Appears in Proceedings of ACM SIGMETRICS/Performance 2006

June 26-30, 2006, Saint Malo, France

Applying Architectural Vulnerability Analysis to Hard Faults in the Microprocessor

Fred A. Bower
IBM, Duke University Department of Computer Science
3039 Cornwallis Rd., B205/F4N
Research Triangle Park, NC 27709 USA
bowerf@us.ibm.com

Derek Hower, Mahmut Yilmaz, Daniel J. Sorin, Sule Ozev
Duke University Department of Electrical and Computer Engineering
Box 90291
Durham, NC 27708 USA
{drh5, my, sorin, sule}@ee.duke.edu

ABSTRACT

In this paper, we present a new metric, Hard-Fault Architectural Vulnerability Factor (H-AVF), to allow designers to more effectively compare alternate hard-fault tolerance schemes. In order to provide intuition on the use of H-AVF as a metric, we evaluate fault-tolerant level-1 data cache and register file implementations using error correcting codes and a fault-tolerant adder using triple-modular redundancy (TMR). For each of the designs, we compute its H-AVF. We then use these H-AVF values in conjunction with other properties of the design, such as die area and power consumption, to provide composite metrics. The derived metrics provide simple, quantitative measures of the cost-effectiveness of the evaluated designs.

Categories and Subject Descriptors
B.8.1 [**Performance and Reliability**]: Reliability, Testing, and Fault Tolerance

C.1.0 [**Processor Architectures**]: General

General Terms
Design, Measurement, Reliability

Keywords
Computer Architecture, Hard-Fault Tolerance, Reliability

1. INTRODUCTION

Microarchitects seeking to implement low-cost hard-fault tolerance are currently stymied by the lack of a simple, quantitative measure with which to analyze their designs. Having such a tool would help microarchitects in understanding where a design is in need of additional hardening. Further, this metric would allow comparison of competing hard-fault tolerant designs.

Both mean time to failure (MTTF) and failures in time (FIT) are used as metrics of component reliability, but both obscure important effects from the fault-tolerance microarchitect. First, these metrics do not consider the inputs to the component, and fault masking is a function of the inputs. Second, these metrics do not consider the utilization of the component.

For analyzing a processor's vulnerability to soft faults, previous work has introduced the widely-embraced architectural vulnerability factor (AVF) metric [2]. AVF is an insightful measure of the vulnerability of a storage structure (e.g., reorder buffer, reservation station, etc.) to soft faults. In the equation for AVF, the occupancy of the structure is divided into bits that affect architecturally correct execution (ACE) and the total of all bits. The metric reflects the fraction of time a storage cell is occupied by an ACE bit. Unfortunately, AVF does not apply to hard faults or to combinational logic.

2. A METRIC FOR HARD-FAULT TOLERANCE

Intuitively, H-AVF for a structure is directly proportional to the probability that a fault in that structure—for all possible fault sites and fault models—causes an instruction to commit erroneous state. For a given software benchmark, the H-AVF of a structure is computed as shown below.

$$H-AVF=\frac{1}{insts}\frac{1}{faultsites}\sum_{\forall fault\,\mathrm{mod}\,els}\sum_{\forall insts} insts_{error}$$

In the equation, we compute the absolute number of instructions that would commit erroneous architectural state for a given instruction's set of inputs (*insts*$_{error}$). This sum is then divided by the absolute number of fault sites for the structure under evaluation (a constant for the structure). Results are averaged over all instructions to provide accurate accounting of masking effects for a given workload. This process is repeated for each of the fault models that are considered. The most common fault models are single-bit-stuck-at-0 and single-bitstuck-at-1. Other fault models exist, however, and may be appropriate for a particular analysis. H-AVF supports the application of multiple fault models, if appropriate. Lower H-AVF values indicate that a given design is less vulnerable. As intuition would suggest, a design with fewer possible fault models, each with an equal effect on architectural correctness, will have a lower H-AVF than a design that is subject to more fault models.

Fault densities are generally considered to be constant for a given process technology. Thus, without area normalization, it is possible for raw H-AVF numbers to provide misleading conclusions. By normalizing the H-AVF to H-AVF per transistor, we can compare designs with equivalent functionality, but wildly differing implementations. Additionally, H-AVF can be composed with cost metrics such as area and power consumption. These normalized, composed metrics provide the microarchitect with the ability to balance hard-fault tolerance and cost requirements in a design space.

3. USING H-AVF TO EVALUATE PROCESSOR SUB-STRUCTURES

Now that we have developed H-AVF as a metric, we will analyze two representative storage structures and one combinational logic structure found in the modern microprocessor, the register file, the L1 data cache, and the 64-bit integer adder. We selected these because they are representative structures within the microprocessor and well-known protection mechanisms exist for all three. Error correcting codes (ECC) are used for the two storage structures and triple-modular redundancy (TMR) is used for the adder.

3.1 Structure Details

In the microprocessor, the register file is a storage array that stores the inputs and outputs of instructions. It is a small, fast structure whose correct operation is critical to the correctness of the processor. For purposes of this study, we model a register file with 126 64-bit entries.

Cache memories are commonly used in microprocessors to hide the main memory access latency and, thus, increase performance. The level of a cache indicates how close it is to the processor, with the closest cache having the smallest level. The L1 data cache is the first cache that the processor accesses when a memory value is needed. For the experiments we run, we model after the Pentium 4, using a 16KB L1 data cache, consisting of 256 64-byte blocks.

The 64-bit integer adder is at the heart of the modern microprocessor. It is used for integer arithmetic operations as well as common activities like calculating memory addresses.

3.2 Determining H-AVF

It is an intractable problem with current compute resources to evaluate all 64-bit inputs to a structure. Even if we could perform such an evaluation, we would not want to assume equal weighting for each input, as certain inputs are much more likely than others, and should thus weight our H-AVF measurement commensurate with their impact for an actual workload.

Thus we use the SPECCPU 2000 benchmark suite to provide inputs for all of our experiments. For the data cache and register file, we simulated the first 10 million instructions from each benchmark, using the sim-cache simulator from the SimpleScalar simulation environment [1]. For the adder, we used SimPoint [3] sampling of 100 million instructions per benchmark simulated with a modified version of the sim-mase simulator [1].

The results in Table 1 are averaged over the entire benchmark suite. To account for the area overhead of the protected implementations, we normalized results by scaling by the ratio of transistors in the protected implementation to the base implementation. For the register file, this factor was ~1.15. For the L1 data cache, this factor was ~1.10. For the TMR adder, this factor was ~3.31. The small magnitudes of the results are due to the large size of the structures with respect to the values we are storing and, in the case of the cache, the low cache miss rates that the benchmark code exhibits.

Table 1. Average H-AVF Results for Unprotected and ECC-Protected Register File and L1 Data Cache and Unprotected and TMR-Protected 64-Bit Integer Adder

Structure	Base H-AVF	Protected H-AVF	Normalized Protected H-AVF
Register File	0.08388	0.00871	0.00958
L1 Data Cache	0.00486	0.00076	0.00084
64-Bit Adder	0.1488	0.0161	0.0533

For all structures, the fault-tolerance increases are significant. The ECC-protected register file is ~8.4 times more hard-fault tolerant than its unprotected counterpart. The ECC-protected L1 data cache is ~5.8 times more hard-fault tolerant than its unprotected counterpart. Finally, the TMR-protected adder is ~2.8 times more hard-fault tolerant.

4. CONCLUSIONS

With the development of H-AVF, we have provided a general-purpose metric for the microarchitect. It can be used first to evaluate whether a particular sub-structure in the microprocessor will benefit from hardening. It can then be used, either by itself or composed with other design metrics, to compare hard-fault tolerant designs to provide a quantitative basis for comparison of competing designs.

5. ACKNOWLEDGMENTS

This work is supported in part by the National Science Foundation under grants CCF-0444516 and CCR-0309164, the National Aeronautics and Space Administration under grant NNG04GQ06G, Intel Corporation, and a Warren Faculty Scholarship. We thank Alvin Lebeck for his insightful comments on this work.

6. REFERENCES

[1] T. Austin, E. Larson, and D. Ernst. SimpleScalar: An Infrastructure for Computer System Modeling. *IEEE Computer*, 35(2):59–67, Feb. 2002.

[2] S. S. Mukherjee, C. Weaver, J. Emer, S. K. Reinhardt, and T. Austin. A Systematic Methodology to Compute the Architectural Vulnerability Factors for a High-Performance Microprocessor. *In Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, Dec. 2003.

[3] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. Automatically Characterizing Large Scale Program Behavior. In *Proceedings of the Tenth International Conference on Architectural Support for Programming Languages and Operating Systems*, Oct. 2002.