News Analysis

Formal verification expands its use model

By Bill Murray

02/25/08

Papers and panels at this year's Design and Verification Conference and Exhibition (DVCon 2008) showed that formal verification is expanding its use model. Speakers described the use of formal methods in post-silicon debug at Intel, as well as in XML-assisted SystemC virtual platform verification and aggregate register verification at Infineon. Additionally, formal verification experts shared their viewpoints at a panel session.

Post-silicon debug

Erik Seligman, formal verification architect in Intel's corporate design solutions team, presented a post-silicon verification and debug methodology for validating "silicon stepping." This term refers to a specific revision of the silicon that enables the elimination of bugs found at silicon test, as well as the implementation of a small number of last-minute features. The number of permissible modifications is very small, but they must be implemented and validated with high confidence because the cost of an error is a silicon respin. Intel has combined formal and simulation methodologies into a "zero escape plan" (ZEP) approach.

Seligman gave an example of the importance of post-silicon debug. A chipset required a refresh stepping to satisfy a customer request, plus two steppings to fix bugs. The three steppings had 13 logically-significant changes that were verified successfully using formal-based ZEP.

Because the number of edits is small, verification can be highly focused. But if the fix doesn't work – or if it creates even more problems than it solves – Intel uses a "defeature" approach that disables the fix, and the design reverts to its original function. Seligman gave an example. Figure 1 shows a circuit that fails to function as intended. The fix was to add an AND gate.

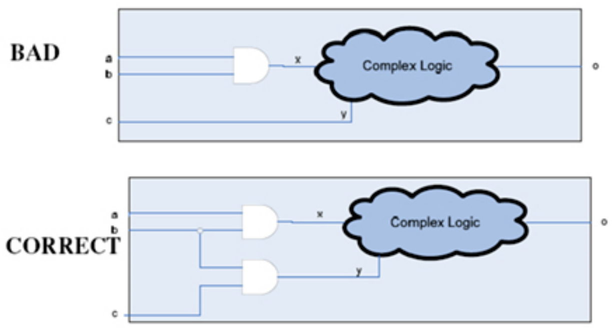


Figure 1: Faulty circuit and logic fix (Source: Intel)

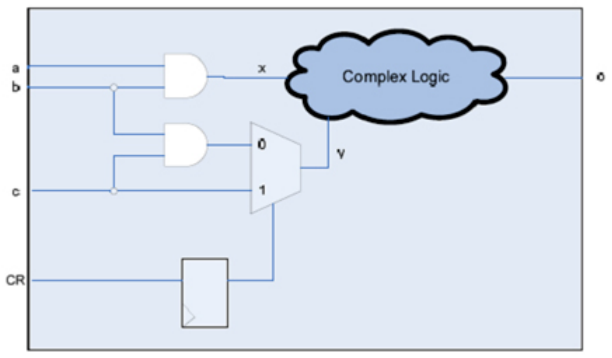


Figure 2: Fixed circuit with de-feature logic (Source: Intel)

The defeature circuit (see figure 2) consists of a mux inserted in the new logic path to the complex logic. The select input to the mux is activated by the output of a flip-flop. The input of the flip-flop is connected to the configuration register, and is thus controllable by BIOS firmware.

The challenge is that the complex logic is really quite complex. To ensure that the fix did not invalidate existing logic functionality, Intel executed formal equivalence verification (FEV) on the "old" and "new" designs. The defeature bit is tied high, thus forcing the original path through the mux. This check ensured that Intel could revert to the original circuit if the fix did not work as required.

Seligman gave another example in which the fix was to recode a state machine. Of course, this modifies the definitions of some states as well as the transition logic. Because Intel's sequential FEV methodology was not available for the chipset projects in question, this required a workaround to change it to a formal property verification (FPV) problem (see figure 3).

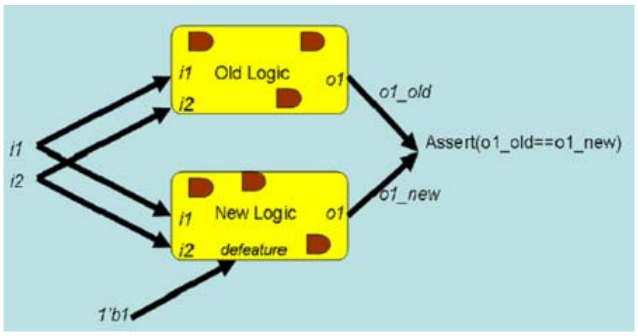


Figure 3: Sequential defeature FEV (Source: Intel)

Intel verified the change by creating a wrapper module that contained the old and new state machines with inputs tied together. Intel then created an assertion that their outputs must be identical. Intel then ran FPV tools to verify the assertion.

In response to an audience question about whether Intel uses the stepping proofs in regression testing, Seligman said "yes. But I should qualify that by saying that it may not happen after the formal expert has moved on to another project."

An audience member asked whether moving from equivalence checking to property checking introduces complexity problems. Seligman answered "yes, it always increases complexity versus combinational equivalence. We handle it at a very local level with highly focused changes and verification. We have to find the lowest level at which the change is effective. Comparing two complex clusters is much more difficult."

Another question from the audience: Do you feed back data into the pre-silicon verification process? According to Seligman, "we don't have a formal process for that. If a block is to be used on the next chip, we pass the verification data back for that purpose."

SystemC TLM verification

Wolfgang Ecker, principal engineer in Infineon's communications solution business group, presented a methodology that formally verifies SystemC virtual platform interconnect at the transaction level modeling (TLM) level. Using an XML description based upon the IP-XACT standard, the methodology uses assertions extracted from the architectural model to automatically check TLM interconnect properties.

The approach automatically generates several thousands lines of assertion code, reducing SoC design and verification effort by several engineer-months. It also simplifies bug hunting in the integrated design because verified interconnectivity localizes and reduces the bug search space.

Interconnect verification ensures the correct operation of functionality such as bus addressing and external interface multiplexing, as well as of the integration as a whole. The Infineon formal methodology attacks the interconnect verification problem at the electronic system level (ESL). This not only simplifies and speeds interconnect verification later in the flow at RTL – it also expands the ESL infrastructure to maximally exploit SystemC TLM models for early system description, verification, and analysis.

The barrier to SystemC TLM interconnect verification is that SystemC does not support native temporal assertions. Previous efforts to rectify this deficiency in SystemC apply assertion-based verification (ABV) only at the RT level. To describe TLM assertions, Infineon uses its proprietary Universal Assertion Language (UAL), which is similar in structure to PSL and SVA. The language operates with the usual multiple layers – modeling, verification, property, sequence, and Boolean (see figure 4).

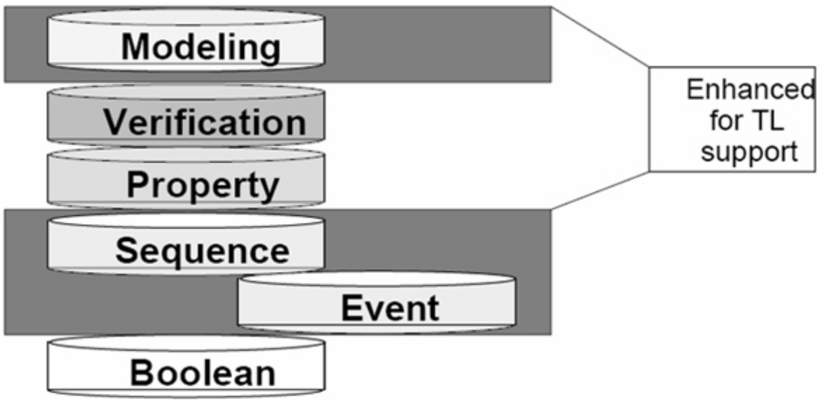


Figure 4: Infineon enhances assertion language with event layer

In Infineon’s language, however, there is an additional layer – the event layer – that specifies temporal sequences. These sequences are evaluated using transaction events, which are signaled both at the start and at the end of a transaction call, allowing immediate assertion checking at both points. This approach does not interfere with the SystemC simulation kernel, and thus maintains the original timing integrity.

Using IP-XACT’s vendor extension capability, Infineon has implemented extensions to the IP-EXACT schema to describe all TLM connectivity. The extensions enable generators to automatically generate a top-level netlist, complete with all interconnects. To counter-check the correctness of these generators, Infineon uses automatically-generated assertions. The assertion generator uses the same data as the top level netlist generator, but executes independently of it, so the assertions independently check the correctness of the top level generator.

Infineon claims that the approach has been applied to several industrial virtual platform models.

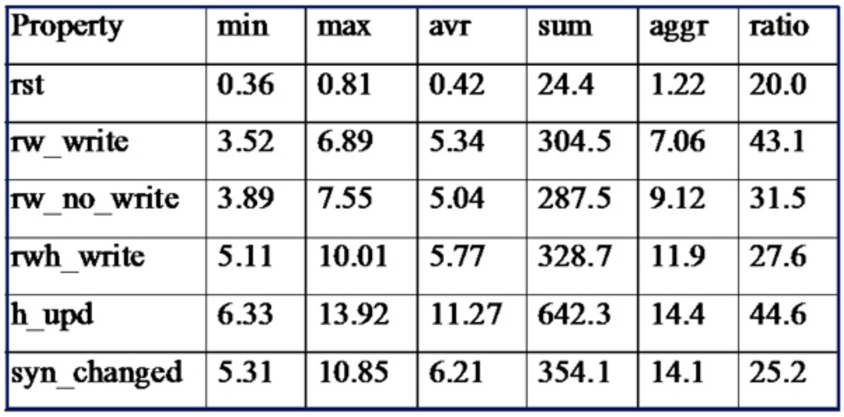
Aggregate register verification

Holger Busch, senior staff engineer for verification in the automotive, industrial and multi-market group at Infineon Technologies, described an XML-assisted aggregate property checking methodology that performs a “vast” number of checks simultaneously to verify the special function registers in a control unit. The functionality of such registers constitutes a significant proportion of the control unit’s specification. Consequently, simultaneous checking using aggregate properties can verify the complex functionality of modules with considerably less effort than individual property checking or simulation. In addition, the methodology’s completeness checking identifies gaps in the property set, which can then be filled by the verification team.

Infineon specifies the register functionality in Adobe FrameMaker, which allows XMLSpy – an XML modeling and debug environment – to automatically extract register data such as bit-field structures and types, addresses, and access restrictions in XML format. Two different Perl parsers then arrange and transform the XML data into data macros. These data macros are then incorporated into generic aggregate properties – developed in OneSpin Solutions’ 360MV formal environment – that collectively specify the operations of entire sets of registers. This aggregate properties approach greatly reduces the quantity of proofs required to quite a small number.

The technology’s completeness checking formally proves that a given set of properties fully checks a given set of output signals for every specified input scenario. If an input scenario has been omitted, completeness checking detects the gap and provides diagnosis data that enables the verification team to devise additional properties to cover the missed scenario. Conversely, when an output signal is not covered by the property set, the tool generates a counter-trace that highlights the non-deterministic behavior. Again, the verification team can take remedial action.

Representative results are shown in table 1. It shows the run times in seconds for six aggregate properties, together with those of the “equivalent” individual properties, for 58 single registers in the system control unit of an automotive 8-bit controller. The column ‘aggr’ refers to the aggregate properties, while the columns ‘min’, ‘max’, ‘avr’ (= average) and ‘sum’ refer to the corresponding individual property checks. The aggregate properties execute 20X to nearly 45X faster.



Property	min	max	avr	sum	aggr	ratio
rst	0.36	0.81	0.42	24.4	1.22	20.0
rw_write	3.52	6.89	5.34	304.5	7.06	43.1
rw_no_write	3.89	7.55	5.04	287.5	9.12	31.5
rwh_write	5.11	10.01	5.77	328.7	11.9	27.6
h_upd	6.33	13.92	11.27	642.3	14.4	44.6
syn_changed	5.31	10.85	6.21	354.1	14.1	25.2

Table 1: Aggregate property sets execute up to 45X faster than single property sets (Source: Infineon)

The complete verification used 17 aggregate properties in total, corresponding to 986 individual properties. Busch observed that “the time to create the generic aggregate properties was about three days. The aggregate properties are written once for any given product family, and can be reused and enhanced in derivative designs in the same product family.”

Busch pointed out that the automatic extraction of register data guarantees consistency between register specifications and register verification – something that coverage metrics and even formal completeness proofs cannot achieve.

In response to an audience question about the verification effort, Busch said “I can only guesstimate, because we don’t use simulation on these blocks. It’s probably about 30% of the effort that simulation would require.” But that is not an apples-to-apples comparison. Busch emphasized that the real value of the formal methodology is in the thoroughness of the verification. He noted “The formal verification completely checks all register functionality. This is not the case with simulation, which is generally terminated after achieving coverage of only a subset of register behavior.”

Answering an audience question about silicon success, Busch confirmed that the methodology has been applied to 8-bit, 16-bit, and 32-bit microcontrollers, which have gone into full production.

Experts’ panel – a cross-section of opinion

The panel “Driving design verification results: formal verification comes of age” discussed the status and progress of formal adoption. The panelists were two users – from Intel and IBM – and three vendors, from Cadence, Jasper and Mentor.

Limor Fix, principal engineer and associate lab director at Intel Research, said that "formal tools are growing in maturity. Assertion development, assertion libraries, compilers and compiler-generated assertions are all indications that the tools are being supported for real work. Some early ideas for coverage are emerging, but they are not yet developed into mature tools and methodologies."

Avi Ziv, Research staff member at IBM's Haifa Research Lab and IBM's server technology group, said "formal is moving into a position as an integral part of verification, and is changing the focus in verification from simulation. It's important to know the major difference between formal and simulation: simulation sees events, formal checks aspects of those events. The verification must identify holes in formal properties with simulation, and ensure full coverage of all constraints. The challenge for formal methods is to find bugs in the design before you spend any time on the simulators."

Axel Scherer, formal architect at Cadence Design Systems, said "formal must take some work away from the rest of the verification tools and reduce simulation cycles. Contributions from formal have to be measurable."

Rajeev Ranjan, CTO of Jasper Design Automation, said "formal's productivity increases can help to ensure correctness and can eliminate redundant verification. The levels of risk for functions and use modes should drive the verification plans."

Harry Foster, chief engineer, design verification and test at Mentor Graphics, said "even designs that are very difficult to check, such as a cache controller, can be completed by either set of tools [simulation and formal]. This is not a function of the tools, but shows the need for people with skills."

Wrap-up

The experts' panel focused on formal's efficacy and productivity in the traditional RTL verification task. But the paper sessions show that formal is expanding its use model. The pre-silicon RTL formal verification approach is now used in post-silicon debug and has been adapted for use in SystemC TLM verification. Also, formal verification now encompasses a complete aggregate properties methodology that deploys relatively few proofs to verify very large numbers of registers and their complex interactions. The world of formal is changing.

Tets Maniwa contributed to this article.

Related articles

Formal property checking -- what the users say
Formal property checking – what the vendors say
Register description format gets 'Spirit' of standardization

References

Zero Escape Plans: Tying Together Design, Simulation, and Formal Methods for Bulletproof Stepping Validation. Erik Seligman, Carl Dreyer, Ken Haren, Raman Nayyar. Intel Corporation. DVCon 2008.

Formal Techniques Speed Up Interconnect Verification of SystemC Virtual Platform Models. Wolfgang Ecker, Volkan Esen, Robert Schwencker, Thomas Steininger, Michael Velten. Infineon Technologies. DVCon 2008.

Generation of Complete Aggregate Formal Properties. Holger Busch, Infineon Technologies. DVCon 2008.

Add Comment - please log-in to comment

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

Back to Home Page

All materials on this site Copyright © 2007-2009 Tech Source Media, Inc. All Rights Reserved | Privacy Statement