Hardware Verification Planning Tools

A Comprehensive Compilation

Mark Chen
Angelia Technologies
www.ipcoredesign.net
February 18, 2023

1. FOREWORD

Verification planning is one of the most important as well as the very first steps in the entire hardware verification process. Without a careful planning, the verification is very likely to fail.

Traditionally verification planning is done by hand, manually with a piece of paper and a pen or some text files or word documents at most. That's not enough nowadays. In a era when chip designs involve with teams of many design and verification engineers, automatic, coordinative planning is required. And what's more demanding is that planning must be kept in pace with the entire verification and even design progress, or in another words, make the verification planning and plans executable.

Therefore, integration of the verification planning and plans into the overall design and verification process is the new and challenging tasks facing the verification engineers. We are seeing this trend to become reality in some of the EDA tools.

The big guys in the EDA industries have developed, over the years verification planning tools in one way or other, with differing features and functionality. Yet these is no standalone verification planning apps nor are there any free and open source tools for verification planning.

So if you want to plan your verification, you either have to do it manually or use some of the ready-to-go EDA tools where you can find the verification planning module within these EDA tools.

2. MENTOR QUESTA® VERIFICATION MANAGEMENT / MENTOR GRAPHICS® ENTERPRISE VERIFICATION PLATFORM

Verification Planning with Questa® Verification Management

Questa Verification Management manages the data complexity, guides the process and provides automation across all verification engines to improve verification productivity. It offers analysis and optimization features built upon the Unified Coverage Interoperability Standard Database (UCISDB), with results and trend analysis, test plan tracking and run management. Questa Verification Management efficiently ties all verification-related tasks together and gives all parties — system architects, software engineers, designers and verification specialists — real-time visibility into the project. This visibility helps to hit market windows on schedule, manage risk and improve throughput and debug turnaround times.

Verification of complex SoC (System on Chip) requires tracking of all low level data (i.e. regression results, functional and code coverage). Usually, verification engineers do this type of tracking manually or using some automation through scripting.

Manual efforts in order to get above information while verifying complex SoC may lead towards the delay in project execution. A verification planning tool can help to reduce such manual efforts and make the tracking process more efficient. Mentor, A Siemens Business has such a Verification Planning tool for **QuestaSim** within their Verification Management tool suite known as "Questa® Testplan Tracking".

INTRODUCTION

Mentor's Verification Testplan Tracking facilitates the verification process management across all relevant aspects. Without such automation, tracking of verification progress requires a lot of manual effort in order to get information like coverage and regression status in a common place. Functional and code coverage numbers are key parameters in verification and without it verification cannot be closed. Regression analysis (individual test status) is also an important aspect of tracking the verification progress.

Questa® Verification Management automates the tracking of verification progress by offering the analysis and optimization facilities followed by additional features like trend analysis, back annotation of results, filtration of specific data and other features. It provides the top level summary of all verification aspects, which provides real time visibility into the project to all parties, i.e. designers, verification engineers, project managers, etc. Due to this visibility, it becomes easy to manage risk factors and efficiency so project execution can be improved.

Questa as a verification tool has its own procedure in order to enable verification engineers to check if their verification plan is met or not. In this document we will see steps that can be followed in order to use Questa to apply your verification plan.

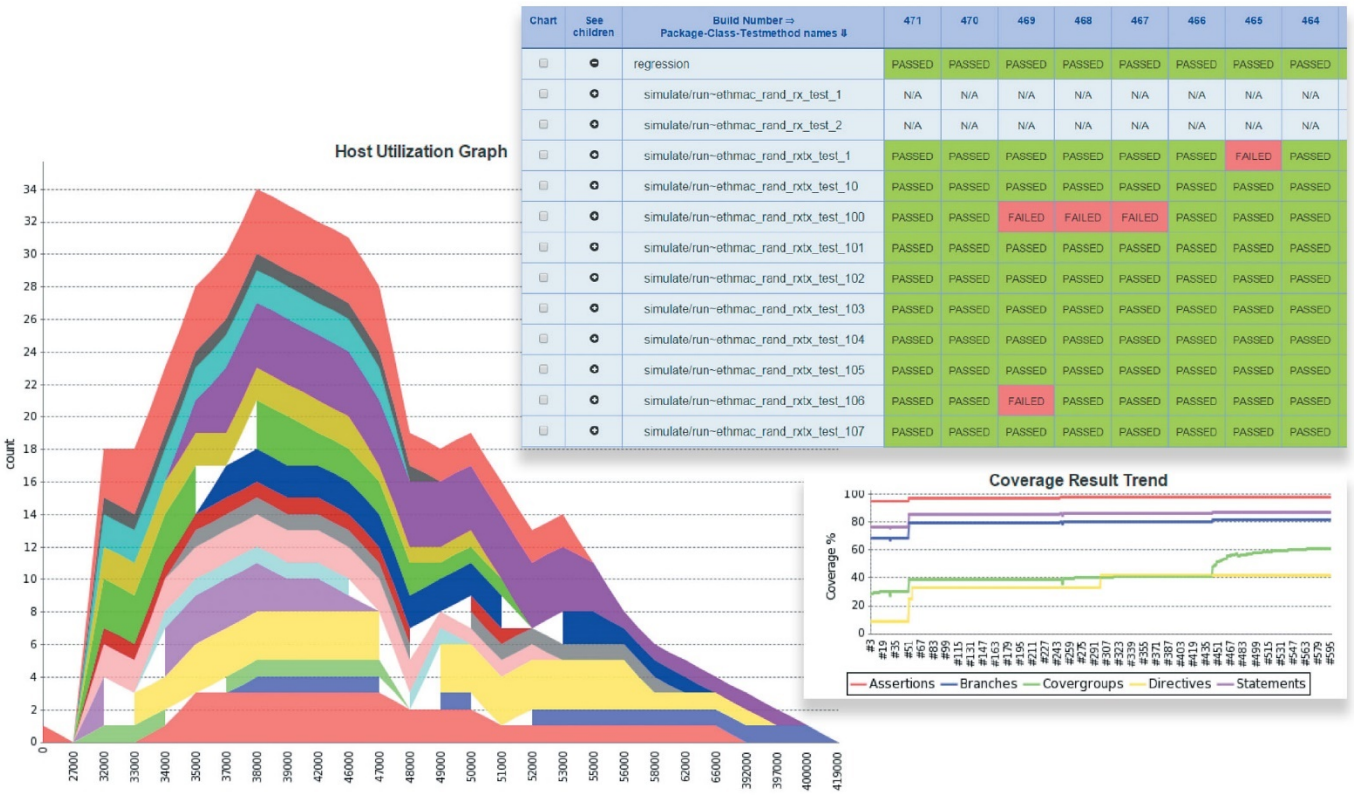
Features

Automated Productivity

Questa's verification run manager provides project consistency through broad automation. An intuitive web dashboard lets teams observe and analyze project results and trends, allowing dispersed project teams to improve time to coverage and time to next bug, and enhancing their ability to accurately estimate the time to completion.

High Performance, High Capacity

Built on multiple powerful technologies, the Questa Verification Platform enables teams to select the best application or tool for the job, and combine results from all the engines to dynamically track the progress of the entire verification program, dramatically increasing productivity and more efficiently managing resources.



Questa's verification management capabilities are built upon the Unified Coverage Database (UCDB). The UCDB captures any source of coverage data generated by verification tools and processes; Questa and ModelSim use this format natively to store code coverage, functionality coverage and assertion data in all supported languages.

UCDB also enables the capability to capture information about the broader verification context and process, including which verification tools were used and even which parameters constrained these tools.

The result is a rich verification history, one that tracks user information about individual test runs and also shows how tests contribute to coverage objects.

Process Management

Verification is driven by requirements concerning both the functionality of the final product and the intended methods of testing this functionality. By providing tools to import verification or test plans and then guide the overall process, Questa verification management helps deal with this complexity and shepherd a project toward electronic closure. It also provides the ability to store snapshots of data across the lifetime of a project, which helps to concentrate efforts where they are most needed.

MOTIVATION FOR QUESTA® TESTPLAN DEVELOPMENT

Functional verification has been described as a major challenge in SoC designs with proper visibility into the verification process as a major factor contributing to this challenge. This lack of visibility impacts design quality, schedule predictability and cost.

A testplan is a document which captures the important features of a design and details regarding how they will be verified. Questa testplan is such a plan which can be linked directly to the coverage database and results can be annotated in that plan itself. By putting such a plan in place that captures list of verification intent, one can organize the tracking of verification progress better. While verifying the complex SoC (System on Chip), it may be possible that all the features we have listed in the Questa Testplan do not have equal priority. For that case, one can add separate user defined attributes in the testplan to provide priority for individual features. Verification engineers can focus their coverage closure efforts on the critical features that have been identified as a whole followed by important features and then by the "nice to have" features.

BASIC FLOW FOR QUESTA® VM VERIFICATION PLANNING

Questa Testplan includes feature wise test scenarios, functionality covered by that feature (functional coverage), assertions, code coverage and such more in ***XML format***. After preparing such Questa Testplan, conversion of such plan in UCDB (Unified Coverage Database) format is required using ***xml2ucdb*** command. After running the regression, separate coverage databases will be available for individual test scenarios. At the time of merging of these coverage databases, Questa Testplan in UCDB format needs to be merged so that the results can be linked with the data available in Questa Testplan.

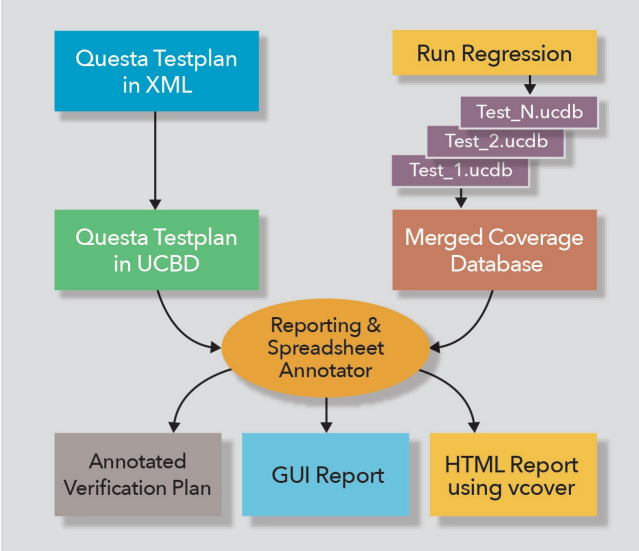


Figure 1. Basic Flow Diagram

One can back annotate the results in XML format of Questa Testplan and the same results will be available in HTML report or in GUI as well. Creation of the Questa Testplan required certain steps and syntax, which is explained in the upcoming section.

DETAILED STEPS FOR QUESTA® TESTPLAN DEVELOPMENT

In many cases, the features will be verified in simulation and recorded as verified using coverage analysis. For Questa Testplan which facilitates back annotation of information, linking between the testplan and simulations is required. For that, certain document format must be followed. The format for Questa Testplan is described below with detailed steps.

Step: 1. Install Questa add-in in Microsoft® Excel

By default, Microsoft Excel does not contain Questa option in toolbar. So the user needs to install Questa add-in which is open source and freely available. After installing this add-in, it will be available the toolbar as shown in figure 2.

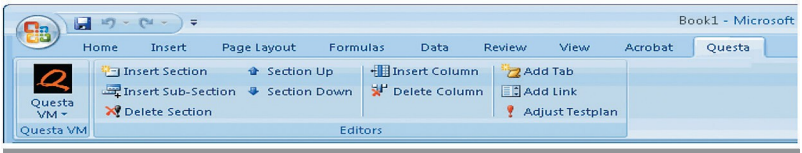


Figure 2. Questa add-in in Microsoft Office Toolbar

Step: 2. Prepare the Questa Testplan

For creating the Questa Testplan, select create testplan option from Questa VM option as shown in figure 3.

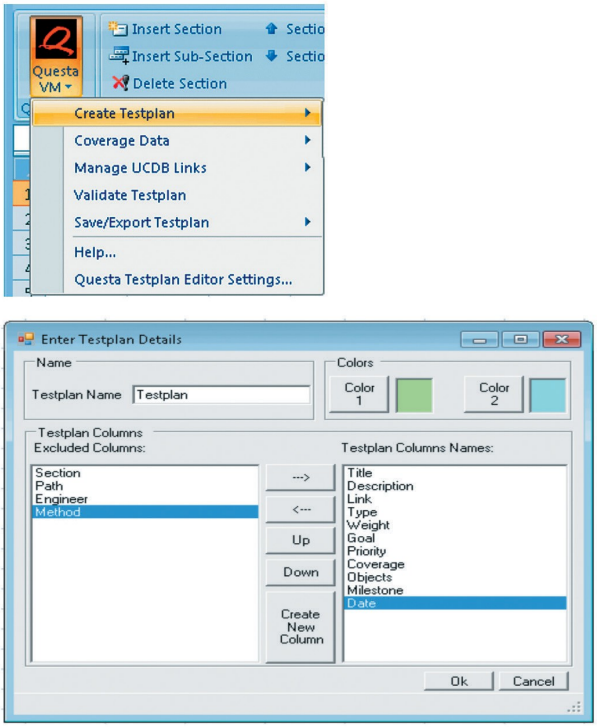


Figure 3. Questa® VM options and Testplan Details

Questa's Testplan Tracking requires four distinct pieces of information for each requirement captured. They are ***Section, Title, Link*** and ***Type***. ***Link*** is one of the attributes used while developing the Questa Testplan, which is required to link plan information to the actual results. The additional information for each requirement includes a ***Description***, a ***Weight*** and a ***Goal***. While these additional fields are not required, they are used the majority of the time.

Questa's Testplan Tracking also has the flexibility to allow user defined fields. If we look at the typically used fields in a spreadsheet format, each field is represented by a column in the spreadsheet as shown in figure 4.

Section	Title	Description	Link	Type	Weight	Goal
1						
1.1						
1.2						

Figure 4. Sample Questa Testplan Format

After preparing the Questa Testplan, it can be saved as XML or can be directly exported to XML from Questa VM options as shown in figure 3.

User can track different information by specifying the different options in ***Type*** field. Description for options available for ***Type*** field is as shown in table 1, below.

Table 1. Coverage Construct Type Field options

Coverage Construct — in "Type" field	Description
Assertion	Assertion statement
Bin	Coverage item bin
Branch	Branch coverage scope
Condition	Condition coverage scope
CoverGroup	SystemVerilog covergroup statement
CoverPoint	SystemVerilog coverpoint statement
CoverItem	Generic name for any coverage or design object in a UCDB. This can be used to specify any objects not fitting into another category of construct.
Cross	SystemVerilog cross-coverage statement
Directive	PSL cover directives and SystemVerilog "cover" statements/properties
DU	All coverage on a given design unit
Expression	Expression coverage scope
Formal_Proff Formal_Assumption	Formal property of assertion object
FSM	State Machine coverage scope
Instance	All coverage on a given instance
Rule	Forms a link using an automatically created virtual covergroup "User Rules" — either from a set of pre-defined Rules, or one you create.
Tag	Forms a link using any coverage tag command arguments which are specified in the Link column.
Test	Link to test attribute record. This is the test name.
Toggle	Toggle coverage scope
XML	Triggers hierarchical (nested) testplan import.

According to the requirement, user can set several rules as per the option available in ***Add Rule*** option. In figure 5, rule is applied to get the recursive toggle and statement coverage for some particular module, as shown on the page.

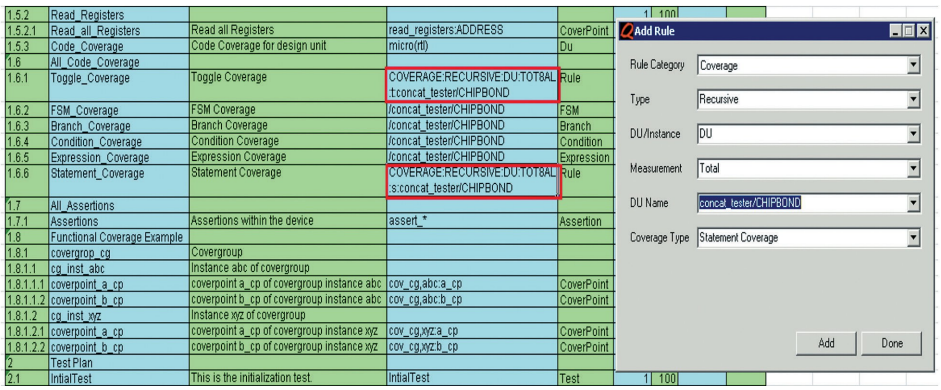


Figure 5. Add Rule option in Questa Testplan

Step: 4. Regression and Coverage Database generation

After preparing the Testplan, next step is to develop test cases to verify the features listed in plan. After running the regression of those test cases, command for merge all separate coverage databases is as shown below:

vcover merge merged_cov.ucdb
<test1.ucdb> <test2.ucdb>…
<testn.ucdb> verification_plan.ucdb

We can back annotate the results in existing XML format Questa Testplan using option below:
Questa VM > coverage data > annotate

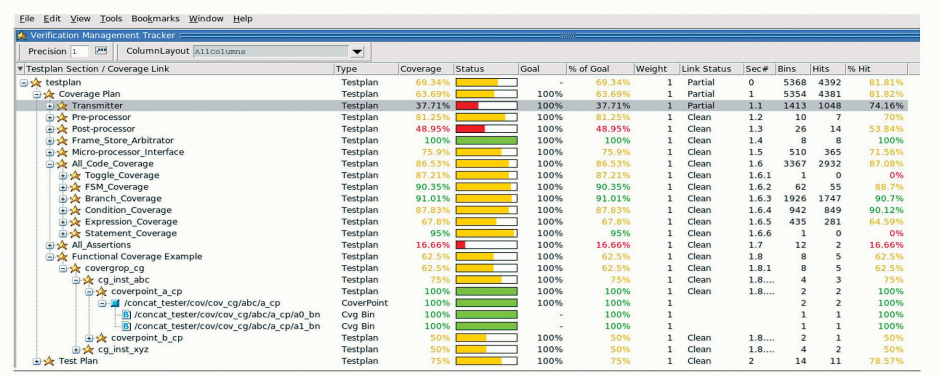
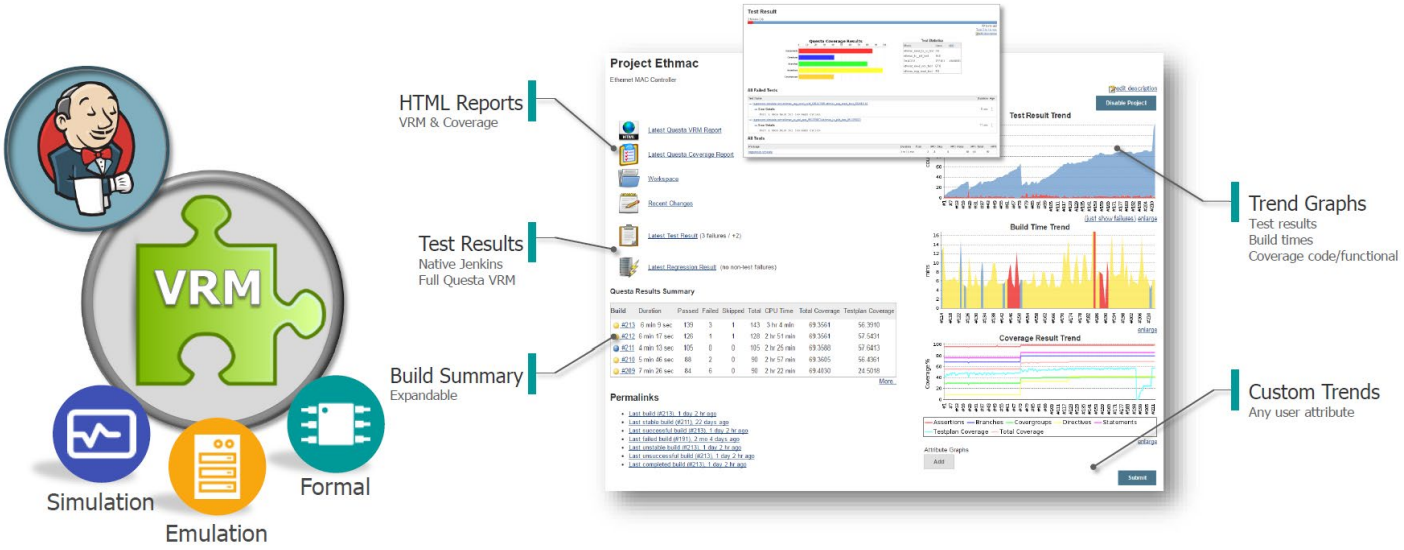


Figure 6. Verification Management Tracker in GUI

Step: 5. HTML Report generation

Command for the HTML report generation from merged coverage database is as shown below:

vcoverreport -html -htmldircovreport merged_cov.ucdb -details -testhitdata

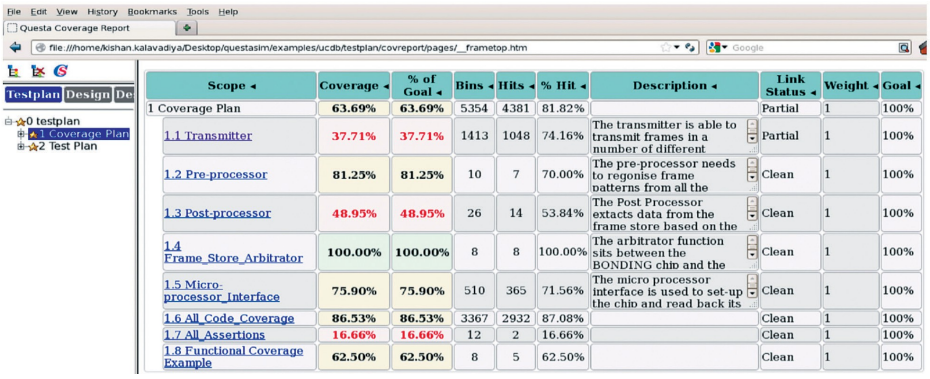


VERIFICATION RESULTS ANALYSIS

Results Analysis
—Quickly Identify and Organize Results
—Improve debug turn-around time

Questa's Testplan Tracking facilitates with multiple formats for analysis of the results hierarchically so that it will become easy to address the failures identified in regression and track the coverage holes to complete the verification process.

Figure 6 and figure 7 contain the top level summary of Questa Testplan in GUI and HTML format respectively. Annotated Questa Testplan in XML format in figure 8 contains the same information which is shown in figure 6 and figure 7. Figure 8 shows the syntax for different ***Link*** options as per the requirements.



Scope	Coverage	% of Goal	Bins	Hits	% Hit	Description	Link Status	Weight	Goal
1 Coverage Plan	63.69%	63.69%	5354	4381	81.82%		Partial	1	100%
1.1 Transmitter	37.71%	37.71%	1413	1048	74.16%	The transmitter is able to transmit frames in a number of different	Partial	1	100%
1.2 Pre-processor	81.25%	81.25%	10	7	70.00%	The pre-processor needs to regonise frame patterns from all the	Clean	1	100%
1.3 Post-processor	48.95%	48.95%	26	14	53.84%	The Post Processor extacts data from the frame store based on the	Clean	1	100%
1.4 Frame_Store_Arbitrator	100.00%	100.00%	8	8	100.00%	The arbitrator function sits between the BONDING chip and the	Clean	1	100%
1.5 Micro-processor_Interface	75.90%	75.90%	510	365	71.56%	The micro processor interface is used to set-up the chip and read back its	Clean	1	100%
1.6 All_Code_Coverage	86.53%	86.53%	3367	2932	87.08%		Clean	1	100%
1.7 All_Assertions	16.66%	16.66%	12	2	16.66%		Clean	1	100%
1.8 Functional Coverage Example	62.50%	62.50%	8	5	62.50%		Clean	1	100%

Figure 7. Questa Testplan in HTML

Section	Title	Description	Link	Type	Weight	Goal	Coverage	Objects
1.6	All_Code_Coverage				1	100	87.0642	1136
1.6.1	Toggle_Coverage	Toggle Coverage	RECR_COV_INST:t:/concat_tester/CHI	Rule	1	100	58.7475	1
1.6.2	FSM_Coverage	FSM Coverage	/concat_tester/CHIPBOND	FSM	1	100	90.3559	3
1.6.3	Branch_Coverage	Branch Coverage	/concat_tester/CHIPBOND	Branch	1	100	69.1738	596
1.6.4	Condition_Coverage	Condition Coverage	/concat_tester/CHIPBOND	Condition	1	100	33.9369	359
1.6.5	Expression_Coverage	Expression Coverage	/concat_tester/CHIPBOND	Expression	1	100	68.0871	176
1.6.6	Statement_Coverage	Statement Coverage	RECR_COV_INST:s:/concat_tester/CHI	Rule	1	100	82.0843	1
1.7	All_Assertions				1	100	16.6667	12
1.7.1	Assertions	Assertions within the device	assert_*	Assertion	1	100	16.6667	12
1.8	Functional Coverage Example				1	100	62.5	4
1.8.1	covergrop_cg	Covergroup			1	100	62.5	4
1.8.1.1	cg_inst_abc	Instance abc of covergroup			1	100	75	2
1.8.1.1.1	coverpoint_a_cp	coverpoint a_cp of covergroup instance abc	cov_cg,abc:a_cp	CoverPoint	1	100	100	1
1.8.1.1.2	coverpoint_b_cp	coverpoint b_cp of covergroup instance abc	cov_cg,abc:b_cp	CoverPoint	1	100	50	1
1.8.1.2	cg_inst_xyz	Instance xyz of covergroup			1	100	50	2
1.8.1.2.1	coverpoint_a_cp	coverpoint a_cp of covergroup instance xyz	cov_cg,xyz:a_cp	CoverPoint	1	100	50	1
1.8.1.2.2	coverpoint_b_cp	coverpoint b_cp of covergroup instance xyz	cov_cg,xyz:b_cp	CoverPoint	1	100	50	1
2	Test Plan				1	100	88.6364	16
2.1	IntialTest	This is the initialization test.	IntialTest	Test	1	100	100	1
2.2	ModeTwoTest	Tests the chip in mode two.	ModeTwoTest	Test	1	100	100	1
2.3	DataTest	Runs Data from end point to end point.	DataTest	Test	1	100	100	1
2.4	FifoTest	Tests the FIFOs overflows.	FifoTest	Test	1	100	100	1
2.5	CPURegisterTest	Tests the micro interface to the host.	CPURegisterTest Random*	Test	1	100	75	4
2.6	VariableTest	Checks the user RAM.	VariableTest	Test	1	100	100	1
2.7	TxDataTest	Transmit only test.	TxDataTest	Test	1	100	100	1
2.8	SyncTest	Sync test from end to end.	CPURegisterTest	Test	1	100	0	1
2.9	ShutDownMode	Tests the shutdown mode of the chip.	VariableTest	Test	1	100	100	1
2.10	PowerSaveMode	Tests the power saving mode of the chip.	TxDataTest	Test	1	100	100	1
2.11	Random Testcases		Random*	Test	1	100	100	3

Figure 8. Annotated Questa Testplan in XML

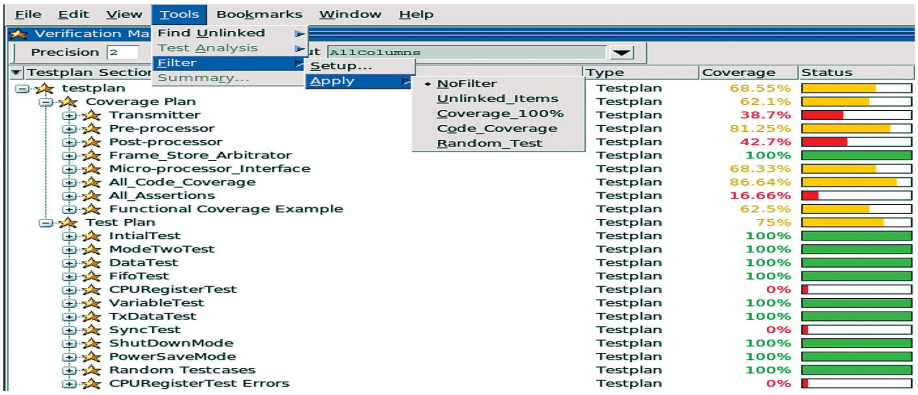


Figure 9. Filtration in Verification Planner

Asterisk (*) is permissible in case of the same naming convention. Here in the above annotated Questa Testplan, **section 2.5** contains 4 random test cases, out of which, 3 are passing and 1 is failing. To link these test cases with merged database, ' * ' is being used in *Link* attribute. Here, pass/fail status of these test cases is in terms of coverage itself (i.e., getting 75% coverage for 4 objects means 3 test cases are passing out of 4 random cases). Similarly, we can get the results in cases of multiple iteration of the same test case.

Filtration is also one of the key features of Testplan Tracking, with which a user can filter out user- specific results from the whole report. A user can use *Attribute Name, Coverage Numbers, Weight, Link Type, Instance Type* and much more to filter the required data. Users need to create individual filters according to the requirements, as shown in the GUI illustrated in figure 9.

Select Tools > Filter > Setup > Create
Users can apply the filter using the option below:
Select Tools > Filter > Apply > Name_of_Filter
Example for the filtration feature is shown in figure 9.

TREND ANALYSIS

Trend Metrics
—Monitor progress throughout the project
—Mitigate verification risks
—Help improve future project timescale estimations

Trend analysis offers the ability to track the progress of the coverage over time of all the objects. For trend analysis, user needs to manage one trend UCDB after each regression. Trend UCDB is nothing but a special purpose UCDB file which contains the coverage data over the period for trend analysis.

Trend reporting offers 2-dimensional representation of coverage number over the period for individual feature or instance of DUT. Trend report can be viewed in multiple formats (i.e. HTML Report, XML Report, etc.)
Trend reports appear in the GUI using trend UCDB at
Verification Management > Browser > Trend Analysis

Command for trend UCDB generation from command line is as shown below:

vcover merge -trend [-output] <trend ucdb> <ucdb inputs>

Sample trend reports and a graphical representation for the same is shown in figure 10 and figure 11 respectively.

Trender

Trendable Object Name	Type	Feb-12-09 15:21	Mar-12-09 15:21	Apr-12-09 15:21	Jan-12-09 15:20	Ma
Design units						
work.top	DU Module	26.00%	18.60%	23.06%	25.00%	
work.coverpkg	DU Package	0.00%	50.00%	75.00%	0.00%	
statecover	Covergroup	0.00%	50.00%	75.00%	0.00%	
cvpi	Coverpoint	0.00%	100.00%	100.00%	0.00%	
cvpstate	Coverpoint	0.00%	25.00%	75.00%	0.00%	
i_x_state	Cross	0.00%	25.00%	50.00%	0.00%	
Vtop/mach/i...	Covergroup inst...	0.00%	29.17%	41.67%	0.00%	
Vtop/machin...	Covergroup inst...	0.00%	29.17%	41.67%	0.00%	
work.statemach	DU Module	4.10%	38.05%	50.95%	0.00%	

Library | Files | Instance | merged

Figure 10. Date wise trend report in VM Trender (GUI)

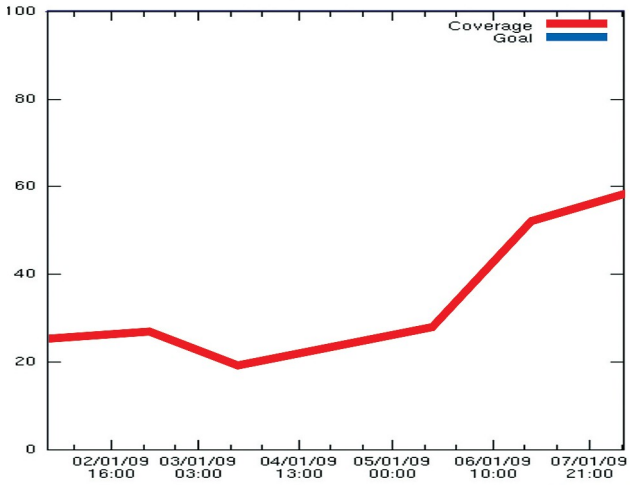


Figure 11. 2-Dimensional Graph in Trend Analysis

CONCLUSIONS

Verification of complex SoC projects is a complex process to manage without Verification Management. Automation of Verification Planning provides deeper visibility into the regression process to allow for throughput optimization.

Questa VM's Testplan Tracking Tool enables us to analyze the regression failures and coverage holes more efficiently by providing the reports in multiple formats. Use of Testplan Tracking can reduce the manual effort which is required to update the verification documents at the time of closure and quicken the documentation in a more appropriate format.

3. MENTOR QUESTA VERIFICATION IQ

Verification Planning with Questa Verification IQ

Questa Verification IQ is the Siemens EDA collaborative, data-driven verification solution that transforms the verification process using analytics, collaboration, and traceability, supporting verification management, debug, and coverage.

Features

Verification Traceability and Safety Compliance

- Collaborative Verification Plan Authoring
- Supporting Plan and Requirements Driven Verification
- Unified VIQ Platform supports Multiple Engines

Employing the Power of Lifecycle Management

- Cloud Ready Team Based Verification Plan Authoring
- OSLC Standards Based Integration with ALM
- Tight Integration with all other Verification Tasks

The company is positioning the new toolset as a team-based, cloud-enabled, data-driven platform that makes use of artificial intelligence (AI) technology. Questa Verification IQ is aimed at helping IC design engineers to do verification closure faster, streamline traceability, optimize resources, and shrink overall design process times.

Applying ML

Questa Verification IQ employs predictive and prescriptive verification analytics to accelerate closure, accelerate debug turnaround time, and provide regression efficiency. Questa Verification IQ facilitates identifying coverage holes, uncovering patterns within data, prioritizing tests that are predicted to fail, and identifying tests more likely to increase coverage.

Questa Verification IQ uses machine learning to analyze data from Siemens' portfolio of verification and simulation solutions including Questa, OneSpin, Symphony, and Veloce. Questa Verification IQ analyzes this data to help logic verification teams do their work more efficiently. The new software also integrates with continuous integration (CI) tools like Jenkins to help automate workflows.

Applying ML analytics provides quicker navigation and highlights possible causes of low coverage while supporting what-if-analysis.

Web-Based Application Framework

Questa is implemented in a web-based application framework. This makes it easy to scale while also reducing installation costs and ensuring device and OS independence. According to the company, Questa supports public, private and hybrid cloud configurations with native collaboration and centralized data access.

The a web-based application framework provides scalable verification management. Questa Verification IQ presents all tasks within a familiar, modern, user interface protected by a secure, login-based licensing model, and it supports URL sharing and user-based notification systems.

Questa Verification IQ's browser-based framework includes a process guide so that you can build a safety critical flow using lifecycle management to plan and track all requirements. The regression navigator enables your team to create and execute tests, monitor the results, and have a complete verification history. With the coverage analyzer you know how complete your coverage is for code, functional blocks and test plans. Finally, the data analytics presented provide you with a metric platform, using project dashboards and providing cross analytics.

The web-based framework scales for any size of electronics project.

The Questa Verification IQ software is based on the RESTful web-based API to pull together a wide range of verification engines to allow multiple teams, including functional safety, to work on the same project.

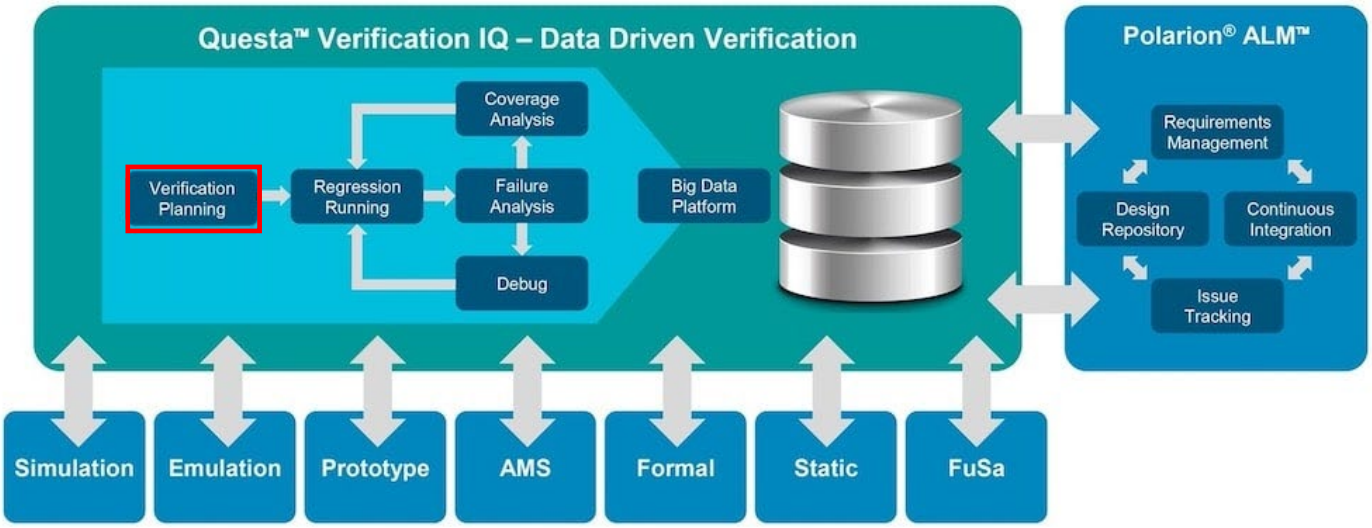
AI tool chain optimized for 5nm ARM chips
AI-driven verification platform boosts bug hunting
AI optimization moves into system design tools

The use of the RESTful API also opens up access to requirements management tools such as Doors and Siemens Polarion as well as product lifecycle management (PLM) tools and both in-house and third party machine learning tools.

The new framework allows any device with a browser to access the tools using the standard REST web API and Open Services for Lifecycle Collaboration (OSLC) API which also uses REST interface.

Verification IQ Framework

Here's the big picture of how Quest Verification IQ connects together all of the data from various verification engines into a data-driven flow, along with an ALM tool.

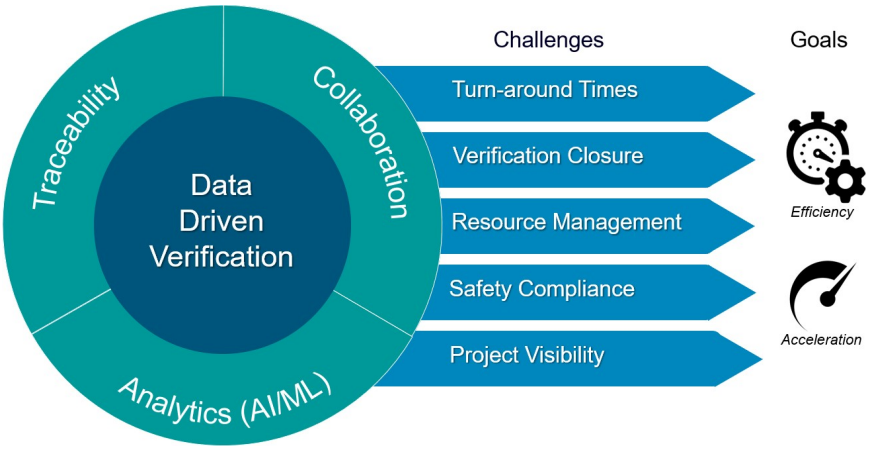


Questa Verification IQ

Questa Verification IQ integrates with Siemens' Polarion Requirements software for requirements management, resulting in a platform that automatically captures all data from every engine run across the life of a project.

The framework runs on a single server with one database instance to allow engineers to collaborate, with the verification and AI engines running on other servers, either in-house or in the cloud, for scalability.

The coverage data is gathered from logic simulation (Questa), Emulation and Prototyping (Veloce), AMS (Symphony), Formal (OneSpin), Static and FuSa. The ML feature analyzes all of this data in order to predict patterns and reveal any holes, point out root causes, then prescribe action to improve coverage. The ALM shown is Polarion from Siemens, although you could use another ALM, just like you can use your favorite verification engines.



Testplan Author

The tools in the Quest framework include a Testplan Author to couple to the Polarion PLM tool using web interfaces, with dynamic trace each tool's data without the need for synchronization. This can also link to Doors or the Siemens TeamCentre requirements tools. It features:

Built on the Questa Unified Coverage DataBase (UCDB) testplan infrastructure to seamlessly integrate within existing verification flows

Take advantage of verification process guidance to follow plan and requirements driven methodologies

Associate design requirements with testplan elements and RTL implementation to deliver full, safety compliant traceability

Testplan Author is the plan and requirements driven front-end, providing a collaborative testplan editor to capture the testplan that drives the verification process. Testplan Author is integrated tightly with the other Questa Verification IQ applications and supports coverage from multiple engines. It employs the power of lifecycle management through integration with ALM tools, such as Siemens Polarion™.

Using Open Services for Lifecycle Collaboration (OSLC), Testplan Author allows data to be both managed within each domain and dynamically available to each domain. It maintains a tight digital thread using requirements tools, like Polarion, and by providing complete traceability — from requirements to implementation and verification results — for safety compliance. OSLC provides the ability to have an agnostic solution to support other tools like Siemens Teamcenter® and IBM Engineering Requirements Management DOORS Next.

Regression Navigator

The Regression Navigator uses reinforcement learning to provide consistency and coherence with existing verification flows.

Regression Navigator is the collaborative front-end to the regression engine, providing visibility and controllability of regressions in a device and OS independent interface. Integrated tightly with Questa Verification Run Manager (VRM), it enables collaborative launching of regressions and the ability to analyze results across multiple regressions, from multiple users, across multiple projects.

Regression Navigator's power is not only due to the collaborative running of regressions but also in regression analysis. With all results stored in a single place, it is possible to gain insights into results across the history of a test. This allows analytics to be applied to data to provide information that would be hard to correlate any other way. Automated bucketing of failures is achieved by analyzing the results of each test in the regression and applying ML to improve debug turnaround time.

Coverage Analyzer

The Coverage Analyzer uses unsupervised learning to help teams find coverage holes quicker, adjust stimulus, and use heat maps, distribution graphs and network graphs.

Coverage Analyzer accelerates coverage closure by applying analytics to the coverage closure problem. By providing team based collaborative closure and sharing URLs, collaborative exclusions, and many more team-based features, it improves the understanding of the coverage model, supporting code coverage, functional coverage, and testplan coverage. Coverage Analyzer applies analytics to visualizations, giving insights into coverage in different dimensions, using heat maps, distribution graphs, network graphs, expression trees, and much more. Customers claim 3–5X faster closure with Coverage Analyzer compared to structural-based coverage analysis tools.

Verification Insight

The Verification Insight tool can extract any verification metric from the other tools to create a customisable dashboard.

Verification Insight provides the ability to build project metric dashboards.

A major benefit is its ability to capture verification data that would traditionally be lost in most user environments. These metrics can be viewed in various dashboards made up of customizable widgets. Although Verification Insight is tightly integrated with the other Questa Verification IQ applications, it can be used as a standalone tool. Verification Insight concentrates on providing visualizations and insights into what happened, and why, utilizing trending data, gauges, and cross analytics.

Verification Insight stores data throughout the verification process and across multiple projects, with features for historical data management. This allows ML based predictive and prescriptive analytics to be applied to accelerate and improve the verification process.

Questa Verification IQ vs Synopsys Designdash

Synopsys DesignDash is focused on ML for design data whereas Questa Verification IQ is focused on data driven verification using analytics, including ML, to accelerate verification closure, reduce turn-around times and provide maximum process efficiency. Questa Verification IQ provides applications needed for team-based collaborative verification management in a browser-based framework with centralized access to data.

Questa Verification IQ vs Cadence Verisium

Cadence Verisium focuses only on ML assisted Verification. In comparison Siemens Questa Verification IQ provides complete data driven verification solution powered by Analytics, Collaboration and Traceability. Verification Management is provided in a browser-based tool with applications built around Collaboration. Coverage Analyzer brings the industry's first collaborative coverage closure tool using analytical navigation assisted by ML. Question Verification IQ interfaces with Siemens Polarion using OSLC and provides a tight digital thread traceability with Application Lifecycle Management with no UI context change, bringing the power of ALM to hardware verification.

4. CADENCE INCISIVE VMANAGER

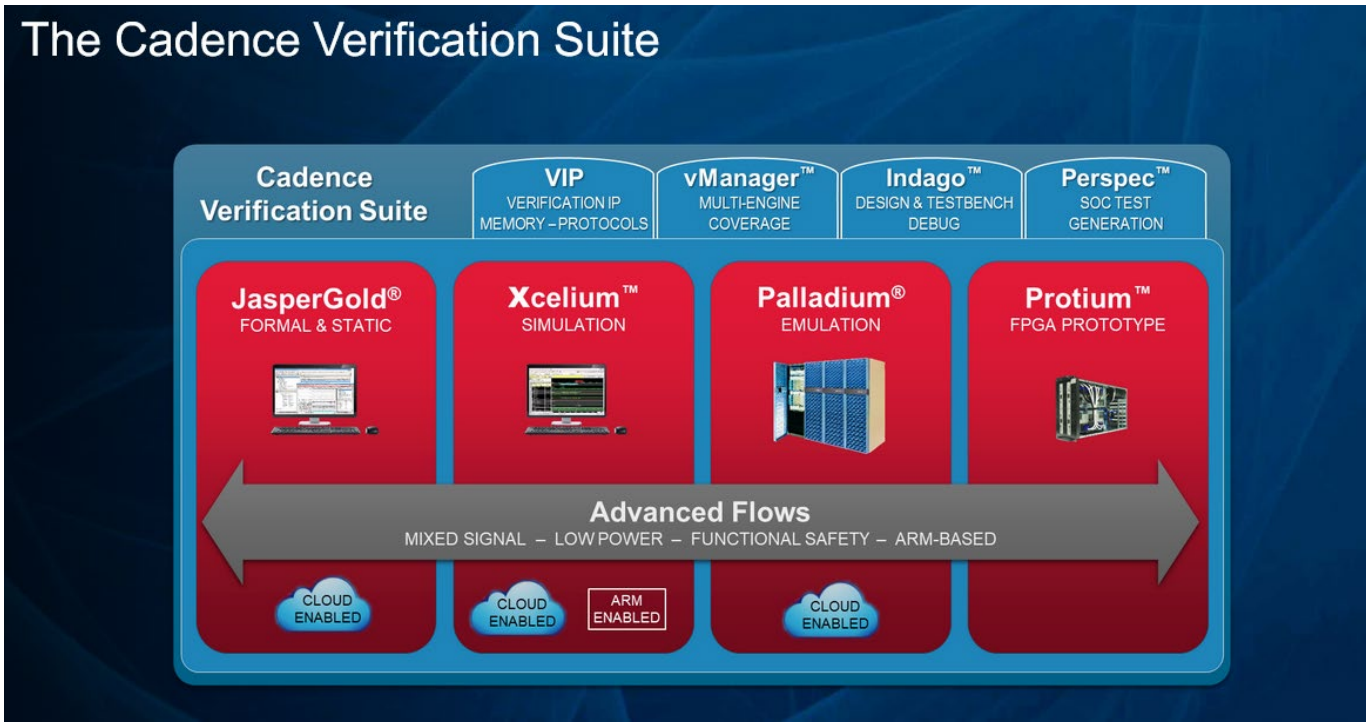
Cadence Incisive vManager for SoC verification planning and management

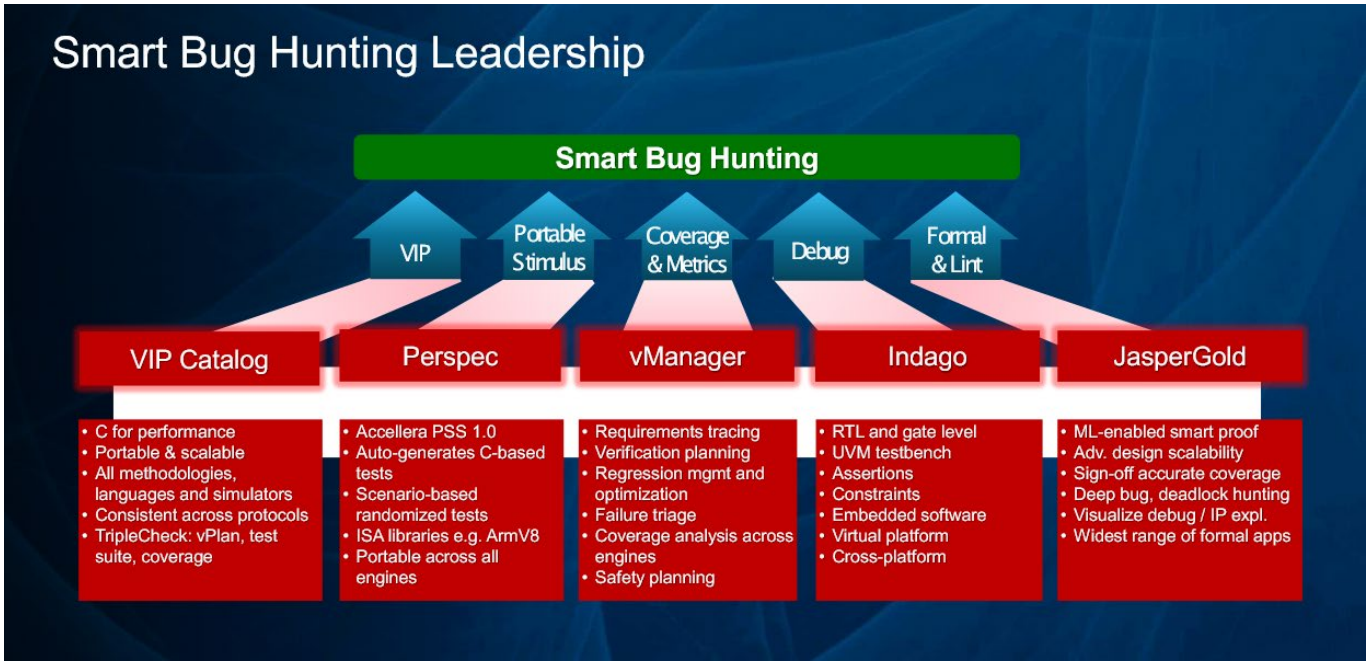
A powerful, scalable, and automated verification planning and management solution supporting multi-user, multi-engine, multi-projects, and multi-sites simultaneously

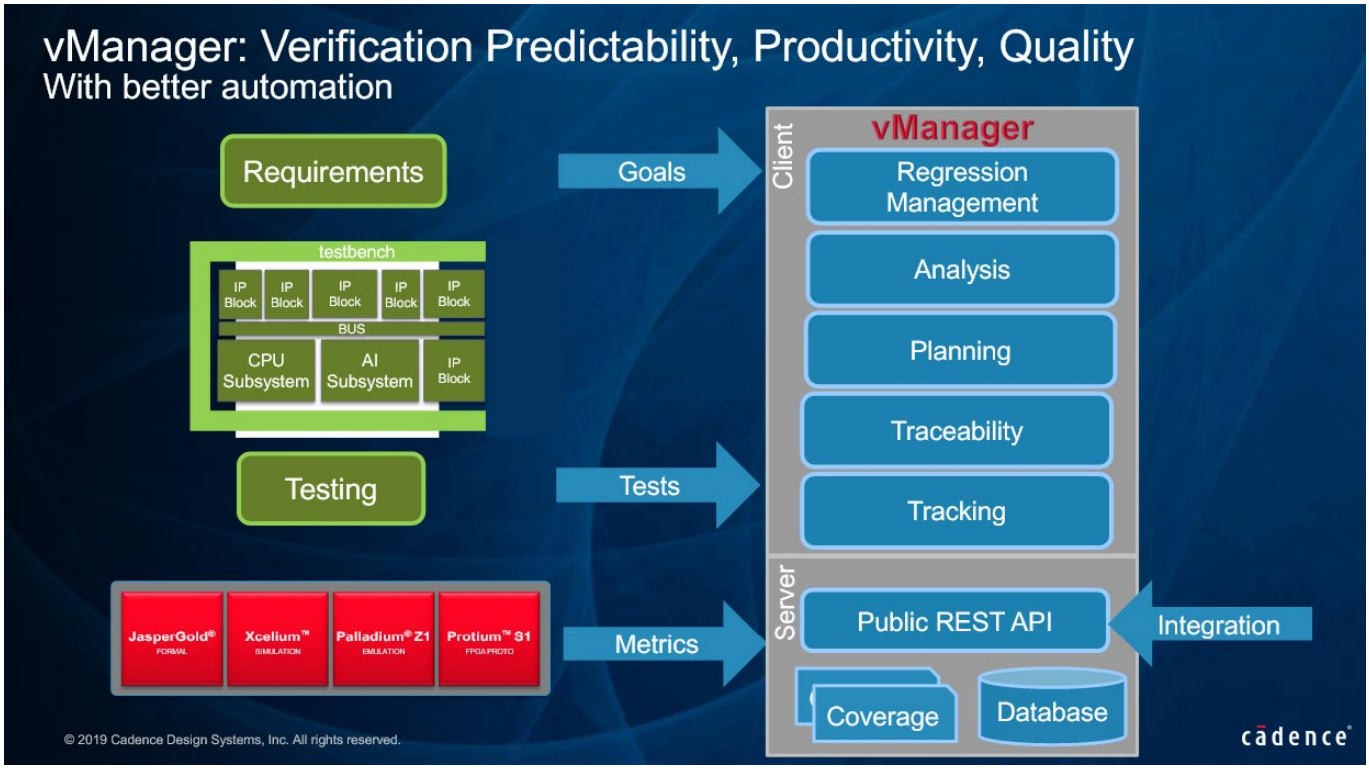
How to turn a verification plan into an executable specification that remains both a natural language document and becomes machine readable? This is followed by linking the verification plan to the verification environment and using it to control and monitor verification progress.

The Cadence® Incisive® Manager will illustrate the use of an executable verification plan.

Verification-Futures







vManager Verification Management

The Cadence® vManager™ Verification Management is a scalable, reliable, and feature-rich verification planning and management solution for pre- and post-silicon functional verification. This enterprise-class solution lets you connect people and processes in small to ultra-large verification projects. The vManager platform provides a flexible use model to automate tasks from simply executing regression tests to large-scale, data-driven, metric-driven verification (MDV) programs. With the vManager platform, users can increase productivity, predictability, and quality (PPQ) through improved automation, visibility, and traceability.

Incisive® vManager™ solution, a verification planning and management solution, is enabled by client/server technology to address the growing verification closure challenge driven by increasing design size and complexity. The Incisive vManager solution, with its metric-driven verification (MDV) methodology, improves verification

productivity by 2X or greater over traditional methods by combining executable verification plans, coverage optimization techniques, collaborative management utilities, deep failure and coverage analysis, and clear visibility to see when to shift resources.

Part of the Cadence® Incisive functional verification platform, the Incisive vManager solution utilizes commercial SQL database technology and enables broad scalability from small intellectual property (IP) projects to gigascale system-on-chip (SoC) designs. Additionally, to ensure SoC developers have a consistent methodology for design quality enhancements, the Incisive vManager solution supports several MDV extensions addressing applications including acceleration, low power and mixed-signal verification.

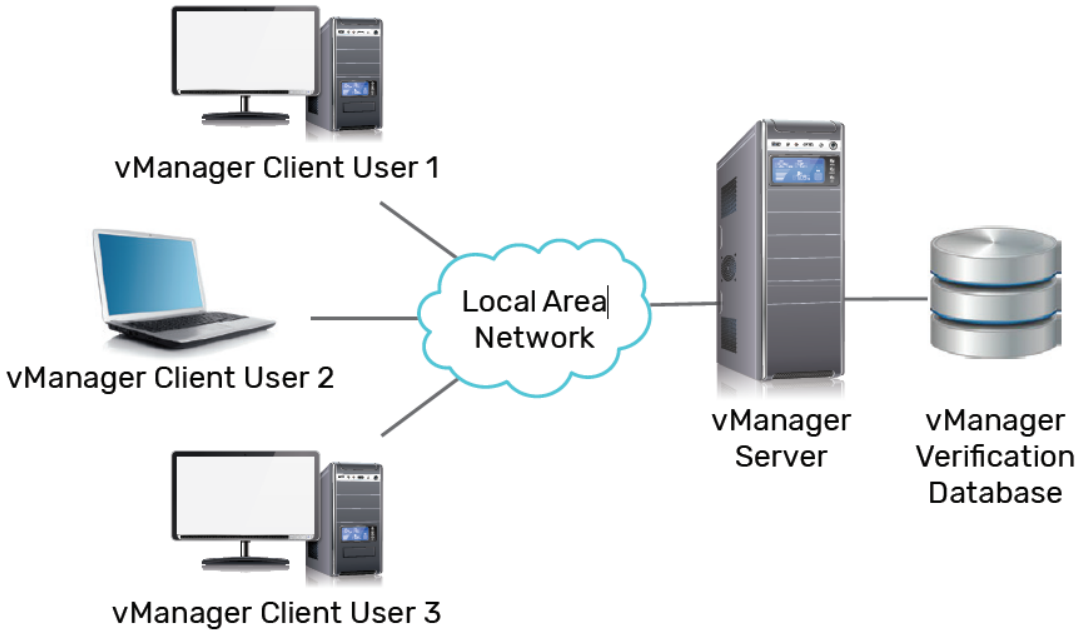
Overview

In 2004, the vManager product pioneered verification planning and management with the industry's first commercial solution to automate the end-to-end management of complex verification projects—from goal setting to closure. Now in its fourth generation, with the introduction of the High Availability feature, the vManager platform provides capabilities tailored for verification on top of scalable and robust data management technology.

The vManager Verification Management propels verification from a simulation-centric individual tool activity to team-based verification productivity, spans multiple disciplines and geographies, and is architected for expansion into the cloud.

The vManager platform automates the verification process at the block, chip, system, and project level, automating management of activities from spec to execution to signoff. Easy-to-adopt regression management and failure triage features improve productivity, eliminating time-consuming and tedious data organization tasks. Built-in automation for creating reports, emailing results, and generating debug data provides even further productivity, ensuring engineering resources are spent solving real design and verification problems.

Users can enable closed-loop verification by utilizing the verification plan (vPlan) capabilities of the vManager platform. A verification-specific set of authoring capabilities includes directly connecting the vPlan to spec documents as well as point-and-click mapping of verification metrics to device features. The vPlan can be annotated with data generated by Cadence Xcelium™ Logic Simulator as well as the Cadence JasperGold® Formal Verification Platform, Cadence Palladium™ Emulation Platform, and Cadence Protium™ Prototyping Platforms. The vManager platform also boasts connection with OpsHub Integration Manager, a commercial application lifecycle management (ALM) solution, enabling it to synchronize defect and requirements data with third-party systems like JIRA, JAMA, and Doors. Using this approach, users can eliminate subjectivity associated with complex and disparate sets of verification, requirements, and defect data.



vManager client and server components

Key Benefits

Predictability

Drives the complete verification process from planning to closure.
Enables reporting and dashboards for management-level tracking of defined project milestones.

Productivity

Analyzes and ranks tests to improve regression efficiency and optimize farm utilization.
Shortens overall failure turnaround time to optimize bug-closure productivity.
Reduces overall regression environment maintenance by automating execution and results aggregation within and across teams, farms, and sites.

Quality

Provides objective feedback about verification completeness against the plan, enabling data-driven schedule and resource decisions.
Aggregates into a single intuitive interface verification closure metric across the Xcelium, JasperGold, Palladium, and Protium platforms.

Features

Incisive vManager Key Features

Multi-user support: Allows unlimited simultaneous users for improved collaboration and better visibility into team-based verification

Multi-engine support: Operates seamlessly with Incisive Enterprise Simulator, Incisive Formal Verifier and Palladium® XP Verification Computing Platform

Multi-project capability: Enables multiple projects to be managed independently within the same environment—an industry first. Users can view project status, progress over time, and key metrics enabling verification signoff

Multi-analysis feature: With the fully integrated Incisive Metrics Center, users can analyze coverage, test failures, perform failure triage, create and analyze executable plans, find coverage holes and design problems, all to determine focus areas to complete verification

High Availability, Multi-Region, and Cloud

The High Availability feature enhances the vManager platform's infrastructure, utilizing a distributed processing model to increase scalability, reliability, and maintainability. The vManager platform can now utilize compute resources across the farm, balancing load and adding significant scale and robustness, while reducing overall dedicated infrastructure requirements.

The distributed processing model provided by this new feature of the vManager platform also enables powerful capabilities for managing verification projects across multiple geographies and into the cloud. A single vManager server can now service geographically diverse projects, efficiently connecting across the wide area to enable verification users to view the complete set of verification data, regardless of where or how the data is generated.

Team-Based Verification Productivity

The vManager platform provided the first, and continues to provide most robust and scalable, platform to enable collaboration across verification teams. The High Availability feature of the vManager platform expands this support to include teams across multiple sites. Verification teams can direct and track regressions, with the vManager platform providing a single, holistic view of what is being executed and what results have been achieved across the entire team.

The vManager platform provides a single portal, combining data across multiple sites, to allow for real-time collaboration between IP and SoC design and verification teams. Results are available in real time among all project members, with advanced sorting and filtering capabilities bringing the most relevant data to the forefront.

Verification Planning

Feature-based verification plan creation and a unified interface to span verification engines helps optimize productivity of the verification team. The vManager platform has vPlan authoring capabilities to help verification engineers capture the verification plan efficiently, with a verification-focused, hierarchical user interface. The vManager platform connects to both specification documents via the PDF Spec Annotation feature, as well as requirements management systems via our partnership with OpsHub. Point-and-click mapping to connect the verification plan to the verification environment ensures the plan will provide objective feedback based on real verification data throughout the project.

vPlans in the vManager platform are built for flexibility and reuse, offering powerful parameterization reference features. Hierarchical vPlans, reused from IP to SoC level, are widely used by vManager customers. vPlan perspectives also enable the user to focus on the portions of the vPlan that are most relevant to individual verification team members, and the hierarchical nature of the vPlan means that project leaders always have the top-level view of verification progress.

Coverage Closure

The vManager platform provides the industry's most comprehensive set of code and functional coverage analysis and closure features. The tool automatically manages coverage merge and management, dynamically re-merging and re-calculating as the user selects different sets of results, and enabling powerful what-if analysis vManager Verification Management capabilities. The vManager platform provides a powerful coverage waiver and refinement flow, including features like Smart Refinements for refinement automation, flexible refinement file management, and refinement re-mapping and reuse.

The vManager platform also integrates with the JasperGold Coverage Unreachability App (UNR) to provide further automation to the coverage closure flow. The UNR App utilizes JasperGold formal verification's industry-leading structural analysis capabilities to determine which coverage metrics in the DUT are impossible to hit, reducing the number of coverage holes that the user must analyze and either address or waive.

Multi-Engine MDV

Multi-engine MDV lets the user apply MDV concepts across the entire set of Cadence System Design and Verification Suite engines and technologies. This enables the user to merge and combine metrics to roll up meaningful results to the top level. When metrics and results can be combined across the entire verification effort, users are free to choose the best platform and engine for each verification task, while maintaining the ability to track the results against the plan. This improves overall productivity and efficiency of the verification team, while improving predictability of the overall verification effort.

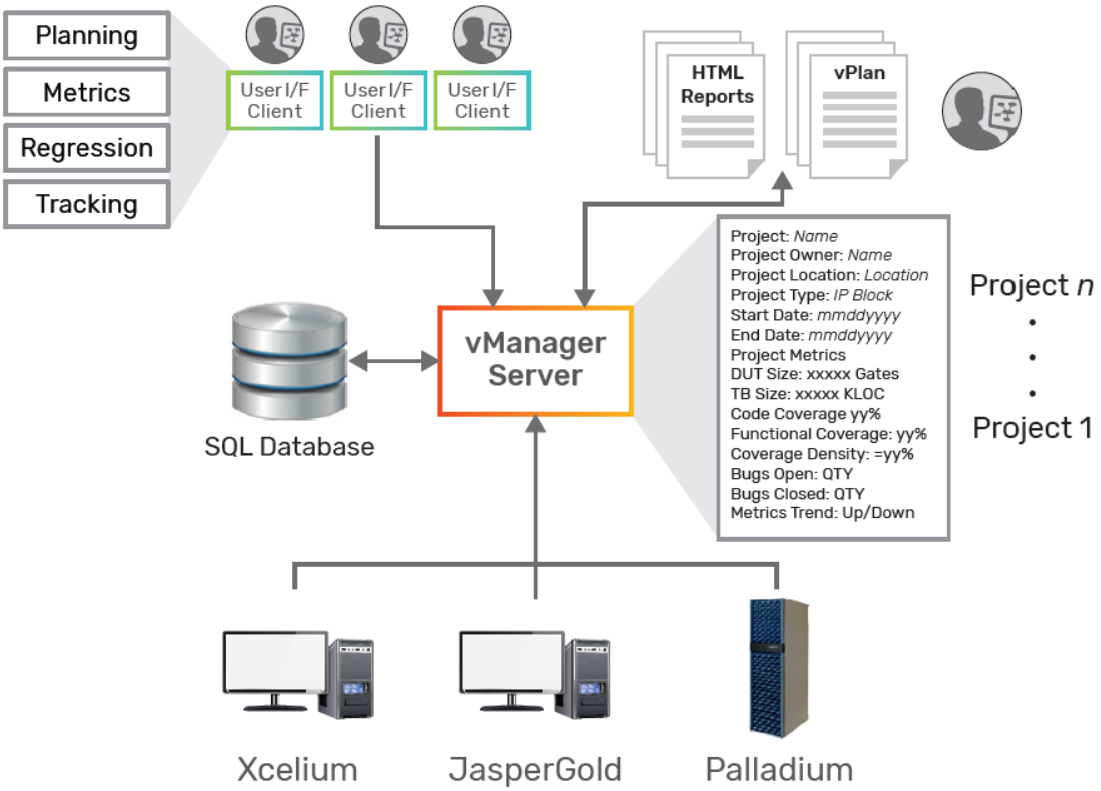
In addition to rolling up data from Xcelium simulation, JasperGold formal verification, Palladium emulation, and Protium prototyping, the vManager platform has added multi-engine MDV capabilities for the Cadence Perspec™ System Verifier as well as integration for analog simulation metrics via Cadence Virtuoso® ADE Verifier. The vManager platform also provides extensive analysis capabilities across various engines, including domain-specific features like specialized regression optimization features for the Perspec System Verifier.

Failure Analysis and Triage

The vManager platform simplifies the overall debug effort and shortens failure debug time. Users can separate design failures from environment and IT issues, sorting and grouping failures by failure signature and type, to enable easy assignment and action. Quickly identify the least costly path to reproducing a failure, and automatically generate the debug information, keeping verification engineers focused on debugging rather than data management.

User Interface

The vManager multi-window graphical user interface (GUI) takes the guesswork out of which views are needed by organizing default views by activities. Speed up searching, sorting, and filtering with the GUI's detachable panes and SQL-based tables. Improve productivity with customized views and fields, an intuitive forward-and-backward one-click history, hyperlinked source-code windows, and detailed metric-analysis windows. HTML and CSV reports can be easily generated and customized. Moving from single-run coverage analysis with the Cadence Integrated Metrics Center (IMC) to regression-level analysis with the vManager platform is seamless due to the platforms sharing a coverage analysis user interface.



vManager highlights

Components of Metric-Driven Verification Integrated Metrics Center

f Provides single-run coverage analysis environment for Cadence verification engines.
f Automates coverage viewing, merging, and analysis.
f Supports block, toggle, expression, FSM, and functional coverage from all design and verification languages, as well as PSL and SVA assertions.
f For more information, see the Integrated Metrics Center technical brief.
vManager Client
f Presents unified user interface for verification engineers, designers, and managers.
f Organizes activities for the key functions of planning, analysis, regression, and tracking.
f Enables execution of all functions from interactive GUI, batch command line interface, and programmatic API.
f Operates over both local (LAN) and wide (WAN) area networks.
f Embeds all aspects of the IMC into the vManager Analysis

interface.
f Provides comprehensive verification plan authoring to build executable vPlans.
Figure 2: vManager highlights
Xcelium JasperGold Palladium
HTML vPlan
User I/F Reports
Client
Planning
Metrics
Regression
Tracking
vManager
Server
SQL Database
Project: Name
Project Owner: Name
Project Location: Location
Project Type: IP Block
Start Date: mmddyyyy
End Date: mmddyyyy
Project Metrics
DUT Size: xxxxx Gates
TB Size: xxxxx KLOC
Code Coverage yy%
Functional Coverage: yy%
Coverage Density: =yy%
Bugs Open: QTY
Bugs Closed: QTY
Metrics Trend: Up/Down
Project n
•
•
•
Project 1
User I/F
Client
User I/F
Client
Cadence is a pivotal leader in electronic design and computational expertise, using its Intelligent System Design strategy to turn design concepts into reality. Cadence customers are the world's most creative and innovative companies, delivering extraordinary electronic products from chips to boards to systems for the most dynamic market applications. www.cadence.com

vManager Verification Management
f Captures and tracks projects metrics, which can populate web dashboards.
f Supports regular and complex express in SQL-style data

search, sort, and filtering operations.
f Generates customizable, interactive HTML reports for multiple types of results and metrics.
vManager Server
f Provides multi-user- and multi-project-backed services, structure, and support.
f Manages all analysis calculations and user views.
f High Availability feature provides reliability, scalability, and multi-region capabilities.
f Establishes user-defined security and authorization rules for connecting users.
f Includes commercial, robust, fully embedded SQL database.
f Provides web-based interfaces for triage, management dashboards, and web-based regression interface.
f Delivers vManager API connectivity for programmatic access to verification data and vManager features, including ALM integration.

The vPlan within the vManager platform - Metric-Driven Verification Signoff

Functional verification entails changing the state of a logic design and measuring that the response generated by the design is correct. Verification environments change the state of designs by driving stimulus in the form of directed or constrained random inputs. But when are you done? In verification, signoff is the process of defining criteria, and objectively measuring metrics against the criteria as the development progresses, until they match.

Metric Driven Signoff

Metric-driven signoff is defined as the final stage of all design and functional tests prior to physical implementation or coinciding with tapeout. The signoff stage includes all milestones and metrics from previous stages, plus adds all final structural checks such as clocking and low power. This final signoff stage will add gate-level tests, state

machine (FSM), clock (CDC), synthesis, X-prop, failure mode (fault) testing, pre-silicon power measurements, and more as required. The vPlan and vManager platform can be used to collect and organize this data and enable a comprehensive Metric-Driven Signoff environment, with automatic data collection, manual checklists, and manual test result entry. Tools used at this stage may include:

Xcelium Parallel Simulator - Multi-core for GLS, low power, and built-in self test (BIST)
Genus™ Synthesis solution – Synthesis solution
Conformal® technologies – LEC, CDC, and low-power checks
Joules™ RTL Power Solution – RTL power estimations
vManager Metric-Driven Signoff Platform – vPlan, regressions, metrics, coverage

Metric-Driven Signoff is a unique Cadence® methodology and technology for measuring and signing off on the design and verification metrics used during the many milestones typical in any integrated circuit (IC) development. While milestones and metrics vary by design type and end application, the final verification signoff will at, a minimum, contain the criteria and metrics within a flexible, human-readable, user-defined organizational structure. Automated data collection, project tracking, dashboards, and in-depth report techniques are mandatory elements to eliminate subjectivity, allowing engineers to spend more time on verification and less time manually collecting and organizing data.

vManager, now called **Enterprise Manager**, is a product for automating functional verification from creating a verification plan to reaching verification closure. The vPlan is a file format which is read by Enterprise Manager, and contains the functional verification intent of a logic design project. This document is relatively open format, and can contain all manner of text to describe the project, plus the formatted information needed to specify the metrics of verification. Enterprise Manager helps improve project predictability by motivating creation of a higher quality verification plan, monitoring detailed progress, measures against metrics of verification completion, and increases productivity of individuals and the overall team by automating the common, high value tasks of verification.
vPlan

Enterprise Planner, a licensed feature of Incisive Enterprise Manager

If you're looking for a really easy way to create an executable verification plan, you need take a look at Enterprise Planner, which is a licensed feature of Incisive Enterprise Manager. Enterprise Planner allows you to easily create, edit and maintain verification plans, either starting from scratch, or by linking and tracking to your functional specifications. What's really nice is being able to incrementally create a plan that you execute and see immediate results, or alternatively having multiple people to contribute to various sections of a comprehensive plan concurrently - both approaches are used extensively by users of Enterprise Planner.

For those of you who are not familiar with Enterprise Planner, it's basically a user friendly and intuitive GUI based application used to create executable functional verification plans in a format that can be consumed by Enterprise Manager. What makes it special is that users can take their specifications and connect important areas back to the plan for tracking purposes, allowing Enterprise Planner to detect specification changes. Enterprise Planner can also map the various metrics from functional coverage, code coverage, assertions and dedicated tests back to the plan. From here, Enterprise Manager, our core technology for driving Metric Driven Verification, can read in the plan and report real time results with color bars and percentages, so you can see the immediate status of your verification project. Enterprise Planner and Enterprise Manager are both proven products, but the new updates have incorporated a number of ease of use features. This latest release is included as part of the INCISIVE software release.

Enterprise Planner makes connecting with the specification a lot easier by allowing users to develop their plans in

both Windows and Linux. Simply load the specification in Enterprise Planner (EP) and create verification plan (vPlan) elements based on important areas that you would like to see verified. It's as simple as pulling up the spec in EP and highlighting with the mouse the areas in the spec you think are important. From here you can create a section or planned coverage element that will track changes with future versions of the spec. Detecting spec changes became easier as well by allowing you to update your spec and seeing the difference in a side by side comparison between the old spec and the new spec.

You are able to easily map any combination of metrics to the vPlan sections from Cadence's wide variety of solutions such as functional coverage, code coverage and assertion coverage and checks to name a few. Cadence's has taken coverage mapping a bit further by showing you, graphically, what has been mapped already and what still needs to be mapped allowing you to see what coverage elements are not included in the plan.

As with developing reusable verification components you will also want to develop reusable verification plans as well and for similar reasons. Cadence's planning technology allows you to easily create and use verification plans at variable hierarchies or to easily build verification plans that reference other verification plans. Cadence has made developing reusable verification plans easier than before. Whether you're someone who is developing the reusable plans or someone who is using them, you will find that our new push button interface makes it a snap to do.

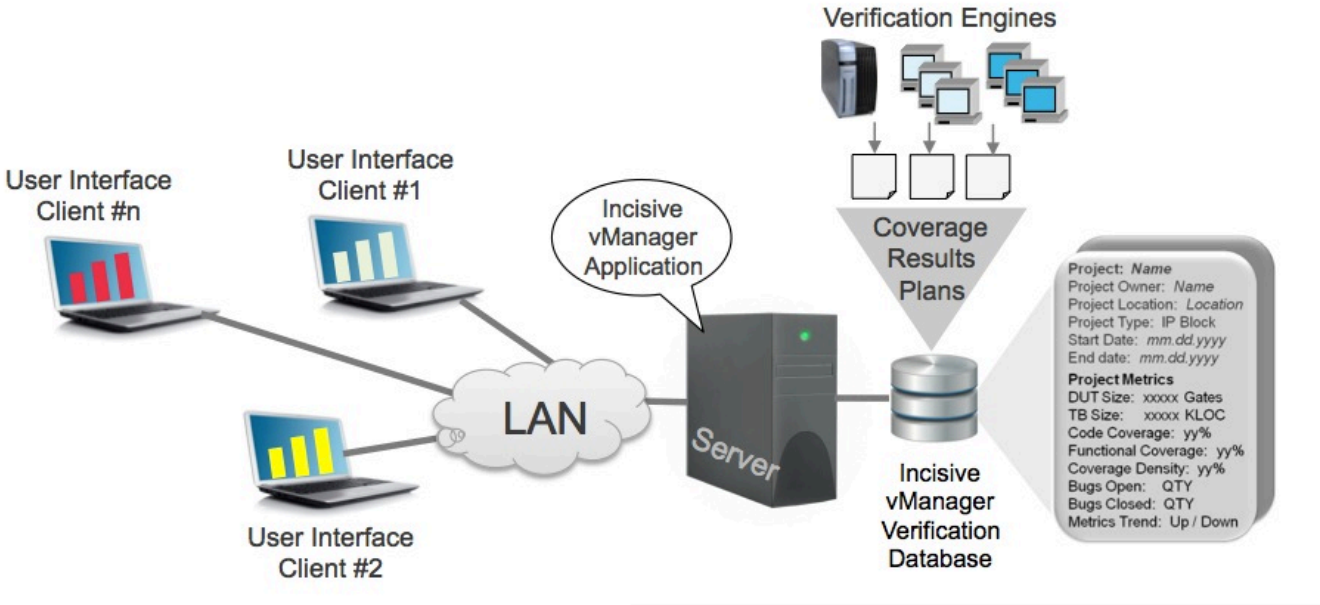
If you have not adopted a Metric Driven Verification approach using an executable plan there's never been a better time to easily gain visibility and control of your verification process and shave months off of manual verification plan creation. If you have adopted an MDV approach but are still creating verification plans without Enterprise Planner you will find that using Enterprise Planner will allow you to create a plan within hours instead of days or weeks and make verification planning much more enjoyable.

Incisive vManager SoC Functional Verification Planning and Management

Big SoC designs typically break existing EDA tools and old methodologies, which then give rise to new EDA tools and methodologies out of necessity. Such is the case with the daunting task of verification planning and management where terabytes of data have simply swamped older EDA tools, making them unpleasant and ineffective to use.

This is a product area familiar to Cadence users with a 10 year history of the Incisive Enterprise Manager(IEM) tool. The new tool is called Incisive vManager and it was designed to handle the biggest SoC verification tasks by using:

A client-server approach
Sophisticated verification management
A scalable, database-driven technique



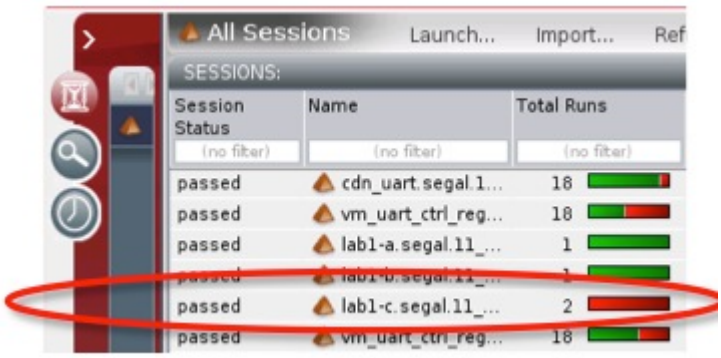
With the old way you had to comb through reams of data to see if you can optimize your verification, while with the new way you collaborate with all team members throughout the design and verification process, where everyone has easy access to the progress. Benefits of using the new approach include about a 2X improvement in functional verification efforts, once you get fully trained.

Reducing verification times and improving coverage are a big deal because the typical SoC at 40nm had a $38M verification cost, from data supplied by IBS 2013. For projects using 20nm, that verification cost can be $100M.

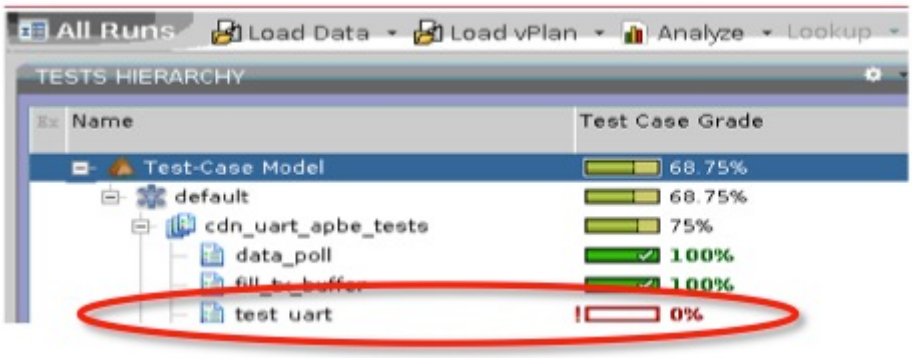
A project manager really wants to know a few things:

What is my schedule until DV is complete?
What are my costs to reach tape out?
Are there any functional bugs that will cause a re-spin or recall?

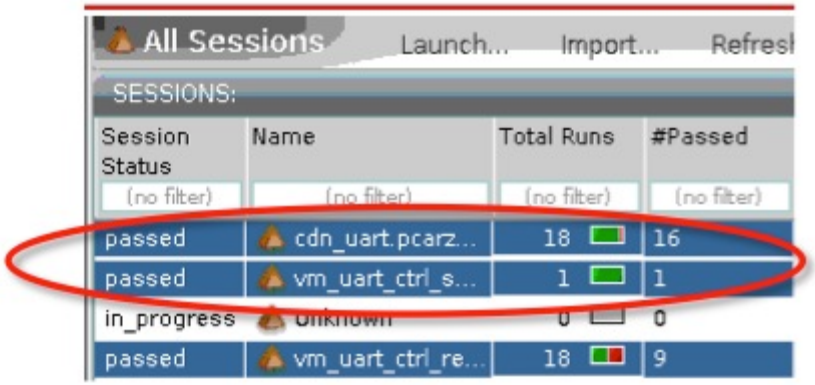
With vManager you can improve schedule predictability, verification productivity and design quality. Here's a closer look at how this happens. By having all of your functional verification metrics visible in a GUI you can work on the most critical failures first:



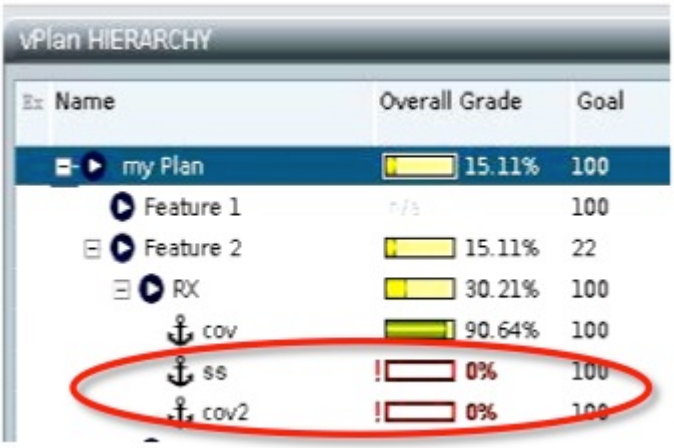
If there's a block on your design with a low test grade, then the manager can shift verification resources to get caught up:



Failure analysis determines which failures are the same, and helps identify only the most critical, thereby eliminating redundant cycles:



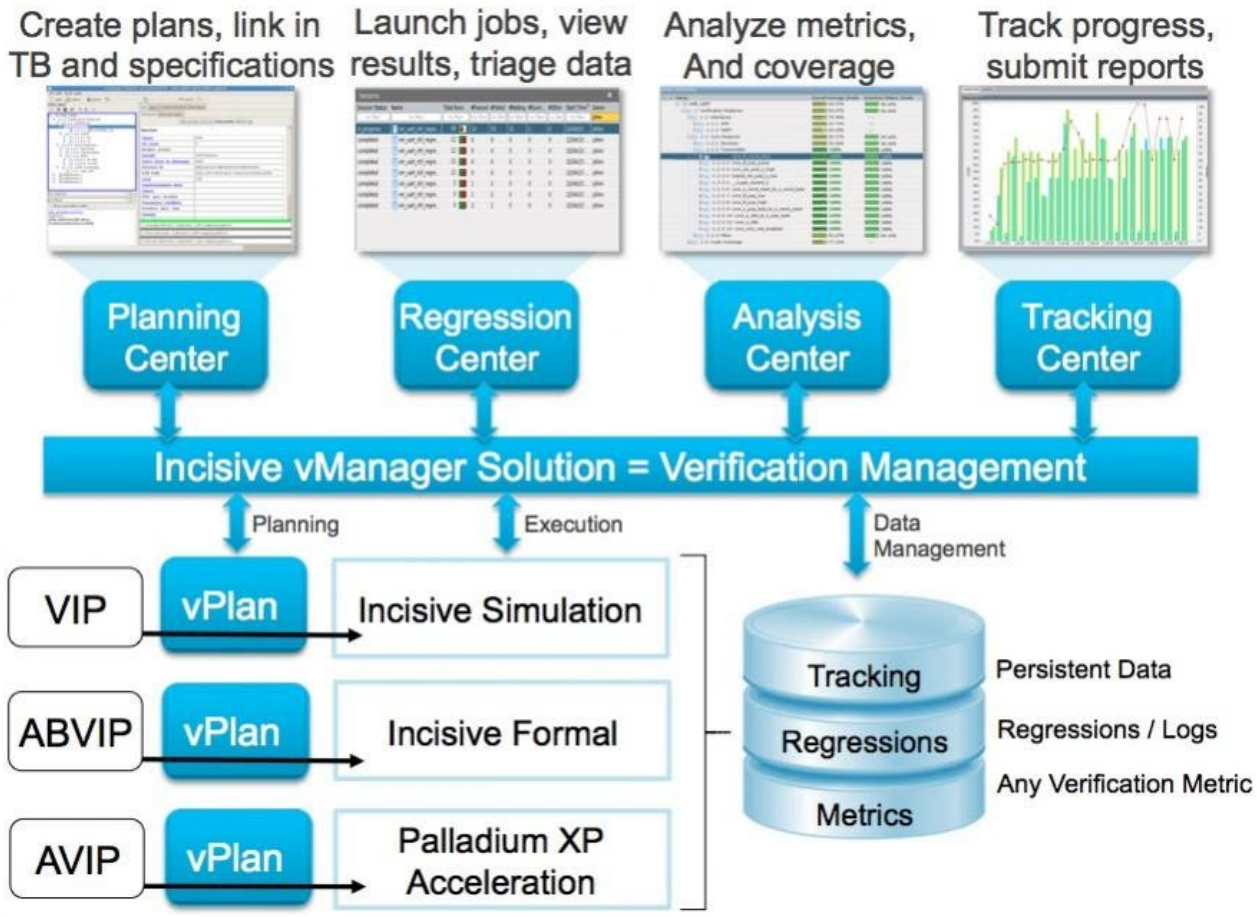
Finally, for optimizing your verification plan there is benefit to finding precisely where your coverage holes are, so that you are quickly aware and can take early action:

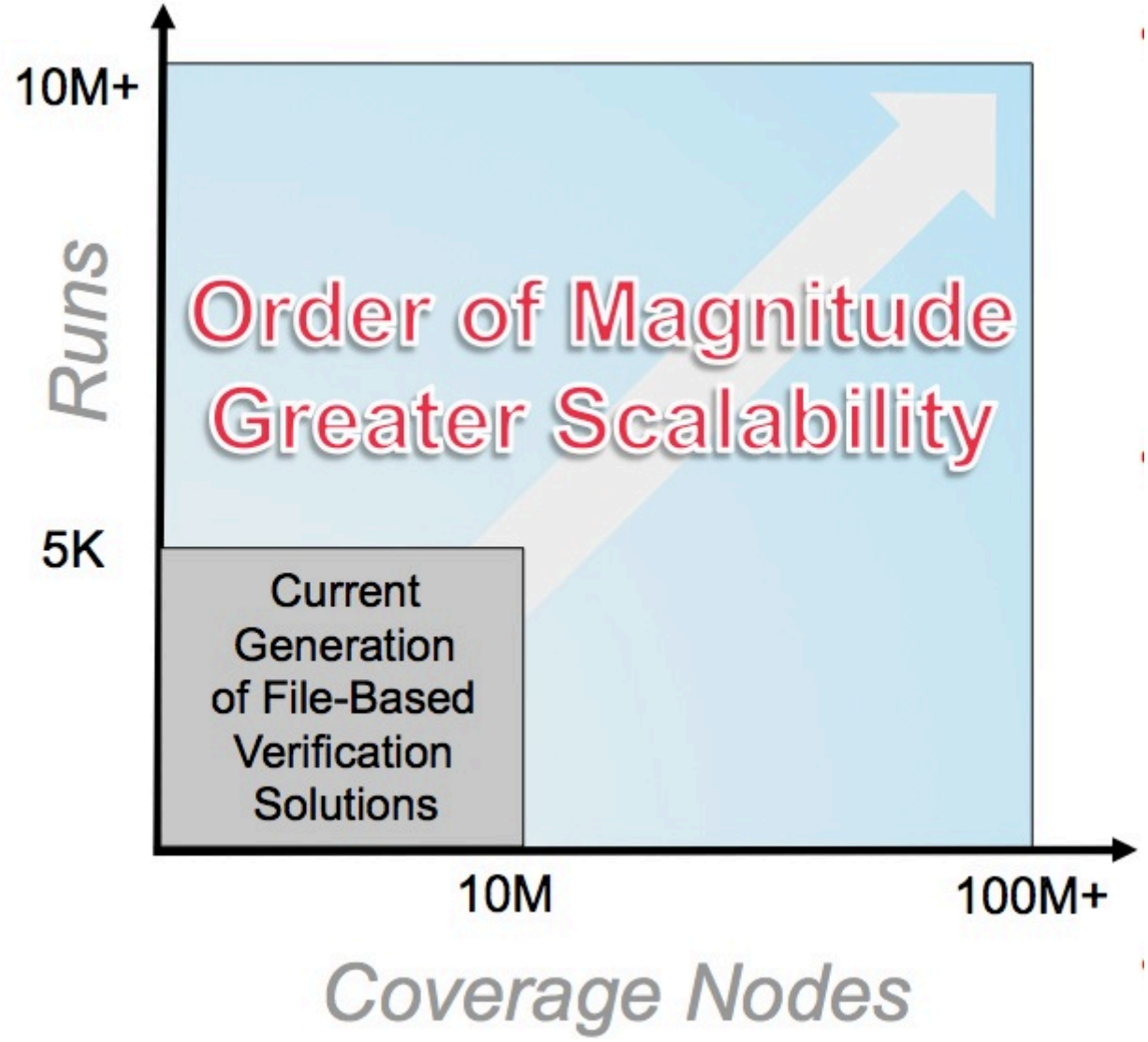


Verification productivity improves by:

Up to 30% using reporting automation and closure automation
About 25% better compute farm utilization with MDV (Metric Driven Verification) versus directed testing
Up to 10X improvement in bug discovery with MDV versus directed testing
A 60% reduction in verification time with MDV

The vManager tool flow has the following components:


Create plans, link in TB and specifications
Launch jobs, view results, triage data
Analyze metrics, And coverage
Track progress, submit reports
Planning Center
Regression Center
Analysis Center
Tracking Center
Incisive vManager Solution = Verification Management
Planning
Execution
Data Management
VIP
vPlan
Incisive Simulation
ABVIP
vPlan
Incisive Formal
AVIP
vPlan
Palladium XP Acceleration
Tracking
Regressions
Metrics
Persistent Data
Regressions / Logs
Any Verification Metric



vManager - Project Verification Planning for Analog Designs

Successful projects leverage the investment in comprehensive methodology and resource planning, covering design and analysis flows – that planning effort is especially important for functional verification.

Advanced system designs – especially automotive systems – incorporate a rich mix of digital and analog components. The verification strategy for analog components has several unique facets. Whereas digital verification explores model functionality, analog simulations focus on performance measures. Digital verification is primarily based on batch job regressions, whereas analog blocks are evaluated using a mix of interactive and batch simulations. Analog block verification is typically the direct responsibility of the design engineer, working within a tool "cockpit". This is fundamentally different from digital verification, which primarily uses a mix of (software and hardware accelerated) simulation and formal methods.

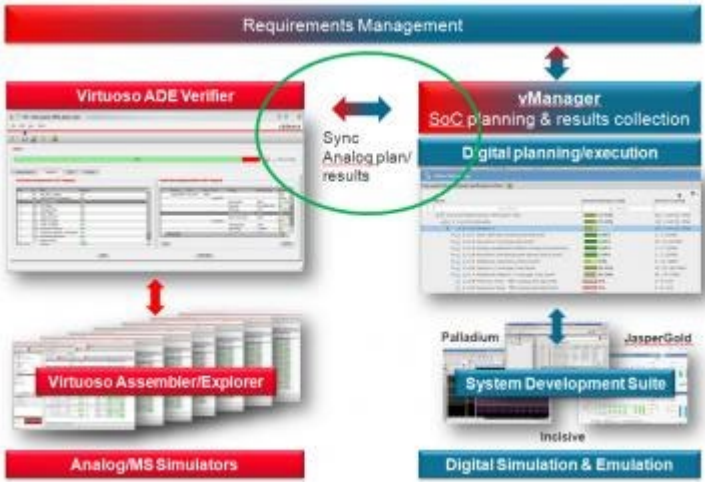
Yet, the nature of advanced (mission-critical) system applications necessitates that verification planning extends to the analog design environment, as well – i.e., traceability to performance requirements, analysis across numerous operating modes, and sub-component integration. In short, analog block verification planning is as crucial to

overall project success and compliance to product development standards as digital verification planning.

Cadence developed the vManager product, to address the need for an overall 'umbrella' to the overall verification planning and tracking activity. This top-level environment provides the necessary links between specifications and testbenches. vManager incorporates features to submit functional regressions, analyze failing tests, and rerun individual tests with additional signal tracing.

As verification utilizes an incremental, continuous integration flow, vManager works with source control management tools and model history data to optimize regressions – if model components are unchanged, associated regression testbenches need not be submitted. The vManager traceability to functional requirements provides a full coverage-driven view of the project verification status.

A figure depicting vManager utilizing data from Cadence functional simulation products in appended below. The digital simulation (and formal) tools provide functional verification data to vManager (on the right-hand side of the figure), whereas the left-hand side of the figure illustrates the link to analog verification.



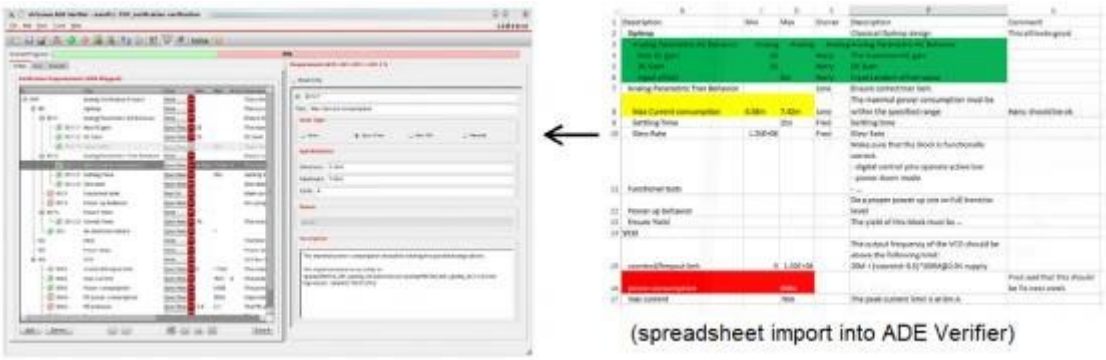
(Also, please see an earlier Semiwiki article on vManager – link.)

The Analog Design Environment product suite is the heart and soul and analog block design and simulation. Due to the unique nature of ADE compared to functional verification, a bridge to the overall vManager environment was needed, providing a consistent project status view. The ADE Verifier feature is the 'Golden Gate Bridge' between the functional verification environment(of Marin County) and the ADE simulation environment(the city of San Francisco).

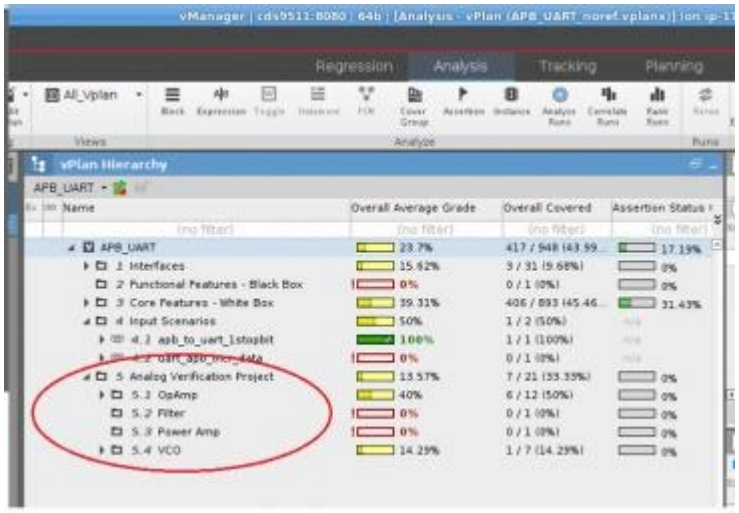
Analog block designs have traditionally been developed with rather ad hoc methodologies toward verification planning. ADE Verifier helps the designer to formalize analog performance requirements, with pass/fail measurement ranges for block specifications. Design groups can import their verification plan from other sources to help getting started with ADE Verifier.

As analog blocks commonly evolve in a mix of top-down and bottom-up integration styles, leveraging the hierarchical design views in ADE was key.

The figures below illustrates the ADV Verifier interface, and the import of an initial verification plan from an (unstructured) spreadsheet. The verification plan hierarchy is also highlighted.

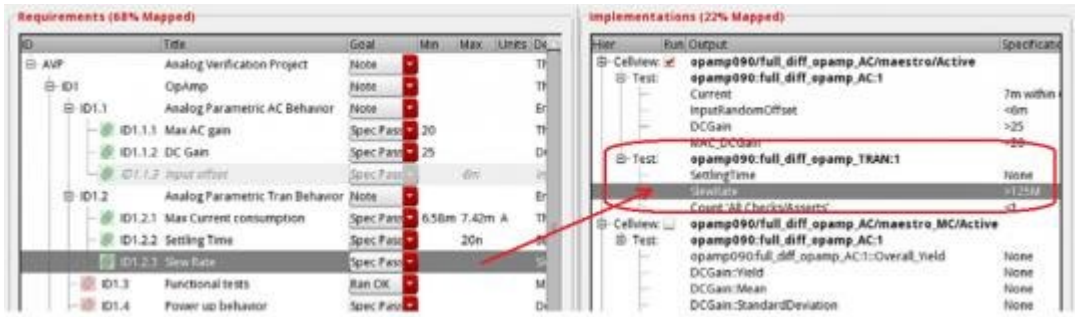


(spreadsheet import into ADE Verifier)



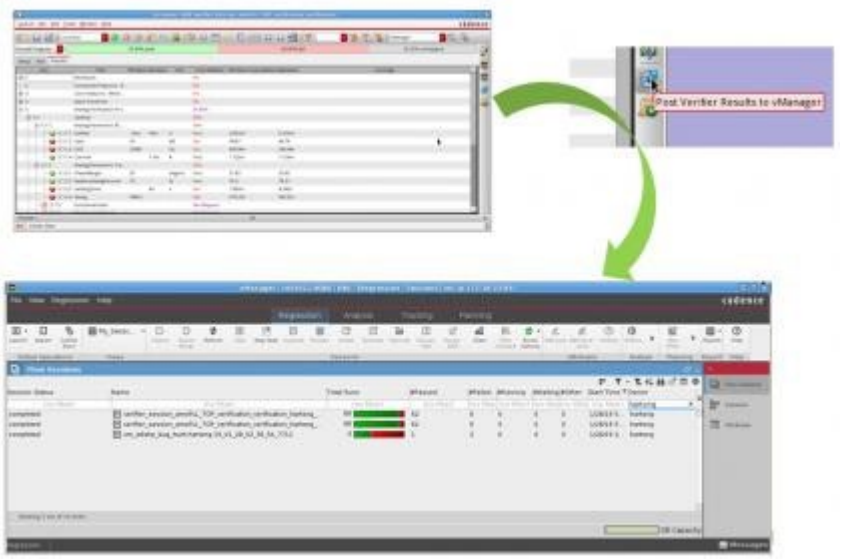
After the block specifications are captured, the key step in ADE Verifier is 'mapping' the requirement to a design verification testbench. The(one-to-one or one-to-many) mapping indicates that the product requirement will be fulfilled by the selected testbench measure(s).

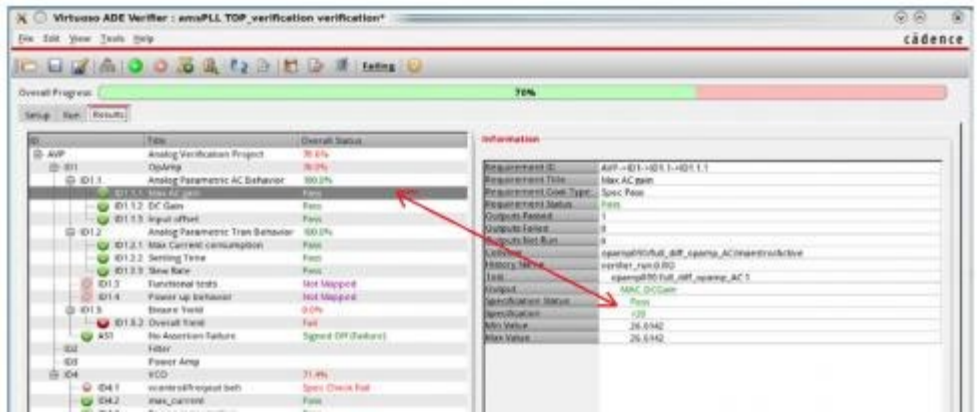
The figure below illustrates the ADE Verifier interface for mapping requirements to simulation measures – note that the tree structure of the design hierarchy is part of the mapping linkage. ADE Verifier also tracks the simulation requirements across environment conditions (PVT corners) and operating modes.



An additional benefit of adopting ADE Verifier for analog block verification planning is the direct applicability to IP reuse. Most analog designs evolve through multiple variants – a verification plan for an existing design provides a starting point for the development of a block revision.

As mentioned above, a key benefit of verification planning is a concise, consistent, and accurate view of the block status. As ADE Verifier is bridges to the vManager database, project-wide reporting status is readily available – the figures below illustrate the upload of ADE Verifier data to vManager, and the detailed results information available for each testbench.





Structured verification planning has become de rigueur for project managers, providing traceability between system specifications and the associated testbench(es). The unique nature of analog block design requires a specific (cockpit-based, metric-driven) approach to verification planning, as exemplified by ADE Verifier. Further, a "bridge" is required to consolidate digital (functional) and analog (performance) verification status, as embodied in vManager.

Getting Started with vManager

vManager facilitates the basic, every-day tasks of the verification process. It provides you with a robust and interactive GUI using which you can:

■ Launch thousands of runs in a single session using a distributed resource manager
■ Facilitate failure analysis by automatically extracting, filtering, and grouping key information from all log files in one or more sessions
■ Launch the rerun of failed tests and manage failure scenarios
■ Facilitate metric analysis by presenting data gathered by multiple agents (Incisive Simulator and Specman) in the same window

Launching vManager Client

You can launch vManager client in any of the following modes:
■ Regression Center Launcher (RCL) client — Supports only the session management and metrics analysis functionality.
■ Full client — Supports complete vManager functionality.

Launching the vManager RCL Client

The RCL client mode supports only the session management and the metrics analysis functionality. The RCL client uses the same license as IMC, which is the Affirma_sim_analysis_env license.

The following command line launches vManager RCL client in GUI mode:

vmanager -regr
or
vmanager -regrcenter

Note: The -regr and -regrcenter options are the same (-regr is a shortcut). For more

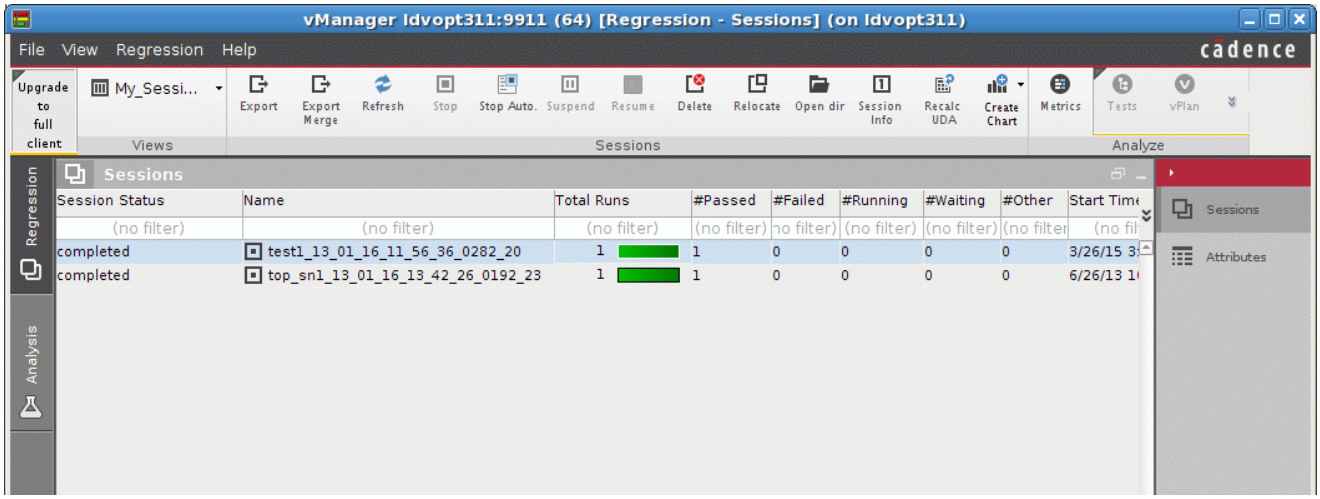
The above commands launch vManager RCL client, and displays the vManager Welcome screen, as shown in the Figure below.

vManager Welcome Screen

The Welcome screen shows the available activity centers in vManager. From the Welcome screen, you can launch only the Regression center. The Analysis center can be launched from the Regression center.

To proceed with the analysis tasks, click the Regression Center link on the Welcome screen.

vManager Regression Center -- RCL Client Mode



In this mode, only the session management and metrics analysis functionality is available.

When you place the mouse over the buttons with unavailable functionality (for example, vPlan or Tests in the Analyze toolbar), a tooltip is shown to indicate that -- to enable this functionality, upgrade the license.

When in RCL client mode, you can upgrade the license and enable the Full client mode (which includes all the functionality) by clicking the Upgrade to full client button.

Note: When you click the Upgrade to full client button, the tool checks out for the vmanager_client license.

■ If the license is available, then the Affirma_sim_analysis_env license is released and all the functionality of Full client is available.
■ If the license is not available, an error is reported and the tool continues to run in RCL client mode.
■ If the license is not available but the user launched the application with -licqueue option, then the application waits till the license becomes available.

Launching the vManager Full Client

The Full client mode supports complete vManager functionality. It requires the vmanager_client license.

The following command line launches vManager Full client in GUI mode:

vmanager

The above command launches vManager Full client, and displays the vManager Welcome screen, as shown in the Figure below.

vManager Welcome Screen

The Welcome screen shows the available activity centers in vManager. You can launch the Regression center and the Tracking center from the Welcome screen. The Analysis center can be launched from the Regression center.
Note: From the Welcome screen, you can also turn off the showing of the welcome screen on the next invocation of vManager. In case you turn off the welcome screen, then next time you launch vManager, by default, the Regression center will be shown.

Analyzing Verification Plans

Using vManager, you can analyze verification plans in detail.

Project Tracking

The Tracking center allows you to track data over time. Using the Tracking center you can track progress of sessions, tests, metrics, and verification plans. You can also generate charts and tracking reports.

Invocation and Command-Line Interface

The vmanager command is used to launch vManager.

-load_vplan <vplan_file>

Launches vManager and loads the specified vPlan file (<vplan_file>).

For example, to load vPlan file named newplan27_3.vplanx at the time of launching vManager, use:

vmanager -load_vplan /home/bob/ivm/ep/newplan27_3.vplanx

-load_refinement <refinement_file>

Launches vManager and loads the specified refinement file (<refinement_file>).

You cannot specify multiple refinement files on the command line.

Note: Session must be loaded before loading a refinement file.

To launch vManager in batch mode, and to load a refinement file named my_exclude.vRefine at the time of launching vManager, use:

vmanager -batch -load_session cdn_uart.segal.11_01_04_23_14_13_1797
-load_refinement /home/bob/ivm/ep/analyze/my_exclude.vRefine
-load_vplan_refinement <refinement_file>

Launches vManager and loads the specified vPlan refinement file (<refinement_file>).

You cannot specify multiple vPlan refinement files on the command line.

Note: Session and verification plan must be loaded before loading a vPlan refinement file.

To launch vManager in batch mode, and to load a vPlan refinement file named my_vp_exclude.vpRefine at the time of launching vManager, use:

vmanager -batch -load_session cdn_uart.segal.11_01_04_23_14_13_1797 -load_vplan
/home/bob/ivm/ep/newplan27_3.vplanx -load_vplan_refinement

/home/bob/ivm/ep/my_vp_exclude.vpRefine

planner

Launches vPlanner GUI.

-analyze_vplan

Immediately opens the vPlan view after launching vManager.

For example, the following command-line launches vManager in GUI mode, loads the specified vplan, and opens the vPlan view.

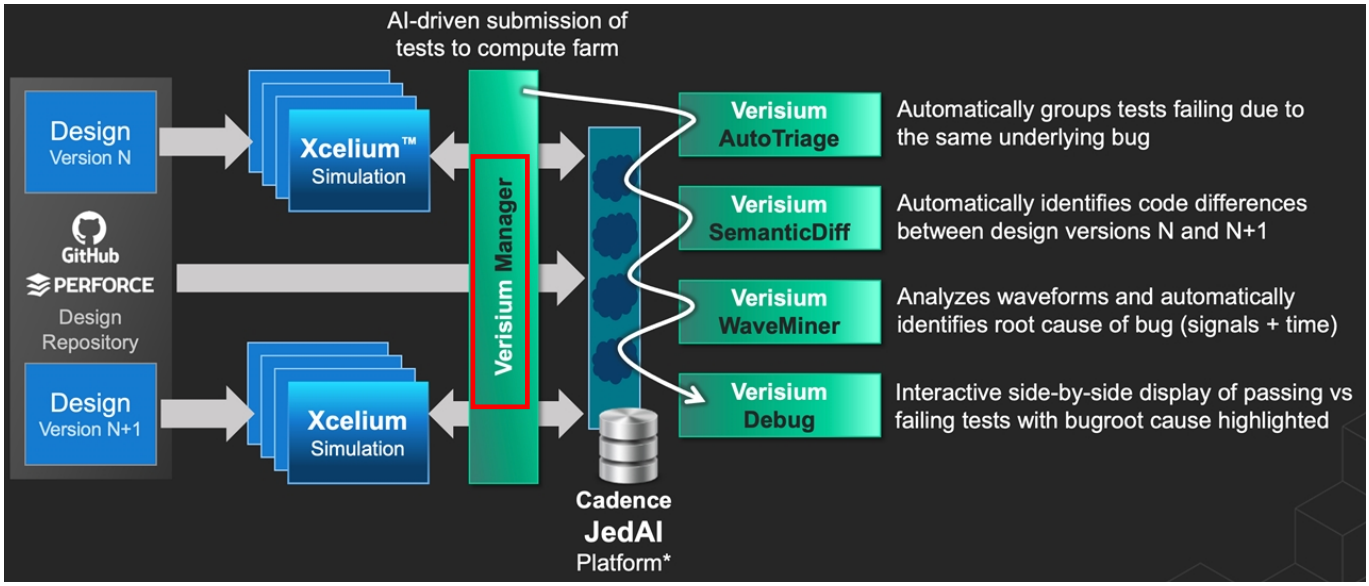
vmanager -load_vplan /servers/scratch03/bob/doc_examples/my_plan.vplanx
-analyze_vplan

report_vplan

The report_vplan command is used to generate a vPlan report. The report is created based on current context. The relevant sessions, vPlan, and refinement files must be loaded before executing this command.
Syntax
report_vplan
[-out <output_directory>]
[-overwrite]
[-title <report_title>]
[-view <view_name> [-filter <filter>]]
[-email <email_address>]
[-report_type regular | tabulated | extended]
[-extended true | false]
[-perspective {<perspective_name> | Unmapped} |-vplan_node <node> | -hierarchy
<hierarchy>]
{[-detail
[-metrics <metrics_type>]

5. CADENCE VERISIUM MANAGER / VERISIUM AI-DRIVEN VERIFICATION PLATFORM

The Cadence Verisium Artificial Intelligence (AI)-Driven Platform represents a generational shift from single-run, single-engine algorithms to algorithms that leverage big data and AI across multiple runs of multiple engines throughout an entire SoC verification campaign. The Verisium platform optimizes verification workloads, boosts coverage, and accelerates root cause analysis of bugs. It is built on the Cadence Joint Enterprise Data and AI (JedAI) Platform, enabling Cadence to unify its computational software innovations in data and AI across the full portfolio of Cadence products and solutions.



Cadence Verisium Artificial Intelligence (AI)-Driven Platform

The Verisium Platform is a suite of applications leveraging big data and AI to optimize verification workloads, boost coverage, and accelerate root cause analysis of design bugs on complex SoCs.

Verisium Manager: AI-Driven Verification Management

Industry-Leading Test Suite Management, Verification Planning, and Coverage Closure

Verisium Manager brings Cadence's full flow IP and SoC-level verification management solution with verification planning, job scheduling, and multi-engine coverage natively onto the Cadence JedAI Platform and extends it to support AI-driven testsuite optimization to improve compute farm efficiency. Verisium Manager also integrates directly with other Verisium apps, enabling interactive push-button deployment of the complete Verisium platform from a unified browser-based management console.

Cadence Verisium Manager automates end-to-end management of complex verification projects from planning to closure. Verisium Manager tightly integrates with the Cadence Verisium Artificial Intelligence (AI)-Driven Verification Platform, leveraging big data and AI to reduce silicon bugs and accelerate time to market. It is built on the new Cadence Joint Enterprise Data and AI (JedAI) Platform, providing the best multi-engine, multi-run, multi-user, and multi-site verification management capabilities.

Cadence Verisium Manager builds verification plans and close coverage across multiple engines to provide the clearest, most up-to-date view of verification progress.

Benefits

Improves Farm Efficiency

AI-driven test suite and compute resource optimization improve farm efficiency

Enhances Productivity

Intuitive planning, analysis, and coverage closure flows along with connection to other Verisium apps dramatically enhances verification productivity

Simplifies Total Verification Management

Built with an enterprise-grade, distributed architecture, and integrated with the Cadence JedAI Platform, Verisium Manager delivers a scalable, reliable, multi-user, multi-engine solution for verification teams across multiple projects and multiple sites.

Some of its features:

Verisium Manager provides a wide array of verification management capabilities and aggregates verification closure metrics across Cadence verification engines and applications into a single, intuitive interface.

Verification Planning

Connections to both specifications, requirements management tools (such as Jama, JIRA, and Doors), and native integration with coverage metrics generated by Cadence's Xcelium, Jasper, Palladium, and Protium platforms allow Verisium Manager to provide the best, most comprehensive multi-engine verification planning solution.

Powerful APIs

A rich portfolio of Python APIs enables direct access to the Verisium Manager data, metrics, and flows to enable customized data mining and test suite automation flows.

Job Scheduling and Execution

Verisium Manager connects with resource management and continuous integration tools such as LSF, Altair Accelerator, and Jenkins, as well as in-rack hardware job management engines on the Palladium and Protium platforms to deliver the highest levels of compute farm utilization.

Coverage Closure Analysis

Verisium Manager provides a comprehensive set of features and utilities for driving coverage closure, from fine-grained, bin-level coverage refinement through management-level dashboards and reporting.

Data at Your Fingertips

A unified browser-based management console provides regression data reporting and aggregation from management-level dashboards through fine-grained deep technical analysis.

6. SYNOPSYS HIERARCHICAL VERIFICATION PLAN (HVP)

Smart Tracking of SoC Verification Progress Using Synopsys' Hierarchical Verification Plan (HVP)

Introduction

SoC (System-on-Chip) Verification effort mainly includes three key phases: Planning, Development and Verification. Planning phase includes preparing verification strategy in terms of Test plan, Coverage plan and Assertion plan. Verification of complex SoC requires all micro level data (i.e. Individual Test status in Regression, Functional and Code Coverage numbers, etc.) to be collected at a common place for better tracking. Manual efforts to collect the above mentioned information may lead to human errors in reports, making the tracking data inaccurate and additional engineering efforts. Automating the whole verification tracking process is the ideal solution, which guarantees the accuracy and avoids tedious management from engineers.

Synopsys' VCS addresses aforesaid problem using Hierarchical Verification Plan (HVP), which provides flexibility to the user through user defined attributes while preparing the verification plan as per the project requirement. Once the verification environment is ready and the process of regression testing begins, using Hierarchical Verification Plan, one can gather all required information as mentioned earlier, at a common place in a spreadsheet or document format. Also one can back-annotate the results in individual plans, as well as in sub-sheets of those plans (if maintained in excel format). This article is targeted for Synopsys' VCS users, contains the flow of the Hierarchical Verification Plan creation in excel format, along with the detailed steps for integration of the same in verification environment with suitable example.

Basic flow for HVP Generation

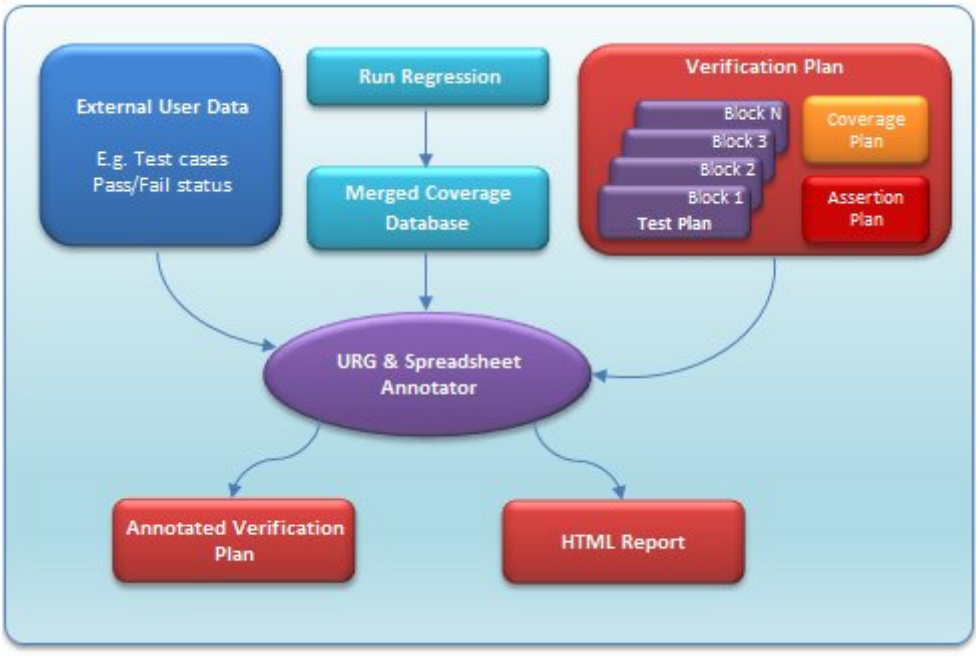


Figure 1. Basic Flow Diagram

In planning phase of the project, the initial step is to generate verification plan which contains test plan, coverage plan and assertion plan. Hierarchical Verification Plan (HVP) supports multiple formats like XML, Doc and others. In this article, the flow of HVP is explained using XML format. When the Regression phase begins, after every regression, the next step is to generate the merged coverage database using Unified Report Generator (URG). It is permissible to provide any user defined information which cannot be available in coverage database (like Regression results, Assertion summary and others) as external user data from command line. Using URG and spreadsheet annotator, user can get all the verification results back-annotated in respective plans. Detailed steps to generate HVP followed by the example are covered in the upcoming section.

Detailed Steps for HVP generation

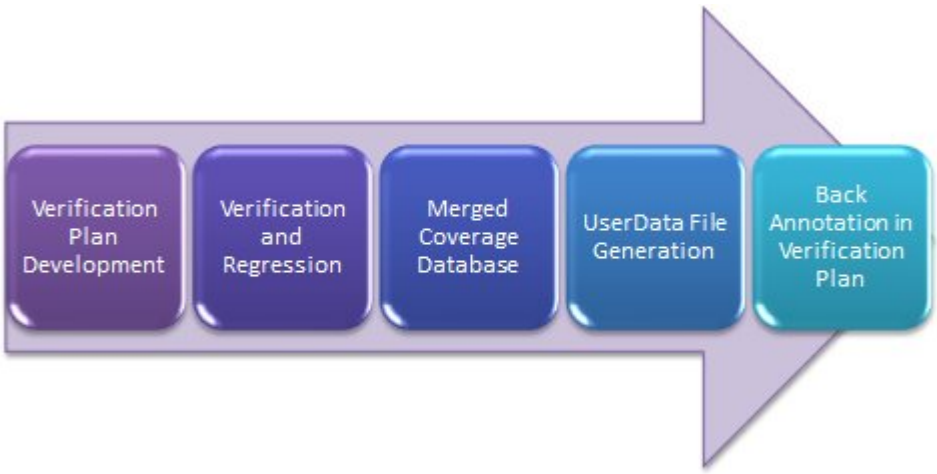


Figure 2. Steps for HVP Generation

A detailed explanation for each step mentioned in Figure 2 for HVP generation is explained below,

Step 1: Verification Plan Development

Generate plan file which contains the mapping of functional as well as code coverage, test scenarios and assertions in terms of features and measures. User can easily generate this plan file from DVE.

Sample plan file in .hvp format for code coverage is as shown in Figure 3.

```
plan ProjectX;
  owner = "eInfochips";
  description = "Project Description";

  feature Code_Coverage;
    at_least = 1;
    description = "Metrics to measure complete code coverage";
    measure Line, Cond, Toggle, FSM, Branch, SnpsAvg Code_Cov;
      source = "tree: proj_tb_top.proj_rtl_inst";
    endmeasure
  endfeature
endplan
```

Figure 3. Sample Plan File for HVP

Next step is to convert the above plan file into XML format using the below command:

hvp genxls -lca -plan ei_SoC_planfile.hvp

This command will generate the output file: **ei_SoC_planfile.hvp.xml**

It is not necessary to generate plan file in .hvp format in DVE every time. User can maintain one generalized format of XML or Doc verification plan, helping the user to update project specific information easily. If the test plan contains separate sheets for each block, those sheets needs to be added as Sub plan in the top level verification plan. Also the user needs to include all respective plans (i.e. Testplan.xml, Coverageplan.xml, Assertionplan.xml) in the top level verification plan as shown below in Figure 4.

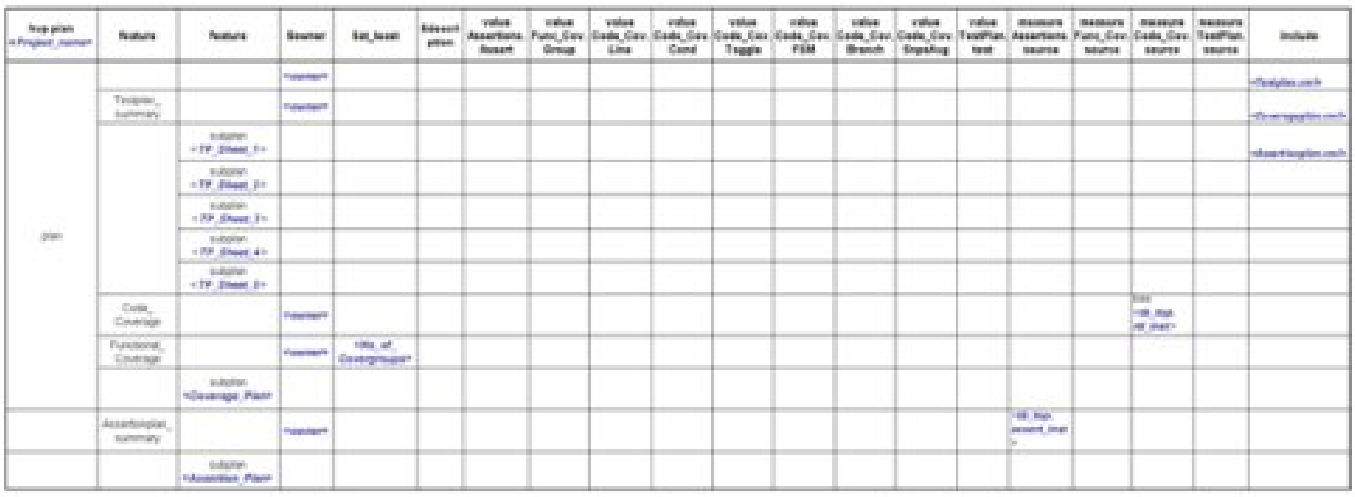


Figure 4. Generalize Format of Top Level Verification Plan

A sample top level XML plan file for automotive ei_SoC verification is shown in Figure 5. This Automotive SoC contains the blocks like SPI, I2C, Ethernet, GPIO, HSIC-USB2.0, etc. ei_SoC_Testplan.xml, ei_SoC_Coverageplan.xml and ei_SoC_Assertionplan.xml are included in the last column as shown in Figure 5. Testplan ei_SoC_Testplan.xml contains the separate sheet for each block of the SoC. Sample testplan for SPI block and System level scenarios are shown in Figure 6.

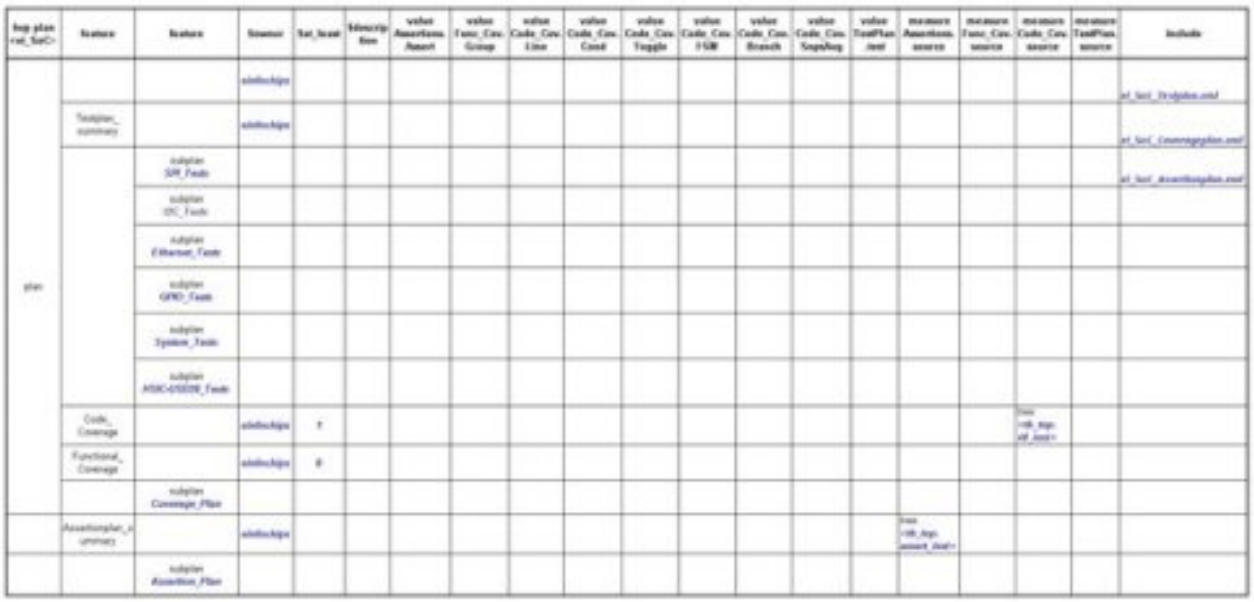


Figure 5. Sample Top Level Verification Plan

After creating top level verification plan, user needs to add some relevant options, which are required for the back-annotation of information, as in the respective plan(s) shown in Figure 6, 7 and 8. The same options are also required in the Top level verification plan for back-annotation of results.

Sr. No.	Test Case Name	Test Feature		Sequence	Remarks	Priority	Bug # In Case of Design Issue	Tlst	Test status
hvp plan SPI_Tests	feature		measure vnps.source	$description					value vnps.test
1	SPI_sanity_test	Sanity Testcase	SPI_sanity_test	Detailed steps for stimulus generation for the scenario				spi.tlst	
2	SPI_wr_after_rd_test	Send SPI read transactions after write	SPI_wr_after_rd_test	Detailed steps for stimulus generation for the scenario				spi.tlst	
3	SPI_on_the fly_rst_test	On the fly reset while sending SPI trascactions	SPI_on_the fly_rst_test	Detailed steps for stimulus generation for the scenario				spi.tlst	
4	SPI_rand_test	SPI transafctions with random address and random operation(wr or rd)	SPI_rand_test	Detailed steps for stimulus generation for the scenario				spi.tlst	

Sr. No.	Test Case Name	Test Feature		Sequence	Remarks	Priority	Bug # In Case of Design Issue	Tlst	Test status
hvp plan System_Tests	feature		measure vnps.source	$description					value vnps.test
1	System_Test_0	Objective of the scenario	System_Test_0	Detailed steps for stimulus generation for the scenario				system.tlst	
2	System_Test_1	Objective of the scenario	System_Test_1	Detailed steps for stimulus generation for the scenario				system.tlst	
3	System_Test_2	Objective of the scenario	System_Test_2	Detailed steps for stimulus generation for the scenario				system.tlst	
4	System_Test_3	Objective of the scenario	System_Test_3	Detailed steps for stimulus generation for the scenario				system.tlst	
5	System_Test_4	Objective of the scenario	System_Test_4	Detailed steps for stimulus generation for the scenario				system.tlst	
6	System_Test_5	Objective of the scenario	System_Test_5	Detailed steps for stimulus generation for the scenario				system.tlst	

Figure 6. Test Plan(SPI, System)

SR. NO.	BLOCK	Cover Group		Cover Point	Cover Bins	Description	Status
	hvp plan Coverage_Plan	feature	measure Func_cov.source				value Func_cov.Group
1	SPI	spi_cg	group: ei_SoC_env_pkg::ei_SoC_coverage_class_c::spi_cg	spi_addr_cp spi_trans_type_cp spi_enable_cp			
2	I2C	i2c_cg	group: ei_SoC_env_pkg::ei_SoC_coverage_class_c::i2c_cg	i2c_addr_cp i2c_trans_type_cp i2c_enable_cp			
3	GPIO	gpio_cg	group: ei_SoC_env_pkg::ei_SoC_coverage_class_c::gpio_cg	gpio_cp			

Figure 7. Coverage Plan

Sr No.	Assertion ID	Discription	Result		
			Attempts	Success	Failures
hvp plan Assertion_Plan	feature	$description	value at.attempts	value sc.success	value fl.failures
1	ei_SoC_tb_top.assert_inst.ASSERT_ID_1	Description for Assertion_1			
2	ei_SoC_tb_top.assert_inst.ASSERT_ID_2	Description for Assertion_2			

Figure 8. Assertion Plan

Step 2: Verification and Regression

After completing the verification plan development, next step would be to develop the test cases for functional verification of SoC and parallelly start running the regression of all test cases.

Step 3: Merge Coverage Database

After running the regression, merge the coverage database of all the test cases which is required for the back-annotation of coverage numbers in coverage plan using below command:

urg -dir ei_SoC_merged_cov.vdb <list of vdb files for individual test>

Step 4: UserData File Generation

If user wants to annotate some user defined information in any of the plans, it can be given as input using command line option -userdata. Here we have given ei_SoC_regression_status.txt to annotate the pass/fail status of individual test cases and ei_SoC_assertion_status.txt to annotate the assertion results in assertion plan. This user defined information is generated by using the regression result post-processing script. Sample user data file for regression status is as shown in Figure 9.

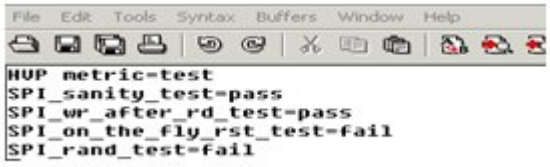


Figure 9. Sample Userdata File for Regression Status

Step 5: Back-Annotation in Verification Plan

To get back-annotated HVP report in .xml.ann format, user needs to apply the following command with inputs shown below:

hvp annotate -lca -plan ei_SoC_planfile.hvp.xml -dir
ei_SoC_merged_cov.vdb -userdata ei_SoC_regression_status.txt
ei_SoC_assertion_status.txt -plan_out ei_SoC_planfile.ann.xml

User can also get the annotated XML with user defined name using –plan_out option.

For HTML report of HVP, user needs to apply the following command with inputs shown below:

urg -plan ei_SoC_planfile.hvp.xml.hvp -dir ei_SoC_merged_cov.vdb
-userdata ei_SoC_regression_status.txt ei_SoC_assertion_status.txt -report
<path to o/p directory>

Result Analysis

After following the above mentioned steps, results get annotated in XML sheet. Annotated XML contains a separate sheet for top level results followed by individual sheet for each sub plan (i.e. **SPI_Tests, I2C_Tests, Ethernet_Tests, GPIO_Tests, System_Tests, HSIC-USB20_Tests, Coverage_Plan, Assertion_Plan**) as shown in Figure 9. User can jump to the individual sub sheet by clicking on the name of that sub plan in the features column of top level sheet as shown in Figure 10. Annotated test plan for SPI Block and System level scenarios with pass/fail status of individual test are shown Figure 11. Similarly, Figure 11 and 12 show the annotated Coverage numbers and Assertion status with different colors as per the different range of numbers.

Figure 10. Top Level Annotated Report

Sr. No.	Test Case Name	Test Feature		Sequence	Remarks	Priority	Bug # In Case of Design Issue	Tlst	Test status
hvp plan SPI_Tests	feature		measure srqs source	$description					value srqs test
1	SPI_sanity_test	Sanity Testcase	SPI_sanity_test	Detailed steps for stimulus generation for the scenario				spi.tlst	totals=1 passs=1
2	SPI_wr_after_rd_test	Send SPI read transactions after write	SPI_wr_after_rd_test	Detailed steps for stimulus generation for the scenario				spi.tlst	totals=1 passs=1
3	SPI_on_the fly_rst_test	On the fly reset while sending SPI trascactions	SPI_on_the fly_rst_test	Detailed steps for stimulus generation for the scenario				spi.tlst	totals=1 passs=1
4	SPI_rand_test	SPI transafctions with random address and random operation(wr or rd)	SPI_rand_test	Detailed steps for stimulus generation for the scenario				spi.tlst	totals=1 fails=1

Sr. No.	Test Case Name	Test Feature		Sequence	Remarks	Priority	Bug # In Case of Design Issue	Tlst	Test status
hvp plan System_Tests	feature		measure srqs source	$description					value srqs test
1	System_Test_0	Objective of the scenario	System_Test_0	Detailed steps for stimulus generation for the scenario				system.tlst	totals=1 passs=1
2	System_Test_1	Objective of the scenario	System_Test_1	Detailed steps for stimulus generation for the scenario				system.tlst	totals=1 passs=1
3	System_Test_2	Objective of the scenario	System_Test_2	Detailed steps for stimulus generation for the scenario				system.tlst	totals=1 fails=1
4	System_Test_3	Objective of the scenario	System_Test_3	Detailed steps for stimulus generation for the scenario				system.tlst	totals=1 passs=1
5	System_Test_4	Objective of the scenario	System_Test_4	Detailed steps for stimulus generation for the scenario				system.tlst	totals=1 fails=1
6	System_Test_5	Objective of the scenario	System_Test_5	Detailed steps for stimulus generation for the scenario				system.tlst	totals=1 passs=1

Figure 11. Annotated Test Plan (SPI, System)

SR. NO.	BLOCK	Cover Group		Cover Point	Cover Bins	Description	Status
	hvp plan Coverage_Plan	feature	measure Func_cov.source				value Func_cov.Group
1	SPI	spi_cg	group: ei_SoC_env_pkg::ei_SoC_coverage_class_c::spi_cg	spi_addr_cp spi_trans_type_cp spi_enable_cp			76.82%
2	I2C	i2c_cg	group: ei_SoC_env_pkg::ei_SoC_coverage_class_c::i2c_cg	i2c_addr_cp i2c_trans_type_cp i2c_enable_cp			100.00%
3	GPIO	gpio_cg	group: ei_SoC_env_pkg::ei_SoC_coverage_class_c::gpio_cg	gpio_cp			42.60%

Figure 12. Annotated Coverage Plan

Sr No.	Assertion ID	Discription	Result		
			Attempts	Success	Failures
hvp plan Assertion_Plan	feature	$description	value at attempts	value sc success	value fl failures
1	ei_SoC_tb_top.assert_inst.ASSERT_ID_1	Description for Assertion_1	7945106	198	99
2	ei_SoC_tb_top.assert_inst.ASSERT_ID_2	Description for Assertion_2	7945106	297	0

Figure 13. Annotated Assertion Plan

Conclusion

The development of Hierarchical Verification Plan (HVP) using Synopsys' Unified Report Generator (URG) can facilitate an easier and more efficient way to track the verification progress. Hierarchical Verification Plan (HVP) provides deeper visibility into the regression process and coverage analysis. Key features of HVP like HTML report generation, multiple supported formats (i.e. XML, Doc etc.), back-annotation of reports in plan itself can help us in reducing the time and manual efforts required while preparing the closure documents of the project.

7. SYNOPSYS VERDI COVERAGE

Highlights

Verdi, the industry's open debug platform, now provides innovative planning and coverage technology integrated across all debug views, which allows users to quickly analyze and cross-probe holes identified through coverage analysis.

- Provides the ability to quickly generate verification plans and link them to specification and requirements documents to ensure that projects stay on track as high-level requirements change
- Integrated with regression management solution, enabling users to execute verification tasks, view results and triage data to meet coverage goals and achieve verification closure more quickly
- Supports integrated visualization of results across simulation static checking, formal verification, verification IP (VIP) and FPGA-based prototyping solutions to deliver a unified view of verification closure

Synopsys Verdi® Coverage Advanced Planning and Coverage Technology

This solution addresses the growing challenge of verification closure for complex system-on-chips (SoCs) by introducing advanced technology that allows users to quickly create efficient verification plans, integrate third-party and user-defined metrics, link plans to requirement documents, and intuitively track project and test-level metrics across simulation, static checking, formal verification, VIP and FPGA-based prototyping. Verdi Coverage

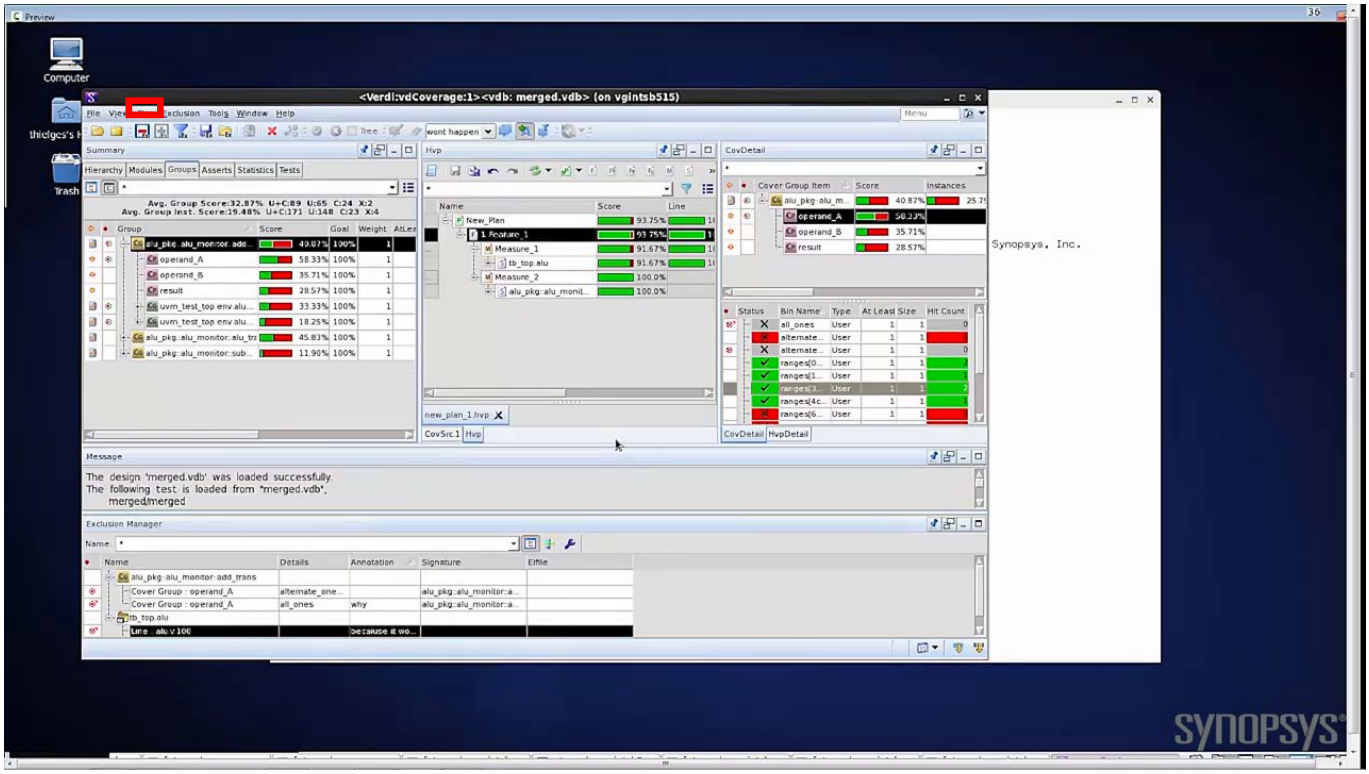
enables users to understand project progress, manage regression data, launch verification jobs, track project trends, generate reports and ultimately optimize resource allocation.

Verdi Coverage successfully shortens time-to-revenue and minimizes schedule risks which require extensive upfront verification planning, flexible regression management and sophisticated coverage analysis. With its integrated coverage and debug capabilities, Verdi Coverage enables design teams to collaborate, identify the root cause of coverage holes and quickly drive clients' projects to verification closure.

Synopsys' innovative planning and coverage analysis technology is built on top of the Verdi environment recognized for being optimized, extensible and easy to use. Fully integrated with all Verdi debug views, it allows users to quickly analyze and cross-probe any holes identified through coverage analysis. It is also interoperable across a wide range of solutions including simulation, static checking, formal verification, VIP and FPGA-based prototyping. This unique level of integration and interoperability provides a unified view of project status that allows users to focus only on tasks required to improve the predictability of verification closure.

Synopsys have made significant investment in verification planning and coverage visualization to address the design challenges. Verification planning, coverage analysis and debug are fundamental elements of the Verification Compiler™ product, providing customers with superior technology to help them meet their ever-shrinking schedules..

A Quick Tour of Verdi Coverage | Synopsys



Synopsys's popular Verdi tool is conceived as a flexible debug tool, it also has an open scripting environment that gives engineers access to data in the fast signal database (FSDB) file. With that capability, folks have been bolting analysis utilities onto Verdi for a while on an ad hoc basis.

This hasn't gone unnoticed at Synopsys, and they're now in the process of repositioning Verdi: it's not just for debug anymore. While it obviously still includes debug in its expanded portfolio, Synopsys is adding features that

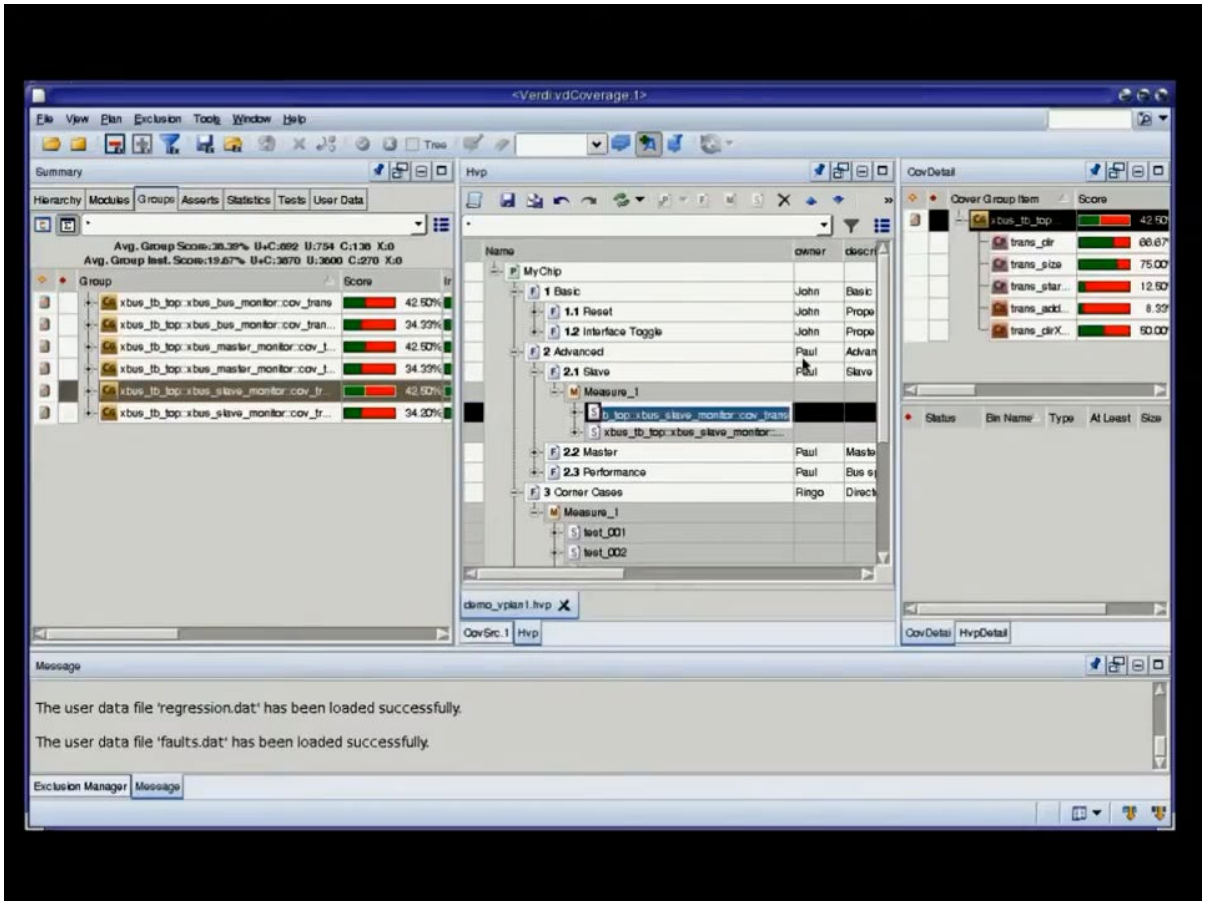
don't necessarily fit the debug profile.

One of those is Verdi Coverage. This is intended to help build and track a verification plan that is tightly synchronized with the design requirements. This concept might be familiar to any of you that have seen similar tools in the software space from companies like LDRA.

The assumption here is that verification tests spring from requirements. (If it's not required, then why are you testing it?) And all requirements should be documented in a requirements document. Verdi Coverage lets you tie tests to requirements and tick off coverage at the requirements level.

A Quick Tour of Verdi Coverage

Cool Things You Can Do with Verdi – Verification Planning (Introduction)



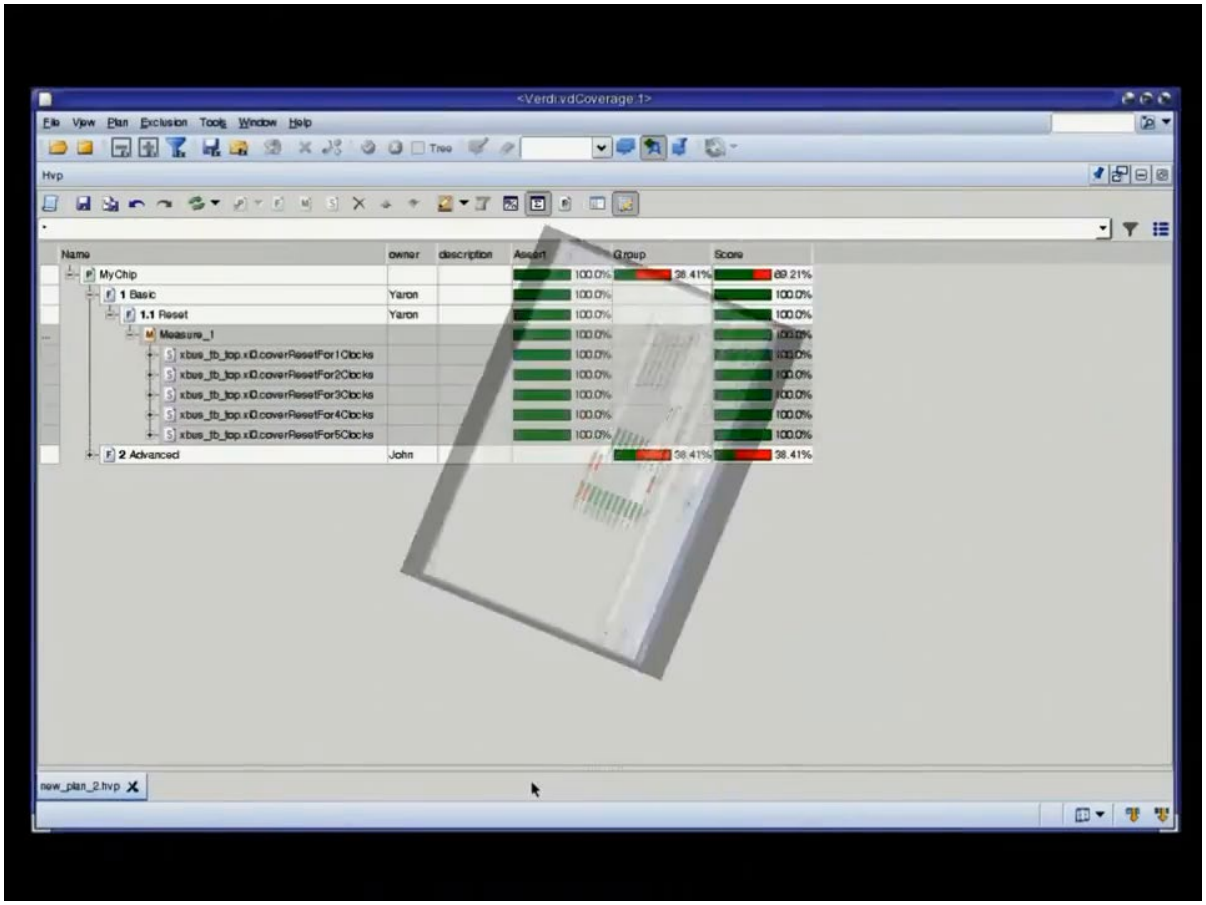
Where this can be particularly helpful is when requirements change. Yeah, it's a thing; it happens. Who knows?

Verdi Coverage tracks the requirements documents and can notice when changes occur. This allows you to go in and modify the verification plan accordingly, if needed.

How do they do that? They rely on the PDF file format for the document. Best practice is to use an outline structure in that document. They capture the text from the document, along with some meta-information about where the text is to be found.

Cool Things You Can Do with Verdi – Verification Planning (Introduction)

Cool Things You Can Do with Verdi – Verification Planning (Advanced)



And when the document changes? How can they pinpoint the changes? Diff technology. Off the shelf, actually. Apparently the ability to diff two files has gotten pretty good these days. (It's not as easy as you might think: as soon as one thing changes, then everything after it might seem different unless you can identify the type and scope of the change and then get back on track with unchanged text.) The important thing is this: there's no special

formatting you need to do so that this will work. Write a well-organized, well-structured document (so that a human can process it well) and Verdi Coverage will be able to handle it.

Far from being a debug thing, this becomes a planning tool up front, creating a specific link between requirements and the elements of the verification plan. It applies across verification technologies (formal, simulation, etc.). As long as the requirements document is being kept up to date, there's no reason for the verification plan to get out of synch with it.

Cool Things You Can Do with Verdi – Verification Planning (Advanced)

8. SYNOPSYS VC EXECUTION MANAGER (EXECMAN)

VC Execution Manager
https://www.synopsys.com/verification/soc-verification-automation/vc-execution-manager.html

Coverage-Driven Execution Management for SoC Verification Flows.

Overview

VC Execution Manager ("ExecMan") drives and automates coverage-driven SoC functional verification flows, and is a key element of the Synopsys SoC Verification Automation solution. ExecMan is a versatile, modular system for managing compilation, regression, and tracking of the design verification process. It is preconfigured to work with VCS flows, and is natively integrated with Verdi's Planning and Coverage features.

Remember the days when verification meant running a simulator with directed tests? (Back then we just called them tests.) Then came static and formal verification, simulation running in farms, emulation and FPGA prototyping. We now have UVM, constrained random testing and many different test objectives (functional, power, DFT, safety, security, cache coherence). Over giant designs are now needing hierarchies of test suites. And giant regressions to ensure backward compatibility, compliance and coverage while aiming to optimize use of compute farms and clouds. It's all become a bit more complicated than it used to be. To achieve the productivity and efficiency gain needed to keep up, automating verification management of this complex and diverse set of objectives becomes essential.

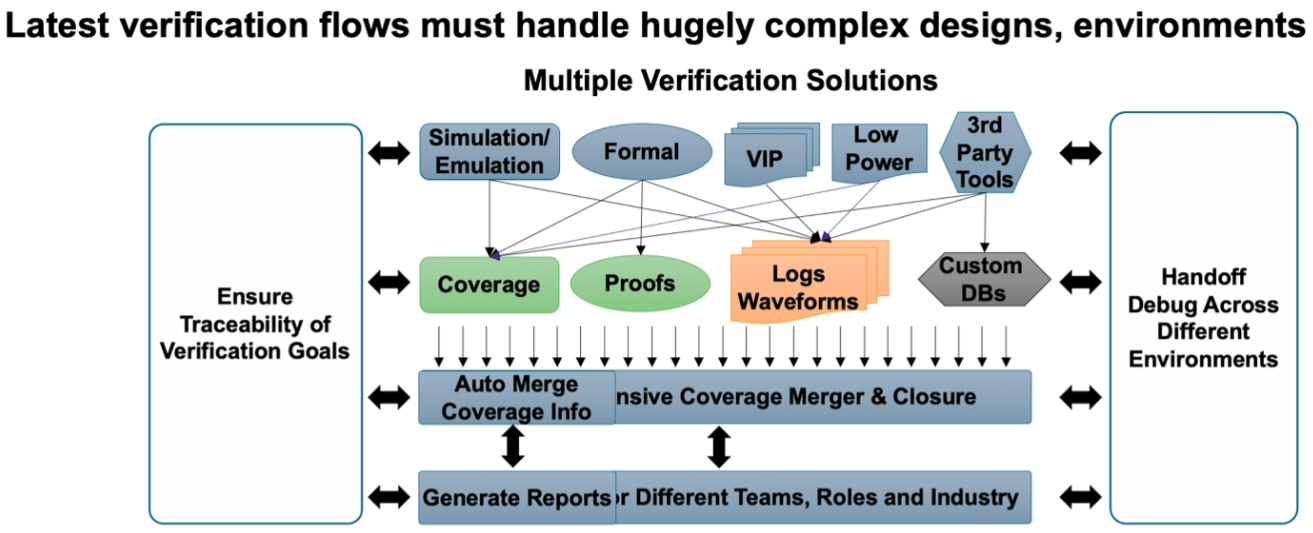


Figure 1 Verification Management

Introduction

VC Execution Manager dispatches regression jobs, tracks and collects regression results, annotates these results back to the verification plan, and provides full reporting capabilities. Key values:

- Supports all functional verification engines
- Optimizes regressions for TAT and costs
- Collects and tracks regression data with a full relational database
- Automatically re-runs select simulation failures
- Provides fault tolerance for large regression runs, including for disk access interruptions and failed hosts
- Flexibly supports local execution, as well as widely-used grid managers

VC Execution Manager ("ExecMan") is a powerful and flexible system for managing compilation, regression test execution, data collection, reporting and tracking of the design verification process. ExecMan automates coverage-driven SoC functional verification flows, tracks and collects regression results data in a relational database, and supports annotation of coverage results in Verdi Planner.

VC Execution Manager is preconfigured to work with VCS flows, and is natively integrated with Verdi's Planning and Coverage features. ExecMan can also scale to support any number and any size of projects and users, is fully extensible and customizable, and provides full reporting capabilities, including API access for custom reports.

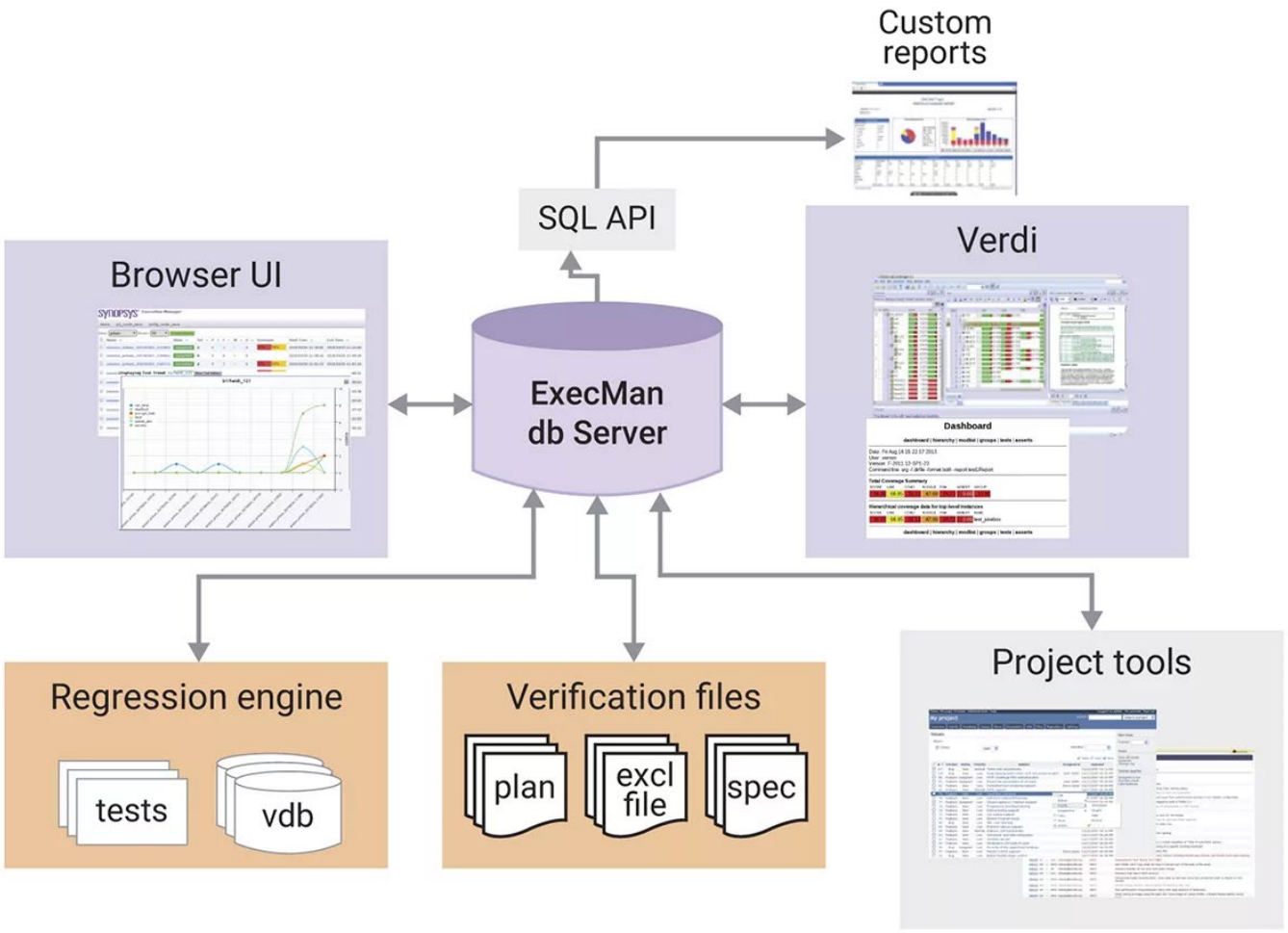


Figure 2 Integration Block

VC Execution Manager Key Features

Interactive launch and control
Automatic coverage merging
Verification plan annotation
Coverage driven
Automatic debug re-run
Failure binning
Full regression database server support, and serverless capability for smaller designs
Test correlation
Graded test lists

VC Execution Manager consumes data produced by individual tools and technologies during the functional verification flow (e.g. VCS, VC Formal, etc.), and tracks the resulting changes over time in its relational database. Results can be annotated onto the verification plan via native integration with Verdi Planner. ExecMan also provides a fault-tolerant execution layer—the Test Control Engine—to manage job dispatch and data collection tasks.

Comprehensive verification management

The ExecMan solution has five primary goals:

Provide a systematic path linking from testplan to execution, debug and coverage and trend analysis
Optimize regression turn-around times
Minimize debug turn-around times
Optimize time to closure
Utilize the grid as effectively as possible

The planning phase links a design plan to a test plan and subsequently through to analysis and debug. This is very useful in establishing traceability between specs and testing.

Verdi Planning and Coverage Integration

ExecMan is natively integrated with Verdi to provide regression results annotation, coverage viewing, and job progress monitoring. This enables verification engineers and teams to easily manage verification execution in the same familiar environment used for planning, coverage and debug tasks.

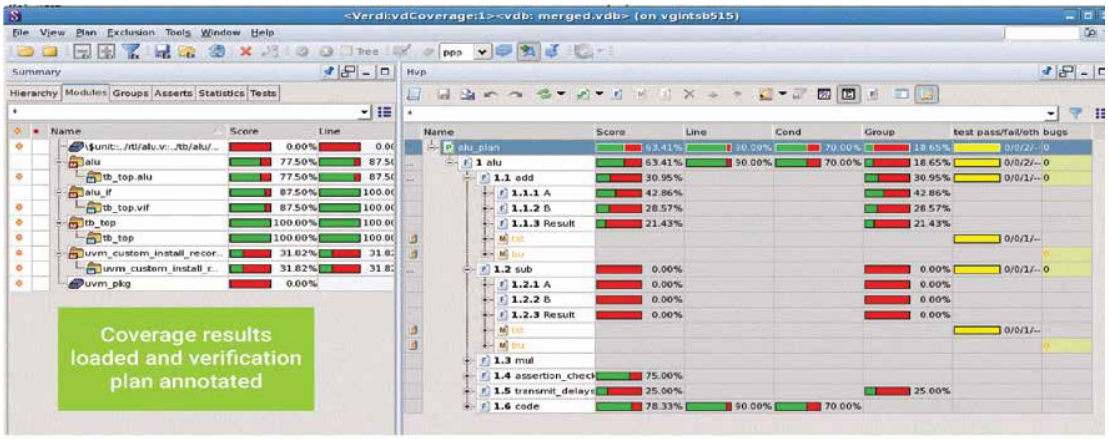


Figure 3. Verdi planning and coverage integration

VC Execution Manager Database Server Features

ExecMan tracks and stores regression run results in a relational database, which is architected to easily plug into existing regression systems. The ExecMan database is managed and tracked for the length of the project, and supports full administrative and reporting capabilities.

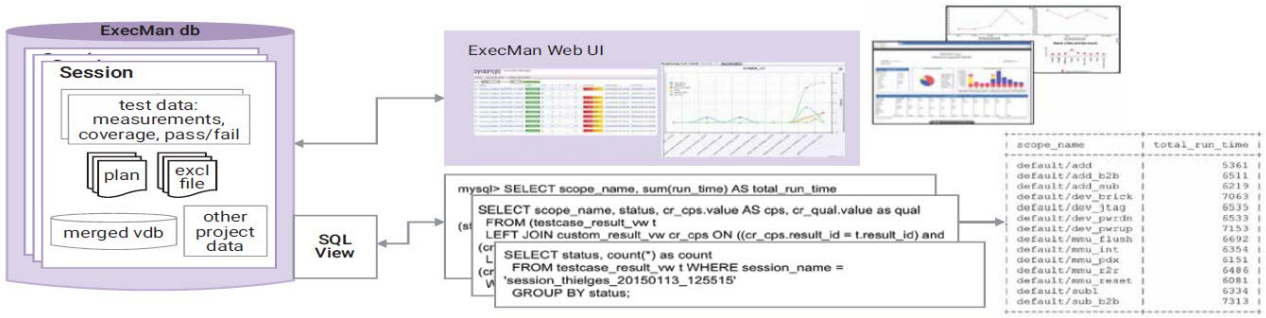


Figure 4. ExecMan database

VC Execution Manager is available as Server and Client products. VC Execution Manager Server enables all functionality including ExecMan database setup and management. VC Execution Manager Client does not provide ExecMan database management capabilities, and works in conjunction with VC Execution Manager Server-administered databases, or can also work standalone in 'serverless' mode for smaller verification environments.

Optimizing regression turnaround-time and debug productivity

One important consideration in optimizing regression throughput is simply load-balancing. Packing jobs in such a way that total turn-around time per regression pass is minimized to the greatest extent possible. The manager helps optimize this balancing. It also apparently does some level of reduction in redundant test identification, using coverage analytics. There's also a note in the slides on VCS engine performance enhancement in this release – I believe VCS 2020.12.

To optimize debug productivity, the manager provides help in several ways. First it automatically sets up debug runs to run in parallel with ongoing regression runs. You can supply debug hooks up-front to drive such runs. There's also mention in the slides of ML-based failure triage and debug assistant(s).

Optimizing closure turn times and grid utilization

Here there's more focus on test grading by coverage, to filter out tests which don't contribute significantly. Synopsys have also just introduced a feature called Intelligent Coverage Optimization (ICO), using ML to bias constraints for randomization, again to minimize low value sims. They claim 5X reduction in turn-around time using this technique for stable CR regressions.

Finally, on this general optimization theme, the manager optimizes for grid efficiency, looking at the best way to assign tasks to specific grid hosts. The manager does this by analyzing environment and historical data.

More goodies

ExecMan adds further automation for results binning, re-run and debug through Verdi. It further supports coverage analysis through test grading and plan grading tools and can link with bugs tracked in Redmine issue-tracking.

9. ONESPIN VERIFICATION PLANNING INTEGRATION (VPI) APP / ONESPIN PORTABLECOVERAGE SOLUTION

Integration of Formal and Simulation Results and Coverage

https://www.onespin.com/
https://www.onespin.com/portablecoverage

PortableCoverage verification plan

Now that formal verification is part of mainstream verification and used on most chip designs, project managers need to understand the role played by formal tools and what they have contributed to the overall verification process. They also want to reduce verification effort by minimizing the overlap between formal and simulation. This requires an integration of formal and simulation coverage metrics for a unified view of coverage status. Similarly, the results of simulation tests and formal analysis must be annotated back into the verification plan to provide a comprehensive view of verification status against the project schedule. These quantitative metrics for results and coverage enable the verification team to determine next steps and assess when the design is ready for tape-out (ASICs) or the bring-up lab (FPGAs).

OneSpin's PortableCoverage™ solution

OneSpin's PortableCoverage™ solution provides this integration in an open flow that works with any simulator, coverage database, coverage viewer or verification planning tool. Users are free to choose best-in-class tools from multiple vendors while getting full credit for the verification performed by the OneSpin suite of formal tools. The PortableCoverage solution uses four formal apps to provide different aspects of the integration:

Coverage Closure Accelerator (CCA) App

The **Coverage Closure Accelerator (CCA) App** analyzes the design, identifies unreachable coverage targets, and provides them to the simulator or coverage viewer. This provides more accurate coverage metrics, ensures that the verification team doesn't waste time in simulation trying to hit coverage targets that are unreachable, and accelerates coverage closure.

Quantify™ App

The **Quantify™ App** calculates how well the assertions cover the design, providing guidance on where additional assertions may be needed. This calculation is performed by model-based mutation coverage, a unique technology that is far more precise than other assertion quality measures. No changes to the design or re-compilations are required.

Verification Coverage Integration (VCI) App

The **Verification Coverage Integration (VCI) App** adds the model-based mutation coverage metrics from Quantify into the user's existing coverage database. The coverage viewer can then display the integrated formal and simulation metrics, providing a unified view of coverage status.

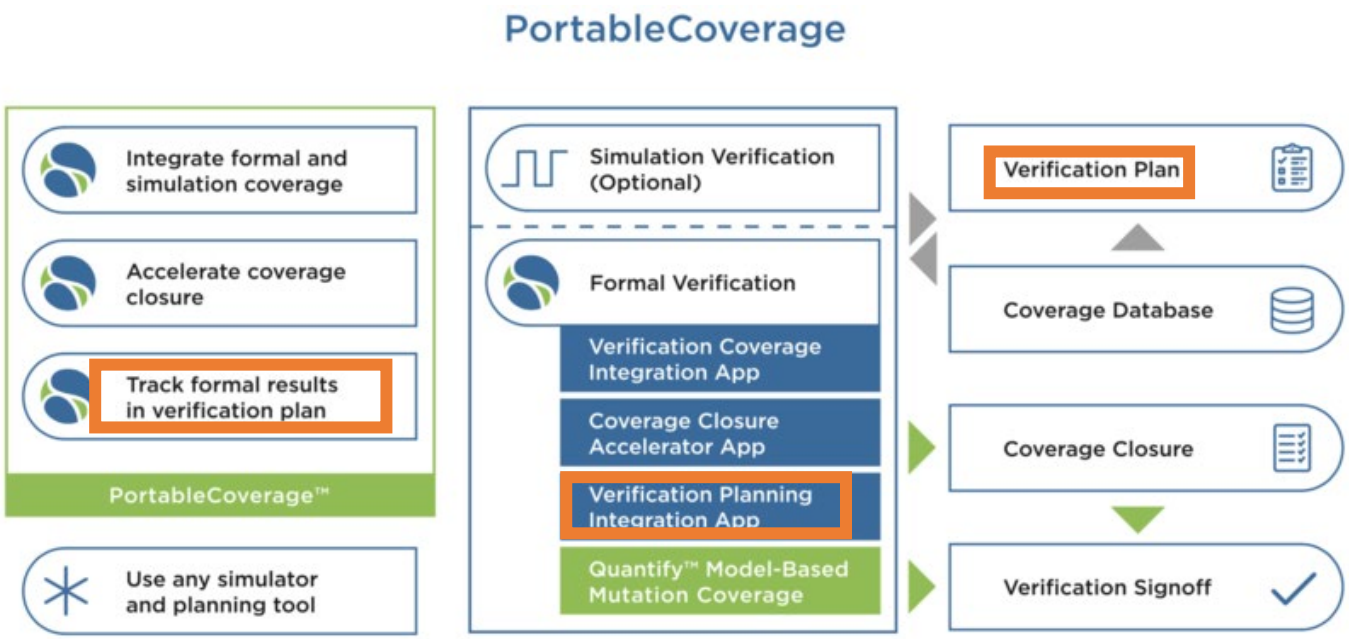
Verification Planning Integration (VPI) App

The **Verification Planning Integration (VPI) App** annotates formal results from OneSpin's tools into the verification plan. The integrated formal and simulation results provide a comprehensive view of progress against the verification plan. It also ensures that engineers can meet stringent safety standards by providing links that track verification results back to the design specification.

The Verification Planning Integration App import the results from formal analysis into any of the leading planning tools. Users must understand the mappings of results between simulation and formal, and examine bounded (incomplete) proofs carefully to see if more verification is needed.

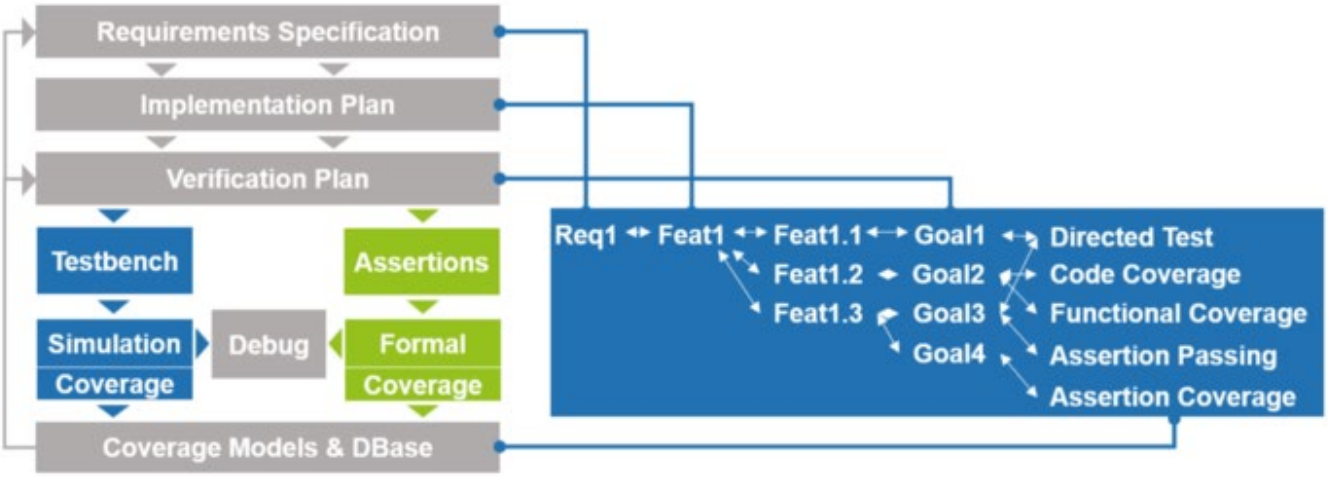
OneSpin 360 DV-Inspect™ and OneSpin 360 DV-Verify™

All of these apps leverage the formal verification technology provided by OneSpin 360 DV-Inspect™ and OneSpin 360 DV-Verify™.



Integrating assertion results of OneSpin 360 DV with planning tools
https://www.onespin.com/products/specialized-apps/verification-planning-integration/

In a systematic verification flow, requirements tracking and coverage plays an indispensable role. Generally, this starts from requirements specification, where individual requirements are broken down into features, implementations, verification goals and metrics. In order to achieve these goals, verification engineers are taking different approaches, like writing testbenches for simulation environment or properties for assertion-basedformalverification. For each approach taken, to keep track of what requirements have been met users are generating coverage databases from simulation and formal environments. Because these are coming from different vendors, the big challenge is how can these coverage databases can be merged into one and to annotate the results into existing verification plans.


Requirements Specification
Implementation Plan
Verification Plan
Testbench
Assertions
Simulation
Coverage
Debug
Formal
Coverage
Coverage Models & DBase
Req1
Feat1
Feat1.1
Feat1.2
Feat1.3
Goal1
Goal2
Goal3
Goal4
Directed Test
Code Coverage
Functional Coverage
Assertion Passing
Assertion Coverage