Simulating Future Chip Multi Processors

Faculty Advisor: Dr. Glenn D. Reinman

Name: Mishali Naik

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.

I will be adding more here

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, M5, Simics/GEMS, and SESC. Each of these simulators have 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.

--- I will be adding prior work RF -I here

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.

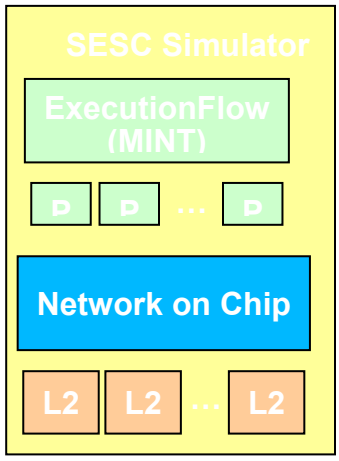

Figure 1 – High level layout of SESC.

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. 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.

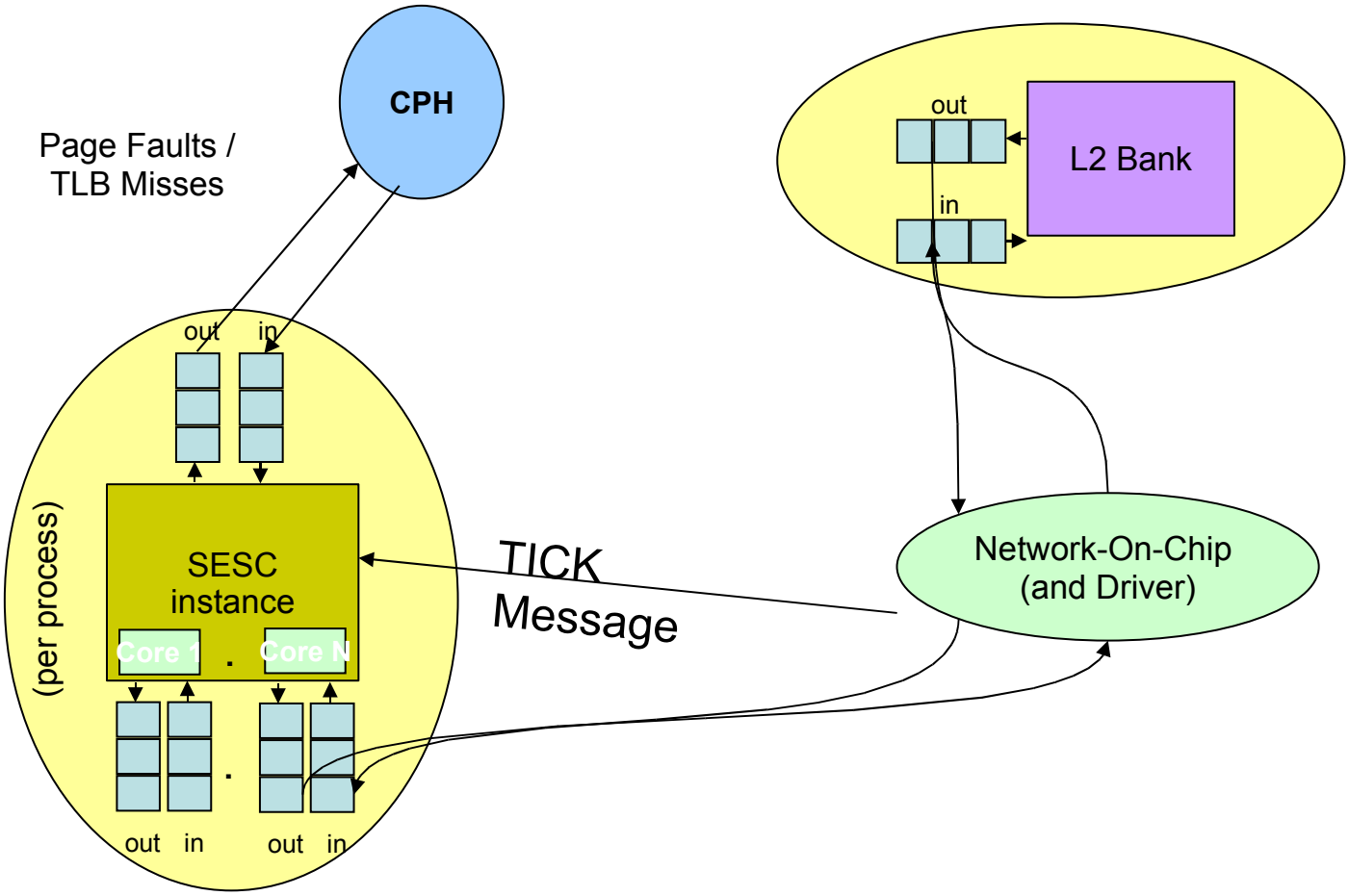

Figure 2. High – level design of MCsim infrastructure.

4 RF - Interconnect

5 Results

6 Conclusion

I will be filling in the sections 4 – 6 later.

References

[1] 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.

[2] 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-dromiven Multiprocessor Simulator (GEMS) Toolset. *Computer Architecture News (CAN),* September 2005.

[3] SESC: cycle accurate architecute simulator.