Общая информация
Оргкомитет
Регистрация
Требования к докладам
Программа конференции
Место проведения
Контакты
История конференции

ПРОГРАММА МЕЖДУНАРОДНОЙ НАУЧНО-ТЕХНИЧЕСКОЙ КОНФЕР
"Автоматизация' 2021"

.

№	Фамилии авторов, название доклада
	Секция 1. Автоматизация технологических процессов
8926	*Yu. S. Petrov, Yu. V. Sakhanskiy, S. P. Maskov* Electrical Firing System Optimization at High Frequency Ignition for Industrial Explosives
10005	*A. G. Soshinov, O. S. Atrashenko, T. V. Kopeikina* The Feasibility Assessment for the Introduction of 20 kV Voltage Class to Power Distribution N
10007	*A. F. Burkov, V. V. Mikhanoshin, V. K. Nguyen* The Impact of Nonlinear Converters on the Quality of Electrical Energy
10009	*Yakov M. Kashin, Lev E. Kopelevich, Alexandr V. Samorodov* Generator Set for Hybrid Power Systems
10010	*Alexander V. Starikov Viktor I. Domanov Abbas A. Karim Altakher* Mathematical Model of Electrical Complex Single Drum Belt Conveyor with Asynchronous Ex
10020	*E. Merzlikina, Hoang Van Va, G. Farafonov* Automatic Control System with an Autotuning Module and a Predictive PID-Algorithm Processes
10023	*K. Smirnov, A. Nazarov, A. Ulyahin* Automation of Interface Boards Design Based on VLSI Functional Control Tests
10031	*Oleg S. Malakhov, Dmitriy Y. Usatiy, Natalja V Dyorina* The Engine Control Unit Improvement for Air-Mixture Control and Engine Power Developmer
10043	*S. S. Shovkoplyas, L. A. Zhlobitsky* Innovative Control of the High-Voltage Thyristor Arm of the Ice Melting Plant with Rectified Wires of Overhead Transmission Lines
10048	*V.R. Gasiyarov, B.M. Loginov, M.A. Zinchenko, A.Yu. Semitko* Improving the Load Balancing System of the Rolling Mill Stand Drives
10161	*Svetlana V. Kuznetsova, Alexander L. Simakov* Part Position Correction During Assembly According to Force and Torque Sensor Signals
10164	*Isroil I. Jumanov, Olimjon I. Djumanov, Rustam A. Safarov* Methodology of Optimization of Identification of the Contour and Brightness-Color Picture Micro-Objects
10172	*Dmitrii V. Efanov, German V. Osadchy, Valerii V. Khóroshev* Digital Train Traffic Control System with Perfect Level of Diagnostics and Monitoring: Virt Interlocking
10183	*Pavel V. Ilyushin, Sergey P. Filippov* Static Devices to Prevent Unnecessary Disconnections of Gas-Piston Units in Transients
10185	*Pavel V. Ilyushin, Aleksandr L. Kulikov* Towards More Efficient Emergency Power Systems in Low- and Medium-Voltage Distribution
10189	*Yu.V. Bebikhov, Yu.A. Podkamenniy, A.S. Semenov* Development of an automatic process control system for biological wastewater treatment
10206	*Andrey V. Smolyaninov, Irina V. Pocebneva, Antonina R. Deniskina* Fuzzy Control of the Polymerizer Start Process in the Production of Bottle-Making
10220	*Viktor.P. Lapshin, Ilia. A. Turkin, V. U. Omelechko* Automation the Processes of Wood Processing by Drilling, Due to the Development a Apparatus for Accounting the Interrelated Feed Drive and Drive of Cutting
10226	*Vladimir Trofimov, Yana Neudakhina* About Designing an Intelligent System for Slag Detection in Oxygen Converter Steelmaking
10309	*M. Sidorova, L. Gorbushin, N. Koneva* Analytical review of electronic devices of modern supercomputing systems

Development of Automation System for HDL Modeling of the Communication Subsystem for Networks-on-Chip

E. V. Lezhnev
HSE University
Moscow, Russian Federation
elezhnev@hse.ru

2021 International Russian Automation Conference (RusAutoCon) | 978-1-6654-1523-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/RusAutoCon52004.2021.9537542

***Abstract*—With the active development and application of multi-core systems-on-chip, the design of efficient communication systems- on-chip is becoming a particularly challenging task. Designing a communication subsystem of networks-on-chip (NoCs) is a time-consuming process whose task is to select the optimal characteristics in a given range of values. Low-level modeling is an integral design step that allows obtaining accurate network characteristics, although it is time-consuming compared to high-level modeling. Most low-level NoC models include all the system-on--chip (SoC) components, thus slowing down modeling the communication subsystem (because to check one parameter, it is required to simulate the entire system). The proposed low-level model of a communication subsystem allows for automated generation of HDL model, as well as for modeling of both topology and routing algorithms for NoCs. The experiments carried out as exemplified by the study of circulant topologies showed an increase in the modeling rate, as well as the correctness and usefulness of such a model for various applications.**

Keywords—NoC, modeling automation, low-level modeling, NoC communication subsystem

I. INTRODUCTION

Currently, one of the most important areas of research in the field of informatics and computing systems is the construction of multi-core processors. The transition to multi-core processors allows us to overcome the performance degradation in the design of more and more complex single-core systems [1]. In the context of growing interest in technologies for constructing SoCs / multiprocessor SoCs for computing and neural networks, with the number of cores reaching hundreds of thousands [2, 3], there is the matter of the efficiency of communication between the cores. In a multi-core processor with a small number of cores, communication between IP cores and other components occurs using a common bus [4, 5] which is not able to provide communication between a large number of cores since its operating speed decreases, and the bus no longer meets the throughput requirements of multiprocessor SoCs [1]; therefore, NoCs are becoming widespread.

NoC design is a multi-step task that aims to define many different network characteristics: selection of main NoC components, the topology of router connections, routing algorithm, structure and features of router performance, method of control and arbitration of data flows in the network [6, 7]. To analyze the impact on NoC performance, to make certain decisions, modeling is required. The use of a network to connect computational nodes makes it possible to achieve efficient operation of a large number of nodes, as well as to maintain the acceptable speed of data transmission between them [8, 9].

Depending on what characteristics of the system are investigated, and at what stage, several types of modeling can be distinguished:

- high-level modeling. It is usually carried out in the early stages of design and is carried out using high-level languages [10, 11, 12, 13, 14]. At this stage components such as, communication subsystem topology, routing algorithm, type of computing nodes are primary selected;
- behavioral modeling. It is used to assess network traffic and analyze the throughput and delays of packet transmission, as well as study the effectiveness of routing algorithms in NoCs [15, 16]. Based on the simulation results, recommendations are made for changing the NoC components;
- low-level modeling. It is used to study the consumption of chip resources and power consumption of the designed system at the level of its prototype described in HDL [17, 18].

High-level and behavioral models are usually developed in high-level languages. Their main task is to check the operability of the network, its throughput, saturation level. High-level models are not synthesized into a real NoC; so, they are not enough to finally assess the possibility of using the selected network configuration on FPGA. They only put forward a hypothesis about which network parameters are better.

HDL models are more accurate because they are actually an NoC prototype. Investigation of HDL models gives information about the chip resources occupied by the network, show the real frequency, the network works with a short. However, modeling takes much longer since events in each element of the circuit are analyzed at each moment of time, and for this, event-driven modeling tools are used (for example, ModelSim [19]). This allows an accurate NoC model to be obtained, as exemplified by Netmaker [20, 21]. The main disadvantage of this approach is that modeling takes too long.

978-1-6654-1523-1/21/$31.00 ©2021 IEEE

II. Low-Level NoC Models

Low-level NoC models are complex systems that include computational nodes, a connection structure of these nodes (communication topology), routers that control the transfer of data between computational nodes. NoC design consists of the selection of such network parameters so that the required characteristics can be achieved. An example of such a model is Netmaker [20]. The model is developed in SystemVerilog and includes a description of a router with virtual links, network topology structure, routing algorithms, arbitration systems for access to virtual links, and other components. Normally, a description of the communication topology of computational nodes is deeply integrated into the model. The number of topologies supported is limited to the most commonly used ones, such as mesh and torus. Modification of such models (adding a new topology, communication with other NoC components) is a difficult task.

This model allows for a comprehensive NoC study. However, when the selection of network parameters occurs, in one iteration of the simulation, one investigated parameter is subject to change. Since the HDL models are highly connected, then when even one parameter in one of its components changes, it is necessary to simulate the entire network as a whole. Which leads to a strong increase in the time required to obtain results. Changes made at each stage of modeling affect only one component of the network, thus, there is no need to re-model the part of the network in which no changes occurred [22]. Separate modeling of system components will speed up the modeling process of both one component and the network as a whole by reducing the time required to verify network parameters.

III. HDL Model of NoC Communication Subsystem

Due to the existing shortcomings of the available low-level methy-on-a-chip models, which do not allow modeling networks with a large number of nodes, and also do not allow changing model components, an original HDL model which allows for the simulation of a separate NoC communication subsystem and provides the ability to separately simulate NoC conponents was developed. Its structure is shown in Fig. 1.

The model consists of 4 components:

- the core of the model (generated files in Verilog which implement communication subsystem of NoC under study with possibilities for its comprehensive parameterization);
- configuration module which produces a parametric generation of NoC communication subsystem allowing to configure the characteristics of the network (the number of nodes, the topology of communication between them, the choice of the routing algorithm), and also allows you to additionally view the topology of the network and get a list of all optimal routes between the nodes for further analysis of the results;
- data processing module which is required to obtain the results of the model and generates test data for submission to the model in real time;
- testing infrastructure which contains files for automatic testing of the model on pre-prepared data instead of prototyping.

The core of the model [23, 24] consists of two main files – Router.v, Topology.v and an auxiliary file Signal_generator.v. Router.v file implements the investigated routing algorithm and is actually a state machine, based on which a router operates in the network. Using the information about communication subsystem topology (stored in the router), as well as the service information (stored in transmitted data packets), the port to which the packet must be sent so that it reaches the destination node is selected

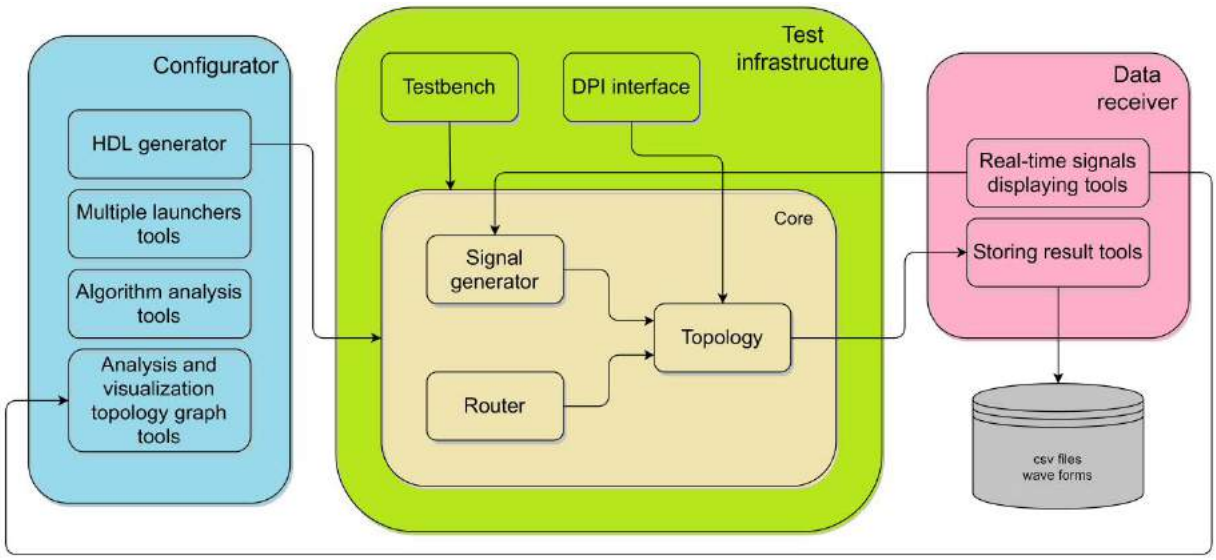

Fig. 1. Structure of HDL model of communication subsystem

For the working of the vast majority of routing algorithms, it is necessary to store information about the number of routers in the network, about the number of ports to which a packet can be sent, as well as information about the type of connection of nearest routers. Structurally, this file consists of three parts: a block that is responsible for receiving packet data, determining which part of the router it came from and extracting data from it are necessary for the routing algorithm to work; a block that implements the routing algorithm, which calculates to which port the packet must be transmitted in order for it to reach the destination node; a block that changes the original received packet, overwriting the service information prepared by the routing algorithm into it, and sends the changed packet to the required port.

Topology.v file implements the investigated communication subsystem topology. It connects all routers to each other in accordance with the investigated topology, transfers data between them and outputs information to the computing node. Additionally, auxiliary modules for interacting with the model are connected in this module.

Signal_generator.v file is auxiliary and, in fact, is not a structural element of the communication subsystem; it is needed to emulate the generation of data packets when testing network performance and is a replacement for computational nodes which generate and receive data in the network. All router ports in charge of communication with computing nodes are connected to Signal_generator.v module. To conduct high-level testing of the model in this module, all the logic necessary for the operation of the DPI interface is additionally implemented.

Signal_generator.v module generates test data packets that are fed to the network, and also receives packets from the network and analyzes for errors when the packet passes through the network. Since the main task is to test the very operation of network topology and routing algorithm, 1 bit and a certain number of bits of service information act as the payload data of the data packet which in the general case contains the numbers of the source and receiver nodes of the data packet.

The model is built in such a way that allows easy integrating third-party modules without deep reworking of the original code structure. For example, to add computational nodes which are to generate data transmitted over the network, it is necessary, in Router.v module, instead of 1 bit which indicates the presence of transmitted data, to add the packet length required for the computational node. Additionally, it is necessary, in the Topology.v module, instead of the Signal_generator.v module, to connect the required computational modules as sources of packages. In a similar way, it is possible to connect other network modules to implement complex modeling of the communication subsystem.

Since Signal_generator.v is an auxiliary file, its modeling together with the main network files introduces redundancy and adds distortions to the occupied amount of chip resources during prototyping. In this case, it is possible to calculate how much space on the chip this module takes and consider this data when analyzing the space occupied by the communication subsystem.

The assessment of redundancy impact introduced by Signal_generator.v module on prototyping communication subsystem is shown in Fig. 2 and Fig. 3.

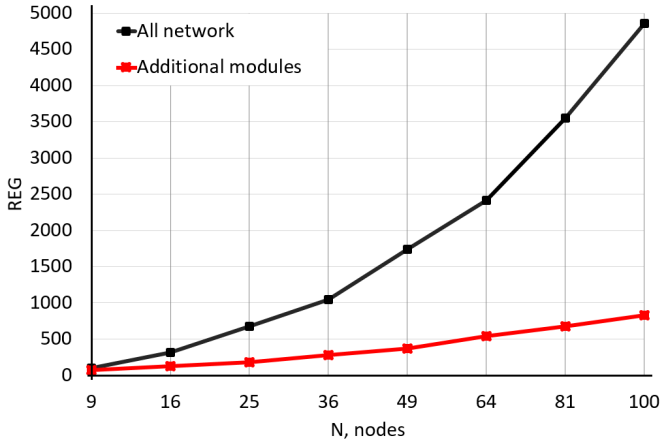

Fig. 2. Dependence of the use of FPGA registers for all network and auxiliary modules on the number of nodes in the network.

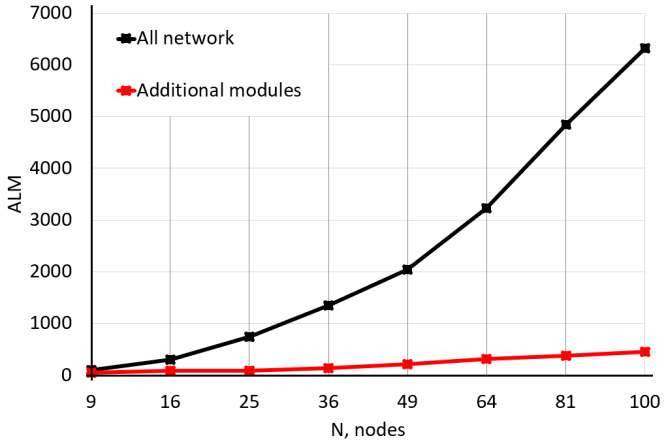

Fig. 3. Dependence of the use of logical FPGA resources for all network and auxiliary modules on the number of nodes in the network.

The figures show that the module takes much less resources than the communication subsystem. Numerical calculation of the resources occupied by Signal_generator.v module can be represented as

$$U_{alm} = 0{,}0034 * x^2 + 4{,}2420 * x + 2{,}7879 \quad (1)$$

$$U_{reg} = 0{,}0076 * x^2 + 7{,}7416 * x + 9{,}5715 \quad (2)$$

where x – number of nodes in the network.

If for networks with less than 10 nodes, auxiliary modules occupy a good part of the resources occupied by the network (more than 60% of registers and more than 40 % of logical blocks), then with an increase in the number of nodes to 100, auxiliary modules occupy 17 % of registers and 7 % logical blocks. For networks with more than 100 nodes, the number of occupied logical blocks can be considered to be a constant.

The number of crystal resources occupied by the auxiliary module practically does not depend on the selected routing algorithm or topology. It mainly depends on the number of nodes in the network. The main element that is contained in the module is the data buses that go to all routers and it is they who set the main load on the registers, as well as the algorithms for choosing the router to which the packet is initially fed make up the main load on the logical elements.

For the modeling communication subsystem, specialized test benches required for preliminary testing of the model were developed. Additionally, the ability to call files for testing (developed in a high-level language using DPI interface) is implemented.

IV. Modeling Automation of NoC Subsystem

For each analyzed configuration of the communication subsystem, it is necessary to create original model core files, and it is a routine and time-consuming task. To automate this process, the Configurator module in C# was developed. The task of this module is to generate model core files based on information about the topology and routing algorithm.

There are two options for file generation: by prototype and by translation from C#.

When generating files based on a prototype, the developer needs to make one-time coding of model core and testbenches. When subsequent iterations of modeling, the configuration module is specified which parameters in the files need to be changed in order to bring them into conformity with the configuration under study. It is also possible to specify a list of modeling configurations.

When generating HDL files by translation from C#, code tokens are generated. To simplify the analysis, certain requirements are imposed on the C# algorithm in terms of the structure and used language constructs. An element that is to be exposed as a Verilog module must be developed as a separate function in C#. Only a limited set of language keywords can be used in the implementation of the algorithm. Currently, the following data types are supported: integer, long, boolean. To reduce the use of memory registers, the maximum possible size of a one-dimensional array of registers is calculated for integer and long types. In most cases, the maximum possible length of an array of registers for variables is limited by the number of bits equal to $log_2 N$ (N – number of nodes in the network) required to store the number of nodes in the network. There is also a limitation on the use of operators. The following operators are currently supported: if, for, do, while, case. The given set of data types and operands allows the vast majority of routing algorithms to be implemented in C# and does not require complex code translation procedures [25].

The first method of generating files is suitable for any files, including those developed by third-party developers. However, it requires initial modeling and prototyping, which is time-consuming. The second method is faster, since there is no need to develop the initial development of files for the prototype model. But this method imposes a number of restrictions on the implementation of algorithms in a high-level language.

Configuration module can automatically generate all the files required by Verilog and start modeling and prototyping of the communication subsystem using TCL commands. Additionally, the configuration module implements the possibility of a high-level check of the correctness of the structure of the communication subsystem and routing algorithm by generating routing paths from zero router to all the others. Thus, it becomes possible to combine high-level models and low-level models in one software package to carry out end-to-end simulation of the NoC.

Signal_generator.v file allows submitting generated packets to check the operation of the communication subsystem in automatic mode. To obtain the results of network operation in the form of a file, a Data receiver module was developed. This module is also developed in C#. The data processing module allows connecting to Signal_generator.v module, receiving simulation results, and saving them to a file for further analysis. Also, the ability to supply data in manual mode using the UART interface employed in Signal_generator.v was implemented.

V. Investigation of the Developed NoC Model

To test the proposed model, we simulated circulant topology networks with the number of generators 2 and the number of nodes from 9 to 100 [26]. Modeling was done using Intel Core i5-9400 2.90 GHz – 16 GB RAM. An estimate of the resources occupied by the communication subsystem on FPGA (depending on the selected routing algorithm) was obtained. Two simulations were carried out: manual - when the developer for each network developed an HDL model and tested it; generation of HDL codes was carried out automatically using the developed model and its testing was also carried out. The results met all expectations and confirmed the correct operation of the model during simulation, and also when it was implemented on an FPGA chip.

We also compared the modeling rate of communication subsystem using Modelsim and using the developed model for Clockwise routing algorithm. The comparison result is shown in Fig. 4.

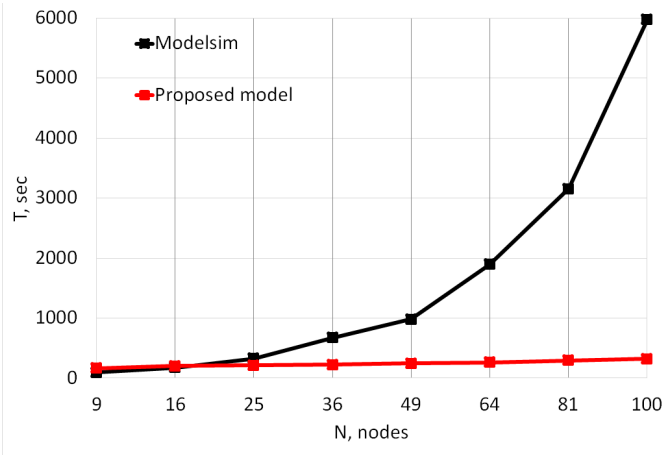

Fig. 4. Comparison of modeling rate of communication subsystem using Modelsim and the model proposed.

The figure shows that on topologies with a small number of nodes, the proposed modeling method requires more time to

conduct an experiment due to the need to perform more steps to start modeling, and to configure model generation, to create generation files as well. However, when the number of beats is more than 20, the speed of modeling using Modelsim exceeds the speed of modeling using the developed model. And with a large number of nodes, a significant gain in modeling rate is achieved. For the selected topology and routing algorithm, the increase in time for modeling using the proposed method grows linearly and depends on how many crystal resources will be used.

VI. CONCLUSION

The proposed low-level model of NoC communication subsystem allows automation of development of a single-type code for models of SoC communication subsystems with a change in certain parameters due to the automated generation of HDL code from the high-level code from C#, as well as modeling the system, collecting and primary processing of simulation results. The consequence of automation of development and modeling is an increase in modeling rate by 1500% due to a decrease in the time required to prepare the model. The modular structure of the proposed model makes it easy to adapt it to any task by adding the necessary components, without losing the possibility of separate modeling of NoC The carried out approbation of the model exemplified by the study of the amount of resources occupied by communication subsystem (depending on the selected routing algorithm) showed the operability of the model, significant reduction in time to obtain results and correctness of HDL files it creates.

REFEENCES

[1] G. Nychis, C. Fallin, T. Moscibroda, and O. Mutlu, "Next Generation On-Chip Networks: What Kind of Congestion Control Do We Need?" Hotnets-IX: Proc. of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, pp. 1–6, 2010.

[2] N. Vassilieva, P. Buitrago, N. Nystrom, and S. Sanielevici, "Technical overview of the Cerebras CS-1, the AI compute engine for Neocortex". [Online]. Available: https://www.cmu.edu/psc/aibd/neocortex/technical-overview-webinar.html.

[3] W. Cesario, D. Lyonnard, G. Nicolescu, Y. Paviot, S. Yoo, and A. Jerraya, "Multiprocessor SoC platforms: A component-based design approach," IEEE Design and Test of Computers, vol. 19, is. 6, pp. 52–63, 2002.

[4] M. Sayeed and M. Atiquzzaman, "Multiple-bus multiprocessor under unbalanced traffic," Computers & Electrical Engineering, vol. 25, is. 2, pp. 77–94, 1999.

[5] T. Nikolic, S. Djosic, G. Nikolic, and G. Djordjevic, "Token ring arbitration scheme for on-chip CDMA bus architectures," Microelectronics Journal, vol. 106, 2020.

[6] O. Romanov and O. Lysenko, "The Comparative Analysis of the Efficiency of Regular and Pseudo-optimal Topologies of Networks-on-Chip Based on Netmaker," Proc. of Mediterranean Conf. on Embedded Computing, pp. 13–16, 2012.

[7] A. Ramy, M. Hassan, and A. Khalil, "Design of a reconfigurable network-on-chip for next generation FPGAs using Dynamic Partial Reconfiguration," Microelectronics Journal, vol. 108, 2021.

[8] Y. Ouyang, F. Tang, C. Hu, W. Zhou, and Q. Wang, "MMNNN: A tree-based Multicast Mechanism for NoC-based deep Neural Network accelerators," Microprocessors and Microsystems, vol. 85, 2021.

[9] Z. Yu, F. Luo, B. LI, W. Zhou, L. Zong, and Q. Tao, "Reconfigurable mesh-based inter-chip optical interconnection network for distributed-memory multiprocessor system," Optik, vol. 121, is. 20, pp. 1845–1847, 2010.

[10] N. Jiang, G. Michelogiannakis, D. Becker, B. Towles, and W. Dally, BookSim 2.0 User's Guide, 2010.

[11] R Kourdy, S. Yazdanpanah, and M. Rad, "Using the NS-2 network simulator for evaluating multi-Protocol label switching in network-on-Chip," Second Int. Conf. on Computer Research and Development, pp. 795–799, 2010.

[12] P. Prilepko, A. Romanov, and E. Lezhnev, "Modification of a High-Level NoCModel 2.0 for Modeling Networks-on-Chip with Circulant Topologies," Problems of Perspective Micro- and Nanoelectronic Systems Developmen, is. 4, pp. 23–30, 2020.

[13] S. Lee and N. Bagherzadeh, "A high level power model for Network-on-Chip (NoC) router," Computers & Electrical Engineering, vol. 35, is. 6, pp. 837–845, 2019.

[14] R. Kunthara, R. James, S. Sleeba, and J. Jose, "Traffic aware routing in 3D NoC using interleaved asymmetric edge routers," Nano Communication Networks,vol. 27, 2021.

[15] D. Bertozzi, A. Jalabert, and L. Benini, "NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-Chip," IEEE Transactions on Parallel and Distributed Systems, vol. 16, no. 2, pp. 113–129, 2005.

[16] N. Genko, D. Atienza, and L. Benini, "Feature – Noc Emulation: A Tool and Design Flow for MPSoC," IEEE Circuits and Systems Magazine, vol. 7, no. 4, pp. 42–51, 2008.

[17] A. Romanov and A. Ivannikov, "SystemC Language Usage as the Alternative to the HDL and High-level Modeling for NoC Simulation," Int. Journal of Embedded and Real-Time Communication Systems, vol. 9, no. 2, pp. 18–31, 2018.

[18] X. Duan and Y. Li, "A Multiphase Routing Scheme in Irregular Mesh-Based NoCs," Fourth Int. Symposium on Parallel Architectures, Algorithms and Programming, pp. 277–280, 2011.

[19] A. Oukaira, O. Ettahri, M. Tabaa, S. Taheri, and A. Lakhssassi, "Simulation, Validation and FPGA Implementation of a Ring Oscillator Sensor for Thermal Management and Monitoring," Procedia Computer Science, vol. 155, pp. 83–88, 2019.

[20] R. Mullins, A. West, and S. Moore, "Low-latency virtual-channel routers for on-chip networks," Proc. 31st Annual Int. Symposium on Computer Architecture, pp. 188–197, 2004.

[21] K. Parane, P. Prabhu, and B. Talawar, "FPGA based NoC Simulation Acceleration Framework Supporting Adaptive Routing," IEEE Int. Conf. on Electronics, Computing and Communication Technologies, pp. 1–6, 2018.

[22] N. Venkataraman and R. Kumar, "Design and analysis of application specific network on chip for reliable custom topology," Computer Networks, vol. 158, pp. 69–76, 2019.

[23] E. Lezhnev and A. Romanov, HDLNoCGen. [Online]. Available: https://github.com/evgenii-lezhnev/HDLNoCGen.

[24] A. Romanov and E. Lezhnev, "Generator Verilog koda podsistemy svyazi setej na kristalle Verilog Code Generator of Communication Subsystem for Networks-on-Chip (HDLNoCGen)," RU Certificate of registration of the computer program 2021616623, 2021.

[25] K. Skovhede and B. Vinter, "Building hardware from C# models," Int. Workshop on FPGAs for Software Programme, vol. 3, pp. 1–9, 2016.

[26] A. Romanov, "Development of routing algorithms in networks-on-chip based on ring circulant topologies," Heliyon, vol. 5, no. 4, 2019.