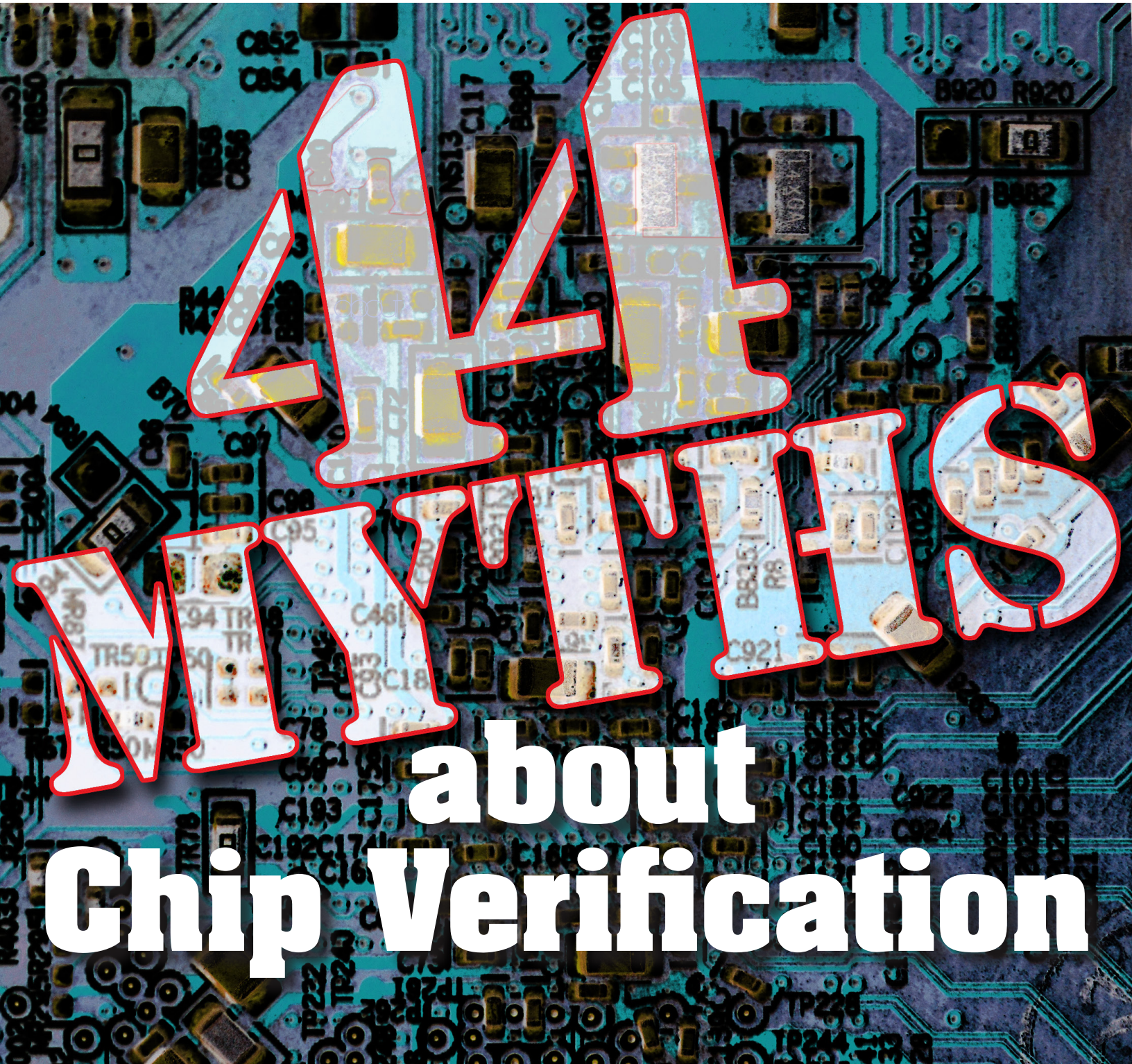




Electronic Design[®] LIBRARY

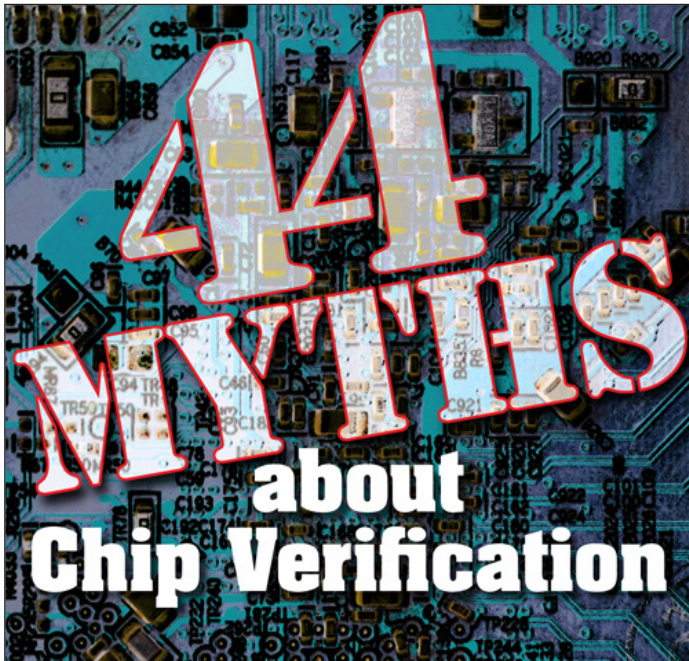
Copyright © 2023 by Endeavor Media.
All rights reserved.

A compendium of articles from *Electronic Design*



4411S

about Chip Verification



INTRODUCTION

COMPANIES AND DEVELOPERS

want their chips to work the first time they are created but this means that their design and verification must be done first. Simulation and verification tools to check everything from timing to thermals make this possible but there are many myths and misunderstandings about the tools, their capabilities and their operation.

We debunk a few of the myths in this ebook that consists of four 11 Myths articles. The 11 Myths series has been very popular on *Electronic Design* and these are not the only ones on the website that address chip verification so we may have another book like this in the future. What this does indicate is that there are a lot of details about the verification process that many engineers and developers might misinterpret or misunderstand. Knowing these will help engineers ask more questions and get more answers so they can readily employ the verification tools that are quite complex. We hope you enjoy this myth series and check out the additional ones we have online.



Bill Wong
Editor,

*Senior Content
Director, Electronic
Design & MWRF*



02 CHAPTER 1
11 Myths About Verification IP



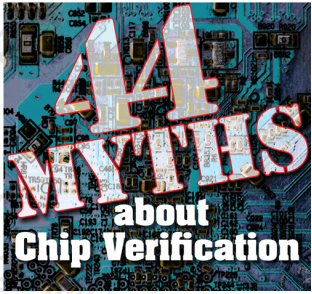
09 CHAPTER 3
11 Myths About Hardware-Assisted Verification



06 CHAPTER 2
11 Myths About Formal Verification



14 CHAPTER 4
11 Myths About Chip Specifications



CHAPTER 1:

11 Myths About Verification IP

BIPUL TALUKDAR, Director of Applications Engineering, SmartDV

To a savvy chip design verification engineer, VIP is much more than a catchy acronym. Designers understand that verification intellectual property is a mainstay of the verification flow with libraries of reusable verification components and pre-defined functional blocks to accelerate verification sign-off.

It's time to put to rest 11 of the most common myths about verification intellectual property (VIP). [SmartDV's](#) Bipul Talukdar, Director of Applications Engineering, explains why it's used in a verification environment to improve debug, coverage closure, and quality; accelerate project delivery; increase return on investment; and reduce the risk of silicon re-spin.

1. Verification IP is an optional part of the verification flow.

Verifying the functional correctness of complex system-on-chip (SoC) designs filled with integrated blocks of IP, many based on complicated industry-standard interface protocols, is a difficult task. That's why verification teams implement Verification IP in their verification strategy and consider it a key component of the verification flow. It ensures debug, coverage closure, and quality are improved, and project schedules are reduced. It creates an infrastructure for industry-standard interface and interconnect protocol support and offers a known reference to compare with the design under test/verification (DUT).

2. Instead of streamlining coverage-driven verification, VIP complicates it.

Verification is complicated no matter what tool is used, and it consumes an estimated 60-80% of a project's resources. A testbench for a complex SoC requires a variety of Verification IP to verify system-level functionality and validate target performance by generating application-specific traffic and checkers.

The use of quality VIP removes the requirement for the designer to become an expert on multiple protocols. The VIP does the "heavy lifting" of verifying the design against the details of the protocol specification. It generates comprehensive tests that stimulate and verify different interfaces and standard bus protocols, shortening SoC verification and increasing test coverage. It includes transactions/sequences, drivers, configuration components, a test plan for a specific interface, and test suites to connect to a DUT inside the testbench to simulate or emulate an IP or SoC design.



VIP comes as industry-standard-compliant, plug-and-play modules that verify system-level functionality and validate target performance by generating application-specific traffic.

3. Verification IP is just another verification methodology.

VIP isn't a verification methodology. It's unlike either the universal verification methodology (UVM), an Accellera interoperability standard for building testbenches, or the Open Verification Methodology (OVM), a methodology and block library. Verification IP is a valuable component implemented in an available standard verification methodology such as UVM.

4. Verification IP often isn't compatible with verification languages and methodologies, nor is it platform-independent or reusable.

VIP consists of libraries of reusable verification components and pre-defined functional blocks that create an infrastructure to support industry-standard interfaces, hardware-verification languages (HVLs) SystemVerilog and SystemC, and methodologies such as UVM. Quality VIP is platform-independent and created to work across all methodologies and languages.

5. Verification IP isn't portable across verification platforms such as simulation and emulation.

Yes, some VIP is targeted to one simulation and/or emulation platform rather than all commercial platforms.

Transitions from simulation to emulation can be difficult, time-consuming, and resource-intensive. However, some vendors provide platform-independent VIP that works with a coverage-driven verification flow seamlessly across different platforms, including simulation, emulation, FPGA prototyping, and formal verification.

6. VIP is an incomplete solution. The verification group needs to build on additional capabilities.

Verification IP blocks for emulation and FPGA prototyping come as synthesizable register-transfer-level (RTL) code with full API compatibility to move designs from simulation to emulation. These VIP blocks are built-in intelligent debuggers, offering fast compile and system-level simulation run times and fast firmware/software development. The infrastructure framework or testbench comes with stimulus generators, monitors, scoreboards/checkers, and functional coverage models.

7. It's difficult to find Verification IP for new or revised protocols.

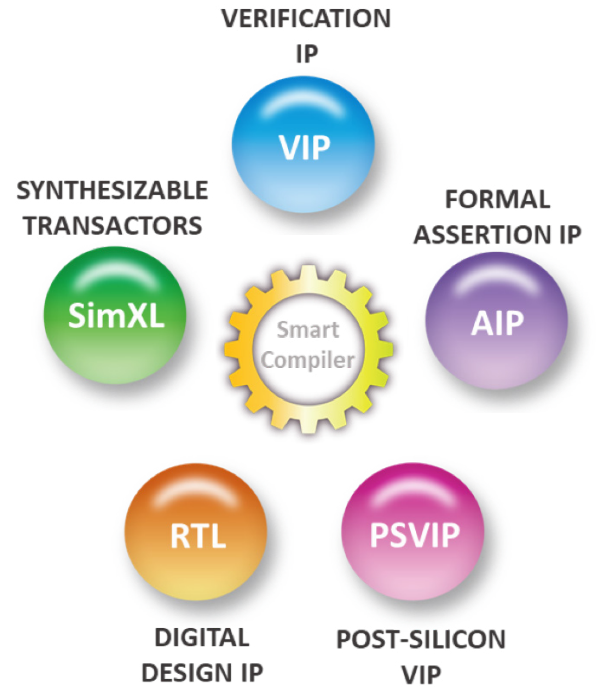
It can be a challenge to find VIP that supports new or revised protocols or open standards, or that addresses the entire verification flow. New or customized VIP development takes time and may not be portable between verification platforms. Specialized compilers that can rapidly create new VIP or customize existing VIP help to cut the time required to introduce new VIP or incrementally customize an existing VIP.

8. Verification IP is slow to compile.

Some but not all VIP can be slow to compile, while others may only target specific

verification languages and methodologies and aren't compatible with others. Platform independence and reusability can be concerns as well. One common complaint is the difficulty verification engineers have analyzing and debugging results. Also, handwritten VIP may not maintain an equal level of performance, efficiency of production, ease of debug ability, and readability.

Verification IP written at a higher level of abstraction that uses a smart compiler/generator (see figure) to produce the final deployable version of the VIP can prove to be the exception against these complaints. Machine-generated VIP can maintain code quality and linting at a prescribed level to efficiently compile and execute. Such VIP can complement today's evolving hardware design methodology that generates RTL code from a higher level of abstraction-coded design specification.



One third-party VIP vendor uses a smart compiler to create verification IP for a range of verification tasks, including formal assertions and synthesizable transactors, post-silicon and design IP, and (not pictured) test suites and documentation. (Source: SmartDV)

9. Not all Verification IP is fully production-worthy and documentation isn't comprehensive.

VIP often is provided by third-party vendors who are active in interface standards development organizations for networking, storage, automotive, bus, MIPI, and display protocols. They can have an advantage over in-house resources because they verify the correct functionality and compliance with the industry standard.

However, verification engineers must research and evaluate VIP from third-party vendors. They need to understand the technical support for VIP usage and customization, if required. Support can be inconsistent, and customization can be costly and time-consuming. Another important aspect of the evaluation process is to be sure the third-party VIP solution is fully compatible with the standard and addresses the entire verification flow.

10. Transitions from simulation to hardware acceleration are difficult, time-consuming, and resource-intensive.

No cohesive Verification IP solution addresses the entire verification flow. In fact, different verification IP is needed for different steps in the process; there are often gaps in the type of verification coverage offered by a VIP. For example, VIP for simulation may exist without a good solution for emulation, formal assertion verification, and/or even post-silicon verification. Dedicated VIP vendors can fill in the gaps.

One popular myth proposes that the best source for VIP is from the same vendor that




provides simulation and emulation platforms. This may or may not be true. The primary business of these vendors typically focuses on software and hardware sales required to build the verification environment. While VIP is offered, it's not the core focus. Instead, it's an investment made to support selling the more expensive software and hardware platforms. Vendors that focus on developing and licensing VIP often have broader offerings and better support.

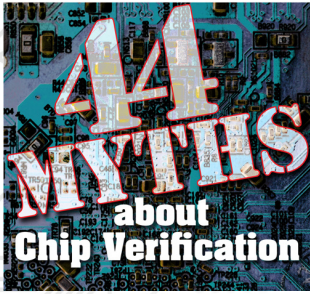
11. Test suite availability for a delivered VIP can be limited.

Creating test suites to cover 100% of a protocol can be a daunting job due to the range of scenarios that tests need to cover and the number of tests that need to be produced. Such a task can be a burden for VIP providers whose main business isn't to create VIP, but rather to create VIP as enabling solutions for their tools. This isn't true for companies specializing in the production of VIP. They are experts and machined the process to produce required test suites for 100% protocol coverage paired with the required documentation.

BIPUL TALUKDAR is SmartDV's director of Applications Engineering, North America. He is an expert in hardware functional verification with a specialty in Verification IP development, formal property verification, and hardware emulation. His recent experience is in formal verification of RISC-V-based cores and subsystems and coverage-based closure. He holds a Bachelor of Science in Engineering, Electronics and Telecommunication from the National Institute of Technology, Silchar, India.

to view this article online,  [click here](#)

 [BACK TO TABLE OF CONTENTS](#)



CHAPTER 1:

11 Myths About Formal Verification

TOM ANDERSON, Technical Marketing Consultant

Formal verification is used by almost every chip development and verification group, though myths about it persist and may deter engineers who could benefit from its value.

Formal verification, which uses mathematical analysis rather than simulation tests, has been available in commercial EDA tools for more than 20 years and in academia much longer. As with many new technologies, initial adoption was slow and limited to companies who had in-house formal experts. This has changed dramatically in the last dozen years or so. Almost every chip-development team makes some use of formal tools, and the market continues to grow. Nevertheless, some myths about formal persist, and they may still be deterring some engineers who could benefit from it. It's time for the truth to be told.

The main attraction of a formal methodology is clear: Exhaustive analysis of a semiconductor design. Simulation provides only scattershot verification by its very nature. No matter how long simulation (or emulation) runs, only a tiny portion of possible design behavior will be exercised. Unverified scenarios may hide serious bugs.

Since formal verification is exhaustive, it considers all legal design behavior over all time throughout the entire design. It finds corner-case design bugs, but also proves that no bugs are remaining. It relies on assertions about intended design functionality and constraints to keep the analysis restricted to legal behavior. This ensures that no false “bugs” are found by violating the input rules or protocols for which the chip was designed.

1. Formal verification can only be performed by PhDs.

This was largely the case for the early academic tools targeted at research projects rather than industrial applications. Today, proving end-to-end properties for large designs may require significant formal expertise, though not a PhD in mathematics. Moreover, many applications of formal analysis are being used every day by designers and verification engineers who have no special training. The tools, languages, and methodologies have all improved a great deal since the pioneer days.

2. It takes a lot of work to get results from formal tools.

This is another myth that applied to early tools, but is no longer the case. Electronic design

automation (EDA) vendors offer a wide range of formal applications (apps) that run automatically on the design and deliver results immediately. These include such important verification challenges as connectivity checking, tracing clock and reset networks, avoiding propagation of unknown values, and analyzing the effects of random faults during chip operation.

3. Formal results require writing lots of assertions and constraints.

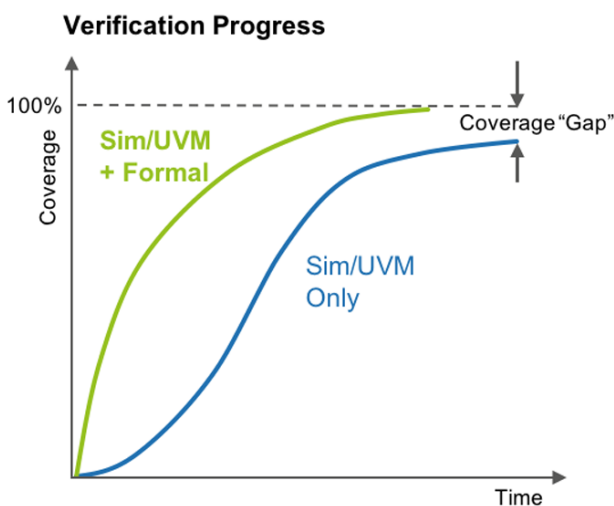
One of the advantages of formal apps is that they don't require any assertions to be written by the development team. These tools generate their own assertions from the design and related supplemental files. Just as with user-written assertions, the generated assertions are analyzed to find design bugs and then prove that no further bugs exist. In some cases, the analysis can be refined by user-provided constraints, but these tend to be quite simple.

4. Assertions and constraints for formal verification are hard to write.

This myth also applies to early academic tools, which often used abstruse mathematical expressions as inputs. Today, SystemVerilog Assertions (SVA) are a subset of the widely used SystemVerilog design and verification language. Some formal tools also support SVA with designs written in VHDL and SystemC. All designers and verification engineers receive training in SystemVerilog, so expressing assertions and constraints is easy and natural.

5. It's hard to write assertions for complex protocols.

Capturing all rules for a complex interface protocol isn't a simple task, but it has more to do with the complexity of the rules than the format used. Writing a Universal Verification Methodology (UVM) simulation model isn't simple either. Pre-packaged verification intellectual property (VIP) for standard protocols is commercially available for UVM simulations. Formal tool vendors have extended this approach to make standards-based assertion-based VIP available as well (see figure).



The combination of UVM-based simulation and formal verification eliminates the coverage gap associated only with simulation and UVM models.

6. There's no way to know when enough assertions have been written.

This limitation has only been overcome fairly recently, with the "model-based mutation coverage" metrics provided by OneSpin's tools. Other methods of estimating assertion completeness are overly optimistic, resulting in missed bugs. Mutation coverage reports the portions of the design in which a bug would not be detected by any existing assertion. This makes it easier for designers or verification engineers to determine which assertions must be added for full coverage.

7. Formal verification works only on small design blocks.

Once again, this is a myth whose roots lie in experience with early formal tools. Every aspect of formal technology, from the underlying algorithms to ease of use, has been improved and continues to improve. Though running end-to-end assertions on large designs takes effort, it's feasible today. In addition, many apps focusing on specific verification challenges automatically minimize the formal model, so they routinely run on complete chips.



8. Formal verification doesn't work on data paths.

Users traditionally focused formal tools on control logic where bugs lurked due to combinations of corner-case conditions never hit in simulation. But some of these conditions often derive from data-path logic—an arithmetic overflow, for example. Today's leading formal tools handle both control and data equally well. In fact, Xilinx recently presented a paper at DVCon documenting the full formal proof of a 32-bit multiplier, long considered impossible.

9. Formal equivalence checking is limited to application-specific integrated circuits (ASICs).

Equivalence checking is formal verification in which two designs are compared, rather than one design and a set of assertions. Equivalence checking is routinely applied in ASIC development. One common use is to prove that the register-transfer-level (RTL) design and its corresponding synthesis netlist implement exactly the same functionality. Historically, tools have been able to handle only combinational logic, so there had to be a 1-to-1 mapping between state elements in the two designs.

Commercially available formal-verification tools include sequential equivalence checkers that compare two designs even when state elements don't match. A good example is when timing has been optimized by moving logic across register stages. This is crucial to enable the application of formal equivalence checking to field-programmable gate-array (FPGA) design flows, where many optimizations change the state between RTL and netlist.

10. There's no way to integrate coverage results from simulation and formal.


One common past complaint from chip project managers was that formal teams “worked off in the corner” and it was hard to assess their contribution to the verification effort. This is no longer a major issue due to the Unified Coverage Interoperability Standard (UCIS) and various EDA vendor partnerships. These provide ways to have formal verification and simulation work on some of the same coverage targets and for their results to be integrated.

11. There's no way to get a unified view of verification progress.

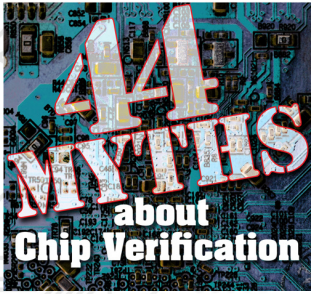
Beyond integrating coverage metrics, project managers need to have a single view of verification progress across all techniques. Today, verification engineers can annotate the verification plan to indicate which goals will be addressed by simulation, emulation, and formal tool. Results, including uniquely formal metrics such as bounded and complete proofs, can be reported back against the plan.

This final myth can't be entirely dismissed. The complex web of EDA vendor partnerships and the lack of an accepted industry standard for verification plans means that not all combinations of tools provide seamless interoperability. However, this should not dissuade potential users from adopting and embracing formal verification, secure that none of the other myths will impede their progress.

TOM ANDERSON is technical marketing consultant at [OneSpin Solutions](#).

to view this article online,  [click here](#)

 [BACK TO TABLE OF CONTENTS](#)



CHAPTER 3:

11 Myths About Hardware-Assisted Verification

LAURO RIZZATTI, Verification Consultant

JEAN-MARIE BRUNET, Senior Director of Product Management and Engineering

Verification expert Dr. Lauro Rizzatti debunks the myths surrounding the two tool classes of HAV platforms—hardware emulators and FPGA prototypes.

Two classes of tools—hardware emulators and field-programmable gate-array (FPGA) prototypes—fall into the HAV category.

Emulators verify hardware and integrate hardware and software of any size and type of system-on-chip (SoC) designs. They also can head-start validation of software and final system validation the entire SoC.

By comparison, FPGA prototypes, running at an order of magnitude faster than emulators on the same design size, are ideal for software and final system validation ahead of silicon. They come in two basic configurations. One is the desktop board that serves single users with an upper limit in design capacity of about 100 million gates. Enterprise platforms supporting multiple concurrent users reaching a maximum capacity of over 10 billion gates is the other.

Below are some of the myths that have surfaced regarding HAV platforms, and explanations that help debunk them:

1. HAV replaces simulation.

Not so fast! While it's true that HAV is mandatory for software and full system validation, hardware-design-language (HDL) simulation is still required to perform thorough hardware verification at intellectual (IP), block, and subsystem levels. As long as the size of the design under test (DUT) doesn't slow the simulator to a crawl at roughly 100 million gates, the interactivity, fast turnaround time (TTA), and ease of use of the simulator should be the preferred approach for hardware debug.

2. Only experienced verification engineers or teams of engineers can adopt HAV because it's too complicated a methodology.

This was the case in earlier versions of the technology. Installing, operating, and maintaining a platform required a large number of specialized engineers. Over time, progress and innovations smoothed the path to adoption. New architectures, new capabilities, and simplified usage eased and facilitated their deployment in all segments of the semiconductor industry to shorten the verification cycle and increase design quality.

Adopting an HAV methodology today isn't an option. It's a requirement.

3. HAV may be useful for emerging applications, such as computing and storage, AI/ML, 5G, networking, and automotive, but not for the functional verification of current applications.

Emerging applications set trends in design capacity, low-power consumption, and high performance, characteristics that mandate the use of HAV. On the other hand, current applications share the same time-to-tape out (TTT) pressure as emerging applications.

Aggressive competition and shrinking profits force electronics providers to bring to market new devices regardless of their sizes. For example, the universe of IoT gadgets that's ahead of their rivals. While HDL simulation can easily and effectively verify their hardware, any design that includes software may benefit from deploying an HAV platform to accelerate TTT.

HAV systems are highly scalable. Smaller configurations that accommodate IoT designs may fit within the verification budget and save the day.

4. The shift-left verification methodology works for software testing, not hardware verification.

The shift-left verification methodology accelerates the entire SoC design verification/validation process, not just software validation. The methodology calls for adopting a modern HAV system suite, including emulation and prototyping deployed in virtual mode.

A comprehensive test environment supported by virtual peripherals, such as Virtual-AB, which is incorporated in the Veloce HAV system from Siemens EDA, can mimic the functional behavior of the physical target system where the SoC ultimately will be plugged in. The environment is conducive to execute real-world software workloads and industry benchmarks before silicon availability. It helps shrink the TTT and enables more testing, increasing the quality of the design.

5. It's close to impossible to integrate emulation and FPGA prototyping.

This was true until recently. Advances in compilation technologies, ease of use, and expansion of use models enhanced the deployment of emulation and prototyping, removing integration bottlenecks. Sharing the front-end compilation flow between the two platforms ensures consistency of the DUT database running in either platform. Loading and offloading the DUT between the two platforms allows for DUT debugging using the emulator and for rapid software workload execution using the prototype.

6. Power analysis is an unfeasible verification task for an HAV platform.

That's far from the truth. In fact, it's just the opposite.

Best-in-class HAV platforms enable early evaluation of power consumption by generat-



ing activity plots and hotspot maps of the DUT described at a high level of abstraction. A quick review of plots and maps reveals the DUT areas of excessive power consumption and the time windows when peak power usage happens during workload processing. Focusing on those areas and time windows, the same HAV platform then can generate detailed DUT activity data at the register transfer level (RTL) to feed a power estimation tool. Some HAV platforms bypass Fast Signal Database (FSDB) and Switching Activity Interchange Format (SAIF) file generation and directly feed the power tool to speed up the process and reduce storage requirements.

While analysis at high levels of abstraction may produce discrepancies of 20% versus the actual silicon, the differences drop to 5% at RTL.

7. FPGA prototyping can replace hardware emulators.

This is incorrect. While some emulators use FPGA devices in their architectures, the differences between the two tools are numerous.

For instance, FPGA prototypes are designed to achieve the highest speed of execution conceivable by trading off fast DUT mapping and compilation efforts, DUT debugging capabilities, and deployment versatility. Emulators, regardless of their architectures—custom processor-based, custom emulator-on-chip-based, commercial FPGA-based—share several characteristics that set them apart from FPGA prototypes. These include:

- The time spent in DUT mapping and compiling with a modern emulator ranges between one day and several days versus several weeks with an FPGA prototyping system.
- Emulators support 100% visibility into the design without requiring probe compilation. While all emulators support this capability, commercial FPGA-based emulators can do so at lower execution speed. Their differences pale when compared to debug with an FPGA prototyping system.
- Emulators can be used in several modes of operation and support a spectrum of verification objectives, from hardware verification and hardware/software integration to firmware/operating-system testing all the way to system validation. They can be used for multi-power-domain design verification and generate switching activity for power estimation. Prototypes are focused on software validation and full system testing.

Emulators and FPGA prototypes have specific strengths that, when combined, can benefit the verification team as in the shift-left verification methodology.

8. HAV can only be used with physical traffic, not software testbenches.

Not anymore. Emulators and FPGA prototypes were originally conceived to test a pre-silicon design with physical traffic. Called in-circuit-emulation (ICE) mode, it was an attractive proposition that promised more thorough testing of the DUT than a software testbench could achieve. The practice came with strings attached. The entire setup was laborious and subject to several hardware dependencies. Among others, it required a speed adapter to slow down the real traffic to the slower speed of the HAV platform. Further, it didn't allow for corner-case testing.

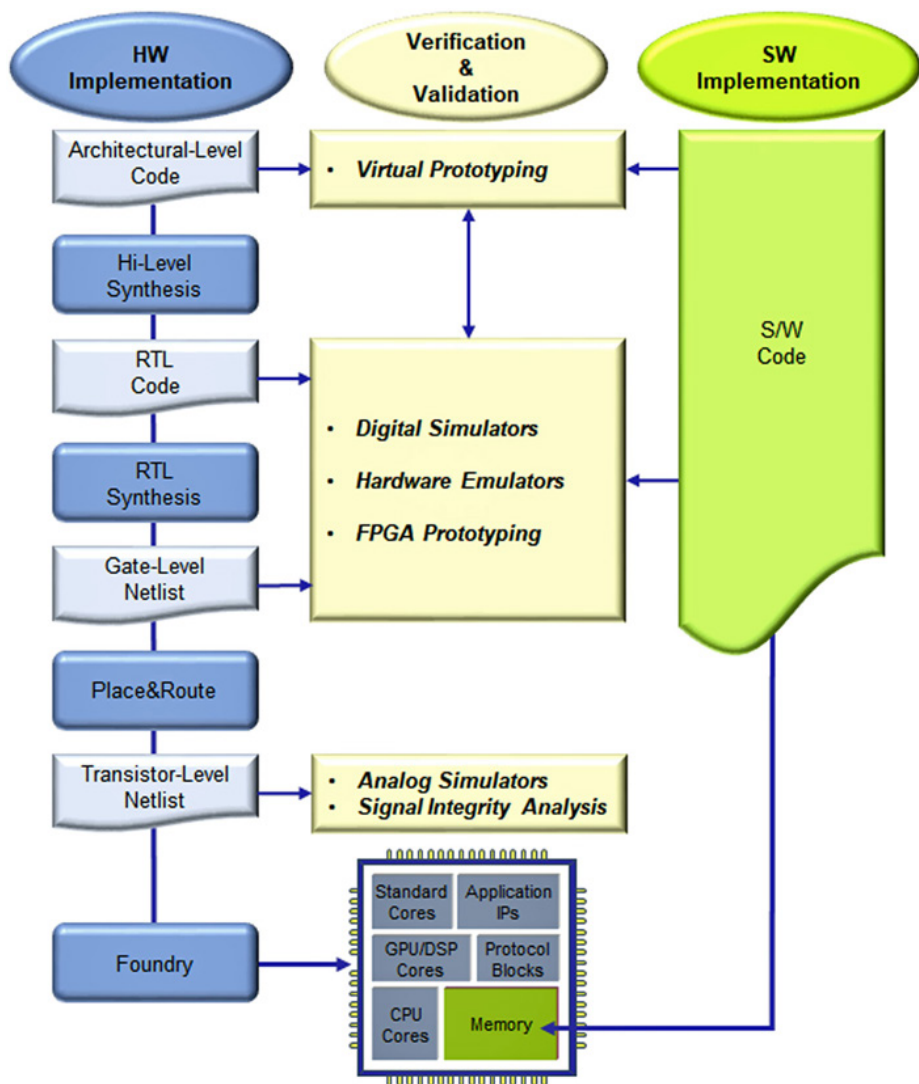
Transaction-based communication between the software testbench and the DUT in the HAV opened a world of possibilities, including new verification tasks such as power analysis. Today, ICE is still used for its inherent advantage, though it's not the main deployment mode.

9. Using HAV requires on-site attendance from the verification group, a nonstarter during a pandemic and as more engineers opt to work remotely.

The opposite is true with a caveat. The platform should be deployed in virtual mode to avoid hardware dependencies such as mounting speed adapters on I/O channels.

To recap, an HAV platform can be used in two modes: ICE mode and virtual mode. In ICE mode, the DUT mapped inside the emulator is driven by the physical target system, where the taped-out chip eventually resides. In virtual mode, the physical target system is replaced by a virtual, software-based equivalent target system that communicates to the DUT inside the HAV via a set of protocol-dependent transactors.

HAV platforms include emulation and prototyping deployed in physical (ICE) and virtual mode. In virtual mode, a comprehensive test environment supported by virtual peripherals mimic the functional behavior of the physical target system where the SoC will be plugged in.



HAV platforms include emulation and prototyping deployed in physical (ICE) and virtual mode. In virtual mode, a comprehensive test environment supported by virtual peripherals mimic the functional behavior of the physical target system where the SoC will be plugged in.

Among its benefits, the virtual mode allows for deployment of the platform in data centers without requiring manual assistance. Other benefits include the ability to perform corner-case analysis, what-if analysis, and more, not possible in ICE mode.

10. An HAV platform is an expensive line item for all but the most complex chip design projects and well-funded startups.

Relative to the acquisition cost of an HDL simulator, the statement is certainly correct, though misleading. The acquisition cost of a modern HAV platform pales when considered in relation to the verification power and flexibility of the tool. The HAV platform has the performance and capacity necessary to tackle even the most complicated debugging scenarios, which often include embedded software content.

As strange as it may sound, the tool's versatility makes hardware emulation the cheapest verification solution when measured on a per-cycle basis.

The total cost of ownership also has dropped significantly. Gone are the days when, figuratively, the emulator was delivered with a team of application engineers in the box to operate and maintain it. Multi-user support shares the acquisition cost among team members. The



improved reliability of the product further reduces the cost of maintenance by orders of magnitude.

11. In the future, all HAV platforms will be based on commercial FPGAs.

While this has been and will continue to be true for prototyping platforms, it's incorrect for emulators.

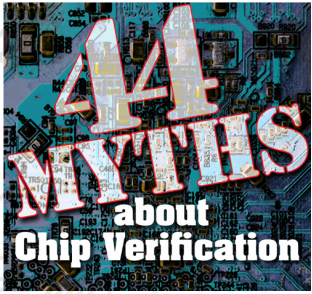
The three emulator providers adopt three different architectures: proprietary emulator-on-chips, custom processors, and commercial FPGAs. They claim to believe in the benefits of their architectures and will continue development to enhance them. For the proprietary/custom approaches, the path to the future lies in re-spinning their chips to lower technology nodes. For the FPGA-based approach, the future is in the hands of FPGA suppliers. This begs the question: Will the two main FPGA providers that recently changed ownership (Altera to Intel and Xilinx to AMD) continue to develop devices with properties required by prototyping applications? The jury is out.

LAURO RIZZATTI is a verification consultant. He was formerly general manager of EVE-USA and its vice president of marketing before Synopsys' acquisition of EVE. Previously, he held positions in management, product marketing, technical marketing, and engineering.

JEAN-MARIE BRUNET is the senior director of product management and engineering for the Scalable Verification Solutions Division at Siemens EDA. He has served for over 20 years in application engineering, marketing, and management roles in the EDA industry, and has held IC design and design management positions at STMicroelectronics, Cadence, and Micron among others. Jean-Marie holds a Master's degree in Electrical Engineering from I.S.E.N Electronic Engineering School in Lille, France.

to view this article online,  [click here](#)

 [BACK TO TABLE OF CONTENTS](#)



CHAPTER 4:

11 Myths About Chip Specifications

ANUPAM BAKSHI, Founder & CEO, Agnisys

Creation of a specification for a semiconductor can be time-consuming and costly, especially if the project marches on with continued refinements. Automating the process is the best solution, but various myths have given designers pause.

Every successful engineering endeavor starts with a solid specification, and chip development is no exception. A semiconductor project generally begins with the marketing requirements for a new product, either the chip itself or a system incorporating the chip. System architects develop a high-level specification for the chip, typically including both aspects of the hardware design and the interfaces used by software to control and communicate with the hardware.

The specification is refined as the project continues, and many of these refinements result in additions or changes to the specification. Choosing the target silicon technology may have ripple effects back to the hardware specified—for example, if the original memory size can't be supported. In addition, features may be added in response to competitive products introduced during the development timeline.

Further specification refinements happen while the hardware and programming teams are creating the design and its associated software. Sometimes the vision of the architects proves impractical or too expensive to implement. Adding clocks, resets, power-management features, and testability support to the design may also require unanticipated specification updates. Engineers often say that change is the only certainty when it comes to chip specifications.

Development teams always want to save resources to lower project costs and shrink the schedule to reduce time to market. This is possible only through higher levels of abstraction and increased automation of the design and verification process. Finding a better way to create and maintain chip specifications is part of such a solution. Taking this next step in chip development requires recognizing and dispelling some common myths.



1. Chip specifications must be written in natural language.

It's true that most specifications are written in natural language, probably most commonly in English. It's also true that no other existing language can specify 100% of a chip's requirements from both the hardware and software viewpoints.

However, many aspects of a chip can be described using more formalized domain-specific languages (DSLs) and formats. These include registers and memories, programming sequences, verification sequences, silicon test sequences, power-management structures, and hierarchical chip assembly. These portions of the specification are inherently unambiguous and more precise than natural language.

2. Chip specifications aren't executable.

Development teams have long wanted to be able to feed their specifications into some sort of magical electronic-design-automation (EDA) tool that would generate the register-transfer-level (RTL) design as well as the testbench and tests needed to verify it. While some might argue that the RTL descriptions themselves qualify as executable specifications, in practice this implementation is too detailed, and a higher level of abstraction is required.

Though no full-chip solution is available, the DSLs and formats used for specific parts of the specification can be executable given the right tools. It's possible to generate the RTL design, testbench components, verification tests, programming headers and sequences, files for automated test equipment (ATE), and end-user documentation for these portions of the chip.

3. Natural language isn't executable.

It's true that a complete chip specification isn't executable, but there's excellent progress being made in understanding natural language descriptions for certain aspects of the chip. Artificial intelligence (AI) and machine learning (ML) can transform these descriptions into executable form. Two particularly active areas involve generating SystemVerilog Assertions (SVA) from statements of design intent and converting natural language sequence descriptions into SystemVerilog verification tests and C/C++ code for drivers and embedded software.

4. Chip design and verification specifications are separate.

In the past, this has been the case. The chip-verification team took the design specification and developed a verification plan that iterated all of the design features that had to be verified. With the advent of constrained-random verification, the focus shifted from manual tests to achieving coverage goals.

Verification planning tools can tightly link a list of coverage targets with the design specification, ensuring that all design features are covered in the verification plan. This approach is automated even further when testbench components, including coverage checkers, and verification tests are generated from the executable portions of the chip specification.

5. Chip hardware and software specifications are separate.

This has also been the case in the past, and it remains true to some extent. Some aspects of the software specification are independent of the underlying hardware. However, a significant portion of the software interacts directly with the hardware, usually through reading and writing registers.

As noted above, C/C++ driver and embedded code can be generated automatically from executable specifications. Thus, part of the software specification is shared with the hardware specification.



8. IP specifications can't be easily integrated at the chip level.

With natural language specifications, details for IP and other lower-level blocks can simply be merged into the text for the full chip. It turns out that easy integration is also possible with executable portions of specifications.

Since the generation process works at multiple levels, engineers can move up and down the specification hierarchy with predictable and consistent results. Executable specifications are in text-based formats, so they can be merged into a chip-level specification in the same way as natural language.

9. IP block integration must be done manually.

Especially in a large system-on-chip (SoC) design, the task of integrating and interconnecting IP blocks can be huge. Tens of thousands of blocks are not uncommon. However, with executable interconnect specification, it's no longer necessary to do this manually. IP libraries can include the information needed for a chip assembly tool such as Agnisys SoC Enterprise to create the higher levels of hierarchy, all the way up to the full chip, and automatically generate all interconnecting logic. Users can provide guidance using standard formats such as Tcl and Python.

10. Custom block integration can't be automated.

Automatic integration and interconnection work for custom blocks as well as standards-based IP. Users can specify information for their blocks using the same Tcl/Python approach, including features such as pattern matching to recognize ports and signals that should be connected.


11. Chip specification is a one-time investment.

As described above, chip specifications evolve over the course of the project, changing dozens or even hundreds of times. Without automation, every single time that the specification changes, the designers must update the RTL implementation; the verification engineers must revise the testbench, tests, and PSS models; the embedded programmers must modify their code; and the technical writers must edit the documentation. Executable specifications enable regeneration of all these output files at the simple push of a button.

Thus, automation saves time and resources in the initial creation of the specification and adds even more value on every subsequent revision. Every member of the development team directly benefits. For SoCs and all chips of significant size or complexity, specification-based automation is a must-have technology.

ANUPAM BAKSHI is the founder and CEO of Agnisys. He has more than two decades of experience implementing a wide range of products and services in the high-tech industry. Prior to forming Agnisys, he held various management and technical lead roles at companies such as Avid Technology Inc., PictureTel Corp., Blackstone Consulting Group, Cadence Design Systems, and Gateway Design Automation.

Anupam earned a HighTech MBA from Northeastern University, Massachusetts, a Master's in Computer Engineering also from Northeastern University, and a Masters in Science (Electronics) from Delhi University.

to view this article online,  [click here](#)

 [BACK TO TABLE OF CONTENTS](#)