

Hardware implementation of a biomimicking hybrid CA

Menelaos Madikas¹, Michail-Antisthenis Tsompanas², Nikolaos Dourvas¹,
Georgios Ch. Sirakoulis¹, Jeff Jones², and Andrew Adamatzky²

¹ Democritus University of Thrace, Department of Electrical &
Computer Engineering, Xanthi, Greece
mmadikas, ndourvas, gsirak@ee.duth.gr,
WWW home page: <http://gsirak.ee.duth.gr>

² Unconventional Computing Laboratory,
University of the West of England, Bristol BS16 1QY, UK
antisthenis.tsompanas, andrew.adamatzky@uwe.ac.uk

Abstract. A hybrid model, combining a Cellular Automaton (CA) and a multi-agent system, was proposed to mimic the computation abilities of the plasmodium of *Physarum polycephalum*. This model was implemented on software, as well as, on hardware, namely on a Field Programmable Gate Array (FPGA). The specific ability of the *P. polycephalum* simulated here is given in brief, also bringing attention to the approximation of a Kolmogorov-Uspensky machine (KUM), an alternative to the Turing machine. KUM represent data and program by a labeled indirected graphs and a computation is performed by adding/removing nodes/edges. The proposed model implementation is taking full advantage of the inherent parallel nature of automaton networks, and CA, as a result of the mapping of the local rule to a digital circuit. Consequently, the acceleration of the computation for the hardware implementation, compared to the software, is as high as 6 orders of magnitude.

Keywords: Slime Mould, Cellular Automata, Hardware, Agents, Kolmogorov Machine

1 Introduction

Physarum polycephalum is widely used in the last decade as an unconventional computing substrate, because it demonstrates complex behavior, regardless of its apparent simplicity, easy culturing and very low-cost experimentation. *In vivo* experiments with that biological, massively parallel computing prototype were developed [1]. A plasmodium, the ‘vegetative’ stage of *P. polycephalum* life cycle, was persuaded to scout for nutrients in its vicinity and link all of them with a tubular network. The importance of the functionality of the tubular network to the survival of the plasmodium (transfer of nutrients, metabolites, and chemical and electrical messages) subjects its actual configuration in rounds of optimization by the actual plasmodium and the local conditions of the experiment. This

is the reason why the plasmodium is such a successful paradigm of a biological substrate computer with inputs of geometrical configuration of nutrient sources (NSs) and an output of an interconnected graph [2].

Some examples of the vast range of problems tackled by *P. polycephalum* [3] are labyrinth solving [4, 5], approximation of Voronoi diagrams and Delaunay triangulation [6], travelling salesman problem [7], simulation of Boolean logic [8] and evaluation of transport [9] and ancient road networks [10]. Nonetheless, the abilities of *P. polycephalum* inspired scientists to invest efforts towards the simulation of its behavior and its usage to solve complex problems, like designing routes in information networks [11]. Some examples of the computerized approximation of the behavior of the plasmodium are a mathematical model with feedback of the tube dynamics [12], agent-based particle models [13] and cellular automata (CA) [14–16]. We propose a novel approach towards computerizing the abilities of the slime mould by designing a hybrid model, which combines CA and a multi-agent system is introduced.

Moreover, a rather interesting paradigm of in-vivo experiments utilizing *P. polycephalum* is the approximation of Kolmogorov–Uspensky machine [17, 18]. Thus, here the results produced from these experiments were used as a control for the novel hybrid model proposed. Kolmogorov and Uspensky [19, 20] proposed that a dynamically changing graph could represent an abstract computing machine. This graph has a finite amount of nodes and the edges between them are not directed. Each of the graph’s nodes are alternatively labelled and only one of the nodes can be activated for a given time step. This structured was initially defined as the Kolmogorov complex and, then, was more commonly known as Kolmogorov-Uspensky machine (KUM). This term will be used hereafter.

KUM is an alternative to the Turing machine, with the main difference of the replacement of the tape of the Turing machine by a graph. Given this geometric substrate of the KUM (the graph), it can accurately mimic growth phenomena and computation influenced by structure in natural systems (like biological systems, chemical systems etc). An additional meaningful distinction is that whereas the Turing machine was intended to replicate a human-executed computation, the KUM aims to portray “computation as a physical process” [21].

Despite their differences, Turing and KU machines belong the same classes of abstract mathematical machines. A physical implementation of a KUM is searched within biological substrates, like the plasmodium of *P. polycephalum* [17, 18]. *P. polycephalum* is an ideal choice for this because of its capacity of exploring and growing in a graph-like configuration and dynamic reshaping of the edges and nodes in the graph.

Moreover, as the plasmodium is stimulated by chemoattractant nutrient sources, utilizing a vast distributed array of membrane-bound sensor proteins, distributed tools were chosen for its simulation, like CA and multi-agent models. In this paper, we focus on the hardware implementation of the proposed hybrid model of *P. polycephalum* to mimic the physical implementation of a KUM in the plasmodium of *P. polycephalum*.

2 Description of the CA-Multiagent model

A new model of CA was developed, in order to combine CA [15, 16], and multi-agent modeling [22]. This new model integrates the parallel nature of CA along with the dynamic behavior in space and time of the multi-agent system without, however, increasing the complexity of the system. The model is computationally simple by using distributed local sensory behaviors, although approaching some of the complex phenomena observed in *Physarum*. In order to configure *Physarum*'s adaptive motif, the selected mechanism must be able to adjust its pattern over time, i.e. the pattern must be more flexible to show the emerging properties, which are extensible and can be calculated.

In this model, the inherent flow of the colloidal solution is provided by agent-to-cell mediation, and gel matrix resistance is provided by the agent-agent collisions. The consistency of this set of "crowd" is ensured by the fact that there is mutual attraction to the stimuli deposited by the population of the agents. The directional orientation and movement of the activated plasmodium front is created by coupling the emerging mass behaviors by dragging local source stimuli. Changing the value of cell identities is equivalent to secreting a quantity of chemoattractant during their successful motion and detecting the largest amount in the cell's neighborhood by three sensors. As a result, they adjust the cell's angle, so the behavior of other agents can also affect their behavior.

The area of the experiment is divided into a grid of identical square cells, with the side of length equal to α and represented by a CA assuming that each square element of the surface is a cell of a CA. The width of the neighborhood of (i, j) cell is assumed to be equal to one on both sides, meaning that, we have a Moore neighborhood. The state of the new CA model at time t is:

$$C_{i,j}^t = \{griddata_{i,j}^t, trail_{i,j}^t, particle_ids_{i,j}^t, angle_{i,j}^t\} \quad (1)$$

Parameter $griddata_{i,j}^t$ can get values at time t that indicate whether the cell is a wall or a food stimulus. Referring to $trail_{i,j}^t$, is the value derived from the food diffusion equation and represents the strength of the smell of the food stimuli at time t in the (i, j) cell. Additionally, $particle_ids_{i,j}^t$ is the value of the agent's (i, j) identity at time t . When there is no agent in the cell, then, as normal, its value is equal to zero. Finally, $angle_{i,j}^t$ is a variable that shows the angle of the agent in a cell.

Some program variables are specified at the initialization of the model. The main parameters are the amount of agents in the population, grid size, consumption of food value (*diffdamp*) and the variable that indicates how large the value of the data nodes are. This is displayed as $griddata_{i,j}^t$ to $trail_{i,j}^t$ and can get two values, the *projectvalue* (when there is no agent nearby or on the central cell) or the *suppressvalue* (when there is at least one agent in the neighborhood of the central cell on which the food is placed).

Subsequently, the agents are placed in random positions within the CA grid with their velocity vectors oriented at random angle values. When a CA neighborhood contains one agent or is fully occupied of agents then the value of the trail value is generated by the equation:

$$trail_{i,j}^{t+1} = trail_{i,j}^t + suppressvalue \quad (2)$$

Else, if the area is free from agents then it gets the value

$$trail_{i,j}^{t+1} = trail_{i,j}^t + projectvalue \quad (3)$$

During the evolution of the model, the cell motor behavior is activated to move the agents along their velocity vectors, thus renewing the values of the *particle_ids* and their velocity vectors. The new coordinates of an agent are calculated and an attempt is made to move by one cell in the Moore neighborhood of the central cell (where the agent is located).

$$tvi = i + \cos(\text{angle}(i,j)) \times speed \quad \text{and} \quad tvj = j + \sin(\text{angle}(i,j)) \times speed \quad (4)$$

If a cell is bound or wall and is selected by an agent to make a move towards it, then that agent will not move to that cell but the angle of the velocity vector will be renewed to a random value, i.e. $angle_{i,j}^{t+1} = \text{random}() \times 360$. However, if all the prerequisites are met in order to make a movement, the value of the agent's identifier of the given cell will be given to the new cell, as well as its angle, and the trail value will be renewed by increasing it by an amount $depT$. This will be an attractive means for other agents to move towards it in order to create a single structure as the biological organism.

$$\begin{aligned} particle_ids_{i,j}^{t+1} &= particle_ids_{i+1,j+1}^t \\ angle_{i,j}^{t+1} &= angle_{i+1,j+1}^t \\ trail_{i,j}^{t+1} &= trail_{i,j}^t + depT \end{aligned} \quad (5)$$

The algorithm proceeds by calculating the sensory inputs of the cells. Each cell containing an agent has 3 sensors that are located in the neighborhood of this cell in the direction of the agent's angle. The neighborhood could be larger, e.g. by integrating all the cells that are spaced 2, 3, . . . cells away from the central one. Moreover, this would act as an escalation parameter in a large CA, a cell would receive values in its sensors from a more distant area and could direct its agent towards that direction (Fig. 1).

The sensors receive the trail values of neighboring cells in which they are located. Then those three variables are compared, in order to decide the highest food value from the trail of the neighboring cells. The prevailing sensor, will be the one who will also indicate which direction the agent should follow to reach the point of feeding.

If **f** sensor has the highest value, then the cell's angle retains the same value. If the value of **f** is less than the other two, then if **fl** is less than **f**, $angle_{i,j}^{t+1} = angle_{i,j}^t + 45$, otherwise, if **fr** is less than **f** then $angle_{i,j}^{t+1} = angle_{i,j}^t - 45$. Finally, if **f** is greater than **fl** and less than **fr** then $angle_{i,j}^{t+1} = angle_{i,j}^t + 45$, otherwise, if **f** is greater than **fr** and less than **fl** then $angle_{i,j}^{t+1} = angle_{i,j}^t - 45$.

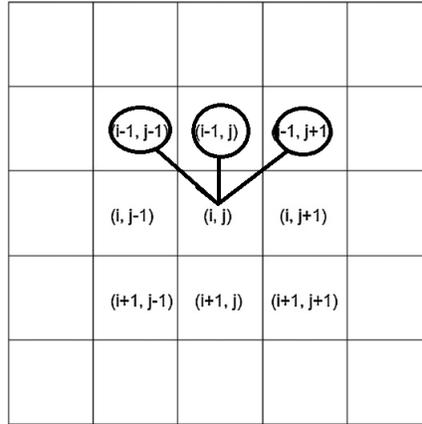


Fig. 1. Placement of 3 sensors in the Moore neighborhood of the cell according to its angle.

The final step of the algorithm is to update the trail of each cell, where its value is given by the equation of food diffusion.

$$trail_{i,j}^{t+1} = \{(trail_{i-1,j-1}^t + trail_{i,j-1}^t + trail_{i+1,j-1}^t + trail_{i-1,j}^t + trail_{i,j}^t + trail_{i+1,j}^t + trail_{i-1,j+1}^t + trail_{i,j+1}^t + trail_{i+1,j+1}^t)/9 * (1 - diffamp)\} \quad (6)$$

3 Hardware implementation of the proposed model

The proposed hardware system is comprised of the same basic structure representing a CA cell. Each cell circuit (Fig. 2) is interconnected with other cells in its vicinity.

This circuit is equipped with 46 inputs and 19 outputs. As inputs we consider eight signals, 9 bits each, that indicate an integer from 0 to 511, representing the identifier number of the agents; eight signals, 9 bits each, that indicate an integer number with range from 0 to 511, which represents the food value; eight signals, 9 bits each, that indicate an integer number ranging from 0 to 360, representing the angle of agents in the CA; eight signals one bit each, needed to inform the cell that an agent is going to move in its direction.

Eight additional signals one bit each that inform the central cell that the agent cannot move to a neighboring cell; an 8-bit signal used to characterize the central cell ($food = 255$, $wall = 51$, $normalcell = 0$); two signals, 9 bits each that are used to initialize agents and angles in the grid and three signals, 1 bit each for `clk`, `reset` and `initialize`.

As outputs there are eight signals, one bit each to inform neighbors that the agent will move in that direction; eight signals, one bit each indicating that

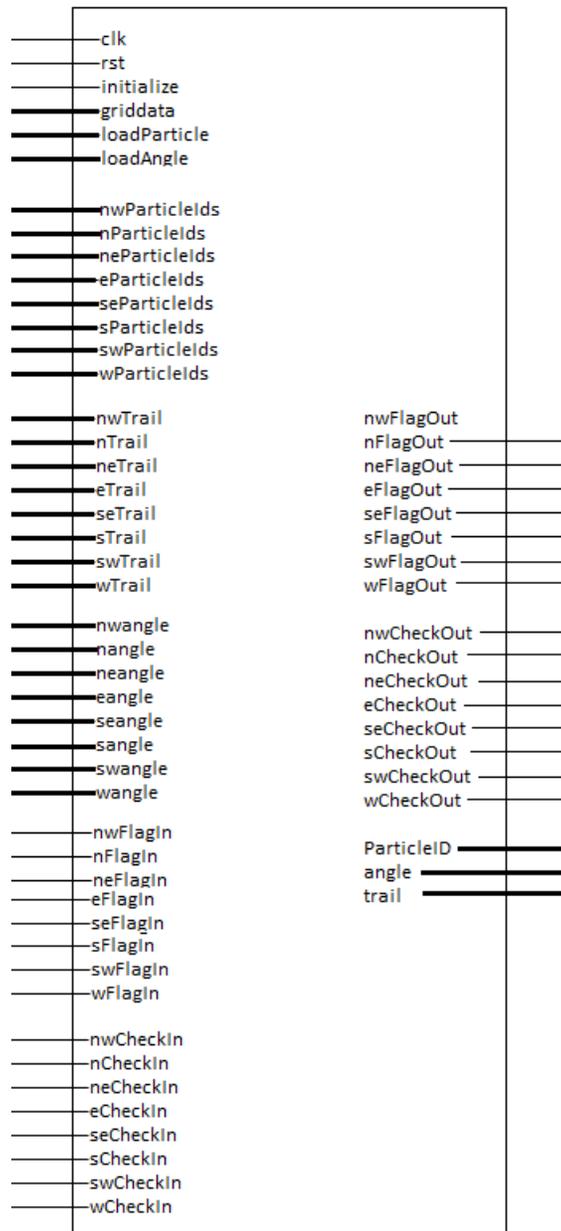


Fig. 2. Input and output signals of the circuit generated by the VHDL code.

the central cell is free to receive an agent and three signals, 9 bits each, which represent the identifier, the angle and the smell of the food in the central cell.

Firstly, `initialize` signal is set high, for cells to obtain random values needed. In the first stage, the cell in which the food stimuli is found, renews the trail value by noticing whether there is an agent in the neighborhood. The following stage is the motor behavior where each cell will signal if there is an agent to attempt to move in the direction which is indicated by the angle. If the agent cannot move, then the angle will be modified to a random value and the algorithm will continue.

However, if the agent can move, then the central cell will raise a flag to inform its neighbors. Two additional steps in relation to the software code are introduced to check whether two or more agents attempt to move to the central cell. In the next stage, if the previous prerequisites are met, the agent's identifier and angle of the neighboring cell will be transferred to the central cell and its trace value will be increased by $depT$. Afterwards, the sensory behavior of the food and the adjustment of the angle of each cell are followed. Direction is indicated by the higher smell of food detected by the sensors. Sensors f , fl and fr receive the trail of the neighboring cells and the front is impacted by the angle of the central cell. The final stage is the food diffusion equation.

A $n \times n$ grid of cells is designed and interconnected. Then the grid circuit received the following input signals: $p + 1$ ($0 \leq p \leq 511$) signals, 9 bits each, that indicate an integer with range from 0 to 511. These signals are used to load agents' identifiers into different cells within the grid. Another $p + 1$ signals are used to give angle values in the cells. An 8-bit signal set to zero, i.e. to the absence of food in a cell and another one 8-bit signal corresponding to the number 255 and given to the cells that we intend to have food stimulus. Three signals, one bit each corresponding to `clk`, `reset` and `initialize`. The outputs are $n \times n$ signals, 9 bits each representing the ID number of the agent in each cell of the grid.

In order to prove that the software program and the CA hardware circuit generated by the VHDL code have the same behavior, the following section presents the results of trying to create a Kolmogorov–Uspensky machine.

4 Simulation results

In this section, the proposed CA–multiagent model will be used to reproduce functions FUSE and MULT. These functions are characteristic of the *Physarum* machine presented in [18], which mimics universal storage modification machines, like the Kolmogorov–Uspensky machine. The CA is selected to have a size of 20×20 . The agents are placed in random locations and with random angles. The deposition value of the chemoattractant is equal to one for each successive transfer of the particle's identifier from one cell to another. The input signals given to the system designed with the VHDL code were: the period of the `clk` signal is set to 1 ns, the `rst` signal for 9ns equal to 1 and then 0, the `initialize` signal equal to one for 10 ns and then equal to 0.

The experimental operations with the active zones as recorded by Adamatzky and Jones [17, 18] were used as verification of the model.

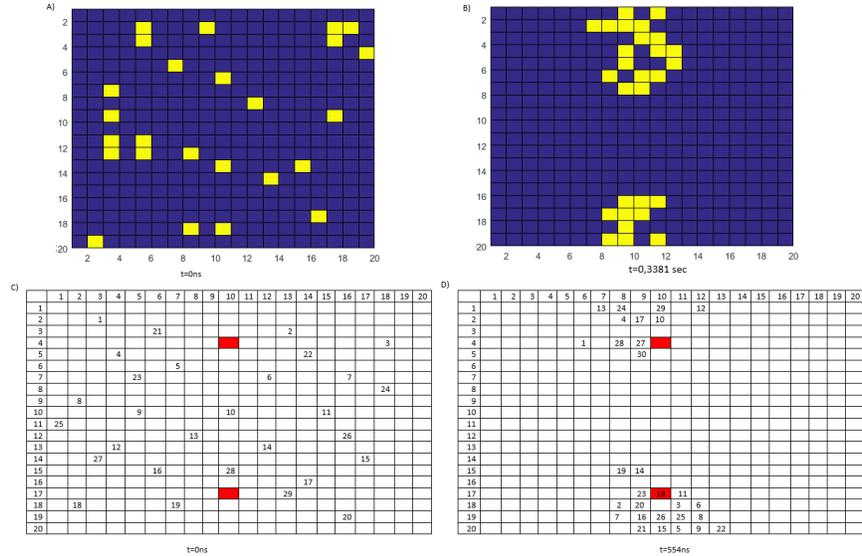


Fig. 3. Output of the model in software (a,b) and in hardware (c,d) with initial and final positions of agents.

Initialization of the CA is illustrated in Fig. 3a for the software model and in Fig. 3c for the hardware model. It is noted that cell values that are closer to food sources will modify their status and will undergo changes in both identifiers and angles. As already mentioned the output signals are 400, so for convenience, all the changes noted in the cells are shown in a table. In the example of Fig. 3, each agent has a sensory behavior of a larger neighborhood (3 cells), so that it can detect the smell of the food of another cell on the grid without staying only around a constant food stimuli. The pieces of plasmodium are attracted by the food sources (red color) and plasmodium spots are created around these sources. The food stimuli are initially placed in the cells (4, 10) and (17, 10).

By comparing these two models, we observed that they present similar results. On the other hand, the time spent by the CA circuit to form the result (554 ns) is much less than the time spent in the software (0.3381 s).

In the following, a new stimulus is placed to attract the plasmodium in its location. The new food source is installed in cell (10, 10). The growing plasmodium is attracted by the inner source and the propagation continues inwards from each initial source (Fig. 4). These two active zones fuse and retain a structure spanning the array of nodes. After a few time steps, it is observed that the cells near the new food source are starting to change their ID values, which means that the active zones migrate in these directions. In software the chain structure is formed at 0.7366 s (Fig. 4.b), while in hardware at 786 ns (Fig. 4.d).

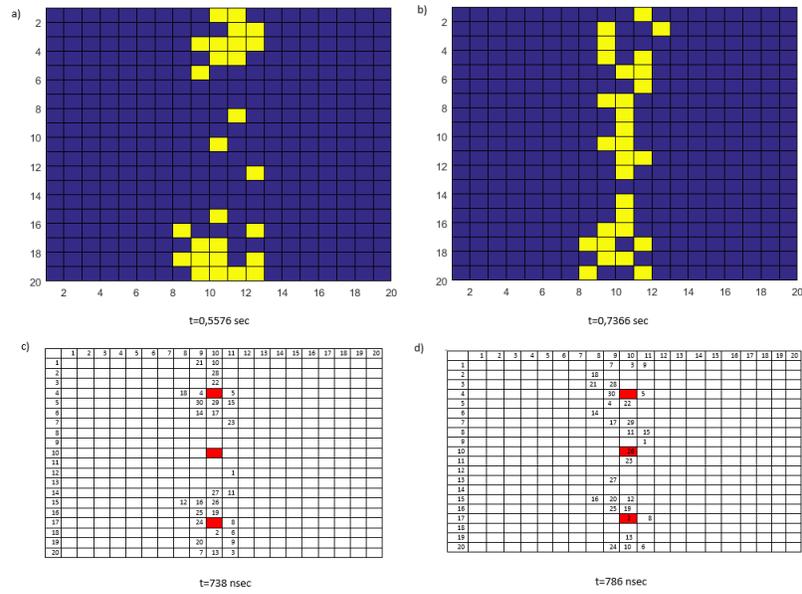


Fig. 4. The FUSE operation in software (a,b) and in hardware (c,d).

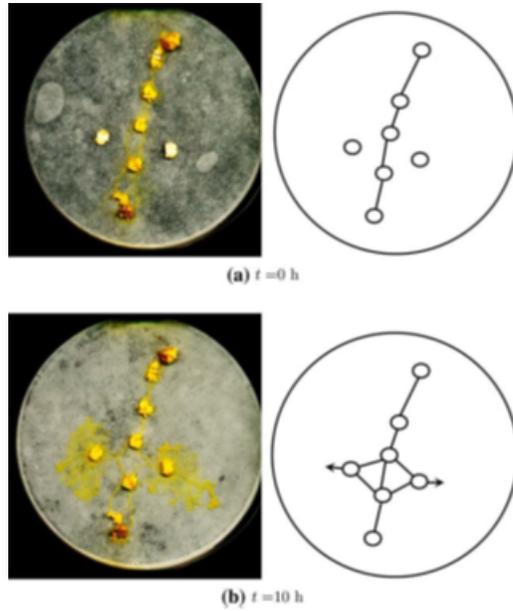


Fig. 5. The MULT operation of the Physarum machine (adopted from [18]).

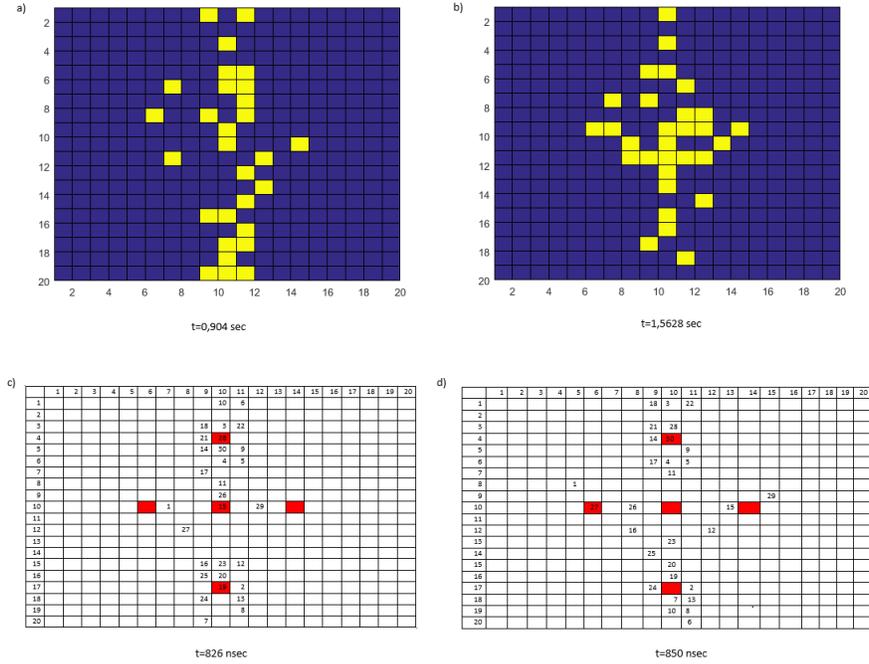


Fig. 6. (a,b,c,d). MULT operation of the virtual model. At the time of 1.5628 sec the formation result for the software program (top) is produced, while for the program that is designed with hardware description language (below) the formation of the network is generated at 850 nsec.

An example is displayed in Fig. 5, given a food source chain where the plasmodium has formed protoplasmic tubes, two oat flakes can be added right and left and cause new active zones (Fig. 5.a). After 10 hours, 2 new active zones A1 and A2 are formed (Fig. 5.b).

Using the chain structure created by the previous method (Fig. 4) two additional food sources are added to each side of the array, at points (10, 6) and (10, 14). Two active fronts were produced to engulf the sources (Fig. 6). In Fig. 6 is illustrated the function MULT of physarum machine, some agents will gradually become aware of the existence of new food stimuli and will start moving in that direction developing a diamond area.

5 Conclusions

A novel hybrid model was used to approximate the computing abilities of a (geometrically represented) biological substrate, namely the plasmodium of *P. polycephalum*. The model proposed here is a CA-based method incorporating a multi-agent system. The results of the computerized plasmodium were closely

imitating the results obtained from in vivo experimental studies. The developed methodology was implemented in software as well as in hardware. The motivation was the fact that the parallel nature of CA is lost in the software implementation. The higher the complexity of the problem, the longer it will take to be resolved. Whereas, the circuit generated by the VHDL code uses the advantage of parallel processing and, therefore, solves the problem within a vast range of complexity, using the resources for a given amount of time.

References

1. Adamatzky, A.: *Physarum Machines: Computers From Slime Mould*. Volume 74. World Scientific (2010)
2. Nakagaki, T., Yamada, H., Toth, A.: Path finding by tube morphogenesis in an amoeboid organism. *Biophysical chemistry* **92**(1-2) (2001) 47–52
3. Adamatzky, A.: *Advances in Physarum Machines: Sensing and Computing with Slime Mould*. 1st edn. Springer Publishing Company, Incorporated (2016)
4. Nakagaki, T., Yamada, H., Tóth, Á.: Intelligence: Maze-solving by an amoeboid organism. *Nature* **407**(6803) (2000) 470
5. Adamatzky, A.: Slime mold solves maze in one pass, assisted by gradient of chemo-attractants. *IEEE transactions on nanobioscience* **11**(2) (2012) 131–134
6. Adamatzky, A.: Developing proximity graphs by physarum polycephalum: does the plasmodium follow the toussaint hierarchy? *Parallel Processing Letters* **19**(01) (2009) 105–127
7. Aono, M., Zhu, L., Hara, M.: Amoeba-based neurocomputing for 8-city traveling salesman problem. *International Journal of Unconventional Computing* **7**(6) (2011) 463–480
8. Tsuda, S., Aono, M., Gunji, Y.P.: Robust and emergent physarum logical-computing. *BioSystems* **73**(1) (2004) 45–55
9. Adamatzky, A.: *Bioevaluation Of World Transport Networks*. World Scientific (2012)
10. Evangelidis, V., Tsompanas, M.A., Sirakoulis, G.C., Adamatzky, A.: Slime mould imitates development of roman roads in the balkans. *Journal of Archaeological Science: Reports* **2** (2015) 264–281
11. Tsompanas, M.A.I., Mayne, R., Sirakoulis, G.C., Adamatzky, A.I.: A cellular automata bioinspired algorithm designing data trees in wireless sensor networks. *International Journal of Distributed Sensor Networks* **11**(6) (2015) 471045
12. Tero, A., Takagi, S., Saigusa, T., Ito, K., Bebbber, D.P., Fricker, M.D., Yumiki, K., Kobayashi, R., Nakagaki, T.: Rules for biologically inspired adaptive network design. *Science* **327**(5964) (2010) 439–442
13. Jones, J.: *From Pattern Formation to Material Computation: Multi-agent Modelling of Physarum Polycephalum*. Volume 15. Springer (2015)
14. Gunji, Y.P., Shirakawa, T., Niizato, T., Yamachiyo, M., Tani, I.: An adaptive and robust biological network based on the vacant-particle transportation model. *Journal of theoretical biology* **272**(1) (2011) 187–200
15. Tsompanas, M.A.I., Sirakoulis, G.C.: Modeling and hardware implementation of an amoeba-like cellular automaton. *Bioinspiration & Biomimetics* **7**(3) (2012) 036013
16. Tsompanas, M.A.I., Sirakoulis, G.C., Adamatzky, A.: Cellular automata models simulating slime mould computing. In: *Advances in Physarum Machines*. Springer (2016) 563–594

17. Adamatzky, A.: Physarum machine: implementation of a kolmogorov-uspensky machine on a biological substrate. *Parallel Processing Letters* **17**(04) (2007) 455–467
18. Adamatzky, A., Jones, J.: Programmable reconfiguration of physarum machines. **9**(1) (March 2010) 219–237
19. Kolmogorov, A.N.: On the concept of algorithm. *Uspekhi Mat. Nauk* **8**(4) (1953) 175–176
20. Kolmogorov, A.N., Uspenskii, V.A.: On the definition of an algorithm. *Uspekhi Matematicheskikh Nauk* **13**(4) (1958) 3–28
21. Blass, A., Gurevich, Y.: Algorithms: A quest for absolute definitions. *Bulletin of the EATCS* **81** (2003) 195–225
22. Jones, J.: Approximating the behaviours of physarum polycephalum for the construction and minimisation of synthetic transport networks. In: *International Conference on Unconventional Computation*, Springer (2009) 191–208