(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 111950695 A
(43)申请公布日 2020.11.17

(21)申请号 202010413990.7

(22)申请日 2020.05.15

(30)优先权数据
16/413,293 2019.05.15 US

(71)申请人 辉达公司
地址 美国加利福尼亚州

(72)发明人 刘洺堉 K·林

(74)专利代理机构 北京市磐华律师事务所 11336
代理人 高伟

(51)Int.Cl.
G06N 3/04(2006.01)
G06N 3/08(2006.01)
G06K 9/62(2006.01)

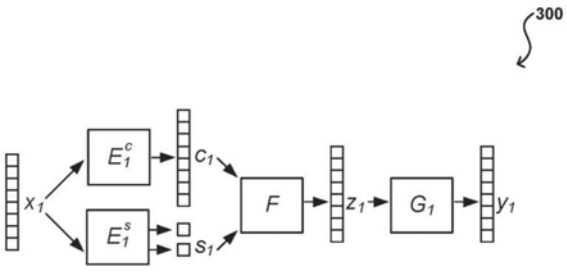
权利要求书3页 说明书59页 附图36页

(54)发明名称

使用一个或更多个神经网络进行语法迁移

(57)摘要

公开了使用一个或更多个神经网络进行语法迁移。用于在句子之间迁移语法的装置、系统和技术。在至少一个实施例中，使用一个或更多个神经网络将一个或更多个第一句子翻译成具有不同语法的一个或更多个第二句子。



CN 111950695 A

1.一种处理器,包括:

一个或更多个算术逻辑单元(ALU),所述一个或更多个ALU被配置为使用一个或更多个神经网络将一个或更多个第一句子翻译成一个或更多个第二句子,所述一个或更多个第二句子具有与所述一个或更多个第一句子不同的语法。

2.根据权利要求1所述的处理器,其中所述一个或更多个ALU进一步被配置为:

使用所述一个或更多个神经网络中的至少一个,确定所述一个或更多个第一句子中的每一个的内容代码,所述内容代码提供各自的第一句子的内容的与风格无关的表达。

3.根据权利要求2所述的处理器,其中所述一个或更多个ALU进一步被配置为:

使用所述一个或更多个神经网络中的至少一个,确定用于所述一个或更多个第二句子的风格代码,所述风格代码从各自的句子推断出。

4.根据权利要求3所述的处理器,其中所述一个或更多个ALU进一步被配置为:

使用所述内容代码和所述风格代码生成所述一个或更多个第二句子的潜在表示;以及

使用至少一个解码神经网络,将所述潜在表示解码成所述一个或更多个第二句子。

5.根据权利要求4所述的处理器,其中所述一个或更多个ALU进一步被配置为:

使用传递函数生成所述潜在表示,所述传递函数将所述风格代码的偏差应用于所述内容代码的归一化均值。

6.根据权利要求1所述的处理器,其中所述不同的语法对应于所述一个或更多个第一句子的内容的不同表达,所述不同表达在风格、情感、结构或类型中的至少一个方面不同。

7.根据权利要求1所述的处理器,其中所述一个或更多个第一句子中的至少一个被接收为音频数据并转换为文本数据,或者所述一个或更多个第二句子中的至少一个被提供为使用对应于各自的风格代码的数字语音生成的音频数据。

8.根据权利要求1所述的处理器,其中所述一个或更多个ALU进一步被配置为:

在训练所述一个或更多个神经网络的至少子集期间,最小化循环一致性损失。

9.一种系统,包括:

一个或更多个处理器,所述一个或更多个处理器被配置为至少部分地基于一个或更多个神经网络将一个或更多个第一句子翻译成一个或更多个第二句子,所述一个或更多个第二句子具有与所述一个或更多个第一句子不同的语法;以及

一个或更多个内存,用于存储所述一个或更多个神经网络。

10.根据权利要求9所述的系统,其中所述一个或更多个处理器进一步被配置为:

使用所述一个或更多个神经网络中的至少一个,确定所述一个或更多个第一句子中的每一个的内容代码,所述内容代码提供各自的第一句子的内容的与风格无关的表达。

11.根据权利要求10所述的系统,其中所述一个或更多个处理器进一步被配置为:

使用所述一个或更多个神经网络中的至少一个确定将用于所述一个或更多个第二句子的风格代码,所述风格代码从各自的句子推断出。

12.根据权利要求11所述的系统,其中所述一个或更多个处理器进一步被配置为:

使用所述内容代码和所述风格代码生成所述一个或更多个第二句子的潜在表示;以及

使用至少一个解码神经网络,将所述潜在表示解码成所述一个或更多个第二句子。

13.根据权利要求12所述的系统,其中所述一个或更多个处理器进一步被配置为:

使用传递函数生成所述潜在表示,所述传递函数将所述风格代码的偏差应用于所述内

容代码的归一化均值。

14.根据权利要求9所述的系统,其中所述不同的语法对应于所述一个或更多个第一句子的内容的不同表达,所述不同表达在风格、情感、结构或类型中的至少一个方面不同。

15.根据权利要求9所述的系统,其中所述一个或更多个第一句子中的至少一个被接收为音频数据并转换为文本数据,或者所述一个或更多个第二句子中的至少一个被提供为使用对应于各自的风格代码的数字语音生成的音频数据。

16.根据权利要求9所述的系统,其中所述一个或更多个处理器进一步被配置为:

在训练所述一个或更多个神经网络的至少子集期间,最小化循环一致性损失。

17.一种处理器,包括:

一个或更多个算术逻辑单元(ALU),用于训练一个或更多个神经网络以将一个或更多个第一句子翻译成一个或更多个第二句子,所述一个或更多个第二句子具有与所述一个或更多个第一句子不同的语法。

18.根据权利要求17所述的处理器,其中所述一个或更多个ALU进一步被配置为:

使用所述一个或更多个神经网络中的至少一个,确定所述一个或更多个第一句子中的每一个的内容代码,所述内容代码提供各自的第一句子的内容的与风格无关的表达。

19.根据权利要求18所述的处理器,其中所述一个或更多个ALU进一步被配置为:

使用所述一个或更多个神经网络中的至少一个确定要用于所述一个或更多个第二句子的风格代码,所述风格代码从各自的句子推断出。

20.根据权利要求19所述的处理器,其中所述一个或更多个ALU进一步被配置为:

使用所述内容代码和所述风格代码生成所述一个或更多个第二句子的潜在表示;以及

使用至少一个解码神经网络将所述潜在表示解码成所述一个或更多个第二句子。

21.根据权利要求20所述的处理器,其中所述一个或更多个ALU进一步被配置为:

使用传递函数生成所述潜在表示,所述传递函数将所述风格代码的偏差应用于所述内容代码的归一化均值。

22.根据权利要求17所述的处理器,其中所述不同的语法对应于所述一个或更多个第一句子的内容的不同表达,所述不同表达在风格、情感、结构或类型中的至少一个方面不同。

23.根据权利要求17所述的处理器,其中所述一个或更多个第一句子中的至少一个被接收为音频数据并转换为文本数据,或者所述一个或更多个第二句子中的至少一个被提供为使用对应于各自的风格代码的数字语音生成的音频数据。

24.根据权利要求17所述的处理器,其中所述一个或更多个ALU进一步被配置为:

在训练所述一个或更多个神经网络的至少子集期间,最小化循环一致性损失。

25.一种系统,包括:

一个或更多个处理器,用于至少部分地使用循环一致性损失函数计算与一个或更多个神经网络相对应的参数,所述循环一致性损失函数被应用以将一个或更多个第一句子翻译成一个或更多个第二句子,所述一个或更多个第二句子具有与所述一个或更多个第一句子不同的语法;以及

一个或更多个内存,用于存储所述参数。

26.根据权利要求25所述的系统,其中所述一个或更多个处理器进一步被配置为:

使用所述一个或更多个神经网络中的至少一个确定所述一个或更多个第一句子中的每一个的内容代码,所述内容代码提供各自的第一句子的内容的与风格无关的表达。

27.根据权利要求26所述的系统,其中所述一个或更多个处理器进一步被配置为:

使用所述一个或更多个神经网络中的至少一个确定要用于所述一个或更多个第二句子的风格代码,所述风格代码从各自的句子推断出。

28.根据权利要求27所述的系统,其中所述一个或更多个处理器进一步被配置为:

使用所述内容代码和所述风格代码生成所述一个或更多个第二句子的潜在表示;以及

使用至少一个解码神经网络将所述潜在表示解码成所述一个或更多个第二句子。

29.根据权利要求28所述的系统,其中所述一个或更多个处理器进一步被配置为:

使用传递函数生成所述潜在表示,所述传递函数将所述风格代码的偏差应用于所述内容代码的归一化均值。

30.根据权利要求25所述的系统,其中所述不同的语法对应于所述一个或更多个第一句子的内容的不同表达,所述不同表达在风格、情感、结构或类型中的至少一个方面不同。

31.根据权利要求25所述的系统,其中所述一个或更多个第一句子中的至少一个被接收为音频数据并转换为文本数据,或者所述一个或更多个第二句子中的至少一个被提供为使用对应于各自的风格代码的数字语音生成的音频数据。

32.根据权利要求25所述的系统,其中所述一个或更多个处理器进一步被配置为:

在训练所述一个或更多个神经网络的至少子集期间,最小化循环一致性损失。

使用一个或更多个神经网络进行语法迁移

背景技术

[0001] 越来越多的内容在以电子方式生成、存储和修改。在某些情况 下,可能想要获取(take)某些内容并以不同的方式表达该内容。例 如,可能想要获取句子的一般内容或思想并以不同的方式表达出来，例如以传达不同的意图或以不同的格式呈现内容。虽然存在常规的、基于规则的方法来进行特定类型的更改,但这些方法在执行此类转换 的能力方面非常有限。

附图说明

[0002] 将参考附图描述根据本公开的各种实施例,其中：

[0003] 图1示出了根据至少一个实施例的使用一个或更多个神经网络 生成输入句子的示例语法翻译；

[0004] 图2A和图2B示出了根据至少一个实施例的使用内容和风格代 码执行语法翻译的示例方法；

[0005] 图3A和图3B示出了根据至少一个实施例的可用于执行语法翻 译的示例组件；

[0006] 图4示出了根据至少一个实施例的可用于执行语法翻译的示例 组件；

[0007] 图5示出了根据至少一个实施例的用于将句子翻译为具有可利 用的不同语法表达的示例过程；

[0008] 图6示出了根据至少一个实施例的用于使用可以利用的经训练 的神经网络来生成推理的示例过程；

[0009] 图7示出了可以在其中实现各个实施例的各方面的示例环境；

[0010] 图8示出了根据至少一个实施例的用于训练可以利用的图像合 成网络的示例系统；

[0011] 图9示出了根据至少一个实施例的可以利用的示例统计模型的 各层；

[0012] 图10示出了根据至少一个实施例的推理和/或训练逻辑；

[0013] 图11示出了根据至少一个实施例的推理和/或训练逻辑；

图12示出了根据至少一个实施例的深度神经网络的训练和部 署；

[0014] 图13示出了根据至少一个实施例的示例数据中心系统；

[0015] 图14是示出了根据至少一个实施例的计算机系统的框图；

[0016] 图15是示出了根据至少一个实施例的计算机系统的框图；

[0017] 图16示出了根据至少一个实施例的计算机系统；

[0018] 图17示出了根据至少一个实施例的计算机系统；

[0019] 图18示出了根据本文所述的各个实施例的示例性集成电路和 相关的图形处理器,其可以使用一个或更多个IP核心来制造。

[0020] 图19A-图19B示出了根据本文所述的各个实施例的示例性集 成电路和相关的图形处理器,其可以使用一个或更多个IP核心来制 造。

[0021] 图20A-图20B示出了根据本文所述实施例的附加的示例性图 形处理器逻辑；

[0022] 图21示出了根据至少一个实施例的计算机系统；

[0023] 图22示出了根据至少一个实施例的并行处理器；

[0024] 图23示出了根据至少一个实施例的分区单元；

[0025] 图24示出了根据至少一个实施例的处理集群；

[0026] 图25示出了根据至少一个实施例的图形多处理器；

[0027] 图26是示出根据至少一个实施例的用于处理器的处理器微架 构的框图；

[0028] 图27示出了根据至少一个实施例的深度学习应用处理器；

[0029] 图28是示出根据至少一个实施例的示例神经形态处理器的框 图；

[0030] 图29示出了根据一个或更多个实施例的图形处理器的至少部 分；

[0031] 图30是根据至少一个实施例的图形处理器核心的至少部分的 框图；

[0032] 图31A和图31B示出了根据至少一个实施例的线程执行逻辑；

[0033] 图32示出了根据至少一个实施例的并行处理单元(“PPU”)；

[0034] 图33示出了根据至少一个实施例的通用处理集群(“GPC”)；

[0035] 图34示出了根据至少一个实施例的并行处理单元(“PPU”)的 内存分区单元；以及

[0036] 图35示出了根据至少一个实施例的流式多处理器。

具体实施方式

[0037] 在下面的描述中，将描述各个实施例。出于说明的目的，阐述 了具体的配置和细节，以便提供对实施例的彻底理解。然而，对于本 领域技术人员也显而易见的是，实施例可以在没有具体细节的情况下 实践。此外，为了不使所描述的实施例模糊，可以省略或简化众所周 知的特征。

[0038] 根据各个实施例的方法提供了机器学习的训练和使用，例如可 涉及一个或更多个神经网络，用于诸如语法迁移(grammar transfer) 之类的任务。根据各个实施例的方法可以接受输入句子或其他文本表 达，并生成一个或更多个输出句子或其他文本表达，每个输出句子或 其他文本表达具有不同的语法。输入句子的内容可以用神经网络来确 定，不同的风格代码可以与内容相结合，生成一组潜在表示，每个潜 在表示将内容与不同的风格相关联。风格可以包括不同方式的表达内 容，也可以包括不同的文本风格、情感、句子结构等。解码器网络可 以获取(take)该潜在表示，同时生成一个或更多个输出句子，这些 输出句子使用对应于各种风格代码的不同表达的不同语法来表达输 入句子的内容。

[0039] 各种其他功能可以在各个实施例中实现，并且在本文的其他地 方也可以讨论和建议。

[0040] 数字内容可以以许多不同的方式进行操纵。例如，通过应用各 种过滤器或基于规则的转换，可以修改图像以提供不同的外观或具有 不同的风格。操纵文本内容可能更具挑战性，因为这些方法必须能够 区分内容和表达。图1示出了可以为同一输入句子生成的表达的示例 集100。在本例中，输入句子102表示“我再也不会去这家餐厅了”。 例如，这可能是刚刚访问过这家餐厅并在社交媒体上发表评论的用户 提供的句子。这也可能来自不同的其他来源，如媒体公司或出版商。 出于多种原因，可能想要更改该句的表达。例如，可能想要将陈述句 改为疑问句，例如可能会说“我会再去这家餐厅吗?”。可以看出，基 本内容是一样的，但表达不同。另一个可能的表达106是“我迫不及 待地想回到这家餐厅!”。在这

个示例中,表达不仅从消极情感变成 了积极情感,而且由于感叹号和短语的使用,也从一个相对没有感情 的陈述句变成了一个充满兴奋的句子。第三个表达108可能更适合用 于对餐厅的评论,它可以获取输入句子,而不是陈述诸如“这家餐厅 需要改进才能有回头客”之类的事情。这种转换保持了情感,但改变 了表达的结构。另一个示例表达110是将输入的风格转换为诗歌风格 的结果。可以看出,基本内容是由输入句子确定的,然后,内容可以以多种不同的方式表达,包括表达的不同方面,可以包括如风格、情 感和结构等。这在本文中通常被称为使用不同的“语法(grammar)” 来表达内容,这可能导致任何这些或其他的影响所用单词的设置和顺 序的变化。

[0041] 在各个实施例中,文本迁移(transfer)或转换(transform)是 指使用不同的语言语法对给定文本重新措辞的任务。转换框架可用于 从非平行文本学习文本语法迁移模型。与以往学习将输入句子转化为 一个输出句子的一对一映射的方法不同,这种框架可以学习可以将输 入句子转化为多个输出句子或表达的一对多映射。虽然讨论将集中于 作为示例的句子,但是应当理解,单词和字符的各种其他分组可以与 在本文中别处讨论和建议的各个实施例一起使用,并且根据本文所包 含的教导和建议,对于本领域普通技术人员来说是显而易见的。在一 些实施例中,通过将对抗训练与潜在分解方案相结合,可以实现学习一对多映射的能力。具体地说,输入句子的潜在表示可以被分解为捕 获其语言风格变化的“风格”代码和编码与风格无关的内容的内容代 码。本文中使用的“风格”将广泛地包括可能影响用于表达具体内容的 语法的任何事物,例如可包括句子类型、情感、结构等。然后,内容 代码可以与目标域中的一个或更多个句子(即,随机抽样或指定的) 的风格代码组合,以生成文本语法迁移输出。通过将相同的内容代码 与一个或更多个不同句子的风格代码组合,可以生成不同的表达作为 输出,每个表达可以使用不同的语法。

[0042] 这样的方法可以用于,例如,将输入句子的语言风格更改为目 标风格,并限制与风格无关的内容在整个迁移过程中应保持不变。有 着广泛的潜在应用,但正如前面提到的,许多常规方法将文本风格迁 移机制视为一对一的映射函数,其将一种语言风格的输入句子转化为 特定目标语言风格的单个对应句子。如上所述,这种方法提供的有用 性有限。

[0043] 根据各个实施例的方法可以代替地执行本文所称的一对多映射 建模。然而,应当理解,在各个实施例中,输入可以被迁移到任意数 量的不同文本表达中,并且不受特定映射或其他这样的方面的限制。 一对多的方法可以利用这样一个事实:同一个问题可以用几种不同的 方式来描述。例如,在对真空产品进行评论时,第一句话可能会说, “这种轻便的真空非常有效”,而第二句话可能会说,“这种容易携带 的真空能很好地阻挡灰尘和垃圾。”这两句话都可以被认为基本上包 含了相同的正面评论内容,但是用不同的语法以不同的方式表达。类 似地,该内容可能有多种表达方式,对相同的基本内容有负面情感。 这种一对多的映射性质对于更抽象的风格迁移任务尤其重要,例如歌 词和浪漫描写之间的文本风格迁移,或者形式和幽默描写之间的文本 风格迁移。因此,将文本风格迁移建模为一对多映射可以提供许多好 处,例如,能够在推理时为用户提供若干表达选择,以及便于学习更 准确的文本风格迁移模型,因为一对多映射更准确地描述了文本风格 迁移机制。

[0044] 在一个实施例中,使用非并行文本训练框架。这种情况下的训 练数据可以由两个不同风格的语料库(corpus)组成,没有成对的输 入和输出句子可用。相应的框架可以建立在通过对抗训练学习的潜在 分解方案上,以将句子的潜在表示分解为两部分,一部分编码

句子的 风格，而另一部分编码与风格无关的内容。对于迁移，首先从输入句 子中提取内容代码。然后从目标风格语料库的训练数据集中对句子采 样并提取其风格代码。内容代码和风格代码被组合起来以生成输出句 子，该输出句子将携带相同的内容，但具有目标风格并使用相应的语 法表达。对不同风格的句子进行采样提供了不同的风格代码，其用于 生成不同的风格迁移输出。

[0045] 图2A和图2B示出了根据各个实施例的可用于执行语法转移的 一个这样的方法。在这个例子中，X_1和X_2对应于不同语言风格的句子 的两个域。分别设S_1和S_2为控制X_1和X_2中风格变化的潜在空间。设C为 X_1和X_2共享的潜在空间，其控制句子中与风格无关的变化。在图2A 的状态200中，分析输入句子X_1以将该句子分解为表示输入句子的内 容或基本概念的内容代码c_1以及表示输入句子的风格的风格代码s_1。 对于产品评论迁移任务示例，X_1和X_2可以分别表示积极和消极的产品 评论的空间，C中的元素编码在句子中评论的产品及其特征，S_1中的 元素表示积极风格的变化，例如偏好程度和写作风格。为了实现一对 多映射，可以假设句子x_1可以被分解为内容码$c_1 \in C$和风格代码 $s_1 \in S_1$。如图2B的示例状态250所示，还可以假设句子x_1可以通过 融合其内容代码c_1和其风格代码s_1来重构。风格迁移可以通过融合内 容代码c_1与风格代码s_2来实现，其中s_2从目标风格空间S_2被采样。如 图2B所示，通过向所提取的输入句子的内容代码应用不同的风格代 码，可以生成不同输出语法的多个变体。

[0046] 在这个公式中，文本风格迁移机制是由条件分布$p(x_{1\to 2}|x_1)$给出 的，其中$x_{1\to 2}$是通过将句子x_1迁移到目标域X_2生成的句子。如前所述， 一些常规方法强制将风格迁移实施为一对一映射，其将输入句子转化 为仅单个对应的输出句子。因此，它们的条件分布可以简化为狄拉克 δ函数等。这样的一对一映射无法生成不同风格迁移输出。

[0047] 图3A和图3B示出了根据各个实施例的可使用的示例语法迁移 框架的视图300、350。图3A示出了在训练时通过组件的示例流，图 3B示出了在推理时通过组件的示例流。训练时间示例使用具有所提 取的内容代码和风格代码的单个输入句子。对于推理时间示例，有两 个输入句子X_1和X_2。X_1是用户输入，X_2是从具有不同表达的句子的数 据库中采样的。对于风格代码s_2，示出了两个正方形，其表示风格代 码的均值和偏差值。用户输入X_1被转化成一系列代码，其以使得嵌入 的单词知道句子中的其他单词的方式表示每个单词的潜在表示。在这 个表示C_1中，每个正方形表示向量，并且C_1的每个向量可以被认为 是其分布具有均值和偏差的多维高斯分布。均值和偏差控制风格和文 本。C_1有均值和偏差，其可以被归一化为单位偏差。然后，来自S_2的均值和分布可以被用于移位分布，使具有来自X_1的均值和来自S_2的偏差。然后这可以导致改变内容的风格，例如从消极情感改变为积 极情感。然后，C_1可以被映射到称为$Z_{1\to 2}$的新的潜在表示。解码器 G_2然后可以将潜在表示迁移为新句子。在本例中，F是一个函数，它 提供了一种特定的方法来使用前面提到的风格代码的均值和偏差来 迁移风格。在本例中，E和G是网络。任何适当的语言模型技术都可 以使用，可以包含基于长短期记忆(LSTM)的神经网络。

[0048] 示例模型包括用于每个域X_i的内容编码器E_i^c、风格编码器E_i^s和 解码器G_i。内容编码器E_i以m个元素的序列$x_i = \{x_i^1, x_i^2, \ldots, x_i^m\}$作为输 入，并计算内容代码$c_i = \{c_i^1, c_i^2, \ldots, c_i^m\} = E_i^c(x_i)$，它是描述句子含义的 向量序列。风格编码器将$x_i$转化为风格

代码$s_i \equiv (s_{i,\mu}, s_{i,\sigma}) = E_i^s(x_i)$，这 是一对向量。注意$s_{i,\mu}$和$s_{i,\sigma}$将用作输入$x_i$的特征激活的新均值和标准偏 差。为了计算文本风格迁移输出$y_{i\to j} = \{y_{i\to j}^1, y_{i\to j}^2, \ldots, y_{i\to j}^{n_j}\}$，其中$n_j$不一 定等于m，我们首先使用组合函数F将内容代码$c_i$和风格代码$s_j$组合 起来，得到$z_{i\to j} = \{z_{i\to j}^1, z_{i\to j}^2, \ldots, z_{i\to j}^m\}$。然后，解码器$G_j$将表示$z_{i\to j}$映射到 输出序列$y_{i\to j}$。注意使用$E_j^s$从$X_j$随机采样的$x_j$中提取$s_j$。通过采样不同 的$x_j$，将有不同的$s_j$，因此具有不同的风格迁移输出$y_{i\to j}$。

[0049] 至少一些实施例利用可以通过控制神经网络中特征激活的均值 和方差来实现风格迁移的事实。采用这种方法，潜在代码组合函数F 可以由以下公式给出：

[0050] $$F(c_i^k, s_j) = s_{j,\mu} \otimes \left(c_i^k - \mu(c_i)\right) \oslash \sigma(c_i) + s_{j,\sigma},$$

[0051] 其中⊗表示元素级积，⊘表示元素级除法，μ(•)和σ(•)表示通过 将c_i中的每个向量视为随机变量的独立实现来计算内容潜码(latent code)的均值和标准偏差的操作。也就是说，潜在表示$z_{i\to j}^k = F(c_i^k, s_j)$可 以通过首先在潜在空间中对内容代码$c_i$进行归一化，然后应用非线性 变换来构造，非线性变换的参数由目标风格的句子提供。合成函数不包含可学习的参数，并且在本例中可以被视为解码器的一部分。

[0052] 在一些实施例中，可以使用卷积神经网络(CNN)来实现内容 编码器E_i^c。为了确保输出序列c的长度等于输入句子的长度，可以用 左右两侧的m-1个零向量填充输入，其中m是输入序列的长度。对于 卷积操作，在至少一些实施例中不使用跨步卷积。风格编码器E_j^c也可 以使用卷积网络来实现。为了提取风格代码，可以在几个卷积层之后 应用全局平均池化。然后，使用两层多层感知将结果投影到$s_{i,\mu}$和$s_{i,\sigma}$。 对数指数非线性可被应用于计算$s_{i,\sigma}$，以确保输出严格为正，这是对 偏差建模所需要的。解码器G_i可以使用具有注意力机制的卷积网络和 卷积序列到序列网络来实现。组合函数F在解码器输入处可以被实现 为数据归一化层。

[0053] 示例文本语法迁移模型可以使用若干对象术语进行训练。在一 些实施例中，重构损失可用于正则化文本风格迁移学习。具体地说， 可以假定内容编码器E_i^c和风格编码器E_i^s以及解码器G_i对形成自动编码 器。可以通过最小化训练语料库的负对数似然来训练该对：

[0054] $$\mathcal{L}_{rec}^i = E_{x_i}\left[-\log P\left(y_i^k \middle| x_i^k;\ \theta_{E_i^c}, \theta_{E_i^s}, \theta_{G_i}\right)\right]$$

[0055] 其中，变量$\theta_{E_i^c}$、 $\theta_{E_i^s}$和θ_{G_i}分别表示内容编码器、风格编码器和解码器的参 数。

[0056] 对于每个训练句子，G_i通过基于潜在表示$z_i \equiv \{z_i^1, z_i^2, \ldots, z_i^m\}$，和先 前的输出预测$\{y^1, y^2, \ldots, y^{t-1}\}$来预测最可能的令牌(token)$y^t$来合成输 出序列，因此可以通过以下公式计算句子的概率：

$$P\left(y\middle|x;\theta_{E_i^c},\theta_{E_i^s},\theta_{G_i}\right)=$$

[0057] $$\prod_{t=1}^{T}p\left(y^t\middle|z_i,y^1,y^2,\ldots,y^{t-1};\theta_{G_i}\right),$$

[0058] 其中t表示标记索引,T是句子长度。通过使用softmax函数对解码器 输出进行线性投影,计算令牌预测的概率。

[0059] 在至少一些实施例中,还可以使用循环一致性损失来保持输入 的内容。循环一致性损失可以进一步用于正则化学习,这可以基于内 容代码应在迁移过程中保留的思想。为了实现这一目标,如图4的示 例配置400所示,可以将输入x_1迁移到另一风格域x_2,然后使用其原 始风格代码s_1迁移回到原始域x_1。通过这样做,得到的句子$x_{1\to2\to1}$应 该与原始输入x_1非常相似。对于输入x_1,内容代码和风格代码是确定 的。生成经采样文本的风格代码s_2的均值和标准偏差。如前所述,函 数F可用于组合内容代码和风格代码。然而,不是解码器G_2生成句 子,而是示出了一对激活层402。网络激活是可微的实数,辅助训练。 这些激活值可以直接传递给编码器E_2。理想情况下,$C_{1\to2}$应该与C_1相同,这样,通过函数F将原始风格代码S_1与$C_{1\to2}$组合起来,将得 到一个值$x_{1\to2\to1}$,该值与输入x_1基本相同。例如,积极的句子应该能 够被改成消极的句子并返回,从而产生本质上相同的句子。这称为循 环一致性,可用于训练网络。

[0060] 为了使内容代码独立于风格,将x_1和$x_{1\to2\to1}$之间的差异最小化 (从而使循环一致性最大化)。优化问题可以表述为以下循环一致性 损失函数:

[0061] $$\mathcal{L}_{cyc}^1=E_{x_1,x_2}\left[-\log P(x_1^k\middle|x_{1\to2\to1}^k;\,\theta,)\right]$$

[0062] $$\mathcal{L}_{cyc}^2=E_{x_1,x_2}\left[-\log P(x_2^k\middle|x_{2\to1\to2}^k;\,\theta,)\right]$$

[0063] 其中,$\underline{\theta}=\{\theta_{E_i^c},\theta_{E_i^s},\theta_{G_1},\theta_{E_2},\theta_{E_2^s},\theta_{G_2}\}$

[0064] 为了避免波束搜索的不可微性,在最小化循环一致性损失的情况下, 可以使用解码器G_2和内容编码器E_1^c之间的一组可微非线性变换来代 替$x_{1\to2}$的硬解码。具体地说,非线性变换将解码器G_2的倒数第二个层 的特征激活投影到内容编码器E_1^c的第二层。这些非线性投影由多层感 知器(MLP)学习,多层感知器与文本风格迁移任务一起训练。这样, 在至少一些实施例中,可以纯粹地使用反向传播来训练模型和更新网 络参数。

[0065] 为了确保MLP将特征激活正确地投影到E_2^c的第二层,可以强制 MLP的输出尽可能类似于E_1^c的第二层的特征激活。在至少一些实施 例中,这是基于这样的思想,即在迁移过程中x_1和$x_{1\to2}$应具有相同的 内容代码,并且它们在内容编码器中的特征激活也应该相同。因此, 可以应用均方误差(MSE)损失函数来实现这一目标,如下所示:

[0066] $$\mathcal{L}_{mse}^1=E_{x_1,x_2}\left[\left\|h(E_2^c)(x_{1\to2})-h(E_1^c)(x_1)\right\|_2^2\right]$$

[0067] 其中$h(E_1^c)$和$h(E_2^c)$分别表示用于计算E_1^c和E_2^c的第二层的特征激活的函 数。另一个域的损失$\mathcal{L}_{mse}^2$被相应地定义。在学习过程中,风格分类损 失可以用标准的交叉熵损失

$\mathcal{L}_{cls}^{i}$强制施加在风格代码$s_1 = E_i^s(x_i)$上。这 鼓励风格代码s_i捕获输入句子的风格属性。

[0068] 一些实施例使用生成式对抗网络(GAN)将来自重构流的解码 器的输入潜码的分布与来自翻译流的解码器的输入潜码的分布相匹 配。也就是说,存在$z_{i\to j}$的分布与z_j的分布相匹配,这有助于确保迁 移输出的分布与目标风格句子的分布相匹配,因为它们使用相同的解 码器。由于对抗训练被应用于潜在表征,因此可以避免波束搜索的不 可微性。第二域的对抗损失可由下式给出:

[0069] $$\mathcal{L}_{adv}^{2} = \mathbb{E}_{x_1,x_2}\left[\log\left(1 - D_2(z_{1\to 2})\right)\right] + \mathbb{E}x_2\left[\log\left(D_2(z_2)\right)\right]$$

[0070] 其中D_2是旨在区分句子$z_{1\to 2}$与$z_2 = C_z(c_2, s_2)$的潜在表示的鉴别器。对 抗损失$\mathcal{L}_{adv}^{1}$以类似的方式被定义。总损失函数由下式给出:

[0071] $$\min_{E_1,E_2,G_1,G_2,} \max_{D_1,D_2,} \mathcal{L} = \sum_{i=1}^{2} \mathcal{L}_{rec}^{i} + \mathcal{L}_{cyc}^{i} + \mathcal{L}_{mse}^{i} + \mathcal{L}_{cls}^{i} + \mathcal{L}_{adv}^{i}$$

[0072] 由于文本风格迁移是一项多目标任务,因此存在可以用于评估 模型的性能的几种不同的度量。一方面,迁移输出应该带有目标风格, 可以由风格得分给出。另一方面,迁移应该保留与风格无关的内容, 可以由内容保留得分给出。此外,在至少一些实施例中,可以期望测 量可以由多样性得分给出的模型的不同迁移输出生成能力。

[0073] 在一些实施例中,分类器可用于评估风格迁移结果的保真度。 具体来说,Byte-mLSTM(或其他情感分类模型)可用于对文本风格 迁移模型生成的输出句子进行分类。当将否定句迁移为肯定句时,迁 移模型应该能够生成被分类器认为是肯定句的句子。如果是这种情 况,则结果可以表示为准确的迁移。然后,模型的总体风格迁移性能 可以由分类器测量的对测试集的平均精度给出。

[0074] 各个实施例还可以利用内容保存得分。可以建立与风格无关的 距离度量,其可以量化两个句子之间的内容相似度,例如在去掉其风 格词后比较句子的嵌入情况。具体来说,可以计算句子中每个非风格 词的嵌入,例如通过使用word2vec。接下来,计算嵌入的平均值, 其用作句子的嵌入。两个句子的内容相似度由它们的余弦距离给出。 根据语言风格在n-gram分布中大型编码的观察,可以计算出n-gram 的相对频率来确定哪个词是风格词。设D_1和D_2为不同风格的两个语料 库的n-gram频率。n-gram u在风格域i中的风格大小可以由下式给出:

[0075] $$s_i(u) = \frac{D_i(u) + \lambda}{\sum_{j\neq i} D_j(u) + \lambda}$$

[0076] 其中λ是小常数。我们在实验中使用1-gram。如果$\min_{k\in\{i,j\}} s_k(u)$大于 阈值,则该词被视为风格词。

[0077] 各个实施例还可以利用多样性得分。为了量化风格迁移输出的 多样性,可以计算诸如self-BLEU得分之类的得分。给定输入句子, 可以多次(例如5次)应用风格迁移模型以获得相应数量的输出(例 如5个输出)。然后可以计算任意两个生成句子(10对)之间的self-BLEU得分。这样的程序可以应用于测试集中的所有句子,并计 算出平均self-BLEU得分。之后,多样性得分可以定义为100-υ。具 有较高多样性得分的模型意味着该模型更能生

成不同的输出。

[0078] 在学习过程中，希望可以用不同的风格或语法来改写句子，但 还保存与风格无关的内容。在至少一些实施例中，可以在风格得分和 内容保存得分之间进行权衡。具体地说，示例模型可以在早期的训练 阶段迁移风格，但可能不会保存句子内容。经过多次迭代的训练后，这样的模型可以逐渐提高内容保存得分，但也可能降低风格得分。这 一结果表明，在风格迁移和内容保存之间保持平衡的性能可能是个挑 战。为了提高模型训练的稳定性，如果模型在验证集上达到均衡性能，则可以降低学习率。

[0079] 一些实施例可以利用能够导致更精确的风格迁移的采样方案。在推理过程中，示例网络可以将输入句子作为查询，并检索其内容信 息与该查询相似的目标风格句子池。这可以通过估计句子嵌入之间的 余弦相似度来实现。可以从检索到的池中(随机地或其他方式)对目 标风格代码进行采样，并生成输出句子。

[0080] 需要注意的是，不同的风格可包括同一风格的不同程度或变化。 例如，句子可以被生成，它们都有积极的情感，但表达方式不同。例 如，正派、善良、伟大都是积极的，但表达不同的情感。此外，句号 和感叹号指示特定情感的程度。在一些实施例中，这些可以被认为是 沿着风格或语法空间中的情感轴的值，其中风格代码可以表示n维风 格空间中的点。因此，模型可以将输入映射到风格空间中的任何点， 该点可以是相关属性的任意组合，也可以具有这些属性中每个属性的 相对值。

[0081] 应当注意，在所有实施例中，输入内容不必是文本或字母数字 输入的形式。例如，输入也可以以语音输入的形式提供，例如某人说 一个句子，该句子由麦克风捕获，然后对其内容和风格进行分析。在 这样的实施例中，输出可以是文本或音频，其中输出的类型也可以是 风格空间的维度。例如，输出可以是具有类似于特定人的语音模式的 计算机生成的语音，或者具有特定语音属性集的人。输出也可以是口 头的、大声的、以歌曲的形式或以另一种这样的风格。对于游戏或动 画应用，可以使用角色的语音来训练网络，使得演员可以读取角色语 音中的线(line)，并且相应的输出也将以角色的语音和预期的风格 生成。如果角色有特定的说话方式或句子的措辞方式，也可以通过一 个或更多个相关网络学习和推理。

[0082] 图5示出了根据各个实施例的用于执行可利用的语法迁移的示 例性过程500。应当理解，对于本文所讨论的该过程和其他过程，除 非另有说明，否则在各个实施例的范围内，可以以类似的或替代的顺 序或并行地执行附加的、替代的或更少的步骤。此外，该示例讨论了 使用文本数据训练卷积神经网络(CNN)，但如本文其他部分所讨论 的，在各个实施例的范围内，可以存在使用各种不同类型的数据训练 的各种类型的模型。在这个示例中，接收第一句子作为输入502。这 可以通过许多不同的方式中的任何一种从许多不同的来源接收，并且 可能需要进行预处理或语音到文本的翻译等选项。还可以选择至少一 个第二句子504。句子可以随机选择，也可以使用句子库中的选择函 数，或者可以从上面讨论的形式之一的来源之一接收等等。本示例中 的第二句子将具有不同的风格或表达类型，其可包括不同的情感、结 构，或本文讨论和建议的其他这样的方面。

[0083] 可以使至少一个神经网络确定第一句子和第二句子的代码 506。例如，第一卷积神经网络(CNN)可用于推理代表输入句子的 内容或概念的第一句子的内容代码。第二CNN可以用来推理每个选 定的第二句话的风格代码。如本文所讨论的，风格代码可以包括相应

第二句子的均值和偏差值。然后，该过程可以使得生成至少三分之一 的句子508，其包括第一句子的内容或概念，但使用不同的语法表达。 语法可以使用从第二句子之一中推理出的各自的风格代码来确定。该 过程可以生成多个第三句子，每个第三句子都有不同的语法来表达第 一句子的内容，其中每个语法部分取决于各自的风格代码。如前所述， 这些可以在风格(诗歌、歌词、句子)、情感(积极、消极、快乐、 愤怒)、结构(问题、陈述、格式)和其他方面有所不同。

[0084] 图6示出了根据各个实施例的用于使用可利用的至少一个经训 练的神经网络进行推理的另一示例过程600。在该示例中，接收要执 行文本迁移的输入句子602。如上文所述，句子可以通过任何方式从 多个地点和/或来源中的任何一个接收或获取。第一神经网络可用于 确定与输入句子的内容相对应的内容代码(或其他这样的表示)604。 如本文所讨论的，内容代码代表输入句子的概念或实质，与该概念在 输入句子中的表达方式不同。一个或更多个风格代码(或其他这样的 表示)也可以被确定为用于文本迁移606。这些可以基于风格选择， 根据随机采样或确定的采样函数选择，或者根据所确定的其他方式选 择。此外，在一些实施例中，可以通过选择表示的句子并用至少一个 神经网络处理这些句子来获得代码，以获得相关的风格代码。如前所 述，在至少一些实施例中，风格代码可以包括可用于迁移的均值和偏 差值。一旦获得，可以使用至少一个传递函数使用或组合来自输入句子的风格代码与所确定的风格代码中的至少一个来生成至少一个潜 在表示608。传递函数可以利用上述风格代码的均值和偏差值。然后， 可以使用至少一个经解码训练的神经网络对每个潜在表示进行解码， 以生成具有对应于所确定的风格代码之一的新语法的输出句子610。 在一些实施例中，可以从相同的输入句子同时推理出语法不同的多个 句子，每个句子包含单个输入句子或文本等的内容的不同表达。

[0085] 如上所述，越来越多的行业和应用正在利用机器学习。例如， 在处理器上开发的深度神经网络(DNN)已用于各种用例，从无人 驾驶汽车到更快的药物开发，从用于安全系统的自动图像分析到视频 聊天应用中的智能实时语言翻译。深度学习是一种建模人脑的神经学 习过程，不断学习，不断变得更聪明并随着时间的推移更快地提供更 准确的结果的技术。最初，成年人教孩子正确识别和分类各种形状， 最终无需任何指导即能够识别形状。类似地，将需要对被设计为完成 类似任务的深度学习或神经学习系统进行训练，以使其在识别基本对 象、被遮挡对象等方面变得更聪明、更高效，同时还为这些对象分配 上下文。

[0086] 在最简单的级别上，人脑中的神经元查看接收到的各种输入， 将重要性级别分配给这些输入中的每一个，然后将输出传递给其他神 经元以对其进行操作。人工神经元或感知器是神经网络的最基本模 型。在一个示例中，感知器可以接收表示该感知器正被训练以识别和 分类的对象的各种特征的一个或更多个输入，并且基于在定义对象的 形状中的特征的重要性为这些特征中的每一个分配某一权重。

[0087] 深度神经网络(DNN)模型包括许多连接的感知器(例如，节 点)的多个层，可以用大量的输入数据对其进行训练，从而以高精度 快速解决复杂的问题。在一个示例中，DNN模型的第一层将汽车的 输入图像分解为各个部分，并寻找诸如线条和角之类的基本图案。第二层组装线条以寻找更高级别的图案，例如车轮、挡风玻璃和后视镜。 下一层识别车辆的类型，最后几层为输入图像生成标签，识别特定汽 车品牌的模型。一旦训练了DNN，就可以

部署DNN并将其用于在称 为推理的过程中识别和分类对象或图案。推理的示例(DNN从给定输入中提取有用信息的过程)包括识别存入ATM机的支票上的手写 数字,识别照片中朋友的图像,提供电影推荐,在无人驾驶汽车中识 别和分类不同类型的汽车、行人以及道路危险,或近乎实时地翻译人 类语音。

[0088] 在训练过程中,数据在前向传播阶段流经DNN,直到产生表示 对应于输入的标签的预测为止。如果神经网络没有正确标记输入,则 将分析正确标签和预测标签之间的误差,并在反向传播阶段调整每个 特征的权重,直到DNN正确标记输入和训练数据集中的其他输入。 训练复杂的神经网络需要大量的并行计算性能,包括所支持的浮点乘 法和加法。推理比训练计算量要少,这是一个对延迟敏感的过程,其 中将经训练的神经网络应用于以前从未见过的新输入,以对图像进行 分类,翻译语音并通常推理出新信息。

[0089] 神经网络在很大程度上依赖于矩阵数学运算,而复杂的多层网 络需要大量的浮点性能和带宽以提高效率和速度。具有数千个处理核 心,针对矩阵数学运算进行了优化,并提供数十至数百TFLOPS的性 能,计算平台可以提供基于深度神经网络的人工智能和机器学习应用 所需的性能。

[0090] 图7示出了根据各个实施例的可用于训练和利用机器学习的示 例系统700的组件。如将要讨论的,可以由可以在单个实体或多个实 体的控制下的计算设备和资源或单个计算系统的各种组合来提供各 种组件。此外,可以由不同实体触发、发起或请求各个方面。例如, 在一些实施例中,可以由与供应商环境706相关联的供应商来指导对 神经网络的训练,而在其他实施例中,可以由能够通过客户端设备 702或其他此类资源访问供应商环境的顾客或其他用户来请求训练神 经网络。训练数据(或要由经训练的神经网络分析的数据)可以由供 应商、用户或第三方内容供应商724等提供。在一些实施例中,客户 端设备702可以是车辆或对象,可以代表用户进行导航,例如,该用 户可以提交请求和/或接收有助于设备导航的指令。

[0091] 在该示例中,能够通过至少一个网络704提交请求以将其接收 至供应商环境706。客户端设备可以是使用户能够生成和发送这样的 请求的任何适当的电子和/或计算设备,如可以包括台式计算机、笔 记本计算机、计算机服务器、智能手机、平板电脑、游戏机(便携式 或其他方式)、计算机处理器、计算逻辑和机顶盒等。一个或更多个 网络704可以包括用于发送请求或其他这样的数据的任何适当的网 络,例如可以包括因特网、内联网、以太网、蜂窝网络、局域网(LAN)、在节点之间进行直接无线连接的网络等等。

[0092] 可以将请求接收至接口层708,在该示例中,该接口层可以将 数据转发到训练和推理管理器710。在至少一些实施例中,该管理器 可以是包括用于管理与数据或内容相一致的服务和请求的硬件和软 件的系统或服务。该管理器可以接收训练神经网络的请求,并且可以 将请求的数据提供给训练管理器712。如果请求未指定,则训练管理 器712可以选择要使用的适当模型或网络,并可以使用相关的训练数 据来训练模型。在一些实施例中,训练数据可以是从客户端设备702 接收的或从第三方供应商724获得的等存储到训练数据存储库714的 一批数据。训练管理器712可以负责训练数据,例如通过使用本文讨 论的基于LARC的方法。网络可以是任何适当的网络,例如递归神 经网络(RNN)或卷积神经网络(CNN)等。一旦训练了网络并成 功评估了网络,就可以将经训练的网络存储到模型存储库716,例如, 其可以存储用于用户、应用程序或服务等的不同模型或网络。如上所 述,在一

些实施例中,针对单个应用程序或实体可能存在多个模型，如可以基于多个不同因素来利用多个模型。

[0093] 在随后的时间点,可以从客户端设备702(或另一个这样的设 备)接收对至少部分地受经训练的神经网络确定或影响的内容(例如，路径确定)或数据的请求。该请求可以包括例如要使用神经网络处理 的输入数据,以获得一个或更多个推理或其他输出值、分类或预测。尽管在各个实施例中也可以使用不同的系统或服务,但是输入数据可 以被接收至接口层708并且被引导到推理模块718。如果尚未本地存 储到推理模块718,则推理模块718可以从模型存储库716中获得适 当的训练网络,例如本文所述的训练深度神经网络(DNN)。推理 模块718可以将数据作为输入提供给训练网络,然后可以生成一个或 更多个推理作为输出。例如,这可以包括输入数据实例的分类。然后 可以将推理发送到客户端设备702以向用户显示或与用户进行其他 通信。用户的环境数据也可以存储到用户环境数据存储库722,该数 据库可以包括关于用户的数据,该数据可以用作生成推理或确定获得 实例后返回给用户的数据的网络输入等。包括输入或推理数据中的至 少一部分的相关数据也可以被存储到本地数据库720以用于处理将 来的请求。在一些实施例中,用户可以使用帐户或其他信息来访问供 应商环境的资源或功能。如果允许和可用,还可以收集用户数据并将其用于进一步训练模型,以便为将来的请求提供更准确的推理。在一 些实施例中,可以通过用户界面接收对在客户端设备702上执行的机 器学习应用726的请求,并通过相同的界面显示结果。客户端设备可 以包括资源,诸如用于生成请求和处理结果或响应的处理器728和内 存730,以及用于存储用于机器学习应用726的数据的至少一个数据 存储元件732。

[0094] 在各个实施例中,处理器728(或训练管理器712或推理模块 718的处理器)将是中央处理单元(CPU)。但是,如上所述,此类 环境中的资源可以利用GPU来处理用于至少某些类型的请求的数 据。GPU具有数千个核心,旨在处理大量的并行工作负载,因此在 用于训练神经网络和生成预测的深度学习中变得很流行。虽然使用 GPU进行离线构建可以更快地训练更大、更复杂的模型,但离线生 成预测意味着无法使用请求-时间输入特征,或者必须针对所有特征 排列生成预测并将其存储在查找表中以进行服务实时请求。如果深度 学习框架支持CPU模式,并且模型足够小且简单以至可以合理的延 迟在CPU上执行前馈,则CPU实例上的服务可以托管模型。在这种 情况下,可以在GPU上离线进行训练,并在CPU上实时进行推理。如果CPU方法不是可行的选项,则服务可以在GPU实例上运行。但 是,由于GPU具有与CPU不同的性能和成本特征,因此运行将运行 时算法卸载到GPU的服务可能要求将其设计成与基于CPU的服务不 同。

[0095] 图8示出了根据各个实施例的可用于对数据进行分类或生成推 理的示例系统800。根据本文所包含的教导和建议,应当显而易见的 是,也可以针对输入数据生成各种类型的预测、标签或其他输出。此 外,可以在本文讨论的各个实施例中使用有监督训练和无监督训练。 在该示例中,提供一组训练数据802(例如,分类或标记的数据)作 为输入,以用作训练数据。训练数据可以包括针对其要训练神经网络 的至少一种类型的对象的实例,以及识别该类型的对象的信息。例如， 训练数据可能包括一组图像,每个图像都包含对象类型的表示,其中 每个图像还包含标签、元数据、分类或其他识别在各自的图像中表示 的对象类型的信息或与之相关联。各种其他类型的数据也可以用作训 练数据,并且可以包括文

本数据、音频数据、视频数据等。在此示例 中，训练数据802作为训练输入提供给训练管理器804。训练管理器 804可以是包括硬件和软件的系统或服务，例如执行训练应用程序的一个或更多个计算设备，用于训练神经网络(或其他模型或算法等)。 在该示例中，训练管理器804接收指示要用于训练的模型的类型的指 令或请求。模型可以是可用于此类目的的任何适当的统计模型、网络 或算法，例如可以包括人工神经网络、深度学习算法、学习分类器、 贝叶斯网络等。训练管理器804可以从适当的存储库806中选择初始 模型或其他未经训练的模型，并利用训练数据802来训练模型生成可 以用于对相似类型的数据进行分类的经训练的模型808(例如，经训 练的深度神经网络)，或生成其他此类推理。在不使用训练数据的一 些实施例中，仍然可以根据训练管理器804选择适当的初始模型来对 输入数据进行训练。

[0096] 可以以多种不同方式训练模型，这可能部分取决于所选模型的 类型。例如，在一个实施例中，可以向机器学习算法提供一组训练数 据，其中模型是通过训练过程创建的模型伪像。训练数据的每个实例 包含正确答案(例如，分类)，其可以被称为目标或目标属性。学习 算法在训练数据中找到将输入数据属性映射到目标的模式，要预测的 答案，并输出捕获这些模式的机器学习模型。然后可以使用机器学习 模型来获得对未指定目标的新数据的预测。

[0097] 在一个示例中，训练管理器804可以从一组机器学习模型中进 行选择，所述机器学习模型包括二进制分类、多分类和回归模型。要 使用的模型的类型可以至少部分取决于要预测的目标的类型。用于二 进制分类问题的机器学习模型可预测二进制结果，例如两个可能的类 之一。学习算法(诸如逻辑回归)可用于训练二进制分类模型。针对 多类别分类问题的机器学习模型允许针对多个类别生成预测，例如预 测两个以上结果中的一个。多项式逻辑回归对于训练多类模型可能很 有用。回归问题的机器学习模型可预测数值。线性回归对于训练回归 模型很有用。

[0098] 为了训练根据一个实施例的机器学习模型，训练管理器必须确 定输入训练数据源以及其他信息，例如包含要预测的目标的数据属性 的名称、所需的数据转换指令，以及训练参数以控制学习算法。在训 练过程中，在一些实施例中训练管理器804可以基于训练数据源中指 定的目标类型自动选择适当的学习算法。机器学习算法可以接受用于 控制训练过程和所得机器学习模型的某些属性的参数。这些在本文中 称为训练参数。如果未指定任何训练参数，则训练管理器可以利用已 知的默认值来很好地处理大范围的机器学习任务。可以为其指定值的 训练参数的示例包括最大模型大小、通过训练数据的最多次数、随机 播放类型、正则化类型、学习率和正则化量。可以指定默认设置，具 有用于调整值以微调性能的选项。

[0099] 最大模型大小是在训练模型期间创建的模式的总大小(以字节 为单位)。默认情况下，可以创建指定大小的模型，例如100MB的 模型。如果训练管理器无法确定足够的图案来填充模型大小，则可以 创建较小的模型。如果训练管理器发现图案数量超出了指定大小所能 容纳的数量，则可以通过修整对学习模型的质量影响最小的图案来强 制实施最大截止。选择模型大小可以对模型的预测质量和使用成本之 间的权衡进行控制。较小的模型可能会导致训练管理器移除许多模式 以适应最大大小限制，从而影响预测的质量。另一方面，较大的模型 查询实时预测的成本可能更高。较大的输入数据集不一定会导致较大 的

模型,因为模型存储图案而不是输入数据。如果模式少且简单,则 生成的模型将很小。具有大量原始属性(输入列)或派生特征(数据 转换的输出)的输入数据可能会在训练过程中找到并存储更多图案。

[0100] 在一些实施例中,训练管理器804可以对训练数据进行多次通 过或迭代以尝试发现图案。可以存在默认的通过次数,例如十次,而 在一些实施例中,可以设置最大通过次数,例如高达一百次通过。在 一些实施例中,可能没有最大集合,或者可能存在会触发训练过程结 束的收敛准则或其他因素集合。在一些实施例中,训练管理器804可 以在训练期间监视图案的质量(即,模型收敛),并且当没有更多的 数据点或图案需要发现时可以自动停止训练。仅具有少量观察结果的 数据集可能需要更多的数据遍历才能获得足够高的模型质量。较大的 数据集可能包含许多相似的数据点,这可以减少对大量通过的需求。选择更多通过数据的潜在影响是,模型训练可能需要更长的时间,并 且在资源和系统利用率方面会花费更多。

[0101] 在一些实施例中,在训练之前或在训练的传递之间混洗 (shuffle)训练数据。在许多实施例中,混洗是一种随机或伪随机混 洗,用于生成真正的随机排序,尽管可能存在一些约束条件以确保不 对某些类型的数据进行分组,或者如果存在这种分组,则可以对混洗 的数据进行重新混洗等。混洗改变了将数据用于训练的序列或安排,以使训练算法不会遇到相似类型的数据的分组或连续太多观察的单 一类型数据。例如,可以训练模型来预测对象。上传之前,数据可能 会按对象类型排序。然后,算法可以按对象类型按字母序列处理数据,首先仅遇到特定对象类型的数据。模型将开始学习该类型对象的图 案。然后,模型将仅遇到用于第二种对象类型的数据,并将尝试调整 模型以适合该对象类型,这可能会使适合第一种对象类型的图案退 化。对象类型之间的这种突然切换可能会产生无法学习如何准确预测 对象类型的模型。在将训练数据集划分为训练子集和评估子集之前,在一些实施例中,可以执行混洗,从而对于两个阶段都利用相对均匀 的数据类型分布。在一些实施例中,训练管理器804可以使用例如伪 随机混洗技术来自动混洗数据。

[0102] 在一些实施例中,当创建机器学习模型时,训练管理器804可 以使用户能够指定设置或应用定制选项。例如,用户可以指定一个或 更多个评估设置,以指示用于评估机器学习模型的预测质量的输入数 据的要保留的一部分。用户可以指定策略,其指示哪些属性和属性转 换可用于模型训练。用户还可以指定控制训练过程和所得模型的某些 属性的各种训练参数。

[0103] 一旦训练管理者确定完成了模型的训练,例如通过使用本文讨 论的至少一个最终标准,就可以提供训练的模型808,以供分类器814 用于分类(或以其他方式生成推理)验证数据812。如图所示,这涉 及模型的训练模式和模型的推理模式之间的逻辑转换。然而,在许多 实施例中,经训练的模型808将首先被传递到评估器810,该评估器 可以包括在用于评估经训练的模型的质量(或其他方面)的至少一个 计算资源(例如,至少一个服务器的CPU或GPU)上执行的应用程 序、过程或服务。对模型进行评估,以确定该模型在预测新数据和未 来数据的目标时是否至少会提供性能的最低的可接受或阈值水平。如 果否,则训练管理器804可以继续训练该模型。由于将来的数据实例 通常将具有未知的目标值,因此可能希望在已知目标答案的数据上检 查机器学习的精度度量,并将该评估用作未来数据的预测精度的代 理。

[0104] 在一些实施例中，使用被提供用于训练的训练数据802的子集 来评估模型。可以使用如上所述的混洗和拆分方法来确定该子集。此 评估数据子集将标有目标，因此可以充当评估地面实况的资源。使用 用于训练的相同数据来评估机器学习模型的预测精度是没有用的，因 为对于记住训练数据而不是对其进行概括的模型可能会产生肯定的 评估。一旦训练完成，则使用经训练的模型808来处理评估数据子集，并且评估器810可以通过将地面实况数据与该模型的相应输出(或预 测/观察)进行比较来确定该模型的精度。在一些实施例中，评估器 810可以提供摘要或性能度量，其指示预测值和真实值的匹配程度。如果经训练的模型不满足至少最小性能标准或其他这样的精度阈值，则可以指示训练管理器804进行进一步的训练，或者在某些情况下，尝试训练新的或不同的模型等。如果经训练的模型808满足相关标 准，则可以提供经训练的模型以供分类器814使用。

[0105] 当创建和训练机器学习模型时，在至少一些实施例中，可期望 指定将导致能够进行最准确预测的模型的模型设置或训练参数。示例 参数包括要执行(前向和/或反向)的传递数、正则化、模型大小和 混洗类型。但是，如上所述，选择对评估数据产生最佳预测性能的模 型参数设置可能会导致模型过度拟合。当模型存储了训练和评估数据 源中出现的图案，但未能概括数据中的图案时，就会发生过度拟合。 当训练数据包括评估中使用的所有数据时，经常会发生过度拟合。过 度拟合的模型在评估期间可能会表现良好，但可能无法对新数据或其 他验证数据做出准确的预测。为了避免选择过度拟合的模型作为最佳 模型，训练管理器可以保留额外的数据以验证模型的性能。例如，训 练数据集可分为60％用于训练，40％用于评估或验证，其可分为两 个或更多阶段。在选择了最适合评估数据的模型参数，导致收敛于验 证数据的子集(例如该验证数据的一半)之后，可以使用其余的验证数据执行第二次验证，以确保该模型的性能。如果此模型符合验证数 据的期望，则此模型不会过度拟合数据。可选地，可以将测试集或保 留集用于测试参数。使用第二个验证或测试步骤有助于选择适当的模 型参数以防止过度拟合。但是，从训练过程中拿出更多数据进行验证 会使可用于训练的数据更少。对于较小的数据集，这可能会出现问题， 因为可能没有足够的数据可用于训练。在这种情况下的一种方法是执 行交叉验证，如本文其他地方所述。

[0106] 有许多度量或洞察可用于审查和评估给定模型的预测精度。一 个示例评估结果包含用于报告模型总体成功的预测精度度量，以及帮 助探索模型的精度超出预测精度度量的可视化。结果还可以提供查看 设置得分阈值(如二进制分类)的影响的能力，并可以生成有关用以 检查评估的有效性的标准的警报。度量和可视化的选择可以至少部分 取决于要评估的模型的类型。

[0107] 经过令人满意的训练和评估后，可以使用经训练的机器学习模 型来构建或支持机器学习应用程序。在一个实施例中，构建机器学习 应用程序是涉及一系列步骤的迭代过程。核心机器学习问题可以根据 观察到的内容以及模型要预测的答案来构建。然后可以收集、清理和 准备数据，以使其适合于通过机器学习模型训练算法使用的数据。可 以对这些数据进行可视化和分析，以进行完整性检查，以验证数据质 量和理解数据。这可能是原始数据(例如输入变量)和答案数据(例 如目标)没有以可用于训练高度预测模型的方式表示的情况。因此，可能希望从原始变量构建更具预测性的输入表示或特征。可以将结果 特征输入到学习算法中，以构建模型并根据从模型构建中保留的数据 评估模型的质量。然后可

以使用模型为新数据实例生成目标答案的预 测。

[0108] 在图8的示例性系统800中，在提供评估之后，将经训练的模 型810提供给分类器814或使其可用，该分类器能够使用经训练的模 型来处理验证数据。例如，这可能包括从用户或未分类的第三方接收 到的数据，例如正在查询有关这些图像中所表示内容的信息的查询图 像。验证数据可以由分类器使用经训练的模型进行处理，并且可以将 产生的结果816(即，分类或预测)发送回相应的来源，或者进行其 他处理或存储。在一些实施例中，并且在允许这种使用的情况下，可 以将这些目前分类的数据实例存储到训练数据存储库，可以由训练管 理器将其用于经训练的模型808的进一步训练。在一些实施例中，将 在新数据可用时对模型进行连续训练，但是在其他实施例中，将根据 诸如数据集的大小或模型复杂度等因素，定期对这些模型进行训练， 例如每天或每周一次。

[0109] 分类器814可以包括适当的硬件和软件，用于使用经训练的模 型来处理验证数据812。在某些情况下，分类器将包括一个或更多个 计算机服务器，每个服务器具有一个或更多个能够处理数据的图形处 理单元(GPU)。GPU的配置和设计可能使它们比CPU或其他此类组件更适合用于处理机器学习数据。在一些实施例中，可以将经训练 的模型加载到GPU内存中，并且将接收到的数据实例提供给GPU以 进行处理。GPU可以具有比CPU多得多的核心，并且GPU核心可 以更不复杂。因此，给定的GPU可能能够通过不同的硬件线程同时 处理数千个数据实例。GPU也可以配置为最大化浮点吞吐量，这可 以为大型数据集提供明显的额外处理优势。

[0110] 即使在使用GPU、加速器和其他此类硬件来加速诸如模型训练 或使用此类模型进行数据分类之类的任务时，此类任务仍可能需要大 量时间、资源分配和成本。例如，如果要使用800次通过来训练机器 学习模型，并且数据集包括要用于训练的1,000,000个数据实例，则 每次通过都需要处理所有百万个实例。架构的不同部分也可以由不同 类型的设备支持。例如，可以在逻辑上集中的位置处使用一组服务器 来执行训练，如可以作为服务提供的那样，而原始数据的分类可以由 这种服务或在客户端设备上以及其他这样的选项来执行。在各个实施 例中，这些设备也可以由同一实体或多个实体拥有、操作或控制。

[0111] 图9示出了根据各个实施例的可以被训练或以其他方式利用的 示例神经网络900。在该示例中，统计模型是人工神经网络(ANN)， 其包括多个节点层，包括输入层902、输出层906和中间节点的多个 层904，通常称为“隐藏”层，因为内部层和节点通常在常规神经网 络中不可见或不可访问。尽管仅出于解释目的示出了几个中间层，但 是应当理解，对可以利用的中间层的数量没有限制，并且对层的任何 限制通常将是使用模型处理所需的资源或时间的因素。如本文其他地 方所讨论的，除了其他这样的选项之外，还可以使用其他类型的模型、网络、算法或过程，其可以包括节点和层的其他数量或选择。验证数 据可以由网络的各层处理以生成一组推理或推理分数，然后可以将其 馈送到损失函数908。

[0112] 在该示例网络900中，给定层的所有节点互连到相邻层的所有 节点。如图所示，中间层的节点然后将分别连接到两个相邻层的节点。 在某些模型中，节点也称为神经元或连接的单元，节点之间的连接称 为边缘。每个节点都可以为接收到的输入执行一个功能，例如通过使 用指定的功能。节点和边缘可以在训练过程中获得不同的权重，并且 节点的各个层可以对接收到的输入执行特定类型的转换，在训练过程 中还可以学习或调整这些转换。学习可以是有监督的学习，也可以是 无监督的学习，这可能至少部分取决于训练数据

集中包含的信息类 型。可以利用各种类型的神经网络，例如可以包括卷积神经网络(CNN)，所述卷积神经网络(CNN)包括许多卷积层和一组池化 层，并已被证明对诸如图像识别的应用程序是有益的。由于要确定的 参数数量相对较少，因此CNN比其他网络也更易于训练。

[0113] 在一些实施例中，可以使用各种调整参数来训练这种复杂的机 器学习模型。选择参数、拟合模型和评估模型是模型调整过程的一部 分，通常称为超参数优化。在至少一些实施例中，这种调整可以包括 对基础模型或数据进行内省。在训练或生产设置中，稳健的工作流程 对于避免超参数的过度拟合非常重要，如本文其他地方所述。交叉验 证和向训练数据集添加高斯噪声是避免对任何一个数据集过度拟合 的有用技术。对于超参数优化，在一些实施例中，可能需要保持训练 集和验证集固定。在一些实施例中，可以在某些类别中调整超参数， 例如可以包括数据预处理(即，将单词转换为向量)，CNN架构定 义(例如，过滤器尺寸、过滤器数量)，随机梯度下降(SGD)参数 (例如学习率)，正则化(例如丢弃概率)以及其他此类选项。

[0114] 在示例预处理步骤中，可以将数据集的实例嵌入到特定大小的 较低维空间中。该空间的大小是要调整的参数。CNN的架构包含许 多可调参数。过滤器尺寸的参数可以表示信息的解释，该信息与将要 分析的实例的大小相对应。在计算语言学中，这称为n-gram大小。 示例CNN使用三种不同的过滤器尺寸，它们代表了可能不同的 n-gram大小。每个过滤器尺寸的过滤器数量可以对应于过滤器的深 度。每个过滤器都尝试学习与实例结构不同的内容，例如文本数据的 句子结构。在卷积层中，激活函数可以是整流线性单位，并且池类型 设置为最大池。然后可以将结果串联到一维向量中，最后一层完全连 接到二维输出上。这对应于可以应用优化功能的二进制分类。一种这 样的功能是梯度下降的均方根(RMS)传播方法的实现，其中示例超 参数可以包括学习率、批大小、最大梯度法线和历元。神经网络、正 则化可能是一个非常重要的考虑因素。如所述，在一些实施例中，输 入数据可以是相对稀疏的。在这种情况下，主要的超参数在倒数第二 层处可以被丢弃，这表示一定比例的节点在每个训练周期不会“触 发”。示例训练过程可以基于对先前配置的性能的反馈来建议不同的 超参数配置。可以使用建议的配置来训练该模型，可以在指定的验证 集上进行评估，并可以进行性能报告。可以重复此过程，例如权衡探 索(了解更多有关不同配置的信息)和开发(利用先前的知识以获得 更好的结果)。

[0115] 由于训练CNN可以并行化并且可以利用GPU支持的计算资 源，因此可以针对不同场景尝试多种优化策略。复杂的场景允许调整 模型架构和预处理以及随机梯度下降参数。这扩展了模型配置空间。 在基本情况下，仅调整预处理和随机梯度下降参数。与基本方案相比， 在复杂方案中可以有更多的配置参数。可以使用线性或指数步数执行 关节空间的调整，并通过模型的优化循环进行迭代。这样的调整过程 的成本可以大大低于诸如随机搜索和网格搜索之类的调整过程，而不 会造成任何明显的性能损失。

[0116] 一些实施例可以使用反向传播来计算用于确定神经网络的权 重的梯度。反向传播是微分的一种形式，如上所述，可以使用梯度下 降优化算法来调整应用于各种节点或神经元的权重。在一些实施例 中，可以使用相关损失函数的梯度来确定权重。反向传播可以利用损 失函数对统计模型生成的输出的导数。如上所述，各个节点可以具有 定义各个节点的输出的关联激活功能。可以适当地使用各种激活函 数，如可以包括径向基函数(RBF)

和Sigmoid函数,它们可以被各 种支持向量机(SVM)用于数据转换。节点的中间层的激活函数在 本文中被称为内部积核心。这些函数可以包括,例如,识别函数、阶 梯函数、Sigmoid函数、斜坡函数等等。激活函数也可以是线性的或 非线性的,以及其他此类选项。

[0117] 推理和训练逻辑

[0118] 图10示出了用于执行与一个或更多个实施例相关联的推理和/ 或训练操作的推理和/或训练逻辑1015。下面结合图10和/或图11提 供关于推理和/或训练逻辑1015的细节。

[0119] 在至少一个实施例中,推理和/或训练逻辑1015可以包括但不 限于数据存储1001,以存储对应于在一个或更多个实施例的方面中 被训练为和/或用于推理的神经网络的神经元或层的前向和/或输出权 重和/或输入/输出数据。在至少一个实施例中,数据存储1001存储在 使用一个或更多个实施例的方面训练和/或推理期间的输入/输出数据 和/或权重参数的前向传播期间结合一个或更多个实施例训练或使用 的神经网络的每个层的权重参数和/或输入/输出数据。在至少一个实 施例中,数据存储1001的任何部分都可以包括在其他片上或片外数 据存储内,包括处理器的L1、L2或L3高速缓存或系统内存。

[0120] 在至少一个实施例中,数据存储1001的任何部分可以在一个 或更多个处理器或其他硬件逻辑设备或电路的内部或外部。在至少一 个实施例中,数据存储1001可以是高速缓存内存、动态随机可寻址 内存(“DRAM”)、静态随机可寻址内存(“SRAM”)、非易失 性内存(例如闪存)或其他存储。在至少一个实施例中,对数据存储 1001是在处理器的内部还是外部的选择,例如,或者由DRAM、 SRAM、闪存或某种其他存储类型组成,可以取决于存储片上或片外 的可用存储空间,正在执行训练和/或推理功能的延迟要求,在神经 网络的推理和/或训练中使用的数据批量大小或这些因素的某种组 合。

[0121] 在至少一个实施例中,推理和/或训练逻辑1015可以包括但不 限于数据存储1005,以存储与在一个或更多个实施例的方面中被训 练为和/或用于推理的神经网络的神经元或层相对应的反向和/或输出 权重和/或输入/输出数据神经网络。在至少一个实施例中,在使用一 个或更多个实施例的方面训练和/或推理期间,数据存储1005存储在 输入/输出数据和/或权重参数的反向传播期间结合一个或更多个实施 例训练或使用的神经网络的每个层的权重参数和/或输入/输出数据。 在至少一个实施例中,数据存储1005的任何部分可以与其他片上或 片外数据存储一起包括,包括处理器的L1、L2或L3高速缓存或系统内存。在至少一个实施例中,数据存储1005的任何部分可以在一 个或更多个处理器或其他硬件逻辑设备或电路上的内部或外部。在至 少一个实施例中,数据存储1005可以是高速缓存内存、DRAM、 SRAM、非易失性内存(例如闪存)或其他存储。在至少一个实施例 中,数据存储1005是在处理器的内部还是外部的选择,例如,是由 DRAM、SRAM、闪存还是其他某种存储类型组成,取决于可用存储 是片上还是片外,正在执行的训练和/或推理功能的延迟要求,在神 经网络的推理和/或训练中使用的数据批量大小或这些因素的某种组 合。

[0122] 在至少一个实施例中,数据存储1001以及数据存储1005可以 是分开的存储结构。在至少一个实施例中,数据存储1001以及数据 存储1005可以是相同的存储结构。在至少一个实施例中,数据存储 1001以及数据存储1005可以是部分相同的存储结构和部分分离的存 储结构。在至少一个实施例中,数据存储1001和数据存储1005的任 何部分可以与其他片上或片外数据存储包括在一起,包括处理器的 L1、L2或L3高速缓存或系统内存。

[0123] 在至少一个实施例中，推理和/或训练逻辑1015可以包括但不 限于一个或更多个算术逻辑单元(“ALU”)1010，用于至少部分地 基于训练和/或推理代码或由其指示来执行逻辑和/或数学运算，其结 果可能会导致(例如，来自神经网络内部的层或神经元的输出值)存 储在激活存储1020中的激活，其是存储在数据存储1001和/或数据 存储1005中的输入/输出和/或权重参数数据的函数。在至少一个实施 例中，激活响应于执行指令或其他代码，由ALU 1010执行的线性代 数和/或基于矩阵的数学生成在激活存储1020中存储的激活，其中存 储在数据存储1005中和/或数据存储1001中的权重值用作具有其他 值的操作数，例如偏置值、梯度信息、动量值或其他参数或超参数，可以将任何或所有这些存储在数据存储1005和/或数据存储1001或 其他片上或片外存储中。在至少一个实施例中，一个或更多个处理器 或其他硬件逻辑设备或电路中包括一个或更多个ALU 1010，而在另 一实施例中，一个或更多个ALU 1010可以在处理器或其他硬件逻辑 设备或使用它们(例如协处理器)的电路外。在至少一个实施例中，可以将一个或更多个ALU 1010包括在处理器的执行单元之内，或者 以其他方式包括在由处理器的执行单元可访问的ALU组中，该处理 器的执行单元可以在同一处理器内或者分布在不同类型的不同处理 器之间(例如，中央处理单元、图形处理单元、固定功能单元等)。在至少一个实施例中，数据存储1001、数据存储1005以及激活存储 1020可以在同一处理器或其他硬件逻辑设备或电路上，而在另一实 施例中，它们可以在不同的处理器或其他硬件逻辑设备或电路或相同 和不同处理器或其他硬件逻辑设备或电路的某种组合中。在至少一个 实施例中，激活存储1020的任何部分可以与其他片上或片外数据存 储包括在一起，包括处理器的L1、L2或L3高速缓存或系统内存。此外，推理和/或训练代码可以与处理器或其他硬件逻辑或电路可访 问的其他代码一起存储，并可以使用处理器的提取、解码、调度、执 行、退出和/或其他逻辑电路来提取和/或处理。

[0124] 在至少一个实施例中，激活存储1020可以是高速缓存内存、DRAM、SRAM、非易失性内存(例如，闪存)或其他存储。在至少 一个实施例中，激活存储1020可以完全地或部分地在一个或更多个 处理器或其他逻辑电路内部或外部。在至少一个实施例中，可以取决 于片上或片外可用的存储，进行训练和/或推理功能的延迟要求，在 推理和/或训练神经网络中使用的数据的批量大小或这些因素的某种 组合，选择激活存储1020是处理器的内部还是外部，例如，或者包 含DRAM、SRAM、闪存或其他存储类型。在至少一个实施例中，图 10中所示的推理和/或训练逻辑1015可以与专用集成电路(“ASIC”) 结合使用，例如来自Google的Tensorflow®处理单元、来自Graphcore ™的推理处理单元(IPU)或来自Intel Corp的Nervana®(例如“Lake Crest”)处理器。在至少一个实施例中，图10所示的推理和/或训练逻辑1015可与中央处理单元(“CPU”)硬件，图形处理单元(“GPU”) 硬件或其他硬件(例如现场可编程门阵列(“FPGA”))结合使用。

[0125] 图11示出了根据至少一个各种实施例的推理和/或训练逻辑 1015。在至少一个实施例中，推理和/或训练逻辑1015可以包括但不 限于硬件逻辑，其中计算资源被专用或以其他方式唯一地连同对应于 神经网络内的一层或更多层神经元的权重值或其他信息一起使用。在 至少一个实施例中，图11中所示的推理和/或训练逻辑1015可以与 专用集成电路(ASIC)结合使用，例如来自Google的Tensorflow®处理单元，来自Graphcore™的推理处理

单元(IPU)或来自Intel Corp 的Nervana®(例如“Lake Crest”)处理器。在至少一个实施例中，图11中所示的推理和/或训练逻辑1015可以与中央处理单元(CPU) 硬件、图形处理单元(GPU)硬件或其他硬件(例如现场可编程门阵 列(FPGA))结合使用。在至少一个实施例中，推理和/或训练逻辑 1015包括但不限于数据存储1001以及数据存储1005，其可以用于存 储权重值和/或其他信息，包括偏置值、梯度信息、动量值和/或其他 参数或超参数信息。在图11中所示的至少一个实施例中，数据存储 1001以及数据存储1005中的每一个都分别与专用计算资源(例如计 算硬件1002和计算硬件1006)相关联。在至少一个实施例中，计算 硬件1002和计算硬件1006中的每一个包括一个或更多个ALU，这 些ALU仅分别对存储在数据存储1001和数据存储1005中的信息执 行数学函数(例如线性代数函数)，执行函数的结果被存储在激活存 储1020中。

[0126] 在至少一个实施例中，数据存储1001和1005以及相应的计算 硬件1002和1006中的每一个分别对应于神经网络的不同层，使得从 数据存储1001和计算硬件1002的一个“存储/计算对1001/1002”得 到的激活提供作为数据存储1005和计算硬件1006的下一个“存储/ 计算对1005/1006”的输入，以便反映神经网络的概念组织。在至少 一个实施例中，每个存储/计算对1001/1002和1005/1006可以对应于 一个以上的神经网络层。在至少一个实施例中，在推理和/或训练逻 辑1015中可以包括在存储计算对1001/1002和1005/1006之后或与之 并行的附加存储/计算对(未示出)。

[0127] 神经网络训练和部署

[0128] 图12示出了根据至少一个实施例的深度神经网络的训练和部 署。在至少一个实施例中，使用训练数据集1202来训练未经训练的 神经网络1206。在至少一个实施例中，训练框架1104是PyTorch框 架，而在其他实施例中，训练框架1104是Tensorflow，Boost，Caffe，Microsoft Cognitive Toolkit/CNTK，MXNet，Chainer，Keras，

[0129] Deeplearning4j或其他训练框架。在至少一个实施例中，训练框架1104 训练未经训练的神经网络1106，并使它能够使用本文所述的处理资 源来训练，以生成经训练的神经网络1108。在至少一个实施例中， 权重可以被随机选择或通过使用深度信念网络预训练。在至少一个实 施例中，可以以有监督、部分有监督或无监督的方式执行训练。

[0130] 在至少一个实施例中，使用有监督学习来训练未经训练的神经 网络1106，其中训练数据集1102包括与用于输入的期望输出配对的 输入，或者其中训练数据集1102包括具有已知输出的输入和神经网 络是手动分级的输出。在至少一个实施例中，以有监督的方式来训练 未经训练的神经网络1106，以处理来自训练数据集1102的输入，并 将结果输出与一组期望或想要的输出进行比较。在至少一个实施例 中，然后通过未经训练的神经网络1106将误差传播回去。在至少一 个实施例中，训练框架1104调整控制未经训练的神经网络1106的权 重。在至少一个实施例中，训练框架1104包括用于监视未经训练的 神经网络1106向模型(例如，经训练的神经网络1108)收敛的程度 的工具，适于基于已知输入数据(例如新数据1112)生成正确答案 (例如结果1114)的模型。在至少一个实施例中，训练框架1104反 复训练未经训练的神经网络1106，同时调整权重以使用损失函数和 调整算法(例如随机梯度下降)来改善未经训练的神经网络1106的 输出。在至少一个实施例中，训练框架1104训练未经训练的神经网 络1106，直到未经训练的神经网络1106达到期望的精度为止。在至 少一个实施例中，然后可以部署经训练的神经网络1108以实现任何 数量的机器

学习操作。

[0131] 在至少一个实施例中，使用无监督学习来训练未经训练的神经 网络1106，其中未经训练的神经网络1106尝试使用未标记的数据来 训练自己。在至少一个实施例中，无监督学习训练数据集1102将包 括输入数据，而没有任何关联的输出数据或“地面实况”数据。在至 少一个实施例中，未经训练的神经网络1106可以学习训练数据集 1102内的分组，并且可以确定各个输入如何与未经训练的数据集 1102相关。在至少一个实施例中，可以使用无监督训练来生成自组 织图，其是一种类型的经训练的神经网络1108，能够执行对减少新数据1112的维度有用的操作。在至少一个实施例中，无监督训练也 可以用于执行异常检测，这允许识别新数据集1112中偏离新数据集 1112的正常模式的数据点。

[0132] 在至少一个实施例中，可以使用半监督学习，这是一种技术， 其中在训练数据集1102中包括标记数据和未标记数据的混合。在至 少一个实施例中，训练框架1104可以用于例如通过转移的学习技术 来执行递增学习。在至少一个实施例中，递增学习使得经训练的神经 网络1108能够适应新数据1112，而不会忘记在初始训练期间注入到 网络内的知识。

[0133] 数据中心

[0134] 图13示出了示例数据中心1300，其中可以使用至少一个实施 例。在至少一个实施例中，数据中心1300包括数据中心基础设施层 1310、框架层1320、软件层1330和应用层1340。

[0135] 在至少一个实施例中，如图13所示，数据中心基础设施层1310 可以包括资源协调器1312、分组的计算资源1314和节点计算资源 (“节点C.R.”)1316(1)-1316(N)，其中“N”代表任何完整 的正整数。在至少一个实施例中，节点C.R.1316(1)-1316(N)可 以包括但不限于任何数量的中央处理单元(“CPU”)或其他处理器 (包括加速器、现场可编程门阵列(FPGA)、图形处理器等)，内 存设备(例如动态只读内存)，存储设备(例如固态硬盘或磁盘驱动 器)，网络输入/输出(“NW I/O”)设备，网络交换机，虚拟机(“VM”)， 电源模块和冷却模块等。在至少一个实施例中，节点C.R.1316(1) -1316(N)中的一个或更多个节点C.R.可以是具有一个或更多个上述 计算资源的服务器。

[0136] 在至少一个实施例中，分组的计算资源1314可以包括容纳在 一个或更多个机架内的节点C.R.的单独分组(未示出)，或者容纳在 各个地理位置的数据中心内的许多机架(也未示出)。分组的计算资 源1314内的节点C.R.的单独分组可以包括可以被配置或分配为支持 一个或更多个工作负载的分组的计算、网络、内存或存储资源。在至 少一个实施例中，可以将包括CPU或处理器的几个节点C.R.分组在 一个或更多个机架内，以提供计算资源来支持一个或更多个工作负 载。在至少一个实施例中，一个或更多个机架还可以包括任何数量的 电源模块、冷却模块和网络交换机，以任意组合。

[0137] 在至少一个实施例中，资源协调器1322可以配置或以其他方 式控制一个或更多个节点C.R.1316(1)-1316(N)和/或分组的计 算资源1314。在至少一个实施例中，资源协调器1322可以包括用于 数据中心1300的软件设计基础结构(“SDI”)管理实体。在至少一 个实施例中，资源协调器可以包括硬件、软件或其某种组合。

[0138] 在至少一个实施例中，如图13所示，框架层1320包括作业调 度器1332、配置管理器1334、资源管理器1336和分布式文件系统 1338。在至少一个实施例中，框架层1320可以包括支持软件层1330 的软件1332和/或应用程序层1340的一个或更多个应用程序1342的

框架。在至少一个实施例中，软件1332或应用程序1342可以分别包 括基于Web的服务软件或应用程序，例如由Amazon Web Services，Google Cloud和Microsoft Azure提供的服务或应用程序。在至少一个 实施例中，框架层1320可以是但不限于一种免费和开放源软件网络 应用框架，例如可以利用分布式文件系统1338来进行大范围数据处 理(例如“大数据”)的Apache SparkTM(以下称为“Spark”)。 在至少一个实施例中，作业调度器1332可以包括Spark驱动器，以 促进对数据中心1300的各个层所支持的工作负载进行调度。在至少 一个实施例中，配置管理器1334可以能够配置不同的层，例如软件 层1330和包括Spark和用于支持大规模数据处理的分布式文件系统 1338的框架层1320。在至少一个实施例中，资源管理器1336能够管 理映射到或分配用于支持分布式文件系统1338和作业调度器1332的 集群或分组计算资源。在至少一个实施例中，集群或分组计算资源可 以包括数据中心基础设施层1310上的分组的计算资源1314。在至少 一个实施例中，资源管理器1336可以与资源协调器1312协调以管理 这些映射的或分配的计算资源。

[0139] 在至少一个实施例中，包括在软件层1330中的软件1332可以 包括由节点C.R.1316(1)-1316(N)的至少一部分，分组计算资源 1314和/或框架层1320的分布式文件系统1338使用的软件。一种或 更多种类型的软件可以包括但不限于Internet网页搜索软件、电子邮 件病毒扫描软件、数据库软件和流视频内容软件。

[0140] 在至少一个实施例中，应用层1340中包括的一个或更多个应 用程序1342可以包括由节点C.R.1316(1)-1316(N)的至少一部 分、分组的计算资源1314和/或框架层1320的分布式文件系统1338 使用的一种或更多种类型的应用程序。一种或更多种类型的应用程序 可以包括但不限于任何数量的基因组学应用程序，认知计算和机器学 习应用程序，包括训练或推理软件，机器学习框架软件(例如PyTorch、TensorFlow、Caffe等)或其他与一个或更多个实施例结合使用的机 器学习应用程序。

[0141] 在至少一个实施例中，配置管理器1334、资源管理器1336和 资源协调器1312中的任何一个可以基于以任何技术上可行的方式获 取的任何数量和类型的数据来实现任何数量和类型的自我修改动作。在至少一个实施例中，自我修改动作可以减轻数据中心1300的数据 中心操作员做出可能不好的配置决定并且可以避免数据中心的未充 分利用和/或执行差的部分。

[0142] 在至少一个实施例中，数据中心1300可以包括工具、服务、软件或其他资源，以根据本文所述的一个或更多个实施例来训练一个 或更多个机器学习模型或者使用一个或更多个机器学习模型来预测 或推理信息。例如，在至少一个实施例中，可以通过使用上文关于数 据中心1300描述的软件和计算资源，根据神经网络架构通过计算权 重参数来训练机器学习模型。在至少一个实施例中，通过使用通过本 文所述的一种或更多种训练技术计算出的权重参数，可以使用上面与 关于数据中心1300所描述的资源，使用对应于一个或更多个神经网 络的经训练的机器学习模型来推理或预测信息。

[0143] 在至少一个实施例中，数据中心可以使用CPU、专用集成电 路(ASIC)、GPU、FPGA或其他硬件来使用上述资源来执行训练和 /或推理。此外，上述的一个或更多个软件和/或硬件资源可以配置成 一种服务，以允许用户训练或执行信息推理，例如图像识别、语音识 别或其他人工智能服务。

[0144] 推理和/或训练逻辑1015用于执行与一个或更多个实施例相关 联的推理和/或训

练操作。下面结合图10和/或图11提供关于推理和/ 或训练逻辑1015的细节。在至少一个实施例中，推理和/或训练逻辑 1015可以在系统图13中使用，至少部分地基于使用神经网络训练操 作、神经网络功能和/或架构或本文所述的神经网络用例计算出的权 重参数来推理或预测操作。

[0145] 根据至少一个实施例，数据中心基础设施1310可以接收输入 文本，并且为了如本文所述的训练和/或推理的目的，使该输入被定 向到应用层1340和软件层1330的相应组件。

[0146] 计算机系统

[0147] 图14是示出根据至少一个实施例示例性计算机系统的框图， 该示例性计算机系统可以是具有互连的设备和组件的系统，片上系统 (SOC)或它们的某种形成有处理器的组合1400，该处理器可以包 括执行单元以执行指令。在至少一个实施例中，根据本公开，例如本 文所述的实施例，计算机系统1400可以包括但不限于组件，例如处 理器1402，其执行单元包括逻辑以执行用于过程数据的算法。在至 少一个实施例中，计算机系统1400可以包括处理器，例如可从加利 福尼亚圣塔克拉拉的英特尔公司(Intel Corporation of Santa Clara，

[0148] California)获得的PENTIUM®处理器家族、XeonTM、Itanium®、 XScaleTM和/或StrongARMTM，Intel® Core™或Intel ®Nervana™微 处理器，尽管也可以使用其他系统(包括具有其他微处理器的PC、工程工作站、机顶盒等)。在至少一个实施例中，计算机系统1400 可以执行可从华盛顿州雷蒙德市的微软公司(Microsoft Corporation of Redmond,Wash.)获得的WINDOWS操作系统版本，尽管其他操作 系统(例如UNIX和Linux)、嵌入式软件和/或图形用户界面也可以 使用。

[0149] 实施例可以用在其他设备中，例如手持设备和嵌入式应用。手 持设备的一些示例包括蜂窝电话、互联网协议(Internet Protocol)设 备、数码相机、个人数字助理(“PDA”)和手持PC。在至少一个 实施例中，嵌入式应用可以包括微控制器、数字信号处理器(“DSP”)、片上系统、网络计算机(“NetPC”)、机顶盒、网络集线器、广域 网(“WAN”)交换机，或根据至少一个实施例可以执行一个或更 多个指令的任何其他系统。

[0150] 在至少一个实施例中，计算机系统1400可包括但不限于处理 器1402，该处理器1402可包括但不限于一个或更多个执行单元1408，以根据本文描述的技术执行机器学习模型训练和/或推理。在至少一 个实施例中，该系统是单处理器台式机或服务器系统，但是在另一实 施例中，该系统可以是多处理器系统。在至少一个实施例中，处理器 1402可以包括但不限于复杂指令集计算机(“CISC”)微处理器、精简指令集计算(“RISC”)微处理器、超长指令字(“VLIW”) 微处理器、实现指令集组合的处理器，或任何其他处理器设备，例如 数字信号处理器。在至少一个实施例中，处理器1402可以耦合到处 理器总线1410，该处理器总线1410可以在处理器1402与计算机系 统1400中的其他组件之间传输数据信号。

[0151] 在至少一个实施例中，处理器1402可以包括但不限于1级 (“L1”)内部高速缓存内存(“cache”)1404。在至少一个实施 例中，处理器1402可以具有单个内部高速缓存或多级内部缓存。在 至少一个实施例中，高速缓存内存可以驻留在处理器1402的外部。根据特定的实现和需求，其他实施例也可以包括内部和外部高速缓存 的组合。在至少一个实施例中，寄存器文件1406可以在各种寄存器 中存储不同类型的数据，包括但不限于整数寄存

器、浮点寄存器、状 态寄存器和指令指针寄存器。

[0152] 在至少一个实施例中，包括但不限于执行整数和浮点运算的逻 辑的执行单元1408，其也位于处理器1402中。处理器1402还可以 包括微码(“ucode”)只读内存(“ROM”)，用于存储某些宏指 令的微代码。在至少一个实施例中，执行单元1408可以包括用于处 理封装指令集1409的逻辑。在至少一个实施例中，通过将封装指令 集1409包括在通用处理器1402的指令集中，以及要执行指令的相关 电路，可以使用通用处理器1402中的封装数据来执行许多多媒体应 用程序使用的操作。在一个或更多个实施例中，可以通过使用处理器的数据总线的全宽度来在封装的数据上执行操作来加速和更有效地 执行许多多媒体应用程序，这可能不需要在处理器的数据总线上传输 较小的数据单元来一次执行一个数据元素的一个或更多个操作。

[0153] 在至少一个实施例中，执行单元1408也可以用在微控制器、嵌入式处理器、图形设备、DSP和其他类型的逻辑电路中。在至少一 个实施例中，计算机系统1400可以包括但不限于内存1420。在至少 一个实施例中，内存1420可以被实现为动态随机存取内存(“DRAM”)设备、静态随机存取内存(“SRAM”)设备、闪存 设备或其他存储设备。内存1420可以存储由处理器1402可以执行的 由数据信号表示的指令1419和/或数据1421。

[0154] 在至少一个实施例中，系统逻辑芯片可以耦合到处理器总线 1410和内存1420。在至少一个实施例中，系统逻辑芯片可以包括但 不限于内存控制器集线器(“MCH”)1416，并且处理器1402可以 经由处理器总线1410与MCH 1416通信。在至少一个实施例中，MCH 1416可以提供到内存1420的高带宽内存路径1418以用于指令和数 据存储以及用于图形命令、数据和纹理的存储。在至少一个实施例中， MCH 1416可以在处理器1402、内存1420和计算机系统1400中的其 他组件之间启动数据信号，并且在处理器总线1410、内存1420和系 统I/O 1422之间桥接数据信号。在至少一个实施例中，系统逻辑芯片 可以提供用于耦合到图形控制器的图形端口。在至少一个实施例中， MCH 1416可以通过高带宽内存路径1418耦合到内存1420，并且图 形/视频卡1412可以通过加速图形端口(Accelerated Graphics Port) (“AGP”)互连1414耦合到MCH 1416。

[0155] 在至少一个实施例中，计算机系统1400可以使用系统I/O 1422 作为专有集线器接口总线来将MCH 1416耦合到I/O控制器集线器 (“ICH”)1430。在至少一个实施例中，ICH 1430可以通过本地I/O 总线提供与某些I/O设备的直接连接。在至少一个实施例中，本地I/O 总线可以包括但不限于用于将外围设备连接到内存1420、芯片组和 处理器1402的高速I/O总线。示例可以包括但不限于音频控制器 1429、固件集线器(“Flash BIOS”)1428、无线收发器1426、数据 存储1424、包含用户输入的传统I/O控制器1423和键盘接口、串行 扩展端口1427(例如通用串行总线(USB))和网络控制器1434。数据存储1424可以包括硬盘驱动器、软盘驱动器、CD-ROM设备、闪存设备或其他大容量存储设备。

[0156] 在至少一个实施例中，图14示出了包括互连的硬件设备或“芯 片”的系统，而在其他实施例中，图14可以示出示例性片上系统 (“SoC”)。在至少一个实施例中，图14中示出的设备可以与专 有互连、标准化互连(例如，PCIe)或其某种组合互连。在至少一个 实施例中，系统1400的一个或更多个组件使用计算快速链路(CXL) 互连来互连。

[0157] 推理和/或训练逻辑1015用于执行与一个或更多个实施例相关 的推理和/或训练操作。下面结合图10和/或图11提供关于推理和/或 训练逻辑1015的细节。在至少一个实施

例中,推理和/或训练逻辑1015 可以在图14的系统中使用,用于至少部分地基于使用神经网络训练 操作、神经网络功能和/或架构或本文所述的神经网络用例计算的权 重参数来推理或预测操作。

[0158] 在一些实施例中,例如,可以通过扩展端口1427或无线收发 器1426接收视频数据流,然后将其定向到处理器1402和/或视频图 形卡1412以进行处理。根据这些组件是设备(诸如自主车辆)的一 部分还是单独的设备,然后输出可以经由I/O到达控制系统或经由无线收发器传输到车辆,以及其他这样的选择。

[0159] 图15是示出根据至少一个实施例的用于利用处理器1510的电 子设备1500的框图。在至少一个实施例中,电子设备1500可以是, 例如但不限于,笔记本电脑、塔式服务器、机架服务器、刀片服务器、 膝上型计算机、台式机、平板电脑、移动设备、电话、嵌入式计算机 或任何其他合适的电子设备。

[0160] 在至少一个实施例中,系统1500可以包括但不限于通信地耦 合到任何合适数量或种类的组件、外围设备、模块或设备的处理器 1510。在至少一个实施例中,处理器1510使用总线或接口耦合,诸 如1℃总线、系统管理总线(“SMBus”)、低引脚数(LPC)总线、串行外围接口(“SPI”)、高清音频(“HDA”)总线、串行高级 技术附件(“SATA”)总线、通用串行总线(“USB”)(1、2、3 版)或通用异步接收器/发送器(“UART”)总线。在至少一个实施 例中,图15示出了系统,该系统包括互连的硬件设备或“芯片”, 而在其他实施例中,图15可以示出示例性片上系统(“SoC”)。 在至少一个实施例中,图15中所示的设备可以与专有互连线、标准化互连(例如,PCIe)或其某种组合互连。在至少一个实施例中,图 15的一个或更多个组件使用计算快速链路(CXL)互连线来互连。

[0161] 在至少一个实施例中,图15可以包括显示器1524、触摸屏 1525、触摸板1530、近场通信单元(“NFC”)1545、传感器集线 器1540、热传感器1546、快速芯片组(“EC”)1535、可信平台模 块(“TPM”)1538、BIOS/固件/闪存(“BIOS,FW Flash”)1522、 DSP1560、驱动器(“SSD”或“HDD”)1520(例如固态磁盘(“SSD”) 或硬盘驱动器(“HDD”))、无线局域网单元(“WLAN”)1550、 蓝牙单元1552、无线广域网单元(“WWAN”)1556、全球定位系 统(GPS)1555、相机(“USB 3.0相机”)1554(例如USB 3.0相 机)或以例如LPDDR3标准实现的低功耗双倍数据速率(“LPDDR”) 内存单元(“LPDDR3”)1515。这些组件可以各自以任何合适的方 式实现。

[0162] 在至少一个实施例中,其他组件可以通过以上讨论的组件通信 地耦合到处理器1510。在至少一个实施例中,加速度计1541、环境 光传感器(“ALS”)1542、罗盘1543和陀螺仪1544可以可通信地 耦合到传感器集线器1540。在至少一个实施例中,热传感器1539、风扇1537、键盘1546和触摸板1530可以通信地耦合到EC1535。在 至少一个实施例中,扬声器1563、耳机1564和麦克风(“mic”) 1565可以通信地耦合到音频单元(“音频编解码器和D类放大器”) 1564,其又可以通信地耦合到DSP1560。在至少一个实施例中,音频 单元1564可以包括例如但不限于音频编码器/解码器(“编解码器”) 和D类放大器。在至少一个实施例中,SIM卡(“SIM”)1557可 以通信地耦合到WWAN单元1556。在至少一个实施例中,组件(诸如WLAN单元1550和蓝牙单元1552以及WWAN单元1556)可以 被实现为下一代形式因素(NGFF)。

[0163] 推理和/或训练逻辑1015用于执行与一个或更多个实施例相关 联的推理和/或训练操作。下面结合图10和/或图11提供关于推理和/ 或训练逻辑1015的细节。在至少一个实

施例中,推理和/或训练逻辑 1015可以在系统图15中使用,用于至少部分地基于使用神经网络训 练操作、神经网络功能和/或架构或本文所述的神经网络用例计算的 权重参数来推理或预测操作。

[0164] 图16示出了根据至少一个实施例的计算机系统1600。在至少 一个实施例中,计算机系统1600配置为实现贯穿本公开描述的各种 过程和方法。

[0165] 在至少一个实施例中,计算机系统1600包括但不限于至少一 个中央处理单元("CPU")1602,该中央处理单元("CPU")1602 连接到使用任何合适协议实现的通信总线1610,诸如PCI("外围设 备互联")、外围组件互连Express("PCI-Express")、AGP("加 速图形端口")、超传输或任何其他总线或点对点通信协议。在至少 一个实施例中,计算机系统1600包括但不限于主内存1604和控制逻 辑(例如,实现为硬件、软件或其组合),并且数据可以采取随机存 取内存("RAM")的形式存储在主内存1604中。在至少一个实施 例中,网络接口子系统("网络接口")1622提供到其他计算设备 和网络的接口,用于从计算机系统1600接收数据并将数据传输到其 他系统。

[0166] 在至少一个实施例中,计算机系统1600在至少一个实施例中 包括但不限于输入设备1608、并行处理系统1612和显示设备1606, 它们可以使用常规的阴极视线管("CRT")、液晶显示器("LCD")、发光二极管("LED")、等离子显示器或其他合适的显示技术实现。在至少一个实施例中,从输入设备1608(诸如键盘、鼠标、触摸板、麦克风等)接收用户输入。在至少一个实施例中,前述模块中的每一 个可以位于单个半导体平台上以形成处理系统。

[0167] 推理和/或训练逻辑1015用于执行与一个或更多个实施例相关 联的推理和/或训练操作。下面结合图10和/或图11提供关于推理和/ 或训练逻辑1015的细节。在至少一个实施例中,推理和/或训练逻辑 1015可以在系统图16中使用,以至少部分地基于使用神经网络训练 操作、神经网络功能和/或架构或本文所述的神经网络用例计算出的 权重参数来进行推理或预测操作。

[0168] 图17示出了根据至少一个实施例的计算机系统1700。在至少 一个实施例中,计算机系统1700包括但不限于计算机1710和USB 盘1720。在至少一个实施例中,计算机1710可以包括但不限于任何 数量和类型的处理器(未示出)和内存(未示出)。在至少一个实施 例中,计算机1710包括但不限于服务器、云实例、膝上型计算机和 台式计算机。

[0169] 在至少一个实施例中,USB盘1720包括但不限于处理单元 1730、USB接口1740和USB接口逻辑1750。在至少一个实施例中, 处理单元1730可以是任何指令执行系统、装置或能够执行指令的设 备。在至少一个实施例中,处理单元1730可以包括但不限于任何数 量和类型的处理核心(未示出)。在至少一个实施例中,处理核心 1730包括专用集成电路("ASIC"),该专用集成电路被优化为执 行与机器学习相关联的任何数量和类型的操作。例如,在至少一个实 施例中,处理核心1730是张量处理单元("TPC"),其被优化以 执行机器学习推理操作。在至少一个实施例中,处理核心1730是视 觉处理单元("VPU"),其被优化以执行机器视觉和机器学习推理 操作。

[0170] 在至少一个实施例中,USB接口1740可以是任何类型的USB 连接器或USB插座。例如,在至少一个实施例中,USB接口1740 是用于数据和电源的USB 3.0 Type-C插座。在至少一个实施例中, USB接口1740是USB 3.0 Type-A连接器。在至少一个实施例中,USB 接口逻辑1750可以包括使处理单元1730能够经由USB连接器1740 与设备(例如计算机1710)相连

接的任何数量和类型的逻辑。

[0171] 推理和/或训练逻辑1015用于执行与一个或更多个实施例相关 联的推理和/或训练操作。下面结合图10和/或图11提供关于推理和/ 或训练逻辑1015的细节。在至少一个实施例中,推理和/或训练逻辑 1015可以在系统图17中使用,至少部分地基于使用神经网络训练操 作、神经网络功能和/或架构或本文所述的神经网络用例计算出的权 重参数来推理或预测操作。

[0172] 图18是示出根据至少一个实施例的可使用一个或更多个IP核 心制造的芯片集成电路1800上的示例性系统的框图。在至少一个实 施例中,集成电路1800包括一个或更多个应用处理器1805(例如, CPU)、至少一个图形处理器1810,并且可以另外包括图像处理器1815和/或视频处理器1820,其中任意一个可能是模块化IP核心。 在至少一个实施例中,集成电路1800包括外围或总线逻辑,其包括 USB控制器1825、UART控制器1830、SPI/SDIO控制器1835和 I.sup.2S/I.sup.2C控制器1840。在至少一个实施例中,集成电路1800 可以包括显示设备1845耦合到高清多媒体接口(HDMI)控制器1850 和移动工业处理器接口(MIPI)显示接口1855中的一个或更多个。 在至少一个实施例中,存储可以由闪存子系统1860提供,包括闪存 和闪存控制器。在至少一个实施例中,可以经由内存控制器1865提 供内存接口以用于访问SDRAM或SRAM内存设备。在至少一个实 施例中,一些集成电路还包括嵌入式安全引擎1870。

[0173] 推理和/或训练逻辑1015用于执行与一个或更多个实施例相关 联的推理和/或训练操作。下面结合图10和/或图11提供关于推理和/ 或训练逻辑1015的细节。在至少一个实施例中,推理和/或训练逻辑 1015可以在集成电路1800中用于至少部分地基于使用神经网络训练 操作、神经网络功能和/或架构或本文描述的神经网络用例计算的权 重参数来推理或预测操作。

[0174] 例如,推理和/或训练逻辑1015可以接受输入视频流,并为视 频流中表示的对象生成推理,如本文所述。在至少一些实施例中,图 像处理器1815可以用于在接收到视频帧时对其进行处理。

[0175] 图19A-19B示出了根据本文所述的各个实施例的示例性集成 电路和相关联的图形处理器,其可以使用一个或更多个IP核心来制 造。除了图示之外,在至少一个实施例中可以包括其他逻辑和电路, 包括附加的图形处理器/核心、外围接口控制器或通用处理器核心。

[0176] 图19A-19B是示出根据本文描述的实施例的在SoC内使用的 示例性图形处理器的框图。图19A示出了根据至少一个实施例的芯 片集成电路上系统的示例性图形处理器1910,其可以使用一个或更 多个IP核心来制造。图19B示出了根据至少一个实施例的芯片集成 电路上系统的另外示例性图形处理器1940,其可以使用一个或更多 个IP核心来制造。在至少一个实施例中,图19A的图形处理器1910 是低功耗图形处理器核心。在至少一个实施例中,图19B的图形处 理器1940是更高性能的图形处理器核心。在至少一个实施例中,每 个图形处理器1910、1940可以是图18的图形处理器1810的变体。

[0177] 在至少一个实施例中,图形处理器1910包括顶点处理器1905 和一个或更多个片段处理器1915A-1915N(例如1915A、1915B、 1915C、1915D至1915N-1和1915N)。在至少一个实施例中,图形 处理器1910可以经由单独的逻辑来执行不同的着色器程序,使得顶 点处

理器1905被优化以执行针对顶点着色器程序的操作,而一个或 更多个片段处理器1915A-1915N执行片段(例如,像素)着色操作 用于片段或像素或着色器程序。在至少一个实施例中,顶点处理器 1905执行3D图形管线的顶点处理阶段并生成图元和顶点数据。在至 少一个实施例中,一个或更多个片段处理器1915A-1915N使用由顶 点处理器1905生成的图元和顶点数据来生成在显示设备上显示的帧 缓冲区。在至少一个实施例中,一个或更多个片段处理器 1915A-1915N被优化以执行如在OpenGL API中所提供的片段着色器 程序,其可以用于执行与在Direct 3D API中所提供的像素着色器程 序类似的操作。

[0178] 在至少一个实施例中,图形处理器1910附加地包括一个或更 多个内存管理单元(MMU)1920A-1920B、一个或更多个高速缓存 1925A-1925B和一个或更多个电路互连1930A-1930B。在至少一个实 施例中,一个或更多个MMU 1920A-1920B提供用于图形处理器1910的虚拟到物理地址的映射,包括用于顶点处理器1905和/或片段处理 器1915A-1915N,其可以引用存储在内存中的顶点或图像/纹理数据, 除了存储在一个或更多个高速缓存1925A-1925B中的顶点或图像/纹 理数据之外。在至少一个实施例中,一个或更多个MMU 1920A-1920B 可以与系统内的其他MMU同步,包括与图18的一个或更多个应用 处理器1805、图像处理器1815和/或视频处理器1820相关联的一个 或更多个MMU,使得每个处理器1805-1820可以参与共享或统一的 虚拟内存系统。在至少一个实施例中,一个或更多个电路互连1930A-1930B使图形处理器1910能够经由SoC的内部总线或经由直 接连接与SoC内的其他IP核心相连接。

[0179] 在至少一个实施例中,图形处理器1940包括图19A的图形处 理器1910的一个或更多个MMU 1920A-1920B、高速缓存 1925A-1925B和电路互连1930A-1930B。在至少一个实施例中,图形 处理器1940包括一个或更多个着色器核心1955A-1955N(例如, 1955A、1955B、1955C、1955D、1955E、1955F到1955N-1和1955N), 其提供了统一的着色器核心架构,其中单个核心或类型或核心可以执 行所有类型的可编程着色器代码,包括用于实现顶点着色器、片段着 色器和/或计算着色器的着色器程序代码。在至少一个实施例中,多 个着色器核心可以变化。在至少一个实施例中,图形处理器1940包 括核心间任务管理器1945,其充当线程分派器以将执行线程分派给 一个或更多个着色器核心1955A-1955N和分块单元1958,以加速基 于图块渲染的分块操作,其中在图像空间中细分了场景的渲染操作, 例如,以利用场景内的局部空间相干或优化内部缓存的使用。

[0180] 推理和/或训练逻辑1015用于执行与一个或更多个实施例相关 联的推理和/或训练操作。下面结合图10和/或图11提供关于推理和/ 或训练逻辑1015的细节。在至少一个实施例中,推理和/或训练逻辑 1015可以在集成电路图19A和/或图19B中用于至少部分地基于使用 神经网络训练操作、神经网络功能或架构,或本文所述的神经网络用 例计算的权重参数来进行推理或预测操作。例如,推理和/或训练逻 辑1015可以接受输入视频流,并为视频流中表示的对象生成推理, 如本文讨论的。

[0181] 图20A-20B示出了根据本文描述的实施例的附加的示例性图 形处理器逻辑。在至少一个实施例中,图20A示出了可以包括在图 18的图形处理器1810内的图形核心2000,在至少一个实施例中,可 以是图19B中统一的着色器核心1955A-1955N。图20B示出了在至 少一个实施例中的适用于在多芯片模块上部署的高度并行的通用图 形处理单元2030。

[0182] 在一个实施例中,图形核心2000包括共享指令高速缓存2002、纹理单元2018和高

速缓存/共享内存2020,它们是图形核心2000内 的执行资源所共有的。在至少一个实施例中,图形核心2000可以包 括多个切片(slice)2001A-2001N或每个核心的分区,图形处理器可 以包括图形核心2000的多个实例。切片2001A-2001N可以包括支持 逻辑,该支持逻辑包括本地指令高速缓存2004A-2004N、线程调度器 2006A-2006N、线程分派器2008A-2008N和一组寄存器 2010A-2010N。在至少一个实施例中,切片2001A-2001N可以包括一 组附加功能单元(AFU 2012A-2012N)、浮点单元(FPU 2014A-2014N)、整数算术逻辑单元(ALU 2016A-2016N)、地址计 算单元(ACU 2013A-2013N)、双精度浮点单元(DPFPU 2015A-2015N)和矩阵处理单元(MPU 2017A-2017N)。

[0183] 在一个实施例中,FPU 2014A-2014N可以执行单精度(32位) 和半精度(16位)浮点运算,而DPFPU 2015A-2015N可以执行双精 度(64位)浮点运算点操作。在一个实施例中,ALU 2016A-2016N 可以以8位、16位和32位精度执行可变精度整数运算,并且可以被 配置用于混合精度运算。在一个实施例中,MPU 2017A-2017N还可 被配置用于混合精度矩阵运算,包括半精度浮点运算和8位整数运 算。在一个实施例中,MPU 2017A-2017N可以执行各种矩阵操作以 加速机器学习应用框架,包括使得能够支持加速的通用矩阵到矩阵乘 法(GEMM)。在一个实施例中,AFU 2012A-2012N可以执行浮点 数或整数单元不支持的附加逻辑运算,包括三角运算(例如,Sine、Cosine等)。

[0184] 推理和/或训练逻辑1015用于执行与一个或更多个实施例相关 联的推理和/或训练操作。下面结合图10和/或图11提供关于推理和/ 或训练逻辑1015的细节。在至少一个实施例中,推理和/或训练逻辑 1015可以在图形核心2000中用于至少部分地基于使用神经网络训练 操作、神经网络功能和/或架构或本文描述的神经网络用例计算的权 重参数来推理或预测操作。

[0185] 图20B示出了在至少一个实施例中的通用处理单元(GPGPU) 2030,其可以被配置为使得高度并行的计算操作能够由图形处理单元 阵列来执行。在至少一个实施例中,GPGPU 2030可以直接链路到 GPGPU 2030的其他实例,以创建多GPU集群以提高用于深度神经 网络的训练速度。在至少一个实施例中,GPGPU 2030包括主机接口 2032以实现与主机处理器的连接。在至少一个实施例中,主机接口 2032是PCI Express接口。在至少一个实施例中,主机接口2032可 以是厂商专用的通信接口或通信结构。在至少一个实施例中,GPGPU 2030从主机处理器接收命令,并使用全局调度器2034将与那些命令 相关联的执行线程分派给一组计算集群2036A-2036H。在至少一个实 施例中,计算集群2036A-2036H共享高速缓存内存2038。在至少一 个实施例中,高速缓存内存2038可以用作计算集群2036A-2036H内的高速缓存内存的高级高速缓存。

[0186] 在至少一个实施例中,GPGPU 2030包括经由一组内存控制器 2042A-2042B与计算集群2036A-2036H耦合的内存2044A-2044B。 在至少一个实施例中,内存2044A-2044B可以包括各种类型的内存 设备,包括动态随机存取内存(DRAM)或图形随机存取内存,例如 同步图形随机存取内存(SGRAM),包括图形双倍数据速率(GDDR) 内存。

[0187] 在至少一个实施例中,计算集群2036A-2036H各自包括一组 图形核心,诸如图20A的图形核心2000,其可以包括多种类型的整 数和浮点逻辑单元,其可以以包括用于在适用于机器学习计算的精度 范围内执行计算操作。例如,在至少一个实施例中,每个计算集群2036A-2036H中的浮点单元的至少一个子集可以配置为执行16位或 32位浮点运算,而不同

的浮点单元的子集可以配置为执行64位浮点 运算。

[0188] 在至少一个实施例中,GPGPU 2030的多个实例可以被配置为 操作为计算集群。在至少一个实施例中,计算集群2036A-2036H用 于同步和数据交换的通信在实施例之间变化。在至少一个实施例中, GPGPU 2030的多个实例通过主机接口2032进行通信。在至少一个 实施例中,GPGPU 2030包括I/O集线器2039,其将GPGPU 2030与 GPU链路2040耦合,使得能够直接连接至GPGPU 2030的其他的实 例。在至少一个实施例中,GPU链路2040耦合到专用GPU到GPU 桥接器,其使得能够在GPGPU 2030的多个实例之间进行通信和同 步。在至少一个实施例中,GPU链路2040与高速互连耦合,以向其 他GPGPU或并行处理器发送和接收数据。在至少一个实施例中, GPGPU 2030的多个实例位于单独的数据处理系统中,并经由可经由 主机接口2032访问的网络设备进行通信。在至少一个实施例中,GPU 链路2040可被配置为能够连接到主机处理器,附加或替代主机接口 2032。

[0189] 在至少一个实施例中,GPGPU 2030可以配置为训练神经网络。 在至少一个实施例中,可以在推理平台内使用GPGPU 2030。在其中 使用GPGPU 2030进行推理的至少一个实施例中,相对于当使用 GPGPU训练神经网络时,GPGPU可以包括更少的计算集群 2036A-2036H。在至少一个实施例中,与内存2044A-2044B相关联的 内存技术可以在推理和训练配置之间有所不同,其中更高带宽的内存 技术专用于训练配置。在至少一个实施例中,GPGPU 2030的推理配 置可以支持推理特定指令。例如,在至少一个实施例中,推理配置可 以提供对一个或更多个8位整数点积指令的支持,该指令可以在部署 的神经网络的推理操作期间使用。

[0190] 推理和/或训练逻辑1015用于执行与一个或更多个实施例相关 联的推理和/或训练操作。下面结合图10和/或图11提供关于推理和/ 或训练逻辑1015的细节。在至少一个实施例中,推理和/或训练逻辑 1015可以在GPGPU 2030中用于至少部分地基于本文描述的使用神 经网络训练操作、神经网络功能和/或架构或神经网络用例计算的权 重参数来推理或预测操作。

[0191] 图21是示出根据至少一个实施例的计算系统2100的框图。在 至少一个实施例中,计算系统2100包括具有一个或更多个处理器 2102的处理子系统2101和经由可以包括内存集线器2105的互连路 径通信的系统内存2104。在至少一个实施例中,内存集线器2105可以是芯片组组件内的单独组件,也可以集成在一个或更多个处理器 2102内。在至少一个实施例中,内存集线器2105通过通信链路2106 与I/O子系统2111耦合。在一实施例中,I/O子系统2111包括I/O集 线器2107,其可以使计算系统2100能够接收来自一个或更多个输入设备2108的输入。在至少一个实施例中,I/O集线器2107可以使能 显示控制器,其包括在一个或更多个处理器2102中,用于向一个或 更多个显示设备2110A提供输出。在至少一个实施例中,与I/O集线 器2107耦合的一个或更多个显示设备2110A可以包括本地、内部或 嵌入式显示设备。

[0192] 在至少一个实施例中,处理子系统2101包括经由总线或其他 通信链路2113耦合到内存集线器2105的一个或更多个并行处理器 2112。在至少一个实施例中,通信链路2113可以是许多基于标准的 通信链路技术或协议中的一种,例如但不限于PCI Express,或者可 以是针对供应商的通信接口或通信结构。在至少一个实施例中,一个 或更多个并行处理器2112形成计算集中的并行或向量处理系统,该 系统可以包括大量的处理核心和/或处

理集群,例如多集成核心 (MIC)处理器。在至少一个实施例中,一个或更多个并行处理器2112 形成可以将像素输出到经由I/O集线器2107耦合的一个或更多个显 示设备2110A之一的图形处理子系统。在至少一个实施例中,一个 或更多个并行处理器2112还可以包括显示控制器和显示接口(未示 出),以使得能够直接连接到一个或更多个显示设备2110B。

[0193] 在至少一个实施例中,系统存储单元2114可以连接到I/O集 线器2107,以提供用于计算系统2100的存储机制。在至少一个实施 例中,I/O开关2116可以用于提供接口机制,以实现I/O集线器2107 与其他组件之间的连接,例如可以集成到平台中的网络适配器2118和/或无线网络适配器2119,以及可以通过一个或更多个附加设备 2120添加的各种其他设备。在至少一个实施例中,网络适配器2118 可以是以太网适配器或另一有线网络适配器。在至少一个实施例中, 无线网络适配器2119可以包括Wi-Fi、蓝牙、近场通信(NFC)的一 个或更多个或其他包括一个或更多个无线电的网络设备。

[0194] 在至少一个实施例中,计算系统2100可以包括未明确示出的 其他组件,包括USB或其他端口连接、光存储驱动器、视频捕获设 备等,也可以连接到I/O集线器2107。在至少一个实施例中,对图 21中的各个组件进行互连的通信路径可以使用任何合适的协议来实现,诸如基于PCI(外围组件互连)的协议(例如,PCI-Express), 或其他总线或点对点通信接口和/或协议(例如,NV-链路高速互连或 互连协议)。

[0195] 在至少一个实施例中,一个或更多个并行处理器2112包括针 对图形和视频处理而优化的电路(包括例如视频输出电路),并构成 图形处理单元(GPU)。在至少一个实施例中,一个或更多个并行处 理器2112包括针对通用处理而优化的电路。在至少实施例中,计算 系统2100的组件可以与单个集成电路上的一个或更多个其他系统元 件集成。例如,在至少一个实施例中,一个或更多个并行处理器2112、内存集线器2105、一个或更多个处理器2102和I/O集线器2107可以 被集成到片上系统(SoC)集成电路中。在至少一个实施例中,计算 系统2100的组件可以被集成到单个封装中以形成系统级封装(SIP) 配置。在至少一个实施例中,计算系统2100的组件的至少一部分可 以被集成到多芯片模块(MCM)中,该多芯片模块可以与其他多芯 片模块互连到模块化计算系统中。

[0196] 推理和/或训练逻辑1015用于执行与一个或更多个实施例相关 联的推理和/或训练操作。下面结合图10和/或图11提供关于推理和/ 或训练逻辑1015的细节。在至少一个实施例中,推理和/或训练逻辑 1015可以在图21的系统2100中使用,用于至少部分地基于使用神 经网络训练操作、神经网络功能和/或架构或本文所述的神经网络用 例计算出的权重参数来推理或预测操作。

[0197] 处理器

[0198] 图22示出了根据至少一个实施例的并行处理器2200。在至少 一个实施例中,并行处理器2200的各种组件可以使用一个或更多个 集成电路设备来实现,例如可编程处理器、专用集成电路(ASIC) 或现场可编程门阵列(FPGA)。在至少一个实施例中,所示的并行 处理器2200是根据示例性实施例的图21所示的一个或更多个并行处 理器2112的变体。

[0199] 在至少一个实施例中,并行处理器2200包括并行处理单元 2202。在至少一个实施例中,并行处理单元2202包括I/O单元2204, 其使得能够与其他设备进行通信,包括并行处理单元2202的其他实 例。在至少一个实施例中,I/O单元2204可以直接连接到其他设备。在至少一个实施例中,I/O单元2204通过使用集线器或交换机接口 (例如,内存集线器

2105)与其他设备连接。在至少一个实施例中，内存集线器2105与I/O单元2204之间的连接形成通信链路2113。在 至少一个实施例中，I/O单元2204与主机接口2206和内存交叉开关2216连接，其中主机接口2206接收用于执行处理操作的命令，而内 存交叉开关2216接收用于执行内存操作的命令。

[0200] 在至少一个实施例中，当主机接口2206经由I/O单元2204接 收命令缓冲区时，主机接口2206可以引导工作操作以执行那些命令 到前端2208。在至少一个实施例中，前端2208与调度器2210耦合， 调度器2210配置成将命令或其他工作项分配给处理集群阵列2212。 在至少一个实施例中，调度器2210确保在将任务分配给处理集群阵 列2212中的处理集群阵列2212之前，处理集群阵列2212被正确地 配置并且处于有效状态。在至少一个实施例中，调度器2210通过在 微控制器上执行的固件逻辑来实现。在至少一个实施例中，微控制器 实现的调度器2210可配置成以粗粒度和细粒度执行复杂的调度和工 作分配操作，从而实现对在处理阵列2212上执行的线程的快速抢占 和环境切换。在至少一个实施例中，主机软件可以证明用于通过多个 图形处理门铃之一在处理阵列2212上进行调度的工作负载。在至少 一个实施例中，工作负载然后可以由包括调度器2210的微控制器内 的调度器2210逻辑在处理阵列2212上自动分配。

[0201] 在至少一个实施例中，处理集群阵列2212可以包括多达“N”个处理集群(例如，集群2214A、集群2214B到集群2214N)。在至 少一个实施例中，处理集群阵列2212的每个集群2214A-2214N可以 执行大量并发线程。在至少一个实施例中，调度器2210可以使用各 种调度和/或工作分配算法将工作分配给处理集群阵列2212的集群 2214A-2214N，其可以根据每种程序或计算类型产生的工作负载而变 化。在至少一个实施例中，调度可以由调度器2210动态地处理，或 者可以在配置为由处理集群阵列2212执行的程序逻辑的编译期间部分地由编译器逻辑来辅助。在至少一个实施例中，可将处理集群阵列 2212的不同的集群2214A-2214N分配用于处理不同类型的程序或用 于执行不同类型的计算。

[0202] 在至少一个实施例中，处理集群阵列2212可以配置成执行各 种类型的并行处理操作。在至少一个实施例中，处理集群阵列2212 配置成执行通用并行计算操作。例如，在至少一个实施例中，处理集 群阵列2212可以包括执行处理任务的逻辑，该处理任务包括对视频 和/或音频数据的过滤，执行建模操作，包括物理操作以及执行数据 转换。

[0203] 在至少一个实施例中，处理集群阵列2212配置成执行并行图 形处理操作。在至少一个实施例中，处理集群阵列2212可以包括附 加逻辑以支持这种图形处理操作的执行，包括但不限于执行纹理操作 的纹理采样逻辑，以及镶嵌逻辑和其他顶点处理逻辑。在至少一个实 施例中，处理集群阵列2212可以配置成执行与图形处理有关的着色 器程序，例如但不限于顶点着色器、曲面细分着色器、几何着色器和 像素着色器。在至少一个实施例中，并行处理单元2202可以经由I/O 单元2204从系统内存传送数据以进行处理。在至少一个实施例中， 在处理期间，可以在处理期间将传送的数据存储到片上内存(例如， 并行处理器内存2222)，然后将其写回到系统内存。

[0204] 在至少一个实施例中，当并行处理单元2202用于执行图形处 理时，调度器2210可以配置成将处理工作负载划分为近似相等大小 的任务，以更好地将图形处理操作分配给处理集群阵列2212的多个 集群2214A-2214N。在至少一个实施例中，处理集群阵列2212的部 分可以配置成执行不同类型的处理。例如，在至少一个实施例中，第 一部分可以配置成

执行顶点着色和拓扑生成，第二部分可以配置成执 行镶嵌和几何着色，并且第三部分可以配置成执行像素着色或其他屏 幕空间操作，以生成用于显示的渲染图像。在至少一个实施例中，可 以将由集群2214A-2214N中的一个或更多个产生的中间数据存储在 缓冲区中，以允许在集群2214A-2214N之间传输中间数据以进行进 一步处理。

[0205] 在至少一个实施例中，处理集群阵列2212可以经由调度器 2210接收要执行的处理任务，该调度器2210从前端2208接收定义 处理任务的命令。在至少一个实施例中，处理任务可以包括要被处理 的数据的索引，例如可以包括表面(补丁)数据、原始数据、顶点数据和/或像素数据，以及状态参数和定义如何处理数据的命令(例如， 要执行什么程序)。在至少一个实施例中，调度器2210可以配置成 获取与任务相对应的索引，或者可以从前端2208接收索引。在至少 一个实施例中，前端2208可以配置成确保在启动由传入命令缓冲区(例如，批缓冲区(batch-buffer)、推送缓冲区等)指定的工作负载 之前，处理集群阵列2212配置成有效状态。

[0206] 在至少一个实施例中，并行处理单元2202的一个或更多个实 例中的每一个可以与并行处理器内存2222耦合。在至少一个实施例 中，可以经由内存交叉开关2216访问并行处理器内存2222，所述内 存交叉开关2216可以接收来自处理集群阵列2212以及I/O单元2204 的内存请求。在至少一个实施例中，内存交叉开关2216可以经由内 存接口2218访问并行处理器内存2222。在至少一个实施例中，内存 接口2218可以包括多个分区单元(例如，分区单元2220A、分区单 元2220B到分区单元2220N)，其可各自耦合至并行处理器内存2222的一部分(例如，内存单元)。在至少一个实施例中，多个分区单元 2220A-2220N为配置为等于内存单元的数量，使得第一分区单元 2220A具有对应的第一内存单元2224A，第二分区单元2220B具有对 应的内存单元2224B，第N分区单元2220N具有对应的第N内存单 元2224N。在至少一个实施例中，分区单元2220A-2220N的数量可 以不等于内存设备的数量。

[0207] 在至少一个实施例中，内存单元2224A-2224N可以包括各种 类型的内存设备，包括动态随机存取内存(DRAM)或图形随机存取 内存，例如同步图形随机存取内存(SGRAM)，包括图形双倍数据 速率(GDDR)内存。在至少一个实施例中，内存单元2224A-2224N 还可包括3D堆叠内存，包括但不限于高带宽内存(HBM)。在至少 一个实施例中，可以跨内存单元2224A-2224N来存储诸如帧缓冲区 或纹理映射的渲染目标，从而允许分区单元2220A-2220N并行地写 入每个渲染目标的部分，以有效地使用并行处理器内存2222的可用 带宽。在至少一个实施例中，可以排除并行处理器内存2222的本地 实例，以有利于利用系统内存与本地高速缓存内存结合的统一内存设 计。

[0208] 在至少一个实施例中，处理集群阵列2212的集群2214A-2214N 中的任何一个都可以处理将被写入并行处理器内存2222内的任何内 存单元2224A-2224N中的数据。在至少一个实施例中，内存交叉开 关2216可以配置为将每个集群2214A-2214N的输出传输到任何分区 单元2220A-2220N或另一个集群2214A-2214N，集群2214A-2214N 可以对输出执行其他处理操作。在至少一个实施例中，每个集群 2214A-2214N可以通过内存交叉开关2216与内存接口2218通信，以 从各种外部存储设备读取或写入各种外部存储设备。在至少一个实施例中，内存交叉开关2216具有到内存接口2218的连接以与I/O单元 2204通信，以及到并行处理器内存2222的本地实例的连接，从而使 不同处理集群2214A-2214N内的处理单元与系统内存或不是并行处 理单元2202本地的其他内存进行通信。在至少一个实施例中，内存

交叉开关2216可以使用虚拟通道来分离集群2214A-2214N和分区单 元2220A-2220N之间的业务流。

[0209] 在至少一个实施例中,可以在单个插入卡上提供并行处理单元 2202的多个实例,或者可以将多个插入卡互连。在至少一个实施例 中,并行处理单元2202的不同实例可以配置成相互操作,即使不同 实例具有不同数量的处理核心,不同数量的本地并行处理器内存和/ 或其他配置差异。例如,在至少一个实施例中,并行处理单元2202 的一些实例可以包括相对于其他实例而言更高精度的浮点单元。在至 少一个实施例中,结合并行处理单元2202或并行处理器2200的一个 或更多个实例的系统可以以各种配置和形式因素来实现,包括但不限 于台式机、膝上型计算机或手持式个人计算机、服务器、工作站、游 戏机和/或嵌入式系统。

[0210] 图23是根据至少一个实施例的分区单元2320的框图。在至少 一个实施例中,分区单元2320是图22的分区单元2220A-2220N之 一的实例。在至少一个实施例中,分区单元2320包括L2高速缓存 2321、帧缓冲区接口2325和ROP 2326(光栅操作单元)。L2高速 缓存2321是读/写高速缓存,其配置成执行从内存交叉开关2316和 ROP 2326接收的加载和存储操作。在至少一个实施例中,L2高速缓 存2321将读取未命中和紧急回写请求输出到帧缓冲区接口2325以进 行处理。在至少一个实施例中,还可以经由帧缓冲区接口2325将更 新发送到帧缓冲区以进行处理。在至少一个实施例中,帧缓冲区接口 2325与并行处理器内存中的内存单元(诸如图22的内存单元2224A -2224N(例如,在并行处理器内存2222内))之一相互作用。

[0211] 在至少一个实施例中,ROP 2326是一种处理单元,其执行光 栅操作,诸如模版、z测试、混合等。在至少一个实施例中,ROP 2326 然后输出存储在图形内存中的处理后的图形数据。在至少一个实施例 中,ROP 2326包括压缩逻辑以压缩被写入内存的深度或颜色数据并 解压缩从内存读取的深度或颜色数据。在至少一个实施例中,压缩逻 辑可以是利用多种压缩算法中的一种或更多种的无损压缩逻辑。ROP 2326执行的压缩的类型可以基于要压缩的数据的统计特性而变化。 例如,在至少一个实施例中,基于每图块基础上的深度和颜色数据执 行增量颜色压缩。

[0212] 在至少一个实施例中,ROP 2326包括在每个处理集群内(例 如,图22的集群2214A-2214N),而不是在分区单元2320内。在至 少一个实施例中,通过内存交叉开关2316而不是像素片段数据传输 对像素数据的读取和写入请求。在至少一个实施例中,经处理的图形 数据可以在显示设备上(诸如图21的一个或更多个显示设备2110之 一)显示,由处理器2102路由以供进一步处理,或者由图22的并行 处理器2200内的处理实体之一路由以供进一步处理。

[0213] 图24是根据至少一个实施例的并行处理单元内的处理集群 2414的框图。在至少一个实施例中,处理集群是图22的处理集群2214A-2214N之一的实例。在至少一个实施例中,处理集群2414可 以配置成并行执行许多线程,其中术语“线程”是指在特定的一组输入数据上执行的特定程序的实例。在至少一个实施例中,单指令多数 据(SIMD)指令发布技术用于支持大量线程的并行执行而无需提供 多个独立的指令单元。在至少一个实施例中,使用单指令多线程 (SIMT)技术来支持并行执行大量一般同步的线程,这使用了公共 指令单元,该公共指令单元配置成向每个处理集群内的一组处理引擎 发出指令。

[0214] 在至少一个实施例中,可以通过将处理任务分配给SIMT并行 处理器的管线管理器2432来控制处理集群2214的操作。在至少一个 实施例中,管线管理器2432从图22的调度器2210接收指令,通过 图形多处理器2434和/或纹理单元2436管理这些指令的执行。在至少一个实施例中,图形多处理器2434是SIMT并行处理器的示例性 实例。然而,在至少一个实施例中,处理集群2414内可以包括不同 架构的各种类型的SIMT并行处理器。在至少一个实施例中,在处理 集群2414内可以包括图形多处理器2434的一个或更多个实例。在至 少一个实施例中,图形多处理器2434可以处理数据,并且数据交叉 开关2440可以用于将处理后的数据分发到多个可能的目的(包括其 他着色器单元)地之一。在至少一个实施例中,管线管理器2432可 以通过指定要经由数据交叉开关2240分配的处理后的数据的目的地 来促进处理后的数据的分配。

[0215] 在至少一个实施例中,处理集群2414内的每个图形多处理器 2434可以包括相同的一组功能执行逻辑(例如,算术逻辑单元、加 载存储单元等)。在至少一个实施例中,可以以管线方式配置功能执 行逻辑,其中可以在先前的指令完成之前发出新的指令。在至少一个 实施例中,功能执行逻辑支持多种操作,包括整数和浮点算术、比较 操作、布尔运算、移位和各种代数函数的计算。在至少一个实施例中, 可以利用相同的功能单元硬件来执行不同的操作,并且可以存在功能 单元的任何组合。

[0216] 在至少一个实施例中,传送到处理集群2414的指令构成线程。 在至少一个实施例中,跨一组并行处理引擎执行的一组线程是线程 组。在至少一个实施例中,线程组在不同的输入数据上执行程序。在 至少一个实施例中,线程组内的每个线程可被分配给图形多处理器 2434内的不同处理引擎。在至少一个实施例中,线程组可包括比图 形多处理器2434内的多个处理引擎更少的线程。在至少一个实施例 中,当线程组包括的线程数少于处理引擎的数量时,一个或更多个处 理引擎在正在处理该线程组的循环期间可能是空闲的。在至少一个实 施例中,线程组还可以包括比图形多处理器2434内的多个处理引擎 更多的线程。在至少一个实施例中,当线程组包括比图形多处理器 2434内的处理引擎的数量更多的线程时,可以在连续的时钟周期内 执行处理。在至少一个实施例中,可以在图形多处理器2434上同时 执行多个线程组。

[0217] 在至少一个实施例中,图形多处理器2434包括内部高速缓存 内存,以执行加载和存储操作。在至少一个实施例中,图形多处理器 2434可以放弃内部高速缓存并使用处理集群2414内的高速缓存内存 (例如,L1高速缓存2248)。在至少一个实施例中,每个图形多处理器2434还可以访问分区单元(例如,图22的分区单元 2220A-2220N)内的L2高速缓存,这些分区单元在所有处理集群2414 之间共享并且可以用于在线程之间传输数据。在至少一个实施例中, 图形多处理器2434还可以访问片外全局内存,其可以包括本地并行 处理器内存和/或系统内存中的一个或更多个。在至少一个实施例中, 并行处理单元2402外部的任何内存都可以用作全局内存。在至少一 个实施例中,处理集群2414包括图形多处理器2434的多个实例,它 们可以共享可以存储在L1高速缓存2448中的公共指令和数据。

[0218] 在至少一个实施例中,每个处理集群2414可以包括配置成将 虚拟地址映射为物理地址的内存管理单元(“MMU”)2445。在至 少一个实施例中,MMU 2445的一个或更多个实例可以驻留在图22 的内存接口2218内。在至少一个实施例中,MMU 2245包括一组页 表条目(PTE),其用于将虚拟地址映射到图块(谈论有关图块的更 多信息)的物理地址以及可选

地映射到高速缓存行索引。在至少一个 实施例中,MMU 2445可以包括地址转换后备缓冲区(TLB)或可以 驻留在图形多处理器2434或L1高速缓存或处理集群2414内的高速 缓存。在至少一个实施例中,处理物理地址以分配表面数据访问局部 性,以便在分区单元之间进行有效的请求交织。在至少一个实施例中, 高速缓存行索引可以用于确定对高速缓存线的请求是命中还是未命 中。

[0219] 在至少一个实施例中,可以配置处理集群2414,使得每个图 形多处理器2434耦合到纹理单元2436,以执行纹理映射操作,例如, 可以涉及确定纹理样本位置、读取纹理数据以及过滤纹理数据。在至 少一个实施例中,根据需要从内部纹理L1高速缓存(未示出)或从图形多处理器2434内的L1高速缓存中读取纹理数据,并从L2高速 缓存、本地并行处理器内存或系统内存中获取纹理数据。在至少一个 实施例中,每个图形多处理器2434将处理后的任务输出到数据交叉 开关2440,以将处理后的任务提供给另一处理集群2414以进行进一步处理或将处理后的任务存储在L2高速缓存、本地并行处理器内存、或经由内存交叉开关2416的系统内存中。在至少一个实施例中, preROP 2442(光栅前操作单元)配置成从图形多处理器2434接收数 据,将数据引导至ROP单元,该ROP单元可以与本文所述的分区单 元(例如,图22的分区单元2220A-2220N)一起定位。在至少一个 实施例中,PreROP 2442单元可以执行用于颜色混合的优化、组织像 素颜色数据以及执行地址转换。

[0220] 推理和/或训练逻辑1015用于执行与一个或更多个实施例相关 联的推理和/或训练操作。下面结合图10和/或图11提供关于推理和/ 或训练逻辑1015的细节。在至少一个实施例中,推理和/或训练逻辑 1015可以在图形处理集群2214中用于至少部分地基于使用本文描述 的神经网络训练操作、神经网络功能和/或架构或神经网络用例计算 的权重参数来进行推理或预测操作。

[0221] 图25示出了根据至少一个实施例的图形多处理器2534。在至 少一个实施例中,图形多处理器2534与处理集群2514的管线管理器 2532耦合。在至少一个实施例中,图形多处理器2534具有执行管线, 该执行管线包括但不限于指令高速缓存2552、指令单元2554、地址 映射单元2556、寄存器文件2558、一个或更多个通用图形处理单元 (GPGPU)核心2562和一个或更多个加载/存储单元2566。GPGPU 核心2562和加载/存储单元2566与高速缓存内存2572和共享内存 2570通过内存和高速缓存互连2568耦合。

[0222] 在至少一个实施例中,指令高速缓存2552从管线管理器2532 接收要执行的指令流。在至少一个实施例中,将指令高速缓存在指令 高速缓存2552中并将其分派以供指令单元2554执行。在一个实施例 中,指令单元2554可以分派指令作为线程组(例如,线程束),将线程组的每个线程分配给GPGPU核心2562内的不同执行单元。在 至少一个实施例中,指令可以通过在统一地址空间内指定地址来访问 任何本地、共享或全局地址空间。在至少一个实施例中,地址映射单 元2556可以用于将统一地址空间中的地址转换成可以由加载/存储单 元2566访问的不同的内存地址。

[0223] 在至少一个实施例中,寄存器文件2558为图形多处理器2534 的功能单元提供了一组寄存器。在至少一个实施例中,寄存器文件 2558为连接到图形多处理器2534的功能单元(例如,GPGPU核心 2562、加载/存储单元2566)的数据路径的操作数提供了临时存储。 在至少一个实施例中,在每个功能单元之间划分寄存器文件2558, 使得为每个功能单元分配寄存器文件2558的专用部分。在至少一个 实施例中,寄存器文件2558在图形多处理器2534

正在执行的不同线 程束之间划分。

[0224] 在至少一个实施例中,GPGPU核心2562可以各自包括用于执 行图形多处理器2534的指令的浮点单元(FPU)和/或整数算术逻辑 单元(ALU)。GPGPU核心2562在架构上可以相似或架构可能有所 不同。在至少一个实施例中,GPGPU核心2562的第一部分包括单精 度FPU和整数ALU,而GPGPU核心的第二部分包括双精度FPU。 在至少一个实施例中,FPU可以实现用于浮点算法的IEEE 754-2008 标准或启用可变精度浮点算法。在至少一个实施例中,图形多处理器 2534可以另外包括一个或更多个固定功能或特殊功能单元,以执行 特定功能,诸如复制矩形或像素混合操作。在至少一个实施例中, GPGPU核心中的一个或更多个也可以包括固定或特殊功能逻辑。

[0225] 在至少一个实施例中,GPGPU核心2562包括能够对多组数据 执行单个指令的SIMD逻辑。在一个实施例中,GPGPU核心2562可 以物理地执行SIMD4、SIMD8和SIMD9指令,并且在逻辑上执行 SIMD1、SIMD2和SIMD32指令。在至少一个实施例中,用于GPGPU 核心的SIMD指令可以在编译时由着色器编译器生成,或者在执行针 对单程序多数据(SPMD)或SIMT架构编写和编译的程序时自动生 成。在至少一个实施例中,可以通过单个SIMD指令来执行为SIMT 执行模型配置的程序的多个线程。例如,在至少一个实施例中,可以 通过单个SIMD8逻辑单元并行执行执行相同或相似操作的八个 SIMT线程。

[0226] 在至少一个实施例中,内存和高速缓存互连2568是将图形多 处理器2534的每个功能单元连接到寄存器文件2558和共享内存2570 的互连网络。在至少一个实施例中,内存和高速缓存互连2568是交 叉开关互连,其允许加载/存储单元2566在共享内存2570和寄存器 文件2558之间实现加载和存储操作。在至少一个实施例中,寄存器 文件2558可以以与GPGPU核心2562相同的频率操作,从而在 GPGPU核心2562和寄存器文件2558之间进行数据传输的延迟非常 低。在至少一个实施例中,共享内存2570可以用于启用在图形多处 理器2534内的功能单元上执行的线程之间的通信。在至少一个实施 例中,高速缓存内存2572可以用作例如数据高速缓存,以高速缓存 在功能单元和纹理单元2536之间通信的纹理数据。在至少一个实施 例中,共享内存2570也可以用作程序管理的高速缓存。在至少一个 实施例中,除了存储在高速缓存内存2572中的自动高速缓存的数据 之外,在GPGPU核心2562上执行的线程还可以以编程方式将数据 存储在共享内存中。

[0227] 在至少一个实施例中,如本文所述的并行处理器或GPGPU通 信地耦合到主机/处理器核心,以加速图形操作、机器学习操作、图 案分析操作以及各种通用GPU(GPGPU)功能。在至少一个实施例 中,GPU可以通过总线或其他互连(例如,诸如PCIe或NVLink的 高速互连)通信地耦合到主机处理器/核心。在至少一个实施例中, GPU可以与核心集成在相同的封装或芯片上,并通过内部处理器总 线/互连(即,封装或芯片的内部)通信地耦合到核心。在至少一个 实施例中,不管GPU连接的方式如何,处理器核心可以以工作描述 符中包含的命令/指令序列的形式向GPU分配工作。在至少一个实施 例中,GPU然后使用专用电路/逻辑来有效地处理这些命令/指令。

[0228] 推理和/或训练逻辑1015用于执行与一个或更多个实施例相关 联的推理和/或训练操作。下面结合图10和/或图11提供关于推理和/ 或训练逻辑1015的细节。在至少一个实施例中,推理和/或训练逻辑 1015可以在图形多处理器2234中用于至少部分地基于使用本文描述 的神经网络训练操作、神经网络功能和/或架构或神经网络用例计算 的权重参数

来进行推理或预测操作，。

[0229] 图26是根据至少一个实施例的说明用于处理器2600的微架构 的框图，该处理器2600可以包括用于执行指令的逻辑电路。在至少 一个实施例中，处理器2600可以执行指令，包括x86指令、ARM指 令、用于专用集成电路(ASIC)的专用指令等。在至少一个实施例中，处理器2610可以包括用于存储封装数据的寄存器，例如作为加 利福尼亚州圣克拉拉市英特尔公司采用MMX技术启用的微处理器 中的64位宽MMXTM寄存器。在至少一个实施例中，整数和浮点数 形式可用的MMX寄存器可以与封装的数据元素一起运行，所述封装 的数据元素伴随单指令多数据(“SIMD”)和流式SIMD扩展(“SSE”) 指令。在至少一个实施例中，与SSE2、SSE3、SSE4、AVX或更高 版本(一般称为“SSEx”)技术有关的128位宽XMM寄存器可以保存此类封装数据操作数。在至少一个实施例中，处理器2610可以 执行指令以加速机器学习或深度学习算法、训练或推理。

[0230] 在至少一个实施例中，处理器2600包括有序前端(“前端”) 2601，以提取要执行的指令并准备稍后在处理器管线中使用的指令。 在至少一个实施例中，前端2601可以包括几个单元。在至少一个实 施例中，指令预取器2626从内存中获取指令并将指令提供给指令解码器2628，指令解码器2628又对指令进行解码或解释。例如，在至 少一个实施例中，指令解码器2628将接收到的指令解码为机器可执 行的所谓的“微指令”或“微操作”(也称为“微操作”或“微指令”) 的一个或更多个操作。在至少一个实施例中，指令解码器2628将指 令解析为操作码以及相应的数据和控制字段，其可以由微架构用来使 用以根据至少一个实施例来执行操作。在至少一个实施例中，跟踪高 速缓存2630可以将解码的微指令组装成微指令队列2634中的程序排 序的序列或追踪以供执行。在至少一个实施例中，当追踪高速缓存2630遇到复杂指令时，微码ROM 2632提供完成操作所需的微指令。

[0231] 在至少一个实施例中，可以将一些指令转换成单个微操作，而 另一些指令则需要几个微操作来完成全部操作。在至少一个实施例 中，如果需要多于四个的微指令来完成一条指令，则指令解码器2628 可以访问微码ROM 2632以执行指令。在至少一个实施例中，可以将 指令解码为少量的微指令以在指令解码器2628处进行处理。在至少 一个实施例中，如果需要多个微指令完成操作，则可以将指令存储在 微码ROM 2632中。在至少一个实施例中，追踪高速缓存器2630参 考入口点可编程逻辑阵列(“PLA”)以确定正确的微指令指针，用 于根据至少一个实施例从微码ROM 2632读取微码序列以完成一个 或更多个指令。在至少一个实施例中，在微码ROM 2632完成对指令 的微操作排序之后，机器的前端2601可以恢复从追踪高速缓存2630 获取微操作。

[0232] 在至少一个实施例中，乱序执行引擎(“乱序引擎”)2603 可以准备用于执行的指令。在至少一个实施例中，乱序执行逻辑具有 多个缓冲区，以使指令流平滑并重新排序，以在指令沿管线下降并被 调度执行时优化性能。乱序执行引擎2603包括但不限于分配器/寄存 器重命名器2640、内存微指令队列2642、整数/浮点微指令队列2644、内存调度器2646、快速调度器2602、慢速/通用浮点调度器(“慢速 /通用FP调度器”)2604和简单浮点调度器(“简单FP调度器”) 2606。在至少一个实施例中，快速调度器2602、慢速/通用浮点调度 器2604和简单浮点调度器2606也统称为“微指令调度器2602、2604、 2606”。分配器/寄存器重命名器2640分配每个微指令按序列执行所 需要的机器缓冲区和资源。在至少一个实施例中，分配器/寄存器重 命名器2640将逻辑寄存器重命名为寄存器文件中的条目。在至少一

个实施例中，分配器/寄存器重命名器2640还为两个微指令队列之一 中的每个微指令分配条目，内存微指令队列2642用于内存操作和整 数/浮点微指令队列2644用于非内存操作，在内存调度器2646和微 指令调度器2602、2604、2606的前面。在至少一个实施例中，微指令调度器2602、2604、2606基于它们的从属输入寄存器操作数源的 就绪性和需要完成的执行资源微指令的可用性来确定何时准备好执 行微指令。在至少一个实施例中，至少一个实施例的快速调度器2602 可以在主时钟周期的每个一半上调度，而慢速/通用浮点调度器2604 和简单浮点调度器2606可以在每个主处理器时钟周期调度一次。在 至少一个实施例中，微指令调度器2602、2604、2606对调度端口进 行仲裁，以调度用于执行的微指令。

[0233] 在至少一个实施例中，执行块b11包括但不限于整数寄存器文 件/支路网络2608、浮点寄存器文件/支路网络(“FP寄存器文件/支 路网络”)2610、地址生成单元(“AGU”)2612和2614、快速算 术逻辑单元(“快速ALU”)2616和2618、慢速算术逻辑单元(“慢 速ALU”)2620、浮点ALU(“FP”)2622和浮点移动单元(“FP 移动”)2624。在至少一个实施例中，整数寄存器文件/支路网络2608 和浮点寄存器文件/旁路网络2610在本文中也称为“寄存器文件2608、 2610”。在至少一个实施例中，AGU 2612和2614、快速ALU 2616 和2618、慢速ALU 2620、浮点ALU 2622和浮点移动单元2624在本 文中也称为“执行单元2612、2614、2616、2618、2620、2622和2624”。 在至少一个实施例中，执行框b11可以包括但不限于任意数量(包括 零)和类型的寄存器文件、支路网络、地址生成单元和执行单元(以 任何组合)。

[0234] 在至少一个实施例中，寄存器文件2608、2610可以布置在微 指令调度器2602、2604、2606与执行单元2612、2614、2616、2618、 2620、2622和2624之间。在至少一个实施例中，整数寄存器文件/ 支路网络2608执行整数运算。在至少一个实施例中，浮点寄存器文件/支路网络2610执行浮点操作。在至少一个实施例中，寄存器文件 2608、2610中的每一个可以包括但不限于支路网络，该支路网络可 以绕过或转发尚未写入寄存器文件中的刚刚完成的结果到新的从属 对象。在至少一个实施例中，寄存器文件2608、2610可以彼此通信数据。在至少一个实施例中，整数寄存器文件/支路网络2608可以包 括但不限于两个单独的寄存器文件、一个寄存器文件用于低阶32位 数据，第二寄存器文件用于高阶32位数据。在至少一个实施例中， 浮点寄存器文件/支路网络2610可以包括但不限于128位宽的条目，因为浮点指令通常具有宽度为64至128位的操作数。

[0235] 在至少一个实施例中，执行单元2612、2614、2616、2618、 2620、2622、2624可以执行指令。在至少一个实施例中，寄存器文 件2608、2610存储微指令需要执行的整数和浮点数据操作数值。在 至少一个实施例中，处理器2600可以包括但不限于任何数量的执行 单元2612、2614、2616、2618、2620、2622、2624及其组合。在至 少一个实施例中，浮点ALU 2622和浮点移动单元2624，可以执行浮 点、MMX、SIMD、AVX和SSE或其他操作，包括专门的机器学习 指令。在至少一个实施例中，浮点ALU 2622可以包括但不限于64 位乘64位浮点除法器，以执行除法、平方根和余数微操作。在至少 一个实施例中，可以用浮点硬件来处理涉及浮点值的指令。在至少一 个实施例中，可以将ALU操作传递给快速ALU 2616、2618。在至少 一个实施例中，快速ALUS 2616、2618可以以半个时钟周期的有效 延迟执行快速操作。在至少一个实施例中，大多数复杂的整数运算进 入慢速ALU 2620，因为慢速ALU 2620可以包括但不限于用于长延 迟类型操作的整数执行硬件，例如乘法器、移位、标志逻辑和分支处 理。在至少一个实施例中，内存加载/存储操作可以由AGUS 2612、 2614执行。在至少一个实施

例中，快速ALU 2616、快速ALU 2618 和慢速ALU 2620可以对64位数据操作数执行整数运算。在至少一 个实施例中，可以实现快速ALU 2616、快速ALU 2618和慢速ALU 2620以支持包括十六、三十二、128、256等的各种数据位大小。在 至少一个实施例中，浮点ALU 2622和浮点移动单元2624可以实现 为支持具有各种宽度的位的一定范围的操作数。在至少一个实施例 中，浮点ALU 2622和浮点移动单元2624可以结合SIMD和多媒体 指令对128位宽封装数据操作数进行操作。

[0236] 在至少一个实施例中，微指令调度器2602、2604、2606在父 加载完成执行之前调度从属操作。在至少一个实施例中，由于可以在 处理器2600中推测性地调度和执行微指令，处理器2600还可以包括 用于处理内存未命中的逻辑。在至少一个实施例中，如果数据高速缓 存中的数据加载未命中，则可能存在在管线中正在运行的从属操作，其使调度器暂时没有正确的数据。在至少一个实施例中，一种重放机 制追踪踪并重新执行使用不正确数据的指令。在至少一个实施例中， 可能需要重放从属操作并且可以允许完成独立操作。在至少一个实施 例中，处理器的至少一个实施例的调度器和重放机制也可以设计为捕 获用于文本串比较操作的指令序列。

[0237] 在至少一个实施例中，术语“寄存器”可以指代可以用作识别 操作数的指令的一部分的机载处理器存储位置。在至少一个实施例 中，寄存器可以是那些可以从处理器外部使用的寄存器(从程序员的 角度来看)。在至少一个实施例中，寄存器可能不限于特定类型的电 路。相反，在至少一个实施例中，寄存器可以存储数据、提供数据并 执行本文描述的功能。在至少一个实施例中，本文描述的寄存器可以 通过处理器内的电路使用多种不同技术来实现，例如专用物理寄存 器、使用寄存器重命名动态分配的物理寄存器、专用和动态分配的物 理寄存器的组合等。在至少一个实施例中，整数寄存器存储32位整 数数据。至少一个实施例的寄存器文件还包含八个用于封装数据的多 媒体SIMD寄存器。

[0238] 推理和/或训练逻辑1015用于执行与一个或更多个实施例相关 联的推理和/或训练操作。下面结合图10和/或图11提供关于推理和/ 或训练逻辑1015的细节。在至少一个实施例中，可以将推理和/或训 练逻辑1015的部分或全部并入执行块2611以及示出或未示出的其他 内存或寄存器。例如，在至少一个实施例中，本文描述的训练和/或 推理技术可以使用执行框2611中示出的一个或更多个ALU。此外， 权重参数可以存储在片上或片外内存和/或寄存器(示出或未示出) 中，该寄存器和/或寄存器配置执行块2611的ALU以执行一种或更 多种本文所述的机器学习算法、神经网络架构、用例或训练技术。

[0239] 图27示出了根据至少一个实施例的深度学习应用处理器 2700。在至少一个实施例中，深度学习应用处理器2700使用指令， 如果由深度学习应用处理器2700执行，则指令使深度学习应用处理 器2700执行贯穿本公开描述的一些或全部过程和技术。在至少一个实施例中，深度学习应用处理器2700是专用集成电路(ASIC)。在 至少一个实施例中，应用处理器2700执行矩阵乘法运算或者“硬连 线”到硬件中，作为执行一个或更多个指令或两者的结果。在至少一 个实施例中，深度学习应用处理器2700包括但不限于处理集群2710(1)-2710(12)、芯片间链路(“ICL”)2720(1)-2720(12)、 芯片间控制器(“ICC”)2730(1)-2730(2)、第二代高带宽内存 (“HBM2”)2740(1)-2740(4)、内存控制器(“Mem Ctrlr”) 2742(1)-2742(4)、高带宽内存物理层(“HBM PHY”)2744(1) -2744(4)、管理控制器中央处理单元(“管理控制器CPU”)2750、 串行外围设备接口、内部集成电路和通用输入/输出块(“SPI、

I2C、GPIO”)2760,外围组件互连快速控制器和直接内存访问块(“PCIe 控制器和DMA”)2770、以及十六通道外围组件互连快速端口(“PCI Express x 16”)2780。

[0240] 在至少一个实施例中,处理集群2710可以执行深度学习操作,包括基于一种或更多种训练技术计算的权重参数的推理或预测操作,包括本文所述的那些技术。在至少一个实施例中,每个处理集群2710 可以包括但不限于任何数量和类型的处理器。在至少一个实施例中,深度学习应用处理器2700可以包括任何数量和类型的处理集群 2700。在至少一个实施例中,芯片间链路2720是双向的。在至少一 个实施例中,芯片间链路2720和芯片间控制器2730使多个深度学习 应用处理器2700能够交换信息,包括从执行一个或更多个神经网络 中体现的一种或更多种机器学习算法而产生的激活信息。在至少一个 实施例中,深度学习应用处理器2700可以包括任意数量(包括零) 和类型的ICL 2720和ICC 2730。

[0241] 在至少一个实施例中,HBM2 2740提供总共32GB的内存。HBM2 2740(i)与内存控制器2742(i)和HBM PHY 2744(i)都相 关联。在至少一个实施例中,任何数量的HBM2 2740可以提供任何 类型和总量的高带宽内存,并且可以与任何数量(包括零)和类型的 内存控制器2742和HBM PHY 2744相关联。在至少一个实施例中,可以用任何数量和类型的块替换SPI、I2C、GPIO 3360、PCIe控制器 2760和DMA 2770和/或PCIe2780,以任何技术上可行的方式实现任 何数量和类型的通信标准。

[0242] 推理和/或训练逻辑1015用于执行与一个或更多个实施例相关 联的推理和/或训练操作。下面结合图10和/或图11提供关于推理和/ 或训练逻辑1015的细节。在至少一个实施例中,深度学习应用处理 器用于训练机器学习模型(例如神经网络),以预测或推理提供给深 度学习应用处理器2700的信息。在至少一个实施例中,深度学习应 用处理器2700用于基于已经由另一处理器或系统或由深度学习应用 处理器2700训练的经训练的机器学习模型(例如,神经网络)来推 理或预测信息。在至少一个实施例中,处理器2700可以用于执行本 文所述的一个或更多个神经网络用例。

[0243] 图28是根据至少一个实施例的神经形态处理器2800的框图。在至少一个实施例中,神经形态处理器2800可以从神经形态处理器 2800外部的源接收一个或更多个输入。在至少一个实施例中,这些 输入可以被传输到神经形态处理器2800内的一个或更多个神经元 2802。在至少一个实施例中,可以使用包括一个或更多个算术逻辑单 元(ALU)的电路或逻辑来实现神经元2802及其组件。在至少一个 实施例中,神经形态处理器2800可以包括但不限于成千上万个神经 元2802的实例,但是可以使用任何合适数量的神经元2802。在至少一个实施例中,神经元2802的每个实例可以包括神经元输入2804和 神经元输出2806。在至少一个实施例中,神经元2802可以生成可以 传输到神经元2802的其他实例的输入的输出。在至少一个实施例中,神经元输入2804和神经元输出2806可以经由突触2808互连。

[0244] 在至少一个实施例中,神经元2802和突触2808可以互连,使 得神经形态处理器2800操作以处理或分析由神经形态处理器2800接 收的信息。在至少一个实施例中,当通过神经元输入2804接收到的 输入超过阈值时,神经元2802可以发送输出脉冲(或“触发”或“峰 值”)。在至少一个实施例中,神经元2802可以对在神经元输入2804 处接收到的信号进行求和或积分。例如,在至少一个实施例中,神经 元2802可以实现为有泄漏的积分-触发神经元,其中如果求和(称为 “膜电位”)超过阈值,则神经元2802可以使用诸如sigmoid或阈值函数的传递函数来产生输出(或“触发”)。在至少一个实施例中,泄漏的积分-触发神经

元可以将在神经元输入2804处接收到的信号求 和成膜电位,并且可以应用衰减因子(或泄漏)以减小膜电位。在至 少一个实施例中,如果在神经元输入2804处接收到足够快以超过阈 值的多个输入信号(即,在膜电势衰减得太低而不能触发之前),则 泄漏的积分-触发神经元可能会触发。在至少一个实施例中,神经元 2802可以使用接收输入、将输入积分到膜电位、并衰减膜电位的电 路或逻辑来实现。在至少一个实施例中,可以对输入求平均,或者可 以使用任何其他合适的传递函数。此外,在至少一个实施例中,神经 元2802可以包括但不限于当将传递函数应用于神经元输入2804的结 果超过阈值时在神经元输出2806处产生输出尖峰的比较器电路或逻 辑。在至少一个实施例中,一旦神经元2802触发,它可以通过例如 将膜电位复位为0或另一合适的默认值来忽略先前接收的输入信息。 在至少一个实施例中,一旦膜电位被重置为0,则神经元2802可以 在合适的时间段(或修复期)之后恢复正常操作。

[0245] 在至少一个实施例中,神经元2802可以通过突触2808互连。 在至少一个实施例中,突触2808可以操作以将从第一神经元2802的 输出的信号传输到第二神经元2802的输入。在至少一个实施例中, 神经元2802可以在一个以上的突触2808实例上传输信息。在至少一 个实施例中,神经元输出2806的一个或更多个实例可以通过突触 2808的实例连接到同一神经元2802中神经元输入2804的实例。在 至少一个实施例中,相对于突触2808的那个实例,神经元2802的实 例产生要在突触2808的实例上传输的输出可以被称为“突触前神经元”。在至少一个实施例中,相对于突触2808的实例,神经元2802 的实例接收通过突触2808的实例传输的输入可以被称为“突触后神 经元”。在至少一个实施例中,关于突触2808的各种实例,因为神 经元2802的实例可以接收来自一个或更多个突触2808实例的输入, 并且还可以通过一个或更多个突触2808实例传输输出,因此神经元 2802的单个实例可以既是“突触前神经元”又是“突触后神经元”。

[0246] 在至少一个实施例中,神经元2802可以被组织成一层或更多 层。神经元2802的每个实例可以具有一个神经元输出2806,该神经 元输出2806可以通过一个或更多个突触2808扇出到一个或更多个神 经元输入2804。在至少一个实施例中,第一层2810中的神经元2802 的神经元输出2806可以连接到第二层2812中的神经元2802的神经 元输入2804。在至少一个实施例中,层2810可以被称为“前馈层”。 在至少一个实施例中,在第一层2810的实例中神经元2802的每个实 例可以扇出到第二层2812中的神经元2802的每个实例。在至少一个 实施例中,第一层2810可以被称为“完全连接的前馈层”。在至少 一个实施例中,在第二层2812的每个实例中的神经元2802的每个实 例扇出到少于在第三层2814中的神经元2802的所有实例。在至少一 个实施例中,第二层2812可以被称为“稀疏连接的前馈层”。在至 少一个实施例中,第二层2812中的神经元2802可以扇出到多个其他 层中的神经元2802,包括(相同)第二层2812中的神经元2802。在 至少一个实施例中,第二层2812可以被称为“循环层”。神经形态 处理器2800可以包括但不限于循环层和前馈层的任何合适的组合, 包括但不限于稀疏连接的前馈层和完全连接的前馈层。

[0247] 在至少一个实施例中,神经形态处理器2800可以包括但不限 于可重新配置的互连架构或专用硬连线互连,以将突触2808连接到 神经元2802。在至少一个实施例中,神经形态处理器2800可以包括 但不限于电路或逻辑,其根据神经网络拓扑结构和神经元扇入/扇出, 允许根据需要将突触分配给不同神经元2802。例如,在至少一个实 施例中,可以使

用互连结构(诸如片上网络)或通过专用连接将突触 2808连接到神经元2802。在至少一个实施例中,可以使用电路或逻 辑来实现突触互连及其组件。

[0248] 图29是图形处理器2900的框图,该图形处理器2900可以是 分立的图形处理单元,或者可以是与多个处理核心集成的图形处理 器。在至少一个实施例中,图形处理器2900经由内存映射的I/O接 口与图形处理器2900上的寄存器以及放置在内存中的命令进行通信。在至少一个实施例中,图形处理器2900包括用于访问内存的内 存接口2914。在至少一个实施例中,内存接口2914是到本地内存、一个或更多个内部高速缓存、一个或更多个共享的外部高速缓存和/ 或到系统内存的接口。

[0249] 在至少一个实施例中,图形处理器2900还包括用于将显示输 出数据驱动到显示设备2920的显示控制器2902。在至少一个实施例 中,显示控制器2902包括用于显示设备2920的一个或更多个覆盖平 面的硬件以及多层视频或用户界面元素的组合。在至少一个实施例 中,显示设备2920可以是内部或外部显示设备。在至少一个实施例 中,显示设备2920是头戴式显示设备,例如虚拟现实(VR)显示设 备或增强现实(AR)显示设备。在至少一个实施例中,图形处理器 2900包括视频编解码器引擎2906,以将媒体编码、解码或转码为一 种或更多种媒体编码格式,从一种或更多种媒体编码格式编码、解码 或转码,或在一种或更多种媒体编码格式之间进行编码、解码或转码, 所述媒体编码格式包括但不限于运动图像专家组(MPEG)格式(例 如MPEG-2),高级视频编码(AVC)格式(例如H.264/MPEG-4AVC,以及美国电影电视工程师协会(SMPTE)421M/VC-1)和联合图像专 家组(JPEG)格式(例如JPEG)和Motion JPEG(MJPEG)格式。

[0250] 在至少一个实施例中,图形处理器2900包括块图像传送 (BLIT)引擎2904,以执行二维(2D)光栅化器操作,包括例如位 边界块传送。但是,在至少一个实施例中,使用图形处理引擎(GPE) 2910的一个或更多个组件来执行2D图形操作。在至少一个实施例中, GPE 2910是用于执行图形操作(包括三维(3D)图形操作和媒体操 作)的计算引擎。

[0251] 在至少一个实施例中,GPE 2910包括用于执行3D操作的3D 管线2912,例如使用对3D图元形状(例如,矩形、三角形等)进行 操作的处理功能来渲染三维图像和场景。3D管线2912包括执行各种 任务和/或产生到3D/媒体子系统2915的执行线程的可编程和固定功能元件。虽然3D管线2912可用于执行媒体操作,但是在至少一个 实施例中,GPE 2910还包括媒体管线2916,其用于执行媒体操作, 诸如视频后处理和图像增强。

[0252] 在至少一个实施例中,媒体管线2916包括固定功能或可编程 逻辑单元,用于执行一种或更多种专门的媒体操作,例如视频解码加 速,视频去隔行和视频编码加速,代替或代表视频编解码器引擎2906。 在至少一个实施例中,媒体管线2916还包括线程产生单元,用于产 生线程以在3D/媒体子系统2915上执行。在至少一个实施例中,产 生的线程在3D/媒体子系统2915中包含的一个或更多个图形执行单 元上执行媒体操作的计算。

[0253] 在至少一个实施例中,3D/媒体子系统2915包括用于执行3D 管线2912和媒体管线2916产生的线程的逻辑。在至少一个实施例中, 3D管线2912和媒体管线2916将线程执行请求发送到3D/媒体子系 统2915,其包括用于仲裁各种请求并将其分派给可用线程执行资源的线程分派逻辑。在至少一个实施例中,执行资源包括用于处理3D 和媒体线程的图形执行单元的阵列。在至少一个实施例中,3D/媒体 子系统2915包括用于线程指令和数据的一个或更多个内部高速缓 存。在至少一个实施例中,子系统2915还包括共享内存,其包括寄 存

器和可寻址内存,以在线程之间共享数据并存储输出数据。

[0254] 推理和/或训练逻辑1015用于执行与一个或更多个实施例相关 联的推理和/或训练操作。下面结合图10和/或图11提供关于推理和/ 或训练逻辑1015的细节。在至少一个实施例中,可以将推理和/或训 练逻辑1015的部分或全部合并到处理器2900中。例如,在至少一个 实施例中,本文描述的训练和/或推理技术可以使用3D管线2912中 包含的一个或更多个ALU。此外,在至少一个实施例中,本文描述 的推理和/或训练操作可以使用除图10或图11所示的逻辑以外的逻 辑来完成。在至少一个实施例中,权重参数可以存储在片上或片外内 存和/或寄存器(示出或未示出)中,其配置图形处理器2900的ALU 以执行一种或更多种机器学习算法、神经网络架构、用例或本文介绍 的训练技术。

[0255] 图30是根据本文所述的至少一个实施例的图形处理器核心 3000的硬件逻辑的框图。在至少一个实施例中,图形处理器核心3000 被包括在图形核心阵列内。在至少一个实施例中,图形处理器核心 3000(有时称为核心切片)可以是模块化图形处理器内的一个或更多 个图形核心。在至少一个实施例中,图形处理器核心3000是一个图 形核心切片的示例,并且本文所述的图形处理器可以基于目标功率和 性能包络线包括多个图形核心切片。在至少一个实施例中,每个图形 核心3000可以包括与多个子核心3001A-3001F耦合的固定功能块 3030,也称为子切片,其包括通用和固定功能逻辑的模块块。

[0256] 在至少一个实施例中,固定功能块3030包括几何/固定功能管 线3036,例如,在较低性能和/或较低功率的图形处理器实施方式中, 该几何/固定功能管线3036可以由图形处理器3000中的所有子核心 共享。在至少一个实施例中,几何/固定功能管线3036包括3D固定 功能管线、视频前端单元,线程产生器和线程分派器以及管理统一返 回缓冲区的统一返回缓冲区管理器。

[0257] 在固定的至少一个实施例中,功能块3030还包括图形SoC接 口3037、图形微控制器3038和媒体管线3039。图形SoC接口3037 提供了图形核心3000以及片上集成电路系统中的其他处理器核心之 间的接口。在至少一个实施例中,图形微控制器3038是可编程子处理器,其可配置为管理图形处理器3000的各种功能,包括线程分派、调度和抢占。在至少一个实施例中,媒体管线3039包括有助于对包 括图像和视频数据的多媒体数据进行解码、编码、预处理和/或后处 理的逻辑。在至少一个实施例中,媒体管线3039经由对子核心 3001-3001F内的计算或采样逻辑的请求来实现媒体操作。

[0258] 在至少一个实施例中,SoC接口3037使图形核心3000能够与 通用应用处理器核心(例如,CPU)和/或SoC内的其他组件通信, 包括内存层次结构元素,诸如共享的最后一级高速缓存、系统RAM 和/或嵌入式片上或封装DRAM。在至少一个实施例中,SoC接口3037 还可以使得能够与SoC内的固定功能设备(例如,相机成像管线) 进行通信,并且使得能够使用和/或实现可以在图形核心3000和SoC 内部的CPU之间共享的全局内存原子。在至少一个实施例中,SoC 接口3037还可以实现用于图形核心3000的电源管理控制,并且启用 图形核心3000的时钟域与SoC内的其他时钟域之间的接口。在至少 一个实施例中,SoC接口3037使得能够从命令流转化器和全局线程 分派器接收命令缓冲区,其配置为向图形处理器内的一个或更多个图 形核心中的每一个提供命令和指令。在至少一个实施例中,当要执行 媒体操作时,可以将命令和指令分派给媒体管线3039,或者当要执 行图形处理操作时,可以将其分配给几何形状和固定功能管线(例如, 几何形状和固定功能管线3036、几何形状和固

定功能管线3014)。

[0259] 在至少一个实施例中,图形微控制器3038可以配置为对图形 核心3000执行各种调度和管理任务。在至少一个实施例中,图形微 控制器3038可以在子核心3001A-3001F中的执行单元(EU)阵列 3002A-3002F、3004A-3004F内的各种图形并行引擎上执行图形和/或计算工作负载调度。在至少一个实施例中,在包括图形核心3000的 SoC的CPU核心上执行的主机软件可以提交多个图形处理器门铃之 一的工作负载,其调用适当的图形引擎上的调度操作。在至少一个实 施例中,调度操作包括确定接下来要运行哪个工作负载、将工作负载 提交给命令流转化器、抢先在引擎上运行的现有工作负载、监控工作 负载的进度以及在工作负载完成时通知主机软件。在至少一个实施例 中,图形微控制器3038还可以促进图形核心3000的低功率或空闲状 态,从而为图形核心3000提供在图形核心3000内独立于操作系统和 /或系统上的图形驱动程序软件的跨低功率状态转换的保存和恢复寄 存器的能力。

[0260] 在至少一个实施例中,图形核心3000可以具有比所示的子核 心3001A-3001F多或少达N个模块化子核心。对于每组N个子核心, 在至少一个实施例中,图形核心3000还可以包括共享功能逻辑3010、共享和/或高速缓存内存3012、几何/固定功能管线3014以及附加的 固定功能逻辑3016以加速各种图形和计算处理操作。在至少一个实 施例中,共享功能逻辑3010可以包括可由图形核心3000内的每个N 个子核心共享的逻辑单元(例如,采样器、数学和/或线程间通信逻 辑)。共享和/或高速缓存内存3012可以是图形核心3000内的N个子核心3001A-3001F的最后一级高速缓存,并且还可以用作可由多个 子核心访问的共享内存。在至少一个实施例中,可以包括几何/固定 功能管线3014来代替固定功能块3030内的几何/固定功能管线3036, 并且可以包括相同或相似的逻辑单元。

[0261] 在至少一个实施例中,图形核心3000包括附加的固定功能逻 辑3016,其可以包括供图形核心3000使用的各种固定功能加速逻辑。 在至少一个实施例中,附加的固定功能逻辑3016包括用于仅位置着 色中使用的附加的几何管线。在仅位置着色中,存在至少两个几何管 线,而在几何/固定功能管线3016、3036内的完整几何管线和剔除管 线中,其是可以包括在附加的固定功能逻辑3016中的附加几何管线。 在至少一个实施例中,剔除管线是完整几何管线的修整版。在至少一 个实施例中,完整管线和剔除管线可以执行应用程序的不同实例,每 个实例具有单独的环境。在至少一个实施例中,仅位置着色可以隐藏 被丢弃的三角形的长剔除运行,从而在某些情况下可以更早地完成着 色。例如,在至少一个实施例中,附加固定功能逻辑3016中的剔除 管线逻辑可以与主应用程序并行执行位置着色器,并且通常比完整管 线更快地生成关键结果,因为剔除管线获取并遮蔽顶点的位置属性, 无需执行光栅化和将像素渲染到帧缓冲区。在至少一个实施例中,剔 除管线可以使用生成的临界结果来计算所有三角形的可见性信息,而 与这些三角形是否被剔除无关。在至少一个实施例中,完整管线(在 这种情况下可以称为重播管线)可以消耗可见性信息来跳过剔除的三 角形以仅遮盖最终传递到光栅化阶段的可见三角形。

[0262] 在至少一个实施例中,附加的固定功能逻辑3016还可包括机 器学习加速逻辑,例如固定功能矩阵乘法逻辑,用于实现包括用于机 器学习训练或推理的优化。

[0263] 在至少一个实施例中,在每个图形子核心3001A-3001F内包括 一组执行资源,其可用于响应于图形管线、媒体管线或着色器程序的 请求来执行图形、媒体和计算操作。在

至少一个实施例中,图形子核 心3001A-3001F包括多个EU阵列3002A-3002F、3004A-3004F,线 程分派和线程间通信(TD/IC)逻辑3003A-3003F,3D(例如,纹理) 采样器3005A-3005F,媒体采样器3006A-3006F,着色器处理器 3007A-3007F和共享本地内存(SLM)3008A-3008F。EU阵列 3002A-3002F、3004A-3004F每个都包含多个执行单元,这些执行单 元是通用图形处理单元,能够为图形、媒体或计算操作提供服务,执 行浮点和整数/定点逻辑运算,包括图形、媒体或计算着色器程序。 在至少一个实施例中,TD/IC逻辑3003A-3003F为子核心内的执行单 元执行本地线程分派和线程控制操作,并促进在子核心的执行单元上 执行的线程之间的通信。在至少一个实施例中,3D采样器 3005A-3005F可以将与纹理或其他3D图形相关的数据读取到内存 中。在至少一个实施例中,3D采样器可以基于与给定纹理相关联的配置的采样状态和纹理格式来不同地读取纹理数据。在至少一个实施 例中,媒体采样器3006A-3006F可以基于与媒体数据相关联的类型和 格式来执行类似的读取操作。在至少一个实施例中,每个图形子核心 3001A-3001F可以可替代地包括统一的3D和媒体采样器。在至少一 个实施例中,在每个子核心3001A-3001F内的执行单元上执行的线程 可以利用每个子核心内的共享本地内存3008A-3008F,以使在线程组 内执行的线程能够使用片上内存的公共池来执行。

[0264] 推理和/或训练逻辑1015用于执行与一个或更多个实施例相关 联的推理和/或训练操作。下面结合图10和/或图11提供关于推理和/ 或训练逻辑1015的细节。在至少一个实施例中,推理和/或训练逻辑 1015的部分或全部可以被合并到图形处理器3010中。例如,在至少 一个实施例中,本文描述的训练和/或推理技术可以使用在3D管线 3010、图形微控制器3038、几何和固定功能管线3014和3036或图 30中的其他逻辑中体现的一个或更多个ALU。此外,在至少一个实 施例中,本文描述的推理和/或训练操作可以使用除图10或图11所 示的逻辑以外的逻辑来完成。在至少一个实施例中,权重参数可以存 储在片上或片外内存和/或寄存器(示出或未示出)中,所述寄存器 配置图形处理器3000的ALU以执行一种或更多种本文介绍的机器学 习算法、神经网络架构、用例或训练技术。

[0265] 图31A-31B示出了根据至少一个实施例的包括图形处理器核 心的处理元件的阵列的线程执行逻辑3100。图31A示出了至少一个 实施例,其中使用了线程执行逻辑3100。图31B示出了根据至少一 个实施例的执行单元的示例性内部细节。

[0266] 如图31A中所示,在至少一个实施例中,线程执行逻辑3100 包括着色器处理器3102、线程分派器3104、指令高速缓存3106、包 括多个执行单元3108A-3108N的可缩放执行单元阵列、采样器3110、 数据高速缓存3112和数据端口3114。在至少一个实施例中,可缩放执行单元阵列可以例如基于工作负载的计算要求,通过启用或禁用一 个或更多个执行单元(例如,执行单元3108A、3108B、3108C、3108D 到3108N-1和3108N中的任何一个)来动态缩放。在至少一个实施 例中,可缩放执行单元通过链路到每个执行单元的互连结构互连。在至少一个实施例中,线程执行逻辑3100包括通过指令高速缓存3106、 数据端口3114、采样器3110和执行单元3108A-3108N中的一个或更 多个到内存(诸如系统内存或高速缓存内存)的一个或更多个连接。 在至少一个实施例中,每个执行单元(例如3108A)是独立的可编程 通用计算单元,其能够执行多个同时的硬件线程,同时针对每个线程 并行处理多个数据元素。在至少一个实施例中,执行单元 3108A-3108N的阵列可缩放以包括任意数量的单独执行单元。

[0267] 在至少一个实施例中，执行单元3108A-3108N主要用于执行 着色器程序。在至少一个实施例中，着色器处理器3102可以处理各 种着色器程序并经由线程分派器3104来分派与着色器程序相关联的 执行线程。在至少一个实施例中，线程分派器3104包括用于仲裁来 自图形和媒体管线的线程初始化庆祝以及在执行单元3108A-3108N 中的一个或更多个执行单元上实例化请求的线程的逻辑。例如，在至 少一个实施例中，几何管线可以将顶点、镶嵌或几何着色器分派到线 程执行逻辑以进行处理。在至少一个实施例中，线程分派器3104还 可以处理来自执行着色器程序的运行时线程产生请求。

[0268] 在至少一个实施例中，执行单元3108A-3108N支持一种指令 集，该指令集包括对许多标准3D图形着色器指令的本机支持，从而 使图形库(例如Direct 3D和OpenGL)中的着色器程序只需最少的 翻译即可执行。在至少一个实施例中，执行单元支持顶点和几何处理(例如，顶点程序、几何程序、顶点着色器)、像素处理(例如，像 素着色器、片段着色器)和通用处理(例如，计算和媒体着色器)。 在至少一个实施例中，每个执行单元3108A-3108N包括一个或更多 个算术逻辑单元(ALU)，能够执行多发出单指令多数据(SIMD)， 并且多线程操作实现了高效的执行环境尽管有更高的延迟内存访问。 在至少一个实施例中，每个执行单元内的每个硬件线程具有专用的高 带宽寄存器文件和相关的独立线程状态。在至少一个实施例中，执行 是每个时钟到管线的多次发出，管线能够进行整数、单精度和双精度 浮点运算、SIMD分支功能、逻辑运算、先验运算和其他其他运算。 在至少一个实施例中，在等待来自内存或一个或更多个共享功能的数 据时，执行单元3108A-3108N内的依赖性逻辑使等待线程休眠直到 返回了所请求的数据。在至少一个实施例中，在特定的等待线程正在 休眠时，硬件资源可以专用于处理其他线程。例如，在至少一个实施 例中，在与顶点着色器操作相关联的延迟期间，执行单元可以对像素 着色器、片段着色器或另一类型的着色器程序(包括不同的顶点着色 器)执行操作。

[0269] 在至少一个实施例中，执行单元3108A-3108N中的每一个执 行单元在数据元素的阵列上进行操作。在至少一个实施例中，多个数 据元素是“执行大小”或指令的通道数。在至少一个实施例中，执行 通道是用于指令内的数据元素访问、屏蔽和流控制的执行的逻辑单 元。在至少一个实施例中，多个通道可以独立于用于特定图形处理器 的多个物理算术逻辑单元(ALU)或浮点单元(FPU)。在至少一个 实施例中，执行单元3108A-3108N支持整数和浮点数据类型。

[0270] 在至少一个实施例中，执行单元指令集包括SIMD指令。在至 少一个实施例中，各种数据元素可以作为封装数据类型存储在寄存器 中，并且执行单元将基于那些元素的数据大小来处理各种元素。例如， 在至少一个实施例中，当对256位宽的向量进行操作时，将向量的 256位存储在寄存器中，并且执行单元对向量进行操作，作为四个单 独的64位封装数据元素(四字(QW)大小数据元素)、八个单独 的32位封装数据元素(双字(DW)大小数据元素)、十六个单独 的16位封装数据元素(单词(W)大小数据元素)或三十二个单独 的8位数据元素(字节(B)大小的数据元素)。然而，在至少一个 实施例中，不同的向量宽度和寄存器大小是可能的。

[0271] 在至少一个实施例中，一个或更多个执行单元可以被组合成具 有执行对于融合EU的线程控制逻辑(3107A-3107N)的融合执行单 元3109A-3109N。在至少一个实施例中，可以将多个EU合并成一个 EU组。在至少一个实施例中，融合EU组中的EU的数量可以配置 为

执行单独的SIMD硬件线程。融合的EU组中的EU的数量可以根 据各个实施例而变化。在至少一个实施例中，每个EU可以执行各种 SIMD宽度，包括但不限于SIMD8、SIMD16和SIMD32。在至少一 个实施例中，每个融合图形执行单元3109A-3109N包括至少两个执 行单元。例如，在至少一个实施例中，融合执行单元3109A包括第 一EU 3108A、第二EU 3108B以及第一EU 3108A和第二EU 3108B 共有的线程控制逻辑3107A。在至少一个实施例中，线程控制逻辑3107A控制在融合图形执行单元3109A上执行的线程，从而允许融 合执行单元3109A-3109N内的每个EU使用公共指令指针寄存器来执 行。

[0272] 在至少一个实施例中，一个或更多个内部指令高速缓存(例如 3106)被包括在线程执行逻辑3100中以高速缓存用于执行单元的线 程指令。在至少一个实施例中，包括一个或更多个数据高速缓存(例 如3112)以在线程执行期间高速缓存线程数据。在至少一个实施例 中，包括采样器3110以提供用于3D操作的纹理采样和用于媒体操 作的媒体采样。在至少一个实施例中，采样器3110包括专门的纹理 或媒体采样功能，以在将采样数据提供给执行单元之前在采样过程中 处理纹理或媒体数据。

[0273] 在执行期间，在至少一个实施例中，图形和媒体管线通过线程 产生和分派逻辑将线程发起请求发送到线程执行逻辑3100。在至少 一个实施例中，一旦一组几何对象已经被处理并光栅化成像素数据， 则在着色器处理器3102内的像素处理器逻辑(例如，像素着色器逻 辑、片段着色器逻辑等)被调用以进一步计算输出信息并且导致将结 果写入输出表面(例如，颜色缓冲区、深度缓冲区、模板缓冲区等)。 在至少一个实施例中，像素着色器或片段着色器计算要在光栅化对象 上插值的各种顶点属性的值。在至少一个实施例中，着色器处理器 3102内的像素处理器逻辑然后执行应用程序接口(API)提供的像素 或片段着色器程序。在至少一个实施例中，为了执行着色器程序，着 色器处理器3102经由线程分派器3104将线程分派到执行单元(例如 3108A)。在至少一个实施例中，着色器处理器3102使用采样器3110 中的纹理采样逻辑来访问存储在内存中的纹理贴图中的纹理数据。在 至少一个实施例中，对纹理数据和输入几何数据的算术运算为每个几 何片段计算像素颜色数据，或者丢弃一个或更多个像素以进行进一步 处理。

[0274] 在至少一个实施例中，数据端口3114提供了一种用于线程执 行逻辑3100的内存访问机制，以将处理后的数据输出到内存以在图 形处理器输出管线上进行进一步处理。在至少一个实施例中，数据端 口3114包括或耦合到一个或更多个高速缓存内存(例如，数据高速 缓存3112)以高速缓存数据以便经由数据端口进行内存访问。

[0275] 如图31B所示，在至少一个实施例中，图形执行单元3108可 以包括指令获取单元3137、通用寄存器文件阵列(GRF)3124、架 构寄存器文件阵列(ARF)3126、线程仲裁器3122、发送单元3130、 分支单元3132、一组SIMD浮点单元(FPU)3134，以及在至少一个 实施例中，一组专用整数SIMD ALU 3135。在至少一个实施例中， GRF 3124和ARF 3126包括一组与可以在图形执行单元3108中活跃 的每个同时硬件线程相关联的通用寄存器文件和架构寄存器文件。在 至少一个实施例中，在ARF 3126中维护每个线程架构状态，而在线 程执行期间使用的数据存储在GRF 3124中。在至少一个实施例中， 每个线程的执行状态，包括每个线程的指令指针，可以被保存在ARF 3126中的线程专用寄存器中。

[0276] 在至少一个实施例中，图形执行单元3108具有一种架构，该 架构是同时多线程(SMT)和细粒度交错多线程(IMT)的组合。在 至少一个实施例中，架构具有模块化配置，该

模块化配置可以在设计 时基于同时线程的目标数量和每个执行单元的寄存器数量来进行微 调,其中执行单元资源在用于执行多个同时线程的逻辑上分配。

[0277] 在至少一个实施例中,图形执行单元3108可以共同发布多个 指令,每个指令可以是不同的指令。在至少一个实施例中,图形执行 单元线程3108的线程仲裁器3122可以将指令分派到发送单元3130、分支单元3142或SIMD FPU 3134之一以供执行。在至少一个实施例 中,每个执行线程可以访问GRF 3124中的128个通用寄存器,其中 每个寄存器可以存储32个字节,可以作为32位数据元素的SIMD 8 元素向量进行访问。在至少一个实施例中,每个执行单元线程可以访 问GRF 3124中的4KB,尽管实施例不限于此,并且在其他实施例中可以提供更多或更少的寄存器资源。在至少一个实施例中,尽管每个 执行单元的线程数量也可以根据实施例而变化,但是最多可以同时执 行七个线程。在其中七个线程可以访问4KB的至少一个实施例中, GRF 3124可以存储总共28KB。在至少一个实施例中,灵活的寻址模式可以允许将寄存器一起寻址以有效地建立更宽的寄存器或表示 跨步的矩形块数据结构。

[0278] 在至少一个实施例中,经由由消息传递发送单元3130执行的 “发送”指令来调度内存操作、采样器操作和其他更长延迟的系统通 信。在至少一个实施例中,将分支指令分派到专用分支单元3132促 进SIMD发散和最终收敛。

[0279] 在至少一个实施例中,图形执行单元3108包括一个或更多个 SIMD浮点单元(FPU)3134,以执行浮点操作。在至少一个实施例 中,一个或更多个FPU 3134还支持整数计算。在至少一个实施例中, 一个或更多个FPU 3134可以SIMD执行多达M个32位浮点(或整 数)运算,或者SIMD执行多达2M个16位整数或16位浮点运算。 在至少一个实施例中,一个或更多个FPU中的至少一个提供扩展的 数学能力以支持高吞吐量的先验数学函数和双精度64位浮点。在至 少一个实施例中,还存在一组8位整数SIMD ALU 3135,并且可以 被专门优化以执行与机器学习计算相关的操作。

[0280] 在至少一个实施例中,可以在图形子核心分组(例如,子切片) 中实例化图形执行单元3108的多个实例的阵列。在至少一个实施例 中,执行单元3108可以跨多个执行通道执行指令。在至少一个实施 例中,在图形执行单元3108上执行的每个线程在不同的通道上执行。

[0281] 推理和/或训练逻辑1015用于执行与一个或更多个实施例相关 联的推理和/或训练操作。下面结合图10和/或图11提供关于推理和/ 或训练逻辑1015的细节。在至少一个实施例中,推理和/或训练逻辑 1015的部分或全部可以被结合到执行逻辑3100中。此外,在至少一 个实施例中,可以使用除了图10或图11中所示的逻辑之外的逻辑来 完成在此描述的推理和/或训练操作。在至少一个实施例中,权重参 数可以存储在片上或片外内存和/或寄存器(示出或未示出)中,其 配置执行逻辑3100的ALU以执行一种或更多种机器学习算法、神经 网络架构、用例或本文介绍的训练技术。

[0282] 图32示出了根据至少一个实施例的并行处理单元(“PPU”) 3200。在至少一个实施例中,PPU 3200配置有机器可读代码,该机 器可读代码如果由PPU 3200执行,则使得PPU 3200执行贯穿本公 开描述的一些或全部过程和技术。在至少一个实施例中,PPU 3200 是在一个或更多个集成电路设备上实现的多线程处理器,并且利用多 线程作为被设计为处理在多个线程上并行执行的计算机可读指令(也 称为机器可读指令或简单的指令)的延迟

隐藏技术。在至少一个实施 例中,线程是指执行线程,并且是被配置为由PPU 3200执行的一组 指令的实例。在至少一个实施例中,PPU 3200是图形处理单元 (“GPU”),图形处理单元配置为实现用于处理三维(“3D”) 图形数据的图形渲染管线,以便生成用于在显示设备(诸如液晶显示 器(“LCD”)设备)上显示的二维(“2D”)图像数据。在至少 一个实施例中,PPU 3200用于执行计算,诸如线性代数运算和机器 学习运算。图32仅出于说明性目的示出了示例并行处理器,并且应 被解释为在本公开的范围内设想的处理器架构的非限制性示例,并且 可以采用任何适当的处理器来对其进行补充和/或替代。

[0283] 在至少一个实施例中,一个或更多个PPU 3200配置成加速高 性能计算(“HPC”)、数据中心和机器学习应用程序。在至少一个 实施例中,PPU 3200配置成加速深度学习系统和应用程序,包括以 下非限制性示例:自动驾驶汽车平台、深度学习、高精度语音、图像、文本识别系统、智能视频分析、分子模拟、药物发现、疾病诊断、天 气预报、大数据分析、天文学、分子动力学模拟、财务建模、机器人 技术、工厂自动化、实时语言翻译、在线搜索优化以及个性化用户推 荐等。

[0284] 在至少一个实施例中,PPU 3200包括但不限于输入/输出 (“I/O”)单元3206、前端单元3210、调度器单元3212、工作分配 单元3214、集线器3216、交叉开关(“Xbar”)3220、一个或更多 个通用处理集群(“GPC”)3218和一个或更多个分区单元(“内 存分区单元”)3222。在至少一个实施例中,PPU 3200通过一个或 更多个高速GPU互连(“GPU互连”)3208连接到主机处理器或其 他PPU 3200。在至少一个实施例中,PPU 3200通过互连3202连接 到主机处理器或其他外围设备。在一实施例中,PPU 3200连接到包 括一个或更多个内存设备(“内存”)3204的本地内存。在至少一 个实施例中,内存设备3204包括但不限于一个或更多个动态随机存 取内存(“DRAM”)设备。在至少一个实施例中,一个或更多个 DRAM设备配置和/或可配置为高带宽内存(“HBM”)子系统,并 且在每个设备内堆叠有多个DRAM管芯。

[0285] 在至少一个实施例中,高速GPU互连3208可以指代系统使用 其来进行缩放的基于线的多通道通信链路,并包括与一个或更多个中 央处理单元结合的一个或更多个PPU 3200(“CPU”),支持PPU 3200 和CPU之间的缓存相干以及CPU主控。在至少一个实施例中,高速 GPU互连3208通过集线器3216将数据和/或命令传输到PPU 3200 的其他单元,例如一个或更多个复制引擎、视频编码器、视频解码器、电源管理单元和/或在图32中可能未明确示出的其他组件。

[0286] 在至少一个实施例中,I/O单元3206配置为通过系统总线3202 从主机处理器(图32中未示出)发送和接收通信(例如,命令、数 据)。在至少一个实施例中,I/O单元3206直接通过系统总线3202 或通过一个或更多个中间设备(例如内存桥)与主机处理器通信。在 至少一个实施例中,I/O单元3206可以经由系统总线3202与一个或 更多个其他处理器(例如一个或更多个PPU 3200)通信。在至少一 个实施例中,I/O单元3206实现外围组件互连Express(“PCIe”) 接口,用于通过PCIe总线进行通信。在至少一个实施例中,I/O单元 3206实现用于与外部设备通信的接口。

[0287] 在至少一个实施例中,I/O单元3206对经由系统总线3202接 收的分组进行解码。在至少一个实施例中,至少一些分组表示被配置 为使PPU 3200执行各种操作的命令。在至少一个实施例中,I/O单 元3206如命令所指定的那样将解码的命令发送到PPU 3200的各种其 他单元。在至少一个实施例中,命令被发送到前端单元3210和/或被 发送到集线器3216

或PPU 3200的其他单元，例如一个或更多个复制 引擎、视频编码器、视频解码器、电源管理单元等(图32中未明确 示出)。在至少一个实施例中，I/O单元3206配置为在PPU 3200的 各种逻辑单元之间路由通信。

[0288] 在至少一个实施例中，由主机处理器执行的程序在缓冲区中对 命令流进行编码，该缓冲区将工作负载提供给PPU 3200以进行处理。在至少一个实施例中，工作负载包括指令和要由那些指令处理的数 据。在至少一个实施例中，缓冲区是可由主机处理器和PPU 3200两 者访问(例如，读/写)的内存中的区域—主机接口单元可以配置为 访问经由I/O单元3206通过系统总线3202传输的内存请求连接到系 统总线3202的系统内存中的缓冲区。在至少一个实施例中，主机处 理器将命令流写入缓冲区，然后将指示命令流开始的指针发送给PPU 3200，使得前端单元3210接收指向一个或更多个命令流指针并管理 一个或更多个命令流，从命令流中读取命令并将命令转发到PPU 3200的各个单元。

[0289] 在至少一个实施例中，前端单元3210耦合到调度器单元3212， 该调度器单元3212配置各种GPC 3218以处理由一个或更多个命令 流定义的任务。在至少一个实施例中，调度器单元3212配置为跟踪 与调度器单元3212管理的各种任务有关的状态信息，其中状态信息 可以指示任务被分配给哪个GPC 3218，任务是活跃的还是非活跃的， 与任务相关联的优先级等等。在至少一个实施例中，调度器单元3212 管理在一个或更多个GPC 3218上执行的多个任务。

[0290] 在至少一个实施例中，调度器单元3212耦合到工作分配单元 3214，该工作分配单元3214配置为分派任务以在GPC 3218上执行。 在至少一个实施例中，工作分配单元3214跟踪从调度器单元3212接 收到的多个调度任务并且工作分配单元3214管理每个GPC 3218的待处理任务池和活跃任务池。在至少一个实施例中，待处理任务池包 括多个时隙(例如32个时隙)，这些时隙包含分配给要由特定的GPC 3218处理的任务；活跃任务池可包括用于由GPC 3218主动处理的任 务的多个时隙(例如4个时隙)，以使随着GPC 3218中的一个完成任务的执行，该任务将从GPC 3218的活动任务池中逐出，并且从待 处理任务池中选择其他任务之一，并安排其在GPC 3218上执行。在 至少一个实施例中，如果活跃任务在GPC 3218上处于空闲状态，例 如在等待数据依赖性解决时，则活跃任务从GPC 3218中驱逐并返回到待处理任务池，同时选择了待处理任务池中的另一个任务并调度在 GPC 3218上执行。

[0291] 在至少一个实施例中，工作分配单元3214经由XBar3220与 一个或更多个GPC 3218通信。在至少一个实施例中，XBar3220是互 连网络，其将PPU 3200的许多单元耦合到PPU 3200的其他单元， 并且可以配置为将工作分配单元3214耦合到特定的GPC3218。在至少一个实施例中，一个或更多个PPU 3200的其他单元也可以通过集 线器3216连接到XBar3220。

[0292] 在至少一个实施例中，任务由调度器单元3212管理，并由工 作分配单元3214分配给GPC 3218之一。GPC 3218配置为处理任务 并产生结果。在至少一个实施例中，结果可以由GPC 3218中的其他 任务消耗，通过XBar3220路由到不同的GPC 3218或存储在内存3204中。在至少一个实施例中，结果可以通过分区单元3222写到内存3204 中，其实现了用于向内存3204写入数据或从内存3204读取数据的内 存接口。在至少一个实施例中，结果可以经由高速GPU互连3208传 输到另一PPU 3204或CPU。在至少一个实施例中，PPU 3200包括但不限于U个分区单元3222，其等于耦合到PPU 3200的分离且不同的 存储设备3204的数量。

在至少一个实施例中,下面结合图34更详细 地描述分区单元3222。

[0293] 在至少一个实施例中,主机处理器执行驱动器核心,该驱动程 序核心实现应用程序编程接口(API),该应用程序编程接口使在主 机处理器上执行的一个或更多个应用程序能够调度操作以在PPU 3200上执行。在一个实施例中,多个计算应用由PPU 3200同时执行,并且PPU 3200为多个计算应用程序提供隔离、服务质量("QoS") 和独立的地址空间。在至少一个实施例中,应用程序生成指令(例如, 以API调用的形式),该指令使驱动器核心生成一个或更多个任务以 供PPU 3200执行,并且驱动器核心将任务输出至由PPU 3200处理 的一个或更多个流。在至少一个实施例中,每个任务包括一个或更多 个相关线程组,其可以被称为线程束(warp)。在至少一个实施例中, 线程束包括可以并行执行的多个相关线程(例如32个线程)。在至 少一个实施例中,协作线程可以指代多个线程,包括用于执行任务并 且通过共享内存交换数据的指令。在至少一个实施例中,结合图34 根据至少一个实施例更详细地描述了线程和协作线程。

[0294] 推理和/或训练逻辑1015用于执行与一个或更多个实施例相关 联的推理和/或训练操作。下面结合图10和/或图11提供关于推理和/ 或训练逻辑1015的细节。在至少一个实施例中,深度学习应用处理 器用于训练机器学习模型(诸如神经网络),以预测或推理提供给 PPU 3200的信息。在至少一个实施例中,深度学习应用处理器3200 用于基于已由另一处理器或系统或PPU 3200训练过的训练过的机器 学习模型(例如,神经网络)推理或预测信息。在至少一个实施例中, PPU 3200可用于执行本文所述的一个或更多个神经网络用例。

[0295] 图33示出了根据至少一个实施例的通用处理集群("GPC") 3300。在至少一个实施例中,GPC 3300是图32的GPC 3218。在至 少一个实施例中,每个GPC 3300包括但不限于用于处理任务的多个 硬件单元,并且每个GPC 3300包括但不限于管线管理器3302、预光 栅操作单元("PROP")3304、光栅引擎3308、工作分配交叉开关 ("WDX")3316、内存管理单元("MMU")3318、一个或更多 个数据处理集群("DPC")3306,以及部件的任何合适组合。

[0296] 在至少一个实施例中,GPC 3300的操作由管线管理器3302控 制。在至少一个实施例中,管线管理器3302管理一个或更多个DPC 3306的配置,以处理分配给GPC 3300的任务。在至少一个实施例中, 管线管理器3302配置一个或更多个DPC 3306中的至少一个以实现图形渲染管线的至少一部分。在至少一个实施例中,DPC 3306配置 为在可编程流式多处理器("SM")3314上执行顶点着色器程序。 在至少一个实施例中,管线管理器3302配置为将从工作分配单元接 收的数据包路由到GPC 3300内的适当逻辑单元,以及在至少一个实 施例中,可以将一些数据包路由到PROP 3304和/或光栅引擎3308中 的固定功能硬件单元,而可以将其他数据包路由到DPC 3306以由原 始引擎3312或SM3314进行处理。在至少一个实施例中,管线管理 器3302配置DPC 3306中的至少一个以实现神经网络模型和/或计算 管线。

[0297] 在至少一个实施例中,PROP单元3304配置为在至少一个实 施例中将由光栅引擎3308和DPC 3306生成的数据路由到分区单元 3222中的光栅操作("ROP")单元,上面结合图32更详细地描述。 在至少一个实施例中,PROP单元3304配置为执行用于颜色混合的 优化、组织像素数据、执行地址转换等等。在至少一个实施例中,光 栅引擎3308包括但不限于配置为执行各种光栅操作的多个固定功能 硬件单元,并且在至少一个实施例中,光栅引擎3308包括但不限于 设置引擎、粗光栅引擎、剔除引擎、裁剪引擎、精细光栅引擎、图块 聚合

引擎及其任意合适的组合。在至少一个实施例中,设置引擎接收 变换后的顶点并生成与由顶点定义的几何图元相关联的平面方程;平 面方程式被传送到粗光栅引擎以生成基本图元的覆盖信息(例如,图 块的x、y覆盖范围掩码);粗光栅引擎的输出将传输到剔除引擎,在剔除引擎中与z测试失败的图元相关联的片段将被剔除,并传输到 剪切引擎,在剪切引擎中剪切位于视锥范围之外的片段。在至少一个 实施例中,将经过裁剪和剔除的片段传递给精细光栅引擎,以基于设 置引擎生成的平面方程式生成像素片段的属性。在至少一个实施例 中,光栅引擎3308的输出包括将由任何适当的实体(例如,由在DPC 3306内实现的片段着色器)处理的片段。

[0298] 在至少一个实施例中,包括在GPC 3300中的每个DPC 3306 包括但不限于M管线控制器(“MPC”)3310;图元引擎3312;一 个或更多个SM 3314;及其任何合适的组合。在至少一个实施例中, MPC 3310控制DPC 3306的操作,将从管线管理器3302接收的分组 路由到DPC 3306中的适当单元。在至少一个实施例中,将与顶点相 关联的分组路由到图元引擎3312,图元引擎3312配置为从内存中获 取与顶点关联的顶点属性;相反,可以将与着色器程序相关联的数据 包发送到SM 3314。

[0299] 在至少一个实施例中,SM 3314包括但不限于可编程流式处理 器,其配置为处理由多个线程表示的任务。在至少一个实施例中,SM 3314是多线程的并且配置为同时执行来自特定线程组的多个线程(例 如32个线程),并且实现单指令、多数据(“SIMD”)架构,其中将一组线程(例如,线程束)中的每个线程配置为基于相同的指令集 来处理不同的数据集。在至少一个实施例中,线程组中的所有线程执 行相同的指令。在至少一个实施例中,SM 3314实施单指令、多线程 (“SIMT”)架构,其中一组线程中的每个线程配置为基于相同的指令集来处理不同的数据集,但是其中线程组中的各个线程允许在执 行期间发散。在至少一个实施例中,为每个线程束维护程序计数器、调用栈和执行状态,从而当线程束中的线程发散时,实现线程束和线 程束内的串行执行之间的并发性。在另一个实施例中,为每个单独的 线程维护程序计数器、调用栈和执行状态,从而使得在线程束内和线 程束之间的所有线程之间具有相等的并发性。在至少一个实施例中, 为每个单独的线程维持执行状态,并且可以收敛并并行地执行执行相 同指令的线程以提高效率。下面更详细地描述SM 3314的至少一个 实施例。

[0300] 在至少一个实施例中,MMU 3318在GPC 3300和内存分区单 元(例如,图32的分区单元3222)之间提供接口,并且MMU 3318 提供虚拟地址到物理地址的转换、内存保护以及内存请求的仲裁。在 至少一个实施例中,MMU 3318提供一个或更多个转换后备缓冲区(“TLB”),用于执行虚拟地址到内存中的物理地址的转换。

[0301] 推理和/或训练逻辑1015用于执行与一个或更多个实施例相关 联的推理和/或训练操作。下面结合图10和/或图11提供关于推理和/ 或训练逻辑1015的细节。在至少一个实施例中,深度学习应用处理 器用于训练机器学习模型(诸如神经网络),以预测或推理提供给 GPC 3300的信息。在至少一个实施例中,GPC 3300用于基于已由另 一处理器或系统或GPC 3300训练过的机器学习模型(例如,神经网 络)推理或预测信息。在至少一个实施例中,GPC 3300可用于执行 本文所述的一个或更多个神经网络用例。

[0302] 图34示出了根据至少一个实施例的并行处理单元(“PPU”) 的内存分区单元3400。在至少一个实施例中,内存分区单元3400包 括但不限于光栅操作(“ROP”)单元3402;二级

(“L2”)高速 缓存3404;内存接口3406;及其任何合适的组合。内存接口3406耦 合到内存。内存接口3406可以实现32、64、128、1024位数据总线，或者类似的实现方式用于高速数据传输。在至少一个实施例中,PPU 包括U个内存接口3406,每对分区单元3400一个内存接口3406，其中每对分区单元3400连接到对应的内存设备。例如,在至少一个 实施例中,PPU可以连接至多达Y个内存设备,例如高带宽内存堆 栈或图形双数据速率版本5同步动态随机存取内存(“GDDR5 SDRAM”)。

[0303] 在至少一个实施例中,内存接口3406实现高带宽内存第二代 (“HBM2”)内存接口,并且Y等于U的一半。在至少一个实施 例中,HBM2内存堆栈与PPU位于同一物理封装上,与传统的GDDR5 SDRAM系统相比,可提供大量功率并节省面积。在至少一个实施例 中,每个HBM2堆栈包括但不限于四个内存管芯,且Y等于4,每 个HBM2堆栈包括每个管芯两个128位通道,用于总共8个通道和 1024位的数据总线宽度。在至少一个实施例中,内存支持单错误校 正双错误检测(“SECDED”)错误校正码(“ECC”)以保护数据。ECC为对数据损坏敏感的计算应用提供更高的可靠性。

[0304] 在至少一个实施例中,PPU实现了多级内存层次结构。在至少 一个实施例中,内存分区单元3400支持统一内存以为中央处理单元 (“CPU”)和PPU内存提供单个统一虚拟地址空间,从而实现虚 拟内存系统之间的数据共享。在至少一个实施例中,追踪PPU对位 于其他处理器上的内存的访问频率,以确保将内存页面移动到更频繁 地访问页面的PPU的物理内存。在至少一个实施例中,高速GPU互 连3208支持地址转换服务,其允许PPU直接访问CPU的页表,并 通过PPU提供对CPU内存的完全访问。

[0305] 在至少一个实施例中,复制引擎在多个PPU之间或PPU与 CPU之间传输数据。在至少一个实施例中,复制引擎可以为未被映 射到页表中的地址生成页面错误,并且内存分区单元3400然后为页 面错误提供服务,将地址映射到页表中,之后复制引擎执行传输。在 至少一个实施例中,为多个处理器之间的多个复制引擎操作固定(即 不可分页)内存,从而实质上减少了可用内存。在至少一个实施例中， 在硬件页面故障的情况下,可以将地址传递给复制引擎,而无需考虑 是否驻留内存页面,并且复制过程是透明的。

[0306] 根据至少一个实施例,来自图32的内存3204或其他系统内存 的数据由内存分区单元3400获取,并将其存储在L2高速缓存3404 中,L2高速缓存3404位于芯片上并且在各种GPC之间共享。在至 少一个实施例中,每个内存分区单元3400包括但不限于与对应的内 存设备相关联的L2高速缓存的至少一部分。在至少一个实施例中， 在GPC内的各个单元中实现较低级别的高速缓存。在至少一个实施 例中,每个SM 3314可以实现一级(“L1”)高速缓存,其中L1高 速缓存是专用于特定SM 3314的私有内存,并且从L2高速缓存3404 中获取数据并将其存储在每个L1高速缓存中,用于在SM 3314的功 能单元中进行处理。在至少一个实施例中,L2高速缓存3404耦合到 内存接口3406和XBar 3220。

[0307] 在至少一个实施例中,ROP单元3402执行与像素颜色有关的 图形光栅操作,诸如颜色压缩、像素混合等。在至少一个实施例中， ROP单元3402结合光栅引擎3308实施深度测试,从光栅引擎3308 的剔除引擎接收与像素片段相关联的样本位置的深度。在至少一个实施例中,针对在与片段关联的样本位置的深度缓冲区中的相应深度测 试深度。在至少一个实施例中,如果片段通过了针对样本位置的深度 测试,则ROP单元3402更新深度缓冲区,并将深度测试的结果发送 给光栅引擎3308。将意识到,分区单元3400的数量可以不同于GPC

的数量,因此,可以在至少一个实施例中将每个ROP单元3402耦合 到每个GPC。在至少一个实施例中,ROP单元3402追踪从不同GPC 接收到的分组,并确定通过XBar3220将ROP单元3402产生的结果 路由到哪个。

[0308] 图35示出了根据至少一个实施例的流式多处理器(“SM”) 3500。在至少一个实施例中,SM 3500是图33的SM。在至少一个实 施例中,SM 3500包括但不限于指令高速缓存3502;一个或更多个 调度器单元3504;寄存器文件3508;一个或更多个处理核心(“核 心”)3510;一个或更多个特殊功能单元(“SFU”)3512;一个或 更多个加载/存储单元(“LSU”)3514;互连网络3516;共享内存/ 一级(“L1”)高速缓存3518;及其任何合适的组合。在至少一个 实施例中,工作分配单元调度任务以在并行处理单元(“PPU”)的 通用处理集群(“GPC”)上执行,并且每个任务被分配给GPC内 部的特定数据处理集群(“DPC”),并且如果任务与着色器程序相 关联,则将任务分配给SM 3500之一。在至少一个实施例中,调度 器单元3504从工作分配单元接收任务并管理分配给SM 3500的一个 或更多个线程块的指令调度。在至少一个实施例中,调度器单元3504 调度线程块以作为并行线程的线程束来执行,其中,每个线程块被分 配至少一个线程束。在至少一个实施例中,每个线程束执行线程。在 至少一个实施例中,调度器单元3504管理多个不同的线程块,将线 程束分配给不同的线程块,然后在每个时钟周期内将来自多个不同的 协作组的指令分派给各种功能单元(例如,处理核心3510、SFU 3512 和LSU 3514)。

[0309] 在至少一个实施例中,合作组可以指用于组织通信线程组的编 程模型,其允许开发人员表达线程正在通信的粒度,从而能够表达更 丰富、更有效的并行分解。在至少一个实施例中,协作启动API支持 线程块之间的同步以执行并行算法。在至少一个实施例中,常规编程 模型的应用提供了用于同步协作线程的单一、简单的构造:跨线程块 的所有线程的屏障(例如,syncthreads()函数)。但是,在至少一 个实施例中,程序员可以在小于线程块粒度的情形下来定义线程组, 并在所定义的组内进行同步,以实现更高的性能、设计灵活性以及以 集合组范围功能接口的形式实现软件重用。在至少一个实施例中,协 作组使程序员能够以子块(即,小到单个线程)和多块粒度明确定义 线程组,并执行集合操作,例如对协作组中的线程进行同步。编程模 型支持跨软件边界的干净组合,从而库和实用程序功能可以在其本地 环境中安全地同步,而不必进行关于收敛的假设。在至少一个实施例中,协作组图元使协作并行的新图案成为可能,包括但不限于生产者 -消费者并行,机会主义并行以及整个线程块网格上的全局同步。

[0310] 在至少一个实施例中,分派单元3506配置为将指令发送到功 能单元中的一个或更多个,并且调度器单元3504包括但不限于两个 分派单元3506,该两个分派单元3506使得来自相同线程束的两个不 同指令能够在每个时钟周期被分派。在至少一个实施例中,每个调度 器单元3504包括单个分派单元3506或附加分派单元3506。

[0311] 在至少一个实施例中,每个SM 3500在至少一个实施例中包 括但不限于寄存器文件3508,该寄存器文件3508为SM 3500的功能 单元提供了一组寄存器。在至少一个实施例中,寄存器文件3508在 每个功能单元之间划分,从而为每个功能单元分配寄存器文件3508的专用部分。在至少一个实施例中,寄存器文件3508在由SM 3500 执行的不同线程束之间划分,并且寄存器文件3508为连接到功能单 元的数据路径的操作数提供临时存储。在至少一个实施例中,每个 SM 3500包括但不限于多个L个处理核心3510。在至少一个实施例 中,

SM 3500包括但不限于大量(例如128个或更多)不同的处理核 心3510。在至少一个实施例中,每个处理核心3510在至少一个实施 例中包括但不限于全管线、单精度、双精度和/或混合精度处理单元，其包括但不限于浮点算术逻辑单元和整数算术逻辑单元。在至少一个实施例中,浮点算术逻辑单元实现用于浮点算术的IEEE 754-2008标 准。在至少一个实施例中,处理核心3510包括但不限于64个单精度 (32位)浮点核心、64个整数核心、32个双精度(64位)浮点核心 和8个张量核心。

[0312] 根据至少一个实施例,张量核心配置为执行矩阵运算。在至少 一个实施例中,一个或更多个张量核心包括在处理核心3510中。在 至少一个实施例中,张量核心配置为执行深度学习矩阵算术,例如用 于神经网络训练和推理的卷积运算。在至少一个实施例中,每个张量 核心在4×4矩阵上操作并且执行矩阵乘法和累加运算D＝A×B+ C,其中A、B、C和D是4×4矩阵。

[0313] 在至少一个实施例中,矩阵乘法输入A和B是16位浮点矩阵，并且累加矩阵C和D是16位浮点或32位浮点矩阵。在至少一个实 施例中,张量核心对16位浮点输入数据进行32位浮点累加运算。在 至少一个实施例中,16位浮点乘法使用64个运算,并得到全精度乘 积,然后使用32位浮点加法与其他中间乘积累加起来,以进行4x4x4 矩阵乘法。在至少一个实施例中,张量核心用于执行由这些较小的元 件构成的更大的二维或更高维度的矩阵运算。在至少一个实施例中,, API(诸如CUDA 9C++API)公开专门的矩阵加载、矩阵乘法和累 加以及矩阵存储操作,以有效地使用来自CUDA-C++程序的张量核 心。在至少一个实施例中,在CUDA级别,线程束级别接口假定跨 越所有32个线程束线程的16×16大小的矩阵。

[0314] 在至少一个实施例中,每个SM 3500包括但不限于执行特殊 功能(例如,属性评估、倒数平方根等)的M个SFU 3512。在至少 一个实施例中,SFU 3512包括但不限于配置为遍历分层树数据结构 的树遍历单元。在至少一个实施例中,SFU 3512包括但不限于配置为执行纹理映射过滤操作的纹理单元。在至少一个实施例中,纹理单 元配置为从内存中加载纹理映射(例如,纹理像素的2D阵列)和采 样纹理映射,以产生采样的纹理值以供由SM 3500执行的着色器程 序使用。在至少一个实施例中,将纹理映射存储在共享内存/L1高速缓存3518中。在至少一个实施例中,根据至少一个实施例,纹理单 元使用mip映射(mip-maps)(例如,细节级别不同的纹理映射)来 实现纹理操作(诸如过滤操作)。在至少一个实施例中,每个SM 3500 包括但不限于两个纹理单元。

[0315] 在至少一个实施例中,每个SM 3500包括但不限于实现共享 内存/L1高速缓存3518与寄存器文件3508之间的加载和存储操作的 N个LSU 3514。在至少一个实施例中,每个SM 3500包括但不限于 互连网络3516,互连网络3516将每个功能单元连接到寄存器文件3508,并且LSU 3514连接到寄存器文件3508和共享内存/L1高速缓 存3518。在至少一个实施例中,互连网络3516是交叉开关,其可以 配置为将任何功能单元连接到寄存器文件3508中的任何寄存器,并 且将LSU 3514连接到寄存器文件3508和共享内存/L1高速缓存3518中的内存位置。

[0316] 在至少一个实施例中,共享内存/L1高速缓存3518是片上内 存的阵列,其在至少一个实施例中允许SM 3500与图元引擎之间以 及SM 3500中的线程之间的数据存储和通信。在至少一个实施例中， 共享内存/L1高速缓存3518包括但不限于128KB的存储容量,并且 位于从SM 3500到分区单元的路径中。在至少一个实施例中,共享 内存/L1高速缓存

3518在至少一个实施例中用于高速缓存读取和写 入。在至少一个实施例中,共享内存/L1高速缓存3518、L2高速缓 存和内存中的一个或更多个是后备存储。

[0317] 在至少一个实施例中,将数据高速缓存和共享内存功能组合到 单个内存块中,为两种类型的内存访问提供了改进的性能。在至少一 个实施例中,容量由不使用共享内存的程序使用或将其用作高速缓 存,例如如果共享内存配置为使用一半容量,则纹理和加载/存储操 作可以使用剩余容量。根据至少一个实施例,在共享内存/L1高速缓 存3518内的集成使共享内存/L1高速缓存3518能够用作用于流传输 数据的高吞吐量管线,同时提供对频繁重用的数据的高带宽和低延迟 访问。在至少一个实施例中,当配置用于通用并行计算时,与图形处 理相比,可以使用更简单的配置。在至少一个实施例中,绕过固定功 能图形处理单元,从而创建了更加简单的编程模型。在至少一个实施 例中,在通用并行计算配置中,工作分配单元直接将线程的块分配和 分布给DPC。在至少一个实施例中,块中的线程执行相同的程序, 在计算中使用唯一的线程ID以确保每个线程生成唯一的结果,使用 SM 3500执行程序并执行计算,使用共享内存/L1高速缓存3518在 线程之间进行通信,以及使用LSU 3514通过共享内存/L1高速缓存 3518和内存分区单元来读写全局内存。在至少一个实施例中,当被 配置用于通用并行计算时,SM 3500向调度器单元3504写入可以用 来在DPC上启动新工作的命令。

[0318] 在至少一个实施例中,PPU被包括在台式计算机、膝上型计算 机、平板电脑、服务器、超级计算机、智能电话(例如,无线、手持 设备)、个人数字助理(“PDA”)、数码相机、车辆、头戴式显示 器、手持式电子设备等中或与之耦合。在至少一个实施例中,PPU被 实现在单个半导体衬底上。在至少一个实施例中,PPU与一个或更多 个其他设备(例如附加的PPU、内存、精简指令集计算机(“RISC”) CPU,一个或更多个内存管理单元(“MMU”)、数模转换器(“DAC”) 等)一起被包括在片上系统(“SoC”)中。

[0319] 在至少一个实施例中,PPU可以被包括在包括一个或更多个存 储设备的图形卡上。图形卡可以配置为与台式计算机主板上的PCIe 插槽相连接。在至少一个实施例中,PPU可以是包括在主板的芯片组 中的集成图形处理单元(“iGPU”)。

[0320] 推理和/或训练逻辑1015用于执行与一个或更多个实施例相关 的推理和/或训练操作。下面结合图10和/或图11提供关于推理和/或 训练逻辑1015的细节。在至少一个实施例中,深度学习应用处理器 用于训练机器学习模型(诸如神经网络),以预测或推理提供给SM 3500的信息。在至少一个实施例中,SM 3500用于基于已由另一处 理器或系统或由SM 3500训练过的机器学习模型(例如,神经网络) 推理或预测信息。在至少一个实施例中,SM 3500可用于执行一个或 更多个本文所述的神经网络用例。

[0321] 在至少一个实施例中,单个半导体平台可以指唯一的单一的基 于半导体的集成电路或芯片。在至少一个实施例中,可以使用具有增 加的连接性的多芯片模块,其模拟芯片上的操作,并且相对于利用传 统的中央处理单元(“CPU”)和总线实施方式进行了实质性的改进。 在至少一个实施例中,根据用户的需求,各种模块也可以分开放置或 以半导体平台的各种组合放置。

[0322] 其他变型在本公开的精神内。因此,尽管公开的技术易于进行 各种修改和替代构造,但是其某些示出的实施例在附图中示出并且已 经在上面进行了详细描述。但是,应当理解,无意将公开内容限制为 所公开的一种或更多种特定形式,而是相反,其意图是涵盖

落入如所 附权利要求书所定义的公开内容的精神和范围内的所有修改、替代构 造和等同物。

[0323] 除非另有说明,除非另有说明或显然与上下文明显矛盾,否则 在描述所公开的实施例的环境中(特别是在所附权利要求的环境中) 对术语“一”,“一个”和“该”以及类似指代的使用应解释为涵盖 单数和复数。术语“包含”,“具有”,“包括”和“内含”应解释 为开放式术语(意思是“包括但不限于”)。术语“连接”在未经修 改时指的是物理连接,应理解为部分或全部包含在,连接到或连接在 一起的部分或全部,即使有任何介入。除非在此另外指出,否则本文 中数值范围的引用仅旨在用作分别指代落入该范围内的每个单独值 的速记方法,并且每个单独值都被并入说明书中,就如同其在本文中 被单独叙述一样。除非环境另外指出或矛盾,否则术语“组”(例如 “一组项目”)或“子集”的使用应解释为包括一个或更多个成员的 非空集合。此外,除非环境另外指出或矛盾,否则相应集合的术语“子 集”不一定表示相应集合的适当子集,而是子集和相应集合可以相等。

[0324] 除非以其他方式明确指出或与环境明显矛盾,否则诸如“A、B和C中的至少一个”或“A、B和C的至少一个”形式的词组等 联合语言在环境中理解为通常用来表示项目,术语等可以是A或B 或C,也可以是A和B和C集合的任何非空子集。例如,在具有三 个成员,连接短语“A、B和C中的至少一个”和“A、B和C的 至少一个”是指以下任意集合:{A},{B},{C},{A,B},{A,C}, {B,C},{A,B,C}。因此,这种联合语言通常不意图暗示某些实 施例要求存在A中的至少一个、B中的至少一个和C中的至少一个。 另外,除非另有说明或与环境矛盾,否则术语“多个”表示复数的状 态(例如,“多个项目”表示多个项目)。复数中项目的数量是至少两个,但是当明确地或通过环境指示时可以是多个。此外,除非另有 说明或从环境中清楚得知,否则短语“基于”是指“至少部分基于” 而不是“仅基于”。

[0325] 除非本文另外指出或与环境明显矛盾,否则本文描述的过程的 操作可以任何合适的序列执行。在至少一个实施例中,诸如本文所述 的那些过程(或其变形和/或其组合)的过程在配置有可执行指令的 一个或更多个计算机系统的控制下执行,并且被实现为代码(例如, 可执行指令、一个或更多个计算机程序或一个或更多个应用程序)通 过硬件或其组合在一个或更多个处理器上共同执行。在至少一个实施 例中,代码例如以计算机程序的形式存储在计算机可读存储介质上, 该计算机程序包括可由一个或更多个处理器执行的多个指令。在至少 一个实施例中,计算机可读存储介质是非暂时性计算机可读存储介质,其不包括暂时性信号(例如,传播的瞬态电或电磁传输),但包 括暂时性信号的收发器中的非暂时性数据存储电路(例如,缓冲区、高速缓存和队列)。在至少一个实施例中,代码(例如,可执行代码 或源代码)被存储在其上存储有可执行指令的一组一个或更多个非暂时性计算机可读存储介质(或用于存储可执行指令的其他内存)上, 当由计算机系统的一个或更多个处理器执行(即,由于被执行)而导 致的计算机系统执行本文所述的操作。在至少一个实施例中,一组非 暂时性计算机可读存储介质包括多个非暂时性计算机可读存储介质 以及缺少所有代码的多个非暂时性计算机可读存储介质的一个或更 多个单个非暂时性存储介质,而多个非暂时性计算机可读存储介质共 同存储所有代码。在至少一个实施例中,执行可执行指令,使得不同 的指令由不同的处理器执行,例如,非暂时性计算机可读存储介质存 储指令,并且主中央处理单元(“CPU”)执行一些指令,而图形处 理单元(“GPU”)执行其他指令。在至少一个实施例中,计算机系 统的不同组件具有单独的处理器,

并且不同的处理器执行指令的不同 子集。

[0326] 因此,在至少一个实施例中,计算机系统被配置为实现单独地 或共同地执行本文所描述的过程的操作的一个或更多个服务,并且这 样的计算机系统配置有能够实现操作的适用的硬件和/或软件。此外, 实现本公开的至少一个实施例的计算机系统是单个设备,并且在另一 实施例中,是一种分布式计算机系统,其包括以不同方式操作的多个 设备,使得分布式计算机系统执行本文所述的操作,并且使得单个设 备不执行所有操作。

[0327] 本文提供的任何和所有示例或示例性语言(例如,“诸如”) 的使用仅旨在更好地阐明本公开的实施例,并且不对公开的范围构成 限制,除非另有声明。说明书中的任何语言都不应被解释为表示对于 实施公开必不可少的任何未要求保护的要素。

[0328] 本文引用的所有参考文献,包括出版物、专利申请和专利,均 以引用的方式并入本文,如同每个参考文献被单独且具体地指出以引 用的方式并入本文一样。

[0329] 在描述和权利要求中,可以使用术语“耦合”和“连接”及其 派生词。应当理解,这些术语可能不旨在作为彼此的同义词。相反, 在特定示例中,“连接”或“耦合”可用于指示两个或更多个元件彼 此直接或间接物理或电接触。“耦合”也可能意味着两个或更多个元素彼此不直接接触,但仍彼此协作或交互。

[0330] 除非另外特别说明,否则可以理解,在整个说明书中,诸如“处 理”、“计算”、“运算”、“确定”等,是指计算机或计算系统的 动作和/或过程。或类似的电子计算设备,将计算系统的寄存器和/或 内存中表示为物理量(例如电子)的数据处理和/或转换为类似表示为计算系统的内存、寄存器或其他此类信息存储、传输或显示设备中 的物理量的其他数据。

[0331] 以类似的方式,术语“处理器”可以指处理来自寄存器和/或 内存的电子数据并将该电子数据转换成可以存储在寄存器和/或内存 中的其他电子数据的任何设备或设备的一部分。作为非限制性示例, “处理器”可以是CPU或GPU。“计算平台”可以包括一个或更多个处理器。如本文所使用的,“软件”过程可以包括例如随时间执行 工作的软件和/或硬件实体,诸如任务、线程和智能代理。而且,每 个过程可以指代多个过程,以连续地或间歇地序列地或并行地执行指 令。因为系统可以体现一种或更多种方法并且方法可以被认为是系 统,术语“系统”和“方法”在本文中可互换使用。

[0332] 在本文件中,可以参考获得、获取、接收或将模拟或数字数据 输入子系统、计算机系统或计算机实现的机器中。可以以多种方式来 完成获得、获取、接收或输入模拟和数字数据的过程,例如通过接收 作为函数调用或对应用程序接口的调用的参数的数据。在一些实施方 式中,获得、获取、接收或输入模拟或数字数据的过程可以通过经由 串行或并行接口传输数据来完成。在另一实施方式中,可以通过经由 计算机网络将数据从提供实体传输到获取实体来完成获得、获取、接 收或输入模拟或数字数据的过程。也可以参考提供、输出、传输、发 送或呈现模拟或数字数据。在各种示例中,提供、输出、传输、发送 或呈现模拟或数字数据的过程可以通过将数据作为函数调用的输入 或输出参数,应用程序编程接口或进程间通信机制的参数进行传输来 实现。

[0333] 尽管上面的讨论阐述了所描述的技术的示例实现,但是其他架 构可以用于实现所描述的功能,并且意图在本公开的范围内。此外, 尽管出于讨论目的在上面定义了具体的责任分配,但是根据情况,可 以以不同的方式分配和划分各种功能和职责。

[0334] 此外,尽管已经用针对结构特征和/或方法动作的语言描述了 主题,但是应该理解,所附权利要求书所要求保护的主题不必限于所 描述的特定特征或动作。而是,公开了特定的特征和动作作为实现权 利要求的示例性形式。

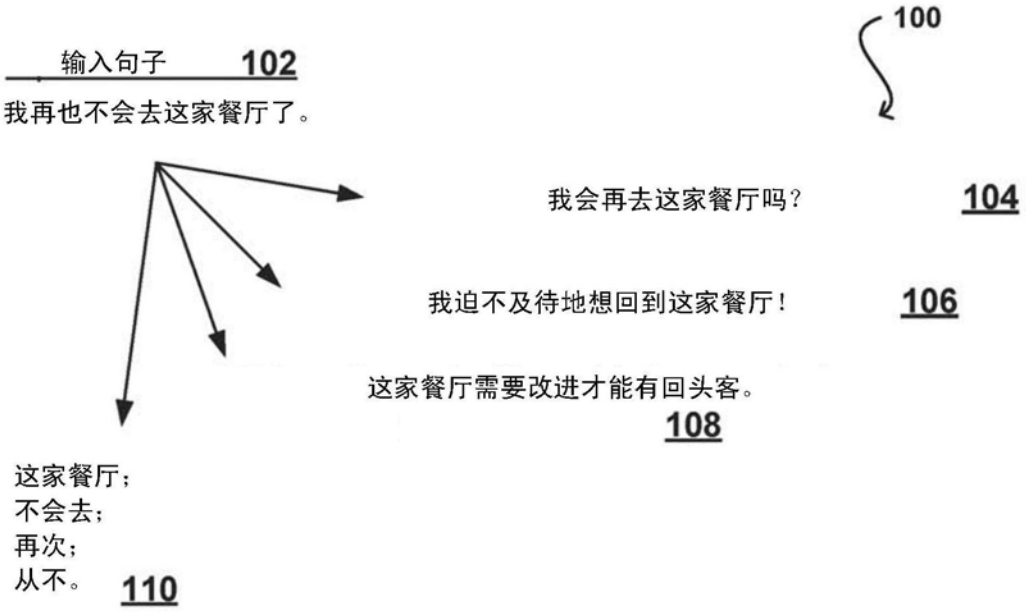

100
输入句子 102
我再也不会去这家餐厅了。
我会再去这家餐厅吗? 104
我迫不及待地想回到这家餐厅! 106
这家餐厅需要改进才能有回头客。
108
这家餐厅;
不会去;
再次;
从不。 110

图1

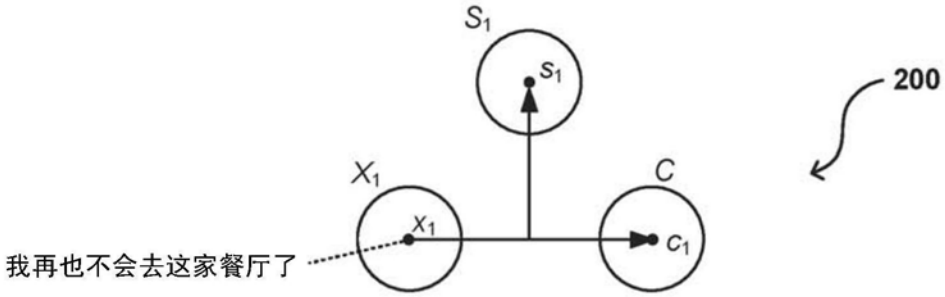

200
S_1
s_1
X_1
x_1
C
c_1
我再也不会去这家餐厅了

图2A

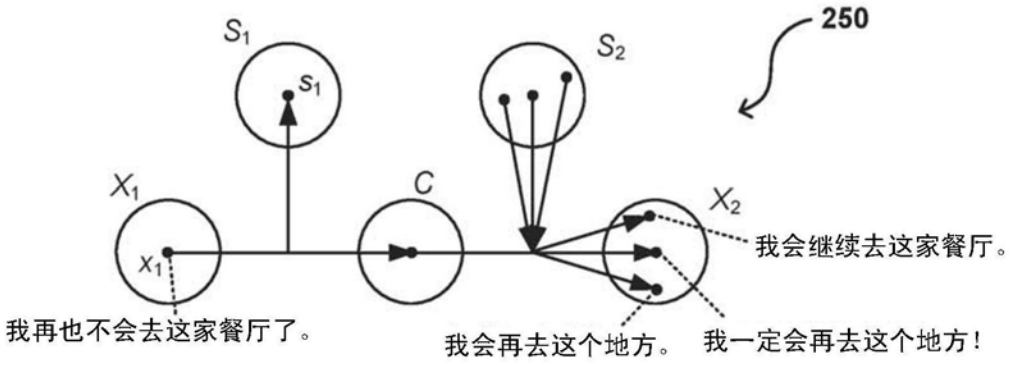

250
S_1
s_1
S_2
X_1
x_1
C
X_2
我会继续去这家餐厅。
我再也不会去这家餐厅了。
我会再去这个地方。
我一定会再去这个地方!

图2B

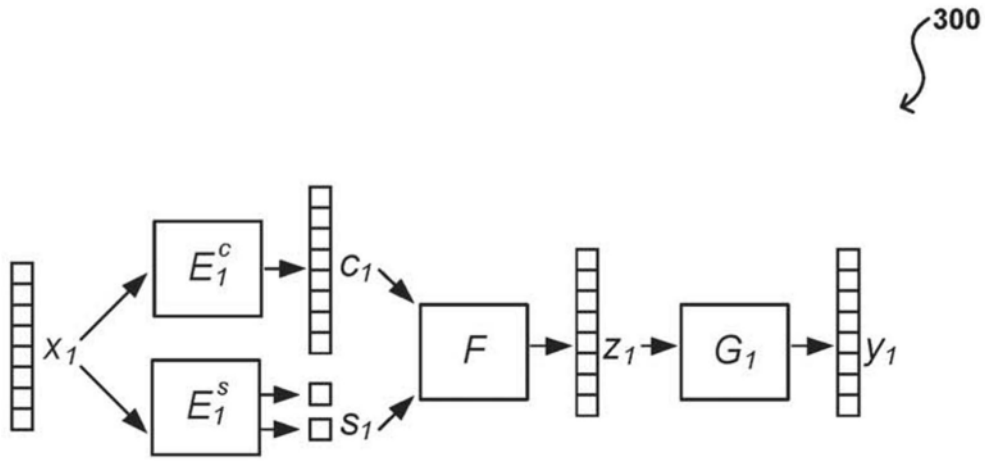

300
x_1
E_1^c
E_1^s
c_1
s_1
F
z_1
G_1
y_1

图3A

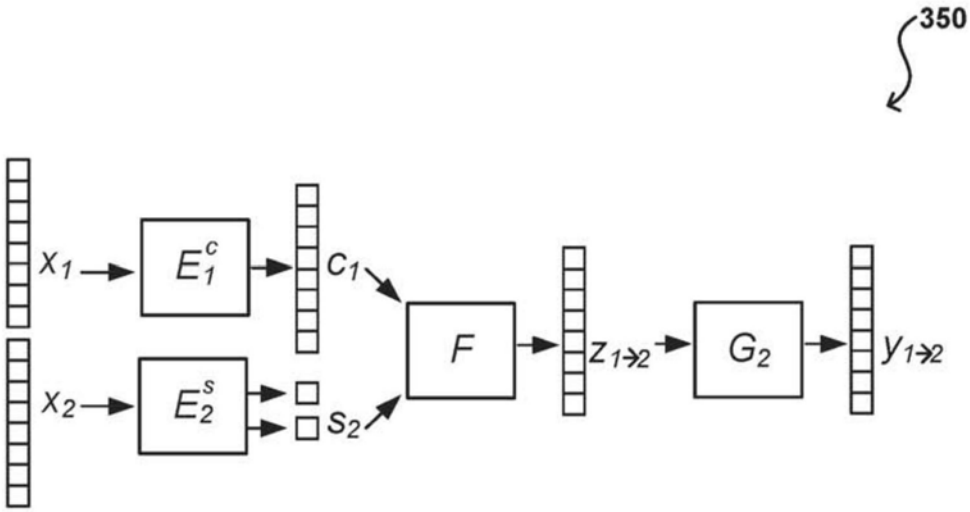

350
x_1
x_2
E_1^c
E_2^s
c_1
s_2
F
$z_{1\to 2}$
G_2
$y_{1\to 2}$

图3B

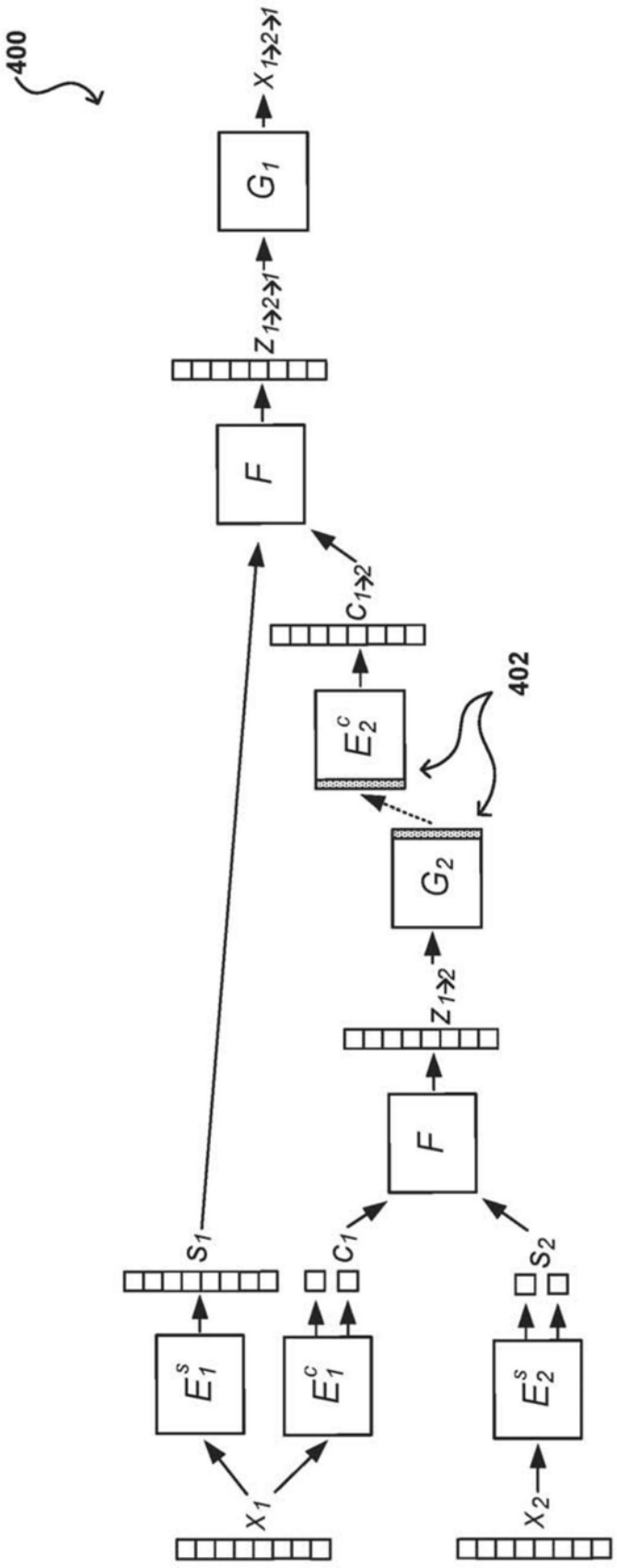

400
$x_{1\to2\to1}$
G_1
$z_{1\to2\to1}$
F
$c_{1\to2}$
402
E_2^c
G_2
$z_{1\to2}$
F
s_1
c_1
s_2
E_1^s
E_1^c
E_2^s
x_1
x_2

图4

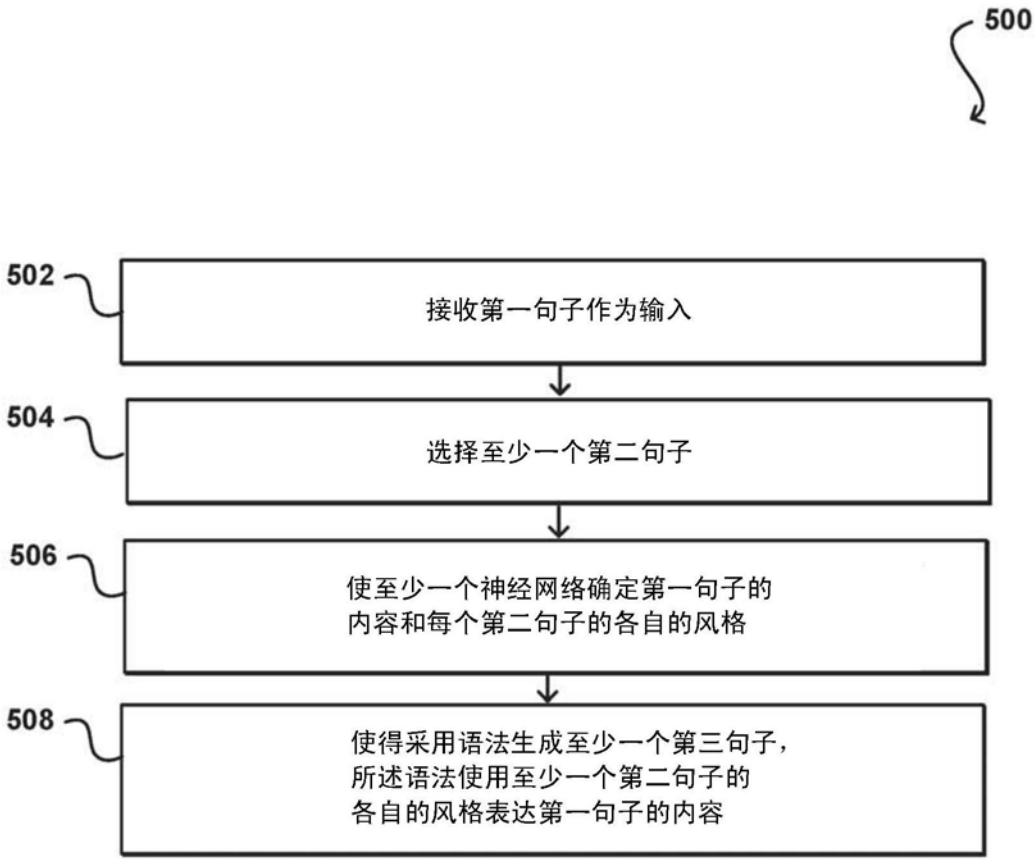

500
502
接收第一句子作为输入
504
选择至少一个第二句子
506
使至少一个神经网络确定第一句子的
内容和每个第二句子的各自的风格
508
使得采用语法生成至少一个第三句子，
所述语法使用至少一个第二句子的
各自的风格表达第一句子的内容

图5

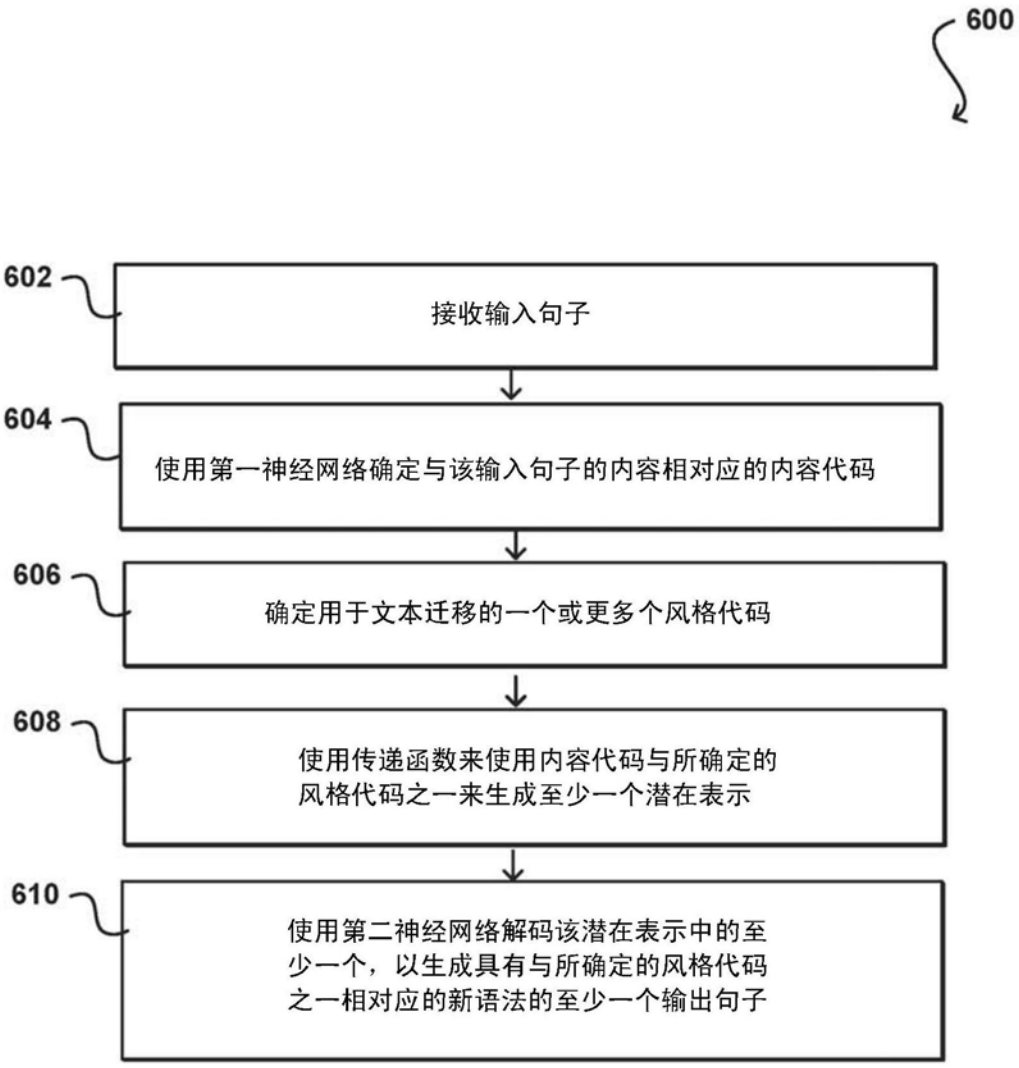

600
602
接收输入句子
604
使用第一神经网络确定与该输入句子的内容相对应的内容代码
606
确定用于文本迁移的一个或更多个风格代码
608
使用传递函数来使用内容代码与所确定的风格代码之一来生成至少一个潜在表示
610
使用第二神经网络解码该潜在表示中的至少一个，以生成具有与所确定的风格代码之一相对应的新语法的至少一个输出句子

图6

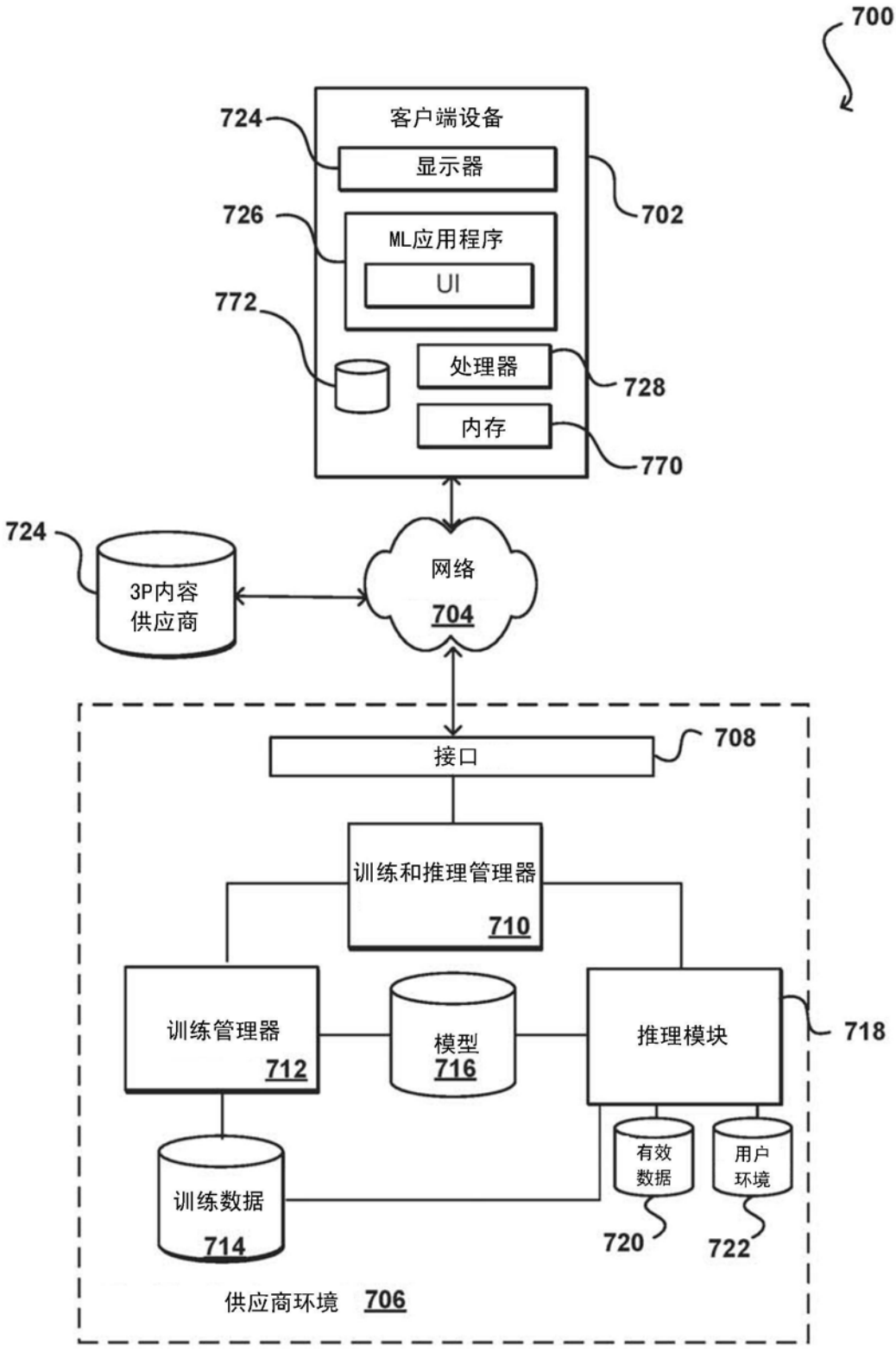

700
客户端设备
724
显示器
726
ML应用程序
UI
772
处理器
728
内存
770
702
724
3P内容
供应商
网络
704
接口
708
训练和推理管理器
710
训练管理器
712
模型
716
推理模块
718
有效
数据
用户
环境
训练数据
714
720
722
供应商环境 706

图7

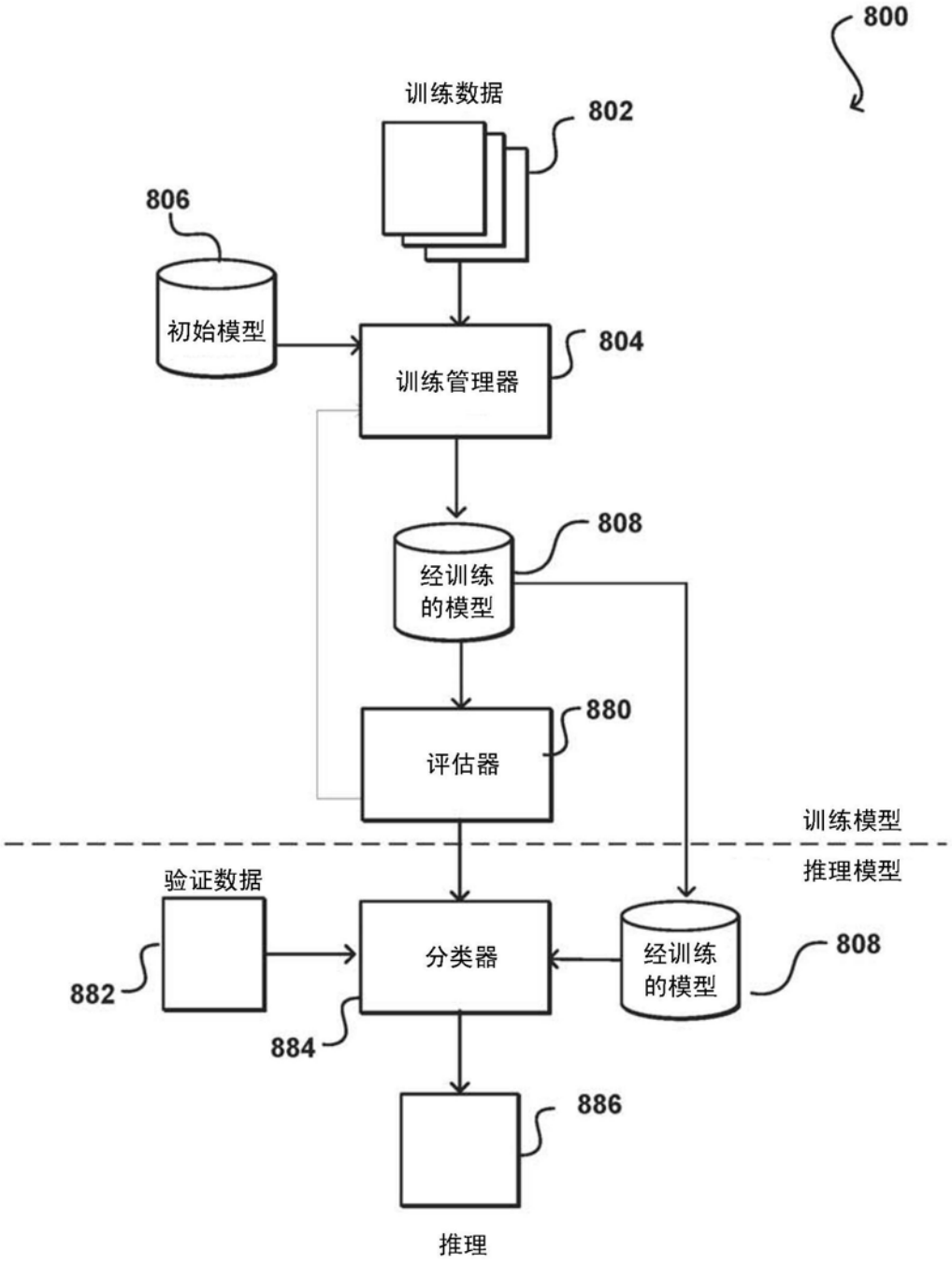

800
训练数据
802
806
初始模型
804
训练管理器
808
经训练
的模型
880
评估器
训练模型
推理模型
验证数据
882
分类器
经训练
的模型
808
884
886
推理

图8

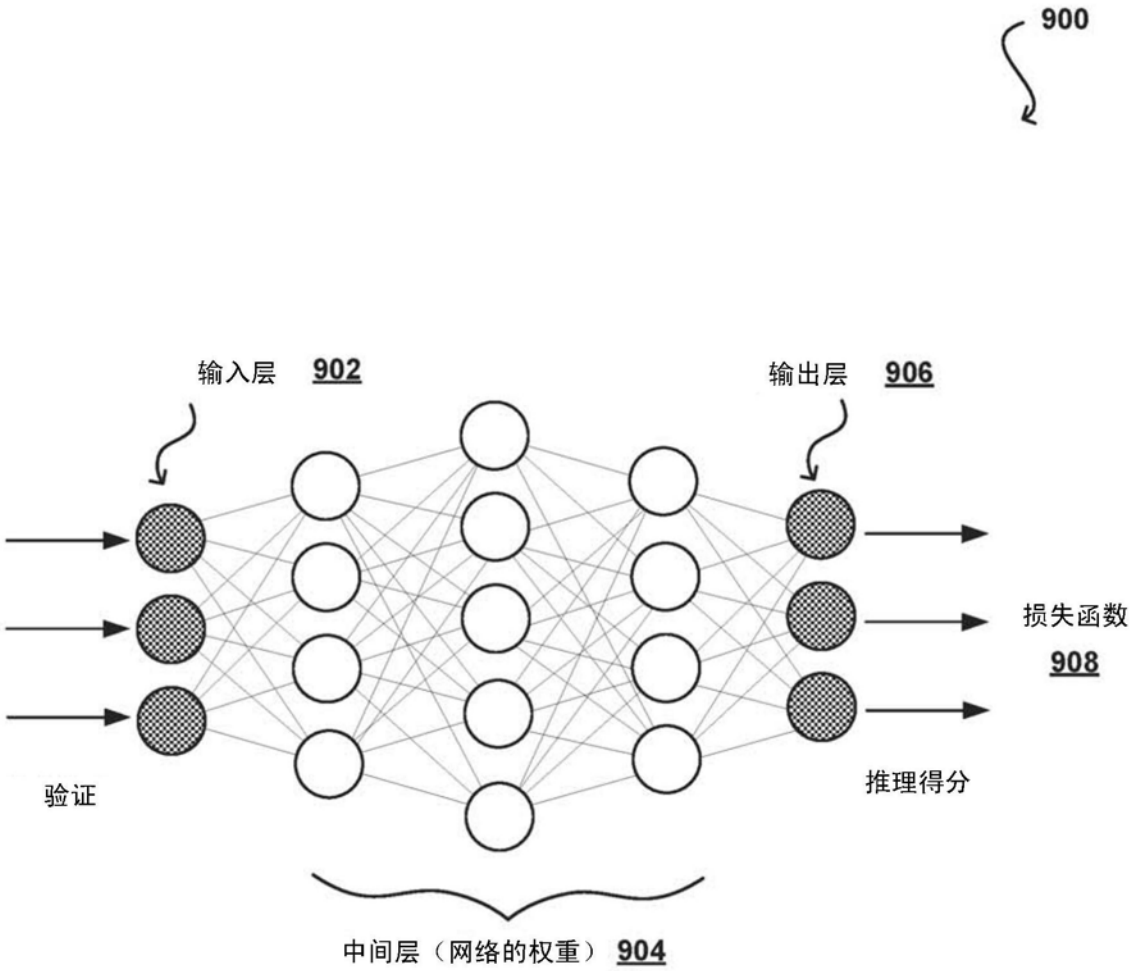

900
输入层 902
输出层 906
损失函数
908
验证
推理得分
中间层（网络的权重） 904

图9

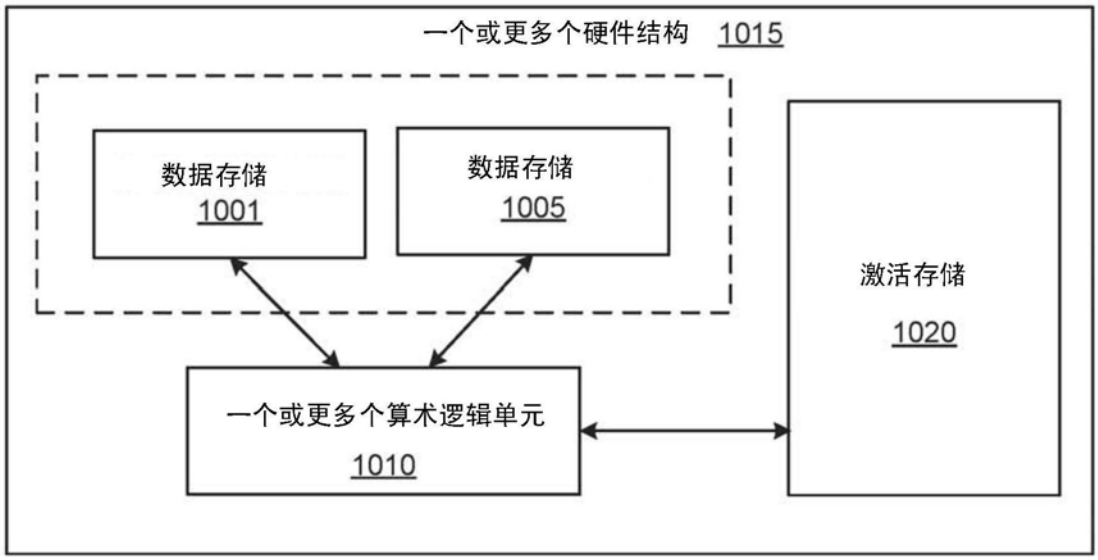

一个或更多个硬件结构 1015
数据存储
1001
数据存储
1005
激活存储
1020
一个或更多个算术逻辑单元
1010

图10

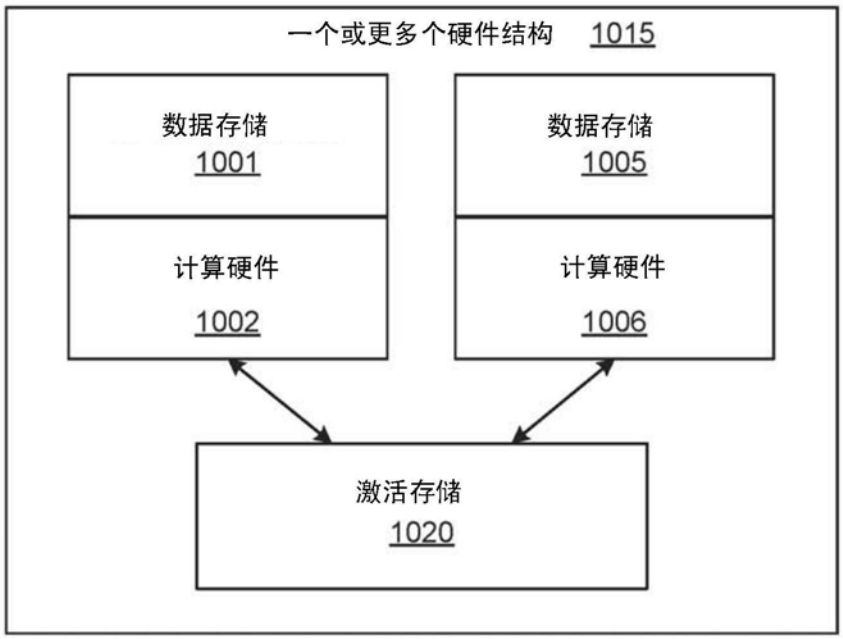

一个或更多个硬件结构 1015
数据存储
1001
数据存储
1005
计算硬件
1002
计算硬件
1006
激活存储
1020

图11

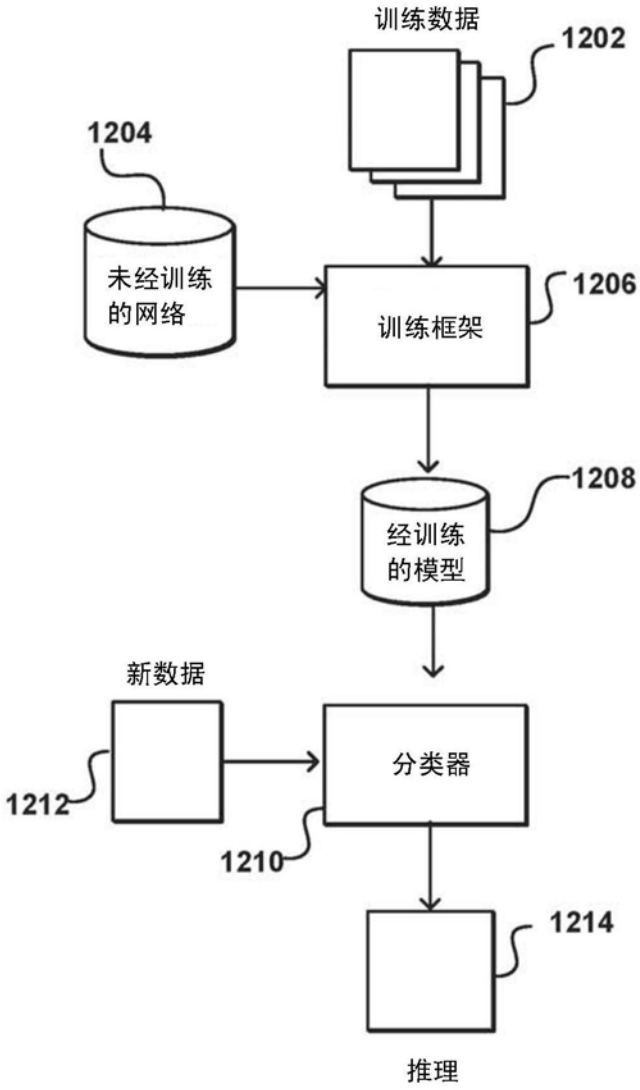

训练数据
1202
1204
未经训练的网络
训练框架
1206
1208
经训练的模型
新数据
1212
分类器
1210
1214
推理

图12

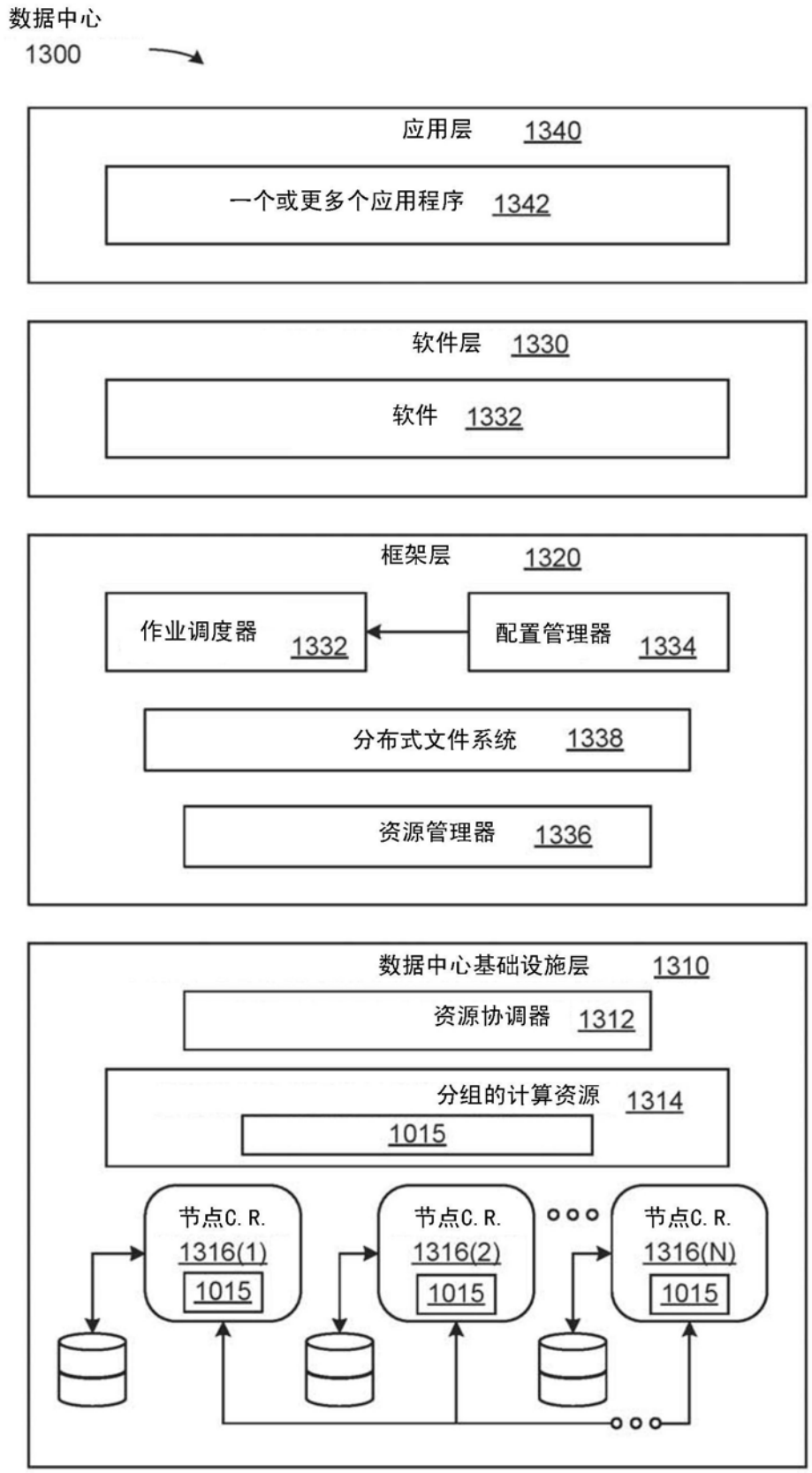

数据中心
1300
应用层 1340
一个或更多个应用程序 1342
软件层 1330
软件 1332
框架层 1320
作业调度器 1332
配置管理器 1334
分布式文件系统 1338
资源管理器 1336
数据中心基础设施层 1310
资源协调器 1312
分组的计算资源 1314
1015
节点C. R. 1316(1) 1015
节点C. R. 1316(2) 1015
节点C. R. 1316(N) 1015

图13

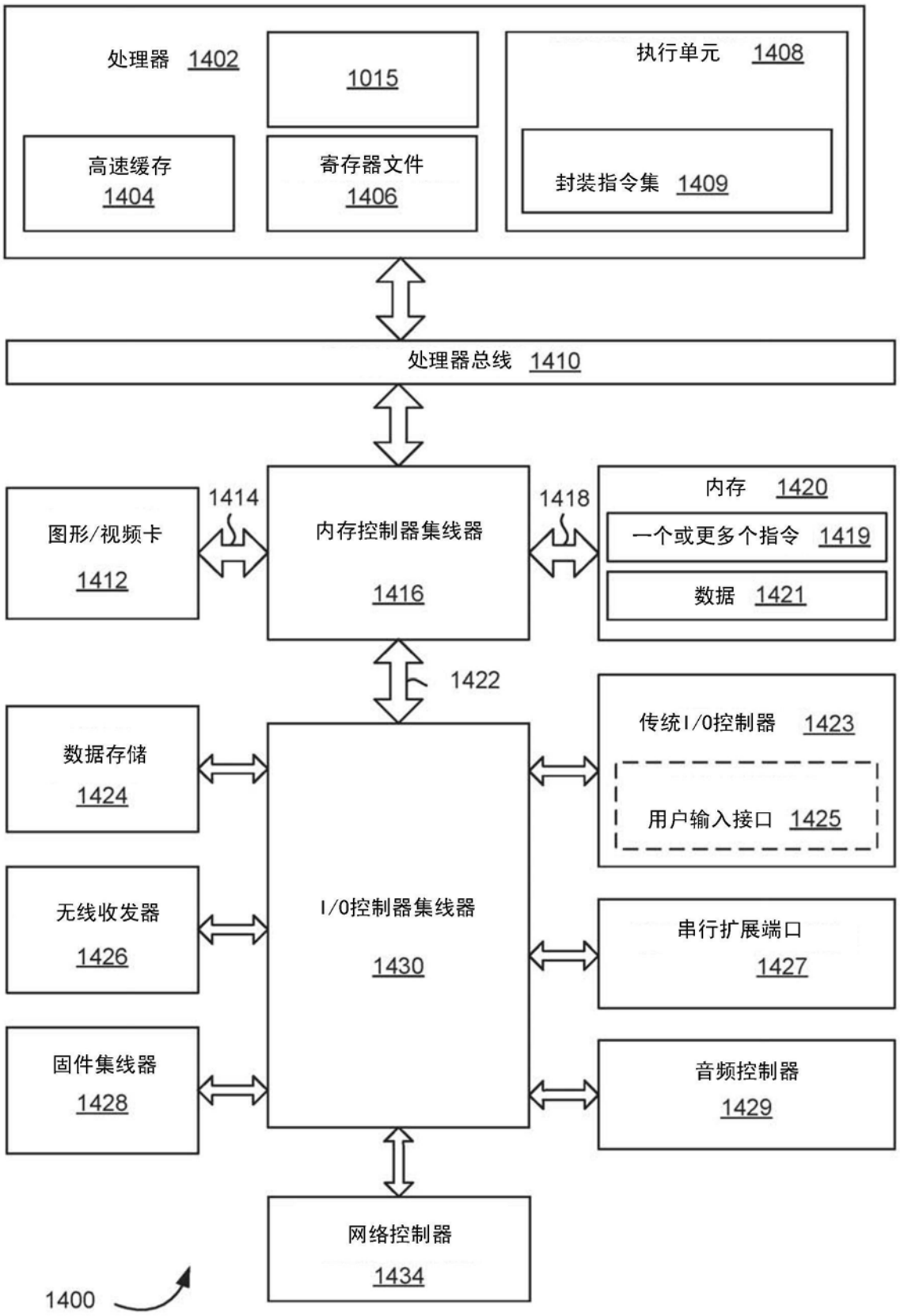

处理器 1402
1015
执行单元 1408
高速缓存 1404
寄存器文件 1406
封装指令集 1409
处理器总线 1410
1414
1418
图形/视频卡 1412
内存控制器集线器 1416
内存 1420
一个或更多个指令 1419
数据 1421
1422
传统I/O控制器 1423
数据存储 1424
用户输入接口 1425
无线收发器 1426
I/O控制器集线器 1430
串行扩展端口 1427
固件集线器 1428
音频控制器 1429
网络控制器 1434
1400

图14

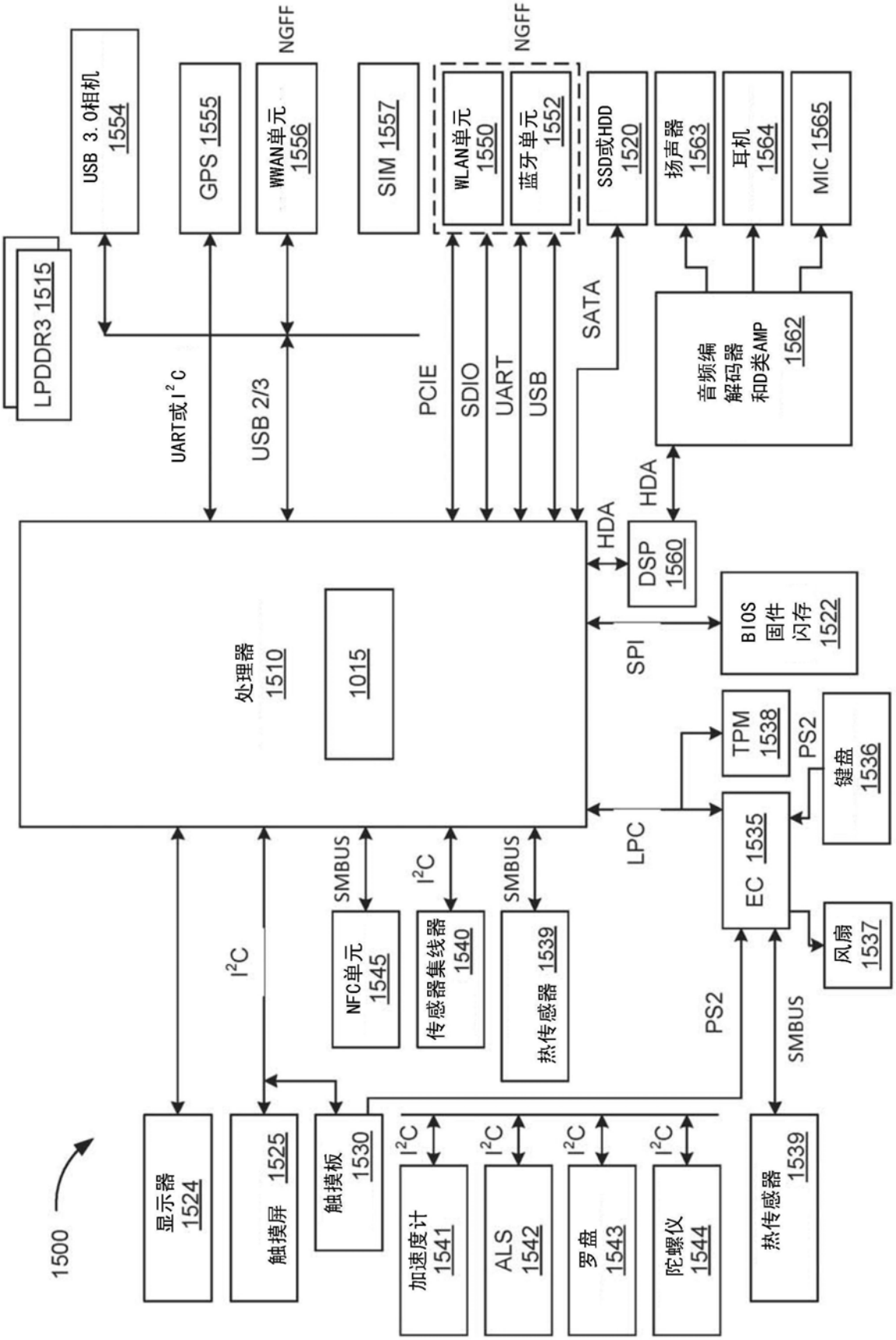

USB 3.0相机 1554
GPS 1555
NGFF
WWAN单元 1556
SIM 1557
NGFF
WLAN单元 1550
蓝牙单元 1552
SSD或HDD 1520
扬声器 1563
耳机 1564
MIC 1565
LPDDR3 1515
UART或I²C
USB 2/3
PCIE
SDIO
UART
USB
SATA
音频编解码器和D类AMP 1562
HDA
HDA
DSP 1560
处理器 1510
1015
SPI
BIOS固件闪存 1522
TPM 1538
PS2
键盘 1536
LPC
EC 1535
风扇 1537
SMBUS
I²C
SMBUS
I²C
SMBUS
NFC单元 1545
传感器集线器 1540
热传感器 1539
PS2
SMBUS
I²C
显示器 1524
触摸屏 1525
触摸板 1530
I²C
I²C
I²C
I²C
加速度计 1541
ALS 1542
罗盘 1543
陀螺仪 1544
热传感器 1539
1500

图15

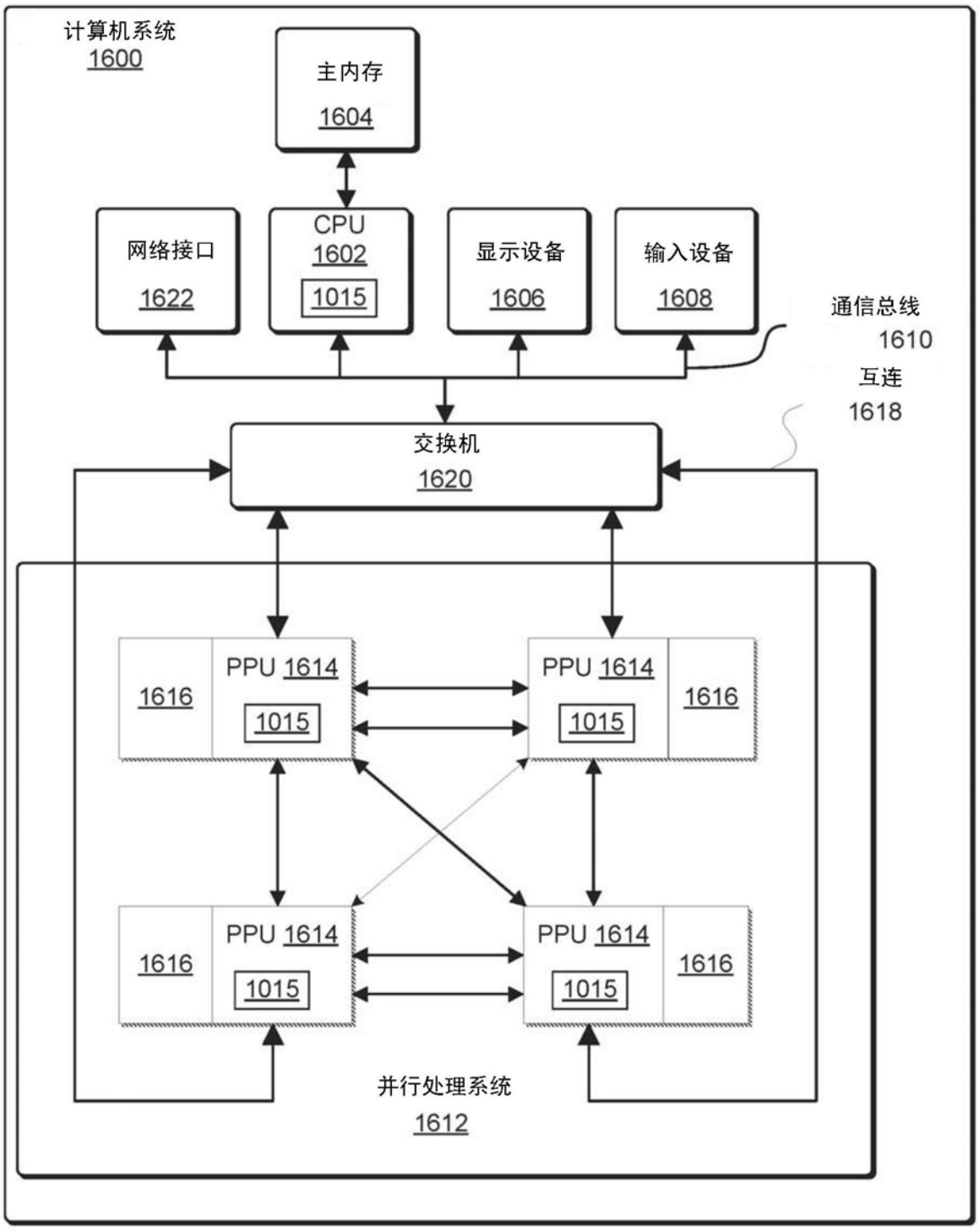

计算机系统
1600
主内存
1604
网络接口
1622
CPU
1602
1015
显示设备
1606
输入设备
1608
通信总线
1610
互连
1618
交换机
1620
1616
PPU 1614
1015
PPU 1614
1015
1616
1616
PPU 1614
1015
PPU 1614
1015
1616
并行处理系统
1612

图16

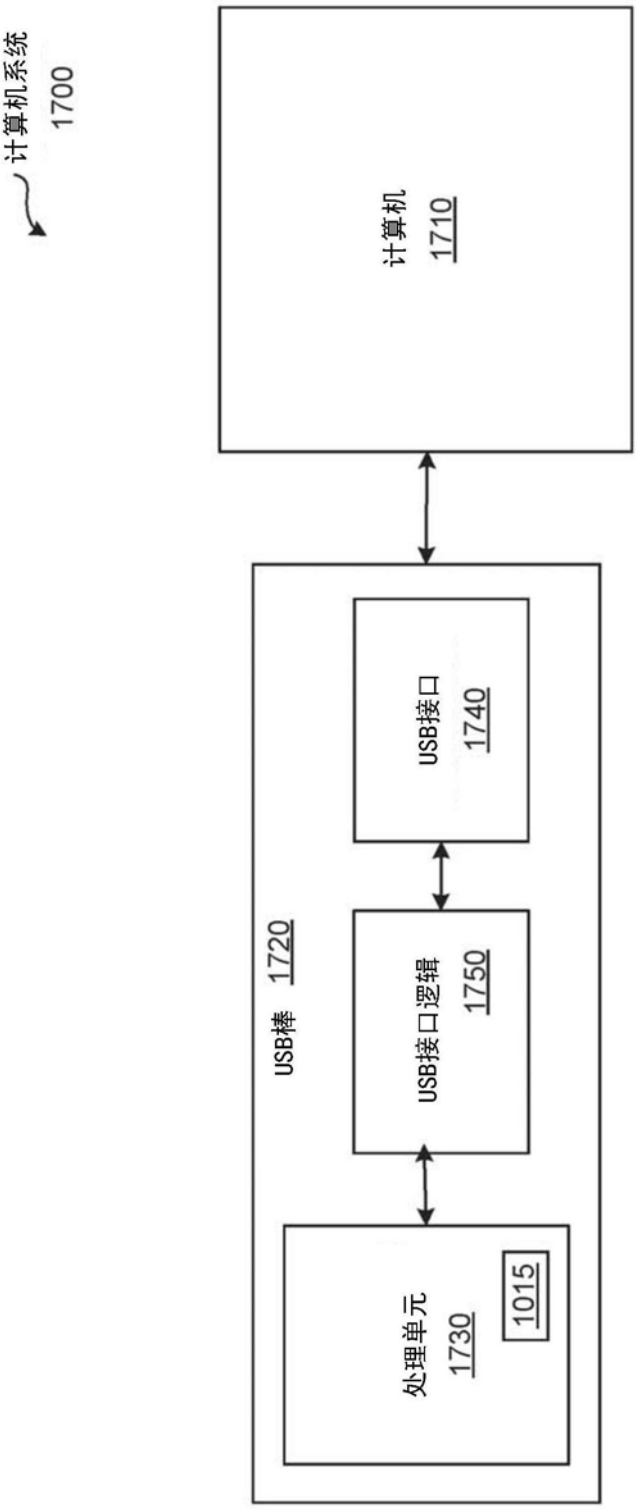

计算机系统
1700
计算机
1710
USB棒
1720
USB接口
1740
USB接口逻辑
1750
处理单元
1730
1015

图17

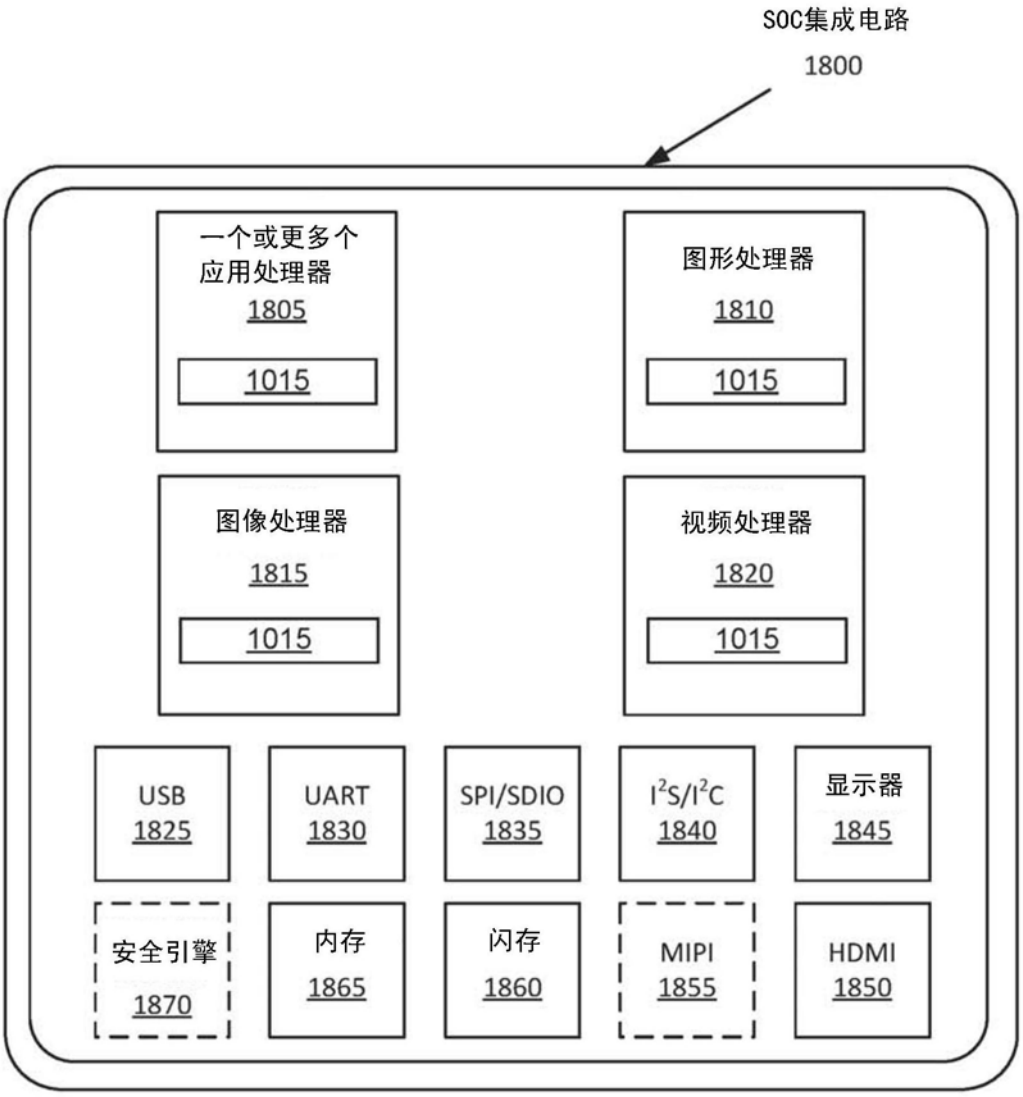

SOC集成电路
1800
一个或更多个
应用处理器
1805
1015
图形处理器
1810
1015
图像处理器
1815
1015
视频处理器
1820
1015
USB
1825
UART
1830
SPI/SDIO
1835
I²S/I²C
1840
显示器
1845
安全引擎
1870
内存
1865
闪存
1860
MIPI
1855
HDMI
1850

图18

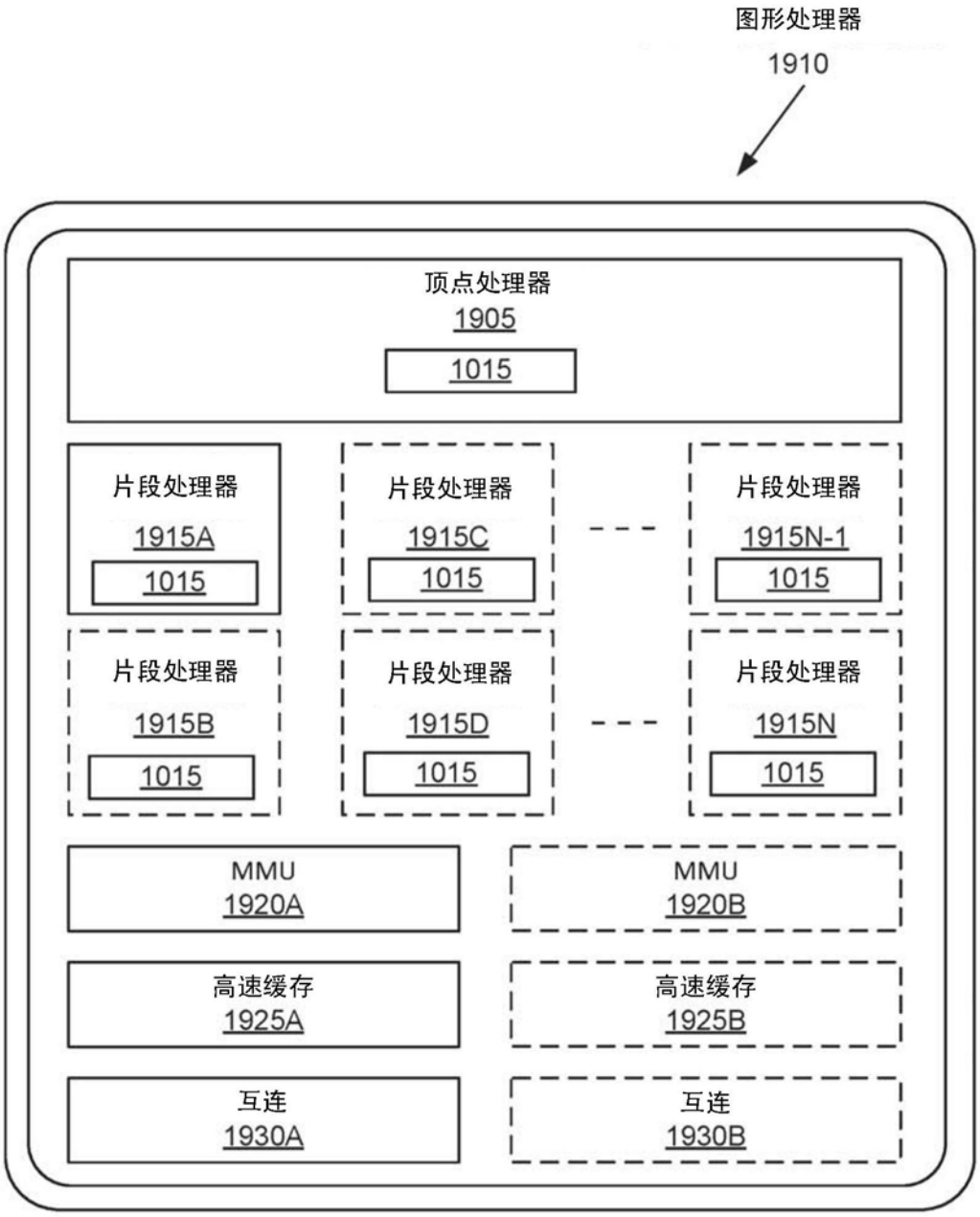

图形处理器
1910
顶点处理器
1905
1015
片段处理器
1915A
1015
片段处理器
1915C
1015
片段处理器
1915N-1
1015
片段处理器
1915B
1015
片段处理器
1915D
1015
片段处理器
1915N
1015
MMU
1920A
MMU
1920B
高速缓存
1925A
高速缓存
1925B
互连
1930A
互连
1930B

图19A

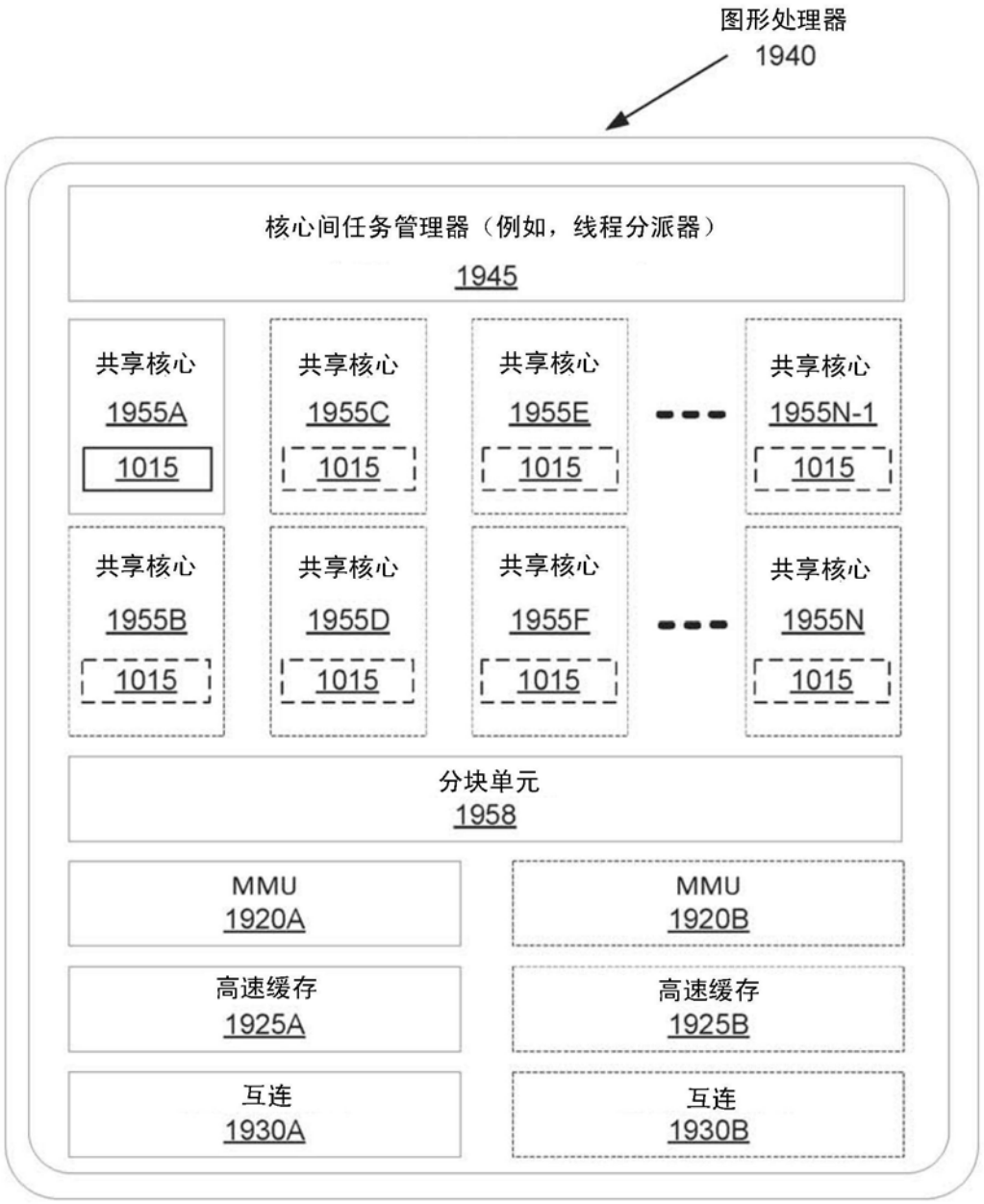

图形处理器
1940
核心间任务管理器（例如，线程分派器）
1945
共享核心
1955A
1015
共享核心
1955C
1015
共享核心
1955E
1015
共享核心
1955N-1
1015
共享核心
1955B
1015
共享核心
1955D
1015
共享核心
1955F
1015
共享核心
1955N
1015
分块单元
1958
MMU
1920A
MMU
1920B
高速缓存
1925A
高速缓存
1925B
互连
1930A
互连
1930B

图19B

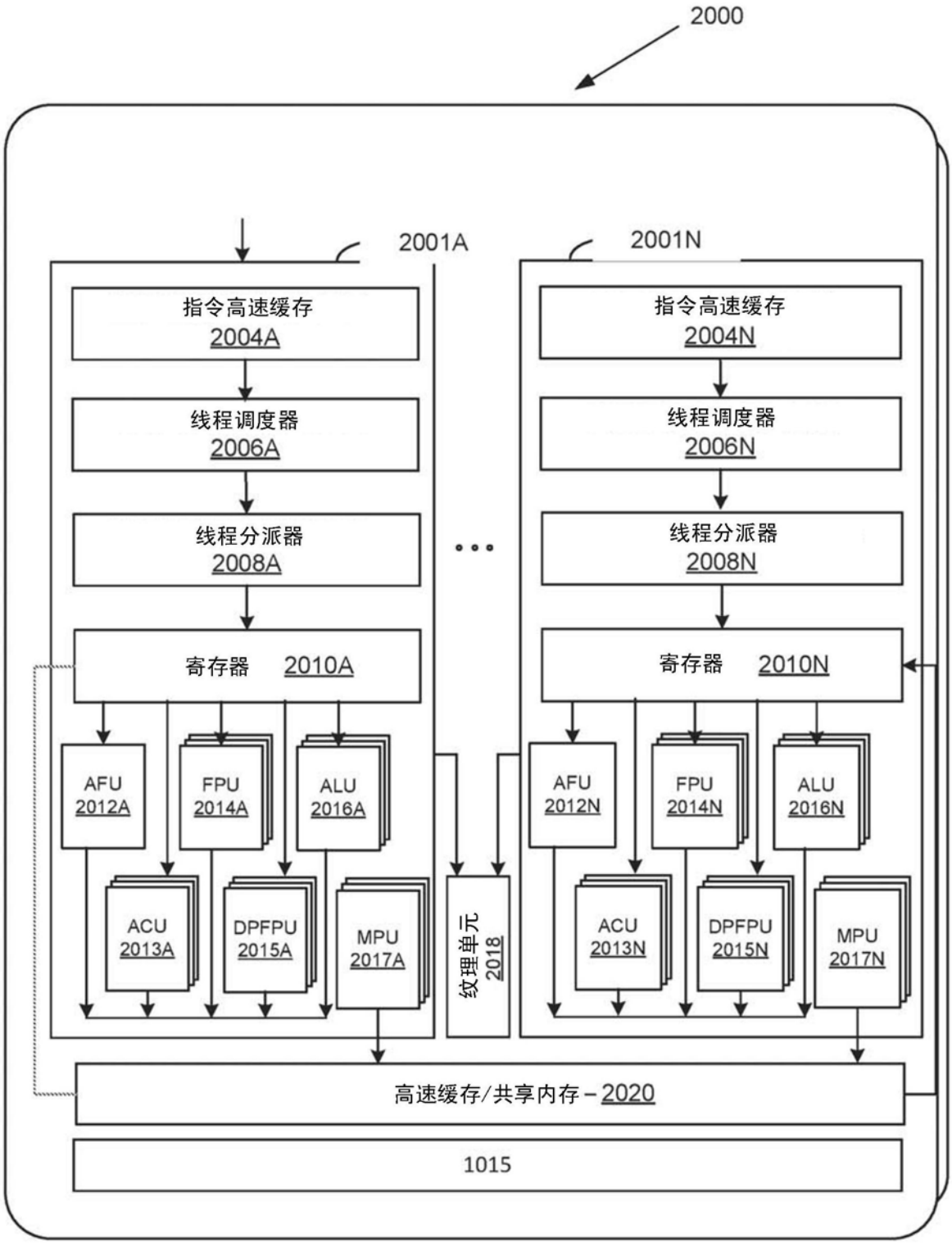

2000
2001A
2001N
指令高速缓存
2004A
指令高速缓存
2004N
线程调度器
2006A
线程调度器
2006N
线程分派器
2008A
线程分派器
2008N
寄存器 2010A
寄存器 2010N
AFU
2012A
FPU
2014A
ALU
2016A
AFU
2012N
FPU
2014N
ALU
2016N
ACU
2013A
DPFPU
2015A
MPU
2017A
纹理单元
2018
ACU
2013N
DPFPU
2015N
MPU
2017N
高速缓存/共享内存－2020
1015

图20A

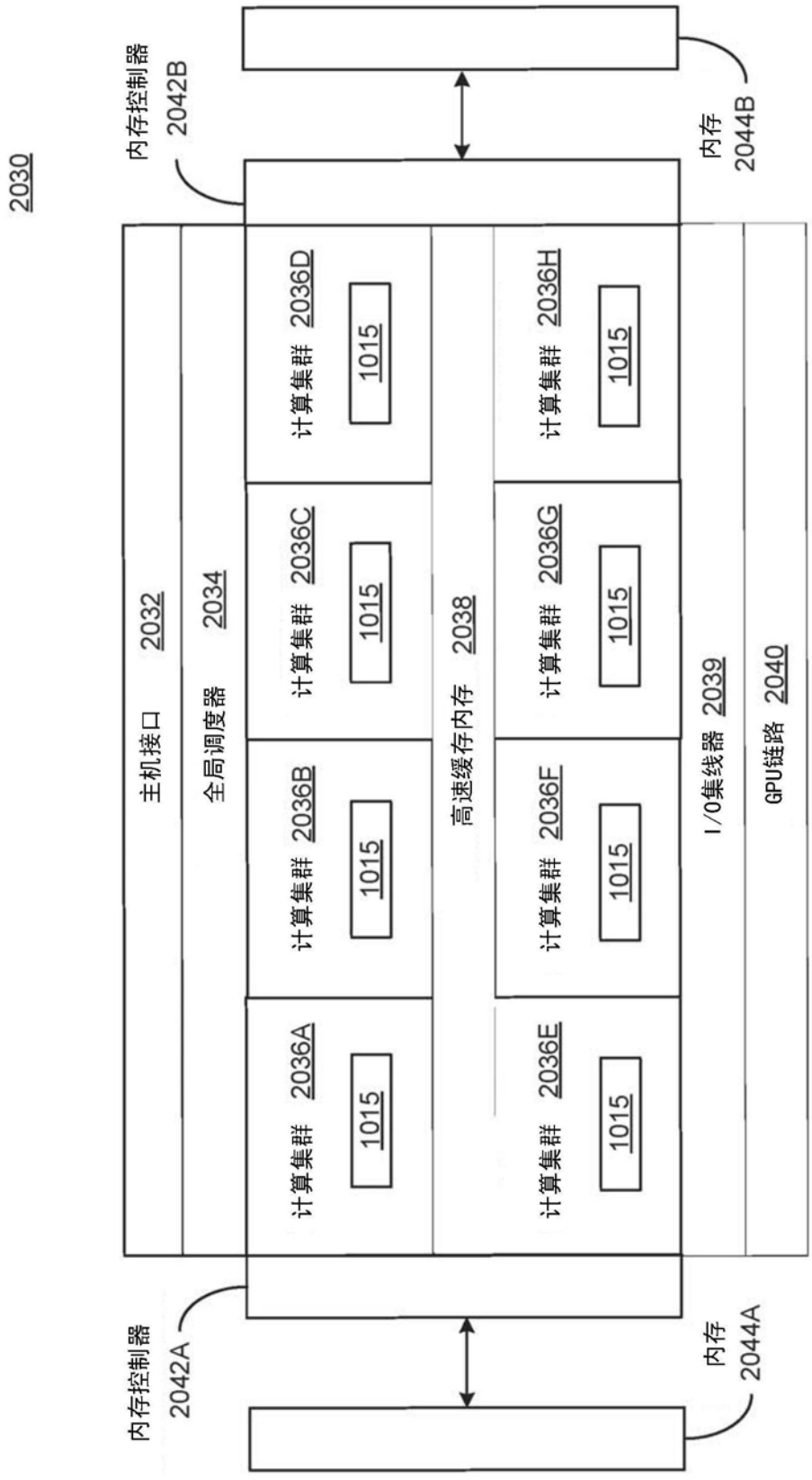

2030
内存控制器 2042B
内存 2044B
主机接口 2032
全局调度器 2034
计算集群 2036D
计算集群 2036H
计算集群 2036C
计算集群 2036G
计算集群 2036B
计算集群 2036F
计算集群 2036A
计算集群 2036E
1015
高速缓存内存 2038
I/O集线器 2039
GPU链路 2040
内存控制器 2042A
内存 2044A

图20B

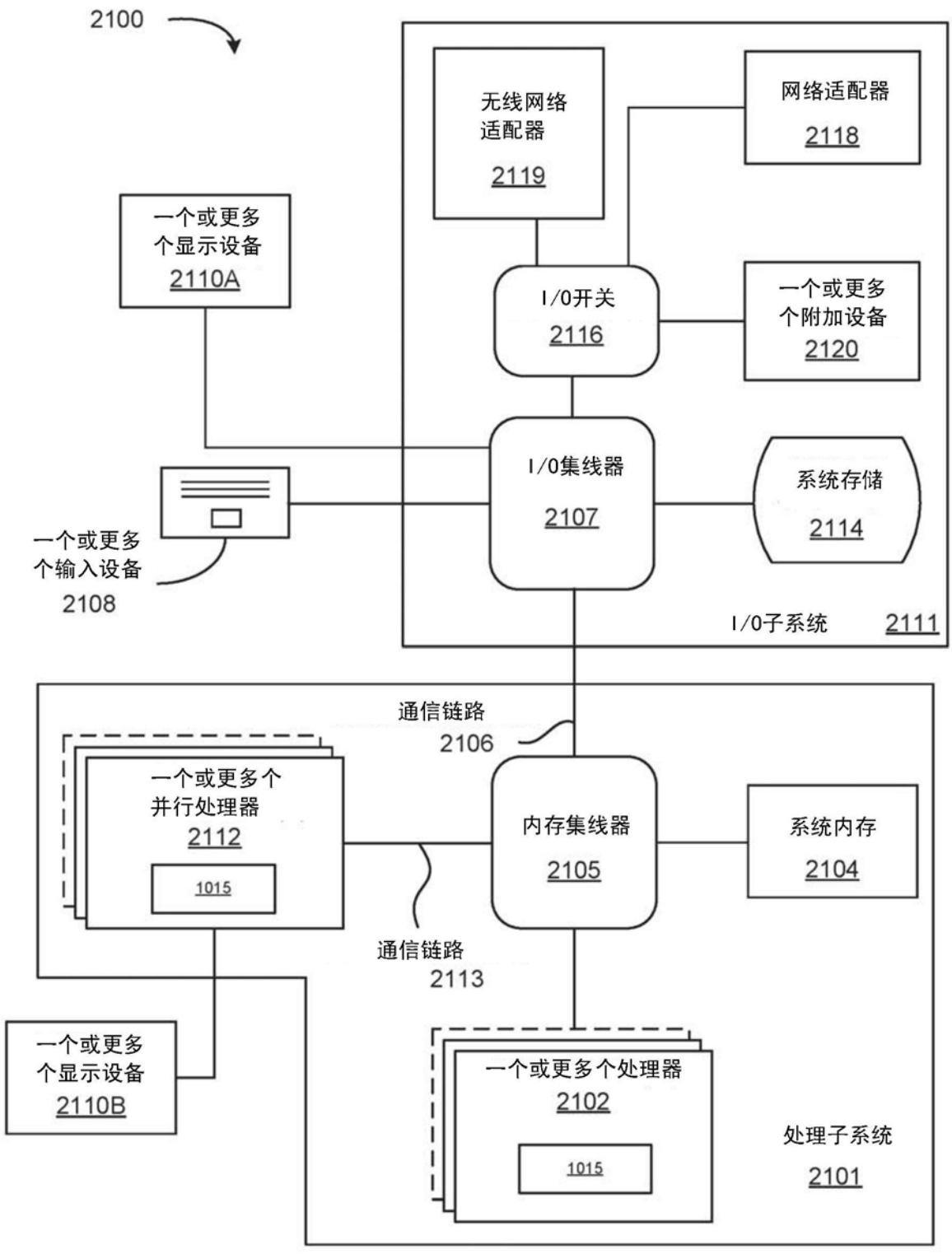

2100
无线网络
适配器
2119
网络适配器
2118
一个或更多
个显示设备
2110A
I/0开关
2116
一个或更多
个附加设备
2120
I/0集线器
2107
系统存储
2114
一个或更多
个输入设备
2108
I/0子系统
2111
通信链路
2106
一个或更多个
并行处理器
2112
1015
内存集线器
2105
系统内存
2104
通信链路
2113
一个或更多
个显示设备
2110B
一个或更多个处理器
2102
1015
处理子系统
2101

图21

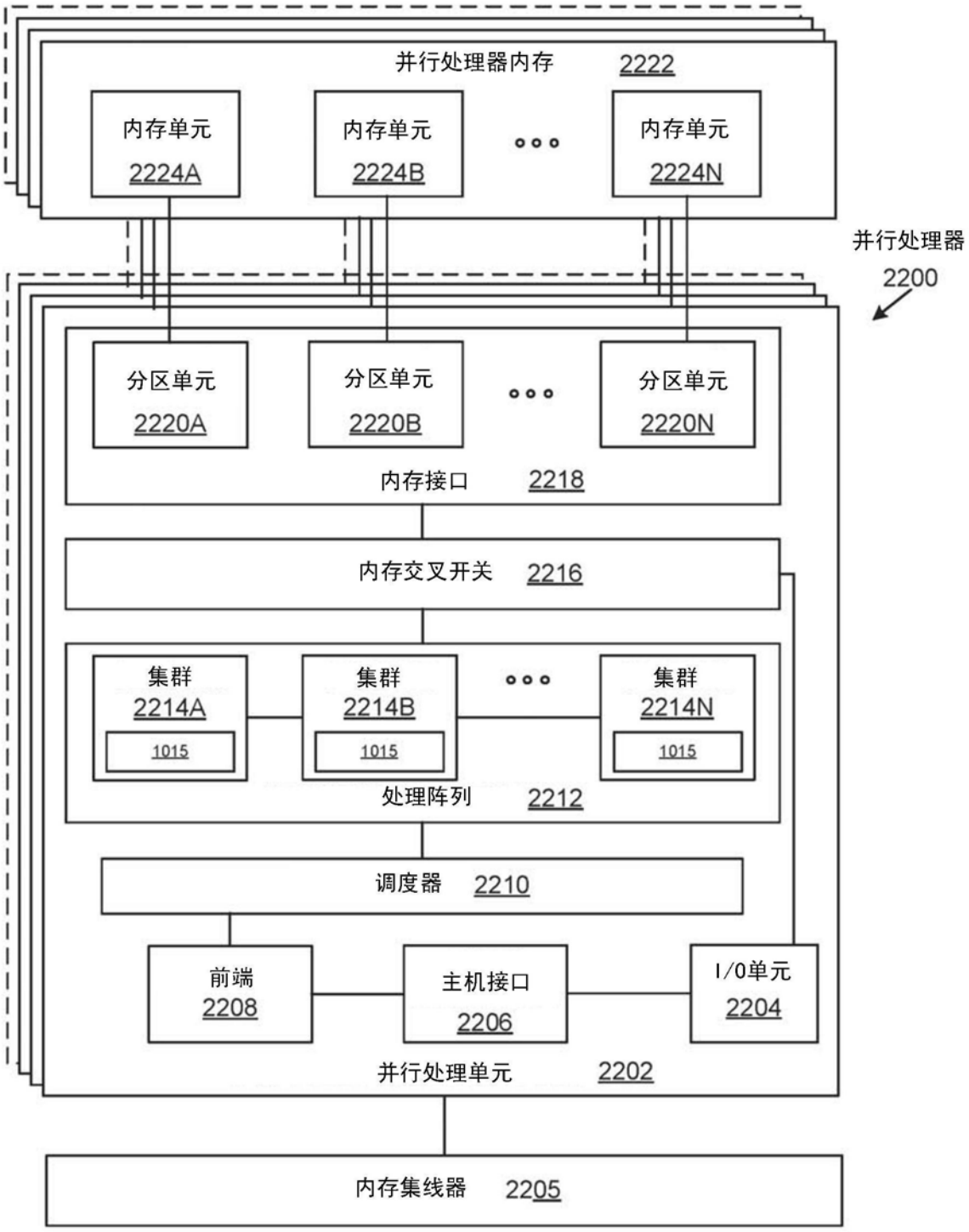

并行处理器内存 2222
内存单元 2224A
内存单元 2224B
内存单元 2224N
并行处理器 2200
分区单元 2220A
分区单元 2220B
分区单元 2220N
内存接口 2218
内存交叉开关 2216
集群 2214A
1015
集群 2214B
1015
集群 2214N
1015
处理阵列 2212
调度器 2210
前端 2208
主机接口 2206
I/O单元 2204
并行处理单元 2202
内存集线器 2205

图22

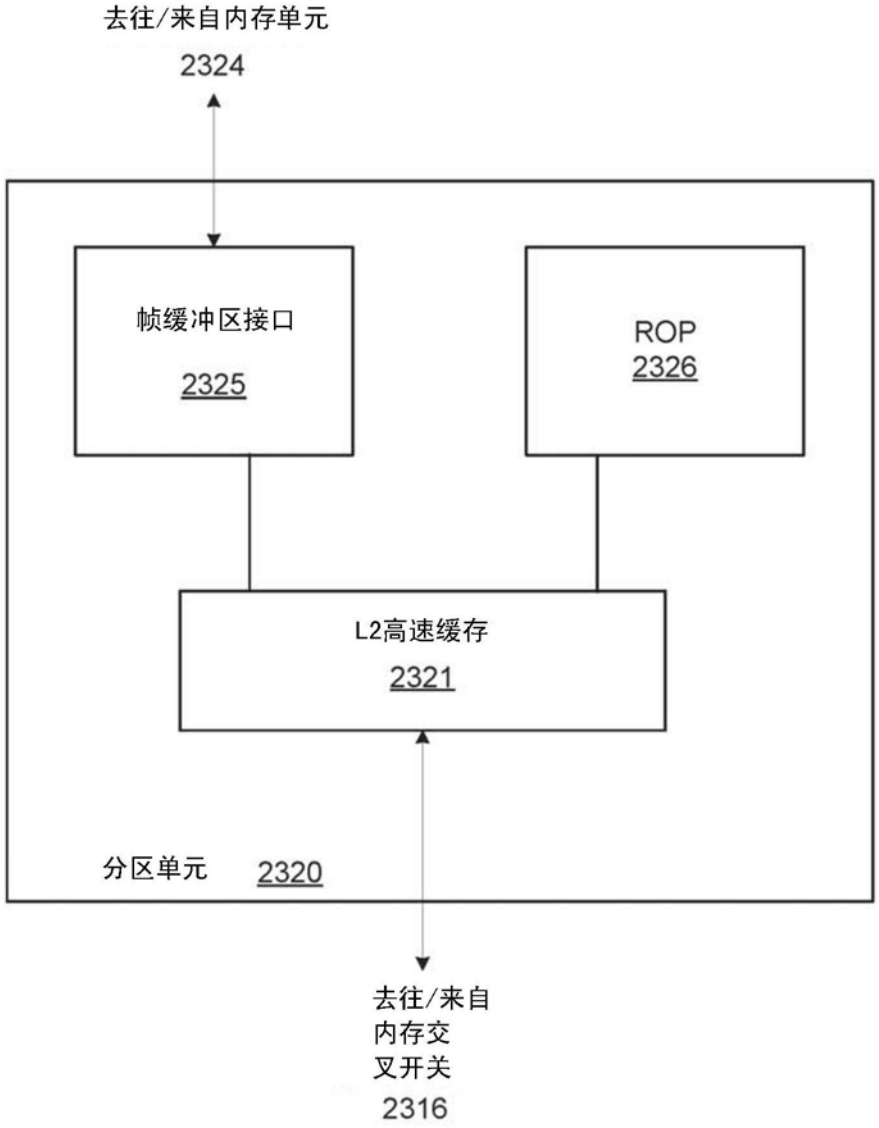

去往/来自内存单元
2324
帧缓冲区接口
2325
ROP
2326
L2高速缓存
2321
分区单元 2320
去往/来自
内存交
叉开关
2316

图23

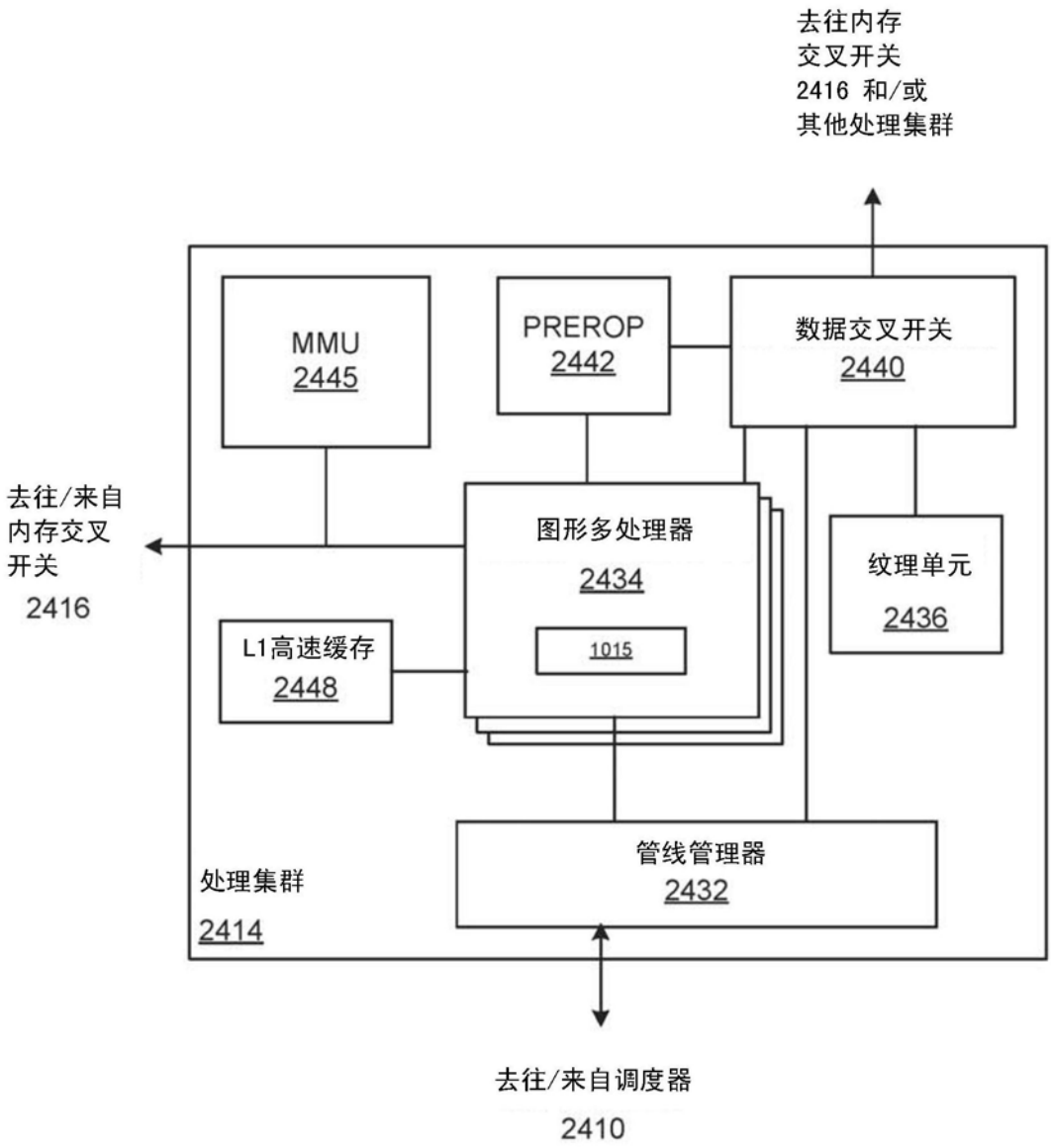

去往内存
交叉开关
2416 和/或
其他处理集群
MMU
2445
PREROP
2442
数据交叉开关
2440
去往/来自
内存交叉
开关
2416
图形多处理器
2434
1015
纹理单元
2436
L1高速缓存
2448
管线管理器
2432
处理集群
2414
去往/来自调度器
2410

图24

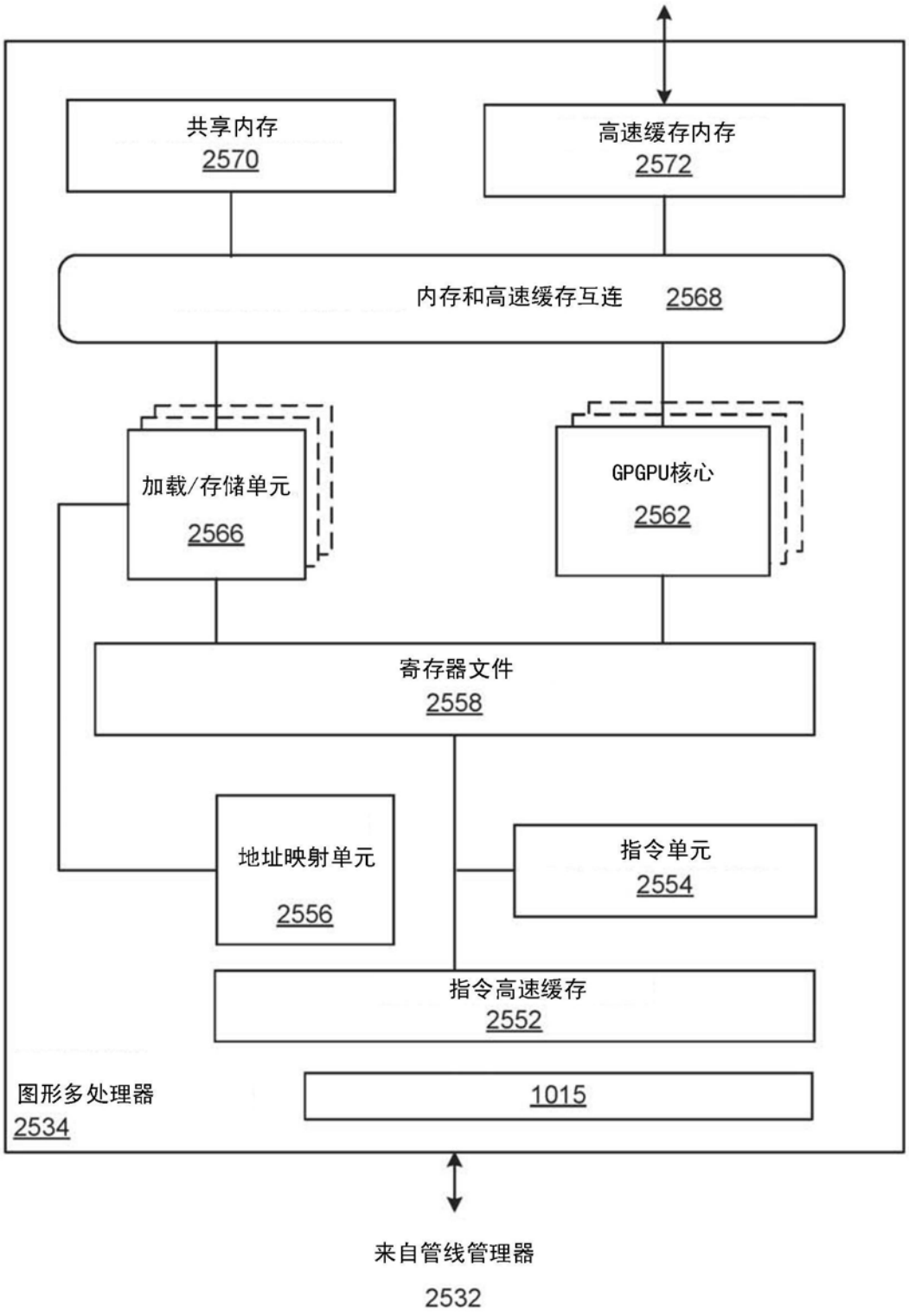

共享内存
2570
高速缓存内存
2572
内存和高速缓存互连 2568
加载/存储单元
2566
GPGPU核心
2562
寄存器文件
2558
地址映射单元
2556
指令单元
2554
指令高速缓存
2552
图形多处理器
2534
1015
来自管线管理器
2532

图25

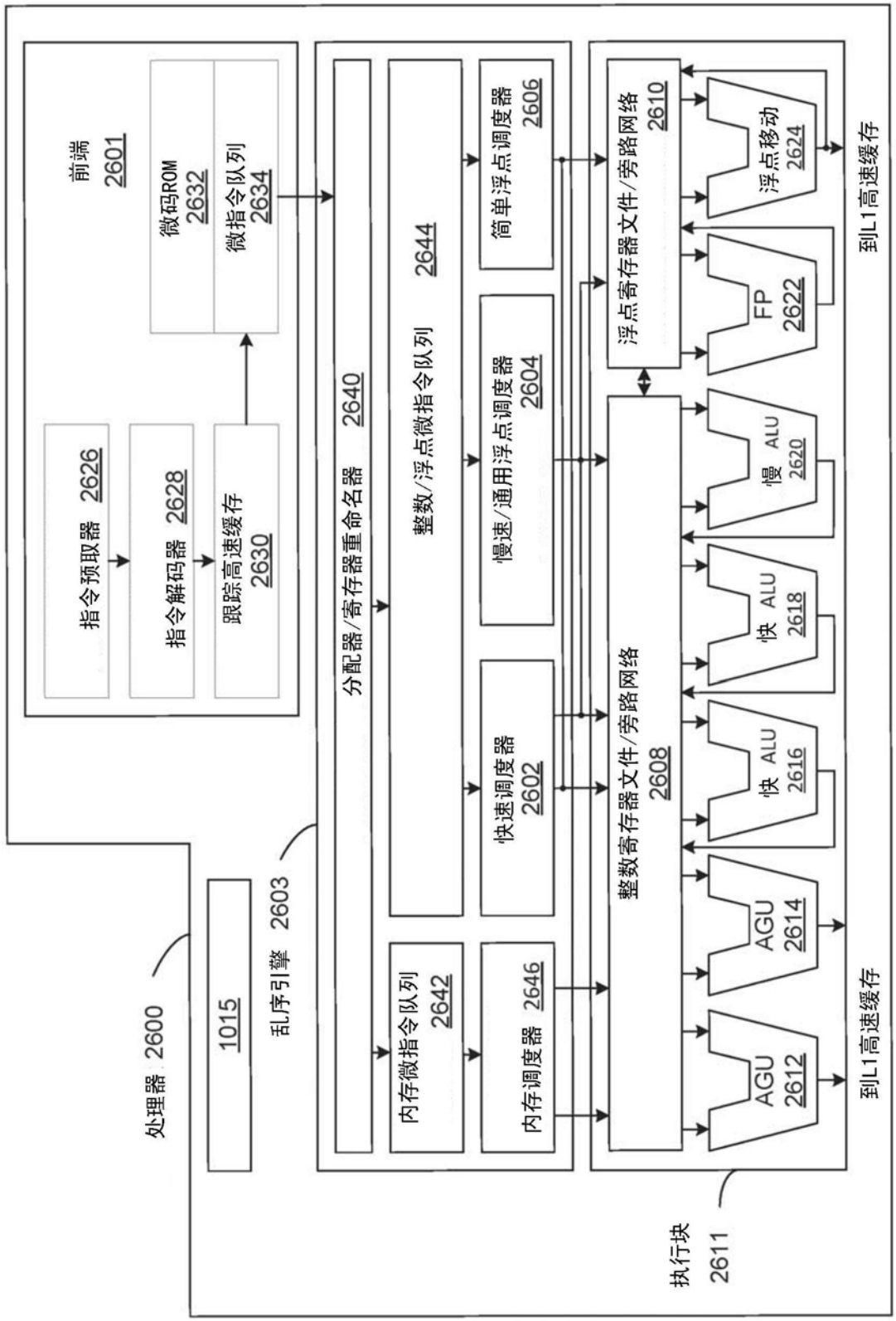

处理器 2600
1015
乱序引擎 2603
前端 2601
指令预取器 2626
指令解码器 2628
跟踪高速缓存 2630
微码ROM 2632
微指令队列 2634
分配器/寄存器重命名器 2640
内存微指令队列 2642
内存调度器 2646
整数/浮点微指令队列 2644
快速调度器 2602
慢速/通用浮点调度器 2604
简单浮点调度器 2606
整数寄存器文件/旁路网络 2608
浮点寄存器文件/旁路网络 2610
执行块 2611
AGU 2612
AGU 2614
快 ALU 2616
快 ALU 2618
慢 ALU 2620
FP 2622
浮点移动 2624
到L1高速缓存
到L1高速缓存

图26

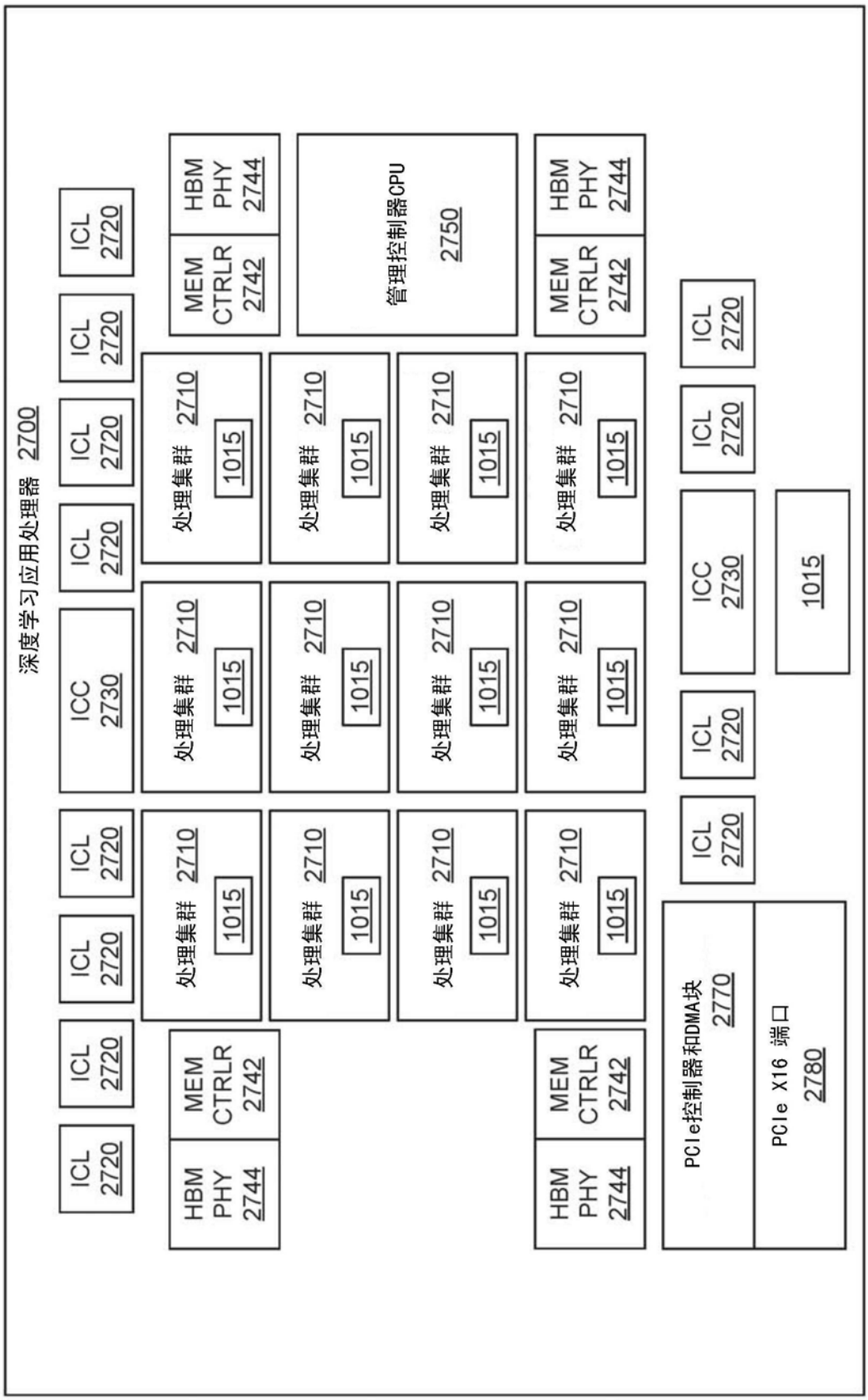

深度学习应用处理器 2700
ICL 2720
ICC 2730
处理集群 2710
1015
MEM CTRLR 2742
HBM PHY 2744
管理控制器CPU 2750
PCIe控制器和DMA块 2770
PCIe X16 端口 2780

图27

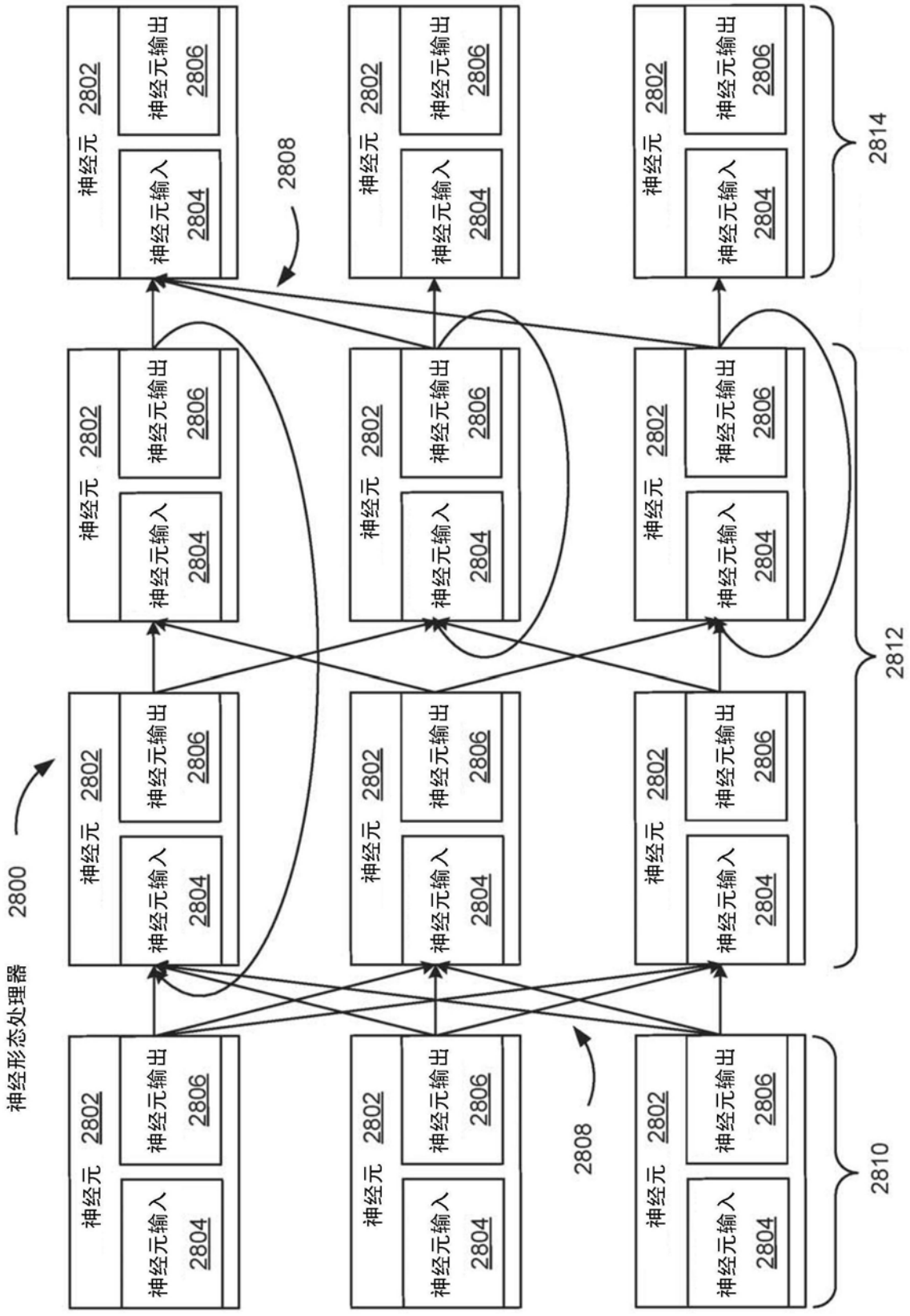

神经形态处理器
2800
神经元 2802
神经元输入 2804
神经元输出 2806
2808
2810
2812
2814

图28

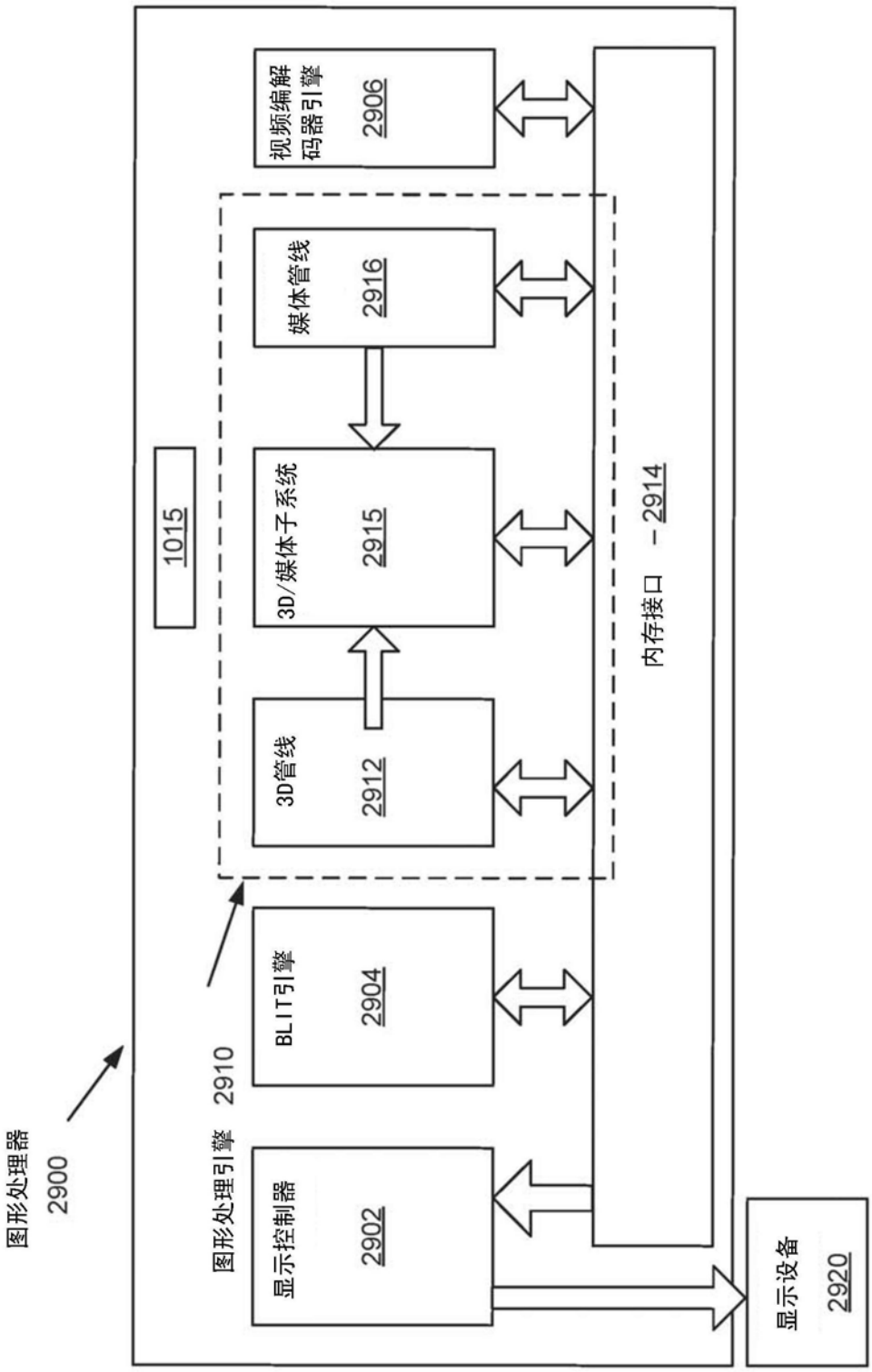

图形处理器
2900
图形处理引擎
2910
显示控制器
2902
BLIT引擎
2904
3D管线
2912
3D/媒体子系统
2915
媒体管线
2916
视频编解码器引擎
2906
1015
内存接口
2914
显示设备
2920

图29

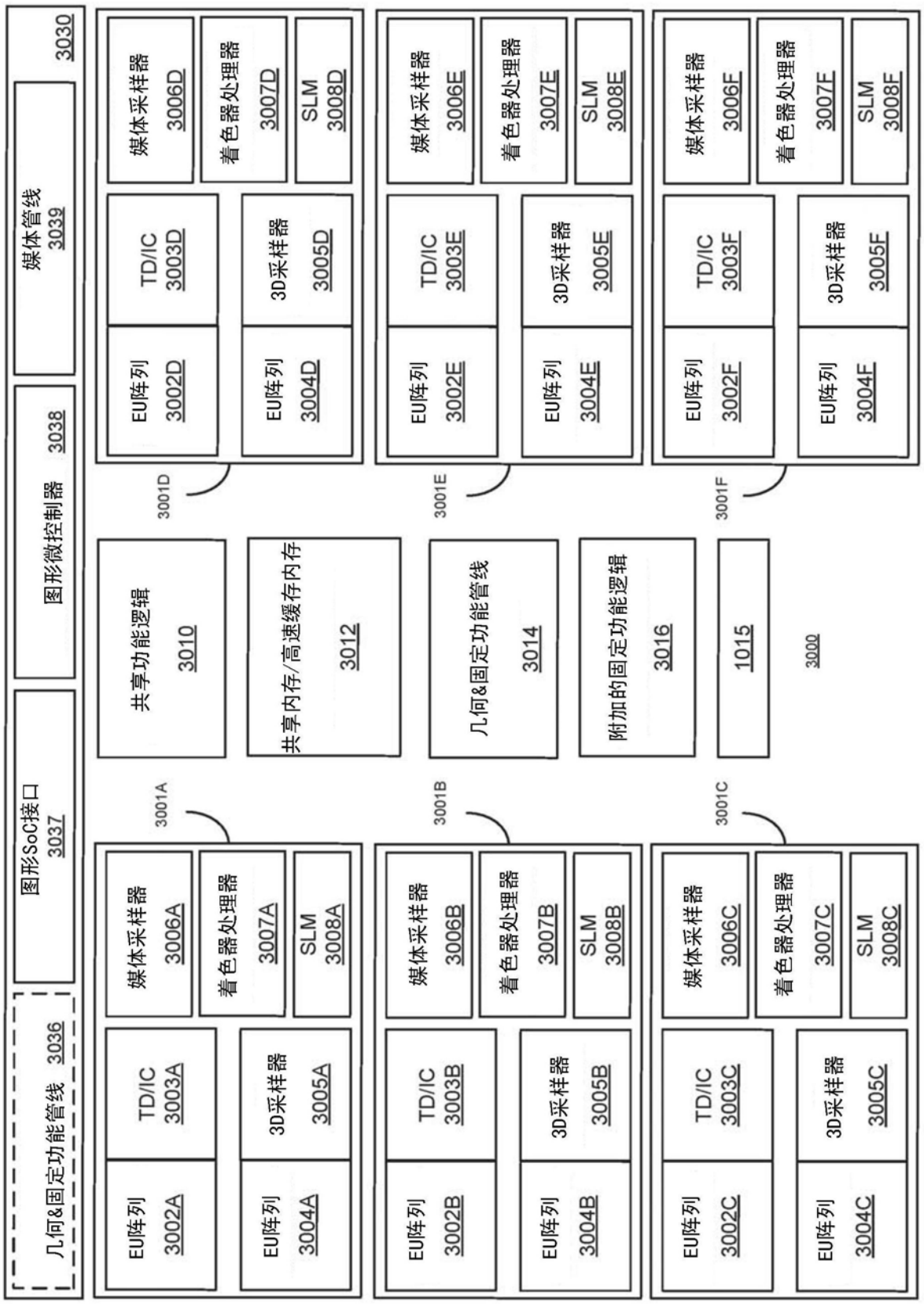

几何&固定功能管线 3036
图形SoC接口 3037
图形微控制器 3038
媒体管线 3039
3030
3001A
3001B
3001C
3001D
3001E
3001F
3000
EU阵列 3002A
TD/IC 3003A
媒体采样器 3006A
着色器处理器 3007A
SLM 3008A
EU阵列 3004A
3D采样器 3005A
EU阵列 3002B
TD/IC 3003B
媒体采样器 3006B
着色器处理器 3007B
SLM 3008B
EU阵列 3004B
3D采样器 3005B
EU阵列 3002C
TD/IC 3003C
媒体采样器 3006C
着色器处理器 3007C
SLM 3008C
EU阵列 3004C
3D采样器 3005C
共享功能逻辑 3010
共享内存/高速缓存内存 3012
几何&固定功能管线 3014
附加的固定功能逻辑 3016
1015
EU阵列 3002D
TD/IC 3003D
媒体采样器 3006D
着色器处理器 3007D
SLM 3008D
EU阵列 3004D
3D采样器 3005D
EU阵列 3002E
TD/IC 3003E
媒体采样器 3006E
着色器处理器 3007E
SLM 3008E
EU阵列 3004E
3D采样器 3005E
EU阵列 3002F
TD/IC 3003F
媒体采样器 3006F
着色器处理器 3007F
SLM 3008F
EU阵列 3004F
3D采样器 3005F

图30

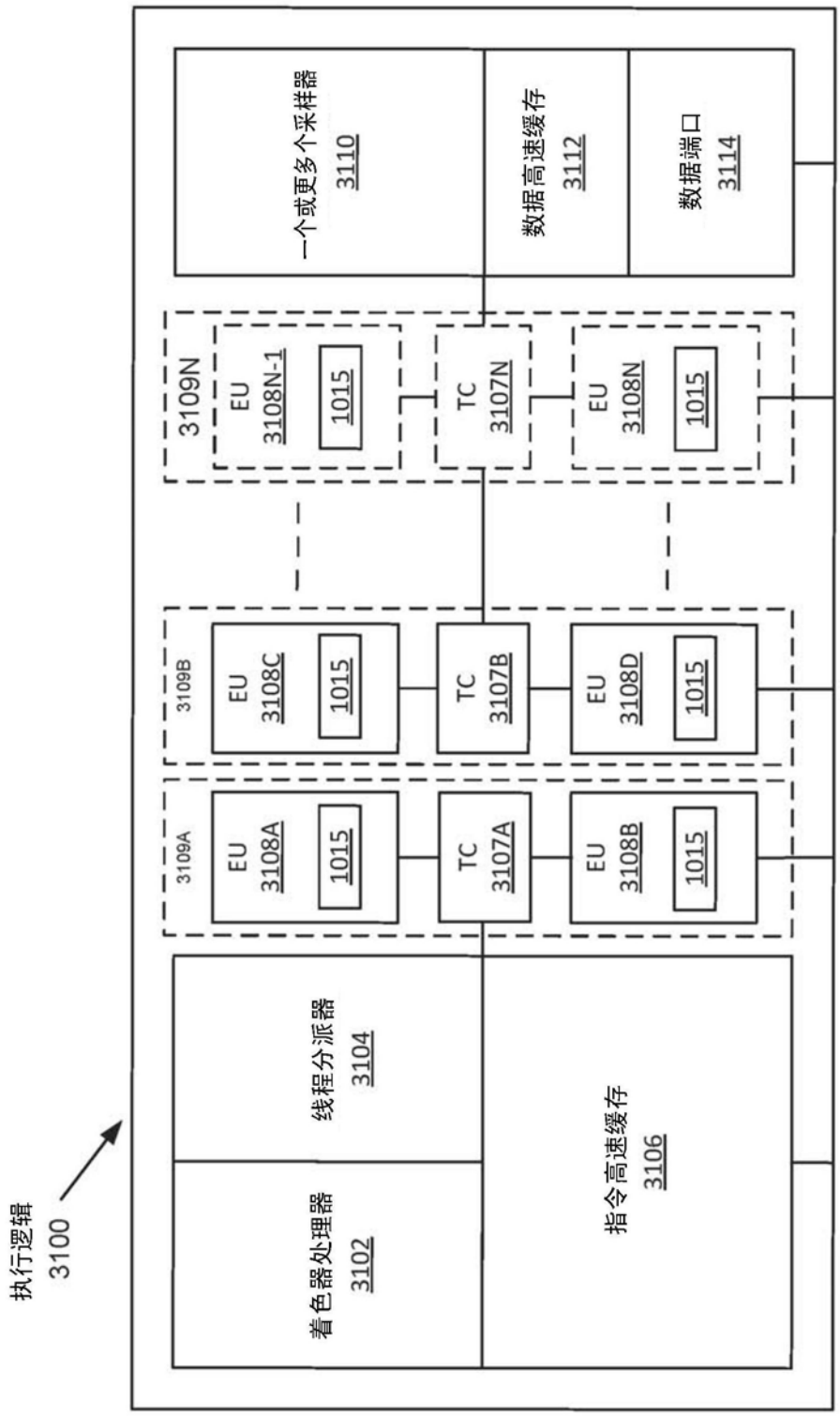

执行逻辑
3100
着色器处理器
3102
线程分派器
3104
指令高速缓存
3106
3109A
EU
3108A
1015
TC
3107A
EU
3108B
1015
3109B
EU
3108C
1015
TC
3107B
EU
3108D
1015
3109N
EU
3108N-1
1015
TC
3107N
EU
3108N
1015
一个或更多个采样器
3110
数据高速缓存
3112
数据端口
3114

图31A

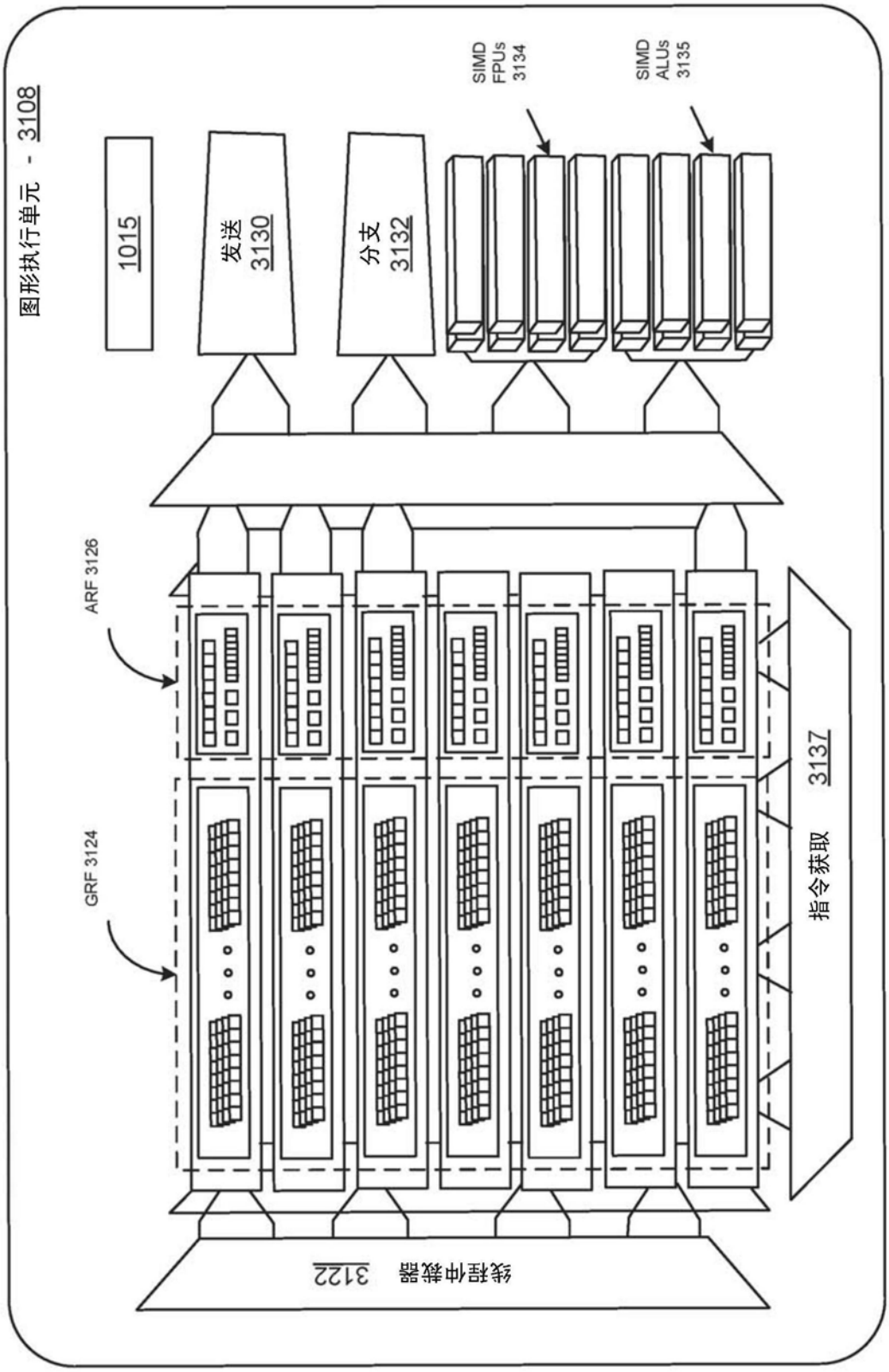

图形执行单元 - 3108
1015
发送 3130
分支 3132
SIMD FPUs 3134
SIMD ALUs 3135
ARF 3126
GRF 3124
指令获取 3137
线程仲裁器 3122

图31B

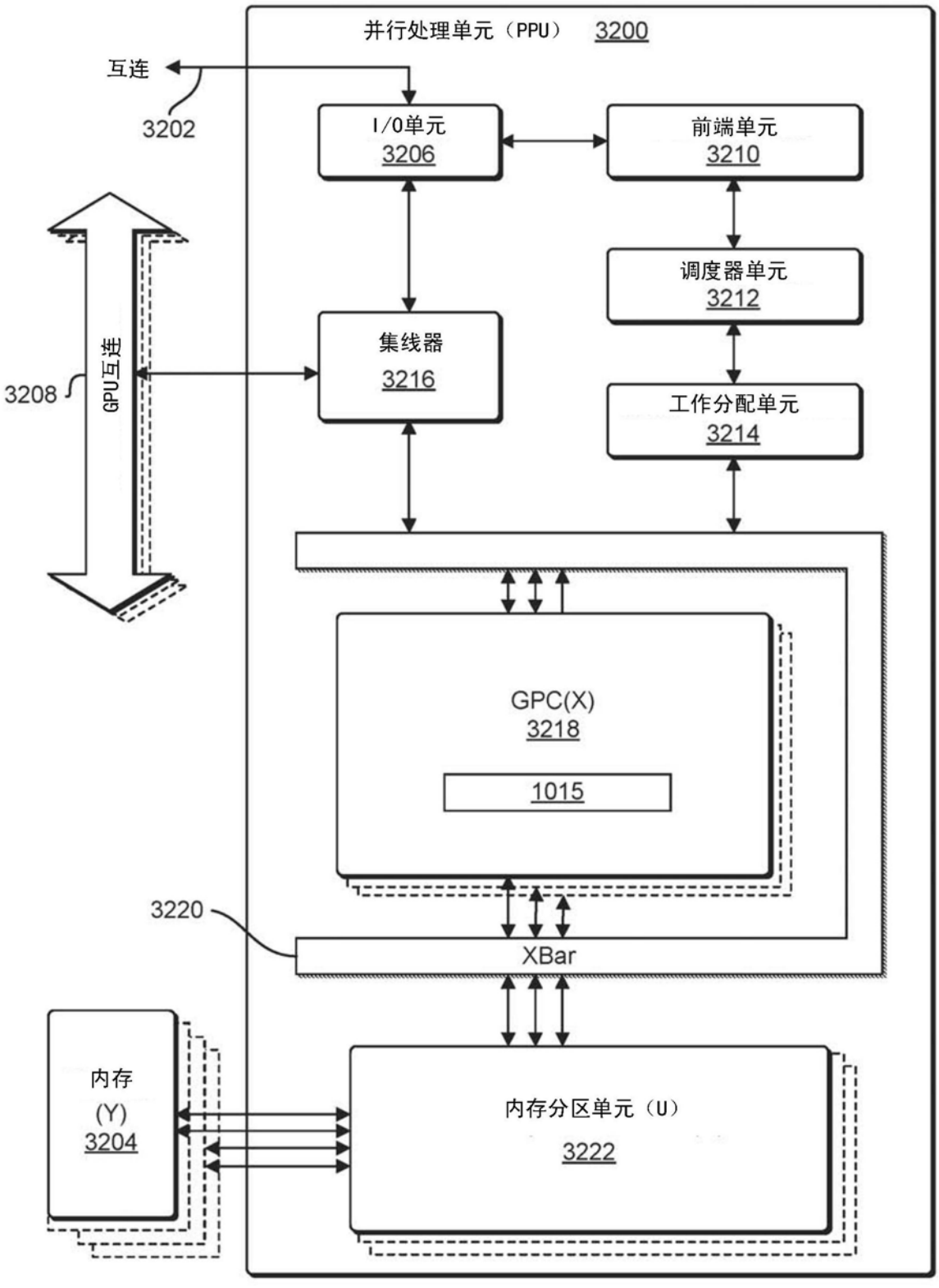

并行处理单元（PPU） 3200
互连
3202
I/O单元
3206
前端单元
3210
调度器单元
3212
集线器
3216
工作分配单元
3214
3208
GPU互连
GPC(X)
3218
1015
3220
XBar
内存
(Y)
3204
内存分区单元（U）
3222

图32

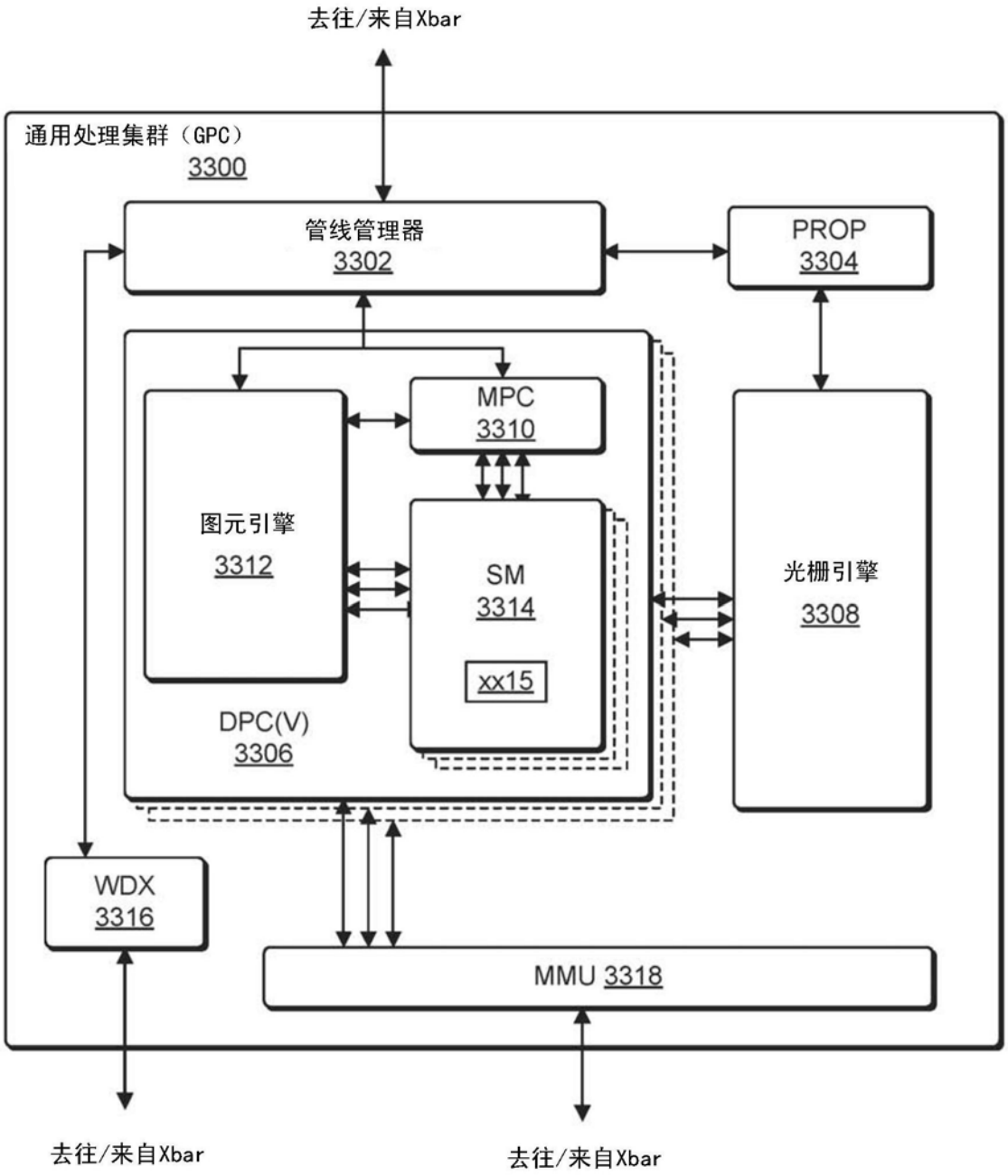

去往/来自Xbar
通用处理集群（GPC）
3300
管线管理器
3302
PROP
3304
MPC
3310
图元引擎
3312
SM
3314
xx15
光栅引擎
3308
DPC(V)
3306
WDX
3316
MMU 3318
去往/来自Xbar
去往/来自Xbar

图33

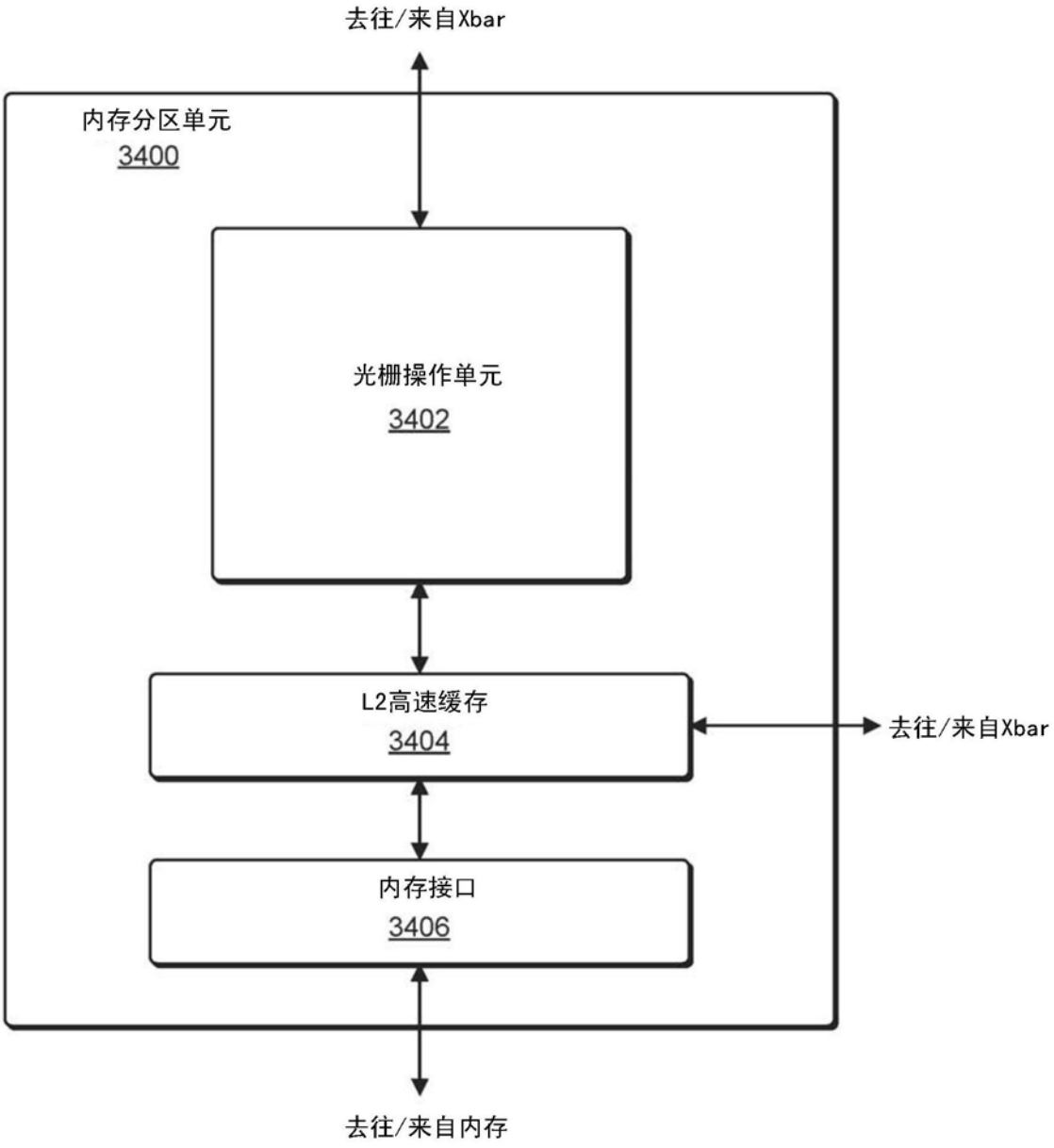

去往/来自Xbar
内存分区单元
3400
光栅操作单元
3402
L2高速缓存
3404
去往/来自Xbar
内存接口
3406
去往/来自内存

图34

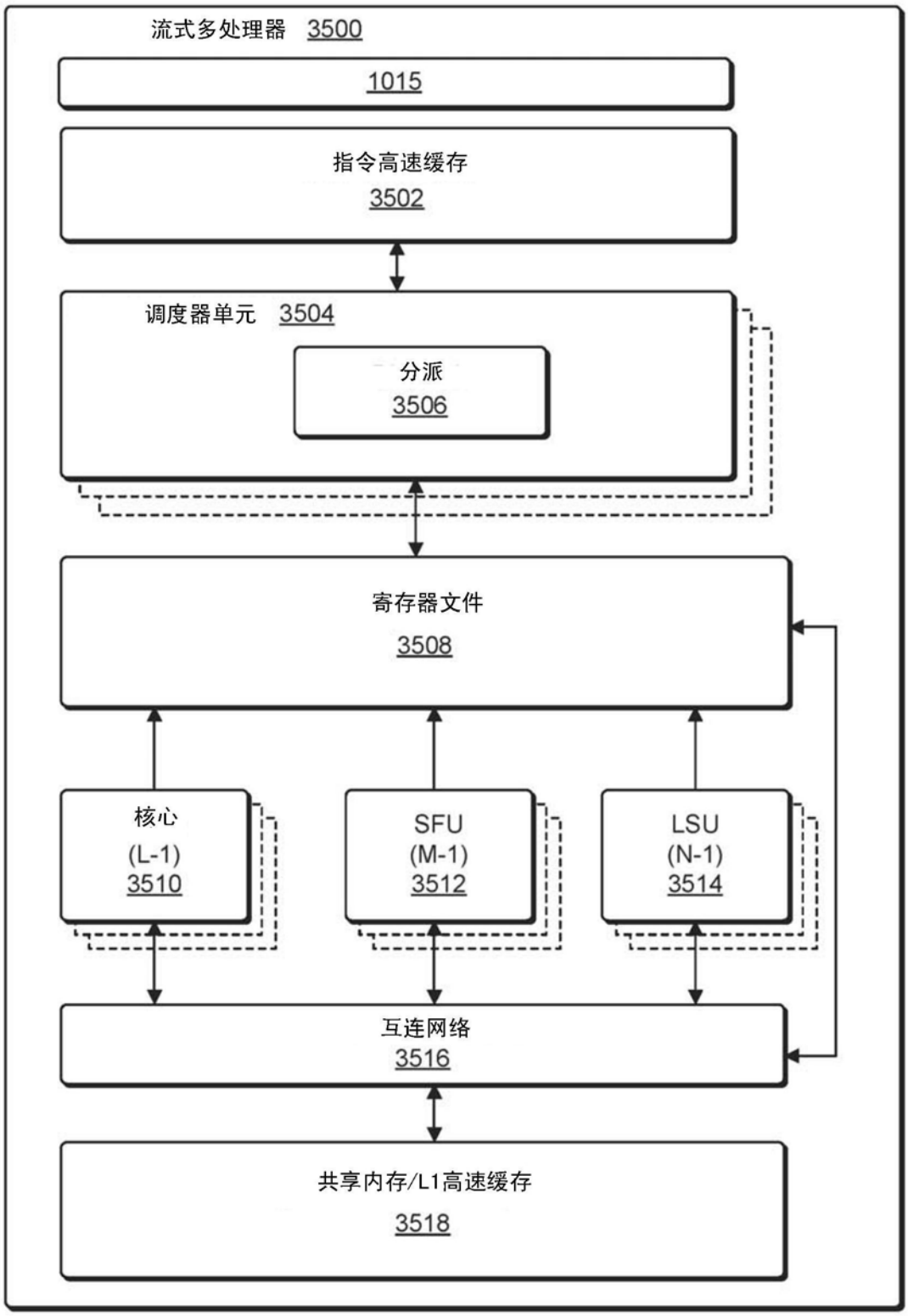

流式多处理器 3500
1015
指令高速缓存
3502
调度器单元 3504
分派
3506
寄存器文件
3508
核心
(L-1)
3510
SFU
(M-1)
3512
LSU
(N-1)
3514
互连网络
3516
共享内存/L1高速缓存
3518

图35