Introduction

Guiding programs to “first-pass success” on schedule is the desired outcome for any IC design program officer. However, accurately monitoring progress of complex IC designs has become more difficult at the same rate as the designs have increased in size and complexity, leading to surprises from backwards-looking reporting and management processes that do not forewarn coming crises. The Cadence® Metric-Driven Verification Methodology provides a more uniform and standardized method of reporting progress towards closure in meeting specification requirements. Driven by the vManager™ Metric-Driven Signoff Platform, the solution provides transparent program status, data-mining capabilities, and progressive access to the details of issues that have potential schedule impact, by providing program managers and oversight officers the information to achieve the desired goal of first-pass success.
Management and oversight responsibilities for IC programs have become more stressful and riskier propositions as system complexity has continued at super-linear rates. Some of the basic requirements to improve the management of these programs are:

- Timely access to status and progress
- Accuracy/objectivity of information
- Aggregation of key metrics across multiple sites and sub-projects
- “Drill down” transparency into underlying functional results
- Uniformity of content and reporting from dashboard level to technical level
- Consistent collection of test results from diverse execution engines
- Real-time traceability with change management linked to requirements

Additionally, care must be given so as not to burden already overhead-laden programs with more processes whose sole function is “keeping management in the loop”.

It has been shown in the commercial electronics industry that by moving to the Metric-Driven Verification Methodology—which includes automation for planning, execution, tracking and reporting—a vast majority of the oversight requirements can be automatically collected without manual effort.

The key to higher quality program management and oversight is then to ensure that the industry best practices for IC verification are a requirement. Because more than half of all IC re-spins are due to functional failures, relentless focus on functional signoff early in the program is the only viable path forward. Program oversight and project managers should be demanding adoption of best practices in verification, as it just makes good dollars and sense, especially for important contracts and programs.

There are many documented case studies that have measured the benefits of adopting a full Cadence functional verification methodology. (An extensive list of references is available on request.) These case studies provide a detailed look at the application of each element/practice of the flows, as well as practical insights into deployment and usage. Customer reports in these references show up to 60% verification effort reduction from traditional test methodologies, elimination of manual reporting overhead, and an on-going track record of first-pass success.

**Why Change (or what goes wrong for many today)?**

From a program management and oversight perspective, failure to diagnose the severity of current issues or foresee many upcoming problems prevents effective mitigation strategies from being executed prior to crisis level. Many programs that have extreme delays and ultimate failures already suffer from an “overhead-rich” program structure and processes that hamper efficient use of front-line engineering resources. Because of IC complexity, adding additional reporting requirements, reviews, and processes simply starves the energy that should be applied to making actual progress. Rather than making progress, there is decision churn trying to get to the root cause of the problem (Figure 1).

![Figure 1: Decision churn hampers progress](www.cadence.com)
These added delays translate to real costs, where the metric-driven approach enables visibility from the highest to the lowest level of detail. The after effect of most of these manual processes is reluctance to consider ICs for many applications where they would be ideal if the right development methodology were applied. Instead, the risks of IC first-pass success are traded down for inferior approaches that have larger size, weight, area, and power characteristics, such as choosing an FPGA when a better choice would be to do an ASIC design.

So, if so much energy is being expended on manual progress and issue reporting, what is going wrong? Our experience has shown that there are typically a number of systematic contributing factors that hinder the effectiveness of what are probably otherwise great policies and processes. These factors include:

- No live/automated link between updated requirements and daily engineering tasks
- Lack of mechanisms to quickly triage failures and issue trackable bug data
- Incomplete recording of functional results due to incompatible data from multiple execution engines
- Difficult to share/interpret tool reports and understand trends over time
- No clear visibility to easily and objectively identify holes in the test plan
- Inability to move resources to critical areas due to lack of actual test metrics

These factors are just a sampling of the shortcomings found in many programs that have contributed to overall design failures or lengthy/costly program delays due to the complexities of the IC.

**Solution Elements**

There are many technical descriptions and sources of information on the Cadence Metric-Driven Verification Methodology, so only those elements that directly relate to program management and oversight will be discussed in this paper. A non-technical overview of how the process works can be found in the Metric-Driven Verification Methodology white paper posted on Cadence.com.

The essential and integral part of this methodology is the Cadence vManager Metric-Driven Signoff Platform, which provides several critical functions to codify, provide uniformity, and enable visibility to this critical verification process. These functions include:

- Use of executable verification plans (vPlan), importable from spreadsheets
- Ability to cross-link design requirements and specifications to these verification plans
- Flexible and automated data collection of functional coverage and functional test data
- Back annotation of functional data collected directly to the verification plan
- Ability to score, rank, and easily identify effectiveness in a test suite
- Execute, collect, and combine data consistently across diverse verification engines (simulation, formal, hardware, FPGA prototypes, etc.)
- Verification plan reports organized by design feature to more easily identify functional deficiencies
- Merging/unification of data across multiple projects and sites
- Embedded/commercial SQL database for tracking large-scale data trends over time
- Web dashboards for management reports
- Open and customizable format using commercially standard API to link to other management or enterprise tools

Figure 2 shows the top-level elements of the vManager platform.
The underpinnings of a flexible and scalable solution come from the vManager architecture, which, like any modern management tool, uses a centralized server and SQL database as the heart of the system. From here, many users can connect to the server through a user-interface client, to execute tests and measure their verification progress. This client-server structure enables some key features enabling:

- Multiple users—up to 100 users per server
- Web access for regression control and management dashboards
- Management of multiple projects and programs simultaneously, yet independently
- High-level security between projects, and ability to define “black” projects
- Pre-engineered and optimized integration to the Cadence execution engines
- Data aggregation across global teams and development sites
- Remote engineers using the user-interface client with a secure HTTPS connection
- Simplified web-based regression setup and execution with embedded triage features
- Remote access for project teams for status, dashboards, and centralized reports
- Automated mechanism to track goals and progress over time and abstract to meaningful dashboards

**A Different Approach – Metric-Driven Signoff**

The Metric-Driven Verification Methodology is a cyclic process that drives programs to closure (Figure 3). The process begins with the creation of a plan that details the verification goals and the corresponding metrics, which will execute regressions or tests in the different verification environments running on simulation, hardware acceleration, and formal verification platforms. Execution across the engines is automated using an integrated distributed resource manager (DRM) so jobs can run in parallel. Support for all popular DRMs, such as LSF, are included in the vManager platform. (See the vManager Metric-Driven Signoff Platform datasheet for specific details.)
As tests complete, test failures enter into a failure analysis, triage, and debug sub-cycle. Tests that have failed can be automatically re-tested by the vManager platform, and verified failures are rerun with debug data turned on, then sent to the responsible engineer with a failure signature indicating where to look. After analysis, confirmed failures are sent to a centralized bug management system (like Bugzilla or Jira), where a bug ID can be initiated by the vManager platform or from the bug system, where status updates on the bug stay synchronous using the powerful vManager API (vAPI).

The data collection process (Multi-Engine Coverage Analysis) happens post execution. Functional and code coverage metrics are collected and merged together so that a single set of results can be seen. Coverage is directly correlated to bugs, and is used extensively to forewarn users of possible design defects. This forward-looking view of bugs is critical in avoiding cost and schedule overruns down the line.

Next, the vManager platform automatically connects the test results and coverage metrics to the vPlan. Goals that are fully met, partially met, or not met at all are clearly visible in percentages and with color coding. You can clearly see in Figure 4 that the I/O subsystem is yellow, and that the main cause is failed connectivity, shown in red. This is an easy way for program managers to see the features of the design with real-time updates on its functional status, without having to understand all of the underlying technology. Organization of the vPlan is fully user defined, and because it is hierarchical, top-level program manager views can be seen as well as lower-level engineer views at the lowest levels of hierarchy. Pre-defined and user-defined views (perspectives) are stored in the vManager platform, and can be shared with everyone on the project.
The next step is project tracking, which is the process of tracking key metrics over time. All of the results data from execution is stored in a SQL database. If reporting is done weekly, the vManager platform will be setup to “snapshot” the results weekly, and also to store this information in the database. Any data can be captured and tracked this way, including passing tests, failures, coverage results, plan data, progress data, trends, goals, and even external data like number of bugs. Data is presented to the user in absolute and cumulative fashion, with highly configurable options for how the chart should be configured. Export of this data via CSV is also possible for greater levels of charting in spreadsheets. Each project will have its own data set, such that it can be reported and maintained separately. This provides a valuable trajectory view towards closure of the program.

![Snapshot Data](image)

Figure 5: Snapshot data for progress-over-time reporting

All external data, like the last bug example, comes into the vManager platform using an industry-standard API that can connect to virtually any commercial enterprise system. All enterprise systems use a REST-based API mechanism, and this is what the vManager platform uses for batch automation and connectivity, and to populate web dashboards. The underlying mechanism in any web page found through Google, for example, uses this REST API technology.

**Program Management and Oversight Reporting**

Based on the connection between the daily engineering activities and the top-level plan, management tracking data can be extracted and pushed to a web page, which sees live data coming directly from the embedded vManager web service. At any time, users can get up-to-the-minute progress reports. Any HTML5-compatible device such as a tablet, smart phone, or web browser can log in and see the project data if they are granted permission. The charts are automatically created, and users can visualize many charts or views of the same program, or you can define each web chart as a different program.
The underlying reporting works via a snapshotting capability that can be configured for each program. The snapshot data includes time/date information so that progress can be reviewed as a function of time.

**Security Features – AAA**

Ensuring integrity in the process and reports is an essential element to strong management and oversight functions. The vManager platform’s security protocol is based on three key areas, authentication, authorization and audit, referred to as AAA, that support the NIST 800-171 standard for US Government projects. The verification engineer’s client interface (Linux) and the web-based client interface are tied to the same security protocol within the server.

- **Authentication** provides a private user password for each user. This typically can come from the corporate LDAP password server, or can be setup by the vManager administrator.
- **Authorization** levels tell the vManager platform which users have which permissions to see and or interact with which program or project using their email user ID and password. The vManager platform offers system-level permissions, project-level permissions, and user or group permissions. Additionally, each data area can be walled off if needed, special functions such as the ability to change project-tracking data or modify user defined attributes that may be setup to capture bug data.
- **Audit** is the process of monitoring and capturing who did what when, such that if authorization changes, the audit mechanism catches that information.

**Reporting**

Not all effort turns into progress. Being able to spot inefficient expenditures of time, energy, and resources is a hallmark of good program oversight. Of course, obvious gaps in progress are easy to spot with the vManager platform reporting. Figure 8 shows the (customizable) color coding for metrics reporting. The “red flags” are flagged red. The vManager platform provides links back to the verification where further diagnosis is possible for each test associated with each design feature. In essence, the vManager platform provides a data-mining continuum from the high-level dashboard views down to, if necessary, the lowest level of actual design and verification artifacts.
There are live HTML links for all the reports that are generated. Reports are posted in the vManager platform's web reports portal, and also serve to provide a mechanism to dig deeper into the source and nature of the open issues that stand in the way of closure. HTML reports are also drillable, like the vPlan, to expose the top-level results, all the way to the lowest level of detail needed to identify status. And last, all report data is accessible thru the API for users to integrate into their own management systems.

**Insights for Program Managers**

Based on the experience of many high-complexity IC programs, the following practices have been effective in ensuring programs stay on track:

- First priority is tracking how much of the verification plan has been implemented
- Second priority is to track how much of the functional coverage has been closed in order to know how well the project is tracking to verification plan
- Set milestones (not just the final goal) and have visibility into how well the project is tracking to your verification plan goals for a given milestone
• Look at the code coverage to ensure you have thoroughly exercised the implementation:
  – If some code can’t be covered, use unreachability analysis to ensure there is no dead code that should be removed
  – If there is high functional coverage, but low code coverage, that team probably did not do a good job planning and has a weak vPlan and functional coverage metrics
• Don’t focus so much on “test suite” growing, but do expect to see a growing number of simulation cycles over time, which will give some indication how much verification is running
• Observe code churn metrics by measuring revision control check-ins over time. Expect this to be going down over time. A lot of code churn late in the project usually means more bugs are being inserted, which will extend verification. This data can also be sent to the vManager platform through the API from your source code repository system.
• Use more formal verification since it can be more effective/efficient than simulation for different types of design features, but you should understand what is covered by formal so there are no duplicate efforts in simulation or something was missed due to the assumption it was being done in formal
• Know how much your system-level software tests exercise the chip when running on the Palladium platform so you understand how much they are contributing to your verification plan goals

Following these program oversight practices has been a key ingredient in achieving first-pass success, on schedule and within budget, in numerous ICs.

**Summary**

Strong program management and oversight for ICs is predicated upon timely, accurate information availability containing forward-looking predictive metrics. Having the ability to have reports generated automatically from daily engineering activities has the benefits of timely availability and accuracy of actual status metric measurement for early bug detection. Further, because of the automation, the actual overhead normally needed to collect, collate, and summarize status can be reduced to virtually zero. Through automation, rather than adding manpower in the program schedule, the engineering teams can expend a larger percentage of their time on program closure.

The key capabilities for management and oversight provided by the vManager platform and discussed in this paper are:

• Timely access to status and progress
• Accuracy/objectivity of information
• Transparency into underlying details
• Uniformity of content and format of reporting

The Cadence Metric-Driven Verification Methodology, driven by the vManager Metric-Driven Signoff Platform, has been used on hundreds of successful IC programs, including some of the largest and complex ever designed. The scalability and extensibility of the solution makes it adaptable to diverse sets of requirements from simple projects that need greater visibility, to the most demanding large-scale global programs.

**Further Information**
