A Multi-Agent-Based Network-on-Chip Simulator

Gustavo L. Lima¹, Diana F. Adamatti², Eduardo W. Brião³,
Cristina Meinhardt^{2,4} and Odorico M. Mendizabal^{2,4}

¹ Universidade Federal de Pelotas – UFPEL, PPGC, Pelotas, Brazil

² Universidade Federal de Rio Grande – FURG, PPGComp, Rio Grande, Brazil

³ Instituto Federal do Rio Grande do Sul – Campus Rio Grande (IFRS), Rio Grande, Brazil

⁴ Universidade Federal de Santa Catarina – UFSC, PPGCC, Florianópolis, Brazil

e-mail: gustavolameirao@inf.ufpel.edu.br

Abstract— Integrated systems have incorporated a variety of functionalities within the same chip, requiring on-chip communication mainly based on the Network-on-Chip (NoC) design paradigm. Complex systems may impose strict requirements concerning performance, power and reliability. Thus, the choice of proper routing algorithms to the on-chip communication for NoCs merits special attention. To assist designers to adjust communication parameters, implement routing algorithms, and check system correctness, simulators become powerful and cost-effective solutions. This article explore Multi-Agent System to develop a high level abstraction NoC simulation environment. The simulator allows designers to evaluate routing algorithms, test alternative configurations and message formats. The simulation provides important measurements such as rate of utilization of routers, network contention and delay in sending messages. As a case study, the XY and WF routing algorithms were modeled and analyzed. Results highlight the pros and cons of each algorithm, and outline the simulator overall performance.

Index Terms— Simulation; Network-on-Chip; Routing algorithms; MPSoC; Multi-agent Systems.

I. INTRODUCTION

Nanotechnology advances allow designers to create fully functional systems, requiring on-chip communication mainly based on the Network-on-Chip (NoC)[1] design paradigm, being used in high performance architectures, IoT, and embedded systems in general. NoCs resemble the traditional networking solutions, where packets are sent through the network following routing policies. A variety of topologies and routing algorithms can be chosen to better meet performance or reliability requirements. Both traditional networks and NoCs provide good flexibility to adapt to general communication patterns, but NoCs impose restrictions about size, especially for memories used in the routers. Therefore, routing algorithms for NoCs must be simple and efficient in the overall memory usage to avoid traffic contention [2].

Simulators become powerful and cost-effective solutions to assist NoC designers. By running specific scenarios, these tools help to determine channel and buffer sizing, project routing and switching techniques, assess communication performance, specify packet format, and avoid starvation caused by deadlocks. NoC simulation tools have been adopted academically and by the industry in the last years [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]. However, most of these tools present a complex interface and require extra runtime and knowledge when explored together with hardware simulation tools. There is a lack on the literature of user-

friendly NoC simulators, that abstract intrinsic hardware details during the NoC design and configuration.

This article explores Multi-Agent System (MAS) to develop an NoC simulation environment. The MAS similarity with the intrinsic NoC behavior brings good agreement between the simulations and the expected behavior of the routing algorithms. Besides presenting the first NoC simulator based on MAS, the main contributions of this work include: (i) providing a high level of abstraction and hardware-independent simulator of routing algorithms for NoCs; (ii) enabling the fast and easy implementation of new algorithms in a modular platform. The high-level abstraction allows a fast validation of the routing algorithms. The modular implementation of the simulator will enable users to easily insert or adapt the current modules to the detailed application targets. The adaptive platform offered by the simulator facilitates more complex evaluation metrics to be added according to the abstraction level desired by the users.

The proposed simulator addresses routing algorithms for NoCs, enabling the implementation and test of new algorithms with a high level of abstraction and hardware independence. Simulation output shows important results such as the rate of utilization of routers, network contention and delay in sending messages. These metrics help analysts take design decisions and balance tradeoffs. As a case study, the XY and *West-First* (WF) algorithms [2] are compared using a wide selection of configurations. Results highlight the pros and cons of each algorithm. For instance, the simulator indicated a better use of spare routers by the WF algorithm, which is expected due to its partially adaptive behavior.

In [16], a first version of the simulator was proposed and some preliminary results were discussed. This work extends the previous one by enhancing the simulator with extra load generation patterns, modular design and implementation of routing algorithms, and new output metrics for analysis, such as the saturation rate. In addition, this article presents implementation details of the simulator and provides a comprehensive assessment of NoCs usage under different settings and workloads.

The rest of this work is structure as follow. Section II. reviews other NoC simulators found in the literature. Section III. presents the NoC simulator and the adoption of MAS on its development. Section IV. evaluates the simulator and presents a detailed analysis of traditional NoC routing algorithms. Section V. concludes the article.

II. NOC SIMULATORS

NoC intercommunication adds complexity to the hardware design. Thus, cost-effective tools are required to support the design, configuration and evaluation of NoCs, especially in the early stages of the developing cycle. Simulators have demonstrated suitable to evaluate traffic, reliability and energy consumption of MPSoCs. This section gives emphasis to the simulators intended for routing analysis. For a comprehensive survey on existent NoCs simulators the reader is referred to reports in [17, 18, 3, 19].

In [18] is presented a review of frameworks and simulators proposed to predicting the performance of NoC-based multicore systems, covering traffic models, latency evaluation, and the design challenges for NoC simulators. The study highlights that the evaluation can be speed-up when NoC simulators abstract the router components in a higher-level language providing cycle- and flit-accurate assessments of the target design. Compared to hardware description-based NoC-simulators, more flexible data types and structures can be adopted in these models. Therefore, many router components such as buffers and allocators can be implemented more efficiently. Another feature of these simulators is the parameterized design, which allows the users to explore over a large design space such as the buffer sizes, link bandwidth, routing schemes without recompiling the simulators.

We also observe in the literature frameworks proposed to address three main requirements: (i) automated NoC generation; (ii) automated production of NoC-IP core interfaces; (iii) analysis of NoC traffic parameters. These approaches are system-level simulations where the evaluations are based on hardware description to provide more accurate simulations by implementing every component, with the drawback of the simulation runtime. MAIA framework intends to solve all these three points and has been used to prototype SoCs in FPGAs [20]. The traffic generation and analysis provided in the MAIA framework are similar to our developed simulator in the leading goals. However, our simulator presents a higher level of abstraction, removing the hardware synthesis steps to accelerate a behavioral evaluation of the routing algorithms.

Noxim [5] allows the configuration of input parameters, such as network size, buffers, and packets. It returns as output latency statistics and power consumption. The simulator proposed in this work provides similar parameters settings. In both of them the analyst can configure the number of nodes, and size of buffers and messages. Regarding traffic generation, our tool offers two modes of packets injection: the manual sending of packets, suitable for interactive experimentation; and random mode, where the senders are evenly distributed throughout the NoC. Noxim follows a discrete probability distribution to the injection of packets.

Like Noxim, Booksim [21] allows the configuration of number of nodes, topology, routing algorithm and load generation. A similarity of Booksim with the proposed simulator is the possibility of associating costs to the actions in the routers given in number of cycles. Thus, it is possible to calculate route packets transmission costs.

DARSIM [6] stands out from other simulators for the large set of load patterns available. Users can set up typical work-

loads such as transpose, bit-complement, and H.264 decoder profiles. In addition, DARSIM can play simulation traffics defined *a priori*. The simulator proposed in this work is able to generate messages defined manually or provided by origin/destination pairs randomly chosen. In our tool, the maximum distance between origin and destination nodes is also configurable. DARSIM adopts distinct ports for input and output with each neighbor, where each port has separate buffers. In our simulator, the buffer is shared between all the packets the router receives, and there is only one bidirectional channel with each neighbor router.

HNOCS [14] is a modular open-source simulator for heterogeneous NoCs based on OMNeT++, a framework for NoC modeling. HNOCS allows the setup of variable link capacities and the number of virtual channels per port. It informs statistical measurements such as end-to-end latencies, throughput, and transfer latencies. The PANE simulator [15] also provides a modeling and system analysis tool for asynchronous, synchronous, and mixed synchronous-asynchronous (heterogeneous), but with good improvements compared with HNOCS [14]. For example, PANE allows users to configure the NoC with delays for each functional unit. Arbitration in asynchronous models can also take an unequal amount of time depending on when the requests arrive. In addition, the authors report that HNOCS may consume up to four times more memory than PANE to simulate a 4×4 mesh NoC with uniform traffic. Both HNOCS and PANE are promising to be open-source system-level NoC simulators. As in our proposal, HNOCS and PANE are designed for extensibility. However, a difference is that our simulator also implements an interactive mode, allowing users to inspect and change parameters at runtime.

Alternative approaches were also considered to evaluate NoCs. For instance, the *Network Simulator* (NS-2) is originally developed to simulate computer networks. Under certain assumptions, in [22], it was used to represent and analyze NoCs. NS-2 is an open-source tool developed in C++. It implements a discrete event-driven simulator, where its modularity and extensibility have facilitated its adoption by researching communities. However, the workaround to translate NoCs into NS-2 components discourage its use.

Noxim, Booksim, DARSIM, and PANE lack from a graphical interface. Most of parameters are defined through input files and simulation results are summarized by an output report. Differently from the aforementioned simulators, our NoC simulator provides a graphical interface that makes the tool easy to use. Input parameters are configured through sliders, switches, and buttons, which are intuitive. The graphical interface also allows users to check the status of NoC routers at run-time. An iterative mode enables users to perform manual actions, such as sending targeted messages, choosing the source and destination routers. Besides providing a report at the end of the execution, our tool can be configured to depict state of routers on-the-fly. Users can stop, resume, or even modify parameters during the simulation process. Therefore, the proposed simulator provides simple and versatile means to model and evaluate routing algorithms on NoCs.

III. THE MAS-BASED NOC SIMULATOR

The NoC simulator proposed in this work provides a high level of abstraction, avoiding details of the physical synthesis or hardware prototypes and providing a fast alternative to explore different routing algorithms. Most of existent NoC simulators require the installation of additional *Application Programming Interface* (API) and library dependencies. In contrast, the proposed simulator does not depend on external libraries, being limited only to the installation of NetLogo. The developed tool is open source and is available at [23].

The graphical interface facilitates the configuration of parameters and provides detailed information about routers and traffic of packets. The interface offers a wide range of configurable settings to model NoCs and their routing algorithms. For instance, users can set up the network size, packet size, routing algorithm and traffic generation pattern. During a simulation, the internal state of routers, traffic measurements and general statistics are shown, with the possibility to interrupt and resume execution. This allows you to observe momentary behaviors of the network as a whole, or of internal states of routers. At the end of a simulation, raw data with simulation results and a comprehensive report are made available.

Multi-Agent Systems models the behavior of autonomous artificial agents working collectively for some purpose. In general, a MAS is characterized by a set of autonomous agents that can reason, react, and interact with each other and with the environment [24]. An agent can be defined as an autonomous entity, being able to act on its own, performing actions within its knowledge to achieve goals. Agents can interact with each other via message passing, perceiving changes or actively changing the environment. Agents may have individual or collective goals, and to achieve these goals, they make use of their resources. As agents are usually analyzed in sets, forming multi-agent systems, they have a partial representation of the environment.

Table I draws parallels between agents of a SMA and routers of a NoC. While agents are virtual, routers are real entities. An agent is inserted in a simulated environment; in the router's context, the environment is the NoC itself. Agents communicate with other agents; routers communicate by message passing with other routers through their physical interconnections. An agent has goals and satisfactions, and a router forwards messages to other routers to reach a destination node. An agent has its own resources, just like a router, which maintains input and output ports, memory buffers, and a processing unit. Each agent perceives only a partial representation of the environment, and a router is capable of detecting congestion with directly connected neighbors.

A. NoC Simulator Design

The simulator adopts a 2D mesh topology. Agents are organized in the grid apart from each other by a unit of measure. Then, the links between an agent and its neighbors are created within a distance of 1. The mesh forms a $N \times N$ matrix, where N is an input value. The type of packet switching used in the simulator is the wormhole.

Figure 1 shows the structure of a router. Routers communicate each other through ports interconnected by a crossbar.

There are 4 input and 4 output ports: *North*, *South*, *East* and *West*; and an input and output ports for the *Processing Unit*, aka *Processing Element* (PE). Buffers are implemented as bounded lists, with size defined by the user. Flits are the atomic units that form packets. In most cases, a flit corresponds to a *phit* (physical unit), which is the minimum amount of data transmitted in one link transaction. In this case, the flit width matches the width of the channel [25]. Only one flit is removed or added at a time and buffer access follows a FIFO order.

Fig. 1 Router internal structure.

Routers store usage information of its ports, which are checked whenever a flit transferred from a port to another. When a flit is being sent, the crossbar connects the input buffer from the source port with the output buffer of the destination port. Once a port is allocated to a packet, other packets are unable to access the port until it is released. The simulator uses a credit-based flow control to implement the port allocation. When two or more ports compete for the same resource, the arbiter determines which port has the priority in each execution cycle in a *Round Robin* fashion.

At each cycle, all routers execute as an atomic step. The flits generated by the message generator are transferred to the PE input buffer. On each router with messages in transit, a destination port is chosen for the next flit, according to the routing algorithm. To process a port, it is checked whether the buffer of that port contains a flit. If a flit exists, it is checked whether the destination buffer has enough space and the flit has access to the destination port. If both conditions are met, the flit is routed to its destination. Otherwise, it remains in the buffer and awaits for the next cycle. Once all router ports are processed, simulation metrics are calculated and a new cycle begins, unless the last simulation cycle is reached.

B. Implementation Details and Internal Structures

The simulator provides extensible modules to implement routing algorithms, arbiter, and traffic generation. Customized modules can be easily added to the simulator. In addition to the code information, it is also important to note the interface of the tool. To illustrate this interface, Figure 2 shows a screenshot of the simulator interface during an execution. The left-hand side exhibits the buttons for configuring the NoC and simulation parameters. Users can also

Table I. Similarities between agents and routers.

Agent	Router
Virtual entity	Real entity
Environment	Network on Chip
Communicate with other agents	Send messages to destination nodes
Manage its own resources	Manage its buffers and processing unit
Have a partial representation of the environment	Messages are sent to the routers directly interconnected

configure logging settings, simulation size, and running controls. and some types of packet send. At the middle of the screen are available metrics displays, Metrics such as the hop count and latency, as well as traffic generation settings are displayed and updated on-the-fly. The simulator exhibits a grid with NoC routers. Different colors are used to indicate the level of use for each router. colored according to the most used port at the moment. Further details about the output metrics are discussed in Section D.

The main parts of the simulator are:

Packets: NoCs communication protocols split messages into packets to improve bandwidth usage despite the application workload. In addition to the payload, packets piggyback information indicating the beginning and end of a message, as well as the destination node address. In the wormhole packet switching, each packet consists of at least 3 flits: the *header*, one or more flits containing payload chunks, and the *tail*. When a router receives a flit header, the recipient's information is extracted and the routing algorithm determines which port the packet flits must follow to reach the destination. Headers are composed by: *packet identification*, *flit type* and *destination address*. The packet identification indicates the packet that the flit belongs to, and it is used by routing algorithms to ensure that flits with the same identifier correspond to chunks of the same packet. The *flit type* contains a code for each flit type (*header*, *body* or *tail*). The *destination address* informs a coordinate (x, y) of the destination router. According to the header information, routers reserve ports for a packet and update the routing table. Ports are released upon receipt of *tail* flits;

Cycles: Simulation time is given by cycles. Once per cycle, each router executes, measurements are updated, and the cycle counter is increased. Time markers and number of hops are piggyback on the flits during the simulation. Thus, for each flit, the cycle when the packet arrived at the router, the hop count, and the cycle in which the flit was created, are tracked. The simulator measures the communication latency by computing the number of cycles took to the flits reach their destination.

Arbiter: The arbiter is responsible for deciding which packet has the priority to access a destination port. To guarantee fairness, the arbiter implemented in the simulator follows a Round-Robin schema, changing the priority every cycle. The arbiter behavior can be configured before simulation starts. Other arbitration policies might be easily added to the simulator due to its modular design.

Routing: In the simulator, routing is divided into two parts: algorithm and routing table. The routing algorithm implements the routing strategy, while the routing table is an auxiliary structure used by a variety of routing algorithms.

Upon receipt of a flit *header*, the router extracts the destination coordinate (x, y) , and executes the routing algorithm to determine an output port. The decision is recorded into the routing table to avoid re-execution of the routing algorithm for the upcoming flits from the same packet. These steps are performed by every router throughout the packet route. This way, the *header* makes room for the body flits, drawing a communication path. When a router receives a flit *body*, it checks if the output port is reserved to that flit. If it is, the flit is forwarded to that port; otherwise, it is kept in the port buffer and waits until the port is available. When a flit *tail* is received, the router terminates allocations made to that packet, i.e., the packet information is removed from the routing table and output port used in the communication is released. It is worth noting that whether a flit is allowed to move either from one internal buffer to another, or to another router's buffer, it is first checked if there is enough space in the destination buffer. If the destination buffer is full, the flit remains in the same position and awaits for the next cycle. Similarly to the arbiter module, other routing algorithms can be added to the simulator.

Traffic Generator: This module is responsible for generating packets. There are options for both manual and random traffic generation. For manual traffic, users choose source and destination routers through the simulator interface. Random traffic generation can be oriented or random. In the former, users explicitly indicate source and destination pairs, and one of these pairs is chosen at random for each packet generated. In random traffic, the simulator randomly chooses a source and a destination router within a radial distance from the source router determined by a message radius. Figure 3 illustrates possible destination routers for a source router in the center of the network. Figures 3(a) and 3(b) represent message radius with values 1 and 2, respectively. In the current version of the simulator, senders are evenly distributed throughout the NoC following an uniform distribution. However, other load patterns may be coupled to the traffic generator.

C. Input Parameters

The main input parameters available on the simulator are:

Mesh size: the number of routers in the NoC. It is configured by setting row and column length, forming a 2D grid.

Buffers size: the size of buffers associated to input and output ports. The size is given in number of flits the buffer is capable to store.

Arbitration: the type of arbiter used in the simulation.

Routing algorithm: the routing algorithm to be used. In the current version the XY and WF (West-First) algorithms are available.

Fig. 2 Simulator graphical interface.

Fig. 3: Random traffic generation based on radial distance from a source router.

Routing Cost: determines how many cycles it takes for the routing algorithm calculates a route. By choosing the value 0, the routing is calculated instantaneously, causing the routing cost to be disregarded. More realistic scenarios depend on NoCs specifications.

Transmission cost: counts the number of cycles taken to a flit be transferred to the input buffer in the destination router.

Packet size: determines the number of flits of each packet.

Traffic type: defines the traffic type. When random traffic is set, the message radius must be informed. To configure the traffic generator, the user informs the frequency in which new packets are generated as well as the number of packets created by generation.

Simulation size: the duration of a simulation given in number of cycles.

Output log: when this option is selected, the simulator records detailed information about the state buffers for all routers at specified cycles interval.

D. Output Metrics

Some output metrics are presented at run-time, while others are logged into a file. The most important metrics are:

Packets and Flits accounting: measures the number of packets and flits created and received. The difference between them determines how many have not reached the destination yet.

Execution time: reports the simulation time in seconds.

Hops count: measures the minimum, average and maximum number of hops counted in the packets route. A hop is counted when a flit moves from one router to another.

Delay count: measures the minimum, average and maximum number of cycles since the insertion of flits in the NoC until they reach the destination.

Coloring Occupancy Rate: indicates the routers' buffers occupancy. For each cycle, buffers usage is calculated and the routers receive a color corresponding to their level of occupation, as detailed in Table II. The simulator provides a visual output of router usage, coloring routers. Furthermore, an output file records the buffer occupancy rate of each buffer at specified cycles interval.

Table II. Color table to represent routers occupancy rate.

Color	Occupancy rate
White	0% of buffer capacity is occupied
Blue	1 $\approx$ 24% of buffer capacity is occupied
Green	25 $\approx$ 49% of buffer capacity is occupied
Yellow	50 $\approx$ 74% of buffer capacity is occupied
Red	75 $\approx$ 99% of buffer capacity is occupied
Black	100% of buffer capacity is occupied

IV. SIMULATOR VALIDATION

This section highlights the routing algorithm influence on NoCs performance. Results reveal the practicality of using

the simulator and its versatility for analysis. The XY and West-First (WF) algorithms are evaluated [2]. In XY algorithm, packets are routed in the x direction until reach the x target, then they follow the y direction until reach their destination [2]. The WF algorithm is partially adaptive, being able to choose less congested routes [2]. If the x target destination is smaller than the x position of the current router, routing follows to the left regardless the y position of the destination. Thus, packets can be forwarded to the east, north, or south directions, depending on the network traffic.

Three groups of tests were devised. The first group aims to verify the latency impact for different volumes of load. The second group explores output data from simulation logs to perform routers analysis over time. The third group evaluates the simulator performance. All simulations were executed in a computer with a processor with 4 cores and 4 threads at 3.1GHz, and 8 GB of RAM. It was used the Windows 10 OS and NetLogo 6.0.4. While some parameters vary according to the test case, others remain the same for every experiment and they are described in Table IV. The first 1,000 cycles of each execution were disregarded, as they represent the warm up period.

Table IV. Common parameters adopted in the experiments.

Parameter	Settings
Mesh size	3×3
Input/Output buffer size	8 flits
Routing cost	0 cycles
Flits transferring cost	0 cycles
Duration of simulation	10,000 cycles
Arbiter	Round-Robin

Fig. 4: Load generated through oriented traffic: (a) one source/destination pair, (b) three source/destination pairs without route intersection, (c) three source/destination pairs with route intersection.

A. Group 1: Evaluation of latency and network contention

These experiments explore congestion-free scenarios and gradually increase the workload to induce congestion in some routes. Three scenarios were derived, varying the number of source and destination pairs in the network, as well as the load intensity. Figure 4 illustrates the arrangement of routers in the network. Source routers are marked in gray and destination routers are marked in black. The resulting path by applying the XY algorithm is given by arrows directions. Scenario 1 is a simple case where only three routers are needed to route packets from nodes 0 to 6. Figure 4(b) illustrates the scenario 2, where three independent routes transfer packets from nodes 0 to 6, 1 to 7, and 2 to 8. Finally, Figure 4(c) shows the scenario 3, where three routes transfer packets from nodes 1, 3, and 5 to node 7. Different from the scenario 2, in this case all routes have a common router (node 4) along their paths.

For each of these scenarios, three load patterns were applied. All adopt 3 flits per packet, but they differ in the number of cycles to generate new packets. Packets are generated every 3 cycles to represent low load (workload A), 2 cycles to moderate load (workload B) and 1 cycle to heavy load (workload C). The combination of workloads and scenarios described is presented in Table III. As expected, in congestion-free routes the latency is affected only by the distance between source and destination routers. Under congestion, latency is also impacted by the paths where packets wait for accessing ports. By applying higher loads, both buffers occupancy and latency rates increase.

The minimum, average and maximum number of hops remains the same in all cases, once the XY routing algorithm sets a fixed and deterministic route for any pair of nodes. Therefore, packets are always forwarded through 2 nodes to reach the destination, with a minimum latency equals to 3 for all cases. This value corresponds to the arrival of the flit

Table III. Results for experiments of Group 1.

Workload.Scenario	Hops			Latency			Flits	
	min.	avg.	max.	min.	avg.	max.	delivered	dropped
A.1	2	2	2	3	4	5	9995	0
A.2	2	2	2	3	4	5	9995	0
A.3	2	2	2	3	4	5	9995	0
B.1	2	2	2	3	9	9	9996	4986
B.2	2	2	2	3	5	9	14992	0
B.3	2	2	2	3	49	88	9996	4935
C.1	2	2	2	3	9	9	9996	19986
C.2	2	2	2	3	8	9	28185	1785
C.3	2	2	2	3	49.9	88	9996	19929

head of a packet, two transmission cycles between routers plus one cycle for delivering the flit to the PE at the destination. Other flits (body and tail) can just be delivered after their predecessors flits. Thus, the average and maximum latency vary according to the buffer occupancy, but are never smaller than the latency to deliver the flit header plus one. In workload A, where 1 packet is generate every 3 cycles, there is no saturation in the network, so all flits are delivered.

Patterns B and C increase the frequency of packets generated. The impacts are perceptible in the number of flits delivered at the end of the simulation and the increasing in the average and maximum latency. In Scenario 1 there is a single route, so the higher latency derives exclusively from high buffers occupancy. Heavy buffer usage incurs a increasingly number of dropped flits, especially with workload C. Scenario 2 draws three independent routes, which provides a balanced access to the network routers. When compared to Scenario 3, where a central router takes part in three distinct routes, Scenario 2 exhibits lower latencies and less dropped flits.

B. Group 2: Assessment of occupancy and saturation rates

These experiments evaluate the buffer usage and congestion for NoCs with different sizes. The following network sizes are created: 5×5 , 10×10 and 15×15 . Radial distance between source and destination are configured to simulate messages exchanged exclusively by neighbors (using *small radius*), and by routers close one another (*average radius*). The load intensity is modeled in a way that 10% and 20% of nodes send packets at each load generation cycle to mimic light and high load patterns, respectively. Table V summarizes the radial distance between communicating routers and the amount of packets created per cycle for all scenarios of group 2. Other parameters were configured according to the settings presented in Table IV.

Table V. Parameters settings for experiments of Group 2.

Scenario	Radius		Packets created per cycle	
	Small	Medium	Low load	High load
5×5	1	4	3	5
10×10	1	8	10	20
15×15	1	11	23	45

Two figure of merits are adopted to analyze the state of internal buffers and congestion: the *occupancy rate* and the *saturation rate*, as stated in Equations 1 and 2. The occupancy rate measures the average number of flits stored in each port from the router's input buffers for the whole simulation. The saturation rate calculates the average number of flits stored in the buffer ports with the highest utilization for the whole simulation.

$$occupancy_rate = \frac{\sum_{n=1}^{simulation_size} I_N + I_E + I_S + I_W}{simulation_size \times input_buffer_size \times 4} \quad (1)$$

$$saturation_rate = \frac{\sum_{n=1}^{simulation_size} \max(I_N, I_E, I_S, I_W)}{simulation_size \times input_buffer_size} \quad (2)$$

For each scenario described in Table V, 5 repetitions were performed, and presented the same results. That suggests the simulation time and the warm up period were properly configured. Individual measurements for North, East, South, and West ports of each router were extracted from the logs generated by the simulator. These files contain the number of flits stored in each buffer for each simulation cycle. Thus, calculating occupancy and saturation rates is straightforward.

To illustrate the expressiveness of the occupancy and saturation rates, heat maps for both rates were generated. These maps show the intensity of both rates in each router, where darker and lighter coloration means high and low rates, respectively. The x and y axis indicate the position of routers in the network, given by their X and Y coordinates.

Figure 5 depicts the heat map of the 5×5 NoC occupancy rate, varying the radial distance, and applying low and high workloads (see Table V for details). When radius is equal to 1, represented by Figures 5(a) and 5(b), messages are only exchanged between neighboring routers. As observed, this causes an homogeneous use of the network, presenting a low utilization for all the routers. As the radius increases, regions with concentration of use can be observed, especially in more intense traffic, as can be observed in Figure 5(d).

Fig. 5: Routers occupancy rate for a 5×5 mesh with the XY routing algorithm.

Figure 6 shows the saturation rate for the same scenarios of Figure 5. Although the occupancy and saturation rates look similarly at first glance, there are spots with high saturation, especially when workload is high. For example, the router located in the coordinate (3,2) in Figure 6(d) has not indicated high occupancy rate during the simulation, even though some of its ports were highly demanded by the routing algorithm.

To further analyze the saturation of the router indicated by the coordinate (3,2), the average number of flits in every port

Fig. 6: Routers saturation rate for a 5×5 mesh with the XY routing algorithm.

buffer is calculated showing that port *West* buffer stored 7.5 flits in average during the execution, while other ports stored less than 1 flit in their buffers. Figure 7 exhibits the state of router's input ports for simulation window from cycle 5700 to 5900. Port *West* was the one that had the highest usage, being completely occupied most of the time. While a majority of the traffic comes from west, a low traffic comes from east, south, and north, contributing to the moderate or low occupancy rate. This behavior might be expected since the router is located near to the right border and it is vertically centralized. The XY routing algorithm does not take into account the state of buffers to define routes, then saturation in the intermediate routers can be observed.

Fig. 7 State of ports of the router given by the coordinate (3,2).

Figures 8 and 9 show the heat map graphs for the occupancy and saturation rates of a 10×10 NoC, using radial distances of 1, and 8, with low and high loads (see Table V for the complete setup). Compared to the previous results, it is possible to observe that, with the increase of the network size, routers located in central locations are the most requested. This is expected with the use of XY routing algorithm.

In order to compare different routing algorithms, we implemented the XY and WF algorithms and the traffic congestion and routers state for the scenarios described in Table V are evaluated considering both of them. Figures 10 and 11 show the occupancy rate when algorithms XY and WF are used, respectively. They consider a 15×15 NoC, radial dis-

Fig. 8: Routers occupancy rate for a 10×10 mesh with the XY routing algorithm.

Fig. 9: Routers saturation rate for a 10×10 mesh with the XY routing algorithm.

tances of 1, and 11, with low and high loads. Although both figures indicate a higher load in routers placed in the center, the flits concentration when using WF is slightly shifted to the left. Furthermore, the adaptive behavior of WF algorithm reduces overloaded routers. The coloring pattern for the WF algorithm is more homogeneous than the XY. Notice the better usage of routers near to the borders (see the difference in the occupancy rate on the top, bottom, and corner regions in these figures).

Figures 12 and 13 show the saturation rate for the previous experiments. Independently of the routing algorithm, there is a greater saturation of routers located at the center of the NoC. However, similar to the occupancy rate observation, when using WF algorithm, the load is slightly shifted to the left. In general, there was observed an homogeneous use of routers when packets are exchanged by neighbors. Despite the algorithm in use, as the number of hops between source and destination routers increases, it is possible to observe the high occupancy of the routers located in the central area. This is expected since packet destinations are further away from the source, obligating flits to travel through the entire network.

C. Group 3: Simulator Performance

These tests aim to evaluate how the simulation time may be affected by network size, refreshing of statistics and col-

Fig. 10: Routers occupancy rate for a 15×15 mesh with the XY routing algorithm.

Fig. 11: Routers occupancy rate for a 15×15 mesh with the WF routing algorithm.

Fig. 12: Routers saturation rate for a 15×15 mesh with the XY routing algorithm.

oring at the graphical pane, and logging of measurements. To observe the impact of network size, the number of routers is gradually increased, running scenarios with 9 routers (3×3), 25 routers (5×5), 49 routers (7×7), 100 routers (10×10), 225 routers (15×15), 400 routers routers (20×20), 625 routers (25×25) and 900 routers (30×30). Traffic generation is random and creates 10 packets per cycle. The radial distance was set to 10.

Fig. 13: Routers saturation rate for a 15×15 mesh with the WF routing algorithm.

Figure 14 depicts the simulation time according to the network size and variations on the logging description level and refreshing of the graphical interface. Four different settings were tested: (i) simulator updates the graphical interface, but do not generate the detailed log. That means users can check metrics captured at run time, but do not have access to the log of routers states over time; simulator does not update the graphical interface, but logs routers information every (ii) 1 cycle, (iii) 10 cycles, and (iv) 50 cycles. Although cases (iii) and (iv) shrink the log size and the reduce the number of records to be analyzed, they could impact the accuracy of the analysis.

Fig. 14: Simulation time according to the the network size, graphical refreshing and logging level.

As expected, the simulation time increases as routers are added to the network. Furthermore, it can be observed that the refreshing of the graphical interface panel has the greatest impact in the simulation time. For instance, a simulation of network with 900 routers with graphical update enabled can take 5 hours, while the same simulation without graphical refreshing takes less than 1.5 hours. The frequency of logging has little impact in the simulation time.

The use of a multi-agent platform for the development of the NoC simulator demonstrate a good performance when compared to the performance of state-of-the-art high-level abstraction simulators. For instance, in [15] scenarios simi-

lar to ours were used to evaluate NoCs with mesh topologies and different sizes using the PANE simulator. Test scenarios were configured with one-flit-long packets injected at 0.05 packets/router and the simulation took 235 minutes, i.e., ≈ 4 hours, to execute a 32×32 NoC. Our 30×30 test case took less than 1.5 hours to execute with the graphical refreshing disabled. Despite the good performance observed with our simulator, it is worth noting that the comparison between PANE performance and our simulator is based on similar, but not identical, scenarios. Furthermore, the tests were conducted in experimental environments with different configurations. Concerning system-level approaches, the simulation is more time-consuming due to the number of details considered in the component's description. According to [26], a single simulation scenario may easily take several hours, even for small networks (e.g., 4×4 NoC-based MP-SoCs), and it can be unfeasible for larger scenarios. Some approaches, such as the one proposed in [20] can drastically reduce the execution time of system-level simulations. In [27], authors report a reduction from 2 hours to 5 minutes to simulate a 4×4 NoC-based MPSoCs. However, this type of simulation is still more expensive than high-level abstraction simulations, as expected.

V. CONCLUSION

This work presented a NoC simulator developed as a Multi-Agent System. The simulator provides a high degree of abstraction, with hardware independence and easy of use. Thus, a vast set of NoCs can be modeled and analyzed under different settings and workloads. The simulator is designed to facilitate the addition of new routing algorithms and enables the traffic analysis for each routing strategy. Packets latency and network queuing can be investigated in terms of routers occupancy and saturation, with the support of output metrics returned by simulator.

To demonstrate the potential of the simulator, this article implemented two traditional routing algorithms for NoCs as a case study, the XY and West-First (WF). Different scenarios concerning the NoC capacities and workload were used to evaluate the pros and cons of each algorithm. It was possible to observe how the workload impacts latency, packet loss, occupancy and saturation of routers under different NoC configurations. When comparing XY and WF algorithms, results demonstrated the advantages of the WF algorithm, which are consequence of its partially adaptive behavior. From the case study observation, the simulator allows the easily and quickly introduction of new routing algorithms for NoCs and reconfiguration of the NoC architecture, providing a powerful environment for design space exploration and finding the best routing solution for particular NoC applications. Also, the Multi-Agent System demonstrated a powerful approach to represent NoCs.

ACKNOWLEDGMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) - Finance Code 001, and Fundação de Amparo à pesquisa do Estado do RS (FAPERGS).

REFERENCES

- [1] S. Kumar, A. Jantsch, J. . Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani, "A network on chip architecture and design methodology," in *Proceedings IEEE Computer Society Annual Symposium on VLSI. New Paradigms for VLSI Systems Design. ISVLSI 2002*, 2002, pp. 117–124. [Online]. Available: <http://dx.doi.org/10.1109/ISVLSI.2002.1016885>
- [2] M. Palesi and M. Daneshmand, *Routing algorithms in Networks-on-Chip*. Springer, 2014. [Online]. Available: <http://dx.doi.org/10.1007/978-1-4614-8274-1>
- [3] M. Alalaki and M. Agyeman, "A Study of Recent Contribution on Simulation Tools for Network-on-Chip (NoC)," *International Journal of Computer Systems*, vol. 11, no. 4, 2017. [Online]. Available: <http://dx.doi.org/10.5281/zenodo.1130379>
- [4] R. Al-Badi, M. Al-Riyami, and N. Alzeidi, "A parameterized NoC simulator using OMNet++," in *2009 International Conference on Ultra Modern Telecommunications & Workshops*. IEEE, 2009, pp. 1–7. [Online]. Available: <http://dx.doi.org/10.1109/ICUMT.2009.5345363>
- [5] V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti, "Noxim: An open, extensible and cycle-accurate Network on Chip simulator," in *2015 IEEE 26th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. IEEE, 2015, pp. 162–163. [Online]. Available: <http://dx.doi.org/10.1109/ASAP.2015.7245728>
- [6] M. Lis, K. S. Shim, M. H. Cho, P. Ren, O. Khan, and S. Devadas, "DARSIM: a parallel cycle-level NoC simulator," in *MoBS 2010-Sixth Annual Workshop on Modeling, Benchmarking and Simulation*, 2010. [Online]. Available: <http://hdl.handle.net/1721.1/59832>
- [7] S. D. Chawade, M. A. Gaikwad, and R. M. Patrikar, "Review of XY routing algorithm for Network-on-Chip architecture," *International Journal of Computer Applications*, vol. 43, no. 21, pp. 975–8887, 2012. [Online]. Available: <http://dx.doi.org/10.5120/6387-8691>
- [8] H. M. Kamali, K. Z. Azar, and S. Hessabi, "Ducnoc: A high-throughput FPGA-based NoC simulator using dual-clock lightweight router micro-architecture," *IEEE Transactions on Computers*, vol. 67, no. 2, pp. 208–221, 2017. [Online]. Available: <http://dx.doi.org/10.1109/TC.2017.2735399>
- [9] L. Liu, Z. Zhu, D. Zhou, and Y. Yang, "A fair arbitration for network-on-chip routing with odd-even turn model," *Microelectronics Journal*, vol. 64, pp. 1–8, 2017. [Online]. Available: <https://doi.org/10.1016/j.mejo.2017.04.002>
- [10] G. Oxman and S. Weiss, "An NoC Simulator That Supports Deflection Routing, GPU/CPU Integration, and Co-Simulation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 10, pp. 1667–1680, 2016. [Online]. Available: <http://dx.doi.org/10.1109/TCAD.2016.2527698>
- [11] D. Ghosh, P. Ghosal, and S. P. Mohanty, "A highly parameterizable simulator for performance analysis of NoC architectures," in *2014 International Conference on Information Technology*. IEEE, 2014, pp. 311–315. [Online]. Available: <http://dx.doi.org/10.1109/ICIT.2014.66>
- [12] M. Zolghadr, K. Mirhosseini, S. Gorgin, and A. Nayebi, "GPU-based NoC simulator," in *Ninth ACM/IEEE International Conference on Formal Methods and Models for Codesign (MEMPCODE2011)*. IEEE, 2011, pp. 83–88. [Online]. Available: <http://dx.doi.org/10.1109/MEMCOD.2011.5970514>
- [13] L. Liu, R. Ma, and Z. Zhu, "An encapsulated packet-selection routing for network on chip," *Microelectronics Journal*, vol. 84, pp. 96–105, 2019. [Online]. Available: <https://doi.org/10.1016/j.mejo.2019.01.002>

- [14] Y. Ben-Itzhak, E. Zahavi, I. Cidon, and A. Kolodny, "HNOCS: Modular open-source simulator for heterogeneous nocs," in *2012 International Conference on Embedded Computer Systems (SAMOS)*, 2012, pp. 51–57. [Online]. Available: <http://dx.doi.org/10.1109/SAMOS.2012.6404157>
- [15] S. N. Ved, S. Singh, and J. Mekie, "PANE: Pluggable asynchronous network-on-chip simulator," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 15, no. 1, pp. 1–27, 2019. [Online]. Available: <https://doi.org/10.1145/3241051>
- [16] G. L. Lima, N. F. Traversi, D. F. Adamatti, G. P. Dimuro, C. Meinhardt, E. W. Brião, and O. M. Mendizabal, "Exploring MAS to a High Level Abstraction NoC Simulation Environment," in *2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2018, pp. 365–368. [Online]. Available: <http://dx.doi.org/10.1109/ICECS.2018.8617979>
- [17] A. B. Achballah and S. B. Saoud, "A survey of Network-on-Chip tools," *arXiv preprint arXiv:1312.2976*, 2013. [Online]. Available: <http://dx.doi.org/10.14569/IJACSA.2013.040910>
- [18] Z. Qian, P. Bogdan, C.-Y. Tsui, and R. Marculescu, "Performance evaluation of noc-based multicore systems: From traffic analysis to noc latency modeling," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 21, no. 3, May 2016. [Online]. Available: <https://doi-org.ez46.periodicos.capes.gov.br/10.1145/2870633>
- [19] S. Khan, S. Anjum, U. A. Gulzari, and F. S. Torres, "Comparative analysis of network-on-chip simulation tools," *IET Computers & Digital Techniques*, vol. 12, no. 1, pp. 30–38, 2018. [Online]. Available: <https://doi.org/10.1049/iet-cdt.2017.0068>
- [20] L. Ost, A. Mello, J. Palma, F. Moraes, and N. Calazans, "MAIA: A framework for networks on chip generation and verification," in *Proceedings of the 2005 Asia and South Pacific Design Automation Conference*, ser. ASP-DAC '05. New York, NY, USA: Association for Computing Machinery, 2005, p. 49–52. [Online]. Available: <https://doi.org/10.1145/1120725.1120741>
- [21] 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 *2013 IEEE international symposium on performance analysis of systems and software (ISPASS)*. IEEE, 2013, pp. 86–96. [Online]. Available: <http://dx.doi.org/10.1109/ISPASS.2013.6557149>
- [22] M. Ali, M. Welzl, A. Adnan, and F. Nadeem, "Using the NS-2 network simulator for evaluating Network on Chips (NoC)," in *2006 International Conference on Emerging Technologies*. IEEE, 2006, pp. 506–512. [Online]. Available: <http://dx.doi.org/10.1109/ICCRD.2010.145>
- [23] G. L. Lima, N. F. Traversi, D. F. Adamatti, G. P. Dimuro, C. Meinhardt, E. W. Brião, and O. M. Mendizabal, "Multiagent noc simulator," <https://github.com/GustavoLLima/MultiagentNoCSimulator>, 2018.
- [24] M. Wooldridge, *An introduction to multiagent systems*. John Wiley & Sons, 2009.
- [25] É. Cota, A. de Moraes Amory, and M. S. Lubaszewski, *Reliability, Availability and Serviceability of Networks-on-Chip*. Springer Science & Business Media, 2011. [Online]. Available: <https://doi.org/10.1007/978-1-4614-0791-1>
- [26] L. Ost, G. M. Almeida, M. Mandelli, E. Wachter, S. Varyani, G. Sassatelli, L. S. Indrusiak, M. Robert, and F. Moraes, "Exploring heterogeneous noc-based mpsoCs: From fpga to high-level modeling," in *6th International Workshop on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC)*. IEEE, 2011, pp. 1–8. [Online]. Available: <https://doi.org/10.1109/ReCoSoC.2011.5981517>
- [27] C. A. Petry, E. W. Wächter, G. M. de Castilhos, F. G. Moraes, and N. L. Calazans, "A spectrum of mpsoC models for design space exploration and its use," in *2012 23rd IEEE International Symposium on Rapid System Prototyping (RSP)*. IEEE, 2012, pp. 30–35. [Online]. Available: <https://doi.org/10.1109/RSP.2012.6380687>