Dynamic Errors: Symptoms and Solutions

Jaume Abella,
Javier Carretero, Pedro Chaparro, Xavier Vera, Antonio González
Intel Barcelona Research Center, Intel Labs-UPC

Burn-in and post-silicon test constraints limit their effectiveness with technology scaling. On the one hand, burn-in must be shorter and less aggressive due to several reasons: (i) burn-in produces high degradation in otherwise well-functioning devices, (ii) its cost increases with technology scaling, and (iii) it may cause thermal runaway due to high leakage and higher activity in the increased gate count. Therefore, many small latent defects may survive burn-in and become large enough during operation due to degradation. On the other hand, the increased gate count requires longer test patterns during post-silicon test because many circuits cannot be tested simultaneously due to power and temperature constraints. Unfortunately, increasing test time raises its cost and therefore, time-constrained tests lose some coverage. As a consequence, some defects that show up only in hard-to-test corner cases may escape testing. Overall, an increased number of actual and latent defects survive burn-in and post-silicon test, thus increasing the likelihood of in-the-field errors.

Increasing error rates pose a challenge for some market segments like desktops, notebooks and servers. Those error rates are more challenging for supercomputers because hundreds or even thousands of processors may work coordinately during some weeks to run a parallel program. Even if the failure rate for one processor is low, any single error in any processor during those long execution periods may jeopardize several millions of hours of error-free computation.

In this talk we analyze technology scaling implications on burn-in and post-silicon test. We show how hard-to-test failures and latent defects escaping burn-in and test may cause intermittent in-the-field errors. We pay special attention to the new challenges affecting multi-core processors such as testing the coherence protocol and inter-core communication interfaces.

Once sources and symptoms of errors have been presented, we move towards existing and potential solutions to timely detect and correct errors. First, schemes based on protecting data such as parity and ECC are presented. Then, we introduce some schemes to detect errors in combinational logic such as re-execution, residue codes, hard-signatures, and ad-hoc logical checking for some circuits. We illustrate the importance of identifying the source of any error to isolate faulty hardware and prevent further errors, as well as the need for checkpoints to recover from errors (e.g., some errors updating unique data, errors in parallel programs, etc.).

Finally, we conclude this talk pointing to some solutions based on verifying semantics of programs such as setting up pre-conditions, post-conditions and invariants in the code, as well as identifying error tolerant data that do not need to be checked for errors.