DOI:10.20079/j.issn.1001-893x.220819003

动态深度神经网络的硬件加速设计及 FPGA 实现*

王 鹏[a,c],任轶群[a,b],范毓洋[a,c],张嘉诚[a]

(中国民航大学 a.民航航空器适航审定技术重点实验室;b.电子信息与自动化学院;c.安全科学与工程学院,天津 300300)

摘 要:基于现场可编程门阵列(Field Programmable Gate Array,FPGA)实现的卷积神经网络由于具有优秀的目标识别能力,广泛应用在边缘设备。然而现有的神经网络部署多基于静态模型,因此存在无效特征提取、计算量增大、帧率降低等问题。为此,提出了动态深度神经网络的实现方法。通过引入模型定点压缩技术和并行的卷积分块方法,并结合低延迟的数据调度策略,实现了高效卷积计算。同时对神经网络动态退出机制中引入的交叉熵损失函数,提出便于硬件实现的简化方法,设计专用的加速电路。根据所提方法,在 Xilinx xc7z030 平台部署了具有动态深度的 ResNet110 网络,平台最高可完成 2.78×10^4 MOPS(Million Operations per Second)的乘积累加运算,并支持 1.25 MOPS 的自然指数运算和 0.125 MOPS 的对数运算,相较于 i7-5960x 处理器加速比达到 287%,相较于 NVIDIA TITAN X 处理器加速比达到 145%。

关键词:边缘设备;动态深度神经网络;动态退出机制;硬件加速;加速电路

开放科学(资源服务)标识码(OSID):

中图分类号:TP302;TN79 **文献标志码**:A **文章编号**:1001-893X(2024)03-0358-08

Design and FPGA Implementation of Dynamic Deep Neural Network Hardware Acceleration

WANG Peng[a,c],REN Yiqun[a,b],FAN Yuyang[a,c],ZHANG Jiacheng[a]

(a. Key Laboratory of Civil Aircraft Airworthiness Technology; b. College of Electronic Information and Automation; c. College of Safety Science and Engineering, Civil Aviation University of China, Tianjin 300300, China)

Abstract: Convolutional neural network (CNN) based on field programmable gate array (FPGA) has been widely used in edge devices because of their excellent object detection capability. However, existing neural network deployment is mostly based on static models, resulting in invalid feature extraction, computation increase, and frame rate reduction. In order to solve these problems, the authors propose a deployment method for a dynamic depth neural network. By introducing model fixed-point compression technology and parallel convolution blocking method, combined with low latency data scheduling strategy, efficient convolution computation is achieved. At the same time, a simplified hardware implementation method is proposed for the cross-entropy loss function used in the neural network dynamic exit mechanism, and a dedicated acceleration circuit is designed. According to the proposed method, dynamic depth ResNet110 is deployed on the Xilinx xc7z030 platform, which reaches 2.78×10^4 million operations per second (MOPS) Multiply and Accumulate (MAC), 1.25 MOPS natural index operation, and 0.125 MOPS logarithmic operation. The acceleration ratio is 287% compared with that of i7-5960x and 145% compared with that of NVIDIA TITAN X.

Key words: edge device; dynamic deep neural network; dynamic exit mechanism; hardware acceleration; accelerating circuit

0 引 言

卷积神经网络(Convolutional Neural Network,CNN)为基础的计算机视觉算法,在汽车自动驾驶、无人机图像识别、人体行为识别等领域有着广泛的

* **收稿日期**:2022-08-19;**修回日期**:2022-10-19
基金项目:国家重点研发计划(2021YFB1600600);中央高校基本科研业务费(XJ2021003601)
通信作者:任轶群 Email:renyq970808@163.com

应用[1-3],典型的卷积神经网络有 AlexNet、VGGNet、GoogLeNet 和 ResNet 等[4]。为了使 CNN 网络获得更好的精度,模型引入了更深层次的网络架构,导致正向推理过程所需要的计算复杂度增加,边缘设备难以高帧率地运行 CNN 网络[5]。实际应用中图像的特征识别难度不同,一些特征明显的图像只需要较少层数的网络即可准确识别[6]。为此,动态深度神经网络[7-9]通过在 CNN 中插入侧分枝(BranchyNet)分类器,允许交叉熵损失满足一定置信度的简单图像提前退出,减少不必要的特征提取次数,提升 CNN 模型在边缘部署的计算效率。BranchyNet 的加入虽然减少了识别部分图像的计算量,但在边缘平台部署动态深度神经网络仍面临计算量大、算法复杂度高等问题。

现场可编程门阵列(Field Programmable Gate Array,FPGA)有着强大的并行性,不但可以数据并行还可以数据流并行,能够满足卷积神经网络的大量并行计算需求,并且具有低延时和低功耗的特性,还可以通过硬件适应算法的特性[10],设计针对特殊算法的加速电路。因此,FPGA 可以很好地契合动态深度神经网络在边缘部署的需求。在 FPGA 加速神经网络方面,Venieris 等人[11]采用同步数据流架构(Synchronous Dataflow Model)设计了一种将 CNN 模型计算任务映射到 FPGA 平台的流水处理方法,为了避免因数据缺失而导致流水线停顿,将需要加速的 CNN 模型权重和各层之间的特征图全部缓存到片上,但因为片上内存受限,此类方法只能适用于规模较小的 CNN 模型。Li 等人[12]提出一种多 FPGA 组成 FPG-A 集群的架构,虽然可以达到较高的吞吐率以及很高的计算能力,但由于需要多个 FPGA 平台协作完成,不适用于边缘计算的应用场景。Liu 等人[13]使用 Winograd 算法提出了一个在 FPGA 上运行的 Winograd 算法与脉动阵列结合的架构,需要专门的访存子系统,设计较为复杂,并且 Winograd 算法只能适用于步长为 1 的卷积,因此通用性受限。

本文基于 FPGA 设计了一种面向嵌入式平台的针对插入侧分支后具有动态深度的 ResNet110 网络(BranchyNet-ResNet110,B-ResNet110),并从定点量化、并行运算、流水线化等 3 个方面对模型进行加速,针对提前退出机制设计专用的加速电路。在 XILINX xc7z030ffg676-2 芯片上对上述内容进行实现,首先设计针对卷积层和池化层的通用 IP,以及提前退出模块的专用加速 IP;然后将 IP 组成片上系统(System On Chip,SOC),完成 B-ResNet110 部署;最后分别从计算速度、资源利用率、识别准确率等方面对设计进行综合性能分析。

1　动态深度神经网络

本设计中的动态网络是带有提前退出分支 BranchyNet 的神经网络。图 1 给出了 B-ResNet110 的结构,可以看出该动态网络由主干网络和两个分支网络组成。其中,主干网络是 ResNet110 网络,可分为 5 个部分。第一部分是一个 16 通道卷积核尺寸 3×3 步长(stride)为 1 的卷积层。第二部分至第四部分,由 18 个常规残差模块(plain residual block)组成,通道数分别为 16 通道,32 通道,64 通道。不同的是,第三部分和第四部分的第一个残差模块需要对输入特征图进行下采样操作。第五部分为池化层和全连接层组成的输出层。第一个分支网络(BranchyNet 1)插入在第二部分之后,第二个分支网络(BranchyNet 2)插入在第三部分之后。分支网络 1 由一个残差模块、一个全连接层、一个交叉熵损失函数层组成。分支网络 2 由一个 3×3 卷积层、一个全连接层、一个交叉熵损失函数层组成。交叉熵损失函数如公式(1)所示:

$$\mathrm{entropy}(y) = -\sum_{c\in C} y_c \ln \hat{y}_c \tag{1}$$

式中:C 表示所有标签类的集合;y_c 表示每个类的标签值;$\hat{y}_c$ 表示 Softmax 函数,

$$\hat{y}_c = \mathrm{softmax}(z) = \frac{\exp(z)}{\sum_{c\in C}\exp(z_c)} \tag{2}$$

式中:z_c 是该出口点上全连接层的输出值。

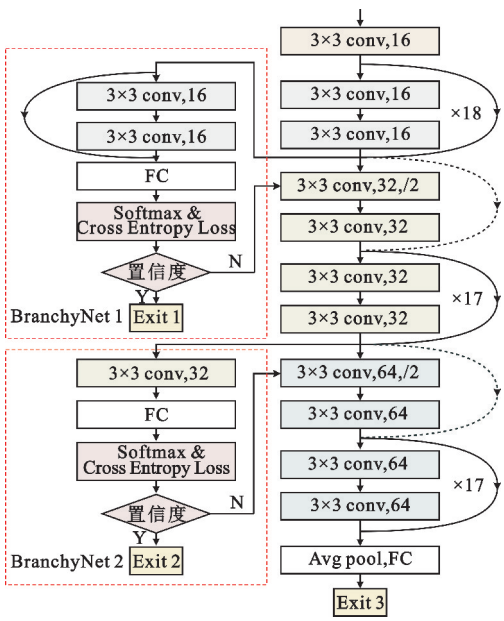

图 1　B-ResNet110 网络结构

Fig. 1 Network architecture of B-ResNet110

在每一个分支的末端都有一个出口点(Exit),使用分类结果的交叉熵损失作为预测置信度的度量。如果分支 1 分类结果的交叉熵损失满足置信度要求,认为图像经过分支 1 可以进行识别,则图像在 Exit 1 预测结果后退出网络,不被主干网第三和第四部分处理;反之,则认为该出口点的分类结果不确定,样本继续回到网络中并在分支 2 的 Exit 2 进行预测。若 Exit 2 也不满足预测要求,图像将经过全部网络在 Exit 3 处进行预测。B-ResNet110 架构可以根据输入图像的特征明显程度自适应选择主干网的层数,从而避免将所有图像逐层处理,显著减少了计算量以及正向推理时间。

B-ResNet 的网络结构中包含的卷积核尺为 3×3 和 1×1,其中 3×3 的卷积核步长是 1 和 2,1×1 卷积核的步长为 2。池化层中池化单元的尺寸为 2×2。主干网和分支网络都包含全连接层,每个分支中都包含交叉熵损失函数。对 B-ResNet110 而言,需要设计的硬件加速单元有通用卷积计算单元、池化计算单元、全连接层计算单元、残差结构的卷积输出加法单元,以及分支网络中提前退出机制计算单元。其中,提前退出机制包含的交叉熵损失函数较为复杂,需要根据推理阶段的实际需求对其函数进行简化。

2 硬件加速设计

2.1 定点量化设计

在 FPGA 中,完成不同类型的数据运算所消耗的资源和时间是不同的。表 1 分别列举了 FPGA 中 16 b 定点数和浮点数完成加法、减法、乘法、指数运算、对数运算所需要的 DSP 处理单元的个数,以及完成上述运算所需要的时钟周期。从数据中可以看出,在 FPGA 中定点数的运算效率以及资源利用率远大于浮点数。

表 1 FPGA 浮点数和定点数运算时间以及所耗资源对比

Tab. 1 Comparison of FPGA floating-point, fixed-point operation time and resource consumption

Type	DSP		Period	
	fixed<16>	float	fixed<16>	float
+	1	2	1	4
−	0	2	1	4
*	1	3	3	3
/	0	0	28	11
exp	2	7	4	7
ln	3	13	8	12

因此,为了使 FPGA 在运行神经网络时的具有更高的效率,可根据深度神经网络包含足够多的冗余信息,并且裁剪这些冗余信息不会明显导致网络准确率下降的特性,对深度神经网络权重和偏执中浮点数据类型进行 16 b 定点数量化,量化完成后的定点数格式如图 2 所示。

图 2 16 b 定点数格式

Fig. 2 Format of 16 b fixed-point number

定点数首位是 1 b 的符号表示位,符号位后是(15−fl)位的整数部分,最后是 fl 位的小数部分。因此,量化后的 16 b 定点数可由公式(3)表示:

$$V_{\text{fixed16}} = \sum_{i=0}^{15} B_i \cdot 2^{-fl} \cdot 2^i, B_i \in \{0,1\} \tag{3}$$

2.2 卷积计算单元设计

虽然 FPGA 可以将卷积并行运算,但是卷积神经网络的计算量较大,将卷积层的计算在 FPGA 中完全展开十分困难。以 B-ResNet110 为例,其中主干网络的卷积层中包含的乘积累加运算(Multiply Accumulate,MAC)操作总数超过 1. 3 亿次,单层网络所需最少 MAC 运算也有 2.36×10^6 次之多,这远超过适用于边缘部署的 FPGA 内部 DSP 处理单元的个数。因此,卷积神经网络部署在 FPGA 当中时,需要将卷积神经网络的卷积运算拆分成更小的循环,才能在 FPGA 的计算核心中执行。卷积计算循环分块如图 3 所示,卷积计算输入为 N 个的尺寸 $H\times L$ 的特征图,卷积核为 M 个 $N\times K\times K$ 的三维卷积核,输出特征图的尺寸为 $M\times R\times C$。将输入特征图和输出特征图按通道数、长、宽划分成 $Tn\times Th\times Tl$ 和 $Tm\times Tr\times Tc$ 的块,对应的卷积核被划分为 $Tn\times Tm\times K\times K$ 块。其中,Th 和 Tl 与 Tr 和 Tc 的对应关系为 $Th=(Tr-1)S+K$,$Th=(Tc-1)S+K$,S 和 K 分别为卷积核的步长和尺寸。经过分块之后,卷积计算会从原来的 4 层循环转换成图 3 中的 8 层循环。这 8 层循环中黑色方框外部的 4 层循环描述的是分块数据调度方法,内部的 4 层表示的是 FPGA 计算核心中进行的分块卷积计算。

```
for(r=0;r<R;r+=Tr)                                   //Loop R.
  for(c=0;c<C;c+=Tc)                                 //Loop C.
    for(m=0;m<M;m+=Tm)                               //Loop M.
      for(n=0;n<N;n+=Tn)                             //Loop N.
      /*Computation in the core.*/
        for(tr=r;tr<min(r+Tr,R);tr ++)               //Loop Tr.
          for(tc=c;tc<min(c+Tc,C);tc ++)             //Loop Tc.
            /*Unroll*/
            for(tm=m;tm<min(m+Tm,M);tm ++)           //Loop Tm.
              for(tn=n;tn<min(n+Tn,M);tn ++)         //Loop Tn.
                o[tm][tr][tc]+=                      //Convolution.
                ∑_{i=0}^{K-1} ∑_{j=0}^{K-1} W[tm][tn][i][j]*I[tn][tr*S+i][tc*S+j];
```

图 3 卷积循环分块

Fig. 3 Convolutional loop partitioning

分块之后特征图与卷积核之间的卷积运算本质上来说仍是一个三维的 MAC 操作。每一个输入的特征图都有与之相对应的卷积核,各个特征图只与其对应的多个卷积核进行卷积运算,不同通道特征图对应卷积核之间不存在依赖关系。同通道卷积核共享输入的特征图,但这些卷积核的计算结果之间不进行累加操作,同通道的不同卷积核之间无依赖关系。因此,在计算处理单元(Processing Element,PE)阵列设计时,可以将图 3 红色框中的两层无依赖关系的循环并行运算,提高计算效率。

卷积计算单元的硬件设计如图 4 所示,由 AXI Lite 总线传入控制信息,AXI 总线传输权重数据和特征图。卷积计算单元内部由计算模块、数据调度模块以及片上缓存组成。计算模块中的 PE 计算对象是输入通道分块 Tn 展开(Unroll)后的卷积运算,并行度为 n。m 个 PE 组成 PE 阵列,对输入通道不同卷积核并行展开运算,并行度为 m。由图 3 外部循环中的循环 M 可以看出,每计算出一个 Tm 都需要循环 N 完整循环一次。因此,数据调度模块需要对外部数据反复进行读取,并且复用输出数据。为了减少数据传输带来的延迟,采用粗粒度流水线,即双缓存乒乓操作,对数据进行传输。这种方法可以在一组缓存中的数据参与计算的过程中,并行加载下一组输入数据以及缓存下一组输出数据。该方法既可以保证卷积计算对缓存数据乱序访问时不受输入数据影响,又可以减少数据传输的时间,从而减少访存带来的推理延迟。对于偏置权重,在第一次加载数据时与其他数据并行加载到输出缓存,将偏置值与第一次分块卷积输出结果相加。这种偏置处理方法既能代替输出缓存的复位操作从而节约复位时间,又能节省卷积运算完成后再加偏置带来的计算时间。若没有偏置,则加载数据“0”,复位输出缓存。

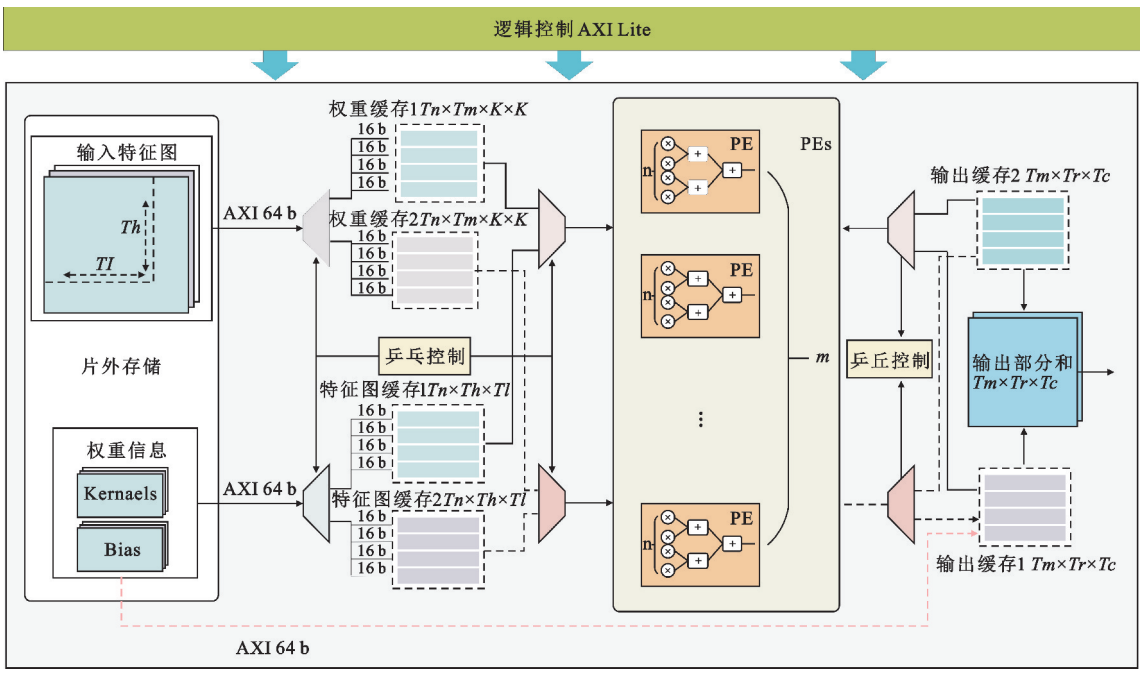

图 4　卷积计算单元

Fig. 4 Convolutional computing unit

2.3　池化计算单元设计

池化单元的设计与卷积设计相比较为简单。池化运算单元不需要加载权重,不同特征图之间的池化计算得出的数据没有依赖关系,计算量小,输入通道数等于输出通道数,输出的部分数据不需要复用,因此池化模块只需将特征图按通道分批加载到片上计算后按序输出即可。平均池化与最大池化都采用 2×2 的滑窗,因此最大池化和平均池化可以复用同一个片上缓存区域从而减少资源的占用。池化单元的设计如图 5 所示。

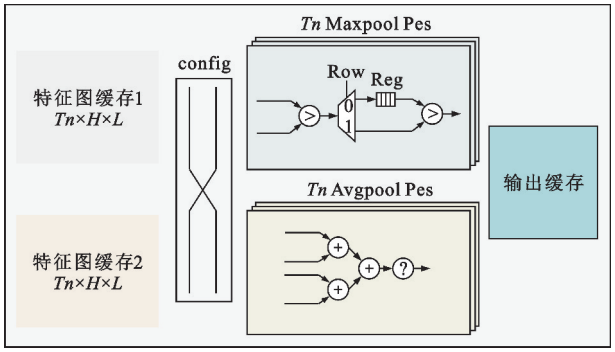

图 5 池化模块
Fig. 5 Pooling module

池化运算单元的数据输入方式与卷积运算单元一样,均由 AXI 总线将划分好的特征图输入并且采用粗粒度流水线减少数据传输带来的延迟。传入的特征图通过配置(config)信号,选择最大池化操作或者平均池化操作。平均操作池化首先将 2×2 的卷积窗的第一行的 2 个数据进行大小比较,存入寄存器,然后将第二行的 2 个数据输入进行比较,最后将两次输出结果中最大值作为最大池化操作的输出;平均池化计算将 4 个数据通过加法树并行相加之后取平均值。池化得出的部分结果存入输出缓存,待整张特征图计算完成后将其结果传输到片外存储。

2.4 全连接层与卷积输出加法设计

全连接层与卷积输出加法层,只涉及到顺序的加法运算和乘法运算,不需要复杂的内存访问,因此对于全连接层与卷积输出加法层,仍采用双缓存片上数据存储方式,以及粗粒度流水线加载方式。输入的数据采用加法树以及乘法器级联加法树的方法并行计算,与卷积层和池化层的设计方法相同,这里不再赘述。

2.5 提前退出模块设计实现

在动态深度神经网络中提前退出模块是核心部分,然而提前退出模块中包含的 Softmax 函数以及交叉熵损失函数等运算通常运用在训练时的反向传播中。该运算所包含的多次除法运算、e 指数运算以及对数运算,在部署过程中会消耗大量的硬件资源并且带来过高延迟。如果用全连接层输出的最大值等指标作为置信度测量指标,虽然能减少计算量,但精度非常低,很容易造成识别结果错误。因此,可以将交叉熵损失判断置信度与全连接层的输出的最大值相结合,改进交叉熵损失函数使其更符合硬件实现的需求。

2.5.1 交叉熵函数的数学变换

公式(1)中的 y_c 代表的标签的值,该值是“0”或“1”,并且由 y_c 对应 Softmax 函数的输出决定,Softmax 函数输出最大值对应的 y_c 的标签为“1”,其他为“0”。由公式(2)可以得出,Softmax 函数的输出值的大小由全连接成层的输出大小决定。因此,结合公式(1)和公式(2),在部署阶段可以用全连接层的输出值代替 Softmax 函数的输出值对标签值进行判断。根据判断结果,对交叉熵损失简化如公式(4)所示,因为标签为“0”的类别对应的 Softmax 其系数为“0”,不需要计算,所以只需要计算标签为“1”的类别对应的 Softmax 值,根据对数函数性质,将除法变成减法。这种变换方法可以减少 $C-1$ 次 Softmax 运算,并且去掉所有的除法运算。该变换将较大提升推理速度。

$$\begin{aligned}\text{entropy}(y) &= -\ln \hat{y}_c = \\ &-\ln\left(\frac{\exp(z)}{\sum_{c\in C}\exp(z_c)}\right) = \\ &\ln \sum_{c\in C}\exp(z_c) - z\end{aligned} \tag{4}$$

当 z 增加时,对 z 取 e 指数运算得出的值可能超出 16 b 定点数所能表示的范围,故 16 b 定点数并不能充分地表示该计算结果,导致计算置信度时有误差。因此,可以通过公式(5)的变换,将 z 取 e 指数运算变成 $z-z_{\max}$ 取 e 指数运算,将 e 指数运算的计算结果约束在(0,1)范围内。

$$\begin{aligned}\text{entropy}(y) &= -\ln\left(\frac{\exp(z-z_{\max})}{\sum_{c\in C}\exp(z_c-z_{\max})}\right) = \\ &\ln\sum_{c\in C}\exp(z_c-z_{\max}) - (z-z_{\max})\end{aligned} \tag{5}$$

由于在部署的阶段,$z_{\max}$ 是最终的预测结果,$z=z_{\max}$,因此交叉熵损失函数还可以进一步转化为公式(6)以减少计算量。

$$\text{entropy}(y) = \ln \sum_{c\in C}\exp(z_c-z_{\max}) \tag{6}$$

2.5.2 提前退出模块硬件实现

如图 6 所示,交叉熵损失函数硬件实现的过程中,充分利用 FPGA 可以进行并行运算的特性,将交叉熵损失函数涉及到的加法运算、比较运算并行展开,然后通过比较树以及加法树等结构将并行计算结果汇聚到一起,完成对交叉熵损失函数的加速计算。交叉熵损失函数的输出与阈值的比较结果作为提前退出的判断条件。

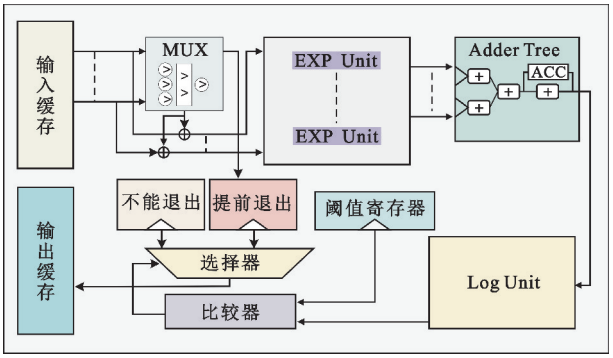

图 6　提前退出模块
Fig. 6 Exit in advance module

3　设计与结果分析

3.1　实验环境

本文硬件系统设计采用的是 Xilinx 公司 ARM+FPGA 架构的 Zynq xc7z030ffg676-2 芯片。搭建硬件系统时的开发环境是 Vivado 2018.3 和 Vivado HLS 2018.3。为了与 FPGA 进行比较,在 Inter i7-5960x CPU 和 NVIDIA TITAN X GPU 平台上使用 Ubuntu 16.04 操作系统以及 Chainer 5.3.0 框架,实现了与 FPGA 完全一致的 B-ResNet110 模型。实验测试采用的数据集是 CIFAR-10 数据集。

3.2　硬件系统设计

SOC 数据流如图 7 所示,卷积(Conv)计算单元、池化(Pool)计算单元、提前退出(Exit)计算单元、全连接层(FC)计算单元,卷积输出加法(Adder)计算单元的输入输出均通过 AXI 4 总线与 ARM 核的高速从机接口(Slave HP)相连。当 SOC 工作时,ARM 作为主机按照 B-ResNet 的计算要求,通过低速主机口(Master GP)使用轻量化 AXI 4 总线(AXI 4 Lite)发送控制信号给各个计算单元。各个计算单元收到信号后作为主机(Master),借助 Slave HP 从 DDR3 中读取特征图信息和权重信息,计算完成后再写回到 DDR3。

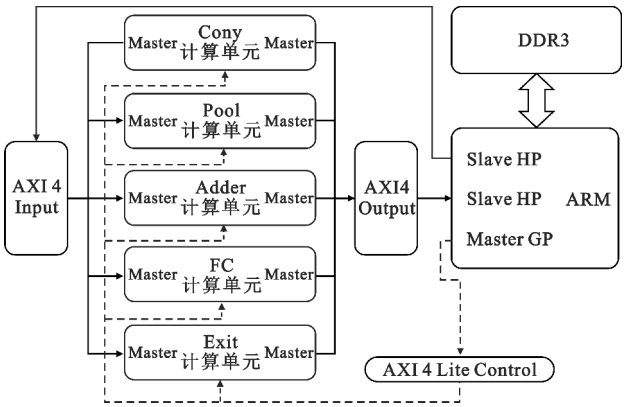

图 7　SOC 数据流图
Fig. 7 SOC data stream

根据图 7 所示的数据流图,在 Vivado 中设计搭建的 SOC 如图 8 所示。其中,计算单元中的 m_axi 是主机接口通过 AXI 连接模块(AXI SmartConnect)与 AMR 核的高速从机接口 S_AXI_HP 相连,传输数据信号,s_axi_CTRL 与 AMR 核的 M_AXI_GP 相连接收控制信号。

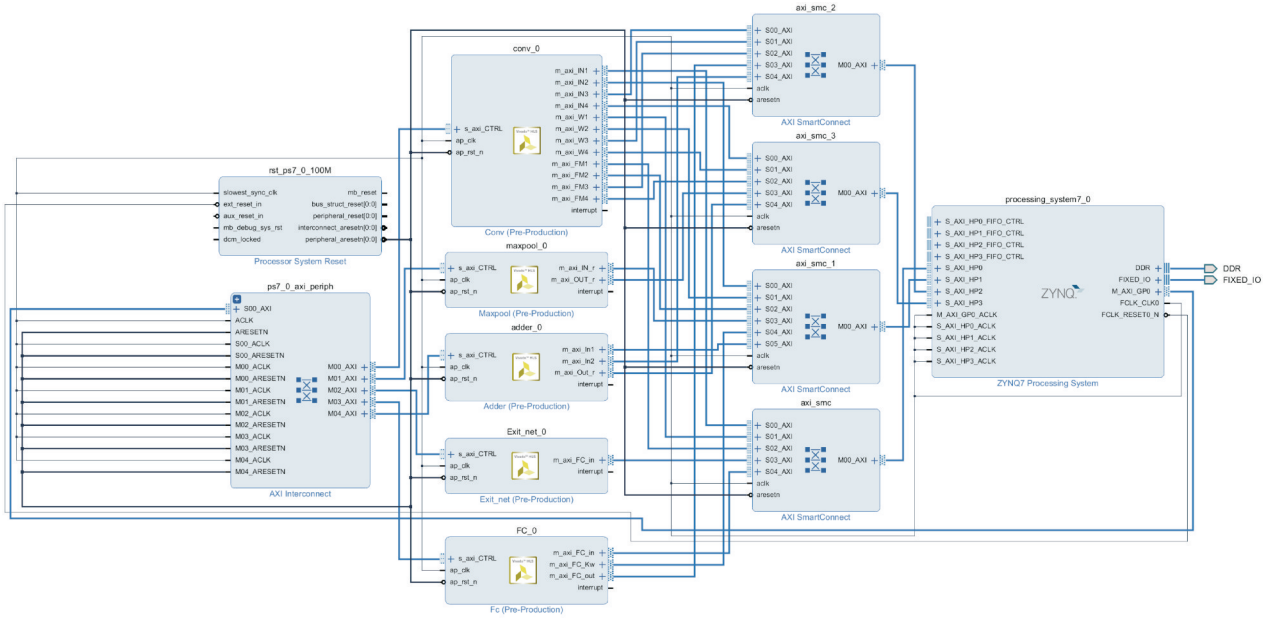

图 8　SOC 设计
Fig. 8 SOC desgin

硬件设计结果如表 2 所示,该 SOC 工作在 100 MHz 下,功耗为 2.853 W,使用了 70% 左右的 LUT 和 50% DSP 的处理单元,达到了最高 2.78×10^4 MOPS(Million Operations per Second)的 MAC 运算,1.25 MOPS 的 e 指数运算,以及 0.125 MOPS 的自然对数运算。

表 2 SOC 性能参数

Tab. 2 Performance parameters of SOC

相关参数	总资源	设计使用	利用率/%
LUT	78 600	55 268	70.32
DSP48E1	400	189	47.25
BRAM_18K	256	100.5	37.92
FF	157 200	55 501	35.31
功耗/W		2.853	
MAC		2.78×10^4 MOPS	
exp		1.25 MOPS	
ln		0.125 MOPS	
频率/MHz		100	

3.3 实验过程与结果分析

神经网络模型进行定点数量化前,需要在软件端将模型权重中浮点数转化为定点数,并对比不同类型定点数对应的准确率,根据对比结果将准确率最高的定点数类型进行量化。表 3 给出了对神经网络在不同定点数下的准确率对比,在数据类型中用 fixed<*m*,*fl*>来表示定点数,*m* 代表定点数的位宽,为 16 b,*fl* 代表 *m* b 定点数中小数的位数。根据不同类型定点数对应的准确率可以看出,当定点数的小数位数为 7 b 时,准确率明显低于原始网络;定点数的小数位数为 8 b 和 9 b 时,准确率虽有所提高但误差仍较大;当小数点位数为 10 b 时,准确率有着明显提升,与原网络的误差不到 0.01%;而小数点位数从 10 b 上升到 12 b 的过程中准确率却有所下降。因此,本设计将 fixed<16,10>的量化结果作为加速器的整体数据类型。

表 3 量化结果对比

Tab. 3 Comparison of quantitative results

网络类型	数据类型	准确率/%
原始网络	float 32	79.5499 9
量化网络	fixed<16,7>	50.580 001 0
	fixed<16,8>	76.579 999 0
	fixed<16,9>	79.350 000 1
	fixed<16,10>	79.540 001 0
	fixed<16,11>	79.530 000 2
	fixed<16,12>	79.530 000 21

定点数量化后的权重将加载到外部存储 DDR3,使用 ARM 控制 FPGA 中的计算 IP 实现 B-ResNet110 功能,对测试数据集进行识别。测试集图片在每个出口点的退出数量,以及每张图片的平均运行时间等由 ARM 统计并且通过串口传输到上位机。

表 4 中展示了在 FPGA 中实现的 B-ResNet110 对 CIFAR-10 数据集 10 000 张测试集的测试结果,以及 CPU 和 GPU 中 B-ResNet110 和 ResNet110 的测试结果。表中给出了网络在每个出口点输出图片的数量占测试集总数量的百分比,以及处理一张图片所用的平均时间。在退出结果误差不大的情况下,在时间方面 FPGA 相较于在 CPU 平台运行的 B-ResNet110,加速比达到 287%;对于 CPU 未加入提前退出机制的普通 ResNet110,平均推理时间节省了 80%。相较于 NVIDIA TITAN X GPU 运行的 B-ResNet110,加速比达到 145%;对于 GPU 中未加入提前退出机制的普通 ResNet110,平均推理时间节省了 63.9%。

表 4 FPGA 与 CPU 部署性能对比

Tab. 4 Comparison of FPGA and CPU deployment performance

设备	网络	平均时间/ms	输出图片数量占测试总数量的百分比/%		
			Exit 1	Exit 2	Exit 3
FPGA	B-ResNet110	25.56	41.15	13.57	45.28
CPU	ResNet110	137.20			100.00
	B-ResNet110	73.50	41.50	13.80	44.70
GPU	ResNet110	70.90			100.00
	B-ResNet110	37.20	41.50	13.80	44.70

4 结束语

本文从网络模型定点压缩,并行化、粗粒度流水线以及数学优化等角度完成了对卷积、池化、全连接和提前退出判断模块的 IP 设计,以及对 B-Resnst-110 网络的硬件实现。在 Zynq xc7z030ffg676-2 芯片搭建 SOC 系统,从消耗资源、计算能力、计算延迟等角度对硬件系统进行了分析。最终该系统获得了 2.78×10^4 MOPS 的最高算力,平均帧率达到了 39.12 FPS(Frame per Second)的正向推理速度,系统功耗仅为 2.853 W。

动态深度网络可以有效减少不必要的计算量,减轻神经网络在边缘部署时给边缘计算平台带来的

压力。未来将会继续探索其他动态网络在边缘高效部署的方法。

参考文献:

[1] LIU L,OUYANG W,WANG X,et al. Deep learning for generic object detection: a survey [J]. International Journal of Computer Vision,2020,128(2):261-318.

[2] 张宇,张雷. 融入注意力机制的深度学习动作识别[J]. 电讯技术,2021,61(10):1205-1212.

[3] 程千顷,王红军,丁希成,等. 基于迁移集成学习的无人机图像识别算法[J]. 电讯技术,2023,63(9):1277-1284.

[4] HE K,ZHANG X,REN S,et al. Deep residual learning for image recognition[C]//Proceedings of 2016 IEEE Conferenceon Computer Vision and Pattern Recognition. Las Vegas:IEEE,2016:770-778.

[5] 陈浩敏,姚森敬,席禹,等. YOLOv3-tiny 的硬件加速设计及 FPGA 实现[J]. 计算机工程与科学,2021,43(12):2139-2149.

[6] HAN Y, HUANG G, SONG S, et al. Dynamic neural networks:a survey[EB/OL]. (2021-02-06)[2022-08-19]. https://arxiv.org/abs/2102.04906.

[7] TEERAPITTAYANON S, MCDANEL B, KUNG H T. BranchyNet: fast inference via early exiting from deep neural networks [C]//Proceedings of the 23rd International Conference on Pattern Recognition. Cancun Center:IEEE,2016:2464-2469.

[8] MCGILL M, PERONA P. Deciding how to decide: dynamic routing in artificial neural networks [C]// Proceedings of 2017 International Conference on Machine Learning. Sydney:IEEE,2017:2363-2372.

[9] LI H,ZHANG H,QI X,et al. Improved techniques for training adaptive deep networks [C]//Proceedings of 2019 International Conference on Computer Vision. Seoul:IEEE,2019:1891-1900.

[10] 吴艳霞,梁楷,刘颖,崔慧敏. 深度学习 FPGA 加速器的进展与趋势[J]. 计算机学报,2019,42(11):2461-2480.

[11] VENIERIS S I, BOUGANIS C S. fpgaConvNet: a framework for mapping convolutional neural networks on FPGAs[C]//Proceedings of 2016 IEEE International Symposium on Field-Programmable Custom Computing Machines. Washington DC:IEEE,2016:40-47.

[12] LI R,LIU K,ZHAO M,et al. Maximizing CNN throughput on FPGA clusters[C]//Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. Seaside:ACM,2020:319-330.

[13] LIU X H, CHEN Y, HAO C, et al. WinoCNN: kernel sharing Winograd systolic array for efficient convolutional neural network acceleration on FPGAs [C]// Proceedings of 2021 IEEE 32nd International Conference on Application-specific Systems, Architectures and Processors. New Jersey:IEEE,2021:258-265.

作者简介:

王　鹏　男,1982 年生于新疆霍城,博士,研究员,主要研究方向为民机系统安全性设计与评估、机载电子硬件适航技术等。

任轶群　男,1997 年生于天津,硕士研究生,主要研究方向为神经网络硬件加速。

范毓洋　男,1988 年生于山西晋中,硕士,助理研究员,主要研究方向为机载电子适航设计验证、机载电子软错误防护、神经网络硬件加速等。

张嘉诚　男,1999 年生于湖北武汉,硕士研究生,主要研究方向为神经网络硬件加速。