Guest Editors' Introduction: GALS Design and Validation

Mike Kishinevsky
Intel

Kenneth S. Stevens
University of Utah

Sandeep K. Shukla
Virginia Tech

GLOBALLY ASYNCHRONOUS, LOCALLY synchronous (GALS) design is practically a buzzword in industry today. This popularity is due to several factors: First, there has been unabated improvement in semiconductor manufacturing technology, leading to continual decreases in feature size (even as we write this, 45-nm process technology is becoming common in high-end microprocessor manufacturing) and increases in the number of devices that can fit on a single die. This in turn makes it more difficult to design a global-clock network that can control all the blocks in the design, and such a network significantly increases the overall power consumption. Second, shorter time to market leads to increased IP reuse, in which each IP block is designed and optimized for different clock speeds by distinct design groups. Third, many designs inherently require multiple clock domains with different clock frequencies because of the nature of the computations and communications they perform.

The term *GALS* was first used (to the best of our knowledge) in the work of Chapiro in his doctoral dissertation, "Globally-Asynchronous Locally-Synchronous Systems" (Dept. of Computer Science, Stanford Univ., 1984), in which he provided a solution using pausible-clock circuitry. Since then, the term has gained popularity in both academia and industry. Breaking the synchrony assumption in digital design is often unsettling for designers, and to alleviate the difficulty, researchers in EDA have been proposing various GALS-based solutions. However, the tools, verification techniques, and testing methodologies for asynchronous designs are not as widespread as for synchronous digital design, leading to the hitherto limited usage of GALS design approaches even after more than 20 years from Chapiro's introduction of the term and the concept.

Some key questions that a synchronous-design practitioner might have about GALS design, verification, and test are as follows:

- Synchronous digital design is well understood; the design methodology and flow are established; and tool vendors provide a plethora of design environments that aid in design, verification, synthesis, and testing. Even though ample problems still exist because of size, complexity, power consumption, manufacturing variations, and validation cost, using synchronous abstraction and measuring clock-cycle computation delay simplify design methodology and are deeply embedded in current design practices. So, is there a way to avoid a GALS-based design and remain entirely in the world of synchrony?
- It may be possible to remain in the world of synchrony by elasticizing the interconnects and computation blocks through latency-insensitive (synchronous elastic) protocols. Elastic designs remain synchronous but allow flexible changes in design latencies and can be viewed as discretized implementations of asynchronous protocols. Are there general theories for designing latency-insensitive systems, and what are the best ways of implementing such systems? Would they alleviate the need for a GALS approach to some extent or could they be used in the context of multiple clock domains?
- There are many different ways of designing GALS-based systems using different clocking disciplines (for example, fully asynchronous local clocks; synchronous sources with clock transfers from sender to receiver; pausible clocks; and harmonic, synchronous multiclocks). Some of

0740-7475/07/$25.00 © 2007 IEEE Copublished by the IEEE CS and the IEEE CASS

these disciplines make the clock domains completely independent at the cost of synchronization problems between the domains. Others require satisfying some timing constraints in clock generation but remove the synchronization problem. Which disciplines are preferable and for which design classes?

- Asynchronous circuit design has been around for a long time. However, despite some advantages (for example, improved average-case performance and reduced electromagnetic interference), this design approach has not caught on in industry on a grand scale. The tools and design methodologies for asynchronous designs are not amply available from EDA vendors. Will we face the same problem with GALS design?
- Given that we have tools for test and verification of synchronous designs, can we design synchronously and then partition the system into synchronous islands, so that the GALS implementation of the design will simply be a refinement of the original validated design while still preserving the correctness of this design?
- What signaling and clocking issues must be solved to correctly implement high-level GALS protocols? Synchronization failures may be endemic in this kind of design. Can the absence of such problems be guaranteed for the eventual silicon implementations?
- Since the protocols between synchronous IP blocks will be self-timed, latency tolerant, and not necessarily clock driven, the simplicity of central control could be lost in GALS systems, leading to issues of deadlock and other distributed-system phenomena. How can we guarantee the absence of such problems?

Besides these questions, researchers have been looking into communication fabrics such as networks on chips (NoCs), on-chip communication infrastructures, and other possibilities. These arise in the context of SoC communication across heterogeneous, self sustained IP blocks—especially heterogeneous multiprocessor chips. Chances are, such fabrics will not be driven by a single, synchronized global clock. So, GALS protocols will have to run on such on-chip networks, and the resulting issues will generate more research and engineering questions.

Formal methods must play a key role in GALS design. The entire history of asynchronous design clearly demonstrates this. Successes in both analysis and synthesis of such systems are firmly based on formal models of concurrency, and the development of methods for validation and manipulation of such models. Fortunately, formal methods have matured over the years and can be applied to distributed systems with multiple synchronous islands.

In designing real-time software for a particular embedded platform, timing concerns can be abstracted away via the synchrony assumption. The pure functionality of the software component can be modeled with so-called *synchronous languages* such as Esterel, Signal, and Lustre. Correctness proofs of the functional models of such software can be established under the synchrony hypothesis. The analysis of the schedulability of computation, event acceptance and generation, and so on, can be done using formal Calculus. The real-time embedded code can then be generated, such that it is correct by construction. However, if the software is to run on distributed nodes, the synchrony hypothesis breaks down, because the communication of events between various computation entities is no longer fast enough to justify such simplifying assumptions. With this problem in mind, researchers working in the field of synchronous programming have proposed using GALS design to resolve this issue of distributed code generation, where the synchrony assumption is justified for the parts of the code that run on the same node of a distributed architecture, but asynchronous protocols are needed to bridge between these different parts of code and still guarantee correct behavior. There has been quite a bit of progress in formal methods for GALS design in embedded software. However, this special issue focuses on only GALS design in the hardware domain. Unfortunately, robust formal approaches to GALS design are still lacking in the hardware industry, and many of the solutions are often ad hoc.

This special issue introduces some of the basic issues of GALS design and validation in the hardware domain. We solicited articles on GALS taxonomy, a survey of prevailing techniques, case studies, general approaches of latency-insensitive designs, and the use of GALS protocols in NoC-type communication fabrics. We wanted to emphasize the need for more research and engineering innovation for GALS design and test. We received several very enlightening articles, and we selected five of them for publication in this issue.

The first article, ''A Survey and Taxonomy of GALS Design Styles,'' by Paul Teehan, Mark Greenstreet, and

Guy Lemieux, categorizes GALS design styles into three distinct classes: pausible clocks, asynchronous interfaces, and loosely synchronous interfaces. Examples, advantages, and relative pitfalls are described. Engineers looking into GALS-style integration of synchronous IP blocks and cross-domain communications will find the concepts and taxonomy presented in this article very useful.

"Globally Asynchronous, Locally Synchronous Circuits: Overview and Outlook," by Miloš Krstić et al., is another survey, but this article focuses more on the state of the art in GALS architectural techniques, design flow, and applications. These authors also prescribe several industrial inventions and changes in methodology, tools, and design flow that must occur before GALS-based integration of IP blocks can be more frequently used in industry.

Next, in "Adaptive Latency-Insensitive Protocols," Mario Casu and Luca Macchiarulo present a class of interblock protocols designed to overcome long multiclock interconnects. Several solutions have been proposed in the past, which the authors categorize as *static* solutions. The authors also present an adaptive solution, which they show to be more effective than these earlier solutions in terms of power, area, and throughput. Designers and researchers may consider this article to be a compact survey of the LIP literature, and may find the new solution useful.

The fourth article, "A GALS Infrastructure for a Massively Parallel Multiprocessor," by Luis Plana et al., presents a case study of a massively parallel multiprocessor aimed at real-time simulation of billions of neurons. Every node of the design comprises 20 ARM9 cores, a memory interface, a multicast router, and two NoC structures for communicating between internal cores and the environment. The NoCs are asynchronous, whereas the cores and RAM interfaces are synchronous, operating in independent synchronous-clock domains. The choice of a GALS design decouples clocking concerns for different parts of the die and improves power efficiency.

Finally, to address the issue of NoCs and GALS design, "A Highly Scalable GALS Crossbar Using Token Ring Arbitration," by Tejpal Singh and Alexander Taubin, describes a token-ring-based asynchronous crossbar that can be used as a communication fabric for connecting cores operating at different frequencies. This solution has advantages compared to previously published tree arbitration schemes for certain classes of applications. Since GALS design and NoCs are of growing interest, this article could be useful for designers.

If this special issue can generate more interest by researchers and industry practitioners in creating design tools, techniques, and validation methodologies for GALS design, we shall consider that it has served its purpose. We hope you enjoy this special issue! ■

Mike Kishinevsky is a principal engineer at Strategic CAD Labs of Intel, where he is responsible for front-end design. His research interests include high-level and asynchronous design, reactive systems, and models of concurrency. Kishinevsky has a PhD in computer science from the Electrotechnical University of St. Petersburg. He is a senior member of the IEEE.

Sandeep K. Shukla is an associate professor in the Department of Electrical and Computer engineering at Virginia Tech. He is also founder and deputy director of the Center for Embedded Systems for Critical Applications, and he directs the Fermat (Formal Engineering Research with Models, Abstractions, and Transformations) research lab. His research interests include formal methods for system design, reliability of nanoscale architectures, embedded-software design and verification, and system-level design languages. Shukla has a PhD in computer science from the State University of New York at Albany. He is a College of Engineering Faculty Fellow at Virginia Tech and is a senior member of the IEEE and the ACM. He is on the editorial boards of *IEEE Design & Test* and *IEEE Transactions on Industrial Informatics*.

Kenneth S. Stevens is an associate professor in the Department of Electrical and Computer Engineering at the University of Utah. His research interests include asynchronous circuits, VLSI, architecture and design, hardware synthesis and verification, and timing analysis. Stevens has a PhD in computer science from the University of Calgary. He is a senior member of the IEEE.

■ Direct questions and comments about this special issue to Sandeep K. Shukla, Department of Electrical and Computer Engineering, Virginia Polytechnic and State University, Blacksburg, VA 24061; shukla@vt.edu.