(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 108470211 B
(45)授权公告日 2022.07.12

(21)申请号 201810312903.1

(22)申请日 2018.04.09

(65)同一申请的已公布的文献号
申请公布号 CN 108470211 A

(43)申请公布日 2018.08.31

(73)专利权人 郑州云海信息技术有限公司
地址 450018 河南省郑州市郑东新区心怡路278号16层1601室

(72)发明人 张纪伟

(74)专利代理机构 北京安信方达知识产权代理有限公司 11262
专利代理师 李红爽 李丹

(51)Int.Cl.
G06N 3/04(2006.01)
G06N 3/063(2006.01)

(56)对比文件
CN 107885700 A,2018.04.06
CN 106203621 A,2016.12.07
CN 103955443 A,2014.07.30
CN 107329734 A,2017.11.07
CN 107463990 A,2017.12.12
CN 107832804 A,2018.03.23

审查员 王建培

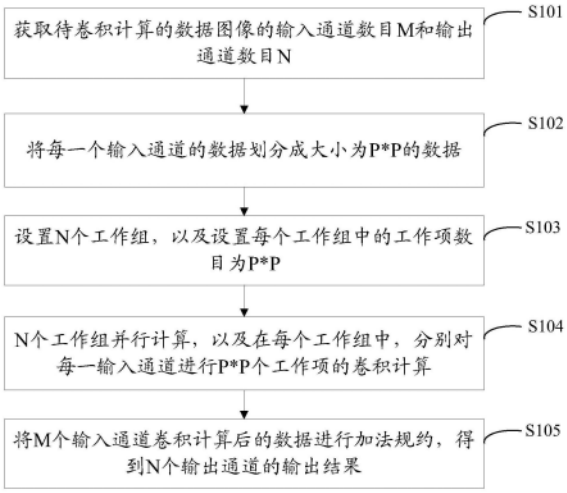
权利要求书2页 说明书8页 附图3页

(54)发明名称

一种卷积计算的实现方法、设备和计算机存储介质

(57)摘要

本发明公开了一种卷积计算的实现方法、设备和计算机存储介质，该方法包括：获取待卷积计算的数据图像的输入通道数目M和输出通道数目N；将每一个输入通道的数据划分成大小为P*P的数据；设置N个工作组，以及设置每个工作组中的工作项数目为P*P；N个工作组并行计算，以及在每个工作组中，分别对每一输入通道的数据进行P*P个工作项的卷积计算；将M个输入通道卷积计算后的数据进行加法规约，得到N个输出通道的输出结果。本发明提供了一种卷积计算的实现方法、设备和计算机存储介质，实现对数据的并行卷积计算，满足FPGA的OpenCL程序的并行模式的工作模式。



CN 108470211 B

1.一种卷积计算的实现方法,包括:

获取待卷积计算的数据图像的输入通道数目M和输出通道数目N;

将每一个输入通道的数据划分成大小为P*P的数据;

设置N个工作组,以及设置每个工作组中的工作项数目为P*P;

所述N个工作组并行计算,以及在每个工作组中,分别对每一输入通道的数据进行P*P个工作项的卷积计算;

将M个输入通道卷积计算后的数据进行加法规约,得到N个输出通道的输出结果;

其中,M、N和P均为正整数,一个工作项对应一个线程。

2.根据权利要求1所述的方法,其特征在于,所述P根据卷积计算的输入图片或输出图片大小确定。

3.根据权利要求1所述的方法,其特征在于,所述设置N个工作组,以及设置每个工作组中的工作项数目为P*P时,所述方法还包括:

设置每个工作组中的最大工作项数目。

4.根据权利要求1-3任一项所述的方法,其特征在于,所述设置N个工作组,以及设置每个工作组中的工作项数目为P*P时,所述方法还包括:

设置多个计算单元。

5.根据权利要求1-3任一项所述的方法,其特征在于,在所述对每一输入通道的数据进行P*P个工作项的卷积计算时,以循环展开方式进行。

6.根据权利要求1-3任一项所述的方法,其特征在于,在所述N个工作组并行计算,以及在每个工作组中,分别对每一输入通道的数据进行P*P个工作项的卷积计算;将M个输入通道卷积计算后的数据进行加法规约,得到N个输出通道每个工作组的输出结果时,将中间计算结果或变量存储在寄存器中,所述输出结果存储在全局内存中。

7.一种卷积计算的实现设备,其特征在于,包括:

获取模块,用于获取待卷积计算的数据图像的输入通道数目M和输出通道数目N;

划分模块,用于将每一个输入通道的数据划分成大小为P*P的数据;

设置模块,用于设置N个工作组,以及设置每个工作组中的工作项数目为P*P;

计算模块,用于所述N个工作组并行计算,以及在每个工作组中,分别对每一输入通道的数据进行P*P个工作项的卷积计算;

累加模块,用于将M个输入通道卷积计算后的数据进行加法规约,得到N个输出通道的输出结果;

其中,M、N和P均为正整数,一个工作项对应一个线程。

8.根据权利要求7所述的设备,其特征在于,所述P根据卷积计算的输入图片或输出图片大小确定;

所述设置模块,还用于:

设置每个工作组中的最大工作项数目;

和/或;

设置多个计算单元。

9.根据权利要求7或8所述的设备,其特征在于,

所述计算模块,在对每一输入通道的数据进行P*P个工作项的卷积计算时,以循环展开

方式进行；

和/或；

所述计算模块，在N个工作组并行计算，以及在每个工作组中，分别对每一输入通道的数据进行P*P个工作项的卷积计算；将M个输入通道卷积计算后的数据进行加法规约，得到N个输出通道每个工作组的输出结果时，将中间计算结果或变量存储在寄存器中，所述输出结果存储在全局内存中。

10.一种卷积计算的实现设备，其特征在于，包括存储器和处理器，存储器用于存储执行指令；处理器调用所述执行指令，用于执行如权利要求1-6任一项所述的卷积计算的实现方法。

11.一种计算机可读存储介质，其上存储有计算机指令，其特征在于，所述指令被处理器执行时实现权利要求1-6任一项所述的方法的步骤。

一种卷积计算的实现方法、设备和计算机存储介质

技术领域

[0001] 本发明涉及计算机技术，尤指一种卷积计算的实现方法、设备和计算机存储介质。

背景技术

[0002] 在深度学习卷积神经网络模型中，卷积层多达几十到上千层，每层训练参数可以达到几万到几十万，总训练参数更是达到了千万级别。深度学习网络的训练时间可以多达几周甚至几个月的时间。

[0003] 在深度学习计算的硬件层面，现场可编程门阵列(Field-Programmable Gate Array，简称FPGA)已经成为深度学习计算的重要平台。相比于图形处理器(Graphics Processing Unit，简称GPU)，FPGA作为可编程重构的硬件，拥有更强大的可调控能力，拥有更高的计算效率；相比于专用集成电路芯片(Application Specific Integrated Circuit，简称ASIC)，省去了ASIC方案的流片过程，使开发周期缩短。与此同时，开放运算语言(Open Computing Language，简称OpenCL)与FPGA的结合更能让研发者快速上手，一定程度上降低了研发者的开发时限，使研究更加简单易行。其中，OpenCL与FPGA的结合指的是使用OpenCL编程语言在FPGA上实现卷积计算。

[0004] 然而，FPGA的OpenCL程序的工作模式是并行模式，而目前使用OpenCL编程语言在FPGA上进行卷积计算Fast Algorithms只能串行实现，无法满足FPGA的OpenCL程序的并行模式的工作模式。

发明内容

[0005] 为了解决上述技术问题，本发明提供了一种卷积计算的实现方法、设备和计算机存储介，实现对数据的并行卷积计算，满足FPGA的OpenCL程序的并行模式的工作模式。

[0006] 为了达到本发明目的，第一方面，本发明提供了一种卷积计算的实现方法，包括：

[0007] 获取待卷积计算的数据图像的输入通道数目M和输出通道数目N；

[0008] 将每一个输入通道的数据划分成大小为P*P的数据；

[0009] 设置N个工作组，以及设置每个工作组中的工作项数目为P*P；

[0010] 所述N个工作组并行计算，以及在每个工作组中，分别对每一输入通道的数据进行P*P个工作项的卷积计算；

[0011] 将M个输入通道卷积计算后的数据进行加法规约，得到N个输出通道的输出结果；

[0012] 其中，M、N和P均为正整数。

[0013] 第二方面，本发明提供了一种卷积计算的实现设备，包括：

[0014] 获取模块，用于获取待卷积计算的数据图像的输入通道数目M和输出通道数目N；

[0015] 划分模块，用于将每一个输入通道的数据划分成大小为P*P的数据；

[0016] 设置模块，用于设置N个工作组，以及设置每个工作组中的工作项数目为P*P；

[0017] 计算模块，用于所述N个工作组并行计算，以及在每个工作组中，分别对每一输入通道的数据进行P*P个工作项的卷积计算；

[0018] 累加模块，用于将M个输入通道卷积计算后的数据进行加法规约，得到N个输出通道的输出结果；

[0019] 其中，M、N和P均为正整数。

[0020] 第三方面，本发明提供了一种卷积计算的实现设备，包括存储器和处理器，存储器用于存储执行指令；处理器调用所述执行指令，用于执行如第一方面实施例所述的卷积计算的实现方法。

[0021] 第四方面，本发明提供了一种计算机可读存储介质，其上存储有计算机指令，所述指令被处理器执行时实现第一方面实施例所述的方法的步骤。

[0022] 本申请与现有技术相比具有以下有益效果：1)基于OpenCL+FPGA对卷积计算Fast Algorithm程序的任务划分，每次可以处理N个P*P的数据，以实现对数据的并行卷积计算。2)对核函数设置工作组中最大工作项为所有P*P的最大值，对计算点积的循环设置循环展开，能有效利用FPGA硬件资源，提高计算效率。3)用寄存器来优化核函数中各个输入图片通道的规约计算，减少全局访存时间。

[0023] 本发明的其它特征和优点将在随后的说明书中阐述，并且，部分地从说明书中变得显而易见，或者通过实施本发明而了解。本发明的目的和其他优点可通过在说明书、权利要求书以及附图中所特别指出的结构来实现和获得。

附图说明

[0024] 附图用来提供对本发明技术方案的进一步理解，并且构成说明书的一部分，与本申请的实施例一起用于解释本发明的技术方案，并不构成对本发明技术方案的限制。

[0025] 图1为本发明实施例提供的卷积计算的实现方法的流程图；

[0026] 图2为本发明实施例提供的数据划分的结构示意图；

[0027] 图3为本发明实施例提供的将M个输入通道卷积计算后的数据进行加法规约的结构示意图；

[0028] 图4为现有技术采用全局存储规约的代码示意图；

[0029] 图5为本发明实施例提供的局部存储规约的代码示意图；

[0030] 图6为本发明实施例一提供的卷积计算的实现设备的结构示意图；

[0031] 图7为本发明实施例二提供的卷积计算的实现设备的结构示意图。

具体实施方式

[0032] 为使本发明的目的、技术方案和优点更加清楚明白，下文中将结合附图对本发明的实施例进行详细说明。需要说明的是，在不冲突的情况下，本申请中的实施例及实施例中的特征可以相互任意组合。

[0033] 在附图的流程图示出的步骤可以在诸如一组计算机可执行指令的计算机系统中执行。并且，虽然在流程图中示出了逻辑顺序，但是在某些情况下，可以以不同于此处的顺序执行所示出或描述的步骤。

[0034] 本申请主要针对Kaiming He等人提出的残差网络(resnet)模型进行优化，然而，其提出的resnet代码是基于串行传统卷积计算的代码，无法满足FPGA的OpenCL程序的并行模式的工作模式。本申请主要基于resnet-50代码对卷积计算快速算法(Fast Algorithms)

的FPGA平台设计实现与优化，以及以FPGA的OpenCL程序基于数据的并行模式的工作模式为例进行阐述。其中，resnet-50代码一共有50层卷积层，卷积核大小为3*3的卷积层一共16层，本申请基于Intel FPGA SDK for OpenCL来完成这16层代码实现，以对卷积计算的FPGA实现。

[0035] 图1为本发明实施例提供的卷积计算的实现方法的流程图，如图1所示，本发明实施例提供的卷积计算的实现方法，包括：

[0036] S101：获取待卷积计算的数据图像的输入通道数目M和输出通道数目N。

[0037] 其中，通道用于输入或输出图片，通道的数量对应于卷积计算时每一层输入或输出数据图像的个数。M和N均为正整数。

[0038] 具体的，基于resnet(残差网络)即可确定待卷积计算的数据图像的输入通道数目M和输出通道数目N；其确定的实现方式与现有技术相同，本实施例在此不进行赘述。

[0039] 需要说明的是，在S101之前，利用FPGA平台进行计算需要对设备与FPGA平台等信息进行初始化，主要包括：设备初始化，平台初始化，创建指令队列等。具体的可采用现有技术中调用Intel FPGA SDK for OpenCL中的接口函数完成初始化工作。

[0040] S102：将每一个输入通道的数据划分成大小为P*P的数据。

[0041] 具体的，对需要进行卷积核大小为m*m的卷积计算的每个输入通道的数据进行数据划分，大小为P*P。

[0042] S103：设置N个工作组，以及设置每个工作组中的工作项数目为P*P。

[0043] 具体的，本申请在计算每一个输入通道P*P的数据时，分为工作组和工作项，一个工作项对应一个线程；每一层卷积计算设置的工作组和工作项不一样，即每一层卷积计算设置的线程不一样。

[0044] S104：N个工作组并行计算，以及在每个工作组中，分别对每一输入通道进行P*P个工作项的卷积计算。

[0045] 具体的，图2为本发明实施例提供的数据划分的结构示意图，如图2所示，对于一个输入通道数目为M，输出通道数目为N的卷积计算，设置工作组为N，每个工作组设置的工作项数目为P*P，可以并行的对N个P*P的数据进行卷积计算。这样本地工作组参数为local_work＝{p,p}，全局工作组参数Global_work＝{p*N,p}，实现了基于OpenCL+FPGA对卷积计算Fast Algorithm程序的任务划分。

[0046] 需要说明的是，本发明实施例中对每个输入通道划分后的数据进行卷积计算，具体计算时可采用现有Fast Algorithms算法公式，也可以采用现有其他卷积计算公式，本发明实施例在此不进行限定和赘述。本发明实施例通过设置计算通道数据的工作组的个数为卷积计算中输出通道数目N，以及设置每个工作组中的工作项数目为P*P，P*P个工作项完成一个数据图像的一个通道的数据计算，每次可以处理N个P*P的数据，以实现对输入通道数据的并行卷积计算，满足FPGA的OpenCL程序的并行模式的工作模式，避免现有技术中基于串行传统卷积计算的resnet代码计算数据时每次只计算一个数据，计算完一个数据后再计算一个数据，需要计算N次，增加计算时间的缺陷。

[0047] 需要说明的是，每个工作项计算输出一个n*n的一个计算结果。

[0048] 其中，m，n和P均为正整数。

[0049] 可选的，P根据卷积计算的输入图片或输出图片大小确定。具体的，可根据输入图

片或输出图片大小和每个工作项计算输出的n*n的计算结果确定。比如，一个输出数据图像的大小为28*28，每个工作项计算输出的2*2的计算结果，则此时P可以设置为14。

[0050] 可选的，每个工作项处理的数据大小根据卷积计算时输入卷积核大小m*m中的m和输出结果n*n中的n确定，比如m＝3，n＝2，则每个工作项可处理4*4大小的数据。具体的，本发明实施例中每个工作项处理可以根据m和n的大小而定，以m＝3，n＝2为例，每个工作项处理的数据大小可以通过公式m+n-1获得。其中，m，n基于现有resnet代码以及Fast Algorithms等卷积算法可预先确定，其确定方法和实现原理与现有技术相同，本发明实施例在此不进行限定和赘述。

[0051] S105：将M个输入通道卷积计算后的数据进行加法规约，得到N个输出通道的输出结果。

[0052] 具体的，图3为本发明实施例提供的将M个输入通道卷积计算后的数据进行加法规约的结构示意图，如图3所示，每个工作项计算时对M个输入通道分别进行Fast Algorithms等卷积算法的计算，最终将结果进行加法规约。

[0053] 本发明实施例提供的卷积计算的实现方法，将每一个输入通道的数据进行划分，划分成大小为P*P的数据，设置N个工作组，以及设置每个工作组中的工作项数目为P*P，N个工作组并行计算，以及在每个工作组中，分别对每一输入通道的数据进行P*P个工作项的卷积计算，将M个输入通道卷积计算后的数据进行加法规约，得到N个输出通道的输出结果，实现了基于OpenCL+FPGA对卷积计算Fast Algorithm程序的任务划分，每次可以处理N个P*P的数据，以实现对数据的并行卷积计算，避免现有技术中基于串行传统卷积计算的resnet代码计算数据时每次只计算一个数据，计算完一个数据后再计算一个数据，需要计算N次，增加计算时间的缺陷。

[0054] 进一步地，在上述实施例中，本发明实施例可以通过以下一种或任意组合对FPGA程序进行优化：

[0055] 1、设置工作组中最大工作项数目。具体的，设置N个工作组，以及设置每个工作组中的工作项数目为P*P时，所述方法还包括：设置每个工作组中的最大工作项数目。

[0056] 可选的，根据P确定每个工作组中的最大工作项数目。具体的，卷积计算时不同计算层设置的P值不同，在设置每个工作组中的最大工作项数目时取所有P*P值的最大值。一般情况，编译器根据设置的最大工作项数目可进行自动优化，产生最优化的工作项数目。

[0057] 一般来说Intel FPGA SDK for OpenCL对程序的离线编译时设置默认的工作项为256，而本申请通过max_work_group_size()函数可以设置工作组中最大工作项数目，编译器根据设置的最大工作项数目可进行优化，产生最优化的工作项数目，从而能有效利用FPGA硬件资源，提高计算效率。比如，所有P*P值的最大值为28*28，则可设置一个工作组中最大工作项数目为28*28＝784，在代码中的kernel函数之前添加__attribute__((max_work_group_size(784)))，接口实现设置最大工作项，从而有效利用FPGA资源。

[0058] 2、设置多个计算单元。具体的，设置N个工作组，以及设置每个工作组中的工作项数目为P*P时，所述方法还包括：

[0059] 设置多个计算单元，以执行N个工作组的计算。其中，设置的多个计算单元是硬件单元的增加，多个计算单元执行N个工作组的计算，使得工作组中工作项计算的硬件资源增多了，从而加快计算速度。

[0060] 为了实现更高的吞吐量，Intel FPGA SDK for OpenCL离线编译可以为每个核函数产生多个计算单元。其中，核函数是一个可能独立执行特定功能的函数，核函数中的代码可以并行运行，本发明实施例的核函数可以执行每个工作组的功能。一般情况下，在编译kernel函数时，默认状态下编译器不会自动产生最优化的计算单元数目，而本申请通过num_compute_units()函数可以设置多个计算单元，编译器根据设置的计算单元的数量，产生最优化的计算单元数目，从而能有效提高程序访存吞吐量。计算单元数目可根据研发者多次试验的经验数据而定，比如，研发者多次试验的经验数据为8，则可设置计算单元数目为8，在代码中kernel函数之前添加__attribute__((num_compute_units(8)))函数，设置计算单元数目，以提高访存吞吐量。

[0061] 3、循环展开。具体的，在对每一输入通道的数据进行P*P个工作项的卷积计算时，以循环展开方式进行。

[0062] 其中，循环展开指的是将一个循环体中卷积计算的代码展开，增加了代码量从而增加了计算单元，减少了计算时间。

[0063] 对每一输入通道进行P*P个工作项的卷积计算进行循环展开，一个循环进行展开可以增加FPGA的硬件执行单元的数目，从而提高对硬件的利用率。以每个P处理4*4数据大小为例，代码中对Fast Algorithm的计算中会对一个4*4的矩阵进行点积计算，本申请可通过调用编译指导语句#pragma unroll，对这个4*4的矩阵点积计算循环进行展开完成循环展开，提高硬件利用率，减少了循环次数，从而提高计算效率。

[0064] 4、私有存储访问代替全局存储访问规约。具体的，在N个工作组并行计算，以及在每个工作组中，分别对每一输入通道的数据进行P*P个工作项的卷积计算；将M个输入通道卷积计算后的数据进行加法规约，得到N个输出通道每个工作组的输出结果时，将中间计算结果或变量存储在寄存器中，输出结果存储在全局内存中。

[0065] 具体的，将每一输入通道数据进行P*P个工作项的卷积计算结果，以及每个工作项进行卷积计算的计算结果存储在寄存器；将M个输入通道计算后的累加数据存储在全局内存中。

[0066] 在FPGA中，寄存器的是非常充足的，寄存器的访问速度要远远快于全局存储，尽量用寄存器存储代替全局存储能有效减少访存时间。图4为现有技术采用全局存储规约的代码示意图，如图4所示，现有技术代码中对各个通道的数据需要进行规约到存储在全局存储的输出结果中，这个过程需要多次访问输出的规约数据。图5为本发明实施例提供的局部存储规约的代码示意图，如图5所示，本申请利用寄存器暂时存储各个通道的数据，最终将各个通道的数据的累加结果数据写入全局存储的输出结果中。本发明实施例通过将每一输入通道数据进行P*P个工作项的卷积计算结果，以及每个工作项进行卷积计算的计算结果进行局部存储，将M个输入通道计算后的累加数据进行全局存储，能有效减少访存时间。

[0067] 对本发明实施例的技术效果可通过以下实验验证：

[0068] 1、运行环境

[0069] 表1给出了本发明实施例运行的软硬件环境，如表1所示，本发明实施例采用CPU+FPGA异构架构，其中主机端通用CPU为genuine intel 2.4G Hz，FPGA为Altera Arria 10。操作系统为Linux，编译语言采用OpenCL。

[0070] 表1

[0071]

硬件环境	软件环境
CPU：genuine intel 2.4G Hz FPGA: Altera Arria 10	操作系统：Linux SDK：Intel FPGA SDK for OpenCL

[0072] 2、实验内容：

[0073] 对rensnet-50代码的16个卷积核大小为3*3的卷积层进行卷积，待计算数据图像大小为224*224，输入通道为三通道，表2为本申请采用FPGA+CPU与现有技术采用CPU程序的运行时间对比，运行时间结果如表2。

[0074] 表2

[0075]

版本	CPU	CPU+FPGA
时间(ms)	960	188

[0076] 从表2可以看出，采用本申请上述实施例对FPGA程序进行实现与优化后，代码CPU+FPGA运行时间远远小于单CPU运行时间，说明本申请采用上述实施例对FPGA程序进行实现与优化能减少卷积计算时间，提高深度学习效率。

[0077] 本申请与现有技术相比具有以下有益效果：1)基于OpenCL+FPGA对卷积计算Fast Algorithm程序的任务划分，每次可以处理N个P*P的数据，以实现对数据的并行卷积计算。2)对核函数设置工作组中最大工作项为所有P*P的最大值，对计算点积的循环设置循环展开，能有效利用FPGA硬件资源，提高计算效率。3)用寄存器来优化核函数中各个输入图片通道的规约计算，减少全局访存时间。

[0078] 图6为本发明实施例一提供的卷积计算的实现设备的结构示意图，如图6所示，本发明实施例提供的卷积计算的实现设备，包括：获取模块61、划分模块62、设置模块63、计算模块64和累加模块65。

[0079] 获取模块61，用于获取待卷积计算的数据图像的输入通道数目M和输出通道数目N；

[0080] 划分模块62，用于将每一个输入通道的数据划分成大小为P*P的数据；

[0081] 设置模块63，用于设置N个工作组，以及设置每个工作组中的工作项数目为P*P；

[0082] 计算模块64，用于所述N个工作组并行计算，以及在每个工作组中，分别对每一输入通道的数据进行P*P个工作项的卷积计算；

[0083] 累加模块65，用于将M个输入通道卷积计算后的数据进行加法规约，得到N个输出通道的输出结果；

[0084] 其中，M、N和P均为正整数。

[0085] 本发明实施例提供的卷积计算的实现设备用于执行图1所示方法实施例的技术方案，其实现原理和实现效果类似，此处不再赘述。

[0086] 进一步地，所述P根据卷积计算的输入图片或输出图片大小确定。

[0087] 进一步地，所述设置模块63，还用于：

[0088] 设置每个工作组中的最大工作项数目。

[0089] 进一步地,所述设置模块63,还用于:

[0090] 设置多个计算单元。

[0091] 进一步地,所述计算模块64,在所述对每一输入通道的数据进行P*P个工作项的卷积计算时,以循环展开方式进行。

[0092] 进一步地,所述计算模块64,在N个工作组并行计算,以及在每个工作组中,分别对每一输入通道的数据进行P*P个工作项的卷积计算;将M个输入通道卷积计算后的数据进行加法规约,得到N个输出通道每个工作组的输出结果时,将中间计算结果或变量存储在寄存器中,所述输出结果存储在全局内存中。

[0093] 图7为本发明实施例二提供的卷积计算的实现设备的结构示意图,如图7所示,本发明实施例提供的卷积计算的实现设备,包括:存储器71和处理器72。

[0094] 存储器71用于存储执行指令,处理器72可以是一个中央处理器(Central Processing Unit,简称CPU),或者是特定集成电路(Application Specific Integrated Circuit,简称ASIC),或者完成实施本发明实施例的一个或多个集成电路。当主控设备运行时,处理器72与存储器71之间通信,处理器72调用执行指令,用于执行以下操作:

[0095] 获取待卷积计算的数据图像的输入通道数目M和输出通道数目N;

[0096] 将每一个输入通道的数据划分成大小为P*P的数据;

[0097] 设置N个工作组,以及设置每个工作组中的工作项数目为P*P;

[0098] 所述N个工作组并行计算,以及在每个工作组中,分别对每一输入通道的数据进行P*P个工作项的卷积计算;

[0099] 将M个输入通道卷积计算后的数据进行加法规约,得到N个输出通道的输出结果;

[0100] 其中,M、N和P均为正整数。

[0101] 进一步地,所述P根据卷积计算的输入图片或输出图片大小确定。

[0102] 进一步地,处理器72还用于:

[0103] 设置每个工作组中的最大工作项数目。

[0104] 进一步地,处理器还用于:

[0105] 在每个工作组中设置多个计算单元,以对每一输入通道的数据进行P*P个工作项的卷积计算。

[0106] 进一步地,处理器72在对每一输入通道的数据进行P*P个工作项的卷积计算时,以循环展开方式进行。

[0107] 进一步地,处理器72在N个工作组并行计算,以及在每个工作组中,分别对每一输入通道的数据进行P*P个工作项的卷积计算;将M个输入通道卷积计算后的数据进行加法规约,得到N个输出通道每个工作组的输出结果时,将中间计算结果或变量存储在寄存器中,所述输出结果存储在全局内存中。

[0108] 本发明实施例还提供一种计算机可读存储介质,其上存储有计算机指令,所述指令被处理器执行时实现上述任一实施例所述的卷积计算的实现方法的步骤。

[0109] 虽然本发明所揭露的实施方式如上,但所述的内容仅为便于理解本发明而采用的实施方式,并非用以限定本发明。任何本发明所属领域内的技术人员,在不脱离本发明所揭露的精神和范围的前提下,可以在实施的形式及细节上进行任何的修改与变化,但本发明

的专利保护范围,仍须以所附的权利要求书所界定的范围为准。

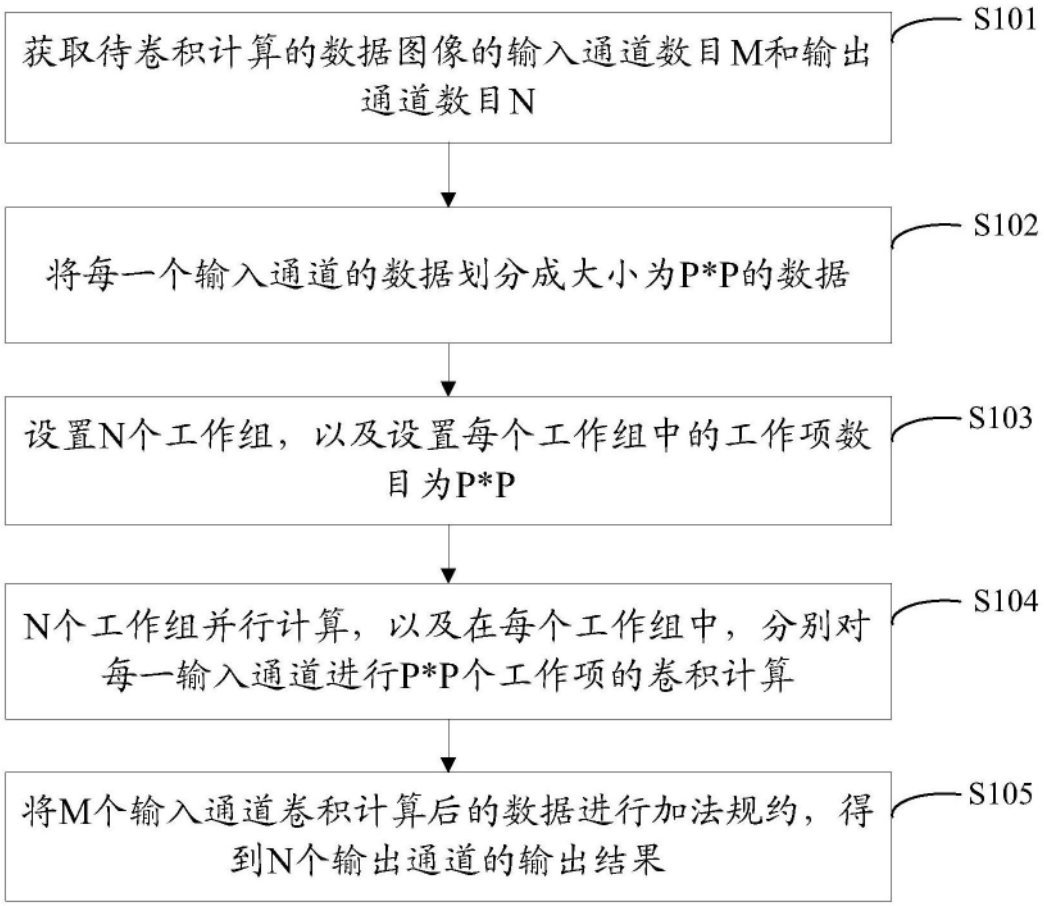

获取待卷积计算的数据图像的输入通道数目M和输出通道数目N
S101
将每一个输入通道的数据划分成大小为P*P的数据
S102
设置N个工作组，以及设置每个工作组中的工作项数目为P*P
S103
N个工作组并行计算，以及在每个工作组中，分别对每一输入通道进行P*P个工作项的卷积计算
S104
将M个输入通道卷积计算后的数据进行加法规约，得到N个输出通道的输出结果
S105

图1

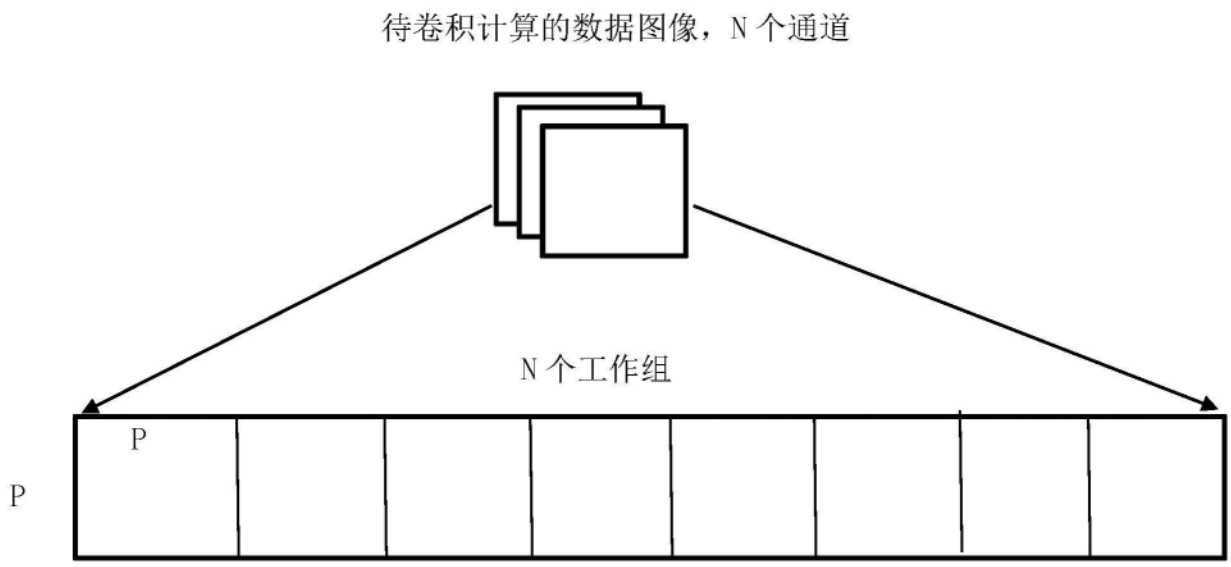

待卷积计算的数据图像，N 个通道
N 个工作组
P
P

图2

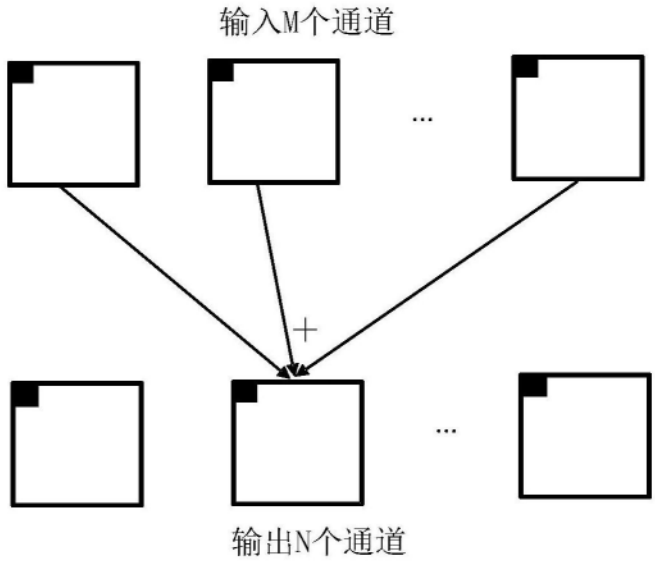


图3

```
for (int pre_mapid = 0; pre_mapid < num_pre_maps; pre_mapid++){
    //结果计算
    ......

    output[offset_out] += A_U_V[0] + A_U_V[1] + A_U_V[2];
    output[offset_out + 1] += A_U_V[1] - A_U_V[2] - A_U_V[3];
    output[offset_out + map_size + 1] += A_U_V[4] + A_U_V[5] + A_U_V[6];
    output[offset_out + map_size + 2]  += A_U_V[5] - A_U_V[6] - A_U_V[7];
}
```

图4

```
float tile_res[4] = {0};
for (int pre_mapid = 0; pre_mapid < num_pre_maps; pre_mapid++)  {
    //结果计算
    ......

    tile_res[0] += A_U_V[0] + A_U_V[1] + A_U_V[2];
    tile_res[1] += A_U_V[1] - A_U_V[2] - A_U_V[3];
    tile_res[2] += A_U_V[4] + A_U_V[5] + A_U_V[6];
    tile_res[3] += A_U_V[5] - A_U_V[6] - A_U_V[7];
}

output[offset_out] = tile_res[0];
output[offset_out + 1] = tile_res[1];
output[offset_out + map_size + 1] = tile_res[2];
output[offset_out + map_size + 2] = tile_res[3];
```

图5

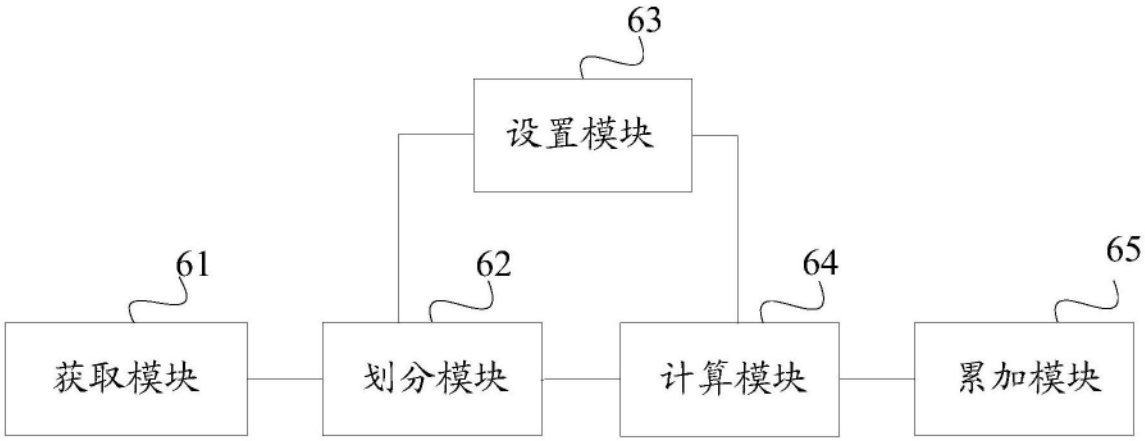


图6

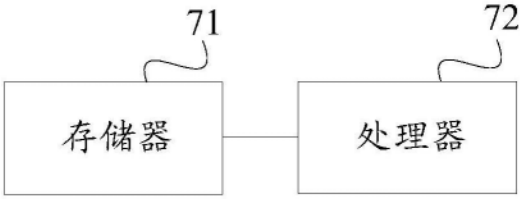


图7