Quantifying and Comparing the Impact of Wrong-Path Memory References in Multiple-CMP Systems

Ayse Yilmazer[1], Resit Sendag[1], Joshua J. Yi[2]

[1] - Department of Electrical and Computer Engineering
University of Rhode Island
Kingston, Rhode Island
{yilmazer, sendag}@ele.uri.edu

[2] - Networking and Computing Systems Group
Freescale Semiconductor, Inc.
Austin, The Great State of Texas
joshua.yi@freescale.com

Abstract

Out-of-order execution processors with aggressive branch prediction are the core of current-generation high-performance multiprocessor systems. Despite their relatively high branch prediction accuracies, these processors still execute many memory instructions on the mispredicted path. These wrong-path memory references pollute the caches and increase the amount of memory traffic, but may also prefetch useful data into the caches. For multiprocessor systems, wrong-path memory references also affect the number of writebacks, cache-to-cache transfers, and cache line state transitions. In this paper, we explore the effects that wrong-path memory references have in multiple chip-multiprocessor systems. Our results show that wrong-path memory references increase the amount of L1 and L2 cache traffic by 16% and 35%, respectively, and the traffic on the internal and external networks by 36% and 30%, respectively. These additional L1 and L2 cache replacements increase the number of additional writebacks by 70%, L1 copy invalidations up to 68%, and extra cache line state transitions in the L1 and L2 caches by 4% and 16%, respectively.

1 Introduction

As shrinking transistors widths enable computer architects to build chips with multiple processor cores (chip-multiprocessors (CMP)), multiple CMP systems (multi-CMPs) have become increasingly popular. The fundamental building block of these CMP systems is a processor core that aggressively predicts the direction and target of branch instructions. But when the branch predictor mispredicts a branch, the processor executes many memory references down the mispredicted path. Although these wrong-path memory references do not change the processor's architectural state, they can change the data in the cache and its cache coherence state, both of which may degrade the processor's performance.

Previous work [5, 8, 9] showed that wrong-path memory references affect the memory subsystem (MSS) behavior of uniprocessor systems in two key ways. First, wrong-path memory references may function as prefetches by bringing useful data into the cache. Second, wrong-path memory references can pollute the caches and increase the amount of memory traffic. In [12], we quantified the effects that wrong-path memory references have on cache-coherent shared-memory multiprocessor systems (SMPs). Our results showed that for both broadcast-based and directory-based SMPs, not only do the wrong-path memory references affect the performance of each individual processor in the system, they also affect the performance of the entire system by increasing the number of writebacks and invalidations, cache line state transitions, and the amount of resource contention.

In this paper, we analyze the effects that wrong-path memory references have on multi-CMPs systems. Since multi-CMP systems maintain cache coherence at two different levels, intra-CMP and inter-CMP, wrong-path memory references have more opportunities to affect the memory system behavior.

2 Effects Due to Wrong-Path Memory References

As described in the introduction, wrong-path memory references can affect MSS performance of an SMP or multi-CMP system by increasing the number of cache line state transitions, cache coherence transactions (which increases the number of writebacks and invalidations), and the amount of resource contention. In addition to these effects, the hierarchical directory protocol in a multi-CMP system adds an additional twist. When the intra-CMP and inter-CMP protocols are combined into a hierarchy, races could occur among messages within each CMP and between CMPs, which could lead to many transient states in L1 caches, L2 caches, and directory controllers.

In this paper, each CMP node contains four processors, private L1 instruction and data caches, a shared L2 cache, an on-chip interconnect, a global interconnect interface, and an interface to an off-chip memory controller, which in turn connects to DRAMs. Therefore, the intra-CMP cache coherence protocol affects the L1 caches of the CMP, while the inter-CMP cache coherence protocol affects the L1 and L2 caches of each CMP.

The L2 cache controller of our base system uses a

MOSI protocol to maintain coherence in the L2 cache. Since the L1 and L2 caches are inclusive, the L2 cache also tracks the coherence of the cache blocks in the L1 cache. The meaning of the cache coherence states is as follows:

- **M** The L2 cache owns the write privilege for this block.
- **S** The L2 cache shares this block with another L2 cache, but not with any of the internal L1s.
- **O** The L2 cache owns this block, and it is not present in any of the local L1 caches.
- **MT** The L2 cache owns this block, and it is dirty in one of the local L1 caches.
- **SO** The L2 cache owns this block, which is dirty, and it is present in one or more of the local L1s.
- **SS** The cache block is shared and is present in one or more of the local L1 caches.

When the cache block from a wrong-path memory reference replaces an M or O block, the L2 issues a writeback. But when a MT or SO block is replaced, the L2 cache has to first invalidate the L1 copies and then writeback the block. When a SS block is replaced, L2 cache needs only to invalidate the local L1 copies. For wrong-path memory references, these invalidations and writebacks increase the amount of internal and external traffic, which is especially significant for the replacement of a MT or SO block.

When a wrong-path memory reference causes an SO line replacement, the following steps occur in the MSS: 1) The L2 cache sends invalidation messages to the local L1 caches that share this block, 2) The L2 cache receives the invalidation acknowledgements from the local L1s, 3) The L2 cache sends a writeback to the directory, and 4) The L2 cache receives acknowledgement of that writeback, and 5) The block becomes invalid. From this example, we can easily see that a single wrong-path memory reference results in extra invalidations on the internal network (Step 1), extra acknowledgments from L1 caches sharing the line (Step 2), an extra writeback to the directory (Step 3), and an extra acknowledgment from the directory on the external network (Step 4). Furthermore, this wrong-path memory reference is also the cause for four extra state transitions in the L2 cache and one extra state transition in the L1 cache.

When a wrong-path memory reference from a remote L2 requests a cache block that is modified-dirty, the local L2 cache changes the state of the block from M→O and forwards the data to the remote L2 cache. When the request comes from one of local L1 caches, the local L2 cache issues a downgrade request to L1 cache that has modified the cache block. In response, that L1 cache downgrades the cache block from M→S and forwards a copy of block to L2 cache, which then changes the state of that block from modified-dirty to shared-owned and forwards the block to requesting cache. In both cases, the local L2 cache loses the write privileges for the line; in the latter case, the writing L1 cache also loses its write privileges, which leads to additional writebacks.

3 Experimental Methodology

Table 1 lists the five benchmarks that we evaluated in this paper. The first four benchmarks are from the SPLASH-2 benchmark suite [10], while *em3d* [13] is an electromagnetic force simulation benchmark.

Table 2 shows the configuration for the multi-CMP system that we evaluated in this paper.

Table 1. Benchmarks and input data sets

Benchmark	Input Data Set
fft	64K points
radix	2M integers, radix 1024
ocean	128x128 ocean
water-spatial	512 molecules
em3d	400K nodes, degree 2, span 5, 15% remote

Table 2. CMP system parameters

Processor Configuration
2 GHz 15-stage pipeline, OoO execution
8-wide dispatch/retirement
256/128-entry ROB/scheduler
10 cycle branch misprediction penalty
gshare branch predictor with 4K PHT
64-entry RAS and RAS exception table
32-entry CAS and CAS exception table
CMP Configuration
4 CMPs
4 processors per CMP
L1 caches: 32 KB, 2-way, 2 cycle latency
L2 cache: 2M, 2-way, 20 cycle latency
128 byte cache block size
32-entry MSHRs
4 GByte per bank
240 cycle DRAM latency

We collect our simulation results using the GEMS [3] simulator. GEMS builds on Virtutech's Simics [11], which is a full system simulator that allows functional

emulation of unmodified applications and operating systems. GEMS adds cycle-accurate models of an out-of-order processor core, cache hierarchy, various coherence protocols, multi-banked memory, and various `interconnection networks.

To avoid measuring the time needed for thread-forking, we begin our measurements at the start of the parallel phase by using Simics' functional simulation to execute the benchmarks until the start of the parallel phase. Then, we use first iteration to warm-up the caches and branch predictors. After warm-up, we simulate the benchmarks for one iteration to gather our simulation results.

4 Quantifying the Wrong-Path Effects

In this section, we quantify the effects that executing wrong-path memory references have on the local cache, on the caches of the other processors in the CMP, and on the communication between them due to coherence transactions. Unless stated otherwise, we present the results for four CMPs with four processors (nodes) each. To measure the various wrong-path effects, we track the speculatively generated memory references, and mark them as being on the wrong-path when the branch misprediction is known.

4.1 Number of Cache Block Replacements and Writebacks

For multi-CMP systems that use a two-level directory protocol and have inclusive L1 and L2 caches, replacing modified and shared cache blocks in both the L1 cache and L2 cache results in extra state transitions. The replacement of modified blocks is especially important because the cache loses the ownership of the block and, more importantly, the ability to silently update its value, which can significantly increase the number of invalidations needed for write updates. Figure 1 shows the percentage increase in the number of L1 data cache and L2 cache replacements, and hence write-backs, categorized by the cache line states for 4-node and 8-node CMPs.

The top pair of figures in Figure 1 shows that the number of L1 cache shared and modified line replacements due to the wrong-path memory references increased by an average of 30%. The bottom pair of figures shows the percentage increase in the number of L2 caches replacements due to the wrong-path memory references by category. Note that replacing M (Modified), MT (Modified-Dirty), O (Owned), and SO (Shared-Owned) cache blocks requires a writeback. Furthermore, for MT, SO and SS line replacements, the L2 cache also must invalidate the shared L1 copies of the lines and L1 caches must send invalidation acknowledgements (or writeback data when the line is MT). These two figures show that, for *water-spatial*, due to wrong-path memory references, the number of MT block replacements on a 4-node CMP system increases by 6%, while the number of MT block replacements increases by 5% on an 8-node CMP system. For *em3d* on 4-node CMPs, the number of S and M block replacements increases by about 70% and 3%, respectively.

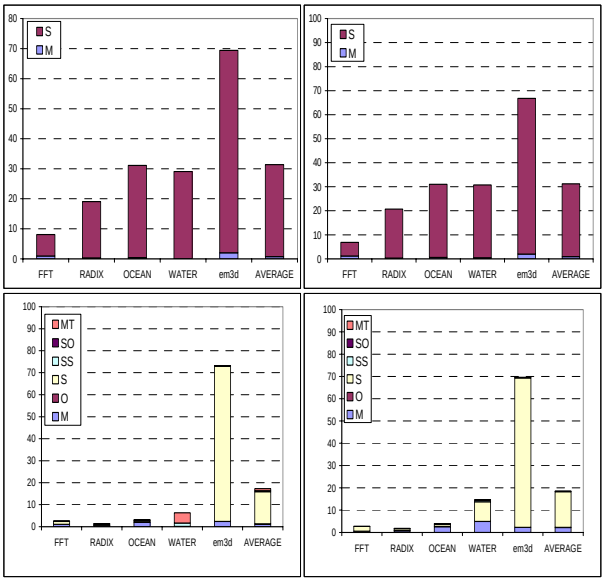

Figure 1. Percentage increase in the number of replacements in the L1 data cache (Top pair) and L2 cache (Bottom pair) for 4 (Left pair) and 8-node (Right pair) CMPs

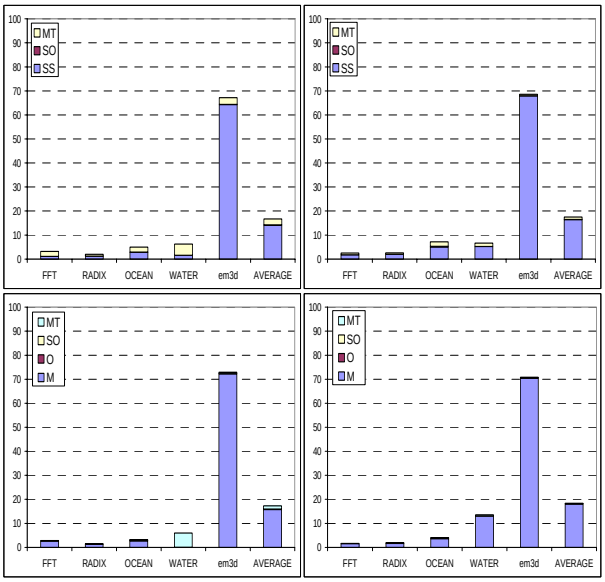

Figure 2. Percentage increase in the number of the L1 data cache copy invalidations (Top pair) and L2 cache writebacks (Bottom pair) for 4 (Left pair) and 8-node (Right pair) CMPs

Figure 2 shows the percentage increase in the

number of L1 copy invalidations and L2 writebacks, due to the wrong-path memory references, for 4 and 8-node CMPs. For *em3d*, wrong-path memory references increases the number of L1 copy invalidations by about 70%. This is due to the large amount of writebacks for modified L2 blocks. Additionally, some of the L1 copy invalidations touch modified blocks, which also increases the number of L1 writebacks. Modified block writebacks are the cause for most of the wrong-path L2 writebacks. On average, the number of L2 writebacks increases by about 20%.

4.2 Cache Line Replacements

To determine the potential performance impact that wrong-path memory references have in multi-CMP systems, in this paper, we categorize the misses caused by wrong-path memory references into four categories: *unused*, *used*, *direct miss*, and *indirect miss*. In the *unused* category, the wrong-path cache block is either evicted before being used or is never used by a correct-path. By contrast, cache blocks in the *used* category are eventually used by a correct-path memory reference. *Direct miss* cache blocks can severely degrade the system's performance because they replace a cache block that a later correct-path memory reference accesses, but it itself is evicted before being used. Finally, since *unused* misses change the LRU state of cache blocks in that set, which may eventually cause correct-path misses; we call these misses *indirect misses*. For example, suppose that cache blocks *A*, *B*, *C* and *D* map to the same set. Assuming that the 1) Cache is two-way set-associative initially containing blocks *A* and *B*, 2) Where *B* is the LRU block, 3) *C* is the wrong-path reference, and 4) Both *A* and *D* are on the correct-path. In this situation, the sequence of operations is as follows: Wrong-path block *C* replaces *B*, correct-path miss block *D* replaces *A*, correct-path miss block *A* replaces *C*. If wrong-path reference for block *C* did not occur, then the correct-path reference for block *A* would have been a cache hit because block *D* would replaced block *B* instead.

Figure 3 classifies the cache misses due to wrong-path memory references into the aforementioned four categories. The results show that 30% to 75% and 43% to 78% of the wrong-path replacements in the L1 data cache and 1% to 64% and 0% to 19% of the wrong-path replacements in the L2 cache are *used* in 8-node CMP and 4-node CMP systems, respectively. *Direct misses* account for 0% to 88% and 0% to 10% of all wrong-path-caused replacements in 4 and 8-node CMP systems, respectively. Finally, *indirect misses* account for less than 5% of all wrong-path misses for most of the benchmarks and systems tested.

It is important to note that *direct* and *indirect* misses are responsible for the pollution caused by the wrong-path memory references. On average, in 4-node CMP systems, less than 10% of the L2 replacements are *used*. Our results show that, the percentage of the wrong-path replacements in the L1 and L2 caches is significantly higher in multi-CMP systems as compared to their SMP counterparts.

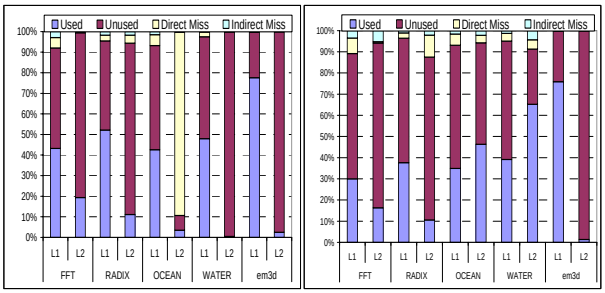

Figure 3. Replacements due to wrong-path memory references for 4 (Left pair) and 8-node CMPs (Right pair)

4.3 L1 Cache, L2 Cache, and Cache Coherence Traffic

In this section, we first present the increase in the number of L1 cache and L2 cache references, and then present the increase in the amount of cache coherence traffic due to wrong-path memory references.

L1 and L2 cache traffic: Figure 4 shows the percentage increase in the number of references to the L1 data and L2 caches due to wrong-path memory references, as a percentage of the total number of memory references to each cache. From Figure 4, we observe a higher percentage increase in the number of L2 cache accesses, although the total number of memory references, *i.e.*, L1 cache accesses, does not change as significantly. More specifically, for 4 and 8-node CMP systems, the average increase in the total number of memory references is 16% and 14%, respectively, while the average increase in number of L2 cache accesses is 35% and 36%, respectively. For *em3d*, wrong-path memory references significantly increase the number of L2 cache accesses, up to 70% for the 4-node CMP system.

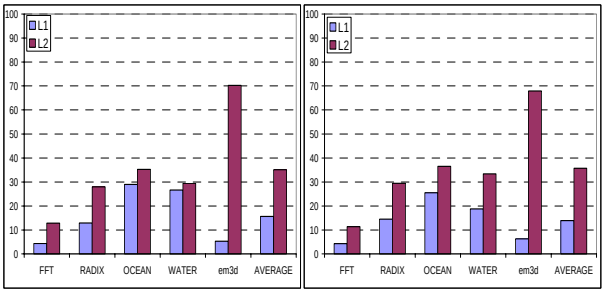

Figure 4. Percentage increases in L1 and L2 traffic due to wrong-path references for 4-node (Left pair) and 8-node CMPs (Right pair)

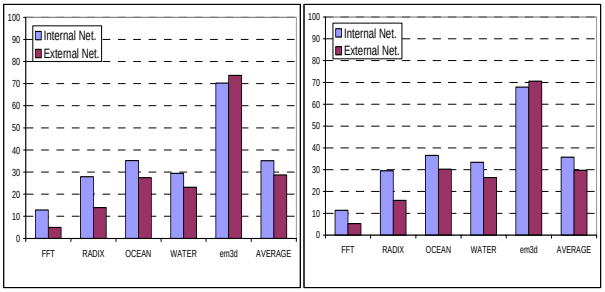

Figure 5. Percentage increase in coherence traffic due to wrong-path references for 4-node (Left pair) and 8-node CMPs (Right pair)

Cache coherence traffic: In multi-CMP systems, L1 cache misses increase the amount of traffic on the internal network (between processors) while L2 cache misses increase the amount of traffic on the external network (between the CMPs). In multi-CMP systems, L1 misses can be serviced by other internal L1 caches, the shared L2 cache, remote or local memory, remote L2 caches, or even remote L1 caches. By contrast, L2 cache misses can be serviced by remote or local memory, remote L2 caches, or remote L1 caches.

Figure 5 shows the increase in the amount of coherence traffic due to wrong-path memory references for 4 and 8-node multi-CMPs. For both multi-CMP system configurations, the amount of coherence traffic increases by an average of 36% and 30% for internal and external network, respectively. For *em3d*, wrong-path memory references increase the amount of coherence traffic by over 70% on both networks.

4.4 Extra Cache Line Transitions due to Wrong-Path Memory References

Wrong-path memory references can cause additional state transitions, *e.g.*, M→O and O→M. If a local L1 cache requests a copy of a modified-dirty block, the shared L2 cache issues a downgrade request to the writing processor's L1 cache. Then that L1 cache downgrades its cache block from M→S and sends a valid copy of the block to L2 cache. The L2 cache changes the state of the cache block from modified-dirty to shared-owned and forwards it on to the requesting processor's L1 cache. When a remote processor requests a modified block, the local L2 cache changes the state of the block from M→O and forwards the block to the requestor. If the requested block is modified-dirty, L2 cache issues a downgrade request to the writing processor's L1 cache. This L1 cache then downgrades that block from M→S and sends a valid copy of that block to the L2 cache. The L2 cache then changes state of that block from modified-dirty to shared-owned and finally sends the data to requesting L1 cache. In both cases, the local L2 cache loses the write privileges for the line; in the latter case, the writing L1 cache also loses its write privileges, which leads to additional writebacks.

Figure 6 shows the impact that wrong-path memory references have on the number of cache block state transitions. The results show that the number of state transitions on the internal network increases by 2% to 13% and by 2% to 17% for 4 and 8-node CMP systems, respectively. On the other hand, the number of state transitions on the external network increases by 1% to 9% and by 1% to 10% for the 4 and 8-node CMP systems, respectively.

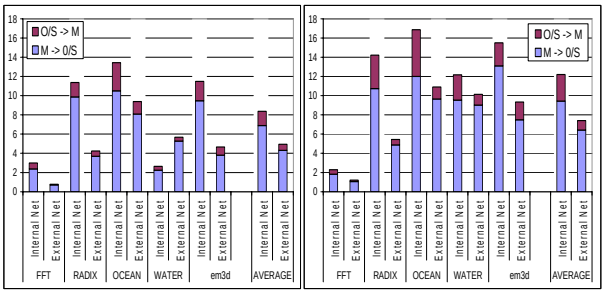

Figure 6. Percentage increase in the number of state transitions due to wrong-path memory references for 4-node (Left pair) and 8-node CMPs (Right pair)

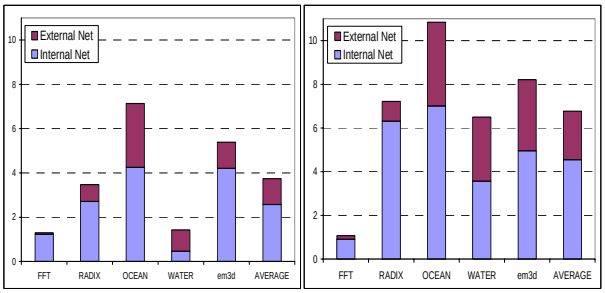

Figure 7. Write misses due to wrong-path memory references for 4-node (Left pair) and 8-node CMPs (Right pair)

Figure 7 shows the number of write misses caused by wrong-path memory references. This figure shows that up to 7% (4-node) and 11% (8-node) of the write misses are caused by wrong-path memory references, each of which subsequently causes an invalidation.

5 Related Work

Other than our previous work in [12], several papers examined the effect that speculative execution had on the performance of uniprocessor systems. Mutlu *et al.* [5] analyzed the performance impact that wrong-path references have for different memory latencies and instruction window sizes. Their results showed that the major reason for performance degradation due to wrong-path memory references is L2 cache pollution.

Sendag *et al.* [8] proposed using the fully-

associative *wrong-path cache* (WPC) to eliminate the cache pollution caused by wrong-path references. The WPC stores data brought into the processor by wrong-path load instructions and evicted from the L1 cache. The processor accesses the WPC and the L1 data cache in parallel. Hence, the WPC functions both as a victim cache [2] and a buffer to store data fetched by wrong-path references. This approach eliminates the pollution caused by wrong-path references in the L1 cache. Sendag *et al.* [9] also studied the effects of incorrect speculation on the performance of a concurrent multithreaded architecture. They analyzed how wrongly-forked threads affected the memory system performance in addition to the known effects by the wrong-path load instructions in a uniprocessor.

Finally, Pierce and Mudge studied the effect of wrong-path memory references on cache performance [6]. Their study used trace-driven simulation, where they injected a fixed number of instructions to emulate the wrong-path. However, this is not realistic because the number of instructions executed on the wrong-path is not fixed in a real processor [1]. They also introduced an instruction cache prefetching mechanism, which shows the usefulness of wrong-path memory references to the instruction cache [7]. Their mechanism fetches both the fall-through and target addresses of conditional branch instructions, *i.e.*, their mechanism prefetches instructions on the taken and not-taken paths.

6 Conclusion

In this paper, we evaluate the effects of executing wrong-path memory references on the memory behavior of cache coherent multi-CMP systems. Our evaluation reveals the following key conclusions:

1. It is important to model wrong-path memory references in multi-CMP systems. Neglecting to model them may result in incorrect design decisions, especially for future systems with longer memory interconnect latencies and processors with larger instruction windows.
2. For multi-CMP systems, not only do the wrong-path memory references affect the performance of the individual processors due to prefetching and pollution, they also affect the performance of the entire system by increasing the number of cache coherence transactions, the number of cache line state transitions, the number of writebacks and invalidations due to wrong-path coherence transactions, and the amount of resource contention.
3. As exemplified by *em3d*, for a workload with many cache-to-cache transfers, wrong-path memory references can significantly affect the coherence actions.
4. Wrong-path memory references may or may not reduce the overall performance of a system. However, if their negative effects, such as, cache pollution, unnecessary coherence transactions and cache block state transitions, are reduced, the system performance can be improved [12].

References

[1] J. Combs, C. Combs, and J. Shen, “Mispredicted path cache effects,” Euro-Par, 1999.

[2] N. Jouppi, “Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers,” International Symposium on Computer Architecture, 1990.

[3] M. Martin, D. Sorin, B. Beckmann, M. Marty, M. Xu, A. Alameldeen, K. Moore, M. Hill, and D. Wood, “Multifacet's General Execution-driven Multiprocessor Simulator (GEMS) Toolset,” Computer Architecture News, 2005

[4] C. Mauer, M. Hill, and D. Wood, “Full-System Timing-First Simulation,” Joint Conference on Measurement and Modeling, 2002.

[5] O. Mutlu, H. Kim, D. Armstrong, and Y. Patt, “Understanding the effects of wrong-path memory references on processor performance,” Workshop on Memory Performance Issues, 2004.

[6] J. Pierce and T. Mudge, “The effect of speculative execution on cache performance,” International Parallel Processing Symposium, 1994.

[7] J. Pierce and T. Mudge, “Wrong-path instruction prefetching,” International Symposium on Microarchitecture, 1996.

[8] R. Sendag, D. Lilja, and S. Kunkel, “Exploiting the prefetching effect provided by executing mispredicted load instructions,” Euro-Par, 2002.

[9] R. Sendag, Y. Chen and D. Lilja, “The Impact of Incorrectly Speculated Memory Operations in a Multithreaded Architecture,” IEEE Transactions on Parallel and Distributed Systems, Vol. 16, No. 3, pp. 271-285, 2005.

[10] S. Woo, M. Ohara, E. Torrie, J. Singh, and A. Gupta, “The SPLASH-2 programs: Characterization and methodological considerations,” International Symposium on Computer Architecture, 1995.

[11] P. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, and B. Werner, “Simics: A full system simulation platform,” IEEE Computer, Vol. 35, No. 2, pp. 50–58, 2002.

[12] R. Sendag, A. Yilmazer, J. Yi, and A. K. Uht, “Quantifying and Reducing the Effects of Wrong-Path Memory References in Cache-Coherent Multiprocessor Systems,” International Parallel and Distributed Processing Symposium, April 2006.

[13] D. Culler, A. Dusseau, S. Goldstein, A. Krishnamurthy, S. Lumetta, T. von Eicken, and K. Yelick. Parallel programming in Split-C,” Supercomputing, 1993.