ISSN 1004-9037,CODEN SCYCE4
Journal of Data Acquisition and Processing Vol. 38,No. 5,Sep. 2023,pp. 1151－1161
DOI:10. 16337/j. 1004-9037. 2023. 05. 013

http://sjcj. nuaa. edu. cn
E-mail:sjcj@nuaa. edu. cn
Tel/Fax:＋86-025-84892742

基于深度强化学习的雷达智能抗干扰决策FPGA加速器设计

李梓瑜，葛　芬，张劲东，赵家琛

（南京航空航天大学电子信息工程学院，南京 211106）

摘　要：针对高动态环境下的雷达连续智能抗干扰决策和高实时性需求问题，本文构建了一种适用于雷达智能抗干扰决策的深度Q网络（Deep Q network，DQN）模型，并在此基础上提出了一种基于现场可编程门阵列（Field programmable gate array，FPGA）的硬件决策加速架构。在该架构中，本文设计了一种雷达智能决策环境交互片上访问方式，通过片上环境量化存储和状态迭代计算简化了DQN智能体连续决策时的迭代过程，在实现智能体深度神经网络的并行计算与流水控制加速的同时，进一步提升了决策实时性。仿真和实验结果表明，在保证决策正确率的前提下，所设计的智能抗干扰决策加速器相比已有的基于CPU平台的决策系统，在单次决策中实现了约46倍的速度提升，在连续决策中实现了约84倍的速度提升。

关键词：深度强化学习；深度Q网络；加速器；现场可编程门阵列；雷达智能决策

中图分类号：TN911.7　　**文献标志码：**A

Design of FPGA Accelerator for Radar Intelligent Anti-jamming Decision-Making Based on Deep Reinforcement Learning

LI Ziyu，GE Fen，ZHANG Jindong，ZHAO Jiachen

（College of Electronic and Information Engineering，Nanjing University of Aeronautics & Astronautics，Nanjing 211106，China）

Abstract：Aiming at the continuous intelligent anti-jamming decision-making and high real-time requirements of radar in high-dynamic environment，this paper constructs a deep Q network（DQN）model for radar intelligent anti-jamming decision-making，and proposes a hardware decision acceleration architecture based on field programmable gate array(FPGA). In this architecture，an on-chip access mode is designed for radar intelligent decision-making environment interaction to improve real-time performance，which simplifies the iterative process of continuous decision-making of the DQN agent through the on-chip quantitative storage and state iterative calculation for environment interaction. In the proposed architecture，both the parallel computing and pipeline control acceleration of agent deep neural network are adopted，which further improves the real-time performance of decision-making. Simulation and experimental results show that，on the premise of ensuring the accuracy of decision-making，the designed intelligent anti-jamming decision-making accelerator achieves a speedup of nearly 46 times in single decision-making and a speedup of nearly 84 times in continuous decision-making compared with the existing decision-making system based on the CPU platform.

Key words: deep reinforcement learning; deep Q network(DQN); accelerator; field programmable gate array(FPGA); radar intelligent decision-making

收稿日期：2022-04-13；**修订日期：**2022-08-28

引　言

深度强化学习(Deep reinforcement learning,DRL)由强化学习与深度学习结合而成,其同时拥有深度学习的感知能力和强化学习的决策能力,克服了传统强化学习算法处理复杂环境能力的不足,被应用于许多需要感知原始高维度输入和决策的任务中[1-3]。深度Q网络(Deep Q network,DQN)作为深度强化学习的典型算法,在实现智能决策方面拥有良好的表现[4]。DQN智能体通过深度神经网络(Deep neural network,DNN)将高维度的环境映射为可执行动作,从而得出应对复杂环境的策略。由于DQN决策计算过程具有较大的计算量与计算复杂度,传统的中央处理器(Central processing unit,CPU)无法满足一些智能决策场景的实时性要求[5]。而根据深度强化学习计算过程合理设计硬件结构,不仅可以有效提高智能体决策的实时性,还能降低存储资源与功耗开销[6-7]。

在硬件加速平台中,现场可编程门阵列(Field programmable gate array,FPGA)凭借其可重构、高能效比、高性能、便携且延迟低等优势,成为近年来许多算法加速领域最重要的手段。目前已有一些研究工作针对不同深度强化学习算法研究FPGA加速器设计[8-11],其中文献[8]分析了DQN算法中决策计算、梯度计算和参数更新计算的乘累加相似性,在FPGA上使用多模式矩阵乘法器与加法树相结合完成DQN决策与训练,与CPU相比实现了最高77倍速提升。文献[9]充分挖掘了深度强化学习的计算并行性并在FPGA平台上进行实现,与CPU相比实现了43倍的计算速度提升。上述加速器设计工作重点解决了DQN训练阶段的决策计算与参数更新问题,其中决策交互过程依赖于上位机CPU与FPGA的数据传输。但是,在DQN决策阶段的连续决策过程中,其决策速度不仅取决于智能体计算速度,同时也取决于DQN智能体与环境交互的速度,通过对交互过程进行硬件加速设计可有效提升决策实时性。

与其他强化学习应用领域相比,雷达智能决策系统需要在高动态环境下连续智能决策,具备实时性高、准确性高的要求。文献[12]使用Q学习算法实现了某干扰场景下的雷达智能抗干扰决策,文献[13]通过Q学习实现了最优电子战干扰实施策略的制定,文献[14]提出了一种认知雷达的DQN控制方式,提高了雷达在复杂光谱下的探测性能。这些工作都是强化学习算法在雷达智能化方面的应用与实现,但均没有涉及智能算法硬件加速器的设计。

本文对适用于雷达智能抗干扰决策的DQN模型进行研究,并且针对DQN雷达智能决策系统的特点完成了DQN环境交互过程、动作选择以及智能体DNN网络的设计。在此基础上,提出了一种基于FPGA的DQN雷达智能抗干扰决策加速器架构,并重点针对DQN雷达智能决策系统连续决策的特点对环境交互过程进行优化,设计了一种片上状态迭代方式,实现了DQN雷达智能体与环境的快速交互,在满足决策准确性的同时提升实时性。

1　基于DQN的雷达智能抗干扰决策模型

1.1　DQN雷达智能抗干扰决策模型

DQN算法原理如下:使用状态集$\boldsymbol{S}$,将t时刻环境观察值描述为$\boldsymbol{S}_t$,由智能体使用深度神经网络将其映射为动作-状态价值$\boldsymbol{Q}_t$,根据该映射值使用如式(1)所示的ε-贪婪算法从动作集$\boldsymbol{A}$中选择动作a_t,其中ε为贪婪算子,$\boldsymbol{Q}_t$为智能体t时刻输出。在a_t作用于环境后再由环境给出立即回报r_t和$t+1$时刻观察值$\boldsymbol{S}_{t+1}$,并重复该交互过程直至结束。在智能体与环境不断的交互中,通过深度学习的参数训练方式使智能体映射结果收敛,从而可以将贪婪算子ε下降为0来选出最优动作[4]。

$$a_t=\begin{cases}\arg\max \boldsymbol{Q}_t(\boldsymbol{S}_t,a) & p=1-\varepsilon\\ \text{random}(a) & p=\varepsilon\end{cases} \tag{1}$$

在DQN雷达智能抗干扰决策模型中,雷达智能决策系统可以根据当前雷达工作状态以及由接收

的回波信号经过提取后得到的关于目标和场景的信息,在信号处理环节合理选择副瓣对消、副瓣匿影、捷变频和盲源分离等手段和次序,优化抗干扰性能,使雷达更好地适应复杂电磁环境。

上述抗干扰手段和次序可以通过如图1所示的DQN算法模型进行决策。将雷达的当前工作状态以及提取的目标和场景信息作为状态集$\boldsymbol{S}$,信号处理环节可使用手段组成动作集$\boldsymbol{A}$,使用卷积神经网络(Convolution neural network,CNN)进行Q值计算,使用ε-贪婪算法作为动作选择探索方法。将该DQN雷达智能决策模型训练后进行雷达抗干扰决策,可以有效地提升决策的速度和智能化水平。

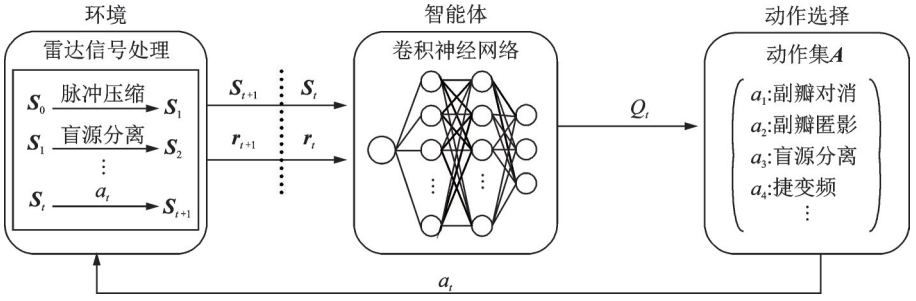

图1 DQN雷达智能决策模型

Fig.1 Model of DQN radar intelligent decision-making

1.2 环境状态表示与迭代

在图1所示环境模块中,需要提取出雷达智能决策系统的输入,该输入是指雷达当前状态和提取的外部干扰信息数据,具有种类多、变化范围大且维度高等特点。将同一时刻的外部干扰信息数据按行列排列,可以形成$m\times n$的二维混合类型状态矩阵$\boldsymbol{S}$,作为DQN模型的状态集,如图2所示。其中,列表示所提取干扰源的不同数据类型信息,行表示由不同数据类型信息描述的一条干扰,n代表提取信息种类上限,m代表同时可处理干扰数目上限。在本文所仿真的大部分干扰场景下,状态矩阵$\boldsymbol{S}$不同列主要包含干扰强度P、干扰角度θ和干扰类型i等多种类型数据,每一行状态分量则表示为一条干扰信息。

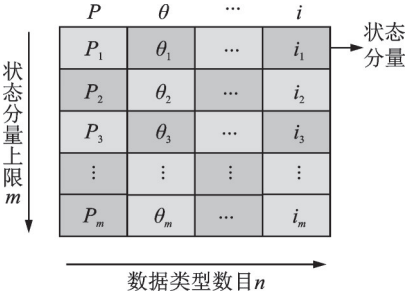

图2 混合类型状态矩阵

Fig.2 Mixed-type state matrix

当雷达智能抗干扰决策系统作出决策后,需要在雷达信号处理系统中对所提取的信号进行处理,并重新提取处理后的数据将状态矩阵迭代至$\boldsymbol{S}_{t+1}$,然后重复决策和交互过程直至结束。在该过程中,每次决策只能得到单步的决策结果,但是在雷达抗干扰策略制定中,抗干扰操作是以组合的形式来去除干扰,单步的抗干扰处理并不能有效地去除干扰。而且在连续决策中,得到一条完整决策序列所需要的时间不仅取决于深度网络决策计算时间,也取决于智能体与环境的交互速度与状态迭代速度,通过优化该过程可有效提升连续决策速度,使决策结果更具实时性。

1.3 智能体的CNN结构

在DQN雷达智能抗干扰决策流程中,智能体部分是指将状态输入矩阵$\boldsymbol{S}$映射为Q值输出的CNN,它由输入层、隐藏层和输出层构成,其中隐藏层由多个卷积层、全连接层构成,输出层节点数目由可执行动作数目k决定。输入层需要将状态矩阵$\boldsymbol{S}$作为卷积神经网络输入,由隐藏层完成卷积、全连接和激活等操作后,再由输出层输出至k个输出节点。目前,DQN算法中还没有固定的CNN模型,根据DQN决策场景不同,所使用CNN的结构与参数也有所不同[15-18]。在本文使用的模型中,状态矩阵$\boldsymbol{S}$每一行

分别代表着不同类型的干扰，且不同干扰数据之间的物理含义联系较弱，因此采用$1\times n$的卷积核进行卷积，如图3所示，激活函数使用线性修正单元(Rectifier linear unit，ReLU)进行计算。

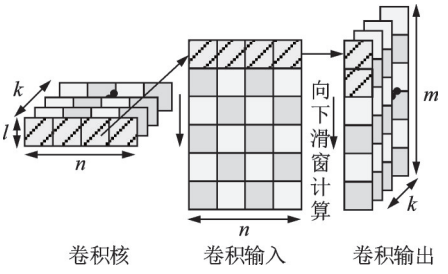

图3 DQN雷达智能决策卷积计算过程

Fig.3 Convolution calculation process of DQN radar intelligent decision-making

DQN雷达智能抗干扰决策的CNN计算速度直接关系到其决策性能，而CNN计算主要集中在卷积层和全连接层，通过对其进行有针对的硬件加速设计，可有效减少智能决策时间，提升决策效率。

1.4 动作集的避错与停止设计

在动作选择部分，需要根据动作探索方式与智能体t时刻输出结果Q_t，从动作集A的k种信号处理方式中选择对应动作a_t。

由于在雷达抗干扰决策问题中，部分被提取信息的处理手段有着较为严格的使用前提和次序要求，当不满足次序要求时，很可能会无法与后续信息处理手段进行交互；部分信息处理手段在智能体决策时要求避免被重复选择，例如，重复选择相同抗干扰手段可能会丢失回波中的目标信息并且导致信噪比降低。另外，由于雷达智能抗干扰决策场景的复杂性，无法直接定义雷达工作性能的最优标准，因此无法直接给出探索过程的结束条件。

为解决上述问题，本文将t时刻可执行的k个动作中，会造成错误或重复的动作标记为错误动作a_{wrong}，当智能体选择a_{wrong}时将不对雷达状态和信息数据进行调整或处理，并且结束探索，以避免错误动作与重复动作对智能体与环境造成影响，加快DQN模型的收敛速度。同时，本文提出在动作集中加入停止动作a_{stop}，当智能体选择a_{stop}时会结束当前迭代探索过程。该动作意义在于当任何动作都无法继续优化结果时，智能体会选择停止动作来结束当前决策，由此可以使智能体在合理的时间内找到最优决策，使雷达信号处理系统工作性能达到最优。

2 基于FPGA的DQN雷达智能抗干扰决策加速器架构

2.1 总体架构

本文从DQN雷达智能抗干扰决策的应用角度出发，注重智能抗干扰决策的实时性，提出一种基于FPGA的片上DQN决策加速器架构，如图4所示，该片上架构主要由CNN计算单元、环境量化模块和动作选择输出模块构成。

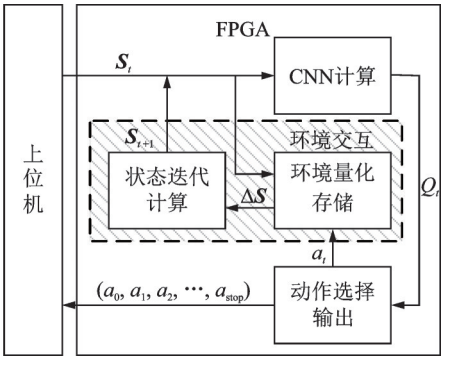

图4 DQN雷达智能决策加速器架构

Fig.4 DQN radar intelligent decision-making accelerator architecture

上位机从t时刻的雷达状态和外部干扰信息中提取出初始状态矩阵$\boldsymbol{S}_t$作为FPGA加速器输入。由CNN计算单元模块完成DQN决策中卷积神经网络的加速计算，由动作选择输出模块根据CNN计算单元的计算结果Q_t选择动作a_t，并输出至环境量化模块，同时对a_t进行存储。环境量化模块需要根据当前状态矩阵$\boldsymbol{S}_t$和动作a_t完成$\boldsymbol{S}_t$向$t+1$时刻状态矩阵$\boldsymbol{S}_{t+1}$的迭代过程，并将$\boldsymbol{S}_{t+1}$作为下一时刻状态矩阵输入CNN计算单元。重复上述循环直至选择停止动作a_{stop}后，由动作选择模块将抗干扰决策动作序列$\left(a_0, a_1, a_2, \cdots, a_{\text{stop}}\right)$输出至上位机，进行后续雷达状态调整和信息处理。

2.2 雷达抗干扰决策处理环境量化

环境量化是指对于状态矩阵$\boldsymbol{S}_t$迭代为$\boldsymbol{S}_{t+1}$过程中的变化量矩阵$\Delta\boldsymbol{S}$,通过采样和量化后用数值化的方式进行表示与存储,这样在状态迭代时可以直接根据$\boldsymbol{S}_{t+1}=\boldsymbol{S}_t+\Delta\boldsymbol{S}$完成迭代。通常情况下,DQN决策加速器系统需要将当前决策动作a_t传输至片外上位机的雷达信号处理模块或雷达状态控制模块并执行,再由上位机根据$t+1$时刻外部信息和雷达状态提取出新的状态矩阵$\boldsymbol{S}_{t+1}$。这意味着每进行一次单步抗干扰决策,FPGA加速器就需要与上位机进行一次数据传输,并且等待数据处理,待上位机完成状态提取后再进行下一步决策。在该过程中,每次决策都要等待单步处理完毕后,再进行下一个抗干扰操作选择,在决策时间上有一定的滞后,同时频繁的片内外数据传输也会增加决策时间。

为此,本文提出了一种雷达抗干扰决策处理环境量化方法,并在FPGA加速器中设计了环境量化存储和状态迭代计算模块,如图4虚线框所示,将量化数值存储于片上,在片上完成状态迭代过程。在所提出的环境量化方法中,首先将状态集中的不同类型数据按照数值区间进行划分,然后在各区间内均匀采样,并将采样数据进行排列组合,组成状态矩阵集合,最后将这些状态矩阵作为初始状态$\boldsymbol{S}_t$,用动作集$\boldsymbol{A}$中所有动作进行处理,得到处理结果$\boldsymbol{S}_{t+1}$。将$\boldsymbol{S}_t$由不同动作a_t变为$\boldsymbol{S}_{t+1}$的平均变化量$\Delta\boldsymbol{S}$记录并存储于片上,在状态迭代时,只需对当前状态矩阵$\boldsymbol{S}_t$进行解析,在片上存储区域中找到$\boldsymbol{S}_t$在动作a_t作用下的平均变化量$\Delta\boldsymbol{S}$,按照$\boldsymbol{S}_{t+1}=\boldsymbol{S}_t+\Delta\boldsymbol{S}$的关系即可完成$\boldsymbol{S}_t$向$\boldsymbol{S}_{t+1}$的迭代。

由于对状态集各类型数据进行了区间划分,同时对各动作处理$\boldsymbol{S}_t$后的变化量$\Delta\boldsymbol{S}$进行了平均化处理,会将部分类型数据(如角度、功率等取值)由连续量变为离散量,因此,该迭代方法得到的$\boldsymbol{S}'_{t+1}$与通常迭代过程得到的$\boldsymbol{S}_{t+1}$在数值上会有差异。该差异会导致DQN决策中,深度神经网络计算结果Q_t产生误差,本文将该误差称为迭代误差σ。该误差可能会引起决策结果不同,从而导致产生错误决策。根据1.1节所述,在DQN决策阶段,智能体只会选择最大Q值所对应的动作,因此只需要使t时刻迭代误差σ_t满足式(2),即可避免错误决策的产生。

$$\operatorname{argmax}(Q_t)=\operatorname{argmax}(Q_t+\sigma_t) \tag{2}$$

通过片上状态迭代,可以有效减少连续决策片内外数据传输次数,进一步提升决策速度。所设计的决策系统只需要进行一次初始状态输入,就能在短暂时间内得到完整的抗干扰操作序列输出。

3 基于FPGA的DQN决策加速器设计

3.1 环境量化存储与状态迭代计算

环境量化存储模块需要在连续抗干扰决策中完成状态迭代,并将迭代后的状态矩阵输入至后续模块进行决策计算。环境量化后的数据具有数量多、规模大的特点,为了减少存储开销,同时简化查询逻辑,需要对存储结构与查询方式进行优化设计。

本文将存储器划分为多个子区域,同时将环境量化值$\Delta\boldsymbol{S}$按照其元素组成类型进行分类,把不同元素组合量化后的$\Delta\boldsymbol{S}$分别存于不同的子区域内,如图5所示。例如,当雷达抗

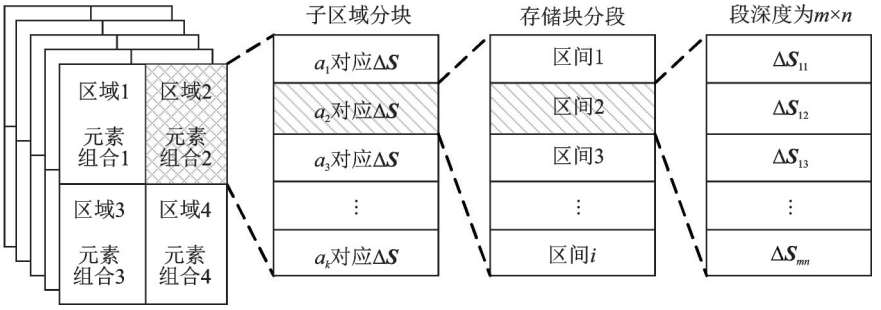

图5 环境量化存储分区

Fig.5 Environment quantization storage partition

干扰场景仅包含A、B两类干扰时，将仅由A类干扰组成$\boldsymbol{S}$所对应$\Delta\boldsymbol{S}$存储于区域1，B类干扰组成$\boldsymbol{S}$所对应$\Delta\boldsymbol{S}$存储于区域2，A类与B类混合干扰组成$\boldsymbol{S}$所对应$\Delta\boldsymbol{S}$存储于区域3，当干扰类型数目增加时按照该方式增加分区进行存储。

子区域内部，设置存储位宽为8位，数据使用8位定点形式表示。因为不同的决策动作$a_i(i=1,2,\cdots,k)$带来的$\Delta\boldsymbol{S}$不同，因此将子区域分为多个存储块，分别存储不同动作所对应的$\Delta\boldsymbol{S}$。

在各存储块中，通过将存储块划分为多个存储段对不同数值区间的$\Delta\boldsymbol{S}$进行存储，各存储段使用相同的存储深度对$\Delta\boldsymbol{S}$所有数值进行存储。以$m\times n$的状态矩阵$\boldsymbol{S}$为例，由于其$\Delta\boldsymbol{S}$由$m\times n$个数值构成，因此一次读数深度为$m\times n$，所以一个存储段深度即为$m\times n$。

根据$\Delta\boldsymbol{S}$的分区存储方式，其存储地址由区域地址、块地址与段地址3部分组成。在寻址时先对当前输入状态矩阵$\boldsymbol{S}$的构成元素进行解析，确定$\Delta\boldsymbol{S}$所在子区域地址x_1；同时对$\boldsymbol{S}$不同类型的数据大小进行解析，确定$\Delta\boldsymbol{S}$所在数据段地址x_3；最后等待决策动作产生后得到$\Delta\boldsymbol{S}$所在块地址x_2，从而得出其精确地址，如图6所示。通过该查询方式可以快速从大量的存储数据中，找到目标$\Delta\boldsymbol{S}$所在位置。

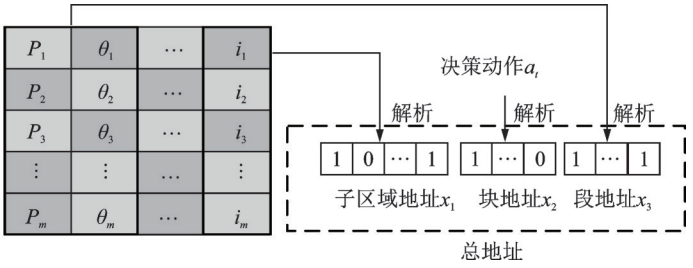

图6 查表地址解析

Fig.6 Table address calculation

当状态矩阵$\boldsymbol{S}$包含元素数目m'少于上限m时，会出现$m-m'$行全0元素。这些全0元素并不会因为动作a产生变化，其对应$\Delta\boldsymbol{S}$为0。因此，在量化存储时可以除去这些全0元素，以减少存储资源开销。

状态迭代计算时，只需要使用多个加法器并行，按照$\boldsymbol{S}_{t+1}=\boldsymbol{S}_t+\Delta\boldsymbol{S}$关系完成计算，即可得到下一轮神经网络输入$\boldsymbol{S}_{t+1}$。

3.2 CNN加速设计

本文通过使用CNN控制模块，控制多个处理单元(Processing element，PE)并行的方式，分别实现卷积层和全连接层的并行计算与流水控制，如图7所示。其中，中间结果缓存用于存储层间计算数据，部分和结果缓存用于存储层内节点计算产生的部分和数据。

卷积层计算时，使用一个PE完成一个卷积窗口的滑窗计算，多个PE并行实现不同卷积窗口之间

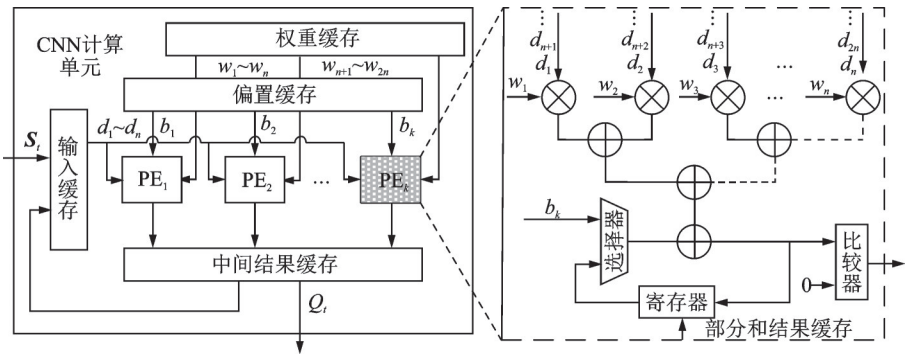

图7 CNN计算单元与PE内部结构

Fig.7 CNN computing unit and PE internal structure

的计算并行。在PE内部,将卷积窗口内n个数据$d_i(i\in1,2,\cdots,n)$并行输入,同时由权重缓存给出卷积核权重$w_i(i\in1,2,\cdots,n)$,分别与d_i相乘。在经过$\lfloor\log_2 n\rfloor$级加法后,与偏置b_i相加,然后将结果与0比较完成ReLU激活。卷积层计算结果将输出至中间结果缓存,作为下一层运算输入数据。

全连接层计算时,使用一个PE完成一个输出节点的计算,多个PE并行实现不同输出节点的计算并行。当全连接层输入节点数l大于PE内部乘法器数目n时,需要对PE内部资源进行复用来完成一个输出节点的计算。在PE内部,将上一层l个计算结果前n个数据$d_i(i\in1,2,\cdots,n)$并行输入,同时由权重缓存给出卷积核权重$w_i(i\in1,2,\cdots,n)$分别与d_i相乘。在经过$\lfloor\log_2 n\rfloor$级加法后,与偏置b_i相加并将该部分和存于部分和缓存中,等待与后续n个输入节点的乘累加结果求和,直至l个节点全部输入完毕,将累加结果进行激活计算并输出。当输出节点数目大于PE数目j时,可以等待前j个输出节点计算完成后,再对整体PE间资源进行复用直至完成后续输出节点计算。

在执行卷积计算时,在第1个时钟周期输入前n个数据$d_1\sim d_n$,第2个时钟周期输入后续n个数据$d_{n+1}\sim d_{2n}$,以此类推将数据按照时钟周期连续输入,实现卷积计算的流水控制。在执行全连接计算时,不仅需要将输入数据进行连续输入,还需要将数据所对应的权重数据进行连续输入,以实现全连接层的流水控制。

3.3 动作选择输出模块设计

动作选择输出模块在进行动作选择时,需要从CNN输出的k个结果中选择出最大Q值所对应动作。将CNN计算结果$Q_i(i\in1,2,\cdots,k)$并行输入,通过使用$\lfloor\log_2 k\rfloor$级比较,从k个结果中筛选出最大Q值,最后将该值所对应的动作a_t存入动作输出缓存并输出至环境量化存储模块,如图8所示。

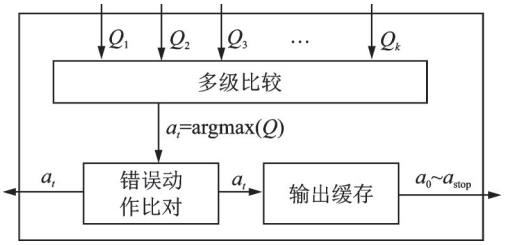

图8 动作选择输出模块

Fig.8 Action selection output module

当选择动作a_{stop}或决策动作数目累计达到上限时,将动作输出缓存中的动作序列一并输出至上位机,进行后续的雷达状态控制与信号处理。为了避免选择错误动作a_{wrong},本设计预先将错误动作序列存储于片上,并且在每次动作决策后,将决策动作序列与错误动作序列进行比对,当出现错误决策时立即终止决策,并反馈至上位机。

4 实验结果及分析

4.1 实验环境与参数设置

本文使用的DQN雷达智能抗干扰模型各参数如下:设置干扰种类上限数目为4,同时存在干扰数目上限为6,将干扰源的干扰强度、角度、干扰源数目和干扰源类型作为构成混合状态矩阵$\boldsymbol{S}$的4种数据类型;设置包含停止动作与7种抗干扰操作的动作集$\boldsymbol{A}$。使用如表1所示CNN网络进行决策计算。

表1 CNN模型

Table 1 CNN model

网络层类型	参数量	激活函数
输入层	6×4	无
卷积层	1×4×16	ReLU
全连接层1	96×32	ReLU
全连接层2	32×8	无
输出层	8	无

本文所提出加速器架构实现使用的FPGA平台是XILINX公司的Zynq UltraScale+ ZCU106开发板,加速器综合后FPGA工作频率为200 MHz。对比实验中CPU采用Intel Xeon Gold 5215处理器,主频为2.50 GHz;GPU采用NVIDIA GeForce RTX 2080 Ti,核心频率为1 350 MHz。

其中，CPU正向推理过程使用numpy1.22库实现，GPU软件代码采用TensorFlow2.0深度学习框架，FPGA硬件实现环境为Xilinx Vivado 2018。

4.2 DQN决策功能验证

为验证所设计DQN雷达智能抗干扰决策加速器的功能正确性，本文将4种类型干扰随机组合，产生3 000个初始状态矩阵$\boldsymbol{S}$作为测试集，并将测试集分别在FPGA和CPU平台上进行DQN雷达智能决策，得到3 000条完整的抗干扰操作决策结果，其中每条完整决策的动作序列长度即为连续决策次数。以CPU平台决策结果为参照，将决策结果按照连续决策次数进行分类，对硬件加速器决策结果进行统计，与CPU平台决策动作结果相同即为正确，同时对所有样本在两种决策平台下，每次CNN计算结果中最大Q值的平均值进行对比，结果分别如图9、10所示。

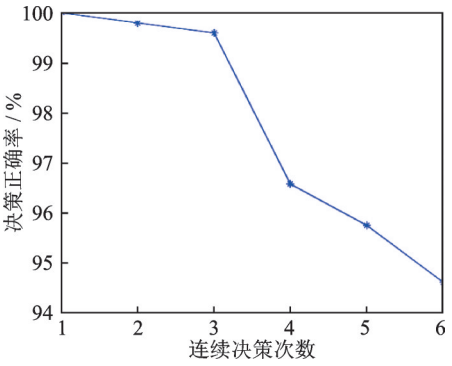

图9 硬件加速器决策正确率

Fig.9 Hardware accelerator decision-making correct rate

由图9可知，在决策次数为1时硬件加速器决策正确率为100%，该结果说明加速器单次决策结果均与CPU平台决策结果一致，验证了DQN决策的功能正确性。随着连续决策次数增加，硬件决策正确率逐渐下降，当连续决策次数为6时，硬件决策正确率为94.61%。这是由于片上环境交互方式在状态迭代中引入了迭代误差σ，从而导致决策正确率下降，但仍然在90%以上，可以满足设计需求。

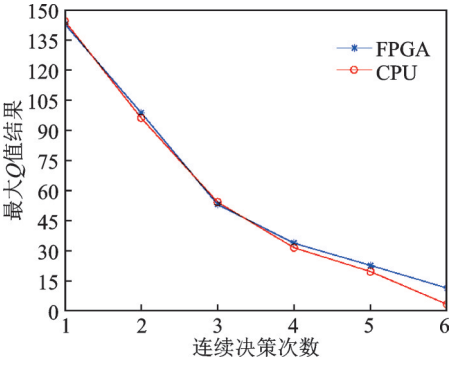

图10 最大Q平均值对比

Fig.10 Comparison of maximum Q averages

由图10可知，在DQN决策模型中，随着连续决策次数增加，决策计算结果中的最大Q值会逐渐减小，这会导致CNN计算结果对误差σ更加敏感。因此，在误差相近的情况下，随着连续决策次数的增加，决策正确率也会逐渐下降。而通过增加状态迭代查表模块的$\Delta\boldsymbol{S}$区间划分数目，可以减少迭代误差σ，从而减少连续决策带来的正确率下降。在上述3 000个初始状态矩阵测试集下，$\Delta\boldsymbol{S}$区间划分数目与平均决策正确率如图11所示。

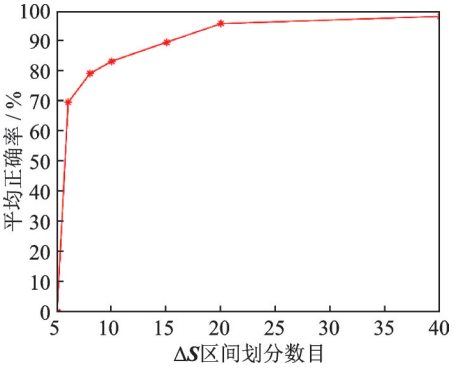

图11 分区数目与决策正确率

Fig.11 Number of segments and decision-making correct rate

在图11中，平均正确率是通过对测试集中不同长度决策结果的正确率求平均值后得到。由图11可知，当$\Delta\boldsymbol{S}$区间划分数目由5增加至40时，平均正确率也由0上升至98%。需要指出，随着分区数目增加，也会带来更大的存储消耗。当分区数目为20时，连续抗干扰决策的平均正确率为95.6%，因此综合资源与设计需求考虑，本文使用20作为$\Delta\boldsymbol{S}$区间划分数目。

4.3 加速器性能评估

使用4.1节DQN雷达智能抗干扰决策模型，分别在图形处理器(Graphics processing unit，GPU)、CPU和FPGA平台上进行单次DQN雷达智能决策，其中单次决策不需要进行状

态迭代。3种决策平台完成决策所需要的决策时间结果如表2所示。由表2可知,在4.1节的DQN智能决策模型下,本文设计加速器相较于GPU平台,在单次决策任务中加速比达到近232倍,而相较于CPU平台,在单次决策任务中加速比达到约46倍。在DQN决策过程中,输入数据是在智能体与环境交互中过程中得到,而每次交互只能得到一张输入状态图,无法充分使用GPU的大规模并行计算资源,使GPU并行结构利用率较低,同时由于GPU数据通信和同步的成本较高,因而在决策中耗时最长。而本文所设计加速器通过PE内部计算并行、PE间的计算并行和数据流水控制的方式实现了单次决策的速度提升,在3种平台中决策耗时最短。在功耗开销上,CPU平台功耗为9.95 W,GPU平台功耗为11.92 W,而加速器功耗为0.916 W,分别为CPU、GPU的9.21%与7.68%,具有极高的能耗优势。

表2 单次决策比较

Table 2 Comparison of single decision-making

硬件平台	时间/μs	功耗/W	频率/GHz
CPU	25.10	9.95	2.5
GPU	125.21	11.92	1.35
FPGA	0.54	0.916	0.20

为了对加速器的连续决策性能进行验证,本文使用CPU与FPGA平台同时进行多次决策任务,其中一次决策任务指输入初始状态矩阵$\boldsymbol{S}$后,连续决策得到完整抗干扰操作序列的过程,且每次任务所需决策次数上限不超过6次。两种平台在多次决策任务中的时间开销如表3所示。由表3可知,在多次连续决策任务中,加速器相较于CPU平台的平均加速比约为84倍,相较于完成单次无状态迭代决策任务的加速比有明显提高。由于连续决策任务中,需要多次进行状态迭代,而本文设计的片上的环境交互模块通过对环境进行量化后存于片上RAM,简化了状态迭代过程,实现状态矩阵的快速迭代,使加速比相较于单次决策进一步提升。由结果可知,所设计的加速器可以快速完成DQN雷达智能抗干扰决策中的状态迭代过程,输出完整抗干扰决策序列,提高决策的实时性。

表3 连续决策运行时间

Table 3 Continuous decision-making running time

决策任务数目	CPU运行时间/s	FPGA运行时间/s	加速比
1 000	0.34	4.14×10^{-3}	82
2 000	0.73	8.29×10^{-3}	88
5 000	1.72	2.088×10^{-2}	82
8 000	2.61	3.307×10^{-2}	79
10 000	3.56	4.023×10^{-2}	88

本文所设计的DQN加速器资源开销如表4所示。由于在本文提出架构中,环境量化数据存储于片上,因此消耗了41.36%的查找表寄存器(Look-up table RAM,LUTRAM)资源,但是通过3.1节所述的存储优化方式,删去表格中的全0数据,可以将该资源开销下降至25.38%。在雷达智能抗干扰决策系统中,快速决策更有利于提升系统性能,能保证雷达系统对实时性的要求,而存储资源消耗相较决策速度,对性能影响较小。因此,本文提出的片上环境交互设计更有利于DQN雷达智能决策系统性能提升。

表4 资源开销

Table 4 Resource consumption

资源名称	资源占用数目	资源占用比/%
LUTRAM(优化前)	42 092/101 760	41.36
LUTRAM(优化后)	25 832/101 760	25.38
块寄存器(Block RAM,BRAM)	29.5/312	9.46
查找表(Look-up table,LUT)	50 169/230 400	21.77
触发器(Flip-Flop,FF)	8 983/460 800	1.95
数字信号处理器(Digital signal processor,DSP)	18/1 728	1.04

5 结束语

本文构建了一种DQN雷达智能抗干扰决策模型,通过引入错误动作与停止动作优化了决策动作

选择过程，满足了雷达决策中的避错要求。在此基础上，提出并设计实现了一种基于FPGA的片上DQN决策加速器。该加速器通过并行计算、流水线结构以及资源复用提高CNN计算性能，降低硬件资源开销；并采用环境量化存储和状态迭代计算的方式在片上实现了DQN连续决策中的状态迭代过程，有效提升了连续决策的速度。实验结果表明，相比CPU平台，本文所设计加速器在单次决策中可以获得约46倍的加速比，在连续决策过程中可以获得约84倍的加速比。

参考文献：

[1] 刘建伟，高峰，罗雄麟. 基于值函数和策略梯度的深度强化学习综述[J]. 计算机学报，2019，42(6)：1406-1438.
LIU Jianwei，GAO Feng，LUO Xionglin. Survey of deep reinforcement learning based on value function and policy gradient[J]. Chinese Journal of Computers，2019，42(6)：1406-1438.

[2] KONDA V R，TSITSIKLIS J N. Actor-citic agorithms[C]//Proceedings of the 12th International Conference on Neural Information Processing Systems (NIPS′99). Cambridge，MA，USA：MIT Press，1999：1008-1014.

[3] KENDALL A，HAWKE J，JANZ D，et al. Learning to drive in a day[C]//Proceedings of 2019 International Conference on Robotics and Automation (ICRA). Montreal，QC，Canada：IEEE，2019：8248-8254.

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

[5] WANG W，CASELLE M，BOLTZ T，et al. Accelerated deep reinforcement learning for fast feedback of beam dynamics at KARA[J]. IEEE Transactions on Nuclear Science，2021，68(8)：1794-1800.

[6] SHIRI A，PRAKASH B，MAZUMDER A N，et al. An energy-efficient hardware accelerator for hierarchical deep reinforcement learning[C]//Proceedings of the 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS). Washington DC，DC，USA：IEEE，2021：1-4.

[7] KIM C，KANG S，CHOI S，et al. An energy-efficient deep reinforcement learning accelerator with transposable PE array and experience compression[J]. IEEE Solid-State Circuits Letters，2019，2(11)：228-231.

[8] SU J，LIU J，THOMAS D B，et al. Neural network based reinforcement learning acceleration on fpga platforms[J]. ACM SIGARCH Computer Architecture News，2017，44(4)：68-73.

[9] GANKIDI P R，THANGAVELAUTHAM J. FPGA architecture for deep learning and its application to planetary robotics [C]//Proceedings of 2017 IEEE Aerospace Conference. Big Sky，MT，USA：IEEE，2017：1-9.

[10] CHO H，OH P，PARK J，et al. Fa3c：FPGA-accelerated deep reinforcement learning[C]//Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS). New York，NY，USA：ACM，2019：499-513.

[11] GEROGIANNIS G，BIRBAS M，LEFTHERIOTIS A，et al. Deep reinforcement learning acceleration for real-time edge computing mixed integer programming problems[J]. IEEE Access，2022，10：18526-18543.

[12] 汪浩，王峰. 强化学习算法在雷达智能抗干扰中的应用[J]. 现代雷达，2020，42(3)：40-44.
WANG Hao，WANG Feng. Application of reinforcement learning algorithms in anti-jamming of intelligent radar[J]. Modern Radar，2020，42(3)：40-44.

[13] 李云杰，朱云鹏，高梅国. 基于Q-学习算法的认知雷达对抗过程设计[J]. 北京理工大学学报，2015，35(11)：1194-1199.
LI Yunjie，ZHU Yunpeng，GAO Meiguo. Design of cognitive radar jamming based on q-learning algorithm[J]. Transaction of Beijing Institute of Technology，2015，35(11)：1194-1199.

[14] THORNTON C E，KOZY M A，BUEHRER R M，et al. Deep reinforcement learning control for radar detection and tracking in congested spectral environments[J]. IEEE Transactions on Cognitive Communications and Networking，2020，6(4)：1335-1349.

[15] DONG Z，WANG H，WANG Z，et al. Research on gait planning of humanoid robot climbing based on DQN algorithm[C]//Proceedings of the 3rd Advanced Information Management，Communicates，Electronic and Automation Control Conference (IMCEC). Chongqing，China：IEEE，2019：1128-1131.

[16] ZHOU S，LIU X，XU Y，et al. A deep Q-network (DQN) based path planning method for mobile robots[C]//Proceedings of 2018 IEEE International Conference on Information and Automation (ICIA). Wuyishan，China：IEEE，2018：366-371.

[17] LI Kang, JIU Bo, LIU Hongwei, et al. Reinforcement learning based anti-jamming frequency hopping strategies design for cognitive radar[C]//Proceedings of 2018 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC). Qingdao, China: IEEE, 2018: 1-5.

[18] CHIEN J T, HUNG P Y. Multiple target prediction for deep reinforcement learning[C]//Proceedings of 2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC). Auckland, New Zealand: IEEE, 2020: 1611-1616.

作者简介:

李梓瑜(1998-),男,硕士研究生,研究方向:深度强化学习FPGA加速器设计,E-mail:liziyu@nuaa.edu.cn。

葛芬(1981-),**通信作者**,女,副教授,博士,研究方向:数字集成电路设计,E-mail:gefen@nuaa.edu.cn。

张劲东(1981-),男,教授,博士,研究方向:雷达信号处理,E-mail:zjdjs@126.com。

赵家琛(1998-),男,硕士研究生,研究方向:雷达信号处理、雷达智能抗干扰,E-mail:felixwin@nuaa.edu.cn。

(编辑:陈珺)