Simulating Future Chip Multi Processors

Faculty Advisor: Dr. Glenn D. Reinman

Name: Mishali Naik

Winter 2007 – Spring 2007

Abstract

With the ever growing advancements in chip technology, chip designers and computer architects are now faced with a new range of challenges. Current generation processors can deliver high performance, unlike the processors of the past decade. The problems that were tackled by researchers in silicon industry five or ten years ago have undergone a paradigm shift. Performance was the focus of the past with no constraints on power and area, but now power and area have become the real constraints. Chip multiprocessors being the trend in chip technology, its important to study and analyze the trade offs between power, area, and performance. On chip interconnects which provide for processor to processor as well as processor to cache communication, occupy a significant factor of the die area, and have a direct impact on the performance. It is important to study what levels of latency can be tolerated by a set of applications, and to investigate how should the interconnect be designed to guarantee a certain level of performance by minimizing power dissipation and also minimizing the area of the chip. Moreover, with the increase in the number of cores and caches on chip an intelligent design of the network on chip becomes the winning factor. By studying communication patterns exhibited by a set of applications, we can gain insight into the various bottlenecks and can lead us to find ways in which the interconnect can be dynamically configured to mitigate the impact of latency on performance.

1 Introduction

Simulators are an important component of architecture studies. Computer architecture simulators are used to model real computer systems. Full system simulators simulate processor cores, caches, memories, interconnect networks, and I/O devices. Architectural simulators are essential for the evaluation of new hardware designs without actually fabricating the design on chip. Computer architects and chip designers first study the performance data by simulating existing systems and then explore new designs by implementing them in the simulator. Debugging a design in simulator is lot easier than debugging the design in hardware.

In order to model future CMP architectures with large number of cores on chip, an architectural simulator is required which can simulate a system in a reasonably small run time. At the same time it should also provide flexibility, so that future extensions to the simulator can be made. The simulator must support both multitasking and multithreading.

MCsim infrastructure that has been developed will be used to explore the space of on chip interconnects, and to study dynamic adaptation techniques such as RF – Interconnect. RF – Interconnect offers a dynamic way of adapting to the varying demands of applications by providing shortcuts for on chip communications. With the use of RF – I, components can be assigned frequencies/codes for communicating with other on chip components. The goal of RF – Interconnect is to alleviate the impact of latency on performance.

This thesis is organized as follows: Related work is presented in Section 2. In Section 3, I describe the MCsim infrastructure that was developed by our group. The results from the exploration are presented in Section 4. In Section 5, I introduce the design of the RF interconnect in CMP's. Finally, conclusion is Section 6.

2 Prior Work

There are a number of simulators available to the architecture community, such as SimpleScalar[14], M5[3], Simics/GEMS[12], and SESC[13]. Each of these simulators has a limitation of its own. SimpleScalar does not support CMP simulations, it also doesn't have the support for the multithreading. Also, SimpleScalar simulates Alpha ISA which is an obsolete ISA. Although, Simics/GEMS does allow us to simulate CMP architectures and it also supports multithreading and multiasking, the simulation run time is very large. SESC, the simulator that we have leveraged in building MCSim has a small run time but the NoC model is not flexible enough to carry out interesting NoC explorations. Lot of researchers in the computer architecture community are studying NoC design space, since it seems to have a significant impact on the performance.

Beckmann and Wood [2] introduced the use of transmission lines for mitigating the impact of the communication latency between L2 cache banks and the cache controllers. They have outlined CMP floorplans optimized for less complex circuitry, where the cache banks reside near the edges of the chip and cache controllers are located in the center of the chip. Transmission lines provides a low latency shortcut between two components distantly located from each other. However, in future CMP's with a large number of cores and cache banks on the die, it is essential to extend such schemes for improving the latency of both core-to-core and as core-to-cache communication. Kumar et al. [9] proposed a shared bus fabric to show that in order to converge to best CMP architecture, it is critical to co-design on-chip components which includes the interconnect and the caches. But scalability seems to be an issue with such a shared fabric when designing it for a chip with more than 100 cores. Cheng et al. [4] propose a heterogeneous interconnect by demonstrating the mapping of communications to different wire implementations while trying to increase performance and decrease the power

dissipation. They have achieved this by observing that different coherence requests have varying needs for bandwidth and latency. None of these studies have considered dynamically configuring the interconnect on flight during an application run. Applications exhibit a wide range of communication patterns, and for future CMP's with large number of cores on chip it becomes very important to be able to dynamically adapt the NoC to the varying communication demands.

Most recent work has considered improvements to both the routers [1, 5, 7, 11] and topology [6] of future NoC designs, in order to alleviate the effect of interconnect latency. These techniques are largely orthogonal to our design, but despite the gains seen by these studies, the fundamental latency of wires makes interconnect alternatives like RF more attractive for future technology generations.

For future CMP's, Kirman et al. [8] have employed optical technology to design a low-latency, high-bandwidth shared bus. But they do not consider the additional benefit achievable from dynamically adapting wavelength allocation to communication demand.

3 Simulator Methodology - Building the simulator "MCsim"

During the first half of this program, I have worked towards building a simulator infrastructure using Super ESCalar (SESC) which is a microprocessor architectural simulator. SESC provides cycle accurate timing model by using MINT for functional simulation. Multiple applications can not be run concurrently in vanilla SESC without actually modifying the application source code. The following figure describes the high-level layout of the SESC simulator.

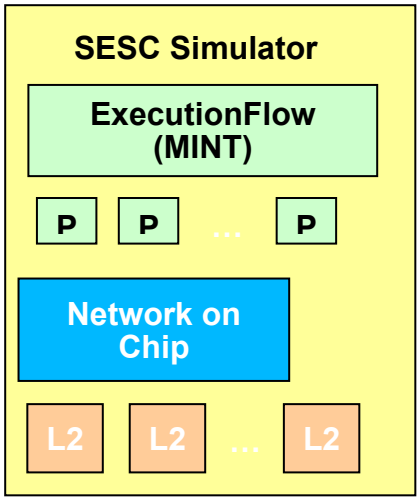


Figure 1 – High level layout of vanilla SESC.

Our goal was to decouple some of the functionality from SESC, and provide support for both

multi tasking as well cooperative multi threading. So the very first step was to study the source code and understand the interaction between the timing model and functional simulation. The interface between the timing model and MINT emulator is well defined, and no physical state is stored as part of the timing model. Since the two are already decoupled, now the question was how do we built in support for multiple applications. After weighing the different options we had, in order to accomplish this goal, we decided to build a system where we would have multiple instances of SESC each running a single application, and a central driver which synchronizes these SESC instances by making use of Internet sockets for process to process communication.

We further divided this goal into following sub-goals:

1. To build in support for simulating multiple SESC instances which are controlled by a centralized driver, where the SESC instances do not share physical memory and hence do not interact with each other.
2. To be able to simulate future CMP-like architectures, by having the multiple SESC instances share physical memory. Here, we subdivided the task into the following:

Extract the virtual memory component from that handles the allocation and eviction of pages and maintains the level 1 and level 2 page tables. This functionality should be supported within the driver process by a "Central Page Handler" (CPH.)

- Extract the shared memory components, L2 cache and memory, and modify the L1 cache to now communicate with the NoC on misses. This required creating an L2 cache, a memory and a NoC Interface within the driver process. The coherence protocol we have modelled is MESI. This phase has been very challenging, involved implementing a sophisticated directory-based cache coherence protocol at the L1 cache and the L2 cache. Evictions of blocks at L2 is quite complex, because of the difference in L1/L2 block granularities.

3. To integrate the SESC instances, L2 banks and the memory using a NoC model. We obtained the COSI NoC, SystemC model from Alessandro Pinto,UC Berkeley. It provides a framework for custom NoC synthesis, given a set of cores and their communication requirements. One of the limitations of using SystemC model currently is application simulation is too time consuming. We have replaced the SystemC NoC model with C++ NoC model to speed up the simulations. C++ NoC models routing tables, queues, packes, link contention but is not synthesizable.

MCSim consists of the following components:

1. A number of cache banks

- Shared cache state that can be accessed by any SESC instance.

2. A central page handler
   - To dole out physical pages SESC instances
   - Allows support for multitasking.
3. A functional network switch
   - To functionally route messages between components.
4. A SystemC NoC model
   - To accurately model latency and power.
5. A number of SESC instances
   - Each SESC instance is a number of cores cooperating on a single application.

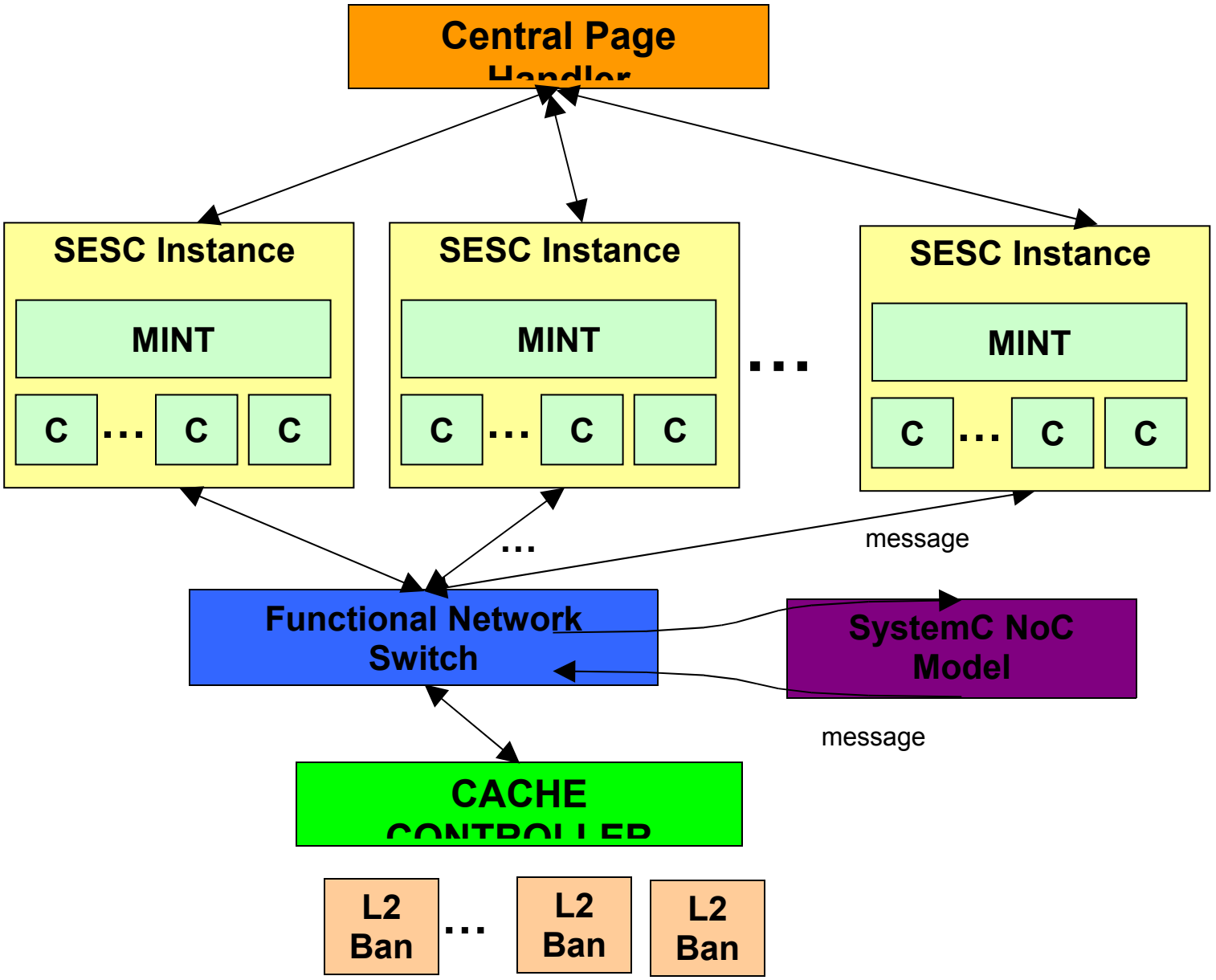


Figure 2. High – level design of MCsim infrastructure.

4 RF - Interconnect

RF(Radio Frequency) interconnect can be used to alleviate the effect of interconnect latency on the performance of a CMP system. RF-I can be leveraged in two ways, first to provide direct shortcuts between two points on a conventional interconnect, and second to dynamically allocate frequency channels to components that communicate more frequently. Applications exhibit wide range of communication patterns, RF-I can be used to speed up the frequent communications observed for an application run. Figure 3, demonstrates the frequency of communication between cores and cache for a Radix run with 8 cores and 16 cache banks. Color closer to red represents more frequent communication, color closer to blue represent less frequent motivates. The horizontal numbers represent the cache banks, and the vertical numbers represent the cores.

CACHE BANKS

CORES

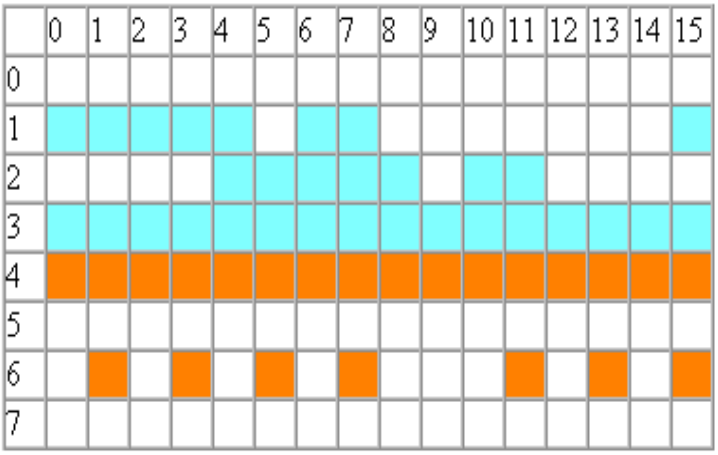


Figure 3 – Communication exploration for RADIX.

This diversity in on-chip communications, motivates the use of RF- I in chip multiprocessors. By dynamically assigning frequency channels to components, RF-I can be used to reconfigure the interconnect for frequently demanded communications.

5 Results

Table 1, describes the configuration used for the simulations. The applications used to evaluate the

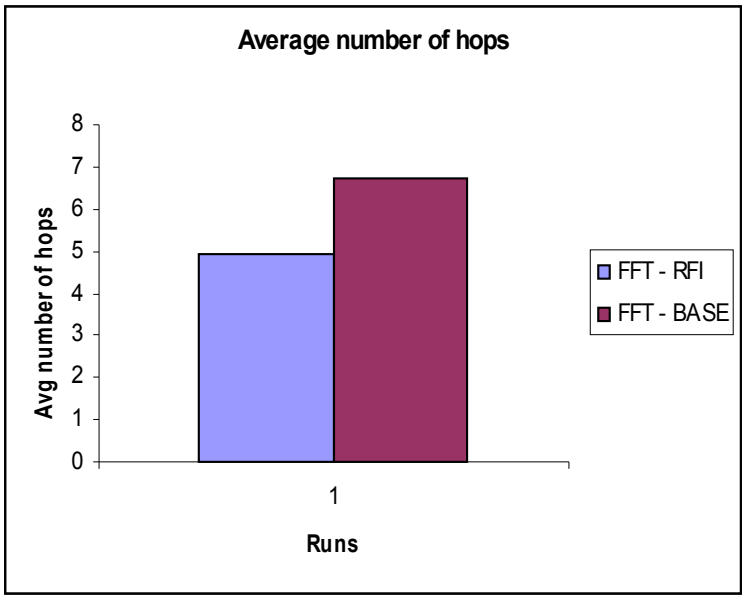
performance of RF-I are FFT and Radix from SPLASH suite [15].

Parameter	Value
number of cores	64
Number of L2 cache banks, size	16, 128KB – 8 way
L1 size	8KB – 4 way
Mesh dimensions	10 * 10
Memory Ports	4
Latency to memory	320 cycles

Table 1- System Configuration

The baseline we used for comparing RF-I against is a conventional mesh interconnect, which itself is an aggressive baseline. A mesh provides sufficient bandwidth for on-chip communications, and seems to be a scalable interconnect for chip multiprocessors with large number of cores on the chip. Figure 4 compare the number of hops in a 10*10 mesh topology, with the use of RF-I and without the use of RF-I. Although, we do see a decrease in the average number of hops it is not significant, for example for FFT the average number of hops drops by approximately two hops when using RF-I as compared to conventional mesh. We observed that the shortcuts provided in RF-I are overly utilized, and so it results in congestion at the routers. Due to the congestion at the routers, the benefit seen from using RF-I is not significant. We introduced some schemes to reduce the congestion at the routers, and did not see a huge improvement in performance. The inherent problem is the latency incurred for a communication from the source to the router, which serves as on-ramp for the RF-I shortcut. If the shortcut is congested, the communication will be re-routed to use surface trees instead of using the RF-I shortcut. One approach, that could be used to address this problem is to determine whether the RF-I shortcut should be used or not at the source. We tried to adapt our network model to this scenario, and observed that this scheme caused re-ordering of packets from the same source. Although, our coherence protocol supports out-of –order packets from different sources, it does not currently support out-of-order packets from a single source.

Our observations reinforce the fact that shortcuts are utilized for frequently demanded communications, and are beneficial to the performance of an application.



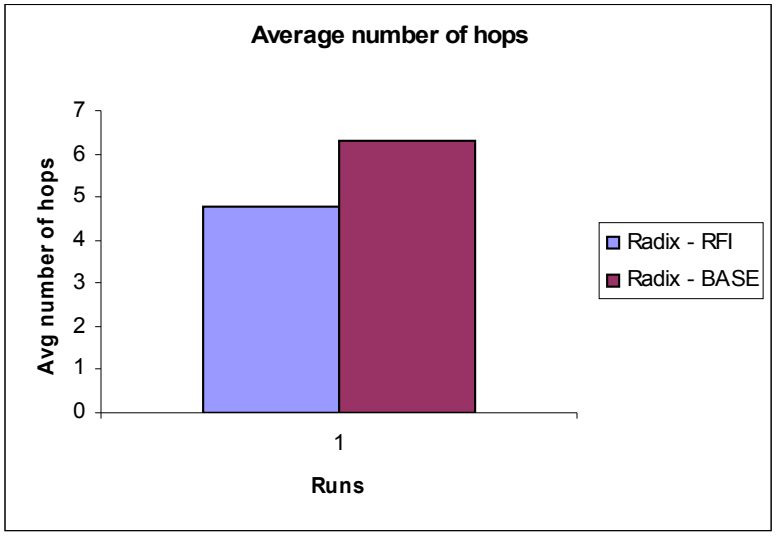
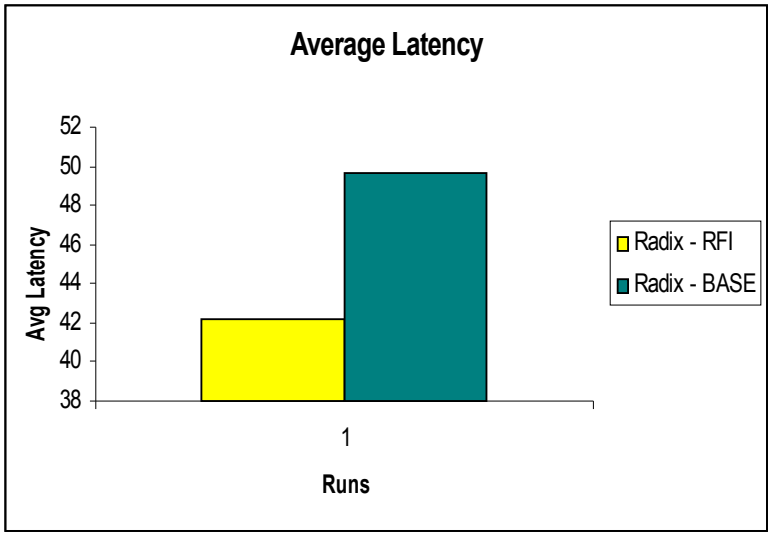


Figure 4 – Comparison of Average number of hops.



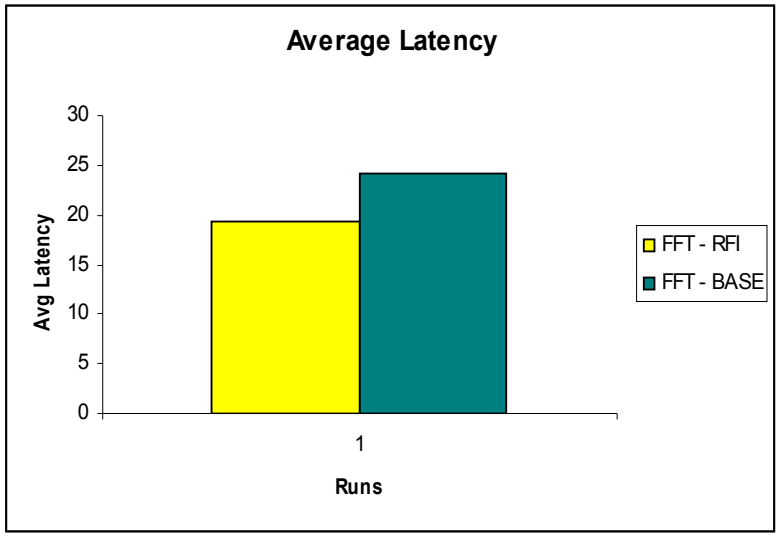


Figure 5 – Comparison of Average Latency.

6 Conclusion

MCSim infrastructure allows a fast and flexible exploration of a CMP – like system with a large number of cores. It supports multi-tasking, and multi-threading. Different types of topologies and types of interconnect can be modeled using the MCSim NoC model. The NoC model provides the flexibility to study the diverse range of communication demands seen within and across applications. The SystemC model can be leveraged to explore the design points.

For future work, we plan on making the coherence protocol more robust to the re-ordering of

packets sent from a single source. We also plan to consider other types of interconnects that could be used in future CMP's to evaluate the benefit of using RF-I.

Our initial evaluation of RF-I, indicates that RF-I could be used to provide a dynamic way of adapting to the diverse communication demands of applications.

References

[1] P. Abad, V. Puente, J.A. Greogorio, and P. Prieto. Rotary Router: An Efficient Architecture for CMP Interconnection Networks. In *Proceedings of ISCA-34*, June 2007.

[2] B. Beckmann and D. Wood. TLC: Transmission Line Caches. In *Proceedings of MICRO-36*, December 2003.

[3] Nathan. L. Binkert, Erik. G. Hallnor, and Steven. K. Reinhardt. Network-Oriented Full-System Simulation using M5. In *Proceedings of the Sixth Workshop on Computer Architecture Evaluation Using Commercial Workloads*, February 2003.

[4] L. Cheng, N. Muralimanohar, K. Ramani, R. Balasubramonian, J.B. Carter. Interconnect-Aware Coherence Protocols for Chip Multiprocessors. In *Proceedings of ISCA-33*, June 2006.

[5] Y. Jin, E.J. Kim, and K.H. Yum. A Domain-Specific On-Chip Network Design for Large Scale Cache Systems. In *Proceedings of HPCA-13*, February 2007.

[6] J. Kim, W.J. Dally, and D. Abts. Flattened Butterfly : A Cost-Efficient Topology for High-Radix Networks. In *Proceedings of ISCA-34*, June 2007.

[7] J.Kim, C. Nicopoulos, D. Park, R. Das, Y. Xie, N. Vijaykrishnan, M.S. Yousif, C.R. Das. A Novel Dimensionally-Decomposed Router for On-Chip Communication in 3D Architectures. In *Proceedings of ISCA-34*, June 2007.

[8] N. Kirman, M. Kirman, R.K. Dokania, J.F. Martinez, A.B. Apsel, M.A. Watkins, and D.H. Albonesi. Leveraging Optical Technology in Future Bus-based Chip Multiprocessors. In *Proceedings of MICRO-39*, December 2006.

[9] A. Kumar, L.S. Peh, P. Kundu, and N.K. Jha. Express Virtual Channels: Towards the Ideal Interconnection Fabric. In *Proceedings of ISCA-34*, June 2007.

[10] R. Kumar, V. Zyuban, D.M. Tullsen, Interconnections in Multi-core Architectures: Understanding Mechanisms, Overheads and Scaling. In *Proceedings of ISCA-32*, June 2005.

[11] Milo M. K. Martin, Daniel J. Sorin, Bradford M. Beckmann, Michael R. Marty, Min Xu, Alaa R Alameldeen, Kevin E. Moore, Mark D. Hill, and David A. Wood. Multifacet's General Execution-driven Multiprocessor Simulator (GEMS) Toolset. *Computer Architecture News (CAN),*

September 2005.

[12] C.A. Nicopoulos, D. Park, J. Kim, R. Das, Y. Xie, N. Vijaykrishnan, M.S. Yousif, C. R. Das.ViChaR: A Dynamic Virtual Channel Regulator for Network-on-Chip Routers. In *Proceedings of MICRO-39*, December 2006.

[13] SESC: cycle accurate architecute simulator.

[14] SimpleScalar.

[15] SPLASH: Stanford Parallel Applications for Shared Memory.