Feature Story

Formal property checking -- what the vendors say

By Richard Goering

02/26/08

Users profiled in the first part of this report said that formal property checking tools have significant value, but require expertise and have capacity limits. In this conclusion, vendors of formal tools give their perspectives on these challenges and discuss what sets their tools apart.

In the first part of this report, engineers at IBM, Infineon, D. E. Shaw Research, MIPS Technology, and Sun Microsystems discussed the advantages and limitations of formal property checking or "bug hunting" tools, and described how their companies use them. These companies employ dedicated formal verification specialists, and are trying to make formal tools more accessible to both design and verification engineers. Most of these companies use both dynamic or "hybrid" tools, which run with simulation, and purely static property checkers, which offer full proofs and do not involve simulation.

Now we turn to the providers of formal property checking tools to get their views on such questions as ease of use, capacity, and dynamic versus static property checking. Vendors included here represent some of the most-used formal tools according to a 2007 survey of verification engineers by John Cooley.

The need for specialists

To get the best value out of formal tools, users said, companies really need dedicated specialists who know how to write complex proofs, avoid state-space explosion, and work with the tool when proofs don't complete. Tools that offer automated checks, such as clock domain crossing checks (CDC), are the exception. Vendors generally claim their tools are getting easier to use, but don't discount the value of expertise.

"I'd say most deployment of formal tools is in the hands of specialists," said Dan Benua, senior manager of corporate applications engineering at Synopsys. "Specialists are a relatively small population compared to design and verification engineers, so everybody is scrambling to figure out how to best increase deployment of these tools."

The Averant Solidify static property checker has been most successful where one person in the user company becomes a "specialist" and understands the tool well, said Ramin Hojati, Averant CEO. That engineer then becomes the "support person" for Solidify. "We have basically never been successful with a bunch of casual users," he said. "You have to get one person and train them really well."

Harry Foster, chief verification technologist at Mentor Graphics, noted that automated static property checkers such as CDC utilities don't require expertise. With less automated tools, he said, you don't need a lot of expertise for simple blocks like arbiters or timing controllers. But sometimes engineers "overreach" and try to verify complicated blocks like PCI Express or cache controllers without sufficient experience. "Advanced blocks do need advanced skills," Foster said.

Real Intent has two versions of its EnVision property checking tool – Ascent, which provides automatically-generated properties, and Conquest, which requires users to write properties using SystemVerilog assertions (SVA) or Property Specification Language (PSL). "People often say they're interested in Conquest, but they have no background or training and they're not ready to write all those properties themselves," said Rich Faris, vice president of marketing and business development at Real Intent. "So we end up working with them with Ascent, and they're a lot happier and more successful with that."

Tom Anderson, product marketing director at Cadence Design Systems, claimed that his company's Incisive Formal Verifier (IFV) really doesn't require formal verification experts. "We've been remarkably successful with mainstream RTL designers," he said. Anderson acknowledged, however, that "certainly there are advanced techniques that are used only by specialists."

Until two years ago, said Rajeev Ranjan, CTO at Jasper Design Automation, most Jasper users were formal experts. But since then things have changed. "We see a big shift in which designers and verification engineers far outnumber users who are formal experts," he said. "There is enough usability in the product that designers are able to leverage formal technology without getting bogged down in complexity."

Capacity challenges

Of all the limitations that users cite with formal property checking tools, capacity is the biggest challenge, especially with static tools. If a block is too large or a proof is too complex, a state-space explosion may occur, and the proof might not complete. It's then up to the user to rewrite the proof or

restructure the logic.

There is no simple metric that can predict when capacity limits will be overcome. Perhaps the best "metric" is the one described by Mentor's Foster: "The maximum amount of work I can do before I have to start manually doing work is what truly tells me the capacity."

Capacity is the "fundamental issue" for formal tools, said Jasper's Ranjan. "Expectations have to be set properly. If I want a complete result, I have to be willing to put in the effort." Jasper's JasperGold suite, he noted, offers a "design tunneling" technique that can minimize the amount of logic needed to prove a property. It claims to boost capacity by 10X or more for complex, end-to-end proofs, but requires interactivity. Last fall Jasper added proof accelerators that provide a more automated capacity boost.

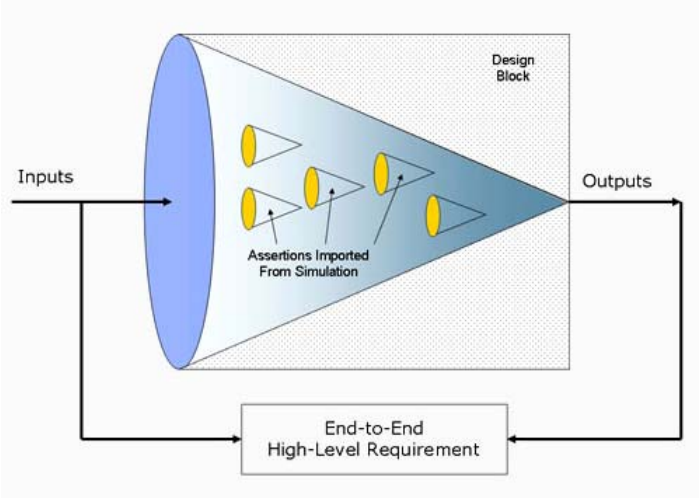

Figure 1: Jasper claims high capacity for solving high-level, end-to-end properties. (Source: Jasper Design Automation)

OneSpin Solutions claims to achieve high capacity with its 360MV static property checker. In part one of this report, 360MV user Claudia Blank, senior staff engineer for formal verification at Infineon Technologies, talked about using the tool on a TriCore processor block with 100,000 lines of code and 1,500 properties.

But Peter Feist, OneSpin CEO, noted that "there will always be properties that cannot be verified. You cannot overcome the issue of scalability purely by proof engines. It's a question of methodology." OneSpin, he said, provides "systematic guidance" that shows users how to write properties that avoid capacity problems.

"Formal tools have capacity limitations and don't always get an answer, or the amount of run time to get the answer can be very unpredictable," said Synopsys' Benua. One part of the solution, he said, is more powerful engines. Benua also noted that Synopsys is investigating the use of parallel processing for formal verification. "It's still relatively early, but the users who are trying it are getting good results," he said.

One of the strengths of IBM's RuleBase PE static property checker, said Sharon Keidar-Barner, manager of the formal verification group at IBM's Haifa Research Lab, is parallel processing. This allows concurrent verification sessions, thus reducing run times and memory consumption. Even so, she noted, "capacity and datapaths are still a concern. But verification groups tend to quickly acquire a hunch for formal-friendly blocks."

Dynamic or static?

Several users interviewed for part one of this report noted their appreciation for dynamic or "hybrid" formal tools that run with simulation. While these tools generally don't provide complete proofs, they can handle larger blocks than purely static tools, and they can leverage the advantages of both simulation and formal verification.

"Static property checking has its limitations," noted Mentor's Foster. "If I have a design with a pipeline in it, and I want to formally prove it, I would be wasting a lot of cycles if I statically started to fill up that pipeline. But if I can simulate and fill that pipeline with something interesting, extract that state out, and run formal verification around it, I've got a big win."

Coupling formal with simulation, Foster said, produces the "equivalent to billions of simulations, because I'm exploring paths the original simulation trace didn't explore. That's why you can uncover bugs using dynamic [formal verification] that you didn't uncover with simulation alone."

Dynamic verification is the primary focus of Synopsys' Magellan product, which comes with an embedded version of Synopsys' VCS simulator. "We feel formal needs to be tightly integrated into the simulation flow," said Synopsys' Benua. Magellan can be run in a purely static mode, he said, but simulation "helps the formal engine find errors more efficiently."

"A purely static tool can't focus on coverage," Benua said. "We can integrate coverage information from simulation with coverage information from formal, and help users close coverage holes and determine coverage points that are unreachable or unachievable."

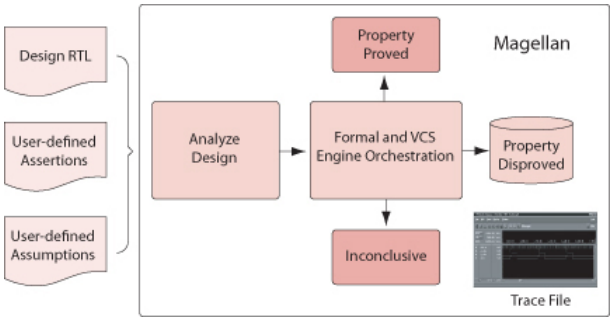

Fig. 2: Synopsys' Magellan runs with simulation to prove or disprove properties. (Source: Synopsys)

Static formal property checking tools are more numerous, however, and their big win is the capability to get full proofs. But how complete are those proofs? Can static formal property checking reduce or even eliminate the need for block-level simulation? Users queried in part one of this report generally said no, except perhaps for very simple blocks. Vendor viewpoints differ.

"We don't claim formal is a complete methodology. It really needs to be a supplement for simulation," said Benua. "We have seen a few instances where small blocks such as arbiters have been exhaustively verified with formal, but I think that's the exception. It's not going to eliminate block-level simulation."

"I haven't seen anyone in the industry give away their simulators yet," said Real Intent's Faris. "I think we need to focus on where formal tools are strong – with control logic where there are so many levels of concurrency that you can't imagine all the combinations of states. Certain blocks in a design may be very amenable for formal proofs, but others may not be amenable."

OneSpin claims that its 360MV tool can indeed provide full module-level verification, thus replacing the need for module-level simulation. The company claims to offer "gap-free" verification. This is provided through a "completeness analysis" that, according to Feist, can inspect a set of properties to make sure they cover every possible input sequence of the design under verification (DUV), and predict the expected values of all output signals. If the completeness condition is not fulfilled, 360MV informs the user, who can then decide whether or not to modify the properties.

On Feb. 18, OneSpin announced a "GapFreeVerification" process that helps users create a high-level, functional reference model consisting of a gap-free property set. The model is a formal specification of the entire expected functionality of the DUV. It's analogous to golden reference models used in testbenches. According to OneSpin, this practice can help mitigate capacity, scalability, and spurious counter-example limitations in static formal property checking.

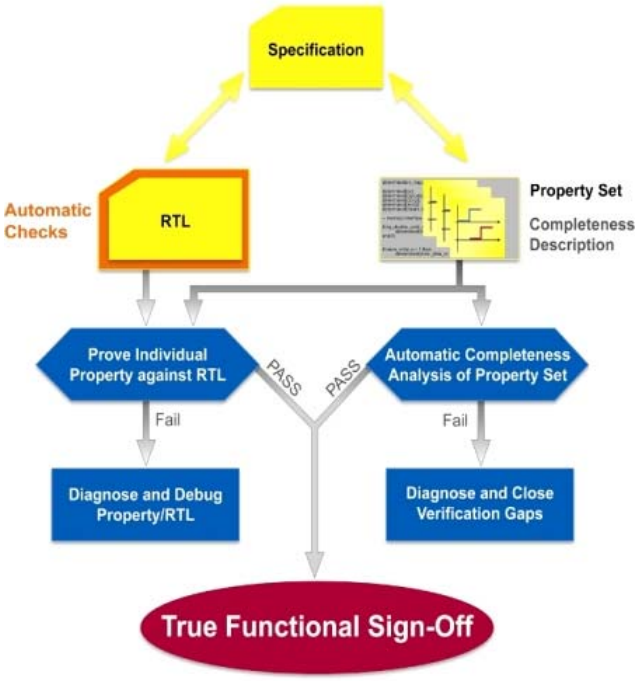

Figure 3: OneSpin offers a "completeness analysis" that claims to provide a full functional verification at the block level. (Source: OneSpin Solutions)

IBM also claims that RuleBase can replace block-level simulation. "Blocks that were fully verified using RuleBase are usually not simulated by themselves, and not even checked in a unit simulation framework," said Keidar-Barner.

Some Jasper users write comprehensive properties that "totally replace block-level simulation," according to Ranjan. He noted, however, that "sometimes they don't want to replace the entire simulation because of complex features in the block." In any case, Ranjan said, it's important to integrate formal property checking into an overall verification plan, and to develop unified coverage metrics between simulation and formal verification. For this reason, Jasper is a supporter of the Accellera Unified Coverage Interoperability Standard (UCIS) working group.

Is formal property checking only for control logic? That's a "common misperception," according to OneSpin's Feist. 360MV, he said, has been used to verify processor datapaths, as well as multimedia modules and error correction blocks.

Formal tools can be used for datapaths, but this requires "more advanced techniques," according to Mentor's Foster. He noted that there are two types of datapath blocks – those that offer data transport, and those that perform data transformations. Property checkers can be used on data transport blocks, he said. They cannot be used on data transform blocks, such as floating point units or comparators.

How the tools compare

In part one of this report, Richard Ho, researcher at D. E. Shaw Research, commented that many formal tools are "comparable" and that what's really important for users is developing the right methodology. With that caveat in mind, we offer a brief synopsis of what the formal verification vendors have to say about their tools.

Averant's Solidify, according to Hojati, stands out because of its speed and capacity, automated checks for conditions such as deadlock and CDC, and coverage capabilities. Solidify supports multiple property languages, offers source-code debugging, and provides interfaces to simulation. It also creates a testbench from a debug trace.

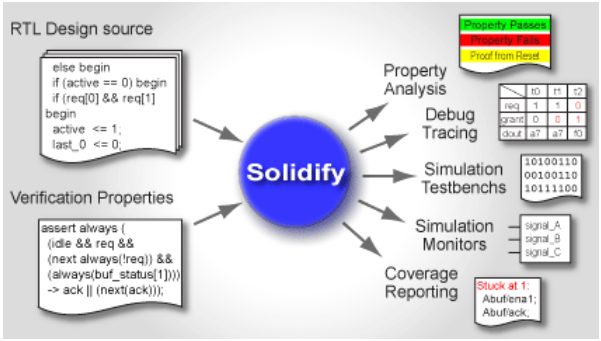

Fig. 4: Averant's Solidify offers numerous outputs. (Source: Averant)

One strength of Cadence's IFV is tight integration with the rest of Cadence's Incisive verification environment, where the same assertions can be used with simulation, acceleration, and emulation. Another strength, said Anderson, is its adoption by designers. Easy setup is one reason – "most people get up and running quickly," he said. He noted that Cadence has a dedicated team of formal experts who works continually to improve the engines.

IBM's RuleBase PE, according to Keidar-Barner, offers capabilities ranging from a fully automatic mode to "expert-tunable" features. Its parallel processing framework executes concurrent verification sessions. RuleBase offers the best support for advanced PSL constructs, she said, and in particular, has very strong support for liveness assertions.

Jasper's JasperGold is a purely static property checker, but it offers a lot of versatility. It offers a "light formal" mode for assertion-based verification, and a "deep formal" mode for end-to-end, high-level requirements. Design tunneling, proof accelerators, and modeling components called "parameterizable property macros" help users bring proofs to convergence. The InFormal Design Analyst lets designers quickly check RTL code.

Mentor Graphics' 0-In suite emphasizes versatility. Perhaps best known for its dynamic capabilities, 0-In also offers automated CDC and a purely static capability. Mentor's CDC checks, Foster said, identify metastability problems, automatically extract protocol properties, and verify reconvergence. While many static formal engines are two-state engines, 0-In has a four-state capability that includes 0, 1, X and Z.

The OneSpin 360MV static property checker isn't just a bug hunter, according to the company – it provides complete functional verification and debug at the block level. Its completeness analysis claims to detect all verification gaps and guide the user to fill them. The tool claims to handle blocks with up to a few hundred thousand lines of RTL code.

Real Intent's Envision family offers a variety of choices as well. Ascent is an automated tool that can be run early on RTL code, while Conquest is a more conventional static property checker for which users write properties. Meridian is an automated CDC tool. "Nothing is easier to use than Ascent or Meridian," said Faris.

Synopsys' Magellan can run as a static checker, but its real strength lies in its dynamic capabilities and its integration with simulation. With that integration, Benua said, users can close coverage holes and find coverage points unreachable by simulation. Users can also find synthesis-simulation mismatch issues. The integration of Synopsys' VCS simulator is a plus for teams that already use VCS, but Magellan use is not restricted to VCS customers.

Advice for new adopters

How can new adopters of formal tools be successful? Several vendor representatives emphasized the importance of training. "So much of why we're successful is high quality training," said Cadence's Anderson. "If you really want to be successful, take the time and effort to be trained well."

The best approach, said Hojati, is to take one engineer and train him really well. "I wouldn't try to get everybody trained," he said. "Train one person and let him support the tool and make sure the team uses it. Once you have that guy, you can let it propagate from there on."

Use automated formal verification first, Real Intent's Faris said. "You can get it working in a few hours and you can immediately see the benefits. Then you can go from there."

Plan ahead and take it easy, Mentor's Foster said. "Look at your high-level verification plan and decide from a high level what you want to accomplish," he said. "Then get started on blocks that do not require a lot of expertise. Timing controllers, arbiters, bus bridges, SRAM controllers – focus on those types of blocks. You won't be frustrated and you'll actually see value."

Related articles

Formal property checking – what the users say
Verification coverage measures up to higher level
Jasper eases formal proofs, updates planning tool
Achieving completeness in IP functional verification
Formal verification expands its use model

Add Comment - please log-in to comment

SCDsource newsletter subscribers may post a comment - Register for free!

Back to Home Page