An Application-Specific Framework for Detecting Transient Faults in Processors

Srivaths Ravi, Texas Instruments, India
[Email:srivaths.ravi@ti.com]

***Abstract*—This paper proposes an application-specific framework for detecting transient faults in processors, based on the observation that soft errors in some scenarios manifest themselves as aberrations in a program's control flow before resulting in an erroneous output or a system crash. The proposed architecture therefore consists of a hardware based application-specific checker that monitors a program's control flow during its execution, and compares against pre-determined control flow signatures.**

I. EXTENDED SUMMARY

Transient faults or soft errors are a major concern in many systems and system-on-chips (SoCs) [1-4]. Conventional solutions for this problem include use of error correcting codes (ECC) in memories and triple modular redundancy (TMR) in logic [5].

With processor based SoCs being used in many embedded systems today, recent work have focused on the development of techniques that can improve resilience of processors to soft errors that may occur during the course of program execution. For example, DIVA [8] uses the concept of redundancy by implementing a low-overhead checker that replicates the computations in the processor pipeline and dynamically verifies that the results of the computations are unaffected by various factors including soft errors. Many existing solutions are general-purpose in the sense, that the mechanisms provide a generic way to protect the system and all applications that run on it. From a design standpoint, these solutions require intrusive changes to the processor architecture, cause constant performance overheads even for an application that does not warrant this protection, and so on. In this work, we examine the question: *"Can we develop application-specific protection mechanisms that are less intrusive to the processor architecture, and comparatively lightweight (in overheads)?"*

In order to develop application-specific measures, we need to understand ways in which soft errors affect program execution. Various analyses [6,7] have shown that soft errors manifest themselves as aberrations in a program's control flow and/or errors in intermediate data variables. In our work, we restrict our attention to the scenario wherein soft errors manifest themselves as aberrations in program behavior (control flow) before potentially resulting in erroneous output. We argue that if we can identify data-independent invariants of correct program execution, we can potentially design a checker that can monitor the program during the course of execution, "compare" against the invariants, and signal if a deviation from correct behavior is seen. Towards this objective, we identify a program's functional call graph, intra-function control flow graph, and basic block level signatures as potential invariants of the control flow associated with correct program execution. These program properties are invariants that can be identified easily through static program analysis, and can be easily modeled as a part of a hardware checker.

Figure 1 shows a high-level description of the proposed framework. On the hardware front, the architecture consists of a checker that monitors selected signals of the processor, compares against its programmed signatures, and signals if any deviations from expected values are seen. The programmed signatures are extracted from the application through static analysis of the program. The program's function call graph and intra-function basic block control flow graph are modeled as finite state automata (FSA). For each basic block, we also compute the hash or message digest of data-invariant fields of the instruction (such as the instruction opcode).

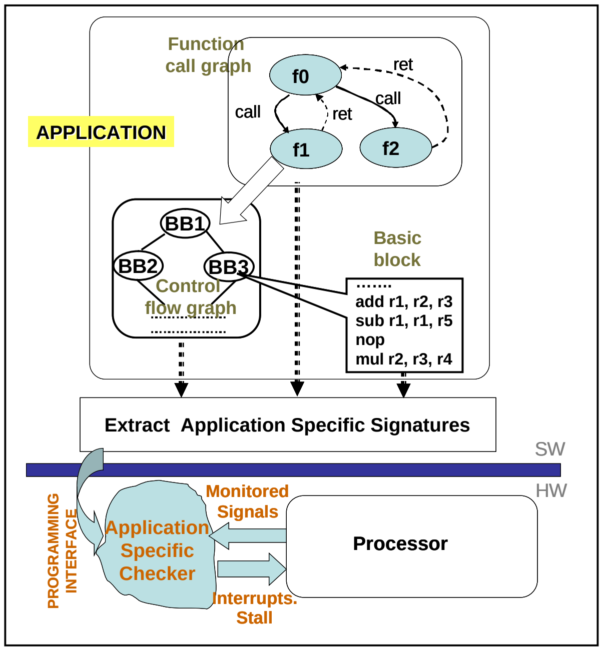

Figure 1. Proposed Framework

The operation of the checker is as follows. At the function level, the checker uses the FSA corresponding to the function call graph in order to enforce caller-callee relationships, and the callee returns. Within a function, the checker enforces the control flow across basic blocks. Thus, any inadvertent jump can be detected. Finally, basic block level signatures catch any changes to the instruction invariants within a basic block. Each level of monitoring provides an opportunity to trade-off between faster detection latencies and better coverage of soft error incidence.

Control flow monitoring is a well-known technique that has been used in the context of security [9], wherein malevolent program behavior triggered in the event of an attack can be detected by monitoring the application's control flow. In our work, we wish to examine the potential and limitations of using this concept for detecting soft errors.

References

[1] J. Ziegler et al, "IBM experiments in soft fails in computer electronics (1978-1994)", IBM J. R & D pp. 3 -18, Vol. 40, No. 1, 1998.
[2] R. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies", IEEE Trans. Device and Materials Reliability, pp: 305-316, Vol.5, Iss.3, Sept. 2005.
[3] S. Mitra et al," Logic soft errors in sub-65nm technologies design and CAD challenges," Proc. DAC, pp. 2-4, June 2005.
[4] S. S. Mukherjee et al, "The Soft Error Problem: An Architectural Perspective." Proc. HPCA, Feb. 2005.
[5] M. Nicolaidis, "Design for soft error mitigation", IEEE Trans. on Device and Materials Reliability, Vol.5, Iss.3, pp. Pages: 405- 418, Sept. 2005.
[6] S. S. Mukherjee et al, "A Systematic Methodology to Compute the Architectural Vulnerability Factors for a High-Performance Microprocessor," Proc. MICRO, Dec. 2003.
[7] C. Weaver et al, "Techniques to Reduce the Soft Errors Rate in a High-Performance Microprocessor," Proc. ISCA, June 2004.
[8] T. Austin, "DIVA: A Reliable Substrate for Deep Submicron Microarchitecture Design," Proc. Micro, Nov. 1999.
[9] D. Arora et al, "Hardware-Assisted Run-Time Monitoring for Secure Program Execution on Embedded Processors," IEEE TVLSI, pp:1295-1308, Vol.14, Iss.12, Dec. 2006.