电 子 科 技 大 学

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

专业学位硕士学位论文

MASTER THESIS FOR PROFESSIONAL DEGREE

论文题目 **适用于神经形态硬件的缓存架构研究**

专业学位类别	电子信息
学　　号	**202022140425**
作者姓名	黄衍林
指导教师	游宏志　　副教授
学　　院	生命科学与技术学院

分类号 _____ 密级 公开

UDC [注1] _____

学 位 论 文

适用于神经形态硬件的缓存架构研究

黄衍林

指导教师 游宏志 副教授

电子科技大学 成 都

申请学位级别 硕士 专业学位类别 电子信息

专业学位领域

提交论文日期 2023 年 3 月 20 日 论文答辩日期 2023 年 5 月 25 日

学位授予单位和日期 电子科技大学 2023 年 6 月

答辩委员会主席 王玲

评阅人

Research of Cache Architecture for Neuromorphic Hardware

A Master Thesis Submitted to
University of Electronic Science and Technology of China

Discipline	**Electronic Information**
Student ID	**202022140425**
Author	**Huang Yanlin**
Supervisor	**A.P. You Hongzhi**
School	**School of Life Science and Technology**

摘 要

脉冲神经网络（Spiking Neural Network, SNN）是一种仿生神经网络，它的优点包括能够以更低的能量消耗实现更高的计算效率、更好地处理时空信息以及更适合进行基于突触可塑性的在线学习。这些优点使得 SNN 在许多领域中得到了广泛应用，例如图像处理、语音识别、机器人控制和神经科学研究等。

SNN 由于其事件驱动的计算方式，需要在专用的神经形态硬件上运行才能发挥其优势。然而，传统的神经形态硬件在芯片上通常只能存储少量的突触权重，这限制了能够在硬件上部署的 SNN 的大小。这是因为大部分的神经形态硬件将突触权重保存于片上存储，而片上存储容量较小，只能容纳有限个数的突触和神经元。如果采用低成本大容量的片外存储方案，则会因为片外存储带宽较低，影响芯片性能。

本文受到神经形态计算中神经元在编码信息时连续发放特性的启发，认为连续发放的脉冲能够反复触发同一批权重的累加。这种数据反复多次且批量地被访问的特性称为时空局域性。采用缓存架构可以利用时空局域性提高片内数据的复用率，提升硬件的存储性能。于是，本文将传统计算机体系结构中的缓存进行适用于神经形态硬件的改造，使其支持事件驱动的计算模式。本文的缓存架构使得同样容量的片上存储所能支持的 SNN 规模提升了 4 到 8 倍，推理的准确率损失在 1%以内。

此外，本文还对在缓存架构中实现在线学习进行了探索，提出了一种事件驱动的突触时间可塑性（Synapse Time Dependent Plastic，STDP）零开销惰性计算方案，不需要额外的周期进行学习。在实验正确复现了 STDP 的学习曲线。

关键词：脉冲神经网络，现场可编程门阵列，缓存，神经形态硬件

ABSTRACT

Spiking neural network (SNN) is a type of biomimetic neural network, which has the advantages of achieving higher computational efficiency with lower energy consumption, better processing of spatio-temporal information, and being more suitable for online learning based on synaptic plasticity. These advantages make SNN widely used in many fields, such as image processing, speech recognition, robot control, and neuroscience research.

Due to its event-driven computing mechanism, SNN needs to run on specific neuromorphic hardware to fully exploit its advantages. However, traditional neuromorphic hardware can only store a small amount of synaptic weights on the chip, limiting the size of SNN that can be deployed on hardware. This is because most neuromorphic hardware stores synaptic weights on-chip, and on-chip storage capacity is limited, only able to accommodate a limited number of synapses and neurons. If a low-cost large-capacity off-chip storage scheme is adopted, the performance of the chip may be affected due to the low off-chip storage bandwidth.

Inspired by the continuous firing characteristics of neurons in neuromorphic computing, this paper assumes that continuous spikes can repeatedly trigger the accumulation of the same batch of weights. This characteristic is known as spatiotemporal locality, involves accessing data in a repetitive and batch-wise manner. By leveraging this spatiotemporal locality, a cache architecture can be employed to enhance the reuse rate of on-chip data and improve hardware storage performance. Therefore, this paper transforms the cache in the traditional computer architecture to fit neuromorphic hardware, make it supports event-driven computing mode. The cache architecture increases the SNN scale that can be supported by on-chip storage with the same capacity by 4 to 8 times, with the loss of inference accuracy within 1%.

In addition, we also explore the realization of online learning in the cache architecture, and propose an event-driven lazy computing scheme of synapse time dependent plastic (STDP) with zero overhead, which does not require additional learning epoch. The learning curve of STDP can be correctly reproduced in the experiment.

Keywords: SNN, FPGA, Cache, Neuromorphic Hardware

目 录

图目录

表目录

缩略词表

英文缩写	英文全称	中文全称
AER	Address Event Representation	事件地址表示法
ANN	Artificial Neural Network	人工神经网络
ASIC	Application Specific Integrated Circuit	专用集成电路
BRAM	Block RAM	块随机访问存储器
CANN	Continuous Attractor Neural Networks	连续吸引子网络
CPU	Central Processing Unit	中央处理器
CSR	Control and State Register	控制状态寄存器
DRAM	Dynamic RAM	动态随机访问存储器
DSP	Digital Signal Process	数字信号处理
DVS	Dynamic Vision Sensor	动态视觉传感器
FPGA	Field Programable Gate Array	现场可编程门阵列
GPU	Graphic Processing Unit	图形处理单元
ID	Identification	标识
LIF	Leaky Integrate and Fire	整合发放
LRU	Least Recently Used	最少最近使用
LTD	Long Term Depression	长时程抑制
LTP	Long Term Potentiation	长时程增强
MNIST	Modified National Institute of Standards and Technology	混合的国家标准和技术研究所数据库
RAM	Random Access Memory	随机访问存储器
SNN	Spiking Neural Network	脉冲神经网络
SRAM	Static RAM	静态随机访问存储器
STDP	Synapse Time Dependent Plastic	突触时间可塑性

第一章 绪论

1.1 研究背景及意义

脉冲神经网络（Spiking Neural Network, SNN）是受生物神经系统启发而提出的一种人工神经网络（Artificial Neural Network，ANN），它是计算神经科学领域广泛使用的网络模型。SNN 可基于突触可塑性等在线学习算法训练调整，在现实世界中适用于认知决策等任务。SNN 由于采用了事件驱动的计算方式，与传统的 ANN 相比具有更少的计算量和更快的响应速度等优点。然而事件驱动的计算方式必须依赖于专用的集成电路实现才能发挥其优势，这类专用于 SNN 推理或训练的集成电路称为神经形态芯片。

在此背景下，许多研究提出了自己的神经形态硬件架构，并在专用集成电路（Application Specific Integrated Circuit ，ASIC）或者现场可编程门阵列（Field Programable Gate Array，FPGA）中进行了实现。神经形态芯片主要在集成电路上实现仿生的突触和神经元的结构。这类电路系统一般通过事件地址表示法（Address Event Representation,，AER）[1]进行脉冲信号的传递。然而大部分神经形态硬件所能支持的突触、神经元数量有限，只能运行较小规模的 SNN，这是由于这些芯片使用片上存储权重。片上存储可以提供非常快速的访问速度，并且具有低功耗的能力，对于实现高效的神经形态计算来说非常重要。然而，片上存储的容量有限，这也限制了神经形态芯片能够支持的模型的大小。

硬件上的随机访问存储器（Random Access Memory，RAM）主要有静态随机存储器（Static RAM，SRAM）和动态随机存储器（Dynamic RAM，DRAM）两种类型[2]。其中 SRAM 支持快速的随机读写，但容量小造价贵。DRAM 具有容量大成本低的优点，但是访存速度慢，只适合按顺序的连续读写。片上存储器一般用 SRAM 制造，集成在芯片内部，提供高速的片上带宽；片外存储器一般使用独立的 DRAM 芯片，为系统提供较大的存储空间。所以本文在后续章节使用 SRAM 代指片上存储器，DRAM 代指片外存储器。

使用 SRAM 存放全部 SNN 参数的神经形态芯片计算能力虽然很强，但是在实际应用中，由于支持的模型规模太小，无法应对复杂的认知任务。如果要支持大规模的 SNN，则必须将参数置于大容量的 DRAM 中。但是这会导致芯片的性能受限于 DRAM 较低的存取速度，无法达到实时处理的效果。此外，神经形态硬件的主要任务是将网络模型映射到芯片的各个核心上，并减少核心之间的通信成本[3]。如果神经元需要从核心外的部件获取权重，这还会造成核心之间的通信拥塞。

因为基于 DRAM 的方案的性能瓶颈客观存在，所以该方案的研究较少。大多数工作的研究内容是基于 SRAM 实现小规模 SNN。虽然大规模的 SNN 理论上已经可行[4]，但是要真正落地 SNN 的应用，还需要硬件上的支持。所以，设计一款支持大规模 SNN 的神经形态硬件是一个迫切的需求。

本文受到生物神经元在活动时连续发放现象的启发，经过实验证明在 SNN 中也存在神经元连续发放编码信息的现象。本文假设这种现象在 SNN 事件驱动的计算方式下能够带来很强的数据时空局域性。在传统计算机体系结构领域，缓存（指 Cache 而非 Buffer）能够利用时空局域性从而提升性能。于是本文在神经形态芯片中引入了缓存架构，并进行定制化的设计，使其适用于事件驱动的计算方式。由于缓存架构的引入，本文的神经形态硬件可以用更少的 SRAM 完成更大规模的 SNN 计算，在成本和性能之间取得较好的平衡。

除此之外，对于智能系统来说，在运行时不断增加知识库和适应不断变化的环境的能力至关重要[5]。为了应对愈加复杂的任务，神经形态硬件往往会支持在线学习功能。人脑是一个不断学习的系统，随时根据现实环境的变化做出决策上的调整。在线学习类似，能够实现模型的动态更新，适应实时变化的环境。将在线学习应用于神经形态硬件上，不仅能够使模型更加准确地应对复杂的认知任务，而且还能够在硬件资源受限的情况下，快速调整模型以适应新任务，能极大提高模型的认知决策能力。但是在数字神经形态硬件中，实现在线学习所需要的权重更新会给硬件带来极大的访存开销，不能合理处理的情况下，学习效率较低。在数字神经形态硬件中实现在线学习不仅是重要的功能点，也是一个难点。

大多数的神经形态硬件采用突触时间可塑性（Synapse Time Dependent Plastic，STDP）[6,7]作为在线学习的实现方案。基于 STDP 实现的在线学习在神经形态硬件的设计中具有广泛的应用前景，可以在实际应用中更好地满足不同的任务需求。

1.2 国内外研究现状

神经形态硬件是一种仿生学的技术，可以实现类似于大脑的神经元和突触结构的功能，以实现高效的计算。近年来，基于神经形态硬件的研究逐渐受到越来越多的关注。同时，在神经形态计算领域，软件算法端的 SNN 模型和在线学习训练方法的研究成果为该领域的发展提供了新的思路和方法。本章将从 SNN 算法和应用开始，对神经形态计算领域进行综述，之后介绍神经形态硬件的相关研究。

神经形态硬件的相关研究根据存储架构的不同，可将相关工作分为完全基于片上存储的架构和使用了片外存储的架构。其中，在使用了片外存储的架构中，根据是否采用了缓存架构，可再细分为两类。本章节对这三类架构的神经形态硬件进

行介绍并总结。通过这些研究现状的介绍，我们将更好地了了解不同存储架构的优点和局限性。

1.2.1 脉冲神经网络应用与算法

神经形态硬件的设计应当以应用为导向，速度和灵活性不可兼得[8]。不同的应用适用于 SNN 的不同特性，需要结合实际具体分析。

SNN 具有实时性的特点，十分适合机器人的控制。Hulea M[9]使用 SNN 实现了机器手指的控制。提出了能够自适应响应压力传感器的 SNN 模型，使得机械手具有拟人化的特性。

SNN 天然适合处理具有高动态范围的事件传感器数据。动态视觉传感器（Dynamic Vision Sensor，DVS）具有极高的时间分辨率，配合 SNN 能对快速运动的物体进行识别和定位。Falanga D[10]在无人机上安装了 DVS 并使用 SNN 控制无人机在飞行中避障。相比传统方案几十毫秒的延迟，该方案能够在 3.5 毫秒的时间内检测并躲避突然出现的多个大小不同的障碍物。

SNN 部署到神经形态硬件后具有超低功耗的特性，可以长时间工作，因而十分适用于生物医学类的可穿戴设备。Sharifshazileh M[11]设计了一个可穿戴的 SNN 软硬件系统，用于癫痫病患者的颅内脑电图监测。该系统可以可靠地捕捉脑电中出现的高频振荡，达到了最高的准确性、灵敏度和特异性。

SNN 虽然具有推理上的各种优势，但是它的训练一直以来是一个难点。因为 SNN 中的脉冲神经元的激活函数不是线性函数，无法直接使用深度学习中基于梯度下降的算法进行训练。SNN 作为一种包含时序状态的网络，一般的有监督训练方法采用时间梯度反向传播（Back Propagation Through Time，BPTT）算法进行训练，但是这种方式的训练效率极低。为了解决这个问题，Bellec G[12]提出了称为 e-prop 的学习方法，该方法不仅解决了 SNN 中误差梯度回传的问题，还能对具有循环连接的 SNN（Recurrent SNN，RSNN）模型进行低成本的在线学习。

类似的工作还有 Zenke F[13]将 BPTT 与实时循环学习（Real Time Recurrent Learning，RTRL）[14]相结合，提出了一个适用于神经形态硬件的 RSNN 计算、训练的数学框架。Neftci E O[15]提出了多种使用替代导数解决 SNN 训练中激活函数和脉冲非线性的问题，并且这些算法适用于局部学习。

从以上关于 SNN 训练的研究可以看出，SNN 训练十分关注数据的局域性，尽量避免类似梯度回传这种开销极大的数据更新。除了这类基于反向传播进行改进的训练方案，另一些工作从生理机制启发，提出了具有生物上可解释性的方案，同样具有能够进行局部学习的优点。

Zhang T[16]受到蝌蚪视皮层中突触活动长距离跨层传播现象[17]的启发，提出了具有可解释性的误差跨层传播训练算法，在分类任务上达到了相比其它突触可塑性算法的最好效果。

目前研究较为充分的是基于脉冲的突触可塑性训练方式。Taherkhani A[18]将基于脉冲的学习算法根据触发的规则不同分为了 6 类。其中，最为广泛使用的是基于突触前后脉冲对的 STDP 方案。STDP 一般情况下是无监督的学习，Ponulak F[19]提出的 ReSuMe 算法，将 STDP 与有监督的方式相结合，实现了高效的在线学习。

1.2.2 基于片上存储的神经形态硬件

使用片上 SRAM 存放神经元和突触的所有参数，是大多数神经形态硬件采用的做法。这类神经形态硬件往往采用同构多核心的设计，通过片上网络[20]进行核心间通信，从而构建大型神经形态系统。这种架构的代表工作可参考 Intel 公司的 Loihi 芯片[21]。Loihi 芯片中共包含 128 个核心，每个核心至多支持 1024 个神经元和 128 KB 的突触参数。多核架构中，神经元之间连接信息存储对核心之间的路由资源消耗极大，Loihi 中采用了轴突的模型解决神经元通信的问题，每个核心至多支持 4096 个轴突的输入输出。Loihi 支持 STDP 和强化学习等多种在线学习机制，提出了时间和事件混合驱动的方案，基于事件驱动进行推理，然后在独立的时间驱动的周期里执行学习算法。Loihi 具有易于使用的软件接口，方便研究者进行 SNN 实验。例如，Stewart K[22]基于 Loihi 实现了基于 DVS 的手势识别的 SNN 模型，成功验证了一种误差触发的在线学习算法的有效性；Viale A[23]在 Loihi 上实现了低功耗的自动驾驶系统，可以控制小车灵活地避障。

与 Loihi 类似的架构工作还有 Merolla P A[24]等人设计的可容纳百万神经元的神经形态芯片。该芯片同样采用多核心架构，一共包含 4096 个核心。通过片上网络的拓扑联系，核心之间能够构建任意大小的类似大脑皮层的突触连接。在实际任务中，能达到 30 帧每秒的图像处理能力，可用于多目标识别与分类。

在神经形态硬件中，实现在线学习功能往往要求芯片的 SRAM 中有完的整权重矩阵。Kim G[25]设计了专用于手写数字数据库（Modified National Institute of Standards and Technology，MNIST）识别的 SNN 芯片。一般的 RAM 只能按行进行连续读取，而该芯片使用了可转置 RAM 的工艺，使得权重矩阵既可以按行也能按列进行连续读取。这种特殊的 RAM 有帮助于实现高效的 STDP。不过这种方式并不适合 FPGA 的实现，因为 FPGA 中的 SRAM 仅支持按行读取。

神经形态硬件根据 SNN 类型的不同，其设计的难点也不相同。其中，用于卷积类型 SNN 的芯片[26]，其难点在于核心之间的事件传递，对权重存储需求不高。具有片上存储瓶颈的主要是用于全连接类型的 SNN 芯片。

连续吸引子神经网络（Continuous Attractor Neural Networks, CANN）的主要部分由一群神经元通过循环连接组成，全连接类型的权重幅值遵循高斯分布。赵鲲鹏[27]设计了用于 CANN 模型仿真的多核心神经形态处理器。CANN 中为了支持丰富的突触动力学特性，对突触的参数量有更大的要求。该工作中提出了新型的突触循环连接算法，在使用更少的 SRAM 的情况下，可支持的神经元数量达到原算法实现的 8 倍。

用于全连接类型 SNN 的神经形态芯片的结构较为同质化，一般包含一块专用于突触参数存储的 SRAM，一块专用于神经元参数存储的 SRAM，此外还有神经元单元，学习单元。突触 SRAM 的资源消耗为神经元 SRAM 资源消耗的二次方。Frenkel C[28]和徐志康[29]分别提出了支持在线学习的神经形态处理器，他们的处理器所能支持的网络大小相同，均为 256×256 大小的全连接层，并且均使用了 32 KB 的 SRAM 存放突触权重。

Li S[30]提出了一种快速高能效的 SNN 处理器，其神经元计算方式可以自适应切换为时间驱动和事件驱动中的任意模式。它的每个处理单元有独立的本地存储，通过片上网络可以共享，处理单元可以将非本地的事件发送到其它核心进行处理。该处理器可以负载规模为 784×200×100×10 的全连接 SNN。

1.2.3 基于片外存储的神经形态硬件

为了解决 SRAM 不足以容纳大规模 SNN 参数的问题，一些研究工作采用基于 DRAM 存储的参数的方案设计神经形态硬件。DRAM 的容量足够大，存储参数不再是问题，但权重需由 DRAM 搬运至 SRAM 中才能进行计算。这类设计的关键在于如何在 DRAM 带宽的瓶颈的限制下优化数据搬运的性能。

1.2.3.1 非缓存架构

由于 SNN 事件驱动的特性，不需要所有的权重行参与计算，使用权重压缩的方案能够有效减少数据的搬运量。Guo S[31]利用了 SNN 脉冲的稀疏性，提出了一种基于脉动阵列的软硬件协同优化的 SNN 加速器。在进行计算之前，加速器会根据脉冲信息对 DRAM 中的权重进行压缩，只有有效脉冲所对应的权重行会被压缩编码并送入 SRAM 中。这种权重压缩方案提高了 SRAM 的利用率，仅用 256 KB 的 SRAM 即可负载参数量为 1 MB 左右的网络。

Panchapakesan S 设计的 SyncNN[32]采用了权重预取的方式对硬件系统进行优化。对于全连接层，该芯片将预取片上缓冲所能存放的最大数量的权重矩阵行数。如果脉冲输入能从预取的数据中访问权重，则直接从片上缓冲读取权重，否则将从 DRAM 中加载权重。相比于权重压缩的方案，这种权重预取的方式对带宽的利用率较低，预取的权重可能是无用的，因为它对应的神经元不一定发放。但是权重预取可以在脉冲到来之前完成，对于单个步长的推理来说，有更快的响应速度。

在 Cassidy A S[33]设计的神经形态架构中，则采用了更加直接的设计。既然片上 SRAM 资源较少，而片外 DRAM 速度慢，那么就折中选择片外的 SRAM 作为存储器。对于常见的 SNN 模型的参数量，DRAM 的容量冗余较大，采用片外 SRAM 而获得带宽的提升是一种可行的方案。但是 SRAM 始终成本较 DRAM 昂贵，在实际应用中，对于大模型人们更加倾向于片外 DRAM 和片上 SRAM 的组合。

浙江大学提出的达尔文芯片[34]是一个高能效的可配置神经形态处理器。芯片设计了两套存储体系，片内和片外存储分工合作。DRAM 中存储了 SNN 的拓扑信息和突触的权重和延迟信息等参数，另外还有一个地址阵列用于神经元地址的快速寻址。在芯片内部，一共有 8 个物理神经元，每个神经元配备一个独立的 SRAM，可以存储 16 KB 的参数。神经元通过时分复用的方式完成神经形态计算，所需权重直接从 DRAM 中读取，在片上只保留了权重和，当权重和达到突触延迟对应的时间戳，则将权重发送至神经元进行计算。

1.2.3.2 缓存架构

不管是采用了权重压缩还是权重预取，对于每个推理步长，都需要重新从 DRAM 中读取权重。但如果可以把在下个步长中会使用到权重在片上 SRAM 中保留，那么就能够减少下个步长的数据搬运量。这就是缓存在神经形态硬件中所解决的问题。

为了解决神经形态硬件的访存功耗，Saha S[35]在架构中加入了缓存的设计，减少了芯片对 DRAM 的访问次数。他认为神经形态硬件缓存相比于传统缓存的不同之处在于，神经形态系统中具有先验信息，即当一批事件是已知的，那么它们接下来的需要访问的权重范围也是确定的。该设计中使用队列存储输入的事件，累积一定个数的事件后，根据先验信息决定缓存中的权重是否需要替换。不过 SNN 中发放率较低的层级可能无法产生足够多的事件用于先验信息的累计，该设计为此设计了发放率的阈值，低于阈值的事件则通过旁路直接从 DRAM 中取权重。相比于没有缓存架构的设计，该设计成功减少了 12%-44%的访存功耗。

Minitaur[36]是一个基于 FPGA 平台的低功耗高性能的事件驱动 SNN 加速器。它包含 32 个可并行计算的核心，所有核心由控制单元统一调度。每个核心内部包含 2 MB 的神经元状态缓存、8 MB 的突触权重缓存。整个芯片可容纳 9 千个神经元，能够做到每秒 1900 万次突触电流累加的算力。单位容量的突触缓存可以容纳 5 倍大小的突触权重。该设计中的缓存对神经形态系统定制设计了缓存替换方案：记录每个缓存行的缺失次数，当容量不足时替换连续缺失多次的缓存行。

1.2.4 研究现状总结

在本文的分类中，神经形态硬件的存储架构包括基于片上存储、基于非缓存的片外存储和基于缓存的片外存储三种类型。其中，基于片上存储的优势在于它可以提高计算速度，并且可以较容易地实现在线学习算法。然而，片上存储的网络大小受到 SRAM 容量的限制，因此难以部署较大的网络。

相比之下，基于非缓存的片外存储可以部署较大的网络，但是受到存储器带宽的限制，导致计算速度无法与片上存储相媲美。为了解决这个问题，各项工作提出了自己的数据搬运优化方案。这些方案中可以发现，缓存的使用能够最大程度地利用带宽和片上存储资源，是一种较为优秀的解决方案。但是现有的缓存方案所采用的缓存策略较为粗糙或者不够灵活，没有完全发掘缓存架构在神经形态计算中的潜能，并且只能用于仅有推理功能的场景。如何用低资源（DRAM 和少量 SRAM）实现高性能的神经形态计算是待解决的问题[37]。

1.3 研究内容

为了解决神经形态硬件的在存储方面的困境，本文提出了一种基于缓存的神经形态硬件架构。该缓存架构能够充分利用神经元连续发放带来的时空局域性，极大减少对片外的访存，达到加速推理和减少片上存储成本的效果。本文的主要贡献有：

（1）提出了适用事件驱动计算模式的缓存架构，使得缓存能够被应用在神经形态硬件中。

（2）提出了零开销的 STDP 惰性计算方案，可高效地实现在线学习。

本文最终的成果是在FPGA上实现了一款神经形态芯片，该芯片能够部署SNN模型，运行推理和学习。芯片的完整开发过程中包含以下内容：

（1）通过实验验证了神经元连续发放现象具有时空局域性，论证了基于该特性设计的缓存架构的可行性。

（2）设计芯片的整体架构及其子模块，并充分仿真测试。

（3）通过实验验证芯片推理和在线学习功能的正确性，并分析缓存架构的效用。

1.4 论文章节安排

本文章节安排如下：

第一章：绪论。介绍了神经形态硬件的研究意义及现状。分析了多种相关工作的创新点及其不足之处。介绍了研究的主要内容。

第二章：神经形态计算基础与缓存原理。此章为论文所在领域的基础知识。神经形态计算基础包含了 SNN、硬件架构、在线学习算法的介绍。此外，论文将缓存应用在神经形态硬件上，所以在基础知识中对缓存有较为详细的介绍。

第三章：系统架构设计。为了更清晰地了解整个系统的设计，该章节首先对完整架构进行概述。之后指出本设计的创新点：使用缓存架构优化神经形态硬件；在芯片上实现在线学习功能。后续的每个小章节是对系统中子模块的细节介绍。子模块均围绕创新点进行设计。

第四章：实验结果与分析。本章给出了神经形态硬件在实际运行 SNN 的实验中的结果。实验分为推理实验和学习实验，逐个验证硬件功能的正确性。在分析中与相关工作进行参数对比，展示本设计的优势。

第二章 神经形态计算基础与缓存原理

神经形态计算这一领域主要研究类脑神经网络的模型设计及其对应的神经形态硬件设备。本文的工作属于神经形态计算，涉及从 SNN 模型的计算到集成电路的硬件实现，在本章中将介绍与之相关的基础知识。

2.1 脉冲神经网络

SNN 是一种基于脉冲编码的神经网络模型，它模拟了生物神经元之间的信息传递方式，如图 2-1 所示。在 SNN 中，神经元在接收到输入信号后，会以脉冲的形式向下一个神经元传递信息。这些脉冲被认为是非常短暂的电信号，在时间上是离散的，而且它们之间的传递是基于时间的。与传统的人工神经网络不同，SNN 的处理方式是基于事件驱动的。它只有在输入信号到达时才会执行计算，而且这些计算是基于神经元之间的脉冲传递时间的。在 SNN 中，神经元的行为是通过其自身的膜电位水平进行调节。神经元会对输入信号进行整合和放大，然后产生输出脉冲。这些脉冲会随着时间的推移而逐渐减弱，脉冲的传递通过突触实现。SNN 被广泛应用于模式识别、控制、感知和决策等领域。它能够有效地处理时序数据，并且具有更好的计算效率和更低的功耗，因为它只在必要时才会执行计算。同时，SNN 的突触还具有可塑性，可以自适应地调整自身的参数以适应不同的任务。

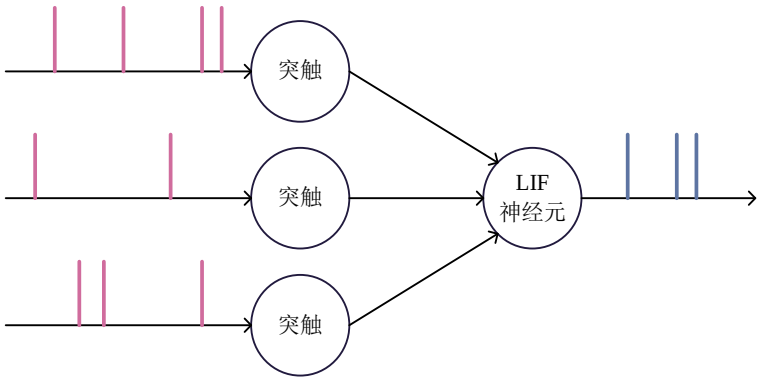

图 2-1 脉冲神经网络

用于 SNN 的神经元模型有很多种，其中最为流行的是整合发放（Leaky Integrate and Fire，LIF）神经元模型[38]，其数学模型如式(2-1)所示。

$$U(t) = (1-\alpha)U(t-1) + \alpha I(t) \tag{2-1}$$

式中，U为某时刻的神经元膜电位。$\alpha = h/\tau$，τ是神经元的时间常数，h为时间步长。$I(t)$为当前时刻神经元所整合的电流总和。当膜电位U达到了阈值U_{th}后，神经元会发放产生脉冲$S(t)$，见式(2-2)，并将膜电位清零。

$$S(t) = \begin{cases} 1, U(t) \geq U_{th} \\ 0, U(t) < U_{th} \end{cases} \tag{2-2}$$

LIF 神经元表达式比较简单，其参数易于调整。因此很容易在定制的硬件设备中实现。因为只有在神经元电位超过阈值时才会产生输出脉冲。这意味着在处理大量输入信号时，LIF 神经元能显著减少计算资源的消耗，从而提高 SNN 计算效率。

2.2 突触时间可塑性与在线学习

STDP 是一种神经系统中连接强度的可塑性现象，指的是神经元之间突触强度会根据突触前后神经元脉冲发放情况进行增强或者抑制的现象。突触可塑性被认为是大脑中学习和记忆的重要机制之一，能够帮助神经元根据它们的活动情况来调整它们之间的连接强度，从而实现信息的存储和提取。生理实验中已经观察到不少于 20 种的 STDP 规则[6]。在神经网络中较为常见的是基于脉冲对的规则[25,39]和基于迹的规则[21]。基于迹的 STDP 虽然能降低其在普通计算机上的计算成本，但它并不适合事件驱动的架构。本文采用的是基于前后脉冲对的 STDP 规则，如图 2-2 所示。

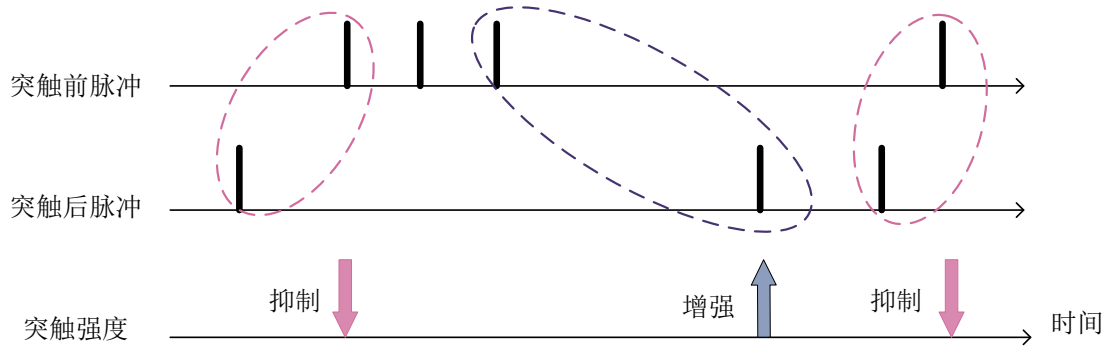

图 2-2 基于脉冲对的 STDP 规则

当一个突触的后神经元在前神经元之前发放，那么突触的权重将被抑制，这种现象称为长时程抑制（Long Term Depression，LTD）。如果后神经元在前神经元之后发放，说明后神经元的发放与该突触有关，突触的权重需要被增强，这种现象叫做长时程增强（Long Term Potentiation，LTP）。突触可塑性的数学模型如式(2-3)所示。

$$\Delta W = \begin{cases} A^{+}e^{-|\Delta t|/\tau_{+}}, t_{pre} \le t_{post} \\ A^{-}e^{-|\Delta t|/\tau_{-}}, t_{pre} > t_{post} \end{cases} \tag{2-3}$$

其中$\Delta t = t_{post} - t_{pre}$，表示两个突触前后脉冲之间的时间差。$A$是突触强度的变化量，$\tau$是和神经元时间常数有关的常数。当$\Delta t$大于等于 0 时，发生 LTP，突触权重被增强。当Δt小于 0 时，发生 LTD，突触权重被抑制。增强或抑制的强度随着的增加而减弱，意味着在时间上越相邻的两个脉冲队能产生的增强或抑制效果更好。

在线学习则是指在神经网络中实时更新神经元之间的连接权重，从而使网络能够根据实时输入的数据进行自适应学习。SNN 中在线学习通常需要实现 STDP 机制，使得网络能够对输入数据的时间顺序进行学习和适应。这种机制可以让神经网络实现类似于生物神经元的连续编码，从而更好地模拟大脑中的信息处理过程。

与深度学习中常见的反向传播学习算法相比，在线学习更加具有潜力。因为在线学习具有生物上的可解释性。类似 STDP 的在线学习方式具有无监督、计算局域性的优点。如何利用无监督局部学习实现实用的生物启发智能是一个有待探索的领域。

2.3 神经形态硬件

神经形态硬件是一种基于神经科学原理设计的芯片，旨在模拟人类大脑神经元之间的连接和信息传递方式，以高效地方式实现 SNN 在硬件上的部署。它与传统的基于冯诺依曼体系结构的计算机不同，后者的架构使用 DRAM 统一存储指令和数据，并使用中央处理器（Central Processing Unit，CPU）进行计算。相比之下，神经形态硬件使用类似于人脑神经元之间的连接方式来存储和处理数据，每个计算单元拥有独立的存储和计算逻辑，各个核心之间分布式并行计算。如图 2-3。

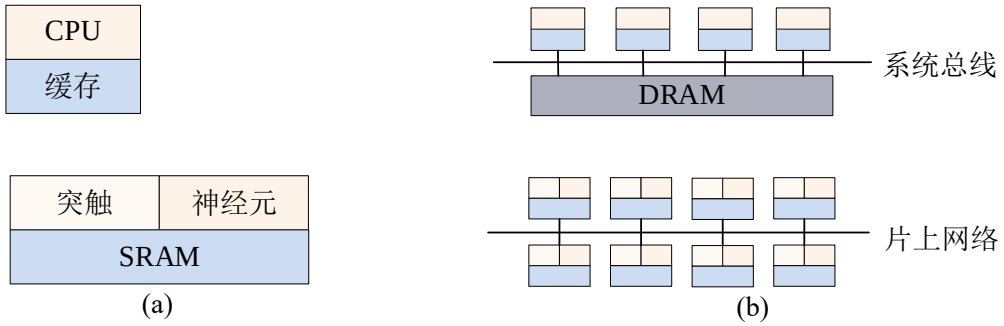

图 2-3 冯诺伊曼架构与神经形态架构对比图。(a)计算单元；(b)系统结构

在冯诺伊曼架构中，最小的计算单元是 CPU，多个 CPU 核心通过系统总线访问主主存储器进行数据交互，它的计算和存储是解耦的。而神经形态架构往往是分布式

的，没有统一的主存储器，SNN 的参数独立存储在每个核心的 SRAM 中，各个核心之间通过片上网络进行数据的交互。

神经形态硬件根据其硬件材料的不同可以分为不同的种类。一类是基于模拟电路的芯片[40,41]，通过模拟电路元件的组合，使其电气特性具有突触和神经元动力学特性的效果。一类是基于数模混合电路实现[42-44]。通用性较强的神经形态硬件一般是基于数字电路实现。数字电路的神经形态硬件广泛使用 FPGA 作为实现的基础[45,46]。FPGA 的可编程特性可以快速地实现完全不同的网络拓扑、模型和算法。然而如果要实现小型低功耗的系统，那么专用集成电路 ASIC 实现 SNN 会是一个更好的做法[47]。

基于事件驱动的神经形态硬件中常用 AER 协议进行通信。在 AER 中，每个神经元都被赋予了一个唯一的地址。当一个神经元产生输出脉冲时，它会发送一个事件消息，这个事件消息中除了必要的该神经元的 AER 地址，还可以自定义其它信息字段，具有灵活的表达方式。其他神经元可以通过接收和解码这些事件消息，实现与其他神经元的连接和信息传递。神经形态硬件和 AER 协议结合使用，可以实现高效的神经形态计算，并且在处理大规模神经网络时具有非常大的优势。

神经形态系统中，SNN 的所有神经元被映射到各个核心当中，而突触连接则以路由表的形式保存。当某个核心的神经元发放，它会产生一个 AER 包送入片上网络，片上网络根据网络的拓扑信息找到目的神经元所在的核心。当目的核心接收到代表脉冲事件的 AER 网络包则触发该神经元的推理计算，否则处于空闲状态。这种事件驱动的并行计算不需要对全局信息进行同步，反而 CPU 中如果多核心并行计算，为了维护共享内存中数据的一致性，需要付出较大的代价用于同步。

2.4 缓存原理

计算机设计早期使用的冯诺伊曼架构时，CPU 是直接访问 DRAM 的。但是随着后期集成电路技术的进步，CPU 的计算速度越来越快，造成 DRAM 的访问速度无法跟上 CPU 的需求。于是人们通过在 CPU 和 DRAM 之间插入缓存解决了这个问题。缓存是一种高速的存储器，使用 SRAM 制造，集成在芯片内部。缓存中存储了最近访问过的数据和指令，如果 CPU 所需要的数据在缓存中存在，那么就可以快速读取数据，无需花时间从 DRAM 中读取，从而提高计算机的性能。

当 CPU 需要访问内存中的数据时，它首先会检查缓存中是否存在所需的数据。如果数据在缓存中，则称为“命中”，CPU 可以直接从缓存中读取数据，这个过程非常地快速。如果缓存中不存在所需的数据，则称为“缺失”，此时 CPU 需要访问内存才能获取数据，并将数据搬运到缓存中，以便在未来提高该数据的命中率。

缓存采用局域性原理来提高访问速度。局域性原理包括时间局域性和空间局域性。时间局域性指的是在一段时间内，CPU 访问的数据往往是相同的，所以历史上访问过的数据，在未来有更大的概率会被访问。空间局域性指的是 CPU 访问的数据通常是相邻的，当前访问数据的临近数据在未来有更大的概率会被访问。

缓存越大命中率越高，能带来的性能提升越大。但是常见的缓存并不会被制造得太大。在最常见的三级缓存体系中，最大的第三级缓存只有 10 MB 数量级的容量。首先这是因为大容量缓存的地址更多，译码逻辑更加复杂，导致它的访问速度变慢，无法满足需求，其次 SRAM 造价较高，缓存的容量是在平衡成本后的结果。

缓存通过地址映射的方式，将缓存中的存储空间映射为主存储器的存储空间。3 种常见的地址映射方式如下：

（1）直接映射：每个主存的地址唯一对应一个缓存地址，是一个满射。这种映射方式的优点是原理简单易于实现，但是容易出现地址冲突，多个主存地址竞争一个缓存地址，导致缓存替换不充分。

（2）全相联映射：允许任何主存地址映射到缓存的任何位置。这种映射方式的优点是能最大程度地使用缓存空间。缺点是地址分配速度较慢，需要进行全局的搜索。

（3）组相联映射：将缓存划分为多个组，使用直接映射和全相联映射的结合方式，在组间使用直接映射，组内使用全相联映射。每个组包含多个缓存行，其中每一行存储主存中一段连续的数据。这种映射方式在提高缓存效率的同时也能保持较高的检索速度。

实践中较为常见的是组相联映射，其中每个组内的条目数量（又称为路）的设置和资源、效率的取舍有关，比较常用的是 4 路或 8 路组相联缓存。

当缓存映射中产生地址冲突时，需要通过缓存策略进行替换。一大部分缓存使用最近最少使用（Least Recently Used，LRU）作为缓存策略。LRU 策略选在缓存满时将最近最少使用的缓存块替换掉。LRU 策略要求在硬件中记录缓存条目的最近被使用的时间戳。LRU 策略的优点在于它可以充分利用缓存的空间，保留最近最常使用的数据，提高缓存的命中率。缺点在于需要记录每个缓存块的访问时间或计数器值，增加了额外的空间和时间开销，而且在一些场景下，LRU 策略并不一定是最优的选择，比如数据访问具有周期性，最近访问的数据可能在不久的将来再次访问。

在基础的缓存架构之上，可以根据实际应用对缓存进行定制化的设计，以提高系统的效率。如为了避免某些经常使用的缓存行被频繁替换，可以使用动态重用距离优化缓存的替换策略[48]，使得缓存行的复用率提高。对于某些在计算过程中数

据空间局域性较低的应用中存在的缓存行利用率低的问题，可以使用不规则缓存解决[49]，为数据动态分配不同大小的缓存行。在人工智能应用中，Zebin T[50]设计的卷积神经网络协处理器的数据端口直连 CPU 的二级缓存，极大提升了处理性能。

2.5 本章小结

本章首先介绍了 SNN 中神经元的数学模型，阐述了 SNN 的特点。然后介绍了 SNN 中 STDP 的原理及其和在线学习的关系。先从数学模型的角度了解神经形态计算这一领域的基本模型，对理解神经形态硬件的设计较为重要。对于神经形态硬件，比较重要的部分是它的一般架构和基于 AER 的通信方式。由于缓存在冯诺依曼架构中较为常见，而在神经形态硬件的研究中较少涉及。为了给后续章节在神经形态硬件中引入缓存架构做铺垫，本章节以和冯诺伊曼架构作对比的方式介绍了神经形态硬件，并在最后详细介绍了缓存的原理。

第三章 系统架构设计

本章将对所设计的神经形态架构作详细的介绍。了解总体设计之后，会介绍本设计的主要贡献，然后对各个子模块进行单独分析。

3.1 总体架构

如图 3-1，本文的神经形态架构包含 6 个核心单元。其中，有 4 个用于突触电流累加具有在线学习功能的突触核心；一个用于神经元脉冲发放的神经元核心；一个用于配置寄存器和与外界交互的控制核心。缓存架构在突触核中实现。整个模块通过总线挂载到 ZYNQ 系列的 FPGA 的片内总线接口。外部上位机通过网络接口将 SNN 部署到硬件上。

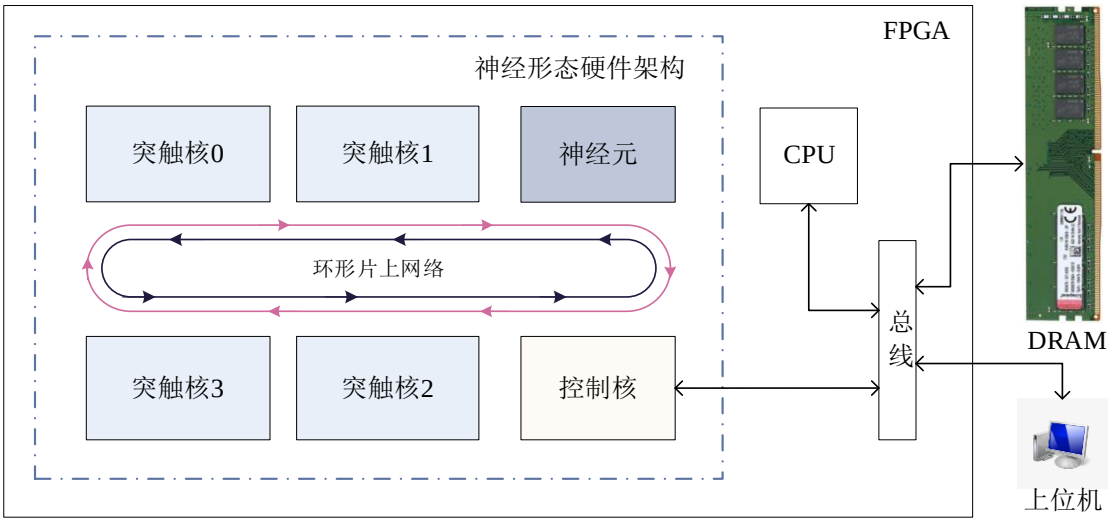

图 3-1 总体架构

该芯片启动后，首先将网络的权重载入 DRAM，配置各个核心的寄存器从而将 SNN 负载映射到每个核心当中。配置完成后即可进行 SNN 的推理和学习。其中一个步长的推理过程如下：

（1）控制核从 DRAM 中读取输入脉冲，向突触核发送突触前脉冲事件

（2）突触核接收到突触前脉冲事件并开始推理计算。在计算过程中，控制核与突触核通过 AER 交互，为突触核中的缓存提供所需的权重。

（3）突触核计算完成，将电流部分和发送到神经元。

（4）神经元集齐所有电流部分和，进行神经元计算更新膜电位并发放，产生的突触后脉冲发送到控制核。

（5）控制核将突触后脉冲写入 DRAM，通知上位机该步长的推理结束。

3.2 主要贡献

本文的主要贡献是提出了适用于神经形态硬件事件驱动架构的缓存，使神经形态硬件能够利用缓存的优点进行运算的优化。此外，使用了缓存的神经形态硬件给支持在线学习带来了挑战。为了使得缓存架构能够适配在线学习算法，本文针对性地提出了突触前脉冲驱动的 STDP 零开销惰性计算方案，能够在无显著增加计算耗时的情况下完成突触学习。

3.2.1 适用于事件驱动的缓存架构

本文的主要贡献是提出了适用于神经形态硬件事件驱动的缓存架构，使神经形态硬件能够利用缓存的优点进行运算的优化。

神经形态芯片的架构中，通常不会使用传统意义上的缓存。因为神经形态芯片通常采用的是基于事件驱动的数据流体系结构，不同于传统的基于指令顺序的处理器体系结构，因此没有 CPU 中使用的多层级缓存等传统的存储器层次结构。当然，在某些情况下，神经形态硬件可以使用缓存来加速数据访问，但这与传统意义上的缓存有所不同，需要根据具体的应用场景来设计相应的缓存。

在基于 AER 的神经形态架构中没有全局的地址空间，地址作为表神经元的标识（Identification，ID）。因而可以考虑使用神经元 ID 作为缓存的地址，该地址对应的缓存行存储该神经元所对应的突触权重。虽然可以通过这种方式设计适用于神经形态硬件缓存，但是缓存是否有优化效果取决于 SNN 的计算中是否含有时空局域性，也取决于缓存策略和地址映射方案的设计。

3.2.1.1 脉冲神经网络的时空局域性

一个体系结构如果要使用缓存进行优化，它的计算过程中必须蕴含着时空局域性，否则加入缓存是没有意义的。事件驱动的架构中天然具有空间局域性，即当一个脉冲事件产生，则它会触发与之相连的所有突触的电流累加，在模型上的表现形式则是读出权重矩阵的一行然后进行计算。这种空间局域性强于 CPU 中的空间局域性，因为脉冲所触发的整行权重必然会整个地被计算，而 CPU 中相邻的数据只是有更大的可能被计算。但是时空局域性是一个整体，仅仅只有空间局域性是无

法发挥缓存的作用的。本文受到生物神经元编码信息时连续发放现象的启发，认为如果在 SNN 中广泛存在这种现象，则能带来很强的时间局域性。

为了验证该假设，本文对多个神经网络进行了实验，研究其神经元的发放规律。在实验中观察到，SNN 网络中发放率最高的 20%的神经元贡献了 80%以上的脉冲，说明脉冲的发放集中于少数的神经元。为了验证这种时间局域性能带来多少的优化效果，本文使用软件对缓存架构进行建模，研究连续发放现象所能带来的命中率。实验所选取三个网络模型：

（a）用于 MNIST 识别的 RSNN

（b）用于光流估计的 SNN[51]

（c）用于手势识别的 SNN[52]

本文使用软件建模了简单的缓存策略。缓存中只能容纳 SNN 中少部分的权重，进行网络推理所需的权重必须从缓存中获得。实验中采用的是全相联的缓存模型，以最大化缓存的命中率，体现 SNN 中时空局域性的上限。实验中缓存的容量可以动态调整，可以得到不同缓存占比的命中情况，为后续拟定硬件上缓存的大小提供依据。实验研究了使用缓存优化后的命中率，以及缓存的容量和优化效果的关系，实验结果见表 3-1。

表 3-1 各模型的缓存命中率实验结果

模型	层	该层神经元数	缓存神经元数	缓存占比	命中率
a	1	784	128	16%	72%
	2	400	64	16%	68%
b	1	2048	512	25%	25%
	2	110	16	14%	82%
c	1	260k	60k	23%	63%
	2	130k	30k	23%	68%

实验结果中可以看出，网络各层中神经元连续发放现象的强度不同。后层的网络相比浅层网络在缓存中能达到更高的命中率，这是因为它的脉冲更集中于少部分神经元，更加明显地呈现活动的规律性。随着缓存容量的提升，命中率也随之提高。

在实际使用中没有必要追求 100%的命中率，但是命中率也不能太低。前文提到过，缓存的使用是为了解决 DRAM 的带宽瓶颈。假如在缓存架构中的 4 个脉冲事件有 3 个命中 1 个缺失，只有缺失的事件需要占用 DRAM 带宽取数据，而 3 个命中事件由缓存提供带宽。这样一来，在片上可以实现 4 个事件的并行计算，而没有使用缓存的架构只能依赖 DRAM 的带宽进行 1 个事件的并行。这种情况下缓存

能够带来 4 倍的加速。在这里定义加速倍率为m，它和命中率h的关系如式(3-1)所示。

$$m = \frac{1}{1-h} \tag{3-1}$$

由于本设计使用了 4 个突触核心用于缓存架构下的 SNN 计算，所以在的情况下，每个核心的命中率达到 75%左右即可达到最佳的优化效果。对于命中率较低的网络层，可以通过分配给它更多缓存资源的方式提高命中率。从实验结果来看大部分网络层级可以只缓存 20%左右神经元的情况下接近预期命中率。

3.2.1.2 缓存策略

在 2.4 章节中介绍了多种缓存的策略。本文的缓存采用组相联的地址映射方案和 LRU 替换策略。首先要解决的问题是组相联的路数（组内的条目数）的设定。缓存路数越多，命中率越接近理论最优的全相联缓存，但是资源消耗量以 2 的指数倍增长。表 3-1 的实验基于全相联的配置，即路数等于缓存行数。本文通过减少路数再次进行实验，发 8 路组相联能接近全相联的命中率，若继续添加路数则边际效益递减。

由于在 SNN 计算中天然蕴含时间信息，计算按照步长一轮一轮进行，易于实现 LRU 策略。在硬件实现上，需要为每个缓存行添加时间戳标记最近使用的时间。当一个事件经过缓存映射，发现可分配的地址已经被其它事件占据，此时需要在这些地址当中选择一个进行替换。根据 LRU 策略，将每个地址的时间戳与当前的时间求差值，对差值最大的地址进行替换，插入新的数据并更新时间戳为当前时间。事件命中时也需要更新对应的时间戳，这样就能保证通过时间戳标记最近未被使用的缓存行。

但是在实际模拟中发现，这种策略在一个时间步长中有大量事件的情况下表现不佳。因为当所有缓存的时间戳都是最新时间，如果从中随机替换则会替换掉大部分下个步长可能会使用到的数据。针对这个问题，本文在替换策略上进行了修改，在缓存策略上引入“不应期”这个参数。生物神经元在发放过后会有一段时间的不应其，在此期间神经元对输入敏感性降低，不会再次发放脉冲。本文借鉴不应期这个概念，在缓存中将当前时间步长发放过的神经元锁定，保证它在当前时间步长内不会被其它神经元的事件替换，能将权重保留到下一个步长。这种做法会带来缓存已满但是又无法进行替换你的问题。为此本文将缓存的最后一路舍弃，用于暂存这些无法分配的事件，并合理分配缓存资源给 SNN 的每一层，尽量减少这种冲突情况。

综上所述，本文对缓存架构进行了适应神经形态硬件的改造。使用 AER 中的神经元 ID 作为缓存的地址，一个缓存行存储一个神经元所对应的突触权重。使用 8 路组相联的地址映射方案。使用 LRU 缓存替换策略，策略使用 SNN 中的步长作为时间信息，同一步长内的神经元处在不应期中，被强制锁定不可替换。

3.2.2 适用于缓存架构的突触可塑性零开销惰性计算

在神经形态硬件中实现突触的可塑性等学习算法一直是一个难点。主要的难点在于：

（1）需要额外的周期用于学习

（2）脉冲发放时间的存储

（3）减少内存的读写

在缓存架构中，一切计算都是由突触前脉冲驱动，并且突触前脉冲在命中的情况下，其所对应的突触权重一定存在于缓存中。本文认为如果能实现仅由突触前脉冲驱动的 STDP，则能将学习融入到推理的过程中，并且没有额外的访存开销。最终，本文结合惰性计算的方式，在实现了适用于缓存架构的 STDP。

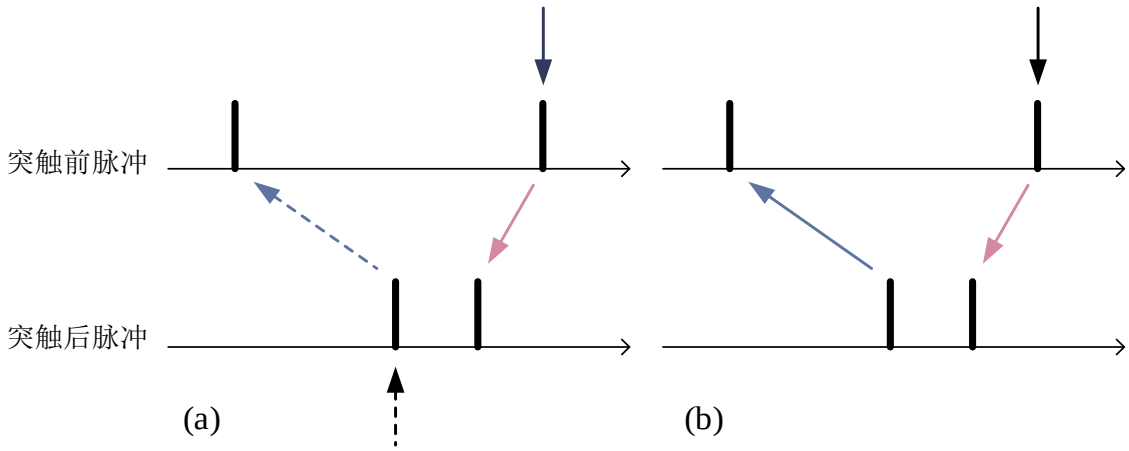

图 3-2 脉冲驱动的 STDP 计算。(a)突触前/后脉冲分别驱动；(b)仅由突触前脉冲驱动

一般的基于突触前/后脉冲驱动 STDP 如图 3-2（a）所示。突触前脉冲（黑色实现箭头所指）会寻找对应的突触后脉冲，触发 LTD 计算。突触后脉冲（黑色虚线箭头所指）寻找对应的突触前脉冲，触发 LTP 计算。

在神经形态硬件中，脉冲历史记录一般以二值序列的形式进行存储，本文采用的也是这种方案。一个脉冲只占用一个比特，脉冲所在位的偏移量则表示脉冲发放的时间。Liu Y[39]设计的 SNN 加速器便采用了这种二值序列的脉冲存储方式（图 3-

3）。在学习周期中将所有的突触前脉冲和突触后脉冲的历史发放记录进行两两比对，则可以得到前后脉冲的时间间隔。但这种方式的效率有待提高，因为脉冲记录的两两比对对应着整个权重矩阵的遍历，即使只有少部分突触的权重需要更新。

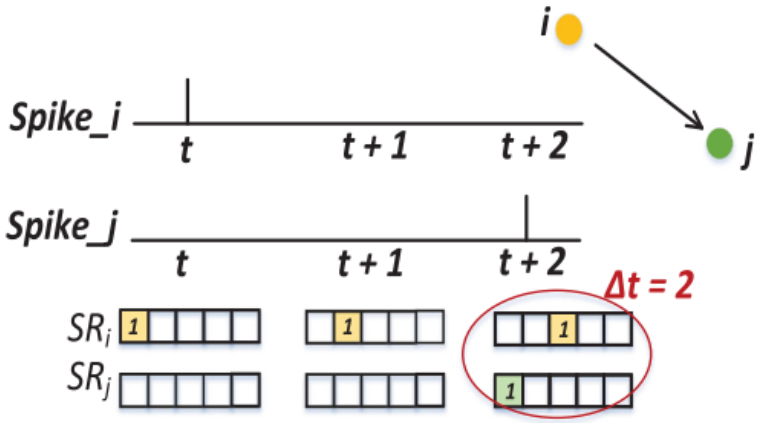

图 3-3 使用二值序列存储脉冲发放记录[39]，可根据前后脉冲发放记录的偏移量差值得到发放时间的间隔

大部分神经形态硬件需要额外的学习周期用于权重更新。Loihi[21]为了在硬件上支持复杂的学习模型，在学习功能的实现上采用了基于步长驱动而非事件驱动。因为用事件驱动的方式实现学习，要求将脉冲发放信息反向传递，会造成事件路由的巨大开销。所以它单独挂载了一个 CPU 在独立于事件驱动的学习周期里面进行权重的调整。在 Payvand M [42]设计的神经形态硬件中，同样使用了一个单独的通用处理器单元作为学习模块。当学习模块收到脉冲后，将误差传回神经元。神经元当收到误差事件时会停止工作直到权重更新的完成。需要额外学习周期的情况很多时候无法避免，因为下一轮推理依赖当前步长学习到的最新权重。但本文通过惰性计算实现的 STDP 可以避免这种情况，实现权重更新的零开销。

大部分相关工作中 LTP 和 LTD 需要分开计算。LTP 和 LTD 的访存方式不同，如果其中一个是按行遍历权重矩阵，那么另一个就是按列遍历。按列遍历在 DRAM 中的效率是不可接受的，而 SRAM 则可以快速响应任意方式的访存。这也是为什么大部分实现在线学习的神经形态硬件都使用 SRAM 存放所有权重的原因。

为了解决访存效率的问题，Kim G[25]在 STDP 的实现中提出了基于突触后脉冲驱动 STDP 方案，仅有突触后脉冲能触发学习，避免了遍历权重的开销。他基于突触后脉冲发放率较突触前脉冲低的事实，认为只用突触后脉冲触发 STDP 能有效减少访存。虽然只用一类脉冲即可触发学习，但是这项工作没有解决不规则访存的问题，他的实现依然依赖于可按列读取的 SRAM。本文使用了缓存，只有部分权重行存储在芯片上，意味着没有完整的列，无法支持权重的列遍历。

如果想要避免按列访存，那么学习则不能由突触后脉冲驱动，必须和推理一样，由突触前脉冲驱动。本文发现仅由突触前脉冲驱动 STDP 是可行的，而且能够与缓存的架构完美契合。在事件驱动的架构中，只有突触前脉冲会触发推理，只需要在推理发生之前把权重更新到最新即可，而不需要在突触后脉冲到达时马上更新权重。这种当一个值只有在用到的时候才会被计算出来的方法称为惰性求值。当惰性求值与事件驱动相结合，即可做到零开销的 STDP。实现方式如图 3-2（b）所示。当突触前脉冲到来，首先查询该突触前脉冲的发放记录，然后遍历突触后脉冲的历史发放记录进行一一比对，由此可以得到一行更新后的权重，而这行最新权重可以马上用于下游的推理计算。因为推理功能本身就需要读取这行权重，所以可以认为本文的在线学习方案没有额外的访存开销。

3.3 基于事件的多核心互联

神经形态硬件一般都是多核心的架构，核心之间使用片上网络通信。相比于一般的集成电路系统，神经形态硬件需要互联的核心数量较多，并且核心间的脉冲数据、电流数据的通信会造成片上数据通路的极大负载。高效的片上互联设计是神经形态硬件设计中首要且重要的工作。

一般的计算机体系结构中的片上互联中存在主设备和从设备两种类型的模块，片上网络或者总线都是基于主从类的通信协议进行设计。然而神经形态硬件中的众多核心往往是同构的，不存在主从关系，一般采用基于事件的通信协议进行片上互联的设计。同构一般指多个核心是完全一样的，其在 SNN 中完成同样的功能。对于常见的神经形态硬件多核架构，每个核心被设计为突触和神经元耦合的形式。核心接收脉冲事件的输入，根据事件的地址寻址突触和神经元，依次进行突触的计算和神经元的计算。这种同构的多核互联的架构会带来脉冲传递的问题，因为网络的拓扑连接信息需要依赖片上网络的路由表进行实现，通过传递事件的形式表示拓扑连接。虽然表面上简化了多核心架构的设计，但是实际上将系统的复杂性转移到了多核通信上。如图 3-4（a）所示，每个脉冲事件单独传递，一个脉冲会消耗一个网络包的资源。

为了高效地在核心间传递脉冲，一些研究工作采用将多个脉冲聚合到一个网络包发送的方式进行脉冲传递，能够显著降低片上通信的流量，如图 3-4（b）所示。例如，张兆名[53]在脉冲包中只传递源神经元的 ID，这样仅需要在目的核心解包即可得到与目的神经元需要的多个脉冲。类似的，在 Loihi[21]的架构中，为了减少路由资源的消耗，采用传递轴突 ID 的方代替传直接传递脉冲，目标核心对接收到的轴突 ID 进行查表即可寻址到对应的目的神经元。

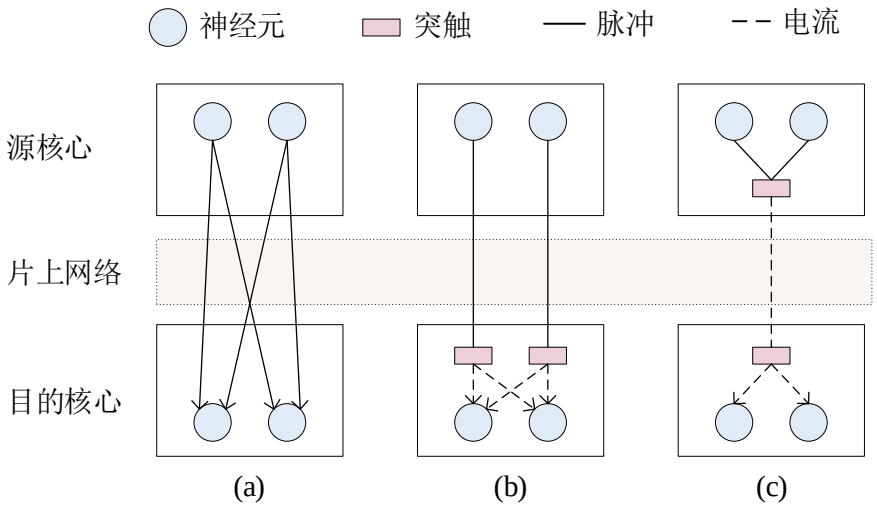

图 3-4 事件传递方法。(a)同构多核的脉冲传递；(b)通过聚合脉冲减少通信开销；(c)使用电流事件代替脉冲事件

本文认为这种事件通信的开销是设计者依赖片上互联的路由来实现神经网络的拓扑导致的。在这种情况下，只能同过脉冲聚合的方式尽量减少通信开销。如果在单个核心内部实现网络拓扑的连接，可以大大降低事件传递的复杂性。于是本文提出了专用于全连接层电流累加计算的突触核心，用单核代替多核，避免了多余的脉冲传递。当突触核心完成电流部分和的累加时，只需要将电流打包为电流事件传递至神经元核心即可。如图 3-4（c），对于多个神经元互联的拓扑，片上网络只需要传递一次事件即可。

除了电流事件以外，本文基于缓存的设计，提出了权重事件这一概念，用于传统缓存架构的事件驱动改造。本文的设计中一共包含 6 个核心，围绕事件驱动设计了片上互联的通信协议，并且实现了该协议所对应的片上网络电路。

3.3.1 通信协议

通信协议见图 3-5，包含三个层级，每一层由上一层协议拓展而来。

首先，最为基础的是片上网络的通信协议，其目的是实现核心间的通信，不关心通信的具体内容。包头中包含源地址和目的地址的信息，和用于拓展的自定义字段。包体能够传输一段连续的数据。

在片上网络协议之上，本文拓展了基础协议。基础协议包含寄存器读写的命令和数据读写的命令。可以通过基础协议读写任意核心的寄存器完成配置、控制等操作，也可以直接访问某个核心的内部存储的数据。基础协议中约定：每个包包含一

个 ID 字段，当从模块处理完命令类型的包后，必须向主模块返回一个包含该 ID 的响应包。

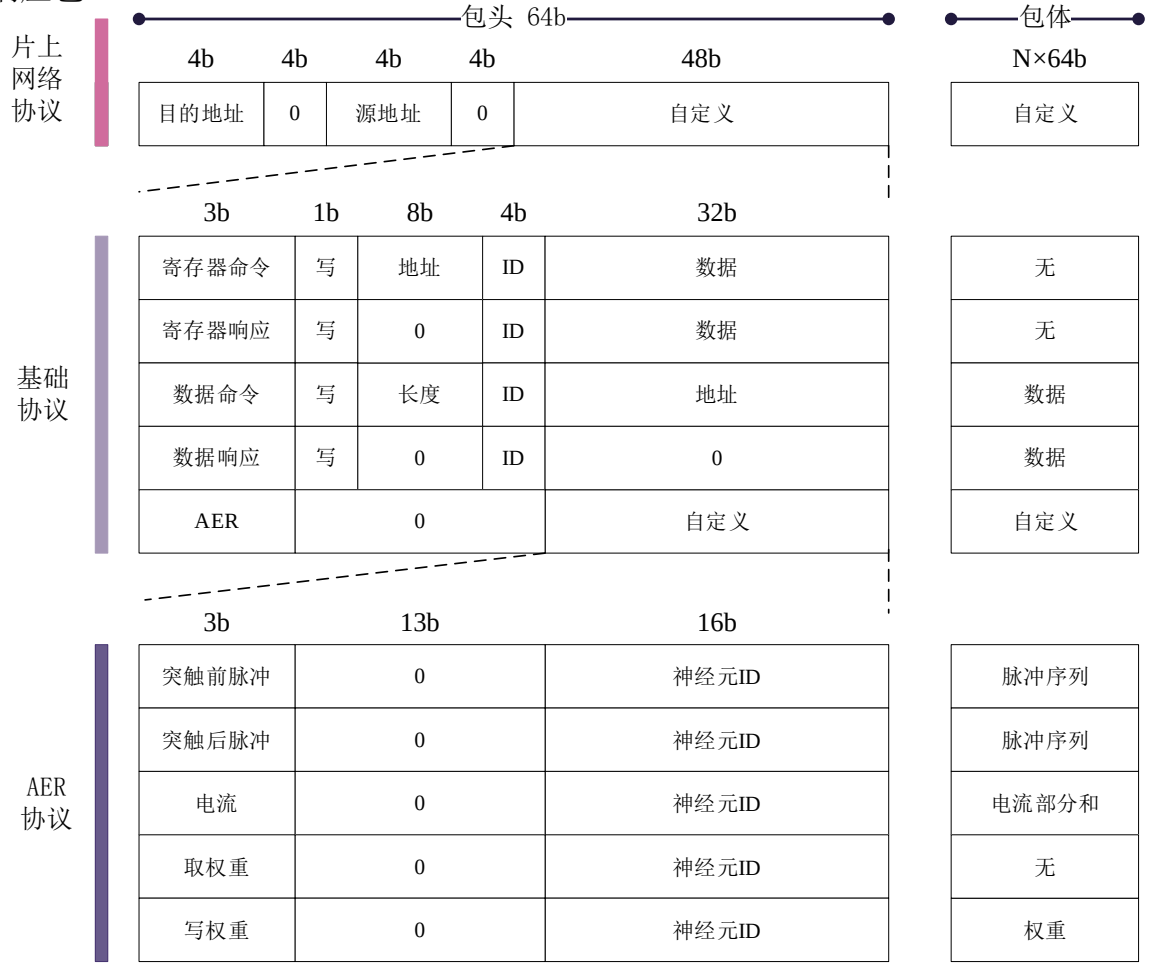

图 3-5 通信协议

AER 协议由基础协议拓展而来。与基础协议不同的地方在于，AER 协议只发送事件，而没有返回值，是无状态的协议。AER 协议中包含 5 种类型的事件：突触前脉冲；突触后脉冲；电流；取权重；写权重。脉冲事件中的脉冲序列使用单比特表示一个脉冲，根据神经元 ID 和比特的偏移量，即可得到脉冲的发放信息。之所以将脉冲区分为前后脉冲两种类型，是因为不同类型的脉冲在计算当中对应不同的行为。比如在突触核心当中，突触前脉冲将会触发 SNN 的计算，而突触后脉冲只会触发脉冲历史记录的更新。多种事件类型还可以实现系统级别的事件驱动（图 3-6），使得多核心之间直接使用事件进行通信，相比于对每个核心进行单独控制的方案更加高效。

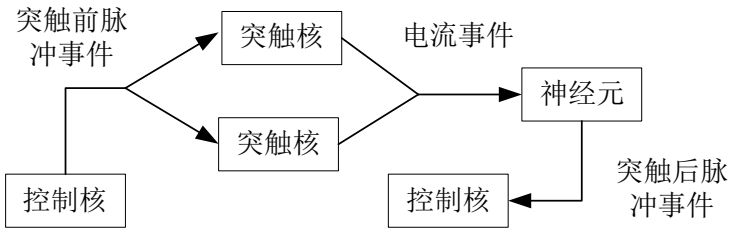

图 3-6 基于事件的控制流

在控制流中，一个步长的推理从控制核发出突触前脉冲事件开始。突触核进行事件驱动的计算完成后，产生电流事件，这可以理解为事件发生了转化传递。神经元再将电流事件转化为突触后脉冲事件传递至控制核，即完成了一个步长的计算。从基于事件的控制流中可以发现，外部程序只需要对控制核施加控制即可发起计算和判断计算完成，无需单独对系统中的其它核心施加控制。基于 AER 协议进行控制极大地简化了控制方案，也使得系统更加高效。

由于使用了缓存的设计，当权重不在缓存中，即发生缺失的时候，需要通过片上网络从 DRAM 中获取权重数据。如果通过基础协议的数据读写命令完成取权重，需要在每个突触核中实现神经元 ID 到 DRAM 地址的映射，增加设计的复杂度。而且网络返回数据的顺序不一定与发送时相同，虽然可以通过返回的包头的 ID 进行区分，但是这在硬件实现上不够轻便。当今很多深度神经网络的加速器采用数据流的架构[56]，以避免片上核心直接访存。每个核心只对收到的包进行计算、转发，对于已经发送的包则不会再有任何联系。使用这种架构的系统能够做到易于拓展并且能够有效避免片上网络的拥塞。本文收到该架构的启发，提出了基于 AER 协议获取权重的方案。在 AER 协议中，突触核心只负责发送取权重事件和处理写权重事件。由于 AER 协议中只需要神经元 ID，所以突触核心不需要考虑地址的映射，也就是说 DRAM 对于突触核心是不可见的。在收到写权重事件时，可以直接从包头中获取神经元 ID 进行缓存的写入。这样的设计合理地解耦了权重的获取和写入，带来更加轻便的设计。

3.3.2 片上网络

大部分的神经形态硬件因为核心数量较多，在互联方面一般采用 Mesh 结构的片上网络[54]。在本文的架构当中，由于核心数量较少，使用 Mesh 作为片上网络的实现方案会有性能的冗余并且结构较为复杂，相比之下，环形网络更加适合较少核心的互联，并且其路由器结构较为轻量。本文的 6 核心环形片上网络如图 3-7 所示。

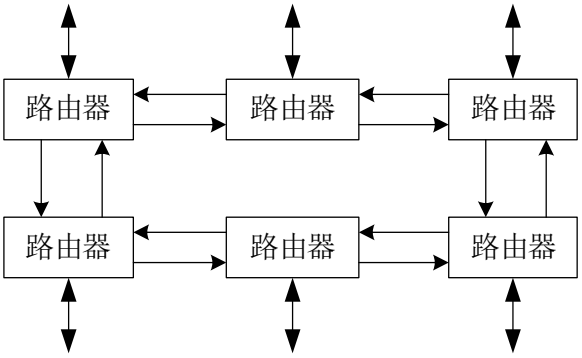

图 3-7 环形片上网络

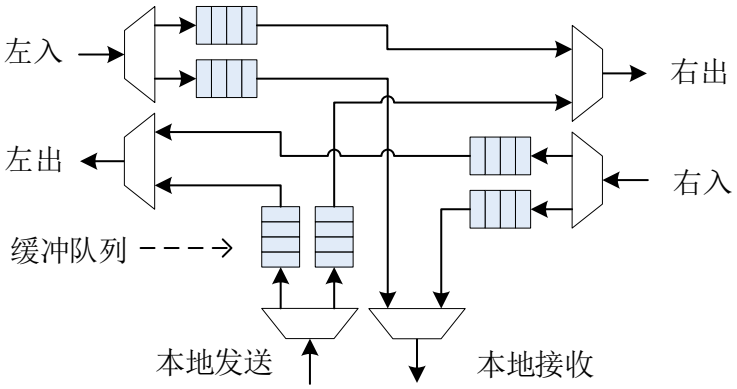

图 3-8 路由器模块

路由器是片上网络的核心部件，参考常见的设计[55]，包含缓冲、路由分发、出口仲裁这几个主要模块。如图 3-8 所示，对于任意输入端口，网络包首先经过路由分发，根据出口的方向被送入对应的缓冲队列；对于每个输出端口，对应两个上游缓冲队列，输出端口对其进行仲裁输出。

路由器只实现最低级的片上网络协议。为了每个核心能够便捷地通过协议进行控制而不需要考虑底层细节，本文设计了包处理单元，将其置于核心与片上网络端口之间，能够将原始的网络包解码为各种总线格式。包处理单元如图 3-9 所示。包处理单元内部包含一个解包器模块。该模块内部通过状态机控制，根据包头的类型进入对应的处理状态。基础协议包含寄存器读写和数据读写两种类型，分别被解码为先进外设接口（Advanced Peripheral Bus，APB）用于寄存器读写、数据总线用于核心内部存储体的读写。基础协议的响应由包处理单元自动完成。AER 协议的包会被解码为 AER 总线的信号。该模块自动处理通信协议，能够极大方便其它核心的开发工作。

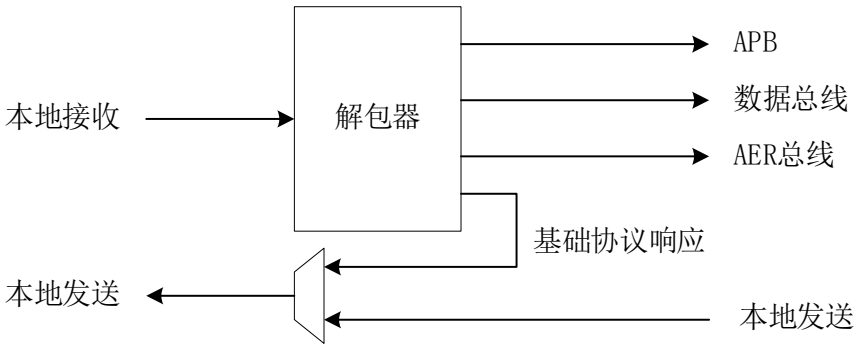

图 3-9 包处理单元

3.4 突触模块

突触模块是本文神经形态硬件的核心模块，完成 SNN 的推理和学习功能。突触核接收突触前脉冲输入，进行突触后电流的累加求和计算，之后将电流部分和以事件的形式打包发送至神经元模块。

在一般的神经形态硬件中，用于突触学习的模块为了支持对突触权重的随机读写，需要使用大面积的 SRAM 存储突触权重。这导致突触模块往往是存储资源的最大消耗者。本文提出的缓存架构便是在突触模块中应用，最大程度地减少存储资源的消耗。突触模块的完整架构见图 3-10。

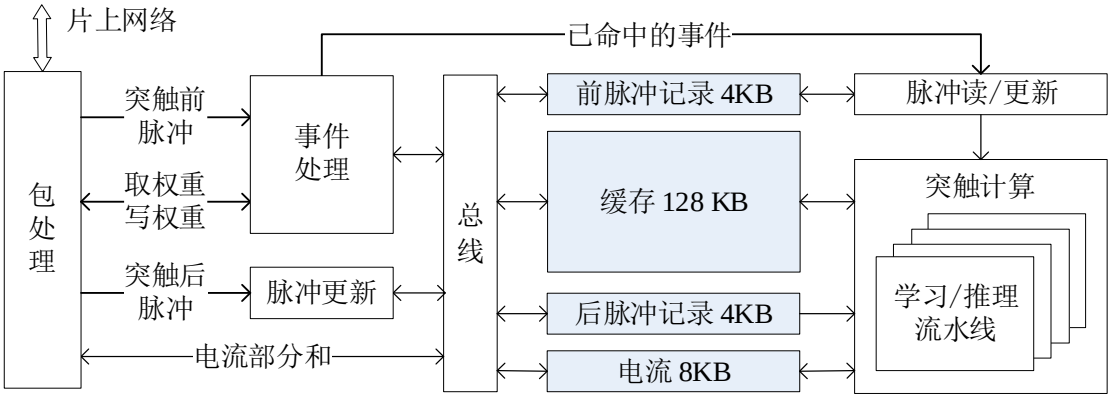

图 3-10 突触模块

架构中包含了数个子模块。包处理模块将收到的网络包按照事件类型发送给不同的子模块，并且还负责主动打包并发送事件包。突触前后脉冲的发放记录被存储于两个 4 KB 的 SRAM 中。缓存是一块定制的多端口 SRAM，与其它存储器一起挂载在模块内部总线上。图中缓存只是一块普通的存储器，缓存的功能主要依靠事件处理模块进行实现。

架构中最主要的两个模块是事件处理模块和突触计算模块。事件处理模块接受脉冲事件的输入，通过脉冲控制缓存，保证脉冲事件所需的权重已经存在于缓存中后，将脉冲事件发送至突触计算模块进行计算。在脉冲事件到达突触计算模块前，会读取该脉冲对应的历史发放记录，将其附加到事件信息当中。随后突触计算模块根据事件中的信息进行计算。突触计算模块可以配置为学习和推理两种模式，这两种模式共用同一条计算流水线。当完成完整步长的计算后，突触模块将从电流存储器中读取电流的累加和，将其打包为电流事件发送到片上网络中。这一过程由突触模块的主状态机进行控制，见图 3-11。

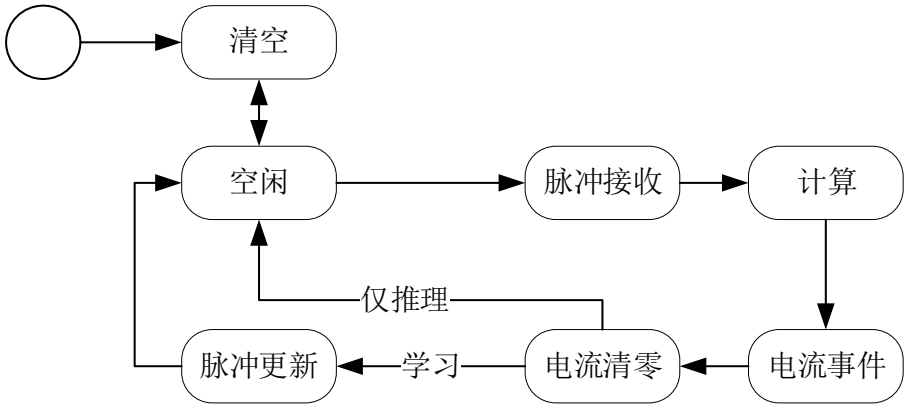

图 3-11 突触模块主状态机

其中，状态机在复位后，首先进入清空状态，将数据通路和缓存的有效标签清零。在空闲状态中，如果收到了突触前脉冲，将进入脉冲接收状态，将脉冲包暂存。在计算状态中，模块中的部件自主处理脉冲包中的事件信息，处理完毕后结束该状态。电流事件状态将电流读取并发送，之后进如电流清零状态将电流存储器清零，为下个步长的电流累加做准备。之后根据配置信息进行状态的跳转：如果是配置为仅推理状态，则直接进入空闲状态等待下一次推理；如果处于学习状态，则突触模块需要等待神经元返回的突触后脉冲事件进行脉冲的更新，为下一次的学习准备最新的脉冲历史记录。

3.4.1 事件处理模块

由于缓存中只包含部分神经元的权重数据，当一个脉冲事件到达突触模块时，它所对应的权重可能存储于缓存当中，也可能不存在。所以脉冲处理的第一步是将脉冲分为命中脉冲和缺失脉冲。其中，命中脉冲代表该脉冲的权重数据已经准备好，可以马上进行计算；缺失脉冲代表缓存中不存在该脉冲的权重，需要从片外 DRAM 中载入权重到缓存。这一系列的操作由事件处理模块完成，见图 3-12。

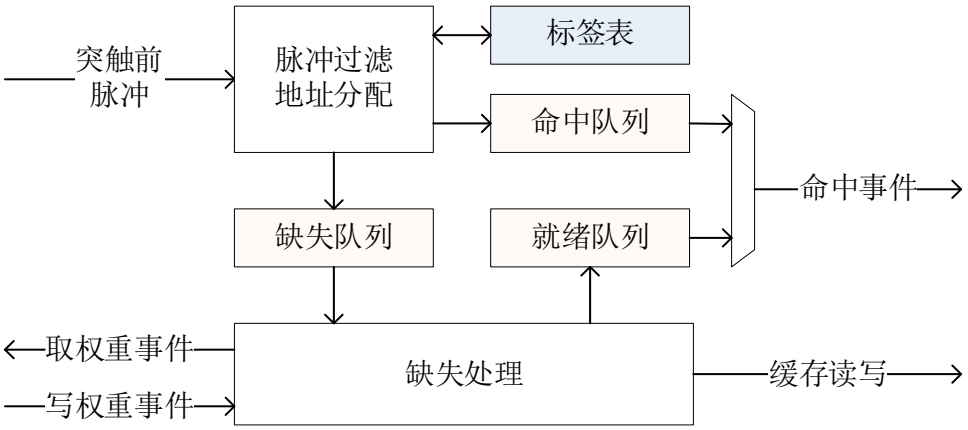

图 3-12 事件处理模块

经过脉冲过滤和地址分配后的脉冲，如果是命中脉冲，会附加上其权重所在的缓存地址，加入命中队列中；如果是缺失脉冲，则会分配一个缓存中地址用于权重将来从 DRAM 中写入，脉冲事件将加入缺失队列。缺失处理单元从缺失队列中获取脉冲事件，将其转化为取权重事件，经由片上网络发送到控制核。控制核负责获取权重并返回写权重事件。当缺失处理单元接收到写权重事件，会将权重写入之前分配的缓存地址中。权重完全写入之后，从缺失队列中取出缺失事件，将其转化为命中事件存入就绪队列中。就绪队列中的事件相比命中队列中的事件具有更高的仲裁优先级。这是因为缺失处理需要较长的时间，如果优先消耗命中队列的事件，那么当就绪队列事件也为空时，下游将会没有事件可以处理，降低计算单元的效率。

3.4.1.1 脉冲过滤和地址分配

模块中包含一个标签表，脉冲的过滤和地址分配通过查表实现，如图 3-13。图中简单描述了脉冲事件的查询过程及其缓存地址的分配。脉冲事件中神经元 ID 被分为高位的标签和低位的组索引两部分。标签表是一块 RAM，表中的一个标签对应标记缓存中一个缓存行的状态。组索引被直接当作标签表的地址进行查询。示意图中的标签表对应的是 4 路组相联缓存的结构，一次可查询出 4 个标签。将这些标签和神经元 ID 的标签字段进行比对，相等则表示命中。将命中标签对应的路地址作为缓存地址的低位，和组索引拼接成缓存地址。每个缓存行的地址都有一个对应的标签表项进行状态记录。

对于查询结果为缺失的情况，经过优先级的判断，选择缓存中的一路分配给缺失事件。未被分配的地址具有最高优先级，只需要将获取的权重直接写入缓存即可。其次如果所有路都已经被分配，则执行替换算法，将最近未使用的一路作为替换目

标。被替换的权重将在学习模式下将被写回 DRAM；在推理模式下则直接被新的权重覆盖。

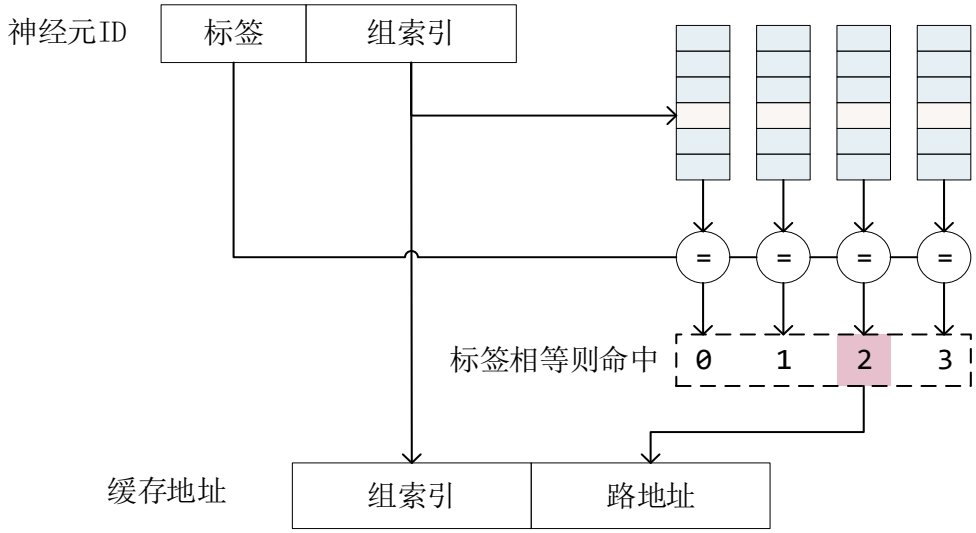

图 3-13 脉冲查询示意图

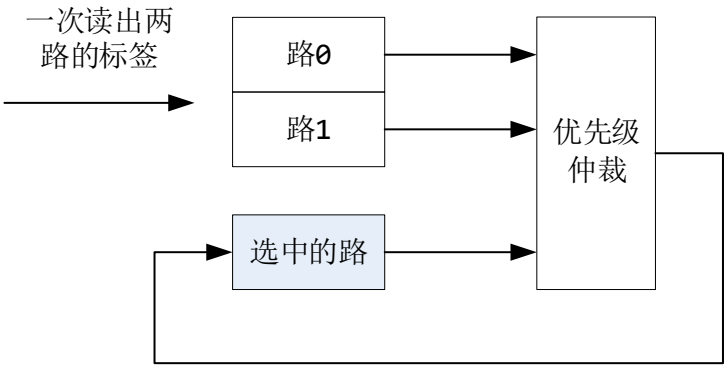

图 3-14 缓存地址分配逻辑

此外本文还对标签查询进行了针对性的优化。CPU 中缓存标签的查询有着严格的延迟要求。对于最靠近 CPU 的一层往往要求在 1 到 2 个时钟内完成命中的判断。对于 8 路组相联缓存，这意味着需要 8 个比较器并行地比较标签和基于优先级的地址分配。这一块是缓存中较为占用逻辑资源且影响时序的部分。本文认为神经形态硬件中对缓存标签查询的延迟较为容忍，提出了多周期查询的方式以减少逻辑。在本文的架构中，一个事件在下游突触计算的处理需要 100 个以上的时钟周期，事件处理模块只需要能够满足下游的吞吐量即可。如图 3-14 所示，对于 8 路组相联缓存，多周期查询的方案每次只查询 2 个标签，在 10 个时钟周期内可处理一个事件，不仅满足了性能要求，还减少了逻辑资源的使用量。优先级的仲裁选

出最优的可分配的缓存路地址，暂存到寄存器中。经过 4 次这样的优先级仲裁迭代，即可将最合适的地址与脉冲事件绑定。

表 3-2 标签状态优先级

优先级	标签状态
1	命中
2	可分配
3	可替换
4	不可替换

根据标签的状态，可将待绑定的缓存地址分为多个优先级。如

表 3-2 所示，其中，优先级的数字越小，代表优先级越高。如果可选的路中存在命中的状态，则表明该脉冲事件的权重已存在于缓存中。因为命中状态具有最高优先级，所以只要存在命中状态，那么最后一定会选中命中状态所在的路地址。当脉冲发生缺失的时候，需要分配一个地址用于存储它对应的权重，此时优先分配空闲的空间要优于替换已经有数据存在的空间，所以可分配的优先级要高于可替换。当对应的路地址的缓存行正处在不应期中，需要强制保留，此时无法替换，优先级最低。

3.4.1.2 缺失处理

缺失处理单元负责处理缺失脉冲。其中，缺失脉冲有三种类型，如表 3-3 所示。缺失处理单元根据缺失的类型发送和接收对应的权重事件。当权重已经取得并写入缓存之后，缺失脉冲即转变为命中脉冲，放入就绪队列中通往下游计算。

表 3-3 缺失脉冲的类型及其行为

缺失脉冲的类型	场景	发送事件	接收事件
覆盖	推理模式下发生缺失	取权重	写权重
写回	学习结束，清空缓存	写权重	
替换	学习模式下发生缺失	写权重、取权重	写权重

在于推理模式下，缺失均为覆盖类型。此时缺失处理单元向控制核发送取权重事件，并等待控制核返回写权重事件。当接收到写权重事件后，将权重写入缓存，并将对应的缺失脉冲转变为命中脉冲，放入就绪队列中通往下游计算。

处于学习模式下，缺失事件为替换类型。此时缺失处理单元首先从缓存中读取待替换的旧权重，将其打包为写权重事件发送到控制核，该权重最终将被写回 DRAM 中。在发送写权重事件之后，接着发送取权重事件，并等新的权重返回。

当学习结束时，需要清空缓存，将所有改变后的权重写回 DRAM，保证 DRAM 中的权重和学习结果的一致性。

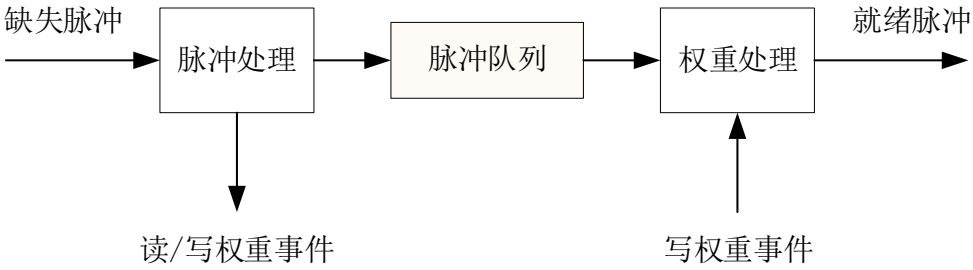

图 3-15 缺失处理流程

缺失处理单元的设计如图 3-15 所示。缺失脉冲首先会经过脉冲处理，如果是处在推理模式下，则向控制核发送读权重事件。当控制核收到读权重事件后，会从 DRAM 中读取权重并打包为写权重事件返回突触核的缺失处理单元。缺失处理单元将权重写入在事件处理单元中已经分配好的缓存地址当中，将缺失脉冲标记为就绪脉冲。就绪脉冲和命中脉冲本质是一样的，表示权重数据已经准备好，可以直接进行计算。在学习模式下，因为缓存中的权重经过学习的更新，需要在脉冲处理这一步，提前将其打包为写权重事件，写回 DRAM。

由于发送取权重事件到其响应有较长的延迟，为了掩盖这部分延迟，该模块设计为可以连续发送多个取权重事件，不必等待之前的取权重事件返回。这一功能通过图 3-15 中的脉冲队列实现。这样一来，在工作的时候可连续地收到返回的写权重事件。

3.4.2 多端口缓存

在本文设计中，事件处理单元和突触计算单元会同时访问缓存。事件处理单元执行替换操作需要先读出旧的权重数据然后写入新的权重数据；突触计算单元从缓存中读取权重运行学习算法，然后将更新后的权重写回缓存。而实际上，如果要支持这两个单元并行访问缓存中的RAM，在设计上会面临RAM端口不足的问题。

FPGA 中使用块随机访问存储器（Block RAM，BRAM）作为存储单元。BRAM 可以根据需求进行配置为单端口或者多端口存储器。其中多端口存储器分为两类：

（1）简单双口 RAM：拥有一个读端口和一个写端口。

（2）真双口 RAM：拥有两个同步读写端口。同步读写端口根据选择信号进行读或者写，一个端口无法同时进行读和写。

本文的设计中要求缓存拥有一个同步读写端口专用于事件处理单元，一个写端口和一个读端口用于突触计算单元并行的连续读取和连续写入。而 BRAM 最多配置为两个端口，无法提供本文所需的三个端口。为了满足需求，本文提出了多端口缓存的设计，见图 3-16。

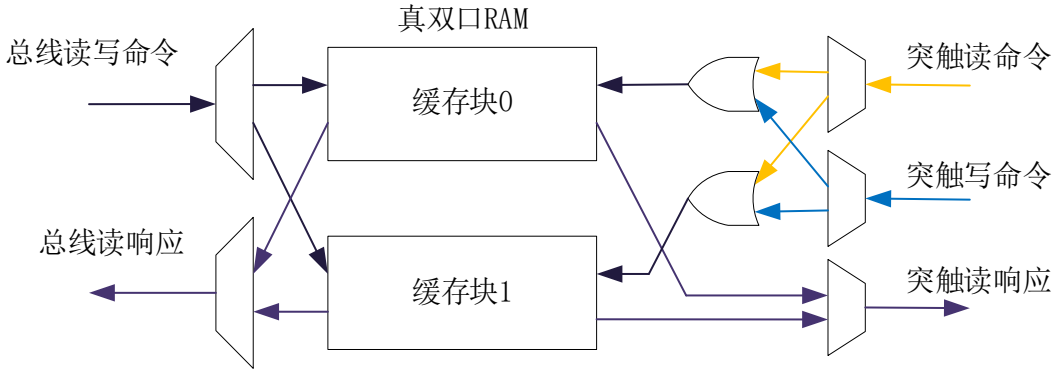

图 3-16 多端口缓存

多端口缓存使用两块真双口 RAM 拼接而成。缓存地址空间中的偶数地址映射到缓存块 0 中，奇数地址映射到缓存块 1 中。缓存块的端口之一用于组成连接到总线的访存端口，另一端口用于组成突触方向的访存端口。在总线端口中，一次总线读写命令根据地址的奇偶性选择一个缓存块进行访问。在突触端口中，读和写命令同样被分别分解为两个方向，目的地是同一个缓存块的命令会经过或门的仲。这里使用或门而不是多路复用器，是因为突触方向的端口采用无反压的设计，即不采用握手信号，当读/写命令到来时，必须立刻处理不能停顿。理论上，正确的外部逻辑不应该产生使得缓存发生冲突的错误信号，所以缓存块的使能信号只需要将到该缓存块的读/写信号相或即可。

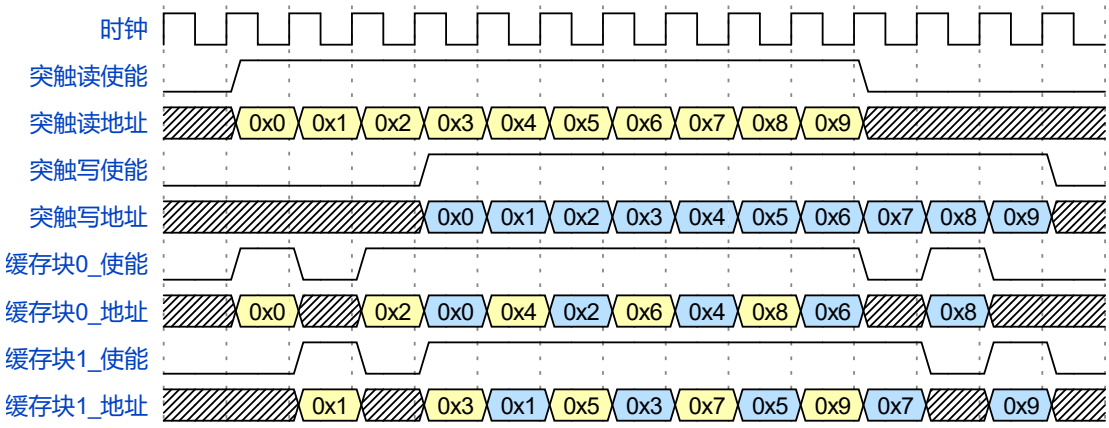

图 3-17 缓存的突触端口读写时序图

由于突触是先读取一行连续地址的数据，计算过后将数据写回原来的地址，地址是奇偶交替的，所以只写命令比读命令延迟奇数个时钟周期，就能够避免读端口和写端口同时访问同一个缓存块的错误情况。不过对于单个模块的设计，不能依赖外部模块的正确性。本文在突触端口增加了对于地址冲突的断言，能够在仿真中产生地址冲突的报错。突触端的读写时序如图 3-17。图中模拟了突触从缓存中读取连续的 10 个数据，经过流水线计算后写回缓存的情况。设定流水线发送读命令之后，延迟三个时钟发送写命令。由于读写命令的地址是连续的，每个地址被交替地送往对应的缓存块。任意一个缓存块全程处于使能的工作状态，并且一个时刻只处理读或者写事务，不会产生读写冲突。从图中可以看到，仅使用 RAM 的单个端口，突触端的读写可以实现无阻塞和并行。

3.4.3 突触计算模块

突触计算模块接收带有突触前脉冲历史记录的脉冲事件，基于 3.2.2 提出的突触前脉冲驱动的 STDP 零开销惰性计算方案，可在推理的过程中完成 STDP 的学习。该模块由七级流水线构成，见图 3-18。流水线经过精心设计，对各个部件的延迟进行统筹规划，将其在流水线合适的位置插入。

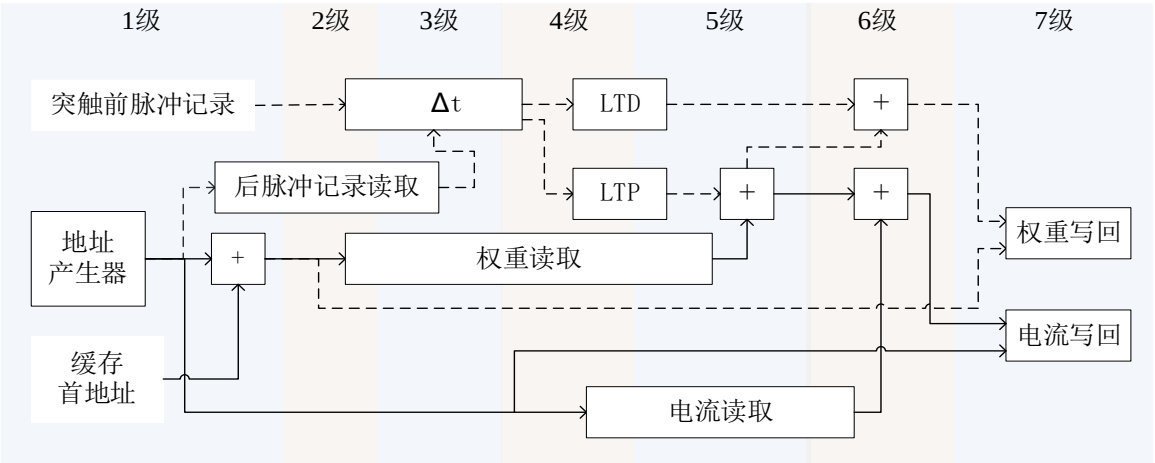

图 3-18 突触 7 级流水线

图中实线路径是在仅推理模式下流水线的工作路径。在仅推理模式中，只需要将脉冲对应的权重累加到电流累加值中即可。首先在第 1 级由地址产生器产生地址的偏移量，偏移量与缓存首地址累加产生权重的具体地址。在第 2 级向缓存发送读权重命令，缓存读取的延迟为 3 个时钟。在第 4 级读取电流累加值，读取延迟为 2 个时钟。在第 6 级得到电流的最新累加值，在第 7 级进行写回。

在学习模式下，需要在电流累加（第 6 级）之前准备好当前步长的最新权重。由于权重的更新都是突触前脉冲驱动，所以可能存在未计算的依赖于突触后脉冲的 LTP 事件。在流水线中通过对比突触前、后脉冲的发放记录，可判断是否有 LTP 事件及其时间差。将 LTP 带来的权重增量与旧权重相加可得到当前的最新权重。此外。对于当前步长的最新权重，在完成电流累加后，将其进行 LTD 的抑制操作后直接写回缓存，不需要额外的学习周期用于权重的更新。

流水线中的 LTD 和 LTP 模块通过查找表的方式实现突触可塑性计算。在硬件初始化之时即将各个Δt的权重变化值存入查找表中。这种方式足够灵活，可以根据实际情况使用不同的 STDP 算法，只要是基于脉冲对的时间差的算法即可。

从流水线中可以看出，本文的 STDP 方案做到了零开销的结果。即使在学习模式下，能够利用 STDP 惰性计算的特性，把学习的路径和推理的路径相融合，达到完成推理的同时完成学习的效果。

3.4.3.1 脉冲时间差计算

在突触计算模块中，第 2 到 3 级流水线中存在脉冲时间差（Δt）计算单元。该单元输入脉冲的二值序列，分别输出用于计算 LTP 和 LTD 的脉冲对的时间差。具体实现如图 3-19 所示。

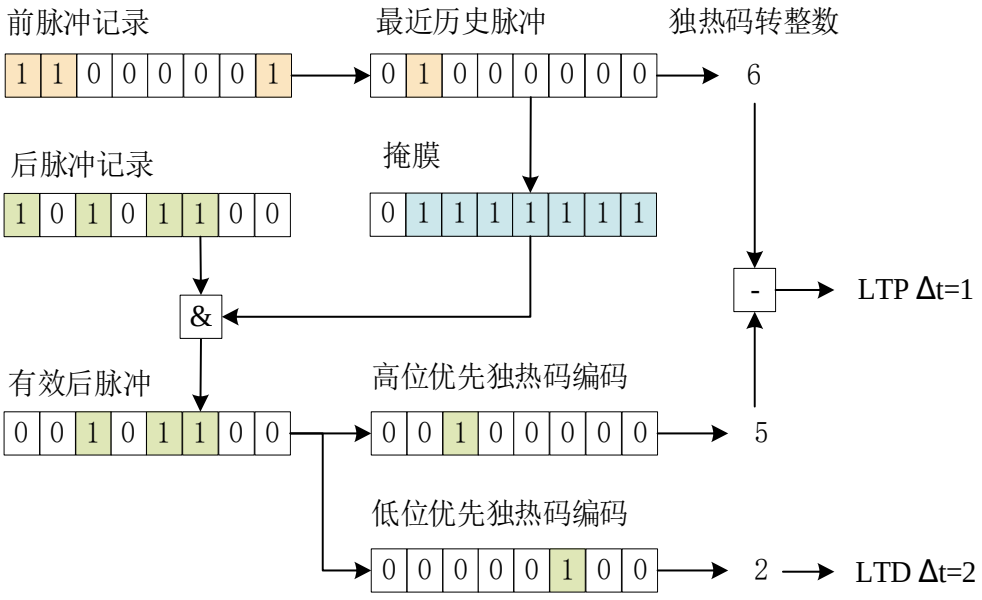

图 3-19 脉冲时间差计算示意图

从图中可见，该单元首先对突触前脉冲记录进行最低位除外的低位优先独热码编码得到最近的历史脉冲，该脉冲可能触发 LTP。为了判断是否存在 LTP，需要由最近历史脉冲生成掩膜和后脉冲记录相与，得到有效的后脉冲记录的范围。之后对有

效后脉冲分别进行高、低位优先的独热码编码，只保留有效的 LTP 和 LTD 后脉冲。此时已经能得到 LTP 和 LTD 的脉冲对，对独热码的脉冲进行整数转换，得到脉冲确切的发放时间。最后即可计算得到脉冲对的时间差。

3.5 神经元模块

当突触核心完成电流累加的计算后，电流部分和以电流事件包的形式经由片上网络到达神经元模块进行进一步的处理。神经元模块的行为较为简单，对膜电位进行阈值操作，完成脉冲的发放并对膜电位进行复位。神经元的计算流水线如图 3-20 所示。

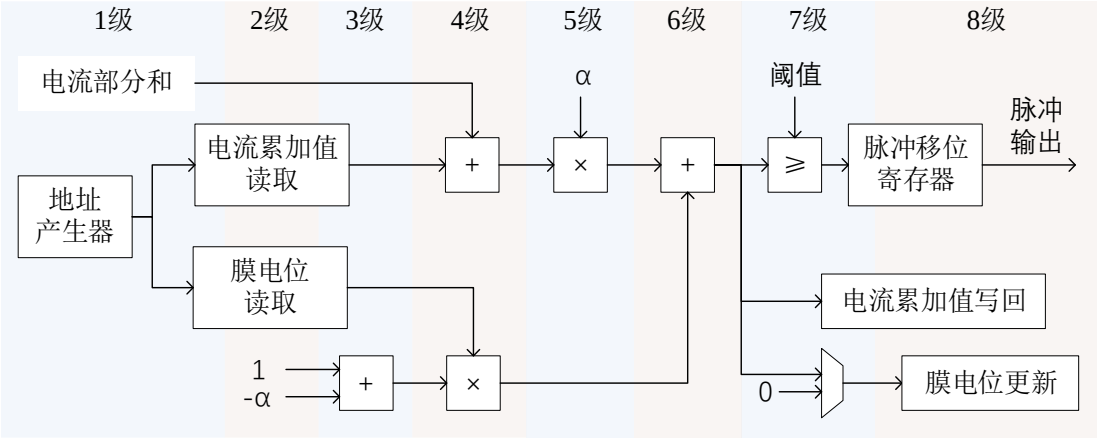

图 3-20 神经元计算流水线

该流水线共有 8 级，完成式(2-1)的神经元膜电位的整合。在流水线中需要先对电流部分和进行累加，是因为可能一层的神经元被分配到多个突触核心当中，每个突触核心发送的只是该层突触后神经元的输入电流部分和，在神经元模块内，需要将多个突触核的电流事件进行累加。在电流事件累加的过程中，流水线的膜电位计算路径是不工作的，除非是最后一个电流事件。神经元模块中会对电流事件进行计数，只有收到最后一个电流事件时，才会并行的进行电流累加和膜电位的更新、脉冲发放这三种计算。

流水线一次并行计算 4 个神经元的膜电位，也就是一次能产生 4 个神经元的发放结果。如果发放，则向脉冲移位寄存器中写入 1，否则写入 0。脉冲移位寄存器的作用是将脉冲压缩为二值序列输出。输出后的脉冲将被暂存，用于后续脉冲事件的发送。

在所有神经元发放完毕后，神经元模块会向控制模块发送突触后脉冲事件，用于标志当前步长的计算已经完成。如果神经元处于学习模式，还会将脉冲事件复制相应的份数，分别发送给对应的突触核心，用于突触核心内部脉冲记录的更新。

本文之所以使用多级寄存的方式实现神经元计算，是因为希望流水线中的乘累加运算被综合工具自动推导为 FPGA 中的数字信号处理（Digital Signal Process，DSP）单元。通过合理的时序设计，该流水线的综合报告与预期一致，共使用了 8 个 DSP 单元。流水线中的所有加法器和乘法器都被 DSP 所代替。

3.6 控制模块

控制模块可以实现对整个形态硬件的控制，用于和外部程序的交互。其结构图 3-21 所示。

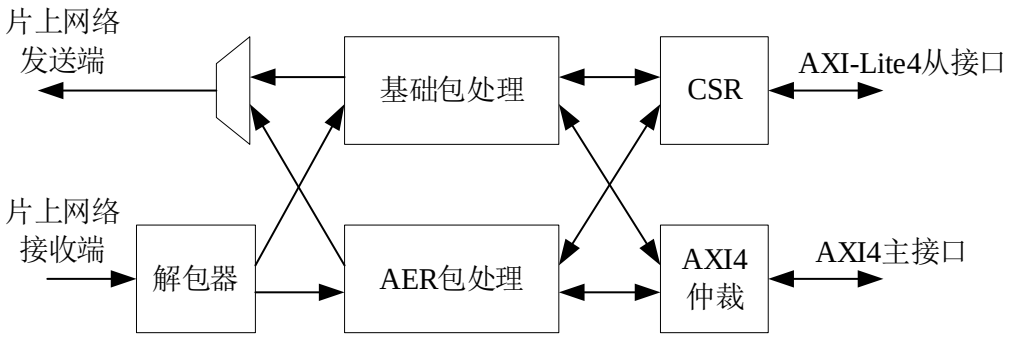

图 3-21 控制模块

在控制模块中，通过 AXI4 接口可以直接访问片外 DRAM 的地址空间。外部程序通过 AXI-Lite4 接口读写控制状态寄存器（Control and Status Register，CSR）可以对控制模块直接进行配置信息的写入，或者向网络中发送基础包。基础包中携带其它核心的配置信息，通过片上网络对其它核心进行配置的写入。基础包的发送和接收主要由基础包处理模块完成。控制模块的另一个重要功能是处理接收到的事件信息。AER 包处理模块负责接收并处理由突触核心发送来的取权重事件和由神经元核心发送来的突触后脉事件，并且还负责发送突触前脉冲事件用于触发突触核心的计算。

3.6.1 基础包处理单元

基础包处理单元主要功能是向网络中发送基础协议的网络包。由于片上网络具有延迟，一个命令包发送后短时间内无法收到响应包，所以基础包处理单元中必须为基础包的发送提供未响应深度，即陆续发送多个包无需等待之前的包响应。但

因为片上网络到各个核心的延迟不同，支持未响应深度会带来返回包的顺序与发送顺序不同的问题。本文采用基于队列的方式提供了基础包发送的未响应深度，并且支持响应包的乱序返回，处理方式见图 3-22。

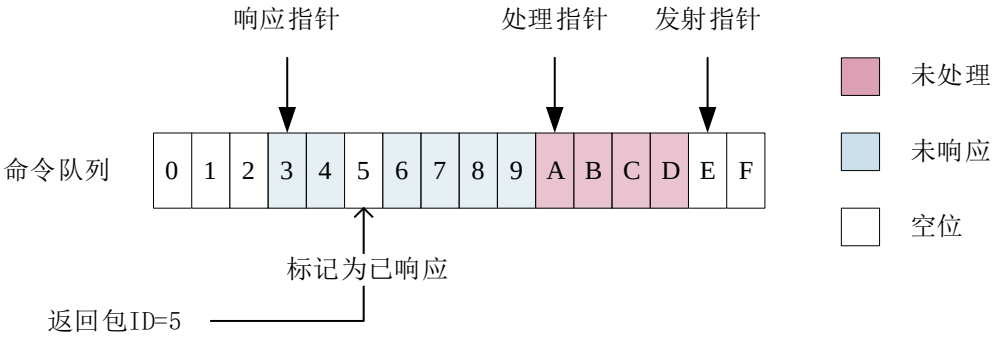

图 3-22 基于队列的基础包处理

基础协议中提供了使用 ID 字段作为包的唯一标识可用于解决乱序问题。在本处理单元中未响应深度被设定为 16，可以在未响应的情况下连续发送 16 个命令。当命令队列存在空位，即发射指针未超过响应指针，则可以插入待处理的命令。处理指针按照插入顺序依次处理有效的命令。当收到返回包时，根据返回包的 ID 直接寻址命令队列，将对应的命令标记未失效。如果响应指针所在的命令失效，则自曾至有效命令处直到队列为空。

3.6.2 AER 包处理单元

AER 包处理单元能够主动发送突触前脉冲事件到其它核心。在发送前，二值编码的脉冲事件暂存于 DRAM 中。CSR 提供脉冲在 DRAM 中的首地址，和目的核心的片上网络 ID。AER 包处理单元将通过 AXI4 接口从 DRAM 中读取脉冲事件，将其发送到目的核心。

AER 包处理单元被动接收神经元产生的突触后脉冲事件。突触后脉冲事件必须让软件程序能够获取，于是脉冲将被写入提前配置好的指定 DRAM 地址中。脉冲完全写入 DRAM 后，AER 包处理单元将标志 CSR 中的状态寄存器，表示当前步长结束。程序端可以通过 CPU 的总线从 DRAM 中取得突触后脉冲。

对于性能要求较为严苛的是权重事件处理的逻辑。其中取权重事件会向 DRAM 发起读请求；写权重事件会向 DRAM 发起读请求。为了最大化利用 DRAM 的带宽，在事件足够的情况下，要求设计必须能够以流水线的方式处理事件，使得 AXI4 接口始终处于数据传输的状态。具体设计如图 3-23 所示。

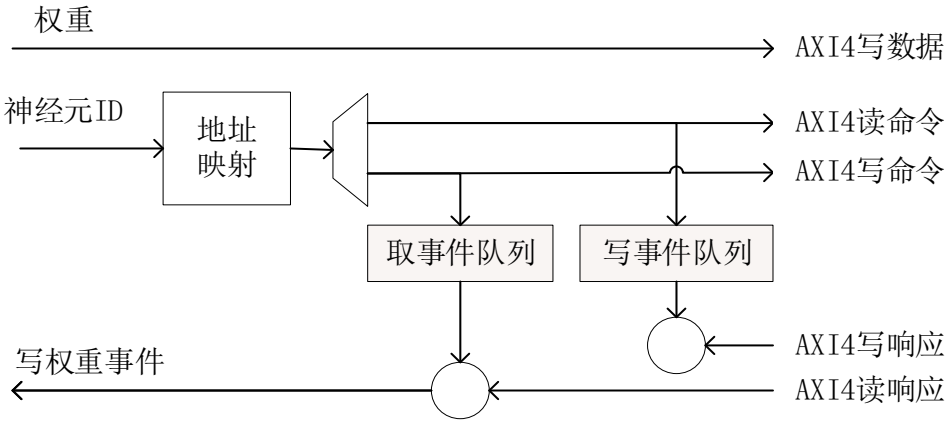

图 3-23 无阻塞权重读写

图中权重事件首先进行地址映射，将神经元 ID 映射为它的权重所在的地址。根据事件的类型，如果是取权重事件，则向该地址发送读命令；如果是写权重事件，则向该地址发送写命令。在发送命令的同时，将事件写入对应的队列。无需等待已发送命令的事件完成，可以继续处理下一个事件向 AXI4 发送命令。图中圆圈代表“汇聚”，即当两个基于握手协议的数据流输入都有效时，消耗两个输入，产生一个输出。AXI4 的写响应会消耗掉队列中的一个写事件，由于写事件没有后续操作，所以消耗后没有输出。AXI4 的读响应会消耗队列中的一个取事件，与取事件合并转化为写权重事件，将该事件将会被传送回对应的突触核心。

3.7 本章小结

本章自顶向下地介绍了整个神经形态芯片，先介绍了完整系统的结构，之后在每个小节对局部模块进行详细介绍。在总体介绍章节，清晰地展示了芯片包含的核心数量及其类型。简单明了地介绍了多个核心之间如何配合完成一次 SNN 推理的过程。

本章在主要贡献中指出了 SNN 中所蕴涵的时空局域性，提出可以使用缓存架构优化神经形态硬件，达到加速、减少资源消耗的效果。此外，创新地提出了基于突触前脉冲驱动的 STDP 零开销惰性计算方案，能够完美地契合使用了缓存的神经形态硬件。

为了加入缓存到神经形态硬件中，需要做很多架构上的定制设计，这一主线贯穿本章的每一个小节。本系统从顶层进行规划，在通信协议中为定制了适用于缓存架构的 AER 协议，包含多种事件类型；在突触核心中提出了事件驱动的缓存控制和多端口缓存的解决方案；在突触核心的突触计算流水线中实现了零开销的 STDP

计算；在神经元核心中实现了简洁的脉冲整合发放流水线；控制核心不仅为外部程序提供了高效便捷的控制接口，还实现了无阻塞的权重存取。

第四章 测试实验与结果分析

本章对本文提出的神经形态硬件进行应用上的实验。主要内容包括：硬件指标报告；基准测试；在线学习实验。其中硬件指标报告主要反映本文提出的架构对硬件资源的使用情况；基准测试通过与相关工作对比，确保本设计达到该领域的前沿水平；在线学习实验的目的是验证本文提出的 STDP 零开销惰性计算方案的正确性。

4.1 硬件指标报告

实验使用 ZYNQ UltraScale+型号的 FPGA。开发板的片外存储器为 4 GB 容量的 DDR4，在实验中的主频为 1200 MHz，位宽为 64 bit。经过综合和布局布线之后，硬件资源使用情况如表 4-1。

表 4-1 资源占用表

资源	使用量	总量	使用率
LUT	19964	341280	5.85%
LUTRAM	2494	184320	1.35%
FF	16247	682560	2.38
BRAM	176.5	744	23.72%
DSP	8	3528	0.23%

BRAM 是 FPGA 内部的主要存储资源。在本文的硬件中，4 个突触核心是存储资源的主要消耗者。BRAM 用于构建突触核心的缓存块。未经过 FGPA 综合软件优化的情况下，每个突触核心占用 128 个 BRAM。整个硬件用于缓存的存储资源为 512 KB。

4.2 基准测试实验

为了验证芯片的推理功能，本实验中使用了基于 MNIST 数据集的分类任务作为基准测试。MNIST 数据集是一个手写数字识别数据集，由于它的规模较小, 可以在许多神经形态芯片上进行快速处理，而且在处理时也可以采用较简单的神经网络结构。因此，MNIST 数据集被广泛应用于测试各种神经形态芯片的性能和功能。此外，由于有很大部分的神经形态芯片的设计工作采用 MNIST 数据集作为基准测试，因此使用 MNIST 数据集可以方便与不同的研究工作进行比较，从而更好地评估本文所提出的架的性能和功能。

对于本工作而言，作为一个神经形态硬件，应当满足基础的准确率和推理速度要求。由于本文的硬件上采用定点数据代替图形处理单元（Graphic Processing Unit，GPU）中的浮点运算，会导致计算精度下降，影响模型的准确率，所以在硬件上准确率的损失也是一个很重要的指标。实验预期的结果是以上三个指标达到相关工作的近似或者更高的水平。

满足准确率和推理速度只是本文架构作为神经形态硬件的基础要求。本文的创新点主要体现在缓存架构的设计所带来的优化效果。所以与其它工作不同，在基准测试中本文还添加了缓存优化效果的量化分析，讨论缓存带来的加速倍率和片上存储资源的节约量。

4.2.1 实验过程

用于实验的 SNN 模型如图 4-1 所示，一共包含 3 层 LIF 神经元，每层直接采用全连接的方式连接。

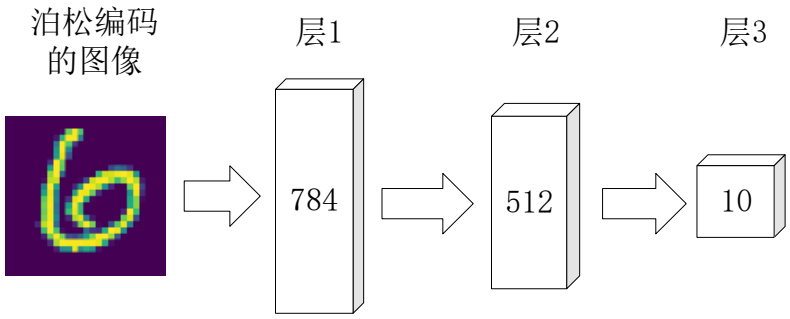

图 4-1 推理实验模型

模型在硬件上各个突触核心上的映射方案如表 4-2 所示。

表 4-2 突触核心负载映射

突触核心	突触前神经元	突触后神经元
0		
1	层 1：0-408	层 2：0-512
2	层 1：408-768	层 2：0-512
3	层 2：0-512	层 3：0-64

在模型的负载分配中，虽然每个核心能够容纳 1024 个突触前神经元，但是考虑到网络浅层的神经元发放的时空局域性较弱，为了达到缓存的最优效率，应当减少映射到缓存上的神经元总量。所以将网络的第一层拆分至两个突触核心进行负载。突触核心 3 单独负载第二层神经元的突触。虽然最后一层只有 10 个神经元，

但是在硬件部署中必须拓展为 64 个，因为在硬件设计过程中对神经元部署有最少数量的要求。最终一共使用了 3 个核心用于部署本实验的网络。

在模型进行推理之前，首先要对输入进行处理，将灰度图像转化为脉冲序列。本实验选用泊松编码[57]的方式对图像进行转换。在 SNN 中，神经元的激活可以被视为一系列离散时间的脉冲信号。泊松编码是一种常用的编码方式，用于将连续的信号转换为离散的脉冲信号，以便在 SNN 中进行处理。泊松编码的优点是可以有效地表示连续的信号，并且具有较低的计算复杂度。另外，由于泊松编码是一种随机编码方式，可以有效地抑制信号噪声和干扰，提高了 SNN 的鲁棒性。

泊松编码的基本原理是将连续的信号通过随机的脉冲序列进行表示，其中脉冲序列的频率与信号的幅值成正比。具体而言，假设某个信号的最大值为A，采样周期为T，泊松编码器会生成一系列随机的脉冲，使得每个脉冲出现的概率与信号的幅值成正比。通常情况下，脉冲的出现概率可以表示为：

$$P(t) = \frac{x(t)}{A} \tag{4-1}$$

其中，$P(t)$表示在时刻t处脉冲出现的概率，$x(t)$表示被编码信号在时刻t处的幅值。在 SNN 中，将输入的灰度值作为$x(t)$，使用无状态的泊松编码器进行多次求值，即可将该灰度值转化为一个二值的脉冲序列。最终输入的脉冲序列如图 4-2（b）所示。

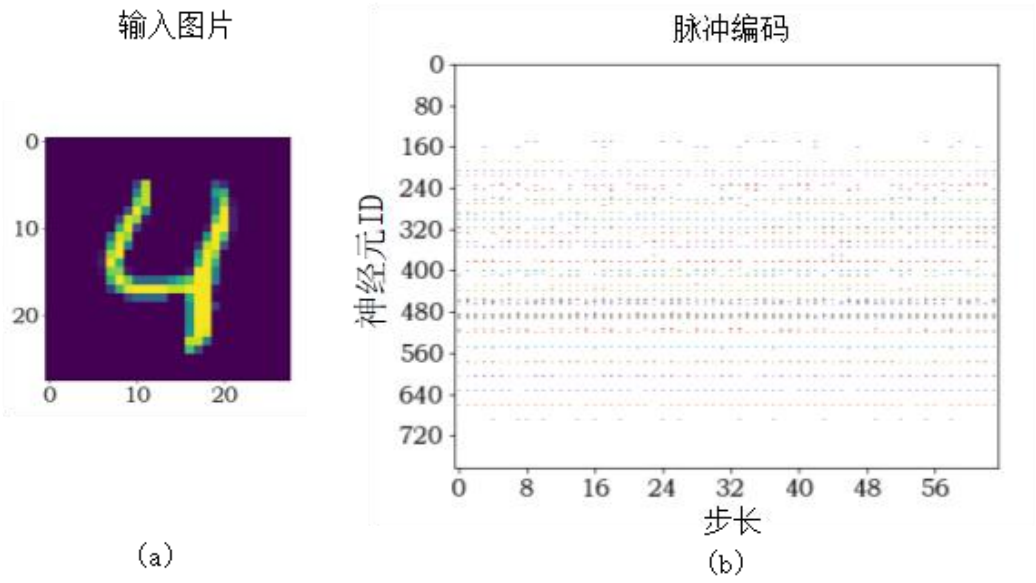

图 4-2 泊松编码。(a)被编码的输入图片；(b)脉冲编码结果

如图所示，进行编码时，首先将二维图像展平为一维的序列，每个神经元通过 ID 进行标识，然后对每个像素用编码器进行 64 个周期的连续求值，得到硬件上最终可输入到 SNN 中的脉冲序列。

模型的权重在实验之前已经训练完成。数据集中 6 万张图片用于训练，1 万张图片用于测试。模型中神经元的参数配置见表 4-3。

表 4-3 神经元参数

时间常数	复位电平	阈值	编码长度
2	0	1	100

4.2.2 推理性能分析

为了评估模型在硬件上进行推理的效果，实验中将相同 SNN 模型的软件运行结果作为基准，与本文的神经形态硬件的输出作比较。希望本文硬件上的准确率损失在可接受的范围内，且推理速度超过 GPU。软件端的模型使用 SpikingJelly 框架[58]编写，并使用 GPU 运行，GPU 型号为 NVIDIA GeForce GTX 960。实验结果如表 4-4 所示。

表 4-4 推理性能对比（软件）

平台	数据类型	准确率	处理速度（帧/秒）
软件	Float 32	98.74%	27
本工作	Int 16	98.29%	50

从表中可见，在部署到硬件之后，模型准确率的损失在 1%以内。并且本工作的处理速度已经超过了 GPU 在批处理量为 1 时的处理速度。

为了准确评估本文硬件与其它相关工作的差异，在用于对比的相关工作中，选取同样采用片外存储架构的神经形态处理器，并且模型均为全连接类型，均使用 MNIST 数据集作为测试基准，实验过程均为片外学习之后进行片上部署。推理性能对比见表 4-5。

表 4-5 推理性能对比（硬件）

工作	主频	模型结构	准确率	准确率损失	处理速度（帧/秒）
本工作	100 MHz	784-512-10	98.29%	0.45%	50
Li S[30]	100 MHz	784-200-200-10	92.93%		317
Darwin[34]	25 MHz	784-500-500-10	93.8%		6.25
Minitaur[36]	75 MHz	784-500-500-10	92%	2.2%	6.57
Han J[59]	200 MHz	784-1024-1024-10	97.06%	1.42%	161

表中的模型结构中每个数字代表网络中每层的神经元个数。与其它实验采用四层网络不同，本实验采用三层网络，因为三层网络的准确率已经足够高。表中准确率代表的是模型在硬件上部署后的准确率。虽然本实验的准确率在本文所选的相关工作中达到最高，但是这是由于用于训练的软件框架表现优异，其它工作可能受限于当时的软件训练框架效果较差，无法达到更高的准确率。为了更好地评估硬件上的准确率指标，表中列举了每个工作在将训练好的模型部署到硬件后的准确率损失。可以发现本工作的准确率损失是最小的。此外，在推理速度上，本工作达到了 50 帧每秒的算力。虽然推理速度与一些工作尚有差距，但是本工作的优点主要是节约片上的存储成本，在推理速度方面满足应用的实际要求即可。

4.2.3 缓存优化效果分析

模型部署在硬件上的准确率指标合格只是神经形态硬件的基础要求，为了评估本文提出的架构的作用，需要对缓存的指标进行分析，量化优化效果。

在传统的计算机体系结构的量化分析[61]中，采用存储器平均访问时间这一指标对缓存的效用进行量化。量化公式如式(4-2)所示：

$$C = C_h + M_r C_m \tag{4-2}$$

其中，C为总的代价，一般指存储器的平均访问时间；C_h为命中代价，指缓存查询过程中花费的时间；M_r为缺失率，C_m为缺失代价，指缓存缺失的时候从 DRAM 中取数据的时间。

但是传统计算机的缓存量化公式无法照搬到神经形态硬件中。在本文的架构中，通过流水线的设计，完全抵消了缓存的命中代价。CPU 缓存每次缺失的时候都从 DRAM 中读取固定长度的缓存行，缺失代价是固定的。而突触核心从 DRAM 中读取的缓存行的大小取决于突触后神经元的个数。在不同的配置下，每个核心的权重行长度也不同，导致取权重时对 DRAM 端口的占用时长不同。所以本文不计式(4-2)的命中代价，并设置不同的缺失代价，对缓存的效用进行量化分析。

表 4-6 缓存量化分析

突触核心	脉冲数/帧	命中率	缺失代价	总代价
1	3085	98.8%	128	47×10^3
2	3005	98.9%	128	42×10^3
3	5378	72.5%	16	24×10^3

表 4-6 统计了核心平均每帧图像处理的脉冲数量、命中率、缺失代价和总代价。结合表 4-2 中各个核心的负载情况进行分析。突触核心 1 与 2 中每个权重行包含

512 个神经元的权重，需要 128 个时钟从 DRAM 中读取，所以缺失代价为 128。核心 3 虽然缺失代价较少，但是由于只使用了一个核心负载整层网络，导致命中率较低。不过，核心 3 的总代价并未比其它核心高。

实验中将模型的第一层拆分到核心 1 和 2 进行负载，如果两个核心的实际工作负荷不同，会导致命中率的不同。由于采用多核心并行计算的方式，整个硬件系统的性能是由代价最高的核心决定，最优的情况是各个核心的总代价相近，并且保持一定的命中率。本文在设计之初即考虑到这一问题，突触核心的负载被设计为可灵活配置。负载的分配并不是简单地均分，而是通过预实验统计缓存指标，总结得到表 4-2 中较为理想的配置。

在本实验的较高命中率下，芯片的 3 个核心可以达到极高的利用率。64 位的 DRAM 端口最多提供每时钟 4 个突触权重，然而在片上实际能并行计算 12 个突触权重的累加，达到了 3 倍的加速效果。如果模型的规模更大，将 4 个核心都使用上，则能达到 4 倍的加速。

对于纯 SRAM 方案的神经形态硬件，运行该模型至少需要 794 KB 的片上存储空间。在本文的缓存架构中，只需要 272 KB 的有效缓存空间，相比节约了 65.7% 的片上存储成本。

4.3 在线学习实验

本文在缓存架构的神经形态硬件上实现了基于 STDP 的在线学习功能。为了验证该功能的有效性，希望在实验中能够复现 STDP 的学习曲线。

实验将硬件中的一个核心设置为学习模式，并部署了一层 SNN 的全连接层。本文的 STDP 采用了惰性计算的方案，只有突触前脉冲能够触发。实验中通过发送脉冲的方式，在突触核心的脉冲发放记录中构造不同时间间隔的脉冲对，在最后时刻以一组突触前脉冲触发 STDP 的计算。之后将突触核心清空，权重的变化值将被写回到 DRAM。

图 4-3 显示了用于触发 STDP 而构造的突触前后脉冲对。在 LTD 脉冲对的构造中，在步长最后由一个突触前脉冲与各个历史上发放过的突触后脉冲两两组合，构造不同的时间差；在 LTP 的构造中，由于 LTP 无法由突触后脉冲触发，所以需要在最后一个步长，给每个神经元发送突触前脉冲，触发 LTP 的计算。最终得到的 STDP 学习曲线如图 4-4 所示。图中横轴表示突触前后脉冲的时间间隔，纵轴表示权重变化值。可以看到，权重变化的幅值随着突触前后脉冲时间间隔的绝对值以指数形式衰减，符合 STDP 学习规则的预期。实验结果表明，本文的硬件能够正确地产生 STDP 的学习曲线，为实现在线学习奠定了基础。

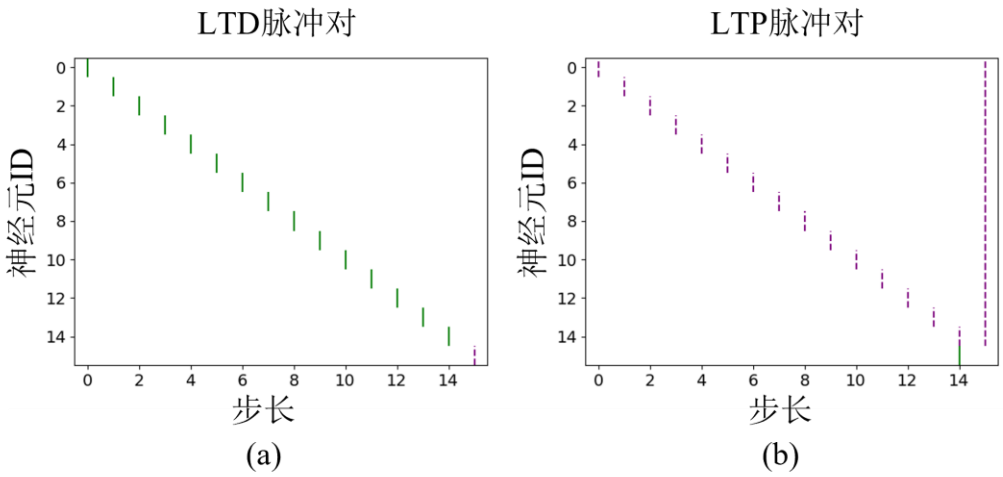

图 4-3 不同间隔的脉冲对，其中绿色实线为突触后脉冲，紫色虚线为突触前脉冲。(a)用于触发长时程增强的脉冲对；(b)用于触发长时程抑制的脉冲对

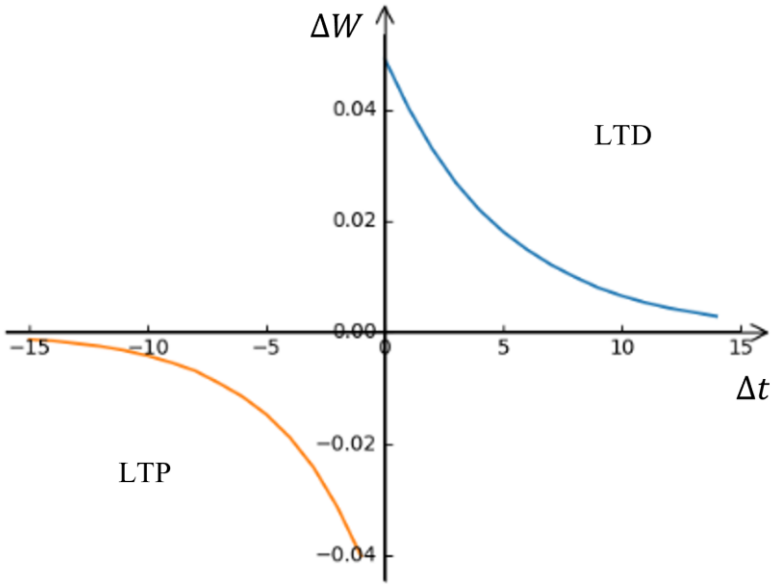

图 4-4 STDP 学习曲线

4.4 本章小结

本章首先报告了本文提出的神经形态硬件在 FPGA 上综合和布局布线后的硬件资源使用报告。然后搭建了用于实验的软硬件环境。进行了实验验证本文提出的神经形态硬件的功能的正确性以及性能和应用了缓存架构后所带来的优化效果。

在基准测试实验中，使用 SNN 对 MNIST 数据集进行分类。分类的准确率达到了相关工作的最高，并且准确率损失最小。处理速度达到 50 帧每秒。根据实验结果分析任，本文的缓存架构在神经形态硬件中能达到节约 60%以上片上存储的效果。

在在线学习的实验中，本文通过在硬件上复现 STDP 学习曲线的形式，验证了本文提出的突触前脉冲驱动的 STDP 惰性计算方案的正确性。

第五章 总结与展望

5.1 研究总结

传统的神经形态硬件采用片上存储方案存在容量限制，难以支持大规模的SNN 部署。为了解决这一问题，本研究提出了一种适用于神经形态硬件的缓存架构，通过有限的片上存储实现更大规模 SNN 的部署，并实现了基于 STDP 的在线学习。

本研究的目的是解决神经形态硬件中的存储容量限制问题，实现更大规模的SNN 部署，并在此基础上实现基于 STDP 的在线学习。本文想要证明缓存架构和突触前脉冲驱动的 STDP 零开销惰性计算能够提高 SNN 的计算效率和准确率。

本研究采用了自顶向下的方法，从整体上分析神经形态硬件系统的构造，提出了适用于事件驱动架构的缓存这一创新点，并进行了实验验证。本研究对相关工作进行了分类。在背景知识介绍中提供了脉冲神经网络、神经形态硬件、缓存原理的介绍。

在实验章节，本研究将硬件上的运行结果与软件端进行对比，推理的准确率损失在 1%以内，并在各项指标上取得了较好的结果。与其它相关工作对比，本研究在缓存架构在硬件成本上具有明显优势，与仅基于 SRAM 的架构相比，能减少 60%以上的片上存储资源。

5.2 未来展望

本研究对在神经形态硬件中引入缓存架构做了初步的尝试，取得了显著的效果。但缓存架构所能带来的优化并在本工作中其实发掘的并不充分。由于工作量过大的原因，本文对整个神经形态系统的设计并不充分。比如突触核心的计算效率较低，在步长迭代的时候存在较多的空闲周期。这是因为一个突触核心只负责一层网络，如果在突触核心中加入多线程设计，使其能够并行计算两层网络，则可以大大提高突触的计算效率，从而提高整个芯片的算力。此外，软件驱动的设计不够完善，上位机通过以太网对芯片进行控制具有较大延迟。虽然能够满足实验要求，但是离实际应用尚有一段距离。未来可以使用更加高效的控制接口，进一步发掘芯片的性能。

本文关于在线学习实验的篇幅较少，由于工作量较大的原因，没能在计划内完成神经元的群体学习实验。STDP 用于神经元群体学习效果较好的文章[62,63]一般有

自己的优化方案。但是碎片化的优化方案不利于硬件上的通用实现，所以本文在设计之初只采用了基础的 STDP 方案。但在实际的群体学习实验中，基础的 STDP 方案学习效果不佳。在本文中，已经验证了基于脉冲对的 STDP 可以在缓存架构中以惰性计算的方式实现零开销的在线学习。基于这个正确的架构，群体学习的功能有望在未来的工作中调试改进。

5.3 应用前景

本文设计的神经形态硬件以低资源的方式实现高性能的推理，这意味着它在计算和存储资源有限的情况下能够实现出色的推理能力。这种高效能的特点使得硬件可以应用于各种嵌入式系统和移动设备，满足对功耗和资源消耗严格的需求。同时，本文的硬件还支持在线学习，这是一个重要的优势。在线学习意味着硬件可以在运行过程中不断地更新和改进模型，从而适应不断变化的环境和数据。这种动态学习的能力使得硬件具备了自适应性和个性化的特点，可以根据用户的需求和反馈进行实时调整和优化。最重要的特点是缓存架构的使用大大减少片上存储的资源消耗，这意味着硬件上的成本可以足够低，对与于大规模应用，能节约的成本是巨大的。

对于边缘计算设备：在物联网和边缘计算领域，设备通常具有有限的资源和功耗要求。本文的神经形态硬件可以帮助实现低功耗、高效能的边缘设备，用于图像处理、语音识别、智能监控等应用。同时，支持在线学习的能力使得这些设备能够适应环境的变化和个性化需求。

对于智能家居：随着智能城市和智能家居的发展，对于具有实时分析和智能决策能力的摄像头的需求不断增加。缓存的使用减少了存储的功耗，与 DVS 相结合，神经形态硬件可以用极低的功耗完成高效的图像处理和目标检测，实现实时监控、人脸识别、行为分析等功能，延长智能家居设备的续航能力。

对于医疗诊断和健康监测：在医疗领域，快速而准确的诊断对于病人的治疗和健康监测至关重要。本文的硬件可以用于生物信号处理等任务，提供高效的医疗辅助工具，改善诊断准确性和速度。同样的，超低功耗的续航能力对于可穿戴的医学监测设备及其重要。减少了存储器器的功耗意味着可以使用小型电池供电，减少设备的体积。

对于机器人和自动驾驶：在机器人和自动驾驶领域，实时感知和决策至关重要。目前基于 DVS 事件流的视觉深度估计和光流分析，对提高自动驾驶的决策能力有重要作用。本文的神经形态硬件能够提供高性能的脉冲事件处理能力，适应自动驾驶环境中较大的数据吞吐量。

参考文献

[1] Lazzaro J, Wawrzynek J, Mahowald M, et al. Silicon auditory processors as computer peripherals[J]. Advances in Neural Information Processing Systems, 1992, 5.

[2] Drepper U. What every programmer should know about memory[J]. Red Hat, Inc, 2007, 11(2007): 2007.

[3] Neftci E O. Data and power efficient intelligence with neuromorphic learning machines[J]. Iscience, 2018, 5: 52-68.

[4] Zenke F, Bohté S M, Clopath C, et al. Visualizing a joint future of neuroscience and neuromorphic engineering[J]. Neuron, 2021, 109(4): 571-575.

[5] Shrestha A, Fang H, Mei Z, et al. A Survey on Neuromorphic Computing: Models and Hardware[J]. IEEE Circuits and Systems Magazine, 2022, 22(2): 6-35.

[6] Feldman D E. The spike-timing dependence of plasticity[J]. Neuron, 2012, 75(4): 556-571.

[7] Morrison A, Diesmann M, Gerstner W. Phenomenological models of synaptic plasticity based on spike timing[J]. Biological Cybernetics, 2008, 98: 459-478.

[8] Bouvier M, Valentian A, Mesquida T, et al. Spiking neural networks hardware implementations and challenges: A survey[J]. ACM Journal on Emerging Technologies in Computing Systems (JETC), 2019, 15(2): 1-35.

[9] Hulea M, Uleru G I, Caruntu C F. Adaptive SNN for anthropomorphic finger control[J]. Sensors, 2021, 21(8): 2730.

[10] Falanga D, Kleber K, Scaramuzza D. Dynamic obstacle avoidance for quadrotors with event cameras[J]. Science Robotics, 2020, 5(40): eaaz9712.

[11] Sharifshazileh M, Burelo K, Sarnthein J, et al. An electronic neuromorphic system for real-time detection of high frequency oscillations (HFO) in intracranial EEG[J]. Nature Communications, 2021, 12(1): 3095.

[12] Bellec G, Scherr F, Subramoney A, et al. A solution to the learning dilemma for recurrent networks of spiking neurons[J]. Nature Communications, 2020, 11(1): 3625.

[13] Zenke F, Neftci E O. Brain-inspired learning on neuromorphic substrates[J]. Proceedings of the IEEE, 2021, 109(5): 935-950.

[14] Marschall O, Cho K, Savin C. A unified framework of online learning algorithms for training recurrent neural networks[J]. The Journal of Machine Learning Research, 2020, 21(1): 5320-5353.

[15] Neftci E O, Mostafa H, Zenke F. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks[J]. IEEE Signal Processing Magazine, 2019, 36(6): 51-63.

[16] Zhang T, Cheng X, Jia S, et al. Self-backpropagation of synaptic modifications elevates the efficiency of spiking and artificial neural networks[J]. Science Advances, 2021, 7(43): eabh0146.

[17] Du J, Wei H, Wang Z, et al. Long-range retrograde spread of LTP and LTD from optic tectum to retina[J]. Proceedings of the National Academy of Sciences, 2009, 106(45): 18890-18896.

[18] Taherkhani A, Belatreche A, Li Y, et al. A review of learning in biologically plausible spiking neural networks[J]. Neural Networks, 2020, 122: 253-272.

[19] Ponulak F, Kasiński A. Supervised learning in spiking neural networks with ReSuMe: sequence learning, classification, and spike shifting[J]. Neural Computation, 2010, 22(2): 467-510.

[20] Dimitrakopoulos G, Psarras A, Seitanidis I. Microarchitecture of Network-on-chip Routers[M]. Berlin, Germany: Springer, 2015.

[21] Davies M, Srinivasa N, Lin T H, et al. Loihi: A neuromorphic manycore processor with on-chip learning[J]. IEEE Micro, 2018, 38(1): 82-99.

[22] Stewart K, Orchard G, Shrestha S B, et al. Online few-shot gesture learning on a neuromorphic processor[J]. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 2020, 10(4): 512-521.

[23] Viale A, Marchisio A, Martina M, et al. Carsnn: An efficient spiking neural network for event-based autonomous cars on the loihi neuromorphic research processor[C]//2021 International Joint Conference on Neural Networks (IJCNN). IEEE, 2021: 1-10.

[24] Merolla P A, Arthur J V, Alvarez-Icaza R, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface[J]. Science, 2014, 345(6197): 668-673.

[25] Kim G, Kim K, Choi S, et al. Area-and energy-efficient STDP learning algorithm for spiking neural network SoC[J]. IEEE Access, 2020, 8: 216922-216932

[26] Moradi S, Qiao N, Stefanini F, et al. A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs)[J]. IEEE Transactions on Biomedical Circuits and Systems, 2017, 12(1): 106-122.

[27] 赵琨鹏.基于连续吸引子神经网络的神经形态电路系统研究[D].电子科技大学, 2022.004156.

[28] Frenkel C, Lefebvre M, Legat J D, et al. A 0.086-mm2 12.7-pJ/SOP 64k-synapse 256-neuron online-learning digital spiking neuromorphic processor in 28-nm CMOS[J]. IEEE Transactions on Biomedical Circuits and Systems, 2018, 13(1): 145-158.

[29] 徐志康. 脉冲神经网络在线学习处理核设计与实现[D].电子科技大学,2020.001774.

[30] Li S, Zhang Z, Mao R, et al. A Fast and Energy-Efficient SNN Processor With Adaptive Clock/Event-Driven Computation Scheme and Online Learning[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2021, 68(4): 1543-1552.

[31] Guo S, Wang L, Wang S, et al. A systolic SNN inference accelerator and its co-optimized software framework[C]//Proceedings of the 2019 on Great Lakes Symposium on VLSI. 2019: 63-68.

[32] Panchapakesan S, Fang Z, Li J. SyncNN: Evaluating and accelerating spiking neural networks on FPGAs[J]. ACM Transactions on Reconfigurable Technology and Systems (TRETS), 2022.

[33] Cassidy A S, Georgiou J, Andreou A G. Design of silicon brains in the nano-CMOS era: Spiking neurons, learning synapses and neural architecture optimization[J]. Neural Networks, 2013, 45: 4-26.

[34] Ma D, Shen J, Gu Z, et al. Darwin: A neuromorphic hardware co-processor based on spiking neural networks[J]. Journal of Systems Architecture, 2017, 77: 43-51.

[35] Saha S, Duwe H, Zambreno J. An adaptive memory management strategy towards energy efficient machine inference in event-driven neuromorphic accelerators[C]//2019 IEEE 30th International Conference on Application-specific Systems, Architectures and Processors (ASAP). IEEE, 2019, 2160: 197-205.

[36] Neil D, Liu S C. Minitaur, an event-driven FPGA-based spiking network accelerator[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2014, 22(12): 2621-2628.

[37] Schuman C D, Potok T E, Patton R M, et al. A survey of neuromorphic computing and neural networks in hardware[J]. ArXiv Preprint ArXiv:1705.06963, 2017.

[38] Gerstner W, Kistler W M, Naud R, et al. Neuronal dynamics: From single neurons to networks and models of cognition[M]. Cambridge University Press, 2014.

[39] Liu Y, Yenamachintala S S, Li P. Energy-efficient FPGA spiking neural accelerators with supervised and unsupervised spike-timing-dependent-plasticity[J]. ACM Journal on Emerging Technologies in Computing Systems (JETC), 2019, 15(3): 1-19

[40] 武长春,周莆钧,王俊杰等.基于忆阻器的脉冲神经网络硬件加速器架构设计[J].物理学报,2022,71(14):304-312.

[41] Zhao J, Kim Y B. Circuit implementation of FitzHugh-Nagumo neuron model using field programmable analog arrays[C]//2007 50th Midwest Symposium on Circuits and Systems. IEEE, 2007: 772-775.

[42] Payvand M, Fouda M E, Kurdahi F, et al. On-chip error-triggered learning of multi-layer memristive spiking neural networks[J]. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 2020, 10(4): 522-535.

[43] Han I S. Biologically inspired hardware implementation of neural networks with programmable conductance[C]//2007 International Joint Conference on Neural Networks. IEEE, 2007: 2336-2340.

[44] Vogelstein R J, Tenore F V G, Guevremont L, et al. A silicon central pattern generator controls locomotion in vivo[J]. IEEE transactions on biomedical circuits and systems, 2008, 2(3): 212-222.

[45] Khalifa K B, Girau B, Alexandre F, et al. Parallel FPGA implementation of self-organizing maps[C]//Proceedings. The 16th International Conference on Microelectronics, 2004. ICM 2004. IEEE, 2004: 709-712.

[46] 李宏伟,吴庆祥.脉冲神经网络中神经元突触的硬件实现方案[J].计算机系统应用,2014,23(02):17-21

[47] Liu J, Wang C. A survey of neuromorphic engineering--biological nervous systems realized on silicon[C]//2009 IEEE Circuits and Systems International Conference on Testing and Diagnosis. IEEE, 2009: 1-4.

[48] Duong N, Zhao D, Kim T, et al. Improving cache management policies using dynamic reuse distances[C]//2012 45Th annual IEEE/ACM international symposium on microarchitecture. IEEE, 2012: 389-400.

[49] Kumar S, Zhao H, Shriraman A, et al. Amoeba-cache: Adaptive blocks for eliminating waste in the memory hierarchy[C]//2012 45th Annual IEEE/ACM International Symposium on Microarchitecture. IEEE, 2012: 376-388.

[50] Zebin T, Scully P J, Peek N, et al. Design and implementation of a convolutional neural network on an edge computing smartphone for human activity recognition[J]. IEEE Access, 2019, 7: 133509-133520.

[51] Lee C, Kosta A K, Zhu A Z, et al. Spike-flownet: event-based optical flow estimation with energy-efficient hybrid neural networks[C]//European Conference on Computer Vision. Springer, Cham, 2020: 366-382.

[52] Fang W, Yu Z, Chen Y, et al. Incorporating learnable membrane time constant to enhance learning of spiking neural networks[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021: 2661-2671.

[53] 张兆民. 高能效可重构脉冲神经网络处理器设计[D].电子科技大学,.2022.002623.

[54] 王春颜. 基于 FPGA 的多核片上网络平台设计[D].电子科技大学,2018.

[55] Salihundam P, Jain S, Jacob T, et al. A 2 Tb/s 6×4 Mesh Network for a Single-Chip Cloud Computer With DVFS in 45 nm CMOS[J]. IEEE Journal of Solid-State Circuits, 2011, 46(4): 757-766.

[56] Vasiljevic J, Bajic L, Capalija D, et al. Compute substrate for Software 2.0[J]. IEEE Micro, 2021, 41(2): 50-55.

[57] Donnelly D F, Panisello J M, D B oggs. Effect of sodium perturbations on rat chemoreceptor spike generation: implications for a poisson model[J]. Journal of Physiology, 2010, 511(1):301-311.

[58] Fang W, Chen Y, Ding J ,et al. SpikingJelly[EB/OL].2020. https://github.com/fangwei123456/spikingjelly.

[59] Han J, Li Z, Zheng W, et al. Hardware implementation of spiking neural networks on FPGA[J]. Tsinghua Science and Technology, 2020, 25(4): 479-486.

[60] Zhang G, Li B, Wu J, et al. A low-cost and high-speed hardware implementation of spiking neural network[J]. Neurocomputing, 2020, 382: 106-115.

[61] Hennessy J L, Patterson D A. Computer architecture: a quantitative approach[M]. Elsevier, 2011.

[62] Diehl P U, Cook M. Unsupervised learning of digit recognition using spike-timing-dependent plasticity[J]. Frontiers in Computational Neuroscience, 2015, 9: 99.

[63] Fang H, Zeng Y, Zhao F. Brain inspired sequences production by spiking neural networks with reward-modulated stdp[J]. Frontiers in Computational Neuroscience, 2021, 15: 612041.