一种在线时间序列预测的核自适应滤波器向量处理器

庞业勇* 王少军 彭 宇 彭喜元
(哈尔滨工业大学自动化测试与控制研究所 哈尔滨 150080)

摘 要：针对信息物理融合系统中的在线时间序列预测问题，该文选择计算复杂度低且具有自适应特点的核自适应滤波器(Kernel Adaptive Filter, KAF)方法与 FPGA 计算系统相结合，提出一种基于 FPGA 的 KAF 向量处理器解决思路。通过多路并行、多级流水线技术提高了处理器的计算速度，降低了功耗和计算延迟，并采用微码编程提高了设计的通用性和可扩展性。该文基于该向量处理器实现了经典的 KAF 方法，实验表明，在满足计算精度要求的前提下，该向量处理器与 CPU 相比，最高可获得 22 倍计算速度提升，功耗降为 1/139，计算延迟降为 1/26。

关键词：核自适应滤波器；现场可编程逻辑门阵列；向量处理器；微码
中图分类号：TP391 **文献标识码**：A **文章编号**：1009-5896(2016)01-0053-10
DOI: 10.11999/JEIT150157

A Kernel Adaptive Filter Vector Processor for Online Time Series Prediction

PANG Yeyong WANG Shaojun PENG Yu PENG Xiyuan
(*Automatic Test and Control Institute*, *Harbin Institute of Technology*, *Harbin* 150080, *China*)

Abstract: To address the online time series prediction problem in CPS (Cyber-Physical System) system, both KAF (Kernel Adaptive Filter) with low computation complexity and adaptive characteristic and FPGA computing system are employed. A novel FPGA implementation of vector processor targeting KAF algorithm is proposed. The parallelized datapath and multi-stage pipeline are introduced to enhance the performance and reduce the power consumption and latency. The microcoding technology is further employed to improve the reusability and extensibility. The classical KAF algorithms are implemented based on the vector processor. Experiments results show that the proposed vector processor improves the execution speed by factors of 22, the power consumption decrease to 1/139, while the latency decrease to 1/26 compared with a CPU, on the condition that the precision meets the requirement.

Key words: Kernel Adaptive Filter (KAF); Filed-Programmable Gate Arrays (FPGA); Vector processor; Microcode

1 引言

信息物理融合系统(Cyber-Physical System, CPS)是将计算、通信和控制能力深度融合的网络化物理系统，数据的在线实时处理是 CPS 的核心问题之一[1]。而实际物理系统产生的数据往往具有时间序列特性，因此时间序列预测广受工业界和研究机构的关注，越来越多的嵌入式在线时间序列预测系统被广泛地应用到变电站无线监测与预警，可穿戴机器人运动控制以及嵌入式环境监测等领域。然而，对于在线应用，非线性时间序列预测方法需要不断地加入新样本并且对模型进行更新，导致所需要的计算量大大增加，从而限制了其在 CPS 等先进智能信息处理系统中的应用。针对此问题，已有若干研究[2−4]分别尝试将 SVM(Support Vector Machine), LS-SVM(Least Square-Support Vector Machine)和 RVM(Relevance Vector Machine)3 种在线时间序列预测方法在嵌入式系统实现，对于此类高性能、低功耗和低延迟的嵌入式在线时间序列预测应用，它们都是从高效率的计算方法和高性能的计算平台两个方面开展研究。

在线时间序列预测方法主要包含统计学方法、人工神经网络方法和以支持向量机为代表的核方法等几类，其中核方法将非线性问题通过 Mercer 核映射为核空间内的线性问题，利用相对简单的核函数运算，替代了核空间内复杂的内积计算，降低了计算复杂度，在时间序列预测领域被广泛使用，常见

收稿日期：2015-01-27；改回日期：2015-09-28；网络出版：2015-11-17
*通信作者：庞业勇 yeyongpang@126.com
基金项目：国家自然科学基金(61571160/F011305)，中央高校基本科研业务费专项资金资助(HIT.NSRIF.201615)
Foundation Items: The National Natural Science Foundation of China (61571160/F011305), Fundamental Research Funds for the Central Universities (HIT.NSRIF.201615)

的核方法分为两类：(1)一类是支持向量机、高斯过程、正则化网络等批处理方法，其全局最优化需要根据全部输入样本来进行建模，随着特征空间维数的增加，存在计算复杂度激增和正则化性能降低的问题；(2)另外一类是计算复杂度较低的核自适应滤波器(Kernel Adaptive Filter, KAF)方法，它随着新样本的到来以一种递归增量的形式更新状态模型，降低了计算复杂度和计算延迟。KAF 的典型方法包括核递归最小二乘(Kernel Recursive Least Square, KRLS)和核归一化最小均方(Kernel Normalized Least Mean Square, KNLMS)算法等，其中 KRLS 方法可以通过滑窗、固定预算、归一化等方式进一步降低计算成本，保证了其在线应用的低延迟特性，更适合于在线应用。

对于嵌入式计算平台，ARM, PowerPC 等嵌入式微处理器(Embedded Micro Processor Unit, Embedded MPU)和数字信号处理器(Digital Signal Processor, DSP)应用最为广泛。虽然它们的灵活性高且设计复杂度低，开发和维护成本低，但在实现复杂在线时间序列预测任务时，对嵌入式处理器的计算速度、功耗和计算延迟提出了极大挑战；近几年，基于 FPGA 的可重构计算已被证明是一种低延迟计算的新兴技术，特别是对于包含大量矩阵和向量运算的在线时间序列预测方法，基于 FPGA 的计算性能不仅可以超越嵌入式微处理器、DSP[5]，相对于 CPU 也可实现更高的计算速度、更低的功耗和计算延迟[6]。目前工业界和学术界针对基于 FPGA 的计算系统主要有两种设计方式：

(1)采用 HDL 代码设计专用的 FPGA 加速引擎：这种方法有许多成功的设计实例，如文献[7]利用不同类型的计算核、不同的并行度来优化性能，采用定点和浮点计算单元的混合设计实现了高性能的 SVM 训练过程；文献[8]中的 FPGA 加速器由若干定点 SVM 计算子模块组成，可同时用于完成 SVM 的训练与预测过程，其计算速度相对于 CPU 提高了 5 倍；Abhinandan 等设计了多核 MAPLE 结构[9]，可以用于加速包括 SVM 在内的大量时间序列预测算法，该计算结构相对于 CPU 获得了高达 1000：1 的加速比。虽然此类设计都实现了很高的计算加速比，但是通用性较差，当目标算法发生变化时，大部分设计甚至是全部设计都需要手动更改，其设计复杂度和设计周期限制了此类设计方式的适用范围。

(2)基于 FPGA 的向量处理器：向量处理器被认为是进行向量计算的最佳平台，并且是学术研究领域的一个热点。如加州大学发起的 VIRAM 项目[10]，将高性能向量处理器和大容量 RAM 集成在单粒芯片内，获得了很高的计算性能。多伦多大学的 VESPA 是一种适合于 FPGA 结构的通用向量处理器[11]，同时兼容向量计算和标量计算，其计算性能虽明显高于 Altera Nios II 和 Xilinx MicroBlaze 软核处理器，但其运行频率只有 131 MHz，与 CPU 相比性能仍差很多；英属哥伦比亚大学的 VIPERS 向量处理器采用了传统 RISC 处理器的 Load/Store 结构[12]，并采用多路并行和多级流水线的设计方式，但其设计仍是一种通用向量处理器，ALU 结构没有进行针对性优化，其计算性能仍无法与前述第(1)类专用加速引擎相比拟。

由此可见，使用 HDL 语言设计的加速引擎实现虽然性能很高，但是开发难度高，开发周期长，通用性和可扩展性差。而已有向量处理器是按照通用处理器的结构进行设计的，仅支持浮点或定点计算之中的一种，无法针对特定算法进行优化，性能难以满足在线时间序列预测的需求。典型 KAF 方法包含大量的向量和矩阵运算，可以通过向量化进行加速，这种特点恰好契合了向量处理器的优势。因此，本文尝试采用基于 FPGA 的向量处理器实现 KAF 方法，实现算法与计算结构之间的匹配，发挥 FPGA 的并行计算优势，以实现高性能的 KAF 在线时间序列预测。

有鉴于此，本文针对在线时间序列预测需求，以 KAF 的 FPGA 加速计算作为实例化研究，主要贡献：(1)提出一种基于 FPGA 的向量处理器计算体系结构，通过多路并行、流水线技术提高了处理器的计算速度；(2)利用微码的强控制能力，通过编写微码程序快速实现多种 KAF 方法，使其通用性、可扩展性提高；(3)算术逻辑单元(ALU)同时支持定点和浮点计算单元，异构设计方式保证数据通路数量同时兼顾算法的计算需求，实现计算速度和 FPGA 资源消耗的平衡。最后，本文通过典型 KAF 方法的向量处理器实现，并与相对于 ARM, PowerPC 和 DSP 性能更高的 Intel 酷睿 CPU 进行对比，来证明该向量处理器在计算速度、功耗以及计算延迟方面的优势。该向量处理器结构扩展能力较强，可为嵌入式环境下的高性能低延迟计算结构研究提供一种新的思路，并对其它复杂计算方法的嵌入式应用具有较强的参考价值。

2 KAF 方法的数学背景

核自适应滤波器(KAF)是线性自适应滤波器在核空间的扩展，应用最为广泛的 KAF 方法包括 KRLS 算法和 KNLMS 算法。KNLMS 是最简单的

核自适应滤波器，计算复杂度最低且具有鲁棒性和自正则化特性。KRLS 算法利用自相关矩阵之逆，对输入样本进行了处理，因此它具有快速收敛能力，但其计算复杂度高于 KNLMS。文献[13]提出一种滑窗核递归最小二乘(Sliding-Window KRLS, SW-KRLS)算法，通过设定窗口宽度来固定样本量和核矩阵维数，即存储量和计算量，因此该方法适合在资源有限的嵌入式平台上实现[14]。文献[15]提出一种固定预算核递归最小二乘(Fixed-Budget KRLS, FB-KRLS)算法，该算法的样本词典内样本量也是固定的，但是按照一种在线裁剪准则来进行判断，以丢弃重要性最不显著的样本。本节主要对两种 KRLS 方法的数学背景作介绍，限于本文篇幅，KNLMS 算法的计算过程请参见文献[16]。

2.1 SW-KRLS 算法

滑窗方法只选取窗口内的样本进行建模，通过设定窗口宽度 N(即样本词典容量为 N)，来限定计算量。在线时间序列预测中，随着样本 $\{(x_1,y_1),(x_2,y_2),\cdots\}$ 的不断增加，滑动窗口只包含最新的 N 个样本 $\boldsymbol{X}_n=[x_n,x_{n-1},\cdots,x_{n-N+1}]$，对应的输出向量为 $\boldsymbol{Y}_n=[y_n,y_{n-1},\cdots,y_{n-N+1}]$，核矩阵为 $\boldsymbol{K}_n=\widetilde{\boldsymbol{X}}_n\widetilde{\boldsymbol{X}}_n^{\mathrm{T}}+c\boldsymbol{I}$。为了更新系数向量 $\boldsymbol{\alpha}_n$，需要计算 $N\times N$ 维的逆矩阵 $\boldsymbol{K}_n^{-1}$，为消除这个计算过程，Van Vaerenbergh 提出一种通过新的输入 $\{\boldsymbol{X}_n,\boldsymbol{Y}_n\}$ 和前一次训练过程中的 $\boldsymbol{K}_{n-1}^{-1}$ 来计算 $\boldsymbol{K}_n^{-1}$ 的方法。已知 $\boldsymbol{K}_{n-1}$，$\boldsymbol{K}_n$ 可以通过式(1)计算：

$$\boldsymbol{K}_n=\begin{bmatrix}\widetilde{\boldsymbol{K}}_{n-1} & \boldsymbol{k}_{n-1}(x_n)\\ \boldsymbol{k}_{n-1}(x_n)^{\mathrm{T}} & k_{nn}+c\end{bmatrix}\tag{1}$$

其中 $\boldsymbol{k}_{n-1}(x_n)=[k(x_{n-N+1},x_n),\cdots,k(x_{n-1},x_n)]^{\mathrm{T}}$，$k_{nn}=k(x_n,x_n)$。$\widetilde{\boldsymbol{K}}_{n-1}$ 可以通过移除 $\boldsymbol{K}_{n-1}$ 的第 1 行和第 1 列获得，公式如下：

$$\boldsymbol{K}_{n-1}=\begin{bmatrix}k_{n-N,n-N}+c & \boldsymbol{p}^{\mathrm{T}}\\ \boldsymbol{p} & \widetilde{\boldsymbol{K}}_{n-1}\end{bmatrix}\tag{2}$$

其中 $\boldsymbol{p}=[k(x_{n-N},x_{n-N+1}),\cdots,k(x_{n-N},x_{n-1})]^{\mathrm{T}}$，$k_{n-N,n-N}=k(x_{n-N},x_{n-N})$。

已知 $\boldsymbol{A}=\widetilde{\boldsymbol{K}}_{n-1}$，$\boldsymbol{b}=\boldsymbol{k}_{n-1}(x_n)$，$d=k_{nn}+c$，$\boldsymbol{K}_n^{-1}$ 可以通过式(3)计算：

$$\boldsymbol{K}_n^{-1}=\begin{bmatrix}\boldsymbol{A}^{-1}(\boldsymbol{I}+\boldsymbol{b}\boldsymbol{b}^{\mathrm{T}}\boldsymbol{A}^{-1\mathrm{H}}\boldsymbol{g}) & -\boldsymbol{A}^{-1}\boldsymbol{b}\boldsymbol{g}\\ -(\boldsymbol{A}^{-1}\boldsymbol{b})^{\mathrm{T}}\boldsymbol{g} & \boldsymbol{g}\end{bmatrix}\tag{3}$$

其中 $\boldsymbol{g}=(d-\boldsymbol{b}^{\mathrm{T}}\boldsymbol{A}^{-1}\boldsymbol{b})^{-1}$。

以上计算过程向字典添加了一行和一列。在式(3)中，首先需要计算 $\widetilde{\boldsymbol{K}}_{n-1}^{-1}$。根据 $\widetilde{\boldsymbol{K}}_{n-1}$ 的定义，这个计算过程是从字典中移除一行和一列。

设定 $\boldsymbol{K}_{n-1}^{-1}=\begin{bmatrix}e & \boldsymbol{f}^{\mathrm{T}}\\ \boldsymbol{f} & \boldsymbol{G}\end{bmatrix}$，其中 e 是一个标量，$\boldsymbol{f}$ 是一个 $N\times1$ 的向量，$\boldsymbol{G}$ 是一个 $(N-1)\times(N-1)$ 维的矩阵。那么 $\widetilde{\boldsymbol{K}}_{n-1}^{-1}$ 可以通过式(4)计算：

$$\widetilde{\boldsymbol{K}}_{n-1}^{-1}=\boldsymbol{G}-\frac{\boldsymbol{f}\boldsymbol{f}^{\mathrm{T}}}{e}\tag{4}$$

综上，SW-KRLS 算法的伪代码如表 1 所示。

表 1　SW-KRLS 算法伪代码

Initializing $\boldsymbol{K}_0=(1+c)\boldsymbol{I}$ and $\boldsymbol{K}_0^{-1}=\boldsymbol{I}/(1+c)$.
for n=1, 2, $\cdots$ **do**
　Getting $\widetilde{\boldsymbol{K}}_{n-1}$ from $\boldsymbol{K}_{n-1}$ with Eq.(2)
　Calculating $\boldsymbol{K}_{n-1}^{-1}$ with Eq.(4)
　Getting $\boldsymbol{K}_n$ with Eq.(1)
　Calculating $\boldsymbol{K}_n^{-1}$ with Eq.(3)
　Getting the updated solution $\boldsymbol{\alpha}_n=\boldsymbol{K}_n^{-1}\boldsymbol{Y}_n$
end for

SW-KRLS 算法的每次训练过程都会更新核逆矩阵，其计算复杂度为 $O(N^2)$。

2.2 FB-KRLS 算法

FB-KRLS 算法的计算过程与 SW-KRLS 算法类似，样本词典容量 N 也是固定的，从而核矩阵维数也是固定的。FB-KRLS 算法与 SW-KRLS 算法的区别在于，FB-KRLS 并不直接丢弃样本词典内最旧的样本，而是通过一种裁剪准则，在样本词典中找出重要性最不显著的样本进行丢弃。利用文献[17]中描述的裁剪准则计算出丢弃后导致最小误差的样本，对于样本 (x_i,y_i) 其相关性表达式为

$$d(x_i,y_i)=\left|\alpha_i\right|\Big/\left[\boldsymbol{K}_N\right]_{i,i}\tag{5}$$

其中，α_i 表示系数向量第 i 个元素，$[\boldsymbol{K}_N]_{i,i}$ 表示 N 维核逆矩阵的第 i 个对角元素，这里 $\boldsymbol{K}_N$ 为正定方阵。它们都是 KRLS 算法在线更新过程中的迭代变量，故可直接对式(5)进行计算。固定预算方法的计算复杂度也是 $O(N^2)$。

3　KAF 向量处理器结构设计分析

向量处理器(vector processor)研究开始于 20 世纪 60 年代末，最初应用于超级计算领域。第一台商业化的向量计算机是 Cray-1，该向量计算机的运行主频为 80 MHz，共有 16 路 64 位字长运算单元，其峰值计算能力 250 MFLOPs，是 20 世纪 70 年代以前全球最快的计算机，而最新的 NEC-SX9 向量处理器则是当今性能最高的超级计算机之一。向量处理器具有多条并行的数据通路，专用于向量运算，向量处理器的每一条指令都在并行执行几十个甚至上百个标量操作，等效于标量处理器多次循环执行一条指令。对于典型的 KAF 方法，例如 SW-KRLS,

FB-KRLS 和 KNLMS 算法，其计算过程包含大量的向量和矩阵运算，这种特点恰好契合了向量处理器的优势。因此，采用向量处理器实现 KAF 算法能够保证计算的高性能。

在计算机体系结构中，一些功能复杂的指令会被分解为一系列相对简单的指令来执行，这样一系列的简单指令就叫做微码(microcode)。微码的概念最早出现在 1947 年，微码指令是比汇编指令更底层的指令，更加接近机器指令，通常由 CPU 工程师在设计阶段编写。本文采用的结构是类似文献[18]的微码处理器结构，微码指令的操作数和结果都是向量。相对于 C/C++和其它高级语言来说，虽然微码编程相对困难，但是其执行效率高，控制能力强，可以实现指令层次的设计优化，因此，本文采用了基于微码的编程方式。

在数据表示方面，虽然多数科学计算系统都使用符合 IEEE-754 标准的浮点数据形式，特别是存在病态数据的时候，浮点计算具有更强的鲁棒性。然而定点计算可以大大缩减指令的执行周期，并且定点运算单元消耗的 FPGA 资源更少，从而可以在同等规模的 FPGA 内实现更大的并行度，进一步提高处理器的性能。实际时间序列预测应用中，算法的计算精度与使用的测试数据集存在依赖性，文献[19]已经证明在已知计算方法和测试数据集的前提下，可以通过蒙特卡罗仿真方法分析出为实现预设的计算精度，所需要的定点数的数据宽度，以通过足够的数据位宽实现与浮点计算方式相同的计算精度。因此，本文采用了精度可配置的算术逻辑单元，同时支持定点计算和浮点计算，在精度满足要求的情况下，使用定点 ALU 可节约 FPGA 资源并且实现更高的运行频率，获得相对浮点向量处理器更高的计算速度和更低的计算延迟。

本文将向量处理器结构、微码以及可配置(定点/浮点)的 ALU 相结合，实现了处理器架构和指令优化，通过充分挖掘 KAF 方法的特点，通过微码编程实现了高性能、低功耗低延迟的 KAF 计算系统，对其它在线机器学习方法的嵌入式加速计算实现具有很高的参考价值。

3.1 KAF 向量处理器结构

针对嵌入式在线时间序列预测需求，在满足计算性能需求的条件下，需要同时降低 FPGA 资源消耗和能量消耗，因此设计一种简洁、优化、易于实现流水线设计，且具有高度扩展性的向量处理器结构是本项研究的目标。该向量处理器不包含 SRAM, DRAM 等外部存储器接口，训练数据和测试数据均被预先存储在 FPGA 的片上 RAM。对于实际应用，可以采用 USB 总线、PCI 总线等标准接口逻辑与本处理器耦合即可，由于外设接口不是本文研究的重点，在此不做过多讨论。

如图 1 所示，向量处理器由程序计数器(PC)、微码存储器(microcode memory)、向量存储器(vector memory)、算术逻辑单元(ALU)4 部分组成。向量处理器的工作原理如下：(1)向量处理器顺序执行指令时，程序计数器的计数值在每个机器周期结束时自动加 1；当程序运行至分支指令(BRANCH)时，目标跳转地址将被加载到程序计数器。程序计数器的计数值被直接用作微码存储器的地址输入；(2)用于完成特定算法的微码程序全部预先存储于微码存储器，微码指令在程序计数器的控制下被顺序取出并执行；(3)在微码指令的控制下，存储于向量存储器特定地址的向量数据被取出，并输出至 ALU 进行运算；(4)ALU 在微码指令的控制下，完

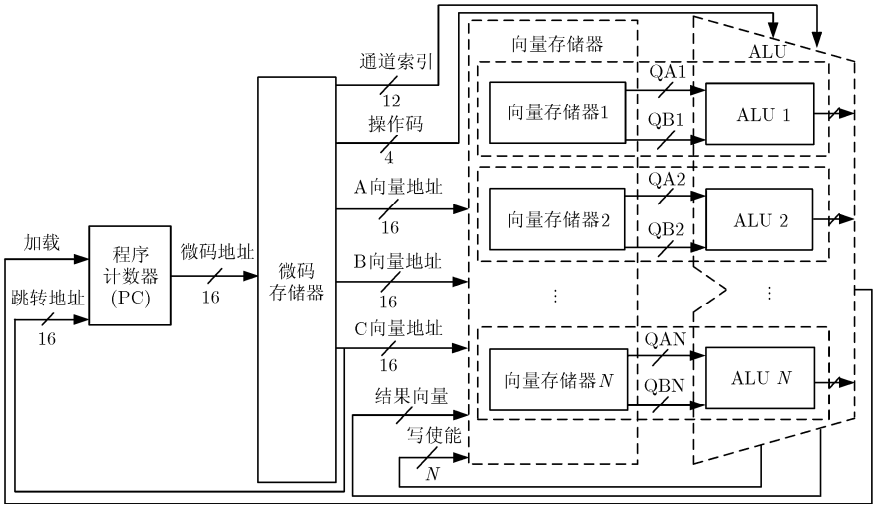


图 1 向量处理器结构框图

成所有向量运算，并将计算结果传输至数据控制单元；(5)ALU 同时将根据不同的微码指令，产生向量存储器写使能(WR)信号，完成向量存储器的存储控制；ALU 的异构计算单元完成指数函数(EXP)、除法(DIV)和开方运算(SQRT)等功能。

每一个向量存储器与一个 ALU 构成一条数据通路，二者直接相连，并具有相同的数据位宽(M)，本文设计的向量处理器的数据通路数量可以根据特定的计算需要灵活设定，为了降低硬件设计的复杂度，向量处理器的通路数量目前被限制为 2 的指数次幂$(16, 32, 64, 128)$，但实际按照本文的设计方式，可以实现 1～128 路任意数据通路数量的向量处理器，本文第 4 节将详细分析不同通路数量的向量处理器的计算速度提升和计算功耗差异。

3.2 微码存储器和向量存储器接口

传统 RISC 处理器一般采用寄存器文件来存储中间数据，寄存器文件一般具有十几个可以同时读写的端口，以实现很高的计算性能。由于 FPGA 中没有可用的多端口(端口数量大于 2)存储器，本文采用 FPGA 的片上双端口 RAM 作为向量存储器。

每个数据通路中包含两个相同的双端口 RAM 作为向量存储器，其读地址端口分别由微码指令(表 2)中的“A”和“B”两个码段所驱动，写地址端口均由“C”驱动。数据输出端口“QA”，“QB”直接与 ALU 输入耦合。“WR”信号为每个向量存储器的写使能信号，由 ALU 产生，用于完成向量数据的存储控制。微码存储器被设定工作于 ROM 模式，上电加载预先存储的微码程序，因此微码存储器不能够像向量存储器一样可以被在线改写。

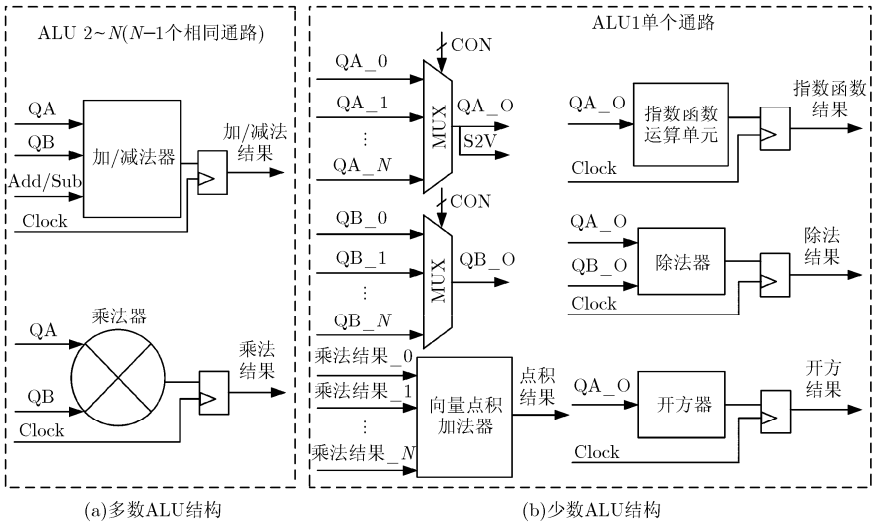
向量存储器的存储深度为 2048，存储数据宽度(M)可以根据计算数据位宽灵活设定，在编译前通过宏变量以参数的形式进行定义即可。向量存储器的地址空间被划分为 3 段：其中 0x0000～0x0BFF 用于存储训练和在线测试数据，该区域数据是在编译时预先设定的；0x0C00～0x0FEF 用于存储中间计算结果；最后的 16 个字 0x0FF0～0x0FFF 则用来存储向量计算过程中需要使用的常量。

3.3 异构算术逻辑单元

对于常见 KAF 方法，计算过程中需要用到少量的除法运算、指数运算以及开方运算，这些运算单元使用频率很低，却需要消耗大量 FPGA 片上 DSP 资源。本文作者前期的实验证明，如果在每个 ALU 中包含除法和指数运算模块，向量处理器的最大通路数量仅为 17，将严重影响向量处理器的性能。因此，本文采用了异构的算术逻辑单元设计，在不影响总体性能的前提下，保证了数据通路数量，平衡计算性能和 FPGA 资源消耗。

如图 2 所示，在该异构设计中共有两种算术逻辑单元：多数 ALU$(2\sim N)$，共 $N-1$ 个通路具有相同的结构，每个 ALU 中包含一个定点加/减法器和一个单精度乘法器，只支持定点加法、减法和乘法运算；少数 ALU1，除包含一个定点加/减法器和一个单精度乘法器外，还包含定点指数运算单元、定点除法器以及定点开方运算单元。虽然只有 ALU1 中包含除法、指数运算以及开方运算单元，向量处理器可通过类似标量计算机的循环执行方式实现向量运算。图 2(b)中“MUX”为 N 输入多路选择器。

本文所采用的异构算术逻辑单元设计方式，使得该向量处理器具有很强的指令扩展能力，通过修改 ALU1 的结构，可以添加多种计算单元，如三角函数 Tan/Atan 和对数函数 Log 等。此种设计方式在原理上可以实现任何标准运算以及用户自定义运算，实现了向量指令集的扩展。



(a)多数ALU结构　　(b)少数ALU结构

图 2 向量处理器算术逻辑单元结构图

3.4 KAF 向量处理器微码指令结构

微码的长度需要根据码段的数量以及每个码段的长度进行确定，其中地址段的长度取决于需要访问的地址空间的大小。例如，如果处理器需要访问 4GB 地址范围，则地址宽度应为 32 位。微码指令格式如表 2 所示，“A”，“B”分别为输入向量地址，“C”表示结果存储的目标地址，“A”，“B”和“C”3 个码段的宽度均设定为 16 位，可以访问 64 kB 的地址空间，虽然目前没有用到全部地址空间，此设计可为后续更大存储深度的访问留有充足的余量；通路索引“L”是一个 12 位宽的码段，用于指定“PVDOT”指令结果(该结果为标量)存储在哪一路向量存储器；“OP”为操作码，指示指令类型，目前向量处理器只有 13 种指令，“OP”的宽度设定为 4。因此，微码总宽度为 64 位。

如表 3 所示，目前该向量处理器共有 13 条微码指令，该表给出了每条指令的具体功能和执行机器周期数。为了进一步提升向量处理器的运行频率和指令执行效率，我们设计了全流水线(fully pipelined)的数据通路。除 VNOP 和 BRANCH 两条指令之外的 11 条指令均可以在相同的周期内完成，全流水线设计使得指令的执行(execution)过程与数据写回(write back)过程重叠，每个机器周期都可以发起一条新的指令，指令执行效率更高。对于浮点向量处理器，乘法器的计算延迟为 6 个周期，二输入浮点加法器的计算延迟为 9 个周期，浮点 VDOT 指令使用乘法器和 $\log_2(N)$级二输入加法器完成内积运算，因此 VDOT 的执行过程共需要 $6+9\log_2(N)$个周期。另外，从微码存储器中取出指令和从向量存储器中取出数据分别需要一个机器周期，因此浮点向量处理器指令的总周期数为 $8+9\log_2(N)$。而定点向量处理的所有运算均可以在 2 个周期内完成，因此定点向量指令的总周期数为 4，定点向量处理器数据通路的设计如图 3 所示。

4 实验设计与结果分析

本节将描述具有不同通路数量的向量处理器的 FPGA 资源消耗、计算速度、功耗以及计算延迟的

表 2 微码格式

码段	A 向量地址	B 向量地址	C 向量地址	通路索引	操作码
符号	A	B	C	L	OP
位宽	16	16	16	12	4

表 3 微码程序指令列表

微码指令(OP)	指令功能	机器周期	
		浮点	定点
VNOP (0000)	空指令	1	1
BRANCH(0111)	分支指令	4	4
VADD(0001)	向量加法	$8+9\log_2(N)$	4
VSUB(0010)	向量减法	$8+9\log_2(N)$	4
VMUL(0011)	向量对应元素相乘	$8+9\log_2(N)$	4
SDIV(0100)	向量对应元素相除	$8+9\log_2(N)$	4
SEXP(0110)	向量元素的指数函数	$8+9\log_2(N)$	4
SSQRT(1100)	向量元素的开平方	$8+9\log_2(N)$	4
S2VE(1000)	将一个标量扩展为向量	$8+9\log_2(N)$	4
VABS(1001)	向量元素的绝对值	$8+9\log_2(N)$	4
VCOPY(1010)	复制向量内所有元素	$8+9\log_2(N)$	4
SCOPY(1011)	复制向量内指定元素	$8+9\log_2(N)$	4
VDOT(0101)	向量点积	$8+9\log_2(N)$	4

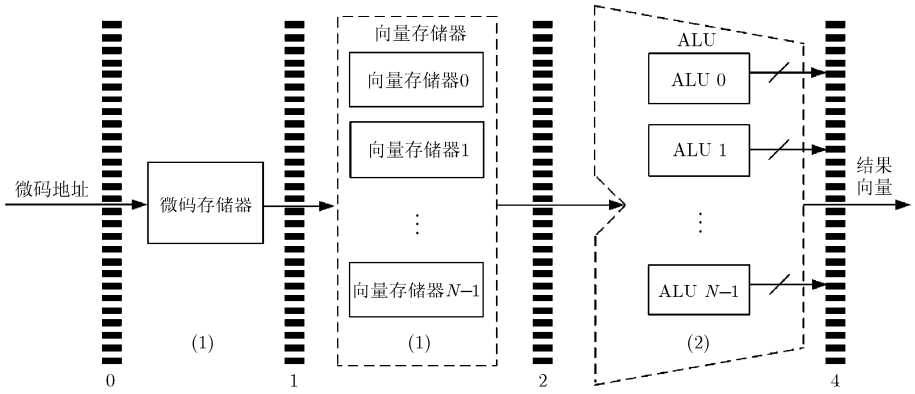


图3 全流水线定点数据通路设计

测试结果，并与 CPU 的相应性能进行对比。

本文使用硬件描述语言(Hardware Description Language, HDL)语言完成了向量处理器的设计工作，采用 Altera 提供的 LPM 运算 IP 库实现了算术逻辑单元的设计，并使用 Quartus II 15.0 工具进行了综合和编译，在基于 Stratix V FPGA 的 DE5 板卡上进行了实际硬件测试。值得说明的是，虽然本文实验都是基于 Stratix V FPGA 进行，但该向量处理器的设计可以很容易地移植到其它 FPGA 器件。

实验通过编写不同的微码程序，未对处理器结构进行修改，就实现了 KAF 中最为经典的 SW-KRLS, FB-KRLS 以及 KNLMS 3 种方法，并在向量处理器中进行了性能分析。同时，使用时间序列预测中最常用的公开数据集 Mackey-Glass(MG-30)数据作为基准数据进行训练和测试，其中，两种 KRLS 方法选取的正则化参数为 $C = 0.01$，内核参数 $\sigma = 0.6$。通道数量 N=16, 32, 64 和 128 分别对应字典大小 M=15, 31, 63 和 127。本文实验将硬件计算结果与 KAF 工具箱 KAFBOX[20]的双精度计算结果进行对比，根据我们的应用要求，设定定点计算结果与浮点计算结果的均方差(MSE)要小于 0.3%，定点计算数据位宽为 19 位。

4.1 向量处理器 FPGA 资源消耗

本文使用 Altera DE5 开发板进行原型验证，FPGA 芯片是 Stratix V 5SGXEA7N2F45，速率等级为 C2，定点和单精度浮点向量处理器的 FPGA 资源消耗情况表 4 所示。从表 4 可以看出，定点处理器相对于浮点处理器消耗更少的 FPGA 资源。定点向量处理器的 DSP 乘法器使用量相对于浮点向量处理器低 1/3~1/2，其原因是：对于单精度浮点处理器，指数函数运算单元需要使用 9 个 DSP 硬件乘法器，除法器需要 5 个 DSP 硬件乘法器，乘法器需要使用 2 个 18 位 DSP 硬件乘法器；而对于定点处理器，指数函数运算单元仅需要使用 6 个 DSP 硬件乘法器，除法模块仅需要 3 个 DSP 硬件乘法器，乘法器模块仅需使用 1 个 18 位 DSP 硬件乘法器。因此，对于同等规模的 FPGA 芯片，定点向量处理器可以实现更大的通道数量，进而实现更高的计算性能；同时，定点向量处理器因具有更小的数据位宽，片上 M20K RAM 和 ALMs(Adaptive Logic Modules)资源消耗更少，仅为与浮点向量处理器的 1/2~2/3。另外，表 4 同时给出了不同通路数量的向量处理器的最高运行频率 Fmax，在通道数量相同的情况下，定点向量处理器的 Fmax 相对于浮点向量处理器的 Fmax 高出 7~126 MHz。

4.2 向量处理器计算速度

向量处理器的计算速度是最重要的评价指标，我们分别对通路数量 N=16, 32, 64 和 128 四种向量处理器进行了测试。向量处理器均工作在表 4 所示的最高系统频率 Fmax。由于向量处理器的计算速度完全可与通用 CPU 相比拟，故本文直接采用相对于 ARM, DSP 等嵌入式处理器性能更高的 CPU 作对比平台，用于对比实验的 CPU 平台的具体配置如下：英特尔 i5-2400 4 核 CPU，主频为 3.10 GHz，4 GB 系统内存，操作系统为 Ubuntu 12.10 LTS, GCC 编译器的版本为 4.7.2，使用 ATLAS 库针对多核 CPU 进行了 C 程序优化。

如表 5 所示，当通道数量小于 32 时，定点向量处理器的性能没有超越 CPU，主要原因是定点向量处理器的运行频率远低于 CPU 的运行频率，而定点处理器的通道数量较小，没有获得足够的计算并行度；而当通道数量高于 32 时，定点向量处理器实现相对于 CPU, KRLS 方法获得了 2.4~22.4 倍的加速比，而 KNLMS 仅获得了 1.4~3.9 倍的加速比，但两种方法都随着通道数量的增多加速比增大，这是因为通道数量的增多直接提高了向量处理器的计算并行度。在 KAF 算法中，向量运算和标量运算组成全部运算，向量运算所占的比例越高，向量处理器的并行计算优势发挥越明显，获得的加速效果也就

表 4 向量处理器资源消耗情况

资源		M20KAM	18bit×18bitDSP 乘法器	ALMs	Fmax (MHz)
	总量	2560	512	234720	–
VP@16(%)	浮点	10	12	11	253
	定点	6	8	7	379
VP@32(%)	浮点	19	18	20	265
	定点	12	11	13	345
VP@64(%)	浮点	39	30	42	275
	定点	23	17	29	312
VP@128(%)	浮点	84	55	88	284
	定点	43	29	47	291

表 5 性能测试

性能	SW-KRLS				FB-KRLS				KNLMS			
	16	32	64	128	16	32	64	128	16	32	64	128
浮点向量处理器运行时间(μs)	5.2	8.0	10.8	16.9	5.7	8.8	11.7	18.0	3.7	4.8	6.1	8.1
浮点计算能力(GFLOPS)	1.1	2.8	8.2	20.6	1.1	2.8	8.1	21.0	0.4	0.6	0.9	1.2
定点向量处理器运行时间(μs)	2.1	2.8	3.4	4.7	2.4	3.0	3.7	4.8	0.9	1.0	1.1	1.2
CPU 运行时间(μs)	1.6	6.9	21.1	101.5	2.2	8.7	23.4	107.5	0.8	1.4	2.6	4.7
加速比(定点/浮点)	2.5	2.9	3.2	3.6	2.4	2.9	3.2	3.8	4.1	4.8	5.5	6.8
加速比(定点/CPU)	0.8	2.5	6.2	21.6	0.9	2.9	6.3	22.4	0.9	1.4	2.4	3.9

越高。由于 KNLMS 算法中向量运算的比例相对于其他两种 KRLS 算法较低，因此 KNLMS 算法获得的计算加速比也相对较低。本文使用 3 种典型 KAF 方法进行实验，是为了验证该向量处理器对多种 KAF 方法的通用性。

定点向量处理器相对于浮点向量处理器具有明显的计算速度优势，主要原因有二：第一，定点向量处理器的单指令执行周期更小，第二，如表 4 所示，相同数据通路数量的定点向量处理器可以运行于更高的系统频率。定点向量处理器数据通路数量达到 128 时，获得最高计算加速比，相对于 CPU 的计算加速比高达 22，大幅提升了计算速度。

4.3 功耗分析

本节以 SW-KRLS 为例进行向量处理器和通用处理器 CPU 的计算功耗分析。本文使用安捷伦 U8001A 电源测量向量处理器的功耗，CPU 的功耗数据则来源于操作系统提供的电量消耗报告。用于对照实验的 CPU 平台的具体配置情况与 4.2 节相同。从表 6 可以看出，定点向量处理器的能耗最低，当通路数量 N=128 时，其能量消耗仅为浮点向量处理器的 1/7，仅为通用 CPU 的 1/139，特别适用于低功耗的嵌入式应用。

表 6 向量处理器与 CPU 功耗对比分析

处理器	通路数量	功率 (mW)	运行时间 (μs)	能耗 (1×10^{-5} J)
浮点处理器	16	17580	5	9
	32	19120	8	15
	64	22390	11	25
	128	26880	17	46
定点处理器	16	8180	2	2
	32	10390	3	3
	64	13470	4	5
	128	14760	5	7
CPU	16	95576	2	19
	32		7	67
	64		22	210
	128		102	975

4.4 计算延迟测试

这一小节详细分析向量处理器和 CPU 两种计算平台的系统计算延迟，延迟越低系统响应速度越快。如图 4 所示，实验系统由 3 部分组成，模数转换器 ADC 作为数据输入设备，向量处理器或 CPU 作为数据处理器，数模转换器 DAC 作为数据输出设备。因此计算系统延迟由 3 部分组成：即输入延迟 L_{i}，即输入数据由输入设备传输至 CPU 的延迟；计算延迟 L_{c}，即 KAF 方法在处理器上的运行时间；输出延迟 L_{o}，即输出数据经输出设备传输至外设的延迟；系统总延迟 $L_{\mathrm{a}}=L_{\mathrm{i}}+L_{\mathrm{c}}+L_{\mathrm{o}}$。

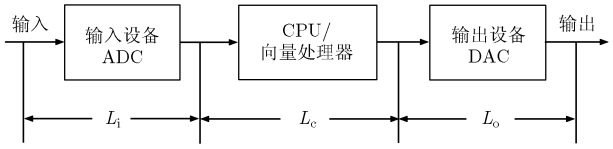


图4 系统计算延迟框图

两种计算平台的具体实验配置如下：(1)向量处理器工作于 Altera DE5 开发平台，向量处理器通路数量为 64，输入输出设备采用的是 Terasic 公司 AD/DA 子板，该子板使用的是 AD9248 和 AD9767 芯片，为了实现最小的输入输出延迟，ADC 和 DAC 的工作频率均被设定在最高频率，分别是 62.5 MHz, 125 MHz，其输入延迟为 L_{i}=0.12 μs，输出延迟为 L_{o}=0.04 μs。(2)CPU 平台采用的是美国 NI 公司的 NI5781 基带收发器模块和 NI7954 FPGA FlexRIO 模块，NI5781 被设定工作在最高采样频率(100 M Sample/s)以降低数据传输延迟，并使用安捷伦示波器精确测量输入数据和输出数据的延迟。

通道数 N=64 时的计算延迟测试结果如表 7 所示，CPU 平台的数据输入和输出都要经过 PCI 总线进行传输，其输入输出延迟 $L_{\mathrm{i}}=L_{\mathrm{o}}$=7.32 μs 均较大，其计算延迟为 L_{c}= 78.50 μs 也相对较大，因而具有很高的系统总延迟 L_{c}=93.14 μs；定点向量处理器的计算延迟最小，其输入输出延迟也小于 CPU 平台，因此具有最小的系统总延迟 L_{c}=3.56 μs，仅为 CPU 的 1/26，特别适合用于要求低延迟快速响应的应用。

表 7 多处理器平台系统延迟(μs)

处理器	输入延迟	计算延迟	输出延迟	总延迟
浮点 VP@64	0.12	10.8	0.04	10.96
定点 VP@64	0.12	3.4	0.04	3.56
CPU	7.32	78.50	7.32	93.14

5 结束语

本文提出了一种基于FPGA的向量加速计算结构，为嵌入式的高性能在线时间序列预测研究提供一种新颖的向量处理器设计和实现思路。通过微码编程解决了传统FPGA计算系统设计方法的通用性和可扩展性差的问题；提出定点/浮点可配置的异构ALU 实现了计算速度和 FPGA 资源消耗的平衡；最后以典型KAF方法的向量处理器实现为例，证明该向量处理器在满足精度要求的条件下，相对于通用处理器CPU在资源消耗、计算速度、功耗以及计算延迟方面均获得明显提升。本文提出的向量处理器结构，对于其它嵌入式环境下的高性能低延迟可重构计算系统设计同样具有较强的参考价值。

虽然微码编程具有较强的控制能力，但编程效率较低，因此，后续研究工作包括微码程序仿真器、基于GCC的C/C++语言编译器等工具的开发，这些工具将使该向量处理器更加易于使用，提高算法开发和程序调试的效率。同时，将进一步优化向量处理器的结构和性能，并将其应用到其它在线时间序列预测方法中。

参 考 文 献

[1] 王中杰, 谢璐璐. 信息物理融合系统研究综述[J]. 自动化学报, 2011, 37(10): 1157–1166.
WANG Zhongjie and XIE Lulu. Cyber-physical system: a survey[J]. *Acta Automatica Sinica*, 2011, 37(10): 1157–1166.

[2] 周建宝, 王少军, 马丽萍, 等. 可重构卫星锂离子电池剩余寿命预测系统研究[J]. 仪器仪表学报, 2013, 34(9): 2034–2044.
ZHOU Jianbao, WANG Shaojun, MA Li-ping, *et al*. Study on the reconfigurable remaining useful life estimation system for satellite lithium-ion battery[J]. *Chinese Journal of Scientific Instrument*, 2013, 34(9): 2034–2044.

[3] 王少军. 时间序列预测的可重构计算研究[D]. [博士论文], 哈尔滨工业大学, 2012.
WANG Shaojun. Research on reconfigurable computing for time series forecasting[D]. [Ph.D. dissertation], Harbin Institute of Technology, 2012.

[4] 曹葵康. 支持向量机加速方法及应用研究[D]. [博士论文], 浙江大学, 2010.
CAO Kuikang. Acceleration and application of support vector machines[D]. [Ph.D. dissertation], Zhejiang University, 2010.

[5] 江洁, 凌思睿. 一种投票式并行RANSAC算法及其FPGA实现[J]. 电子与信息学报, 2014, 36(5): 1145–1150. doi: 10.3724/SP.J.1146.2013.00962.
JIANG Jie and LING Sirui. Parallel voting RANSAC and its implementation on FPGA[J]. *Journal of Electronics & Information Technology*, 2014, 36(5): 1145–1150. doi: 10.3724/SP.J.1146.2013.00962.

[6] 兰亚柱, 杨海钢, 林郁. 动态自适应低密度奇偶校验码译码器的FPGA实现[J]. 电子与信息学报, 2015, 37(8): 1937–1943. doi: 10.11999/JEIT141609.
LAN Yazhu, YANG Haigang, and LIN Yu. Design of dynamic adaptive LDPC decoder based on FPGA[J]. *Journal of Electronics & Information Technology*, 2015, 37(8): 1937–1943. doi: 10.11999/JEIT141609.

[7] PAPADONIKOLAKIS M. A scalable FPGA architecture for non-linear svm training[C]. Proceeding of 2008 International Conference on Field Programmable Technology, Taipei, China, 2008: 337–340.

[8] ANGUITA D, CARLINO L, GHIO A, *et al*. A FPGA core generator for embedded classification systems[J]. *Journal of Circuits, Systems and Computers*, 2011, 20(2): 263–282.

[9] MAJUMDAR A, CADAMBI S, BECCHI M, *et al*. A massively parallel, energy efficient programmable accelerator for learning and classification[J]. *ACM Transactions on Architecture and Code Optimization*, 2012, 9(1): 6:1–6:30.

[10] KOZYRAKIS C and PATTERSON D. Vector vs. superscalar and vliw architectures for embedded multimedia benchmarks[C]. Proceeding of 35th Annual IEEE/ACM International Symposium on Microarchitecture, Califonia, USA, 2002: 283–293.

[11] YIANNACOURAS P, STEFFAN J G, and ROSE J. Portable, flexible, and scalable soft vector processors[J]. *IEEE Transactions on Very Large Scale Integration Systems*, 2012, 20(8): 1429–1442.

[12] YU J, EAGLESTON C, CHOU C H Y, *et al*. Vector processing as a soft processor accelerator[J]. *ACM Transactions on Reconfigurable Technology and Systems*, 2009, 2(2): 12:1–12:34.

[13] VAN VAERENBERGH S, VIA J, and SANTAMARIA I. A sliding-window kernel RLS algorithm and its application to nonlinear channel identification[C]. 2006 IEEE International Conference on Acoustics, Speech and Signal Processing, Toulouse, France, 2006: 789–792.

[14] PANG Yeyong, WANG Shaojun, PENG Yu, *et al*. A low latency kernel recursive least squares processor using FPGA

technology[C]. 2013 International Conference on Field-Programmable Technology (FPT), Kyoto, Japan, 2013: 144–151.

[15] VAN VAERENBERGH S, SANTAMARIA I, WEIFENG L, *et al*. Fixed- budget kernel recursive least-squares[C]. 2010 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), Dalas, USA, 2010: 1882–1885.

[16] RICHARD C, BERMUDEZ J C M, and HONEINE P. Online prediction of time series data with kernels[J]. *IEEE Transactions on Signal Processing*, 2009, 57(3): 1058–1067.

[17] KRUIF B J and VRIES T J A. Pruning error minimization in least squares support vector machines[J]. *IEEE Transactions on Neural Networks*, 2003, 14(3): 696–702.

[18] LEONG P H W and LEUNG I K H. A microcoded elliptic curve processor using FPGA technology[J]. *IEEE Transactions on VLSI Systems*, 2002, 10(5): 550–559.

[19] CHAU T C P, KUREK M, TARGETT J S, *et al*. SMCGen: Generating reconfigurable design for sequential Monte Carlo applications[C]. Proceeding of 22nd IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM), Boston, USA, 2014: 141–148.

[20] VAN VAERENBERGH S and SANTAMARIA I. A comparative study of kernel adaptive filtering algorithms[C]. Proceedings of 2013 IEEE Digital Signal Processing and Signal Processing Education Meeting, California, USA, 2013: 181–186.

庞业勇： 男，1985 年生，博士生，研究方向为可重构计算.

王少军： 男，1982 年生，讲师，研究方向为时间序列分析与预测技术、可重构计算等.

彭 宇： 男，1973 年生，教授，博士生导师，研究方向为测试诊断技术、无线传感器网络技术和数据挖掘技术等.

彭喜元： 男，1961年生，教授，博士生导师，研究方向为计算智能、模式识别、故障诊断技术等.