ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume 19, Number 4, 2016, 360–368

Cellular Automata Hardware Implementations - an Overview

Monica DASCĂLU

Politehnica University of Bucharest, Romania and Center for New Electronic Architectures, Romanian Academy, Bucharest, Romania
Email: monica.dascalu@upb.ro

Abstract. Cellular automata model grew very popular decades ago with software applications like the Game of Life. From the computing theory's perspective, it is a fundamental model of massive parallelism and was believed to offer extremely efficient solutions for hardware implementations. This paper overviews scientific literature and presents existing implementations of cellular automata, either in general form or dedicated to specific applications. The model is still considered to be very interesting and was proved to be efficient for applications like modeling of complex phenomena and implementation of encryption systems and randomizers. However, the hardware versions of cellular automata are still belonging to the academic realm, as they did not reach the industry and are not available as commercial products.

1 Introduction

Cellular automata offer a very elegant computing model: a (typically) homogeneous regular network of simple processing elements, also named cells, locally interconnected. It is derived from the model of natural systems consisting of many simple parts that interact only locally and exhibit complex and self-organizing global behavior.

As digital system's, cellular automata are networks of finite automata (finite state machines). Therefore, the model seem to be very appropriate for hardware implementation as ASIC digital integrated circuits. This paper overviews some hardware implementations of cellular automata reported in the scientific literature of the last two decades, in order to estimate the development of this field, that was considered to be so promising in the 1980's [1].

The cells have a finite set of possible states, usually small. The computing task is realized through the evolution of the global state, and this global evolution is the consequence of local evolution laws. The time is discrete in such systems, and the local evolution laws include in their definition the local interactions between cells, giving the next state value for each cell depending on present state of its neighboring cells in the lattice. Cellular automata model was introduced by J. von Neumann and S. Ulam in the early 1950 as a general framework for modeling complex structures capable of self-reproduction and self-repair functions. Von Neumann, the same scientist that proposed

the classical, centralized and sequential computing model (nowadays denoted by his name), introduced also this totally different computing model and structure, completely decentralized and massively parallel.

The primary research of J. von Neumann in cellular automata was oriented towards artificial self-reproduction of structures. He derived a computational universal cellular space with self-reproduction configurations. A cellular automata with 29 states per cell realized this function by growing an arm that could construct a new configuration similar to the initial one. Further, the model was used to emulate different computing structures and to perform various computing tasks and was proved to be a universal computer [2]. Different researchers continued to improve von Neumanns demonstration and model, simplifying the set of rules, reducing the number of states per cell or introducing simpler ways to compute several tasks.

Another famous cellular automata model, probably the most famous cellular automata, is J. Conway's Game of Life, introduced in the 1970's but still fascinating researchers with its possibilities [3]. This improperly called Game - because the player simply watch the evolution of the system - models a population of living cells developing in a cellular space under very simple basic laws of evolution. In the original version, the cells have only two states, therefore the model is much simpler than von Neumann's, but the global behavior is very vast and also was proved to be capable of universal computation and self-reproduction. These two examples are enough to sustain the idea that cellular automata had a great influence in the emergence and development of the entire field of artificial life or a-life.

In a first period of development, cellular automata based models were proposed for different computing tasks, such as parallel language recognition, modeling of biological systems, image processing. Theoretical studies in the field of cellular automata became significant only in the 1980's, with Wolfram's works [4]. Wolfram's investigated cellular automata as mathematical models for self-organizing statistical systems and introduced a phenomenological classification of cellular automata, based on the complexity of their global behavior.

In the VLSI era, cellular automata field has been consolidated, since cellular automata offer a parallel computing model almost perfectly adapted to the VLSI design. Designers always look for simple, regular and modular logic circuit structure to realize a complex function and cellular automata ideally meet all these requests. ASIC implementations of cellular automata based pattern generators, cryptographic systems, associative memories, image processing circuits etc. were proposed [5], [6].

Nowadays, the cellular automata community of researchers is still small, especially when compared to the much greater interest raised by other nature-inspired models like neural networks or genetic algorithms. However, good results and useful applications are developed every year. Three major trends of development may be identified for the future of cellular automata field [7]:

- fundamental research in complexity studies;
- application development, especially oriented towards ASIC or FPGA implementations, and parallel computing systems;
- modeling and simulation of complex natural or artificial systems.

2 Cellular Automata: Self-Organization and Parallelism

Self-organization is one of the modern scientific concepts, like complexity or chaos, that try to cover a vast frame of phenomena and natural realities that are less understood. It refers to the property of the natural systems to manifest a coherent evolution and a certain order and organization even in the absence of something that impose this order. Scientists suppose that the intimate microscopic processes, together with the property of self-organization, lead to a spatial-temporal macroscopic order.

In the case of cellular automata, the self-organization is usually illustrated by the patterns generated by a linear or two-dimensional lattice of cells that evolve according to simple rules but generate exquisite patterns. The behavior generated by a simple, regular structure of simple processing elements appear to be complex.

The cellular automata are defined by the interconnection network and the specificity of the cell (number of states and local evolution rule). It is obvious that the parallelism is the main feature of such a computing system and it is even more attractive as proven to be able of universal computation. The problem is that there is no method that tells us how to apply the model to perform a specific computing task.

This is the main dilemma of the massive parallelism: to decompose a complicate task in one and only elementary function executed in parallel by the processing elements. For cellular automata, the processing elements are finite state machines (i.e., perform just simple functions). The problem of synthesis has no general solution and is probably the reason that cellular automata are still used for just few computing tasks. The implementation of the model in the general version is therefore not so interesting as a computing system, but rather as an application development tool.

Only few prototypes of cellular automata machines or co-processors were developed so far and are not available as commercial products. Hence, the paradox of cellular automata: they are essentially parallel, but mostly used in simulations. An advantage of simulation is that it allows experiments with different parameters and local rules, in order obtain an useful application of model a certain process. Speeding up simulation is another direction of scientific research in the field of cellular automata.

3 Hardware for General Cellular Automata

The hardware version of cellular automata should consist of a network of finite state machines with a predefined maximum number of states (related to the dimension of the state register) and a programmable logic circuit, with number of inputs depending of the interconnection topology. The neighborhood is the set of cells that are implied in the computation of the next state.

A universal cellular automata chip should be able to:

- implement any local rule;
- implement inhomogeneous topologies having inhomogeneous neighborhoods and inhomogeneous local rules;
- define various boundary conditions (the theoretical model is infinite).

The implementation of any local law implies the assumption that, sometimes, each local law will be used or applied, or in other words, that all local rules have some applications. The number of all existing local rules depends exponentially on two

parameters: the number of state bits and the dimension of neighborhood. Are all these laws useful for some applications? Obviously, not.

Two-dimensional rectangular network seem to be appropriate for hardware version, since it may be easily transformed through topological transformations in any particular version of interconnection: hexagonal, triangular, rectangular with 4 neighbors or 8 neighbors and also linear.

Theoretically, binary cellular automata offer a universal computing model. Any computation that can be done with a multi-state cellular automata can certainly be realized with binary automata, too [2], [4]. Also, binary cellular automata seem to be feasible in hardware version. For accuracy, we have to mention that there is no algorithm that can transform the multi-state automata in its binary correspondent. Most applications developed for multi-state automata exploit their features to obtain elegant computing algorithms.

However, there are enough applications were binary automata offer the most appropriate implementation tool, like binary image processing or cryptography [1], [6]. A binary cellular automata chip can find its own utility for such applications.

General cellular automata chips were proposed in the scientific literature, but not implemented so far in this version. The moment when these projects appeared was followed by a fast development of the FPGA technology. Reconfigurable hardware was preferred to the direct ASIC transposition of the model.

4 Programmable Cellular Automata Hardware in Reconfigurable Hardware

An interesting approach to cellular automata implementation on reconfigurable hardware like FPGA circuits and boards combines the use of a transputer processor with an early generation Xilinx product [8]. The transputer (T800) is programmed in Occam language, that was specifically created for transputers and generates the required data for programming the reconfigurable device (Xilinx XC3090). The software description of a particular cellular automata is compiled and transformed in specific place and routing data for the XC3090 chip. As presented in [8] the board was actually implemented and operational at the Balliol College, Oxford.

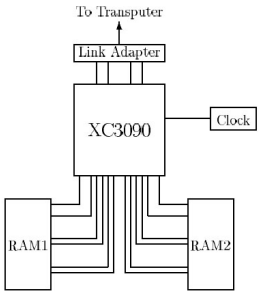

Fig. 1: The prototype board components of transputer-FPGA implementation [8], page 6.

A general implementation of cellular automata should be adaptable to particular local rules, and such implementations are called programmable cellular automata, as

opposed to particular implementations for a given local rule, that are much easier to implement.

Recent solutions for general cellular automata implementation on FPGA use last generation FPGAs, that include microprocessor cores and high speed data transfer. For instance, a project successfully realized at the University of Porto (Portugal) use also Xilinx products (Spartan 6 family) and the software microprocessor core provided by Xilinx (micro Blaze) to implement a cellular automata programmable board [9]. The system generates the FPGA configuration for a specific cellular automata implementation - the local rules and initial state are introduced in a Java application on the host computer. It was tested for particular cases like Game of Life, for small size two-dimensional cellular automata. According to the authors, for a 56x56 array of cells, the Game of Life hardware implementation offers a speed-up of 168 as compared to software simulation. Other applications verified by the authors are Greenberg-Hastings and lattice gas automata [9]. The system architecture is presented in Figure 2 .

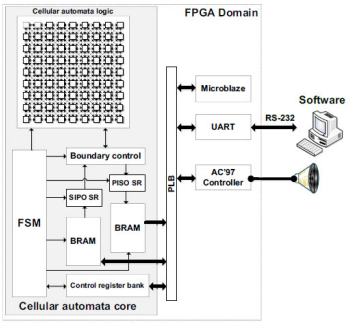

Fig. 2: The experimental board that generates FPGA implementations of cellular automata [9], page 53.

5 Cellular Automata Machines

Another approach for general cellular automata hardware is to use serial processing of the cells, combined with pipelining techniques. In this approach, standard memory modules are used to hold the cell contents (state). The local laws for cell's state updating can be realized as a look-up table or using other processing elements, such as signal processors.

The serial architecture suggested above is the one applied in the cellular automata machines developed so far: CAM and CEPRA. Both systems contain as a key component a memory manager capable of providing the processing element with the input very quickly. This memory manager needs to have internal shift registers to keep several (depending on the neighborhood) rows of cells accessible without need to read the cell memory multiple times.

Figure 3 illustrates the operating principle of such architecture. Cells states are read sequentially from Cell Memory 1 into the shift register. Once the shift registered are filled, the data from one cell and its neighbors is fed into the rule processor which computes the value of the next state of the cell. This value is written into Cell Memory 2, while another cell is read from memory 1 and shifted into the shift register. Then the next cell is ready to be processed. At the end of one sweep, Memory 1 and Memory 2

are exchanged in the algorithm. Values are read from Memory 2, processed and written in Memory 1. In this architecture the speed of processing is limited by the rate with which cells can be read from and written to memory, and the rate at which they are processed by the rule processor.

The CAM (Cellular Automata Machine) family of special purpose machines have been developed by Toffoli, Margolus and co-workers at the Massachusetts Institute of Technologys Information Mechanics Group. The concept was launched in 1987 [10] and further developed with different models. The CAM-6 was produced commercially (as a PC board) and CAM-8, the latest version [11], has been produced in small series in 1996.

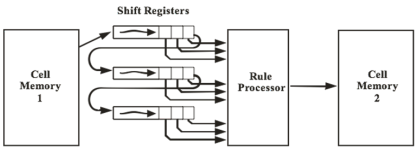

Fig. 3: Block diagram of special cellular automata hardware simulators, such as CAM-6 and CEPRA.

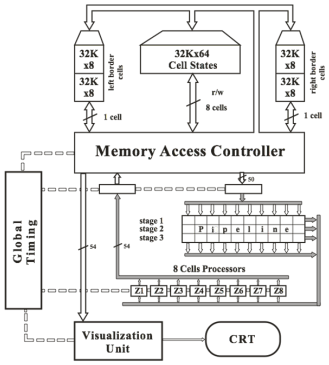

Fig. 4: Block diagram of the CEPRA - 8 [13]

The CAM architecture match most closely the concept of lattice gas cellular automata [11], where the same sequence of propagation of bits (particles) and local interaction (collision) is used. But the architecture can also efficiently simulate other cellular automata, the most severe restriction being the size of the look-up table of 16 bits. However, many CA functions can be simulated with a short sequence of 16 bit functions. For this purpose it is possible to change the look-up table between two updating steps without time delay. The machine can also be configured in an arbitrary way as a d-dimensional hyper-torus, with a maximum of 23 dimensions, and where all dimensions beyond 3 must fit into one module.

MIT's CAM project was followed by the Brain Machine project, with the CAM-Brain Machine (CBM), a complex machine that combines the model of cellular automata with neural networks [12]. While CAM-8 was meant as a cellular automata accelerator, CBM has been constructed for simulation of neural nets, although it has a CA architecture and claimed to be the hardware for the first artificial intellect.

CEPRA stands for Cellular Processing Architectures. Several prototypes were realized in the 1990's at Technical University of Darmstadt, Germany. Most of those

machines are based on programmable technology and were implemented with Altera FPGAs, but also models with DSP were developed. The authors (see for instance, [13], [14]) proved that the FPGA implementation of the CEPRA machine is much more effective than the simulation version of cellular automata. The main difference between CEPRA and the CAM machines is the structure of the rule processor that is here made of 8 FPGAs.

The block diagram of CEPRA- 8L is given in figure 4. Cell states are moved through the stages of the compute-window. Thus the 8 cell processors have direct access to the states of their neighbors. The machine computes 20-30 generations per second.

The CEPRA project was continued in the first decade of this century, with most significant new development, the CEPTRA - 1X, a configurable coprocessor for speeding the computation needed in cellular automata applications [15]. A special programming language (CDL) was developed for the use this coprocessor, that allow the description of complex cellular automata.

6 Application Specific Hardware with Cellular Automata

Specific implementations are reported in the scientific literature as hardware solutions dedicated to application (not as general computing machines). Most of these examples are also implemented in reconfigurable hardware as FPGAs.

Cellular automata applications in hardware versions most encountered in the scientific literature are applications that include randomizers (hence, cellular automata are used as generators and the synthesis problem is eluded). Such applications are randomizers themselves, Built-In Self-Test circuitry, encryption systems. Variations of the model were explored in order to obtain better performances of the cellular automata randomizers, like a global feed-back loop [22], [23].

Chauduri et al [5] proposed general schemes for an associative memory and several other interesting applications, with no details of practical implementations. In [6], the authors propose electronic digital schemes for image processing, BIST signal generation and encryption.

Implementation for particular local rules are also available, starting with the classic Game of Life [1], lattice gas cellular automata [10], stochastic Greenberg-Hastings cellular automata [16] or reaction-diffusion cellular automata models [17]. The encryption applications of cellular automata often require versatility, therefore implementations must afford changing of local rules (see [18] for an example).

All these recent applications are reported in scientific literature as implemented in FPGA. Even projects intended for ASIC implementation, like the RFID security block in [19] was in fact emulated on a FPGA experimental board.

However, FPGAs are not the only hardware solution for cellular automata implementation. Several other implementations were reported so far in the scientific literature. Some new developments include systolic array implementation of cellular automata model [20].

Taking into consideration the specificity of the model, particular developments of exotic models should not surprise. An ”artificial insect eye” is presented in [6]. In Hong Kong, researchers effectively constructed a cellular automata to simulate adaptive systems in the field of urban development (architecture) in order to grow models of future possible architectural developments [21]. 48 cellular automata boards (figure

5, a) run scenarios for high density facade development in a shelve-style construction (figure 5, b).

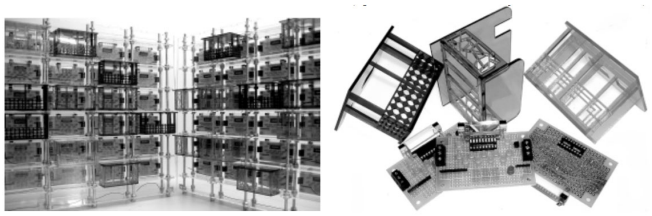

Fig. 5: The cellular automata boards (a) and the system (b) used to simulate architectural evolution in Hong Kong [21].

7 Conclusions

Although cellular automata model is one of the fundamental models of massive parallelism and was proven from the very beginning as able of universal computation, a general solution for its hardware implementation was not yet found. Hardware implementations available so far are either serial accelerators or particular programmable cellular automata configuration generators, bun not the ultimate parallel computers as researchers expected 30 years ago. Probably because of the development of the reconfigurable hardware, efforts in the direction of cellular automata ASIC general implementations seemed to be abandoned, as FPGA offer an excellent environment for emulation of cellular automata. Some famous implementations of cellular automata are overviewed in this paper, together with less famous but creative solutions like the combination of FPGAs with transputer.

References

[1] DASCĂLU M., *Automatele celulare o solutie pentru reducerea complexitatii VLSI*, Editura PRINTECH, 154 pagini, 1998.

[2] GARZON M., *Models of Massive Parallelism*, Springer-Verlag, 2012 (first edition 1995).

[3] CONWAY J., *The Game of Life*, Scientific American magazine, October 1970, pp 120-123.

[4] WOLFRAM S., Statistical Properties of Cellular Automata, Rev. Mod. Phys., vol 58, pp. 601-644, 1983.

[5] CHAUDHURI P., CHOWDHURY D., NANDI S., CHATTOPADHYAY S.: *Additive Cellular Automata: Theory and Application [Volume 1]*, IEEE Computer Society Press, 1997.

[6] DASCĂLU M., FRANŢI E.: *Automatele celulare. Modelare si Aplicatii*, Editura Tehnica, Bucuresti, 2009.

[7]] ACRI 2014, *Cellular Automata: 11th International Conference on Cellular Automata for Research and Industry*, edited by Jaroslaw Was, Georgios Ch. Sirakoulis, Stefania Bandini, Springer-Verlag, 2014.

[8] THOMAS P., *Hardware Compilation of Cellular Automata Algorithms*, Balliol College, Oxford, scientific report, 1993, available on Internet on CiteSeerX.

[9] LIMA A. C., FERREIRA J. C., *Automatic Generation of Cellular Automata on FPGA*, ISBN: 978-972-8822-27-9, REC 2013, pages 51-58.

[10] TOFFOLI T., MARGOLUS N.: *Cellular automata machines: a new environment for modeling*. MIT Press, Cambridge, MA, USA, 1987.

[11] MARGOLUS N.: CAM-8 *A computer architecture based on cellular automata*, 1995, MIT Laboratory For Computer Science, available at https://arxiv.org/abs/comp-gas/9509001.

[12] DE GARIS H., KORKIN M., *The CAM-Brain Machine (CBM): an FPGA-based hardware tool that evolves a 1000 neuron-net circuit module in seconds and updates a 75 million neuron artificial brain for real-time robot control*, Neurocomputing, Volume 42, Issues 14, January 2002, Pages 3568.

[13] HOFFMANN R., VOLKMANN K., SOBOLEWSKI M., *The cellular processing machine CEPRA - 8L*. Mathematical Research, 8L: 179-188, 1994.

[14]] HALBACH M., HOFFMANN R, RÖDER P., *FPGA Implementation of Cellular Automata Compared to Software Implementation*, ARCS 2004 - Organic and Pervasive Computing, Workshops Proceedings, March 26, 2004, Augsburg, Germany.

[15] HOCHBERGER C., HOFFMANN R, VOLKMANN K., WALDSCHMIDT S., *The cellular processor architecture CEPRA - 1X and its configuration by CDL*, in J. Rolim et. al (editors): IPDPS 2000 Workshops, LNCS 1800, pp 898-905, 2000.

[16]] VLASSOPOULOS N., FATES N., BERRY H., GIRAU B.: *An FPGA design for the stochastic Greenberg-Hastings cellular automata*. In 2010 International Conference on High Performance Computing and Simulation (HPCS), pages 565574, 2010.

[17] ISHIMURA K., KOMURO K., SCHMID A., ASAI T., MOTOMURA M., *FPGA implementation of hardware-oriented reaction-diffusion cellular automata models*, Nonlinear Theory and Its Applications, IEICE, vol. 6, no. 2, pp. 252262, IEICE 2015 DOI: 10.1587/nolta.6.252, 2015.

[18] FRANŢI E., DASCĂLU M., *More security and autonomy for users:encryption system with evolutive key*, in "Data, Text and Web Mining and their Business Applications", WIT Press, pag. 335-344, ISBN 978-1-84564-081-1, ISSN 1746-4463, Cambridge Printing, Great Britain, 2007.

[19] BAG J., SARKAR S., *Development and VLSI implementation of a data security scheme for RFID system using programmable cellular automata*, International Journal of Radio Frequency Identification Technology and Applications, Jan 2013, Vol. 4, Issue 2, pp. 197-211.

[20] YARAHMADI A., SETAYESHI S., MOAREFI N., *Hardware Implementation of Cellular Automata on Systolic Array*, Computer Modeling and Simulation, International Conference on, vol. 00, no., pp. 426-429, 2011, doi:10.1109/UKSIM.2011.87.

[21] CHRISTIANE H., FISCHER T. *Using Hardware Cellular Automata to Simulate Use in Adaptive Architecture*. In Proceedings of the 9th International Conference on Computer-Aided Architectural Design Research in Asia, 815-828. CAADRIA. Seoul, Korea: Institute of Millennium Environmental Design and Research, Yonsei University and The Korean Housing Association, 2004.

[22] ŞTEFAN G., *Looking for the lost noise*, in Semiconductor Conference CAS98 Proceedings, vol. 2, IEEE, 1998, pp. 579582.

[23] GHEOLBĂNOIU A., MOCANU D., HOBINCU R., PETRICĂ L., *Cellular Automaton pRNG with a Global Loop for Non-Uniform Rule Control*, Advances in Information Science and Applications - Volume II, pp. 415-420, 2014.