Applications of the Open Source HW Design Framework zamiaCAD

Anton Chepurov, Valentin Tihhomirov, Syed Saif Abrar, Maksim Jenihhin, Jaan Raik
Tallinn University of Technology, ESTONIA
{ anchep | valentin | saif | maksim | jaan }@ati.ttu.ee

Abstract

zamiaCAD is a modular and extensible open source framework for advanced hardware design, analysis, and research. Implemented with high performance, scalability and usability in mind, zamiaCAD offers in itself an elaboration core, upon which several different HW related tasks are performed. An Eclipse plug-in with a built-in simulator is used as zamiaCAD's graphical front end. The framework is capable of handling very large industrial designs, such as a SoC made of 3500 Leon3MP-s.

1. Introduction to zamiaCAD

For powering zamiaCAD [1] applications a custom designed and highly optimized for scalability and performance HDL independent object database ZDB is used to accommodate arbitrarily large designs after they have been fully elaborated by language dependent front-ends (so far only VHDL). Full elaboration, resulting in a set of scalable instantiation graph (IG) data structures, allows clients to perform all kinds of tasks described below in detail (Fig. 1). The abstract syntax tree (AST) model serves mainly as an intermediate step in the IG elaboration process and supports some editor features. Design database is automatically and efficiently persisted to disk to save time on later elaboration.

2. Application of zamiaCAD

The open-source zamiaCAD framework addresses mainly advanced HW RTL design, verification and analysis, and offers the following functionality:

Code entry features comprise syntax highlight, code entry, content assist (identifier auto-completion), extensible set of HDL templates and an incremental IG model builder. zamiaCAD's HDL code editor is based on Eclipse's editor.

Static analysis (SA) tasks and navigation are feasible due to the fully elaborated IG design model, where all identifiers (including types) are resolved. SA tasks include tracing of parts of signals, precise global tracing, precise matching of overloaded subprograms, tracing through generate-statements, advanced signal value annotations (e.g. annotating only one bit of a vector), computing expressions on the fly, source-less and sink-less signal detection, FSM recognition (work-in-progress), code outline and code hierarchy view, declaration search.

VHDL simulator, implemented in accordance with the IEEE Standard VHDL Language Reference Manual. Simulator also provides code coverage measurements, value/timing source back-annotations and importing of waveform files (VCD).

Debugging features include an experimental algorithm for automatic design error localization, which brings together SA and simulator to narrow down the search space where the design bugs are to be localized.

SystemC generation out of VHDL. While the ultimate goal here is to explore and define methodologies of abstracting from RTL to TLM, and generating SystemC TLM model directly from the VHDL RTL with zero/minimal manual interaction, currently working prototype is already capable of generating a cycle-accurate, pin-accurate SystemC model out of RTL VHDL, thus preserving the level of abstraction. SystemC TLM generation would enable IP design houses to provide TLM descriptions of their legacy RTL libraries, increasingly requested by their customers developing large modern SoCs.

RTL graphs generation, which provides a designer with a graphical view of the written code and shows inferred memory elements (work-in-progress). This obviously requires partial synthesis and also drives zamiaCAD team towards creating a fully formalized synthesis algorithm and its open source implementation within the framework (work-in-progress).

As a framework, zamiaCAD offers a scripting interface, implemented in *JPython*, for controlling external tools, such as *ghdl*, *ModelSim* etc. JPython scripts also make zamiaCAD itself easier to use from the command line.

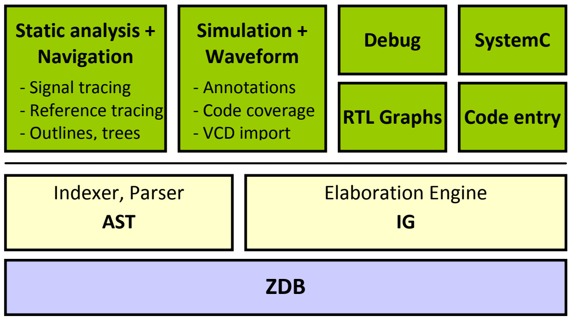

Figure 1. zamiaCAD structure and applications

References

[1] zamiaCAD webpage: http://zamiacad.sf.net/

Acknowledgements

The authors would like to thank Guenter Bartsch, the main developer and project founder of zamiaCAD, for the source code and the help throughout the development and debug, and IBM Boeblingen, Germany for running tests on the IBM designs.

The work has been supported in part by Estonian Science Foundation grant 8478, EU project FP7-2009-IST-4-248613 DIAMOND, EU through European Regional Development Fund and Estonian Information Technology Foundation (EITSA).