Polychrony for refinement-based design

Jean-Pierre Talpin[1], Paul Le Guernic[1], Sandeep Kumar Shukla[2], Rajesh Gupta[3], Frédéric Doucet[3]
[1] INRIA/IRISA, [2] Virgina Tech, [3] UC San Diego, [4] UC Irvine

Abstract

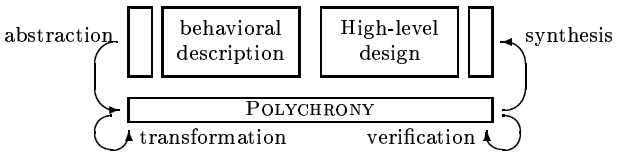
Rising complexities and performances of integrated circuits and systems, shortening time-to-market demands for electronic equipments, growing installed bases of intellectual property, requirements for adapting existing IPs *with new services, all stress high-level design as a prominent research topics and call for the development of appropriate methodological solutions. In this aim, system design based on the so-called "synchronous hypothesis" consists of abstracting the non-functional implementation details of a system away and let one benefit from a focused reasoning on the logics behind the instants at which the system functionalities should be secured. From this point of view, synchronous design models and languages provide intuitive models for integrated circuits. This affinity explains the ease of generating synchronous circuits and verify their functionalities using compilers and related tools that implement this approach. In the relational model of the* SIGNAL/POLYCHRONY *design language/plateform [3, 5] this affinity goes beyond the domain of purely synchronous circuits to embrace the context of architectures consisting of synchronous circuits and desynchronization protocols:* GALS *architectures. The unique features of this model are to provide the notion of* polychrony*: the capability to describe multi-clocked (or partially clocked) circuits and systems; and to support formal design* refinement*, from the early stages of requirements specification, to the later stages of synthesis and deployment, and by using formal verification techniques.*

Introduction In practice, a multi-clocked system description is often the representation or the abstraction of an asynchronous system or of a GALS architecture. In system-level design, the asynchronous implementation of a system is obtained through the refinement of its description towards hardware-software co-design. However, clocks are often left unspecified at the functional level, and no choice on a master clock made at the architectural level. As communication and implementation layers are reached, however, multiple clocks might be a way of life. In the polychronous design paradigm, one can actually design a system with partially ordered clocks and refine it to obtain master-clocked components integrated within a multiply-clocked architectures, while preserving the functional properties of the original high-level design, thanks to the formal verification methodology provided by the formal theory (model and theorems) of polychronous signals. Our goal is to derive conditions on specifications under which design refinement principles work. We seek towards tools and methodologies to allow to take a high-level SYSTEMC/SPECC specification and to refine it in a semantic-preserving manner into a GALS implementation. In this aim, we put POLYCHRONY to work in the context of the emerging high-level languages such as SYSTEMC/SPECC [2] and have studied the refinement of a high-level specification, the even-parity checker (EPC) in SPECC and show how it can be refined towards a GALS implementation with the help of POLYCHRONY.

The tagged model of polychronous signals (see [3]) is used for the formal study of protocol properties in this context. In this model, a *process* p is a set of behaviors b that have the same domain of signal names, written $\mathrm{vars}(p)$. Scalability is a key concept for engineering systems and reusing components in a smooth design process. A formal support for allowing time scalability in design is given in POLYCHRONY by the so-called *stretch-closure property*. The intuition behind it is to consider a signal as an elastic with ordered marks (the tags). If it is stretch, marks remain in the same relative order but have more space (time) between each other. The same holds for a set of elastics: a behavior b. If elastics are equally stretched, written $b \leq c$, the partial order between marks in c is unchanged. Stretching is a partial-order relation. It gives rise to an equivalence relation between behaviors $b \lesssim c$ (a clock equivalence relation). To model asynchrony, we consider a weaker relation which dis-

cards synchronization relations and allows for comparing behaviors w.r.t. the sequences of values signals hold. The *relaxation* relation $b \sqsubseteq c$ allows to individually stretch the signals of a behavior. Relaxation is a partial-order relation that defines the flow-equivalence relation $b \approx c$ and the meaning of asynchronous composition $p \parallel q$ by the set of behaviors d that are a relatxation of a behavior b of p and a behavior c of q. The model of polychronous signals defines formal properties that are essential for the component-based design of GALS architectures. *Endochrony* is a key design property design. A process is endochronous iff, given an external (asynchronous) stimulation of its inputs I, it reconstructs a unique synchronous behavior (up to stretch-equivalence). Endochrony denotes the class of processes that are insensitive to (internal and) external propagation delays. A process p is endochronous on its input signals I iff $\forall b, c \in p$, $(b|_I)_\approx = (c|_I)_\approx \Rightarrow b \lesssim c$. *Flow-equivalence* offers the right metric for checking the refinement of a high-level system specification with distributed communication protocols correct. *Flow-invariance* is the property that ensures that the refinement of a functional specification $p \| q$ by an asynchronous implementation $p \parallel q$ preserves flow-equivalence. Formally, p and q are flow-invariant iff, for all $b \in p \,|\, q$, for all $c \in p \parallel q$, $(b|_I)_\approx = (c|_I)_\approx$ implies $b \approx c$ for I the inputs of $p \| q$. In SIGNAL, GALS architectures are modeled as *endo-isochronously* communicating endochronous components. We say that two endochronous processes p and q are endo-isochronous iff $(p|_I) \,|\, (q|_I)$ is endochronous (with $I = \text{vars}(p) \cap \text{vars}(q)$). Endo-isochrony implies flow-invariance.

Capturing high-level design with polychrony In the polychronous design paradigm, one can give a functional-level specification of a system in terms of relations and partially-ordered clocks. A refinement, at the architecture-level, consists of isolating the master-clock of components and of integrating them within a multi-clocked architectures, while preserving the functional properties of the original design, thanks to the formal verification of flow-invariance. The main benefit of considering the model of polychronous signals for high-level C-like design languages lies in the formal semantics backbone/platform it provides, on which verification and optimization techniques can then be plugged in. There are several ways to envisage applying the POLYCHRONY model to high-level GALS architectures modeling in C-like design languages. The polychronous model of the SIGNAL design language offers formal support for the capture of behavioral abstractions for both very high-level system descriptions (e.g. SYSTEMC/SPECC) and behavioral-level IP components (e.g. VHDL). Its platform, POLYCHRONY, provides formal methods for a rapid, refinement-based, integration and a formal conformance-checking of GALS hardware/software architectures. In the aim of automating the present study within a versatile component integration platform, the use of refinement (model) checking tools directly provides the required support for automating this process by using controller synthesis techniques. Whereas model-checking consists of proving a property correct w.r.t. the specification of a system, controller synthesis consists of using this property as a control objective and to automatically generate a coercive process that wraps the initial specification so as to guarantee that the objective is an invariant.



Conclusion We have put a polychronous design model to work for the refinement of a high-level even-parity checker in SPECC from the early stages of its functional specification to the late stages of its hardware/software GALS implementation. We demonstrated the effectiveness of this approach by showing in what respects and at which critical design refinement stages formal verification and validation support was needed, highlighting the benefits of using the tool POLYCHRONY in that design chain. The novelty of integrating POLYCHRONY in a high-level design tool-chain lies in the formal support offered by the former to automate critical and complex design verification and validation stages yielding a correct-by-construction system design and refinement in the latter.

References

[1] F. DOUCET, M. OTSUKA, R. GUPTA AND S. SHUKLA "Efficient System Level Co-Design Environment using Split Level Programming". Technical Report 01-34, CECS/UCI, June 2001.

[2] D. GAJSKI, F. VAHID, S. NARAYAN, AND J. GONG. "Specification and Design of Embedded Systems". Prentice Hall, 1994.

[3] LE GUERNIC, P., TALPIN, J.-P., LE LANN, J.-L. Polychrony for system design. In *Journal of Circuits, Systems and Computers. Special Issue on Application Specific Hardware Design*. World Scientific, 2002.

[4] The BALBOA project. `http://www.ics.uci.edu/~balboa`

[5] The ESPRESSO project. `http://www.irisa.fr/espresso`