2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum

Self-Correction Trace Model: A Full-System Simulator for Optical Network-on-Chip

Mingzhe Zhang*, Liqiang He†
College of Computer Science
Inner Mongolia University
Hohhot, Inner Mongolia, P.R. CHINA
Email: *zhangmingzhe@ict.ac.cn
†liqiang.he@gmail.com

Dongrui Fan
State Key Laboratory of Computer Architecture
Institute of Computing Technology，Chinese Academy of Sciences
Beijing, China
Email: fandr@ict.ac.cn

***Abstract*—The improvement of the emerging technology involves the nanophotonic into the on-chip interconnection, which provides a large communication capability for the future large-scale CMP processor. As an important way to the architecture research, full–system simulation has been adopted by many researchers. Since the optical devices are fundamentally different from the conventional electronic elements, new methodology and tools are needed to simulate an Optical Network-on-Chip (ONOC) with real workload. In this paper, we introduce a high precise full-system ONOC simulation system. To build this system, we propose a self-correction trace model for accurate simulation in a reasonable period of time. Finally, to test our simulation system, we present a simple case-study to compare our system running real application with a baseline NOC simulator. The result shows that our simulation system achieves a high precision, while not substantially extend the total simulation time.**

Keywords-nanophotonic; Network-on-Chip; full-system simulation; trace; self-correction

I. INTRODUCTION

Simulator is one of the most important tools for Network-on-Chip (NOC) research. In recent years, researchers have proposed varies NOC architectures base on different simulators [1] [2] [3].

While the CMP chip scales from tens of cores to hundreds of core, the electronic based NOCs face more and more severe power problems. In some high performance processors, the power consumed by NOC accounts for over 50% of the overall dynamic power overhead [4]. In addition, electronic NOC also faces the area budget overhead, routing complexity, long transmission latency, crosstalk and other issues [5].

In recent years, the development of the optical devices integration scheme turns the silicon photonics into a promising technology for the chip-scale interconnection networks [6]. Compared with the electronic NOC, Optical NOC (ONOC) has obvious advantages in power consumption, and provides higher bandwidth through wavelength-division-multiplexed (WDM) transmission. Meanwhile, the crosstalk in optical transmission is well comparing with that in circuit. These features make ONOC the ideal chip-scale interconnection technology for next generation CMP processor.

Due to the following reasons, simulator needs to make changes for ONOC research: 1) optical devices have fundamentally different model from the one used for electronic elements which the traditional simulator cannot simulate accurately; 2) ONOC provides high bandwidth by using WDM, which the existing simulator is hard to model the behavior and calculate the dynamic power consumption; 3) the crosstalk and power consumption caused by optical signal transmitting through the waveguide also require new model to calculate.

We propose a full-system simulator for modeling and analyzing the performance of CMP systems which may use electronic NOC, optical NOC, or a hybrid network that combining both technologies together. The implemented simulator includes the front-end execution engine and the optical NOC simulating part. When a real application runs on the system, the impacts of all components in the system can be simulated. We use Simics and GEMS [7] as the front-end engine, and then use the traces from the front-end to drive the ONOC simulator. The existing ONOC simulators focus on system-level behaviors [8] and the physical-layer simulation tools. In our simulator, we choose PhoenixSim [9], which can capture the physical characteristics of the optical components. In order to make the trace more accurate in recording the request of the interconnection elements, we present a self-correction trace model which can obtain good portability. This trace model divides the trace into dependent and independent parts, and injects them separately to the ONOC simulator in different ways. With this model, we can avoid accuracy loss caused by using synthetic traffic pattern, and get the simulation result in a reasonable time.

The rest of this paper is structured as follows. Section II presents the related work. Section III introduces the structure of our simulator. Section IV describes the self-correction trace model. Section V presents a simple case-study for the simulator. Finally, Section VI presents the conclusion.

978-0-7695-4676-6/12 $26.00 © 2012 IEEE
DOI 10.1109/IPDPSW.2012.26

II. RELATED WORK

Many researchers use simulator to build and test varies of chip-scale interconnection architectures. Obviously, simulated features and performance affect the results of the study. To meet different requirements, researchers have developed a number of simulators focusing on different parts of the CMP system.

Noxim [10] and Sicosys [11] are both widely used general purpose NOC simulator for CMP systems, which can provide precious results about a variety of on-chip message routers. Using such tools, researchers can obtain NOC transmission latency, bandwidth and dynamic power consumption in short time. However, these simulators are synthetic traffic-pattern driven, which researchers cannot obtain accurate NOC performance with real application. In other words, this type of system-level simulators ignores the impact of other components to the NOC in CMP architecture such as the processing cores, memory hierarchy, and other possible on-chip elements.

Garnet [12] is a NOC simulator as a part of Simics-GEMS infrastructure. It provides a detailed simulation of the interaction between interconnection elements and other on-chip components. Since the Simics allows operating system and application running on it, Garnet is an excellent tool which presents the full-system simulation for NOC. Unfortunately, Garnet only supports the 2-D mesh topology, and no interface between the core and NOC is simulated.

The development of ONOC technology calls for new simulator to exploit and evaluate varies designs, since existing tools cannot model optical devices. Previous work presents an ONOC model focusing on the system-level behavior, which is developed using SystemC [8]. In contrast to the system-level simulation, PhoenixSim provides a physical-layer model to capture the detail characteristics and metrics of the optical interconnection elements which have no electronic equivalent. Unfortunately, both two simulators above are synthetic traffic pattern driven.

In order to access the ONOC performance when running real application, a full-system simulator is required. There are two ways to achieve the target: one is to integrate the ONOC simulator into an execution simulate infrastructure such as Simics-GEMS and GEM5 [13], the other one is to drive the ONOC simulation with the trace captured from the execution simulator. The first one can simulate all components involved in a CMP system, but also leads to long simulation time. The trace-driven one is a compromising choice to achieve full-system simulation which takes care of both accuracy and simulation time. Netrace [14] provides varies portable trace files for propelling NOC simulators. Netrace assumes a fixed latency for message communication which is the average number of hops calculated from an analytical model. Since the fixed latency is similar to the variant latency in the electronic NOC, Netrace works well with electronic NOC simulator in contrast to ONOC simulation which is serious inference base on the fact that optical transmission latency is much shorter than it is in electronic. Self-related trace presents a general purples trace model by adding dependency information for the trace record to improve simulation accuracy [15]. However, this model cannot record the occurrence of independent traces, which affects the accuracy of the result especially for ONOC simulation. In order to obtain more precious evaluation study in a reasonable time, we design a self-correction trace model. Using this model, we integrate the PhoenixSim with Simics-GEMS, and set up a highly configurable full-system ONOC simulator.

III. FULL-SYSTEM OPTICAL NOC SIMULATOR

In this section we will introduce our ONOC full-system simulator and each tool involved in the system. Using the simulator, we can simulate optical/electronic NOC in different topologies with various parts of the system configurations, including processors, memory hierarchy storage, and on-chip interconnection.

In our system, we use Simics to run the operating-systems and the applications. Simics is a high performance platform for deterministic execution of the applications which can simulate single-/multi-core systems, peripherals and other components. It attempts to achieve the balance between the performance and accuracy in simulation. The General Execution-driven Multiprocessor Simulator (GEMS), as the supplement of Simics, presents the detailed memory hierarchy and interconnect system simulation. GEMS consists two main modules: Opal and Ruby, which support the out-order execution and cache hierarchy simulation separately. Ruby contains an electronic NOC model called Garnet, which presents detailed simulation for all parts in an on-chip interconnect system in mesh topology.

Since we need to simulate ONOC, new module should be involved to replace Garnet. We choose PhoenixSim to model computer systems incorporated silicon nanophotonic devices, which is an OMNET++ [16] based high-precision ONOC simulator developed in C++. PhoenixSim provides a new library for modeling both electronic and optical devices at physical-layer level and at architectural and system levels, such as the electronic/optical routers, waveguide, and modulator. It also integrates ORION [17] to calculate the dynamic power and area overhead for the electronic circuit, while using a new built-in model to evaluate the optical devices since the existing models cannot describe some peculiar phenomena preciously such as insertion loss, crosstalk in waveguide, and energy dissipation in resonator rings. PhoenixSim is a discrete-event driven simulator, and provides several typical traffic patterns for analog. We achieve full-system ONOC simulator by connecting the PhoenixSim and Simics-GEMS.

There are mainly two ways for connecting simulators. The first choice is to directly integrate PhoenixSim with Simics-GEMS. In this way, the execution part calls the ONOC module through the interface to get the most precious interaction of application and architecture. Unfortunately, this choice requires much more computation time and resources because of the high number of detail elements which are involved in the simulation. As an alternative choice, trace-driven simulation significantly saves time and resources due to the reuse of the trace-file and omission of simulation for the less important components. In order to

reduce research cost, we choose the second way. We firstly capture the access requests to the Garnet and the message back from the NOC module, and then inject the trace into the PhoenixSim. In order to record the accesses to the NOC accurately, while avoiding the interference to the simulation result due to the latency information from Garnet, we design a self-correction related trace model which we will introduce in section IV.

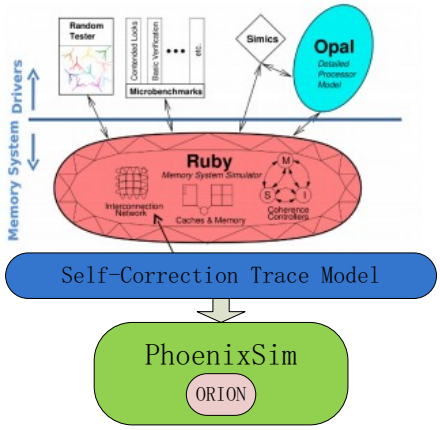

Figure 1. Full-System ONOC Simulation System Composed by Simics-GEMS, PhoenixSim and the Self-Correction Trace Model

Fig. 1 shows a global view of our simulation system. The top of the figure presents the Simics simulator, on which we run the operating system and benchmarks. GEMS extends the Simics to simulate multiprocessor architectures in detail. In GEMS, Ruby module describes the memory hierarchy and interconnection system. We modify the code and insert a module in Ruby to capture the trace. Then we build a module to read the trace file and drive the PhoenixSim simulator according to the trace.

IV. SELF-CORRECTION TRACE MODEL

Compared to the full-system simulation with an entire tool, trace-driven simulation significantly reduce the computation requirements and time, making the simulation which originally could take days or even weeks in a reasonable period of time. In this section, we introduce a self-correction trace model. Using this model, we connect PhoenixSim and Simics-GEMS, while avoiding the latency information generated in Garnet disturbing the simulation result PhoenixSim simulator.

To facilitate the discussion, we assume that all the cores in the CMP chip are in-order ones. When one core sends a request to the NOC, it will suspend until the feedback message arrives. The entire process is shown in Fig 2.

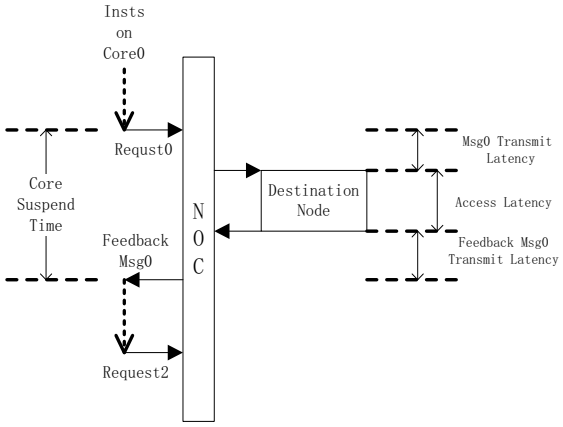

Figure 2. Communication Model for NOC Request

Since the trace file is generated while applications are running, directly using the trace to drive a NOC simulator will bring the original latency into the new simulation. Figure 3(a) presents an example. The trace file records the injection time of the NOC messages. We assume that core0 sends a message (msg0) to the NOC at time TC0, and then msg0 arrives the destination node at time TC0+LG0. LG0 indicates the msg0's transmit latency through the electronic network-on-chip. In our system, transmit latency through the ONOC usually is much shorter than it is in electronic link. The arrival time of msg0 through ONOC will be TC0+LT0. If we inject the trace file into the ONOC simulator directly, after msg0 reach the destination node, the following messages will keep the same injection times as that in the original electronic NOC. The entire ONOC simulation will get the same latency performance as the electronic NOC (as shown in Figure 5). To solve the problem, we improve the traditional trace model and design a self-correction trace model.

In contrast of traditional traces, self-correction trace model divides the trace message into independent type and dependent type. The independent messages record the initial request to the network, whose occurrence only depends on the instruction being executed. In contrast, dependent messages record the occurrence of the response packets on the network. Since each system component processing a network packet needs some time to finish some certain operations, the dependent messages will occur at a certain cycles after they received the previous message. Therefore, we only need to modify the independent messages' occurrence time to remove the affection from the Garnet latency.

As shown in Figure 3(b), msg0 enters the ONOC at time TC0 and reaches the destination node at time TC0+TL0. Accordingly, all the following injections from the same core should be amended. For example, TC1 is turned to be TC1'= TC1+TL0-LG0. This method is also proper to the situation that TL0>TG0 in short distance transmission through the optical links.

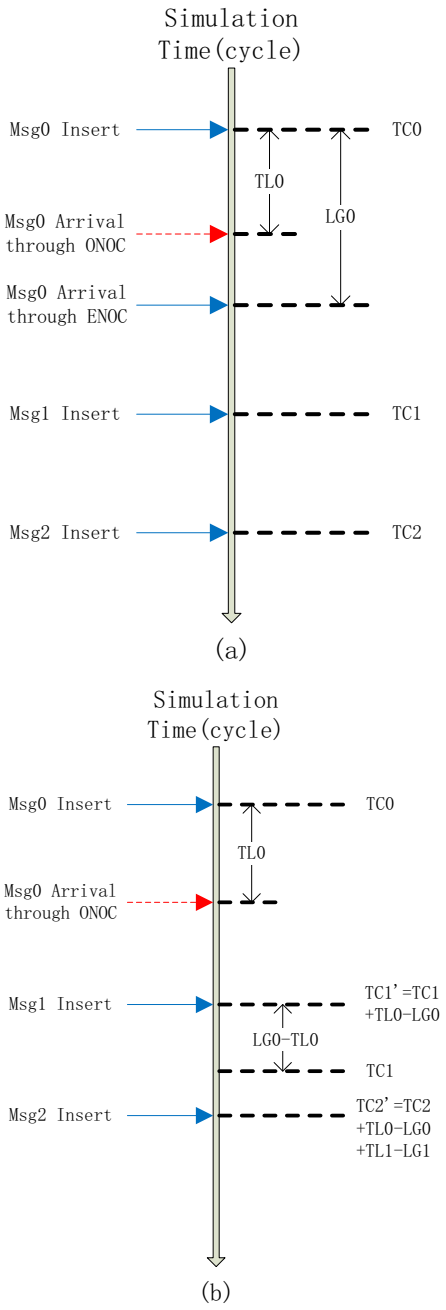

Figure 3. Self-Correction Model

Essentially, each message in self-correction trace file contains following fields: the message ID as unique identifier, the message type field (MT) which indicts whether the message is independent type or not, the source node component (SC), the destination node component (DC), the type of the coherence protocol (PT), the size of the flit (S), the operating latency of the component (LC), the id of the dependent message which takes the value of -1 in independent message (DM), the occurrence time of the message in original simulation (TC), and the transmission latency of the message in Garnet module (LG). The last two fields are the key of self-correction trace model, by which we can remove the original latency information.

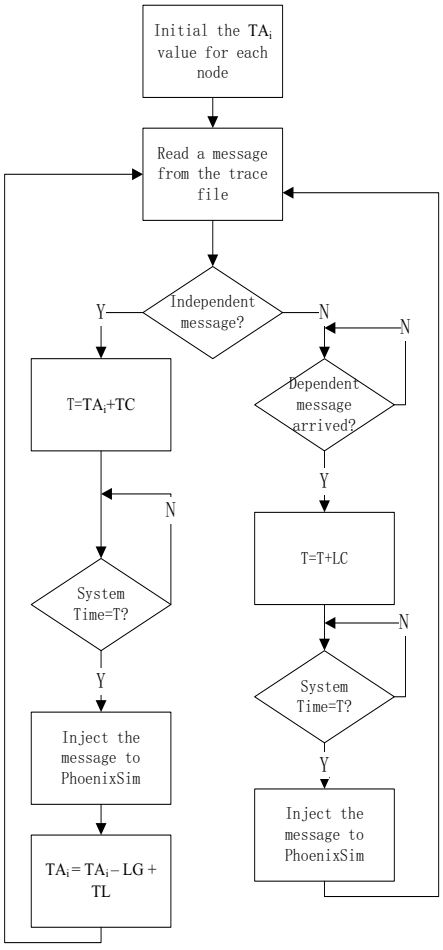

Figure 4. Routine for drving PhoenixSim with Self-Correction Model

To drive the PhoenixSim with the trace file, we develop a driver module. It will prepare an adjustment value TA_i for each node component in the network and set their value to be 0 in initialization phase. Then the module reads the trace messages in the order of ID fields. If the message is an independent one, the program will add its TC value to the TA_i that belongs to the corresponding SC node and take the result as the time to inject the message to PhoenixSim. When the message finishes the transmission in the new network, we can get the latency TL, and adjust the source node's TA_i by $TA_i = TA_i - LG + TL$. In contrast, a dependent message will be injected into PhoenixSim simulator LC cycles after the SC node received the message DM. Fig. 4 shows the whole routine.

Table I presents a part of self-correction trace file. In this example, the message with ID 536 has a dependency with message 533, and message 535 has a dependency with message 407.

TABLE I. EXAMPLE OF A SELF-CORRECTION TRACE

ID	MT	SC	DC	PT	
	S	LC	DM	TC	LG
533	I	0-L1Cache	0-L2Cache	Get	
	32	3	-1	25539	5
534	I	1-L1Cache	1-L2Cache	Get	
	64	3	-1	25539	3
535	D	4-Directory	4-L2Cache	Data	
	512	2	407	25621	15
536	D	0-L2Cache	0-L1Cache	Data	
	32	8	533	25670	10

We modify the source code of Ruby and insert a module to capture the trace. Through the previous method, we obtain a trace-driven ONOC simulation system which support real workload, while avoiding the impact to the simulation result from the original latency information in the trace file. This model offers a significantly reduction in the simulation requirements while maintaining the accuracy of the result.

V. CASE STUDY

In this section, we present a simple case study to show the efficiency and accuracy of self-correction trace model.

Our simulation system consists of Simics-GEMS, trace driven module and PhoenixSim simulator. Thus, we can evaluate the performance of ONOC architectures with real workload.

TABLE II. SIMULATION CONFIGURATON

	Simulation Parameter
Core	Ultra-Sparc III
L1 Cache	32Kbytes, Private
L2 Cache	4*1MByte, Shared
Topology	4*4 mesh
Routing Algorithm	Static X-Y
Others	Directory-Based MOESI protocol 4 Memory Controller
Workload	PARSEC V2.1

Table II shows the configuration of our simulation. Our target system is a 4*4 2D-mesh based CMP architecture with 16 Ultra-Sparc III in-order processing cores. Each core with a private 32Kbytes L1 cache is considered as a network node. We use static X-Y routing algorithm in order to prevent deadlock. There are 4 L2 cache with memory controller at the edge of the chip. The physically distributed L2 cache banks are logically shared by all cores. The coherence protocol between L1 and L2 cache is directory-based MOESI.

We take four applications from the PARSEC v2.1 suite [18] as the benchmark which consists of Blackscholes (BS), Bodytrack (BT), Streamcluster (SC), and Swaptions (SW). We run the benchmarks on Simics-GEMS platform, and then drive the PhoenixSim simulator with the self-correction trace file. To evaluate the performance of the simulator, we run the benchmarks on three NOCs: electronic NOC in Garnet, electronic NOC and ONOC in PhoenixSim. All the parameters for electronic NOC and ONOC used are the same as the default ones in PhoenixSim.

Fig. 5 shows the simulation result of network latency. Considering the result from Garnet as 1, latency from PhoenixSim-based electronic NOC arise by 5% maximum due to the physical-layer simulation. Since the ONOC has much shorter transmission latency, the latency is decrease by 64% maximum compared with PhoenixSim-based electronic NOC. Especially, when we use the original trace file to drive the ONOC simulation, we get the same latency performance as the ENOC.

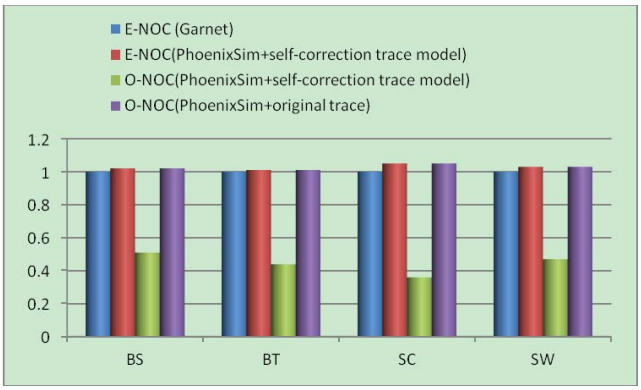

Figure 5. Normalized Network Latency.

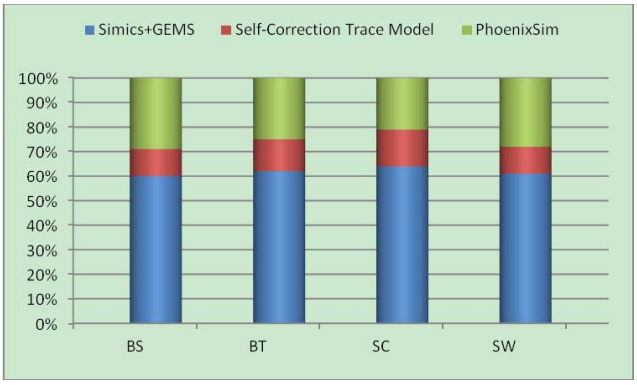

Figure 6. Simulation Time of each Full-System Component.

Fig. 6 shows the simulation time on different simulators. About 68% of the time was taken by the Simics-GEMS platform. So we can get about 68% of simulation time reduction through the reusing of the trace file.

VI. CONCLUSION

The development of nanophotonic technology calls for new tools for simulation and evaluation. Consequently, a series of work have been presented on this area in recent years. However, most of the new tools work with synthetic workloads. Since not including real benchmark can alter the result, we need to build a highly configurable simulator for

evaluating the performance of the ONOC architecture with real applications.

In this paper, we present an ONOC simulation system by connecting PhoenixSim simulator, a physical-layer ONOC simulator, with Simics-GEMS platform. In order to balance the simulation requirements and the accuracy, we use the trace from execution part to drive the ONOC simulator. To avoid the interference by the latency from original simulator, we design a self-correction trace model. This combination of models and tools provides a platform for the design exploration of the new systems with optical interconnection in our future work.

Finally, we test our simulation system with four different applications from PARSEC v2.1 benchmark suite. We compare the simulation results from both our system and Simics-GEMS, finding no significant differences between the self-correction model and the integrated full-system simulator.

In the future, we plan to evaluate some new design about ONOC and Hybrid NOC architecture base on our full-system simulation system. We will also be considering performance improving of the self-correction trace model, possibly in higher execution speed.

ACKNOWLEDGMENT

This work is supported by Open Project of Key Laboratory of Computer Science under Grant No. SYSKF1104.

REFERENCES

[1] D. Bertozzi and L. Benini, "Xpipes: A network-on-chip architecture for gigascale systems-on-chip", IEEE Circuits and Systems Magazine, vol. 4, Issue 2, pp. 18-31, 2004.

[2] T. Bjerregaard and J. Sparso, "A router architecture for connection-oriented service guarantees in the MANGO clockless Network-on-Chip", Proceedings of IEEE on Design Automation and Test, vol. 2, pp. 1226-1231, 2005.

[3] A. Agarwal and R. Shankar, "A layered architecture for NOC design methodology," IASTED International Conference on Parallel and Distribution Computing and Systems, pp. 659-666, 2005.

[4] B. G. Lee, A. Biberman, K. Bergman, N. Sherwood-Droz, and M. Lipson, "Multi-wavelength message routing in a non-blocking four-port bidirectional switch fabric for silicon photonic networks-on-chip," Optical Fiber Communication Conference, OSA Technical Digest (CD) (Optical Society of America, 2009), paper OMJ4.

[5] J. Liu, J. Psota, N. Beckmann, J. Miller, J.Michel, J.Eastep, et.al. "ATAC: A Manycore Processor With On-Chip Optical Network," Series/Report no: MIT-CSAIL-TR-2009-018, 2009.

[6] M.R. Watts, D.C. Trotter, R.W. Young, A.L. Lentine, "Ultralow power silicon microdisk modulators and switches", IEEE International Conference on Group IV Photonics, Sep. 2008.

[7] M. M. K. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E.Moore, M. D. Hill, and D. A. Wood, "Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset," SIGARCH Comput. Archit. News, vol. 33, no. 4, pp. 92–99, 2005.

[8] M. Briere, E. Drouard, F. Mieyeville, D. Navarro, I. O'Connor, F. Gaffiot, "Heterogeneous modelling of an optical network-on-chip with SystemC," in IEEE International Workshop on Rapid System Prototyping, pp. 10–16, 2005.

[9] J.Chan, G.Hendry, A.Biberman, K.bergman, L.P.Carloni, "PhoenixSim: A Simulator for Physical-Layer Analysis of Chip-Scale Photonic Interconnection Networks," in Proceedings of the Conference on Design, Automation and Test in Europe, 2010.

[10] "Noxim Simulator", 2009. http://noxim.sourceforge.net

[11] V. Puente, J. A. Gregorio, and R. Beivide, "SICOSYS: An ntegrated framework for studying interconnection network n multiprocessor systems," in 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing, Jan.2002, pp. 15–22

[12] N. Agarwal, T. Krishna, L.-S. Peh, and N. K. Jha, "GARNET: A detailed on-chip network model inside a full-system simulator," in ISPASS, Apr. 2009, pp. 33–42.

[13] "gem5: A Multiple-ISA Full System Simulator with Detailed Memory Modeling," in International Symposium on Computer Architecture, June. 2011.

[14] J. Hestness, B. Grot, S.W. Keckler, "Netrace: Dependency-Driven Trace-Based Network-on-Chip Simulation," in NoCArc 2010, Dec. 2010.

[15] F. Trivino, F. J. Andujar, F.J. Alfaro, J. L. Sanchez, A. Ros, "Self-related traces: An alternative to full-system simulation for NoCs", in High Performance Computing and Simulation (HPCS), 2011

[16] A. Varga, "OMNeT++ discrete event simulation system," http://www.omnetpp.org.

[17] H. Wang, X. Zhu, L. Peh, S. Malik, "ORION: A power-performance simulator for interconnection networks," in Proceedings. 35th Annual IEEE/ACM International Symposium on Microarchitecture, 2002.

[18] C. Bienia and K. Li, "PARSEC 2.0: A New Benchmark Suite for Chip-Multiprocessors," in MoBS, 2009.