(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 112734011 B
(45)授权公告日 2021.12.28

(21)申请号 202110010198.1
(22)申请日 2021.01.04
(65)同一申请的已公布的文献号
申请公布号 CN 112734011 A
(43)申请公布日 2021.04.30
(73)专利权人 北京大学
地址 100871 北京市海淀区颐和园路5号
(72)发明人 梁云 肖倾城
(74)专利代理机构 北京万象新悦知识产权代理有限公司 11360
代理人 黄凤茹
(51)Int.Cl.
G06N 3/04(2006.01)
G06N 3/063(2006.01)
G06F 15/78(2006.01)
(56)对比文件
CN 110892417 A,2020.03.17
CN 110321999 A,2019.10.11
CN 108805277 A,2018.11.13
CN 110766147 A,2020.02.07
CN 111752879 A,2020.10.09
US 2017124451 A1,2017.05.04
US 2018247196 A1,2018.08.30
卢丽强 等.面向卷积神经网络的FPGA设计.《中国科学》.2019,第49卷(第3期),第277-294页.
杜伟健 等.QingLong:一种基于常变量异步拷贝的神经网络.《计算机学报》.2020,第43卷(第4期),第587-599页.
Qingcheng Xiao et al..Fune:An FPGA Tuning Framework for CNN Acceleration.《IEEE Design&Test》.2020,第46-55页.
Ziming Xiong et al..A Survey of FPGA Based on Graph Convolutional Neural Network Accelerator.《2020 ICCEIC》.2020,第92-96页.
审查员 张娇

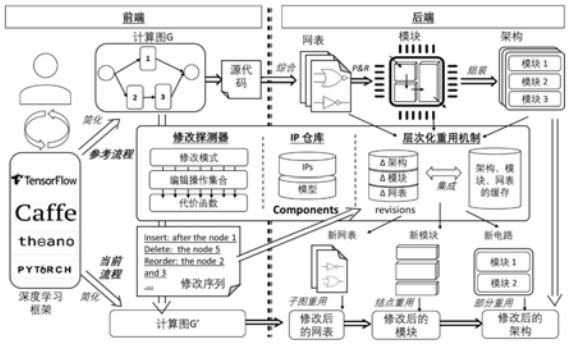
权利要求书3页 说明书10页 附图3页

(54)发明名称

一种基于增量综合的深度神经网络加速器协同设计方法

(57)摘要

本发明公布了一种基于增量综合的深度神经网络加速器协同设计方法,通过增量构造深度神经网络加速器,通过计算图修改检测方法识别上层应用的改变,再利用上层神经网络应用的改变修改层次化重用加速器硬件部分,减少深度神经网络加速器硬件的设计周期,从而提升加速器协同设计的效率。采用本发明方法,自动检测用户对深度神经网络进行的修改,通过多层次重用加速器设计参考,从而节省协同设计周期。同时,增量式生成的加速器可实现与人工设计的加速器相当的性能。



CN 112734011 B

1.一种基于增量综合的深度神经网络加速器协同设计方法,通过计算图修改检测方法识别上层应用的改变,再利用上层神经网络应用的改变修改层次化重用加速器硬件部分,增量构造深度神经网络加速器,减少深度神经网络加速器硬件的设计周期,从而提升加速器协同设计的效率;包括如下步骤:

1)将选取的深度神经网络模型DNN简化为计算图;

计算图中的结点为DNN模型中的层,对应于加速器硬件中的一个功能模块;计算图中边表示数据依赖关系,对应于加速器中功能模块间的数据通路;计算图划分为多个部分,每个部分为一张子图;

2)若未设置用于指导加速器硬件设计的加速器设计参考,则执行步骤3);若已有设置网络加速器设计参考,则执行步骤4);

3)构造神经网络加速器,作为协同设计的加速器设计参考,用于指导加速器硬件设计;

为每个深度神经网络构造专属的加速器硬件,并对FPGA平台进行编程;包括:

使用异构加速器设计,包括多个加速器架构,每个架构是一组功能模块的集合,用于处理计算图的一张子图;每个功能模块为架构中的硬件模块,占用FPGA物理位置上的硬件资源,包括可编程逻辑功能块,可编程I/O块和布线资源,实现子图中一个结点即神经网络层的计算功能;当一个加速器架构处理完某一个子图后,FPGA被重新配置成另一种加速器架构,继续处理另一张子图;

重复上述重配置过程,直至计算图的所有子图均被处理完;即构造得到神经网络加速器;

当构造的神经网络加速器的目标DNN模型算法更新时,转至步骤1)执行;

4)采用增量综合的方法完成加速器的设计,节省加速器实现时间;具体包括以下操作:

4A)定义计算图的修改模式,包括:删除,插入,替换,调序,复制,压缩和分解模式;

4B)通过比较当前的计算图与已有网络加速器设计参考中的计算图,使用图形编辑距离算法自动检测得到计算图的修改模式;定义计算图修改模式相对应的图形编辑操作及操作代价函数;包括如下步骤:

4B1)根据计算图修改模式定义一组图形编辑操作,分别对应每一个计算图修改模式,包括:插入模式、删除模式、替换模式、调序模式、复制模式;

4B2)定义每个图形编辑操作的代价函数;具体包括:

首先使用特征向量$\vec{v}$来抽象结点v;特征向量由归一化的计算复杂度和层参数组成,包括结点v的输入形状,过滤器形状,步幅和最大操作数量;基于特征向量,定义每种编辑操作的代价;

通过自定义的编辑操作及其代价,即可通过图形编辑距离算法自动检测出计算图中所做的修改;

4C)层次化重用加速器设计参考:当检测出计算图修改后,复用不需要修改的加速器架构和功能模块,采用三个重用层次实现加速器重用机制;

三个重用层次分别是:

1)子图重用:重用子图对应的加速器架构的完整物理实现;当被复制的结点来自同一子图时,在子图级别处理重复模式;

2)结点重用:重用处理某一结点即神经网络层的功能模块的物理实现;

3)部分重用:重用加速器中某一功能模块的网表,该网表需重新运行布局布线P&R得到物理实现;

对于FPGA平台上的深度神经网络加速器设计,每个硬件功能模块通过在FPGA资源上布局布线实现,即一个功能模块的寄存器转换级电路RTL代码在经过逻辑综合后可得到该功能模块的网表;网表进一步在FPGA资源上布局布线,得到该功能模块的物理实现;一个加速器架构中所有功能模块的物理实现即为该架构的完整物理实现;

通过层次化重用加速器设计参考,即得到适配当前DNN算法的神经网络硬件加速器;

通过上述步骤,即实现基于增量综合的深度神经网络加速器协同设计。

2.如权利要求1所述基于增量综合的深度神经网络加速器协同设计方法,其特征是,步骤1)将选取的深度神经网络模型DNN具体简化为神经网络开放标准ONNX的计算图形式。

3.如权利要求1所述基于增量综合的深度神经网络加速器协同设计方法,其特征是,步骤3)中,异构加速器设计包含的加速器架构数量及每个架构中各层占用的FPGA资源,根据计算图划分算法得到,具体可通过设置知识产权核IP核参数来配置IP核仓库内的知识产权核实现各个功能模块;IP核参数由IP核仓库中的性能和资源模型确定。

4.如权利要求1所述基于增量综合的深度神经网络加速器协同设计方法,其特征是,步骤3)中,构造得到神经网络加速器后,可对加速器的质量,性能,准确性和面积信息进行测评。

5.如权利要求1所述基于增量综合的深度神经网络加速器协同设计方法,其特征是,步骤4A)中定义计算图的修改模式具体为:

a)删除模式是删除计算图中的结点;

b)插入模式是在图中插入一个新结点;

c)替换模式是将一个结点替换为具有不同操作的另一个结点;

d)调序模式是对两个相邻结点重新排序,即互换序号;

e)复制模式是将子图的副本插入计算图中;

f)压缩模式是压缩某些结点,包括减少某结点的特征图和过滤器的数量,以降低计算复杂度;

g)分解模式是分割结点,包括将某个结点分解成两个结点,通过用多个小过滤器替换一个过滤器,以减少计算。

6.如权利要求1所述基于增量综合的深度神经网络加速器协同设计方法,其特征是,步骤4B)所述图形编辑距离算法具体采用基于容错的近似图匹配算法。

7.如权利要求1所述基于增量综合的深度神经网络加速器协同设计方法,其特征是,步骤4B1)中,图形编辑操作模式的符号表示分别为:包括:插入模式、删除模式、替换模式、调序模式、复制模式;

插入模式是将空结点ε映射到新计算图中的结点μ;

删除模式是将旧计算图中的结点v映射到空结点ε;

替换模式和压缩模式使用相同的符号表示:v→μ,其中v和μ分别是旧计算图和新计算图中的结点;通过神经网络层类型和参数数量区分;如果一对结点具有不同的层类型,则将其识别为替换模式;如果两个结点的层类型相同但参数数量不同,则为压缩模式;

调序模式是两个编辑操作v1→μ2和v2→μ1,v1、v2是旧计算图相连的结点,μ1、μ2是新

计算图中相连的结点；

复制模式：使用后期处理检测复制模式的修改，即：找到所有插入的结点后，枚举由这些结点组成的子图；对于每个子图，如果它与旧计算图中的任何子图同构，则为复制模式。

8.如权利要求1所述基于增量综合的深度神经网络加速器协同设计方法，其特征是，步骤4B2)基于特征向量，定义每种编辑操作的代价，包括：

用α·OPs(v)/OPsmax表示插入操作的代价，其中OPs(v)表示结点v的操作数，OPsmax表示全部结点中最大的操作数，α为协因子；

用α·OPs(v)/OPsmax表示删除操作的代价；

替换操作的代价定义为max{OPs(v)，OPs(μ)}·β/OPsmax，其中OPs(v)、OPs(μ)表示结点v、μ的操作数，OPsmax表示全部结点中最大的操作数，β为协因子；

压缩操作v→μ的代价定义为压缩前后结点特征向量的欧几里得距离；

调序操作的代价定义为常数；

分解操作的代价定义为max{OPs(v)，∑OPs(μ)}·α/OPsmax，其中OPs(v)为分解前结点的操作数，∑OPs(μ)为分解后结点的操作数总和，OPsmax表示全部结点中最大的操作数，α为协因子。

9.如权利要求1所述基于增量综合的深度神经网络加速器协同设计方法，其特征是，构造得到神经网络加速器后，具体可在Caffe，PyTorch框架中对加速器的质量，性能，准确性和面积信息进行测评，并修改DNN模型算法，得到最终的神经网络加速器设计。

一种基于增量综合的深度神经网络加速器协同设计方法

技术领域

[0001] 本发明涉及加速器协同设计技术,尤其涉及一种基于增量综合的深度神经网络加速器硬件协同设计方法。

背景技术

[0002] 深度神经网络(Deep Neural Network,DNN)因其接近甚至优于人类的出色准确率而收到越来越多的关注。从光学字符识别到语音动作识别,从行人识别到物体分类,深度神经网络已广泛应用于各个领域。DNN所要求的大量计算能力使得高性能、低功耗的神经网络加速器成为一种刚需。在各种硬件平台中,现场可编程门阵列(Field Programmable Gate Array,FPGA)由于其可重新配置的特性,最常被用做深度学习加速器的原型验证,并且可在Microsoft Azure和Amazon Web Services等云服务中广泛使用。

[0003] 深度神经网络及其加速器协同设计方法是指通过将DNN拓扑结构、运算与硬件加速器体系结构相对应,将DNN定制为加速器硬件。FPGA的可重配置特性使其非常适合协同设计方案。协同设计方法的另一个好处是,开发人员可以获取最终产品对结果质量的早期反馈,包括质量,性能,准确性和面积。通过协同设计硬件和软件,开发人员可以通过迭代修改或添加功能来完善设计。然而在典型的协同设计流程中,每当更改DNN软件时,都必须重新综合生成加速器硬件。

[0004] 表1现有加速器协同设计工具的性能比较

[0005]

项目	TVM[1],[2]	DNNWeaver[3],[4]
架构	可编程	专用
性能	低	中
软件编译时间缩减	有	有
硬件综合时间缩减	无	无
协同设计代价	中	高

[0006] 综合时间长是目前DNN及其加速器协同设计方法的主要障碍之一。表1所示为现有加速器协同设计工具的性能比较。例如,TVM工具可以用于在各种设备(例如通用Tensor Accelerator(VTA)硬件加速器)上部署DNN,实现协同设计。TVM可以根据DNN软件来调整VTA的硬件固有属性,内存大小和数据类型。使用DNNWEAVER工具实现协同设计时,开发人员可以通过在硬件描述语言(Hardware Description Language,HDL)级别上使用可自定义的模板来修改硬件。然而,这些协同设计方法工具都需要经历漫长的硬件综合过程。硬件综合步骤将把加速器源代码综合为硬件的最终物理实现,这涉及高层次综合(High-level Synthesis,HLS),逻辑综合,布局和布线(P&R)。硬件综合时间取决于综合工具,设计大小,频率等。HLS,逻辑综合和P&R通过都需要数小时,因此硬件综合要比软件编译步骤慢几个数量级。

[0007] 参考文献

[0008] [1]Tianqi Chen,Thierry Moreau,Ziheng Jiang,Haichen Shen,Eddie Yan,

Leyuan Wang,Yuwei Hu,Luis Ceze,Carlos Guestrin,and Arvind Krishnamurthy.2018.TVM:End-to-End Optimization Stack for Deep Learning.In SysML Conference.

[0009] [2]G.Hegde,Siddhartha,N.Ramasamy,and N.Kapre.2016.CaffePresso:An optimized library for Deep Learning on embedded accelerator-based platforms.In 2016International Conference on Compliers,Architectures,and Sythesis of Embedded Systems(CASES).1-10.

[0010] [3]Hardik Sharma,Jongse Park,Divya Mahajan,Emmanuel Amaro,Joon Kyung Kim,Chenkai Shao,Asit Mishra,and Hadi Esmaeilzadeh.2016.From high-level deep neural models to FPGAs.In Microarchitecture(MICRO),2016 49th Annual IEEE/ACM International Symposium on.IEEE,1-12.

[0011] [4]Deguang Wang,Junzhong Shen,Mei Wen,and Chunyuan Zhang.2019.An Efficient Design Flow for Accelerating Complicated-connected CNNs on a Multi-FPGA Platform.In Proceedings of the 48th International Conference on Parallel Processing.ACM,98.

发明内容

[0012] 为了克服上述现有技术的不足,本发明提供一种基于增量综合的深度神经网络加速器协同设计方法,通过基于增量编译的加速器构建方法,克服深度神经网络加速器硬件在协同设计中耗时过长的问题。

[0013] 为方便起见,本发明采用以下术语定义:

[0014] FPGA(Field Programmable Gate Array)现场可编程门阵列

[0015] DNN(Deep Neural Network)深度神经网络

[0016] CNN(Convolutional Neural Network)卷积神经网络

[0017] P&R(Placement and Routing)布局布线

[0018] IP核(Intellectual Property Core)知识产权核:在集成电路的可重用设计方法学中,指某一方提供的、形式为逻辑单元、芯片设计的可重用模组

[0019] RTL(Register Transistor Level)寄存器转换级电路

[0020] HLS(High-level Synthesis)高层次综合

[0021] PE(Processing Element)计算单元

[0022] HDL(Hardware Description Language)硬件描述语言

[0023] I/O(Input/Output)输入输出

[0024] 开源项目:

[0025] ONNX(Open Neural Network Exchange)神经网络开放标准:是微软和Facebook发布了一个开放的深度学习开发工具生态系统

[0026] Caffe:由伯克利人工智能研究小组和伯克利视觉和学习中心开发的一个兼具表达性、速度和思维模块化的深度学习框架。

[0027] PyTorch:是一个Python优先的深度学习框架,能够在强大的GPU加速基础上实现张量和动态神经网络。

[0028] 本发明提供的技术方案是：

[0029] 一种基于增量综合的深度神经网络加速器协同设计方法。该方法应用于深度神经网络加速器设计技术领域，通过增量构造深度神经网络加速器，减少深度神经网络加速器硬件的设计周期，从而提升加速器协同设计的效率。总的来说，本发明提供的技术方案包括层次化重用加速器硬件部分，由计算图修改检测方法识别上层应用的改变，再利用上层神经网络应用的改变尽可能少的修改层次化重用加速器硬件部分，从而缩短设计开发的周期。具体来说，本发明方法包括如下步骤：

[0030] 1)将选取的DNN模型简化为计算图，计算图中的结点是DNN模型中的层，对应于加速器硬件中的一个功能模块，而计算图中边表示数据依赖关系，对应于加速器中功能模块间的数据通路；本发明中，将计算图划分为多个部分，每个部分为一张子图。

[0031] 具体实施时，计算图定义为神经网络开放标准(Open Neural Network Exchange，ONNX)的图定义形式。

[0032] 2)若未设置用于指导加速器硬件设计的加速器设计参考，则执行步骤3)；若已有设置网络加速器设计参考，则执行步骤4)；

[0033] 3)构造神经网络加速器，作为协同设计的加速器设计参考，用于指导加速器硬件设计；

[0034] 本发明方法为每个深度神经网络构造专属的加速器硬件，并对FPGA平台进行编程。具体而言，本发明方法使用异构加速器设计，如附图4所示。该异构加速器设计包括多个加速器架构，每个架构是一组功能模块的集合，用于处理计算图的一张子图。每个功能模块为架构中的硬件模块。它们占用一部分FPGA物理位置上的硬件资源(包括可编程逻辑功能块，可编程I/O块和布线资源)，实现子图中一个结点(神经网络层)的计算功能。当一个加速器架构处理完某一个子图后，FPGA被重新配置成另一种加速器架构，从而继续处理另一张子图。重复该重配置过程，直至计算图的所有子图均被处理完。异构加速器设计包含的加速器架构数量以及每个架构中各层占用的FPGA资源将由计算图划分算法决定。异构加速器设计实现为物理硬件时，本发明方法将通过设置知识产权核(Intellectual Property Core，IP核)参数来配置IP核仓库内的知识产权核(IP Core)实现各个功能模块。IP核参数由IP核仓库中的性能和资源模型确定。

[0035] 构造得到神经网络加速器后，可以对加速器的质量，性能，准确性和面积等信息进行测评。可在Caffe，PyTorch或其他框架中修改DNN模型算法，使得加速器测评结果能够满足需要，由此得到最终的神经网络加速器设计，结束步骤流程。当构造的神经网络加速器的目标DNN模型算法更新时，转至步骤1)执行；

[0036] 4)采用增量综合的方法完成加速器的设计，节省加速器实现时间；具体包括以下操作：

[0037] 4A)定义计算图的修改模式，包括：删除，插入，替换，调序，复制，压缩和分解模式；

[0038] a)具体地，计算图的修改模式分别为：删除模式是删除计算图中的结点。

[0039] b)插入模式是在图中插入一个新结点。

[0040] c)替换模式是将一个结点替换为具有不同操作的另一个结点。

[0041] d)调序模式是对两个相邻结点重新排序，即互换序号。

[0042] e)复制模式是将子图的副本插入计算图中。

[0043] f)压缩模式是压缩某些结点,例如减少某结点的特征图和过滤器的数量,以降低计算复杂度。

[0044] g)分解模式是分割结点,例如,将某个结点分解成两个结点。此模式是通过用多个小过滤器替换一个过滤器,以减少计算。

[0045] 4B)通过比较当前的计算图与已有网络加速器设计参考中的计算图,自动检测得到计算图的修改模式;

[0046] 使用图形编辑距离算法来自动检测计算图中的修改模式。具体实施时,图形编辑距离算法采用基于容错的近似图匹配算法。

[0047] 定义计算图修改模式相对应的图形编辑操作及操作代价函数。

[0048] 4B1)根据计算图修改模式定义了一组图形编辑操作,分别对应每一个计算图修改模式。计算图修改模式的图形编辑操作的符号表示分别为:

[0049] 插入模式是将空结点ε映射到新计算图中的结点μ。

[0050] 删除是将旧计算图中的结点v映射到空结点ε。

[0051] 替换模式和压缩模式使用相同的符号表示:v→μ,其中v和μ分别是旧计算图和新计算图中的结点。通过神经网络层类型和参数数量来区分它们。如果一对结点具有不同的层类型,则将其识别为替换模式。否则,如果两个结点的层类型相同但参数数量不同,则是一种压缩模式。

[0052] 调序模式可以看作是两个编辑操作v1→μ2和v2→μ1,v1、v2是旧计算图相连的结点,μ1、μ2是新计算图中相连的结点。

[0053] 使用后期处理来检测复制模式的修改。找到所有插入的结点后,枚举由这些结点组成的子图。对于每个子图,如果它与旧计算图中的任何子图同构,则将其识别为复制模式。

[0054] 4B2)定义每个图形编辑操作的代价。

[0055] 主要根据结点的复杂性来定义代价函数。具体地:

[0056] 首先使用特征向量$\vec{v}$来抽象结点v。特征向量由归一化的计算复杂度和层参数组成,包括结点v的输入形状,过滤器形状,步幅和最大操作数量等。基于特征向量,我们定义每种编辑操作的代价,如表2所示。

[0057] 表2计算图修改模式对应的图形编辑操作集合

[0058]

模式	图形编辑操作表示	代价函数
插入模式	$\varepsilon \rightarrow v$	$\alpha \cdot OPs(v)/OPsmax$
删除模式	$v \rightarrow \varepsilon$	$\alpha \cdot OPs(v)/OPsmax$
替换模式	$v \rightarrow \mu$	$\max\{OPs(v), OPs(\mu)\} \cdot \beta/OPsmax$
压缩模式	$v \rightarrow \mu$	$\lVert \vec{v} - \vec{\mu} \rVert$
调序模式	$v1 \rightarrow \mu2 \,\&\, v2 \rightarrow \mu1,$	Const.

[0059]

	$\langle v1, v2\rangle \in E$ & $\langle \mu 1, \mu 2\rangle \in E'$	
分解模式	$v \rightarrow (\mu 1, \ldots, \mu n)$	$\max\{OPs(v), \sum OPs(\mu)\} \cdot \alpha / OPsmax$

[0060] 表中ε表示空结点。对于插入操作ε→v,我们用α·OPs(v)/OPsmax表示其代价,其中OPs(v)表示结点v的操作数,OPsmax表示全部结点中最大的操作数,α为协因子。类似的,对于删除操作v→ε,我们也用α·OPs(v)/OPsmax表示其代价。

[0061] 对于替换操作v→μ,即用v替换结点μ,我们将其代价定义为max{OPs(v),OPs(μ)}·β/OPsmax,其中OPs(v)、OPs(μ)表示结点v、μ的操作数,OPsmax表示全部结点中最大的操作数,β为协因子。

[0062] 对于压缩操作v→μ,我们将代价定义为压缩前后结点特征向量的欧几里得距离。

[0063] 对于调序操作,我们将代价定义为常数。

[0064] 对于分解操作v→(μ1,...,μn),我们将代价定义为max{OPs(v),∑OPs(μ)}·α/OPsmax,其中OPs(v)为分解前结点的操作数,∑OPs(μ)为分解后结点的操作数总和,OPsmax表示全部结点中最大的操作数,α为协因子。

[0065] 通过自定义的编辑操作及其代价,即可以通过图形编辑距离算法自动检测出计算图中所做的修改。

[0066] 4C)层次化重用加速器设计参考;

[0067] 当计算图修改被检测出之后,本发明方法将复用不需要修改的加速器架构和功能模块,从而缩短深度学习加速器的设计时间。对于FPGA平台上的深度神经网络加速器设计,每个硬件功能模块是通过在一部分FPGA资源上布局布线实现的。具体而言,一个功能模块的寄存器转换级电路(Register Transistor Level,RTL)代码在经过逻辑综合后可以得到该功能模块的网表。网表进一步在FPGA资源上布局布线,得到该功能模块的物理实现。一个加速器架构中所有功能模块的物理实现即为该架构的完整物理实现。

[0068] 本发明在三个重用层次实现了加速器重用机制,分别是:

[0069] 1)子图重用:重用子图对应的加速器架构的完整物理实现。当被复制的结点来自同一子图时,可以在子图级别处理重复模式;

[0070] 2)结点重用:重用处理某一结点(神经网络层)的功能模块的物理实现。对于调序,替换和压缩模式,可以直接重用未更改的模块的物理实现;

[0071] 3)部分重用:重用加速器中某一功能模块的网表,但该网表需要重新运行布局布线(P&R)从而得到物理实现。插入,删除和分解模式会影响子图的结点数量和拓扑结构。结点数量和拓扑的变化需要重新综合硬件设计。因此,我们只能重用一些未更改模块的网表。

[0072] 本发明通过上述层次化重用机制来处理所有检测到的计算图修改模式。具体而言,子图重用是完全重用子图对应的加速器架构。我们使用A,B和B'表示三个子图。B'来自当前DNN图,而A和B来自参考DNN图。如果未对子图进行任何修改(B'=B),则相应的加速器架构的物理实现将保持不变并直接重用。如果新子图B'与A完全重复,则A和B'将共享相同的架构物理实现。在更一般的情况下,从A的一部分复制了子图B',本方法将完全复用A的硬件,然后删除冗余功能模块。

[0073] 结点重用是重用子图中的结点(层)对应功能模块的完整物理实现。调序模式通常发生于卷积层及其后的池化函数之间。在我们的加速器设计中,卷积功能模块都内嵌了池化函数,但是默认情况下并不会激活。将当前卷积层及其后的池化函数调序,需要进行的物理实现改变是将当前卷积模块的池化函数关闭,然后激活前一卷积层模块的池化函数。对于替换模式和压缩模式,本发明方法将利用被替换模块或被压缩模块的资源和位置,对新模块进行逻辑综合和布局布线。

[0074] 部分重用是仅重用某些结点对应功能模块的网表,但仍需要经过P&R步骤。通过不激活某个模块,可以快速删除对应的结点。但是,当不激活的模块占用大量资源时,加速器性能将受到影响。为了避免资源利用不足,本方法将重新探索模块参数并重新实现子图的架构。插入和分解修改模式必定会更改子图拓扑结构,并向架构引入新模块。由于现有的模块具有固定的物理位置并占用最多的资源,因此直接插入新模块难以满足FPGA固定的资源限制。本方法将为修改后的子图生成新的架构。

[0075] 重新生成处理子图的加速器架构时,已有功能模块的IP核参数可能会更改,也可能不会更改。本方法将重用参数不变的模块的网表。这些重用的网表避免了逻辑综合,可以布线到新位置。对于参数已更改的模块,本方法将重新配置IP核并生成RTL代码。然后,将RTL代码综合、布局布线并与最终设计集成。本框架可以通过快速组装网表进一步加快模块的综合。

[0076] 通过层次化重用加速器设计参考,该方法将得到适配当前DNN算法的硬件加速器,并输出给用户。

[0077] 构造得到神经网络加速器后,可以对加速器的质量,性能,准确性和面积等信息进行测评。可在Caffe,PyTorch或其他框架中修改DNN模型算法,使得加速器测评结果能够满足需要,由此得到最终的神经网络加速器设计,结束步骤流程。当构造的神经网络加速器中的DNN模型算法更新时,转至步骤1)执行。

[0078] 通过上述步骤,即可实现基于增量综合的深度神经网络加速器协同设计。

[0079] 与现有技术相比,本发明的有益效果:

[0080] 本发明所提供的深度神经网络加速器协同设计方法,自动检测用户对深度神经网络进行的修改,然后通过从子图、结点、部分重用三个层次重用加速器设计参考,从而与重新生成加速器的设计方法相比,节省了高达4.43倍的协同设计周期。同时,这样增量式生成的加速器可实现与人工设计的加速器相当的性能。

附图说明

[0081] 图1是本发明提供的基于增量综合的深度神经网络加速器协同设计方法的流程框图。

[0082] 图2是本发明方法定义的计算图修改模式的流程示意图。

[0083] 图3是本发明实施例中层次化重用加速器参考具体实现流程示意图。

[0084] 图4是本发明具体实施深度学习加速器设计结构示意图。

具体实施方式

[0085] 下面结合附图,通过实施例进一步描述本发明,但不以任何方式限制本发明的范

围。

[0086] 在软硬件协同设计技术领域，一般地，对深层神经网络模型的大部分修改是局部的和较小的，能够满足硬件重用。在硬件重用中，仅对深层神经网络模型修改的部分进行重新合成。例如，仅需要重新综合那些修改了像素尺寸和操作类型的网络层，而不变的层及其在硬件上的实现则保持不变。

[0087] 本发明提供一种基于增量综合的深度神经网络加速器协同设计方法，通过基于增量编译的加速器构建方法，克服深度神经网络加速器硬件在协同设计中耗时过长的问题。

[0088] 附图1描述了基于增量综合的协同设计方法的整体流程，本发明方法包括如下步骤：

[0089] 1)用户输入用Caffe，PyTorch或其他框架描述的DNN算法，如附图1最左侧所示。本发明将给定的DNN软件简化为计算图，用于指导加速器硬件设计。计算图中结点是层，对应于加速器硬件中的一个功能模块，而图中边表示数据依赖关系，对应于加速器中功能模块间的数据通路。

[0090] 2)然后该方法将根据有无加速器设计参考确定下一步执行步骤。如无加速器设计参考，执行图步骤3)，若有，则执行步骤4)。

[0091] 3)本方法将构造一个全新的神经网络加速器，并作为下一次协同设计的加速器设计参考。如附图1顶部所示，该方法将使用计算图划分算法将计算图划分为多个部分，每个部分即为一张子图。该方法将构造多个加速器架构，每个架构是一组功能模块的集合，用于处理一张子图。一旦加速器处理完某个子图，它将重新配置成另一个架构继续处理另一张子图，直至计算图的所有子图被处理完。进一步来说，架构中的每个功能模块有其固定的物理位置(占用的FPGA资源)，处理子图中的一个结点。该方法在实现每个功能模块时，将设置参数来配置IP核仓库内的知识产权核(IP Core)。IP核参数由库中的性能和资源模型确定。所有的加速器架构组成了最终的加速器设计，并输出给用户。接着，该方法执行步骤5)。

[0092] 4)若存在加速器设计参考，该方法将采用增量综合的方法节省加速器实现时间。该步骤又可细分为如下几个子步骤：

[0093] a)检测计算图的修改。该方法比较当前的计算图与参考设计中的计算图，检测出计算图中所作的修改。我们首先定义了七种常见的修改模式，如附图2所示：删除，插入，替换，调序，复制，压缩和分解。

[0094] I.删除模式是删除计算图中的结点。在附图2(a)中，删除了结点4。

[0095] II.插入模式是在图中插入一个新结点。在附图2(b)中，在结点3和4之间插入了一个新结点5。

[0096] III.替换模式是将一个结点替换为具有不同操作的另一个结点。在附图2(c)中，替换了结点2。

[0097] IV.调序模式是对两个相邻结点重新排序。结点3和4在附图2(d)中互换。

[0098] V.复制模式是将子图的副本插入图中。在附图2(e)中，复制了由结点1、2和3组成的子图。

[0099] VI.压缩模式是压缩某些结点。在附图2(f)中，减少了结点3的特征图和过滤器的数量，以降低计算复杂度。

[0100] VII.分解模式是分割结点。在附图2(g)中，结点3被分解为结点3a和3b。此模式是

减少计算的另一种方法。此模式不是减少过滤器数量，而是用多个小过滤器替换一个过滤器。

[0101] 然后，我们使用图形编辑距离算法来自动检测计算图中的修改模式。图形编辑距离算法需要定义图形编辑操作及操作代价。我们根据修改模式定义了一组图形编辑操作，如表2中所列。插入模式是将空结点ε映射到新图中的结点μ。删除是将旧图中的结点v映射到空结点ε。替换和压缩模式使用相同的符号表示v→μ，v和μ分别是旧图和新图中的结点。我们通过神经网络层类型和参数数量来区分它们。如果一对结点具有不同的层类型，则将其识别为替代。否则，如果两个结点的参数数量不同，则是一种压缩。调序模式可以看作是两个编辑操作v1→μ2和v2→μ1，v1、v2是旧图相连的结点，μ1、μ2是新图中相连的结点。我们使用后期处理来检测复制模式的修改。找到所有插入的结点后，我们枚举由这些结点组成的子图。对于每个子图，如果它与旧图中的任何子图同构，则将其识别为复制模式。

[0102] 在表2中，我们已经根据结点的特征向量定义了每种编辑操作的代价。通过自定义的编辑操作及其代价，我们可以通过图形编辑距离算法自动检测出计算图中所作的修改。

[0103] b)层次化重用加速器设计参考。当计算图修改被检测出之后，该方法将复用不需要修改的加速器设计，从而缩短综合时间。该方法在三个层次实现了加速器重用机制。这三个重用层次分别是：1)子图重用：重用子图的完整硬件实现；2)结点重用：重用结点(层)对应功能模块的完整物理实现；3)部分重用：重用架构中部分功能模块的网表，但需要重新运行布局布线(P&R)。该方法可以通过该层次化重用机制来处理所有检测到的计算图修改。具体来说，当被复制的结点来自同一子图时，我们可以在子图级别处理重复模式。插入，删除和分解模式会影响子图的结点数量和拓扑结构。结点数量和拓扑的变化需要重新综合硬件设计。因此，我们只能重用一些未更改模块的网表。对于其他修改模式(例如调序，替换和压缩模式)，我们可以直接重用未更改的模块的物理实现。

[0104] 通过层次化重用加速器设计参考，该方法将得到适配当前DNN算法的硬件加速器，并输出给用户。接着，该方法执行步骤5)。

[0105] 5)用户得到加速器设计后，可以对加速器的质量，性能，准确性和面积等信息进行测评。用户将在Caffe，PyTorch或其他框架中修改DNN算法，直到用户满意加速器测评结果。一旦DNN算法有所修改，该方法将转至步骤1)继续执行。

[0106] 具体实施时，步骤1)的计算图定义实现方法为ONNX的图定义形式。步骤3)中的计算图划分算法为参考文献(Xiao,Q.,Liang,Y.,Lu,L.,Yan,S.and Tai,Y.W.,2017,June.Exploring heterogeneous algorithms for accelerating deep convolutional neural networks on FPGAs.In Proceedings of the 54th Annual Design Automation Conference 2017(pp.1-6).)中的动态规划算法。步骤4.a)中的图形编辑距离算法基于容错的近似图匹配算法。

[0107] 我们以附图3为例说明步骤4b)中的层次化重用加速器设计参考。附图3中给出DNN算法软件被简化为一张计算图，并进一步被划分为五个子图(子图1～5)的一个示例。每张子图对应一份参考的加速器设计中的架构，子图中的每个结点对应架构中的每个功能模块。例如，子图2包括五个结点，对应于架构2原始电路中的Conv1、Conv2、Conv3、Concat、FC五个功能模块。每个模块可以被单独激活或关闭。在此示例中，子图4从子图3复制而来。附图3(a)是子图2的原始电路。附图3(b)、(c)、(d)是对子图2可能进行的一些修改。

[0108] 1)子图重用是完全重用子图对应的加速器架构。我们使用A,B和B'表示三个子图。B'来自当前DNN图,而A和B来自参考DNN图。如果未对子图进行任何修改(B'＝B),则相应的硬件架构将保持不变并直接重用。在附图3中,子图1、3和5属于这种情况。则对应的,参考的加速器设计中的架构1、架构3和架构5保持不变。如果新子图B'与A完全重复,则A和B'将共享相同的架构物理实现。在附图3中,子图4与3相同,则它们的架构3和4共享相同的物理实现。在更一般的情况下,从A的一部分复制了子图B',本方法将完全复用A的硬件,然后删除冗余功能模块。

[0109] 2)结点重用是重用子图中的结点(层)对应功能模块的完整物理实现。在我们的加速器设计中,卷积模块都内嵌了池化函数,但是默认情况下并不会激活。对于子图2,如果将Conv3结点及其后的池化函数(pl)调序,对于硬件架构2而言,需要进行的物理实现改变是将Conv 3模块的池化函数关闭,然后激活Conv 2模块的池化函数,如图3(c)所示。这样,卷积模块就可以完全重用而不经过P&R步骤。

[0110] 对于替换模式,本发明方法将利用被替换模块的资源和位置实现新模块。附图3(b)给出了一个示例,与(a)中的原始电路相比,全相联模块FC被卷积模块Conv 4代替。本方法将直接重用FC以外的所有其他模块的实现。新模块Conv4将被重新综合生成,并放置在原来FC模块所在区域。压缩模式类似于替换模式。压缩层时,其模块参数可能会更改。本方法将在原始模块的相同区域中重新实现压缩后的功能模块。

[0111] 3)部分重用是仅重用某些结点对应功能模块的网表,但仍需要经过P&R步骤。通过不激活某个模块,可以快速删除对应的结点。但是,当不激活的模块占用大量资源时,加速器性能将受到影响。为了避免资源利用不足,本方法将重新探索模块参数并重新实现子图的架构。插入和分解修改模式必定会更改子图拓扑结构,并向架构引入新模块。由于现有的模块具有固定的物理位置并占用最多的资源,因此直接插入新模块难以满足资源限制。本方法将为修改后的子图生成新的架构。

[0112] 重新生成处理子图的加速器架构时,已有功能模块的IP核参数可能会更改,也可能不会更改。本方法将重用参数不变的模块的网表。这些重用的网表避免了逻辑综合,可以布线到新位置。附图3(d)说明了将Conv 4模块插入架构2的例子。实现新架构时,Conv 1和Conv2模块的会缩小(分配资源减少),以便为Conv 4模块腾出FPGA物理空间和资源。Conv 3和FC模块的参数保持不变,所以网表可以被完全重用。

[0113] 对于参数已更改的模块,本方法将重新配置IP核并生成RTL代码。然后,将RTL代码综合、布局布线并与最终设计集成。本框架可以通过快速组装网表进一步加快模块的综合。具体来说,在卷积模块和完全连接的模块中,本框架以网状拓扑连接其计算单元(Processing Element,PE),如附图3(e)所示。每个PE处理卷积和矩阵乘法的部分结果。这些PE是同质的,并且具有相同的网表。本框架可以重用一个PE的网表来构建卷积或任意形状的全相连模块。

[0114] 使用这种层次化重用方法,可以在生成新加速器设计时重用加速器设计参考的大部分内容,从而减少总体运行时间。

[0115] 需要注意的是,公布实施例的目的在于帮助进一步理解本发明,但是本领域的技术人员可以理解:在不脱离本发明及所附权利要求的精神和范围内,各种替换和修改都是可能的。因此,本发明不应局限于实施例所公开的内容,本发明要求保护的范围以权利要求

书界定的范围为准。

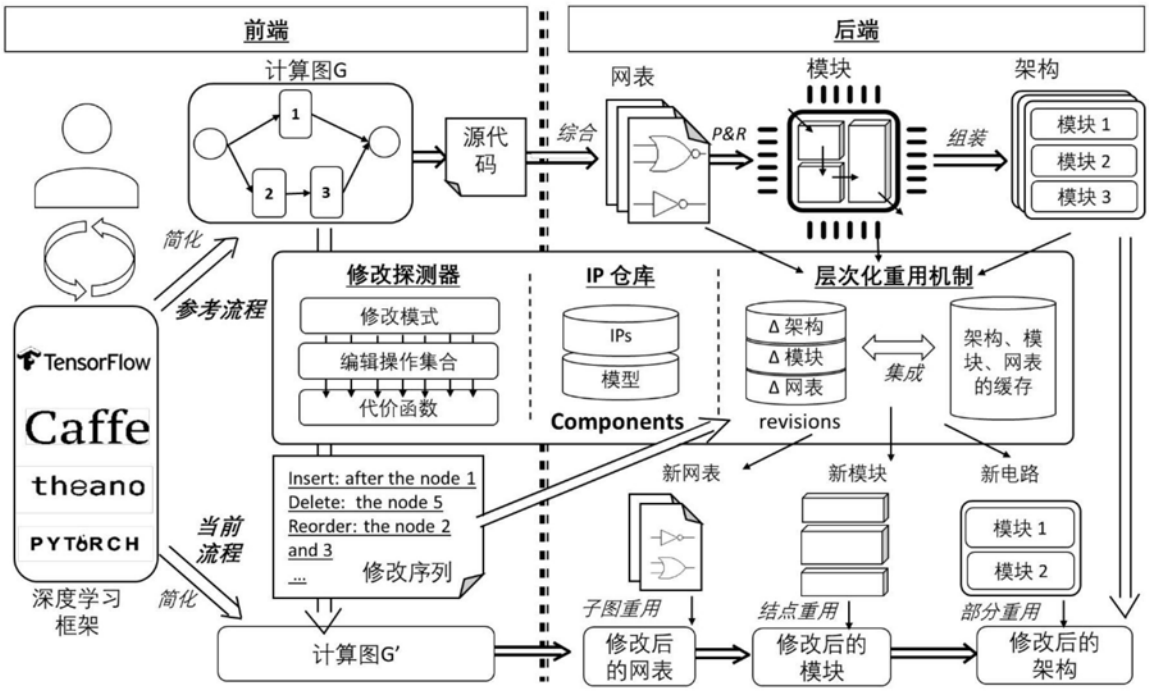

前端
后端
计算图G
源代码
网表
模块
架构
综合
P&R
组装
模块 1
模块 2
模块 3
简化
参考流程
修改探测器
修改模式
编辑操作集合
代价函数
IP 仓库
IPs
模型
Components
层次化重用机制
Δ架构
Δ模块
Δ网表
revisions
集成
架构、模块、网表的缓存
TensorFlow
Caffe
theano
PYTORCH
深度学习框架
当前流程
简化
Insert: after the node 1
Delete: the node 5
Reorder: the node 2 and 3
...
修改序列
新网表
新模块
新电路
子图重用
结点重用
部分重用
计算图G'
修改后的网表
修改后的模块
修改后的架构

图1

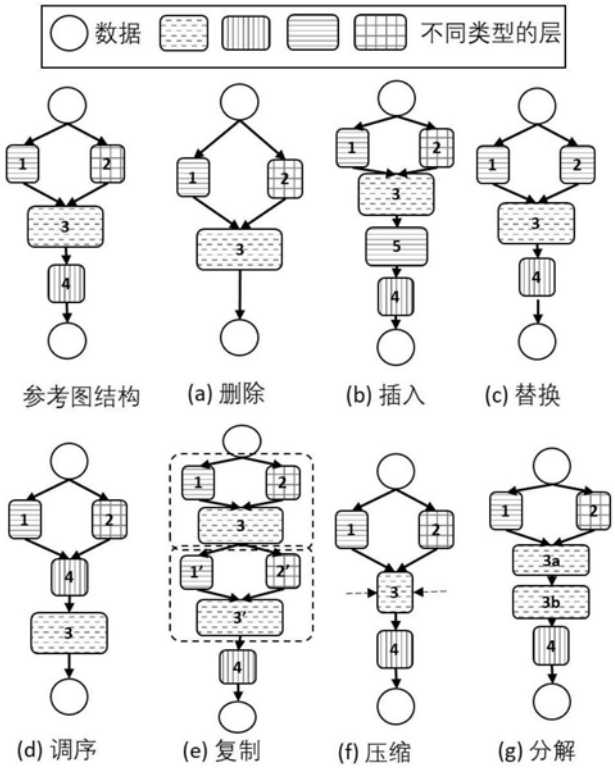

数据
不同类型的层
参考图结构
(a) 删除
(b) 插入
(c) 替换
(d) 调序
(e) 复制
(f) 压缩
(g) 分解

图2

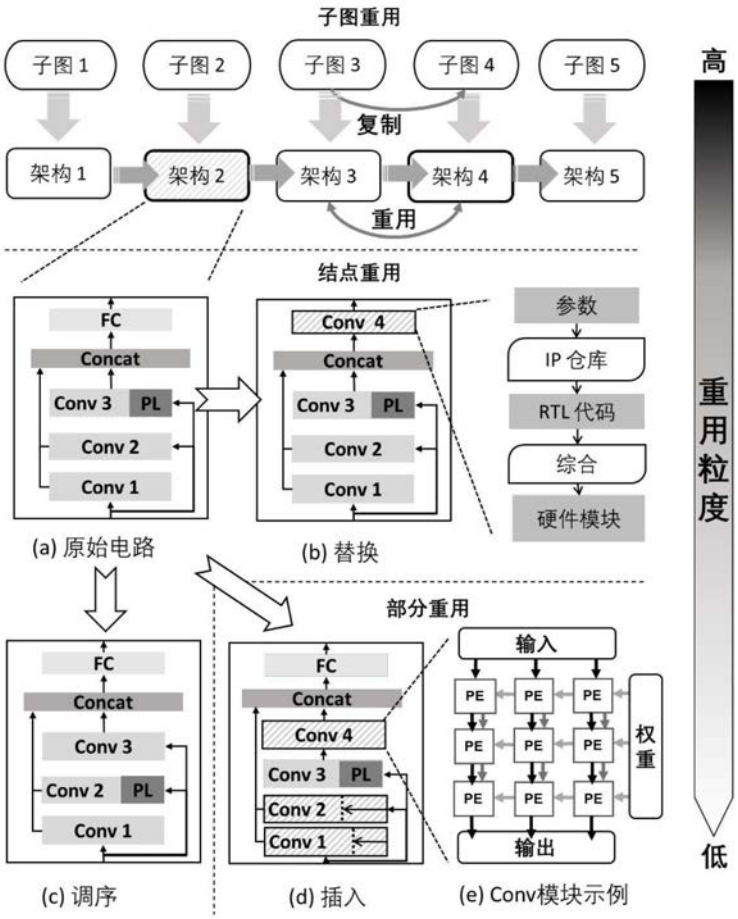

子图重用
子图1
子图2
子图3
子图4
子图5
复制
架构1
架构2
架构3
架构4
架构5
重用
结点重用
FC
Concat
Conv 3
PL
Conv 2
Conv 1
Conv 4
参数
IP 仓库
RTL 代码
综合
硬件模块
(a) 原始电路
(b) 替换
部分重用
输入
PE
权重
输出
(c) 调序
(d) 插入
(e) Conv模块示例
高
重用粒度
低

图3

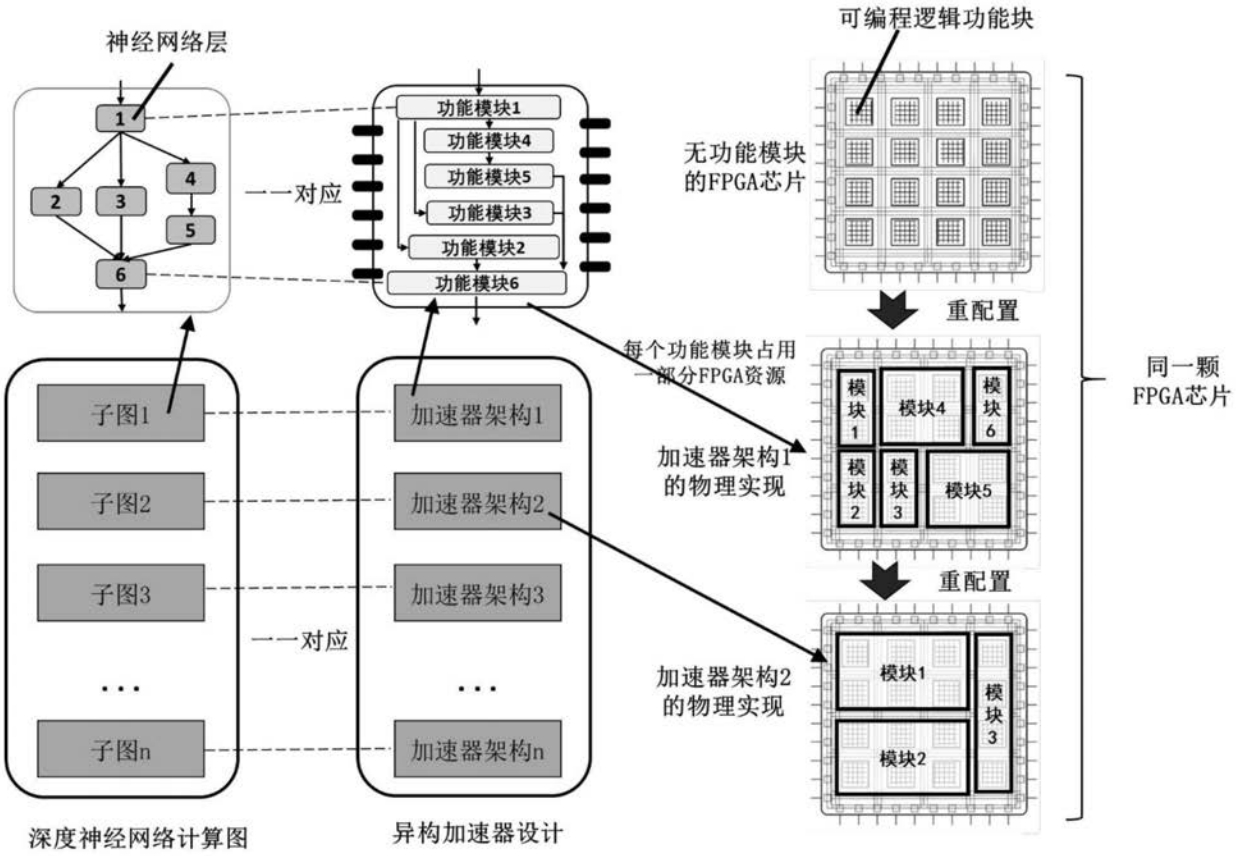

神经网络层
1
2
3
4
5
6
一一对应
功能模块1
功能模块4
功能模块5
功能模块3
功能模块2
功能模块6
子图1
子图2
子图3
...
子图n
一一对应
加速器架构1
加速器架构2
加速器架构3
...
加速器架构n
深度神经网络计算图
异构加速器设计
可编程逻辑功能块
无功能模块的FPGA芯片
重配置
每个功能模块占用一部分FPGA资源
加速器架构1的物理实现
模块1
模块4
模块6
模块2
模块3
模块5
重配置
加速器架构2的物理实现
模块1
模块3
模块2
同一颗FPGA芯片

图4