AN INTERACTIVE SYSTEMC COURSE FEATURING REAL-TIME ONLINE COMPILING AND ANALYSIS

Oliver A. Pfänder
Christophe Layer
Wolfgang Schlecker
Hans-Jörg Pfleiderer
Department of Microelectronics, University of Ulm,
Albert-Einstein-Allee 43, D-89081 Ulm, Germany
oliver.pfaender@uni-ulm.de

Abstract: An interactive tutorial is a very efficient way to teach a rich programming language to a large group of e.g. graduate students in the field of IC design. We are creating a novel online course for SystemC that provides real-time compiling and result analysis functions, using server-side tools exclusively. The tutorial is called SCOTT (SystemC Online Tutorial and Training) and represents an advancement of our previously established interactive course on VHDL. The users following the SCOTT course submit their SystemC code over a standard web browser and get instant feedback from the tutorial system, including error description, context-sensitive hints, proposals for improvement, and simulation results. In this contribution, we will point out the emerging importance of teaching SystemC, discuss the functionality of our interactive tutorial system and illustrate the methodology by practical examples.

Keywords: Interactive Tutorial, SystemC, Teaching, Compiling and Analysis

1. INTRODUCTION

Especially with the advent of embedded microprocessors and the ongoing node reduction in transistor technology, the design of electronic systems has become more and more challenging. Also, the importance of coherently simulating complete systems is constantly growing in modern circuit design tasks. The ever-increasing complexity of today's integrated circuits and the decreasing time-to-market have driven EDA (Electronic Design Automation) companies to provide tools supporting both the design and verification of electronic systems. Heterogeneous programming environments and multiple languages – e.g., HDL vs. C/C++ – are used for co-design, co-simulation and

co-verification [1]. As a result, SystemC was born to ease the integration of heterogenous components onto System-on-Chip (SoC) and to help bridge the verification gap between software developers and hardware designers.

Responding to the industrial demand for IC designers with experience in SystemC, graduate students and professionals need to gain SystemC programming skills in an efficient way. Thus, we are developing an interactive tutorial which will allow them to learn this emerging language in detail over the Internet, while going through the course at their own rhythm, with no other need than a standard web browser.

This paper is organized as follows: Section 2 points out the rising importance of SystemC, section 3 presents our approach of teaching this language, section 4 describes the technical realization of our interactive tutorial, and section 5 sums up important results and conclusions.

2. SYSTEMC'S GROWING IMPORTANCE IN SOC DESIGN

The emerging concept of SystemC [2] intends to connect software system design on a high level of abstraction to the hardware development at various levels of detail. It provides a complete and consistent approach for designing a system from the specification down to the implementation, accounting for different domains (hardware, software, analog, digital) and miscellaneous system components such as DSPs (Digital Signal Processors) or FPGAs (Field Programmable Gate Arrays). Moreover, the exchange of system-level IP (Intellectual Property) models for creating executable specifications has become a key element for an efficient SoC design flow.

SystemC is based on the computer software language ANSI C++, amended by means to describe the following attributes: [3], [4] and [5]

- Concurrency – hardware is inherently parallel
- Reactivity – for synchronizing concurrent events
- Communication means to exchange data
- Hierarchical structures and hardware-oriented data types
- A concept of time

Systems can be modeled at various levels of abstraction, from the C++ algorithmic model down to RTL (Register Transfer Level). As seen in Figure 1, SystemC provides a single modeling framework to cover a wide range of computational models. SystemC is a stable and interoperable development platform, providing exchange and creation of fast C++ models. It is used to evaluate system performance, explore architectural decisions and to assess the correctness of an implementation via system modeling. To use SystemC, one only needs a standard C++ compiler and the SystemC libraries. The executable specifications resulting from the compilation of the hierarchical design can be stimulated by a test-bench at the top-level, also controlling the simulation time.

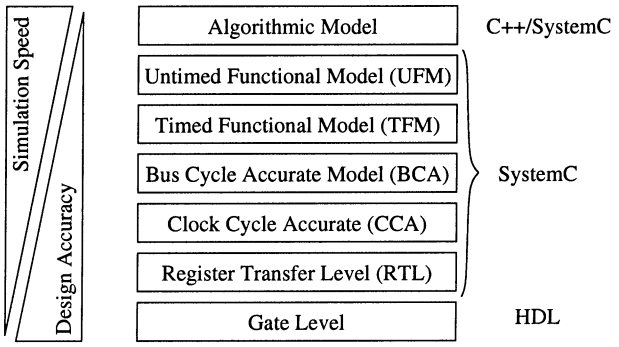

Figure 1. Refinement steps over the abstraction levels in SystemC follow the usual top-down design flow.

Similar to all simulation based approaches, the designer needs to design this test-bench by hand. A system model is usually described and implemented in several SystemC source files. These files then get compiled and linked together with the SystemC libraries by a standard C++ compiler to form the simulation executable. The synthesis capability of SystemC can significantly improve the designer's productivity since synthesis tools are commercially available both for behavioral and RTL.

The OSCI (Open SystemC Initiative) [2] is an independent and non-commercial organization consisting of numerous hardware makers, software companies, independent developers and research institutes. The purpose of this pooling is the support of SystemC and its consolidation as an open standard, having the official IEEE standardization as one of the main objectives. The official SystemC Internet home page [2] is the central portal for both contributors and interested party and offers a wealth of continuative information.

3. EFFICIENT SYSTEMC TEACHING

With SystemC being the defacto industry standard for electronic system modeling, we feel that it is therefore essential for students of electrical engineering and microelectronics. To get in touch with this language is a key addendum to lectures on system design, VLSI (Very Large Scale Integration) or related topics, providing students with valuable additional skills.

Teaching an extremely rich programming language like SystemC requires multiple-hour introductory lectures given as presentations or demonstrations. The necessary subsequent practical exercises must be organized at a later point in time, when important objectives might already be forgotten, and on computers where the necessary software tools are installed. Lessons on SystemC must incorporate fundamentals of object-oriented programming such as inheritance, class and interface design etc, as well as hardware design principles

including module hierarchy, communication, concurrency and time. This high level of complexity can often overwhelm students. But for graduate students in the field of IC design, applying the appropriate methodologies and getting to know the corresponding tools is crucial. We believe that learning by doing is probably the best way to apprehend a complex programming language like SystemC.

Computer based learning is very popular nowadays and it is preferred to books when it leads to an appreciation in the learned knowledge. All kinds of contents are supported and the teaching material can be adapted to individual learning situations. Because of their targeted evolution and their animated features, multimedia techniques shorten the process of gathering information and learning. The individualism aspect of e-learning is welcome by students who wish or need their own progression speed.

Existing SystemC crash courses on the Internet provide only a one-way teaching without interactive functions or direct feedback. They present a basic scheme with a few yes-or-no questions. But there are significant advantages of the interactive teaching method: First, the course participant has the ability to control the speed of the presentation by himself and navigate in the tutorial following his own needs. Secondly, compared to a pure verbal explanation, many learning objectives are easier to understand when they are presented by animated graphics. Finally, the different objectives are divided into smaller parts and come with short related exercises.

Our department has collected valuable experience in the the process of realizing an interactive VHDL tutorial in the past few years [6], which incorporates the basic e-learning ideas and realizes the interactivity approach pointed out above. Now, as an advancement to the well-established and successful VHDL course, we are designing a new tutorial as a practical introduction to the design of integrated systems with SystemC. It has been entitled SCOTT – SystemC Online Tutorial and Training [8]. This tutorial is addressed to students with a background in electronic system design, to people who look to refresh their knowledge in SystemC, and to those who want to complement lectures on system design or simply want to practice this language.

4. SCOTT'S TECHNICAL REALIZATION

SCOTT is a web-based interactive online tutorial using server-side tools for compilation, simulation and testing of SystemC codes submitted by the user over the Internet. The system's infrastructure is adapted from our department's VHDL tutorial mentioned in the previous section.

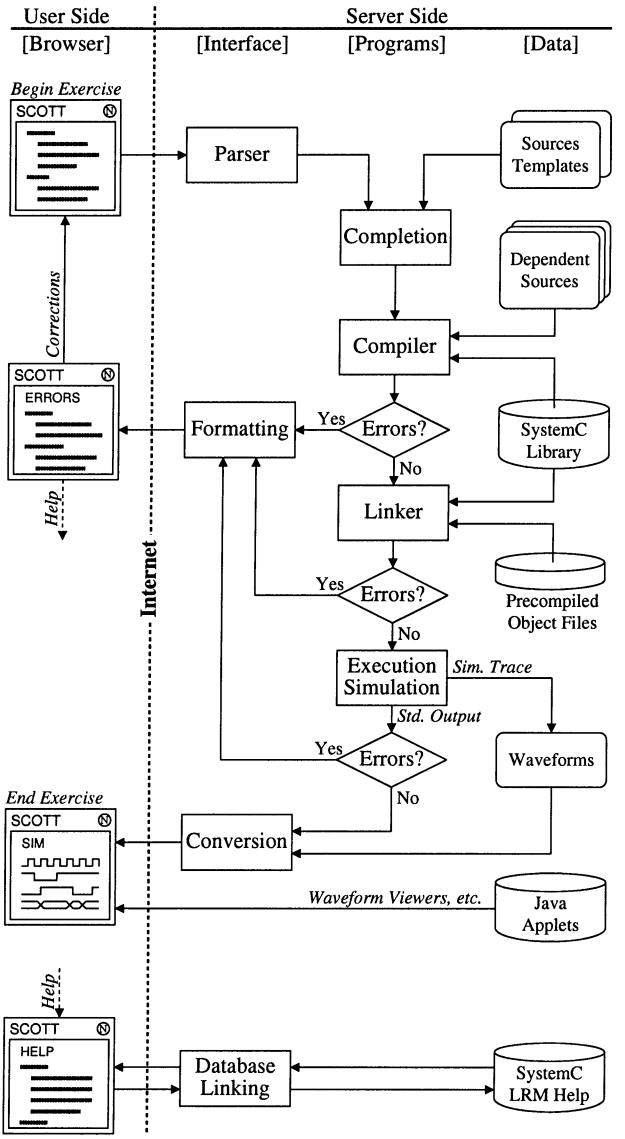

Figure 2. Flow of a typical exercise for SCOTT [8]: Communication between learner and our server-sided SystemC environment is enabled by the Internet.

4.1 User Interactivity and Administration

An important part of an interactive tutorial has been foreseen for practical exercises. In the case of SCOTT, two different types of exercises are provided: Some are directly related to the SystemC syntax and are preferably presented

using Java applets as seen in Figure 3. Other exercises intend to be managed within a complete simulation environment and require the use of a compiler, as seen in Figure 2. The compiler and the other necessary tools are hosted on our server. The user's input – e.g., code fragments being placed in a provided code framework – is acquired through processing HTML (Hyper-Text Markup Language) forms and then fed to the GCC (GNU Compiler Collection) [7]. Thus, the only user-side precondition is an Internet browser with Java and JavaScript enabled. Relevant output messages are being parsed, and the user gets either a success statement or a commented error message, along with suggestions for debugging or improvement. Moreover, a help functionality is constantly provided through a separate database of keywords, non-exhaustively extracted from the SystemC Language Reference Manual [2].

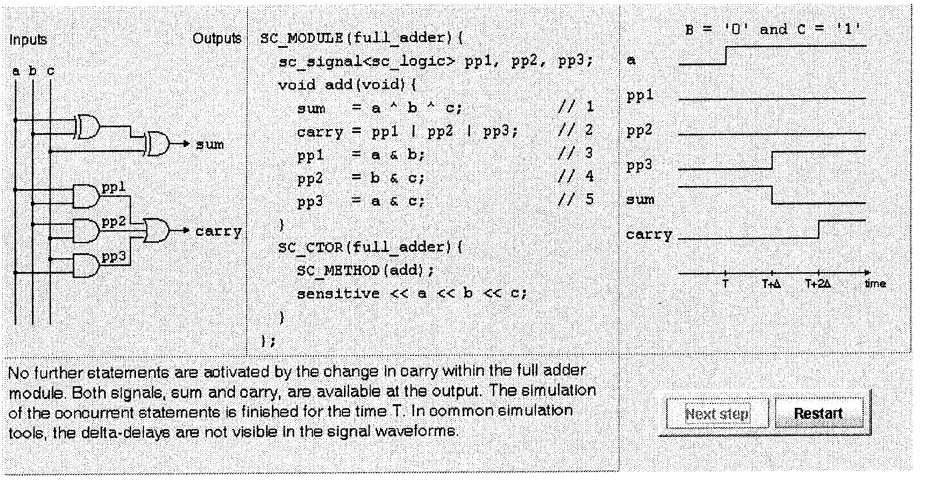

Figure 3. Example of a Java applet in the learner's web browser explaining simulation delta cycles step by step

To access the tutorial, a user name and a password are required to login. For the initial registration, the users enters his e-mail address to receive the authentication data. Each user's tutorial progress and related data is saved in a dedicated directory on the server. This permits an interruption in the tutorial and a resuming at the same position at a later time. Also, the submitted code does not have to be retyped in the next session. Appropriate security measures are taken to protect the user directories from unauthorized access.

4.2 Exercises with Simulation

As depicted in Figure 2, six steps are required for executing an exercise that includes simulation. First, the user's code entered by the user is parsed

in order to reject apparently malicious or syntactically incorrect code. Secondly, the source code is completed with declarations, function implementations and other templates the user is not supposed to modify but required to form a complete and compilable project. Then, this reconstructed source code is compiled into object files according to the definition of the SystemC headers. If all syntax errors have been removed and no problem occurs at the compilation, the linker brings together the compiled code from the user with the user-independent pre-compiled files and the obligatory SystemC library components. This step creates the executable simulation model. The execution of the produced binary file on the server is the next step. The simulation of the design is performed in a local directory with restricted access to the rest of the file system. Finally, additional information – e.g., a wave output window displaying interesting waveforms in VCD (Value Change Dump) format – is shown with the help of a Java applet for the analysis of the produced results on an answer page. At each of these steps, possible errors are formatted in English, analyzed and returned to the user. Then, he can choose either to find and fix the errors by himself or to take a look at the dedicated help page, as seen in the bottom of Figure 2. On the server, the software packages CGI (Common Gateway Interface) and SSI (Server Side Includes) ease the automation of such exercises and allow a rapid evolution of the tutorial contents.

SCOTT has begun its development phase and intends to cover all the specifications of SystemC version 2.0 [2]. For license reasons, it currently supports all abstraction levels down to synthesizable RTL but excludes the synthesis step which would require too specific EDA tools. It provides different kinds of exercises and offers the possibility for the users to test their knowledge in each part of the tutorial. Moreover, its navigation system includes not only "back" and "next" keys for pursuing the tutorial, but also a supplementary level of information which may be skipped if no further information on the current topic is requested. Two more options give the ability to consult the on-line help and to return to the chapter overview.

4.3 Exemplary SCOTT Lessons

Figure 4 shows an interactive applet for the visualization of different logic functions. According to the SystemC datatype `sc_logic`, also the unknown 'X' and high-impedance 'Z' states are featured. The truth table cells' entries can be changed by a mouse click. Once the table has been completed, the "verify" button reveals eventual mistakes by highlighting cells with incorrect values.

The second type of applet serves for demonstration purposes without the need for interactive functionality. By using a combination of syntax highlighting, graphical visualization and explanatory text, understanding the matter is

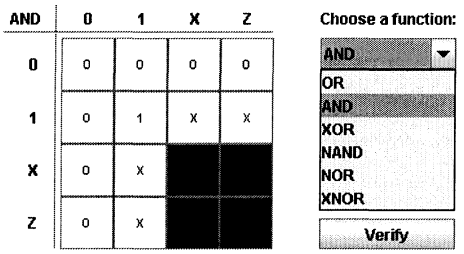

Figure 4. Exercise applet for the interactive visualization of different logic functions using the `sc_logic` datatype. The highlighted cells reveal incorrect truth table entries.

facilitated. An example illustrating the construction of a modular design is given in Figure 5. The course of action is controlled by the buttons in the bottom left corner.

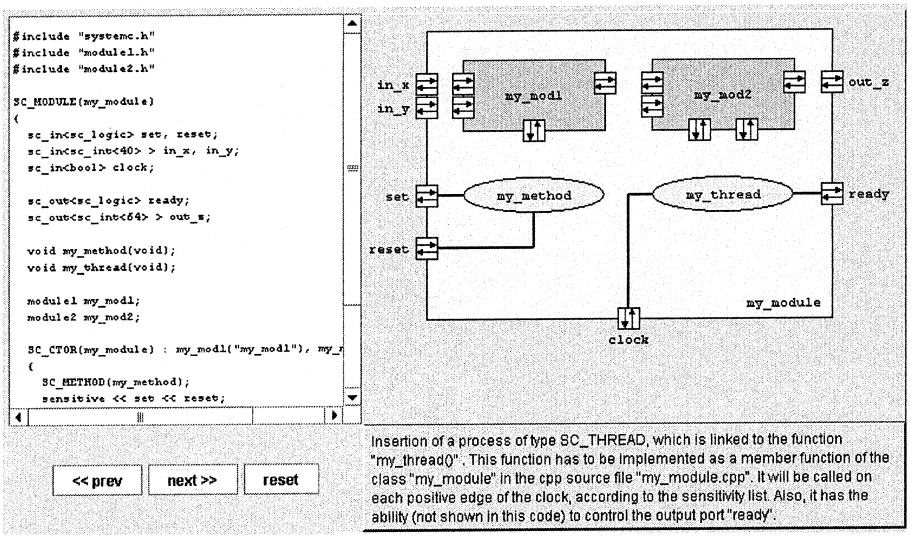

Figure 5. Demonstration applet for the visualization of constructing a modular design including ports, signals, processes and submodules

For a hands-on practical training of programming in SystemC, SCOTT enables the user to enter his original self-made code into online HTML forms as depicted in Figure 6. A template or framework is given and the user's task is to complete the source appropriately in order to solve a specific problem. Also, the user can save the text field's content in his dedicated server directory for reaccess when resuming the tutorial at a later time. Local files can be uploaded

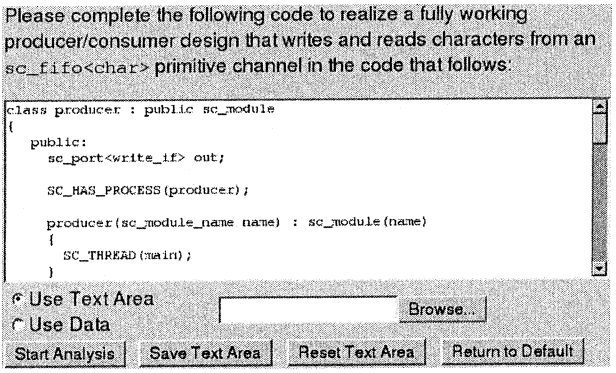

Figure 6. Online form for submitting user code and interfacing with the user's directory

using the "Browse" button. Of course, the default can be recalled anytime. When the "Start Analysis" button is clicked, the flow of Figure 2 is initiated.

Interface classes are used to declare the access methods that channels provide. Reciprocally, those are used to implement the access methods declared within the interfaces. In a good design, one uses class inheritance in order to minimize the number of distinct interfaces and C++ multiple inheritance to group common interface methods and reduce code duplication. Exploited in one exercise of SCOTT, a bus channel implements simple read and write transactions using pure virtual functions. The class `simple_rw_if` inherits virtually from the standard SystemC class `sc_interface`.

```
class simple_rw_if : virtual public sc_interface {
 public:
  virtual void read(unsigned addr, char* data) = 0;
  virtual void write(unsigned addr, char* data) = 0;
}
```

Hierarchical channels in SystemC are meant to facilitate TLM (Transaction Level Modeling). They are modules used to implement the methods declared in the interfaces and encapsulate both the structural elements as well as the communication protocols. Since they may also embed child modules, they provide a good starting point to begin a global design in order to refine and extend it progressively. One of the exercises of our tutorial is based on an example from [4], the goal of which is to explain the refinement process of a primitive channel. In this example as seen in Figure 7, a hierarchical channel `hw_fifo<T>` implements with a template parameter `<T>` the same interface as `sc_fifo<T>` from the SystemC library. SystemC's refinement strategy consists of making only small and conservative changes to the design and to

revalidate each refined model. The purpose of this exercise is to give a good introduction to the process of communication refinement.

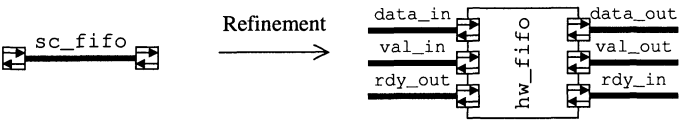

Figure 7. Refinement of a primitive channel into a hierarchical channel

5. RESULTS AND CONCLUSIONS

SCOTT is a multimedia training program intended for graduate students who want to attend further education and training. It is also applicable to professionals in the industry where dedicated time to learn is scarce. On the basis of its server-sided structure, using the tutorial does not require special hardware or software. We utilize CSS (Cascading Style Sheets) to ensure the clear separation of contents and layout, and to increase the tutorial's portability across different browser platforms and operating systems. Thus, the user only needs Internet access and a standard web browser to take full advantage of all lessons and functions.

An outstanding benefit of SCOTT compared to other online resources conveying knowledge on SystemC is the concept of real-time interactivity: Instead of a one-way transfer with pre-defined code examples, the SCOTT user can submit his own original code. The submission is being processed instantaneously and all interesting results are presented right away, along with error analysis, additional information, and even simulation results and waveforms. Every participant can go through the course at his own pace and take advantage of the automatically generated hints and context-sensitive reference, in order to internalize the SystemC language very quickly.

Moreover, the knowledge taught in the different parts of our tutorial can be deepened in an additional series of exercises available to students at our department, based on commercial software environments for the simulation, co-simulation and synthesis of larger SystemC projects. For further immersion in this field after completing SCOTT, the trainee is invited to install the necessary compiling environment [7] and the SystemC libraries [2] on his own computer and practice the acquired modeling language at no cost.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Endric Schubert and Thomas Kumpf for helpful advice and suggestions. This work is being supported by the University

of Ulm's "Lehrfond 2005: Einsatz neuer Medien in der Lehre" funds for new media in teaching projects.

REFERENCES

[1] J. DeGroat, A. Raman, B. Younis, "A Design Project for System Design with SystemC", *IEEE International Conference on Microelectronic Systems Education*, pp. 108-109, 2003.

[2] Open SystemC Initiative (OSCI) *http://www.systemc.org/.*

[3] S. Liao, "Towards a new standard for system-level design", *Proceedings of the* 8^{th} *International Workshop on Hardware/Software Codesign*, ACM, pp. 2-6, 2000.

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

[5] J. Bhasker, *A SystemC Primer*, Star Galaxy Publ., 2002.

[6] R. Geissler, S. Hirsch, H.-J. Pfleiderer, "A Web-Based-Only Interactive VHDL Tutorial", 6^{th} *International Conference on Mixed Design of Integrated Circuits and Systems*, 1999.

[7] GNU Compiler Collection (GCC), Free Software Foundation, *http://gcc.gnu.org/.*

[8] SCOTT (SystemC Online Training & Tutorial). Hosted by the University of Ulm in the Department of Microelectronics at *http://scott.e-technik.uni-ulm.de/*, 2005.