FPGA based Cellular Automata for Environmental Modeling

Ioannis Vourkas and Georgios Ch. Sirakoulis
Department of Electrical & Computer Engineering
Democritus University of Thrace (DUTH)
Xanthi, Greece
{ivourkas, gsirak}@ee.duth.gr

Abstract— Cellular automata (CA), an inherently parallel computing paradigm, have been successfully applied to a range of computational problems to model the behavior of complex environmental systems. Field Programmable Gate Arrays (FPGA) are an ideal platform to investigate the hardware (HW) realization of CA models. FPGA based CA accelerators are not new to the scientific community, however mapping CA models to the HW logic remains a major challenge. In the present work we report on computationally intensive CA-based algorithms and their implementation on FPGA. We highlight intrinsic similarities between CA and FPGA and discuss possible constraints and trade-offs towards appropriate mapping of CA-based models and better exploitation of the favorable HW characteristics. We also refer to various FPGA realizations of our CA environmental systems that deliver significant performance evaluation compared to the corresponding software solutions.

Keywords— Cellular Automata; FPGA; modeling; environmental engineering; real-time decision support systems

I. INTRODUCTION

Ecological systems are generally considered among the most complex ones, being characterized by a large number of diverse components, nonlinear interactions, multiple scaling, and spatial heterogeneity [1]. The constantly increasing need for protection of the environment imposes the development of electronic, almost real-time, computing models able to handle the largest possible number of parameters, in order to simulate fast and effectively the systems under consideration. The mathematical tools mostly used for environmental modeling processes include partial differential equations (PDEs) and numerical methods to compute values of physical quantities in discretized time/space. The problem with PDEs is that while they may naturally fit the situation, they can be mathematically complex and difficult to solve. Thus, the requirement for examining lots of evolution scenarios in a relatively short period of time may account for a different approach. Cellular automata (CA) constitute an alternative solution to the above speculation and have been successfully applied to many real problems in physics, chemistry, biology, and geology [2-4]. Also, coupled map lattices (CMLs) [5], which are comparable to CA in terms of their discrete features, have been used to qualitatively study the dynamics of spatially extended systems. An advantage of CA with respect to systems of PDEs is the stability of their dynamics. They also have the benefit of being mathematically simple, easily simulated, because they are already discrete, and highly parallelizable [2]. Several existing CA models are associated with important natural hazards like forest fires, oil slicks and flow-type phenomena such as debris flows and lava, proving the strong contribution of this computational tool [6-8]. CA constitute a computation paradigm able to capture the essential features of systems in which global behavior emerges from the collective effect of simple and locally interacting components. The CA-based models are important in ecological engineering and they could be used as decision support systems in case of environmental emergency [9]. Moreover, a way to speed-up execution of computationally intensive applications is by using the potential of FPGAs, which enable parallel processing of data using custom digital structures. This way, a CA-based circuit design reduces to the design of a single, relatively simple cell, and the total layout is uniform. CA-based models are straightforward to implement in hardware, leading to algorithms which are fast, especially when implemented on FPGA platforms, because they exploit the inherent parallelism of the CA structure [10]. The CA-based FPGA architecture meets the demand for very fast processing of information imposed by the problem, and also offers a number of beneficial features such as simplicity, regularity, and locality of interconnections. This need to look for new computing approaches which meet the demands of future applications led us to research the possible achievements by implementing computationally intensive applications, based on massively parallel computing paradigms like CA, using FPGAs. However, a proper mapping of an application from an abstract level to the underlying hardware (HW) remains always a challenge.

In the present work we discuss the application potential of FPGAs for computationally greedy CA-based algorithms, like the environmental models. We focus on the constraints and limitations regarding an appropriate possible mapping of CA
algorithms to the FPGA logic space in order to exploit their favorable performance characteristics, considering the existing trade-offs when designing the CA accelerators. Complete understanding of the behavior of a specific CA environmental model in terms of its execution time is a very important aspect towards this goal. We show how a semi-parallel data transfer mode can be used to upload (download) information to (from) HW, both when the needed I/O package pins outnumber or are larger than the available FPGA size. We also report on various environmental CA-based applications that have been implemented on FPGAs and record significant performance evaluation compared to the corresponding software implementations, highlighting the merits of CA-based applications on FPGA platforms that are completely justified in computationally demanding applications regarding ecological engineering.

II. CELLULAR AUTOMATA (CA)
CA are computational models of physical systems, where space and time are discrete and interactions are local [11]. We present here a formal definition of a CA [2]. In general, a CA requires:

1. A regular lattice of cells covering a portion of a d-dimensional space;
2. A set \( C(\vec{r}, t) = \{ C_1(\vec{r}, t), C_2(\vec{r}, t), \ldots, C_m(\vec{r}, t) \} \) of variables attached to each site \( \vec{r} \) of the lattice, giving the local state of each cell at the specific time value \( t \);
3. A rule \( R = \{ R_1, R_2, \ldots, R_m \} \) which specifies the time evolution of the states \( C(\vec{r}, t) \) in the following way:
   \[ C_j(\vec{r}, t+1) = R_j(C(\vec{r}, t), \ldots, C(\vec{r} + \vec{\delta}_j, t)) \]
   where \( \vec{r} + \vec{\delta}_j \) designate the cells which belong to a given neighbourhood of cell \( \vec{r} \).

In the above definition, the rule \( R \) is identical for all sites and it is applied simultaneously to each of them, leading to a synchronous dynamics. However, spatial (or even temporal) inhomogeneities can be introduced. Furthermore, in the above definition, the new state of a particular cell \( \vec{r} \) at time \( t+1 \) is only a function of the previous state of the specific cell and of the cells which belong to its designated neighbourhood. The neighbourhood of cell \( \vec{r} \) is the spatial region in which a cell needs to search in its vicinity. For two-dimensional (2-d) CA, two types of neighbourhood are usually considered: the von Neumann neighbourhood, which consists of a central cell (the one which is to be updated) and its four geographical neighbours, namely north, west, south and east, and the Moore neighbourhood which contains, in addition, second nearest neighbours northeast, northwest, southeast and southwest; i.e. a total of nine cells, whereas the von Neumann neighbourhood comprises only five cells (Fig. 1). CA are decentralized spatially extended systems consisting of simple and identical components with local connectivity. They have the potential to perform complex computations with a high degree of efficiency and robustness, as well as to model the behavior of complex systems from nature. Their implicit spatial locality allows for very efficient high performance implementations.

The CA approach is consistent with the modern notion of unified space–time. In computer science, space corresponds to memory and time to processing unit. In CA, memory (CA cell state) and processing unit (CA local rule) are inseparably related to a CA cell.

III. FPGA/S AND CA IMPLEMENTATIONS
Current FPGAs include logic density equivalent to millions of gates per chip [12-13] and can implement very complex computations. CA consist of a uniform n-dimensional structure composed of many identical synchronous cells where both memory and computation are involved, thus matching the inherent design layout of FPGA HW. Fig. 1 demonstrates the aforementioned structural similarities and summarizes how memory and processing unit are closely related both in CA cells and FPGA configurable logic blocks (CLBs). The structure of a cell consists of a combinational part connected with one or more memory elements in a feedback loop shape. The state of the memory elements is also defined by the inputs and the present state of these elements. FPGAs appear to be very attractive for CA algorithms. Consequently, many FPGA-based CA systems have been proposed and implemented. In [10] researchers developed a series of FPGA-based CA accelerators and concluded 1000x speedup, compared to optimized software, using the then available latest FPGA technology. Kobori et al. [14] implemented lattice gas models on FPGAs and demonstrated a 200x speedup compared to a software version on a 1.8GHz Athlon processor. Besides, tests based on double-precision floating point operations’ performance showed that FPGAs are competitive to multicore processors. More specifically, an AMD dual-core 2.8 GHz Opteron processor has a peak of 11.2 Gflop/s and an Intel 3 GHz quad-core Xeon has a peak of 24 Gflop/s. However, it has been evaluated that theoretically Xilinx Virtex4 LX200 and Altera Stratix II EP2S180 can perform 15.9 and 25.2 Gflop/s, whereas Xilinx Virtex3 LX330 and Altera Stratix III EP3SE260 perform 28 and 50.7 Gflop/s, respectively [15].

Copyright (c) 2012 IEEE. Personal use is permitted. For any other purposes permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org
CA systems from an abstract concept to the HW logic available HW resources sounds exciting. However, mapping may be approximated by a CA, by introducing finite result, any physical system satisfying differential equations issues and requirements of mapping CA algorithms to the dependence on the values at each cell is nonlinear. As a site of the CA lattice may become quite large, especially that each CA cell is processed using a dedicated processing physical realities of special purpose HW.

The structure of each CA cell is separated in two parts: the combinational one, which mainly includes all computations taking place into cells, and the memory part that passes the combinational results to adjacent cells during next time step. When implementing CA-based models in HW one major problem is the fast and efficient data upload/download to/from the HW platform, e.g during initialization step. An effective solution in order to practically confront such issues is to exploit the definition of different kinds of cells in the CA grid; namely here external and internal cells. In fact, the difference is that external cells have a serial data port that supplies initial data to the grid, while internal cells have a parallel port that forwards initial data to internal grid. Initialisation process takes place in a semi-parallel way; initial data are serially provided to the processor through external cells. They are provided, though, simultaneously (in parallel) to all cells located at the left part of the CA grid. Inside the grid data propagates through parallel channels from one internal cell to the other. The adoption of such an initialisation method considerably

Fig 3. Semi-parallel data uploading/downloading method for I/O bound CA-based HW implementations on FPGA. Existence of two kinds of cells shortens the number of input pins as well as the length of interconnections.

IV. CA ENVIRONMENTAL MODEL MAPPING ON FPGA

The combined use of parallel computing systems and CA provides high-performance computational simulation environments to be used for solving real world problems in ecology. These environments allow the researcher to define ecological problems in a cellular form and map then on parallel machines. In [16] a methodology for modeling ecological systems using CA was presented, which enables capturing the essential features of an ecological system and translates them into a suitable form, delivering an effective CA-based environmental model. The produced models can facilitate the development of novel algorithms to simulate environmental systems based on real data, appropriate for implementation on parallel computational systems. Prior and more recent works proved that CA are very effective in simulating environmental systems, because they can capture the essential features of systems where global behaviour arises from the collective effect of simple components which interact locally [15-17]. Nontrivial CA are obtained whenever the dependence on the values at each cell is nonlinear. As a result, any physical system satisfying differential equations may be approximated by a CA, by introducing finite differences and discrete variables [18-19]. Having powerful available HW resources sounds exciting. However, mapping CA systems from an abstract concept to the HW logic remains a major challenge. Hence this work focuses on the issues and requirements of mapping CA algorithms to the physical realities of special purpose HW.

In the basic setup, where the CA lattice is small enough that each CA cell is processed using a dedicated processing element (PE), the whole CA lattice completely fits into the FPGA space. For the CA computation, the whole lattice is downloaded to the FPGA and is run for the required number of generations. Eventually, the resulting generation is uploaded back to the host system where all the required pre- and post-processing is performed. Of course, as the complexity of a physical system increases, the number of bits necessary to represent the values of the physical quantities at each site of the CA lattice may become quite large, especially when these bits are interpreted as the binary representation of an integer, or a floating point number; the usual arithmetic operators +, -, * and / are then used to build the CA rule. In specific, this generalization of the original nature of CAs, working with real-valued quantities and arithmetic operators, is quite useful in ecological modeling. An interesting issue would be how to use a chip with a limited number of processing elements for multi-dimensional CA lattices. Considering again the direct mapping of a 2-d lattice to the dedicated processor, if we have a rectangular lattice of $n \times m$ cells, where $n=kx_i$ and $m=lx_j$, i.e. $kx_l$ times a basic $lx_j$ grid that efficiently fits onto the FPGA platform and which we name “the base grid”, we can imagine the lattice layout in a map cross-fold like zigzag form, as shown in Fig. 2. This way direct mapping is again possible. As long as the CLBs have sufficient memory, each lattice layer is mapped onto CLBs with each CLB storing and processing one CA cell that belongs to the corresponding layer. CLBs start by computing the first layer of the zigzag, followed by the second and so on, until all $kx_l$ layers, which compose the total CA grid, are processed. Of course, using multiple FPGA devices is also an option to the above speculation, though in this case proper distribution of data to the available devices, as well as extra appropriate communication circuitry should be considered.

The structure of each CA cell is separated in two parts: the combinational one, which mainly includes all computations taking place into cells, and the memory part that passes the combinational results to adjacent cells during next time step. When implementing CA-based models in HW one major problem is the fast and efficient data upload/download to/from the HW platform, e.g during initialization step. An effective solution in order to practically confront such issues is to exploit the definition of different kinds of cells in the CA grid; namely here external and internal cells. In fact, the difference is that external cells have a serial data port that supplies initial data to the grid, while internal cells have a parallel port that forwards initial data to internal grid. Initialisation process takes place in a semi-parallel way; initial data are serially provided to the processor through external cells. They are provided, though, simultaneously (in parallel) to all cells located at the left part of the CA grid. Inside the grid data propagates through parallel channels from one internal cell to the other. The adoption of such an initialisation method considerably
shortens the number of input pins as well as the length of interconnections, thus accelerating the operation of the processor. This procedure, summarized in Fig. 3, is characterised as semi-parallel, meaning that data starts to be loaded simultaneously, i.e. in parallel, from the very left column of the CA grid where the external cells are placed (of course any side of the CA could be chosen to define external cells). On the other hand, this process continues serially with initial bit strings moving from one internal cell to its adjacent at the same row of the grid until all cells are properly initiated. Also, parallel-to-serial converters, that drive parallel outputs to serial inputs in order to minimize the number of pins of the processor, and serial-to-parallel converters inside the block, that modify the serial routes to parallel, as well as supplementary memory components in order to balance delays during signal propagation, are involved.

The aforementioned approach has been pursued in the FPGA realization of computationally intensive CA-based environmental models including, among others, oil slick spreading simulation. In contrast to a true, fully parallel function mode, where all data are considered to be readily transferred in parallel to/from all used FPGA CLBs, the semi-parallel method proved to be highly efficient for such I/O bound applications. Specifically, statistical information for the oil slick spreading model HW implementation for a 20x20 CA lattice, provided by the compilation report of the design software (Altera Quartus II), showed satisfying coverage of the available I/O processor pins (85%) for a Stratix IV family FPGA (EP4SE820F43C4N device). The achieved performance results highlight the importance of considering more efficient data transfer modes for FPGA realizations of CA environmental models, like the presented here semi-parallel mode, in order to take advantage of their performance assets. This data transfer method could be successfully pursued in other environmental model implementations, like fire spreading and lava flow models.

V. CONCLUSIONS

In this article we present the basics of CA and argue that they consider an effective alternative to commonly used FDEs and numerical methods for modeling computationally intensive environmental processes. We demonstrate why FPGAs constitute ideal HW platforms for the realization of such models, given their favorable performance characteristics, also being more energy efficient than general-purpose CPUs, compared to corresponding software implementations. Improvement of computations both in terms of speed and accuracy are of great importance in high performance computing, and one of the ways to improve computations is to understand the process of mapping an application to the available hardware. To this end, we focus on the trade-offs and constraints imposed by the mapping process of CA-based environmental models on FPGA, which remains always a challenging task, and discuss practical solutions to possible problems. This work combines well with our previously presented methodology for modeling ecological systems using CA. Environmental engineers and circuit designers here should encounter a handy tutorial regarding alternative cost-effective solutions for ecological systems modeling, as well as a straightforward manner for the HW implementation of their CA-based models.

REFERENCES