VisSAT: Visualization of SAT Solver Internals

Robert Wille   André Sülflow   Christian Genz   Rolf Drechsler

Institute of Computer Science
University of Bremen
28359 Bremen, Germany
{*rwille,suelflow,genz,drechsle*}*@informatik.uni-bremen.de*

Verification gains an increasing amount of design costs in modern VLSI-CAD. With increasing complexity and continuous demands for correctness, the application of formal methods in verification becomes indispensable. SAT solvers [1] are essential tools building the basis of many formal verification approaches.

Typically, the following flow is thereby applied: The problem is encoded into an instance of Boolean satisfiability which is passed to the SAT solver. The SAT solver returns either a satisfying assignment or proves that no such assignment exists. From this result, a solution of the verification problem is deduced. In this sense, the SAT solver is utilized as a black-box.

However, hard to solve problem instances cause long run times or even time-outs, which effect the duration of subsequent re-spins. When facing such challenging designs, verification engineers can modify the problem instance, the decision heuristics, or parameters of the solve engine. Internal data structures of a SAT solver thereby give valuable information concerning the traversion as well as the structure of the search space. This may be used to speed up the solve process. Unfortunately, such information is hard to extract.

In this work, we present the graphical interface *VisSAT* to visualize statistics of a SAT solver. An evaluator collects statistical data of the solve process, e.g. the number of decisions or conflicts a circuit signal is involved with. Afterwards, this data is mapped to an RT or gate level schematic of the underlying verification problem. Therefore, the visualization engine from [2] is utilized.

Using *VisSAT*, the user is pinpointed to critical parts of the problem instance, e.g. hotspots with a large emergence of conflicts. To support modifications of the problem formulation, the visualization engine allows cross-probing of the statistical data to the source code of a design. Thus, a designer gets feedback about critical statements in the design which helps to reconfigure the solver or alter the problem formulation accordingly. The explicit choice and the application of an optimization technique stays in the hands of the engineer.

As an example, Fig. 1 shows the distribution of conflicts occurred while solving a property checking instance. More precisely, the correct behavior of multiplication in an ALU circuit is verified. The coloring intuitively differentiates parts with a large number of conflicts (highlighted in red) from parts with a smaller number of conflicts (highlighted in yellow) and from parts without conflicts (highlighted in green), respectively. As indicated by the red signal, the multiplier module frequently causes conflicts. With these information observed, the designer can apply changes to the SAT solver parameters (e.g. preferring signals of the multiplier) or modify the design (e.g. replacing the multiplier by shifters), respectively.

References

[1] N. Eén and N. Sörensson, "An extensible SAT solver," in *SAT 2003*, ser. LNCS, vol. 2919, 2004, pp. 502–518.

[2] Concept Engineering GmbH, *RTLvision PRO*, http://www.concept.de, 2009.

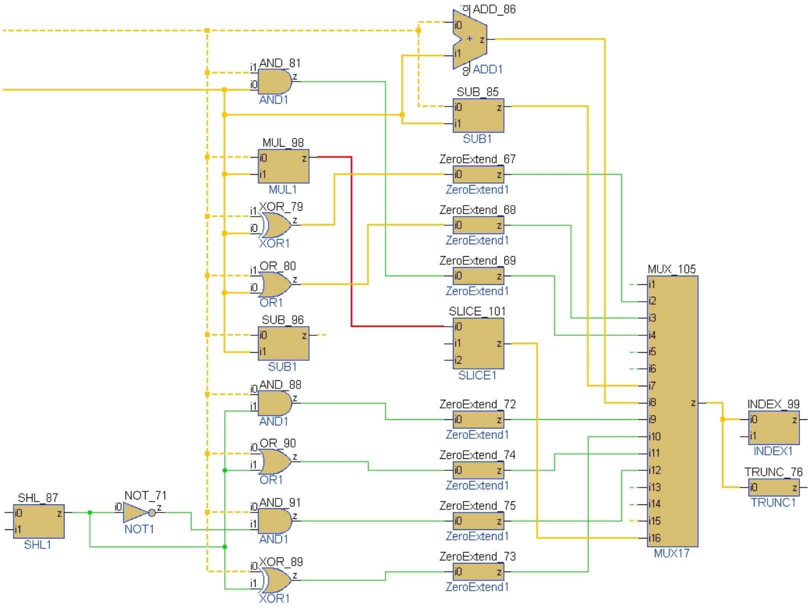

Fig. 1. Visualization of conflict statistics