MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

KHARKOV NATIONAL UNIVERSITY OF RADIOELECTRONICS

ISBN 966-659-113-8

Proceedings of IEEE East-West Design & Test Workshop (EWDTW'06)

Sochi, Russia, September 15 – 19, 2006

CONTENTS

System Level Methodology for Functional Verification Soc

Alexander Adamov, Sergey Zaychenko, Yaroslav Myroshnychenko, Olga Lukashenko
Design Automation Department, Kharkov National University of Radio Electronics,
Lenin ave, 14, Kharkov, 61166 Ukraine
E-mail: adlex@onet.com.ua

Abstract

Building a verification environment and the associated tests is a highly time-consuming process. Most project reports indicate that between 40% and 70% of the entire effort of a project is spent on verification, with 70% being much closer to the normal level for successful projects [1]. This high level of effort indicates that the potential gains to be made with successful re-use are significant. Most projects do not start with a complete set of hardware designs available for a functional verification. Usually a design comes together as smaller blocks. Then the blocks are integrated into larger blocks, which may eventually be integrated into a system. That is reason for performing functional verification at a system level. The paper describes the system-level modeling environment for a functional verification System-on-a-Chip models. System level allow design teams to rapidly create large system-on-a-chip designs (SOCs) by integrating premade blocks that do not require any design work or verification.

Keywords: *device simulation, verification, system level modeling, System-on-a-Chip.*

1. Introduction

One of the hottest topics in embedded system design today is Electronic System Level (ESL) design. Although the idea of being able to describe a system at an abstract level has been around for a decade, only now are various parts of the design flow becoming available to make it practical. ESL describes a System-on-chip (SoC) design in an abstract enough and fast enough way to explore the design space and provide virtual prototypes for hardware and software implementation. It is becoming a fundamental part of the design flow because we can now use it throughout the iterative design process rather than just in the early system architecting phase.

ESL provides tools and methodologies that let designers describe and analyze chips on a high level of abstraction, easing the pain of designing electronic systems which would otherwise be too costly, complex or time consuming to create.

The adoption of ESL can be seen in the same light as the transition to register transfer level (RTL) methodologies 10-15 years ago when complexity and time-to-market pressures obliged the industry to step up to another design level.

As designs become larger with more and more IP blocks, engineers will re-use more IP. ESL methodologies that enable platform-based design will be increasingly necessary to create and test a complete system.

For the most complex SoCs, IP reuse can only help up to a point. For a 40-million-gate SoC, filling even 75% of the device with existing IP leaves 10 million gates to design with original content. ESL methodologies which allow rapid creation of new blocks are likely to be leveraged by designers to quickly develop and verify original content to fill the 10 million gate void while meeting time-to- market requirements.

Among the 24% percent of respondents who have implemented some form of ESL design methodology an overwhelming 87% believe ESL provides an acceptable or greater return on investment (Figure 1).

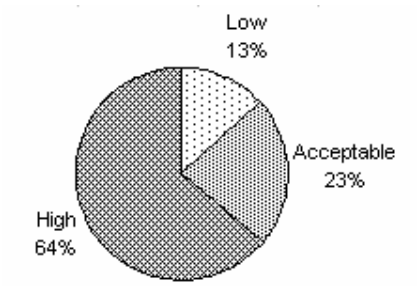

Figure 1. Return on investment for implementing ESL

This is an important statistic as it illustrates that ESL is succeeding in early deployments. Early success fosters positive references which can be a critical factor in mainstream designers' decision to investigate a new methodology [2].

Currently system level design is not well supported in industry [4]. This hinders verification and validation

of system properties and functionality. These activities should begin as early as possible in the development process.

Goal of this paper is to describe effectiveness of the system-level simulation on the base of available digital simulation system, development of the functional prototype of the verification environment software. The *main research topics* include:

- benefits of using system description at high level of abstraction;
- architectural and functional features for building system-level verification environment within experimental Integrated Verification System.

2. ESL Methodology Overview

ESL design is being enabled in three ways:

- constraints and parameters from the high level of abstraction can now pass down into different implementation tools;
- a tighter linking of system designs into the early verification process better justifies the engineering investment;
- bit-true and cycle-accurate versions of execution platforms early in the design cycle are enabling embedded software development and integration to start earlier.

The growth of the ESL is based on emerging standards for passing data and design constructs through the development process between different design tools. Early tools that tried to tackle ESL design imposed new, and often proprietary, design flows, languages and coding styles on the designer, and that was the major reason these tools failed to take hold in the late 1990s. Another reason was that the design world lacked the libraries of blocks of intellectual property (IP) and the models of functions that could be used to construct these systems.

Providing virtual prototypes means that software development can start much earlier. With the models of the hardware available, embedded software can be written and tested by multiple software engineering groups in parallel, and tested against a standard set of criteria on a virtual prototype. This saves time in the development and makes the hardware/software co-verification process a lot simpler.

Getting an early functional representation of the SoC is a key advantage to adopters of the ESL design process and it mirrors the way Register Transfer Level (RTL) design was originally adopted by designers in the 1980s.

Today, the ESL tools are being proven in the same way, being used initially for modeling, verification and validation, and then moving out into other parts of the design flow where the automatic generation of hardware and software speeds up the design process. Block-assembly and configuration data flow from high level SystemC descriptions to RTL and then to the gate level has now been demonstrated, and standards for broad-based adoption of these techniques are emerging.

As more and more of the system is configurable, such as different processor cores, different bus widths, different memory sizes and configuration, different programmable engines, and many, many different mixes of peripherals, the ability to simulate the different options early on in the design is extremely valuable. This can be used to develop a platform of devices, ensuring that they will meet the requirements of the applications and allowing the ever increasing cost of custom designs – both in the development and mask costs - to be spread across a wider range of devices and customers.

There are three distinct approaches to ESL, and some of the tool vendors today are using the term in ways that confuse the three. At the highest level is the Algorithmic approach, while at the design and implementation level there is the Architectural exploration, and thirdly there is the Automatic generation phase.

Simulation is the key at this level of design abstraction. SystemC models provide the ability to simulate the functions of the system thousands or millions of times faster than at the RTL level, but the trade-off is the timing accuracy.

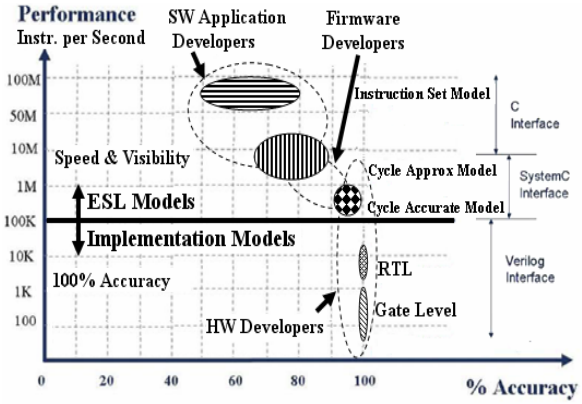

Figure 2. Where high level ESL models can be used

For example, functionally accurate (bit-true) models that are un-timed or cycle-count accurate can run from 10MIPS to multiple-100MIPS in system simulation, while the cycle accurate versions run in the 100KMIPS to megahertz range (Figure 2). These "benchmark simulations" are true to RTL timings with only a few noted timing exceptions and these can be used to check system-behavior down to the clock-cycle level.

The key to linking ESL design to the hardware and software tool chains is the definition of transaction level communication between abstract models and hardware implementation. A "transaction" is a collection of detailed signal data, such as a full memory read or write, which in a hardware simulation is spread over a number of specific input /output signals and clock-periods. These transactions can be used whether the block is SystemC, C code or RTL. To interface models to hardware simulations, transactions are decoded by a transactor, which transforms the transactions into signal values for hardware blocks, behaving like a data level translator with a buffering capability [3].

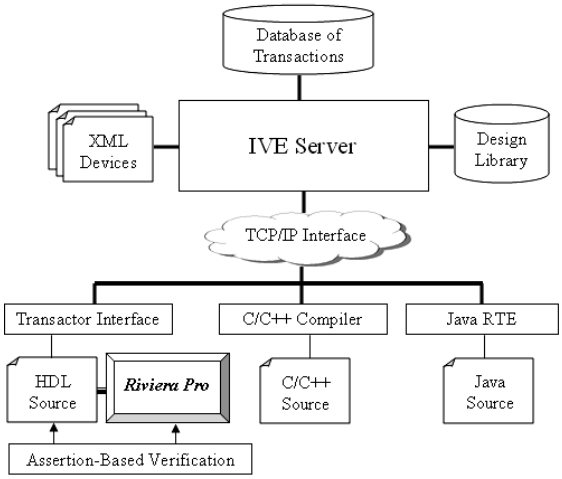

Figure 3. Verification Environment Architecture

3. Experimental Integrated Verification System

Heart of the verification environment is the server software, which controls the dataflow in the system. Devices are entered in the system by means of loading their interfaces in XML format. At the same time in a project library the prototype of the system component is created. Adding these devices we connect them with the corresponding prototype.The communication environment used for data transferring is TCP/IP network protocol. Handing on of packages is carried out with the help of Windows Sockets 2 mechanism. The information about simulation is saved within a transaction's database. Subsequently this information can be used for a static verification and revealing of the observed parameters violation.

Using the network interface given by specially developed communication library client models can be hooked up. How we can see from the picture it can be the HDL models connected with the help of transactors and simulated within hardware simulator. Also we can use prototypes and the testbenches written in high-level languages, such as C/C++, Java.

3.2 Functionality Overview

Despite of experimental character of the developed program system, implementation allows to present perspective functionalities of valuable implementation. The basic functions are:

- entering of SoC components in the simulation system, described in XML format;
- connecting simulated devices by means of network TCP/IP protocol;
- co-simulation of components at a different levels of abstraction.

As was mentioned before with help of Integrated Verification System we can simulate components described as RTL models. Figure 4 represent dataflow in case of RTL components simulation. For system-level modeling we need to transform data from RTL signals to more abstract implementation level. We can do that by means of involving transactor's layer, which allows us to make such conversion. Transformed data sends throw the Server of the system.

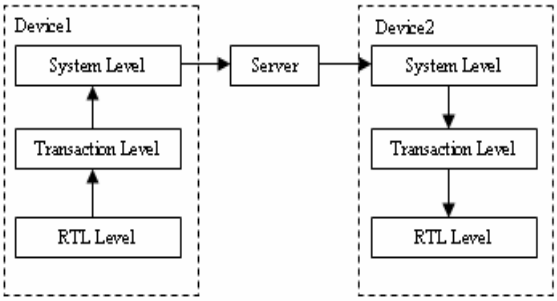

Figure 4. System Objects Interconnection

For the purpose of data transmitting within network environment we need to append additional information about transaction for packets' transmission (Figure 5). This information includes recipient address, instruction code and the data itself. After receiving the data corresponding signal values for hardware blocks will be established inside of transactor's method and then they will be delivered to HDL simulator.

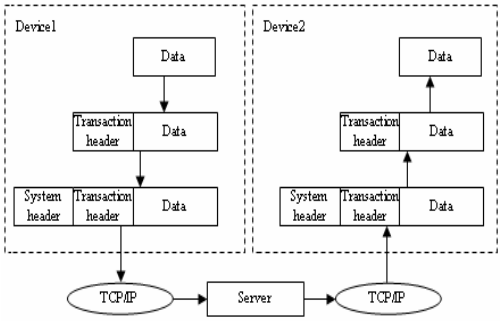

Figure 5. Data forwarding throw the system

The designed system allows:

- engineering the virtual prototype of the system;
- early software validation of the system;
- performance analysis of the system;
- transaction-level verification of the project;
- architectural analysis of SoC;
- power analysis of a chip;
- searching hardware/software tradeoffs.

Thus the system enables to speed up the process of simulation going up to higher level of abstraction.

3.3 Graphical User Interface

Figure 6 shows a graphical user interface of the system. On the background we can see Riviera Verification Solution, which is integrated to simulation environment. The top application is a server part, which contains three panels. In the left upper panel we can see the list of loaded devices with a pointing of their interfaces. In the right panel there are graphics representations of the components, and also their current state is shown. Below there is a console that print out information about modeling and the transmitted data, which can be saved to the appropriate log-file.

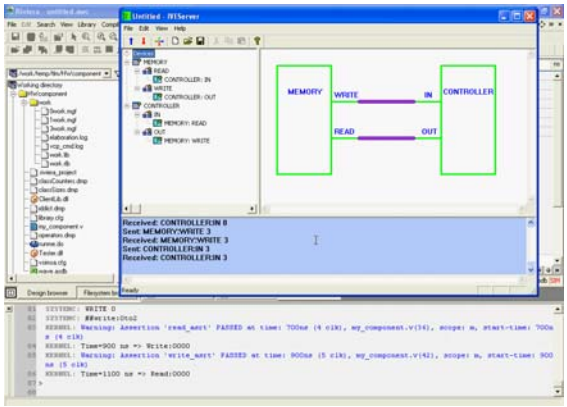

Figure 6. Graphical User Interface

4. Conclusion

The nowadays trends of using system-level simulation for SoC models was presented in this paper. The prototype of the mixed-level verification environment with support of TCP/IP communication interface, transaction-level modeling and assertion-based verification on the base of existing digital simulation tool – Aldec Riviera has been developed.

The following tasks have been solved: 1) Defined basic mixed-level simulation software architecture; 2) Suggested new technique for interconnection of SoC components; 3) Considered possibility of implementing ESL methodology for functional verification SoC models; 4) Showed potential of using different kinds of verification within SoC model; 5) Implemented results of the research within experimental software system.

Using ESL methodology in design and verification tasks that traditionally took weeks or months can now be done in minutes. It will provide a reliable way to get tens of millions of gates and hundreds of thousands of lines of embedded code written and validated quickly to meet the ever increasing customer demand. Tools in this area come from companies such as CoWare with ConvergenSC for Architectural ESL, Synopsys with its CoCentric System Studio, Prosilog's Magillem and ARM's RealView MaxSim. All these target the early design exploration, which indicates a maturing of the market for ESL tools. For instance, Samsung has reported savings of approximately 40% in design time through the use of ESL tools [3].

A large Japanese printer company adopted ESL design methodology because its RTL-based methodology could no longer cope with the architectural modifications required by each generation of printers. The company uses the same basic algorithms for its whole product range, from low-end home printers to high-end network printers. However, major variations in data communications, processing and storage requirements between the different printer types mandate different implementations of those algorithms, including different memory and communications bus architectures [5]. Qualcomm Inc.'s experience demonstrates improved HW/SW co-verification at the system level over that at the C/RTL implementation level. A Viterbi decoder design executed a packet in 20ms, but took 6hrs to simulate at C/RTL level. Qualcomm estimates that 1,000 packets must be simulated to achieve a reasonable confidence level, but considers the necessary 6,000 hours of simulation time to be impractical. Co-verification of 1,000 packets with a ESL would have taken 6 hours or less [5].

Further research will be done in the area of developing methods for simulation results analyzing with help of statistical and data mining approaches.

5. References

[1] *Andreas S. Meyer,* "Principles of functional verification". Elsevier Science, 2004, 206 p.
[2] Mentor Graphics, "ESL Survey Report", December 2005.
[3] Chris Lennard, Davorin Mista, "Taking Design to the System Level", April 2005.
[4] P.H.A. van der Putten, J.P.M. Voeten, M.C.W. Geilen and M.P.J. Stevens, "System Level Design Methodology".
[5] Seyul Choe VP and General Manager Asia-Pacific CoWare Inc., "ESL enables software-driven SoCs".

AUTHORS INDEX