Methods of information systems synthesis

UDC 004.312.2, 004.383 (577.218) **doi:** https://doi.org/10.20998/2522-9052.2023.3.03

Oleg Vasylchenkov, Dmytro Salnikov, Dmytro Karaman

National Technical University «Kharkiv Polytechnic Institute», Kharkiv, Ukraine

ACCELERATION OF BOOLEAN GENE REGULATORY NETWORKS ANALYSIS USING FPGA

Abstract. Gene expression does not occur arbitrarily and spontaneously, it obeys certain patterns that can be expressed as a connected graph or network. The disclosure of these patterns requires a large amount of experimental research and accumulation of necessary statistical information. Then this information is subjected to mathematical processing, which involves significant computing resources and takes a lot of time. Boolean networks are often used as the basis for building mathematical models in those calculations. Recently, models based on Boolean networks have increasingly grown in size and complexity causing increased demands on traditional software solutions and computing tools. Field-programmable gate arrays (FPGAs) are a powerful and reconfigurable platform for implementing efficient and high-performance computing. The use of FPGA will significantly speed up the process of calculating sequential chains of gene states, both through the use of hardware acceleration in the calculation of logical dependencies, and through the implementation of an array of parallel computing cores, each of which can perform its own individual task. Another solution that can significantly simplify the work of researchers of gene regulation networks is the creation of a universal computing architecture that will allow dynamic reconfiguration of its internal structure when the task or logical dependencies for the current Boolean network change. Such a solution will relieve the researcher of the need to perform the entire set of actions for the technological preparation of a new FPGA configuration, from making changes to the HDL code that describes the network to uploading the updated configuration to the hardware accelerator. The article discusses how to use FPGA for the implementation and modeling of arbitrary Boolean networks, describes the concept of a universal reconfigurable architecture of a logical dependency calculating core for an arbitrary Boolean network and proposes a practical implementation of such a calculating core for modeling gene regulation networks.

Keywords: gene regulatory network; biological system modeling; field-programmable gate arrays; hardware acceleration architectures; Boolean network model; computational biology systems; bioinformatics.

Introduction

Boolean models are widely used in modern research to model the behavior of the complexity of dynamical systems. A Boolean model is a structure that consists of simple nodes interconnected. Each node is a simple object that can be in one of two states – active or inactive. It is convenient to encode such an object in Boolean terms of states – an active state (1) and an inactive state (0). A change in the state of a node occurs under the influence of the input states of neighboring nodes according to certain rules. It is convenient to represent such rules as a Boolean function of several arguments. The result of evaluating the value of such a function will be a value that is the new state of the node. Thus, the model of some dynamical system can be a combination of very simple objects with a small number of possible states and transitions between states that are simple from the point of view of the computational requirements of transformation. Another advantage of Boolean models is the ease of scaling the model of a dynamic system. Adding new nodes to expand the system model is not a time-consuming process. Studies of the behavior of Boolean networks for system modeling have shown that such networks are characterized by certain features. Long-term monitoring of the states of the simulated system demonstrates that the system eventually reaches stable states or generates attractors. An attractor is a stable cyclic state, when a set of identical states is repeated for a fixed number of steps. Moreover, if there is some trajectory of states in the observed cycle of states, then this trajectory cannot leave this cycle of states, see, for example, [1–3]. The search for attractors is the main task in modeling a complex dynamic system. This task can be computationally difficult. In addition, when the system is expanded, the configuration and behavior of attractors can completely change. Also, in the most general case, the duration of the transition state leading to the attractor can be long [4]. It should also be noted that the dynamic system model built using the Boolean system is stochastic. Therefore, the simulation process must be run several times to get the average dynamic behavior of the system. Thus, the total simulation time can be significant.

It can be concluded that the use of Boolean models is a convenient tool for modeling complex dynamic systems, since the model of such a system based on a Boolean network is a simple combination of functionally simple nodes with an elementary set of states. The disadvantages of using Boolean models include a significant increase in the computational complexity of the model with an increase in the size of the model, the non-triviality of the search for the key characteristics of the model, which include attractors inherent in the model and static states.

One of the approaches in the study of complex diseases is now the analysis of behavior in gene regulatory networks (GNR). In the literature, there are studies based on data sources for models of such diseases as models of T-cell leukemia of large granular lymphocytes [5], prostate cancer [6], signaling pathways involved in cancer [6, 7], colon cancer [8], Fanconi's anemia and breast cancer [9]. Genes are quite simple chemical and biological objects in terms of a set of states that interact with each other, thus forming the genotype of an organism. Boolean models [10] are a possible alternative approach to the study of GRN and are used in the systems biology community [11–16]. Boolean network models can help build a qualitative description of the GRN. The concentrations or activities of chemicals can be represented using a finite set of discrete values. When constructing a Boolean GNR model, synchronous and asynchronous schemes for updating cell states can be applied. The synchronous circuit is simpler and more understandable. Time in such a scheme is discrete and the state of all cells of the model is updated at the same time for all elements of the model. A Boolean network with a synchronous circuit is easier to build and easier to model. But a model with a synchronous scheme is unlikely to be adequate to a real biological object with its inherent variety of states. An asynchronous scheme for updating model states seems to be more adequate. The asynchronous scheme [17] takes into account that the update of states occurs at arbitrary times and this is due to the different reaction rates of biological systems. In [18], it is proposed to distinguish between various asynchronous state update schemes such as deterministic asynchronous, stochastic asynchronous, random asynchronous, etc.

Thus, it makes sense to consider models of dynamic biological systems using Boolean networks with asynchronous state updates. The paper describes general approaches and methods for generating such models, as well as the possibilities of practical hardware implementation of such models.

Related Works

The simplest simulation systems are synchronous simulators. These include those that perform simulations in this way and include simulators such as BooleanNet [19] and BoolNet [20]. If modeled using asynchronous system state updates, then the system analysis time will be very significant precisely because of the difficulty of adequately implementing such parallelism. However, these systems are used for GNR analysis and there are a number of problems that can be solved with their help. This is a feature of the digital approach to modeling. The disadvantage of synchronous modeling is the complete inconsistency of the complex biological system being modeled. Processes in a biological system are performed simultaneously, which is difficult to implement digitally.

An interesting modeling technique is the use of decision diagrams. For example, the method of binary decision diagrams (BDD). In this method, for the system model to represent the model, its decomposition is performed and rigid links between its components are established [28]. The simulation systems geneFAtt [26] and boolSim/genYsis [27] are another variant of the decision diagram, which is called ROBDD — reduced ordered binary decision diagrams. This method is convenient for representing complex logical functions and uses a directed acyclic graph to represent the model. Binary decision methods are symbolic and do not go through the entire state space. How often does the network analysis modeling process with TEMporal-LOGic specifications (Antelope) use model validation tools, a set of methods for automatically checking the properties of discrete systems, as well as for analyzing and constructing Boolean GRNs [29]. Model validators can prove properties of an infinite number of paths. Also, they can handle new, unexpected properties. It is noted that the main disadvantage of such methods is the impossibility of estimating the amount of memory required to complete the simulation, therefore, such methods require significant computing power. When using symbolic methods, attractors become available only at the very end of the simulation, while the computing power, in particular memory, may not be enough and the simulation process will end without results.

The use of FPGAs for building models seems promising. It has already been mentioned above that one of the main requirements for modeling tools is the possibility of parallel processing of the state space. This is a consequence of using the asynchronous state-space change model. There are a fairly large number of works in which the authors use FPGAs and achieve significant results in terms of the adequacy of the computational model, as well as acceptable simulation time. In [30], [31], [32], the calculation of scale-free GRNs was proposed; the search for attractors was accelerated by the use of FPGAs. Also, in some works [33], [34], variants of the Gillespie stochastic simulation algorithm on FPGAs are implemented. The above sources demonstrate the ability to use FPGA technology to simulate variants of the Gillespie algorithm, achieving performance up to 20 times faster than a competing general-purpose CPU.

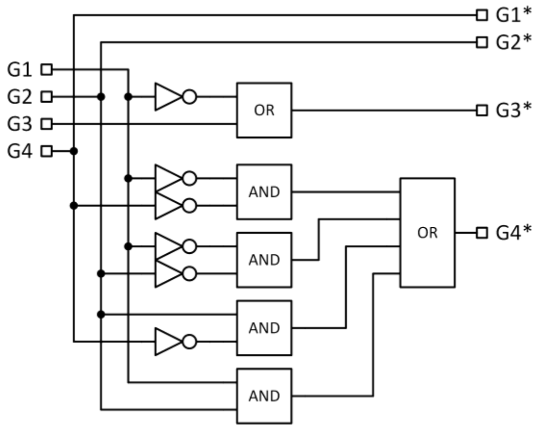
Methods of hardware implementation

With an increase in the number of genes in the studied gene regulatory networks, the performance of the selected software and hardware tools becomes an important factor. When the dimension reaches dozens of genes, the search for attractors by successive enumeration of all possible transitions between different combinations of gene states can take weeks and even months of continuous operation of software tools, even if very productive computing resources are used. In this case, it is obvious that there is a desire to speed up the calculation of chains of possible gene states as much as possible, as well as to parallelize the calculation processes for different states. A freely configurable hardware architecture such as an FPGA is great for this. The FPGA structure just allows you to implement any number of arbitrary rather complex logical dependencies, has sufficient resources to store state values, and can be configured in such a way as to implement a large number of parallel computing cores. The number of simultaneously working computing cores can be limited only by the complexity of the simulated Boolean network and the available resources of the selected FPGA chip [37].

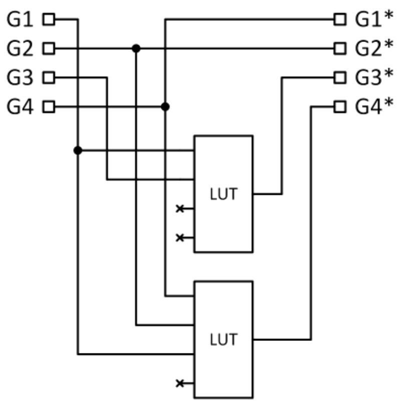
Consider a simplified example of a Boolean gene regulation network that includes four genes $\{G1, G2, G3, G4\}$, to clearly demonstrate the basic principles proposed in this article. The Boolean network is defined by the following system of logical equations, which allow, based on the current state of the genes, to calculate their next state:

$$\begin{aligned} &G1^* = G4 \\ &G2^* = G2 \\ &G3^* = (!\,G1 \mid G3) \\ &G4^* = ((!\,G1\ \&\ !\,G4) \mid (!\,G2\ \&\ !\,G1) \mid \\ &\qquad (G2\ \&\ !\,G4) \mid (G2\ \&\ G4)) \end{aligned} \tag{1}$$

To the left of the equal sign, new values of the gene states are formed, they are indicated with an asterisk symbol. At the next iteration of calculations, these values will be used in the expressions to the right of the equal sign. Logical operations are performed on the current values of the gene states: by the symbol ! the logical inversion is denoted, by symbols & and | the logical operations AND and OR, respectively, are indicated. The implementation of the presented system of logical equations is shown in Fig. 1.



a



b

Fig. 1. Implementation of Boolean network using logic elements (a) or FPGA Look-Up Tables (b)

In general, the scheme for calculating the states of the Boolean network genes must be sequential, since the obtained set of values of the gene states must be iteratively used to obtain the next set, then the next, and so on. To do this, a multi-bit memory element (register) is introduced into the circuit, which remembers the intermediate state of all genes and allows you to pass through the entire chain of successive states and, ultimately, detect the looping of the chain and trap states — attractors [38]. A circuit that can be easily implemented in FPGA hardware is shown in Fig. 2.

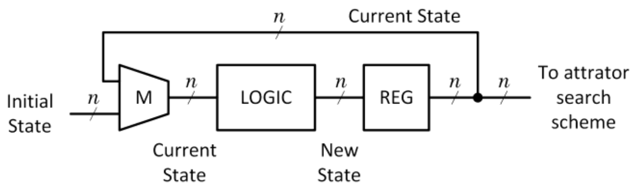


Fig. 2. Proposed structure of the FPGA-based Boolean network sequencer

In fact, any Boolean network represents a finite state machine. Each state vector represents the current state values of all genes according to Boolean functions in a network, while the transition condition may be represented as a block of logic to compute the next state.

Such a block of logic may be implemented as a raw fixed logic (non-flexible way) or as a configurable block, that can compute the state for any set of logic functions. The second option will be slightly inferior in performance and consume more resources than the first but gives an opportunity to quickly prepare for modeling any Boolean network topology without the need to perform laborious and time-consuming steps of technological synthesis and preparation of a new configuration for the FPGA chip. Thus, it is more useful in the case of usage of software tools for research or network structure experiments.

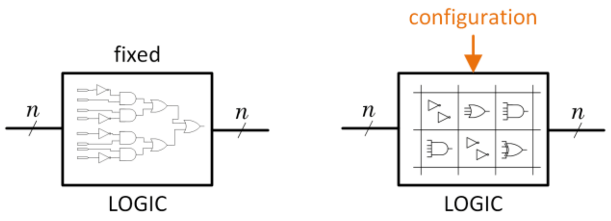


Fig. 3. Configurable logic element

One of the options to implement the next state selection logic is to use memory. In such cases, the input of the block is the binary value for the current state. The output is the value for the next state. Thus, to implement a transition table for a one 16-gene Boolean network function requires 64Kb of memory. While 32-gene function requires significantly more memory – over 4GB. 64KB of memory may be allocated even in FPGA memory blocks, but a bigger gene count requires a bunch of dedicated DRAM chips to follow such schema.

For implementation on the FPGA, the resulting multi-output combinational circuit is described using the selected HDL language, after which the operations of synthesis, Place and Route, and generation of a bitstream configuration file are performed. At the same time, such an approach is not as flexible as desired and requires regeneration of HDL design for each individual network topology if errors were found and corrected, and if the original model was refined or it is necessary to perform simulation run with a change in the behavior of a particular gene. This includes HDL code corrections, complete design synthesis, and fitting for certain FPGA chips before topology simulation and/or attractors search.

Obviously, software tools to perform automatic code generation, synthesis and other required steps may be implemented. In any case that doesn't eliminate the time-consuming operations of FPGA design compilation and load of generated binary to the device. Such an approach can take significantly more time than the simulation itself.

Mostly, simulation of gene reduction problems is limited to some predefined set of parameters, like maximum attractor length and number of genes in a network. For such a parameter subset, it is possible to create a single user-configurable component that can be configured to simulate an arbitrary Boolean network with characteristics that do not exceed some constraints. Such a component can be configured using a special code that is created using application software, without the need to make changes to the HDL code of the project and perform all stages of synthesis, technological preparation of the project and reconfiguring FPGA resources.

At the same time, state calculation logic might be implemented with a configurable block which includes some amount of dedicated logic elements with configurable connections between them. Such a structure can be seen in Fig. 3. This approach gives significantly lower memory consumption but uses logic elements of the FPGA.

Principles of reconfiguration

The primary objective of the proposed hardware accelerator is to search for closed-loop structures in the chain of consequent gene states. The researcher might need to change several transition functions in a Boolean network repeatedly to conclude how such a structure works and what properties it has. Taking this into consideration, we need to provide a reliable way to use the proposed accelerator in such conditions.

The proposed hardware structure is based on the principle that any Boolean function can be implemented as a Disjunctive Normal Form - a logical sum of logical products (the so-called SOP form of representing a logical expression) that includes input variables or their inverses. Another principle is the use of multi-output implicants. This principle can be used only when the implemented logical equations depend on the same set of input variables, which is the case in our case when constructing logical dependencies for Boolean networks. The multi-output implicants of a Boolean function are a reduced set of logical products of the input variables, which are sufficient to implement each logical expression.

The block diagram of the proposed solution for the implementation of the configurable logic element is shown in Fig. 4.

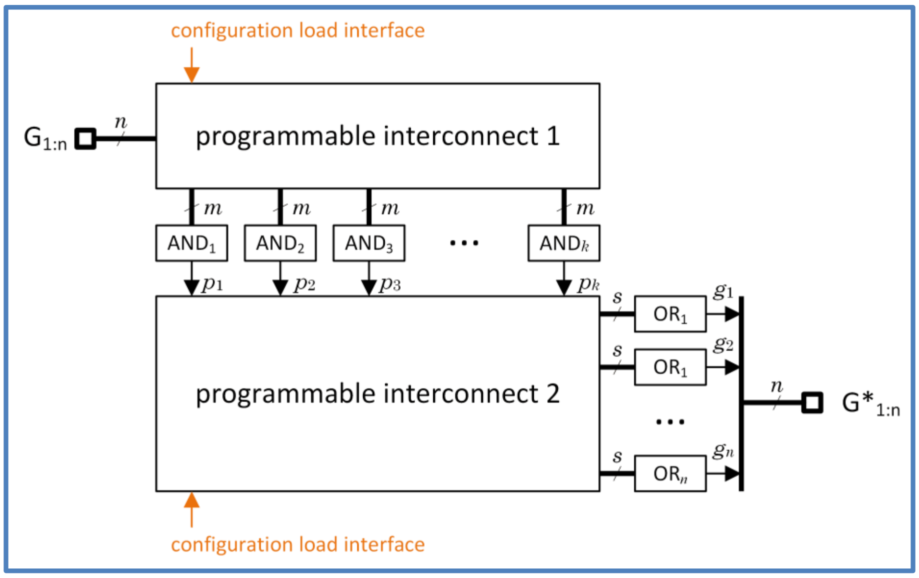
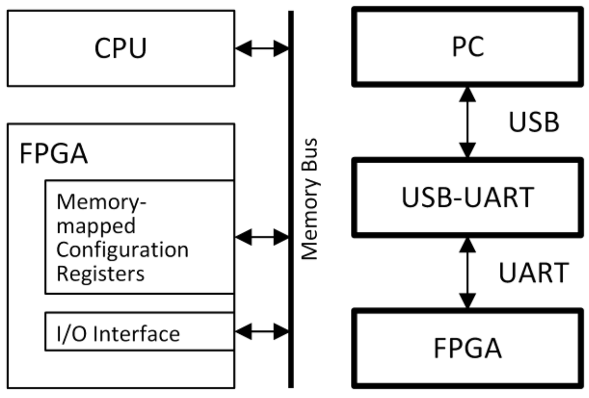


Fig. 4. Configurable logic of proposed hardware accelerator

The circuit contains a matrix of programmable interconnect 1, which ensures the formation of the necessary implicant products p_i using a set of m-input AND logic elements, as well as a matrix of programmable interconnect 2, in which the supply of the required products to the s-input OR logic elements is controlled, which provide implementation of logical dependencies of the required Boolean network and the formation of new values of gene variables.

Both matrices include special selector elements, the operation of which is determined by a special code that is fed to their configuration interface. The interfaces of all configurable elements are combined in such a way as to simultaneously configure their operation when a serial bit stream is fed to the common configuration interface of the logical dependency implementation module.

The configuration sequence is formed on the researcher's personal computer using special application software. Then, the way this sequence is loaded may differ depending on the chosen hardware platform and the way the system interacts with the user. Examples of system organization at the top level of its presentation are shown in Fig. 5.



a b

Fig. 5. Configuration interface with memory mapped device (a) and external interface converter (b)

It is widely adopted in the biological community to use SBML files to share gene models and write R language scripts to perform research. From a programmer perspective, each SBML file is an XML-formatted entity that contains a genes list and a collection of Boolean algebra equations – transition functions.

The proposed software should allow the processing of such models, load equations from the file and use them. At the same time, a user should have the possibility to express equations as a program and perform a research task with it (attractors search, topology optimization etc.).

The proposed hardware structure should receive a configuration, a sequence of bytes, that represent mux switches configurations and other interconnect option switches states. From the algorithmic perspective, it is convenient to use one of the normal Boolean forms (conjunctive or disjunctive) to generate configuration bytes. There is not much difference in which form to use. We use a Disjunctive Normal Form (DNF) for proposed structures and hardware.

Moreover, users might use non-optimal gene transition functions.

This leads to a waste of hardware resources and performance loss. It is unlikely that a lot of biologists have deep Boolean algebra knowledge. While optimization and conversion to a normal form are well studied in the literature [35] it is appearing a quite hard and annoying task for most researchers. Certainly, making a calculation each time after an equation change is not convenient and time-wasting. Thus, configuration software should verify the equation, optimize it if possible and convert it to DNF.

As a result, one can conclude general steps to perform research of the Boolean network structure presented on Fig. 6.

At the moment, we adopted Python usage. It is more convenient to implement SBML parser and accelerator configuration builder. In addition, there is a SymPy [36] package that allows to work with Boolean symbol arithmetic, minimize Boolean logic functions and convert them to DNF.

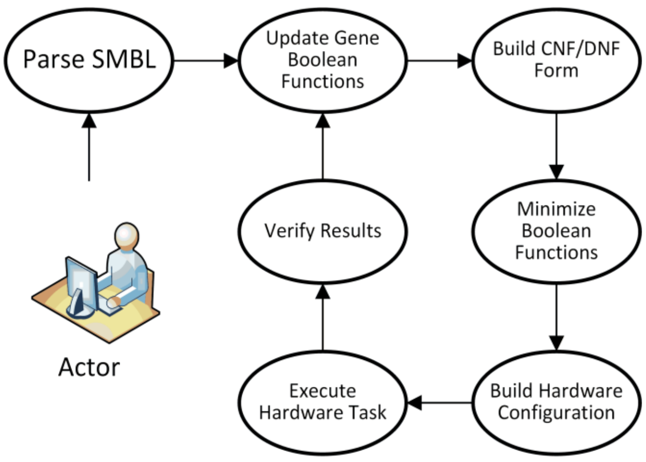

Parse SMBL
Update Gene Boolean Functions
Build CNF/DNF Form
Actor
Verify Results
Minimize Boolean Functions
Execute Hardware Task
Build Hardware Configuration

Fig. 6. Boolean function network research process

The hardware structure of the proposed accelerator requires configuration data built in an appropriate format.

For a basic network we should perform the following steps to build a configuration:

1. Load SBML file to obtain involved genes list and transition functions.
2. Translate gene names to indexed form.
3. Convert transition functions to a product of sums form.
4. Minimize Boolean functions.
5. Split transition functions to sums and products.
6. Build binary configuration data and store it to the FPGA device.

Ultimately, the configuration must be a multi-bit binary vector in which the configuration sequences for each interconnect matrix for all logical blocks of the hardware accelerator are concatenated. To ensure the delivery of the configuration to the device via the selected interface (this can be, for example, the UART interface that is widely used and used in digital technology, or rather, its implementation over the USB computer interface, special high-speed interfaces can also be used, such as PCI, or network interface, such as Ethernet) you need to add special headers to the common packet indicating the target nodes for which the transferred configuration is intended, as well as a checksum field to check the integrity of the configuration before applying it to the target logical block.

An example of the configuration package structure is shown in Fig. 7.

LD CONF TYPE 1	LEN N	BLK CONF 1	BLK CONF 2	...	BLK CONF N	CRC

LD CONF TYPE 2	LEN N	BLK CONF 1	BLK CONF 2	...	BLK CONF N	CRC

Fig. 7. Configuration package structure

Results

In this paper, we propose a flexible method of implementation of a hardware accelerator device, capable to speedup boolean gene regulatory networks simulation and analysis.

Results of FPGA resources consumption are presented in Table 1.

Table 1 – **FPGA Resources Consumption**

Implementation	**FPGA Resources**		
	Registers	***Memory blocks***	***LUTs***
Raw logic	32	0	25120
Complete solution	512	65K×16	37680

For the raw logic approach, time to get execution results consists of FPGA design synthesis, binary load time and execution time. In contrast, our approach time consists only of binary load time, configuration time and execution time. By configuration time we assume a lightweight “compilation” of mux enable and neg logic enable switches states.

Results of FPGA performance comparison are presented in Table 2.

Table 2 – **Performance estimation**

Implementation	**Execution performance**	**Startup time**
Raw logic	1clk per state	~ 7 min
Complete solution	1clk per mux state	< 10 sec

Our results show that proposed tools enable simulation of Boolean network models with increased performance and flexibility. While other solutions focus on execution performance, we also consider startup/compilation/synthesis time. Thus, overall performance has increased.

As a future research direction, we see a study of different Boolean gene reduction models to find the most relevant next state selection structure in terms of performance/flexibility.

Conclusions

The article considers the possibility of using FPGA to study the processes of gene co-expression, presented as a model based on a Boolean network. It is difficult to overestimate the importance of this solution: the study of co-expression for various cases in the world of biology and medicine will allow researchers to better understand the course of biological processes in living cells, which, in turn, can help in the search for treatments for serious diseases, various types of cancer, etc. From one side proposed approach is simple and suitable for building simulation structures but informative enough and able to get relevant results.

Approaches used for Boolean network simulation are analyzed. One of the main problems for most approaches is relatively long time for gene Boolean network re-simulation if some changes are needed after previous simulation.

Most existing simulation software systems require processes that include several steps such as updating the model, transforming and simplification of Boolean functions, applying changes to selected simulation software systems etc. In this article an approach is proposed that could reduce total modelling time significantly. The idea is to implement a generic infrastructure that could be quickly reconfigured for particular gene network instance. The promising idea is using FPGA to implement computation unit. It could be implemented with external CPU based or computation-specific hardware, however, there are some published works that proposed the use of FPGA–based solutions for simulation. As it was said before Boolean network of any genes amount could be represented by set of CNF/DNF simplified Boolean functions. It could be mapped easily to FPGA structure because of hardware array of logic gates which FPGA consists of. We proposed FPGA-based generic structure that could be quickly reconfigured using general purpose I/O interface (UART as an example). Specifics of this hardware unit were analyzed. It provides general functionalities of Boolean network element such as: deterministic state set, state transitions according to implemented Boolean functions, connection to attractor search engine.

The configurable part of computing unit is able to implement any set of *n* CNF/DNF Boolean functions that describe gene Boolean network to be studied. Also, an auxiliary part that provides configuring properties was added. Proposed implementation uses Boolean functions converted to DNF. Computing unit includes set of AND/OR logic gates according to number of genes that can be simulated. Configuring subsystem builds internal interconnections for AND/OR gate inputs according to loaded configuration bitstream that will provide desired logic function set for simulation.

Also, possible approach for configuration structure is described that allows to configure FPGA device used for simulation. Configuration bitstream could be prepared using general purpose PC or another suitable computing device using high-level language like Python.

Configuration data is presented as binary sequence and could be uploaded to FPGA using standard interface like UART.

FPGA resources utilization and reduce of simulation time were estimated if proposed solution is used for gene Boolean network simulation.

References

1. Faure, A., Naldi, A., Chaouiya, C. and Thieffry, D. (2006), “Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle”, *Bioinformatics*, vol. 22, pp. 124–131, doi: http://doi.org/10.1093/bioinformatics/btl210.

2. Li, F., Long, T., Lu, Y., Ouyang, Q. and Tang, C. (2004), "The yeast cell-cycle network is robustly designed", *Proc. Nat. Acad. Sci*., vol. 101, no. 14, pp. 4781–4786, doi: http://doi.org/10.1073/pnas.0305937101.
3. Huang, S. (1999), "Gene expression profiling, genetic networks, and cellular states: An integrating concept for tumorigenesis and drug discovery", *J. Molecular Med*., vol. 77, no. 6, pp. 469–480, doi: http://doi.org/10.1007/s001099900023.
4. Garg, A., Di Cara, A., Xenarios, I., Mendoza, L., and De Micheli, G. (2008), "Synchronous versus asynchronous modeling of gene regulatory networks", *Bioinformatics*, vol. 24, no. 17, pp. 1917–1925, doi: http://doi.org/10.1093/bioinformatics/btn336.
5. Zhang, R., Shah, M. V., Yang, J., Nyland, S. B., Liu, X., Yun, J. K., Albert, R. and Loughran, T. P. (2008), "Network model of survival signaling in large granular lymphocyte leukemia", *Proc. Nat. Acad. Sci*., vol. 105, no. 42, 2008, pp. 16308–16313, doi: http://doi.org/10.1073/pnas.0806447105.
6. Hu, Y., Gu, Y., Wang, H., Huang, Y. and Zou, Y. M. (2015), "Integrated network model provides new insights into castration-resistant prostate cancer," *Sci. Rep*., vol. 5, no. April, pp. 1–12, Nov. 2015, doi: https://doi.org/10.1038/srep17280.
7. Fumia, H. F., and Martins, M. L. (2013), "Boolean network model for cancer pathways: Predicting carcinogenesis and targeted therapy outcomes", *PLoS One*, vol. 8, no. 7, Art. no. 11, doi: https://doi.org/10.1371/journal.pone.0069008.
8. Lu, J., Zeng, H., Liang, Z., Chen, L., Zhang, L., Zhang, H., Liu, H., Jiang, H., Shen, B., Huang, M., Geng, M., Spiegel, S. and Luo, C. (2015), "Network modelling reveals the mechanism underlying colitis-associated colon cancer and identifies novel combinatorial anti-cancer targets," *Sci. Rep*., vol. 5, 2015, Art. no. 14739, doi: https://doi.org/10.1038/srep14739.
9. Rodrıguez, A., Sosa, D., Torres, L., Molina, B., Frıas, S. and Mendoza, L. (2012), "A Boolean network model of the FA/BRCA pathway," *Bioinf.*, vol. 28, no. 6, 2012, pp. 858–866, doi: https://doi.org/10.1093/bioinformatics/bts036.
10. Glass, L. and Kauffman, S. A. (1973), "The Logical Analysis of Continuous, Non-linear Biochemical Control Networks", *J. Theoretical Biol*., vol. 39, 1973, pp. 103–129, doi: https://doi.org/10.1016/0022-5193(73)90208-7.
11. Melas, N., Chairakaki, A. D., Chatzopoulou, E. I., Messinis, D. E., Katopodi, T., Pliaka, V., Samara, S., Mitsos, A., Dailiana, Z., Kollia, P. and Alexopoulos, L. G. (2014), "Modeling of signaling pathways in chondrocytes based on phosphoproteomic and cytokine release data", *Osteoarthritis Cartilage*, vol. 22, no. 3, pp. 509–518, https://doi.org/10.1016/j.joca.2014.01.001.
12. Chen, H., Wang, G., Simha, R., Du, C. and Zeng, C. (2016), "Boolean models of biological processes explain cascade-like behavior", *Sci. Rep*., vol. 6, 2016, Art. no. 20067, doi: https://doi.org/10.1038/srep20067.
13. Grieco, L., Calzone, L., Bernard-Pierrot, I., Radvanyi, F., Kahn-Perles, B. and Thieffry, D. (2013), "Integrative modelling of the influence of MAPK network on cancer cell fate decision", *PLoS Com.put. Biol*., vol. 9, no. 10, Sep. 2013, pp. 1–15, doi: doi: https://doi.org/10.1371/journal.pcbi.1003286.
14. Cohen,D. P., Martignetti, L., Robine, S., Barillot, E., Zinovyev, A. and Calzone, L. (2015), "Mathematical modelling of molecular pathways enabling tumour cell invasion and migration," *PLoS Comput. Biol*., vol. 11, no. 11, Sep. 2015, Art. no. e1004571, doi: https://doi.org/10.1371/journal.pcbi.1004571.
15. Saez-Rodriguez, J., Simeoni, L., Lindquist, J. A., Hemenway, R., Bommhardt, U., Arndt, B., Haus, U. U., Weismantel, R., Gilles, E. D., Klamt, S. and Schraven, B. (2007), "A logical model provides insights into T cell receptor signaling", *PLoS Computational Biology*, vol. 3, no. 8, Sep. 2007, pp. 1580–1590, doi: https://doi.org/10.1371/journal.pcbi.0030163.
16. Dorier, J., Crespo, I., Niknejad, A., Liechti, R., Ebeling, M. and Xenarios, I. (2016), "Boolean regulatory network reconstruction using literature based knowledge with a genetic algorithm optimization method", *BMC Bioinf*., vol. 17, no. 1, Art. no. 410, doi: https://doi.org/10.1186/s12859-016-1287-z.
17. Thomas, R. (1991), "Regulatory networks seen as asynchronous automata: A logical description", *J. Theoretical Biol*., vol. 153, no. 1, 1991, pp. 1–23, doi: https://doi.org/10.1016/S0022-5193(05)80350-9.
18. Purandare, M., Polig, R., and Hagleitner, C. (2017), "Accelerated analysis of Boolean gene regulatory networks," *Proc. 27th Int. Conf. Field Programmable Logic Appl*., 2017, pp. 1–6, doi: https://doi.org/10.23919/FPL.2017.8056778.
19. Albert, I., Thakar, J., Li, S., Zhang, R. and Albert, R. (2008), "Boolean network simulations for life scientists", *Source Code Biol. Med*., vol. 3, no. 1, 2008, Art. no. 16, doi: https://doi.org/10.1186/1751-0473-3-16.
20. Mussel, C, Hopfensitz, M., and Kestler, H. A. (2010), "BoolNet - an R package for generation, reconstruction and analysis of Boolean networks", *Bioinf.*, vol. 26, no. 10, pp. 1378–1380, doi: http://dx.doi.org/10.1093/bioinformatics/btq124.
21. Stoll, G., Caron, B., Viara, E., Dugourd, A., Zinovyev, A., Naldi, A., Kroemer, G., Barillot, E., and Calzone, L. (2017), "MaBoSS 2.0: An environment for stochastic Boolean modeling", *Bioinf.*, vol. 33, no. 14, Jul. 2017, pp. 2226–2228, doi: https://doi.org/10.1093/bioinformatics/btx123.
22. Saadatpour, A., Albert, R., and Reluga, T. C. (2013), "A reduction method for Boolean network models proven to conserve attractors", *SIAM J. Appl. Dynamical Syst*., vol. 12, no. 4, pp. 1997–2011, doi: http://dx.doi.org/10.1137/13090537X.
23. Ay, F., Xu, F. and Kahveci, T. (2009), "Scalable steady state analysis of boolean biological regulatory networks", *PLoS One*, vol. 4, no. 12, Dec. 2009, pp. 1–9, doi: http://dx.doi.org/10.1371/journal.pone.0007992.
24. Berntenis, N. and Ebeling, M. (2013), "Detection of attractors of large Boolean networks via exhaustive enumeration of appropriate subspaces of the state space", *BMC Bioinf*., vol. 14, no. 1, doi: https://doi.org/10.1186/1471-2105-14-361.
25. Mendes, D., Henriques, R., Remy, E., Carneiro, J., Monteiro, P. T. and Chaouiya, C. (2018), "Estimating attractor reachability in asynchronous logical models", *Frontiers Physiology*, vol. 9, 2018, Art. no. 1161, doi: https://doi.org/10.3389/fphys.2018.01161.
26. Zheng, D., Yang, G., Li, X., Wang, Z., Liu, F. and He, L. (2013), "An efficient algorithm for computing attractors of synchronous and asynchronous boolean networks", *PLoS One*, vol. 8, no. 4, Apr. 2013, pp. 1–7, doi: http://dx.doi.org/10.1371/journal.pone.0060593.
27. Garg, A., Di Cara, A., Xenarios, I., Mendoza, L. and De Micheli, G. (2008), "Synchronous versus asynchronous modeling of gene regulatory networks," *Bioinf*., vol. 24, no. 17, pp. 1917–1925, doi: http://dx.doi.org/10.1093/bioinformatics/btn336.
28. Mizera, A., Pang, J., Qu, H. and Yuan, Q. (2019), "Taming asynchrony for attractor detection in large boolean networks", IEEE/ACM Trans. Comput. Biol. Bioinf., vol. 16, no. 1, pp. 31–42, doi: http://dx.doi.org/10.1109/TCBB.2018.2850901.
29. Arellano, G., Argil, J., Azpeitia, E., Benıtez, M., Carrillo, M., Gongora, P., Rosenblueth, D. Aand Alvarez-Buylla, E. R. (2011), "“Antelope”: A hybrid-logic model checker for branching-time Boolean GRN analysis," *BMC Bioinf.*, vol. 12, no. 1, Art. no. 490, doi: https://doi.org/10.1186/1471-2105-12-490.
30. Zerarka, M., David, J. and Aboulhamid, E. M. (2004), "High speed emulation of gene regulatory networks using FPGAs", *Proc. 47th Midwest Symp. Circuits Syst*., Aug. 2004, pp. I–545, doi: http://dx.doi.org/10.1109/MWSCAS.2004.1354048.

31. Pournara, I., Bouganis, C. and Constantinides, G. A. (2005), “FPGA-accelerated Bayesian learning for reconstruction of gene regulatory networks”, *Proc. Int. Conf. Field Programmable Logic Appl.*, Sep. 2005, pp. 323–328, doi: http://dx.doi.org/10.1109/FPL.2005.1515742.
32. Ferreira, R. and Vendramini, J. C. G. (2010), “FPGA-accelerated attractor computation of scale free gene regulatory networks”, *Proc. Int. Conf. Field Program. Logic Appl*., pp. 550–555, DOI: http://dx.doi.org/10.1109/FPL.2010.108.
33. Salwinski, L. and Eisenberg, D. (2004), “In silico simulation of biological network dynamics”, *Nature Biotechnology*, vol. 22, no. 8, Aug. 2004, pp. 1017–1019, doi: http://dx.doi.org/10.1038/nbt991.
34. Keane, F., Bradley, C. and Ebeling, C. (2004), “A compiled accelerator for biological cell signaling simulations”, *FPGA '04: Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays*, Art. no. 233, pp. 233–241, doi: https://doi.org/10.1145/968280.968313.
35. Whitesitt, J.E. (2012), *Boolean Algebra and Its Applications*, Courier Corporation, 192 p., available at: https://www.scribd.com/book/365215162/Boolean-Algebra-and-Its-Applications.
36. Meurer, A., Smith, C. P., Paprocki, M. and Čertík, O. (2017), “SymPy: symbolic computing in Python”, *PeerJ Comput. Sci.*, 3:e103, doi: https://doi.org/10.7717/peerj-cs.103.
37. Vasylchenkov, O. G., Salnikov, D. V. and Karaman, D. G. (2022), “Hardware model for boolean network attractors search,” *Proc. of 22 Intl. scient. and pract. conf. Problems of informatics and modeling* (PIM–2022), Kharkiv – Odesa, 2022, p. 20, available at: https://repository.kpi.kharkov.ua/handle/KhPI-Press/59901.
38. Vasylchenkov, O. G., Salnikov, D. V. and Karaman, D. G. (2022), “Hardware computational infrastructure for boolean network attractors search”, *Proc. of Intl. scient. and tech. conf. Automation, electronics, information and measurement technologies: education, science, practice*, Kharkiv, Dec. 01-02, 2022, pp. 13–14, available at: http://repository.kpi.kharkov.ua/handle/KhPI-Press/60498.

Received (Надійшла) 31.05.2023
Accepted for publication (Прийнята до друку) 16.08.2023

Відомості про авторів / About the Authors

Васильченков Олег Георгійович – кандидат технічних наук, доцент кафедри автоматики та управління в технічних системах, Національний технічний університет «Харківський політехнічний інститут», Харків, Україна;
Oleg Vasylchenkov – Candidate of Technical Sciences, Associate Professor of the Department of automation and control in technical systems, National Technical University «Kharkiv Polytechnic Institute», Kharkiv, Ukraine.
e-mail: oleh.vasylchenkov@khpi.edu.ua; ORCID ID: http://orcid.org/0000-0002-0969-2248.

Сальніков Дмитро Валентинович – кандидат технічних наук, асистент кафедри автоматики та управління в технічних системах, Національний технічний університет «Харківський політехнічний інститут», Харків, Україна;
Dmytro Salnikov – Candidate of Technical Sciences, Assistant Professor of the Department of automation and control in technical systems, National Technical University «Kharkiv Polytechnic Institute», Kharkiv, Ukraine.
e-mail: dmytro.salnikov@khpi.edu.ua; ORCID ID: http://orcid.org/0009-0007-6201-5370.

Караман Дмитро Григорович – старший викладач кафедри автоматики та управління в технічних системах, Національний технічний університет «Харківський політехнічний інститут», Харків, Україна;
Dmytro Karaman –Senior Lecturer of the Department of automation and control in technical systems, National Technical University «Kharkiv Polytechnic Institute», Kharkiv, Ukraine.
e-mail: dmytro.karaman@khpi.edu.ua; ORCID ID: http://orcid.org/0000-0002-7252-3172.

Прискорення аналізу булевих мереж регуляції генів за допомогою FPGA

О. Г. Васильченков, Д. В. Сальніков, Д. Г. Караман

Анотація. Експресія генів не відбувається довільно та спонтанно, вона підпорядковується певним закономірностям, які можна виразити у вигляді зв’язаного графу чи мережі. Розкриття цих закономірностей вимагає великого обсягу експериментальних досліджень і накопичення необхідної статистичної інформації. Потім ця інформація піддається математичній обробці, яка залучає значні обчислювальні ресурси та займає багато часу. Булеві мережі часто використовуються як основа для побудови математичних моделей у цих розрахунках. Останнім часом моделі, засновані на булевих мережах, дедалі більше зростають у розмірі та складності, викликаючи підвищені вимоги до традиційних програмних рішень і обчислювальних інструментів. Програмовані вентильні матриці (FPGA) — це потужна платформа з можливістю реконфігурації для забезпечення ефективних і високопродуктивних обчислень. Використання FPGA може значно прискорити процес обчислення послідовного ланцюга станів генів, як за рахунок використання апаратного прискорення при обчисленні логічних залежностей, так і за рахунок реалізації масиву паралельних обчислювальних ядер, кожне з яких може виконувати свою власне індивідуальне завдання. Іншим рішенням, яке може істотно спростити роботу дослідників мереж регуляції генів, є створення універсальної обчислювальної архітектури, яка дозволяє динамічно реконфігурувати свою внутрішню структуру при зміні завдання або логічних залежностей для поточної булевої мережі. Таке рішення позбавить дослідника від необхідності виконувати весь комплекс дій з технологічної підготовки нової конфігурації ПЛІС, від внесення змін до коду HDL, що описує мережу, до завантаження оновленої конфігурації в апаратний прискорювач. У статті обговорюється, як використовувати FPGA для реалізації та моделювання довільних булевих мереж, описується концепція універсальної архітектури ядра, що реконфігурується, для обчислення логічних залежностей довільної булевої мережі та пропонується практична реалізація такого обчислювального ядра для моделювання генної регуляції мережі.

Ключові слова: мережа регуляції генів; моделювання біологічних систем; програмовані логічні інтегральні схеми; архітектури апаратного прискорення; булева модель мережі; системи обчислювальної біології; біоінформатика.