Modification of the BookSim simulator for modeling networks-on-chip based on two-dimensional circulant topologies

Romanov Aleksandr
PhD, Associate professor
National Research University Higher School of Economics
34 Tallinskaya Ulitsa, 123458, Moscow, Russian Federation
a.romanov@hse.ru

Lezhnev Evgeny
Assistant
National Research University Higher School of Economics
34 Tallinskaya Ulitsa, 123458, Moscow, Russian Federation
elezhnev@hse.ru

Amerikanov Aleksandr
Assistant
National Research University Higher School of Economics
34 Tallinskaya Ulitsa, 123458, Moscow, Russian Federation
aamerikanov@hse.ru

Abstract: In this paper, a review of various high-level models of networks-on-chip, their performance capabilities, and characteristics was carried out. As a result, the BookSim simulator was chosen. Application of circulant topologies for implementation of communication subsystems in networks-on-chip (NoCs) as alternatives to other regular topology was proved. The process of modifying the BookSim simulator described allowed high-level modeling of NoCs with any topologies (including two-dimensional circulants), as well as testing new routing algorithms in such networks.

Keywords: network-on-chip, high-level modeling, BookSim simulator, circulant topology, regular topology.

1 Introduction

Nowadays, the use of multiprocessor systems is becoming an increasingly common solution. The growing complexity of tasks requires that computer systems continually improve performance. In multicore systems with a small amount of cores (2-10 cores), communication between computing cores and other components of a chip occurs successfully with the help of the common bus. The bus is an infrastructure for data transmission between the core blocks (for example, AMBA [1] and Avalon [2]). Generally, only one block of processor can manage data transmission on the bus at one time, while the others can only accept data. Therefore, with an increase in the number of cores, this approach in the organization of their interaction leads to a decrease in the overall speed of the entire system [3]. In order not to significantly change the concept of the block communication subsystem, but at the same time to increase the speed of the entire system, various variants of common bus segmentation were developed [4, 5]. Such an approach complicates the entire system and, ultimately, does not solve the problem as a whole.

Acknowledged and widespread approach is usage of core communication options by creation of NoC [6]. Such network allows division between the chip elements, responsible for data transmission, and the computing units. Each computational unit connects to a network via an interface connected to a router that provides data transmission between units. At the same time the NoC is a separate subsystem and it does not influence the internal organization of a computing unit which can use classical buses to connect its structural elements [7].

2 Network-on-chip topologies

A topical area for research of NoCs is the search for optimal topologies and routing algorithms for them. Classical topologies (mesh [8], torus [9], hypercube [10], spidergon [11]) have significant limitations to their use, since their characteristics deteriorate with an increase in the number of nodes [11]. The use of circulant topologies that have better characteristics than the classical topologies allows creating networks with a large number of nodes (from 100 and more). This is well demonstrated in Figure 1 and Figure 2 which show the dependence of the diameter and the average distance between nodes in hops on the number of nodes in the topology of circulant $C(N; s_1, s_2)$ in comparison with the classical regular topologies. In this case, circulants maintain a regular structure which facilitates routing in them in comparison, for example, with irregular topologies [12].

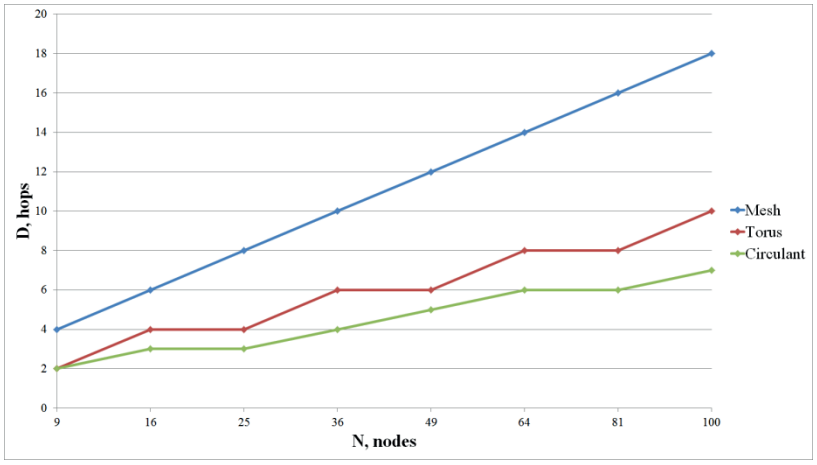

Figure 1 – Dependence of diameter on number of nodes

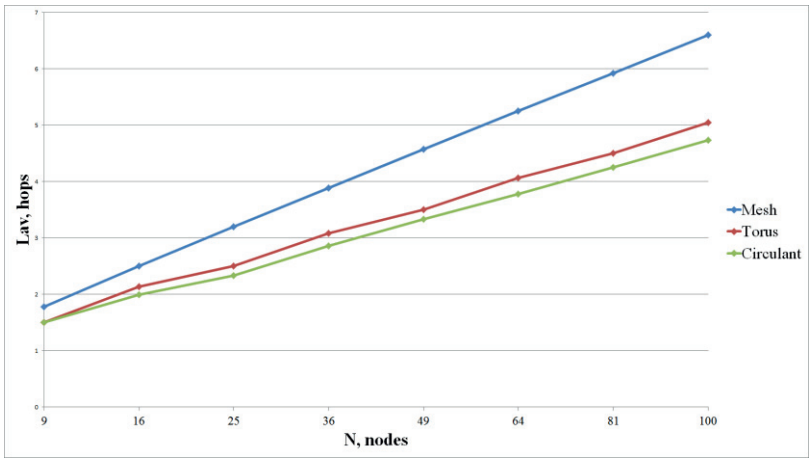

Figure 2 – Dependence of average distance on number of nodes

For the accurate estimates of the effectiveness of using the proposed topologies and routing algorithms in such networks, it is necessary to develop new and modify the existing high-level NoC models which would support new topologies.

3 NoC modeling

Simulation is one of the main stages in NoC design. The purpose of the simulation is to assess the main characteristics of the network: assessment of the consumption of crystal resources, analysis of throughput, energy consumption, network time delays, etc. There are several modeling methods that differ in the level of abstraction:

- High-level – network modeling using high-level languages [13–17];
- Middle-level – network traffic modeling and analysis of bandwidth, packet transfer delays, analysis of the effectiveness of routing algorithms in NoC [18–21];
- Low-level – research of the use of the crystal's resources and the energy consumption of a designed system at the level of prototype, described in HDL [17, 22, 23].

Low-level modeling is simulation modeling during which emulation of the network at the level of logic gates is performed. Model elements are usually described in Verilog/VHDL hardware description languages. Then they analyze

their functioning and synthesis at the RTL level using specialized software (for example, using CAD ModelSim, Quartus Prime, Synopsys IC Compiler, etc.). The low-level modeling allows getting time diagrams of the system as a whole, and after the synthesis – to evaluate the resources used. This approach is applied when it is necessary to simulate as close as possible to the actual operation of the device. For example, in work [24], the VHDL model is used to estimate energy costs, and in work [25], by using various HDL descriptions, various implementations of routers are synthesized to estimate the on-chip area and maximum clock frequency.

An approach, that uses low-level modeling in the context of developing networks on a chip with circulant topologies, was applied in [26]. Several algorithms were proposed for routing in circulant networks of type $C(N; 1, s_2)$.

The first table routing algorithm allows routing based on a previously calculated route table between each network node. The table stores the port numbers to which the packet is to be sent so that it reaches the correct node. The table is stored in a distributed form so that each router stores only one step of the complete route. For the algorithm to work, it is necessary for the head fleet of the packet to store the destination node number. The router here is a digital apparatus which from the head fleet, takes the node number of the destination and uses it as an index to refer to a line of the table stored in the router from which it takes the port's number to send the packet.

The second and third algorithms do not store routing data, but iteratively calculate the number of the next node in the route. The second algorithm calculates the router and the port connected to it as follows: according to certain rules, the direction of transmission of the packet is hourly or counterclockwise after which a transition takes place along a large or small generatrix of the circulant with a mandatory approach to the destination node. The third algorithm is a development of the second algorithm in which, for the sake of simplification, optimal paths between nodes are not always chosen. By introducing additional checks on the choice of the direction and the generatrix, as well as taking into account situations where the optimal route contains a cyclic network traversal, the calculation of the optimal routes in the network was achieved. The third algorithm is a development of the second algorithm in which, due to its simplicity, the optimal route between the nodes were not always calculated. Introducing additional checks on the choice of direction and the generatrix, and also considering that with a certain configuration of the topology, there may be cases when the optimal route contains a cyclical bypass of the network, the calculation of the optimal routes in the network was achieved.

Low-level modeling was performed in the Verilog language [26] with further testing of the operation of the algorithms, as well as the calculation of the occupied chip resources after synthesis. The results obtained allowed us to confirm the hypothesis about the best characteristics of NoC based on circulant topologies, smaller occupied chip resources compared to the use of other topologies. However, in spite of the encouraging results obtained, a higher level simulation is required to analyze the distribution of traffic across networks and the effectiveness of routing algorithms in them.

3.1 High-level NoC simulators

When modeling NoC, an important task is to choose a suitable simulator. A number of simulators were developed; thus, it is possible to carry out studies on NoC parameters. They can be divided into two categories:

- Simulators of general purpose networks – designed to simulate any network as a whole, but can be used for NoC as well: Gem5, Graphite, Sniper, Hornet, Fusionsim, Esesc, Wattch, Hotspot, NS2, NS3, Omnet ++, Netsim, Orion [13, 27, 28, 29].
- Simulators designed specifically for modeling NoC: BookSim, WormSim, Vnoc, Matrics, SICOSYS, Garnet, Ocintsim, Noxim, Nostrum, Nirgam, Occn, Nocsim, Access Noxim, NoCTweak, Atlas, gpNoCsim, Xmulator, Phoenixsim, SUNMAP, NOCMAP, ReliableNoC, MapoNoC [14-17, 30-32].

In general, the structure of NoC simulators can be represented as follows (Figure 3):

1. Setting up the simulator (model initialization, setting limits on simulation time and warm-up time, selection of simulation mode);
2. Setting parameters of synthetic tests (size and structure of the network, traffic profile, location of hot spots of the network, etc.);
3. Selection of parameters for embedded tests (determination of the characteristic graph of the problem and the algorithm for projecting the graph of the problem onto the core);
4. Configuring traffic (number of streams, packet length, packet rate);
5. Configuring the router (type of routers, number and length of packet buffers);
6. Routing configuration (routing algorithm);
7. Selection of measured parameters (network bandwidth, energy consumption, packet transmission delays).

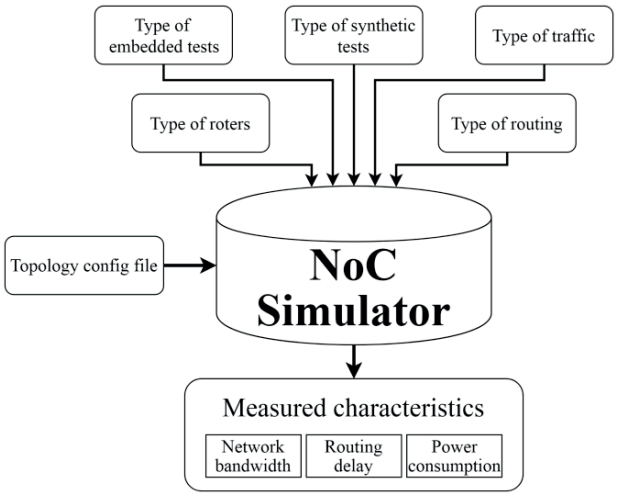

Figure 3 – General structure of NoC simulators

3.2 High-level NoC simulators review

Let's consider a number of the most common NoC simulators.

BookSim [14, 30] is a simulator developed in C++. It allows modeling a large number of different topologies, for example, 2D mesh, 2D torus, folded torus, butterfly fat tree, etc., as well as routing algorithms in them. There is an implemented support for managing the queues of the router and virtual circuits, as well as the event-driven microarchitecture of the router. The simulator also supports resizing buffers, arbitration policies, and synthetic traffic patterns. Using the simulator enables to evaluate a large range of performance parameters, to analyze the packet transmission delays and the dependence of bandwidth on the network load.

Vnoc [31] is a simulator for modeling dynamic voltage and frequency scaling (DVFS) in NoCs. It is developed in C++ and is based on the Orion model. This simulator supports such topologies as 2D-mesh, torus and allows generating various types of traffic. The simulator estimates energy consumption depending on the traffic of the entire network and is not suitable for analyzing the functioning of individual routers, as well as other topologies and specific routing algorithms in the network.

Noxim [32] is a SystemC based simulator that has a command line interface to change input parameters. The simulator is also based on the Orion power model and is currently designed for 2D-mesh topology and allows synthetic tests to be performed to analyze power consumption, throughput, and delays. In the simulator, one can set the network size, packet generation rate, router buffer size, static or adaptive routing algorithm.

Nostrum [33] is a multi-level simulator on SystemC. It supports 2D mesh, torus, ring, tree topologies, XY routing algorithms, and deflection. Nostrum allows projecting a task graph onto network nodes. The user can also change the settings of the router (buffer size) and select the type of traffic.

Nirgam [15] is an open source simulator written in SystemC and designed to carry out a discrete-event simulation of the GCN. The current version of the simulator supports 2D-mesh and torus topologies. The simulator supports routing algorithms Deterministic XY, Adaptive Odd-Even (OE), Source Routing. It is possible to configure the number of buffers, their size, size of the fleet, as well as the frequency of the network. There is also a choice of the type of traffic. The result of the work of the simulator is the data on the average delay of the packets and the bandwidth for each channel.

NoCTweak [16, 17] simulator is developed in SystemC. It is intended for an early study of the performance and energy efficiency of NoC, for example, the transmission capacity, latency, and estimation of the energy consumed by parts of NoC. The current version of the simulator is intended only for 2D-mesh topology. The simulator supports many settings, such as network size, choice of router type, location of hot spots, packet generation frequency, packet switching strategy, CMOS process selection, chip frequency, and voltage selection. NoCTweak supports both embedded tests and synthetic tests.

NOCMAP [34] is an open source C++ simulator. This simulator works only with 2D-mesh topology. This tool implements two algorithms for displaying the characteristic graph of the problem on the network topology, such as BB and SA. It uses the bit energy model to calculate the minimum total binding energy of the GCN. The BB algorithm is used for the topological allocation of network IP nodes to minimize the total consumption of communication energy. Based on the EPAM bit energy model (XY, OE and WF), which is a comparison of energy and performance with a different routing algorithm (XY, OE and west-first), this simulator implements an efficient BB algorithm. For comparison, the SA algorithm is also implemented which shows that the proposed algorithms are more efficient than SA with respect to the optimal result and simulation speed. ReliableNoC [35] is an enhanced version of the NOCMAP simulator that adds reliability parameters to the NOCMAP simulator.

Thus, on the basis of the in-depth analysis of the considered NoC simulators (Table 1), we can conclude that 2D-mesh topology is the prevailing topology in most common high-level NoC models. At the same time, the topology is often the basis of the program code, all the basic model calculations and its structure of classes and interactions between them depend

on it in an implicit form, i.e. a change in topology entails the reworking of virtually the entire source code of the model. This significantly limits the capabilities of the models.

Table 1 – Comparison of most popular simulators of networks on chip

Simulator	Language	Support topologies	Tests (synthetic/ embedded)	Routing algorithm	Hot spot	Performance parameters	Input parameters
BookSim [14]	C++	2D-mesh torus fat tree dragonfly kn-cube et al.	yes/no	Deterministic dimension-order XY (and variations) Nearest common ancestor Ran-min et al.	No	Delays Bandwidth	File
Vnoc [30]	C++	2D-mesh	yes/no	XY	No	Energy consumption	Command line
Noxim [31]	SystemC	2D-mesh	yes/no	XY Negative first West first North last OE Dyad Fully adaptive Lookup table	Yes	Delays Energy consumption Bandwidth	Command line
Nostrum [32]	SystemC	2D-mesh torus	yes/no	XY Deflection routing	No	Bandwidth capability Delays Use of links	Command line
Nirgam [15]	SystemC	2D-mesh torus	yes/yes	XY OE	No	Delays Bandwidth capability Capacity	File
NoCTweak [16]	SystemC	2D-mesh	yes/yes	XY Negative first West first North last OE Lookup table	Yes	Bandwidth capability Delays Energy consumption	Command line
NOCMAP [17]	C++	2D-mesh	yes/yes	XY West first OE Dyad	No	Energy consumption Durability	Command line

Among the NoC simulators considered, BookSim [14, 30] supports the largest number of topologies and routing algorithms and also has the capabilities to customize the modeling process, traffic, packet distribution, support for virtual channels, etc. The support of a large number of topologies required the developers to implement a clearer model structure and standardize the interfaces of interaction between individual modules. This, as well as the fact that the model is open source, suggests that this simulator can be the basis for further modification.

4 BookSim NoC simulator

BookSim [14, 30] is a console application for modeling traffic for various topologies. This simulator is initially focused on running on UNIX-like systems. The source code is broken down into separate modules, which makes it convenient to modify and add new functionality. The structure of modules of the simulator is shown in Figure 4.

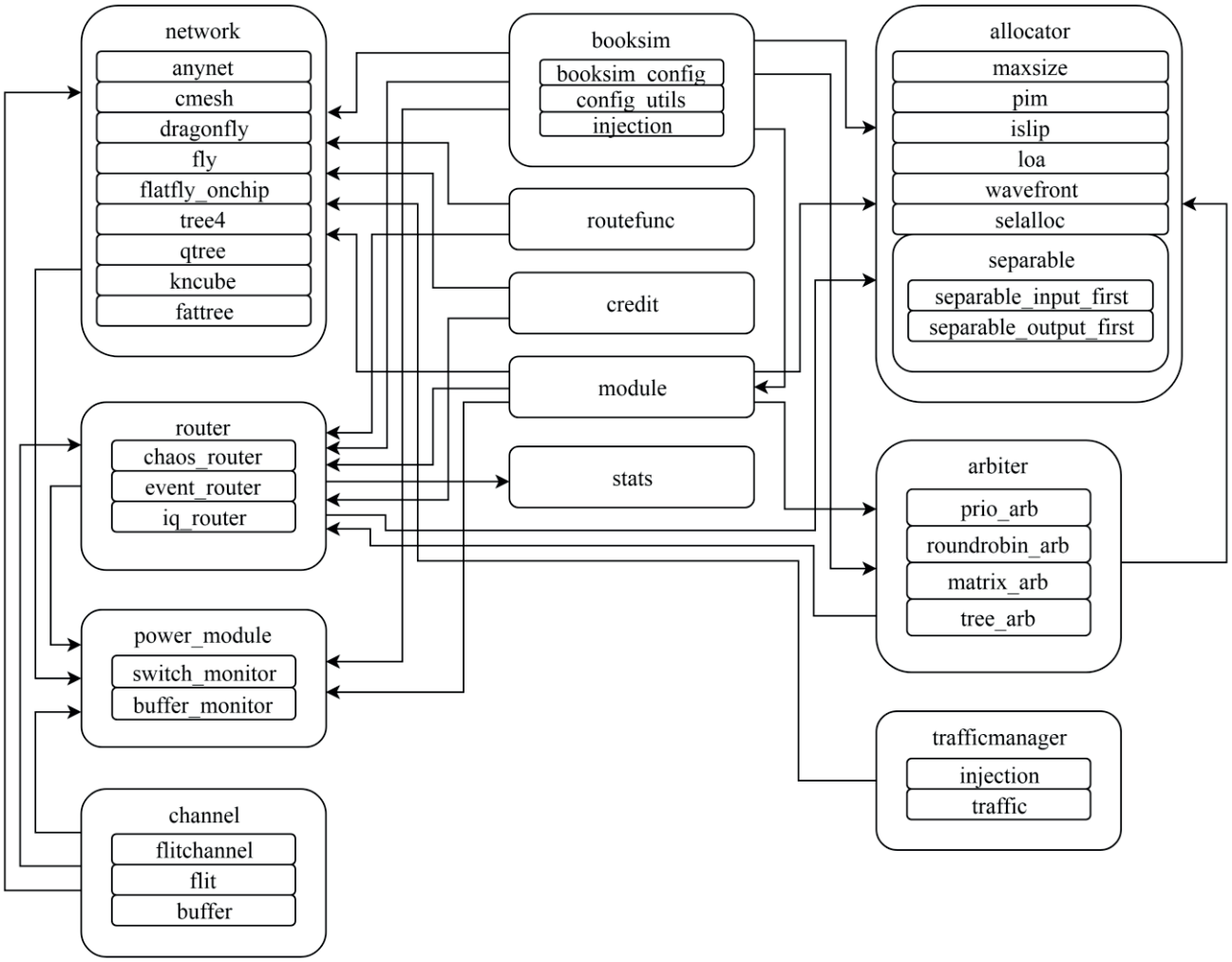

Figure 4 – Class structure of BookSim simulator

The simulator can be divided into 7 modules: a module describing topologies, a module describing routers, a module describing allocators, a module for arbitrating access to router resources, a module describing traffic, a module for analyzing power consumption. A module is a set of classes inherited from a general class. This is done to simplify modification of the simulator.

The topology description module contains a base class in which variables common for topologies, such as the number of nodes, the size of the network, the number of channels, are then redefined when forming the topology. The module also contains general functions that do not depend on the selected topology, for example, creating allocators and reading / writing data to channels. It calls the topology class specified in the configuration file to calculate specific topology parameters. Currently, developers support 10 topologies: mesh [8], torus [9], cmesh [36], fly [37], qtree [38], tree4, fat tree [39], flat fly [40], AnyNet, DragonFly [41].

The module of description of routers contains a list of channels and other variables for the router operation, and functions for adding input and output channels. One can use 3 types of routers: input-queue router, event-driven router, and chaos router.

The module of description of the allocators contains a description of the methods of distributing the resources of routers among their customers, each of which can use one or several resources. Multiple type blockers are supported: iSLIP separable allocator, parallel iterative matching (pim), separable output-first allocator, lonely output allocator (loa), wavefront allocator, separable input-first allocator.

The arbitration description module contains algorithms for distributing access to router resources. The following arbitration algorithms are possible: round-robin, matrix, tree, priority.

The traffic description module controls how the packets will go to the network, how fast and which nodes will generate the traffic. It is possible to set uniform generation of traffic by all nodes, according to a specific algorithm, as well as random generation.

The power consumption analysis module allows analyzing energy consumption of the network being modeled depending on the specified parameters of the chip production processes.

All these modules are united by the main module to the input of which a configuration file with a description of the simulated network arrives. It directly creates and configures the network, starts the simulation, selects a routing algorithm, and collects statistics.

Configuring the simulator is made through the configuration file. This definitely structured file contains the definition of all the necessary parameters for the simulator operation [14]:

1. Name of the topology to be tested, and also (depending on the selected topology) its parameters (number of nodes, type of topology, its dimension, number of routers in the network);
2. Description of number of channels, width of the channel buffers, number of virtual channels;
3. Adjusting microarchitecture of the router (algorithm for distributing packets across channels, as well as algorithm for distributing the load on routers);
4. Description of traffic (its structure, size, generation algorithm, and rate of packets entering the network in one cycle);
5. General settings (number of simulation cycles, duration of one cycle and number of cycles to heat the network).

BookSim conducts a simulation in three phases: warming up, measurement, analysis. The duration of these phases depends on the parameters set in the configuration file. Delay and throughput are displayed after each phase, and at the end, the average for these values is calculated. After the warm-up phase is completed, a message is displayed, and the simulator moves to the next measurement phase, reconfiguring all the parameters. The statistical data of the simulated topology is displayed before each next phase of the simulation. Upon completion of the simulation, BookSim generates an output file in which all simulation results are recorded.

5 BookSim simulator modification

In order to use BookSim to model circulant topologies, it was necessary to make certain changes to its source code. To begin with, a description of the topology was created. All structures, in which topologies supported by the simulator are described, are built on the same principle: there is a class in which the main characteristics of the topology are stored. Also, the topology class has a set of functions that, when called, will calculate the size of the network, calculate the total number of physical channels, and configure individual network routers. The class also contains functions for building a network and creating links between routers.

All classes describing topologies are descendants of the general *Network* class which contains general information about the network (number of nodes, number of channels, number of network cycles, etc.). It is used to configure parameters of the virtual channels and the mechanism for confirming transfers using credits, as well as selecting the necessary class for implementing the topology described in the configuration file and registering the routing algorithm for the topology. At this layer of the model, there is a choice of algorithms for distributing traffic over the virtual channels of routers. An instance of the *Network* class controls reading and writing of flits to ports defined by the routing algorithm and controls reading and writing of credits.

Thus, the Network class is an intermediary between the topology classes and the rest of the simulator logic. This is an important advantage of the BookSim simulator, since, unlike many other simulators, the interface for connecting new NoC topologies is implemented.

Thus, the first thing done was the creation of a new class called *Circulant* which stores the number of network nodes and the list of generators. Also, in the developed class, functions for calculating the size of the circulant network and number of channels calculating the numbers of the nearest nodes (with the distance to the selected node equal to one hop) were implemented. Next, changes were made to *Network* class by adding connection condition of the developed new topology class, when it is mentioned in the configuration file.

BookSim supports modeling of various topologies among which there are many variations of mesh and torus topologies. For most of these variations, one can use the same routing algorithms. Therefore, instead of setting up a routing algorithm in each class, the developers created a separate *Routerfunc* class which collects all the routing algorithms for all possible topology classes as functions. To implement the selection of the required routing algorithm, *gRoutingFunctionMap*, container is used; it stores references to the routing algorithms. In the configuration file, in the part of the topology setting, the name of selected algorithm is specified, and then when setting up the routers, it is selected. In this file, the algorithms, given in [26] for routing in circulant topologies, were specified and added to the *gRoutingFunctionMap* container.

Since the definition of settings of the simulator occurs through the configuration file, the data is read from it in a certain way and the corresponding variables are initialized. In order for the simulator to correctly read the configuration file to simulate the circulant topology, changes were made to the class that implements processing of the read data from the configuration file *booksim_config.* These changes might not be made; however, in this case, the possibility of describing the generatrices in the form of a string, separated by a comma, is lost, which complicates the use of the simulator to simulate circulant topologies with a large number of generatrices. At the same time, the classes, in which the configuration file parser was implemented, did not need to be changed, since it simply read lines and distributes data from them into several associative arrays. The read string with the generatrices is stored in a string variable and is further divided into an array of numbers. An example of a configuration file for circulant network modeling is presented in Figure 5.

```
// Topology
topology = circulant;
k = 100;
s = 1, 18;

warmup_period = 3;
sim_count = 1;
sample_period = 1000;

// Routing
routing_function = simple;
// Flow control
num_vcs = 2;
// Traffic
traffic = uniform;
injection_rate = 0.15
```

Figure 5 – Example of configuration file to circulant topology

6 Approbation

To test the operation of the modified simulator, modeling and comparison of networks with mesh, torus, and circulant topologies was performed. The optimal shapes of the mesh, torus, and circulant topologies were chosen for testing in order to eliminate the influence of the geometric shape of the topologies on the result. For the circulant topology, restrictions were imposed on the choice of the number of nodes in the network. They are calculated by the formula $N = n^2$, where N – number of nodes in the network, n – natural number. Thus, mesh(10x10), torus(10x10), and circulant $C(100; 1, 18)$ networks were simulated. The topology $C(100; 1, 18)$, obtained as a result of synthesis using the generatrix proposed in [42], is optimal and characterized by the best values of diameter and average distance between nodes for circulants with 100 nodes [43]. The comparison was carried out on the average rate of transmission and reception of fleets when changing the rate of generation of packets. The simulation was carried out with the following model parameters:

- Packet length – 10 flits;
- Warming up period – 3;
- Sampling period – 1000;
- Number of virtual channels – 8;
- Traffic type – uniform;
- Rate of traffic generation – from 0.05 to 1.0.

The simulation results are shown in Figures 6 and 7.

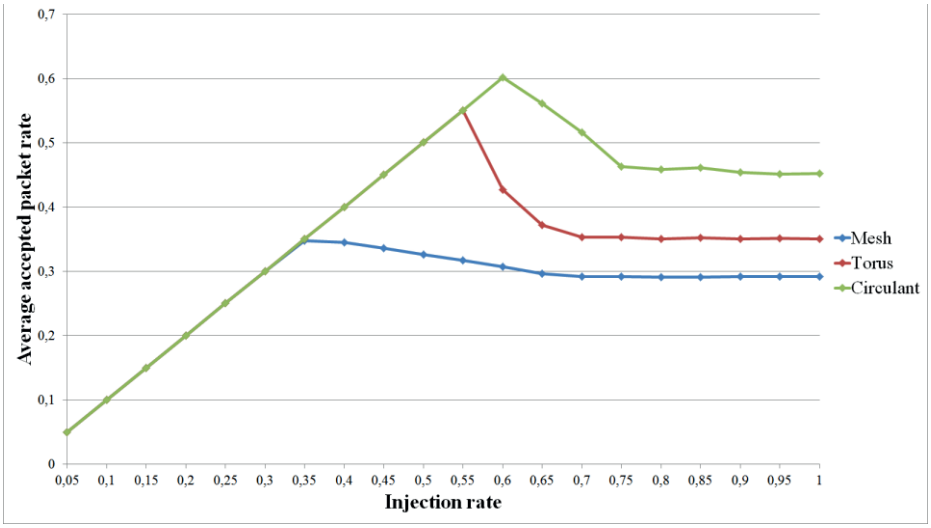

Figure 6 – Dependence of average accepted packet rate on injection rate

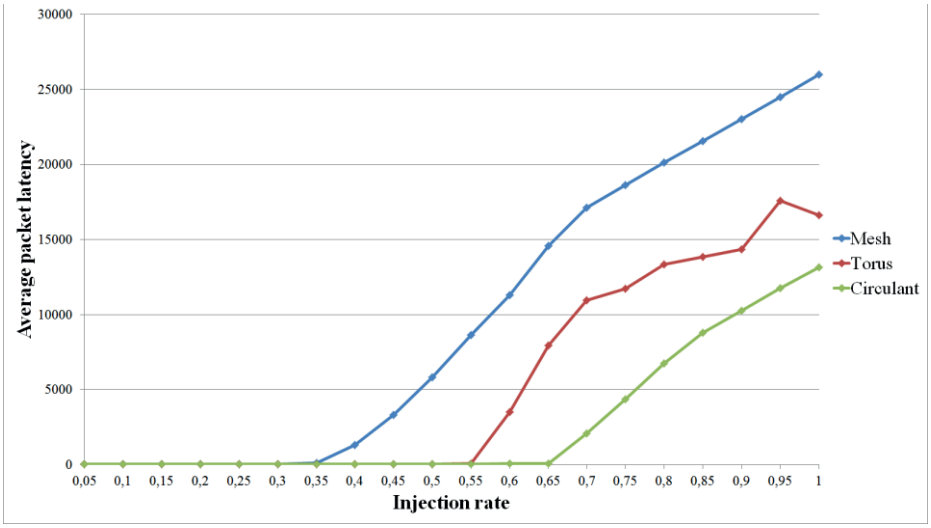

Figure 7 – Dependence of average packet latency on injection rate

The results obtained demonstrate that with the selected topology configuration, the throughput of mesh(10x10) topology stabilized at 0.3; torus(10x10) – at 0.35; circulant $C(100;\ 1, 18)$ – at 0.55. Also, the saturation of NoC with the circulant topology occurs later than in mesh and torus topologies. These results confirm the previously obtained results [26] and demonstrate high efficiency of using circulant topologies as the basis of NoCs.

7 Conclusion

Thus, a review of high-level NoC simulators was conducted. It showed that most high-level simulators support the implementation of NoC modeling only with mesh and torus topologies. Meanwhile, it is the BookSim simulator which is capable to carry out topology simulations not only of mesh and torus, but also their modifications, as well as other topologies. It was the key feature for choosing a simulator for further modification. The facts of the BookSim (it is an open source code, it is written in C++, and it can simulate various characteristics of networks) confirmed the correctness of the choice of this simulator. The changes, proposed to the structure of simulator, ensured development of a class describing the circulant topology; routing algorithms, allowed simulation of NoC with two-dimensional circulant topologies and a comparative analysis of the results of their modeling with other topologies, were also added. To test the operation of the modified simulator, networks with mesh, torus and two-dimensional circulant networks were simulated, and data were obtained on the delays of receiving and transmitting fleets in the network. This makes it possible to conduct high-precision modeling of NoCs with high-level circulant topologies.

Acknowledgments

The research leading to these results has received funding from the Basic Research Program at the National Research University Higher School of Economics.

References

[1] AMBA SPECIFICATIONS. https://www.arm.com/products/system-ip/amba-specifications.

[2] Avalon Bus Specification Reference Manual, 2002. http://coen.boisestate.edu/clarenceplantingfiles/2011/09/avalon_bus_spec.pdf.

[3] G. Nychis, C. Fallin, T. Moscibroda, O. Mutlu. Next Generation On-Chip Networks: What Kind of Congestion Control Do We Need? Hotnets-IX: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, ACM, article 12, October 2010.

[4] T. Seceleanu. Communication on a Segmented Bus Platform. Proc. of the IEEE International SOC Conference, 205-208, October 2004.

[5] J. Y. Chen, W. B. Jone, J. S. Wang, H. Lu, T. F. Chen. Segmented bus design for low-power systems. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 7(1): 25–29, March 1999.

[6] A. Jantsch, H. Tenhunen. Networks on Chip. Kluwer Academic Publishers, 2003.

[7] W. J. Dally, B. Towles. Principles and practices of interconnection networks. Elsevier, 550, December 2003.

[8] D. Deb, J. Jose, S. Das. Cost effective routing techniques in 2D mesh NoC using on-chip transmission lines. Journal of Parallel and Distributed Computing, 123: 118-129, January 2019.

[9] A. Q. Ansari, M. R. Ansari, M. A. Khan. Modified quadrant-based routing algorithm for 3D Torus Network-on-Chip architecture. Perspectives in Science, 8: 718-721, September 2016.

[10] M. B. Marvasti, T. H. Szymanski. The performance of hypermesh NoCs in FPGAs. IEEE 30th International Conference on Computer Design (ICCD), 492-493, December 2012.

[11] R. Bishnoi, P. Kumar, V. Laxmi. Distributed adaptive routing for spidergon NoC. 18th International Symposium on VLSI Design and Test (IEEE), 1-6, August 2014.

[12] A. Y. Romanov, I. I. Romanova. Use of irregular topologies for the synthesis of networks-on-chip. IEEE 35th International Conference on Electronics and Nanotechnology (ELNANO), 445–449, July 2015.

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

[14] N. Jiang, G. Michelogiannakis, D. Becker, B. Towles W. J. Dally. BookSim 2.0 User's Guide, March 2013.

[15] L. Jain. NIRGAM: A Simulator for NoC Interconnect Routing and Application Modeling. Design Automation and Test in Europe (DATE), 1-2, September 2007.

[16] A. T. Tran, B. M. Baas. NoCTweak: a Highly Parameterizable Simulator for Early Exploration of Performance and Energy of Networks On-Chip. Technical Report, VLSI Computation Lab, ECE Department, 12: July 2012.

[17] A. Romanov, A. Ivannikov. SystemC Language Usage as the Alternative to the HDL and High-level Modeling for NoC Simulation. International Journal of Embedded and Real-Time Communication Systems (IJERTCS), 9(2): 18-31, 2018.

[18] N. Genko, D. Atienza, L. Benini. Feature – Noc Emulation: A Tool and Design Flow for MPSoC. IEEE Circuits and Systems Magazine, 7(4): 42–51, January 2008.

[19] D. Bertozzi, A. Jalabert, L. Benini. NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-Chip. IEEE Transactions on Parallel and Distributed Systems, 16(2): 113–129, February 2005.

[20] K. Goossens, J. Dielissen. The Aethereal Network on Chip: Concepts, Architectures and Implementations. IEEE Design and Test of Computers, 22(5): 414–421, October 2005.

[21] D. Siguenza-Tortosa, J. Nurmi. VHDL-based simulation environment for Proteo NoC. Proceedings of the Seventh IEEE International High-Level Design Validation and Test Workshop, 1–6, October 2002.

[22] R. G. Mota, J. Silveira, et al. Efficient routing table minimization for fault-tolerant irregular Network-on-Chip. IEEE International Conference on Electronics, Circuits and Systems (ICECS), 632–635, 2016.

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

[24] J. C. S. Palma, C. A. M. Marcon, F. G. Moraes. Mapping embedded systems onto NoCs: the traffic effect on dynamic energy estimation. Proceedings of the 18th annual symposium on Integrated circuits and system design (SBCCI'05), 196–201, September 2005.

[25] K. Goossens, J. Diellisen, A. Radulescu. AEthereal network on chip: concepts, architectures, and. IEEE Design & Test of Computers, 22(5): 414–421, September 2005.

[26] A. Yu. Romanov. Development of routing algorithms in networks-on-chip based on ring circulant topologies. Heliyon, 5(4): article e01516, April 2019.

[27] Y. Ben-Itzhak, E. Zahavi, I. Cidon. NoCs simulation framework for OMNeT++. Fifth IEEE/ACM Int. Symp. Networks on Chip (NoCS), 265–266, May 2011.

[28] H. Wang, X. Zhu, L. Peh. Orion: a power-performance simulator for interconnection networks. Microarchitecture (MICRO-35). Proceedings. 35th Annual IEEE/ACM International Symposium on. – IEEE, 294-305, November 2002.

[29] The Network Simulator - ns-2. https://www.isi.edu/nsnam/ns/.

[30] N. Jiang, D. U. Becker, G. Michelogiannakis, et al. A detailed and flexible cycle-accurate Network-on-Chip simulator. IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 86–96, 2013.

[31] H. Phan, X. Tran, T. Yoneda. Power consumption estimation using VNOC2.0 simulator for a fuzzy-logic based low power Network-on-Chip. IEEE International Conference on IC Design and Technology (ICICDT), 1–4, May 2017.

[32] V. Catania, A. Mineo, S. Monteleone, M. Palesi, D. Patti. Cycle-accurate network on chip simulation with noxim. ACM Transactions on Modeling and Computer Simulation (TOMACS), 27(1): article 4, November 2016.

[33] Z. Lu. NNSE: Nostrum network-on-chip simulation environment. Swedish system-on-chip conference, March 2005.

[34] J. Hu, R. Marculescu. Energy-aware mapping for tile-based NoC architectures under performance constraints. Proceedings of the ASP-DAC Asia and South Pacific Design Automation Conference, 233–239, April 2003.

[35] C. Ababei, H. S. Kia, O. P. Yadav, J. Hu. Energy and reliability oriented mapping for regular networks-on-chip. Proceedings of the Fifth ACM/IEEE International Symposium, 121–128, May 2011.

[36] H. J. Kim, J. T. Seo, T. H. Han. 3CEO: Three dimensional Cmesh based electrical-optical router for networks-on-chip. ICTC 2011, 114–119, 2011.

[37] P. Yebenes, J. Escudero-Sahuquillo, P. J. Garcia, et al. An Effective Queuing Scheme to Provide Slim Fly Topologies with HoL Blocking Reduction and Deadlock Freedom for Minimal-Path Routing. IEEE 3rd International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB), 25–32, 2017.

[38] K. Hose, M. Karnstedt, A. Koch, et al. Processing Rank-Aware Queries in P2P Systems. Databases, Information Systems, and Peer-to-Peer Computing: Proceedings of the 2005/2006 international conference on Databases, information systems, and peer-to-peer computing, 171–178, August 2005.

[39] H. Ghalwash, C. H. Huang. QOS for SDN-based fat-tree networks. Future of Information and Communication Conference FICC 2019: Advances in Information and Communication, 691–705, 2019.

[40] J. Kim, W. J. Dally, D. Abts. Flattened butterfly: a cost-efficient topology for high-radix networks. 34th International Symposium on Computer Architecture (ISCA 2007), 126–137, 2007.

[41] M. Benito, P. Fuentes, E. Vallejo, R. Beivide. ACOR: Adaptive congestion-oblivious routing in dragonfly networks. Journal of Parallel and Distributed Computing, 131: 173–188, September 2019.

[42] A. Y., Romanov, I. I. Romanova, A. Y. Glukhikh. Development of a Universal Adaptive Fast Algorithm for the Synthesis of Circulant Topologies for Networks-on-Chip Implementations. IEEE 38th International Conference on Electronics and Nanotechnology (ELNANO),110–115, 2018.

[43] Optimal Circulants Dataset, available at: https://github.com/RomeoMe5/circulantGraphs/ (accessed: 21.03.2019).