

Getting it right

CHUCK MOORE

crmoore@cs.utexas.edu

..... When it comes to functional verification, the short, simple goal of “getting it right” stands in stark contrast to the mountain of obstacles that stand between the design team and an appropriately verified microprocessor design. That’s why project leaders must think carefully about the verification strategies they will employ within a project and how they communicate those strategies to their design team.

A fundamental passion

Best-of-breed verification results come from a fundamental passion about the challenges and realities of today’s designs. Modern microprocessors are among the most complex devices ever created by mankind, and the verification challenge is enormous. Successful leaders must establish a strong discipline and commitment within the team to overcome these challenges.

The most successful teams in the industry seem to demonstrate several common philosophies about verification. These philosophies help to form strategies and overall structure for increasing the verification effort’s efficiency and effectiveness. I believe it is important for designers and design teams to recognize, share, and build on such philosophies as a way of improving the quality of digital design in general.

Here are some ideas that I have detected and found to be the most helpful in projects that I have led.

The cost of finding functional problems increases, in dramatic steps, as the design and development proceeds.

Problems found after RTL freeze cause rework in all back-end physical design efforts. Problems found after tape-out cause new masks and new manufacturing. Problems found in the field can cause recalls and/or broad deployment of patches. In each of these three scenarios, the cost of managing late-breaking problems increases by an order of magnitude.

As a result, an appropriate investment in verification technology is very cost effective. In particular, it is important to optimize the verification process to find bugs as early as possible. Verification cannot be an afterthought.

Fully deterministic verification plans are not possible.

Quantitatively, the problem is on the order of 2^N , where N is the sum of all of the state and inputs into the system—that is, an astronomically large number. As a result, the number of deterministic tests (those targeting a specific combination of events) is far too large to handle exhaustively. Instead, the verification attack must combine a sensible deterministic test plan with a broad range of other techniques, such as low-level protocol checkers, random-test generation, directed random-test generation, random tests seeded from coverage-based feedback, and sliding window tests, to name just a few. Each of these methods offers additional

coverage, but they also carry an associated infrastructure (and resource) cost. It is important to factor these costs into the overall project plan so that team members can apply the custom and creative approaches appropriate for their particular designs.

For modern designs, 100 percent coverage is not possible.

Although this certainly isn’t good news, this point is critically important to recognize and address. To help manage the problem, the team must apply rational thinking to eliminate cases, categorize relationships, and organize the verification attack to be as broad as possible. Even with the best of intentions, the verification process will overlook important cases, which can surface as bugs later on. To mitigate the risk associated with this, it is important that the architecture and the design employ mechanisms to help work around latent problems. The design team and the verification team must work closely together to identify the areas of the design that are most risky, along with defining mechanisms to work around potential escapes.

Finding the most difficult (and costly) bugs requires a systematic and hierarchical attack.

Verification of basic functionality is only a starting point; the difficult bugs hide beneath the basic functionality. Uncovering these bugs requires extraordinary

attention to detail in the corner cases and/or seemingly unrelated system activity. The abstraction and stimulation of this sort of system activity must occur at all levels of the design—mechanism, block, unit, chip, and full system—to find these difficult bugs.

At the mechanism-level, it should be the responsibility of designers themselves to assure basic mechanical operation as early as possible. At the next level, after combining several mechanisms into a functional block, the goal should be to do nearly exhaustive simulation of the internal function and to push the interface protocols through their full range. At the unit-level, the goal should be to create as much interaction between the component blocks as possible. Although exhaustive sequencing is typically not possible, many verification schemes work to identify transaction types and then aggressively slide combinations of potentially related interactions past one another to stress these interfaces. In chip-level simulation, the goal expands to include operations that involve multiple functional units and/or interaction with the global chip infrastructure.

To achieve all of this requires dedicated people with special skills at each level of the design. In microprocessor design, experience indicates that an effective verification program requires a 2:1 (or even 3:1) ratio of verification personnel to logic design personnel.

Every bug found is a symptom, and should be treated it as such.

At the heart of every bug, there is a reason for its introduction into the design and a reason for its identification at that particular point in the verification attack. It is important to understand these reasons, and use them as seeds in exploring the possibility (or probability) of more problems along the same line. In the case of a late-breaking bug, it is particularly important to determine why it took so long to find—and then go back and adjust the verification environment for that part of the design. The most robust verification environments are dynamic and iterative.

A low bug rate only means that it is time to get even more creative on the verification attack.

Although it is always tempting to interpret a low bug rate as a statement of design quality, the appropriate interpretation is usually that the verification attack is becoming soft. The verification team should take pride in finding bugs, and should recognize that there are *always* more of them.

The functional verification process extends into the post-silicon phase of design.

The most difficult-to-find bugs don't come to light in the design's simulation models. Rather, you find them in the actual hardware (and then typically recreate them in the simulation models to aid in debugging and regression testing). Actual hardware, although limited in the granularity of control and stimulus, is actually the fastest "simulator" possible. It offers unique opportunities for driving the design into corners and exploring the associated neighborhood by leveraging the raw speed of execution. It encourages a level of verification attack based on the power of higher-level programming inconceivable in a simulation environment because of the number of cycles required.

The best verification plans go beyond simply stumbling on problems while trying to bring up classic software on the hardware. They recognize the opportunity of developing special software-based exercisers to run on the hardware to aggressively stress the design in its most sensitive areas.

Verification considerations are a fundamental constraint on the architecture and design process.

Architects and designers should be keenly aware of the verification implications for any design tradeoffs they make. Verification representatives should participate in all technical design reviews. An unverifiable design—or even one that takes too long to verify—is worthless (in fact, it can be a huge liability). In the near future, it will be common for designers to

actually add hardware to help aid the verification efforts (*design for verification*).

Excellence in verification technology is an outstanding career opportunity.

The philosophies, technologies, techniques, and skills required for quality verification are a key asset. Experienced verification personnel are one of the most sought-after skill groups in industry today. Teams should resist the temptation to think of verification as a training ground for future designers, and instead recognize that developing these skills is non-trivial and can represent a significant competitive advantage.

Making it stick

Stating these philosophies is the easy part; making the ideas stick and actually implementing them is difficult. It is important for leaders to demonstrate respect for the scope of the problem, and a passion for attacking it at all levels of the design progression. When everyone on the team has a solid and common set of philosophies to refer to, the process of managing design complexity into a successful product becomes a tractable goal.

Charles Moore is a senior fellow at Advanced Micro Devices where he is responsible for the architecture of a next-generation microprocessor. Previously, he was the chief engineer on IBM's Power4 and PowerPC 601 microprocessors, and most recently, he was a senior research fellow at The University of Texas at Austin.