

学校代码： 10286

分类号： TN47

密 级： 公开

UDC： 621.3

学 号：

**电子信息硕士学位论文论文**

基于BNN的ECG信号检测神经网络加速器的设计

（学位论文形式：应用研究）

研究生姓名**：**

导师姓名**：**

|  |  |
| --- | --- |
| 申请学位类别 *工程硕士* | 学位授予单位  **东 南 大 学** |
| 工程领域名称 *集成电路工程* | 论文答辩日期 **2023年 月 日** |
| 研究方向*系统芯片与嵌入式系统* | 学位授予日期 **2023年 月 日** |
| 答辩委员会主席 | 评 阅 人 |
| 2023年 5月 日 | |
|  | |

|  |
| --- |
| Dd1  **电子信息硕士学位论文**  **基于BNN的ECG信号检测神经网络加速器的设计**  **专 业 名 称**： **电子信息**  **研究生姓名：**  **导 师 姓 名**： |

Design of ECG signal detection neural network accelerator based on BNN

A Thesis Submitted to

Southeast University

For the Professional Degree of Master of Engineering

BY

Supervised by

and

School of Microelectronics

Southeast University

May 2023

**东南大学学位论文独创性声明**

本人声明所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得东南大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

研究生签名： 日期：

**东南大学学位论文使用授权声明**

东南大学、中国科学技术信息研究所、国家图书馆、《中国学术期刊（光盘版）》电子杂志社有限公司、万方数据电子出版社、北京万方数据股份有限公司有权保留本人所送交学位论文的复印件和电子文档，可以采用影印、缩印或其他复制手段保存论文。本人电子文档的内容和纸质论文的内容相一致。除在保密期内的保密论文外，允许论文被查阅和借阅，可以公布（包括以电子信息形式刊登）论文的全部内容或中、英文摘要等部分内容。论文的公布（包括以电子信息形式刊登）授权东南大学研究生院办理。

研究生签名： 导师签名： 日期：

摘要

心律失常是心血管疾病中的一组常见并发症，具有发作时间短、发病后果严重的特点，因此在嵌入式设备上利用心电图（Electrocadiogram，ECG）对心律失常进行实时检测具有重要的意义。卷积神经网络（Convolution Neural Network, CNN）在图像识别领域有着出色的性能，基于心电图和图像的相似性，使用CNN实现ECG信号的识别检测已成为研究的热点。然而，CNN作为一种计算密集、访存密集的模型，部署到计算资源和内存空间有限的嵌入式设备中时面临着较大挑战。本文针对这一问题，设计了一种基于二值量化的高能效ECG检测CNN硬件加速器。

本文的主要工作如下：1.设计了一种对心电图信号实现分类检测的CNN模型，并使用二值量化对模型进行压缩，降低了模型的计算量和内存占用。2.利用二值量化的特性，设计了一种高效的CNN计算单元，通过消除乘法操作和减少数据位宽，降低了硬件的功耗和资源占用。3.通过CNN模型和硬件加速器的协同设计，实现了高能效的ECG检测CNN硬件加速器，并在FPGA上进行了功能验证和性能评估。

本文基于MIT-BIH Arrhythmia数据集训练和测试了ECG检测CNN模型，其识别准确率达到96.45%。经过二值量化后，模型识别准确率仍达到92.83%，模型内存占用减少至28.1KB，压缩率达到29.5倍。最终在Xilinx Zynq-7035开发板上实现了基于二值量化的ECG检测CNN硬件加速器，测试结果表明其吞吐率达到51.6 GOPS，能效达到146.2 GOPS/W。与其他心律检测的硬件方案相比具有更高的能效和吞吐率。

**关键词：**心律信号检测，二值量化，卷积神经网络，硬件加速器

Abstract

Arrhythmia is a common complication of cardiovascular diseases, which has the characteristics of short onset time and serious consequences. Therefore, it is of great significance to use electrocardiogram (ECG) to realize real-time detection of arrhythmia on embedded devices. Convolutional neural network (CNN) has excellent performance in image recognition field. Based on the similarity between ECG and image, using CNN to realize ECG signal recognition and detection has become a hot research topic. However, CNN, as a computation-intensive and memory-intensive model, faces great challenges when deployed on embedded devices with limited computing resources and memory space. In view of this problem, this paper designs a high-energy-efficiency ECG detection CNN hardware accelerator based on binary quantization.

The main work of this paper is as follows: 1. A CNN model for ECG signal classification and detection is designed, and binary quantization is used to compress the model, reducing the computation and memory consumption of the model. 2. Based on the characteristics of binary quantization, an efficient CNN computation unit is designed, which eliminates multiplication operations and reduces data bit width, thus reducing the power consumption and resource consumption of hardware. 3. Through the co-design of CNN model and hardware accelerator, a high-energy-efficiency ECG detection CNN hardware accelerator is realized and verified and evaluated on FPGA.

This paper trains and tests the ECG detection CNN model based on MIT-BIH Arrhythmia dataset, which achieves a recognition accuracy of 96.45%. After binary quantization, the model recognition accuracy still reaches 92.83%, the model memory consumption is reduced to 28.1KB, and the compression ratio reaches 29.5 times. Finally, the ECG detection CNN hardware accelerator based on binary quantization is implemented on Xilinx Zynq-7035 development board. The test results show that its throughput reaches 51.6 GOPS and its energy efficiency reaches 146.2 GOPS/W. Compared with other hardware schemes for arrhythmia detection, it has higher energy efficiency and throughput.

**Keywords:** arrhythmia signal detection; binary quantization; convolutional neural network; hardware accelerator

目 录

[摘要 I](#_Toc134556513)

[Abstract III](#_Toc134556514)

[目 录 V](#_Toc134556515)

[第一章 绪论 1](#_Toc134556516)

[1.1 课题研究的背景和意义 1](#_Toc134556517)

[1.2 国内外研究现状 2](#_Toc134556518)

[1.2.1 传统ECG信号检测方法 2](#_Toc134556519)

[1.2.2 基于机器学习的ECG信号检测 3](#_Toc134556520)

[1.2.3 ECG信号检测网络压缩及硬件实现 5](#_Toc134556521)

[1.3 研究内容及设计指标 7](#_Toc134556522)

[1.3.1 研究内容 7](#_Toc134556523)

[1.3.2 设计指标 8](#_Toc134556524)

[1.4 论文组织结构 10](#_Toc134556525)

[第二章 卷积神经网络原理和网络压缩方法概述 13](#_Toc134556526)

[2.1 传统ECG信号分类方法 13](#_Toc134556527)

[2.2 卷积神经网络基本原理 14](#_Toc134556528)

[2.3 卷积神经网络量化方法 17](#_Toc134556529)

[2.3.1 卷积神经网络量化原理 18](#_Toc134556530)

[2.3.2 卷积神经网络量化策略 20](#_Toc134556531)

[2.4 卷积神经网络二值量化及其专用硬件设计 21](#_Toc134556532)

[2.4.1 卷积神经网络二值量化简述 21](#_Toc134556533)

[2.4.2 卷积神经网络二值量化专用硬件简述 24](#_Toc134556534)

[2.5 本章小结 26](#_Toc134556535)

[第三章ECG检测卷积神经网络与网络二值化方法设计 29](#_Toc134556536)

[3.1 ECG信号检测网络训练及量化流程 29](#_Toc134556537)

[3.2数据集预处理 29](#_Toc134556538)

[3.3ECG信号检测神经网络设计 32](#_Toc134556539)

[3.3.1 卷积神经网络结构探索 33](#_Toc134556540)

[3.3.2 网络性能分析对比 36](#_Toc134556541)

[3.4 ECG信号检测神经网络二值量化设计 38](#_Toc134556542)

[3.4.1 二值量化训练流程设计 38](#_Toc134556543)

[3.4.2 二值量化训练过程介绍 39](#_Toc134556544)

[3.4.3 量化后网络性能分析对比 44](#_Toc134556545)

[3.5 本章小结 45](#_Toc134556546)

[第四章 基于BNN的ECG信号检测神经网络加速器的设计 47](#_Toc134556547)

[4.1 加速器整体结构设计 47](#_Toc134556548)

[4.2 加速器数据流分析及设计空间探索 49](#_Toc134556549)

[4.2.1 加速器数据流分析 49](#_Toc134556550)

[4.2.1 设计空间探索 52](#_Toc134556551)

[4.3 加速器二值化计算单元设计 55](#_Toc134556552)

[4.4 加速器各模块设计及验证 58](#_Toc134556553)

[4.4.1 主控模块设计 58](#_Toc134556554)

[4.4.2 缓存模块设计 66](#_Toc134556555)

[4.4.3 存储模块设计 72](#_Toc134556556)

[4.4.4 计算模块设计 74](#_Toc134556557)

[4.5加速器计算过程 78](#_Toc134556558)

[4.6本章小结 82](#_Toc134556559)

[第五章 ECG信号检测神经网络加速器的FPGA验证 83](#_Toc134556560)

[5.1实验测试平台及验证方案 83](#_Toc134556561)

[5.1.1实验工具介绍 83](#_Toc134556562)

[5.1.2系统验证方案 84](#_Toc134556563)

[5.2 FPGA验证实现 86](#_Toc134556564)

[5.3 系统性能测试 88](#_Toc134556565)

[5.4 测试结果分析 90](#_Toc134556566)

[5.5 本章小节 91](#_Toc134556567)

[第六章 总结与展望 93](#_Toc134556568)

[6.1 论文总结 93](#_Toc134556569)

[6.2 未来展望 93](#_Toc134556570)

[致谢 95](#_Toc134556571)

[参考文献 97](#_Toc134556572)

[攻读研究生硕士期间的成果 101](#_Toc134556573)

第一章 绪论

1.1 课题研究的背景和意义

随着社会经济的发展，城镇化以及人口老龄化的加速，居民不健康的生活方式日益普遍。心血管疾病（Cardiovascular disease,CVD）的发病率持续升高，是我国城乡居民死亡的首要原因[1]。心律失常作为心血管疾病中重要的一组疾病，可能并发其他疾病，而心电图(Electrocardiogram, ECG)是检测心律失常的重要依据。心律失常的早期症状不明显，在常规的心律检测中难以被发现，而一旦出现严重的心律失常可能导致死亡[2]。因此，对心律进行长时检测，并及时对心律中的异常发出预警，可以有效降低因心律失常导致的疾病风险。所以在嵌入式设备上实现实时的心律检测是十分有意义的。

相较于专业医生对心电图的分析诊断，利用算法进行心律失常检测更加便捷，能够提供实时的反馈和建议。近年来随着机器学习的发展，越来越多学者开始使用机器学习的方法来实现心律的检测分类。早期的研究中，人们通常手工设计一个心律的特征提取流程，然后利用传统的机器学习方法对此特征进行分类[3][4]。然而，随着深度神经网络的发展，深度学习的方法能够端到端的完成特征提取和检测分类，而不会因为手工设计的特征提取过程优劣而影响精度。目前最先进的基于卷积神经网络的心律检测识别方法，其分类精度已经超过了人类专家的水平[5]。

在利用深度学习实现心律检测的研究中，主流的方式是使用卷积神经网络（Convolutional Neural Networks，CNN）和循环神经网络（Recurrent Neural Network，RNN）[6]。卷积神经网络在图像识别任务中发挥出色，而心电图也和图像一样有着局部相关性和平移不变性的特征，因此卷积神经网络在心律检测中有着良好的性能。相较于需要处理时序依赖性问题的循环神经网络而言，卷积神经网络因其并行性，更便于部署在硬件上。

卷积神经网络虽然具有优异的性能，但是其作为一种计算密集型和访存密集型的算法，其在低功耗的嵌入式设备上的部署面临着计算资源不足、能耗过高等挑战。卷积神经网络中包含大量的简单重复的乘累加计算，这些运算在CPU上执行效率低下。而GPU虽然能够发挥卷积计算的并行性，但是难以适应功耗受限的嵌入式平台。因此，设计高效的专用硬件加速器来实现嵌入式平台上CNN算法的实时推理是一种更优的方案。

为了解决卷积神经网络部署在嵌入式设备上面临的资源受限和带宽受限的问题，网络压缩是一种常用的优化策略。主要的网络压缩方法为权重剪枝和量化。剪枝分为针对卷积核进行全体删除的结构化剪枝和针对单个权重进行删除的非结构化剪枝，其通过去除网络中不重要或冗余的权重来使得网络稀疏化，进而降低参数量。而量化则是将网络的权重、激活值等用低位宽的定点数来近似表示，相较于剪枝而言更为直接有效。传统的量化方法将权重量化到8位而不明显损失精度，而最新的研究表明，将网络量化到三值甚至二值仍然能够保持相当的精度。

综上所述，由于心律失常的危害性和症状难以监测的隐蔽性，在嵌入式设备上实现实时的心律监测和预警是十分有意义的。卷积神经网络算法在心律检测任务上有着良好的性能，但由于其计算量大、访存量大的特点，在资源受限的嵌入式设备上部署时会面临资源不足、功耗过大等问题。为解决上述问题，本文通过算法和硬件的协同设计，研究心律检测卷积神经网络，采用二值量化的方法对网络进行轻量化设计，并设计出其专用的高效硬件架构，最终实现高能效的心律信号检测识别神经网络专用加速器，并在FPGA平台上进行验证实现。

1.2 国内外研究现状

长期以来，利用心电图数据进行心律检测一直是研究的热点。随着机器学习的发展，特别是深度学习的发展，国内外学者对心律检测的神经网络算法及其硬件实现展开了广泛的研究。

1.2.1 传统ECG信号检测方法

ECG信号的检测分类通常分为信号去噪、特征提取和特征分类三个阶段。传统的ECG信号检测方法对ECG信号进行手工提取特征，然后根据提取的特征进行线性的分类。

2010年，Miad Faezipour等人使用自适应分析(adaptive profiling)的方法[7]对ECG信号进行分类。其使用了一种基于小波变换的方法从ECG信号中提取特征，然后将ECG信号视为数据包流，使用基于重复检测(repetition-detection)的技术推导出每个患者的正常心律区域，当待检测信号超出正常心律区域即可识别为异常心律。最终此方法在MITDB(MIT-BIH Arrhythmia Database)上取得了97.42%的准确率。

2014年，Swati Banerjee等人[8]使用基于阈值的方法对ECG信号进行分类。其使用交叉小波变换对ECG信号进行分析，计算待测心律和正常心律之间的小波交叉谱和小波相关性。然后从小波交叉谱和小波相关性中提取参数，根据参数的数值范围将其分为五种不同类别。最终在MITDB中对五种常见心律类别实现了97.6%的准确率。

传统的ECG信号检测方法的特征参数提取和分类标准均是手工设计的，依赖于设计者的经验和医学知识。其在较小的数据集和较简单的分类任务上有着优异的精度表现，且拥有着较低的计算量需求。但是传统的ECG信号检测方法也存在着泛化性不佳、对噪声敏感等问题，且在大型的数据集和复杂的分类任务上表现不佳。

1.2.2 基于机器学习的ECG信号检测

随着机器学习特别是深度学习的发展，越来越多研究开始采用机器学习相关方法来实现ECG信号的检测。不同于传统的方法，其使用机器学习模型自动的实现分类任务，虽然相较于传统方法不具有可解释性，但是在精度上取得了更好的表现。

2015年，Usha Desai等人[9]使用支持向量机(Support Vector Machines,SVM)的方法实现了ECG信号的五分类。其通过离散小波变换对ECG信号进行特征提取，然后使用独立成分分析对提取的特征进行降维，最后使用SVM进行分类，最终在MITDB上取得了98.49%的平均准确率。

2016年，Taiyong Li等人[10]使用随机森林(Random Forest, RF)的方法实现了ECG信号的五分类。其从ECG信号中提取小波包熵和R峰间隔作为特征，然后使用RF进行五分类，最终在MITDB上取得了94.61%的平均准确率。相较于SVM的方法，RF有着更少的推理计算量。

传统机器学习方法仍然依赖于手工提取的特征，这可能会导致算法过拟合。深度学习的方法可以让算法充分利用数据的信息量，随着计算机算力的发展，越来越多研究尝试使用深度学习的方法实现ECG信号的检测分类。由于ECG信号是一个时间序列，同时也可以视为一个二维图像，因此通常采用循环神经网络和卷积神经网络的方法实现ECG的检测分类。

在循环神经网络的方法中，长短期记忆神经网络（Long Short Term Memory，LSTM）缓解了传统循环神经网络中的梯度消失或爆炸问题，因此很多工作使用长短期记忆神经网络来对心律信号进行检测分类[11][12][13]。

2018年，Yıldırım等人[11]将心律信号通过小波变换分解为不同尺度的频率子带，然后将这些子带作为长短期记忆网络的输入信号，最终在MITDB数级集上实现了五种常见心律信号的检测分类，准确度达到了99.39%。

2020年，Borui Hou等人[13]利用基于长短期记忆网络的自动编码器网络（Auto-Encoder，AE），结合支持向量机实现了常见心律和心拍的分类。网络由一个编码器和一个解码器组成，心律信号经过长短期记忆网络编码后，将提取的特征交由支持向量机进行分类，同时使用基于长短期记忆网络的解码器将提取特征还原成原始信号。最终网络在美国医疗仪器进步协会（AAMI）心拍分类标准下，在MITDB上取得了99.45%的准确率。

由于心电图信号和图像的相似性，卷积神经网络能够有效的从心电图中提取到心律的特征，许多研究使用卷积神经网络实现心律信号的检测识别。2017年，U Rajendra Acharya等人[14]使用了一个9层的CNN网络，直接使用原始的心跳作为输入，在MITDB上得到了94.03%的准确率。2019年，Zhi Li等人[15]则更进一步，使用深达31层的残差卷积神经网络在MITDB上实现了99.38%的准确率。同年，Hannun等人[5]设计了一个更深的残差卷积神经网络，并采用了更大的数据集，其准确度已经达到了人类专家的水平。

1.2.3 ECG信号检测网络压缩及硬件实现

相较于传统ECG信号检测方法和基于传统机器学习的ECG信号检测方法，使用深度学习实现ECG信号检测可以达到更佳的精度和泛化性。但是深度学习算法有着计算量大、参数量大的特点，这使得其在部署到硬件上时面临诸多困难。因此许多研究在利用深度学习方法实现ECG信号检测的同时，使用网络压缩的方法降低算法的参数量和计算量，从而利于部署到硬件上。常见的网络压缩方法有剪枝、量化等。

由于深度神经网络的参数量很大，因此参数具有相当的冗余性。剪枝是指删除网络中的一些不重要参数，从而降低网络的参数量。在深度神经网络中，权重参数的分布通常是正态分布的，因此有较多参数接近于零值，在计算中对推理结果的影响很小。因此可以指定一个阈值，通过删除低于此阈值的参数来对网络进行剪枝。同时对网络的剪枝也分为不同的粒度，粗粒度的剪枝是指删除卷积核中不重要的通道，而细粒度的剪枝则是指删除卷积核中不重要的权重。细粒度的剪枝会导致剪枝后的卷积核具有不规则性，难以硬件实现，因此更多的研究采用粗粒度的剪枝方法。Lu等人[16]使用粗粒度的剪枝方法对实现ECG五分类的CNN网络进行压缩，删除了权重中60%的通道，极大的降低了计算的参数量。剪枝后网络的准确率仍为99.17%，对网络的精度几乎没有影响。

剪枝的方法可以有效的降低硬件的计算量，但是需要硬件具有更复杂的计算控制逻辑，同时硬件在计算时也会有更高的空置率。而量化的方法可以直接降低激活和权重的位宽，能够直接有效的降低硬件部署的资源需求，是在嵌入式设备上部署神经网络算法更为直接有效的手段。

量化是指将网络的浮点参数映射到整型参数，浮点参数和整型参数之间的误差会导致网络精度的损失，但是这种损失可通过量化后的再训练来弥补。Ribeiro等人[17]将ECG五分类CNN网络量化到8bit，模型权重参数量压缩了4倍，而其准确率却仅从99.9%下降到了99.6%。Wong等人[18]则进一步将ECG五分类CNN网络量化到1bit，量化后的网络准确率达到了97.5%。可见量化能够更大程度的压缩网络参数，从而可以为硬件带来更低的计算消耗和存储消耗。在嵌入式设备上实现ECG信号检测的场景中，能效比是十分重要的指标，而极端的二值量化可以有效降低算法对硬件的资源需求，因此二值量化是此场景下直接有效的手段。

由于二值量化的方法将权重和激活限制到1bit，大大降低了部署在硬件上的带宽需求。同时1bit的乘法运算可以用同或运算实现，部署在硬件上时无需进行昂贵的乘法运算。因此近年来，将神经网络二值量化后部署到硬件上成为研究的热点。

最初在硬件上实现的二值化卷积神经网络加速器和BNN的前向推理流程保持一致[19][20]。2017年，Zhao[19]按照典型BNN推理流程设计了其专用加速器，即第一层的输入不做量化处理，同时网络的基本块按照卷积层、池化层、批处理归一化层（Batch Noramlization，BN）和激活层的顺序组成。其为整型输入卷积、二值输入卷积和全连接层三种不同类型的计算分别设计了专用的硬件，并在池化后执行归一化和二值量化。

2017年，Umuroglu等人[21]提出了BNN加速器FINN，其中直接将网络的第一层输入量化到二值，通过一定的精度损失换取更精简的硬件设计。2018年，Liang等人[22]针对第一层输入多比特的问题,通过添加一个移位器的方式，将不同位的输入分别和权重进行卷积运算，并将计算中间值根据其位数进行移位后累加到中间值，使得在增加较少额外硬件的情况下支持第一层多比特输入。2019年,Guo等人[23]则通过将多比特输入展开为多通道单比特输入的方式消除了多比特输入的问题。研究人员尝试使用各种手段以实现使用统一的硬件架构实现BNN中各种类型的运算，使硬件更加简洁高效。

2018年，Ghasemzadeh等人[24]针对BNN的精度损失问题，引入了残差量化网络ReBNet，通过多级残差连接来增加激活值的信息量，同时只增加了较少的额外硬件资源。此设计在使用统一的硬件高效的实现各种类型计算的同时，让硬件可以灵活的处理不同位宽的激活输入，提供了在精度和硬件资源消耗之间权衡的手段。

2018年，Liang等人[22]结合FPGA的六输入LUT基本单元，设计了硬件友好的POPCOUNT压缩树算法，在FPGA上使用较少的硬件资源实现乘累加。2019年，Wang等人[25]提出LUT-NET的硬件架构，通过将网络稀疏化，将卷积运算展开成布尔逻辑表达式，可以充分利用上FPGA基本单元LUT。这类研究通过充分发挥BNN的特性，实现高效的硬件设计。但是缺乏灵活性，无法支持不同位宽激活输入的计算。

综上所述，深度学习的方法在ECG信号检测任务中相较于传统的ECG信号检测方法和传统机器学习方法有着泛化性好、精度高的优点，因此成为目前ECG信号检测的主流方法。然而深度神经网络具有参数量大、计算量大的特点，在嵌入式设备中部署面临资源受限、带宽受限的困难。对深度神经网络进行压缩是解决此问题的有效手段，其中量化的方法特别是二值量化的方法可以极大程度的降低算法的参数量，相较于剪枝的方法能够更有效地部署到硬件上。因此本文采用二值量化的手段对ECG信号检测分类网络进行压缩，并设计专用的硬件加速器将算法部署到硬件上。

本设计的算法可以直接使用长时心律信号进行输入，因此在硬件部署中可以直接使用原始ECG信号作为输入，免去了心拍分割带来的额外硬件开销。在二值量化方面，本文使用的量化策略实现了权重和第一层以外的激活的全二值化，且将多个缩放因子和BN层系数融合为统一的阈值，使得网络可以更高效地部署到硬件上。在硬件设计方面，本设计充分利用二值量化的特性，使用同或运算代替乘法，同时充分地进行数据复用以减少访存次数，提高了硬件的能效。最终通过利用二值量化，实现高能效的端到端的ECG信号检测分类硬件加速器，具有较大实用价值。

1.3 研究内容及设计指标

1.3.1 研究内容

本课题的目标是设计一个基于BNN的高能效ECG检测卷积神经网络加速器，并在FPGA平台上验证实现。在保证精度的前提下，充分利用BNN网络内存占用小和计算开销少的特点，减少在硬件上部署神经网络的资源和带宽占用，实现高能效的嵌入式神经网络加速器。

本课题的主要内容分为如下三部分。

（1）基于BNN的ECG检测卷积神经网络的设计：主要包括网络结构设计和网络的二值量化。网络结构设计包括网络基本块的设计，以及网络深度和宽度的结构探索，最终确定一个最优的网络结构。网络的二值量化包括量化流程的设计，以及量化完毕后参数的定点化，在压缩网络参数的同时保证一定的网络精度。

（2）基于BNN的CNN加速器架构：采用了层折叠的数据流设计，并通过数学建模分析了不同加速器尺寸下的能效和硬件资源占用，确定了最优的加速器尺寸。根据二值量化的特点，设计了专用的计算单元，使用同或运算代替二值激活和二值权重的乘法运算，减少了硬件资源消耗。同时将量化的缩放因子和批处理归一化层的系数融合，使用统一的阈值比较单元处理，同时实现结果的缩放和量化，简化了算法中的非二值运算，从而降低了硬件资源占用。此外，设计了高效的数据复用模块，从而降低访存开销，最终实现高能效的硬件设计。

（3）硬件实现和性能测试：将基于BNN的ECG检测CNN硬件加速器在Xilinx ZYNQ 7035 FPGA平台上综合、实现，对其功能进行验证，同时测试了性能功耗等参数。最后与其他ECG信号检测的CNN硬件加速器的FPGA方案进行了对比和分析。

1.3.2 设计指标

对心电图信号的分类，包括单次心跳的心拍分类和长时间心律信号的心律分类，本文的主要研究内容为对心电信号中的异常心拍的检测识别。在MIT-BIH数据集的注释中，有15种心拍的类别。其中，有五种心拍的样本数量较多，这5种心拍包括：NOR、AP、LBBB、RBBB、PVC，因此许多研究将这五种常见心拍的检测分类作为算法精度的衡量标准。由于不同数据集之间的注释分类方式有差异，AAMI为了统一标准，便于比较不同算法的性能，将各种心拍归纳为五大类，以五大类心拍分类作为比较算法性能的标准，这五大类包括：N、S、V、F、Q。由于其中V类和S类的心拍更为常见，所以AAMI也建议可以主要以这两类的分类性能作为比较的标准，称之为VEB和VSEB。VEB是指V类和非V类的二分类任务，VSEB是指V类、S类和其他类型的三分类。在本文中主要以AAMI标准的五分类作为性能的评判标准。

对于一个标注为真的样本，若算法的分类结果也为真，则称之为真阳性，即TP（True Positive），相反若分类结果为假，则称之为假阴性FN（False Negative）。相应的，假阳性FP（False Positive）指的是标注为假的样本被算法分类为真，真阴性TN（True Negative）则是指标注为假的样本被分类为真。在心律检测分类算法中，通常比较的主要指标是精确率Accuracy，它反映了算法正确分类样本数占总样本数，具体表达式如（1，1）所示。

本文参考表1.1中的文献制定了算法精确度的指标，[26]和[27]都是基于AAMI的标准对心拍进行五分类，在公开的MIT-BIH数据集上取得了优异的性能,所以对本文的算法性能有着指导作用。相较于最先进的算法[26]，本文的算法和其参数量相仿，但是免去了心拍分割的额外处理，同时在准确率上只下降了1.93%。

表1.1 ECG心拍分类准确率指标对比表

|  |  |  |  |
| --- | --- | --- | --- |
| **参数** | **本文** | **Wang**[26] | **Kiranyaz[27]** |
| **识别准确率**  **（%）** | **96.45** | **98.38** | **96.59** |
| **权重参数量** | **206144** | **198037** | **-** |
| **数据集** | **MIT-BIH Arrhythmia** | | |

通常使用吞吐率GOPS（Giga Operations Per Second）来作为神经网络加速器的算力衡量标准。吞吐率代表加速器每秒可以执行乘法和加法的运算次数，它可以直观的反映出神经网络加速器的运算性能。加速器的能效可以用用每瓦能够带来的算力来表示，单位为GOPS/W。

由于不同算法的神经网络结构差异较大，实现不同神经网络算法的硬件加速器之间无法直接比对。为直观的表现本设计的优越性，选取了和本课题使用相似结构、面向相同数据集的ECG检测神经网络加速器来进行比对，说明本设计能充分发挥BNN的特性，实现高能效的ECG检测卷积神经网络加速器。

2021年的Wei[28]，Lu[29]设计的ECG信号检测神经网络加速器，均采用一维卷积神经网络结构，在MIT-BIH Arrhythmia数据集上执行对心律信号的5分类任务。这两个设计通过优化加速器数据流，充分进行输入复用和权重复用，实现了较高的资源利用率和能效。本课题的加速器采用和上述设计相似的一维卷积神经网络结构，在相同的数据集上执行心律信号五分类任务。本设计部署在ZYNQ7035平台上，工作频率和对比设计是相同的200MHz。因此本文选择这两个设计作为表1.2的对比对象。

表1.2 基于BNN的ECG信号检测分类卷积神经网络加速器指标对比

|  |  |  |  |
| --- | --- | --- | --- |
| **参数** | **本文** | **ISNE2021[28]** | **TCAS2021[29]** |
| **验证平台** | **ZYNQ7035** | **ZC706** | **ZC706** |
| **工作频率（MHz）** | **200** | **200** | **200** |
| **参数位宽** | **1bit** | **16bit** | **16bit** |
| **识别准确率（%）** | **92.83** | **98.94** | **99.10** |
| **吞吐率**  **（GOPS）** | **51.6** | **26.6** | **25.7** |
| **能效**  **（GOPS/W）** | **146.2** | **32.5** | **\_** |
| **数据集** | **MIT-BIH Arrhythmia** | | |

综上所述，本文所实现的指标如下：

（1）吞吐率：在ZYNQ7035平台上，在200MHz频率下，达到>40GOPS的吞吐率。

（2）能效：在ZC7035平台上，在200MHz频率下，达到>60GOPS/W的能效。

（3）准确率：在MIT-BIH Arrhythmia数据集上对心律信号进行5分类，准确率大于92%。

本文的设计在准确率仅降低6.11%的前提下，相比现有的对标文献，实现了更高的吞吐率和能效比。这主要归功于二值量化的有效应用，降低了计算单元的硬件资源需求，从而提升了能效性能。同时，加速器采用了并行运算和参数复用的策略，增加了运算单元的工作效率，实现了更大的吞吐率。本文的加速器兼顾了二值量化的优势和精度的要求，实现了高能效的ECG检测识别硬件。

1.4 论文组织结构

本文一共分为六个章节，论文的组织架构如下：

第一章为绪论，首先介绍了心律异常检测的重要性和卷积神经网络在ECG信号检测识别上的优势，然后分析了ECG检测算法和神经网络量化压缩方法及硬件实现的国内外研究现状，最后阐述了本文的研究内容、设计指标和论文章节安排。

第二章为卷积神经网络原理和神经网络压缩方法的综述，首先对传统ECG检测方法进行介绍，分析了深度学习方法在ECG检测上的优势。然后介绍了卷积神经网络的基本原理，并进一步介绍了神经网络量化的相关方法和原理，重点分析了二值量化方法及其在硬件设计中的优化策略。

第三章为ECG检测卷积神经网络与网络二值量化方法设计，首先阐述了本文使用的心电信号数据集以及采用的数据处理方法，然后介绍了卷积神经网络基本块的设计以及网络结构的设计空间探索，最后介绍二值量化的训练策略以及部署到硬件上的定点化方法。

第四章为基于BNN的ECG检测CNN硬件加速器设计，首先分析了加速器的整体数据流，并对加速器整体设计空间进行探索，根据能效比和硬件资源占用率确定了加速器的尺寸维度。然后介绍了本文根据二值化网络特点设计的PE计算单元。随后介绍了加速器其余硬件模块的具体功能，包括主控制模块、缓存模块、存储模块和计算模块。最后通过举例说明了加速器的整体计算过程。

第五章为ECG检测卷积神经网络硬件加速器的FPGA验证，在Xilinx ZYNQ 7035 FPGA平台中搭建基于BNN的ECG检测卷积神经网络硬件加速器，通过RTL模块整合和Vivado工具的功能仿真以及综合实现，得到硬件加速器的整体功耗、资源使用情况和性能。最后，选取两个相同领域的对比设计，详细分析了不同硬件加速器方案间性能的优劣。

第六章为总结与展望，总结本文全部工作内容，并对之后的工作发展和安排做出了展望。

第二章 卷积神经网络原理和网络压缩方法概述

本章主要介绍了基于卷积神经网络的基本原理，以及对其量化压缩的方法。首先，我们回顾了ECG的传统分类方法，分析了其优缺点，以及深度神经网络在ECG分类上的优势和挑战。其次，我们介绍了卷积神经网络（CNN）的基本原理和结构。然后，我们介绍了量化的概念和方法，特别是二值量化的原理和优势，以及它在降低CNN的计算复杂度和存储需求上的作用。最后，介绍了二值量化专用硬件设计的基本方法。

2.1 传统ECG信号分类方法

心电图上的四个波形，即P波、QRS波群、T波和U波，反映了心脏各部位的除极和复极过程。一个完整的心拍周期从P波开始到下一个P波之前结束，包含P波、QRS波群、T波和U波，以及它们之间的间期和段，如图2.1所示。传统的ECG信号分类方法是分阶段的，需要对ECG信号进行去噪、分割得到心拍，然后从心拍的各个波段中提取特征来进行分类。分类的方法主要有基于结构模式识别的方法和基于统计模式识别的方法。

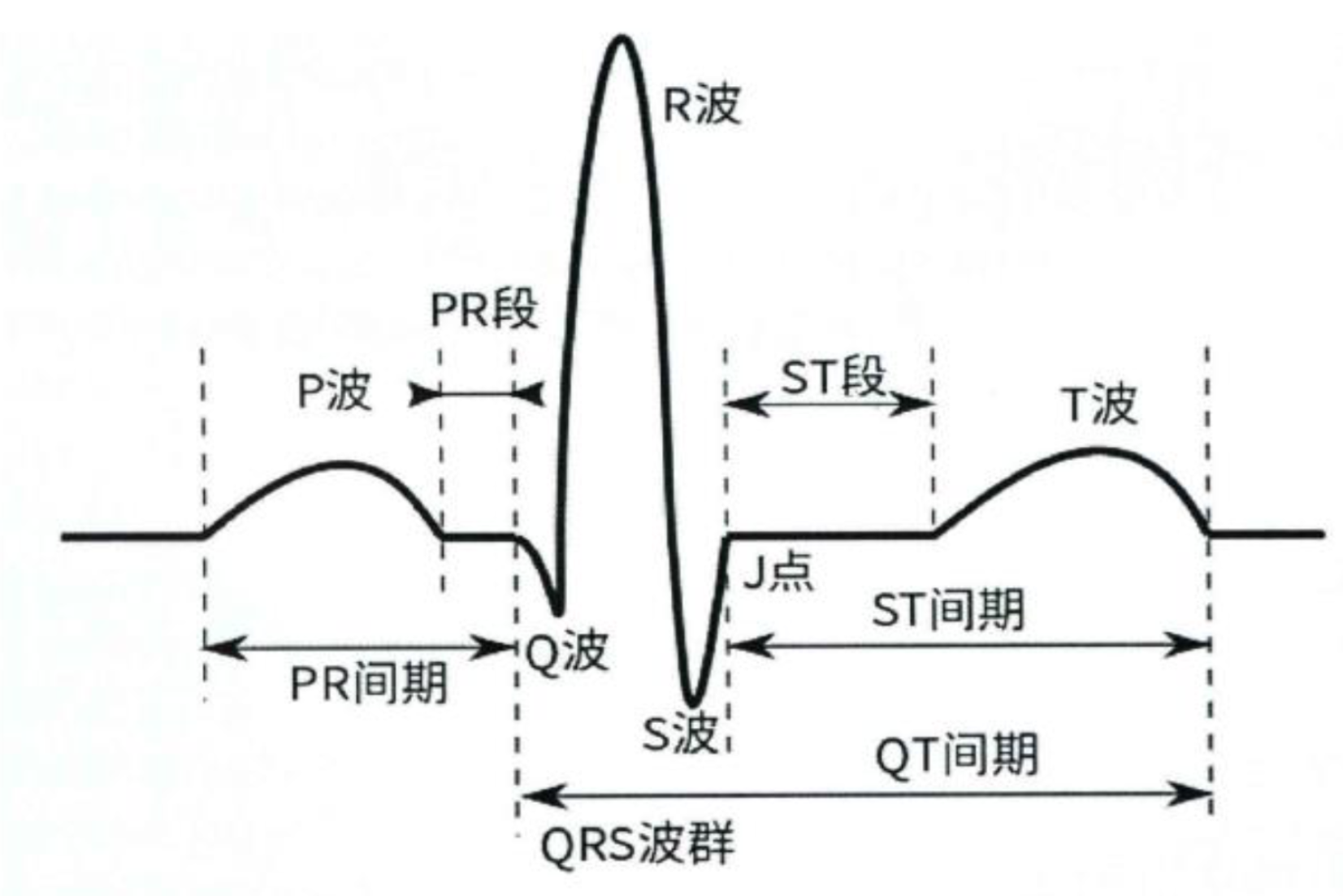


图2.1心拍示意图

基于结构模式识别的方法利用心电图波形的形状和结构特征来进行心拍分类。这类方法需要从心电图波形中提取有意义的特征，如幅度、持续时间、斜率、面积等，并将它们作为分类器的输入。如Kumar[30]使用ST段的斜率和长度等信息，并与预设的阈值进行比较，来判断是否为缺血性心拍。Miad[6]则通过分析心拍是否超出正常区域范围来诊断异常心拍。基于结构模式识别的方法依赖于心拍的分割质量，且分类标准由手工制定，只能实现简单的二分类任务。

基于统计模式识别的方法利用心电图波形的统计特征来进行心拍分类。这类方法需要从心电图波形中提取有意义的特征，如均值、方差、偏度、峰度、熵等，并将它们作为分类器的输入。通常使用主成分分析、独立成分分析等方法建立模式空间，然后使用支持向量机、随机森林树等机器学习的方法对ECG信号进行分类。Shen[31]等人使用K邻近算法和支持向量机实现了心拍的13分类，准确率达到98.92%。

基于统计模式识别的方法相较于基于结构模式能够实现更加复杂的分类任务，且分类的标准不依赖于专家知识，而是可以通过训练分类器来实现。但是其特征的选取依然是基于手工，导致网络泛化性不佳，同时分阶段的特征提取和特征分类可能造成信息损失。而深度学习网络可以自动地从ECG信号中提取特征，充分利用原始信号中的信息量。特别地，深度学习的方法还可以实现端到端的心律识别检测，让算法自动地提取特征并分类，在简化推理流程的同时还能依靠深度网络的学习能力取得更佳的分类性能。

2.2 卷积神经网络基本原理

卷积神经网络算法的起源最早可以追溯到60年代关于生物大脑皮层对视觉信息处理的研究[32]，Fukushima[33]实现了最早的卷积神经网络雏形。1998年，LeCun[34]第一次提出了卷积网络（Convolutional network），设计了著名的网络结构LeNet-5。2012年，AlexNet[35]在ImageNet图像分类大赛中展现出远优于其他网络的性能，卷积神经网络迅速成为研究的热点。

典型的卷积神经网络的基本骨干结构包括卷积层、激活层、池化层和全连接层等，本小节将详细介绍上述卷积神经网络的各个基本组成单元。

（1）卷积层

卷积层是卷积神经网络的核心部分，能够从输入信号中提取出丰富的特征。卷积计算受到感受野的启发，具有平移不变性和局部相关性的特点。如图2.2所示，卷积计算将输入和权重经过卷积的方式乘累加，最终得到输入。

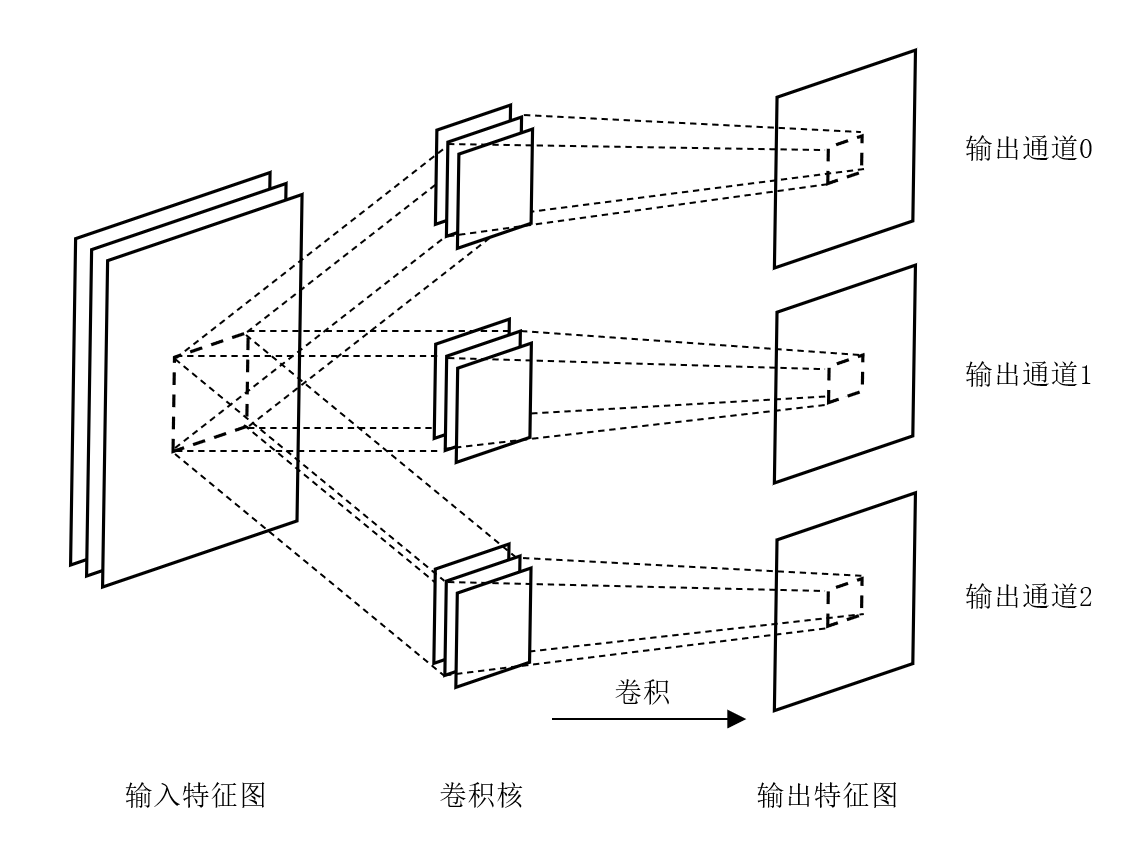


图2.2 卷积计算示意图

卷积核在输入特征图上滑动的具体过程如图2.3所示。卷积核从输入特征图的左上角开始滑动，例如一个3\*3的卷积核和输入特征图左上角3\*3的矩阵进行计算，对应位置的权重和输入相乘，并将所有的乘积累加，得到一个输出结果。由于输入特征图有多通道，不同通道的乘累加结果要进一步累加，得到最终的输出结果。计算完毕之后，卷积核相乘的输入对象向右滑动一个步长（Stride），重复上述的步骤。若已经滑动到最右，则执行下一行的滑动，整体呈Z字形的滑动方式，直到所有输入都被计算完毕。由于卷积计算的特性，输入特征图的边角可能会损失信息，所以在输入特征图周围添加一圈零值，从而使得不损失输入的边角信息，这个过程称之为填充（Padding）。

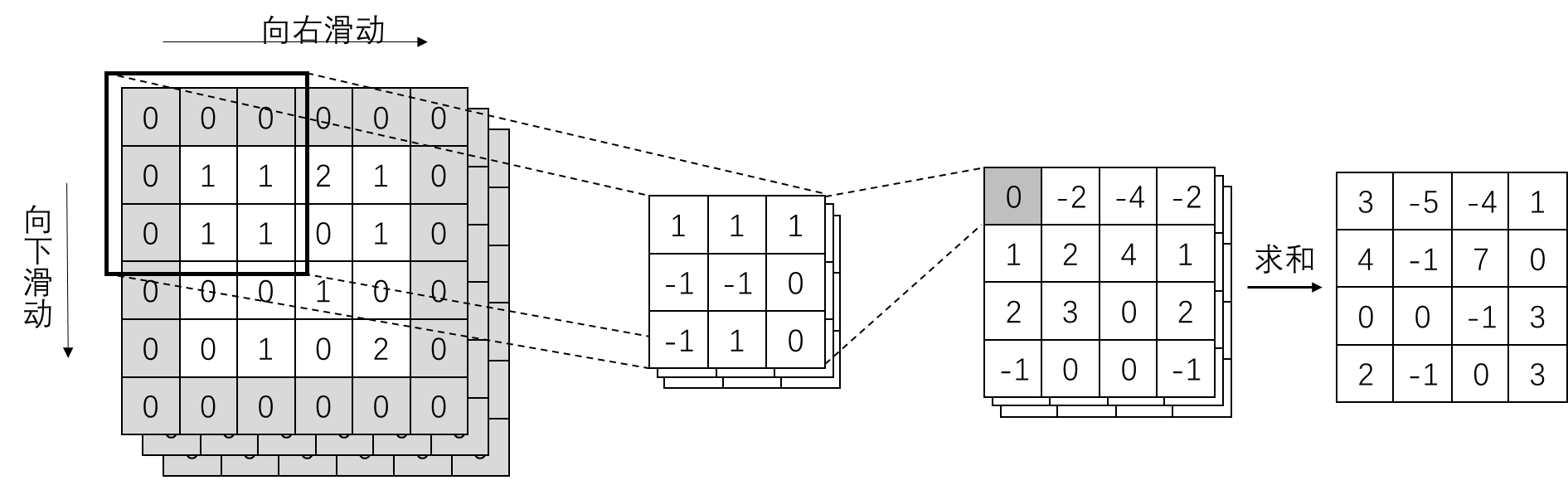


图2.3 滑动卷积过程示意图

（2）激活层

卷积的计算本质而言是对输入信息进行乘累加计算，其函数本质是一个线性的函数，因此需要引入非线性的激活函数来增加网络的表达度。目前广泛使用的经典激活函数包括Relu、Tanh、Sigmoid等。Relu函数的图像及其梯度如图2.4所示，Relu函数的表达式即为,即将小于零的输入均置零。

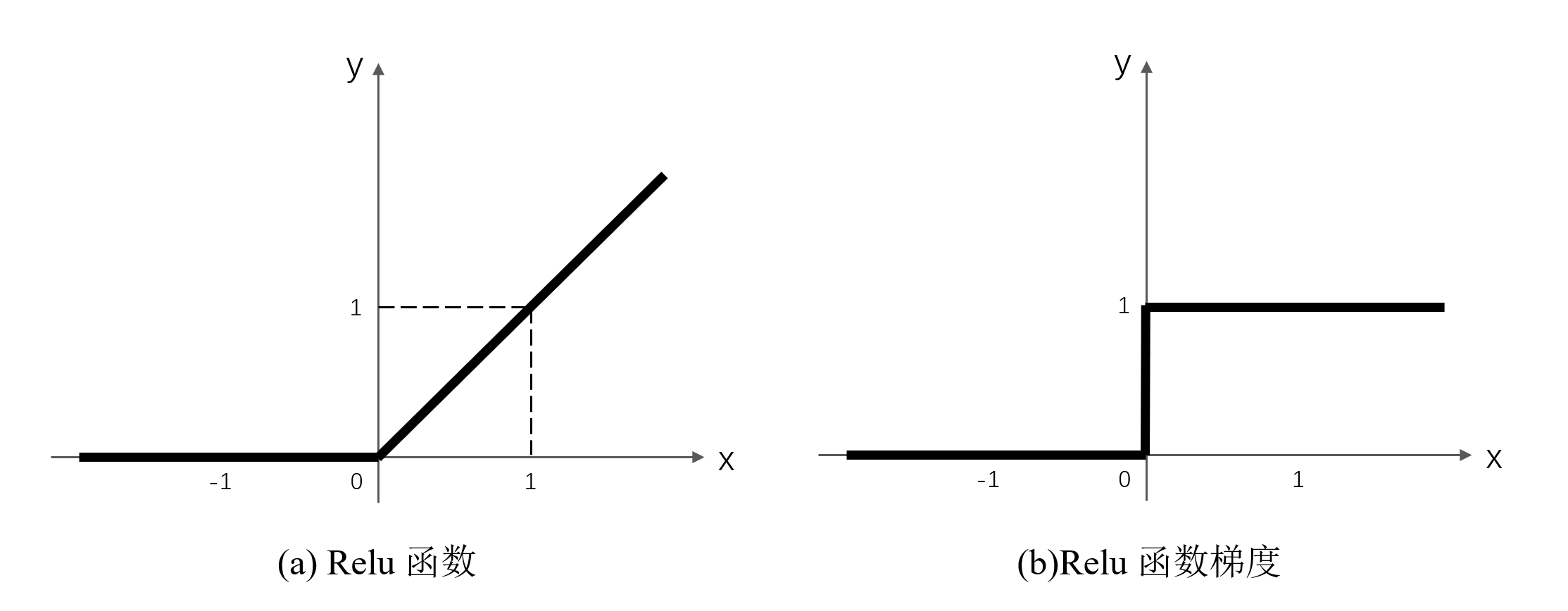


图2.4 Relu函数及其梯度

（3）池化层

在卷积运算中，我们通常通过设置步长和填充零的数量，使得卷积的输出和输入的维度相同。为了降低运算的计算量，同时也为了对特征进行提取，所以引入了池化的操作。主流的池化操作包括最大池化（Max Pooling）和平均池化（Average Pooling），如图2.5所示。最大池化是取池化窗口中最大值为输出结果，而平均池化则是取池化窗口中所有值的均值作为输出结果，因此考虑到计算的简便性，通常采用最大池化的方法。

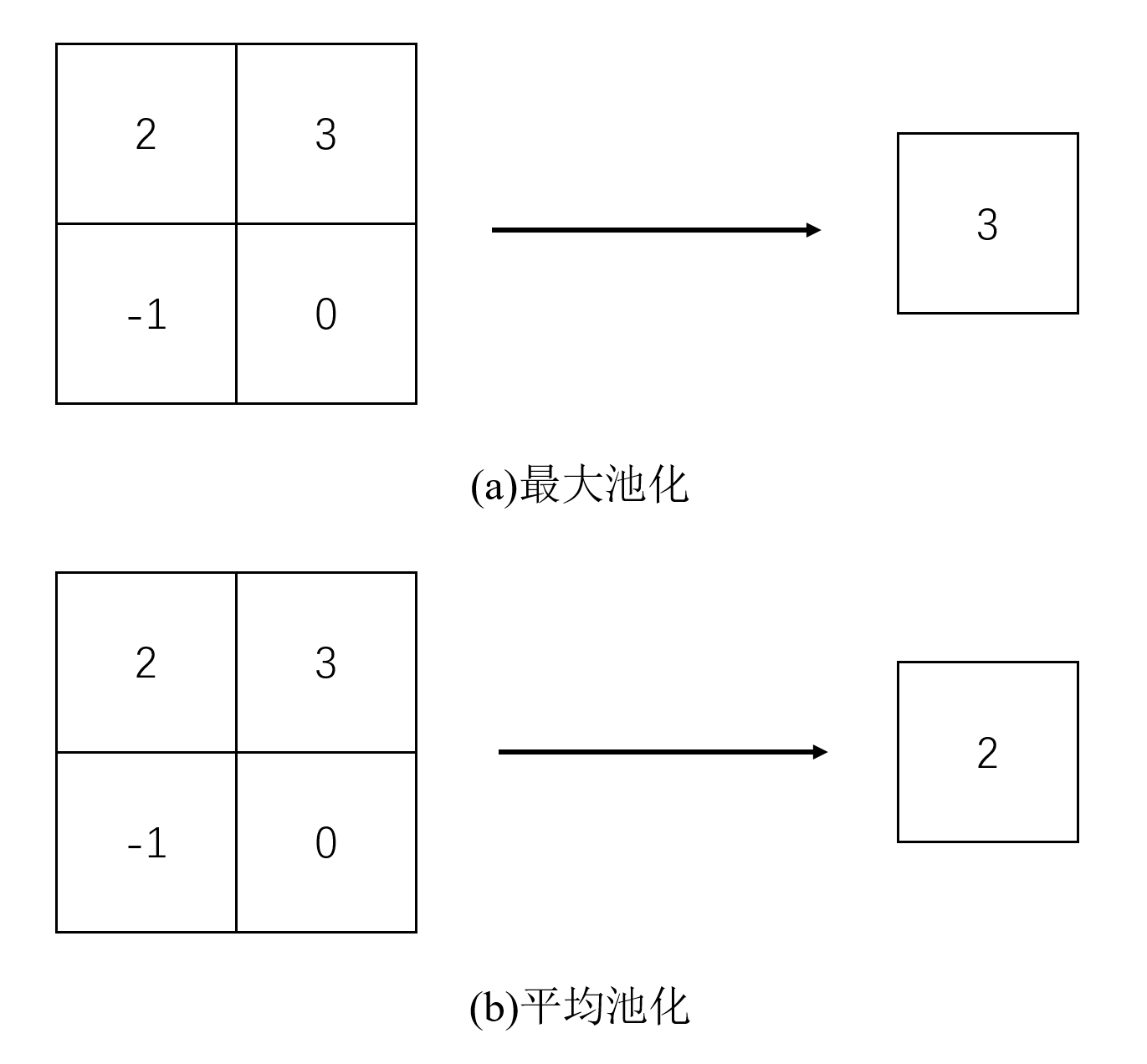


图2.5 池化示意图

（4）全连接层

卷积层有着良好的特征提取能力，而全连接层则负责执行对提取特征的分类。如图2.6所示，全连接层的每个输出都和所有输入有关，是所有输入和权重乘累加之和。

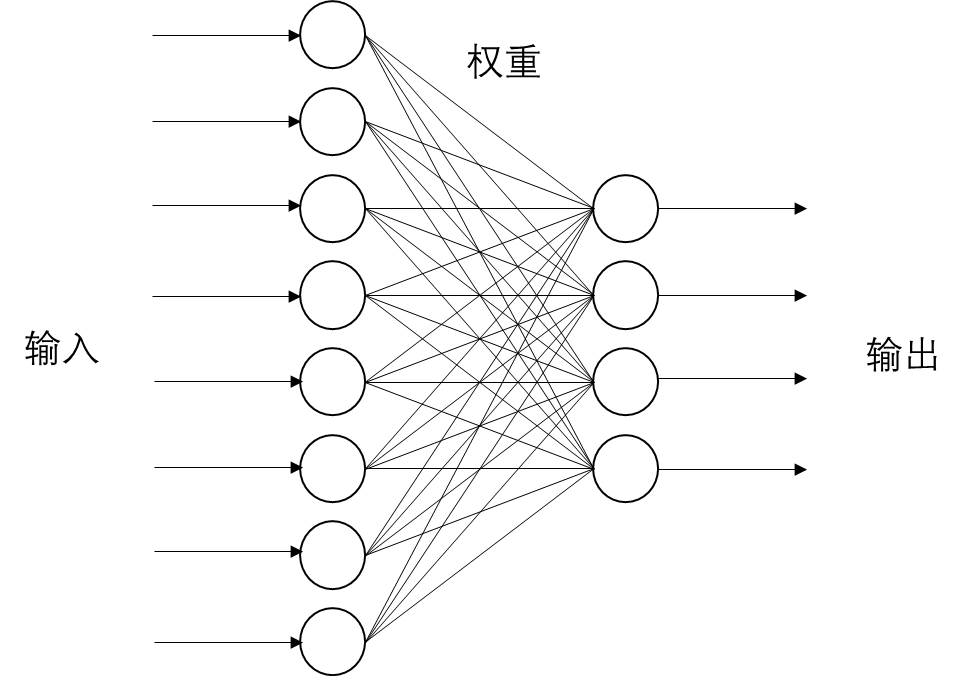


图2.6全连接层示意图

（5）批量归一化层

批量归一化(Batch Normalization)自提出以来[36]，被广泛的应用到神经网络之中。在训练过程中，网络的参数在每次反向传播更新之后都会发生变化。虽然输入数据的样本分布是固定的，但是由于参数的变化可能导致后续几层网络的输入范围变化波动较大，特别是对深层的网络而言。批量归一化层则是将每层的输出进行归一化处理之后，再输入到下一层之中，缓解了输入波动导致的网络收敛困难。批量归一化层的前向推导函数如公式(2.1)所示，其中和均是可学习的网络参数，而和则代表选取的一批输入的方差和均值，则是一个固定的常量。

2.3 卷积神经网络量化方法

由于神经网络的深度不断加深，网络的参数量变得十分巨大，这使得嵌入式设备的算力难以负担神经网络庞大的计算量。因此对卷积神经网络压缩的方法成为研究的热点，而针对本文所需要解决的问题，量化是最为直接有效的手段。研究表明[37]，降低数据的位宽能够显著降低计算的功耗和硬件资源的占用，如图2.7所示。对神经网络进行量化对于在硬件上部署神经网络而言是十分有益的，本小节接下来将详细的介绍卷积神经网络的相关量化方法。

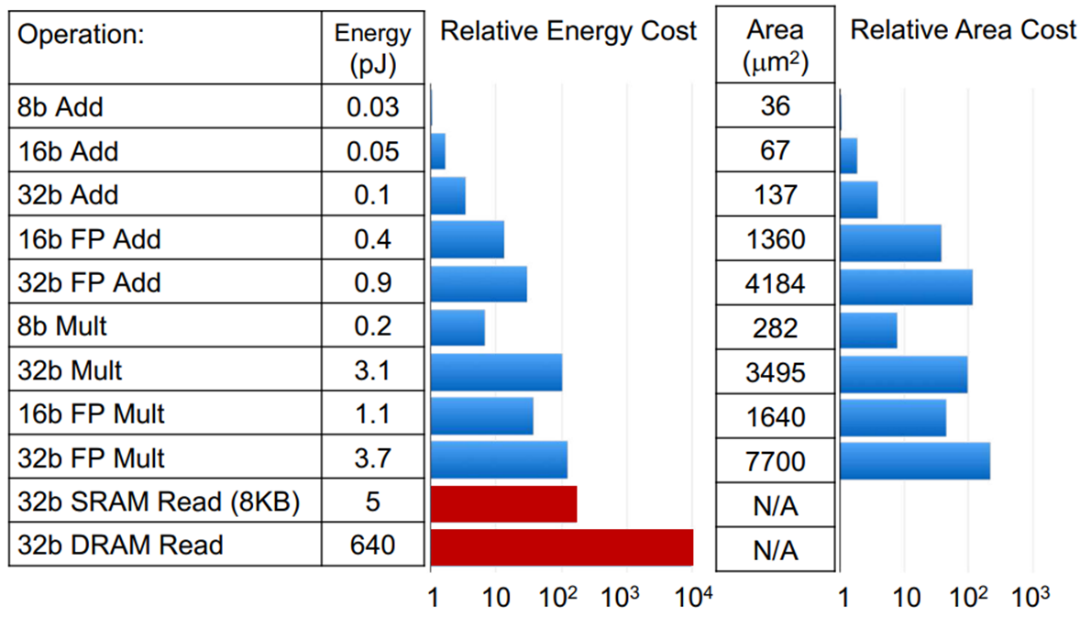


图2.7不同数据位宽计算的硬件功耗和面积对比

2.3.1 卷积神经网络量化原理

卷积神经网络的量化是指将激活和权重的位宽进行削减，使用低位宽的权重和激活进行卷积计算。因此量化的本质是确定一个函数，其能够将原始32位浮点的参数映射到一个唯一的低位宽的数上，这种映射会造成计算结果的不精确。如何减少精度的损失是研究的重点，目前常见的映射方式包括均匀对称量化、均匀仿射量化、幂指数量化和二值量化等，接下来将简要介绍这些映射方法。

（1）均匀对称量化

量化是浮点数向整数的映射，因此量化之前需要确定映射前浮点数的范围和映射后整数的范围。均匀对称量化是指，选取的浮点数范围和映射的整数范围均在坐标轴上是关于零点对称的，其量化的过程如图2.8所示。

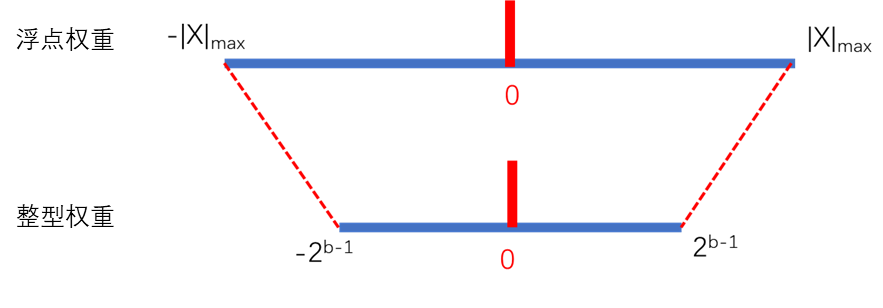


图2.8均匀对称量化映射示意图

为了使得范围关于零点对称，同时能够囊括所有浮点数，所以在均匀对称量化中浮点数的范围是,是浮点数绝对值的最大值。整数的范围则根据需要的量化后位宽来决定，范围是。确定范围之后，即可得到映射的缩放系数，如公式(2.2)所示。

有了缩放系数之后，浮点权重就可以缩放到整型之上了。将浮点权重直接除以缩放系数，然后进行取整，得到的整型数即为映射后的整型数，如公式(2.3)、公式(2.4)所示。公式(2.4)代表超出范围的输入均截取，按照范围两端的数来进行代替。

（2）均匀仿射量化

均匀仿射量化又称之为均匀不对称量化，和均匀对称量化相比，其选取的映射前的浮点数范围是不对称的，而映射后的整型数范围则一般是对称的。由于卷积神经网络参数的分布很难关于原点对称，均匀对称量化粗暴的选取绝对值最大值映射的方式会导致整型数表达范围的浪费，并使得精度不佳。均匀仿射量化选组浮点数的最大值和最小值作为待映射的范围，即。而整型数的范围依据量化后位宽，范围是。缩放系数的计算方法如公式(2.5)所示。

由于选取的浮点数范围并不关于零点对称，而映射后的整型范围是关于零点对称的。因此浮点数除了缩放系数之外，还需要一个关于原点的偏移值，它的计算方法如公式(2.6)所示。

最终映射的整数计算方法为，先将浮点数除以缩放系数，然后取整之后再加上偏移值，最后对超出范围的输出进行截取，最终得到的结果即为映射后的整型数。具体过程如公式(2.7)所示。

均匀仿射量化的映射过程如图2.9所示。均匀仿射量化解决了因为浮点数关于原点不对称使得映射整型数范围浪费，导致精度损失的问题。但是相应的引入了额外的偏移值，这导致在计算过程中引入了额外的计算量，因此在实践中需要具体衡量相关问题。

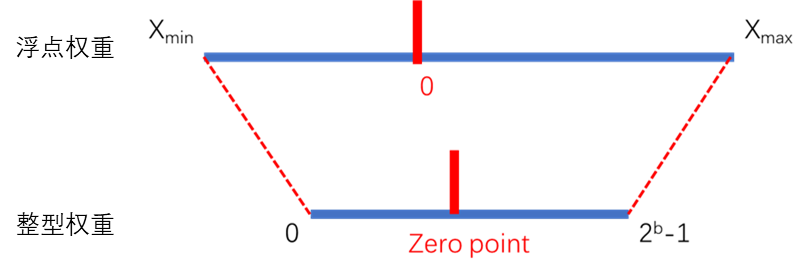


图2.9均匀仿射量化映射示意图

（3）幂指数量化和二值量化

幂指数量化是指映射的整型数均为2的幂指数。幂指数的分布符合正态分布，这和网络参数的分布相似。因此幂指数的优势在于在网络参数密集分布的地区有大量的整型数来近似，从而获得更小的总的近似误差。二值量化则是更为简单粗暴的映射方法，直接根据浮点数的输入的符号位，将输入映射到+1或者-1，这在极大程度压缩网络参数的同时也带来了很大的精度损失，需要巧妙的设计训练方法来获得最佳的效果。

2.3.2 卷积神经网络量化策略

在确定了量化的映射规则之后，就需要制定策略将映射规则部署到卷积神经网络的参数之上。典型的部署策略包括训练后量化和量化感知训练，即在网络参数训练完毕之后离线的进行映射和在网络训练过程中在线的映入量化。接下来将对这两种策略进行简要的介绍。

（1）训练后量化

网络训练完毕之后，权重是固定的，但是输入时未知的。因此需要选定一个分布和数据集相似的参考集，以此参考集的数值范围作为激活值的浮点数范围。所谓训练后量化就是将权重和激活各取定一个范围，然后将其映射到整型数之上，映射的具体方法主要是上述介绍的均匀对称量化和均匀仿射量化。

由于量化的映射使用一个不精确的值代替一个精确的值，那么一次性的精度损失是不可避免的。如何尽力避免损失就是需要解决的问题，也就是使得浮点数和整型数的差之和最小的优化问题，主流的方法包括最小化均方误差、最小化交叉熵等。

（2）量化感知训练

不同于训练后量化离线的一次性量化，量化感知训练将量化本身引入训练过程。由于神经网络强大的学习能力，可以使得网络本身适应量化噪声的影响，从而获得更好的精度。量化感知训练的具体过程如图2.10所示，将量化函数引入前向推导和反向传播的过程之中。在前向推导的时候，激活值和权重均经过量化函数映射为整型的和。在反向传播的时候，也需要乘上量化函数的导数，这也就要求量化映射的函数必须是可微分的。

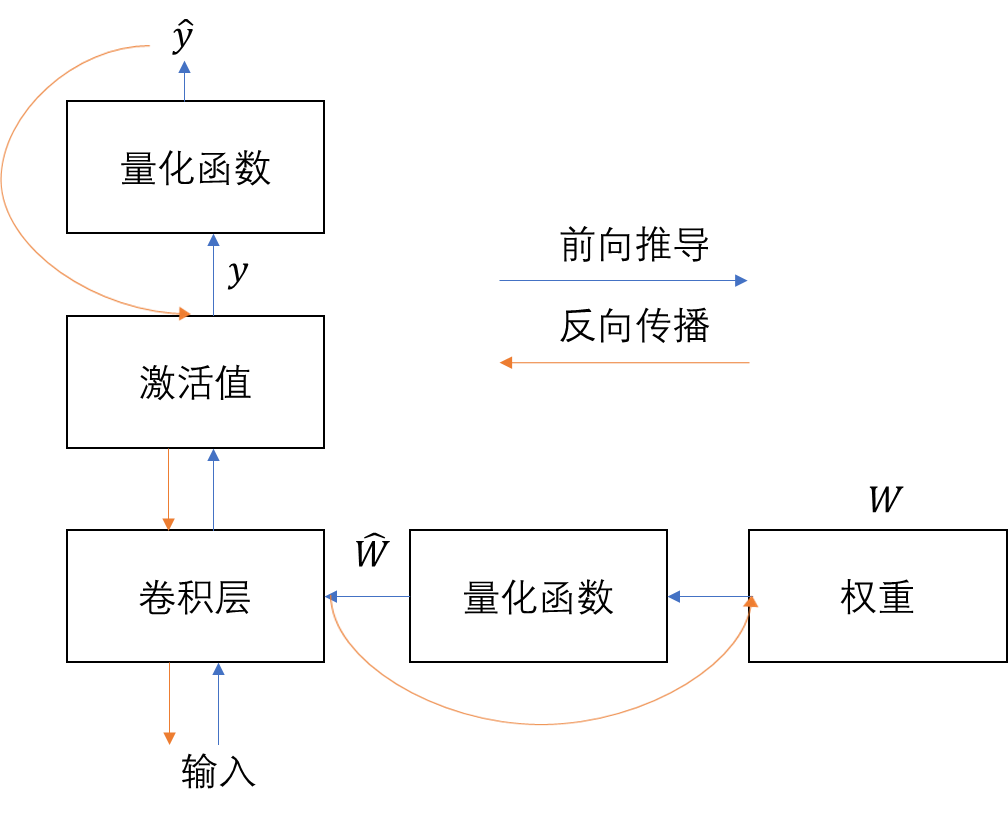


图2.10量化感知训练示意图

2.4 卷积神经网络二值量化及其专用硬件设计

2.4.1 卷积神经网络二值量化简述

在上一节中本文简要介绍了量化的相关原理，基于本文目标的严苛应用场景，二值量化作为一种极致的量化方法是一种合适的选择。二值量化的映射函数如公式(2.8)所示，如果待量化的浮点数大于等于零，则将其映射为+1，否则均映射为-1。不同于公式(2.8)中使用确定的函数进行二值化的方法，最早的Matthieu Courbariaux提出二值化网络[38]时采用了一种随机二值化的方法，更符合实际情况的使用概率来二值化，但是由于其随机性难以在硬件上实现。

由于二值化的极致简化，必然会产生很大的误差，为了减少精度的损失，一般均采用量化感知训练的策略来进行量化。由之前的介绍可以得知，量化感知训练要求量化函数时可微分的，而二值量化函数及其导数的图像如图2.11(a)所示。可以看出，二值量化函数的导数在零点为无穷大，其余为零。零值的梯度会屏蔽掉反向传播的更新，使得网络无法进行训练。因此Matthieu Courbariaux进一步引入直通估计器[39]，采用一个二值量化函数的近似函数来在反向传播的过程中代替二值量化函数。这个近似的函数及其导数的图像如图2.11(b)所示，其导数在的范围内为1，其他则为0。采用近似函数的方法虽然使得反向传播可行，从而实现量化感知训练，但是由于近似函数导致正向推导和反向传播使用的函数之间的误差，产生了梯度不匹配的问题，这会导致网络参数训练最后收敛方面的问题，造成精度上的损失[40]。

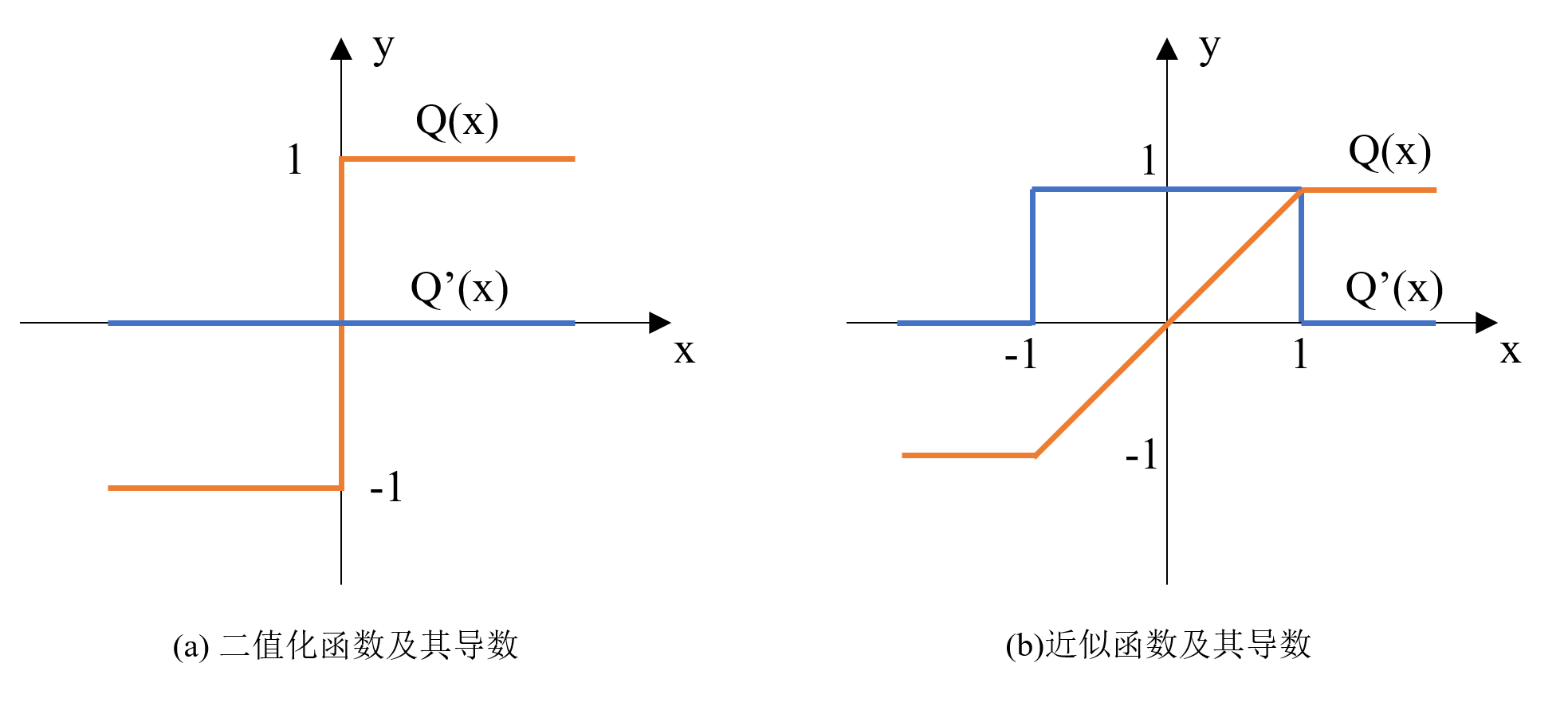


图2.11二值化函数和近似函数及其相应导数

有了映射函数和反向传播的手段，采用量化感知训练策略对网络参数进行二值化便可行了。如图2.10中所示，每层输出的激活值需要经过量化函数之后再输入下一层。二值量化的量化函数是直接对输入进行取符号操作，如果输入的待量化值范围的分布严重偏离原点，就会导致大量的参数均会被量化成相同的1或者-1，这会损失大量的信息，使得网络训练无法收敛。而2.2中介绍过，批量归一化层是解决激活分布问题的有效手段，因此为了有效进行二值量化，网络中通常需要在卷积层之后加入批量归一化层。由于输出经过非线性激活函数之后会损失较多信息，因此要将批量归一化放在激活函数之前。同时在非线性的激活函数之后要进行输出的二值化，而激活函数又将所有的负值置零，所以通常将激活函数和二值量化相结合，直接使用二值量化函数来代替激活函数。

由于浮点数和二值之间误差较大，因此优化量化损失是提升精度的重要手段。在经典的XNOR-NET[41]中为权重和输入分别引入了逐通道的缩放因子，从而增加网络的表达度，后续大多数工作均采用了此方法。缩放因子的计算方法如公式(2.9)所示，均为待量化的参数的L1范数除以参数数量n。

综上所述，典型二值化网络的构造如图2.12所示。网络参数的二值化可以极大的简便计算，但是会带来精度上的损失和训练上的困难。在精度上，通过引入缩放因子、增加激活位宽等方式，可以实现精度和参数量之间的权衡。在训练上，需要精心设计量化流程，避免一次性量化损失大量信息，导致网络收敛困难。因此需要精心设计训练流程，实现网络参数的二值化。同时根据应用需要，引入缩放系数、增加激活位宽，实现精度和参数压缩的平衡。

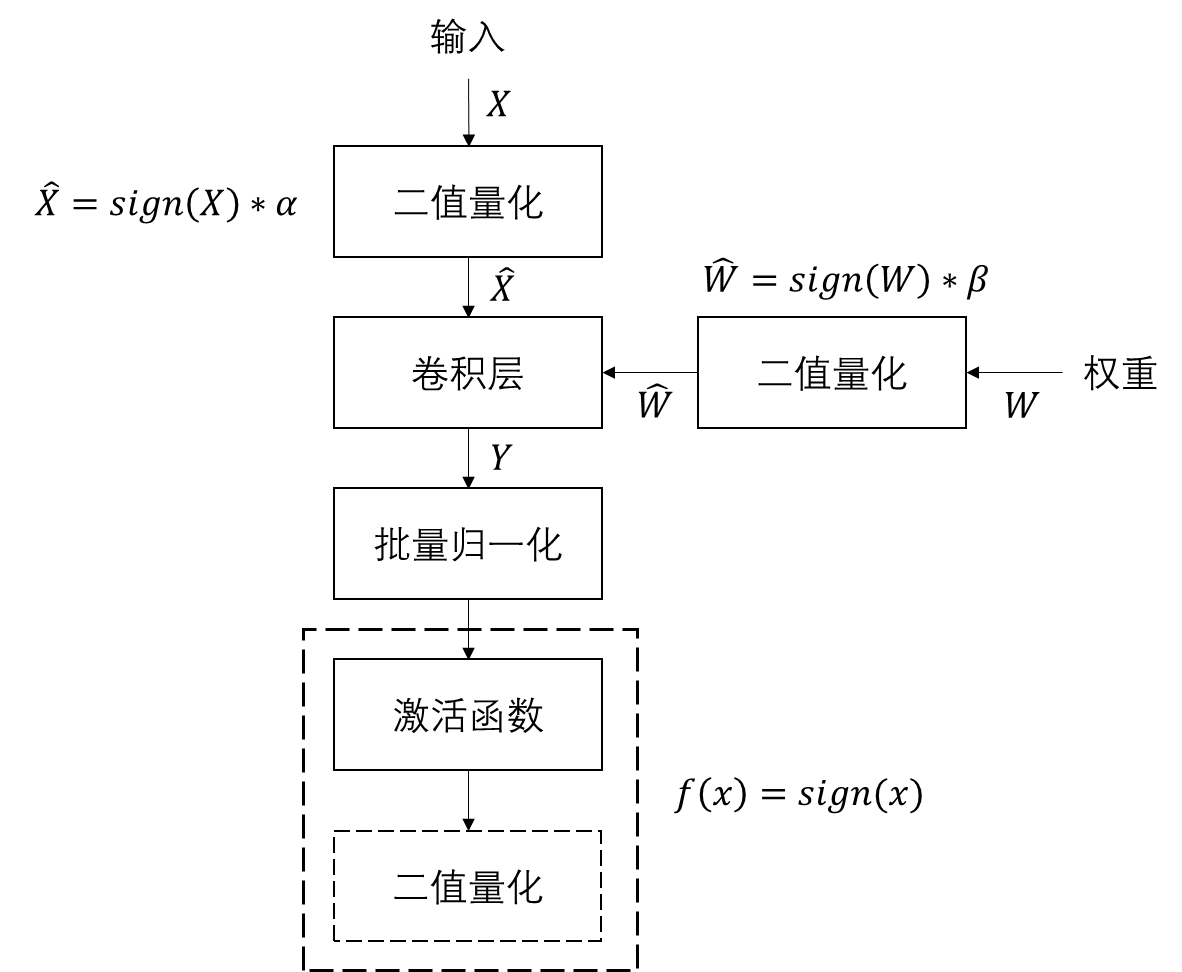


图2.12典型二值化网络基本块

2.4.2 卷积神经网络二值量化专用硬件简述

二值量化作为一种硬件友好的量化方法，除了部署在通用处理器上之外，设计专用的硬件加速器能够更好的发挥其参数量小、计算量低的优势，提高硬件的能效比。上一小节介绍了典型的二值量化网络的基本块，大部分硬件相关的研究都是以此基本块为基础的网络结构来进行研究的。针对基本块的各个部分，不同研究提出了不同的硬件设计方法。主要包括多比特输入二值化权重卷积模块、全二值化卷积模块、批处理归一化层模块以及全连接层模块的硬件设计优化，接下来将简要介绍这些设计方法。

（1）多比特输入的二值化权重卷积模块

第一层的原始数据输入含有着丰富的信息量，因此很多研究中均将第一层的输入保留为多比特的位宽。在硬件部署中，如何同时实现多比特输入和二值输入的卷积是需要考虑的问题。最为简单直接的方法是设计一个专门的多比特输入计算单元[19],但是这会造成资源利用率的浪费。为了提高硬件效率，则需要执行全二值化的卷积的计算单元能够支持多比特激活输入的卷积计算。一种常用的方式是在全二值化模块的计算单元中添加少量额外硬件资源来支持多比特输入，例如添加一个移位器，将多比特的输入按位宽依次输入计算，然后将计算结果按照位数移位后累加[22]，如图2.13所示。

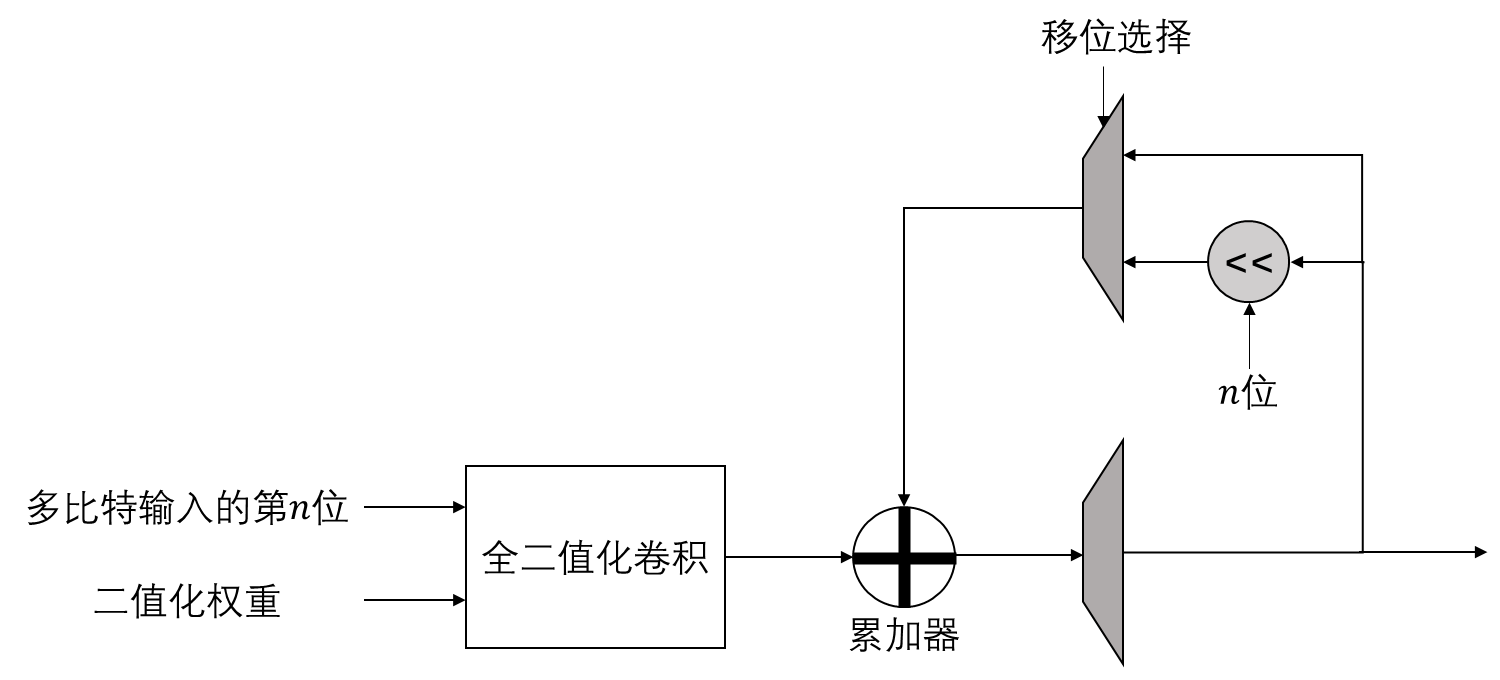


图2.13支持多比特输入的二值化计算单元

除了添加额外硬件单元之外，还有从多比特输入本身入手的解决方法。例如将多比特输入展开成多通道的单比特输入来消除多比特输入[23]，或者直接将第一层也量化到一比特来简化硬件设计[42]。由于应用需要，二值量化后的网络可能需要更多的激活位宽来保证精度，因此需要支持不同位宽激活输入的二值化卷积运算单元。高效的多比特输入的二值化权重卷积模块以较少的额外硬件资源占用为代价，提供了精度和参数量之间权衡的手段。

（2）二值化卷积模块

卷积运算的实质是乘累加运算，在二值量化之后，其就变成了单比特数之间的同或运算和单比特运算结果的累加。可以针对加速器实现的平台，实现特别的卷积模块优化设计。例如FPGA中有着丰富的LUT资源，可以和二值量化的独特计算方式有效结合，设计出高效的二值化卷积模块。对于乘法的过程，LUT可以高效的执行同或运算[42]，使得乘法简化。对于累加的过程，也可以利用FPGA基本单元六输入LUT的特性，设计POPCOUNT压缩树来减少累加的关键路径[22]，如图2.14所示。

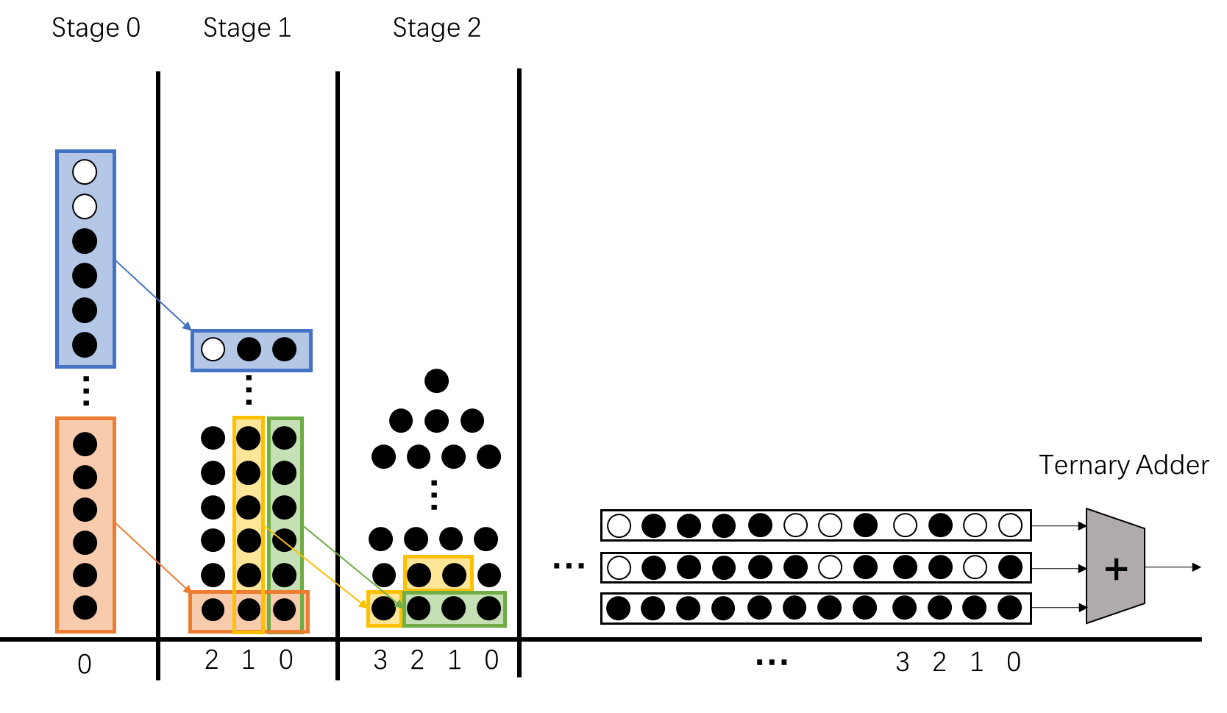


图2.14 POPCOUNT压缩树累加模块示意图[22]

除了乘法单元和累加单元的优化性设计，还可以更进一步根据二值量化的特性设计独特的二值化卷积计算单元。例如可以将二值量化之后的乘累加运算均为布尔逻辑运算，所以可以将网络稀疏化之后直接将卷积展开成布尔逻辑表达式[25]。还可以利用二值量化后卷积运算结果的有限性，复用中间值来简化计算[44]。更多的还有利用二值量化阈值跳过冗余计算[45]、引入低秩分解[46]等设计方法。

二值化卷积模块充分发挥二值化的优势，极大的简化了乘累加单元的设计。但是如展开成逻辑表达式等设计方法，使得硬件完全不同于一般的乘累加模块，从而导致其难以支持不同位宽的激活输入。

（3）批处理归一化层模块

批处理归一化层的参数和输入批次数据的均值、方差相关，因此是高位宽的浮点数值。如公式(2.1)所示，批处理归一化层的函数本质是一个线性函数。为了在硬件中高效的执行批处理归一化层，一般要将其线性函数的系数和偏置训练后量化。一种方式是将其系数进行幂指数量化，偏置则量化为多比特整型数，从而在硬件上添加移位器和加法器来实现[22]。另一种方式则是直接将批处理归一化层和后续的非线性激活层以及二值量化融合起来，得到一个阈值，通过卷积乘累加输出和阈值的比较得到二值化的输出[43]，如公式(2.10)所示。

(4)全连接层模块

在硬件上简单处理全连接计算的方法是设计一个全连接层专用的计算单元[22]，在算法上的简单处理方式是直接使用全局平均池化层代替全连接层，消除了全连接层带来的大量的参数[47]。也有研究将全连接层直接展开为矩阵运算处理[48]，就一般而言全连接层可以复用二值化卷积单元来进行计算而无需额外改造。

目前的二值化硬件主要是通过利用二值量化后的参数特点，简化卷积运算的计算单元，从而降低硬件资源的占用，实现高能效的硬件设计。然而，在不同的应用场景中，精度的要求也不尽相同，因此需要二值化卷积运算模块能够高效地支持不同位宽的激活输入。因此，二值化加速器的卷积模块既要发挥二值化特性，简化硬件设计，又要在不增加过多额外硬件的前提下，支持多比特激活值输入的二值化权重卷积计算。除了卷积计算模块外，还需要设计高效的硬件单元，执行二值化网络中的批处理归一化、缩放系数还原等非二值运算部分，最终实现高效的二值化卷积神经网络加速器。

2.5 本章小结

本章首先概述了ECG分类的传统方法，以及CNN在嵌入式平台上实现ECG分类的优势和挑战。然后简要的介绍了CNN的基本原理，并阐述了CNN量化的基本原理，由于本文面向嵌入式应用场景，所以使用极端的二值量化来获取更高的能效比。最后介绍了二值量化的基本原理，并简述了目前的二值量化专用硬件的设计方法。在目前的二值量化硬件中常用简化的计算单元来实现二值量化后的卷积运算，实现高能效的运算单元设计。如何处理运算中的非二值部分，和二值化运算单元有机结合，实现整体硬件的高能效是目前硬件设计中的主要挑战。

第三章ECG检测卷积神经网络与网络二值化方法设计

为了解决在嵌入式设备中部署ECG检测的神经网络算法时遇到的计算量大和参数量大的问题，需要设计一个轻量化、硬件友好的ECG检测卷积神经网络算法。本章针对这个目标，设计了一个卷积神经网络算法实现ECG信号的检测分类，并采用二值量化的方法对网络进行压缩，使其利于硬件部署。

3.1 ECG信号检测网络训练及量化流程

本文提出了一种基于二值量化的ECG信号检测卷积神经网络的算法设计，分为两个部分：网络结构的训练和网络参数的量化。在网络结构的训练部分，根据ECG信号检测分类的任务和二值量化的目标，设计了卷积神经网络的基本模块，并通过调整网络的深度和宽度，寻找最优的网络结构。在网络参数的量化部分，以训练好的全精度网络为基础，采用量化感知训练的方法，对网络进行二值量化。为了保证量化后网络的精度表现，使用了分阶段的二值量化策略，即先对权重进行二值量化，再对激活进行二值量化。最后，将网络中其他参数定点化，得到最终的量化后的ECG信号检测分类卷积神经网络。本文的算法设计流程如图3.1所示。

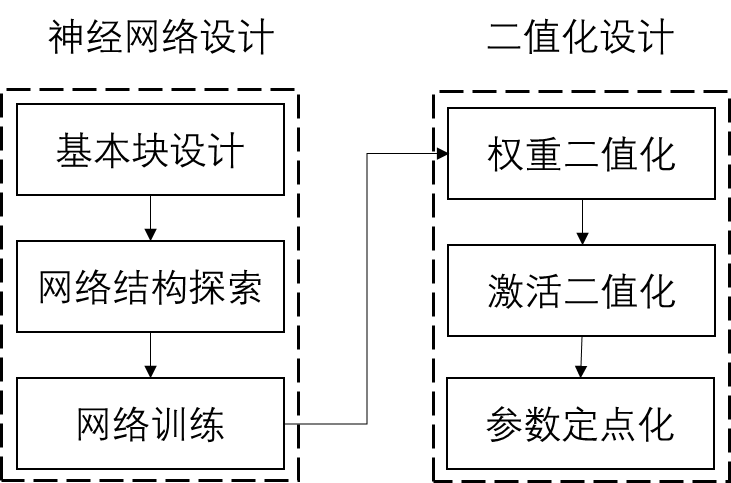


图3.1 ECG检测卷积神经网络与网络二值化方法设计流程图

3.2 数据集预处理

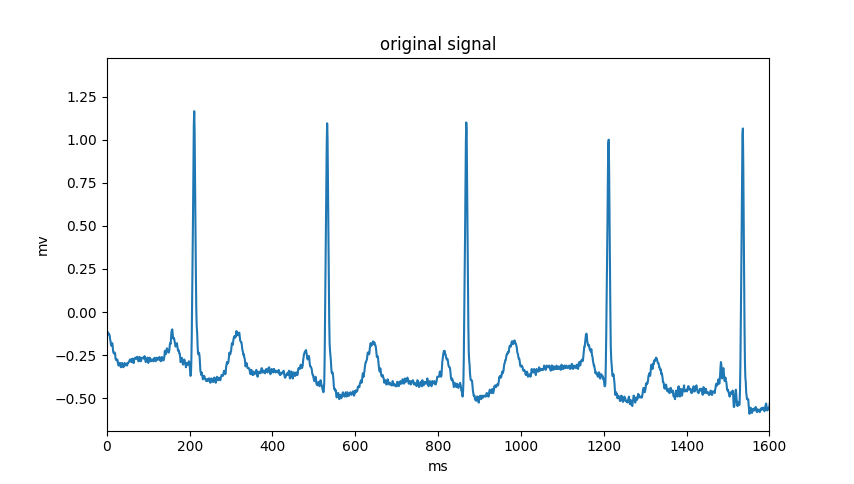
ECG信号是由心脏的电生理活动引发的，每次心脏收缩和舒张都会产生一定的电位变化，这些电位变化可以通过体表的电极和导联来检测和记录。从ECG信号中我们可以看出心律和心拍等信息。心拍是指心脏的每一次收缩和舒张，一个典型的心拍波形包括P波、QRS波群、T波和U波等部分，每个部分对应着不同的心脏结构和功能。而心律则是指多个心拍形成的节奏、规律。

MIT-BIH 是一个用于研究心律失常的心电信号数据集，它由美国麻省理工学院（MIT）和波士顿贝斯以色列医院（BIH）的心律失常实验室提供。它是目前国际上公认的可作为标准的心电数据库之一。MIT-BIH数据集包含了48条双通道动态心电信号记录，每条记录超过了30分钟，共有约110000个心拍。这些记录来自于47名受试者，其中有25名男性和22名女性，年龄在23到89岁之间。这些受试者都有不同类型的心律失常或正常的窦性心律。MIT-BIH数据集对心拍和心律均进行了标注，在本文的研究中主要关注心拍。在MIT-BIH数据集中一共有19种心拍分类，而AAMI标准建议使用五种心拍分类来评价算法的优劣，MIT-BIH的心拍和AAMI标准心拍的对应关系如表3.1所示。

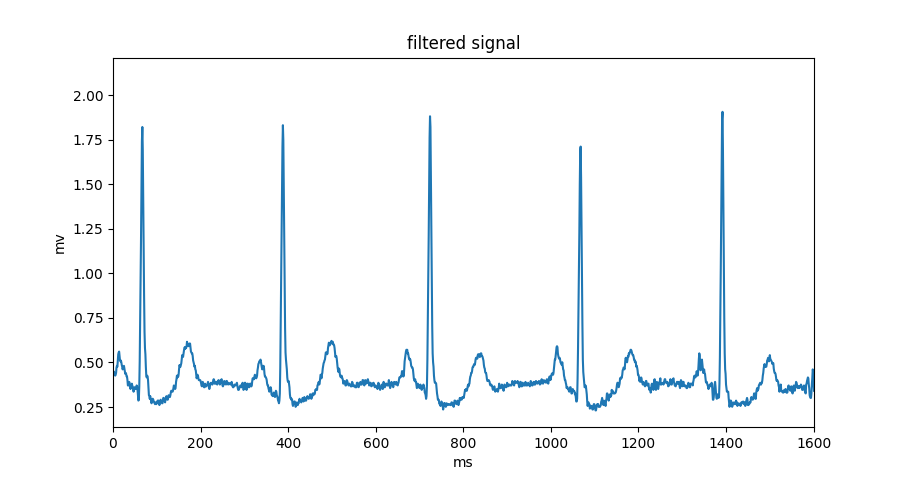
表3.1 MIT-BIH心拍标注和AAMI标准心拍分类对应关系

|  |  |  |
| --- | --- | --- |
| 编号 | MIT-BIH心拍分类 | AAMI标准心拍分类 |
| 1 | normal beat | N |
| 2 | left bundle branch block beat |
| 3 | right bundle branch block beat |
| 4 | atrial escaoe beats |
| 5 | nodal(junctional) escape beat |
| 6 | bundle branch block beat (unspecified) |
| 7 | artrial premature beat | S |
| 8 | aberrated atrial premature beat |
| 9 | nodal(junctional) premature beat |
| 10 | supraventricular premature beat |
| 11 | supraventricular escape beat (atrial or nodal) |
| 12 | premature ventricular contraction | V |
| 13 | ventricular escape beat |
| 14 | R-on-T premature ventricular contraction |
| 15 | fusion of ventricular and normal beat | F |
| 16 | paced beat | Q |
| 17 | fusion of paced and normal beat |
| 18 | unclassified beat |
| 19 | beat not classified during learning |

由于呼吸、电极接触不良等原因会造成心电信号的上下波动，因此需要通过信号处理的手段滤除基线漂移的噪声。本文采用中值滤波的方式来对信号进行去噪，中值滤波使用信号某点邻域内各点值的中位数代替各点值，这样可以在保留心电信号尖峰和细节的同时去除基线漂移。如图3.2所示，中值滤波之后去除了基线漂移的噪声，同时保留了原始的信号细节。



(a)原始心电图信号



（b）滤波后心电图信号

图3.2 中值滤波前后心电图信号对比

MIT-BIH的双通道信号中本文选用modified limb lead II，在滤波后将心电信号按照10秒为一个窗口进行分割，每个窗口内的信号不重叠。最后根据每个窗口内的心拍注释信息，为每个窗口内的心电信号切片赋予相应的类别标签。若10秒窗口内均为正常心拍则视为N类，若含有异常心拍则视为相应异常心拍的类别。最终按照AAMI的心拍分类标准得到的每个类别的数据分布如表3.2所示。

表3.2 数据集样本分布

|  |  |  |
| --- | --- | --- |
| 编号 | 心拍类别 | 样本数量 |
| 1 | N  （Any heartbeat not in the S,V,F or Q classes） | 5186 |
| 2 | S  （Suoraventricular ectopic beat） | 545 |
| 3 | V  （Ventricular ectopic beat） | 1890 |
| 4 | F  （Fusion beat） | 100 |
| 5 | Q  （Unknown beat） | 19 |

由于数据集的样本数较少，本文采用8:2的方式对数据集进行划分，即百分之八十的数据集用作训练集，剩下的百分之二十的数据集用作测试集。从表3.2可以看出，由于某些心拍不常见的原因，AAMI标准下MIT-BIH五个分类的样本数量严重不均衡。对于神经网络实现分类任务而言，训练集样本不均衡会导致模型对多数类别的样本过拟合的同时，也会导致对少数类别样本的欠拟合，从而影响模型的泛化能力和分类性能。为解决样本不均衡的问题，本文采用重采样的方法对训练集进行了处理，通过复制增加了少数类别的样本数以增强网络的泛化性能。同时测试集的分布保持不变，依然和原始数据集的分布一致，可以客观的反应网络的分类性能。

3.3 ECG信号检测神经网络设计

心电图信号和图像都是由多个基本元素组成的复杂数据，它们之间存在一定的相似性。心电图信号由P波、QRS波、T波等不同的波形组成，每个波形反映了心脏的不同电生理过程。图像由像素点组成，每个像素点反映了图像的不同颜色或灰度信息。它们都具有一定的规律性和周期性，如心电图信号每个周期包含一个完整的心跳过程，图像在水平和垂直方向上有一定的重复性或对称性。它们都可以通过卷积操作来提取局部特征，并通过池化操作来降低维度和增加不变性。因此，我们可以将心电图信号看作是一种特殊的图像，或者说是一种只有一个通道的灰度图像。基于心电图和一般图像的相似性，以及卷积神经网络在图像识别方面的优异性能，使用卷积神经网络算法实现ECG信号的分类是一种有效的方法。

3.3.1 卷积神经网络结构探索

算法输入的ECG信号是单通道的一维数据，单个输入的维度为(1,3600)。针对一维的输入，在算法中选用了一维卷积层来实现直接对一维信号的处理。若使用二维卷积层，则需要对输入信号进行升维，导致计算量的增大。参考典型的卷积神经网络，本算法的卷积神经网络包含一维卷积层、一维最大池化层以及激活层。同时由于最终要将网络二值量化之后部署到硬件上实现，如2.4中所述，还需要在网络中引入批处理归一化层来将卷积的输出进行归一化。因此网络的基本块如图3.3所示，输入经过一维卷积之后进行最大池化之后，先归一化处理再对其进行激活。

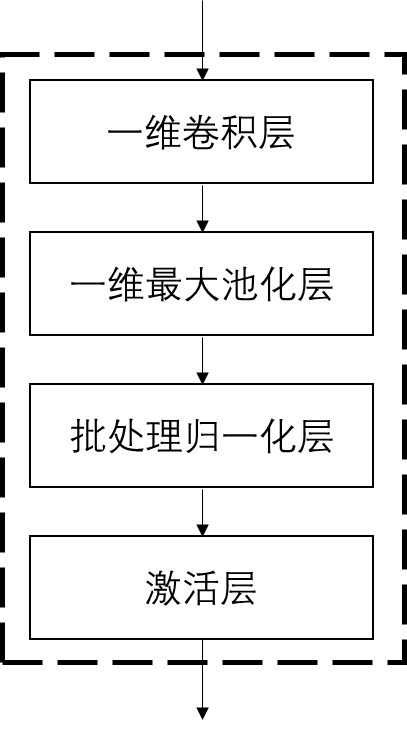


图3.3 卷积神经网络基本块

ECG检测神经网络的整体网络结构如图3.4所示。使用多个基本块来实现局部特征的提取并将其组合成全局特征，然后使用全局平均池化层对特征进行降维，最后使用全连接层对特征进行分类，并输入Softmax层来生成每类的概率。

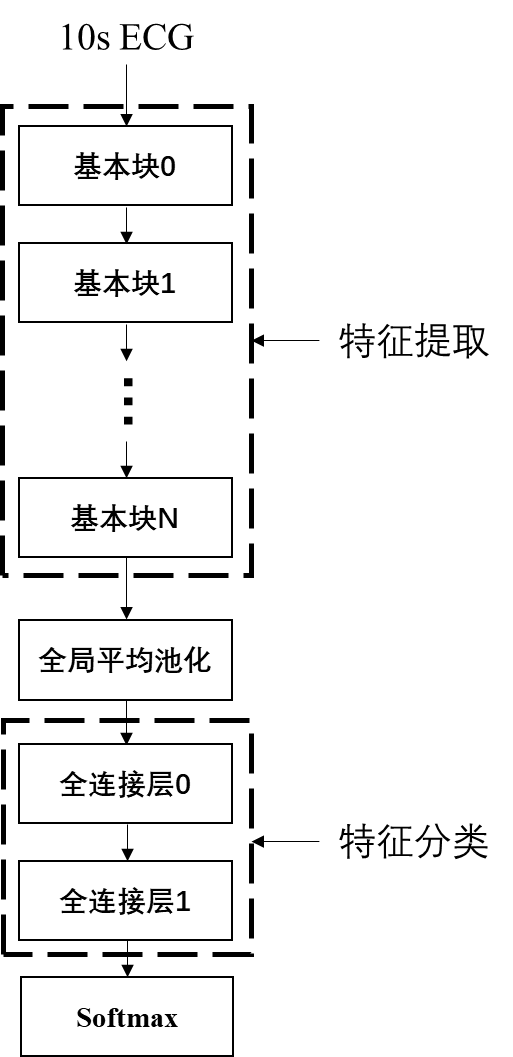


图3.4 ECG检测分类网络基本结构

对于网络的超参数选择方面，基于经验手工的选择了一些超参数，并探索了超参数选择对于网络准确率和二值化后准确率的影响。卷积核大小是重要的超参数，由于输入是一个长达3600的一维数据，且其中每个心拍的长度均较长，所以需要第一层的卷积核有较大的长度来捕捉全局的信息，从而能够提取到更为抽象的特征信息。而之后的卷积核长度则均选择为3，它是常用的卷积核大小，可以捕捉局部信息，减少模型的参数量和计算量，提升泛化性。在池化层方面选择池化的步长和大小均为2，因其不存在交叠，可以更方便的硬件实现。

对于网络中卷积层的宽度和深度，我们进行了超参数的探索。卷积层的宽度，也就是卷积核的数量，即输出通道数。随着卷积层宽度的增加，网络可以拥有更佳的学习能力和表达能力，但是相应会导致参数量和计算量的增加，同时也带来了过拟合的风险。本文设计的网络结构和VGG-16类似[49]，每层的宽度随着深度的增加逐渐变宽。在网络宽度探索中我们统一使用了五个基本块组成的卷积神经网络，同时其分类部分均使用一层全局平均池化层和两层全连接层的组合。如图3.5所示，随着网络宽度的增加，网络的分类性能不断提升，但是当第一层的宽度达到128个通道的时候准确率反而有所下降，这可能是由于过拟合而导致的泛化性不佳的问题。同时随着网络的通道数的增加，模型占用的内存空间也迅速增大，但是由于本网络最终还要进行二值化处理，可以缓解内存占用的问题。同时保留一定的宽度有利于网络在量化之后还有相当的表达度，因此最终本网络选择了使用64个输出通道作为第一层的宽度，并且随后基层逐步加宽。

图3.5 网络宽度对网络模型大小和识别准确率的影响

同时本文也对网络的深度，即基本块的个数进行了探索。所有网络的第一层通道数皆为64，随着基本块数量的增加网络的宽度也逐渐增加。结果如图3.6所示，可以看到随着基本块数量的增多网络的准确率也不断增加，然后当基本块数量超过5之后准确率反而会降低。这可能是由于网络深度过深导致梯度消失或梯度爆炸导致训练困难，反而引起了网络的退化（degration）。或是因为本算法使用的数据集是一个小样本数量的数据集，过深的网络导致了过拟合的问题。经过探索之后最终选取的基本块数量为5。

图3.6 网络深度对网络模型大小和识别准确率的影响

经过网络结构的探索，最终选择了第一层输出通道数为64，由五个基本块和两个全连接层组成的网络结构。此时网络的识别准确率为96.45%，网络模型参数占用的内存大小为786.3KB。网络的具体结构如表3.3所示，其中卷积核的三个维度分别为输入通道数、输出通道数和卷积核长度。

表3.3 网络结构及参数分布

|  |  |  |  |
| --- | --- | --- | --- |
| 网络块 | 卷积核大小 | 池化大小 | 参数量 |
| 基本块1 | 1🞨64🞨80 | 2🞨2 | 5120 |
| 基本块2 | 64🞨64🞨3 | 2🞨2 | 12288 |
| 基本块3 | 64🞨128🞨3 | 2🞨2 | 24576 |
| 基本块4 | 128🞨128🞨3 | 2🞨2 | 49152 |
| 基本块5 | 128🞨256🞨3 | Avg pol | 98304 |
| 全连接层1 | 256🞨64 | - | 16384 |
| 全连接层2 | 64🞨5 | - | 320 |

3.3.2 网络性能分析对比

为了评估本文提出的网络模型在心拍分类任务上的性能，我们在测试集上进行了实验，并绘制了混淆矩阵，如图3.7所示。混淆矩阵的横轴表示真实的心拍类别，按照AAMI标准分为N,S,V,F,Q五类。混淆矩阵的纵轴表示网络模型预测的心拍类别。因此，混淆矩阵的对角线元素表示正确分类的样本数。浮点参数的原始网络模型在测试集上的准确率为96.45%。由1.3.2介绍得知，准确率（Accuracy）可以反映出网络的整体精度，而精确度（Presicion）可以更好的反应出多分类任务中网络在某一类上的性能。精确度的计算方法如公式(3.1)所示。

从图中可以看出，本文的网络模型在N类和V类上达到了较高的精确度（97%以上）。然而，在F类和Q类上，尽管我们采用了样本均衡的策略来增加这两类样本的数量，但是由于类别间的不平衡程度过高，原始数据集的缺乏导致这两类的分类精度较低。不过，在实际应用中，F类和Q类样本出现的频率也很低，而且本文的网络模型在SVEB标准下，即常见的心拍三分类下也表现出了较好的准确率（96.64%）。因此，本文提出的网络模型具有良好的心拍分类能力。

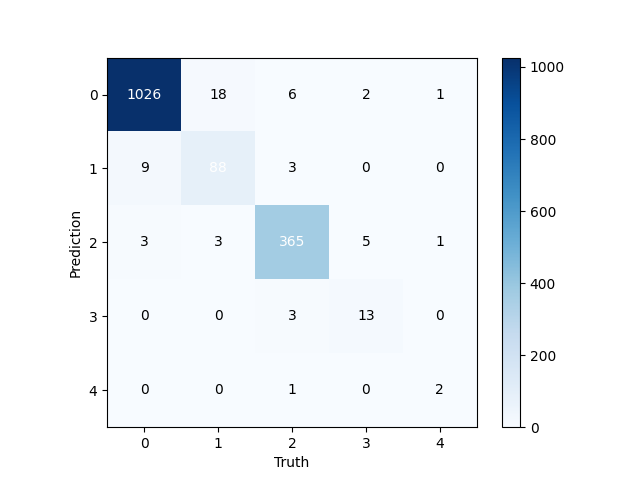


图3.7 混淆矩阵

为了验证本文提出的全精度网络模型在心拍分类任务上的优势，我们将其与其他一些近年来基于AAMI标准的五分类卷积神经网络模型进行了比较，结果如表3.4所示。由于目前针对长时间ECG信号的分类方法较少，我们主要选取了一些先对ECG信号进行心拍分割，然后再对每个心拍进行分类的方法作为对照组。从表中可以看出，本文提出的全精度网络模型在准确率和参数量上与最先进的心拍分割加分类方法相当。同时，本文提出的方法具有以下优点：

（1）它可以直接输入长时间ECG信号，无需进行心拍分割，从而节省了计算资源；

（2）它只需要对整个信号进行一次推理，而不是对每个心拍进行单独推理，从而降低了推理次数和延迟。

表3.4 本文提出的网络与其他文献工作的性能对比

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 指标 | 本项目 | ITBCS2019[26] | TBME2017[51] | ACCESS2019[50] |
| VEB  分类准确率 | 98.39% | 99.18% | 98.92% | 99.99% |
| SVEB  分类准确率 | 96.64% | 98.66% | 99.37% | 99.93% |
| 五分类准确率 | 96.45% | 98.38% | 94.81% | - |
| 参数量 | 206144 | 198037 | - | 16960 |
| 输入信号 | 长时信号 | 分割后的心拍 | | |

综上所述，与现有的心拍分类方法相比，本文的模型具有以下优势：它可以直接处理长时间的ECG信号，而无需进行切片或分段，它只需要一次推理就可以处理长时间的ECG信号。虽然本文的模型在精度上略低于一些最先进的方法，且有较大的参数量，但其可以端到端的对长时信号进行检测分类。

3.4 ECG信号检测神经网络二值量化设计

3.4.1 二值量化训练流程设计

为了实现神经网络的压缩和加速，我们需要将训练好的全精度卷积神经网络转换为二值网络，即将权重和激活值量化为1位。然而，由于二值化会导致大量的信息损失，并且量化函数（sign函数）不可微分或导数为零，所以不能直接使用梯度下降法来优化参数。因此需要使用直通估计器(Straight Through Estimator ，STE) 来近似梯度，以克服量化函数不可微分的问题。此外，我们采用了一种分阶段的量化训练方法，即先对权重进行量化训练，再对激活值进行量化训练。这样可以使网络逐渐适应量化误差，并加快训练收敛。

本文的网络量化训练流程如图3.8所示。该流程分为两个阶段：权重二值化和激活值二值化。在权重二值化阶段，首先对全精度的网络模型参数为基础，进行量化感知训练，通过给所有卷积层和全连接层的权重添加二值量化函数，实现前向推理过程中的权重二值化，并利用直通估计器在反向传播过程中模拟梯度更新，以学习到二值量化误差的噪声分布。此外，由于第一层输入的原始心电图信号具有丰富的信息量，因此不对其进行二值量化，而是在权重的量化感知训练中将其量化到8bit。在激活值二值化阶段，基于权重二值化后的网络模型参数，进一步对网络中所有激活值进行量化感知训练，实现网络除第一层输入外的全二值化，并得到本文所需的最终量化后网络。

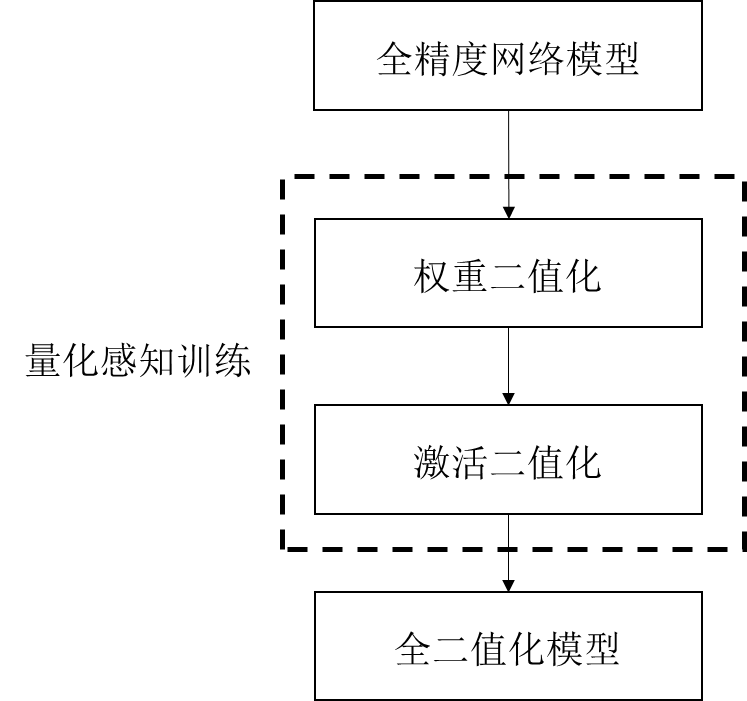


图3.8 二值量化训练流程

为了缓解直接全参数二值化带来的网络性能劣化，本文采用了分阶段的二值量化设计，使得网络可以逐渐适应二值量化引入的噪声。如果一开始就对所有参数进行二值化，网络会受到大量噪声的干扰，导致训练难以收敛。分阶段二值化可以保留全精度网络中学习到的分类能力，而直接全二值化训练会使得网络精度在训练后快速下降，无法利用之前学习到的特征

3.4.2 二值量化训练过程介绍

下面将对二值量化训练的流程进行详细的介绍：

（1）权重二值化

权重二值化的量化感知训练是以全精度的网络模型为基础，在模型中添加量化层，将第一层输入的ECG信号量化成8bit整数，以及将所有卷积层和全连接层的权重量化成1bit。

首先介绍第一层输入量化成8bit的流程。由2.3的介绍可以得知，量化是确定一个缩放系数和一个零点，然后将浮点数映射到整型数上。量化的函数如公式(3.2)所示，其中为量化前的浮点数，为量化后的整型数，为零点，为缩放系数。

一般而言量化的缩放系数是根据待量化浮点数的最大最小值确定的。我们可以通过选取一个校准数据集，以此数据集的浮点数范围作为参考的最大最小值来确定缩放系数。然而在量化感知训练中还可以通过将缩放系数最为可训练的参数在训练中学习获得。由于取整的函数不可导，无法计算缩放系数的梯度，所以使用直通估计器对梯度进行近似，如公式(3.3)所示。

利用直通估计器，就可以计算出缩放系数的梯度了，如公式(3.4)所示。利用此方式，缩放系数也变成了可训练的参数，有利于网络训练的收敛，增加泛化性。

同样的，零点也可以作为训练的参数。但是考虑到增加零点偏移值产生的额外计算负担，以及第一层输入分布大致均匀的原因，在本文对第一层输入的量化中采用均匀对称量化，即不采用偏移的零点。最终将第一层的输入映射到[-127,128]的整型数范围内。

然后是对权重的二值化，公式(2.8)是传统的二值量化方法。本文参考改进的XNOR-Net[41]，引入了缩放系数来提升网络的精度。假设未量化的权重为，量化后的二值化权重为，他们均是矩阵向量。二范数可以衡量两个矩阵之间的差值，因此使用原始权重和二值化权重的差值的二范数L作为衡量误差的标准。由于量化后二值权重是原始值的符号位，只有两种取值，因此想要优化量化的误差只能够通过引入其他参数的方式。假设引入缩放系数，那么就需要缩放系数能够使得量化的误差最小。将二范数展开之后如公式(3.5)所示。由于是实数矩阵，因此可以用实数表示。而矩阵均是二值，所以等于矩阵元素的数量。

有了误差的表达式之后，因为误差是关于缩放系数的一元二次函数，就可以求得使误差最小的缩放系数。如公式(3.6)所示。缩放系数等于原始权重矩阵的L1范数的均值。

在权重量化过程中，本文对每个输出通道即每个卷积核均求出一个缩放系数。在前向推导过程中，首先使用符号函数将原始权重量化为二值权重，然后计算出其缩放系数，接下来再在卷积运算中用权重和输入进行卷积运算。为了硬件实现的便利性，同时由于本文网络带有批处理归一化层，已经对激活值分布进行约束的原因，本文网络的卷积运算并没有偏置值。因此只需要将乘累加的最后结果乘上缩放系数即可输入以后的计算流程。

在训练中，首先将原始的模型进行修改，按照上述的方式为第一层输入和所有的权重添加量化层。然后载入之前训练完毕的浮点函数，以此基础进行继续训练，并保留最优的模型以供下个阶段继续量化。训练的过程如图3.9所示，可见经过较少轮次的训练，模型的准确率达到了96.10%,接近全精度模型的精度水平。通过缩放因子的引入，提升了模型的表达度，因此在权重二值化的情况下网络几乎不损失精度。

图3.9 网络权重二值化训练过程图

（2）激活二值化

得到权重二值化的模型之后，再在此模型的基础上继续进行激活值的二值化。和权重的量化相同，激活值的二值化将激活量化为1或者-1。为了提高精度和权重的二值量化一样也为激活值引入了缩放系数，此缩放系数是用本层输入的L1范数的均值计算所得，其粒度是按层的，而不是像权重一样有着按通道的细粒度。由于卷积计算的线性特点以及本网络的卷积不含偏置值，可以使用二值的权重和激活输入进行卷积计算完毕之后，再统一乘上缩放系数。

除了第一层外，每层输入的激活值均要经过二值化处理之后再进行卷积计算。原始的网络模型中的激活函数是Relu，Relu函数将小于0的所有输入值均置为0。这在全精度的训练和二值化权重的训练中均不影响，但是在激活二值化的情况下，Relu函数的输出导致激活值二值量化后只有1值。同时符号函数本身也具有非线性，因此要在模型中去除激活函数Relu，使用符号函数同时发挥激活函数和量化函数的功能。

在去除网络的激活函数Relu，为每层激活值添加二值量化函数之后，即可载入之前权重二值化的最优模型参数，以此为基础训练完全二值化的网络模型。在训练过程中需要采用更低的学习率，避免参数更新过大破坏已经学习好的特征。训练的具体过程如图3.10所示，首先以0.01的学习率在权重二值化的模型基础上继续训练，如图3.10(a)所示。然后取最佳模型，以此为断点使用更低的学习率0.001继续训练从而获得最佳的模型参数，如图3.10(b)所示。

(a)

(b)

图3.10 激活二值化训练流程

（3）参数定点化

在激活二值化训练完毕之后，第一层输入被定点化为8bit整型，其他层输入被量化到二值，同时所有的权重均是量化到二值。然而，量化过程中为了更高的精度引入的权重按通道的缩放系数，以及激活按层的缩放系数，还有批处理归一化层的相关参数均是浮点数。考虑到硬件上部署浮点计算所需的硬件资源较大，因此需要对这些参数进行定点化的处理。

由之前介绍的批处理归一化层公式(2.1)可以看出，其相当于一个线性函数，包含一个权重和一个偏置，如公式（3.7）所示。

如公式(3.8)所表示的一维卷积，第a个输出通道第b个输出点的计算是由M个输入通道的长度为K的卷积核卷积计算而来。其中，和均是量化为二值后的权重和激活。由于其是线性的乘累加运算，所以用于提升精度的浮点缩放系数可以被提取出来。

计算完毕的累加值在经过最大池化层之后，就开始进行批处理归一化层的处理。经过(3.7)的化简，批处理归一化层也是一个线性函数，以此二值量化的缩放系数可以进一步被提取，融合进批处理归一化层的参数，如公式(3.9)所示。

经过公式(3.9)的计算，可以得出经过激活函数之前的中间值。本文使用二值量化函数来代替激活函数来发挥作用，而二值量化函数就是符号函数。符号函数将大于等于零的数输出1，其他输入均输出-1，也就是说只要将中间值和零值相比较就可以得到最终的量化后输出。我们得到的中间值是一个整型数乘上一个浮点系数后加上一个浮点偏置，若我们将其假设等于零，那么就可以得到一个用于整型数比较的浮点阈值。进一步的，我们可以将阈值取整，得到一个整型数的阈值，如公式(3.10)所示。

经过系数的融合，我们得到了一个整型数的阈值，此阈值的粒度是输出通道相关的，也就是说每个输出通道都有一个对应的整型数阈值。我们使用二值的权重和输入计算完毕的输出仅需要和此整型阈值相比较即可得到量化后的激活值。通过这种方法，最终消除了网络中的浮点参数。至此，网络中所有的参数以及在前向推导过程中的中间值均实现了定点化。

3.4.3 量化后网络性能分析对比

经过权重和激活的分阶段二值化之后，得到最优的二值量化后的网络模型，将其在验证集上进行测试。经过测试，二值化后的网络在AAMI五分类任务上取得了92.83%的分类准确度（Accuracy），其混淆矩阵如图3.11所示。二值化之后的模型在N和V这两大类上取得了超过93%的精确度（Precision）。



图3.11 混淆矩阵

将本文的二值化模型和最先进的量化后ECG信号分类网络（AAMI标准）进行对比如表3.5所示。可以看出相较于[52]中的二分类网络，本文的网络有着和其相仿的二分类性能。相较于**[27]中的五分类网络，虽然**准确率**下降了6.11%，但是内存占用量只有其59.3%。同时由于本网络的计算均是二值运算，虽然计算量次数较多，但是在硬件上消耗的计算资源却远远小于[27]。**

表3.5 本文提出的网络与其他文献工作的性能对比

|  |  |  |  |
| --- | --- | --- | --- |
| 指标 | 本项目 | ISCAS2021[52] | ISNE2021**[27]** |
| VEB  分类准确率 | 96.06% | 96.8% | - |
| SVEB  分类准确率 | 93.29% | - | - |
| 五分类准确率 | 92.83% | - | 98.94% |
| 参数位宽 | 1bit | 1bit | 16bit |
| 内存占用量（KB） | 28.1 | - | 47.4 |
| 计算量  （MOPS） | 1.96 | - | 1.03 |

通过分阶段的量化设计和引入按层粒度的激活和权重的缩放系数，尽量减少了网络精度的损失，因此本算法有着较好的精度表现。二值量化带来了更低的计算量和内存占用量。同时，利用卷积层的线性特点，将缩放系数和BN系数相融合，使得用于提升精度的缩放系数在部署到硬件上时无需额外的硬件支持。

3.5 本章小结

本章首先介绍了本文采用的ECG信号数据集及其预处理策略。然后介绍了本文设计的卷积神经网络基本模块，并通过设计空间探索确定了最终的网络结构。本文提出的网络结构可以直接实现长时的心律检测，相比于其他ECG检测卷积神经网络，在保持相似的精度和参数量的情况下，具有更高的执行效率。最后介绍了本文对卷积神经网络进行二值量化和定点化的过程及其策略，通过分阶段二值量化和缩放系数的引入，保证了一定的精度，同时减少了内存占用，便于部署到硬件上。

第四章 基于BNN的ECG信号检测神经网络加速器的设计

基于第三章训练得到的二值化ECG信号检测卷积神经网络，本章将设计其专用的硬件加速器。本章将对加速器的整体结构和数据流进行介绍，并详细描述加速器的计算单元及加速器各个模块的设计，同时对硬件功能进行验证。

4.1 加速器整体结构设计

由于二值量化将计算的输入参数转换为二值，因此可以使用简化的硬件来执行卷积运算。例如卷积的乘累加运算可以通过XNOR和Bitcount组合来实现。同时，二值量化也减少了参数访存的带宽需求，因此在相同的资源占用下，二值量化的硬件可以堆叠更多的并行计算单元。由于二值量化降低了卷积运算的硬件资源占用，算法中其他非二值化的部分占用的硬件资源比例会增多，因此需要精心设计非二值化运算部分来充分发挥二值量化的优势。

加速器的整体框架如图4.1所示。根据二值量化的特点，设计了专用的二值化卷积计算单元，并通过添加较少的额外硬件支持了算法中的非二值部分运算。由于计算单元的简化，所以整体计算阵列得以堆叠更多并行的计算单元，从而实现多维度的数据并行，并在数据并行中充分挖掘数据的可复用性。通过计算单元的简化和数据复用，从而使得加速器得到了更高的吞吐率和能效比。

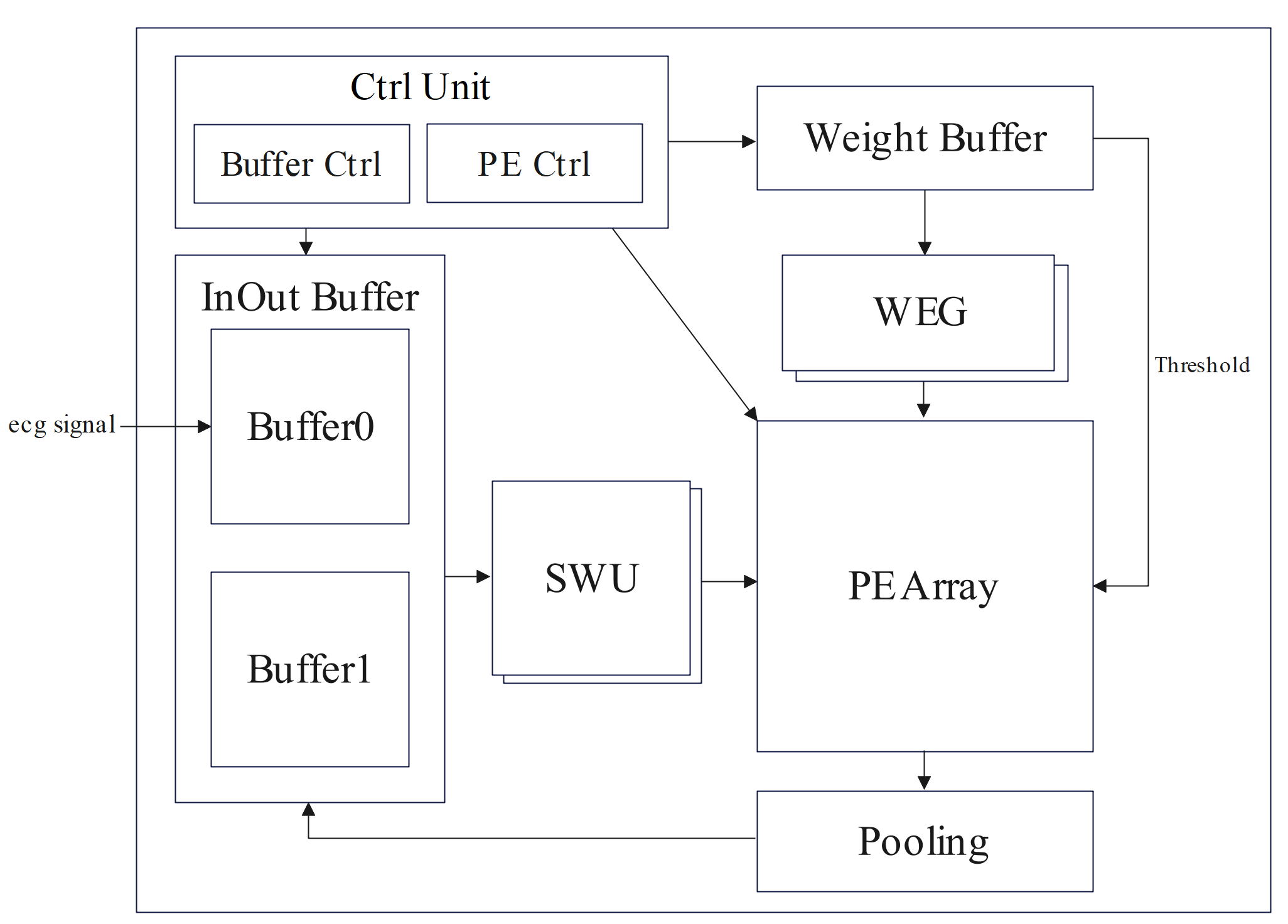


图4.1 加速器整体框架图

加速器各个模块的功能如表4.1所示。Contrl Unit是加速器的控制单元，它负责整个加速器工作状态的控制，以及缓存数据的调度。存储单元包含InOut Buffer、Weight Buffer。InOut Buffer模块是负责激活值存储的模块，其负责向SWU输出待计算的激活值，以及写回经过计算单元PE计算完毕和Pooling单元池化之后的结果。Weight Buffer则负责向WEG输入待计算的权重，以及为PE提供当前输出通道对应的阈值。SWU和WEG是缓存待计算参数的单元，通过对缓存单元中的参数复用实现更少的数据读写次数。SWU单元负责向PE阵列广播激活值输入，其将从InOut Buffer读取的输入按照卷积滑动的方式进行滑动，然后将选中的激活输入到PE单元之中。而WEG则向每个PE单元分发权重。计算单元主要包括PE阵列和Pooling单元，分别执行核心的卷积运算和激活结果的池化运算，最终将计算结果返回给InOut Buffer。其中PE阵列由多个PE单元组成，它们可以并行的进行计算。

表4.1 硬件加速器主要模块及模块功能说明表

|  |  |
| --- | --- |
| **模块** | **模块说明** |
| **Contrl Unit** | **控制模块，**负责控制整个加速器的工作状态、InOut Buffer和Weight Buffer读写地址的计算以及网络参数的传输。 |
| InOut Buffer | **特征数据存储模块，存放输入的ECG信号和各层计算的中间值。** |
| Weight Buffer | 权重数据存储模块，存储权重数据以及阈值数据。 |
| **SWU** | **激活数据寄存器堆，缓存单批次待计算的特征数据，通过滑动操作对数据进行输出，提高激活输入的数据复用。** |
| **WEG** | **权重数据寄存器堆，缓存计算所需的权重。** |
| **PE Array** | **高并行度卷积计算阵列，执行CNN网络模型中的卷积计算、批处理归一化运算以及激活量化操作。** |
| **Pooling** | 池化模块，执行MaxPooling操作 |

在本加速器的各个模块中，PE Array中的PE单元使用简化的计算单元实现二值化的卷积运算，同时能高效的执行算法中的非二值化部分。由于计算单元的简化，PE Array可以支持输入通道、输出通道和输出特征图多维度的并行。为PE Array缓存计算所需参数的SWU和WEG则通过和PE单元的配合，充分的进行数据复用。加速器的具体数据复用将在下一小节详细介绍，同时根据数据复用系数进行数学建模，探索加速器的最优尺寸，即各个维度的最佳并行数。

4.2 加速器数据流分析及设计空间探索

4.2.1 加速器数据流分析

典型的加速器的数据流分为层折叠的加速器和流水线设计的加速器。卷积神经网络是由多个卷积层和全连接层等模块组成的，每层的输入依赖于前一层输出的计算结果。流水线设计的加速器将卷积神经网络中每一层的计算设计为流水线的一级，通过流水线的设计方法使得卷积网络中每一层的并行运算，极大的提高了硬件的吞吐率。除了吞吐率的优势外，流水线的设计还可以根据每层卷积的维度定制设计，避免硬件的空置。但是流水线设计也会带来硬件资源需求大，设计复杂度高的缺点，例如每一层的输出均需要经过特别设计来提前输入到下一层来降低延迟。层折叠的数据流是指使用统一的计算单元来进行卷积神经网络所有层的计算，其优点在于硬件资源需求少，适合本文的应用场景。其缺点在于可能带来硬件闲置率高、延迟大的问题。本文设计的网络能够通过一次推理完成10s心率信号的处理，应用场景对延迟不敏感。因此层折叠的数据流是对本文的应用场景而言最优的数据流设计方法。

（1）加速器整体数据流

本加速器采用层折叠的数据流设计方法，其整体数据流即是完成一次前向推导的数据流。根据本文采用的网络模型，加速器需要完成5层卷积网络的计算和2层全连接的计算，同时第一层的输入位宽为8bit，和其余层输入的位宽不同。接下来将用二值输入的卷积层运算来介绍加速器的典型数据流。

InOut Buffer中两个Buffer进行乒乓操作，轮流负责读写操作。例如SWU向Buffer0读取激活输入，而Pooling单元的输出则写入到Buffer1中。假设切片系数TM、TN和TR，分别对应输出通道、输入通道和输出特征图的并行数。KL是卷积核的长度，每轮计算开始前，SWU向InOut Buffer中读取计算所需的长为（TR+KL-1）的TN个输入通道的激活值，然后分TR次向32个PE单元广播相同的输入激活值。Weight Buffer则向TM个PE单元配置TM个不同输出通道的TN个输入通道的权重，在此轮计算中保持不变。待SWU中TR次广播的输入计算完毕之后，再向InOut Buffer中读取另外TN个输入通道的激活值，循环上述的计算直到所有输入通道的激活被计算完毕。最后PE单元将累加完成的TR个输出特征值经过阈值比较器，输出到Pooling单元中。在Pooling单元中经过比较之后，将最大值再写入回InOut Buffer中。

综上所述，加速器的整体数据流是从InOut Buffer输出，经过PE阵列、Pooling单元处理后再返回InOut Buffer的循环。

（2）加速器数据复用

对卷积循环的切片是为了实现硬件上对卷积运算进行并行运算。在可并行的卷积运算中，有相当一部分输入参数是可复用的。本加速器是固定输出特征图，通过改变权重和激活最终计算出结果的。对于所有的输出而言均只写入一次缓存，而权重和激活则需要反复读取。本文通过数据流的设计，实现了权重和激活的复用，降低了输入和激活的访存次数。在不存在任何复用的情况下，访存次数如公式4.1所示。

在公式(4.1)中，NUMimap、NUMomap和NUMweight分别代表输入特征图、输出特征图和权重的参数数量。而其对应的系数、和则表示复用的系数，复用系数不同会导致访存次数的不同。未复用时复用系数如公式(4.2)所示。其中KS和KL表示卷积核的步长和卷积核大小，OL表示输出特征图的长度。

对于激活的复用主要包括两方面。一是用于计算相邻的输出特征图的输入特征图之间存在交叠的部分，复用交叠的数据可以减少访存的次数。在SWU模块中利用卷积滑动的特性，每次广播的激活值输入均复用了上一次广播的激活值输入的一部分。另一个方面则是通过激活的广播达成了不同卷积核之间输入激活值的复用。通过这两种方式的复用，激活值的复用系数如公式(4.3)所示。相较于原始的复用系数，本文使用的复用方法大约使得访存次数减少了（OC/TM）倍。这也进一步说并输出通道的并行可以更大的影响激活的访存效率。

对于权重的复用也主要来源于SWU的滑动产生的收益。SWU广播的相邻输入特征图共享相同的权重数据，因此一次配置的权重数据可以被复用多次。新的复用系数如公式(4.4)所示，相较于不复用的情况下访存次数减少了TR倍。这也说明输出特征图的并行对权重参数的复用有直接益处。

综上所述，考虑本文的实际应用场景，本加速器采用层折叠的整体数据流。在卷积运算中，将输出值固定，通过变换激活和权重的方式进行卷积计算。为了减少访存次数，对输入的激活值和权重均进行了复用，从而降低了访存次数，使得硬件更加高效。

4.2.1 设计空间探索

在确定加速器的整体数据流和数据复用方式之后，接下来需要确定加速器的尺寸，即输出通道、输入通道和输出特征图的并行系数TM、TN和TR。由于数据的复用方式已经确定，所以可以知晓不同系数下的访存次数，对加速器硬件资源占用和能效进行建模，从而确定最优的切片系数。

一维卷积神经网络的具体计算流程如伪代码4.1所示，由外到内分别是输出特征图、输出通道、输入通道以及卷积核的循环。

|  |  |
| --- | --- |
| **伪代码4.1：** |  |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |

为了发挥硬件并行性的特点，在硬件部署中需要对卷积进行切片处理，在切片内实现运算的并行。一维的卷积运算在输出特征图、输出通道和输入通道这三个维度上均可以进行切片来实现并行，如伪代码4.2所示。在一次的并行运算过程中，每次同时计算TM个输出通道的TR个输出点，并且每轮计算完成TN个输入通道的累加。

|  |  |
| --- | --- |
| **伪代码4.2：** |  |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |

由上可知，一维卷积运算有输出通道、输入通道和输出特征图三种并行方式，需要对并行的超参数TM、TN和TR进行选择。针对超参数的选择，本文从能效比方面进行了设计空间的探索，能效比如公式(4.5)所示：

能效比是硬件卷积算力操作数和耗能的比值，对于固定的网络算法而言，其算力的值是固定的，如公式(4.6)所示。由于硬件切片和网络各层参数不一定匹配，可能导致某些部分硬件在计算中闲置。利用率的降低会使得硬件不能保持峰值的吞吐率，但是其有效的计算量确实不变的。因此无论何种硬件方式其一次前向推理的总算力是一个固定的值。

对于能耗方面的预估，主要包括访存功耗和计算功耗，如公式(4.7)所示。其中Ebuffer、Eoperation分别表示一次访存的功耗和一次计算的功耗，这在确定的硬件平台上是可知的固定值。例如在Xlinx-7平台上缓存但比特数据的访存功耗为0.45pJ[53]。Operations则表示硬件执行计算的总操作数，由于硬件中存在闲置的问题，总操作数会大于等于计算量。同样的，不同的切片方式使得数据复用的效率不同，其访存数也存在差别。

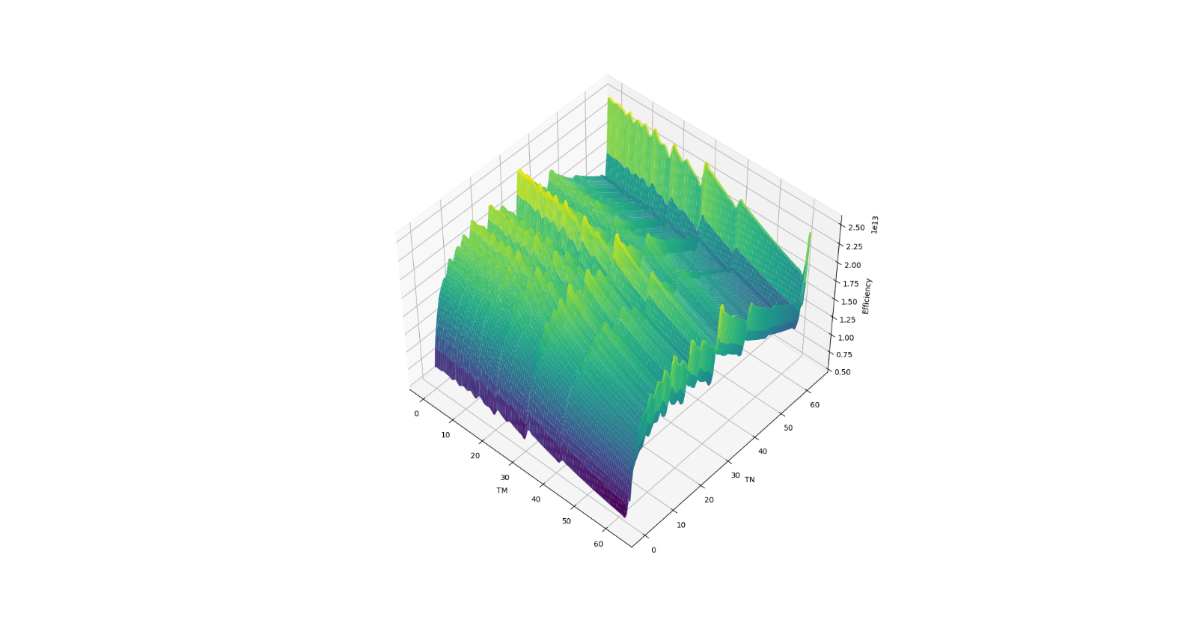
为了计算耗能，需要确定不同切片超参数下产生的总操作数以及访存数。首先计算激活值输入的访存数，由于卷积运算滑动的特性，不同输出点之间的输入激活值可以复用。因此输出特征图的切片参数TR越大，输入激活值复用的比例就越高。除此之外，不同输出通道之间可以复用激活值，因此激活的复用主要和TM和TR有关。激活值访存次数的具体计算如公式(4.8)所示，其中ceil函数表示小数的向上取整。

对于权重的访存数量如公式(4.9)所示，权重的复用主要和TR有关，在滑动变换输入的时候，权重是可复用的固定值。

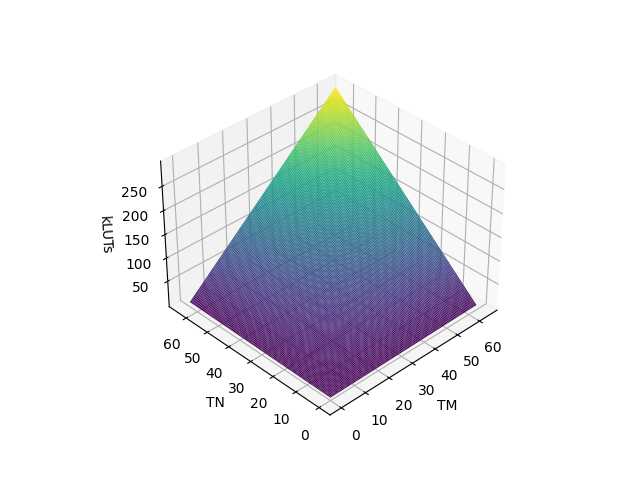
最后是输出的访存次数，如公式(4.10)所示。输出的访存次数是固定的，仅仅和硬件的利用率相关，若输出维度和硬件尺寸不匹配会导致输出访存的浪费。

至此我们获得了运算过程中的所有三种操作的访存次数，然后根据位宽即可求出访存的功耗。在确定了功耗和性能的计算方法之后，就可以开始依据能效比的设计空间探索。由第三章可知，除第一层外，本文采用的输入通道数的范围是[64,128],输出通道数范围是[64,256]。在网络训练过程中考虑到硬件部署的方便，所以输入通道数和输出通道数均是64的倍数。为了避免资源的浪费，以及考虑到小型FPGA平台上有限的资源数，所以取空间探索中TM和TN的范围均为[1,64]。对于输出TR而言，本文采用的网络的输出特征图长度最短为18，因此TR的范围是[1,18]。

为了计算方便，我们固定TR的值，然后对TM和TN两个进行设计空间探索，得到在不同硬件尺寸下能效比如图4.2(a)所示。可以看出随着TM和TN的增大，权重和激活值的复用不断增多，能效比不断提升。但是由于利用率空置等原因，在某些和网络尺寸不能整除的硬件尺寸上，能效比产生了下降，最终产生了波浪形上升的情况。除了能效比之外，TM和TN的增加会导致硬件规模的指数级增加，如图4.2(b)所示，在FPGA上也会导致布局困难、静态功耗增加等问题。综合考虑能效比和硬件资源占用等方面，最终采用的输出通道并行数为32，输入通道并行数为2。对于输出通道的超参数TR，随着TR的增加可以有效增加权重和激活值的复用，但是也会导致输入带宽和中间值缓存空间占用方面的压力，最终选取的TR值为12。



(a)



(b)

图4.2 不同硬件尺寸下的能效比和硬件资源占用

4.3 加速器二值化计算单元设计

本文所使用的卷积神经网络包括卷积层、池化层、激活层以及批处理归一化层。为此本文设计了一个统一的PE（Processing Element）单元用于执行卷积、激活量化以及批处理归一化运算。PE单元的整体结构如图4.3所示，包括有不同位宽输入的乘累加计算单元、中间值累加单元以及批处理归一化的计算单元。接下来将对其进行详细的介绍。

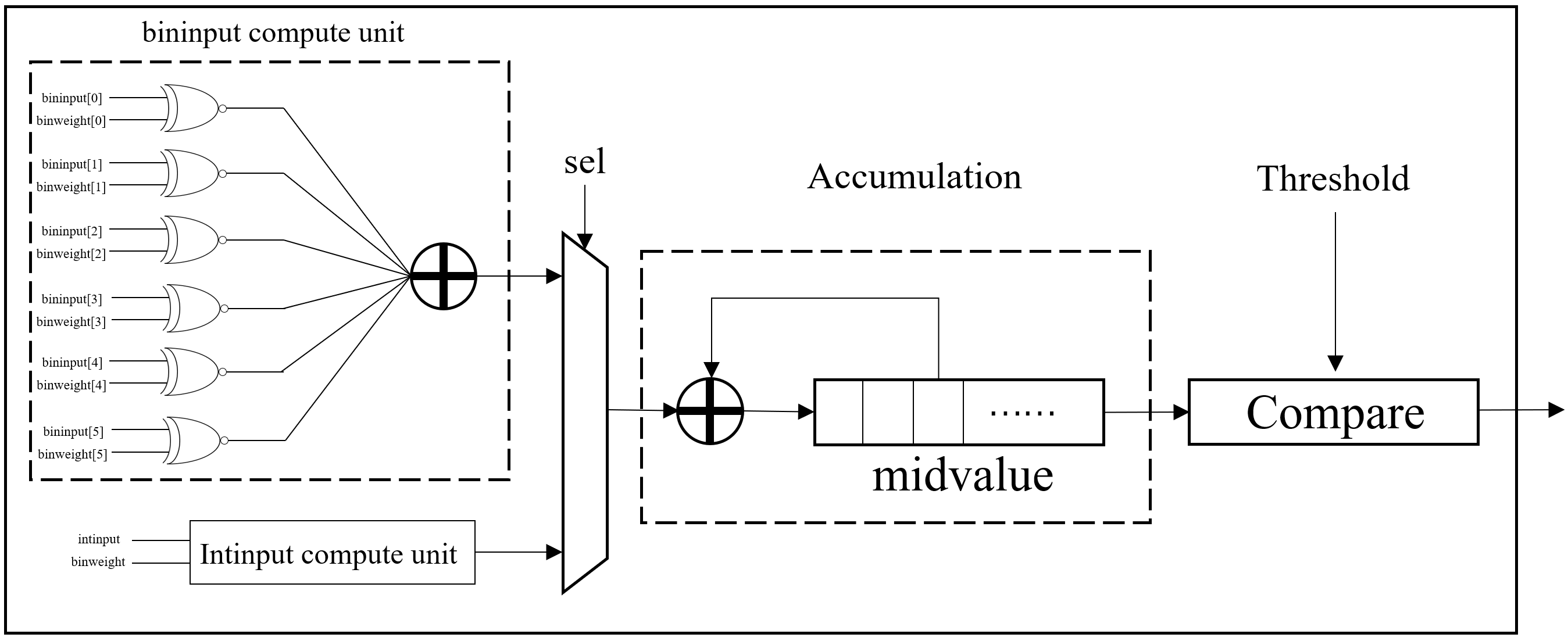


图4.3 加速器PE单元示意图

（1）二值化乘累加计算单元

本设计充分发挥二值化特性，使用同或门和查找表实现了二值化卷积运算的乘累加单元，同时添加了较少的额外硬件使其支持多比特激活输入。

对于1bit激活和1bit权重的乘累加运算，所有的乘法运算可以等效于同或的布尔逻辑运算。因此PE单元中使用多个同或门来执行二值激活的乘法，并将其结果累加得到中间值。对于计算结果的累加，只需要一个六输入LUT即可实现。整个二值激活的计算单元对硬件资源的占用极小，发挥出了二值量化的优势。在二值量化中，本文使用符号函数将权重和激活量化为二值，根据其符号分别将浮点参数映射到1和-1，如表4.2所示。

表4.2 模型参数和硬件表示对比

|  |  |  |  |
| --- | --- | --- | --- |
| 模型参数 | 硬件表示 | 真实结果 | 硬件结果 |
| 1🞨1 | 1🞨1 | 1 | 1 |
| 1🞨-1 | 1🞨0 | -1 | 0 |
| -1🞨1 | 0🞨1 | -1 | 0 |
| -1🞨-1 | 0🞨0 | 1 | 1 |

可见，使用0值代替-1进行计算会使得所有计算结果本该为-1的值被计算为了0。因此为了得到正确的计算结果，只需要将目前的乘累加之和减去本该为-1值的数量，就可以得到真实的计算结果，如公式(4.11)所示。其中Y表示使用0代替-1得到的乘累加的结果，N则表示参与计算的乘法数目。对于本文采用结构而言，只需要将乘累加的结果左移一位再减去6即可得到真实的结果。

二值化激活输入的乘累加单元如图4.4所示，由六个同或门、查找表和加法器组成。

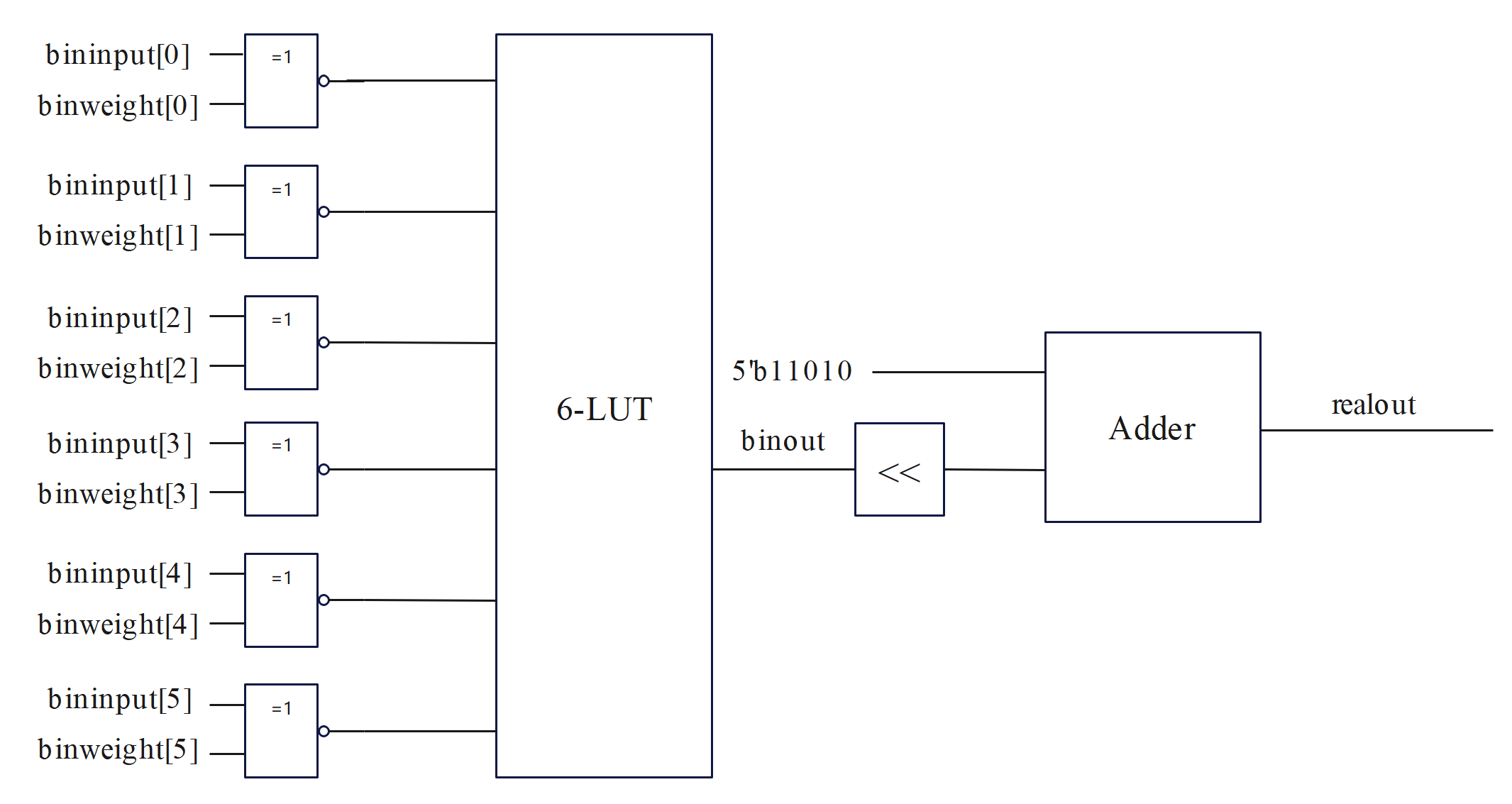


图4.4 二值化激活乘累加单元示意图

对于第一层的8bit整型输入，其权重仍然是1bit。若权重为0则将激活值取反，若权重为1则激活值不变，然后再将激活值相加即可得到卷积的中间值结果。此外，对于全连接层的输入，其可以直接使用二值激活的计算单元进行计算，因为全连接层本质也是全二值输入的乘累加。

本设计使用同或门实现二值激活和权重的乘法，然后使用一个LUT实现6个乘法结果的累加，最后通过一个加法器实现乘累加结果的还原。相比于传统的多比特激活函数和权重矩阵的卷积运算，本设计节省了大量的乘法器和加法器资源，提高了硬件效率。此外，对于8位激活输入的卷积运算，本设计只需根据权重矩阵对输入进行逻辑运算后求和，无需使用乘法器。

（2）中间值累加单元

无论是8bit的激活还是1bit的激活值，其之后的累加、激活和批处理归一化操作是相同的，因此整型激活的运算和二值激活的运算才可以融合到同一个PE单元之中。在二值激活计算单元和整型激活计算单元之后，控制模块根据当前计算的层数来选择乘累加中间值结果进行后续的累加。累加模块由中间值缓存的寄存器和一个累加单元组成。中间值缓存的寄存器个数由输出特征图的分片系数TR决定，每轮计算中会对TM个输出通道、TN个输入通道的一部分输入特征图进行计算，最终会输出TR个中间值缓存结果。为了避免数据的反复读写，中间值在累加模块中累加完毕之后在进行输出。因此需要对TM个输出通道所有输入通道的TR个输出中间值计算完毕之后再进行下一批输出通道的计算。所以对每个由计算单元输出的中间值，累加单元会根据当前计算输出特征图的位置，选择出寄存器中缓存的对应待累加中间值，然后将其和刚输入的中间值累加之后，再缓存回寄存器中。这个累加操作会重复ceil（IC/TN）次，最终的累加结果就是最终的卷积输出值。

中间值累加单元实现了中间值缓存和累加的功能，避免了中间值的反复读写。同时通过缓存实现了输出特征图的并行，使得SWU可以通过滑动的方式实现数据的复用。此外，中间值累加单元还可以实现全局平均池化的功能，将在4.4.4中介绍。

（3）批处理归一化单元

在3.4.2中介绍过，由于激活二值量化的特性，激活函数Relu被融合进了批处理归一化函数。除此之外，在硬件部署前向推导中，如公式(3.10)所示，激活和权重的缩放系数和批处理归一化层的系数均被融合成一个阈值比较函数。因此批处理归一化单元含有一个寄存器缓存此输出通道的阈值，由累加单元输出的累加结果和阈值进行比较，大于等于阈值输出1，其余则输出0值。

批处理归一化单元将系数缩放、归一化、激活和量化等操作集成在一起，通过软硬件的协同设计，仅用一个比较器便完成了复杂的数据处理流程，极大地节约了加速器的硬件开销，有效地实现了算法中的非二值化运算部分。

综上所述，本文设计的加速器PE单元能够完成卷积神经网络的核心计算功能。支持包括二值激活输入的卷积、整型激活输入的卷积以及二值化全连接层输入的计算。使用同或代替乘法，降低了硬件资源的占用。融合系数的阈值比较单元将缩放系数、BN层和激活的二值量化用一个简单的数值比较器实现。同时通过额外添加简单的整型计算单元来支持8bit输入的计算。中间值累加单元实现了输出特征图的并行同时，还可以执行全局平均池化的功能。基于以上的设计，实现了高效的加速器计算单元。

4.4 加速器各模块设计及验证

本小节对加速器整体模块进行了设计并详细介绍，并对加速器进行行为级仿真以验证功能正确性。由于加速器的信号众多，这里只挑选重要模块的仿真波形进行说明。

4.4.1 主控模块设计

加速器的主控模块Contrl Unit主要包括计算控制模块和存储控制模块。计算控制模块主要负责控制加速器的计算状态，而存储控制模块则负责整体数据流的流动。

（1）计算控制模块

本文的计算控制模块由主状态机和从状态机组成。主状态机负责控制加速器的整个计算流程，从状态机负责调度具体的计算任务。计算控制模块的示意图如图4.5所示。计算控制模块内含有主状态机和从状态机，共同控制参数缓存模块向PE阵列发送数据。

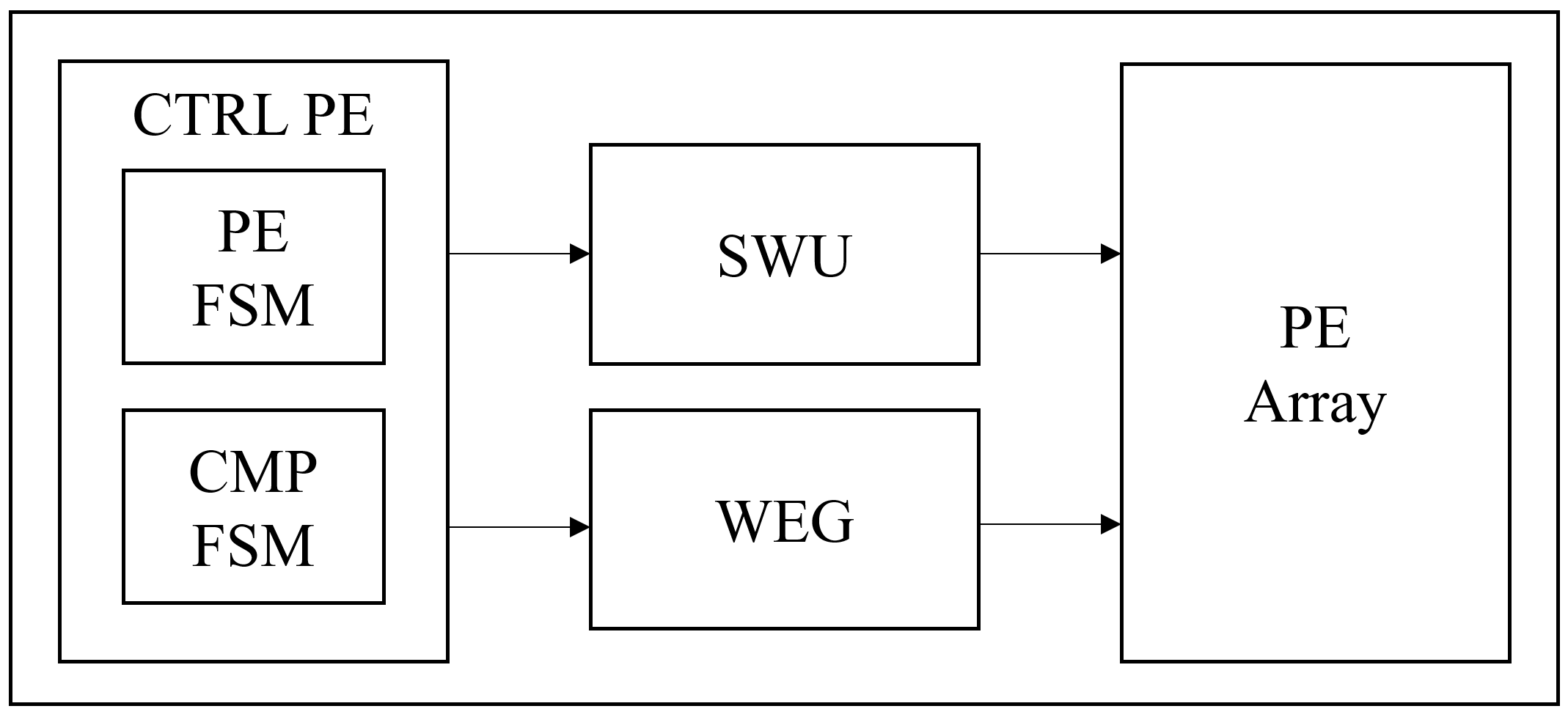


图4.5 计算控制模块示意图

图4.6显示了主状态机的状态跳转逻辑，各状态的具体含义如表4.3所示。当外部ECG信号写入BRAM后，主状态机收到START信号，开始执行前向推导的计算流程。由于卷积神经网络包含整形输入卷积层、二值输入卷积层、全局平均池化层和全连接层，它们需要不同的计算策略，所以主状态机会根据当前的前向推导进度，切换到不同的状态。首先，主状态机进入INT\_CMP状态，执行整型输入的卷积计算。当第一层计算完成后，进入BIN\_CMP状态，执行二值输入的卷积计算。当3层二值输入卷积计算完成后，主状态机进入DEN\_CMP状态，执行全连接层的计算。当2层全连接层计算完成后，进入DONE状态，输出计算结果。输出结果后，主状态机返回IDLE状态，等待新的ECG信号写入。

表4.3 加速器主状态机各状态功能描述表

|  |  |
| --- | --- |
| **状态** | **状态描述** |
| IDLE | **空闲状态,等待ECG写入** |
| INT\_CMP | **整型输入的卷积层计算** |
| **BIN\_CMP** | **二值输入的卷积层计算** |
| DEN\_CMP | **全连接层计算** |
| DONE | **输出分类结果** |

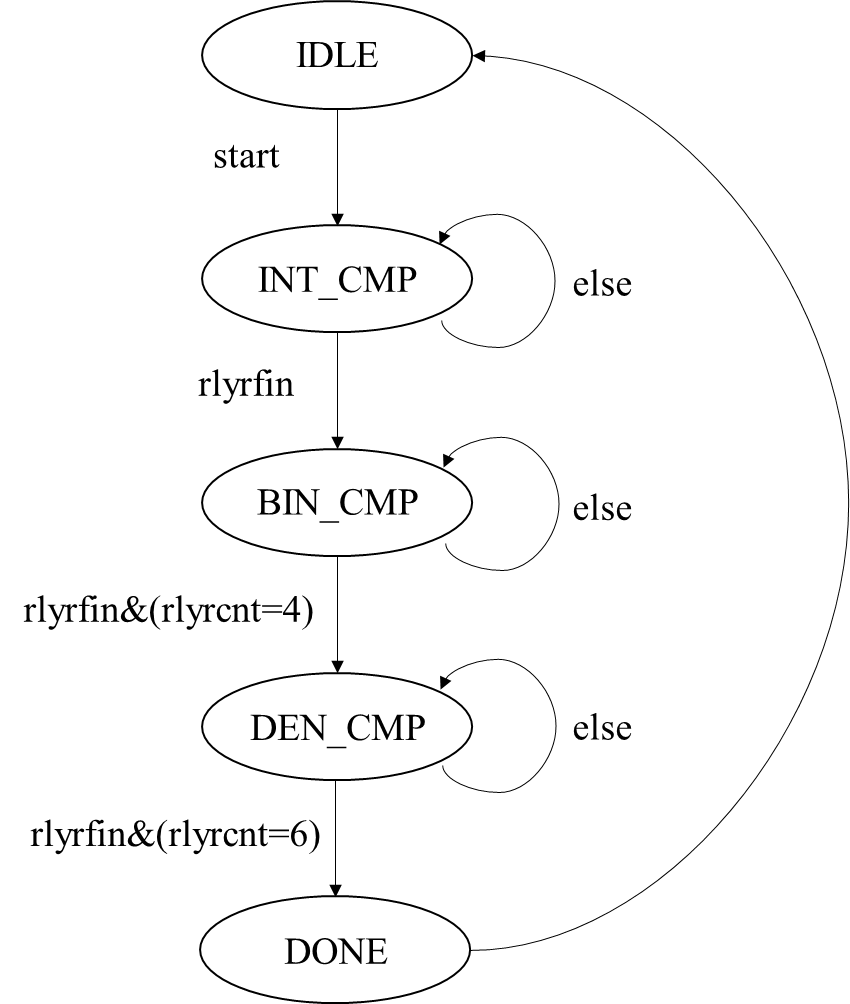


图4.6 计算控制模块主状态机

在主状态机的每个计算状态，即INT\_CMP、BIN\_CMP和DEM\_CMP中，均需要PE阵列进行循环的乘累加计算才能够完成最终的计算结果。在本加速器的设计中，PE阵列进行计算所需要的激活值和权重值均通过双缓冲的方式存储在SWU和WEG中。因此从状态机需要根据主状态机表示的当前计算类型，使用不同的计算策略对SWU和WEG中的参数进行读取。从状态机的各个状态及其功能如表4.4所示。

表4.4 加速器从状态机各状态功能描述表

|  |  |
| --- | --- |
| **状态** | **状态描述** |
| **S0W0IDLE** | **等待SWU0和WEG0中参数写入完毕** |
| **S0W0CMP** | **读取SWU0和WEG0中参数输入PE计算** |
| **S0W1IDLE** | **等待SWU0和WEG1中参数写入完毕** |
| **S0W1CMP** | **读取SWU0和WEG1中参数输入PE计算** |
| **S1W0IDLE** | **等待SWU1和WEG0中参数写入完毕** |
| **S1W0CMP** | **读取SWU1和WEG0中参数输入PE计算** |
| **S1W1IDLE** | **等待SWU1和WEG1中参数写入完毕** |
| **S1W1CMP** | **读取SWU1和WEG1中参数输入PE计算** |

根据主状态机的三种计算状态INT\_CMP、BIN\_CMP和DEN\_CMP，从状态机有三种不同的跳转策略。接下来将分别介绍主状态机处于三种不同状态时从状态机的跳转策略。

（a）INT\_CMP

图4.8展示了从状态机在整型输入计算时的状态转移策略。从状态机有两类状态：空闲状态IDLE和计算状态CMP。在空闲状态IDLE时，状态机等待SWU和WEG的读有效信号。当SWU和WEG都准备好被读取时，状态机进入计算状态CMP。在计算状态CMP时，模块内部开始计数，并且输出读ready信号。同时，PE阵列也开始读取SWU和WEG的参数。INT\_CMP状态下PE阵列每轮计算需要从SWU读取80个8bit的有符号整型数，从WEG读取32组80个1bit的权重。每个周期，SWU输出4个8bit的整型数，WEG输出32个4bit的权重。因此，模块内部需要计数20个周期，然后输出rswufin信号。当rswufin信号为高时，状态机返回空闲状态IDLE，并且取消读ready信号。整体读参数的时序图如图4.7所示。



图4.7 计算控制模块从状态机读参数时序示意图

由于本网络的第一层只有一个输入通道，所以每个输出通道的输出特征图都使用同一个长度为80的卷积核。这意味着当一个输出通道的计算没有完成时，即rolfin信号为低时，下一次计算只需要更换激活值，不需要更换权重。因此，状态机会进入相同SWU、不同WEG的空闲状态。反之，当一个输出通道的计算完成时，即rolfin信号为高时，下一次计算需要同时更换SWU和WEG。因此，状态机会进入不同SWU、不同WEG的空闲状态。

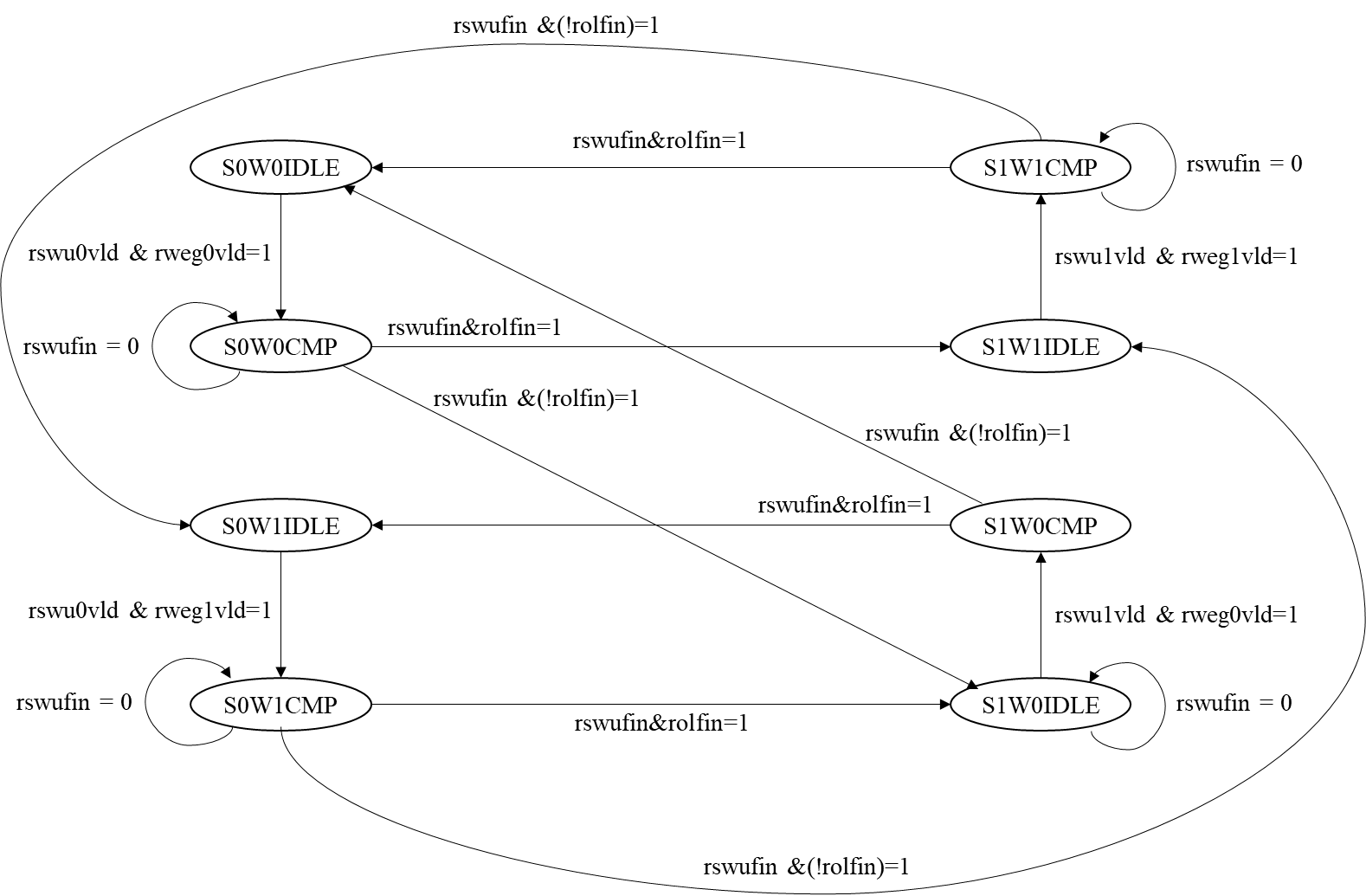


图4.8计算控制模块从状态机在整形输入卷积时的跳转策略

对主状态机INT\_CMP下的计算状态跳转进行仿真验证，如图4.9所示。从图中可以看到其主状态机pe\_cs处于INT\_CMP状态，而从状态机则根据和缓存单元的握手信号进行跳转。仿真结果符合预期的功能，在一条输出特征图计算完毕之前，反复读取同一WEG中的权重，即rweg0rdy反复被拉高而rweg1rdy一直置低。而SWU中的数据则在计算完毕后被切换，即从状态机跳转状态后和不同的SWU进行握手。

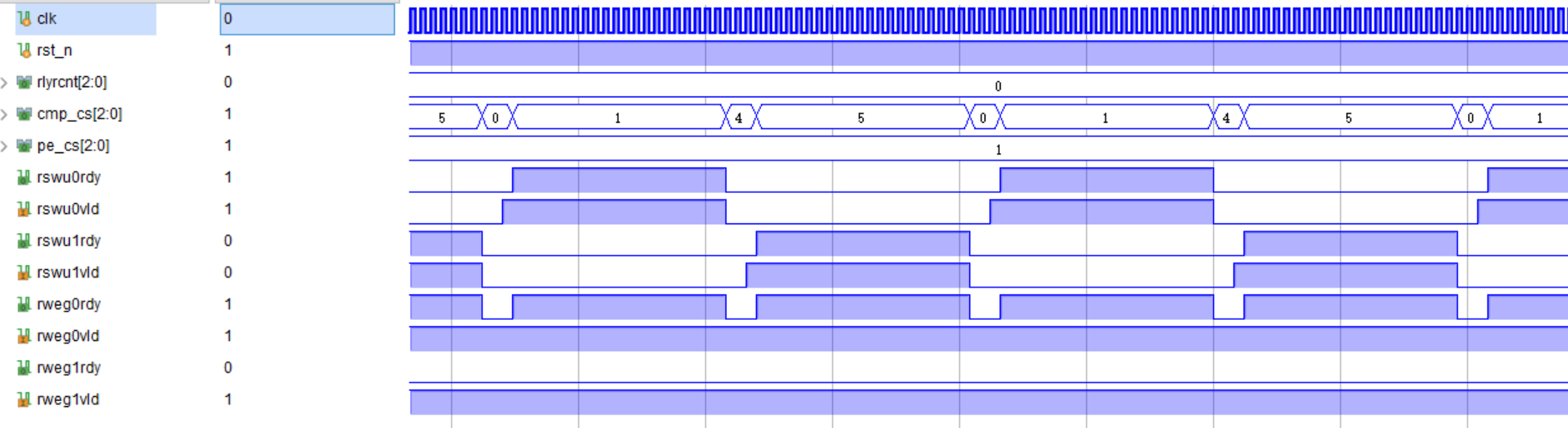


图4.9整型输入时计算控制模块仿真波形图

(b)BIN\_CMP

图4.10展示了在进行二值输入的卷积运算时，计算控制模块的从状态机的跳转策略。和主状态机在INT\_CMP时相同，当SWU和WEG均读有效时，从状态机从空闲状态IDLE跳转到计算状态CMP。在计算状态CMP时，读ready信号被拉高，PE阵列读取SWU和WEG输入的参数。在全二值运算中，SWU使用滑动的方式对WEG中的权重进行复用。在PE阵列的一轮计算中，每个周期SWU输出6bit的二值输入，而WEG则保持相同的6bit权重输出不变。当SWU滑动完毕之后，rswufin被拉高，从状态机跳转到空闲状态IDLE。由于每次写SWU时一次性写入了32个输入通道的激活值，而一轮PE计算的滑动只消耗了两个输入通道的激活值。因此，在SWU中的所有输入通道被消耗完之前，即rswuicfin低位时，从状态机跳转到SWU相同、WEG不同的空闲状态。相反，若rswuicfin是高位，则跳转到SWU和WEG均不同的空闲状态。

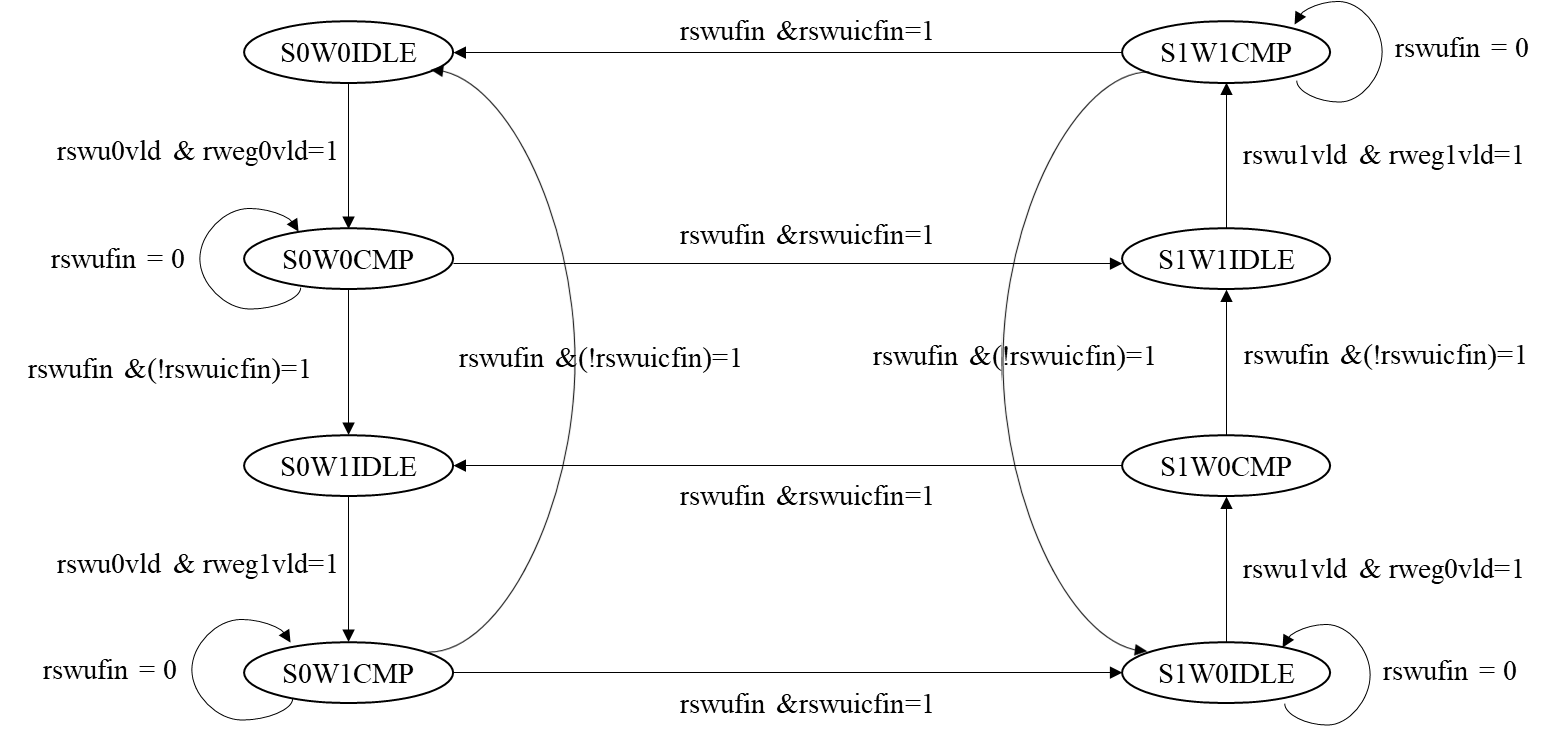


图4.10计算控制模块从状态机在计算二值输入卷积时的跳转策略

计算控制模块主状态机处于BIN\_CMP时候其状态跳转的行为及仿真如图4.11所示。从图中可以看到其主状态机pe\_cs处于BIN\_CMP状态，SWU中的参数在所有输入通道被消耗完之前被反复的读取。而WEG则在一次计算后就进行切换，同时也可以看出，计算消耗参数的周期大于向WEG中写数据所需要的周期，所以PE可以无缝的进行连续的计算。

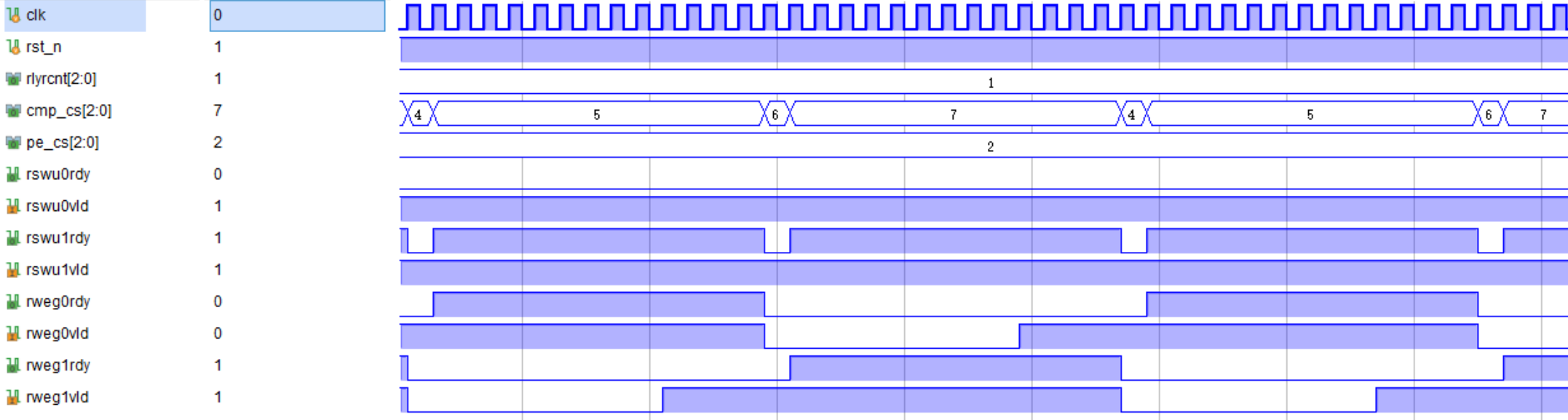


图4.11二值输入时计算控制模块仿真波形图

(c)DEN\_CMP

当主状态机位DEN\_CMP时，从状态机的跳转策略如图4.12所示。在全连接层的计算中，由于输入的数据量很小，两层的输入分别为256bit和64bit。因此仅使用一个SWU即可，而权重WEG则不断进行变换。同样的，当读参数的握手成功之后，从状态机从空闲状态IDLE跳转到计算状态CMP，PE阵列开始读取数据。每轮计算中向PE阵列输入6bit的激活值，以及32个6bit的权重值，一个周期即可完成此轮计算，然后从状态机跳转到SWU相同，WEG不同的空闲状态中。

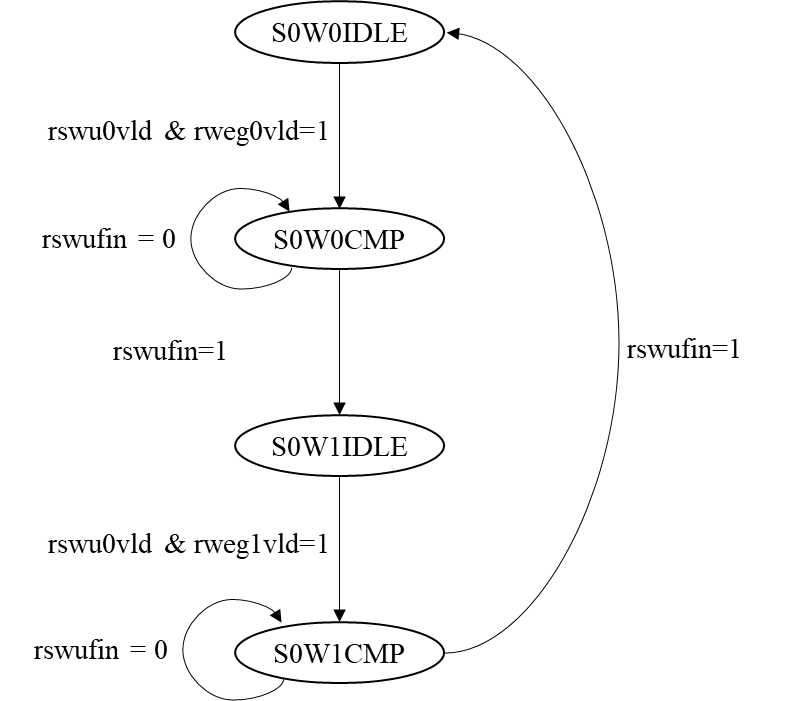


图4.12计算控制模块从状态机在二值输入卷积时的跳转策略

图4.13是全连接层计算时计算控制模块的状态跳转仿真波形图。从图中可以看到其主状态机pe\_cs处于DEN\_CMP状态。由于全连接层的输入数据量较少，可以被单个SWU存储，因此只需要反复从SWU0中读取参数即可。而WEG则在一次计算后需要切换，同时每次计算只需要一个周期即可完成，PE需要等待WEG参数的写入完成才能进行计算。

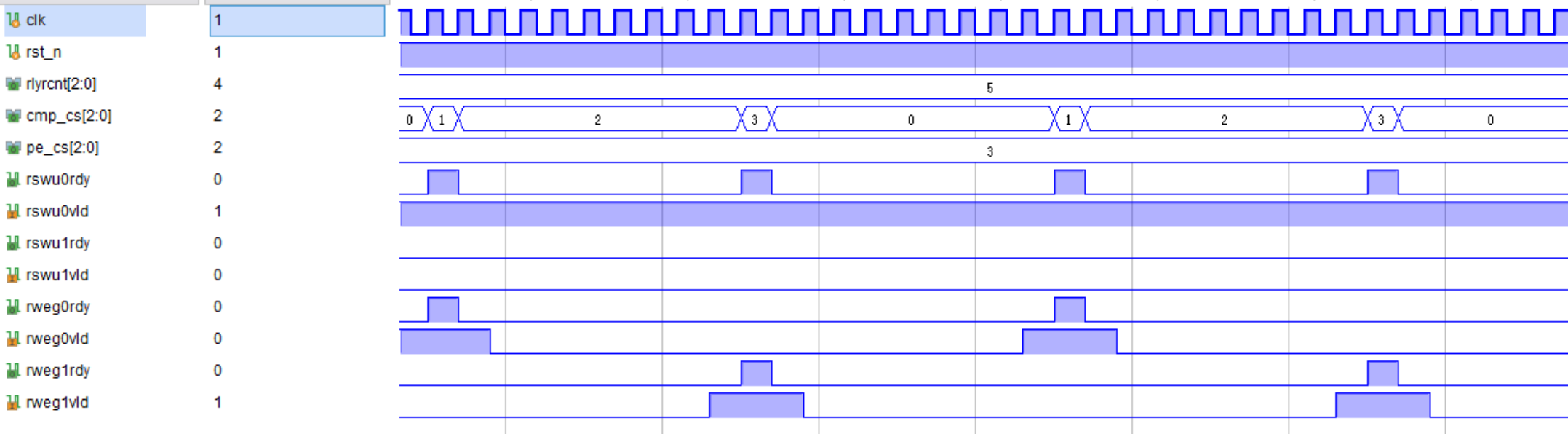


图4.13全连接层计算控制模块仿真波形图

（2）存储控制模块

存储控制模块负责权重和输入的写入调度，同时根据当前的参数层数生成读参数的地址。当缓存模块准备好接收数据时，存储控制模块会从BRAM中读取相应地址的参数，并发送给缓存单元，同时置位写有效信号。当所需参数全部写入后，存储控制模块会复位写有效信号，并更新下一个参数的读地址。存储控制模块的示意图如图4.14所示。

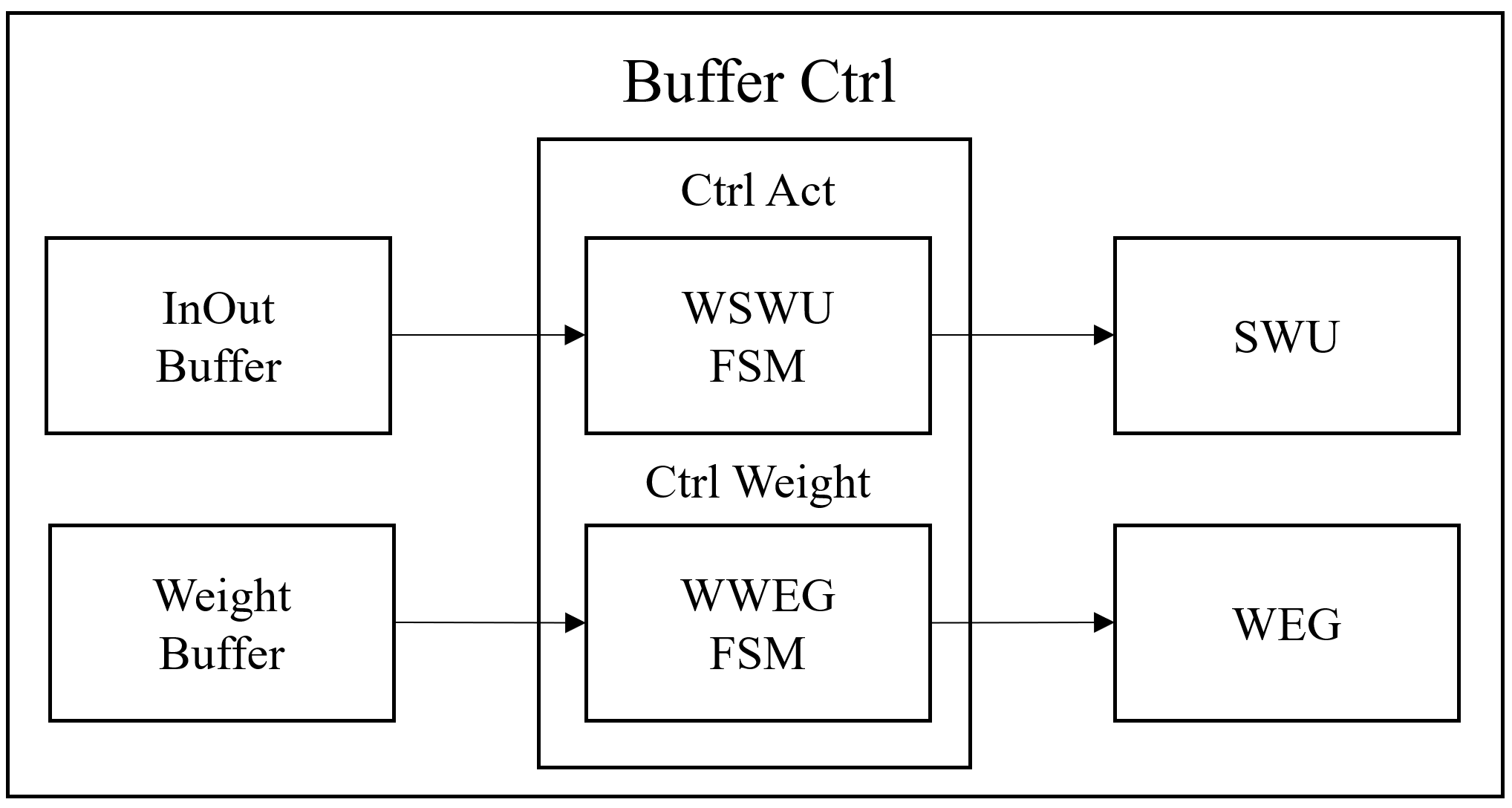


图4.14存储控制单元示意图

为了减少计算阵列等待参数写入的时间，本文的设计采用了双缓冲的方法，即SWU和WEG都有两个缓存单元进行乒乓操作，交替进行读写。当一个缓存单元中的参数用完后，就立即置位写入准备信号。存储控制模块使用两个状态机来控制缓存单元的交替读写操作，如图4.15所示。

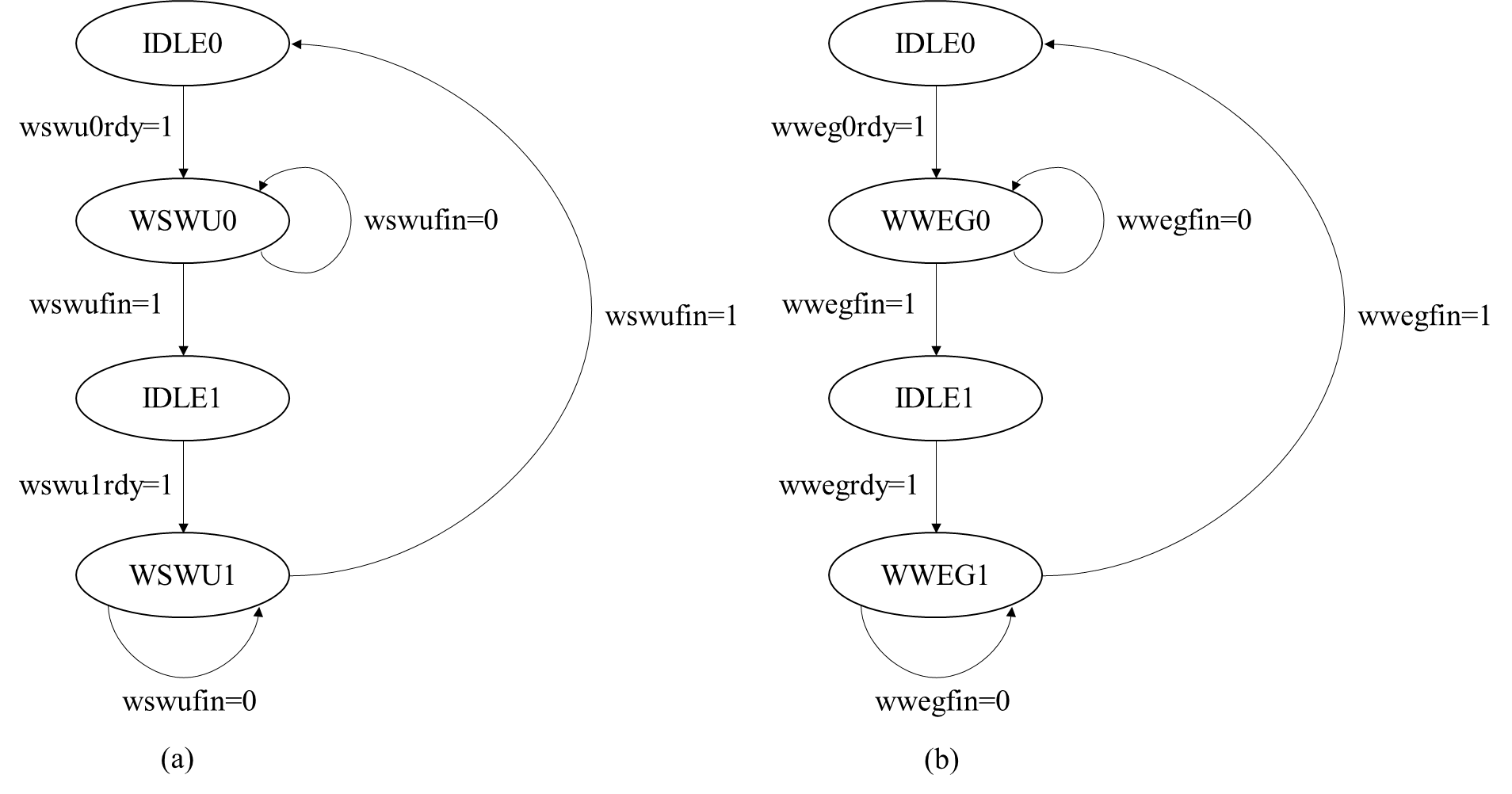


图4.15存储控制模块状态机

存储控制模块向SWU写参数时的仿真波形图如图4.16所示。在第一层中需要向SWU中写入20个32bit的数，由于BRAM的写延时所以需要花费22个周期才能写入完毕。同时计算一个SWU中缓存的参数也只需要20个周期，因此SWU需要无缝的进行写入。



图4.16存储控制模块写激活缓存仿真波形图

存储控制模块向WEG写参数时的仿真波形图如图4.17所示。存储控制模块一次连续向WEG写入6个32bit的权重。由于每个WEG中的权重需要消耗12个周期才能计算完毕，所以写WEG需要等待WEG中的参数被消耗完毕之后才能继续写入。

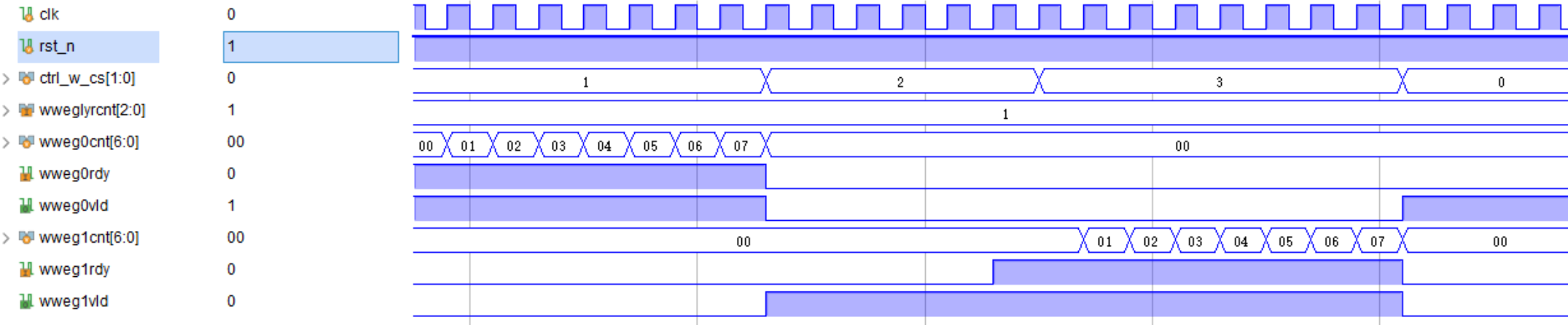


图4.17存储控制模块写权重缓存仿真波形图

4.4.2 缓存模块设计

本加速器的缓存模块分为激活值缓存模块SWU和权重缓存模块WEG，每个模块都有两个缓存单元，交替进行读写操作，实现双缓冲技术，即当一个缓存单元读取参数时，另一个缓存单元写入参数，减少读参数的延迟。缓存模块与存储控制模块和计算控制模块分别进行握手，完成参数的写入和读出。下面分别介绍SWU和WEG。

（1）激活值缓存模块SWU

激活值缓存模块由一组32bit宽的存储单元组成。其深度由算法的卷积核长度以及PE阵列中输出缓存个数共同决定。在本设计中，由于第一层的卷积核长度较大，所以取其值为缓存模块的深度。SWU的示意图如图4.18所示，缓存控制模块从InOut Buffer中取一组32bit的数据写入SWU之中，然后SWU进行滑动，将选中的激活值广播到PE阵列。

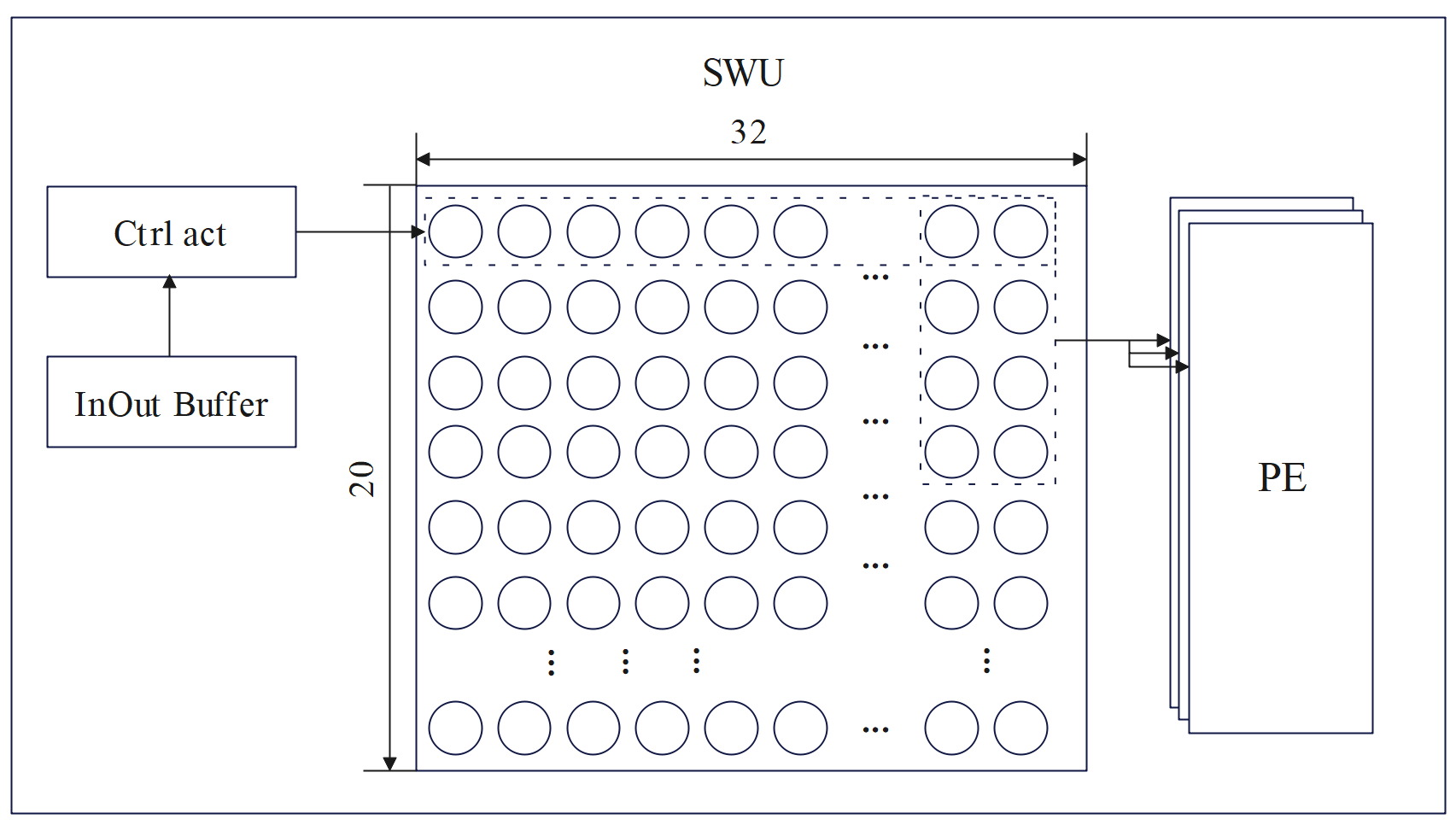


图4.18激活值缓存模块SWU

在存储控制单元从InOut Buffer中读取参数写入SWU时，写入的参数的位宽为32bit。SWU根据存储控制单元提供的写入地址，将写入的参数存储到对应的32bit存储单元之中。存放在SWU中的数据分为二值激活和整型激活，其存放方式与InOut Buffer中相同。对于整型数而言，每一行存储4个8bit的输入，一个SWU中存储80个8bit的输入，即一个完整的卷积核。PE中可以缓存的中间值数量为tmpwid个，即输出特征图的并行数TR。对于二值数而言，每行存放32个不同输入通道的1bit激活值，一共存放(tmpwid+kl-1)行，可供PE阵列滑动读取tmpwid次。

在计算控制单元从SWU中读取参数时，SWU根据计算控制单元的当前计算状态是整型输入计算还是二值输入计算，分别输出整型输入和二值输入。如图4.19(a)所示，当SWU输出二值输入时，每次选取TN列长度为KL的激活进行输出，在本文的设计中即是2列3行，共6bit的输出。如图4.19(b)所示，当SWU输出整型输入时，每次输出一行4个8bit的输出。激活缓存模块的输出是通过广播的方式发送到所有计算单元的，也就是说PE阵列中所有的PE单元共享相同的输入。

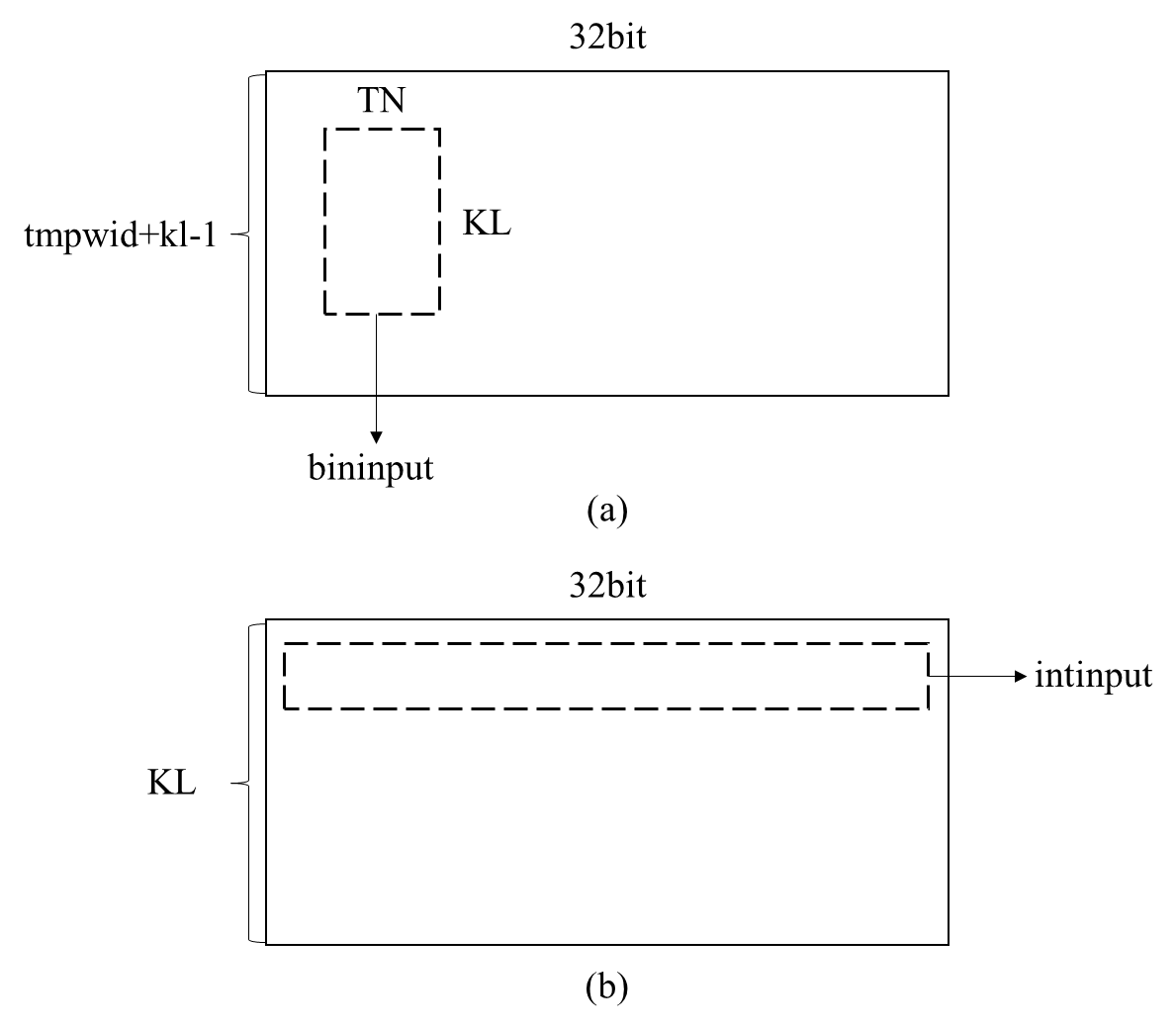


图4.19激活缓存模块SWU输出示意图

激活缓存模块的状态机如图4.20所示。当存储控制模块准备好写入时，且SWU处于写入空闲的状态IDLEW，则跳转到WSWU状态并进行参数的写入。当存储控制模块发出写入完毕信号wswufin之后，跳转到IDLER状态等待计算控制模块读取SWU内缓存的参数，同时拉高读有效的信号。当计算控制模块发送读准备信号rswurdy之后，跳转到RSWU状态进行参数的读取。SWU根据计算控制模块的状态进行计数，当参数读取完毕后拉高rswufin。对于二值的激活值输入而言，由于SWU中存放了多个输入通道的激活值，需要分多轮计算才能消耗完毕。因此在所有激活值消耗完毕即rswuicfin为低时，跳转到IDLER状态等待下一次的读取。否则在其他情况下，SWU中的参数已经被消耗完毕，可以被再一次写入覆盖，即跳转到IDLEW状态等待存储控制模块的写入。

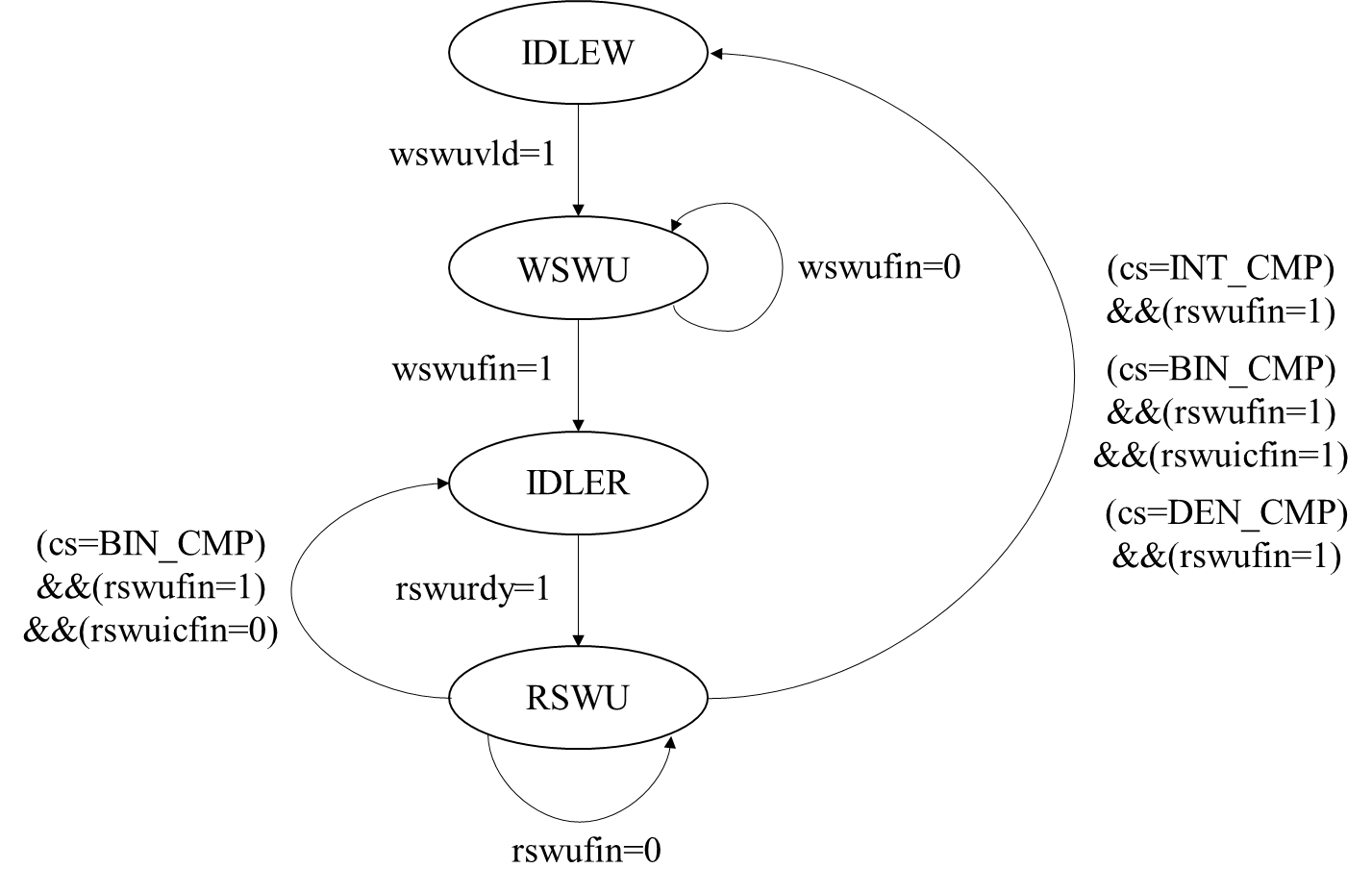


图4.20激活缓存模块SWU状态机

对激活缓存模块SWU进行功能仿真。图4.21是加速器进行第一层整型激活值卷积计算时SWU读写激活值的仿真图。在进行整型激活的卷积计算时，由于激活的位宽更大，有限的缓存空间难以进行复用，SWU需要不断的进行读写。输入的wswudata由两个SWU共用，通过握手协议控制SWU当前的读写状态。可以看到向SWU中写入20个数据后拉低wswurdy，同时拉高rswuvld等待被读取。

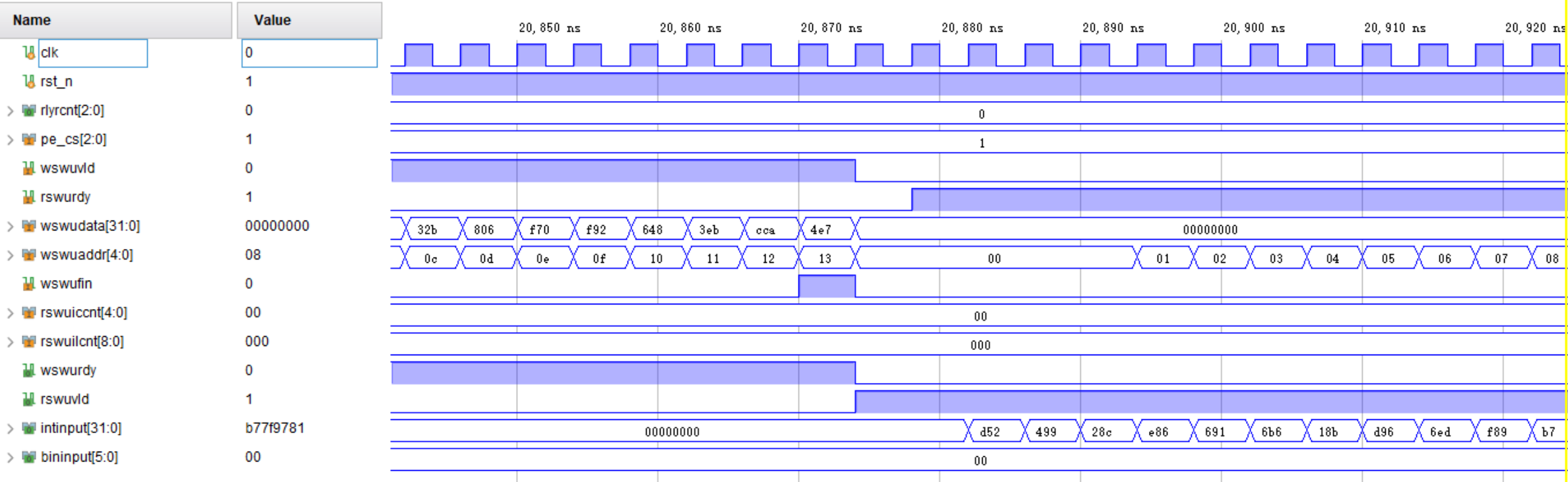


图4.21激活缓存模块整型激活值读写仿真图

当SWU缓存二值激活时，SWU可以滑动输出以对数据进行复用，对SWU读取二值激活的功能仿真如图4.22所示。当和计算控制模块握手成功之后，在下一个周期开始连续滑动12次，输出两个通道的12个输入激活值。输入完毕之后切换到下两个输入通道，等待计算阵列的读取。

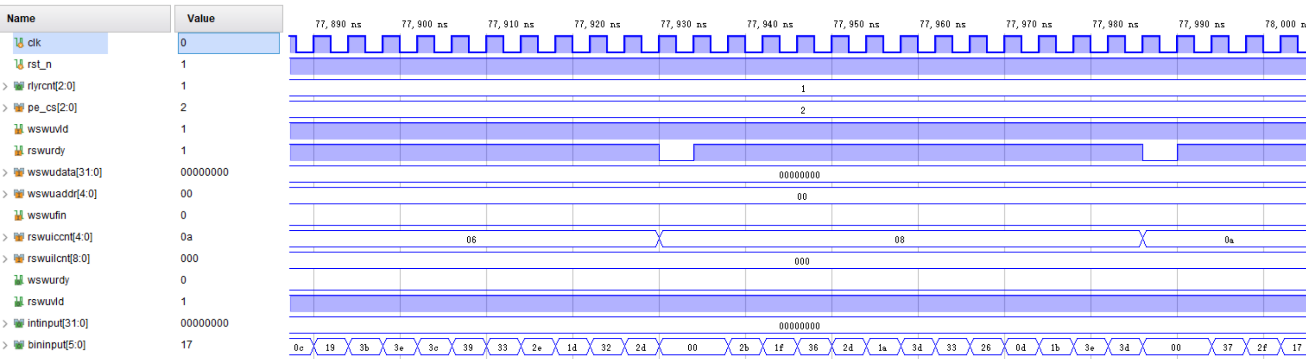


图4.22激活缓存模块二值激活值读取仿真图

（2）权重值缓存模块WEG

权重缓存模块WEG是由一组存储单元组成，其深度由卷积核的大小决定，而宽度则和输出通道的并行数TM相关。本加速器采用32个输出通道的并行计算，因此WEG的存储单元位宽是32bit。而本文算法中第一层的卷积核较大，因此深度采用第一层卷积核的大小为80。权重缓存模块的示意图如图4.23所示，缓存模块的每一列负责一个PE的权重的输入。

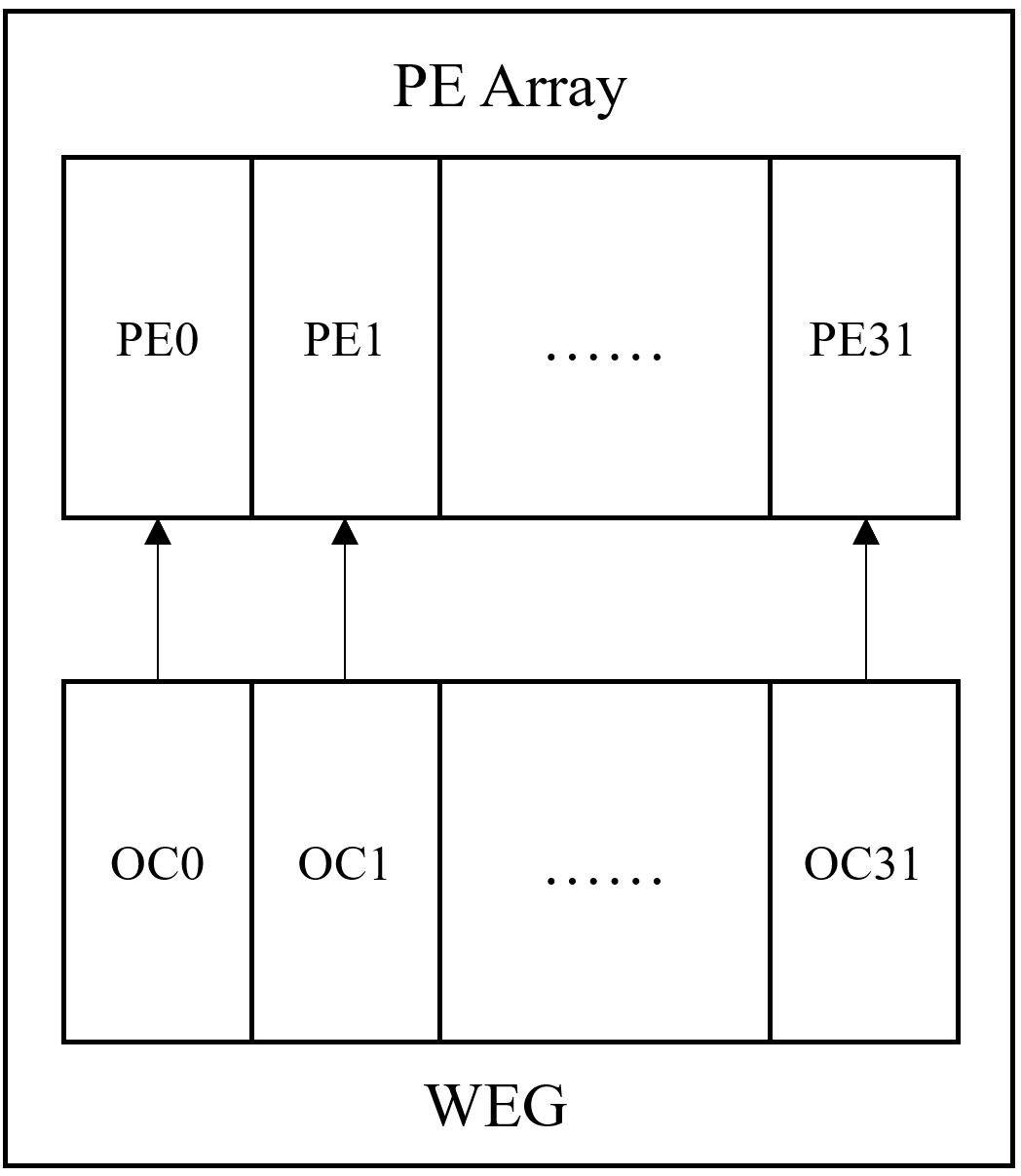


图4.23权重缓存模块示意图

在存储控制单元从Weight Buffer中读取参数写入WEG时，写入的参数的位宽为32bit。WEG根据存储控制单元提供的写入地址，将写入的参数存储到对应的32bit存储单元之中。存放在WEG中的权重参数的存储方式和在Weight Buffer中相同。整型输入的权重如图4.24(a)所示，每个32bit存储单元中的每一位权重属于不同的输出通道，即存储单元的每一列代表一个卷积核。二值输入的权重如图4.24(b)所示，和整型输入的权重相同，每列存储不同的卷积核。在此之外由于本加速器实现两个输入通道的并行，因此每列存储了一个输出通道的两个输入通道。

在计算控制模块从WEG中读取参数时，WEG根据计算控制模块的当前计算状态以不同的方式进行输出。若计算控制模块处于INT\_CMP状态，则从WEG中每次读取4行32bit的参数，且进行连续的参数输出，直至所有的参数被输出完毕。若计算控制模块处于BIN\_CMP或DEN\_CMP状态，则每次计算输出6行32bit的参数，并维持此参数输出不变直至计算控制模块跳转到空闲状态。WEG中的权重参数被分别送给32个不同的PE单元，因此在输出时每个PE单元会接收到输出参数的不同位。

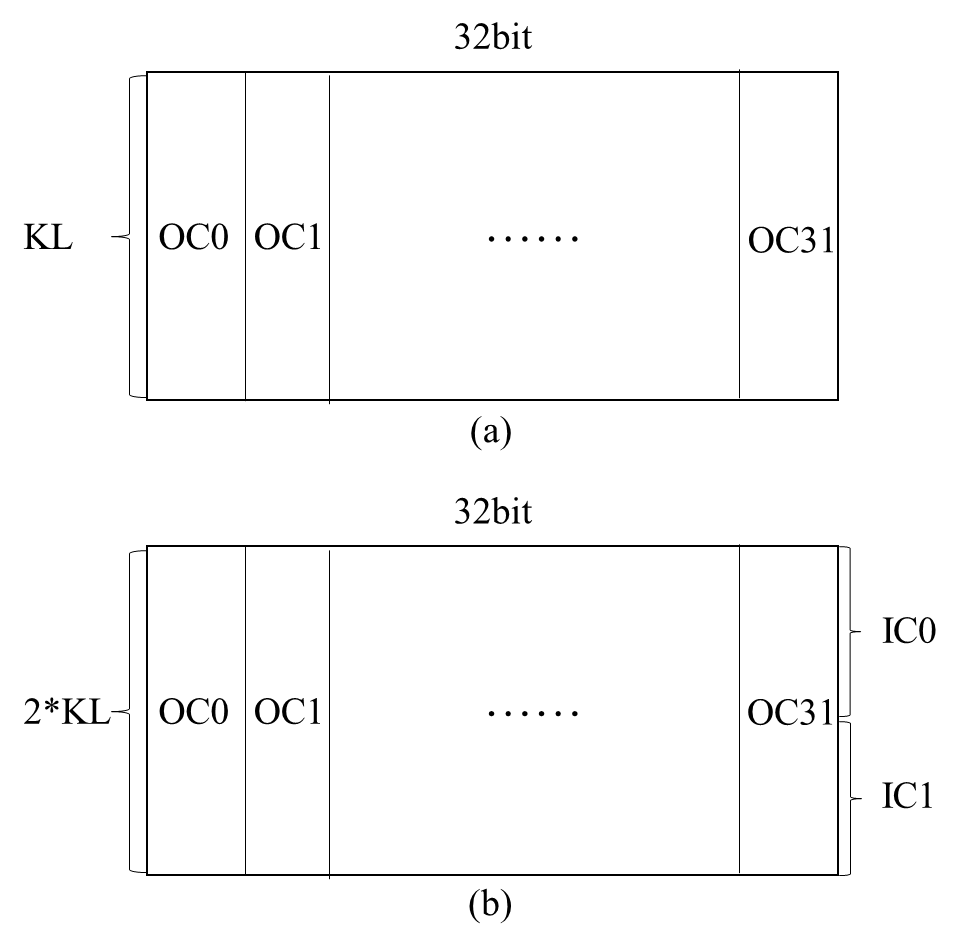


图4.24激活缓存模块WEG示意图

权重缓存模块的状态机如图4.25所示。当存储控制模块准备好写入时，且WEG处于写入空闲的状态IDLEW，则跳转到WWEG状态并进行参数的写入。当存储控制模块发出写入完毕信号wwegfin之后，跳转到IDLER状态等待计算控制模块读取WEG内缓存的参数，同时拉高读有效的信号。当计算控制模块发送读准备信号rwegrdy之后，跳转到RWEG状态进行参数的读取。WEG根据计算控制模块的状态进行计数，当参数读取完毕后拉高rwegfin。对于第一层整型输入的计算，即计算控制单元处于INT\_CMP时，由于第一层的输入通道只有一个，所以WEG中的权重无需进行更换即可完成整个输出通道的计算。因此WEG中的参数可以被多批次SWU中的参数重复使用，直至整个输出通道计算完毕即rwegolfin为高。因此在rwegolfin为低时跳转到IDLER等待计算控制单元下一次复用WEG中的参数。在其他情况下，WEG中的参数已经被消耗完毕，可以被再一次写入覆盖，即跳转到IDLEW状态等待存储控制模块的写入。

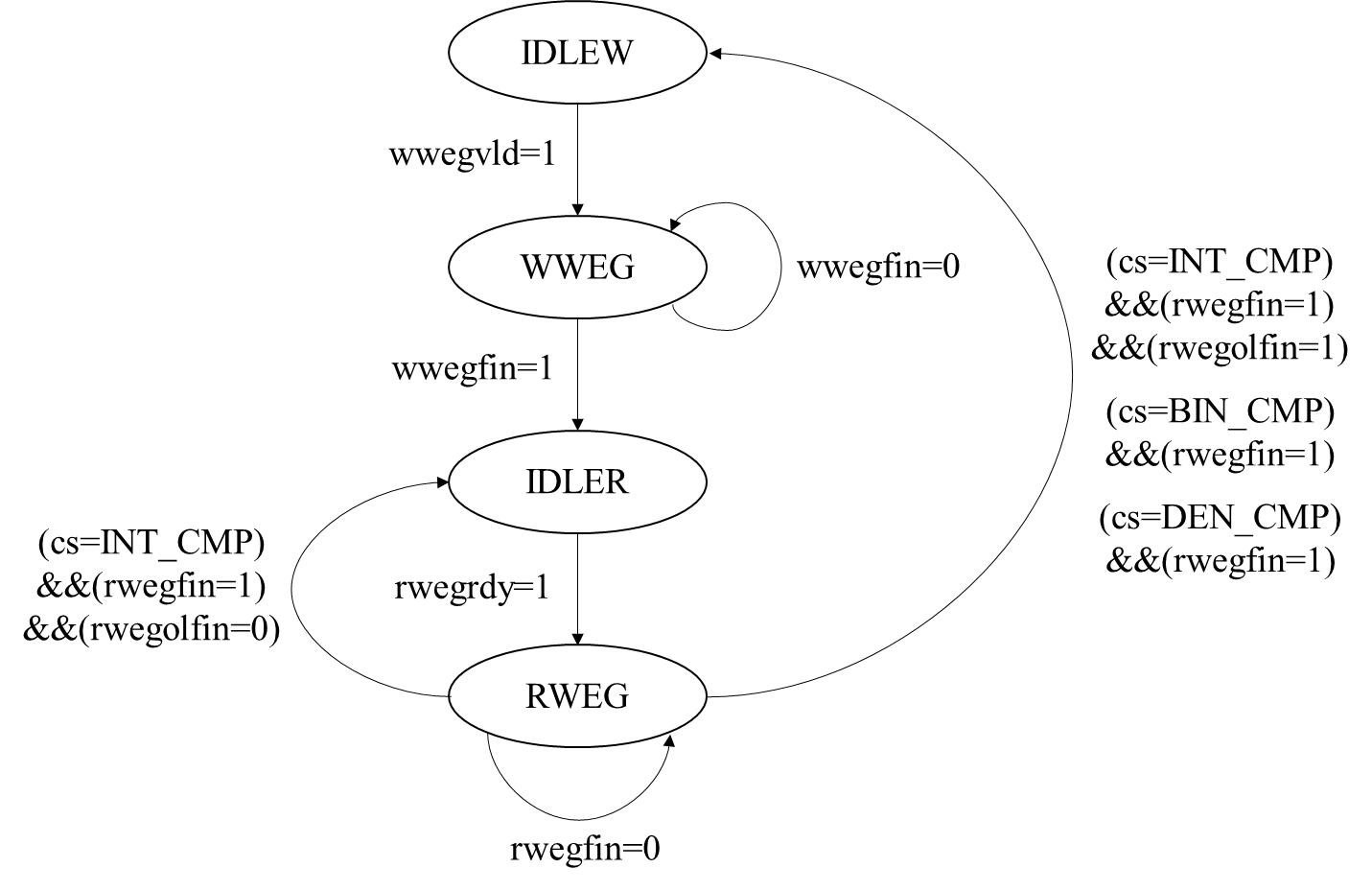


图4.25权重缓存模块WEG状态机

4.4.3 存储模块设计

加速器的存储模块分为激活值的存储模块InOut Buffer和权重的存储模块Weight Buffer。

（1）激活值存储模块InOut Buffer

激活存储模块由两块Buffer组成，称之为Buffer0和Buffer1。这两块Buffer执行乒乓操作，轮流进行激活值参数的读写。即当计算卷积神经网络的一层时，存储控制单元从Buffer0中读取输入的激活值时，计算单元输出的激活值，也就是下一层的激活值输入会被存放到Buffer1之中。当进行下一层计算时，则从Buffer1中读取输入的激活值，而计算结果则直接输出到Buffer0中进行写覆盖。通过乒乓操作的方式提高了并行的效率，参数的写入和读出互不干扰。

激活值在InOut Buffer中的存储方式根据其类型不同也有所不同，如图4.26所示。在图4.26(a)中是整型激活的排布方式，每行存放4个8bit有符号整型输入。由于第一层的输出只有一个输入通道，是一个连续的一维数据，因此只需要连续的存放在InOut Buffer中即可。在图4.26(b)中是二值激活的排布方式，由于其他层拥有32的倍数个输入通道，所有InOut Buffer中每行的每一位都是不同的输入通道的激活值，即每一列是不同的输入通道。每列中一个输入通道的数被连续的存储，长度为输入的长度IL，其后则存储下一组32个输入通道的激活值。对于全连接层的激活值，由于其数量较少，仅为256bit或64bit，所以将其简单的连续存储到InOut Buffer中。

InOut Buffer的存储方式方便参数的存取，是由本加速器的并行方式所决定的。对于整型激活的读取，每次读取20行的输入，且下次读取的地址在上一次读取的地址基础上递增(KS/4)。对于二值激活的读取，每次读取(tmpwid+kl-1)行输入，一次取出32个输入通道。下一次则地址递增IL，读取下一组32个输入通道的激活值。同时，对于计算单元的输出而言，其存在32组输出，分别对应32个输入通道。因此可以将每组输入拼接至一个32bit数的不同位，然后存放到InOut Buffer中。

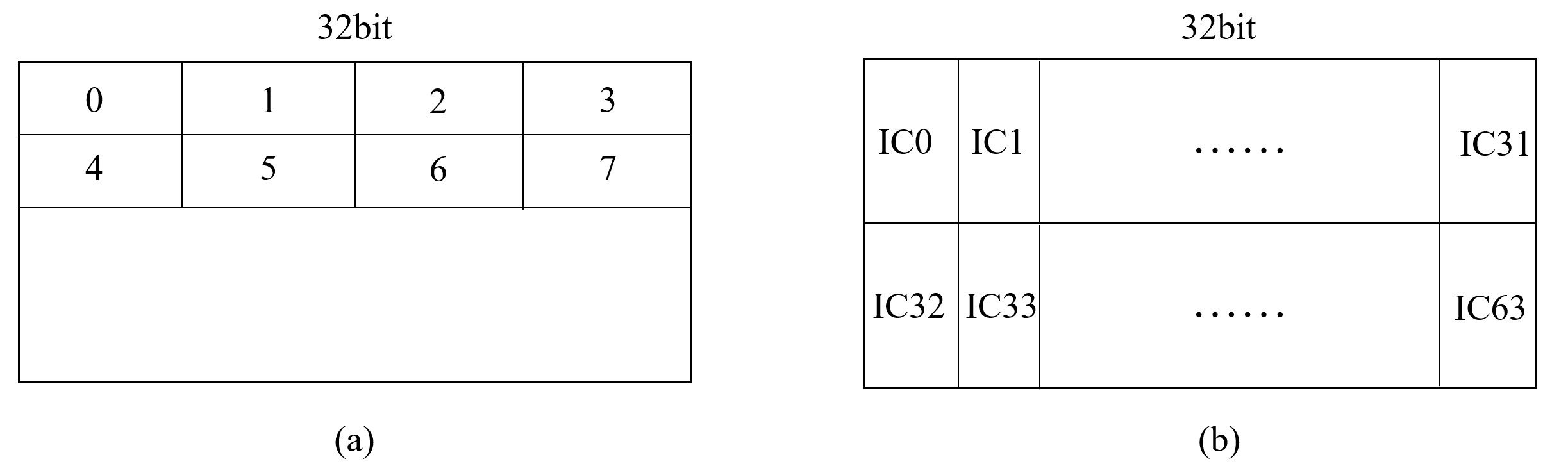


图4.26 InOut Buffer中数据排布方式

（2）权重值存储模块Weight Buffer

权重存储模块是一块整体Buffer，且在计算过程中仅进行读取而不需要进行写入。如图4.27所示，权重存储模块的位宽是32bit，每bit存储不同输出通道的权重，即每一列存放不同输出通道的权重。由于卷积计算需要将所有输入通道计算完毕之后，中间值才可以计算完毕被输出。所以在同属于一个输入通道的32组输出通道权重之后，存放的是下一个输入通道的相同32个输出通道的权重。直到这组32个输出通道的权重的所有输入通道被存放完毕之后，再存放下一组32个输出通道的权重。对于全连接的权重而言，可以将全连接层的输出长度视为特征通道数，以相同的方式进行存储，而输入通道视为仅有一个。

权重存储模块的数据排布方式是由计算阵列的切片方式决定的。由于计算阵列的输出通道切片TM是32，输入通道切片TN是2，所以对于二值激活输入的计算，每次存储控制单元从Weight Buffer中读取一组32个输出通道的权重中的两个输入通道的权重。

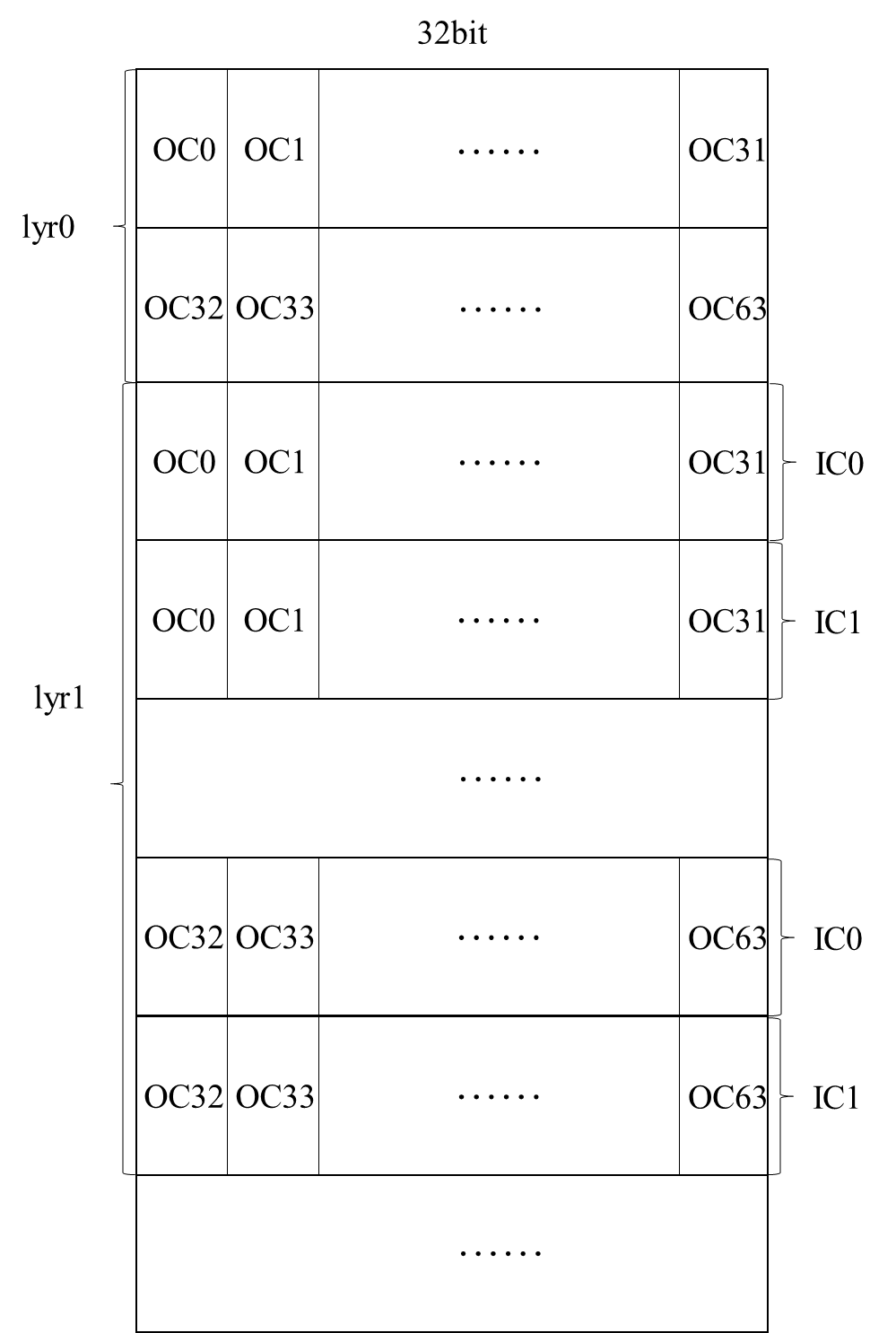


图4.27 Weight Buffer中数据排布方式

4.4.4 计算模块设计

（1）PE阵列

加速器的PE阵列模块由32个PE单元组成，每个PE单元负责处理一个输出通道的卷积计算。PE单元的结构和功能如4.3节所述，它包含三级流水线：乘累加、中间值累加和批处理归一化。

图4.28展示了PE单元的主要端口。为了实现高效的参数复用和高吞吐率的计算，我们在4.1节中对加速器进行了设计空间探索，确定了输出通道并行指数为32。因此，我们采用了32个PE单元并行处理不同输出通道的卷积核。这意味着每个PE单元需要输入不同的权重参数，这些参数由WEG模块输出的binweight经过拆分后分配给各个PE单元。而输入特征图则可以在所有PE单元之间共享，无论是intinput还是bininput都由SWU模块广播给所有PE单元。

当计算控制模块从状态机跳转到计算状态时，它会拉高pe\_en信号来使能PE阵列模块开始计算。同时，它会根据当前输入特征图的类型（整型或二值）向PE阵列模块输入sel信号来选择相应的计算方式。另外，中间值累加也会根据计算控制模块提供的acc\_sel信号来选择要累加的值。由于每个输出通道的阈值在一层卷积计算中是固定不变的，因此只需要在每轮输出通道的卷积计算开始时根据当前层数和输出通道数从缓存模块读取一次，并分配给所有PE单元。当计算控制模块计数到中间值累加完成时，它会拉高out\_en信号来使能PE阵列模块输出结果。PE阵列模块会将每个PE单元缓存的TMPWID个累加中间值与阈值进行比较，并将比较结果以TMPWID行32bit的形式输出。

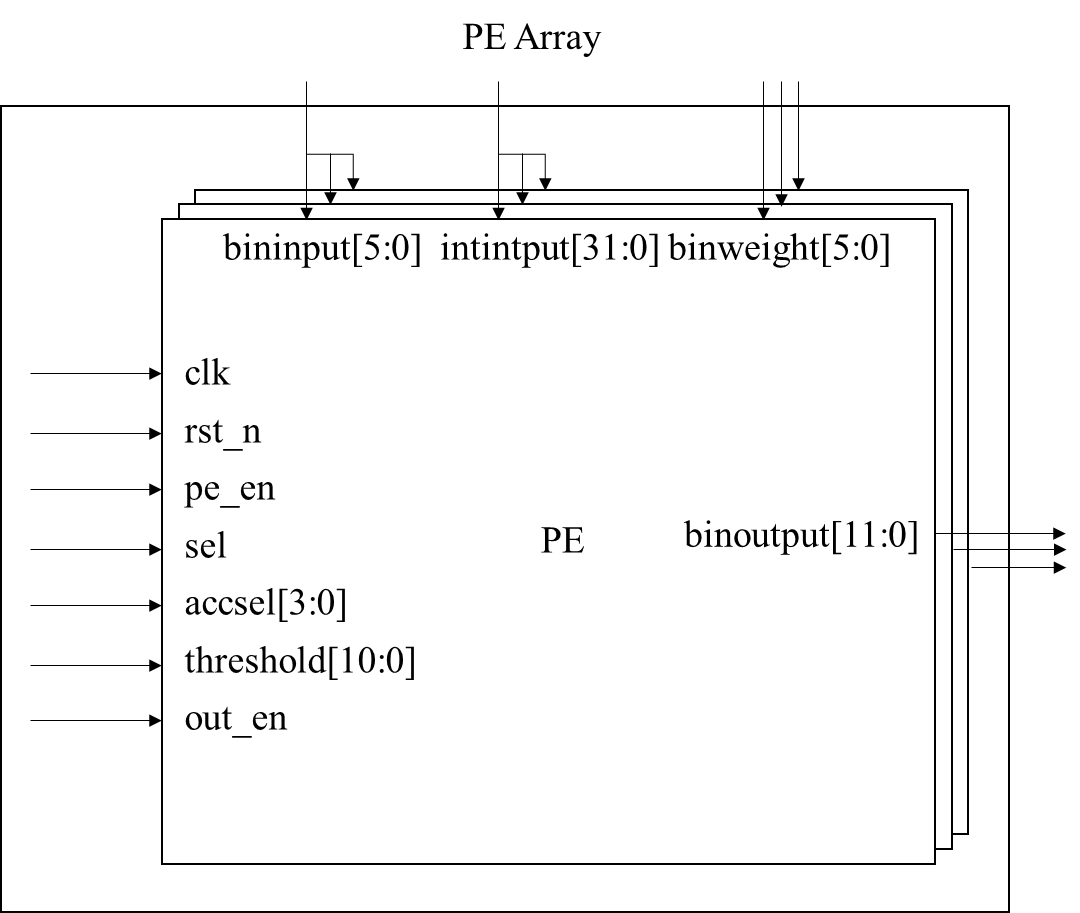


图4.28 PE阵列示意图

（2）池化模块Pooling

池化单元如图4.29所示。本加速器实现的算法中所有的池化步长均为2，池化的长度也为2，这意味着所有的池化之间不存在交叠。在本文采用的算法中，除了最后一层卷积层外，所有的卷积层均采用最大池化的方式。在本加速器中，激活层和批处理归一化层被融合进了阈值比较器之中，同时使用符号函数来代替激活层。因此池化单元的输入是已经经过激活且量化后的1bit数，每个PE单元输出TMPWID个1bit数。最大池化是指取输入值中的最大值作为输出结果，而1bit的输入只有0和1两种情况。取两个1bit值的最大值等效于或运算，因此在池化单元中只需要将TMPWID个1bit输入两两相邻取或运算，即可得到池化后的(TMPWID/2)个输出值。池化单元内部进行计数，依次将输出值写回InOut Buffer中。

此外，由于卷积计算需要对输入进行padding操作，由于需要对输出padding的卷积层的卷积核长度均为3，因此只需要在输出的头尾添加1个0即可。对于Pooling单元而言，只需要在每轮输出通道计算开始和结束时直接向InOut Buffer写一行0值即可完成padding。



图4.29池化单元Pooling示意图

在本文的算法中，除了最大池化外，还存在全局平均池化。全局平均池化就是将一个输出通道的所有输出作为输入，然后取其平均值作为输出。在本加速器中可以利用PE单元的中间值累加单元来实现一个输出通道的所有输出值的累加，然后通过阈值比较器中阈值的修改来实现平均值的输出。全局平均池化的具体计算过程如公式(4.12)所示，卷积乘累加的整型结果需要乘上批处理归一化、激活以及权重的缩放系数，然后再加上批处理归一化的权重。再将这些值累加之后取平均值，并以其符号位作为输出的最终结果。

经过公式(4.12)的推导，我们可以在PE单元中直接将所有的acc\_sel均置0，那么中间值缓存单元中的第0个reg的值即为此输出通道输出的累加值。然后由公式(4.13)可得为全局平均池化修正过的新的阈值。将累加值和此阈值进行数值比较，得到的结果便是全局平均池化的结果。因此对于第五层卷积层而言，PE单元的输出无需再进行池化处理，可以直接写回InOut Buffer。

在设计完成后，对计算阵列的功能进行仿真，验证其执行整型输入卷积、二值输入卷积和全连接层三种不同类型计算的功能正确性。下面对其中整型输入卷积和二值输入卷积的功能仿真结果进行介绍。

图4.30是计算第一层整型输入激活值的卷积层时一个PE单元的仿真波形图。从SWU中输出的intinput每个周期输出4个8bit的有符号函数，连续输出20个周期，同时WEG也每周期向此PE单元输出4bit的权重。在PE的第一级流水中，先用binweight[3:0]和4个8bit有符号数相乘，得到intmulti[3:0]。然后将4个intmulti相加，并将结果intmultisum送往下一级。下一级中进行中间值累加，将intmultisum和acc\_sel指定的中间值缓存单元中的中间值累加。虽然第一层的输入通道为1无需等待所有输入通道累加完毕，但是为了池化计算的方便性，等待12个输出点均输出完毕之后再一次性进行阈值比较。当out\_en使能时，阈值比较器将中间值缓存单元中的所有值分别和阈值threshold进行比较，并将结果拼接成12bit的binout。

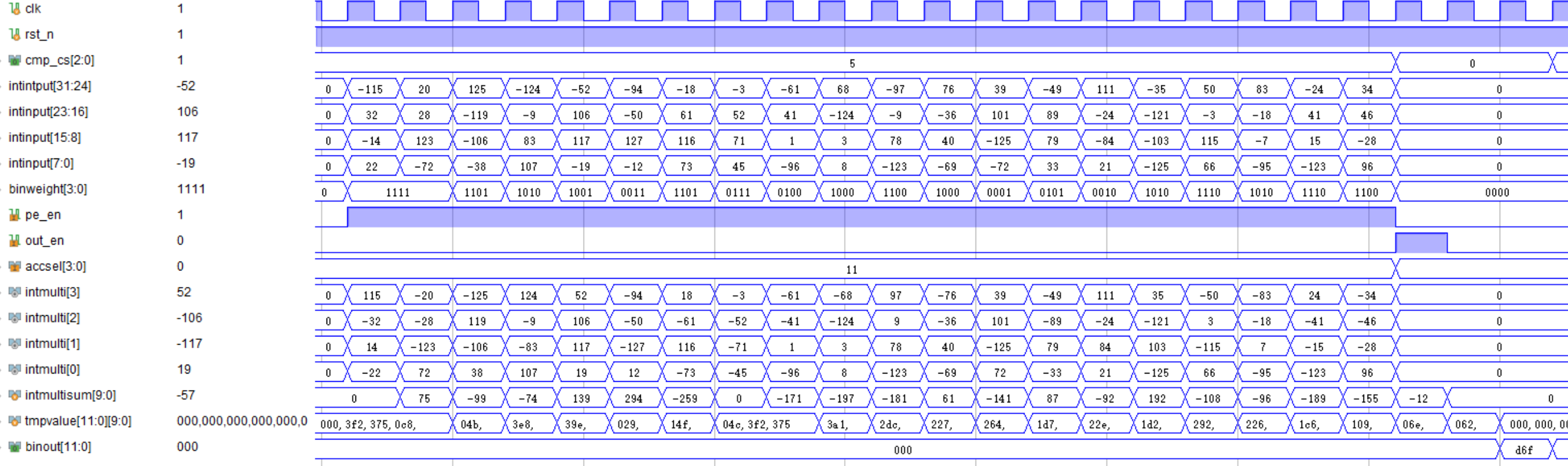


图4.30整型输入时计算单元PE仿真波形图

计算二值输入时加速器计算单元的仿真波形图如图4.31所示。PE单元从SWU中每个周期输入6bit的bininput，连续输入12个周期。WEG中则保持相同的6bit权重输出，这是因为一个输出通道中共享相同权重。输入的bininput和binweight进行同或运算，并将运算的结果进行处理生成4bit的有符号数binmulti送往下一级。中间值累加模块根据当前的acc\_sel将输入的binmulti和对应的tempvalue缓存单元累加。当所有的输入通道均被累加完毕之后，out\_en被拉高，第三级中的阈值比较单元将所有输出和阈值进行比较，然后输出为12bit的binout。

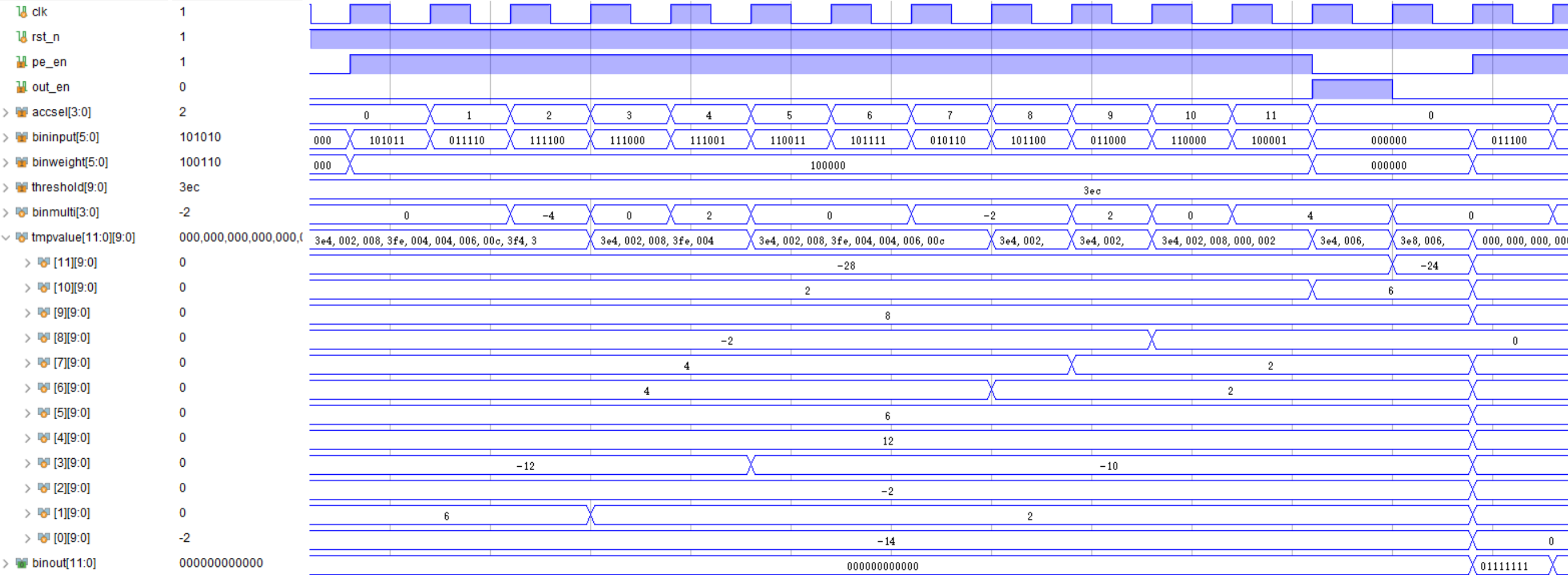


图4.31二值输入时计算单元PE仿真波形图

4.5加速器计算过程

在本章之前的内容中，介绍了加速器的各个模块功能。本节将通过举例的方式，简要的介绍加速器三种计算状态下的数据流，以此来陈述加速器的整体计算流程。

（1）整型输入的卷积计算

当计算控制模块的主状态机为INT\_CMP时，执行算法中第一层整型输入卷积的计算。卷积计算的输入是(1,3600)的一维ECG信号，经过Padding之后被存放在InOut Buffer中的Buffer0内，其数据类型是8bit的有符号数。Buffer0位宽是32bit，每行存放4个输入值，输入的一维ECG信号的地址范围是[0,917]。

然后开始第一组32个输出通道的计算。计算控制模块等待缓存模块的参数被写入完毕，存储控制模块则开始向缓存模块写入数据。对于激活参数,如图4.32所示，存储控制模块向SWU0写入80个8bit输入信号，使用20个周期进行连续的传输。SWU0中的参数写入完毕之后，存储控制模块将地址递增4，再向SWU1连续写入20个周期，共写入80个8bit的输入，SWU0和SWU1中的数据存在交叠重复。



图4.32整型输入时向激活缓存单元输入过程示意图

对于权重参数，存储控制模块向WEG0中写入32组长度为80的1bit权重，分80个周期连续写入。WEG0中写入完毕之后，存储控制模块向WEG1中写入另一组32个输出通道的权重。权重的输入过程如图4.33所示。



图4.33整型输入时向权重缓存单元输入过程示意图

当权重缓存单元和激活缓存单元被写完毕，进入等待读出的状态。当计算控制模块和缓存单元握手成功，缓存单元开始向PE阵列传送参数，如图4.34所示。激活缓存单元SWU0每周期发送一行4个8bit整型数据广播到每个PE单元，而权重缓存单元WEG0每周期发送32个4bit的权重数据分别送给32个PE单元，一共发送20个周期之后结束计算。计算结束之后，下一次计算由SWU1和WEG0来传送数据。由于单个SWU的写入需要20个周期，可以被计算过程覆盖。单个WEG的写入虽然需要80个周期，但是只需要配置一次即可反复使用直到本层计算结束，所以PE阵列可以做到连续无等待的计算。

权重缓存单元中的数据被分发给32个PE单元，而激活缓存单元中的数据被广播给32个PE单元。PE单元内部是三级流水线，其中前两级是无阻塞的流水线。输入的激活值和权重值乘累加之后输入第二级，第二级根据中间值选择信号，选择对应的中间值累加，并将累加结果缓存。对于整型输入的运算而言，由于输入通道只有一个，中间值只需要和0值累加之后存入对应缓存单元即可。当所有TMPWID个中间值被计算完毕之后，将所有累加的中间值送往第三级进行阈值比较，并在一个周期后全部输出至池化单元。



图4.34 整型输入的计算过程示意图

PE单元的计算结果经过池化单元通过或运算进行最大池化之后，依次存放到InOut Buffer的Buffer1中，一共有6个32bit的输出，分为6个周期即可输出完毕。由于InOut Buffer中的两块Buffer执行乒乓操作，所以存储控制模块读取激活到SWU和池化单元写回下一层的激活值之前不会有读写冲突。

（2）二值输入的卷积计算

当计算控制模块的主状态机处于BIN\_CMP时执行二值输入的卷积计算，其输入由池化单元在上层计算执行时经过Padding后写入InOut Buffer中。以第二层卷积为例，其输入是(64,150)的矩阵，即64个输入通道、长度为150的1bit激活输入。首先由存储控制模块向激活缓存模块和权重缓存模块输入待计算的参数。对于激活而言，一次向SWU中输入32个输入通道，长度为13的1bit激活输入，一共需要写入13个周期。对于权重而言，一次向WEG中输入32个输出通道，2个输入通道，长度为3的1bit权重，一共需要写入6个周期。激活和权重的写入过程如图4.35所示。当SWU和WEG中参数写入完毕，计算控制模块和缓存单元握手成功后，缓存单元开始向PE阵列输出参数。每轮计算持续TMPWID个也就是12个周期，SWU通过滑动的方式输出参数。对于WEG而言，在12个周期内维持相同的权重输出，输出192bit的数据分发给32个PE单元，每个单元接收两个输入通道一个输出通道共6bit的权重数据。



图4.35二值激活时缓存单元输入输出示意图

二值激活的PE单元计算过程和整型激活时相同，仅是切换为同或实现的乘累加单元。PE单元计算的结果经过池化单元通过与操作执行最大池化，然后写入InOut Buffer中。

（3）全连接层的计算

当计算控制模块的主状态机处于DEN\_CMP时执行全连接层的计算。以第一层全连接层为例，其输入是(1,256)的一维1bit激活，权重是(256,64)的二维矩阵。因此全连接层可以视为1个输入通道，64个输出通道，卷积核和输入特征图长度一致的卷积运算。由于激活只有256bit，存储控制模块可以一次性花费8个周期将所有激活值输入SWU中。而WEG则有64个输出通道，每个输出通道的权重长度为256。每次计算前存储控制模块向WEG一次输入32组6bit的数据，共花费6个周期。当缓存单元参数写入完毕之后，计算控制模块进入计算状态，一次计算从SWU中读取6bit的输入，WEG中读取32个6bit的权重。一个周期的计算完毕之后，切换另一个WEG输出32个6bit权重，而SWU则输出下一组6bit的输入。计算单元的空闲主要是因为在等待WEG中参数的配置。

4.6本章小结

本章首先介绍了加速器的整体结构，并分析了加速器的数据流，对加速器的数据复用系数进行推导。基于数据流的分析，对加速器的资源占用和能效比进行建模，确定了最优的加速器尺寸。然后设计了加速器的基本运算单元，使用同或运算简化了乘累加运算，并通过系数融合的方式，使用较少的额外硬件来支持非二值部分运算，实现高能效的运算单元设计。最后设计了加速器的其他各个模块，并进行仿真验证。加速器的缓存模块和计算阵列配合，充分的进行数据复用，降低访存次数。最终得益于简化的运算单元和对数据的充分复用，实现了高能效的加速器设计。

第五章 ECG信号检测神经网络加速器的FPGA验证

本章结合上两章设计的基于BNN的ECG信号检测卷积神经网络及其专用加速器，在FPGA平台上对硬件进行实现并进行指标测试。

5.1实验测试平台及验证方案

本文将在XLINX的FPGA开发板上对二值化加速器及其系统进行实现，然后对其硬件资源占用和功耗进行分析。对于硬件的吞吐率和精度方面，将通过行为级仿真和中间层参数比对的方式保证软硬件的一致性，从而确定硬件的计算量和精度。

5.1.1实验工具介绍

本文采用Zynq-7000 Miz7035 FPGA开发板作为面向心律检测的轻量化卷积神经网络加速器的硬件验证平台。开发板整体结构如图5.1所示，开发板搭载了一颗Xilinx Zynq XC7Z35-FGG676-2I的可全编程SOC芯片和4片512MB DDR3内存。此外，该开发板还拥有270K逻辑单元（Logic Cell），500块36Kb的块随机存储器（Block RAM，BRAM），6输入查找表（Look-Up-Table）和数字信号处理单元（Digital Signal Processing，DSP）等资源。

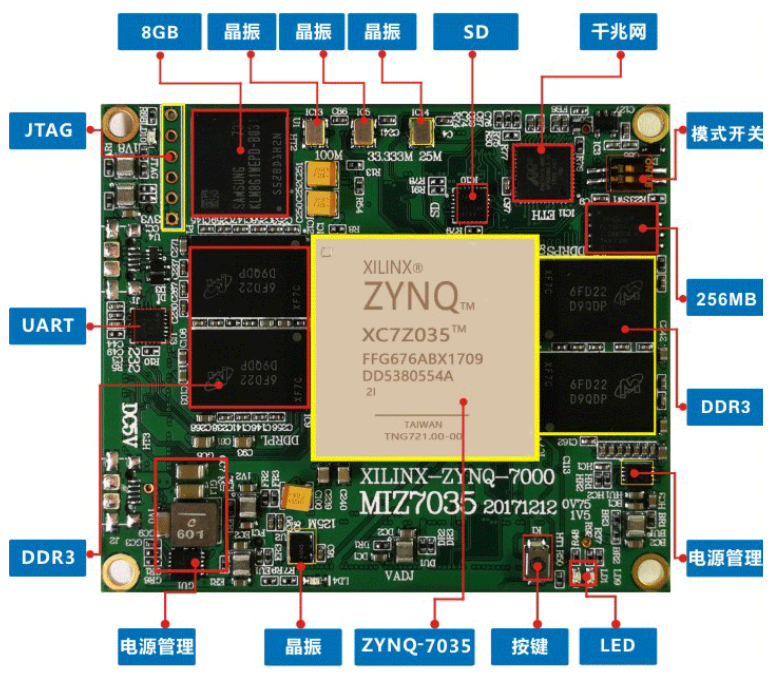


图5.1 Zynq-7000 Miz7035 FPGA芯片核心板

Zynq-7000 Miz7035 FPGA开发板的详细资源信息如表5.1所示：

表5.1 Miz7035 FPGA开发板资源明细

|  |  |
| --- | --- |
| 资源 | 参数 |
| LUT | 171900 |
| 逻辑单元 | 270K |
| FF | 343800 |
| BRAM | 17.6Mb/(500\*36Kb Blocks) |
| DSP | 900 |
| MMCM | 8 |
| IO | 130 |

本文使用Xilinx公司的Vivado2018.2设计套件作为设计工具，该套件提供了以系统为中心的SoC开发环境，助力硬软件及系统生产力能力的提升。此外，Vivado还提供标准IP，方便系统开发；提供开发过程中的功耗、资源利用率和时序日志，有助于解决系统集成和实现过程中的问题，以上特性都极大地方便了设计者的使用。

5.1.2系统验证方案

系统的功能验证分为FPGA实现前的功能验证和实现后的整体系统验证。在第4章中已经对系统功能各个模块进行验证保证功能正确性。由于待测数据量较大，本文先通过对实现前加速器系统的验证保证软硬件功能的一致性，再使用测试用例对实现后的系统进行冒烟测试，最终验证系统功能的正确性。

实验前的功能验证如图5.2所示。加速器的参考模型使用Python语言编写，其具体功能和硬件的预期功能保持一致，包括运算顺序、参数位宽和存储顺序，可以作为硬件功能的正确对照。通过挑选一些ECG输入信号写入到系统的BRAM中，然后对加速器进行仿真，通过仿真的系统函数将加速器硬件计算过程中的中间值导出，然后和参考模型的输出进行对比。最终通过多个数据输入例子的比对，保证软硬件的一致性。然后使用仿真速度更快的参考模型对整个测试集的ECG信号进行测试，根据参考模型的分类结果得到加速器硬件的精度。

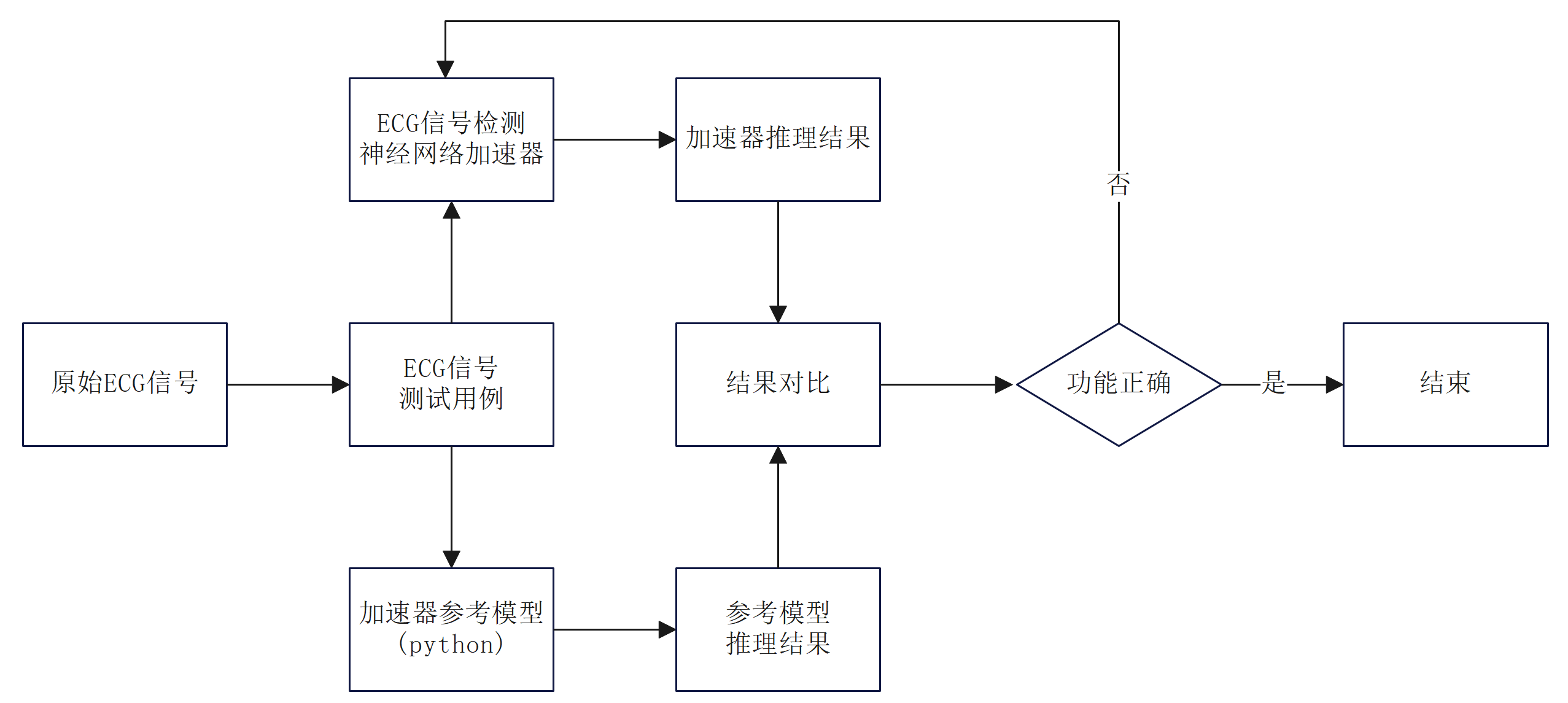


图5.2加速器功能验证流程

实现后的加速器整体系统如图5.3所示，使用此系统对加速器整体功能进行测试。ZYNQ CPU在本系统中作为加速器的上位机，完成待测数据的写入以及读取加速器输出分类结果的功能。CPU通过AXI GP（General-Purpose）主机端口和由Vivado自带IP生成的BRAM控制器相连。然后BRAM控制器直接和加速器中InOut Buffer中的Buffer0直接相连。配置待测数据时PS端直接通过BRAM控制器向Buffer0中写入数据。CPU和加速器之间直接使用AXI\_Lite接口相连，当向Buffer0中写入待测数据完毕之后向加速器中的寄存器写入ECG信号写入完毕信号。待加速器前向推理完成，PS端通过AXI\_Lite总线从PL端寄存器中读取分类的结果。

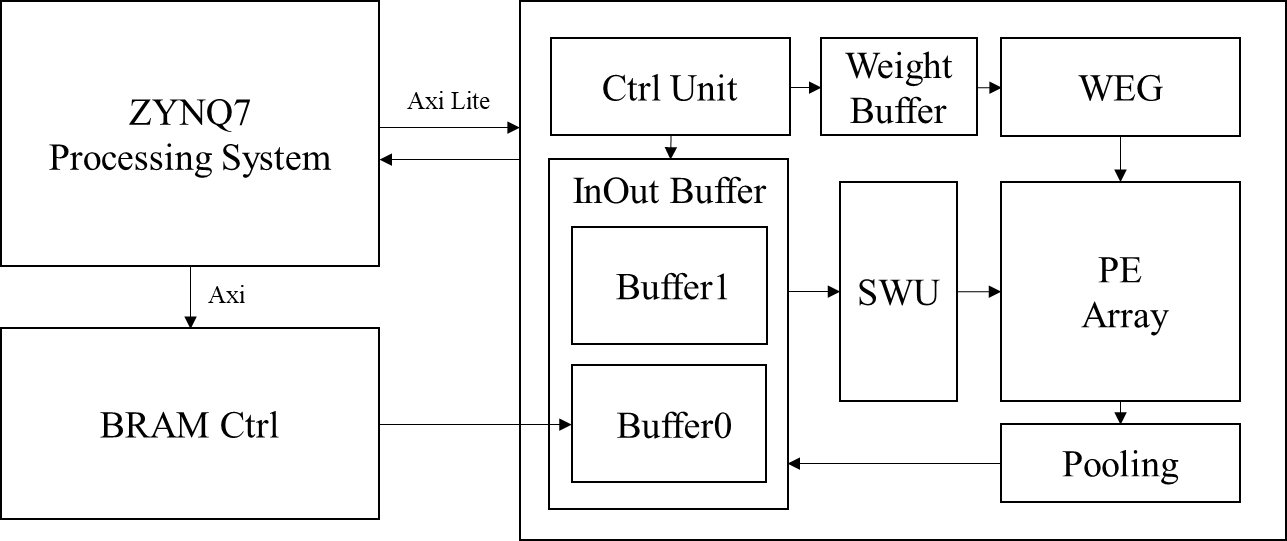


图5.3加速器功能验证系统

通过实现前行为级仿真和实现后的数据实测，确保了软硬件的一致性，从而得出硬件的计算量和精度。利用VIVADO对系统的功耗和硬件资源占用进行评估，最终得到硬件的吞吐率和能效比。

5.2 FPGA验证实现

FPGA的实现过程包括设计输入、分析综合、布局布线和下载配置。设计输入阶段是将FPGA的功能描述用HDL（硬件描述语言）编写成源代码文件或者用原理图或IP核等方式构建FPGA的逻辑结构。本文的加速器部分使用verilog语言进行编写，整体构建为一个IP核。然后以block design的方式，调用vivado中提供的IP核和系统PS端相连，如图5.4所示。

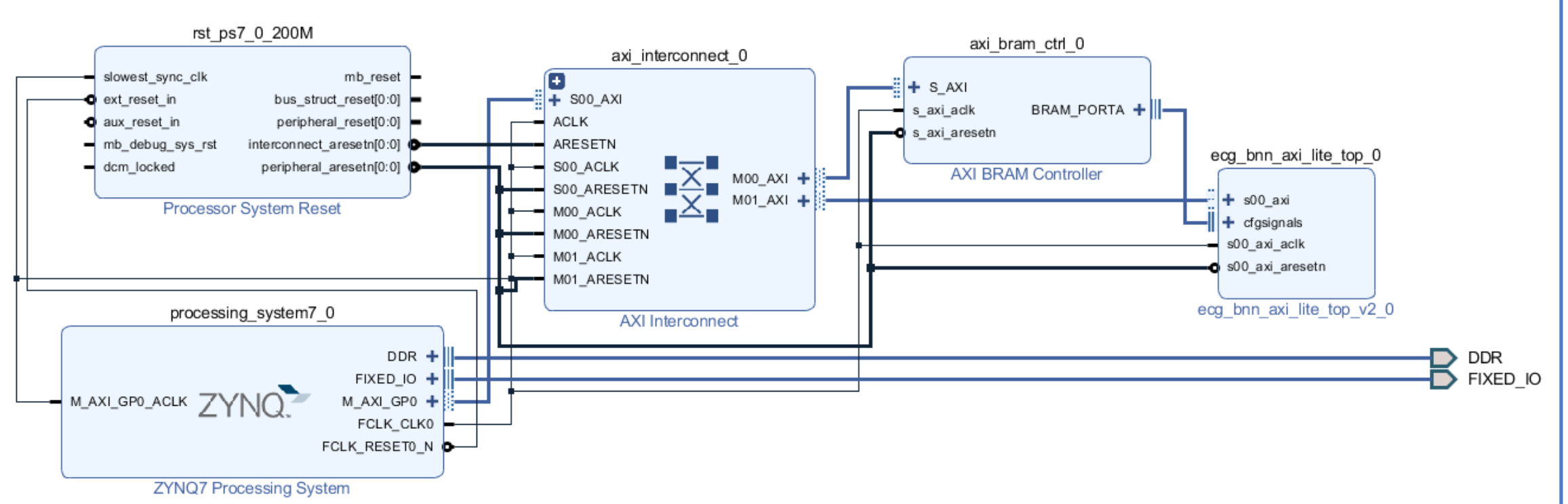


图5.4 整体系统示意图

分析综合阶段是将设计输入转换为FPGA上的硬件单元（如LUT、寄存器、乘法器等），形成门级网表，并根据不同的约束条件（如速度优先或面积优先）进行优化。本文设计的加速器是一个单时钟电路，仅对加速器的输入时钟进行约束。在综合中选择了默认的设置，其将电路的层次结构打平，综合的优化时序和面积。在编码风格中，对于寄存器尽量添加了时钟门控，以降低其动态功耗。对于较长的逻辑电路，如参数缓存的地址计算等模块，进行分割打拍，避免时序上的紧张。在综合中，计算单元均采用LUT实现，而没有使用DSP，这是因为代码中不含有乘法的部分，而中间值的加法运算和同或运算均由LUT实现。特别的，本加速器的二值输入乘累加单元采用2输入通道并行，最终6个输出结果的累加可使用1个6输入LUT基本单元来实现。通过以上的手段，使得综合出的电路资源占用少，时序余量大。

布局布线阶段是将分析综合得到的门级网表映射到FPGA的具体位置，并完成各个模块之间的连接，并根据不同的约束条件（如位置约束或时序约束）进行优化。本设计中加速器IP的时钟由PS端直接提供，没有对位置进行特别约束。在布局布线之后，查看生成的时序分析报告如图5.5所示。由于本设计避免了较长的组合逻辑设计，以及整体流水线的设计，最终不存在时序违例，且有一定的时序余量。

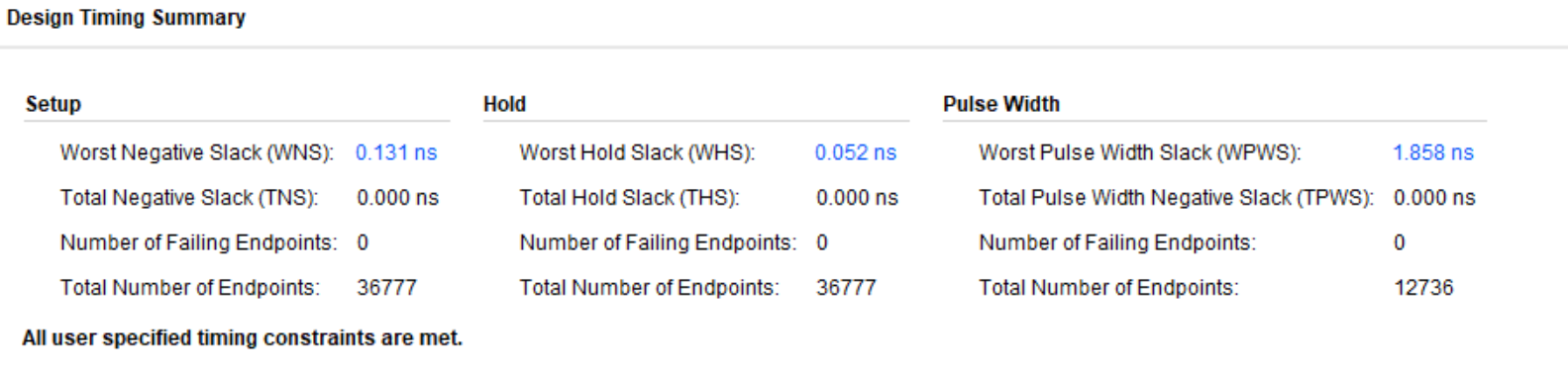


图5.5时序分析报告

布局布线后加速器整体的资源利用率报告如图5.6所示，除加速器IP之外，还包含PL端AXI-lite接口以及BRAM控制器的资源占用。

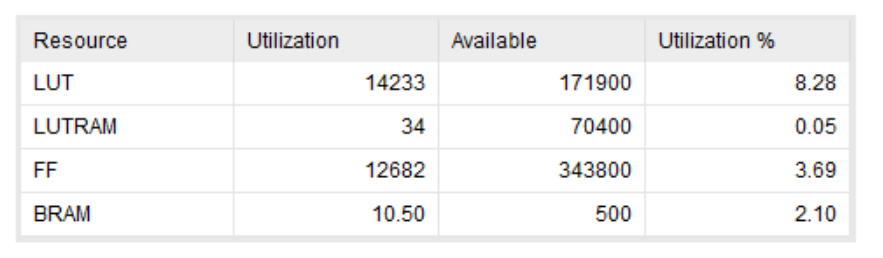


图5.6 资源利用率报告

二值化专用加速器的资源占用情况如表5.2所示。可以看到LUT中使用了13kLUTs，其中PE阵列占用了大部分，这是由于本设计中使用较多LUT实现加法运算导致的。由于二值量化的参数和较为高效的中间值缓存设计，使用了较少的BRAM。

表5.2 ECG信号检测卷积神经网络加速器资源利用率

|  |  |  |  |
| --- | --- | --- | --- |
| 资源 | 使用数量 | 可用总量 | 利用率（%） |
| LUT | 13906 | 171900 | 8.09 |
| FF | 12257 | 343800 | 3.57 |
| BRAM | 10.5 | 500 | 2.01 |

使用VIVADO对实现后的加速器设计进行功耗预估，使用XPE工具导出查看其功耗分布如图5.7所示。其中PS端的功耗为1.576W，PL端的功耗为0.353W。PL端功耗中，动态功耗为0.184W，静态功耗为0.169W。

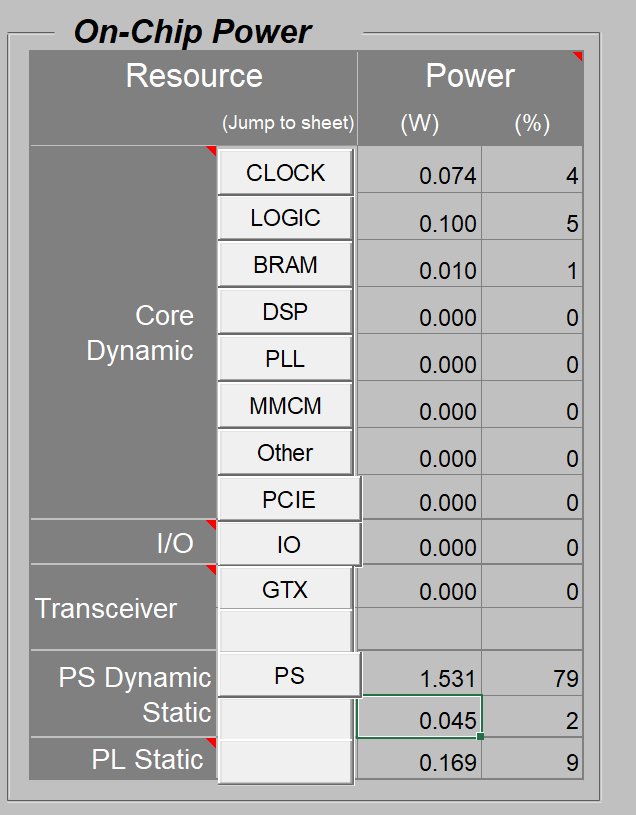


图5.7 加速器功耗分析报告

5.3 系统性能测试

根据图5.2的验证方案，对软硬件一致性进行了测试。使用$writememb系统函数将加速器仿真的中间缓存值写入文本文件，然后和python编写的参考模型进行比对，保证了软硬件的一致性。然后使用仿真速度更快的参考模型进行仿真，对硬件的精度进行测试。硬件的精度如图5.8所示，和算法的分类结果相比有少量误差，这是由于中间值位宽较少造成的。但是最终的精度没有降低，为92.83%。

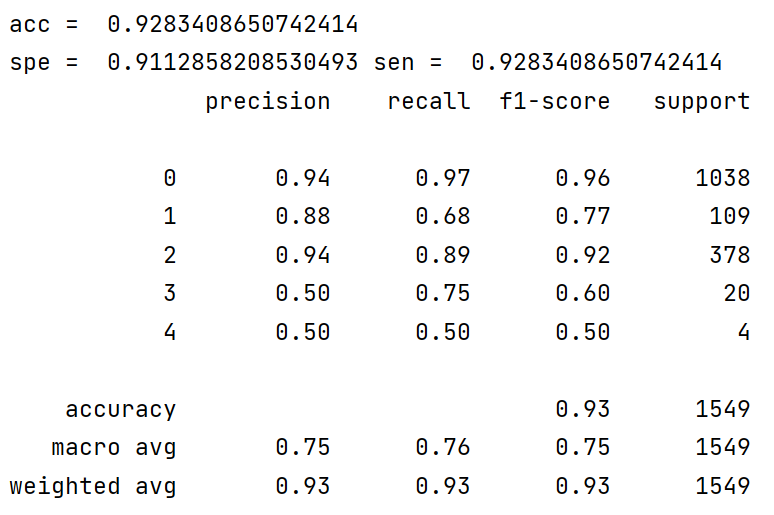


图5.8 硬件精度

在硬件实现后，将比特流下载到开发板。当上电之后，PS端通过BRAM控制接口向加速器的BRAM中写入一条分类为N的ECG信号，写入完毕之后通过AXI-lite接口向加速器寄存器中写入使能信号。当加速器推理完毕之后，PS端读取到PL端推理完成信号被拉高，然后从PL端寄存器中读取推理结果。PS端将收到的推理结果处理之后，通过UART发送给上位机。最后实测收到的结果如图5.9所示。实测的推理结果中第0类也就是N类的值是最大值42，因此推理的分类结果为N类，分类正确。且和参考模型的推理结果对比一致。



图5.9 串口打印结果及参考模型仿真结果

在确保软硬件功能一致的情况下，可以认为硬件最终实现的准确率为92.83%。根据表3.3的网络结构可以方便的计算得，加速器执行一次推理的执行了1.76🞨107次乘加操作。同时利用VIVADO进行仿真，得到硬件完成一次推理的时间为342us，如图5.10所示。结合一次推理的操作数，可以计算出加速器的吞吐率为**51.6GOPS。同时根据之前测得的功耗，可以得出加速器的能效为146.2GOPS/W。**

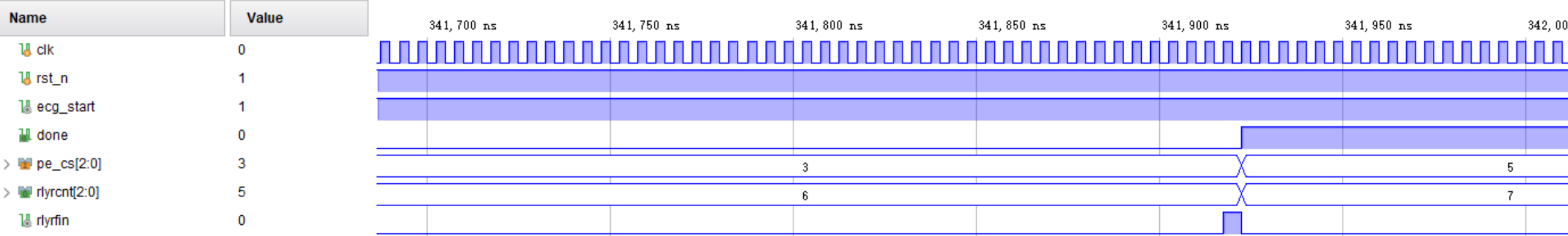


图5.10 硬件执行一次推理所需时间

加速器的计算状态仿真波形图如图5.11所示，其中cmp\_cs为计算控制模块的从状态机，可以反应当前PE状态是否处于计算状态。加速器在执行整型输入激活的卷积计算时，一轮连续的计算持续20个周期。但是SWU在每次计算完毕之后需要重新花费20个周期写入新的缓存数据，而BRAM的读取有2周期的延迟，所以每轮计算之间会等待3个周期。加速器在执行二值输入的计算时，加速器连续执行12个周期的计算，由于SWU中的数据包含多个输入通道可以反复用于多轮计算，且WEG的写入只需要耗费6个周期，所以中间只空闲等待1个周期。加速器在执行全连接层运算时，权重的数量远大于激活输入的数量，所以需要等待权重写入缓存，计算一个周期需要等待6个周期。根据以上的分析，结合加速器执行各种类型计算的时间，最终可得PE阵列进行计算时的工作效率为90.05%。

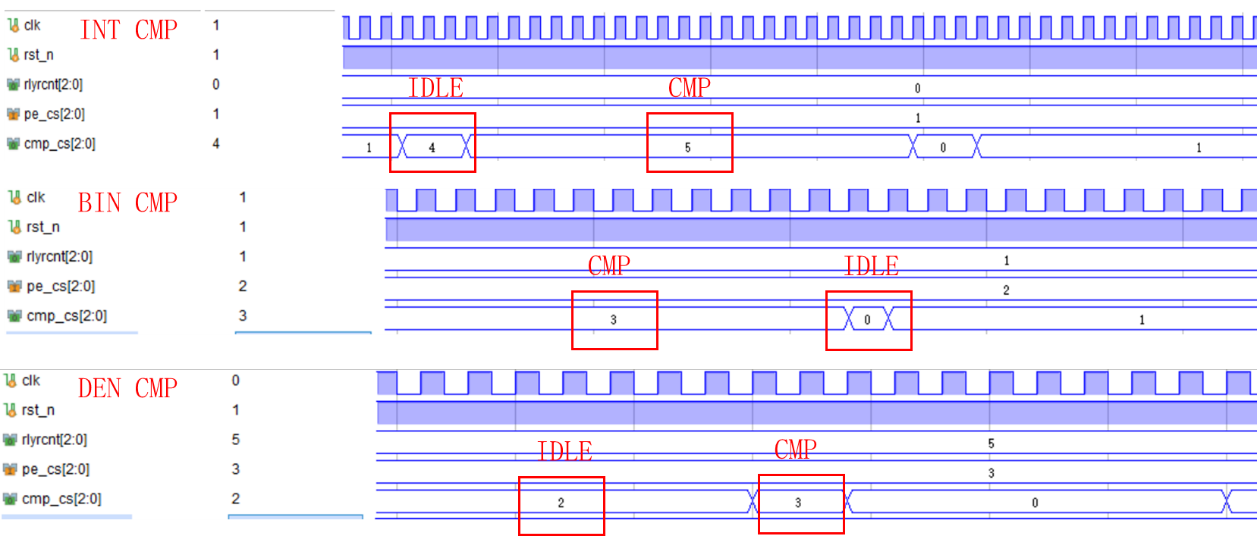


图5.11 加速器计算状态仿真波形图

由于激活和权重的缓存均采用双缓冲的策略，减少了PE阵列等待缓存写入的时间。同时激活的存储InOut Buffer采用乒乓操作，使得中间激活值可以无缝读写。单个PE单元的计算部分是无阻塞的流水线，可以连续的计算输入的数据。综合以上几点，加速器实现了实现PE的高效利用。在高效利用PE的同时，还堆叠了32个PE并行计算，最终使加速器实现了较高的吞吐率。

5.4 测试结果分析

本文最终的硬件性能和对标文献的对比如表5.3所示。ISNE2021和TCAS2021均是ECG信号检测分类一维卷积神经网络的专用硬件加速器，是本文主要对标的对象。而FPGA2017是一篇典型的基于BNN的大型卷积神经网络加速器，可以给同样基于BNN设计硬件加速器的本设计提供参考。

表5.3 基于BNN的ECG信号检测分类卷积神经网络加速器指标对比

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **参数** | **本文** | | **ISNE2021[28]** | **TCAS2021[29]** | | **FPGA2017[20]** |
| **验证平台** | **ZYNQ7035** | | **ZC706** | **ZC706** | | **ZC706** |
| **工作频率（MHz）** | **200** | | **200** | **200** | | **200** |
| **参数位宽** | **1bit** | | **16bit** | **16bit** | | **1bit** |
| **识别准确率（%）** | **92.83** | | **98.94** | **99.10** | | **88.1** |
| **吞吐率（GOPS）** | **51.6** | | **26.6** | **25.7** | | **2465.5** |
| **能效（GOPS/W）** | **146.2** | | **32.5** |  | | **210.7** |
| **数据集** | | **MIT-BIH Arrhythmia** | | | **CIFAR-10** | |

ISNE2021和TCAS2021均使用脉动阵列的架构，实现了一维卷积神经网络的加速器，通过计算单元之间的数据传递进行充分的数据复用，实现了较好的能效和吞吐率。相较于ISNE2021和TCAS2021,本设计将参数位宽量化到1bit，相比于16bit位宽的设计损失了<7%的准确率，但是相应的带来了更高的吞吐率和能效。得益于二值化，本设计的计算单元资源占用更少，因此可以堆叠更大的计算规模，也能更多的挖掘并行数据之间的可复用部分。同时本设计中的缓存单元均采用双缓冲的设计，减少了计算单元的等待时间，最终实现了较高的吞吐率。相应的，因为使用简化的计算单元设计，使得加速器单位算力的硬件开销更少，所以最终取得了更好的能效比。在嵌入式的ECG信号检测场景中，长时间的心律检测预警对能效的要求更高，因此本设计损失一定精度，实现高能效的检测具有一定的实用价值。

5.5 本章小节

本章首先介绍了加速器的验证平台和功能验证方案，然后将加速器系统在FPGA上实现，并利用VIVADO对其功耗、资源占用率进行评估。对实现后的加速器系统进行测试，保证软硬件一致性，得出硬件的ECG分类精度。最后将本文的设计和近年来的设计进行对比，最终得出本文的设计在对精度不敏感的嵌入式的应用场景下有着更高的能效比，具有一定的实用价值。

第六章 总结与展望

6.1 论文总结

本文提出了基于BNN的ECG检测卷积神经网络，并基于二值量化的特点设计了其专用的硬件加速器，分析了设计方案的算法和硬件性能，总结本文工作主要由以下几点：

（1）设计了基于BNN的ECG检测卷积神经网络，首先根据本文实现的检测分类目标设计了网络的基本块，并进行了设计空间探索，确定了最优的卷积神经网络结构。然后使用二值量化方法分段对网络的激活和权重进行二值化，并制定了定点化策略，降低了部署到硬件的精度损失。最终本网络在二值量化后精度达到了92.83%，内存占用量28.1KB，以一定的精度损失取得了相较于其他设计更低的计算量和内存占用。

（2）搭建了卷积神经网络的硬件加速器，包括控制模块、存储模块、缓存模块和计算模块。首先对加速器的设计空间进行了探索，确定了加速器的尺寸和工作数据流。然后设计了加速器的PE单元，并围绕PE单元搭建了控制模块、存储模块和缓存模块。完成加速器的RTL设计之后，进行了功能验证和功耗仿真。

（3）提出了基于二值量化数据特点的PE计算单元，使用同或实现了二值激活和输入的乘法操作，并通过添加较少额外硬件的方式使得PE单元对整形输入激活值卷积计算的支持。此外，通过激活和权重缓存单元的配合，实现了PE单元的较高利用率的同时，对激活和权重缓存单元中的参数也进行了多次复用。本设计的PE单元有着较少的硬件资源占用和较高的参数复用率，因此带来了更优的能效表现。

6.2 未来展望

本文设计了基于BNN的ECG检测卷积神经网络加速器，并有着较好的能效表现。虽然从算法准确率和硬件能效上取得了和预期相符合的结果，不过本论文的所做的工作仍然有改进的空间，以及设计存在一些局限性，主要有如下几点。

（1）虽然本文对卷积神经网络的权重和激活进行了二值量化的处理，极大程度的压缩网络参数，但仍然有进一步压缩的空间。其一是本文使用的网络通道数众多，可能存在一定的参数冗余。可以通过知识蒸馏等方法对网络进一步的压缩，获得更为轻量化的网络结构。其二是为了保证算法精度，对第一层输入的激活值依然保持了8bit的位宽。可以通过增加训练数据集、扩大网络规模、优化二值化策略等方法，进一步的将网络的第一层量化到更低的位宽，从而获得更小的算法内存占用。特别的，若能将网络第一层量化到二值，则可以在硬件上不必额外设计整型输入的支持模块，进一步提高硬件设计的能效。

（2）本文使用的数据集数量有限，难以保证算法的泛化性。可在日后的工作中，获取更多的有效数据集，从而可以增加算法泛化性的同时，也可以提高算法的精度表现。

（3）本文的设计在计算密集型的卷积运算中表现高效，但是在参数密集型的全连接运算中，PE阵列的存在较多的等待和闲置。可以通过改进PE阵列的设计，使其在全连接层的运算中能够适应参数密集型运算的特点，实现更高的运算效率。

以上内容可作为本课题日后的研究重点，作者相信对上述不足之处进行不断改进能够获得更加优异的性能。

由于作者能力有限，水平一般，本文难免存在一些纰漏与错误，恳请各位老师不吝赐教，予以批评和指正。

致谢

参考文献

1. 《中国心血管健康与疾病报告》编写组.《中国心血管健康与疾病报告2021》概述[J]. 中国心血管病研究, 2022, 20(7):20.
2. Organization W H . Noncommunicable diseases: country profiles 2011.[J]. Scand J Soc Med, 2014, 14(1):7-14.
3. A R J M , B U R A , C K M M , et al. Application of principal component analysis to ECG signals for automated diagnosis of cardiac health[J]. Expert Systems with Applications, 2012, 39( 14):11792-11800.
4. Daamouche A , Hamami L , Alajlan N , et al. A wavelet optimization approach for ECG signal classification[J]. Biomedical Signal Processing and Control, 2012, 7(4):342-349.
5. Hannun A Y , Rajpurkar P , Haghpanahi M , et al. Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network (vol 25, pg 65, 2019)[J]. Nature medicine, 2019(3):25.
6. Wei Y , Zhou J , Wang Y , et al. A Review of Algorithm & Hardware Design for AI-Based Biomedical Applications[J]. IEEE Transactions on Biomedical Circuits and Systems, 2020, PP(99):1-1.
7. FAEZIPOUR, M., SAEED, A., BULUSU , S. C., et al. A Patient-Adaptive Profiling Scheme for ECG Beat Classification[J]. IEEE transactions on information technology in biomedicine: A publication of the IEEE Engineering in Medicine and Biology Society,2010,14(5):1153-1165.
8. Banerjee S , Mitra M . Application of Cross Wavelet Transform for ECG Pattern Analysis and Classification[J]. IEEE Transactions on Instrumentation and Measurement, 2014, 63(2):326-333.
9. Desai U , Martis R J , Nayak C G , et al. Machine intelligent diagnosis of ECG for arrhythmia classification using DWT, ICA and SVM techniques[C]// 2015 Annual IEEE India Conference (INDICON). IEEE, 2016.
10. Taiyong, Li, Min, et al. ECG Classification Using Wavelet Packet Entropy and Random Forests[J]. Entropy, 2016.
11. Yildirim O . A Novel Wearable el wavelet sequence based on deep bidirectional LSTM network model for ECG signal classification[J]. Computers in Biology and Medicine, 2018(96-):96.
12. Hammoud I , Ramakrishnan I V , Djuric P . Classification of 12-lead ECGs Using Intra-Heartbeat Discrete-time Fourier Transform and Inter-Heartbeat Attention[C]// 2020 Computing in Cardiology Conference. 2020.
13. Hou B , Yang J , Wang P , et al. LSTM Based Auto-Encoder Model for ECG Arrhythmias Classification[J]. IEEE Transactions on Instrumentation and Measurement, 2019, PP(99):1-1.
14. Acharya U R , Oh S L , Hagiwara Y , et al. A deep convolutional neural network model to classify heartbeats[J]. Computers in Biology and Medicine, 2017, 89.
15. Li Z , Zhou D , Wan L , et al. Heartbeat classification using deep residual convolutional neural network from 2-lead electrocardiogram[J]. Journal of Electrocardiology, 2019, 58.
16. Lu J H, Liu D S, Hu A, et al. A low-cost and configurable hardware architecture of sparse 1-D CNN for ECG classification [C]//2022 IEEE 16th International Conference on Solid-State & Integrated Circuit Technology (ICSICT). 2022: 1-3.
17. Ribeiro H , Arnold A , Howard J P , et al. ECG-based real-time arrhythmia monitoring using quantized deep neural networks: A feasibility study[J]. Computers in Biology and Medicine, 2022, 143:105249-.
18. Wong D , Li Y , John D , et al. Resource and Energy Efficient Implementation of ECG Classifier using Binarized CNN for Edge AI Devices[C]// 2021 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2021.
19. Zhao R , Song W , Zhang W , et al. Accelerating Binarized Convolutional Neural Networks with Software-Programmable FPGAs[C]// Field Programmable Gate Arrays. ACM, 2017.
20. Zhou Y , Redkar S , Huang X . Deep learning binary neural network on an FPGA[C]// IEEE International Midwest Symposium on Circuits & Systems. IEEE, 2017.
21. Umuroglu Y , Fraser N J , Gambardella G , et al. FINN: A Framework for Fast, Scalable Binarized Neural Network Inference[J]. ACM, 2016.
22. Liang, Shuang, Yin, et al. FP-BNN: Binarized neural network on FPGA[J]. Neurocomputing, 2018.
23. Guo P , Ma H , Chen R , et al. A High-Efficiency FPGA-Based Accelerator for Binarized Neural Network[J]. Journal of Circuits, Systems and Computers, 2019.
24. Ghasemzadeh M , Samragh M , Koushanfar F . ReBNet: Residual Binarized Neural Network[J]. IEEE, 2018.
25. Wang E , Davis J J , Cheung P , et al. LUTNet: Rethinking Inference in FPGA Soft Logic[J]. IEEE, 2019.
26. Wang N, Zhou J, Dai G H, et al. Energy-efficient intelligent ECG monitoring for wearable devices [J]. IEEE Transactions on Biomedical Circuits and Systems, 2019, 13(5): 1112-1121.
27. Kiranyaz S , Ince T , Gabbouj M . Real-Time Patient-Specific ECG Classification by 1-D Convolutional Neural Networks[J]. IEEE Transactions on Biomedical Engineering, 2016, 63(3):664-675.
28. Wei L , Liu D , Lu J , et al. A low-cost Hardware Architecture of Convolutional Neural Network for ECG Classification[C]// 2021 9th International Symposium on Next Generation Electronics (ISNE). 2021.
29. Lu J , Liu D , Liu Z , et al. Efficient Hardware Architecture of Convolutional Neural Network for ECG Classification in Wearable Healthcare Device[J]. Circuits and Systems I: Regular Papers, IEEE Transactions on, 2021, PP(99):1-10.
30. Kumar A , Singh M . Ischemia detection using Isoelectric Energy Function[J]. Computers in Biology and Medicine, 2016.
31. Shen C P , Kao W C , Yang Y Y , et al. Detection of cardiac arrhythmia in electrocardiograms using adaptive feature extraction and modified support vector machines[J]. Expert Systems with Applications, 2012, 39(9):7845–7852.
32. Hubel D H , Wiesel T N . Receptive fields, binocular interaction and functional architecture in the cat's visual cortex.[J]. Journal of Physiology, 1962, 160(1):106-154.
33. Fukushima K . Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position[J]. Biological Cybernetics, 1980, 36(4):193-202.
34. Lecun Y , Bottou L . Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11):2278-2324.
35. Krizhevsky A , Sutskever I , Hinton G . ImageNet Classification with Deep Convolutional Neural Networks[J]. Advances in neural information processing systems, 2012, 25(2).
36. Ioffe S , Szegedy C . Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift[J]. JMLR.org, 2015.
37. Horowitz M . 1.1 Computing's energy problem (and what we can do about it)[C]// 2014 IEEE International Solid- State Circuits Conference (ISSCC). IEEE, 2014.
38. Courbariaux M , Bengio Y , David J P . BinaryConnect: Training Deep Neural Networks with binary weights during propagations[J]. MIT Press, 2015.
39. Courbariaux M , Hubara I , Soudry D , et al. Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1[J]. 2016.
40. Liu Z , Wu B , Luo W , et al. Bi-Real Net: Enhancing the Performance of 1-bit CNNs With Improved Representational Capability and Advanced Training Algorithm[J]. 2018.
41. Rastegari M, Ordonez V, Redmon J, et al. XNOR-Net: ImageNet classification using binary convolutional neural networks [C]//Leibe B, Matas J, Sebe N, et al. Computer vision - ECCV 2016. Cham: Springer, 2016: 525-542.
42. Blott M , Preusser T , Fraser N , et al. FINN-R: An End-to-End Deep-Learning Framework for Fast Exploration of Quantized Neural Networks[J]. ACM Transactions on Reconfigurable Technology and Systems, 2018, 11(3):1-23.
43. Andri R , Karunaratne G , Cavigelli L , et al. ChewBaccaNN: A Flexible 223 TOPS/W BNN Accelerator[J]. 2020.
44. Fu C , Zhu S , Su H , et al. Towards Fast and Energy-Efficient Binarized Neural Network Inference on FPGA[C]// the 2019 ACM/SIGDA International Symposium. ACM, 2019.
45. Kim T H , Shin J . A Resource-Efficient Inference Accelerator for Binary Convolutional Neural Networks[J]. Circuits and Systems II: Express Briefs, IEEE Transactions on, 2020, PP(99):1-1.
46. Lin J H , Xing T , Zhao R , et al. Binarized Convolutional Neural Networks with Separable Filters for Efficient Hardware Acceleration[C]// 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). IEEE, 2017.
47. Nakahara H , Fujii T , Sato S . A fully connected layer elimination for a binarizec convolutional neural network on an FPGA[C]// 2017 27th International Conference on Field Programmable Logic and Applications (FPL). IEEE, 2017.
48. Chi C C , Jiang J . Logic synthesis of binarized neural networks for efficient circuit implementation[C]// the International Conference. 2018.
49. Simonyan K , Zisserman A . Very Deep Convolutional Networks for Large-Scale Image Recognition[J]. Computer Science, 2014.
50. Xia Y , Xie Y . A Novel Wearable Electrocardiogram Classification System Using Convolutional Neural Networks and Active Learning[J]. IEEE Access, 2019:1-1.
51. Li P , Wang Y , He J , et al. High-Performance Personalized Heartbeat Classification Model for Long-Term ECG Signal[J]. IEEE Transactions on Biomedical Engineering, 2016:78-86.
52. Wong D , Li Y , John D , et al. Resource and Energy Efficient Implementation of ECG Classifier using Binarized CNN for Edge AI Devices[C]// 2021 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2021.
53. Becker J , Huebner M , Ullmann M . Power Estimation and Power Measurement of Xilinx Virtex FPGAs: Trade-Offs and Limitations[C]// 16th Symposium on Integrated Circuits and Systems Design, 2003. SBCCI 2003. Proceedings. IEEE, 2003.

攻读研究生硕士期间的成果