(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 112149794 A
(43)申请公布日 2020.12.29

(21)申请号 202010466951.3
(22)申请日 2020.05.28
(30)优先权数据
2019-119018 2019.06.26 JP
(71)申请人 富士通株式会社
地址 日本神奈川县
(72)发明人 清水俊宏
(74)专利代理机构 北京集佳知识产权代理有限公司 11227
代理人 高岩 杨林森
(51)Int.Cl.
G06N 3/04(2006.01)
G06N 3/063(2006.01)
G06N 3/08(2006.01)
G06F 17/15(2006.01)
G06F 17/16(2006.01)

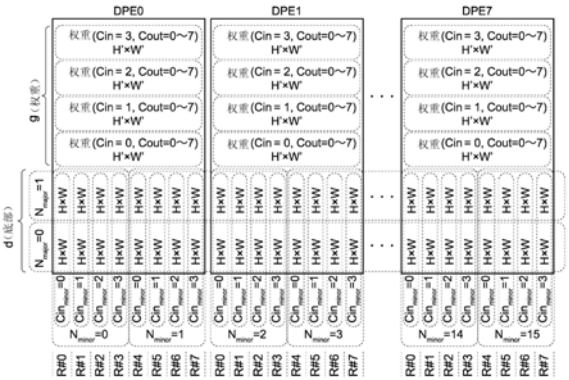
权利要求书2页 说明书26页 附图39页

(54)发明名称

信息处理设备、计算机可读存储介质和信息处理方法

(57)摘要

本发明涉及信息处理设备、计算机可读存储介质和信息处理方法。信息处理设备包括：计算单元，计算使当q个计算核使用Winograd算法并行计算多个第一矩阵与多个t行t列的第二矩阵之间的卷积时的计算时间最小化的t和q的组合，多个第一矩阵和多个第二矩阵的元素的总数不超过寄存器的q个存储区域中的各存储区域中可以存储的数据集的数目，q个计算核分别对应于q个存储区域；输出单元，输出用于使计算机器执行处理的程序，处理包括：使用t和q的组合将多个第一矩阵和多个第二矩阵存储在q个存储区域中的各存储区域中；由q个计算核中的每个使用Winograd算法计算第一矩阵与第二矩阵之间的卷积，计算机器包括q个计算核和寄存器。



CN 112149794 A

1.一种信息处理设备,包括:

计算单元,所述计算单元计算使当q个计算核使用Winograd算法并行计算多个第一矩阵与t行t列的多个第二矩阵之间的卷积时的计算时间最小化的t和q的组合,其中,所述多个第一矩阵和所述多个第二矩阵的元素的总数目不超过寄存器的q个存储区域中的各存储区域中能够存储的数据集的数目,并且所述q个计算核分别对应于所述q个存储区域;以及

输出单元,所述输出单元输出用于使计算机器执行处理的程序,所述处理包括:使用计算出的t和q的组合将所述多个第一矩阵和所述多个第二矩阵存储在所述q个存储区域中的各存储区域中,以及由所述q个计算核中的每一个使用所述Winograd算法计算所述第一矩阵与所述第二矩阵之间的卷积,所述计算机器包括所述q个计算核和所述寄存器。

2.根据权利要求1所述的信息处理设备,其中,

所述第一矩阵和所述第二矩阵中的每一个是深度学习的卷积层中的矩阵。

3.根据权利要求1所述的信息处理设备,其中,

当所述计算时间由第一函数f(t,q)表示并且存储在所述存储区域之一中的所述多个第一矩阵和所述多个第二矩阵的元素的总数目由第二函数g(t,q)表示时,所述计算单元计算在所述第二函数g(t,q)的值不超过一个存储区域中能够存储的数据集的数目的范围内使所述第一函数f(t,q)的值最小化的q和t的组合。

4.根据权利要求3所述的信息处理设备,其中,

所述第一矩阵和所述第二矩阵中的每一个是深度学习的卷积层中的矩阵,并且

所述深度学习的后向处理中的所述第一函数f(t,q)和所述第二函数g(t,q)分别不同于所述深度学习的前向处理中的所述第一函数f(t,q)和所述第二函数g(t,q)。

5.根据权利要求1所述的信息处理设备,其中,

所述多个第二矩阵中的每一个通过第一标识符和第二标识符的组合来标识,并且

所述程序使所述计算机器执行将q个第二矩阵中的各第二矩阵存储在所述q个存储区域中的各存储区域中的处理,其中,所述q个第二矩阵的所述第一标识符彼此不同,并且所述q个第二矩阵的所述第二标识符相同。

6.根据权利要求5所述的信息处理设备,其中,

所述程序使所述计算机器执行包括以下操作的处理:

将所述第一标识符彼此相同的所述第一矩阵和所述第二矩阵存储在相同的存储区域中,并且

计算存储在所述相同的存储区域中的所述第一矩阵与所述第二矩阵之间的卷积。

7.根据权利要求1所述的信息处理设备,其中,

所述程序使所述计算机器执行包括以下操作的处理:

针对所述多个存储区域中的每一个来计算所述元素的值的平均值和离差,以及

针对所述多个存储区域中的每一个通过将所述元素的值与所述平均值之间的差除以所述离差来归一化所述元素的值。

8.一种非暂态计算机可读存储介质,其存储有使计算机执行处理的信息处理程序,所述处理包括:

计算使当q个计算核使用Winograd算法并行计算多个第一矩阵与t行t列的多个第二矩阵之间的卷积时的计算时间最小化的t和q的组合,其中,所述多个第一矩阵和所述多个第

二矩阵的元素的总数目不超过寄存器的q个存储区域中的各存储区域中能够存储的数据集的数目,并且所述q个计算核分别对应于所述q个存储区域;以及

输出用于使计算机器执行处理的程序,所述处理包括:使用计算出的t和q的组合将所述多个第一矩阵和所述多个第二矩阵存储在所述q个存储区域中的各存储区域中,以及由所述q个计算核中的每一个使用所述Winograd算法计算所述第一矩阵与所述第二矩阵之间的卷积,所述计算机器包括所述q个计算核和所述寄存器。

9.一种通过计算机实现的信息处理方法,所述信息处理方法包括:

计算使当q个计算核使用Winograd算法并行计算多个第一矩阵与t行t列的多个第二矩阵之间的卷积时的计算时间最小化的t和q的组合,其中,所述多个第一矩阵和所述多个第二矩阵的元素的总数目不超过寄存器的q个存储区域中的各存储区域中能够存储的数据集的数目,并且所述q个计算核分别对应于所述q个存储区域;以及

输出用于使计算机器执行处理的程序,所述处理包括:使用计算出的t和q的组合将所述多个第一矩阵和所述多个第二矩阵存储在所述q个存储区域中的各存储区域中,以及由所述q个计算核中的每一个使用所述Winograd算法计算所述第一矩阵与所述第二矩阵之间的卷积,所述计算机器包括所述q个计算核和所述寄存器。

信息处理设备、计算机可读存储介质和信息处理方法

技术领域

[0001] 本文描述的实施方式的某些方面涉及信息处理设备、非暂态计算机可读存储介质和信息处理方法。

背景技术

[0002] 使用多层神经网络的机器学习被称为深度学习，并且被应用于各种领域。在深度学习的每一层中执行各种计算。例如，在卷积层中，执行图像数据与滤波器之间的卷积，并且将其结果输出至后续层。由于卷积是矩阵之间的运算，因此其计算量大，从而导致学习的处理速度的延迟。因此，提出了Winograd算法作为用于减少卷积计算量的算法。注意，与本公开内容有关的技术还在以下文献中公开：“Fast Algorithms for Convolutional Neural Networks”，Andrew Lavin等人，IEEE计算机视觉和模式识别(CVPR)会议，2016年，第4013至4021页；以及“Deep Residual Learning for Image Recognition”，Kaiming He等人，IEEE计算机视觉和模式识别(CVPR)会议，2016年，第770至778页。

[0003] 然而，就进一步提高卷积的处理速度而言，Winograd算法具有改进的空间。

发明内容

[0004] 鉴于这些情况做出了本发明，并且本发明的目的是提高卷积的计算速度。

[0005] 根据实施方式的方面，提供了一种信息处理设备，包括：计算单元，其用于计算使当q个计算核使用Winograd算法并行计算多个第一矩阵与多个t行t列的第二矩阵之间的卷积时的计算时间最小化的t和q的组合，其中，多个第一矩阵和多个第二矩阵的元素的总数目不超过寄存器的q个存储区域中的各存储区域中可以存储的数据集的数目，并且q个计算核分别对应于q个存储区域；以及输出单元，其用于输出用于使计算机器执行处理的程序，所述处理包括：使用计算出的t和q的组合将多个第一矩阵和多个第二矩阵存储在q个存储区域中的各存储区域中，以及由q个计算核中的每一个使用Winograd算法计算第一矩阵与第二矩阵之间的卷积，计算机器包括q个计算核和寄存器。

附图说明

[0006] 图1示意性地示出了深度学习的处理流程；

[0007] 图2示意性地示出了在卷积层中执行的卷积；

[0008] 图3A至图3C示意性地示出了底部矩阵与权重矩阵之间的卷积；

[0009] 图4A至图4C示意性地示出了前向处理中的Winograd算法；

[0010] 图5是用于在深度学习中执行卷积的计算机器的硬件配置图；

[0011] 图6A是一个DPU链的硬件配置图，以及图6B是一个DPU的硬件配置图；

[0012] 图7是每个DPE的硬件配置图；

[0013] 图8是DPE0的硬件配置图；

[0014] 图9是用于描述分配给存储体R#0至R#7的行号的图；

[0015] 图10A至图10C是用于描述顺序方法的(第一)示意图；

[0016] 图11A至图11C是用于描述顺序方法的(第二)示意图；

[0017] 图12是用于描述多播方法的示意图；

[0018] 图13示意性地示出了每个DPE的寄存器G#0的内容；

[0019] 图14示意性地示出了主存储器中的数组g的数组元素；

[0020] 图15示出了在通过多播方法传送了数组元素紧之后DPE0的寄存器G#0的内容；

[0021] 图16示出了排序之后DPE0的寄存器G#0的内容；

[0022] 图17示出了排序之后DPE0至DPE7的寄存器G#0的内容；

[0023] 图18是DPE0的寄存器G#0的存储体R#0的示意图；

[0024] 图19是根据实施方式的信息处理设备的硬件配置图；

[0025] 图20是根据实施方式的信息处理设备的功能配置图；

[0026] 图21是计算机器的功能框图；

[0027] 图22示出了在实施方式中当执行前向处理时由存储单元将数组d和g存储于的DPE0至DPE7的寄存器G#0的内容；

[0028] 图23A和图23B是示出在实施方式中当计算单元使用Winograd算法执行卷积时DPE0的寄存器G#0至G#3的内容的(第一)图；

[0029] 图24是示出在实施方式中当计算单元使用Winograd算法执行卷积时DPE0的寄存器G#0至G#3的内容的(第二)图；

[0030] 图25是示出在实施方式中当计算单元使用Winograd算法执行卷积时DPE0的寄存器G#0至G#3的内容的(第三)图；

[0031] 图26是按步骤顺序示出实施方式的等式(19)的计算的示意图；

[0032] 图27是按步骤顺序示出实施方式的等式(21)的计算的示意图；

[0033] 图28是根据实施方式的信息处理方法的流程图；

[0034] 图29A至图29C是根据实施方式在后向处理中使用Winograd算法执行顶部矩阵与权重矩阵之间的卷积时的示意图；

[0035] 图30示出了根据实施方式由存储单元将数组y和g存储于的DPE0至DPE7的寄存器G#0的内容；

[0036] 图31A和图31B是根据实施方式在后向处理中使用Winograd算法执行的顶部矩阵与底部矩阵之间的卷积的示意图；

[0037] 图32A至图32C是根据实施方式在后向处理中使用Winograd算法执行的顶部矩阵与底部矩阵之间的卷积的示意图；

[0038] 图33是示出根据实施方式由存储单元将数组y和d存储于的DPE0至DPE7的寄存器G#0的内容的图；

[0039] 图34示出了在实施方式中当执行1×1卷积时由存储单元将数组d和g存储于的DPE0的寄存器G#0的内容；

[0040] 图35示出了在批量归一化期间根据实施方式由存储单元将子底部矩阵d存储于的DPE0的寄存器G#0的内容;以及

[0041] 图36A和图36B示出了DPE0的寄存器G#0的内容,并且是用于描述在批量归一化期间根据实施方式由计算单元执行的计算的图。

具体实施方式

[0042] 在描述实施方式之前，将描述发明人研究的项目。

[0043] 图1示意性地示出了深度学习的处理流程。在深度学习中，神经网络通过对识别目标例如图像的监督学习来学习识别目标的特征。使用学习之后的神经网络使得识别目标能够被识别。

[0044] 神经网络是模仿大脑的神经元的单元被分层连接的网络。每个单元从另一单元接收数据，并将数据传送至又一单元。在神经网络中，可以通过经由学习改变单元的参数来识别各种识别目标。

[0045] 在下文中，参照图1，将描述用于图像的识别的卷积神经网络(CNN)。

[0046] 该神经网络具有包括卷积层、子采样层和全连接层的多层结构。在图1的示例中，交替地布置两个卷积层和两个子采样层，但是可以提供三个或更多个卷积层和三个或更多个子采样层。此外，可以提供多个全连接层。神经网络的多层结构和每层的配置可以由设计人员根据要识别的目标预先确定。

[0047] 通过神经网络识别图像的处理也称为前向处理。在前向处理中，如图1所示，从左到右交替地重复卷积层和池化层。然后，最后，在全连接层中识别图像中包括的识别目标。

[0048] 此外，通过神经网络学习图像的处理也称为后向处理。在后向处理中，获得识别结果与正确答案之间的误差，并使获得的误差通过神经网络从右向左向后传播以改变卷积神经网络的各层的参数。

[0049] 图2示意性地示出了在卷积层中执行的卷积。

[0050] 图2示出了底部矩阵与权重矩阵之间的卷积，在底部矩阵中输入图像的像素数据存储在每个元素中，权重矩阵表示作用在输入图像上的滤波器。在该示例中，准备了多个底部矩阵和多个权重矩阵，并且执行它们之间的卷积。

[0051] 每个底部矩阵由批号N和输入通道号Cin标识。另一方面，每个权重矩阵由输出通道号Cout和输入通道号Cin来标识。

[0052] 在图2的示例中，卷积如下所述地执行。首先，选择批号N和输出通道号Cout的一个组合。例如，N＝0并且Cout＝0。

[0053] 然后，从具有所选择的批号N的多个底部矩阵和具有所选择的输出通道号Cout的多个权重矩阵的组合之中，选择具有相同输入通道号Cin的底部矩阵和权重矩阵的组合。例如，当如上所述N＝0并且Cout＝0时，选择N＝0并且Cin＝0的底部矩阵以及Cout＝0并且Cin＝0的权重矩阵。

[0054] 然后，执行所选择的底部矩阵与所选择的权重矩阵之间的卷积。在下文中，通过该卷积获得的矩阵称为顶部矩阵。

[0055] 通过在批号N和输出通道号Cout固定的情况下在Cin＝0至255的底部矩阵与权重矩阵之间执行这样的卷积，获得256个顶部矩阵。此后，通过将这256个顶部矩阵相加，获得由批号N和输出通道号Cout标识的一个输出矩阵。

[0056] 此外，通过在改变批号N和输出通道号Cout的情况下执行上面的计算，获得批号N的总数×输出通道号Cout的总数的输出矩阵。在图2的示例中，获得了64×384个输出矩阵。

[0057] 以前述方式，执行多个底部矩阵与多个权重矩阵之间的卷积。

[0058] 如上所述，在这样的卷积中，计算具有相同输入通道号Cin的底部矩阵与权重矩阵

之间的卷积。因此，将详细描述这些矩阵之间的卷积。

[0059] 图3A至图3C示意性地示出了底部矩阵与权重矩阵之间的卷积。

[0060] 首先，如图3A所示，准备要经受卷积的底部矩阵与权重矩阵。在该示例中，底部矩阵是13×13的方形矩阵，权重矩阵是3×3的方形矩阵。

[0061] 然后，如图3B所示，通过在底部矩阵周围填充零获得15×15的矩阵M。

[0062] 然后，如图3C所示，在矩阵M中提取大小与权重矩阵相同的子矩阵P_{ij}。在下文中，子矩阵P_{ij}的第k行第l列中的元素由$(P_{ij})_{kl}$(0≤k,l≤2)表示，权重矩阵的第k行第l列中的元素由g_{kl}(0≤k,l≤2)表示。

[0063] 此外，如上所述，通过矩阵M与权重矩阵之间的卷积获得的矩阵称为顶部矩阵。在这种情况下，可以通过下面的等式(1)来计算顶部矩阵中的每个元素r_{ij}。

[0064] $$r_{ij}=\sum_{k,l=0}^{2}(P_{ij})_{kl}g_{kl}\cdots(1)$$

[0065] 然而，在该方法中，为了获得顶部矩阵中的一个元素r_{ij}，需要将乘法执行与权重矩阵的元素的数目(即，3×3)一样多的次数。因此，不可能提高卷积的计算速度。

[0066] 已知Winograd算法是提高卷积计算速度的算法。因此，下面将描述Winograd算法。

[0067] 如上所述，深度学习中存在前向处理和后向处理。这里，将描述前向处理中的Winograd算法。

[0068] 图4A至图4C示意性地示出了前向处理中的Winograd算法。

[0069] 首先，如图4A所示，从底部矩阵分割出t×t的子底部矩阵d。这里，t是自然数。然后，根据下面的等式(2)获得子顶部矩阵y。

[0070] $y=A^{T}\{(GgG^{T})\odot(B^{T}dB)\}A\cdots(2)$

[0071] 子顶部矩阵y是形成顶部矩阵的一部分的矩阵。

[0072] 等式(2)中的B、G和A是常数矩阵。这些常数矩阵B、G和A的元素和大小根据每个矩阵g、d的大小而变化。例如，当权重矩阵g的大小为3×3并且子底部矩阵d的大小为4×4时，每个常数矩阵B、G、A的元素和大小由下面的等式(3)表示。

$$B^{T}=\begin{pmatrix}1&0&-1&0\\0&1&1&0\\0&-1&1&0\\0&1&0&-1\end{pmatrix}$$

[0073] $$G=\begin{pmatrix}1&0&1\\\frac{1}{2}&\frac{1}{2}&\frac{1}{2}\\\frac{1}{2}&-\frac{1}{2}&\frac{1}{2}\\0&0&1\end{pmatrix}\qquad\cdots(3)$$

$$A^{T}=\begin{pmatrix}1&1&1&0\\0&1&-1&-1\end{pmatrix}$$

[0074] 等式(2)中的运算符“⊙”表示矩阵的逐元素(element-wise)乘法。例如，当具有相同维数的任意矩阵U和V中的每一个矩阵的元素分别由u_{ij}和v_{ij}表示，并且$U \odot V$的ij元由$(U \odot V)_{ij}$表示时，$(U \odot V)_{ij} = u_{ij}v_{ij}$。

[0075] 然后，如图4B所示，将从底部矩阵分割子底部矩阵d的位置从图4A情况下的位置移动两列，并且分割的子底部矩阵d经历与上面的计算相同的计算。所获得的子顶部矩阵y在顶部矩阵中形成与在图4A中获得的子顶部矩阵y相邻的块。

[0076] 如上所述，通过将从底部矩阵分割子底部矩阵d的位置重复移动两列和两行，获得如图4C所示由子顶部矩阵y形成的顶部矩阵。

[0077] 通过上面的处理，完成了使用Winograd算法的底部矩阵与权重矩阵之间的卷积。

[0078] 在等式(2)的Winograd算法中，一旦矩阵GgG^T和矩阵B^TdB形成，就可以高速计算卷积，因为可以仅通过计算矩阵GgG^T和矩阵B^TdB的逐元素乘积来执行卷积。

[0079] 发明人针对与上面的示例一样的权重矩阵g的大小为3×3并且子底部矩阵d的大小为4×4的情况计算了计算时间。在不使用Winograd算法的图3A至图3C的示例中，计算出的计算时间为1152个周期。注意，“周期”的数目等同于将数据写入寄存器的次数。

[0080] 另一方面，当使用Winograd算法时，计算时间为940个周期，并且结果表明，计算速度相对于图3A至图3C的示例中的计算速度增加了1.23(=1152/940)倍。

[0081] 接下来，将描述使用Winograd算法执行卷积的计算机器。

[0082] 图5是用于在深度学习等中执行卷积的计算机器的硬件配置图。

[0083] 如图5所示，计算机器10包括通过总线13互连的主存储器11和处理器12。

[0084] 主存储器11是临时存储数据并与处理器12协作执行各种程序的设备，例如动态随机存取存储器(DRAM)。

[0085] 另一方面，处理器12是包括计算单元例如算术逻辑单元(ALU)的硬件设备。在该示例中，深度学习单元(DLU:注册商标)被用作处理器12。DLU是具有适合于深度学习的架构的处理器，并且包括八个深度学习处理单元(DPU)链14。

[0086] 图6A是一个DPU链14的硬件配置图。

[0087] 如图6A所示，DPU链14包括四个DPU 15。在这些DPU 15中的每一个中执行并行计算，如稍后所描述的。

[0088] 图6B是一个DPU 15的硬件配置图。

[0089] 如图6B所示，DPU 15包括16个深度学习处理元件(DPE)0至15。图7是每个DPE的硬件配置图。

[0090] 尽管如图6B所示DPE的总数为16，但是在下文中，将仅描述DPE0至DPE7。

[0091] 如图7所示，DPE0至DPE7中的每一个包括八个计算核C#0至C#7以及可以由计算核C#0至C#7读取/写入的寄存器文件20。

[0092] 计算核C#0至C#7是单独的单指令多数据(SIMD)计算单元，并且可以在计算核C#0至C#7中执行并行计算。

[0093] 另一方面，寄存器文件20经由总线13(参见图5)耦合至主存储器11，寄存器文件20中存储有从主存储器11读取的数据，并且存储有由计算核C#0至C#7进行的计算的结果。

[0094] 在该示例中，寄存器文件20被划分为四个寄存器G#0至G#3，四个寄存器G#0至G#3

被配置成可以并行地读取/写入。例如，当寄存器G#0从主存储器11读取数据时，可以与由寄存器G#0读取数据并行地将由计算核C#0至C#7进行的计算的结果存储在寄存器G#1中。

[0095] 图8是DPE0的硬件配置图。由于DPE1至DPE15与DPE0具有相同的硬件配置，因此省略其描述。图8仅示出了寄存器文件20的寄存器G#0至G#3中的寄存器G#0的硬件配置。其他寄存器G#1至G#3与寄存器G#0具有相同的硬件配置。

[0096] 如图8所示，寄存器G#0包括八个存储体R#0至R#7。存储体R#0至R#7中的每一个是存储区域的示例，并且被提供以对应于计算核C#0至C#7中的每一个。例如，存储体R#0是与计算核C#0相对应的存储区域。当计算核C#0执行计算时，计算核C#0读取存储体R#0中的数据，或者计算核C#0将计算结果写入存储体R#0中。

[0097] 图9是用于描述分配给存储体R#0至R#7的行号的图。

[0098] 行号是用于标识存储体R#0至R#7的每个条目的标识符。在该示例中，使用了128个行号：L_0至L_{127}。每个条目中存储的数据没有特别限制。在该示例中，浮点数据存储在一个条目中。因此，可以在存储体R#0中存储127组浮点数据。这同样适用于存储体R#1至R#7。

[0099] 当执行深度学习的卷积时，要经受卷积的矩阵的元素存储在每个条目中。在这种情况下，矩阵的元素在主存储器11中存储为数组元素。

[0100] 这里，将给出用于将存储在主存储器11中的数组元素扩展至DPE0至DPE7的扩展方法的描述。

[0101] 存在作为扩展方法的顺序方法和多播方法。首先，将描述顺序方法。

[0102] 图10A至图11C是用于描述顺序方法的示意图。

[0103] 在该示例中，存储在主存储器11中的数组元素a[0]、a[1]、a[2]、…、a[127]被扩展到DPE0至DPE7。

[0104] 在这种情况下，如图10A所示，第一数组元素a[0]存储在DPE0的存储体R#0中的由行号L_0标识的条目中。

[0105] 然后，如图10B所示，在不改变行号L_0的情况下，将下一数组元素a[1]存储在与存储体R#0相邻的存储体R#1中。

[0106] 如图10C所示，以相同的方式，在不改变行号L_0的情况下，将数组元素连续地存储在彼此相邻的存储体中。因此，DPE0至DPE7的存储体R#0到R#7中的由行号L_0标识的条目填充有数组元素a[0]、a[1]、a[2]、…a[63]。

[0107] 此后，如图11A所示，将下一数组元素a[64]存储在DPE0的存储体R#0中的由行号L_1标识的条目中。

[0108] 然后，如图11B所示，在不改变行号L_1的情况下，将下一数组元素a[65]存储在下一存储体R#1中。

[0109] 此外，在不改变行号L_1的情况下，将数组元素连续地存储在彼此相邻的存储体中。因此，如图11C所示，DPE0至DPE7的存储体R#0至R#7中的由行号L_1标识的条目填充有数组元素a[64]、a[65]、a[66]、…、a[127]。

[0110] 通过上面的处理，通过顺序方法将数组元素a[0]、a[1]、a[2]、…、a[127]扩展到DPE0至DPE7。根据上面描述的顺序方法，DPE0至DPE7的具有相同行号L_i的条目被顺序地填充，并且在行号L_i的最后一个条目被填充时，将数组元素存储在具有下一行号L_{i+1}的条目中。

[0111] 接下来，将描述多播方法。图12是用于描述多播方法的示意图。

[0112] 在该示例中，存储在主存储器11中的数组元素a[0]、a[1]、a[2]、…、a[23]被扩展到DPE0至DPE7。

[0113] 在多播方法中，将数组元素a[0]、a[1]、a[2]、…、a[23]顺序地存储在DPE0中。以相同的方式，将数组元素a[0]、a[1]、a[2]、…、a[23]存储在DPE1至DPE7的每一个中。在这种方法中，相同的数组元素被存储在DPE0至DPE7的每一个中。

[0114] 然后，将描述当计算机器10利用Winograd算法执行卷积时寄存器的内容。

[0115] 图13示意性地示出了每个DPE的寄存器G#0的内容。

[0116] 在下文中，将使用与表示矩阵的符号相同的符号来表示存储该矩阵的元素的数组。例如，由d表示存储t×t的底部矩阵d的元素的数组，由g表示存储3×3的权重矩阵g的元素的数组。

[0117] 此外，这些数组d和g由下面的表达式(4)表示。

[0118] $$\begin{matrix} d[Cin][H][W][N] \\ g[Cout][Cin][H'][W'] \end{matrix} \cdots (4)$$

[0119] 在表达式(4)中，N是具有0至63的值的批号。Cin是具有0至255的值的输入通道号，以及Cout是具有0至383的值的输出通道号。

[0120] H和W中的每一个都是标识一个底部矩阵中的元素的变量。类似地，H′和W′中的每一个都是标识一个权重矩阵中的元素的变量。

[0121] 在这种情况下，通过顺序方法将数组d扩展到DPE0至DPE7的寄存器G#0。

[0122] 在诸如数组d的多维数组的情况下，从最低级别的数组元素开始将数组元素依次存储在寄存器G#0中。数组d中的最低级别的元素由批号N标识。因此，批号N为0、1、…、7的数组元素被顺序地分别存储在DPE0的存储体R#0、R#1、…、R#7中。然后，批号N为8、9、...、15的数组元素被顺序地分别存储在DPE1的存储体R#0、R#1、...、R#7中。以这种方式，将批号N为0至63的元素扩展到DPE0至DPE7。

[0123] 此外，在数组d[Cin][H][W][N]中，由Cin、H和W标识的较高级别的元素被如下所述地处理。

[0124] 首先，如图4A所示，固定从底部矩阵分割t×t的子底部矩阵d的位置，并且然后，将分割的子底部矩阵d的t×t个元素存储在[H][W]中。另外，对于Cin，选择0至255的值中的0至3。

[0125] 因此，对应于Cin＝0的t×t个矩阵元素被扩展到DPE0至DPE7。类似地，对应于Cin＝1、Cin＝2和Cin＝3中的每一个的t×t个矩阵元素也被扩展到DPE0至DPE7。

[0126] 另一方面，通过多播方法将数组g扩展到DPE0至DPE7中的每一个的寄存器G#0。

[0127] 在该示例中，以输入通道号Cin为单位对Cout的值为0至7的数组元素进行多播。例如，将Cout的值为0至7的数组元素之中的Cin＝0的元素多播到DPE0至DPE7中的每一个。类似地，通过多播将Cin＝1、Cin＝2、Cin＝3的数组元素传送到DPE0至DPE7。

[0128] 然而，当如上所述通过多播方法传送数组g时，DPE0的存储体R#0中的输入通道号Cin的值与输出通道号Cout的值之间的规则性丢失。这使得与存储体R#0相对应的计算核C#0使用Winograd算法对数组g和d进行卷积不方便。这同样适用于计算核C#1至C#7和DPE1至DPE7。因此，将数组g的元素如下所述排序。

[0129] 图14示意性地示出了主存储器11中的数组g的数组元素。

[0130] 如上所述，数组g是表示权重矩阵的数组，并且对应于3×3的方形矩阵。因此，在下文中，将编号0、1、…、8分配给3×3的方形矩阵的相应元素，以通过分配的编号来标识每个元素。

[0131] 因此，当如表达式(4)那样将数组g描述为g[Cout][Cin][H'][W']时，将编号0、1、…、8分配给[H']和[W']中的每一个。

[0132] 图15示出了在通过上述多播方法传送了数组元素紧之后DPE0的寄存器G#0的内容。

[0133] 如图15所示，当通过多播方法传送数组元素时，从g[Cout][Cin][H'][W']的较低级别的元素开始依次用g[Cout][Cin][H'][W']的元素填充存储体R#0至R#7的第一行。然后，最后一个存储体R#7的第一行被填充，依次填充第二行。

[0134] 权重矩阵g的元素的数目为九，而存储体R#0至R#7的数目为八。因此，两者的数目不匹配。因此，当如上所述通过多播方法将矩阵元素传输至寄存器时，Cin＝0和Cout＝0的九个元素跨两行存储在寄存器中。这同样适用于Cin和Cout的其他组合。

[0135] 因此，具有不同的Cin和Cout值的各种数组元素存储在存储体R#0中，从而导致存储体R#0中Cin与Cout之间的规则性降低。

[0136] 因此，在该示例中，DPE0的计算核C#0至C#7中的每一个使用DPE0的其余寄存器G#1至G#3之一作为缓冲器来对寄存器G#0中的数组g的元素进行排序。

[0137] 图16示出了排序之后DPE0的寄存器G#0的内容。

[0138] 如图16所示，通过排序，将具有相同Cout值的元素存储在同一存储体中。例如，仅Cout＝0的元素被存储在存储体R#0中。

[0139] 图17示出了如上所述的排序之后的DPE0至DPE7中的每一个的寄存器G#0的内容。

[0140] 如图17所示，例如，将数组g的Cout＝0且Cin＝0至3的元素存储在DPE0的存储体R#0中。此外，将数组d的N＝0且Cin＝0至3的元素存储在存储体R#0中。

[0141] 这使得存储体R#0中的数组d和g的Cin的值相同，从而使得计算核C#0能够根据Winograd算法在具有相同的Cin值的数组d与g之间执行卷积。

[0142] 存储体R#0至R#7中的每一个与批号N一一对应，并且在存储体R#0至R#7中执行关于不同批号的卷积。这同样适用于其他DPE1至DPE7。

[0143] 因此，期望通过DPE0至DPE7中的每一个的计算核C#0至C#7并行执行上述卷积来高速执行深度学习的前向处理和后向处理。

[0144] 然而，发明人进行的研究表明，使存储体R#0至R#7中的每一个与批号N一一对应的方法具有下面的问题。

[0145] 图18是用于描述该问题的图，并且是DPE0的寄存器G#0的存储体R#0的示意图。

[0146] 在该示例中，使每个存储体R#0至R#7与批号N一一对应，并且具有相同输入通道号Cin的子底部矩阵d和权重矩阵g被存储在一个存储体中。因此，有必要在一个存储体中存储相同数目的子底部矩阵d和权重矩阵，而如果子底部矩阵d的大小增加，则子底部矩阵d的元素从存储体中溢出。

[0147] 例如，考虑如图18所示在存储体R#0中存储四个子底部矩阵d和四个权重矩阵g的情况。子底部矩阵d的大小为t×t，权重矩阵g的大小为3×3。因此，要存储在存储体R#0中的

元素的数目为$4\times t^2+4\times 3^2$。如上所述，由于可以存储在一个存储体中的数据集的数目为127，因此t需要为4或小于4，以使得元素的数目不超过127。

[0148] 当t小时，通过等式(2)获得的子顶部矩阵y的大小变得小。因此，需要计算大量的子顶部矩阵y以获得顶部矩阵，从而导致卷积所需的计算时间增加。作为结果，不能充分利用Winograd算法的可以提高卷积的计算速度的特性。

[0149] 下面将描述可以高速计算卷积的实施方式。

[0150] 实施方式

[0151] 图19是根据实施方式的信息处理设备31的硬件配置图。

[0152] 信息处理设备31是用于生成可以由计算机器10(参见图5)执行的程序的计算机例如个人计算机(PC)，并且包括存储设备32、主存储器33、处理器34、输入设备35和显示设备36。这些部件通过总线37彼此连接。

[0153] 存储设备32是辅助存储设备，例如但不限于硬盘驱动器(HDD)或固态驱动器(SSD)，并且存储根据实施方式的信息处理程序39。

[0154] 信息处理程序39的执行使得能够如稍后所描述地生成可由计算机器10(参见图5)执行的程序。

[0155] 应当注意的是，信息处理程序39可以存储在可由计算机读取的存储介质38中，并且可以使处理器34读取存储介质38中的信息处理程序39。

[0156] 存储介质38的示例包括物理便携式存储介质，例如但不限于致密盘只读存储器(CD-ROM)、数字多功能盘(DVD)和通用串行总线(USB)存储器。可替选地，半导体存储器例如闪存或硬盘驱动器可以用作存储介质38。这些存储介质38不是诸如没有物理形式的载波的临时存储介质。

[0157] 可替选地，信息处理程序39可以存储在连接至公共网络、因特网或局域网(LAN)的设备中，并且处理器34可以读取信息处理程序39并执行它。

[0158] 另一方面，主存储器33是临时存储数据的硬件设备，例如动态随机存取存储器(DRAM)，并且信息处理程序39被扩展在主存储器33上。

[0159] 处理器34是控制信息处理设备31的各个部件并且与主存储器33协作地执行信息处理程序39的硬件设备，例如中央处理单元(CPU)。

[0160] 输入设备35是诸如由用户操作的键盘和鼠标的输入设备。显示设备36是显示在信息处理程序39的执行期间由用户使用的各种命令的显示设备，例如液晶显示器。

[0161] 图20是根据实施方式的信息处理设备31的功能框图。如图20所示，信息处理设备31包括输出单元41和计算单元42。每个单元通过在处理器34与主存储器33之间协作执行上述信息处理程序39来实现。

[0162] 输出单元41是生成可以由计算机器10(参见图5)执行的程序50的功能块。该程序可以是写入中间代码的文件或可执行二进制文件。

[0163] 计算单元42是优化程序50中的各种参数的功能块。参数的示例包括要如图4A至图4C所示从底部矩阵分割的子底部矩阵d的大小t。另外，稍后描述的存储体的数目q是要优化的参数的示例。

[0164] 图21是通过执行程序50实现的计算机器10的功能框图。

[0165] 如图21所示，计算机器10包括接收单元51、选择单元52、存储单元53、计算单元54

和输出单元55。这些单元通过在图5中的主存储器11与DLU 12之间协作执行程序50来实现。

[0166] 接收单元51接收底部矩阵和权重矩阵的输入。选择单元52如图4A至图4C所示从底部矩阵中选择t×t的子底部矩阵d。如上所述，通过计算单元42优化大小t的值，并且选择单元52通过使用优化的大小t来选择子底部矩阵d。

[0167] 存储单元53将子底部矩阵d和权重矩阵g中的每一个中的元素存储在DPE0至DPE7的存储体R#0至R#7中。

[0168] 计算单元54通过使用存储在存储体R#0至R#7中的元素来计算卷积。输出单元55输出作为卷积的计算结果的子顶部矩阵y(参见图4A至图4C)。

[0169] 接下来，将详细描述存储单元53的功能。存储单元53是将从主存储器11读取的每个数组的元素存储到存储体R#0至R#7中但在前向处理与后向处理之间使用不同的存储方法的功能块。

[0170] 这里，描述前向处理。在前向处理的情况下，存储单元53按照下面的表达式(5)所呈现地对从主存储器11读取的每个数组的元素进行排序，并将每个元素存储到DPE0至DPE7的存储体R#0至R#7。

[0171]
$$d[N_{major}][Cin_{major}][H][W][N_{minor}][Cin_{minor}]$$
$$g[Cout][Cin][H'][W'] \quad \cdots(5)$$
$$y[N_{major}][Cout_{major}][H''][W''][N_{minor}][Cout_{minor}]$$

[0172] 数组y是用于存储通过在子底部矩阵d与权重矩阵g之间进行卷积获得的子顶部矩阵的元素的数组。注意，在该示例中，权重矩阵g是第一矩阵的示例，并且t×t的子底部矩阵d是第二矩阵的示例。

[0173] 另外，(Cin的数目)＝(Cin_{major}的数目)×(Cin_{minor}的数目)。因此，输入通道号Cin可以通过组合(Cin_{major}，Cin_{minor})来标识。因此，在下文中，组合(Cin_{major}，Cin_{minor})等同于输入通道号Cin。例如，Cin_{major}＝0，Cin_{minor}＝0的数组元素对应于Cin＝0，并且Cin_{major}＝0，Cin_{minor}＝1的数组元素对应于Cin＝1。

[0174] 以相同的方式，(N的数目)＝(N_{major}的数目)×(N_{minor}的数目)，并且批号N可以通过组合(N_{major}，N_{minor})来标识。因此，在下文中，组合(N_{major}，N_{minor})等同于批号N。例如，N_{major}＝0，N_{minor}＝0的数组元素对应于N＝0，并且N_{major}＝0，N_{minor}＝1的数组元素对应于N＝1。

[0175] 根据表达式(5)，一个子底部矩阵d可以通过标识输入通道号Cin和批号N来标识。该示例中的输入通道号Cin是如上所述标识子底部矩阵d的第一标识符的示例。类似地，该示例中的批号N是标识子底部矩阵d的第二标识符的示例。

[0176] 另外，在该示例中，假定Cin_{minor}的总数为4，并且N_{minor}的总数为16。此外，假定Cin_{major}的总数为1，并且N_{major}的总数为4。因此，对由如图2所示的256个输入通道号Cin中的4(＝1×4)个输入通道号Cin和64(＝4×16)个批号中的每一个标识的底部矩阵执行卷积。

[0177] 此外，数组d中的元素[H][W]对应于t×t的子底部矩阵d的元素。

[0178] 另一方面，数组g的元素[H'][W']对应于3×3的权重矩阵g的元素。另外，假定数组g的输入通道号Cin的总数为四，该数目等于数组d的输入通道号的数目。此外，假定输出通道号Cout的总数为八。

[0179] 图22示出了当执行前向处理时由存储单元53将每个数组d、g存储于的DPE0至DPE7的寄存器G#0的内容。

[0180] 在DPE0中，多个计算核中的每一个计算存储在存储体R#0至R#7的对应存储体中的矩阵d与g之间的卷积。由于在多个计算核中并行地计算卷积，因此可以提高卷积的计算速度。对于DPE1至DPE7，也是这种情况。

[0181] 以与图13相同的方式通过顺序方法将数组d和g中的数组d存储在DPE0至DPE7的存储体R#0至R#7中。这里，只有具有相同Cin_{major}的数组d被同时存储在存储体R#0至R#7中。然后，在完成数组d的卷积之后，将具有不同Cin_{major}的数组d存储在存储体R#0至R#7中。图22假定具有Cin_{major}＝0的数组d存储在存储体R#0至R#7中的情况。

[0182] 在这种情况下，在本实施方式中，由于如表达式(5)所示Cin_{minor}是数组d的最低级别索引，而N_{minor}是更高一级的索引，因此每个存储体与在相同N_{minor}范围内的Cin_{minor}一一对应。因此，当Cin_{minor}的总数为q(＝4)时，输入通道号(Cin_{major},Cin_{minor})彼此不同并且批号(N_{major},N_{minor})相同的q个子底部矩阵d存储在一个DPE中的q个存储体中。

[0183] 例如，在DPE0中，批号N为(0,0)并且输入通道号Cin为(0,0)、(0,1)、(0,2)、(0,3)的四个子底部矩阵d存储在四(＝q)个存储体R#0至R#3中。

[0184] 因此，与如图13所示相对于存储体R#0至R#7中的每一个改变批号N的情况不同，q个计算核可以并行地计算具有相同批号N的q个子底部矩阵d的卷积。

[0185] 另一方面，存储单元53以与图13的示例相同的方式通过多播方法将权重矩阵g从主存储器11存储在DPE0至DPE7的每个存储体中。

[0186] 这里，存储单元53将与子底部矩阵d具有相同的输入通道号Cin的权重矩阵g存储在DPE0至DPE7中的每一个的每个存储体中。通过将输入通道号Cin彼此相等的矩阵d和g存储在同一存储体中，计算单元54可以如图2所示计算输入通道号Cin彼此相等的矩阵d与g之间的卷积。

[0187] 然而，如参照图15所描述的，当通过多播方法将数组g传送至每个存储体时，一个存储体中的输入通道号Cin与输出通道号Cout之间的规则性降低。因此，在本实施方式中，当利用Winograd算法计算卷积时，计算单元54如下所述对数组g的元素进行排序。

[0188] 图23A至图25示出了当计算单元54利用Winograd算法计算卷积时DPE0的寄存器G#0至G#3的内容。在图23A至图25中，仅示出了寄存器G#0至G#3的存储体R#0，以防止附图变得复杂。

[0189] 在计算卷积之前，如图23A所示，数组d和g的元素被存储在寄存器G#0的存储体R#0中。具有不同上述批号N(＝(N_{major},N_{minor}))的多个数组d作为数组d存储在存储体R#0中。

[0190] 然后，根据等式(2)，将数组d从数组d的两侧乘以矩阵B^T和B，并将因此得到的矩阵B^TdB存储在仍存储数组d的行中。矩阵B^T和B的元素存储在存储体R#0的常数区域cst中。

[0191] 此时，表示权重矩阵的数组g如图15所示具有无序的规则性。

[0192] 因此，在下一步中，如图23B所示，通过将存储在寄存器G#0的存储体R#0中的数组g的每个元素传送至寄存器G#3的存储体R#0来对这些元素进行排序。

[0193] 在排序之后的寄存器中，如图16所示，存储体R#0至R#7与输出通道号Cout一一对应，并且仅Cout＝0的元素被存储在存储体R#0中。

[0194] 然后，如图24所示，根据等式(2)，将数组g从数组g的两侧乘以矩阵G和G^T，并将因此得到的矩阵GgG^T存储在存储体的空闲空间中。矩阵G和G^T的元素存储在存储体R#0的常数区域cst中。

[0195] 然后，如图25所示，对寄存器G#0的存储体R#0中的两个矩阵B^TdB和寄存器G#3的存储体R#0中的一个矩阵GgG^T执行等式(2)的逐元素乘法“⊙”。

[0196] 如参照图2所描述的对具有相同输入通道号Cin的两个矩阵执行卷积。因此，使用寄存器G#3的存储体R#0中四个矩阵GgG^T中的Cin＝0的矩阵和寄存器G#0的存储体R#0中Cin_{minor}＝0的两个矩阵B^TdB来执行逐元素乘法“⊙”。

[0197] 此后，根据等式(2)，将$[GgG^T]\odot[B^TdB]$从$[GgG^T]\odot[B^TdB]$的两侧乘以矩阵A^T和A，以获得子顶部矩阵y。

[0198] 通过以上处理，完成了由计算单元54执行的使用Winograd算法进行的卷积的计算。

[0199] 根据前述的卷积计算，如图23A所示，具有不同批号N(＝(N_{minor},N_{major}))的底部矩阵被存储在寄存器G#0的存储体R#0中。

[0200] 因此，与如图17所述的将具有相同批号N和不同输入通道号Cin的多个子底部矩阵d存储在同一存储体中的示例相比，减少了存储在一个存储体中的子底部矩阵d的数目。作为结果，可以增加底部矩阵d的大小t，并且可以使用Winograd算法高速计算卷积。

[0201] 当发明人针对t＝6的情况进行试算时，在不使用Winograd算法的图3A至图3C的示例中，卷积所需的时间为2304个周期。与之相比，本实施方式的计算时间为1264个周期，表明计算速度提高了1.82(＝2304/1264)倍。

[0202] 为了进一步提高卷积的计算速度，要使t的值尽可能大。然而，当使t太大时，将子底部矩阵d存储在存储体R#0至R#7中的每一个中变得不可能。另一方面，当t的值小时，子底部矩阵d可以可靠地存储在存储体R#0至R#7中的每一个中，但是卷积的计算时间变长。

[0203] 因此，在本实施方式中，如下所述获得t的最优值。首先，如下所述定义参数。

[0204] p:一个DPE中的存储体的数目

[0205] q:一个DPE中存储具有相同N_{minor}的子底部矩阵d的存储体的数目

[0206] R:一个存储体中可以存储的数据集的数目

[0207] 在图22的示例的情况下，这些参数的具体值如下。

[0208] p:8

[0209] q:4

[0210] R:128

[0211] 此外，定义了以下参数。

[0212] Cin':在DPE0中要同时处理的输入通道号Cin的数目

[0213] Cout':在DPE0中要同时处理的输出通道号Cout的数目

[0214] N':在DPE0中要同时处理的批号N的数目

[0215] 将参照图22的示例来描述这些参数。

[0216] Cin'是如上所述在DPE0中要同时处理的输入通道号Cin的数目。输入通道号Cin由组合(Cin_{major},Cin_{minor})标识。然而，由于在图22的示例中仅在DPE0中处理(Cin_{major},Cin_{minor})＝(0,0)、(0,1)、(0,2)和(0,3)的数组g和d，因此Cin'＝4。

[0217] 另一方面，Cout'是如上所述在DPE0中要同时处理的输出通道号Cout的数目。在图22的示例中，由于将Cout的值为0至7的八个权重矩阵g存储在DPE0中，因此Cout'＝8。

[0218] 此外，N'是如上所述在DPE0中要同时处理的批号N的数目。在图22的示例中，由于在DPE0中处理组合(N_{major},N_{minor})＝(0,0)、(0,1)、(1,0)、(1,1)的四个子底部矩阵d，因此N′＝4。接下来，将检查卷积的计算时间。

[0219] 首先，将检查如图23A所示从t×t的子底部矩阵d获得矩阵B^TdB时的计算时间。为了获得矩阵B^TdB，例如，首先计算B^Td，并且然后，从计算结果的右边将计算结果乘以矩阵B。为了计算B^Td，将t×t的子底部矩阵d分解为t个列向量，并计算列向量与矩阵B^T的乘积。

[0220] 因此，在该示例中，用于计算构成t×t的子底部矩阵d的t个列向量之一与矩阵B^T的乘积所需的计算时间由b(t)表示。通过使用函数b(t)，在一个DPE中获得B^TdB所需的计算时间由下面的表达式(6)表示。

[0221] $$(t+t)\cdot b(t)\cdot Cin'\cdot N'\cdot\frac{1}{q}\cdots(6)$$

[0222] 表达式(6)包括“t”的原因是：因为需要将矩阵B^T乘以子底部矩阵d的t个列向量以获得B^Td，所以需要由函数b(t)表示的计算时间的t倍长的计算时间。类似地，需要将矩阵B^Td乘以矩阵B的t个列向量以获得矩阵B^Td与B的乘积。因此，总计算时间变为函数b(t)表示的计算时间的(t+t)倍。因此，表达式(6)包括因子“t+t”。

[0223] 此外，如图22所示，由于Cin'·N'个子底部矩阵d在一个DPE中，因此，每个存储体的子底部矩阵d的数目变为Cin'·N'/q。由于计算核C#0至C#7中的每一个需要针对对应的存储体中Cin'·N'/q个子底部矩阵d中的每一个计算B^TdB，因此表达式(6)包括因子Cin'·N'/q。

[0224] 接下来，将检查如图24所示从3×3的权重矩阵g获得矩阵GgG^T时的计算时间。

[0225] 为了获得矩阵GgG^T，例如，首先计算Gg，并且然后，从计算结果的右边将计算结果乘以矩阵G^T。为了计算Gg，将权重矩阵g分解为三个列向量，并且计算列向量与矩阵G的乘积。

[0226] 因此，在该示例中，用于获得构成3×3的权重矩阵g的三个列向量之一与矩阵G的乘积所需的计算时间由w(t)表示。通过使用函数w(t)，在一个DPE中获得GgG^T所需的计算时间由下面的表达式(7)表示。

[0227] $$(3+t)\cdot w(t)\cdot Cin'\cdot Cout'\cdot\frac{1}{p}\cdots(7)$$

[0228] 表达式(7)包括“3”的原因是：因为需要将矩阵G乘以权重矩阵g的三个列向量以获得矩阵Gg，所以需要由函数w(t)表示的计算时间的三倍长的计算时间。

[0229] 另外，为了获得矩阵Gg和矩阵G^T的乘积，需要将矩阵Gg乘以矩阵G^T的t个列向量。因此，总计算时间变为由函数w(t)表示的计算时间的(t+3)倍长。因此，表达式(7)包括因子“t+3”。

[0230] 另外，如图22所示，由于Cin'·Cout'个权重矩阵g在一个DPE中，因此一个存储体中的权重矩阵g的数目变为Cin'·Cout'/p。由于计算核C#0至C#7中的每一个需要针对对应存储体中的Cin'·Cout'/p个权重矩阵g中的每一个获得GgG^T，因此表达式(7)包括因子Cin'·Cout'/p。

[0231] 接下来，如图25所示，将检查在矩阵B^TdB与GgG^T之间执行逐元素乘法所需的计算时间。

[0232] 如图22所示，存储在一个DPE中的子底部矩阵d的数目为N'·Cin'·Cout'/p。此外，子底部矩阵d的元素的数目为t^2。因此，当在矩阵$B^T dB$与GgG^T之间执行逐元素乘法时的乘法次数由下面的表达式(8)来表示。

[0233] $$t^2 \cdot N' \cdot Cin' \cdot Cout' \cdot \frac{1}{p} \cdots (8)$$

[0234] 表达式(6)至(8)是当从N个批号中选择N'个批号、从Cout个输出通道号中选择Cout'个输出通道号以及从Cin个输入通道号中选择Cin'个输入通道号时的计算时间。因此，为了计算图2中所有底部矩阵与所有权重矩阵之间的卷积，需要将计算执行与由下面的表达式(9)表示的次数一样多的次数。

[0235] $$\frac{HW}{(t-2)^2} \cdot \frac{Cin}{Cin'} \cdot \frac{N}{N'} \cdot \frac{Cout}{p} \cdots (9)$$

[0236] 表达式(9)中的因子HW/(t-2)^{2}表示从H×W的底部矩阵分割t×t的子矩阵的方式的总数目。

[0237] 根据上述表达式(6)至(9)，计算时间不仅取决于t，还取决于q。因此，在本实施方式中，当在一个DPE中计算卷积时的计算时间由第一函数f(t,q)表示。第一函数f(t,q)通过将表达式(6)和(7)之和与表达式(9)相乘来由下面的表达式(10)表示。

[0238] $$\frac{HW}{(t-2)^2} \cdot Cin \cdot \frac{N}{N'} \cdot \frac{Cout}{p} \left\{ 2tb(t) \frac{N'}{q} + (3+t)w(t) \frac{Cout'}{p} + t^2 N' \frac{Cout'}{p} \right\} \cdots (10)$$

[0239] 为了减少卷积所需的计算时间，需要在权重矩阵g和子底部矩阵d中的元素的数目不超过寄存器中可以存储的元素的数目的条件下找到使第一函数f(t,q)的值最小化的t和q的组合。

[0240] 因此，接下来将检查子底部矩阵d和权重矩阵g的元素的数目。首先，将描述子底部矩阵d的元素的数目。

[0241] 一个DPE的一个存储体中的子底部矩阵d的元素的数目E_b由下面的等式(11)表示。

[0242] $$E_b = t^2 \cdot Cin' \cdot \frac{N'}{q} \cdots (11)$$

[0243] 在等式(11)中，t^2表示一个子底部矩阵d的元素的数目。Cin'·N'/q表示要存储在一个存储体中的子底部矩阵d的数目。

[0244] 另一方面，一个DPE的一个存储体中的权重矩阵g的元素的数目E_w由下面的等式(12)表示。

[0245] $$E_w = 3^2 \cdot Cin' \cdot \frac{Cout'}{p} \cdots (12)$$

[0246] 在等式(12)中，3^2是一个权重矩阵g的元素的数目。另外，Cin'·Cout'/p是要存储在一个存储体中的权重矩阵g的数目。

[0247] 基于等式(11)和等式(12)，表示子底部矩阵d和权重矩阵g的元素的总数目的第二函数g(t,q)通过下面的等式(13)来表示。

[0248] $$g(t,q)=E_b+E_w=t^2\cdot Cin'\cdot\frac{N'}{q}+3^2\cdot Cin'\cdot\frac{Cout'}{p}\cdots(13)$$

[0249] 如上所述，当如上所述一个存储体中可以存储的数据集的数目为R时，获得由下面的等式(14)表示的约束条件。

[0250] $$g(t,q)=t^2\cdot Cin'\cdot\frac{N'}{q}+3^2\cdot Cin'\cdot\frac{Cout'}{p}\leq R\cdots(14)$$

[0251] 因此，可以通过从满足等式(14)的约束条件的t和q的组合中找到使由表达式(10)表示的第一函数f(t,q)的值最小化的t和q的组合来提高卷积的计算速度。

[0252] 因此，在本实施方式中，计算单元42计算满足等式(14)的约束条件的t和q的组合之中使由表达式(10)表示的第一函数f(t,q)的值最小化的t和q的组合。

[0253] 在本实施方式中，由于R＝128，因此满足等式(14)的t和q的候选组合不是很多。因此，计算单元42可以通过穷举搜索找到满足等式(14)的t和q的组合，并且可以从找到的组合中识别使表达式(10)的第一函数f(t,q)的值最小化的组合。

[0254] 在表达式(10)中，将b(t)和w(t)视为已知函数。这里，b(t)和w(t)可以如下所述地获得。

[0255] 首先，将描述获得w(t)的方法。如上所述，w(t)是当计算Gg时获得构成3×3的权重矩阵g的三个列向量之一与矩阵G的乘积所需的计算时间。当t＝6时，矩阵G的元素由下面的等式(15)表示。

[0256] $$G=\begin{pmatrix}\frac{1}{4} & 0 & 0\\ -\frac{1}{6} & -\frac{1}{6} & -\frac{1}{6}\\ -\frac{1}{6} & \frac{1}{6} & -\frac{1}{6}\\ \frac{1}{24} & \frac{1}{12} & \frac{1}{6}\\ \frac{1}{24} & -\frac{1}{12} & \frac{1}{6}\\ 0 & 0 & 1\end{pmatrix}\cdots(15)$$

[0257] 该矩阵G可以被变换为下面的等式(16)。

$$G=\begin{pmatrix}\frac{1}{4} & 0 & 0\\ -\frac{1}{6} & -\frac{1}{6} & -\frac{1}{6}\\ -\frac{1}{6} & \frac{1}{6} & -\frac{1}{6}\\ \frac{1}{24} & \frac{1}{12} & \frac{1}{6}\\ \frac{1}{24} & -\frac{1}{12} & \frac{1}{6}\\ 0 & 0 & 1\end{pmatrix}$$

[0258]

$$=\begin{pmatrix}\frac{1}{4} & 0 & 0 & 0 & 0 & 0\\ 0 & -\frac{1}{6} & 0 & 0 & 0 & 0\\ 0 & 0 & -\frac{1}{6} & 0 & 0 & 0\\ 0 & 0 & 0 & \frac{1}{6} & 0 & 0\\ 0 & 0 & 0 & 0 & \frac{1}{6} & 0\\ 0 & 0 & 0 & 0 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 0 & 0\\ 1 & 1 & 1\\ 1 & -1 & 1\\ \frac{1}{4} & \frac{1}{2} & 1\\ \frac{1}{4} & -\frac{1}{2} & 1\\ 0 & 0 & 1\end{pmatrix}\cdots(16)$$

[0259] 等式(16)右侧的两个矩阵被定义为下面的等式(17)和(18)。

[0260]

$$G'=\begin{pmatrix}1 & 0 & 0\\ 1 & 1 & 1\\ 1 & -1 & 1\\ \frac{1}{4} & \frac{1}{2} & 1\\ \frac{1}{4} & -\frac{1}{2} & 1\\ 0 & 0 & 1\end{pmatrix}\cdots(17)$$

[0261]

$$G''=\begin{pmatrix}\frac{1}{4} & 0 & 0 & 0 & 0 & 0\\ 0 & -\frac{1}{6} & 0 & 0 & 0 & 0\\ 0 & 0 & -\frac{1}{6} & 0 & 0 & 0\\ 0 & 0 & 0 & \frac{1}{6} & 0 & 0\\ 0 & 0 & 0 & 0 & \frac{1}{6} & 0\\ 0 & 0 & 0 & 0 & 0 & 1\end{pmatrix}\cdots(18)$$

[0262] 因此,为了计算Gg,首先计算G'g,并且然后,将计算的G'g从G'g的左边乘以G"。因此,将描述计算G'g的方法。

[0263] 在下文中,将3×3的权重矩阵g的一列g'描述为$(g_0,g_1,g_2)^T$。因此,G'g'可以由下面的等式(19)来表示。

[0264] $$G'g' = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & -1 & 1 \\ \frac{1}{4} & \frac{1}{2} & 1 \\ \frac{1}{4} & -\frac{1}{2} & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} g_0 \\ g_1 \\ g_2 \end{pmatrix} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} \cdots (19)$$

[0265] 这里，$(x_0, x_1, x_2, x_3, x_4, x_5)^T$是在其中存储G'g'的每个元素的变量。

[0266] 这里，为了执行等式(19)的计算，准备了六个数组元素a[0]、a[1]、a[2]、a[3]、a[4]和a[5]。然后，将g_0、g_1和g_2分别存储在a[0]、a[1]和a[2]中。然后，准备两个数组元素b[0]和b[1]作为计算的缓冲。

[0267] 在这种情况下，可以通过按图26的顺序插入针对每个数组元素的值来计算等式(19)。

[0268] 图26是按步骤顺序示出等式(19)的计算的示意图。这里，图26中的“//”是指示每一步的含义的注释说明。这同样适用于稍后描述的图27。

[0269] 当根据图26所示的顺序执行计算时，最终(a[0]，a[1]，a[2]，a[3]，a[4]，a[5])＝$(x_0, x_1, x_5, x_2, x_4, x_3)$，并且G'g'的计算结果可以存储在数组元素a[0]、a[1]、a[2]、a[3]、a[4]和a[5]中的每一个中。

[0270] 可以以八步来计算G'g'。因此，w(6)＝8。即使当t的值不同于6时，也可以以与上面描述的方式相同的方式获得w(t)的值。

[0271] 接下来，将描述获得b(t)的方法。如上所述，b(t)是用于获得构成t×t的子底部矩阵d的t个列向量之一与矩阵B^T的乘积B^Td所需的计算时间。当t＝6时，矩阵B^T的元素通过下面的等式(20)表示。

[0272] $$B^T = \begin{pmatrix} 4 & 0 & -5 & 0 & 1 & 0 \\ 0 & -4 & -4 & 1 & 1 & 0 \\ 0 & 4 & -4 & -1 & 1 & 0 \\ 0 & -2 & -1 & 2 & 1 & 0 \\ 0 & 2 & -1 & -2 & 1 & 0 \\ 0 & 4 & 0 & -5 & 0 & 1 \end{pmatrix} \cdots (20)$$

[0273] 此外，在下文中，将6×6子底部矩阵d的一列d'描述为$(d_0, d_1, d_2, d_3, d_4, d_5)^T$。在这种情况下，$B^T$d'可以通过下面的等式(21)表示。

[0274] $$B^T d' = \begin{pmatrix} 4 & 0 & -5 & 0 & 1 & 0 \\ 0 & -4 & -4 & 1 & 1 & 0 \\ 0 & 4 & -4 & -1 & 1 & 0 \\ 0 & -2 & -1 & 2 & 1 & 0 \\ 0 & 2 & -1 & -2 & 1 & 0 \\ 0 & 4 & 0 & -5 & 0 & 1 \end{pmatrix} \begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{pmatrix} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} \cdots (21)$$

[0275] 这里，$(x_0, x_1, x_2, x_3, x_4, x_5)^T$是在其中存储$B^T$d'的元素的变量。

[0276] 这里，为了计算等式(21)，准备了六个数组元素a[0]、a[1]、a[2]、a[3]、a[4]和a[5]，并且d_0、d_1、d_2、d_3、d_4和d_5分别预先存储在数组元素a[0]、a[1]、a[2]、a[3]、a[4]和a[5]中。

[0277] 另外,准备四个数组元素b[0]、b[1]、b[2]和b[3]作为用于计算的缓冲。

[0278] 在这种情况下,可以通过按图27的顺序插入针对每个数组元素的值来计算等式(21)。

[0279] 图27是按步骤顺序示出等式(21)的计算的示意图。当以图27所示的顺序执行计算时,最终(a[0],a[1],a[2],a[3],a[4],a[5])=(x_0,x_1,x_2,x_3,x_4,x_5),并且B^Td'的计算结果可以存储在数组元素a[0]、a[1]、a[2]、a[3]、a[4]和a[5]中的每一个中。

[0280] 因此,可以以15步来计算B^Td'。因此,b(6)=15。即使当t的值不同于6时,也可以以与上面描述的方式相同的方式获得b(t)的值。

[0281] 基于上面描述的事实,根据本实施方式的信息处理设备31执行以下信息处理方法。

[0282] 图28是根据本实施方式的信息处理方法的流程图。首先,在步骤S1中,计算单元42(参见图20)计算t和q的组合。例如,计算单元42计算满足等式(14)的约束条件的t和q的组合之中使表达式(10)的第一函数f(t,q)的值最小化的组合。这使得能够从允许将权重矩阵g和t×t的子底部矩阵d的元素存储在q个存储体中的t和q的组合中获得使计算时间最小化的组合。

[0283] 然后,在步骤S2中,输出单元41(参见图20)输出可由计算机器10(参见图5)执行的程序50。

[0284] 在程序50中使用在步骤S1中计算出的t和q的组合。例如,当计算机器10执行程序50时,选择单元52(参见图21)从底部矩阵选择t×t的子底部矩阵d。

[0285] 然后,存储单元53将t×t的子底部矩阵d和权重矩阵g存储在DPE0的存储体R#0至R#7的q个存储体中。此后,计算单元54根据图23A至图25的过程使用Winograd算法来计算子底部矩阵d与权重矩阵g之间的卷积。

[0286] 通过上面的处理,完成了根据本实施方式的信息处理方法的基本步骤。

[0287] 根据上面描述的实施方式,计算单元42计算在可以将子底部矩阵d和权重矩阵g存储在一个存储体中的等式(14)的约束条件下使表示卷积的计算时间的第一函数f(t,q)最小化的t和q的组合。

[0288] 因此,当子底部矩阵d和权重矩阵g存储在寄存器的存储体中时,可以使用子底部矩阵d和权重矩阵g高速计算卷积。

[0289] 后向处理

[0290] 在图22的示例中,使用Winograd算法计算深度学习的前向处理中的卷积。

[0291] 在下文中,将描述深度学习的后向处理中的Winograd算法。后向处理包括通过顶部矩阵与权重矩阵之间的卷积获得底部矩阵的处理以及通过顶部矩阵与底部矩阵之间的卷积获得权重矩阵的处理。

[0292] 首先,将描述通过顶部矩阵与权重矩阵之间的卷积获得底部矩阵的处理。

[0293] 图29A至图29C是在后向处理中使用Winograd算法计算顶部矩阵与权重矩阵之间的卷积时的示意图。

[0294] 首先,如图29A所示,选择单元52(参见图21)从H行W列的顶部矩阵中选择t×t的子顶部矩阵y。

[0295] 然后,根据下面的等式(22),计算单元54通过权重矩阵g与子顶部矩阵y之间的卷

积获得子底部矩阵d。

[0296] $d=A^T\{(GgG^T)\circledcirc(B^TyB)\}A\cdots(22)$

[0297] 然后，如图29B所示，将从顶部矩阵分割子顶部矩阵y的位置从图29A的情况的位置移动两列，分割的子顶部矩阵y经历与上面描述的计算相同的计算。所得到的子底部矩阵d在底部矩阵中形成与在图29A中获得的子底部矩阵d相邻的块。

[0298] 如上所述，通过将从矩阵分割子顶部矩阵y的位置重复地移动两列和两行，获得由子底部矩阵d形成的底部矩阵，如图29C所示。

[0299] 通过以上步骤，完成了后向处理中顶部矩阵与权重矩阵之间的卷积的计算。在该示例中，权重矩阵g是第一矩阵的示例，并且t×t的子顶部矩阵y是第二矩阵的示例。

[0300] 接下来，将详细描述当以上述方式执行后向处理时存储单元53的功能。

[0301] 存储单元53对如由下面的表达式(23)所表示的每个数组的元素进行排序，并将所述元素存储在DPE0至DPE7的存储体R#0至R#7中。

[0302]
$$d[N_{major}][Cin_{major}][H][W][N_{minor}][Cin_{minor}]$$
$$g[Cout][Cin][H'][W'] \qquad \cdots(23)$$
$$y[N_{major}][Cout_{major}][H''][W''][N_{minor}][Cout_{minor}]$$

[0303] 这里，当N为批号时，(N的数目)＝(N_{major}的数目)×(N_{minor}的数目)，(Cout的数目)＝($Cout_{major}$的数目)×($Cout_{minor}$的数目)。在这种情况下，与表达式(5)一样，批号N由组合(N_{major},N_{minor})标识。在后向处理中，批号N是用于标识子顶部矩阵y的第二标识符的示例。

[0304] 输出通道号Cout也由组合($Cout_{major}$,$Cout_{minor}$)标识。例如，$Cout_{major}$＝0，$Cout_{minor}$＝0的数组元素对应于Cout＝0，以及$Cout_{major}$＝0，$Cout_{minor}$＝1的数组元素对应于Cout＝1。此外，在后向处理中，输出通道号Cout是用于标识子顶部矩阵y的第一标识符。

[0305] 此外，在该示例中，如图2中那样，假定批号N的总数目为64，并且输出通道号Cout的总数目为384。还假定N_{major}的总数目为16，以及$Cout_{minor}$的总数目为4。

[0306] 数组y中的元素[H”][W”]对应于t×t的子顶部矩阵y的元素。

[0307] 图30示出了由存储单元53将数组y和g存储于的DPE0至DPE7的寄存器G#0的内容。

[0308] 数组y由存储单元53通过顺序方法存储在DPE0至DPE7的存储体R#0至R#7中。

[0309] 在这种情况下，在本实施方式中，如表达式(23)所示，$Cout_{minor}$是数组y的最低级别的索引，而N_{minor}是下一较高级别的索引。因此，每个存储体与相同的N_{minor}范围内的$Cout_{minor}$一一对应。因此，当$Cout_{minor}$的总数目为q(＝4)时，具有不同输出通道号($Cout_{major}$,$Cout_{minor}$)和相同批号(N_{major},N_{minor})的q个子顶部矩阵y存储在一个DPE的q个存储体中。

[0310] 例如，在DPE0中，批号N为(0,0)且输出通道号Cout为(0,0)、(0,1)、(0,2)、(0,3)的四个子顶部矩阵y分别存储在四个存储体R#0至R#3中。

[0311] 因此，与如图13所示相对于每个存储体R#0至R#7改变批号N的示例不同，可以在q个计算核中并行地计算具有相同批号N的q个子顶部矩阵y的卷积。

[0312] 另一方面，如图22的示例中那样，由存储单元53通过多播方法将权重矩阵g从主存储器11传送至DPE0至DPE7。

[0313] 如参照图15所描述的，在多播方法中，输入通道号Cin的值与输出通道号Cout的值之间没有规则性。因此，同样在该示例中，计算单元54如图23A至图25中那样对数组g进行排序。

[0314] 接下来，将检查该后向处理中的卷积的计算时间。

[0315] 在一个DPE中获得由等式(22)表示的$B^T yB$所需的计算时间可以通过用Cout'代替表达式(6)中的Cin'而由下面的表达式(24)表示。

[0316] $$(t+t)\cdot b(t)\cdot Cout'\cdot N'\cdot\frac{1}{q}\cdots(24)$$

[0317] 另外，由于与表达式(7)的原因相同的原因，在一个DPE中获得由等式(22)表示的GgG^T所需的计算时间可以由表达式(25)表示。

[0318] $$(3+t)\cdot w(t)\cdot Cin'\cdot Cout'\cdot\frac{1}{p}\cdots(25)$$

[0319] 此外，在执行等式(22)中矩阵$B^T yB$与GgG^T之间的逐元素乘法时的乘法次数由下面与表达式(8)一样的表达式(26)来表示。

[0320] $$t^2\cdot N'\cdot Cin'\cdot Cout'\cdot\frac{1}{p}\cdots(26)$$

[0321] 为了计算所有顶部矩阵与所有权重矩阵之间的卷积，需要将计算执行与下面的表达式(27)表示的次数一样多的次数，其中将表达式(9)中的p替换为Cout'。

[0322] $$\frac{HW}{(t-2)^2}\cdot\frac{Cin}{Cin'}\cdot\frac{N}{N'}\cdot\frac{Cout}{cout'}\cdots(27)$$

[0323] 表示在一个DPE中计算卷积时的计算时间的第一函数f(t,q)可以通过将表达式(24)至(26)的总和乘以表达式(27)而由下面的等式(28)表示。

[0324] $$\mathrm{f(t,q)}=\frac{HW}{(t-2)^2}\cdot\frac{N}{N'}\cdot Cout\cdot\frac{1}{p}\left\{\frac{2tb(t)N'Cin'p}{q}+(3+t)w(t)+t^2N'\right\}\cdots(28)$$

[0325] 接下来，将检查子顶部矩阵y和权重矩阵g的元素的数目不超过寄存器中可以存储的元素的数目的条件。首先，将描述子顶部矩阵y的元素的数目。

[0326] 一个DPE的一个存储体中的子顶部矩阵y的元素的数目E_y可以通过将等式(11)中的Cin'替换为Cout'而由下面的等式(29)表示。

[0327] $$E_y=t^2\cdot Cout'\cdot\frac{N'}{q}\cdots(29)$$

[0328] 另一方面，与等式(12)一样，可以通过下面的等式(30)表示一个DPE的一个存储体中的权重矩阵g的元素的数目E_w。

[0329] $$E_w=3^2\cdot Cin'\cdot\frac{Cout'}{p}\cdots(30)$$

[0330] 基于等式(29)和等式(30)，表示子顶部矩阵y和权重矩阵g的元素的总数目的第二函数g(t,q)可以通过下面的等式(31)来表示。

[0331] $$g(t,q)=E_y+E_w=t^2\cdot Cout'\cdot\frac{N'}{q}+3^2\cdot Cin'\cdot\frac{Cout'}{p}\cdots(31)$$

[0332] 因此，当存储在一个存储体中的数据集的数目为R时，获得由下面的等式(32)表示的约束条件。

[0333] $$g(t,q) = E_y + E_w = t^2 \cdot Cout' \cdot \frac{N'}{q} + 3^2 \cdot Cin' \cdot \frac{Cout'}{p} \leq R \cdots (32)$$

[0334] 因此,可以通过从满足等式(32)的约束条件的t和q的组合中找到使等式(28)的第一函数f(t,q)的值最小化的t和q的组合来提高卷积的计算速度。

[0335] 因此,当执行用于通过顶部矩阵与权重矩阵之间的卷积获得子底部矩阵d的后向处理时,计算单元42识别满足等式(32)的约束条件的t和q的组合。然后,计算单元42计算识别出的组合之中使等式(28)的第一函数f(t,q)的值最小化的t和q的组合以提高卷积的计算速度。

[0336] 接下来,将描述用于通过顶部矩阵与底部矩阵之间的卷积获得权重矩阵的后向处理。

[0337] 图31A至图32C是在后向处理中使用Winograd算法计算顶部矩阵与底部矩阵之间的卷积时的示意图。

[0338] 首先,如图31A所示,选择单元52从H×W的顶部矩阵中选择t'×t'的子顶部矩阵y。

[0339] 然后,如图31B所示,选择单元52从H'×W'的底部矩阵中选择(t'-2)×(t'-2)的子底部矩阵d。

[0340] 然后,如图32A所示,计算单元54从子顶部矩阵y中选择(t'-2)×(t'-2)的矩阵y'。然后,计算单元54根据下面的等式(33)获得权重矩阵g的11分量。

[0341] $g_{11} = A^T\{(Gy'G^T) \circledcirc (B^TdB)\}A \cdots (33)$

[0342] 然后,如图32B所示,将从子顶部矩阵y选择矩阵y'的位置从图32A的情况的位置移动一列,并且计算单元54对所选择的矩阵y'执行与上面描述的计算相同的计算以获得权重矩阵g的12分量。

[0343] 如上所述,通过将从子顶部矩阵y中分割矩阵y'的位置在列方向和行方向上重复移动,获得如图32C所示的3×3的权重矩阵g的每个元素。

[0344] 通过上面的处理,完成了后向处理中的顶部矩阵与底部矩阵之间的卷积的计算。在该示例中,(t'-2)×(t'-2)的子底部矩阵d是第一矩阵的示例,以及t'×t'的子顶部矩阵y是第二矩阵的示例。

[0345] 接下来,将详细描述当执行后向处理时存储单元53的功能。

[0346] 存储单元53对由下面的表达式(34)表示的每个数组的元素进行排序,并且然后将每个元素存储至DPE0至DPE7的存储体R#0至R#7。

[0347] $$\begin{matrix} d[N_{major}][Cin_{major}][H][W][Cin_{minor}][N_{minor}] \\ g[Cin_{minor}][Cout_{major}][H'][W'][Cin_{minor}][Cout_{minor}] \\ y[N_{major}][Cout_{major}][H''][W''][N_{minor}][Cout_{minor}] \end{matrix} \cdots (34)$$

[0348] 同样在该示例中,子底部矩阵d通过批号N(=(N_{major},N_{minor}))和输入通道号Cin(=(Cin_{major},Cin_{minor}))的组合来标识。批号N(=(N_{major},N_{minor}))是第一标识符的示例,以及输入通道号Cin(=(Cin_{major},Cin_{minor}))是第二标识符的示例。

[0349] 图33示出了由存储单元53将数组y和d存储于的DPE0至DPE7的寄存器G#0的内容。

[0350] 数组d由存储单元53通过顺序方法存储在DPE0至DPE7的存储体R#0至R#7中。

[0351] 在这种情况下,在本实施方式中,由于如表达式(34)所示N_{minor}是数组d的最低级别

的索引，以及Cin_{minor}是下一较高级别的索引。因此，每个存储体与同一Cin_{minor}范围内的N_{minor}一一对应。因此，当N_{minor}的总数目为q(＝4)时，将具有不同批号(N_{major},N_{minor})和相同输入通道号(Cin_{major},Cin_{minor})的q个子底部矩阵d存储在一个DPE的q个存储体中。

[0352] 例如，输入通道号Cin为(0,0)以及批号N为(0,0)、(0、1)、(0、2)、(0、3)的四个子底部矩阵d分别存储在DPE0的四个存储体R#0至R#3中。

[0353] 因此，与如图13所示相对于存储体R#0至R#7中的每一个改变批号N的示例不同，可以通过q个计算核并行地计算具有相同输入信道号Cin的q个子底部矩阵d的卷积。

[0354] 由存储单元53通过多播方法将子顶部矩阵y从主存储器11传送至DPE0至DPE7。

[0355] 与图30的示例不同，在该示例中，如表达式(34)中所示，$Cout_{minor}$是数组y的最低级别的索引，以及N_{minor}是下一较高级别的索引。另外，假定$Cout_{minor}$的总数目为4，以及N_{minor}的总数目为4。

[0356] 因此，例如在DPE0中，以$Cout_{minor}$值的升序将N_{major}＝0且N_{minor}＝0的数组y的元素中的元素存储在存储体R#0至R#3中。然后，以$Cout_{minor}$值的升序将数组中的N_{major}＝0且N_{minor}＝1的元素存储在存储体R#4至R#7中。

[0357] 数组y中的N_{major}＝1的元素也以$Cout_{minor}$值的升序存储在存储体R#0至R#3中，以及N_{minor}大1的元素存储在存储体R#4至R#7中。

[0358] 因此，具有相同$Cout_{minor}$值的数组y的元素存储在一个存储体中。因此，不必对数组y的元素进行排序以使存储体中的$Cout_{minor}$值相同。

[0359] 接下来，将检查该后向处理中的卷积的计算时间。

[0360] 在一个DPE中获得由等式(33)表示的$Gy'G^T$所需的计算时间将通过将表达式(24)中的t替换为t'而由下面的表达式(35)表示。

[0361] $$(t'+t')\cdot b(t')\cdot Cout'\cdot N'\cdot\frac{1}{q}\cdots(35)$$

[0362] 此外，用于在一个DPE中获得由等式(33)表示的B^TdB的计算时间将通过分别将表达式(25)中的3、t和cout'替换为t'-2、t'和N'而由下面的表达式(36)表示。

[0363] $$\left((t'-2)+t'\right)\cdot w(t')\cdot Cin'\cdot N'\cdot\frac{1}{p}\cdots(36)$$

[0364] 此外，在等式(33)中，当执行矩阵$Gy'G^T$与矩阵B^TdB之间的逐元素乘法时的乘法次数由下面与表达式(8)一样的等式(37)表示。

[0365] $$t^2\cdot N'\cdot Cin'\cdot Cout'\cdot\frac{1}{p}\cdots(37)$$

[0366] 为了计算所有顶部矩阵与所有权重矩阵之间的卷积，需要将计算执行与由下面与表达式(27)一样的表达式(38)表示的次数一样多的次数。

[0367] $$\frac{HW}{(t-2)^2}\cdot\frac{Cin}{Cin'}\cdot\frac{N}{N'}\cdot\frac{Cout}{cout'}\cdots(38)$$

[0368] 表示在一个DPE中计算卷积时的计算时间的第一函数f(t,q)可以通过将表达式(35)至(37)的总和乘以表达式(38)而由下面的等式(39)来表示。

[0369] $$f(t,q)=\frac{HW}{(t'-2)^2}\cdot\frac{Cin}{Cin'}\cdot N \cdot\frac{Cout}{Cout'}\left\{\frac{2t'b(t')Cout'}{q}+2(t'-1)w(t')\frac{Cin'}{p}+t'^2Cin'\frac{Cout'}{p}\right\}\cdots(39)$$

[0370] 接下来,将检查子底部矩阵d和子顶部矩阵y的元素的数目不超过寄存器中可以存储的元素的数目的条件。

[0371] 首先,将描述子顶部矩阵y的元素的数目。一个DPE的一个存储体中的子顶部矩阵y的元素的数目E_y可以由下面的等式(40)表示。

[0372] $$E_y=t'^2\cdot N'\cdot\frac{Cin'}{p}\cdots(40)$$

[0373] 在等式(40)中t'^{2}是一个子顶部矩阵y的元素的数目。另外,N'·Cin'/p是要存储在一个存储体中的子顶部矩阵y的数目。

[0374] 另一方面,一个DPE的一个存储体中的子底部矩阵d的元素的数目E_d可以通过下面的等式(41)来表示。

[0375] $$E_d=(t'-2)^2\cdot N'\cdot\frac{Cout'}{p}\cdots(41)$$

[0376] 在等式(41)中,(t'-2)^{2}是一个子底部矩阵d的元素的数目。另外,N'·Cout'/p是要存储在一个存储体中的子底部矩阵d的数目。

[0377] 基于等式(29)和等式(30),表示子顶部矩阵y和权重矩阵g的元素的总数目的第二函数g(t,q)可以通过下面的等式(42)表示。

[0378] $$g(t,q)=E_y+E_d=t'^2\cdot N'\cdot\frac{Cin'}{p}+(t'-2)^2\cdot N'\cdot\frac{Cout'}{p}\cdots(42)$$

[0379] 因此,当可以存储在一个存储体中的数据集的数目为R时,获得由下面的等式(43)表示的约束条件。

[0380] $$g(t,q)=E_y+E_d=t'^2\cdot N'\cdot\frac{Cin'}{p}+(t'-2)^2\cdot N'\cdot\frac{Cout'}{p}\leq R\cdots(43)$$

[0381] 因此,可以通过从满足等式(43)的约束条件的t和q的组合中找到使等式(39)的第一函数f(t,q)的值最小化的t和q的组合来提高卷积的计算速度。

[0382] 相应地,当执行如本示例中所述的通过底部矩阵与顶部矩阵之间的卷积获得权重矩阵的后向处理时,计算单元42识别满足等式(43)的约束条件的t和q的组合。然后,计算单元42计算识别出的组合中使等式(39)的第一函数f(t,q)的值最小化的t和q的组合以提高卷积的计算速度。1×1卷积

[0383] 在深度学习中,可以执行1×1卷积。例如,ResNet-50或ResNet 101使用1×1卷积。因此,将描述本实施方式中的1×1卷积。

[0384] 尽管对要经受1×1卷积的矩阵没有特别限制,但是在下文中,将描述子底部矩阵d与权重矩阵g之间的卷积。

[0385] 当在矩阵d与g之间执行1×1卷积时,存储单元53将每个矩阵的元素存储在由表达

式(44)表示的对应数组中,并将元素存储在DPE0至DPE7的存储体R#0至R#7中。

[0386] $$\begin{matrix} d[N_{major}][Cin_{major}][H][W][N_{minor}][Cin_{minor}] \\ g[1][1][Cin][Cout] \end{matrix} \quad \cdots(44)$$

[0387] 表达式(44)中的每个数组d、g的元素的顺序与表达式(5)的顺序相同。例如,在数组d中,Cin_{minor}是最低级别的索引,以及N_{minor}是下一较高级别的索引。

[0388] 图34示出了当执行1×1卷积时由存储单元53将数组d和g存储于的DPE0的寄存器G#0的内容。

[0389] 在表达式(5)的情况下,如图22所示,通过顺序方法将数组d存储在DPE0至DPE7中,然而,在该示例中,通过多播方法将数组d存储在DPE0至DPE7中。

[0390] 因此,例如,N_{major}=0和N_{minor}=0的元素以Cin_{minor}=0、1、2、3的顺序存储在存储体R#0、R#1、R#2和R#3中。当存储了N_{major}=0和N_{minor}=0的所有元素时,然后将N_{major}=0和N_{minor}=1的元素以Cin_{minor}=0、1、2、3的顺序存储在存储体R#4、R#5、R#6和R#7中。因此,存储体R#0至R#7中的每一个的第一行被填充,并且因此,N_{minor}=2或大于2的元素被存储在下一行中。

[0391] 在完成对N_{major}=0的元素的卷积之后,将N_{major}=1的数组d的元素扩展到DPE0。这同样适用于N_{major}为2或大于2的数组d的元素。

[0392] 另外,对于数组g,通过多播方法将数组g存储在存储体R#0中。

[0393] 没有适用于1×1卷积的Winograd算法。因此,在该示例中,计算单元54通过使用存储在存储体R#0至R#7中的元素根据图3A至图3C中所示的过程来执行卷积。

[0394] 批量归一化

[0395] 在深度学习中,可以通过执行批量归一化来提高性能。批量归一化是当像素数据的值在多个图像之间差异很大时使每个图像的像素数据的平均值为0并且使像素数据的分布为1的归一化方法。在下文中将描述该方法。

[0396] 当执行批量归一化时,存储单元53对如下面的表达式(45)所示的每个数组d、y的元素进行排序,并通过多播方法将元素存储在DPE0至DPE7的存储体R#0至R#7中。

[0397] $$\begin{matrix} d[N_{major}][Cin_{major}][H][W][N_{minor}][Cin_{minor}] \\ y[N_{major}][Cin_{major}][H][W][N_{minor}][Cin_{minor}] \end{matrix} \quad \cdots(45)$$

[0398] 批量归一化适用于底部矩阵和顶部矩阵二者。在下文中,将描述对作为底部矩阵的一部分的子底部矩阵d执行批量归一化的情况。

[0399] 图35示出了当执行批量归一化时由存储单元53将子底部矩阵d存储于的DPE0的寄存器G#0的内容。

[0400] 在该示例中,如图34中那样,存储单元53通过多播方法将子底部矩阵d存储在存储体R#0中。如表达式(45)所示,Cin_{minor}是子底部矩阵d的最低级别的索引。因此,当着眼于存储体R#0至R#7之一时,具有相同Cin_{minor}值的元素被存储在一个存储体中。例如,仅Cin_{minor}=0的元素存储在存储体R#0中。

[0401] 另外,根据表达式(45),在子底部矩阵d中,N_{minor}是级别比Cin_{minor}高的索引。因此,当着眼于存储体R#0至R#7之一时,具有不同批号(N_{major},N_{minor})的元素被存储在一个存储体中。例如,具有(N_{major},N_{minor})=(0,0)、(0,2)、…(0,14)、(1,0)、(1,2)、…(1,14)、…(3,0)、(3,2)、…(3,14)的元素存储在存储体R#0中。

[0402] 如上所述，具有相同Cin_{minor}和不同批号(N_{major},N_{minor})的元素存储在一个存储体中。因此，计算核C#0至C#7中的每一个可以通过仅使用对应的一个存储体来计算具有相同Cin_{minor}和不同批号(N_{major},N_{minor})的多个元素的平均值以及这些元素的离差(dispersion)。

[0403] 由计算单元54如下所述地执行计算。图36A和图36B示出了DPE0的寄存器G#0的内容，并且是用于描述当执行批量归一化时由计算单元54执行的计算的图。

[0404] 首先，如图36A所示，计算核C#0将存储体R#0中的子底部矩阵d的元素的值相加，并将获得的值x_0存储在存储体R#0的行L_{sum_1}中。同样在其他存储体R#1至R#7中，计算核C#1至C#7中的每一个将对应存储体中的子底部矩阵d的元素的值相加，并且然后将所获得的值x_1至x_7分别存储在存储体R#1至R#7的行L_{sum_1}。

[0405] 这里，如图35所示，仅N_{minor}为偶数的元素被存储在存储体R#0中。因此，值x_0变为不是所有批号(N_{major},N_{minor})的元素之和，而是N_{minor}为偶数的元素的值之和。

[0406] 因此，计算单元54将值x_0至x_7之中与相同Cin_{minor}对应的值相加。例如，值x_0和值x_4二者都对应于$Cin_{minor}=0$。因此，计算单元54将这两个值相加并将结果写入值x_0。所获得的值x_0等于跨全部批号(N_{major},N_{minor})将$Cin_{minor}=0$的元素相加获得的值。类似地，计算单元54执行以下计算。

[0407] $x_1=x_1+x_5$

[0408] $x_2=x_2+x_6$

[0409] $x_3=x_3+x_7$

[0410] 然后，计算核C#0通过将存储在存储体R#0中的值x_0除以批号来计算平均值m_0，并将所获得的平均值m_0存储在存储体R#0的行L_{mean}中。同样在存储体R#1至R#3中，计算核C#1至C#3分别计算值x_1至x_3的平均值m_1至m_3，并将这些值分别存储在存储体R#1至R#3的行L_{mean}中。

[0411] 通过上面的处理，分别针对存储体R#0至R#3获得了子底部矩阵d的元素的平均值m_0至m_3。接下来，将描述计算离差的方法。

[0412] 首先，如图36B所示，计算核C#0对存储体R#0中的子底部矩阵d的每个元素的值求平方，并且将通过对所获得的值求和获得的值y_0存储在存储体R#0的行L_{sum_2}中。同样在其他存储体R#1至R#7中，计算核C#1至C#7中的每一个对对应存储体中的每个元素的值求平方，对所获得的值求和，并将所获得的值y_1至y_7存储至存储体R#1至R#7中的对应存储体的行L_{sum_2}。

[0413] 如在图36A的示例中那样，值y_0不是跨所有批号(N_{major},N_{minor})的元素的值的平方的和，而是仅通过对——作为N_{minor}为偶数的元素的值的平方的——值求和获得的值。因此，计算单元54执行以下计算，并且将跨所有批号(N_{major},N_{minor})的子底部矩阵d的元素的平方的和写在值y_0至y_3中。

[0414] $y_0=y_0+y_4$

[0415] $y_1=y_1+y_5$

[0416] $y_2=y_2+y_6$

[0417] $y_3=y_3+y_7$

[0418] 然后，计算核C#0通过将存储在存储体R#0中的值y_0除以批号来计算平均值a_0，并将计算出的平均值a_0存储在存储体R#0的行L_{mean_2}中。同样在存储体R#1至R#3中，计算核C#1至C#3计算值y_1至y_3的平均值a_1至a_3，并将这些值分别存储在存储体R#1至R#3的行L_{mean_2}中。

[0419] 通过上面的处理，针对存储体R#0至R#3获得了子底部矩阵d的元素的平方的平均值a_0至a_3。

[0420] 然后，计算单元54计算$v_0=a_0-m_0^2$以计算存储体R#0的子底部矩阵d的元素的离差v_0，并且然后将离差v_0存储在存储体R#0的行L_{var}中。以相同的方式，计算单元54执行以下计算以计算存储体R#1至R#3的元素的离差v_1至v_3，并将离差v_1至v_3分别存储在存储体R#1至R#3的行L_{var}中。

[0421] $v_1=a_1-m_1^2$

[0422] $v_2=a_2-m_2^2$

[0423] $v_3=a_3-m_3^2$

[0424] 此后，计算单元54通过如下面的等式(46)所示将子底部矩阵d的每个元素的值(d[N_{major}][Cin_{major}][H][W][N_{minor}][i])与平均值m_i之间的差除以离差v_i来对Cin_{minor}=i(i=0、1、2、3)执行批量归一化。

[0425]
$$d[N_{major}][Cin_{major}][H][W][N_{minor}][i] = \frac{1}{v_i}\left(d[N_{major}][Cin_{major}][H][W][N_{minor}][i] - m_i\right) \cdots (46)$$

[0426] 通过上面的处理，完成了批量归一化。

[0427] 通过如上所述执行批量归一化，期望深度学习的学习性能的改善。

[0428] 尽管已经详细说明了本发明的例示性实施方式，但是本发明不限于上面提到的实施方式，并且在不脱离本发明的范围的情况下可以作出其他实施方式、变型和修改。

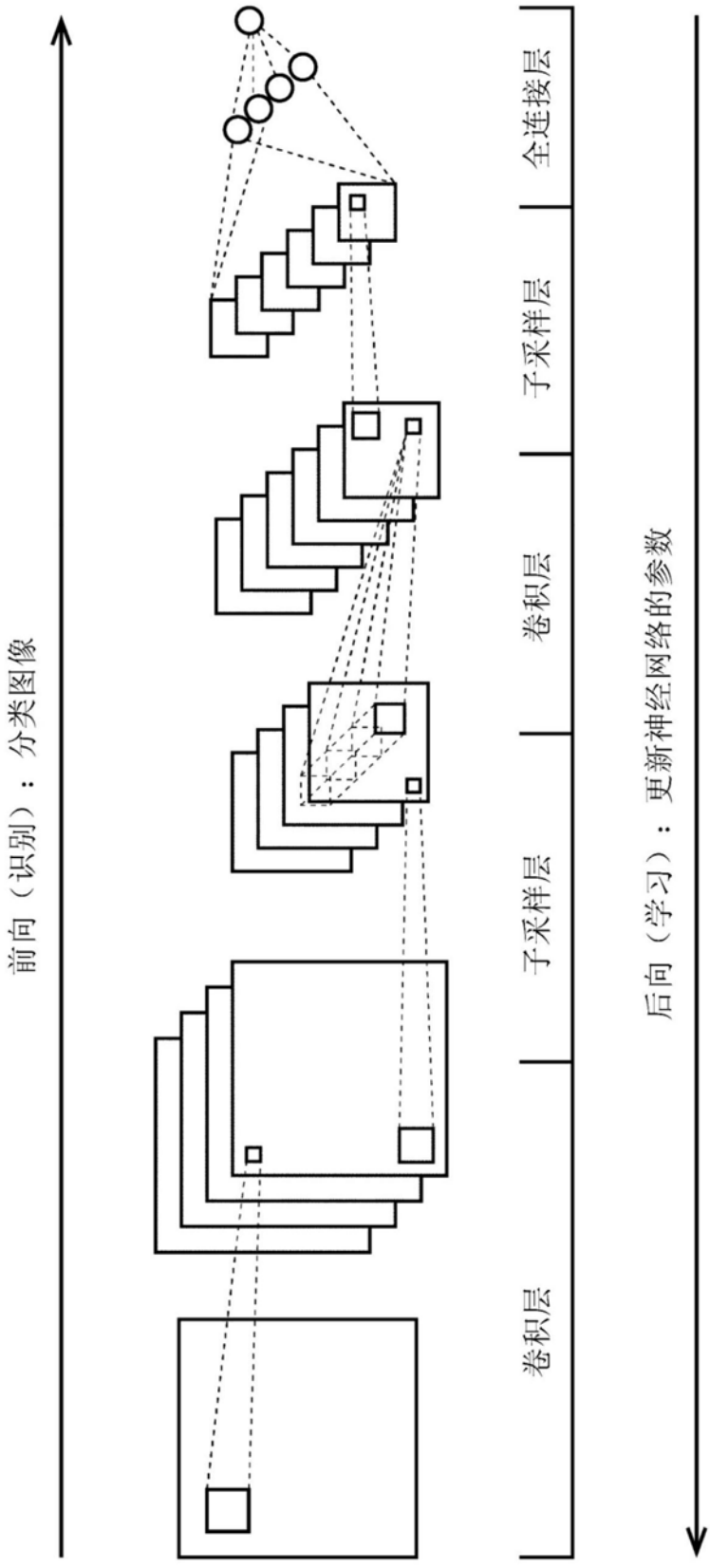


图1

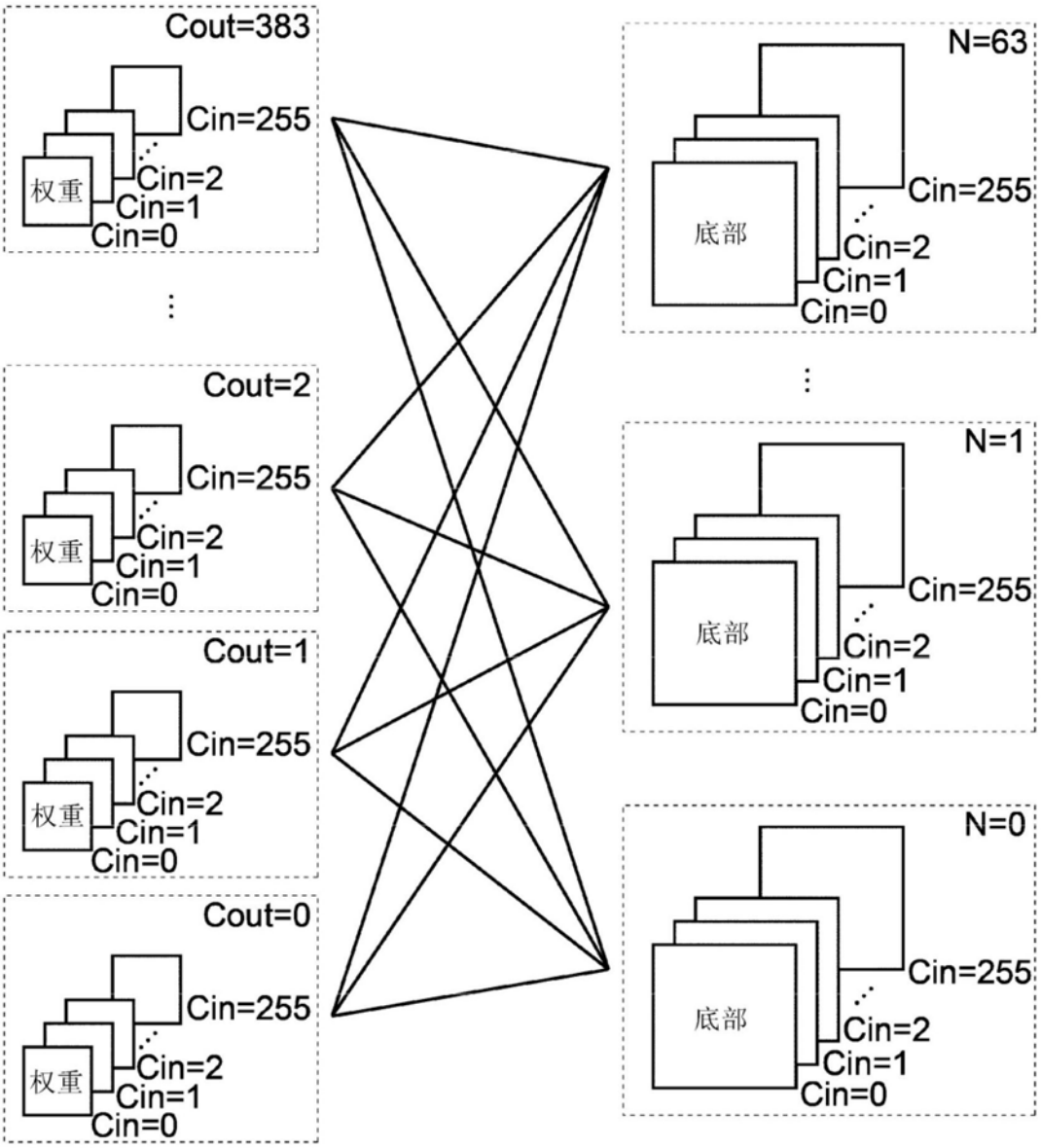

Cout=383
Cin=255
Cin=2
Cin=1
Cin=0
权重
Cout=2
Cout=1
Cout=0
N=63
N=1
N=0
底部

图2

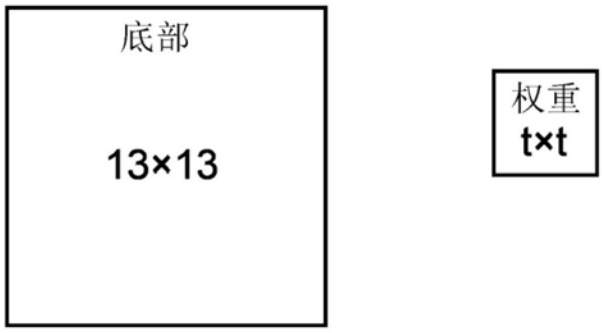

底部
13×13
权重
t×t

图3A

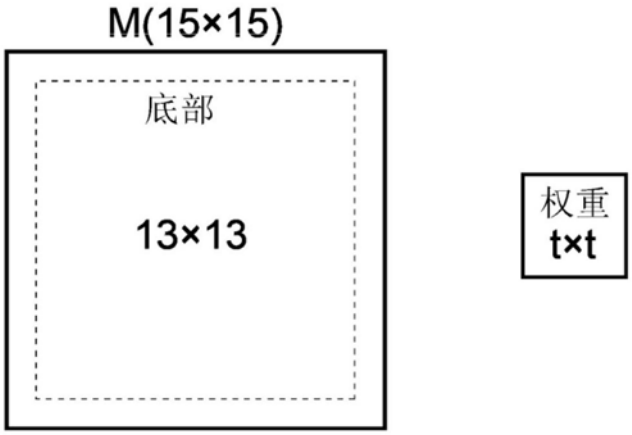


图3B

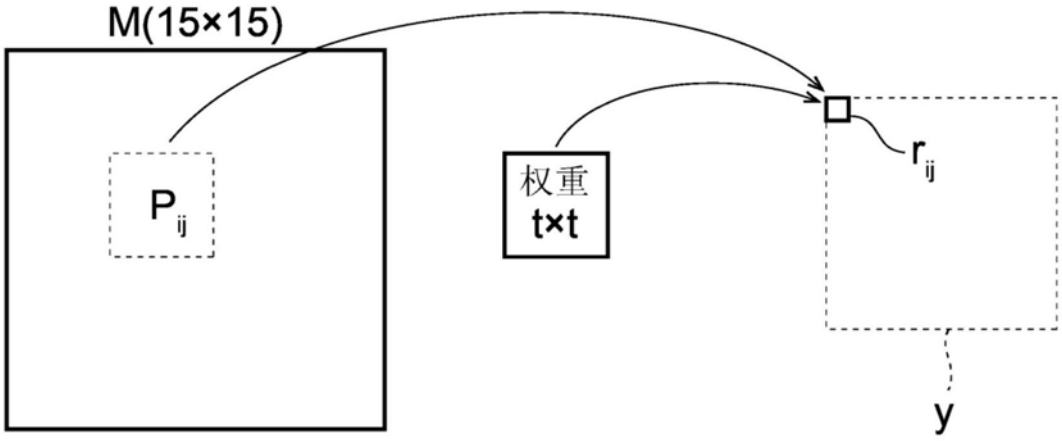


图3C

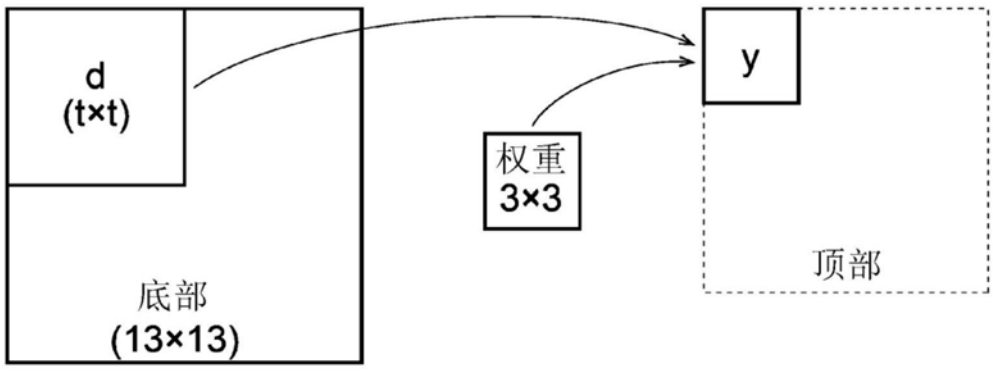


图4A

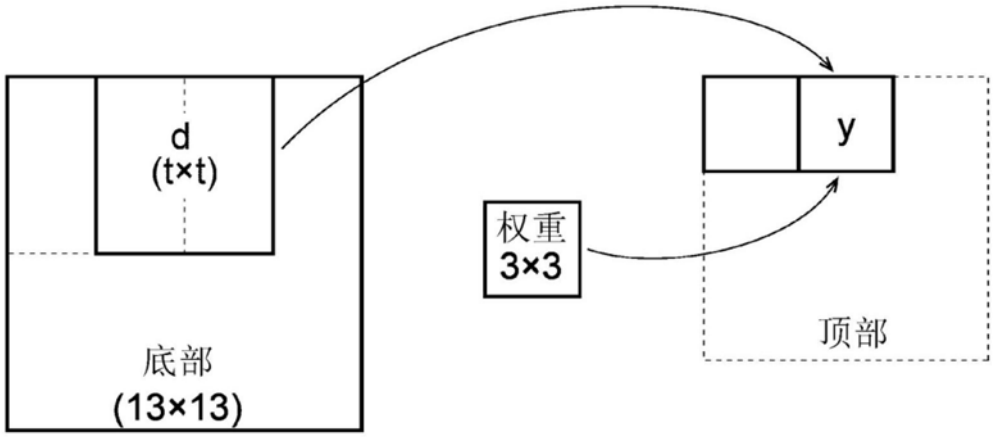


图4B

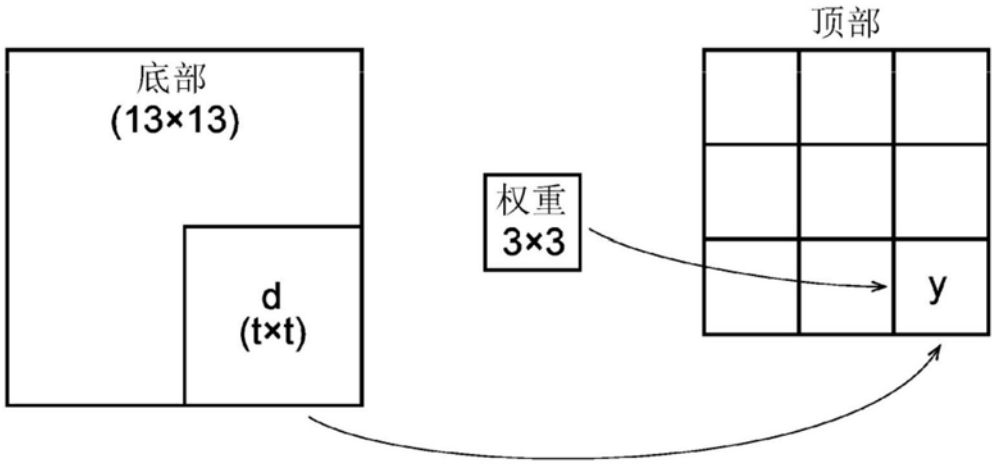


图4C

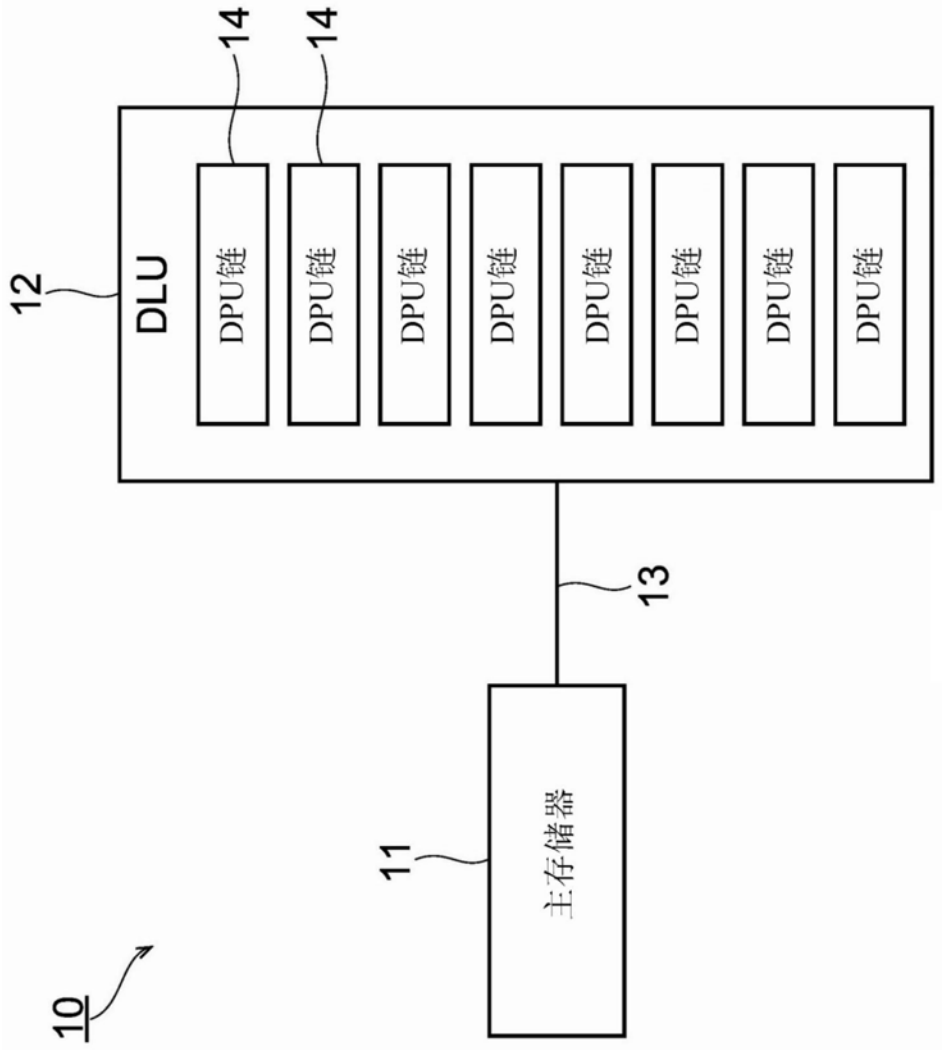

14
14
DPU链
DPU链
DPU链
DPU链
DPU链
DPU链
DPU链
DPU链
12
DLU
13
11
主存储器
10

图5

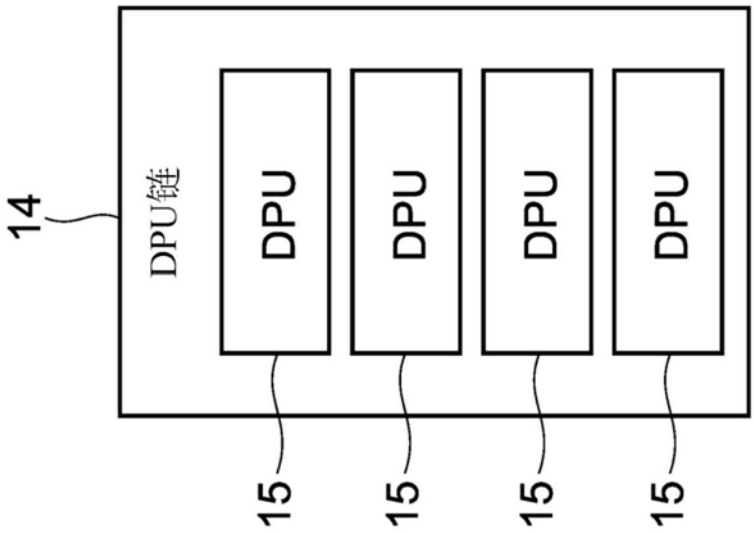

14
DPU链
DPU
DPU
DPU
DPU
15
15
15
15

图6A

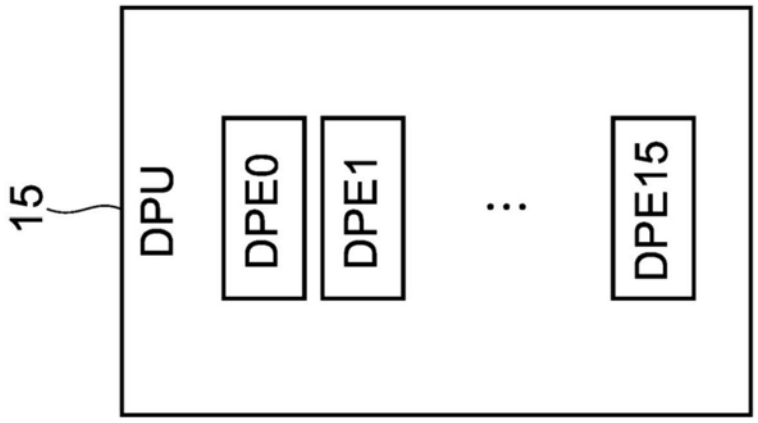


图6B

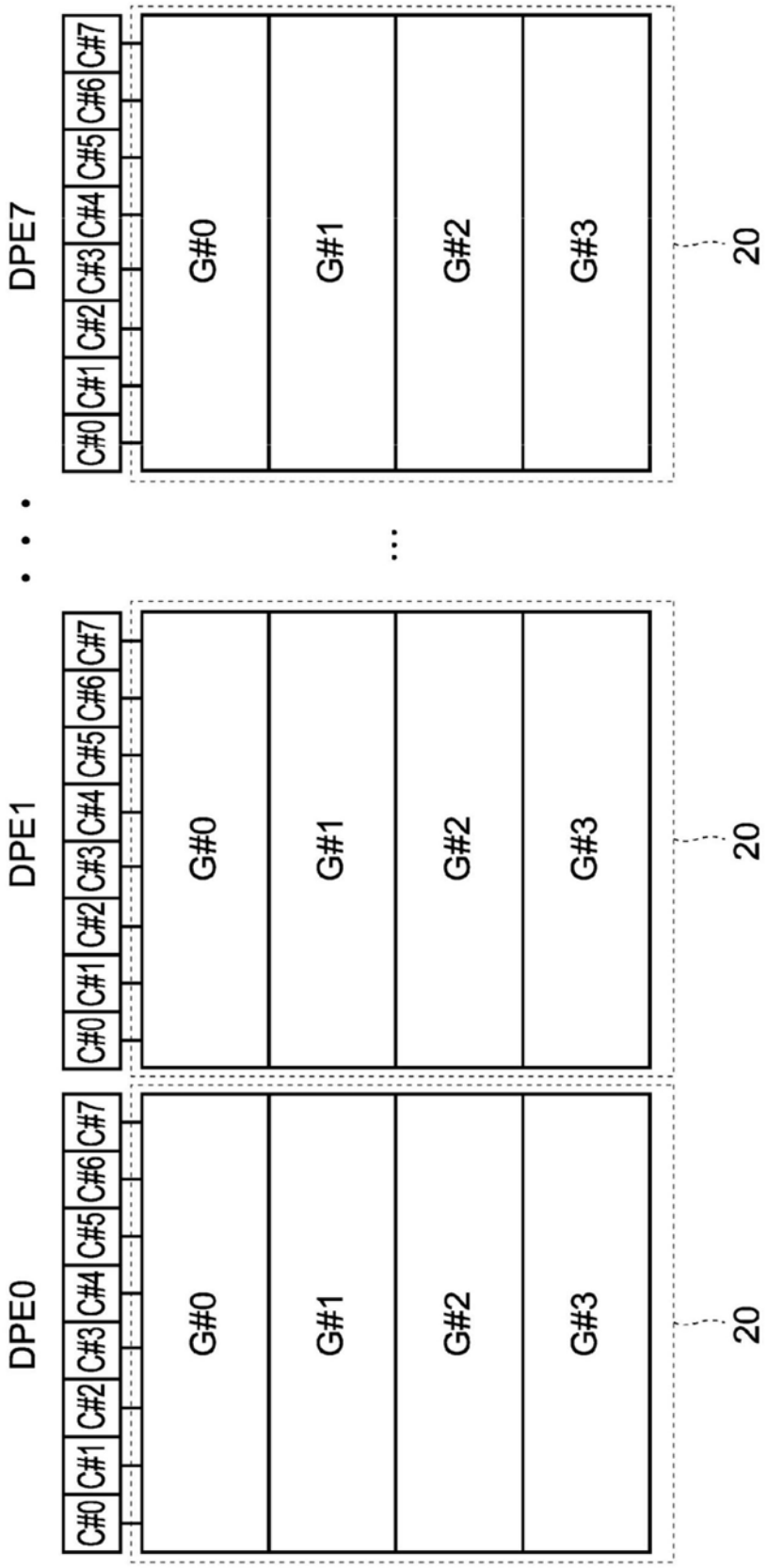


图7

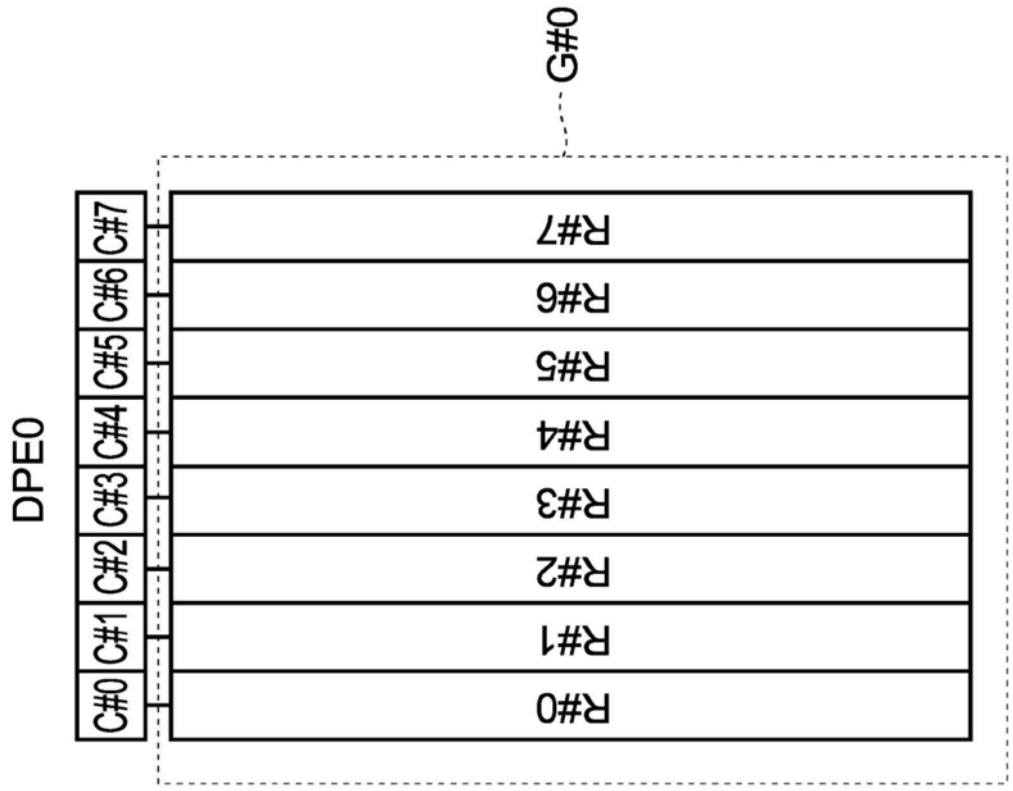

G#0
DPE0
C#0
C#1
C#2
C#3
C#4
C#5
C#6
C#7
R#0
R#1
R#2
R#3
R#4
R#5
R#6
R#7

图8

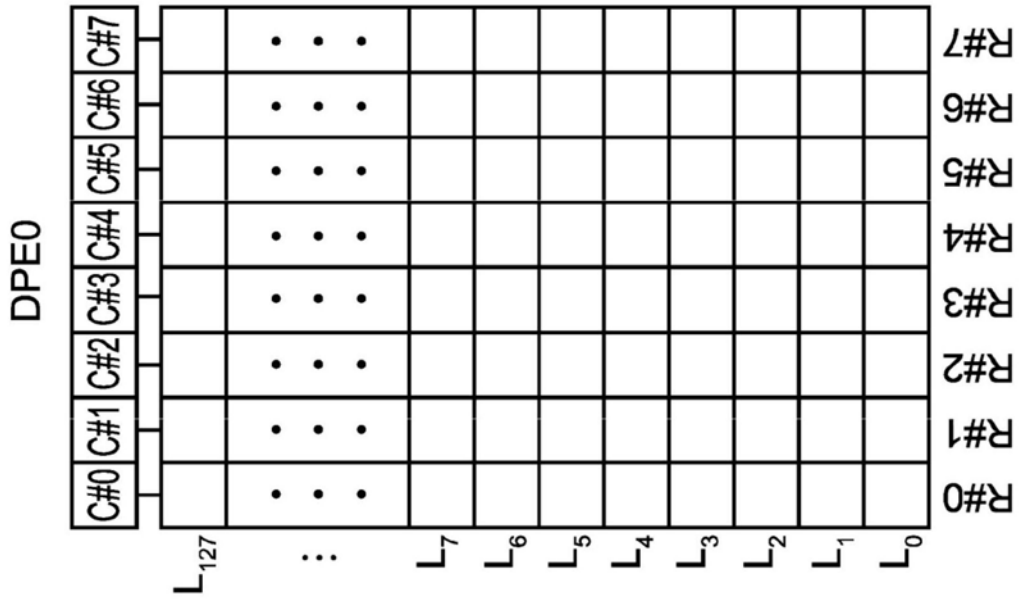

DPE0
C#0
C#1
C#2
C#3
C#4
C#5
C#6
C#7
R#0
R#1
R#2
R#3
R#4
R#5
R#6
R#7
L127
L7
L6
L5
L4
L3
L2
L1
L0

图9

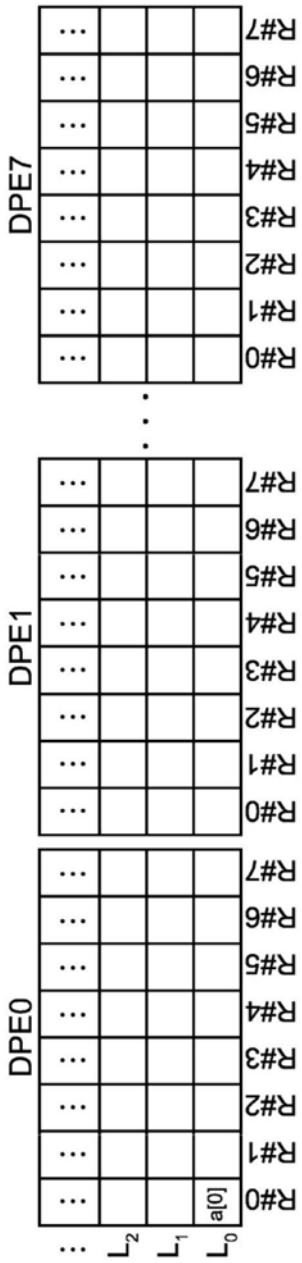


图10A

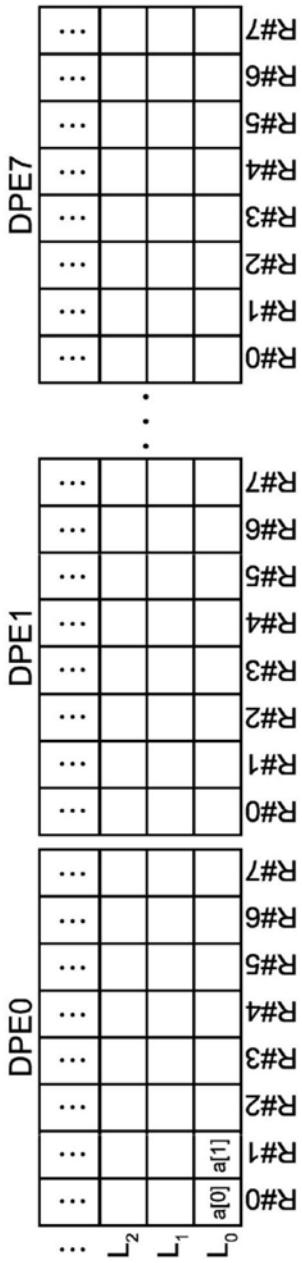


图10B

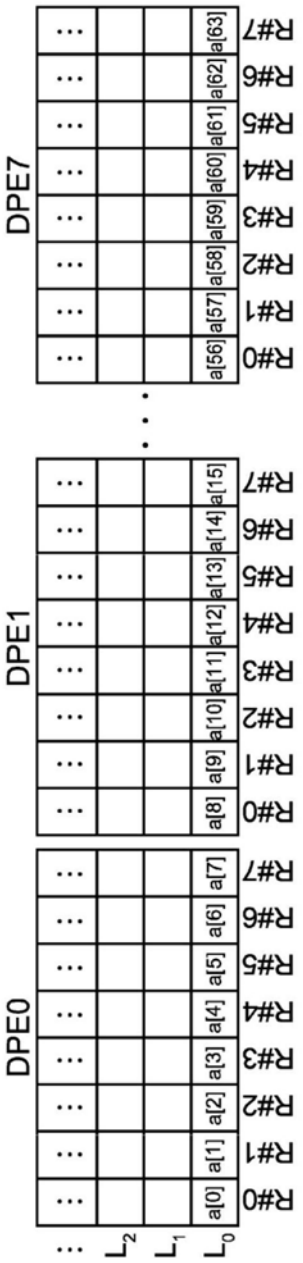

DPE7
DPE1
DPE0
R#0
R#1
R#2
R#3
R#4
R#5
R#6
R#7
a[0]
a[1]
a[2]
a[3]
a[4]
a[5]
a[6]
a[7]
a[8]
a[9]
a[10]
a[11]
a[12]
a[13]
a[14]
a[15]
a[56]
a[57]
a[58]
a[59]
a[60]
a[61]
a[62]
a[63]
L0
L1
L2

图10C

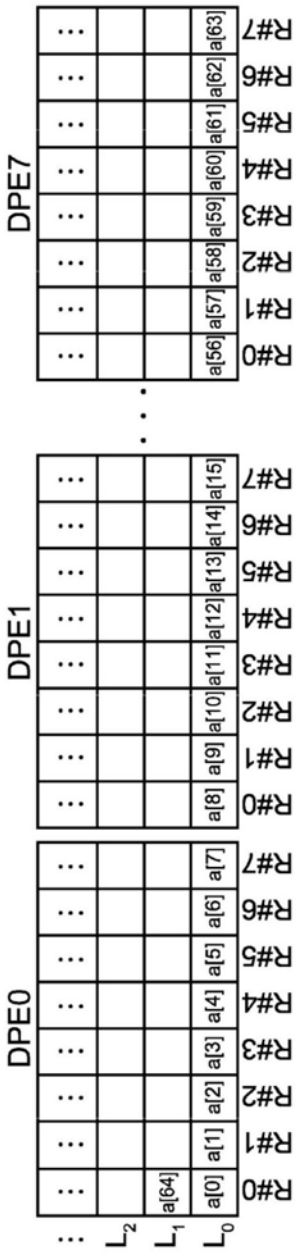

DPE7
DPE1
DPE0
R#0
R#1
R#2
R#3
R#4
R#5
R#6
R#7
a[0]
a[1]
a[2]
a[3]
a[4]
a[5]
a[6]
a[7]
a[8]
a[9]
a[10]
a[11]
a[12]
a[13]
a[14]
a[15]
a[56]
a[57]
a[58]
a[59]
a[60]
a[61]
a[62]
a[63]
a[64]
L0
L1
L2

图11A

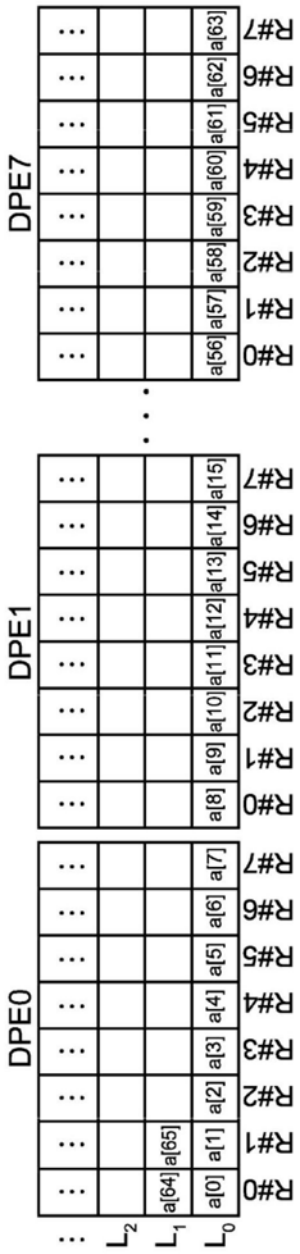

DPE7
DPE1
DPE0
R#0
R#1
R#2
R#3
R#4
R#5
R#6
R#7
a[0]
a[1]
a[2]
a[3]
a[4]
a[5]
a[6]
a[7]
a[8]
a[9]
a[10]
a[11]
a[12]
a[13]
a[14]
a[15]
a[56]
a[57]
a[58]
a[59]
a[60]
a[61]
a[62]
a[63]
a[64]
a[65]
L0
L1
L2

图11B

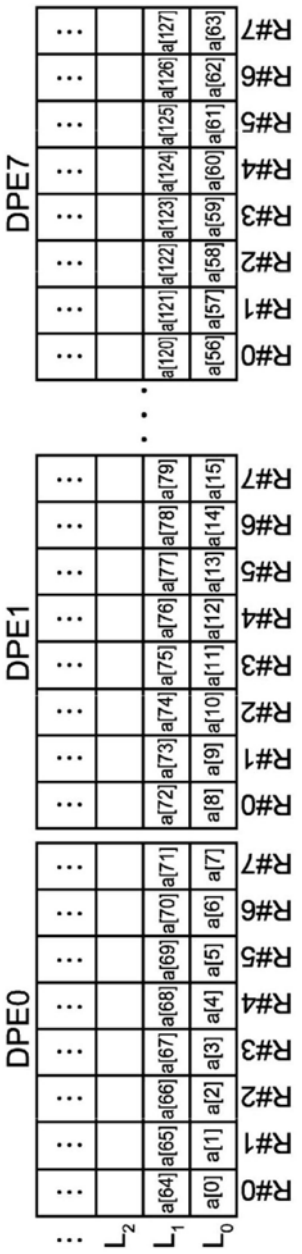


图11C

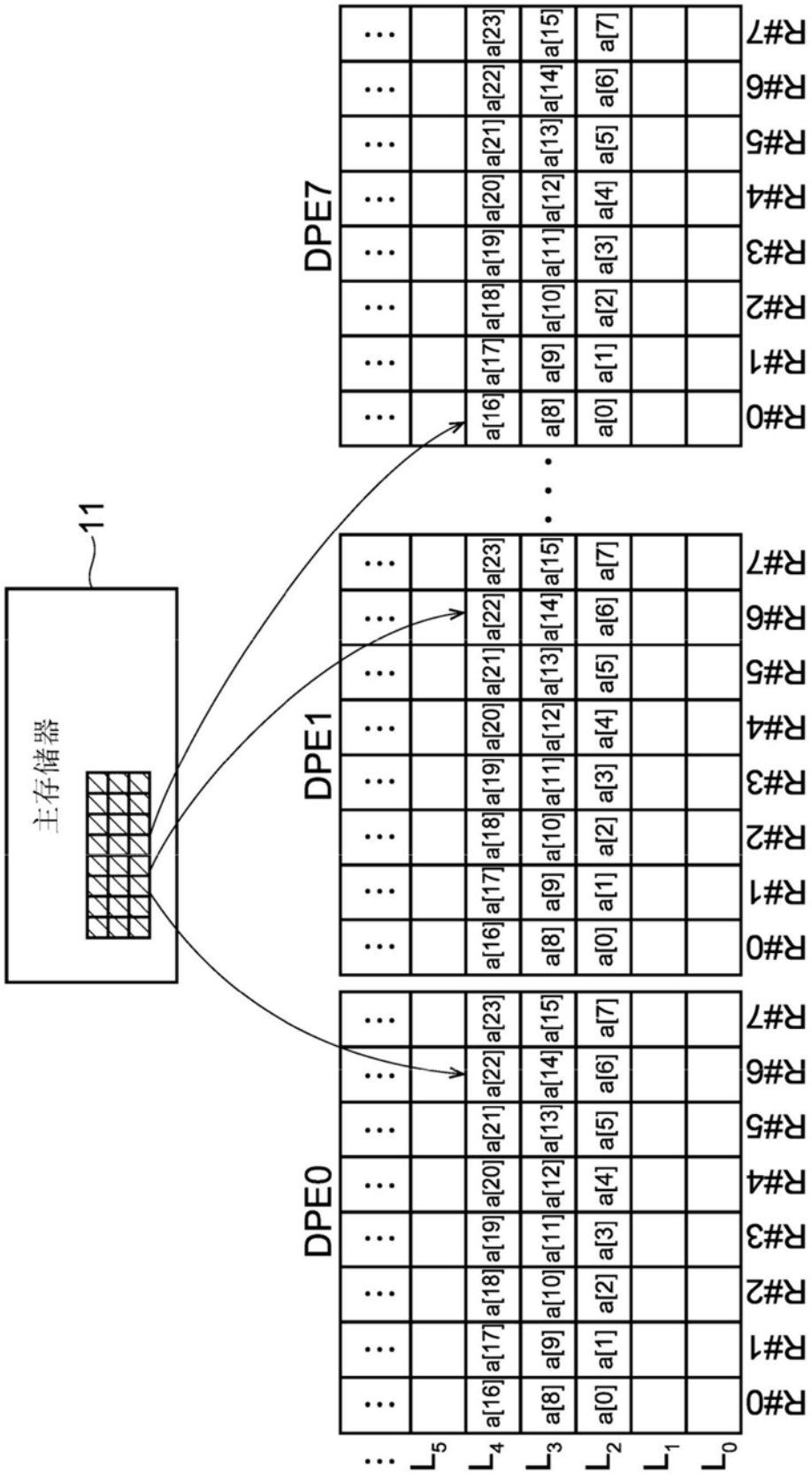


图12

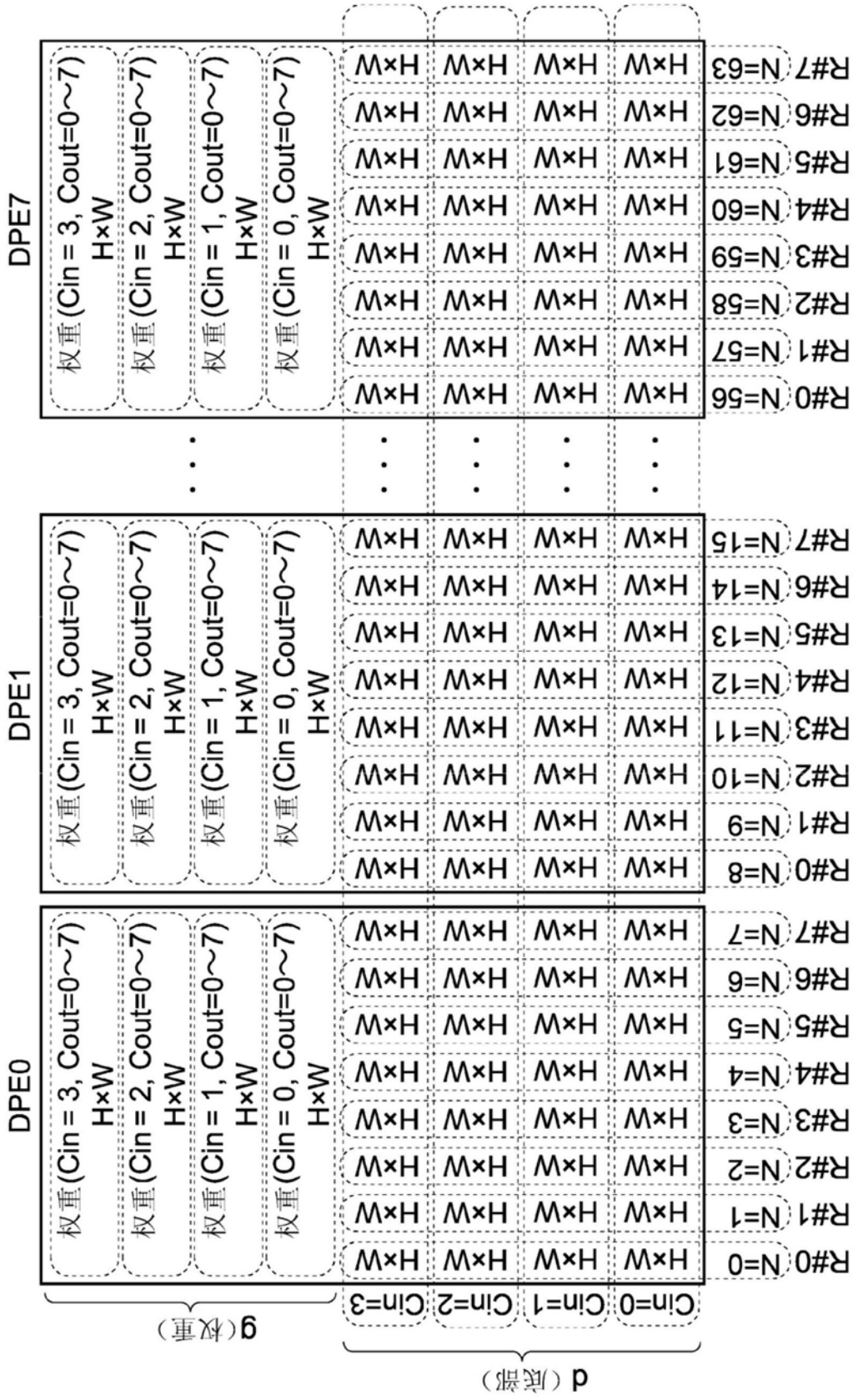

DPE7
DPE1
DPE0
权重(Cin = 3, Cout=0~7) H×W
权重(Cin = 2, Cout=0~7) H×W
权重(Cin = 1, Cout=0~7) H×W
权重(Cin = 0, Cout=0~7) H×W
R#0 N=0
R#1 N=1
R#2 N=2
R#3 N=3
R#4 N=4
R#5 N=5
R#6 N=6
R#7 N=7
R#0 N=8
R#1 N=9
R#2 N=10
R#3 N=11
R#4 N=12
R#5 N=13
R#6 N=14
R#7 N=15
R#0 N=56
R#1 N=57
R#2 N=58
R#3 N=59
R#4 N=60
R#5 N=61
R#6 N=62
R#7 N=63
H×W
Cin=0
Cin=1
Cin=2
Cin=3
d（底部）
g（权重）

图13

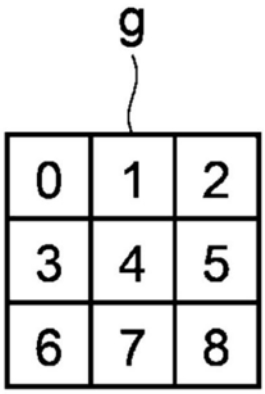


图14

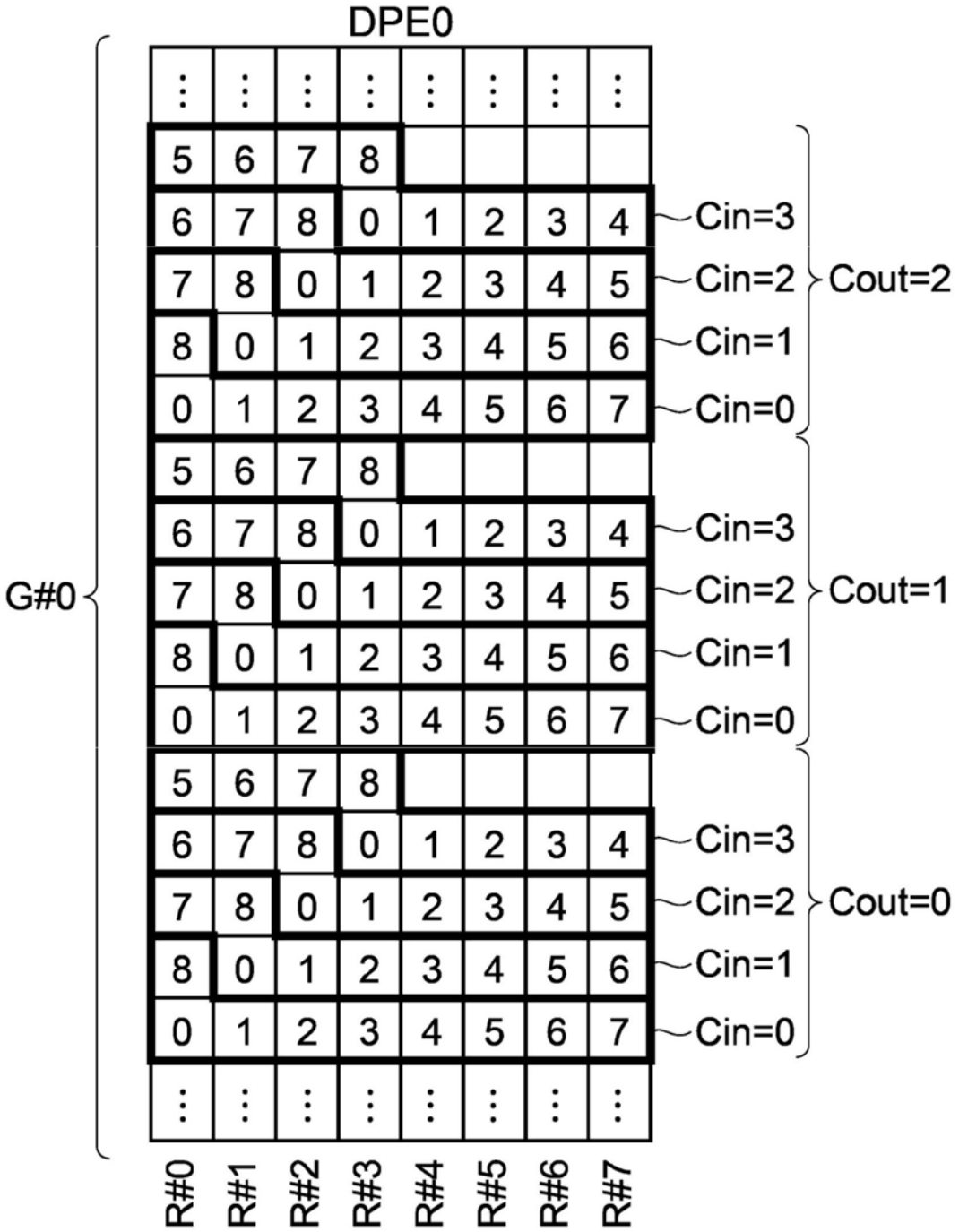

DPE0
5 6 7 8
6 7 8 0 1 2 3 4
7 8 0 1 2 3 4 5
8 0 1 2 3 4 5 6
0 1 2 3 4 5 6 7
Cin=3
Cin=2
Cin=1
Cin=0
Cout=2
Cout=1
Cout=0
G#0
R#0 R#1 R#2 R#3 R#4 R#5 R#6 R#7

图15

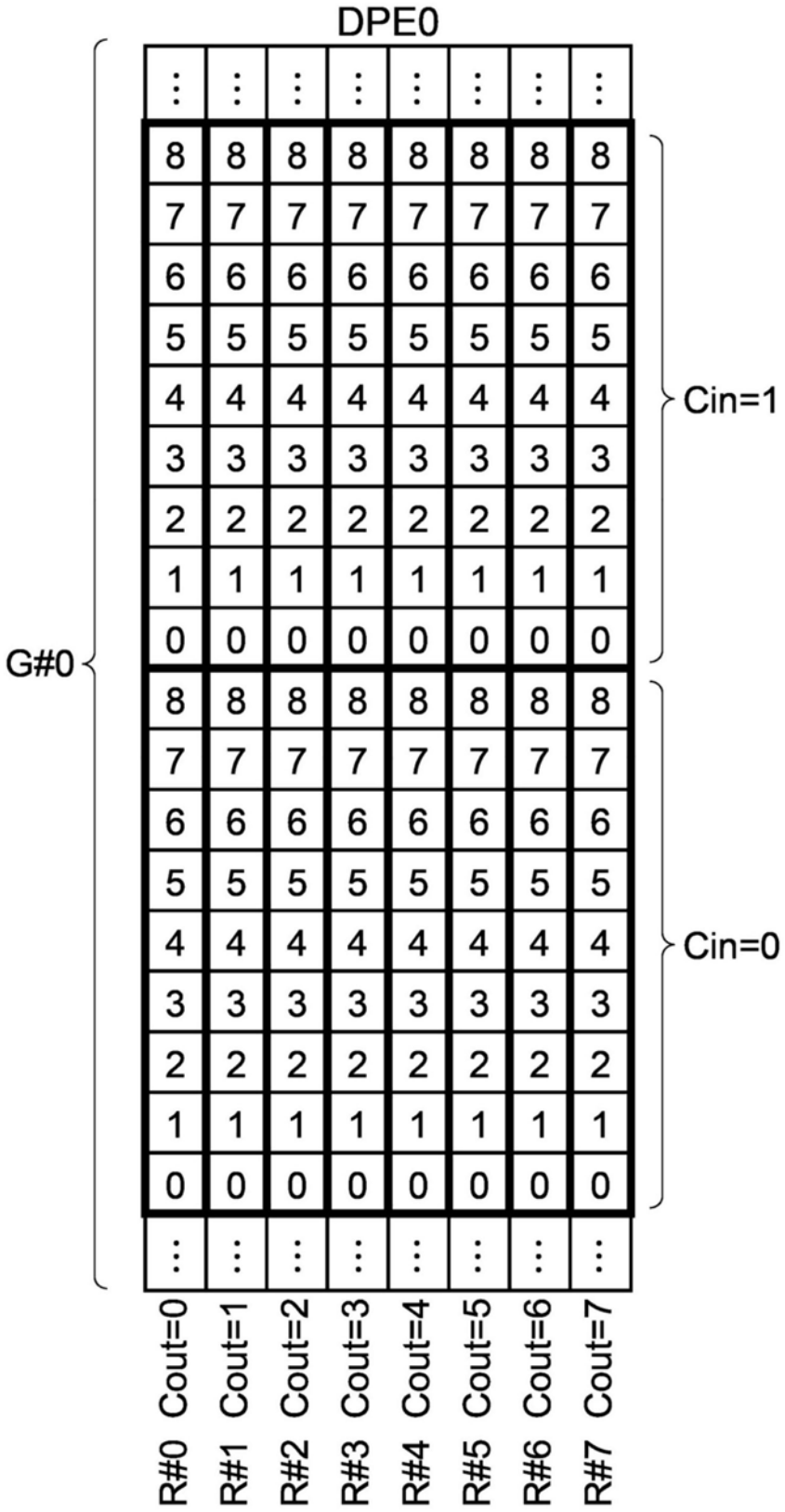

DPE0
G#0
Cin=1
Cin=0
R#0 Cout=0
R#1 Cout=1
R#2 Cout=2
R#3 Cout=3
R#4 Cout=4
R#5 Cout=5
R#6 Cout=6
R#7 Cout=7

图16

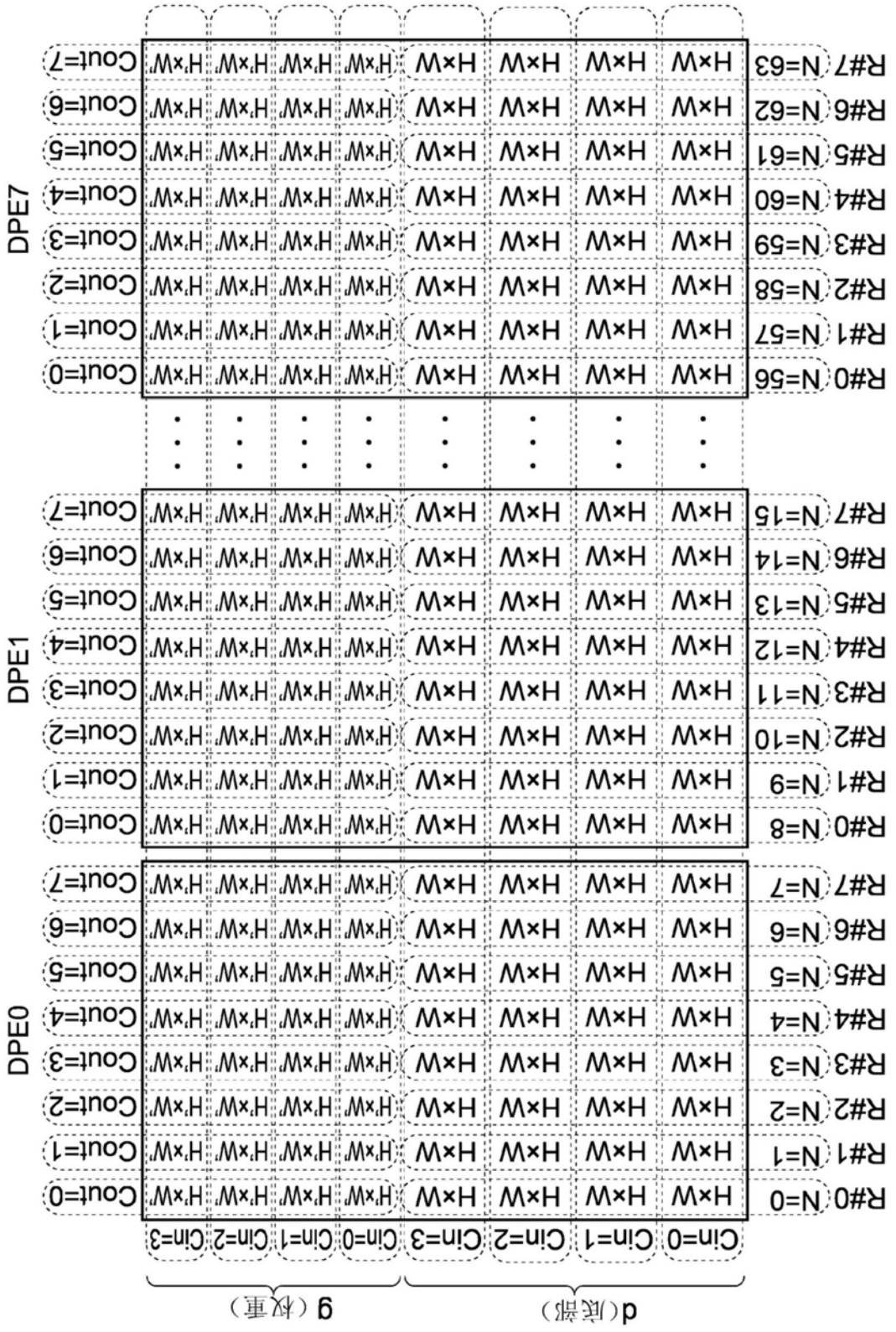

DPE7
DPE1
DPE0
R#0 N=56
R#1 N=57
R#2 N=58
R#3 N=59
R#4 N=60
R#5 N=61
R#6 N=62
R#7 N=63
R#0 N=8
R#1 N=9
R#2 N=10
R#3 N=11
R#4 N=12
R#5 N=13
R#6 N=14
R#7 N=15
R#0 N=0
R#1 N=1
R#2 N=2
R#3 N=3
R#4 N=4
R#5 N=5
R#6 N=6
R#7 N=7
H×W
H'×W'
Cout=0
Cout=1
Cout=2
Cout=3
Cout=4
Cout=5
Cout=6
Cout=7
Cin=0
Cin=1
Cin=2
Cin=3
d（底部）
g（权重）

图17

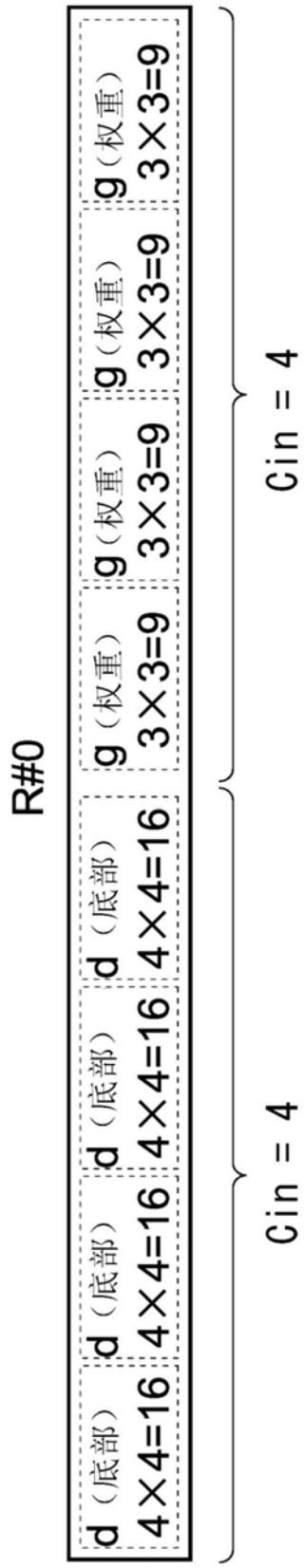

R#0
d（底部）
4×4=16
d（底部）
4×4=16
d（底部）
4×4=16
d（底部）
4×4=16
g（权重）
3×3=9
g（权重）
3×3=9
g（权重）
3×3=9
g（权重）
3×3=9
Cin = 4
Cin = 4

图18

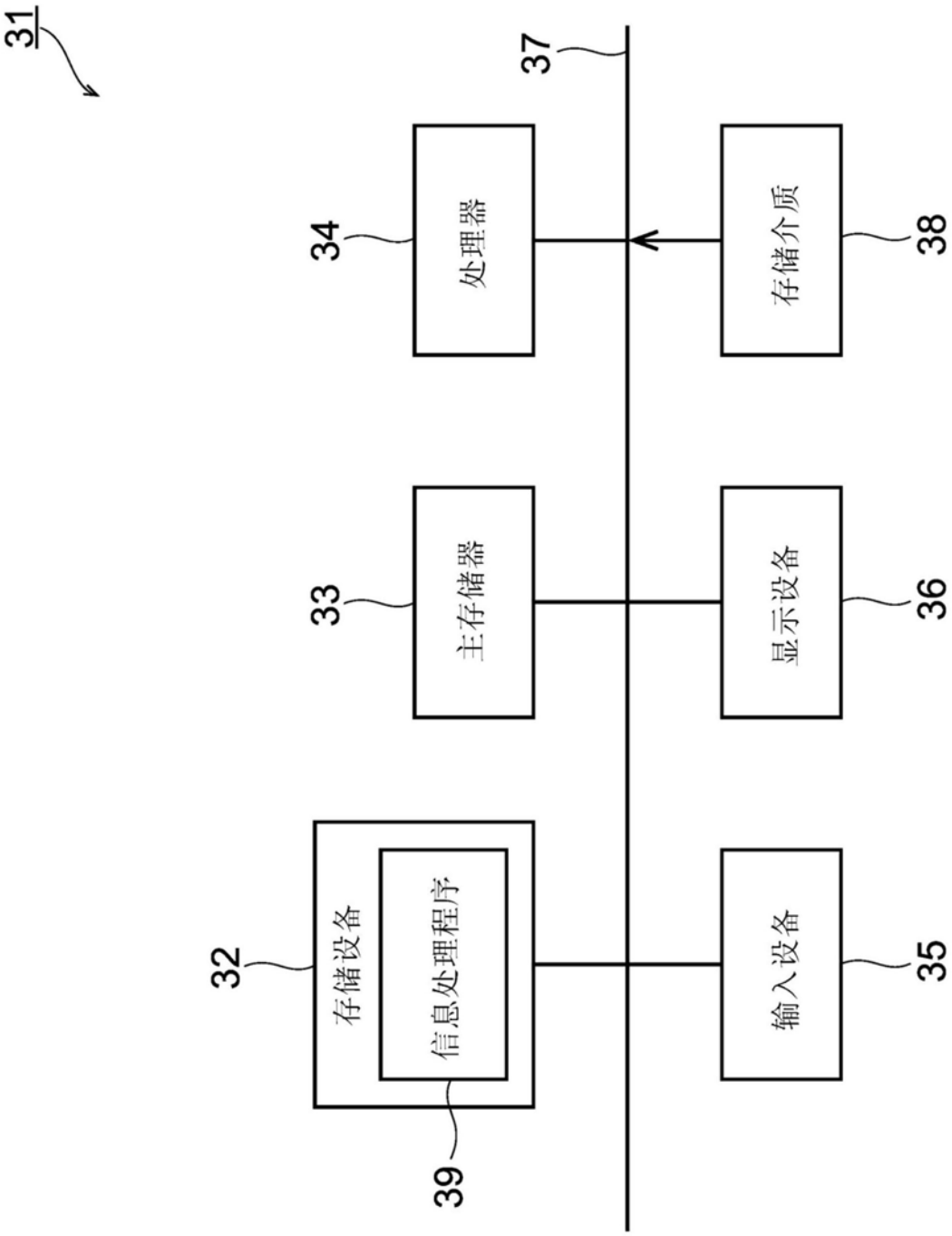


图19

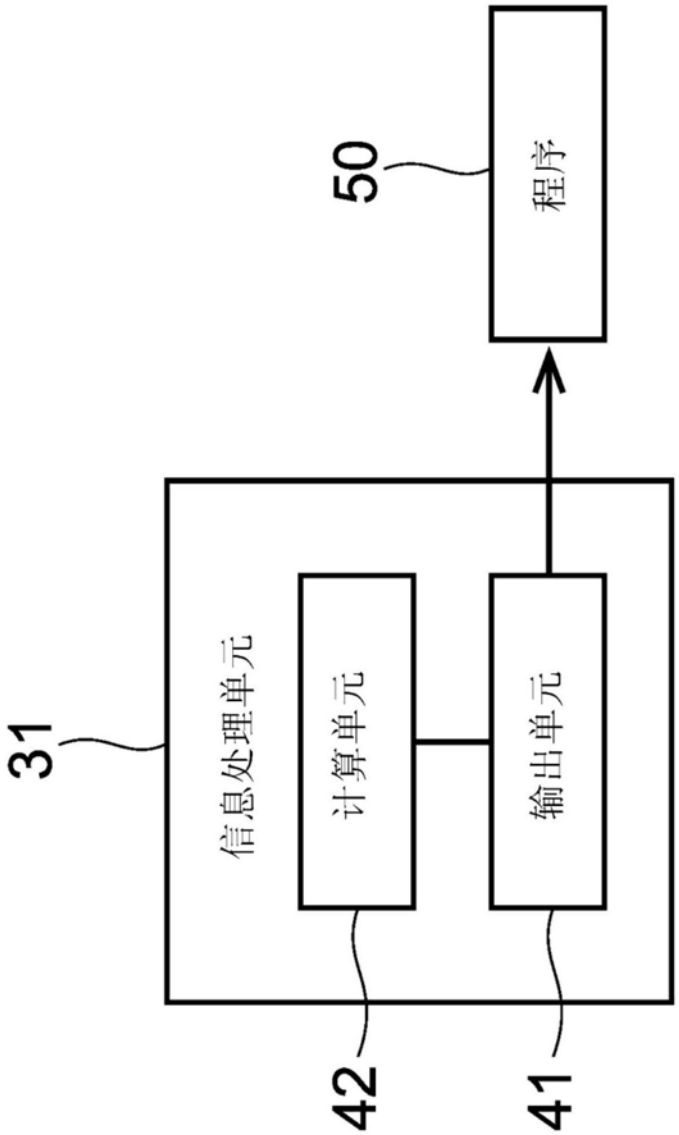


图20

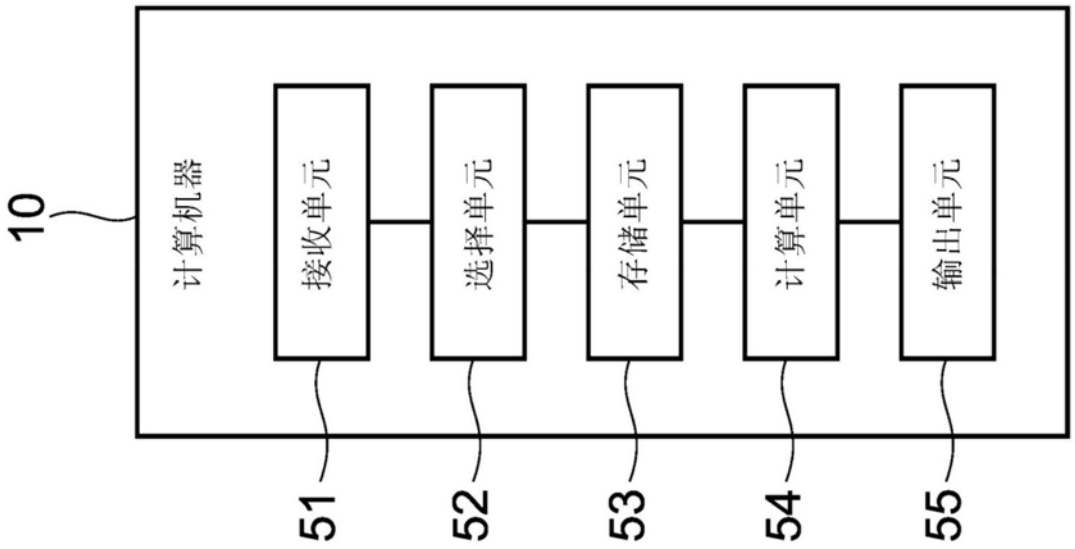


图21

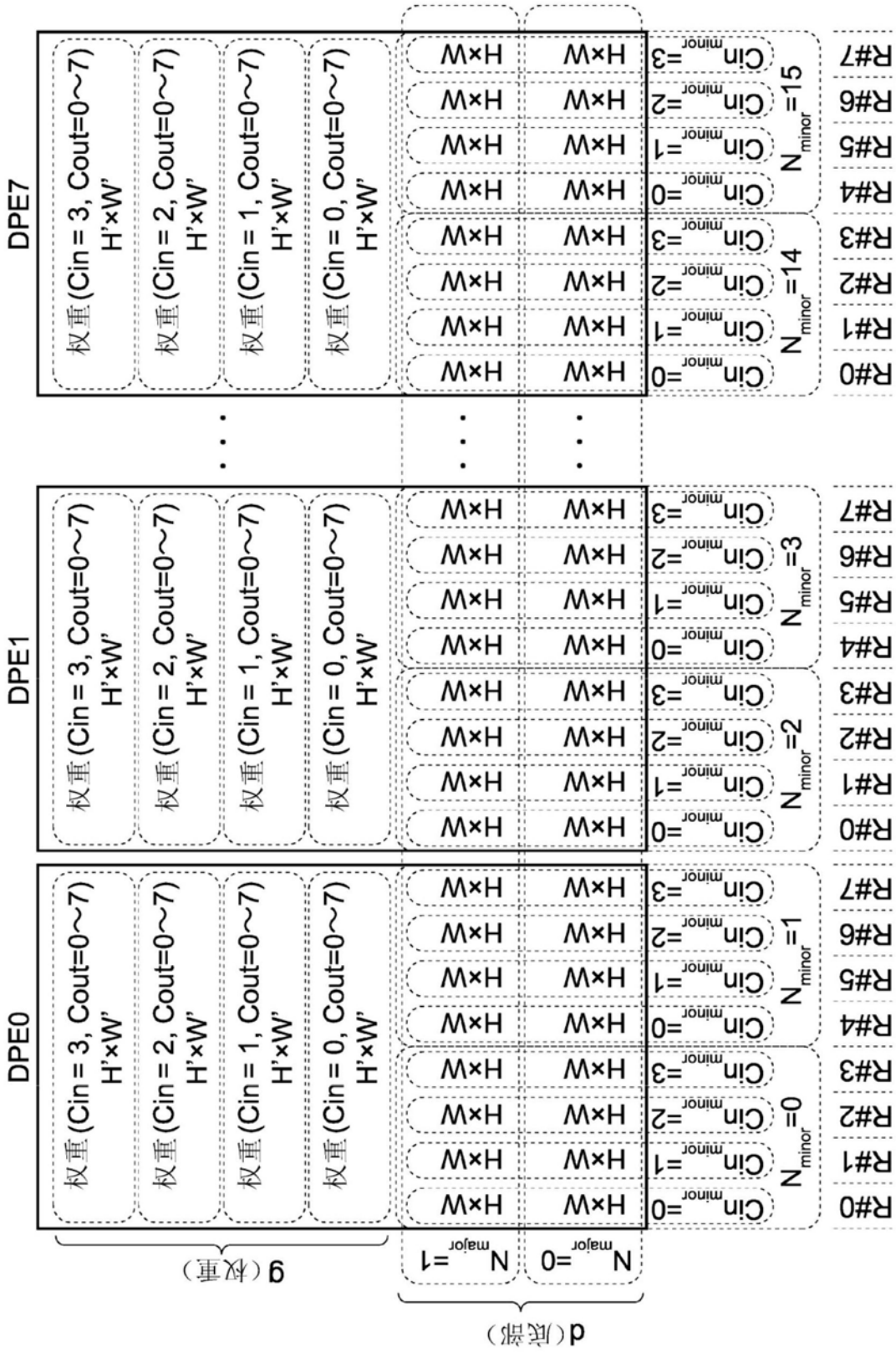

DPE0
DPE1
DPE7
权重 (Cin = 3, Cout=0~7) H'×W'
权重 (Cin = 2, Cout=0~7) H'×W'
权重 (Cin = 1, Cout=0~7) H'×W'
权重 (Cin = 0, Cout=0~7) H'×W'
H×W
R#0
R#1
R#2
R#3
R#4
R#5
R#6
R#7
Cin minor =0
Cin minor =1
Cin minor =2
Cin minor =3
N minor =0
N minor =1
N minor =2
N minor =3
N minor =14
N minor =15
N major =0
N major =1
d (底部)
g (权重)

图22

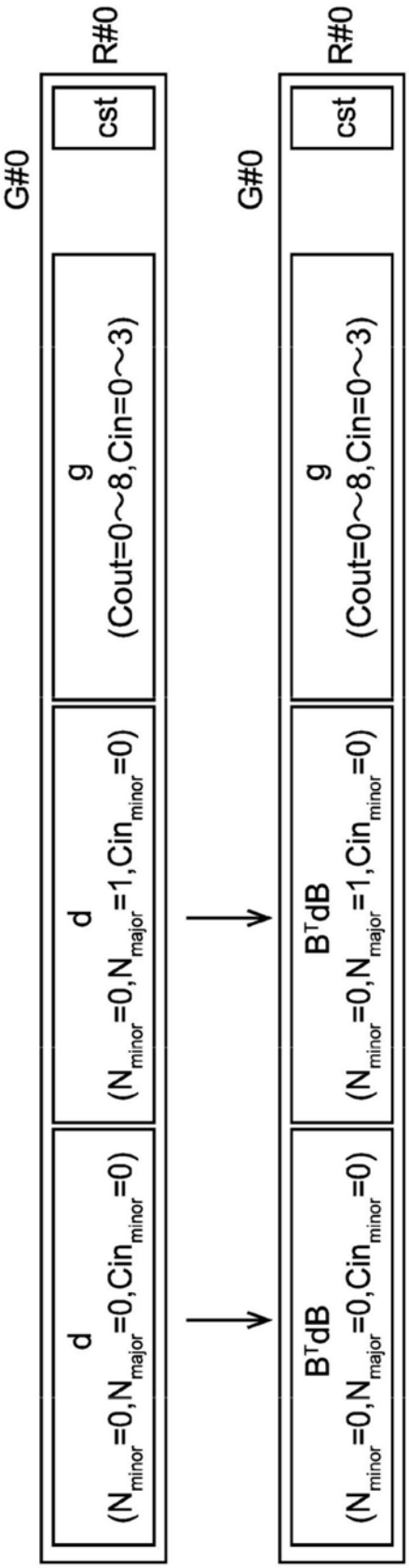

G#0
R#0
d
(N_{minor}=0,N_{major}=0,Cin_{minor}=0)
d
(N_{minor}=0,N_{major}=1,Cin_{minor}=0)
g
(Cout=0~8,Cin=0~3)
cst
G#0
R#0
B^TdB
(N_{minor}=0,N_{major}=0,Cin_{minor}=0)
B^TdB
(N_{minor}=0,N_{major}=1,Cin_{minor}=0)
g
(Cout=0~8,Cin=0~3)
cst

图23A

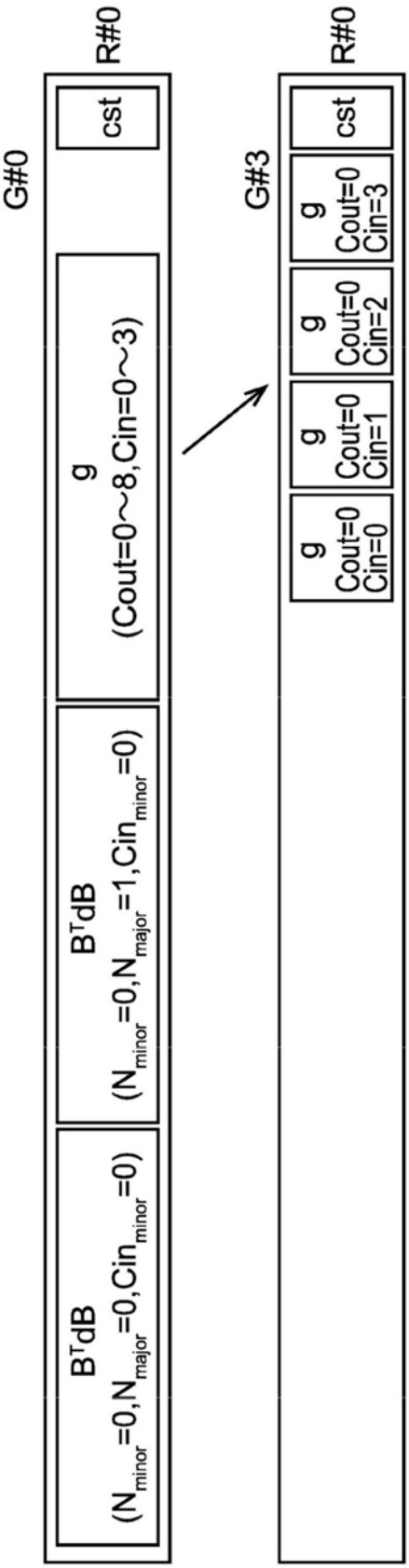

G#0
R#0
cst
g
(Cout=0~8,Cin=0~3)
BᵀdB
(N_minor=0,N_major=1,Cin_minor=0)
BᵀdB
(N_minor=0,N_major=0,Cin_minor=0)
G#3
R#0
cst
g
Cout=0
Cin=3
g
Cout=0
Cin=2
g
Cout=0
Cin=1
g
Cout=0
Cin=0

图23B

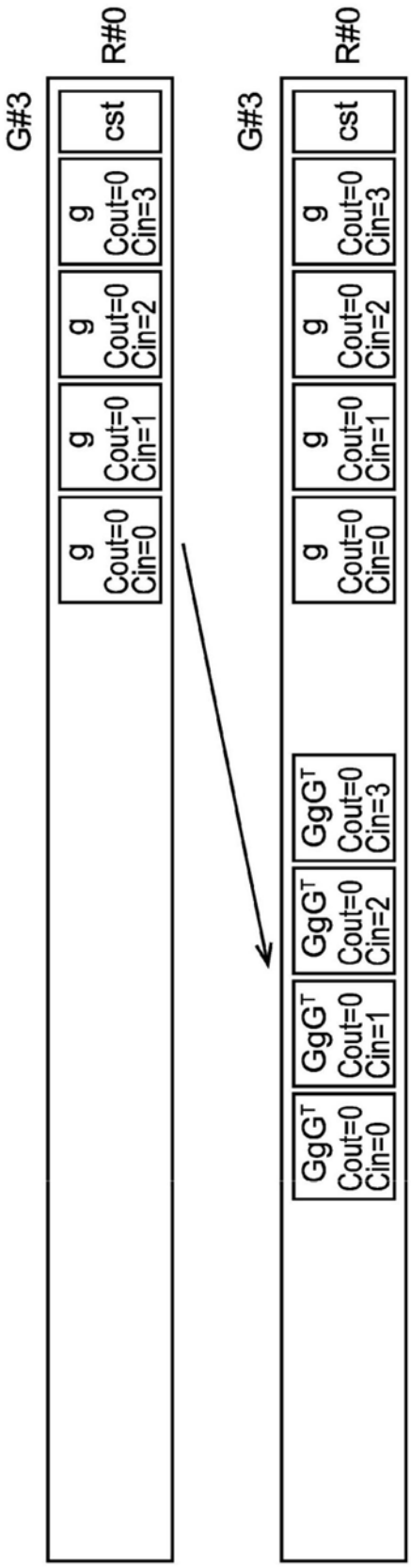

G#3
R#0
g Cout=0 Cin=0
g Cout=0 Cin=1
g Cout=0 Cin=2
g Cout=0 Cin=3
cst
GgG^T Cout=0 Cin=0
GgG^T Cout=0 Cin=1
GgG^T Cout=0 Cin=2
GgG^T Cout=0 Cin=3

图24

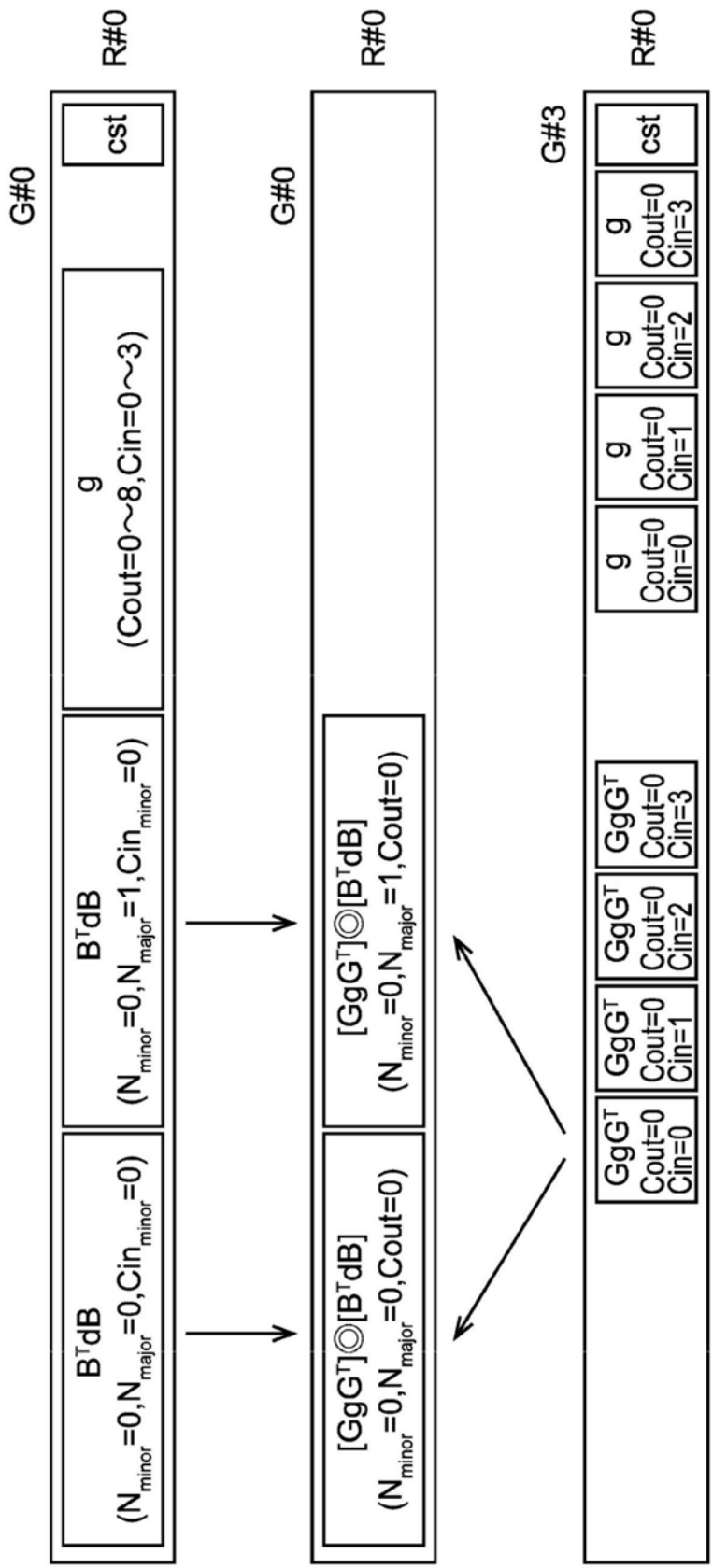

R#0
G#0
cst
g
(Cout=0～8,Cin=0～3)
BᵀdB
(N_minor=0,N_major=0,Cin_minor=0)
BᵀdB
(N_minor=0,N_major=1,Cin_minor=0)
R#0
G#0
[GgGᵀ]◎[BᵀdB]
(N_minor=0,N_major=0,Cout=0)
[GgGᵀ]◎[BᵀdB]
(N_minor=0,N_major=1,Cout=0)
R#0
G#3
GgGᵀ Cout=0 Cin=0
GgGᵀ Cout=0 Cin=1
GgGᵀ Cout=0 Cin=2
GgGᵀ Cout=0 Cin=3
g Cout=0 Cin=0
g Cout=0 Cin=1
g Cout=0 Cin=2
g Cout=0 Cin=3
cst

图25

1. b[0] = a[2]; // g_2

2. a[3] = a[1]*1/2; // $\frac{1}{2}g_1$

3. b[1] = a[0] + b[0]; // g_0+g_2

4. b[0] += a[0]*1/4 ; // $\frac{1}{4}g_0+g_2$

5. a[4] = b[0] - a[3]; // $\frac{1}{4}g_0-\frac{1}{2}g_1+g_2=x_4$

6. a[5] = b[0] + a[3]; // $\frac{1}{4}g_0+\frac{1}{2}g_1+g_2=x_3$

7. a[3] = b[1] - a[1]; // $g_0-g_1+g_2=x_2$

8. a[1] = b[1] + a[1]; // $g_0+g_1+g_2=x_1$

图26

```
1.   b[2] = a[2];          // d2
2.   b[3] = a[3];          // d3

3.   b[0] = a[0] - b[2];   // d0 - d2
4.   a[0] = a[4] - b[2];   // - d2 + d4

5.   a[4] += (-4)*b[2];    // - 4d2 + d4
6.   b[1] = b[3] - a[1];   // - d1 + d3

7.   a[5] = a[5] - b[3];   // - d3 + d5
8.   b[3] += (-4)*a[1];    // - 4d1 + d3
9.   b[1] *=2;             // - 2d1 + 2d3

10.  a[5] += (-2)*b[1];    // 4d1 - 5d3 + d5 = x5
11.  a[1] = b[3] + a[4];   // - 4d1 - 4d2 + d3 + d4= x1

12.  a[2] = a[4] - b[3];   // 4d1 - 4d2 - d3 + d4= x2
13.  a[3] = a[0] + b[1];   // - 2d1 - d2 + 2d3 + d4= x3

14.  a[4] = a[0] - b[1];   // 2d1 - d2 - 2d3 + d4= x4
15.  a[0] += 4*b[0];       // 4d0 - 5d2 + d4= x0
```

图27

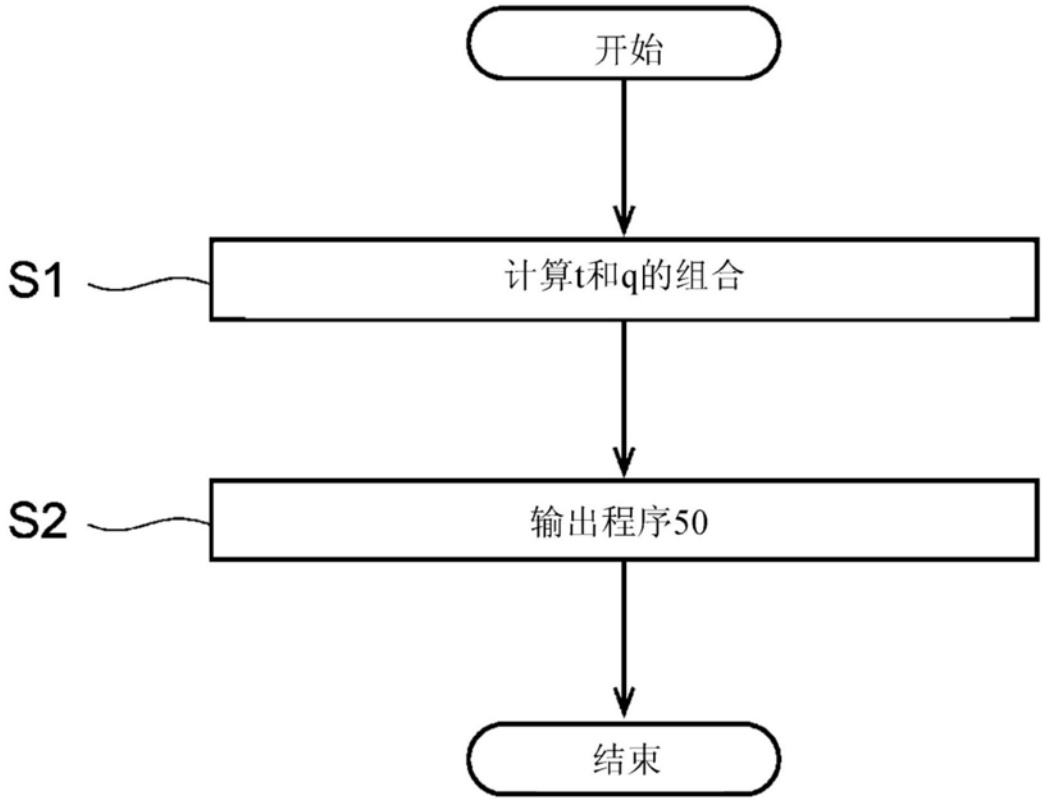


图28

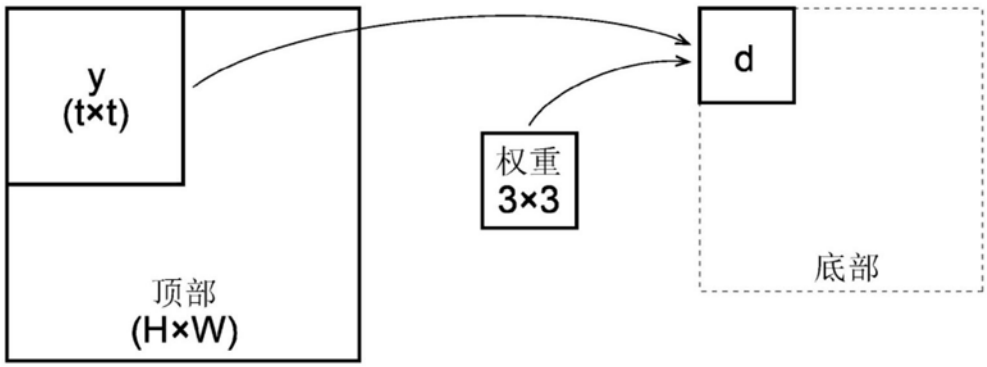


图29A

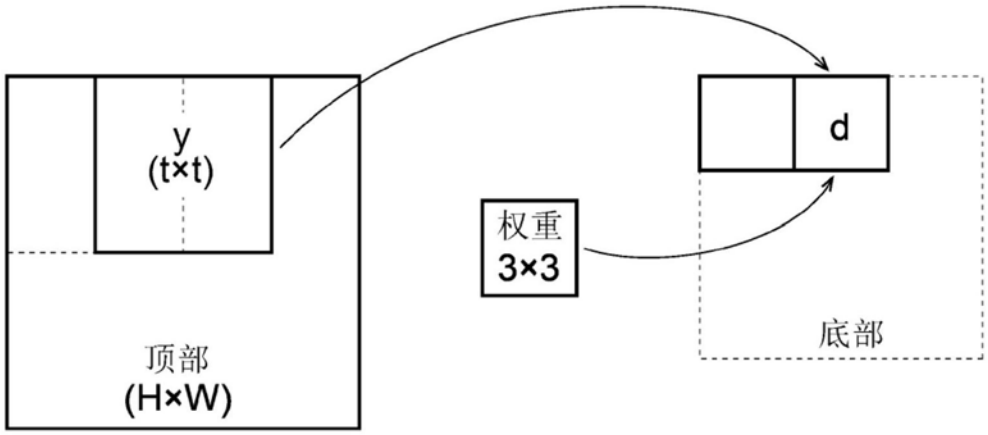


图29B

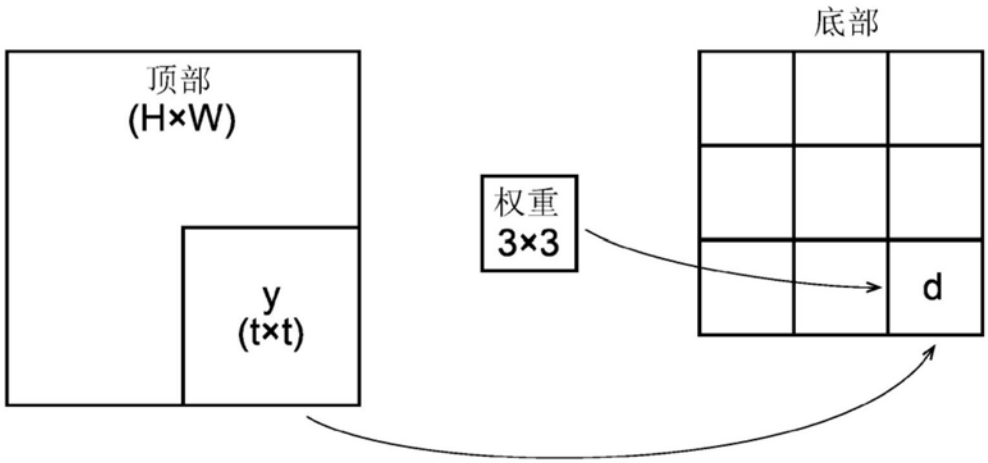


图29C

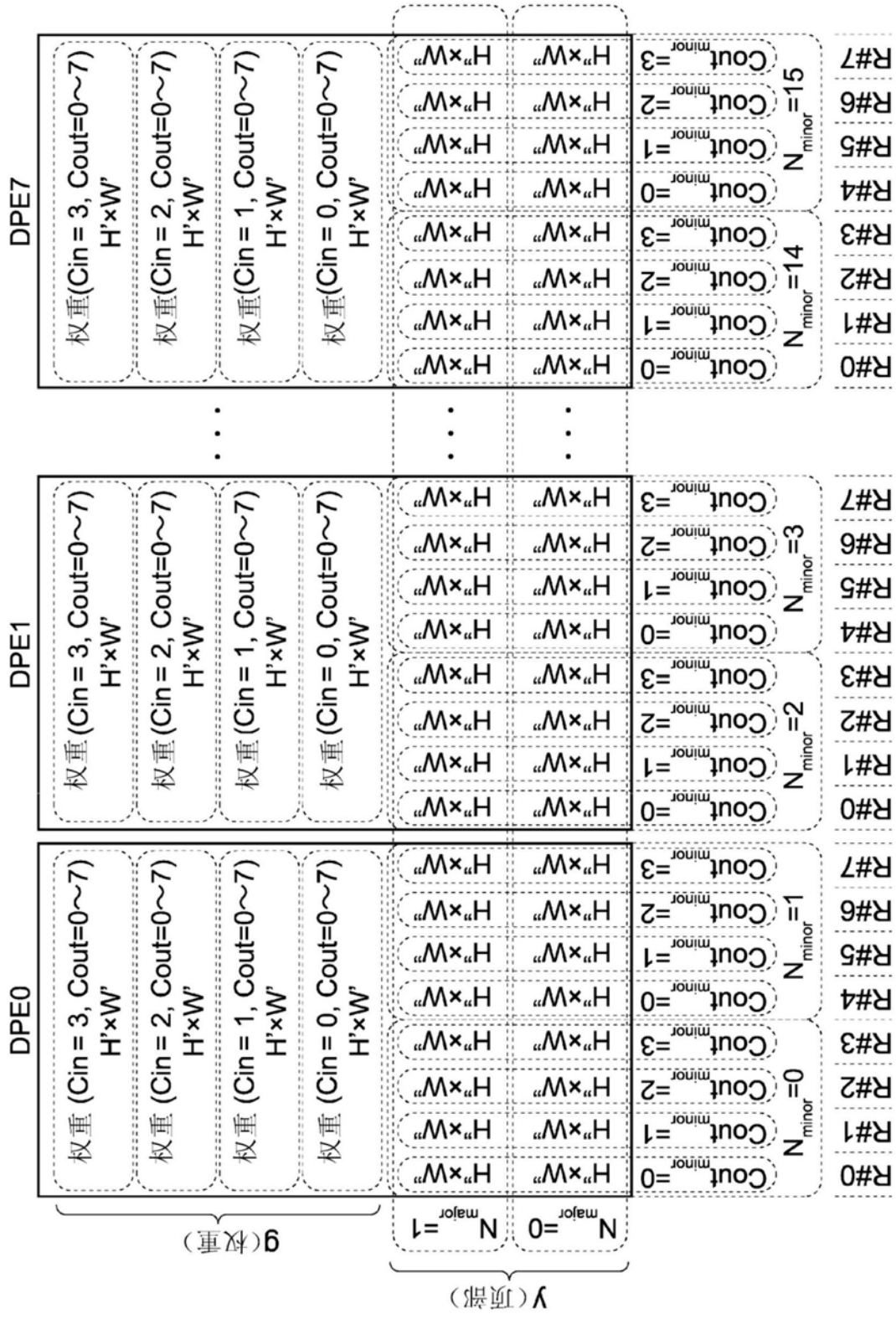

DPE0
DPE1
DPE7
权重 (Cin = 3, Cout=0~7) H'×W'
权重 (Cin = 2, Cout=0~7) H'×W'
权重 (Cin = 1, Cout=0~7) H'×W'
权重 (Cin = 0, Cout=0~7) H'×W'
H"×W"
Cout_minor=0
Cout_minor=1
Cout_minor=2
Cout_minor=3
N_minor=0
N_minor=1
N_minor=2
N_minor=3
N_minor=14
N_minor=15
R#0
R#1
R#2
R#3
R#4
R#5
R#6
R#7
N_major=0
N_major=1
y(顶部)
g(权重)

图30

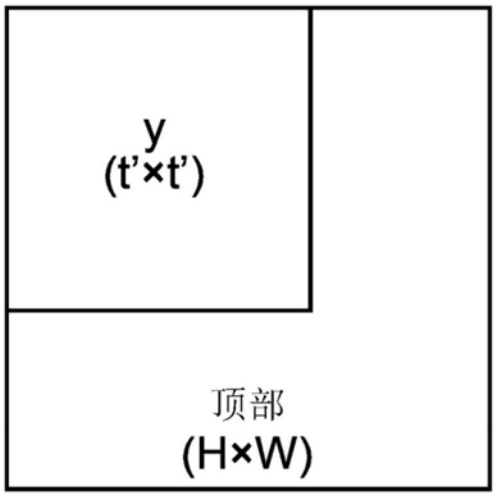


图31A

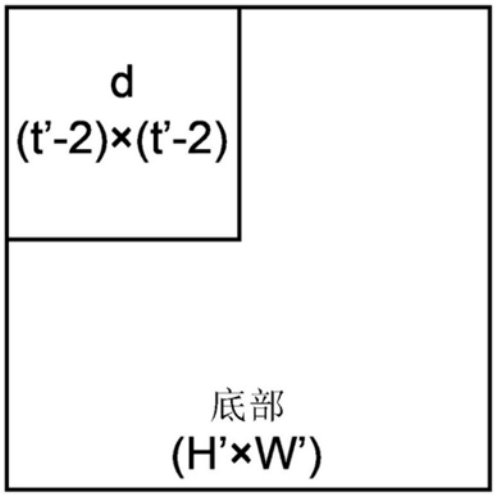


图31B

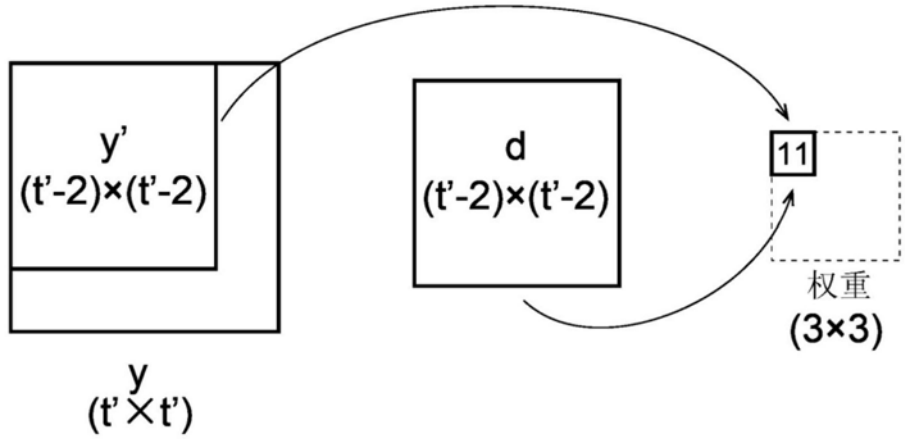


图32A

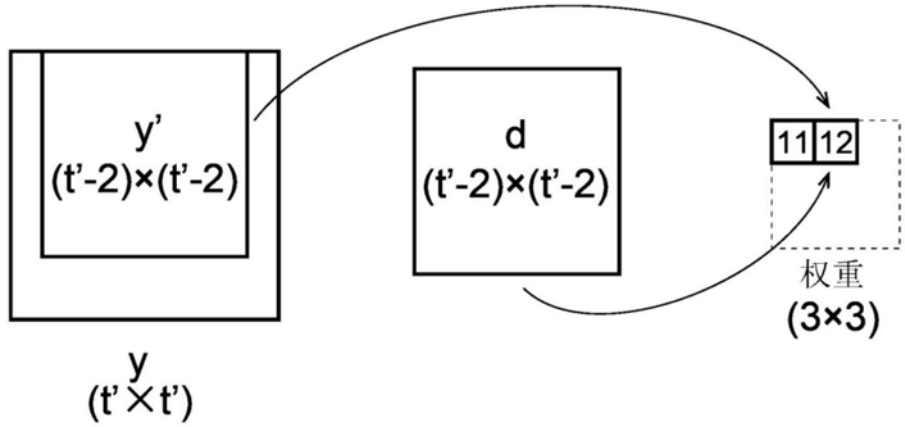


图32B

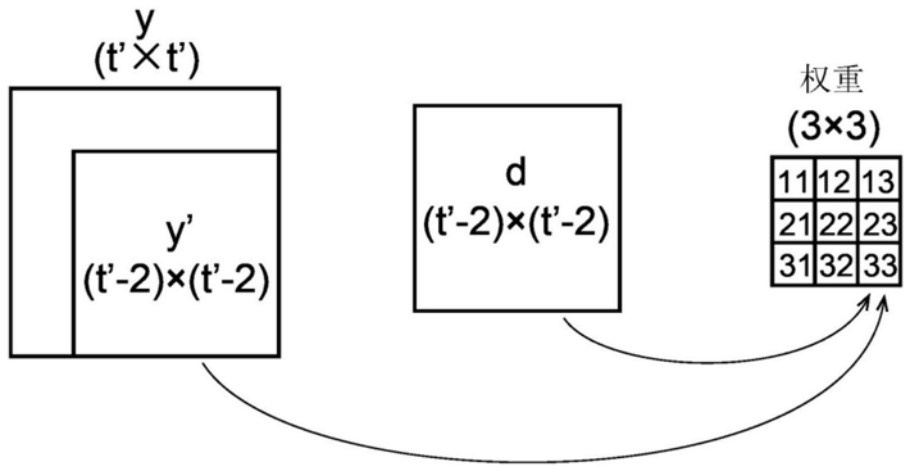


图32C

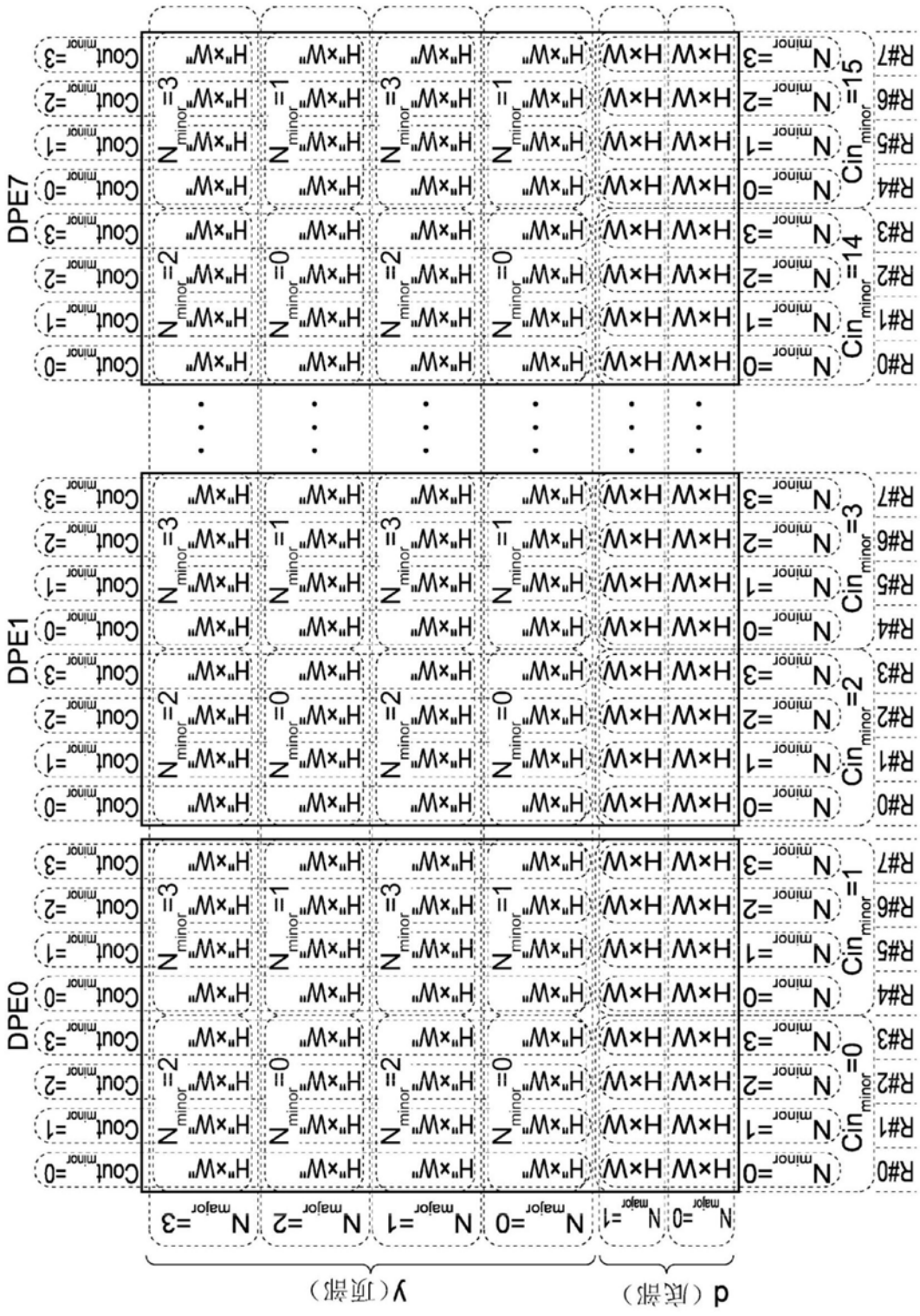


图33

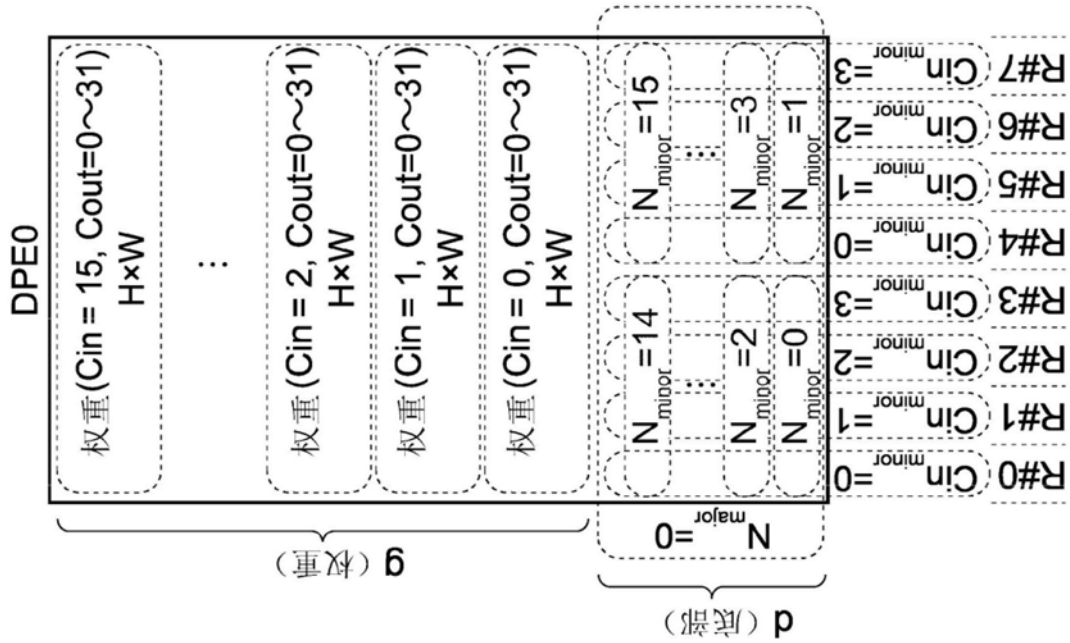

DPE0
权重(Cin = 15, Cout=0～31) H×W
权重(Cin = 2, Cout=0～31) H×W
权重(Cin = 1, Cout=0～31) H×W
权重(Cin = 0, Cout=0～31) H×W
g（权重）
d（底部）
R#0 Cin minor =0
R#1 Cin minor =1
R#2 Cin minor =2
R#3 Cin minor =3
R#4 Cin minor =0
R#5 Cin minor =1
R#6 Cin minor =2
R#7 Cin minor =3
N major =0

图34

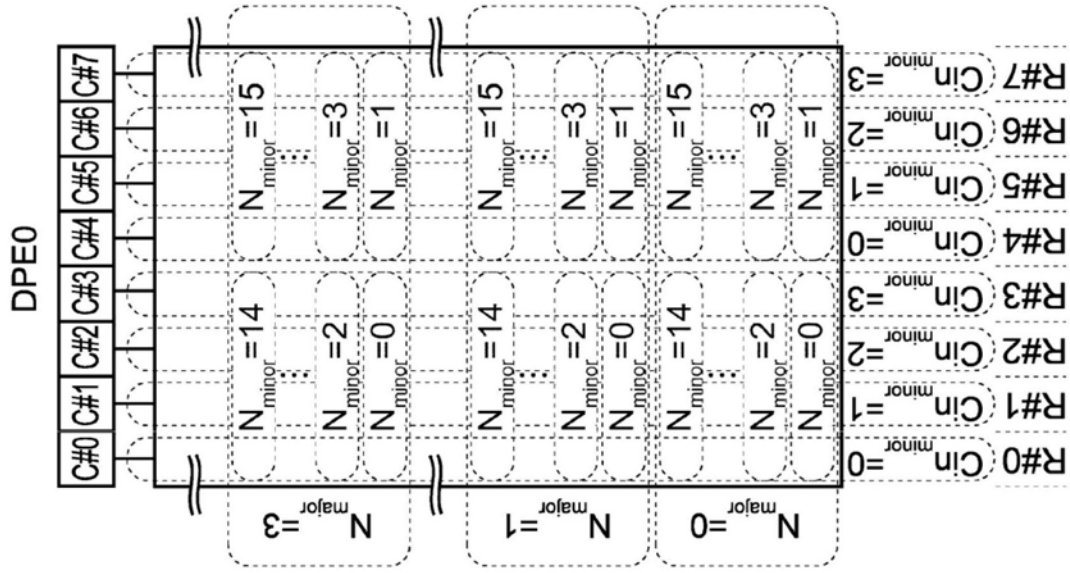

DPE0
C#0 C#1 C#2 C#3 C#4 C#5 C#6 C#7
R#0 Cin minor =0
R#1 Cin minor =1
R#2 Cin minor =2
R#3 Cin minor =3
R#4 Cin minor =0
R#5 Cin minor =1
R#6 Cin minor =2
R#7 Cin minor =3
N major =0
N major =1
N major =3

图35

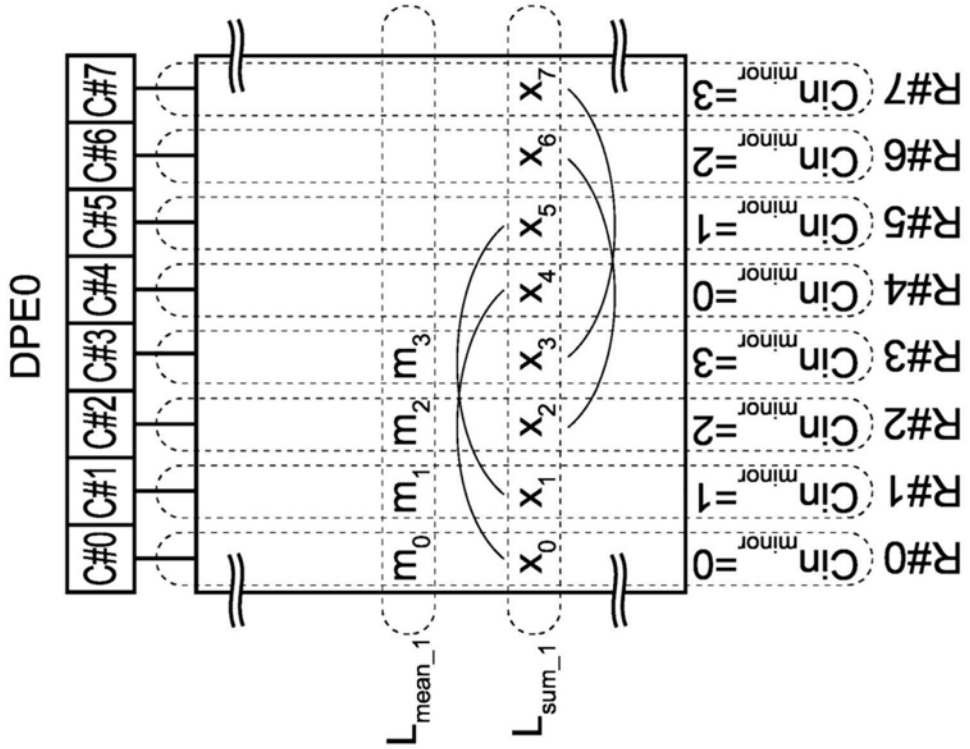

DPE0
C#0
C#1
C#2
C#3
C#4
C#5
C#6
C#7
R#0 Cin minor=0
R#1 Cin minor=1
R#2 Cin minor=2
R#3 Cin minor=3
R#4 Cin minor=0
R#5 Cin minor=1
R#6 Cin minor=2
R#7 Cin minor=3
m0 m1 m2 m3
x0 x1 x2 x3 x4 x5 x6 x7
Lmean_1
Lsum_1

图36A

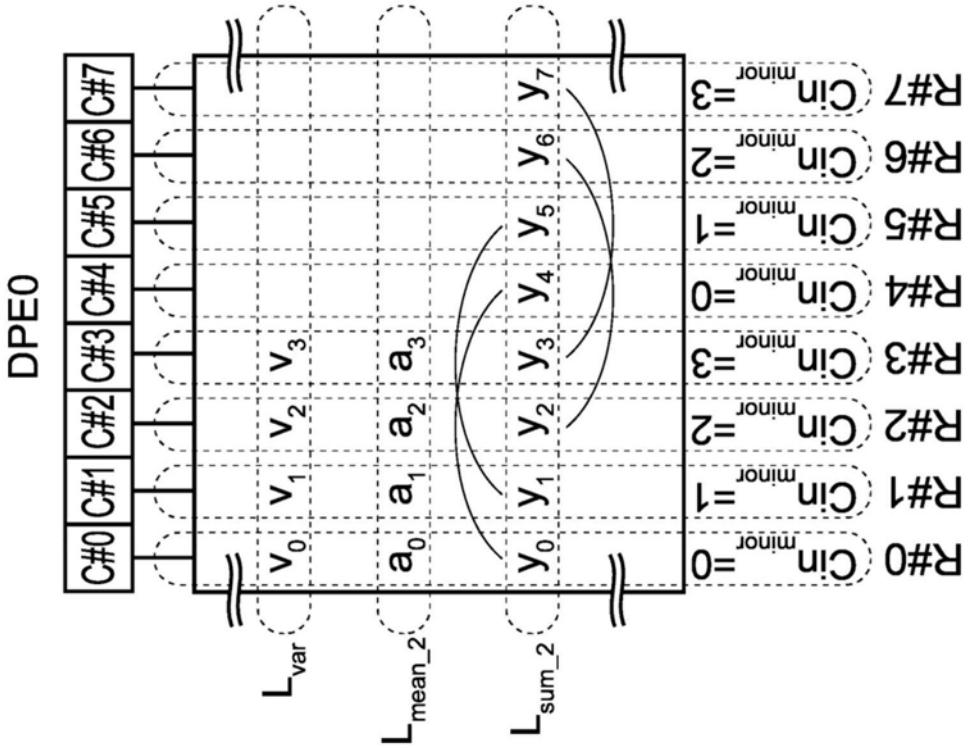

DPE0
C#0
C#1
C#2
C#3
C#4
C#5
C#6
C#7
R#0 Cin minor=0
R#1 Cin minor=1
R#2 Cin minor=2
R#3 Cin minor=3
R#4 Cin minor=0
R#5 Cin minor=1
R#6 Cin minor=2
R#7 Cin minor=3
v0 v1 v2 v3
a0 a1 a2 a3
y0 y1 y2 y3 y4 y5 y6 y7
Lvar
Lmean_2
Lsum_2

图36B