Building a Common ESL Design and Verification Methodology – Is it Just a Dream?

Organizer:
Francine Bacchini
Francine Bacchini, Inc., San Jose, CA
+1.408.839.8153
francine.b@comcast.net

Chair:
Gary Smith
Gartner-Dataquest, San Jose, CA
+1.408.468.8000
gary.smith@gartner.com

Anoosh Hosseini
Cisco Systems
Milpitas, CA

Ashish Parikh
Pixelworks, Inc.
Campbell, CA

H. Tony Chin,
HD Lab, Inc.
Shin-Yokohama, Japan

P. Urard
ST Microelectronics
Cedex, France

Emil Girczyc
Summit Design, Inc.
Los Altos, CA

Simon Bloch
Mentor Graphics Corp.
San Jose, CA

Categories and Subject Descriptors
B.7.2 Hardware, INTEGRATED CIRCUITS, Design Aids

C.3 SPECIAL-PURPOSE AND APPLICATION-BASED SYSTEMS

General Terms: Algorithms, Design, Languages Standardization, Verification

Keywords: C/C++, Design, ESL, Methodology, Modeling Rapid Hardware Prototyping, RTL, SystemC, SystemVerilog, Verification

PANEL SUMMARY

In recent years, industry cooperation has established common language and methodology standards to support electronic system level (ESL) modeling, design and verification: SystemC, SystemC Verification (SCV) and SystemC Transaction Level Modeling (TLM). What is still conspicuously absent, is a common, standard methodology for ESL design and verification itself. As a result, leading ESL adopters have been forced to devise their own custom methodologies. Does everyone need to develop their own custom 'vertical' methodology using standard 'horizontal' languages, libraries, and data formats, or is it possible to devise a common, standard methodology – open to all – that would help to *mainstream* ESL design and verification?

This panel of ESL users and suppliers will take a close look at what is being done today and debate the issues around what more is needed to address increasing system complexity. What are the major attributes and requirements for a standard methodology? How would it deliver the diverse requirements of algorithm development, system architecture exploration and optimization, integration and re-use of intellectual property (IP) from diverse sources, processor and coprocessor development, software development, RTL design, and testbench generation for system-to-implementation verification?

DAC 2006, July 24 28, 2006, San Francisco, California, USA.
ACM 1-59593-381-6/06/0007.

Panelists will discuss these and other issues to illuminate some of the steps toward a standard ESL design and verification methodology.

PANELISTS' VIEWPOINTS

Roberto Guizzetti: "It is well known that technology is progressing at a rate of doubling capacity every 2 years. Design productivity is struggling to keep the same pace. New design methodologies and tools are mandatory to fill the gap. Today's technologies allow as much as 800k gates in just a square mm. With present design productivity, it would require 4 man years of effort to fill a square mm - something not economically viable. IP reuse of high-level models (C/C++/SystemC/SystemVerilog) is one of the major contributors to the required design productivity improvement. And, obviously, High-Level Synthesis is the missing link in the chain, in order to allow such high-level models to be easily, quickly and deterministically translated to lower levels of abstractions. Another aspect to consider is whether the tools and methodologies will be adopted by the system-level and more SW-oriented engineers. Present tools, and synthesizable subsets of various languages, are more easily adopted by hardware-oriented designers, since they are obviously languages to describe hardware. Will the two worlds converge? For sure, high-level model adoption would greatly facilitate exchange of information between system and implementation engineers. One of the most important factors that could contribute to HLS becoming a mainstream approach in IC design is the existence of a common language and synthesizable subset of this language. Presently, the more promising language is SystemC. It encompasses features such as concurrency and H/W oriented datatypes, which are well-suited to model H/W blocks. At the same time, it is based on the standard C/C++ languages used by most software engineers. But, again, a common synthesizable subset adopted by all the major CAD vendors will be a winning point in the widespread adoption of such a methodology. OSCI should work towards progress in such a direction. Will this be flexible enough to be used in different application domains? I'll tend to say yes to such a statement."

Anoosh Hosseini: "We see the benefits of ESL when we are able to build a platform which starts out providing architectural

exploration and performance estimation, and transforms into the embedded software development platform, followed by co-simulation with higher level management software. Parts of the platform can also be leveraged in the DV and RTL simulation cycle. In order for any methodology and associated tools to be adopted, they need to have significant and measurable impact on time to market and quality metrics of the end product. One example is enabling software development far in advance of hardware availability. The longer the hardware design cycle, the more value the simulation has. On paper there may be many equally viable ESL methodologies, however many times it is more related to how the engineering team is organized, the roles architecture, design, DV, and software teams play, and resource allocation among them. The characteristics of a good ESL methodology are efficient and rapid prototyping of hardware, followed by ease of integration with other simulation environments. In my discussion I will discuss the various things that facilitate this process. In our line of business we have more proprietary IP versus integration of standard or third party components. Assuming the latter is our target, common conventions applied across the full SOC design will have been standardized and adopted by IP providers in order to facilitate an eco-system. This includes module interfacing, tracing, logging, register specification, configuration management, etc. The pace of development in these areas has not matched other consortiums where companies make regular and large contributions to the community and all abide by common standards."

Emil Girczyc: "There is a common oversimplification that views ESL simply as a language, such as C or SystemC. ESL is methodology, not merely the next level of abstraction above RTL, therefore, it must evolve as methodology. SystemC has the unique benefit of being a standard that was developed prior to the existence of a de facto industry standard, such as was the case with Verilog. This has caused SystemC to evolve differently than the HDL language standards, aided with the existence of the OSCI reference simulator. Now we have powerful features such as Transaction-level modeling (TLM) which are consistent across multiple IP vendors and EDA tools. As with the evolution of any standard, the ESL design and verification methodology reflect the problem they need to solve. The current ESL methodologies have evolved out of the C-modeling and behavioral synthesis spaces. They are being extended to address additional needs of system-level design, such as architectural design, software development and IP exchange and reuse. A common (or at least consistent) standards-based flow effectively enables design teams to execute true system design, which includes hardware, software, and IP from multiple sources, to significantly reduce the time to market pressures inherent in today's designs. In order to realize the full benefit of ESL, there is still work to be done. Traditional tool and IP suppliers need to cooperate to deliver open models that work in multiple environments and fully address the need for hardware and software modeling in an ESL flow."

Simon Bloch: "Of the three design domains (algorithmic, bus-centric, and control), the algorithmic domain is the most mature in terms of ESL adoption. In this space, ESL tools are being adopted based on the proven ability to provide a 10X to 100X productivity improvement. In the other domains, ESL point tools are emerging for initial application areas of modeling, simulation and synthesis. In general, ESL design needs to start at the highest level of abstraction, with ANSI C++ as the preferred design language, and SystemC the preferred verification language. Uniting C++ for design and SystemC for verification across all three domains represents the next big step in ESL development. The creation of a full-fledged ESL flow will take time. The RTL flow has been evolving for the last 20 years, so the industry needs to understand that a compete ESL flow will take time."

Ashish Parikh: "We are looking at ESL pragmatically, trying to take a practical approach to developing a design and verification methodology around emerging standards and tools. ESL is still in its infancy; although shades of it have been around for many years, industry's demand and adoption of ESL has become critical only in recent years. As the complexity of SoCs and increasing mask costs drives the evolution of ESL tools, infrastructure and methodology, there is still a long road ahead before the industry will find real convergence and wide adoption. Pixelworks is an exemplar of a huge market presently ill-served by EDA companies; today third-party tools only solve a small percentage of our problem in system design. In the meantime, companies like us are forced to use the few existing commercial tools and must develop the internal infrastructure and supporting tools to complete our ESL design flows and methodologies. As ESL continues to absctract above the RTL level, the areas of behavioral synthesis and equivalence lack robust solutions and require fresh new ideas. Verification, which has been focused around RTL, gate level, and FPGA emulation, must shift to higher level functional simulations. Without this shift in conjunction with equivalence and proper ESL synthesis methodologies, ESL can not mature. The requirement for cross-discipline training of the engineering work force is also a requirement. A HW engineer and SW engineer can no longer be separate entities. With the increasing complexity and sheer size of SoC subsystems increasing, leveraging reuse in all areas of the design is key to success."

H. Tony Chin: "At the present time, we finally have standards for ESL design and verification languages. These standards are SystemC and SystemVerilog. However, design and verification sources that are written in these new ESL-based languages are not easily reusable. For example, EDA tool vendors have yet to open specifications for their bus models. Because of this situation, models written specifically for a particular bus transaction model cannot be connected to another bus model. Further, in behavioral synthesis, directives need to be written into the source to clarify design intent. Currently there is no open standard for these synthesis directives. Therefore, directives are based on a particular behavioral synthesis tool being used. In shifting from gate-level to RTL design methodology, it was enough to standardize RTL language subsets. This is because registers are fixed in both gate-level views and RTL views, meaning that clock cycle latencies do not affect the design model itself. This in turn allowed for HDLs to successfully describe hardwares' parallel behavioral aspects. On the other hand, in the ESL world, register structures and clock cycle latencies are flexible and change depending on various design requirements and intent. This situation requires further improvement of the ESL language itself as well as the establishment of useable methodologies. At the moment, we see the language, methodology and design techniques are in a state of evolution."