DOI:10.11918/201812159

一种高度并行的卷积神经网络加速器设计方法

徐 欣[1],刘 强[1],王少军[2]

(1.天津市成像与感知微电子技术重点实验室(天津大学),天津 300072;
2.哈尔滨工业大学 电子与信息工程学院,哈尔滨 150001)

摘 要:为实现卷积神经网络数据的高度并行传输与计算,生成高效的硬件加速器设计方案,提出了一种基于数据对齐并行处理、多卷积核并行计算的硬件架构设计和探索方法.该方法首先根据输入图像尺寸对数据进行对齐预处理,实现数据层面的高度并行传输与计算,以提高加速器的数据传输和计算速度,并适应多种尺寸的输入图像;采用多卷积核并行计算方法,使不同的卷积核可同时对输入图片进行卷积,以实现卷积核层面的并行计算;基于该方法建立硬件资源与性能的数学模型,通过数值求解,获得性能与资源协同优化的高效卷积神经网络硬件架构方案.实验结果表明:所提出的方法,在 Xilinx Zynq XC7Z045 上实现的基于16位定点数的 SSD 网络(single shot multibox detector network)模型在 175 MHz 的时钟频率下,吞吐量可以达到 44.59 帧/s,整板功耗为 9.72 W,能效为 31.54 GOP/(s·W);与实现同一网络的中央处理器(CPU)和图形处理器(GPU)相比,功耗分别降低85.1%与93.9%;与现有的其他卷积神经网络硬件加速器设计相比,能效提升20%~60%,更适用于低功耗嵌入式应用场合.

关键词:现场可编程门阵列;卷积神经网络;并行处理;硬件结构优化;SSD 网络
中图分类号:TP391.4 **文献标志码**:A **文章编号**:0367-6234(2020)04-0031-07

A highly parallel design method for convolutional neural networks accelerator

XU Xin[1], LIU Qiang[1], WANG Shaojun[2]

(1. Key Laboratory of Imaging and Sensing Microelectronic Technology (Tianjin University), Tianjin 300072, China;
2. School of Electronic and Information Engineering, Harbin Institute of Technology, Harbin 150001, China)

Abstract: To achieve highly parallel data transmission and computation of convolutional neural network acceleration and generate efficient hardware accelerator design, a hardware design and exploration method based on data-alignment and multi-filter parallel computing was proposed. In order to improve the data transmission and computation speed and adapt to various input image sizes, the method first aligned the data according to the input image size to achieve highly parallel transmission and computation at the data level. The method also used the multi-filter parallel computing method so that different filters can simultaneously convolve the input image to achieve parallel computing at the filters level. Based on this method, mathematical models of hardware resources and performance were formulated and numerically solved to obtain the performance and resource co-optimized neural network hardware architecture. The proposed design method was applied to the single shot multibox detector (SSD) network, and results show that the accelerator on Xilinx Zynq XC7Z045 at 175 MHz clock frequency could achieve the throughput of 44.59 FPS, power consumption of 9.72 W, and power efficiency of 31.54 GOP/(s·W). The accelerator consumed 85.1% and 93.9% less power than the central processing unit (CPU) and graphics processing unit (GPU) implementations respectively. Compared with the exiting designs, the power efficiency of the proposed design increased 20% ~ 60%. Therefore, the design method is more suitable for embedded applications with low power requirements.

Keywords: field programmable gate array (FPGA); convolutional neural network; parallelism; structure optimization; single shot multibox detector (SSD) network

卷积神经网络近年来在目标检测[1]、图像修复[2]、自然语言处理、机器视觉等应用领域获得了巨大突破.面对这些复杂任务,神经网络系统需要强大的数据存储以及数据处理能力.高速定制化的硬件处理器为处理能力的提高提供了一个可行的途径[3-6].相比于其他硬件处理器,FPGA 拥有良好的性能、较短的开发周期、更高的能源利用率及重配置功能,因此越来越多的开发者选择基于 FPGA 设计卷积神经网络硬件加速器[3,7-8].

收稿日期:2018-12-25
基金项目:国家自然科学基金(61574099);天津市交通运输委科技发展基金(2017b-40)
作者简介:徐 欣(1994—),女,硕士研究生;
刘 强(1979—),男,副教授,博士生导师
通信作者:刘 强,qiangliu@tju.edu

目前,基于 FPGA 的卷积神经网络硬件加速器仍面临诸多挑战. 1)卷积硬件加速器具有较高的并行计算特性,因此数据传输速度往往成为性能继续提升的瓶颈,例如 Qiu 等[9]设计提出的采用多个卷积处理单元并行计算的方法,其较低的片上数据传输速度就限制了整体系统的计算性能;2)硬件设计空间巨大,不同的硬件并行设计方法将影响加速器性能与资源使用率,例如端到端的设计[10-11]的方法把所有卷积层的计算引擎都放置到片上. 这种方法对片上资源和带宽要求较高,并且只适用于较小网络模型. 如何设计加速器的高度并行传输与计算架构以及如何在有限的硬件资源下选择出最佳并行方案以实现加速器的高性能是目前亟需解决的问题.

针对上述问题,本文首先提出了数据对齐并行处理的方法以实现数据层面的高速并行传输与计算,采用多卷积核并行处理的方法实现同一卷积层的多个卷积核并行计算,并通过建立性能与资源评估模型,生成性能与资源协同优化的并行方案. 最终在 Xilinx Zynq XC7Z045 上的 SSD 网络模型测试结果表明,该加速器在 175 MHz 时钟下可以达到 44.59 FPS与 306.6 GOP·s^{-1}.

1　卷积神经网络

卷积神经网络是一种具有共享权重结构、平移不变性特征的人工神经网络,其主要由卷积层、下采样层和全连接层组成. 卷积层的作用是通过权值共享来提取特征,下采样层则主要是为了降低数据维度与防止图像过拟合,全连接层的作用是把最后一层卷积的输出图像由高维变成低维,并且把输入的信息进行提取整合,再经过激活函数的映射,以实现特征到标签的映射.

卷积层从上层局部邻域特征图中提取特征,然后施加一个加性偏差,其结果如下:

$$Q_i(x,y) = b_i + \sum_{j=0}^{M_i-1}\sum_{p=0}^{P_i-1}\sum_{q=0}^{Q_i-1} W_{ij}(p,q) \times g_j(x+p,y+q). \tag{1}$$

式中:$Q_i(x,y)$为第 i 张输出图像上的(x,y)处的输出特征;b_i为第 i 张输出特征的偏差;$g_j(x+p,y+q)$为第 j 张输入图像在$(x+p,y+q)$处特征值;$W_{ij}(p,q)$为卷积核(p,q)处的权重值;M_i为输入图像的通道数;P_i、Q_i分别为卷积核长宽.

除此之外,卷积层的输出还要经过激活函数和批量归一化处理. 激活函数可以把特征保留并映射出来,以增强卷积神经网络系统的非线性,常用的激活函数有 tanh, sigmoid, ReLU, PReLU 等. 本文选择下式所示 ReLU 函数[12]作为非线性激活函数.

$$\mathrm{ReLU}(x) = \mathrm{Max}(x,0),$$

式中 x 为输入像素.

经过卷积层权值共享和局部连接后,图像权值参数已经大大减少,但是对于全连接层来说仍有着计算量方面的挑战,并且还会出现过拟合现象. 为了解决这些问题,可按着一定的方法对特征图进行下采样操作,提升整个网络模型的性能.

全连接层是将特征图进行整合和分类的高层,以得到整个图像的信息. 该层的每一个神经元都与上一层的所有神经元相连,简单来说全连接层就是矩阵向量乘法过程,输入向量与权重矩阵相乘,以实现特征的分类.

为了兼顾目标检测的时效性和不变性,SSD 算法在 2016 年被 Liu 等[13]提出. SSD 算法的网络模型是一个基于 VGG16 结构的卷积神经网络,其核心是利用小型卷积滤波器来预测目标类别与边界框,并达到速度与精确度的平衡. 如图 1 所示为一个基本的 SSD 网络架构.

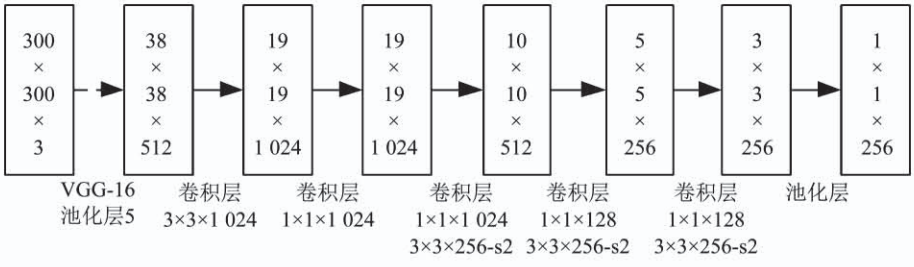

图1　基本 SSD 网络模型

Fig. 1　Basic SSD network model

2　硬件实现及优化

本文所提出的系统整体架构如图 2 所示,主要包含:片外双倍速率同步动态随机存储器(double data rate synchronous dynamic random access memory, DDR SDRAM)、ARM(advanced risc machine)处理器、直接内存存取单元(direct memory access, DMA)、片上缓冲器、控制器以及卷积计算单元. 片外 DDR 将储存外设获取的输入图像数据以及网络的权值系数;ARM 处理器对整体系统进行控制;卷

积计算单元主要完成卷积神经网络的卷积计算，还包括池化、激活等操作；片上缓冲器用于缓存计算所需图像数据，权值系数以及计算结果；DMA 用于片外储存器与片上缓冲器之间的数据传输；控制器从外部获取指令并对其进行解码，以控制可编程逻辑端的所有模块.

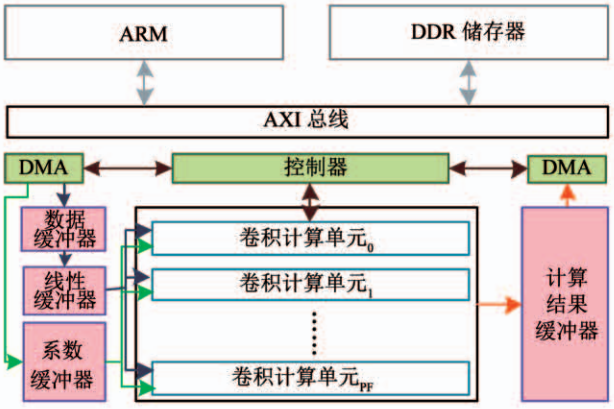

图 2　系统整体硬件架构图

Fig. 2　An overview of the accelerator architecture

2.1　数据对齐并行处理

本文首先提出了一种数据对齐并行预处理的方法实现数据层面的高度并行传输与计算，以提高数据传输、并行计算速度和硬件设计的通用性. 文中用 PD 表示数据传输的并行度，即一个时钟周期内并行传输的数据个数. PV 表示数据计算的并行度，为一个卷积核在图像上可同时计算的结果个数，K 为卷积核大小. 预处理单元由两部分组成：移位寄存器与先进先出（first input first output，FIFO）线性缓冲器. 根据式（1），卷积运算要求数据按照行对齐输出，所以首先要对图像数据进行行缓存. 由于一个时钟内数据传输的个数为 PD，当图像边长不能被数据传输并行度 PD 整除，需要对输入图像添加无效数据以使边长为 PD 的整数倍，保证图像的每行像素以每个时钟周期传输 PD 个数据的方式经过 N 个时钟周期传输完毕. 这样，图像数据就可以以行对齐的方式存储在线性 FIFO 中，不会产生行错位. 其次，行对齐输出要求对线性缓冲器 FIFO 进行同时读取，使不同行的相同位置的数据一起输出（如图 3 中的 X_{p0}，X_{p6}，X_{p8}同时输出）. 最后将读取到的图像数据组合成卷积矩阵，输入到计算模块中与系数进行卷积运算.

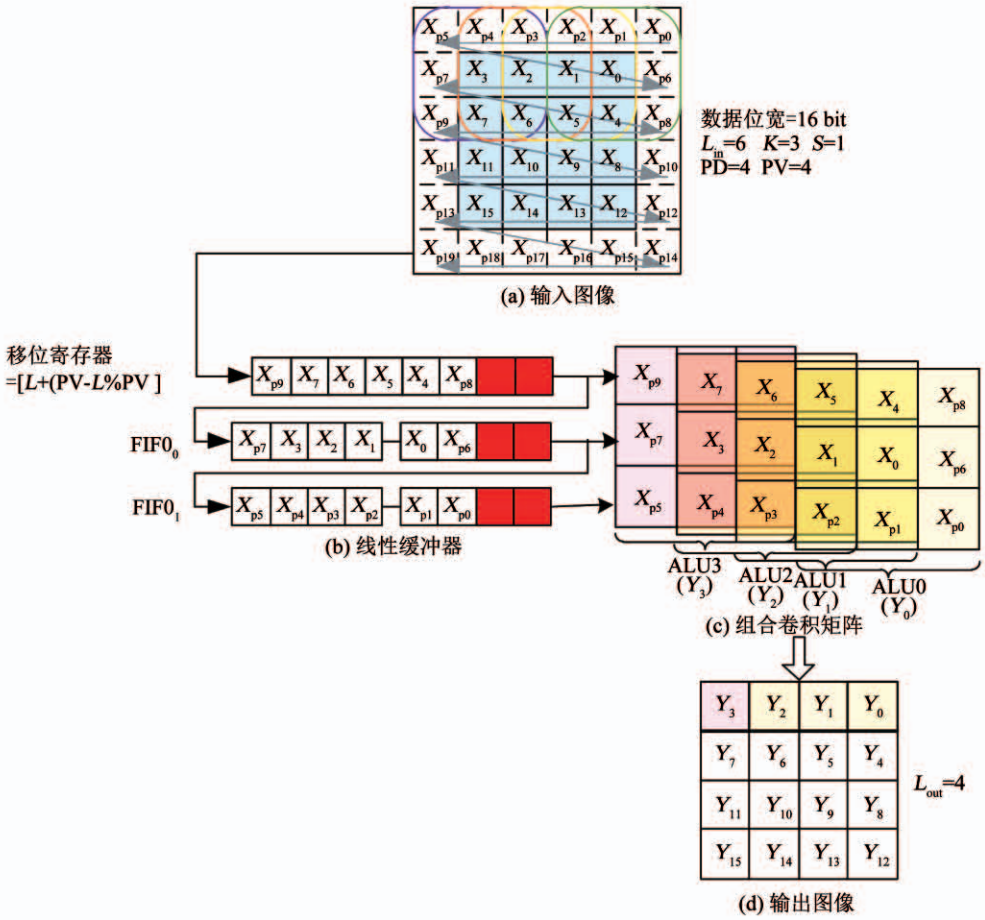

图 3　输入数据对齐并行处理

Fig. 3　Data alignment parallel processing

图 3 所示为一个 3 ×3 大小的卷积核在 6 ×6 输入图像上滑动的过程. 其中蓝色表示原始图像大小为 4 ×4. 由于应用需要,原始图像进行了像素填充,因此实际输入图像变成 6 ×6(这里填充像素层数设置为 1). 输入图像数据将按行依次读取. 数据并行传输个数 PD 设置为 4,数据计算并行度 PV 为 4,卷积核尺寸 K 为 3,滑动步长为 1. 由于此时图像边长 6 不为数据传输并行度 4 的整数倍,需要在每行的左侧添加 2 个无效数据(图中的红色像素)存入寄存器中补充成 8 个数据,以保证每行数据正好可以经过 2 个时钟周期传完. 该例中实现的 3 ×3 的卷积核计算需要对 3 行的数据进行行对齐读取以组成卷积矩阵. 因此使用 2 个 FIFO 缓存前两行输入数据,在第 3 个周期时便可与移位寄存器中的第 3 行数据实现行对齐输出,其结构如图 3(b)所示. 由于数据传输并行度为 PD,一个时钟周期便可得到 3 × PD 个数据. 根据式(1)可知当卷积核滑动步长为 1 时,每个卷积操作的输入矩阵之间有重叠,按照图 3(c)中的方式进行组合可得到 PV 个卷积矩阵. 然后将这 PV 个矩阵按所标顺序分别输入到 PV 个 $K \times K$ 卷积核计算单元的中与系数进行卷积运算,这样一个周期内就可以得到 PV 个卷积结果. 边长为 6 的图像,经过卷积操作后输出图像边长为 4. 如图 3(d)所示,输出图像的一行像素可经过一次数据并行计算同时得到.

综上所述,PD、PV 的选择主要受到输出图像尺寸的限制. 本文要实现一个完整的网络模型,因此硬件加速器的 PD 与 PV 的大小最终将需要考虑网络的所有卷积层的尺寸来确定.

2.2 多卷积核并行计算

经过对卷积层的分析发现,网络的每层卷积层都是用不同数量的卷积核在同一个输入图像上不断滑动进行卷积. 因此,可以在硬件上实现 PF 个卷积核运算单元,使 PF 个不同的卷积核同时与图像数据进行卷积运算. 图 4 是一个多卷积核并行计算的硬件结构. 其中数据预处理单元表示数据对齐并行处理的移位寄存器与 FIFO 线性缓冲器组成的预处理单元. 权值系数缓冲器缓存计算所需的权值系数. KCU 为一个 $K \times K$ 卷积核计算单元. 这里假设卷积核的并行度 PF 为 3,即可同时进行 3 个不同卷积核的卷积运算. 依上文所述,每一层卷积层的所有卷积核处理的为同一张图像,所以其输入图像数据是同时读取的,只是被映射到不同的卷积核的卷积单元. 假设输入图像为 RGB 三通道图像,图像将依照通道顺序依次进入预处理单元. 一个通道的图像经过数据预处理单元处理后得到的 PV 个卷积矩阵将会被同时输送到每个卷积核的 CCU 卷积单元,然后再对应输入到 CCU 的 PV 个 $K \times K$ 卷积核计算单元当中与卷积系数进行乘累加操作. 由于有 PF 个卷积核,每个卷积核的 CCU 单元可同时进行一个卷积核的 PV 个卷积结果的计算,因此一个周期内可以得到 PF × PV 个卷积运算结果. 每个卷积核的卷积单元计算完输入图像的一个通道后,把结果暂时存放在 FIFO 当中,等待下一个通道计算结果的到来再进行

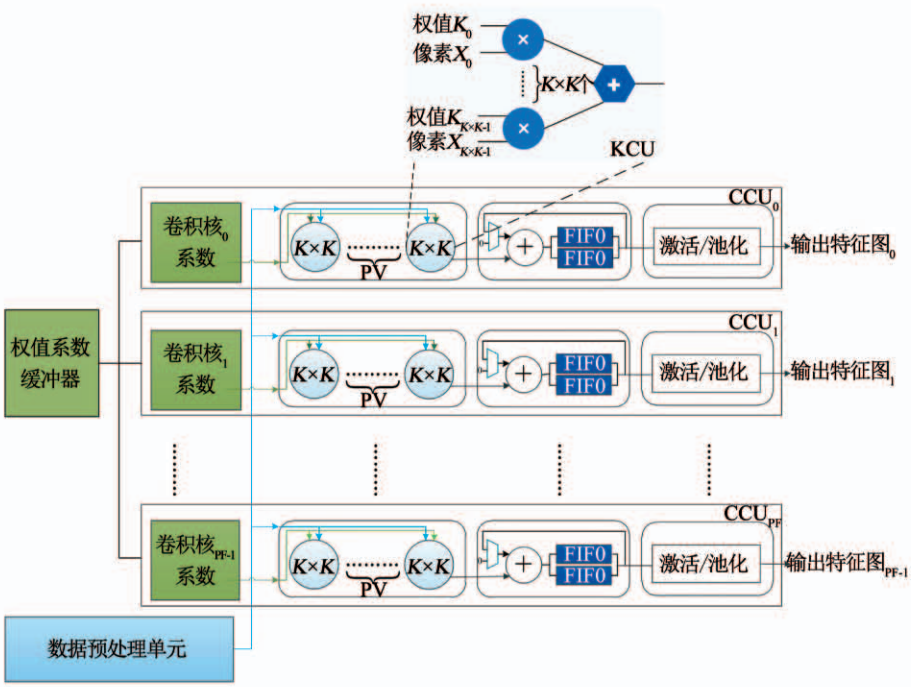

图 4 多卷积核并行计算单元

Fig. 4 Parallel computation unit with multiple filters

累加. 最终，输入图像的所有通道计算完后，存放在 FIFO 中的结果依次输出. 多卷积核并行计算单元的计算速度快于暂存结果的 FIFO 的数据输出速度. 为了减少计算等待时间，使上一周期的 PF 个卷积核的结果计算完后，无需等待 FIFO 中的数据全部取出，即可开始下一周期 PF 个卷积计算，在最后的通道累加单元中，使用了乒乓结构，使两个 FIFO 交替存取并行计算单元的卷积结果.

2.3　硬件架构分析

根据多卷积核并行计算，加速器的 PV 大小将由输出图像确定，而卷积核并行度 PF 则主要受到片上资源的限制. 卷积模块中的乘法全部利用 FPGA 的数字信号处理资源(digital signal processing，DSP)实现. 采用图 4 所示多卷积核并行策略后，所用的乘法器总数应该小于总的 DSP 资源为

$$\mathrm{PV}\times\mathrm{PF}\times\mathrm{Kernel}_k\leqslant\mathrm{DSP}_{\mathrm{sum}}. \tag{2}$$

式中：Kernel_k为计算一个 $K\times K$ 大小卷积所需乘法器个数；$\mathrm{DSP}_{\mathrm{sum}}$为 FPGA 片上 DSP 资源总数.

除此之外，还需要使用随机存取存储器块(block random access memory，BRAM)存储卷积计算所需的临时数据，因此需要考虑片上的存储资源. 片上存储主要包括：输入缓冲区、线性缓冲区以及临时计算结果存储区. 每层卷积层的输出图像大小公式如下：

$$H_{\mathrm{out}}^i=\frac{H_{\mathrm{in}}^i-K}{s}+1, \tag{3}$$

$$W_{\mathrm{out}}^i=\frac{W_{\mathrm{in}}^i-K}{s}+1. \tag{4}$$

式中：H_{in}^i、W_{in}^i分别为第 i 层输入图像的长和宽；K 为卷积核大小；s 为卷积滑动步长小；H_{out}^i、W_{out}^i分别为第 i 层输出图像的长宽.

输入缓冲区需要存放下网络的最大输入图像与系数，所以大小为 $\max[(H_{\mathrm{in}}^i\times W_{\mathrm{in}}^i+K\times K\times\mathrm{PF})\times C_{\mathrm{in}}^i]$. 由于输出缓冲区也需要储存最大输出图像并且采用了乒乓结构，所以总共大小为 $\max(H_{\mathrm{out}}^i\times W_{\mathrm{out}}^i)\times\mathrm{PF}\times 2$. 线性缓冲区大小为 $\max(W_{\mathrm{in}}^i)\times(K-1)$. 所以，总的存储资源需要满足：

$$\begin{aligned}&\max[(H_{\mathrm{in}}^i\times W_{\mathrm{in}}^i+K\times K\times\mathrm{PF})\times C_{\mathrm{in}}^i]+\\&\max(W_{\mathrm{in}}^i)\times(K-1)+\max(H_{\mathrm{out}}^i\times W_{\mathrm{out}}^i)\times\\&\mathrm{PF}\times 2\leqslant\mathrm{BRAM}_{\mathrm{sum}}.\end{aligned} \tag{5}$$

式中：$\mathrm{BRAM}_{\mathrm{sum}}$为总的片上 RAM 资源；$K$ 为卷积核大小；C_{in}^i为第 i 层输入图像的通道个数.

除资源限制之外，为了保证数据并行传输速度与计算速度匹配，PD 应尽量大于等于 PV；同时为了减少再对输出结果的处理，PV 应尽量能被输出图像边长整除，即：

$$W_{\mathrm{out}}^i\bmod\mathrm{PV}=0, \tag{6}$$

$$\mathrm{PD}\geqslant\mathrm{PV}, \tag{7}$$

式中，W_{out}^i为第 i 层的输出图像边长.

图 5 为卷积层执行时间示意图，可以看到一次的硬件执行时间 T_{total}主要由计算所需图像的输入时间 T_{data}，PF 个卷积核的权值系数输入时间 $T_{\mathrm{coef}}^{\mathrm{PF}}$，计算时间 $T_{\mathrm{compute}}^{\mathrm{total}}$以及最终 PF 个计算结果输出时间 $T_{\mathrm{output}}^{\mathrm{PF}}$组成. 其中：

$$T_{\mathrm{coef}}^{\mathrm{PF}}=K\times K\times\mathrm{PF}\times C_{\mathrm{in}}\times t_{\mathrm{clk}}, \tag{8}$$

$$T_{\mathrm{data}}=H_{\mathrm{in}}\times W_{\mathrm{in}}\times C_{\mathrm{in}}\times\delta\times t_{\mathrm{clk}}, \tag{9}$$

$$T_{\mathrm{output}}^{\mathrm{PF}}=\frac{H_{\mathrm{out}}\times W_{\mathrm{out}}\times\mathrm{PF}}{\mathrm{PV}}\times t_{\mathrm{clk}}, \tag{10}$$

$$T_{\mathrm{compute}}^{\mathrm{total}}=\frac{H_{\mathrm{out}}\times W_{\mathrm{out}}\times C_{\mathrm{in}}\times F_{\mathrm{out}}}{\mathrm{PV}\times\mathrm{PF}}\times\theta\times t_{\mathrm{clk}}, \tag{11}$$

$$T_{\mathrm{total}}=T_{\mathrm{data}}+T_{\mathrm{coef}}^{\mathrm{PF}}+T_{\mathrm{compute}}^{\mathrm{total}}+T_{\mathrm{output}}^{\mathrm{PF}}. \tag{12}$$

式中：K 为卷积核大小；W_{in}、H_{in}分别为该次卷积计算的输入图像尺寸；W_{out}、H_{out}分别为输出图像尺寸；C_{in}为输入图像通道数；F_{out}为卷积核个数；δ、θ 分别为架构所决定的传输与计算效率；t_{clk}为时钟周期.

网络硬件执行时间为所有卷积层的执行时间总和. 每层卷积层的执行时间则为该层所有的分割块的执行时间总和，即

$$\mathrm{Time}_{\mathrm{all}}=\sum_{i=1}^{M}(T_{\mathrm{total}}^i\times N_{\mathrm{block}}^i). \tag{13}$$

式中：$\mathrm{Time}_{\mathrm{all}}$为网络的总共硬件执行时间；$T_{\mathrm{total}}^i$为式(12)所表示的第 i 层的一个分割块的硬件运算时间；N_{block}^i为第 i 层的分割块数.

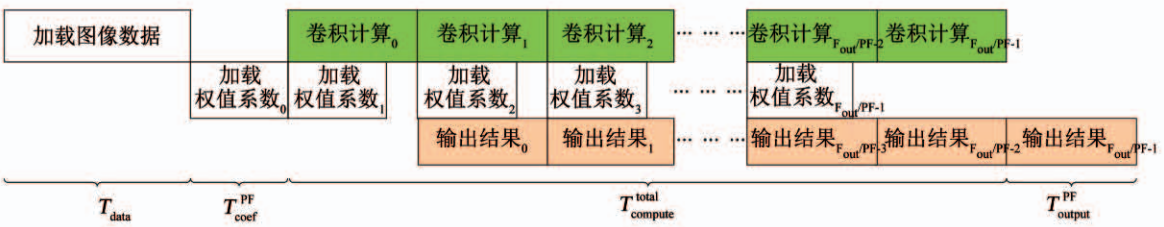

图 5　卷积层硬件执行时间

Fig. 5　Execution time of each convolution layer

3 结 果

本文将根据上述硬件架构分析对设计架构进行评估,以确定出最佳的并行方案.

表1为本文所实现的SSD网络模型所包含的卷积层,加速器需要将每层卷积层的图像数据与权值系数从片外DDR读取到片上进行临时储存.当图像数据和系数的数量较为庞大时,由于片上资源有限,无法将完整一层的数据和参数全部储存到片上存储器中.这时,需要对图像进行分割,以分块的方式进行处理.表1中列出了对于较大的卷积层所需进行的分割尺寸.这里相同输入图像大小的卷积层只列举了一个.为了降低加速器的资源与功耗,本文采用16位定点实现SSD网络模型,相比于单浮点数,其mAP仅下降了0.8%.

表1 SSD网络模型卷积层

Tab.1 Convolutional layer of SSD network model

层数	输入图像	通道数	分割尺寸	输出尺寸	分割块数	卷积核
第1层	258×258	3	66×66	64×64	16	3×3
第3层	130×130	32	34×34	32×32	16	3×3
第5层	66×66	48	34×34	32×32	4	3×3
第11层	34×34	48		32×32	1	3×3
第14层	34×34	96	18×18	32×32	4	3×3
第17层	18×18	128		16×16	1	3×3
第20层	10×10	128		8×8	1	3×3
第22层	8×8	128		8×8	1	1×1

图6是根据硬件架构分析式(2)~(13)所绘制的资源与性能评估模型图.在充分利用实验所用Xilinx Zynq XC7Z045片上DSP资源的情况下,理论执行时间与存储资源都与PV成反比.由于PV需满足式(6)的限制,因此根据表1中的输出尺寸分别对PV=2、4、8、16、32这5种可选方案进行取点.可以看出当PV小于8时,硬件执行时间明显增加,并且需要的存储资源大于实际片上存储资源.网络中最小的输出图像大小边长为8,因此最终根据式(2)~(7)选择PD=8、PV=8、PF=12的并行方案.对该方案进行实际综合,DSP、BRAM、查找表单元(look-up-table, LUT)以及触发器(flip-flop, FF)的使用率见表2.

表3列出了将同一SSD网络分别在CPU、GPU、FPGA32和FPGA16上实现的结果对比.从表3中可以看出,本文基于16位定点数实现的SSD网络模型在175 MHz的时钟频率下,可以达到44.59帧/s,比基于32位定点数的硬件设计快4.99倍,分别是CPU与GPU的1.39倍与0.41倍.动态功耗为3.72 W,相比CPU与GPU分别降低85.1%与93.9%,是FPGA32的1.04倍.综上所述,本文所提出的16位卷积神经网络加速器与CPU、GPU相比,具有较低的功耗,与32位的加速器相比,具有较高性能的优势,因此更适用于移动设备上.

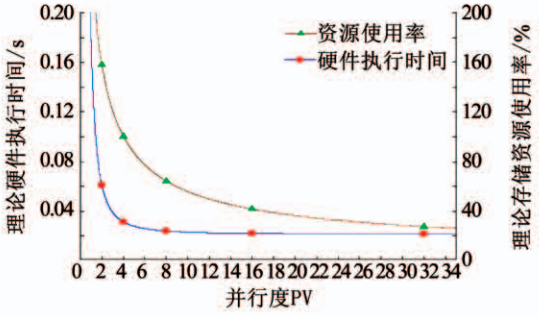

图6 性能与资源评估

Fig.6 Evaluation of performance and resource

表2 硬件实现资源使用情况

Tab.2 Resource utilization

资源类别	可用总量	总使用量	使用率/%
FF	437 200	104 891	23.99
BRAM	545	319	58.53
DSP	900	864	96.00
LUT	218 600	78 910	36.10

表4列出了3种基于Xilinx Zynq XC7Z045实现的卷积神经网络加速器[9,14-15].可以看出,相比于其他设计,本设计采用数据对齐并行处理、多卷积核并行计算等方法能够充分利用片上资源,实现高度并行计算,使加速器的能效达到31.54 GOP/(s·W),比文献[9,14-15]有较大性能提升.

表3 SSD网络硬件实验结果比较

Tab.3 Comparison of SSD implementations

对比类别	CPU	GPU	FPGA32	FPGA16
平台	Intel Xeon E5-2630	NVIDIA Tesla K40c	Zynq XC7Z045	Zynq XC7Z045
存储空间/GB	8	12	$1_{(DDR)}$ $0.018\,7_{(on\text{-}chip)}$	$1_{(DDR)}$ $0.018\,7_{(on\text{-}chip)}$
精度	32位浮点	32位浮点	32位定点	16位定点
频率/MHz	2 200	745	150	175
帧/s	32.05	109.05	8.92	44.59
帧数对比	0.72x	2.44x	0.20x	1.00x
动态功率/W	24.95	61.96	3.56	3.72
动态功率对比	6.71x	16.65x	0.95x	1.00x

表 4　与已有文献实验结果比较

Tab. 4　Comparison between several existing implementations and the proposed design

对比文献	精度	频率/MHz	性能/(GOP·s^{-1})	功率/W	能效/(GOP·(s·W^{-1}))	能效对比
文献[9]	16 位定点	150	136.97	9.63	14.22	0.45x
文献[14]	16 位定点	150	187.80	9.63	19.50	0.62x
文献[15]	16 位定点	100	229.50	9.40	24.42	0.77x
本文所提设计	16 位定点	175	306.60	9.72	31.54	1.00x

4　结　论

1)本文所提出的卷积神经网络加速器采用数据对齐并行处理与多卷积核并行计算的方法，实现了数据的高速并行传输，与多个层面的并行计算，并根据该设计方法建立了资源与性能模型，找出最佳并行方案，实现了加速器整体硬件架构的高能效.

2)实验结果表明，加速器在 175 MHz 时钟频率下实现的 SSD 网络模型，其速度可以达到44.59 FPS，功耗为 9.72 W，整体能效达到 31.54 GOP/(s·W).

参考文献

[1]岳颀，马彩文. 指数弹性动量卷积神经网络及其在行人检测中的应用[J]. 哈尔滨工业大学学报，2017，49(5)：159
YUE Qi，MA Caiwen. A deep convolution neural network for object detection based[J]. Journal of Harbin Institute of Technology，2017，49(5)：159. DOI:10.11918/j.issn.0367－6234.201603145

[2]郭继昌，郭昊，郭春乐. 多尺度卷积神经网络的单幅图像去雨方法[J]. 哈尔滨工业大学学报，2018，50(3)：185
GUO Jichang，GUO Hao，GUO Chunle. Single image rain removal based on multi-scale convolutional neural network[J]. Journal of Harbin Institute of Technology，2018，50(3)：185. DOI：10.11918/j.issn.0367－6234.20170407

[3] KRIZHEVSKY A，SUTSKEVER I，HINTON G E. Imagenet classification with deep convolutional neural networks[C]// Proceedings of the 25th International Conference on Neural Information Processing Systems. Lake Tahoe，USA：ACM，2012：1097. DOI:10.1145/3065386

[4] CHAKRADHAR S，SANKARADAS M，JAKKULA V，et al. A dynamically configurable coprocessor for convolutional neural networks[C]//Proceedings of the 37th Annual International Symposium on Computer Architecture，ser. ISCA'10. New York，NY：ACM，2010：247. DOI:10.1145/1815961.1815993

[5] DU Zidong，FASTHUBER R，CHEN Tianshi，et al. ShiDianNao：Shifting vision processing closer to the sensor[C]//Proceedings of the 42nd Annual International Symposium on Computer Architecture. Portland，OR，USA：IEEE，2015，43(3)：92. DOI:10.1145/2749469.2750389

[6] CHEN Tianshi，DU Zidong，SUN Ninghui，et al. Diannao：A small-footprint high-throughput accelerator for ubiquitous machine-learning[C]//Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems. Salt Lake City，USA：ACM，2014，49(4)：269. DOI:10.1145/2541940.2541967

[7] SANKARADAS M，JAKKULA V，CADAMBI S，et al. A massively parallel coprocessor for convolutional neural networks[C]// Proceedings of the 20th IEEE International Conference on Application-Specific Systems，Architectures and Processors. Boston，USA：IEEE，2009：53. DOI:10.1109/ASAP.2009.25

[8] SUNG Wonyong，PARK J. Architecture exploration of a programmable neural network processor for embedded systems[C]//Proceedings of International Conference on Embedded Computer Systems：Architectures，Modeling and Simulation (SAMOS). Agios Konstantinos，Greece：IEEE，2016：124. DOI:0.1109/SAMOS.2016.7818339

[9] QIU Jiantao，WANG Jie，YAO Song，et al. Going deeper with embedded FPGA platform for convolutional neural network[C]// Proceedings of 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. Monterey，California：ACM，2016：26. DOI:10.1145/2847263.2847265

[10] LI Himin，FAN Xitian，JIAO Li，et al. A high performance FPGA-based accelerator for large-scale convolutional neural networks[C]// Proceedings of the 26th International Conference on Field Programmable Logic and Applications. Lausanne，Switzerland：IEEE，2016：1. DOI:10.1109/FPL.2016.7577308

[11] HUANG Chao，NI Siyu，CHEN Gengsheng. A layer-based structured design of CNN on FPGA[C]//Proceedings of the 12th International Conference on ASIC. Guiyang，China：IEEE，2017：1037. DOI:10.1109/ASICON.2017.8252656

[12] GLOROT X，BORDES A，BENGIO Y. Deep sparse rectifier neural networks[C]//Proceedings of the 14th International Conference on Artificial Intelligence and Statistics. Fort Lauderdale，FL，USA：IEEE，2011：315

[13] LIU Wei，ANGUELOV D，ERHAN D，et al. SSD：Single shot multibox detector[C]//Proceedings of Computer Vision-European Conference on Computer Vision. Cham：Springer，2016：21. DOI:10.1007/978－3－319－46448－0_2

[14] GUO Kaiyuan，SUI Lingzhi，QIU Jiantao，et al. Angel-eye：A complete design flow for mapping CNN onto embedded FPGA[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems，2018，37(1)：35. DOI:10.1109/TCAD.2017.2705069

[15] XIAO Qingchen，LIANG Yun，LU Liqiang，et al. Exploring heterogeneous algorithms for accelerating deep convolutional neural networks on FPGAs[C]//Proceedings of the 54th Annual Design Automation Conference. Austin，TX：IEEE，2017：1. DOI:10.1145/3061639.3062244

(编辑　张　红)