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.

Parallel Neural Hardware: The Time is Right

Ulrich Rückert[1] and Erzsébet Merényi[2] *

1- Bielefeld University, Cognitive Interaction Technology - Center of Excellence
Bielefeld, Germany
2- Rice University - Department of Statistics
Houston, Texas, U.S.A.

Abstract. It seems obvious that the massively parallel computations inherent in artificial neural networks (ANNs) can only be realized by massively parallel hardware. However, the vast majority of the many ANN applications simulate their ANNs on sequential computers which, in turn, are not resource-efficient. The increasing availability of parallel standard hardware such as FPGAs, graphics processors, and multi-core processors offers new scopes and challenges in respect to resource-efficiency and real-time applications of ANNs. Within this paper we will discuss some key issues for parallel ANN implementation on these standard devices compared to special purpose ANN implementations.

1 Introduction

The implementation of artificial neural networks (ANNs) was mainly technology driven in the past. In the 1960s the transistor replaced the electronic tube and small discrete electronic components came up on the market. Researchers like Karl Steinbuch [1] in Germany or Bernard Widrow [2] in the United States used these devices in their construction of electronic ANN implementations with a low number of neurons. Computers for simulating ANNs were not widely available at that time; hence building ANNs out of electronic components was a first approach to study functional principles and dynamics of small artificial neuron groups.

Realizing ANNs with discrete electronic devices was tedious and error prone. Furthermore, it was expensive and space consuming to scale up the size of the ANN. Hence, with the increasing availability of computers and especially personal computers (PCs) software simulations of ANNs were the better choice. Software simulations offer a high flexibility but do not exploit the spatio-temporal parallelism that is inherent in biological neural networks. Hence, especially for larger ANNs with hundreds of neurons the simulation time was quite long in the early days of PCs. Furthermore, real-time processing in practical applications was not feasible at all.

In the late 1980s, the revolutionary progress of microelectronics had reached feature sizes of one micrometer (Figure 1) and became the driving force behind the constant development of new technical products that have markedly improved functionality and higher performance, yet at a lower cost. By this time, Moore´s law gathered momentum, the first independent fabless companies were launched, and the computer-aided design (CAD) automation industry was born. An affordable way to personalized integrated circuit implementations was established even for small design

* This research is partly supported by the German Science Foundation (DFG), 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.

teams from industry as well as from academia. These new and fascinating opportunities motivated intensive efforts to develop ANN chips and neurocomputers for parallel ANN implementation [3,4]. The European conference on "Microelectronics for Neural Networks" (MicroNeuro [5]) emerged as the only international forum specifically devoted to all aspects of implementing ANNs in hardware (1990 Dortmund, Germany; 1991 Munich, Germany; 1993 Edinburgh, Scotland; 1994 Torino, Italy; 1996 Lausanne, Switzerland; 1997 Dresden, Germany; 1999 Granada, Spain). Even hardware products appeared on the market - from both small businesses and large companies. All these impressive approaches had a real problem trying to keep up with the effects of Moore's law coming into full swing, as microprocessors, digital signal processors, and field-programmable gate-arrays (FPGAs) all grew faster and faster. The fabless design teams had only access to technologies one generation or two behind the semiconductor companies, who also could afford mass production pricing.

Since 2005, the performance increase of microprocessors slowed down and the trend to multi-core architectures started. Furthermore, GPUs (graphics processing units) became widely available and the complexity of state-of-the-art FPGAs allowed system-on-chip designs. These off-the-shelf devices offer new perspectives for massively parallel ANN implementation. In the following, we will discuss some key issues for realizing ANNs on these standard devices compared to special purpose implementations dedicated to a specific ANN model.

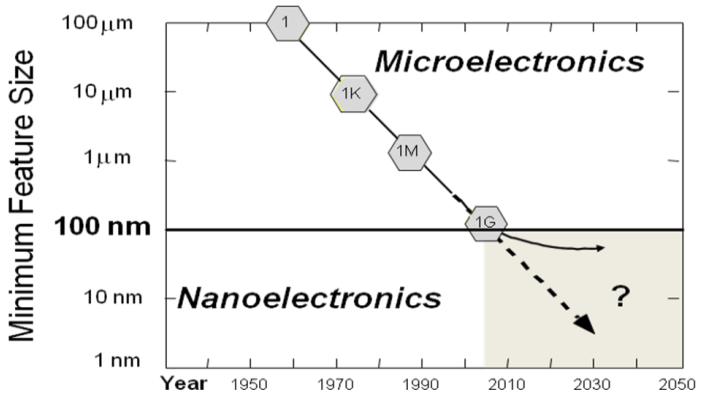

Fig. 1: Decreasing feature sizes of integrated circuits over time, and the increasing number of devices in an unit area (shown in the hexagonal symbols)

2 Field-Programmable Gate Arrays (FPGAs)

FPGAs have a modular and regular architecture containing mainly programmable logic blocks, embedded memory, and a hierarchy of reconfigurable interconnects for wiring the logic blocks. Furthermore, they may contain digital signal processing blocks and embedded processor cores. After manufacturing they can be configured before and during runtime by the customer. Today, system-on-chip designs with a complexity of about a billion logic gates and several Mega-Bytes of internal SRAM

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.

memory can be mapped on state-of-the-art FPGAs. Clock rates approach the GHz range boosting the chip-computational power in the order of GOPS (billion operations per second) at a power consumption of several watts. Hence, FPGAs offer an interesting alternative for parallel implementation of ANNs providing a high degree of flexibility and a minimal time to market. The time for the development of FPGA and application specific integrated circuit (ASIC) designs is comparable. A big advantage of FPGAs is that no time for fabrication is needed. A new design can be tested directly after synthesis for which efficient CAD tools are available. A disadvantage of FPGAs is the slower speed, bigger area, and higher power consumption compared to ASICs. Compared to software implementations FPGAs offer a higher and more specialized degree of parallelization.

The implementation of ANNs on a reconfigurable hardware makes it possible to realize powerful designs that are optimized for dedicated algorithms [6]. Another great advantage is the feature of reconfigurability that enables the change to a more efficient algorithm whenever possible. If, at the beginning of the training of an ANN, a low data precision is satisfying, we are able to implement a highly parallel implementation to get a rough order of the network. Using a lower precision allows us to set up an optimized architecture that can be faster, smaller or more energy efficient than a high precision architecture. For a fine-tuning of the ANN, the FPGA can be reconfigured to implement high-precision elements. Additionally, we are able to adapt the implemented algorithms to the network size that is required for a certain problem. Thus we can always use the most suitable algorithms.

Furthermore, dynamic (or runtime) reconfiguration enables to change the implementation on the FPGA during runtime. Dynamic reconfiguration is used to execute different algorithms on the same resources. Thus, limited hardware resources can be used to implement a wide range of different algorithms. In the field of ANN hardware, reconfiguration can be used, e.g., to implement algorithms with variable precision or to implement heterogeneous architectures with different ANN types. In ANN simulation we are often interested in providing as much computing power as possible to the simulation of the algorithm. But pre- and post-processing of the input and output data often also requires quite a lot of calculations. In this case dynamic reconfiguration offers the opportunity to implement special pre-processing algorithms in the beginning, switch to the ANN simulation and in the end reconfigure the system for post-processing. Thus, we do not require the system resources that would be necessary to calculate all algorithms in parallel [7].

3 Graphics processing units (GPUs)

A GPU is a specialized integrated circuit designed to rapidly process floating point-intensive calculations, related to graphics and rendering at interactive frame rates. The rapid evolution of GPU architectures from a configurable graphics processor to a programmable massively parallel co-processor make them an attractive computing platform for graphics as well as other high performance computing having substantial inherent parallelism such as ANNs. The demand for faster and higher definition graphics continues to drive the development of increasingly parallel GPUs with more than 1000 processing cores and larger embedded memory. At the same time, the architecture of GPUs will evolve to further increase the range of other applications.

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.

GPUs are well suited for single instruction and multiple data (SIMD) parallel processing. In order to assist the programmers specialized programming systems for GPUs evolved (e.g., CUDA [8]) enabling the development of highly scalable parallel programs that can run across tens of thousands of concurrent threads and hundreds of processor cores. However, even with these programming systems the design of efficient parallel algorithms on GPUs for other applications than graphics is not straight-forward. Significant re-structuring of the algorithms is required in order to achieve high performance on GPUs. Furthermore, it is difficult to feed the GPUs fast enough with data to keep them busy. Nevertheless, the increasing number of papers on this topic shows that GPUs are an interesting implementation platform for simulating large ANNs [9].

4 Many-core processors

A multi-core processor is a single computing component with two or more conventional uniprocessors (called cores). A many-core processor is a multi-core processor with a considerably higher number of cores (e.g. more than 100). The promise of parallelism has fascinated researchers for at least three decades. In the past, parallel computing efforts have shown promise and gathered investment, but in the end, uniprocessor computing always prevailed. Nevertheless, general-purpose computing is taking an irreversible step toward parallel architectures because single-threaded uniprocessor performance is no longer scaling at historic rates. Hence, parallelism is required to increase the performance of demanding applications. Since real world applications are naturally parallel and hardware is naturally parallel, the missing links are programming models and system software supporting these evolving massively parallel computing architectures. Furthermore, there is no clear consensus about the right balance of computing power, memory capacity, and internal as well external communication bandwidth of integrated many-core architectures.

ANNs are inherently parallel and hence, it is obvious that many-core processors are an attractive implementation platform for them. Various techniques for simulating large ANNs on parallel supercomputers or computer networks are known which can be reused for mapping ANNs to many-core architectures. Furthermore, many-core processors can be embedded in mobile devices such as robots or smart phones opening up new application vistas for ANNs. Consequently, the number of ANN many-core implementation is increasing [10].

5 Conclusion

Different approaches are known for supporting ANNs on parallel computing architectures: general-purpose parallel architectures (e.g. many-cores, GPUs) for emulating a wide range of ANN models, reconfigurable implementations on FPGAs, and ASICs dedicated to a specific ANN model. While general-purpose architectures offer an unrivalled flexibility and a high degree of observability to the inner states and dynamics of neural algorithms, special-purpose (neuromorphic) designs offer resource-efficiency in respect to speed, compactness and power consumption. FPGA implementations offer a compromise between both extremes (Figure 2).

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.

State-of-the-art CMOS technologies are able to integrate billions of nanoelectronic devices on a single chip with an area of a few cm². A strong growth of FPGA complexity and in the number of cores per GPU and many-core processor is expected in the future. Even more computational power may be obtained by emerging technologies like quantum computing, molecular electronics, or novel nano-scale devices (memristor, spintronics, nanotubes (CMOL)), but these technologies will not be available on broad basis in the next decade. With structure sizes smaller than 0.1 micron, nanoelectronics start falling below the level of biological structures forming the brain. However, the brain efficiently uses all three dimensions, whereas nanoelectronics mainly use only the two physical dimensions of the silicon die surface. Nevertheless, on an area of one square millimeter - roughly the square dimension of a Purkinje cell (a type of neuron) in the cerebellar cortex, shown in Figure 3 - we can integrate a digital artificial neuron with about 70,000 16-bit weights (synapses) and a 32-bit microprocessor as a neural processing unit (Figure 3).

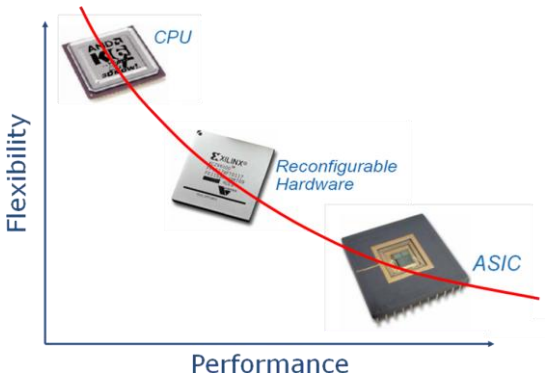

Fig. 2: Qualitative performance and flexibility grading of hardware platforms

The challenge lies in mastering the resulting design complexity and achieving economic viability for integrated systems with more than a billion devices per square centimeter. This requires system concepts that both exhaust the possibilities of future technologies and reduce the design and test complexity. Due to their highly regular and modular structure, inherent fault tolerance, and learning ability, ANNs offer an attractive alternative for ultra-large-scale-integration.

Despite the impressive development of nanoelectronics during the last decades, there is still no clear consensus on how to exploit this technological potential for massively parallel ANN implementations. It is currently quite difficult to determine the best way to perform ANN calculations for any given application. This is one reason for the huge variety of approaches to ANN hardware implementation known in literature. The problem of benchmarking and an adequate metric for performance evaluation is still open, too. So the discussion is open about the best way to achieve very large neural systems and, in the long term, how to produce so-called artificial brains. We are still a long way from fully comprehending the functional mechanisms of the brain; and the construction of an artificial brain will remain for a very long time, if not forever, a fantasy. Nevertheless, we do have something to learn from nature about resource-efficient technical systems. This is why a hardware

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.

realization of neural networks does not aim for an exact reproduction of nervous systems, but simply for an efficient use of available technologies for solving practical problems. The papers selected for the ESANN special session on "Parallel hardware architectures for acceleration of neural network computation" present interesting new results on different aspects of parallel ANN hardware implementation and the benefits for practical applications.

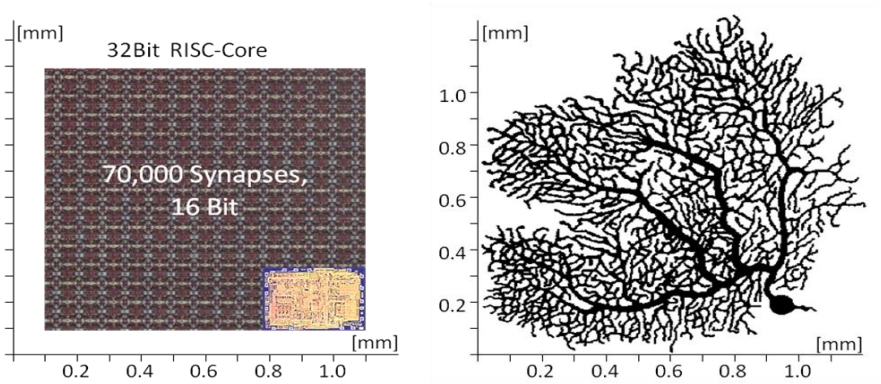

Fig. 3: Area comparison of a digital neuron in 100nm standard CMOS technology and a biological neuron (Purkinje cell).

References

[1] K. Steinbuch, Adaptive networks using learning matrices, Kybernetik 2, pages 148-152, 1965.

[2] B. Widrow, Pattern Recognition and Adaptive Control, IEEE Transactions on Applications and Industry, 83(74), pages 269-277, 1964.

[3] M. Verleysen, B. Sirletti, A. Vandemeulebroeke, and P. Jespers, Neural Networks for high-storage content-addressable memory: VLSI circuit and learning algorithm. IEEE Journal of Solid-State Circuits, vol.24, no. 3, 1989.

[4] D. Hammerstrom and N. Nguyen, System Design for a Second Generation Neurocomputer. Proceedings of the IJCNN, pagers II 80-83, 1990.

[5] U. Ramacher and U. Rückert, eds., VLSI Design of Neural Networks. Kluwer Academic, Boston, 1991.

[6] A.R. Omondi, J.C. Rajapakse, editors, FPGA Implementations of Neural Networks, Springer-Verlag, 2005.

[7] Porrmann, M.; U. Witkowski, U. Rückert, Implementation of Self-Organizing Feature Maps in Reconfigurable Hardware. In Omondi, Amos; Rajapakse, Jagath, editors, FPGA Implementations of Neural Networks, Springer-Verlag, pp. 253-276, 2005.

[8] M. Gerland et al. Parallel Computing Experiences with CUDA. IEEE Micro, vol. 28, no. 4, pages 13-27, 2008.

[9] K.S. Oh and K. Jung. GPU implementation of neural networks. Pattern Recognition, 37(6), pages 1311-1314, Elsevier, 2004

[10] L. A. Plana et al. SpiNNaker: Design and Implementation of a GALS Multi-Core System-on-Chip. ACM Journal on Emerging Technologies in Computing Systems vol. 7, no. 4, pages 17:1-17:18, 2011.