

# New Challenges in Benchmarking Future Processors

Shubhendu S. Mukherjee  
Intel Corporation  
Shubu.Mukherjee@intel.com

## ABSTRACT

*The advent of system on a chip, fault-tolerant features, and multiple power modes in the mainline processor market has significant impact on how we would benchmark future processors. Unfortunately, only a subset of these modes may provide the highest performance for these chips. If vendors report performance numbers only for these highest performing modes, then customers are faced with the challenge of selecting the specific processor and the specific configuration for his or her operating environment without the benefit of any comparable benchmark numbers.*

*In this paper I examine three different categories of hardware modes that a processor chip could be configured in. These are the use of a snoopy or directory protocol, the presence or absence of fault tolerance, and presence of different power modes. Based on these examples, I classify the modes into performance-centric and environment-centric modes and propose that vendors report performance numbers for these different modes. This would allow customers to compare processor chips from different vendors under different operating conditions.*

## 1. INTRODUCTION

Four technology trends—exponential proliferation of on-chip transistors, the constant RC delay of long on-chip wires, transient faults due to cosmic ray strikes, and power constraints—may necessitate changes to the way we would benchmark future processors. The exponential proliferation in the number of on-chip transistors—fueled by Moore’s Law—is forcing designers to find innovative ways to use these transistors to remain competitive in the high-performance processor market.

Unfortunately, the speed of long on-chip wires is remaining constant due to the constant RC delay, unlike transistors that speed up by roughly 30% with every new technology generation. This trend, coupled with the complexity of dealing with such an enormous

number of transistors, is forcing designers to refrain from building a big, complicated, and monolithic uniprocessor on a single chip. Rather, the trend is towards Chip Multiprocessors (CMPs) with many processors on the same chip (e.g., IBM Power4 [1], HP Mako [2]), and/or processors with large caches that can use up the available number of on-chip transistors.

Thus, greater amounts of system functionality, such as multiple processors and large caches, are getting integrated on the same chip. Additional system components, such as memory controllers, multiprocessor network routers, and cache-coherent directory controllers, have also started migrating to the same die as the processor chip (e.g., Alpha 21364 [3]). Clearly, we are seeing the emergence of *system on a chip* in the mainline processor market.

Unfortunately, while most such on-chip functions improve performance for specific application domains, two other trends have begun limiting the performance of these processors. First, soft errors due to cosmic ray strikes promise to increase dramatically in the next few generations. This is arising because of the proliferation of number of on-chip transistors and the reduction in voltage levels of the chips. Consequently, designers will be forced to use some of the on-chip transistors for ECC, parity, or other forms of error detection and/or correction. Other mechanisms, such as lockstepping two complete processors (e.g., as in Compaq Himalaya [4] or IBM G5 [5]) or the use of redundant multithreading (e.g., AR-SMT [7], DIVA [6], SRT [8], or CRT [9]) to detect faults, may also become popular with CMP designs. Additionally, support for transparent fault recovery in hardware, such as tri-modular redundancy [10], may necessitate even more sophisticated hardware support, which may further limit a chip’s performance.

Second, processors are facing severe limits on average and peak power dissipation [11]. A greater number of on-chip transistors demands greater power consumption (and consequent heat dissipation), if we follow the same design methods as in the past. So, designers must carefully use the on-chip resources to

reduce wasted work. Additionally, designers may have to slow down processors, either statically or dynamically, if they consume too much power or completely shut some of them off to balance the total power dissipation in a CMP chip.

The above trends have significant implications on the way we would benchmark future processors and systems. Currently, two of the most popular benchmark suites are the SPEC CPU suite and the TPC suite. The SPEC CPU 2000 suite (<http://www.spec.org>) measures a processor's integer, floating point, and memory system performance using a set of 25 benchmarks. SPEC allows us to characterize uniprocessor performance using two speed metrics—*SPEC base* and *SPEC peak* performance. Base performance measures the performance of a processor with a set of fixed and uniform compiler flags applied to all benchmarks in the suite. In contrast, peak performance measures the performance of a processor with benchmark-specific compilation flags. Similarly, SPEC allows us to characterize multiprocessor or multithreaded performance using *SPEC rate*, which measures the rate at which one or more processors may complete multiple copies of the same benchmark. SPEC rate can be similarly categorized into base and peak rates.

TPC (<http://www.tpc.org>) consists of a suite of benchmarks to measure the performance of a system running transaction processing and database workloads. Performance measurements can be reported as absolute performance or in terms of price/performance. For example, performance measurements for the TPC-C benchmarks are reported as both *tpm* (transactions per minute) and *price/tpm*.

Unfortunately, neither the SPEC nor the TPC suite was designed to measure a system-on-a-chip's performance. The key problem is that these systems on a chip could potentially have optional hardware modes that will be configured in different ways. For example, to deliver high performance on TPC-C like benchmarks, a vendor may statically configure the system on a chip to use a snoopy cache coherence protocol that provides fast cache-to-cache transfers between processors. In contrast, the same vendor may configure the chip to use a directory-based coherence protocol if they incorporate these chips in a large, scalable, multiprocessor system (Section 2). Similarly, the inclusion of fault detection and/or recovery can dramatically reduce the performance of a system on a chip when the fault detection mode is turned on (Section 3). Finally, systems that will be configured for lower power dissipation may switch to a specific

static mode—for example, by turning off several processors in a CMP—or reduce the frequency of the entire chip itself (Section 4).

Neither the SPEC nor the TPC suite requires vendors to benchmark their processors or systems in all such possible configurations. Thus, a vendor may report its performance numbers in its highest performing modes—with benchmark suite-specific system configuration, no fault detection, and highest power dissipation.

Unfortunately, such reporting can both be confusing and unrealistic. It can be confusing because, for example, a customer may have no clue how to compare systems-on-a-chip for processors with fault detection and recovery enabled. The customer may desire to buy a highly reliable computing system, but the only numbers he/she may find for comparison are numbers for these systems in the non-fault detection mode.

It can be unrealistic because, for example, vendors can potentially produce results in an environment (e.g., in a basement or in a building with thick concrete) that could have reduced effect of cosmic rays. Such measurements are of little value to a customer who wants to use systems in operating environments (e.g., in an office with large glass windows or in an airplane) where the effect of cosmic rays may be quite high.

Based on the above discussions, I examine potential solutions in Section 5. We can divide these solutions into two broad categories based on the nature of the hardware modes. These two modes are performance-centric modes and environment-centric modes. Performance-centric modes are those that configure the system in specific hardware modes for specific benchmarks and benchmark suites. One possible solution for such modes is to use the SPEC characterization of base and peak. The base mode could be one mode specified by the system on a chip for all configurations. However, the peak mode can be expanded to include benchmark-specific hardware modes, beyond the benchmark-specific compiler flags that are already in use.

Environment-centric modes are those that specify and evaluate benchmarks based on the specific environment the system-on-a-chip is embedded in during measurements. Both fault detection and power modes fall under this category. At the minimum, systems must specify in what environment the measurements were done. It would be even better if systems could specify the cosmic ray flux and the maximum power dissipation possible in that environment (although gathering such data can be quite

expensive [12]) as well as multiple numbers for different points in the environment spectrum, such as high and low cosmic ray flux and high and low power dissipation modes.

## 2. SNOOPY VS. DIRECTORY PROTOCOLS

Most shared-memory multiprocessors use a coherence protocol to keep per-processor caches coherent. Typically, a cache block in a processor's cache in a cache-coherent multiprocessor system has at least three basic states—invalid, shared, and exclusive. Usually, a cache block becomes shared when a processor retrieves the cache block in read-only state (possibly resulting from a cache miss from a load instruction). Similarly, a cache block becomes exclusive when a processor retrieves a cache block in writeable state (possibly resulting from a cache miss from a store instruction).

The two most common coherence protocols are snoopy and directory protocols, which differ in how these states are manipulated. A snoopy protocol usually relies on a broadcast mechanism, such as a shared bus, to facilitate the state transitions. To request a shared block a processor sends snoop requests to all processors and memories that can have that block. Either a processor's cache or a memory returns the block to the processor in the shared state. A similar sequence of events occurs when a processor requests an exclusive block. Unfortunately, snoopy protocols do not scale well to large systems. This is either because the shared broadcast medium may not scale or because expensive broadcast messages must be sent on most processor cache misses.

Directory protocols allow cache-coherent shared-memory multiprocessors to scale to a large number of processors by avoiding a broadcast medium and broadcast messages. Instead, to request a shared block, a processor sends its request to a pre-selected "home" node. Typically, the home node would return the data. However, different variations in this strategy are possible. To request an exclusive block, a processor sends its request to the home node. The home node tracks down one or more sharers of the block, invalidates the cache blocks in all these sharers, and then sends the response back to the requester. Alternatively, the home node can also request one of the sharers to forward the block to the original requestor.

Although directory protocols allow systems to scale well, they are not optimized for producer-consumer or migratory sharing patterns in which a processor writes data that may be needed by another processor in the near future. For snoopy protocols, a processor can

obtain the data in two hops: one for the request message and the second one for the response message. However, for directory protocols, this can take three hops: the first one for request to the home node, the second for the directory to forward the request to the processor with exclusive access to the block, and finally the third one for the response to propagate back to the requestor.

Thus, a directory protocol penalizes sharing patterns that require cache-to-cache transfers, particularly for small-scale systems that can benefit from a snoopy protocol. This can have a significant impact on the performance of benchmarks, such as TPC-C, that are dominated by such cache-to-cache transfers. For example, researchers have estimated that such cache-to-cache transfers could account for more than 50% of L2 cache misses in TPC-C like benchmarks [13][14]. Worse, the impact of such cache-to-cache transfers rises as cache sizes increase. This is because a bigger cache reduces other kinds of misses, such as conflict and capacity misses, but does not significantly reduce sharing misses that cause cache-to-cache transfers. Thus, small-scale commercial systems, such as the IBM Northstar [16], rely on snoopy protocols to provide aggressive performance on TPC-C like benchmarks. Recently, Martin, et al. [15] have shown that it is possible to build a protocol that adapts between a snoopy and directory protocol. Such adaptive protocols may work for large systems. It is not clear whether such adaptive protocols are a winner in price/performance for small-scale systems.

This dichotomy between snoopy and directory protocols in the performance and scalability spectra makes it challenging for vendors to build processor chips with full or partial support for the protocol on the chip. Typically, vendors would like to optimize systems that sell the most. And, systems that sell the most are small-scale systems that can be built with snoopy protocols. However, vendors also like to support large systems because they typically provide higher profit margins and assure a customer of a scalable system. Consequently, vendors are likely to end up supporting both a snoopy and a directory protocol on the same chip. In the past, such a dichotomy was not a problem because the protocol controllers would be off-chip; vendors would manufacture different chip sets for systems of different sizes.

With the advent of this new generation of processors, a customer must potentially deal with different hardware modes for small-scale and large-scale systems and ensure that the system he or she is buying reflects the

benchmark numbers for the appropriate mode he or she is interested in. Unfortunately, today vendors do not have to publish numbers for these different modes for the same processor.

### 3. FAULT DETECTION AND RECOVERY

Support for fault tolerance—that is, fault detection and recovery mechanisms—in the mainline processor market may also force vendors to create different hardware modes. This is because today’s microprocessors are vulnerable to transient hardware faults caused by alpha particle and cosmic ray strikes.

Strikes by cosmic ray particles, such as neutrons, are particularly critical because of the absence of any practical way to protect microprocessor chips from such strikes. As individual transistors shrink in size with succeeding technology generations, they become less vulnerable to cosmic ray strikes. However, decreasing voltage levels and exponentially increasing transistor counts cause overall susceptibility of a chip to increase rapidly. Further, the impact of cosmic ray strikes grow by orders of magnitude in an airplane or on higher elevations.

To compound the problem, achieving a particular failure rate for a large multiprocessor server requires an even lower failure rate for the individual microprocessors that comprise it. Due to these trends, fault detection and recovery techniques, currently used only for mission-critical systems, would very likely become common in all but the least expensive microprocessor devices.

Unfortunately, current known techniques for fault detection and fault recovery degrade a chip’s performance quite dramatically. Traditionally, processors have used error detection and/or correction codes (e.g., parity, ECC, CRC) to meet failure specifications. However, such error detection and correction codes do not cover logic blocks effectively. Additionally, error codes may degrade a pipeline’s performance by adding extra pipeline stages.

Consequently, vendors may have to resort to other techniques for fault detection and recovery. There are three other known techniques for fault detection: lockstepping, instruction recycling, and redundant multithreading. In lockstepping, two complete and identical processors run in lockstep—performing the same computation in every cycle. Inputs to both processors are replicated in each cycle. Similarly, in each cycle outputs from both processors are compared for mismatch. On a mismatch, the hardware typically initiates a hardware or software recovery sequence.

The advent of Chip Multiprocessors with two or more processors on the same chip would enable such lockstep checking on the chip itself. Because lockstepping uses two completely separate processor cores, it can catch both permanent faults (e.g., from electron migration) and transient faults (e.g, from cosmic ray strikes).

Instruction recycling is another technique for fault detection. Instead of comparing instructions from different processors, instruction recycling runs the same instruction twice through the same processor and checks the corresponding outputs from these instructions for mismatch. Since instruction recycling runs copies of the same instruction through the same processor, it could potentially catch all single transient faults, but not all permanent faults.

Finally, there is a new class of fault detection techniques that can be classified under Redundant Multithreading (RMT). RMT runs two identical copies of the same program as independent threads either on a multithreaded processor or on different processors and compares their outputs for mismatch. Several researchers (e.g., [7],[6],[8],[9]) have proposed variations of this technique.

Thus, all current fault detection techniques use some form of redundancy that could have been used otherwise to boost the performance of the chip itself. If duplicate processor cores are used for fault detection, then we sacrifice an entire processor’s performance. If multithreaded processors are used, then we sacrifice about 20 – 40% of the processor’s performance (e.g., [7],[8],[9]).

Fault recovery techniques can further reduce the performance of these processor chips. Fortunately, not every processor or environment would require fault recovery. Instead, on a detected fault, the processor could be halted and, then restarted. In such cases, fault recovery pays very little penalty in performance. (assuming fault recovery is not triggered too often). However, more aggressive and highly available systems may want to support aggressive fault recovery techniques, such as tri-modular redundancy (TMR) or pair-and-spare techniques. In TMR, outputs of three processors are compared for mismatch using a voting scheme. When a fault occurs in one of the processors, the other two processors’ outputs would match and the program can continue executing. Of course, the processor that had a fault has to be restarted with the correct state, which may require synchronization with the other processors. In contrast, pair-and-spare uses two pairs of processors. Each pair

of processor has its own fault detection mechanism. When a fault is detected in one of the pairs, the other pair takes over as the main execution engine for the program. Thus, both TMR and pair-and-spare have very little recovery time. Other techniques, such as software-based recovery mechanisms, can incur significantly higher overhead in performance.

Clearly, the advent of CMPs with several processors on-chip can allow both fault detection and recovery techniques to be implemented on-chip. Fortunately, not all applications and environments will require aggressive fault detection and recovery mechanisms. Unfortunately, however, this could lead vendors to create several modes: with or without fault detection and with or without fault recovery. Once again, customers must be aware of what they are buying compared to which environment the processor has been benchmarked in.

#### 4. POWER CONSTRAINT

Like cosmic ray strikes, increased power dissipation from microprocessor chips has also begun to plague the microprocessor industry. As the number of on-chip transistors and clock frequency increase, dynamic power dissipation from transistor switching begins to reach exorbitant levels. Borkar [11] predicts that dynamic power itself will increase from about 100W in 1999 to 2,000W in 2010. Worse, as supply voltage scales down, static power dissipation from leakage current starts becoming a major source of power dissipation as well.

Power dissipation has two implications—that of peak power dissipation and average power dissipation. The higher the peak power dissipation, the higher is the cost for packaging the chip. Additionally, higher peak power requirement may also dictate how many processor chips can be bundled in a certain cubic feet. The chips not only have to be cooled efficiently, but there must also be adequate power supplies to run these machines. Thus, densely-packed rack-mounted systems with several processors must closely monitor and reduce the power supply to and dissipation from these chips. Average power dissipation, on the other hand, has greater impact on the power supply, particularly if these machines are running on batteries.

Thus, to configure processor chips in certain packaging material and certain environments (e.g., densely-packed “blade” systems), vendors may create multiple modes with different power supply and dissipation characteristics for each mode. Designers have several tricks to reduce the peak power dissipation. For example, certain portions of the instruction queue,

and register ports could potentially be shut down in lower power modes. Also, vendors could reduce the frequency or reduce wasted work by choking the amount of speculation in processors with aggressive pipelines. In a CMP system, entire processors may be shut down to facilitate inclusion in lower power systems. Both architecture and circuit conferences abound in techniques to reduce the power supply to and dissipation from such chips.

Usually, however, the processor chips provide the highest performance in the highest power dissipation modes. Consequently, if vendors report the highest performance numbers from these chips, a customer may not be able to compare numbers for the same chips in lower power modes.

#### 5. CONCLUSIONS

The above discussions suggest the advent of processor with several different hardware modes. Processor chips could support several orthogonal modes:

- Processors with snoopy protocol and processors with directory protocols,
- Processors with no fault detection, processors with fault detection but no transparent recovery, and processors with both fault detection and transparent hardware recovery, and
- Processors with high power mode and processors with low power mode.

The above combination itself would create 12 different hardware configurations. Unfortunately, only two of these configurations are the highest performing modes depending on the specific benchmark.

The above hardware modes can be classified into two categories: performance-centric and environment-centric modes. Performance-centric modes are those that provide best performance for specific sets of benchmarks. Thus, the choice between a snoopy and directory protocol falls under the performance-centric mode. The SPEC suite of benchmarks already allows vendors to provide two different software modes: *base* and *peak*. The base mode must use uniform compiler flags for all benchmarks, whereas the peak mode can use benchmark-specific flags. One solution to address the performance-centric hardware mode would be to allow vendors to include hardware flags in the peak mode. Then, customers can get some sense of the performance of the benchmarks they are interested in.

The environment-centric mode includes support for fault tolerance and support for different power modes. Customers would like to know the performance of

these processor chips in the fault tolerant modes and/or in low power modes. Then, customers must assess the environment they would operate these machines in and decide which processor chips to buy.

Currently, processor vendors are not required to report such numbers. Worse, a processor that has the highest performance in the absence of fault tolerance and in the high power mode may not be the highest performance processor when these features are turned on. Both vendors and customers must come together to decide what would be an appropriate benchmarking standard for this new generation of “system-on-a-chip” processors.

## ACKNOWLEDGMENTS

I would like to thank Joel Emer and Geoff Lowney for their helpful comments on this paper.

## REFERENCES

- [1] IBM, “Power4 System Microarchitecture,” <http://www-1.ibm.com/servers/eserver/pseries/hardware/whitepapers/power4.html>.
- [2] David J.C.Johnson, “HP’s Mako Processor,” Fort Collins Microprocessor Lab, October 16, 2001. Available from [http://cpus.hp.com/technical\\_references/mpf\\_2001.pdf](http://cpus.hp.com/technical_references/mpf_2001.pdf).
- [3] Shubhendu S. Mukherjee, Peter Bannon, Steven Lang, Aaron Spink, and David Webb, “The 21364 Network Architecture,” Hot Interconnects IX, 2001.
- [4] Alan Wood, “Data Integrity Concepts, Features, and Technology,” White paper, Tandem Division, Compaq Computer Corporation.
- [5] T.J.Slegel, et al., “IBM’s S/390 G5 Microprocessor Design,” IEEE Micro, pp 12–23, March/April, 1999.
- [6] Todd M. Austin, “DIVA: A Reliable Substrate for Deep Submicron Microarchitecture Design,” Proceedings of the 32<sup>nd</sup> Annual International Symposium on Microarchitecture, November 1999.
- [7] Eric Rotenberg, “AR-SMT: A Microarchitectural Approach to Fault Tolerance in Microprocessor,” Proceedings of Fault-Tolerant Computing Systems (FTCS), 1999.
- [8] Steven K. Reinhardt and Shubhendu S. Mukherjee, “Transient Fault Detection via Simultaneous Multithreading,” International Symposium on Computer Architecture (ISCA), June-July, 2000.
- [9] Shubhendu S. Mukherjee, Michael Kontz, and Steven K. Reinhardt, “Detailed Design and Evaluation of Redundant Multithreading Alternatives,” submitted for publication.
- [10] Daniel P. Siewiorek and Robert S. Swarz, “Reliable Computer Systems: Design and Evaluation,” A.K. Peters Ltd, October 1998.
- [11] Shekhar Borkar, “Design Challenges of Technology Scaling,” IEEE Micro, pp 23 – 29, July/August, 1999.
- [12] Norbert Seifert, Compaq Computer Corporation, Personal Communication.
- [13] Luiz Andre Barroso, Kourosh Gharachorloo, and Edouard Bugnion, “Memory System Characterization of Commercial Workloads,” Proceedings of the 25<sup>th</sup> Annual International Symposium on Computer Architecture (ISCA), pages 3 – 14, Barcelona, Spain, June/July 1998.
- [14] Parthasarathy Ranganathan, Kourosh Gharachorloo, Sarita V. Adve, and Luiz Andre Barroso, “Performance of Database Workloads on Shared-Memory Systems with Out-of-Order Processors,” Proceedings of the Eighth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), pages 307 – 318, San Jose, October 1998.
- [15] Milo K. Martin, Daniel J. Sorin, Mark D. Hill, and David A. Wood, “Bandwidth Adaptive Snooping,” Proceedings of the 8<sup>th</sup> Annual International Symposium on High-Performance Computer Architecture (HPCA), Feb 2002.
- [16] J. Borkenhagen and S. Storino, “4<sup>th</sup> Generation 64-bit PowerPC-Compatible Commercial Processor Design,” IBM Server Group Whitepaper, Jan 1999.