



(12) 发明专利申请

(10) 申请公布号 CN 118035625 A

(43) 申请公布日 2024. 05. 14

(21) 申请号 202410258938.7

(74) 专利代理机构 北京市金杜律师事务所

(22) 申请日 2018.06.20

11256

专利代理师 鄧迅

(30) 优先权数据

62/536,399 2017.07.24 US

15/710,433 2017.09.20 US

62/625,251 2018.02.01 US

15/920,156 2018.03.13 US

(62) 分案原申请数据

201880059404.4 2018.06.20

(71) 申请人 特斯拉公司

地址 美国得克萨斯州

(72) 发明人 D·达斯·萨玛 E·塔尔佩斯

P·J·班农

(51) Int.Cl.

G06F 17/16 (2006.01)

G06F 15/16 (2006.01)

G06F 15/163 (2006.01)

G06N 3/048 (2023.01)

G06N 3/063 (2023.01)

G06N 3/045 (2023.01)

G06F 15/80 (2006.01)

G06F 7/575 (2006.01)

G06F 7/544 (2006.01)

G06F 7/50 (2006.01)

G06F 7/523 (2006.01)

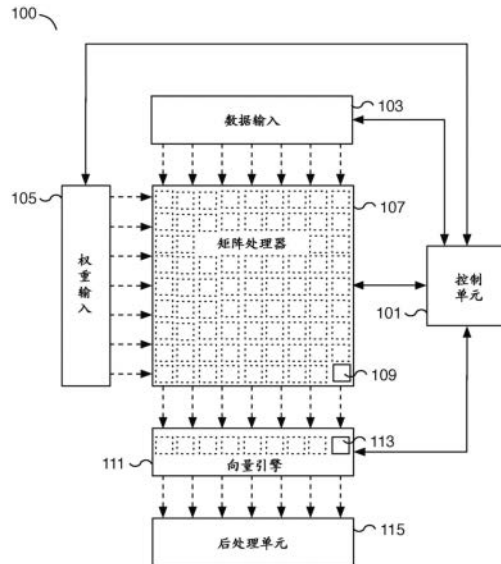
权利要求书2页 说明书24页 附图12页

(54) 发明名称

向量计算单元

(57) 摘要

一种微处理器系统包括计算阵列和向量计算单元。该计算阵列包括多个计算单元。该向量计算单元与计算阵列通信并且包括多个处理元件。处理元件被配置为从计算阵列接收输出数据元素,并且并行处理所接收的输出数据元素。



1. 一种微处理器系统,包括:

向量计算单元,包括多个处理元件,其中每个处理元件连接到计算阵列中包括的最后一行的多个计算单元中的相应计算单元,

其中所述计算单元被分组为多个通道,其中每个通道包括所述计算单元的子集,其中多个计算单元的至少子集被配置为并行接收一行数据元素,

其中所述通道并行操作,并且将所述一行数据元素移位通过所述通道到所述向量计算单元,使得与所述一行数据元素相关联的信息被并行移位通过所述通道,并且所述信息从所述最后一行的所述计算单元并行地被提供到相应处理元件;以及

控制单元电路,被配置为向向量计算单元提供至少单个处理器指令,所述控制单元电路被配置为将来自所述多个通道的所述数据元素的接收同步到相应处理元件,

其中所述单个处理器指令指定至少三个不同的组件指令,所述向量计算单元响应于所述单个处理器指令而执行所述组件指令,并且所述向量计算单元的所述多个处理元件中的每个处理单元被配置为响应于所述单个处理器指令而处理接收到的所述信息。

2. 根据权利要求1所述的系统,其中所述计算单元被配置为在单个周期内执行乘法、加法运算和移位运算。

3. 根据权利要求1所述的系统,其中所述计算单元被布置在瓦片中,其中每个瓦片包括多个计算单元,使得所述多个通道中的每个通道包括的垂直布置的一个或多个瓦片。

4. 根据权利要求3所述的系统,其中每个瓦片包括计算单元矩阵,使得所述多个通道中的每个通道包括一个或多个计算单元矩阵。

5. 根据权利要求4所述的系统,其中响应于所述单个处理器指令,所述计算单元矩阵的尺寸被动态配置。

6. 根据权利要求1所述的系统,其中所述至少三个不同的组件指令包括:编码的存储器访问操作组件指令和编码的算术逻辑单元操作组件指令。

7. 根据权利要求6所述的系统,其中所述编码的存储器访问操作组件指令是编码的加载操作组件指令或编码的存储操作组件指令。

8. 根据权利要求1所述的系统,其中针对所述向量计算单元的特定时钟周期,与第一组件指令相关联的加载操作、与第二组件指令相关联的算术逻辑单元操作以及与第三组件指令相关联的存储操作被并行执行。

9. 根据权利要求1所述的系统,其中所述向量计算单元被配置为并行处理所述至少三个不同的组件指令中的每个组件指令的执行阶段。

10. 根据权利要求1所述的系统,其中所述向量计算单元包括多个向量寄存器、控制逻辑、输入缓冲器和输出缓冲器,其中所述向量寄存器、所述控制逻辑、所述输入缓冲器或者输出缓冲器中的至少一者被配置为基于所述向量寄存器的大小来混叠所述向量寄存器。

11. 根据权利要求1所述的系统,其中所述至少三个不同的组件指令之一引用所述向量计算单元的一个或多个向量寄存器。

12. 根据权利要求1所述的系统,其中所述向量计算单元包括一个或多个混叠的向量寄存器。

13. 根据权利要求12所述的系统,其中所述一个或多个混叠的向量寄存器包括混叠的8位向量寄存器、混叠的16位向量寄存器或混叠的32位向量寄存器。

14. 根据权利要求1所述的系统,其中所述至少三个不同的组件指令之一引用三个源寄存器和一个目的地寄存器。

15. 根据权利要求1所述的系统,其中所述单处理器指令对向量屏蔽移动指令进行编码。

16. 根据权利要求1所述的系统,其中所述至少三个不同的组件指令之一包括:向量位屏蔽、寄存器大小字段、屏蔽位或立即有效位。

17. 根据权利要求1所述的系统,其中基于所述最后一行的所述计算单元被配置为使所述一行数据元素出队,出队准备好的信号被所述向量计算单元接收以指示所述计算阵列被配置为接收队列操作。

18. 一种方法,包括:

从控制单元电路接收单个处理器指令,其中所述单个处理器指令指定多个组件指令;

将所述单个处理器指令解码为所述多个组件指令;

使用包括多个处理元件的向量计算单元来执行所述组件指令,所述处理元件连接到被包括在计算阵列中的多个计算单元中的相应计算单元,

其中所述计算单元被分组为多个通道,其中每个通道包括所述计算单元的子集,其中所述多个计算单元的至少子集被配置为并行接收一行数据元素,

其中所述通道并行操作,并将所述一行数据元素移位通过所述通道到所述向量计算单元,使得与所述一行数据元素相关联的信息被并行移位通过所述通道,并且所述信息从最后一行的计算单元并行地被提供到相应处理元件;以及

使用所述多个处理元件中的每个处理元件来处理接收到的相应信息,

其中所述控制单元电路被配置为将来自所述多个通道的所述数据元素的接收同步到相应处理元件。

19. 根据权利要求18所述的方法,其中所述计算单元被布置在瓦片中,其中每个瓦片包括多个计算单元,使得所述多个通道中的每个通道包括的垂直布置的一个或多个瓦片。

20. 根据权利要求19所述的方法,其中每个瓦片包括计算单元矩阵,使得所述多个通道中的每个通道包括一个或多个计算单元矩阵。

向量计算单元

[0001] 本申请是申请号为201880059404.4、申请日为2018年6月20日、题为“向量计算单元”的中国发明专利申请的分案申请。

[0002] 相关申请的交叉引用

[0003] 本申请要求于2018年2月1日提交的题为“VECTORCOMPUTATIONAL UNIT”的第62/625,251号美国临时专利申请的优先权,并且要求于2017年7月24日提交的题为“ACCELERATEDMATHEMATICAL ENGINE”的第62/536,399号美国临时专利申请的优先权,并且是于2017年9月20日提交的题为“ACCELERATEDMATHEMATICAL ENGINE”的共同未决的第15/710,433号美国专利申请的部分继续申请,该申请要求于2017年7月24日提交的题为“ACCELERATED MATHEMATICAL ENGINE”的第62/536,399号美国临时专利申请的优先权,其全部通过引用并入本文以用于所有目的。

背景技术

[0004] 用于机器学习和人工智能的处理通常需要对大型数据的集合执行数学操作,并且通常包括求解多重卷积层和池化层。机器学习和人工智能技术通常利用矩阵操作和非线性函数(诸如激活函数)。机器学习的应用包括自动驾驶汽车和驾驶员辅助汽车。在一些场景中,计算机处理器用于执行机器学习训练和推理。传统计算机处理器能够非常快速地执行单个数学操作,但是通常只能同时地对有限数量的数据进行操作。作为备选方案,可以利用图形处理单元(GPU),并且该GPU能够执行相同数学操作,但是可以对更大数据的集合执行并行数学操作。通过利用多个处理器核,GPU可以并行执行多个任务,并且通常能够完成比传统计算机处理器更快利用并行性的大型图形处理任务。然而,GPU和传统计算机处理器均非最初为机器学习或人工智能操作而设计。机器学习和人工智能操作通常依赖于对非常大的数据集重复应用特定机器学习处理器操作的集合。因此,需要一种微处理器系统,其支持对大型数据集并行执行机器学习和人工智能特定处理操作,而无需对于每个并行操作的多个处理核心的开销。

发明内容

[0005] 本公开的实施例涉及微处理器系统,包括计算阵列和向量计算单元。该计算阵列包括多个计算单元。该向量计算单元与计算阵列通信并且包括多个处理元件。处理元件被配置为从计算阵列接收输出数据元素,并且并行处理所接收的输出数据元素。

附图说明

[0006] 在以下具体实施方式和附图中公开了本发明的各种实施例。

[0007] 图1是图示了用于执行机器学习处理的微处理器系统的实施例的框图。

[0008] 图2是图示了用于执行机器学习处理的微处理器系统的实施例的框图。

[0009] 图3是图示了用于执行机器学习处理的微处理器系统的实施例的框图。

[0010] 图4A是图示了用于执行机器学习处理的向量计算单元的实施例的框图。

- [0011] 图4B是图示了示例性的向量寄存器的混叠的表。
- [0012] 图5是图示了对于确定微处理器系统的处理器指令的过程的实施例的流程图。
- [0013] 图6A是图示了对于向量计算单元的运行执行的过程的实施例的流程图。
- [0014] 图6B是图示了对于由向量计算单元处理向量数据的过程的实施例的流程图。
- [0015] 图7是图示了用于向量计算单元指令的编码格式的实施例的框图。
- [0016] 图8是图示了对于由向量计算单元执行单个向量计算单元指令的过程的实施例的流程图。
- [0017] 图9是图示了向量计算单元的示例性指令周期的图。
- [0018] 图10是图示了计算阵列的计算单元的实施例的框图。

具体实施方式

[0019] 本发明可以以多种方式实现,其包括被实现为过程;装置;系统;物质的组合;体现在计算机可读存储介质上的计算机程序产品;和/或处理器,诸如被配置为执行存储在耦合到该处理器的存储器上和/或由耦合到该处理器的存储器提供的指令的处理器。在本说明书中,这些实现方式或本发明可以采用的任何其他形式可以被称为技术。一般而言,在本发明的范围内,可以更改所公开的过程的步骤的次序。除非另有说明,否则被描述为被配置为执行任务的诸如处理器或存储器的组件可以被实现为通用组件或特定组件,该通用组件被临时配置为在给定时间执行任务,该特定组件被制造为执行任务。如本文中所使用的,术语“处理器”是指被配置为处理诸如计算机程序指令的数据的一个或多个设备、电路和/或处理核心。

[0020] 下文提供了对本发明的一个或多个实施例的详细描述以及说明本发明原理的附图。结合这样的实施例对本发明进行了描述,但是本发明不限于任何实施例。本发明的范围仅由权利要求书限制,并且本发明涵盖许多备选方案、修改和等同物。在以下描述中阐述了许多具体细节,以便提供对本发明的透彻理解。出于示例的目的而提供这些细节,并且可以根据权利要求来实践本发明,而无需这些具体细节中的一些或所有具体细节。为了清楚起见,尚未对与本发明有关的技术领域中所已知的技术材料进行详细描述,从而不会不必要地使本发明晦涩难懂。

[0021] 公开了一种利用向量计算单元和向量计算单元指令集合架构的微处理器系统。例如,微处理器系统包括与向量计算单元通信的计算阵列。在各种实施例中,计算阵列是能够对两个输入向量执行算术操作的矩阵处理器,并且包括接收来自输入向量的M个操作数和N个操作数的多个计算单元。在一些实施例中,计算单元是包括算术逻辑单元、累加器和影子寄存器的子电路,其用于执行诸如生成点积和执行用于卷积的各种处理的操作。与传统图形处理单元 (GPU) 或中央处理单元 (CPU) 处理核心不同,其中每个核心被配置为接收其自己的唯一处理指令,计算阵列的计算单元各自响应于由计算阵列所接收的单个指令而并行执行相同计算。在各种实施例中,向量计算单元包括多个处理元件,其用于对输入数据的向量并行执行加载操作、算术操作和存储操作。向量计算单元的处理元件被配置为接收来自计算阵列的输出。在各种实施例中,计算阵列的输出和到向量计算单元的输入是数据的阵列。所接收到的向量计算单元的输入是响应于单个处理器指令而被并行处理的。与计算阵列类似,向量计算单元的处理元件各自响应于由向量计算单元所接收的各个指令而并行执行相

同的计算。在一些实施例中,微处理器系统还包括控制单元,其被配置为向向量计算单元提供指令。每个单个处理器指令可以指定要由向量计算单元执行的多个组件指令。响应于单个指令,向量计算单元的多个处理元件中的每个处理元件与其他处理元件并行处理向量输入的不同数据元素。在一些实施例中,向量计算单元的输出被馈送到后处理单元以用于执行诸如池化操作的后处理。

[0022] 在一些实施例中,微处理器系统至少包括计算阵列和向量计算单元。例如,计算阵列可通信地连接到向量计算单元,使得计算阵列的输出作为输入被馈送到向量计算单元。在各种实施例中,计算阵列包括多个计算单元。例如,计算单元可以是矩阵处理器的子电路,其包括用于执行一个或多个乘法操作、加法操作和移位操作的功能。作为另一示例,计算单元可以是包括用于执行点积操作的功能的子电路。在各种实施例中,计算阵列包括足够数目的计算单元,用于对数据输入并行执行多个操作。例如,被配置为接收M个操作数和N个操作数的计算阵列可以包括至少 $M \times N$ 个计算单元。在各种实施例中,微处理器系统还包括控制单元,该控制单元用于在计算阵列与向量计算单元之间的协调处理。例如,控制单元可以协调来自存储器的要被馈送到计算阵列中的数据、来自计算阵列的要被馈送到向量计算单元中的数据和/或来自向量计算单元的要被存储在存储器中或馈送到后处理单元中的数据。在一些实施例中,控制单元被配置为向计算阵列提供计算阵列指令,向向量计算单元提供向量计算单元指令,和/或向后处理单元提供后处理指令。

[0023] 在一些实施例中,与计算阵列通信的向量计算单元包括多个处理元件,该多个处理元件被配置为接收来自计算阵列的输出数据元素作为输入。例如,向量计算单元(诸如向量引擎)接收用于处理的向量作为输入。向量计算单元可以包括用于输入向量的每个元素的处理元件。被配置为接收N个元素(或操作数)的向量的示例向量计算单元可以包括用于并行处理N个元素的N个处理元件。在各种实施例中,处理元件被配置为接收来自计算阵列的输出数据元素。例如,来自计算阵列的输出可以是被馈送以由向量计算单元的处理元件接收的数据元素的向量。在各种实施例中,每个向量计算单元响应于单个处理器指令而并行处理所接收的来自计算阵列的输出数据元素。例如,单个处理器指令适用于要在对应数据元素上执行的向量计算单元的处理元件中的每个处理元件。

[0024] 在一些实施例中,控制单元被配置为向向量计算单元提供至少单个处理器指令。单个处理器指令指定要由向量计算单元(例如,响应于单个处理器指令)执行的多个组件指令。例如,控制单元向向量计算单元提供单个向量指令,诸如指令三元组,其包括多个组件指令。在一些实施例中,指令三元组是简单的处理器指令,其包括多达三个组件指令,诸如单独的加载指令、算术逻辑单元(ALU)指令、以及存储指令。由向量计算单元接收并且(例如,响应于指令三元组)执行这三个组件指令。例如,接收捆绑了加载指令、ALU指令和存储指令的指令三元组的向量计算单元执行加载指令、算术指令和存储指令。在各种实施例中,响应于单个处理器指令,向量计算单元的多个处理元件被配置为与其他处理元件并行处理不同的数据元素。例如,每个处理元件能够并行处理与向量计算单元的输入向量不同的数据元素。作为另一示例,单个向量处理器指令三元组的组件指令中的每个组件指令可以被应用于向量输入的元素中的每个元素,以使用向量计算单元完成N个元素的整个输入向量的并行处理。

[0025] 图1是图示了用于执行机器学习处理的微处理器系统的实施例的框图。在所示的

示例中,微处理器系统100包括控制单元101、数据输入103、权重输入105、矩阵处理器107、向量引擎111和后处理单元115。数据输入103和权重输入105是用于准备用于矩阵处理器107的数据的输入模块。在一些实施例中,数据输入103和权重输入105各自都包括输入数据格式化器、高速缓存或缓冲器、和/或用于准备用于矩阵处理器107的数据的逻辑电路。例如,数据输入103可以准备与图像数据相对应的来自二维阵列的N个操作数,以及权重输入105可以准备与要由矩阵处理器107处理的权重值的向量相对应的M个操作数。在一些实施例中,执行图5的过程以准备用于在微处理器系统100上操作的指令,这些指令包括用于矩阵处理器107的矩阵处理器指令和用于向量引擎111的向量引擎指令。在一些实施例中,包括向量引擎111的微处理器系统100执行下文关于图6A,图6B和图8所描述的过程。

[0026] 在一些实施例中,矩阵处理器107是包括多个计算单元的计算阵列。例如,分别从权重输入105和数据输入103接收M个操作数和N个操作数的矩阵处理器包括 $M \times N$ 个计算单元。在所示的图中,矩阵处理器107内部的小方块描绘了矩阵处理器107包括计算单元的二维阵列。计算单元109是矩阵处理器107的多个计算单元中的一个计算单元。在一些实施例中,每个计算单元被配置为从数据输入103接收一个操作数,并且从权重输入105接收一个操作数。在一些实施例中,根据逻辑二维阵列配置计算单元,但是矩阵处理器不一定使用布局为物理二维阵列的计算单元制作。例如,数据输入103的第i个操作数和权重输入105的第j个操作数被配置为由矩阵处理器107的第 $i \times j$ 个计算单元进行处理。

[0027] 在各种实施例中,组件数据输入103、权重输入105、矩阵处理器107、向量引擎111和后处理单元115的数据宽度是宽数据宽度,并且包括并行传送多于一个操作数的能力。在一些实施例中,数据输入103和权重输入105的宽度各自为96个字节。在一些实施例中,数据输入103的宽度为192个字节,而权重输入105的宽度为96个字节。在各种实施例中,数据输入103和权重输入105的宽度是动态地可配置的。例如,数据输入103可以被动态地配置为96或192个字节,而权重输入105可以被动态地配置为96或48个字节。在一些实施例中,动态配置由控制单元101控制。在各种实施例中,96个字节的数据宽度允许对96个操作数进行并行处理。例如,在数据输入103的宽度被配置为96个字节的实施例中,数据输入103可以将96个操作数并行传送到矩阵处理器107。

[0028] 在各种实施例中,矩阵处理器107被配置为从数据输入103接收N个字节并且从权重输入105接收M个字节,并且包括至少 $M \times N$ 个计算单元。例如,矩阵处理器107可以被配置为从数据输入103接收96个字节并且从权重输入105接收96个字节,并且包括至少 96×96 个计算单元。作为另一示例,矩阵处理器107可以被配置为从数据输入103接收192个字节并且从权重输入105接收48个字节,并且包括至少 192×48 个计算单元。在各种实施例中,可以对矩阵处理器107的尺寸进行动态地配置。例如,矩阵处理器107的默认尺寸可以被配置为从数据输入103接收96个字节并且从权重输入105接收96个字节,但是输入尺寸可以分别被动态地配置为192个字节和48个字节。在各种实施例中,每个计算单元的输出大小等于或大于输入大小。例如,在一些实施例中,每个计算单元的输入是两个1字节操作数,一个与来自数据输入103的操作数相对应,而另一个与来自权重输入105的操作数相对应,处理这两个操作数的输出是4字节结果。作为另一示例,矩阵处理器107可以被配置为从数据输入103接收96个字节并且从权重输入105接收96个字节,并且输出96个4字节结果。在一些实施例中,矩阵处理器107的输出是向量。例如,被配置为接收宽度为96个字节的两个输入向量(其中输

入向量的每个元素(或操作数)的大小为一个字节)的矩阵处理器可以输出宽度为96个字节的向量结果(其中向量结果的每个元素的大小为4个字节)。

[0029] 在各种实施例中,矩阵处理器107的每个计算单元是包括算术逻辑单元、累加器和影子寄存器的子电路。在所示的示例中,矩阵处理器107的计算单元可以分别对来自权重输入105和数据输入103的M个操作数和N个操作数执行算术操作。在各种实施例中,每个计算单元被配置为执行一个或多个乘法操作、加法操作、累加操作和/或移位操作。在一些实施例中,每个计算单元被配置为执行点积操作。例如,在一些实施例中,计算单元可以执行多个点积分量操作以计算点积结果。例如,矩阵处理器107的计算单元的阵列可以用于执行使用机器学习模型执行推理所需的卷积步骤。可以格式化二维数据集合(诸如图像),并且使用数据输入103将其馈送到矩阵处理器107中,一次一个向量。并行地,权重向量可以使用权重输入105通过对权重进行格式化并且将其作为向量馈送到矩阵处理器107中来应用于二维数据集合。矩阵处理器107的对应计算单元对权重输入和数据输入的对应操作数并行执行矩阵处理器指令。

[0030] 在一些实施例中,向量引擎111是通信耦合到矩阵处理器107的向量计算单元。向量引擎111包括多个处理元件,其包括处理元件113。在所示的图中,向量引擎111内的小方块描绘了向量引擎111包括被布置为向量的多个处理元件。在一些实施例中,处理元件以与数据输入103相同的方向被布置在向量中。在一些实施例中,处理元件以与权重输入105相同的方向被布置在向量中。向量引擎111的处理元件的大小等于或大于矩阵处理器107的计算单元的数据大小。例如,在某些实施例中,计算单元109接收两个操作数,每个操作数的大小为1字节,并且输出大小为4个字节的的结果。处理元件113从计算单元109接收4字节结果作为大小为4个字节的输入。在各种实施例中,向量引擎111的输出与向量引擎111的输入大小相同。在一些实施例中,向量引擎111的输出与向量引擎111的输入大小相比较小。例如,向量引擎111最多可以接收96个元素,每个大小为4个字节,并且输出96个元素,每个大小为1个字节。在各种实施例中,向量引擎111对输出结果执行量化,从而导致向量引擎111的输出大小相比于向量引擎111的输入而变小。在各种实施例中,量化作为单个指令的部分来执行。例如,量化函数和非线性函数被作为单个处理器指令来执行。如上所述,在一些实施例中,从数据输入103和权重输入105到矩阵处理器107的通信信道的宽度为96个元素,其中每个元素的大小为1个字节,并且与向量引擎111的输出大小(宽度为96个元素,其中每个元素的大小为1个字节)相匹配。

[0031] 在一些实施例中,包括处理元件113在内的向量引擎111的处理元件各自都包括算术逻辑单元(ALU)(未示出)。例如,在一些实施例中,每个处理元件的ALU能够执行算术操作。在一些实施例中,处理元件的每个ALU能够并行执行修正线性单元(ReLU)函数和/或缩放函数。在一些实施例中,每个ALU能够执行包括非线性激活函数的非线性函数。在各种实施例中,向量引擎111的每个处理元件包括用于接收输入操作数的一个或多个触发器。在一些实施例中,每个处理元件可以访问向量引擎累加器的切片和/或向量引擎111的向量寄存器。例如,能够接收96个元素的向量引擎包括宽度为96个元素的累加器和一个或多个96个元素的向量寄存器。每个处理元件都可以访问累加器寄存器和/或向量寄存器的一元素切片。在一些实施例中,每个元素的大小为4个字节。在各种实施例中,累加器寄存器和/或向量寄存器的大小被确定为至少适合输入数据向量的大小。在一些实施例中,向量引擎111包

括大小被设计为适合于向量引擎111的输出的附加向量寄存器。

[0032] 在一些实施例中,向量引擎111的处理元件被配置为从矩阵处理器107接收数据,并且处理元件中的每个处理元件可以并行处理所接收到的数据的部分。作为处理元件的一个示例,向量引擎111的处理元件113从矩阵处理器107的计算单元109接收数据。在各种实施例中,向量引擎111接收单个向量处理器指令,并且进而处理元件中的每个处理元件与其他处理元件并行执行处理器指令。在一些实施例中,处理器指令包括一个或多个组件指令,诸如加载操作、存储操作和/或算术逻辑单元操作。在各种实施例中,可以使用无操作操作来代替组件指令。

[0033] 在所示例中,数据输入103与矩阵处理器107、权重输入105与矩阵处理器107、矩阵处理器107与向量引擎111、以及向量引擎111与后处理单元115之间的虚线箭头描绘了能够发送多个数据元素(诸如数据元素的向量)的相应组件的对之间的耦合。作为示例,矩阵处理器107和向量引擎111之间的通信信道可以为 96×32 个字节宽,并且支持并行传送96个元素,其中每个元素的大小为32个字节。作为另一示例,向量引擎111与后处理单元115之间的通信信道的宽度可以为 96×1 字节宽并且支持并行传送96个元素,其中每个元素的大小为1个字节。在各种实施例中,数据输入103和权重输入105耦合到存储器模块(图1中未示出),并且每个都可以从存储器模块接收输入数据。在一些实施例中,向量引擎111附加地耦合到存储器模块(图1中未示出),并且可以接收来自存储器模块的输入数据,附加地或备选的来自矩阵处理器107的输入。在各种实施例中,存储器模块通常是静态随机存取存储器(SRAM)。

[0034] 在一些实施例中,矩阵处理器107的一个或多个计算单元可以一起被分组到通道中,使得矩阵处理器107具有多个通道。在各种实施例中,矩阵处理器107的通道可以与数据输入103或权重输入105对准。例如,与权重输入105对准的通道包括一组计算单元,其被配置为接收权重输入105的每个操作数作为输入。同样,与数据输入103对准的通道包括一组计算单元,其被配置为接收数据输入103的每个操作数作为输入。在图1所示的示例中,通道以竖直列沿着权重输入105对准,并且每个通道馈送到向量引擎111的对应通道。在一些实施例中,每个通道是包括乘法功能、加法功能和/或累加功能和移位功能的子电路的竖直列。在一些实施例中,矩阵处理器107包括瓦片矩阵,并且每个瓦片是计算单元的矩阵。例如,96x 96矩阵处理器可以包括6x 6个瓦片的矩阵,其中每个瓦片包括16x 16个计算单元。在一些实施例中,竖直通道是单个瓦片列。在一些实施例中,水平通道是单个瓦片行。在各种实施例中,可以动态配置通道的尺寸,并且可以将其用于对矩阵处理器107、向量引擎111和/或后处理单元115的输入执行对准操作。在一些实施例中,动态配置由控制单元101或使用控制单元101和/或通过使用由控制单元101控制的处理器指令执行。

[0035] 在一些实施例中,控制单元101同步由矩阵处理器107、向量引擎111和后处理单元115执行的处理。例如,控制单元101可以将处理器特定指令发送到矩阵处理器107、向量引擎111以及后处理单元115中的每个。控制单元101可以将矩阵处理器指令发送到矩阵处理器107。矩阵处理器指令可以是计算阵列指令,其指令计算阵列使用来自数据输入103和/或权重输入105的指定操作数执行算术操作(诸如点积或点积分量)。控制单元101可以将向量处理器指令发送到向量引擎111。例如,向量处理器指令可以包括与要由向量计算单元一起执行的多个组件指令的单个处理器指令。控制单元101可以将后处理指令发送到后处理单

元115。在各种实施例中,控制单元101同步从数据输入103和权重输入105馈送到矩阵处理器107的数据、从矩阵处理器107馈送到向量引擎111的数据、以及从向量引擎111馈送到后处理单元115的数据。在一些实施例中,控制单元101通过利用处理器特定存储器、入队和/或出队操作同步包括数据输入103、权重输入105、矩阵处理器107、向量引擎111和/或后处理单元115之间在内的微处理器系统100的不同组件之间的数据。在一些实施例中,数据和指令同步由控制单元101执行。在一些实施例中,数据和指令同步由包括一个或多个定序器的控制单元101执行,一个或多个定序器用于同步矩阵处理器107、向量引擎111和/或后处理单元115之间的处理。

[0036] 在一些实施例中,矩阵处理器107和向量引擎111用于处理卷积层。在一些实施例中,向量引擎111用于对矩阵处理器107的输出执行非线性函数,诸如激活函数。例如,矩阵处理器107可以用于计算点积,并且向量引擎111可以用于执行激活函数,诸如修正线性单元(ReLU)函数或S型函数。在一些实施例中,后处理单元115用于执行池化操作。在一些实施例中,后处理单元115用于将经处理的数据格式化并且将其存储到存储器,并且可以被用于同步存储器写入等待时间。

[0037] 图2是图示了用于执行机器学习处理的微处理器系统的实施例的框图。在所示的示例中,微处理器系统200包括控制单元201、向量输入203、向量引擎输入队列207、向量引擎211和后处理单元215。向量引擎输入队列207包括多个计算单元,其包括计算单元209和221-229;并且向量引擎211包括多个处理元件,其包括处理元件213和231。向量输入203是用于将数据馈送到向量引擎输入队列207的输入模块。在一些实施例中,向量输入203包括输入数据格式化器、高速缓存或缓冲器和/或逻辑电路,其用于为向量引擎输入队列207准备数据。例如,向量输入203可以利用向量引擎输入队列207作为先进先出(FIFO)输入队列来从由向量引擎211处理的二维阵列中准备N个操作数。在一些实施例中,向量输入203耦合到存储器(图2中未示出),诸如用于检索数据的静态随机存取存储器(SRAM)。

[0038] 在各种实施例中,控制单元201、向量输入203、向量引擎输入队列207、向量引擎211和后处理单元215分别是图1的控制单元101、数据输入103、矩阵处理器107、向量引擎111和后处理单元115。例如,图1的矩阵处理器107可以用于通过从图1的数据输入103接收数据并且向图1的向量引擎111重复移位每个输入向量来实现输入队列,诸如向量引擎输入队列207。

[0039] 在一些实施例中,向量引擎输入队列207是计算阵列单元,并且包括其列是先进先出(FIFO)队列的计算单元矩阵。在所示的示例中,向量引擎输入队列207是向量输入203的输入队列,并且用作宽的先进先出(FIFO)队列,以将多个数据元素从向量输入203馈送到向量引擎211。例如,计算单元221-229组成作为单个FIFO队列一起工作的计算单元的竖直列。在各种实施例中,向量引擎输入队列207包括由与计算单元221-229类似的计算单元的竖直列组成的多个FIFO队列。例如,在向量引擎输入队列207的宽度为96个计算单元的实施例中,向量引擎输入队列207具有与96个FIFO队列相对应的计算单元的96个竖直列。作为另一示例,在向量引擎输入队列207的长度为96个计算单元的实施例中,向量引擎输入队列207具有长度为96个阶段的FIFO队列。

[0040] 在各种实施例中,每个先进先出(FIFO)队列并行工作,并且将从向量输入203接收的输入沿着FIFO队列转移到向量引擎211。向量引擎输入队列207(包括计算单元221)的计

算单元的第一行连接到向量输入203。计算单元的第一行被配置为从向量输入203并行接收数据的整行。向量引擎输入队列207的计算单元的最后一行连接到向量引擎211的处理元件的行。例如,向量引擎输入队列207的计算单元的最后一行包括计算单元229和209。计算单元209连接到处理元件213,而计算单元229连接到处理元件231。处理元件213和231被配置为分别接收计算单元209和229的数据输出元件。向量引擎211的处理元件从向量引擎输入队列207的计算单元的最后一行并行接收数据的整行。在各种实施例中,当向量引擎输入队列207的计算单元的最后一行具有可用于出队的信号时,向量引擎211接收出队准备好的信号以指示向量引擎输入队列207准备接收队列操作。

[0041] 在所描述的示例中,来自计算单元的第一行的数据朝着向量引擎211以逻辑方向向下移位到计算单元的下一行。例如,与向量输入203的数据元素相对应的输入在计算单元221处接收作为操作数,并且从计算单元221移位到计算单元222,从计算单元222移位到计算单元223,从计算单元223移位到计算单元224,依此类推,直到在计算单元221处接收的操作数经由中间计算单元222-228从计算单元221递增地移位到计算单元229。在各种实施例中,推入FIFO的数据元素进行的移位与FIFO在计算单元中的深度一样多。例如,长度为96个计算单元和96个阶段的FIFO队列需要96个移位才能使插入的元素出队。在各种实施例中,FIFO的每一阶段可以与其他阶段并行移位操作数。例如,当FIFO队列中的每个中间计算单元将其操作数移位到下一计算单元时,第一计算单元可以从向量输入203中检索下一数据元素,而最后一个计算单元可以使它的数据元素出队以由向量引擎211的对应处理元件接收。在所描述的示例中,沿着计算单元的每一行的每个计算单元并行工作,以将最初从向量输入端203接收到的其对应数据元素移位到向量引擎211。

[0042] 在一些实施例中,向量引擎输入队列207耦合到向量输入203,并且计算单元的矩阵的一个尺寸与向量输入203的尺寸相匹配。例如,在其中向量输入203的宽度为96个字节的实施例中,向量引擎输入队列207具有宽度为至少96个字节的计算单元的矩阵。在一些实施例中,向量输入203的宽度和向量引擎输入队列207的输入的对宽度是动态地可配置的。例如,向量输入203可以被动态地配置为96个字节或96x 2个字节,向量引擎输入队列207的输入的对宽度分别可以被配置为96个字节或96x 2个字节。在一些实施例中,使用控制单元201和/或向量引擎输入队列207的处理器指令来执行配置。

[0043] 在一些实施例中,向量引擎211是通信地耦合到向量引擎输入队列207的向量计算单元。向量引擎211包括多个处理元件,其包括处理元件213和231。在所示的图中,向量引擎211内部的小方块描绘了向量引擎211包括的多个处理元件,该多个处理元件被布置为向量。在一些实施例中,处理元件以与向量输入203相同的方向布置在向量中。在各种实施例中,向量引擎211的处理元件的数据大小与向量引擎输入队列207的计算单元的数据大小相同或比其更大。例如,在一些实施例中,计算单元209接收大小为1个字节的操作数,并且使出队到处理元件213的输出的大小也为1个字节。处理元件213接收从计算单元209的1字节输出作为大小为1字节的输入。在各种实施例中,向量引擎211的输出与向量引擎211的输入的大小相同。在各种实施例中,向量引擎211的输出与到向量引擎211的输入相比尺寸更小。例如,向量引擎211可以接收多达96个元素,每个元素大小为4个字节,并且输出96个元素,每个元素大小为1个字节。在一些实施例中,从向量输入203到向量引擎输入队列207的通信信道的宽度为96个元素,其中每个元素的大小为1个字节,并且与向量引擎211的输出大小

(宽度为96个元素,每个元素的大小为1个字节)相匹配。

[0044] 在一些实施例中,包括处理元件213和231的向量引擎211的处理元件各自都包括算术逻辑单元(未示出),并且相对于图1的向量引擎111进行更详细的描述。在一些实施例中,向量引擎211的处理元件被配置为从向量引擎输入队列207接收数据,并且处理元件中的每个处理元件可以并行处理所接收的数据的部分。作为处理元件的一个示例,向量引擎211的处理元件213和231分别从向量引擎输入队列207的计算单元209和229接收数据。在各种实施例中,向量引擎211接收单个向量处理器指令,并且处理元件中的每个处理元件进而与其他处理元件并行执行处理器指令。在一些实施例中,处理器指令包括一个或多个组件指令,诸如加载操作、存储操作和/或算术逻辑单元操作。在各种实施例中,可以使用非操作操作(no-op operation)来代替组件指令。

[0045] 在所示的示例中,向量输入203与向量引擎输入队列207、向量引擎输入队列207与向量引擎211、向量引擎211与后处理单元215之间的虚线箭头描绘了能够发送多个数据元素的相应组件之间的耦合。作为示例,向量引擎输入队列207和向量引擎211之间的通信信道的宽度可以为 96×32 个字节并且支持并行传送96个元素,其中每个元素的大小是32个字节。作为另一示例,向量引擎211与后处理单元215之间的通信信道的宽度可以为 96×1 个字节并且支持并行传送96个元素,其中每个元素的大小为1字节。在各种实施例中,向量输入203耦合到存储器模块(图2中未示出),并且可以从存储器模块接收输入数据。在一些实施例中,向量引擎211附加地耦合到存储器模块(图1中未示出),并且可以附加地或备选地从向量引擎输入队列207接收输入还从存储器模块接收输入数据。在各种实施例中,存储器模块通常是静态随机存取存储器(SRAM)。

[0046] 在一些实施例中,向量引擎输入队列207的一个或多个计算单元可以一起分组为竖直列,使得向量引擎输入队列207具有多个竖直列通道。在图2所示的示例中,通道沿着与上文所描述的先进先出(FIFO)队列相同的竖直列对准,并且每个通道馈送到向量引擎211的对应通道。在一些实施例中,每个通道是子电路的竖直列,这些子电路包括乘法功能、加法功能和/或累加功能、以及移位功能。在一些实施例中,竖直通道是计算单元的单个列。在一些实施例中,竖直通道是一组相邻计算单元的多个列。在各种实施例中,可以动态配置通道的尺寸,并且可以将其用于对向量引擎输入队列207、向量引擎211和/或后处理单元215的输入执行对准操作。在一些实施例中,动态配置由控制单元201或使用控制单元201和/或通过使用由控制单元201控制的处理器指令执行。

[0047] 在一些实施例中,控制单元201同步由向量引擎输入队列207、向量引擎211和/或后处理单元215执行的处理。例如,控制单元201可以向向量引擎输入队列207、向量引擎211和后处理单元215中的每个发送处理器特定指令。控制单元201可以向向量引擎输入队列207发送向量引擎输入队列指令。在一些实施例中,向量引擎输入队列指令是矩阵处理器指令的子集,图1的矩阵处理器107能够对向量引擎输入队列指令做出响应并且关于图1对其进行进一步描述。向量引擎输入队列指令可以是指令计算阵列执行加载操作、移位操作或其他操作的计算阵列指令,或与输入队列接口的其他适当指令。控制单元201可以向向量引擎211发送向量处理器指令。例如,向量处理器指令可以包括具有要由向量计算单元一起执行的多个组件指令的单个处理器指令。控制单元201可以向后处理单元215发送后处理指令。在各种实施例中,控制单元201同步从向量输入203馈送到向量引擎输入队列207的数

据、从向量引擎输入队列207馈送到向量引擎211的数据、以及从向量引擎211馈送到后处理单元215的数据,在一些实施例中,控制单元201通过利用处理器特定处理器、入队和/或出队操作来同步不同组件(向量输入203、向量引擎输入队列207、向量引擎211和/或后处理单元215)之间的数据。关于图1的控制单元101对控制单元201的功能进行进一步详细描述。

[0048] 在一些实施例中,控制单元201用于配置要由向量引擎输入队列207、向量引擎211和/或后处理单元215接收的数据元素的大小和数目。例如,在一些实施例中,控制单元201可以用于将向量引擎输入队列207的输入配置为每个大小为1个字节的96个元素或其他适当的变化,诸如每个大小为2个字节的48个元素、每个大小为2个字节的96个元素、每个大小为192个元素等。在一些实施例中,向量引擎输入队列207能够通过执行一系列加载操作和逻辑移位操作来输出其大小大于其可以接收的大小的数据元素。例如,4字节输入数据元素通过读取该4字节输入数据元素的四个顺序1字节部分并且将每个字节逻辑移位到适当字节字段来加载到向量引擎输入队列207中。作为另一示例,在一些实施例中,控制单元201可以用于将到向量引擎211的输入配置为每个大小为4个字节的96个元素,或其他适当变化,诸如每个大小为1个字节的96个元素、每个大小为2个字节的48个元素等。

[0049] 在各种实施例中,后处理单元215用于对向量引擎211的输出执行后处理。关于图1的后处理单元115而对后处理单元215的后处理功能进行进一步详细的描述。

[0050] 图3是图示了用于执行机器学习处理的微处理器系统的实施例的框图。在所示的示例中,微处理器系统300包括控制单元301、存储器307、向量引擎311和后处理单元315。在各种实施例中,存储器307通常是静态随机存取存储器(SRAM)。在各种实施例中,后处理单元315从向量引擎311接收输入数据,并且用于对来自向量引擎311的输出执行后处理。关于图1的后处理单元115而对后处理单元315的后处理功能进行进一步详细描述。

[0051] 图3的框图描绘了系统体系架构实施例,其中向量引擎311耦合到存储器307,并且可以直接从存储器307检索数据。在各种实施例中,存储器307与向量引擎311之间的通信信道的大小可以被配置为将多个数据元素从存储器307并行传送到向量引擎311。例如,在向量引擎311能够并行接收每个大小为32个字节的96个元素的实施例中,存储器307与向量引擎311之间的通信通道的大小被配置为将每个大小为32个字节的96个元素从存储器307并行传送到向量引擎311。在一些实施例中,存储器307包括数据格式化器(未示出),该数据格式化器可以包括数据高速缓存或缓冲器和/或逻辑电路,其用于在传送到向量引擎311之前格式化来自存储器的数据。例如,大小为1个字节的数据元素可以存储在存储器307中的字边界上,并且数据格式化器用于将数据格式化和/或屏蔽到字节边界。在各种实施例中,控制单元301、向量引擎311和后处理单元315分别是图1的控制单元101、向量引擎111和后处理单元115。在各种实施例中,向量引擎311可以进一步耦合到关于图1的矩阵处理器107所描述的矩阵处理器(未示出)。

[0052] 在一些实施例中,向量引擎311是通信耦合到存储器307的向量计算单元。向量引擎311包括多个处理元件,其包括处理元件313。在所示的图中,向量引擎311内部的小方块描绘了向量引擎311包括多个处理元件,多个处理元件被布置为向量。在一些实施例中,包括处理元件313在内的向量引擎311的处理元件各自都包括算术逻辑单元(未示出)。向量引擎311的处理元件被配置为从存储器307接收数据,并且处理元件中的每个处理元件可以并行处理接收的数据的部分。在各种实施例中,向量引擎311接收单个向量处理器指令,进而

处理元件中的每个处理元件与其他处理元件并行执行处理器指令。在一些实施例中,处理器指令包括一个或多个组件指令,诸如加载操作、存储操作和/或算术逻辑单元操作。分别关于图1和图2的向量引擎111和211而对向量引擎311的功能进行进一步详细描述。

[0053] 在一些实施例中,控制单元301同步向量引擎311和后处理单元315所执行的处理,并且访问存储器307。例如,控制单元301可以向向量引擎311和后处理单元315中的每个发送处理器特定指令。在一些实施例中,控制单元301可以向向量引擎311发送向量处理器指令。例如,向量处理器指令可以包括具有要由向量计算单元一起执行的多个组件指令的单个处理器指令。在一些实施例中,控制单元301可以向后处理单元315发送后处理指令。在各种实施例中,控制单元301同步由向量引擎311从存储器307接收的数据与由后处理单元315从向量引擎311接收的数据。在一些实施例中,控制单元301通过利用向量引擎和/或后处理单元处理器特定操作来同步不同组件(向量引擎311和/或后处理单元315)之间数据。关于图1的控制单元101而对控制单元301的功能进行更详细的描述。

[0054] 在一些实施例中,控制单元301用于配置要由向量引擎311和/或后处理单元315接收的数据元素的大小和数目。例如,在一些实施例中,控制单元301可以用于将向量引擎311配置为接收每个大小为4个字节的96个数据元素,或其他适当的变化,诸如每个大小为1个字节的96个元素、每个大小为2个字节的48个元素等。如关于图1和图2进一步描述的,向量引擎311与后处理单元315之间的虚线箭头描绘了能够发送多个数据元素的相应的组件的对之间的耦合。作为示例,向量引擎311与后处理单元315之间的通信信道的宽度可以为 96×1 字节并且支持并行传送96个元素,其中每个元素的大小为1个字节。

[0055] 图4A是图示了用于执行机器学习处理的向量计算单元的实施例的框图。在所示的示例中,微处理器系统400包括向量计算单元401、输入总线411和输出总线431。向量计算单元401的输入从输入总线411到达。向量计算单元401的输出写入到输出总线431。在实施例中,输入总线411和输出总线431是单个总线,其包括输入总线411和输出总线431两者的功能。在各种实施例中,输入总线411和输出总线431是宽数据总线,其允许并行传送多个数据元素。例如,输入总线411的宽度可以为 96×32 个字节,输出总线431的宽度可以为96个字节,以适应计算单元401的并行处理功能。在一些实施例中,向量计算单元401经由输入总线411接收向量计算单元指令。在一些实施例中,向量计算单元401经由除诸如指令总线(未示出)的输入总线411之外的通信信道接收向量计算单元指令。

[0056] 在各种实施例中,向量计算单元401分别是图1、图2和图3的向量引擎111,211和/或311。在一些实施例中,输入总线411连接到图1的矩阵处理器107、图2的向量引擎输入队列207、和/或图3的存储器307。在一些实施例中,输出总线431分别连接到图1、图2和图3的后处理单元115,215和/或315。在各种实施例中,向量计算单元401双向耦合到向量计算单元401外部的微处理器系统400的控制单元(未示出),诸如分别双向耦合到图1、图2和图3的控制单元101,201和/或301。在各种实施例中,微处理器系统400的控制单元向向量计算单元401发送向量计算单元指令。在一些实施例中,微处理器系统400的控制单元包括一个或多个定序器,其用于将指令和数据同步到向量计算单元401。

[0057] 在所示的示例中,向量计算单元401包括寄存器421、向量引擎控制逻辑423、输入缓冲器425、算术逻辑单元(ALU)427和输出缓冲器429。来自输入总线411的输入数据由输入缓冲器425接收,并且,写入到输出总线431的输出从输出缓冲器429写入。在一些实施例中,

输入缓冲器425和输出缓冲器429是数据缓冲器或高速缓存,并且提供存储器同步功能。例如,在一些实施例中,来自输入总线411的输入读取和/或对输出总线431的输出写入具有不可预测的等待时间,其可以通过利用接收输入数据的输入缓冲器425和用于存储计算的结果的输出缓冲器429而被消除。作为另一示例,当来自ALU 427的输出准备写入时,输出总线431可能不可用。在一些实施例中,输出缓冲器429允许ALU 427继续处理挂起数据,直到输出总线431可用于写入存储在输出缓冲器429处的结果为止。在各种实施例中,输入总线411和输出总线431是由微处理器系统400的控制单元(未示出)控制的通信信道。

[0058] 如上所述,在各种实施例中,向量计算单元包括多个处理元件。在一些实施例中,每个处理元件包括用于加载数据、存储数据以及执行算术逻辑单元操作的单独功能。图4A的框图中未对各个处理元件进行描绘。在各种实施例中,算术逻辑单元(ALU) 427包括每个处理单元的对应的算术逻辑单元(ALU)。同样,输入缓冲器425和输出缓冲器429包括对应的用于每个处理单元的输入缓冲器和输出缓冲器。在各种实施例中,ALU 427包括ALU逻辑,其用于并行处理到向量计算单元401的输入向量的每个元素。在一些实施例中,ALU 427包括用于量化ALU结果的逻辑。在各种实施例中,可以响应于单个处理器指令来执行ALU逻辑,例如,用于执行非线性函数和量化的逻辑。

[0059] 在各种实施例中,寄存器421包括用于实现向量计算单元401的功能的寄存器。例如,寄存器421可以用于存储用于执行向量计算单元指令的操作数,实现字节屏蔽以及使用不同的存储器大小的寄存器混叠(alias)以及其他适当功能引用向量元素。在一些实施例中,寄存器421包括算术指令向量寄存器;屏蔽寄存器;用于执行算术操作(诸如加法操作、减法和浮点操作)的寄存器;和/或用于混叠向量元素的寄存器。在一些实施例中,用于混叠向量元素的寄存器还用于执行算术操作。

[0060] 在一些实施例中,寄存器421包括算术指令向量寄存器。例如,寄存器可以作为用于加载操作、存储操作和算术逻辑单元(ALU)操作的操作数。作为另一示例,在一些实施例中,ALU操作可以采用多达四个向量寄存器作为参数,其中三个作为源寄存器,一个作为目的寄存器。在各种实施例中,处理器操作所使用的向量寄存器基于向量元素的大小而被混叠为不同的向量元素。例如,在一些实施例中,一组不同的向量寄存器可用于对8字节值,16字节值,32字节值和/或浮点值进行操作。在一些实施例中,用于32字节值的该组向量寄存器也用于浮点值。在各种实施例中,32字节向量寄存器混叠为16字节向量寄存器和8字节向量寄存器。例如,一个32字节向量寄存器混叠为两个16字节向量寄存器和四个8字节向量寄存器。作为另一示例,具有八个96x 32字节向量寄存器(寄存器RD0-RD7)的向量计算单元401混叠为16个96x 16字节向量寄存器(寄存器RW0-RW15)和32个96x 8字节位向量寄存器(寄存器RB0-RB31)。RD0是96x 32字节向量寄存器,RW0是96x 16字节向量寄存器,RB0是96x 8字节向量寄存器。图4B中描绘了向量寄存器混叠的另一示例。

[0061] 在一些实施例中,寄存器421基于向量计算单元401的处理元件的数目来包括一个或多个字节屏蔽寄存器。例如,具有96个处理元件的向量计算单元可以包括一个或多个96字节屏蔽寄存器。在各种实施例中,可以通过从存储器加载字节屏蔽来设置屏蔽寄存器。屏蔽寄存器可以用于将对输入数据执行的逻辑操作的结果存储到向量计算单元401。

[0062] 在一些实施例中,寄存器421包括用于执行算术操作(诸如加法操作、减法操作、以及浮点操作)的寄存器。例如,在一些实施例中,向量计算单元401包括寄存器,其用于存储

用于向量加法和减法指令的进位字节以及与浮点指令相对应的状态字节。

[0063] 在一些实施例中,向量计算单元401包括指令缓冲器(未示出),其用于存储一系列向量计算单元指令。在一些实施例中,指令缓冲器是命令队列。在各种实施例中,指令缓冲器包括一个或多个指针,一个或多个指针引用要执行的当前指令和/或最后一条指令。在各种实施例中,指令缓冲器充当向量计算单元指令的高速缓存。例如,一个或多个向量计算单元指令加载到向量计算单元401的指令缓冲器中并且进行高速缓存,直到指令可以被执行为止。随着指令被执行并且不再被需要,新指令可以加载到指令缓冲器中。在一些实施例中,经由微处理器系统400的控制逻辑(未示出)从外部指令命令队列接收向量计算单元指令。

[0064] 在一些实施例中,向量计算单元401包括向量引擎控制逻辑423。向量引擎控制逻辑423用于实现向量计算单元401的功能,其包括获取向量计算单元指令,解码指令和/或执行指令。在各种实施例中,向量引擎控制逻辑423包括用于经由输入缓冲器425、输出缓冲器429和寄存器421读取、写入、屏蔽和/或混叠数据的逻辑。在一些实施例中,向量计算单元401接收出队准备好的信号,并且使用向量引擎控制逻辑423确定数据能够经由输入总线411获得。例如,向量引擎控制逻辑423可以在接收到出队准备好的信号时使数据从附接到输入总线411的输入先进先出队列(未显示)中出队。

[0065] 图4B是图示了示例性的向量寄存器的混叠的表。表450图示了向量计算单元实施例的向量寄存器的混叠,其中八个 96×32 字节向量寄存器(寄存器RD0-RD7)混叠为十六个 96×16 字节向量寄存器(寄存器RW0-RW15)和32个 96×8 字节向量寄存器(寄存器RB0-RB31)。在一些实施例中,表450中的向量寄存器是图4A的向量计算单元401的寄存器421中的向量寄存器。在所示的示例中,行451包括字节0,1,2和3的列,这些列混叠为列于该行451下面的各行中的相应寄存器。行453,463和473与 96×32 位向量寄存器RD0,RD1和RD7相对应。行455,465和475与 96×16 位向量寄存器RW0-3和RW14-15相对应。行457,467和477与 96×8 字节向量寄存器RB0-7和RB28-31相对应。在该示例中,字节0-3是向量计算单元(诸如分别图1、图2和图3的向量引擎111,211和/或311)的96个通道中的一个通道。

[0066] 在所示的示例中,表450图示了向量计算单元实施例的96个通道中的单个通道的向量寄存器混叠。 96×32 字节向量寄存器RD0使用从字节0到字节3排序的四个字节。 96×16 字节向量寄存器RW0和RW1各自混叠为2个字节。向量寄存器RW0混叠为字节0和字节1,而向量寄存器RW1混叠为字节2和字节3。 96×8 字节向量寄存器RB0-RB3混叠为1个字节,其各自与字节0-3相对应。同样, 96×32 字节向量寄存器RD1分别混叠为 96×16 字节向量寄存器RW2(字节0和1)和RW3(字节2和3),而 96×8 位字节向量寄存器RB4-RB7分别混叠为字节0-3。作为另一示例, 96×32 字节向量寄存器RD7分别混叠为 96×16 字节向量寄存器RW14(字节0和1)和RW15(字节2和3),而 96×8 位向量寄存器RB28-RB31分别混叠为字节0-3。

[0067] 在各种实施例中,向量计算单元指令对向量寄存器的所有96个通道进行并行操作。例如,对于96个通道中的每个通道,向量寄存器RB0对字节0进行操作,向量寄存器RB5对字节1进行操作,向量寄存器RW2对字节0和1进行操作,向量寄存器RW15对字节2和3进行操作,并且向量寄存器RD7对字节0-3进行并行操作。

[0068] 图5是图示了对于确定微处理器系统的处理器指令的过程的实施例的流程图。在一些实施例中,图5的过程将用高级编程语言编写的软件程序转换为用于具有计算阵列和

向量计算单元的微处理器系统的一系列计算阵列和向量计算单元指令。在各种实施例中,微处理器系统是图1的微处理器系统100,计算阵列是图1的矩阵处理器107,向量计算单元是图1的向量引擎111。在各种实施例中,图5的过程用于实现依赖于机器学习的应用,这些应用包括使用机器学习模型执行推理的应用,诸如自动驾驶汽车和驾驶员辅助汽车。

[0069] 在501处,确定要执行的处理以及要被指派到不同的协同处理组件(诸如计算阵列、向量计算单元和/或后处理单元)的处理子集。在各种实施例中,基于不同的协同处理组件的功能和效率来指派处理。例如,某些矩阵相关操作被指派到计算阵列,并且涉及非线性函数(诸如激活函数)的操作可以被指派到向量计算单元。在一些实施例中,池化操作被指派到后处理单元。作为另一示例,在一些实施例中,在501处,确定卷积操作需要点积操作,并且确定点积操作最佳地利用了由计算阵列执行的矩阵处理。在一些实施例中,该确定通过编译机器学习应用来执行,以针对本文中所描述的微处理器系统。

[0070] 在503处,确定与在501处确定和指派的处理相对应的一个或多个矩阵处理器指令。例如,在501处确定的要由矩阵处理器执行的点积操作被转换为一个或多个矩阵处理器指令。在各种实施例中,矩阵处理器指令是计算阵列指令。作为示例,计算阵列指令可以要求从诸如图1的数据输入103的数据输入组件接收一个或多个数据向量,并且从诸如图1的权重输入105的对应权重输入组件接收一个或多个权重向量。附加计算阵列指令可以包括用于处理点积操作的乘法操作、累加操作和移位操作。例如,一个或多个点积分量操作可以用于计算点积结果。在各种实施例中,计算阵列指令被引导到由对应的计算阵列的计算单元对接收到的输入数据执行的处理。在一些实施例中,附加计算阵列指令包括用于准备点积结果以由向量计算单元进行处理的指令。

[0071] 在505处,做出关于要由向量计算单元执行的向量引擎指令的确定。例如,在501处确定的要由向量引擎执行的与激活函数有关的操作被转换为一个或多个向量引擎指令。在各种实施例中,向量引擎指令是向量计算单元指令。作为示例,向量计算单元指令可以要求从诸如图1的矩阵处理器107的计算阵列接收一个或多个数据向量。附加向量计算单元指令可以包括用于执行诸如修正线性单元(ReLU)函数的非线性激活函数的操作。在各种实施例中,向量计算单元指令被引导到由向量计算单元的对应处理元件对接收的输入数据执行的处理。在一些实施例中,附加的向量计算单元指令包括用于准备处理元件的结果以供后处理单元进行后处理的指令。

[0072] 在各种实施例中,每个向量计算单元指令是单个处理器指令,其指定要由向量计算单元一起执行的多个组件指令。响应于单个向量计算单元指令,由向量计算单元的处理元件对不同的数据输入元件并行执行多个组件指令的执行。例如,在一些实施例中,单个处理器指令包括三个组件指令:单独加载、算术逻辑单元、以及存储指令。由向量计算单元接收并且执行这三个组件指令。在一些实施例中,在505处执行组件指令到单个处理指令中的绑定。在各种实施例中,用于组件指令到向量计算单元指令中的绑定的排序和选择取决于所确定的数据危害。

[0073] 在507处,做出关于要由后处理单元执行的后处理指令的确定。例如,与后处理功能有关的操作在501处被确定为要由后处理单元执行,并且被转换为一个或多个后处理指令。作为示例,后处理指令可以要求从诸如图1的向量引擎111的向量计算单元接收一个或多个数据向量。附加的后处理指令可以包括用于执行池化层功能的操作,诸如最大池化。在

各种实施例中,后处理指令可以包括用于配置池化功能(诸如内核大小、步幅和/或空间范围等)的指令。在一些实施例中,附加的后处理指令包括用于准备和写出后处理结果的指令。

[0074] 在509处,调度与在503,505和507确定处的协处理器指令集合的执行相对应的序列。例如,确定用于各种协处理器(例如,计算阵列、向量计算单元和/或后处理单元)的相应处理器指令的相对次序和/或顺序。在一些实施例中,该顺序取决于协处理器之间的相互作用和依赖性。例如,向量计算单元的输入可以取决于来自计算阵列的输出结果的可用性。在各种实施例中,确定并且解决包括数据危害的依赖性。例如,在各种实施例中,向量计算单元指令包括多个组件指令,并且可以被执行使得并行执行多个向量计算单元指令。确定并且解决基于不可用数据资源的数据危害。例如,非操作可以插入向量计算单元指令的组件指令中,以允许在执行取决于加载操作完成的算术逻辑单元操作之前完成加载操作。在一些实施例中,在509处确定将组件指令捆绑到单个向量计算单元指令中。在一些实施例中,分别针对矩阵处理器和向量引擎在503和505处执行一些或全部指令调度,诸如协处理器指令的排序。例如,在一些实施例中,在505处确定针对每个单个向量计算单元指令的组件指令的捆绑。

[0075] 在一些实施例中,微处理器系统的控制单元和/或一个或多个定序器用于发起和协调协处理器指令集合的处理。例如,诸如图1的控制单元101的控制单元和/或一个或多个定序器利用在509处确定的指令顺序被,以将对应的协处理器指令发布到诸如矩阵处理器107的计算阵列、诸如图1的向量引擎111的向量计算单元和/或诸如图1的后处理单元113的后处理单元。在一些实施例中,一个或多个定序器的功能由控制单元执行。例如,在一些实施例中,控制单元包括执行定序器、存储器访问定序器、网络定序器和/或向量引擎定序器等。

[0076] 图6A是图示了对于向量计算单元的运行执行的过程的实施例的流程图。向量计算单元可以执行图6A的过程以并行处理向量的元素。在各种实施例中,向量计算单元分别是图1、图2、图3和图4A的向量引擎111,211,311和/或向量计算单元401。在一些实施例中,诸如图1的控制单元101的控制单元发起图6A的过程。在各种实施例中,由诸如图4A的向量引擎控制逻辑423的向量计算单元的控制逻辑来执行图6A中的过程的步骤之间的过渡。

[0077] 在601处,检索向量引擎指令。在各种实施例中,向量引擎指令是向量计算单元指令,并且指定多个组件指令。例如,指令三元组是指定多达三个组件指令的单个向量计算单元指令。示例指令三元组包括加载操作、算术逻辑单元操作和存储操作作为单个指令。在601处,一旦检索到指令,则过程继续进行到603和605两者。

[0078] 在603处,关于附加指令是否挂起进行确定。例如,下一向量引擎指令可以是可用的并且准备检索。作为另一示例,用于缓存挂起指令的指令缓冲器可以为空,并且需要检索和/或等待下一可用指令。在一些实施例中,附加指令的可用性基于检查引用指令缓冲器中的最后一个有效指令的指针。响应于没有可用的附加指令,过程进行到步骤609。响应于一个或多个附加指令的可用性,过程返回到601。

[0079] 在605处,对在601处检索的向量引擎指令进行解码。在各种实施例中,单个向量引擎指令指定一个或多个组件指令。在各种实施例中,对指令和组件指令进行解码。例如,包含加载指令、算术逻辑单元指令和存储组件指令的指令三元组解码为单独的组件操作。在

一些实施例中,解码针对每个组件操作确定操作码和与该操作码相对应的参数。作为一个示例,加载组件指令既包含与字节向量出队操作相对应的操作码,又包含用于存储作为出队结果的字节向量的对应目的地向量寄存器。作为另一示例,加法组件指令既包含与带符号的16字节加法操作相对应的操作码,又包含源参数和目的地参数的对应向量寄存器。

[0080] 在607处,执行在605处解码的指令。在一些实施例中,向量计算单元的处理元件执行指定多个组件指令的单个向量引擎指令。例如,处理元件的向量执行在605处解码的单个向量引擎指令。在一些实施例中,还通过每个处理元件并行执行单个向量引擎指令的组件指令中的每个组件指令。例如,对于每个处理元件,可以并行执行加载指令和算术逻辑单元指令。在一些实施例中,可以并行执行加载指令、算术逻辑单元指令和存储指令。例如,向量引擎的每个处理单元并行执行以下组件操作:输入数据的向量从输入累加器加载到向量寄存器中,算术逻辑单元(ALU)对两个不同的向量寄存器执行浮点乘法操作,并且及16字节元素的向量从向量寄存器存储到存储器。在各种实施例中,一旦处理元件完成了组件指令的执行,就完成了向量引擎指令的处理。

[0081] 在609处,向量计算单元等待下一指令。例如,向量计算单元等待直到用于高速缓存挂起指令的指令缓冲器包含要被执行的有效指令为止。作为另一示例,向量计算单元等待直到从存储器接收到下一指令并且将其提供给向量计算单元为止。在一些实施例中,向量计算单元在609处暂停挂起附加指令的可用性。在各种实施例中,向量计算单元可以在等待附加指令的同时在609处对中断做出响应。响应于附加指令的到达,处理继续回到601。

[0082] 图6B是图示了对于由向量计算单元处理向量数据的过程的实施例的流程图。例如,图6B图示了应用于向量计算单元从诸如计算阵列和/或先进先出(FIFO)队列的输入源所接收的向量数据的过程。在一些实施例中,图6B的过程图示了由向量计算单元执行的用于对向量输入执行向量操作以计算向量结果的步骤。在各种实施例中,图6B的过程利用向量计算单元的多个处理元件来对向量的元素并行执行处理。在各种实施例中,向量计算单元分别是图1、图2、图3和图4A的向量引擎111,211,311和/或向量计算单元401。

[0083] 在651处,对加载操作进行解码和发布。在一些实施例中,需要加载操作以将数据接收到向量计算单元中。例如,在一些实施例中,出队操作是使数据元素的向量从计算阵列中出队以由向量计算单元的处理元件接收的加载操作。在各种实施例中,加载操作可以是组成单个向量计算单元指令的多个组件指令中的一个组件指令。加载操作的解码确定了加载操作的特定类型和适当操作。例如,存在各种加载操作以将不同大小的向量元素加载到不同的指定向量寄存器中。在651处,对加载操作进行解码和发布以发起输入数据的接收,诸如使数据结果的向量从先进先出(FIFO)队列出队。

[0084] 在653处,作为在651处发布的加载操作的结果,向量计算单元接收向量形式的输入数据。例如,向量计算单元从诸如图1的矩阵处理器107的计算阵列、诸如图2的向量引擎输入队列207的先进先出(FIFO)队列或其他适当的数据源接收输入数据元素的向量。在一些实施例中,输入数据存储在输入缓冲器中。在一些实施例中,输入缓冲器利用一组触发器和/或一个或多个累加器来存储输入数据。大小为输入向量的输入缓冲器可以用于存储输入数据,使得在步骤655处,该输入数据可以加载到一个或多个向量寄存器中。

[0085] 在655处,在653处接收的向量数据加载到适当的寄存器中。例如,在653处读取的向量数据加载到由加载指令指定的向量寄存器中。在一些实施例中,寄存器混叠用于确定

如何将数据加载到向量寄存器中。例如,数据可以基于所利用的指令和混叠的寄存器来加载到同一寄存器的存储器位置,但与字节、半字或字边界对准。在一些实施例中,将向量数据加载到向量寄存器中利用字节屏蔽(诸如向量字节屏蔽)来确定将向量的哪些字节加载到哪些寄存器存储器位置中。例如,96字节屏蔽可以用于确定向量寄存器的哪些元素应当接收数据。

[0086] 在657处,做出是否需要附加数据的确定。例如,基于当前向量计算单元指令,在执行算术逻辑单元(ALU)操作之前可能需要其他数据。响应于无需附加数据,处理继续进行到661。作为示例,如果当前向量计算单元指令包括不是非操作操作的ALU组件操作(诸如加法操作),则处理继续进行到661。响应于需要附加数据,例如,加载操作挂起并且没有ALU操作挂起,处理继续进行到659。在一些实施例中,指令三元组可以使用指示不对当前指令执行ALU操作的非操作替换ALU操作。

[0087] 在659处,附加数据加载到向量计算单元中以供处理。例如,可以通过读取存储器、接收矩阵处理器的结果、使先进先出(FIFO)队列出队或其他适当的技术来加载附加输入数据,诸如输入权重的向量。在一些实施例中,可以通过读取诸如静态随机存取存储器(SRAM)的存储器来加载附加数据。在各种实施例中,诸如读取缓冲器的附加组件可以用于同步数据的加载和/或考虑读取延迟和等待时间。在各种实施例中,在659处加载的数据可以是输入数据的向量,诸如权重输入的向量。

[0088] 在661处,执行向量算术逻辑单元(ALU)操作。在各种实施例中,向量ALU操作包括用于加法(带符号的和不带符号的)、减法(带符号的和不带符号的)、乘法、绝对值和逻辑操作符的向量操作。可以对不同的操作数大小执行向量ALU操作。示例操作数大小包括8字节值、16字节值、32字节值和浮点值。在一些实施例中,基于寄存器混叠和/或操作的操作码来确定不同的操作数大小。例如,对8字节操作数的向量加法操作利用8字节向量寄存器。如关于图4A和图4B所更详细地解释的,寄存器混叠允许使用不同混叠来引用相同的存储器位置。例如,依据所需结果,32字节存储器块可以引用为单个4字节操作、两个2字节操作数或四个1字节操作数。在各种实施例中,向量计算单元的每个处理元件与其他处理元件并行执行相同的ALU操作(例如,加法、减法、乘法等)。在一些实施例中,输出结果是ALU结果的量化版本。例如,输出结果是量化版本,其与ALU结果相比,需要更少的字节来表示。在一些实施例中,使用比输入操作数少的字节代表的结果来计算ALU结果。例如,输入操作数各自可以是4个字节,输出结果的大小可以是1个字节。

[0089] 在663处,在661处执行的算术逻辑单元(ALU)操作的向量结果写入向量计算单元之外。在一些实施例中,利用输出缓冲器来写出向量结果,该输出缓冲器在输出总线不可用于接收数据的情况下允许处理以继续进行下一ALU操作。在一些实施例中,向量输出结果传送到后处理单元,诸如分别图1、图2和图3的后处理单元115,215和/或315。例如,执行ALU操作的结果写入到用于执行后处理池化操作的后处理单元。在一些实施例中,输出向量结果写入到诸如静态随机存取存储器(SRAM)的存储器。在各种实施例中,写出输出作为元素的向量,诸如96元素向量,其中每个元素的大小为1个字节。

[0090] 图7是图示了用于向量计算单元指令的编码格式的实施例的框图。在所示的示例中,向量计算单元指令710描绘了由单个指令指定的多个组件指令的编码。向量计算单元指令740进一步详述了由单个指令指定的多个组件指令中的每个组件指令的格式。向量计算

单元指令710是编码的指令三元组,并且包括加载操作711、算术逻辑单元 (ALU) 操作713和存储操作715。向量计算单元指令740包括字段:操作码741、寄存器743、操作码751、寄存器753、操作码配置字段755、立即位757、操作码761和寄存器763。向量计算单元指令710所描绘的(与加载操作、ALU操作和存储操作相对应的)组件指令的字段映射到向量计算单元指令740。向量计算单元指令740包括编码的加载操作(操作码741和寄存器743)、算术逻辑单元操作(操作码751、寄存器753、操作码配置字段755和立即位757)以及存储操作(操作码761和寄存器763)。

[0091] 在一些实施例中,向量计算单元指令是指定三个组件指令的指令三元组。例如,加载操作、算术逻辑单元 (ALU) 操作和存储操作可以使用128位格式捆绑为单个指令。在各种实施例中,视情况而定,更大或更小的位格式可以用于捆绑三个组件指令。在一些实施例中,加载操作和存储操作被编码为13个位,而ALU操作被编码为64个位。在各种实施例中,捆绑的加载操作、存储操作和ALU操作未使用的任何剩余字节是填充字节。在一些实施例中,操作码被编码为8个位,寄存器被编码为5个位,并且立即位被编码为32个位。在各种实施例中,视情况而定,可以利用不同长度的编码,并且这些不同长度的编码取决于指令大小、所支持的向量操作的数目、寄存器的数目、向量大小和/或其他适当的因素。在一些场景中,当没有利用组件指令中的一个或多个组件指令时,使用非操作操作。

[0092] 在所示的示例中,向量计算单元指令740的编码的加载操作包括操作码741和寄存器743。操作码741与向量加载操作相对应,而寄存器743是加载操作的对应的目的地向量寄存器。例如,操作码741可以用于存储用于加载数据的出队操作的操作码,并且寄存器743是用于存储加载的数据的目的地寄存器。在各种实施例中,加载操作用于将输入数据的向量加载到向量寄存器中,以由向量计算单元进行处理。在一些实施例中,操作码741是8位字段,并且寄存器743是5位字段。

[0093] 在所示的示例中,向量计算单元指令740的编码的存储操作包括操作码761和寄存器763。操作码761与向量存储操作相对应,寄存器763是对应的源向量寄存器,存储操作应当从该源向量寄存器读取数据向量。例如,操作码761可以用于存储用于存储操作的操作码,该存储操作将数据从寄存器763存储到诸如静态随机存取存储器 (SRAM) 的外部存储器。在一些实施例中,用于存储的存储器的起始地址由外部定序器或控制单元使用引用存储器位置的写入指针来维护。在一些实施例中,存储操作用于将数据向量写入输出数据总线。在一些实施例中,操作码761是8位字段,并且寄存器763是5位字段。

[0094] 在所示的示例中,编码算术逻辑单元 (ALU) 操作包括操作码751、寄存器753、操作码配置字段755和立即位757。操作码751用于对ALU操作码进行编码。例如,ALU操作码可以包括与加法(带符号的和不带符号的)、减法(带符号的和不带符号的)、乘法、绝对值和逻辑操作符等的向量操作相对应的操作码。依据向量ALU操作,该操作可以利用字段:寄存器753、操作码配置字段755和立即位757。在一些实施例中,寄存器753指定多达四个的向量寄存器,其包括三个源寄存器和一个目的地寄存器。在一些实施例中,寄存器753是20位字段,并且每个寄存器利用5位。

[0095] 在一些实施例中,编码算术逻辑单元 (ALU) 操作包括由某些ALU操作利用的操作码配置字段755。在一些实施例中,操作码配置字段755是5位字段,并且包括寄存器大小字段(2位)、屏蔽位(1位)和立即有效位(1位)。例如,在一些场景中,存储在寄存器大小字段(2

位)中的值可以用于指定寄存器的大小(例如,8位、16位或32位)。作为附加示例,屏蔽位(1位)可以用于处理立即位757作为位屏蔽,立即有效位(1位)可以用于标识立即位757的有效性。立即位757是32位字段,其用于需要立即位的ALU操作。例如,向量移动操作可以被配置为将32位值从立即位757移动到目的地向量寄存器。

[0096] 在一些实施例中,向量计算单元支持向量屏蔽移动指令(未示出)以将向量位屏蔽加载到向量屏蔽寄存器中。在一些实施例中,向量屏蔽移动指令包括对应操作码字段、目的地寄存器字段、以及立即位。例如,向量屏蔽移动将存储在立即位中的向量位屏蔽加载到向量屏蔽寄存器。在一些实施例中,由向量计算单元支持的向量的大小(例如,宽度为96个元素)需要足够大的立即位(例如,96个位)来存储位屏蔽。在一些实施例中,向量屏蔽移动指令不限于向量计算单元指令710和740的编码格式。例如,基于立即位的大小,向量屏蔽移动不会与其他组件指令捆绑在一起。

[0097] 在各种实施例中,向量计算单元指令中的组件指令使用图5的过程捆绑在一起。在一些实施例中,诸如图1、图2、图3和图4A的向量引擎111,211,311和/或向量计算单元401的向量计算单元利用图7的编码格式。在一些实施例中,向量计算单元指令通过包含定序器的微处理器系统或控制单元的定序器发布到向量计算单元。

[0098] 图8是图示了对于由向量计算单元执行单个向量计算单元指令的过程的实施例的流程图。可以由向量计算单元利用向量计算单元的处理元件对向量的元素并行执行图8的过程。在一些实施例中,分别通过诸如图1、图2、图3和图4A的向量引擎111,211,311和/或向量计算单元401的向量计算单元执行图8的过程。

[0099] 在801处,获取向量计算单元指令。在一些实施例中,从指令缓冲器和/或命令队列中获取指令。在各种实施例中,指令缓冲器包括引用要被执行的当前指令的一个或多个指针。在各种实施例中,指令缓冲器充当向量计算单元指令的高速缓存。

[0100] 在821处,对向量计算单元指令进行解码。例如,作为指令三元组的向量计算单元指令被解码成它的三个组件指令。在各种实施例中,对由每个组件指令所利用的参数和字段进行解码。例如,由寄存器字段指定的向量寄存器(诸如图7的寄存器753)被解码成源寄存器和目的地寄存器。

[0101] 在831处,发布组件指令。在一些实施例中,组件指令的发布包括确定资源和/或数据危害是否存在。在存在危害的情况下,在一些实施例中,向量计算单元等待危害被解决。例如,在由前一时钟周期中的加载操作引起的资源危害的情况下,向量计算单元为加载完成并且资源可用而等待一个或多个时钟周期。

[0102] 在一些实施例中,多个组件指令一起发布并且并行执行。例如,指令三元组的加载操作、算术逻辑单元(ALU)操作和存储操作在同一时钟周期期间一起执行。在一起执行组件指令的场景中,在同一时钟周期中发起与执行加载操作(步骤845)、ALU操作(步骤855)和存储操作(步骤865)相对应的步骤以及与非操作备选项(步骤843、854和863)对应的步骤中的每个步骤,并且执行并行进行。

[0103] 在一些实施例中,不同的组件指令以交错开始的方式执行。例如,在一些实施例中,首先执行加载操作,然后执行算术逻辑单元(ALU)操作,然后执行存储操作。在交错场景中,第一向量计算单元指令的ALU操作可以与下一向量计算单元指令的加载操作并行执行。

[0104] 在各种实施例中,包括不同的算术逻辑单元(ALU)的操作的不同的操作要花费一

个或多个时钟周期来完成,并且不能保证不同的操作能够在同一时钟周期结束时完成。在一些实施例中,可以在同一指令周期期间执行获取步骤(步骤801)、解码步骤(步骤821)和发布步骤(步骤831)步骤中的一个或多个步骤。

[0105] 在841处,做出向量计算单元指令是否包括加载操作的确定。例如,在一些场景中,可以使用非操作操作替换加载操作以指示不应当执行加载操作。响应于非操作操作,处理继续进行到843。在存在加载操作的情况下,处理继续进行到845。

[0106] 在843处,对非操作操作进行处理并且不执行加载操作。例如,841处的指令中不存在加载指令,而是使用非操作操作的操作码。

[0107] 在845处,向量计算单元执行加载操作。例如,执行出队操作以从先进先出队列(诸如向量引擎输入队列207)加载输入向量。

[0108] 在851处,确定向量计算单元指令是否包括算术逻辑单元(ALU)操作。例如,在一些场景中,可以使用非操作操作替换ALU操作以指示不应执行ALU操作。响应于非操作操作,处理继续进行到853。在存在ALU操作的情况下,处理继续进行到855。

[0109] 在853处,对非操作操作进行处理并且不执行算术逻辑单元(ALU)操作。例如,851处的指令中不存在ALU指令,而是使用非操作操作的操作码。

[0110] 在855处,向量计算单元执行算术逻辑单元(ALU)操作。例如,响应于向量加法操作,向量计算单元的算术逻辑单元执行向量加法操作以将两个源向量寄存器的内容相加,并且将结果存储在目的地向量寄存器中。在一些实施例中,向量计算单元的算术逻辑单元是图4A的算术逻辑单元(ALU) 427。

[0111] 在861,做出向量计算单位指令是否包括存储操作的确定。例如,在一些场景中,可以使用非操作操作替换存储操作以指示不应当执行存储操作。响应于非操作操作,处理继续进行到863。在存在存储操作的情况下,处理继续进行到865。

[0112] 在863处,对非操作操作进行处理并且不执行存储操作。例如,861处的指令中不存在存储指令,而是使用非操作操作的操作码。

[0113] 在865处,向量计算单元执行存储操作。例如,执行将向量寄存器中的向量数据存储在存储器的存储操作。

[0114] 图9是图示了向量计算单元的示例性指令周期的图。图9的过程图示了并行执行但呈交错开始的方式的三个向量计算单元指令的示例排序和顺序。在一些实施例中,图1、图2、图3和图4A的向量引擎111,211,311和/或向量计算单元401分别利用图9的示例性指令周期。在图9的示例中,捆绑在一起作为单个指令的组件指令以交错开始的方式被执行,使得首先执行加载操作,然后执行算术逻辑单元(ALU)操作,然后执行存储操作。在一些实施例中,一系列向量计算单元指令呈流水线形式,但是组件指令并行执行,并且不会遵循图9中描绘的交错开始。

[0115] 在所示的示例中,第一指令周期910包括获取步骤911、解码步骤921、发布步骤931、加载执行步骤941、算术逻辑单元(ALU)执行步骤951、以及与第一向量计算单元指令相对应的存储执行步骤961。第二指令周期920包括获取步骤923、解码步骤933、发布步骤943、加载执行步骤953、算术逻辑单元(ALU)执行步骤963、以及与第二向量计算单元指令相对应的存储执行步骤973。第三指令周期930包括提取步骤935、解码步骤945、发布步骤955、加载执行步骤965、算术逻辑单元(ALU)执行步骤975、以及与第三向量计算单元指令相对应的存

储执行步骤985。在一些实施例中,竖直虚线是时钟周期边界。在各种实施例中,在相同时钟周期期间开始相同时钟周期边界内的步骤。

[0116] 在一些实施例中,指令周期的开始错开一个阶段。例如,与第二指令周期920相比较,第一指令周期910在处理上领先一个阶段,并且比第三指令周期930领先两个阶段。在任何给定的时钟周期期间,不同的向量计算单元指令可以利用与不同阶段相关联的硬件资源:获取、解码、发布、加载执行、算术逻辑单元 (ALU) 执行、以及存储执行。作为示例,第一、第二和第三指令周期910,920和930的发布阶段931、解码阶段933和获取阶段935分别在同一时钟周期期间执行。作为另一示例,分别在同一时钟周期期间执行第一、第二和第三指令周期910,920和930的存储执行步骤961、算术逻辑单元 (ALU) 执行步骤963和加载执行步骤965。

[0117] 在一些实施例中,向量计算单元的指令周期实现了每个时钟周期一个向量计算单元指令的吞吐量。在一些实施例中,获取步骤、解码步骤和/或发布步骤被压缩为单个时钟周期。例如,在一些实施例中,指令缓冲器用于使获取时间最小,并且一起执行获取步骤和解码步骤。在一些实施例中,指令周期的每个阶段可以花费一个或多个时钟周期来完成。在一些实施例中,阶段本身呈流水线形式。例如,在执行步骤花费一个以上的周期来完成的情况下,执行步骤可以呈流水线形式以在多个时钟周期内完成。在一些实施例中,可以以流水线形式并行处理多个执行步骤,并且每个执行步骤可以与不同的向量计算单元指令相对应。在一些实施例中,获取步骤911,923和935与图8的步骤801相对应,解码步骤921,933和945与图8的步骤821相对应,发布步骤931,943和955与图8的步骤831相对应,加载执行步骤941,953和965与图8的步骤845相对应,算术逻辑单元 (ALU) 执行步骤951,963和975与图8的步骤855相对应,并且存储执行步骤961,973和985与图8的步骤865相对应。

[0118] 在备选实施例(未示出)中,以与图9相同的次序执行指令周期的获取阶段,解码阶段和发布阶段。与图9的示例性实施例相反,在同一时钟周期期间一起并行执行加载、算术逻辑单元 (ALU) 和存储执行步骤。例如,一起执行同一向量计算单元指令的加载执行步骤941、ALU执行步骤951和存储执行步骤961。

[0119] 图10是图示了计算阵列的计算单元的实施例的框图。在所示的示例中,计算单元1000包括输入值,即,权重1002、数据1004和ResultIn 1006;信号,即,ClearAcc信号1008、时钟信号1010、ResultEnable信号1012、ResultCapture信号1014和ShiftEn信号1016;组件,即,累加器1024、多路复用器1026、影子寄存器1028、乘法器1030和加法器1032;逻辑1034,1036和1038;以及输出值,即,ResultOut 1050。在一些实施例中,逻辑1034,1036和1038是AND门。在一些实施例中,视情况而定,附加信号包括在内。在各种实施例中,对于诸如图1的矩阵处理器107的计算阵列的多个计算单元中的每个计算单元(诸如计算单元109),重复图10的计算单元。计算单元1000可以用于实现并行计算操作。在各种实施例中,计算阵列的每个计算单元与其他计算单元并行执行计算。在各种实施例中,计算单元1000是矩阵处理器的子电路,其包括用于执行一个或多个乘法操作、加法操作、累加操作和/或移位操作的功能。例如,计算单元1000可以是包括用于执行点积操作的功能的子电路。在各种实施例中,计算单元1000是图1的计算单元109和/或图2的计算单元209和/或221-229。

[0120] 在一些实施例中,时钟信号1010是计算单元1000所接收的时钟信号。在各种实施例中,计算阵列的每个计算单元接收同一时钟信号,并且该时钟信号用于使每个计算单元

的处理与其他计算单元的处理同步。

[0121] 在所示的示例中,乘法器1030接收输入值(数据1004和权重1002)并且对其执行乘法操作。乘法器1030的输出馈送到加法器1032。加法器1032接收乘法器1030的输出和逻辑1034的输出并且对这些输出执行加法。加法器1032的输出馈送到累加器1024。在一些实施例中,输入值(数据1004和权重1002)是与计算单元交叉的线并且将对应的数据和/或权重馈送到相邻的计算单元。例如,在一些实施例中,数据1004馈送到同一列中的所有计算单元,而权重1002馈送到同一行中的所有计算单元。在各种实施例中,数据1004和权重1002分别与从数据输入103和权重输入105馈送到计算单元1000的输入元素相对应。在各种实施例中,数据1004和权重1002分别与从数据硬件数据格式化器和权重硬件数据格式化器馈送到计算单元1000的输入元素相对应。

[0122] 在一些实施例中,ClearAcc信号1008清零累加器1024的内容。作为示例,累加操作可以通过清零累加器1024来重置,并且用于累加乘法器1030的结果。在一些实施例中,ClearAcc信号1008用于清零用于执行新点积操作的累加器1024。例如,通过乘法器1030执行逐元素乘法,并且使用加法器1032和累加器1024将部分点积结果相加。

[0123] 在各种实施例中,累加器1024是能够累加加法器1032的结果并且间接地累加乘法器1030的结果的累加器。例如,在一些实施例中,累加器1024被配置为基于ClearAcc信号1008的状态来累加乘法器1030的结果与累加器1024的内容。作为另一示例,基于ClearAcc信号1008的状态,加法器1032可以忽略存储在累加器1024中的当前结果。在所示的示例中,累加器1024是宽度为32位的累加器。在各种实施例中,视情况而定,累加器1024的大小可以不同,例如,8位、16位、64位等。在各种实施例中,计算阵列的多个计算单元中的每个累加器的大小相同。在各种实施例中,累加器1024可以累加并且保存数据,累加并且清零数据,或仅清零数据。在一些实施例中,累加器1024可以被实现为累加寄存器。在一些实施例中,累加器1024可以包括一组算术逻辑单元(ALU),该组算术逻辑单元包括寄存器。

[0124] 在一些实施例中,响应于确定数据1004有效,激活ResultEnable信号1012。例如,可以启用ResultEnable信号1012以使得能够通过计算单元进行处理,诸如使得能够通过乘法器1030和加法器1032进行处理进入累加器1024。

[0125] 在一些实施例中,ResultCapture信号1014用于确定多路复用器1026的功能。多路复用器1026接收ResultIn 1006、累加器1024的输出和ResultCapture信号1014作为输入。在各种实施例中,ResultCapture信号1014用于使得ResultIn 1006或累加器1024的输出能够作为多路复用器1026的输出通过。在一些实施例中,多路复用器1026被实现为输出寄存器。在一些实施例中,ResultIn 1006连接到与计算单元1000的相同列中的计算单元。例如,相邻计算单元的输出作为输入值ResultIn 1006馈送到计算单元1000。在一些实施例中,相邻计算单元的输入是计算单元的对应ResultOut值。

[0126] 在一些实施例中,影子寄存器1028接收多路复用器1026的输出作为输入。在一些实施例中,影子寄存器1028被配置为取决于ResultCapture信号1014的值经由多路复用器1026接收累加器1024的输出。在所示的示例中,影子寄存器1028的输出是输出值ResultOut1050。在各种实施例中,一旦结果插入到影子寄存器1028中,累加器1024就可以用于着手新计算。例如,一旦最终点积结果存储在影子寄存器1028中,累加器1024就可以被清零,并且用来累加并且存储部分结果,并且最终累加并且存储对新输入值(权重和数据)

进行新点积操作的最终结果。在所示的示例中,影子寄存器1028接收信号ShiftEn信号1016。在各种实施例中,ShiftEn信号1016用于启用或禁用值在影子寄存器1028中的存储。在一些实施例中,ShiftEn信号1016用于将存储在影子寄存器1028中的值移位到输出值ResultOut1050。例如,当启用ShiftEn信号1016时,存储在影子寄存器1028中的值从影子寄存器1028中移出作为输出值ResultOut1050。在一些实施例中,ResultOut 1050连接到相邻计算单元的输入值ResultIn。在一些实施例中,计算单元的列的最后一个单元连接到计算阵列的输出。在各种实施例中,计算阵列的输出馈送到向量引擎,诸如图1的向量引擎111,以进行向量处理。例如,诸如图1的计算单元109的计算单元的输出ResultOut 1050可以馈送到诸如图1的向量引擎111的处理元件113的向量引擎的处理元件中。

[0127] 在所示的示例中,影子寄存器1028的宽度为32位。在各种实施例中,视情况而定,影子寄存器1028的大小可以不同,例如,8位、16位、64位等。在各种实施例中,计算阵列的多个计算单元的每个影子寄存器的大小相同。在各种实施例中,影子寄存器1028的大小与累加器1024的大小相同。在各种实施例中,多路复用器1026的大小基于累加器1024和/或影子寄存器1028的大小(例如,大小相同或更大)。

[0128] 在一些实施例中,逻辑1034,1036和1038接收诸如控制信号的信号,以启用和/或配置计算单元1000的功能。在各种实施例中,逻辑1034,1036和1038使用AND门和/或与AND门相对应的功能来实现。例如,如上文所描述的,逻辑1034接收ClearAcc信号1008以及与存储在累加器1024中的值相对应的输入值。基于ClearAcc信号1008,确定逻辑1034的输出并且将其馈送到加法器1032。作为另一示例,逻辑1036接收ResultEnable信号1012和Clock信号1010。基于ResultEnable信号1012,确定逻辑1036的输出并且将其馈送到累加器1024。作为另一示例,逻辑1038接收ShiftEn信号1016和Clock信号1010。基于ShiftEn信号1016,确定逻辑1038的输出并且将其馈送到影子寄存器1028。

[0129] 在各种实施例中,计算单元可以同时(即,在单个周期内)执行乘法操作、加法操作和移位操作,从而使每个周期发生的操作的总数加倍。在一些实施例中,结果在单个时钟周期内从多路复用器1026移动到影子寄存器1028,即,无需中间执行操作和保存操作。在各种实施例中,时钟周期基于在时钟信号1010处接收到的信号。

[0130] 在各种实施例中,输入值(权重1002和数据1004)是8位值。在一些实施例中,权重1002是带符号的值,而数据1004是不带符号的值。在各种实施例中,视情况而定,权重1002和数据1004可以带符号或不带符号。在一些实施例中,ResultIn 1006和ResultOut 1050是32位值。在各种实施例中,使用比输入操作数(权重1002和数据1004)更多的位来实现ResultIn 1006和ResultOut1050。通过利用大量位,可以累加多对权重1002和数据1004相乘例如以计算点积结果的结果,而不会溢出标量结果。

[0131] 在一些实施例中,计算单元1000在累加器1024中生成中间和/或最终计算结果。然后,最终计算结果经由多路复用器1026被存储在影子寄存器1028中。在一些实施例中,多路复用器1026用作输出寄存器,并且存储累加器1024的输出。在各种实施例中,最终的计算结果是卷积操作的结果。例如,ResultOut 1050处的最终结果是由计算单元1000接收的使用权重1002作为输入值的滤波器与由计算单元1000接收的使用数据1004作为输入值的传感器数据的二维区域之间的卷积的结果。

[0132] 作为示例,可以使用计算单元1000对与传感器数据的区域相对应的2x2数据输入

矩阵 $[d0\ d1; d2\ d3]$ 以及与权重 $[w0\ w1; w2\ w3]$ 的矩阵相对应的过滤器执行卷积操作。 2×2 数据输入矩阵具有第一行 $[d0\ d1]$ 和第二行 $[d2\ d3]$ 。滤波器矩阵具有第一行 $[w0\ w1]$ 和第二行 $[w2\ w3]$ 。在各种实施例中,计算单元1000经由数据1004接收数据矩阵作为一维输入向量 $[d0\ d1\ d2\ d3]$ (每个时钟周期一个元素),并且经由权重1002接收权重矩阵作为一维输入向量 $[w0\ w1\ w2\ w3]$ (每个时钟周期一个元素)。使用计算单元1000,执行两个输入向量的点积以在ResultOut 1050处产生标量结果。例如,乘法器1030用于将输入权重和数据向量的每个对应元素相乘,并且结果存储到累加器1024并且与累加器1024中的先前结果相加。例如,元素 $d0$ 与元素 $w0$ 相乘(例如, $d0 \times w0$)的结果首先存储在已清零的累加器1024中。接下来,元素 $d1$ 与元素 $w1$ 相乘,并且使用加法器1032与存储在累加器1024中(例如, $d0 \times w0$)的先前结果相加,以计算 $d0 \times w0 + d1 \times w1$ 的等效值。处理继续进行到第三对元素 $d2$ 和 $w2$,以在累加器1024处计算 $d0 \times w0 + d1 \times w1 + d2 \times w2$ 的等效值。最后元素对相乘,点积的最终结果现在存储在累加器1024中(例如, $d0 \times w0 + d1 \times w1 + d2 \times w2 + d3 \times w3$)。然后,点积结果拷贝到影子寄存器1028。一旦存储在影子寄存器1028中,就可以例如使用传感器数据的不同区域来启动新点积操作。基于ShiftEn信号1016,存储在影子寄存器1028中的点积结果移位离开影子寄存器1028到达ResultOut1050。在各种实施例中,权重和数据矩阵的尺寸可以与上述示例的尺寸不同。例如,可以使用更大的尺寸。

[0133] 在一些实施例中,引入偏差参数,并且使用累加器1024与点积结果相加。在一些实施例中,在权重1002或数据1004处该偏差参数作为输入而被接收,以及乘法标识元素作为其他输入值。偏差参数与标识元素相乘以保留偏差参数,并且使用加法器1032将乘法结果(例如,偏差参数)与点积结果相加。加法结果(偏差值所抵消的点积结果)存储在累加器1024中,稍后使用影子寄存器1028在ResultOut1050处移出。在一些实施例中,使用诸如图1的向量引擎111的向量引擎引入偏差。

[0134] 尽管出于理解清楚的目的,对前述实施例进行了一些详细描述,但是本发明不限于所提供的细节。存在许多实施本发明的备选方式。所公开的实施例是说明性的而非限制性的。

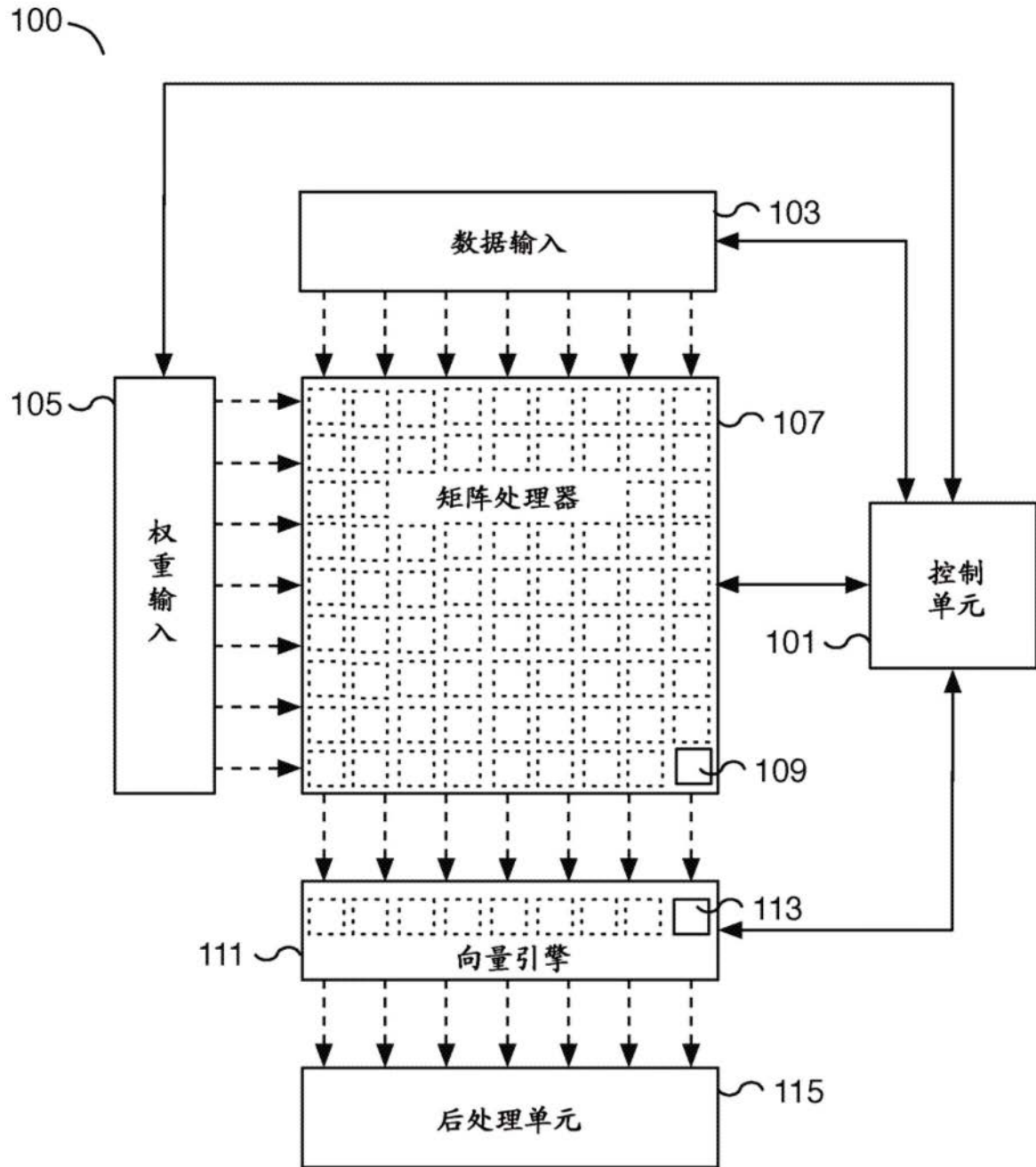


图1

200

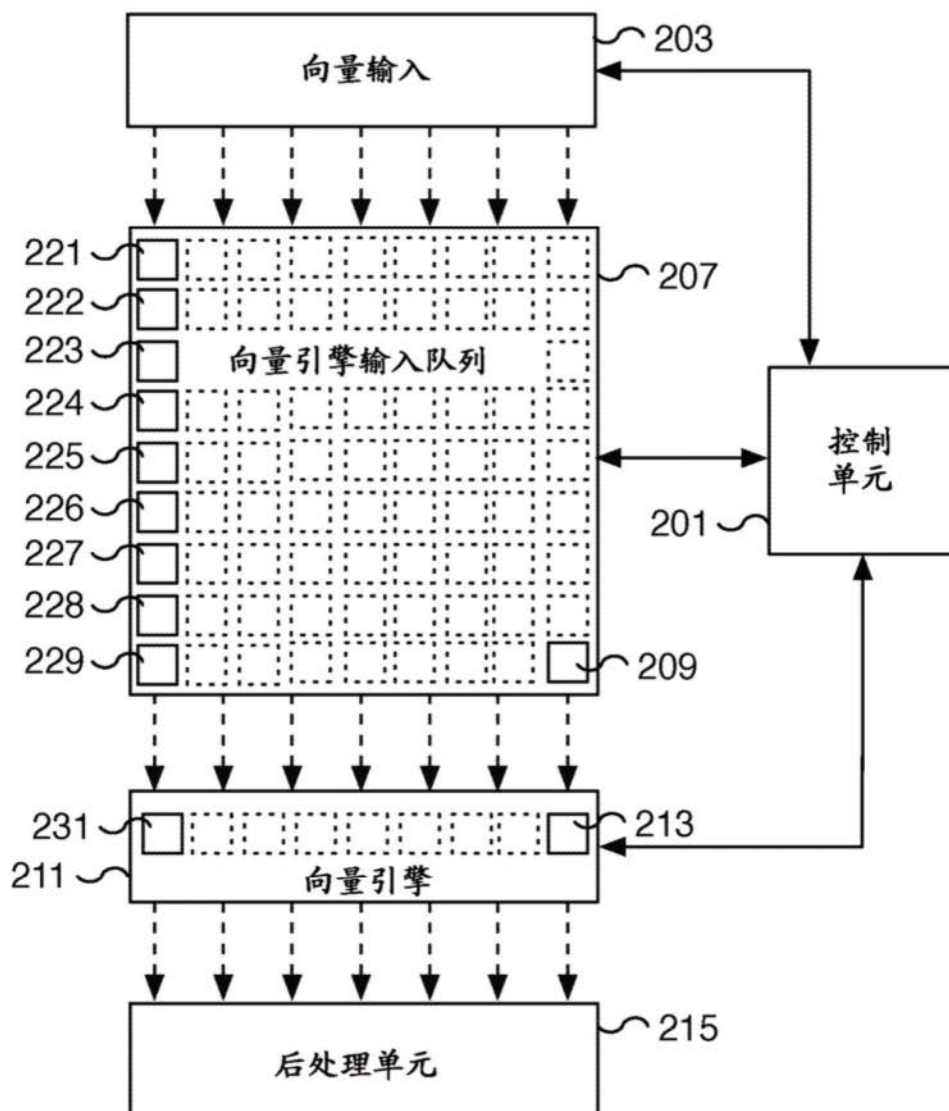


图2

300

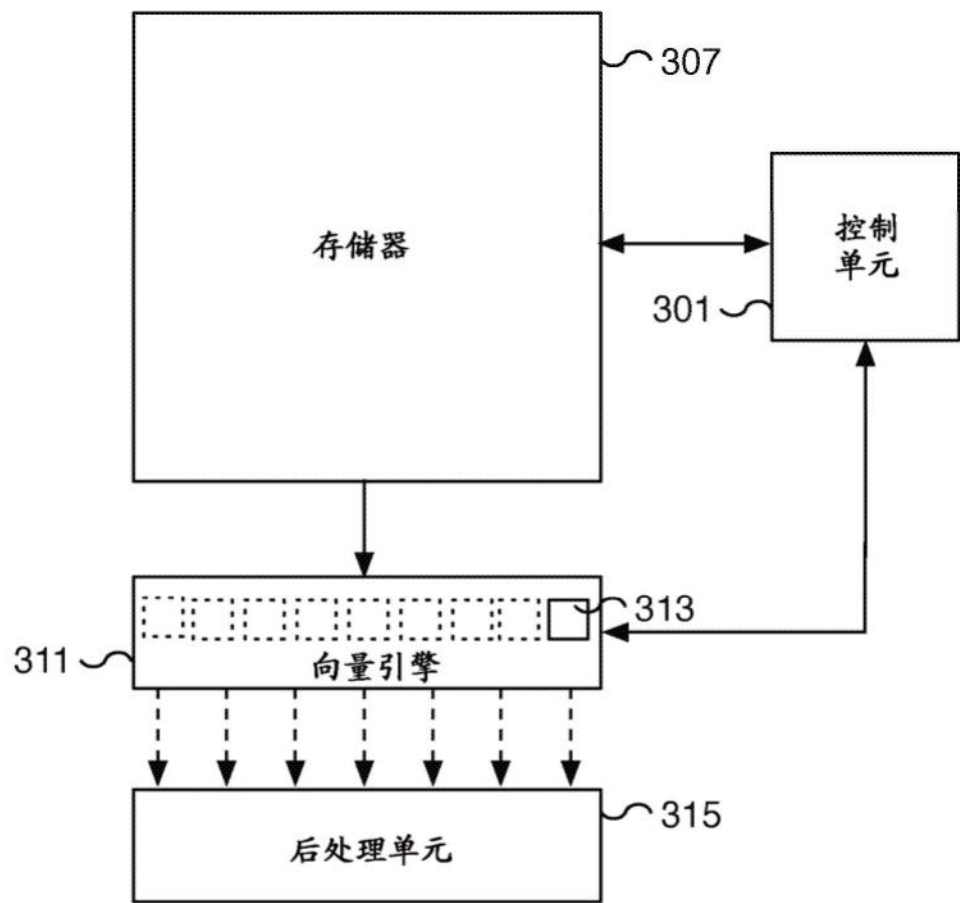


图3

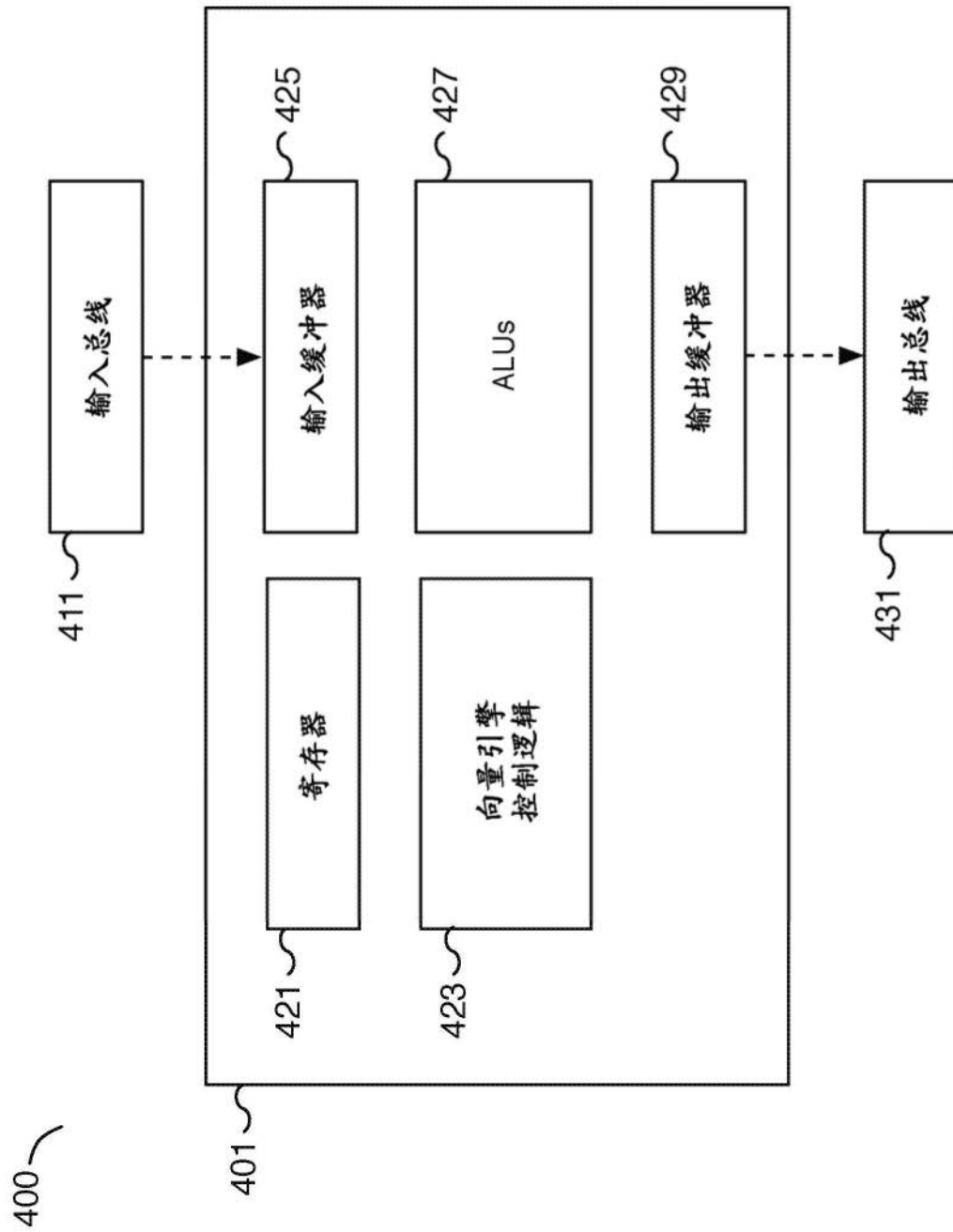


图4A

表 450

451 ~	字节3	字节2	字节1	字节0
453 ~	RD0			
455 ~	RW1		RW0	
457 ~	RB3	RB2	RB1	RB0
463 ~	RD1			
465 ~	RW3		RW2	
467 ~	RB7	RB6	RB5	RB4
473 ~	RD7			
475 ~	RW15		RW14	
477 ~	RB31	RB30	RB29	RB28

图4B

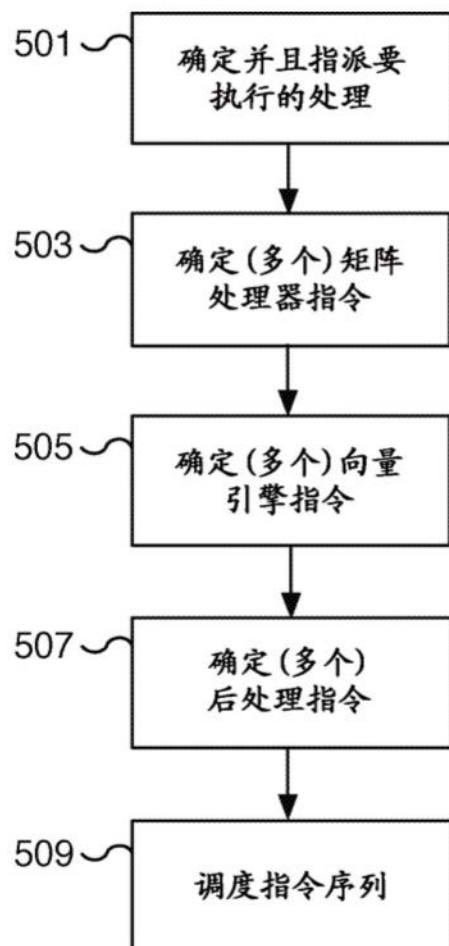


图5

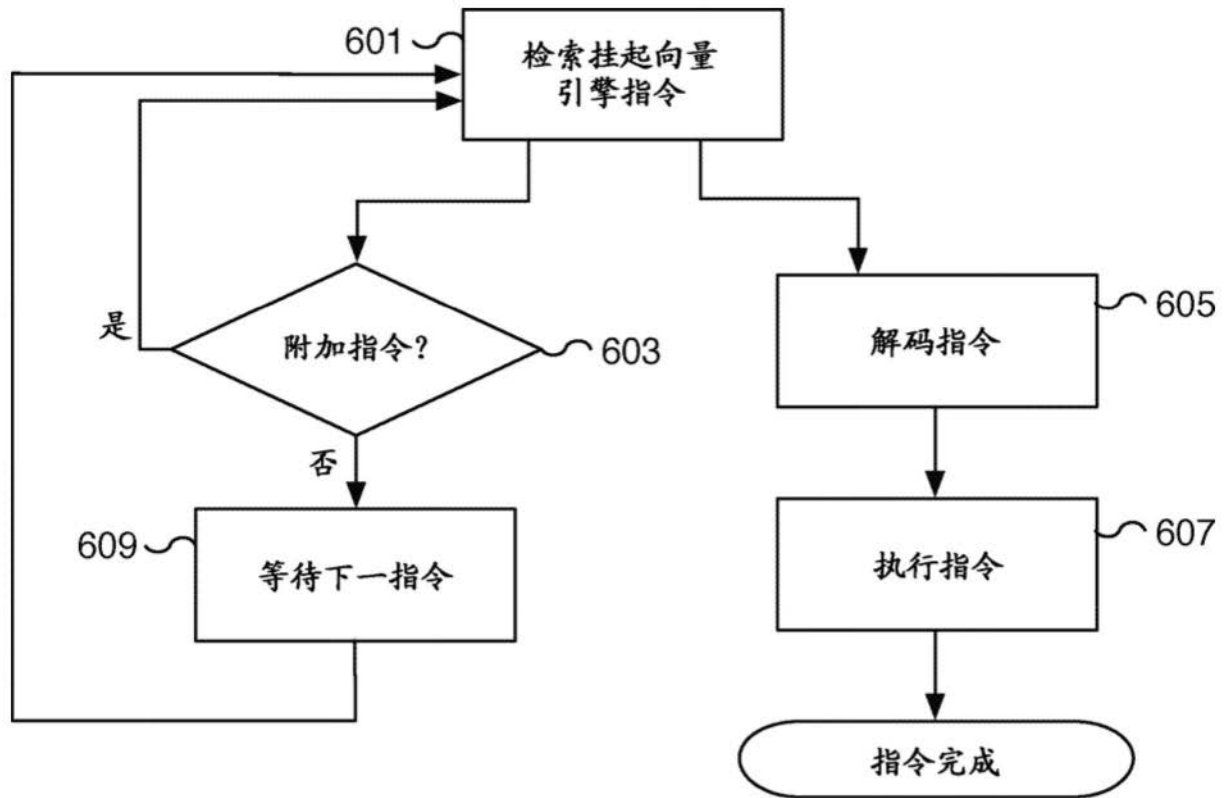


图6A

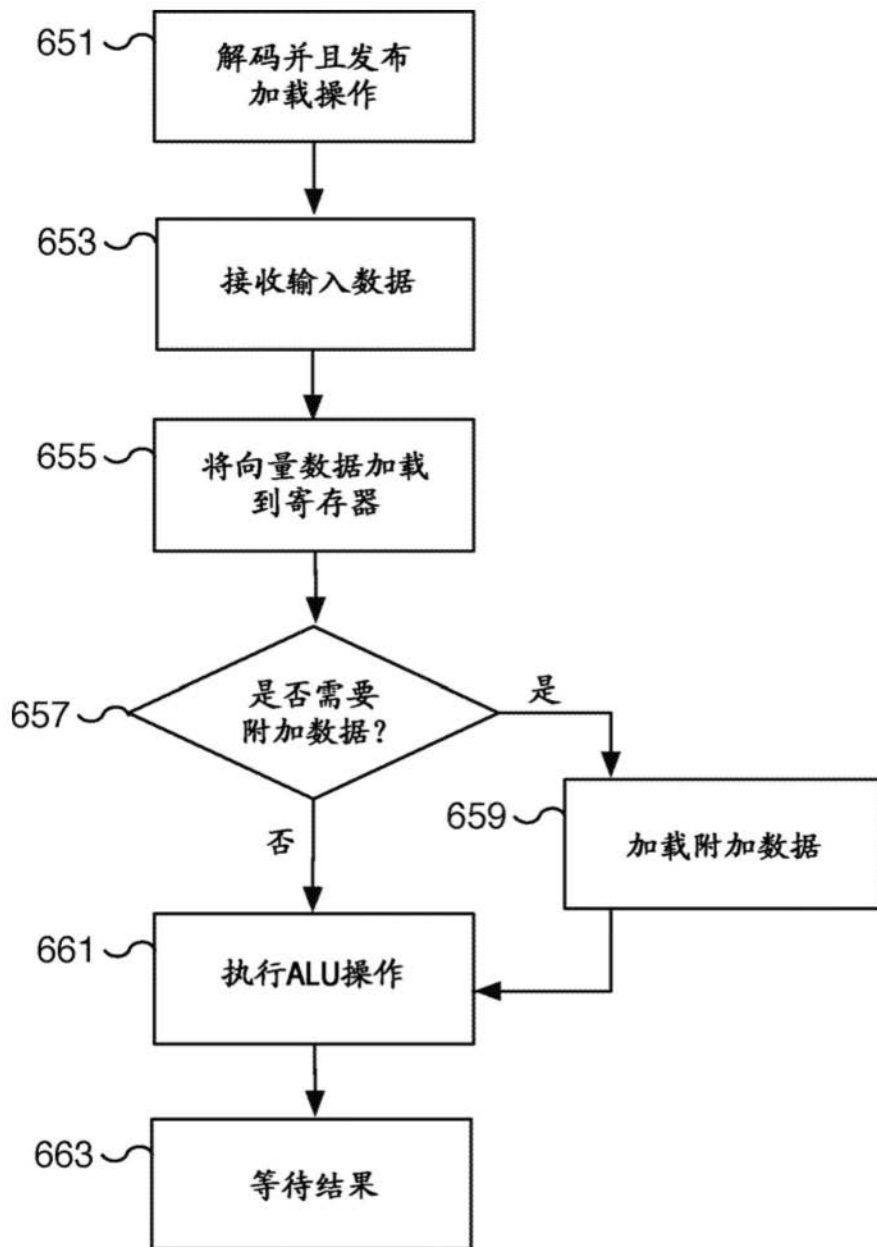


图6B

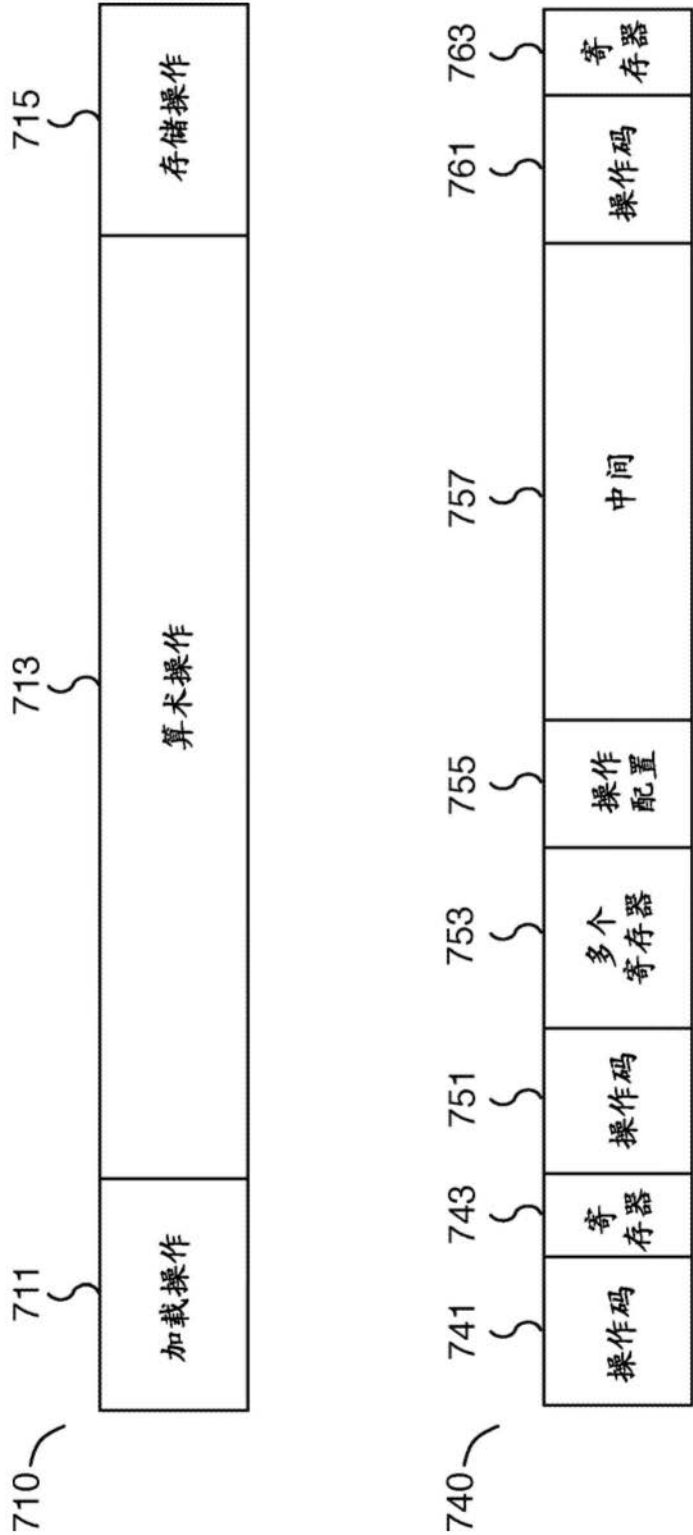


图7

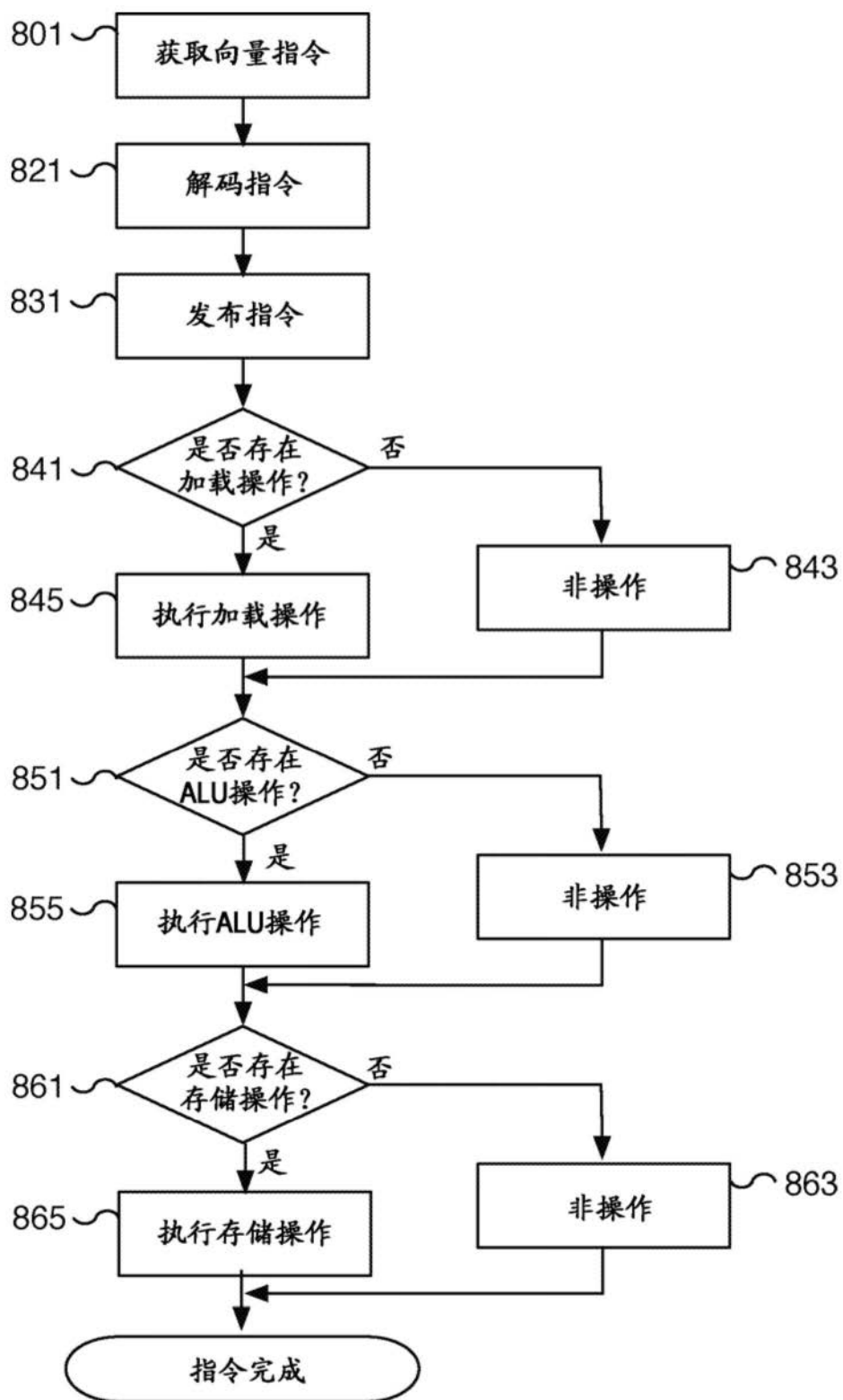


图8

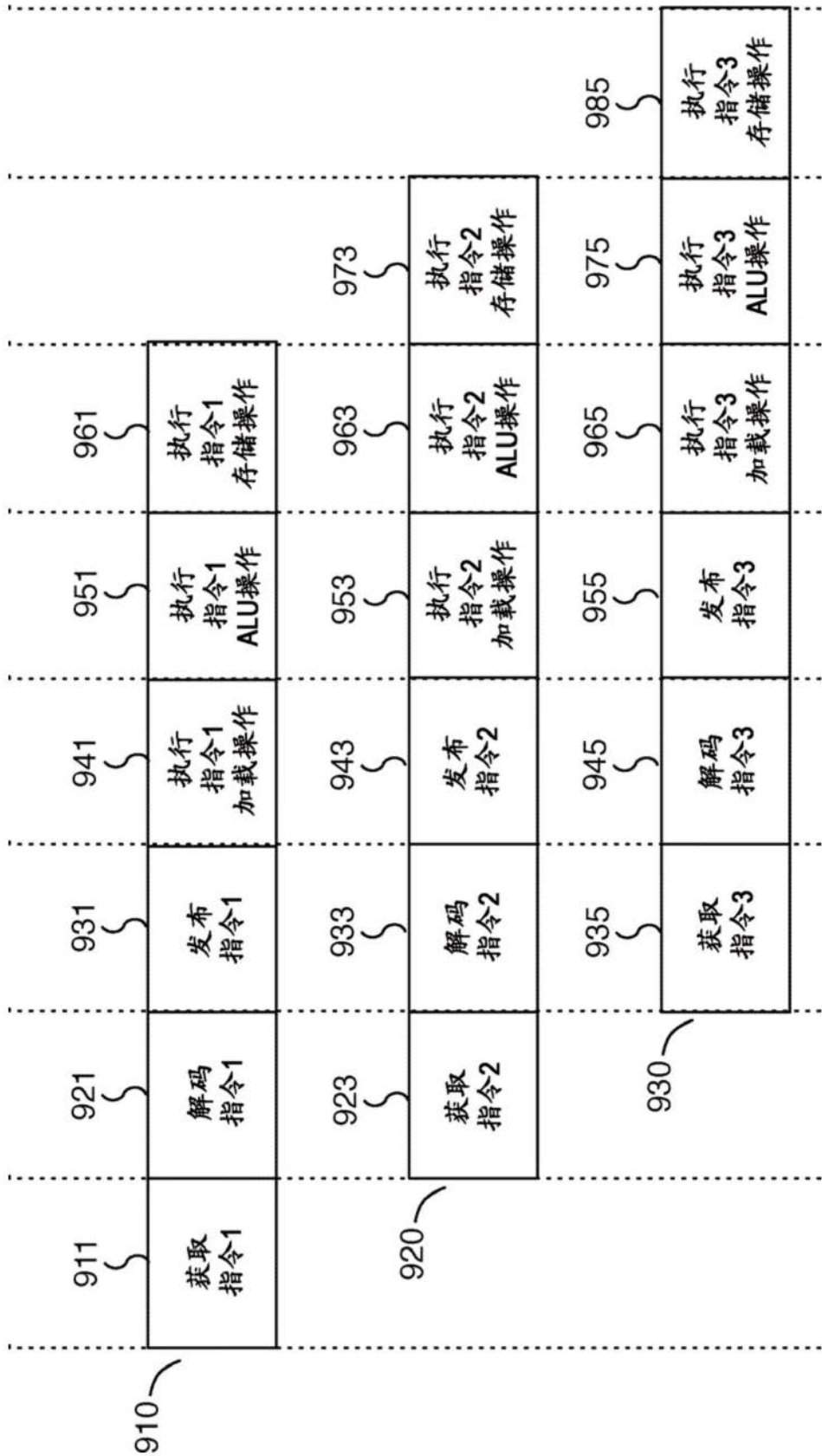


图9

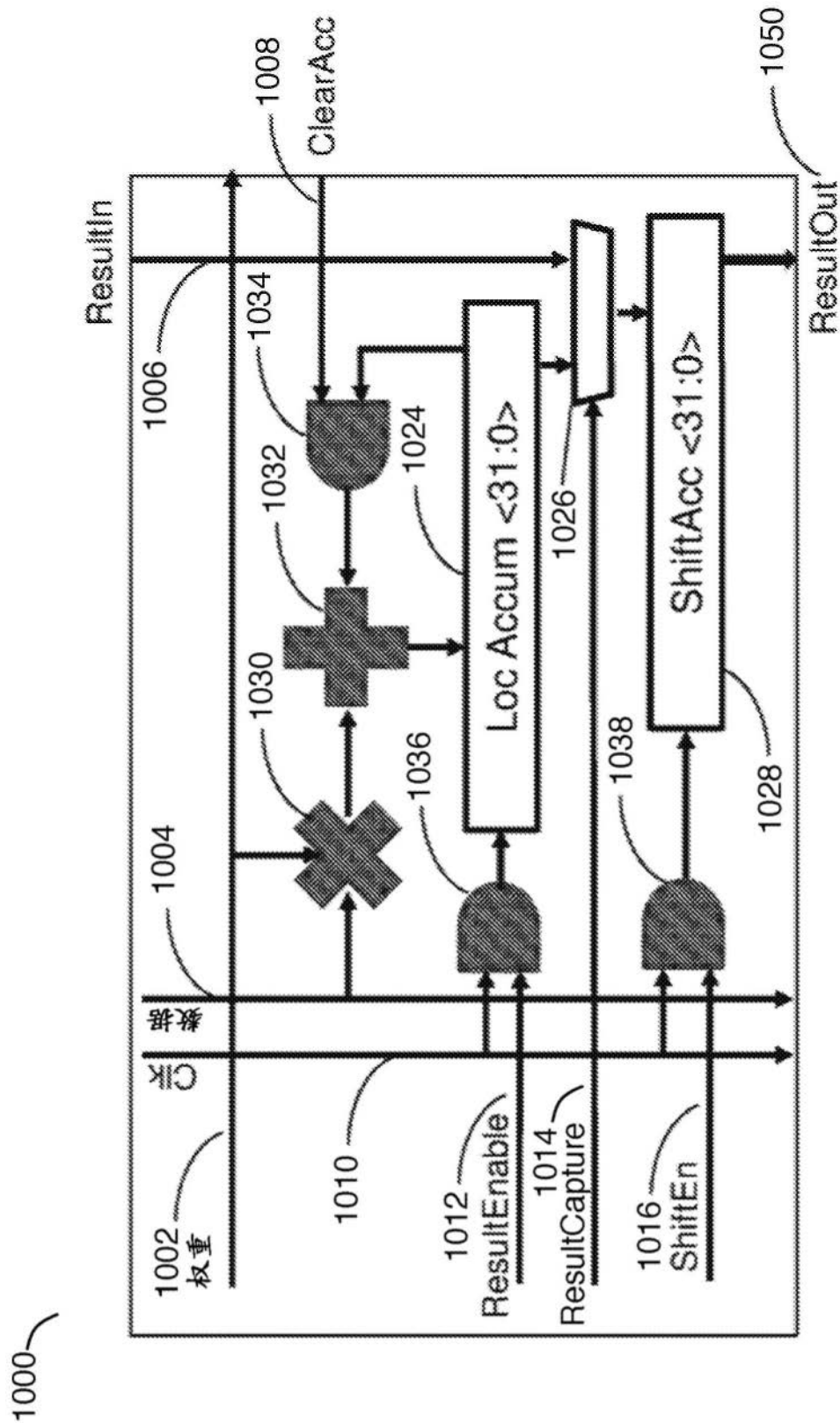


图10