Development of a Graphical Configurator for Networks on a Chip

N. I. Grushevskii
HSE Tikhonov Moscow Institute of Electronics and Mathematics
HSE University
Moscow, Russian Federation
nikarpenko@edu.hse.ru

S. M. Talibov
HSE Tikhonov Moscow Institute of Electronics and Mathematics
HSE University
Moscow, Russian Federation
smtalibov@edu.hse.ru

M. D. Shcheglev
HSE Tikhonov Moscow Institute of Electronics and Mathematics
HSE University
Moscow, Russian Federation
mdshcheglev@edu.hse.ru

2025 International Russian Automation Conference (RusAutoCon) | 979-8-3315-6708-8/25/$31.00 ©2025 IEEE | DOI: 10.1109/RusAutoCon65989.2025.11177335

***Abstract*— This paper describes the development of a graphical configurator of networks-on-chip (NoC) as a stage of development of the existing tool for low-level modeling of NoCs HDLNoCGen [1]. The main goal of developing the graphical configurator is to create an intuitive and functional graphical interface for visual design of NoCs, providing the capabilities of a visual editor with parameterization of NoC components. The developed graphical configurator of NoCs is implemented in Python, which makes it cross-platform. The key features of the system are: support for three design stages (mathematical model, basic representation, extended representation), interactive design mode with drag-and-drop technology, as well as extended parameterization of components. A mechanism for converting data between graphical representation and text format with support for import and export via JSON format was implemented. The result of the work was a new graphical interface for the low-level modeling environment NoCs HDLNoCGen, which is suitable for both educational tasks and research work.**

Keywords—automation, network-on-chip, NoC, interconnect, modeling, HDL

I. Introduction

In today's world, as computing tasks become more complex, so do the performance and integration of microprocessors. Commercial solutions are emerging with hundreds and thousands of processors combined into a single system on a chip [2]. Networks-on-Chip (NoC) are becoming a preferred solution because they provide more efficient communication between components [3]. NoCs provide communication through the use of routers and communication channels, which allows for parallel data transfer. This reduces latency and minimizes congestion, making NoCs especially useful in complex multiprocessor systems that require fast and efficient data transfer between cores.

With the development of NoCs, the need for a more flexible and convenient design method has become even more acute. The traditional method of developing networks on chip requires manual coding using complex mathematical models [4], which not only increases time costs, but also makes the process prone to errors. At the moment, two key difficulties can be identified in the development of a NoC. The first concerns the choice of the method for arranging cores and other elements on the chip to form an optimal topology, finding a routing mechanism that minimizes delays and increases throughput [5]. The second is low-level modeling. To analyze the developed NoC, its low-level modeling is carried out to obtain accurate characteristics of its operation. This usually occurs in event-based modeling environments [6] and can take a significant amount of time, depending on the size of the network [7]. In this case, the development and modeling of NoCs are carried out in various programs.

To overcome these difficulties, a fast and reliable way to comprehensively develop and simulate networks on chip is being actively sought.

II. Low-Level Network-on-Chip Models

When designing a NoC, the developer faces a large number of parameters: from the number and types of elements to the specifics of the target application. As a result, a large number of different low-level models of NoCs have been developed. However, most studies in the field of NoC modeling are diverse and unrelated. This problem is raised in the work [8], which provides a review of low-level models based on the criteria for synthesizing a hardware description, based on the NoC components under study, and concludes that the results of modeling using different models are not consistent with each other in terms of data presentation formats.

There is a shortage of specialized CAD systems for developing NoC and low-level modeling in particular. This problem is demonstrated in [9]. It analyzes NoC modeling tools according to various criteria: the language of the modeling tool implementation, the ability to add third-party components, the ability to generate an RTL description, and the complexity of working with the modeling tool. The author describes a number of problems with these tools that complicate their use. There is often no standardization when entering the description of the system under study, so developers need additional training to design using a specific development tool. Also, to study the designed NoC and optimize its architecture, a deep modification of the entire source code is required.

There are many low-level models available in the public domain, most of which either study individual parts of the NoC or model the NoC with a predetermined structure. For example, in the model [10], the NoC with mesh topology and without computing cores is studied. However, the ability to configure the network size, the size of packet flits and the length of buffers is supported.

In [11], NoC with mesh topology is synthesized, in which the distribution of physical connections of network elements and its impact on the occupied resources of the FPGA are studied.

In [12], NoC with mesh topology and an extensive router structure is implemented. However, the network does not provide for the connection of computing cores.

In [13], the authors propose a methodology for designing NoCs at an early stage using various tools for analyzing performance, delays and power consumption. However, the work does not provide low-level modeling.

In [14], the authors study NoC with torus topology for deadlocks. A network model is proposed that prevents deadlocks.

In [1, 15], the authors propose a model of a communication subsystem for constructing NoC based on the circulant, mesh, and torus topology, as well as the HDLNoCGen tools for automated NoC design. It allows configuring and generating NoC with various topologies in the HDL language Verilog. NoC can include not only a communication subsystem, but also computing cores, as well as additional components for modeling.

In [16], the authors present the CONNECT tool, which allows configuring a NoC router and constructing networks with various configurations based on it. However, this tool is currently unavailable, since the authors have removed it from open access. It also did not allow connecting computing cores.

In [17] the authors demonstrate a NoC simulator. It allows simulating an extremely limited set of topologies, only mesh and torus are supported. Only 2 routing algorithms are investigated. Also, the simulator does not support connections of computing cores and HDL code generation.

In [18], the authors propose a parameterized model with HDL code generation for modeling the NoC. The possibility of parameterizing the router structure, selecting a traffic template and various routing algorithms is provided. However, the work does not provide links to source codes for using the model.

In [19], the authors study a NoC with a bi-torus topology. The main emphasis of the model is on studying the efficiency of data packet transmission in the network. A distinctive feature of the model is the use of a computing unit to generate data packets.

In [20] demonstrates the development of a 16-core processor based on the NoC. The development process is shown in which only the Intel Quartus Prime CAD system was used, which is not a specialized tool for the development of the NoC and does not contain tools for automating this process.

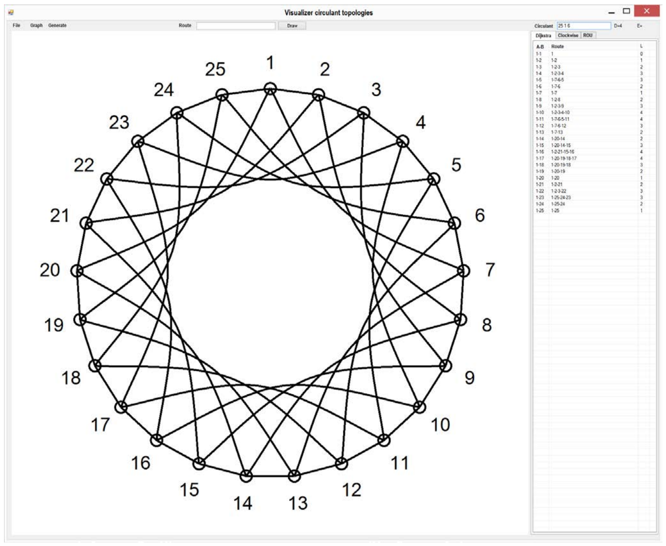
During the review of existing low-level models of NoC, the conclusions from [8] and [9] were confirmed, since most models study only individual network elements. Also, as a result of the analysis of these works, it was concluded that all the sources presented above contain only HDL descriptions of networks, but there are no automation tools. Also, none of the works present a graphical interface for convenient work with the model.

The exceptions are works [1] and [16], which contain not only a low-level model, and design automation tools. However, since the source codes are only available from HDLNoCGen, and there is also a graphical interface, its use as an automated tool for designing NoC is justified.

III. Modification of the HDLNoCGen GUI

A. Problems with the existing solution

Upon detailed examination of the HDLNoCGen tool, certain shortcomings of its graphical interface were revealed. Thus, the graphical interface demonstrated only the graph of the basis of the NoC communication subsystem, and also demonstrated the routes in the network calculated according to the specified algorithms. An example of the old graphical interface is shown in Fig. 1. All other capabilities, such as connecting computing nodes, more detailed network configuration, and the output of simulation results were visually hidden and configured exclusively by configuration files.



A-B	Route	L
1-1	1	0
1-2	1-2	1
1-3	1-2-3	2
1-4	1-2-3-4	3
1-5	1-7-6-5	3
1-6	1-7-6	2
1-7	1-7	1
1-8	1-2-8	2
1-9	1-2-3-9	3
1-10	1-2-3-4-10	4
1-11	1-7-6-5-11	4
1-12	1-7-6-12	3
1-13	1-7-13	2
1-14	1-20-14	2
1-15	1-20-14-15	3
1-16	1-2-21-15-16	4
1-17	1-20-19-18-17	4
1-18	1-20-19-18	3
1-19	1-20-19	2
1-20	1-20	1
1-21	1-2-21	2
1-22	1-2-3-22	3
1-23	1-25-24-23	3
1-24	1-25-24	2
1-25	1-25	1

Fig. 1. The original graphical interface of the HDLNoCGen program.

Thus, when developing a graphical configurator with an intuitive and functional graphical interface for visual design of the NoC, providing the capabilities of a visual editor with parameterization of network components, the following tasks were set:

- Development of a graphical interface for the HDLNoCGen program, allowing visualization of the topology of the NoC communication subsystem;
- Development of logical and graphical stages of the NoC configuration;

- Implementation of the parameterization of the NoC components;
- Development of the structure of the JSON file for storing the NoC configuration.

B. *Implementation of logical and graphical stages of the NoC configuration*

When designing complex systems, one of the classic methods is top-down design [21], when the entire system can be presented in several views, differing in the level of detail. The general structure of the NoC is shown in Fig. 2.

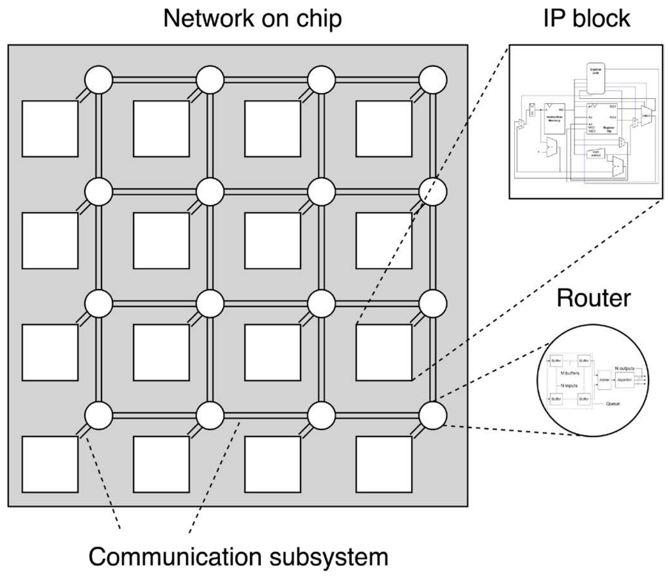


Fig. 2. Classical structure of the NoC.

The main components of the NoC are the topology of the communication subsystem, which in general can be represented as an undirected graph, a router and a computing cores (IP block). When designing the NoC, each of these components can be developed independently. And only at the final stage of design, when it is necessary to combine all elements at the signal level, the final network structure is formed.

As a result, a three-level design system was implemented, including both logical and graphical stages of the NoC configuration. This approach allows for a consistent transition from an abstract representation of the NoC in the form of a graph to its detailed implementation, which corresponds to modern methods for designing complex network structures. The logical stages of design represent a hierarchy of models that provide a gradual complication of the network description.

The first level uses a simplified mathematical model that represents the network as an undirected graph, where the nodes correspond to routers and the edges correspond to communication channels between them. This representation allows us to analyze basic topological properties of the network, such as diameter, average path length, and throughput.

The extended model adds functional network elements to consideration – computing cores, memory blocks, input/output interfaces. At this level, not only the connection structure is taken into account, but also the functional purpose of each element, which allows for a more accurate analysis of network performance. The connections between components at the second level are not detailed.

The third level of the NoC display includes all network components, taking into account their parameters and characteristics, as well as a detailed description and display of all types of connections between them. The connections are presented in the form in which they are actually used in the structure of the network on a chip when it is implemented in RTL.

Graphic design stages correspond to logical models and are implemented in the form of three visualization modes: basic, extended and advanced, each of which includes a different set of components and also has different functions.

The basic mode displays the network as an undirected graph, where routers are represented by nodes and links are simple lines. This mode is designed for quickly creating and analyzing the overall NoC structure without being distracted by implementation details. When switching between modes, all components are saved, but in the basic mode, the kernels, DS, and Uart are temporarily hidden from view. However, it is not recommended to add elements of the extended and advanced modes and then edit the location and number of routers in the basic mode, as this may lead to an incorrect topology. An example of a NoC configuration with 16 nodes and a circulant topology in the basic mode is shown in Fig. 3.

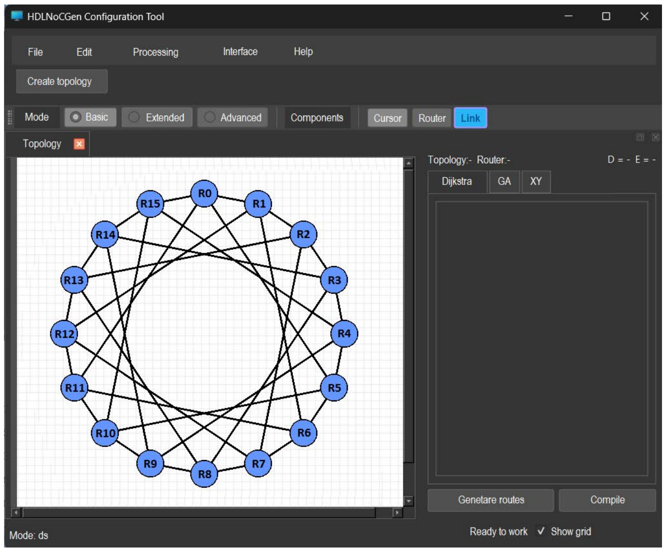
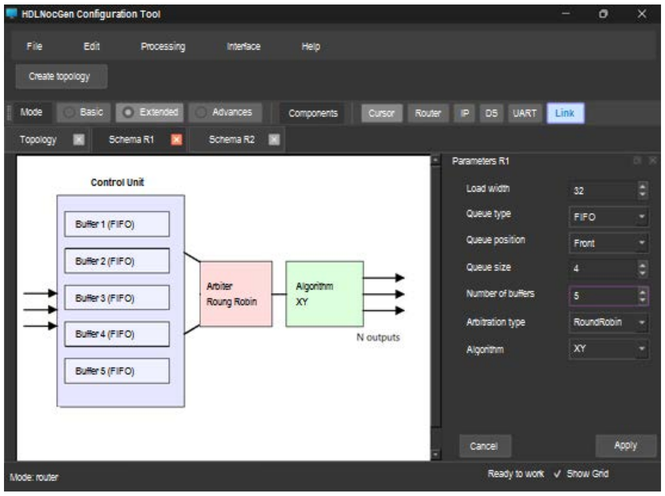


Fig. 3. Configuration of the NoC in basic mode.

In the extended mode, all functional network elements are added to the routers: computing cores, memory blocks, Uart interfaces. At this level, parameterization of elements becomes available, in particular, the ability to configure the internal structure of routers through a special property editor (Fig. 4). During automatic design, loading of components is carried out through Verilog code, and in interactive mode by adding the corresponding graphic elements.



Parameters R1

Load width	32
Queue type	FIFO
Queue position	Front
Queue size	4
Number of buffers	5
Arbitration type	RoundRobin
Algorithm	XY

Fig. 4. Configuration of router parameters.

The advanced mode provides a complete detailed view of the network, including all connections between components. Unlike previous modes, where connections were displayed as a single line, here each physical connection line is presented separately. When switching to the advanced mode, all connections that are in the characteristics of the NoC components are automatically displayed next to each existing connection line. Any of the additional lines can be deleted, while the main connection line displayed in all modes will remain.

Another feature of this mode is the ability to edit additional connections - break them and add specialized elements, such as power elements (VCC and GND). Standard designations for circuit diagrams were chosen as graphic primitives for them. These elements can be selected and deleted without deleting the communication lines themselves. When deleting a line on which the power is located, the element is deleted automatically.

When you delete one of the connected components, all the connection attributes (additional lines and power elements) are automatically deleted. When you switch from advanced to basic or extended mode, all additional attributes are hidden, and when you return, they are displayed again. This allows you to create accurate models of real network structures, taking into account all the features of their implementation.

The developed system of configuration stages provides a flexible approach to design, allowing the user to select the optimal level of detail depending on the task at hand. The transition between modes is performed without data loss, which makes it possible to consistently refine the project from the general concept to detailed implementation. This approach significantly increases the efficiency of work, especially when designing complex multi-level structures.

C. *Interactive design mode of NoC*

The user interactive design mode of the NoC is a tool for visual design of NoC with irregular topologies. In the HDLNoCGen program, a module was added that allows you to specify not only regular topologies, such as mesh, torus, circulant or others, for which the configuration of the router connection can be calculated using certain formulas, but also allows to specify it in the form of an adjacency matrix. For such topologies, the correct placement of their components on the screen is difficult. Therefore, to implement this feature, the ability to change the location of graphic components is implemented in the new graphical interface.

The elements of the NoC are displayed as graphic primitives with a name (the first letter of the name of structural NoC elements) and a serial number: the router is displayed as a blue circle, the core is an orange square, and the traffic differentiation block (DS) is a green rectangle. All elements support movement around the scene together with text labels.

After clicking the "Interactive design" button, the advanced graphic design mode of NoC opens in the main application window, it is shown in Fig. 5. The figure also shows an example of constructing a topology in interactive mode. When hovering over all elements, hints appear, eliminating difficulties when using the application for new users.

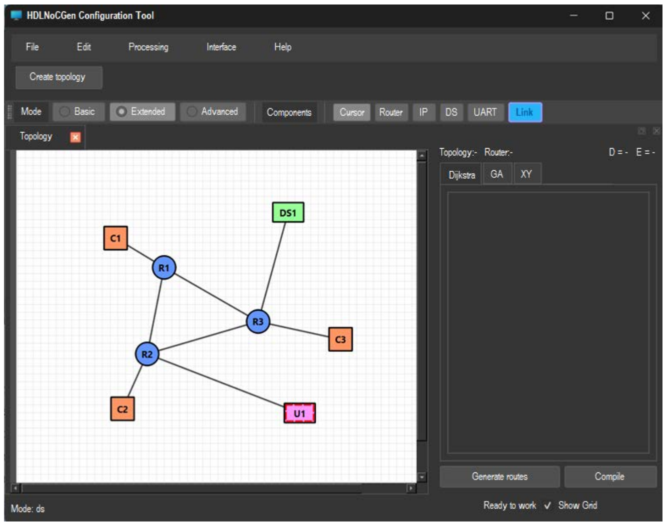


Fig. 5. NoC design interface in advanced mode.

To design volumetric topologies with a large number of components, zoom-in and zoom-out functions were implemented, caused by scrolling the wheel. Moving around the scene is carried out by scrolling the side sliders to the right and below the scene.

IV. Parameterization of the Components of the NoC

As noted earlier, in the extended and advanced design modes, the user has access to parameterization of components. To display a detailed structure, right-click on the desired component, then select the "Show structure" option from the drop-down list.

Router parameterization opens as an additional tab on the top panel of the interface, allowing the user to freely switch between tabs. The structure tabs include the ordinal number of the selected router in their name, allowing the structure of several components to be edited simultaneously.

By default, the router has 4 input buffers, a data packet selection arbiter, and a routing algorithm. These elements can be changed via the properties panel, which opens when switching to the structure display mode. The user can select the type, position, and length of the queue, the arbiter type, and change the number of buffers and the algorithm. Changes to the number of buffers and other parameter values are automatically updated when applying the settings and are immediately displayed in the construction area.

When you right-click on a router, the drop-down list also includes the "Copy Properties" function, which automatically assigns the selected configuration to the selected routers. This will simplify the design process for heterogeneous NoC, which may contain several types of routers.

All applied parameterizations of components are based on their associated graphical representations of HDL descriptions in the Verilog language, which allows the user to use not only those elements that are available in the program, but also to add their own.

V. Saving the NoC Project

Another problem with the HDLNoCGen program is the inability to save a configured NoC project to continue working. Also, the lack of saving required the network design to be performed again each time the program was launched. To solve this problem, an additional program module was developed that saves both the graphic and HDL network project as a JSON file. It implements a structure that ensures complete preservation of the project state, including not only the location of the components, but also all their parameters. The file architecture is based on the principle of dividing the data into three logical blocks: project metadata, description of the elements, and information about the connections between them. This approach was chosen as an optimal compromise between the readability of the format and the efficiency of its processing by the program.

The metadata block stores key information about the format version (the "version" field), which allows for the implementation of a backward compatibility mechanism in the future when changing the file structure. The state of the visual interface elements is also saved here like the visibility of the coordinate grid ("grid_visible") and the current scaling factor ("zoom_factor"). These parameters are critically important for recreating an identical working environment when loading a project, so they are placed in a separate section. Saving the above-mentioned display settings is important when developing complex NoC configurations and will allow the design to continue from the point where the user stopped, if he zooms the scene into the area of this component before saving.

The description of the system elements is organized into a hierarchical structure with division by component types. Each element has a unique identifier, formed on the basis of the element type and its ordinal number (for example, "R1" for the first router, "C3" for the third core). This approach to naming ensures easy identification of elements both during automatic processing and during possible manual editing of the file.

Particular attention is paid to storing router parameters, which are saved in full in the "properties" subsection. All configurable characteristics are recorded here: data bus width ("data_width"), queue type and parameters ("queue_type", "queue_length"), arbitration algorithms ("arbiter_type") and routing ("algorithm").

Also, the JSON file for each NoC component stores the path to the component description file in Verilog, which makes independent configuration of NoC components possible. Such a detailed approach ensures that when loading the project, all fine-tuning of the routers will be restored exactly as when saving.

The connection block is implemented using a reference model, where each connection is defined not by coordinates, but by identifiers of the connected elements. This solution significantly increases the reliability of loading, eliminating problems associated with inaccurate positioning of elements. Additionally, control points of connection lines ("points") are saved, which ensures the visual identity of the project when reloading. If the project was modified in advanced mode by adding power elements or removing additional communication lines, then all changes made are also saved when saving and loading.

The process of saving data is organized through a sequential traversal of all scene elements and the formation of corresponding records in the JSON structure. For routers, their parameters are additionally extracted and serialized. Connections are saved after all elements, when a dictionary of correspondences between graphic objects and their identifiers is guaranteed to be formed.

When loading a file, the reverse process is used: first, all scene elements are created and the dictionary of correspondences is filled, then the connections between them are restored. A special feature of the implementation is the two-stage processing of connections - first by element identifiers, and if such a comparison is impossible (to ensure backward compatibility), then by coordinate correspondence.

The chosen JSON file structure provides several key benefits:

- Complete preservation of the project state, including all component parameters;
- Convenience of manual editing due to clear structure and naming;
- Possibility of format expansion without breaking backward compatibility;
- Efficient processing due to optimal data organization;
- Reliability of loading due to a well-thought-out system of links between elements.

Also, using a JSON file to save a NoC project allows you to organize interaction with other NoC design and modeling tools, due to the use of a single organization of program output files.

VI. Conclusion

The developed graphic configurator for the low-level modeling program of NoC HDLNoCGen allows to significantly expand the capabilities of the program. It became possible to configure NoC with any topology of the communication subsystem, due to the addition of a module for forming a topology through an adjacency matrix to the program. The ability to configure NoC with various settings of components and, in particular, routers has been added, which allows more flexible configuration of NoC for the required characteristics. The introduction of three levels of abstraction in network design allows to study only certain configurations of NoC, without carrying out its full parameterization. Saving the parameters of the NoC project as a JSON file allows not only to conveniently carry out the development process, but also allows combining third-party programs into a single set by combining the results in a single file.

Thus, the changes made to the low-level modeling program for the HDLNoCGen NoC design make it a convenient and powerful tool for conducting research in the field of networks on a chip.

References

[1] E. V. Lezhnev, V. V. Zunin, A. A. Amerikanov, and A. Y. Romanov, "Electronic computer-aided design for low-level modeling of networks-on-chip," in IEEE Access, vol. 12, pp. 48750–48763, 2024, doi: 10.1109/ACCESS.2024.3382710.

[2] J. Kelm, D. Johnson, N. Crago, et al. "Rigel: An architecture and scalable programming interface for a 1000-core accelerator," in Proceedings of the 36th Annual International Symposium on Computer Architecture, 2009.

[3] R. Poovendran, S. Billclinton., R. Darshan., R. Dinakar, and M. Fazil, "Design and analysis of a mesh-based adaptive wireless network-on chips architecture with irregular network routing," in 2019 IEEE International Conference on System, Computation, Automation and Networking, 2019, pp. 1–6, doi: 10.1109/ICSCAN.2019.8878683.

[4] S. Mosin, M. Hassan, and A. Tuxtamirzaev, "Mathematical model of user network-on-chip," (in Russian), in Software Products and Systems, no. 3, pp. 249–25, 2012.

[5] A. Punhani, N. Faujdar, and S. Kumar, "Design and evaluation of cubic torus network-on-chip architecture," in International Journal of Innovative Technology and Exploring Engineering, vol. 8, no. 6, pp. 2278–3075, 2019.

[6] H. Zhao and S. Mai, "Simulation of ASIC/FPGA cochlear implant speech processor with SIMULINK and Modelsim cosimulation method," in 2013 IEEE International Conference of IEEE Region 10 (TENCON 2013), 2013, pp. 1–4, doi: 10.1109/TENCON.2013.6718808.

[7] N. Genko, D. Atienza, and G. Micheli, "A complete network-on-chip emulation framework," in Proceedings of the Design, Automation and Test in Europe, 2005, pp. 246–251.

[8] A. Ben and S. Ben, "A survey of network-on-chip tools," in International Journal of Advanced Computer Science and Applications, vol. 4, no. 9, pp. 61–67, 2013, doi: 10.14569/ijacsa.2013.040910.

[9] W. Meeus, K. Van Beeck, and T. Goedeme, "An overview of today's high-level synthesis tools," in Des. Autom. Embed. Syst, vol. 16, no. 3, pp. 31–51, 2012.

[10] "NoCPhoenixVerilog". [Online] Available: https://github.com/peterfalcao/NoCPhoenixVerilog

[11] I. Korotkyi and O. Lysenko, "Application-specific network-on-chip with link aggregation," in Mediterranean Conference on Embedded Computing (MECO), 2012, pp. 9–12.

[12] "Network-On-Chip". [Online] Available: https://github.com/mohasnik/Network-On-Chip

[13] L. Ost, L. S. Indrusiak, S. Määttä, et al., "Model-based design flow for NoC-based MPSoCs," in 17th IEEE International Conference on Electronics, Circuits and Systems, 2010, pp. 750–753, doi: 10.1109/ICECS.2010.5724621.

[14] S. Das and C. Karfa, "Arc model and DDG: Deadlock avoidance and detection in Torus NoC," in IEEE Embedded Systems Letters, vol. 14, no. 2, pp. 67–70, 2022, doi: 10.1109/LES.2021.3113355.

[15] E. Lezhnev, "HDLNoCGen". [Online]. Available: https://github.com/evgenii-lezhnev/HDLNoCGen

[16] M. Papamichael and J. Hoe, "The CONNECT network-on-chip generator," in Computer, vol. 48, no. 12, pp. 72–79, 2015, doi: 10.1109/MC.2015.378.

[17] N. Baby, S. Mathew, S. Abraham, et al., "Network on chip simulator: Design, implementation and comparison of Mesh, Torus and RiCoBiT topologies," in 2nd International Conference on Next Generation Computing Technologies (NGCT), 2016, pp. 46–50, doi: 10.1109/NGCT.2016.7877388.

[18] K. Parane, P. Prabhu, and B. Talawar, "FPGA based NoC simulation acceleration framework supporting adaptive routing," in 2018 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), 2018, pp. 1–6, doi: 10.1109/CONECCT.2018.8482386.

[19] M. Schoeberl, L. Pezzarossa, and J. Sparsø, "S4noc: A minimalistic network-on-chip for real-time multicores," in Proceedings of the 12th International Workshop on Network on Chip Architectures, 2019.

[20] E. Fernandez-Alonso, D. Castells-Rufas, S. Risueño, et al., "A NoC-based multi-{soft}core with 16 cores," in 17th IEEE International Conference on Electronics, Circuits and Systems, Athens, 2010, pp. 259–262, doi: 10.1109/ICECS.2010.5724503.

[21] M. Hallwood, "VHDL and top-down methods prove successful in automotive electronic design", in IEE Colloquium on Computer Aided Engineering of Automotive Electronics. IET, 1994.