Preface

The book *ESL Design and Verification* (Elsevier 2007) [1] coincided with a point in time at which the authors believed that ESL had started to become something real and tangible. Before then it had been a collection of university projects, isolated pockets of tool development, and even patchier levels of adoption. Most of the tools that were created were based on proprietary languages, with little if any external interfaces or extensibility. When we started to write the book, we believed that ESL had turned the corner and that we were at the starting point of something big happening in the industry. That book laid the groundwork for the entire domain that is categorized as ESL and we provided our best assessments about where the technology was heading and the things that people should pay close attention to.

That was just a few years ago, and in that very short period of time, there has been a significant maturing and development in certain areas of the ESL flow. These developments have brought ESL from a technology to keep your eye on to one that you should be considering adopting if you want to have the highest levels of productivity, or improved power and/or performance in the products that you design, and to attain the highest possible levels of quality. At the same time, almost nothing from that first book has been made redundant or has been shown to be a wrong prediction of where the industry was heading. As such this book is not a replacement for the previous book, but one that expands on some of the concepts that were first introduced there.

It is also uncanny how a small development can have such a profound set of implications, especially when that development is not even very advanced. As people have often said in the past, when we look at the progress we have made in a year, we are usually disappointed. When we look at the progress we made over the period of 10 years, we are often amazed at how much was accomplished. In the past year, one such event happened that on the surface of it was not that significant, but it already showing the magnitude of the implications that it will have.

We are talking about the release of the OSCI TLM 2.0 specification, which creates a standardized way to connect models described at the untimed or approximately timed transaction level. While it is essentially able to connect any forms of models, it really only addresses the memory-mapped communications domain. The ramifications of its introduction have been huge. Instead of every vendor of system-level virtual platforms having their own proprietary languages, models, and tools,
every major developer of these platforms is now beginning to standardize on the use of TLM 2.0 as the way in which to interconnect models or is planning to do so within their next development cycle. Models developed for one system will be able to work on another, meaning that the problem of model availability and true interoperability is now being solved. A concerted effort is going into making SystemC a usable simulation environment and is becoming the accepted interconnect language for a large part of the ESL flow.

This does not mean that the industry has agreed upon the general usefulness of SystemC as a modeling language, and many vendors still only use it where they want to create interoperability layers in their products. This may change as each of the pieces of a SystemC flow becomes more optimized. High-level synthesis tool vendors are still locked in a battle for the best input language, with C, C++, and SystemC being the three strong contenders. At the same time, the software community is looking for solutions that enable them to embed the concepts of distributed computing into their domain and to remove the unwanted dependencies created in languages such as C. We think we have a long way to go before we eventually settle on the ideal language(s) on which we can base the next few decades of evolution.

This book will take a more practical approach than the first book, in that it will only cover the areas of the flow that are seeing real adoption and where the level of maturity is such that significant gains can be achieved by those who adopt them. It will concentrate on the models that are the cornerstone of those flows and will provide concrete examples showing how they are created, how they are used, and how they get transformed.

In order to provide that level of detail, it becomes necessary to showcase specific languages, tools, and flows. The choice of tools is in most cases arbitrary and readers should not take away any impression that we are endorsing those tools – even if personally we consider them among the best available! In many cases, several competing tools exist, and you should evaluate all of them before making any purchase decision.

Who Should Read This Book

We hope that this book will be useful to many types of readers, including

- **The novice in ESL**, who may be a student in electronic design, EDA, software design, system design, or related fields of study; or may be a designer seeking to learn more about ESL modeling, design, and verification. This book will provide you with an in-depth understanding of the areas of ESL that are actively being adopted by the design community. For those just entering the domain, it is hoped that the taxonomy will provide a clear understanding of the attributes of models as they exist in this domain, something that is often obfuscated by marketing departments.

- **The experienced designer** who has some base in ESL design and verification may find that there are many aspects of the process with which they are not familiar. For example, someone who is familiar with the concepts of high-level
synthesis may not be aware of the developments in the virtual prototyping space. In addition, someone contemplating the adoption of virtual prototypes may be confused by the wide variation in capabilities offered by the vendors, not understanding that the platforms are directed toward completely different tasks and have thus made implementation choices that are optimized for those tasks. This book provides a good opportunity for specialists in all aspects of the design flow to fill in gaps in their experience and knowledge base and improve their understanding of the overall tradeoffs and issues.

- **The design manager** who wishes to understand particular parts of the ESL flow, to improve the capabilities of his or her design team, who wants to reduce the risk in their development projects, who wants to expand the scope of their design projects and identify key missing components of the team will have a lot to gain by reading this book from cover to cover. It is also suggested that they also read the earlier book *ESL Design and Verification* [1], which will provide them with a more complete overview of the entire ESL flow.

- **The researcher** in academia or in an industrial research lab may have an excellent view of their subject area, but not have a good view of how it fits into the overall ESL design space. In addition, they may not be fully aware of the state-of-the-art industrial practice in ESL: those methods that have been proven to work and are recommended and those methods that have not found favor with practical design teams. This book will give them considerable insight in particular into industrial practice. We hope as well that the taxonomy defined in this book will be adopted by the academic community and extended, or modified, with the advances that they are making. A common classification scheme will help to remove a lot of the ambiguity that has been so common in this field.

- **The educator** in academia or within an industrial company who needs a text on which they can base a class or course. Many such courses were based on the first book *ESL Design and Verification*, especially at the post-graduate level, and this book provides a lot more detailed material on which more extensive courses can be based. While it is not specifically written as a textbook, the authors are willing to work with educators to help them develop materials.

### Structure of the Book

This book is constructed in two major sections:

- The first section concentrates on the languages and models that are becoming commonly used within ESL flows. We try to be open and honest about these, giving both the positives and the negatives associated with them. We have tried to keep personal bias out of this as much as possible, but in some cases it may be a little obvious what the authors’ opinions are. Tools are introduced when it helps to demonstrate the concepts being discussed. In many cases dedicated books exist that will provide a lot more depth than is possible in this book but the information
Preface

The first section of the book will be especially useful to those who want to understand some of the concepts behind the models and languages that are in use today, particularly in the functional modeling and verification spaces. The second section may be more useful to those who want to understand particular aspects of the ESL flow in more detail and how tools are developing to meet the needs of various types of practitioners.

Chapter Listing

- **Chapter 1: Introduction**: This chapter presents the fundamentals of models and languages by taking a historical look at the development of some early models. It looks at how models can evolve over time and why several abstractions of those models are necessary. Languages are used to build those models and we briefly review some of the basics of language construction. The chapter briefly describes various model types that are in active use for the design and verification of systems within an ESL flow. The model taxonomy is used to show how some of the functional languages that are being used have different characteristics that make them better at certain kinds of tasks. Finally, the chapter provides a set of definitions that are used throughout the book.

- **Chapter 2: IP Meta Models for SoC Assembly and HW/SW Interfaces**: This chapter explores in detail one class of models that are called meta models. It looks at the efforts of the SPIRIT standards organization in their attempts to create an IP metadata model. The goal of this was to enable platform assembly in a somewhat intelligent manner by encapsulating knowledge about the interfaces. The most valuable part of this effort has led to the standardization of register definitions, which has enabled metadata to be shared between the hardware and software teams. This chapter explores this usage of metadata extensively.

- **Chapter 3: Functional Models**: This chapter looks at models that are the mainstay of most design and verification flows, namely functional models. Starting from high-level mathematical models it examines the models in active use in ESL flows, concentrating on the SystemC language and TLM 2.0. It looks at the main concepts of the SystemC language that enable computation, communications, and
synchronization. TLM 2.0 provides communications and timing capabilities that enable modeling at various levels of timing accuracy. The chapter also examines the rules that go along with TLM usage and the need for compliance checking in order to insure interoperability. Finally, the chapter takes a brief look at some declarative models.

• Chapter 4: Testbench Models: Testbenches are constructed from a number of models that work together with a variety of tools to produce a verification environment. This chapter looks at how those models work together and briefly surveys some of those models, such as comparison models, constraint models, coverage models, and verification plans. It looks at some of the problems created when these environments are constructed from multiple languages and some techniques that are being used to overcome them. Finally the chapter looks at verification IP and standardization efforts that are underway.

• Chapter 5: Virtual Prototypes and Mixed Abstraction Modeling: This is the first of several chapters that take an in-depth look at models, and how they are used within specific parts of an ESL flow. This chapter looks at a system-level virtual prototype of the hardware system that is specifically intended to be used for the development and testing of software. This chapter also demonstrates the transition that is going on in the industry: away from proprietary systems and interfaces toward open standards. The chapter uses the Innovator tool from Synopsys to demonstrate many of the capabilities that can be found in tools of this type.

• Chapter 6: Processor-Centric Design: Processors, Multi-processors and Software: Most embedded systems have one or more processors at their core. This chapter looks into the issues involved in processor-centric design using a commercial configurable and extensible processor IP as an example. The chapter uses the Xtensa product line from Tensilica to demonstrate the design of a JPEG decoder. Starting from performance analysis, a series of design possibilities are investigated which lead to some surprising tradeoffs being made between performance, area, and power. Finally the chapter looks at some of the issues associated with the design and debugging of multi-processor systems.

• Chapter 7: Codesign Experiences Based on a Virtual Platform: This chapter delves into the world of hardware–software codesign where hardware and software are considered to be equal partners. Using a JPEG decoder as an example, this chapter explores the entire flow from specification to implementation using a platform-based development approach based on SystemC. The methodology allows easy migration of modules between the two domains and explores a number of ways in which the design can be partitioned and mapped onto a number of platforms having different numbers of processors, communications channels, and memory architectures.

• Chapter 8: Transaction-Level Platform Creation: Not all system-level virtual prototypes concentrate on the same issues or are used for the same purpose. In this chapter we look at a transaction-level platform that is intended for making hardware architectural choices. This requires being able to quickly and easily modify the timing associated with computation and communications within the platform. This requires layering of the timing model onto a functional model without
requiring changes to the functional model, all while using open industry stand-
ards. This chapter looks at the way that the Vista product from Mentor Graphics
overcomes these challenges.

- **Chapter 9: C/C++ Hardware Design for the Real World:** This chapter explores
  the domain of high-level synthesis of hardware from a C/C++ description. It
  explores the language choices that could be made and looks at some of the funda-
  mentals associated with this level of synthesis. Unlike RTL synthesis, high-level
  synthesis is an exploration process and requires a lot of analysis and feedback
tools as well as understanding how languages embed architectural constraints.
  Using the CatapultC synthesis tool from Mentor Graphics, an almost real-time
  example is provided for the development of a matrix inversion block, which is
core to many complex communications systems.

### Relationship to First Book

*ESL Design and Verification* was published in 2007 and contained a complete
overview of the ESL domain and the state of tools and languages at that time. It
developed many of the foundations and concepts that are extended in this second
book and as such these two books are meant to be read as if they were a single
larger work.

We have attempted to make this second book completely self-contained and this
does mean that there are several places where concepts were introduced in *ESL
Design and Verification* and need to be understood within the scope of this book.
In those cases we have provided a summary of the concepts in this book, but have
not attempted to bring the entirety of the section forward. Those wishing to fully
understand those concepts should read the corresponding sections of *ESL Design
and Verification*.

One piece of feedback we received on the first book was that it did not go into
enough detail. We hope we have rectified that problem with this book. While it may
still not provide enough detail for people to become experts in the use of the various
tools, we hope that there will now be enough detail that they can understand the
concepts well and be able to fill in some of the blanks for themselves.

### Companion Web Site

We are always open to feedback, either good or bad. We like to hear about the prob-
lems you are facing and what you would like to appear in the next book (if there
is a third). We would encourage you to go to the companion site for these books
http://ElectronicSystemLevel.com where we will be providing additional correc-
tions and in some cases additional material that did not make it into the book, such
as complete code listings. We also have a discussion forum attached to that site and
we would encourage you to ask the community for help or guidance for anything related to the ESL space.

**Reference**

ESL Models and their Application
Electronic System Level Design and Verification in Practice
Bailey, B.; Martin, G.
2010, XXIV, 446 p. 177 illus., Hardcover
ISBN: 978-1-4419-0964-0