Bridging Pre-Silicon Verification and Post-Silicon Validation

Amir Nahir, Avi Ziv (Organizers)
IBM Haifa Research Lab
{nahir, aziv}@il.ibm.com
Haifa, Israel

Rajesh Galivanche (Organizer)
Intel
rajesh.galivanche@intel.com
Santa Clara, CA

Alan Hu (Chair)
University of British Columbia
ajh@cs.ubc.ca
Vancouver, BC

Miron Abramovici
Tiger's Lair
miron.abramovici@tigers-lair.com
Vienna, VA

Bob Bentley
Intel
bob.bentley@intel.com
Hillsboro, Or

Valeria Bertacco
University of Michigan
valeria@umich.edu
Ann Arbor, MI

Albert Camilleri
Qualcomm
albert@qualcomm.com
San Diego, CA

Harry Foster
Mentor Graphics
harry_foster@mentor.com
Plano, TX

Shakti Kapoor
IBM
skapoor@us.ibm.com
Austin, TX

PANEL SUMMARY

Post-silicon validation is a necessary step in a design's verification process. Pre-silicon techniques such as simulation and emulation are limited in scope and volume as compared to what can be achieved on the silicon itself. Some parts of the verification, such as full-system functional verification, cannot be practically covered with current pre-silicon technologies.

This panel brings together experts from industry, academia, and EDA to review the differences and similarities between pre- and post-silicon, discuss how the fundamental aspects of verification are affected by these differences, and explore how the gaps between the two worlds can be bridged.

Categories and Subject Descriptors

B [Hardware]; B.6 [Logic Design]; B.6.3 [Design Aids];

General Terms

Design, Verification.

Keywords

Verification, Validation. Pre-Silicon, Post-Silicon

PANELIST VIEWPOINTS

Miron Abramovici: The fundamental building blocks in bridging between pre-silicon verification and post-silicon validation are reconfigurable engines introduced into the design. These engines extend the powerful simulation capabilities, such as transaction-level modeling and assertion-based verification, to the post-silicon domain. Using on-chip logic analysis and signal capture provides the ability to capture a suspected behavior in silicon and analyze it in simulation. Crossing the bridge in the other direction, on-chip dedicated (yet programmable) logic can be configured to apply stimuli extracted from simulation to the real block on the chip. Another scenario is to take assertions used to verify the behavior of a block in simulation and implement them in silicon using dynamically reconfigurable instrumentation.

Bob Bentley: Pre-silicon verification and post-silicon validation environments have fundamentally different characteristics. Pre-silicon simulation is highly controllable, repeatable, observable and hence easy to debug. On the other hand it is maddeningly slow, which limits the depth and breadth of coverage that can be achieved even with farms of high-performance compute servers. Post-silicon validation is almost the inverse - extremely fast, but with very coarse controllability, problematic reproducibility, very limited observability, and is very difficult to debug. The lack of controllability and observability makes it difficult to get meaningful coverage data with which to implement a coverage-directed validation methodology. Post-silicon validation also suffers from platform stability, electrical and circuit issues which exacerbate the problem.

To date, efforts to bridge the gap between these two worlds have been focused on various forms of simulation acceleration such as emulation. These approaches certainly have value, but they are typically several orders of magnitude slower than real silicon and require a non-trivial amount of unique engineering work from an already-stretched design team. They are also expensive, and have a limited window of opportunity - typically from the late stages of design when bugs start to become hard to find until the arrival of first silicon.

An alternate approach to bridging the gap is to try and give silicon some of the characteristics of the simulation environment. This can include silicon hooks to provide better observability and controllability, and on-die debug mechanisms such as hardware break points and trace buffers. An often-overlooked aspect is the need for survivability and repair mechanisms to allow post-silicon validation to continue to "peel the onion" even in the presence of known bugs.

Valeria Bertacco: Effective post-silicon validation requires one to adapt and shift methodologies from pre-silicon verification so as to leverage the advantages of the silicon prototype (high performance) while minimizing its disadvantages (limited

DAC'10, June 13-18, 2010, Anaheim, California, USA.
ACM 978-1-4503-0002-5/10/06.

observability, diagnosis, and repair). For instance, test-benches can be more verbose because they run much faster; however, it is important to have a set of checking strategies in place so that bugs are detected even without internal node observability. Because mainstream repair would require a system re-spin, it is necessary to have solutions in place to bypass known bugs at runtime. New solutions must be developed to support automatic diagnosis as a scale far beyond what is common in pre-silicon verification. On the other hand, pre-silicon diagnosis can complement this effort, so that post-silicon diagnosis is only required to narrow down an issue to the model-level, paired with compact traces that can be replayed in simulation.

Albert Camilleri: Pre-silicon verification and post-silicon validation are merely phases within which to demonstrate quality and attributes of a product. One is tasked with mapping requirements to an architecture, refine the architecture to an implementation, and ultimately productize a design. Along every step one needs to verify the correctness and validate that the product built meets the desired requirements. The challenge is to achieve the highest quality, as efficiently as possible.

Yet rarely do we see the overall challenge of verification and validation addressed as a unified and holistic test-plan, where the primary emphasis is on what needs to be tested, rather than how or when. The earlier we discover and address issues in our design, the more cost effective the solution. Today, technologies such as formal verification, simulation acceleration and FPGA-based emulation enable us to address much larger portions of our test-plans than we had traditionally relied on via regular simulation, and these technologies have enabled us to produce higher quality silicon, first time around. No matter how much of the problem we move forward to pre-silicon, however, we will always be faced with challenges in post-silicon, and to complete the remainder of the test-plan to achieve the required quality (such as long tests, and running real applications). To truly approach the overall verification and validation challenge holistically, we need a more seamless verification infrastructure to carry on from pre-silicon to post-silicon, but unfortunately this is where current methodology often breaks apart. The lack of debug capability and the absence of coverage measurement in silicon often lead to brute force ways of testing and validation in place of efficient execution. Design for debug and a common infrastructure for pre- and post-silicon testing are two areas of technology which would stand to improve post-silicon validation completion. The nature of post-silicon testing is very different to pre-silicon, but the need for debug and to measure progress towards quality remains as important, so innovative ways to adapt debug and coverage to post-silicon are crucial.

Harry Foster: At 65nm, we witnessed the post-silicon validation effort often consuming more than 50% of an SoC's overall design effort, as measured in cost, and the problem grows as the industry continues to move to even smaller geometries. Unlike pre-silicon verification, which has historically (and conveniently) partitioned the verification effort into separate concerns (such as, electrical, functional, performance, and software), identifying failures in post-silicon requires skills spanning multiple validation disciplines. Furthermore, the process of post-silicon validation is hindered by both poor observability and poor controllability. To address today's escalating validation effort requires establishing a stronger link between the pre-silicon verification and post-silicon validation processes. Certainly assertions are one technique that can bridge pre- and post-silicon by providing improved observability on critical functionality. However, the improvements obtained by silicon assertions are only as effective as the quality of their pre-silicon form. Realistically, only a small set of critical assertions could be shared between the pre- and post-silicon processes. What is needed is a means to instrument into the silicon observability in a reconfigurable fashion, thus allowing the post-silicon validation engineer to shift focus on specific areas of concern. Concerning test generation, pre-silicon test provides insufficient coverage to stress the post-silicon design. Hence, what is needed is a means to capture post-silicon test associated with a failure in an abstract form that can be demonstrated on a pre-silicon model. Finally, concerning triage and error isolation, both the pre-silicon verification and post-silicon validation processes could benefit from automatic techniques that identify a set of candidate causes behind the detected failure.

Shakti Kapoor: It is important to take the pre-silicon methods and algorithms to post-silicon validation. Pre-silicon tools allow us to measure our coverage unlike most post-silicon tools. Furthermore most pre-silicon tools give better control over the corners we want to hit. This control is not there in post-silicon tools. Presently it is very hard to measure our coverage in post-silicon validation. Obviously we have been fairly successful there, but we do not have a clear way of explaining this success.

On the other hand, I feel pre-silicon need some work before they can be applied to post-silicon. Generally, pre-silicon methods and algorithms are very slow by virtue of their rigor. Most of the speed up of the hardware gets soaked by generation path length or results checking. Moreover, these methods are based on the assumption that we know all the paths we want to cover in the hardware, which we do not. We need to come up with a hybrid method of pre- and post-silicon to address these two problems.

Post-silicon testing depends mainly on instruction streams to stimulate the hardware and generally lacks external methods to stimulate micro-architectural events. This is true especially in a multi-threaded. We need to invent and implement the stimuli which are easy in pre-silicon for the post-silicon testing also. Most of the bugs found in the hardware are dependent on these micro-architectural events. Most of these events are more easily created in the hardware by extremely random tools compared to relatively well-behaved architectural tools. Pre-silicon tools are biased to being architectural tool and have limited in scope for generation of random micro-architectural events. Extremely random post-silicon tools accomplish the creation of these events by virtue of their extremely low cost generation or extremely low cost results checking. This allows them to use the hardware mainly for "deliberate testing" compared to architectural tools and also getting micro-architectural events which are sometimes difficult to think of.

On other side of the equation, one can argue that we need to enhance our post-silicon tools to run in pre-silicon more efficiently. Presently, post-silicon tools are dependent on getting extremely long run time on the hardware. If they can be optimized/tuned to be efficient in "deliberate testing" in view of the limited number of cycles of pre-silicon then perhaps these tools can contribute far more to the pre-silicon bugs found before they reach the hardware. We have made lot progress in this area but clearly there is lot more which can be done here.