Panel: The Future of System Design Languages

Chair: Richard Goering, EETimes
Organizers: Clifford E. Cummings, Richard Goering, Nanette Collins

Abstract

Verilog HDL was a breakthrough for the hardware design community in 1986. Over the years, the methodology based on the Verilog HDL has been extended with utilities and enhancements. With 0.25- and 0.18-µ processes enabling a system to be packed onto a single integrated circuit (IC), design problems have surfaced that no one could have predicted 13 years ago. As a result, several new design language proposals have been introduced since the last Design Automation Conference (DAC), all claiming to aid system-on-chip (SOC) design. Several claim to improve the designer’s ability to efficiently create, implement, and verify SOC designs from architectural specification through functional implementation.

The panel, comprised of experienced designers and representatives of organizations submitting design language proposals, will debate the various proposals and will try to identify what future trend will accelerate system design.

Questions and issues to be considered include:

- What’s in store for the future — C, Java, Superlog, HDL or SLDL?
- A comparison of modeling, gate-level and behavioral simulation capabilities of a new design language to current languages/tools/methods
- A comparison of the software development capabilities of a new design language to current languages/tools/methods

A review of the projected design environment of a new design language to current languages/tools/methods.

Position Statements

Clifford E. Cummings, Sunburst Design

I will mostly propose Verilog-evolution as the short-term answer to system design. I have chaired the behavioral task force on the Verilog Standards Group, which has proposed most of the exciting enhancements that are coming to Verilog 1999/2000. My basic position is that, with the exception of Superlog, nothing has captured my fancy with respect to SLDLs. My current take on SLDL is that, if as John Cooley has stated that VHDL is Verilog on steroids, SLDL is VHDL on steroids! To me, SLDL appears to be another cumbersome language-by-committee that is trying to make everybody happy (Verilog and VHDL interfaces) but will end up being dog-slow after it comes out of committee in another five years.

Steven E. Schulz, Texas Instruments

Today’s HDLs have served the electronics hardware industry well over the last 15 years. However, they are not sufficient for the next 15 years, in which complex architectural tradeoffs, more silicon concerns, and embedded analog, RF, and software will require a stronger semantic foundation than either HDL can provide alone. Similarly, software languages have also served us well, yet are not sufficient for complex hardware design needs. Finally, we should avoid adopting a third HDL syntax that offers little new capability in terms of verification scalability, system-level semantics, or handling of complex system-wide design constraints. SLDL was founded by system and silicon suppliers worldwide, and is the result of years of study and research by noted experts in the field. Rather than compete on “dueling syntaxes,” we are best served by directly addressing the challenges of integrating multiple semantic styles on a single piece of silicon. SLDL will offer the industry a backplane that compliant language syntaxes can use to communicate and integrate multiple views of the system, including our C-based and HDL-based implementations.

Simon Davidman, Co-Design Automation

It is clear to most people in the electronics design community that a new design language is needed, particularly given the increase in SOC designs and the complexity of modern electronic systems. What is unclear is the direction that should be taken. Our opinion is that we must make sure we meet the requirements of the new design paradigm, rather than rushing to adopt a current standard which looks like it might meet some of the needs. Requirements involve unifying the disparate parts of the design flow, as well as interacting teams, and making the flow more efficient, not only in terms of the tools, but also the ability to create designs and use the language. The language must also allow an evolutionary approach from existing methodologies, which also means that the language and accompanying tools must be open to other design description techniques.

John Sanguinetti, CynApps

I am a proponent of C++ as the natural next step in hardware description languages. C++ is sufficiently close to C to leverage the enormous body of existing IP in C, along with the large body of knowledge about software engineering developed around C and C++. It has been demonstrated that hardware can be described at
the current level of abstraction with standard C++, through class libraries. This gives a natural, proven way of designing hardware at higher levels of abstraction, just like software has been developed. The use of general purpose languages, like C++, for hardware design is continuing the long process of making hardware design indistinguishable from software design. We are simply closing the gap between good software engineering practices and hardware engineering practices.

**Joachim Kunkel, Synopsys**

The goal of any System Design environment should be three-fold. 1) It should enable the user to model design functionality without regard to whether it will eventually be implemented in hardware or software; 2) it should significantly shorten the design cycle by allowing the designer to create and test an executable specification at a higher level of abstraction, and 3) it should be an open standard to facilitate the free and unencumbered exchange of IP between companies. We feel that the best approach to unify these design goals is to use the existing language of system architects and software designers (C/C++), and expand it through the use of class libraries to enable it to also capture RTL designs. With SystemC and the Open SystemC Initiative, we feel that we have created a standard that enables these design goals with the most efficient and complete solution available today, and the tremendous adoption rate of SystemC validates this.

**Oz Levia, Improv**

I will propose a language independent semantic model that will support multiple implementations with different languages (Java, C++, others). The goal of this approach is to allow specification of an IC system that is destined for SoC IC in a specifications that can be mapped into multiple physical styles (SW, SW and HW or HW) by multiple tools. The focus on the semantics model is needed if we want to avoid dialects, subsets and extensions to existing languages. The semantic model is the backbone of every design, simulation, specifications or implementation language. By agreeing on common semantics (what things mean) we free designers to use multiple tools and languages, and we free tool companies to innovate where innovation is needed - in design transformation and verification, not in creating a new dialect of a known language.

The Directed Control Dataflow Network (DCDN) semantic model developed by Improv, is such a semantic model. It has been adopted by OVI (pending) and is evaluated by VSI for standardization. This semantic model is truly implementation independent in so far that it does not assume any specific mapping for design realization in HW. Additionally no assumptions are made regarding the language syntax used to express the design. This semantic model supports specifications of reactive synchronous and asynchronous systems, with explicit specifications of control flow and data flow. Using common concepts such as interface and hierarchy allow for modularity and encapsulation to support reuse and verification.