ESL: Tales from the Trenches

Organizer:
Francine Bacchini
Thinkbold Corporate Communications, San Jose, CA
+1-408-267-6602
francine@thinkbold.com

Chair:
David Maliniak
Electronic Design, Paramus, NJ
+1.201.845.2434
dmaliniak@penton.com

Terry Doherty
Emulex
Bothell, WA

Suhas A. Pai
Qualcomm
San Diego, CA

Dr. Soo-Kwan Eo
ST Microelectronics
SEOUL, Korea

Peter McShane
Northrop Grumman
Redondo Beach, California

Sriram Sundararajan
Texas Instruments Inc.
Dallas, TX

Pascal Urard
ST Microelectronics
The Netherlands

PANEL SUMMARY

Electronic System-Level design has arrived - but can ESL provide the bridge from systems to silicon? Comprised of real world designers, this DAC ESL panel will examine and debate what works, what doesn't, and what the gaps are in the methodology and tool offerings. Panelists from a variety of industry segments, including Military/aerospace, storage area networks (SAN), wireless communications and consumer electronics, will share their experiences, lessons learned and further needs.

Does ESL bridge the gap between systems to silicon? Hear from designers about their real world experience with ESL. What worked according to expectations? What didn't? What are the gaps in the methodology and tool offerings that need to be filled, and why?

This panel of ESL design methodology users will give us a "reality check" that will enable potential users to make an adoption decision, and enable ESL design tool suppliers to evaluate their product strategies against "big picture" requirements.

Panelists will address primary areas of concern, such as:

Methodology Usage: What do you use ESL design for? Is it for algorithm development alone? Are you using it for hardware/software partitioning? Have you used it for embedded system architecture development for performance optimization and/or for SoC platform development? Are you using ESL for embedded software development, using the system architecture model as a development platform? Are you doing any high-level synthesis of RTL? Did ESL help you with your system testbench development or HW/SW co-verification?

Industry-Level Initiatives: How has language standardization, such as SystemC, impacted your design efforts? What other industry-level initiatives would be of use - standard TLM methodology, or other?

Tools: Do you use commercial tools or open source software? What was your selection criteria? Do you use domain-specific tools for algorithm development and implementation? Did you develop your own tools - if so, why? Do your proprietary tools have specific attributes that you think can be incorporated into commercial tools? How do ESL tools compare with your original expectations?

ROI: What was your overall payback in terms of time, effort and money consumption, re-usability, risk management and overall success? How do these compare with your original expectations?

DAC 2005, June 13–17, 2005, Anaheim, California, USA.
ACM 1-59593-058-2/05/0006.

Categories and Subject Descriptors

C.3 Special purpose and application-based systems
C.4 Performance of systems
I.6 Simulation and modeling
J.7 Computers in other systems

General Terms: Algorithms

Keywords: Electronic system-level design

1. PANELISTS VIEWPOINTS

Terry Doherty, Emulex

Emulex develops and sells Fibre Channel Host Bus Adapters and Switches for Storage Area Networks. At Emulex we develop Transaction Level Models with SystemC for architecture exploration, algorithm development and large configuration evaluation. Solving traffic congestion problems is a primary concern for large SANs. ESL has helped Emulex develop proprietary traffic shaping algorithms that will allow customers to grow their storage networks to unprecedented levels. In addition, ESL allows Emulex to simulate and characterize large configurations which are cost prohibitive to build in a lab environment. System Architect from Summit Design, along with home grown tools, have been instrumental in evaluating and debugging models. However, further commercial tools are needed to ease debugging.

Peter McShane, Northrop Grumman

NGC is using ESL for performance analysis of a weather satellite. ESL models are used to study effects of changing parameters on system performance. We run multiple simulations

(on the order of thousands) to see patterns and trends of performance as parameters are varied. We need to understand the sensitivity of system to parameter variations. We use the models to communicate with subcontractors the system architecture and how each subsystem relates to the overall system. The economic reward - much easier and faster system-level design process, reduced system integration problems, as well as reduced project delays.

Suhas A. Pai., Qualcomm
ESL is mostly a language and methodology-based approach. The language offers features and the methodology defines the way that the features will be used in a standardized way. Our conclusion was that the standardization is far more important to our ESL design strategy because it is directly tied to the productivity. We use the Open SystemC Initiative's SystemC library for architecture characterization, SystemC-wrapped processor models in RTL and software verification, SCV for TL co-emulation, and TLM lib to improve our modeling style. Unless ESL tools lead to a cookie-cutter approach in using the language, we won't see the impact on design cycles we had hoped to see. This is also true for the virtual software platforms where standardization in debug, monitoring/profiling, and API integration calls for higher priority.

Sriram Sundararajan, Texas Instruments Inc.
Bridging the gap between system and silicon has, of course, been the holy grail of EDA. The successful strides that have been made in ESL have been in the areas of modeling, performance analysis, and co-simulation. The areas that have floundered are behavioral synthesis, synthesis from pure algorithmic descriptions in languages such as C, and automatic approaches to system partitioning. The reasons for this are varied and complex: An optimal algorithm/application to architecture mapping often requires intimate knowledge of the application; designers use an intuitive sense of what functions need to be kept flexible and hence must be mapped to software; legacy intellectual property also plays a part in driving system partitioning. All these issues make automatic architecture synthesis well nigh impossible. Where ESL can help is to provide a designer some tools as the designer wades through the maze of possible choices, and help him/her identify pitfalls and hotspots. In my group at TI, we are responsible for translating high-level standards documents as well as customer specified performance requirements into cellular base station SOC architectures. We have used multiple tools for performance analysis ranging from Matlab/SPW for algorithm simulations to SystemC (open source) based Transaction Level Modeling to get a feel for data flows in the system and to identify potential bottlenecks. We have also used co-simulation (Mentor Seamless) to validate our chosen HW/SW partition, ascertain performance, and test out interfaces. Thus our approach has been to mix and match commercial tools with a home grown ESL design/verification methodology.

Dr. Soo-Kwan Eo, Samsung
I believe that ESL bridges the economic gap between systems and silicon. We were able to set-up a Virtual Platform design technology in the last two years and have been applying it to various applications, including pre-silicon eSW design and debugging, optimum architecture exploration, performance analysis, and cost estimation for HW/SW partitioning. However, there are several gaps in today's ESL design methodology. TLM sign-off, behavioral synthesis, formal verification of the system specification, and top-down/bottom-up design constraints annotations are a few examples. At the same time, simulation speed, equivalence checking among behavioral C, TLM and RTL, power estimation and optimization, and software design are the gaps in the ESL tool. On the other hand, the huge modeling effort with expensive tool cost causes the low ROI, heightening the threshold of ESL methodology-in.

Pascal Urard, ST Microelectronics
Five years ago, we developed an in-house Matlab-to-RTL flow based on generic IP libraries, to bridge the gap between algorithm and RTL. This flow included some formal proof techniques (theorem proving) to guarantee equivalence between generic views. Despite the fact that this flow is not automated, it has been used to produce high complexity chips and is still used currently (ISSCC'04-23.4, ISSCC'05-24.3). We started to drive EDA vendors in 2001 to have commercial tools doing High Level Synthesis (HLS) for datapath oriented ASICs starting from a C-based language (we defined a subset of C/C++ with only a few parts of SystemC), in the way we want. We produced silicon with an HLS tool last year. We now work to enhance this flow with formal techniques (verification flow). The benefit is real for area/power by extensive architecture explorations.

2. ESL: DELIVERING THE BRIDGE FOR SYSTEMS TO SILICON

While the press and analysts have dedicated much publication space and conference discussion time to ESL, it's the tales of real world users that will help to reveal whether ESL will be able to deliver the promised bridge for systems to silicon.

Having used ESL modeling, design & verification on actual designs now in working silicon, our panelists of successful real world users of ESL methodologies, tools and technologies, will share their experiences, discuss their flows, review the industry initiatives, examine their productivity gains, and define their missing pieces and future needs from EDA.

Join us to hear how panelists from six different companies are finding the necessary solutions to their engineering challenges by operating above the register transfer level, using ESL to build the bridge from Systems to Silicon.