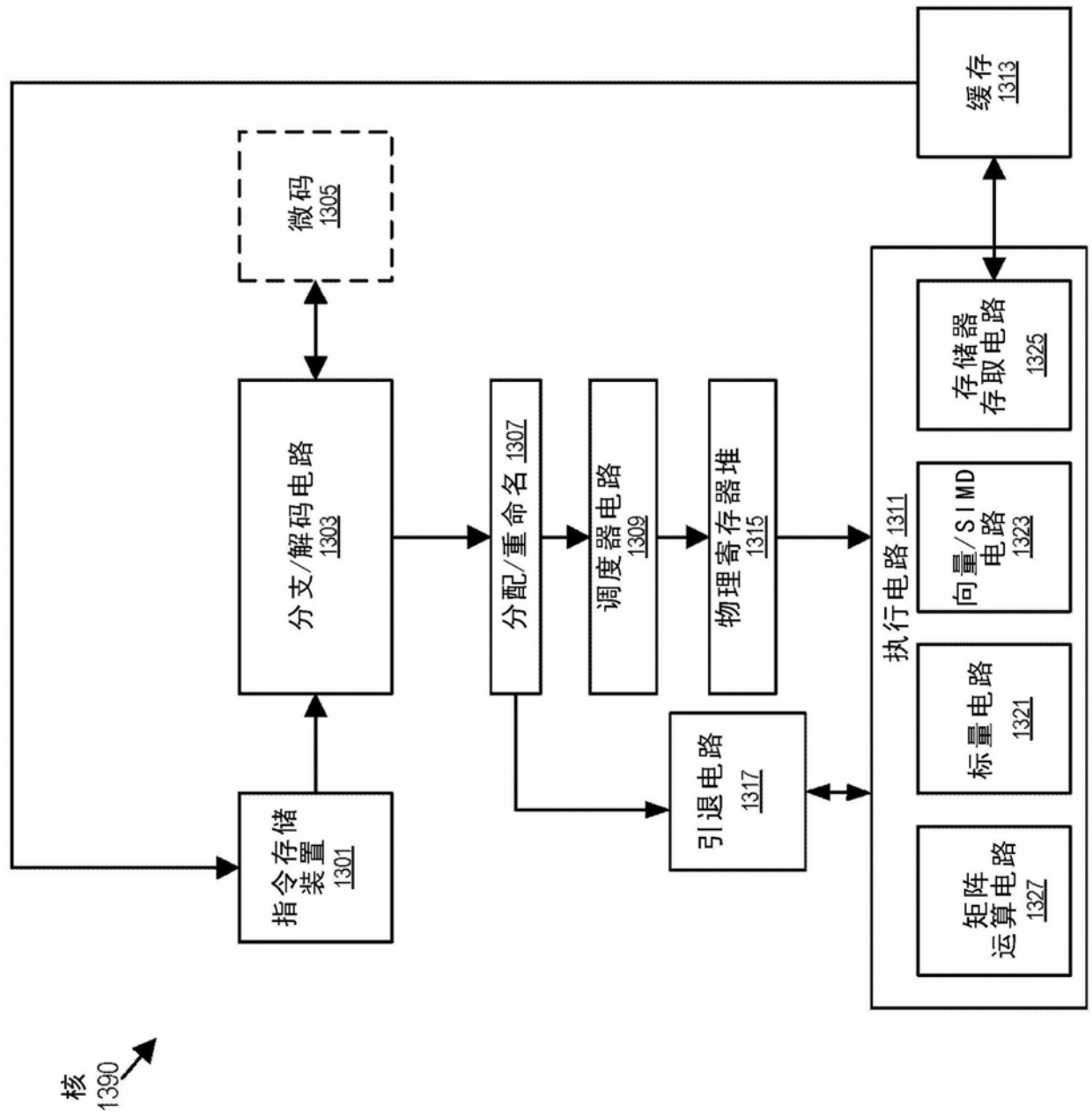

缓存 1313
微码 1305
分支/解码电路 1303
分配/重命名 1307
调度器电路 1309
物理寄存器堆 1315
存储器存取电路 1325
向量/SIMD 电路 1323
执行电路 1311
标量电路 1321
矩阵运算电路 1327
引退电路 1317
指令存储装置 1301
核 1390

图13

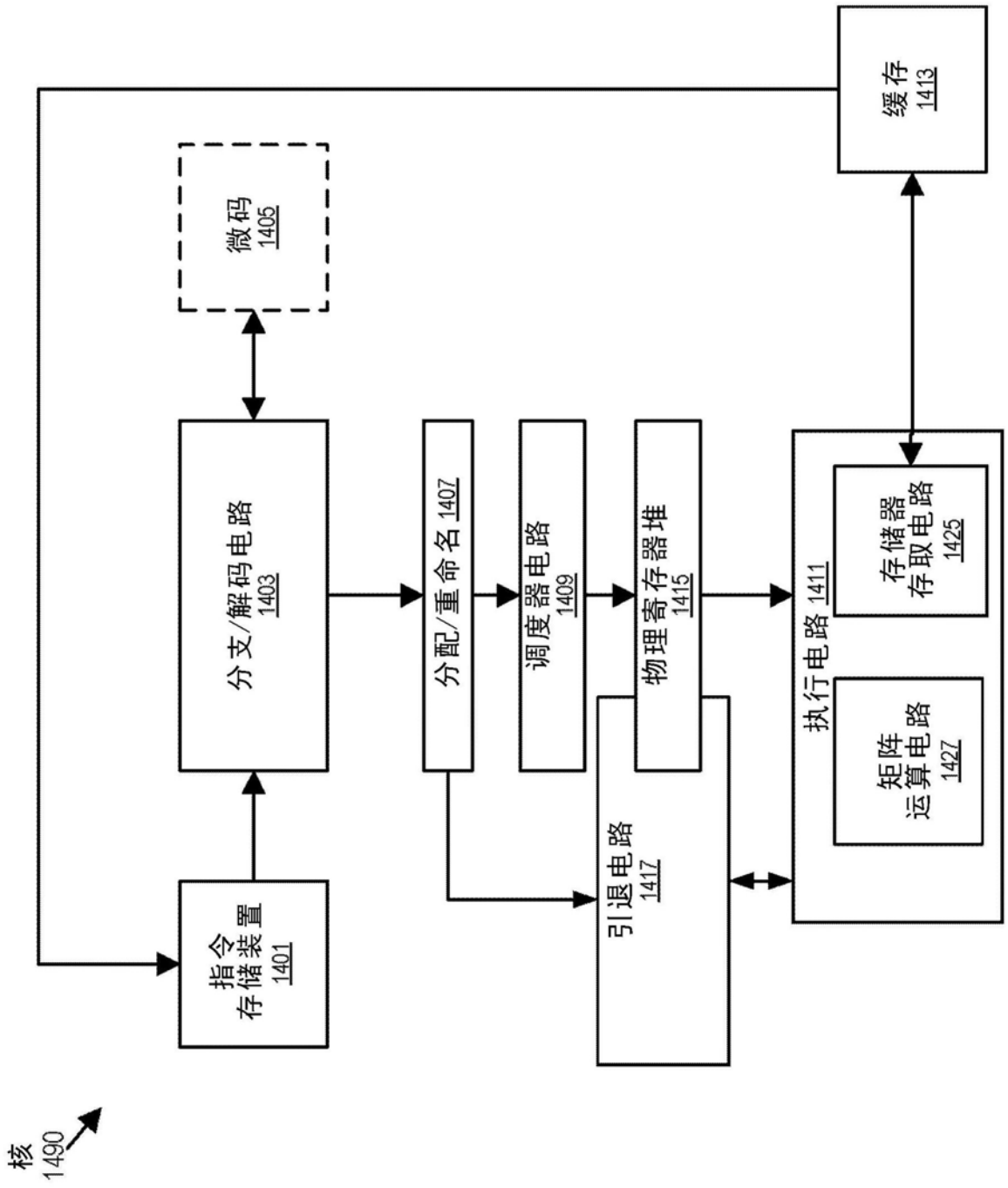

缓存
1413
微码
1405
分支/解码电路
1403
分配/重命名 1407
调度器电路
1409
物理寄存器堆
1415
执行电路 1411
存储器
存取电路
1425
矩阵
运算电路
1427
引退电路
1417
指令
存储装置
1401
核
1490

图14

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \end{bmatrix}$$

地址	值
0	A_{11}
1	A_{12}
2	A_{13}
3	A_{21}
4	A_{22}
5	A_{23}

行主要

地址	值
0	A_{11}
1	A_{21}
2	A_{12}
3	A_{22}
4	A_{13}
5	A_{23}

列主要

图15

```
TILECONFIG [RAX]
// 假设一些外循环驱动缓存分块（未示出）
{
TILELOAD TMM0, RSI+RDI // RSI指向 C，RDI具有
TILELOAD TMM1, RSI+RDI+N // C的第二块片，在SIMD维度N中展开
MOV KK, 0
LOOP:
TILELOAD TMM2, R8+R9 // SRC2是A的跨越式加载，重用于2个TMMA指令
TILELOAD TMM3, R10+R11 // SRC1是B的跨越式加载
TMMAPS TMM0, TMM2, TMM3 // 更新C的左边块片
TILELOAD TMM3, R10+R11+N // SRC1从下一个最右边块片开始加载B
TMMAPS TMM1, TMM2, TMM3 // 更新C的右边块片
ADD R8, K // 利用循环外部已知的常量来更新指针
ADD R10, K*R11
ADD KK, K
CMP KK, LIMIT
JNE LOOP
TILESTORE RSI+RDI, TMM0 // 更新存储器中的C矩阵
TILESTORE RSI+RDI+M, TMM1
} // 外循环结束
TILERELEASE // 返回块片到初始状态
```

图16

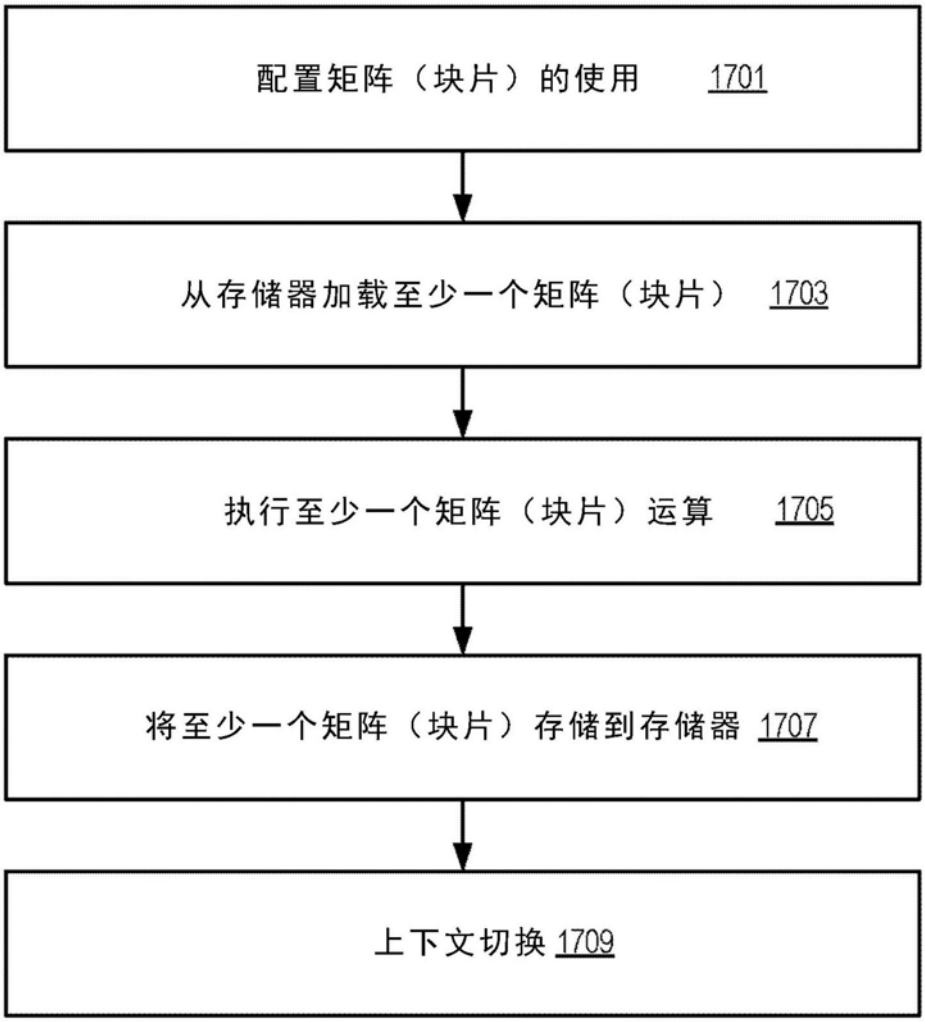

配置矩阵（块片）的使用 1701
从存储器加载至少一个矩阵（块片） 1703
执行至少一个矩阵（块片）运算 1705
将至少一个矩阵（块片）存储到存储器 1707
上下文切换 1709

图17

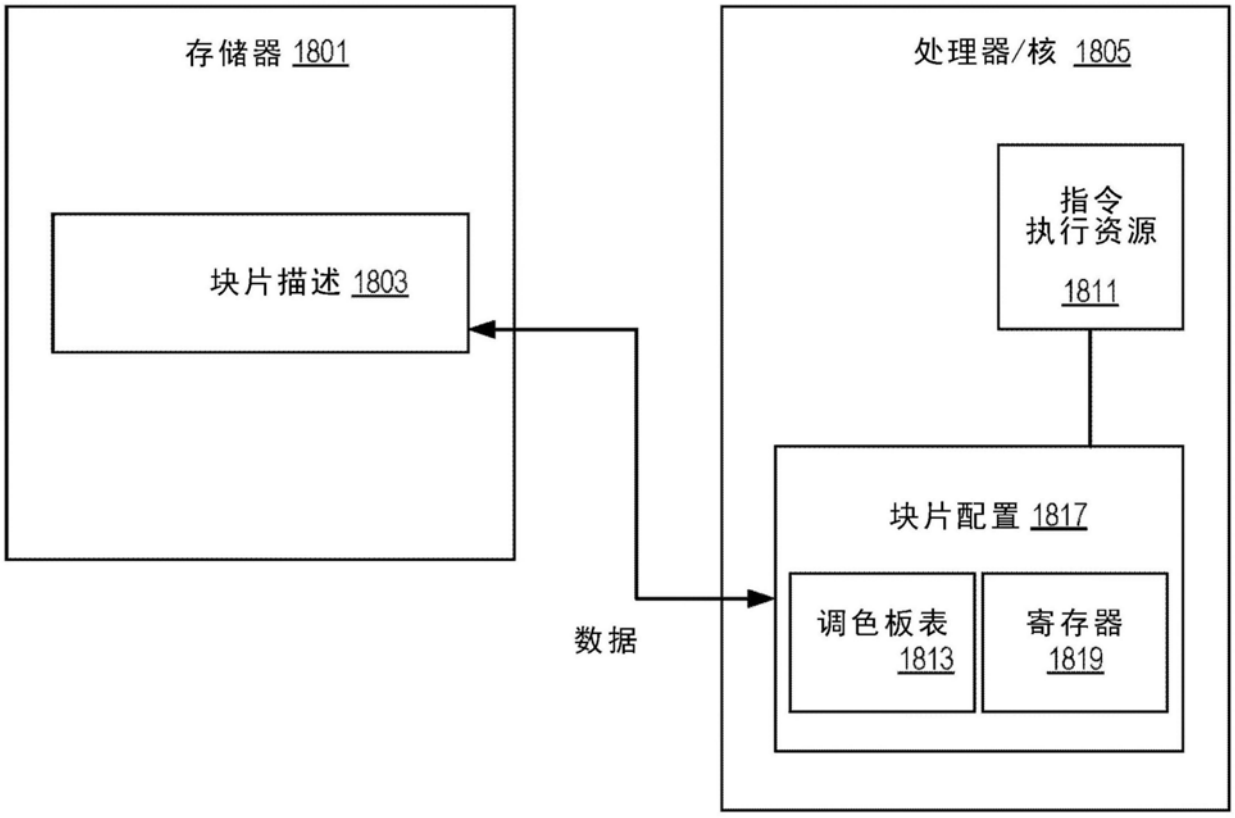

存储器 1801
块片描述 1803
处理器/核 1805
指令
执行资源
1811
块片配置 1817
调色板表
1813
寄存器
1819
数据

图18

调色板ID 1901	STARTM 1903
STARTP 1905	对指示符 1907
0	0
0	0

...

0	0
TMM0行 1913	TMM0列 1915
TMM1行	TMM1列
. . .	
TMM15行	TMM15列
0	

图19

图20(A)

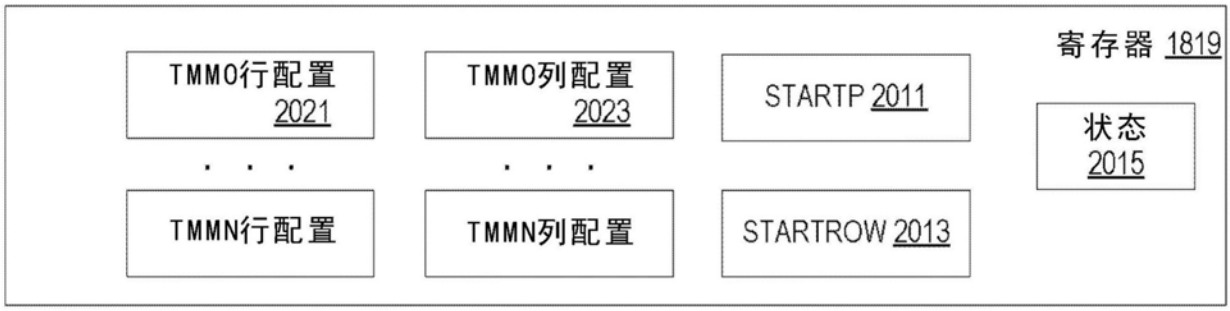


图20(B)

图20(C)

图20(D)

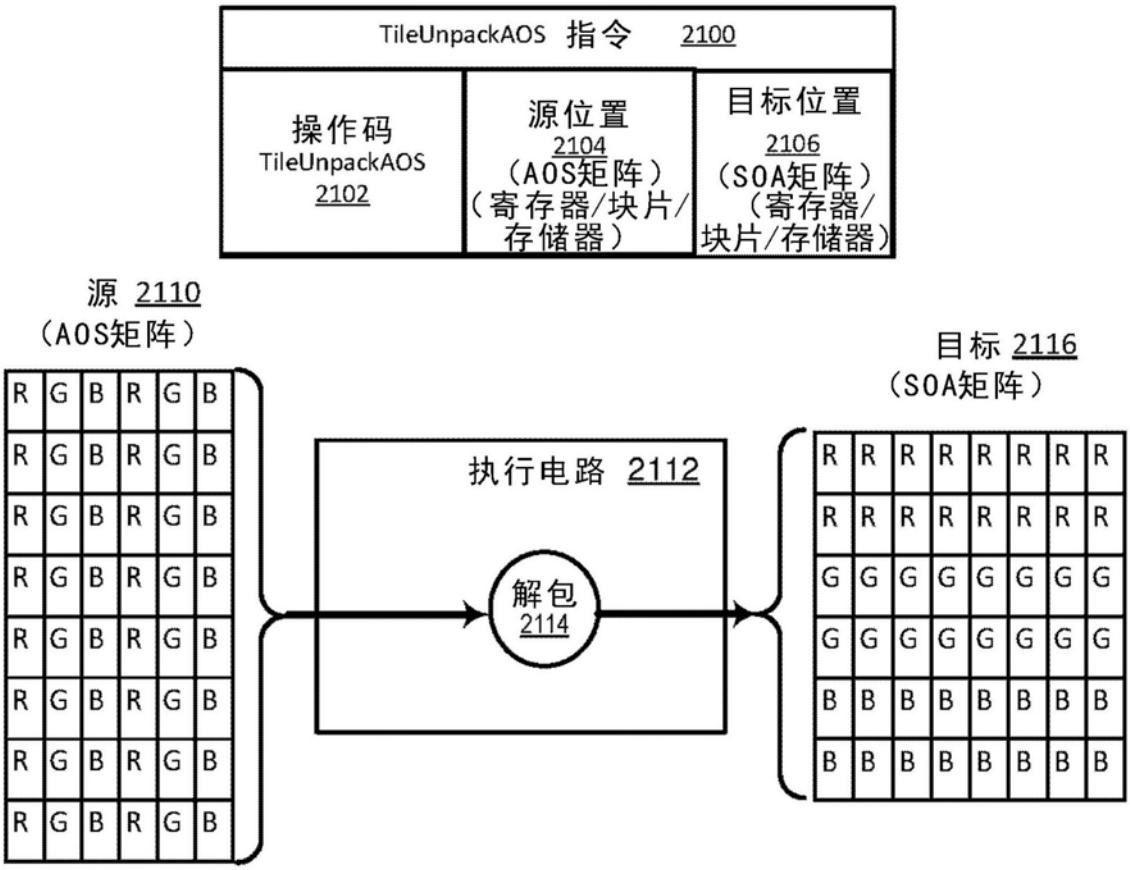

TileUnpackAOS 指令 2100
操作码
TileUnpackAOS
2102
源位置
2104
（AOS矩阵）
（寄存器/块片/
存储器）
目标位置
2106
（SOA矩阵）
（寄存器/
块片/存储器）
源 2110
（AOS矩阵）
目标 2116
（SOA矩阵）
执行电路 2112
解包
2114

图21A

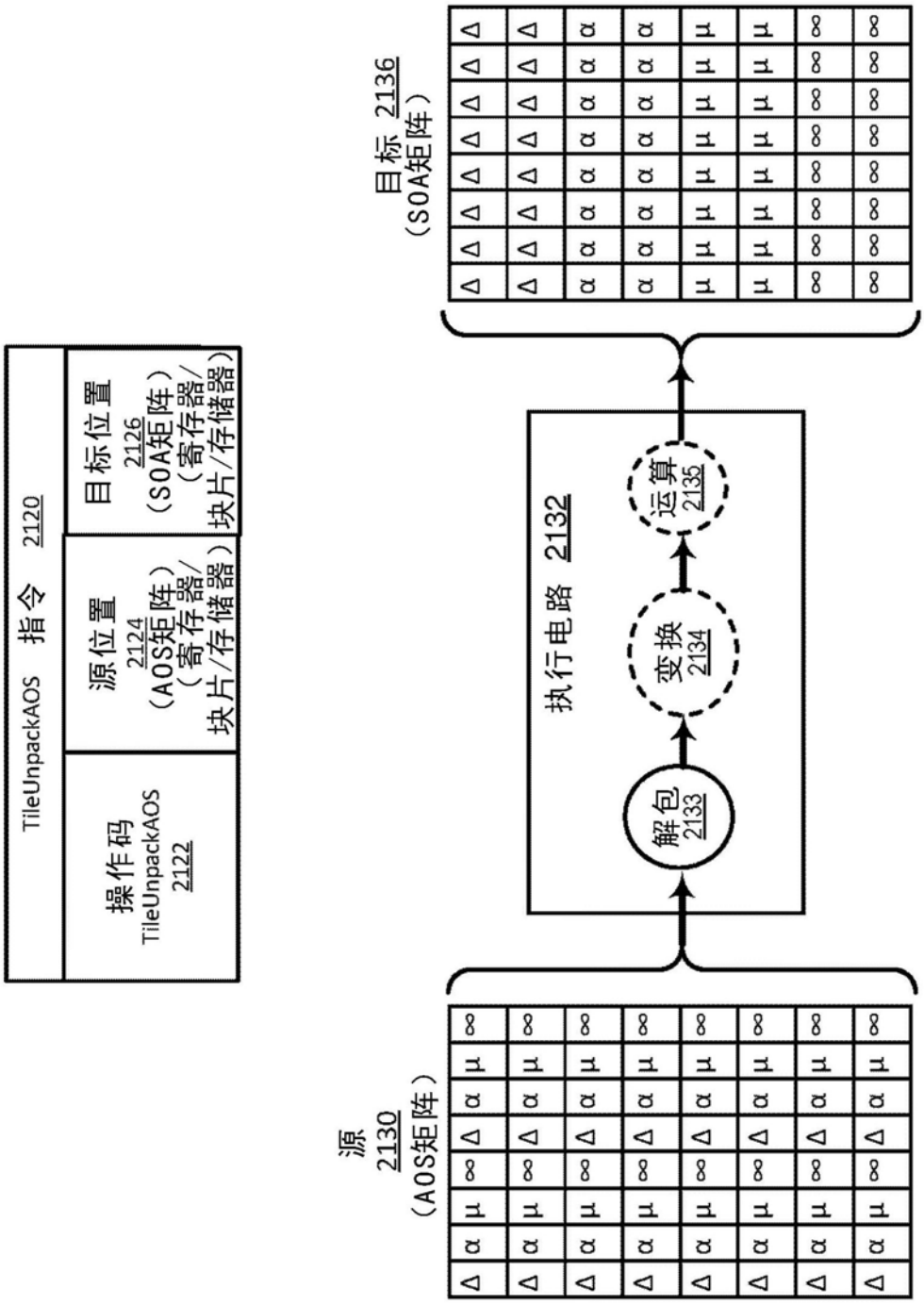

TileUnpackAOS 指令 2120
操作码
TileUnpackAOS
2122
源位置
2124
（AOS矩阵）
（寄存器/
块片/存储器）
目标位置
2126
（SOA矩阵）
（寄存器/
块片/存储器）
源
2130
（AOS矩阵）
执行电路 2132
解包
2133
变换
2134
运算
2135
目标 2136
（SOA矩阵）

图21B

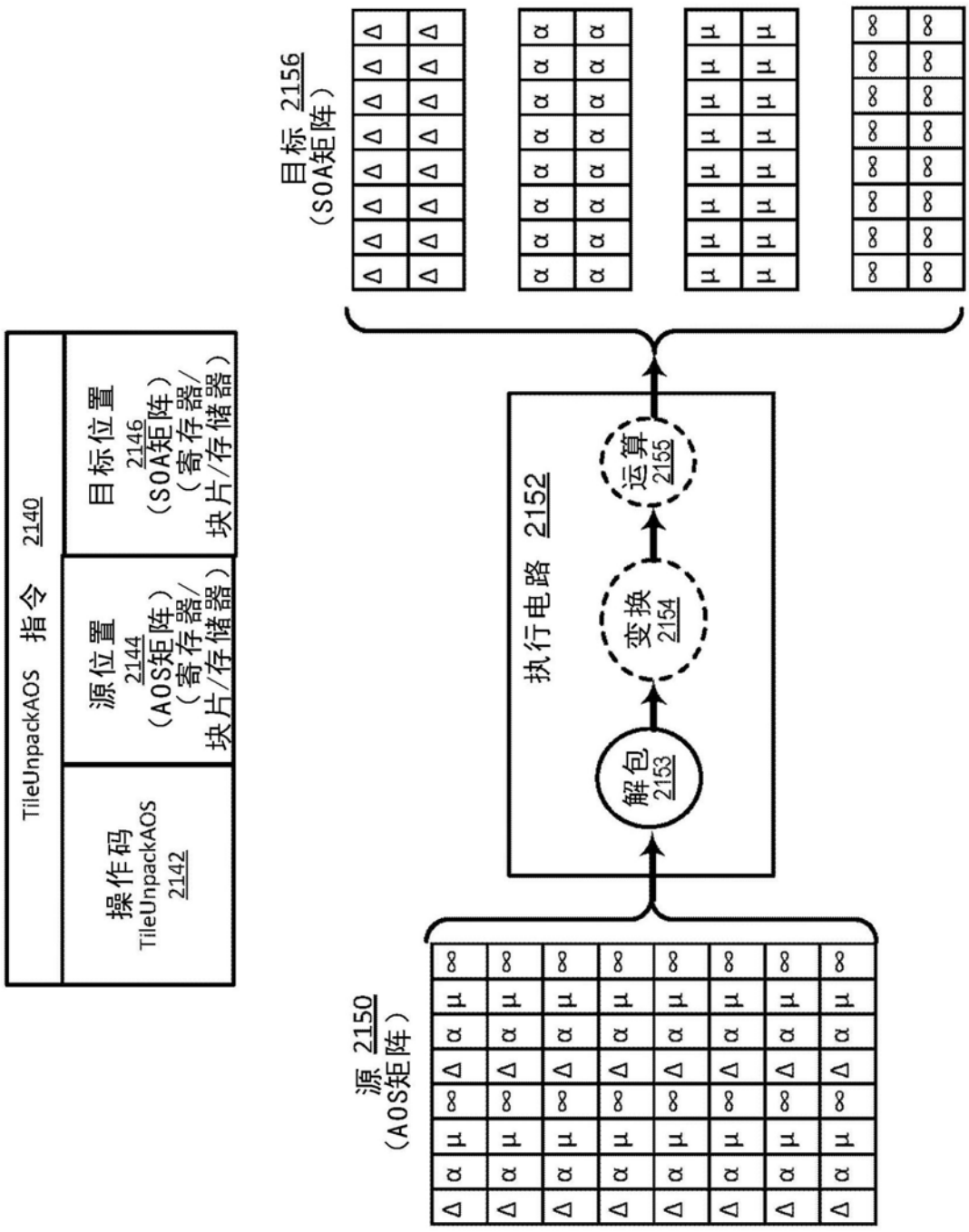

TileUnpackAOS 指令 2140
操作码
TileUnpackAOS
2142
源位置
2144
（AOS矩阵）
（寄存器/
块片/存储器）
目标位置
2146
（SOA矩阵）
（寄存器/
块片/存储器）
源 2150
（AOS矩阵）
执行电路 2152
解包
2153
变换
2154
运算
2155
目标 2156
（SOA矩阵）

图21C

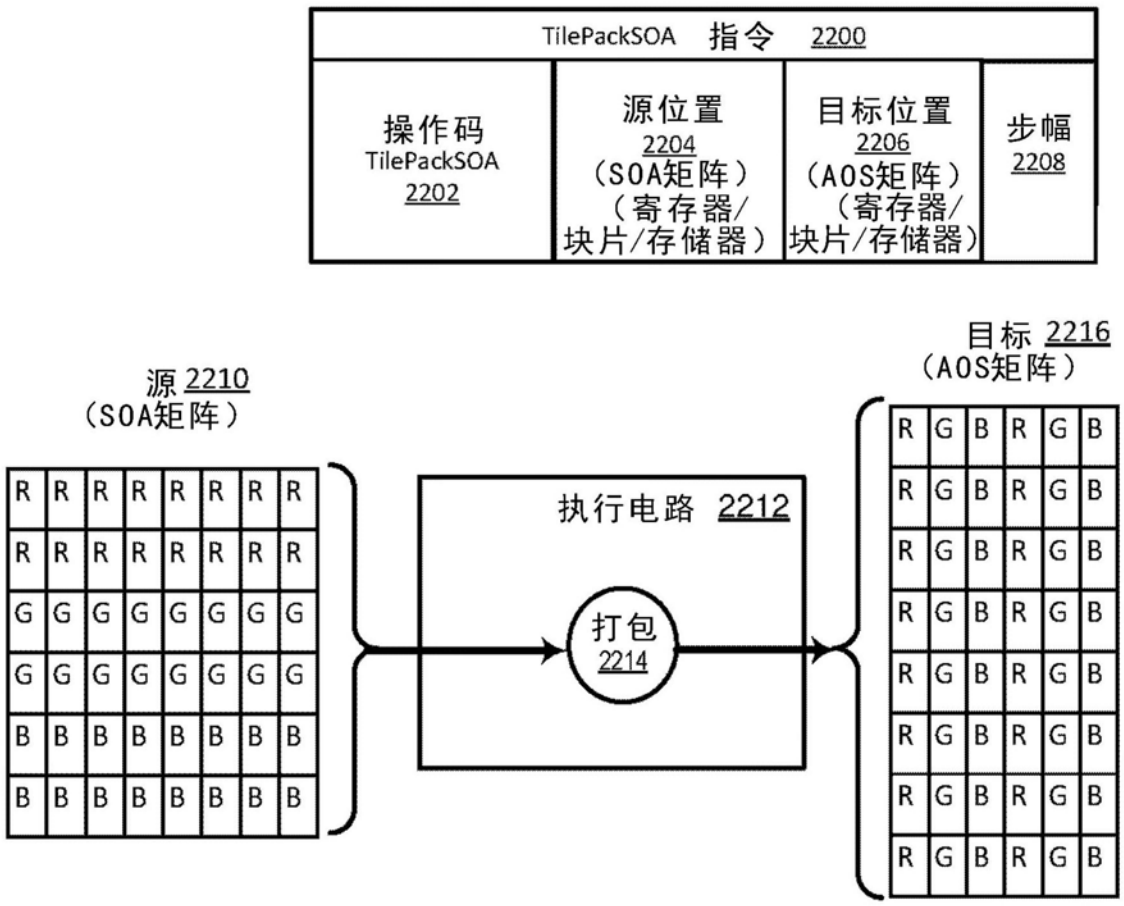

TilePackSOA 指令 2200
操作码
TilePackSOA
2202
源位置
2204
（SOA矩阵）
（寄存器/
块片/存储器）
目标位置
2206
（AOS矩阵）
（寄存器/
块片/存储器）
步幅
2208
源 2210
（SOA矩阵）
目标 2216
（AOS矩阵）
执行电路 2212
打包
2214

图22A

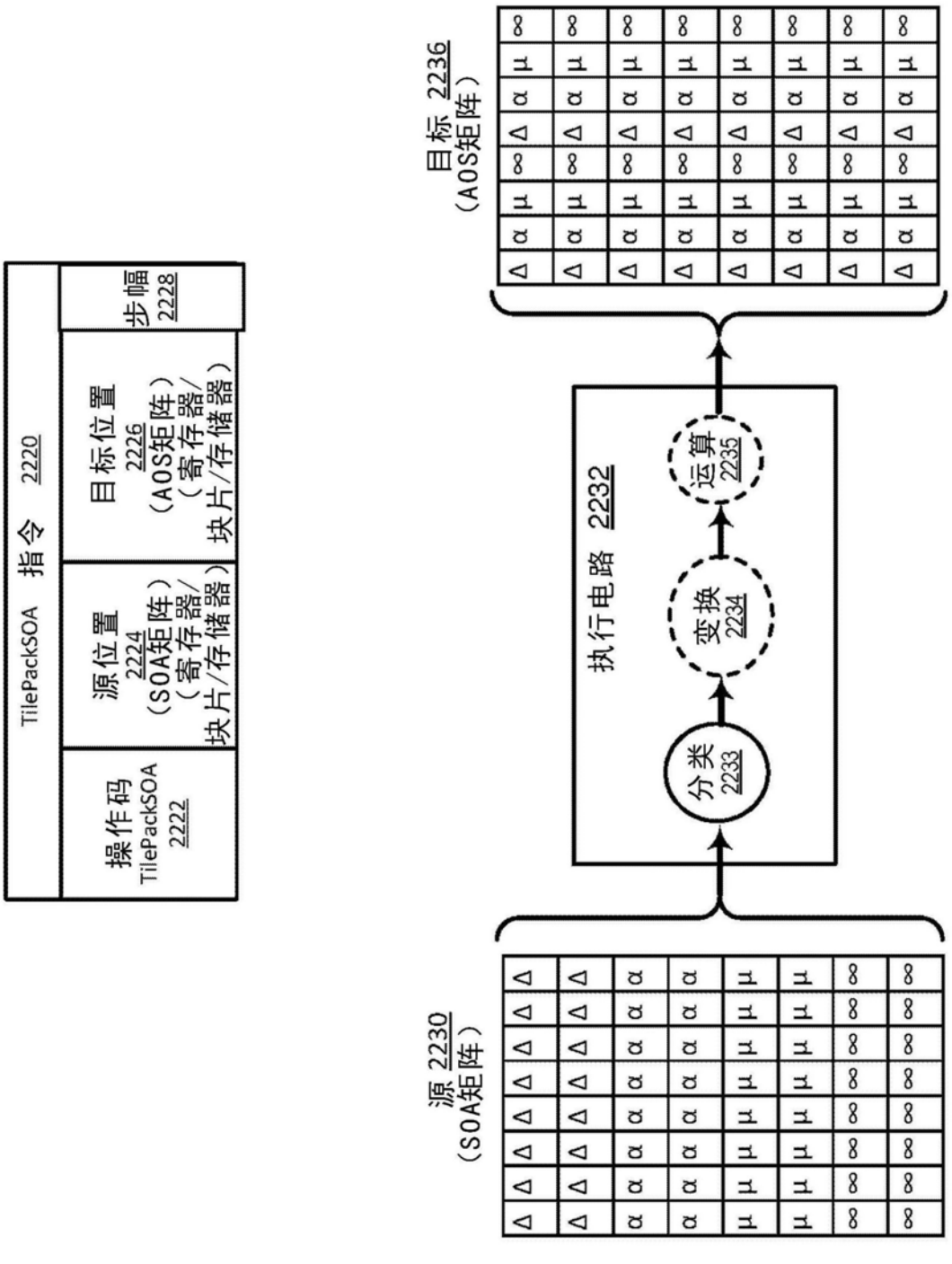

TilePackSOA 指令 2220
操作码 TilePackSOA 2222
源位置 2224 （SOA矩阵）（寄存器/块片/存储器）
目标位置 2226 （AOS矩阵）（寄存器/块片/存储器）
步幅 2228
源 2230 （SOA矩阵）
执行电路 2232
分类 2233
变换 2234
运算 2235
目标 2236 （AOS矩阵）

图22B

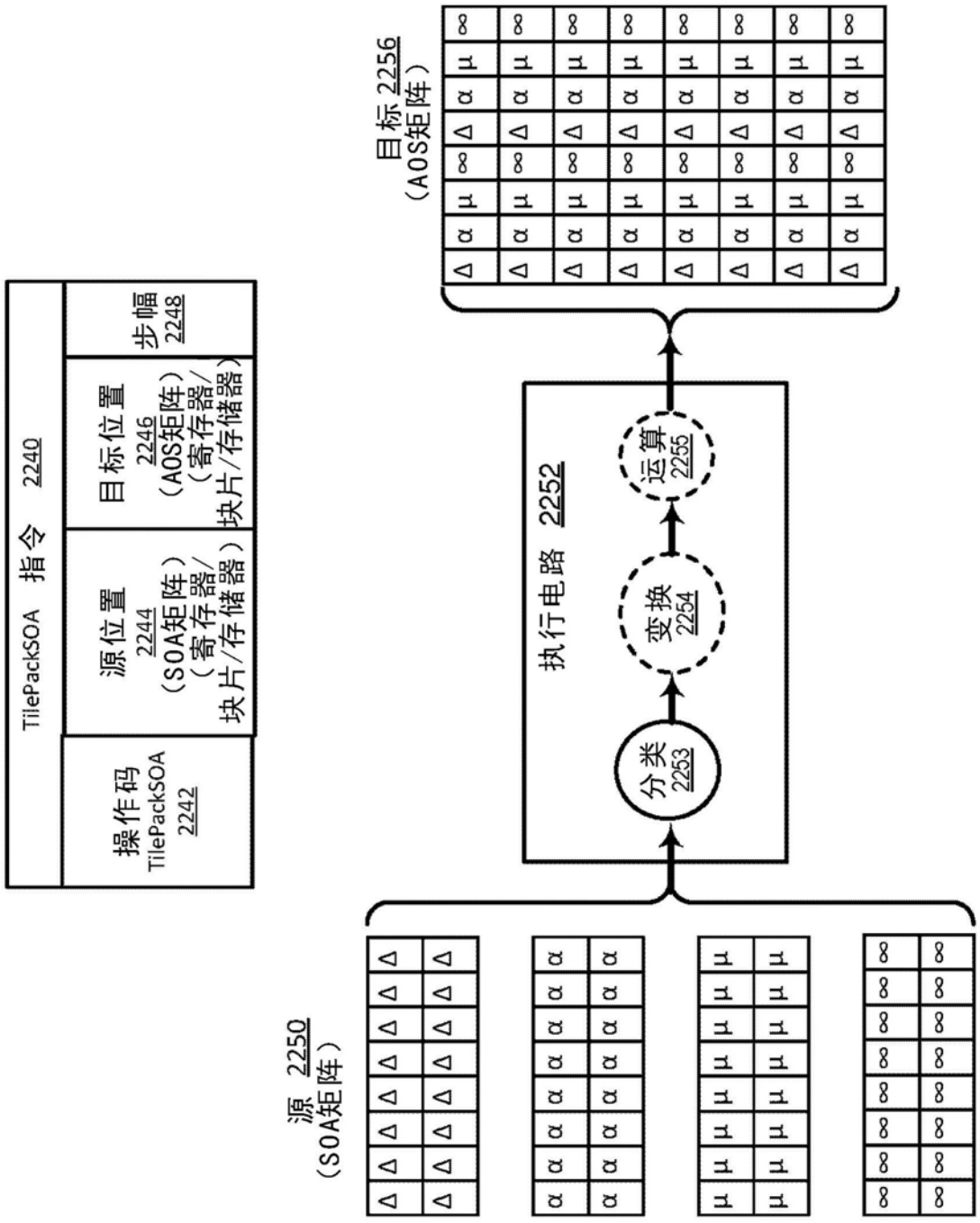

TilePackSOA 指令 2240
操作码 TilePackSOA 2242
源位置 2244 (SOA矩阵) (寄存器/块片/存储器)
目标位置 2246 (AOS矩阵) (寄存器/块片/存储器)
步幅 2248
源 2250 (SOA矩阵)
执行电路 2252
分类 2253
变换 2254
运算 2255
目标 2256 (AOS矩阵)

图22C

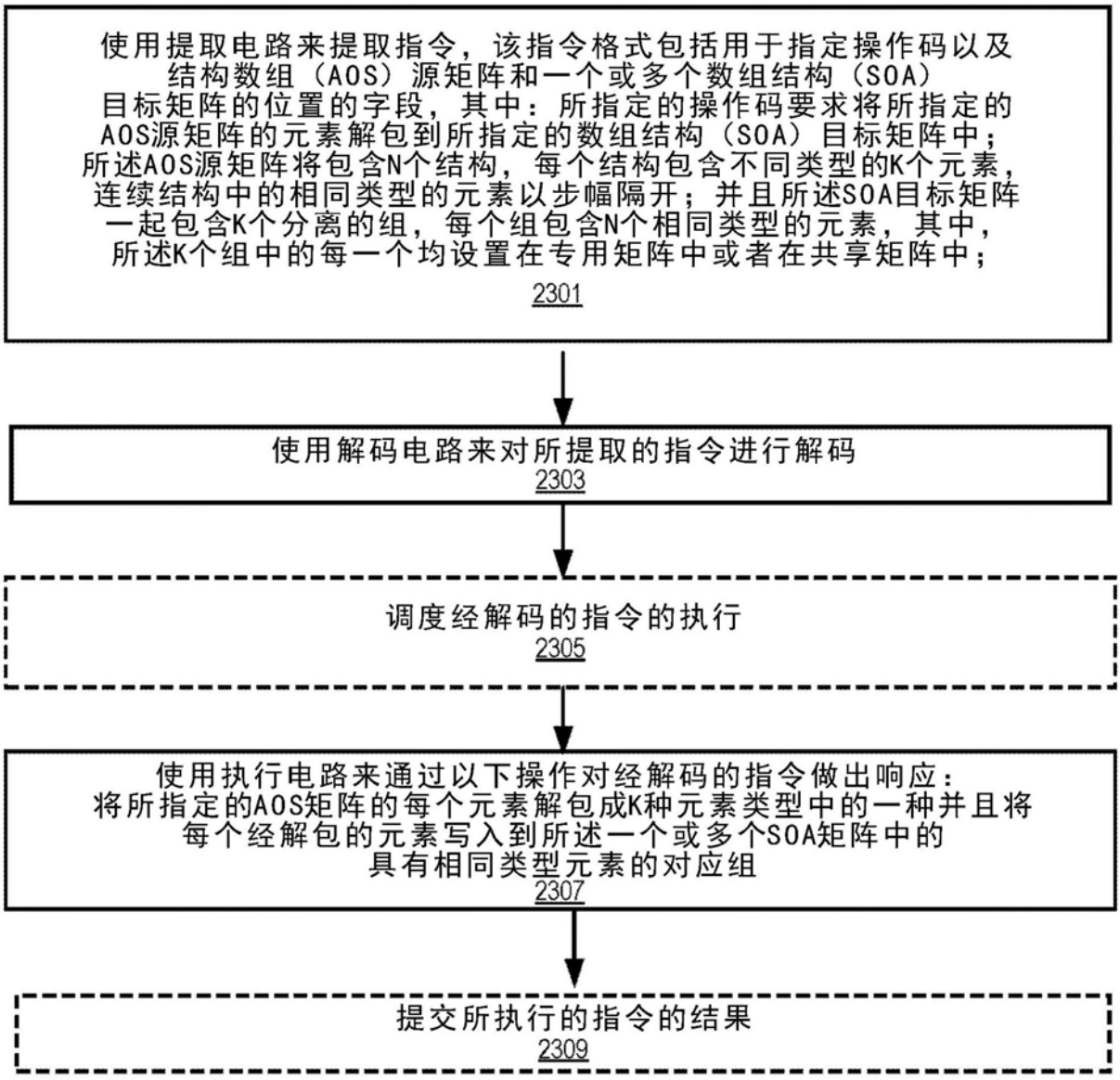

使用提取电路来提取指令，该指令格式包括用于指定操作码以及结构数组（AOS）源矩阵和一个或多个数组结构（SOA）目标矩阵的位置的字段，其中：所指定的操作码要求将所指定的AOS源矩阵的元素解包到所指定的数组结构（SOA）目标矩阵中；所述AOS源矩阵将包含N个结构，每个结构包含不同类型的K个元素，连续结构中的相同类型的元素以步幅隔开；并且所述SOA目标矩阵一起包含K个分离的组，每个组包含N个相同类型的元素，其中，所述K个组中的每一个均设置在专用矩阵中或者在共享矩阵中；
2301
使用解码电路来对所提取的指令进行解码
2303
调度经解码的指令的执行
2305
使用执行电路来通过以下操作对经解码的指令做出响应：将所指定的AOS矩阵的每个元素解包成K种元素类型中的一种并且将每个经解包的元素写入到所述一个或多个SOA矩阵中的具有相同类型元素的对应组
2307
提交所执行的指令的结果
2309

图23A

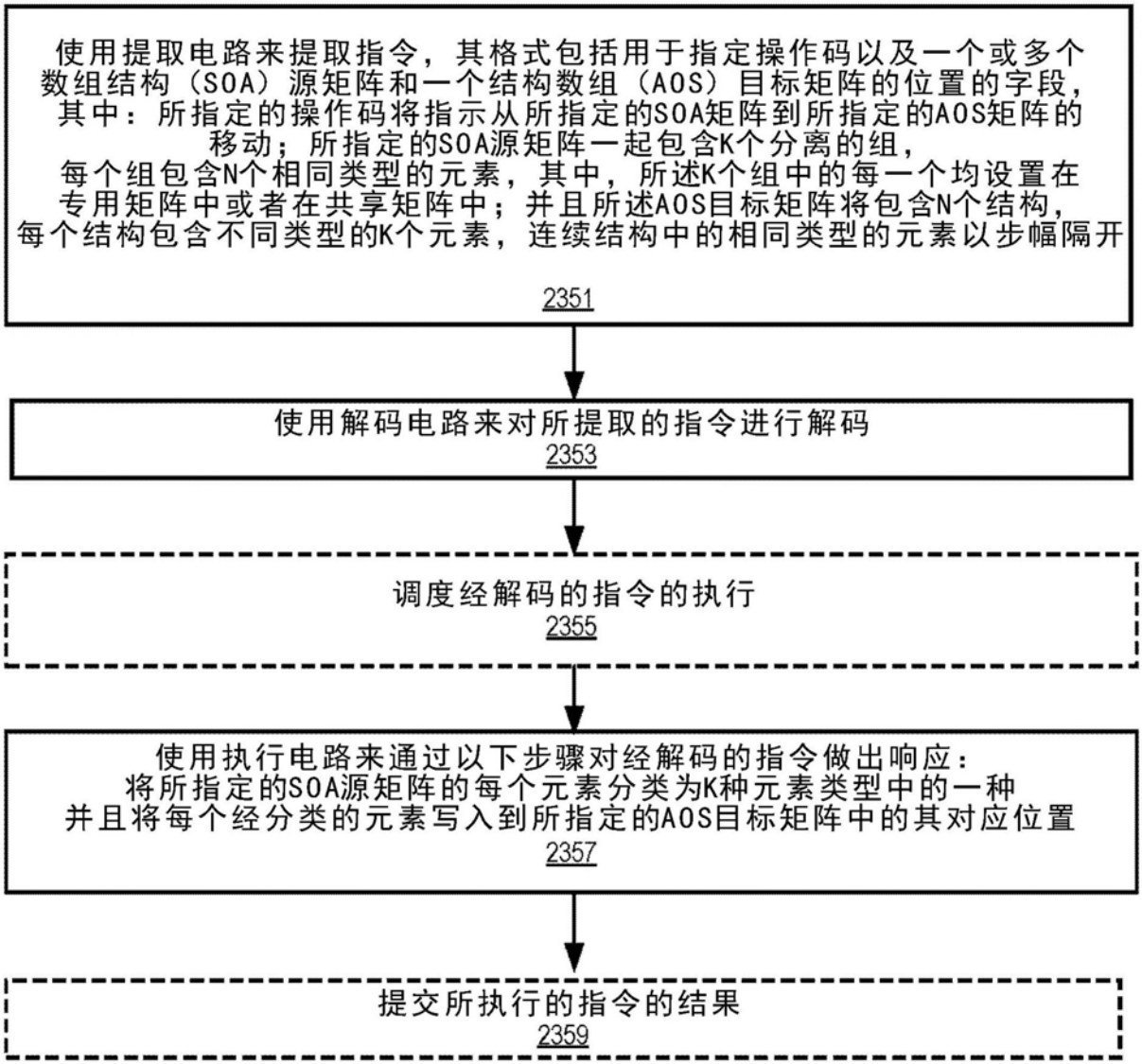


图23B

TileUnpackAOS和TilePackSOA指令格式 2400

操作码 (TileUnpackAOS */ TilePackSOA*) 2402	AOS矩阵（寄存器/块片/存储器） 2404	SOA矩阵（寄存器/块片/存储器） 2406	M（行数） 2408	N（列数） 2410	K（类型数） 2412	元素大小（微字节、半字节、字节、字、双字、四字） 2414	步幅 2416	辅助步幅 2418

图24

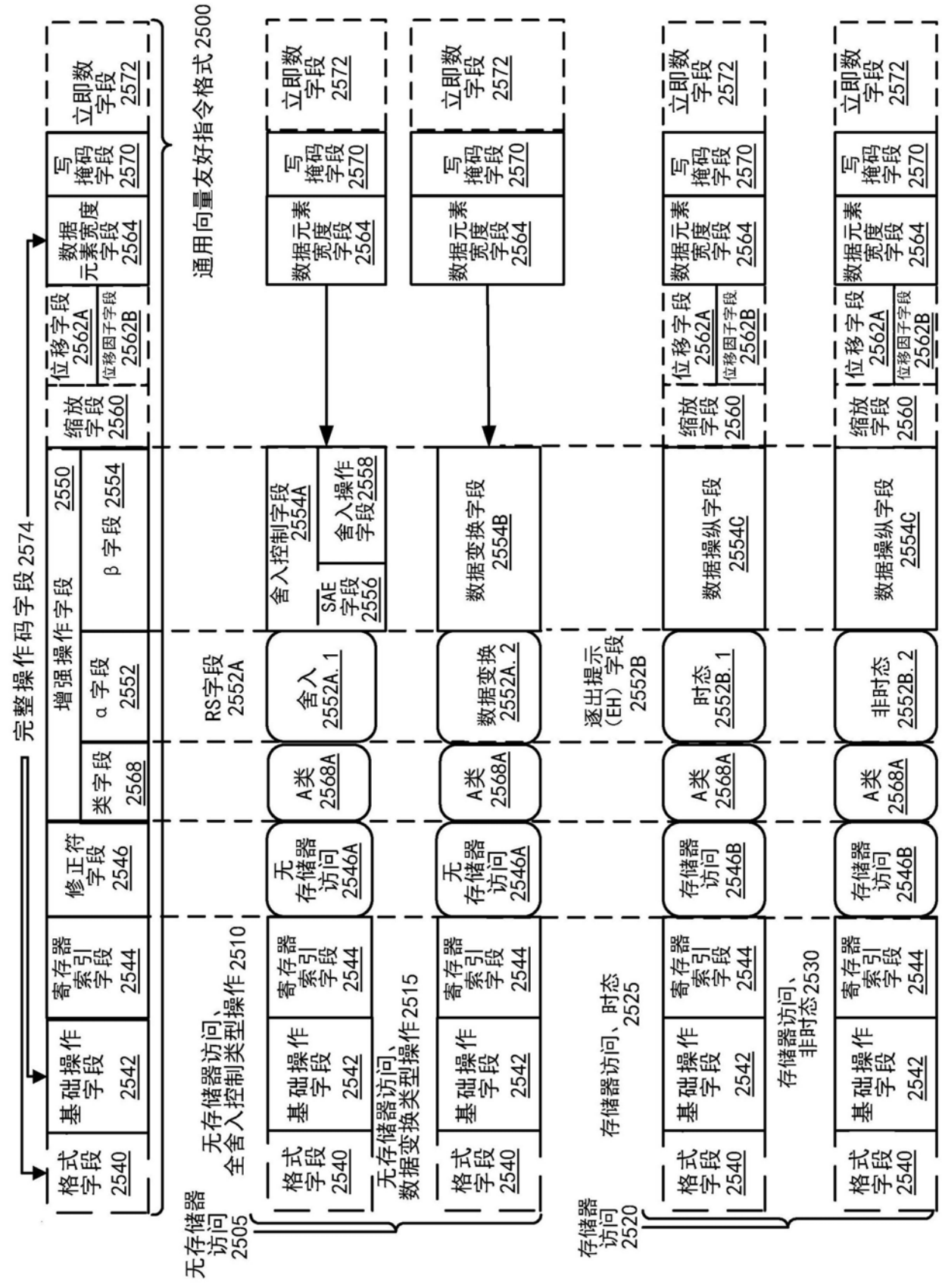

完整操作码字段 2574
通用向量友好指令格式 2500
格式字段 2540
基础操作字段 2542
寄存器索引字段 2544
修正符字段 2546
类字段 2568
增强操作字段 2550
α 字段 2552
β 字段 2554
缩放字段 2560
位移字段 2562A
位移因子字段 2562B
数据元素宽度字段 2564
写掩码字段 2570
立即数字段 2572
RS字段 2552A
舍入 2552A.1
数据变换 2552A.2
舍入控制字段 2554A
SAE 字段 2556
舍入操作字段2558
数据变换字段 2554B
数据操纵字段 2554C
逐出提示（EH）字段 2552B
时态 2552B.1
非时态 2552B.2
A类 2568A
存储器访问 2546B
无存储器访问 2546A
无存储器访问、全舍入控制类型操作 2510
无存储器访问、数据变换类型操作 2515
存储器访问、时态 2525
存储器访问、非时态 2530
无存储器访问 2505
存储器访问 2520

图25A

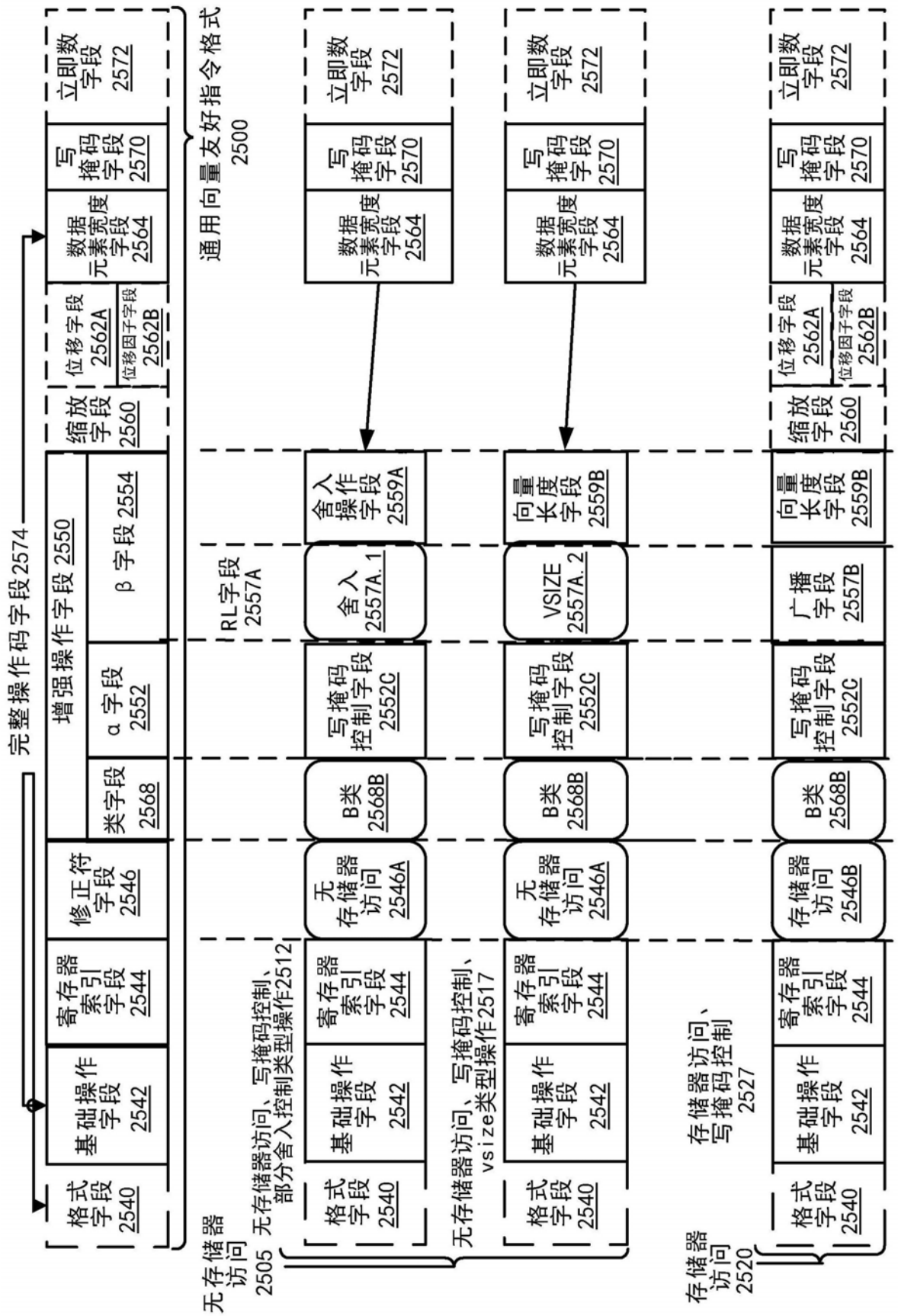

通用向量友好指令格式 2500
完整操作码字段2574
格式字段 2540
基础操作字段 2542
寄存器索引字段 2544
修正符字段 2546
类字段 2568
增强操作字段 2550
α 字段 2552
β 字段 2554
缩放字段 2560
位移字段 2562A
位移因子字段 2562B
数据元素宽度字段 2564
写掩码字段 2570
立即数字段 2572
无存储器访问 2505
无存储器访问、写掩码控制、部分舍入控制类型操作2512
无存储器访问 2546A
写掩码控制字段 2552C
B类 2568B
RL字段 2557A
舍入操作字段 2559A
舍入 2557A.1
无存储器访问、写掩码控制、vsize类型操作2517
VSIZE 2557A.2
向量长度字段 2559B
存储器访问 2520
存储器访问、写掩码控制 2527
存储器访问 2546B
广播字段 2557B

图25B

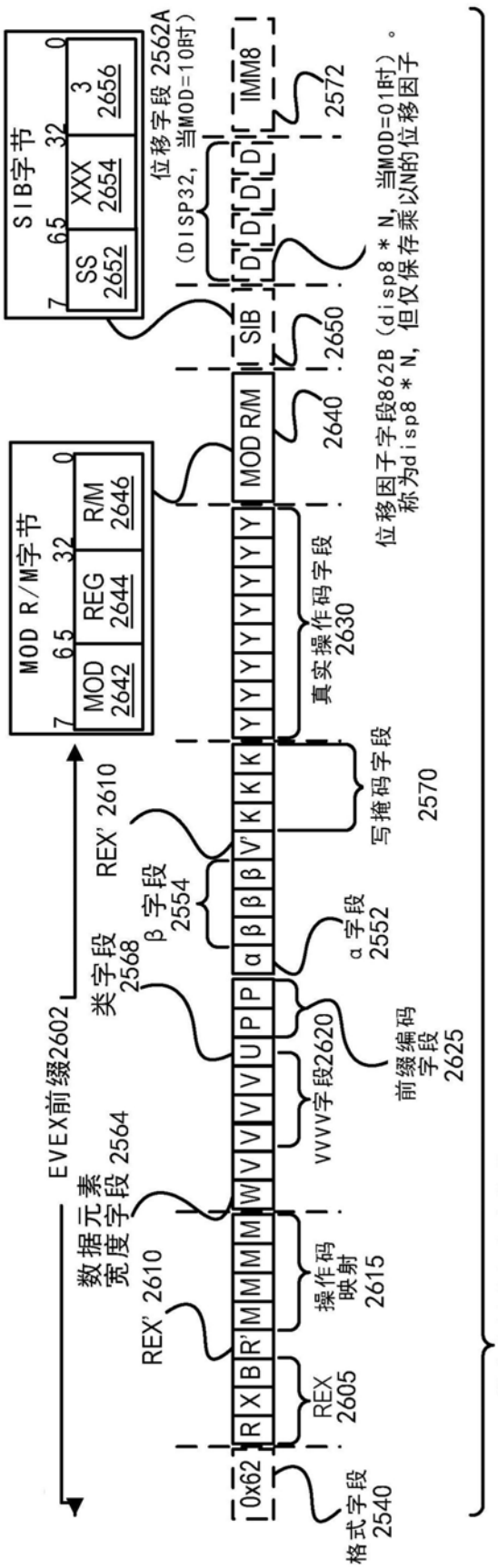

SIB字节
SS 2652
XXX 2654
3 2656
位移字段 2562A
（DISP32，当MOD=10时）
MOD R/M字节
MOD 2642
REG 2644
R/M 2646
SIB
MOD R/M
IMM8
2572
2650
2640
位移因子字段862B（disp8 * N，当MOD=01时）。
称为disp8 * N，但仅保存乘以N的位移因子
真实操作码字段
2630
EVEX前缀2602
类字段 2568
REX' 2610
β 字段 2554
写掩码字段
2570
数据元素宽度字段 2564
α 字段 2552
REX' 2610
前缀编码字段 2625
VVVV字段2620
操作码映射 2615
REX 2605
格式字段 2540
0x62
特定向量友好指令格式2600

图26A

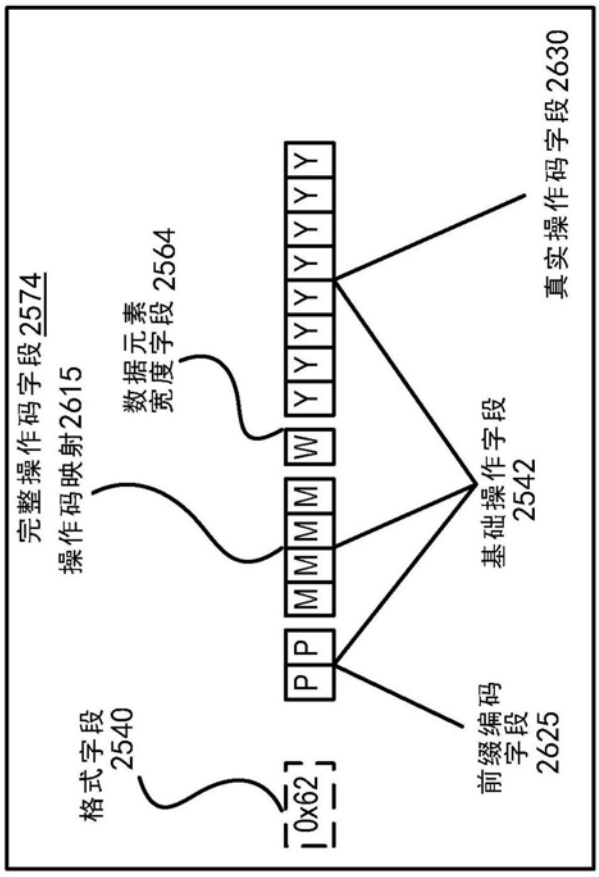

完整操作码字段 2574
操作码映射2615
格式字段
2540
0x62
P P
M M M
W
Y Y Y Y Y Y Y Y
数据元素
宽度字段 2564
前缀编码
字段
2625
基础操作字段
2542
真实操作码字段 2630

图26B

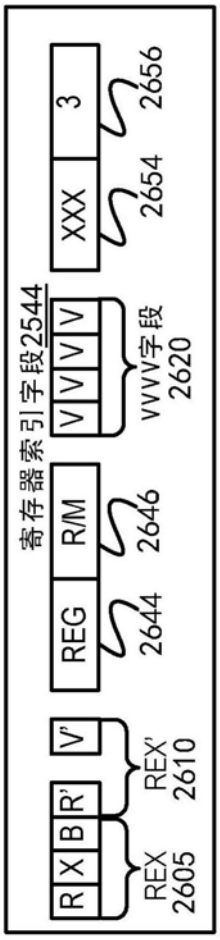

寄存器索引字段2544
R
X
B
R'
V'
REG
R/M
V
V
V
V
XXX
3
REX
2605
REX'
2610
2644
2646
VVVV字段
2620
2654
2656

图26C

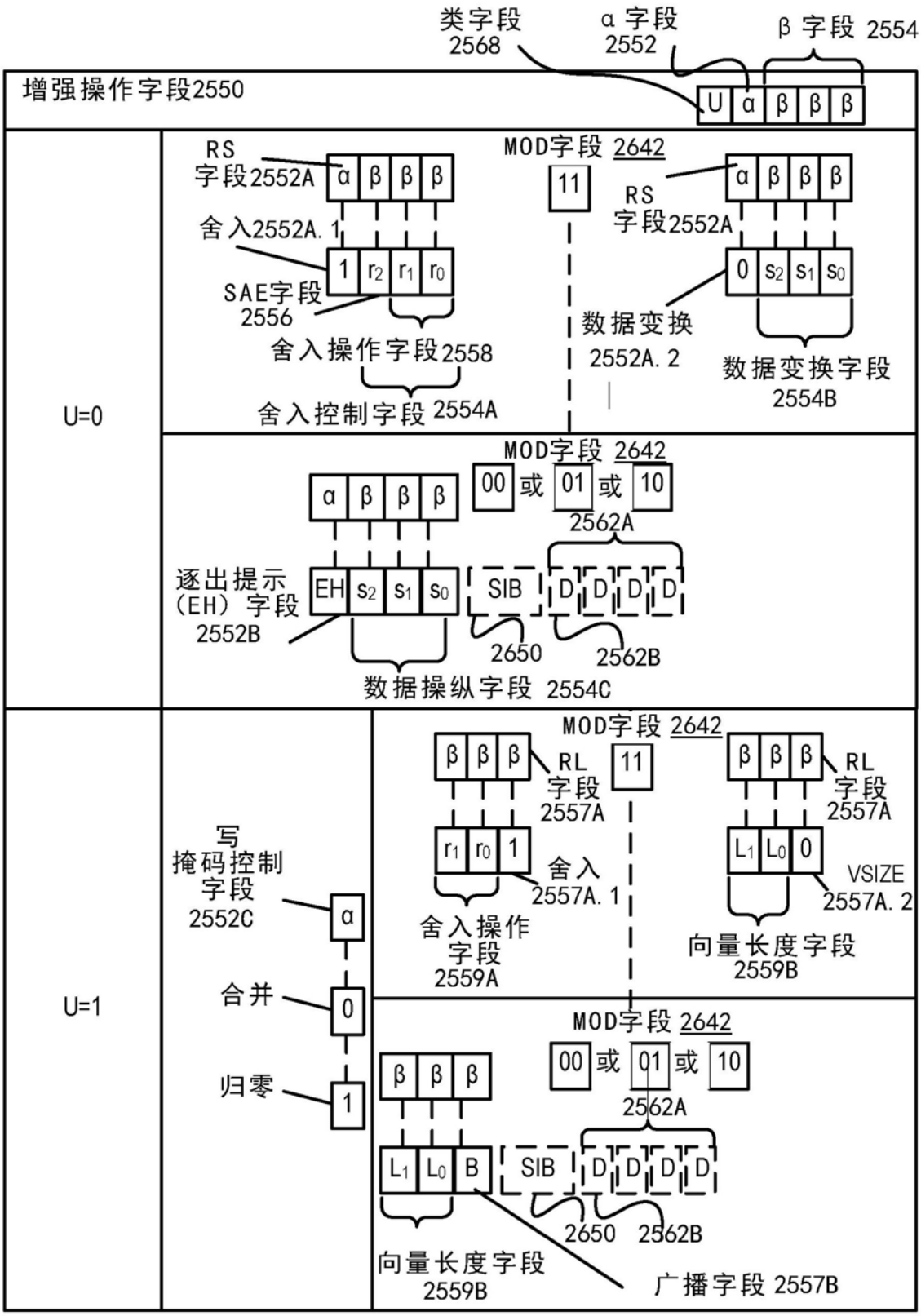

类字段 2568
α 字段 2552
β 字段 2554
增强操作字段2550
U α β β β
U=0
RS 字段2552A
α β β β
舍入2552A. 1
1 r2 r1 r0
SAE字段 2556
舍入操作字段2558
舍入控制字段 2554A
MOD字段 2642
11
RS 字段2552A
α β β β
0 s2 s1 s0
数据变换 2552A. 2
数据变换字段 2554B
MOD字段 2642
α β β β
00 或 01 或 10
2562A
逐出提示（EH）字段 2552B
EH s2 s1 s0
SIB
D D D D
2650
2562B
数据操纵字段 2554C
U=1
写掩码控制字段 2552C
α
合并
0
归零
1
MOD字段 2642
β β β
RL 字段 2557A
11
r1 r0 1
舍入 2557A. 1
舍入操作字段 2559A
β β β
RL 字段 2557A
L1 L0 0
VSIZE 2557A. 2
向量长度字段 2559B
MOD字段 2642
β β β
00 或 01 或 10
2562A
L1 L0 B
SIB
D D D D
2650
2562B
向量长度字段 2559B
广播字段 2557B

图26D

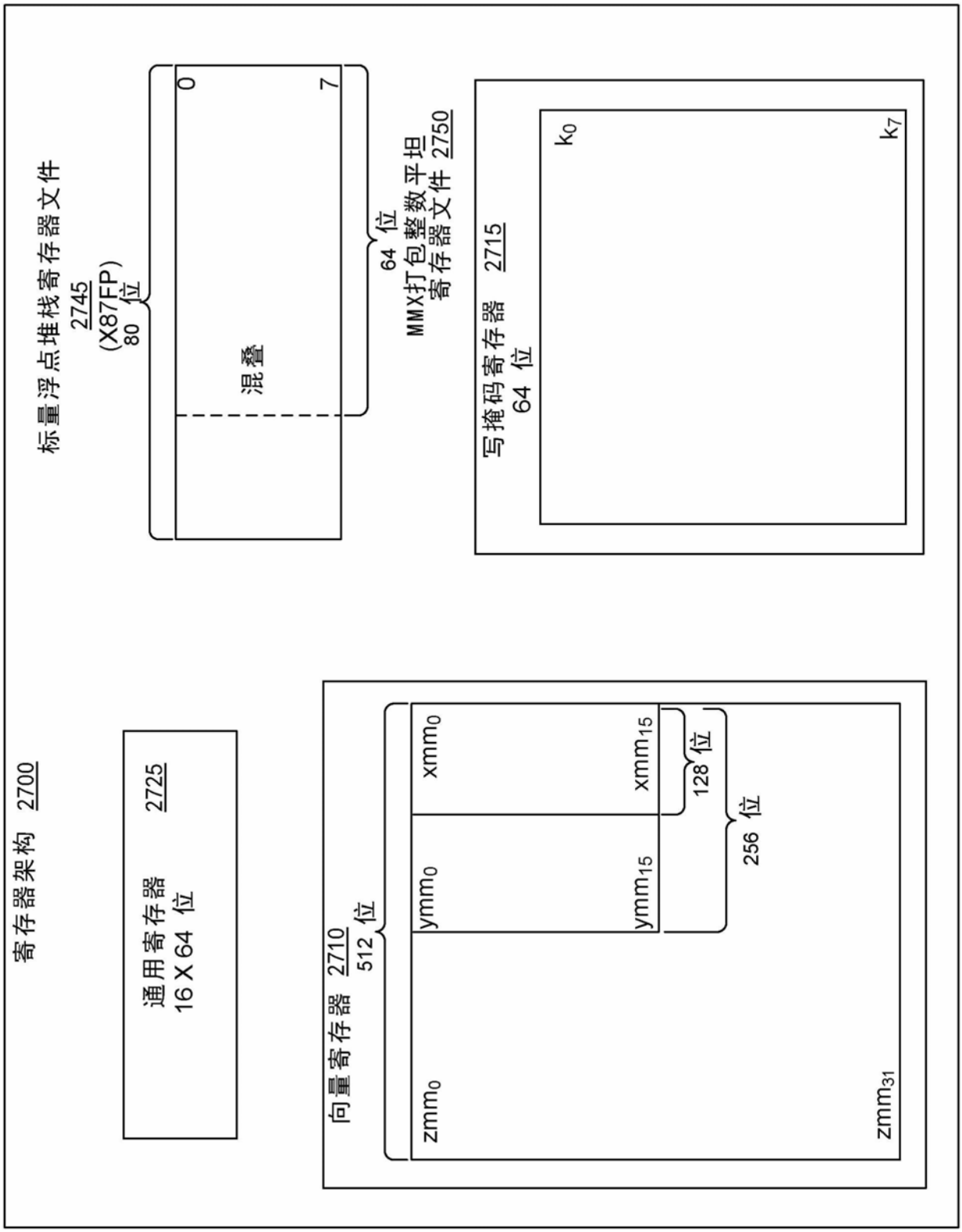

寄存器架构 2700
通用寄存器 2725
16 X 64 位
向量寄存器 2710
512 位
zmm0
zmm31
ymm0
ymm15
xmm0
xmm15
128 位
256 位
标量浮点堆栈寄存器文件 2745
(X87FP)
80 位
0
7
混叠
64 位
MMX打包整数平坦寄存器文件 2750
写掩码寄存器 2715
64 位
k0
k7

图27


流水线 2800
提取 2802
长度解码 2804
解码 2806
分配 2808
重命名 2810
调度 2812
寄存器读取/存储器读取 2814
执行级 2816
写回/存储器写入 2818
异常处理 2822
提交 2824

图28A

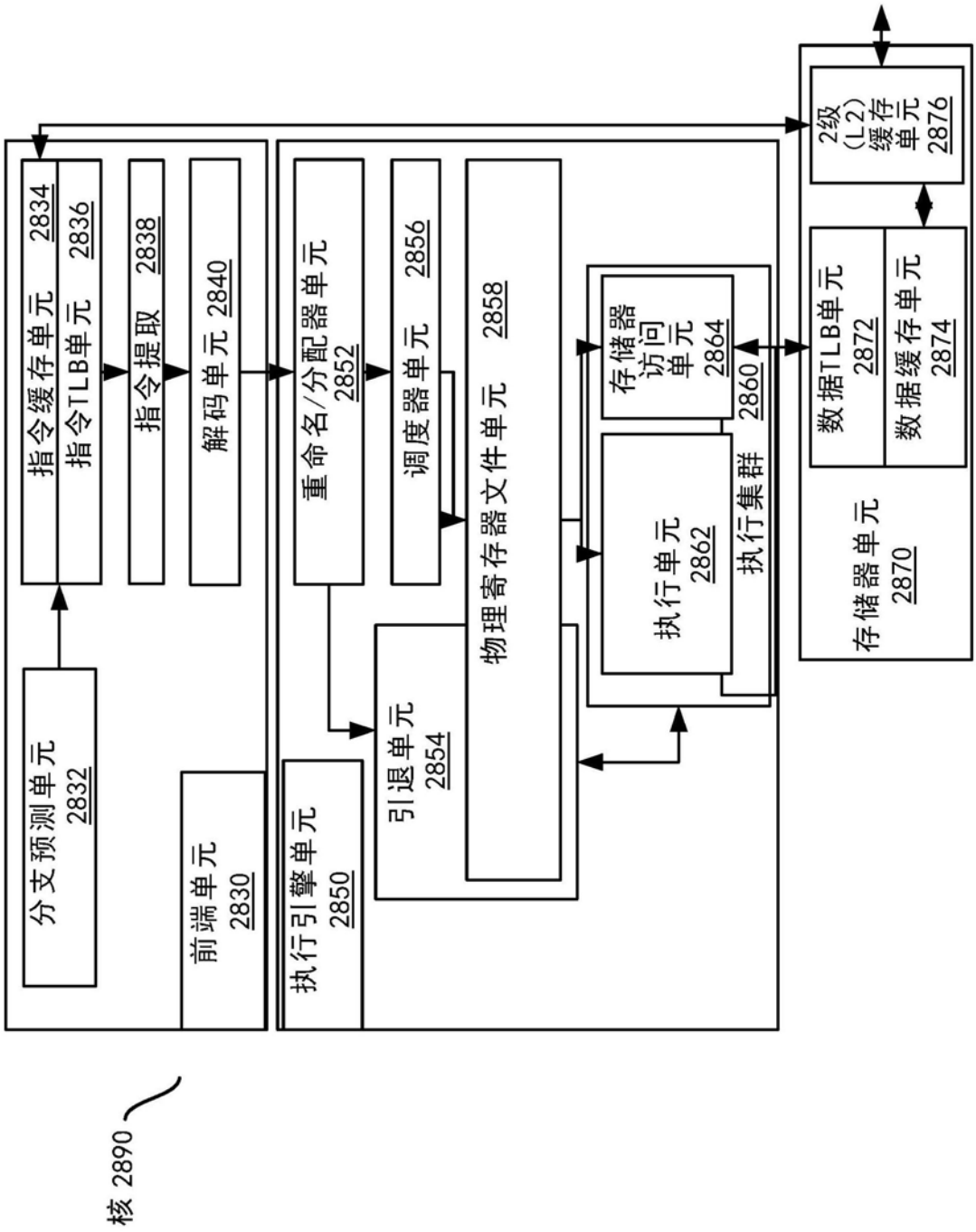

核 2890
分支预测单元 2832
指令缓存单元 2834
指令TLB单元 2836
指令提取 2838
解码单元 2840
前端单元 2830
执行引擎单元 2850
重命名/分配器单元 2852
引退单元 2854
调度器单元 2856
物理寄存器文件单元 2858
执行单元 2862
存储器访问单元 2864
执行集群 2860
存储器单元 2870
数据TLB单元 2872
数据缓存单元 2874
2级(L2)缓存单元 2876

图28B

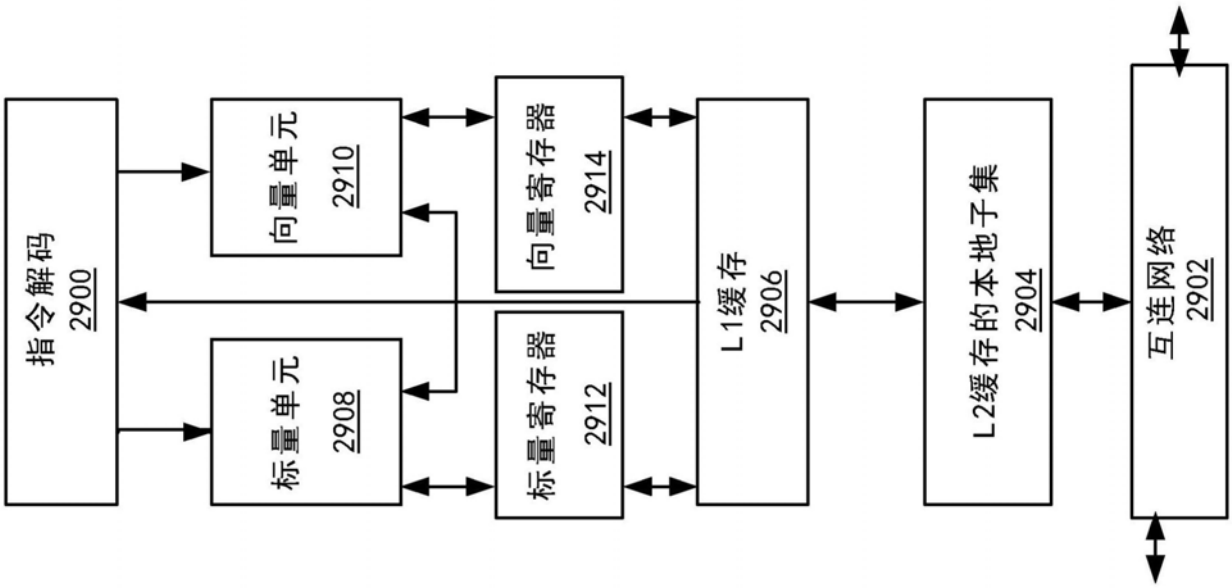

指令解码
2900
向量单元
2910
标量单元
2908
向量寄存器
2914
标量寄存器
2912
L1缓存
2906
L2缓存的本地子集
2904
互连网络
2902

图29A

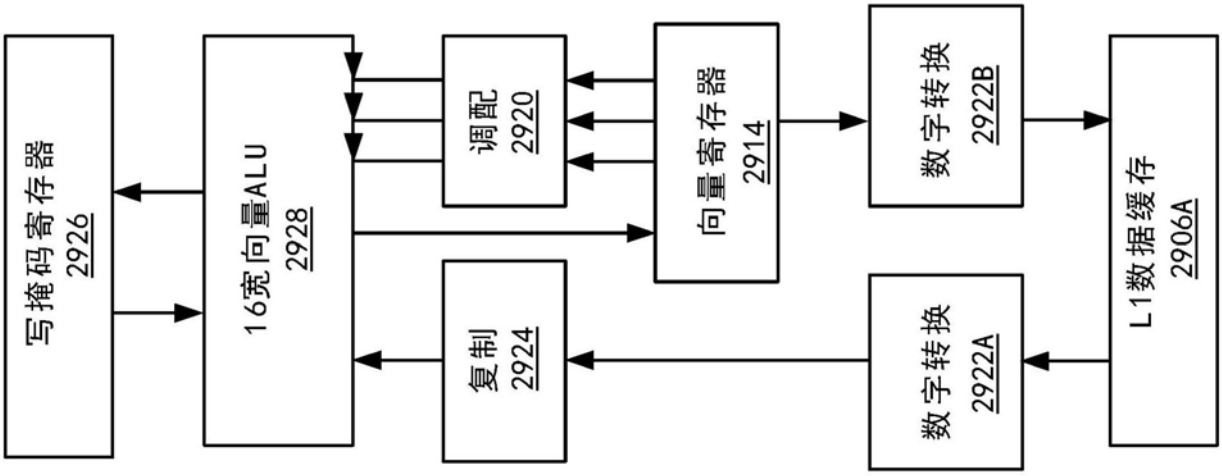

写掩码寄存器
2926
16宽向量ALU
2928
调配
2920
复制
2924
向量寄存器
2914
数字转换
2922B
数字转换
2922A
L1数据缓存
2906A

图29B

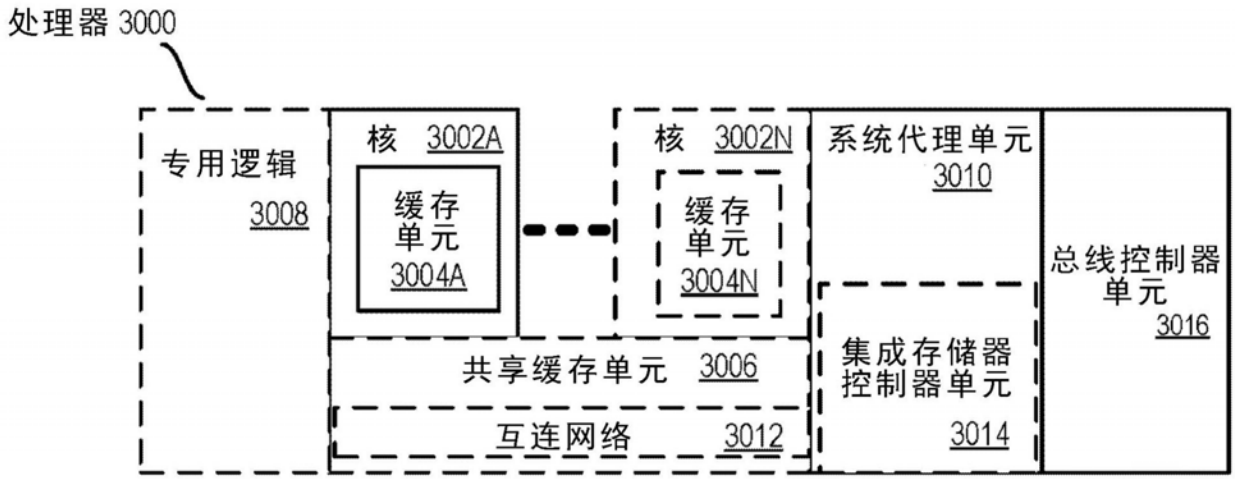

处理器 3000
专用逻辑
3008
核 3002A
缓存单元
3004A
核 3002N
缓存单元
3004N
系统代理单元
3010
总线控制器单元
3016
集成存储器控制器单元
3014
共享缓存单元 3006
互连网络 3012

图30

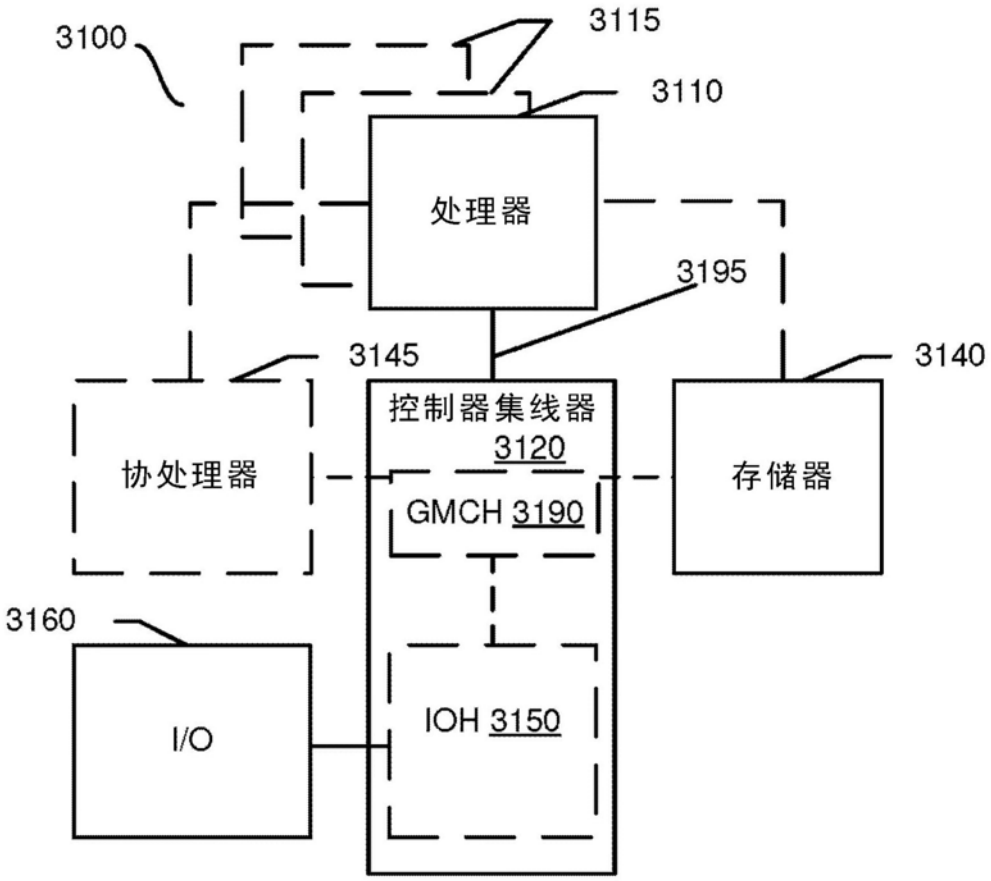

3100
3115
3110
处理器
3195
3145
3140
控制器集线器
3120
协处理器
GMCH 3190
存储器
3160
I/O
IOH 3150

图31

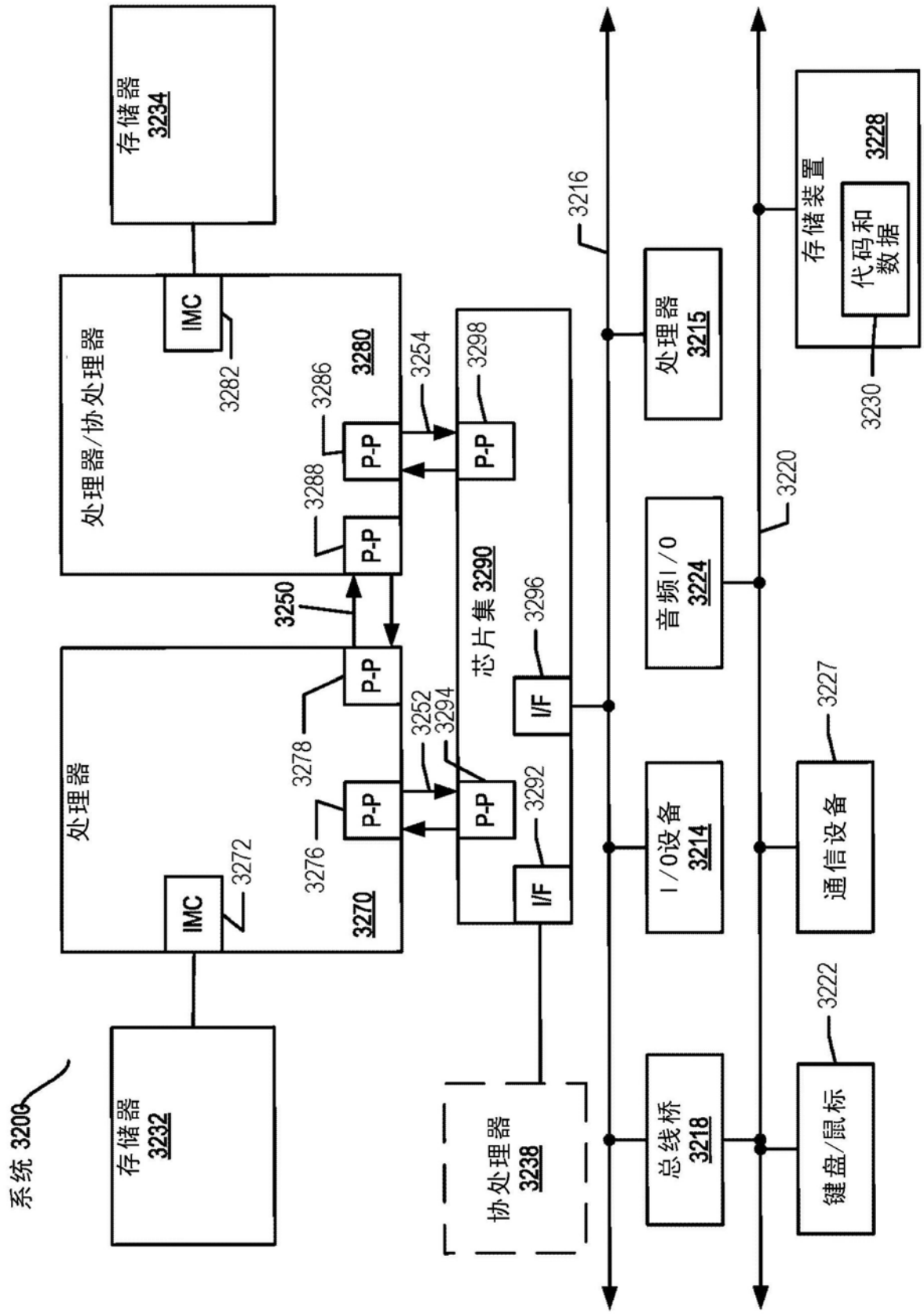

系统 3200
存储器 3232
处理器 3270
IMC 3272
处理器/协处理器
存储器 3234
IMC 3282
3280
P-P 3286
P-P 3288
3250
P-P 3278
P-P 3276
3252
3294
芯片集 3290
P-P 3254
P-P 3298
P-P 3292
I/F 3296
I/F
协处理器 3238
3216
处理器 3215
音频I/O 3224
I/O设备 3214
总线桥 3218
3220
存储装置 3228
代码和数据 3230
3227
通信设备
3222
键盘/鼠标

图32

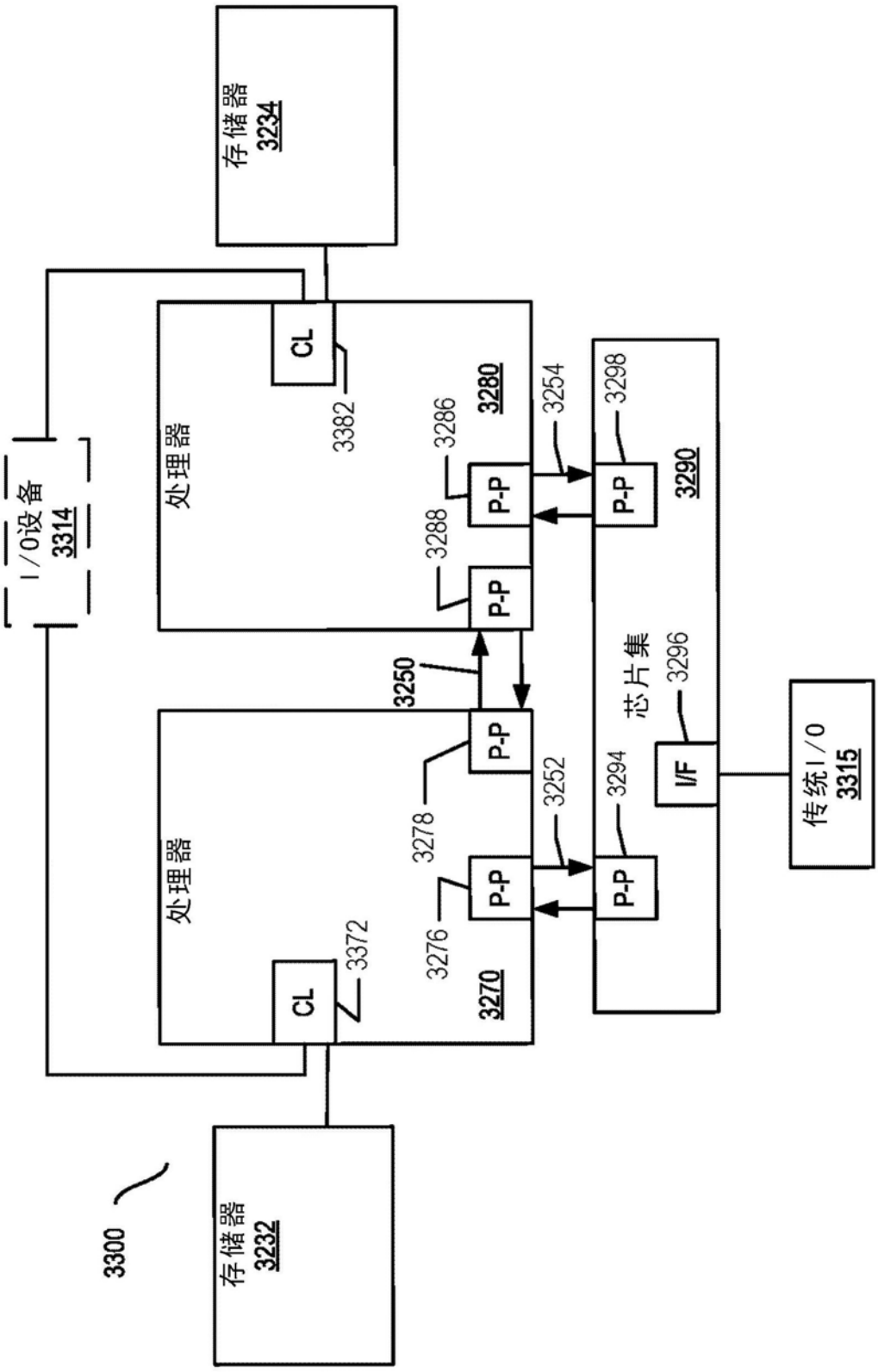

存储器
3234
CL
3382
处理器
3280
3286
3288
P-P
P-P
3254
3298
3290
P-P
I/O设备
3314
3250
芯片集
3296
I/F
传统I/O
3315
处理器
3278
P-P
3252
3294
P-P
3276
3270
CL
3372
3300
存储器
3232

图33

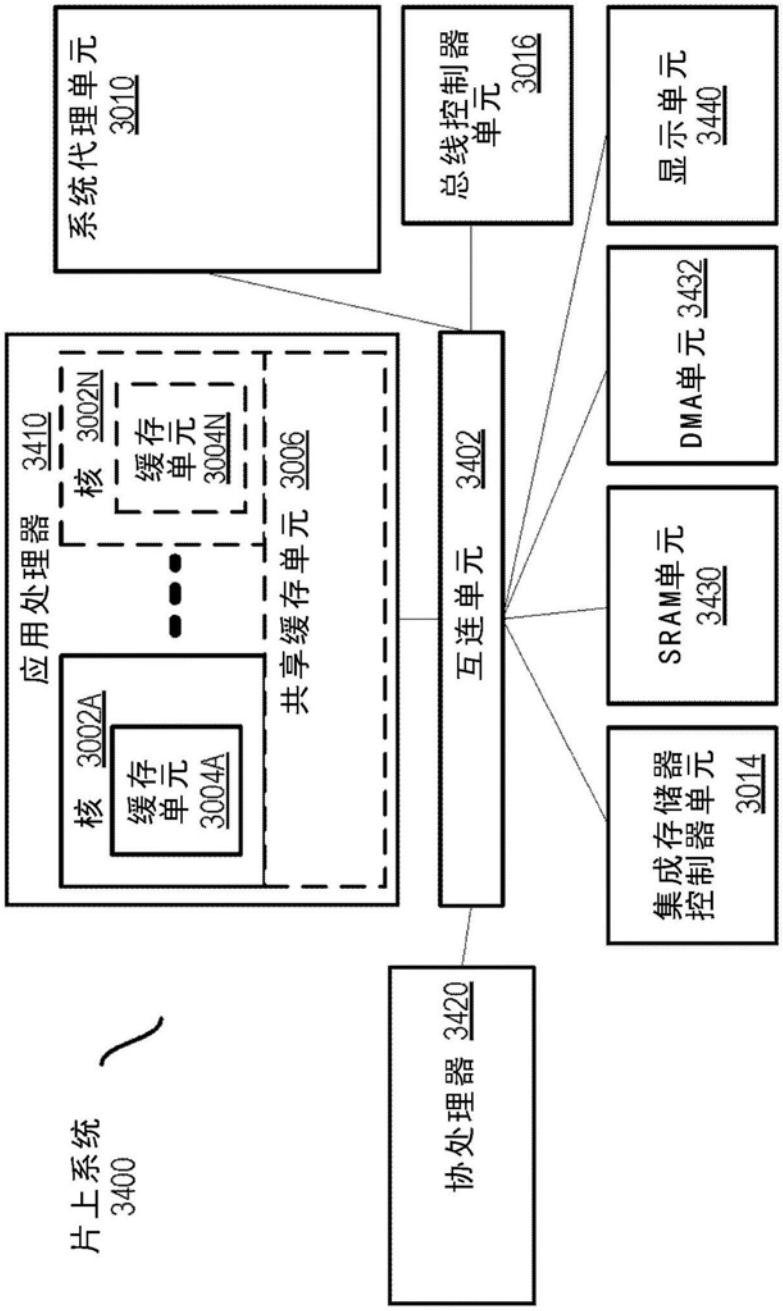

片上系统
3400
应用处理器 3410
核 3002A
缓存单元 3004A
核 3002N
缓存单元 3004N
共享缓存单元 3006
互连单元 3402
协处理器 3420
系统代理单元 3010
总线控制器单元 3016
显示单元 3440
DMA单元 3432
SRAM单元 3430
集成存储器控制器单元 3014

图34

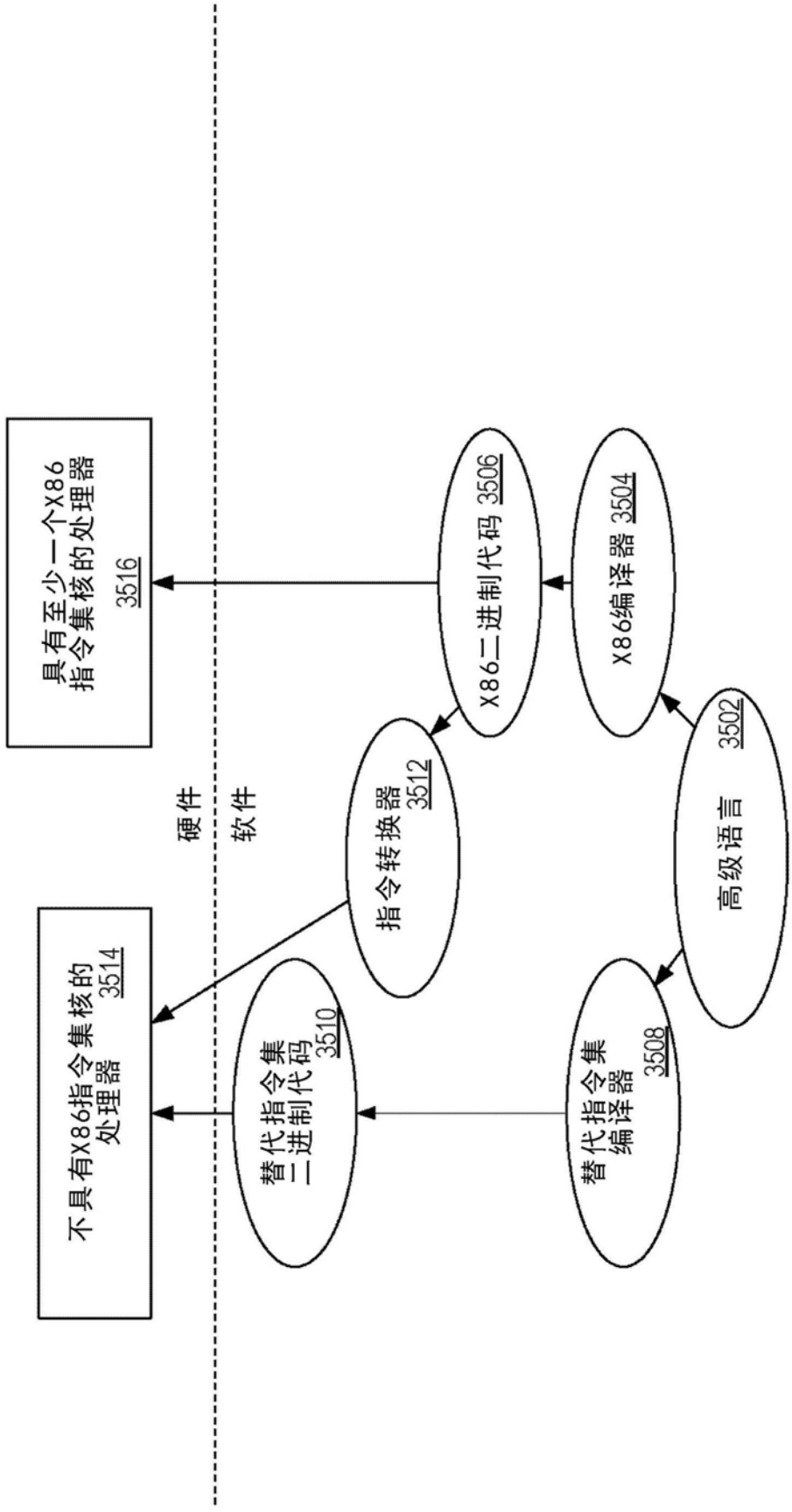

具有至少一个X86
指令集核的处理器
3516
不具有X86指令集核的
处理器 3514
硬件
软件
X86二进制代码 3506
X86编译器 3504
指令转换器
3512
高级语言 3502
替代指令集
二进制代码
3510
替代指令集
编译器
3508

图35