南京工業大學

2021 届毕业设计（论文）

题　　目:	深度神经网络硬件加速系统设计与优化
专　　业:	电子信息工程
学　　号:	1402170120
姓　　名:	王磊
指导老师:	朱艾春
起讫日期:	2020.12.01 ~ 2021.06.07

2021 年 6 月

深度神经网络硬件加速系统设计与优化

摘要

近年来，深度神经网络已经被证明在包括图像分类、目标检测和自然语言处理等任务上能够取得相当不错的效果。现如今，大量的应用程序都配备了与之相关的深度学习算法，但是对于手机、无人机等资源有限的嵌入式设备上，仅用软件方式加速深度神经网络已经不能满足日益增长的速度和功耗要求，如何利用硬件设计加速器已经成为学术领域的研究热点。

本文首先给出了深度神经网络的概述，并介绍了几种当下主流的针对深度神经网络的硬件加速系统设计方案以及实际应用。在考虑了不同加速器设计之间的相似性和差异性之后，本文将对英伟达开源的深度学习加速器框架 NVDLA 的设计思路进行概述。然后，本文基于 ZYNQ 7000 器件，在 FPGA 端实现了 NVDLA 设计，通过 AXI4 总线协议将 NVDLA 加速器挂载到 ARM A9 处理器，并为 ARM A9 处理器移植了 Ubuntu 16.04 操作系统，将加速器的驱动程序加载到了 Linux 内核。在软件设计的前端，使用神经网络编译器接受预训练的深度神经网络模型，做算子融合、内存重用等硬件无关的优化，并且结合 TensorRT 完成了深度神经网络的量化，在软件设计的后端，由 Runtime 自动调度加速器驱动程序推理深度神经网络，打通了硬件栈与软件栈，完成了软件与硬件的协同设计。

关键词： 深度神经网络　FPGA　NVDLA　硬件加速

Hardware Acceleration System Design and Optimization of Deep Neural Network

Abstract

In recent years, deep neural networks have been proven to achieve quite good results in tasks including image classification, target detection, and natural language processing. Nowadays, a large number of applications are equipped with related deep learning algorithms. However, for embedded devices with limited resources such as mobile phones and drones, only using software to accelerate deep neural networks can no longer meet the increasing speed and Power consumption requirements, how to use hardware design accelerators have become a research hotspot in the academic field.

This article first gives an overview of deep neural networks, and introduces several current mainstream hardware acceleration system design schemes and practical examples for deep neural networks. After considering the similarities and differences between different accelerator designs, this article will outline the design ideas of Nvidia's open source deep learning accelerator framework NVDLA. Then, based on the ZYNQ 7000 device, this article implements the NVDLA design on the FPGA side, mounts the NVDLA accelerator to the ARM processor through the AXI4 bus protocol, and transplants the Ubuntu 16.04 operating system for the ARM processor, and mounts the accelerator driver to The Linux kernel uses the Runtime automatic scheduling accelerator to infer the deep neural network. At the front end of the software design, the neural network compiler is used to accept the model trained by Caffe to perform hardware-independent optimizations such as operator fusion and memory reuse, and combined with TensorRT to complete the quantization of the neural network.

Keywords: Deep neural network; FPGA; NVDLA; Hardware speedup

目　录

摘要 …… I

Abstract …… III

第一章　绪论 …… 1

1.1 研究背景及国内外研究现状 …… 1

1.1.1 针对感知机和多层感知机等浅层神经网络研发的神经网络计算机/芯片 …… 1

1.1.2 针对第三次人工智能热潮的深度学习处理器 …… 2

1.2 研究意义及前景 …… 2

1.3 研究内容及结构安排 …… 2

第二章　相关技术介绍 …… 5

2.1 深度神经网络概述 …… 5

2.1.1 单层感知机模型 …… 5

2.1.2 多层感知机模型 …… 6

2.1.3 卷积神经网络概述 …… 8

2.1.4 LeNet5 介绍 …… 10

2.1.5 Resnet 18 网络结构 …… 10

2.2 FPGA 与 ZYNQ 器件 …… 11

2.3 硬件加速体系结构概述 …… 11

2.3.1 众核处理器 …… 11

2.3.2 GPU …… 12

2.3.3 Google TPU …… 13

2.3.4 DianNao 系列 …… 13

2.3.5 NVIDIA Deep Learning Accelerator …… 13

第三章　系统总体设计 …… 15

3.1 NVDLA …… 15

3.1.1 NVDLA 硬件结构分析 …… 15

3.1.2 NVDLA 自定义配置 …… 16
3.1.3 NVDLA 软件工具链 …… 17
3.2 验证平台 …… 17
3.3 AXI4 总线互联 …… 18
第四章 硬件设计实现 …… 19
4.1 Xilinx FPGA 设计套件 …… 19
4.2 NVDLA IP 生成描述 …… 19
4.2.1 基于 tmake 的 RTL 生成 …… 19
4.2.2 csb2apb …… 19
4.2.3 关闭 Clock Gating …… 19
4.2.4 IP Package 封装 AXI 总线 …… 19
4.3 Top Block Design 设计 …… 19
4.3.1 APB to AXI Bridge …… 19
4.3.2 AXI Smart Connect …… 19
4.3.3 ZYNQ 7000+ IP …… 19
4.4 综合 …… 19
4.4.1 资源利用率报告 …… 19
4.4.2 Timing 报告 …… 19
4.4.3 功耗报告 …… 19
4.5 实现 …… 19
4.6 Bitstream 与 hdf 文件生成 …… 19
第五章 软件设计实现 …… 21
5.1 NVDLA 软件工具链概述 …… 21
5.2 Petalinux 工具介绍 …… 21
5.3 TensorRT 与模型量化 …… 21
5.4 Ubuntu 16.04 根文件系统移植 …… 21
5.4.1 读取 Block Design 配置信息 …… 21
5.4.2 Linux 内核裁剪 …… 21
5.4.3 新增 Linux 设备树节点 …… 21

5.4.4 生成 Boot 和 Image 文件 …… 21
5.4.5 替换根文件系统 …… 21
5.5 KMD 内核程序挂载 …… 21
5.6 UMD 应用程序编译 …… 21
第六章 测试与分析 …… 23
6.1 LeNet5 性能分析 …… 23
6.2 ResNet18 性能分析 …… 23
6.3 基于 NVDLA 的图像风格迁移应用 …… 23
第七章 总结与展望 …… 25
参考文献 …… 27
致谢 …… 27

第一章　绪论

1.1　研究背景及国内外研究现状

总的来说，深度神经网络的硬件加速系统设计可以分为两个阶段，一是上世纪 50 年代第一次人工智能热潮开始到第二次人工智能热潮结束期间，针对感知机和多层感知机等浅层神经网络研发的神经网络计算机/芯片，二是针对第三次人工智能热潮中的深度神经网络设计的深度学习处理器。

1.1.1　针对感知机和多层感知机等浅层神经网络研发的神经网络计算机/芯片

在第一次人工智能热潮中，D. Hebb 提出了 Hebb 学习法则，在这之后不久的 1951 年，M. Minsky 研制出了国际上首台神经网络模拟器 SNARC；1957 年，F. Rosenblatt 提出了感知机模型，随后国际上首台基于感知机的神经网络计算机 Mark-I 就在 1960 年被研制出来，它可以连接到照相机上使用单层感知机完成简单的图像处理任务。

在第二次人工智能热潮中，著名的反向传播算法被提出使得神经网络的研究取得了一些重要突破。20 世纪 80 年代和 90 年代初，很多大公司、创业公司和研究机构都参与到了神经网络计算机/芯片的研制，包括 Intel 公司研发的 ETANN、1990 年发布的 CNAPS 、基于脉动阵列结构的 MANTRA I，以及 1997 年由中国科学院半导体研究所研发的预言神等。

然而，从当今深度学习技术发展的角度来看，这些早期的神经网络计算机/芯片有诸多缺陷，由于其只能处理很小规模的浅层神经网络算法，所以没有在工业实践中获得广泛应用。这一方面是因为浅层神经网络的应用比较局限，缺乏像当下的诸如目标检测、自然语言处理等领域的核心应用；另一方面，当时的主流集成电路工艺还是 1 微米工艺（今天的主流集成电路工艺已经达到 7 纳米），在一个芯片上只能放数量相当少的运算器，Intel 的 ETANN 芯片中仅能集成 64 个硬件神经元；最后，受限于当时的计算机体系结构技术还没有发展成熟，没有办法将大规模的算法神经元映射到少数的硬件神经元上。

而第二次人工智能热潮也随着日本的五代机计划失败而结束。基于上述原因，从 20 世纪 90 年代开始，神经网络计算机/芯片的创业公司纷纷破产，大公司也裁减掉了相关的研究部门，各个国家暂停了这方面的科研资助，但这些瓶颈在处于第三次人工智能热潮的今天都得到了解决和改善，于是深度神经网络的硬件加速系统设计进入了第二个阶段。

1.1.2 针对第三次人工智能热潮的深度学习处理器

2006 年，深度学习技术由 G. Hinton、Y. Bengio 和 Y. LeCun 等人的推动而兴起，于是有了第三次人工智能热潮。而深度学习处理器也在这种环境下重新焕发了生机，在 2008 年，中国科学院计算技术研究所的陈云霁、陈天石等人开始了人工智能和芯片设计的交叉研究，之后来自法国 Inria 的 O. Temam 也参与到可项目中。在这些人的推动下，国际上第一个深度学习处理器架构 DianNao 于 2013 年被设计出来。和第一阶段设计的神经网络计算机/芯片不同，DianNao 架构不会受到网络规模的限制，可以灵活、高效地处理上百层的深度学习神经网络，并且显得更对于通用的 CPU，DianNao 可以去的两个以上数量级的能效优势。2014 年，陈云霁等人在 DianNao 架构上改进，设计出了国际上首个多核的深度学习处理器架构 DaDianNao ，以及机器学习处理器架构 PuDianNao 。进一步，中国科学院计算技术研究所提出了国际上首个深度学习指令集 Cambricon 。在这之后，首款深度学习处理器芯片“寒武纪 1 号”问世，目前寒武纪系列处理器已经应用于超过一亿台嵌入式设备中。

1.2 研究意义及前景

深度神经网络已经被证明在包括图像分类、目标检测和自然语言处理等任务上能够取得相当不错的效果。而随着其层数和神经元数量以及突触的不断增长，CPU 和 GPU 等传统体系已经很难满足神经网络增长的速度和需求。例如 2016 年，Google 公司研发的 AlphaGo 与李世石进行围棋对弈时使用了 1202 个 CPU 以及 176 个 GPU ，在这场比赛中 AlphaGo 每盘棋需要消耗上千美元的电费，而作为人类的李世石一盘棋的功耗仅需要 20 瓦，从此可以看出传统芯片的速度和能效难以满足大规模深度学习应用的需求。如今，大量的应用程序都配备了与之相关的深度学习算法，但是对于手机、无人机等资源有限的嵌入式设备上，仅用软件方式加速深度神经网络已经不能满足日益增长的速度和功耗要求，基于深度神经网络的智能应用需要广泛普及的趋势使高性能、低功耗的深度学习处理器研发呼之欲出，如何利用硬件设计加速器已经成为学术领域的研究热点。

1.3 研究内容及结构安排

硬件设计层面，本文主要在 FPGA 平台上设计实现了 NVDLA 深度学习加速器，并将其接口封装为 AXI4 总线协议与双核 ARM A9 处理器互联。软件设计层面，在前端，使用神经网络编译器接受预训练的神经网络模型，并做硬件无关优化。在后端，将驱动程序挂

载到了 Ubuntu 操作系统上，由 Runtime 自动调度推理。最后，分析了所设计的硬件加速器的性能。

本文共有七个章节，论文结构安排如下：

第一章：

第二章：

第三章：

第四章：

第五章：

第六章：

第七章：

第二章　相关技术介绍

由于本硬件加速系统设包含了软件与硬件的协同设计，所以本章将对涉及到的关键技术，包括深度神经网络算法、ZYNQ 器件做基本的介绍。如第一章节所述，本章还会对不同类型的硬件加速体系结构进行基本概述，介绍对应的应用案例。

2.1　深度神经网络概述

2.1.1　单层感知机模型

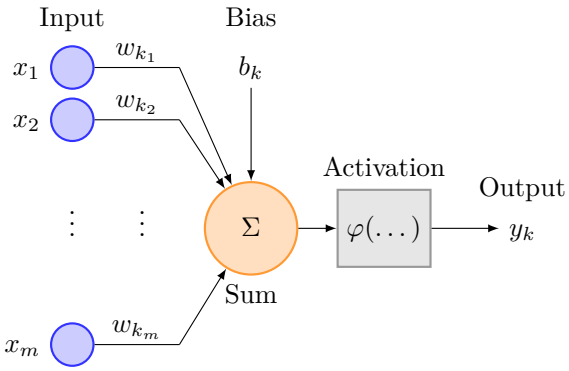

图 2-1 神经元模型

一个神经元的模型如图 **??**所示，该模型也被称为单层感知机，由输入信号、权值、偏置、加法器和激活函数共同构成，其中 w_{kj} 下标的含义为：k 表示第 k 个神经元；j 表示第 j 个输入。因此，w_{kj} 表示第 k 个神经元的第 j 个输入对应的权值。单个神经元的数学公式如公式 (**??**) 所示：

$$\begin{cases} u_k = \sum_{j=1}^{m} w_{kj} x_j \\ v_k = u_k + b_k \\ y_k = \varphi(v_k) \end{cases} \tag{2-1}$$

其中 x_j 表示第 k 个神经元的第 j 个输入，w_{kj} 表示第 k 个神经元的第 j 个输入对应的权值，b_k 表示第 k 个神经元的偏置，$\varphi(\dots)$ 为激活函数，y_k 为该神经元的输出。

感知器模型于 1958 年，由美国心理学家 Frank Rosenblatt 提出，其中激活函数采用的一般是符号函数，如公式 (**??**)所示：

$$o = sgn(x_1 w_1 + x_2 * w_2 + b) \tag{2-2}$$

进一步，为了简化问题分析本质，我们假设神经元只有两个输入: x_1 和 x_2 ，则模型的公式进一步简化为:

$$\begin{cases} 1, & x_1 w_1 + x_2 * w_2 + b >= 0 \\ -1, & x_1 w_1 + x_2 * w_2 + b < 0 \end{cases} \tag{2-3}$$

如果把 o 当作因变量、x_1 和 x_2 当作自变量，对于分界

$$x_1 w_1 + x_2 w_2 + b = 0 \tag{2-4}$$

可以抽象成三维空间里的一个分割面，能够对该面上下方的点进行分类，则该感知机能完成的任务是用简单的线性分类任务，比如可以完成逻辑 “与” 与逻辑 “或” 的分类，如表 **??**所示，在这里，第三维度只有 1 和-1 两个值，分别使用实心点和空心点来表征，这样就可以在二维平面上将问题可视化:

表 2-1 逻辑真值表

逻辑与			**逻辑或**			**逻辑异或**		
x_1	x_2	o	x_1	x_2	o	x_1	x_2	o
0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	0	1	1
1	0	0	1	0	1	1	0	1
1	1	1	1	1	1	1	1	0

但是对于非线性问题，如异或问题，如图 **??**所示，单层感知机没有办法找到一条直线能够完成分类任务。

2.1.2　多层感知机模型

多层感知机模型是在单层感知机模型的基础上引入了一层或多层隐藏层，在图 **??**所示的多层感知机中，输入和输出的神经元个数分别为 4 和 1，中间的隐藏层包含了 5 个神经元，隐藏层中的神经元和输入层中各个输入完全连接，输出层中的神经元和隐藏层中的各个神经元也完全连接。因此，多层感知机中的隐藏层和输出层都是全连接层。

但是不难发现，即便再添加更多的隐藏层，多层感知机的设计依然只能与仅含输出层的单层神经网络等价。上述问题的根源在于全连接层只是对数据做仿射变换（affine

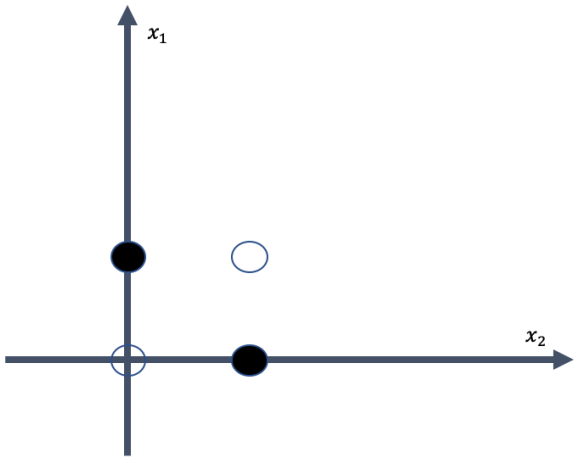

图 2-2 异或平面

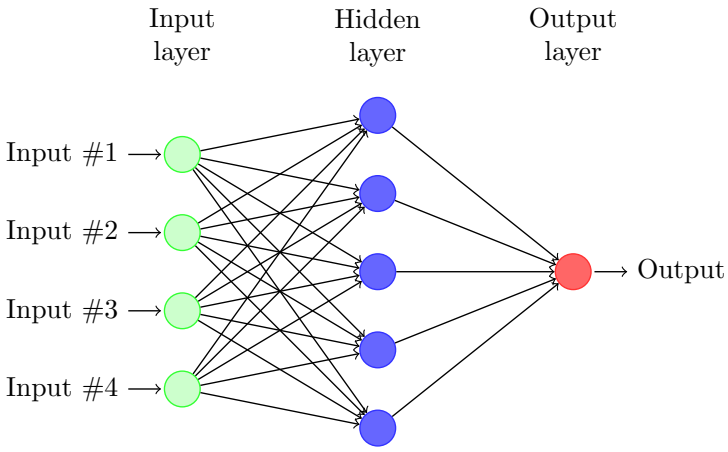

图 2-3 前向神经网络结构

transformation），而多个仿射变换的叠加仍然是一个仿射变换。解决问题的一个方法是引入非线性变换，例如对隐藏变量使用按元素运算的非线性函数进行变换，然后再作为下一个全连接层的输入。这个非线性函数被称为激活函数（activation function），常用的激活函数包括了 Sigmoid、Relu、Tanh 等。

理论上，深度神经网络，及多层感知机已经可以拟合任意的函数[?]。

2.1.3　卷积神经网络概述

卷积神经网络（Convolutional Neural Network, CNN）是目前最常用的深度神经网络，对于大型图像处理有出色表现，卷积神经网络一般由卷积层、非线性激活函数、池化层组成。

1. 卷积层（Convolution Layer）的本质是输入图像与权重矩阵的乘积累加运算 (Multiply Accumulate, MAC)，如图 **??**所示卷积核在输入特征图像上滑动，用来匹配局部的细节。一般来说，随着卷积的层数逐渐加深，卷积核的数目也会随之增加，所表征的细节也会更加抽象。

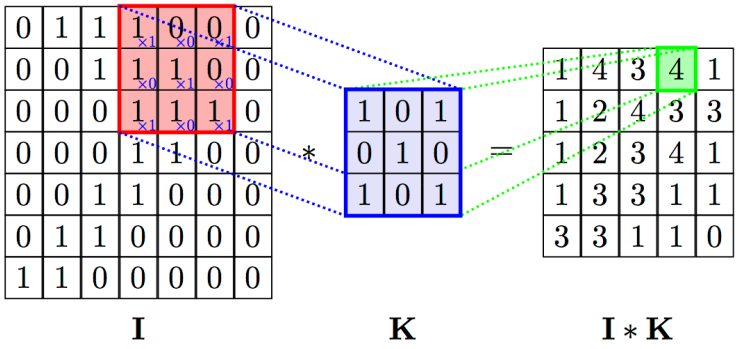

图 2-4 卷积运算

2. 激活函数（Activation Function）是非线性单元，它的存在给深度神经网络体系增加了非线性元素，是深度神经网络能够拟合任意函数的基础。比较重要且常用的激活函数有 ReLU、Sigmoid 和 Tanh 。此外，其余常用的激活函数还有 PReLU, ELU 等，他们的函数图像与取值范围如图 **??**所示：

3. 池化层（Pooling Layer）的作用是对特征图进行下采样，从而可以达到增加感受野、增加卷积神经网络运算的平移不变性、降低优化难度，减少神经网络参数的目的。典型的池化层 Max Pooling 如图 **??**所示，他会选择被池化矩阵选内最大的值作为输出得到新的特征图。Average Pooling 选取池化矩阵内的平均值，是另一种常用的池化层。

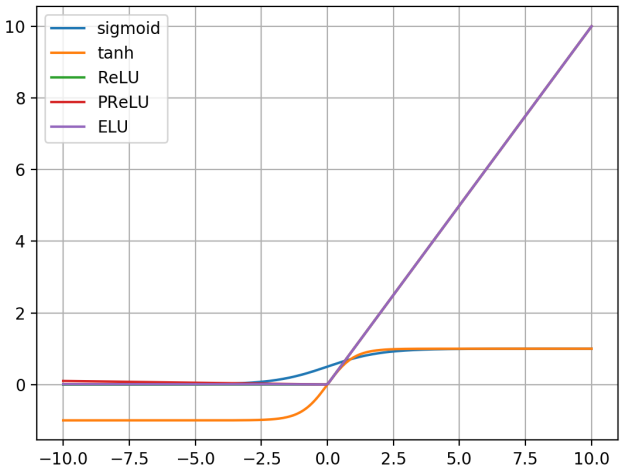

图 2-5 常见的激活函数图像

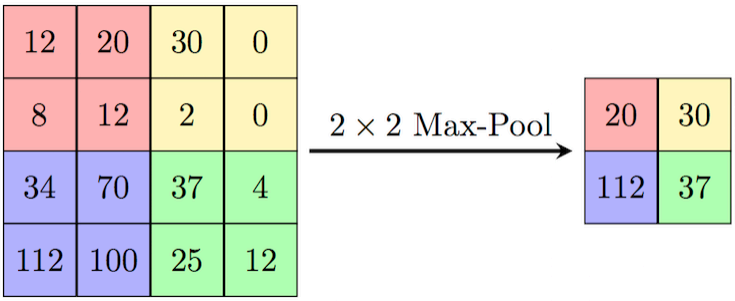

图 2-6 Max Pooling

2.1.4　LeNet5 介绍

手写字体识别模型 LeNet5 诞生于 1994 年，是最早的卷积神经网络之一。

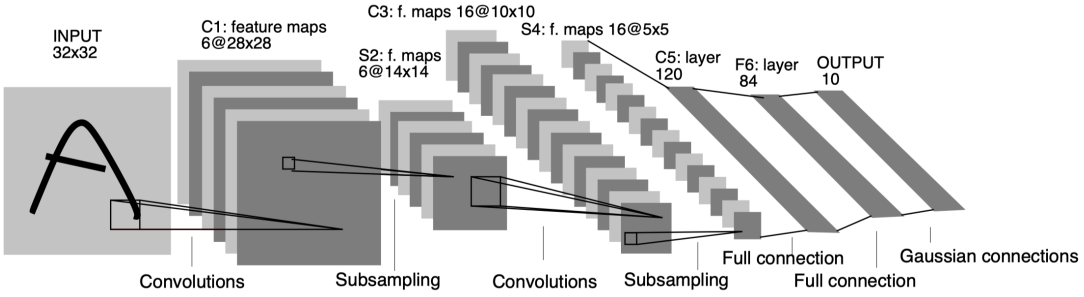

图 2-7 LeNet-5 网络结构

通过巧妙的设计，利用卷积、参数共享、池化等操作提取特征，LeNet5 避免了大量的计算成本，最后再使用全连接神经网络进行分类识别，该网络也是最近大量神经网络架构的起点，LeNet5 的网络结构如图 **??**所示。

2.1.5　Resnet 18 网络结构

纵观神经网络的发展脉络，我们可以看到研究者一直在不断尝试各种技术以加深神经网络的层数。从 8 层的 AlexNet，到十九层的 VGG，再到 22 层的 GoogLeNet，可以说，技术一直在进步。但是在突破神经网络深度的问题上真正最具有颠覆性的技术还是来自 ResNet。

为了避免随着神经网络的深度加深从而带来的退化问题，ResNet 采用了一种不同于常规卷积神经网络的基础结构，其基本单元如图 **??**所示，增加了从输入到输出的直连通道。其卷积拟合的是输出与输入的差，及残差，所以这种结构又被称为残差结构。残差网络的响应小于常规网络[?]。残差网络的优点是对数据波动更加灵敏，更容易求的最优解，能够改善深层网络的训练。

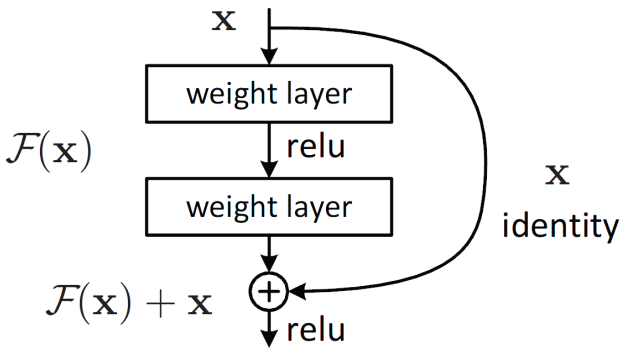

图 2-8 Resnet 残差结构

2.2 FPGA 与 ZYNQ 器件

现场可编程逻辑门阵列（Field Programmable Gate Array, FPGA）, 是一种半定制的集成电路（Integrated Circuit, IC）芯片。FPGA 包含了一组可编程逻辑门阵列（Programmable Logic Blocks, PLB）以及可重配置的互联层次结构，通过这种结构可以将 PLB 连接在一起。FPGA 可以通过重编程，实现不同逻辑特性，从而实现了可重构计算。因此理论上 FPGA 可以实现当前 CPU 上的所有算法，但是具体算法实现的效果会受制于 FPGA 的可用资源、时钟频率以及输入输出的带宽。

ZYNQ 器件是 Xilinx 公司研发的以 ARM 为主导，FPGA 为辅的嵌入式系统结构，其拥有集成度高的开发环境，支持高层次综合、Vivado 硬件逻辑设计、Simulink 协同设计、SDK 软件设计。对于一个不熟悉 FPGA 的软件工程师来说，也完全可以把 ZYNQ 器件当前简单的双核 ARM 来使用，利用 SDK 编写软件程序，如果软件调试过程中发现某些算法的速度太慢，这时候就可以使用 Verilog 硬件描述语言，或者使用高层次综合为这部分算法设计硬件加速器，ARM 处理器与加速器之间通过 AXI 标准总线协议进行通信，Xilinx 提供了若干免费的加速 IP 供用户使用，由于有了 ARM 处理器，我们甚至可以在 ZYNQ 器件上搭建 SOC 降低开发的难度。

2.3 硬件加速体系结构概述

在过去，深度神经网络算法往往被运行在通用体系结构里，如众核处理器与 GPU 设备，但是这些设备的特点是运行功耗大，并且会带来极高的内存使用率。当谈及深度神经网络，往往最让我们头疼的是训练过程，这个步骤一般在使用 GPU 完成加速，然后将预训练好的模型部署在目标平台，当平台对功耗和运算能力有要求的时候，传统的通用处理器的体系结构就不适合了，于是领域专用的，针对深度神经网络的硬件加速系统设计呼之欲出。本小节首先介绍两种典型的通用硬件加速体系结构，然后介绍三种面向深度神经网络运算的领域专用运算。

2.3.1 众核处理器

众核处理器（Manycore Processor）由成百上千的简单分立的处理器核心组成。与传统的多核处理器所不同的是，其设计牺牲了单核运行的性能来换取多核处理时的吞吐量，减少多核处理时的能量损耗。典型的使用众核处理器设计是 Kalray MPPA-256 架构。

如图 **??**所示，使用众核处理器进行深度神经网络运算时，每个处理器核心都可以贡献出一部分性能进行神经网络的运算，之后通过 Router 决定在物理总线上数据传输的方向。

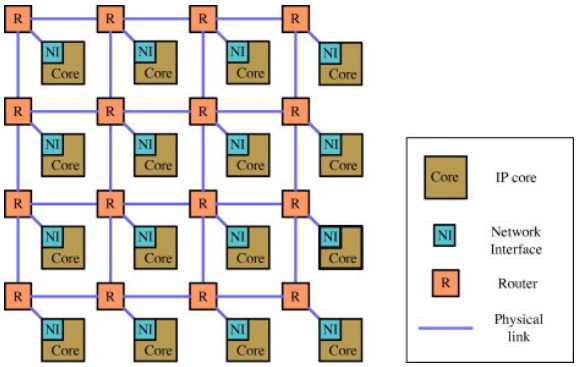

图 2-9 Net work on Manycore Processor

众核处理器是一种通用处理器，这意味着他具有一定的灵活性，除了可以运用在神经网络运算上，还可以用在各种各样的领域，尤其是那些有高并行度要求的场景，例如 HPC。但如果仅针对深度神经网络的推理，通用处理器本身便会存在一些多余、不合适的设计。

2.3.2　GPU

GPU 也可以看作是一种特殊的，针对向量运算的众核处理器。起初 GPU 被设计出来的目的是为了提高计算机对图像与视频的处理能力，但是当下，GPU 设备被广泛应用在深度学习应用的训练和推理加速中。并且对于 GPU 的调用，NVIDIA 还提供了软件支持，即 CUDA，方便了用户手动便携并行算子。

VOLTA 架构是 NVIDIA 推出的新一代的 GPU 设计架构，基于该架构的显卡被广泛运用在深度学习领域。其中，TESLA V100 是第一款被设计出来专门针对深度神经网络训练和推理的 GPU 设备（如图 **??**）。

图 2-10 TESLA V100

2.3.3 Google TPU

张量处理单元（Tensor Processing Unit）是 Google 牵头设计的一款为深度神经网络运算定制的 ASIC 器件，事实上其也取得了非常瞩目的性能，2015 年发布的 29nm 工艺的 TPUv1 版本，能够运行在 700Mhz 的频率，但功率只有 28W ~40W。

2.3.4 DianNao 系列

为了能够实现大规模深度学习网络的专用硬件执行。2014 年，中国科学院计算技术研究所研究团队提出了 DianNao 架构。DianNao 这篇文章是率先探索机器学习加速设计的先驱文章之一，开创了专用处理器实现深度学习的先河。这篇文章在 65nm 工艺，0.98Ghz 的频率，面积为 3.02mm2 的 ASIC 芯片上针对机器学习算法（DNN，CNN）实现了一个高性能的 Diannao 处理器架构，相比于 128bit 2GHz 的 4 发射 SIMD 处理器，达到了 117.87x 的加速比，21.08x 能耗比。

2.3.5 NVIDIA Deep Learning Accelerator

NVDLA 是 NVIDIA 公司开源出来的深度学习加速器框架，其结构与 DianNao 类似。DianNao 系列芯片每个周期最多能够计算 16 个神经元，NVDLA 能够计算 64 个，而 TPU 则能够计算 256 个。

与前两个领域专用的硬件加速体系结构不同，NVDLA 完全开源，其不仅提供了 Verilog 硬件描述语言、CMOD 硬件仿真程序，还为硬件加速器设计了神经网络编译器与运行时，自上而下打通了硬件栈与软件栈，基本能够实现端到端的推理。虽然 NVDLA 自 2018 年以来已经没有人维护，但其系统设计思路仍然具有学习意义和指导意义。

因此，有关 NVDLA 的内容将在下一章节具体介绍。

第三章　系统总体设计

本章将介绍系统的总体设计结构，包括对 NVDLA 的详细介绍，选用的 FPGA 开发板卡资源介绍，NVDLA 与 ARM 互联的解决方案。

3.1　NVDLA

NVDLA 是 NVIDIA 公司在 2017 年年末公布并且开源的，遗憾的是该项目自公布完成一年后便草草停止了维护。NVDLA 是一款模块化、可配置的神经网络推理加速器框架。通过核心 MAC 阵列、与查找表逻辑，NVDLA 可以支持深度神经网络算子，已经支持的有：

- 卷积运算
- 激活函数
- 池化
- 归一化

实际上，由于 NVDLA 是完全开源的，完全可以自己手动添加相关算子的硬件逻辑。

3.1.1　NVDLA 硬件结构分析

前文中提到，NVDLA 具有模块化的属性，这很有助于我们理解加速器的工作流程，如图 **??**所示，NVDLA 的各个模块可以独立、同时工作。

- 卷积引擎配置了卷积 Buffer，在使能卷积运算的时候，需要先通过寄存器配置给出 Input Image 与 Weights 在内存上的地址，然后将数据缓存到 Buffer 中来进行运算，减少访存的开销。NVDLA 支持两种卷积模式：
  - 直接卷积，及标准的卷积，可以通过 MAC 阵列并行加速。
  - Winograd 快速卷积，从算法层面通过共用权重来减少运算量。

卷积引擎的核心是 MAC 阵列，该阵列的大小是可配置的，本文使用阵列大小为 8x8 。卷积的结果不可以直接写回内存，必须经过 SDP，及激活函数层，其他引擎则没有这个限制。

- 除去卷积引擎之外，NVDLA 一共还有五个引擎，他们是负责完成激活函数的 SDP 引擎，负责完成池化操作的 PDP 引擎，负责完成 LRN 操作的 CDP 引擎，做图形 Reshape 的 RUBIK 引擎，最后，NVDLA 设计还提供了一个 BDMA 引擎，用来在 DRAM 和高速存储之间搬移数据。

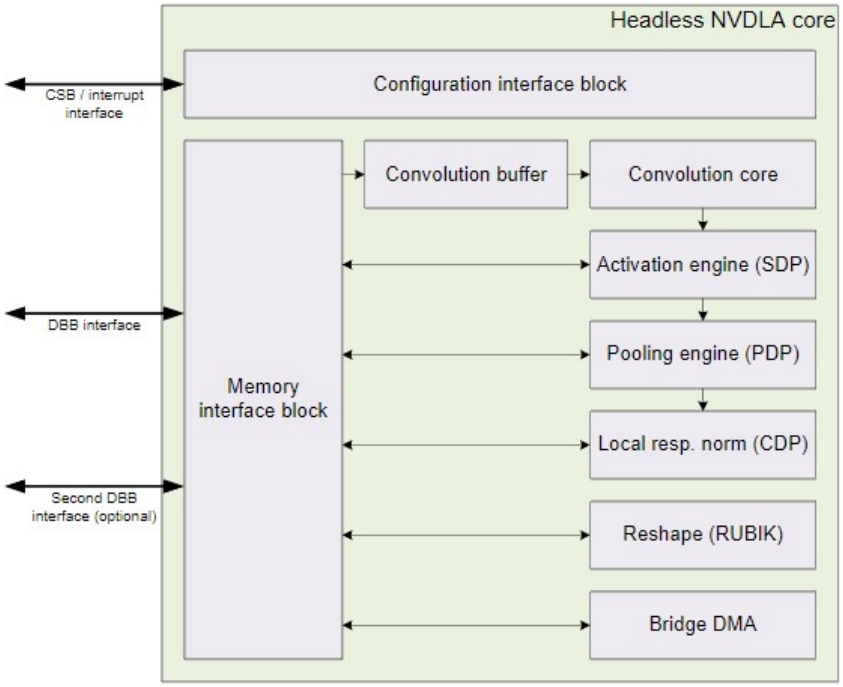

图 3-1 NVDLA 体系结构

NVDLA 在外侧暴露出四个接口，它们分别是 CSB 总线，主存储接口、高速存储接口、中断接口。

- 寄存器配置总线（Configuration Space Bus Interface，CSB），用来读写 NVDLA 的寄存器，对于 CSB 总线，我们不甚讲解，其在读写地址的时候需要做移位来压缩指令大小，NVDLA 提供了 csb2apb 转换电路，在读写的时候使用该电路包裹即可利用 APB 总线协议读写 NVDLA 的寄存器。
- 主存储接口（Data Backbone interface，DBB），用来访存，该接口使用的协议为 AXI4 总线协议，可以挂载在 AXI4 接口的存储控制器上，在本文中，我们将其挂载到片上的 DDR 存储，与 ARM 处理器共享内存。
- 高速存储接口（high-bandwidth interface），可以外接第二块 SRAM 作为辅存储，存储中间数据，降低访存瓶颈。
- 中断接口（Interrupt interface），NVDLA 不仅支持通过寄存器查询的方式查询任务是否完成，也支持产生硬件中断，方便我们编写驱动程序。

3.1.2　NVDLA 自定义配置

在本设计中，我们使用官方提供的 small 配置，即最精简的 NVDLA 配置，该配置有如下特点：

1. 最小化 MAC 阵列，本设计将 NVDLA 的 MAC 阵列大小配置为 8 * 8。

2. 关闭高速存储接口，只使用主存储接口。

3. 仅支持 INT8 格式的 MAC 运算。

4. 不支持权重压缩。

5. 不支持 WINOGRAD 快速卷积。

6. 不支持 Reshape 特征图。

3.1.3 NVDLA 软件工具链

如图 **??**，NVDLA 的软件工具链分为 Compiler 和 Runtime 两个部分。

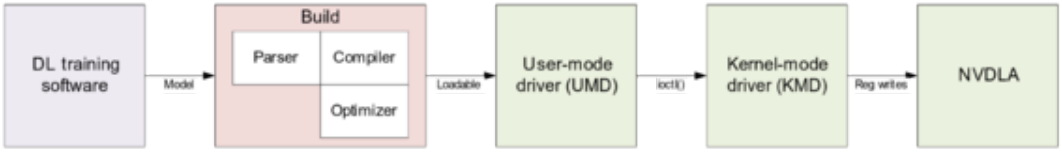

图 3-2 NVDLA 软件栈

Compiler 与硬件无关，可以在主机运行。它可以接受其他深度神经网络训练框架训练完成的模型，完成一些硬件无关的优化，例如算子融合、数据重用等，并且使用 FlatBuffers 将处理结果封装成 Loadable 数据流文件。

Runtime 与硬件紧密结合，其负责接受 Compiler 生成的 Loadable 文件，调度加速器程序，自动分配内存，自动配置寄存器，自动处理中断。而 Runtime 内部又被划分为两个部分，UMD 和 KMD 吗，分别对应了 Linux 的用户态驱动与内核驱动。

- 用户态驱动程序（USER MODE DRIVER，UMD）由 C++ 编写，负责解析 Loadable 文件，分配内存，并将上下文封装成一个 Task，最后发送给底层的 KMD 程序执行。
- 内核态驱动程序（e KERNEL MODE DRIVER）由 C 语言编写，是 NVDLA 固件的驱动程序，他是真正与 NVDLA 进行交互的部分。

软件栈设计的详细内容，我们将在软件实现这一章详细讨论。

3.2 验证平台

NVDLA 加速器设计需要挂载到 Linux 内核上，所以本设计需要一款通用处理器作为主控，并且移植 Linux 操作系统。并且，考虑到本设计采用的 small 配置需要 7 万以上的查找表资源。则可供选型的器件型号有：

1. ZYNQ 7000 系列，该芯片集成了一块双核 32 位 ARM A9 处理器作为主控制器，而 FPGA 侧的资源随芯片的型号不同而不同，想要实现 NVDLA 至少需要使用到 ZYNQ 7045

器件。

2. ZYNQ MPSoc 系列，该芯片集成了一块多核的 64 位的 ARM A53 处理器作为主控制器，性能相较于 ZYNQ 7000 器件强，官方开发板卡的价格在 2 万左右。

3. 纯 FPGA 逻辑器件，如官方板卡 VCU118，有足够大的 LUT 资源，可以将 RISC-V 处理器与 NVDLA 一起实现，并在 RISC-V 处理器上移植 Riscv-Linux，但缺点是板卡价值近十万元，十分昂贵，由于使用了 RISC-V 处理器，开发周期较长。

综合以上，本设计选用的器件型号为 XC7Z045-2FFG900I，开发板卡为如图 **??**的第三方板卡，价格为三千多元人民币。

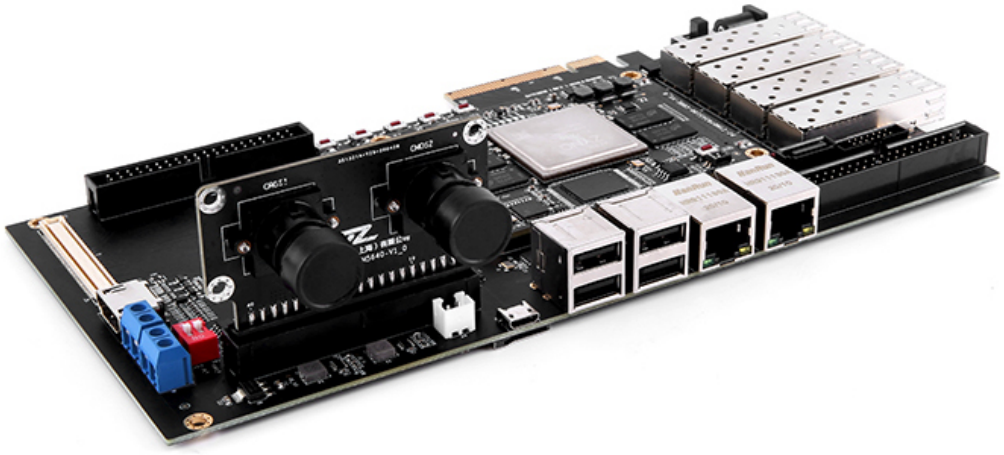

图 3-3 ZYNQ 7045 开发板

3.3　AXI4 总线互联

在 Xilinx 的设计工具中提供的 IP 核大多使用 AXI4 总线来实现各级之间的逻辑控制和数据传输。前文中提到，NVDLA 亦是使用 AXI4 总线访问存储，无疑极大方便了我们的设计，但是 NVDLA 的控制总线协议不是 AXI4 协议，在本设计中，其通过 csb2apb 电路将 CSB 协议转换为 APB 总线协议，在 Vivado 设计中，我们使用 Xilinx 官方提供给的 APB2AXI Bridge IP 将 APB 总线再转换为 AXI4 总线挂载到 ARM 处理器上。

第四章　硬件设计实现

4.1　Xilinx FPGA 设计套件

4.2　NVDLA IP 生成描述

4.2.1　基于 tmake 的 RTL 生成

4.2.2　csb2apb

4.2.3　关闭 Clock Gating

4.2.4　IP Package 封装 AXI 总线

4.3　Top Block Design 设计

4.3.1　APB to AXI Bridge

4.3.2　AXI Smart Connect

4.3.3　ZYNQ 7000+ IP

4.4　综合

4.4.1　资源利用率报告

4.4.2　Timing 报告

4.4.3　功耗报告

4.5　实现

4.6　Bitstream 与 hdf 文件生成

第五章　软件设计实现

5.1　NVDLA 软件工具链概述

5.2　Petalinux 工具介绍

5.3　TensorRT 与模型量化

5.4　Ubuntu 16.04 根文件系统移植

5.4.1　读取 Block Design 配置信息

5.4.2　Linux 内核裁剪

5.4.3　新增 Linux 设备树节点

5.4.4　生成 Boot 和 Image 文件

5.4.5　替换根文件系统

5.5　KMD 内核程序挂载

5.6　UMD 应用程序编译

第六章　测试与分析

6.1　LeNet5 性能分析

对比上学期用 PYNQ 实现的加速器做分析比对，逐步提高工作频率做分析比对。

6.2　ResNet18 性能分析

6.3　基于 NVDLA 的图像风格迁移应用

如果时间够的话，这里可以做一个图像风格迁移的 DEMO，用 Caffe 训练一个 VGG16，然后在屏幕上显示一下。

第七章　总结与展望

致　谢