CRAVE: An Advanced Constrained RAndom Verification Environment for SystemC*

Finn Haedicke[1] Hoang M. Le[1] Daniel Große[1] Rolf Drechsler[1,2]

[1]Institute of Computer Science, University of Bremen, 28359 Bremen, Germany
[2]Cyber-Physical Systems, DFKI GmbH, 28359 Bremen, Germany
crave@informatik.uni-bremen.de
http://www.systemc-verification.org

I. Introduction

A huge effort is necessary to design and verify complex systems. Abstraction-based methodologies have been considered resulting in *Electronic System Level* (ESL) design. A prominent language for ESL design is SystemC [1], [2] offering different levels of abstraction, interoperability and the creation of very fast models for early software development. For the verification of SystemC models, *Constrained Random Verification* (CRV) plays a major role. However, the existing *SystemC Verification* (SCV) library [3] has several deficiencies which limits its practical use.

We present CRAVE, an advanced *C*onstrained *RA*ndom *V*erification *E*nvironment for SystemC. The main features of CRAVE are described in the following section. For more details from the user perspective we refer to [4] while [5] describes the underlying constraint solving framework *metaSMT* which allows an independent programming by providing a unified interface to different solvers.

II. Features of CRAVE

To overcome the limitations of the SCV library CRAVE provides the following features:

- **New constraint specification API**
  An intuitive and user-friendly *Application Programming Interface* (API) to specify random variables and random objects has been developed.
- **Dynamic constraints and data structures**
  Constraints can be controlled dynamically at run-time. Moreover, constraints for elements of dynamic data structures like e.g. STL vectors can be specified.
- **Improved usability**
  Inline constraints can be formulated and changed incrementally at run-time. Furthermore, automatic debugging of unsatisfiable constraints is supported.
- **Parallel constraint-solving**
  BDD-based and SAT/SMT-based techniques have been integrated for constraint-solving. A portfolio approach is used to enable very fast generation of constraint solutions.

*This work was supported in part by the German Federal Ministry of Education and Research (BMBF) within the project SANITAS under contract no. 01M3088 and by the German Research Foundation (DFG) within the Reinhart Koselleck project DR 287/23-1.

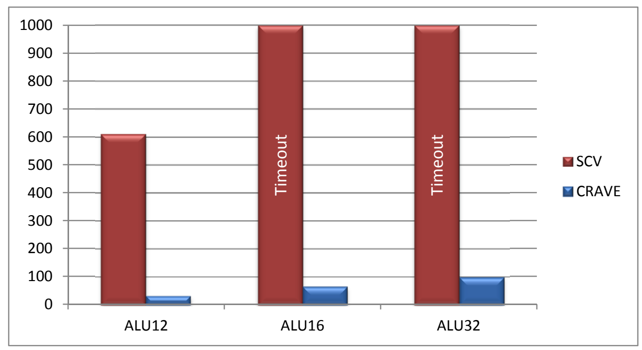

Fig. 1. Comparison of stimuli generation performance

III. Selected Examples

We demonstrate one of the new CRAVE features in the following code snippet. We show how to formulate a constraint on a dynamic data structure, here a random vector:

```
rand_vec< unsigned int > data;
...
constraint(0 < data().size() && data().size() < 1024);
constraint.foreach(data, _i, data()[_i] <= data()[_i−1] + 5);
```

The vector should contain at most 1024 elements expressed by the first constraint. The second constraint ensures that two consecutive elements of the vector have at least a difference of 5. Here, *foreach* allows to iterate over the random vector *data()*, *data()[_i]* refers to a symbolic vector element and *data()[_i - c]* to a previous element relative to *data()[_i]*.

Moreover, we demonstrate the constraint-solving efficiency of CRAVE using a scalable ALU input constraint set. Fig. 1 depicts the results obtained for bitwidths of 12, 16 and 32 when generating 100,000 stimuli using the SCV library and CRAVE. With the SCV library only solutions for a bitwidth of 12 could be determined, for the larger bitwidths the timeout limit of 1h was reached. In contrast, CRAVE was faster even for much larger bitwidths.

References

[1] Accellera Systems Initiative, “SystemC,” 2012, available at http://www.systemc.org.
[2] D. Große and R. Drechsler, *Quality-Driven SystemC Design*. Springer, 2010.
[3] *SystemC Verification Standard Specification Version 1.0e*, SystemC Verification Working Group, http://www.systemc.org, 2003.
[4] F. Haedicke, H. M. Le, D. Große, and R. Drechsler, “CRAVE: An advanced constrained random verification environment for SystemC,” in *MBMV Workshop*, 2012.
[5] F. Haedicke, S. Frehse, G. Fey, D. Große, and R. Drechsler, “metaSMT: Focus on your application not on solver integration,” in *DIFTS'11*, 2011, pp. 22–29.