Microprocessors and Microsystems 36 (2012) 138–150

Contents lists available at ScienceDirect

Microprocessors and Microsystems

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

A scalable pipelined architecture for real-time computation of MLP-BP neural networks

Antony Savich *, Medhat Moussa, Shawki Areibi

School of Engineering, University of Guelph, Guelph, Ontario, Canada N1G 2W1

ARTICLE INFO

Article history:
Available online 13 January 2011

Keywords:
Field programmable gate arrays
Parallel computing
Artificial Neural Networks
Multi-layer perceptron
Scalability
Hardware accelerators
On-line learning

ABSTRACT

In this paper a novel architecture for implementing multi-layer perceptron (MLP) neural networks on field programmable gate arrays (FPGA) is presented. The architecture presents a new scalable design that allows variable degrees of parallelism in order to achieve the best balance between performance and FPGA resources usage. Performance is enhanced using a highly efficient pipelined design. Extensive analysis and simulations have been conducted on four standard benchmark problems. Results show that a minimum performance boost of three orders of magnitude (O^3) over software implementation is regularly achieved. We report performance of 2–67 GCUPS for these simple problems, and performance reaching over 1 TCUPS for larger networks and different single FPGA chips. To our knowledge, this is the highest speed reported to date for any MLP network implementation on FPGAs.

1. Introduction

Machine learning algorithms are techniques for automatically building models of complex systems to discover patterns, and make predictions about future events. They are increasingly used in a large number of applications ranging from financial prediction to on-line face recognition and medical image classification.

Some of the most commonly used machine learning algorithms are Artificial Neural Networks (ANN). A recent business report [1] has predicted that the market of software using ANN will reach $4.5 billion sales with AAGR of 12.6% in 2007. ANN is now globally recognized as the most effective and appropriate AI technology for pattern recognition applicable in business forecasting, fraud detection, process modeling, data classification and analysis. Furthermore, developing ANN solutions that can operate in real-time that use off-the-shelf chips could open the door for far more integration in embedded devices and real-time systems.

Multi-layer perceptron trained using the error back-propagation algorithm (MLP-BP) [2] is by far the most studied and used ANN architecture. It has been used extensively in a wide variety of applications. Yet using MLP-BP in real-time applications faces several challenges. Training of an MLP-BP network is known to be time consuming especially for large networks. This is compounded by the lack of clear methodology in setting up the initial topology and parameters. Topology has a significant impact on the network's computational ability to learn the target function and to generalize from training patterns to new patterns. If the network has too few free parameters (weights), training could fail to achieve the required error threshold. On the other hand, if the network has too many free parameters, then a large data set is needed. In this case the possibility of over-fit is higher, which impacts generalization. It is typically not possible to experiment with a large number of topologies because of the long training sessions required. As a result, heuristics have typically been used to speed the training process while preventing over-fitting [3]. Yet even with the use of heuristics, this training process is limited to open loop learning or to applications where training data is static and the problem domain unchanged for the duration of the network's useful function. However, when closed loop learning is necessary or when the solution space is dynamic and new data is being added continuously, there is a critical need for testing a wide range of topologies in real-time. For example, real-time data mining of customers' databases that are continuously updated is a growing area with significant commercial interest. As a result, developing special hardware ANN accelerators continues to be an area of significant research and commercial interest.

1.1. Previous approaches

Since ANNs are inherently parallel architectures, there have been many efforts to explore parallel computing architectures for implementing ANNs. These activities range from implementations on general-purpose parallel computers to specialized hardware dedicated to ANN simulations (neurocomputers). Examples include ANN simulations on Hypercube, connection machines, and

* Corresponding author.
E-mail address: asavich@uoguelph.ca (A. Savich).

0141-9331/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.micpro.2010.12.001

other supercomputers [4–6]. Other groups have designed and built parallel systems based on transputers [7], or digital signal processors (DSPs) [8]. Several companies have proposed Application Specific Integrated Circuits (ASICs) that act like ANN accelerators. Examples include CNAPS [9], and Synapse-1 [10]. The reader can refer to Nordstrom and Svensson [11] or Dias [12] for an overview of these efforts. Most of these designs require using special hardware boards or ASIC chips which limit their use on a large scale. Furthermore, the simulated ANNs are constrained by size and type of algorithm implemented.

A different approach that has gained considerable attention in the last 10 years focuses on ANN implementation on configurable platforms, and particularly Field Programmable Gate Arrays (FPGAs) [13–18]. Commercial development of large, dense and configurable FPGA chips has allowed implementations with more flexibility of network size, type, topology, and other constraints while maintaining increased processing density using the natural parallel structure of ANNs. FPGAs compare favorably against other emerging acceleration technologies like Clearspeed (www.clearspeed.com) and CUDA (www.nvidia.com) since they are already used extensively in a wide range of embedded applications and relatively inexpensive. However, using FPGAs to implement ANNs, and specifically MLP-BP, raises a number of design issues [19]. In general, an MLP-BP implementation on an FPGA can be evaluated in terms of:

1. Performance: in terms of number of connection updates processed per second (CUPS) and in terms of convergence speed.
2. Resource utilization: in terms of FPGA resources used and its proportion to total FPGA chip resources.

Every implementation is a tradeoff among these two factors. High performance requires high degree of parallelism which requires significant resources, while serial implementation requires less resources but severely impacts performance. To achieve resource efficient implementation, some efforts have focused on determining the most resource efficient arithmetic representation format while maintaining adequate precision to achieve learning [20,21]. Yet performance remained the main goal. Elridge et al. [22] used Runtime Reconfiguration (RTR) to improve the hardware density of FPGAs by dividing the BP algorithm into three sequentially executed stages. The FPGA was configured to execute only one stage at a time. The use of RTR enhanced processing density. However this comes at a cost of deteriorating performance due to the time needed to reconfigure the device.

More recently, Gadea et al. [17] described the implementation of a systolic array for MLP. A pipelined modification of the on-line back-propagation algorithm is presented where both the forward and backward passes are processed in parallel thus achieving a performance boost in training reported around 5 GCUPS. The modification itself requires circumventing some temporal properties of the algorithm. This creates a marginal degradation in training convergence but this performance level was previously unattainable even when using custom neurocomputing platforms. Paul et al. [23] also propose a similar systolic array architecture which achieves 5 GOPS[1] performance. But it is not clear how GOPS relate to the more widely used CUPS metric. Both of these designs have static relationships between network size and resource utilization, where the resources needed are directly proportional to the size of the network and its topology regardless of the FPGA available resource. This led Gadea et al. [17] to conclude that the resources required for implementing large scale networks make their design impractical for current FPGAs regardless of performance achieved. Aliaga et al. [18] presents another approach based on a multinode co-processor embedded in an FPGA and controlled by an on-chip CPU. Performance of about 700 MCUPS is achieved on a single chip, combined with soft programmability features allowing this approach to implement large network. The inefficiencies of a software controlled architecture do not allow this particular approach to produce substantial performance levels and does not provide a notable speedup over software implementations.

Another issue that is often overlooked is the impact of the nature of the application on the resource utilization. Both the network topology and the type of arithmetic format used in the implementation are impacted by this nature of the application. For example, pattern classification problems can allow a higher error threshold than function approximation problems since they basically pick a one-of-N classes. That impacts the choice of the arithmetic representation and, as such, the FPGA resources used in the implementation. This in turn could impact performance since large resource utilization impacts the overall size of the network topology that can be implemented in parallel and overall performance.

In this paper, we present a scalable architecture for implementing MLP-BP on FPGAs. The objective is to balance between maximizing performance and minimizing resources utilized. This is particularly critical for implementing ANNs as part of an embedded system where an FPGAs is typically used to handle a wide range of tasks. In these circumstances resource utilization is as important as performance since the ANN implementation is likely to share the FPGA resources with other tasks. The proposed architecture can scale the ANN implementation to fit within the available resources while achieving the highest possible performance. This built-in balancing mechanism is consistent across a wide range of network topologies and FPGA chips.

1.2. Contributions and organization

The most interesting feature of the novel architecture presented in this paper is *scalability*. Scalability is implemented through using *variable degree of parallelism*. An MLP-BP network is implemented in a fully parallel design, that includes synapse and node parallelism, to maximize performance when the resources available on the FPGA allow that. When resources are limited, the implementation uses a reduced degree of parallelism up to the point needed to fit the network in the available resources while also achieving best performance. This allows the architecture performance and resource utilization to scale up well for large as well as small networks. We are not aware of any other ANN architecture in literature with such scalable design.

The impact of the scalability feature is augmented by incorporating two additional features to maximize performance and resource utilization in all cases regardless of the size of the network. To maximize performance, the architecture implements a highly efficient pipelined design based on using an out of order weight update rule to overcome the data hazard inherent in the BP algorithm. To maximize resource utilization, it selects the most efficient arithmetic representation that balances between precision and area based on our previous study examining the impact of arithmetic representation on area size and precision [20]. To combine all three features of scalability, pipelining and efficient implementation, we developed new formulas that link performance and resource utilization with the network topology and degree of parallelism. This allows custom implementation of every network to maximize performance and resource utilization.

Tests conducted on several standard benchmark problems show high resource utilization efficiency while achieving remarkable algorithm performance – up to five orders of magnitude (O^5) boost over software based ANN simulators. Comparing performance to

[1] In [23], the term GOPS is not specifically defined, but is generally understood to represent Giga Operations Per Second.