(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 110574011 B
(45)授权公告日 2023.06.27

(21)申请号 201880028856.6
(22)申请日 2018.01.09
(65)同一申请的已公布的文献号
申请公布号 CN 110574011 A
(43)申请公布日 2019.12.13
(30)优先权数据
15/594,512 2017.05.12 US
(85)PCT国际申请进入国家阶段日
2019.10.31
(86)PCT国际申请的申请数据
PCT/US2018/012875 2018.01.09
(87)PCT国际申请的公布数据
WO2018/208334 EN 2018.11.15
(73)专利权人 谷歌有限责任公司
地址 美国加利福尼亚州
(72)发明人 H.朴 A.梅克斯纳 Q.朱 W.R.马克
(74)专利代理机构 北京市柳沈律师事务所 11105
专利代理师 金玉洁
(51)Int.Cl.
G06F 9/54(2006.01)
G06T 1/60(2006.01)
G06F 12/084(2016.01)
G06F 12/0842(2016.01)
(56)对比文件
US 2014040855 A1,2014.02.06
WO 2016171869 A1,2016.10.27
WO 2016171893 A1,2016.10.27
CN 106293958 A,2017.01.04
CN 104081449 A,2014.10.01
US 2013194286 A1,2013.08.01
US 2015055861 A1,2015.02.26
审查员 马聪聪

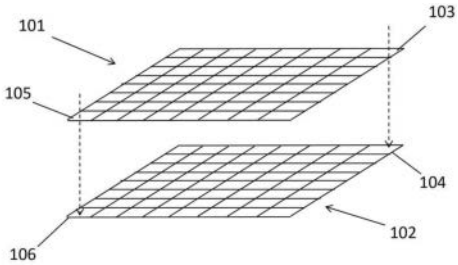
权利要求书3页 说明书20页 附图31页

(54)发明名称

每线缓冲器单元存储器分配的确定

(57)摘要

描述了一种方法。该方法包括模拟图像处理应用软件程序的执行。模拟包括拦截与模拟线缓冲器存储器的内核到内核的通信,该模拟线缓冲器存储器存储并转发从生产内核模型通信到消耗内核模型的图像数据线。该模拟还包括在模拟运行时间内跟踪存储在相应线缓冲器存储器中的相应图像数据量。该方法还包括从跟踪的相应图像数据量中确定对于对应硬件线缓冲器存储器的相应硬件存储器分配。该方法还包括生成用于图像处理器执行图像处理应用软件程序的配置信息。配置信息描述了对于图像处理器的硬件线缓冲器存储器的硬件存储器分配。



100

CN 110574011 B

1.一种包含程序代码的机器可读存储介质，当所述程序代码被计算系统处理时，使得所述计算系统执行方法，所述方法包括：

a)模拟具有多个内核的图像处理应用软件程序的执行，每个内核包括从存储由另外的内核生产的数据的线缓冲器进行读取的加载指令、向存储要由另外的内核消耗的数据的线缓冲器进行写入的存储指令或两者，其中，模拟图像处理应用软件程序的执行包括拦截与存储并转发从生产内核模型传送到消耗内核模型的图像数据线的模拟线缓冲器的内核模型到内核模型通信，以使用相应的多个模拟线缓冲器来模拟多个线缓冲器的操作，所述模拟还包括在模拟运行时间内、通过执行以下操作来跟踪存储在相应模拟线缓冲器中的相应图像数据量：

模拟在多个内核中出现的每个加载指令，包括为模拟加载指令所引用的线缓冲器的相应模拟线缓冲器更新相应读指针，

模拟在多个内核中出现的每个存储指令，包括为模拟存储指令所引用的线缓冲器的相应模拟线缓冲器更新相应写指针；

b)通过为每个模拟线缓冲器计算在模拟期间遇到的模拟线缓冲器的相应读指针和相应写指针之间的相应最大差，从跟踪的相应图像数据量中确定对于对应硬件线缓冲器的相应硬件存储器分配；以及

c)通过基于为模拟线缓冲器计算的相应最大差生成要分配给图像处理器的线缓冲器的相应存储器大小，生成用于图像处理器执行图像处理应用软件程序的配置信息，所述配置信息描述对于所述图像处理器的硬件线缓冲器的硬件存储器分配。

2.根据权利要求1所述的机器可读存储介质，其中，所述模拟还包括施加写策略，所述写策略防止下一个图像数据单元被写入到模拟线缓冲器中，直到消耗图像数据的一个或多个内核模型等待接收下一个图像数据单元。

3.根据权利要求2所述的机器可读存储介质，其中，在生成下一个图像数据单元的生产内核模型处实施写策略。

4.根据权利要求1所述的机器可读存储介质，其中，所述方法还包括如果应用软件程序的模拟执行死锁，则允许违反写策略。

5.根据权利要求1所述的机器可读存储介质，其中，所述内核在硬件图像处理器的不同的处理核心上操作，所述硬件图像处理器包括存储和转发在处理核心之间传递的线组的硬件线缓冲器单元。

6.根据权利要求5所述的机器可读存储介质，其中，所述不同的处理核心包括二维执行通道和二维移位寄存器阵列。

7.根据权利要求1所述的机器可读存储介质，其中，生产内核模型和消耗内核模型包括将图像数据发送到模拟线缓冲器的指令，并且包括从模拟线缓冲器读取图像数据的指令，但是不包括实质上处理图像数据的指令。

8.根据权利要求1所述的机器可读存储介质，其中，图像处理器架构包括耦合到二维移位寄存器阵列的执行阵列。

9.根据权利要求8所述的机器可读存储介质，其中，图像处理器的架构包括线缓冲器、片生成器和/或模板处理器中的至少一个。

10.根据权利要求9所述的机器可读存储介质，所述模板处理器被配置为处理重叠模

板。

11.根据权利要求1所述的机器可读存储介质，其中，数据计算单元包括移位寄存器结构，所述移位寄存器结构具有比执行通道阵列更宽的维度，特别是在执行通道阵列外部有寄存器。

12.一种计算系统，包括：

中央处理单元；

系统存储器；

系统存储器控制器，其位于所述系统存储器和所述中央处理单元之间；

机器可读存储介质，所述机器可读存储介质包含程序代码，当所述程序代码被计算系统处理时，使得所述计算系统执行方法，所述方法包括：

a)模拟具有多个内核的图像处理应用软件程序的执行，每个内核包括从存储由另外的内核生产的数据的线缓冲器进行读取的加载指令、向存储要由另外的内核消耗的数据的线缓冲器进行写入的存储指令或两者，其中，模拟图像处理应用软件程序的执行包括拦截与存储并转发从生产内核模型传送到消耗内核模型的图像数据线的模拟线缓冲器的内核模型到内核模型通信，以使用相应的多个模拟线缓冲器来模拟多个线缓冲器的操作，所述模拟还包括在模拟运行时间内、通过执行以下操作来跟踪存储在相应模拟线缓冲器中的相应图像数据量：

模拟在多个内核中出现的每个加载指令，包括为模拟加载指令所引用的线缓冲器的相应模拟线缓冲器更新相应读指针，

模拟在多个内核中出现的每个存储指令，包括为模拟存储指令所引用的线缓冲器的相应模拟线缓冲器更新相应写指针；

b)通过为每个模拟线缓冲器计算在模拟期间遇到的模拟线缓冲器的相应读指针和相应写指针之间的相应最大差，从跟踪的相应图像数据量中确定对于对应硬件线缓冲器的相应硬件存储器分配；

c)通过基于为模拟线缓冲器计算的相应最大差生成要分配给图像处理器的线缓冲器的相应存储器大小，生成用于图像处理器执行图像处理应用软件程序的配置信息，所述配置信息描述对于所述图像处理器的硬件线缓冲器的硬件存储器分配。

13.根据权利要求12所述的计算系统，其中，所述模拟还包括施加写策略，所述写策略防止下一个图像数据单元被写入到模拟线缓冲器中，直到消耗图像数据的一个或多个内核模型等待接收下一个图像数据单元。

14.根据权利要求13所述的计算系统，其中，在生成下一个图像数据单元的生产内核模型处实施写策略。

15.根据权利要求12所述的计算系统，其中，所述方法还包括如果应用软件程序的模拟执行死锁，则允许违反写策略。

16.根据权利要求12所述的计算系统，其中，图像处理器架构包括耦合到二维移位寄存器阵列的执行阵列。

17.根据权利要求16所述的计算系统，其中，图像处理器的架构包括线缓冲器、片生成器和/或模板处理器中的至少一个。

18.根据权利要求17所述的计算系统，所述模板处理器被配置为处理重叠模板。

19.根据权利要求12所述的计算系统,其中,数据计算单元包括移位寄存器结构,所述移位寄存器结构具有比执行通道阵列更宽的维度,特别是在执行通道阵列外部有寄存器。

20.一种方法,包括:

a)模拟具有多个内核的图像处理应用软件程序的执行,每个内核包括从存储由另外的内核生产的数据的线缓冲器进行读取的加载指令、向存储要由另外的内核消耗的数据的线缓冲器进行写入的存储指令或两者,其中,模拟图像处理应用软件程序的执行包括拦截与储并转发从生产内核模型传送到消耗内核模型的图像数据线的模拟线缓冲器的内核模型到内核模型通信,以使用相应的多个模拟线缓冲器来模拟多个线缓冲器的操作,所述模拟还包括在模拟运行时间内、通过执行以下操作来跟踪存储在相应模拟线缓冲器中的相应图像数据量:

模拟在多个内核中出现的每个加载指令,包括为模拟加载指令所引用的线缓冲器的相应模拟线缓冲器更新相应读指针,

模拟在多个内核中出现的每个存储指令,包括为模拟存储指令所引用的线缓冲器的相应模拟线缓冲器更新相应写指针;

b)通过为每个模拟线缓冲器计算在模拟期间遇到的模拟线缓冲器的相应读指针和相应写指针之间的相应最大差,从跟踪的相应图像数据量中确定对于对应硬件线缓冲器的相应硬件存储器分配;以及

c)通过基于为模拟线缓冲器计算的相应最大差生成要分配给图像处理器的线缓冲器的相应存储器大小,生成用于图像处理器执行图像处理应用软件程序的配置信息,所述配置信息描述对于所述图像处理器的硬件线缓冲器的硬件存储器分配。

21.根据权利要求20所述的方法,其中,所述模拟还包括施加写策略,所述写策略防止下一个图像数据单元被写入到模拟线缓冲器中,直到消耗图像数据的一个或多个内核模型等待接收下一个图像数据单元。

22.根据权利要求21所述的方法,其中,在生成下一个图像数据单元的生产内核模型处实施写策略。

23.根据权利要求20所述的方法,其中,图像处理器架构包括耦合到二维移位寄存器阵列的执行阵列。

24.根据权利要求23所述的方法,其中,图像处理器的架构包括线缓冲器、片生成器和/或模板处理器中的至少一个。

25.根据权利要求24所述的方法,其中,所述模板处理器被配置为处理重叠模板。

26.根据权利要求20所述的方法,其中,数据计算单元包括移位寄存器结构,所述移位寄存器结构具有比执行通道阵列更宽的维度,特别是在执行通道阵列外部有寄存器。

每线缓冲器单元存储器分配的确定

技术领域

[0001] 本发明的领域一般涉及计算科学,并且更具体地,涉及每线(line)缓冲器单元存储器分配的确定。

背景技术

[0002] 图像处理通常涉及对组织成阵列的像素值的处理。这里,空间上组织的二维阵列捕获图像的二维性质(附加维度可以包括时间(例如,二维图像序列)和数据类型(例如,颜色))。在典型场景中,阵列像素值由已经生成静止图像或帧序列以捕捉运动图像的相机提供。传统的图像处理器通常处于两个极端的任一边。

[0003] 第一极端执行作为在通用处理器或通用类处理器(例如,具有矢量指令增强的通用处理器)上执行的软件程序的图像处理任务。尽管第一极端通常提供高度通用的应用软件开发平台,但是它对更精细粒度的数据结构的使用以及相关联的开销(例如,指令提取和解码、片上和片外数据的处理、推测性执行)最终导致在程序代码执行期间每一数据单元消耗更多能量。

[0004] 第二相反的极端是将固定功能硬连线电路应用于更大的数据单元。直接应用于定制设计的电路的更大(相比于更精细粒度的)数据单元的使用大大降低了每一数据单元的功耗。然而,定制设计的固定功能电路的使用通常会导致处理器能够执行的任务集合有限。如此以来,第二极端缺乏广泛通用的编程环境(其与第一极端相关联)。

[0005] 提供高度通用的应用软件开发机会以及提高的每一数据单元的功率效率两者的技术平台仍然是一个理想但缺失的解决方案。

发明内容

[0006] 描述了一种方法。该方法包括模拟图像处理应用软件程序的执行。模拟包括拦截与模拟线缓冲器存储器的内核到内核的通信,该模拟线缓冲器存储器存储并转发从生产内核模型通信到消耗内核模型的图像数据线。该模拟还包括在模拟运行时间内跟踪存储在相应线缓冲器存储器中的相应图像数据量。该方法还包括从跟踪的相应图像数据量中确定对于对应硬件线缓冲器存储器的相应硬件存储器分配。该方法还包括生成用于图像处理器执行图像处理应用软件程序的配置信息。该配置信息描述对于图像处理器的硬件线缓冲器存储器的硬件存储器分配。

附图说明

[0007] 以下描述和附图用于说明本发明的实施例。在附图中:

[0008] 图1示出了模板(stencil)处理器架构的高级视图;

[0009] 图2示出了图像处理器架构的更详细视图;

[0010] 图3示出了可以由图像处理器执行的应用软件程序;

[0011] 图4示出了多个内核模型;

[0012] 图5a和图5b示出了线缓冲器单元模型的写指针和读指针行为；

[0013] 图6a、图6b、图6c、图6d和图6e示出了用于块图像转移的全线组转移模式、虚拟高转移模式和读指针行为；

[0014] 图7示出了用于确定每线缓冲器单元存储器分配的方法；

[0015] 图8a、图8b、图8c、图8d和图8e描绘了将图像数据解析成线组、将线组解析成片以及在具有重叠模板的片上执行的操作；

[0016] 图9a示出了模板处理器的实施例；

[0017] 图9b示出了模板处理器的指令字的实施例；

[0018] 图10示出了模板处理器内的数据计算单元的实施例；

[0019] 图11a、图11b、图11c、图11d、图11e、图11f、图11g、图11h、图11i、图11j和图11k描绘了使用二维移位阵列和执行通道阵列来确定具有重叠模板的一对相邻输出像素值的示例；

[0020] 图12示出了集成的执行通道阵列和二维移位阵列的单元信元(cell)的实施例；

[0021] 图13示出了图像处理器的另一实施例。

具体实施方式

[0022] 1.0独特图像处理器架构

[0023] 如本领域所知，用于执行程序代码的基本电路结构包括执行级(stage)和寄存器空间。执行级包含用于执行指令的执行单元。对于要执行的指令的输入操作数从寄存器空间提供给执行级。根据执行级的指令执行生成的结果(resultant)被写回到寄存器空间。

[0024] 在传统处理器上执行软件线程需要通过执行级顺序执行一系列指令。最常见的，在根据单个输入操作数集生成单个结果的意义上讲，操作是“标量的”。然而，在“矢量”处理器的情况下，由执行级对指令的执行将根据输入操作数的矢量中生成结果矢量。

[0025] 图1示出了独特图像处理器架构100的高级视图，该独特图像处理器架构包括耦合到二维移位寄存器阵列102的执行通道的阵列101。这里，执行通道阵列中的每个执行通道可以被视为包含执行由处理器100支持的指令集所需的执行单元的离散执行级。在各种实施例中，每个执行通道接收相同的指令以在相同的机器周期中执行，使得处理器作为二维单指令多数据(single instruction multiple data,SIMD)处理器操作。

[0026] 每个执行通道在二维移位寄存器阵列102内的对应位置中具有其自己的专用寄存器空间。例如，拐角执行通道103在拐角移位寄存器位置104中具有其自己的专用寄存器空间，拐角执行通道105在拐角移位寄存器位置106中具有其自己的专用寄存器空间，等等。

[0027] 此外，移位寄存器阵列102能够对其内容进行移位，使得每个执行通道能够从其自己的寄存器空间直接对在先前机器周期期间驻留在另一执行通道的寄存器空间中的值进行操作。例如，+1水平移位使得每个执行通道的寄存器空间从其最左边邻居(neighbor)的寄存器空间接收一个值。由于具有沿水平轴在向左和向右两个方向上对值进行移位，以及沿垂直轴在向上和向下两个方向上对值进行移位的能力，所以处理器能够有效地处理图像数据的模板。

[0028] 这里，如本领域所知，模板是用作基本数据单元的图像表面区域的切片。例如，输出图像中特定像素位置的新值可以被计算为输入图像中特定像素位置居中的区域中的像

素值的平均值。例如，如果模板具有3×3像素的维度，则特定像素位置可以对应于3×3像素阵列的中间像素，并且可以计算3×3像素阵列内所有9个像素的平均值。

[0029] 根据图1的处理器100的各种操作实施例，执行通道阵列101的每个执行通道负责计算输出图像中特定位置的像素值。因此，继续上面提到的3×3模板平均示例，在输入像素数据的初始加载和移位寄存器内八个移位操作的协调移位序列之后，执行通道阵列中的每个执行通道会将计算其对应像素位置的平均值所需的所有九个像素值接收到其本地寄存器空间中。也就是说，处理器能够同时处理以例如相邻输出图像像素位置为中心的多个重叠模板。因为图1的处理器架构特别擅长处理图像模板，所以它也可以被称为模板处理器。

[0030] 图2示出了用于具有多个模板处理器202_1至202_N的图像处理器的架构200的实施例。如图2所示，架构200包括多个线缓冲器单元201_1至201_M，其通过网络204(例如，包括片上交换网络、片上环形网络或其他类型的网络的片上网络(network on chip,NOC))与多个模板处理器202_1至202_N和对应的片生成器单元203_1至203_N互连。在实施例中，任何线缓冲器单元201_1至201_M可以通过网络204连接到任何片生成器203_1至203_N和对应的模板处理器202_1至201_N。

[0031] 程序代码被编译并加载到对应的模板处理器202上，以执行软件开发者早先定义的图像处理操作(程序代码也可以加载到模板处理器的相关联的片生成器203上，例如，取决于设计和实施方式)。在至少一些情况下，图像处理管线(pipeline)可以通过将用于第一管线级的第一内核程序加载到第一模板处理器202_1中、将用于第二管线级的第二内核程序加载到第二模板处理器202_2中等来实现。其中第一内核执行管线的第一级的功能，第二内核执行管线的第二级的功能，等等。并且安装了附加的控制流方法来将输出图像数据从管线的一级传递到管线的下一级。

[0032] 在其他配置中，图像处理器可以被实现为具有操作相同的内核程序代码的两个或更多个模板处理器202_1、202_2的并行机器。例如，图像数据的高密度和高数据速率流可以通过跨多个模板处理器扩展帧来处理，该多个模板处理器中的每个模板处理器执行相同的功能。

[0033] 在又其他配置中，基本上内核的任何有向无环图(directed acyclic graph,DAG)都可以通过用它们各自的程序代码内核配置各自的模板处理器并将适当的控制流钩(hook)配置到硬件中以将输出图像从DAG设计中的一个内核引导向下一个内核的输入，而被加载到图像处理器上。

[0034] 作为一般流程，图像数据的帧由宏I/O(input/output，输入/输出)单元205接收，并在逐帧的基础上传递给一个或多个线缓冲器单元201。特定线缓冲器单元将其图像数据帧解析成更小的图像数据区域，称为“线组(line group)”，然后通过网络204将线组传递给特定片生成器。完整或“全部”的单数线组可以包括例如帧的多个连续完整行或列的数据(为了简单起见，本说明书将主要指连续行)。片生成器还将图像数据的线组解析成更小的图像数据区域，称为“片”，并将片呈现给其对应的模板处理器。

[0035] 在图像处理管线或具有单个输入的DAG流的情况下，通常，输入帧被引导到相同的线缓冲器单元201_1，该线缓冲器单元201_1将图像数据解析成线组并将该线组引导到其对应的模板处理器202_1正在执行管线/DAG中的第一内核的代码的片生成器203_1。在模板处理器202_1完成对其处理的线组的操作时，片生成器203_1将输出线组发送到“下游”线缓冲

器单元201_2(在一些使用情况下,输出线组可以被发送回早先已经发送输入线组的相同线缓冲器单元201_1)。

[0036] 表示管线/DAG中的下一级/操作的一个或多个“消耗者”内核在它们相应的其他片生成器和模板处理器(例如,片生成器203_2和模板处理器202_2)上执行,然后从下游线缓冲器单元201_2接收由第一模板处理器202_1生成的图像数据。以这种方式,在第一模板处理器上操作的“生产者”内核将其输出数据转发到在第二模板处理器上操作的“消耗者”内核,其中消耗者内核在生产者内核之后执行与整体管线或DAG的设计一致的下一任务集合。

[0037] 如以上参考图1所暗示的,每个模板处理器202_1至202_N被设计成同时操作图像数据的多个重叠模板。模板处理器的多个重叠模板和内部硬件处理能力有效地确定了片的大小。再者,如上所述,在模板处理器202_1至202_N中的任何一个内,执行通道阵列一致地操作,以同时处理由多个重叠模板覆盖的图像数据表面区域。

[0038] 此外,在各种实施例中,图像数据的片由模板处理器202的对应(例如,本地)片生成器203加载到该模板处理器的二维移位寄存器阵列中。片和二维移位寄存器阵列结构的使用被认为通过将大量数据移动到大量寄存器空间中例如作为单次加载操作,其中此后紧接着由执行通道阵列直接对数据执行处理任务,来有效地提供功耗改进。此外,执行通道阵列和对应的寄存器阵列的使用提供了易于编程/配置的不同模板大小。关于线缓冲器单元、片生成器和模板处理器操作的更多细节将在下文第3.0节中进一步介绍。

[0039] 2.0每线缓冲器单元存储器分配的确定

[0040] 从以上讨论可以理解,硬件平台能够支持无数不同的应用软件程序结构。也就是说,可以支持几乎无限数量的不同且复杂的内核到内核的连接。

[0041] 一个挑战是理解每个线缓冲器单元201_1至201_M应该为任何特定的软件应用分配多少存储器空间。这里,在实施例中,线缓冲器单元中的各种线缓冲器单元,例如,从物理共享的存储器访问已经分配给它们的各自的存储器。如此以来,线缓冲器单元可以更一般地被表征为线缓冲器存储器。在程序执行期间,线缓冲器单元将其例如,从生产内核接收的数据临时存储到其各自的存储器中。在消耗内核准备好接收数据时,线缓冲器单元从其各自的存储器中读取数据并将其转发给消耗内核。

[0042] 在一个或多个或所有线缓冲器单元物理地耦合到相同共享存储器资源的情况下,用于在图像处理器上执行的应用软件程序的配置因而包括定义共享存储器资源的多少存储器容量应该被分别分配给共享存储器资源的每个线缓冲器单元。可能很难确定明确对于每个线缓冲器单元的可工作的存储器分配,尤其是对于具有复杂数据流和相关联的数据依赖性的复杂应用软件程序。

[0043] 图3示出了示例性稍微复杂的应用软件程序(或其一部分)及其在图像处理器上的线缓冲器单元配置的示例。在各种实施方式中,生产内核被允许为不同的消耗内核生成单独的、不同的输出图像流。此外,生产内核还被允许生成由两个或更多个不同内核消耗的单个输出流。最后,在各种实施例中,线缓冲器单元可以仅从一个生产内核接收输入流,但是能够将该流馈送到一个或多个消耗内核。

[0044] 图3的应用软件配置展示了这些配置可能性中的每一个。这里,内核K1为内核K2和K3两者产生第一数据流,并为内核K4产生第二、不同的数据流。内核K1将第一数据流发送给线缓冲器单元304_1,该线缓冲器单元304_1将数据转发给内核K2和K3两者。内核K1还将第

二数据流发送给线缓冲器单元304_2，该线缓冲器单元304_2将数据转发给内核K4。此外，内核K2将数据流发送给内核K4，并且内核K3将数据流发送给内核K4。内核K2将其数据流发送给线缓冲器单元304_3，该线缓冲器单元304_3将数据转发给内核K4。内核K3将其数据流发送给线缓冲器单元304_4，该线缓冲器单元304_4将数据转发给内核K4。

[0045] 这里，要唯一分配给线缓冲器单元304_1至304_4中的每一个的存储器量难以明确计算。将每个这样的存储器分配看作一个队列，如果线缓冲器单元将随时间从生产内核接收大量数据，则所需的存储器量趋于增加。相反，如果线缓冲器单元随时间将从生产内核接收少量数据，则所需的存储器量趋于减少。同样，如果线缓冲器单元将随时间向更多消耗的内核发送少量数据，则所需的存储器量趋于增加，或者，如果线缓冲器单元将随时间向更少消耗的内核发送大量数据，则所需的存储器量趋于减少。

[0046] 线缓冲器单元将随时间从生产者内核接收的数据量可以是以下各项中的任一项的函数：1)生产内核对其自身输入数据的依赖性；2)生产内核生成输出数据的速率，与上述1)的依赖性/速率无关；以及，3)生产内核发送给线缓冲器单元的数据单元的大小。同样，线缓冲器单元将随时间发送的数据量可以是以下各项中的任一项的函数：1)生产内核馈送的消耗内核的数量；2)1)的每个消耗内核准备接收新数据的相应的速率(其可以是消耗内核具有的其他数据依赖性的函数)；以及，3)(多个)消耗内核从线缓冲器单元接收的数据单元的大小。

[0047] 因为，至少对于稍微复杂的应用软件程序结构，各种互相依赖关系和连接速率的复杂性质使得很难明确地计算要分配给每个线缓冲器单元的正确的存储器空间量，所以在各种实施例中，采用启发式方法，该启发式方法在模拟环境中模拟应用软件程序每运行时间的执行，并监控由模拟程序的内部数据流引起的每个线缓冲器单元处的排队的数据量。

[0048] 图4描绘了用于设置模拟环境的图3的应用软件程序的准备过程。在实施例中，通过将每个内核分解到(strip down to)其加载指令和存储指令来创建每个内核的模拟模型。内核的加载指令对应于内核消耗来自线缓冲器单元的输入数据，而内核的存储指令对应于内核产生用于写入到线缓冲器单元中的输出数据。如上所述，内核可以被配置为从例如多个不同的内核/线缓冲器单元接收多个不同的输入流。如此以来，实际内核及其模拟模型内核可以包括多个加载指令(每个不同的输入流一个加载指令)。同样如上所述，内核(并且因此为模拟模型内核)可以被配置为向不同的内核馈送不同的生产流。如此以来，实际内核及其模拟模型内核可以包括多个存储指令。

[0049] 参考图4，模拟模型内核K1示出了一个加载指令(LD_1)和两个存储指令(ST_1和ST_2)，该模拟模型内核K1与图3中内核K1的描述一致，该图3中示出了内核K1接收一个输入流(到图像处理器的输入数据)并提供两个输出流(一个到线缓冲器单元304_1，另一个到线缓冲器单元304_2)。图4还示出了模拟模型内核K2的一个加载指令和一个存储指令，该模拟模型内核K2与图3中内核K2的描述一致，该图3中示出了内核K2从线缓冲器单元304_1接收一个输入流并产生一个输出流到线缓冲器单元304_3。图4还示出了模拟模型内核K3的一个加载指令和一个存储指令，该模拟模型内核K3与图3中内核K3的描述一致，该图3中示出了内核K3从线缓冲器单元304_1接收一个输入流并产生一个输出流到线缓冲器单元304_4。最后，图4示出了具有三个加载指令和一个存储指令的模拟模型内核K4，该模拟模型内核K4与图3中内核K4的描述一致，该图3示出了内核K3从线缓冲器单元304_2接收第一输入流，从线

缓冲器单元304_3接收第二输入流，以及从线缓冲器单元304_4接收第三输入流。内核K4也在图3中示出为生成一个输出流。

[0050] 如图4的循环401_1至401_4所示，模拟模型内核(像实际内核一样)重复循环。也就是说，在执行开始时，内核执行其(多个)加载指令以接收其输入数据，在执行结束时，内核执行其存储指令以根据从其加载指令接收的输入数据产生其输出数据。然后该过程重复。在各种实施例中，每个模拟模型内核还可以包含指示内核为了生成其输出数据而对输入数据执行操作所消耗的时间量(其传播延迟)的值。也就是说，模拟模型内核不允许执行其存储指令，直到其加载指令已经被执行后的一定数量的周期。此外，在各种实施例中，为了减少执行模拟所消耗的时间，内核模型被剥去它们的实际图像处理例程。也就是说，不通过模拟执行实际的图像处理，仅模拟“虚”数据的数据转移。

[0051] 在已经构建了模拟模型内核之后，它们通过与整体应用软件程序的设计/架构一致的线缓冲器单元的相应模拟模型彼此连接。本质上，继续使用图3的应用软件程序作为示例，在模拟环境中构建应用软件程序300的模拟模型，在该模拟环境中，模拟模型包含通过与图3描绘的架构一致的线缓冲器单元304_1至304_4的相应模拟模型互连的图4的内核K1至K4的模拟模型。

[0052] 为了研究每个线缓冲器单元的存储器需求，模拟输入图像数据流(例如，图3的输入图像数据301的模拟)被呈现给应用的模拟模型。然后，应用软件程序的模拟模型用模拟模型内核执行：通过执行它们的(多个)加载指令重复消耗模拟的量的输入数据，通过它们的(多个)存储指令的方式根据接收到的输入数据生成模拟的量的输出数据并重复。

[0053] 这里，每个模拟加载指令可以结合或以其他方式基于存在于原始源内核中的一些输入图像数据格式(诸如输入线组中的线数、最大输入线组速率、输入块的维度/大小、最大输入块速率等)，来确定正在被消耗的输入数据的模拟的量和速率。同样，每个存储指令可以指定或以其他方式基于存在于原始源内核中的一些输出图像格式(诸如输出线组中的线数、最大输出线组速率、输出块的维度/大小、最大输出块速率等)，来确定正在产生输出数据的量和速率。在实施例中，内核模型的加载/存储指令和线缓冲器单元模型对它们的处理反映了应用软件和底层硬件平台的实际握手，其在于：例如，正在产生的图像数据的指定下一部分由生产模型内核的存储指令来标识，以及正在被请求的图像数据的指定下一部分通过消耗模型内核的加载指令来标识。

[0054] 每个线缓冲器单元模型从其各自的生产模型内核接收其各自的模拟输入流，并将其存储到具有例如无限容量的模拟存储器资源中。再者，每事务传输的数据量与生产模型内核的原始源内核的量一致。在由线缓冲器单元模型接收的图像流的(多个)消耗内核执行它们各自的加载指令时，它们向线缓冲器单元模型请求与它们原始源内核的每事务量一致的下一量的输入图像流。作为响应，线缓冲器单元模型从其存储器资源中提供下一个请求的数据单元。

[0055] 在应用软件程序的模型在模拟环境中执行时，每个线缓冲器单元模型的相应存储器状态将随着响应于其(多个)消耗内核的加载指令请求从其读取活动以及响应于其消耗内核的存储指令请求将活动写入到其而衰减和流动。为了最终确定每个线缓冲器单元的存储器容量需求，如图5a和图5b所示，每个线缓冲器单元模拟模型包括写指针和读指针。写指针指定到目前为止有多少来自生产内核模型的输入图像数据已经被写入到线缓冲器单元

模型的存储器中。读指针指定到目前为止已经从线缓冲器单元模型的存储器中读取了多少写入的输入图像数据，以便为来自线缓冲器单元模型的(多个)消耗内核模型的加载指令请求提供服务。

[0056] 图5a的描述表明特定消耗内核每加载指令请求请求X量的图像数据(例如，X可以对应于图像线的指定数量、块大小等)。也就是说，在消耗内核模型已经发送了导向读指针的数据量的情况下，线缓冲器单元将不能为来自消耗内核模型的下一个加载指令请求提供服务，直到写入存储器的数据量达到对应于读指针+X的量(即直到写指针指向等于读指针+X的值)。如图5a中具体描绘的，写指针还没有达到这个水平。如此以来，如果消耗内核已经请求了下一个量(达到读指针+X)，则消耗内核当前被停止，等待来自生产内核的更多输出数据被写入到存储器中。如果消耗内核还没有请求下一个量，则它在技术上还没有停止，并且生产内核仍然有时间来至少提供等于((读指针+X)-写指针)的量，以便在消耗内核请求之前将其写入到存储器中。这个特定事件在图5b中描绘。

[0057] 线缓冲器单元所需的最大存储器容量是在应用软件程序足够长的模拟运行时间执行期间，读指针和写指针之间的最大观察到的差。因此，确定每个线缓冲器单元的存储器容量需要模拟程序的执行达足够数量的周期，同时连续跟踪写指针和读指针之间的差并记录每个新的最大观察到的差。在完成足够数量的执行周期时，与在整个模拟中所观察到的最大差相对应的每个线缓冲器单元模型的其余记录的最大观察到的差对应于每个线缓冲器单元所需的存储器容量。

[0058] 在各种实施例中，为了避免不切实际的情况，在该不切实际的情况中生产者以比其(多个)消耗者可以消耗输出数据快得多的速率生成输出数据，从而导致线缓冲器单元连续写入其存储器并无限制地使用其无限容量，每个内核模型还包括在其存储指令中的每一个处实施的写策略。

[0059] 也就是说，写策略充当对生产内核模型的输出数据写入的线缓冲器单元存储器的量的检查。具体地，在实施例中，直到所有对应的消耗内核都停止(也称为“就绪”)时，才执行生产内核的存储指令。也就是说，只有当每个消耗内核的读指针+X大于生产内核图像流的写指针时，才允许执行生产内核的加载指令。

[0060] 在这种状态下，每个消耗内核都被停止(它们不能为生产内核的图像流的下一个单元执行它们各自的加载指令，因为数据还没有被生产内核产生并写入到线缓冲器单元存储器中)。如此以来，模拟环境的特征在于，生产者不能执行针对引导到特定线缓冲器单元的特定输出流的存储指令，直到消耗来自线缓冲器单元的输出流的每个内核在其将从线缓冲器单元加载流的数据的下一个单元的各自的加载指令处停止。再者，尽管这可能是实际系统的运行时间行为的非典型，但它粗略地对线缓冲器单元处所需的存储器量设置了上限(由最大观察到的写指针到读指针的差与生效的写策略来确定)。

[0061] 例如，如果实际为每个线缓冲器单元分配的存储器量与从最大观察到的写指针到读指针的差所确定的量相同(或略大于)，则实际系统可能永远不会经历任何消耗者停止，因为生产者通常可以自由地随意执行存储指令，直到线缓冲器单元存储器满为止(此时，实际系统中的线缓冲器单元将不允许生产者发送任何更多的数据)。然而，因为在模拟期间，每个生产者直到其所有消耗者停止时才被允许执行其存储指令，所以对于实际系统，通过模拟确定的存储器分配转化为生产者生成新数据以进行消耗大约不晚于其消耗者将停止

的时间。如此以来，平均而言，消耗者不应该在真实系统中停止。以这种方式，模拟结果本质上确定了每个线缓冲器单元处所需的最小存储器容量。

[0062] 理想情况下，在足够数量的模拟运行时间周期之后，可以确定要分配给每个线缓冲器单元的存储器量。然而，在各种模拟运行时间体验中，模拟系统可能会达到完全死锁，在其中没有数据在系统中的任何地方流动。也就是说，系统中的所有内核都不能执行下一个加载指令，因为数据还没有产生，并且所有生产者都不能写入下一个数据量(例如，因为他们自己的加载指令已经停止，并且生产内核没有新的输入来从其创建输出数据)。

[0063] 如果系统达到如上所述的完全死锁，则分析系统状态并找到死锁环路(cycle)。死锁环路是应用的数据流内的一个闭环，它包括等待特定存储执行的特定停止加载，但是该特定存储不能执行，因为它正在等待停止加载执行(注意，停止加载和停止存储不必与彼此直接通信的内核相关联)。

[0064] 例如，在图3的软件程序的模拟模型中，从线缓冲器单元304_4读取数据的K4内核模型的加载指令可以等待由内核K3产生数据。这个特定加载的停止本质上停止了所有内核K4，因此阻止了执行从线缓冲器304_2读取的K4的加载指令。如果线缓冲器304_2的状态使得写指针在读指针+X之前(例如，因为K1在线缓冲器304_2中写入大数据单元)，则写入到线缓冲器304_2中的K1的存储指令将停止，这将停止所有K1，包括写入线缓冲器304_1的存储指令。

[0065] 因为线缓冲器304_1没有被写入，所以K3被停止，这完成了死锁环路的识别分析。也就是说，死锁环路：1)从K1通过线缓冲器单元304_1到内核K3；2)从内核K3通过线缓冲器单元304_4到内核K4；以及，3)从内核K4通过线缓冲器304_2返回内核K1。由于存在这种特定死锁环路，K2也将停止，从而导致整个系统完全死锁(这也在系统内创建了更多的死锁环路)。在实施例中，一旦已经识别出死锁环路，就允许沿该环路的停止的存储指令向前推进一个数据单元，希望该推进将“启动(kick start)”系统以返回操作。例如，如果写入到线缓冲器单元304_1中的内核K1的存储指令向前推进一个数据单元，则这可能足以使得内核K3的停止的加载指令的执行，这继而可能导致系统再次开始操作。

[0066] 在实施例中，仅允许沿着死锁环路的一个停止的存储指令向前推进一个单元。如果推进没有使得系统再次开始运行，则沿着死锁环路选择另一个存储指令进行推进。一次选择一个存储指令进行推进的过程一直持续到系统开始操作，或者在沿着死锁环路的所有存储指令都被允许向前推进一个数据单元之后，保持完全死锁。如果达到后一种条件(系统保持处于完全死锁)，则沿着死锁环路的写入器之一被选择，并被允许自由写入，希望系统能够再次开始操作。如果系统没有开始操作，则沿着死锁环路的另一个存储指令被选择，并被允许自由写入，等等。最终，系统应该开始操作。

[0067] 在各种实施例中，生产/消耗内核模型可以根据不同的传送模式向/从它们各自的线缓冲器单元模型发送/读取图像数据。根据被称为“全线组”的第一模式，在内核模型和线缓冲器单元模型之间传送许多相同宽度的图像数据线。

[0068] 图6a和图6b描绘了全线组模式操作的实施例。如图6a中所观察到的，图像区域600对应于图像数据的全帧或图像数据的全帧的一部分(读者将理解所描绘的矩阵示出了整体图像的不同像素位置)。如图6a所示，并且在内核模型和线缓冲器单元模型之间发送的图像数据的第一转移(例如，第一分组)包含第一组相同宽度的图像线601，其完全延伸穿过被转

移的帧到被转移的帧的部分600。然后，如图6b所示，第二转移包含第二组相同宽度的图像线602，其完全延伸穿过帧或帧的部分600。

[0069] 这里，图6a的组601的转移将把线缓冲器单元模型的写和/或读指针向前推进一个单元。同样，图6b的组602的转移将把线缓冲器单元模型的写和/或读指针向前推进另一个单元。如此以来，上面参考图5a和图5b描述的写指针和读指针行为与全线组模式一致。

[0070] 被称为“虚拟高”的另一种转移模式可以用于转移图像数据块(图像数据的二维表面区域)。这里，如以上参考图1所讨论的以及如以下更详细阐述的，在各种实施例中，整体图像处理器的一个或多个处理核心各自包括二维执行通道阵列和二维移位寄存器阵列。如此以来，处理核心的寄存器空间加载有整个图像数据块(而不仅仅是标量或单个矢量值)。

[0071] 与由处理核心处理的数据单元的二维性质一致，虚拟高模式能够转移图像数据块，如图6c和图6d所描绘。参考图6c，最初，较小高度的全宽度线组被转移611，例如，从第一生产内核模型转移到线缓冲器单元模型。从那时起，至少对于图像区域600，图像数据在较小宽度线组612_1、612_2等中从产生内核模型转移到线缓冲器单元模型。

[0072] 这里，较小宽度线组612_1被转移，例如，在第二生产内核模型中被转移到线缓冲器单元模型事务。然后，如图6d中所观察到的，下一个更小宽度线组612_2被转移，例如，在第三生产内核模型中被转移到线缓冲器单元模型事务。如此以来，线缓冲器单元模型写指针最初递增大的值(以表示全线组611的转移)，但是随后递增较小值(例如，第一较小值以表示较小宽度线组612_1的转移，然后增加下一个较小值以表示较小宽度线组612_2的转移)。

[0073] 如上所述，图6c和图6d示出了将由生产内核模型发送的内容写入到线缓冲器单元模型存储器中。消耗内核模型可以被配置为还接收如上所述的图像数据(在这种情况下，读指针行为与刚刚描述的写指针行为相同)，或者，在图像数据块在线缓冲器存储器中形成时接收图像数据块。

[0074] 也就是说，相对于后者，最初不向消耗内核模型发送第一全线组611。代替的是，在第一较小宽度线组612_1被写入到线缓冲器存储器中之后，最初向消耗内核模型发送对应于第一5×5像素值阵列的数据量，该第一5×5像素值阵列的底边缘由较小宽度线组612_1的底边缘勾勒。然后，在第二较小宽度线组612_2被写入到线缓冲器存储器中之后，向消耗模型发送第二5×5像素值阵列，该第二5×5像素值阵列底边缘由参考标号612_2勾勒。在如上所述向消耗内核模型进行块转移的情况下，如图6e所示，下一个要转移的量包括最近写入到线缓冲器存储器中的较小片段的数据和不久前写入到线缓冲器存储器中的较大片段的数据。

[0075] 图7示出了上述用于确定每线缓冲器单元存储器分配的方法。该方法包括模拟图像处理应用软件程序的执行701。该模拟包括拦截702与线缓冲器存储器模型的内核到内核的通信，该线缓冲器存储器存储并转发从生产内核模型通信到消耗内核模型的图像数据线。该模拟还包括在模拟运行时间内跟踪703存储在相应模拟线缓冲器存储器中的相应图像数据量。该方法还包括从跟踪的相应图像数据量中确定704对于对应硬件线缓冲器存储器的相应硬件存储器分配。

[0076] 从对模拟线缓冲器存储器存储状态的跟踪观察中确定硬件存储器分配可以至少部分地通过考虑彼此而缩放模拟线缓冲器存储器来实施。例如，如果第一模拟线缓冲器存

储器表现出其最大读到写指针差是第二模拟线缓冲器存储器的最大读到写指针差的两倍，则对于第一硬件线缓冲器单元的对应实际硬件存储器分配大约是对于第二硬件线缓冲器单元的对应实际硬件存储器分配的两倍。其余分配将被相应缩放。

[0077] 在已经为应用软件程序确定了存储器分配之后，应用软件程序可以被配置有在目标图像处理器上运行的配置信息，其中，该配置信息通知图像处理器的硬件根据从模拟做出的确定，有多少线缓冲器单元存储器空间被分配给相应的硬件线缓冲器单元。该配置信息还可以包括，例如，指派内核在图像处理器的某个模板处理器上执行，以及向某个硬件线缓冲器单元生产和从某个硬件线缓冲器单元消耗。然后，为应用生成的配置信息语料库可以例如被加载到图像处理器的配置寄存器空间和/或配置存储器资源中，以“设置”图像处理器硬件来执行应用。

[0078] 在各种实施例中，前述线缓冲器单元可以更一般地被表征为存储和转发生产和消耗内核之间的图像数据的缓冲器。也就是说，在各种实施例中，缓冲器不一定需要对线组进行排队。此外，图像处理器的硬件平台可以包括具有相关联的存储器资源的多个线缓冲器单元，并且一个或多个线缓冲器可以被配置为从单个线缓冲器单元操作。也就是说，硬件中的单个线缓冲器单元可以被配置为存储和转发不同的生产/消耗内核对之间的不同的图像数据流。

[0079] 在各种实施例中，可以在模拟期间模拟实际内核，而不是模拟它们的模型。更进一步地，在模拟期间，在内核和线缓冲器单元之间传送的图像数据可以是图像数据的表示(例如，许多线，其中每线被理解为对应于某一数据大小)。为简单起见，术语图像数据应理解为适用于实际图像数据、或图像数据的表示。

[0080] 3.0图像处理器实施方式实施例

[0081] 图8a-e至图12提供了关于上面详细描述的图像处理器和相关联的模板处理器的各种实施例的操作和设计的附加细节。从图2的讨论中回顾，线缓冲器单元将线组馈送到模板处理器的相关联的片生成器，图8a至图8e以高层实施例示出了线缓冲器单元201的解析活动、片生成器单元203的更精细粒度解析活动以及耦合到片生成器单元203的模板处理器702的模板处理活动。

[0082] 图8a描绘了图像数据801的输入帧的实施例。图8a还描绘了模板处理器设计用来在其上操作的三个重叠模板802(每个模板具有3像素×3像素的维度)的轮廓。每个模板分别为其生成输出图像数据的输出像素以纯黑突出显示。为简单起见，三个重叠模板802被描绘为仅在垂直方向重叠。应当认识到，实际上模板处理器可以被设计成在垂直和水平方向两者上都具有重叠的模板。

[0083] 由于模板处理器内的垂直重叠模板802，如图8a所观察到的，所以在帧内存在单个模板处理器可以在其上操作的宽带图像数据。如将在下面更详细讨论的，在实施例中，模板处理器以从左到右的方式在图像数据上处理重叠模板内的数据(然后以从上到下的顺序对下一组线重复)。因此，随着模板处理器继续向前它们的操作，实心黑色输出像素块的数量将向右水平增长。如上所述，线缓冲器单元201负责解析来自传入帧的输入图像数据线组，该输入图像数据线组足以使模板处理器在其上操作达延长数量的即将到来的周期。线组的示例性描绘被示为阴影区域803。在实施例中，线缓冲器单元201可以理解用于向/从片生成器发送/接收线组的不同动态。例如，根据称为“全组”的一种模式，图像数据的完整全宽度

行在线缓冲器单元和片生成器之间传递。根据称为“虚拟高”的第二种模式，线组最初以全宽度行的子集传递。然后，其余的行以较小(小于全宽度)的片段顺序地传递。

[0084] 在输入图像数据的线组803已经由线缓冲器单元定义并被传递到片生成器单元的情况下，片生成器单元进一步将线组解析成更精细的片，这些片更精确地适应模板处理器的硬件限制。更具体地，如下文将更详细描述的，在实施例中，每个模板处理器包括二维移位寄存器阵列。二维移位寄存器阵列本质上将图像数据移位到执行通道阵列的“下方”，其中移位的模式使得每个执行通道对其自己各自模板内的数据进行操作(也就是说，每个执行通道在其自己的信息模板上进行处理，以生成该模板的输出)。在实施例中，片是输入图像数据的表面区域，其“填充”或以其他方式加载到二维移位寄存器阵列中。

[0085] 如下文将更详细描述的，在各种实施例中，实际上存在可以在任何周期移位的多层二维寄存器数据。为了方便起见，本说明书的大部分将简单地使用术语“二维移位寄存器”等来指代具有可以被移位的二维寄存器数据的一个或多个这样的层的结构。

[0086] 因此，如图8b中所观察到的，片生成器从线组803解析初始片804，并将其提供给模板处理器(这里，数据的片对应于通常由参考标号804标识的阴影区域)。如图8c和图8d所观察到的，模板处理器通过在片上以从左到右的方式有效地移动重叠模板802来对输入图像数据的片进行操作。从图8d开始，可以从片内的数据计算输出值的像素数量被耗尽(没有其他像素位置可以具有从片内的信息确定的输出值)。为了简单起见，图像的边界区域被忽略。

[0087] 如图8e中所观察到的，然后，片生成器为模板处理器提供下一个片805以继续操作。注意，模板在下一个片上开始操作时的该模板的初始位置是从第一片上的耗尽点(如先前在图8d中所描绘的)向右的下一个行进。对于新片805，在模板处理器以与处理第一个片相同的方式对新片进行操作时，模板将简单地继续向右移动。

[0088] 注意，由于输出像素位置周围的模板的边界区域，第一片804的数据和第二片805的数据之间存在一些重叠。重叠可以简单地通过片生成器重新发送重叠数据两次来处理。在替代实施方式中，为了将下一个片馈送到模板处理器，片生成器可以继续只向模板处理器发送新数据，并且模板处理器重用来自先前片的重叠数据。

[0089] 图9示出了模板处理器架构900的实施例。如图9中所观察到的，模板处理器包括数据计算单元901、标量处理器902和相关联的存储器903以及I/O单元904。数据计算单元901包括执行通道阵列905、二维移位阵列结构906和与阵列的指定行或列相关联的独立随机存取存储器907。

[0090] I/O单元904负责将从片生成器接收的“输入”数据片加载到数据计算单元901中，并将来自模板处理器的“输出”数据片存储到片生成器中。在实施例中，将片数据加载到数据计算单元901中需要将接收到的片解析成图像数据的行/列，并将图像数据的行/列加载到二维移位寄存器结构906或执行通道阵列的行/列的相应随机存取存储器907中(下面将更详细地描述)。如果片最初被加载到存储器907中，则执行通道阵列905内的各个执行通道然后可以在适当的时候(例如，作为就在对片数据进行操作之前的加载指令)将片数据从随机存取存储器907加载到二维移位寄存器结构906中。在完成将数据片加载到寄存器结构906中时(无论是直接从片数据生成器还是从存储器907)，执行通道阵列905的执行通道对数据进行操作，并最终将完成的数据作为片直接“写回”到片数据生成器，或者到随机存取

存储器907中。如果为后者，则I/O单元904从随机存取存储器907中提取数据，以形成输出片，然后该输出片被转发给片生成器。

[0091] 标量处理器902包括程序控制器909，其从标量存储器903读取模板处理器的程序代码的指令，并将指令发布到执行通道阵列905中的执行通道。在实施例中，单个相同的指令被广播到阵列905内的所有执行通道，以实现来自数据计算单元901的类似SIMD的行为。在实施例中，从标量存储器903读取并发布到执行通道阵列905的执行通道的指令的指令格式包括超长指令字(very-long-instruction-word，VLIW)类型格式，该格式每指令包括超过一个的操作码。在进一步实施例中，VLIW格式包括引导由每个执行通道的ALU执行的数学函数的ALU操作码(如下所述，在实施例中，其可以指定超过一个的传统ALU操作)和存储器操作码(其引导特定执行通道或执行通道集合的存储器操作)。

[0092] 术语“执行通道”是指能够执行指令的一个或多个执行单元的集合(例如，能够执行指令的逻辑电路)。然而，在各种实施例中，执行通道可以包括除执行单元之外的更多类似处理器的功能。例如，除了一个或多个执行单元之外，执行通道还可以包括对接收到的指令进行解码的逻辑电路，或者，在更类似MIMD的设计的情况下，还可以包括提取和解码指令的逻辑电路。关于类似MIMD的方法，尽管在此主要描述了集中式程序控制方法，但是可以在各种替代实施例(例如，包括阵列905的每个执行通道内的程序代码和程序控制器)中实施更分布式的方法。

[0093] 执行通道阵列905、程序控制器909和二维移位寄存器结构906的组合为广泛范围的可编程功能提供了广泛适用/可配置的硬件平台。例如，给定各个执行通道能够执行各种各样的功能并且能够容易地访问任何输出阵列位置附近的输入图像数据的情况下，应用软件开发者能够对具有各种不同功能能力以及维度(例如，模板大小)的内核进行编程。

[0094] 除了充当由执行通道阵列905操作的图像数据的数据存储器之外，随机存取存储器907还可以保存一个或多个查找表。在各种实施例中，一个或多个标量查找表也可以在标量存储器903中被实例化。

[0095] 标量查找涉及根据相同索引将来自相同查找表的相同数据值传递到执行通道阵列905内的每个执行通道。在各种实施例中，上述VLIW指令格式被扩展为还包括标量操作码，该标量操作码将由标量处理器执行的查找操作引导到标量查找表中。指定用于操作码的索引可以是即时操作数，也可以从其他数据存储位置提取。无论如何，在实施例中，从标量存储器内的标量查找表中查找实质上涉及在相同的时钟周期期间向执行通道阵列905内的所有执行通道广播相同的数据值。关于查找表的使用和操作的更多细节将在下面进一步提供。

[0096] 图9b总结了以上讨论的(多个)VLIW指令字实施例。如图9b中所观察到的，VLIW指令字格式包括三个单独的指令的字段：1)由标量处理器执行的标量指令951；2)由执行通道阵列内的相应ALU以SIMD方式广播和执行的ALU指令952；以及，3)以部分SIMD方式广播和执行的存储器指令953(例如，如果沿着执行通道阵列中的相同行的执行通道共享相同的随机存取存储器，则来自不同行中的每行的一个执行通道实际上执行该指令(存储器指令953的格式可以包括标识来自每行的哪个执行通道执行该指令的操作数))。

[0097] 还包括一个或多个即时操作数的字段954。指令951、952、953中的哪一个指令使用哪一个即时操作数信息可以以指令格式来标识。指令951、952、953中的每一个还包括它们

自己各自的输入操作数和结果信息(例如,用于ALU操作的本地寄存器和用于存储器访问指令的本地寄存器和存储器地址)。在实施例中,在执行通道阵列内的执行通道执行指令952、953中的任一个之前,标量指令951由标量处理器执行。也就是说,VLIW字的执行包括执行标量指令951的第一周期,随后是可以执行其他指令952、953的第二周期(注意,在各种实施例中,指令952和953可以并行执行)。

[0098] 在实施例中,由标量处理器执行的标量指令包括发布到片生成器的命令,以从数据计算单元的存储器或2D移位寄存器加载片/将片存储到其中。这里,片生成器的操作可以依赖于线缓冲器单元或其他变量的操作,这些变量阻止对片生成器完成由标量处理器发布的任何命令所需的周期数的预运行时间理解。如此以来,在实施例中,其标量指令951对应于片生成器或以其他方式使得命令被发布到片生成器的任何VLIW字还包括在另外两个指令字段952、953中的无操作(no-operation,NOOP)指令。然后,程序代码进入指令字段952、953的NOOP指令循环,直到片生成器完成向/从数据计算单元的加载/存储。这里,在向片生成器发出命令时,标量处理器可以设置互锁寄存器的一个比特,该互锁寄存器的一个比特由片生成器在命令完成时重置。在NOOP循环期间,标量处理器监控互锁比特的比特。当标量处理器检测到片生成器已经完成其命令时,正常执行再次开始。

[0099] 图10示出了数据计算组件1001的实施例。如图10中所观察到的,数据计算组件1001包括逻辑上位于二维移位寄存器阵列结构1006“上方”的执行通道阵列1005。如上所讨论的,在各种实施例中,由片生成器提供的图像数据片被加载到二维移位寄存器1006中。然后,执行通道对来自寄存器结构1006的片数据进行操作。

[0100] 执行通道阵列1005和移位寄存器结构1006相对于彼此在位置上固定。然而,移位寄存器阵列1006内的数据以策略和协调的方式移位,以使得执行通道阵列中的每个执行通道处理数据内的不同模板。如此以来,每个执行通道确定正在生成的输出片中不同像素的输出图像值。从图10的架构中,应该清楚,重叠模板不仅垂直布置,还水平布置,因为执行通道阵列1005包括垂直相邻的执行通道以及水平相邻的执行通道。

[0101] 数据计算单元1001的一些显著的架构特征包括移位寄存器结构1006,其具有比执行通道阵列1005更宽的维度。也就是说,在执行通道阵列1005之外有寄存器的“晕圈(halo)”1009。尽管晕圈1009被示出存在于执行通道阵列的两侧,但是取决于实施方式,晕圈可以存在于执行通道阵列1005的少(一个)或多(三个或四个)侧。晕圈1009用于在数据在执行通道1005“下方”移位时,为溢出到执行通道阵列1005边界之外的数据提供“溢出”空间。作为简单的情况,当模板最左边的像素被处理时,以执行通道阵列1005的右边缘为中心的5×5模板将需要更靠右的四个晕圈寄存器位置。为了便于绘图,图10示出了晕圈右侧的寄存器仅具有水平移位连接,晕圈底部的寄存器仅具有垂直移位连接,而在标称(nominal)实施例中,任一侧(右侧、底部)的寄存器将具有水平和垂直连接。在各种实施例中,晕圈区域不包括用于执行图像处理指令的对应执行通道逻辑(例如,不呈现ALU)。然而,各个存储器存取单元(M)呈现于每个晕圈区域位置中,使得各个晕圈寄存器位置可以独自地从存储器加载数据并将数据存储到存储器。

[0102] 附加溢出空间由随机存取存储器1007提供,该随机存取存储器1007耦合到阵列中的每行和/或每列,或其部分(例如,随机存取存储器可以被指派给执行通道阵列的“区域”,其按行跨越4个执行通道,以及按列跨越2个执行通道。为简单起见,应用程序的其余部分将

主要参考基于行和/或列的分配方案)。这里,如果执行通道的内核操作要求它处理二维移位寄存器阵列1006外部的像素值(这是一些图像处理例程可能需要的),则图像数据平面能够进一步溢出,例如,从晕圈区域1009溢出到随机存取存储器1007中。例如,考虑6×6模板,其中硬件在执行通道阵列右边缘的执行通道右边仅包括四个存储元素的晕圈区域。在这种情况下,需要将数据进一步移位到晕圈1009右边缘的右边,以完全处理模板。然后,移位到晕圈区域1009之外的数据将溢出到随机存取存储器1007。图9的随机存取存储器1007和模板处理器的其他应用将在下面进一步提供。

[0103] 图11a至图11k展示了以图像数据在如上所述的执行通道阵列“下方”的二维移位寄存器阵列内移位的方式的工作示例。如图11a中所观察到的,二维移位阵列的数据内容在第一阵列1107中描绘,并且执行通道阵列由帧1105描绘。并且,简单描绘了执行通道阵列内的两个相邻执行通道1110。在该简化描绘1110中,每个执行通道包括寄存器R1,该寄存器R1可以接受来自移位寄存器的数据,接受来自ALU输出的数据(例如,用作跨周期的累加器),或将输出数据写入到输出目的地中。

[0104] 在二维移位阵列中。每个执行通道还具有在其“下方”的在本地寄存器R2中可用的内容。因此,R1是执行通道的物理寄存器,而R2是二维移位寄存器阵列的物理寄存器。执行通道包括可以对R1和/或R2提供的操作数进行运算的ALU。如下文将更详细描述的,在实施例中,移位寄存器实际上是用每一阵列位置的多个(深度)存储/寄存器元件来实施的,但是移位活动限于一个存储元件平面(例如,每周期只有一个存储元件平面可以移位)。图11a至图11k将这些更深的寄存器位置之一描绘为用于存储来自相应执行通道的结果X。为了便于说明,更深的结果寄存器被绘制在其对应寄存器R2的旁边,而不是下方。

[0105] 图11a至图11k集中于两个模板的计算,这两个模板的中心位置与执行通道阵列内所描绘的执行通道位置对1111对齐。为了便于说明,该执行通道对1110被绘制为水平邻居,而实际上,根据以下示例,它们是垂直邻居。

[0106] 如最初在图11a中所观察到的,执行通道以它们的中心模板位置为中心。图11b示出了由两个执行通道执行的目标代码。如图11b中所观察到的,两个执行通道的程序代码使得移位寄存器阵列内的数据向下移位一个位置,以及向右移位一个位置。这将两个执行通道与其各自的模板的左上角对齐。然后,程序代码将位于(R2中)各自位置中的数据加载到R1。

[0107] 如图11c中所观察到的,程序代码接下来使得该执行通道对将移位寄存器阵列内的数据向左移位一个单位,这使得每个执行通道各自位置右边的值被移位到每个执行通道的位置。然后将R1的值(先前的值)与已经移位到执行通道位置(R2)的新值相加。结果写入到R1中。如在图11d中所观察到的,重复与上面针对图11c描述的相同的过程,这使得结果R1现在包括上执行通道中的值A+B+C和下执行通道中的值F+G+H。此时,两个执行通道都已经处理了它们各自模板的上面的行。注意,溢出到执行通道阵列左侧的晕圈区域(如果在左侧存在晕圈区域),或者如果在执行通道阵列左侧不存在晕圈区域则溢出到随机存取存储器。

[0108] 如图11e所观察到的,程序代码接下来使得移位寄存器阵列内的数据上移一个单元,这使得两个执行通道与它们各自模板的中间行的右边缘对齐。两个执行通道的寄存器R1当前包括模板顶行和中间行最右边值的总和。图11f和图11g展示了在两个执行通道模板的中间行上向左移动的持续过程。累积相加继续进行,使得在图11g的处理结束时,两个执

行通道都包括它们各自模板的顶行和中间行的值的总和。

[0109] 图11h示出了将每个执行通道与其对应模板的最低行对齐的另一个移位。图11i和图11j示出了在两个执行通道的模板过程中继续移位以完成处理。图11k示出了将每个执行通道与其在数据阵列中的正确位置对齐并将结果写入其中的附加移位。

[0110] 在图11a-图11k的示例中，注意，移位操作的目标代码可以包括指令格式，该指令格式标识以(X,Y)坐标表示的移位的方向和幅度。例如，向上移位一个位置的目标代码可以用目标代码表示为SHIFT 0,+1。作为另一示例，向右移位一个位置可以用目标代码表示为SHIFT+1,0。在各种实施例中，也可以在目标代码中指定更大幅度的移位(例如，SHIFT 0,+2)。这里，如果2D移位寄存器硬件仅支持每周期移位一个位置，则指令可以被机器解释为需要多个周期执行，或者，2D移位寄存器硬件可以被设计为支持每周期移位超过一个位置。后面的实施例将在下面进一步详细描述。

[0111] 图12示出了用于执行通道和对应移位寄存器结构的单元信元的另一个更详细的描述(在各种实施例中，晕圈区域中的寄存器不包括对应执行通道，而是包括存储器单元)。在实施例中，执行通道以及与执行通道阵列中的每个位置相关联的寄存器空间通过在执行通道阵列的每个节点处实例化图12中所观察到的电路来实施。如图12中所观察到的，单元信元包括耦合到由四个寄存器R2至R5组成的寄存器文件1202的执行通道1201。在任何周期期间，执行通道1201可以从寄存器R1至R5中的任何一个读取或向其写入。对于需要两个输入操作数的指令，执行通道可以从R1至R5中的任何一个检索两个操作数。

[0112] 在实施例中，通过允许在单个周期期间通过输出多路复用器1203将寄存器R2至R4中的任一个(仅一个)的内容“移出”到其邻居的寄存器文件中的任一个，以及通过如果其邻居通过输入多路复用器1204使得邻居之间的移位在相同方向上(例如，所有执行通道向左移位，所有执行通道向右移位等等)则使寄存器R2至R4中的任一个(仅一个)的内容替换为从对应的寄存器“移入”的内容，来实现二维移位寄存器结构。尽管相同寄存器使其内容移出并用被替换为在相同周期移入的内容可能是常见的，但是多路复用器布置1203、1204允许在相同周期期间相同寄存器文件内的不同移位源和移位目标寄存器。

[0113] 如图12所描绘的，注意，在移位序列期间，执行通道将内容从其寄存器文件1202移出到其左、右、顶和底邻居中的每一个。结合相同的移位序列，执行通道也将内容从其左、右、顶和底邻居中的特定的一个移位到其寄存器文件中。再者，对于所有执行通道，移出目标和移入源应该与相同的移位方向一致(例如，如果移出到向右邻居，则应该从左邻居移入)。

[0114] 尽管在一个实施例中，每周期每执行通道只允许移位一个寄存器的内容，但是其他实施例可以允许超过一个寄存器的内容被移入/移出。例如，如果图12中所观察到的多路复用器电路1203、1204的第二实例被结合到图12的设计中，则两个寄存器的内容可以在相同周期期间移出/移入。当然，在每周期只允许移位一个寄存器的内容的实施例中，通过消耗更多的时钟周期以用于在数学运算之间的移位，可以在数学运算之间发生来自多个寄存器的移位(例如，通过在数学运算之间消耗两个移位运算，可以在数学运算之间移位两个寄存器的内容)。

[0115] 如果少于在移位序列期间被移出的执行通道的寄存器文件的内容的全部，则注意每个执行通道的未移出寄存器的内容保持不变(不移位)。如此以来，跨移位周期，任何未被

替换为移入内容的未移位内容都将存留在执行通道的本地。在每个执行通道中所观察到的存储器单元(“M”)用于从/向与执行通道阵列内的执行通道的行和/或列相关联的随机存取存储器空间加载/存储数据。这里,M单元充当标准的M单元,因为它通常用于加载/存储不能从/向执行通道自己的寄存器空间加载/存储的数据。在各种实施例中,M单元的主要操作是将来自本地寄存器的数据写入到存储器中,并且从存储器读取数据并将其写入本地寄存器。

[0116] 关于由硬件执行通道1201的ALU单元支持的ISA操作码,在各种实施例中,由硬件ALU单元支持的数学操作码包括,例如,ADD、SUB、MOV、MUL、MAD、ABS、DIV、SHL、SHR、MIN/MAX、SEL、AND、OR、XOR、NOT。如上所述,存储器访问指令可以由执行通道1201执行,以从/向其相关联的随机访问存储器提取/存储数据。此外,硬件执行通道1201支持移位操作指令(右、左、上、下),以在二维移位寄存器结构内移位数据。如上所述,程序控制指令主要由模板处理器的标量处理器执行。

[0117] 4.0实施方式实施例

[0118] 需要指出的是,上述各种图像处理器架构特征不一定局限于传统意义上的图像处理,并且因此可以应用于可能(或可能不)使得图像处理器被重新表征的其他应用。例如,如果上述各种图像处理器架构特征中的任何一个将用于动画的创建和/或生成和/或渲染,而不是实际相机图像的处理,则图像处理器可以被表征为图形处理单元。此外,上述图像处理器架构特征可以应用于其他技术应用,诸如视频处理、视觉处理、图像识别和/或机器学习。以这种方式应用,图像处理器可以与更通用的处理器(例如,作为计算系统的CPU或计算系统的CPU的一部分)集成(例如,作为其协处理器),或者可以是计算系统内的独立处理器。

[0119] 上面讨论的硬件设计实施例可以体现在半导体芯片内和/或作为最终针对半导体制造工艺的电路设计的描述。在后一种情况下,这种电路描述可以采取(例如,VHDL或Verilog)寄存器传输级(register transfer level,RTL)电路描述、栅极级电路描述、晶体管级电路描述或掩模描述或其各种组合的形式。电路描述通常体现在计算机可读存储介质(诸如CD-ROM或其他类型的存储技术)上。

[0120] 从前面的章节可以认识到,如上所述的图像处理器可以体现在计算机系统上的硬件中(例如,作为处理来自手持设备的相机的数据的手持设备的片上系统(System on Chip,SOC)的一部分)。在图像处理器被体现为硬件电路的情况下,注意,由图像处理器处理的图像数据可以直接从相机接收。这里,图像处理器可以是离散相机的一部分,或者是具有集成相机的计算系统的一部分。在后者的情况下,图像数据可以直接从相机或从计算系统的系统存储器接收(例如,相机将其图像数据发送到系统存储器而不是图像处理器)。还要注意,前面章节中描述的许多特征可能适用于图形处理器单元(其渲染动画)。

[0121] 图13提供了计算系统的示例性描述。下面描述的计算系统的许多组件可应用于具有集成相机和相关联的图像处理器的计算系统(例如,诸如智能手机或平板电脑的手持设备)。普通技术人员将能够容易地在两者之间进行划分。此外,图13的计算系统还包括高性能计算系统的许多特征,诸如工作站或超级计算机。

[0122] 如图13中所观察到的,基本计算系统可以包括中央处理单元(Central Processing Unit,CPU)1301(其可以包括例如多个通用处理核心1315_1至1315_N和设置在多核处理器或应用处理器上的主存储器控制器1317)、系统存储器1302、显示器1303(例如,

触摸屏、平板)、本地有线点对点链接(例如,USB)接口1304、各种网络I/O功能1305(诸如以太网接口和/或蜂窝调制解调器子系统)、无线局域网(例如,WiFi)接口1306、无线点对点链路(例如,蓝牙)接口1307和全球定位系统接口(Global Position System,GPS)1308、各种传感器1309_1至1309_N、一个或多个相机1310、电池1311、电源管理控制单元1312、扬声器和麦克风1313以及音频编码器/解码器1314。

[0123] 应用处理器或多核处理器1350可以在其CPU 1301内包括一个或多个通用处理核心1315、一个或多个图形处理单元1316、存储器管理功能1317(例如,存储器控制器)、I/O控制功能1318和图像处理单元(Image Processing Unit,IPU)1319。通用处理核心1315通常执行计算系统的操作系统和应用软件。图形处理单元(Graphics Processing Unit,GPU)1316通常执行图形密集型功能,以例如生成呈现在显示器1303上的图形信息。存储器控制功能1317与系统存储器1302接口,以向/从系统存储器1302写入/读取数据。电源管理控制单元1312通常控制系统1300的功耗。

[0124] 图像处理单元1319可以根据前面章节中详细描述的任何图像处理单元实施例来实施。可替换地或组合地,IPU 1319可以耦合到GPU 1316和CPU 1301中的任一个或两者,作为其协处理器。此外,在各种实施例中,GPU 1316可以用上面详细描述的任何图像处理器特征来实施。图像处理单元1319可以被配置有如上详细描述的应用软件。此外,诸如图13的计算系统的计算系统可以执行程序代码,该程序代码模拟如上所述的图像处理应用软件程序,从而可以确定对于相应线缓冲器单元的相应存储器分配。

[0125] 触摸屏显示器1303、通信接口1304-1307、GPS接口1308、传感器1309、相机1310和扬声器/麦克风编解码器1313、1314中的每一个都可以被视为相对于整体计算系统的各种形式的I/O(输入和/或输出),该整体计算系统在适当的情况下,包括集成外围设备(例如,一个或多个相机1310)。根据实施方式,这些I/O组件中的各种组件可以集成在应用处理器/多核处理器1350上,或者可以位于裸芯外或者应用处理器/多核处理器1350的封装之外。在实施例中,一个或多个相机1310包括能够测量相机和其视野中的物体之间的深度的深度相机。

[0126] 在应用处理器或其他处理器的通用CPU核心(或具有执行程序代码的指令执行管线的其他功能块)上执行的应用软件、操作系统软件、设备驱动软件和/或固件可以执行上述任何功能。

[0127] 本发明的实施例可以包括如上所述的各种过程。这些过程可以体现在机器可执行指令中。这些指令可以用于使得通用或专用处理器执行某些过程。可替换地,这些过程可以由包含用于执行过程的硬连线和/或可编程逻辑的特定硬件组件来执行,或者由可编程计算机组件和定制硬件组件的任意组合来执行。

[0128] 本发明的元件也可以被提供为用于存储机器可执行指令的机器可读介质。机器可读介质可以包括但不限于软盘、光盘、CD-ROM和磁光盘、闪存、只读存储器(Read-Only-Memory,ROM)、随机存取存储器(Random-Access-Memory,RAM)、EPROM(Electrically Programmable Read-Only-Memory,电可编程序只读存储器)、EEPROM(Electrically Erasable Programmable Read-Only-Memory,电可擦除可编程只读存储器)、磁卡或光卡、传播介质或适合于存储电子指令的其他类型的介质/机器可读介质。例如,本发明可以作为计算机程序下载,该计算机程序可以经由通信链路(例如,调制解调器或网络连接)通过体

现在载波或其他传播介质中的数据信号从远程计算机(例如,服务器)传输到请求计算机(例如,客户端)。

[0129] 在前述说明书中,已经参考本发明的具体示例性实施例描述了本发明。然而,显而易见的是,在不脱离所附权利要求中阐述的本发明的更广泛的精神和范围的情况下,可以对其进行各种修改和变化。因此,说明书和附图被认为是说明性的,而不是限制性的。

[0130] 下面描述了一些示例。

[0131] 示例1:一种包含程序代码的机器可读存储介质,当该程序代码被计算系统处理时,使得计算系统执行一种方法,该方法包括:

[0132] a)模拟图像处理应用软件程序的执行,该模拟包括拦截与模拟线缓冲器存储器的内核到内核通信,该模拟线缓冲器存储器存储并转发从生产内核模型通信到消耗内核模型的图像数据线,该模拟还包括在模拟运行时间内跟踪存储在相应线缓冲器存储器中的相应图像数据量;以及,

[0133] b)从跟踪的相应图像数据量中确定对于对应硬件线缓冲器存储器的相应硬件存储器分配;

[0134] c)生成用于图像处理器执行图像处理应用软件程序的配置信息,该配置信息描述对于图像处理器的硬件线缓冲器存储器的硬件存储器分配。

[0135] 示例2:根据示例1的机器可读存储介质,其中,该跟踪还包括跟踪模拟线缓冲器存储器写指针和模拟线缓冲器存储器读指针之间的差。

[0136] 示例3:根据示例1或2的机器可读存储介质,其中,该确定基于模拟线缓冲器存储器写指针和模拟线缓冲器存储器读指针之间的最大观察到的差。

[0137] 示例4:根据前述示例中至少一个示例的机器可读存储介质,其中,该模拟还包括施加写策略,该写策略防止下一个图像数据单元被写入到模拟线缓冲器存储器中,直到消耗图像数据的一个或多个内核模型等待接收下一个图像数据单元。

[0138] 示例5:根据前述示例中至少一个示例的机器可读存储介质,其中,在生成下一个图像数据单元的生产内核模型处实施写策略。

[0139] 示例6:根据前述示例中至少一个示例的机器可读存储介质,其中,该方法还包括如果应用软件程序的模拟执行死锁,则允许违反写策略。

[0140] 示例7:根据前述示例中至少一个示例的机器可读存储介质,其中,该内核在硬件图像处理器的不同处理核心上操作,该硬件图像处理器包括存储和转发在处理核心之间传递的线组的硬件线缓冲器单元。

[0141] 示例8:根据前述示例中至少一个示例的机器可读存储介质,其中,该不同的处理核心包括二维执行通道和二维移位寄存器阵列。

[0142] 示例9:根据前述示例中至少一个示例的机器可读存储介质,其中,生产内核模型和消耗内核模型包括将图像数据发送到模拟线缓冲器存储器的指令,并且包括从模拟线缓冲器存储器读取图像数据的指令,但是不包括实质上处理图像数据的指令。

[0143] 示例10:根据前述示例中至少一个示例的机器可读存储介质,其中,图像处理器架构包括耦合到二维移位寄存器阵列的执行阵列。

[0144] 示例11:根据前述示例中至少一个示例的机器可读存储介质,其中,图像处理器的架构包括线缓冲器、片生成器和/或模板处理器中的至少一个。

[0145] 示例12:根据示例11的机器可读存储介质,其中,该模板处理器被配置为处理重叠模板。

[0146] 示例13:根据前述示例中的至少一个示例的机器可读存储介质,其中,数据计算单元包括移位寄存器结构,该移位寄存器结构具有比执行通道阵列更宽的维度,特别是在执行通道阵列外部有寄存器。

[0147] 示例14:一种计算系统,包括:

[0148] 中央处理单元;

[0149] 系统存储器;

[0150] 系统存储器控制器,其位于系统存储器和中央处理单元之间;

[0151] 机器可读存储介质,该机器可读存储介质包含程序代码,当该程序代码被计算系统处理时,使得计算系统执行方法,该方法包括:

[0152] a)模拟图像处理应用软件程序的执行,该模拟包括拦截与模拟线缓冲器存储器的内核到内核通信,该模拟线缓冲器存储器存储并转发从生产内核模型通信到消耗内核模型的图像数据线,该模拟还包括在模拟运行时间内跟踪存储在相应线缓冲器存储器中的相应图像数据量;以及,

[0153] b)从跟踪的相应图像数据量中确定对于对应硬件线缓冲器存储器的相应硬件存储器分配;

[0154] c)生成用于图像处理器执行图像处理应用软件程序的配置信息,该配置信息描述对于图像处理器的硬件线缓冲器存储器的硬件存储器分配。

[0155] 示例15:根据示例14的计算系统,其中,该跟踪还包括跟踪模拟线缓冲器存储器写指针和模拟线缓冲器存储器读指针之间的差。

[0156] 示例16:根据示例14或15的计算系统,其中,该确定基于模拟线缓冲器存储器写指针和模拟线缓冲器存储器读指针之间的最大观察到的差。

[0157] 示例17:根据示例14至16中的至少一个示例的计算系统,其中,该模拟还包括施加写策略,该写策略防止下一个图像数据单元被写入到模拟线缓冲器存储器中,直到消耗图像数据的一个或多个内核模型等待接收下一个图像数据单元。

[0158] 示例18:根据示例14至17中至少一个示例的计算系统,其中,在生成下一个图像数据单元的生产内核模型处实施写策略。

[0159] 示例19:根据示例14至18中至少一个示例的计算系统,其中,该方法还包括如果应用软件程序的模拟执行死锁,则允许违反写策略。

[0160] 示例20:根据示例14至19中至少一个示例的计算系统,其中,图像处理器架构包括耦合到二维移位寄存器阵列的执行阵列。

[0161] 示例21:根据示例14至20中至少一个示例的计算系统,其中,图像处理器的架构包括线缓冲器、片生成器和/或模板处理器中的至少一个。

[0162] 示例22:根据实例21的计算系统,其中,该模板处理器被配置为处理重叠模板。

[0163] 示例23:根据示例14至22中的至少一个示例的计算系统,其中,数据计算单元包括移位寄存器结构,该移位寄存器结构具有比执行通道阵列更宽的维度,特别是在执行通道阵列外部有寄存器。

[0164] 示例24:一种方法,包括:

[0165] a)模拟图像处理应用软件程序的执行,该模拟包括拦截与模拟线缓冲器存储器的内核到内核通信,该模拟线缓冲器存储器存储并转发从生产内核模型通信到消耗内核模型的图像数据线,该模拟还包括在模拟运行时间内跟踪存储在相应线缓冲器存储器中的相应图像数据量;以及,

[0166] b)从跟踪的相应图像数据量中确定对于对应硬件线缓冲器存储器的相应硬件存储器分配;

[0167] c)生成用于图像处理器执行图像处理应用软件程序的配置信息,该配置信息描述对于图像处理器的硬件线缓冲器存储器的硬件存储器分配。

[0168] 示例25:根据示例24的方法,其中,该跟踪还包括跟踪模拟线缓冲器存储器写指针和模拟线缓冲器存储器读指针之间的差。

[0169] 示例26:根据示例24或25的方法,其中,该确定基于模拟线缓冲器存储器写指针和模拟线缓冲器存储器读指针之间的最大观察到的差。

[0170] 示例27:根据示例24至26中至少一个示例的方法,其中,该模拟还包括施加写策略,该写策略防止下一个图像数据单元被写入到模拟线缓冲器存储器中,直到消耗图像数据的一个或多个内核模型等待接收下一个图像数据单元。

[0171] 示例28:根据示例24至27中至少一个示例的方法,其中,在生成下一个图像数据单元的生产内核模型处实施写策略。

[0172] 示例29:根据示例24至28中至少一个示例的方法。其中,图像处理器架构包括耦合到二维移位寄存器阵列的执行阵列。

[0173] 示例30:根据示例24至29中至少一个示例的方法,其中,图像处理器的架构包括线缓冲器、片生成器和/或模板处理器中的至少一个。

[0174] 示例31:根据示例24至30中至少一个示例的方法,其中,该模板处理器被配置为处理重叠模板。

[0175] 示例32:根据示例24至31中至少一个示例的方法,其中,数据计算单元包括移位寄存器结构,该移位寄存器结构具有比执行通道阵列更宽的维度,特别是在执行通道阵列外部有寄存器。

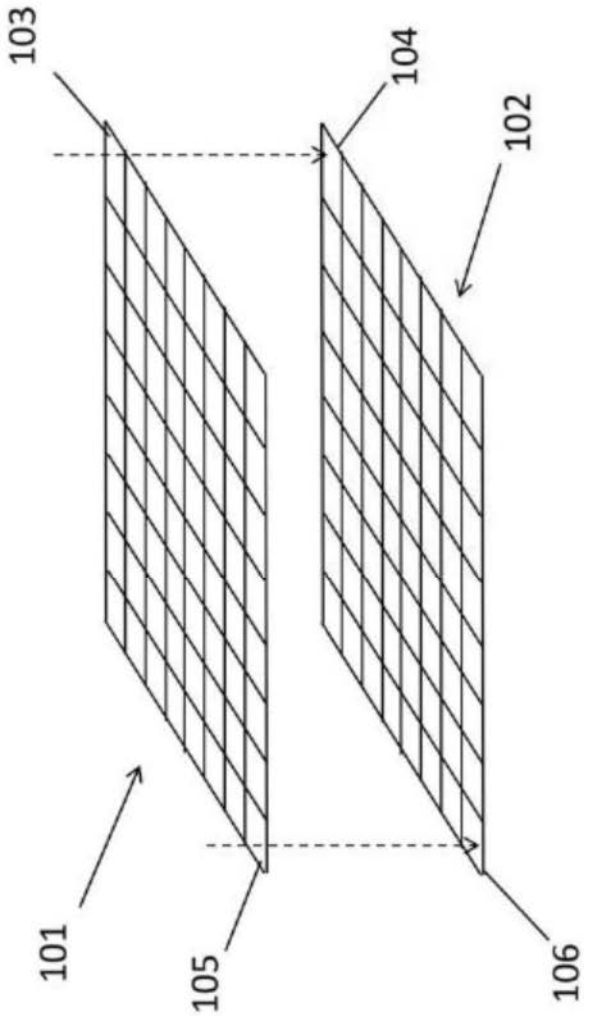


图1

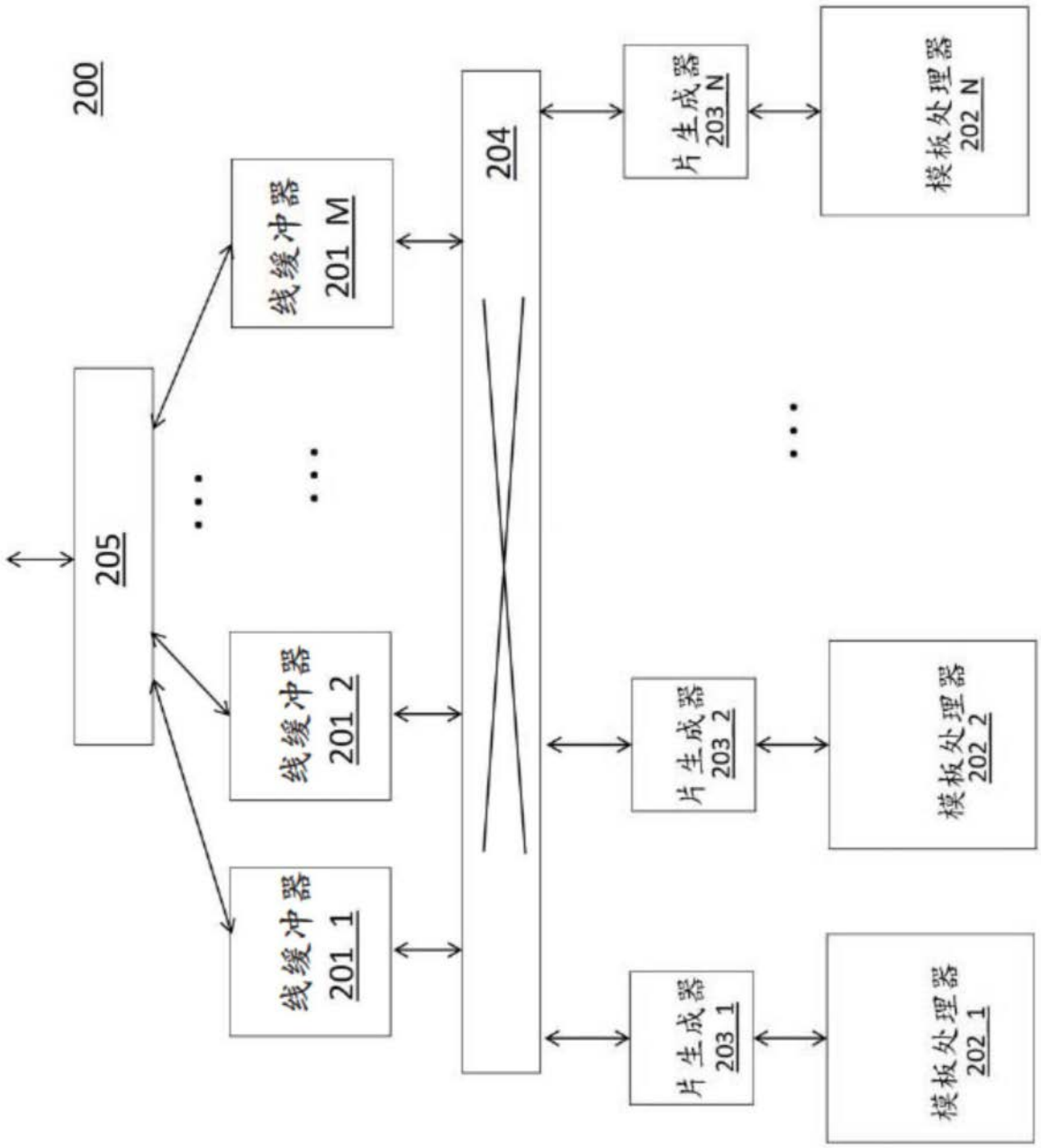

200
205
线缓冲器 201_1
线缓冲器 201_2
线缓冲器 201_M
204
片生成器 203_1
片生成器 203_2
片生成器 203_N
模板处理器 202_1
模板处理器 202_2
模板处理器 202_N

图2

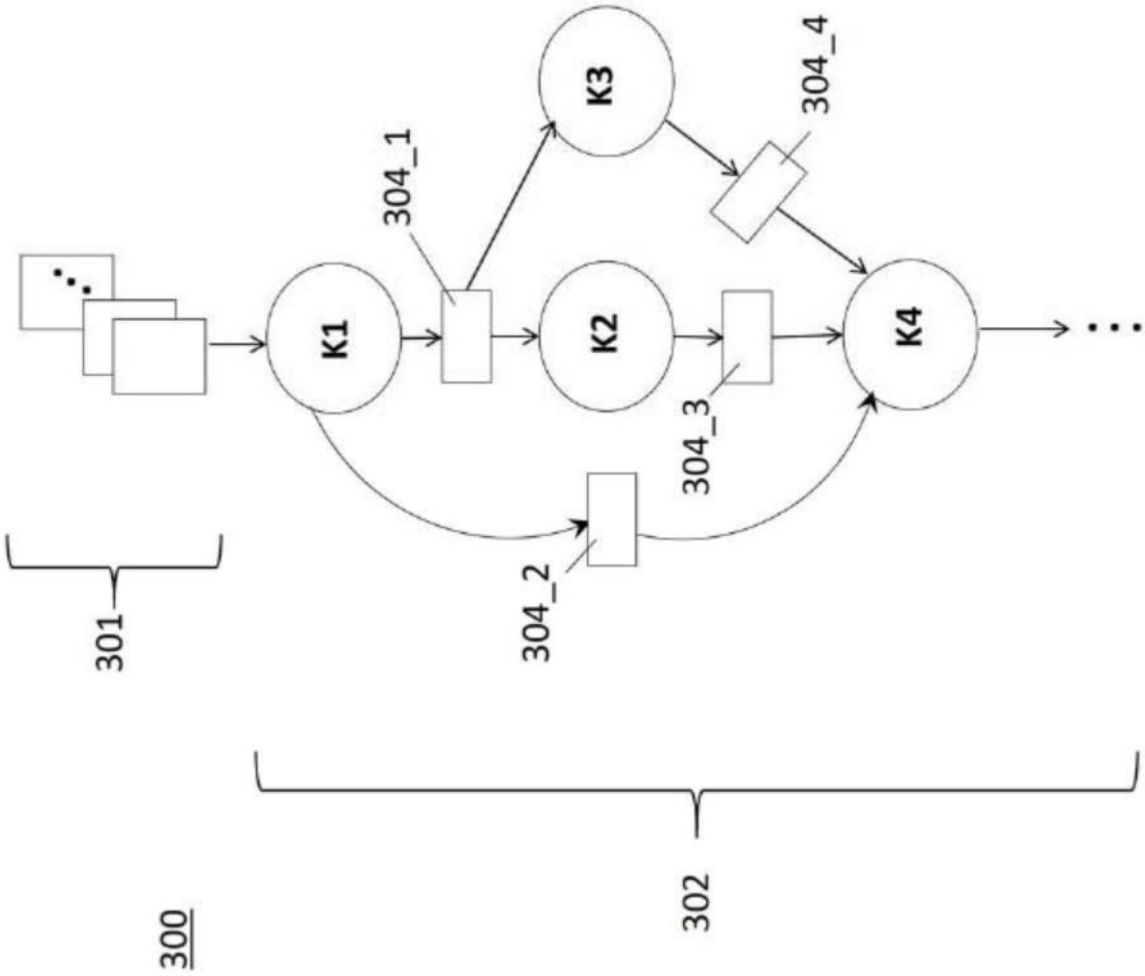


图3

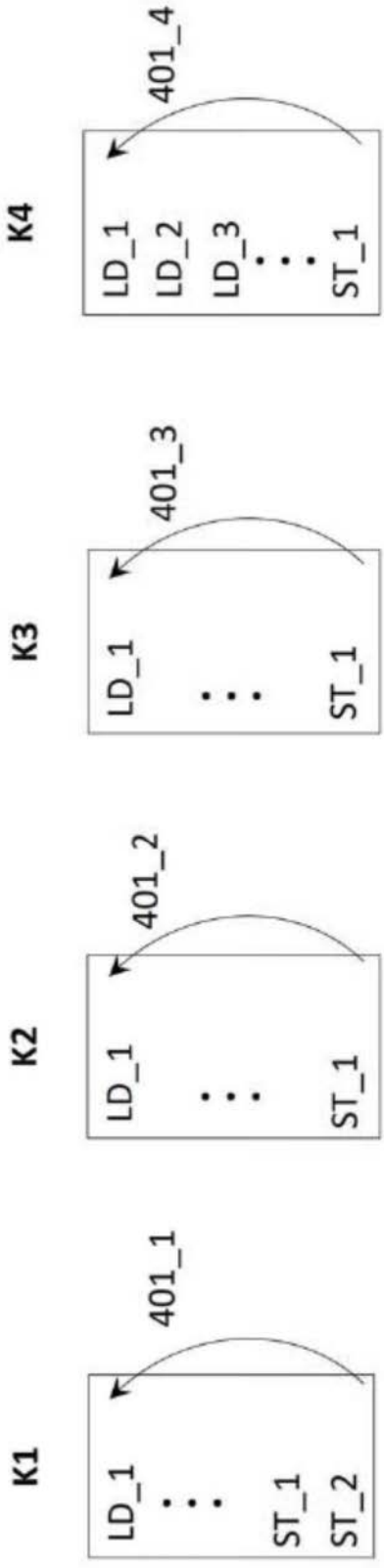

K4
401_4
LD_1
LD_2
LD_3
ST_1
K3
401_3
LD_1
ST_1
K2
401_2
LD_1
ST_1
K1
401_1
LD_1
ST_1
ST_2

图4

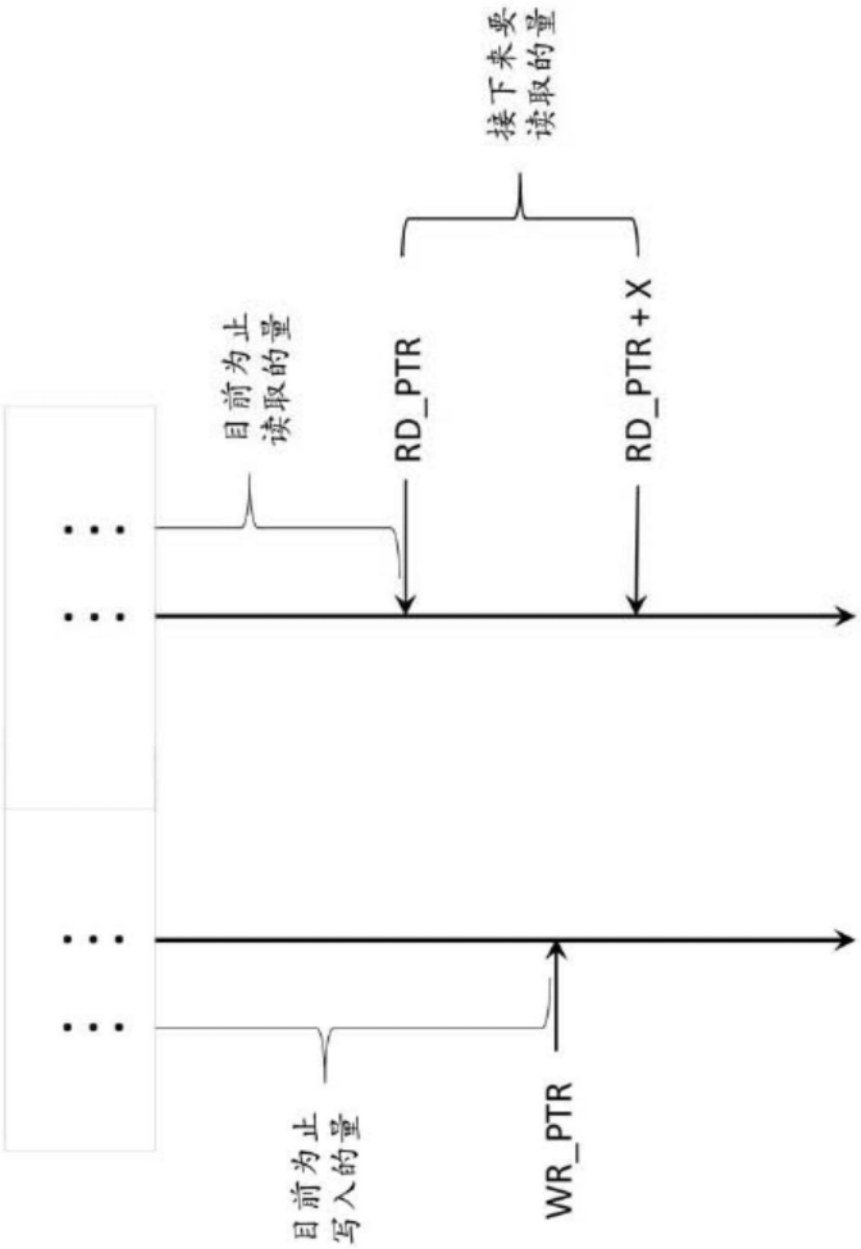


图5a

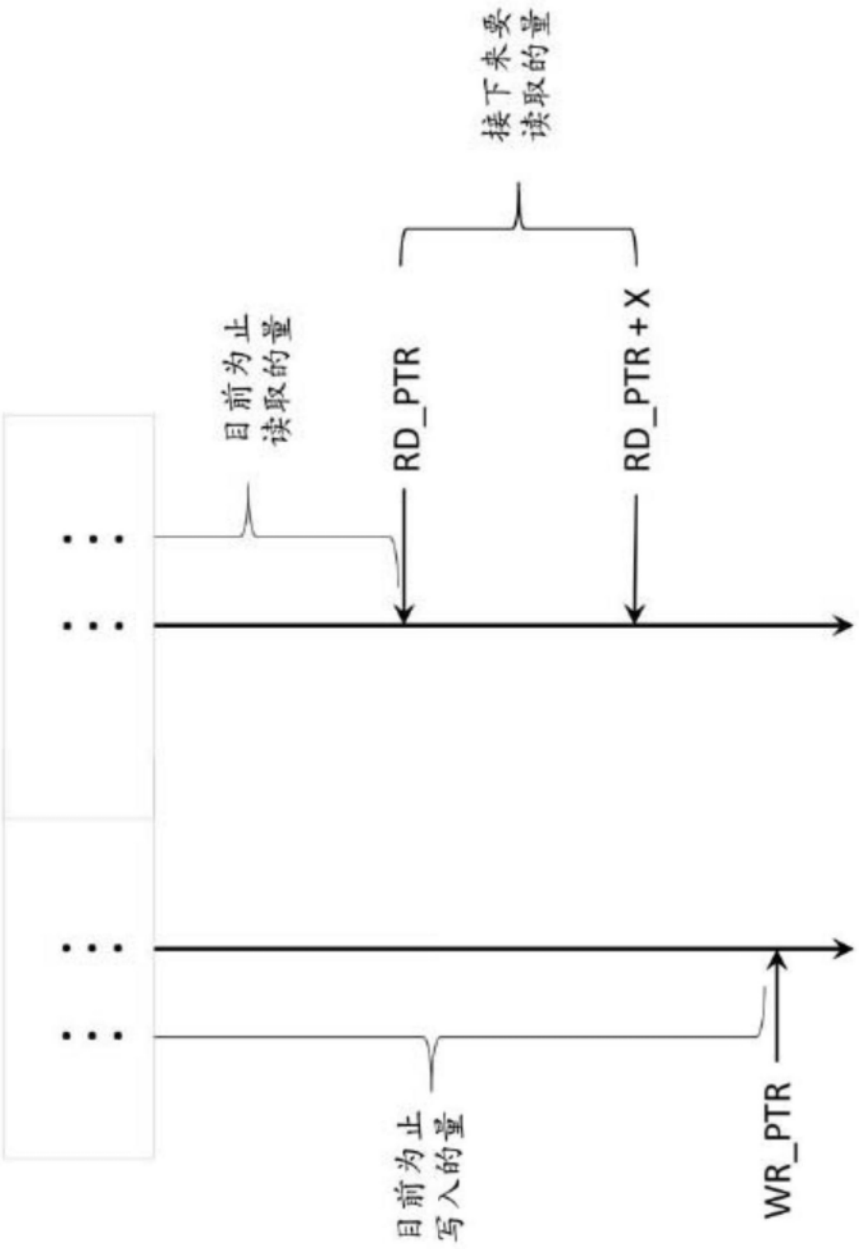


图5b

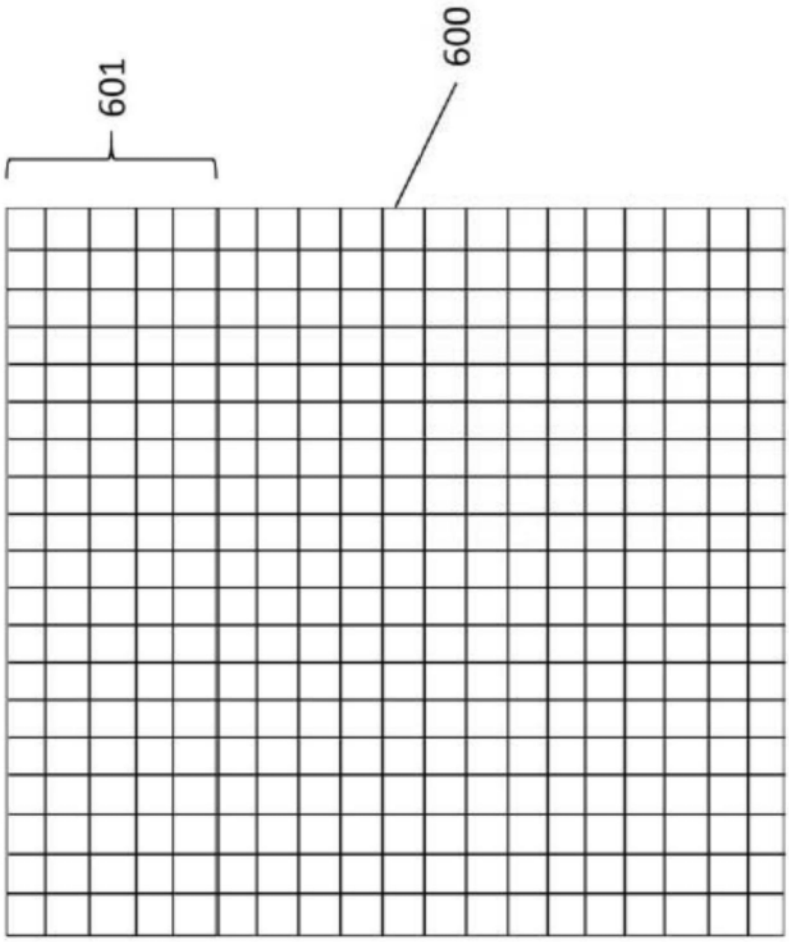


图6a

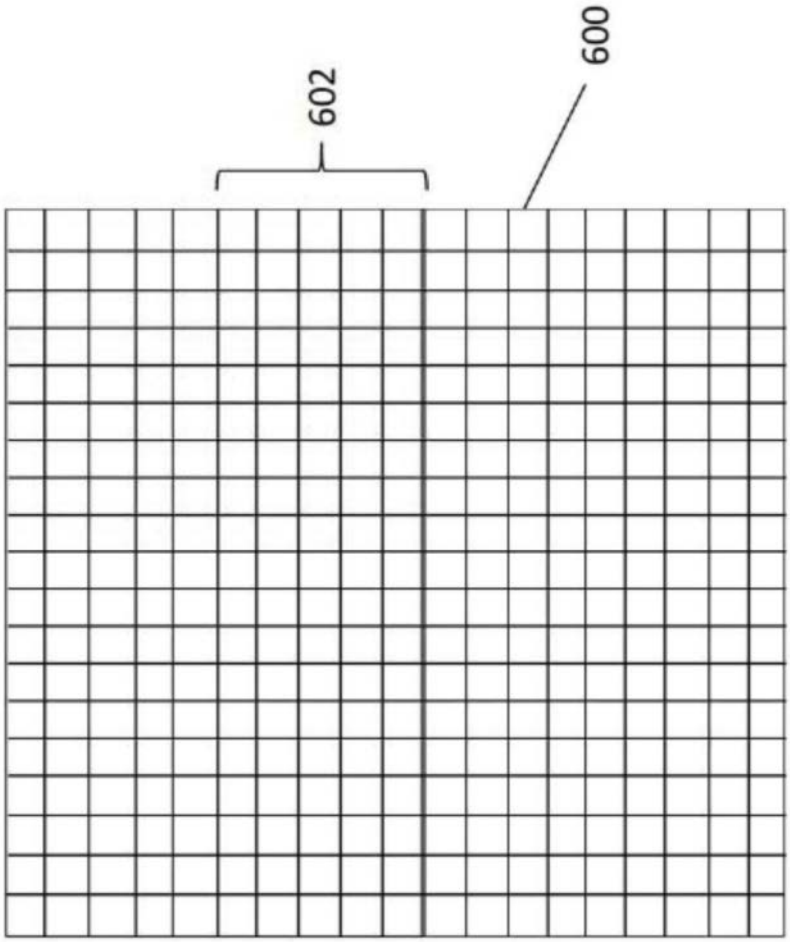


图6b

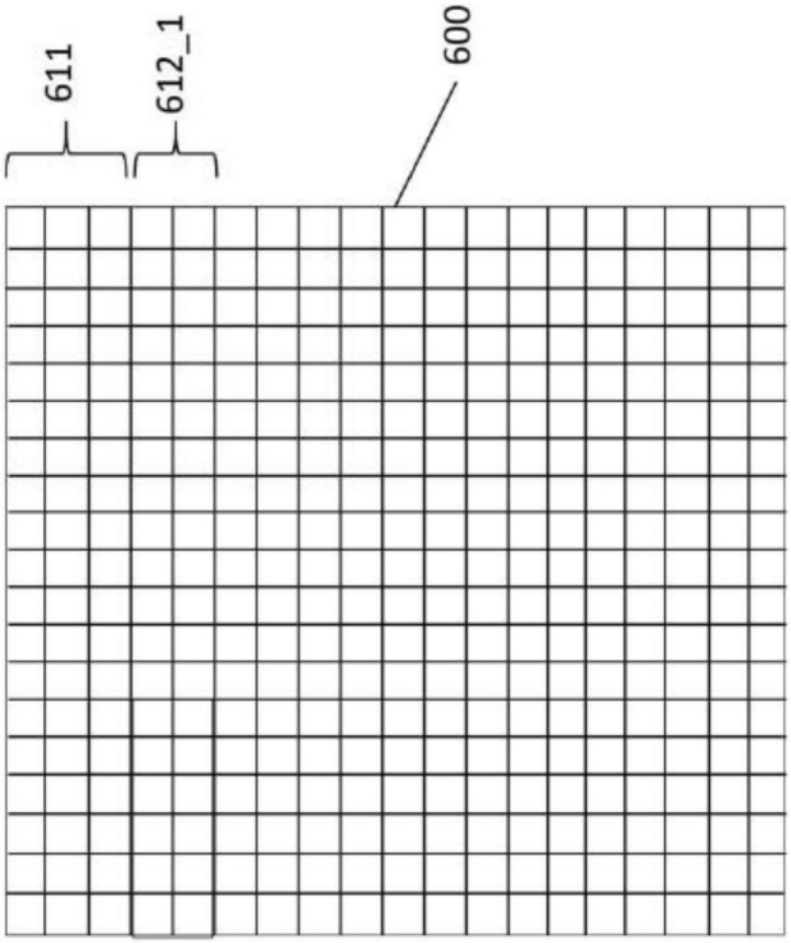


图6c

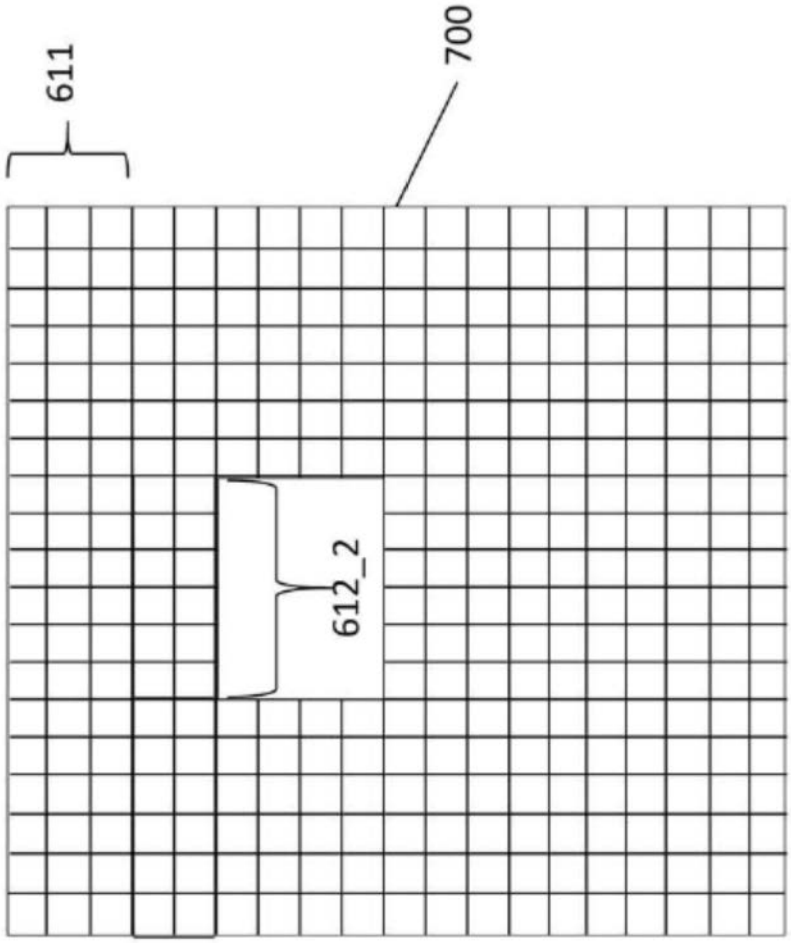


图6d

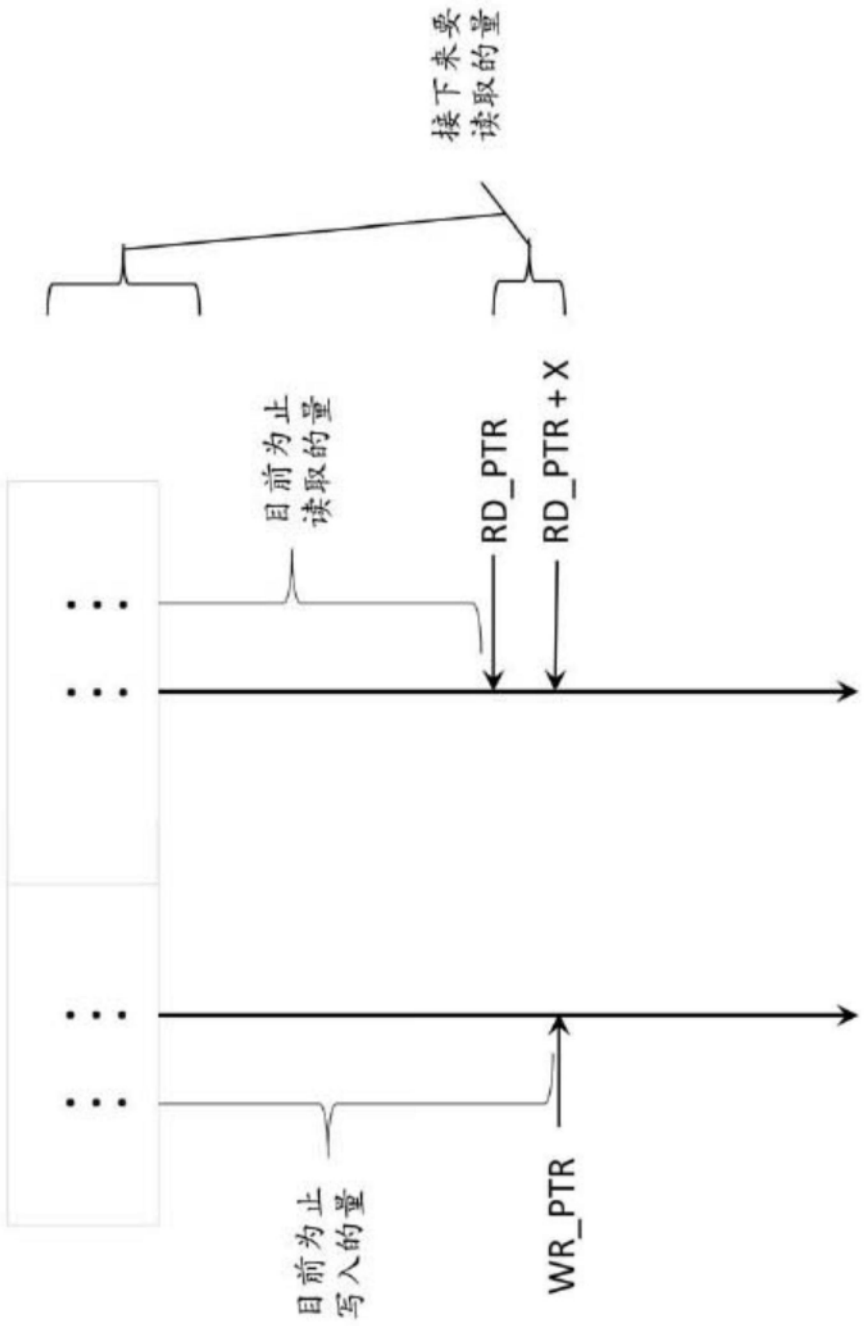


图6e

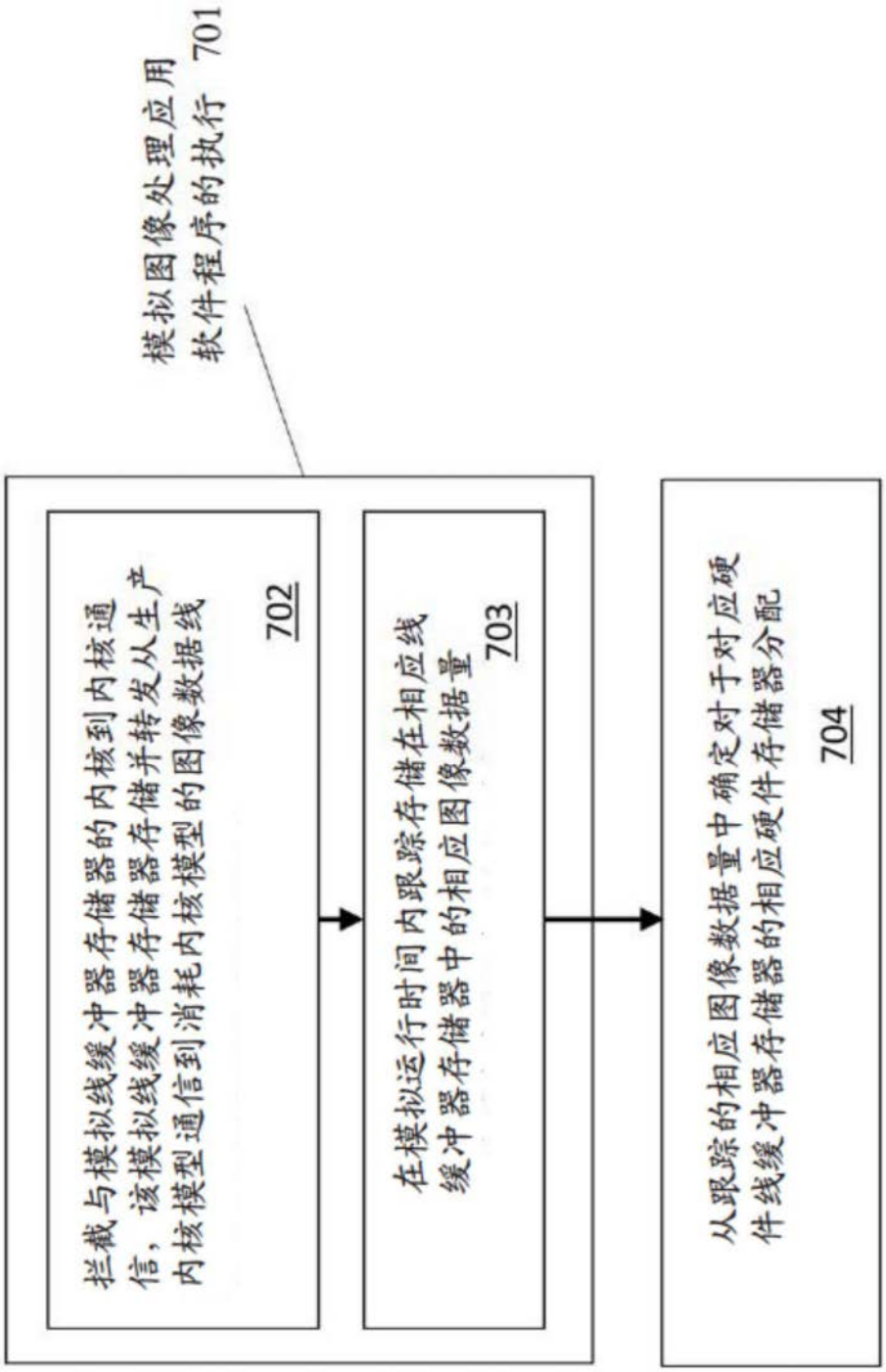

模拟图像处理应用
软件程序的执行 701
拦截与模拟线缓冲器存储器的内核到内核通信，该模拟线缓冲器存储器存储并转发从生产内核模型通信到消耗内核模型的图像数据线
702
在模拟运行时间内跟踪存储在相应线缓冲器存储器中的相应图像数据量
703
从跟踪的相应图像数据量中确定对于对应硬件线缓冲器存储器的相应硬件存储器分配
704

图7

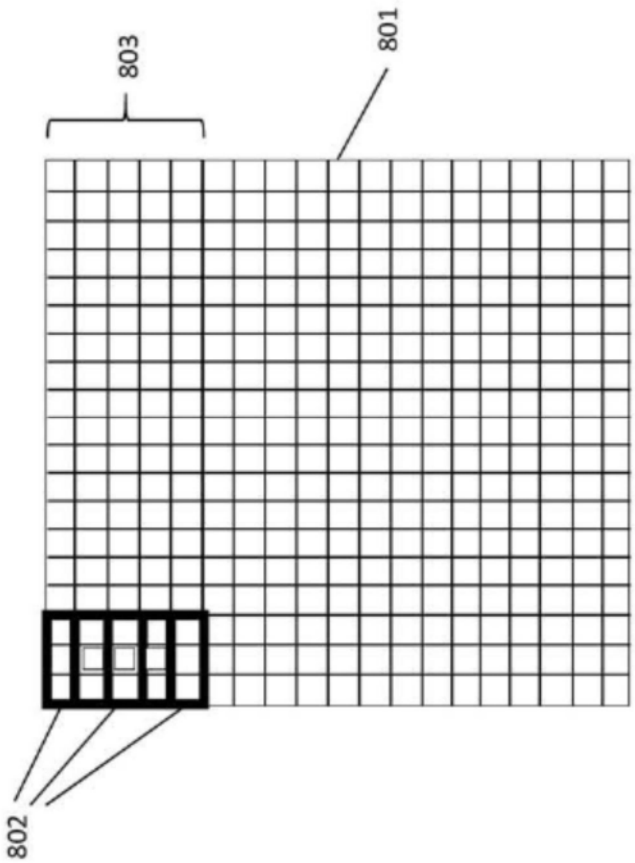


图8a

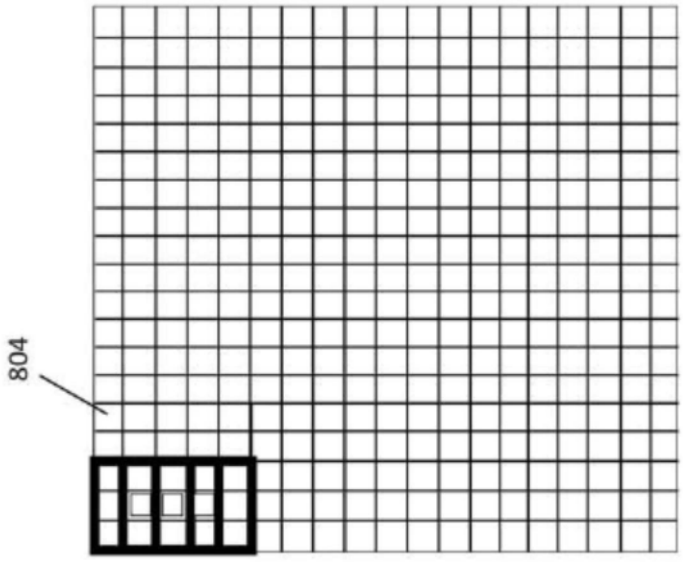


图8b

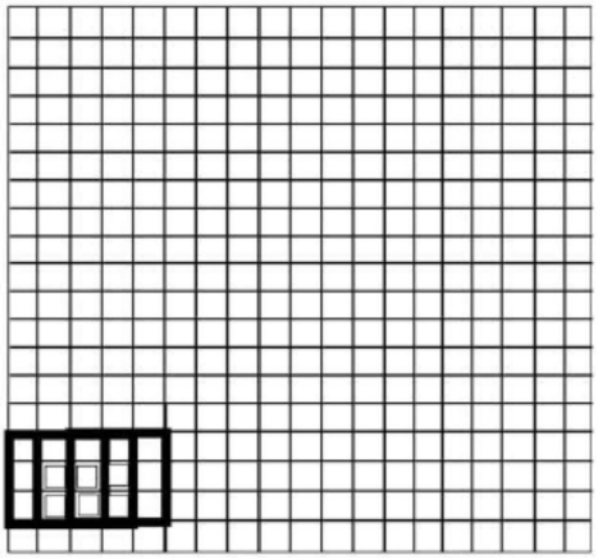


图8c

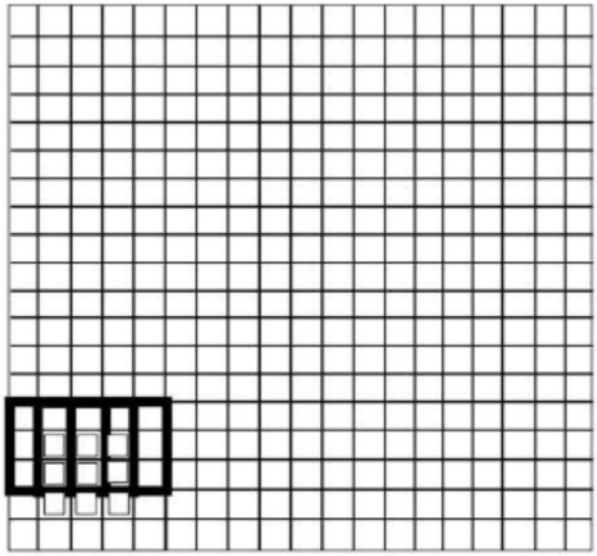


图8d

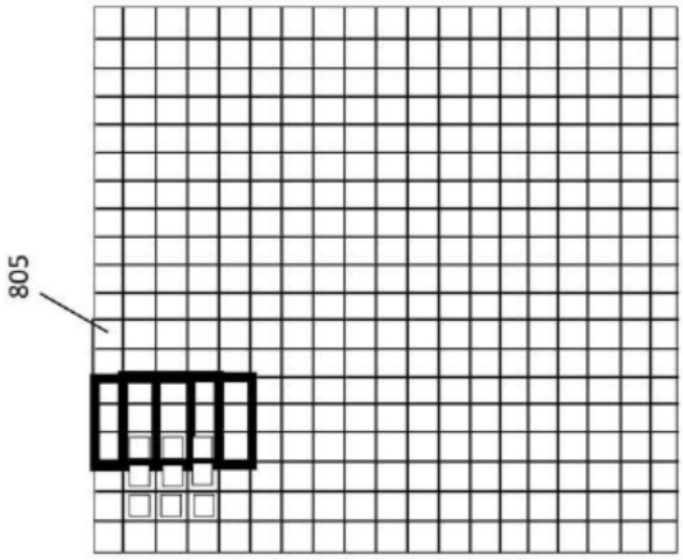


图8e

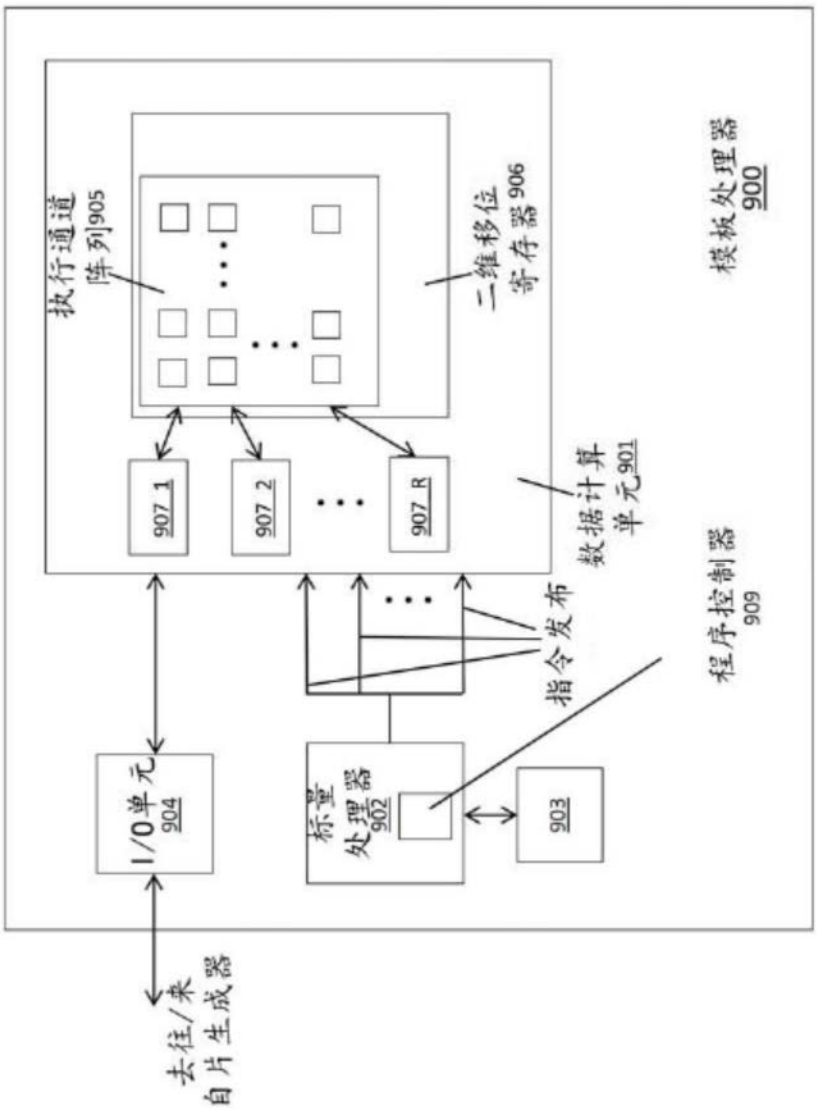

模板处理器
900
执行通道
阵列905
二维移位
寄存器906
907_1
907_2
907_R
数据计算
单元901
指令发布
程序控制器
909
I/O单元
904
标量
处理器
902
903
去往/来
自片生成器

图9a

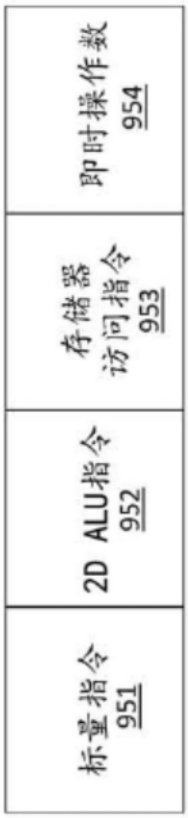


图9b

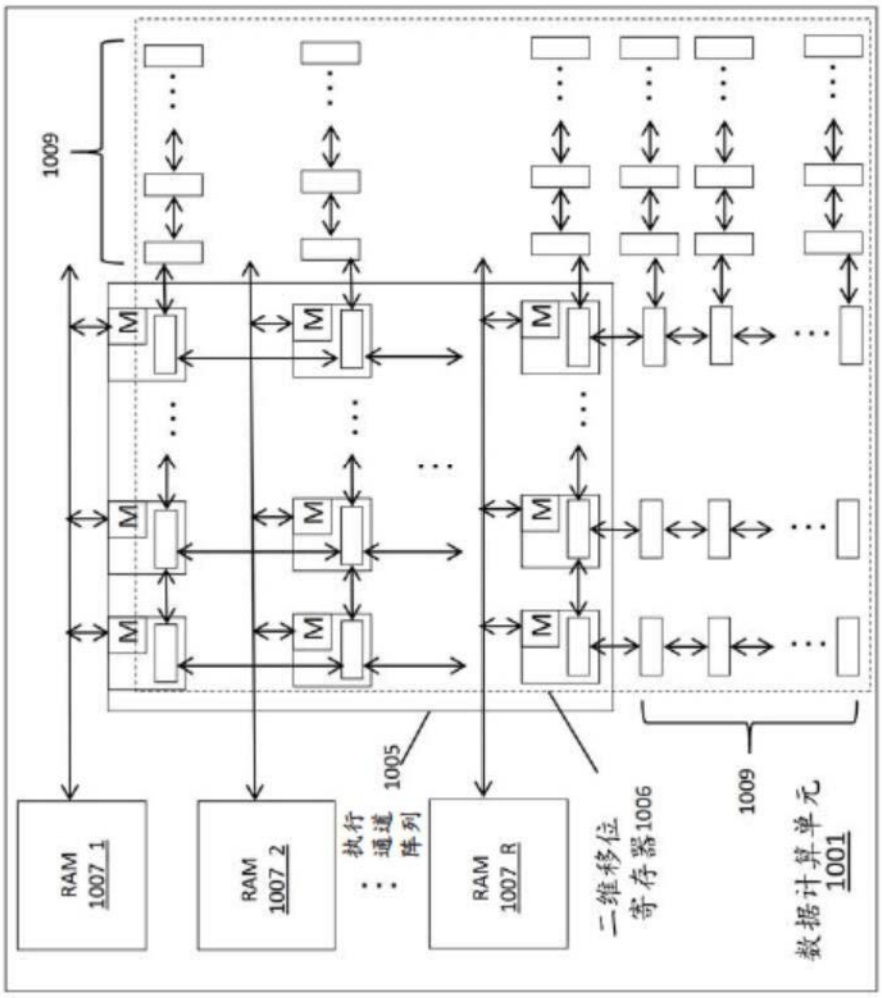

1009
M
RAM
1007_1
RAM
1007_2
执行
通道
阵列
RAM
1007_R
1005
二维移位
寄存器1006
1009
数据计算单元
1001

图10

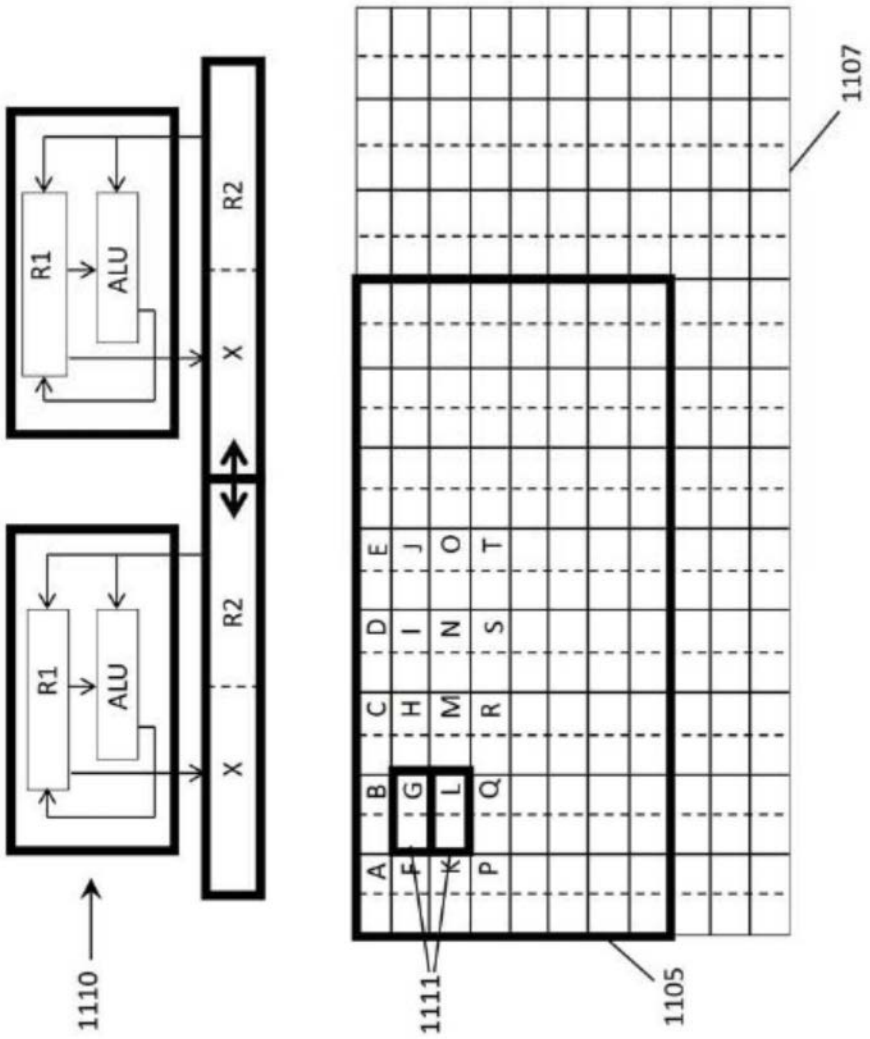

R1
ALU
R2
X
R1
ALU
R2
X
1110
1107
1111
1105
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T

图11a

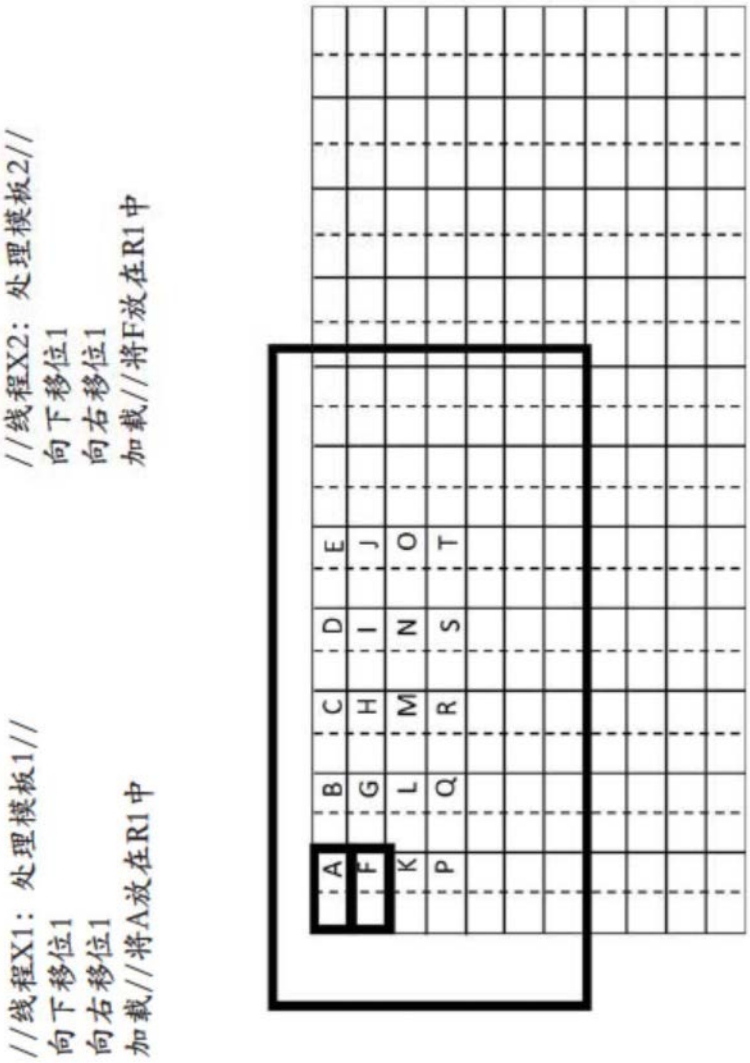

//线程X1：处理模板1//
向下移位1
向右移位1
加载//将A放在R1中
//线程X2：处理模板2//
向下移位1
向右移位1
加载//将F放在R1中
A B C D E
F G H I J
K L M N O
P Q R S T

图11b

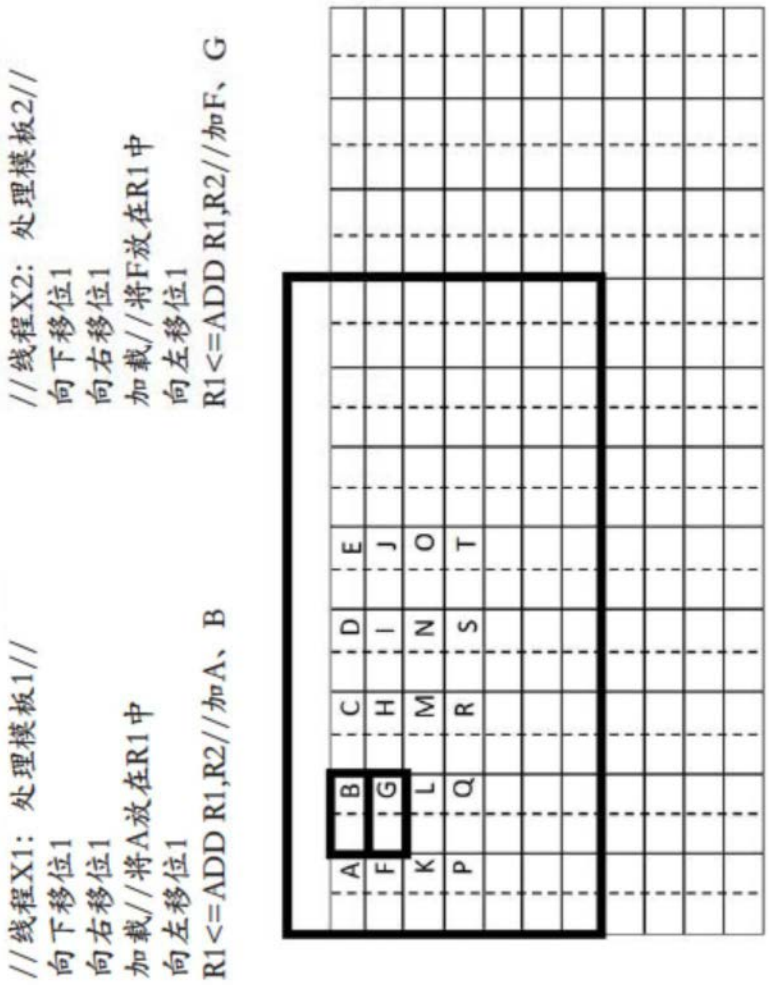

//线程X1: 处理模板1//
向下移位1
向右移位1
加载//将A放在R1中
向左移位1
R1<=ADD R1,R2//加A、B
//线程X2: 处理模板2//
向下移位1
向右移位1
加载//将F放在R1中
向左移位1
R1<=ADD R1,R2//加F、G
A B C D E
F G H I J
K L M N O
P Q R S T

图11c

```
//线程X1：处理模板1//
…
R1<=ADD R1,R2//加A、B
向左移位1
R1<=ADD R1,R2//加C
```

```
//线程X2：处理模板2//
…
R1<=ADD R1,R2//加F、G
向左移位1
R1<=ADD R1,R2//加H
```

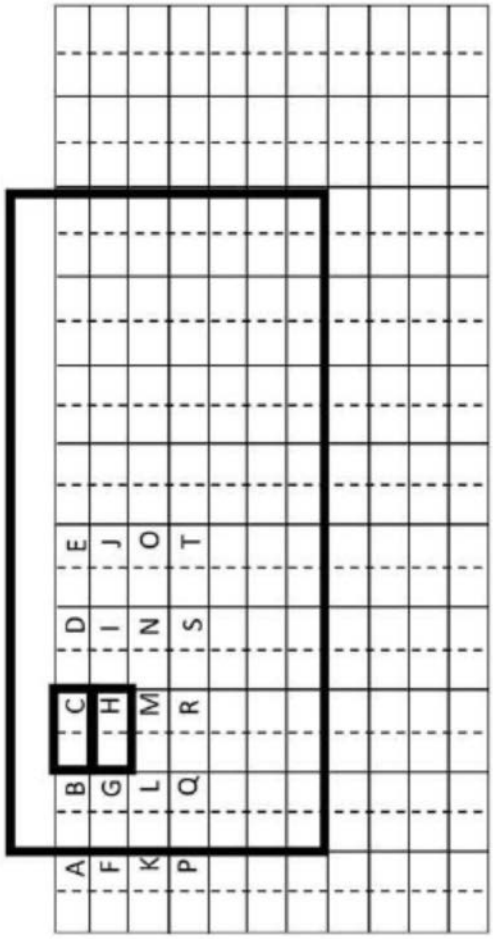


图11d

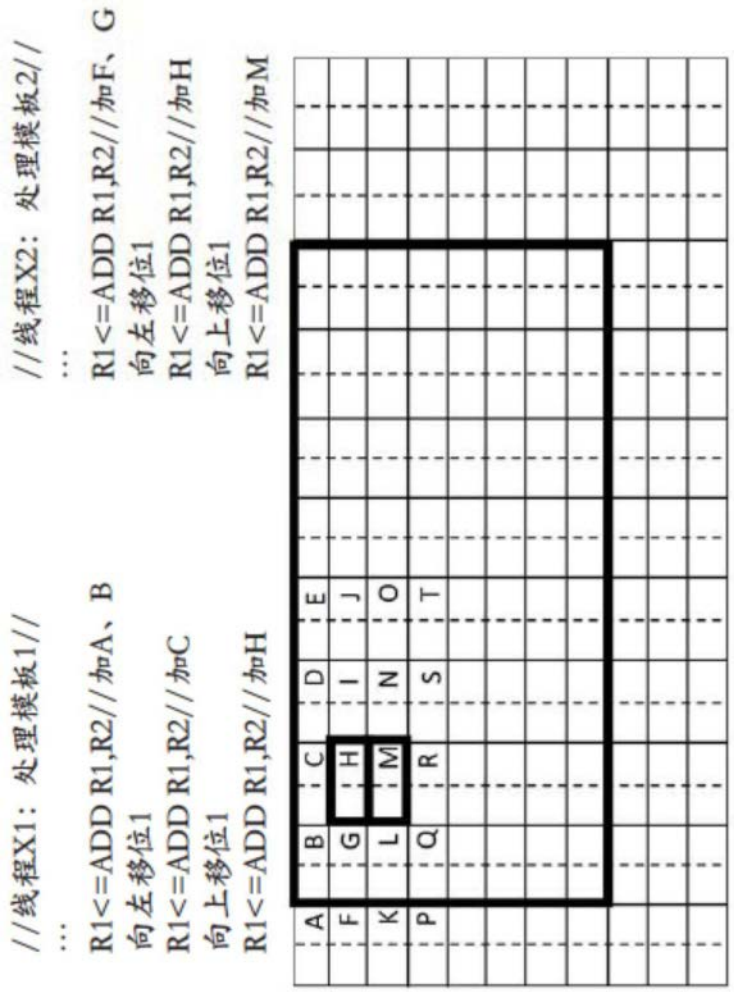

//线程X1：处理模板1//
...
R1<=ADD R1,R2//加A、B
向左移位1
R1<=ADD R1,R2//加C
向上移位1
R1<=ADD R1,R2//加H
//线程X2：处理模板2//
...
R1<=ADD R1,R2//加F、G
向左移位1
R1<=ADD R1,R2//加H
向上移位1
R1<=ADD R1,R2//加M
A B C D E
F G H I J
K L M N O
P Q R S T

图11e

```
//线程X1：处理模板1//
...
向右移位1
R1<=ADD R1,R2//加G
```

```
//线程X2：处理模板2//
...
向右移位1
R1<=ADD R1,R2//加L
```

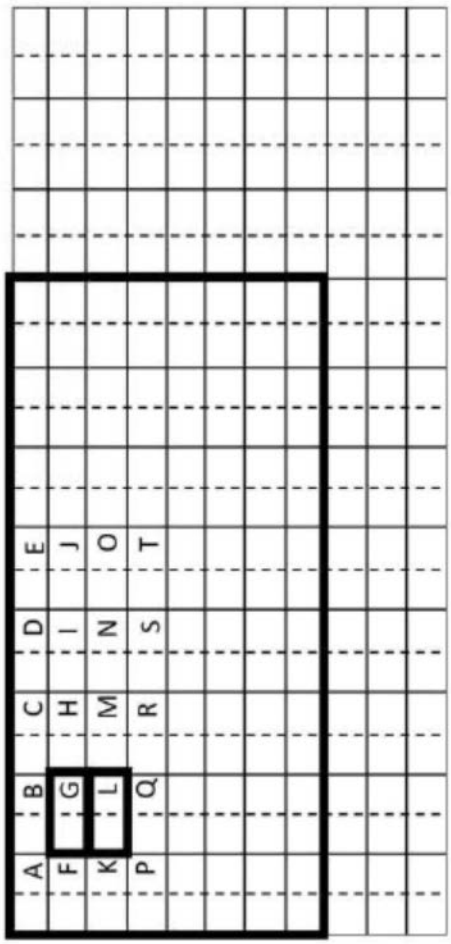


图11f

```
//线程X1：处理模板1//
...
向右移位1
R1<=ADD R1,R2//加F
```

```
//线程X2：处理模板2//
...
向右移位1
R1<=ADD R1,R2//加K
```

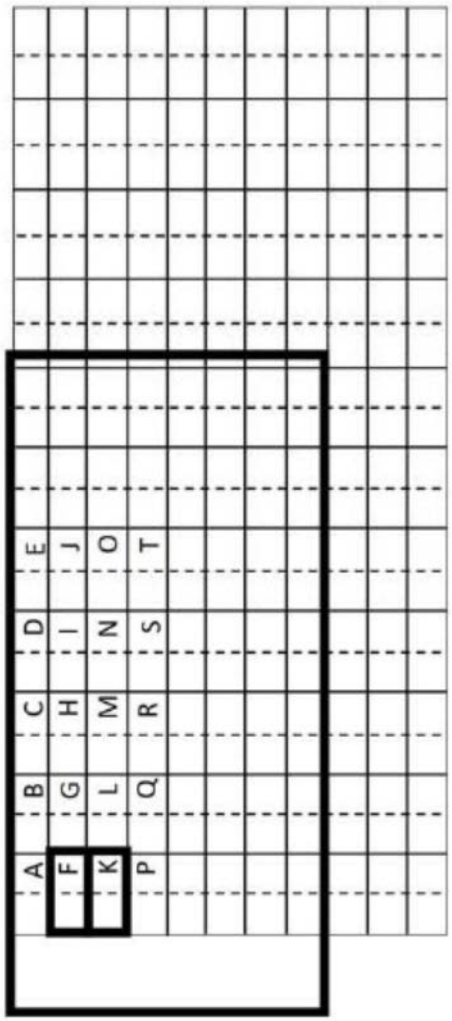


图11g

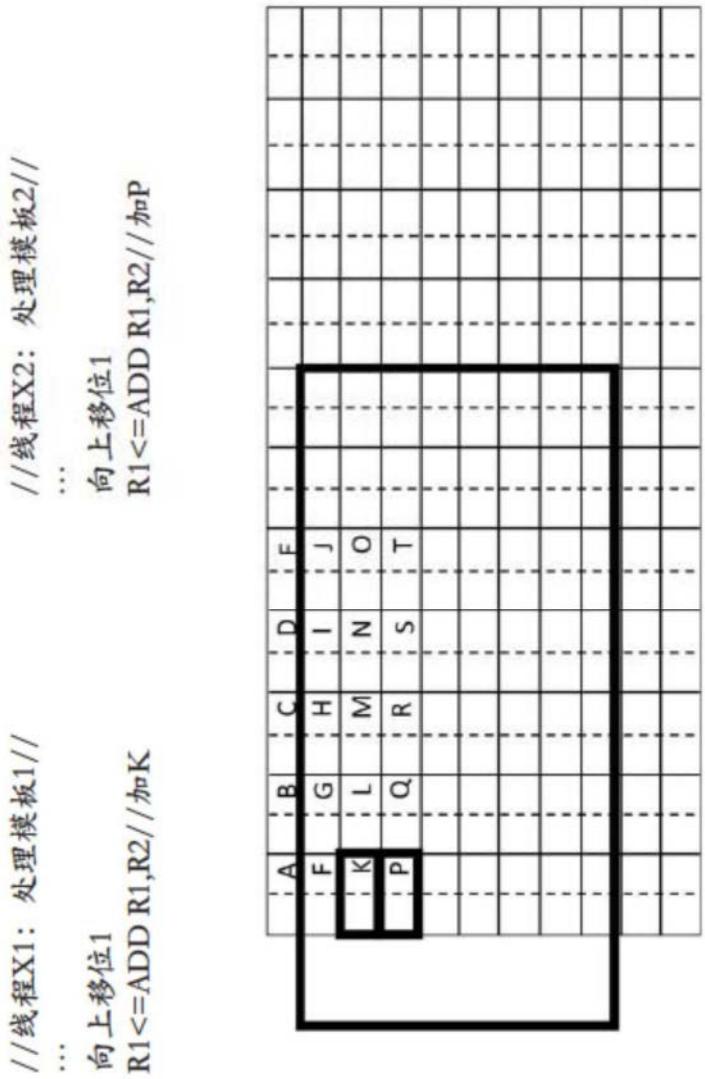

//线程X1： 处理模板1//
…
向上移位1
R1<=ADD R1,R2//加K
//线程X2： 处理模板2//
…
向上移位1
R1<=ADD R1,R2//加P
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T

图11h

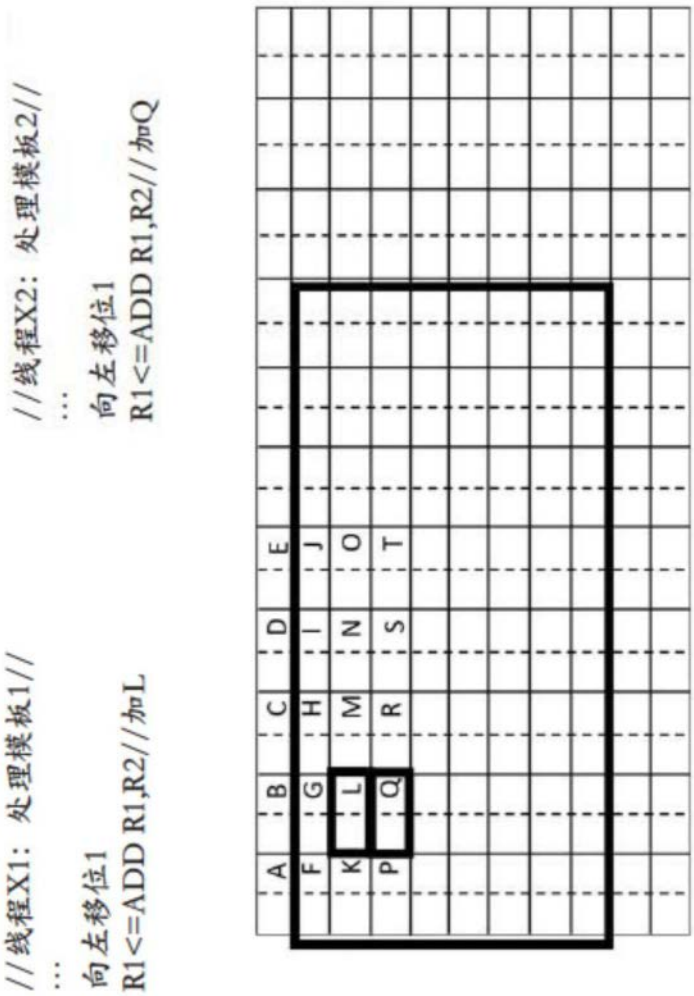

//线程X1：处理模板1//
…
向左移位1
R1<=ADD R1,R2//加L
//线程X2：处理模板2//
…
向左移位1
R1<=ADD R1,R2//加Q
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T

图11i

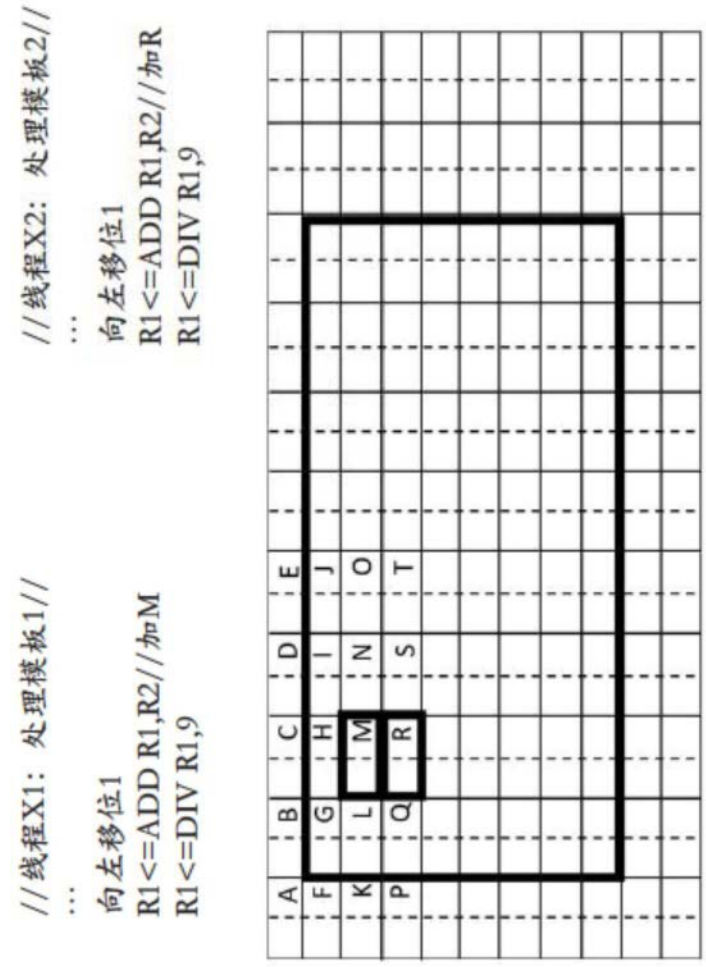

//线程X1：处理模板1//
…
向左移位1
R1<=ADD R1,R2//加M
R1<=DIV R1,9
//线程X2：处理模板2//
…
向左移位1
R1<=ADD R1,R2//加R
R1<=DIV R1,9
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T

图11j

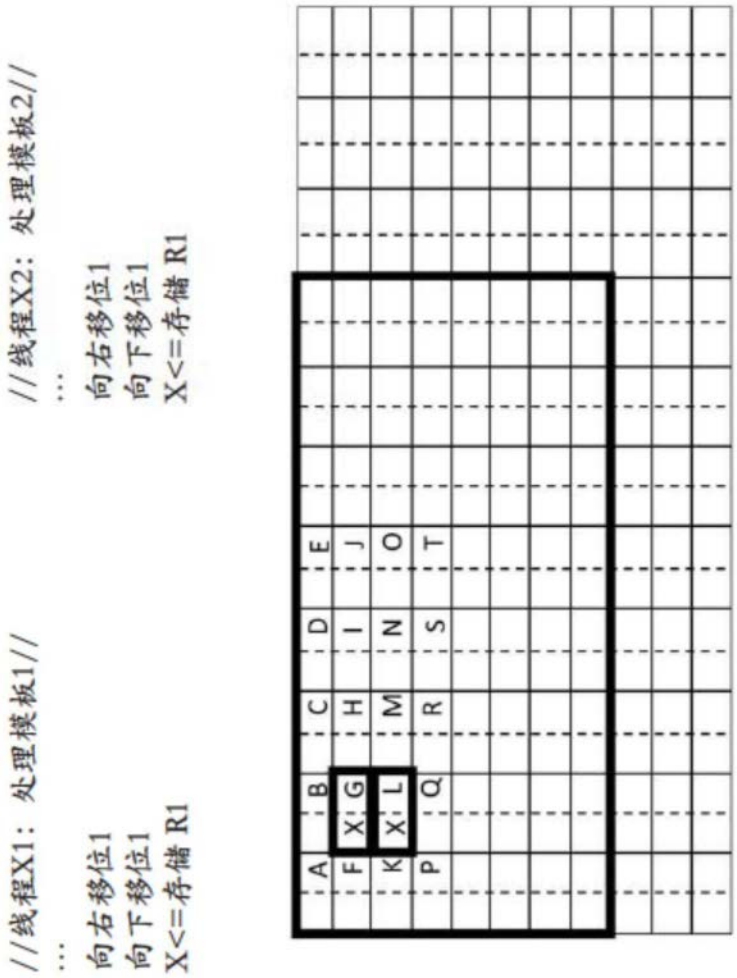

//线程X1：处理模板1//
...
向右移位1
向下移位1
X<=存储 R1
//线程X2：处理模板2//
...
向右移位1
向下移位1
X<=存储 R1
A
B
C
D
E
F
X
G
H
I
J
K
X
L
M
N
O
P
Q
R
S
T

图11k

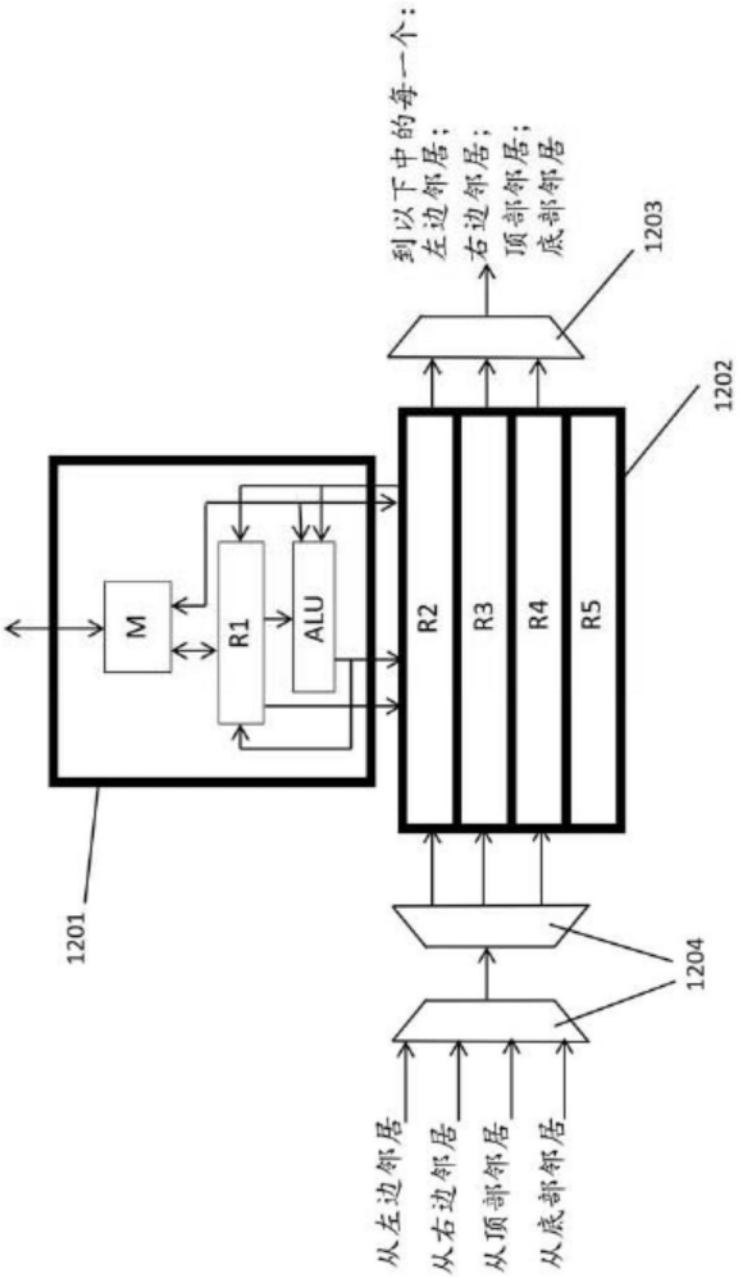

到以下中的每一个：
左边邻居；
右边邻居；
顶部邻居；
底部邻居
1203
1202
M
R1
ALU
R2
R3
R4
R5
1201
1204
从左边邻居
从右边邻居
从顶部邻居
从底部邻居

图12

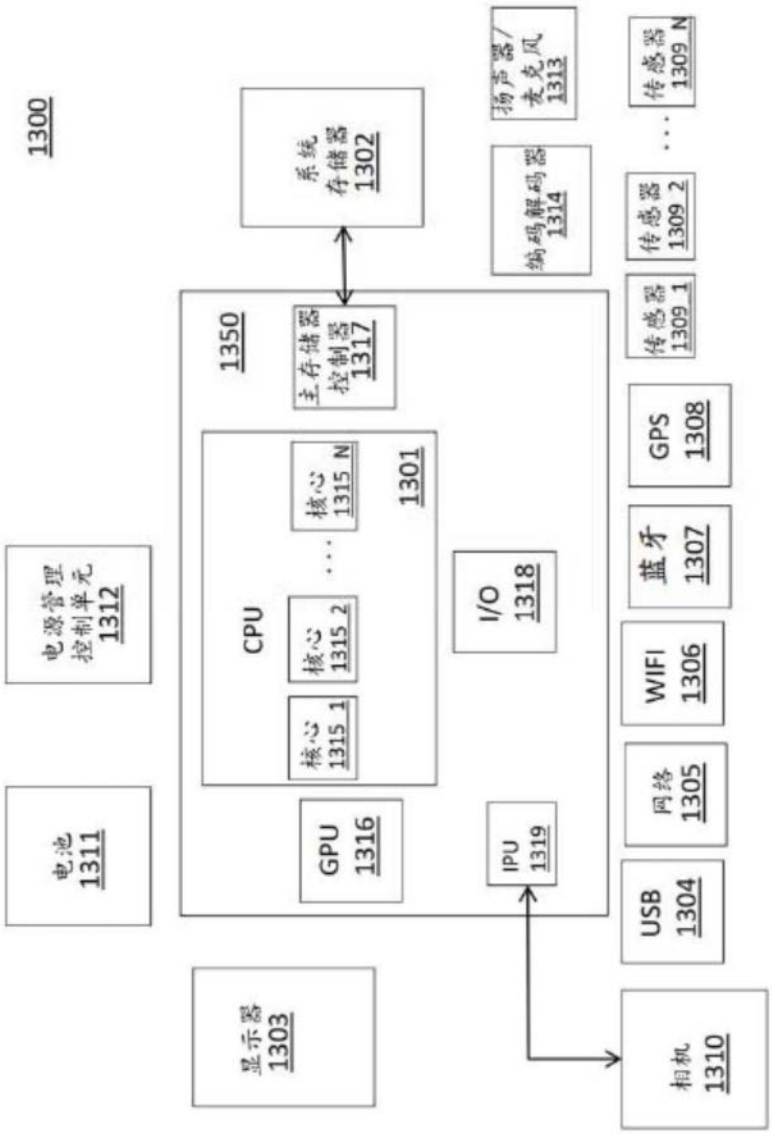

1300
系统存储器 1302
扬声器/麦克风 1313
传感器 1309_N
编码解码器 1314
传感器 1309_2
传感器 1309_1
1350
主存储器控制器 1317
GPS 1308
1301
CPU
核心 1315_N
核心 1315_2
核心 1315_1
蓝牙 1307
I/O 1318
WIFI 1306
电源管理控制单元 1312
网络 1305
GPU 1316
IPU 1319
USB 1304
电池 1311
显示器 1303
相机 1310

图13