A Highly Parameterizable Simulator for Performance Analysis of NoC Architectures

Dhiman Ghosh*, Prasun Ghosal*, Saraju P. Mohanty**
*Indian Institute of Engineering Science and Technology, Shibpur, Howrah 711103, WB, INDIA
**University of North Texas, Denton, TX 76203, USA
Email: dghosh.cse@gmail.com, prasun@ieee.org, Saraju.Mohanty@unt.edu

***Abstract*—Network, wireless, and multimedia applications executing on embedded chips demand massive data processing with lesser power consumption today. Journey of a new paradigm in the domain of parallel processing - Network-on-Chip (NoC) starts here. But unlike its simpler look both the design and test costs for this kind of real many-core chips are too high. So efficient and accurate performance estimation tools with respect to the real application ASICs are needed for system level optimization and performance analysis in a cost-effective and flexible way. Simulator that allow exploring the best design options for a system before actually building it has been becoming inevitable in system design and optimization flows. Very few simulators have been developed so far addressing such problems. Some of them are popular with its better accuracy and others with a large set of configurable architectural parameters and traffic options. In this paper, a novel GUI based highly parameterizable NoC simulator has been proposed designed using Qt and System C that is capable of handling real embedded workload traces with custom task allocation support for early exploration of application specific Network-on-Chips.**

***Index Terms*—NoC Simulator, Real Traffic, Custom Task-mapping, System C and Qt**

I. Introduction

With the rapidly approaching billion transistors era non-scalable wire delays, errors in signal integrity, and unsynchronized communications have become the primary problems to be dealt with. These problems may be overcome by the use of efficient Network on Chip (NoC) architectures [1], [2]. But it demands high speed data transferring as well as low power consumption.

Having a large architectural alternatives for implementation, the design space of Network-on-Chips is also huge. Along with the increase in number of cores and complexity of topology, hardware design becomes more complicated and time consuming. As the hardware prototype boards are not available until very late stages of system design, the embedded software development also get delayed. The only way out is to rely on simulation on host computers. Therefore efficient and accurate performance estimation tools are needed for system level optimization and early exploration of the performance.

In this perspective the aim of this paper is to suggest a novel simulation system for Network-on-chips. The structure of the paper is as follows. In Section II a brief non-technical idea of the novel contribution of this work has been given where the full technical specification is in Section IV. In between Section III contains description of some related works as well the motivation. Section V contains the experimental details that has been carried out and the results as well. Section VI concludes the paper by giving some important extensions and possible future works of our initiative.

II. Novel Contribution

Development of NoC Simulators solved many of the earlier problems those were inevitable without the early exploration of the system. Another problem knocking at the door now-a-days is the simulation accuracy issue as well as realistic performance estimation [3]–[9]. Very Few simulators have been developed and almost all of those simulates with synthetic workloads that can be far from accurate performance. To the best of our knowledge, till date, no dedicated NoC simulator has been developed so far with dedicated parallel benchmark support such as SPEC CPU-2006, SPLASH-2, or PARSEC. As an alternative to use full system simulators very less steps have been taken [3]. Novelties of the proposed simulator lie in many folds.

- Firstly, the simulator is highly parameterizable.
- Secondly, it can address real workload traces [10].
- Thirdly, it has the flexibility of customized task mapping that will be beneficial both for architecture designers and operating firmware developers.

III. Background and Motivation

Among the very few simulators developed so far just some of them have global acceptance in terms of accuracy, ease of use, and configurability. Noxim [11] is one of the popular and powerful simulators developed using SystemC that accepts a large set of configurable parameters over synthetic traffic pattern and gives output in terms of a wide variety of performance measurement metrics. Other popular simulators include Booksim [12] developed in C++ by Jiang et. al. that has a large set of input parameters but the output measurement practice used is not satisfactory compared to today's technology-centric demand. NIRGAM [13] and Gpnocsim [14] are also popular ones developed in SystemC and Java respectively. As there is no dedicated benchmark developed for NoCs yet, all of them have focused on synthetic traffic simulation. Some other groups have tried to incorporate NoC simulators inside or along with Full-System Simulators to inject live traffics but none of them become popular for their complex usage practice.

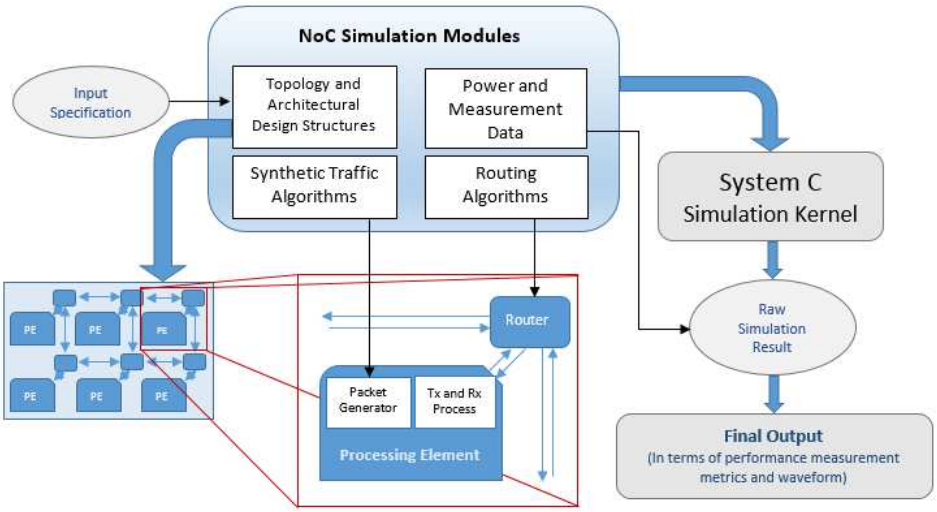

Fig. 1. General Architecture of a System C based NoC Simulator.

NoCTweak [15] is another one recently developed at University of California and probably the first one that supports both synthetic and real embedded application workload traces for real traffic generation that we have adopted in our work for traffic generation. But this simulator supports only a limited predefined task-mapping schemes and also no custom task allocation support which is an essential feature for system application and task-mapping algorithm developers. We have also adopted the basic architectural concept and measurement data as used in Noxim for standardization purpose as well as for comparison.

Figure 1 shows a basic functional block diagram of SystemC based NoC simulator. A sample 3 x 2 2D mesh architecture is considered for illustration purpose. The red square block represents the magnified view of a part of the sample architecture.

IV. Proposed Simulator Model

A. Functional Blocks

Proposed simulator model incorporates typical NoC simulator functional blocks as depicted earlier in figure 1 over the SystemC simulation kernel. Figure 2 shows the additional functional blocks used to incorporate the novel features of the proposed work. The arrows represent flow of data or logic. Input specification includes architectural settings, buffer settings, packet settings, routing settings with custom routing support, hotspot influence settings etc. Final Output is primarily the result in terms of some performance metrics parameters like total received packets/flits, latency, energy and throughput. Optional detailed simulation log lists detailed packet routes, task allocations as well as waveform traces for detailed packet level analysis.

B. System C and the Core Design

The developed simulator is based on System C [16], a C++ class and macro library that provides a simulator interface in C++, which is faster and more flexible than RTL simulators. It helped to make it event-driven and highly parameterizable for analysis in a concurrent way. The architectural components are incorporated in the simulation system in object oriented fashion using separate individual modules. This kind of modular design is very useful for future extension of the work such as incorporating a new topology.

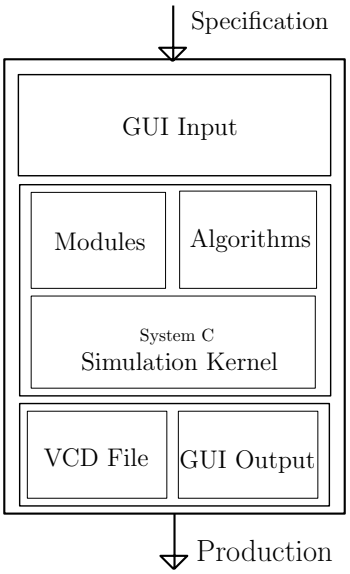

Fig. 3. Simplified Block diagram of our simulator model.

C. Incorporating Real Workloads for Benchmarking

Besides the common synthetic traces, the proposed simulator supports a wide variety of real time embedded application workload traces such as MPEG4 Decoder, WiFi Baseband receiver etc. These embedded application traces are stored in the form of parallel task communication graph and used

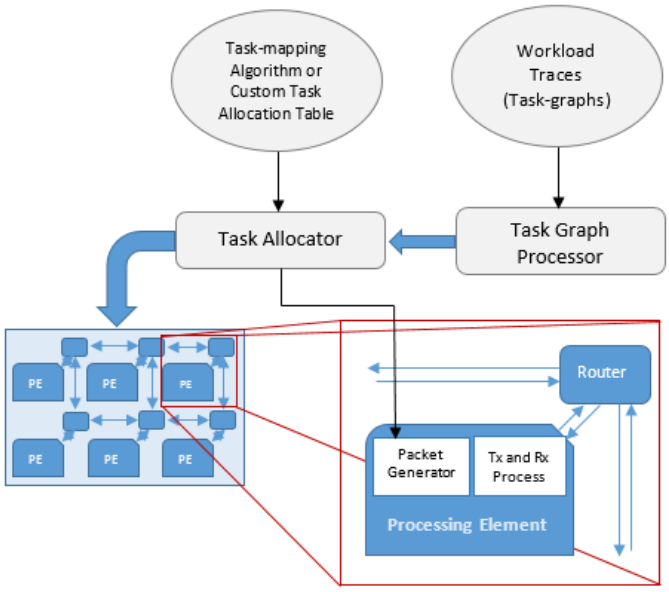

Fig. 2. Additional Functional Blocks Incorporated to handle real workloads and mapping tasks.

for packet generation based on their required bandwidth and global packet injection ratio. This results in better estimation of performance before the physical chip level deployment.

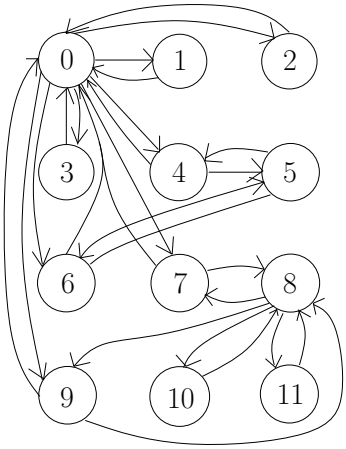

Fig. 4. Task-Graph for MPEG4 Decoder with 12 tasks.

Figure 4 illustrates a simple task-graph of MPEG4 decoder application with 12 parallel tasks. The directed lines show the direction of communication with certain bandwidth between the tasks that are represented by nodes.

D. Qt and the GUI

The proposed simulator interface is developed using Qt [17]. Qt is a cross-platform application and UI framework for C++ and QML developers. This GUI works like a wrapper of the whole tool without which custom configurations, specially the custom task-mapping could be a troublesome job. It also automates several other things and improves ease of use to a large extent considering its large set of configuring parameters. The basic block diagram of our simulation tool is depicted in figure 3.

E. Task-Mapping

Task-mapping is essential for simulating Real application traffics with multiple parallel communications. Each unique communication between any two tasks with a certain bandwidth limit (which is responsible for a certain rate of packet injection in a tile) can be allocated to a core for dedicated processing. Custom mapping of tasks to the cores are supported upto a maximum of 64 parallel tasks to 64 different cores. Traditional task-mapping algorithms like Straight task-to-core, Random, Near-optimal algorithms are also incorporated to measure performance in standard scenario if we are only concerned with other architectural parameters such as different buffer depths, global packet injection ratio etc rather than system software development for task-allocation [18].

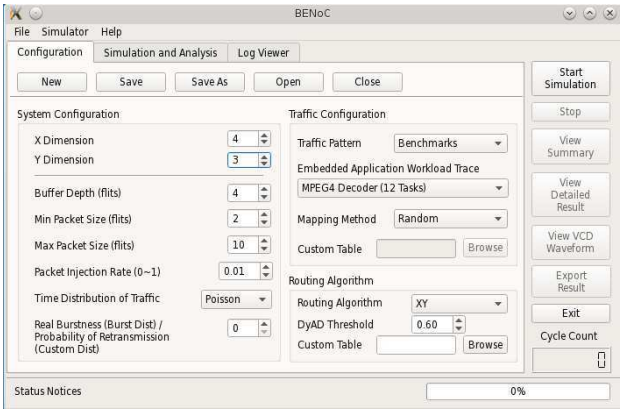

Fig. 5. GUI Screenshot : Architecture Modeling.

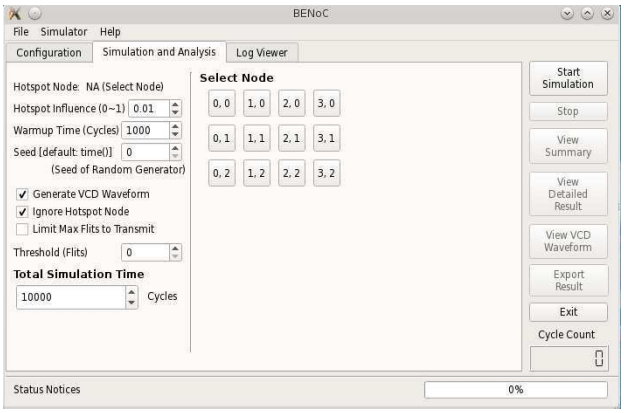

Fig. 6. GUI Screenshot : Advanced Configuration.

F. GUI Screenshots

Two screenshots of the developed simulator have been added in this paper to convey a brief idea about the interface and its ease of use. The interface shown in the figure 5 is the configuration tab containing most of the input parameters and figure 6 is showing the advanced settings tab for all the critical custom operations.

V. Experimental Results

The aim of our experiment is to distinguish the results of different synthetic and real embedded application traffic patterns so that the necessity of performing host computer simulation with real traffics can be emphasized. It may not produce performance estimation as accurate as full system parallel benchmark traffics (as discussed earlier), but the traffic generated using the real embedded application workload traces can give much more clear idea than the synthetic ones before actually building it. The custom task allocation support in this regard will also help in early exploration of performance in the field of system application development.

We have chosen MPEG4 Decoder application task graph with 12 tasks for our experiments shown in the figure 4. $M \times N$ dimension of the architecture is considered where $M \times N \geq L$ for guaranteed parallel execution of the whole task; L is the number of parallel communications for a particular task graph. In our experiment, as we are considering MPEG4 Decoding application, the total number of cores is 12 in optimal case. Therefore, we have considered the $M \times N$ value to be 4×3 (or can be 3×4). The packet size has been fixed between the range 2-10 flits and if not specified, the buffer depth is 4 flits long. XY routing algorithm has been used throughout the whole process. The test simulations were performed for 10000 cycles with a warm up time of 1000 cycles.

A. Impact on Maximum Delay

Figure 7 shows the analysis of different workloads (both synthetic and real). X-axis denotes delay in cycles and Y-axis Packet Injection Rate (PIR) in flits/cycle/tile. Random and Transpose are synthetic traffic models and rest two are real embedded system workloads using Random Task-mapping and Default Task-mapping respectively.

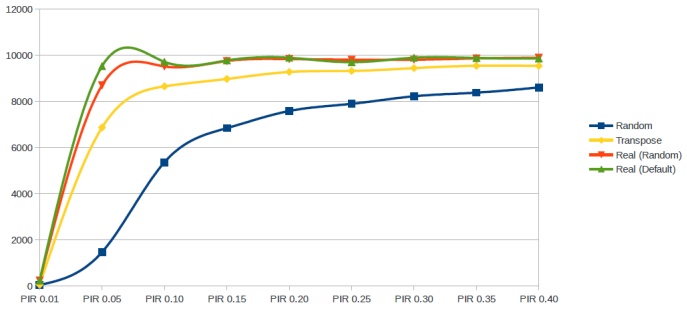

Fig. 7. Maximum Delay vs PIR for different workloads using 4-flit buffers.

From the graph, it is clear that there is a remarkable change not only in numeric difference of cycle count but also in nature of the graph. Unlike traditional S-curve in case of Synthetic Random traffic or almost linearly approaching curve towards threshold of Transpose traffic, a sudden rise in delay against 0.05 PIR (approximately) has been experienced in case of the selected real traffic scenario which is an unavoidable factor to be considered before actual physical level deployment.

B. Impact on Throughput

Figure 8 denotes the impact on throughput for different workloads. Y-axis denotes throughput in $n \times 10^4$ flits/cycle and X-axis denotes the PIR in flits/cycle/tile.

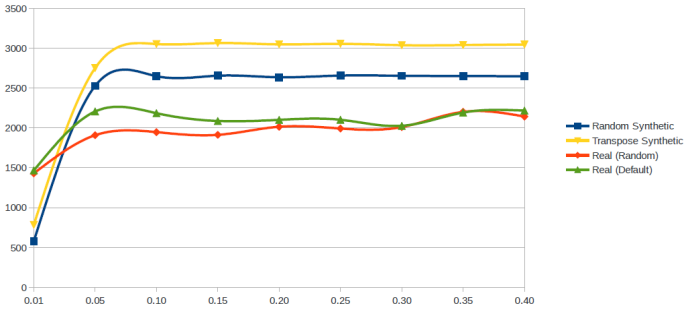

Fig. 8. Throughput vs PIR for different workloads using 4-flit buffers.

The strongest impact experienced is in case of Throughput. Lower PIRs as well as higher PIRs show a complete nature compared to the results from synthetic traffic patterns. Lower PIR performance is poor in case of experiments with real workload traces and a sudden improvement in throughput is also noticed after PIR 0.05 that continues throughout the saturation period.

C. Impact on Latency

Figure 9 denotes the impact on Average Delay/Latency for different workloads. Y-axis denotes Average Delay in cycles and X-axis denotes the PIR in flits/cycle/tile.

There is no severe change in the shape of the curve compared to synthetic traffic in this case. But higher average delay is experienced throughout the session.

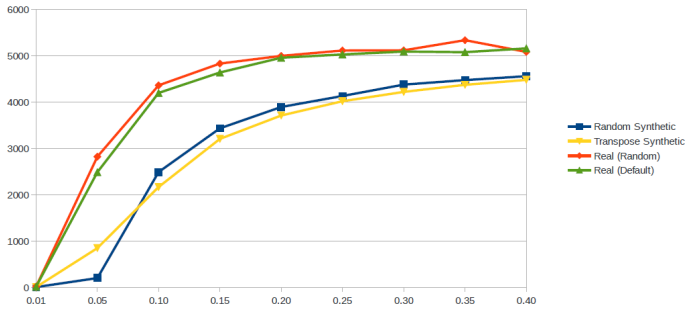

Fig. 9. Average Delay vs PIR for different workloads using 4-flit buffers.

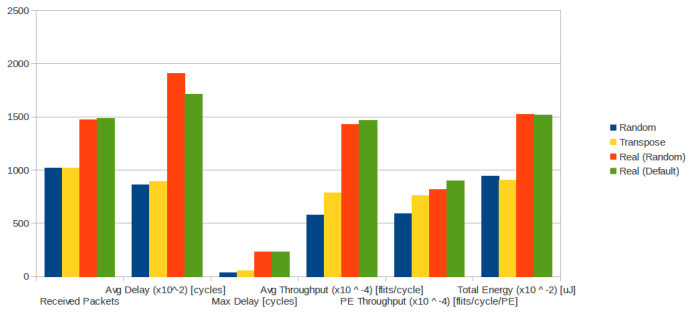

Fig. 11. Summary of Results using 4-flit buffers and 0.01 PIR.

D. Impact on Energy

Figure 10 denotes the impact on Total Energy for different workloads. Y-axis denotes Average Delay in Micro Jule (μJ) and X-axis denotes the PIR in flits/cycle/tile.

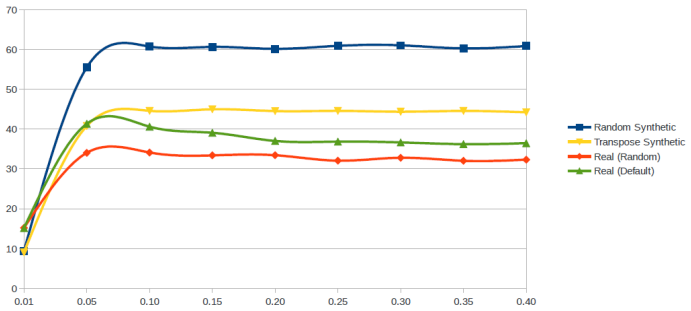

Fig. 10. Total Energy vs PIR for different workloads using 4-flit buffers.

Similar to the last result, the curve structure is also similar to the one corresponding to synthetic traffic. But a notable gap in energy consumption is experienced for higher PIRs in case of real workload traces which is undoubtedly a very important key factor to be considered before physical level design.

VI. Conclusion and Future Work

In this work, we have developed a highly parameterizable GUI based simulator based on System C and Qt for early exploration of performance before actually building it. Its support in custom task-to-core mapping and real embedded traffic simulation made it an useful tool both for chip level designers as well as operating software developers. Our experiment demonstrate the importance of considering real traffics over the synthetic ones as the performance differs by a large extent (in lower global PIRs) for the same configuration and thus the fruitfulness of this work.

However, the current development is based on Mesh topology only. Incorporating full system benchmark traffics over mesh as well as extension of the present work for other topologies may be considered as the future work.

References

[1] H. T. Axel Jantsch, *Network on Chip*. Kluwer Academic Publishers, 2004.

[2] L. Benini and G. D. Micheli, "Networks on Chips: A New SoC Paradigm," *IEEE Computer*, vol. 35, pp. 70–80, January 2002.

[3] F. Trivino, F. Andujar, F. Alfaro, J. Sanchez, and A. Ros, "Self-related traces: An alternative to full-system simulation for nocs," in *High Performance Computing and Simulation (HPCS), 2011 International Conference on*, pp. 819–824, July 2011.

[4] M. R. P. Jaison Valmor Bruch and C. A. Zeferino, "BrownPepper: a SystemC-based Simulator for Performance Evaluation of Networks-on-Chip," in *Embedded and Distributed Systems Group, University of Vale do Itaja UNIVALI, So Jos, BRAZIL*, 2009.

[5] J. Owens, W. Dally, R. Ho, D. N. Jayasimha, S. Keckler, and L.-S. Peh, "Research challenges for on-chip interconnection networks," *Micro, IEEE*, vol. 27, pp. 96–108, Sept 2007.

[6] A. J. R Thid, M Millberg, "Evaluating NoC communication back-bones with simulation," in *21st Norchip Conference*, 2003.

[7] L. Mieszko, S. K. Sup, C. M. Hyon, R. Pengju, K. Omer, and D. Srinivas, "DARSIM: a parallel cycle-level NoC simulator," 2010.

[8] C. Jueping, H. Gang, W. Shaoli, Y. Lei, L. Zan, and H. Yue, "Opnecsim: An efficient simulation tool for network-on-chip communication and energy performance analysis," in *Solid-State and Integrated Circuit Technology (ICSICT), 2010 10th IEEE International Conference on*, pp. 1892–1894, Nov 2010.

[9] R. Al-Badi, M. Al-Riyami, and N. Alzeidi, "A parameterized noc simulator using omnet++," in *Ultra Modern Telecommunications Workshops, 2009. ICUMT '09. International Conference on*, pp. 1–7, Oct 2009.

[10] E. Pekkarinen, L. Lehtonen, E. Salminen, and T. Hamalainen, "A set of traffic models for network-on-chip benchmarking," in *System on Chip (SoC), 2011 International Symposium on*, pp. 78–81, Oct 2011.

[11] F. Fazzino, M. Palesi, and D. Patti, "Noxim: Network-on-chip simulator," *URL: http://sourceforge. net/projects/noxim*, 2008.

[12] N. J. et al., "Booksim interconnection network simulator," in *Performance Analysis of Systems and Software (ISPASS), 2013 IEEE International Symposium*, 2013.

[13] L. Jain, "NIRGAM: A Simulator for Interconnect Routing and Modelling," June 2012.

[14] H. Hossain, M. Ahmed, A. Al-Nayeem, T. Islam, and M. Akbar, "Gpnocsim - a general purpose simulator for network-on-chip," in *Information and Communication Technology, 2007. ICICT '07. International Conference on*, pp. 254–257, March 2007.

[15] A. T. Tran and B. M. Baas, "Noctweak: a highly parameterizable simulator for early exploration of performance and energy of networks on-chip," in *Technical Report, VLSI Computation Lab, ECE Department, UC Davis*, July 2012.

[16] A. S. Initiative, "SystemC." http://www.accellera.org/home/, 2014. [Online; accessed 3rd-July-2014].

[17] D. plc, "Qt Project Hosting." http://qt-project.org/, 2014. [Online; accessed 3rd-July-2014].

[18] H. Orsila, T. Kangas, and T. Hamalainen, "Hybrid algorithm for mapping static task graphs on multiprocessor socs," in *System-on-Chip, 2005. Proceedings. 2005 International Symposium on*, pp. 146–150, Nov 2005.