HIGH THROUGHPUT SYSTOLIC SOM IP CORE FOR FPGAs

I. Manolakos *E. Logaras*

Department of Informatics and Telecommunications

University of Athens, Greece

ABSTRACT

We have designed a modular SOM systolic architecture that can classify data vectors with thousands of elements in real time. The architecture is described as a soft IP core in synthesizable VHDL. The SOM neural network size, the input data vectors dimension, the weight and data element bitwidth precision etc. are all designer tunable parameters. Several SOM neural network instances have been synthesized and their performance evaluated for different Xilinx Virtex-II and Virtex II-Pro FPGAs. Moderate to large size SOM networks that can process data vectors with as many as 4096 elements can fit into a single FPGA device and clocked at frequencies as high as 150MHz. This makes the architecture a useful co-processor candidate for the real-time categorization of large-size genomics and proteomics datasets.

Index Terms— Self-organizing feature maps, Parallel Architectures, Systolic arrays, Design automation, FPGAs.

1. INTRODUCTION

Kohonen's Self Organizing Maps (SOMs) [1] are unsupervised neural network structures, commonly employed for clustering highly dimensional data vectors. SOMs have been used for signal/image processing and machine learning tasks, such as image compression [2] feature extraction and pattern recognition, in several application domains, ranging from industrial monitoring and automotive control to computational biology.

When high dimensional data vectors are presented to the SOM, the network can be trained in an unsupervised manner to construct a low dimensional representation of their distribution (learning mode). This distribution can then be visualized to provide feedback on the organization of the underlying data vectors. It is these dimensionality reduction and visualization capabilities that have made SOMs a popular tool for exploratory data analysis and data mining.

A trained SOM network can be used to classify novel data vectors into as many categories as its neurons based on their statistical characteristics (recall mode). SOMs with a small to moderate number of neurons have been used in genomics and proteomics to group in clusters large sets of input vectors with thousands of elements (e.g.. gene expression profiles [3,4] or peak intensities in mass spectra).

As the number of neurons and input vector elements increase, SOM simulation on a powerful PC may not be sufficient for applications that demand with near real-time performance. A powerful PC (e.g. AMD Athlon, 1GHz) may not exceed 85 MCPS (Million Connections Per Second) in recall mode and 22 MCUPS (Million Connection Updates Per Second) in learning mode, as it was shown in [5]. Therefore, several special purpose designs have been proposed to accelerate SOM processing by employing custom ASICs or FPGAs [5-7]. FPGAs are recently gaining in popularity, since they now have the capacity to house a whole SOM System-On-chip (SoC) into a single device. In addition to rapid prototyping, some FPGAs also support dynamic reconfiguration. This capability allows for adapting the SOM network size and weights bit precision at run-time to deal with dynamically changing data processing demands or energy consumption constraints. However, to harness the full power of FPGAs a high throughput parallel SOM architecture needs to be designed and realized as a flexible soft IP core. It can form the basis for generating different SOM network implementations tailored to the requirements and constraints of the target application and device. This is the main contribution of the work presented here.

Most SOM hardware implementations proposed so far, have adopted the SIMD architectural model [5-7]. In this model during a cycle the same data element is broadcasted to every Processing Element (PE) that is either processing it or remains idle. Theoretically, the need for massive data broadcasting complicates data routing and limits the clock frequency of such a system. In this paper we are presenting a parallel SOM architecture design following the systolic model, in which a single data path traverses all neuron PEs and is aggressively pipelined. With this approach the paths formed in the network become shorter, the cycles needed for classifying a data vector are reduced, and the clock frequency can become very high. The design is described as an IP core in VHDL, where the number of neurons *(N),* the number of elements per input vector elements *(M)* and the number of bits for data and weights are all tunable parameters.

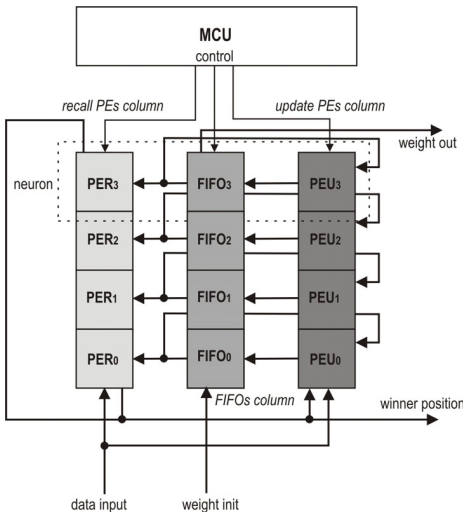

Fig.1 Overview of the modular systolic SOM architecture.

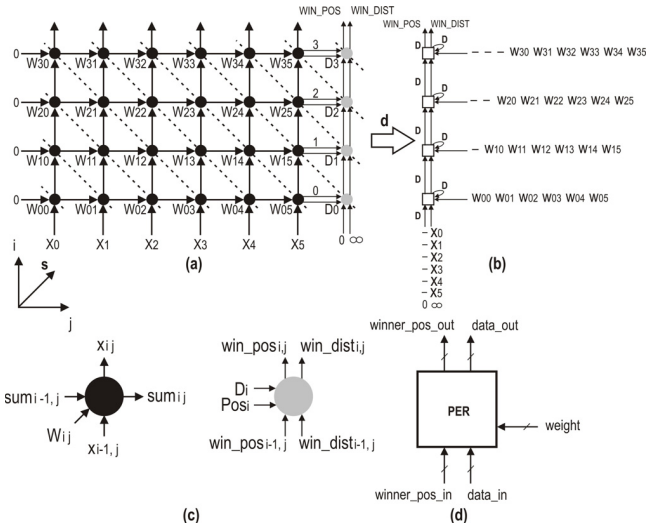

Fig.2 (a) The DG for a vector recall computation, *N=4, M*=6, (b) The resulting SFG array after linear space-time mapping, (c) The two types of DG nodes. (d) The resulting PER I/O structure**.**

The rest of the paper is organized as follows: In section 2 we present the systematic design and performance analysis of the parallel SOM architecture. In section 3, we discuss its realization as a flexible IP core and present the characteristics of synthesized instances for different FPGA devices. We also elaborate on the design of a flexible, parameterized and synthesizable FIFO memory block for the neuron weights, which can support SOMs with different vector sizes and weights precision. Furthermore, we compare our architecture to others proposed for the same problem. Finally, in section 4 we summarize our findings and point to interesting work in progress.

2. SOM ARCHITECTURE DESIGN

2.1 Architecture Overview

The main requirements for the proposed high throughput modular SOM architecture are: a new data element should enter the architecture every cycle, it should flow in a pipelined fashion from PE to PE, and the supported data rates should be as high as possible.

Every SOM neuron is implemented using two simple Processing Elements (PEs): the Recall mode PE (PER), and the weights Update PE (PEU). The two types of PEs in a neuron are organized into two separate "columns" (recall and update linear arrays), but share the same dedicated FIFO memory block storing the neuron's weights, as shown in Figure 1. The number of PEs in each column matches the total number of neurons in the SOM network. An SOM Module Control Unit (MCU) generates all the necessary control signals for both array columns. The winning neuron's position is generated at the topmost PER_{N-1}. It is forwarded to the bottom PEU_0 in the update column, if the weights should be updated after a recall operation (learning mode). A `weight init` (`weight out`) input (output) is available for downloading (uploading) the weight matrix from (to) an external memory device respectively for initialization and debugging purposes.

2.2 The Recall Column PER array design

The objective of the PEs in the PER column is to compute the distances between the locally stored weight vectors (in the FIFOs) and the passing through input data vector, and determine the position (index) of the neuron exhibiting the smallest distance (winner). For the *i-th* neuron, the Manhattan distance D_i is computed as:

$$D_i = \sum_{j=1}^{M} \left| W_{ij} - X_j \right| \qquad (1)$$

where M is the number of elements in the input data vector.

In Fig. 2(a) we show the Dependence Graph (DG) [8] for a recall phase computation, consisting of two sections, one corresponding to the distance calculation (black nodes) and another to finding the winner's position (gray nodes). A black DG node (Fig. 2(c)) models a computation of the absolute value of the difference between a weight and input vector element followed by an accumulation. A gray node models a comparison between a locally computed distance value and a distance value computed in the PE below; it propagates the smaller value and corresponding PE index upstream in the DG. In this fashion, the min. distance and the winner's index emerge at the top right corner DG node (Fig. 2(a)).

The DG is linearly mapped to the Signal Flow Graph (SFG) of Figure 2(b) after applying the projection vector $d^T = [1 \;\; 0]$ and scheduling vector $s^T = [1 \;\; 1]$ [8]. Each element of the resulting linear array (PER) has two

adders and a comparator. Input vector elements are perfectly pipelined and pass from all PERs one by one. The cycles needed for a vector recall operation, C_r, and the corresponding performance, P_r, are

$C_r = M + N$ cycles and

$$P_r = \frac{N(M+1)}{M+N} \cdot f \quad \text{MCPS},$$

where f is the clock frequency of the system. Using *block pipelining*, a new input vector may be introduced to the array as soon as a distance calculation completes at PER_0 and while the previous winner determination is still in progress (gray nodes DG). In this case, the cycles needed for a vector recall and the corresponding performance become:

$C_r^b = M + 1$ cycles, and

$$P_r^b = \frac{N(M+1)}{M+1} \cdot f = N \cdot f \quad \text{MCPS respectively.}$$

2.3 The Update column PEU array

The PEU column is responsible for updating in parallel the neuron weights during a learning mode SOM operation. The position of the winning neuron (found by the PER column) is transmitted to the PEU column where it is used in each PE to update the weights based on the formula:

$$W_{ij}^{new} = W_{ij} + a_i \cdot |X_j - W_{ij}|. \qquad (2)$$

Variable a_i is the update rate, which should decrease as the distance of neuron i from the winning neuron increases. It is computed using the formula $a_i = 1/2^{h_i}$, where

$$h_i = \max\{|i'_x - i_x|, |i'_y - i_y|\} \qquad (3)$$

is the maximal difference between the winning neuron's i' and SOM neuron's i Cartesian coordinates in the x (horizontal) and y (vertical) directions. This definition of a_i leads to a rectangular neighborhood function that approximates a 2D Gaussian (Mexican hat). It has been shown that keeping the neighborhood size fixed and reducing the gain around the winning neuron (spatially decreasing a_i) leads to better results than using a constant gain and reducing gradually the size of the neighborhood as the SOM training process progresses [9].

The DG for the weights update operation is shown in Fig. 3(a). A DG node corresponds to a weight update computation (Equ. (2)), where a_i is calculated using h_i that is derived based on (3). Every PEU is assigned at synthesis time the coordinates (i_x, i_y) of the neuron that it corresponds to. Moreover, it receives at run-time the position (i'_x, i'_y) of the winning neuron produced at PER. Therefore, PEU_i has locally all the information it needs to compute a_i. By using the same projection and schedule vectors as for PER we arrive at the linear SFG array of Fig 3(b) consisting of N

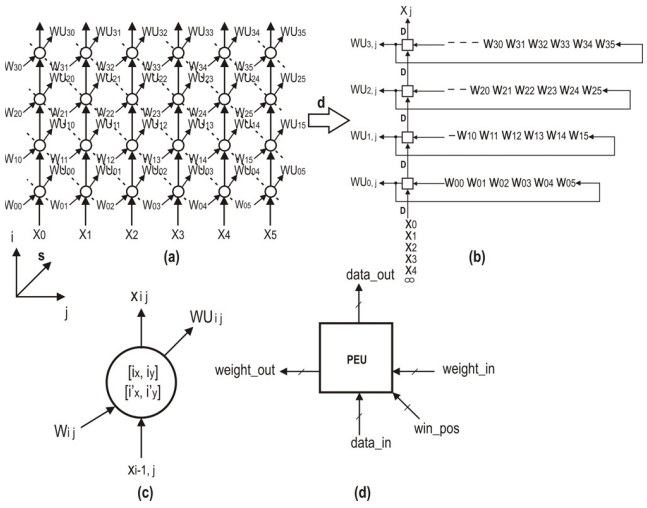

Fig.3 (a) The DG for a vector update computation, *N=4, M=6,* (b) The resulting SFG array after linear space-time mapping, (c) The DG node, (d) The PEU I/O structure.

PEUs. The PEU I/O structure is shown in Fig. 3(d).

The number of cycles C_l needed for a learning operation (recall plus update for one vector) and the corresponding learning performance, P_l, are given by:

$C_l = C_r^b + C_u = M + 1 + M + N - 1 = 2 \cdot M + N$ cycles, and

$$P_l = \frac{N_u \cdot M}{2 \cdot M + N} \cdot f \quad \text{MCUPS},$$

where N_u, is the expected number of updated weights, estimated as the arithmetic mean of the neurons in the neighborhood of the winner and assuming that h_i is in the range [1,4] and all neurons have equal probability to win.

3. RESULTS AND DISCUSSION

The proposed parameterized SOM architecture has been described in VHDL and synthesized using the XST Xilinx FPGA compiler [10]. The design can be implemented using any FPGA device with internal static RAM blocks (BRAMs) that are needed to realize the FIFO neuron weight memories. Table 1 provides the size and performance characteristics of several SOM networks synthesized using the soft IP core but different FPGAs.

The maximal number of neurons that can fit in an FPGA is determined by the available Configurable Logic Blocks (CLBs). For the devices listed in Table 1 we first found the maximum number of neurons N_{max} that does not exceed 75% CLB coverage (considering that at least 25% of device resources should be reserved for external memory interfacing, reconfiguration control etc.). Then, with N_{max} fixed, weights of 8-bit and data elements of 12-bit precision (typical parameters used also in [5]), we have determined M_{max}, i.e. the maximal number of elements per data vector that can be supported using the available BRAMs. For every such configuration, Table 1 provides the achieved clock frequency, the performance of the recall and learning

Device	N_{maz}	M_{max}	f (MHz)	P_r MCPS	P_l MCUPS	CLBs %
XC2V2000-6	25	4096	150	3750	1794	73
XC2V4000-6	56	4096	150	8400	2899	75
XC2V6000-6	100	2048	148	14800	3467	72
XC2V8000-5	140	2048	127	17780	3192	72
XC2VP30-6	32	4096	165	5280	2054	72
XC2VP50-6	56	4096	166	9296	3208	72

Table 1. Synthesis results for Virtex-II and Virtex-II Pro FPGA devices for 8-bit weights and 12-bit data elements.

phases and the percentage of utilized device resources. It is observed that moderate to large size parallel SOM architectures fit into one FPGA and can process effectively large size data vectors. Very high data rates were achieved in all configurations.

Depending on the number of bits used per weight (which is a parameter in the VHDL description) the BRAMs are configured automatically either as 2K x 8-bit or as 1K x 16-bit. The VHDL FIFO entity is very flexible such that if the number of data elements *M* per input data vector (number of elements in each FIFO) exceeds 2K or the weights precision exceeds 8-bits, multiple BRAMs are automatically cascaded in order to form an appropriate FIFO memory structure for each neuron. In this way, the design can support SOM networks with a large number of weights per neuron and any desirable weight precision up to 16-bits, limited only by the available number of BRAMs in the target FPGA.

It is interesting to compare and contrast our system's synthesis results for the XC2V6000 device to those of the SIMD architecture described in [6] when implemented in the same device. Our system is more flexible in terms of memory organization since it was designed to accommodate as large vector sizes *(M)* as possible. Its supported data rates are much higher (as expected for a highly pipelined architecture). The two systems achieve comparable learning performance, but with each one of them favoring by design a different side of the *M* vs. *N* tradeoff. At its maximal capacity $(N_{maz}=100)$ the recall performance of our systolic system is larger. However, it cannot accommodate *N=288* neurons as the SIMD solution, even if *M* is reduced. This is so because a lot of the functionality concentrated in the SIMD controller is replicated in every PE in our case (e.g. the calculation of a_i). Furthermore, the existence of *O(N)* comparators in our design consumes more area, but also, in conjunction with block pipelining, helps determining the winning neuron on the fly. The design in [6] is optimized for maximal *N* while our design is optimized for maximal *M*. For the targeted bioinformatics applications, it is required to support the largest possible *M* for moderate *N*s.

4. CONCLUSIONS

We presented the systematic design and performance evaluation of a new systolic SOM parallel architecture. The architecture has been described as a soft IP core in parameterizable VHDL. The number of neurons, the number of elements in the input data vectors, the weight and data precisions are all parameters tunable by the designer. The architecture is primarily suitable for the real-time classification of high dimensional data vectors. When using a large-size Xilinx Virtex-II FPGA device (XC2V6000) for the implementation of a 10x10 SOM network processing data vectors with 1K elements, synthesis results have shown that a classification rate of 144,000 vectors/sec and a learning rate of 68,900 vectors/sec can be achieved. We are currently investigating the use of the architecture in the real-time clustering of gene expression profiles as well as in clustering mass spectra, two operations commonly used in high throughput genomics and proteomics analysis for biomarker discovery.

5. REFERENCES

[1] T. Kohonen, *Self-Organization and Associative Memory*, Springer Verlag, New York, 1984.

[2] C. Amerijckx, J.-D. Legat, M. Verleysen, "Image compression using self-organizing maps," *Systems Analysis Modelling Simulation*, Vol. 43, pp. 1529-1543, Nov. 2003.

[3] D. Wang, H. Ressom, M. Musavi, C. Domnisoru, "Double Self-Organizing Maps to Cluster Gene Expression Data", Proc. *European Symp. on Artificial Neural Networks*, 2002, pp. 45-50.

[4] T. Kato, K. Fujimura, H. Tokutaka, Y. Kawata, M. Ohkita,"Analysis of DNA Microarray Data by Using SOMs", *Genome Informatics*, Vol. 14, pp. 328-329, 2003.

[5] D. Hendry, A. Duncan, N. Lightowler, "IP Core Implementation of a Self-Organizing Neural Network," *IEEE Trans. on Neural Networks*, Vol. 14, pp. 1085-1096, Sept. 2003.

[6] M. Porrmann, U. Witkowski, H. Kalte, U. Rückert, "Dynamically Reconfigurable Hardware: A New Perspective for Neural Network Implementation", Proc. *12th Int'l. Conf. on Field Programmable Logic and Applications*, 2002, pp.1048-1057.

[7] M. Franzmeier, C. Pohl, M. Porrmann, U. Rückert, "Hardware Accelerated Data Analysis," *Proc. Int'l Conf. on Parallel Computing in Electrical Engineering*, 2004, pp. 309-314.

[8] Keshab K. Parhi, *VLSI Signal Processing Systems – Design and Implementations*, Willey Inter-Science, 1999.

[9] P. Thiran, V.Peiris, P. Heim, B. Hochet, "Quantization effects in digitally behaving circuit implementations of Kohonen networks," *IEEE Trans .on Neural Networks*, Vol. 5, pp. 450-458, May 1994.

[10] XST User Guide 8.1i, www.xilinx.com, 2005.