

Achilles Test Systems, Inc.

*No one is invincible.
Finding and managing flaws is the secret to survival.*

Managing Information Silos: How to Reliably Track Progress in a Multi-tool/Multi-discipline Environment

Chris Kappler
Achilles Test Systems, Inc.
chris.kappler@achillestest.com

Greg Goss
Achilles Test Systems, Inc.
greg.goss@achillestest.com

For a project to succeed, each sub-team and team member must be making steady progress. With each job role and each engineering tool, specific types of progress make sense. Others do not. Tracking many types of progress together avoids surprises and mis-steps in a project.

We will cover the feasibility of tracking a diverse portfolio of progress metrics, side-by-side. Ideally, multi-metric progress tracking is independent of any specific tool chain. We employ a system of up-to-date progress-tracking pages that are visible to all team members via a web browser.

Information Silos

An “information silo” is a system or team where communication is focused on “vertical” or specialized concerns. In contrast, the term can also be used to highlight problems of cross-team communication.

Engineers have specialties and are often valued more for technical depth than breadth. As a result, the silo effect is always a concern. As projects and companies grow, teams are naturally organized according to technical disciplines. Beyond skill-sets, large designs and their engineering teams are segmented according to SoC IP blocks or chip boundaries. The org-chart then illustrates the lines of vertical communication that support this segmentation.

Our methodology is to divorce the concept of progress tracking from any particular tool, metric, group, or milestone. This is done for two reasons. The first is to increase the reliability of progress tracking in each silo by adding a wider diversity of metrics. The second is to implement progress tracking in a way that brings the focus back to the whole team and away from the verification-only focus that has dominated for several years.

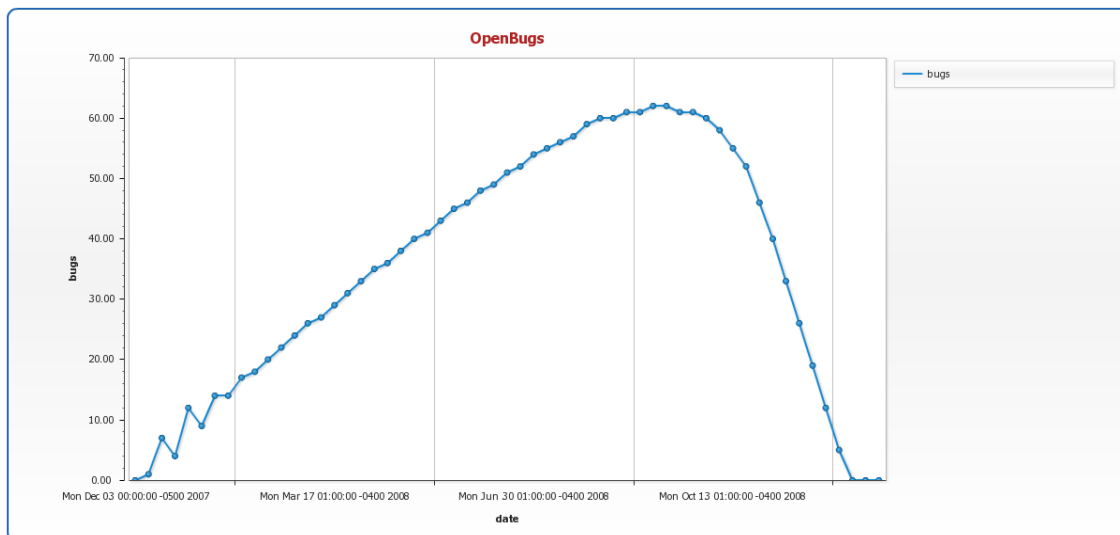


Figure 1 – Open Bug History

Visualizing Project Health

Within a single team, managers bridge the different silos of an org-chart. They collect information daily to assemble an image of overall project health. Team meetings may serve as a forum for that information to trickle back down to each sub-team. However, too often the communication is either one-way, or the view of overall project health simplifies down to a single symptom: “Which person or sub-team has the longest schedule?”

The first principal in reliable progress and risk tracking is to have at least two metrics for any one type of progress. In spite of this, verification bug rates are often the principal metric used for project readiness. The bug rate ramps up as a verification team gains the ability to exercise the design. Later, the bug rate ramps down as the design has fewer and fewer bugs. (See Figure 1)

In recent years, many of the largest semiconductor companies have had internally developed systems to track multi-metric verification progress. Intel’s health-of-model (HOM)¹ environment, as well as methods deployed within AMD and Cisco track verification metrics beyond bug rates, and place results in a historical perspective.

Each of these examples uses centralized data collection to measure the progress, stability, quality, and performance of new designs. The most important goal is to create a custom picture of health that is meaningful for a specific ASIC or FPGA project.

Diversified Verification Tracking

To complement bug rate, a team can track several additional metrics such as the number of tests that pass and fail each week. When taken together, the three numbers create a very reliable metric. Progress looks good when the number of tests passing per week is increasing while at the same time we see decreases in both the number of tests failing and the number of bugs filed per week.

Adding further diversity, scripts can track the number of tests that fail per week as compared to the number of tests debugged per week. To do this, a team captures a comparison between the numbers of simulations run with waveforms enabled vs. simulations that reported a failure status. This can also be done by tracking the random seed values for simulations. If the same seed value is run many times, it indicates that debug activity is going on. The goal is to detect that more tests are failing than are being debugged. Under such circumstances, the team has not yet reached an inflection point where bug rates will begin to decline. (See Figure 2)

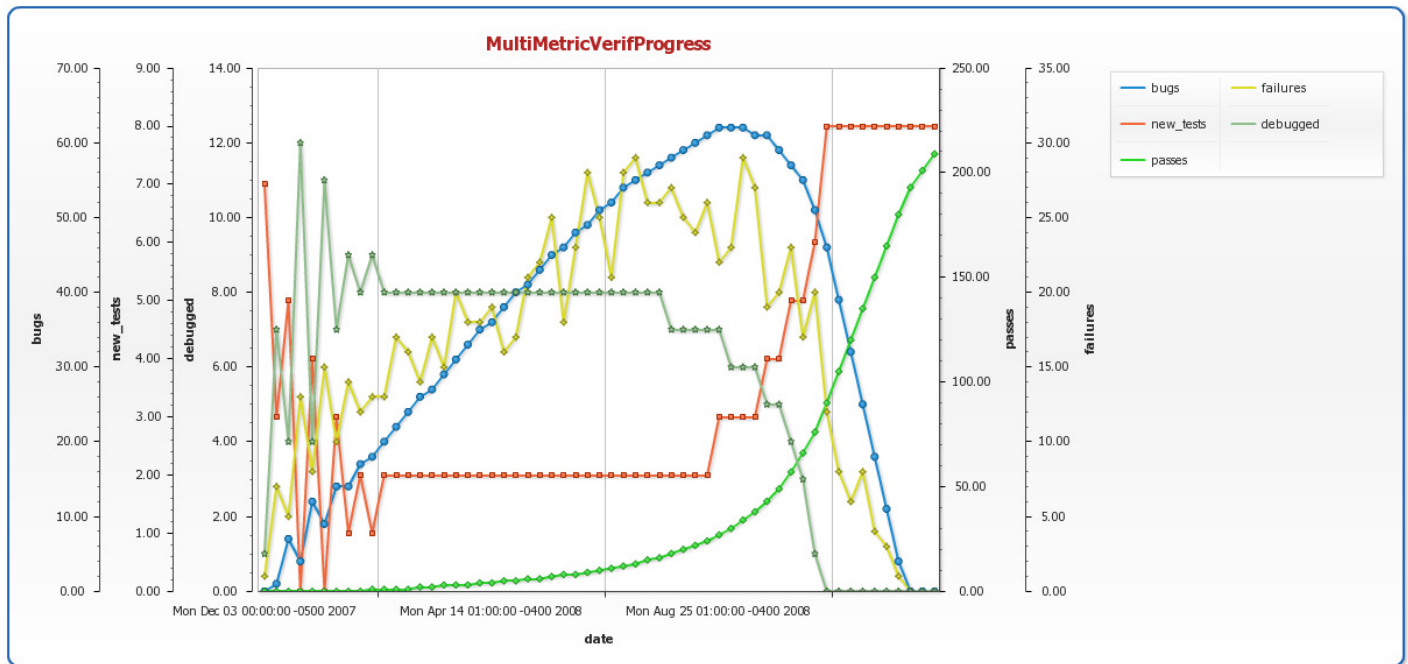


Figure 2 – Multi-Metric Verification Progress

One particularly useful metric is the amount of new testing or new functional coverage created per week. Early in the verification phase, each new test leads to a bug. In later phases, when coverage closure and test plan completion are the focus, tests are written at a faster pace, and most do not result in failures.

Diversified RTL Progress Tracking

While verification metrics are important to a project, they do not represent a full picture of project risk or readiness. It is equally important to track progress metrics in both front-end and back-end design activities.

In RTL coding, harnessing the power of source control tools to measure the number of source-code revisions per week is very helpful. Further analyzing code changes to distinguish between comment lines, non-comment lines, and tool comment lines of code change are all helpful.

Other RTL coding metrics include tracking trends in lint warnings. Beyond the simple number of warnings that a given RTL module has, we can track the number of unique types of warnings in each module. If the trend in lint warnings for a particular module has large jumps in it, it could imply an accidental disabling of important warnings. A more gradual trend over time is indicative of careful RTL code improvements. In Figure 3, the “apple” module has long periods with no lint improvements. When improvement does occur, it is in larger increments. This represents more risky bulk-changes than can be observed in the curves for the “pear” module. (See Figure 3)

Analyzing the demographics of lint warnings adds additional value. If module X is the only module with a particular lint warning, that may indicate higher risk than warnings that are common to many modules.

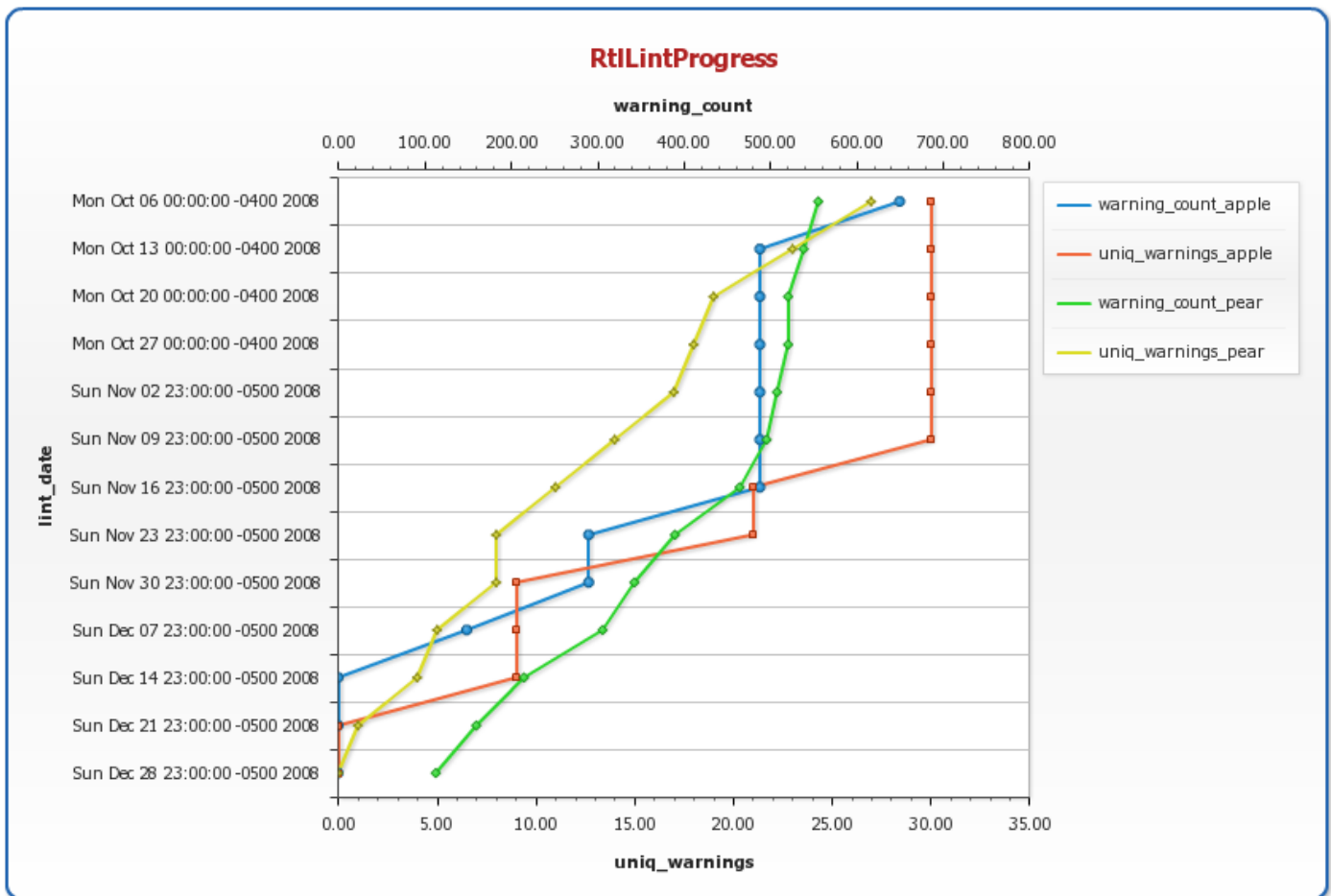


Figure 3 –RTL Lint Progress

Diversified Synthesis and Placement Tracking

Further valuable examples are visible in synthesis and placement work done for FPGAs and ASICs. This is an area where a tool-independent approach is important. Though some common tools exist, many FPGA vendors offer comprehensive tool suites that are specific to their FPGA technology. Likewise, deep sub-micron tool chains differ greatly from front-end synthesis tools and from FPGA flows.

To overcome the differences, we track trends on common or derived progress metrics such as the slack density curves shown in Figure 4. The chart plots the cumulative percentage of nets whose timing slack is worse than a particular value given on the x-axis. (See Figure 4)

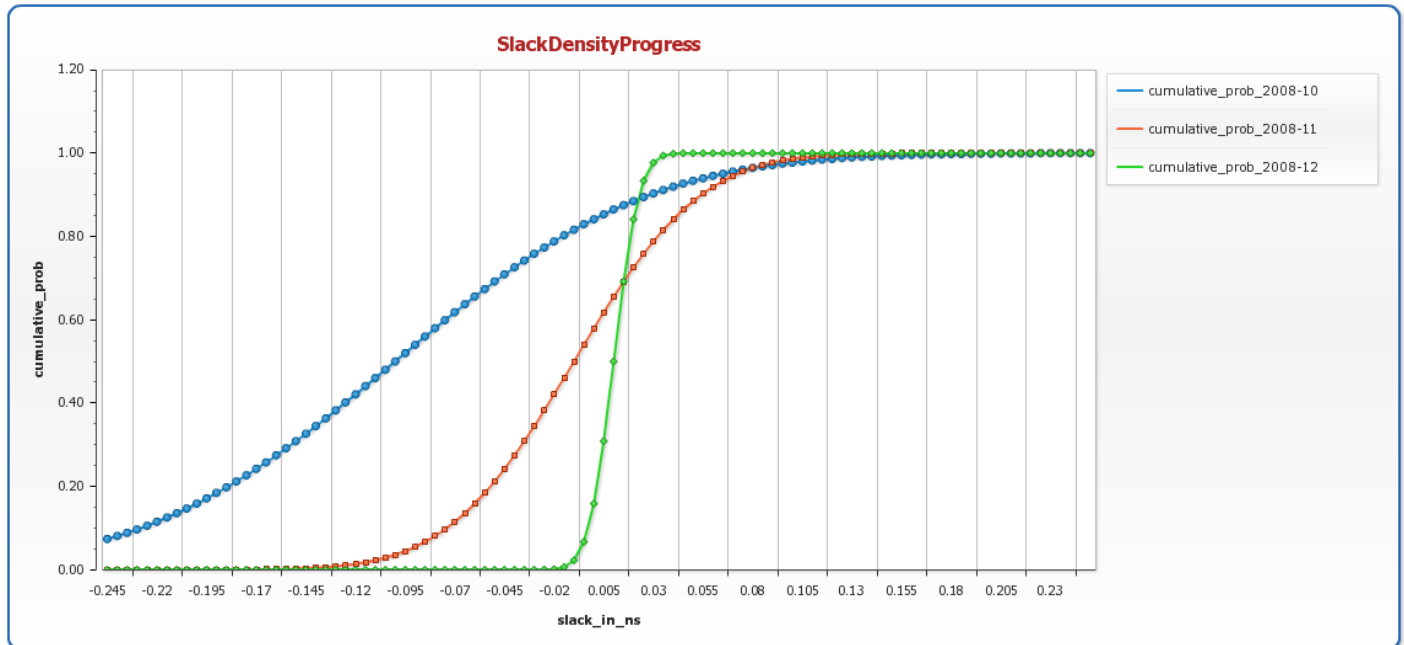


Figure 4 – Slack Density Progress

Trend analysis in other areas of physical design is comparable to the lint and verification examples above. Trends can be found in placement utilizations, path constraint coverage, and per-module compile times. The historical tracking of formal equivalence errors, Design Rule Checks [DRC], Electrical Rule Checks [ERC], Logic vs. Schematic checks [LVS] are all similar to processing lint warnings, and create a diverse picture of a project taking shape.

Conclusions

Though verification will remain the largest project task for some time to come, other risk areas exist. Modern projects overlap several phases of design. As a result, trend analysis in synthesis, physical design, source-code volatility, and static code analysis should not be overlooked.

Tracking cross-tool and cross-team progress metrics adds value to every phase of design. It has become central to the verification environments of some of the largest semiconductor companies. High quality information can help keep a large team in step if it is universally accessible via a web browser.

We have established the precedent and feasibility of extracting each of these information types from common log files and tool output formats. Open-source efforts such as the Unified Coverage Interoperability Standard² represent positive change in the industry and may bring about greater potential for cross-team and cross-tool progress tracking.

References:

- 1) http://www.dvclub.org/images/Presentations/Salamian_DV_Club_Foils_Intel_Austin.pdf
- 2) <http://www.accellera.org/activities/ucis>