ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

gNBXe – a Reconfigurable Neuroprocessor for Various Types of Self-Organizing Maps

Jan Lachmair[1], Erzsébet Merényi[2], Mario Porrmann[1], Ulrich Rückert[1] *

1- University of Bielefeld - Cognitronics and Sensor Systems
Bielefeld - Germany

2- Rice University - Department of Statistics
Houston, Texas, U.S.A.

Abstract. In this paper we present the FPGA-based hardware accelerator gNBXe for emulation of classical Self-Organizing Maps (SOMs) and Conscience SOM (CSOM) in a multi-FPGA environment. After discussing how the CSOM is mapped to a resource-efficient digital hardware implementation, we present how the modular system architecture can be flexibly adapted to various application datasets. The hardware costs and scalability of a multi-FPGA based accelerator using Xilinx Virtex2 and Virtex4 FPGAs are discussed. Compared to a state-of-the-art multi-core PC, a speedup of 9.1 is achieved for a CSOM with 4,840 neurons and 196 synaptic weights.

1 Introduction

Since SOMs were first characterized [1], they became an important tool in analyzing high-dimensional data such as those in hyperspectral imaging tasks [2, 3] or medical research [4]. Following the principle of operation of the human neocortex, a SOM is able to generate a spatially ordered map of functional classes from input space. Some of the main advantages using SOMs for analyzing high-dimensional data are the unsupervised learning and the unique visualization possibilities. Although parallelism is increasing in today's microprocessor architectures, the software-based simulation of large and high-dimensional SOMs still suffers from the mainly sequential processing. Dedicated hardware accelerators as proposed, e.g., in [5, 6] offer an energy-efficient way to speed up SOMs utilizing their high inherent parallelism. However, hardware accelerators often lack flexibility and provide only limited data precision. The proposed hardware accelerator is based on the single-FPGA gNBX neuroprocessor [5] and extends the architecture towards a flexible multi-FPGA based SOM accelerator. In addition to an enhanced scalability, leading to a nearly linear speed-up, we extended the instruction set of our gNBX neuroprocessor towards the simulation of CSOMs according to [7]. In the following section the CSOM algorithm and its realization in digital hardware will be introduced. Section 3 details the modular system architecture and its partitioning. In section 4 the performance of the architecture is evaluated based on an implementation on an FPGA-based prototyping

*This research is partially supported by the Center of Excellence Cognitive Interaction Technology (CITEC).

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

system. The performance of the accelerator and the hardware costs for different FPGAs are demonstrated using a real-world problem.

2 Conscience SOM implementation

The classical SOM algorithm causes a magnification of frequently presented input data of a single cluster. This magnification effect results in a proportional under-representation of sparsely used clusters. The associated magnification factor ρ describes the relation between the point density $D()$ of the SOM weights ($\mathbf{w}_i$) in the input space and the probability density function $P()$ of the stimuli data as $D(w) \propto P(w)^\rho$ [8]. To maximize the information in the trained SOM, a magnification factor of 1 is desired. It is shown in [9] that the CSOM achieves a magnification factor of 1 also for higher dimensional data. The magnification factor of 1 is achieved by adding a bias value g_i (the conscience) to the calculation of the distance between input vector $\mathbf{x}$ and $\mathbf{w}_i$ (equation (1)) to ensure that a neuron which was recently the best matching unit (BMU) won't shortly be the BMU again.

$$\|\mathbf{x}(t)-\mathbf{w}_c(t)\|_p - g_c(t) < \|\mathbf{x}(t) - \mathbf{w}_i(t)\|_p - g_i(t) \;\forall i \neq c \tag{1}$$

$$g_i(t) = \gamma(t)\left(\frac{1}{l} - F_i(t)\right) \tag{2}$$

$$F_i(t) = F_i(t-1) + \beta(t)(y_i - F_i(t-1)) \tag{3}$$

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + h_{c,i}(t)[\mathbf{x} - \mathbf{w}_i]; \quad h_{c,i}(t) = \begin{cases} \alpha(t) & \text{if } \|\mathbf{r}_c - \mathbf{r}_i\|_p \leq 1 \\ 0 & \text{if } \|\mathbf{r}_c - \mathbf{r}_i\|_p > 1 \end{cases} \tag{4}$$

The parameter c is the index of the winner neuron, l is the number of neurons in the SOM (static for classical SOM and CSOM) and F_i is the winning frequency of neuron i. The bias g_i (equation (2)) is calculated at each learning step. The bias depends on the user parameter γ and the winning frequency F_i (equation (3)). The user controlled parameters (α, β, γ) have to decrease during simulation in order to strengthen already learned information. The parameter y_i is 1 for the BMU and 0 for each other neuron. In contrast to classical SOM the CSOM only updates the immediate lattice neighbors (i.e., neighborhood function $h_{c,i}$ = constant box function of radius 1)(equation (4)). An advantage when mapping the conscience SOM to our hardware is the similarity between the calculation of g_i and F_i to the weight update algorithm of the classical SOM. During weight update the distance between each neuron and the BMU in the lattice has to be calculated and the prototypes of the BMU ($\mathbf{w}_c$) as well as the prototypes of its neighbor neurons are adapted according to equation (4). Because of the similarity between the calculation rules for F_i as well as g_i and the weight update calculation also necessary for classical SOMs, F_i and g_i can easily be calculated without the need of additional calculation units. Another advantage when using CSOM is the constant neighborhood function $h_{c,i}$ which can be realized much easier in hardware than a Gaussian function, often used in classical SOMs. For hardware implementation of the CSOM algorithm, the instruction set of the

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

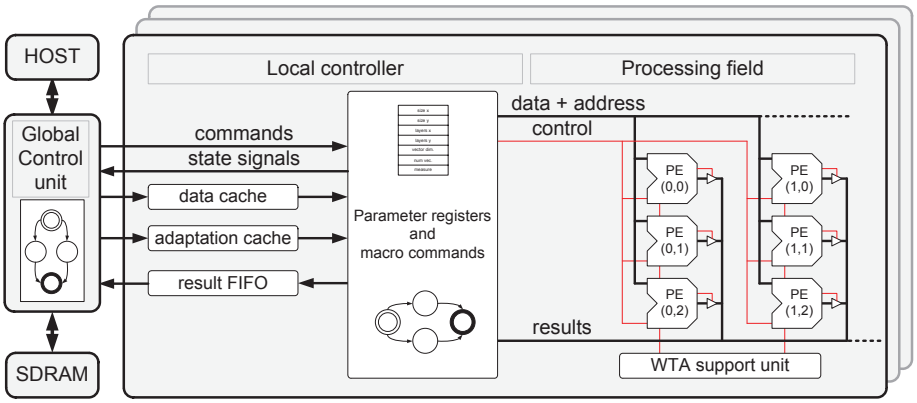

Fig. 1: Principle of the modular system architecture

gNBX has been extended to support bias and winning frequency calculation. The numbers of clock cycles for the two most computationally intensive operations – distance calculation and BMU localization ($t_{\mathrm{CD\_FB}}$) and spreading adaptation values and neuron adaptation ($t_{\mathrm{SA\_AD}}$) – are given by:

$$t_{\mathrm{CD\_FB}} = num\_layers * (14 + \dim(\mathbf{x}) + sum\_width) + 12$$
$$t_{\mathrm{SA\_AD}} = num\_layers * (11 + \dim(\mathbf{x}) + p\_norm) + 4$$

The parameter num_layers is the number of neurons each processing element (PE) has to simulate sequentially (see section 3). The number of synaptic circuits is represented as $\dim(\mathbf{x})$. In case of a diamond neighborhood shape the parameter p_norm will be set to 1 and in case of a squared shape it will be set to 2. The parameter sum_width depends on the desired calculation precision ($data\_width$) as well as on the available address space ($addr\_width$) of the PEs and is set to $sum\_width = 2 * data\_width + addr\_width$ to realize distance calculation with full precision.

3 Modular system architecture

The modular system architecture (figure 1) of our hardware accelerator is specified in VHDL and optimized for Xilinx FPGAs. It consists of a central control FPGA including the global controller (GC) and several PE-FPGAs. The calculation precision, the number of processing elements per FPGA, and the local memory space of each processing element can be flexibly chosen using VHDL generic parameters. The global controller connects the accelerator to the host system, manages the training and adaptation data, finds the global BMU, and controls the simulation process. The PE-FPGAs consist of a local controller, parallel processing elements each capable to simulate one or more neurons, and a winner takes all (WTA) unit determining the local BMU. The local controller

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

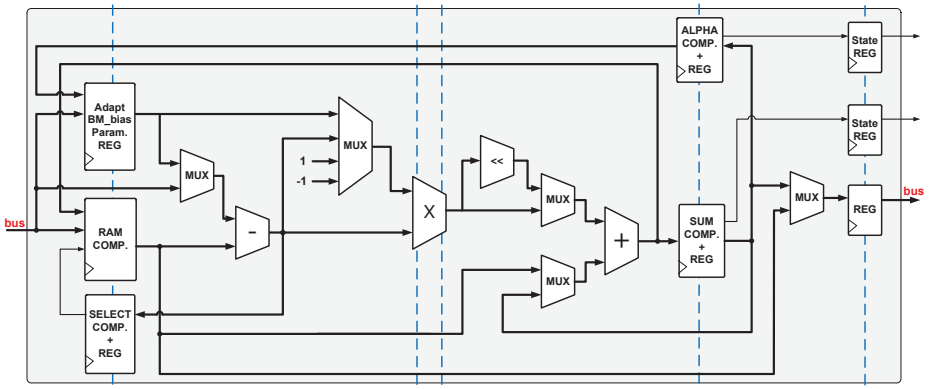

Fig. 2: Architecture of the gNBXe processing elements

translates the macro commands from the GC to control signals for the PEs. The availability of training and adaptation data is managed using additional cache structures. The architecture of the gNBXe processing elements is depicted in figure 2. The data path of the gNBXe-PE is realized in a pipeline structure including additional pipeline bypasses for increased performance. To efficiently realize the $p2$ norm (euclidean), each PE integrates an embedded multiplier of the Xilinx FPGA. Another design goal was a high flexibility with respect to the number of available neurons. Therefore, each processing element is capable of performing the calculations for multiple neurons. The number of neurons that can be emulated by a single PE is only limited by the available memory. Each neuron is represented by an individual address (the position in the $\mathbf{r}$-dimensional map) and its synaptic weights. Both parts are stored in the local address space of a PE, requiring $\dim(\mathbf{w}) + \dim(\mathbf{r})$ addresses. For CSOM each neuron needs an additional address to store its winning frequency. Therefore, in a local address space of k addresses up to $\lfloor k/(\dim(\mathbf{w}) + \dim(\mathbf{r}))\rfloor$ neurons can be stored and calculated sequentially for SOM simulation and $\lfloor k/(\dim(\mathbf{w}) + \dim(\mathbf{r}) + 1)\rfloor$ neurons for CSOM.

4 Performance Evaluation

To demonstrate the performance of the hardware accelerator a hardware implementation with 16 Bit precision, 2048 local addresses, and a clock frequency of 50 MHz was chosen. The accelerator has been implemented utilizing the RAPTOR-X64 prototyping system [10] using four PE-FPGAs (Xilinx Virtex-2Pro as well as Xilinx Virtex-4) and a Xilinx Virtex2-4000 to implement the global controller. Table 1 shows the hardware costs and the number of available neurons for the gNBXe using Xilinx Virtex-2Pro and Virtex-4 FPGAs. Because the gNBXe is the only existing hardware accelerator for CSOM, an optimized software implementation of the CSOM running on a Core-i7 950 Quad Core at

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

Table 1: Hardware costs, available processing elements (PEs) and possible number of neurons (N_N) for different Xilinx FPGAs using 16 Bit precision, $2k$ local address space and varying number of synaptic weights (d).

	V2P30	4× V2P30	V4FX100	4× V4FX100
PEs(max)	36	144	121	484
Slices	13220(96%)	52880	40586(96%)	162344
RAMB16s	82(60%)	328	252(67%)	1008
MULT18x18s	36(26%)	144	121(75%)	484
$N_N(d = 2000)$	36	144	121	484
$N_N(d = 100)$	684	2736	2299	9196
$N_N(d = 8)$	6696	26784	22506	90024

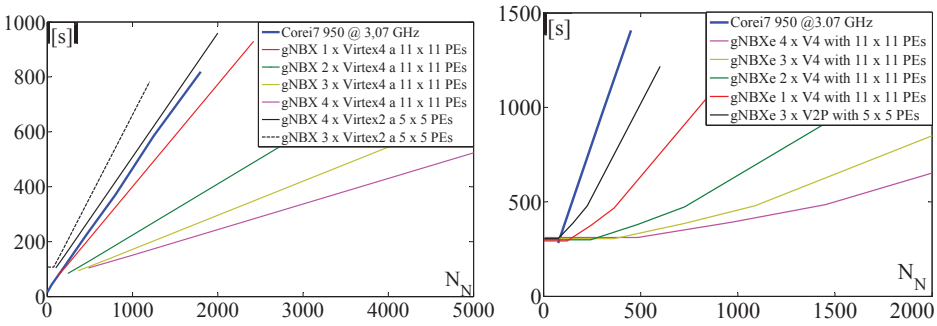

Fig. 3: Execution time for learning $26,214,400$ vectors with $\dim(\mathbf{x}) = 8$ (left) and for learning $15,964,000$ vectors with $\dim(\mathbf{x}) = 196$ (right)

3.07 GHz is used as a reference implementation. The software implementation is written in C++ with double precision and multi-threading enabled using the OpenMP library. The hardware accelerator and the software reference implementation are identically parameterized and trained using a Matlab frontend. Two different datasets are used to evaluate the performance of the implementations during learning. The first dataset is a 512×512 pixel image with 8 spectral bands for each pixel analyzed with software CSOMs in [3]. The second dataset is a hyperspectral image with 614×260 pixel and 196 spectral bands analyzed with software CSOMs in [2]. To compare the different FPGA types, the different number of FPGAs, and the software implementation running on the Core-i7 QuadCore the execution time of the Core-i7 for training all vectors in 100 epochs is measured and compared with the calculated hardware execution time of the gNBXe. The deviation between measured and modeled execution time is less than 0.4% for the core algorithm including initializing of the neuron map as well as read back of the map to the SDRAM of the GC. As can be seen in the left diagram in figure 3, the Core-i7 profits from its high clock speed and multi-core architecture. The older Virtex2 FPGAs using 25 neurons in parallel

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

per FPGA are slower than the softwae implementation on the Core-i7. Using the newer Virtex4 FPGAs, a speed-up of about three can be achieved compared to the Core-i7. Even bigger speed-up is achieved for higher-dimensional data. For the second dataset, the FPGA implementation outperforms the Core-i7 by a factor of 9.1 when utilizing four Xilinx Virtex4 FX100 FPGAs. If the number of neurons is lower than the number of available PEs the hardware execution time stays constant for different map sizes since the available parallelism is not fully utilized. For all measured test cases, the time needed for communication with the host computer does not exceed one percent of the overall computation time.

5 Conclusion

We have shown that the proposed gNBXe-based multi-FPGA hardware accelerator for SOMs and CSOMs significantly speeds up simulation of high-dimensional datasets compared to state-of-the-art multi-core PCs. As a next step we aim to extend the reconfigurable hardware accelerator towards the emulation of additional artificial neural networks. Furthermore, the number of PE-FPGAs will be increased and high-speed serial IOs will be used for communication between the GC and the PEs to further increase the scalability of the system.

References

[1] T. Kohonen. Automatic formation of topological maps of patterns in a self-organizing system. In E. Oja and O. Simula, editors, *Proceedings of 2SCIA, Scand. Conference on Image Analysis*, pages 214–220, Helsinki, Finland, 1981.

[2] E. Merényi. "Precision Mining" of High-Dimensional Patterns with Self-Organizing Maps: Interpretation of Hyperspectral Images. *Quo Vadis Computational Intelligence: New Trends and Approaches in Computational Intelligence*, 2000.

[3] E. Merenyi, K. Tasdemir, and L. Zhang. *Learning Highly Strutured Manifolds: Harnessing the Power of SOMs*. Springer Verlag.LNAI 5400, 2009.

[4] D. G. Covell, A. Wallqvist, A. A. Rabow, and Thanki N. Molecular classification of cancer: Unsupervised self-organizing map analysis of gene expression microarray data. *Molecular Cancer Therapeutics*, 2:317–332, March 2003.

[5] C. Pohl, M. Franzmeier, M. Porrmann, and U. Rueckert. gNBX – Reconfigurable Hardware Acceleration of Self-organizing Maps. In *Proc. IEEE Int Field-Programmable Technology Conf*, pages 97–104, 2004.

[6] D.C. Hendry and R. Cambio. Techniques for power reduction in an simd implementation of the vq/som algorithms. *Neurocomputing*, 74(1-3):291 – 300, 2010.

[7] D. DeSieno. Adding a conscience to competitive learning. In *Proc. IEEE Int Neural Networks Conf*, pages 117–124, 1988.

[8] H.-U. Bauer, R. Der, and M. Herrmann. Controlling the Magnification Factor of Self-Organizing Feature Maps. *Neural Computation*, 8(4):757–771, 1996.

[9] E. Merenyi, A. Jain, and T. Villmann. Explicit magnification control of self-organizing maps for "forbidden" data. *IEEE Trans. on Neural Networks*, 18(3):786–797, 2007.

[10] M. Porrmann, J. Hagemeyer, C. Pohl, J. Romoth, and M. Strugholtz. RAPTOR A Scalable Platform for Rapid Prototyping and FPGA-based Cluster Computing. In *Proceedings of the International Conference on Parallel Computing, ParCo2009, Symposium on Parallel Computing with FPGAs*, Lyon, France, 1 - 4 September 2009.