SystemC and the Future of Design Languages: Opportunities for Users and Research

Grant Martin
Cadence Berkeley Labs
gmartin@cadence.com

Abstract

There has been a lot of discussion, and a lot of confusion, about the various existing and new design languages recently. SystemC, SystemVerilog, Verilog-2005, e, Vera, PSL/Sugar, UML, Analogue and Mixed-Signal versions of Verilog and VHDL make the world a veritable alphabet soup. This paper briefly looks at the evolving world of design languages from a SystemC perspective. Although a design "language war" may seem imminent, there are strong prospects for peaceful coexistence between languages, and flows that connect them together. And such flows give tremendous opportunities for users of languages to significantly improve their methodologies. In addition, the needs of advanced system and System-on-Chip (SoC) design turn up a number of interesting research opportunities for those involved in language-based design. The paper will finish by covering some of these methodology and research possibilities, including those opened up by further evolution in SystemC to include SW task and OS scheduler modelling.

1. Introduction

Recently there has been a tremendous discussion about design languages. Added to the discussion about SystemC and SystemVerilog are the many other design and verification languages extant: IEEE 1364 Verilog, VHDL, Vera, e, PSL/Sugar, and the analogue and mixed-signal versions of Verilog and VHDL. Designers of complex SoCs involving internally created and externally created IP blocks might arguably be asked to deal with all nine of these design languages.

Designers are quite likely to be very confused about these languages - the right use models, and where they are going. The IEEE 1364 Verilog committee has a plan to specify next generation Verilog - 1364-2005 (although it may take a little longer for an approved standard). Verilog 2005 will be based on suggestions for new capabilities from vendors, users and industry associations such as Accellera (likely to donate SystemVerilog, in some version, to the IEEE).

In the meantime, Open SystemC International (OSCI) has a well-formulated plan to standardise SystemC 2.1 via the IEEE, and will follow that up with further open community work to evolve the language based on user needs. For example, activities in 2003 include clarifying and defining the most useful abstraction levels for "transaction-level modeling"; looking at defining a standard synthesisable subset of SystemC; and continuing to examine, in a SystemC 3.0 context, the modelling of software tasks and operating system scheduling algorithms to allow more realistic and accurate modelling of mixed HW-SW systems. And of course, during 2003, the SystemC Verification Library (SCVL) has achieved production status, and the Language Reference Manual for 2.0/2.1 has been completed.

2. SystemC for system-level design

Despite the noise of the press and of standards bodies, which at times places SystemC and SystemVerilog in opposition or competition, it is readily seen that they have complementary roles in the process of moving designs from specification to implementation. From the early definition of SystemC 2.0 and onwards, it was absolutely clear that SystemC is a system design language [1]. It allows design teams to model, and verify designs expressed at true system levels of abstraction, refine these to reflect implementation choices, and finally link the system model to hardware design implementation and verification. SystemC enables the creation of a transaction-level prototype model of a design platform, which allows high-speed verification of system-wide testbenches, prior to further refinement. SystemC is a natural modelling language for dealing with all manner of issues concerned with mixed hardware-software systems and platforms.

3. The evolution of HDLs

Verilog is of course a primary design implementation and verification hardware description language, and Verilog-2005 will strengthen this. Enhanced features for hardware specification and synthesis, verification testbench creation, and IP protection will improve its use for advanced hardware design. Verilog-2005 will be able to link to design and verification models created in SystemC using advanced simulation environments; thus hardware designers will be able to reuse system level models to validate their designs in a true system context.

However, many of the improvements to Verilog are adding features already present in the other important HDL: VHDL has had a number of these advanced capabilities for many years. Neither Verilog nor VHDL, however, allow designers to work at the true system levels of abstraction.

This reinforces a point often forgotten about design languages - they have ceilings, as well as floors. Just as SystemC is not an optimal language for HDL and gate level design, Verilog-2005 (and SystemVerilog) is not the right language for system-level modelling and building high performance system prototypes. It is the flow between languages which enables a high productivity and low risk design process, not an attempt to use a single language for all purposes.

4. New methods and flow possibilities

It is thus appropriate to remember that SystemC also has a ceiling. While it will likely evolve in its 3.0 version to include software task modelling and scheduling capabilities, it is not a software development environment.

Recent developments in the UML software modelling world on UML 2.0, to be finalised this year, promise greatly improved capabilities for system software modelling and code generation, especially for embedded real-time systems [2]. One very promising area for future methodology work is to establish a truly trilingual world, where software specified and modelled in UML can code-generate platform-optimised software tasks, which can then be simulated within a SystemC-based transaction-level platform model, with appropriate OS models. The hardware part of the design will move into Verilog-2005 or VHDL based implementation and verification, re-using the functional prototypes built earlier in the process. Peaceful co-existence is the goal for our truly multi-lingual world.

5. Language-based design research

There are considerable opportunities available to help users in this multi-lingual world through advancing the science of language-based design. Among the areas of greatest need are:

- Paths to implementation from high level design descriptions. For example, first generation behavioural synthesis from behavioural formats did not succeed for many reasons, primarily poor quality of results. SystemC and higher level abstractions offer modelling power, but leave the implementation primarily to manual refinement methods. The issue of high-quality, high-performance code generation ("system synthesis") from model to either HW or SW implementation on a specific target platform (standard cell, structured ASIC or FPGA for hardware; target-optimised C code for software) still has many research opportunities. The emerging field of co-processor synthesis may hold the most promise for advances here.
- The "models of computation" issue for system-level modelling is still open. How best to model all the constituent elements of a hybrid HW-SW system; how to build a system model by composing subsystem models; how to best verify the resulting system - these are still quite open questions. Building linked flows between system specification, platform models and hardware and software implementations requires semantically correct design transformations which could use more formalisation.
- Transaction-level modelling of HW-SW platforms in SystemC holds promise for true system model interoperability, to be a basis for at least one level of system synthesis, and to be a well-defined part of a modelling and refinement methodology, but there is a strong need within the OSCI and external frameworks to reflect not just industrial experience, but language-based design theory in developing the most appropriate context for this methodology.
- Verification methodologies have been going through great change in recent years with the emergence of HVLs and the guidance of methodology experts. But verification is not yet a science; the academic community has not spent enough time developing formal underpinnings for the complete verification problem - including directed semi-formal approaches using simulation, as well as formal methods.

5. Conclusions

This has been a very brief overview of the current issues and opportunities for building new and improved design and verification methodologies with SystemC and other associated languages. The prospects for improved language-based design approaches are encouraging.

10. References

[1] T. Grötker, S. Liao, G. Martin and S. Swan, *System Design with SystemC*, Kluwer Academic Publishers, Boston, 2002.

[2] L. Lavagno, G. Martin and B. Selic (editors), *UML for Real: Design of Embedded Real-Time Systems*, Kluwer Academic Publishers, Dordrecht, 2003.