

Maximizing Verification Effectiveness Using Metric-Driven Verification

Authored by Nick Heaton – Senior Solution Architect, 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 scaleable 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.....	1
Failing to Plan = Planning to Fail ...	2
Metric-Driven Verification	3
Building Strong Testbench Foundations.....	4
Simulation isn't the Only Way	5
Low Power isn't Just the Designer's Problem	5
Reuse isn't Just About Testbench Components.....	6
Does Speed Matter?	7
What About Scalability?.....	8
Is Metric-Driven Verification Just for RTL Hardware?	8
How Do I Get Up to Speed with All this New Stuff?	9
Summary	10
About The Author.....	10

Introduction

MDV is a key to Silicon Realization, part of the greater EDA360 vision. EDA360 helps both design creators and design integrators close the “productivity gap” (through improved approaches to design, verification, and implementation) as well as the “profitability gap” (by providing new capabilities for IP creation/selection/integration and system optimization). Silicon Realization represents everything it takes to get a design into working silicon, including the creation and integration of large digital, analog, and mixed-signal IP blocks. Advanced verification capabilities, such as MDV, are a must.

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 and new flows and tools are still being invented. Against this rapidly changing background, designs have been steadily growing with more IP and more complex IP being integrated into larger and larger SoCs.

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 new approach to managing verification complexity—metric-driven verification—and a valuable resource to back up this new approach, the Cadence Incisive Verification Kit, which together will enable you to plan and embrace new methodologies and approaches in a safe and controlled way. The kit provides clear and realistic examples at each stage to guide you and your teams in their new projects.

Silicon Realization is achieved when metrics can be applied to design intent, when users can work more productively because they are working at a higher level of abstraction, and when successive refinements converge to achieve verification closure. MDV manages intent, abstraction, and convergence—leading to greater predictability, productivity, and eventually profitability.

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 in what you can clearly measure. Historically this was a problem with directed testing. Huge lists would be drawn up to define the verification process, but these 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 needs to be verified but does not define the execution—the "how." This is a crucially important distinction as it is expected that verification may be done using multiple technologies by multiple people. We refer to the "what" as the "features" or "goals" that need to be verified—it is 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, and the results are represented by various forms of coverage.

A verification plan should capture the goals of the verification process such that the sign-off criteria are clearly identified as well as milestones along the way. An example of a milestone would be RTL code freeze, which a team might define as having 75% of all features covered by that time.

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.

Figure 1 shows a sample of a verification plan for the UART block in the Incisive Verification Kit environment.

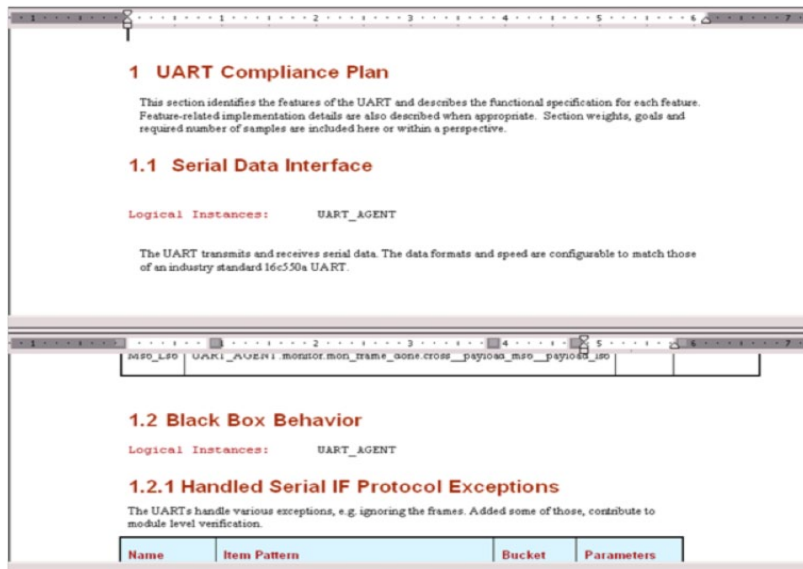


Figure 1: Verification plan for the UART block

As the sample shows, the verification plan can be a regular Microsoft Word document that uses template heading styles to identify structure and specific features. This is easy to create in Microsoft Word or using Incisive Enterprise Planner as the front-end editor of the plan.

The verification plan also identifies abstract features and hierarchies of features that may 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 also include other (sometimes

commercially supplied) verification plans—as in the case of the Incisive Verification Kit—the UART has an 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.

Figure 2 shows an executable verification plan (vPlan) with coverage mapped onto it. This approach allows resource decisions to be made in an informed way such that progress toward closure proceeds according to the original plan. The coverage is accumulated from any number of runs of a verification flow and from a variety of verification engines (formal, simulation, hybrid, or emulation).

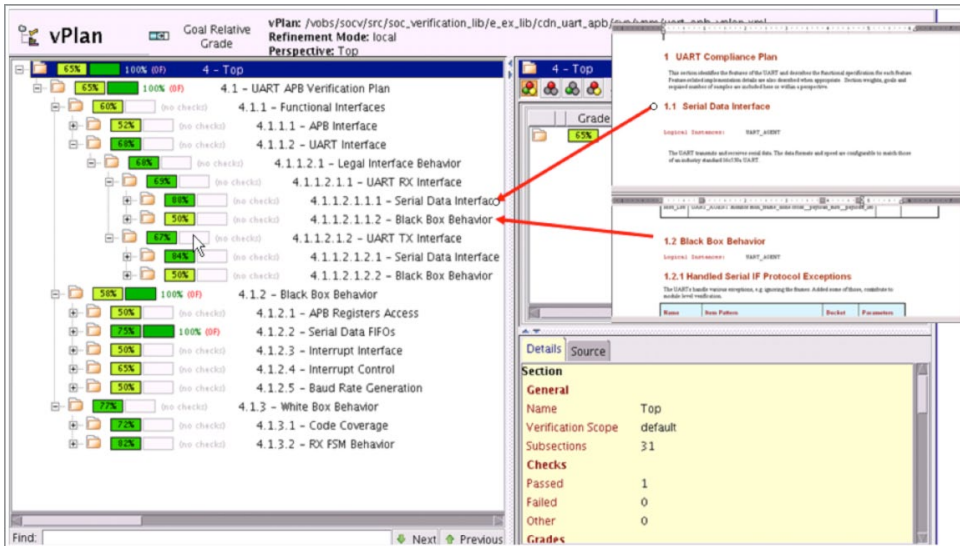


Figure 2: Executable vPlan with coverage

Metric-Driven Verification

Over the past few years, the term "coverage-driven verification" has become widely adopted to refer to the gathering of simulation coverage in all of its various forms, traditionally code and functional coverage. This approach has many limitations that affect its usability and scalability. Once major limitation is that it does not include checking or time-based aspects, which are essential in defining the closure criteria.

Metric-driven verification (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 second enhancement MDV offers over coverage-driven verification is the ability to create feature hierarchies by using an executable verification plan (vPlan). This helps manage the wealth of data captured by all the tools involved in the execution of a verification project.

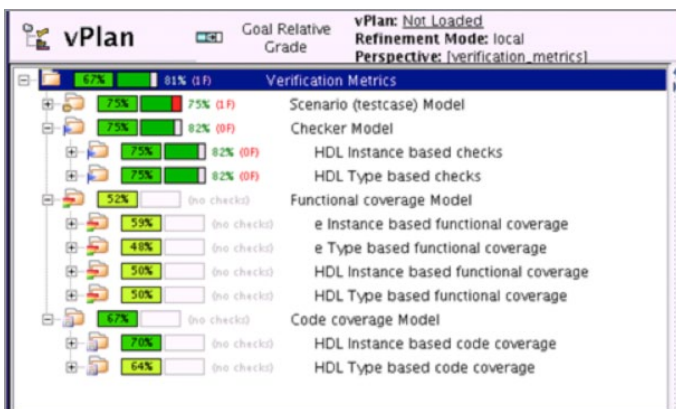


Figure 3: vPlan with metrics

The third enhancement is support for parallel verification execution to optimize verification throughput. One of the primary objectives of MDV is to reduce the dependency on the one resource that is not scalable—the verification engineer. By enabling use of parallel processing via machines through automation, we improve productivity and reduce the number of non-productive tests. Figure 3 contains a vPlan that comprises metrics from a number of different environments and different verification engines.

We can see how metrics are gathered at the UART level from multiple runs of both formal (static) and simulation (dynamic) verification.

Building Strong Testbench Foundations

All of the high-level capabilities of MDV that deliver the abstracted and filtered view of the verification process would amount to nothing without a common framework under which verification environments could be constructed and reused. The Universal Verification Methodology (UVM) has become the de-facto standard for the construction of verification environments. The UVM is based on the Open Verification Methodology (OVM) 2.1.1 release (the OVM itself an evolution from the eReuse Methodology (eRM), which has been in widespread use since 2002 on thousands of projects). The UVM supports all of the important languages customers demand and enjoys industry-wide support.

Given the investment needed in training and development of advanced verification environments, it is reassuring to know that by adopting the UVM you are part of a global community working toward common verification goals. MDV builds on top of the UVM, enables the management of complex SoC developments, and focuses on the key challenges rather than issues of verification environment implementation or reuse.

The Incisive Verification Kit completely adopts the UVM and includes real-world testbench examples in both e and SystemVerilog. 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.

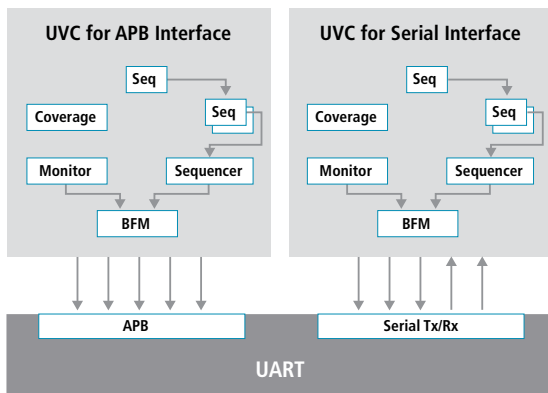


Figure 4: Details of a kit testbench architecture

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 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.

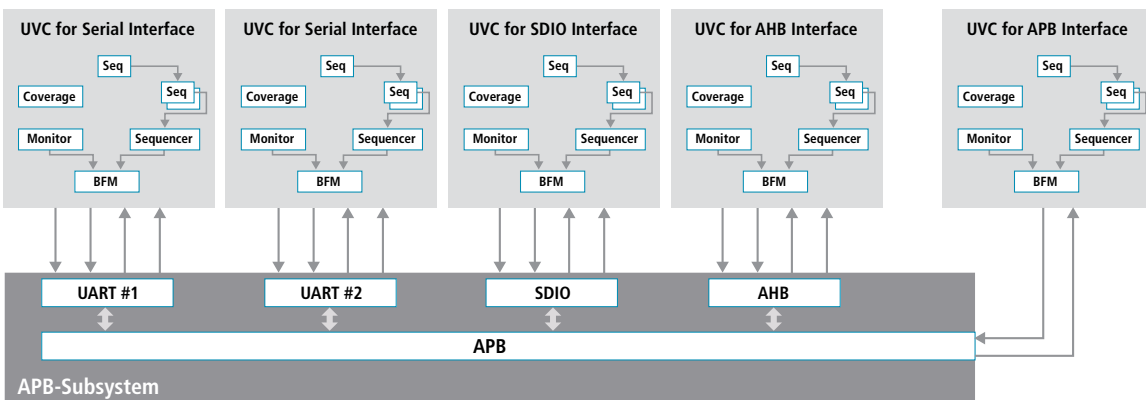


Figure 5: Cluster-level testbench for the kit APB subsystem

One of the key purposes of the kit is to accelerate understanding of UVM concepts, and testbench architecture diagrams like this are available as clickable demos in the kit. These diagrams enable you to quickly familiarize yourself with both the methodology and language usage in the context of a realistic example.

This 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 verification plan that can be quickly and easily incorporated into a master plan. (A subset of this VIP portfolio is demonstrated within the Incisive Verification Kit).

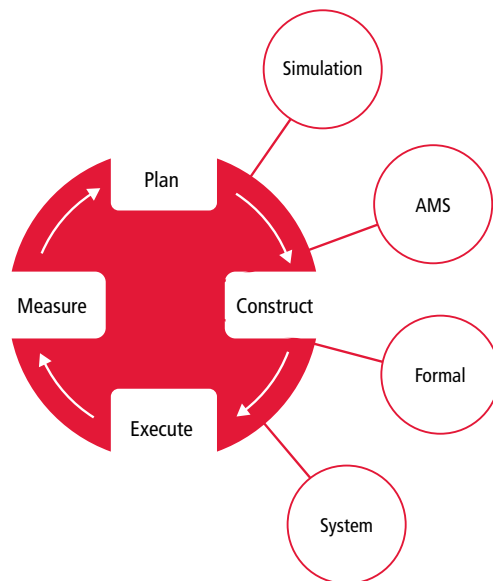


Figure 6: MDV changes an open-loop verification process into a closed-loop process

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 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 assumptions 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 assumptions you made still hold true in your simulation models. This isn't a watertight mechanism, as users may not exhaustively simulate the environment in all possible scenarios. However, if you adopt constrained-random simulation similar to the UVM, you increase your chances of hitting the corner-case where assumptions may have been 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.

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?

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 may choose to use for establishing design correctness. Figure 6 illustrates how MDV changes an open-loop verification process into a closed-loop process through a Plan-Construct-Execute-Measure cycle using the results of multiple 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

In the past, the verification engineer would verify that the RTL implementation precisely matched the design specification, and while that still holds true, it is only part of the problem. With the addition of new low-power techniques that capture power intent in side files, the “golden” design is no longer just the RTL, but is a combination of the RTL plus the power intent. While the power intent defines the physical implementation details, there is also a power control module (PCM) that is—almost without exception—under software control.

The low-power verification challenge therefore has two new dimensions. Firstly, the physical implementation changes can introduce functional bugs as a side effect. These may only manifest themselves during corner-case scenarios. Secondly, the interaction between the power management software and the hardware requires careful 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).

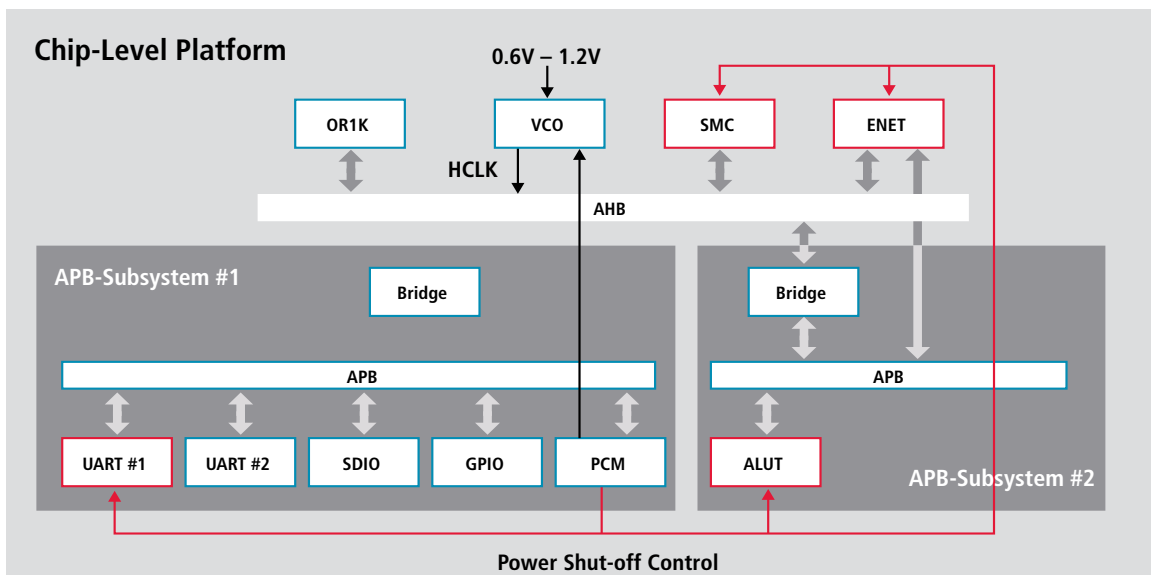


Figure 7: Architecture diagram of the kit platform including a power control module

Reuse isn't Just About Testbench Components

Universal Verification Components (UVCs) are reusable verification components that perform a vital role in accelerating testbench development through reuse. These components may 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 may 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.

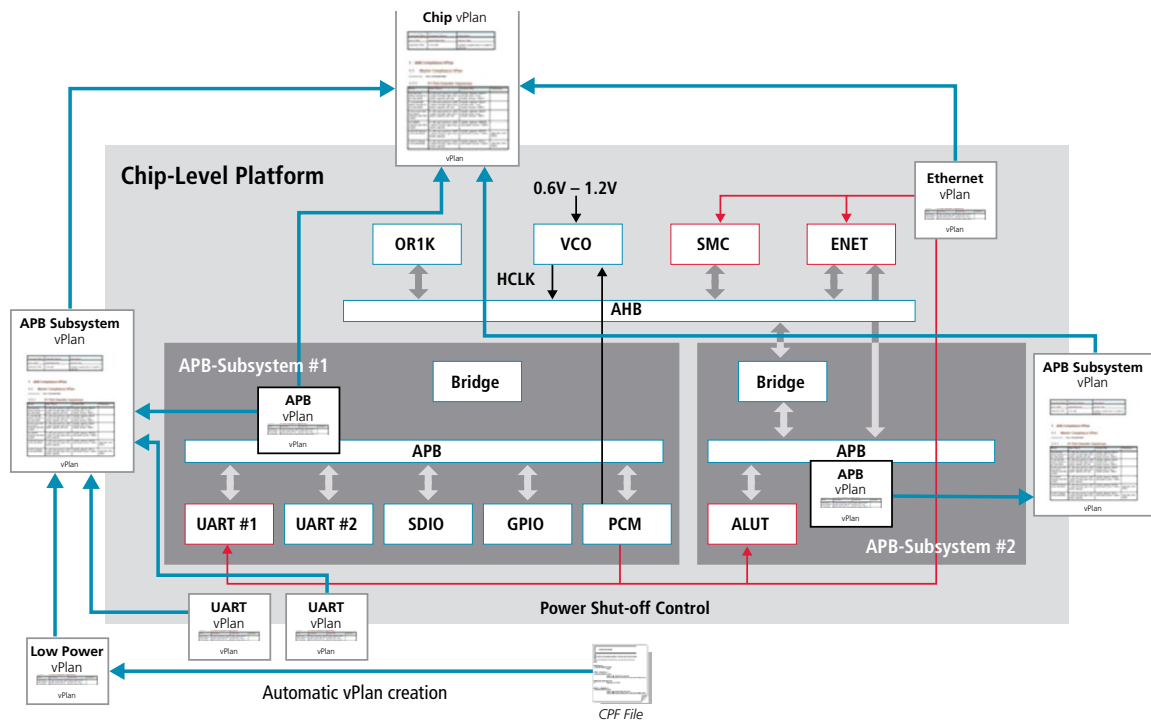


Figure 8: Architectural overview of a chip-level testbench and its corresponding chip-level vPlan

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 2 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.

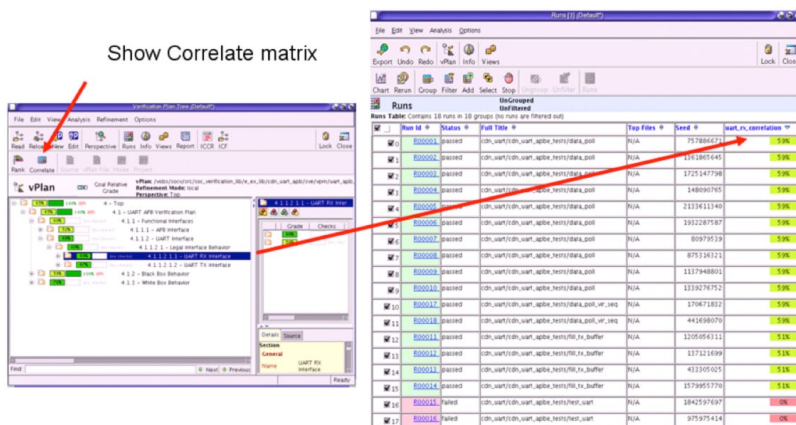


Figure 9: Analysis of coverage contribution for each run of a regression against a specific vPlan feature

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.

What About Scalability?

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 may traverse new states in the design. This scalability has massive potential when allied with infrastructure 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.

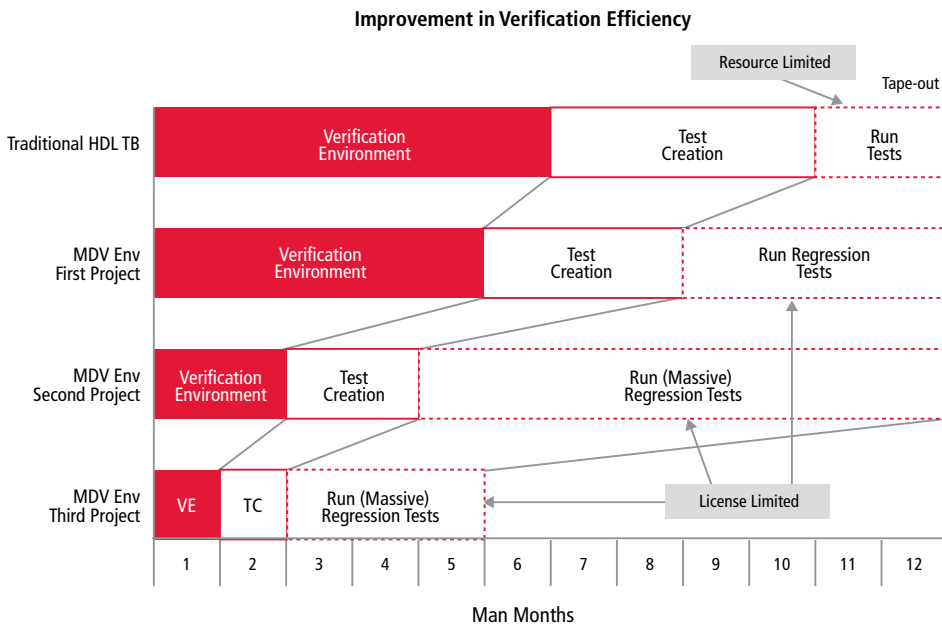


Figure 10: Example of MDV applied to a real customer project

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 run time 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 sign off approaches, the option of reducing run-time by a factor of 10 or 20 can make the difference between hitting a schedule or not.

Is Metric-Driven Verification Just for RTL Hardware?

Metric-driven verification is a structured and comprehensive approach to defining verification goals and measuring and analyzing progress toward those goals. It therefore is not confined explicitly to verifying hardware, but represents a by-product of the escalating cost of failure when developing and implementing complex silicon devices. And so it’s perfectly reasonable to ask, “What systematic and comprehensive approaches do embedded software developers use to ensure adequate quality levels?” and “Could MDV be applied to embedded software?”

The answer to these questions is yes. MDV techniques can easily be applied to embedded software, and the Incisive Verification Kit contains workshops and example flows of how to implement such technology. The techniques demonstrated show how to verify the software interface to the hardware, in a hardware-software co-verification flow.

One of the key points to keep in mind is the difference between validation and verification. Systems are typically designed and validated top-down, validation being the process to ensure that you are developing the right product. This is very much a software-oriented challenge, but still requires some form of hardware model—usually a functional virtual prototype built from transaction-level models (TLMs). Verification is usually performed as a bottom-up process that exhaustively proves the product has been developed right. Verification is normally performed to differing degrees on multiple abstraction levels. In other words, it is exhaustive verification at the block level, but perhaps integration and the programmer’s view might be verified at the cluster or system levels.

Figure 11 shows the continuum that exists between hardware/software, verification, and validation, and the differing levels at which MDV might be applied.

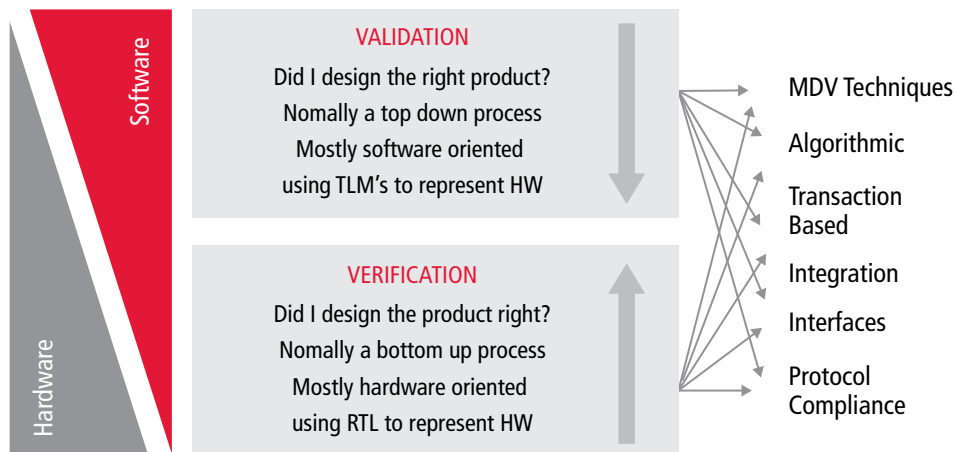


Figure 11: The continuum that exists between hardware/software, verification, and validation

As embedded software content and complexity grows, there is more demand that deeply embedded products should work reliably while always 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.

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, such that it 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 workshops delivered includes:

Assertion-based verification	<ul style="list-style-type: none"> • Introduction • Incisive Formal Verifier productivity flows • Incisive Formal Verifier connectivity flows
UVM – SystemVerilog	<ul style="list-style-type: none"> • Module-based SystemVerilog • Class-based SystemVerilog • Introduction to register modeling with reg_mem

UVM – Specman®/e	<ul style="list-style-type: none"> • Introduction to Specman • Introduction to e • IntelliGen debugging
UVM – mixed language (SystemVerilog and e)	<ul style="list-style-type: none"> • SystemVerilog over e using OIG • SystemVerilog with SystemC® TLMs
Metric-driven verification foundations	<ul style="list-style-type: none"> • Introduction –Metric Assertion • Planning • Infrastructure • Management • Automation
Metric-driven verification using verification IP	<ul style="list-style-type: none"> • Using multiple UVCS • Using the Compliance Management System • HW/SW co-verification

Summary

This paper introduced the concept of metric-driven verification (MDV) and explained how it is a powerful layer of methodology that sits above the Universal Verification Methodology (UVM). 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 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.

About The Author

Nick Heaton is an ASIC and EDA veteran with more than 25 years of experience in the design and verification of complex SoCs. Nick graduated from Brunel University, London in 1983 with First Class Honors in Engineering and Management Systems, initially working as an ASIC designer for ICL in Bracknell. In 1993, he founded specialist ASIC Design and Verification Company Excel Consultants, servicing customers such as ARM® and Altera. In 2002, Nick joined Verisity as Manager of Northern European Consulting Engineering.

Nick currently works in the Cadence Research & Development organization as a Senior Solution Architect with special responsibility for the Incisive Verification Kit, a complex and realistic golden example of verification methodologies and technologies across all of Cadence front-end products..



Cadence is transforming the global electronics industry through a vision called EDA360. With an application-driven approach to design, our software, hardware, IP, and services help customers realize silicon, SoCs, and complete systems efficiently and profitably. www.cadence.com