# Partitioned Cache Shadowing for Deep Sub-Micron (DSM) Regime

Heng Xu and Arun Somani
Dependable Computing and Networking Laboratory
Electrical and Computer Engineering
Iowa State University, Ames, IA 50011-3060
{xuheng,arun}@iastate.edu

## Abstract

*An important issue in modern cache designs is bridging the gap between wire and device delays. This warrants the use of more regular and modular structures to mask wire latencies. This paper advances the basic concepts of shadow caching to offer protection against both data corruption and micro-network disruption in partitioned architectures. Network disruption is tolerated by sending shadow packet along a different route than the original packet, whereas the data corruption problem is addressed by reserving a small portion of the overall cache capacity for in-cache shadow space. Our results show that an average of 96% in data error coverage for Spec2K benchmarks can be achieved and more than 99% of the transient faults on the underlying switched micro-network can also be protected while incurring less than 3% performance degradation in most of the above benchmarks.*

## 1. Introduction

While denser and deeper submicron technology contributes to the increase in transient faults in cache memories, on-chip L2 caches are expected to grow in size. The large caches also have very large latencies due to increased global wire delays across the chip. Bulk of the access time involves routing to and from the banks, not the bank accesses themselves [3]. The access time of conventional lower-level caches has been the longest access time of all sub-arrays, but such uniform access fails to exploit the differences in latencies among sub-arrays. Hence there is a need for more regular and modular cache architectures where the cache is partitioned into banks (connected by a switched micro-network instead of busses) that can be accessed at different latencies. For example, Dynamic Non-Uniform Cache Architecture (D-NUCA) was proposed as a solution to this problem in Kim, Berger, and Keckler [3]. They reported that D-NUCA achieved 1.5 times the IPC of a Uniform Cache Architecture (UCA). While D-NUCA provides performance enhancement in large on-chip caches, issues regarding providing information and communication integrity in partitioned cache architectures remain largely unaddressed.

This paper seeks to improve the transient error behavior of DSM cache architectures that deploy partitioned architectures by exploiting fault tolerance in the underlying micro-network, and overcoming the deficiency of direct application of shadow caching. We propose a fault-tolerant substrate, called Partitioned Cache Shadowing (PCS) and another version of it with micro-network fault tolerance called Shadow-Packet Partitioned Cache Shadowing (SP-PCS), for wire-delay dominated on-chip caches. The basic idea is to accomplish high fault coverage even in the presence of multiple-bit faults without any hardware replication and to avoid the use of a timer-based counter by reusing the distance-based non-uniformity concept exploited in D-NUCA, wherein the *"hottest"* blocks are migrated to the nearest cache banks depending on their usage, and thus making such banks the real candidates for protection. This mechanism can also be bundled with the embedded Error Correction Code (ECC). Our simulation experiments show an average of 96% in cache fault coverage can be achieved for Spec2K benchmarks and more than 99% of the transient faults that can happen to a random hop within the underlying switched micro-network can also be protected while incurring less than 3% performance degradation in most of the above benchmarks.

The rest of the paper is organized as follows. Section 2 explores Partitioned Cache Shadowing (PCS). In section 3, we present a scheme called Shadow-Packet Partitioned Cache Shadowing (SP-PCS), which guards against both data corruption and network disruption in the switched micro-network. Section 4 describes our experimental setup. The simulation results are presented in Section 5. The last section summarizes the contributions of this paper.

## 2 Partitioned Cache Shadowing (PCS)

### 2.1 Our Contributions

Our goal is to provide high protection coverage for spatial or terrestrial applications that require very high data integrity or operate under highly noisy environments where parity- and ECC-based protection schemes are not satisfactory. It is easier to include shadow caching in partitioned architectures due to its specific features such as distance-based access latency and dynamic swapping on cache hits that provide a convenient way to replace timer-based dead-block estimator with a distance-based dead-block predictor in partitioned cache architectures. The accuracy of the timer-based predictor can be further enhanced by exploiting the fact that the MFU lines in partitioned structures are location-wise fixed (i.e. in the way-sets that are closest to the cache controller and in the MFU cache line of each set in the rest of the cache banks). Moreover, previous work on Network-On-Chip(NOC) fault tolerance focused mainly on communication reliability. We propose a novel fault-tolerant substrate superimposed on the partitioned cache architecture by combining it with shadow caching and also include protection against both data corruption and micro-network disruption. We tolerate the micro-network disruption by replicating information packets and sending a shadow packet along a different route than the original packet. The data corruption problem is addressed by reserving a small portion of the overall cache capacity for in-cache shadow space.

### 2.2 Partitioned Cache Shadowing

Similar to [3], this paper organizes the L2 cache in a two dimensional n x n mesh of cache banks as shown in Fig. 1 for an example of n=8. Each of the n*n banks is a k-way set associative cache where k=4 as shown in Fig. 2. Each column of banks represents a *way-set*. Each way-set has similar access time. Way-set0 is thus the fastest and way-set7 is the slowest to access. A new cache line is brought into way-set0 and propagates towards way-set7 with time as swaps happen. The baseline configuration implements various schemes for mapping cache banks in way-sets, data searching, promotion of cache lines, and replacement. We in this paper deploy the following mechanisms:

- *Simple mapping.* With simple mapping, each column of banks in the L2 cache becomes a *bank-set*. All banks within that bank-set have different access time due to their distances from the cache controller. Each row of banks in L2 cache becomes a *way-set*. All banks within each way-set have similar access time.

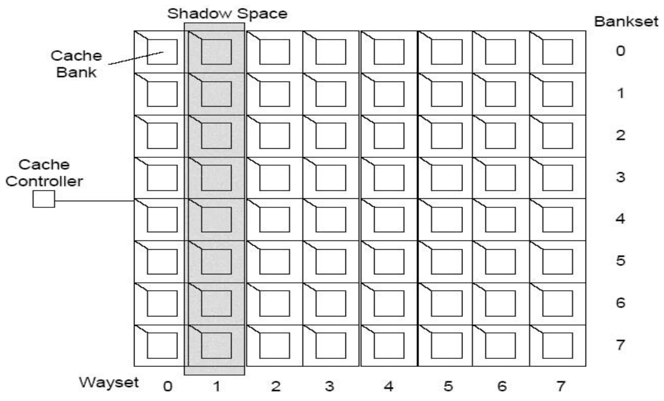

**Figure 1. PCS within a Bank-set in partitioned cache architectures. Each square represents a cache bank. The highlighted column shadows the rest of the cache space.**

- *Multicast Search with Smart Search.* The cache is searched for a line by first selecting a bank-set, then selecting the set within the bank-set and finally performing a tag match on banks within that bank-set. The requested address is multicast to some or all of the banks in the requested bank-set. Tag lookups proceed simultaneously, but at different real times due to wiring delays in the switched micro-network. Smart search, i.e. partial tag comparison, maintains a cache of partial tags for speedy tag lookups.

- *One-bank Promotion, Dynamic Swapping on Cache Hits.* When a hit occurs to a cache line, it is swapped with the line in the bank that is next closest to the cache controller. Most-frequently-used used cache lines are thus promoted towards closer and faster banks.

- *Zero Copy Insertion, Head Replacement.* On cache misses, a replacement is found in the nearest bank for a bank-set. The victim cache line is evicted from the entire L2 cache.

On the **coarse-grained** bank-set level (see Fig. 1), we propose to use way-set1 to provide a complete shadow of way-set0. On the **finer-grained** cache-bank level(see Fig. 2), we propose to use a fixed cache line in each set to shadow the MFU lines in each cache bank. Recall that each bank is a multi-way set associative cache. It was reported in [3] that most of L2 cache hits are concentrated on way-set0 of partitioned cache architecture. Based on this observation, we propose to reserve way-set1 as the shadow for way-set0 (Fig. 1). The use of way-set1 as the shadow is mainly based on performance concerns since way-set1 is nearer to the cache controller and shadow space accesses

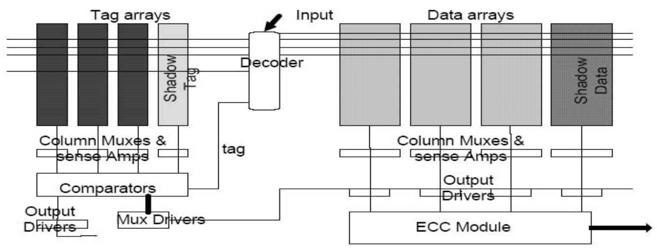

**Figure 2. Shadowing within a Cache Bank. The highlighted tag/data array shadows the rest of the cache bank tag/data arrays respectively.**

can be quite frequent. Other way-sets can also be selected for shadowing. With dynamic swapping, most-frequently-used cache lines will tend to gravitate towards way-set0. Whenever there is a cache access in way-set0, the corresponding shadow cache line in way-set1 is used for verification. While the main cache serves the pipeline's request, the shadow way-set synchronously monitors the integrity of the main cache and updates the shadow cache lines. Since all of the shadow cache operations take place in parallel with primary data cache, performance degradation resulting from using such a scheme is minimal. Notice that we do not require any counter to be associated with a cache line as is the case in [4].

To enhance the protection further, we propose a **finer-grained** scheme along with the above shadow caching at the bank-set level to guard against data corruption in each cache bank. So and Rechtchaffen [5] showed that more than 90% of cache hits were to the most recently used ways in a four-way set associative cache. For banks not protected by the first scheme in bank sets 2 to n-1 (7 in our example), we reserve one cache line per set (including both tag and data) in a set-associative cache bank as the shadow cache line as shown in Fig. 2. Barring positive verification by either ECC or Comparison unit, that primary line is considered unprotected.

- *Scenario 1 (a p-space read hit and a s-space read hit):* Once we have a read-hit in the s-space, we know that it will also be a read-hit in p-space. Both of the data copies are retrieved from respective cache lines and the cache controller verifies their contents (using ECC if available or by comparison). Verified data are presented to the processor. If there is a mismatch, the cache controller retries the operation. If the error persists, the cache controller invalidates cache lines if they are clean, or declares a computation error.
- *Scenario 2 (a p-space read miss and a s-space read miss):* If the cache controller finds that there is a p-space read miss, it doesn't check the s-space because s-space will not have the data either. It will retrieve the new data from the main memory and place separate copies of the data in both spaces.
- *Scenario 3 (a p-space write hit and a s-space write hit):* If both the p-space and s-space find that they have a write hit, both copies are updated.
- *Scenario 4 (a p-space write miss and a s-space write miss)*: If the cache controller finds that there is p-space write miss, it doesn't check the s-space because s-space will not have the tag and data either. It will retrieve the new data from the main memory and place separate copies of the data in both spaces.
- *Scenario 5 (a p-space write hit and a s-space write miss):* When the cache controller finds a write hit in the p-space, it may still incur a s-space write miss. The new write data will be deposited in both spaces.
- *Scenario 6 (a p-space read hit and a s-space read miss):* When the cache controller finds a read hit in the p-space, it may still have a s-space read miss. In this case, the data is not protected by any shadow cache lines. If the ECC module finds this data invalid, it will repeat the same cache transaction again or trap to operating system for further handling.

## 3 Shadow-Packet Partitioned Cache Shadowing (SP-PCS)

SP-PCS (Shadow-Packet PCS) is similar to PCS scheme of the previous section and uses the same underlying shadow caching principle. However, the shadow mechanism deployed is different. In this case, we select one row and one column of cache banks for s-space and do not provide any more shadowing inside a bank in way-set 2 to n-1 as in PCS. As highlighted in Fig. 3, our scheme uses one of the nearest bank-sets (bank-set4) and the second nearest way-set (way-set1) to the cache controller as the shadow space (s-space) for fault tolerance. The reason for this is a design issue with a bias towards balancing protection coverage with performance. Since shadow space access is a fairly frequent operation, it is desirable to pull the shadow space as close as possible to the cache controller to mitigate possible performance degradation. We also assume simple mapping, multi-cast search plus partial tag array search, one-bank swapping, zero-copy and head replacement as in PCS.

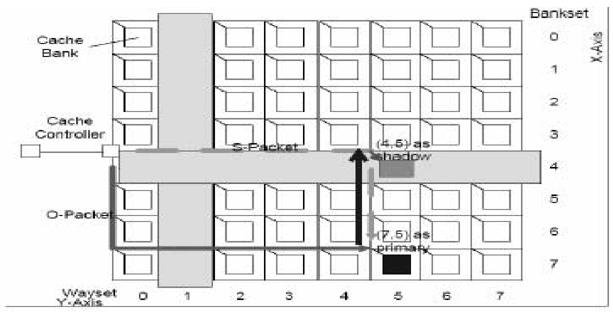

**Figure 3. Shadow-Packet PCS (highlighted cache banks are designated as the shadow space for data fault tolerance, and a shadow packet will be sent along with the original packet for communication fault tolerance)**

In order to deal with possible deadlocks in the micronetworks, we modeled contention by using wormhole-routed flow control [2]. Exactly one packet can be queued at a specific switch in a bank while another is being serviced. Other banks along different network paths can still be accessed. Cycles are allowed due to the x-y and y-x routing of the original and shadow packets. This issue can be aggravated due to the combined use of the shadow cache with shadow packets since more packets are contending the links in the micro-network.

A bank in bank-set4 is used to store the MFU cache lines of the cache banks in the corresponding way-set to which it belongs. This is actually very similar to what we used in PCS. The only difference is that in PCS, we distributed the shadow ways to each of the cache banks, whereas in SP-PCS, we aggregate these shadow ways in a contiguous shadow space in bank-set4. Way-set1, however, is still used as a shadow for way-set0, which contains most of the MFU cache lines due to dynamic swapping. The rest of the L2 cache capacity is used to store primary data, which are protected by the shadow data in the shadow space. SP-PCS stores data copies along with ECC in both p-space and s-space. An additional goal here is to provide integrity checking for communication micro-network. Consider the same example of an 8x8 two dimensional mesh micro-network based partitioned cache architecture as shown in Fig. 3. Each cache bank is a k-way set associative write-back cache where k=4 here. We use (x,y) pairs to indicate the location of a specific cache bank. For example, banks (7, x), x varying from 0 to 7, are banks in bank-set7, where the number '7' is their common 'x' coordinate. The original D-NUCA scheme uses x-y routing by first sending packets along x coordinate, and then perform a multi-cast search along the y coordinate among the corresponding sets in different cache banks. Suppose Bank (7, 5) is the desired destination of a cache line to be accessed. To provide fault tolerance against transient faults in the communication micro-network, our scheme sends out a shadow packet (s-packet) along with the original packet (o-packet) using a different route. It also uses packets with CRC information attached to that. Also, each packet will have a sequence number (seqno). Every time an o-packet (data or control) is sent using x-y routing, a s-packet with the same seqno and payload is also sent using y-x routing. Note that x and y axes in Fig. 3 are rotated by 90 degrees to better accommodate the position of the cache controller.

The operation of SP-PCS scheme involves six operational scenarios based on the micro-network communication pattern. As stated before, any cache line that is not in the primary data space will not be present in the shadow space. So we can have only six possibilities (listed as below). The rest of the scenarios are not possible due to the inclusive nature of our shadow space within the primary data space. Note that our experiments do take contention issues into consideration. We will also use the same example of Fig. 3 in the following discussion on various scenarios. The destination bank is always bank (7,5) and the shadow bank is always bank (4,5). The cache controller always sends out two packets, o-packet via (x-y routing) and s-packet (via y-x routing). The destination or shadow bank, on the other hand, always sends back two packets as well as response packets.

- *Scenario 1 (a s-packet read hit followed by an o-packet read hit):* In this case once we have a s-packet read hit at bank (4,5), we know that we will have a primary cache hit later at bank (7,5) barring the loss of the o-packet en route. After CRC verification, bank (4,5) reads the data from s-packet and sends a reply packet with the retrieved data as payload immediately. Later on, the primary bank (Bank (7,5)) will also send its reply packet for verification when the o-packet is received by it. If the cache controller does not receive packets on both occasions, it may retry. Upon failure of a retry, it may declare a computation error.

- *Scenario 2 (a s-packet read miss followed by an o-packet read miss):* In this case, only the primary bank can initiate a read from main memory as a s-space miss is also a likely scenario. After fetching the new data from the main memory, both bank (4,5) and bank (7,5) will have the copies of the new data installed. Both banks can then send their response data packets back to the cache controller using different routes.

- *Scenario 3 (a s-packet write hit followed by an o-packet write hit):* If bank (4, 5) discovers that it is a write hit the s-packet will update the cache line and set the dirty bit in bank (4,5). The packet is also routed to bank (7, 5). Either the o-packet or the s-packet will update the cache line there.

- *Scenario 4: (a s-packet write miss followed by an o-packet write miss):* When the s-packet reaches bank (4,5) and finds a write miss, the shadow bank forwards the packet to the primary bank. It also keeps a copy of the packet in case the primary bank also finds a write miss and the cache line is read from the main memory. In that case the s-packet is used to update the line with the new data. Upon finding a write miss in the primary bank, the line is read and stored at both the primary and shadow banks along with the data update and dirty status.

- *Scenario 5 (a s-packet write miss followed by an o-packet write hit):* Similar to the scenario 4, the shadow bank forwards the s-packet to the primary bank while retaining a copy for its possible future use. The primary bank will find a hit and will generate both o- and s-packets as the response.

- *Scenario 6 (a s-packet read miss followed by an o-packet read hit):* When the s-packet reaches bank (4,5) it will find a read miss. The shadow bank assumes that the primary has the data and does not resort to any additional action. In this scenario, the o-packet will find a read hit and there is a corresponding data copy in bank (7,5). This is a case where the data is not protected.

## 4 Experimental Settings

We explored a large design space in both PCS and SP-PCS, and evaluated both of them with a variety of parameters. We considered four transient fault models for cache memories (direct, random, adjacent, and column) [6]. Since there is little difference in the final results among the above mentioned fault injection models, we present the results only for the random fault injection model. Since we assume that the PCS scheme executes the shadow cache accesses totally in parallel with the main L2 cache and that the latency of such activities are largely overlapped with the L2 cache access latency, we have mainly focused on the protection coverage measurements. We implemented both of the schemes in sim-outorder, an out-of-order Alpha 21264 superscalar microprocessor simulator [1]. The default simulation values used in our experiments are given in Table 1.

**Table 1. Simulation Setup**

| Config. Parameters | value |
|---|---|
| Functional Units | 4 integer ALUs |
| | 4 integer ALUs |
| | 1 integer multiplier/divider |
| | 4 FP ALUs 1FP mul./div. |
| LSQ size | 8 Instructions |
| RUU size | 16 Instructions |
| Issue Width | 4 instructions/cycle |
| L1 Inst/Data Cache | 16KB, 2-way, 3 cycle lat. |
| Unified L2 Cache | 4MB, 64 x 64 KB |
| L2 Cache Bank | 4-way, 64B block, 6 cycle lat. |
| No. of L2 Cache Banks | 64, 8 rows x 8 cols |
| Technology | 50nm |

We simulated the protection enhancement brought about by transmitting and receiving shadow packets in a L2 cache within a wormhole-routed 2-D mesh micro-network. We use applications from both the Spec2000 Integer suite (specINT) and the Spec 2000 Floating-point suite (specFP) for evaluation. Note that the use of specINT and specFP may be likely to slightly increase the fault coverage due to their small footprint and regular access patterns. Nevertheless, we still elected to use these two benchmark suites due to our intention to compare the results with other methods in the literature. Different design parameters, such as the number of cache banks in the micro-network, the associativity of each individual cache bank, etc. are investigated in our experiments.

### 4.1 Evaluation Metrics

In order to notice the improvement in protection coverage, we use different metrics in the study to show the effectiveness of the proposed schemes. Simulation results have been obtained through the execution of 100 million instructions after skipping around 400 million initial instructions in a variety of benchmarks in the Spec2K suite. *Shadow Capacity Ratio (SCR)* is the percentage of shadow space in the whole cache space and a smaller SCR indicates better real estate utilization. *Loads with Shadow (LWS)* is the fraction of load hits that also find a shadow in the shadow space at the time of the load. A higher fraction implies higher protection coverage. *Successful Transfer Ratio (STR)* is the percentage of successful transfer of at least one of the original or shadow packets to the intended destination cache bank in the underlying switched micro-network. This metric shows the effectiveness of our SP-PCS scheme against transient faults that can occur in the micro-network. *Instructions Per Cycle (IPC)* is the number of instructions executed per cycle. A higher number generally indicates higher performance.

## 5 Results

### 5.1 Area Efficiency

In PCS, we use one way-set (way-set1) and one way in the rest of the banks in way-sets 2 to n-1 for shadow caching. Assuming that each bank is k-way set associative, the shadow cache size in a n x n two dimensional mesh is equal to the combination of one way-set (n cache banks) and 1/k of the remaining n*(n-2) banks. Thus the ratio of shadow cache to total cache is given by PCS shadow cache capacity ratio $SCR_{pcs} = 1/n + (n-2)/(n*k)$. For example for n=8, k=4, it is 31.25%.

For SP-PCS, the shadow cache capacity used is n + n-2 (n-2 for remaining n-2 way-sets, except the first two way-sets) banks. Thus the ratio of shadow cache to total cache is given by SP-PCS shadow cache capacity ratio $SCR_{sp-pcs} = (2n-2)/n*n$. For n=8, the ratio is almost 22%. It is smaller than that of PCS.

### 5.2 Protection Coverage

Benchmarks from both SPEC2kInt and SPEC2kFp are simulated and protection coverage results are obtained for PCS. As shown in Fig. 4, PCS achieves an average Loads with Shadow (LWS) of around 96% in Spec2k, which means the vast majority of the L2 loads are protected by the PCS shadowing mechanism. For most of the Spec2K benchmarks, the LWS of PCS is about 98% or 99%. Note that for *eon, mesa, vpr, equake* and *vortex*, their LWSs are very close, but not equal, to 100%. The *art* benchmark records the lowest loads with shadow (83%), whereas *equake* etc. register almost full load coverage. This large gap in protection coverage is in part due to each benchmark's L2 access pattern. For example, the benchmark *art* has a widely-spaced L2 access pattern after the filtering effect imposed by L1 cache, which can only be loosely fit into the limited shadow space in PCS. So we notice a sharp drop in protection coverage. On the other hand, *equake* has a much more regular access pattern, which can be easily fit into the shadow space. Since PCS is based on the dynamic nature of the underlying partitioned architecture design, we have also conducted experiments with different configurations of the overall micro-network topology and the associativity of each individual cache bank.

#### 5.2.1 Protection Coverage for Different Configurations of Micro-Network Topology

To measure the impact of different cache configurations of the underlying partitioned architecture on the effectiveness of our scheme, we considered the following two cases: *Case 1*: 4 rows x 4 cols with 4-way set-associative cache bank, one shadow way for each bank, and one shadow way-set for each micro-network configuration. *Case 2*: 8 rows x 8 cols with 4-way set-associative cache bank, one shadow way for each bank, and one shadow way-set for each micro-network configuration. Increasing the number of cache banks in the cache micro-network can make the dynamic nature of partitioning more pronounced. Although the relative capacity allocated for shadow space (SCR) drops from around 37.5% to 31.25%, the replication ability is affected very little by the different configuration parameters (a drop of less than 2% in Loads with Shadow for most tested benchmarks), as shown in Fig. 5. Some applications in the Spec2k Floating Point Suite even experience an increase in loads with shadow. Generally speaking, Spec2kFp is much more loop-based and has a smaller and more regular working set than Spec2Kint and thus can have a higher percentage of MFU cache lines located in the cache banks that are closest to

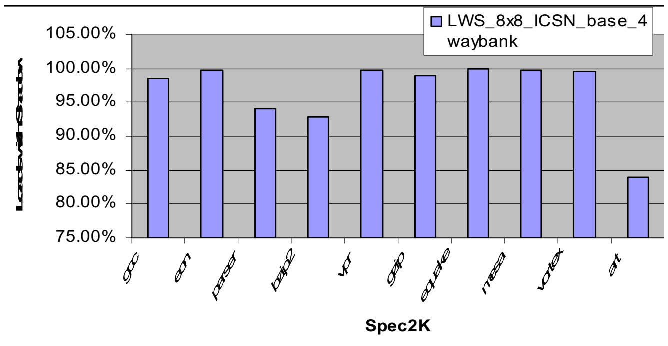

**Figure 4. PCS Protection Coverage (LWS) Results for 10 Applications in Spec2K**

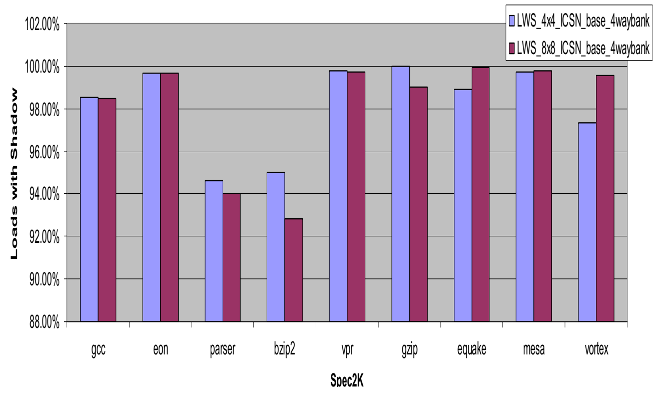

**Figure 5. Loads with Shadow (LWS) for Different Configurations of the Cache Micro-Network**

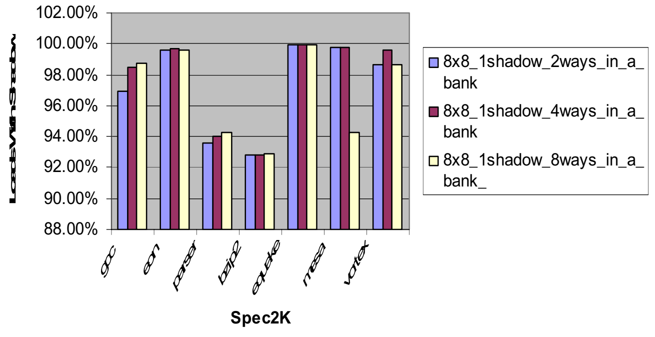

**Figure 6. Loads with Shadow for Different Associativity of a Cache Bank**

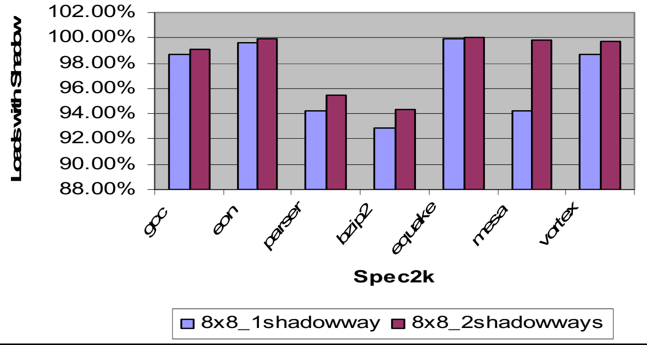

**Figure 7. Protection Coverage for Different Number of Shadow Ways in a Cache Bank**

the cache controller. Since the primary data space that is protected by the shadow space is location-wise fixed (way-set1 and MFU cache lines in the rest of the cache banks), it is not surprising that PCS demonstrates higher protection coverage in Spec2kFp than in Spec2kInt.

#### 5.2.2 Protection Coverage for Different Associativity of a Cache Bank

We also conducted experiments on the following three cases with different associativity in a cache bank. Since we maintain the size of the shadow space as fixed, the increase in cache bank associativity can lead to relative decrease in protection coverage. We considered three cases. *Case 1:* 8x8 with 2-way cache bank with one Shadow way in a set and one shadow way-set. *Case 2:* 8x8 with 4-way cache bank with one Shadow way in a set and one shadow way-set. *Case 3:* 8x8 with 8-way cache bank with one Shadow way in a set and one shadow way-set.

As shown in Fig. 6, although the relative capacity allocated for shadow space is reduced from 50% in case 1 to 22% in case 3, increasing associativity of a cache bank does not change much of our scheme's protection coverage. A relative small shadow space in L2 cache can thus be allocated, which can promote overall system performance with a bigger primary data space.

#### 5.2.3 Protection Coverage for Different Number of Shadow Ways

The impact of different number of shadow ways for in-cache shadowing in the second level cache is further investigated in Fig. 7. The purpose here is to show that one shadow way in banks (other than those in way-set0 and way-set1) is sufficient to provide good protection coverage. We consider 1 way and 2 ways in an 8 x 8 mesh for banks in way-set 2 to n-1 for protection. The cache banks are 8-way set associative. Even though the shadow space capacity in L2 cache increases, protection coverage does not improve significantly. More specifically, one shadow way for each set in a cache bank seems to be a better design choice since more cache capacity can be used to enhance performance of the system.

### 5.3 SP-PCS Protection Results

We do not include protection coverage experimental results for this case for the sake of brevity because the basic protection scheme for protecting way-set0 is identical and the rest of the primary data space are protected in a similar way in both PCS and SP-PCS. Our goal here is to study the impact of communication micro-network transient faults in this case. We considered random fault injection model. In this model, an error is injected in a random link in the switched micro-network at a random time. Transient faults are deemed to have a probability of corrupting a packet during each hop. Since we only intend to show the strengths of our SP-PCS scheme under very heavy bombardments from injected errors, we pick an error injection rate of 1/100000 for all of the tested benchmarks with an instruction size of about 100 million instructions each. Note that transient faults should be much rarer in a more realistic setting. Percentage of successful transfers is a measure of protection coverage. Since each cache transaction sends out two packets, o-packet and s-packet, the scenario where at least one of them arrives at the destination with an uncorrupted CRC is considered a successful transfer.

As shown in Fig. 8, our SP-PCS scheme offers nearly 100% coverage for packet transmission and routing in the switched micro-network for all of the tested benchmarks in Spec2K. Combined with the data fault tolerance capability

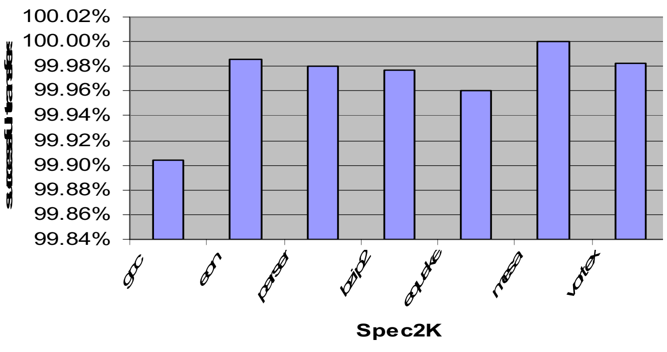

**Figure 8. Percentage of Successful Transfer for Communication Fault Tolerance**

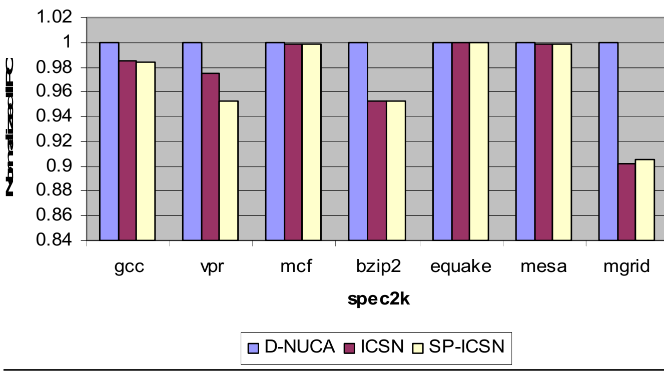

**Figure 9. Normalized Change in IPC from PCS**

in the s-space, SP-PCS offers a complexity-effective alternative to guard against both data errors and network disruption.

## 5.4 Performance Results

As shown in Fig. 9, almost all of the listed benchmarks in Spec2k suffered only slight performance degradation from imposing our fault-tolerant substrate on the underlying D-NUCA design. The IPCs of most of the benchmarks decreased by less than 3%, which is acceptable tradeoff with much higher fault coverage under highly noisy environments. The biggest dent in IPC from using our substrate comes from the benchmark *mgrid* (a little less than 10%), where the benchmark's performance is very sensitive to the reduction in the size of the L2 primary space. The reason for *mgrid* to experience a slight surge in performance from using SP-PCS is that injecting shadow packets into the underlying switched network may change the bank or link contentions we've modeled in the 2-D mesh. Most of the latencies from injecting shadow packets are overlapped by those from injecting original packets and some shadow packets may prevent original packets from entering congested switches, thereby hiding some of the latencies. Based on Fig. 9, it is shown that both using in-cache shadowing for data protection and sending duplicate packet for communication protection bring relatively little performance overhead.

## 6 Conclusion

This paper proposes a novel partitioned cache shadowing substrate in large on-chip caches. We use the existing remote cache banks to hold replicas of a cache block while capitalizing on the non-uniformity of L2 caches, wherein the "hottest" blocks will be migrated to the nearest cache blocks, thus leaving other banks candidates for replication. To guard against transient faults in the underlying micro-network, we send out a duplicate packet along a different route than the original packet. Our results show significant improvements in protection coverage with little performance degradation.

## Acknowledgements

The research reported is funded in part by NSF under grant CCF 0311061 and Jerry R. Junkins Endowment at Iowa State University.

## References

[1] D. Burger and T. M. Austin. The simplescalar tool set, version 2.0. *SIGARCH Comput. Archit. News*, 25(3):13–25, 1997.

[2] J. Duato. New theory of deadlock-free adaptive routing in wormhole networks. *IEEE Transactions on Parallel and Distributed Systems*, 4(12):1320–1331, 1993.

[3] C. Kim, D. Burger, and S. W. Keckler. An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches. In *ASPLOS*, pages 211–222, 2002.

[4] S. Kim and A. K. Somani. Area efficient architectures for information integrity in cache memories. In *ISCA '99: Proceedings of the 26th annual international symposium on Computer architecture*, pages 246–255. IEEE Computer Society, 1999.

[5] K. So and R. N. Rechtschaffen. Cache operations by mru change. *IEEE Trans. Comput.*, 37(6):700–709, 1988.

[6] A. K. Somani and K. Trivedi. A cache error propagation model. In *Proc. of Intl' Symp. on Pacific Rim Fault Tolerant Computing*, pages 15–21, December 1997.