基于快速滤波算法的卷积神经网络加速器设计

王 巍 周凯利* 王伊昌 王 广 袁 军
(重庆邮电大学 光电工程学院/国际半导体学院 重庆 400065)

摘 要：为减少卷积神经网络(CNN)的计算量，该文将2维快速滤波算法引入到卷积神经网络，并提出一种在FPGA上实现CNN逐层加速的硬件架构。首先，采用循环变换方法设计行缓存循环控制单元，用于有效地管理不同卷积窗口以及不同层之间的输入特征图数据，并通过标志信号启动卷积计算加速单元来实现逐层加速；其次，设计了基于4并行快速滤波算法的卷积计算加速单元，该单元采用若干小滤波器组成的复杂度较低的并行滤波结构来实现。利用手写数字集MNIST对所设计的CNN加速器电路进行测试，结果表明：在xilinx kintex7平台上，输入时钟为100 MHz时，电路的计算性能达到了20.49 GOPS，识别率为98.68%。可见通过减少CNN的计算量，能够提高电路的计算性能。

关键词：卷积神经网络；快速滤波算法；FPGA；并行结构
中图分类号：TN432 **文献标识码**：A **文章编号**：1009-5896(2019)11-2578-07
DOI: 10.11999/JEIT190037

Design of Convolutional Neural Networks Accelerator Based on Fast Filter Algorithm

WANG Wei ZHOU Kaili WANG Yichang WANG Guang YUAN Jun
(*College of Electronics Engineering/International Semiconductor College*,
Chongqing University of Posts and Telecommunications, *Chongqing* 400065, *China*)

Abstract: In order to reduce the computational complexity of Convolutional Neural Network(CNN), the two-dimensional fast filtering algorithm is introduced into the CNN, and a hardware architecture for implementing CNN layer-by-layer acceleration on FPGA is proposed. Firstly, the line buffer loop control unit is designed by using the cyclic transformation method to manage effectively different convolution windows and the input feature map data between different layers, and starts the convolution calculation acceleration unit by the flag signal to realize layer-by-layer acceleration. Secondly, a convolution calculation accelerating unit based on 4 parallel fast filtering algorithm is designed. The unit is realized by a less complex parallel filtering structure composed of several small filters. Using the handwritten digit set MNIST to test the designed CNN accelerator circuit, the results show that on the xilinx kintex7 platform, when the input clock is 100 MHz, the computational performance of the circuit reaches 20.49 GOPS, and the recognition rate is 98.68%. It can be seen that the computational performance of the circuit can be improved by reducing the amount of calculation of the CNN.

Key words: Convolution Neural Network(CNN); Fast filter algorithms; FPGA; Parallel structure

1 引言

近年来，卷积神经网络(CNN)已经被广泛应用于图像和语音识别等众多领域[1,2]。其中，CNN涉及的计算常常需要大量的计算资源和时间，使得加速器设计受限于硬件资源，并使深度神经网络的使用受到网络计算速度的限制。面对如此巨大的计算压力，GPU[3], FPGA[4]和ASIC[5]等硬件加速器已被用于加速CNN。在这些加速器中，FPGA凭借其高性能，高能效和可重新编程等优点已成为一种很有前景的解决方案[6–8]。其中，文献[6]表示卷积层的计算量占整个网络计算的90%以上，这表明CNN的计算时间主要由卷积层的计算量决定。因此CNN加速器设计主要是针对卷积层的运算进行的。文献[7]

收稿日期：2019-01-15；改回日期：2019-03-20；网络出版：2019-05-23
*通信作者： 周凯利 2508005354@qq.com
基金项目：国家自然科学基金(61404019)，重庆市集成电路产业重大主题专项(cstc2018jszx-cyztzx0211, cstc2018jszx-cyztzx0217)
Foundation Items: The National Natural Science Foundation of China (61404019), Major Themes of Integrated Circuit Industry in Chongqing (cstc2018jszx-cyztzx0211, cstc2018jszx-cyztzx0217)

提出了并行执行卷积计算用于产生输出特征图的一行值，同时设计相应的输入输出缓存阵列和寄存器阵列以便于计算不间断，从而提高计算速度。文中方案虽然通过提高计算性能(Computation Performance，单位为GOPS: 10^9/s)，达到了计算速度的提升，但其基于传统卷积算法的加速器设计，由于算法效率不高，降低了计算单元的利用率，同时消耗大量硬件资源。因此，当算法本身可以更高效时，便能实现更高的硬件效率。

因此，为了提高算法的效率，目前有些研究开始尝试各种方法来降低卷积层的计算量，以实现较低的卷积神经网络算法复杂度[9,10]。其中，快速Winograd算法[11]也可以对卷积层进行算法强度缩减，使其在具有小滤波器的CNN中发挥作用[12]。不过，在使用Winograd算法时也有一些问题需要注意，首先，设计不仅要最小化内存带宽要求，还要使内存吞吐量与计算引擎相匹配，这增加了设计难度。其次，将Winograd算法映射到FPGA时存在大的设计空间，这使得如果映射的设计不合理时，虽然能节省硬件资源，但是在计算性能(GOPS, 10^9/s)方面并不会有太大提升。所以，本文除了要关注针对小型滤波器的快速算法，还需要找到合适硬件架构设计方案。

为了减少CNN的计算量，本文将2维快速滤波算法[13]引入到卷积神经网络。该算法通过利用相邻卷积窗口之间的结构相似性，消除了2维卷积运算中卷积窗口之间重叠区域计算的冗余，使得算法强度缩减，提高了卷积计算的效率。同时，提出的一种在FPGA上实现CNN的逐层加速硬件架构，可用于有效利用快速滤波算法，从而减少硬件资源。

2　卷积神经网络的快速滤波算法方案

2.1　卷积神经网络

卷积神经网络是一种深度神经网络，它由一系列的层组成，每层又由输入特征图，滤波器和输出特征图组成。在其功能上可分为两部分，第1部分由卷积层和池化层交替组成，用于执行特征提取任务；第2部分由完全连接的层组成，用于分类或识别任务。图1显示了用于图像分类的典型CNN结构。其中卷积层是网络中的主要计算部分，每个卷积层接收输入特征图的数据用于生成输出特征图，这些特征图表示一些特定的特征。图2显示了卷积计算中卷积窗口和卷积核的示例，其中，给定一个预处理的自然图像(也称为输入特征图)，卷积层的结果就是卷积窗口内像素矩阵(图2中的阴影部分)和卷积核(滤波器)矩阵的点积，再经过偏置值和激活函数得到的输出特征图中的元素。其中，卷积计算如式(1)所示。

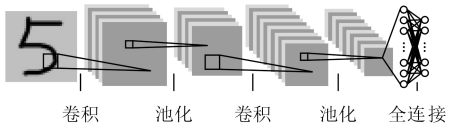


图 1 卷积神经网络的结构

$$h_j^l = f\left(\sum_{i=M_i} h_i^{l-1} * k_{ij}^l + b_j^l\right) \tag{1}$$

式(1)中$h_j{}^l$为第l隐含层的j个输出特征图，$k_{ij}{}^l$为第l层的第j个卷积核的第i个系数，$b_j{}^l$为第l卷积层的第j个输出特征对应的偏置系数，M_j为第l层卷积运算对输入特征图的选择，$f(x)$为激活函数。

2.2　2维快速滤波算法

目前，用于图像识别的CNN网络是以更深层拓扑为趋势的，它具有计算密集和存储密集的特点。这时如果还使用传统的CNN加速器算法，则会降低硬件效率。因此，作为替代方案，使用快速滤波算法来近似卷积层可以更有效地加速CNN。尤其，2维快速滤波算法更适用于CNN的2维矩阵卷积运算。为便于理解，这里将卷积计算式(1)中的卷积符号$*$展开成式(2)。由于这里只演示1张输入特征图的卷积过程，所以，卷积核$k_{ij}{}^l$用$x(i, j)$表示，输出用$Y(m, n)$表示。

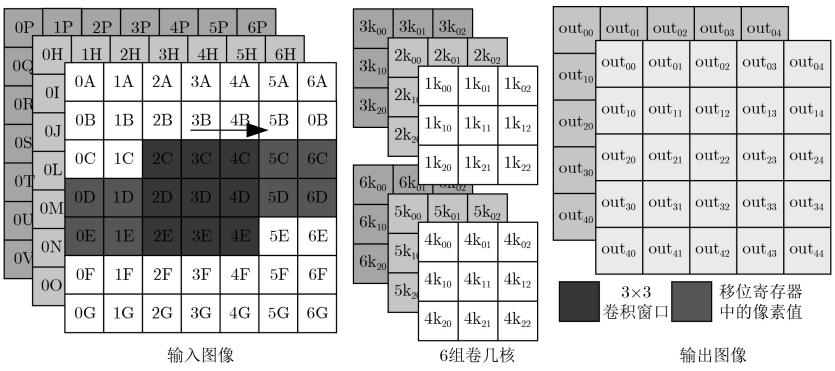


图 2 卷积层的卷积计算过程

$$Y(m,n)=\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}x(i,j)h(i+m,j+n),\quad 0\le m,n\le M-1 \tag{2}$$

其中，N表示卷积核的尺寸；M表示输出特征图的尺寸。当用快速滤波器算法实现2维卷积运算时，以4并行输出的情形为例进行说明。滤波器系数$\boldsymbol{x}(i,j)'$和滤波器输入$\boldsymbol{h}(i,j)'$从i的方向上按间隔2顺序取值如式(3)和式(4)所示

$$\boldsymbol{x}(i,j)'=[x(i,j),x(i+2,j),\cdots,x(i+N-2,j)]^{\mathrm{T}} \tag{3}$$

$$\boldsymbol{h}(i,j)'=[h(i,j),h(i+2,j),\cdots,h(i+N-2,j)]^{\mathrm{T}} \tag{4}$$

然后，再从j的方向上组成$\boldsymbol{X}_{ij}$和$\boldsymbol{H}_{ij}$，矢量$\boldsymbol{X}_{ij}$和$\boldsymbol{H}_{ij}$的长度为$N^2/4$。

$$\boldsymbol{X}_{ij}=\left[\boldsymbol{x}(i,j)',\boldsymbol{x}(i,j+2)',\cdots,\boldsymbol{x}(i,j+N-2)'\right]^{\mathrm{T}} \tag{5}$$

$$\boldsymbol{H}_{ij}=\left[\boldsymbol{h}(i,j)',\boldsymbol{h}(i,j+2)',\cdots,\boldsymbol{h}(i,j+N-2)'\right]^{\mathrm{T}} \tag{6}$$

其中每个元素分别是当前$N\times N$卷积窗口中(i,j)处的卷积核系数和图像像素。因此，对于式(2)的4并行输出：$Y(2m,2n)$, $Y(2m+1,2n)$, $Y(2m,2n+1)$, $Y(2m+1,2n+1)$，可写成式(7)的形式，注意，4并行输出$\boldsymbol{Y}(m,n)$其实是输出特征图中一个2×2的矩阵块。

$$\boldsymbol{Y}(m,n)==\begin{bmatrix}Y(2m,2n)\\Y(2m+1,2n)\\Y(2m,2n+1)\\Y(2m+1,2n+1)\end{bmatrix}=\begin{bmatrix}\boldsymbol{H}_{2m,2n}&\boldsymbol{H}_{2m+1,2n}&\boldsymbol{H}_{2m,2n+1}&\boldsymbol{H}_{2m+1,2n+1}\\\boldsymbol{H}_{2m+1,2n}&\boldsymbol{H}_{2m+2,2n}&\boldsymbol{H}_{2m+1,2n+1}&\boldsymbol{H}_{2m+2,2n+1}\\\boldsymbol{H}_{2m,2n+1}&\boldsymbol{H}_{2m+1,2n+1}&\boldsymbol{H}_{2m,2n+2}&\boldsymbol{H}_{2m+1,2n+2}\\\boldsymbol{H}_{2m+1,2n+1}&\boldsymbol{H}_{2m+2,2n+1}&\boldsymbol{H}_{2m+1,2n+2}&\boldsymbol{H}_{2m+2,2n+2}\end{bmatrix}\begin{bmatrix}\boldsymbol{X}_{00}\\\boldsymbol{X}_{10}\\\boldsymbol{X}_{01}\\\boldsymbol{X}_{11}\end{bmatrix} \tag{7}$$

在式(7)的右侧中，可以通过两次分解操作，变换为式(8)的形式

$$\begin{aligned}\boldsymbol{Y}(m,n)=&\begin{bmatrix}(\boldsymbol{B}_{2m,2n}+\boldsymbol{B}_{2m,2n+1})\boldsymbol{X}_{00}\\(\boldsymbol{B}_{2m+1,2n}+\boldsymbol{B}_{2m+1,2n+1})\boldsymbol{X}_{10}\\(\boldsymbol{B}_{2m,2n+1}+\boldsymbol{B}_{2m+1,2n+2})\boldsymbol{X}_{01}\\(\boldsymbol{B}_{2m+1,2n+1}+\boldsymbol{B}_{2m+1,2n+2})\boldsymbol{X}_{11}\end{bmatrix}+\begin{bmatrix}-1&0\\0&-1\\1&0\\0&1\end{bmatrix}\begin{bmatrix}\boldsymbol{B}_{2m,2n+1}(\boldsymbol{X}_{00}-\boldsymbol{X}_{01})\\\boldsymbol{B}_{2m,2n+1}(\boldsymbol{X}_{10}-\boldsymbol{X}_{11})\end{bmatrix}\\&+\begin{bmatrix}-1&0\\1&0\\0&-1\\0&1\end{bmatrix}\begin{bmatrix}(\boldsymbol{H}_{2m+1,2n}-\boldsymbol{H}_{2m+1,2n+1})\boldsymbol{A}_0\\(\boldsymbol{H}_{2m+1,2n+1}-\boldsymbol{H}_{2m+1,2n+2})\boldsymbol{A}_1\end{bmatrix}+\begin{bmatrix}1\\-1\\-1\\1\end{bmatrix}\boldsymbol{H}_{2m+1,2n+1}(\boldsymbol{A}_0-\boldsymbol{A}_1)\end{aligned} \tag{8}$$

其中，$\boldsymbol{B}_{i,j}$和$\boldsymbol{A}_0$，$\boldsymbol{A}_1$可用下列式(9)和式(10)、式(11)表示

$$\boldsymbol{B}_{i,j}=\boldsymbol{H}_{i,j}+\boldsymbol{H}_{i+1,j} \tag{9}$$

$$\boldsymbol{A}_0=\boldsymbol{X}_{00}-\boldsymbol{X}_{10} \tag{10}$$

$$\boldsymbol{A}_1=\boldsymbol{X}_{01}-\boldsymbol{X}_{11} \tag{11}$$

式(8)中，乘积项$(\boldsymbol{B}_{2m,2n}+\boldsymbol{B}_{2m,2n+1})\boldsymbol{X}_{00}$是2维滤波器形式，$\boldsymbol{X}_{00}$是滤波系数，也就是卷积核系数，阶数为$N/2\times N/2$。同时，式(8)中输出采样运算为原始滤波器运算的1/4。因此，通过应用分解，式(7)的2维卷积被分解为9个计算量为1/16的2维子滤波器和预/后处理运算，得到一个4输出的块。其中，预/后处理运算是通过增加加减法运算数目来减少乘法运算数目的。在预处理中，$N\times N$卷积核被用于计算每个子滤波器的系数，结果用$\boldsymbol{X}_{00}$和$\boldsymbol{A}_0$等表示；而$\boldsymbol{B}_{2m,2n}$和$\boldsymbol{H}_{2m+1,2n}$等表示当前卷积窗口中的像素经过预处理后得到的子滤波器输入。在后处理中，根据每个子滤波器的输出计算最终值$\boldsymbol{Y}(m,n)$。因此，如果忽略这些预/后处理的开销，对于4并行快速滤波器算法，每个输出采样的乘法复杂度可以从N^2减小到$9N^2/16$。这对可编程的FPGA设计来说，不仅可以降低硬件资源的消耗，还可以提升运行速度。此外，4并行快速滤波器算法不仅能够为卷积运算提供更快速的计算能力，而且由于4并行的输出特点与池化层的池化运算所需的池化窗口特点一致，所以在池化运算时，可以省去池化窗口的缓存时间，提高系统的计算速度，并使系统的适用性更加广泛。

3 FPGA的设计实现

3.1 CNN整体架构设计

图3显示了在FPGA上实现CNN的逐层加速硬件架构图。该架构通过设计一系列的功能运算单元来实现卷积神经网络中不同层的计算，并且设计了相应的控制单元来控制何时启动或停止功能运算单元的执行。这种架构有利于使用较少的资源实现CNN中多个层的计算。比如，本文实现了一系列的卷积、池化等运算单元，这些运算单元将被用于逐层加速卷积神经网络的各个层。如图3所示，该架构由若干个子系统构成，各子系统对应卷积神经

网络中的主要运算层，即卷积层，池化层和全连接层。其中，输入数据及卷积核缓存控制单元管理像素数据及卷积核的数据流在架构内各单元之间传递的先后关系以及功能单元之间互联通信的控制、握手信号。因此，在识别开始阶段，当FPGA收到开始的控制信号后，片上缓存开始工作，数据往下一级传输。接着，行缓存循环控制单元开始工作，控制行缓存FIFO单元的读写操作，以便有效地管理不同卷积窗口以及不同运算层之间的输入特征图数据，并通过标志信号启动卷积计算加速单元，以实现逐层加速。在加速过程中，卷积计算加速单元和池化(子采样)层RAM阵列共享于逐层加速的特征提取阶段。最后，在识别阶段，全连接层计算单元执行分类任务，并将数据传输到激活函数单元产生识别结果。全连接层权值缓存控制单元会输出结束标志信号，以便下次识别任务的开始。

3.2 行缓存循环控制单元设计

在行缓存循环控制单元的设计中，本文首先采用循环变换方法对多级循环嵌套的卷积计算进行循环分块，然后利用循环展开实现分块的循环结构到局部并行结构的设计。这里只对2个输入输出特征图维度进行循环分块。如图4所示，行缓存循环控制单元通过控制FIFO的读写使能，实现同一层或不同层中特征数据的循环加载，图4中的h1和h2分别代表来自卷积层C1和卷积层C2的输入数据所需行缓存组。同时，该单元还会给出卷积计算的标志信号，用于控制卷积计算加速单元的启动。在卷积计算中，卷积窗口之间有重叠，这使得计算过程存在大量的数据重用。由于快速滤波算法能够消除2维卷积运算中卷积窗口之间重叠区域计算的冗余，所以对于4并行快速滤波算法来说，其可以很方便地利用卷积窗口在水平和垂直方向上存在的数据重用。首先，为了重用4并行输出运算在垂直方向上的数据，在FIFO阵列上存储了N+1行数据。同时，在水平方向上，经过N+1列数据后就计算出一个2×2的数据块。图4中，每个输入行缓存有M个元素，M代表输入特征图的尺寸；而并行的FIFO阵列个数等于输入通道的数量，也就是需要的输入特征图数量。但是，不同层具有不同的特征图尺寸和通道，因此便将M设置为所有特征图的最大尺寸，h1和h2分别设置为不同卷积层对应的通道数。同时，卷积窗口的滑动步长为S，因此卷积计算加速单元将跳过S行FIFO数据，而跳过的S行数据将被来自输入特征图的接下来的行数据覆盖。

3.3 卷积计算加速单元设计

在卷积神经网路的前向过程中，像素数据需要层层递进式地传播和处理。这说明前后层之间的运

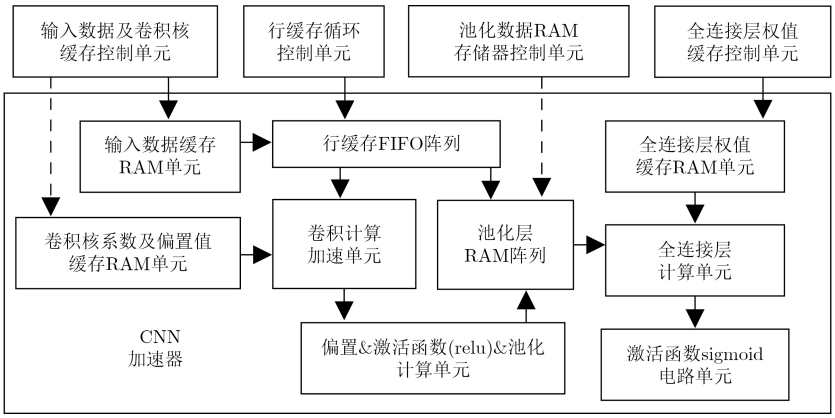


图 3 卷积神经网络的逐层加速硬件架构图

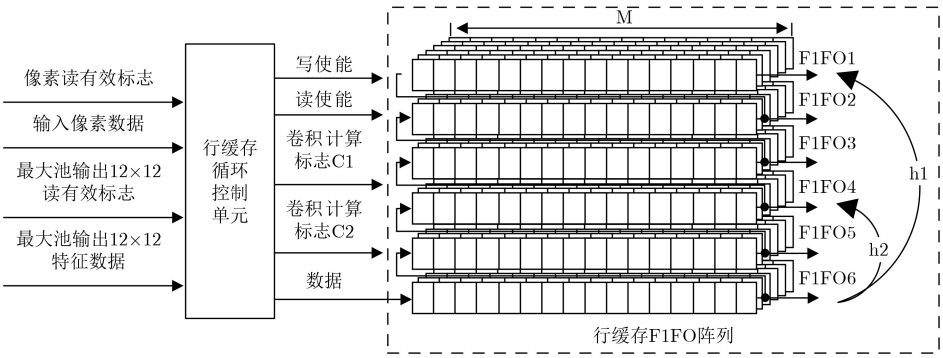


图 4 行缓存循环控制单元

算会相互依赖，层间计算的并行性很低。但是1个卷积层中的多个特征提取过程是在每一层独立完成的，并行性很高。为了减少延迟，根据卷积层中多个特征提取的独立性设计了单输入多输出和多输入单输出的卷积层结构，它与图4中的并行行缓存阵列一一对应。进一步地，特征提取过程就是卷积运算过程，它是指从图像的左上角开始，搜索一个与卷积核大小相同的活动窗口，窗口中图像像素与卷积核系数对应起来相乘再相加，其计算结果是输出特征图中的输出神经元值。以此类推，从左到右、从上到下，即可得到1幅输出特征图像。了解卷积运算在图像滤波中的具体过程，能有助于发现图像滤波中的并行性，得出输出特征图中的每个神经元值可以并行处理。在硬件实现中，如果采用直接结构设计每个神经元的并行结构，会降低硬件效率，并会由于计算量过大而消耗更多资源。因此，可以利用2.2节介绍的4并行快速滤波算法设计卷积计算加速单元，其采用若干子滤波器组成复杂度降低的并行滤波结构来实现等效的滤波效果，结构如图5所示。

在图5中，卷积计算加速单元包括：预处理模块、并行滤波部分和后处理模块，这与算法过程对应。其中，预处理模块的输入端与像素信号及卷积核信号相连接，其主要功能是将量化后的像素和卷积核信号排列成前置加法矩阵中的数据形式，然后经过加减法运算实现预处理，并将输出作为并行滤波部分的输入，部分预处理的逻辑结构如图6(a)所示。

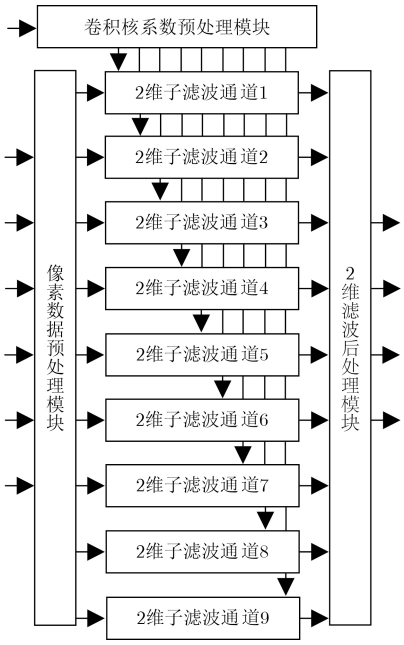
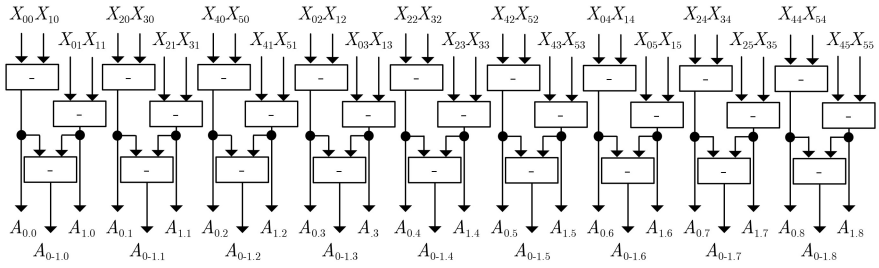


图 5 卷积计算加速单元的结构图

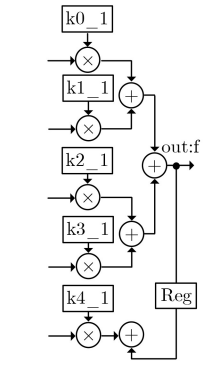
示。并行滤波部分是由9个2维子滤波通道构成的，每一个滤波通道均为$N^2/4$阶的滤波器，其中子滤波通道的基本电路如图6(b)所示。并行滤波部分的输出还需要经过后处理模块将其转化成后置加法矩阵中的数据形式，然后经过加减法运算将输出作为卷积结果，部分后处理的逻辑结构如图6(c)所示。

4 结果分析与讨论

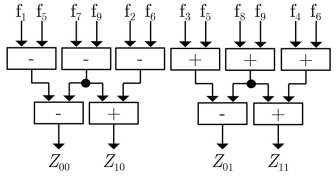
本节使用Xilinx Kintex-7平台进行了CNN硬件



(a) 部分预处理的逻辑结果



(b) 维子滤波通道的基本电路



(c) 部分后处理的逻辑结构

图 6 各部分的具体电路

架构中各个功能单元的硬件实现，并对得到的数据进行分析，以讨论所使用方法的有效性。表1是FPGA设计与PC上的MATLAB设计(使用Intel i5-3210M CPU, 7.88G RAM和64位Windows7)进行比较的情况，主要是为了对比整体的CNN算法在FPGA和MATLAB上实现的精度误差。表1中，测试了10000帧手写数字图像，其中MATLAB的设计采用双精度浮点数表示。而在FPGA的设计上，通过文献[14,15]将得到的参数用定点数表示，以保证网络占用的存储空间最小。同时，由于量化宽度会影响CNN的测试时间，因此在保证精度的前提下，结合测试阶段测试时间的降低情况，确定出了使用16 bit的定点数进行数据量化。从表1中可以看出，在FPGA上实现时，由于数据量化导致了精度从1.19%变化到1.32%，这说明在采用定点数量化数据时，会出现有限字长的量化效应，从而在量化后难以保证原设计精度而产生误差。但是就最终的识别率和测试时间来看，设计能够满足需求。

本文设计的硬件架构在FPGA平台上的工作频率为100 MHz，识别每张图片的时间为0.0199 ms，具体得到的结果与文献[4,6,7]结果显示如表2。表2中，本文的计算性能(GOPS, 10^9/s)得到了明显的提高，这主要是因为通过将快速滤波算法引入到CNN，使得算法强度缩减，从而显著地提高了卷积计算的效率。文献[7]由于采用片外存储器节省了BRAM的使用，然而本文使用了复杂度降低的并行滤波结构以及逐层加速的逻辑架构节省了主要的DSP资源。同时与文献[4,6]相比，在计算性能GOPS上得到明显提升，不过在主要的硬件资源消耗上相对较多。综上所述，本文的设计方案在计算性能GOPS指标上的表现优于表2中其它加速器方案。因此，在一些对计算性能有严格要求的应用，该方案可以提供更好地计算性能。

表 1　MATLAB实现与FPGA实现的比较

类型	时间(ms/frams)	精度(bad/10000 frames)	数据类型
.m文件	0.7854	1.19%	双精度
.v 文件	0.01986	1.32%	16 bit定点数

表 2　卷积神经网络FPGA实现的性能比较

参数	文献[4]	文献[6]	文献[7]	本文方案
FPGA	Virttex-7xc7vx485t	Zynqzc702	Virttex-7xc7vx485t	Kirtex-7xc7k325t
频率(MHz)	100	166	150	100
时间(ms)	2.6368	0.1510	0.0254	0.0199
BRAM	27	96	0	30
DSP	20	95	638	284
FF	54075	27664	66346	36973
LUT	14832	38836	51125	51748
识别率(%)	98.62	99.01	96.80	98.68
GOPS	1.58	2.70	15.87	20.49

5　结束语

本文提出了一种基于快速滤波算法的卷积神经网络加速器设计。其中，快速滤波算法可以显著地提高2维卷积的计算效率，因为它能利用相邻卷积窗口之间的结构相似性，消除重叠区域计算的冗余，使得算法强度缩减。同时，为了节省硬件资源，提出了一种在FPGA上实现的逐层加速的CNN硬件架构方案。该方案通过采用循环变换方法设计了行缓存循环控制单元来管理不同卷积窗口之间以及不同运算层之间的特征图数据，并利用标志信号逐层启动卷积计算加速单元。其中，卷积计算加速单元是基于4并行快速滤波算法实现的，它采用若干子滤波器组成复杂度降低的并行滤波结构来实现等效的滤波效果。上述设计可以显著提高电路的整体计算性能。利用手写数字集MNIST对设计的CNN加速器电路进行测试实验，实验结果表明：输入时钟为100 MHz时，整体电路在计算性能上由达到了20.49 GOPS，识别率达到了98.68%。可见，通过减少CNN的计算量，能够提高电路的计算性能。

参 考 文 献

[1] ZHANG Chen, LI Peng, SUN Guangyu, *et al*. Optimizing FPGA-based accelerator design for deep convolutional neural networks[C]. 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, USA, 2015: 161–170.

[2] KRIZHEVSKY A, SUTSKEVER I, and HINTON G E. ImageNet classification with deep convolutional neural networks[C]. The 25th International Conference on Neural Information Processing Systems, Lake Tahoe, USA, 2012:

1097–1105.

[3] DONG Han, LI Tao, LENG Jiabing, *et al*. GCN: GPU-based cube CNN framework for hyperspectral image classification[C]. The 201746th International Conference on Parallel Processing, Bristol, UK, 2017: 41–49.

[4] GHAFFARI S and SHARIFIAN S. FPGA-based convolutional neural network accelerator design using high level synthesize[C]. The 20162nd International Conference of Signal Processing and Intelligent Systems, Tehran, Iran, 2016: 1–6.

[5] CHEN Y H, KRISHNA T, EMER J S, *et al*. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks[J]. *IEEE Journal of Solid-State Circuits*, 2017, 52(1): 127–138. doi: 10.1109/JSSC.2016.2616357.

[6] FENG Gan, HU Zuyi, CHEN Song, *et al*. Energy-efficient and high-throughput FPGA-based accelerator for Convolutional Neural Networks[C]. The 201613th IEEE International Conference on Solid-State and Integrated Circuit Technology, Hangzhou, China, 2016: 624–626.

[7] ZHOU Yongmei and JIANG Jingfei. An FPGA-based accelerator implementation for deep convolutional neural networks[C]. The 20154th International Conference on Computer Science and Network Technology, Harbin, China, 2015: 829–832.

[8] HOSEINI F, SHAHBAHRAMI A, and BAYAT P. An efficient implementation of deep convolutional neural networks for MRI segmentation[J]. *Journal of Digital Imaging*, 2018, 31(5): 738–747. doi: 10.1007/s10278-018-0062-2.

[9] HUANG Jiahao, WANG Tiejun, ZHU Xuhui, *et al*. A parallel optimization of the fast algorithm of convolution neural network on CPU[C]. The 201810th International Conference on Measuring Technology and Mechatronics Automation, Changsha, China, 2018: 5–9.

[10] LAVIN A and GRAY S. Fast algorithms for convolutional neural networks[C]. 2016 IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, USA, 2016, 4013–4021.

[11] VINCHURKAR P P, RATHKANTHIWAR S V, and KAKDE S M. HDL implementation of DFT architectures using winograd fast Fourier transform algorithm[C]. The 2015 5th International Conference on Communication Systems and Network Technologies, Gwalior, India, 2015: 397–401.

[12] WANG Xuan, WANG Chao, and ZHOU Xuehai. Work-in-progress: WinoNN: Optimising FPGA-based neural network accelerators using fast winograd algorithm[C]. 2018 International Conference on Hardware/Software Codesign and System Synthesis, Turin, Italy, 2018: 1–2.

[13] NAITO Y, MIYAZAKI T, and KURODA I. A fast full-search motion estimation method for programmable processors with a multiply-accumulator[C]. 1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings, Atlanta, USA, 1996: 3221–3224.

[14] JIANG Jingfei, HU Rongdong, and LUJÁN M. A flexible memory controller supporting deep belief networks with fixed-point arithmetic[C]. The 2013 IEEE 27th International Symposium on Parallel and Distributed Processing Workshops and PhD Forum, Cambridge, USA, 2013: 144–152.

[15] LI Sicheng, WEN Wei, WANG Yu, *et al*. An FPGA design framework for CNN sparsification and acceleration[C]. The 2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines, Napa, USA, 2017: 28.

王 巍：男，1967年生，博士后，教授，研究方向为集成电路设计.

周凯利：女，1991年生，硕士生，研究方向为数字集成电路设计.

王伊昌：男，1996年生，硕士生，研究方向为模拟集成电路设计.

王 广：男，1994年生，硕士生，研究方向为半导体光电器件设计.

袁 军：男，1984年生，博士，副教授，研究方向为数模混合集成电路设计.