基于可重构阵列架构的强化学习计算引擎

梁明兰[1,2] 王 峥[1] 陈名松[2]

[1](中国科学院深圳先进技术研究院 深圳 518055)
[2](桂林电子科技大学 桂林 541004)

摘 要 现有神经网络处理器已广泛应用于计算机视觉、自然语言处理等领域。然而，现有片上加速方案对控制领域的强化学习算法支持较少，而基于神经网络的强化学习是智能系统决策技术的核心。该文采用可重构阵列体系结构，通过片上配置、动作与奖励存储的系统设计方案，可实现多种神经网络算法的灵活部署，并支持强化学习使用模式。基于 65 nm CMOS 工艺的逻辑综合结果显示，处理器主频为 200 MHz 时，计算模块面积仅需 0.32 mm^2，计算功率约 15.46 mW。

关键词 人工智能；可重构阵列架构；强化学习；片上自我意识系统
中图分类号 TG 181 **文献标志码** A

A Reconfigurable Computing Engine for Neural Network-Based Reinforcement Learning

LIANG Minglan[1,2] WANG Zheng[1] CHEN Mingsong[2]

[1](*Shenzhen Institutes of Advanced Technology*, *Chinese Academy of Sciences*, *Shenzhen* 518055, *China*)
[2](*Guilin University of Electronic Technology*, *Guilin* 541004, *China*)

Abstract Current artificial intelligent (AI) engines are usually designed for specific supervised learning algorithms, which have been widely used in computer vision and natural language processing domains etc. However, very few AI engines have been designed to support on-chip reinforcement learning algorithms, which is the foremost algorithm kernel for decision-making subsystem of many autonomous systems. In this work, a coarse-grained reconfigurable array like AI computing engine has been designed for the deployments of both supervised and reinforcement learning through on-chip configuration, action Random Access Memory (RAM) and reward RAM. Logic synthesis at the design frequency of 200 MHz based on 65 nm CMOS technology reveals the physical statistics of the proposed engine of 0.32 mm^2 in silicon area, 15.46 mW in power consumption. The proposed on-chip AI engine facilitates the implementation of end-to-end perceptual and decision-making networks, which has great potentials in applications like autonomous driving, robotics and unmanned aerial vehicle.

Keywords artificial intelligence; coarse-grained reconfigurable array; reinforcement learning; on-chip self-awareness system

收稿日期：2018-07-17 **修回日期**：2018-08-13
基金项目：国家自然科学基金项目(61702493)；深圳市科技计划项目(KQJSCX20170731163915914)；SIAT 优秀青年创新基金项目(2017001)
作者简介：梁明兰，硕士研究生，研究方向为可重构强化学习加速器设计；王峥(通讯作者)，助理研究员，研究方向为智能驾驶感知决策系统设计、可重构深度学习处理器设计，E-mail：zheng.wang@siat.ac.cn；陈名松，教授，研究方向为光通信技术。

1 引 言

随着人工智能算法的快速发展与计算机硬件技术的进步，深度学习在智能系统感知与决策领域的广泛应用已逐渐变为现实[1]。当今深度学习算法的实现依赖基于中央处理器(CPU)与图形处理器(GPU)的计算平台，在其上运行神经网络计算的功耗庞大，计算时间较慢，给在物联网终端部署神经网络带来了较大挑战。为解决计算能力与功耗的问题，近年来工业界在定制人工智能处理器领域进行了广泛的探索，出现了诸如谷歌TPU、英特尔 Mobileye、寒武纪、深鉴科技等神经网络加速引擎的设计方案。现有大部分设计均是针对视觉、语音等感知算法的片上部署，而对智能系统决策技术的片上加速实现支持较少。然而，随着基于深度强化学习(Deep Reinforcement Learning)的 AlphaGo、DeepMind Atari 等智能体获得巨大成功，监督学习与强化学习的有机结合在智能系统中的意义愈发重大。可以预见，未来智能系统将具备感知、决策和执行一体化的能力[2,3]，并可以独立于主机进行训练与演化。因此，设计具备可重构能力并支持监督学习和深度强化学习的神经网络计算引擎成为下一代人工智能硬件设计的重要课题。

在学术界，随着卷积神经网络(CNN)[4-6]和深度神经网络技术的快速发展，越来越多研究学者提出了支持神经网络的加速运算定制芯片。例如，Chakradhar 等[7]提出了支持卷积神经网络的加速器；Temam[8]提出支持多层感知机的加速处理。另有一些针对某一特定应用的加速器也不断产生，如 Qadeer 等[9]提出了一种有效的促进卷积运算实现方式。Farabet 等[10]在 2012 年提出了可重构的数据流定制芯片。中国科学院计算技术研究所孵化寒武纪团队开发了国内第一款神经网络处理器，主要用于图像识别处理技术，获得了巨大的关注和反响[11-13]。但现有设计重点研究对特定神经网络，如卷积神经网络[14]、递归神经网络(RNN)[8]等的加速运算，却忽略了不同网络的组合应用情况，特别是针对深度强化学习的网络部署。近年来，随着深度学习技术的迅猛发展，强化学习表现出了优越的性能。2013 年，Mnih 等[15]提出的深度 Q 网络(Deep Q-Networks，DQN）算法模型是深度强化学习领域的开创性工作。2015 年，Mnih 等将卷积神经网络和强化学习结合实现感知和决策任务，在游戏平台 Atari 的一系列游戏中达到了超越人类的表现[16]。2016 年，谷歌基于强化学习的智能系统 AlphaGo 战胜了世界顶尖围棋高手[17]。在自动驾驶领域，谷歌与特斯拉运用强化学习的自主决策能力在无人驾驶领域达到了良好效果[18]。中国科学院深圳先进技术研究院李慧云研究员团队在 2017 年将深度强化学习应用于自动驾驶的研究也受到了广泛关注[19]。

为解决现有神经网络处理器对深度强化学习技术支持不足的问题，本文采用粗粒度可重构阵列(Coarse-Grained Reconfigurable Array，CGRA）架构设计一款支持监督学习和强化学习网络端到端部署的人工智能计算引擎。该引擎根据不同的网络结构，灵活配置运算模式、网络层数和神经元数目，可以部署基于监督学习的感知网络和基于强化学习的决策网络。基于 65 nm CMOS 工艺的逻辑综合结果表明，该芯片依靠其较快的计算吞吐量和较低的计算功耗，可以广泛运用于计算资源受限的场景。本文所设计神经网络处理器在片上支持感知决策一体化的端到端神经网络部署，在智能驾驶、机器人和无人机等智能系统中具有广泛的应用前景。

2 深度强化学习

深度强化学习是将深度学习与强化学习结合起来进而实现从感知(Perception)到动作(Action)

的端对端学习的一种全新算法。简单地说，就是和人类一样，首先输入视觉信息，然后通过深度强化学习网络直接输出动作。对于不同决策任务来说，都包含一系列的动作、观察和奖励值。智能体为了实现自主学习的目的，通过动作与环境状态的相互关系实时更新其强化学习网络的权重。强化学习流程如图 1 所示，具有训练流程和实行两种模式。本文采用状态和动作都是网络的输入形式强化学习网络，网络的输出值则是对应的状态和动作的 Q 值。在这里，主要阐述当智能体处于学习模式的训练流程：首先，智能体在当前状态 s_t 根据 ε-Greedy 策略随机选择一个动作，或根据 $Q_{\max}(s_t, a_t, \theta)$ 来选择当前状态下具有最大 Q 值的动作并执行。其中，a_t 为动作值；θ 为当前网络的参数。然后，来到新的状态 s_{t+1}，并根据 s_{t+1} 的好坏给出即时奖励 r 值。最后，将上述过程得到的 (s_t, a_t, r, s_{t+1}) 存储到样本池中，供后续对网络进行批训练，加快收敛过程。最终，由强化学习网络算出 s_{t+1} 的最大状态-动作值 $Q_{\max}(s_{t+1}, a_{t+1}, \theta)$，则得到状态 s 的目标值 $Target\_Q$，即公式(1)。

$$Target\_Q = r + \gamma Q_{\max}(s_{t+1}, a_{t+1}, \theta) \tag{1}$$

因此，根据现值和目标值得到最小平方差，则偏差就能通过均方差来求得，即求得目标函数 $L(\theta)$，如公式(2)所示。

$$L(\theta) = E\{[Target\_Q - Q(s_t, a_t, \theta)]^2\} \tag{2}$$

其中，$Q(s_t, a_t, \theta)$ 是当前状态 s_t 和所选择的动作值 a_t 作为网络输入的当前输出值；$Target\_Q$ 为下一状态 s_{t+1} 的最大网络输出值，随后通过随机递减方式更新强化学习网络参数。

3　强化学习计算引擎设计

3.1　粗粒度可重构阵列基本架构

基于粗粒度可重构阵列(CGRA)架构的处理器设计具有悠久的历史，其结合现场可编程逻辑门阵列(FPGA)技术的可重构能力与专用集成电路技术(ASIC)的快速计算能力，通过对

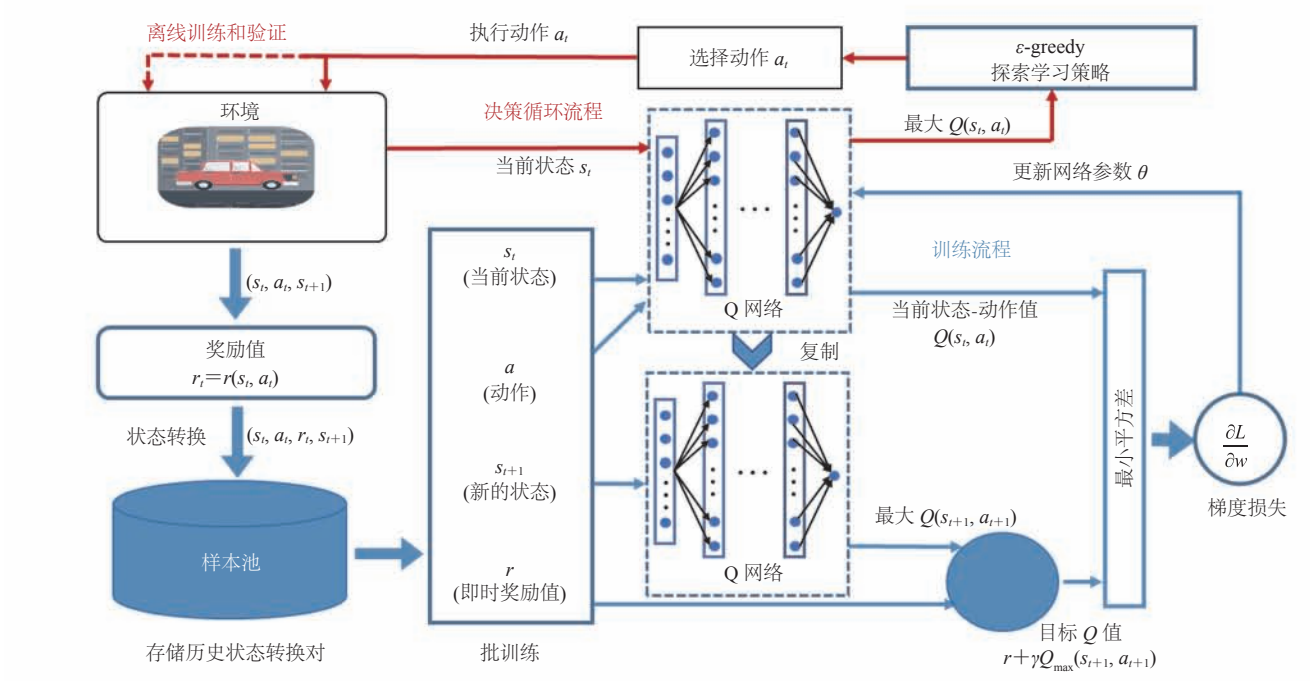


图 1　深度强化学习框架图

Fig. 1　Deep reinforcement learning framework diagram

计算单元之间互联模式进行配置，实现不同计算功能[20]。本文所提出的基于 CGRA 架构的神经网络计算引擎包括内存单元、输入输出单元、人工神经单元和控制单元，具体架构如图 2 所示。

内存单元包括数据内存、参数内存、配置内存、动作内存和奖励内存。其中，内存采用 SRAM 形式，访问速度快，灵活方便配置；数据内存主要存储神经网络的输入数据、层间数据和输出数据，由访问地址生成单元对数据进行灵活读写；参数内存存储神经网络参数；配置内存用于配置处理器的网络模式和网络结构，具有灵活操作特点；动作内存迭代模块针对强化学习模式而设计，主要对动作内存快速遍历循环调用神经网络得到动作值；奖励内存设计存储特殊状态值及其奖励值和一般状态的奖励值。

输入输出单元采用输入和输出移位寄存器实现，对数据的输入输出采用流入(Stream-in)、流出(Stream-out)的流水线形式。基于移位寄存器的流水线形式大幅度减少了数据传输交流次数，从而大大降低了输入输出数据功耗。

CGRA 的工作流程大体如下：首先，对配置内存、数据内存、参数内存、动作内存进行初始化；然后，根据配置内存指令配置网络模式、网络层数和神经元运算模式；最后，根据网络模式开始相应网络运算。其中，网络输入移位寄存器从数据内存读取数据，对每个神经元进行计算得到输出结果，并将结果存储到数据内存中并由下一层网络进行读取，直到完成最后一层运算得到输出结果。CGRA 架构通过各个模块协调工作，引入配置内存和动作内存，使得在网络模式、网络结构、动作机制上都具有可重构、可配置性，灵活实现如今快速迭代的神经网络算法，同时满足监督学习和强化学习，实现人工智能推理引擎。该系统对感知和决策有需求的领域发展具有积极促进意义。

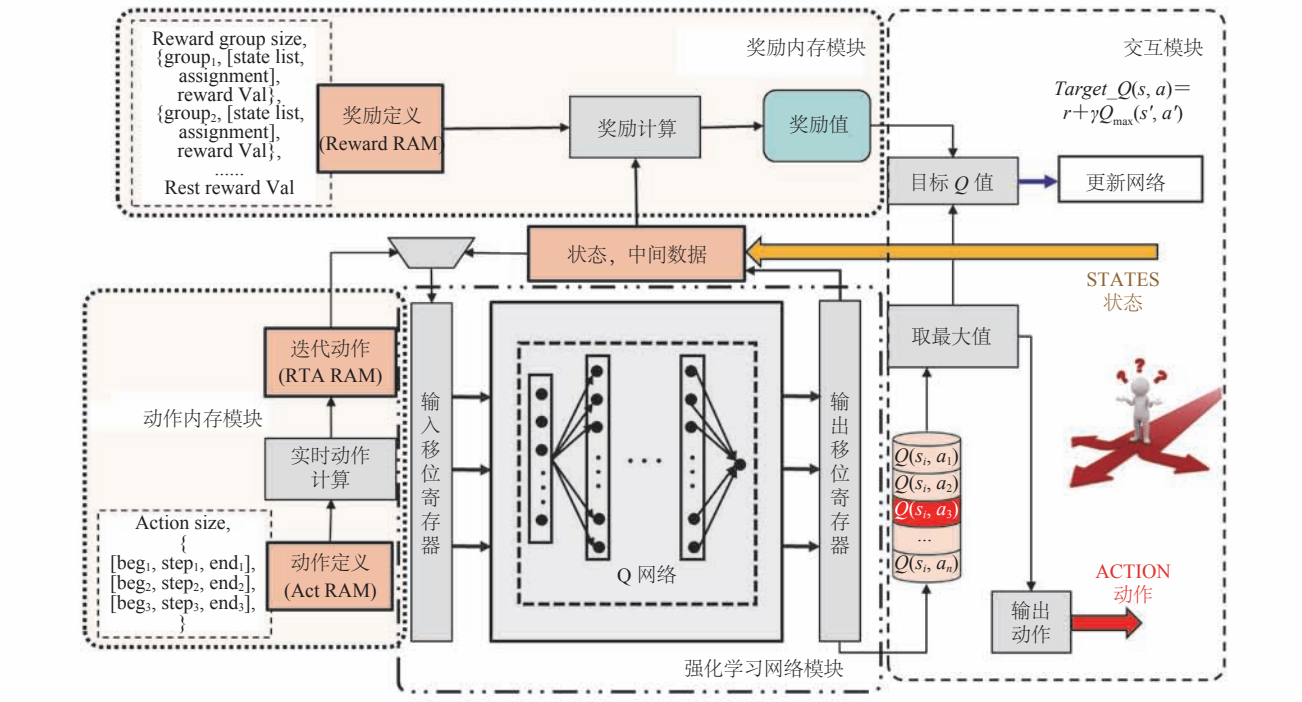


图 2 CGRA 对深度强化学习的架构支持

Fig. 2 CGRA architecture support for reinforcement learning

3.2　粗粒度可重构阵列对强化学习的架构支持

3.2.1 人工神经元设计

本文设计的人工神经元由控制模块、存储模块和基本计算模块构成，具体如图 3 所示。其中，控制模块包括模式配置链、神经元模式控制单元和地址生成控制单元；存储模块包括参数内存、临时内存 SRAM 和累加寄存器；基本计算模块包括乘法器、加法器和激活函数模块。具体操作步骤为：首先，通过配置、参数端口的串行输入控制数据以配置神经元，其中网络参数和配置数据分别存储于参数内存和模式配置链寄存器。然后，神经元在工作模式下，从数据输入端口获得数据，由地址生成模块从参数内存中找到与输入数据相匹配的参数供神经元的乘加模块（MAC）进行乘法和加成运算。此时，如果是强化学习网络模式，则将状态产生的临时贡献值存储到临时内存，与之后动作网络值对应相加；如果是基本网络模式，则将乘加模块运算的临时结果存于累加寄存器，当完成当前层神经元计算则由数据输出端口输出。本设计在神经元引入临时内存用于存储强化学习网络下状态对神经元的贡献值，供后续的动作节点运算一一对应累加使用，且重复利用，提高了数据使用率，减少运算成本。由此可看出，本神经元设计既满足了普通网络即监督学习的运算需要，又支持了强化学习这种特殊网络的运算模式。

3.2.2 状态机控制

本文的神经网络处理器部分状态转换机制如图 4 所示。此状态机首先根据转换条件完成不同状态的转换和对内存配置，其次是控制神经网络的运算流程。其中，模块 1 是对配置内存、参数内存、数据内存和动作内存根据算法需求进行特定配置；模块 2 是 128 级流水线的控制流程，包括指令的更新和解码内容；模块 3 是强化学习通过遍历动作内存实现 Q 快速迭代流程图。

3.2.3 动作内存设计

动作的快速片上遍历机制是 CGRA 架构实现片上决策的关键。本文动作内存（Act RAM）的设计方法如图 5 所示。为实现动作遍历，设定动作内存为 1 k 字节，而在内存中首先存储的是动作维数，之后存储的数据分别是第一维动作的步长值 $step_1$、最大值 end_1 和起始值 beg_1，这组数

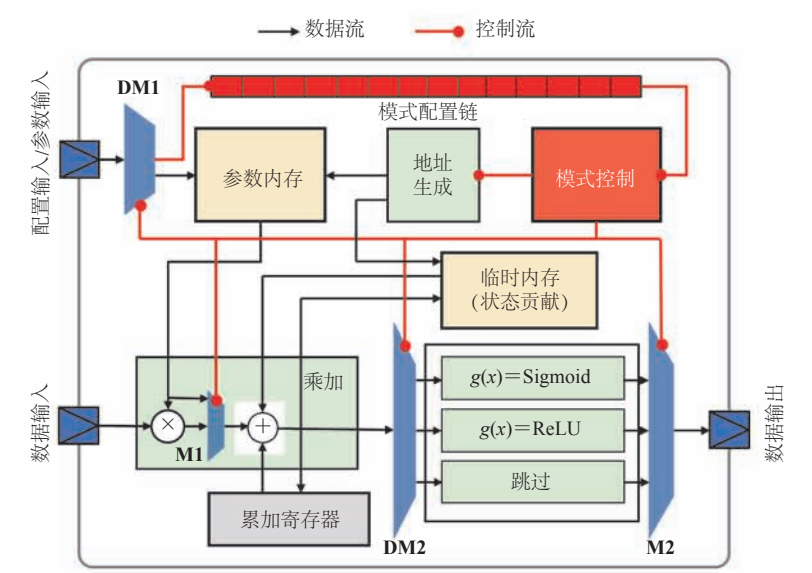


图 3　人工神经元架构图

Fig. 3　Architecture of artificial neuron

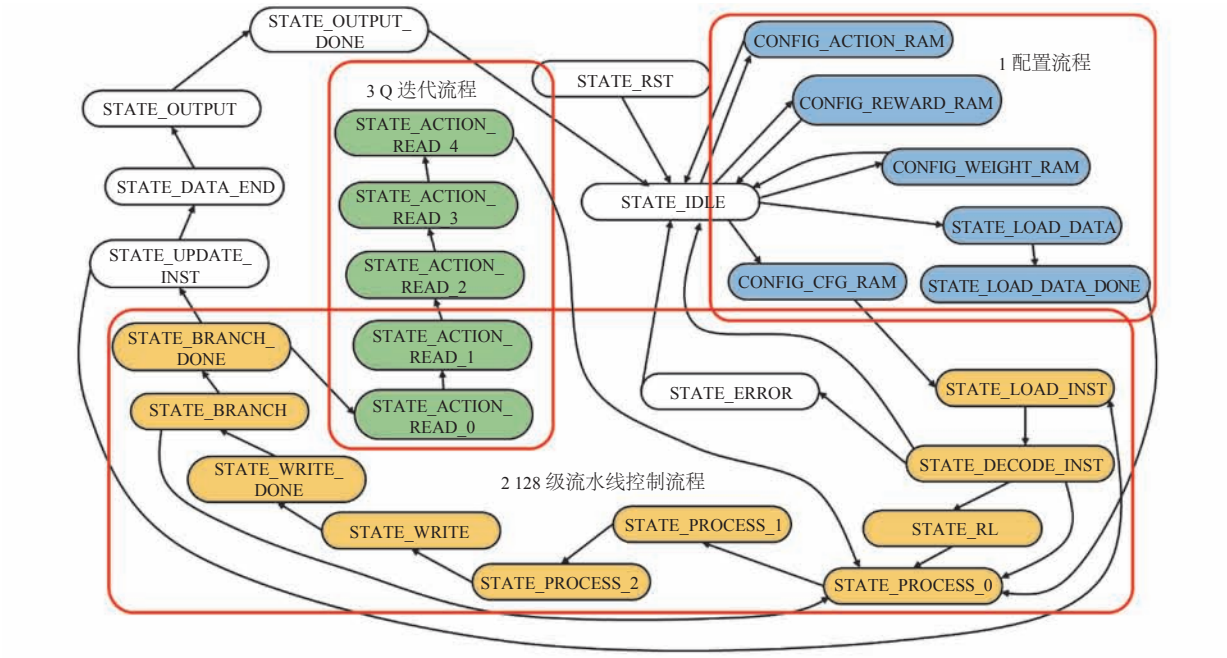


图 4　神经网络处理器中央控制器状态机

Fig. 4　Central controller state machine for neural network processor

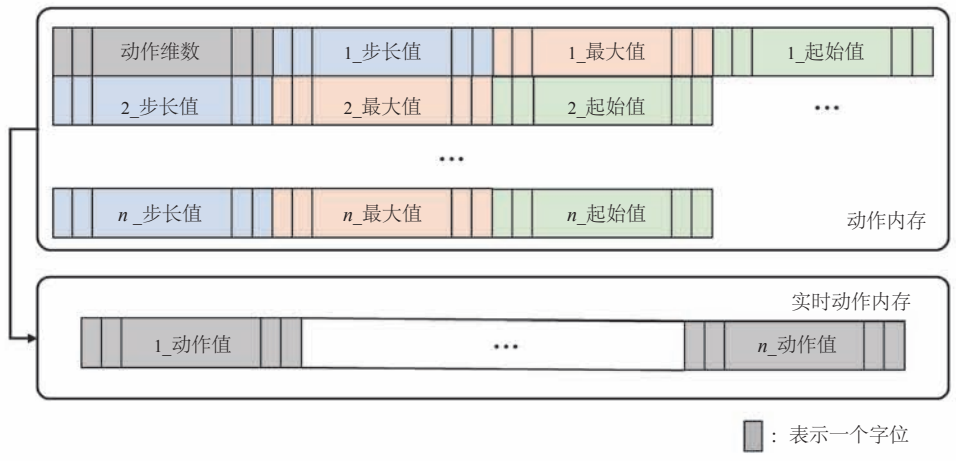


图 5　片上动作存储设计图

Fig. 5　Diagram of on-chip action storage

据决定了第一维动作的范围值，同理其他维动作构成一样。实验过程中，根据代码迭代控制处理器遍历动作内存：首先，将各维动作的初始值按地址控制模块分别存到实时动作内存中；其次，依次遍历每一维动作的取值，实时动作内存 RTA RAM 只需更新当前维动作值，其他不变，更新一次就将实时动作内存值依次顺序输入到神经网络运算；再次，将动作值的神经元贡献值与之前得到的状态贡献值对应相加，直到进行最后一层网络运算得到当前的状态-动作 Q 值；最后，继续更新动作值，重复以上流程，直到更新完最后一维的最后一个值为止，并运算完新的动作值组合，即可得到当前状态下的最大 Q 值。

Q 值选择主要通过以下流程图步骤对动作内

存遍历实现，具体如图 6 所示。

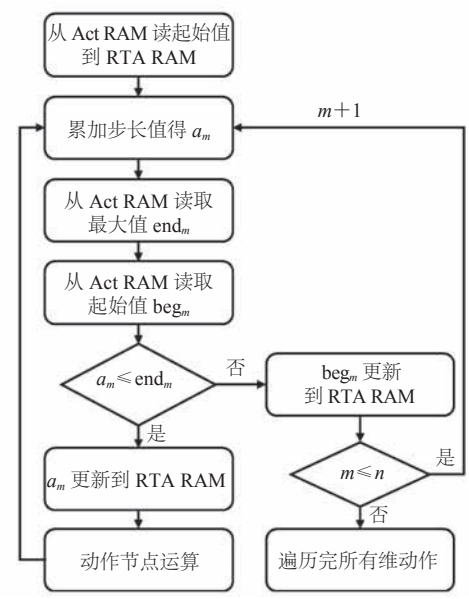


图 6　动作内存遍历流程图

Fig. 6　Transition diagram of on-chip action computation

（1）首先通过内存地址值将动作内存 Act RAM 中各维度动作的起始值，如 beg_1、beg_2 等存储到 RTA RAM 实时动作内存中。

（2）将根据地址控制器分别从 Act RAM 和 RTA RAM 读取步长值 $step_m$（m 表示动作维数）和起始值 beg_m，通过公式 $a_m = beg_m + step_m$ 累加步长值，得到当前时刻第 m 维动作的实时值 a_m。

（3）根据内存地址值从 Act RAM 读取最大值 end_m 并寄存。

（4）根据内存地址值从 Act RAM 读取起始值 beg_m 值并寄存。

（5）先判断状态 1 得到的 a_m 是否小于状态 2 得到的最大值 end_m，若小于，将实时值 a_m 更新到实时动作内存 RTA RAM 中的 a_m 位置中，其他维动作值保持初始值不变。否则将状态 3 得到的起始值 beg_m 更新到 a_m，并跳到 $m+1$ 维度的遍历更新。

（6）将以上步骤得到的实时动作内存 RTA RAM 各维动作值输入到网络中运算。

（7）重复以上过程，在上一状态时刻对应维度动作值的基础上累加步长值 $step_m$，直到 RTA RAM 中最后一维动作值即 $a_n = end_n$（n 表示最后一维动作），结束动作遍历过程，从而实现了对所有维度动作值的遍历组合。本文的动作内存设计方法、访问方式高效地实现了强化学习的动作节点快速迭代运算。

3.2.4 奖励内存设计

奖励计算是强化学习的特色之一，其通过基于当前状态的奖励计算对网络 Q 值进行调整，从而在训练模式下对网络参数进行调整。本文设计片上奖励定义与计算机制，如图 7 所示。首先，将状态值与奖励内存中的第 1 组到第 n 组特殊状

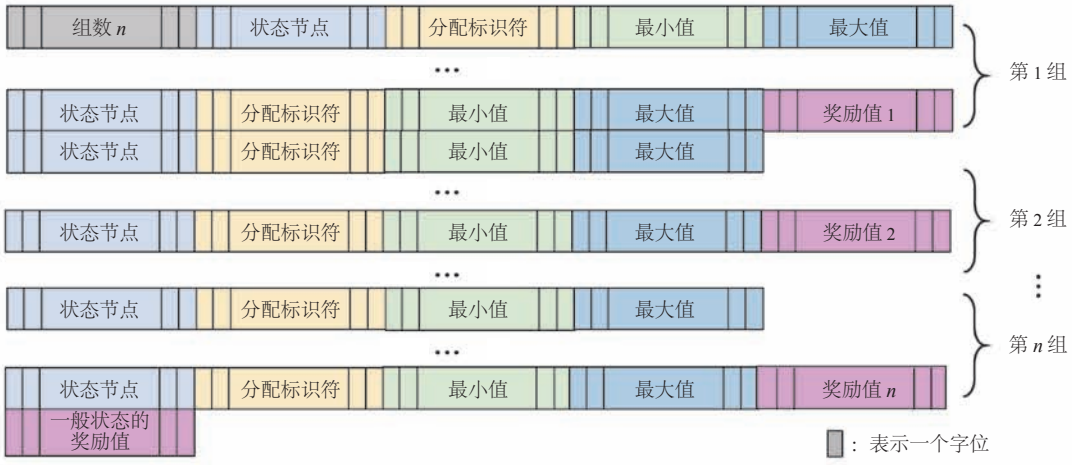


图 7　片上奖励存储设计结构图

Fig. 7　Diagram of on-chip reward storage

态值进行比对。其次，按照具体决策规则，如果当前状态中所要求匹配的状态节点值都满足第 m 组中相应维状态节点的取值范围，则取第 m 组特殊状态的奖励值 m；否则继续遍历下一组 m+1，直到遍历完所有 n 组特殊状态都不匹配时，则取一般状态的奖励值。

3.3 神经网络处理器的应用模式

本文的人工智能推理引擎应用模式如图 8 所示。首先，神经网络芯片对智能车收集到的感知信息采用监督学习网络进行预处理。其次，将处理之后的感知数据，即当前状态 s_t 传入神经网络芯片中，此时是强化学习核心模块，由强化学习处理器对状态进行网络运算，并将对神经元的临时贡献值存储到神经元的临时内存中。再次，通过动作内存模块快速遍历将实时动作值输入到处理器，并循环使用处理器，从而得到当前状态下最大值 Q。最后，根据贪婪策略，得到需要的输出动作值，即包括方向盘转向信息、刹车踏板信息、油门信息等车辆控制信息。将这些动作信息输出到车辆执行端进行，此时车辆进入到下一个新的状态 s_{t+1}，并根据预先定义好的奖励函数得到即时奖励，不断累加奖励值。若是训练模式，则将下一状态 s_{t+1} 输入到强化学习网络运算并进行临时存储，之后快速遍历动作内存得到当前状态下每个动作对应的 Q 值，同时得到最大值 $Q_{\max}(s_{t+1}, a_{t+1}, \theta)$。接下来，更新目标 Q 值为 $r+\gamma Q_{\max}(s_{t+1}, a_{t+1}, \theta)$，随后与当前状态 s_t 和选择的动作 a_t 的 Q 值 $Q(s_t, a_t, \theta)$ 在终端平台，如 MATLAB 软件平台作 loss 处理，并做随机递减变化，同时在 MATLAB 平台进行网络训练，更新网络参数。最终，将上述过程得到的 (s_t, a_t, r, s_{t+1}) 存储到样本池中，供后续对网络进行批训练。

4 实验结果

4.1 仿真分析

CGRA 处理器采用 Verilog 语言进行设计，其代码通过 Modelsim 进行功能仿真。在满足设计功能后采用 Design Compiler(DC)基于 UMC 65 nm CMOS 工艺进行逻辑综合，获得由 DC 综合工具估计的面积、功耗等性能数据并进行时序收敛。如图 9 所示，从寄存器传输级(Register-Transfer Level，RTL)仿真图 9(a)可以看出，首

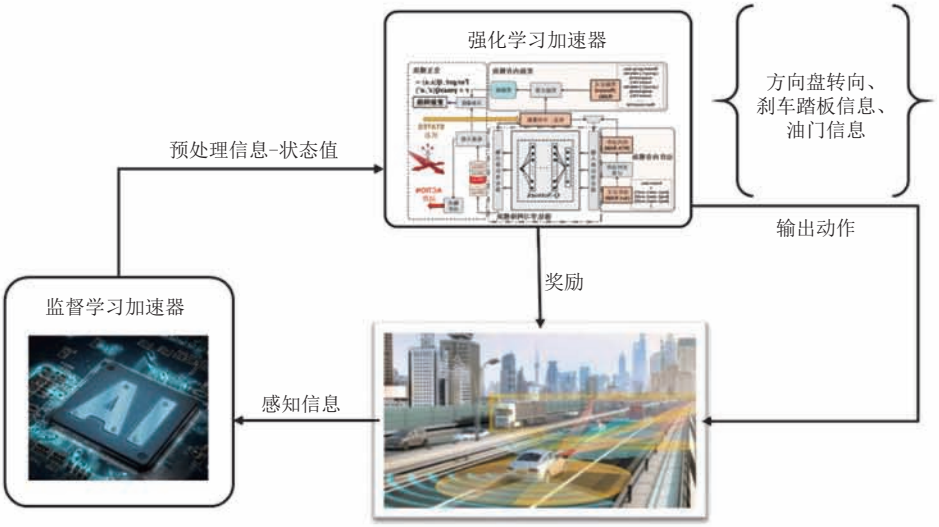


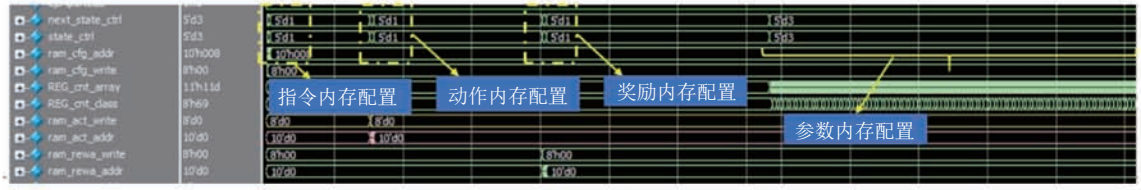
图 8 强化学习处理器在自动驾驶的应用模式

Fig. 8 Application of on-chip reinforcement learning in automatic driving

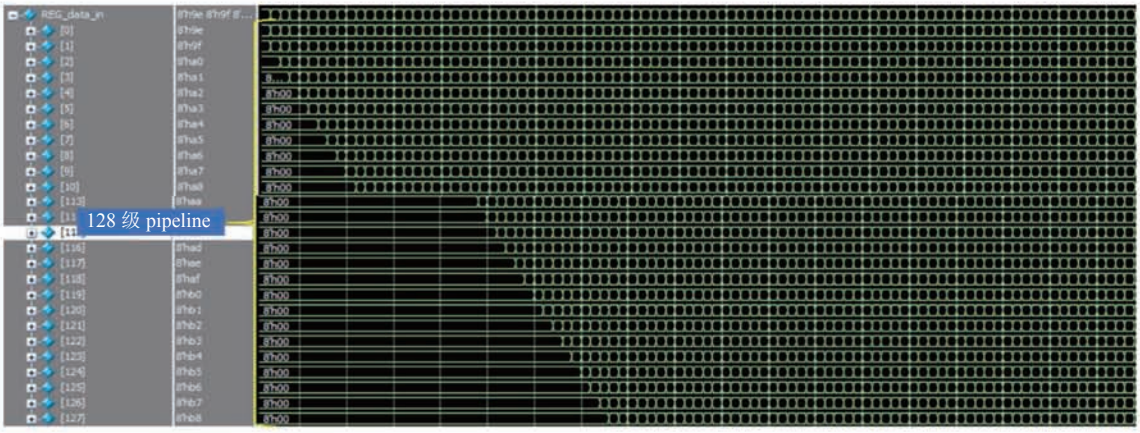
先需要完成各内存配置，为后续网络运算准备好配置和数据，包括指令内存、参数内存、数据内存、动作内存和奖励内存的配置。加速器采用 128 级的流水线增加计算吞吐量，仿真图如图 9(b)所示。采用全连接层实现的深度强化学习网络，Q 迭代过程如图 9(c)所示，通过反复迭代动作内存，最后将计算结果输出到片上缓存中，进行下一步的处理，实现强化学习目的。图 10 所示为部分 RTL 设计结构图。

4.2　物理特性

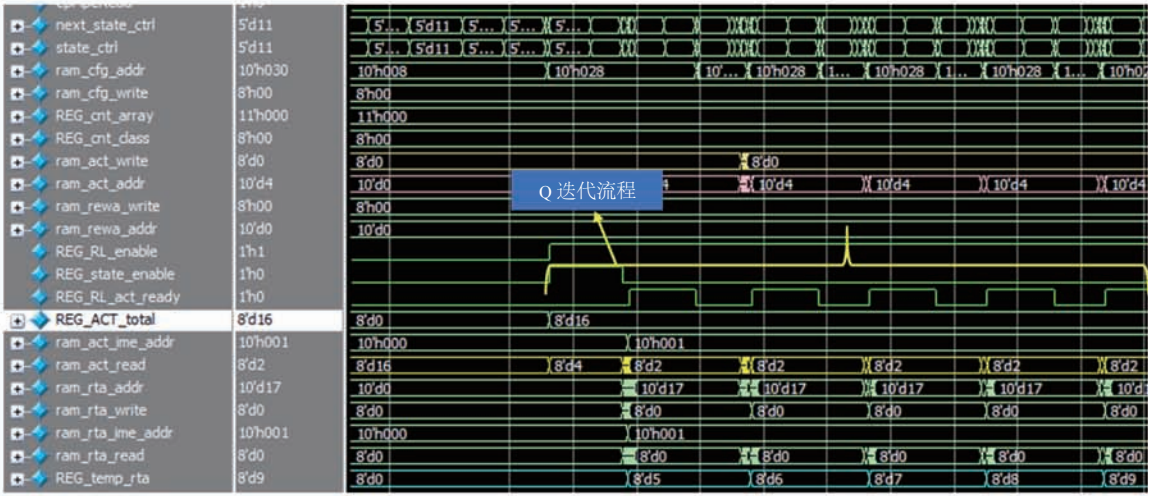
基于 UMC 65 nm CMOS 工艺的逻辑综合结果如表 1 所示，具体包括设计的主要特性以及各部分消耗的功率和占据的面积情况。本设计采用 CGRA 结构以及数据流动的输入输出方式，实现了分层可重构的设计思路，共享了基本计算单元，达到了较低的功耗与面积。



(a) 内存配置仿真图



(b) 128 级数据流水线仿真图



(c) Q 迭代过程仿真图

图 9　基于 Modelsim 的部分仿真结果图

Fig. 9　Simulation diagrams based on Modelsim

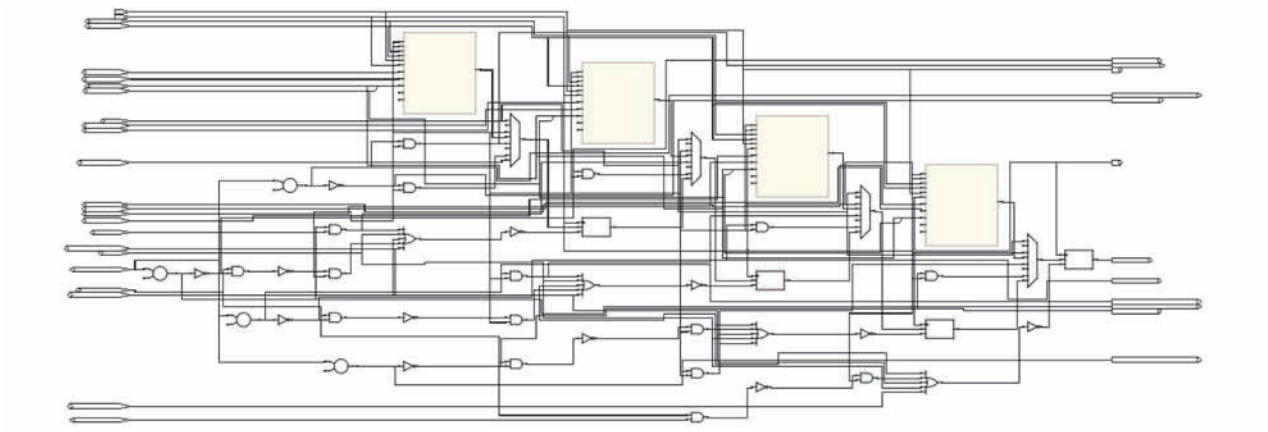


图 10 加速器的部分 RTL 设计结构图

Fig. 10 Register-transfer level design structure of accelerator

表 1 加速器的主要特性

Table 1 Main characteristics of accelerator

频率（MHz）	工艺大小（nm）	电路模块	功耗（mW）	比例（%）	电路模块	面积（μm^2）	比例（%）
200	65	时钟网络	2.985 2	19.18	组合电路	245 570	76.88
		寄存器	12.091 9	77.68	缓冲单元	51 593	16.15
		组合电路	0.488 5	3.14	其他	22 230	6.97
		总消耗	15.460 9	100	总面积	319 385	100

4.3 片上 Q 迭代时间

对于强化学习来说，Q 迭代时间与动作维数的多少、Q 神经网络规模和计算平台有关。在这一部分中，将本设计结构和 CPU(基于 Intel i7 处理器，利用 MATLAB 神经网络工具进行测试)对不同学习网络规模、不同维度的状态空间所需要的 Q 迭代时间进行比较，结果如图 11 所示。具体地，在两种处理器上将 1 维、2 维、4 维和 6 维的动作空间分别应用于 2 层、5 层和 10 层网络结构，从而得到每一种组合所需要的 Q 迭代时间，其中每一维动作均设定 2 种取值。从图 11 可以看到，对于所有测试的动作空间，对应实验中的 3 种不同强化学习网络结构(2 层、5 层、10 层)，本文片上设计所需要的 Q 迭代时间均少于 2 ms，且随着动作空间以及网络结构的增加，时间增加很小。而基于 CPU 处理器所花的时间至少是本文片上设计的 100 倍，并且随着动作空间和网络规模逐渐增大，所花的时间比本设计倍数增加，差距明显，这更加体现出本文设计的优越性。综上所述，本文设计所消耗的时间大幅度降低，且本文设计方法可以满足不同的网络结构和不同类型大小的决策任务，尤其是对实时性要求较高的应用领域，如自动驾驶和机器人控制。该实验结果验证了本文设计的高泛化性，可满足现在快速更新迭代的深度学习算法需求。

5 与国内外相似研究的对比分析

为了应对人工智能(Artificial Intelligence，AI)算法的计算需求，出现了专门应用于加速 AI 算法的 ASIC 芯片，但现有的神经网络加速器主要是针对卷积神经网络、循环卷积神经网络等深度学习算法的加速运算，对于强化学习算法的加速研究还比较少，基本还没有专门用于加速强化学习运算的硬件加速器设计。另外，现有的神经网络加速器更多的只是追求功耗和速度，网络模

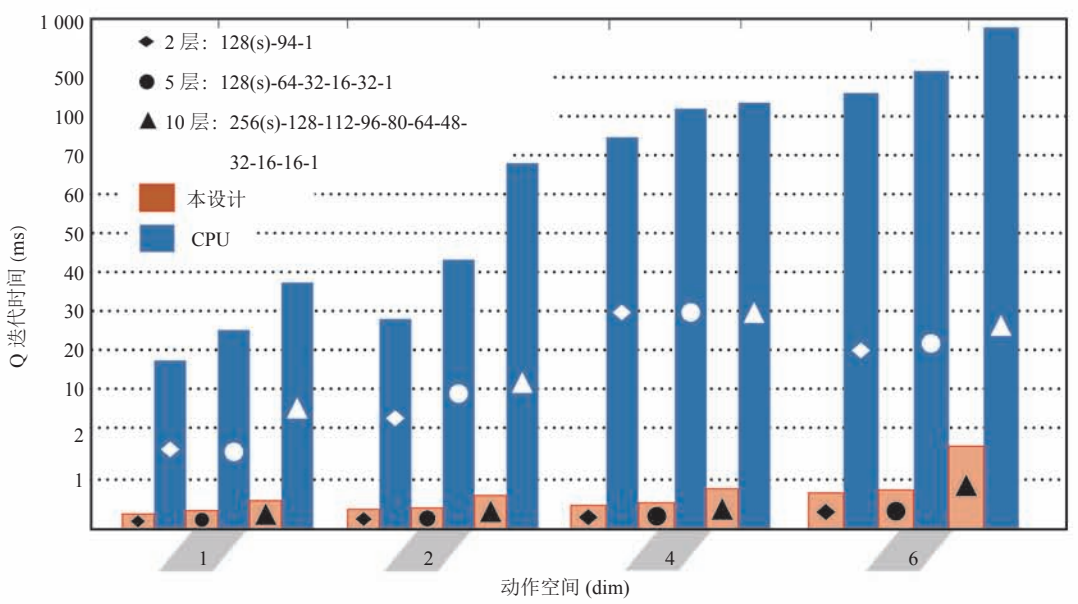


图 11　Q 迭代时间比较图

Fig. 11　Q iteration time comparison graph

式和结构固定，不具有可重构功能，无法满足快速迭代的 AI 算法。在国内，寒武纪研究团队[11,12]提出的第一代神经网络处理器专用于加速卷积神经网络，其功耗为 596 mW、面积为 3.51 mm^2。Chen 等[14]提出用于卷积神经网络运算的加速器，通过减少数据的传输交流，引入较少的扇出电路，大大减少了能量消耗，用于 Alexnet 和 VGG16 运算的功耗分别为 278 mW 和 236 mW。本文研究内容与上述国内外研究相似之处在于均是对人工智能算法的加速研究，不同之处是本文研究支持强化学习和具有可重构可配置功能，面积小(0.32 mm^2)、功耗低(15.46 mW)。这主要是因为本文采用 CGRA 架构，通过引入指令内存、动作内存和奖励内存，设计一种基于可重构阵列架构的强化学习网络计算结构，动态配置不同规模的强化学习网络和动作空间，从而可以实现任意规模的强化学习网络，满足不同复杂场景下的智能体的自我学习和快速决策需求。本文设计核心目标类似于 Sarma 等[21]提出的片上自我意识系统，但区别在于本文设计主要通过在片上实现强化学习决策算法来达到自我意识目标。当然，本设计还存在不足之处，未来工作将重点研究如何利用提出的强化学习计算引擎和训练平台来实现一个完整的自我意识系统，接下来将结合卷积神经网络加速模块，完成感知信息预处理，从而做到智能体在片上实现自我感知和决策功能一体化。

6　结　论

本文提出基于 CGRA 架构的支持监督学习与强化学习的神经网络计算引擎。首先，通过引入 CGRA 架构，实现了对不同神经网络层对计算资源的共享，从而大幅度提升了处理器的可重构性；其次，通过引入动作内存迭代模块和片上快速循环进行强化学习网络的动作选择。结果显示，本文设计处理器可对感知和决策的端到端网络进行芯片上部署，具有功耗低、处理速度快的优点，能够满足不同复杂场景下对存储开销大、功耗低等需求的应用，如自动驾驶、物联网终端等快速发展领域。该处理器对强化学习网络在计算资源受限环境下的部署具有推进意义。

参 考 文 献

[1] Schmidhuber J. Deep learning in neural networks: an overview [J]. Neural Networks, 2015, 61: 85-117.

[2] 刘全, 翟建伟, 章宗长, 等. 深度强化学习综述 [J]. 计算机学报, 2018, 41(1): 1-27.

[3] Arulkumaran K, Deisenroth MP, Brundage M, et al. A brief survey of deep reinforcement learning [J]. 2017, arXiv:1708.05866.

[4] Jia YQ, Shelhamer E, Donahue J, et al. Caffe: convolutional architecture for fast feature embedding [C] // Proceedings of the 22nd ACM International Conference on Multimedia, 2014: 675-678.

[5] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition [J]. 2014, arXiv:1409.1556.

[6] Ren S, He K, Girshick R, et al. Faster R-CNN: towards real-time object detection with region proposal networks [C] // Advances in Neural Information Processing Systems, 2015: 91-99.

[7] Chakradhar S, Sankaradas M, Jakkula V, et al. A dynamically configurable coprocessor for convolutional neural networks [J]. ACM SIGARCH Computer Architecture News, 2010, 38(3): 247-257.

[8] Temam O. A defect-tolerant accelerator for emerging high-performance applications [J]. ACM SIGARCH Computer Architecture News, 2012, 40(3): 356-367.

[9] Qadeer W, Hameed R, Shacham O, et al. Convolution engine:balancing efficiency & flexibility in specialized computing [J]. ACM SIGARCH Computer Architecture News, 2013, 41(3): 24-35.

[10] Farabet C, Martini B, Corda B, et al. NeuFlow: a runtime reconfigurable dataflow processor for vision [C] // Computer Vision and Pattern Recognition Workshops, 2012: 109-116.

[11] Chen YJ, Luo T, Liu SL, et al. DaDianNao: amachine-learning supercomputer [C] // IEEE/ACM International Symposium on Microarchitecture, 2014: 609-622.

[12] Liu DF, Chen TS, Liu SL, et al. PuDianNao: apolyvalent machine learning accelerator [J]. ACM Sigarch Computer Architecture News, 2015, 43(1): 369-381.

[13] Chen TS, Guo Q, Temam O, et al. Statistical performance comparisons of computers [J]. IEEE Transactions on Computers, 2015, 64(5): 1442-1455.

[14] Chen YH, Krishna T, Emer JS, et al. Eyeriss: an energy-efficient reconfigurable accelerator for deep convolutional neural networks [J]. IEEE Journal of Solid-State Circuits, 2017, 52(1): 127-138.

[15] Mnih V, Kavukcuoglu K, Silver D, et al. Playing atari with deep reinforcement learning [J]. 2013, arXiv:1312.5602.

[16] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning [J]. Nature, 2015, 518(7540): 529-533.

[17] Silver D, Huang A, Maddison CJ, et al. Mastering the game of Go with deep neural networks and tree search [J]. Nature, 2016, 529(7587): 484-489.

[18] Burns LD. Sustainable mobility: a vision of our transport future [J]. Nature, 2013, 497(7448): 181-182.

[19] 夏伟, 李慧云. 基于深度强化学习的自动驾驶策略学习方法 [J]. 集成技术, 2017, 6(3): 29-40.

[20] De Sutter B, Raghavan P, Lambrechts A. Coarse-grained reconfigurable array architectures [M] // Handbook of Signal Processing Systems, 2010: 449-484.

[21] Sarma S, Dutt N, Gupta P, et al. Cyberphysical-system-on-chip (CPSoC): a self-aware MPSoC paradigm with cross-layer virtual sensing and actuation [C] // Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015: 625-628.