Trace-driven extension for Noxim simulator

Ivan Luiz Pedroso Pires, Marco Antonio Zanata Alves and Luiz Carlos Pessoa Albini

Department of Informatics – Federal University of Paraná (UFPR) – Curitiba, Brazil
E-mail: {ilppires, mazalves, albini}@inf.ufpr.br

***Abstract*—Simulation is one of the main tools used to analyze new proposals in the Network-on-Chip (NoC) field. Among these simulators for analyzing and testing new ideas in NoC architectures, the Noxim simulator stands out, being used by many researchers due to the wireless support and open-source availability. An important issue at the simulation phase is the choice of workload, as it may affect testing the system and its features. The correct workload can lead to rapid and efficient system development, while the wrong one may compromise the entire system evaluation. To ensure a more realistic simulation, simulators usually relies on real workloads by using a trace-driven approach. Although Noxim provides a simple support for input traces, it is very limited to a general behavior of the system, accepting only a generic injection rate parameter over time. We propose in this paper an extension of the Noxim simulator to address these issues. This extension enables Noxim to easily receive and process a real workload traces from real applications, considering each message sent by each processing element over the NoC. This extension was demonstrated and evaluated using the NAS-NPB workload. Our results show that Noxim still generating trusty results when using our trace extension.**

I. Introduction

NoC [1] is the current interconnection paradigm to design all large-scale chips. It is scalable and can be adapted to several computational paradigms and applicable to various areas, like high performance multiprocessing computers [2]. The NoC communication is performed through packets, divided into small information units called flits, transmitted from source to destination by routers, hubs and network interfaces over wired or wireless links [3].

When computer designers want to evaluate new interconnection architectures and organizations, simulation is one of the main tool used to analyze traffic and measure the performance, power and area of NoC. It is a suitable tool for testing and analyzing results from new concepts and ideas without hardware prototyping. To guarantee the accuracy of the results, the simulator must comply with the state-of-the-art proposals and technologies.

NoC simulators must implement the most widespread concepts, such as the clustered NoC and wireless NoC, in order to achieve high acceptance by the researchers. The clustered NoC [4] consists of a central-NoC that can be connected to one or several sub-NoCs. It may help to reduce the communication overhead and thus improve the network performance for clusters of Systems-on-Chip (SoCs) by expanding resources attaching multiple *sub-NoCs*. However, this solution requires a physical interconnection backbone in which all the devices must be connected. Wireless NoCs extends the traditional wired NoC concept to use wireless link [5], [6], [3].

To ensure a more realistic simulation, any simulator must provide ways to receive input from realistic executions, which can be made using a trace-driven approach to run real data in testing system. A trace is a log of events and usually includes: time, type, size and other important parameters associated with each event, varying depending on the type of simulation is being performed. Trace-driven simulations can be used by resource management algorithms; deadlock prevention; CPU scheduling; cache analysis; etc. Among all advantages, trace-driven simulation allows realistic workload very similar to the actual implementation. However, also has some disadvantages as the complexity and single point of validation [7].

Nowadays, there are several specific simulators for NoC interconnections such as: Noxim [8]; Booksim [9]; Naxim [10]; Hornet [11]; Topaz[12]; HNOCS [13]; NoC for OMNeT++ [14]; WNoC Simulator [15]; Darsim [16]; Netrace [17]; Garnet [18]; gpNoCsim [19]; Nirgam [20]; NNSE [21]. Other simulators non-specific for NoC could be adapted in some way: Gem5 [22]; CACTI [23]; NS-3 [24]; etc. In this work, only the specific-NoC simulator with focus on performance and execution time that were published in scientific papers were considered. Table I shows a brief summary of these simulators and depicts their ability to handle traffic input tracing, to support wireless communication and the source code availability. The combination of all these features results in a powerful simulator tool for realistic workloads to design new concepts for inter and intra-chip communication.

It can be observed that none of the previous proposed NoC simulators provides support to all these features. In this paper, we extend the Noxim Simulator to accept external input traces based on messages. The Noxim simulator was chosen as it features wired and wireless communication and it is open-source. This extension enables the Noxim simulator to receive realistic and controlled traffic traces as input, providing ways to compare real workloads in different NoC scenarios. For our evaluations, the NAS Parallel Benchmark (NPB) [25] workload was used to perform the communication in a traditional 2D-Mesh NoC over wired and wireless communication. Along the paper, we explain how we extracted the Message Passing Interface (MPI) trace in order to use such workload as input for our simulations.

TABLE I: Summary of NoC simulators.

Simulator	Trace-driven traffic	Wireless communication	Source available
Noxim'15 [8]		●	●
Booksim'13 [9]	●		●
Naxim'13 [10]			
Hornet'12 [11]	●		●
Topaz'12 [12]	●		●
HNOCS'12 [13]	●		●
NoC OMNeT++'11 [14]	●		●
WNoC Simulator'11 [15]		●	
Darsim'10 [16]	●		●
Netrace'10 [17]	●		●
Garnet'09 [18]	●		●
gpNoCsim'07 [19]			●
Nirgam'07 [20]	●		●
NNSE'05 [21]			

The rest of this paper is organized as follows: Section II contains an overview of Noxim Simulator and related proposals for Noxim extensions; Section III present the proposed modifications in the simulator to support external traces; Section IV shows the methodology and discussion of the evaluation for this work; Section V brings the final considerations and future work ideas.

II. Noxim Simulator

Noxim [8] is an open-source cycle-accurate simulator developed in C++ and integrated with System-C for heterogeneous wired and wireless NoC architectures which estimates performance and energy consumption. The simulator works with two main conceptual elements: tile nodes and communication infrastructure. Tile nodes are computational or storage nodes. The communication infrastructure consists of router(s) for each tile interconnected by wired links with their neighbors and possibly the wireless hub element. The wireless hub is wired connected with one or more tiles and wireless connected with other hubs. Therefore, the simulator offers three communication patterns: tile-to-tile, tile-to-hub and hub-to-hub. Communication is performed through packet split in small parts called flits.

The modeling of the NoC system is made in a human-readable configuration file in the YAML format. The configuration parameters are organized in three groups: the wired NoC configuration, the wireless setup and the simulation parameters. The wired NoC configuration sets the dimension of the mesh, internal buffers, routing table, routing algorithm and strategy for selection between multiple output directions. The wireless configuration sets the wireless hub and their buffers, the channels with its data rate, policy of medium control access and bit error rate. The simulation parameters set the clock period, time for reset, warm up time, flag to wireless usage, flag to debug mode, output trace mode, packet sizes, packet injection rate, probability of re-transmission and traffic distribution.

Noxim simulator was upgraded and enhanced by several authors. Originally, it supported only mesh topology, in [26], authors present an enhanced Noxim Simulator for performance evaluation of other NoC topologies. They implemented the mesh, torus and twisted torus topologies. These authors also proposed different routing algorithm to test the improvements.

In [27], the Noxim was used together with QEMU to form a hardware/software co-simulator for NoC. Each CPU core was simulated by QEMU and connected by a TCP connection with the Noxim simulator. The Noxim simulator interacted with QEMU as the processing elements were formed by CPU cores.

In this paper, we propose to extend the Noxim simulator to accept external traces formed by messages in such way that real scenarios can be more realistic emulated. Moreover, in our traces we introduce the message size parameter, which must be simulated to approximate the simulate scenario to the real life.

III. Input Trace for Noxim

This section presents our proposal to extend the Noxim simulator to support external input traces. The Noxim existing traffic table based was not modified, instead was created a new way to input trace, called traffic trace based. Moreover, a send-and-wait method was implemented within the traffic trace based to represent the time for a Processing Element (PE) receives, process and reply messages.

A. Simulator Modifications

Modifications were mainly on the YAML configuration file and inside the source code from the simulator. The YAML configuration file modifications are the following:
traffic_distribution: in the original Noxim, the traffic distribution can be set to random, transpose, hot spot, table based, bit reversal, shuffle and butterfly. We extended this parameter to add the *traffic trace* option to support external input traces.

traffic_trace_filename: this parameter informs the name of input trace without the PE identification. For each PE in the system a respective trace file must exist even if empty (in the case the PE has zero packages). For instance, a system with 4 PEs must have the following traces *000_trace.txt*, *001_trace.txt*, *002_trace.txt* and *003_trace.txt*.

traffic_trace_flit_headtail_size: this parameter informs the size of packages head and tail (summed) for the modeled network technology. This feature helps the simulator to estimate the data throughput considering the overhead of each communication technology.

To support the new parameters and the package generation described in the trace files, multiple source code files were modified. Following we point each source code file and a brief description of the modifications.

ConfigurationManager.cpp: the *loadConfiguration()* method was altered to load the new configuration parameters of the traffic trace from YAML file, and these new parameters are validated in the *checkConfiguration()* method.

GlobalTrafficTrace.cpp: new code file added to the simulation. It is responsible to receive the input traces and carry them into the simulator.

Hub.cpp: altered to fix memory leakages (present on the current version of Noxim). The *antennaToTileProcess()* and the *tileToAntennaProcess()* methods contained memory allocation flaws which lead to system crash with huge trace files.

Noc.cpp: a controller of input trace was added in the *buildMesh()* method, in such way the simulator recognizes when all the input traces are complete so it can stop the simulation.

ProcessingElement.cpp: this file received most of our modifications. In the *rxProcess()* method the traffic trace controller performs two operations: i) it implements the send-wait-send control flow to allow or not the transmission of a flit; and ii) it implements the stop trace criteria, which permits to stop the simulation after all the traces from all PEs are sent. The traffic controller uses a queue to control the buffered and transmitted flits, and it is used by *rxProcess()* and *txProcess()* methods. In *canShot()* method, the loaded traces from each PE are executed, one by one, in the following way: a) verify the control flow if the flit can be shot; b) if the control flow allows the shot, a new packet will be created, formed by source and destination addresses, timestamp, payload (message size plus head and tail size) and an eventual padding if the packet size is smaller than the minimum packet size. Considering that the trace file can contain millions of lines, we only generate the packages whenever the simulator can send it, reducing the memory overhead of our proposal.

A new flow controller called *Send-Wait-Send* was implemented in order to control the send and receive communications. It guarantees that packets will be completely transmitted before a new one can be sent, i.e. a new packet is only transmitted after the tail flit of the previous packet transmitted by the same PE reaches the destination. Notice that different PEs can send in parallel, however only a single message from each PE will be inside the network at the same time.

B. The Input Trace File Format

In order to be as generic as possible, we choose to use a trace based on pure text format. The new input trace mode and the trace file name must be informed in the YAML configuration file. This file format is quite simple, each line should contain the identification of destination and size of a single message. The trace may have multiple lines to inform the full workload (multiple messages to be sent). Figure 1 illustrates this format.

```
<dst_id> <message_size>
```

Fig. 1: Input trace format for a specific PE.

C. Usage Example

In order to fully understand the extension in the Noxim, we present a usage example to follow step-by-step a synthetic trace simulation. In this example we model a NoC composed by a 2×2 mesh two-dimensional topology (i.e. four processing elements). Each PE is connected to one router, and the routers are interconnected to each other by a 32-bit wide wired link. Figure 2 illustrates this NoC topology. This topology is built declared inside the YAML configuration file, creating the PEs, routers, channels and hubs. The filename containing the routing table is also defined and the table is loaded, here we consider that XY routing algorithm is defined.

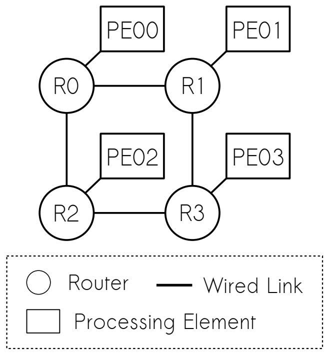
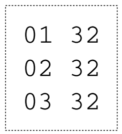


Fig. 2: Example of the NoC Mesh.

An example of input trace for the PE is presented on Figure 3. This trace file is formed by 3 messages, one for each other PE present on the system (similar to a broadcast behavior). Inside the configuration file, the base filename *trace.txt* for the trace was also set for this example, enabling the PE-00 to load the trace file *00_trace.txt*. Each line of this trace will be interpreted as a communication packet, which will be formed by head, payload and tail.

Whenever the simulation starts, the simulator will behave the following way: PE-00 will load its trace file and retrieve the first packet information. The packet is prepared by adding the head, payload and tail, it may also have padding to attend the minimal packet size (for tiny payloads). Once the packet



```
01 32
02 32
03 32
```

00_trace.txt

Fig. 3: Example of the trace file.

is ready to be sent, it is inserted inside the packet queue split in flits with 32 bits. Once the router becomes available, it will transmit the packet over the NoC.

After sending the packet, the PE-00 will not send a new packet until the previous one was received. This condition is part of what we call control flow in our proposal and depends on acknowledge signal from the receiver so the sender can follow the trace. We believe that this control creates a more realistic scenario, avoiding a PE to flood the network.

During the packet transmission, the flits follow the routing present in the routing table or the routing algorithm. Considering that we used an XY routing algorithm, it would mean that first the flits would flow in the horizontal direction and then on vertical direction until achieve the final destination. When modeling a wireless NoC where each router can be connected or not to a wireless hub, this routing algorithm would be valid between the source and the wireless hub, and from the wireless hub (from a different NoC) until the destination. Moreover, depending on the configuration parameters, the routers may have input and output buffers to store the incoming and outgoing flits.

Following the example, when the destination (PE-01) receives the packet coming from PE-00 internally the simulator unlock the PE-00 to perform the next packet transmission (if any left). Notice that any other PE (01, 02 and 03) can perform communications in parallel to the PE-00. Whenever all the PEs fully performs the transmission of their trace files, the simulator recognizes it and finishes the simulation.

IV. Evaluation

This section brings a validation of the simulator functionalities that were implemented in our extension. In order to show the potential of our proposal, we created a MPI wrapper inside the MPE2-2.4.7 from MPICH implementation version 3.2[1]. During the application execution, this wrapper can trace information regarding message size, origin and destination split per thread basis.

This wrapper writes in different trace files all the messages sent by the different MPI processes which used the MPI primitives (i.e. send, isend, bsend, bcast, gather, reduce, scatter, etc). Thus we are able to simulation this MPI application as it was executing in a system interconnected by different interconnections, such as Ethernet (Eth) 10 Gbps [28], Wireless Gigabit (WiGig) [29], InfiniBand (IB) [30], Wireless Interconnection with Code Division Multiple Access (WI-CDMA) [31] and Wireless Interconnection Token-based (WI-Token) [32]. We modeled each of these interconnection standards as if they were the external link, which connects four NoC systems together. A very brief explanation for each interconnection technology used in our evaluations is present bellow:

a) Ethernet: The Eth is a known wired connection widely used in local area networks. The Ethernet 10 Gb standard differs from earlier Ethernet standards because it operates only over fiber and in full-duplex mode [28]. The Media Access Control (MAC) parameters were maintained unaltered, with the maximum and minimum frame size being 1518 B and 64 B, respectively. Considering the head and tail, the minimum and maximum packet size in Eth is 72 B and 1526 B, respectively.

b) Wireless Gigabit: The new WiGig standard [33] is a wireless connection that works at 60 GHz with four channels and each channel transmission rate varies between 7 Gb/s and 10 Gb/s. Although the specification of 60 GHz [33] reports that the information may have 8 B at minimum size, considering the Protocol Adaptation Layer (PAL) for wireless bus extension [34], the packet payload field has variable size. The maximum packet contains the maximum payload size (128 B), plus overhead encapsulation (up to 16 B) and the PAL header is 4 B. Thus, the maximal packet size of the WiGig is 148 B.

c) Infiniband: The IB is an architecture which supports a range of applications on the backplane wired interconnection, such as the clustered hosts and I/O components. It works at full duplex transmission between any two fabrics elements. For the data rate, this work considers 50 Gb/s as predicted by authors for the year of 2017 [30]. The minimum payload in IB is 256 B and the maximum is 4096 B, and their head and tail packet has 126 B. In this way, the maximum packet size of infiniband is 382 B and the minimum is 4222 B.

d) Wireless NoC: WI-CDMA and WI-Token are designed for a seamless hybrid wired and wireless interconnection network for multi-chip systems. Nevertheless, [31] proposes the use of Code Divison Multiple Access (CDMA) achieving only 6 Gb/s in the wireless link, [32] uses a token based collision avoidance method reaching 16 Gb/s. In both, the packet size is fixed in 256 B and the payload, head and tail size was not informed.

A. NAS-NPB Workload

After creating the MPI wrapper proper to trace all the communication from MPI workloads. We used the NPB 3.3-MPI suite as workload to generate our traces. This benchmark suite contains MPI parallel applications that solves numerical methods for aerodynamic simulation problems. The following NPB applications run double-precision floating-point and were written in FORTRAN: Block Tri-diagonal (BT), Conjugate Gradient (CG), Fast Fourier Transform (FT), Lower and Upper triangular system (LU), Multigrid (MG) and Scalar Pentadiagonal (SP); while application Data Traffic (DT) and Integer Sort (IS) were written in C and uses mainly integer and logical

[1]The file /mpe2-2.4.7/src/wrappers/src/trace_mpi_core.c was instrumented to trace all the communication messages sent by the MPI.

TABLE II: Network parameters.

Parameter	Ethernet	Wireless Gigabit	InfiniBand	WiNoc CDMA-based	WiNoC Token-based
Head and Tail (Bytes)	26	4	126	-	-
Minimum Payload (Bytes)	46	4	256	-	-
Maximum Payload (Bytes)	1500	144	4096	-	-
Minimum Packet Size (Bytes)	72	8	382	256	256
Maximum Packet Size (Bytes)	1526	148	4222	256	256
Data Rate (Gb/s)	10	8	50	6	16

operations. The more details for each application can be found in [35], the impact of NPB workload in NoC architectures are presented in [36] and their pattern communication in [37].

B. Metrics and Parameters

The applications from NPB suite have different problem sizes (benchmark classes) named in ascending order as S, W, A, B, C and D [35]. Our evaluations used the problem size A, which is the most used in real machine tests, due to its medium size and reasonable execution time to be evaluated in a simulated environment.

The number of threads was set to sixteen in order to simulate four systems with 2×2 mesh size each, with a total of 16 PEs. Each NPB application have a different communication pattern, already published in different papers [38], [25]. The table II shows the parameters for each network interconnection evaluated in this paper. These parameters are the input values to setup the YAML configuration file for Noxim simulator (extended version). The simulation of the same system was performed with these different parameters to realize a network technology comparison on NoC architecture.

C. Results and Discussion

Table III shows the communication overhead for each interconnection network in order to process the messages from the NPB applications. The estimate in this table were obtained using the analytical model. In this table, the columns WI-CDMA and WI-Token are exactly the same as these wireless interconnections have the same minimum and maximum package size. It is possible to observe that WiGig has a tiny overhead compared with others communication networks for all NPB applications while the IB has the highest one on average due to the minimum packet size restriction. It is possible to correlate the overhead with the performance results (Figure 4). For instance, when comparing IB for the applications with less than 15% overhead (BT, DT, FT, IS, LU, MG, SP), it can be seen that Eth performed on average roughly 69% worse. However, when comparing the CG application which achieved 49% overhead on IB the performance is 61% worst. It is possible to see that the performance difference is reduced to only 8%.

Figure 4 depicts the execution time necessary for each interconnection network to transmit all messages of the workload applications. The results for all the evaluated interconnections are normalized to the Ethernet execution time. The comparison shows that IB has the better performance in relation to Eth, WiGig, WI-CDMA and WI-Token even with larger overhead. This result is because its data rate overcome the overhead issue. This simulation and analysis evidences that our extension on the Noxim simulator generates fair results while accepting external input traces extracted from real applications.

V. Conclusions and Future Work

Simulation is one of the main tool used to analyze new proposals in the NoC field. Among all proposed simulators, the Noxim simulator stands out due to its accuracy near to a real system. However, traces of real applications were not easily supported by the trace table method (present on Noxim), which requires the researcher to inform the general behavior of the system (traffic injection rate) not even considering the packet size parameter. However, such method present clear limitations to model more close to application behavior.

In this work, we improve the Noxim simulator in order to it accept external input traces formed by source, destination and size for each specific packet to be sent over the interconnection system. We performed this extension by modifying the Noxim source code files. A list of the files and required modification was presented together with a validation of these changes.

Our validation was based on the simulation of messages from 8 real application modeling 5 different interconnection systems. The performance results fits to the overhead estimates using analytical formulas that consider the minimum packet size, head and tail overheads. In this way, the presented Noxim extension will help researchers during performance tests of new proposals through real application traces. Future work includes the support of wireless broadcast, virtual channels for communication and improvement in the deadlock avoidance.

Acknowledgment

This work was partially supported by CNPq and CAPES.

References

[1] L. Benini and G. D. Micheli, “Networks on Chips: A New SoC Paradigm,” *Computer*, vol. 35, no. 1, 2002.

[2] G. De Micheli and L. Benini, “Networks on Chips: 15 Years Later,” *Computer*, vol. 50, no. 5, 2017.

[3] S. Deb and A. Ganguly and P. P. Pande and B. Belzer and D. Heo, “Wireless NoC as Interconnection Backbone for Multicore Chips: Promises and Challenges,” *Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 2, no. 2, 2012.

TABLE III: Overhead for NPB applications.

Application	Ethernet	Wireless Gigabit	InfiniBand	WiNoc CDMA-based	WiNoC Token-based
BT	0,13%	0,02%	0,60%	0,04%	0,04%
CG	9,22%	0,71%	49,54%	27,09%	27,09%
DT	0,93%	0,07%	4,97%	2,72%	2,72%
FT	0,06%	0,01%	0,29%	0,13%	0,13%
IS	0,10%	0,01%	0,50%	0,21%	0,21%
LU	2,96%	0,40%	13,82%	1,06%	1,06%
MG	1,72%	0,13%	10,40%	6,24%	6,24%
SP	0,15%	0,02%	0,68%	0,03%	0,03%

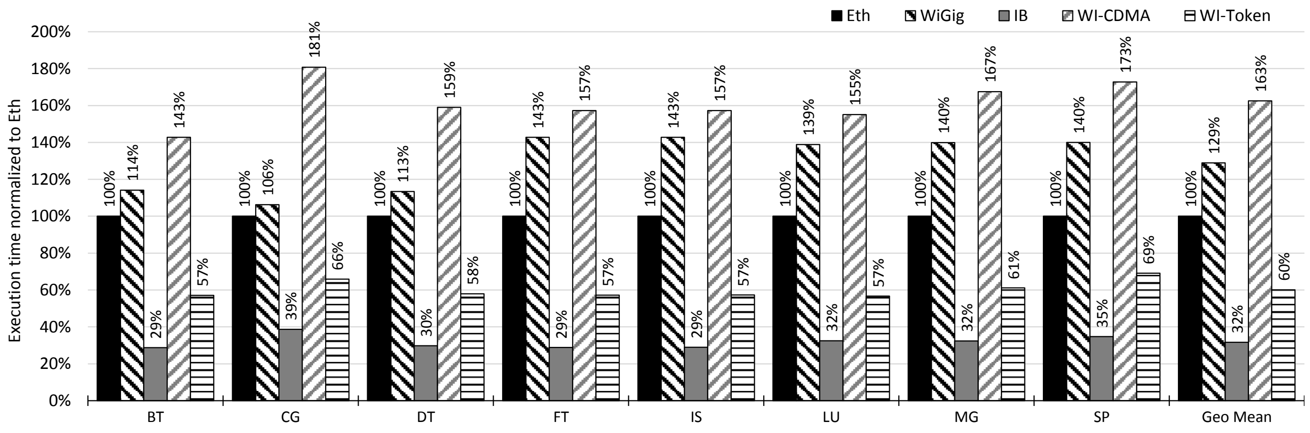


Fig. 4: NPB applications execution time normalized to Ethernet results.

[4] D. Mangano and I.A. Urzi, "System for Designing Network-on-Chip Interconnect Arrangements," 2015, US Patent 9,202,002. [Online]. Available: {www.google.ch/patents/US9202002}

[5] B. A. Floyd and C. Hung and K. K. O, "Intra-Chip Wireless Interconnect for Clock Distribution Implemented with Integrated Antennas, Receivers, and Transmitters," *Journal of Solid-State Circuits*, vol. 37, no. 5, 2002.

[6] M. F. Chang and J. Cong and A. Kaplan and M. Naik and G. Reinman and E. Socher and S. W. Tam, "CMP Network-on-Chip Overlaid with Multi-Band RF-Interconnect," 2008.

[7] Jain, Raj, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley & Sons, 1990.

[8] V. Catania and A. Mineo and S. Monteleone and M. Palesi and D. Patti, "Noxim: An open, extensible and cycle-accurate network on chip simulator," in *IEEE 26th Int. Conf. on Application-specific Systems, Architectures and Processors*, 2015, pp. 162–163.

[9] N. Jiang and J. Balfour and D. U. Becker and B. Towles and W. J. Dally and G. Michelogiannakis and J. Kim, "A Detailed and Flexible Cycle-Accurate Network-on-Chip Simulator," in *Int. Symp. on Performance Analysis of Systems and Software*, 2013.

[10] Nakajima, Keita and Kurebayashi, Shuto and Fukutsuka, Yusuke and Hieda, Takuji and Taniguchi, Ittetsu and Tomiyama, Hiroyuki and Takada, Hiroaki, "Naxim: A fast and retargetable network-on-chip simulator with qemu and systemc," *Int. Journal of Networking and Computing*, vol. 3, no. 2, 2013.

[11] P. Ren and M. Lis and M. H. Cho and K. S. Shim and C. W. Fletcher and O. Khan and N. Zheng and S. Devadas, "HORNET: A Cycle-Level Multicore Simulator," *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 6, 2012.

[12] P. Abad and P. Prieto and L. G. Menezo and A. Colaso and V. Puente and J. Á. Gregorio, "TOPAZ: An Open-Source Interconnection Network Simulator for Chip Multiprocessors and Supercomputers," in *Int. Symp. on Networks-on-Chip*, 2012.

[13] Y. Ben-Itzhak and E. Zahavi and I. Cidon and A. Kolodny, "HNOCS: Modular open-source simulator for Heterogeneous NoCs," in *Int. Conf. on Embedded Computer Systems*, 2012.

[14] Ben-Itzhak, Yaniv and Zahavi, Eitan and Cidon, Israel and Kolodny, Avinoam, "NoCs Simulation Framework for OMNeT++," in *Int. Symp. on Networks-on-Chip*, 2011.

[15] C. Wang and W. H. Hu and N. Bagherzadeh, "A Wireless Network-on-Chip Design for Multicore Platforms," in *Int. Euromicro Conf. on Parallel, Distributed and Network-Based Processing*, 2011.

[16] Lis, Mieszko and Shim, Keun Sup and Cho, Myong Hyon and Ren, Pengju and Khan, Omer and Devadas, Srinivas, "DARSIM: A Parallel Cycle-Level NoC Simulator," in *Workshop on Modeling, Benchmarking and Simulation*, 2010.

[17] Hestness, Joel and Grot, Boris and Keckler, Stephen W., "Netrace: Dependency-driven Trace-based Network-on-chip Simulation," in *Int. Workshop on Network on Chip Architectures*, 2010.

[18] N. Agarwal and T. Krishna and L. S. Peh and N. K. Jha, "GARNET: A detailed on-chip network model inside a full-system simulator," in *Int. Symp. on Performance Analysis of Systems and Software*, 2009.

[19] H. Hossain and M. Ahmed and A. Al-Nayeem and T. Z. Islam and M. M. Akbar, "Gnocsim - A General Purpose Simulator for Network-On-Chip," in *Int. Conf. on Information and Communication Technology*, 2007.

[20] Jain, Lavina and Al-Hashimi, BM and Gaur, MS and Laxmi, V and Narayanan, A, "NIRGAM: a simulator for NoC interconnect routing and application modeling," in *Design, Automation and Test in Europe Conf.*, 2007.

[21] Zhonghai Lu and Rikard Thid and Mikael Millberg and Axel Jantsch, "NNSE: Nostrum Network-on-Chip Simulation Environment," in *Swedish System-on-Chip Conf.*, 2005.

[22] Binkert, Nathan and Beckmann, Bradford and Black, Gabriel and Reinhardt, Steven K. and Saidi, Ali and Basu, Arkaprava and Hestness, Joel and Hower, Derek R. and Krishna, Tushar and Sardashti, Somayeh and Sen, Rathijit and Sewell, Korey and Shoaib, Muhammad and Vaish, Nilay and Hill, Mark D. and Wood, David A., "The Gem5 Simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, 2011.

[23] S. J. E. Wilton and N. P. Jouppi, "CACTI: an enhanced cache access and cycle time model," *Journal of Solid-State Circuits*, vol. 31, no. 5, 1996.

[24] Riley, George F. and Henderson, Thomas R., *The ns-3 Network Simulator*, 2010.

[25] D. Bailey and T. Harris and W. Saphir and R. van der Wijngaart and A. Woo and and M. Yarrow, "The NAS parallel benchmarks 2.0," NAS Technical Report, NAS-95-020, Tech. Rep., 1995.

[26] K. Swaminathan and D. Thakyal and S. G. Nambiar and G. Lakshminarayanan and Seok-Bum Ko, "Enhanced Noxim simulator for performance evaluation of network on chip topologies," in *Recent Advances in Engineering and Computational Sciences*, 2014, pp. 1–5.

[27] Y. Kurimoto and Y. Fukutsuka and I. Taniguchi and H. Tomiyama, "A hardware/software cosimulator for Network-on-Chip," in *Int. SoC Design Conf.*, 2013, pp. 172–175.

[28] LAN/MAN Standards Committee, "IEEE Standard for Ethernet," *IEEE Std 802.3-2015*, 2016.

[29] C. J. Hansen, "WiGiG: Multi-gigabit wireless communications in the 60 GHz band," *IEEE Wireless Communications*, vol. 18, no. 6, 2011.

[30] InfiniBand Trade Association and others, "InfiniBand Architecture Specification, release 1.0," 2000. [Online]. Available: {www.infinibandta.org}

[31] M. S. Shamim and J. Muralidharan and A. Ganguly, "An Interconnection Architecture for Seamless Inter and Intra-Chip Communication Using Wireless Links," in *Int. Symp. on Networks-on-Chip*, 2015.

[32] M. S. Shamim and N. Mansoor and R. S. Narde and V. Kothandapani and A. Ganguly and J. Venkataraman, "A Wireless Interconnection Framework for Seamless Inter and Intra-Chip Communication in Multichip Systems," *Transactions on Computers*, vol. 66, no. 3, 2017.

[33] WiFi Alliance. (2017) 60 GHz Technical Specification v1.0. [Online]. Available: {www.wi-fi.org}

[34] W. Alliance. (2014) Wi-fi alliance wigig wireless bus extension technical specification. [Online]. Available: www.wi-fi.org

[35] R. F. VanderWijngaart and J. Haopiang, "NAS Parallel Benchmarks, Multi-Zone Versions," *NAS Technical Report*, 2003.

[36] H. C. de Freitas and L. M. Schnorr and M. A. Z. Alves and P. O. A. Navaux, "Impact of Parallel Workloads on NoC Architecture Design," in *Euromicro Conf. on Parallel, Distributed and Network-based Processing*, 2010.

[37] E. H. M. da Cruz and M. A. Z. Alves and A. Carissimi and P. O. A. Navaux and C. P. Ribeiro and J. F. Mehaut, "Using Memory Access Traces to Map Threads and Data on Hierarchical Multi-core Platforms," in *Int. Symp. on Parallel and Distributed Processing Workshops and Phd Forum*, 2011.

[38] M. Diener, E. H. M. Cruz, L. L. Pilla, F. Dupros, and P. O. A. Navaux, "Characterizing Communication and Page Usage of Parallel Applications for Thread and Data Mapping," *Performance Evaluation*, vol. 88-89, 2015.