Building A Verification Test Plan: Trading Brute Force For Finesse

Organizer:
Francine Bacchini
Francine Bacchini, Inc., San Jose, CA
+1.408.839.8153
francine.b@comcast.net

Chair:
Sharad Malik
Princeton University, Princeton, NJ
N/A
sharad@princeton.edu

Janick Bergeron
Synopsys
Ottawa, Canada

Harry Foster
Mentor Graphics
Dallas, TX

Andrew Piziali
Cadence Design Systems
Parker, TX

Raj Shekher Mitra
Texas Instruments
Bangalore, India

Catherine Ahlschlager
Sun Microsystems
Sunnyvale, CA

Doron Stein
Cisco Systems, Inc.
Netanya, Israel

Categories and Subject Descriptors
B.0 GENERAL, B.6.3 Hardware, LOGIC DESIGN, Design Aids, Verification

General Terms: Design, Verification

Keywords: Design Verification, Verification Test Plan, Functional Simulation, Formal Verification, Coverage

PANEL SUMMARY

The increasing complexity of today's designs has only served to further intensify the pain of functional verification. Any strategy for success here must include a verification test plan – one that trades brute force with finesse. In so doing, not only is the pain reduced, but additional benefits are quickly derived, such as greater predictability, more aggressive innovation and late stage spec changes which can be made with confidence.

Certainly, verification teams have a need to know, while a design is being verified, if their efforts are progressing according to schedule. In order to declare verification done, the team needs to be able to effectively assess the risk of bug escapes. Teams typically use a verification coverage plan to help them address these issues, but increasing complexity and IP integration place rising demands upon verification teams.

In the past, test plans described the scenarios under which you needed to test a device. With the changes imposed by emerging technologies, test plans have needed to expand and evolve to adapt. While coverage is at the core of functional verification, it is a topic of much discussion and debate.

What are the different coverage metrics being used today, and do these metrics properly address the functional verification challenge? How do you begin to integrate multiple verification processes into a single test plan? This panel will attempt to address these important questions.

As chip complexity and IP integration have increased, companies have begun testing at higher levels of abstraction, moving more and more to ESL techniques to better understand the overall system functionality. Industry leading companies have begun to embrace the use of formal verification, emulation and acceleration to drive greater verification success in the drive for greater quality and schedule predictability. But, what value do these new verification processes bring, and what usage and integration challenges do they pose?

In this panel users and suppliers debate the optimal mix of formal, simulation, hardware acceleration and emulation, examining ways to ensure new features aren't dropped pre-tapeout from 'inadequate verification'.

It's no longer just a test plan discussion, but an examination of how test plans are going to need to change given emerging technologies.

PANELISTS VIEWPOINTS

Catherine Ahlschlager: "An effective test plan encompasses a detailed description of the hierarchical verification methodology at both the unit and multiple unit level, as well as at the full chip level. For instance, it is important to consider at what verification phase directed tests versus random tests will be applied. Or, when to stop investing effort on building a stand alone test environment (which can provide greater coverage) and, instead, migrate to full chip level tests (which delivers a more comprehensive understanding of the state of the chip). A good test plan addresses many questions, such as what tools can be best leveraged for stand alone and full chip, and for what type of tests. Creation of the expected results, along with the

DAC 2006, July 24-28, 2006, San Francisco, California, USA.
ACM 1-59593-381-6/06/0007.

self-checking mechanism, should also be detailed to improve automation and to drive the highest return on performance. In addition, with each verification phase, testbench deliverables (stub model of the interface blocks), dependencies (RTL availability), milestones (tests to be completed and written) and assumptions all need to be specified and understood. Finally, upon completion of the test plan reviewed by the design and verification team, a matrix is created to track test coverage and used to measure each project integration. It is also important to know when and how to apply technologies, such as emulation and formal, to leverage their key strengths, avoid any weaknesses and achieve high design quality for your verification effort."

Janick Bergeron: **"**Verification plans today don't enable predictable, plan-driven verification. They are usually passive - an ad-hoc list of checklist items that are manually converted to tests. They cannot drive the day-to-day creation/execution of tests. Nor do they enable any automation for aggregating data from multiple sources, tracking the progress over time, and measurement against goals. The status of a project is left to the subjective judgment of individuals interpreting all the raw data. Automation of this entire process is a key requirement for more predictable verification. The verification plan should allow for the specification of features and sub-features in a hierarchy, each with associated measurable metrics that could be compared to time-based goals. The plan should come with automation to invoke the test suites associated with each feature, as well as to extract the metric data from the run, annotate it back on the plan, and take user-specified measures if a metric is not in line with its goals. The plan infrastructure should allow for aggregation of data over time, as well as "slice and dice" analysis of this data."

Harry Foster: **"**Successful verification is not ad-hoc in nature. On the contrary, experience repeatedly demonstrates that success depends on methodical verification planning combined with systematic verification processes. The key to success is the verification test plan. While the process of simulation-based testplanning is generally well understood in a traditional verification environment, the process of formal-based testplanning is not well understood due to the lack of industry formal experience and published formal-based testplanning guidelines. As verification teams consider the option of integrating functional formal verification tools into their flow, the lack of a formal-based test plan often results in ad-hoc verification results, with a questionable return on investment. Hence, new ways of thinking about today's verification testplanning process are required."

Raj Mitra: "How to meet a verification schedule? There are two ways to meet a verification schedule - to meet a quality/coverage target or to timeout after the bug-curve saturates. When only random-directed simulation is used for today's complex modules, the first option may take months and is not a viable option. The second option has the risk of not detecting all bugs and going for a respin. Formal Verification seems to be an emerging alternative to bridge this gap, and achieve the quality targets within the defined schedules. The flip side is that formal verification is not suitable for all classes of designs, so a judicious mix of formal verification and simulation needs to be applied. This process has to use uniform coverage metrics for both Formal and simulation, so that there is a final uniform report which can be used for signing off. Secondly, there is an inherent unpredictability of the formal verification process, and this has to be factored in when putting it to mainstream usage. Efficient estimation tools have to be used that indicate which modules are candidates for which verification techniques. A complementary approach to reducing verification time is to use higher level models. This reduces the quantity of verification and also the simulation run times, by using abstractions where needed. At a chip level, verification time can be reduced by 'auto-generation' techniques - i.e. automatically synthesizing the IP-hookup logics and the associated testbenches. The former increases the confidence in quality of the design and thereby reduces the quantity of simulation, and the latter reduces the setup time. Formal verification can be used to statically verify the hookup logic, and thus reduce simulations to a great extent. Finally, emulation can also be used to accelerate the verification schedules, especially the validation with real applications. Constraints developed for the IP can be reused as assertions at this stage to uncover system-level bugs."

Andrew Piziali: "Coverage in the broadest sense is responsible for measuring verification progress across a plethora of metrics and aiding the engineer in assessing their location relative to design completion. The map to be referenced must be created by the design team up front so that they not only know their starting position - specification but no implementation - but they also know where they are going: fully functional first silicon. The metrics of the map must be chosen: RTL written, software code written, features, properties, assertion count, simulation count, failure rate and coverage closure rate. This map is the verification plan, an executable natural language document that defines the scope of the verification problem and its solution. The scope of the problem is defined by coverage models: implicit and explicit. The solution to the verification problem is described by the methodology employed to achieve full coverage: dynamic and static verification. Simulation, acceleration and emulation contribute to coverage closure through RTL execution. Formal verification contributes to coverage closure through proven properties. By annotating the verification plan with these progress metrics, it becomes a live, executable document able to direct the design team to their goal."

Doron Stein: "The task of building a test plan involves staffing, declaring goals, setting a timeline, applying measurements, monitoring convergence, etc. A methodological approach to the test plan development process can reduce the pain of its completion, as well as increase the efficiency and quality of results. When planning, it is important not only to do the proper verification work, but it should also be "handed" to the proper methods (simulation or formal). And, in addition, the coverage goals should be set along with the timeline for achieving it. As absurd as it may sound, there is an important need to declare goals relative to time and to tune this relationship as the project progresses. I will try to analyze the methodology needed for coupling time to the progress of the test plan work and the way it may relate to the decomposition of the verification work into verification methods."