Fast implementation of a bio-inspired model for decentralized gathering

Bernard Girau*
LORIA - INRIA Nancy Grand Est, France
Bernard.Girau@loria.fr

Cesar Torres-Huitzil
Polytechnic University of Victoria, Mexico
ctorresh@upv.edu.mx

Abstract

In the context of the emergence of alternative computing resources to address the challenge of the upcoming end of Moore's law, we consider the feasibility of gathering computational resources by means of decentralized and simple local rules. We study such decentralized gathering by means of a stochastic model inspired from biology: the aggregation of the Dictyostelium discoideum cellular slime mold. The environment transmits information according to a reaction-diffusion mechanism and the agents move by following excitation fronts. Despite its simplicity this model exhibits interesting properties of self-organisation and robustness to obstacles. In this paper we study the major challenges that must be solved when designing a fast embedded implementation of the decentralized gathering model.

1. Introduction

Recent trends of integrated circuit design show that the regular technological improvements in speed and size (Moore's law) should soon reach physical limits, mainly related to power consumption and dissipation [7, 9]. Different lines of research try to propose alternative computing devices and to define their related computing paradigms, based on multiple generic computing units, possibly distributed in an unknown and irregular way [3]. Therefore the main algorithmic question might evolve towards "how to make computing units cooperate to solve a given task ?" [9]. An upstream question is "how to gather or assign enough computing resources to solve a given task ?", in a context where faulty units may appear. In a preliminary study of this question, we consider here the problem of gathering computing units in a strongly constrained context: units use only local rules, units move on a lattice and need to gather to form a compact cluster, units have no idea of their own position and of the position of the other units, units may only send messages that can be relayed (possibly with errors) by the cells of the lattice, units only perceive the state of the neighbouring cells, and the only actions units can undertake is to move to these cells or change the state of these cells. Though this paper describes an upstream work that does not yet pretend to define precisely how the gathering process will be applied to a real system, we may notice that "moving" units might correspond to transmitting the task assignments between units when using computational resources with fixed locations. Further works are required to define more precisely what kind of co-processing units might be in charge of handling the decentralized gathering process, and how the distributed processing of a given task might be assigned to the "moving" units. In this paper, our ambition is only to show that a simple model is able to achieve decentralized gathering, while being suitable for efficient distributed implementations. Our approach is inspired by biology, where such decentralized gathering is observed, so as to derive a model and its implementation.

The cellular slime mold Dictyostelium discoideum is a fascinating living organism that has the ability to live as a mono-cellular organism (amoeba) and to transform into a multi-cellular organism when needed. In normal conditions, the amoebae live as single individuals. However, when the environment becomes depleted of food, a gathering phenomenon is triggered and single amoebae aggregate to form a complex organism that will move and react with coordinated changes. In this work, we take inspiration from the first stage of the multi-cellular organisation process, the aggregation stage, which consists in gathering all the amoebae in a compact mass called a mound (e.g., [17]). In [4], Fatès proposes a simplified model of Dictyostelium discoideum that exhibits the main behavioral properties of the aggregation mechanism: reaction-diffusion and chemotaxis. Our project is built around this model. It uses a cellular automata to describe the environment, and a multi-agent approach to model the amoebae. Section 2 describes Fatès' model to achieve decentralised gathering and its main properties. Section 3 presents its general FPGA implementation. Finally, we derive conclusions about the main obstacles and possible modifications of our approach.

*Member of the *Amybia* INRIA collaborative research project (led by Nazim Fatès, `www.loria.fr/~fates/Amybia/project.html`), that supports this work.

2. A Reaction-diffusion-chemotaxis Model

We point out the main behavioral components that our project's model ([4]) aims at exhibiting and we shortly discuss previous models. Then our project's model is formally described in two parts: the environment and the agents.

2.1. Inspiration and previous models

Decentralised algorithms to gather robots to form circular or polygonal shapes have been proposed in [16], where all robots "see" the positions of each other. A similar problem with a limited visibility range has been studied in [2]. We refer to the work of [15] for recent developments on the decentralised gathering problem. In this paper, we get rid of any assumption on the visibility range using an environment that transmits messages on arbitrarily long distances. The decentralised gathering problem is also related to the Leader Election problem, but our goal is not to select one unit among many, but to gather randomly located units in a compact location that emerges by consensus.

In vitro experiments show that the aggregation phenomenon of Dictyostelium is triggered by one or several amoebae that attract other amoebae that are located in their vicinity to form groups. The first groups merge until only a few clusters remain; these will attract other amoebae to them to form a cluster where cellular differentiations will occur to lead to the multi-cellular organism. Attraction is led by the transmission of waves of chemical messengers, which follow typical evolving reaction-diffusion patterns. This phenomenon is called chemotaxis. The chemical messengers are internally produced by the amoebae. When an amoeba detects a high increase in the external concentration of the messengers, it follows the concentration gradient (chemotaxis) and releases its own internal messengers. Then it becomes insensitive to the messengers during a given refractory period, and in the meanwhile, the released messengers diffuse and excite other sensitive amoebae.

Several models have been proposed to study the dynamics of Dictyostelium (see a review in [14]). Many of them are based on partial differential equations [11, 18]. Some works aim at being very close to the biological inspiration, comparing simulation outputs with observations of the aggregation of Dictyostelium [10], or modelling the receptors of the chemical messengers [12]. Most works use continuous or hybrid models; to our knowledge, our project's model (see [4]) is the first fully discrete model for capturing Dictyostelium's behaviour. By fully discrete, we mean that time and space are discrete and the state of the amoebae is described in qualitative terms rather than quantitative (integers or decimal values). This discretization is an essential property when digital hardware implementations are expected. The reaction-diffusion mechanism alone is well understood, with explicit links between the discrete and continuous models (e.g. [19]). This mechanism shows problem-solving abilities [1]. Our project's model adds virtual chemotaxis as a new feature to study and use. Two layers compose it: the environmental layer is a cellular automaton that models a reaction-diffusion process while the particle layer describes the moves of virtual amoebae.

2.2. The environmental layer

Space is modelled by a regular lattice $\mathcal{L} = \{1, ..., X\} \times \{1, ..., Y\}$ in which each cell $c = (c_x, c_y) \in \mathcal{L}$ is associated to a state. The set of possible states for each cell is $\{0, ..., M\}$, the state of cell c at time t is denoted by σ_c^t. State 0 is the *neutral* state, state M is the *excited* state. A cell may evolve from the neutral state to the excited state if at least one of its neighbours is excited (rule R1). To model the uncertainty on this transition, we will consider that it happens with a given probability p_T, called the *transmission rate*. States 1 to $M-1$ are the *refractory* states. A cell in a refractory state evolves in an autonomous way by decrementing its state by 1 (rule R2) until it reaches the neutral state. A neutral cell surrounded by neutral cells stays neutral (rule R3). To express these rules without ambiguity, for a cell $c \in \mathcal{L}$, let us denote by N_c the neighbourhood of this cell. Let E_c^t be the set of excited cells in the neighbourhood of c at time t: $E_c^t = \{c' \in N_c | \sigma_{c'}^t = M\}$. We also denote by $|X|$ the cardinal of a set X.

With these notations, for a time $t \in \mathbb{N}$ and a cell $c \in \mathcal{L}$, let $B(p)$ be a Bernoulli random variable that equals 1 with probability p and equals 0 with probability $1-p$. The local rule governing the evolution of the environment is:

$$\sigma_c^{t+1} = \begin{cases} M & \text{if } \sigma_c^t = 0 \text{ and } |E_c^t| > 0 \\ & \quad \text{and } B(p_T) = 1 \quad \text{(R1)} \\ \sigma_c^t - 1 & \text{if } \sigma_c^t \in \{1, \ldots, M\} \quad \text{(R2)} \\ 0 & \text{otherwise} \quad \text{(R3)} \end{cases}$$

A set of adjacent cells that are all in the excited state M is called an excitation front. In the next subsection, we explain how the excitation fronts guide the amoebae for moving on the lattice (chemotaxis).

2.3. The particle layer

The amoebae are supposed to be all identical, and in constant number as no birth or death process is considered. Several amoebae may be located at the same cell. We arbitrarily allow only one amoeba to move from a non-empty cell at each time step. We do not limit the number of amoebae that can simultaneously move to a given cell, but we arbitrarily choose to allow an amoeba to go on a neighbouring cell only if this cell contains less than two amoebae [4, 17]. Let us define a cell that contains no amoeba as an *empty*

cell, and a cell that contains less than two amoebae as a *free* cell. The movement rules state that, at each time step, for each non-empty cell, one single amoeba may:

- move to an adjacent free cell (rule R4),
- move to an adjacent excited free cell (rule R5),
- stay on the same cell (rule R6).

To apply rule R4 (noise rule), we consider that each non-empty cell may give an amoeba to one of its neighbours with probability p_A, called the *agitation rate*. This neighbour is randomly selected among all the free neighbours. Similarly, to apply rule R5 (chemotaxis rule), amoebae move to a cell randomly selected among the excited free cells of the neighbourhood. Rules R4 and R5 are made mutually exclusive. Formally, for $t \in \mathbb{N}$ and $c \in \mathcal{L}$, let $\tilde{N}_c^t$, respectively $\tilde{E}_c^t$, be the set of *free* cells, respectively *excited free* cells, in the neighbourhood of c. For a finite set S, we denote by $\mathcal{R}(S)$ the operation of selecting one element in S with uniform probability, with the convention $\mathcal{R}(\emptyset) = \emptyset$. $\mathcal{R}$ randomly selects a neighbour for moving. We use a Bernoulli function to impose noise on the moves of an amoeba with probability p_A. To represent the move of one amoeba from a **non-empty** cell c to another cell Δ_c^t, with the convention $\Delta_c^t = \emptyset$ if no move occurs, we have:

$$\begin{array}{rll} \text{if } \mathcal{B}(p_A) = 1 \text{ then} & \Delta_c^t = \mathcal{R}(\tilde{N}_c^t) & \text{(R4)} \\ \text{else} & \text{if } \sigma_c^t = 0 \text{ then } \Delta_c^t = \mathcal{R}(\tilde{E}_c^t) & \text{(R5)} \\ & \text{else } \Delta_c^t = \emptyset & \text{(R6)} \end{array}$$

2.4. Coupling of environment and particles

Amoebae act on the environment by emitting excitations that propagate to neighbouring cells. We do not take into account the number of amoebae contained in each cell: a non-empty neutral cell may become excited with probability p_E called the *emission rate*. Since this rule may interfere with rule R1, we combine both rules into rule R1':

$$\begin{array}{rl} \sigma_c^{t+1} = M \text{ if: } & \mathcal{B}(p_T) = 1 \text{ and } \sigma_c^t = 0 \quad \text{(R1')} \\ \text{and } (|E_c^t| > 0 \text{ or } (c \text{ non-empty and } & \mathcal{B}(p_E) = 1)) \end{array}$$

2.5. Properties

We simultaneously carry out a theoretical, experimental and hardware study of our project's model. Theoretical and experimental aspects are not the subject of this paper, that focuses on the hardware implementation issues. The main properties of this model are presented in [4]. Its dynamics depends on four parameters: the excitation level M, the transmission rate p_T, the emission rate p_E and the agitation rate p_A. Our study shows that different qualitative behaviours may be observed: the static regime, the non-coherent regime, the extinct regime and the self-organising regime. The static regime is obtained in the case of systematic emission of waves by amoebae ($p_T = 1$ and $p_E = 1$): no moves occur because the excitation fronts collide systematically without being able to transmit any information from an amoeba to another one. The non-coherent regime may be observed in the case of non-perfect transmission conditions ($p_T < 1$): the reaction-diffusion waves are independent from the position of the amoebae and no organisation can occur. The extinct regime is attained when the transmission rate is less than a critical value ($p_T < 0.23$ for $M = 3$). In this case, the waves can only travel small distances before they become extinct. The most promising behaviour, the self-organising regime, is observed when the transmission is perfect and when the emission rate is less than 1 (typically $p_E = 0.1$) and for various values of agitation rate. In this regime, a gathering phenomenon shows a progressive merging of the amoebae from small clusters to large clusters, after a few thousands iterations. The complexity of this hierarchical dynamics results from successive emerging behaviours: formation of waves, formation of first groups, extension and shrinking of the regions according to their respective size, captures of small clusters by a few clusters. Among interesting properties observed in the system, we have shown that gathering could also occur in the presence of obstacles as the virtual amoebae could take advantage of narrow corridors to find their way to an attracting cluster. For further details and illustrations about the model dynamics and its self-organising regime, see [4].

3. Hardware implementation

The hardware part of our project's model is motivated by two main goals: to develop fast implementations to explore complex dynamics (mostly phase transitions that may be obtained after a huge number of iterations), and to perform a preliminary study of the ability of our project's model to provide an efficient decentralized gathering process for a large amount of distributed computing units. From now on, we arbitrarily use the 8-neighbourhood, and we set the excitation level to $M = 3$. Following the study of the dynamical behaviour of the model in [4], we also set $p_T = 1$ since aggregation only occurs with perfect transmission.

3.1. Cell and amoebae implementation

For implementation purposes, we define a *node* as a cell together with the amoebae it contains. The moves of amoebae may be simply described as the evolution of the "population" of each node as a part of its internal state. Figure 1 shows the I/O of the node module.

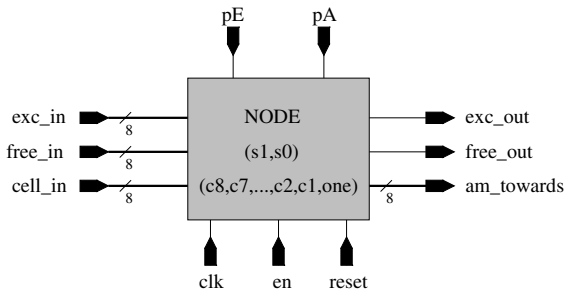

Figure 1. Node module

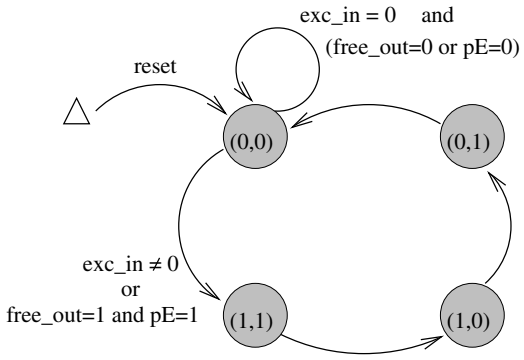

Figure 2. Cell state machine

3.1.1 State of a cell

Considering the environmental layer only, the state of each cell belongs to $\{0, 1, 2, 3\}$, so that we code it with two bits `(s1,s0)`. This state evolves according to rules R1', R2 and R3. These rules may be expressed as the state machine depicted in figure 2. Signal `pE` stands for $\mathcal{B}(p_E)$. Input `free_out` codes for the presence of at least one amoeba in the node (it is an internal signal equal to the output `free_out` of the node).

3.1.2 Population of a node

Considering the amoebae, they are coded as the number of amoebae that are located in the cell that corresponds to the local node. Amoebae may move towards free cells only. Free cells contain at most one amoeba. Since up to 8 amoebae may simultaneously move towards a free cell, each node contains at most 9 amoebae. Instead of coding the population size (using 4 bits and counting at each time the number of arriving amoebae), we use 9 flip-flops: though less compact in terms of number representation, this solution does not require coding and counting resources, so that it uses significantly less logic cells. The first flip-flop stores '1' if there is at least one amoeba. Then the 8 other flip-flops

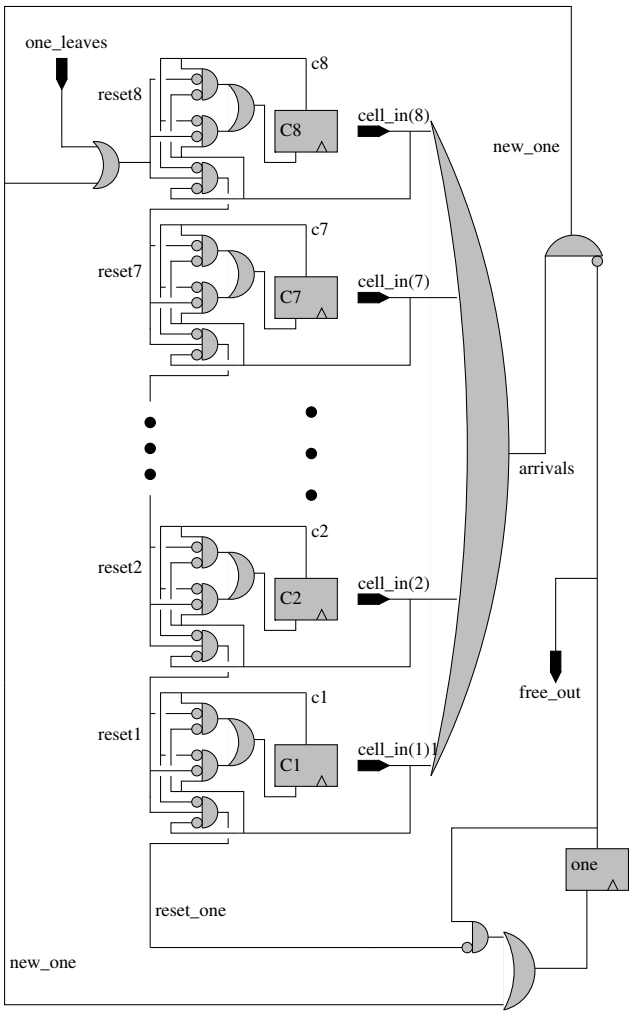

Figure 3. Module to code population of amoebae

F directly receive arriving amoebae. Each time an amoeba leaves the node, one of the flip-flops storing '1' is reset to '0' (the reset command is transmitted among flip-flops until finding a '1'). Similarly, if amoeba arrivals occur when the cell is empty, then the first flip-flop is set to '1' and one of the other flip-flops is reset to '0'. Figure 3 depicts the resulting architecture to store the node population. The node indicates whether it is free or not with signal `free_out`.

3.1.3 Amoeba moves

Figure 4 shows how the moves of the amoebae are implemented (rules R4, R5, R6). Signal `pA` stands for $\mathcal{B}(p_A)$. It controls 8 multiplexers (one for each neighbour) that indicate whether the corresponding neighbour is free or excited and free. Moreover signal `neutral` is used in the sec-

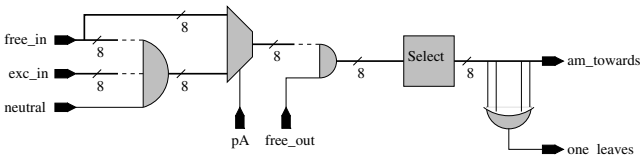

Figure 4. Implementation of the particle layer

ond case (R5). It is internal and it codes for $\sigma_c^t = 0$, i.e. `(s1,s0)=(0,0)`. In the same way, if signal `n_empty` is '0' then all choices are set to '0' because no amoeba may move if the cell is empty. Then the `Select` module randomly selects only one choice among possibly several. Finally an `OR` gate determines if an amoeba will move while the bus `am_towards` indicates where it will move.

The random selection of a signal set to '1' among possibly several is complex. In our implementation, we use a cyclic priority module, where the main priority is given to a signal that is randomly specified by three bits provided by a linear feedback shift register (LFSR). This implementation suffers the following drawbacks: 1. it is not uniformly random, and 2. though it is fair, it introduces some systematic bias in the selection of close signals because of the cyclic priority. Nevertheless, first experiments indicate that these drawbacks do not modify the overall behavior of the model. We have also developed a purely uniform random selection module, but it requires about three times more space. As explained later, we have started the design of a block-synchronous version for which the optimization of the area of each node is less important: this uniform random selection module will be used within this version.

3.2. Random processes

The definition of the model includes several random aspects: $\mathcal{B}(p_T), \mathcal{B}(p_A), \mathcal{B}(p_E), \mathcal{R}(\tilde{N}_c^t), \mathcal{R}(\tilde{E}_c^t)$. In a software implementation, the same random number generator may be used, thanks to the independance of the successively generated numbers. But in a parallel hardware implementation, all streams of stochastic bits must be generated by separate modules, in each node. Since we consider the case where $p_T = 1$, and since our random selection module (used for both $\mathcal{R}(\tilde{N}_c^t)$ and $\mathcal{R}(\tilde{E}_c^t)$) just needs a single LFSR, we finally have to implement $3\,X \times Y$ random number generators. This implies an important cost in space for the random aspects of the model (which corresponds to the computation time that is mostly spent in generating random numbers in the software implementation).

Therefore, our choices for the implementation of the random processes have been carefully studied. Most digital hardware solutions are based on LFSR or cellular automata [8]. Another approach takes advantage of large numbers

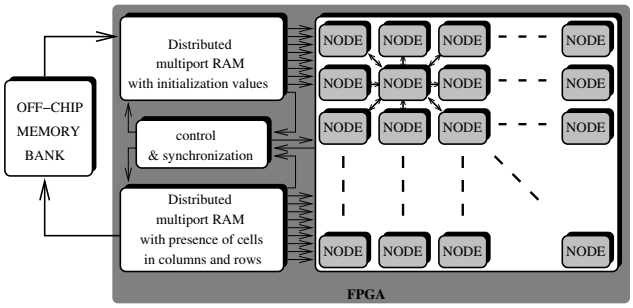

Figure 5. General architecture

stored in parallel [13]. Since the LUTs of the FPGA logic cells may be efficiently configured as shift registers [5], we use LFSR-based random generators.

The selection module uses a 3-bit random counter to define the main priority choice. Since all 3 bits must be simultaneously accessed, 3 flip-flops are required. Instead of only using 3 bits for the random counter (resulting in a 8 cycles-periodicity), we use the 32-bit random counter of [5] that only needs 3 logic cells to strongly increase the periodicity without requiring more resources.

Experiments in [4] show that both emission and agitation rates do not need a high precision. Therefore we use two 52-bit random counter of [5], comparing only 8 of their generated bits to p_E and p_A (coded on 8 bits).

3.3. General architecture

Figure 5 describes the general architecture of our implementation. It consists of a grid of 30×40 identical nodes. Border nodes receive constant inputs from their non-existing neighbours (`exc_in`, `free_in` and `am_in` are set to '0' for these nodes). This grid receives initialization data (positions of amoebae, one per node at most, all nodes neutral) from an external memory through a data distributor module. This data distributor mostly consists in sending each bit coming from the memory to a demultiplexer that chooses the destination node of this bit within its corresponding row of nodes. Instead of sending to the memory the states of all nodes as an output, we take advantage of the quantitative criterion BBR (bounding box ratio) that is used in the experimental study of [4] for the evaluation of the aggregation: minimal relative size of an array of nodes that contains all amoebae. Therefore, we just implement an OR gate for each row and for each column of nodes, and we write the resulting bits in the external memory.

3.4. Implementation results

This work uses a PCI bus board with three FPGAs, the largest one being a Virtex-4 XC4VLX160ff1513-12 FPGA

from Xilinx, that contains 135,168 logic cells (67,584 slices). The design was synthesized, placed and routed automatically in Xilinx Foundation ISE 7.1i. Each node module requires 49 slices (about 25 for the different random number/stream generators). A few 500 slices are sufficient to implement the data distributors (I/Os). So that the whole architecture implements a 30×40 grid on 88 % of the logic cells.

Software implementations on a microprocessor based computer, Pentium 4.2 GHz, require around 170 μs per evolution step for a 30×40 grid. The estimated maximum clock frequency of the proposed hardware is 130 MHz. Thus, the implementation on the Virtex-4 provides a speed factor up to 22000x. Depending on the parameter values, the size of the environment and the obstacles, aggregation occurs in the experiments in [4] after some 2,000 to 20,000 iterations. Therefore the great speedup we obtain will become particularly interesting if we are able to implement much larger grids. This is the goal of our current works (see below).

4. Conclusion and future works

In this paper, a bio-inspired model to solve the decentralized gathering problem is shortly described. It is based on the aggregation properties of the cellular slime mold Dictyostelium discoideum that may live as a mono-cellular organism, and that is able to behave as a multi-cellular organism when needed. We model both the environment and the individual amoebae by means of both cellular automata and reactive agents (simple computational abilities and no memory). We focus on the fully parallel hardware implementation of this model, in order to study its ability to provide a massively distributed computational model for decentralized gathering. Despite an outstanding speedup factor, our implementation work points out two main limitations. In terms of embedability, the area cost of the stochastic aspects of our project's model is important. Therefore, our theoretical study will have to evaluate the robustness of our model to low-quality random streams that may also be spatially highly correlated. In terms of usefulness for large-scale efficient simulations, the grid size we are able to handle does not correspond to interesting experimental environments, and the corresponding software computation time does not justify the use of fast FPGA implementations. Therefore we are currently developping a block-synchronous implementations that takes advantage of the FPGA-embedded SRAM blocks to partition the environment: each SRAM block stores the states of the cells that have the same relative location in the different parts, as in [6]. This approach may enable us to implement up to 500,000 nodes on the same Virtex-4 FPGA.

References

[1] A. Adamatzky. *Computing in nonlinear media and automata collectives*. Institute of Physics Publishing, 2001.

[2] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita. Distributed memoryless point convergence algorithm for mobilerobots with limited visibility. In *IEEE Trans. on Robotics and Automation*, 1999.

[3] A. Dehon. Very large scale spatial computing. In *Proc. Unconventional Models of Computation*, 2002.

[4] N. Fatès. Gathering agents on a lattice by coupling reaction-diffusion and chemotaxis. Technical report, INRIA Nancy Grand-Est, 2008. `http://hal.inria.fr/inria-00132266/`.

[5] M. George and P. Alfke. *Linear Feedback Shift Registers in Virtex Devices*, 2007.

[6] B. Girau, A. Boumaza, B. Scherrer, and C. Torres-Huitzil. *Block-synchronous harmonic control for scalable trajectory planning*, chapter 4. I-Tech, 2008. To be published.

[7] L. Kish. End of moore's law: thermal (noise) death of integration in micro and nano electronics. *Physics Letters A*, 305, 2002.

[8] I. Kokolakis, I. Andreadis, and P. Tsalides. Comparison between cellular automata and linear feedback shift registers bases pseudo-random number generator. *Microprocessors and Microsystems*, 20, 1997.

[9] J. Lawson and D. Wolpert. Adaptive programming of unconventional nano-architectures. *ArXiv Condensed Matter e-prints*, 2006.

[10] S. Mackay. Computer simulaion of aggregation in dictyostelium discoideum. *Journal of Cell Science*, 33, 1978.

[11] A. Marée. *From pattern formation to morphogenesis: Multicellular coordination in Dictyostelium discoideum*. PhD thesis, Utrecht University, 2000.

[12] J.-L. Martiel and A. Goldbeter. A model based on receptor desensitization for cyclic AMP signaling in dictyostelium cells. *Biophysical Journal*, 52(5):807–828, 1987.

[13] M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Transaction on modeling and Computer Simulation*, 8(1), 1998.

[14] S. Nagano. Modeling the model organism dictyostelium discoideum. *Development Growth and Differentiation*, 42(6):541–550, 2000.

[15] G. Prencipe. Impossibility of gathering by a set of autonomous mobile robots. *Teor. Comput. Sci.*, 384(2-3), 2007.

[16] K. Sugihara and I. Suzuki. Distributed motion coordination of multiple mobile robots. In *5th IEEE International Symp. on Intelligent Control*, 1990.

[17] B. Vasiev, T. Bretschneider, and C. Weijer. A model for cell movement during dictyostelium mound formation. *Journal of Theoretical Biology*, 189(1):41–51, 1997.

[18] B. Vasiev, F. Siegert, and C. Weijer. A hydrodynamic model for dictyostelium discoideum mound formation. *Journal of Theoretical Biology*, 1997.

[19] J. Weimar. Cellular automata for reaction-diffusion systems. *Parallel Comput.*, 23(11):1699–1715, 1997.