DOI:10.1145/1646353.1646376

Technical Perspective
Want to be a Bug Buster?

By Shekhar Y. Borkar

MICROPROCESSOR PERFORMANCE HAS increased exponentially, made possible by increasing transistor performance and doubling the number of transistors every two years to realize complex architectures. These chips with ever increasing complexity are not always fully functional on the first attempt, they need to be debugged quickly, bugs fixed, and reworked. We take this for granted, but did you ever wonder about how it is done?

Painfully, is the short answer!

There are two types of bugs: *functional* or *logic* bugs caused by design errors, and electrical bugs due to circuit marginalities, caused by unfavorable operating conditions such as temperature changes, and voltage drops. Although most of the functional bugs are caught during rigorous design validation and verification, it is virtually impossible to ensure that a design is bug-free before tape-out.

Circuit designers make great efforts to improve margins to avoid electrical bugs, but they too are difficult to avoid, and especially difficult to reproduce since they are manifested by various operating conditions. It is extremely important to find these bugs quickly post-fabrication, and current techniques are far too expensive and time-consuming. The novel technique described in the following paper by Sung-Boem Park and Subhasish Mitra entitled "Post-Silicon Bug Localization for Processors Using IFRA" provides the breakthrough.

When an error is detected it could be caused by one or more such bugs, and you need to identify what caused the error; root causing the error is not that straightforward. The error may be caused by encountering a bug during an instruction execution, or it may be thousands, or even billions, of instruction executions before. You must be a very good detective to diagnose and isolate the bug.

Post-silicon bug localization and isolation is time-consuming and costly because you have to reproduce the failures by returning the hardware to an error-free state, activating the failure-causing stimuli, and then reproduce the same failures—the most difficult task, considering complexities such as asynchronous signals and multiple clock domains.

If you are a detective and want to catch bad guys, then one of the first steps is to acquire the surveillance video from the scene to get the clues. If the surveillance camera is rolling around the clock, you will likely be required to watch the entire tape before coming upon the culprit; very tedious indeed.

Wouldn't it be nice if the camera rolled only when sensing an action taking place? Yes, but it would still require watching every activity recorded by the camera and would still be a time-consuming and wasteful task. Better yet, what if the camera recorded only suspicious activity, capturing all the necessary circumstantial evidence, and not necessarily the culprits? That would be an optimal solution and would make the detective's job a lot easier. This scenario is exactly what IFRA does.

IFRA implements circular buffers, capable of recording traces of instructions executed by the processor, and this process is controlled by failure detectors. These detectors use failure detection in the hardware, such as parity errors as well as soft-triggers that suspect an early symptom of a failure. These triggers stop further recording in the circular buffer, capturing the instruction trace of the suspected part of the instruction sequence for future analysis.

IFRA implements circular buffers, capable of recording traces of instructions executed by the processor, and this process is controlled by failure detectors.

The traditional method of isolating a bug is by comparing the captured instruction trace with a golden trace captured by a trusted simulator. Simulators are notoriously slow and the process is very time-consuming. The authors of this paper propose a novel concept of self-consistency to localize the bug by examining the instruction trace. For example, if an instruction uses a faulty operand then you do not need to know the exact value of the operand. It is sufficient to know that the instruction used a different value than the one that was produced for its use. The authors describe in detail how to diagnose and root cause the bug using the captured instruction trace and self-consistency.

Clearly, this is a very novel approach to post-silicon debug, and I would not be surprised to see it catch on quickly; but it's just a start. This paper describes how to debug a microprocessor, and this technique has great potential to go further and help debug of multicores, memory systems, analog circuits, and even complex SOCs.

Finally, as a critique, you may claim that IFRA adds hardware to the chip that is useful only for debugging. Not really. Transistors are inexpensive, so inexpensive that it is almost like incorporating a small logic analyzer or a tester on the chip itself to aid in debugging, and then turned off; you don't even notice it is there.

Is there better use for a transistor than to help bring products to you quickly and inexpensively? C

Shekhar Y. Borkar is an Intel Fellow and Director of Microprocessor Research, Intel Corp., Hillsboro, OR.