

Special Technology Report

[Print](#) [Email](#) [Bookmark](#)

Mixing Formal and Dynamic Verification, Part 1

By Bill Murray

04/30/09

Over the last few years, there has been a noticeable uptick in the use of formal verification to augment dynamic verification. Given that both techniques leverage assertions [1, 2], one would assume that there would be a great deal of collaboration between dynamic testbenches and formal property checking, the user teams and the tools. Indeed, a DVCon 2009 panel discussed mixing the two – thus the title of this special technology report (STR).

In this STR, we dig down to the use case level to determine how formal is being used, and how it augments dynamic verification. We used 17 use cases ranging from early RTL analysis to functional sign-off to survey 19 engineers and engineering managers at 16 industry-leading IP, chip and systems design companies to understand their formal adoption and use in 2006 and 2009, and projected use in 2012. We also interviewed respondents from 9 of those companies to gain even more detailed insight. Of course, the user responses apply only to their individual groups, not necessarily throughout their multi-divisional companies.

We found that there is certainly collaboration between simulation and formal verification, but “mixing” might be a bit of an overstatement at this stage. The methodologies, standard common coverage metrics, and tool interoperability required by true mixing are in a distinctly embryonic stage of development.

The good news is that formal verification has developed to a level of usefulness and maturity at which design and verification teams *want* to mix it with tried and more-or-less trusted dynamic verification. What has changed in formal verification to make this possible?

In part 1 of this STR, we:

- Discuss the upper-level results of our use case survey.
- Learn from the respondents why formal is enjoying increased adoption.
- Review formal’s sweet ‘n’ sour spots and how the sweet spots are expanding.

In [part 2](#) of the STR, we discuss:

- The detailed results for the 17 use cases.
- How formal is currently being used with dynamic verification.
- The application of formal in the ESL space.
- How technology users and providers envisage formal methods in 2012.

Now let’s review what users at Alcatel-Lucent, Analog Devices, ARM, Cisco, DE Shaw Research (DESRES), Fujitsu Microelectronics Europe, HP, IBM, Infineon, Intel, nVidia, Qualcomm, Saab, Silicon Logic Engineering (Tundra), STMicroelectronics and Sun Microsystems said to SCDsource. This is their story.

SCDsource User Survey

User base broadened

Of the 19 respondents, 14 were employing formal verification in 2006 – this STR refers to them as established users. The five respondents who adopted formal after 2006 (dubbed “recent adopters”) constitute a 36 percent increase in user breadth from 2006 to 2009.

Use employment deepened

Our survey shows a significant deepening of proliferation within user groups. Figure 1 shows the average number of use cases employed in 2006 and 2009, and projected employment in 2012. Average use cases grew 46 percent from 6.6 in 2006 to 9.7 in 2009. Our respondents project this to grow by 23 percent to 12.8 use cases in 2012.

project this to grow by 32 percent to 12.8 use cases in 2012.

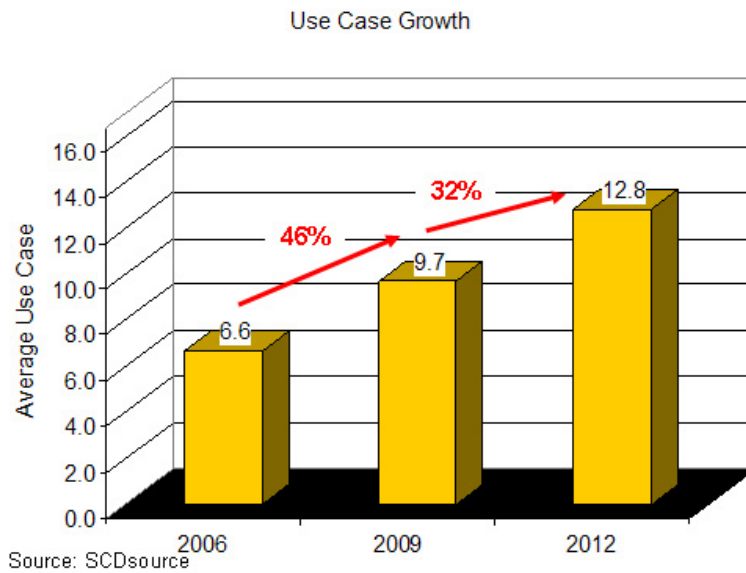


Figure 1: The deepening use of formal verification (Source: SCDsource)

Total use nearly doubled

So, what is the total formal deployment across the sample of 19 user groups? The user breadth and use depth are combined in figure 2. In this population, the use of formal in 2009 is nearly double that in 2006, and 2012 use is projected to increase another 32 percent over 2009. Projected 2012 use is about 2.6X that of 2006. This equates to a compound annual growth rate (CAGR) from 2006 to 2012 of over 17 percent.

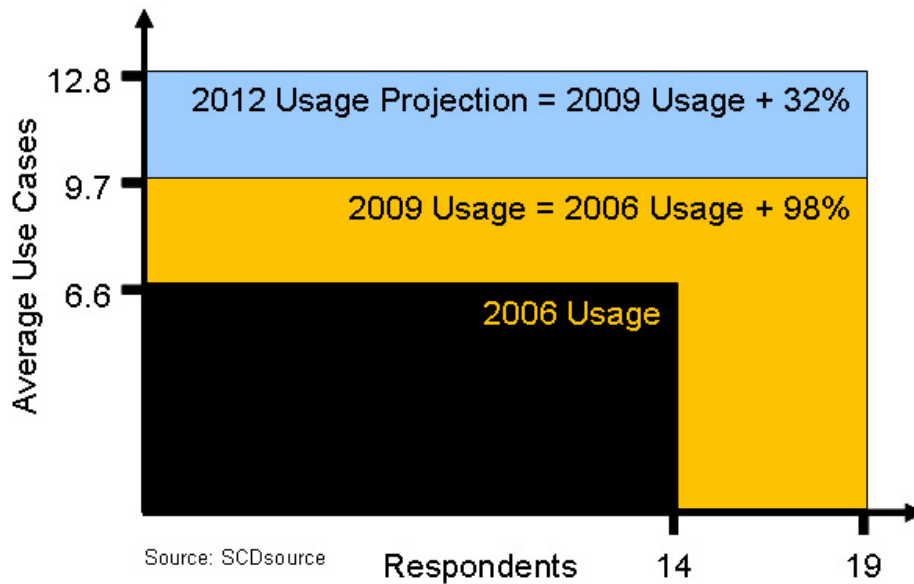


Figure 2: Formal verification use nearly doubled from 2006 to 2009 (Source: SCDsource)

In summary, one could argue that the growth rate in this sample is quite impressive for a verification approach that has a reputation for being difficult to adopt and use.

Recent adopters ramp fast

How difficult to adopt and use? Our survey shows that recent adopters are setting an aggressive pace. Could this mean that formal is becoming easier?

The 5 recent adopters grew their average use case employment from zero in 2006 to 7.8 in 2009, and project that this will grow by roughly 39 percent to 10.8 in 2012 (see figure 3).

The average use case employment by the 14 established users in 2006 was 6.6, and grew about 56 percent to 10.4 by 2009. This group projects that it will expand employment to 13.5 by 2012, a growth of about 30 percent.

Recent Adopters Ramp Fast

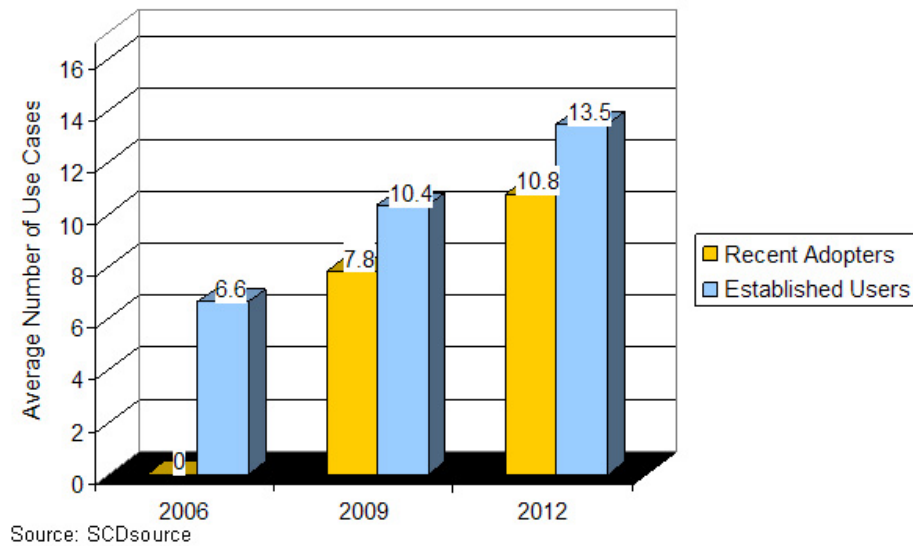


Figure 3: Recent adopters are catching up – fast (Source: SCDsource)

The recent adopters are employing more use cases in 2009 (7.8) than the established users were employing in 2006 (6.6). If use deepens according to projections, the recent adopters' use case employment in 2012 (10.8) will be slightly higher than that of the established users in 2009 (10.4). Recent adopters in this population will thus lag established users by about three years – but bear in mind that many established users adopted formal quite a long time ago. Clearly, recent adopters are catching up with established users – fast.

Advancing backwards in the flow

Formal is renowned for its ability to perform exhaustive verification – and for the decades-long perception that the user must have a Ph.D. in maths to undertake it. However, formal's proponents claim that modern tools and methodologies make it accessible to mainstream verification engineers. Indeed, they claim that formal is expanding into the design domain. Our survey shows that they are right on both counts.

The use case expansion – shown in much more detail in part 2 of this STR – clearly shows formal's advance into mainstream verification, but what about design? One example of formal's expansion into the design domain is that of early RTL analysis. It is already among the most widely-employed use cases in 2009, used by 68 percent of our respondents. Moreover, our survey projects use to increase by the year 2012 to 95 percent of respondents (see figure 4).

Early RTL Code Analysis

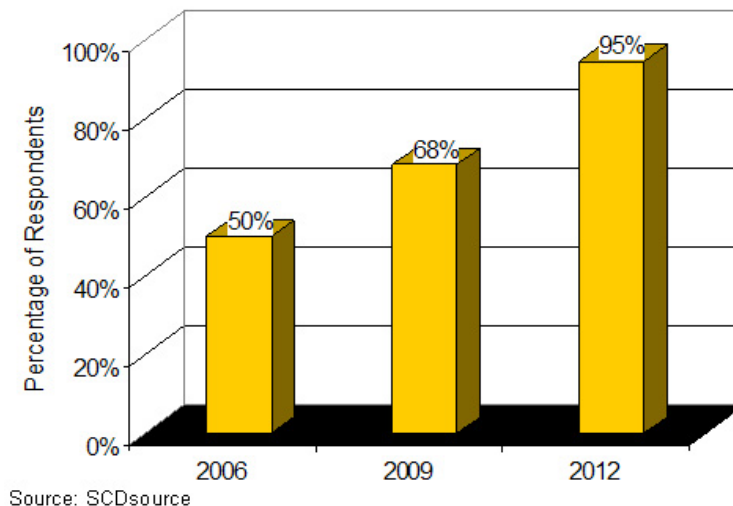


Figure 4: Formal has advanced backward into design (Source: SCDSource)

Formal has expanded its scope, but that doesn't mean that it has lost its focus. Its major traditional perceived strength – exhaustive verification – is being performed by 68 percent of respondents in 2009, expected to increase to 84 percent by 2012 (see our [cartoon](#)).

Exhaustive verification of block features, functions, operations, and transactions remains the leading use case in terms of importance, with about 53 percent of respondents ranking it in their top 3 use cases.

their top 3 use cases.

Clearly, formal is fulfilling its original exhaustive verification mission, but equally clearly, it is has advanced backwards in the design flow – and into mainstream design and verification.

Tools revenue numbers

According to Gary Smith at [Gary Smith FDA](#), the CAGR for formal tools revenue over the period 2007 to 2012 is about three times that of simulation tools. Specifically, formal verification and formal analysis revenues are growing at a CAGR of 4.2 percent and 4.3 percent respectively, while RTL simulation is growing at only 1.5 percent.

The absolute revenue growth results for formal appear to conflict with a common perception – and with our survey results – that it is enjoying noticeably greater proliferation than heretofore. According to Smith, these modest revenue growth numbers are the result of competitive price pressure in RTL design and verification tools in general, especially from Cadence, Mentor and Synopsys.

Formal adoption - why now?

For decades, proponents of formal property checking have claimed that it can achieve higher verification quality because it can reach parts of the design that simulation cannot reach – at least, not efficiently in a bounded project time. They claimed not only that formal can verify that a design behaves according to the specification, but also that its exhaustive nature enables it to detect unintended behavior far more effectively than can dynamic verification. Early in the life of formal property checking, some claimed that it would one day replace simulation. It hasn't. In fact, only in the last few years have we seen it proliferate beyond a relatively small community of formal verification experts into the mainstream. Why? What is happening to fuel this long-predicted and long-awaited move? We start with the crumbling barriers to adoption.

Crumbling adoption barriers

Historically, formal verification has been perceived to be capacity-limited, difficult to use, and lacking robust, systematic methodologies. These barriers have been gradually crumbling.

Capacity

According to Jason Baumgartner, formal technology lead in IBM's Systems and Technology Group, "For decades, formal algorithms bordered on the 'unusably unscalable.' The user had to target a piece of the design small enough to avoid choking the formal verification tool, requiring copious expertise and effort."

IBM has significantly increased scalability by employing advanced techniques such as algorithms that automatically reduce the size of the problem to be analyzed, and which alternate effort between 'proving' and 'disproving.' Baumgartner said "We can now tackle the verification of large blocks such as an entire floating point unit (FPU). This increased capacity eliminates, for example, the need to whittle out from the FPU some small state machine that handles some complex bypassing, the standalone verification of which would otherwise require orders of magnitude greater manual effort."

Most respondents agree that capacity has improved, but many take the same view as Alcatel-Lucent's Joachim Knäblein. He is in an ASIC design support team that works on verification methodology and internal tools, and wants the capacity of formal tools to increase further.

Ease of use

Saab Avionics started investigating formal methods and commercial tools as far back as 1996, according to Håkan Forsberg, technical specialist. "As recently as 2004 to 2006, dynamic (simulation) approaches were still better at verifying our designs – which are largely control-centric – despite improvements in the formal tools. We had to use a lot of constraints to make the formal tools work," he said. "Now, it's easier to write assertions and constraints, but it's not necessarily more automated."

The historical lack of industry-wide standard assertion languages created a barrier to formal adoption. Tim Stremcha, a senior design engineer at Silicon Logic Engineering (SLE), observed that "Formal verification became more compelling with the advent of standard assertion languages for both formal and dynamic verification. A standard language enables us to import formal assertions into the dynamic testbench environment." Stremcha finds this especially helpful while the testbench is still under construction. He said "Mixing formal assertions with the testbench allows us to co-develop the two verification approaches. In effect, importing a new assertion provides a new directed test when running formal, and the input constraints provide a very nice cross-check of the testbench stimulus."

According to Volkan Esen, an ABV specialist in Infineon's Enabling Technologies and Services (ETS) group, the popularity of assertion-based simulation has spurred the adoption of formal verification. He said "ABV knocked down the notation barrier to formal adoption. Users have stepped out of their VHDL comfort zone to write assertions in PSL and SVA, and discovered that it's as intuitive as a waveform spec. These languages eliminate the hardcore maths."

Methodology

Methodology is key. Summing up the view of many respondents, Bryan Dickman, Director, Design Assurance at ARM, said "the performance and capacity of formal tools, though important, are less important than having the appropriate methodologies to verify a particular class of problem. And you still need engineers experienced in formally verifying that class of problem. We spend a lot of time developing methodology with our tool suppliers and are seeing good progress toward some consistent use models."

Of course, improved capacity, ease of use, and methodologies reduce barriers to adoption, and all undergo a "continuous improvement" process. But what are the incentives for using formal verification at all?

Quest for quality

According to our interviewees, it's primarily about verification quality. Productivity is a key factor in the return on investment (ROI) calculation that often determines whether formal is deployed in any given situation. So, higher productivity is a hot button requirement in formal deployment, if not always a driving factor.

Safety first

According to Saab's Forsberg, "We must comply with an avionics industry verification requirement specified in the Radio Technical Commission for Aeronautics ([RTCA](#)) document [DO-254](#) – Design Assurance Guidance for Airborne Electronic Hardware. These guidelines [3] mandate the use of additional design assurance methods to verify complex safety-critical designs at the two highest design assurance levels. Proposed advanced verification methods include safety-specific analysis, elemental analysis, and formal methods."

According to consultant Kristoffer Karlsson, who has worked with both Saab and the European Aeronautic Defence and Space (EADS) company, formal methods afford the design team a more effective means to:

- Analyze design intent and its implications for functional verification.
- Validate functional requirements within their operational context, and
- Validate the correctness and completeness of the assertions, whether used in formal or dynamic assertion-based verification (ABV).

Bug hunting first, but sign-off is the ultimate goal

We asked a recent adopter why his group adopted formal methods. Hewlett-Packard's David Lacey is the verification manager for chipset designs targeted at the company's high end servers. Initially, Lacey is tasking formal with bug hunting, but he envisages it as a sign-off criterion over the longer haul.

"Our goal is to get high-quality silicon out in as few releases as possible," said Lacey. "There are certain classes of bugs that are very difficult, if not impossible, to find with simulation, and so that's where we'd like to derive value from this technology. Just in general, it's another way to uncover bugs that we might otherwise miss."

Exhaustively verify – no anticipation necessary

Alcatel-Lucent's Knäblein tabulated a comparison between simulation and formal verification (see figure 5).

Criterion	Simulation	Formal
Verification Depth	Non-exhaustive checking. Limited even when using constrained-random testing	Exhaustive checking. Detects unanticipated issues and "tricky" error combinations
Verification Preparation	Requires complex testbench	No dedicated testbench needed
Verification Complexity	Suitable for every block size, even top level	Limited computation power and property complexity
	Can be used for arbitrary verification cycle counts	
Skills	System knowledge needed	System, block and formal tool knowledge needed

Source: Alcatel-Lucent

Figure 5: Comparative benefits and costs of simulation and formal methods (Source: Alcatel-Lucent)

He cited verification quality as the main driver for formal adoption. "We ran simulation and formal benchmarks on an ASIC design. We found about the same number of bugs in each, but not the same bugs. Simulation found the more obvious bugs – those which occur at longer system run times. Formal found the more exotic bugs, bugs which seldom occur, but which can be quite serious."

Knäblein pointed out that "Simulation can make sure that something works the way it's supposed to work, but it cannot necessarily ensure that something does not work the way it's not supposed to work. For example, formal found a register map bug whereby some registers were mirrored in the address space. Accessing this address evoked an unexpected response from the register map. Normally, our simulation just checks whether the registers are accessible; it doesn't check whether invalid addresses do not respond to an access."

Infineon has used formal verification for nearly a decade, according to Darren Galpin, a verification team lead. Galpin's group verifies silicon IP such as bus bridges and data mover engines [4] for the company's TriCore single-core 32-bit MCU-DSP architecture. "We use formal to verify interface blocks that translate from one communication protocol to another. It's not necessarily terribly complex, but it is critical that we get it right," he said. "With an exhaustive formal proof at the operation/transaction level, we can be sure that we've completely covered a block's functionality. Using random and/or directed tests would require us to generate rather large coverage sets – and we still couldn't be sure that we've covered everything." Galpin observed that formal is particularly good at detecting bugs in legacy IP.

Exhaustive formal verification is especially useful in SoC design, where a great deal of functionality is implemented in software, according to Wolfgang Ecker, principal engineer in Infineon's ETS group. He observed that "Formal eliminates the need to generate individual, targeted test cases and checkers. For example, when we verify the correct integration of a module or IP into the SoC, we must verify its correct connection and communication with the system bus. Using a non-formal approach, we would first have to generate the transaction or bus access on the CPU side – essentially a software instruction. We would then have to run several checks to ensure that all relevant registers are modified, and that all other registers remain undisturbed. In contrast, exhaustive formal verification executes a complete search of the entire function space, driven by specifications generated from [IP-XACT](#) with [SPRINT](#) extensions. We don't need to worry about software accesses and data bursts."

Advancing backwards in the design flow

Olivier Haller, verification methodology manager at STMicroelectronics, said "We've been using formal verification for about ten years. We adopted a mixed formal and dynamic ABV methodology on blocks and IP about a year ago." Haller is responsible for the definition, deployment and promotion of innovative verification methods across all ST divisions. He states that ST's block and IP designers use assertions to:

- Perform some initial verification.
- Capture design assumptions which assist the integration of the block into chip-level verification.
- Ease and speed debug because assertions are often closer to the root cause of failure than dynamic black-box verification data.
- Communicate desired verification coverage points to the verification engineer.

Richard Ho, researcher at DE Shaw Research (DESRES), said "We use formal methods to complement simulation and have achieved improved quality of results (QoR). For instance, when a coverage point is missed in simulation, we use formal to determine whether or not the point is reachable. If it is, a simulation test case is constructed to find it, based on the formal verification counterexample."

Ho continued "Also, early RTL code analysis using very simple, automatically-generated assertions is invaluable in achieving coverage closure. Formal helps us to rapidly identify sections of code that are complex and may cause coverage closure challenges for simulation. There is a close correlation between these complex sections and the verification challenges that we encounter further down the flow, so their early detection allows us to modify RTL code at the point in the flow when it is easiest to modify." [5]

Silicon Logic Engineering's Stremcha works at the designer-verifier interface. "We use formal at the design stage to improve the quality of design – independent sign-off verification by the verification team remains mandatory." The company develops silicon intellectual property in support of its ASIC and FPGA design services. According to Stremcha, "Configurable IP, such as our chip-to-chip interface protocol core that supports a range of SERDES lane counts and speeds, is a very good candidate for formal property checking. Formal enables the designer to verify a set of configurations supported by the IP to confirm correct operation. The verification team then verifies the extended range of configuration options."

In addition to IP quality, Stremcha points out another value – assertions mitigate issues in customer design support. He said "Using a fully-described definition of the I/O specification enables us to quickly identify customer usage issues, such as illegal traffic scenarios. With formal assertions, the customer receives immediate feedback from the constraints if they violate the I/O specification."

Grappling with multiprocessing

ARM's Dickman pointed to the increasing complexity in key growth areas such as multiprocessing as an example driver for the company's formal adoption. "Coherent systems based upon MESI-like [[Modified, Exclusive, Shared, Invalid](#)] protocols may be a good target for formal verification," he said.

ARM has used formal verification as a late-stage back-up in some projects for a number of years, but has ramped its efforts in formal over the last six months in the hope of reaping a greater ROI. "Using assertions to capture design intent in the design phase and then using them later in simulation is an accepted best practice in ARM. For a number of years, some project teams have applied varying levels of effort later in the flow to exhaustively verify those assertions formally, with varying results. However, we're now exploring more in higher-level assertions because, for example, liveness and safety properties have a potentially higher ROI. For instance, ensuring that a circuit is deadlock-free has a very high value."

What about productivity?

The feedback on productivity varies a great deal. A perception that users must be truly expert in order to derive productivity advantages from formal is not robustly supported by this feedback. The results look like a situation-dependent mixed bag.

HP's Lacey says that his team has realized a productivity value – an increase in simulation speed by reducing the number of assertions used in dynamic verification. "We encourage our designers to write assertions in order to communicate design intent. However, that slows simulation. To mitigate the problem, we identify which assertions we can prove in an under-constrained environment; then we omit them from simulation because we know they can never fire. Our strategy right now is to first understand where we can derive the most value from formal tools, and then use them to make our simulation cycles more productive."

SLE's Stremcha says that "formal verification requires a great deal of work that should be done in design, anyway. As designers increasingly employ assertions during the design stage, the effort to apply formal decreases."

ST's Haller sees occasional productivity boosts because designers must no longer devise block-level testbenches. Nonetheless, the teams must continue to use dynamic verification because formal verification tools still suffer from capacity shortfalls.

According to Infineon's Galpin, his group uses any productivity gains achieved by formal to apply more effort elsewhere in the verification flow. He said "Formal rarely reduces downstream dynamic verification effort because we tend to apply the same number of dynamic tests to IP as we would have done had we not used formal; and we dynamically verify the assumptions that we made in formal about how the IP will be used. In any case, the confidence established by formal enables the team to devote even more cycles to the core behaviors in the dynamic part of the testbench – particularly in the case of blocks that are likely to be extensively reused."

According to Ho at DESRES, "Formal verification effort is often equal to or greater than simulation effort. So, we have to use it to make simulation simpler, shorter or unnecessary. That's where we win the productivity."

Alant's Lucent's Koehler estimates that productivity is about the same for formal and

Aicatel-Lucent's Knäblein estimates that productivity is about the same for formal and dynamic verification. Nonetheless, block-level productivity gains can be quite substantial. In today's edition of SCDsource [6], Knäblein and his colleague, Hans Sahm, [report how they used](#) both automated assertion generation and automated formal methods to verify a complex HW/SW interface in a large SDH/SONET chip – and slashed verification time and effort by 70 percent versus simulation.

What about project risk?

Simulation has long been the more-or-less predictable verification route. The technology limitations of formal verification and the consequent uncertainty of success have historically fuelled a reluctance to adopt it.

However, according to IBM's Baumgartner, "The ability to share assertions across formal and dynamic ABV has made an impact on formal adoption. Assertion-sharing allows us to set up a formal testbench, and to redirect the assertions to a dynamic environment if the formal tool can't cope. The effort to establish these formal assertions is thus not wasted. Eliminating the project risk associated with potentially wasted effort in formal enables us to push it to the limit." He continued, "As formal technology continues to improve, we can tackle verification challenges that used to be impossible in formal. For example, we can set up a reusable formal methodology to verify error-code correction – we don't need CPU months of dynamic effort, which itself still carries the risk of verification gaps."

And methodology?

The importance of methodology came up in almost every interview. So, how do 16 of our 19 respondents rate the methodology support that they receive from technology providers? The responses are shown in figure 6.

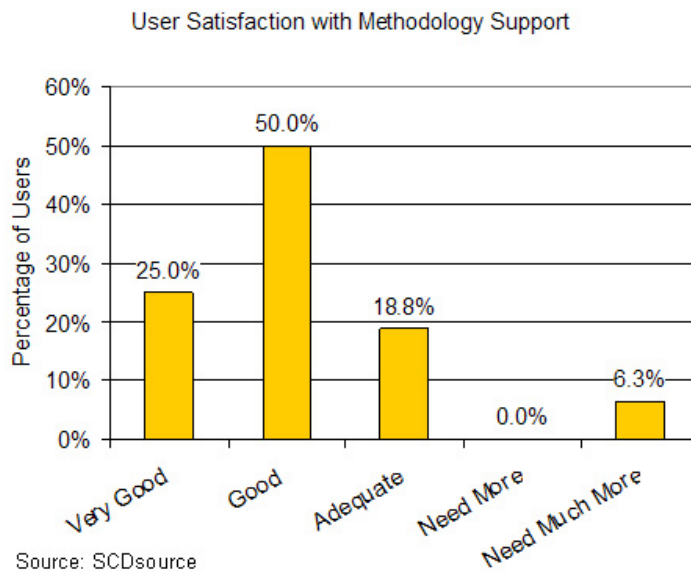


Figure 6: User satisfaction with technology providers' methodology support (Source: SCDsource)

Formal verification - where?

What are the sweet 'n' sour spots of formal verification? Clearly, the answers to these questions go to the heart of mixing formal and dynamic methods. Our user respondents broadly agreed on the sweet spots – control circuits, and datapaths with high concurrency that do not involve data transformations. The sour spot is datapath circuits that *do* involve data transformations.

A DVCon 2006 paper [7] spells it out. Example sweet spots include:

- Arbiters of many different kinds
- On-chip bus bridge
- Power management unit
- DMA controller
- Host bus interface unit
- Scheduler, implementing multiple virtual channels for QoS
- Clock disable unit (for mobile applications)
- Interrupt controller

- Memory controller
- Token generator
- Credit manager block
- Standard interface (for example, PCI Express)
- Proprietary interfaces

According to the paper, example sour spots include:

- Floating point unit
- Graphics shading unit
- Convolution unit in a DSP chip
- MPEG decoder

However, as IBM's Baumgartner previously noted, the company *can* formally verify FPU. In a DATE 2005 paper [8], Baumgartner *et al* report the fully-automated, exhaustive formal verification of fused-multiply-add (FMA) FPUs without the need for customized tools. The approach focuses on the arithmetic correctness of a single arbitrary instruction, and compares the FPU implementation with a simple reference model derived from the processor's architectural specification. The approach uses a combination of case-splitting and automatic model reduction techniques, and isolates the multiplier for dedicated verification. According to the paper, the approach is portable to simulation, emulation, semi-formal, and formal verification.

The foregoing sweet spot list does not include the verification of CPU and application-specific processors or, at least, the parts of those processors that do not perform data transformations. However, formal is being used increasingly to verify such processors.

For example, Infineon used formal to verify its 40-instruction PPv2 network processor [9]. It required only 40 high-level (operation) properties to verify the correct pipelined processing of multiple instructions; the correct operation of permissible, but unpredictable, behaviors such as traps and interrupts; data paths with complex bit-manipulations; and independent execution of multiple threads under all possible combinations of instructions, thread switches, traps and interrupts. An example of a bug detected by formal: an error caused instruction words to be modified while stored in the context switch buffers. The verification team hadn't anticipated this situation. Dynamic ABV might have detected it, but only with a great deal of luck.

DESRES designs ASICs that incorporate custom processors. Ho said "Simply by breaking the verification task down to smaller units, we can very quickly formally verify that an individual processor operation behaves exactly as specified for all possible combinations of inputs. We still have to simulate the whole processor, but the simulation of the formally-verified pieces is essentially "off the table" as far as the verification plan is concerned."

Eliminating the mystique

Some respondents think that formal verification is shrouded in an unwarranted mystique (see our [cartoon](#)).

Ho at DESRES said "Formal is just another tool in the toolkit. We have the objective to mainstream formal in 2009. Our goal is to have every member of the verification team be proficient in the use of both simulation and formal in that timeframe."

Infineon's Galpin said "There's a common perception that you have to be a mathematical genius to use formal. It's not nearly as difficult to use as it's portrayed." He also had a suggestion for conference organizers: "I think that conferences could improve this perception by focusing more on how formal is used in practice, and clearly differentiate this practice from the advanced algorithm development presented by the academics."

We at SCDsource hope that the results of our survey will help to eliminate at least some of that mystique.

In [part 2](#) of the STR, we discuss:

- The detailed results for the 17 use cases.
- How formal is currently being used with dynamic verification.
- The application of formal in the ESL space.
- How technology users and providers envisage formal methods in 2012.


References

1. [The Art of Verification with SystemVerilog Assertions](#). Faisal Haque, Khizar Khan, Jonathan Michelson. [Verification Central](#) 2006.
2. [Practical Approaches to Deployment of SystemVerilog Assertions](#). Faisal Haque, Jon Michelson. EE Times 2007.

3. Structured Assertion Design Verification for Complex Safety-Critical Hardware. Kristoffer Karlsson, Håkan Forsberg. Military and Aerospace Programmable Logic Devices (MAPLD) Conference 2008. [Paper](#) and [Presentation](#).
4. [A comparison of three verification techniques: directed testing, pseudo-random testing and property checking](#). Mike Bartley, Darren Galpin and Tim Blackmore. Proceeding of the 39th Design Automation Conference, 2002.
5. [Early formal verification of conditional coverage points to identify intrinsically hard-to-verify logic](#). Richard Ho, Michael Theobald, Martin M. Deneroff, Ron O. Dror, Joseph Gagliardo, David E. Shaw. Proceedings of the 45th Design Automation Conference, 2008.
6. [Automated formal method verifies highly-configurable HW/SW interface](#). Joachim Knäblein and Hans Sahn. SCDsource 2009.
7. Guidelines for creating a formal verification testplan. Harry Foster, Lawrence Loh, Bahman Rabii, Vigyan Singhal. DVCon 2006. [Paper](#) and [Presentation](#).
8. [Automatic Formal Verification of Fused-Multiply-Add FPUs](#). Christian Jacobi, Kai Weber, Viresh Paruthi, Jason Baumgartner. Proceedings of DATE 2005.
9. [Achieving Certified IP Quality Efficiently](#). Lorenzo di Gregorio, Carlo del Giglio, Michael Siegel. EE Times 2007.
10. Zero Escape Plans: Tying Together Design, Simulation, and Formal Methods for Bulletproof Stepping Validation. Erik Seligman, Carl Dreyer, Ken Haren, Raman Nayyar. Proceedings of DVCon 2008. Reported in [Formal verification expands its use model](#) by Bill Murray, SCDsource 2008.
11. [Post-Silicon Debug Using Formal Verification Waypoints](#) by C. Richard Ho, Michael Theobald, Brannon Batson, J.P. Grossman, Stanley C. Wang, Joseph Gagliardo, Martin M. Deneroff, Ron O. Dror, David E. Shaw. Proceedings of the 43rd Design Automation Conference, 2006.
12. [Can We Really Do Without the Support of Formal Methods in the Verification of Large Designs?](#) Umberto Rossi. 42nd Design Automation Conference, 2005.
13. [Assertion-Based Design](#). Harry Foster, Adam Krolnik, David Lacey. Springer Verlag, 2004.
14. Unified Coverage Database Application Programming Interface (API) Design Specification Document, Draft 29, April 2, 2009. Accellera [Unified Coverage Interoperability Standard](#) (UCIS) committee.
15. Formal Techniques Speed Up Interconnect Verification of SystemC Virtual Platform Models. Wolfgang Ecker, Volkan Esen, Robert Schwencker, Thomas Steininger, Michael Velten. DVCon 2008. Reported in [Formal verification expands its use model](#). Bill Murray, SCDsource 2008.

Further reading

1. [Formal property checking - what the users say](#). Richard Goering, SCDsource 2008.
2. Increasing Confidence of Complex Hardware in Safety-Critical Avionics Using Formal Methods. Kristoffer Karlsson and Håkan Forsberg. Military and Aerospace Programmable Logic Devices (MAPLD) Conference 2006.

 Add Comment - please [log-in](#) to comment

SCDsource newsletter subscribers may post a comment - [Register for free!](#)

[Back to Home Page](#)