

# Teaching System-Level Design using SpecC and SystemC

Robert D. Walstrom, Joseph Schneider, and Diane T. Rover  
*Department of Electrical and Computer Engineering*  
*Iowa State University, Ames, IA 50011*  
*{robw,josschne,drover}@iastate.edu*

## Abstract

*System-level design of embedded computer systems is essential to manage complexity and enhance designer productivity. Viewing designs at different abstraction levels allows developers to evaluate the system performance early in the design cycle. System-level design languages (SLDLs) allow the representation of a system at multiple levels of abstraction. Two leading SLDLs, SpecC and SystemC, are taught in Iowa State University's graduate-level embedded systems course, CPRE 588. SpecC includes a handful of explicit language constructs that directly address characteristics of embedded applications, which gives students a quick start into understanding the concepts of system-level design. Moreover, a clear refinement methodology is defined for SpecC. However, SystemC has widespread industry support in many commercial tools, which gives students experience with actual practices. In this paper, we describe our strategy of teaching both SpecC and SystemC so as to leverage the strengths of both languages and provide a balance of theory and practice.*

## 1. Introduction

A defining aspect of SLDLs is their ability to represent the system at many levels of abstraction. Using abstraction, developers are able to defer the implementation aspects of a system as late as possible during the design phase. The system is first implemented in a behavioral sense, with no reference to communication or implementation details. The developer makes decisions or refinements, defining the system into lower levels of abstraction, eventually culminating the final implementation.

Teaching students the necessary abstraction concepts can be difficult; a problem that is only compounded if concrete steps cannot be presented in a way which students can follow. The graduate

embedded systems course at Iowa State University (CPRE 588) has incorporated two SLDLs, specifically SpecC [1] and SystemC [2], into its curriculum for several years. As a result, we have developed an approach to introducing SLDLs that is intuitive and useful for the students.

## 2. SLDL Educational Considerations

When considering the suitability of a SLDL for use in a course, several considerations should be taken into account. Specifically:

- Ease of the SLDL syntax for target students
- Educational value
- Refinement methodology support
- Industry support

While no SLDL is a clear-cut leader, SystemC has garnered the most industry support in the United States. However it uses advanced C structures, such as templates and macros, which can be distracting when trying to teach the system-level design paradigm. SpecC lacks industry support, but more cleanly represents the abstraction levels of a system and possesses a detailed set of refinement rules to modify a system design through each level of abstraction.

Therefore, we propose a two-step educational curriculum: allow students to become familiar with system-level design techniques using the SpecC methodology and refinement steps, and once their understanding is sufficient, introduce SystemC.

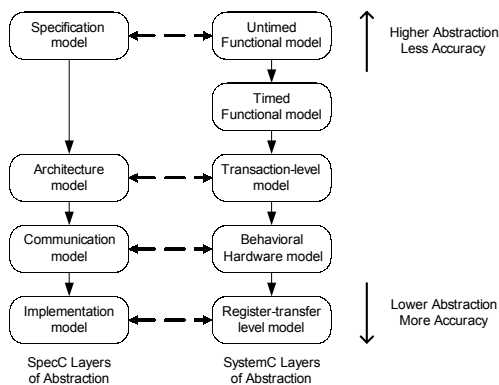
## 3. Two-Step Curriculum using SLDLs

CPRE 588 ([class.ee.iastate.edu/cpre588](http://class.ee.iastate.edu/cpre588)) introduces students to the concepts behind system-level design as well as the SpecC and SystemC SLDLs. The target audience for this course is graduate students in computer engineering, computer science, and electrical engineering. Thus the programming skills of

the students can vary depending on their background.

For this reason, we have found it is easier to start teaching students SpecC as their first SLDL. The textbook [1] we use for the course is especially helpful to students when learning design concepts and the SpecC language concurrently because each level of abstraction is discussed in detail and a set of refinement rules for translating a design to each level of abstraction is presented. Code examples from the textbook and [3] are discussed in lecture, which are useful in demonstrating how particular components of a system may be modeled using SpecC as well as presenting the syntax of the language.

The second phase of the curriculum is the introduction of SystemC. At this point, students are familiar with the layers of abstraction supported by SpecC as well as the design methodology and refinement rules. The first step is to identify the similarities and differences between SpecC and SystemC at a high level. Both SLDLs support multiple layers of abstraction, which are differentiated by how accurately they represent certain aspects of the final implementation. Some metrics common to both SLDLs include functional accuracy, computational timing, structural accuracy, communication timing, and communication protocol accuracy [4]. Based on these metrics, the equivalent layers of abstraction are identified in Figure 1. In [4], a top-down design methodology with detailed refinement steps for SystemC is presented. With this methodology, students recognize that SystemC has the ability to achieve the same goals as SpecC in terms of refinement.



**Figure 1: Equivalent layers of abstraction between SpecC and SystemC.**

The second step is to cover the syntactical differences between SystemC and SpecC. The basic computational units in SpecC are called behaviors while the basic computational units in SystemC are

called processes. In some ways, these units are similar, but there are specific differences in how they may be used. Other modeling components, such as channels and interfaces, are similar in concept when comparing the two SLDLs, but their actual implementation in the code is quite different. The best way to demonstrate the differences is to present a piece of code in SpecC and its equivalent representation in SystemC.

Homework assignments gradually introduce students to the concepts presented in lecture. These assignments range in difficulty from modifying a given piece of code to modeling a portion of a defined system. A design project is also assigned towards the beginning of the semester. Students working in small groups begin by defining a small-scale system and creating a specification model in SpecC. As the SpecC methodology is covered in lecture, students develop architectural and communication models in SpecC. By the end of the semester, students model their system using SystemC at one or more levels of abstraction. The design project exposes students to both SLDLs taught in the course and serves as a hands-on application of the top-down design methodology.

## 4. Conclusion

Teaching system-level design using SLDLs is beneficial to students in terms of facilitating their understanding of the design concepts as well as preparing them for applying their knowledge in industry. In our curriculum, we have found that teaching two SLDLs is the most effective way to covering the theories of system-level design as well as introducing students to a SLDL that has wide industry support.

## 5. Acknowledgment

This work was supported under NSF grant no. EEC-0088071.

## 6. References

- [1] D. Gajski et al., *SpecC: Specification Language and Methodology*, Kluwer Academic Publishers, Boston, 2000.
- [2] T. Grötter et al., *System Design with SystemC*, Kluwer Academic Publishers, Boston, 2002.
- [3] A. Gerstlauer, “SpecC Modeling Guidelines”, tech report CECS-02-16, Center for Embedded Computer Systems, Univ. of California, Irvine, 2002.
- [4] R. Walstrom, “System-Level Design Refinement using SystemC”, Dept. Elect. And Comp. Eng., Iowa State Univ., Ames, IA, 2005.