Universal On-Chip Network Simulator for Networks-on-Chip Development

A. A. Amerikanov
HSE University
Moscow, Russia
aamerikanov@hse.ru

A. S. Ponomarev
HSE University
Moscow, Russia
asponomarev@miem.hse.ru

2021 International Russian Automation Conference (RusAutoCon) | 978-1-6654-1523-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/RusAutoCon52004.2021.9537564

***Abstract*—This work is devoted to Universal On-Chip Network Simulator (UOCNS) for Network-on-Chip (NoC) development. The principle of the topological approach in NoC design was analyzed. A review of high-level NoC simulators was done. A complete analysis of the structure of UOCNS and its main components was carried out, as well as it was tested on different topologies. A comparison of the simulation results of the same network in UOCNS and in other simulators was made, and the correctness of the model was confirmed.**

Keywords—NoC, regular topology, circulant topology, high-level modeling, gpNOCSim, UOCNS.

I. Introduction

Uniprocessor systems cannot cope with tasks requiring high computational performance and are inefficient when dealing with large data streams. Therefore, due to the constant growth of the processed information and the increase in the complexity of the tasks being solved, there is a tendency to an increase in the number of processor cores and various peripheral modules within one device. For example, Cerebras has developed a WSE2 chip which consists of 850,000 cores [1], and progress in the design of multi-core systems continues.

For the design of complex multicore systems, it is possible to use architectural solutions from the field of NoCs. The process of NoC design can be divided into 6 main sequential stages: preparation of technical specifications, design, high-level modeling, low-level modeling, prototyping or cosimulation, production stage.

The high-level modeling stage makes it possible to select a limited number of sets of parameters and network characteristics suitable for the further design selected at the NoC design stage. In high-level modeling, it is worth considering many network parameters the most important of which is NoC topology. It is the network topology that largely determines the efficiency of the device as a whole.

Existing models generally support a limited number of topologies which significantly limits their applicability to arbitrary topologies. Out of more than 100 high-level models, only 22% of models support arbitrary topologies, according to our survey [2]. In this regard, there is a need to analyze and refine existing high-level models capable to calculate parameters for networks with arbitrary topologies.

II. Topological Approach to NoC Design

When designing NoCs, the method of connecting processor cores and peripheral devices of the network (or its topology) is of great importance [3]. Mostly, topologies can be divided into two types: irregular and regular.

Irregular topologies are described by either a list or an adjacency matrix. To transfer data from node to node in such networks, either broadcasting or a special table (contained in the router) that specifies to which node a packet with data should be sent to reach its destination is normally used. As the number of nodes increases, so does the memory occupied by the table in the routers. It is also worth noting that due to the irregularity of the structure, this table will be individual for each network node.

It appears from the above that the design of irregular topologies is a complex and resource-intensive process. These topologies are typically used in systems with few nodes or in systems where the load is unevenly distributed between IP blocks.

Regular topologies are based on some kind of regular structure, for example, circulant [4]. Such topologies, due to their regular structure, usually have simple routing algorithms and are easily synthesized. The most popular topology for NoC design is mesh topology [5]; as well as based on it torus [6], or WK-recursive [7]. WSE2 cores are connected using mesh topology [8].

Mesh topologies allow the use of simple routing algorithms and are highly reliable in contrast to hierarchical and tree topologies (fat-tree [9], butterfly [10], etc.) in which the failure of communication between two nodes can lead to the loss of functionality of the entire network. But, unfortunately, with all their advantages, mesh-based topologies have a significant drawback – poor scalability and the dependence of their characteristics on the geometric shape.

Circulant topologies, having no scaling problem, are the most promising regular topologies for use in NoCs. Topologies of this type are easily scalable and have a regular structure that allows the use of deterministic algorithms for transferring data between nodes. It should be noted that due to the variety of circulant topologies, there is a need to find optimal circulants and to develop optimal adaptive routing algorithms for these circulants.

III. Review of NoC High-Level Simulators

High-level models allow for some simplifications and description in high-level languages which can lead to discrepancies between the simulation results and real data, but modeling is much faster compared to low-level models. According to [11], high-level network simulation for 100 nodes

Authorized licensed use limited to: Higher School of Economics. Downloaded on October 06,2021 at 12:56:06 UTC from IEEE Xplore. Restrictions apply.

in the high-level OCNS takes a few minutes while low-level simulation of the same network in a low-level simulator Netmaker [12] takes several days. This feature makes it possible to use high-level simulators for the initial selection of topology parameters for further modeling on a low-level simulator.

Currently, there are many high-level solutions in the field of NoC modeling. Among them, the BookSim simulator [13] (a console application developed in C ++ and designed to run on UNIX-like systems) can be distinguished.

The program allows configuring the following parameters using the configuration file [14]: topology type (mesh, cmesh [15], torus, butterfly, flattened butterfly [16], fat tree, quad tree [17]) and the number of nodes; routing algorithm (different set for different topologies); number, buffer size, and method of allocating virtual channels; router architecture; data packet size, packet rate, and type of traffic distribution; duration of the simulation period and the total number of periods.

According to the specified parameters, the model calculates the following characteristics: network latency, packet delivery time, and flit delivery time; the size of sent and received packets; rate of generation and reception of packets; rate of generation and reception of flits; the number of transit sections of the packet; simulation run time.

The results are output to the console but can be redirected to a text file.

It should be noted that the model is open source code [14] and can be modified (for example, to work with other types of topologies).

Another model, Noxim, is open source code [18] which allows making changes to the program.

The model allows setting various parameters, including: network topology; routing algorithm; buffer size; the number of virtual channels; packet size and type of traffic distribution; power consumption parameters.

According to the specified parameters, the model calculates a number of characteristics, including: throughput; network latency; the number of received packets and flits; energy consumed by the system.

One of the independent modifications of the model is Newxim [19]. This version of the program adds support for circulant topologies and specialized routing algorithms. When modifying the model, the code was also refactored.

Another simulator to be considered is TOPAZ [20] developed in the C ++ language. The program allows configuring the following parameters: topology type (mostly Mesh type); routing algorithm; traffic distribution type; buffer size and the number of router ports; the size of packets and flits.

The simulator calculates the following NoC characteristics: throughput and network latency; data packet sending rate; data packet processing rate; metrics of network connection utilization.

Like the models already considered, TOPAZ is open source [21].

Another simulator, gpNoCsim, is of great interest [22]. It supports the launch of such topologies as mesh, torus. The gpNoCsim++ modification [23] added support for WK-recursive [7]. The disadvantage of this simulator is the support of a limited number of routing algorithms, high complexity of modification, manual generation of a file with the input parameters of the model.

The development of gpNOCSim is OCNS. In this simulator, the modules for network formation and topology construction were reworked; it allowed specifying any topology with a description in the form of an incidence table. Routing in this simulator is done using routing tables. Currently, OCNS has received another development called UOCNS [24], where support for circulant topologies and routing algorithms appeared. Also, this simulator has a server version of UOCNS-SE which allows running the model on remote computing power.

IV. UOCNS Model

This section provides a detailed description of the internal structure of the simulator program, input parameters, calculated characteristics, and features of the model.

A. Structure of model program

To implement the UOCNS model, the Java language was used and the principles of object-oriented programming were applied. The source code of the program consists of a large number of components that play the role of various NoC elements. Fig. 1 shows a diagram of the relationships between the main logical parts of the simulator. To improve the perception, only the most important classes are shown, but the program also contains other less important classes and interfaces.

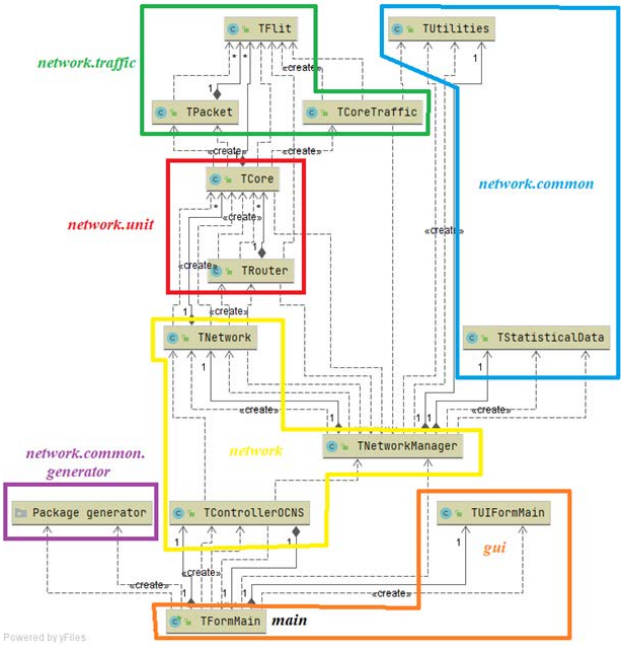

Fig. 1. UML diagram of UOCNS.

Authorized licensed use limited to: Higher School of Economics. Downloaded on October 06,2021 at 12:56:06 UTC from IEEE Xplore. Restrictions apply.

In the diagram, the component which is responsible for representing the data transmitted over the network and which describes flies, packets, and traffic in general is highlighted in green. The component marked in red is responsible for the operation of compute nodes and routers. The yellow in the figure denotes the controller managing the simulation process and calculating characteristics of NoC under study. The generator of configuration files is highlighted in purple, and various auxiliary utilities are highlighted in blue. The component which contains the entry point to the program and which is responsible for the elements of the graphical interface and interaction with them is marked in orange.

B. Configurable parameters

The model allows setting the following NoC parameters: topology (mesh, torus, circulant, optimal circulant); topology arguments (number of nodes vertically and horizontally for mesh and torus topologies; total number of nodes, step of the smaller generator and step of the larger generator for the circulant topology; total number of nodes for optimal circulant topology); routing algorithm (XY algorithm for mesh and torus topologies; Dijkstra's algorithm, Clockwise routing algorithm or adaptive algorithm for circulant and optimal circulant topologies) [25]; the number of virtual channels; the size of the buffer of virtual channel (flits); fleet size (bits); the average packet length (flits); fixed packet size (yes / no); the average period for generating packets (cycles); the number of packets received at which the simulation ends; the number of packets received at which the warm-up completes; the number of simulator runs.

To generate a configuration file, the program (according to topology data, topology arguments, and routing algorithm) forms a node connection table and a routing table.

To consider the structure of routing tables, it is necessary to understand that each computational node contains a certain number of ports numbered in a certain way depending on the topology as shown in Fig. 2.

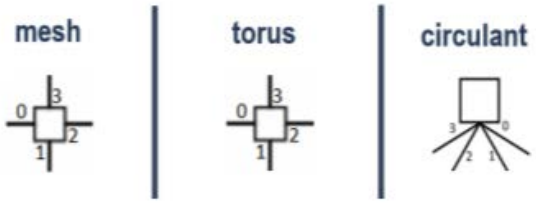

Fig. 2. Accepted numbering of ports of compute nodes.

The connection table describes the connections between node ports. Rows correspond to NoC nodes, columns - to node ports. The value in the cell is the number of the node to which the node specified in the row is connected using the port specified in the port column.

Table 1 shows an example of a table of node connection for circulant $C\ (7;\ 1, 2)$.

The routing table describes the links between NoC nodes. The value in the cell shows which node the packet should be sent to in order to deliver it from the node specified in the row to the node specified in the column. Table 2 shows an example of a routing table of circulant $C(7;\ 1, 2)$.

TABLE I. TABLE OF NODE CONNECTION FOR CIRCULANT $C(7; 1,2)$

Node number	Node port number			
	0	*1*	*2*	*3*
0	1	2	5	6
1	2	3	6	0
2	3	4	0	1
3	4	5	1	2
4	5	6	2	3
5	6	0	3	4
6	0	1	4	5

Fig. 3 shows the circulant which corresponds to Table 1.

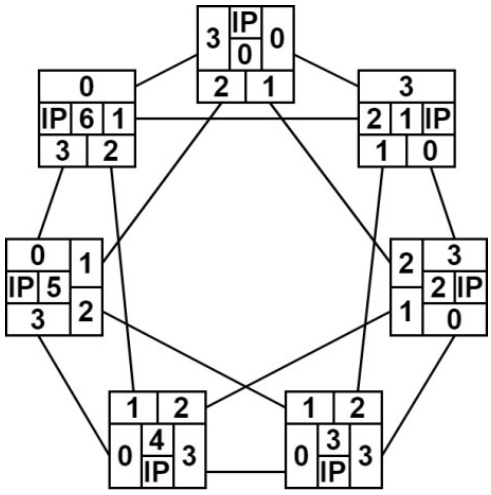

Fig. 3. Circulant $C(7;\ 1,2)$.

TABLE II. THE ROUTING TABLE FOR CIRCULANT $C(7;\ 1,2)$

Source node number	Destination node number						
	0	*1*	*2*	*3*	*4*	*5*	*6*
0	4	0	1	0	1	2	3
1	3	4	0	1	0	1	2
2	2	3	4	0	1	0	1
3	1	2	3	4	0	1	0
4	0	1	2	3	4	0	1
5	1	0	1	2	3	4	0
6	0	1	0	1	2	3	4

After generating the node connection table and the routing table, it becomes possible to generate a configuration file (example in Fig. 4).

Authorized licensed use limited to: Higher School of Economics. Downloaded on October 06,2021 at 12:56:06 UTC from IEEE Xplore. Restrictions apply.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
- <TaskOCNS Description="Mesh-(2, 2)"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <Network>
          <Netlist> -1 2 1 -1 0 3 -1 -1 -1 -1 3 0 2 -1 -1 1 </Netlist>
          <Routing> 4 2 1 1 0 4 1 1 3 3 4 2 3 3 0 4 </Routing>
        - <Link>
              <Parameter FifoSize="4"/>
              <Parameter FifoCount="4"/>
          </Link>
      </Network>
    - <Traffic>
          <Parameter FlitSize="32"/>
          <Parameter PacketSizeAvg="10"/>
          <Parameter PacketSizeIsFixed="true"/>
          <Parameter PacketPeriodAvg="5"/>
      </Traffic>
    - <Simulation>
          <Parameter CountRun="1"/>
          <Parameter CountPacketRx="1100"/>
          <Parameter CountPacketRxWarmUp="100"/>
          <Parameter IsModeGALS="false"/>
      </Simulation>
  </TaskOCNS>
```

Fig. 4. Example of a configuration file.

C. Calculated characteristics

The file with the obtained simulation results contains the following data: simulation time (cycles); number of packets sent; number of packets received; number of packet generation errors; packet generation rate (packets / cycles); flit generation rate (flits / cycles / cores); flit sending rate (flits / cycles / cores); packet delivery time (cycles); number of packet hops; network throughput (flits / cycles); router throughput (flits / cycles); load of receiving node buffers (%); load of transmitting node buffers (%); load of receiving router buffers (%); load of transmitting router buffers (%); load of network buffers (%); load of physical network channels (%).

A wide range of characteristics measured by the model allows getting a fairly accurate idea of the efficiency of the NoC configuration under study.

D. Features of model

One of the differences in the model lies in the presence of a graphical interface since most of the existing simulators are console applications. The graphical interface allows various simulation settings, monitoring the simulation progress, as well as displaying the results. But there is also the possibility of working with the program in console mode. In this case, the simulation parameters are passed as command line arguments.

Another UOCNS feature is the ability to use a custom configuration file for modeling instead of generating a configuration file by the program which allows simulating any NoC topology, including irregular ones.

An important advantage of the program is the built-in mechanism for automatic repeated simulation with a change in the period of packet generation during the transition to the next iteration of the simulation. This approach makes it possible to find a NoC saturation point which, in turn, makes it possible to determine the most optimal NoC configurations of those under study.

V. UOCNS Testing

To confirm the correctness of the model, a number of simulations of various topologies supported by the simulator were carried out, as well as a comparison of the calculated characteristics with the results of the operation of other existing NoC models was made.

A. Modeling Mesh NoC topology

During testing, NoC simulation was carried out, and the parameters were as follows:

- Mesh topology, 8 nodes wide, 8 nodes long.
- Mesh topology, 4 nodes wide, 16 nodes long.

The results are shown in Fig. 5.

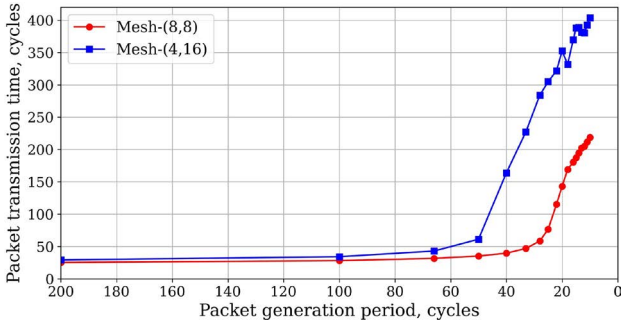

Fig. 5. Comparison results of Mesh NoC topologies.

According to the graph, the optimal shape for mesh topology is square which is consistent with NoC theory.

B. Modeling Torus NoC topology

To check the work of the program when modeling torus NoC topology, characteristics of NoC of the same size as in the study of mesh topology were calculated. The measurement results are shown in Fig. 6.

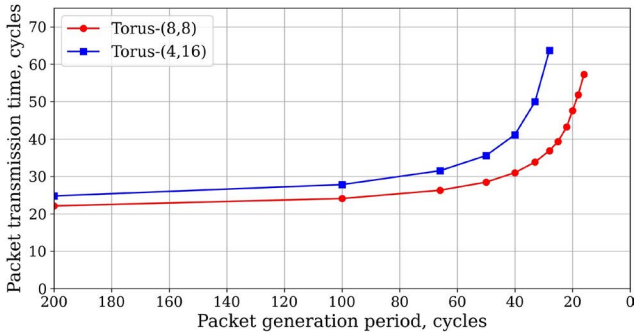

Fig. 6. Comparison results of Torus NoC topologies.

The obtained result shows that the square-shaped torus has better characteristics.

C. Modeling circulant NoC topology

When testing the model, a comparison of the performances of the circulant and the optimal circulant with the same number of nodes was made. The result is given in Fig. 7; it shows the presence of a small gain in packet transmission time when using optimal circulants.

Authorized licensed use limited to: Higher School of Economics. Downloaded on October 06,2021 at 12:56:06 UTC from IEEE Xplore. Restrictions apply.

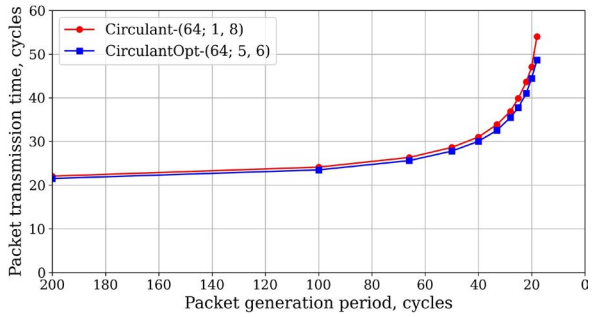

Fig. 7. Comparison results of circulant NoC topologies.

D. Comparison of different NoC topologies

A comparison of NoC performance was made, and the parameters were as follows:

- Mesh topology, 8 nodes wide, 8 nodes long.
- Torus topology, 8 knots wide, 8 knots long.
- Circulant topology, 64 nodes, generators are 5 and 6.

The result (Fig. 8) shows that mesh topology is the least productive, and the circulant topology is slightly superior to the torus.

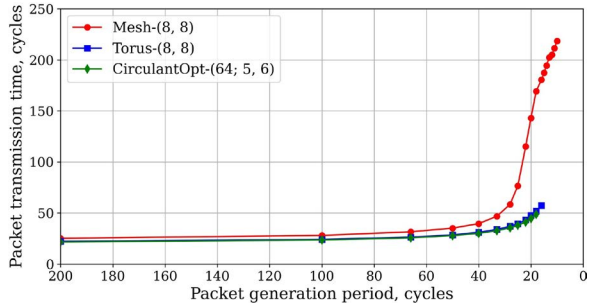

Fig. 8. Comparison results of different NoC topologies.

E. Comparison of UOCNS with other simulators

To assess the correctness of values of characteristics measured by the model, a comparison of the results obtained by UOCNS, BookSim, and Newxim when modeling mesh NoC topology (8 nodes long, 8 nodes wide) was made.

Fig. 9 shows the approximate equality of obtained dependences of the network throughput on the frequency of flit generation. The existing differences can be explained by the features of performing calculations in simulators, as well as by the presence of unique parameters for each of the models.

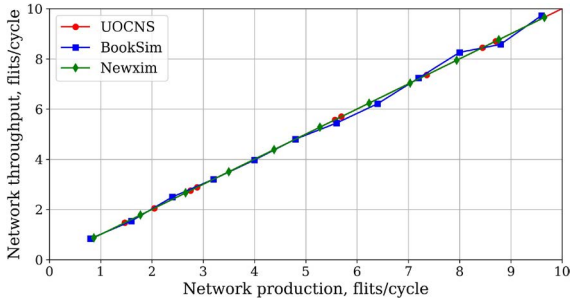

Fig. 9. Performance comparison of simulators.

VI. Conclusions

The article examines the principle of the topological approach in NoC design. It was shown that topology has a large impact on network performance. A review of high-level NoC simulators was done, and all the main characteristics of simulators were considered. A complete analysis of UOCNS performance and its main components was carried out.

UOCNS operability check was carried out, and the results correspond to NoC theory and are comparable with the performance of other simulators; so, the correctness of the UOCNS operation was confirmed. Also, the simulation results showed the advantages of circulant topology in NoC design.

Acknowledgment

This article is an output of a research project implemented as part of the Basic Research Program at the National Research University Higher School of Economics (HSE University).

References

[1] S. K. Moore, “Cerebras’ New Monster AI Chip Adds 1.4 Trillion Transistors,” IEEE Spectrum. [Online]. Available: https://spectrum.ieee.org/tech-talk/semiconductors/processors/cerebras-giant-ai-chip-now-has-a-trillion s-more-transistors

[2] A. Y. Romanov and A. Opekunova, “NoC_High_Level_Models_2020”. [Online]. Available: https://docs.google.com/spreadsheets/d/1IUwYoQHOmhp0tq0HPFFPYP6fPScNbPz34G71JbOBKQ/

[3] M. Coppola, M. D. Grammatikakis, R. Locatelli, G. Maruccia, and L. Pieralisi, Design of Cost-Efficient Interconnect Processing Units. CRC Press, 2020.

[4] F. Boesch and R. Tindell, “Circulants and their connectivities,” J. Graph Theory, vol. 8, no. 4, pp. 487–499, 1984. DOI: 10.1002/jgt.3190080406.

[5] D. Deb, J. Jose, S. Das, and H. K. Kapoor, “Cost effective routing techniques in 2D mesh NoC using on-chip transmission lines,” J. Parallel Distrib. Comput., vol. 123, pp. 118–129, 2019. DOI: 10.1016/j.jpdc.2018.09.009.

[6] A. Q. Ansari, M. R. Ansari, and M. A. Khan, “Modified quadrant-based routing algorithm for 3D Torus Network-on-Chip architecture,” Perspect. Sci., vol. 8, pp. 718–721, 2016. DOI: 10.1016/j.pisc.2016.06.069.

[7] S. Suboh, M. Bakhouya, and T. El-Ghazawi, “Simulation and evaluation of on-chip interconnect architectures: 2D mesh, Spidergon, and WK-recursive network,” Proc. – Second IEEE Int. Symposium on Networks-on-Chip, pp. 205–206, 2008. DOI: 10.1109/NOCS.2008.4492739.

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

[9] A. Bouhraoua and M. E. Elrabaa, “An efficient network-on-chip architecture based on the Fat-Tree (FT) topology,” Proc. of the Int. Conf. on Microelectronics, pp. 28–31, 2006. DOI: 10.1109/ICM.2006.373259.

[10] J. Wang, B. Li, Q. Feng, and W. Dou, “A highly scalable butterfly-based photonic network-on-chip,” Proc. - 2012 IEEE 12th Int. Conf. on Computer and Information Technology, pp. 33–37, 2012. DOI: 10.1109/CIT.2012.31.

[11] A. Romanov and A. Ivannikov, “SystemC Language Usage as the Alternative to the HDL and High-level Modeling for NoC Simulation,” Int. J. Embed. Real-Time Commun. Syst., 2018. DOI: 10.4018/IJERTCS.2018070102.

[12] 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. DOI: 10.1109/ISCA.2004.1310774.

[13] N. Jiang, G. Michelogiannakis, D. Becker, and B. Towles, BookSim 2.0 User’s Guide, 2013.

Authorized licensed use limited to: Higher School of Economics. Downloaded on October 06,2021 at 12:56:06 UTC from IEEE Xplore. Restrictions apply.

[14] N. Jiang, Github: BookSim 2.0. [Online]. Available: https://www.cmu.edu/psc/aibd/ neocortex/technical-webinar-post.html

[15] K. A. Jamali, "MinRoot and CMesh: Interconnection architectures for network-on-chip systems," World Acad. Sci. Eng. Technol., vol. 54, pp. 354–359, 2009.

[16] J. Kim, J. Balfour, and W. J. Dally, "Flattened butterfly topology for on-chip networks," IEEE Comput. Archit. Lett., vol. 6, no. 2, pp. 37–40, 2007. DOI: 10.1109/L-CA.2007.10.

[17] T. Majumder, S. Sarkar, P. P. Pande, and A. Kalyanaraman, "NoC-based hardware accelerator for breakpoint phylogeny," IEEE Trans. Comput., vol. 61, no. 6, pp. 857–869, 2012. DOI: 10.1109/TC.2011.100.

[18] D. Patti and F. Fazzino, Github: Noxim - Network on Chip Simulator. [Online]. Available: https://github.com/davidepatti/noxim

[19] N. Myachin and A. Romanov, Newxim. [Online]. Available: https://github.com/Wertual08/newxim

[20] P. Abad, P. Prieto, L. G. Menezo, A. Colaso, V. Puente, and J. Á. Gregorio, "TOPAZ: An open-source interconnection network simulator for chip multiprocessors and supercomputers," Proc. of the 2012 6th IEEE/ACM Int. Symposium on Networks-on-Chip, pp. 99–106, 2012. DOI: 10.1109/NOCS.2012.19.

[21] P. Abad, P. Prieto, L. Menezo, A. Colaso, V. Puente, and J. A. Gregorio, Github: TOPAZ. [Online]. Available: https://github.com/ceunican/tpzsimul

[22] H. Hossain, M. Ahmed, A. Al-Nayeem, T. Z. Islam, and M. M. Akbar, "gpNoCsim – A general purpose simulator for network-on-chip," ICICT 2007: Proc. of Int. Conf. on Information and Communication Technology, pp. 254–257, 2007. DOI: 10.1109/ICICT.2007.375388.

[23] M. A. J. Jamali, H. Bahrbegi, A. A. A. Ahrabi, and M. Bahrbegi, "A study on WK-recursive topology using gpNoCsim++ simulator and comparison to Other topologies," Proc. – 17th IFIP Int. Conf. on Very Large Scale Integration, pp. 181–184, 2009. DOI: 10.1109/VLSISOC.2009.6041351.

[24] A. Yu. Romanov, "High-level software model Universal On-Chip Network Simulator (UOCNS)," RU Certificate of registration of the computer program 2019616754, 2019.

[25] A. Yu. Romanov, E. V. Lezhnev, A. Yu. Glukhikh, and A. A. Amerikanov, "Development of routing algorithms in networks-on-chip based on two-dimensional optimal circulant topologies," Heliyon, vol. 6, no. 1, 2020. DOI: 10.1016/j.heliyon.2020.e03183.

Authorized licensed use limited to: Higher School of Economics. Downloaded on October 06,2021 at 12:56:06 UTC from IEEE Xplore. Restrictions apply.