Control Engineering Practice 60 (2017) 124–132

Contents lists available at ScienceDirect

Control Engineering Practice

journal homepage: www.elsevier.com/locate/conengprac

Efficient hardware implementation of radial basis function neural network with customized-precision floating-point operations

Helon Vicente Hultmann Ayala[a,*], Daniel M. Muñoz[b], Carlos H. Llanos[c], Leandro dos Santos Coelho[a,d]

[a] Industrial and Systems Engineering Graduate Program, Pontifical Catholic University of Paraná, Brazil
[b] Electronics Engineering Graduate Program, Faculty of Gama, University of Brasilia, Brazil
[c] Department of Mechanical Engineering, University of Brasilia, Brazil
[d] Department of Electrical Engineering, Federal University of Paraná, Brazil

ARTICLE INFO

Keywords:
Radial basis functions neural networks
FPGA
Floating-point
Nonlinear systems
Embedded systems
System identification

ABSTRACT

This paper aims at the proposition of novel architectures for radial basis function neural networks implementation on hardware with custom-precision floating-point operations for black-box system modeling. An analysis tool was built to establish the trade-off between the consumption of hardware resources and the precision of the outputs, on the basis of the usage of the logic blocks on a field-programmable gate array and output quality. The architectures have been tested with a standard system identification benchmark and the speedup factors, when compared to a C implementation, are on the order of hundreds, what shows the importance of ad-hoc hardware architectures for improving computational efficiency.

1. Introduction

The Radial Basis Functions Neural Network (RBFNN) is a kind of Artificial Neural Network (ANN), which has been proposed in the late 1980s (Broomhead & Lowe, 1988). It has characteristics which distinguish it among other ANNs. The RBFNN is composed of three layers (input, hidden and output layers), the activation of the neurons are defined with Radial Basis Functions (RBFs) and the input layer is connected directly with the hidden layer. The fact that the RBFNN has the universal approximation property (Park & Sandberg, 1991) and one may employ simple strategies for learning such as *k*-means clustering (Leonard & Kramer, 1991) as well as orthogonal least squares (Chen, Cowan, & Grant, 1991), has motivated its use ever since.

Due to its wide scope of applications and aiming at real-time applications, ANNs have been implemented in reconfigurable hardware such as Field-programmable Gate Arrays (FPGAs) since the 1990s (Botros & Abdul-Aziz, 1994). In this work, the authors test in two Xilinx XC3042 FPGAs an ANN trained off-line with 5 inputs, 4 neurons and 2 outputs for a classification task. For a recent review on the topic of ANNs implementations on FPGAs, the reader is referred to (Bosque, del Campo, & Echanobe, 2014). Other sources are previous reviews (Blake et al., 1998; Zhu & Sutton, 2003) and an organized book (Omondi & Rajapakse, 2006). With respect to neuro-fuzzy systems, Echanobe, del Campo, and Bosque (2008) implement a two-input one-output special case of the adaptive-network-based fuzzy inference system in Altera Stratix II EP2S15 FPGA.

Otherwise, the implementations of RBFNNs on hardware are not numerous in the literature as we summarize below the works related with this topic. Chen, Tsai, Lin, and Lee (2005) proposed an offline trained RBFNN for the purpose of nonlinear channel equalization to a Xilinx Virtex-2 FPGA, where the exponential function was approximated by a 4th order Taylor expansion. A scheme for RBFNN evaluation and online training based on fuzzy *c*-means and recursive least mean square algorithms are implemented on an FPGA in (Fan and Hwang 2013). In this work, the authors use benchmarks for classification problems to evaluate the implementation on an Altera Cyclone III EP3C120 FPGA and calculate the exponent function with a floating-point Altera megafunction. An FPGA architecture for RBFNN is investigated in (Chou, Kung, Quynh, and Cheng 2013) when applied to uncertainty detection for online controller tuning of a permanent-magnet synchronous motor, and tested through co-simulation using Simulink© and ModelSim© tools. The exponential function is calculated according to a 12th order Taylor expansion and the data types are set in the Q15 format with 16 bits and two complements. In (Souza & Fernandes, 2014), the authors propose the online training of RBFNNs through the least mean squares algorithm, which is tested with the XOR problem and sine function approximation in a Xilinx Virtex-6

* Correspondence to: Pontifical Catholic University of Paraná, Imaculada Conceicao, 1155, 80215-901 Curitiba, Paraná, Brazil.
E-mail address: helonayala@gmail.com (H.V.H. Ayala).

http://dx.doi.org/10.1016/j.conengprac.2016.12.004
Received 27 September 2015; Received in revised form 8 December 2016; Accepted 9 December 2016

FPGA. In this work, the authors analyse, for fixed point operations, the use of hardware, circuit frequency and output accuracy, by varying the word length. The XOR problem is again used to test an RBFNN implementation with 32-bits floating-point operations in FPGAs with online backpropagation learning in (Kim & Jung, 2015), with the Taylor series approximation for the Gaussian function. It is possible to see that the cited works lack to compare the impact of the word length on the precision and circuit consumption for floating-point operations according to a given task, in order to establish an optimal compromise between hardware use and RBFNN accuracy. Moreover, none of them make use of the inherent modular architecture of the RBFNN to provide solutions with the same word length but with less hardware resources. As seen, none is dedicated to the case of dynamical system modeling. The present paper aims to tackle those issues and to test the RBFNN architecture on hardware in the system identification scenario with real acquired data, as will be detailed later in this section.

Among the most important factors when implementing ANNs on FPGA hardware, one may cite (Hu, Huang, Xing, & Wang, 2008): data representation, inner-product calculation, implementation of the activation function, storage and update of the weights. With respect to data representation, precision should be considered as it will directly affect the final result. Note however that the resources needed in the hardware will increase together with the precision. On this matter, floating-point representations present advantages over fixed-point (Smith, 1997) given that a fixed-point representation would require a larger word to obtain the same precision as with the floating-point representation (Ferreira, Ribeiro, Antunes, & Dias, 2007). Nonetheless, there is not extensive support for designers on the topic of floating-point arithmetics in FPGAs (Sahin, Becerikli, & Yazici, 2006). In the following, we present aspects regarding the motivation of the present study which deals with architectures of RBFNNs with custom-precision floating-point operations applied to black-box dynamic model representations on hardware.

1.1. Motivation

The RBFNN implementation on FPGAs with floating-point operations using custom-precision may be employed on a variety of fields, where the compromise between the hardware resources and precision is determinant. It is possible to readily apply, in the context of model-based control approaches, the dynamic model of the system through black-box ANNs. In this context, it is important to have fast and accurate system responses as we shall demonstrate in the present paper. ANNs are able to capture the dynamics of complex systems (Nørgård, Ravn, Poulsen, & Hansen, 2000) and its implementation on hardware may leverage their application to hardware-in-the-loop real-time simulation (Craciun et al., 2014). We highlight below some applications of the herein proposed RBFNN hardware implementations related to system identification, Model Predictive Control (MPC) and moving-horizon state estimation.

In tracking systems which are time variant, adaptive estimation and system identification are of great importance. On these matters, self-adaptation may be implemented in FPGAs through ANNs (Haykin, 1995) what would make possible the application of this class of algorithms online. Under this circumstance, it is needed to readily obtain up-to-date mathematical abstractions of the system in order to change the parameters and structure of the controller. In this context, adaptive system identification with ANNs and metaheuristic optimization algorithms have been implemented in (Cavuslu, Karakuzu, and Karakaya, 2012) and (Karakuzu, Karakaya, and Avuslu, 2016). Fault diagnosis is also related to this topic, which is important in the context of fault tolerant control. In (Cabal-Yepez et al., 2012) the authors implement a neural classifier in FPGA to detect multiple combined faulty modes for an induction motor.

Evolutionary algorithms (Simon, 2013) may be used in the context of self-adaptation in system identification in order to define the weights in the online learning process in FPGAs (Muñoz, Llanos, Coelho, & Ayala-Rincón, 2014)—which are convenient for hardware implementation due to their simple mathematical operations and may thus be implemented more quickly, and have proven to be valuable tools for learning algorithms (Yao, 1999). Yet one might implement computationally intensive powerful neural control techniques in FPGAs using the herein proposed architectures for RBFNN hardware design. The RBFNN may learn the inverse dynamic behavior of the system offline to define the control action (Nørgård et al., 2000). The problem of regulating the temperature of a solar power plant is tackled in (Henriques, Gil, Cardoso, Carvalho, & Dourado, 2010), where the dynamic model is represented by an ANN offline and adapted online with an ad-hoc Kalman filter in order to account for uncertainties. In (Na, Ren, Herrmann, & Qiao, 2011) the authors propose an adaptive neural controller for servo systems with dead-zone and multiple time-delays. A neuro-adaptive auto-landing control algorithm with dynamic inversion was proposed in (Ambati & Padhi, 2016) with an RBFNN employed to model uncertainties and disturbances.

Another possibility is the application of approximate solutions for problems which in general require the resolution of a nonlinear programming problem at each sampling interval—which restrict their scope of application, even though they are powerful techniques. Among them, some examples are MPC and moving-horizon state estimation.

The former may be treated as the resolution of an optimization problem as each sampling instant in order to find the optimal control action considering a prediction horizon, what may be faced as a mapping from the feedback error and the model to the control law. This mapping has been approximated by ANNs in e.g. (Gomez-Ortega & Camacho, 1994), what is convenient when the MPC problem is not possible to be analytically solved and high frequencies are required for the controller. In (Peyrl, Zanarini, Besselmann, Liu, & Boéchat, 2014) the authors adopt a different strategy by implementing a parallel fast gradient descent algorithm on FPGAs and multi-core CPUs to solve linear quadratic MPC problems.

The later amounts to the state estimation problem using a window of most recent output measurements, which may be seen as a mapping from the outputs to the estimated states. This may be treated similarly as in the MPC case, by the offline construction of ANNs to obtain online predictions. Moving-horizon state estimation algorithms for nonlinear systems have been proposed in e.g. (Alessandri, Baglietto, & Battistelli, 2008) together with an ANN-based approximate version; for switching systems see for the linear case (Alessandri, Baglietto, & Battistelli, 2005) and (Baglietto, Battistelli, Ayala, & Tesi, 2012) for the nonlinear case.

As will be described in the present paper, the floating-point implementation in hardware herein presented of RBFNN guarantees time rates in the order of microseconds. This may leverage the application with high frequencies on FPGAs of the aforementioned powerful algorithms, yet computationally intensive, by the relaxation with approximate solutions. Yet, the implementation of model based control techniques may use ANN models. Adaptive system identification may also be implemented on hardware by coupling the architecture herein presented with a parameter estimation algorithm.

1.2. Contribution and manuscript organization

The present work proposes two novel architectures for the implementation of RBFNNs in FPGA. The implementation focuses on (i) custom floating-point representation, aiming at the precision required by the solution; (ii) fully parallel and shared architectures of the RBFNN, which guarantee fast response in the former case and optimized use of the FPGA resources in the later; (iii) validation of the implemented RBFNN approaches in the case of nonlinear black-box system identification problem, by implementing the architectures in a Xilinx Artix-7 FPGA, which is a low-cost hardware alternative. The system identification problem chosen to validate the proposed archi-