Can We Really Do Without the Support of Formal Methods in the Verification of Large Designs?

Umberto Rossi
STMicroelectronics
Agrate Brianza - Italy
umberto.rossi@st.com

Categories & Subject Descriptors: B.5.2 [RTL Implementation]: Design Aids---simulation,verification
B.7.2[Integrated Circuits]: Design Aids---simulation,verification

General Terms: verification, reliability

1. The mystery question

From the IC industry's standpoint, the incubation of formal methods for deployment in EDA verification flows has been very long and is still occurring. Formal methods applied at functional verification have interpreted different roles - e.g. are they good for proving correctness or for catching bugs in deep behavioral corner cases - have played with different techniques - e.g. BDD's, SAT, ATPG, symbolic - and finally have federated with simulation for the purpose of achieving coverage closure. Further, in the last 10 years a fair number of start-up's have emerged, that have been acquired by major vendors in the meantime, and new start-up's have been appearing also this year.

What are the reasons that make the IC industry to accept an unusually long maturation period of the formal methodology & tools and the EDA vendors to put money in the basket of their developments ?

2. Whose affair is this ?

I lead a group devoted to investigating new verification techniques for System on Chip (SoC) and System in Package (SiP). The general approach of the design team leaders is to allocate a significant part of the verification budget to Random Test Pattern Generation (RTPG) driven simulation, by the use of popular Testbench Automation tools, but then sometimes one design team leader fears exposure to unexpected bugs, for a number of different reasons: the team has pioneered a new design configuration never tried before, i.e. it is not possible to find an existing Verification IP (VIP) already proven and working; the design and the verification teams are distributed in the company organization and geographically far away from each others - multiple design teams and multiple verification teams; total quality commitment is becoming ever more demanding in terms of defectiveness margins and some actions have to be taken to improve the verification flow. Very often the budget to formal verification is not planned in advance: usually formal verification is asked to clear uncertainty areas but the different usage options with respect to simulation are not clearly perceived.

DAC 2005, June 13–17, 2005, Anaheim, California, USA.
ACM 1-59593-058-2/05/0006.

3. The scenario at STMicroelectronics

In STMicroelectronics, the verifications of a SoC or a SiP are becoming everyday more challenging. At the 90nm and 65nm technology nodes, hundred million transistors will be implemented in few tenth square millimeter of silicon. Even considering that a fraction of the chip will be occupied by sparse logic, tens of million gates RTL is foreseeable in the next years: no one is going to design such an RTL from scratch and as a single block. Another important factor to be considered is the cost of masks that, at the current technology nodes, amounts at few million dollars per chip. A possibility to reduce the impact of the huge mask cost is to implement on the chip the maximum number of IP's that can be then shared among different applications. Actual SoC's are built over platforms including a processor plus a standard communication infrastructure where pre-defined or user-defined IP's may be plugged in. Another advantage of this approach is that also the cost of the RTL-to-layout process can be conveniently shared; this process is demanding increased effort, as exploration of large solution space is mandatory for reasons like: (i) the impact of low-power applications - in case of portable device; (ii) the predictability of the physical implementation becoming ever poorer and (iii) the manufacturability requirements forcing design revisions due the effects related to nanometer technology. In such a situation there will be less chance for last minute functional fixes and a single application success won't be enough to guarantee the correctness of the overall implementation. For the exposed reasons there is the absolute need to introduce IP's that are 100% functionally guaranteed and the statement "the IP has been implemented once in a working silicon therefore it is correct" will not be accepted any more.

Concerning RTL validation, in ST the penetration of formal methods is marginal with respect to testbench based methods; further, no formal-based model checker clearly prevails over the other ones concerning both performance & usability, therefore three products are adopted. Overall, the most important and visible results have been obtained in bug hunting use.

The "RTL to Layout" digital design flow in ST is based upon automated synthesis & IC implementation. Combinational Equivalence Checking (CEC) is applied at RTL-to-Gate verification and Gate-to-Gate verification. CEC has been also tried in the custom circuit verification as far as the Automatic Transistor Abstraction could be used to obtain an equivalent gate-level model of a physical layout; this kind of extension seemed to gain some popularity a few years ago, but presently its domain is confined to library cell verification. The success of CEC is not reversible and the market has actually designated the only two survivors. Improvements are still possible in the area of "hard" arithmetic descriptions - where one product is proven superior so far - of circuit re-timing/pipelining, in the direct link between RTL and tapeout netlist verification. This last point is the real final purpose, otherwise a large part of the verification community will continue to perceive CEC as a means to

verify the vendor's tools and not the designer work. By setting a proper automated verification flow, possibly working at different hierarchical layers, the verification of hundred million transistor device is at hand.

4. How and where Formal Methods can make the difference

The more convincing fact, that makes a design team leader approach formal methods, is their ascribed ability to find unexpected erroneous behaviors of the RTL description being validated. This is quite an odd situation, as we would expect to formally prove the correctness of a description, not the detection of a defect in a deep corner case; the reason of this, say, collateral advantage is in the "use model" of formal based tools. In few words, this use model can be summarized in the sentence: prove that a simple behavior is exhibited by a circuit of arbitrary complexity. Examples of simple behaviors are: data integrity, algorithmic properties, well formed expressions. In other words, by using formal methods the verification activity may concentrate just on the expected behavior of the Design Under Test (DUT), because the formal analysis itself takes care of the DUT implementation details. By direct or random simulation, often the verification engineer has to visualize an abstract model of the DUT, for the purpose of the controllability of certain internal events and the observability of their effects. In our opinion, it is the use model the reason of success of formal methods. The more complex the use model becomes the less chance to get verification closure exists; this is the case when design interface becomes very complex, requiring elaborated input constraints, thus exposing verification process to the tedious discovery of false negatives; in our experience this mainly happens when the verification plan is poor and the formal task is left as the last resort to the uncovered verification areas. With proper resource budget, Assertion Based Verification (ABV), can conveniently take advantage both from traditional simulation technique and formal methods by providing a common measurement criterion for the effectiveness of the verification job, the so called assertion coverage. Unfortunately the ability to prove assertions does not scale well with the design size. A remedy to this limitation is to partition the proof into smaller cones of influence inside the circuit, but the effect of this is (i) again to disrupt the simple use model with (ii) the risk of implicitly neglecting portions of the DUT state space by excessive constraining, so exposing the verification to the danger of false positives. The verification of the RTL is a tradeoff between the cost of the process and the time budget, thus requiring appropriate planning; to this extent formal based tools are more exposed to the lack of predictability as very often the answer is "don't know" or "this assertion is true for N cycles".

5. Application Areas

In our vision formal methods represent a "continuum" of different solving capabilities. As an example, consider the request to "apply a one-hot enabling schema to a tri-state net": in the frame of a HDL "case" statement may be just a matter of checking exclusive entries, but in general it may require proving a temporal property, if the control is driven by a FSM.

The analysis of the HDL code by itself provides a huge number of checks that can be automatically extracted. They typically concern extended semantic checks, e.g. "check for out-of-range on the result of arithmetic operations", or structural checks, e.g. "check for FIFO underflow/overflow". Other kinds of analysis like, code reachability, Clock Domain Crossing check, False Path & Multicycle Path are able to generate thousands of assertions to be proven as properties of the RTL. Simulation may help to check if some of those assertions are false, but what is expected is the full proof of all those properties, at the level of single IP. Proving this kind of properties at IP level saves a significant effort in writing a testbench that represents an extra to the testbench that will be used for the final top level application. This kind of capability concerns the so called white-box verification and is to be put in the hands of the designer.

IP black-box verification brings significant advantage to the verification engineer by using formal verification. Typically this is structured in a set of assertions, to be used as constraints and checkers, simple and general. Examples of such assertions are: "if the request signal is asserted and no grant signal is asserted then the request must be kept high", "any attempt to write from a port into a memory region that is reserved to another interface must generate an error". In our experience the full proof of such properties is often unfeasible, and we typically have to rely upon a bounded proof or constrained restricted. The support of "liveness" properties represents a plus for writing simple assertions - constraints & properties - although it is still unlikely that any complex design may be addressed by such properties.

6. The present and the future

The success of formal methods will depend both on the early introduction of this technique in the verification plan - this is more about convincing verification engineers rather than design team leaders actually ! - and in the appropriate choice of product segments. As an example, STMicroelectronics is a leading provider of ASSP's for automotive applications, and this offers a good chance to the verification activity in general. Automotive market is becoming ever more demanding concerning safety certification. Particularly in this segment, the product qualification can take a couple of years before the actual mass production starts, therefore there is time for extending the validation period of the SoC, although at different costs in terms of repairing options - i.e. the later a functional bug is found the more expensive the functional fix is.

From the user point of view, verification products still lack integration between dynamic analysis and formal assertion analysis. A closer cooperation or a better merger of the respective results may constitute a significant plus for the users.

Formal methods may deserve important improvements in the System Level verification area.

Commercially available IP's and custom IP's have very often been exposed to sneaky bugs. One of the reasons is related to the fact that very often the IP's are parametric, e.g. in the dimension of the data ports, address ports, internal memories like stacks, FIFO's etc., therefore the correctness cannot be tried on the millions of configurations made of the different parameter options. Here an effort is expected from the industry by the CAD vendors, sooner or later.

Having correct IP's is not enough for verifying the correctness of the whole system. Formal methods have to extend their effectiveness by including the so called Transactional Level Models (TLM) in their analysis, that abstract the mechanisms that SoC blocks use to communicate and synchronize among themselves. Two major problems have to be faced: the first is that TLM includes a range of abstraction levels and the second is that a mapping between TLM assertions and RTL assertions should be provided and formally proven. This seems still an area of research, although a product in this segment is appearing in the market.