ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 337-342

A Reconfigurable SOM Hardware Accelerator

M. Porrmann, M. Franzmeier, H. Kalte, U. Witkowski, U. Rückert

Heinz Nixdorf Institute, System and Circuit Technology,
University of Paderborn, Germany
email: porrmann@hni.upb.de, WWW: http://wwwhni.upb.de/sct

Abstract. A dynamically reconfigurable hardware accelerator for self-organizing feature maps is presented. The system is based on the universal rapid prototyping system RAPTOR2000 that has been developed by the authors. The modular prototyping system is based on XILINX FPGAs and is capable of emulating hardware implementations with a complexity of more than 24 million system gates. RAPTOR2000 is linked to its host – a standard personal computer or workstation – via the PCI bus. For the simulation of self-organizing maps a module has been designed for the RAPTOR2000 system, that embodies an FPGA of the Xilinx Virtex series and optionally up to 128 MBytes of SDRAM. A speed-up of about 50 is achieved with five FPGA modules on the RAPTOR2000 system compared to a software implementation on a state of the art personal computer for typical applications of self-organizing maps.

1. Introduction

Self-organizing maps (SOMs) [1] are successfully used for a wide range of technical applications, in particular, dealing with noisy or incomplete data. Examples of use are controlling tasks, data analysis and pattern matching. In cooperation with an industrial partner we are using SOMs for the analysis of IC (Integrated Circuits) fabrication processes. The large amount of data, that is captured during operation, has to be analyzed in order to optimize the process and to avoid a decrease of yield [2, 3]. Software simulations of medium sized maps on state of the art workstations require calculation times from hours up to several days for these large datasets. The simulation of large maps (i.e. more than one million neurons with vector dimensions in the order of thousands) seems promissing but is not feasible with state of the art PCs or workstations. From the various possibilities to speed up neural computations we have chosen the design of a hardware accelerator for neural netwoks. Our goal is to integrate the system into state of the art workstations if very high performance is required and to enable access to the accelerator via the internet if the accelerator is only sporadically used.

In recent years various hardware implementations for different neural network architectures have been presented [4]. But many of the proposed architectures are dedicated to single neural network algorithms or groups of similar algorithms. The aim of our project is to deliver a system that is capable of accelerating a wide range of different neural algorithms. In most applications different methods of information processing are combined. For example, artificial neural networks are combined with fuzzy logic or with techniques for knowledge based information processing. In contrast to implementing different components for data pre- and postprocessing and for neural networks we use a dynamically (i.e. during runtime) configurable hardware accelerator to implement all of the algorithms that are required for a special application. The system

ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 337-342

can be reconfigured for the different tasks in one application (e.g. different configurations for pre- and postprocessing may be selected). Because of the reconfigurability the hardware can be mapped optimally to the requirements of the application, thus allowing an easy expansion by new algorithms to improve flexibility and performance.

2. Implementing Self-organizing Maps in Hardware

Self-organizing maps as proposed by Kohonen [1] use an unsupervised learning algorithm to form a nonlinear mapping from a given high dimensional input space to a lower dimensional (in most cases two-dimensional) map of neurons. Our goal is to find an efficient implementation on state of the art FPGAs that, on the one hand delivers the required high performance and, on the other hand, fits into the limited ressources of current FPGAs. Because of their internal structure FPGAs seem to be very well suited for the implementation of neural networks. Previous work of the authors concerning highly optimized ASIC implementations of self-organizing maps has emphasized, that avoiding memory bottlenecks by using on-chip memory is a must in order to achieve optimal performance [5]. We use XILINX Virtex FPGAs for our implementations, because these devices come with large internal SRAM blocks that can be used for internal weight storage.

In oder to limit the hardware requirements for the implementation, the original SOM-algorithm has been simplified. In particular the Manhattan distance is used for calculating the distance between the input vector and the model vectors to avoid multiplications as required for the euclidean distance (which is typically used in SOM implementations). The internal precision is set to 16 bit and the accuracy of the input vector components and of the model vectors is set to eight bit. Restricting the values of the neighborhood function to negative powers of two gives us the opportunity to replace the multiplications that are required for adaptation by shift operations. Of cause these simplifications do not come for free (e.g. the convergence time may increase in some cases), but it has been shown that the simplified algorithm is well suited for a lot of applications [6]. Furthermore the actual generation of Xilinx FPGAs (Virtes II) comes with integrated multipliers and our implementations on these chips will thus be able to use euclidean distance instead of manhattan distance with no loss in performance.

Data pre- and postprocessing is a crucial and often ignored aspect in neurocomputer design. The use of reconfigurable hardware enables us to implement optimally fitting hardware implementations - not only for neural networks but also for pre- and postprocessing. As an example, we have integrated the main visualization techniques for self-organizing maps, that had to be performed in software so far, into hardware. The visualization of component maps and pattern position maps is supported as well as all kind of distance matrices like the U-Matrix. Implementing these algorithms in hardware dramatically reduces communication and thus enables a more efficient utilization of the hardware accelerator [7].

Our SOM-implementation consists of processing elements that are working in SIMD-manner and that are controlled by an external controller. Nearly all calculations are performed in parallel on all processing elements. A bidirectional bus is used for data transfers to dedicated elements and for broadcasting data to groups of processor elements (or to all processor elements). Single elements and groups of elements are addressed by row and column lines that are connected to the two-dimensional matrix. The externally generated instructions are transferred to the processor elements via an additional control bus. Two more signals are used for status messages from and to the

ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 337-342

controller. The architecture is able to simulate virtual maps, i.e. it is possible to simulate maps that are larger than the array of processor elements that is implemented. Apart from the typical two dimensional grid of neurons, any other kind of network topology can be implemented (e.g. one dimensional or toroidal maps). Because the values for the adaptation factors are provided by an external controller, any adaptation function and neighborhood function may be realized with the proposed hardware (without any changes in the FPGA configuration).

3. Architecture of the Hardware Accelerator

The hardware accelerator that is presented in this paper is based on the modular rapid prototyping system RAPTOR2000. The system consists of a motherboard and up to six application specific modules (ASMs). Basically, the motherboard provides the communication infrastructure between the ASMs and links the RAPTOR2000 system via PCI bus to a host computer. Additionally, management functions like bus arbitration, memory management and error detection services are integrated in two Complex Programmable Logic Devices (CPLD). The various communication schemes that can be used between different ASMs and between the ASMs and the host computer are depicted in the block diagram in figure 1. Every ASM slot is connected to the *Local Bus* for internal communication with other devices or ASMs and for external communication with the host processor or with other PCI bus devices. An additional *Broadcast Bus* can be used for simultaneous communication between the ASMs. Additionally, a dual port SRAM can be accessed by all ASMs via the *Broadcast Bus* (e.g. utilized as a buffer for fast direct memory accesses to the main memory of the host system). Direct communication between adjacent ASMs is realized by 128 signals that can be variably used, depending on the actual implementation.

A crucial aspect concerning FPGA designs is the configuration of the devices. Each ASM that carries an FPGA has to be configured by an application specific data stream that determines the function of the device. In order to utilize dynamic reconfiguration (i.e. during runtime) it is necessary to minimize this reconfiguration time, therefor the configuration algorithms have been implemented in hardware. Reconfiguration of an ASM can be started by the host computer, another PCI bus device or by another ASM.

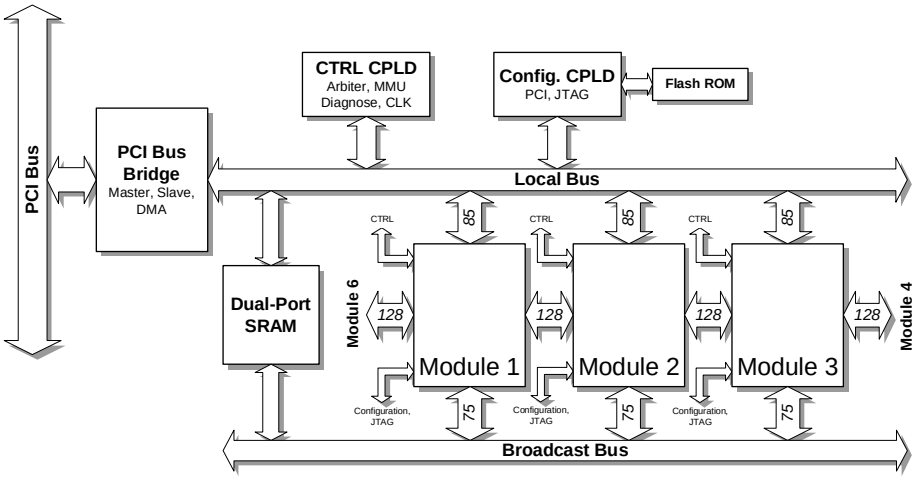

Figure 1. Architecture of the Raptor2000 rapid prototyping system

ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 337-342

Thus, it is possible that an FPGA autonomously reconfigures itself by configuration data that is located anywhere in the system. Due to the hardware implementation of the reconfiguration algorithm, a Xilinx Virtex 1000 FPGA can be completely reconfigured within less than 20 ms. The configuration algorithm implemented into the hardware also supports the partial reconfiguration of the system [8].

For the simulation of self-organizing feature maps the module DB-VS has been designed for the RAPTOR2000 system. This ASM embodies an FPGA of the Xilinx Virtex (-E) series and optionally up to 128 MBytes of SDRAM. The ASM can be equipped with various chips, that emulate circuits with a complexity of 400,000 to 4 million system gates. The SDRAM controller is integrated into the FPGA logic. A photo of the RAPTOR2000 system with two DB-VS modules is shown in figure 2. In the context of this paper we focus on the implementation of self-organizing feature maps on RAPTOR2000. Because of the flexibility of the system many other neural and conventional algorithms may be mapped to the system. As another example for neural networks we have analyzed the implementation of neural associative memories on the RAPTOR2000 system [9]. Another case study focuses on the implementation of octree based 3D graphics [10].

4. SOM-Implementation on RAPTOR2000

In order to implement the SOM-algorithm on the RAPTOR2000 rapid prototyping system, five DB-VS modules are applied. Four ASMs are used to implement a matrix of processing elements while the fifth is used to implement the matrix controller, an I/O controller for the connection to the local bus of the RAPTOR2000 system and a dual port SRAM that is used to buffer input- and output-data. An integrated SDRAM interface controls the external 128 MBytes of SDRAM. The dual port SRAM is used to store single input vectors, commands from the host computer and the results (e.g. best match positions or postprocessed visualizations of the map). The large SDRAM is used to store one or more input data sets. During learning of a self-organizing map, the

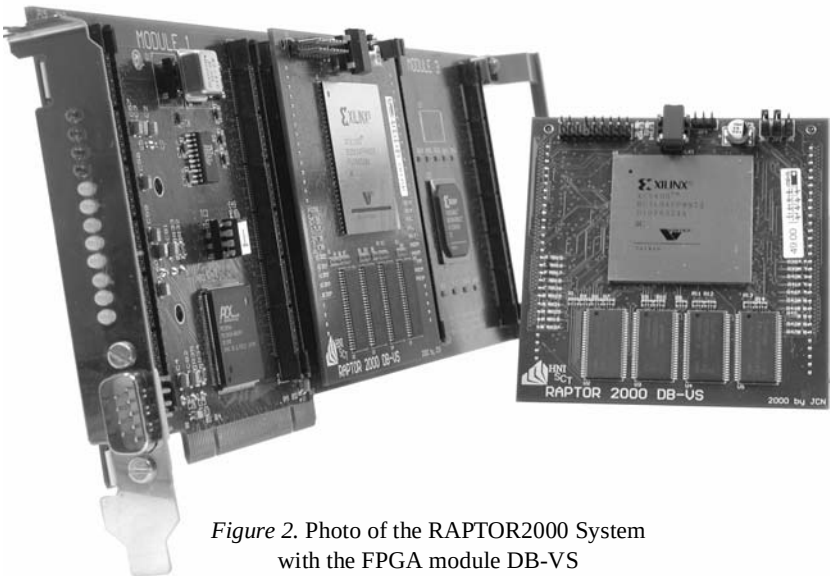

Figure 2. Photo of the RAPTOR2000 System
with the FPGA module DB-VS

ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 337-342

whole input data set has to be presented to the map for several times. Thus it is recommendable to transfer the whole data set in one fast block transfer via the PCI bus to the hardware accelerator in order to minimize the load of the PCI bus.

The design has been described in VHDL and synthesized using the Synopsys FPGA compiler and the Xilinx Alliance tools for place and route. Using Virtex XCV812E-6 devices, N_{PE}=81 processing elements can be implemented, each equipped with 2 kBytes of internal SRAM. The utilization of the FPGAs is about 68%. The numbers of clock cycles that are required for recall (c_{recall}) and for learning (c_{adapt}) of an input vector with a dimension of l are:

$$\begin{aligned} c_{recall} &= n_v \cdot (l + 2 \cdot \lceil \mathrm{ld}(l \cdot 255) \rceil + 4) \\ c_{adapt} &= n_v \cdot (2l + 2 \cdot \lceil \mathrm{ld}(l \cdot 255) \rceil + 12) \end{aligned} \quad (1)$$

The variable n_v is the number of neurons, that is emulated by one processing element. Using Virtex XCV812E-6 devices, a clock frequency of 65 MHz has been achieved for the FPGAs. The maximum performance is achieved, if every processing element represents one neuron (n_v=1). In this case about 17,500 MCPS (Million Connections per Second) can be achieved during recall and 2000 MCUPS during learning. Simulating larger maps, the performance decreases. Using a benchmark data set with a vector dimension of l=9, maps with up to 250×250 Neurons can be simulated with a performance of 4,400 MCPS and 665 MCUPS, respectively. With a software implementation on a state of the art personal computer (AMD Athlon, 1 GHz) only a performance of 85 MCPS and 22 MCUPS can be achieved for this problem. Apart from the described Xilinx XCV812E devices, other FPGAs with BG560 package may be used on DB-VS. Xilinx Virtex devices may be used as well as Virtex E devices. The design has been synthesized to different Xilinx devices leading to a maximum performance of more than 50 GCPS and 5,7 GCUPS on Xilinx XCV3200E-6 devices. The dynamically reconfiguration that is provided by the RAPTOR2000 system can be used to adapt the size of the map that is simulated in order to achieve optimal performance. Additionally, dynamic reconfiguration is used to adapt the precision of the processing elements. Starting with a low precision (e.g. 8 bit), a rough ordering of the map can be achieved. For fine tuning of the map the precision of the processing elements is increased (e.g. to 16 bit), by loading a new configuration file.

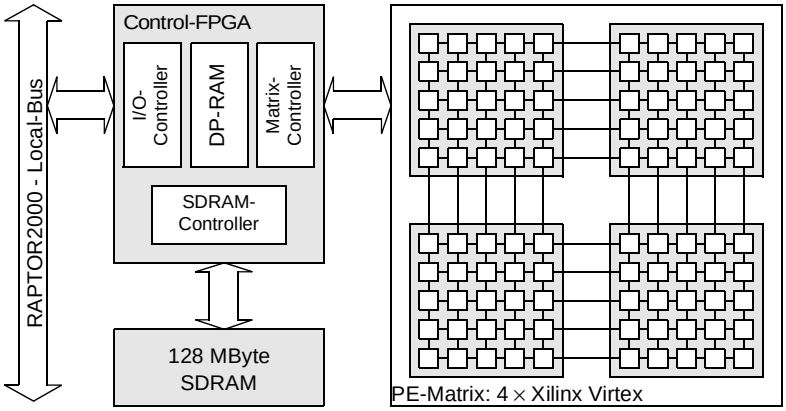

Figure 3. Architecture of the hardware accelerator

ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 337-342

5. Conclusion

A dynamically reconfigurable hardware accelerator for the simulation of self-organizing feature maps has been presented. Equipped with five FPGA modules, the system achieves a maximum performance of more than 50 GCPS (Giga Connections per Second) during recall and more than 5 GCUPS (Giga Connection Updates per Second) during learning. Even higher performance numbers can be achieved by using new FPGA architectures like the Xilinx Virtex-E series. Apart from the high performance the system is capable of doing pre- and postprocessing tasks – either by use of the implemented visualization features or by dynamically reconfiguring the devices during runtime. The latter is supported by the RAPTOR2000 rapid prototyping system by means of the ability to reconfigure the FPGAs very fast via the PCI bus.

Acknowledgement

This work has been partly supported by the Deutsche Forschungsgemeinschaft (German Research Council) DFG Graduate Center "Parallele Rechnernetzwerke in der Produktionstechnik" and by the Robert Bosch GmbH, Automotive Equipment Division 8, Reutlingen, Germany.

References

[1] Kohonen, T.: "Self-Organizing Maps", Springer-Verlag, Berlin, 1995.

[2] Goser, K., Marks, K.-M., Rückert, U., "Selbstorganisierende Parameterkarten zur Prozeßüberwachung und -voraussage", Informatik-Fachberichte Nr. 227, 1989, pp. 225-237, Springer, München.

[3] Rüping, S., Müller, J.: "Analysis of IC Fabrication Processing using Self-Organizing Maps", Proc. of ICANN´99, Edinburgh, 7.-10. Sept. 1999, S. 631-636.

[4] Rückert, U.: "ULSI Implementations for Artificial Neural Networks", 9th Euromicro Workshop on Parallel and Distr. Processing 2001, Feb. 7-9, 2001, Mantova, Italien, S.436-442.

[5] Porrmann, M., Rüping, S., Rückert, U.: "The Impact of Communication on Hardware Accelerators for Neural Networks", Proc. of SCI 2001 Orlando, Floriada USA, 22.-25. Juli 2001, pp. 248-253.

[6] Rüping, S., Porrmann, M., Rückert, U., "SOM Accelerator System", Neurocomputing 21, pp. 31-50, 1998 .

[7] Porrmann, M., Rüping, S., Rückert, U., "SOM Hardware with Acceleration Module for Graphical Representation of the Learning Process", Proc. of the 7th Int. Conference on Microelectronics for Neural, Fuzzy and Bio-Inspired Systems, pp. 380-386, Granada, 1999.

[8] Porrmann, M., Kalte, H., Witkowski, U., Niemann, J.-C., Rückert, U.: "A Dynamically Reconfigurable Hardware Accelerator for Self-Organizing Feature Maps", Proc. of SCI 2001 Orlando, Florida USA, 22.-25. Juli, 2001, pp. 242-247.

[9] Porrmann, M., Witkowski, U., Kalte, H., Rückert, U.: "Implementation of Artificial Neural Networks on a Reconfigurable Hardware Accelerator", 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing (PDP 2002), 9.-11. Januar 2002, Gran Canaria Island, Spain, to be published.

[10] Kalte, H., Porrmann, M., Rückert, U., "Using a Dynamically Reconfigurable System to Accelerate Octree Based 3D Graphics", PDPTA'2000, June 26-29, 2000 Monte Carlo Resort, Las Vegas, Nevada, USA, pp. 2819-2824