Foreword

“A good model can advance fashion by ten years.”
Yves Saint-Laurent

I am very pleased to introduce here a comprehensive and well-balanced guide into model-based approaches for embedded system design, validation, and implementation.

Embedded systems design is rapidly evolving through its own fashions and trends. It is an expanding area of development in computer-based systems engineering. This is caused by both the demand of widening applications for embedded systems in many sectors of economy and people’s daily life, and the availability of enormous computing resources on a single chip. Billion transistor systems-on-chip (SoCs) with multiple processor cores and complex communication networks are already becoming a reality. New design paradigms such as reconfigurable networks on chip and IP core reuse help tackle the biggest challenge of all, the problem of a widening “productivity gap” between what is available on a die for building a system and what can be absorbed by a design process with existing tools. As never before the role of software is increasing in SoCs because it helps to facilitate the incorporation of more complex features into these increasingly more miniaturized devices to render them more reactive and dependable, concurrent and user-friendly etc. In the scope of embedded applications, such as industrial control and automotive, the aspects of heterogeneity, reconfiguration, multi-functionality are paramount, too.

Let’s turn our attention to the 2007 edition of International Technology Roadmap for Semiconductors (ITRS): “Software aspects of IC design can now account for 80% or more of embedded systems development cost, especially as we have fully entered the era of multi-core designs, with both homogeneous and heterogeneous architectures…Embedded software design has emerged as the most critical challenge to SOC productivity…”

Systems are now so complex that designers must use tools and must reuse existing blocks to reduce the cost of the design, time-to-market, and to make the systems meet requirements in performance and reliability. They must use abstractions, too. To facilitate these, models are an intrinsic part of these tools and practices. The concept of design flow is basically a concept of tool flow, and, in turn, a concept of model flow. As much as the models are important, the semantic relationships between these models are crucial too, because of the needs for model transformation and compositionality.

The need for the use of behavioral models is again emphasized by ITRS: “For decades, designers have reasoned about systems at various levels of abstraction (block diagrams, incomplete state charts, program models, etc.) with little support from design automation tools. This situation must change in the near future if necessary advances in productivity are to be achieved. To simplify the specification, verification, and implementation of systems including hardware and software, and to enable more ef-
ficient design space exploration, a new level of abstraction is needed above the familiar register-transfer level.”

Much like in the above quotation from the famous couturier, the success in how we design hardware and software for our complex computer-based systems depends on how good our models are and how well we apply them in our engineering practice. Building complex embedded systems without using behavioral models is unimaginable today. In the past, the scope of models was rather limited—finite state machines, timing diagrams, algorithm flow-charts… Those models were mainly used to capture the essential characteristics of some parts of the system, usually with the aim to improve the designer’s understanding of the behavior of the system, and help him/her to search for a good solution for a circuit or a program by looking at the behavior of the model. Today, the diversity and variety of models and their combinations, as they cover different aspects or views of the system, is such that their presence in the design practice is imperative. Indeed the models are used as a foundation of methods and algorithms of the very process of the construction of systems, their testing and evaluation.

The cost of developing design automation tools based on models is also an important measure. So is the issue of investing a research engineer’s time into development of good models. What is a good model then? What is the model that can help the design flow sustain several years of evolution? Such a model must, first of all, be powerful in terms of its intended (perhaps, even theoretical) use, and at the same time it should be manageable in terms of its practical use. These general truths about good models are eternal. Can we be more specific and argue about the qualities of the models that we need for designing modern embedded systems? Well, those must certainly be models that are universal in capturing the behavior of both hardware and software. In fact the question today is more and more about model-based frameworks rather than individual modeling languages and analysis methods. The frameworks are like environments where the existing algorithmic solutions can be plug-ins. To cope with the productivity crisis, reuse is needed in modeling terms as much as it is in terms of the system designs.

“Behavioral Modeling for Embedded Systems and Technologies: Applications for Design and Implementation” is a book which specifically addresses the issues of what modeling languages are good for embedded systems design and how model-based design can be put into use through correspondence between transformations of models and their system implementations.

I particularly like the structure of this book. Its fifteen chapters are written by individual groups of authors who are experts in their specific fields. The chapters are organized in five sections according to larger themes—Model-based Approaches, Aspect-based Approaches, Verification and Model-checking, Design Automation, and Industrial Applications. Each of these sections includes the chapters which are more basic and more specialized—this should suit the readership with different levels of expertise.

For example, the book has an excellent cover of the fundamental modeling issues relevant to embedded systems and their model-based design approaches. The modern views and recent research developments come into play along with more established material. This gives the book both a good tutorial value as well as a state-of-the-art reference item. The readers who are new in the field can familiarize themselves with UML, use of abstract state machines, model-based optimization for power and performance, principles of separation of concerns, and aspect-oriented design, the latter being absolutely vital in developing complex distributed real-time systems.

Embedded systems design as any system design would be incomplete without validation. This issue is addressed in the book through sections on the modeling of clocks and timed automata for distributed systems, the use of model checking techniques for timed automata, as well as more practical verification methods, which use SystemC platforms for behavioral simulation and performance evaluation.
The book is spot-on about the addressing the most important developments in the design automation for embedded systems and SoCs. My eye has been immediately caught by its in-depth and systematic treatment of the topics of embedded software synthesis, transaction level model automation, as well as programming models for reconfigurable architectures.

Finally, what is often missing from existing texts on embedded systems are solid and comprehensive guides on real-life applications. This book is unique in presenting applications in terms of serious industrial case studies coming from automotive and industrial control domains. These practical match nicely the whole spectrum of the issues discussed in the preceding theoretical material. The reader will find here examples of the practical use of modeling and descriptive languages, temporal logics, extensions of Petri nets, model-checkers and model-based testing methods.

Summing up my overall impression about this book, I strongly recommend it to anyone who likes to use behavioral models in systems design, cares about their interrelationship and diversity, and their role in computer-aided design and validation of complex embedded systems. The book serves as an excellent reference source and may be used as a text for advanced courses on the subject.

Alex Yakovlev
Professor of Computing Systems Design
School of Electrical, Electronic and Computer Engineering
Newcastle University, UK
January 2009

Alexandre (Alex) Yakovlev was born in 1956 in Russia. He received DSc from Newcastle University in 2006, and MSc and PhD from St. Petersburg Electrical Engineering Institute in 1979 and 1982 respectively, where he worked in the area of asynchronous and concurrent systems since 1980, and in the period between 1982 and 1990 held positions of assistant and associate professor at the Computing Science department. He first visited Newcastle as a postdoc British Council scholar in 1984/85 for research in VLSI and design automation. After coming back to Britain in 1990 he worked for one year at University of Glamorgan. Since 1991 he has been at the Newcastle University, where he worked as a lecturer, reader and professor at the Computing Science department until 2002, and is now heading the Microelectronic Systems Design research group (http://async.org.uk) at the School of Electrical, Electronic and Computer Engineering. His current interests and publications are in the field of modelling and design of asynchronous, concurrent, real-time and dependable systems on a chip. He has published four monographs and more than 200 papers in academic journals and conferences, has managed over 20 research contracts. He has chaired programme committees of several international conferences, including the IEEE Int. Symposium on Asynchronous Circuits and Systems (ASYNC) and DATE, and is currently a chairman of the steering committee of the Conference on Application of Concurrency to System Design. He is a senior member of the IEEE and Member of IET. In April 2008 he was general chair of the 14th ASYNC Symposium and 2nd Int. Symposium on Networks on Chip.