New Orleans JAZZED about Engineering Education

Paper ID #14561

ENoCS: An Interactive Educational Network-on-Chip Simulator

Paul William Viglucci, Binghamton University
Prof. Aaron P. Carpenter, Wentworth Institute of Technology

Professor Carpenter is an Assistant Professor at the Wentworth Institute of Technology. In 2012, he completed his PhD at the University of Rochester, focusing on the performance and energy of the on-chip interconnect.

ENoCS: An Interactive Educational Network-on-Chip Simulator

Paul Viglucci* and Aaron Carpenter†
*Dept. of Electrical & Computer Engineering
Binghamton University, pvigluc1@binghamton.edu
†Dept. of Electrical Engineering & Tech.
Wentworth Institute of Technology, carpentera1@wit.edu

Abstract

On-chip networking concepts, which are central to multicore microprocessor design, are often taught using textbooks and the standard lecture model, as it is difficult to provide interactive learning opportunities and hands-on assignments. Available on-chip network simulators typically focus on research-level accuracy and are not suitable for novices or students. Meanwhile, with each new chip generation, the importance of the on-chip interconnect grows. Career opportunities in computer architecture will increasingly rely on an understanding of the chip's communication substrate. The lack of interactive, approachable tools thus leaves students with a gap in their computer architecture education in an increasingly multicore industry.

In this paper, we present ENoCS, the Educational Network-on-Chip Simulator, which allows users of all levels of expertise to explore the on-chip network environment and see the inner workings of the on-chip communication substrate and all of its components. ENoCS contains multiple topologies and network options, as well as multiple traffic patterns for testing. ENoCS also shows microarchitectural details, including router structure, packet breakdown, and routing tables. We also present methods for incorporating ENoCS into an existing computer architecture course curriculum and an evaluation of its effectiveness in a small senior/graduate-level course. Students in the evaluated course who used the tool showed an increased competency in the concepts, as well as interest in using the tool further. While the sample size was small, it demonstrates the promise of using the ENoCS tool in the classroom. The simulator is available on-line for public use at *https://github.com/ProfACarpenter/ENoCS*.

1 Introduction

College-level courses are always improving the learning experience for their students by incorporating new technology into the classroom. On-line courses and on-campus course curricula increasingly rely on the development of a growing library of available tools. Computer engineering curricula are no exception, as a number of publicly available simulators have been

developed for various levels of expertise, ranging from hobbyists to researchers. These tools are invaluable in providing computer architects hands-on experience with concepts that are otherwise difficult to physically implement. Computer architecture simulators often require significant coding skill and experience, and thus, novice students have trouble using the available simulators. This limits their utilization and dissuades students from trying to gain deeper understanding of computer architecture.

There is a wealth of existing computer organization tools (see Section 2), such as pipeline modeling and simple cache organization, typically focusing on single-core systems. However, modern microprocessors incorporate a growing number of cores into a single chip or system[1]. Educational simulation tools for chip multiprocessors are less available, leaving a significant gap in the hands-on experience one can gain in the classroom without learning complicated and cumbersome research-level simulators. This gap leaves students at a disadvantage in the computer architecture job market as well as in graduate research.

We propose *ENoCS*, an interactive Educational Network-on-Chip Simulator. ENoCS provides the user with an interactive medium which contains detailed behavior of the multicore communication substrate. With ENoCS, users can simulate how packets travel through a variety of on-chip network topologies and configurations, gather data and statistics on each system, and view the inner workings of the router pipeline. ENoCS requires no prior knowledge of networks, programming, or even computer architecture, allowing users of all levels of expertise to use it. It also provides enough detail to (a) be useful to more experienced users and (b) teach the user how the network operates, reinforcing concepts and details discussed in textbooks and classroom lectures. ENoCS's visual, interactive, and detailed simulation models provide students with the experimental experience often lacking in modern microprocessor design curricula. ENoCS is available to the public.

In this paper, we describe the ENoCS program and present a small-scale evaluation from a graduate-level computer design. The evaluation shows a significant improvement in retention of the concepts of networks-on-chip as well as student confidence in their understanding.

The rest of the paper is organized as follows. A brief discussion of existing simulators is presented in Section 2. Section 3 presents the details of ENoCS and discusses a brief evaluation from a graduate course in the fall semester of 2014. Section 5 discusses on-going and future work related to ENoCS. Section 6 concludes.

2 Related Work

Simulators for computer engineering education and research range from simple pipeline components to full-system emulation and simulation. However, most educational simulators are aimed at single processor concepts, and few expand to include multiprocessors concepts, despite the shifting focus in the microprocessor development community. Research-centric simulators cover subjects like cache coherence and network performance, but are overwhelming for the classroom environment, particularly for students not accustomed to large-scale programs. Table 1 shows a number of widely-used, relevant simulator tools used both in computer architecture

Simulator	Central Topic	Intended Use
Gem5[2], Simics[3], Multi2Sim[4], Sniper[5]	All computer systems & architecture	Research and development
SimpleScalar[6]	All computer architecture	Educational (sometimes used for research)
CACTI[7]	Caches	Research and development
Garnet[8], Topaz[9]	On-chip interconnects (integrable with Gem5)	Research and development
SMPCache[10]	Cache coherence and multiprocessors	Educational Tool
WinMIPS64[11]	MIPS Pipelines	Educational Tool
HASE[12]	Computer architecture	Educational Tool
NS3[13]	Network simulation	Research and Education
Noxim[14], Booksim[15]	Networks-on-chip	Research and development

Table 1: A brief table of significant simulators, their central focus, and intended use. More simulators are available for both education and research[16], but none focus on on-chip interconnects for the purpose of education.

research and education. It is worth noting that research simulators can be used for coursework, but with suboptimal educational outcomes.

Education-specific computer architecture simulators Many educational simulators focus on the execution pipeline and surrounding logic[12,17,11,18,19], others illustrate cache concepts[7,10,20], and a smaller number explore multi-core concepts[10]. There are also smaller tutorial programs to help teach simple concepts with a collections of small-scale animations[21]. The ENoCS GUI was modeled partly after WinMIPS64[11], which simulates the MIPS 5-stage pipeline. WinMIPS64 is an openly available program with an easy-to-use interface. ENoCS represents an addition to this field, but in an area not adequately covered. For students learning parallel systems and chip multiprocessors, ENoCS represents the first of its kind visual, interactive network-on-chip simulator.

General computer architecture simulators There are a number of architectural simulators[2,6,3,22,23,24,5,4], and specifically network-on-chip simulators[8,9,13,14,15], used mainly for research. While these are useful in some respects for both research and education, they have a steep learning curve and often take months or years to fully understand. These simulators typically also require significant programming knowledge, which makes it difficult for beginners to use. ENoCS does not suffer from these setbacks, as it is designed to be simple and approachable for all levels of computer engineers. There are siginificantly more simulators available for both student and researcher use. A more complete listing can be found in previously published literature[16].

It is worth noting that while each of the simulators offer a set of advantages in terms of accuracy, scalability, extendability, ease-of-custom, and beyond, or offer simple interfaces for beginners,

Network Setting	ENoCS Options	Comments
Topologies	Mesh, Torus, Ring, Bus, Flattened Butterfly	Commonly studied topologies; Latencies for links can also be changed for each topology
Traffic Patterns	Uniform Random, Tornado, HotSpot	Hotspot node can also be specified
Injection Rate	0-100%	Probability packet/flit injected per node per cycle.
Number of Nodes	4, 9, 16	Mesh, torus, and flattened butterfly will be in a conventional grid layout
Flow Control	Wormhole, Virtual Channels	Only flit-based control is used in this version.
No. of Router Stages	From 1 to 5 (4-stage default)	Smaller numbers pipeline stages will combine stages, only for instructional purposes. This does not faithfully model each stage.
Various Sizes	Buffer, Flit, Phit, Packet	Number of buffer entries; number of bits per physical channel, flit, and packet.

Table 2: Available simulator settings. More will added in future versions of ENoCS.

none is optimal for any one application. ENoCS is not intended to be scalable to large numbers of cores or be cycle-precise. It is intended to provide an easy-to-use interface for students to understand the basic concepts of networks-on-chip, a component of increasing importance.

3 Description of the ENoCS

ENoCS is a flexible simulation environment, with a number of topologies, configurations, and algorithms from which to choose. These options are chosen to provide students with insights into the fundamental concepts of on-chip network design. Future work will expand the available configurations. For brevity, only brief descriptions of each characteristic are given here.

3.1 Network Features

ENoCS provides flexibility in its network, traffic, and router characteristics. These particular options are integrated to provide a sufficient amount of flexibility without overwhelming a novice with too many details. Table 2 contains an overview of the main options available for ENoCS. These options can be set by the user using drop-down menus within the program. The definitions and details of each option are considered outside the scope of this paper.

3.2 Graphical User Interface

The main window, which provides a comprehensive view, provides global control and configurability of the simulator. The general graphical interface is shown in Figure 1. Each

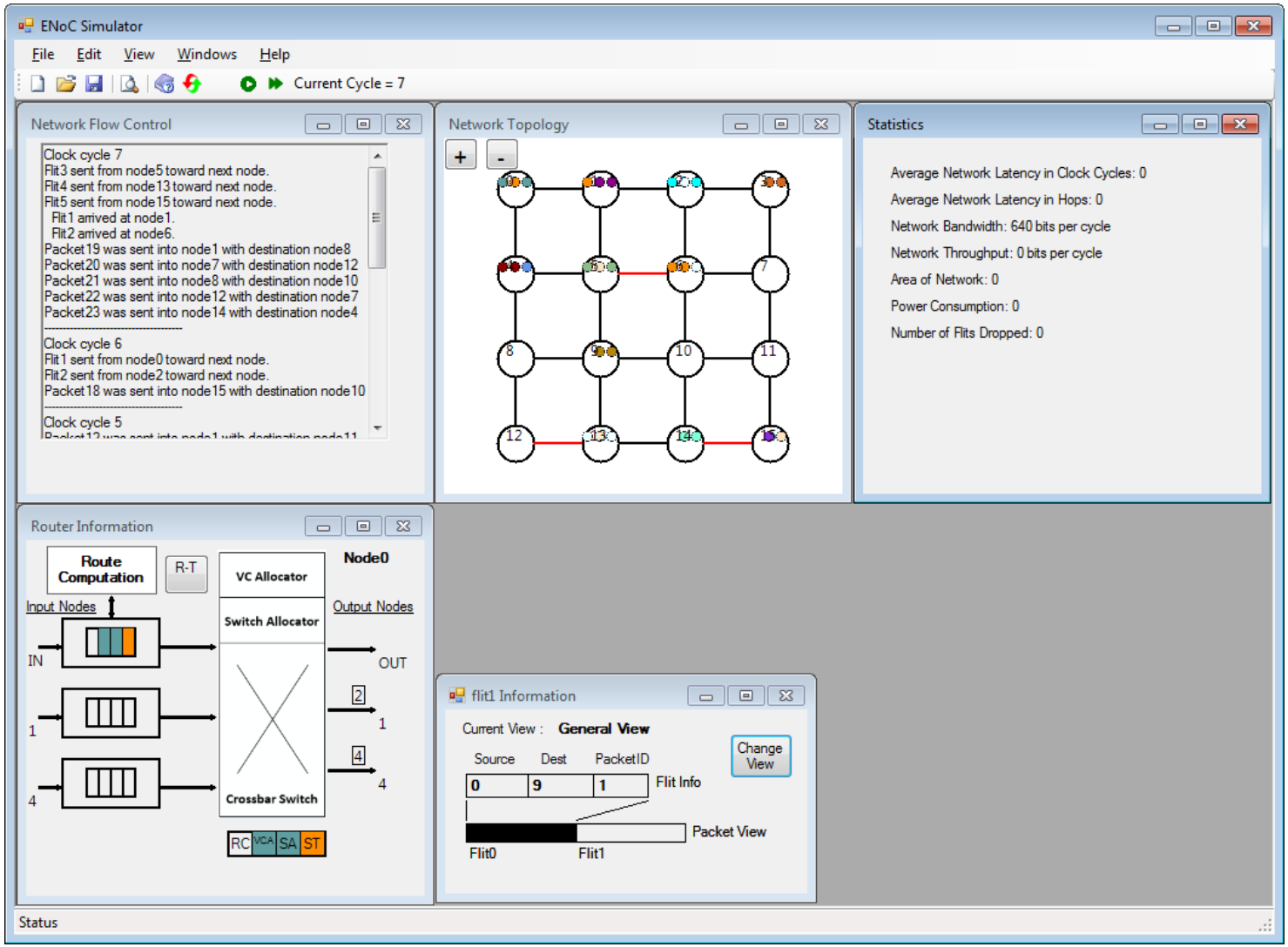

Figure 1: The full ENoCS graphical user interface, including the full topology view, packet history, statistics, and router pipeline. Note that the "Change View" button allows the user to show more or less detail of the packet as desired.

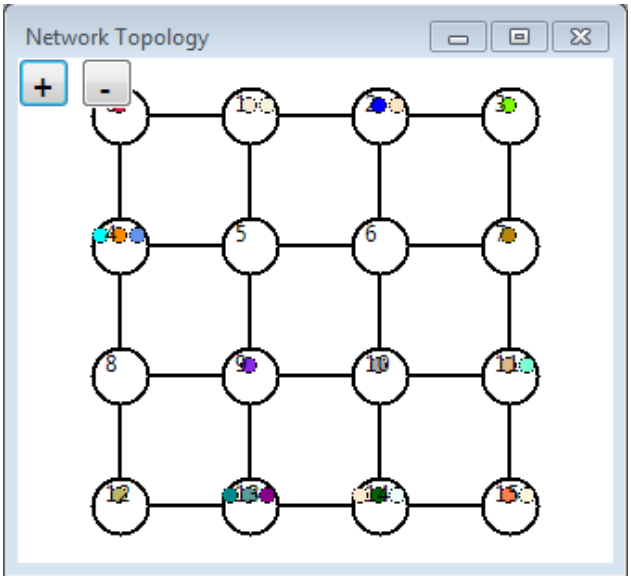

Figure 2: A detailed view of the network topology window, in this case showing a mesh.

window is synchronized to reflect the current state of the entire network. The internal frames in ENoCS show various levels of detail:

- *Network Topology*: Illustrates the routers and links and shows the movement of packets/flits through the chosen topology.
- *Router Information*: Shows the microarchitecture of the router. The buffers in the router fill with color-coded packets (colors match the packets/flits in the topology window). The router stages are also shown and color-coded appropriately.
- *Network Flow*: Tracks each packet from injection to ejection in a text window.
- *Statistics*: Gives run-time statistical analysis of network behavior. This includes per hop latency, total latency, network bandwidth, and bi-section bandwidth.
- *Flit/Packet Information*: Illustrates the content/size of each packet/flit (accessible through the Router Information window).

Global window The global window contains the typical controls for an application, including access to the properties windows, viewing preferences, help windows, and save/load operations to keep the same simulation over multiple sessions. There are also *reset* and *run* controls and a current cycle counter, which control the state of the current simulation.

Network Topology The network topology window displays the nodes in the system and their interconnectivity based on the user-input properties. The illustration of the topology shows the links and the router ID numbers. Additionally, while the simulation is running, the packets currently in the system are displayed in varying colors to allow the user to follow a packet or flit through the system while the simulation is running. This is illustrated in the Router Information

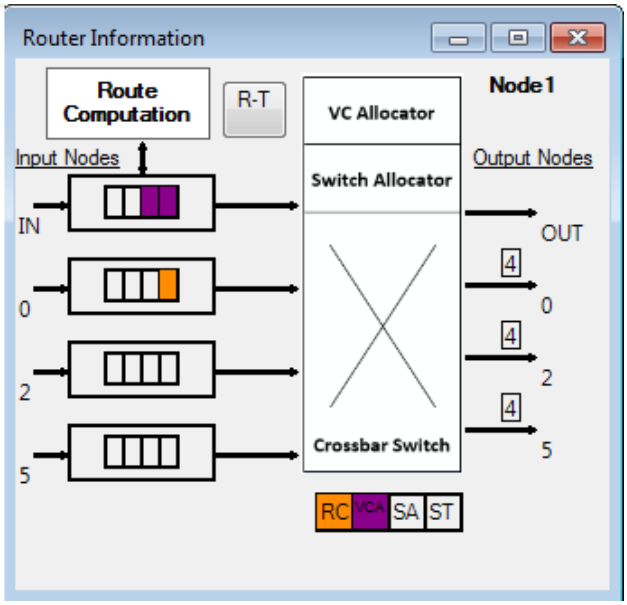

Figure 3: A detailed view of the router information window. The numbers in the boxes represent the credits for the downstream router and the numbers next to the links represent the ID of the connected router.

and Network Topology windows in Figure 1. These windows not only give students a quick reference for topology layouts, but will help to demonstrate routing algorithms, link and router operation, and router interactions. ENoCS can currently simulate a mesh, torus, flattened butterfly, bus, and ring topologies containing 4, 9, and 16 nodes, based on user preferences. Without even running the full simulation, users can see how topologies are formed and interconnected, providing one of the key concepts of networks-on-chip.

ENoCS contains a number of synthetic traffic generation schemes, including uniform random traffic, tornado, and HotSpot. HotSpot also has the option of specifying which node is considered the hotspot and will receive the majority of the traffic. The probability of injection at each node per cycle is specified in the properties. This type of analysis is critical to understanding the performance of a network in isolation.

In addition to being the central illustration for the default view of ENoCS, the nodes and links can be clicked to reveal further details for further scrutiny. The links, when clicked, will reveal which nodes are connected using that wire. When the nodes/routers are clicked, ENoCS will open a new sub-window, called Router Information.

Router Information The Router Information window illustrates the inner workings of a traditional NoC router. Router microarchitecture is an important area of focus for network-on-chip research and development, specifically in the microarchitectural design of the hardware, including the switch and pipeline stages. The diagrams in ENoCS, as well as the visual movement of packets through the network and router stages, provide users with an interactive environment to strengthen their understanding of the concepts.

The Router Information window contains details of the specific router architecture, including input and output ports, crossbar, pipeline stages, virtual channels (if applicable), and a routing table (RT), which shows the packet routing control for the particular algorithm in a separate window. X-Y routing is currently implemented, but more will be added in future versions of the software. This window is synchronized to the other windows, and shows the progress of the individual packets through the router pipeline and position in the buffers.

The router has a number of flexible options. Both wormhole and virtual channel architectures are possible for flit-based flow control, each with variable buffer sizes. The router pipeline can also be adjusted to a variable number of stages, with the default being 4-stages (route computation, virtual channel allocation, switch allocation, switch traversal). If adjusted to be other than the fixed 4-stages, the router architecture is only mimicked, and may not accurately model the inner workings of the router. In this case, it is assumed that router pipeline stages overlap, but the mechanisms for overlap are not discussed or illustrated. This is not critical to basic understanding of how the router functions, but provides students with insight into the upper bounds of microarchitecture impact on overall performance.

The flits within the router flow through the buffer and pipeline stages in color, matching the network topology window, so the user can see as much detail as needed. Multiple Router Information windows can be opened simultaneously. This is especially useful for non-uniform topologies. For example, the mesh topology has routers with 2 to 4 links. The number of input and output ports has a noticeable impact on the router's behavior. Providing multiple windows lets students visualize these differences during a running example.

Flit/Packet Information Within the Router Information window, the user can click on the flit while it is in the router pipeline and examine the details of the flit, including the source, destination and size of the packet/flit. It also demonstrates the place of a flit within a packet (head, body, or tail).

Network Flow The Network Flow window contains a complete text history of each packet/flit injected into the network, including injection, traversal from router to router, and arrival at the destination. The user can then follow the packets from source to destination without monitoring the animation in the Network Topology window. This provides users with the ability to look at a packet's past behavior without needing to restart the simulation. Additionally, the history provides a secondary method for learning the behavior of the network. The flits are marked with ID tags for easy tracking and clock cycles are included to mark time. Future versions of the software will include the ability to save this directly to a file for future parsing and examination.

Statistics Networks are evaluated mainly on two major performance metrics: latency (in terms of both hops and cycles) and total throughput. While the utilization of the throughput fluctuates, the available throughput is static, based on the topology and physical wire design. The latency, on the other hand, is dependent on the run-time characteristics, such as buffer occupancy, source-destination pairs, link availability, and traffic congestion. ENoCS calculates the average latency and hop count for all packets that have reached their destination. This allows the user to

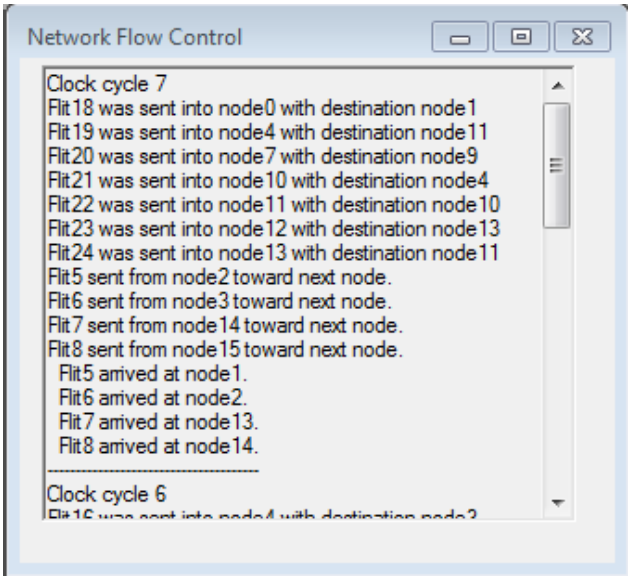

Figure 4: A detailed view of the network flow history window.

manipulate network properties and easily reveal the impact of these changes in real-world scenarios. Thus, it provides a reliable reference to compare network topologies. Future versions of ENoCS will also contain other metrics, such as bisection throughput, power counters (given some average switching activity), and more.

4 Integration of ENoCS into a Curriculum

ENoCS is intended to supplement or replace a traditional lecture-based curriculum. Here we will review some proposed uses for ENoCS as a supplemental tool in a computer architecture course, with suggested use and examples of assignment questions. ENoCS version 1.0 includes basic packet-switched network functions, so the assignments discussed here focus on the basics of network instruction, rather than the more advanced functionality. As such, it is recommended that courses that adopt ENoCS use it only as a supplement to traditional lectures. Future versions of ENoCS will include an in-depth tutorial that will include definitions, short audio/video lectures, and examples in order to help students use the software in isolation and still learn the same concepts effectively.

Within the structure of basic on-chip interconnect curricula, there are number of key concepts: network topology, router architecture, packet/flit construction, routing algorithms, and bandwidth/throughput. The instructor in charge of using ENoCS can give simple exploration assignments, comparing the various traffic patterns and injection rates for various topologies. To test ENoCS in the classroom, such an assignment was created. The assignment was available for 1 week, and the software was released to half the students for the entire week (and beyond). It included the following questions:

1. How does injection rate impact the average latency of a network?

2. Describe the valid paths a packet can take from Node 0 to Node 15 and from 15 to 0 using X-Y routing.
3. How does a torus router differ from a mesh router?

Each of these questions addresses foundational concepts of basic NoC design. These questions could also be answered without using the software, but as our study showed, ENoCS reinforced the concepts and helped students retain the information more effectively.

To further gauge the usefulness of ENoCS, the study included a quiz and survey. The quiz was held in class for approximately 15 minutes, and no additional materials were granted, in order to determine retention of the material. The quiz included the following questions:

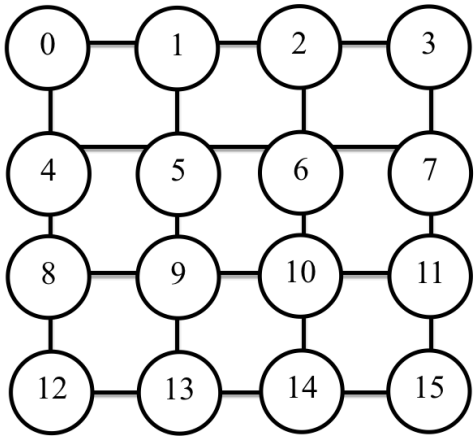

Figure 5: 16-node mesh for evaluative quiz.

1. (3 points) Given the network below (Figure 5), draw valid paths from 0 to 10 and 13 to 3, assuming X-Y routing.
2. (4 points) Describe the basic stages of a router.
3. (3 points) What is meant by ”traffic pattern” in network evaluation?

While the questions from the sample assignment and quiz were aimed at evaluating the ENoCS effectiveness, they could be used as a preliminary assignment for a real course. The survey asked how difficult the assignment and quiz were and asked the student to determine how much help the software provided (if they used it).

To evaluate ENoCS, the above assignment, quiz, and survey were given to an upper-level technical computer architecture course that included 5 senior undergraduates and 27 graduate students. To clarify, of the 27 graduate students, 12 were registered as graduate students as part of a BS/MS program. All students were part of the Electrical and Computer Engineering department pursuing BS, MS, and PhD degree in ECE. The entire course was 13 weeks, with a 2-3 week module on on-chip interconnects toward the end of the fall semester.

Half of the students (chosen at random) were given the software, while the others were not. Not all students completed the assignment, as it was considered optional to participate, so the sample

size is admittedly smaller than desired. From the students who completed the assignment, both with or without the software, the grades were approximately the same. The questions chosen were foundational, and therefore could be answered using any number of available resources (*e.g.*, Internet searches, textbooks, lecture notes).

All students took the in-class quiz, whether they had participated previously or not. When the quiz was completed, the students who had used the software scores approximately 3.2 (out of 10) points higher (without ENoCS: 1.3/10, with ENoCS: 4.5/10), with a significant number of students without ENoCS getting a 0. This shows that the use of the software helped students better retain the information and concepts they had learned. More specifically, question 1 was meant to show basic routing algorithm understanding, but no student that did not use ENoCS was able to answer appropriately and with proper justification. Meanwhile, students who used ENoCS demonstrated a better understanding. For question 2 regarding stages of the router, the students who did not use ENoCS could not identify any router stages, while the students with ENoCS correctly identified an average of 2-3. For question 3, 6% students who did not use ENoCS correctly defined traffic pattern, while 50% of the students with ENoCS answered correctly. The scoring difference between graduate and undergraduate students was negligible.

The survey results were equally encouraging. Of the students who did not use the software, only one said he/she would not use the software if made available. The rest showed either indifference or eagerness to use ENoCS to supplement the textbook and lecture slides available. Of the students who use the software, 75% claimed that the software was helpful and they would use it again, while the remaining 25% claimed that they already fully understood the concepts, and thus the software would not have helped with the assignment or quiz. These students were not against the use of the software, but were among the best students in class and already had a solid understanding of the concepts. No student commented that they disliked the software. Upon the quizzes completion, all students were allowed to use the software, with many commenting that future instances of the course should include the software in a larger portion of the assignments.

While this study is limited to a small number of students (approximately 30, with about 1/3 responding to all assignments and all finishing the quiz), the results are nonetheless encouraging. For the students who used ENoCS, they largely found it informative, easy-to-use, and helpful towards their education. The quiz grades reflect a increased retention of the concepts, as the quiz was conducted a week after the assignment was due. We predict that with further refinement and more exposure and familiarity with the program, students' education would be further enhanced by including ENoCS as a part of the curriculum. We intend to continue these tests as new versions of ENoCS are developed and released. The further tests will also give more insight into the effectiveness of the tool across multiple course iterations.

5 On-going Work

The current version of ENOCs is functional and has proved useful. However, it is under continued development for the foreseeable future. Future versions of ENoCS will include additional functionality, including additional topologies (*e.g.*, indirect networks and clos topologies), flow

control methodologies (*e.g.*, packet-based store-and-forward and cut-through), statistics (*e.g.*, power consumption), and routing algorithms (*e.g.*, adaptive routing). All of these features will provide a more complete view of the on-chip network design space and thus make it a more capable teaching tool. As the tools is used by more people, feedback will help to contribute to the direction of future versions, with more students and faculty becoming involved in the design.

The overall goal of the ENoCS development work is to provide a self-contained, self-sufficient learning tool. For future releases, ENoCS will include educational materials such as in-depth help files, short lectures, and definitions. For example, the user would be able to select "X-Y routing" and have access to additional information in the form of diagrams, short video/audio lectures, and definitions of various routing algorithms. Additionally, network tutorials will be created to guide a novice user through the appropriate levels of detail for their education, starting at the basic functions of an on-chip network, and showing examples of each property or characteristic as necessary. Overall, ENoCS will be a stand-alone educational simulator, suitable for students and users of all levels of expertise, thus preparing them for careers in multiprocessor design.

Further, we will continue to evaluate the ENoCS tool in additional undergraduate classes and adapt the functionality and interface to better suit the needs of the students.

6 Conclusions

Current advanced computer architecture courses incorporate new software tools to provide a more hands-on, interactive learning experience for students across expertise levels. The Educational Network-on-Chip Simulator (ENoCS) adds to this growing pool of software, focusing on the on-chip interconnect, a previously uncharted area for educational simulators. The ENoCS interface allows users to see interactive examples of network topologies, buffer sizes, and routing and flow-control algorithms. This paper describes the graphical interface and available options, as well as a suggestion for incorporating the program into a computer architecture course. The effectiveness of ENoCS is evaluated in a graduate-level course and is found to not only improve the students' understanding of networks-on-chip, but also improve their confidence and satisfaction in the coursework. The code can be found on github at:
https://github.com/ProfACarpenter/ENoCS

References

[1] M.D. Hill. Amdahl's law in the multicore era. In *High Performance Computer Architecture, 2008. HPCA 2008. IEEE 14th International Symposium on*, pages 187–187, Feb 2008.

[2] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad

Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. The gem5 simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7, August 2011. ISSN 0163-5964. doi: 10.1145/2024716.2024718. URL http://doi.acm.org/10.1145/2024716.2024718.

[3] P. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hållberg, J. Högberg, F. Larsson, A. Moestedt, and B. Werner. Simics: A full system simulation platform. *Computer*, 35(2):50–58, February 2002.

[4] R. Ubal, B. Jang, P. Mistry, D. Schaa, and D. Kaeli. Multi2sim: A simulation framework for CPU-GPU computing. In *Int'l Conf. on Parallel Architectures and Compilation Techniques*, pages 335–344, 2012.

[5] T. Carlson, W. Heirman, S. Eyerman, I. Hur, and L. Eeckhout. An evaluation of high-level mechanistic core models. *ACM Transactions on Architecture and Code Optimization (TACO)*, 2014.

[6] Doug Burger and Todd M. Austin. The simplescalar tool set, version 2.0. *SIGARCH Comput. Archit. News*, 25 (3):13–25, June 1997.

[7] CACTI. Cacti: An integrated cache and memory access time, cycle time, area, leakage, and dynamic power model, 2008. URL http://www.hpl.hp.com/research/cacti/. http://www.hpl.hp.com/research/cacti/.

[8] N. Agarwal, T. Krishna, L. Peh, and N.K. Jha. Garnet: A detailed on-chip network model inside a full-system simulator. In *Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on*, pages 33–42, April 2009. doi: 10.1109/ISPASS.2009.4919636.

[9] P. Abad, P. Prieto, L.G. Menezo, A. Colaso, V. Puente, and J.-A. Gregorio. Topaz: An open-source interconnection network simulator for chip multiprocessors and supercomputers. In *IEEE/ACM Int'l Symp. on NoCS*, pages 99–106, May 2012.

[10] M. Vega-Rodriguez, R. Martin, and F. Gallardo. Smpcache: Simulator for cache memory systems on symmetric multiprocessors, January 2006. URL http://arco.unex.es/smpcache/. http://arco.unex.es/smpcache/.

[11] M. Scott. Winmips64, April 2012. URL http://indigo.ie/ mscott/. http://indigo.ie/ mscott/.

[12] HASE. Computer architecture simulation and visualisation: Documents about hase and simjava, July 2006. URL http://www.icsa.inf.ed.ac.uk/research/groups/hase/. http://www.icsa.inf.ed.ac.uk/research/groups/hase/.

[13] Ns3, March 2015. URL https://www.nsnam.org/. https://www.nsnam.org/.

[14] Noxim noc simulator, March 2015. URL http://noxim.sourceforge.net/. http://noxim.sourceforge.net/.

[15] 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 *In the Proceedings of IEEE ISPASS*, pages 86–96, April 2013.

[16] B. Nikolic, Z. Radivojevic, J. Djordjevic, and V. Milutinovic. A survey and evaluation of simulators suitable for teaching courses in computer architecture and organization. *IEEE Transactions on Education*, 52(4):449–458, Nov 2009. ISSN 0018-9359. doi: 10.1109/TE.2008.930097.

[17] Littleman. The littleman computer, March 2015. URL http://www.yorku.ca/sychen/research/LMC/. http://www.yorku.ca/sychen/research/LMC/.

[18] R. Silverman. Sc123 (tm) computer system, 2002. URL http://www.cs.csustan.edu/ rrsilver/html/sc123.html. http://www.cs.csustan.edu/ rrsilver/html/sc123.html.

[19] Dale Skrien. Cpu sim: An interactive java-based cpu simulator for use in introductory computer organization

classes, July 2013. URL http://www.cs.colby.edu/djskrien/CPUSim/. http://www.cs.colby.edu/djskrien/CPUSim/.

[20] J. Edler and M. Hill. Dinero iv trace-driven uniprocessor cache simulator. URL http://www.cs.wisc.edu/ markhill/DineroIV. http://www.cs.wisc.edu/ markhill/DineroIV.

[21] Gurpur M. Prabhu. Computer architecture tutorial, 2006. URL http://www.cs.iastate.edu/˜prabhu/Tutorial/title.html. http://www.cs.iastate.edu/prabhu/Tutorial/title.html.

[22] P. Stanley-Marbell. The sunflower tool suite. URL http://sflr.org/. http://sflr.org/.

[23] M. Perner. Mikrocodesimulator mikrosim 2010: the ultimate cpu-simulation program, November 2013. URL http://www.mikrocodesimulator.de/index_eng.php. http://www.mikrocodesimulator.de/index_eng.php.

[24] CSIM. Atl programs. URL http://www.atl.external.lmco.com/projects/csim/. http://www.atl.external.lmco.com/projects/csim/.