Reed Electronics Group

MICROPROCESSOR REPORT

www.MPRonline.com

THE INSIDER'S GUIDE TO MICROPROCESSOR HARDWARE

BOOK REVIEW

ARCHITECTURE DESIGN FOR SOFT ERRORS

By Max Baron {5/27/08-01}

Dr. Shubu Mukherjee's book is a welcome surprise: books by architecture leaders in major companies are few and far between. Written from the viewpoint of a working engineer, the book describes sources of soft errors and solutions involving device, logic, and architecture design to reduce the effects of soft errors.

Mukherjee's book makes its appearance now, while many other books and conferences are dedicated to mobile architectures, multiple cores, and massively parallel configurations and the first software attempts to make them work. The book is well-timed; today, just a few presentations on logic design will dedicate a slide or two to generic errors and how to overcome some of them, but very little has been said during the past few years about fault-tolerant architecture. Yet, after the power and temperature that have cut short the reign of frequency, soft errors may turn out to be one of the obstacles all computer architects will have to overcome.

In the introductory overview at the beginning of the first chapter, the author sets the stage for the rest of the book. The first evidence of soft errors due to alpha particles impacting on computer chips was reported 30 years ago by Intel. The company was unable to deliver its chips to AT&T because of a problem traced to contaminated ceramic-chip packaging modules. Since that time, package contamination may have been largely reduced, but chips counting billions of transistors are offering more transistors to be stricken. The new chips also offer more die area to be hit by alpha particles emanating from packaging and by neutrons from the atmosphere.

The book's first chapter is recommended reading for everybody. It presents an overview of the soft error environment and the mechanics of its impact on transistors, providing in the process a simple tutorial on metrics such as mean time to failure (MTTF), mean time between failures (MTBF), reliability, and—for engineers who prefer to use additive probability—failure in time (FIT), expressed in units of error/hours. The chapter does include a brief list of other types of random-seeming errors stemming from misuse or marginal design of hardware and software. Chapter one concludes with a taxonomy of errors, helping to identify their sources and persistence. Read it to become familiar with the acronyms used by soft-error and fault-tolerance experts.

The second chapter models the transistor and logic levels—how they are affected by alpha particles and neutrons and how erroneous changes in logic levels may be mitigated or entirely prevented. It offers information on accelerated alpha-particle and neutron tests. The chapter focuses on circuits such as implementing SRAM, logic gates, registers, and clocking trees, but does not forget to survey some of the solutions that have been published at the device level, such as triple-well, SOI, and increased capacitance.

The third and fourth chapters introduce the concept of architecture vulnerability, determined by the total system failures in time, themselves determined by the failures in time of the system's separate components, and the implementation of the architecture itself. The system architecture is not left without analysis; to determine whether an error will actually become visible to the system user, Mukherjee brings into play the less tangible components as well: the

For More Information

Architecture Design for Soft Errors by Shubu Mukherjee, ISBN 978-0-12-369529-1, hardcover, 360 pp. Morgan Kaufman Publishers, $74.95
Book website: *http://www.elsevier.com/wps/find/bookdescription.cws_home/713719/description*

ISA, the workloads, and, indirectly, the compiler/assembler and the operating system. Standard workloads are used to provide examples of the dependence of architecture vulnerability on the software the system is expected to run.

Continuing the analysis of architecture vulnerability, the focus on memory provided in chapter four covers main memory, caches, RAM arrays, and even content-addressable memories (CAM), such as may be employed in cache tags and in networking hardware.

Probably best known for their use in memory systems, error detection and correction codes in memory and execution units are described in chapter five. This chapter also estimates the benefits of scrubbing (refreshing) memory and the price that must be paid in time and silicon area.

While the previous five chapters were meant to establish a basis of knowledge about soft errors and guide processor and system-memory designers, the next three chapters will be found useful by engineers creating systems mostly employing existing on-chip or on-board components, whether or not these components incorporate schemes for reducing the effects of soft errors. Chapters six, seven, and eight survey useful processor configurations, hardware, software, and algorithms. These, in addition to reducing system vulnerability to alpha particles and neutrons, can also improve system resistance to such faults that may be introduced by noise, marginal voltage, process, temperature, and software.

Chapter six describes and analyzes redundant execution approaches that can detect faults by comparing processors executing the same workload and operating in lockstep at cycle or committed instruction boundaries. Multithreaded single-core sequential execution is also described, showing implementations at different levels of granularity.

Chapter seven offers architectural schemes that can be used by hardware to deal with the faults that have been detected by one or more of the configurations described in chapter six. Depending on the type of fault and the damage done to machine state, the spectrum of solutions described ranges from quick full recovery to a reboot of the processor(s) running the workload. Implementation examples are provided, using existing processor architectures.

Chapter nine completes the book's coverage of soft errors, and the means to correct them, by describing software detection via stamps or signatures attached to instruction streams or software comparisons of committed registers. The software correction of faults covers such implementations as re-execution based on check-pointing in software, timing, and logs, such as practiced in databases, and periodically saving the last-known correct state of the program—a simple strategy already found in some applications.

An easy and interesting read, the book provides more than 100 references to papers and presentations made on the subject. Dr. Shubu Mukherjee writes about the devices, logic, and architectures that have become the components used by designers. His timely book offers equations to the engineer who wants to reduce a system's vulnerability, but the equations are easy to read around when all one needs is to understand the underlying concepts.