A Method of Hardware Implementation of Membrane Computing Architectures for Mobile Robot Control

Victor ABABII, Viorica SUDACEVSCHI, Viorel CARBUNE
Computer Science and Systems Engineering Department,
Technical University of Moldova,
Chisinau, Republic of Moldova
victor.ababii@calc.utm.md,
viorica.sudacevschi@calc.utm.md,
viorel.carbune@calc.utm.md

Silvia MUNTEANU, Victoria ALEXEI, Victor LASCO
Computer Science and Systems Engineering Department,
Technical University of Moldova,
Chisinau, Republic of Moldova
silvia.munteanu@calc.utm.md,
victoria.alexei@iis.utm.md,
victor.lasco@iis.utm.md

Abstract— **This paper proposes a method of hardware implementation of the membrane computing architecture for the control of a mobile robot. The basic idea is to use in the development of control systems the models of functional description of living cells, which should ensure a process of design, modeling and implementation based on cognitive models. The implementation algorithm presents a sequence of operations involving: functional development and modeling of Computing Cells, development and modeling of the topology of the membrane computing system (P-System), their implementation in hardware description languages (AHDL), and configuration of the FPGA circuit for realization of the control system. For validation and analysis of the performance of the Computing Cells, Petri Net models are used, which ensure the identification of the concurrent processes while maintaining a maximum parallelism. Functional testing of membrane computing models was performed based on the Quartus II development environment, the AHDL hardware description language, and the Altera DE0 Board.**

Keywords—membrane computing; mobile robot control; hardware implementation; HDL; FPGA; DE0.

I. INTRODUCTION

The Membrane Computing Paradigm was first proposed in 1998 by academician Gheorghe Păun [1, 2, 3]. Over time, these ideas have developed into a separate branch of theoretical calculus [4, 5], which today offers the most advanced models for the development of computational systems based on computation inspired by living biological cells. [6, 7].

The evolution of nature and especially of living organisms has inspired researchers to develop new models and methods for designing decision-making systems or for processing data. Evolutionary systems have provided a lot of properties that are the basis of many applications inspired by nature, namely: dynamics, flexibility, robustness, self-organization, simplicity of basic objects and decentralization. Another important aspect of living organisms is that they always live in conditions of extreme competition. Therefore, they may be subject to actions that may partially or completely destroy the population, but in the most of cases, they can be recovered by fully restoring themselves. The ability to recover is determined by the presence of the immune system of individuals or the interactions within and between populations. Resource limitation is one of the selection criteria that can ultimately lead to more self-organization and a more efficient recovery [8, 9, 10].

Membrane computing models offer the possibility to formally and structurally describe computing and control systems with a diverse complexity. For example: logic structures and specialized processors; reconfigurable computer systems; complex computing architectures, which include functional elements with hierarchical interaction; parallel and concurrent computing architectures; and network topologies for distributed computing systems. Membrane computing cells are considered systems with artificial intelligence and cognitive properties that they implement as rules for data processing, both mathematical and logical models, as well as models based on neural networks, fuzzy logic and evolutionary computation etc. [11, 12].

The advantage of Membrane Computing models consists in the application of the unconventional computing concept which presents a very dynamic field of research. As mentioned in [6, 7] the main motivation for applying the Membrane Calculus is the massive parallelism and the hierarchical ratio of the Cells.

II. STATEMENT OF THE PROBLEM FOR HARDWARE IMPLEMENTATION OF MEMBRANE COMPUTING ARCHITECTURES

The scientific ideas stated above have created essential premises for the development of new directions for the application of the concept of Membrane Computing for the design of control or decision-making systems. The advantages of applying membrane computing can only be realized by organizing data processing processes in parallel or concurrent mode which can only be achieved based on multi-processor or FPGA devices. Since the data processing algorithms based on membrane computation models have a specific character, it is obvious the use of FPGA devices in which the topological and functional specifications of the membranes (living cells) are respected.

The paper proposes a method of hardware implementation of membrane computing models for the control of a mobile robot [13], which moves in a limited space of activity and with obstacles. The perception of the activity space is based on a set of ultrasonic sensors. The direction of movement of the mobile robot is determined by the rotational speed of the drive wheels. A FPGA device will be used to implement the membrane computing architecture.

III. ALGORITHM FOR HARDWARE IMPLEMENTATION OF MEMBRANE COMPUTING ARHITECTURES

The objectives of this paper are to demonstrate the implementation capacity in hardware computing architectures of membrane computing models. Most of the researches in this field are focused towards the development of abstract membrane computing models, their implementation in hardware architectures has an important advantage in their application in various areas for control and decision-making functions. The algorithm that includes the main operations of the process of implementing membrane computing models in hardware architectures is presented in Figure 1.

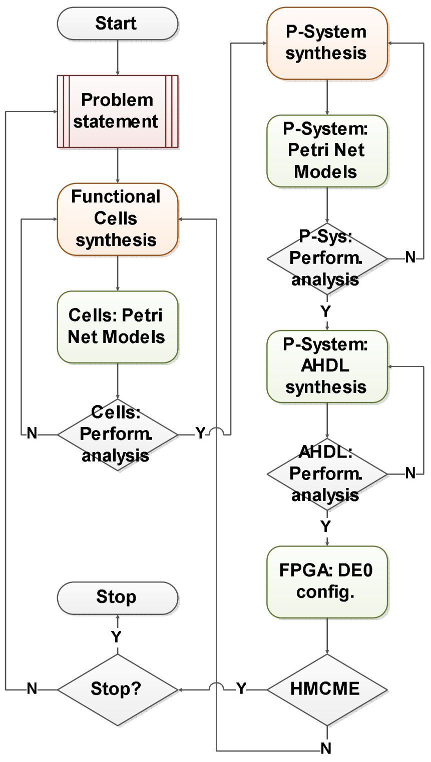

Fig. 1. Algorithm diagram for hardware implementation of membrane computing architectures.

Description of synthesis operations. The process of hardware implementation of membrane computation models begins with the formulation of the design statement (***Problem statement***) which involves a formal description based on models inspired by nature, especially the behavior of living cells. Based on the formal description model, the functional diagram of the Computing Cells (***Functional Cells synthesis***) is performed, in which the functional blocks, the interaction between them and the synchronization mode are specified. The functional diagram of the Cells serves as a model for the elaboration of the Petri Net model (***Cells: Petri Net Models***) [14, 15]. In turn, the Petri Cell Model is used to evaluate performance (***Cells: Perform. analysis***) and extract features of parallelism and concurrency. Subsequently, the cells obtained are used for P-System synthesis (***P-System***) which is used as a control or decision-making system. The evaluation of the performance of the P-System and the extraction of characteristic parameters occur on the basis of the Petri Net Models (***P-System: Petri Net Models***) [16]. The next step in the synthesis algorithm is the transformation of the P-System into AHDL code (***P-System: AHDL Synthesis***), the compilation, the extraction of the wiring diagram and time diagrams, and their functional and parametric analysis (***AHDL: Perform. analysis***) [17]. The last step in the synthesis algorithm is the configuration of the FPGA circuit, which can be part of the structure of a Development Kit (***FPGA: DE0 config.***) [18] and the functional testing / evaluation of the Hardware Membrane Computing Model Evaluation architecture (***HMCME***).

IV. SYNTHESIS OF THE COMPUTING CELLS

Membrane computing architectures present a lot of Computing Cells that work on the basis of an individual algorithm solving a small part of a complex algorithm. The complexity and algorithmic performances of the membrane computing architecture are determined by the performances of the Computing Cells, the topology of the membrane computing architecture, the rules of synchronization and communication between Cells [19, 20, 21].

Figure 2 shows the Computing cell diagram that includes:

$X[T]$ - vector of input data generated by activity environment;

Data Input – input data selection operations (are associated with the functions of a biological membrane that allows the entry of chemical compounds into the biological Cell);

Input Memory – memory for storing input data;

Knowledge Models – knowledge models that determine the sequence of clock signals to synchronize the execution of the rules of operation of the Cell;

Data Processing – data processing logic;

Output Memory – memory for output data storage;

C(DI) – clock signals for synchronization of data input operations;

C(IM) – clock signals for synchronization of data storage operations in the input memory;

C(DP) – clock signals for synchronization of operations performed by the data processor;

C(OM) – clock signals for synchronization of data storage operations in the output memory;

C(DO) – clock signals for synchronization of operations of data transmission in the activity environment;

DI – flow of input data for processing;

DO – flow of output processed data;

DP – data flow for loop formation for repeated data processing;

$Y[T+1]$ – vector of the output data or action on the activity environment.

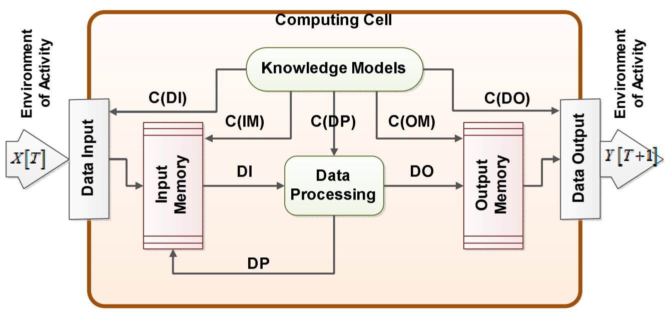

Fig. 2. Computing Cell diagram.

V. Synthesis of the Mobile Robot Control System Based on Membrane Computing

The algorithmic complexity of a membrane computing system depends on the data processing rules performed by each membrane and the interconnection between them. Information sources in the field provide several methods for graphically and formally describing a membrane computational model [22]. The Venn diagram (Figure 3) is used in this paper to describe the topology of the membrane computing system for controlling a mobile robot.

The Venn diagram of the membrane computing system architecture for driving a mobile robot includes 12 membranes from which 9 are elementary and respectively, 3 complexes, which group the Cells into a topological structure. For each Cell, the rules are specified on which the input data is processed and transmitted to the output:

$R1$ - evolution rule of activity environment;

$R2, R3, R4$ - rules for controlling the ultrasonic sensors set (***US1, US2 and US3***) and identifying the distance to obstacles;

$R6, R7, R8$ and $R10, R11, R12$ - rules for processing data obtained from the ultrasonic sensors set;

$R5, R9$ - rules for the formation of command signals with the moving motors of the mobile robot.

According to the functioning algorithm of membrane computing models, all cells operate in parallel and concurrently. This ensures the expected performances by implementing the membrane computing system for controlling a mobile robot.

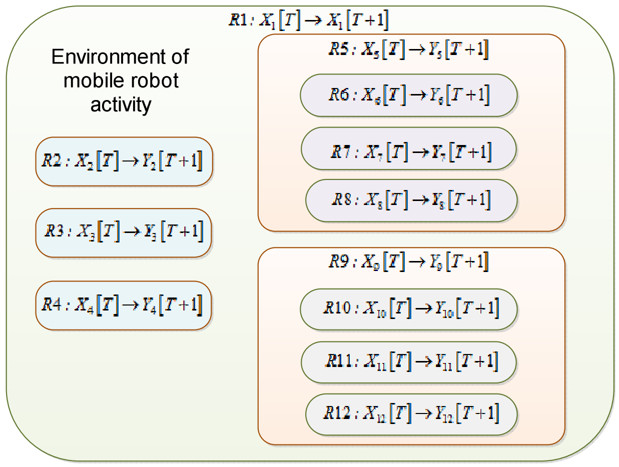

Fig. 3. Mobile robot control system based on Membrane Computing architecture.

VI. Validation and Performance Analysing of Control System

The validation and analysis of the performances of the control system of the mobile robot based on the membrane computing architectures was performed with the help of Petri nets models [14, 15], especially with the help of Visual Petri Net+ application [16]. The result of the elaboration of the Petri Net model is presented in Figure 4.

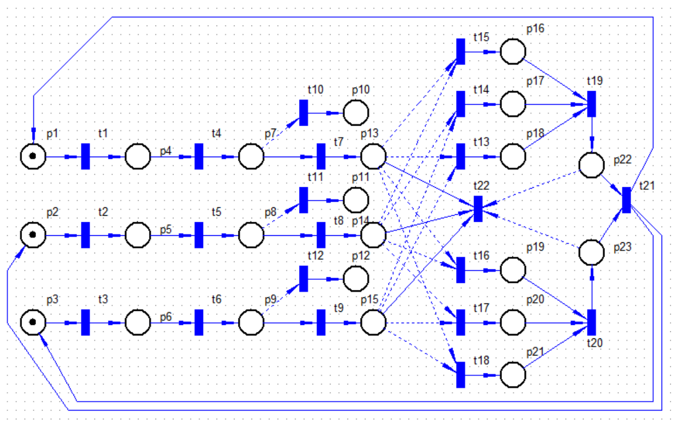

Fig. 4. Petri net model for validation and performance analysis of control system based on Membrane Computing architecture.

The Petri net model (Figure 4) repeats the topology of the control system (Figure 3) and includes the following objects:

$p1, t1, p4, t4 p7, t7, p13, t10, p10$ - model the functionality of the Cell defined by the rule $R2$;

$p2, t2, p5, t5 p8, t8, p14, t11, p11$ - model the functionality of the Cell defined by the rule $R3$;

$p3, t3, p6, t6 p9, t9, p15, t12, p12$ - model the functionality of the Cell defined by the rule $R4$;

$\{t1, t2, t3\}$ - timed transitions which model ***Trig*** impulse with ***10us*** duration;

$\{t4, t5, t6\}$ - timed transitions which model generation of the ultrasonic signal by the sensor HC-SR04;

$\{t11, t12, t13\}$ - timed transitions which model the duration of ***Echo*** signal;

$\{t10, p10\}$, $\{t11, p11\}$ and $\{t12, p12\}$ - model the counter to calculate the propagation time of the ultrasonic signal to the obstacle and back;

$\{t15, p16\}$, $\{t14, p17\}$ and $\{t13, p18\}$ - model the functionality of the Cells with the rules $R6, R7, R8$, where $\{t15, t14, t13\}$ are timed transitions that model the time for processing data;

$\{t16, p19\}$, $\{t17, p20\}$ and $\{t18, p21\}$ - model the functionality of the Cells with the rules $R10, R11, R12$, where $\{t16, t17, t18\}$ are timed transitions that model the time for processing data;

$\{t19, p22\}$ and $\{t20, p23\}$ - model the functionality of the Cells with the rules $R5, R9$, where $\{t19, t20\}$ are timed transitions that model the time for processing data;

$t21$ - timed transition for the synchronization of a new control cycle.

From the Petri Net model, it can be seen that most of the operations are performed in parallel / concurrently which ensures the achievement of the performance objectives.

VII. Functional Testing of the Method for Hardware Implementation of Membrane Computing Architectures

Altera DE0 Board [18] has been selected for the functional testing of membrane computing models, which provides the user with the following hardware resources:

- ***A Cyclone III 3C16 FPGA device with the following parameters:*** 1,408 LEs, 56 M9K Embedded Memory Blocks, 504K total RAM, 56 Embedded Multipliers, 4 PLLs, 346 user I/O pins;
- ***Memory:*** SDRAM – 8MB and Flash – 4MB;
- ***User Interfaces:*** 10 Slide Switches, 10 Green Color LCDs, 4 Seven-segment Displays, 16x2 LCD Interface;
- ***Two 40-pin expansion headers***: 72 Cyclone III 3.3V I/O, 8 power and Ground lines, are brought out to two 40-pin expansion connectors.

The functional testing diagram of the membrane computing models for controlling a mobile robot is presented in Figure 5, where: ***US1, US2 and US3*** are three ultrasonic sensors for measuring the distance to obstacles; development board - ***Altera DE0 Board***; ***L298N*** engine Driver module; and two ***DC*** motors for moving the mobile robot.

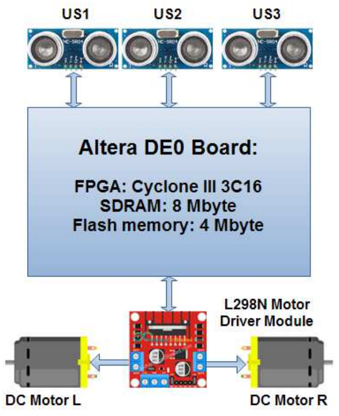

Fig. 5. Functional diagrame for membrane computing model testing.

The connection of the set of sensors ***US1, US2 and US3*** to Altera DE0 Board kit was done through Expansion Header J5 (GPIO 1). Table 1 shows the interconnection of the signals between the sensors set and the Altera DE0 Board. The entire sensors set are connected to 3.3V and GND pins for power supply.

TABLE I. CONNECTING OF SENSORS TO THE ALTERA DE0 BOARD

DE0 GPIO1	FPGA pin	Sensor	Pin Signal
GPIO1_D0	AA20	***US1***	Trig1
GPIO1_D1	AB20	***US1***	Echo1
GPIO1_D2	AB19	***US2***	Trig2
GPIO1_D3	AA18	***US2***	Echo2
GPIO1_D4	AA19	***US3***	Trig3
GPIO1_D5	AB18	***US3***	Echo3

Connection of ***L298N*** motor Driver module to ***Altera DE0 Board*** kit was performed according to Table 2.

TABLE II. CONNECTING OF MOTOR DRIVER TO THE ALTERA DE0 BOARD

DE0 GPIO1	FPGA pin	Motor Driver	Pin Signal
GPIO1_D31	V7	DC1	IN1
GPIO1_D29	U8	DC1	IN2
GPIO1_D27	T9	DC2	IN3
GPIO1_D25	T10	DC2	IN4

The contents of Tables I and II are used in the HDL code connection process, which implements Membrane Computing logic, to the *Cyclone III FPGA* architecture for interaction of the control system with the external environment (sensor set and action device).

Figure 6 shows the picture of the system for testing membrane computing models which includes: ***1*** - PC computer with Quartus II design environment installed; ***2*** - mobile robot platform; ***3*** - Altera DE0 Board; ***4*** - the sensors set (US1 - scans the obstacles placed on the left side of the mobile robot, US2 - scans the obstacles placed in front of the mobile robot and US3 - scans the obstacles placed on the right side of the mobile robot); ***5*** - two DC motors; ***6*** - L298N Motor Driver Module; *7* - power supply.

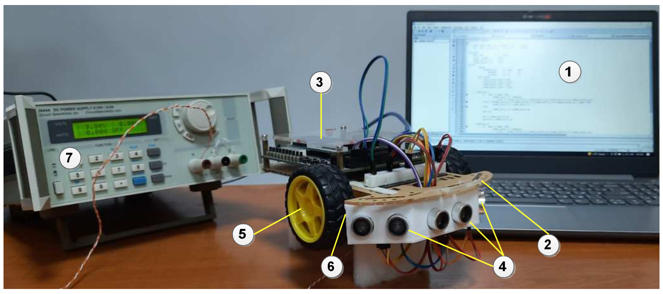

Fig. 6. Picture of the system for testing membrane computing models.

Conclusion

Mankind, by making continuous observations on the evolution and behavior of living organisms, has come to the conclusion that they offer the most efficient and optimal models and solutions for leading complex/concurrent processes or making decisions. Having a heuristic character, these models also create some problems in the implementation process, especially if they have to be implemented in hardware computing architectures. This paper describes a method of implementing membrane computing systems with parallel/concurrent data processing in hardware architectures for controlling a two-wheeled mobile robot.

The method proposed in the paper involves a sequence of operations that includes: problem formulation, synthesis of Computing Cells, their functional and parametric evaluation, synthesis of the membrane computing model and functional and parametric evaluation, synthesis of the AHDL code of the membrane computing architecture, and configuration of FPGA circuit to obtain the control system.

The advantages of the proposed method are the application of several stages of functional and parametric evaluation of the Cells and of the membrane computing architecture: Petri nets, Quartus II and Altera DE0 Board, which will exclude the occurrence of errors and competition.

Acknowledgment

The research was carried out with the technical and technological support provided by the Computer Sciences and Systems Engineering Department, Technical University of Moldova within the project: *20.80009.5007.26 „Models, algorithms and technologies for the control, optimization and security of the Cyber-Physical systems".*

References

[1] Păun, Gh. ”Computing with Membranes”. TUCS Report 208. Turku Centre for Computer Science. 1998. ISBN: 978-952-12-0303-9.

[2] Păun, Gh., Rozenberg, G. ”A guide to membrane computing”. Theoretical Computer Science. 287 (1): 73-100, 2002. DOI: 10.1016/S0304-3975(02)00136-6. ISSN: 0304-3975.

[3] Păun, Gh. ”Introduction to Membrane Computing”. Applications of Membrane Computing. Springer Berlin Heidelberg. pp. 1-42. ISBN: 978-3-540-29937-0.

[4] Alhazov, A. ”Communication in Membrane Systems with Symbol Objects”. Ph.D. Thesis. Tarragona, Spain: Universitat Rovira i Virgili, 2006, 218p.

[5] Alhazov, A. ”Small Abstract Computers. Habilitation” Thesis. Chișinău (RM): Academy of Sciences of Moldova, 2013, 255p.

[6] Siddique, N., Adeli, H. ”Nature Inspired Computing: An Overview and Some Future Directions”. Cognitive Computing, 2015, 7: 706-714. DOI: 10.1007/s12559-015-9370-8.

[7] De Castro, L.N. ”Fundamentals of natural computing: an overview”. Phys Life Rev. 2007; 4: 1–36. doi: 10.1016/j.plrev.2006.10.002.

[8] Păun, Gh., Rozenberg, G., Salomaa, A. ”DNA Computing: New Computing Paradigm”. Texts in Theoretical Computer Science (An EATCS Series). Springer, Berlin, Heidelberg. 1998, 409p. ISBN: 978-3-540-64196-4.

[9] Ridley, M. ”Evolution”. Third edition, Blackwell, 2004, 751p. ISBN: 1-4051-0345-0.

[10] Futuiama, D.J. ”Evolution”. Third edition, Sinauser Associates, 2013, 655p.

[11] Dinneen, M. J., Kim, Y.-B. and Nicolescu, R. ”An Adaptive Algorithm for P System Synchronization”. In: Proceedings of the Twelfth International Workshop on Membrane Computing, (CMC11), Fontainebleau/Paris, France, 2011, pp. 1–26.

[12] Wang, T., Zhang, G., Perez-Jimenez, M.J. ”Fuzzy Membrane Computing: Theory and Applications”. International Journal of Computers Communications & Control, 2015, Vol. 10(6), pp. 144, DOI: 10.15837/ijccc.2015.6.2080.

[13] ”A Generic Guide for Using Membrane Computing in Robot Control”. Advances in Computational Intelligence and Robots – Membrane Computing for Distributed Control of Robotic Swarms, 2017, pp.86-96. DOI: 10.4018/978-1-5225-2280-5.ch005.

[14] Diaz, M. ”Petri Nets: Fundamental Models, Verification and Applications”. Wiley, 2009, 613p. ISBN: 978-1-84821-079-0, DOI: 10.1002/9780470611647.

[15] Blatke M.A., Heiner M., MarwanW. ”Tutorial: Perti Nets for Systems Biology”. 1st Edition, 2011, 100p.

[16] Guțuleac, E., Boşneaga, C., Railean, A. ”VPNP-Software tool for modeling and performance evaluation using generalized stochastic Petri nets”, In: Proceedings of the 6-th International Conference on DAS-2002, 23-25 May 2002, Suceava, România, p. 243-248, ISBN: 973-98670-9-X.

[17] Cofer, R.C., Harding, B.F. ”Rapid System Prototyping with FPGAs. Accelerating the Design Process”. ScienceDirect 2006, 301p., ISBN: 978-0-7506-7866-7, DOI: 10.1016/B978-0-7506-7866-7.X5000-8.

[18] Altera DE0 Board. [online]. [accessed 27.02.2022]. Available: https://www.terasic.com.tw/cgi-bin/page/archive.pl?No=364.

[19] Ababii, V., Sudacevschi, V., Munteanu, S., Borozan, O., Nistiriuc, A., Lasco, V. ”IoT based on Membrane Computing Models”. Proceedings of the 13th International Conference on Electromechanically and Energy Systems (SIELMEN-2021), 7-8 October, 2021, Chisinau, Republic of Moldova, pp. 010-014, ISBN: 978-1-6654-0078-7. DOI: 10.1109/SIELMEN53755.2021.9600341.

[20] Munteanu, S., Sudacevschi, V., Ababii, V., Borozan, O., Ababii, C., Lasco, V. ”Multi-Agent Decision Making System based on Membrane Computing”. The 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications. 22-25 September, 2021, Cracow, Poland, Vol. 2. pp. 851-854. ISBN: 978-1-6654-4210-7.

[21] Sudacevschi, V., Munteanu, S., Ababii, V., Braniste, R., Borozan, O., Alexei, V. ”Cognitive Computing System based on Distributed Knowledge”. In the Extended Abstracts of the 10th International Conference on Electronics, Communications and Computing ECCO-2019, 23-26 October 2019, Chisinau, Moldova, pp. 98, ISBN: 978-9975-108-84-3.

[22] Gaivitto, J.-L., Olivier, M. ”The Topological Structure of Membrane Computing”. Repport de Recherche No. 70-2001. 21p.