cadence®

Maximizing Verification Effectiveness Using MDV

By Nick Heaton, Distinguished Engineer, Cadence Design Systems, Inc.

This paper introduces the Cadence® Incisive® Verification Kit as a golden example of how to maximize verification effectiveness by applying metric-driven verification (MDV) in conjunction with the Universal Verification Methodology (UVM). MDV provides an overarching approach to the verification problem by transforming an open-ended, open-loop verification process into a manageable, repeatable, deterministic, and scalable closed-loop process. Through this transformation, verification project teams and managers greatly increase their chances of consistently delivering working designs at improved quality levels in less time with fewer human resources.

Contents

Introduction

The functional verification landscape has changed beyond all recognition over the last 10 years, and while design paradigms have become mature and stable, verification methodologies and technologies have continued to evolve with new flows and tools being invented. Against this rapidly changing background, designs have been steadily growing to gigascale dimensions, with more intellectual property (IP) and more complex IP being integrated into larger and larger system-on-chip (SoC) designs.

Realizing silicon in the face of these challenges requires new approaches and a very flexible workforce capable of adapting and changing on a regular basis. This paper outlines a deterministic approach to managing verification complexity—MDV—and a valuable resource to back up this approach, the Incisive Verification Kit. Together, MDV and the Incisive Verification Kit will enable you to plan and embrace new methodologies and approaches in a safe and controlled way, with a step-wise progression to the ultimate in verification productivity—plan-based MDV. The kit provides clear and realistic examples at each stage to guide you and your teams in their new projects.

Failing to Plan = Planning to Fail

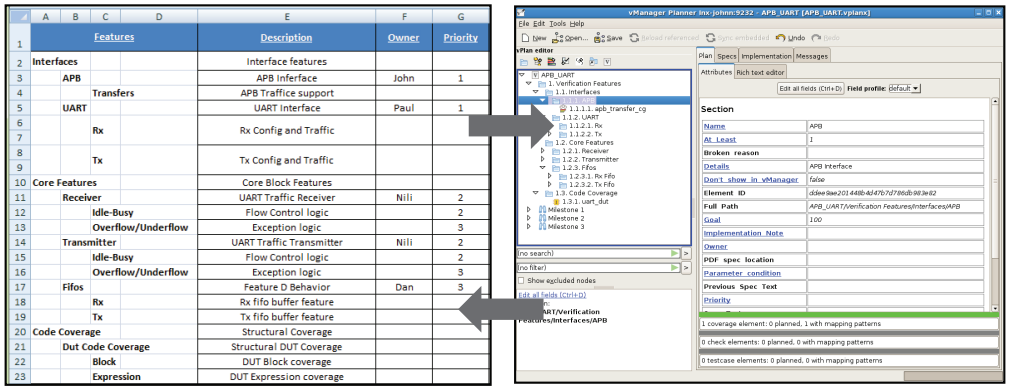
Any management process needs clear and measurable goals, and verification is no exception. Failing to capture these goals at the outset of a project means that there is no clear definition against which to measure either progress or closure. You can only gauge improvement by what you can clearly measure. An example of this problem is seen in directed testing. Huge lists were drawn up to define the verification process, but the lists were not executable or maintainable. This open-ended nature led to big project slips and huge stresses on project teams.

In addition, there is often confusion about the definition of what constitutes a verification plan and what constitutes a test plan. Let's make our definition clear right away: a test plan defines a fully elaborated list of tests that will be created, and the completion of this list defines completion of verification. A test plan alone however tends to be a poor mechanism since it already contains decisions about the execution of the process. In contrast, a verification plan captures the "what" that must be verified but does not define the execution—the "how." This "what" versus "how" is a crucially important distinction as it is expected that verification is performed by multiple people using multiple technologies. We refer to the "what" as the "features" or "goals" that must be verified—the design intention. When planning a project, you should not presume the underlying tool by which a feature is verified. It will most likely come from several sources, with the results represented by various forms of coverage. Luckily, this top-down approach is not the only way to begin, and starting bottom up, with automation of the test plan, provides instant value in and of itself.

A verification plan should capture the goals of the verification process in a way that the signoff criteria and milestones are clearly identified. An example of such a milestone is RTL code freeze, which a team might define as the point when 75% of all features are covered. In addition to capturing goals upfront, a verification plan should be executable. In other words, it is essential that progress toward project closure can be easily and automatically measured. Typically, the goals get refined as a project progresses, which means the verification plan is a living document that matures over the lifetime of a project.

Lastly, the human factors in verification planning must be accounted for up front. People are humans who make mistakes and learn from those mistakes. Changes happen, and a good verification planning methodology should be part of the overall verification process, although, sadly, in most cases it is not. Codifying the planning methodology within tooling can help with everything from capturing brainstorm results, to knowing what has changed in the specification, to setting goals and owners.

Figure 1 shows a sample of a verification plan for the UART block in the Incisive Verification Kit environment, which has the verification planning methodology codified within the tool. As the example shows, the verification plan can start as a regular Microsoft Excel spreadsheet that uses a predefined format to identify test structure and specific features. In addition or alternatively, users can utilize a specific-purpose planning environment, in our case, the vPlanner™ feature, a functionality fully encapsulated within the Incisive vManager™ solution.



	Features	Description	Owner	Priority
2	Interfaces	Interface features		
3	APB	APB Inferface	John	1
4	Transfers	APB Traffice support		
5	UART	UART Interface	Paul	1
6	Rx	Rx Config and Traffic		
8	Tx	Tx Config and Traffic		
10	Core Features	Core Block Features		
11	Receiver	UART Traffic Receiver	Nili	2
12	Idle-Busy	Flow Control logic		2
13	Overflow/Underflow	Exception logic		3
14	Transmitter	UART Traffic Transmitter	Nili	2
15	Idle-Busy	Flow Control logic		2
16	Overflow/Underflow	Exception logic		3
17	Fifos	Feature D Behavior	Dan	3
18	Rx	Rx fifo buffer feature		
19	Tx	Tx fifo buffer feature		
20	Code Coverage	Structural Coverage		
21	Dut Code Coverage	Structural DUT Coverage		
22	Block	DUT Block coverage		
23	Expression	DUT Expression coverage		

Figure 1: vPlanner feature for specific-purpose verification planning

The vPlanner feature is used to identify abstract features and hierarchies of features that closely resemble the hierarchy of the specification. While not mandatory, a verification plan is often a useful convenience that helps designers and verification engineers communicate. Verification plans can be hierarchical and integrate other (sometimes commercially supplied) verification plans.

In the case of the Incisive Verification Kit—the UART has an ARM® AMBA® APB interface and, therefore, the plan includes an APB verification plan. This mechanism enables reuse of plans that relate to standard interfaces or perhaps blocks of IP that are reused across multiple projects. It can also be useful for segregating information such as from block level to system level, or at different milestones where different goals are required at various development stages.

The vPlanner feature creates executable verification plans or vPlans. Plans become executable when they are able to re-direct the verification activities, and when they are a dynamically updated thru the verification process. This approach allows resource decisions to be made in an informed way such that progress toward closure proceeds according to items with the highest priority. The coverage is accumulated from any number of runs of a verification flow and from a variety of verification engines (formal, simulation, acceleration, emulation, or other).

MDV Overview

As shown in Figure 2, the metrics in MDV can start as simple "test case coverage" and extend to plan-based closure of design features using the notions of an executable verification plan. As one would expect, increasing investment in automation provides equally increasing productivity benefits from the perspective of time saved or product quality. The key point is being able to start when the project starts, with something completely automatic and unobtrusive, and grow the verification features according to need.

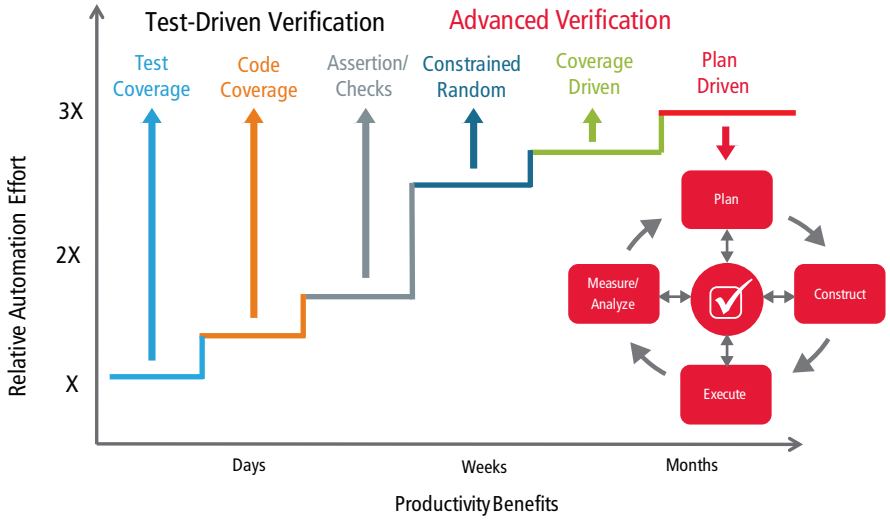


Figure 2: MDV investment alternatives for RTL verification

Test-driven verification is RTL-based and design-centric. It is a very automatic and unobtrusive way to collect test coverage, by measuring which tests have passed and which have failed, and performing failure analysis (first failure, route cause, failure modes). Next, adding code coverage is as simple as turning on a switch, but now you have visibility to see which parts of the code have been exercised, improving the overall quality of the Verilog or VHDL design. There are many ways that the RTL code structure can be automatically analyzed, from block coverage, toggle coverage, expression coverage and finite state machine coverage. Finally, adding assertion-based functional testing with PSL or SVA is a natural fit for additional testing of functionality. Test-driven verification environments also adds the ability to instantly have reports with tracking progress over time, providing substantial benefits over the traditional spreadsheet-based tracking of verification progress.

Advanced RTL verification techniques combine constrained random coverage and plan-based techniques to the RTL verification effort. Constrained random testing has been proven to be able to identify more bugs faster. However, it is difficult to see what has been actually tested because the testbench is by definition randomized. Coverage driven is added to visibly see what is tested and what is not during this randomization process. While relatively easy to add functional coverage, it can produce huge amounts of coverage data, which can be overwhelming when trying to analyze which design features belong to which coverage points. This overwhelming amount of data is why coverage driven has many limitations that affect its usability and scalability.

Plan-based MDV broadens the scope of what is captured and measured to include checks, assertions, software, and time-based data points that are encompassed in the term "metrics." The plan-based notation means that you can organize your verification plans by features, by milestones, by design hierarchy, or all of the above including your own definitions.

MDV creates feature hierarchies organized by the executable verification plan (vPlan). This plan helps manage the wealth of data captured by all the tools involved in the execution of a verification project. It also supports the notion of many-to-many relationships that occur within a feature and how that feature is covered with various testing mechanisms. This traceability is especially important in requirement management flows, such as the automotive international standard ISO 26262, where tracking to requirements is an essential attribute.

For reference, we can see how metrics are gathered at the UART level from multiple runs. Figure 3 shows this executable vPlan that comprises metrics from a number of different environments and different verification engines.

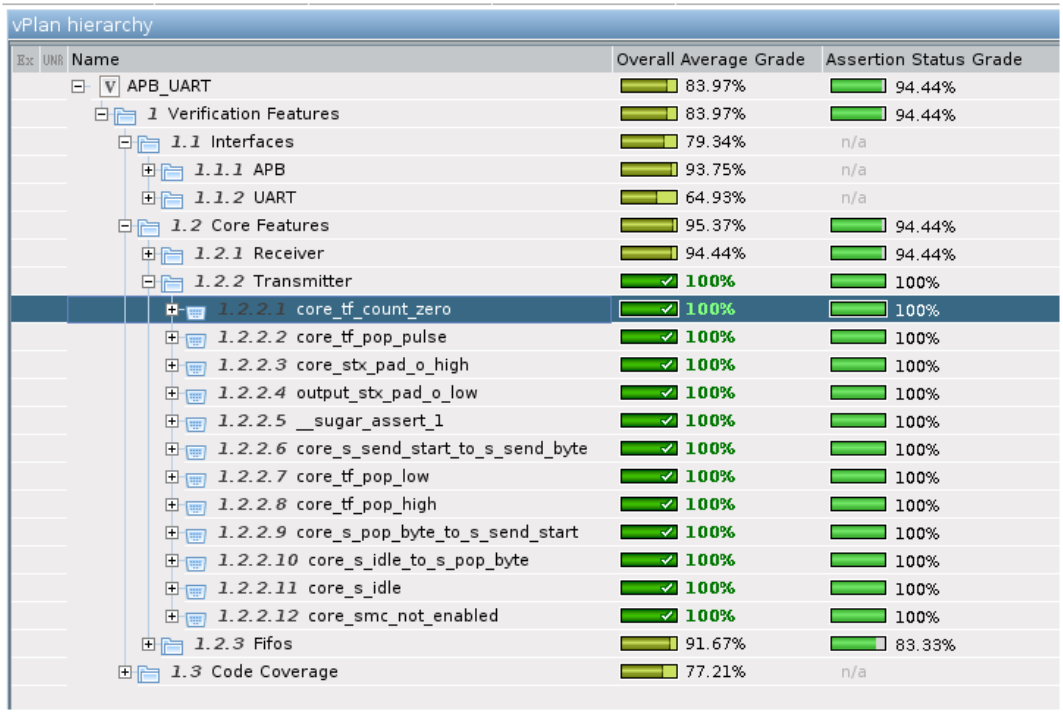


Figure 3: vPlan with coverage metrics

Building a Strong RTL Testbench Foundation

All of the high-level capabilities of MDV that deliver the abstracted and filtered view of the verification process amount to nothing without a common framework under which verification environments can be constructed and reused. The UVM has become the de-facto standard for the construction of verification environments. The UVM is based on its popular predecessor, Open Verification Methodology (OVM), itself an evolution from the e Reuse Methodology (eRM), which has been in widespread use since 2002 on thousands of projects. The UVM supports both e and SystemVerilog and enjoys industry-wide support for both.

The Incisive Verification Kit completely adopts the UVM and includes real-world testbench examples in both e and SystemVerilog. The kit itself includes all aspect of the UVM Reference Flow, but adds many valuable capabilities. At the block level, the kit shows how MDV can be applied to the verification of a UART design. It shows how a verification plan can be constructed upfront, how a UVM environment can be rapidly constructed using Universal Verification Components (UVCs), and how simulations can be run and coverage annotated against the verification plan to monitor and manage the verification process.

Figure 4 details the architecture of one of the kit testbenches, showing the hook-up of the UVCs to the UART DUT. One of the major strengths of the UVM is its approach to reuse. Improving productivity is all about writing complex testbench elements—once and only once—and thereafter reusing these components as widely as possible. One of the dimensions of reuse engineered into the UVM is from block to cluster.

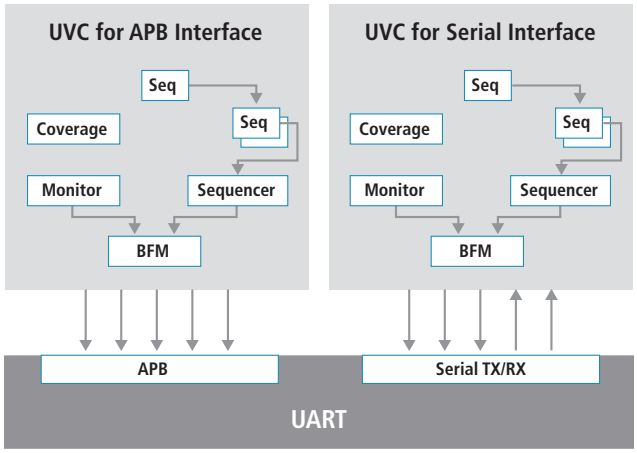


Figure 4: Details of a kit testbench architecture

Figure 5 shows the cluster-level testbench for the kit APB subsystem, clearly illustrating the large amount of reused content. Notice the reused serial interface UVCs and the reused APB UVCs.

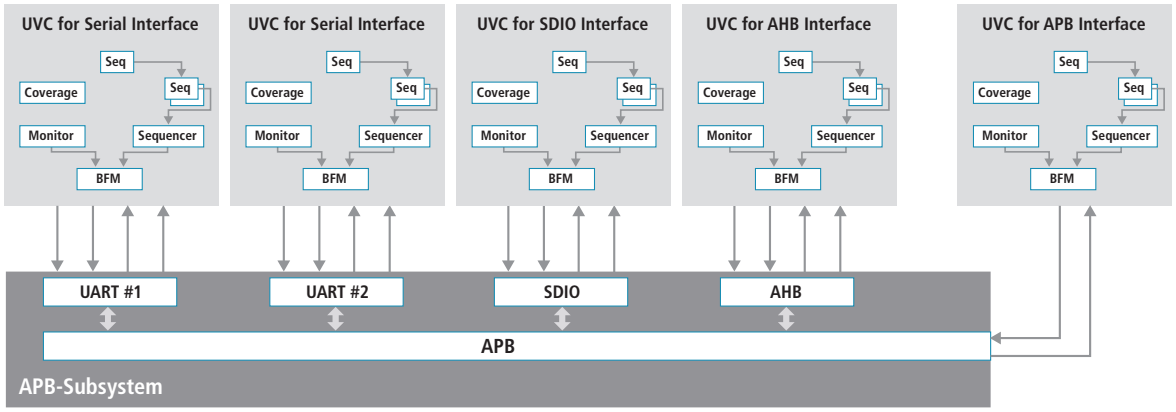


Figure 5: Cluster-level testbench for the kit APB subsystem

One of the key purposes of the kit is to accelerate understanding of UVM concepts, with testbench architecture diagrams, workshops, and hands-on labs. The modular, layered approach also forms the basis for reuse. In addition to the creation of plug-and-play hardware and software verification components that can be reused from block to cluster to chip to system, UVM components can also be reused across multiple projects and platforms.

As there is a frequent need for verification components for standard interfaces such as USB, PCI Express®, AMBA AXI, and so on, Cadence has created a comprehensive portfolio of commercial verification IP (VIP) components. These components are of particular interest when it comes to "compliance testing" (ensuring that the design's interfaces fully comply with their associated specifications). Furthermore, each of these VIP components comes equipped with its own pre-defined executable verification plan that can be quickly and easily incorporated into a master plan. A subset of the VIP portfolio is demonstrated within the Incisive Verification Kit for reference.

Simulation Isn't the Only Way

MDV uses a generic approach that doesn't mandate any specific verification execution engine; in other words, simulation isn't the only tool you can choose to use for establishing design correctness. Figure 6 illustrates how plan-based MDV executes across verification engines, providing coverage interoperability, plan interoperability, and transparent verification interoperability.

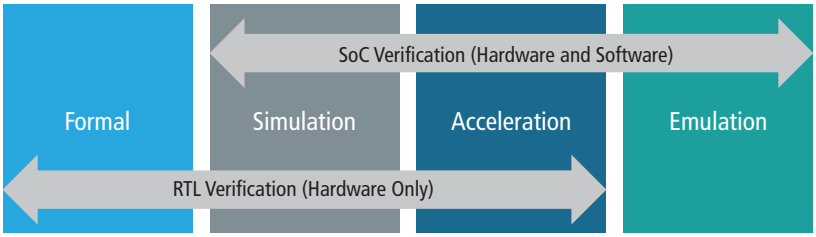


Figure 6: MDV across heterogeneous verification engines

For example, formal tools are another means by which design correctness can be established. Formal properties in the form of assertions are captured and the tools attempt to exhaustively prove that, for all possible circuit states, the property is true. While the user typically writes these assertions manually, automatically extracted assertions and assertion libraries can also accelerate the verification process. The Incisive Verification Kit provides examples of these flows and shows how the checks and assertion coverage points are easily included in the verification plan. The results from the formal process are then annotated onto the plan.

MDV seamlessly provides verification engineers and managers with a consistent view of the verification progress, regardless of where the results are coming from. While exhaustively proving assertions is a goal, there are frequently occasions where the tools need additional guidance to complete their proof. Typically, the user writes additional assertions that define constraints. These assertions narrow the scope of the formal proof, and the tools are then able to complete the proof. The question is: how do you know that these assertions are valid?

Assertions are not only used for formal proof, but will also execute in simulation. By reusing the constraints in your simulation runs, you have a means of cross-checking whether the assertions you made still hold true in your simulation models. This mechanism isn't watertight, as users might not exhaustively simulate the environment in all possible scenarios. However, if you adopt constrained-random simulation as espoused by the UVM, you increase your chances of hitting the corner-case where the assumptions are incorrect.

MDV provides a mechanism for users to include the coverage of the constraints, which, at a minimum, will identify that a constrained condition was triggered. You always have visibility into failures, so the coverage you get gives you a useful cross-check of these constraints. These techniques from both the planning and execution side are included as workshops within the kit.

SDV Overview

As embedded software content and complexity grows, there is more demand that deeply embedded products should work reliably while powered on over longer and longer periods. The impact of software quality on product reliability will continue to rise in the same way that hardware quality has become mandatory for any new SoC design. Deeply embedded software quality is a major concern for complex SoC products, especially as multi-core designs become more prevalent. Thus, the application of a rigorous, scalable quality measure such as MDV will become the norm.

Utilizing the MDV methodology for RTL verification is well proven and utilized by most chip suppliers. It operates without the need for a processor to be present, allowing IP teams to start verifying long before the processor is available. Unlike RTL verification, software-driven verification (SDV) utilizes a top-down software-based approach for verification of the chip functionality, or verification of the embedded software functionality. This approach takes advantage of the fact that the processor is available to stimulate the design. C or C++ tests are run to mimic the various use cases the SoC is required to execute, transactions from the processor are monitored, and both RTL coverage and software coverage can be collected.

The same MDV concepts of randomization, planning, coverage and regressions can apply equally from RTL-based bottom-up to top-down SDV approaches. SDV complements RTL hardware verification by more closely exercising the user view of the application. SDV can use RTL or transaction-level models (TLMs) for improved execution performance, and can also utilize higher performance acceleration or emulation engines. The Cadence MDV approach supports these notions of multiple heterogeneous engines and different levels of design abstraction, as well as different stimulus and collection interfaces to the design.

Low Power Isn't Just the Designer's Problem

Low-power design is no longer a "nice to have," but is mandatory even for non-mobile applications, and it is a crucial "must have" for all "green" products. But as a verification engineer, why should you care about low power?

The low-power verification challenge has two new dimensions. Firstly, the physical implementation changes can introduce functional bugs as a side effect. These bugs can only manifest themselves during cornercase scenarios. Secondly, the interaction between the power-management software and the hardware requires accurate verification to ensure that all power modes and power-mode transitions have been verified in all operating circumstances. Even a design with just a few pieces of IP that can be powered down can have thousands of operational states and power transitions that need exercising.

The Incisive Verification Kit provides comprehensive material and examples covering verification of a hardware platform containing a realistic set of power management features. It also contains workshops on how to use an MDV approach to hardware/software co-verification. Figure 7 shows a simplified architecture diagram of the kit platform including a power control module (PCM).

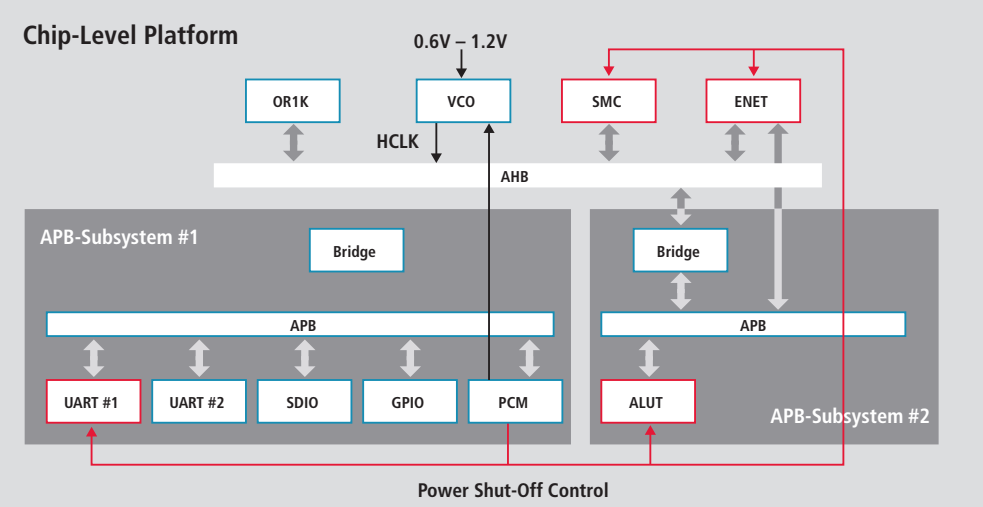


Figure 7: Architecture diagram of the kit platform including a PCM

Reuse Isn't Just About Testbench Components

Reusable UVCs perform a vital role in accelerating testbench development. These components can comprise complex developments, and commercial availability of UVCs for standard protocols provides substantial leverage for assembling complex testbenches. In all of this detail, engineers sometimes overlook the fact that many other pieces of the verification process can be reused in multiple dimensions.

UVCs provide project-to-project reuse at multiple levels. They also provide block-to-cluster reuse, which substantially boosts productivity when assembling a testbench for a hardware platform for which sub-component testbenches exist. Not only can the components be reused, but also sequence libraries, register definitions, and verification plans. The Incisive Verification Kit has comprehensive cluster-level environments that fully illustrate how to apply these reuse techniques. Figure 8 shows the architectural overview of a chip-level testbench and its corresponding chip-level vPlan, showing all of its verification plan reuse.

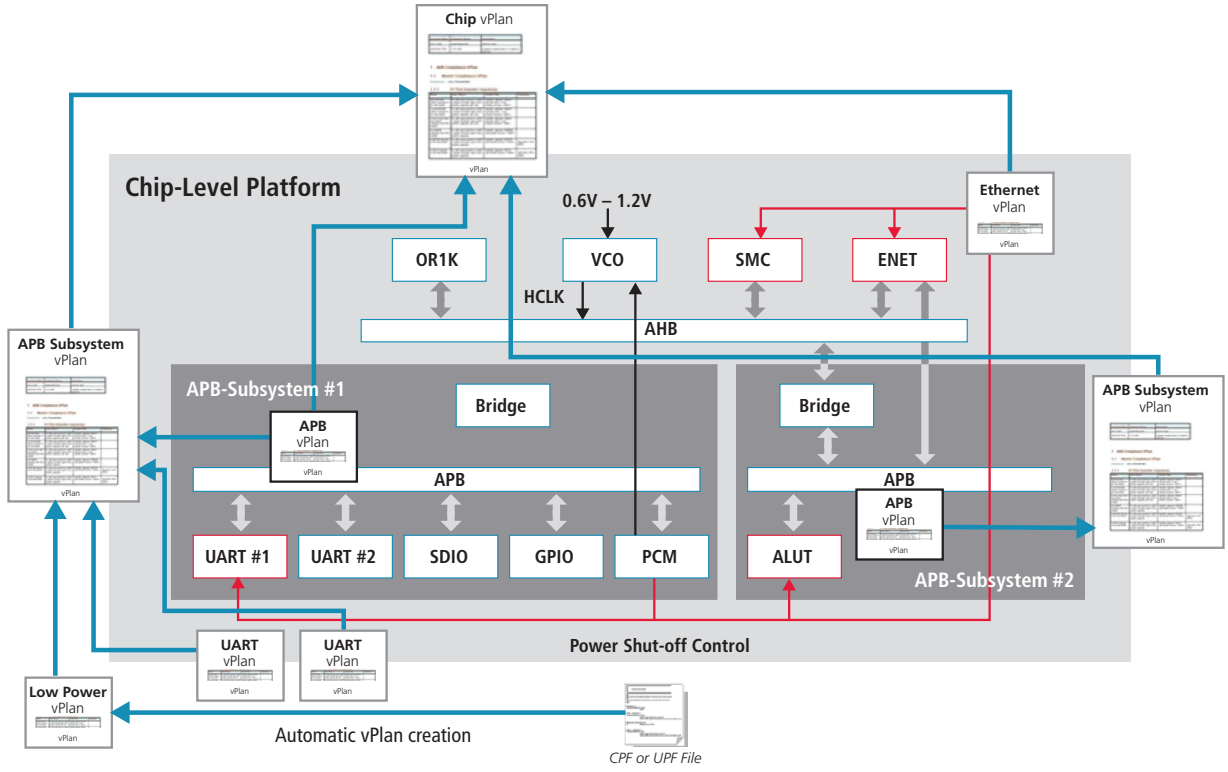


Figure 8: Architectural overview of a chip-level testbench and its corresponding chip-level vPlan

Does Speed Matter?

As a verification engineer, which of the following does your manager care more about: how fast your simulator runs or whether you achieve coverage closure on schedule? MDV enables sophisticated analysis of the coverage from the complete regression runs and allows you to correlate the results for a particular feature against the execution of the verification. In other words, MDV helps you answer the question, "Which simulations should I re-run in order to quickly test feature Y in two hours?"

Effective verification is not about running the most simulations in the least time. It is about running the most effective simulations as much of the time as possible. Simulations that add to coverage are the most valuable, as they are more likely to hit corner-case scenarios and therefore find bugs. MDV enables you to identify the random seeds and tests that contribute the most overall coverage, or coverage for a particular feature or group of features. Running the whole regression suite more and more is like using a sledgehammer—eventually you will hit the target but it will take a lot of effort. Conversely, MDV is like putting a laser in the hands of the verification engineer; it can be focused and fired into specific areas where coverage holes exist. Figure 9 shows the analysis of coverage contribution for each run of a regression against a specific vPlan feature.

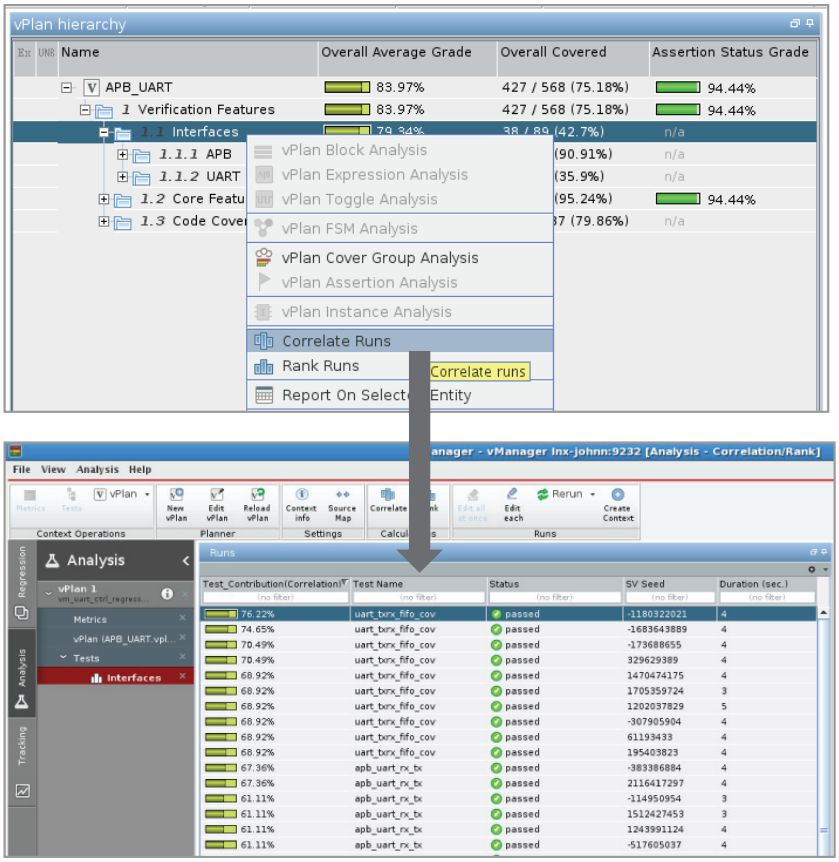


Figure 9: Analysis of coverage contribution for each run of a regression against a specific vPlan feature

The Incisive Verification Kit provides workshops on how to use the analysis features of MDV so you can quickly get up to speed with this powerful capability.

How Does MDV Scale?

One of the key lessons learned in the development of the UVM is the power of automatic test generation. By having constrained-random tests, each time a test scenario is run with a new seed, it is potentially going to find new bugs as it traverses new states in the design. This scalability has massive potential when allied with infra-structure that supports vast processing farms. Huge numbers of concurrent simulations can be launched by one person, all potentially hunting bugs without the need for engineers to write hundreds and hundreds of directed test cases. This ability to scale the verification task is very compelling as it truly starts to automate the progress toward verification closure.

Figure 10 shows an example of how MDV was applied to a real customer project. Using MDV, the customer not only reduced the project timeframe from 12 months to 5 months but also found more bugs and used less manpower.

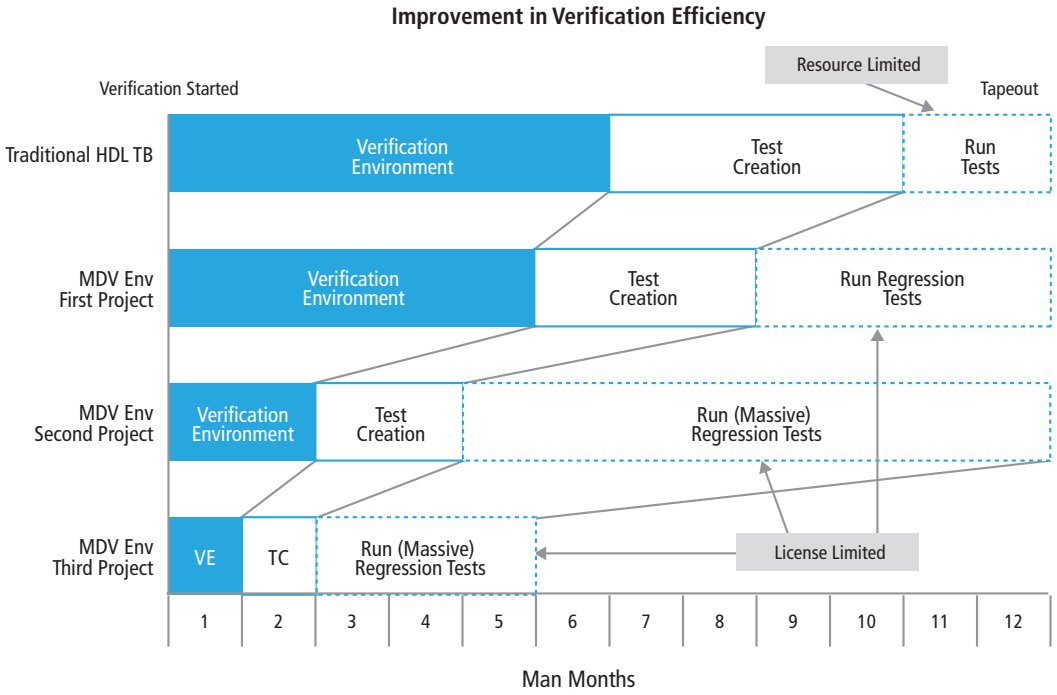


Figure 10: Example of MDV applied to a real customer project

For formal verification, a large set of formal properties traditionally comprise the verification task, and as the design grows and the suite of properties grow, so does the length of formal runtime needed to complete the proof. New features within Incisive Formal Verifier have enabled a scaled "regression" approach to complete these large formal proofs through the use of compute farms. Incisive Formal Verifier has the ability to split a large proof "deck" into a number of distributed smaller proofs, and thereby run the entire regression in a much shorter time. As these types of tasks tend to be run repeatedly as chip signoff approaches, the option of reducing runtime by a factor of 10 or 20 can make the difference between hitting a schedule or not.

How Do I Get Up to Speed with All this New Stuff?

So this new methodology and technology is great, you might say. But how on earth do you get my entire team up to speed in a realistic timeframe that doesn't impact the schedule?

The Incisive Verification Kit contains a large number of comprehensive hands-on workshops, including all the slides and lab instructions, so you can walk through a specific topic in your own time, or perhaps ask a Cadence Application Engineer to run through it with you. The fact that these workshops are pre-packaged and delivered with the tools greatly enhances your ability to manage the ramp-up of teams on selected methodologies, all based on a realistic example SoC.

The current range of Incisive Verification Kit workshops include the following topics:

- Low-power simulation
- Coverage
- UVM
- SystemVerilog design
- Digital mixed signal
- Assertion-based verification
- Verification debug
- VIP
- MDV

Additionally, Cadence has extended many parts of the Incisive Verification Kit to be online with a smaller, more consumable subset of information to get up to speed more rapidly. The Cadence Rapid Adoption Kits covers this subset of workshops and smaller topics thru Cadence Online support.

Summary

MDV is a powerful layer of methodology that sits above the UVM testbench. The Incisive Verification Kit provides a comprehensive set of examples of how MDV can be applied across an entire range of verification technologies in a coherent and consistent way. Once applied, MDV puts powerful capabilities in the hands of engineers and managers. These MDV capabilities, complemented by SDV capabilities, make the crucial difference; they improve verification effectiveness and, hence, managers and engineers alike can maximize their utilization of resources, use the breadth and depth of technology available, improve project visibility, and reduce the number of engineering man months.

Further Information

- Incisive Verification Kit: www.cadence.com/products/fv/iv_kit/Pages/default.aspx
- MDV Overview: www.cadence.com/products/fv/Pages/mdv_flow.aspx
- Incisive vManager Solution: www.cadence.com/products/fv/vmanager/pages/default.aspx
- Requirements Management Flow: http://www.cadence.com/products/fv/Pages/requirements_management_flow.aspx

cadence®

Cadence Design Systems enables global electronic design innovation and plays an essential role in the creation of today's electronics. Customers use Cadence software, hardware, IP, and expertise to design and verify today's mobile, cloud and connectivity applications. www.cadence.com www.cadence.com

© 2014 Cadence Design Systems, Inc. All rights reserved. Cadence, the Cadence logo, and Incisive are registered trademarks and vManager and vPlanner are trademarks of Cadence Design Systems, Inc. ARM and AMBA are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All others are properties of their respective holders. 1906 02/14 SA/DM/PDF