

Debug and Diagnosis in the Age of System-on-a-Chip

Robert Molyneux
Sun Microsystems Inc.
5300 Riata Park Court
Austin, Texas 78727

The current scale of integration is the greatest driving force for needing better and faster, debug capabilities. We are truly building systems on a chip, board level interfaces which could previously be monitored and debugged with logic analyzers are now buried on silicon. We need to consider burying the logic analyzers along with the interfaces.

The debug task is sometimes made more difficult in today's environment with the increased reuse of internal IP, the use of externally supplied IP and remote foundries. Internal reuse of IP may not at first seem like a complicating factor but consider the likely fact that few if any of the original designers of that block are still with the company and you are intending to use it in a design that they had not even thought of. With increased IP reuse our marketing folks, in their infinite wisdom, are tempted to offer more parts in the market, many just a small customization of the base model. "Hey just cut the cache in half, slap on a 10/100 IP block from TI, cut the cost by 30% and we've opened up a whole new market!" Right! We are not plug and play in the microprocessor component domain yet. We are executing these design customizations but at this time the extended debug phase may more than compensate for the design time saved due to IP purchase or reuse.

Time to market is perhaps the single most significant concern in today's VLSI design projects. Early silicon in the lab is the first place where all the pieces of the puzzle come together: new chip design, new test vectors, next generation fab technology, new board design, new version of the operating system. When the system hangs or crashes or fails to boot there are usually lots of suspects as well as a bit of finger pointing. Different people with differing skill sets must be brought together to isolate and diagnose the problem: logic designers, circuit designers, DFT and test engineers, board designers, OS gurus, fib wizards...the list goes on. With all these complicating factors and the emphasis on time to market, the need has never been greater for better, faster, debug and diagnosis tools, techniques, and flows. Tools and flows that will connect and enable this diverse collection of people needed to debug our chips.

Failure isolation is usually the most time consuming step in the debug process and it is often done under the anxious scrutiny of every director and VP in the hardware division each wanting a daily progress report, if not twice daily. If you begin failure isolation with a failing test vector then you are truly blessed. You know it's the processor and you know exactly how to stimulate the failure. Standard

ATPG failure analysis does not account for intermittent failures or wire interactions and my experience is that when all vectors are supplied to the failure analysis tool the candidate list comes up empty. Some serious trimming of the list of passing vectors is needed before any gates are identified as possible cause for failure; some improvement is needed here.

The more difficult and unfortunately more common situation is when the failure is experienced when the system hangs or crashes and no one is sure exactly where in the code the problem occurs or whether it's the application, the OS, the new GigE controller, the processor, the board. Or maybe it's this processor on this board, let's swap in another processor or try another board, stand on one leg, spin counterclockwise and throw salt over your left shoulder while invoking the secret incantation...*confidential stuff deleted*. The components are sent back to test and later returned NDF, this should be a big wake up call to the DFT engineers, "We need to be using new fault models!"

A few helpful features when it comes to isolating microprocessor/SOC based failures: on-chip trace buffers, logic analyzer-like functions on major internal interfaces, clock control with a robust set of debug triggers, full scan access to all flip-flops and memory arrays. The trace buffer acts like the "black box" on an airplane telling you what happened just before the crash. Qualifying when entries are made allows you to make optimal use of limited storage. Once you've gotten into the neighborhood of the failure you would like to have access to all the state elements in your processor, scan is nice for that. Some sort of scan based access to your embedded memory arrays is desirable. Code based array access may be the bug you are trying to isolate. Clock skew and hold time violations are tricky to diagnose. Provide the part with a conservative clocking mode which sacrifices frequency for timing margin, it can be a great help in quickly determining that timing margins are the cause of the failure.

As we integrate more of the system onto the silicon it is critical that we also integrate adequate on-chip debug capabilities. Logic analyzers, bus monitors, trace buffers, robust debug triggers, reliable scan and array access are all important tools in the design toolkit for today's and tomorrow's processors. We need tools, flows, and design environments that bring the universe of designers and test engineers together. It may be possible to define a standard set of debug features and interfaces that tool developers could use to provide an over arching design environment that better facilitates the debug and diagnosis effort.