IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 15 March 2024, accepted 25 March 2024, date of publication 2 April 2024, date of current version 9 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3382710

RESEARCH ARTICLE

Electronic Computer-Aided Design for Low-Level Modeling of Networks-on-Chip

EVGENY V. LEZHNEV, (Member, IEEE), VLADIMIR V. ZUNIN, (Member, IEEE), ALEKSANDR A. AMERIKANOV, (Member, IEEE), AND ALEKSANDR Y. ROMANOV, (Member, IEEE)
HSE University, 101000 Moscow, Russia

Corresponding author: Aleksandr Y. Romanov (a.romanov@hse.ru)

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).

ABSTRACT This article proposes a Network-on-Chip (NoC) communication subsystem model on the basis of which the Electronic Computer-Aided Design (ECAD) architecture in the form of software is implemented. It makes it possible to automate the process of preparing and generating an HDL description of the NoC model in the Verilog language. It is shown that not in all cases it is required to model the entire NoC. Often, it is necessary to model its certain parts, such as a communication subsystem, routing algorithm, and traffic control system. The developed model allows modeling a parameterized NoC communication subsystem to obtain an estimate of the consumed logical blocks and registers required for prototyping the communication subsystem. All components of the communication subsystem are implemented as separate modules due to which the hardware costs for adding the necessary components for the study are reduced because of the absence of the need to completely rework the model program code every time. The effectiveness of using the developed ECAD and low-level modeling automation methods to study the work of routing algorithms for NoC topologies is demonstrated.

INDEX TERMS Electronic computer-aided design (ECAD), HDL, network-on-chip (NoC), modeling, RTL.

I. INTRODUCTION

The constant increase in the complexity of computational problems and in the amount of data required for calculations are becoming important factors influencing the industry of developing computer systems [1]. This is reflected in the transition from single-core to multi-core processors, as well as from single-processor computing systems to multiprocessor ones. In recent years, the development of processors and Systems-on-Chip (SoC) and Multi-Processor System-on-Chip (MPSoC) [2] has reached a new qualitative level, which is a combination of a large number of processors on one chip (from 100 or more) and connecting them into a single Network-on-Chip (NoC) [3]. This requires the development of new approaches to the design of MPSoCs.

Thus, the NoC development is an important scientific and technical task. Although this direction itself has appeared relatively recently, a significant number of publications on this topic and their representation in highly cited international journals (Figure 1) indicates the high attention of the scientific community and engineers to this topic; it is quite evident that many problems in this field have not been solved yet.

Further, in section II, we will look at the basic definitions and NoC structure. In Section III, we consider a comprehensive approach to NoC modeling. The main results and low-level modeling are described in Section IV. Automation of NoC modeling is described in Section V. Evaluation of the developed Electronic Computer-Aided Design (ECAD) on the real problems is given in Section VI. Finally, Section VII summarizes and briefly describes the results obtained.

The main contributions of this research are as follows: first, we examined in detail the NoC design flow as a type of VLSI design flow and showed the features associated with the fact that the NoC is a very complex development object; we described the problems and their solutions, as well as the place of our work in the general theory of NoCs;

The associate editor coordinating the review of this manuscript and approving it for publication was Fabian Khateb.

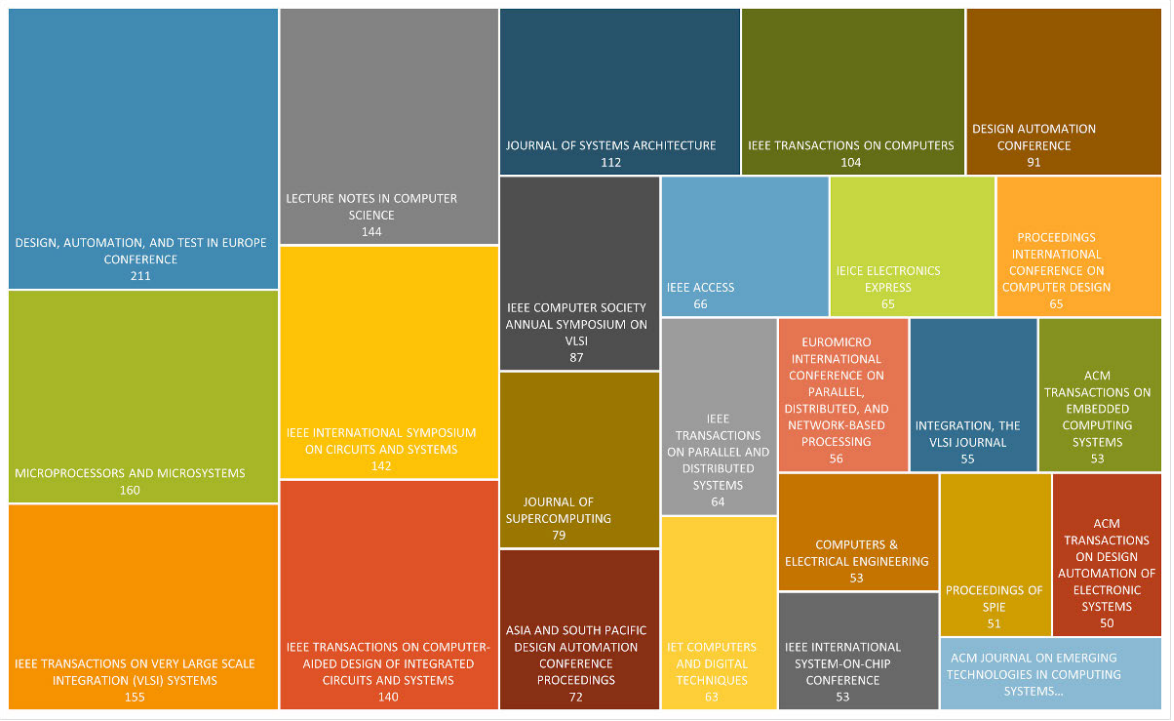

FIGURE 1. Distribution of publications with the keyword "networks-on-chip" in the journals indexed in WoS over the past 10 years.

second, we proposed separate NoC modeling approach, which allows reducing the complexity of the modeling process by separately modeling individual parts of the NoC; third, we implemented a low-level model of a NoC communication subsystem and proposed an architecture of ECAD for low-level NoC modeling based on it; fourth, we described how the interface between high-level and low-level models should be implemented within a single ECAD; fifth, we described the means of automating the low-level NoC modeling process and advantages it gives; finally, we briefly described successful cases of using the developed ECAD and its areas of application.

II. BACKGROUND

A. NoC STRUCTURE

NoC is a complex system with an ultimately non-deterministic structure. Depending on the tasks performed, NoC can contain a large number of different components in its architecture. At the same time, there are basic elements without which NoC cannot be implemented. The basic NoC structure [3], [4] is shown in Figure 2.

The structure of the NoC necessarily includes the following components: computing IP blocks (IP), routers (R), and a topology (T), which determines how the network components are interconnected [5].

A router is connected to each computing IP block through a special communication interface (CI). The IP block generates data and passes it to the router, which ensures that the data is correctly transmitted to the destination IP block. The router is essentially a state machine whose structure is shown in Figure 3 [6], [7].

The router has a set of input and output ports, which (depending on the implementation) can be not only physical but also have virtual channels. Based on the routing algorithm embedded in the router [8], [9], it forms the received data from the computing IP block into data packets. Each data packet is supplemented with auxiliary information required for the correct operation of the algorithm (in the general case, it is destination address). Then the output port of the router to which the data packet must be redirected so that it follows the calculated path is determined. In this way, the connection of the input and output ports in the router is managed.

All the routers are interconnected by links that implement the network topology. In the general case, the NoC topology is an undirected connected graph consisting of vertices: routers and edges – physical communication lines [4].

The smaller the number of vertices, the lower the resource costs; the smaller the average distance between nodes and the diameter of the graph, the faster the packets reach the destination node [10].

B. NoC MODEL AND STAGES OF ITS DEVELOPMENT

For data transmission networks, which include Ethernet networks, wireless local area networks (WLANs), and other similar systems, the OSI model [11] is used to generalize the representation of network components. There are various levels and protocols of system interaction: from the hardware implementation level technologies to program levels

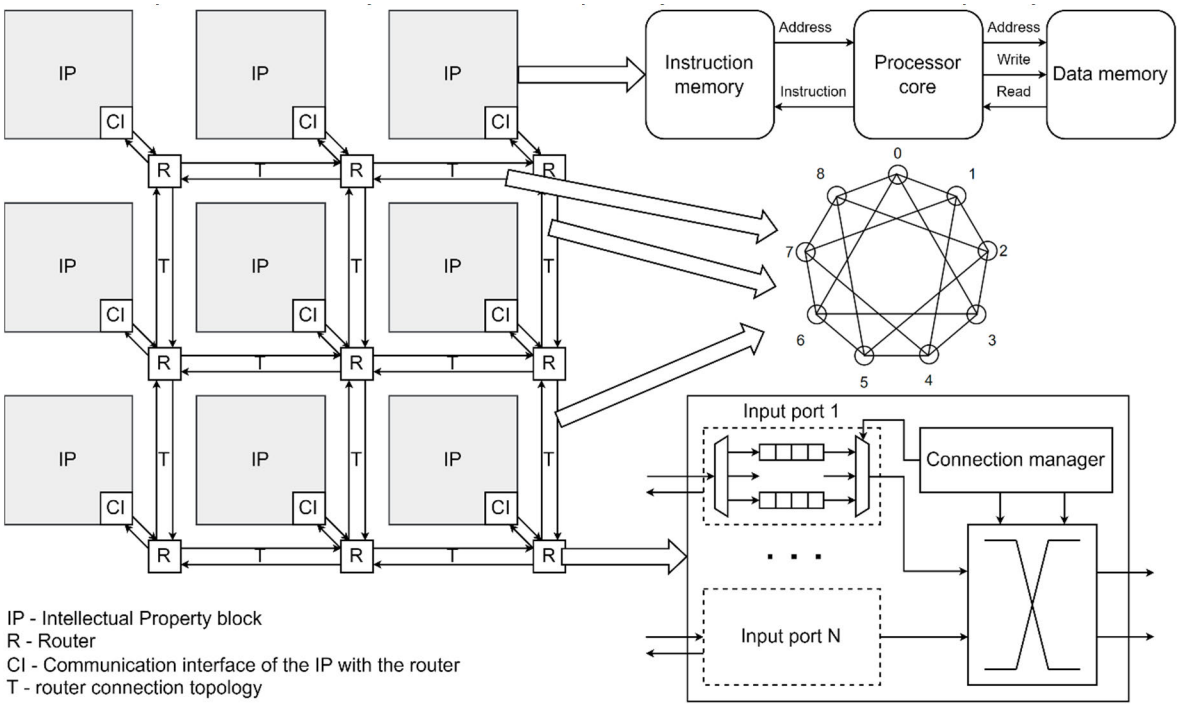

FIGURE 2. Basic structure of the NoC.

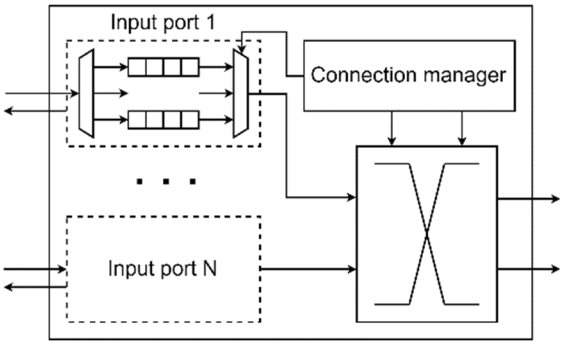

FIGURE 3. General router structure in a NoC.

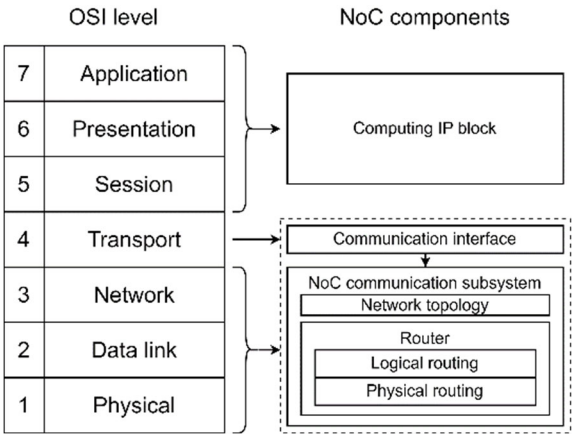

FIGURE 4. OSI model in a NoC.

of the description of protocols and data structures. As NoC is conceptually a data network implemented inside the chip to organize data transfer between IP blocks and other NoC components, it is possible to compare the NoC structure and the tasks solved by its components with the OSI network model (Figure 4) [12], [13].

The application level, presentation level, and session level are implemented by an IP computational block that generates data for transmission over the network and indicates the destination IP block where this data should be transferred. Unlike classical data transmission networks (where, in accordance with the OSI model, the tasks of the transport level include ensuring reliable transmission of data packets between network elements and the implementation of data transmission protocols [14], [15]), in a NoC, the transport level ensures the coordination of data structures between the computing IP block and communication subsystem.

The remaining levels of the OSI model are implemented by the NoC communication subsystem, which receives data from the IP block, converts it into a format suitable for transmission between the routers in a network with a predetermined topology, transmits it along the route calculated by the router, and (upon reaching the target node) performs the reverse transformation for transfer to the IP block.

C. NoC DESIGN FLOW

In the NoC development industry, a certain sequence of design phases has developed [16] (Figure 5).

Traditionally, when designing complex devices, a top-down design model is used [17], [18], [19]. With this approach, the project goes through various phases the result

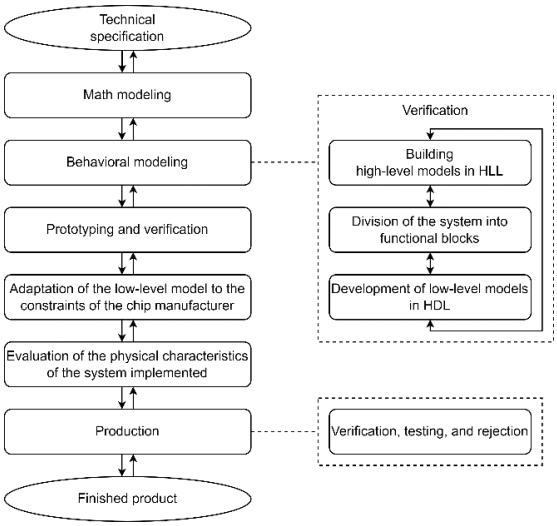

FIGURE 5. Phases of the NoC design flow.

of which is an increasingly detailed representation of the finished product.

The NoC design begins with the development of technical specification, which generally specifies the requirements for the NoC – without taking into account how and from what elements of the base components the NoC will be implemented.

As a result of the analysis of requirements from the specifications, the development of models of the designed system is carried out in order to obtain a preliminary description of the NoC; to determine the set of components it should consist of and clear up the possibility of such an implementation. It should be noted that despite the wide range of tools and modeling methods when designing a NoC, the task of modeling is complicated by the fact that it is necessary not only to develop a NoC model, but also to create the models for testing, as well as to prepare the test data to check the correctness of the model.

The modeling phase is divided into 2 blocks: mathematical modeling and behavioral modeling [20]. With the help of mathematical modeling, the system developed is described in an analytical form using mathematical formulas and terms. Mathematical models (with the help of logical and mathematical constructions) describe the properties of the developed system, its parameters, connection of components, dependence of data transformation. In the general case, a mathematical model is a set of differential equations that completely describe the system.

Behavioral modeling, unlike mathematical modeling, describes the functioning of the system as a set of algorithms. Algorithms represent the behavior of the real elements of the system and reproduce its logical structure. Depending on the simulation results, behavioral models can be implemented in both High-Level Language (HLL) and Hardware Description Language (HDL). As a result of passing through all phases of modeling, a complete description is formed, and a complete functional verification of the technical requirements for the NoC developed is completed.

The next step in the NoC design flow is prototyping and system verification. At this phase, the description of a NoC is carried out using Verilog or VHDL hardware description languages at the register transfer level (RTL). Usually, a NoC prototype is created from its low-level model. All functionality of a NoC is checked using functional verification for compliance with the technical specifications using reference ideal models implemented as test benches or high-level models (in C, SystemC, etc.). In addition to functional verification, timing verification which takes into account signal propagation delays in interconnections is performed.

The task of the remaining phases of the NoC design flow is to prepare the prototype for production. The chip layout is optimized, and library elements are placed that correspond to the production technology based on which the final device will be implemented. A timing verification of the system is re-performed to verify the performance of the production-adapted system. Then the prepared device descriptions in GDSII format can be transferred to production and testing of the chips manufactured.

It is worth noting that modeling is the most important phase in the NoC design because the success of the remaining phases directly depends on how the system was modeled. In the event of a simulation error, the results of all the following phases are discarded as unsatisfactory to the technical specifications, and the simulation is performed again. At the same time, it is at the phase of behavioral modeling that a description of the system being developed is formed, and its components and their characteristics are determined [21].

D. NoC MODELING

As previously shown, when designing a NoC, a separate important task is the development of its communication subsystem. Designing a NoC communication subsystem in general consists in determining many different characteristics of the following elements [22]: connection topology, routing algorithm, router structure, methods of traffic flow control in the network. The general structure of a NoC also includes the following components [4]: computing nodes and external peripherals; however, their influence on the network operation at the initial stage can be neglected because they do not affect the communication subsystem but only generate data that the communication subsystem must transmit. The connection of NoC IP blocks with its communication subsystem occurs only at the level of network routers.

To analyze the impact of the adopted architectural decisions on the performance of the NoC designed, it is required to conduct modeling. Depending on the type of data to be obtained, various types of modeling can be applied at this phase [20]: mathematical, behavioral. Compared to high-level modeling, the low-level modeling phase is one of the most labor-intensive in design, both in terms of describing modules and obtaining simulation results (which is much more compared to high level modeling). Model preparation usually consists in the development of a large amount of the

same type of program code in hardware description languages during which the behavior of NoC components is described, and various model parameters are set. Due to limited FPGA resources, it is not always possible to prototype a NoC with the required number of nodes. Dividing the original model into several models of less complexity that explore the individual components of a NoC can significantly reduce the time to obtain modeling results. The use of the specialized ECAD for the NoC design and modeling can significantly reduce the resource costs. At the same time, such an ECAD can automate both the entire development cycle and its phases.

III. COMPREHENSIVE APPROACH TO NoC MODELING

As mentioned earlier, a NoC must contain computing IP blocks, routers, as well as a module that describes how the network components are interconnected. Therefore, there are models that explore a NoC in a comprehensive manner with all the necessary components present. There are few such models [23], [24], [25], because their usage is associated with a large number of difficulties: from the large time spent on obtaining the result right up to the impossibility of implementation of a NoC with a large number of nodes. At the same time, only comprehensive modeling makes it possible to fully evaluate the work of NoCs, and their application remains necessary.

A. SEPARATE NoC MODELING

Most models designed to study a NoC explore its individual parts: either computing IP blocks [26], [27], [28], [29], [30], or the structure of routers [31], [32], [33], [34], or the connection topology [35], [36], [37], [38], [39], [40], [41], [42]. Such models allow a deeper analysis of the components under study. For routers and topologies, this approach allows analysis of the communication subsystem with a larger number of nodes than with complex modeling while the time to obtain results for a separate component is reduced.

Currently, separate modeling of NoC components has a significant drawback because the results of the work of such models are in no way consistent with each other. This manifests itself both in various ways of parameterizing models and specifying the initial conditions for their operation and in obtaining simulation results. Simulation results cannot just be presented in different formats. But (depending on the model) they can be very different from each other. It should be mentioned separately that low-level modeling is associated with high-level modeling, where the simulation results are also difficult to combine into a single data format [20].

A consequence of the problem described above is the difficulty in data standardization. The problem is not only in combining the formats of input and output results but also in their reliability because the same algorithm can be implemented in different ways in models, which can lead to different results.

Thus, it is necessary to create a new approach to low-level NoC modeling and a specialized low-level NoC model to separate modeling of the system and combine the modeling results of various NoC components for the further use.

IV. LOW-LEVEL MODELING OF A NoC COMMUNICATION SUBSYSTEM

When low-level modeling of a NoC communication subsystem, the developer solves several tasks to obtain the final result: preparing a low-level model, coding it in the HDL, developing a set of tests to check the correctness of the model, prototyping the model, collecting and analyzing the results. To evaluate the correctness of the model, it is necessary to conduct modeling with various parameters, which leads to a constant change in its code to reflect the changes made. This task (like the task of collecting results) is time-consuming.

Thus, a single infrastructure for low-level modeling should include the following elements: a low-level model (a top-level module of the hierarchy, topology module, routing module, router module), verification model (testbenches, an interface module for connecting external reference high-level modules via the DPI interface), automation tools of model configuration and run, collection and processing of modeling results, and preparation of input data for modeling.

A. LOW-LEVEL MODEL OF A NoC COMMUNICATION SUBSYSTEM

To implement the possibility of complex low-level NoC modeling and separate modeling of its individual parts, a low-level model of the NoC communication subsystem was developed [43]. Its structure is shown in Figure 6.

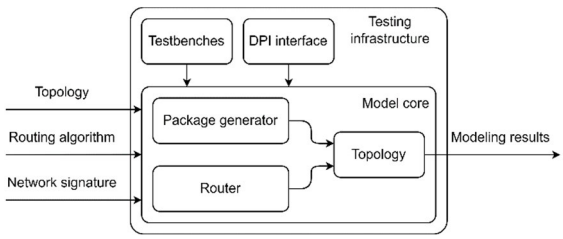

FIGURE 6. Structure of the low-level model of a NoC communication subsystem.

The model consists of two parts:

- model core – files in the Verilog HDL that implement a communication subsystem of the NoC under study;
- testing infrastructure – contains files for conducting automated event modeling of the model operation on the pre-prepared data.

The main components of the model are highlighted in the "Model Core" block. It consists of three elements: a router module, topology module, packet generator module. These modules allow modeling both communication subsystems and a NoC as a whole. These modules are developed in the Verilog HDL and adapted to run in Intel Quartus Prime Lite ECAD to further obtain an RTL description based on which binary programming files for the FPGA development board are prepared. Such an organization of the model core also

allows using the model core in the third-party software for designing and debugging a project on an FPGA.

The router module implements the structure of the input and output ports of the router, algorithm for receiving data packets, routing algorithm, and algorithm for sending packets. The router consists of x ports for receiving packets from other routers, x ports for sending packets to other routers, 1 port for receiving a packet from a compute node, 1 port for sending a packet to a compute node. The number x of input and output ports associated with other routers is configurable by the user depending on the topology selected. The number of ports connecting the router and computing node is fixed.

The router module implements the routing algorithm under study and is a state machine that implements the algorithm based on which the router operates. Using the information about the topology of a communication subsystem stored in the router, as well as the service information stored in the forwarded data packets, the port to which the packet must be sent so that it reaches the destination node is selected. For most routing algorithms to work in a router, it is necessary to store the information about the number of routers in the network, number of ports to which a packet can be forwarded, and also information about the type of connection of the nearest routers.

The task of the router is to determine which port the packet came to, and allocate the auxiliary information in it for the algorithm to work. This information is then processed by the routing algorithm to determine the number of routers to which the packet must be forwarded in order to reach its destination. At the final stage, it is necessary to determine to which port the packet should be sent, pack the auxiliary information received from the algorithm, and send the packet to the appropriate port.

The topology module implements the analyzed topology of the communication subsystem. It connects all the routers to each other, transfers the data between them, and sends the data to the computing node. The topology implements an undirected graph representing the structure of connections between the routers. This is the main module of the model core. It combines routers, computing nodes, and other modules with each other; provides the data transfer between them, and also returns the packet passing results for the further analysis.

The packet generator module allows transferring input packets to the network, tracking their path, and also collects the status of the packet delivery to the required computing node. The module generates the test data packets that are fed into the network, receives the packets from the network, and analyzes errors when the packet passes through the network.

The packet generator supports different types of traffic patterns: from synthetic (random, uniform, bit reversal, tornado, etc.) [22], [44], [45] right up to application-specific and benchmarks like PARSEC [46] and SPLASH [47] using script files.

The packet generator is auxiliary and in fact is not a structural element of the communication subsystem; it is needed to reduce the NoC model (it replaces the computing nodes). It emulates the generation of data packets when testing the network. All the ports of the routers are connected to the packet generator module for communication with computing nodes. This makes it possible to conduct a NoC research with more nodes than when using real computing nodes.

To provide testing of the model operation at the register level, for the model core files, an add-on with testbench files that check the correctness of the model based on ideal signals was developed. Since such a check is quite long, the components to allow making a high-level check were additionally implemented. One of such components is the SignalTap logic analyzer [48], which is part of the Intel Quartus Prime. This tool makes it possible to read the studied signals in the real time of the prototype on the FPGA development board and select the states of interest of the variables. The SystemVerilog DPI direct programming interface implemented helps to organize the call of functions developed in the C / C++ language in the code developed in the SystemVerilog language. Using this interface, one can organize the transfer of test data from the computer to the model, as well as obtain the results of the modeling.

B. ECAD ARCHITECTURE FOR LOW-LEVEL NoC MODELING

The NoC model proposed above is the basis for the ECAD architecture for low-level NoC modeling; the structure of this architecture is shown in Figure 7.

The ECAD software HDLNoCGen [43] was developed in C#; it implements the presented NoC low-level modeling architecture and provides support for circulant topologies for a NoC [8], [49]. It includes components for setting up the generation of the model core, tools for collecting data from the model core, as well as tools for managing the generation of data packets:

- model tuning module, which receives the necessary data for the parametric generation of the NoC communication subsystem, generates the core of the model, and also launches the model to obtain the working results;
- data processing module, which is necessary to obtain the results of the model and generate test data for the submission to the model in real time, as well as to collect the results of the modeling and compare them with the results of high-level modeling.

An important part of the ECAD is the model tuning module whose main component is the developed translator for parametric generation of the model core code. Also, this module allows multiple launch of the model with different parameters, which automates the process of generating the model and obtaining results.

The other modules of the developed ECAD are the result analysis tools. Some of them (tools for displaying network characteristics) are located in the data collector module because they provide an access to the raw data. The routing algorithm analysis tools and topology visualization tools are

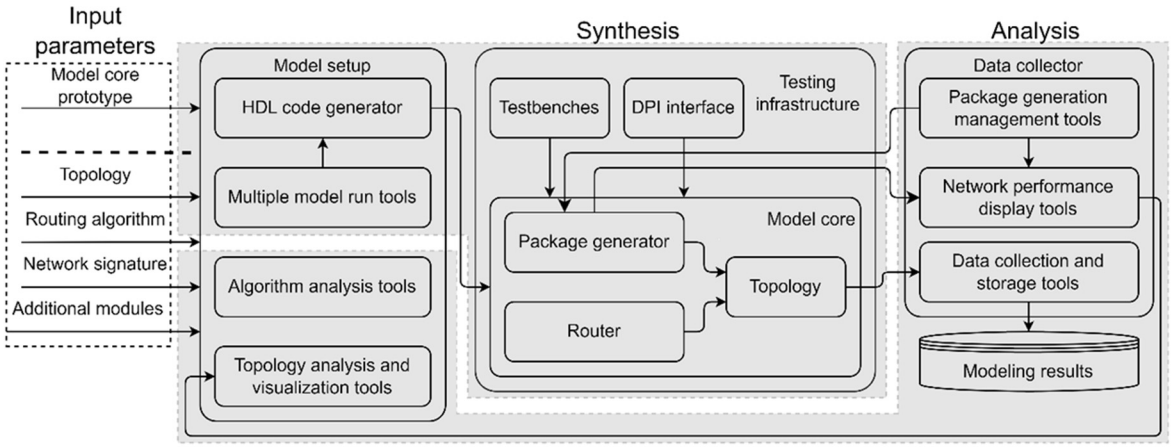

FIGURE 7. ECAD architecture for low-level NoC modeling.

located in the model setup block because they perform data processing from the model and allow them to be compared with the results obtained at the high-level modeling phase. They are needed to synchronize the results of low-level and high-level modeling.

The following set of parameters is fed to the ECAD input: a network topology, routing algorithm, network signature (which contains information about the number of nodes and the topology for their connection), and additional modules to be connected to the network. Optionally, one can set the location of the model core prototype.

The developed ECAD allows modeling a parameterized NoC communication subsystem to obtain an estimate of the number of consumed arithmetic logic blocks (ALM) and registers (REG, memory blocks) required for prototyping the communication subsystem.

All the components of the communication subsystem are implemented as separate modules, due to which the resource costs for adding the necessary components to the model for research are reduced because of the absence of the need for a complete redesign of the model code. This is possible due to the created universal interface, to which all the connected components transmit the data in their own format after which they are converted for the further use by other components. With the help of the universal interface, it is possible to add the additional components to the model for modeling a NoC as a whole.

For the ECAD developed, at the level of its architecture, it was proposed to divide its components into a synthesizing unit and analysis unit. The synthesizing block includes the model core, testing infrastructure, HDL code generator, and multiple model launch tools. Data collector components, algorithm analysis tools, and topology analysis and visualization tools are included in the result analysis block.

In accordance with the requirements of the generally accepted standards for the ECAD composition, it should consist of informational (a database of modeling results), mathematical (approximation models for predicting modeling results), linguistic (low-level models in the Verilog language), methodical (developed modeling techniques), and technical (an operational FPGA prototype) components; and software (ECAD software and auxiliary utilities). Information support includes the input data (based on which the parametric synthesis of the model takes place), as well as the results of the model operation, such as the consumed chip resources, clock frequency of the synthesized network project, and the efficiency of the components placement. The hardware includes a workstation on the Windows operating system, which runs the developed ECAD, Intel Quartus Prime Lite, that creates a network description file for the FPGA chip, and also the FPGA chip itself as part of the development board.

C. DATA FORMAT HARMONIZATION FOR PARAMETER TRANSFER BETWEEN NoC MODELS

High-level and low-level NoC models designed to study the NoC structural components initially have no connection with each other in terms of the mutual use of modeling results. With the traditional approach, when models of different levels are used separately, there is no question of their synchronization. Now, ensuring seamless integration of models with each other has a significant impact on the efficiency of interaction between them. We propose to use the fact that the models have partially the same data – information about the topology (the structure of connections of routers and their number), information about the router (the number of data transmission channels, their size), the routing algorithm, etc. These parameters are set in each model in the corresponding format, which leads to the need to transform the data by the user, and this increases the time of the study, let alone may cause errors. To eliminate these problems, a data preprocessing module was developed to adapt the format of the studied NoC characteristics for use in both high-level and low-level modeling. Models should be able to both receive parameters and issue them.

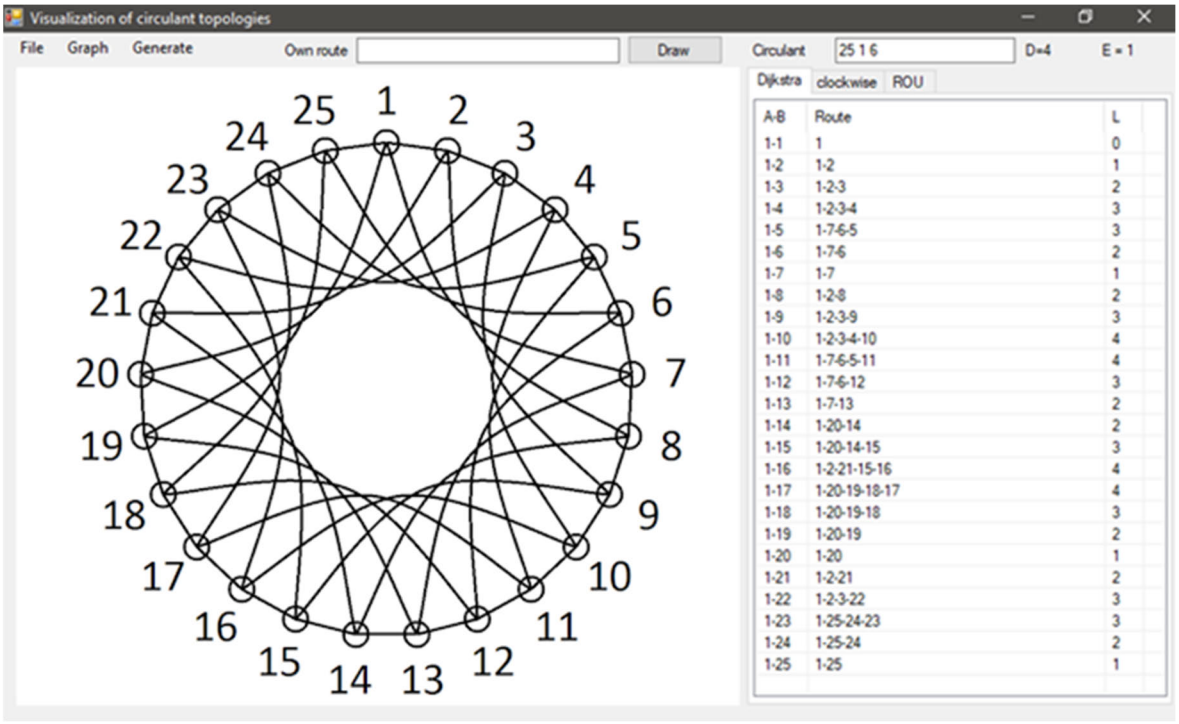

FIGURE 8. Window of topology settings for the low-level NoC model in HDLNoCGen software.

Topology information representation can be implemented in several ways. The user can prepare a configuration file that will contain a description of the characteristics of the topology under study. In this case, the user is limited to the set of topologies implemented in the model. Another way to specify topology information is to describe the topology using an adjacency matrix in which the dimensions of the matrix in height and width are given by the number of routers, and the links between them are indicated as 1 or 0 at the intersection of the row and column of the corresponding routers. The second method is the most universal as it allows using not only the developed proprietary software tools for the topology analysis but also third-party software for the graph analysis. In this case, the file with the description of the topology configuration becomes much larger compared with the one in the first case.

In the developed ECAD HDLNoCGen [43], a combination of the first and second methods of specifying the topology characteristics is implemented. The user can specify information in the form of a configuration file in which characteristics for the topology supported by default are specified, as well as the adjacency matrix (for the unsupported topology) is presented. In the developed ECAD, this module is called ''Topology analysis and visualization tools'' and is part of the ECAD configuration module (Figure 8). This module allows getting information about the topology, displaying it in the graphical window, and preparing the data for setting up a low-level NoC model. An example of a graphical window for entering settings and visualizing the topology is shown in Figure 8.

As the user can initially set the settings of the low-level model and subsequently use them in a high-level one, the ECAD customization module allows getting them in the form of a configuration file. The router settings are transferred in a similar way. The user can set them in the form of a configuration file, which will go to the input of the models.

Particular attention should be paid to the choice of the routing algorithm in the models. This is the most difficult setting in terms of model matching. Low-level models, like high-level ones, can be developed in different languages and have different code structures. The implementation of the supported algorithms in each model is different, which can lead to different results of the same algorithm in different models. To solve this problem, descriptions of routing algorithms are used in the form of separate text files connected to the models. Currently, the same approach is used in preparing a file with an algorithm as in generating a low-level model by translating code from C# into Verilog. The file describes the algorithm in the form of a function in the C# language with restrictions on the data types and syntactic constructions used. The configuration file of the model indicates which algorithm to use for modeling. When the model runs, the selected file with the routing algorithm is recognized by the model and executed.

The advantage of the approach described is that it eliminates the need to implement the same algorithm in several programming languages. The algorithm is the same in all models, and there is no difference in the modeling results depending on the implementation of the algorithm. This also expands the list of algorithms supported by the model.

At the same time, this approach requires the development of additional code to interpret the algorithm, regardless of the language in which the model was developed.

V. AUTOMATION OF LOW-LEVEL NoC MODELING

The developed ECAD for low-level NoC modeling is based on the low-level model. But the most important subsystem that distinguishes the developed ECAD from NoC models are modules that implement modeling automation and provide the achieved increase in the speed, accuracy, and simplification of the process of low-level NoC modeling.

A. GENERATION OF A LOW-LEVEL MODEL OF THE NoC COMMUNICATION SUBSYSTEM

In the classical approach to low-level NoC modeling, for each analyzed configuration of the NoC communication subsystem, it is necessary to create own model core files, which is a routine and time-consuming task. To automate this process, a model tuning module in HDLNoCGen was developed [43]. This module was developed in C#. Its task is to generate model core files based on information about the topology and the routing algorithm. The main component of this module is the HDL code generator. Structurally, the HDL code generator is a translator that can operate in two modes depending on the data supplied to the ECAD. The operating modes of the translator are shown in Figures 9 and 10.

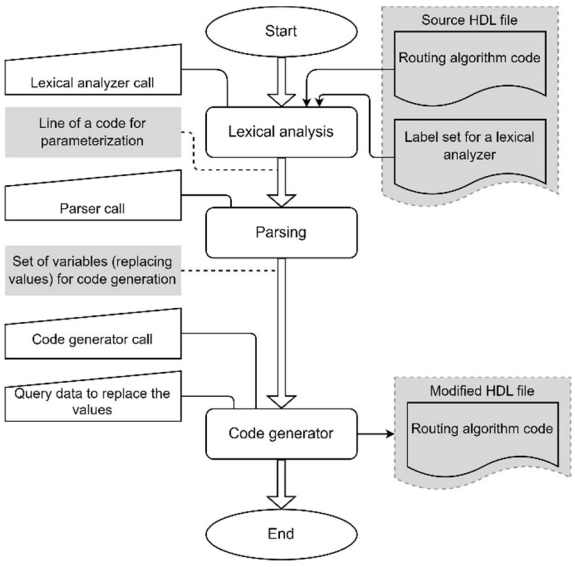

FIGURE 9. Modes of operation of the HDL code generator based on the model core prototype.

In the first mode, the work of the translator is based on the use of the model core prototype (Figure 9). The developer creates a prototype of the model core once. During further operation of the system, it is required to change only the file in which the routing algorithm is implemented. In this file, the developer notates (place specialized comments) the lines of code that need to be changed to create a parameterized model. The translator receives this file as input, reads it line by line,

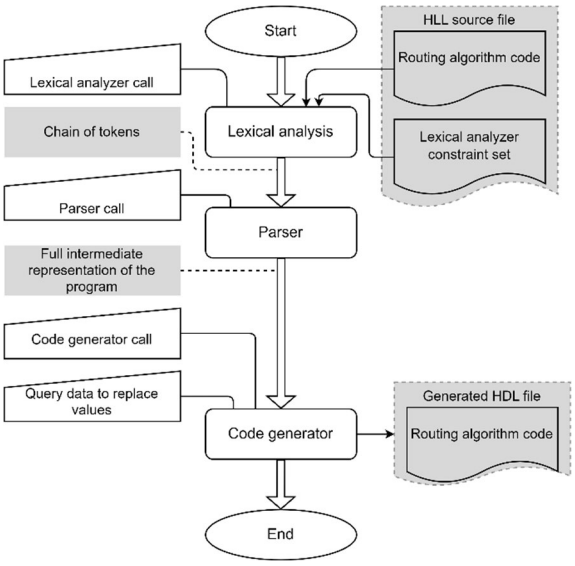

FIGURE 10. HDL code generator operating modes in C#-to-Verilog translator mode.

and passes it to the lexical analyzer. The lexical analyzer in each line searches for a label to change. The found string is passed to the parser, which parses this string, determines the parameter to be changed, and also gives the name of the parameter to be set. A set of variables (consisting of a variable and a value to be replaced) is passed to the code generator, which requests all the necessary data from the model and creates a new file with the routing algorithm.

The second mode of the generator is C#-to-Verilog translation (Figure 10). In this mode, the generator does not need a model kernel prototype. The input of the translator is a file with a routing algorithm developed in C#. A file with restrictions for the lexical analyzer is also supplied. Based on these inputs, the code generator generates a file with a routing algorithm in the Verilog language, as well as all other necessary files.

The translator proposed is not a classical one-pass or two-pass translator [50]. It is a translator adapted to the generation of the NoC communication subsystem. It differs in that the translator checks the type of input data before starting. If the input is Verilog files there is no actual code translation. The code input is parameterized based on additional parameters described in the code itself. If the input is a file describing a routing algorithm in C# language its adapted translation takes place.

For the file with the routing algorithm, we developed a number of restrictions on its structure, as well as on the set of C# tokens that can be used to describe the algorithm. Structurally, the file is a C# function to which all necessary data are passed as parameters.

As a result of the analysis of routing algorithms used in both high-level and low-level models, restrictions on the use of C# tokens were formed. It is allowed to use variables of integer, long, boolean types, as well as arrays and lists based on these data types. It is allowed to use a limited set

of mathematical operations: addition, subtraction, multiplication, division, absolute number calculation, inversion, bitwise shifts. It is allowed to use assignment, strict and non-strict comparison operations. Syntactic constructions of the language are also restricted: it is possible to use the conditional operator if / else, the case selection operator, as well as the for loop. The above-described capabilities of C# language are enough to implement any routing algorithm for a NoC. Thus, the translator can translate only a limited set of lexemes.

Another feature of the translator is that depending on the selected C# algorithm string, it either translates it into Verilog language or performs parametric substitution of the code block. This applies to the description of the module header, as well as the formation of the block for selecting the direction of the packet to the output port of the router.

The description of the Verilog module header depends on the set of topology parameters for which the routing algorithm is implemented, as well as on the parameters for the algorithm itself. And if the parameters for the algorithm operation are equally present in the C# and Verilog code, the information about the topology parameters is set in different ways. For the C# algorithm, only the set of formers for mathematical calculation of the graph is enough, and there is no need to explicitly describe all the ports of the router. For a Verilog module, a description of the router ports is necessary. Thus, when the translator analyzes the input parameters of the algorithm in C#, the number of router ports is calculated, the dimension of the data parcels they can accept is specified, both input and output ports are formed, and special names are given for each of them. At the end, the generated block of code describing this data is inserted into the module header.

The C# algorithm works without taking into account the real data in the data packet that needs to be sent between routers, it only takes into account the auxiliary data for the algorithm to work. Also, the algorithm does not consider the availability of ports for communication with computational nodes. When translating into Verilog language, these data are also taken into account; so, a large block of code in which auxiliary information for the algorithm operation is substituted into the data packet is generated. Also, a block is formed where (based on the results of the algorithm operation) the output port of the router to which the data packet should be sent is calculated.

The developed translator not only allows automatic synthesis of the parametric model of the communication subsystem but also provides the possibility of matching the analysis of the routing algorithm with a high-level model.

The model setup module can generate all additional necessary Verilog files for a NoC creation in automatic mode, as well as generate a project for Quartus Prime and run modeling / prototyping of the communication subsystem using TCL commands.

The possibility of multiple model launching by submitting a pre-formed configuration file to ECAD is implemented. This file specifies the signature of the network topology to be created and the requirements for generating the model kernel code: either parameters for the algorithm and the algorithm itself in C# or the path to the location of the prototype of the model kernel are given. Optionally, one can specify what additional modules should be connected to the model. Additionally, it is possible to specify the settings to be considered when creating a project for Quartus Prime. In one configuration file one can specify several network descriptions that will be sequentially created after which their modeling will be performed. The simulation results will be collected in a single summary table for the further analysis.

Additionally, in the module of model adjustment, the possibility of high-level check of correctness of the structure of the communication subsystem and routing algorithm by generating routing paths from the zero router to all other routers is implemented.

B. METHODOLOGY FOR AUTOMATED END-TO-END NoC DESIGN

Based on the method of separate modeling described in Section III and the use of a specialized generator of the low-level model of the NoC communication subsystem, we formulate a methodology for automated end-to-end NoC design – by automatic parameterization of the low-level model in order to coordinate the results of NoC modeling at all the phases of design and automation of the modeling process. An important element of the proposed methodology is the use of FPGA development boards for cosimulation in order to characterize the NoC operation. Its essence is that a prototype NoC is loaded on the FPGA as part of the model, and the results are processed on the workstation.

The proposed methodology addresses several NoC design and modeling challenges:

- automation of the modeling process by reducing the time required to develop a NoC model;
- development of a special parameterized NoC model providing separate modeling of NoC components and a NoC as a whole;
- synchronization of modeling outputs and outputs between low-level and high-level models.

Application of the method of separate NoC modeling in conjunction with the generator of low-level models allows to automate the process of synthesis of specialized parameterized low-level NoC models, which shortens the developer's resource costs by reducing the time for the preparation of specialized models. Due to these factors, the process of verification the NoC communication subsystem operation can be accelerated up to 15 times.

Figure 11 shows the dependence of the speed of obtaining data describing the network operation on the number of nodes using the NoC methodology of automated end-to-end design, which is the basis of the developed ECAD in comparison with the classical approach based on the use of low-level modeling with the help of event-based models (using ModelSim as an example). In order to evaluate the speed of obtaining results, modeling of the passage of one data packet was carried out.

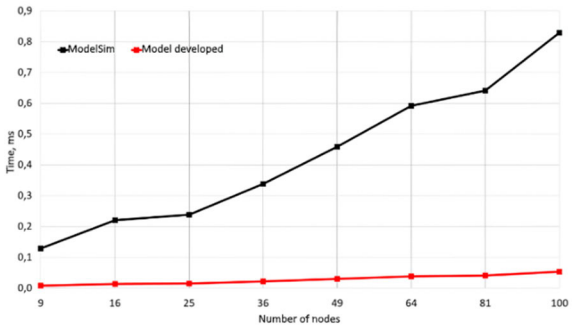

FIGURE 11. Comparison of the speed of obtaining NoC results using the tools developed and ModelSim.

The increment of time required to obtain NoC results, using the proposed methodology, grows linearly in contrast to the increment of time using ModelSim. This is because in event-driven modeling using ModelSim, signals are calculated at each point in time. With the proposed technique, due to cosimulation, results are obtained on the prototype, and the dependence of the time to obtain results on the size of the network is expressed in terms of the path length required for a packet to traverse, as well as the clock frequency of the prototype. The diameter for topologies with increasing number of nodes grows in a stepwise manner rather than linearly, as shown in [51]. For this reason, the increase in the time required to obtain results using the proposed technique should also increase in jumps. The graph flattens out by decreasing the frequency of the prototype as the number of nodes in the network increases. It is also worth noting that (using the proposed technique) the increase in the required simulation time as a function of the number of nodes in the network does not grow as fast as when using ModelSim.

It should be noted that we are not comparing the speed of NoC modeling in ModelSim with execution on an FPGA but two different methodologies when (instead of manually setting up, developing and then executing an HDL model) automated tools that allow the model to be generated end-to-end, configured, synthesized, and run using automated means are used. The cost of performing all these steps is more than offset by the gain in speed of cosimulation and multiple runs of several simulations in a row.

In Section II, different types of NoC behavioral modeling were presented. When analyzing simulation results obtained from high-level models and low-level models, one of the tasks is to synchronize the obtained data. This task is quite time consuming, as the developer needs to analyze the results and compare them between models. Additional complexity in this is created: despite the fact that high-level and low-level model can investigate the same NoC configuration, the implementation of some of its components, such as routing algorithm, may differ in various models, which leads to difficulties in checking the results for their correctness. As a result of our analysis of the NoC design subject area and during the review of low-level NoC models, no tools were found that allow the description of NoC configurations to be compared between models of different levels of abstraction.

The proposed methodology, in terms of using a specialized low-level model generator, allows us to solve this problem. To use the core model generator in C#-to-Verilog mode, it is necessary to specify the description of the routing algorithm in C# language. The low-level model is created automatically based on the rules of translation of the algorithm description from C# to Verilog language. Thus, the synthesized low-level NoC model in the part of the routing algorithm description will fully coincide with the high-level model in which this algorithm was implemented. This ensures that the two models are fully consistent because one is the source model for the other. It should be noted that to provide this possibility, the algorithm description in the high-level model must be implemented in a certain way, in accordance with the requirements and constraints of the generator of the low-level model.

To synchronize the other parameters of the different level models, a configuration file is used, which is input to the developed HDLNoCGen ECAD. This configuration file can also be used in high level models for their parameterization.

VI. APPROBATION OF THE DEVELOPED ECAD IN THE REAL TASKS OF NoC DESIGN

All the proposed solutions, methods, and techniques combined in a single ECAD were used in various studies of different aspects of the NoC functioning. For example, in [10], with the help of the developed ECAD, data were obtained for comparing the routing algorithms of the NoC communication subsystem. With the help of algorithm analysis tools and topology analysis and visualization tools, the correctness of high-level descriptions of the studied routing algorithms was shown. Using the HDL code generator based on a specialized translator which was used in the mode of operation according to the model core prototype, specialized models of the NoC communication subsystem were obtained for the number of nodes from 9 to 100, where the value of the number of nodes was formed as $N = k^2, k > 3$ (k is a natural number). The use of the developed ECAD made it possible to achieve a network size of up to 100 nodes and speed up the process of obtaining results up to 10 times. Accurate estimates of the consumed chip resources were obtained using a specialized low-level model of the NoC communication subsystem, based on which graphs were built, and approximation functions were calculated for interpolation and extrapolation of the theoretical estimate of chip resources for networks with a different number of nodes.

In [37], using the developed ECAD, with the help of the HDL code generator, specialized low level models of the NoC communication subsystem were synthesized to test the performance of the proposed routing algorithm. By means of packet generation controls, the correctness of the algorithm proposed was proved.

In [52], an assessment of the chip resources required to implement the NoC communication subsystem based on the routing algorithm proposed was made. The developed ECAD

made it possible to check the correctness of the routing algorithm implemented in a high-level language for NoC implementations with hundreds of nodes. With the help of ECAD tools, low-level parameterized models of the NoC communication subsystem with the routing algorithm under study were synthesized, and the results of low-level modeling were also obtained. The data collection and storage tools included in the developed ECAD allowed comparing the routing algorithm proposed with other algorithms for the selected topology in order to determine an algorithm that is more compact in terms of the required chip resources.

In [53], using the development methodology implemented in ECAD HDLNoCGen, a modification of the high-level BookSim NoC model was carried out, as a result of which the number of topologies that can be explored using this model was expanded, and new routing algorithms were added. In addition, the structure of the configuration file was modified to add the support for new topologies and routing algorithms. Also, the structure of the configuration file was adapted for use in HDLNoCGen ECAD to match model settings and match modeling results, which made it possible to perform the end-to-end NoC design by combining high-level and low-level modeling results.

The developed tools, low-level model, methods and techniques of design and modeling (which are the basis of the ECAD developed) significantly reduced the resource costs for creating special parameterized low-level NoC models, reduced the time for their preparation and for obtaining modeling results. The conducted extensive testing proved the applicability of the solutions proposed and demonstrated the reliability of the results obtained, as well as their reproducibility.

VII. CONCLUSION

The main result of this work is the solution of an actual scientific and applied problem related to automating the process of developing low-level models of NoC communication subsystems (which makes it possible to speed up the modeling process up to 15 times), as well as increasing the size of the simulated NoC by developing a new low-level model of its communication subsystem, which (due to the universal communication interface of the NoC components) allows modeling both the network as a whole and its individual components without the need to modify the whole model.

Modeling individual NoC components gives an opportunity to evaluate their impact on the network separately and (based on the data obtained) make a parametric selection of the optimal settings for the operation of the selected NoC component. Also, separate modeling of NoC components makes it possible to speed up the process of modeling both the network as a whole and the components under study.

Based on the model of the NoC communication subsystem proposed, the ECAD architecture in the form of the HDLNoCGen software, which allows automating the process of preparing and generating an HDL description of the model in the Verilog language, is developed and implemented. The developed model allows modeling a parameterized NoC communication subsystem to obtain an estimate of the number of consumed logical blocks and registers required for communication subsystem prototyping. All the components of the communication subsystem are implemented as separate modules due to which the resource costs for adding the necessary components for the study are reduced because of the absence of the need to completely rework the model code every time.

The developed ECAD and low-level modeling automation methods applied in the studies of the operation of routing algorithms and the topology influence on the NoC operation, as well as the data on the consumed resources for networks with different parameters obtained are presented. Based on these results, graphs and dependencies are constructed, and they confirm the applicability of the solutions proposed, as well as demonstrate the reliability and reproducibility of the results obtained.

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] G. Nychis, C. Fallin, T. Moscibroda, and O. Mutlu, "Next generation on-chip networks: What kind of congestion control do we need?" in *Proc. 9th ACM SIGCOMM Workshop Hot Topics Netw. (Hotnets)*, 2010, pp. 1–6, doi: 10.1145/1868447.1868459.

[2] J. Paul, W. Stechele, B. Oechslein, C. Erhardt, J. Schedel, D. Lohmann, W. Schröder-Preikschat, M. Kröhnert, T. Asfour, É. Sousa, V. Lari, F. Hannig, J. Teich, A. Grudnitsky, L. Bauer, and J. Henkel, "Resource-awareness on heterogeneous MPSoCs for image processing," *J. Syst. Archit.*, vol. 61, no. 10, pp. 668–680, 2015, doi: 10.1016/j.sysarc.2015.09.002.

[3] M. S. Abdelfattah, A. Bitar, and V. Betz, "Design and applications for embedded networks-on-chip on FPGAs," *IEEE Trans. Comput.*, vol. 66, no. 6, pp. 1008–1021, Jun. 2017, doi: 10.1109/TC.2016.2621045.

[4] W. J. Dally and B. P. Towles, *Principles and Practices of Interconnection Networks*. Amsterdam, The Netherlands: Elsevier, 2004.

[5] R. Marculescu and P. Bogdan, "The chip is the network: Toward a science of network-on-Chip design," *Found. Trends Electron. Design Autom.*, vol. 2, no. 4, pp. 371–461, 2007, doi: 10.1561/1000000011.

[6] G. Michelogiannakis and W. J. Dally, "Router designs for elastic buffer on-chip networks," in *Proc. Conf. High Perform. Comput. Netw., Storage Anal.*, Nov. 2009, pp. 1–10, doi: 10.1145/1654059.1654062.

[7] A. Sai Kumar and T. V. K. Hanumantha Rao, "An efficient low latency router architecture for mesh-based NoC," in *Advances in Communications, Signal Processing, and VLSI* (Lecture Notes in Electrical Engineering), vol. 722. Singapore: Springer, 2021, pp. 241–248, doi: 10.1007/978-981-33-4058-9_21.

[8] E. A. Monakhova, O. G. Monakhov, and A. Y. Romanov, "Routing algorithms in optimal degree four circulant networks based on relative addressing: Comparative analysis for networks-on-chip," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 1, pp. 413–425, Jan. 2023, doi: 10.1109/TNSE.2022.3211985.

[9] R. Singh, M. K. Bohra, P. Hemrajani, A. Kalla, D. P. Bhatt, N. Purohit, and M. Daneshtalab, "Review, analysis, and implementation of path selection strategies for 2D NoCs," *IEEE Access*, vol. 10, pp. 129245–129268, 2022, doi: 10.1109/ACCESS.2022.3227460.

[10] A. Y. Romanov, E. V. Lezhnev, A. Y. 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, Jan. 2020, Art. no. e03183, doi: 10.1016/j.heliyon.2020.e03183.

[11] G. Howser, "The OSI seven layer model," in *Computer Networks and the Internet*. Cham, Switzerland: Springer, 2020, pp. 7–32, doi: 10.1007/978-3-030-34496-2_2.

[12] M. S. Das, "Architecture of multi-processor systems using networks on chip (NoC): An overview," *CVR J. Sci. Technol.*, vol. 22, no. 1, pp. 7–15, 2022, doi: 10.32377/cvrjst2202.
[13] N. Jafarzadeh, A. Jalili, J. A. Alzubi, K. Rezaee, Y. Liu, M. Gheisari, B. Sadeghi Bigham, and A. Javadpour, "A novel buffering fault-tolerance approach for network on chip (NoC)," *IET Circuits, Devices Syst.*, vol. 17, no. 4, pp. 250–257, Jul. 2023, doi: 10.1049/cds2.12127.
[14] S. R. Pokhrel, N. Kumar, and A. Walid, "Towards ultra reliable low latency multipath TCP for connected autonomous vehicles," *IEEE Trans. Veh. Technol.*, vol. 70, no. 8, pp. 8175–8185, Aug. 2021, doi: 10.1109/TVT.2021.3093632.
[15] K. G. Mkongwa, C. Zhang, A. C. Pogaku, Q. Liu, M. Munochiveyi, and A.-T. Le, "Reliable data transmission mechanism in coexisting IEEE 802.15.4 beacon enabled wireless body area networks," in *Proc. 10th Int. Conf. Inf. Autom. Sustainability (ICIAfS)*, Aug. 2021, pp. 483–488, doi: 10.1109/ICIAfS52090.2021.9605877.
[16] N. A. Sherwani, *Algorithms for VLSI Physical Design Automation*. New York, NY, USA: Springer, 1993, p. 488, doi: 10.1007/978-1-4757-2219-2.
[17] R. Hartson and P. Pyla, "Bottom-up versus top-down design," in *The UX Book*, 2nd ed. MA, USA: Morgan Kaufmann Publishers, 2019, pp. 279–291.
[18] J.-P. Morin, F. Lemery, E. Nercessian, V. Sharma, J. Benkoski, and D. Samani, "A practical approach to Top/Down analog circuit design," in *Proc. 19th Eur. Solid-State Circuits Conf. (ESSCIRC)*, vol. 1, Sep. 1993, pp. 49–52.
[19] K. S. Kundert and O. Zinke, *The Designer's Guide to Verilog-AMS*. New York, NY, USA: Springer, 2004, doi: 10.1007/b117108.
[20] A. Romanov and A. Ivannikov, "SystemC language usage as the alternative to the HDL and high-level modeling for NoC simulation," *Int. J. Embedded Real-Time Commun. Syst.*, vol. 9, no. 2, pp. 18–31, 2018, doi: 10.4018/IJERTCS.2018070102.
[21] O. Matoussi, "NoC performance model for efficient network latency estimation," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Feb. 2021, pp. 994–999, doi: 10.23919/DATE51398.2021.9474101.
[22] N. Jiang, D. U. Becker, G. Michelogiannakis, J. Balfour, B. Towles, D. E. Shaw, J. Kim, and W. J. Dally, "A detailed and flexible cycle-accurate network-on-chip simulator," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, Apr. 2013, pp. 86–96, doi: 10.1109/ISPASS.2013.6557149.
[23] R. C. Harting, V. Parikh, and W. J. Dally, "The utility of fast active messages on many-core chips: Efficient supercomputing project," in *Proc. IEEE Hot Chips 23 Symp. (HCS)*, Aug. 2011, p. 1, doi: 10.1109/HOTCHIPS.2011.7477516.
[24] A. E. Ryazanova, A. A. Amerikanov, and E. V. Lezhnev, "Development of multiprocessor system-on-chip based on soft processor cores schoolMIPS," in *Proc. J. Phys., Conf.*, Feb. 2019, vol. 1163, no. 1, Art. no. 012026, doi: 10.1088/1742-6596/1163/1/012026.
[25] T. V. Chu, S. Sato, and K. Kise, "Fast and cycle-accurate emulation of large-scale Networks-on-Chip using a single FPGA," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 10, no. 4, pp. 1–27, Dec. 2017, doi: 10.1145/3151758.
[26] *A small MIPS CPU core SchoolMIPS*. Accessed: Mar. 27, 2024. [Online]. Available: https://github.com/MIPSfpga/schoolMIPS
[27] *Tiny RISCV CPU SchoolRISCV*. Accessed: Mar. 27, 2024. [Online]. Available: https://github.com/zhelnio/schoolRISCV
[28] J. Im and S. Kang, "Comparative analysis between verilog and chisel in RISC-V core design and verification," in *Proc. 18th Int. SoC Design Conf. (ISOCC)*, Oct. 2021, pp. 59–60, doi: 10.1109/ISOCC53507.2021.9614007.
[29] S. Prabhakaran, N. Mathan, and V. Vedanarayanan, "Design and analysis of a multi clocked pipelined processor based on RISC-V," in *Proc. Int. Conf. Commun., Comput. Internet Things (IC3IoT)*, Mar. 2022, pp. 1–5, doi: 10.1109/IC3IOT53935.2022.9767960.
[30] V. Desalphine, S. Dashora, L. Mali, K. Suhas, A. Raveendran, and D. Selvakumar, "Novel method for verification and performance evaluation of a non-blocking level-1 instruction cache designed for out-of-order RISC-V superscaler processor on FPGA," in *Proc. 24th Int. Symp. VLSI Design Test (VDAT)*, Mali, Jul. 2020, pp. 1–4, doi: 10.1109/VDAT50263.2020.9190377.
[31] D. U. Becker, N. Jiang, G. Michelogiannakis, and W. J. Dally, "Adaptive backpressure: Efficient buffer management for on-chip networks," in *Proc. IEEE 30th Int. Conf. Comput. Design (ICCD)*, Sep. 2012, pp. 419–426, doi: 10.1109/ICCD.2012.6378673.
[32] H. Jia and J. Lei, "System level modeling and verification for routers of 3D-NoC based on system C," in *Proc. IEEE Int. Conf. Electron Devices Solid-State Circuits (EDSSC)*, Jun. 2015, pp. 479–482, doi: 10.1109/EDSSC.2015.7285155.
[33] K. Tatas, "High-performance 3D NoC bufferless router with approximate priority comparison," in *Proc. 7th Int. Conf. Modern Circuits Syst. Technol. (MOCAST)*, May 2018, pp. 1–4, doi: 10.1109/MOCAST.2018.8376617.
[34] M. Zhang and C.-S. Choy, "Low-cost VC allocator design for virtual channel wormhole routers in networks-on-chip," in *Proc. 2nd ACM/IEEE Int. Symp. Netw.–Chip (NOCS)*, Apr. 2008, pp. 207–208, doi: 10.1109/NOCS.2008.4492740.
[35] J. D. Owens, W. J. Dally, R. Ho, D. N. Jayasimha, S. W. Keckler, and L.-S. Peh, "Research challenges for on-chip interconnection networks," *IEEE Micro*, vol. 27, no. 5, pp. 96–108, Sep. 2007, doi: 10.1109/MM.2007.4378787.
[36] J. Kim, W. J. Dally, and D. Abts, "Efficient topologies for large-scale cluster networks," in *Proc. Conf. Opt. Fiber Commun. (OFC/NFOEC), Collocated Nat. Fiber Optic Eng. Conf.*, Mar. 2010, pp. 1–3, doi: 10.1364/OFC.2010.OMV1.
[37] E. A. Monakhova, O. G. Monakhov, A. Yu. Romanov, and E. V. Lezhnev, "Analytical routing algorithm for networks-on-chip with the three-dimensional circulant topology," in *Proc. Moscow Workshop Electron. Netw. Technol. (MWENT)*, Mar. 2020, pp. 1–6, doi: 10.1109/MWENT47943.2020.9067418.
[38] 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, Feb. 2007, doi: 10.1109/L-CA.2007.10.
[39] A. W. Yin, T. C. Xu, P. Liljeberg, and H. Tenhunen, "Explorations of honeycomb topologies for network-on-chip," in *Proc. 6th IFIP Int. Conf. Netw. Parallel Comput.*, Oct. 2009, pp. 73–79, doi: 10.1109/NPC.2009.34.
[40] P. Narayanasamy, S. Gopalakrishnan, and S. Muthurathinam, "Custom NoC topology generation using discrete antlion trapping mechanism," *Integration*, vol. 76, pp. 76–86, Jan. 2021, doi: 10.1016/j.vlsi.2020.09.001.
[41] M. N. M. Ali, M. M. H. Rahman, A. A. Ibrahim, M. Al-Naeem, and E. Hossain, "A high static performance hierarchical three-dimensional shifted completely connected network," *IEEE Access*, vol. 10, pp. 43812–43836, 2022, doi: 10.1109/ACCESS.2022.3168728.
[42] I. A. Alimi, R. K. Patel, O. Aboderin, A. M. Abdalla, R. A. Gbadamosi, N. J. Muga, A. N. Pinto, and A. L. Teixeira, "Network-on-chip topologies: Potentials, technical challenges, recent advances and research direction," in *Network-on-Chip-Architecture, Optimization, and Design Explorations*. London, U.K.: IntechOpen, Apr. 2021, doi: 10.5772/INTECHOPEN.97262.
[43] E. V. Lezhnev. *HDLNoCGen: Verilog Code Generator of Communication Subsystem for Networks-on-Chip*. Accessed: Mar. 27, 2024. [Online]. Available: https://github.com/evgenii-lezhnev/HDLNoCGen
[44] Y. Li and P. Zhou, "Fast and accurate NoC latency estimation for application-specific traffics via machine learning," *IEEE Trans. Circuits Syst. II: Exp. Briefs*, vol. 70, no. 9, pp. 3569–3573, Sep. 2023, doi: 10.1109/TCSII.2023.3258700.
[45] S. K. Mandal, A. Krishnakumar, R. Ayoub, M. Kishinevsky, and U. Y. Ogras, "Performance analysis of priority-aware NoCs with deflection routing under traffic congestion," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, Nov. 2020, pp. 1–9, doi: 10.1145/3400302.3415654.
[46] A. S. Kumar and T. V. K. H. Rao, "Performance assessment of adaptive core mapping for NoC-based architectures," *Int. J. Embedded Syst.*, vol. 15, no. 5, pp. 395–403, 2022, doi: 10.1504/IJES.2022.127149.
[47] A. S. Kumar and T. V. K. H. Rao, "Scalable benchmark synthesis for performance evaluation of NoC core mapping," *Microprocessors Microsyst.*, vol. 79, Nov. 2020, Art. no. 103272, doi: 10.1016/j.micpro.2020.103272.
[48] K. Nirmaladevi and J. Sundararajan, "Low power NoC architecture based dynamic reconfigurable system," *Cluster Comput.*, vol. 22, no. S5, pp. 11489–11500, Sep. 2019, doi: 10.1007/s10586-017-1413-3.
[49] O. G. Monakhov, E. A. Monakhova, A. Yu. Romanov, A. M. Sukhov, and E. V. Lezhnev, "Adaptive dynamic shortest path search algorithm in networks-on-chip based on circulant topologies," *IEEE Access*, vol. 9, pp. 160836–160846, 2021, doi: 10.1109/ACCESS.2021.3131635.
[50] H. Wang, J. Li, and K. He, "Hierarchical ensemble reduction and learning for resource-constrained computing," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 25, no. 1, pp. 1–21, Jan. 2020, doi: 10.1145/3365224.

[51] A. Y. Romanov, A. A. Amerikanov, and E. V. Lezhnev, "Analysis of approaches for synthesis of networks-on-chip by using circulant topologies," in *Proc. J. Phys., Conf.*, 2018, vol. 1050, no. 1, Art. no. 012071, doi: 10.1088/1742-6596/1050/1/012071.
[52] E. A. Monakhova, A. Y. Romanov, and E. V. Lezhnev, "Shortest path search algorithm in optimal two-dimensional circulant networks: Implementation for networks-on-Chip," *IEEE Access*, vol. 8, pp. 215010–215019, 2020, doi: 10.1109/ACCESS.2020.3040323.
[53] A. Y. Romanov, E. V. Lezhnev, and A. A. Amerikanov, "Modification of the BookSim simulator for modeling networks-on-chip based on two-dimensional circulant topologies," in *Proc. 6th Int. Conf. Actual Probl. Syst. Softw. Eng.*, vol. 2514, 2019, pp. 182–192.

EVGENY V. LEZHNEV (Member, IEEE) received the B.S. and M.S. degrees in computer networks and systems and the Ph.D. degree in technical sciences from the National Research University Higher School of Economics (HSE University), Moscow, Russia, in 2015, 2017, and 2022, respectively. Since 2018, he has been a Research Trainee with the Laboratory of Computer-Aided Design Systems. Since 2020, he has been an Assistant Teacher with the HSE Tikhonov Moscow Institute of Electronics and Mathematics. He is the coauthor of 35 articles. His research interests include low-level modeling and development of NoC topologies. His awards and honors include the Golden HSE Award 2019 in the Silver Nestling Nomination and the Innovate FPGA (Intel) Award.

VLADIMIR V. ZUNIN (Member, IEEE) received the B.S. and M.S. degrees in computer networks and systems from the National Research University Higher School of Economics (HSE University), Moscow, Russia, in 2020 and 2022, respectively, where he is currently pursuing the Ph.D. degree in computer modeling and design automation. Since 2021, he has been a Research Trainee with the Laboratory of Computer-Aided Design Systems. Since 2022, he has been an Assistant Teacher with the HSE Tikhonov Moscow Institute of Electronics and Mathematics. He is the coauthor of five articles. His research interests include machine learning and development of FPGA systems.

ALEKSANDR A. AMERIKANOV (Member, IEEE) received the B.S. and M.S. degrees in computer networks and systems and the Ph.D. degree in technical sciences from the National Research University Higher School of Economics (HSE University), Moscow, Russia, in 2015, 2017, and 2022, respectively. Since 2018, he has been a Research Trainee with the Laboratory of Computer-Aided Design Systems. Since 2023, he has been an Associate Professor with HSE University. He is the coauthor of 30 publications. His research interests include high-level modeling and development of NoC topologies.

ALEKSANDR Y. ROMANOV (Member, IEEE) received the B.S. and M.S. degrees in computer systems and networks from the National Technical University Kharkov Polytechnic Institute (NTU KhPI), Kharkov, Ukraine, in 2007 and 2009, respectively, and the Ph.D. degree in technical sciences from the Federal State-Funded Institution of Science Institute for Design Problems in Microelectronics of Russian Academy of Sciences (IPPM RAS), Zelenograd, Russia, in 2015. He is currently an Associate Professor with the National Research University Higher School of Economics (HSE University), Moscow; the HSE Tikhonov Moscow Institute of Electronics and Mathematics. He is also the University Teacher of the following subjects: SoC development and system design of electronic devices. He is the author of the book Digital Design: Practical Course and more than 150 scientific articles. His research interests include SoC, NoC, embedded systems, FPGA, and neural networks. He is an IEEE Industrial Electronics Society Member and an ACM Member. He has received numerous awards as the supervisor of students' teams taking part in international competitions in programming, electronics, and robotics, and the HSE University Best Lecturer Award, in 2018, 2019, 2021, and 2024.