硕士学位论文

深度学习的硬件实现与优化技术研究

RESEARCH ON HARDWARE IMPLEMENTATION AND OPTIMIZATION TECHNOLOGY OF DEEP LEARNING

林楗军

哈尔滨工业大学

2017 年 6 月

国内图书分类号：TP183　　学校代码：10213
国际图书分类号：621.3　　密级：公开

工程硕士学位论文

深度学习的硬件实现与优化技术研究

硕士研究生：林楗军
导　　师：朱　敏
申请学位：工程硕士
学　　科：电气工程
所在单位：电气工程及自动化学院
答辩日期：2017 年 6 月
授予学位单位：哈尔滨工业大学

Classified Index: TP183
U.D.C: 621.3

Dissertation for the Master Degree in Engineering

RESEARCH ON HARDWARE IMPLEMENTATION AND OPTIMIZATION TECHNOLOGY OF DEEP LEARNING

Candidate: Lin Jianjun

Supervisor: Zhu Min

Academic Degree Applied for: Master of Engineering

Speciality: Electrical Engineering

Affiliation: School of Electrical Engineering and Automation

Date of Defence: June, 2017

Degree-Conferring-Institution: Harbin Institute of Technology

摘 要

近年来，随着人工智能的兴起，以深度学习为代表的新型智能算法在机器视觉、图像处理、模式识别等多个工程应用领域得到成功应用。但是，在工业大数据的冲击下，传统的软件实现方式无法满足实际工程低成本、高时效、高容错率的需求，因此急需寻求新的解决方案。现场可编程门阵列 FPGA 作为一种常用硬件开发平台，拥有大规模的分布式硬件资源，并且具有开发周期短、功耗低、性能好等特点，非常适合计算密集型的深度学习算法的实现。本文以 FPGA 为硬件开发平台，展开深度学习的硬件化实现与优化技术研究，主要研究内容如下：

首先，深度学习硬件实现总体方案设计。详细分析深度学习的理论基础知识，并以卷积神经网络为例，进行网络的拓扑结构和功能特点研究，给出本文硬件实现的具体网络拓扑。根据网络拓扑的结构特点，进行系统的总体方案设计，将网络拓扑映射到具体的硬件电路。

其次，完成算法硬件移植的优化技术与架构设计。选择 FPGA 作为本文实现的硬件移植平台。结合本文实现低功耗、高效率深度学习算法的目标，分别对硬件移植的优化技术进行深入研究，并应用优化技术完成对卷积神经网络从粗粒度到细粒度的并行架构设计。

然后，完成基于 FPGA 的卷积神经网络设计与实现。以 FPGA 为硬件开发平台，完成卷积神经网络的整体架构设计。根据卷积神经网络的结构特点，完成设计各功能电路模块，包括卷积运算模块、抽样运算模块、激活函数模块。本文设计乒乓缓存结构，优化数据传输结构和数据缓存单元。用仿真软件 Modelsim 分别验证各模块功能正确性。

最后，搭建系统整体实验平台。依据现有的实验条件，配置网络结构与参数，设计“FPGA+CPU”的异构体系，完成卷积神经网络的硬件固化。以手写数字识别为具体应用，完成软件和硬件的对比实验。通过大量的实验统计，结果表明本文设计的基于 FPGA 的卷积神经网络功能完整，性能优异。

关键词：深度学习；神经网络；硬件固化；FPGA；并行架构；优化技术

Abstract

In recent years, with the rise of artificial intelligence, the new intelligent algorithm represented by deep learning has been successfully applied in many engineering applications, such as machine vision, image processing and pattern recognition, etc. However, under the impact of industrial big data, the traditional software can’t meet the requirements of low cost, high real-time and high error-tolerant, so it’s very urgent to find a new solution. As a common hardware platform, Field Programmable Gate Array (FPGA) has distributed hardware resources on a large scale, and its advantages lies in the features of short development cycle, low power consumption and good performance, etc. Therefore, FPGA is very suitable for the implementation of the compute-intensive deep learning algorithm. In this paper, FPGA is used as the hardware development platform to study the hardware implementation and optimization technology of deep learning. This paper makes contributions as follows:

Firstly, the overall scheme of deep learning hardware implementation is designed. The basic theoretical knowledge of deep learning is analyzed in detail. This paper takes convolution neural network (CNN) as a typical example of deep learning, the topological structure and functional characteristics of the network are studied. And the specific network topology of hardware implementation is given. According to the structural characteristics of network topology, the overall scheme of the system is carried out, and the network topology is mapped to the specific hardware circuit.

Secondly, optimization technology and architecture design are completed before the algorithm hardware implementation. Select FPGA as the hardware transplant platform in this paper. And the optimization techniques of hardware implementation are studied deeply to realize the aim of low-power and high efficiency deep learning algorithms. The parallel architecture design of convolution neural network from coarse granularity to fine granularity is completed by optimization technique.

Thirdly, design and optimization of CNN based on FPGA are carried out. The overall architecture design of CNN is completed based on FPGA. And then, according to the structure characteristics of CNN, each functional circuit module is designed, including convolution calculation module, sampling calculation module and activation function module. This paper designs ping-pang cache structure to optimize the data transmission structure and data cache unit. Each model is verified the function correctness by simulation software Modelsim.

Finally, the whole system experimental platform is built. According to the existing experimental conditions, the network structure and parameters are configured, and the heterogeneous system of “FPGA + CPU” is designed to complete the hardware solidification of CNN. This paper completes the contrast experiment of software and hardware implementation with the handwritten numeral recognition as the specific application. The results show that the CNN based on FPGA designed in this paper is fully operational, and has an excellent performance through a large number of experimental statistics.

Keywords: deep learning, neural network, hardware solidification, FPGA, parallel architecture, optimization technology

目　录

第 1 章 绪　论

1.1 课题背景及研究的目的和意义

本课题是顺应“工业 4.0”新时代的发展需要，着眼于人工智能的研究热点，以深度学习的低功耗硬件加速实现技术为研究方向展开研究。得益于互联网技术和存储技术的快速发展，当今社会正大步朝着智能化、智慧化的方向迈进[1]。与此同时，在社会生活和工业领域的方方面面都孕育着形形色色多样化的数据信息，并以爆炸式的方式持续增长。在大数据时代中，如何将这些海量的数据信息进行精细化、精准化管理，挖掘数据更本质、更抽象的高层次认知形态，已经成为数据挖掘和机器学习等相关科学领域面临的一个重大难题和研究热点。

经过多年的摸索和研究，在数据挖掘领域中已经形成了多种成熟的相关技术，比如决策树法、神经网络法、统计分析法和模型法等等[1,2]。但是面临大数据的挑战，这些成熟的技术有时也会显得力不从心。近年来，深度学习在解决高层次抽象的认知问题上取得了众多显著的研究成果[3]。2016 年 3 月，以深度学习为基础开发的 AlphaGo 程序战胜世界围棋顶级棋手李世石，更是点燃了业界对人工智能和深度学习的研究热潮[4]。深度学习诞生于人类对人工神经网络的研究，两者相互交叉又有所区别。深度学习是一类多层人工神经网络，通过搭建特定的信息处理模型，模拟人脑对信息处理过程，完成对信息特征的提取和识别功能。2006 年，Hinton 教授构造出的一种快速逐层非标记式的训练方式，更是大大提升了多层网络的自我无干扰学习能力，Hinton 教授也因此被世人尊称为深度学习之父[5]。随后，国内外众多学者和科研人员纷纷开展对深度学习的研究，并在语音识别、机器视觉、场景应用等多个领域中取得了突破性的进展，从而进一步促进了深度学习的发展与壮大。深度学习的网络深度和每层网络的节点数，很大程度上决定着该网络的信息表达能力。随着网络深度和节点数的增加，势必会加剧整个网络的训练和计算任务量。而目前通常采用传统的软件方式串行执行整个网络。基于逻辑处理和事物处理优化的通用处理器，不仅无法充分挖掘网络内在的并行特性，而且系统在实际应用上存在一定的脆弱性和不稳定性，这些缺陷对于追求低成本、高时效、高容错率的工业应用场合是不可容忍的。

在另一方面，随着集成电路、半导体技术和制造工艺的快速发展，使得数以百万计的门电路设计成的电子分系统，乃至是整个电子系统集成在一块芯片

上，完成系统的多种功能任务[6]。随着工业界一系列实际工程问题的呈现，越来越多的研究者开始着眼于多种硬件开发平台，设计深度学习的硬件实现方案，包括数字电路、模拟电路以及数模混合电路。FPGA(Field-Programmable Gate Array)巧妙地融合了专用集成电路 ASIC 和通用处理器 CPU 的最大优势，不仅具备类似软件编程的可反复编程，修改电路的灵活特性，而且拥有无可比拟的并行分布式存储计算资源，打破了原来的顺序执行的模式。另外，在一些高端的 FPGA 芯片内部，还集成了高速数字信号处理单元和大量存储单元，提高芯片的运算性能，因此，基于 FPGA 开发平台可以快速实现深度学习算法开发的芯片架构设计，节约开发周期和设计成本。

在信息高速发展的大数据时代，我们每天面临的信息是千变万化，错综复杂的。如何在海量的信息中能实时、准确、有效地提取更高层次、更本质的内涵信息，已经成为当今社会的一个研究热潮。本文以此为研究背景，探索和研究深度学习网络的硬件加速实现技术方案，为数据挖掘、机器学习和人工智能等相关领域研究提供一定的理论依据和技术支持。

1.2 国内外研究现状及分析

深度学习作为最近快速发展起来的一类新型智能算法，是一门集人工智能、模式识别、数据挖掘、信号处理等多学科于一身的综合性交叉学科。深度学习是目前属于一个非常接近人工智能的研究方向[7]。近年来，国内外各大科研机构、高科技企业纷纷投入了大量的人力、物力、财力，进行大量深度学习的应用性科学研究，取得了不少为人惊叹得阶段性成果。

1.2.1 深度学习的发展概况

深度学习追根溯源至少可以回归到早期人类对人工神经网络的研究，它是一类具有自我记忆与学习能力的深层网络。早在上个世纪 40 年代，美国学者 McCulloh 和 Pitts[8]总结了前人的研究成果，率先提出了神经元最原始的数学模型——MP 模型，用来描述简单神经元的信息处理过程，该理论也被认为是研究神经网络的起源。经过几年后的世纪中期，Hebb 在基于大脑神经细胞研究的基础上提出了 Hebb 规则[9]，认为相邻神经元的连接权值是不确定且可调的，从而进一步完善了神经网络的理论基础。1957 年，Rosenblatt 跨越式地搭建了一种三层网络结构的神经网络模型[10]，并尝试获取类似生物的自我感知和适应能力。随后，神经网络理论被许多学者不断改进与完善，并在海内外出现了不少成功的应用案例。如 1989 年，Yann LeCun 等[11]人设计的深度神经网络，成功应用于手写邮政编码上，但是需要耗费大量的时间去训练网络，以便获取准确

的网络参数。另外，深度学习在天气预报、驾驶行为预测、人脸定位、数学分类、图像重构等多个领域也取得了成功应用，但是同样存在缺乏有效的网络训练方法的问题，加之受限于当时对神经网络实现技术的不足，以及受其他更加简单模型的挑战与冲击，人工神经网络没有得到更广泛的应用与研究。

直到 2006 年，时任加拿大多伦多教授的 Hinton 教授同他的学生在世界顶级学术期刊《Science》上共同发表了一篇文章，提出了深度学习的概念和模型的逐层训练方法[12]，打破了传统人工神经网络的发展瓶颈。Hinton 教授也因此被世人称为深度学习的缔造者。从此，深度学习的研究在国外率先进入了高速发展时期，尤其是以 Geoffrey E. Hinton、Yann LeCun 和 Yoshua Bengio 三位深度学习顶尖学者为首的科研团队和国外著名高校成为了深度学习的研究重镇。2010 年，美国国防部联合美国 NEC 研究院、纽约大学和斯坦福大学进行深度学习方面研究，旨在高效处理文档情报，挖掘更具有指导行动的隐秘信息，该项目也首次成为美国国防部 DARPA 计划的资助项目[13]。2011 年，微软和谷歌率先将深度学习技术应用到语音识别领域，降低了原先近五分之一到三分之一的误识率，取得了多年来巨大的研究成果[14]。2012 年，深度学习在图像处理上也取得了众多重大进展，如谷歌秘密成立的猫脸实验室，经过充分训练后，开始学会自动识别猫脸图像[15]；在 ImageNet 竞赛中，提出的 AlexNet 网络结构突破性地将图像分类错误率从 26%降低到 15%[16]，从此之后更多更优秀的深层神经网络被不断提出。同年《纽约时报》报道称，深度学习技术还被制药公司应用于药物性能预测，取到了世界上最好的预测效果[17]。DeepMind 公司以深度学习为技术支撑，设计的智能围棋程序 AlphaGo[18]在 2015 年和 2016 年分别以 5：0 和 4：1 的总比分完胜欧洲围棋冠军樊麾和围棋奇才、世界围棋冠军李世石。在 2017 年初，被认为是 AlphaGo 的升级版 Master[19]与中日韩多名围棋高手连战 60 局，无一败绩，引起了世界范围内的轰动。此外，深度学习技术还被用于在线电影点播系统 Netflix 和网络电子商务 Amazon[20]，分析用户行为习惯，实现用户电影和商品推荐。

深度学习自 2006 年被提出以来，国外展开了大规模的深入研究，在语音识别、图像处理、场景应用等众多领域得到了丰厚的应用性成功。而国内对深度学习的研究起步稍晚，目前主要集中在 BAT 为首的知名 IT 行业、各大科研机构以及国内著名高校之间。

2013 年，百度在国内首先成立了深度学习研究院，并成功引进斯坦福教授吴恩达，将深度学习技术成功应用于数十款产品中。2014 年，阿里相关负责人在阿里巴巴举办的数据峰会上宣称，已积攒了超过 100PB 已处理过的数据可用于深度学习技术的研究，包括建立具有自动推荐功能的广告和搜索系统。到 2015

年，腾讯基本完成了由 CPU 和 GPU 集群为基础的深度学习平台 Mariana 的基本框架[21]。2017 年 1 月，腾讯云 FPGA 联合团队推出了国内首款高性能异构计算基础设施——FPGA 云服务器，使运行成本降低到原先的 40%。2016 年 10 月，华为宣布与加州伯克利分校合作成立诺亚方舟实验室，深入研究包括深度学习在内的人工智能基础性理论。海康威视研究院基于深度学习技术，设计的一套高效深度卷积神经网络方法，在 ImageNet 2016 竞赛的场景分类和目标检测中分获第一和第二，并将研究成果应用于车辆检测、车牌识别、智能机器人、人脸识别等产品中。2017 年 3 月，海康威视研究院基于深度学习技术研发的 OCR(Optical Character Recognition，图像中文字识别)技术，刷新了 ICDAR 竞赛数据集的全球最好成绩。

总的来说，在大数据时代的促使下，深度学习自提出以来，受到了国内外各界的热情追捧。但是仍然存在理论基础、工程实现、数学建模等重大难题。

1.2.2 深度学习实现技术现状及分析

深度学习的工程实现方式是推进深度学习进步的一个重要因素。依赖于大规模数据和高性能计算技术的发展，深度学习在工程领域中取得多种实现方式的应用。

目前深度学习主要以软件的方式实现。根据不同的应用场合，谷歌、Facebook、微软等巨头公司和一些专家学者开发了针对深度学习的开源软件框架，如 TensorFlow、Torch、Caffe、Theano、Deeplearning4j 等[22,23]。这些开源软件编程简洁，可移植性强，与用户交流友好，架构设计灵活，甚至支持 CPU 集群，实现并行和分布式计算。近年来，图像处理器 GPU 的急速发展，使得像 TensorFlow、Torch 等开源软件，开始支持在 GPU 上运行，甚至支持“CPU+GPU”、“GPGPU”集群的异构框架模式，在工程实践中也得到了很好的性能效果。如 Hinton 教授在 ImageNet 2012 竞赛中采用 GPU 完成了对卷积神经网络的训练与识别，大幅度地降低了图像误识别率[24]。国防科技大学使用 GPU 的 CUDA 编程框架完成深度信念网络的加速训练，相比 CPU，实现了 22 倍本地预训练和 33 倍全局训练的加速效果[25]。然而，随着大数据时代的深入，深度学习算法对运算能力需求越来越大，CPU 串行的指令形式使其更多的芯片面积用于复杂的控制流和 Cache 缓存，执行深度学习的效率不高。GPU 虽然在深度学习的运算效率上性能优越，但是其高昂的价格、超大的运行功耗和漫长的程序开发周期使在大规模部署运行平台上带来了诸多问题。

早在 1960 年就有学者涉足研究硬件方式实现神经网络，那时斯坦福大学的 Widrow 基于硬件电路设计实现了人工神经网络[26]。1974 年，Louis Gilstrap 和

Roger Barron 联合开发第一块神经元芯片[27]。1990 年初期，LeCun 和他的同事在贝尔实验室开始探索开发卷积神经网络的专用芯片，推出的 ANNA 芯片峰值速度每秒能处理 40 亿次运算[28,29]。近年来，随着高性能 FPGA 的快速发展，极低的单位能耗和灵活的架构设计吸引了更多的研究者的目光，开始着眼于基于 FPGA 设计硬件加速平台，极大地推动了深度学习的硬件实现方式的研究。文献[30]应用卷积神经网络，设计了基于 FPGA 设计移动机器人视频数据处理系统，但是该平台只实现了单个卷积运算，多个卷积核之间的运算是串行执行，无法充分挖掘网络并行特性。文献[31]提出的基于 FPGA 的深度卷积神经网络加速设计，在 MNIST 识别任务中，相比于 CPU/MATLAB 有很好的加速效果，然而其注重网络整体结构设计，忽略了模块化分类设计，缺乏一定的灵活性。文献[32]以优化存储资源为目标设计了一套卷积神经网络加速器，用于嵌入式计算平台，该方案大大降低了设计面积和能量损耗，但是对存储带宽需求较大。文献[33]研究深度学习算法的数据并行特性，将深度计算任务分散到多个分布式计算集群节点中，并用 6 个 FPGA 组成的计算云，实现深度学习的权值加速计算，但是整套系统需要增加完善的协调控制程序，统筹安排集群计算云，确保数据分配的正确性。在国内也吸引了许多研究者开展基于 FPGA 实现深度学习的实现方案，文献[34]以不同的优化目标，研究了基于 FPGA 的卷积神经网络的硬件实现方案。

总体来说，国内外都正如火如荼地开展对深度学习的各方面研究。面临大数据时代的挑战，将深度学习的软件算法移植到以 FPGA 为主的硬件平台上，提出可靠高效的硬件加速方案，已是一种新颖不可逆转的研究热潮。本论文以此为研究着手点，开始探索深度学习的硬件实现方案。

1.3 本文的主要研究内容

本文以深度学习研究热潮为契机，以卷积神经网络为典型代表，深入研究网络内在的并行架构，结合 FPGA 分布式硬件结构特点，将卷积神经网络模块化分类设计，通过调用设计的功能模块电路，实现低功耗、高效率的卷积神经网络预测通路。以此为基础，进行 MNIST 手写数字识别实验效果验证和分析。本文具体研究如下：

第 1 章为绪论部分。简要介绍了论文的研究背景与意义、深度学习的发展概况以及国内外对深度学习领域的研究现状分析。以此为基础，提出本文的研究方向和研究目标。

第 2 章为深度学习硬件实现总体方案设计。深入分析深度学习的理论基础，以卷积神经网络作为深度学习的典型代表，对其拓扑结构和功能特点进行详细

研究，并提出本文硬件实现的具体拓扑结构。最后，对卷积神经网络硬件实现进行系统总体方案设计，将网络拓扑结构映射到具体的硬件电路。

第 3 章是算法硬件优化技术与架构设计研究。结合本文设计内容，选择适合的硬件实现平台。同时，分析讨论软件算法的硬件化移植可能面临的优化技术问题，结合网络结构特点，对卷积神经网络进行并行架构设计，为网络硬件实现做好准备工作。

第 4 章是基于 FPGA 的网络设计与实现。以 FPGA 为硬件平台，首先设计硬件实现的整体架构，并进行功能化模块划分。其次，针对卷积神经网络计算密集性高、重复性大的卷积运算，按数据流驱动方式不同，分别设计了 Z 型卷积运算模块和树型卷积运算模块。然后完成抽样运算模块、激活函数设计，数据缓存结构优化的任务。并通过仿真软件对各个模块进行功能和时序仿真，验证其可靠性和准确性。

第 5 章是结合以上设计实现的硬件电路，搭建完整的“FPGA+CPU”异构系统，采用手写数字 MINST 测试库数据作为实验数据，进行软硬件对比实验，验证功能的正确性，并从实现效果，资源利用，功耗等方面分析性能。

第 2 章 深度学习硬件实现总体方案设计

在本章中，首先从深度学习的理论基础着手，详细分析了深度学习的基本概念、构成原理、网络结构和应用过程。然后以卷积神经网络作为深度学习的典型网络代表，图像识别为其应用背景，对卷积神经网络的拓扑结构和功能特点进行详细研究，包括卷积层、抽样层、全连接层和激活函数，并提出本文网络硬件实现的具体拓扑结构。最后，对卷积神经网络硬件实现进行系统总体方案设计，将网络拓扑结构映射到具体的硬件电路中。

2.1 深度学习理论分析

深度学习是集人工智能、模式识别、数据挖掘、信号处理等学科于一身的交叉综合领域[35]，起源并继承于人工神经网络。

在人类思维学界，专家普遍认为人类大脑的思维可以细分为抽象思维、形象思维和灵感思维[36]。深度学习，正是模拟生物的中枢神经系统（特别是人类大脑）的第二类思维方式建立起来的数学模型，是一个大规模非线性动力学系统。深度学习是由大规模简单细微的加工工厂——神经元相互协作、联接，构成的复杂动态网络，并具备并行协同处理、分布存储信息、自适应学习等天然特性[37]。

从控制论的角度看，神经元作为最小的信息处理单元，描述的是多路输入刺激产生的单路输出综合反应的过程，其中多路输入可以来自神经网络的其他神经元，单路输出又可以作为下一层神经元的输入信号[38]。不同神经元之间的连接都存在一个可变的权重值，代表着相连两个神经元的连接强度，并随着神经元的活动，突触强度也随之改变。当某一个神经元接收到的多路刺激信号，若综合叠加获得的刺激效果超过了其兴奋阈值，那么该单元被激活，否则一直处于休眠状态。图 2-1 为神经元模型。

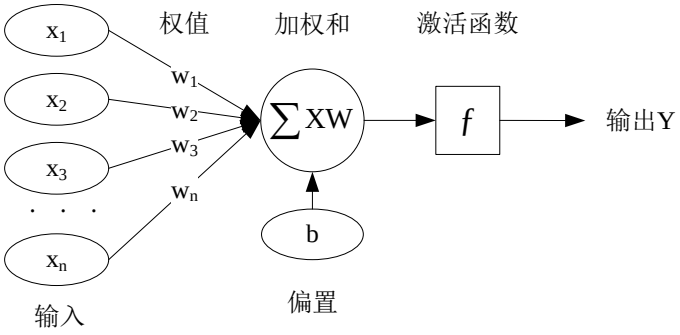


图 2-1 神经元模型

从模型本质上来讲，单个神经元其实就是一个简单运算单元，承载了从输入信号 X 到输出结果 Y 的映射关系。假设神经元接收到 n 个输入，记为 $(x_1, x_2, x_3, ..., x_n)$，对应的 n 个权重值记为 $(w_1, w_2, w_3, ..., w_n)$，用状态 z 表示一个神经元接收到的信号叠加和，输出 Y 表示该神经元的活性值。单个神经元计算过程可表示为：

$$z = \sum_{i=1}^{n} x_i w_i + b \tag{2-1}$$

$$Y = f(z) \tag{2-2}$$

式中，b 是偏置项，常用来调整神经元的活性，f 为激活函数。

为了模拟生物神经元具有的被激活的特性，增强网络的表达能力，引入了连续非线性激活函数，如常见的有斜面函数、阀值函数、Sigmoid 型函数、ReLU 函数等[39]。在传统的神经网络中，最常用的激活函数是以单极性 Logistic 函数 $\sigma(x)$ 和双极性 Tanh 函数 $\mathrm{Tanh}(x)$ 为代表的 Sigmoid 型函数，这类激活函数能有效起到控制信号增益效果的作用。近年来，ReLU 函数 $\mathrm{ReLU}(x)$ 由于其计算简单，导数收敛快，单侧抑制性强的特点，使其开始被广泛应用。公式 2-3，2-4，2-5 分别为三种激活函数的数学表达式。

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2-3}$$

$$\mathrm{Tanh}(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2-4}$$

$$\mathrm{ReLU}(x) = \max(0, x) \tag{2-5}$$

图 2-2 分别为三种函数的曲线图。从输入与输出的激励关系来看，Sigmoid 型函数是一类连续光滑曲线，而 Tanh 函数实际上是 Logistic 函数的一个变形，Logistic 函数是把输入的连续实值压缩到在[0,1]区间内，Tanh 函数的输出则是压缩在[-1,1]区间，ReLU 函数可以看成是一类分段线性函数，当输入变量小于 0 时，输出值都是 0，当输入变量大于等于 0 时，输出等于输入。

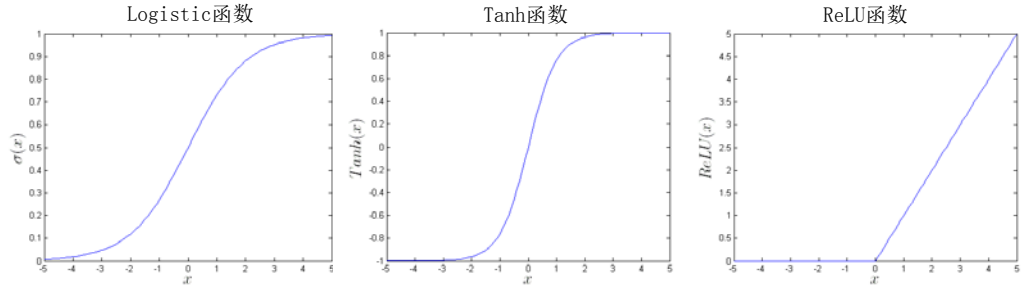
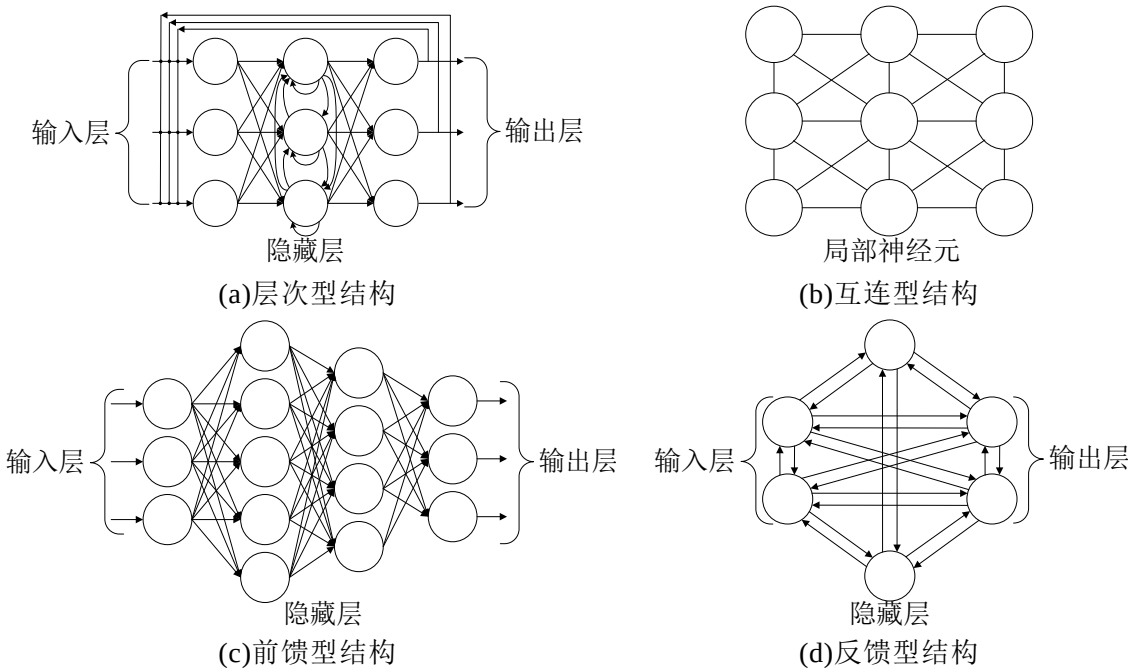


图 2-2 激活函数曲线

深度学习是由大规模的神经元相连构成，多元化的连接方式和拓扑结构也使其表现出强大的功能和优越的特性。但是局限于物理资源和计算性能上的困难，无法达到生物神经数以亿计的网络规模，通常是由一定规模大小的神经元按特定的规律连接构成网络[40]。根据不同的分类标准，深度学习网络有不同的网络拓扑结构，如常见的有层次型结构、互联型结构、前馈型结构、反馈型结构[41,42]等，结构示意图如图 2-3 所示。根据不同的应用场景，采用的数据类型不同，搭建的网络拓扑结构和学习方式也不尽相同，目前比较常用的学习方式有监督式学习、非监督式学习、半监督式学习以及强化学习等[43-46]。



(a)层次型结构　(b)互连型结构

(c)前馈型结构　(d)反馈型结构

图 2-3 深度学习拓扑结构

深度学习是一类数学建模技术的集合，在各个领域中，关于深度学习都有着很多独到的定义和精确的描述。总的归纳概括，深度学习是一种模拟生物对信息学习机制的深层次非线性网络结构，来实现对复杂结构函数逼近，以发现输入数据内涵的分布式特征表示[47]。从字面表述上分析不难看出，与传统的浅层人工神经网络相比，深度学习有两方面强调之处：一方面是深度，即模型结构的深度；另一方面是学习，即信息特征的学习。

深度，主要表现在网络的结构层数。在神经网络中，除输入层外，每一层网络都具有信息处理能力，随着网络深度的增加，整个网络的计算能力得到提高，也意味着具有了更强大的函数模拟能力和更深入的特征表示能力的先天条件。以图像识别来说，像素级的特征几乎是没有任何价值的[48]，因此第一层网络一般是从小范围局部感知域开始提取低层特征，依次组合形成高层特征。例

如第一个隐藏层提取到的是低层抽象，如边缘定向或角点特征，第二个隐藏层是将上一层提取到的边缘定向或角点特征再次组合，进行抽象迭代，得到形状特征，第三个隐藏层又将形状特征组合，提取类似物体或语义特征的高级抽象，以此类推，反复迭代和抽象。通过增加深度，网络可以提取更加抽象和复杂的特征来对事物加以区分，提高事物的识别和分类能力。有研究者已设计使用几十层甚至上百层深度的神经网络进行识别，取得了很好的性能效果。

深度学习采用深度非线性的分层式网络结构，使其对复杂函数的表达具有较好的泛化能力。如要表达一个复杂函数 $\cos(\log(\mathrm{sqrt}(\exp(\sin^3(x)))))$，若采用传统的单层或浅层神经网络拟合，其网络结构和参数均难以确定，而且受限于训练样本和计算单元；而采用深度的多层式网络结构，可用多个简单初等函数 x^3，$\sin(x)$，$\exp(x)$，$\mathrm{sqrt}(x)$，$\log(x)$，$\cos(x)$ 组合构造而成，网络结构简单明确，参数少，易于求解。复杂函数的分层表达方式如图 2-4 所示。

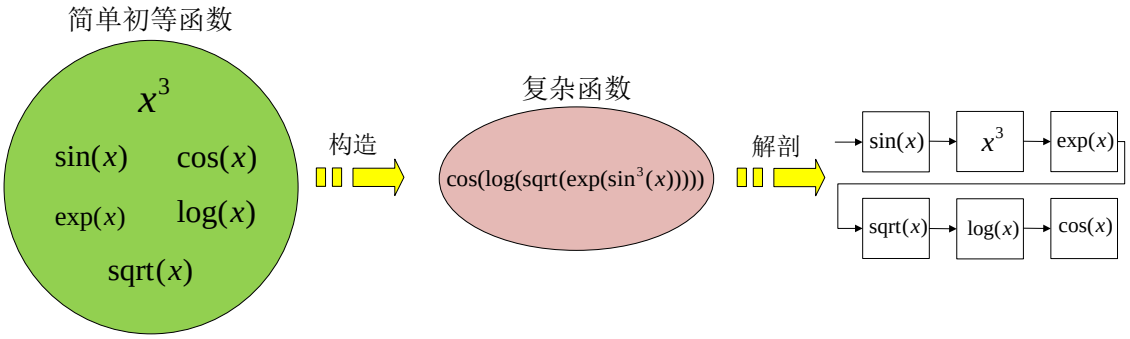


图 2-4 复杂函数的分层表达

学习，即结合大规模训练样本数据进行能力学习，得到识别与分类目标事物的能力。整个学习过程是一种完全的端到端学习方式，从输入到输出完全自动化无干扰学习，不需要显著的人为干预。通常情况下，深度学习采用“逐层优化”的学习方式[49]，即进行逐层特征变换，将样本的原始特征自上而下的监督式学习或自下而上的非监督式学习，具体涉及的算法，因网络拓扑结构不同而不同。网络的学习效果，在很大程度上受限于样本数据的规模大小、网络的复杂程度和训练算法的优化程度。

对于深度学习来说，其基本思想就是对级联层依次迭代的过程，即每一层的计算结果作为下一层的激励输入，逐层递进，以实现对原始输入数据的分层表达[50]。假设一个具有 n 层隐藏层的分层系统 S，各层的输出结果记做 S_i，其中 $i=1,2,...,n$，系统的原始输入数据为 I，输出数据为 O，中间每一层都是对本层输入数据进行刨洗，去除冗余或无效数据，提取有效数据，最终使输入与输出的误差尽可能小。整个表达过程可以表示成：$I => S_1 => S_2 => ... => S_n => O$，

S_i表示每一层对原始输入数据的分级抽象表达。图 2-5 为人脸识别示意图[50]。每一张人脸图像都是由像素组成的，而像素级的特征属于初级特征，为原始输入数据，以目前的技术对人脸识别是没有价值的；通过第一层抽象组合，形成边缘基特征，然而不同的人脸，甚至不同的物体训练得到的边缘基是非常相似的；边缘基特征作为下一层的输入，继续组合形成人脸组成部分，如眼睛、鼻子、耳朵等；以此递归地向上进行特征组合与表达，最终形成人脸模型。

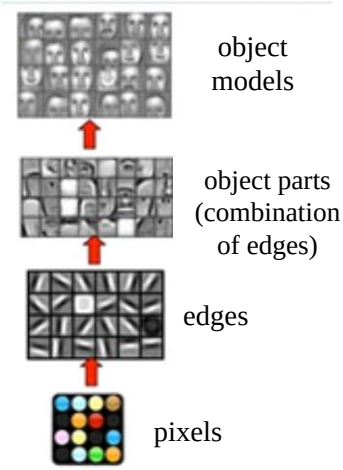


图 2-5 人脸识别示意图

无论是图像识别还是语音识别，深度学习的识别或分类过程都是相似的。其本质就是将低维、无法识别或分类的小块局部特征进行组合，得到更深一层的抽象特征，依次递归向上进行特征学习。不同的对象训练得到基本结构相同，而基本结构又可以按不同的组合规则构成不同的对象或部分特征。

2.2 卷积神经网络拓扑结构

深度学习在不同的应用场景下，涌现出许多独特的模型结构，并在各自领域中发挥着各自模型的优势与特点。常用的深度学习网络拓扑结构有自动编码器（Auto Encoder，AE）、稀疏编码（Sparse Coding，SC）、深度信念网络（Deep Belief Network，DBN）、卷积深度信念网络（Convolutional Deep Belief Neural Network，CDBN）、卷积神经网络（Convolutional Neural Network，CNN）和递归神经网络（Recurrent Neural Network，RNN）[51]。虽然这些网络模型在功能特点上不尽相同，但是深度学习算法均遗传于神经网络，其网络拓扑结构上都有一定的相似性。本文将以卷积神经网络为典型例子，图像处理为应用背景展开研究，进行详细说明。

从结构上可以发现，卷积神经网络是一种源自人工神经网络的扩展网络，

整个系统采用了前馈层次型结构，由输入层、隐藏层、输出层三个功能层组成。在结构上，卷积神经网络存在三个重要特性：局部连接、权值共享以及空间或时间上的子采样，使得卷积神经网络在二维模式识别中，具有一定程度的抗平移、旋转和比例缩放或其他形式畸变的能力[52]。图 2-6 所示为一个典型的卷积神经网络结构。

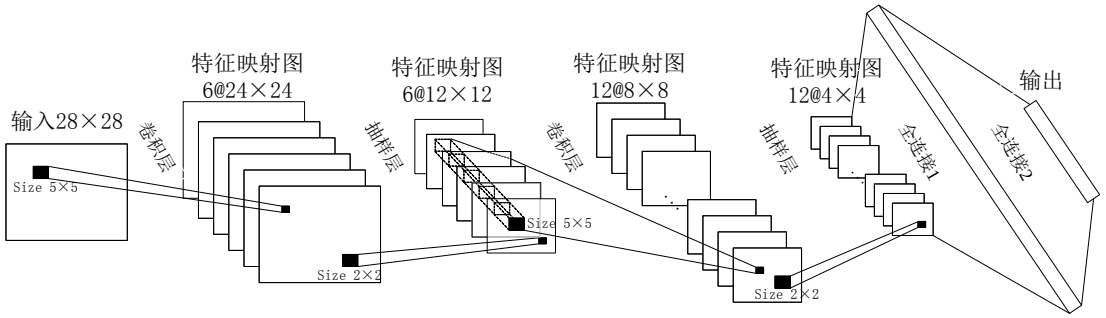


图 2-6 卷积神经网络结构

图例是一个含 6 个功能层的简单卷积神经网络结构，除了输入输出外，在中间的隐藏层包含了 2 个卷积层、2 个抽样层以及 2 个全连接层。从层间连接情况可以看出，卷积层和抽样层一般是相互交替连接，最后一个抽样层再通过全连接的方式连接一个或两个分类器，用于目标特征的分类。卷积神经网络的输入层负责加载原始输入数据，可以是多维数据，如二维或三维图像作为网络的输入，从而避免对复杂特征提取和数据重构的过程。卷积层的每个输出节点与前一层的局部节点连接，负责提取该区域特征。抽样层是为了增强网络的抗扭曲能力，进行滤波和二次特征提取。每一层卷积层和抽样层产生的目标特征是一个或多个二维数据矩阵子层，每个这样的子层一般称之为特征映射图。卷积神经网络每层是由多个二维子层组成，每个子层又是一个多神经元组成的二维平面，整个网络构成一个三维立体结构。本文以此网络拓扑结构，进行后续的设计与实现。

2.2.1 卷积层结构

卷积层是对前一层的特征映射图与学习得到的核进行卷积，得到的运算结果经过偏置参数的活性调节和激活函数的非线性变换，得到输出神经元，从而构成该层输出特征映射图。不同于传统人工神经网络采用全连接的互联计算方式，卷积神经网络在保证性能的前提下，采用非线性的卷积计算来代替全连接的线性变换，增强网络对复杂非线性问题的处理能力，且保证特征空间位置的不变性。卷积，在泛函分析中也叫摺积，从物理意义上可以看作是一个函数在另一个函数上加权叠加的效果。离散函数和连续函数的卷积数学公式分别如下：

$$y(n)=\sum_{i=-\infty}^{\infty}x(i)h(n-i)=x(n)*h(n) \tag{2-6}$$

$$y(t)=\int_{-\infty}^{\infty}x(i)h(n-i)di=x(n)*h(n) \tag{2-7}$$

式中，$x(n)$（$x(i)$）和$h(n)$（$h(i)$）两个序列变量（函数变量），i表示求和变量或积分变量，$*$表示卷积运算。

一幅图像的本质就是一组离散的二维数组。在图像识别过程中，一幅图像做卷积操作，其实际上就是一组二维的离散像素点与一组二维的离散变量进行加权求和，其计算方式与矩阵相乘类似。图 2-7 表示一个简单的卷积运算示例。

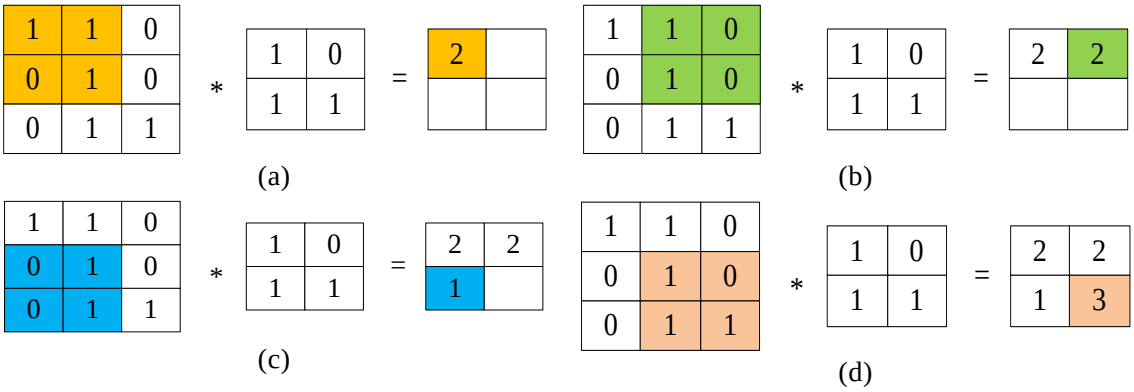


图 2-7 卷积运算示例

一个二维3×3输入矩阵与一个二维2×2权值矩阵卷积，得到一个2×2的输出矩阵数据，其卷积过程可以拆分为以上 4 个过程。3×3输入矩阵可以看成 4 组2×2矩阵的拼接组合，4 组2×2输入矩阵分别与2×2权值矩阵做矩阵卷积，得到 4 个输出结果组成的2×2输出矩阵。如图 a)的计算方式为 1×1+1×0+0×1+1×1=2，得到相应的输出结果。

传统人工神经网络层间节点采用全连接的方式，容易造成网络参数繁多、训练困难等问题。为了避免这些问题，卷积神经网络的卷积层采用局部连接、权值共享以及其他优化策略的计算方式，以大幅度减少连接路径和网络参数。局部连接：由于图像的空间关联是局部的，每个神经元只对局部范围进行感知。权值共享：在同一特征映射图的神经元之间，共享同一卷积核参数。假设第$l-1$层的特征映射图组数为n_{l-1}，每组特征映射图神经元总数为m_{l-1}，则第$l-1$层的特征映射图的总神经元总数为$n_{l-1}\times m_{l-1}$；第l层的特征映射图组数为n_l，每组特征映射图神经元总数为m_l，则第l层的特征映射图的总神经元总数为$n_l\times m_l$。第l层的每一组特征映射图都依赖于第$l-1$层的所有特征映射图，我们称之为全映射。在全映射的情况下，如果采用传统神经网络全连接的计算方式，则总共需

要$n_{l-1}\times m_{l-1}\times n_l\times m_l$个权值参数，庞大的参数总量容易对计算和训练过程造成困难；如采用局部连接的策略，选用$k\times k$大小的卷积窗口（即卷积核大小为$k\times k$，窗口面积k^2远小于m_{l-1}），则每次计算总共需要$n_{l-1}\times k\times k\times n_l\times m_l$个权值参数；如在局部连接的基础上，加入权值共享策略，则所需的权值参数只剩$n_{l-1}\times k\times k\times n_l$个，大幅度缩减了权值参数的数量，若考虑到$l$层每一组特征映射图都需要一个偏置参数，所需的参数也只有$n_{l-1}\times k\times k\times n_l+n_l$个而已。实际上，第$l$层的每一组特征映射图并非必须依赖于第$l-1$层的所有特征映射图。我们可以让第$l$层的每一组特征映射图只依赖于第$l-1$层的几组特征映射图，称之为非全映射。在非全映射的情况下，l层的特征映射图与$l-1$层的特征映射图存在依赖关系的总数设为t（$n_l<t<n_{l-1}\times n_l$），如采用局部连接和权值共享策略，则需要$t\times k\times k$个权值参数，网络参数进一步减少，若考虑到l层每一组特征映射图都需要一个偏置参数，总共需要$t\times k\times k+n_l$个网络参数。上述全映射和非全映射考虑的是l层的特征映射图与$l-1$层的特征映射图的依赖关系中卷积核权值参数不共享的情况，在目前实际图像或语音识别中，也存在不同依赖关系之间卷积核权值参数共享的情况，我们称之为共享映射。在共享映射的情况下，假设第l层的每一组特征映射图依赖的第$l-1$层的特征映射图之间，卷积核权值共享，如采用局部连接和共享参数策略，则总共需要$k\times k\times n_l$个权值参数，网络参数进一步减少，若考虑到l层每一组特征映射图都需要一个偏置参数，总共需要$k\times k\times n_l+n_l$个网络参数。

设定一个卷积层连接第$l-1$层与第l层的特征映射图。第$l-1$层的特征映射图个数为n_{l-1}，作为该卷积层的输入，每个输入特征映射图尺寸大小一致，由$X_{l-1}\times Y_{l-1}$个神经元点（像素点）组成；第l层的特征映射图个数为n_l，为该卷积层的输出，每个输出特征映射图尺寸大小一致，由$X_l\times Y_l$个神经元点（像素点）组成；卷积核尺寸大小为$k_x\times k_y$。通过卷积核提取第$l-1$层的特征映射图内涵特征，以特征组合的形式形成第l层的特征映射图。卷积核数量由相邻两层特征映射图的映射关系决定，若相邻两层特征映射图是全映射关系，则总共需要$n_{l-1}\times n_l$个卷积核。一般卷积层在计算时，不同的输入特征映射图的卷积窗口位置保持一致，窗口内的神经元与相应的卷积核进行卷积运算，将得到的卷积结果累加并经过神经元活性调整和非线性处理，得到输出特征映射图的一个神经元。卷积窗口沿水平和竖直方向，按一定的步进速率*step*平移，直至窗口扫描整个输入特征映射图，得到一幅完整的输出特征映射图。选择不同的输入特征映射图组合，总共n_l种组合，重复上述运算，得到第l层的所有特征映射图。输出输入

特征映射图的尺寸大小存在关系：$X_l = X_{l-1} - k_x + step$，$Y_l = Y_{l-1} - k_y + step$，一般卷积窗口步进速率 $step = 1$，则 $X_l = X_{l-1} - k_x + 1$，$Y_l = Y_{l-1} - k_y + 1$。特征映射图分辨率不断减小。整个过程可以用以下数学公式表示：

$$P_i^l = f(\sum_{j \in S_i^{l-1}} (K_{i,j}^l * P_j^{l-1}) + b_i^l) \tag{2-8}$$

式中，P_i^l 是 l 层的第 i 个特征映射图，P_j^{l-1} 是 $l-1$ 层的第 j 个特征映射图，$K_{i,j}^l$ 表示特征映射图 P_i^l 与对应的卷积核，b_i^l 是特征映射图 P_i^l 对应的偏置参数，S_i^{l-1} 表示特征映射图 P_i^l 与 $l-1$ 层存在依赖关系的特征映射图集合，f 则表示激活函数。

以图 2-6 为例，在第一个卷积层中，输入特征映射图是 1 个 28×28 大小的输入图像，输出为 6 个 24×24 大小的输出特征映射图，若采用全映射的连接关系，则需要 6 个 5×5 大小的卷积核。第二个卷积层中，输入特征映射图是 6 个 12×12 大小的输入图像，输出为 12 个 8×8 大小的输出特征映射图，若采用全映射的连接关系，则需要 72 个 5×5 大小的卷积核。

2.2.2 抽样层结构

一般情况，抽样层跟随在卷积层后面，对前一层特征映射图进行二度特征提取。卷积层得益于其结构特点，虽然可以显著减少神经元之间连接路径，减少网络参数，但是卷积层得到的每一个输出特征映射图的神经元总数并没有显著减少。如果卷积层后面直接加入一个全连接方式的分类器，则分类器的输入维度需要很高，容易造成过拟合的问题。为了解决这一问题，一般在卷积层后面添加一个抽样层，可以看作是一个模糊滤波器，进行数据的采样和特征的二度提取，同时也增强了网络抗扭曲能力。

由于图像具有区域空间特征相似性，即图像的连续变化特性使得相邻空间位置特征是相似的，因此抽样层一般采用下采样的计算策略，对信号进行离散化分割，常用方法有：

（1） 最大采样：选取区域空间内最大值代表该区域特征。

（2） 均值采样：选取区域空间内所有值的平均值代表该区域特征。

（3） 随机采样：选取区域空间内任意某一值代表该区域特征。

图 2-8 是抽样层三种下采样方式的运算示意图。从抽样层的作用可以发现，抽样层的抽样操作是针对特征映射图内部神经元，而不改变相邻特征映射图之间的映射关系，因此抽样操作之后，相邻特征映射图数量保持一致。抽样层在结构上依然延续了局部连接和权值共享的特点，另外为了降低特征维度，缩小特征图分辨率，增加了空间或时间上子采样的特性。

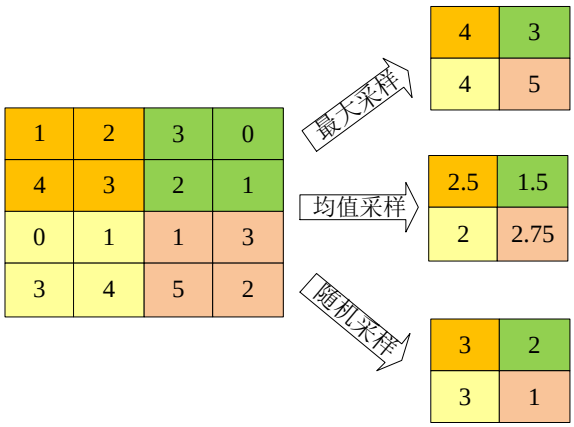


图 2-8 抽样层运算示意图

同样，我们设定抽样层连接第 $l-1$ 层与第 l 层的特征映射图。第 $l-1$ 层的特征映射图个数为 n_{l-1}，为该抽样层的输入，每个输入特征映射图尺寸大小一致，由 $X_{l-1}\times Y_{l-1}$ 个神经元点（像素点）组成；第 l 层的特征映射图个数为 n_l，为该抽样层的输出，每个输出特征映射图尺寸大小一致，由 $X_l\times Y_l$ 个神经元点（像素点）组成。抽样层的主要任务在于降低特征映射图的特征维度，因此相邻两层特征映射图数量保持一致，存在关系 $n_{l-1}=n_l$。在抽样层中，与卷积层类似，存在一个计算核，我们在这里称之为抽样核，抽样核尺寸大小为 $k_x\times k_y$。若采用最大采样的方法，则神经元最大值对应位置上的抽样核参数值为 1，其余均为 0；若采用均值采样的方法，则抽样核内每个参数值为 $1/\left(k_x\times k_y\right)$；若采用随机采样的方法，则抽样核随机选取某一位置上的参数值设为 1，其余均为 0。一般抽样层在计算时，输入特征映射图对输出特征映射图一一对应，若抽样核在抽样层内不共享，则抽样核数量与特征映射图数量保持一致，抽样窗口神经元与相应的抽样核进行抽样运算，得到的运算结果添加一个偏置参数 b 调整神经元活性，经过激活函数非线性处理得到输出特征映射图的一个神经元。抽样窗口沿水平和竖直方向，按一定的步进速率 $step$ 平移，直至窗口扫描整个输入特征映射图，得到一幅完整的输出特征映射图。一般抽样窗口尺寸大小远小于特征图尺寸大小，且 k_x 和 k_y 能被 X_{l-1} 和 Y_{l-1} 整除，输出特征映射图与输入特征映射图尺寸大小存在关系：$X_l=X_{l-1}/k_x$，$Y_l=Y_{l-1}/k_y$，特征映射图分辨率显著减小。整个过程可以用以下数学公式表示：

$$P_i^l=f(W_i^l\cdot pool(P_i^{l-1})+b_i^l) \tag{2-9}$$

式中，P_i^l 表示 l 层的第 i 个特征映射图，P_i^{l-1} 表示 $l-1$ 层的第 i 个特征映射图，W_i^l 表示特征映射图 P_i^l 与 P_i^{l-1} 特征映射图对应的抽样核，b_i^l 表示特征映射图 P_i^l 对应

的偏置参数，$pool$表示抽样函数，f表示激活函数。

抽样层的三种采用方式的数学公式可分别表示为：

$$pool_{max}(T) = \underset{a_i \in T}{Max}(a_i) \tag{2-10}$$

$$pool_{ave}(T) = \underset{a_i \in T}{Average}(a_i) \tag{2-11}$$

$$pool_{rand}(T) = \underset{a_i \in T}{Randbetween}(a_i) \tag{2-12}$$

式中，T表示特征映射图划分的采样区域，a_i表示采样区域内的神经元值。

以图 2-6 为例，第一个抽样层中，输入特征映射图是 6 个24×24大小的输入图像，输出为 6 个12×12大小的输出特征映射图，若抽样核在抽样层内不共享，则需要 6 个2×2大小的抽样核。第二个抽样层中，输入特征映射图是 12 个8×8大小的输入图像，输出为 12 个4×4大小的输出特征映射图，同样，若抽样核在抽样层内不共享，则需要 12 个2×2大小的抽样核。

2.2.3 全连接层结构

在卷积神经网络中，一般在最后一个抽样层后面添加一个或两个以全连接方式存在的全连接层结构，配合分类器将高维的特征数据归纳，整合成低维的特征数据，以便于特征分类与识别。全连接层的网络结构与传统的人工神经网络连接结构相同，每一个输入神经元与相应的权值系数相乘，得到的结果加上一个偏置系数调整神经元活性，再经过非线性激活函数变换得到输出神经元的值。每一个输出神经元连接上一层的所有输入神经元。

以图 2-6 为例，第一个全连接层中，输入是 12 个4×4大小的特征映射图，则相当于共有 192 个输入神经元，若假设第一个全连接层输出为 100 个神经元，第二个全连接层输出为 10 个神经元，则第一个全连接层至少需要192×100个连接路径；第二个全连接层以 100 个神经元作为输入，需要100×10个连接路径。

卷积神经网络与传统人工神经网络采用的激活函数是相同的，最常用的如 Sigmoid 型函数和 ReLU 型函数，在实际应用中都有非常好的表现。

2.3 系统总体方案设计

基于卷积神经网络硬件实现的目标，对整个网络进行总体方案设计。图 2-9 所示为卷积神经网络映射到硬件电路的工作流程框图。

从工作流程图可以看出，整个卷积神经网络网络主要分为时序逻辑控制电

路、索引地址更新配置电路、网络基本参数本地存储单元、运算系统和相应的缓存结构。

时序逻辑电路控制作为网络的核心控制器，相当于整个网络的大脑协处理器，以协调控制硬件电路各个模块的运作；索引地址更新配置电路包括输入索引地址更新配置电路和输出索引地址更新配置电路，主要用于网络基本参数和运算结构的存储位置管理；网络基本参数本地存储单元，则是为了运算当前运算网络结构索取存储在本地存储器内的相应参数，包括输入神经元、输入权值参数、输入偏置参数等；运算系统则是整个卷积神经网络运算的执行者，包括卷积层子系统、抽样层子系统、激活函数子系统和存储结构子系统，其中卷积、抽样、激活函数的子系统根据网络层次不同，又分为许多对应不同结构的运算层和预留可扩展的运算层，存储结构子系统则是配合运算其他各子系统进行的数据传输控制；缓存结构是为了暂存运算过程中的各运算系统计算最终结果和中间结构。

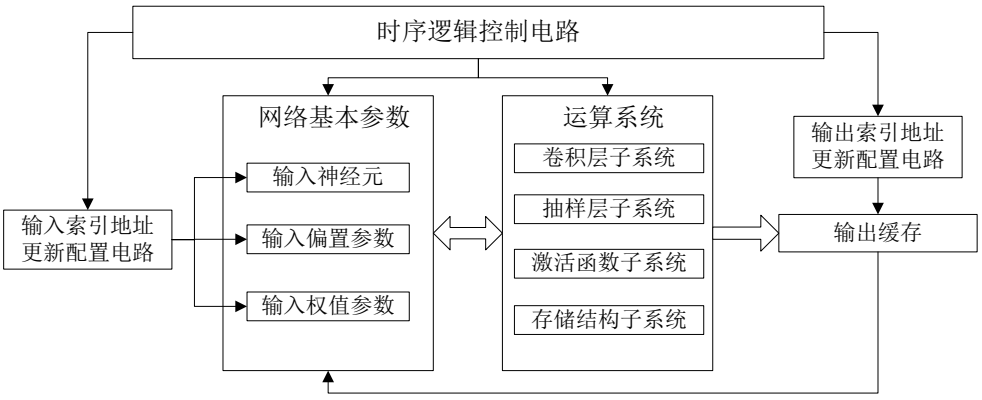


图 2-9 卷积神经网络工作流程框图

当网络运行时，首先，由时序逻辑电路判断当前运算状态，或是更新地址索引，或是读取相关网络参数，或是进入层内运算系统；然后，根据判断的当前状态，启动相关程序，调用相关硬件电路。若判断当前状态为更新地址索引，则继续判断是输入索引地址还是输出索引地址，然后更新相关的索引地址。若判断当前状态为读取网络基本参数，则根据更新的索引地址，访问相应的存储单元，读取各网络基本参数保存到缓存单元，包括当前层的输入神经元、输入偏置参数、输入权值参数等。若判断当前状态为进入运算系统，则继续判断是进行哪一层网络的何种运算，如进入第一层卷积层，或第二层抽样层，或第三层内的激活函数等，并且反馈访问的网络基本参数是否正确，判断基本参数正确则继续调用相关的运算子系统，，如卷积层子系统、抽样层子系统、激活函数子系统等，否则返回到读取网络基本参数；最后，配合输出索引地址更新模块

电路，将运算系统运算得到的结果保存到相应的存储单元中，并更新或扩展相关网络基本参数。将上述运行的信息反馈到时序逻辑控制电路中，继续重复上述操作，直到整个卷积神经网络内所有映射到硬件电路的结构传输完毕。

以模块化设计，整体协调控制的设计思想，将整个卷积神经网络拓扑结构按模块化映射到硬件电路中，图 2-10 所示为卷积神经网络硬件实现的系统总体方案。

整个硬件系统由多个模块电路组成，包括运算阵列模块、本地存储模块、存储管理模块、时钟管理、通信接口、主控制器、计算机和片外存储器等多个子部件。各模块电路通过相关的控制总线和数据总线相互衔接，信息沟通。

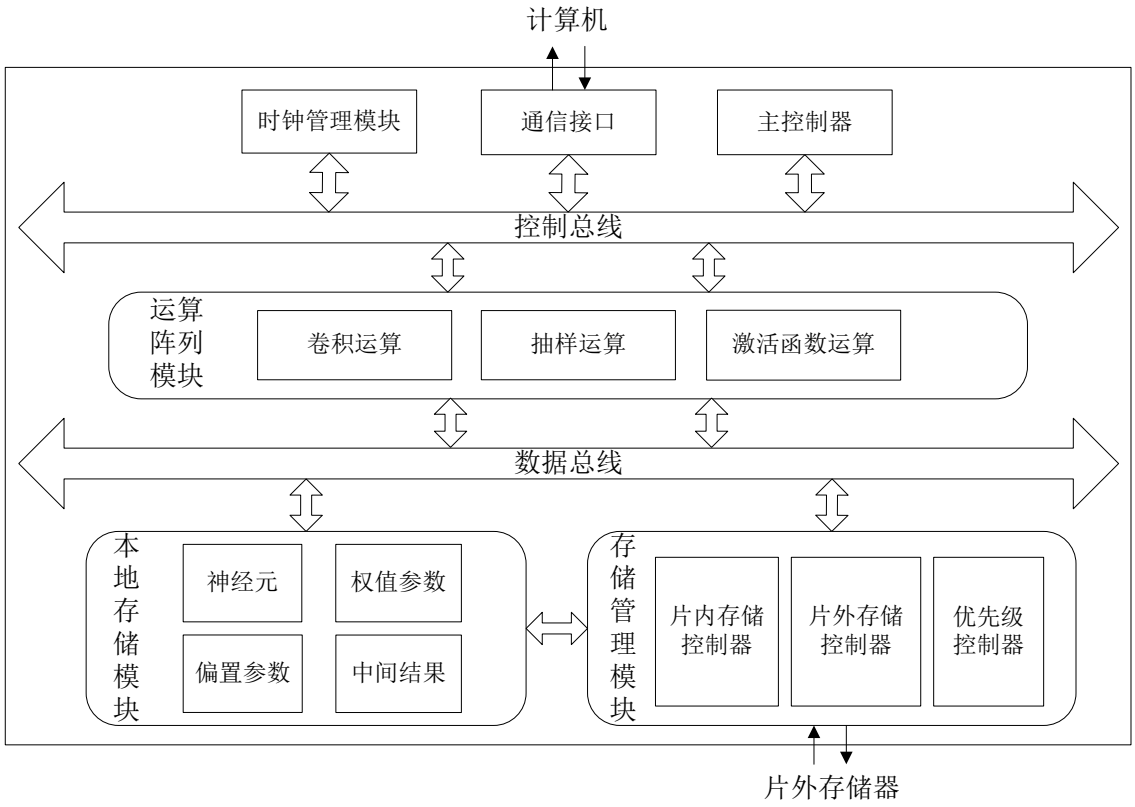


图 2-10 系统总体方案

由图可知，该方案的核心功能部件主要分三个模块，分别是运算阵列模块、本地存储模块和存储管理模块。运算阵列作为卷积神经网络的核心运算执行者，包括具体的卷积运算模块、抽样运算模块、激活函数模块，而每个模块又可以根据实际应用需求，进行并行拓展设计和开发。本地存储模块主要用于当前运算操作的主要缓存单元，包括输入神经元缓存块、输入权值缓存块、输入偏置参数缓存块和中间结果缓存块。存储管理模块是针对本地存储器和片外存储器的控制单元，包括片内控制器、片外控制器和优先级控制器，其中优先级控制

器是为了协调计算结果的存储位置，当存储目标是大规模的中间计算结果，则保存到片外存储器中，此时片外存储控制模块优先级高于片内存储控制模块；当存储目标为当前计算的中间结果或网络基本参数，则片内存储控制模块优先级高于片外存储控制模块。运算阵列模块、本地存储模块和存储管理模块通过数据总线进行相互数据传输和信号传递。

整个硬件电路还包括了时钟管理模块、通信接口模块和主控制器模块。这些模块通过控制总线与其他模块电路进行信息交互。时钟管理模块是根据不同模块电路的执行效率，配以不同频率的时钟信号，进行不同时钟域的工作。通信接口，是与计算机相连接的桥梁。硬件电路和计算机之间进行网络参数、原始数据、控制指令以及运算结果的有序传输。主控制器是整个硬件系统的核心控制单元，协调控制各个模块电路之间的工作，是整个网络的前向通路有序执行。

2.4 本章小结

本章首先在对深度学习理论基础深入研究的基础上，以卷积神经网络为深度学习典型代表，对其拓扑结构（卷积层、抽样层、全连接层、激活函数）和功能特点进行详细研究，并提出本文网络硬件化移植的具体网络拓扑结构，该拓扑结构除了输入输出以外，包含两个卷积层、两个抽样层、两个全连接层。最后对网络拓扑结构硬件移植，进行总体方案设计，将各网络拓扑功能映射到具体的硬件模块电路中。

第 3 章 算法硬件移植的优化技术与架构设计

在上一章对深度学习及拓扑结构研究的基础上，本章对软件算法硬件化移植过程中可能存在的优化技术进行详细研究，并对卷积神经网络做出并行架构设计。首先，比较目前常用的硬件平台，选择合适本文实现的硬件移植平台。然后，结合本文实现低功耗、高效率深度学习算法的目标，分别对速度、面积等方面进行相关技术研究。最后，以卷积神经网络为例，进行网络并行性架构设计。

3.1 硬件平台选择

许多智能算法在传统软件平台上的实现技术日渐成熟，但是随着工业化进程的不断加深，尤其是面临“工业 4.0”、大数据、云计算等新时期的挑战时，很多时候采用软件运行智能算法时，在开发成本、运营维护、时效性、低功耗等诸多方面显得力不从心，因此软件算法的硬件化实现成了许多智能算法实现的一种新途径。

软件算法硬件化移植，是指利用大规模硬件模块将软件实现的功能算法移植到硬件电路中去，以硬件电路固有的快速特性来提高算法的执行效率。目前常用的硬件移植平台有专用集成电路 ASIC、现场可编程门阵列 FPGA、图形处理器 GPU、数字信号处理器 DSP 等等[53]。随着 FPGA 性能的不断提升，其性价比优势不断得以体现。此外 FPGA 的高可靠性和低功耗特点，使其被广泛应用于航空航天、医疗器械、电子通信、视频图像等诸多领域。本文综合考虑了算法性能、设计成本、研发周期等多个因素，选择 FPGA 作为深度学习算法的硬件加速平台。

与 MCU、DSP、ASIC 这类固化的硬件电路相比，FPGA 作为一种半定制集成电路，采用了完全不同的电路结构形式，通过反复修改逻辑单元、功能模块、I/O 之间的连线方式，设计不同功能的电路结构[53]。FPGA 结构主要由一个用于存储编程数据的存储单元和 3 种可编程单元组成，分别是：输入/输出模块（I/O Block，IOB）、逻辑单元（Logic Element，LE）和互连资源（Interconnect Resource，IR）。一些高端的 FPGA 芯片还集成了复杂的功能模块，如数字信号处理器、存储控制器、高速串行/解串收发器、PCI-Express 接口。图 3-1 是 FPGA 的基本结构形式示意图。

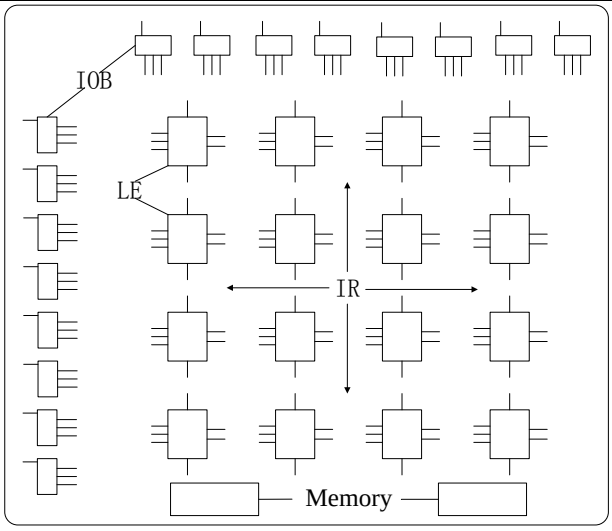


图 3-1 FPGA 基本结构示意图

FPGA 通过大规模独立的逻辑单元阵列和一系列互连资源连接构成复杂的功能电路，并配合存储器和输入输出模块配置电路参数，以实现复杂的逻辑功能。以卷积神经网络为代表的深度学习模型，是由大规模神经元节点和神经元之间的连接线，共同构成复杂的网络拓扑模型，并通过调整网络结构参数，以实现特定的算法功能，如目标识别、分类等等。两者在结构特点、功能实现上有着惊人的相似之处，因此 FPGA 非常适合作为深度学习的硬件实现平台，以充分挖掘网络固有的结构特性。

以 FPGA 为硬件平台设计大规模以及超大规模集成电路时，需要遵循一个重要的原则，就是如何把握资源面积与计算速度的互换原则[54]。在设计过程中，为了系统性能优化，在提高运行速度、节约资源、降低功耗方面，有许多值得注意的优化设计技术。

3.2 优化技术研究

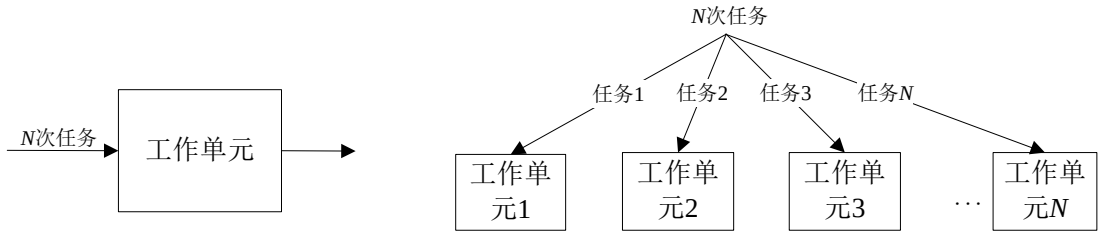
在 FPGA 设计与优化过程中，资源面积和计算速度作为矛盾双方，两者的抉择地位是不相等的，当两者发生冲突时，应当优先保证计算速度，满足工程在较高的工作频率。考虑到卷积神经网络高密度的计算特性，需要进行相关的速度优化技术研究。

3.2.1 并行技术

并行技术是相对于串行技术来说的，是指在一个时钟周期内可以同时执行多个指令操作，以提高系统整体的运算操作速度。并行技术从实现方式来划分，可以分为空间并行和时间并行。

空间并行是指将计算任务分配到多个独立的计算单元进行并行化处理；时间上的并行则主要是指通过流水线技术，将计算任务分配到数据流驱动过程中的每一个细小环节中并行处理。从上一章对卷积神经网络结构的分析中可以发现，多层次的网络本身必然存在多种形式的并行可开发性，这些可能的并行性可以根据各自结构特性，按空间或时间并行技术进行并行实现，而并行技术实际上也是资源面积换取计算速度的思想的一种具体体现。

空间并行非常容易理解，就相当于一个重复性为 N 的任务交给 N 个不同的独立工作单元同时处理，会比交给 1 个工作单元重复工作 N 次效率要高出 N 倍，空间并行示意图如图 3-2 所示。



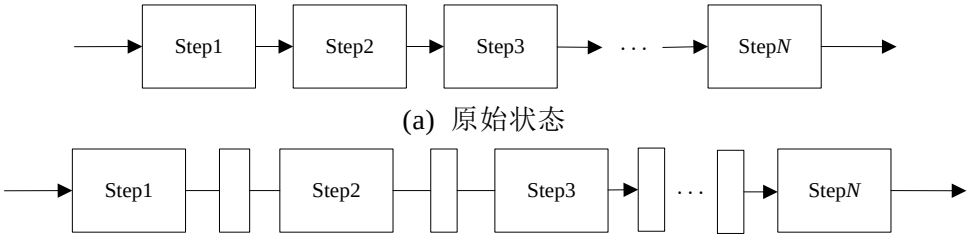
(a) 原始状态　　(b) 并行优化

图 3-2 空间并行示意图

示意图中，图 3-2(a)所示的原始状态是将任务切分成 N 个周期，交给同一个工作单元进行单核任务处理，工作重复性高，处理速度慢；图 3-2(b)是将任务平均分解成 N 个等效子任务，分别分配给不同的 N 个工作单元进行多核并行任务处理，只需一个时钟周期就能完成，处理速度快。

在卷积神经网络中，存在大量的空间并行的可能性。例如，卷积窗口内的卷积操作是一种重复性很高的乘累加操作，可以将所有计算任务分配多个独立的乘累加器中并行计算，以提高系统的计算速度。

时间并行可以认为是一种流水线式并行，相当于将一个单线程的完整大操作流程细分为 N 个子操作，实现逻辑分级分层和多线程操作。时间并行示意图如图 3-3 所示。



(a) 原始状态

(b) 并行优化

图 3-3 时间并行示意图

示意图中，完整操作步骤是连续不间断的，但是在原始状态下，后一步骤

的操作需要在前一步骤完成后启动，相邻步骤之间的逻辑延时较长。若在相邻步骤之间添加逻辑寄存器，锁存每一级步骤的操作结果，使操作分级处理，减少每一步骤之间的逻辑延时，从而减少整个操作的总延时时间。

时间并行是在时间上复制了多个处理模块，提高系统的工作频率。例如，在卷积神经网络计算时，存在大量的乘累加操作，可以在多个累加器之间添加寄存器构成流水线式结构，实现从时间上的单线程到多线程的开发，提高系统的工作频率。

空间并行和时间并行可以组合使用，这种“空间+时间”组合式的并行结构也是收益最高的并行加速实现方式。

3.2.2 乒乓技术

乒乓技术是一个常被用于控制与缓冲数据流。将单通道的数据流分节拍进行双通道切换传输，使得数据得到无缝衔接的缓冲和处理。乒乓操作的核心思想就是以双缓冲器的交互运行方式将数据传输与计算时间进行重叠抵消。乒乓操作示意图如图 3-4 所示。

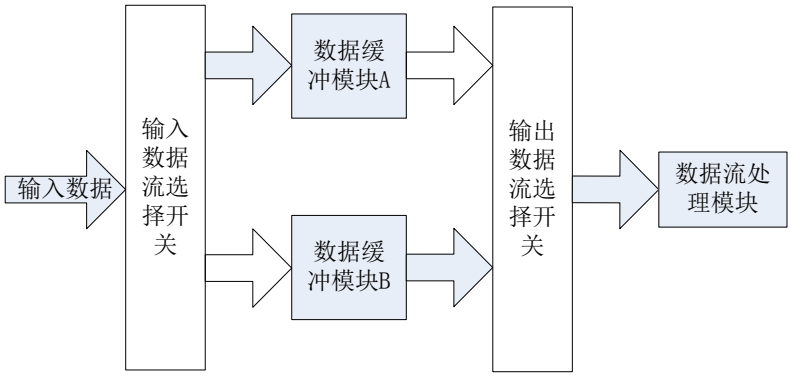


图 3-4 乒乓操作示意图

在乒乓结构中，数据缓冲模块可以是任何的存储单元，如常见的 RAM 和 FIFO。输入数据流选择开关和输出数据流选择开关为一个二选一数据选择器，控制数据流的流向。在第一个缓冲阶段，第一组输入数据通过输入数据流选择开关传递到数据缓冲模块 A 缓存。在第二个缓冲阶段，输入开关切换位置，第二组输入数据通过输入选择开关传递到数据缓冲模块 B 缓存，如此同时，上一阶段缓存在数据缓冲模块 A 中的输入数据，通过输出数据流选择开关传输到数据流处理模块。第三阶段，输入输出开关均切换位置，将下一组输入数据传递到数据缓冲模块 A，同时数据缓冲模块 B 的缓存数据传输到处理模块，进行数据流处理。如此循环，周而复始。

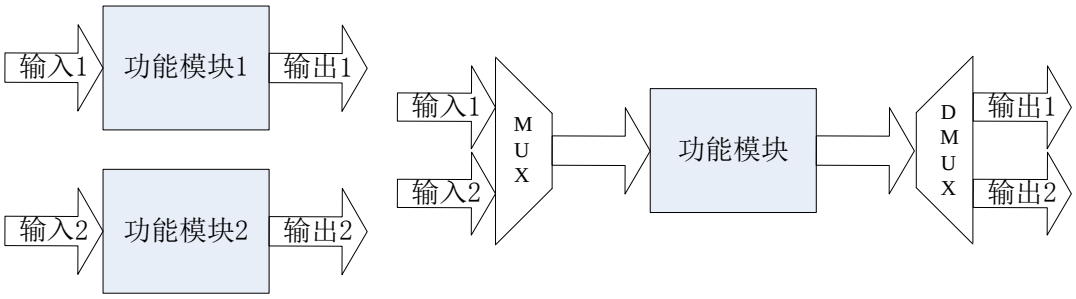
如果把乒乓结构看作是一个整体结构，那么数据流从输入端到输出端是连

续不断的传输过程，类似于一个隐形的流水线式结构，中间缓存部分是不间断运行的。另外，乒乓结构还可以缓解处理模块的运算压力。例如，输入数据的传输速率为每秒 2 组输入数据，数据处理模块的处理速率为每秒 1 组输入数据。如果只用一个数据缓冲模块，或者输入数据直接传递到处理模块，那么容易造成数据堵塞、数据混乱的问题；而采用含有双缓冲单位的乒乓结构，可以按节拍切换使用缓冲单元，避免高速的数据流对处理模块的数据冲击。从另一个角度来讲，对于高速的输入数据流，系统要求的数据处理速率仅为数据流传输速率的一半即可，从而大大的降低了数据处理模块的运算负荷。

卷积神经网络运行时，相邻两层之间数据具有依赖性，上一层输出的特征映射图神经元需要保存到存储器，而下一层结构又需要将上一层的输出特征映射图作为本层结构的输入，分配到计算单元进行处理。如果两层之间不添加乒乓技术，那么执行下一层计算之前，需要等待上一层计算完成，容易造成时间浪费。因此，在相邻两层之间，可以添加乒乓技术，使相邻两层之间的数据流进行不间断传输，提高系统计算速度。

3.2.3 复用技术

复用技术是指针对同一块硬件资源需要被多个电路模块使用时，为了避免资源浪费，使重用的硬件资源共享于多个电路模块，做到资源共享，面积最优。硬件实现过程中，模块复用的实现方式有很多种，如时分复用、设置状态机、调整实现方式等等。图 3-5 是采用时分复用的方式实现资源共享的示意图。



(a) 原始状态　　(b) 资源共享

图 3-5 时分复用实现资源共享

功能模块可以是系统中任意功能的电路资源模块。例如，卷积神经网络的卷积运算模块，需要被多个不同结构层的电路模块复用，进行层间串行模式的并行开发。图 3-5(a)原始状态中，输入 1 和输入 2 分别调用功能模块 1 和功能模块 2，输出对应的两个输出。不同层间结构需要两份的功能相同的硬件资源，分别实现各自功能，造成资源浪费。图 3-5(b)通过信号合成器 MUX 将输入数据

时分复用，调用同一份功能模块的硬件资源，得到的输出结果通过信号分解器 DMUX 分时段应用于不同电路模块中。

另外，还可以通过状态机的方式调整功能模块的复用情况。当状态机的状态数较多时，可以改变有效编码方式，如改成格雷码、二进制码等，来减少资源占用面积。

3.3.4 分块技术

分块技术是结构化设计中的一个重要方法，也是节约硬件资源的一种重要设计策略。卷积神经网络的多层化结构，使其必然需要进行模块划分。

将具有相同或相似的逻辑功能划分成一个模块。一方面，由于数据的局部特性，可以最大程度上提高数据复用和模块复用程度，减少设计所消耗的资源面积；另一方面，相同或相似的逻辑功能也更利于得到更好的综合优化效果，一定程度上能提高计算速度，节约资源占用面积。例如，在卷积神经网络中，卷积操作具有高密度、高频率的重复性，因此，可以将卷积运算划分成一个单独的模块，通过复用卷积运算模块，减少系统对乘法器、加法器和存储单元等硬件资源的占用率。

从理论上来说，模块电路规模越大，越有利于综合优化，节约资源，提高运算速度，但是过大的电路模块对综合器和电脑配置的要求更高，需要同时处理大量的逻辑单元和存储单元，不利于发挥目前增量综合与实现技术的优势。因此需要选择一个合理规模的模块电路进行分块设计。

存储器作为系统的记忆设备，是硬件设计中不可缺少的一部分。借鉴逻辑分块的思想，采用存储单元分片设计。结合卷积神经网络权值共享的特性，可以将共享数据，如权值、同一个特征图的神经元，分片存储在同一个存储单元内，提高数据复用率，也减少数据存取时对资源的占用面积。

3.3 卷积神经网络并行架构设计

深度学习算法的硬件加速效果，很大程度取决于算法并行度的开发程度。目前，卷积神经网络大多采用软件实现，以串行指令的方式将整个网络串联起来。然而，以卷积神经网络为代表的深度学习，在结构和认知上存在一个重要的特点，就是将数据信息以分布式的方式存储在各个分散的神经元上，并行处理各单元信息。这一特点使得软件实现方式与深度学习网络特点大相径庭，无法充分发挥网络结构固有的并行特性。

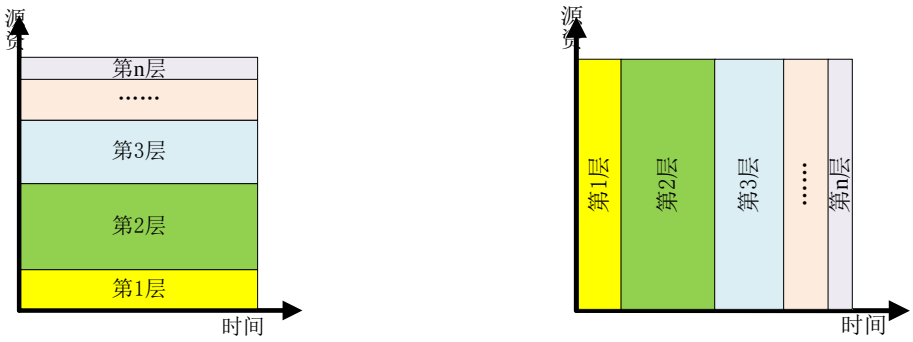
卷积神经网络属于一种前馈型多层网络，其层间结构、层内运算、数据流驱动都具有一定的相似性，因此，卷积神经网络拓扑结构本身存在着多种形式

的并行可开发性，本小节结合硬件移植过程中的优化技术，对卷积神经网络进行并行架构设计，主要包括不同层之间的并行架构、特征映射图之间的并行架构、特征映射图内部的并行架构、卷积运算的并行架构以及其他形式的并行架构。

3.3.1 不同层之间的并行架构

卷积神经网络的层次化结构，使其必然存在不同层之间的并行性。但是从理论上分析，要实现完整的卷积神经网络前向通路硬件加速，需要耗费巨大的硬件资源，实现起来具有一定的困难。一方面，由第二章分析可以看出，全连接层的参数非常多，计算重复性和密集程度不够，若要发挥硬件计算单元最大的计算性，则需要耗费很大的存储带宽。另一方面，分类器的分类数目是不确定的，需要根据实际应用不定时地修改全连接层和分类器的参数。因此，本文将全连接层和分类输出层部分交给软件处理，层间的并行特性只考虑中间隐藏层中的卷积层和抽样层。

在硬件上实现并行特性开发，其实际上就是硬件资源与计算时间之间的互换过程。不同层之间的并行开发，存在两种不同的实现模式，一种是并行模式，另一种是串行模式，示意图如图 3-6 所示。



(a) 并行模式　　(b) 串行模式

图 3-6 层间并行特性两种实现方式示意图

并行模式，是指不同结构层按照每一层的计算任务分配相应的资源，在硬件平台上同时完成所有结构层的计算任务，计算完成后将结果返回给主控制器，进行综合处理，如图 3-6(a)所示。但是，卷积神经网络作为一种前馈型结构，数据流在各层结构之间是逐层向前传递的，层间数据存在数据相关性，而且层间并行任务量十分有限，这也对层间并行架构开发造成巨大挑战。这种并行模式下的并行架构相对较为死板，当卷积神经网络模型发生变动时（如层数增加、卷积核尺寸改变），则资源就需要重新分配，设计灵活性差，增加设计周期。因此，本文不进行并行模式下的层间并行架构开发。

串行模式。另外，我们考虑到卷积层和抽样层在结构和运算上都具有很大的相似性，因此，我们可以充分开发单层网络结构的并行架构，而多层网络结构可以通过时间上的复用，反复调用单层结构，以串行模式来实现多层网络的层间并行架构开发，如图 3-6(b)所示。本文也主要采用串行模式下的层间并行架构开发。

3.3.2 特征映射图之间的并行架构

卷积层和抽样层在结构和运算上都具有很大的相似性，本文将以卷积层为例，分析以下各种类型的并行架构。卷积层是输入特征映射图和输出特征映射图的连接桥梁，也就是说存在一种多对多（n_{l-1} to n_l）的映射关系，因此在层内存在特征映射图之间的并行架构。这种多对多的映射关系是以特征图为最小单位来映射的，因此输入特征映射图数量一般大于 1，否则无法形成完整的一个输出特征映射图。非共享映射的情况下，卷积神经网络单个卷积层的结构示意图如图 3-7 所示。

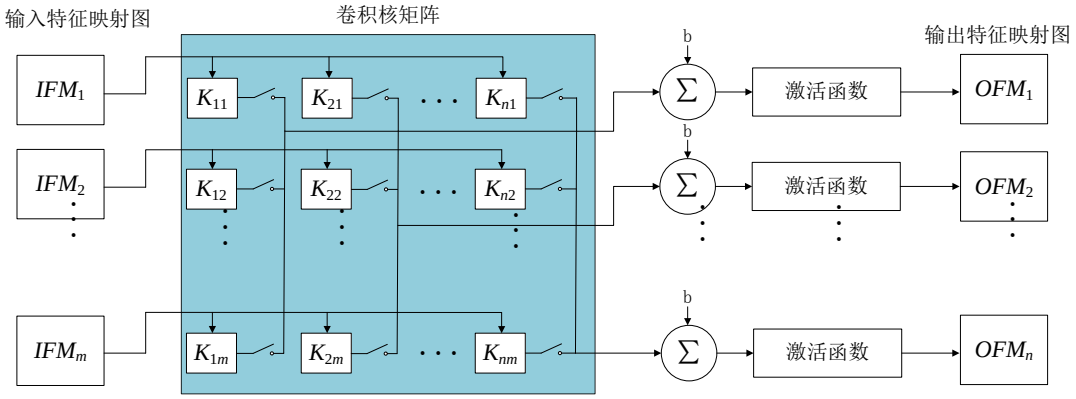


图 3-7 单个卷积层的结构示意图

卷积层连接前后两层的特征映射图。在图中，$IFM_1, IFM_2, ..., IFM_m$ 表示第 $l-1$ 层的 m 个输入特征映射图，$OFM_1, OFM_2, ..., OFM_n$ 表示第 l 层的 n 个输出特征映射图，$K_{11}, K_{12}, ..., K_{1m}, K_{21}, ..., K_{nm}$ 表示输入输出特征映射图之间相对应的卷积核，最多总共有 $m \times n$ 个，形成卷积核矩阵。若卷积层采用全映射的关系，则卷积结果输出通道开关全部闭合，否则通道开关选择性闭合。

在卷积神经网络中，不同卷积层对应 n 和 m 的数量是不确定的，其对应的潜在的并行架构也有多种组合形式，主要包括多输入单输出的并行架构和多输入多输出的并行架构。

（1）多输入单输出的并行架构

多输入单输出的并行架构，是指在一个卷积计算层内，由一个输出特征映

射图对应的多个输入特征映射图，同时与对相应的卷积核做卷积运算，每次运算只输出一个输出特征映射图，结构示意图如图 3-8 所示。m 个输入特征映射图 $IFM_1, IFM_2, ..., IFM_m$ 与 m 个卷积核 $K_{21}, K_{22}..., K_{2m}$ 同时进行卷积运算，得到的卷积结果通过通道开关选择，与偏置系数进行累加，得到的结果进过激活函数非线性变换得到输出特征映射图 OFM_2。

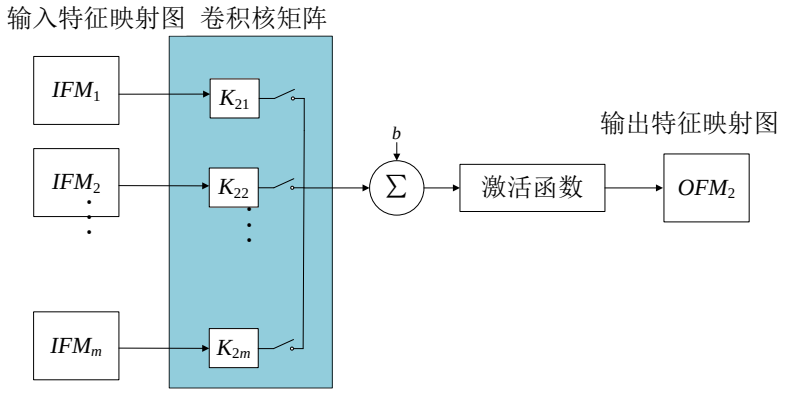


图 3-8 多输入单输出并行结构示意图

假设一个输入特征映射图总共包含 R 个神经元，特征图是按神经元逐个扫描读取，则完成一个输入特征映射图的读取至少需要 R 个单位时钟周期。假设输出特征映射图 OFM_2 需要 m 个输入特征映射图的映射，为了保证多输入单输出的并行结构运算正确性，必须要保证 m 个输入特性映射图并行读取，并与相应的卷积核同时运算。因此，完成一个输出特征映射图，系统至少需要 R 个单位时钟周期，同时要求系统能满足单位时钟周期能加载 m 个神经元位宽的存储带宽要求。若每个单位时钟周期读取 r 个神经元，则系统运算时长将成比例降低，需要至少 R/r 个单位时钟周期，但是系统运算所需的存储带宽和计算资源会成比例增加。

（2）多输入多输出的并行架构

多输入多输出的并行架构，是指在一个卷积计算层内，由多个输出特征映射图对应的多个输入特征映射图，同时与对相应的卷积核做卷积运算，每次运算只输出多个输出特征映射图。多输入多输出结构可以看成是多个多输入单输出并行结构的并行叠加效果，其结构如图 3-9 所示。m 个输入特征映射图 $IFM_1, IFM_2, ..., IFM_m$ 与 $j \times m$ 个卷积核，同时进行卷积运算，得到的卷积结果通过通道开关选择，与偏置系数进行累加，得到的结果进过激活函数非线性变换得到输出特征映射图 $OFM_1, OFM_2, ..., OFM_j$，是由 j 个多输入单输出并行结构叠加组成。当 $j = n$ 时，该多输入多输出并行结构就是单个卷积层结构。

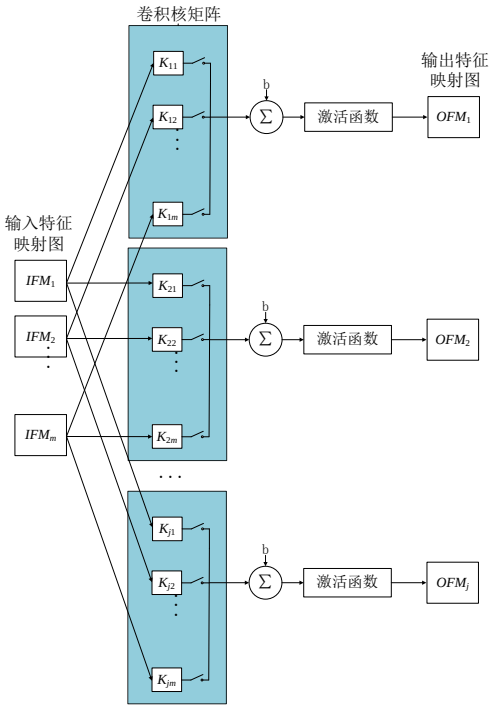


图 3-9 多输入多输出并行结构示意图

多输入多输出并行结构在结构上是多输入单输出的并行叠加，因此系统所需要消耗的资源是随着输出并行度的提升而增加的。当输出 j 个输出特征映射图，此时输出并行度也为 j，若要保证系统在 R 个单位时钟周期内输出 j 个输出特征映射图，则消耗的计算资源和存储资源至少增加 j 倍。

3.3.3 特征映射图内部的并行架构

特征映射图内部的并行架构是一种细粒度的并行开发，若能充分开发这一层面的并行架构，那么卷积神经网络的执行效率将会大大提高。在卷积层中，每一个输入特征映射图都会与卷积核做卷积运算，得到相应的输出特征映射图神经元或中间结果。根据一个特征映射图做卷积运算时，卷积窗口的位置关系和卷积核数量，又可以将特征映射图内部的并行架构分成两种不同的并行架构，分别为多卷积窗口的并行架构和多卷积核的并行架构。

（1）多卷积窗口的并行架构

多卷积窗口的并行架构，是指在一个输入特征映射图内，多个不同位置卷积窗口对应的神经元与同一个卷积核相连，并同时进行卷积运算，与其他输入特征映射图的计算结果累加，经过偏置参数和激活函数调整处理后，得到多个不同输出神经元或者中间结果。

卷积层中，同一个特征映射图内的所有神经元都共享一个卷积核。每次计算时，卷积核的参数只与特征映射图内的局部神经元相连做卷积运算。多卷积窗口的并行架构，其实质就是通过增加卷积窗口，将局部连接的结构特性并行化开发，其并行结构如图 3-10 所示。

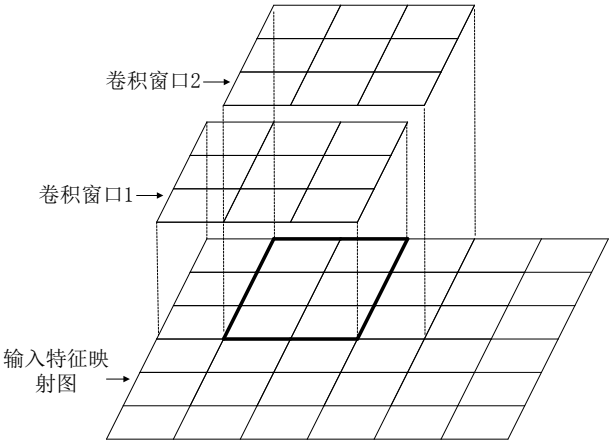


图 3-10 多卷积窗口并行结构示意图

如示意图所示，一个6×6输入特征映射图中，两个3×3卷积窗口的神经元与卷积核相连，同时做卷积运算，卷积窗口的并行度为 2。若两个卷积窗口不存在重叠，那么系统运算时需要同时加载 18 个神经元；若两个卷积窗口存在重叠，那么重叠部分的神经元可以同时被两个卷积窗口使用，减少了读取神经元的数量。我们假定重叠神经元共享的卷积窗口数量为重叠神经元的重叠度。如图中，两个卷积窗口之间存在 6 个重叠的神经元，重叠度为 2，那么系统运算时需要同时加载 12 个神经元。假设实现n个卷积窗口的并行，即卷积窗口并行度为n，卷积窗口尺寸大小为$k\times k$，重叠度为i的重叠神经元数量为T_i，那么系统运算时需要同时加载$k^2\times n-\sum_{i=0}^{n-1}(i-1)\times T_i$个神经元。卷积窗口之间重叠神经元数量越多，重叠度越高，每次计算时神经元的重复利用率就越大，降低对存储器的交互频率和存储带宽要求，也减少了系统的运算功耗和时间。当第一组并行的多个卷积窗口计算完成后，卷积窗口向右或向下按一定步进速率平移，进行下一组的卷积运算。若相邻两组的卷积窗口仍存在神经元重叠，那么重叠的神经元可以在短时间内再次被重复利用，系统只需加载还未加载的神经元即可，对存储带宽的负荷能进一步减小。

若多卷积窗口并行架构被开发，随着卷积窗口并行度的提升，系统运算时长将成比例降低。当卷积窗口并行度能覆盖整个输入特征映射图，那么在一个卷积窗口计算结束时，将得到所有的输出神经元，运算速度大幅度提升。但是

开发这种类型的并行架构，需要以消耗硬件资源为代价。也就是说，随着卷积窗口并行度的提升，系统需要消耗的计算资源和存储资源也将成比例增加。

（2）多卷积核的并行架构

多卷积核的并行架构，是指在一个输入特征映射图内，在同一位置卷积窗口对应的神经元与多个不同的卷积核相连，并同时进行卷积运算，与其他输入特征映射图的计算结果累加，经过偏置参数和激活函数调整处理后，得到多个不同输出神经元或中间结果。

多卷积核的并行架构，其实质上就是通过增加卷积核数量，是将权值共享的特性并行化，其并行结构如图 3-11 所示。

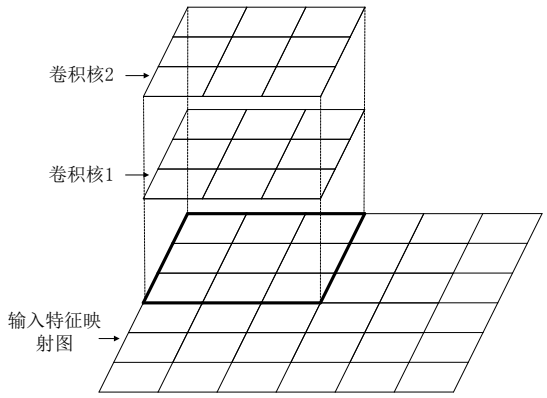


图 3-11 多卷积核并行结构示意图

如示意图所示，一个输入特征映射图中，同一个卷积窗口的神经元以广播式的方式传递到两个不同的卷积核，同时做卷积运算，此时卷积核的并行度为 2。对于同一输入特征映射图不同输出特征映射图的映射关系，重叠神经元通过数据重用策略，只需读取一次输入特征映射图的神经元，大幅度减少了对存储资源的重复访问次数，但是需要额外增加多个卷积核运算所需的计算资源和存储中间结果的存储资源。可根据硬件平台的实际硬件资源，调整卷积核的并行度来实现多卷积核的并行加速效果。

硬件能同时运行卷积窗口和卷积核的数量，对卷积神经网络的加速性能有较大的影响。在资源允许的条件下，同一结构中可以同时现实多卷积窗口的并行架构和多卷积核的并行架构，以实现多种并行特性的组合形式。

3.3.4 卷积运算的并行架构

卷积运算是卷积神经网络运行过程中最核心、最密集的运算操作，也是并行特性的重点优化对象。一个完整的卷积神经网络中包含了成千上万的神经元节点，而每一个神经元节点都相当于一个独立的计算单元，因此卷积神经网络

中存在大量卷积运算的并行，这是一种更为细化的并行架构设计。

卷积运算的并行架构，是指在一个卷积窗口内，所有神经元节点与卷积核对应的参数同时进行乘法运算，经过偏置参数和激活函数调整处理后，得到一个输出神经元节点或中间结果。卷积运算并行结构如图 3-12 所示。

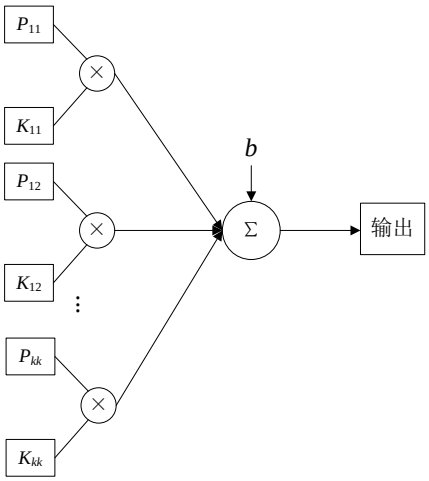


图 3-12 卷积运算并行结构示意图

假设卷积窗口大小为$k\times k$，图中P为卷积窗口中的神经元，K为卷积核的参数，卷积窗口中的神经元与卷积参数相乘，得到的结果与偏置参数累加，得到窗口卷积运算的中间输出结果，该输出结果将与其他输入特征映射图的输出结果进行叠加，最终得出输出特征映射图的一个神经元。若$k\times k$大小卷积窗口内的乘法操作都同时进行，那么此时卷积运算的并行度为k^2。加法器之间添加流水线式寄存器，那么在一个周期内就可以完成k^2个乘累加操作，得到一个输出结果。要实现这种形式的并行架构，系统需要能同时加载k^2个神经元和k^2个卷积核参数，以及提供k^2个乘累加操作的计算资源，对系统资源分配提出了更高的要求。

3.3.5 其他形式的并行架构

特征映射图是由大量的神经元所组成，每个神经元由一定位宽的定点数或浮点数表示。卷积神经网络在硬件实现过程中，针对操作数层面上，存在数据位的并行架构和字节比特的并行架构。而这种底层的并行架构主要由具体实现的计算功能部件所决定。

3.4 本章小结

首先，本章通过比较目前常用的硬件移植平台，选择适合本文设计的 FPGA

作为硬件实现框架。然后，结合本文实现低功耗、高效率深度学习算法的设计目标，分别对并行技术、乒乓技术、复用技术、分块技术等进行相关速度和面积方面的优化技术研究。最后，结合硬件移植的优化技术，对卷积神经网络进行从粗粒度到细粒度的并行架构设计，为下一章的具体硬件实现奠定构架设计方案的基础。

第 4 章 卷积神经网络的硬件设计与实现

本章以卷积神经网络为研究对象，对结构进行层次化划分和模块化硬件电路设计。首先，基于 FPGA 的卷积神经网络整体架构设计，对模型进行更结构化描述，框定网络整体的设计方案。然后，将卷积神经网络划分成卷积运算模块、抽样运算模块、激活函数三个关键模块，结合网络固有的并行特性分别进行详细设计和说明，并以仿真的形式验证各模块在功能正确性和有效性。最后，对整个卷积神经网络实现过程中涉及到的缓存结构进行乒乓优化，完成卷积神经网络前向通路的关键模块电路设计与优化。

4.1 网络整体架构设计

本文以 FPGA 为硬件平台，进行卷积神经网络软件算法的硬件化映射，实现对算法的细致化、结构化的描述，网络的整体架构布局如图 4-1 所示。

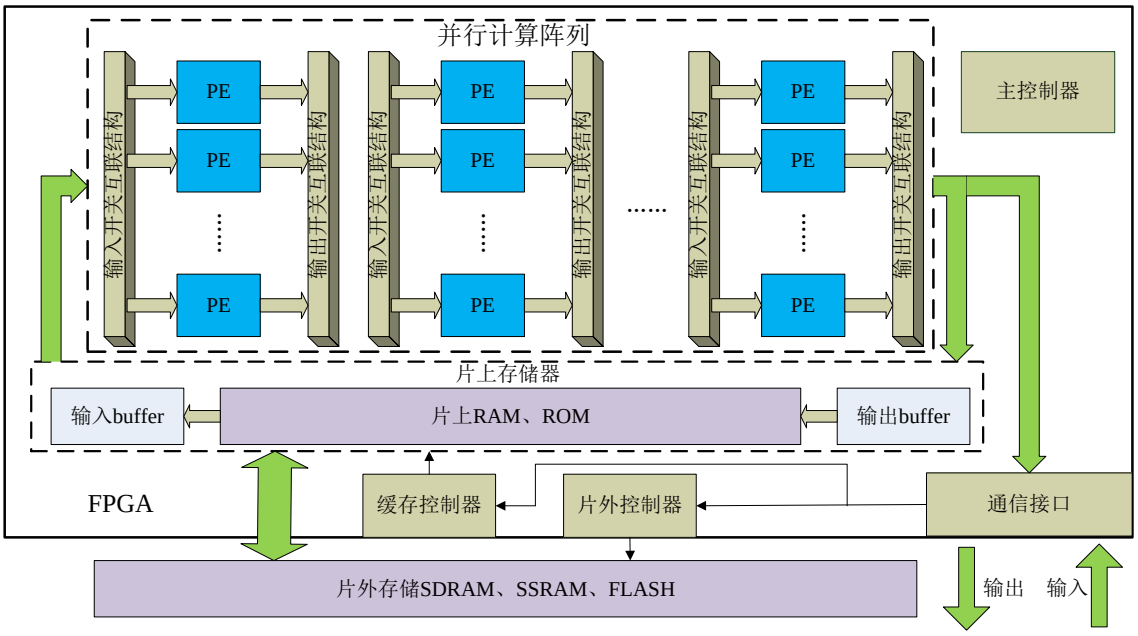


图 4-1 网络的整体架构框图

网络的整体架构主要分两部分资源：一是 FPGA 片上的内部资源；二是 FPGA 片外的外接资源。内部资源包括片上存储器、并行计算资源、控制器和外围标准通信接口等；外接资源主要包括片外存储器和主机，分别用于存储计算过程中产生的大量中间结果和用户的人机界面操作。

卷积神经网络运算所需的网络参数，如卷积核参数、抽样系数，均由主机

的 CPU 软件来训练完成。首先，系统将软件训练得到的网络参数、需要识别分类的输入图像数据和一系列的控制指令通过通信接口传输到 FPGA 内部资源进行保存，通信接口可以是 PCI 总线、PCI-Express 总线、USB、串行接口等。当 FPGA 接收到卷积神经网络开始运行指令后，启动系统主程序，将存储在片上存储器内的输入图像通过输入缓冲 buffer 进入并行计算阵列，并行计算阵列根据不同的计算任务，将输入数据分配到不同的硬件计算单元 PE 中进行数据处理。若并行计算阵列计算得到的是最终输出结果，那么计算结果通过通信接口反馈回主机 CPU;若并行计算阵列计算得到的是一个结构层的输出特征映射图，那么计算结果传输到片外存储器中保存，等待下一结构层计算开始，或者通过输出缓存 buffer 传输到片上存储器中，继续返回到并行计算阵列进行下一结构层的计算；若并行计算阵列计算得到的是输出特征映射图的中间结果，那么直接通过缓存 buffer 返回到并行计算阵列，等待进行下一步的计算处理。片上缓存控制器和片外控制器分别控制片上存储器和片外存储器的工作。主控制器宏观调控 FPGA 内部所有资源的协调工作，例如 ARM、MCU，甚至是自己设计的 IP 核。在并行计算阵列中，计算单元 PE 的输入端和输出端分别添加了输入开关互联结构和输出开关互联结构，以方便输入数据和输出数据的驱动与分配，保证数据计算的流畅性和高效性。

4.2 卷积运算模块

随着数字图像在工业生产和实际生活中得到广泛应用，以数字图像的二维卷积运算技术为代表的数字图像处理技术得到了快速的发展和长足的进步，在图像空间滤波、降噪除噪、边缘检测、特征提取等多个领域都有成熟的应用。

卷积运算模块是整个卷积神经网络前向通路的核心运算模块，不仅具有高密度的运算操作，而且具有高频率的运算重复性，因此，该模块的运行效率也会极大地影响整个卷积神经网络硬件加速性能的效果，是本文的重点设计对象。一方面，卷积神经网络是一种多层次拓扑结构，不同结构层之间，输入特征映射图与输出特征映射图的数量关系具有不确定性，以及卷积窗口和对应的卷积核尺寸大小的多样性，使得整个卷积神经网络拓扑结构在单一硬件平台上完全实现并行化运行是极为耗费资源的，设计难度大，不易实现，也是一种性价比较低的行为。另一方面，不同结构层之间的运算具有相对独立性和高度相似性，因此，可以通过采用反复调用单个或少量卷积运算模块的策略来实现完整的卷积神经网络硬件化加速。因此本文在设计时，从图像数据流驱动方式、硬件资源的角度出发，重点实现单个卷积运算模块，并结合卷积神经网络存在的结构特点和并行特性，进行优化和拓展。

4.2.1 Z 型卷积运算模块

在图像识别和图像处理领域中，对图像的读取扫描方式通常两种方式：逐行扫描和隔行扫描。逐行扫描，是指图像输入时，图像从左上角的第一行第一个像素点开始，逐个向右向下读取每一个像素值，直至扫描到图像的最后一行的最后一个像素点为止。隔行扫描，是指将输入图像以每隔一行像素点为一组，分成两组像素点（一般称一组像素点为一个扫描场或场），通常首先扫描所有奇数行像素，然后再扫描所有偶数行像素，每一行像素从左到右依次扫描。相比之下，逐行扫描比隔行扫描在显示效果上具有更好稳定性，也是目前市场应用中最为常用的图像扫描方式。

本文借鉴逐行扫描的图像输入方式，采用单数据流驱动的方式，设计一种 Z 型卷积运算模块。以3×3卷积核为例，详细说明 Z 型卷积运算模块的设计结构，其结构示意图如图 4-2 所示。

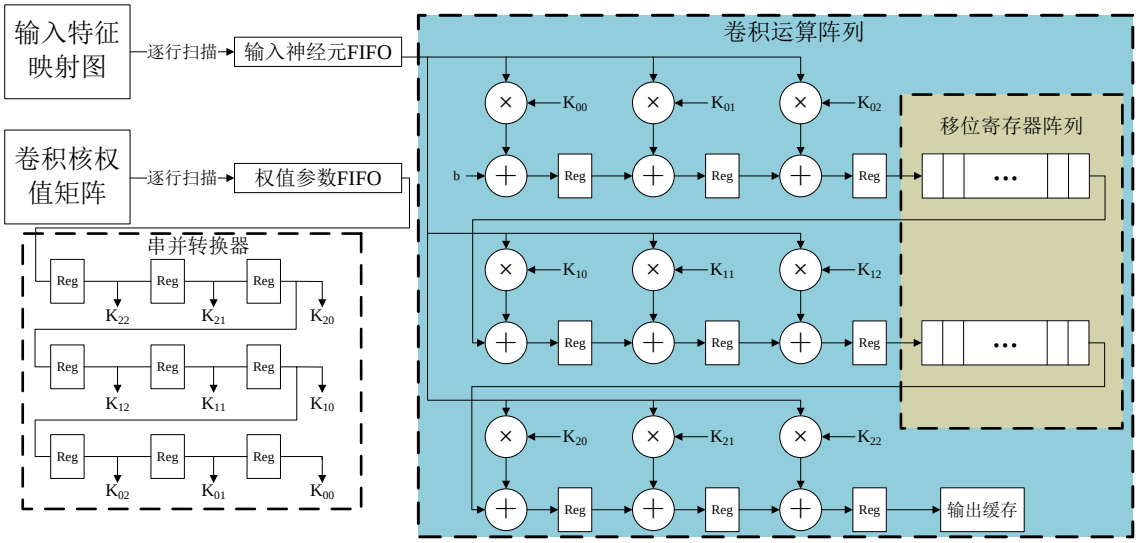


图 4-2 Z 型卷积运算模块结构示意图

Z 型卷积运算模块主要分两部分结构：一是传输部分，用于输入特征映射图、卷积核权值矩阵以及输出结果的传输；二是运算部分，用于神经元与卷积核的卷积运算。

传输部分：假设输入特征映射图尺寸大小为 $N\times N$，那么卷积运算模块需要两个 $N\times N$ 大小和3×3大小的片上存储单元，分别用于存储输入特征映射图的神经元数据和卷积核的权值参数。在两个片上存储单元后面分别连接一定深度和宽度的缓存 FIFO（输入神经元 FIFO 和权值参数 FIFO），用于匹配存储单元和运算单元之间的运行速率，进行输入数据缓冲，保证数据传输和运算的正确性，其中 FIFO 的深度受存储单元和运算单元的数据驱动速率的影响，FIFO 的

宽度大小设置成与输入数据的位宽一致。权值参数 FIFO 后面连接一个由寄存器组成的串并转换器，由于以3×3大小的卷积核为例，因此总共需要 9 个寄存器，寄存器输入输出端之间依次连接，且输出端分别输出卷积核权值参数的其中一个值，将输出值传输到后续的运算部分。在运算部分最后的输出端，需要添加一个片上缓存单元，如 RAM 和 FLASH，用于保存该卷积运算模块的计算结果。

运算部分：即卷积运算阵列。该部分主要的运算操作是输入特征映射图的神经元与卷积核的卷积操作，即乘加操作，因此需要 9 个乘法器和 9 个加法器，以并行的方式来完成一个3×3大小卷积窗口的卷积操作。为了提高数据驱动的流畅性，开发卷积运算的并行特性，在加法器之间添加流水线式寄存器，构成流水线式结构，提高运算部分的工作频率。由此，每行的 3 个乘法器和 3 个加法器构成了一个一维的卷积器。为了实现二维卷积窗口的卷积运算，需要将相邻的两个一维卷积器相连。由于输入特征映射图的神经元是按逐行扫描的方式输入，需要将相邻两行的神经元进行数据匹配，因此在前 2 个一维卷积器后面各添加一个深度为$N-3$的移位寄存器，作为相邻两个一位卷积器的数据通路，并且用于缓存计算的中间结果，进行数据匹配。

输入数据从传输部分传输到运算部分后，输入数据在卷积运算阵列中运行时，卷积窗口和运算数据的流动情况如图 4-3 所示。

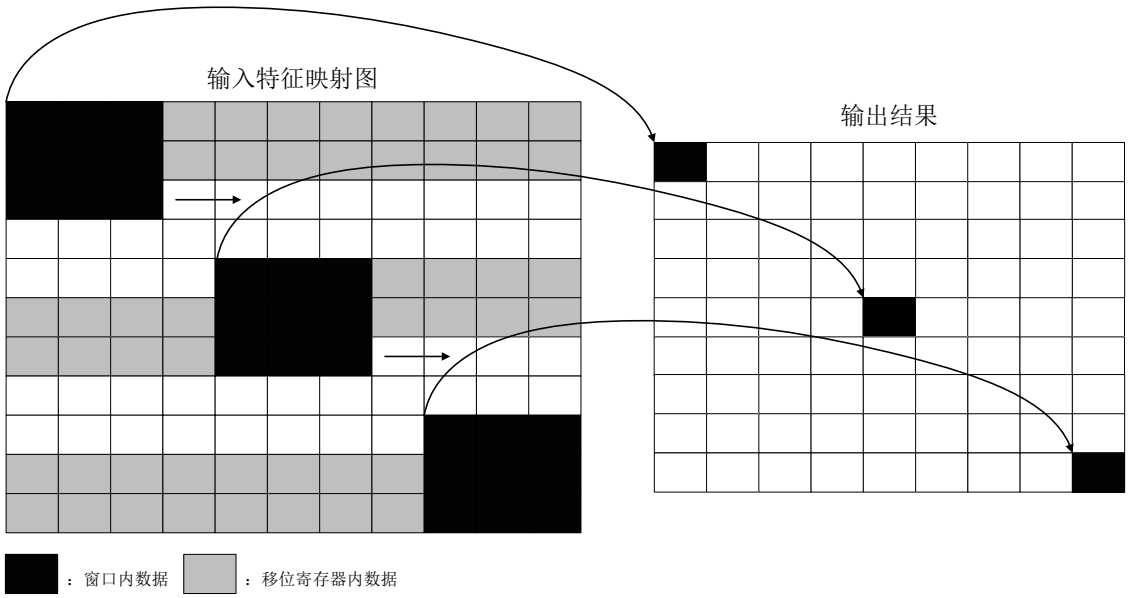


图 4-3 Z 型结构的卷积窗口和数据流动示意图

输入特征映射图按逐行扫描的方式，将输入神经元传输到卷积运算阵列中。当卷积运算阵列接收到输入神经元时，以广播式的方式将神经元传递到所有乘

法器，与对应的卷积参数做乘法运算，并通过流水线式的线型加法器结构将乘法结果逐级累加起来，得到最终的输出结果。由于流水线式结构在数据流开始驱动前，需要把所有的流水线式寄存器填充满，包括移位寄存器阵列，因此，在得到第一个有效输出数据前，存在一个数据填充的缓冲时间段。输入特征映射图的每一个神经元数据，在卷积运算阵列中按“Z”型方式传递，当输出第一个有效数据以后，每输入一个神经元，卷积窗口向右平移一个单位，就会计算得到一个输出结果，当所有输入神经元扫描完毕时，所有的输出结果也计算结束，该卷积运算模块的运算时间等于输入特征映射图的扫描时间。

在示意图 4-3 中，11×11大小的输入特征映射图与3×3大小的卷积核进行卷积运算，得到9×9大小的输出结果。输入特征映射图内的黑色窗口与输出结果内的黑色块一一对应，输入特征映射图的黑色窗口表示即将计算得到的卷积窗口，输出黑色块则为输入黑色窗口的特征表示。当输入特征映射图左上角的卷积窗口开始计算时，黑色块代表的神经元恰好与对应的卷积参数完成卷积运算，卷积窗口右边的灰色块代表的神经元则缓存在移位寄存器中，等待再次进入运算单元中。计算一个输入特征映射图时，卷积窗口从输入特征映射图的左上角开始，从左向右，从上到下，按“Z”型方式移动，直到所有的输入神经元至少被卷积窗口覆盖一次。计算每一个卷积窗口时，从窗口左上角第一个神经元开始，在卷积运算阵列中从左向右，从上到下，按“Z”型方式传递，直到卷积窗口右下角最后一个神经元进入到运算阵列中，完成该卷积窗口的卷积运算。

本文以 Altera Quartus II 作为可编程逻辑的设计环境，设计了5×5卷积窗口的 Z 型卷积运算模块，模块电路的寄存器转换级电路（Register Transfer Level，RTL）如图 4-4 所示。

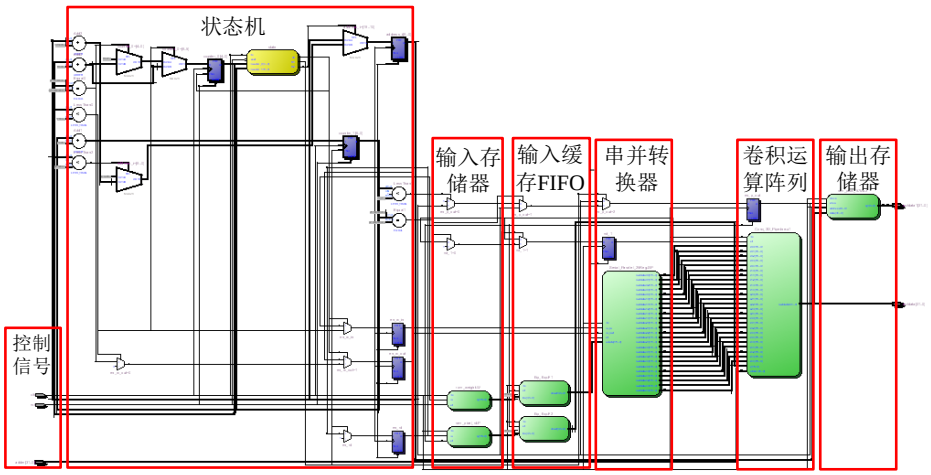


图 4-4 Z 型卷积运算模块 RTL 图

该 RTL 结构主要分 7 个子模块组成：控制信号、状态机、片上输入存储器、输入缓存 FIFO、串并转换器、卷积运算阵列和片上输出缓存。时钟信号和复位信号组成的控制信号，控制整个 Z 型卷积运算模块的正常运行和停止；Z 型卷积运算模块中的各个子模块是根据不同的事件触发控制不同模块的功能转变，因此状态机控制子模块的模块功能转换；输入存储器和输出存储器分别存储卷积运算模块的输入数据和输出结果；输入缓存 FIFO 缓存输入神经元和卷积核参数；多组寄存器构成的串并转换器实现输入的卷积核参数串行输入，并行输出；利用嵌入式乘法器、加法器和一系列寄存器组成卷积运算阵列，完成输入数据的卷积运算。

状态机是该卷积运算模块的核心控制结构，本文采用 Moore 型结构进行设计，转换图如图 4-5 所示。

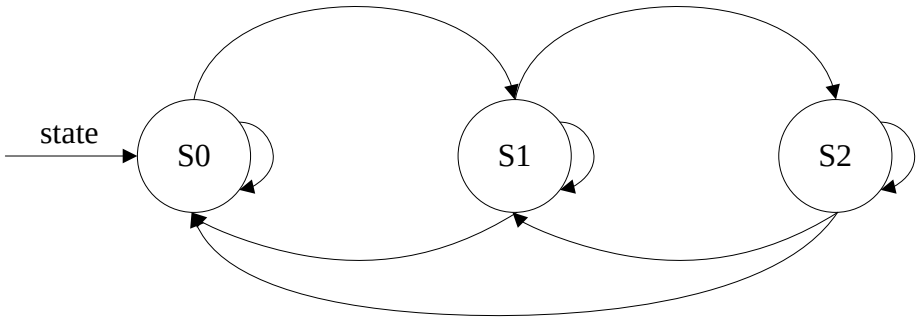


图 4-5 Z 型结构 Moore 型状态转换图

当一个事件触发或条件满足时，将会触发一个动作或者进行一次状态转移。该状态机总共有三种状态 S0，S1，S2。状态 S0：输入权值参数，实现串并转换器功能；状态 S1：串并转换器停止工作，输入神经元开始读取，卷积运算阵列开始运行；状态 S2：输出存储器开始运行，保存输出结果。在状态 S0 时，如果串并计数器未记满，那么保持状态不变；如果串并计数器记满，那么串并转换器并行输出结果，并跳转 S1 状态。在状态 S1 时，如果缓冲计数器未记满，那么保持状态不变；如果缓冲计数器记满，那么启动输出存储器，并跳转 S2 状态；如果接收到复位信号，那么跳转 S0 状态。在状态 S2 时，如果运算计数器未记满，那么保持状态不变；如果运算计算器记满且需更新权值参数，那么跳转 S0 状态；如果运算计算器记满且不需要更新权值参数，那么跳转 S1 状态。

利用 ModelSim 仿真软件对 Z 型卷积运算模块进行功能性仿真，验证其在功能实现上的正确性。在设计中采用 28×28 的输入神经元和 5×5 的卷积核作为仿真的样本数据。本文仿真输入的样本数据均设为 16 位的位宽。仿真图如图 4-6、4-7、4-8 所示。

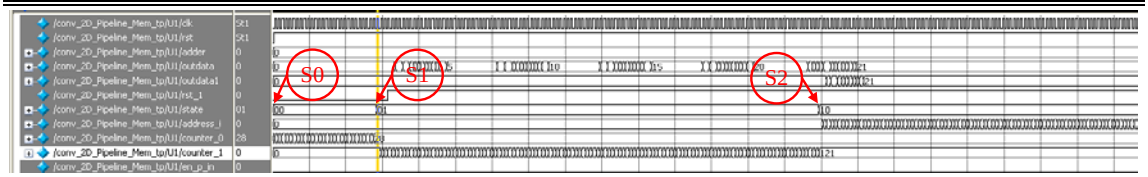


图 4-6 Z 型卷积运算模块仿真图 I

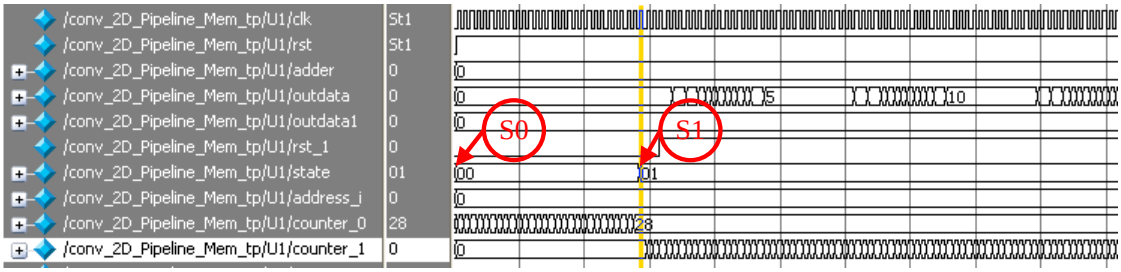


图 4-7 Z 型卷积运算模块仿真图 II

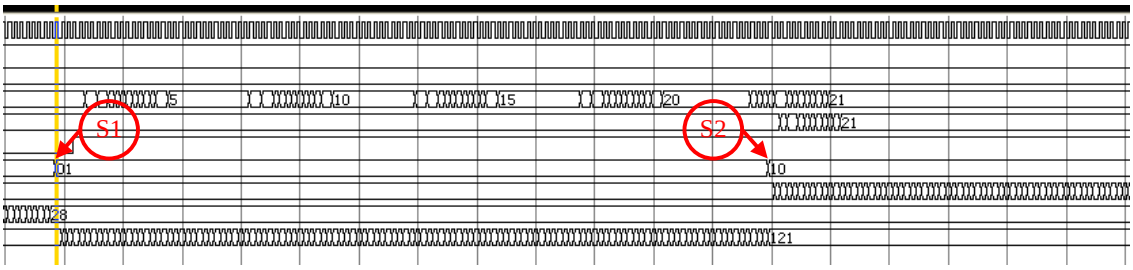


图 4-8 Z 型卷积运算模块仿真图 III

在 S0 状态时，串并计算器计数，只执行输入卷积参数的串并转换，不进行卷积运算，此状态下没有输出数据；当卷积参数串并转换输入到卷积运算阵列后，转入 S1 状态，缓冲计算器计数，开始将输入神经元扫描输入，并开始执行卷积运算阵列，此状态下产生数据，但是为缓冲阶段的无效数据，不进行保存；当缓冲阶段结束后，转入 S2 状态，运算计算器计数，开始启动输出缓存器，此状态下产生的是有效输出数据，保存到输出缓存器中，直到所有输入神经元都被输入，卷积运算结束，结束 S2 状态，转入 S0 状态或 S1 状态进行下一组输入神经元计算。

Z 型卷积运算模块可以实现输入特征映射图与输出特征映射图的单映射的对应关系，即一个输入特征映射图的卷积运算，若要在单卷积层中实现多映射或者全映射的对应关系，需要在时间上多次调用卷积运算模块。在一般的 FPGA 中，较稀缺的计算资源就是乘法器，而这种多次调用的实现方式能节省大量的硬件计算资源，但是需要添加额外的存储单元，用于存储每次调用卷积运算模块时产生的计算结果。如果采用高端的 FPGA 芯片，或者内部嵌入了大量独立的计算资源，那么可以将单个卷积运算模块进行并行化拓展设计，实现卷积神经网络多种形式的并行特性。图 4-9、图 4-10、图 4-11 分别给出了单输入多卷

积核并行、多输入单输出并行以及多输入多输出并行的实现方式，构成了三维卷积运算模块。

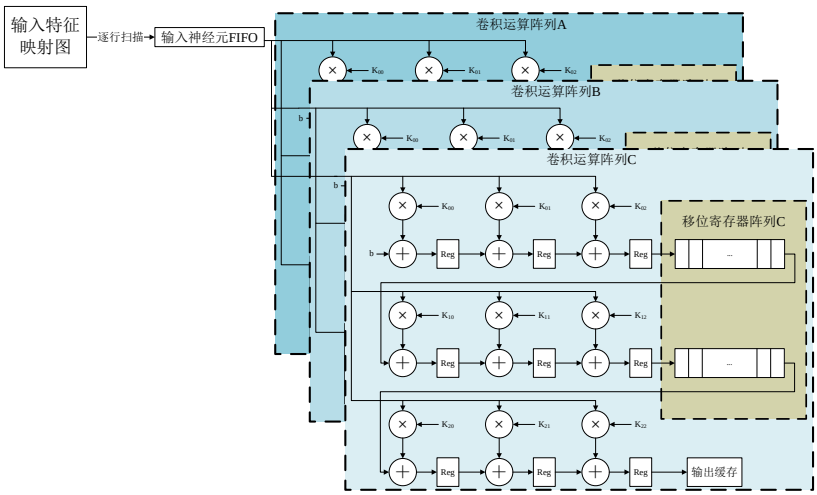


图 4-9 3D-Z 型单输入多卷积核并行运算模块

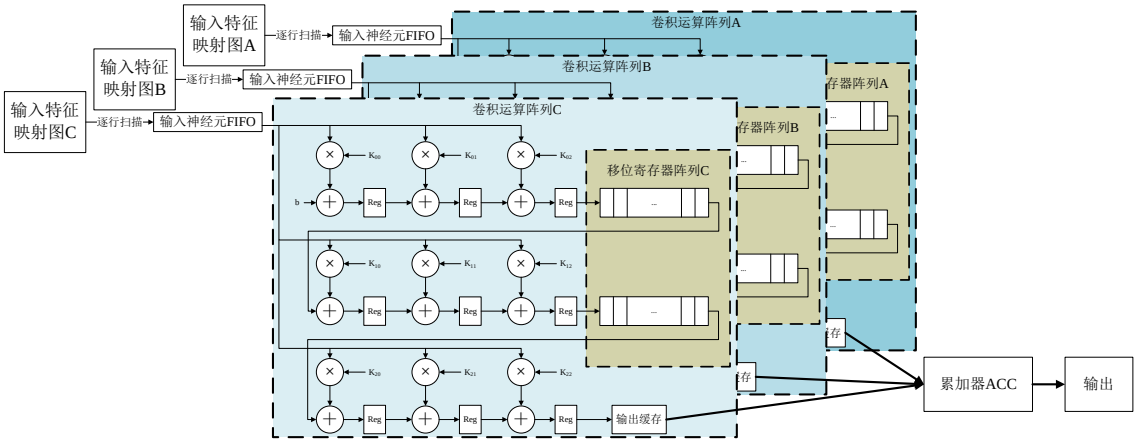


图 4-10 3D-Z 型多输入单输出并行运算模块

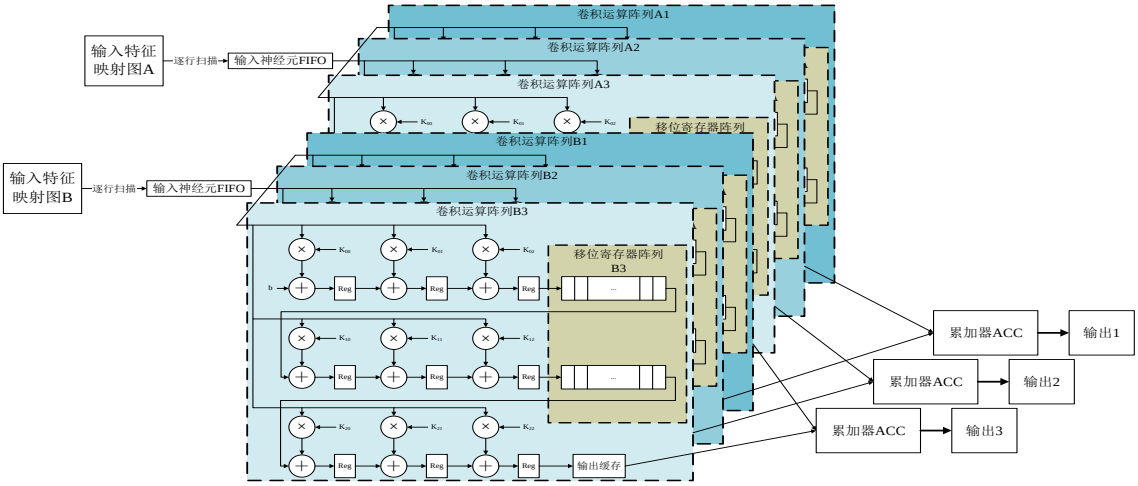


图 4-11 3D-Z 型多输入多输出并行运算模块

实现多卷积窗口的并行特性，更多的是依赖对读取输入神经元存储器的控制电路和计算资源的并行拓展，而 Z 型卷积运算模块采用的是单数据流驱动，逐行扫描的输入方式，因此该电路结构无法实现多卷积窗口的并行特性。其他类型的并行特性均在该电路结构中体现，如采用时间并行技术开发了卷积运算的并行特性。在实际应用中，可以将多种形式的并行特性混合使用，以达到更好的运算效果。

4.2.2 树型卷积运算模块

从 Z 型卷积运算模块的单数据流驱动，逐行扫描的输入方式出发，进一步开发运算模块的运算效率，将单数据流的数据驱动方式拓展成多数据流的数据驱动，提出一种树型卷积运算模块结构。以3×3卷积核为例，树型卷积运算模块的结构示意图如图 4-12 所示。

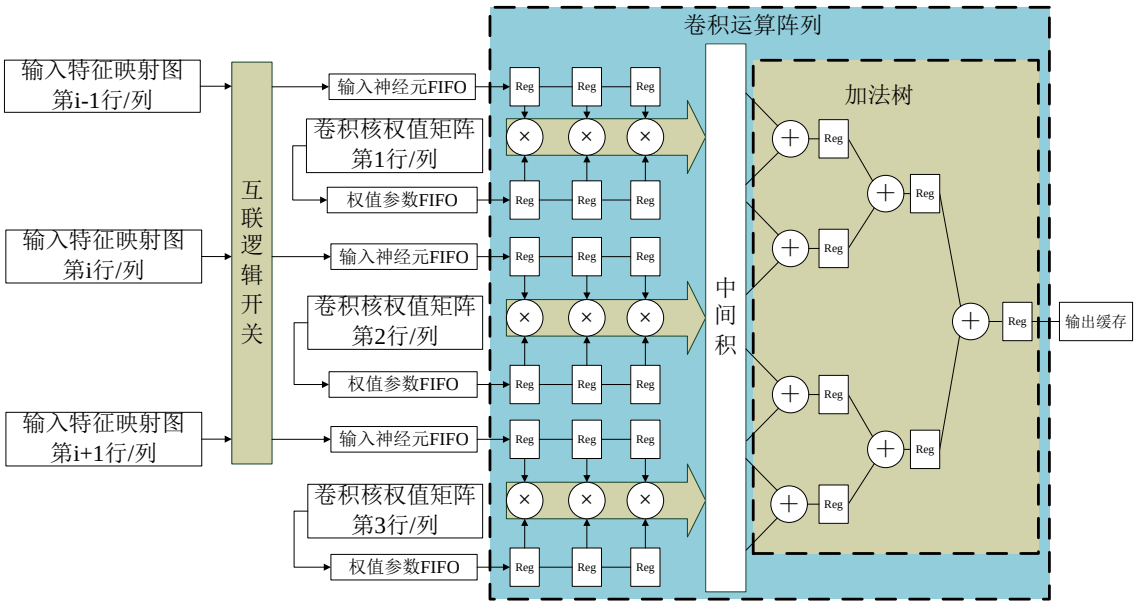


图 4-12 树形卷积运算模块结构示意图

树型卷积运算模块同 Z 型卷积运算模块结构相似，同样分两部分结构组成：一是传输部分；二是运算部分。每部分结构功能与 Z 型卷积运算模块的类似。树型结构需要将输入神经元和卷积核参数按行或列分组（下文以按行分组为例说明），按组进行分块存储，以多数据流驱动的方式将数据并行传递到运算部分，Z 型卷积运算结构并不适合这种多数据流驱动的方式；在卷积运算阵列中，树型结构采用分布式并行乘法结构得到乘法结果，乘法结果再传输到加法树中进行累加操作，得到输出结果，加法树每一层加法器之间添加流水线式寄存器，构成流水线式加法树结构。在输入端和缓存 FIFO 之间添加互联逻辑开关，是

为了跳变输入神经元与权值参数的对应关系，以便卷积窗口移动。

树型卷积运算模块开始运行，输入数据从存储单元内数据分组，到传输部分数据传输，再到运算部分在卷积运算阵列中运算，卷积窗口和运算数据的流动情况如图 4-13 所示。

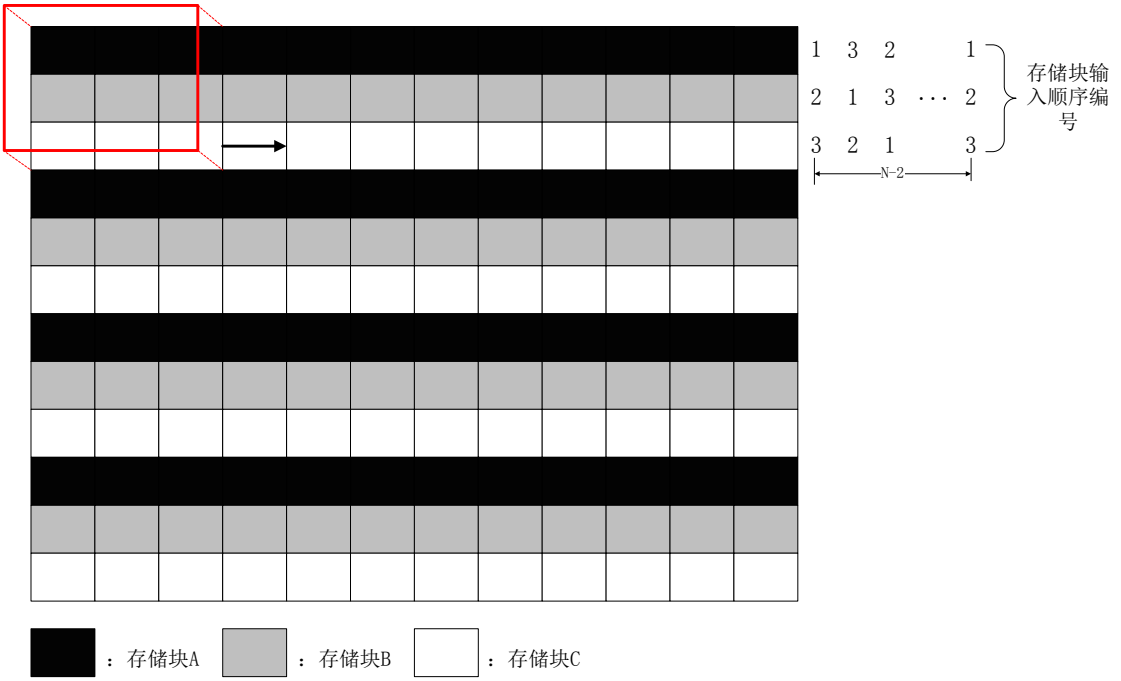


图 4-13 树型结构的卷积窗口和数据流动示意图

输入特征映射图神经元按卷积核尺寸进行分组。在图例中，输入神经元按行切割，每间隔两行的神经元被分为一组，共分成 3 组，如$1+3i$，$2+3i$，$3+3i$行的神经元被单独分成一组，同一组的神经元被存储在同一存储单元内，i为非负整数。若输入特征映射图的行数无法被 3 整除，则用数据 0 填补。同样，权值矩阵按行分组 3 组。图 4-13 中，相同颜色的方块代表同一组内的输入神经元，存储在同一存储单元内，黑灰白神经元存储单元分别编号存储块 A、存储块 B 和存储块 C。在起始状态，存储块 A、B、C 按顺序与权值矩阵 1、2、3 行对应，即存储块 A、B、C 的输入神经元与权值矩阵 1、2、3 行的权值参数对应相乘，此时记存储块的排列顺序为 1、2、3。当树型结构中的所有寄存器单元被数据填充满之后，流水线结构开始启动，产生第一个有效输出数据，并且之后每输入一次数据都会产生一个输出结果，此时卷积窗口向右开始平移。当存储块中的第一行输入神经元都被输入后，改变存储块的排列顺序，存储块 A、B、C 的排列顺序更新为 3、1、2，即存储块 A、B、C 的输入神经元与权值矩阵 3、1、2 行的权值参数对应相乘，此时卷积窗口下移一行，每输入一次数据产生一个输出结果，窗口向右平移。重复上述操作，直到卷积窗口覆盖所有的输入神经

元。在更新存储块的排列顺序时，可以发现卷积窗口存在数据重复性，每次更新顺序，只需读取卷积窗口最后一行数据，其他数据可以从缓存中直接调用，减少对片外存储器的重复操作。

可以看出，树型结构与 Z 型结构相比，消耗的计算资源相差无几，但是完成一个 $N\times N$ 输入特征映射图与 $k\times k$ 卷积核的运算，Z 型结构需要消耗 N^2 的运算时间，而树型结构只需要消耗 $N\times(N-K+1)$ 的运算时间，在计算效率上得到一定的提升。但是树型结构需要添加 $N\times(K-1)$ 大小的片上缓存单元，用于存储卷积窗口下移过程中重复调用的输入神经元，这也是一种资源面积换取计算速度的具体体现。

与 Z 型结构电路设计方式相同，设计了 5×5 卷积窗口的树型卷积运算模块，模块电路的 RTL 图如图 4-14 所示。

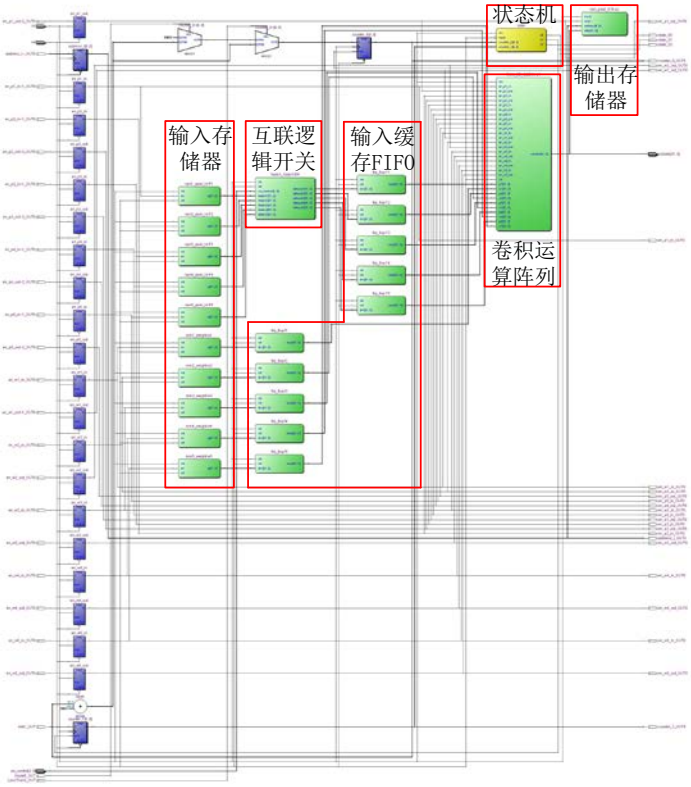


图 4-14 树型卷积运算模块 RTL 图

树型卷积运算模块分多个子模块共同组成，除了控制信号以外，包括输入存储器、输出存储器、互联逻辑开关、输入缓存 FIFO、卷积运算阵列和状态机。状态机作为整个卷积运算模块的核心控制结构，起到了对各个子模块的功能状态切换和子模块之间的协调调度的作用。树型结构子模块的功能与 Z 型结构的子模块功能类似，可以参考上述 Z 型结构的详细分析与介绍，在此不再一一赘

述。状态机的状态转换图如图 4-15 所示。

该状态机总共有 7 种状态切换，S0~S6。状态 S0~S4：互联逻辑开关的通道选择，用于输入神经元存储单元与卷积矩阵的对应顺序切换；状态 S5：卷积运算阵列内部的串并转换寄存器对输入数据是否进行串并转换功能的选择。除 S0 状态以外，状态 S1~S4 均需要直接进行输入数据串行输入，并行输出；状态 S6：卷积运算阵列中启动乘法和加法运算操作，并将输出结果缓存。

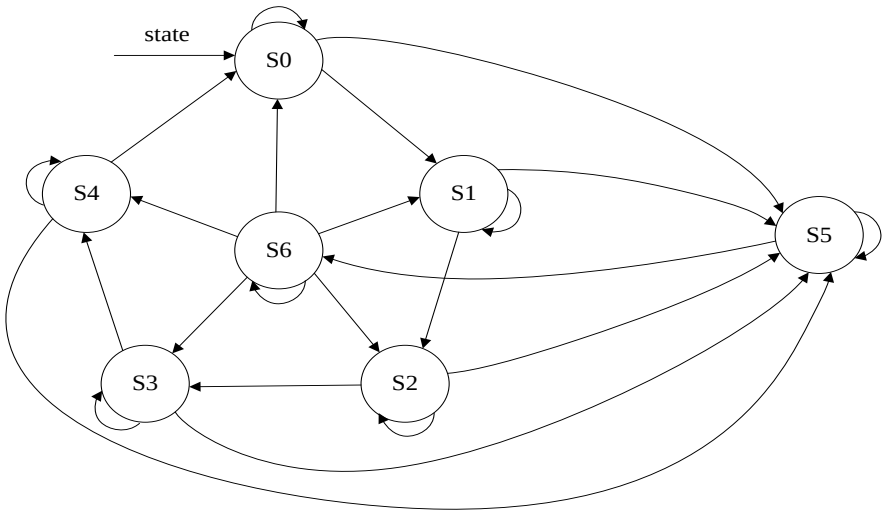


图 4-15 Z 型结构状态转换图

同样在设计中采用 28×28 的输入神经元和 5×5 的卷积核，作为仿真的样本数据，进行模块功能性仿真，仿真图如图 4-16 所示。开始启动时，树型卷积运算模块首先进入状态 S0，互联逻辑开关按初始状态进行通道开关选择，通道选择完毕后进入状态 S5，将输入神经元和卷积参数并行传递到运算单元中，最后进入到状态 S6，直到一行数据计算完毕，重新返回到逻辑开关选择，重复上述状态轮回，直到所有输入神经元被输入计算。

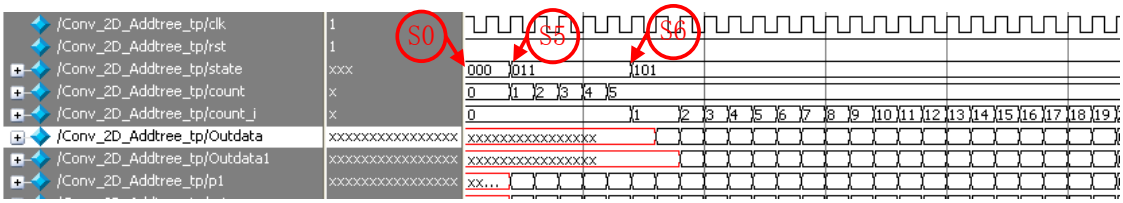


图 4-16 树型卷积运算模块仿真图

单个树型卷积运算模块可以实现单输入卷积运算，在不考虑硬件资源的限制因素下，仿照 Z 型卷积运算模块的设计思路，可以将单个树型卷积运算模块进行并行化拓展设计，实现卷积神经网络多种形式的并行特性。图 4-17、4-18、4-19、4-20 分别给出了单输入多卷积核并行、单输入多卷积窗口并行、多输入

单输出并行和多输入多输出并行的实现方式，构成了三维卷积运算模块。

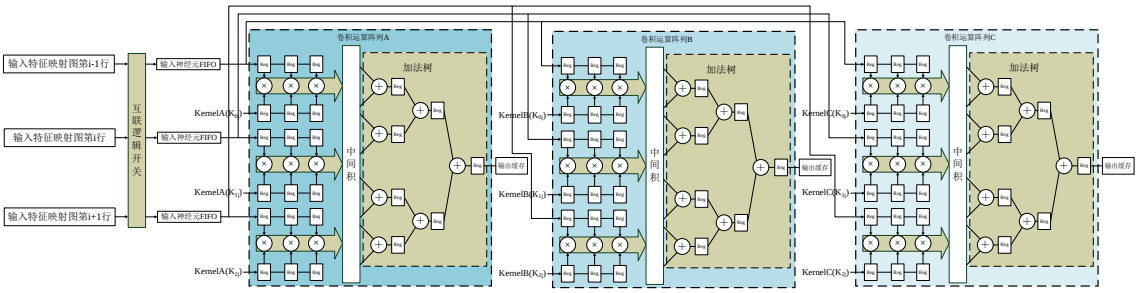


图 4-17 3D-树型单输入多卷积核并行运算模块

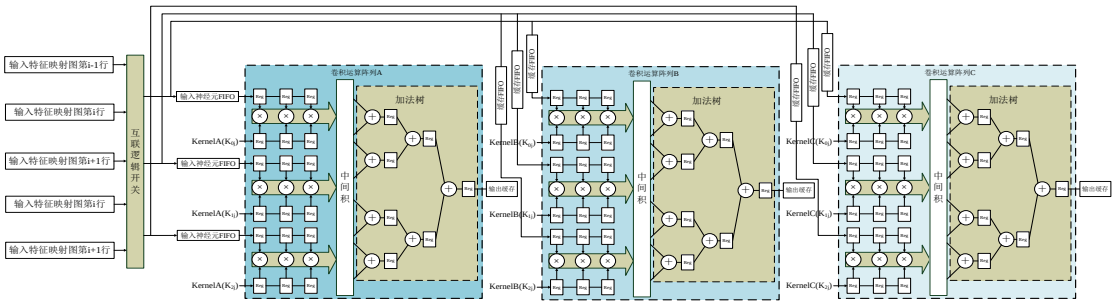


图 4-18 3D-树型单输入多卷积窗口并行运算模块

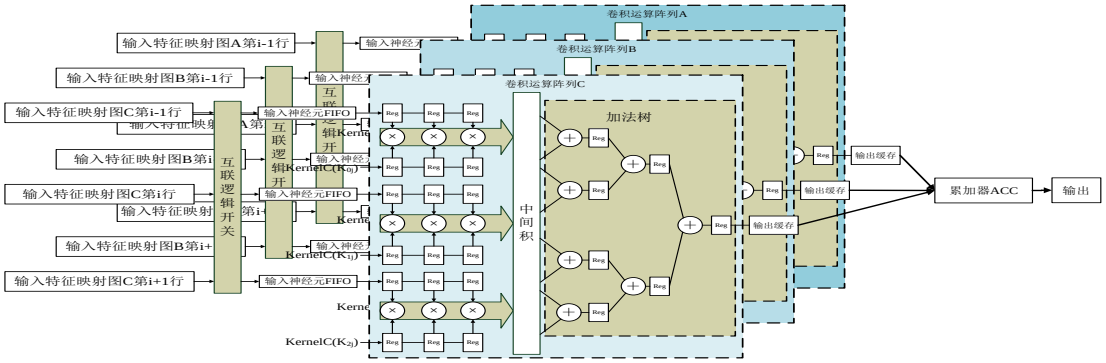


图 4-19 3D-树型多输入单输出并行运算模块

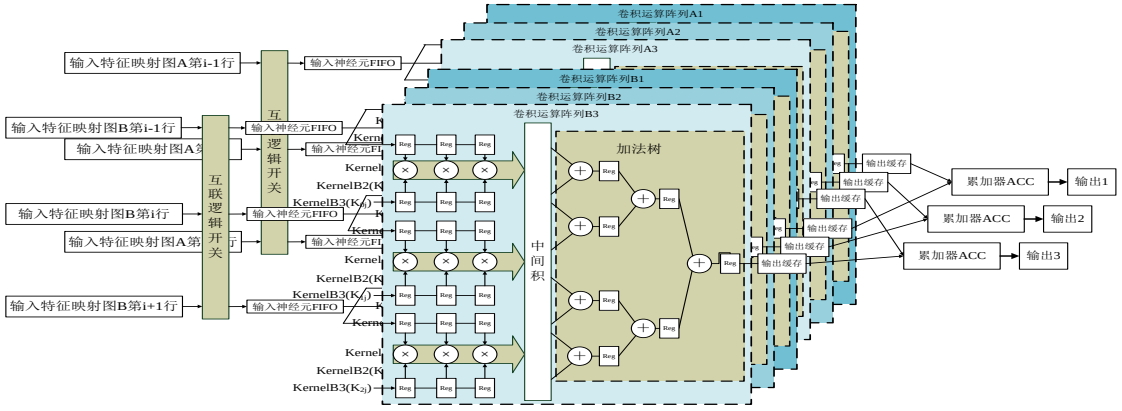


图 4-20 3D-树型多输入多输出并行运算模块

树型结构能实现更多形式的并行特性，而且在运算速度上比 Z 型结构更有一定优势，但是它需要更多的控制电路和存储空间进行配合。在实际应用中，选择加速运算或选择节约资源，可根据开发平台视情况而定，也可以将两种卷积运算结构，多种形式的并行特性混合使用，以达到一定程度的互补效果，提高运算性能。

4.3 抽样运算模块

抽样层一般连接在卷积层之后，是对特征映射图的二次特征提取，起到降低特征映射图的分辨率，减少数据规模，简化网络结构的作用。由于抽样操作是一种模糊的滤波变化，使得网络增强了自身抗空间扭曲的能力。从上一章对卷积神经网络的分析中不难发现，抽样层最主要的运算操作就是进行数据下采样，包括最大采样、均值采样和随机采样。本文选用最大采样的计算策略，即选取抽样窗口内的最大值作为该区域神经元的特征，对抽样运算模块进行逻辑电路设计。

抽样层与卷积层在模型结构上非常相似，在特征映射图中，数据流和运算窗口的移动规则也大致相同。两者之间的窗口运动情况，最明显的差异就在于运算窗口移动的步进值不同，卷积层为了获取更精细、更充分的特征，会充分利用每一个，每一组的有效数据，因此卷积窗口的步进值一般为 1；而抽样层则是为了降低特征维度和特征映射图的分辨率，不希望或减少数据的重叠部分，因此一般抽样窗口的步进值与抽样窗口的尺寸保持一致。

依据抽样窗口的步进特点，本文采用地址索引的策略来跳跃式索引存储器内的输入数据，以代替原先逐行扫描的顺序驱动方式。地址索引，根据需要读取数据的存储地址，然后索引相应输入神经元，再将输入神经元传递到运算比较单元进行后续处理。抽样运算模块在 FPGA 中实现的 RTL 图如图 4-21 所示。

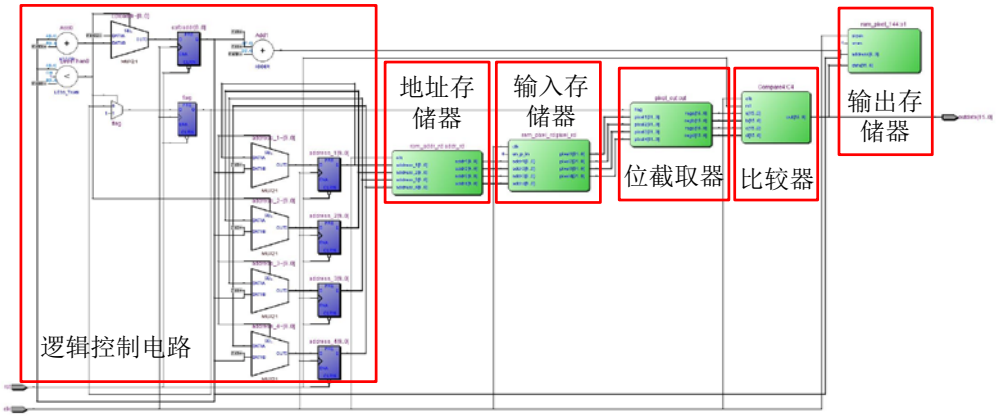


图 4-21 抽样运算模块 RTL 图

抽样运算模块除控制信号以外，主要分 6 个子模块组成：逻辑控制电路、地址存储器、输入存储器、位截取器、比较器和输出存储器。输入/输出存储器功能与其他模块电路设计的类似。地址存储器存放的是存放在输入存储器数据对应的存储地址。逻辑控制电路则是控制地址存储器和输出存储器的地址信号端，使存储器按一定逻辑顺序读取相应地址的数据或将数据写入相应地址。位截取器，是为了配合卷积运算模块对数据的有效位截取，即将冗余的数据位去除，保留足够精度的数据。本文在设计过程中，将位截取之后数据位宽与原始数据的位宽保持一致，这样抽样层运算得到的输出数据可以直接作为卷积层的输入数据输入，而不需要附加额外的预处理操作。比较器是一个四输入的比较器，选取抽样窗口内的最大值。抽样运算模块的功能仿真图如图 4-22 所示。

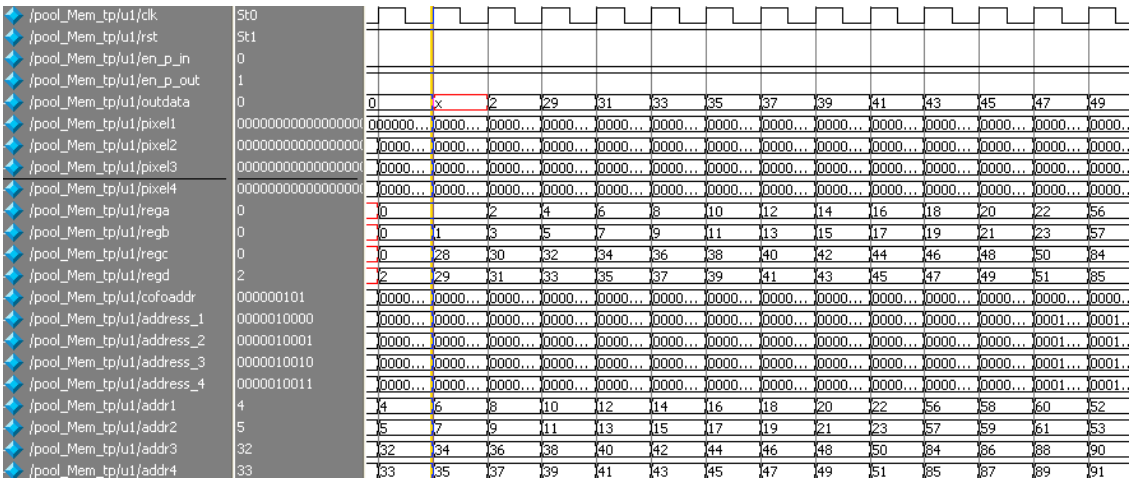


图 4-22 抽样运算模块仿真图

抽样运算模块运行，首先通过逻辑控制电路，对地址存储器内预先存储的地址 addr1~addr4 进行读取，将读取的地址作为地址信号传递给存储输入神经元的输入存储器，读取对应位置上的输入数据 pixel1~pixel4，其次将输入数据进行有效位截取，得到与原始输入数据位宽一致的 16bit 数据，保持在寄存器 rega~regb，然后将位截取后的数据进入四输入比较器，选取最大值，最后将比较的最大值保存到输出存储器中保存。

抽样运算模块同样可以延续卷积运算模块的设计思路，采用分片存储，多数据流驱动的方法进行设计。若采用将 2×2 的抽样窗口，那么输入神经元需要分片存储在 4 个存储单元内，抽样窗口相同位置上的输入神经元存储在同一个存储单元内，并行读取四个存储器的数据，进行卷积窗口的下采样操作。这种设计方法只需将树型卷积运算模块电路稍加修改即可，本文不进行具体的硬件电路设计。

4.4 激活函数模块

激活函数是实现卷积神经网络的一个重要环节。为了增加网络对非线性系统的处理能力，通常在网络中添加非线性的激活函数，通过对神经元阈值的判断调节神经元的兴奋度和活跃性。激活函数在传统的人工神经网络中应用十分广泛，其函数形式也有很多种类型。常见的激活函数有分段函数、Sigmoid 型函数、ReLU 型函数等，在实际应用都有好的表现。

以 FPGA 为代表的数字电路中实现复杂的激活函数，不仅需要考虑实现的代价问题，而且需要考虑运算的性能情况。在实现代价方面：在数字硬件电路中实现非线性函数，常用的方法有分段线性拟合、泰勒展开逼近、多项式近似、CORDIC 算法、查找表法等[55]。泰勒展开逼近需要将非线性函数展开成多个阶乘项，随着展开项和阶数增加，对硬件资源，尤其对乘法器消耗是巨大的。多项式近似同样需要考虑对乘法器的消耗，而且计算速度较慢，不适合要求计算效率的硬件加速系统中应用。CORDIC 算法虽然消耗的计算资源较少，但是运算耗时非常严重，而且存在精度不高的问题。查找表法理论上可以实现任意精度下的任意非线性函数，操作简单方便，但是随着数据精度提升，查找表会被不断细化，消耗大量的存储空间。分段线性拟合是将复杂的函数分段处理，每段函数采用线性函数进行拟合。相比其他方法，分段线性拟合不需要过多的计算资源和存储空间。在运算性能方面：在传统的神经网络中，通常使用一些平滑的非线性函数，如 Sigmoid 函数，但是在训练学习方面存在计算复杂，收敛速度慢等问题。近年来，ReLU 函数得益于其更接近于生物神经元的激活特性，并且函数结构简单、计算便捷和收敛速度快等优点，目前被广泛应用于多层神经网络中。最近又出现了许多经过改良的 ReLU 函数，如 Leaky-ReLU、P-ReLU、R-ReLU 等，但是效果众说纷纭，暂时没有清晰的定论。综合考虑以上因素，本文以 ReLU 函数作为卷积神经网络的激活函数，采用分段线性拟合的方法在 FPGA 上实现，对激活函数模块进行设计。

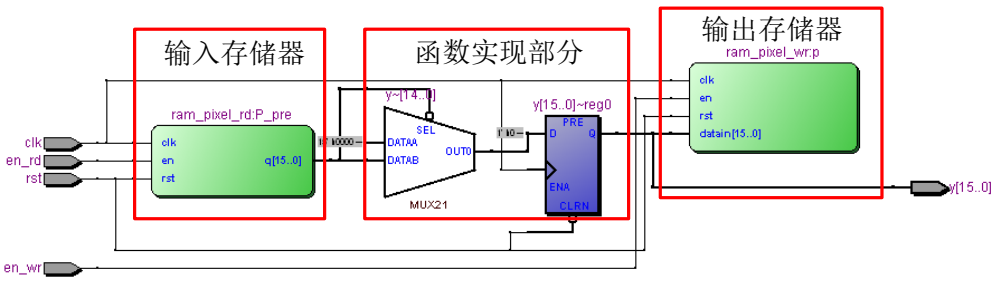


图 4-23 激活函数模块 RTL 图

从 ReLU 函数的数学表达式，很显然可以看出，输入信号大于 0 和小于 0 时，输出表达式不同，因此可以把 ReLU 函数分成两段拟合，一段是输入神经元为负值信号，另一段是输入神经元是非负值信号。这是两段简单的线性函数，在 FPGA 上非常容易实现。激活函数模块的 RTL 图如图 4-23 所示。

激活函数模块除了控制信号以外，主要包括输入缓存器、输出缓存器和函数实现部分。输入、输出缓存器与其他模块功能相仿，函数实现部分由一个比较器和一个 D 触发器组成。输入缓存器输入待处理的神经元数据，然后将输入数据与 0 值比较，当输入数据小于 0，则 D 触发器的输入端置 0，输出信号也为 0，当输入数据大于等于 0，则 D 触发器的输入端为输入数据值保持不变，输出信号也为输入数据，最后将 D 触发器的输出端数据传递到输出存储器中保存。激活函数模块的功能仿真图如图 4-24 所示。

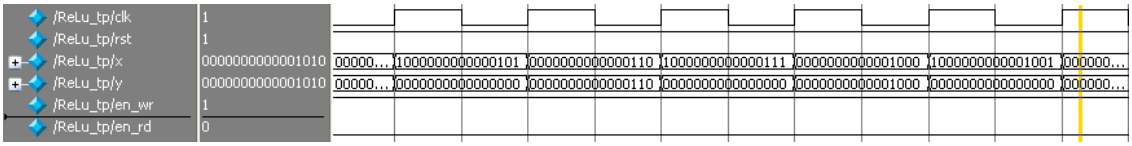


图 4-24 激活函数模块仿真图

在仿真中，采用的是 16bit 的神经元样本数据，最高位是符号位，x 信号是输入存储器的输出端，y 信号是输出存储器的输入端，即模块的输出端。信号 en_wr 和 en_rd 可以与其他模块连接。从功能仿真图和 RTL 图中可以看出，在 FPGA 中采用分段线性拟合的方法非常适合实现 ReLU 函数，不仅硬件电路简单，占用资源少，而且实现效果明显，也可根据实际应用需要调整数据精度，以达到满意的运算效果。

4.5 缓存结构

本文将卷积神经网络模块化划分，以串行模式多次调用各个运算模块，以实现整个卷积神经网络前向通路，而每次调用运算模块，势必需要添加缓存单元对中间计算结果进行数据缓存。为了减少缓存中间计算结果所造成的时间损耗，本文对运算模块中所有涉及到的缓存单元进行乒乓结构设计。乒乓操作的核心思想就是以双缓冲器的交互运行方式将数据传输与计算时间重叠[56]。乒乓结构模型在上文中已经介绍，在此不在重复。

本文以单口 RAM 作为数据缓冲模块，设计了双缓冲单元的乒乓结构，其 RTL 图如图 4-25 所示。根据乒乓操作流程，乒乓结构 RTL 电路分 4 个模块，分别为逻辑控制电路、输入数据流选择开关、数据缓冲模块 A、B 和输出数据流选择开关。其中，逻辑控制电路控制输入输出开关的导通状态和切换时序，

以及缓冲模块的使能信号。

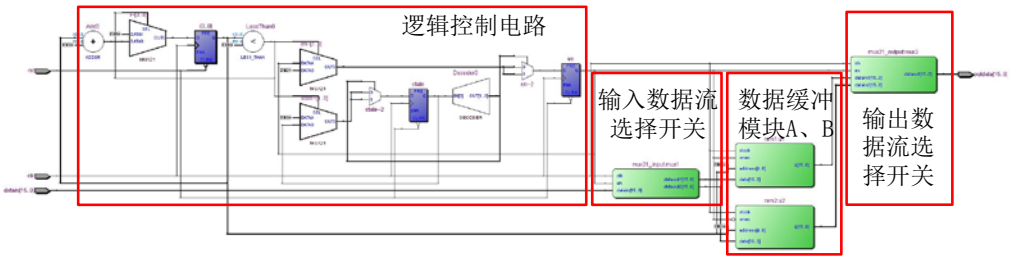


图 4-25 乒乓结构 RTL 图

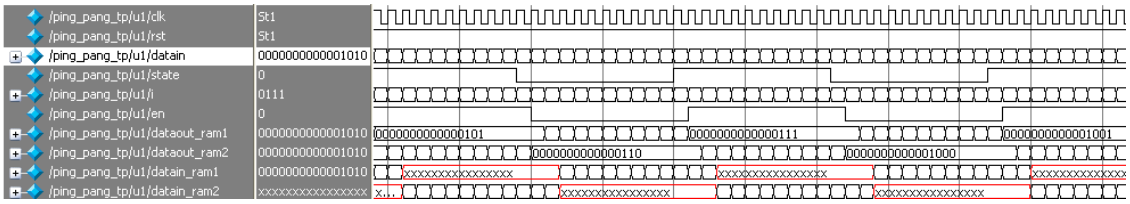


图 4-26 乒乓结构仿真图

乒乓结构的仿真图如图 4-26 所示。由图中可以看出，输入数据流连续不断传输，两个 RAM 存储器则相互交替存储和读取数据，以实现一个连续的数据流处理过程。这种高速的单数据流按乒乓操作分流处理，其实际上是一种以面积与速度互换的一种表达方式。

4.6 本章小结

本章以模块化电路设计为核心内容。首先，以 FPGA 为硬件实现平台，对卷积神经网络硬件电路实现进行整体架构设计和模块化划分，确定数据流传输和运行情况。其次，对卷积神经网络运算过程中最核心的卷积运算模块进行详细的硬件电路设计与功能性仿真，包括 Z 型结构和树型结构设计实现，并结合并行特性对两种卷积运算模块进行并行性拓展设计与优化。然后，对抽样运算模块和激活函数实现模块进行硬件电路设计与仿真验证。最后，对整个卷积神经网络实现过程中涉及到的缓存结构进行乒乓结构优化。本章完成卷积神经网络硬件实现的关键模块电路设计和缓存结构优化。

第 5 章 实验验证与分析

本章将搭建实验验证平台，针对卷积神经网络设计的关键电路模块和完整电路结构进行实验验证。首先，根据已有的实验条件，搭建 FPGA 异构系统。然后，以手写数字识别为应用实验，对卷积神经网络的每一层结构进行分层实现，并将 FPGA 实验数据与软件实验数据的进行实验对比，分析系统性能。最后，对各模块电路的功耗和资源使用情况进行分析。

5.1 实验平台搭建

5.1.1 实验环境介绍

卷积神经网络的实现过程主要分两个阶段进行，第一阶段是网络的学习训练，这一阶段主要由软件来完成，本文由 CPU 来训练；第二阶段是网络的识别分类，这一阶段由硬件来实现，本文采用 FPGA 加速。因此，本节从两方面来介绍实验环境：FPGA 和 CPU。

FPGA：本文采用的 DE-70 开发板作为 FPGA 硬件实现系统平台，开发板集成了一块 ALTERA 公司 Cyclone II 系列的低成本 FPGA 芯片，90nm 工艺，并提供嵌入式乘法器单元，属于低成本的 FPGA，适用于低中端应用市场。DE2-70 开发板如图 5-1 所示。

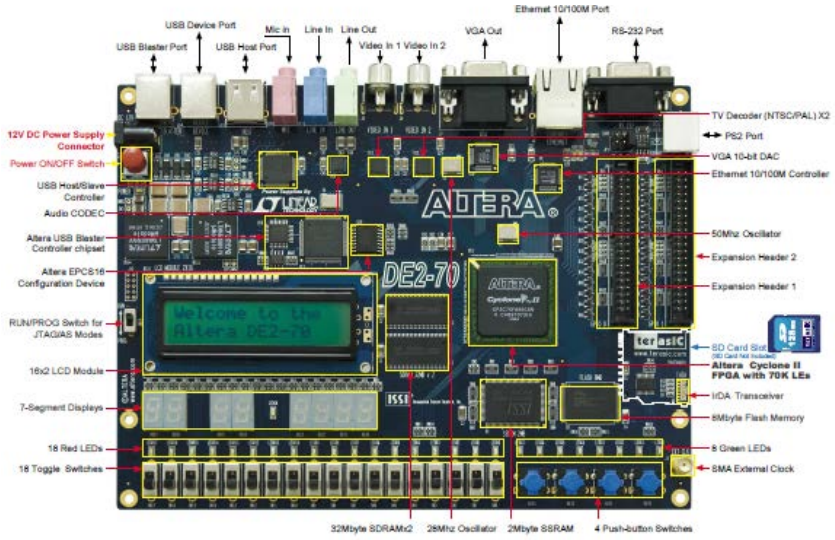


图 5-1 DE2-70 开发板示意图

DE2-70 开发板提供 50MHz 和 28.63MHz 的时钟源，2M 静态随机存储器

SSRAM，8M 闪存 FLASH，两片 32M 动态随机存储器 SDRAM，300 个 9×9 的嵌入式乘法器，4 个锁相环，另外还集成 SD 卡槽、串口 RS-232、USB、以太网、开关等一系列 FPGA 的外围设备。

开发工具采用 ALTERA 公司提供的 Quartus II 作为 FPGA 开发软件，支持 ALTERA 的 IP 核调用，简化设计复杂度，加快设计速度。Modelsim 作为第三方软件，对 FPGA 设计的模块电路进行功能性仿真验证，并配合嵌入式逻辑分析仪测试模块电路的信号时序。PowerPlay Early Power Estimator 和 PowerPlay Power Analyzer 分别作为设计初期和设计中后期的功耗分析工具，对 FPGA 设计的模块电路进行功耗评估。

CPU：为了训练卷积神经网络和进行对比实验，本文 CPU 采用 intel Core i5-3230M 双核处理器，主频 2.6GHz，32 位操作系统。软件工具采用 MATLAB R2015a 版本，支持包括卷积神经网络在内的多种深度学习算法的工具箱。

5.1.2 实验样本集与网络结构

本文以手写数字为图像识别的代表性应用，作为实验的应用背景。MNIST 数据集中[57]包含了 70000 张手写数字样本，其中有 60000 张手写数字图像作为训练样本集，10000 张手写数字图像作为测试样本集。每张手写数字图片不是标准图像文件，而是进过规范化处理后压缩保存在二进制文件的图像数据。每个样本图像均是分辨率为 28×28 的灰度图片。

由于手写数字样本图像的尺寸大小均为 28×28，为了配合卷积神经网络的输入层，将样本图像作为输入数据直接输入，而不做过多额外的预处理操作，本文采用的实验网络结构如图 2-7 所示。

输入层是 28×28 的手写数字图片，共 784 个像素点。

第一层是卷积层，采用 5×5 大小的卷积核，输出 6 个特征映射图，每个特征映射图的尺寸为 24×24，共 3456 个神经元。

第二层是抽样层，采用 2×2 大小的采样窗口，输出 6 个特征映射图，每个特征映射图的尺寸为 12×12，共 864 个神经元。

第三层是卷积层，采用 5×5 大小的卷积核，输出 12 个特征映射图，每个特征映射图的尺寸为 8×8，共 768 个神经元。

第四层是抽样层，采用 2×2 大小的采样窗口，输出 12 个特征映射图，每个特征映射图尺寸为 4×4，共 192 个神经元。

第五层是全连接层，采用全连接的方式，输出 100 个神经元。

第六层也是全连接层，采用全连接的方式，输出 10 个神经元。

输出层紧随其后，通过分类器将 10 个神经元作为网络输出结果输出，以区

分手写数字 0~9。

5.1.3 异构平台搭建

本文采用“FPGA+CPU”的异构体系作为本次实验验证平台，异构系统如图 5-2 所示。

CPU 作为系统的主控制器，主要有以下几点任务：一是完成原始图片简单的预处理操作，如原补码变化、平滑处理、边缘检测等；二是连接 CPU 与 FPGA 之间的通信，将测试图片、CPU 训练好的网络参数、控制指令和程序通过通信接口下载到 FPGA 保存；三是接收并保存 FPGA 的计算结果。FPGA 实现卷积神经网络前向运算，由多个模块共同协调完成。MCU 是 FPGA 的主控制器，向其他模块下发相应指令。Communication Interface 是 FPGA 标准通信接口，与 CPU 进行通信。Local Memory Buffer 存储当前运算所需的数据。External Memory Controller 控制片外存储芯片，本文采用 SDRAM、SSRAM 和 FLASH 片外存储器，下文均以 SDRAM 为例说明。Parallel Computing Arrays 是并行运算阵列。Resource Manager 是资源管理器，进行 FPGA 片内资源管理，模块调度和工作切换等任务。

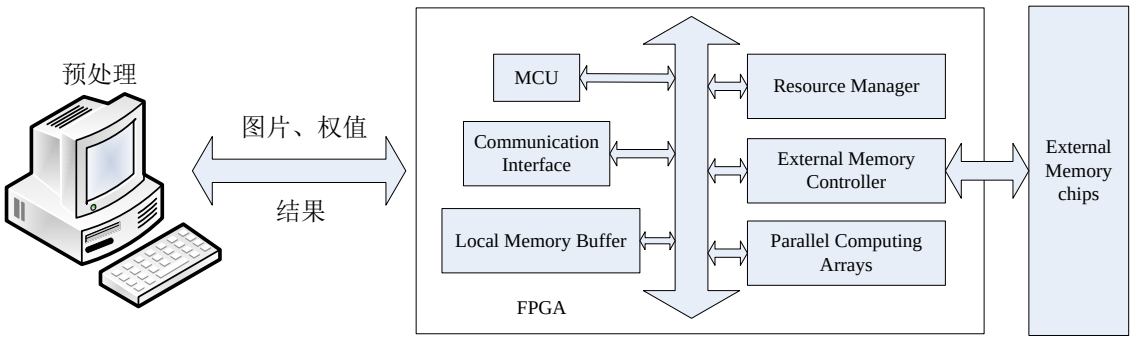


图 5-2 异构系统结构框图

系统的整体运行由 CPU 控制。CPU 向 FPGA 下发测试图片、网络参数，控制程序和指令。图片数据和网络参数保存到片外 SDRAM 中保存，指令和控制程序存于 MCU。当 FPGA 开始运行时，首先将网络第一层所需的运算数据和网络参数加载到片内存储 RAM 和 ROM 中，然后控制并行运算阵列中的运算单元，调用 Local Memory 内的数据进行计算，将计算得到的结果返回片外 SDRAM 中保存。第二层开始计算前，搬运上一层的计算结果到 Local Memory，继续进行第二层运算。如此循环，直到卷积神经网络所有运算结束，最终计算结果返回到 CPU 保存。从片外存储器搬运到片内存储器的数据，若存在数据重用，那么将重用的数据保存在 Buffer 缓存，方便数据直接调用。

本文采用层间的串行模式实现整个卷积神经网络的硬件加速。由于本文采用小规模网络做测试实验，在 FPGA 硬件资源充足的情况下，可以把卷积层和抽样层的级联形式作为一个计算单元，因此该网络只有两个计算层。数据在 CPU、FPGA、SDRAM 之间的交互握手式的传递过程如图 5-3 所示。

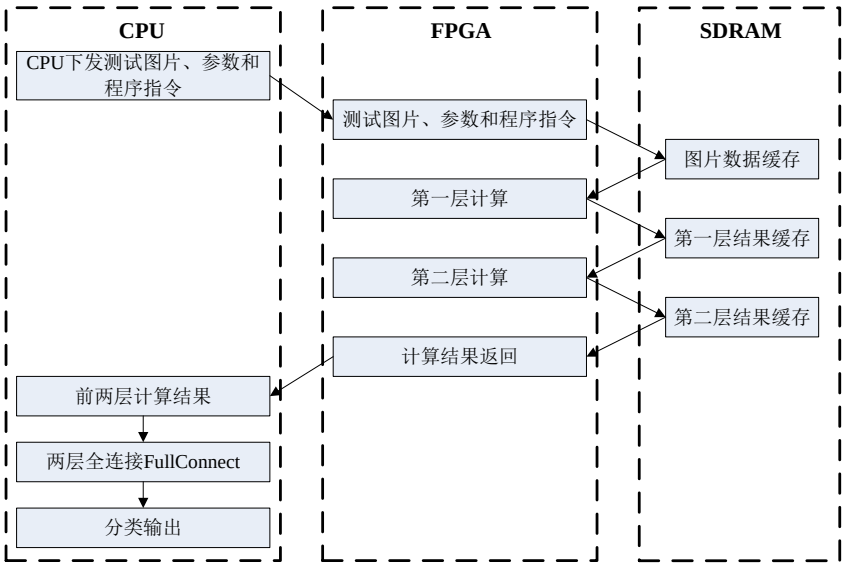


图 5-3 卷积神经网络计算流程图

单个计算层的结构框图如图 5-4 所示，其中所有的数据处理过程都添加流水线式结构。特征映射图和权值参数通过两个独立的通道加载到卷积模块；卷积运算结果传递到抽样模块进行最大采样处理；卷积模块和抽样模块计算的中间结果可进入激活函数模块进行非线性处理；卷即模块和抽样模块计算的中间结果可用乒乓缓存结构存储。

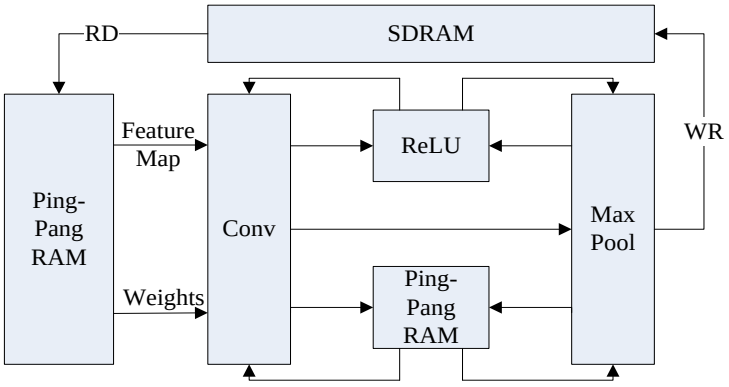


图 5-4 单个计算层的结构框图

为了简化设计的复杂度，本文采用开发板自带的 DE2_Control_Panel 控制面板作为 CPU 与 FPGA 之间通信的上位机用户界面。该控制面板可用于 DE2-70

开发板的功能测试和信息交互，具备 Memory 存储器、USB、SD 卡、开关按钮等多个模块的控制与调试功能。在启动卷积神经网络之前，首先将计算所需数据，通过 JATG 接口下载到开发板的存储器 SDRAM、SSRAM 或 FLASH 中存储，然后断开用户界面，下载运算模块程序，将卷积神经网络前向预测通路固化到 FPGA 中。用户界面与 DE2-70 开发板的连接通信情况如图 5-5 所示。

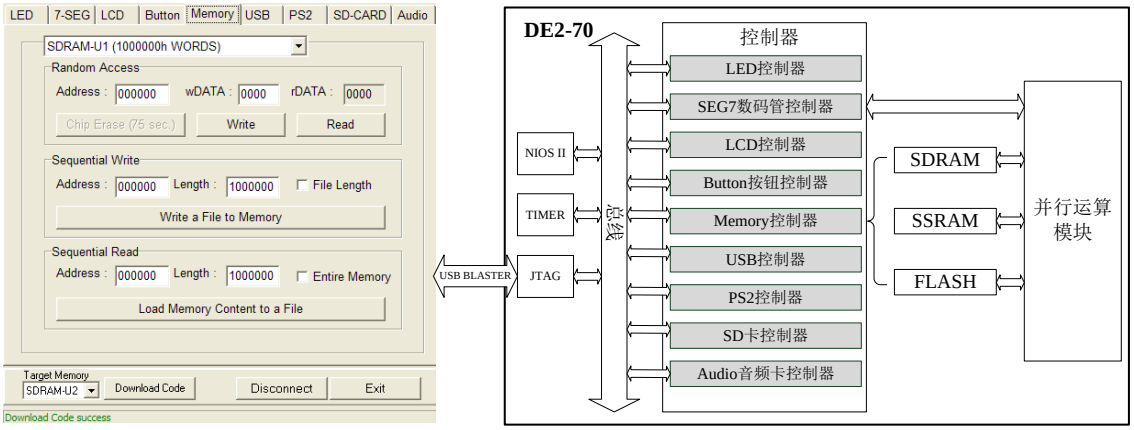


图 5-5 用户界面与 DE2-70 的通信框图

5.2 对比实验与识别性能分析

卷积神经网络计算过程中，数值精度的选择对 FPGA 硬件资源占用和实现效果有很大的影响。根据参考文献[58-60]，在保证实现性能和节约硬件资源的前提下，本文采用 16bit 的定点数表示输入数据和权值参数，其中权值参数是 1bit 符号位，15bit 小数位；输入数据是 1bit 符号位，3bit 整数位和 12bit 小数位。卷积运算输出结果是 31bit 定点数，1bit 符号位，3bit 整数位和 27bit 小数位。抽样运算经过有效位截取，输出结果是 16bit 定点数，1bit 符号位，3bit 整数位，12bit 小数位。由于卷积神经网络运算是带符号的位定点数运算，为了方便计算，运算过程中的定点数均以补码的形式保存。

在 FPGA 设计过程中，硬件的布局规划和时序分析是必不可少的。为了模拟各模块电路的真实运行状况，利用 FPGA 的片内存储单元，将逻辑分析仪嵌入到 FPGA 芯片内，实时观察模块运行时数据流处理情况。由于嵌入式逻辑分析仪的调试与使用需要占用较多的开发时间，为了节约开发周期，本文只选取 Z 型结构卷积运算模块，采样运算模块与 ReLU 激活函数实现模块级联整合模块，作为代表进行时序分析。根据开发板主频 50MHz 时钟，对模块进行时钟约束。逻辑分析仪采集的 FPGA 实际运行结果分别如图 5-6 和 5-7 所示。

通过嵌入式逻辑分析仪采集的数据结果可以验证，上述设计的功能模块在真实环境下运行时，产生的时序结果与 Modelsim 功能仿真的理论时序结果是一

致的。

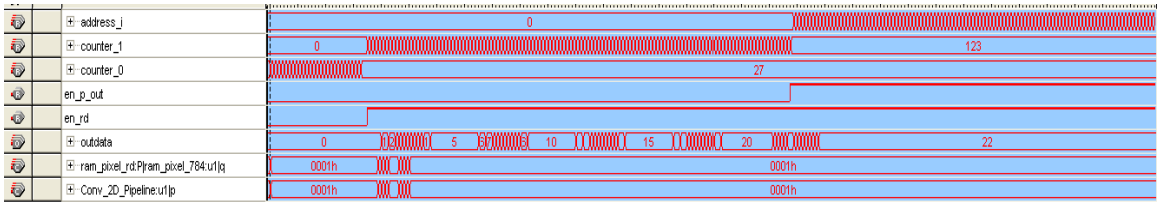


图 5-6 逻辑分析仪采集 Z 型卷积运算模块的运行结果

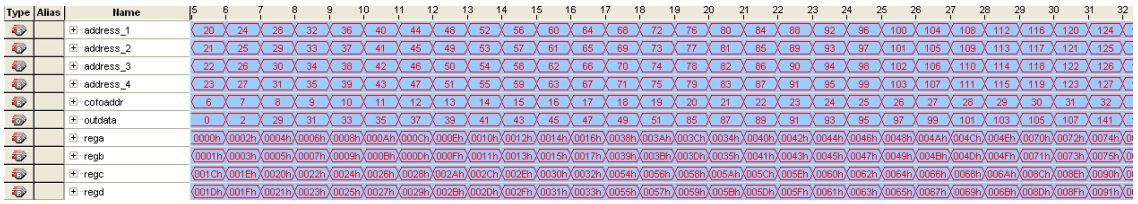


图 5-7 逻辑分析仪采集级联整合模块的运行结果

为了提升卷积神经网络的运算效率，根据不同网络层结构特点，本文采用多种形式的并行特性组合使用，但是由于受到 FPGA 自身资源限制，各种形式的并行度又不宜过大。如卷积神经网络第一个卷积层，输入特征映射图为单幅输入图像，输入为 1，输出为 6 个特征映射图数量，属于一种单输入多输出的网络层结构，因此开发特征映射图内部的并行特性更为适宜，选择 3D 单输入多卷积核并行结构，卷积核并行度为 6。卷积神经网络第二个卷积层，输入为 6 个输入特征映射图，输出为 12 个输出特征映射图，属于一种多输入多输出的网络层结构。为了减少多输出引起的计算中间结果占用缓存单元，以及读写缓存单元消耗缓冲时间的影响，同时又要避免过多的输入和过多的输出对存储带宽的高要求，需要在多输入与多输出之间权衡利弊。从数据重用方面考虑，随着多输入的提升，同样需要成比例增加存储单元来缓存重用数据。而多输出计算得到的中间结果，则可以通过在输出端添加流水线式累加器，得到输出特征映射图的神经元，从而有效利用占用的缓存单元。因此在第二个多输入多输出式的卷积层中，开发特征映射图之间的并行特性更为适宜，选择 3D 多输入多输出并行结果，输入并行度为 2，输出并行度为 3。卷积运算的并行特性是一种细粒度的并行特性，在开发过程中应当优先考虑。卷积神经网络的两个卷积层使用的卷积核大小均为5×5，因此卷积运算的并行度为 25。每个卷积运算模块使用 50 个乘法器（1 个 16bit 乘法需要 2 个 9bit 乘法器组合），每一层卷积层正好充分利用了 FPGA 所有集成的 300 个嵌入式乘法器，不存在闲置的乘法器单元。从理论上分析，单个卷积运算模块每时钟周期能达到的 25 次的峰值乘累加操作，每一卷积层则达到 150 次的峰值乘累加操作。

从 10000 张测试集中随机选取 100 张手写数字图片作为本文的实验样本，

每张实验图片大小为28×28，每个像素点占 16 位字节。首先卷积神经网络在CPU 上采用非监督式的逐层优化方法进行训练，经过充分训练后，网络的识别率达到了 97.73%。训练结束后，将 CPU 训练好的网络参数和选取好的实验样本图片下载到开发板存储器中预存，启动设计的卷积神经网络运行程序，验证FPGA 上实现卷积神经网络的手写数字识别效果，并且导出卷积神经网络每一个网络结构层的运算结果，与 CPU 上 MATLAB 软件运行结果做对比分析。图5-8 是测试样本中的其中一张原始输入图片，手写数字 0。MATLAB 的运算结果与 FPGA 运算结果的对比情况分别图 5-9、5-10 所示。

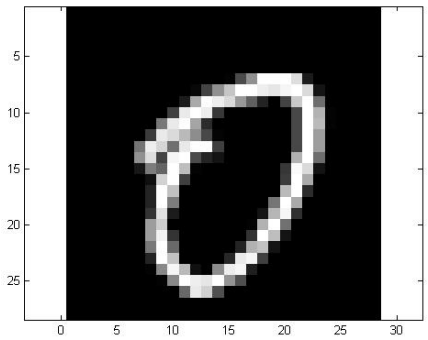
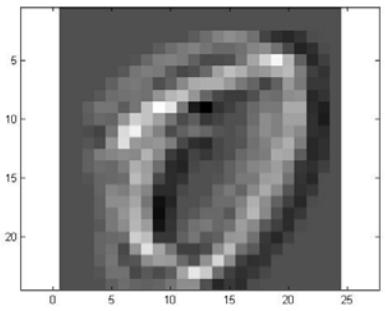
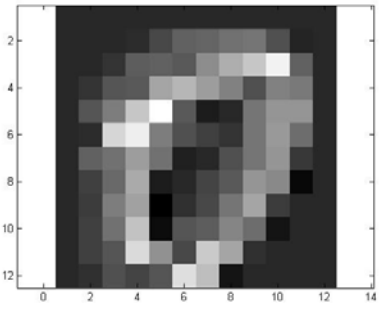


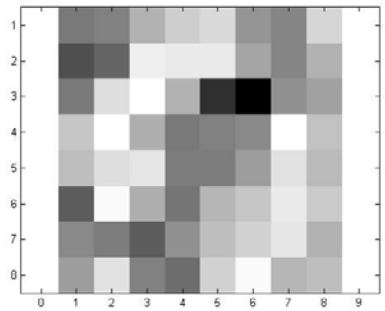
图 5-8 MNIST 手写数字原始输入图片



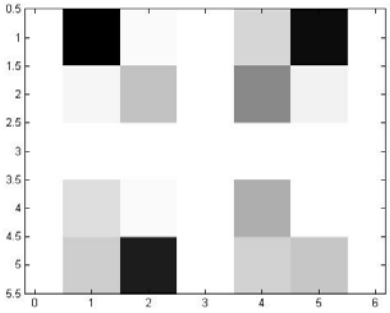
(a) 第一层



(b) 第二层

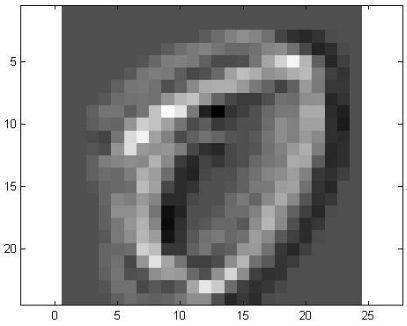


(c) 第三层

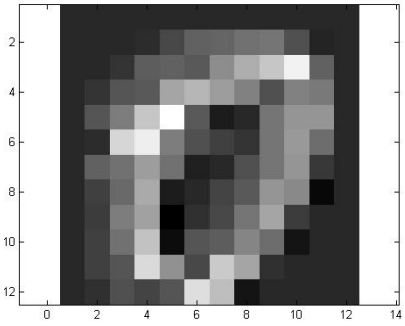


(d) 第四层

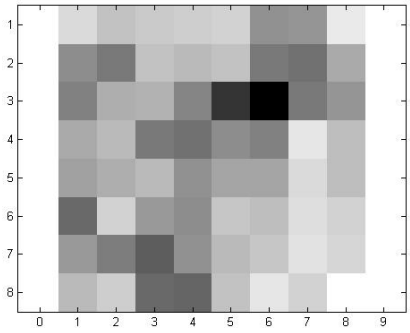
图 5-9 MATLAB 运算结果



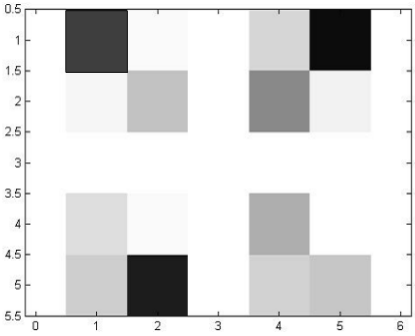
(a) 第一层



(b) 第二层



(c) 第三层



(d) 第四层

图 5-10 FPGA 运算结果

图 5-9、5-10 的(a)、(b)、(c)、(d)分别表示在 MATLAB 和 FPGA 中运行卷积神经网络时，第一层卷积层、第二层抽样层、第三层卷积层、第四层抽样层的运算结果，随着网络层处理深度的加深，输出图像的空间维度逐渐减小。对比分析 MATLAB 和 FPGA 各网络层的运行结果，可以看出在 FPGA 上实现卷积神经网络的手写数字识别效果与 MATLAB 上实现的识别效果基本相同。只有在第三层和第四层的运算结果存在极个别肉眼可见的差异，而这种差异只存在于个别神经元上，对手写数字的识别效果影响不大。

将卷积神经网络第四层的 FPGA 计算结果反馈到 CPU，在 MATLAB 中进行全连接的 softmax 分类器识别，经过 100 张手写数字实验样本的运行与统计，在 FPGA 上实现卷积神经网络识别手写数字的准确度达到 92%。

5.3 各模块资源使用与功耗分析

在 FPGA 设计卷积神经网络，最核心的任务就是开发单功能模块电路，通过重复调用单功能模块电路和开发网络并行特性，实现整个卷积神经网络的前向通路硬件固化。针对 FPGA 的主要硬件资源，综合设计的各模块电路后，统

计各模块电路和实现完整网络时整体结构占用的硬件资源。表 5-1 给出了本文设计的 Z 型卷积运算模块、树型卷积运算模块、抽样运算模块和整体系统模块的资源使用情况。

表 5-1 各模块 FPGA 资源使用情况

模块名称	逻辑单元	存储器	嵌入式乘法器
Z 型卷积	5422（8%）	48571（4%）	50（17%）
树型卷积	6054（9%）	55334（5%）	50（17%）
抽样	2220（3%）	32998（3%）	0（0%）
整体系统	49838（73%）	529703（46%）	300（100%）

表 5-1 中，逻辑单元包括组合逻辑单元（Total combinational functions）和专用逻辑寄存器（Dedicated logic registers），分别用于实现组合逻辑和时序逻辑功能；存储器包括片内缓存资源 RAM、ROM 等。逻辑单元、存储器、嵌入式乘法器的总量分别是 68416、1152000、300。本文设计的整体系统网络结构，其第三、四层网络结构所占用的资源是通过调用第一、二层网络结构使用过的资源，另外，在第一层和第二层网络结构之间同样存在资源共用的情况。从资源使用情况可以看出，在整体系统实现时，系统的存储资源和计算资源使用率都成一定程度的涨幅，而其中嵌入式乘法器使用率则更是达到了 100%。也就是说，在 FPGA 上实现卷积神经网络时，受逻辑存储资源和计算资源的制约限制程度非常明显。FPGA 内可用逻辑存储资源和计算资源的多少，能严重影响卷积神经网络并行特性的开发程度和硬件加速情况。

以设计卷积神经网络专用芯片为最终目标，在保证卷积神经网络运算速度的提前下，需要考虑尽可能地减少整个系统的功耗。表 5-2 是各模块单元和整体系统的功耗情况。整个 FPGA 设计的总功耗主要分三个部分组成。其中设计静态功耗和 I/O 功耗只与 FPGA 芯片和硬件本身制作工艺和设计布局直接相关，很难通过逻辑优化得到有较大的改善。设计动态功耗是 FPGA 正常启动后，设计部分所产生的能耗，与芯片所用电平、FPGA 内部逻辑和布线有关。

表 5-2 各模块 FPGA 功耗情况

模块名称	动态功耗(mW)	静态功耗(mW)	I/O 功耗(mW)	总功耗(mW)
Z 型卷积	121.04	155.00	62.15	338.19
树型卷积	136.72	154.96	70.03	361.71
抽样	71.68	154.98	57.42	284.08
整体系统	1345.92	155.12	77.13	1578.17

表 5-2 中可以看出，在不同的电路模块中，设计静态功耗与 I/O 功耗基本上无明显变化，而设计动态功耗则与设计者设计的电路结构息息相关，包括片内存储读写、逻辑电平跳变以及片外存储访问等。在 FPGA 上实现卷积神经网络手写数字图像的识别，需要反复不断地多次调用设计的各个电路模块。统计运行 100 张手写数字样本实验图片，平均每完成一张实验样本图片的识别，整体系统功耗为 2.93W。相比 CPU、GPU 显卡等平台实现，FPGA 存在非常明显的功耗优势。

5.4 本章小结

本章对本文设计的卷积神经网络 FPGA 实现的电路模块进行了手写数字识别实验。首先，介绍实验环境与卷积神经网络实现的网络结构。并根据已有的实验条件，搭建“FPGA+CPU”的异构系统。然后，依据每一网络层的特点，充分利用 FPGA 的资源，开发不同形式的网络并行架构，并在 FPGA 和 MATLAB 上进行对比实验，在 FPGA 上实现手写数字识别的准确率达到 92%。最后，分析各模块的资源利用和功耗情况，实验表明系统的整体功耗为 2.93W/张，相比 CPU、GPU 平台存在非常明显的功耗优势。

结　论

深度学习起源于人类对人工神经网络的研究，它作为人工智能领域最新开辟的一片天地，在机器视觉、图像处理、模式识别等多个工程应用领域中得到成功应用。目前，深度学习的实现方式主要以软件实现为主。一方面，在面临工业大数据的冲击下，基于软件实现深度学习的运行效率、功耗表现等方面，很多时候都无法满足工业需求；另一方面，随着集成电路设计、半导体技术和制造工艺的快速发展，以 FPGA 为代表的可编程逻辑器件在实现分布式并行结构有着得天独厚的优势，与计算密集型的深度学习算法结构特点不谋而合。本文结合 FPGA，开展深度学习的硬件化实现与优化技术研究。本文取得的主要成果如下：

（1）深度学习硬件实现总体方案设计。考虑到根据不同的研究场景和应用要求，深度学习算法的网络拓扑结构不尽相同。本文以卷积神经网络为深度学习算法的典型网络结构，以图像处理为其应用背景，对深度学习网络拓扑结构进行详细研究，并提出网络具体硬件实现的拓扑结构方案。

（2）算法硬件移植优化技术与架构设计。综合各方面考虑，选择 FPGA 作为本文设计实现的硬件平台。对深度学习软件算法硬件移植过程中可能涉及到的 FPGA 关键优化技术，如并行技术、乒乓技术、复用技术、分块技术进行详细的应用性研究，并根据卷积神经网络的结构特点，应用优化技术进行从粗粒度到细粒度的不同层之间、特征映射图之间、特征映射图内部、卷积运算的并行架构抽象与设计，提高卷积神经网络的执行效率。

（3）基于 FPGA 的卷积神经网络设计与实现。本文以 FPGA 为硬件开发平台，完成卷积神经网络的整体架构设计。根据卷积神经网络的分层式结构和各结构的功能特点，设计与实现网络的整体架构和各功能电路模块，包括卷积运算模块、抽样运算模块、激活函数模块，并用仿真软件验证各电路模块的功能正确性。其中卷积运算作为核心的运算操作，本文依据不同的数据流驱动方式，详细设计了 Z 型卷积运算结构和树型卷积运算结构，并对两种结构进行并行特性拓展设计和优化。为了缓解数据传输过程中的数据缓存和数据冲突的问题，本文采用乒乓结构设计缓存结构，以优化数据传输结构和数据缓存单元。

（4）在现有实验条件的基础上，本文设计了“FPGA+CPU”的异构体系，完成卷积神经网络的硬件固化。本文以 MNIST 手写数字识别为具体应用，进行硬件 FPGA 和软件 MATLAB 的对比实验，经多次实验统计，在 FPGA 实现卷积神经网络的手写数字识别，准确率达到 92%。每识别一张手写数字图片，FPGA

系统整体功耗控制在 2.93W 以内，相比 CPU、GPU 等平台实现，FPGA 有着非常明显的功耗优势，验证本文提出的方案具有良好的可行性和准确性。

本文通过具体的实验验证，在 FPGA 上实现以卷积神经网络为代表的深度学习算法，在工程应用中是可行的，并且相比软件和其他硬件平台的实现方案，拥有高效率、低功耗、低成本等多方面的优势。与此同时，由于现有实验条件、研究时间等因素的限制，本文中仍然存在的许多问题，需要在后续工作中更深入的研究和探讨：

（1）本文 FPGA 硬件实现手写数字识别的准确率没有达到 MATLAB 软件实现的准确率，需要进一步分析影响其识别率的原因，如并行结构的影响、并行度开发的影响、数值精度的影响等多个方面。

（2）本文采用的是 Cyclone II 系列的低成本通用 FPGA 芯片，在制作工艺、硬件资源上存在不足。如果采用高端高集成 FPGA 芯片，配合 OpenCL 应用程序和面向 FPGA 的 SDK 开发包，进一步提升深度学习算法的并行开发度，提高算法的执行效率和综合性能。

（3）单块 FPGA 芯片资源有限，无法大幅度提升运算性能。可以借鉴硬件集群的设计思路，结合 PCI、USB、网口等标准通信接口和固态盘等存储资源，进一步设计成即插即用的深度学习专用 FPGA 板卡，配合 CPU、GPU，实现多种形式的异构体系，进一步探索深度学习算法在不同异构体系和集群方式下的执行效率和综合性能。

参考文献

[1] 王梦雪. 数据挖掘综述[J]. 软件导刊, 2013, 12(10): 135-137.

[2] 梁循. 数据挖掘: 建模、算法、应用和系统[J]. 计算机技术与发展, 2006, 16(01): 1-4.

[3] LeCun Y, Y. Bengio, and G. Hinton. Deep learning[J]. Nature, 2015, 521: 436-444.

[4] Fei-Yue Wang, Jun Jason Zhang, Xinhu Zheng. Where does AlphaGo go: from church-turing thesis to AlphaGo thesis and beyond[J]. IEEE/CAA Journal of Automatica Sinica. 2016, 3(2): 113-120

[5] Hinton G, Osindero S, Teh Y W. A fast learning algorithm for deep belief nets[J]. Neural computation, 2006, 18(7): 1527-1554.

[6] WayneWolf, 沃尔夫, Wolf. FPGA-Based System Design[M]. 机械工业出版社, 2005.

[7] 刘建伟, 刘媛, 罗雄麟. 深度学习研究进展[J]. 计算机应用研究, 2014, 31(7): 1921-1930.

[8] 毛健, 赵红东, 姚婧婧. 人工神经网络的发展及应用[J]. 电子设计工程, 2011, 19(24): 62-65.

[9] Hopfield J. “Artificial Neural Networks”, IEEE Trans. CAS, 1988, 35(9): 3-10.

[10] 戴文战. 基于三层 BP 网络的多指标综合评估方法及应用[J]. 系统工程理论与实践, 1999, 19(05): 29-34.

[11] Yoshua Bengio, Yann LeCun, Craig Nohl. LeRec: A NN/HMM Hybrid for On-Line Handwriting Recognition[J]. MIT Press Journals, 1995, 7(6): 1289-1303.

[12] Hinton G E, Salakhutdinov R R. Reducing the dimensionality of data with neural networks[J]. Science, 2006, 313(5786): 504-507.

[13] Hadsell R, Sermanet P, Ben J, et al. Learning long-range vision for autonomous off-road driving[J]. Journal of Field Robotics, 2009, 26(2): 120-144.

[14] 余凯, 贾磊, 陈雨强, 徐伟. 深度学习的昨天、今天和明天[C]. 中国计算机研究与发展, 2013, (09): 1799-1804.

[15] 陈先昌. 基于卷积神经网络的深度学习算法与应用研究[D]. 浙江工商大学, 2014.

[16] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep

convolutional neural networks[C]. International Conference on Neural Information Processing Systems. Curran Associates Inc. 2012, 25(2): 1097-1105.

[17] 孙志军, 薛磊, 许阳明, 等. 深度学习研究综述[J]. 计算机应用研究, 2012, 29(8): 2806-2810.

[18] 赵冬斌, 邵坤, 朱圆恒等. 深度强化学习综述: 兼论计算机围棋的发展[J]. 控制理论与应用, 2016, 33(06): 701-717.

[19] 石菲. 当 AlphaGo 成为大师[J]. 中国信息化, 2017, (1): 7-7.

[20] 陈达. 基于深度学习的推荐系统研究[D]. 北京邮电大学, 2014.

[21] 邹永强. Mariana 腾讯深度学习平台进展与应用[C]. CCF 大数据学术会议, 2014.

[22] LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.

[23] 白洪涛. 基于 GPU 的高性能并行算法研究[D]. 吉林大学, 2010.

[24] Smirnov E A, Timoshenko D M, Andrianov S N. Comparison of Regularization Methods for ImageNet Classification with Deep Convolutional Neural Networks[J]. Aasri Procedia, 2014, 6(1): 89-94.

[25] Teng Li, Yong Dou, Jingfei jiang, et al. Optimized Deep Belief Networks on CUDA GPUs[C]. International Joint Conference on Neural Networks, 2015: 1-8.

[26] 葛蕾, 霍爱清. Widrow-Hoff 神经网络学习规则的应用研究[J]. 电子设计工程, 2009, 17(6): 15-16.

[27] 余子健. 基于 FPGA 的卷积神经网络加速器[D]. 浙江大学, 2016.

[28] Boser B. E, Sackinger E, Bromley J, et al. An analog neural network processor with programmable topology[J]. IEEE Journal of Solid-State Circuits, 1991, 26(12): 2017-2025.

[29] Sackinger E, Boser B. E, Bromley J, et al. Application of the ANNA neural network chip to high-speed character recognition[J]. IEEE Transactions on Neural Networks, 1992, 3(3): 498:505.

[30] Clement Farabet, Cyril Poulet, Jefferson Y. Han, Yann LeCun. CNP: An FPGA-based processor for Convolutional Networks[C]. International Conference on Field Programmable Logic and Applications, 2009: 32-37.

[31] Zhou Y, Jiang J. An FPGA-based accelerator implementation for deep convolutional neural networks[C]. International Conference on Computer Science and Network Technology. IEEE, 2016, (1): 829-832.

[32] Peemen M, Setio A A A, Mesman B, et al. Memory-centric accelerator design

for Convolutional Neural Networks[C]. IEEE, International Conference on Computer Design. IEEE, 2013: 13-19.

[33] Alhamali A, Salha N, Morcel R, et al. FPGA-Accelerated Hadoop Cluster for Deep Learning Computations[C]. IEEE International Conference on Data Mining Workshop. IEEE, 2015: 565-574.

[34] 陆志坚. 基于 FPGA 的卷积神经网络并行结构研究[D]. 哈尔滨工程大学, 2013.

[35] 王羽. 基于 FPGA 的卷积神经网络应用研究[D]. 华南理工大学, 2016.

[36] H. B. Burke, D. B. Rosen, P. H. Goodman. Comparing artificial neural networks to other statistical methods for medical outcome prediction[C]. Comparing artificial neural networks to other statistical methods for medical outcome prediction, 1994, 4(27): 2213-2216.

[37] Bao J, Zhou B. Optimization of neural network with fixed-point weights and touch-screen calibration[C]. Industrial Electronics and Applications, 2009. Iciea 2009. IEEE Conference on. 2009:3704-3708.

[38] 汪镭, 周国兴, 吴启迪. 人工神经网络理论在控制领域中的应用综述[J]. 同济大学学报(自然科学版), 2001, 29(3): 357-361.

[39] Haykin S, Network N. A comprehensive foundation[J]. Neural Networks, 2004.

[40] 覃光华. 人工神经网络技术及其应用[D]. 四川大学, 2003.

[41] 蒋宗礼. 人工神经网络[M]. 北京: 高等教育出版社, 2001.

[42] 朱大奇, 史慧. 人工神经网络原理及应用[M]. 科学出版社, 2006.

[43] Hinton G E. A Practical Guide to Training Restricted Boltzmann Machines[J]. Momentum, 2012, 9(1): 599-619.

[44] Cun Y L, Boser B, Denker J S, et al. Handwritten digit recognition with a back-propagation network[C] Advances in Neural Information Processing Systems. Morgan Kaufmann Publishers Inc. 1990: 396-404.

[45] Yawei Li, Yuliang Yang, Yueyun Chen, Mengyu Zhu. A pre-training strategy for convolutional neural network applied to Chinese digital gesture recognition[C]. IEEE International Conference on Communication Software, 2016: 620-624.

[46] 陈学松, 杨宜民. 强化学习研究综述[J]. 计算机应用研究, 2010, 27(8): 2834-2838.

[47] Jiangqun Ni, Jian Ye, Yang YI. Deep Learning Hierarchical Representations for Image Steganalysis[J]. IEEE Transactions on Information Forensics and Security, 2017. PP (99): 1-1.

[48] 马冬梅. 基于深度学习的图像检索研究[D]. 内蒙古大学, 2014.

[49] 朱少杰. 基于深度学习的文本情感分类研究[D]. 哈尔滨工业大学, 2014.

[50] Zhu Z, Luo P, Wang X, et al. Deep Learning Identity-Preserving Face Space[C]. IEEE International Conference on Computer Vision. IEEE, 2013: 113-120.

[51] 李海峰, 李纯果. 深度学习结构和算法比较分析[J]. 河北大学学报:自然科学版, 2012, 32(5): 538-544.

[52] 凡保磊. 卷积神经网络的并行化研究[D]. 郑州大学, 2013.

[53] 余奇. 基于 FPGA 的深度学习加速器设计与实现[D]. 中国科学技术大学, 2016.

[54] 夏宇闻. 复杂数字电路与系统的设计技术[M]. 北京: 北京航空航天大学出版社，1998.

[55] 张萧, 黄晞, 仲伟汉, 张亮. Sigmoid 函数及其导函数的 FPGA 实现[J]. 福建师范大学学报(自然科学版), 2011, 27(2): 62-65.

[56] Motamedi M, Gysel P, Akella V, et al. Design space exploration of FPGA-based Deep Convolutional Neural Networks[C]. Design Automation Conference. IEEE, 2016: 575-580.

[57] Wang C, Gong L, Yu Q, et al. DLAU: A Scalable Deep Learning Accelerator Unit on FPGA[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2016, PP(99): 1-1.

[58] Draghici S. On the capabilities of neural networks using limited precision weights[J]. Neural networks, 2002, 15(3): 395-414.

[59] Moussa M, Areibi S, Nichols K. On the arithmetic precision for implementing back-propagation networks on FPGA: a case study[M]. FPGA Implementations of Neural Networks. Springer US. 2006: 37-61.

[60] Holi J. L., Hwang J. N. Finite precision error analysis of neural network hardware implementations [J]. IEEE Transactions on Computers, 1993, 42(3): 281-290.

攻读硕士学位期间发表的论文及其它成果

(一)发表的学术论文

[1] Min Zhu, Jian-Jun Lin, Li Wang, Chun-Ling Yang. A multidimensional features fault diagnosis method for analog circuits[C]. Industrial Electronics Conference (IECON), IEEE, 2016: 382-387. （EI 源检索）

[2] Min Zhu, Jian-Jun Lin, Li Wang, Dong-Yang Zhao, Dong-Hai Fu. A D-M chaotic model ATPG in mixed circuits BIST[C]. International Conference on Artificial Intelligence: Techniques and Applications (AITA), 2016: 64:68.（SCI 源检索）

(二)竞赛获奖

[1] 喻佳健，岳建民，林楗军. 高精度电阻源. 中国研究生电子设计竞赛全国总决赛团体二等奖. 2016.08.

哈尔滨工业大学学位论文原创性声明和使用权限

学位论文原创性声明

本人郑重声明：此处所提交的学位论文《深度学习的硬件实现与优化技术研究》，是本人在导师指导下，在哈尔滨工业大学攻读学位期间独立进行研究工作所取得的成果，且学位论文中除已标注引用文献的部分外不包含他人完成或已发表的研究成果。对本学位论文的研究工作做出重要贡献的个人和集体，均已在文中以明确方式注明。

作者签名：林健早　　日期：2017 年 6 月 26 日

学位论文使用权限

学位论文是研究生在哈尔滨工业大学攻读学位期间完成的成果，知识产权归属哈尔滨工业大学。学位论文的使用权限如下：

（1）学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文，并向国家图书馆报送学位论文；（2）学校可以将学位论文部分或全部内容编入有关数据库进行检索和提供相应阅览服务；（3）研究生毕业后发表与此学位论文研究成果相关的学术论文和其他成果时，应征得导师同意，且第一署名单位为哈尔滨工业大学。

保密论文在保密期内遵守有关保密规定，解密后适用于此使用权限规定。

本人知悉学位论文的使用权限，并将遵守有关规定。

作者签名：林健早　　日期：2017 年 6 月 26 日

导师签名：朱金2　　日期：2017 年 6 月 26 日

致　谢

时光飞逝，转眼间即将结束两年的哈工大研究生生活。在这两年的学习、生活和科研中，有成功的喜悦，有失败的烦恼，有相知相遇的欢声笑语，有相惜相离的悲欢离合，熟悉这里的每一寸土地，走遍这里的每一间教室，感慨万千，爱在哈工大。

首先，我要感谢我的硕士生导师，朱敏老师。朱老师在生活上，就像我的大哥哥一样，给予了我无微不至的关怀和帮助。在科研中，朱老师以敏锐而开阔的思维、认真而严谨的治学态度、求实而务实的工作作风、平易而近人的生活态度潜移默化地向我诠释哈工大“规格严格、功夫到家”的百年校训，令我受益良多，并将终身难忘。在此，我对朱敏老师表示真诚的感谢和深深的敬意，并致以最美好的祝愿。

其次，我还要特别感谢教研室的杨春玲教授、刘思久教授、张岩老师在研究生生涯中对我的指导与帮助，几位知识渊博的老师对科研的热忱、对学生的关爱让人感动，尤其是杨春玲老师，是我们教研室公认的杨妈妈，对学生视如己出，给予无微不至的关爱和照顾，让远在异乡的学生又有了家的温暖。

然后，特别感谢实验室学习过程中，给予我帮助的四位博士师兄：张国亮师兄、王荔师兄、陈宇师兄、张振东师兄。还要感谢同实验室同学：岳建民、喻佳健、侯耀东、常涵宇、赵帅在学习生活过程中的相伴与扶持。另外，还要感谢实验室的师弟师妹：张硕、张淼、孙弘毅、张鹏、董郑宇、况麒麒、刘明、段亦涵，感谢你们营造实验室良好、和谐、融洽、积极向上的学习与工作氛围，为我的学习和论文工作的顺利完成提供了保证。还要感谢同寝室的张文生同学和我的上届师兄师姐：王冰、弓圣阳、邱晓明、孙思贤、颜馨郁，感谢你们的关心与帮助带给我温暖，你们的坚持与努力带给我前进的动力，是你们带给了我研究生最亲密无间的感情，使我变得更加优秀。

最后感谢我的家人对我的关怀和帮助，他们一直是我最强大的后盾与港湾。他们始终如一的鼓励和关爱，让我深刻体会到亲情的温暖，让我拥有成长和进步的勇气和动力。

感谢所有关心和帮助过我的朋友，感谢母校给予我的一切。