Probabilistic Soft Error Detection Based on Anomaly Speculation

Joonhyuk Yoo*

Abstract—Microprocessors are becoming increasingly vulnerable to soft errors due to the current trends of semiconductor technology scaling. Traditional redundant multi-threading architectures provide perfect fault tolerance by re-executing all the computations. However, such a full re-execution technique significantly increases the verification workload on the processor resources, resulting in severe performance degradation. This paper presents a pro-active verification management approach to mitigate the verification workload to increase its performance with a minimal effect on overall reliability. An anomaly-speculation-based filter checker is proposed to guide a verification priority before the re-execution process starts. This technique is accomplished by exploiting a value similarity property, which is defined by a frequent occurrence of partially identical values. Based on the biased distribution of similarity distance measure, this paper investigates further application to exploit similar values for soft error tolerance with anomaly speculation. Extensive measurements prove that the majority of instructions produce values, which are different from the previous result value, only in a few bits. Experimental results show that the proposed scheme accelerates the processor to be 180% faster than traditional fully-fault-tolerant processor with a minimal impact on overall soft error rate.

Keywords—Probabilistic Soft Error Detection, Reliability, Anomaly Speculation

1. INTRODUCTION

Microprocessors are becoming increasingly vulnerable to soft errors due to the current trends of semiconductor technology scaling. Using Redundant Multi-Threading (RMT) [11, 15, 16, 20] or Dynamic Verification Implementation Architecture (DIVA) [1] is an intriguing approach for concurrent error detection and recovery. RMT-like architectures use two separate threads and execute an instruction stream redundantly for perfect fault coverage, while DIVA-like architectures employ a checker processor to re-execute the computations of a complex core processor. However, these fully redundant architectures significantly increase the pressure on the processor resources and result in severe performance degradation [4, 9, 22].

To mitigate the verification workload for fault-tolerant processors, we advocate an Active Verification Management (AVM) approach [22], which employs a filter checker guiding verification priority before the re-execution process starts at either redundant thread or checker processor. In this paper, an anomaly-speculation-based filter checker is proposed as one of pro-active

※ This work was supported by the Daegu University Research Grant, 2010.
Manuscript received September 6, 2010; accepted February 13, 2011.
Corresponding Author: Joonhyuk Yoo
* College of Information and Communication Engineering, Daegu University, Gyeongsan, Korea(joonhyuk@daegu.ac.kr)

verification management techniques. Anomaly speculation is accomplished by exploiting a value similarity property micro-architecturally, characterized by a frequent occurrence of partially identical values with the same subsets of consecutive bits.

The property has been repeatedly exploited so that the distribution of used values is very non-uniform. The observation that a few values account for the majority of values used, leads the distribution of data values to be strongly biased, which means instead of assigning verification resources uniformly to every instruction, a value locality can be exploited to prioritize verification candidates for an efficient fault tolerance mechanism. The bias in data values is taken advantage of by several micro-architectural techniques, such as value prediction [10], instruction reuse [18], cache hierarchy [6], and efficient register file organization [5, 7], etc. This research especially explores the repeated occurrence of multiple live value instances identical in a subset of their bits, described as a partial value locality [5]. After investigating the partial value locality of instruction streams, we found that Hamming distance of two consecutive computation results is usually small, indicating that only a few bits in the result value vary among different execution instances. This implies that the partial value locality exists in destination operands and its scope is strongly biased.

Based on the biased distribution of Hamming distance, this research investigates further how to exploit the observed partial value locality for soft error tolerance. Because a soft error may incur a transient one-bit flip that cannot be easily detected by monitoring its Hamming distance, this paper re-defines a micro-architectural similarity distance metric to detect an abnormal deviation away from a nominal similarity distance of computation results, as a special type of partial value locality. This work shows how the value similarity property can be used in the design of a filter checker to lower the pressure on the processor resources, improving its performance with a minimal effect on overall reliability.

Detection of an anomaly in instruction streams is achieved by a speculative value similarity distance cache, followed by an anomaly tester, to mark a re-execution candidate for further verification. The similarity distance cache dynamically captures a run-time behavior of similarity between result values and keeps learning the nominal scope of variance with a confidence score. When an instantaneous variance produced by instruction instances exceeds the nominal similarity distance, a likely-to-be soft error can be marked by the anomaly tester. Fig. 1 describes a block diagram of the proposed anomaly-speculation-based filter checker. With a value distance predictor and an anomaly tester, the proposed scheme speculates an anomaly and provides soft error tolerance based on the anomaly test. This paper proves the existence and the frequency of occurrence of similar values and presents how the characterized value similarity can be exploited in the design of anomaly-speculation-based filter checker.

This paper is organized as follows. Section 2 discusses the related work. Section 3.1 investi-

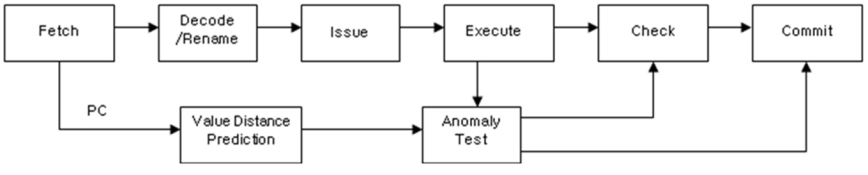


Fig. 1. Block diagram of the proposed soft error prediction architecture

gates a partial value locality and characterizes distributions of both Hamming distance and micro-architectural similarity distance. Section 3.2 describes an anomaly-speculation-based active verification management and Section 3.3 implements an anomaly filter checker. Section 4 evaluates the impact of the proposed mechanism on performance and reliability. Finally, future work and conclusion are discussed in Section 5.

2. RELATED WORK

Computer architects have repeatedly shown that data values have a strongly-biased distribution, implying that a few values account for the majority of values used. This value locality has been exploited by several micro-architectural techniques. Both value prediction [10] and instruction reuse [18] are such techniques, which takes advantage of identical values seen repeatedly in the pipeline. Another usage is the cache hierarchy, which works well because of the principle of address locality. They exploit either temporal or spatial locality in the same data values.

A partial value locality is defined by the repeated occurrence of partially identical data values. Some techniques exploit this property for organizing a content-aware integer register file [5] or sharing physical registers [7]. Value similarity is a special case of partial value locality. A similarity measure is usually used to cluster objects into groups with similar properties. This work characterizes it with a distance metric to improve performance and reliability of fault-tolerant processors. Some techniques exploit narrow values to optimize processor resources [2, 8]. Narrow values have a simple high-order bit pattern with either all 0's or all 1's and do share high-order bits. Therefore, the narrow value is a subset of similar values and it is treated in a simple way in the set of similar values.

A host of concurrent system-level fault tolerance techniques has been proposed exploiting additional redundant execution processors [1, 11, 13, 15, 16, 19, 20]. Such redundant execution techniques employ Simultaneous Multi-Threading (SMT) [11, 15, 16, 20] or Chip Multi-Processing (CMP) [1, 19] to provide coarse-grain instruction replication with a modest amount of additional hardware support. Because the main thread shares the limited resources with the redundant thread, the performance degradation problem caused by redundant execution is common to all fully-fault-tolerant architectures. Most of previous studies have focused mainly on the achievement of reliability with little attention on the performance overhead incurred by re-execution. The proposed work considers performance of fault-tolerant processors to explore the trade-off between performance and reliability.

Exploring a partial redundancy for soft error detection is similar to opportunistic transient fault detection [4], in which the redundancy is opportunistically adjusted according to the amount of Instruction Level Parallelism (ILP). Our approach exploits a partial value locality property and manages the redundancy pro-actively based on the biased distribution of similar values. Given the proposed fault detection is based on an anomaly speculation, it is worth to address the differences between our approach and ReStore [21]. Their symptom-based soft error detection considers mispredictions of high confidence branches as symptoms of soft errors. Instead of using branch mispredictions as symptoms, we use an abnormal result value deviation from the nominal similarity distance as a potential soft error, exploring a partial value locality.

3. PROBABILISTIC SOFT ERROR PREDICTION

3.1 Characterizing Value Similarity Property

Distance metrics can be used to determine similarity or dissimilarity between two data values. One of well-known similarity distance measures is Hamming distance [3], which is the minimum number of bits that should be modified to convert one value into another. The variance between two data values can be computed by using exclusive-OR on corresponding bits. Its Hamming distance is obtained by the number of 1's in the exclusive-OR bit string. In fact, this is a dissimilarity measure since bigger value shows disagreement in two data values.

Fig. 2 (a) and (b) shows the distribution of Hamming distance between two consecutive computation result values in the SPEC2000 benchmark suite. The cumulative distribution function indicates a strong bias in the destination operands, confirming that a small range of values account for the majority of values computed. Since Hamming distance is a measure of how much two values disagree each other, the distribution function proves how likely two consecutive results are similar. For instance, it shows that a single value accounts for 36% of all results across

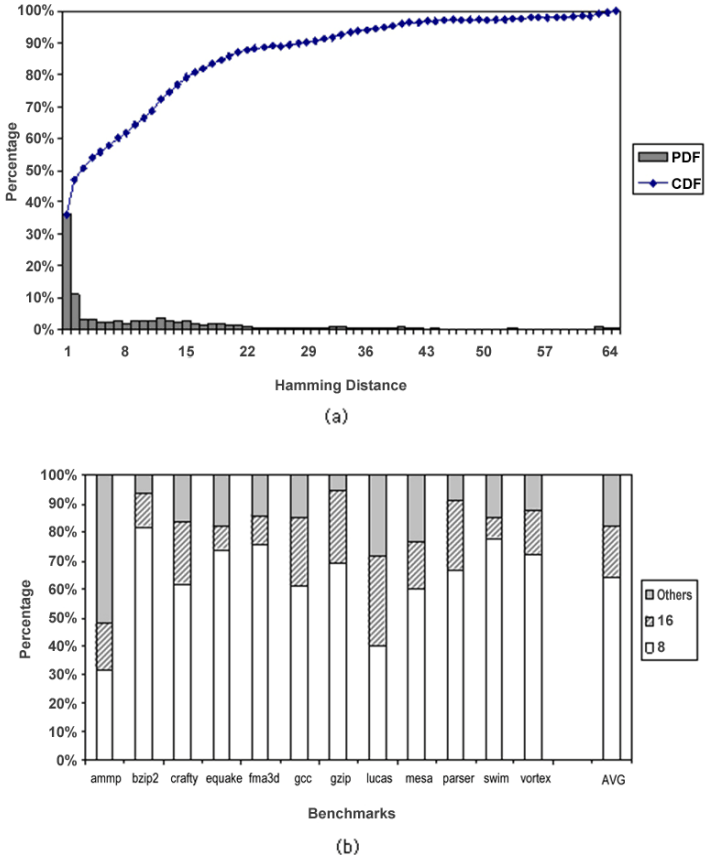


Fig. 2. Distribution functions of (a) Hamming distance and (b) n-dissimilar values

SPEC2000 programs. Furthermore, 82% of all values vary only within the scope of 16-bits width, showing that partially identical values occur frequently in a dynamic instruction stream.

Without a partial value locality, the distribution would have been uniform. This paper extends its application to soft error tolerance. Although Hamming distance is a good indicator of how likely values are similar, the metric is not suitable to detect a soft error of one-bit flip in the original data. For example, let's consider a 64-bit binary value 0x0000000000000001 and a bit flip occurs at the most significant bit position. The value then becomes 0x8000000000000001 and then its Hamming distance with previous value 0x0000000000000003 changes from 1 to 2, which is hard to distinguish whether the value is defected. This paper proposes a new similarity distance measure to easily detect an abnormal deviation from the result values. More specifically, two 64-bit values are called *micro-architecturally n-dissimilar* if they only differ in *n* least significant bits and are equal in the remaining *(64-n)* high-order bits. Micro-architectural similarity distance does not only reflect the fact that two computation results are near one another in the Hamming distance but also captures a deviation scope from one another. Therefore, the proposed similarity distance is a special type of partial value locality.

The result in Fig.3 demonstrates that a lot of similar values do exist. Fig.3 (b) illustrates the distribution of *micro-architecturally n-dissimilar* values, indicating that the amount of similar values is still high. More precisely, *8-dissimilar* values account for 52% of all result values

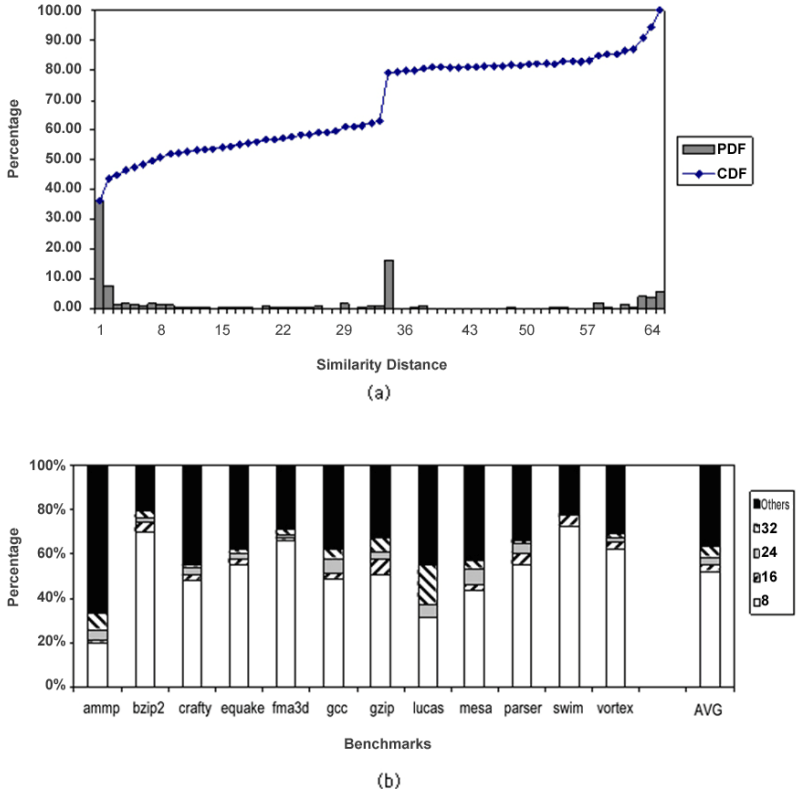


Fig. 3. Distribution of value similarity distance

across the SPEC2000 programs, including 36% of exactly the same values. Different sections of the bar graph show the fraction of *8-dissimilar*, *16-dissimilar*, *24-dissimilar*, *32-dissimilar* and the rest of them, respectively. The cumulative distribution function in Fig.3 (a) shows a similar distribution to the narrow value's, presented in the previous literature [2], indicating that there exists a 16% jump from 33 to 34 distance, besides 36% jump from 0 to 1. This is because the memory address in the Alpha ISA uses 33 bits and reflects the fact that memory address operations followed by usual data operations incur a large deviation from the previous result. To simplify the notation, *micro-architecturally n-dissimilar* values will be just called *similar* in the remainder of this paper.

3.2 Predictability of Similarity Distance

Section 3.1 showed the facts that many values are micro-architecturally similar, a group of similar values shares the high-order bits and each value instance in the group is uniquely represented by its remaining least significant bits. Therefore, if the information on the similarity distance would be available before the re-execution process starts, only the least-significant bits within the similarity distance could be considered to be computationally useful, while the remaining high-order bits would not be susceptible to soft errors. The characterized value similarity property can be exploited to speculate an abnormal behavior due to soft errors during the program execution.

Traditional value prediction has a difficulty to predict data values when instructions do not produce strong value patterns like constants or strides. However, although the computed results do not exhibit a predictable value pattern, their similarity distance is likely to be small and stable over time according to our measurement. For example, given instances with result values of 0, 30, 3, 1, 55, in sequence, if the next instance produces an extremely large result such as 100000000000055, it would indicate an abnormal behavior due to a sudden increase of its similarity distance. Therefore, such an abrupt deviation would hint a high possibility of soft error. Similarity distance also captures a spatial locality, which refers to the fact that memory operations tend to access data in a fixed memory region and generate a certain address sequence within the same heap space. Therefore, an out-of-range address would indicate a likely-to-be soft error. Even for instructions with a repeating stride pattern, the similarity distance is constrained to the maximum value's width and any result showing a larger distance would signal a potential soft error.

Fig.4 shows misprediction rate of the proposed anomaly speculation, which is measured by

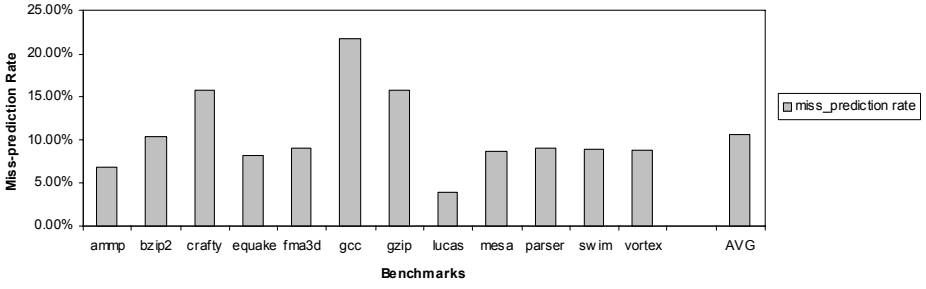


Fig. 4. Accuracy of soft error prediction

the rate of speculating potential errors when there happens no soft errors. Average misprediction rate is computed by 10.6% across the SPEC2000 programs. Therefore, there seems to be an apparent similarity distance pattern, whose variance is similar nine times in a row, then dissimilar once on average. Anomaly speculation is rather similar to branch prediction than value prediction, in that it predicts whether the similarity distance is normal or abnormal based on the previous similarity history. In general, for branches used to form loops, where a branch is taken many times in a row and then not taken once, a simple 1-bit branch predictor mispredicts at twice the rate that the branch is taken. Similarly, in the proposed anomaly prediction, it seems that we should expect that the accuracy of the anomaly predictor would at least match the frequency of similar values.

3.3 Soft Error Prediction for Active Verification Management

Having demonstrated that the similarity distance has a strongly-biased distribution and is easily predictable, we now present one application of value similarity property to design a proactive filter checker based on anomaly speculation. The basic concepts of Active Verification Management (AVM) and anomaly speculation are described in this section. The goal of AVM is to reduce the verification workload and to achieve better fault coverage than non-fault-tolerant processor's, without incurring performance degradation even when resource contention happens. The basic idea is similar to that of a cache hierarchy. Because a small cache is fast and a fast cache is expensive, a cost-effective hierarchical solution can be achieved to obtain both fast speed and large size by exploiting the principle of locality. In the AVM, a simple filter checker gives a hint for each instruction, whether its re-execution should be done or not. For example, instructions un-marked by the filter checker may be directly passed to the 'commit' stage by skipping verification, while marked instructions may proceed to the second-level primary checker for further re-execution.

This paper presents an anomaly-speculation-based filter checker in the AVM design paradigm. Whenever an instruction produces a result that deviates a lot from the previous result, it would hint a potential soft error due to such an abnormal behavior. With the hint given by an anomaly test, only the marked instructions proceed to further verification, otherwise the computed result can be committed to the physical register file. Given a similarity distance n, any error in the least significant n bits cannot be detected by the proposed anomaly speculation, compared to fully-fault-tolerant architectures. However, the majority of data paths producing $(64-n)$ high-order bits of the computed results can be protected, which can significantly reduce Architectural Vulnerability Factor (AVF) of the processor.

The proposed design for anomaly speculation is illustrated in Fig.5. There are two major structures in the anomaly-speculation-based filter checker, which are a speculative similarity distance cache and an anomaly tester. The speculative distance cache dynamically tracks the value similarity of instruction streams and keeps learning its run-time behavior with a saturating counter of confidence score. The similarity distance cache can be accessed by program counter of each instruction. A current execution result is exclusive-OR-ed with the previous result to compute an instantaneous variance. Its similarity distance is obtained by a Leading Zero Counter (LZC) to count the number of consecutive zeros in the high-order bits. Then, an anomaly tester compares the current similarity distance with the speculated one in the similarity distance cache. If the current similarity distance exceeds than the speculated one, the confidence score should be

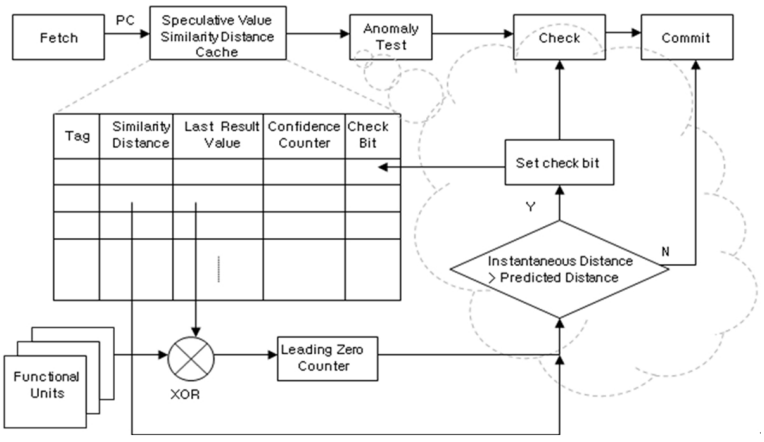


Fig. 5. Anomaly test with speculative similarity distance cache

considered next to check whether it is still learning a nominal scope of value similarity. If the confidence has the maximum score, a potential soft error is detected and a check bit flag is set, otherwise the speculated similarity distance is replaced by the current one and the confidence score is reset by zero. If the current similarity distance is smaller than the speculated one, the confidence score is incremented and no soft error is speculated. The previous result field is then updated by the current one.

4. RESULTS

In order to evaluate the proposed anomaly-speculation-based filter checker, we simulated a dual-core Chip Multi-Processor (CMP) with one complex out-of-order core processor and one simple in-order checker processor. Although the experiment is done only for DIVA-like architecture, the same methodology could be applied to RMT-like one and results presented in this section would be similar. The simulator used in this paper was derived from the M5 Simulator System release 1.1 [14], which we modified to model the proposed AVM with an anomaly-speculation-based filter checker. We collected results for twelve of the SPEC2000 benchmarks suite---six integer and six floating-point application programs. Our experiments were performed using SimpleScalar EIO traces made by SimPoint to choose a 1 billion instruction representative simulation window among 16 billion instructions. For each benchmark, 10 million committed instructions were simulated.

Table 1 shows hardware parameters for the core processor, a 4-way 64-bit superscalar processor. The primary checker processor has a 4-issue checker pipeline, which is an in-order single CPI machine with its own register file, functional units, and L1 cache of the same type as the core processor. The speculative similarity distance cache has 256 entries and is configured as 2-way set-associative. Each entry is composed of several fields, which are a 6-bit speculated similarity distance, a 64-bit last result value, a 2-bit confidence score and a 1-bit check flag.

Fig. 6 (a) shows a dramatic performance slowdown of fully-fault-tolerant processor due to the resource contention, compared with non-fault-tolerant superscalar processor. One advantage of using the proposed anomaly-speculation-based filter checker is that it removes the performance

Table 1. Baseline core processor hardware parameters

Parameter	Value
Fetch Queue Size	32 instructions
Fetch/Decode/Commit Width	4 instructions
Branch Predictor	4K entry BTB and hybrid predictor
Return Address Stack Size	16 entries
Physical Register File	128 INT, 128 FP
Issue Width	4 INT, 4 FP
Issue Queue Size	32 INT, 32 FP
Load-Store Queue / Reorder Buffer	64 /256 entries
INT Functional Units	6 ALU, 2 MUL/DIV
FP Functional Units	4 ALU, 2 MUL/DIV
L1 I-Cache	64K 2-way set-associative, 64B line
L1 D-Cache	64K 2-way set-associative, 64B line
L2 Cache (Shared)	2M 32K set-associative, 64B line

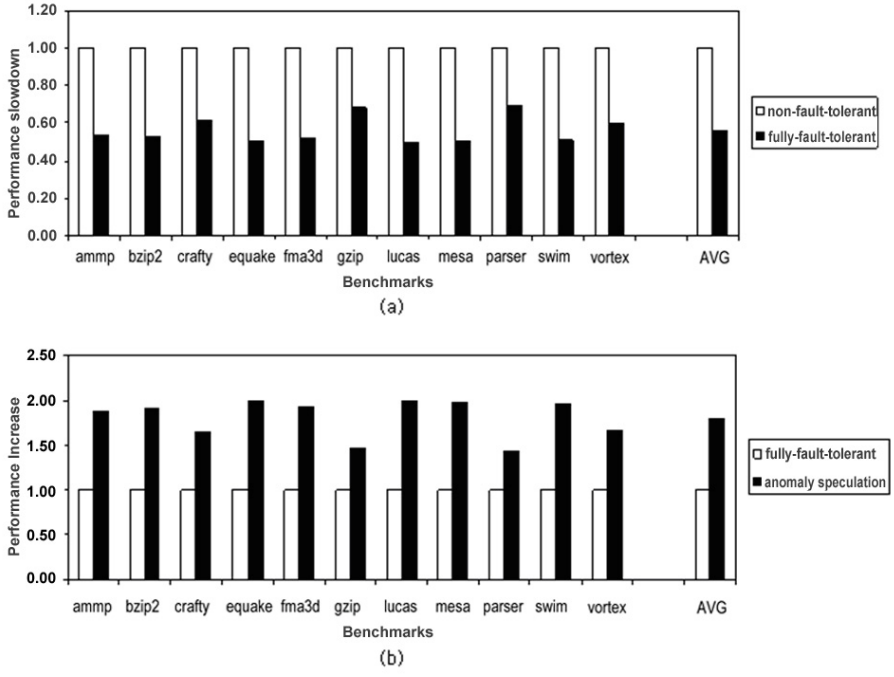


Fig. 6. Relative IPC

degradation due to the resource contention from redundant re-execution, increasing performance relative to fully-fault-tolerant processors. The performance degradation result was already introduced in Section 1. The question is that how well the proposed verification management works. Fig. 6 (b) shows the Instructions Per Cycle (IPC) relative to fully-fault-tolerant DIVA-like architectures. The IPC of proposed anomaly-speculation-based AVM is 1.8 times larger on average

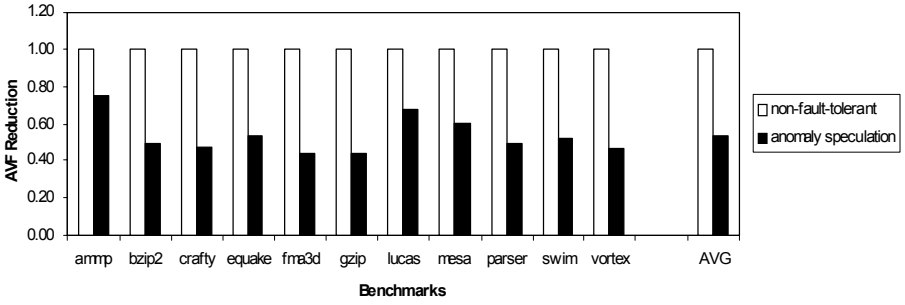


Fig. 7. Relative AVF

across the SPEC2000 programs, than that of fully-fault-tolerant processor. Furthermore, the proposed scheme has the same IPC as that of non-fault-tolerant 4-way superscalar processor. Through bypassing instructions without abnormal results from further re-execution process, the result in Fig. 6 demonstrates that our anomaly-speculation-based scheme works well to completely remove the performance degradation happened at fully-fault-tolerant processors.

Another advantage is that the proposed anomaly-speculation-based filter checker decreases Soft Error Rate (SER) relative to non-fault-tolerant processors. To evaluate the impact on reliability of the processor, we used the same methodology for computing Architectural Vulnerability Factor (AVF), described in [12]. The processor's SER is computed by the product of raw error rate and AVF of the hardware structure. Because raw error rate is generally likely to be proportional to the area of each hardware component, the processor's AVF is obtained by a weighted sum of each component's AVF. Given a floor plan of the Alpha 21364 processor [17], we used the area model to compute the weight.

Any soft error in the least significant bits, within the scope of the similarity distance, cannot be detected by the proposed anomaly-speculation-based filter checker. Compared to fully-fault-tolerant architectures, this refers to decrease fault coverage of the proposed scheme. However, majority of data paths producing high-order bits can be protected from soft errors in its upper bits out of similarity distance. Fig. 7 shows the processor AVF relative to non-fault-tolerant processors. The AVF of anomaly-speculation-based processor is 53% smaller on average across the SPEC2000 programs, than that of our baseline superscalar processor. Therefore, the result in Fig.7 demonstrates that the potential soft error, indicated by the proposed anomaly speculation, can significantly reduce AVF of the processor, resulting in improvement of reliability.

5. CONCLUSION

This paper presented an anomaly-speculation-based fault tolerance, a pro-active verification management technique to mitigate the verification workload of fully-fault-tolerant processors. This was accomplished by exploiting value similarity property. Experimental characterization proves the existence and scope of similar values. Results demonstrate that the proposed scheme increases 1.8 times faster than that of fully-fault-tolerant processor with a minimal impact on

overall soft error rate and reliability of the proposed AVM processor being 53% better than that of non-fault-tolerant processor.

REFERENCES

[1] T. M. Austin, "*DIVA: A reliable substrate for deep submicron microarchitecture design*", *Proceedings of the 32nd International Symposium on Microarchitecture*, November, 1999.

[2] D. Brooks and M. Martonosi, "*Dynamically exploiting narrow width operands to improve processor-power and performance*", *Proceedings of the 5th International Symposium on High Performance Computer Architecture*, January, 1999.

[3] T. M. Cover and J. A. Thomas, "Elements of Information Theory", *John Wiley and Sons*, 1991.

[4] M. A. Gomaa and T. N. Vijaykumar, "*Opportunistic transient fault detection*", *Proceedings of the 32nd International Symposium on Computer Architecture*, June, 2005.

[5] R. Gonzalez, A. Cristal, D. Ortega, A. Veidenbaum and M. Valero, "*A content aware integer register file organization*", *Proceedings of the 31st International Symposium on Computer Architecture*, June, 2004.

[6] J. L. Hennessy and D. A. Patterson, "Computer Architecture: A Quantitative Approach", *Morgan Kaufmann, San Francisco*, CA, 2003.

[7] I. P. Hong, H. Y. Jeong and Y. S. Lee, "Physical register sharing through value similarity detection", *IEICE Transactions on Information and Systems*, E89-D, October, 2006.

[8] J. Hu, S. Wang and S. G. Ziavras, "*In-register duplication: Exploiting narrow-width value for improving register file reliability*", *Proceedings of the 2006 International Conference on Dependable Systems and Networks*, June, 2006.

[9] S. Kumar and A. Aggarwal, "*Reducing resource redundancy for concurrent error detection techniques in high performance microprocessors*", *Proceedings of the 12th International Symposium on High-Performance Computer Architecture*, February, 2006.

[10] M. H. Lipasti and J. P. Shen, "*Exceeding dataflow limit via value prediction*", *Proceedings of the 29th International Symposium on Microarchitecture*, December, 1996.

[11] S. S. Mukherjee, M. Kontz and S. K. Reinhardt, "*Detailed design and evaluation of redundant multithreading alternatives*", *Proceedings of the 29th International Symposium on Computer Architecture*, June, 2002.

[12] S. S. Mukherjee, C. Weaver, J. Emer, S. K. Reinhardt and T. M. Austin, "*A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor*", *Proceedings of the 36th International Symposium on Microarchitecture*, December, 2003.

[13] J. Ray, J. C. Hoe and B. Falsafi, "*Dual use of superscalar datapath for transient-fault detection and recovery*", *Proceedings of the 34th International Symposium on Microarchitecture*, December, 2001.

[14] S. K. Reinhardt, "*Using the M5 Simulator*", *ISCA tutorials and workshops*, University of Michigan, June, 2005.

[15] S. K. Reinhardt and S. S. Mukherjee, "*Transient fault detection via simultaneous multithreading*", *Proceedings of the 27th International Symposium on Computer Architecture*, June, 2000.

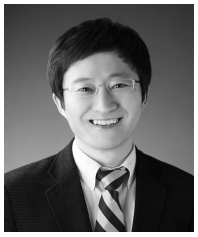
[16] E. Rotenberg, "*AR-SMT: A microarchitectural approach to fault tolerance in microprocessors*", *Proceedings of the 29th International Symposium on Fault-Tolerant Computing*, June, 1999.

[17] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan and D. Tarjan, "*Temparature-aware microarchitecture*", *Proceedings of the 30th International Symposium on Computer Architecture*, June, 2003.

[18] A. Sodani and G. S. Sohi, "*Understanding the differences between value prediction and instruction reuse*", *Proceedings of the 31st International Symposium on Microarchitecture*, December, 1998.

[19] K. Sundaramoorthy, Z. Purser and E. Rotenberg, "*Slipstream processors: Improving both performance and fault tolerance*", *Proceedings of the 33rd International Symposium on Microarchitecture*, December, 2000.

[20] T. N. Vijaykumar, I. Pomeranz and K. Cheng, "*Transient-fault recovery using simultaneous multi-threading*", *Proceedings of the 29th International Symposium on Computer Architecture*, June, 2002.
[21] N. J. Wang and S. J. Patel, "*Restore: Symptom based soft error detection in microprocessors*", *Proceedings of the International Conference on Dependable Systems and Networks*, June, 2005.
[22] J. Yoo and M. Franklin, "Hierarchical Verification for Increasing Performance in Reliable Processors", Journal of Electronic Testing: Theory and Applications, Vol.24, No.1-3, *Springer*, June, 2008.



Joonhyuk Yoo

He received his M.S. and Ph.D. degrees in Computer Engineering from the University of Maryland at College Park in 2007. He had been on the research faculty in the Dual Masters degree program in Embedded Software between Georgia Tech and Korea University from 2008 to 2009. Currently, he is Assistant Professor in the College of Information and Communication Engineering at Daegu University, since then. He also had actively participated in the NASA CREAM flight operation launched in Antarctica and worked at Samsung Electronics as embedded software/hardware engineer. His fields of interest include computer architecture, cyber-physical systems, embedded systems, sensor networks, and operating systems.