文章编号:1671-4598(2022)06-0242-06 DOI:10.16526/j.cnki.11-4762/tp.2022.06.037 中图分类号:TP3 文献标识码:A

基于 FPGA 的深度强化学习硬件加速技术研究

凤 雷,王宾涛,刘 冰,李喜鹏
(哈尔滨工业大学 电子与信息工程学院,哈尔滨 150001)

摘要:深度强化学习(DRL)是机器学习领域的一个重要分支,用于解决各种序贯决策问题,在自动驾驶、工业物联网等领域具有广泛的应用前景;由于 DRL 具备计算密集型的特点,导致其难以在计算资源受限且功耗要求苛刻的嵌入式平台上进行部署;针对 DRL 在嵌入式平台上部署的局限性,采用软硬件协同设计的方法,设计了一种面向 DRL 的 FPGA 加速器,提出了一种设计空间探索方法,在 ZYNQ7100 异构计算平台上完成了对 Cartpole 应用的在线决策任务;实验结果表明,研究在进行典型 DRL 算法训练时的计算速度和运行功耗相对于 CPU 和 GPU 平台具有明显的优势,相比于 CPU 实现了 12.03 的加速比,相比于 GPU 实现了 28.08 的加速比,运行功耗仅有 7.748 W,满足了深度强化学习在嵌入式领域的在线决策任务。

关键词:深度强化学习;FPGA;异构计算;在线决策;嵌入式领域

Research on Hardware Acceleration Technology of Deep Reinforcement Learning Based on FPGA

FENG Lei, WANG Bintao, LIU Bing, LI Xipeng
(School of Electronic and Information Engineering, Harbin Institute of Technology, Harbin 150001, China)

Abstract: Deep reinforcement learning (DRL) is an important branch in the field of machine learning. It is used to solve various sequential decision-making problems. It has a wide application prospects in the fields of automatic driving, industrial Internet of things and so on. Because DRL is computationally intensive, it is difficult to deploy on embedded platforms with limited computing resources and demanding power con-sumption. In view of the limitations of DRL deployment on embedded platform, a DRL oriented FPGA accelerator is designed by using the method of the software and hardware collaborative design, and a design space exploration method is proposed. The online decision-making task of cartpole application is completed on the zynq7100 heterogeneous computing platform. The experimental results show that the computing speed and running power consumption of the research in the training of typical DRL algorithm have obvious advantages over the CPU and GPU platform. Compared with the CPU, the CPU achieves an acceleration ratio of 12.03 and the GPU achieves an acceleration ratio of 28.08, and the running power consumption is only 7.748 W, which meets the online decision-making task of deep reinforcement learning in the embedded field.

Keywords: DRL (deep reinforcement learning); FPGA; heterogeneous computing; online decision-making; embedded field

0 引言

强化学习(RL, reinforcement learning)是指从环境状态到动作映射的学习,以使动作从环境中获得累计奖赏值最大[1],经常被用于序贯决策层问题。与监督学习[2]不同,RL 算法主要强调智能体与环境的交互,在二者的交互的过程中,环境会根据智能体所处的状态以及所决策的动作给予其一定的奖惩信号,智能体则会根据所获得的奖惩信号对自身的决策策略进行优化,从而最大化决策过程中所获得的累计奖励。

2013 年,DeepMind 团队将深度学习中的卷积神经网络(CNN, convolutional neural network)[3]算法与传统强化学习 Q 学习[4]相结合,设计了 DQN 算法[5],在雅达利游戏平台中取得了比人类玩家更高的游戏分数,从此掀起了一股深度强化学习[6]的研究浪潮。后续 DeepMind 团队基于 DRL 算法所研发的 AlphaGo[7]和 AlphaGo Zero[8]在机器博弈领域取得了巨大成功,更是成为了人工智能领域的里程碑事件。目前,DRL 除了在游戏中进行应用外,直接在边缘设备上实现 DRL 同样有巨大的应用价值和广泛的应用前景,例如为充当巡逻机器人的无人机(UAV, unmanned aerial vehicle)[9]提供自主避障和航路规划的能力,为无人车辆[10-11]提供自主驾驶分析决策能力。

DQN 算法作为 DRL 领域的开山之作,被后续许多的 DRL 算法所借鉴。其解决了传统 Q 学习的“维度灾难”问题,采用多层神经网络来完成值函数的非线性逼近功能,替代传统的 Q 表查询决策方式,将神经网络的感知能力和强化学习的决策能力结合,实现端到端的感知与决策。在智能体与环境交互的过程中,同时存在神经网络的推理与

收稿日期:2021-12-20; **修回日期**:2022-01-04。
作者简介:凤 雷(1978-),男,黑龙江兰西县人,博士,副教授,主要从事自动测试系统方向的研究。
通讯作者:刘 冰(1982-),男,黑龙江哈尔滨市人,博士,副教授,主要从事自动测试系统、图像处理处理和 FPGA 加速方向的研究。
引用格式:凤 雷,王宾涛,刘 冰,等.基于 FPGA 的深度强化学习硬件加速技术研究[J].计算机测量与控制,2022,30(6):242-247.

训练两类运算，这两类运算都具备计算密集型的特点，需要较强的算力才能保证算法的实时性。

DQN 计算密集型的特点，对于计算资源和功耗都受限的边缘设备而言，直接实现深度强化学习算法存在一定的挑战。这种挑战主要来自于两方面：一方面是 DQN 算法本身计算密集型特点和计算数据之间较强的依赖关系；另一方面是大多数嵌入式计算平台本身单指令单数据流计算架构的局限性，无法支持面向 DQN 的高性能计算。这导致有关在嵌入式设备部署 DQN 算法的研究进展十分缓慢，相关研究现状在 1.2 节中得到阐述。面向边缘在线决策应用，本文提出一种基于 FPGA 平台的 DQN 算法实现方法，可以在 FPGA 平台上完成 DQN 算法的推理和训练。主要工作如下：

1）提出了一种基于 FPGA 平台的 DQN 算法的硬件实现架构，架构中的加速器 IP 核采用流式架构设计，可以灵活配置算法的训练超参数。

2）在 FPGA 平台计算资源和存储资源的约束下，提出了一种设计空间的探索方法。通过定量分析 DQN 算法实现所需的存储资源和计算资源，获得 DQN 算法在 FPGA 中进行加速部署时每一层的并行计算参数。

3）面向典型应用 Cartpole 搭建了应用验证平台，在 FPGA 平台上进行了设计的功能验证和性能测试，并在网络的训练时间和功耗方面同 CPU 平台和 GPU 平台进行了实验对比。

1　背景

1.1　DQN 算法

RL 的基本模型可以用图 1 表示，通过智能体与环境的信息交互，实现决策功能。整个过程可用四元组 $<A,S,R,P>$ 描述，式中 A 为智能体动作集合；S 为智能体感知的环境信息；R 为智能体得到的奖励或惩罚；P 为智能体交互的环境。

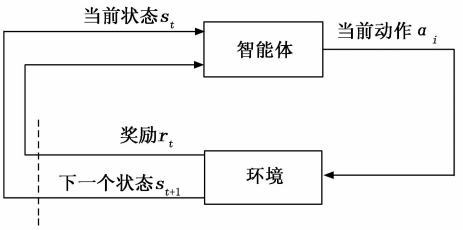

图 1　强化学习基本模型

Q 学习算法[4]作为一种经典的 RL 算法，使用 Q 查询表存储各个动作对应的 Q 值，通过查询每个动作的 Q 值，指导智能体做出相应决策。受限于计算机存储的限制，Q 表在处理高维状态数据方面表现不佳[12]。

为了解决高维输入问题，DeepMind 团队于 2013 年融合神经网络和强化学习，提出了 DQN 算法，并在 RL 领域取得出色的效果。DQN 算法拉开了深度强化学习的序幕，得到工业界和学术届的广泛青睐。该算法较为经典的部分包括：Current _ Q 网络、Target _ Q 网络、经验回放机制等。DQN 的损失函数如公式（1）所示，以最小化损失函数 $L(\theta)$ 为目标，使用反向传播算法[13]对 Current _ Q 的网络参数进行更新。式中 Target _ Q 为目标 Q 值；r_t 为 t 时刻的奖励值；$max\hat{Q}(s_{t+1},a_{t+1};\theta')$ 为 Target _ Q 网络的输出的最大值；θ' 为 Target _ Q 网络的权值参数。

$$Target\_Q = r_t + \gamma max\hat{Q}(s_{t+1},a_{t+1};\theta')$$
$$L(\theta) = E[\frac{1}{2}(Target\_Q - Q(s_t,a_t;\theta))^2] \quad (1)$$

1.2　FPGA 加速强化学习算法

近年来，机器学习在嵌入式边缘设备上的加速实现[14]，已经成为人工智能领域[15]的热门话题，FPGA 由于其低功耗、高能效和可重构的特点，在硬件加速领域[16]备受青睐，目前最先进的机器学习加速器大多支持监督学习，如 CNN[17]、循环神经网路（RNN，recurrent neural network）[18]等，但对深度强化学习的硬件加速目前还处于刚刚兴起的状态。

Shengjia S 等人在 Stratix 系列 FPGA 上对 TPRO 算法（TRPO，trust region policy optimization）进行了硬件加速以应用于机器人控制应用[19]，后续提出了一种设计空间探索方法对 TRPO 算法加速进行进一步的优化[20]。S. Jiang 等人采用单引擎架构，在 Altera Arria 10 上实现了 Deep Q-Learing 算法的硬件加速[21]。但是上述研究，在计算架构和设计空间探索等方面仍有很大提升空间。

2　基于 DQN 的 FPGA 硬件架构和加速器设计

2.1　总体硬件架构

如图 2 所示，总体硬件架构主要包括外部存储器（DDR，direct digital radiography）、处理单元（PS，processing system）、可编程逻辑部分（PL，processing logic）的加速器和片内外总线互联。我们通过 PS 和 PL 协同工作来高效的完成 DQN 算法的计算，其中 PS 部分主要负责与环境进行交互，奖励函数的计算，DDR 中训练经验池的维护，以及对 PL 进行超参数和工作模式的配置；PL 部分定制化设计 DQN 算法加速器，用于实现算法中神经网络的前向推理、误差反向传播和权值更新等计算密集部分，PL 部分加速器结构设计、加速算子设计及相关设计空间探索方法是研究的核心。

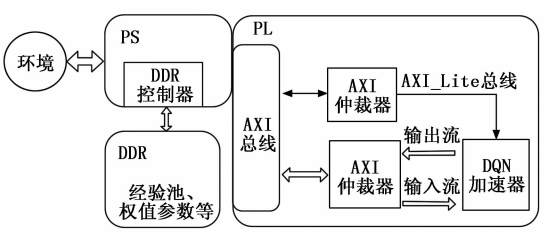

图 2　总体硬件架构

2.2　加速器结构设计

整个加速器采用流式架构，针对 DQN 算法的各个模块进行硬件定制化设计。DQN 算法加速器 IP 核的结构如图 3

所示，定制化设计了 Target _ Q 模块、Current _ Q 模块、损失函数计算模块、权值更新和替换模块、控制模块和参数存储单元。其中 Target _ Q 模块用于完成 Target _ Q 网络的前向推理；Current _ Q 模块用于完成 Current _ Q 网络的前向推理和反向传播；控制模块用于对加速器的工作模式和训练参数进配置，使得各个模块能够协同工作；权值更新与替换模块用于实现 Current _ Q 网络的权值更新和 Target _ Q 网络的权值替换；损失计算模块主要用于实现损失函数的计算。

加速器存在两种工作模式：一种是直接利用学习好的权值参数，与环境交互后，智能体只做出决策但不进行学习；另一种是做出决策后智能体会根据环境给予的奖励反馈进行学习，调整自身权重参数。

Target _ Q 模块和 Current _ Q 模块是整个硬件加速过程中的设计重点，在本设计中，Target _ Q 模块和 Current _ Q 模块均由加速算子（VMPU，vector matrix processing unit）通过流式先入先出寄存器（FIFO，first in first out）级联组成，加速算子的具体细节将会在 2.3 节中介绍，同时将在 2.4 节介绍本设计的设计空间探索方法。

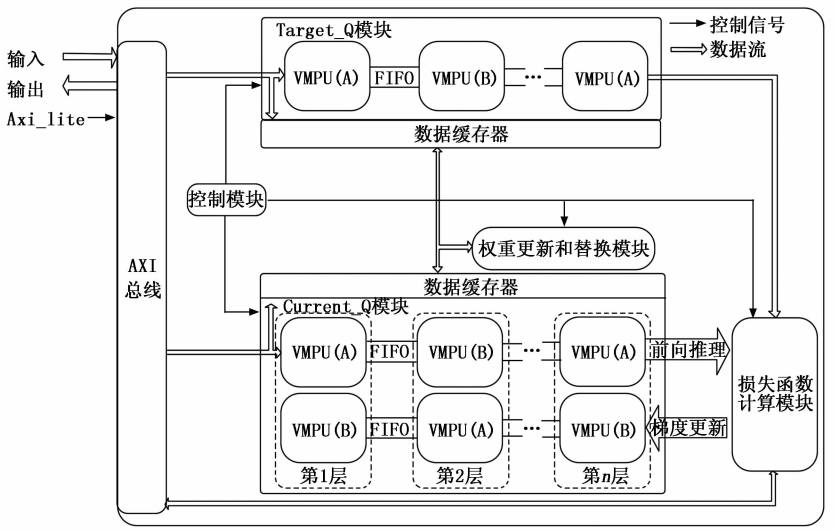

图 3 DQN 加速器结构图

2.3 加速算子 VMPU 设计

DQN 算法中的神经网络选择多层感知机，针对多层感知机中向量矩阵乘法计算密集型的特点，以及前向推理和反向传播计算过程间的数据依赖，基于 FPGA 硬件资源的特点，设计了单指令多数据（SIMD，single instruction multiple data）运算和多处理单元（PE，processing element）两种加速模式，其设计示意图如图 4 所示。SIMD 主要用于完成 CNN 中的层累加单元，即通过在 FPGA 内部定制化设计 SIMD，通过硬件实现基础的层累加模块，完成基于指令集的硬件加速；PE 主要用于实现多个 SIMD 的并行计算；在单个 PE 内实 SIMD 并行的同时，VMPU 单元内还实现了多个 PE 的并行计算，以达到最大化硬件加速性能的目的。

VMPU 的输入是上一个计算单元的输出或者最原始的

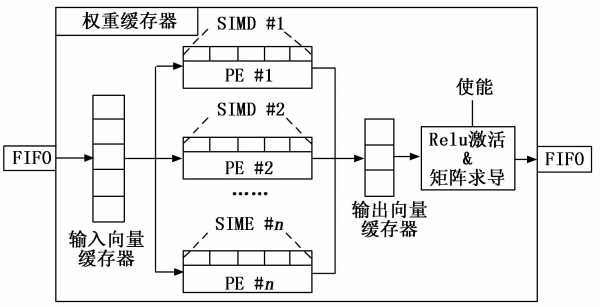

图 4 VMPU 设计示意图

输入，经过 FIFO 寄存器被存储在 VMPU 内置输入向量寄存器，用于神经网络卷积计算的权重已经部署到 FPGA 内部，此使输入向量寄存器的元素与已经部署好的权重展开并行计算，并将结果放到输出向量寄存器内，进而经过激活、矩阵求导等操作后，作为输出被传送到下一计算单元。

在 FPGA 实现 VMPU 的过程中，前向推理与反向训练的计算过程之间存在相关度较高的依赖关系，为了解决 DQN 中层间推理与训练的数据依赖关系，同时实现 FPGA 并行计算，需要在 FPGA 片上对权重矩阵进行分区存储，基于计算过程中权值矩阵维度设计了两类 VMPU，分别是以列为主的 VMPU（A）和以行为主的 VMPU（B），两类 VMPU 的存储方式划分示意如图 5 所示。在 VMPU（A）中，SIMD 并行对应矩阵的行维度，PE 并行对应矩阵的列维度；由于两类 VMPU 的权重矩阵存储方式互为转置，在 VMPU（B）中，并行计算对应的维度与 A 模式相反。在部署 DQN 网络时，对于奇数层而言，前向推理时使用 VMPU（A），反向传播时使用 VMPU（B）；对于偶数层而言，前向推理时采用 VMPU（B），反向传播时采用 VMPU（A）。

如图 6 所示，以 VMPU（A）为例，具体说明矩阵加速算子的计算过程。假设网络第一层的神经元个数为 8，第二层的神经元个数为 4，在进行前向推理时，相当于执行一个 1 * 8 的向量与 8 * 4 的矩阵的乘法。在 VMPU（A）中，如图 6（a）所示，假定 SIMD 取 4，PE 取 2，则行维度的折叠因子 SF 为 2，则列维度的折叠因子 PF 为 2，折叠因子的计算方式将在 2.4 节阐述。由 SF 和 PF 可算得总折叠因子为 4，即循环 4 次便可完成整个向量矩阵乘法的计算。在第一次循环中，如图 6（b）所示，首先计算输入向量前 4 个元素 $a_1 - a_4$ 与权值矩阵对应元素的乘积累加，生成 c_1 和 c_2 的中间结果，第二个循环继续计算权值矩阵当前列未进行计算的元素，生成的乘积累加结果与之前的中间结果相加，生成 c_1 和 c_2 的最终结果，然后加上偏置，进行激活函数的计算。同时，激活函数的梯度也将在此时计算，并存储在片上以供反向传播使用。最后，将两个最终计算结果 c_1 和 c_2 流入 FIFO 中。

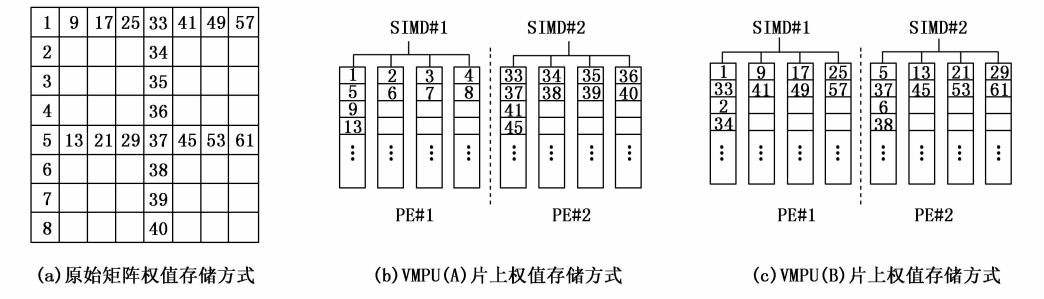

图 5　片上权值存储方式

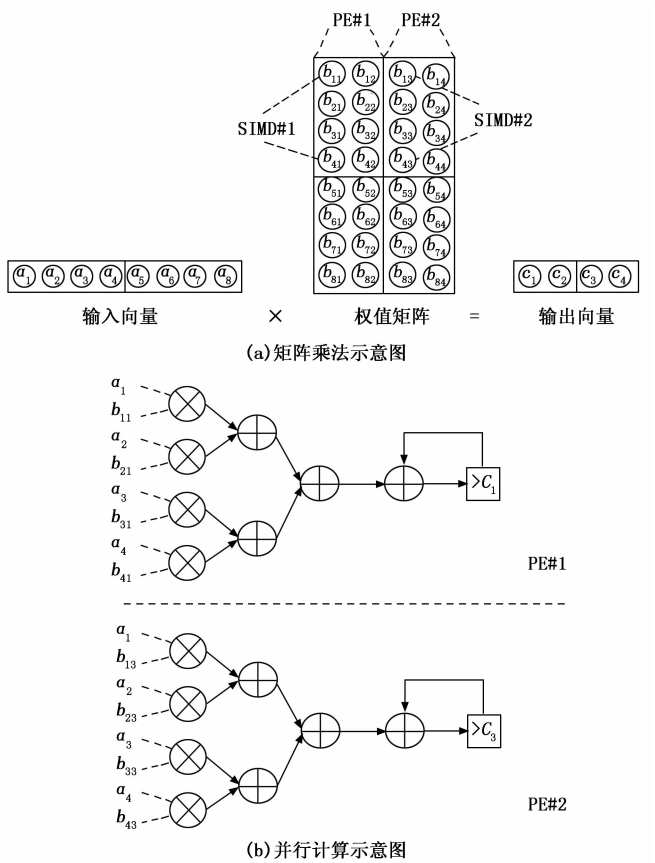

图 6　VMPU（A）计算示意图

VMPU（A）将两个最终结果流入 FIFO 后，后续连接的 VMPU（B）将在此刻收到启动信号，从 FIFO 中取出上一层的输出数据，存入输入寄存器中，开始第二层的计算。与 VMPU（A）类似，VMPU（B）的内部同样为乘累加树结构，但其计算逻辑是以行为主，每完成一次循环，生成的都是中间结果，直到最后一次循环结束，VMPU（B）将会累加出最终的计算结果。通过选取合适的并行参数，便能让级联的 VMPU（A）和 VMPU（B）之间形成流水，只要 VMPU（A）的计算不中断，每隔一段时间，VMPU（B）便会接收来自 VMPU（A）的数据并进行下一层的计算，两层之间形成数据流水后，两层整体的计算时间大概等于 VMPU（A）的计算时间。

2.4　设计空间探索

为了满足 DQN 算法特性和 FPGA 硬件平台资源约束，确定网络部署的并行计算参数，对设计空间探索方法进行研究是十分必要的。不同于研究[21]中仅考虑了 DSP 资源，本设计在建立资源模型的同时考虑了 DSP 资源和 BRAM 资源，达到对计算资源和存储资源设计空间探索的目的。

Target Q 模块和 Current Q 模块的计算过程最为耗时，对于以上两个模块，其内部的 VMPU 单元采用 A-B-A-B 的方式级联而成，通过对总循环次数 $FoldCycle$ 进行建模，完成设计空间探索问题中性能模型的寻优，此处 $FoldCycle$ 可用于表征 VMPU 模块的计算时间。如公式（2）所示，设计空间探索问题，在软硬件资源约束的情况下，寻求最小的总循环展开次数。本文首先将先对 VMPU 单元的 A 模式和 B 模式分别进行建模，然后再对整个系统进行建模。

$$\begin{aligned} &\min_{P} && FoldCycle \\ &subject\quad to && DSP_{\text{total}} \leqslant DSP_{\text{FPGA\_Limit}} \\ & && BRAM_{\text{total}} \leqslant BRAM_{\text{FPGA\_Limit}} \\ & && P_i \leqslant LayerSize_i \end{aligned} \tag{2}$$

式（2）中，$FoldCycle$ 代表总循环展开次数；P 代表循环展开参数的集合，该集合可用 VMPU 中的循环展开参数 PE 和 SIMD 表示；$LayerSize_i$ 代表神经网络第 i 层的大小；DSP_{total}、$BRAM_{\text{total}}$代表设计实际消耗的总的计算资源和存储资源个数；$DSP_{\text{FPGA\_Limit}}$、$BRAM_{\text{FPGA_Limit}}$代表 FPGA 平台的计算资源和存储资源的限制。

首先对 VMPU（A）进行分析，在向量矩阵乘法中，使用 $RowSize$ 表示权重矩阵的行维度大小，$SIMD_{\text{A}}$ 为 VMPU（A）的行循环展开参数，sf_{A} 为其对应的行并行折叠因子；使用 $ColSize$ 表示权重矩阵的列维度大小，PE_{A} 为 VMPU（A）的列循环展开参数，pf_{A} 为其对应的列并行折叠因子；则 VMPU（A）的总循环展开因子 $FoldCycle_{\text{A}}$ 为 sf_{A} 和 pf_{A} 的乘积，相关计算关系可用公式（3）～（5）表示：

$$sf_{\text{A}} = RowSize / SIMD_{\text{A}} \tag{3}$$

$$pf_{\text{A}} = ColSize / PE_{\text{A}} \tag{4}$$

$$FoldCycle_{\text{A}} = sf_{\text{A}} * pf_{\text{A}} \tag{5}$$

然后对 VMPU（B）进行分析，在向量矩阵乘法中，同样使用 $RowSize$ 表示权重矩阵的行维度大小，PE_{B} 为 VMPU

(B) 的行循环展开参数，pf_B 为其对应的行并行折叠因子；使用 *ColSize* 表示权重矩阵的列维度大小，$SIMD_B$ 为 VMPU (B) 的列循环展开参数，sf_B 为其对应的列并行折叠因子；则 VMPU (B) 的总循环展开因子 $FoldCycle_B$ 为 pf_B 和 sf_B 的乘积，相关计算关系可用公式 (6) ～ (8) 表示：

$$pf_B = RowSize/PE_B \tag{6}$$

$$sf_B = ColSize/SIMD_B \tag{7}$$

$$FoldCycle_A = pf_A * sf_A \tag{8}$$

采用 A-B 级联模式，可以实现层间的流水计算与并行计算，进而增加硬件加速的性能。为了实现 A-B 级联模式，级联的两层应当满足公式 (9) 的要求，这样可以解决 VMPU (B) 对 VMPU (A) 的数据依赖。当 VMPU (A) 与 VMPU (B) 之间形成层间的计算流水后，A 模式的计算时间便能掩盖 B 模式的计算时间，如公式 (10) 所示，式 (11) 中 $FoldCycle_{AB}$表示级联的两层计算所需的循环数。

$$sf_A \geqslant sf_B \tag{9}$$

$$FoldCycle_{AB} = FoldCycle_A \tag{10}$$

在本设计中，我们实现了 3 个层次的流水以及并行设计，第一个是 VMPU 内部的乘积累加流水以及多个乘累加树的并行；第二个是 VMPU 通过 A-B 级联时，形成层与层之间的计算流水；第三个是在训练时，Target _ Q 模块与 Current _ Q 模块之间形成并行计算，Target _ Q 模块的计算时间将被 Current _ Q 模块的计算时间掩盖。因此，整个系统的时间消耗主要由 Current _ Q 模块决定。系统处理决策样本所用的循环数如公式 (11) 所示，式中 $FoldCycle_{FP}$ 表示前向推理在 Current _ Q 模块所消耗的时间，$FoldCycle_{BP}$表示反向传播在 Current _ Q 模块所消耗的时间。

$$FoldCycle = FoldCycle_{FP} + FoldCycle_{BP} \tag{11}$$

除运行时间参数进行建模外，设计还对 DSP 计算资源和 BRAM 存储资源进行建模。VMPU 中 DSP 资源的消耗量与流水计算的启动间隔 (II，initiation interval) 有关，当 II 为 1 时，消耗的 DSP 资源最多，当 II 大于 1 时，VMPU 将对 DSP 资源进行复用，所消耗的 DSP 资源如式 (12) 和 (13) 所示，式中常系数 5 表示浮点型层累加单元所消耗的最小 DSP 个数。设计采用了 FPGA 空间换取时间的并行加速思想，存储单元采用 BRAM 实现，其使用数量和加速结构有关。算子并行度为 $SIMD * PE$，则需要 $2 * SIMD * PE$ 个 BRAM 单元来存储输入向量和权重，同时需要 PE 个 BRAM 来存储偏置变量。故对于 VMPU 中 BRAM 资源的消耗可用公式 (14) 和 (15) 表示：

$$DSP_A = ceil((SIMD_A/II_A) * PE_A) * 5 \tag{12}$$

$$DSP_B = ceil((SIMD_B/II_B) * PE_B) * 5 \tag{13}$$

$$BRAM_A = (SIMD_A * PE_A + PE_A) * 2 \tag{14}$$

$$BRAM_B = (SIMD_B * PE_B + PE_B) * 2 \tag{15}$$

本设计中消耗 BRAM 资源和 DSP 资源的主要模块为 Target _ Q 模块、Current _ Q 模块和权值更新与替换模块，因此，可以按照公式 (16) 估计系统相关资源消耗的总体情况。

$$BRAM = Target\_Q_{BRAM} + Current\_Q_{BRAM} + Update_{BRAM}$$

$$DSP = Target\_Q_{DSP} + Current\_Q_{DSP} + Update_{DSP} \tag{16}$$

3 实验评估

3.1 应用介绍

使用 OPEN AI Gym 提供的 Cartpole 环境对设计进行测试，Cartpole 游戏环境如图 7 所示。CartPole 是一个非常经典的车杆游戏，游戏里面有一个小车，车上有竖着一根可以旋转的杆子，每回合中车杆的初始状态都会有所不同，小车需要左右移动来保持杆子竖直。为了保证游戏继续进行，需要满足以下两个条件：

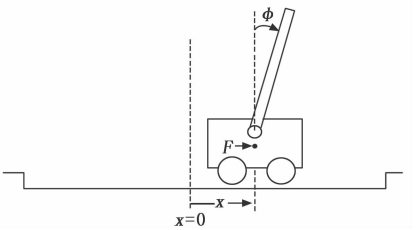

图 7 Cartpole 应用

1) 杆子倾斜的角度 φ 不能大于 15°.

2) 小车移动的位置 x 需保持在一定范围内 (±2.4 个单位长度)。

智能体能观察到的环境输入共有 4 个，为 $< x, \varphi, \dot{x}, \dot{\varphi} >$。$x$ 代表小车在轨道上的位置；φ 代表杆子与竖直方向的夹角；$\dot{x}$ 代表小车速度；$\dot{\varphi}$ 代表角度变化率。智能体可以根据观测到的信息，执行左移或者右移的动作。

3.2 实验过程与参数设置

本文所搭建的应用验证平台如图 8 所示，用于对设计进行功能验证和性能分析。应用验证平台分为两个部分，分别是 PC 机和 ZYNQ7100 平台。我们在 ZYNQ 的 ARM 端搭建了 Linux 操作系统并编写了 Cartpole 应用程序，在 PC 机上运行 gym 中的 Cartpole 环境，边缘侧的 ZYNQ 与 PC 机中的环境通过网口进行通信，以模拟智能体与环境的交互过程。训练开始后，ZYNQ 将会向 PC 机中的环境传递动作信息，环境收到信息会后向 ZYNQ 反馈执行动作后的奖励、下一刻的状态信息和当前回合是否结束的标志。

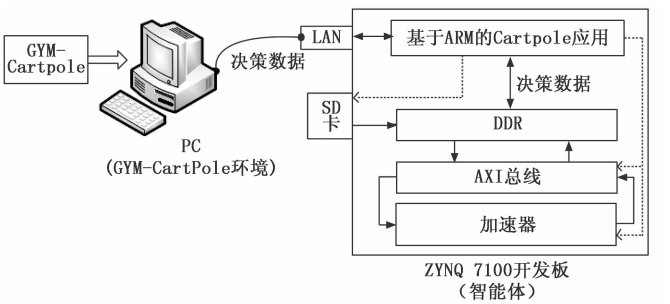

图 8 Cartpole 应用验证平台

我们使用含有一个隐藏层的多层感知机作为 DQN 算法的基础网络，来进行 Cartpole 游戏的控制，网络的结构为 4-320-2。针对三层的感知机网络，可以使用 4 个 VMPU 进

行 A-B 级联完成 Current _ Q 模块的搭建，使用 2 个 VMPU 完成 Target _ Q 模块的搭建。根据本文所提的设计空间探索方法，对 Cartpole 应用中 DQN 网络在 ZYNQ 7100 上的部署实现进行寻优，求得各个单元的循环展开参数大小：$SIMD_A=4$，$PE_A=PE_B=32$，$SIMD_B=2$。

3.3　实验结果与分析

FPGA 中加速器部分的设计使用赛灵思公司的 Vivado HLS（v2018.3）开发工具，表 1 为 Cartpole 应用加速器的资源使用情况，从表中可以看出，通过设计空间探索预估的 BRAM 和 DSP 资源与实际消耗的相差无几，稍微的误差存在于预估过程未加入损失计算单元消耗的相关资源。通过对资源利用率的观察，证明了所提出的设计空间探索方法的高效性。

表 1　资源使用报告

项	Bram_36k	DSP	FF	LUT	Power
估计值	565	1 610	—	—	—
利用值	563	1 645	198 493	159 626	7.748 W
总量	755	2 020	554 800	277 400	—
利用率/%	75.57	81.44	35.78	57.54	—

为了探究本文 FPGA 加速器的对 DQN 算法的整体加速性能，进行了与 CPU、GPU 的对比实验，相关硬件平台的型号、频率如表 2 所示。时间测量采用时间戳计数的方式；GPU 和 CPU 的功耗采用各自额定功率表示，加速器的的运行功耗使用 Vivado 软件综合预估的额定功率表示；相比于 FPGA 的运行时间，计算 3 种平台的加速比，相关实验结果如表 2 所示。

表 2　3 种硬件平台训练 32 个样本处理时间对比

平台	型号	频率/Hz	运行时间/ms	功耗/W	加速比
GPU	RTX TITAN	1.35 G	6.655	280	28.08
CPU	E5-2630v4	2.20 G	2.853	85	12.03
ARM-FPGA	ZYNQ7100	100 M	0.237	7.748	1.00

从表 2 可以看出，与 GPU、CPU 对比，本文提出的 FPGA 加速器运行速度提高了 28.08 和 12.03 倍；基于 FPGA 设计的加速器在运行功耗方面也具有明显的优势。

4　结束语

本文提出了一种用于 DQN 算法加速的硬件架构及其设计空间探索方法，其采用流式结构实现加速器 IP 核设计。通过设计空间探索，我们可以针对不同和 DQN 网络和芯片平台进行硬件加速实现。以 Cartpole 应用为例，我们设计了一个三层的 DQN 网络，通过设计空间探索寻得了全局最优的并行计算参数，然后在 ZYNQ 7100 平台上完成了设计的部署测试，测试结果表明，FPGA 在训练时的计算时间和功耗方面相对于 GPU/CPU 具有明显的优势。

参考文献：

[1] 顾一凡. 基于强化学习的组合优化综述 [J]. 软件导刊. 2021, 20 (9): 74 - 77.

[2] HASTIE T T R F J. Overview of supervised learning [M]. The Elements of Statistical Learning, 2009: 9 - 41.

[3] ALBAWI S, MOHAMMED T A, ALZAWI S. Understanding of a Convolutional Neural Network [C] //The International Conference on Engineering and Technology, 2017.

[4] WATKINS C J C H D P. Q-learning [J]. M-achine Learning, 1992, 8 (3/4): 279 - 292.

[5] MNIH V, KAV K, SILVER D, et al. Human-level control through deep reinforcement learning [J]. Nature, 2015, 518 (7540): 529 - 533.

[6] 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.

[7] DAVID S, HUANG A J A, MADDISON C J, et al. Mastering the game of go with deep neural networks and tree search [J]. Nature. 2016, 529 (7587): 484 - 489.

[8] SILVER D, S J, S K, et al. Mastering the game of Go without human knowledge [J]. Nature, 2017, 550: 354 - 359.

[9] 梁　晨，刘小雄，张兴旺，等. 基于强化学习的四旋翼无人机控制律设计 [J]. 计算机测量与控制，2021，29 (2)：71 - 75.

[10] 单　楠. 基于深度强化学习的无人车控制策略研究 [D]. 哈尔滨：哈尔滨工业大学，2020.

[11] 江其洲，曾　碧. 基于深度强化学习的移动机器人导航策略研究 [J]. 计算机测量与控制，2019，27 (8)：217 - 221.

[12] 周燕艳. 改进的 Q 学习算法及在其 Robo-Cup 中的应用 [J]. 四川理工学院学报（自然科学版），2011，24 (4)：417 - 421.

[13] RUMELHART D E H G E W. Learning representations by back-propagating errors [J]. Nature, 1986, 323 (6088): 533 - 536.

[14] 斯俊烨. 基于嵌入式平台的神经网络加速器研究 [D]. 上海：上海交通大学，2019.

[15] 解　谦，张　睿，刘　红. 移动智能终端基于神经网络的人工智能技术与应用 [J]. 信息通信技术与政策，2019 (12)：45 - 50.

[16] 邹丹音. 基于深度学习的目标检测算法 FPGA 实现 [D]. 哈尔滨：哈尔滨工业大学，2019.

[17] 王　巍，周凯利，王伊昌，等. 卷积神经网络（CNN）算法的 FPGA 并行结构设计 [J]. 微电子学与计算机，2019，36 (4)：57 - 62.

[18] 高　琛，张　帆. 基于 FPGA 的递归神经网络加速器的研究进展 [J]. 网络与信息安全学报，2019，5 (4)：1 - 13.

[19] SHAO S, LUK W. Customised pearlmutter propagation: A hardware architecture for trust region policy optimization [C] // International Conference on Field Programmable Logic & Applications, 2017: 1 - 6.

[20] Shao S T J M M. Towards hardware accelerated reinforcement learning for application-specific robotic control [C] //International Conference on Field Programmable Logic & Applications, 2017: 1 - 6.

[21] JIANG S, LIU J, THOMAS D B, et al. Neural network based reinforcement learning acceleration on FPGA platforms [J]. Computer Architecture Mews, 2017, 44 (4): 68 - 73.