Causality and Scheduling Constraints in Heterogeneous Reactive Systems Modeling[⋆]

Albert Benveniste[1], Benoît Caillaud[1], Luca P. Carloni[2], Paul Caspi[3] and Alberto L. Sangiovanni-Vincentelli[2]

[1] Irisa/Inria, Campus de Beaulieu, 35042 Rennes cedex, France
Albert.Benveniste@irisa.fr
http://www.irisa.fr/sigma2/benveniste/
[2] U.C. Berkeley, Berkeley, CA 94720
{lcarloni,alberto}@eecs.berkeley.edu
http://www-cad.eecs.berkeley.edu/HomePages/{lcarloni,alberto}
[3] Verimag, Centre Equation, 2, rue de Vignate, F-38610 Gieres
Paul.Caspi@imag.fr
http://www.imag.fr/VERIMAG/PEOPLE/Paul.Caspi

Abstract. Recently we proposed a mathematical framework offering diverse models of computation and a formal foundation for correct-by-construction deployment of synchronous designs over distributed architecture (such as GALS or LTTA). In this paper, we extend our framework to model explicitly causality relations and scheduling constraints. We show how the formal results on the preservation of semantics hold also for these cases and we discuss the overall contribution in the context of previous work on desynchronization.

1 Introduction

Embedded systems are intrinsically heterogeneous since they are based on processors that see the world digitally and an environment that is analog. In addition, the processing elements are heterogeneous too since they often include micro-controllers and digital signal processors in addition to special purpose computing engines. These parts must communicate and exchange information in a consistent and reliable way over media that are often noisy and unreliable. Some of the tasks that embedded systems must carry out are safety-critical, e.g., in medical and transportation systems (cars, airplanes, trains) and for this reason have hard constraints on timing and reliability. As technology advances, increasingly more computing power becomes available thus offering the opportunity of adding functionality to the system to such an extent that the complexity of the design task is unmanegeable without a rigorous design methodology based on

[⋆] This research was supported in part by the European Commission under the projects COLUMBUS, IST-2002-38314, and ARTIST, IST-2001-34820, by the NSF under the project ITR (CCR-0225610), and by the GSRC.

sound principles. In particular, the need for fast time-to-market and reasonable development cost imposes design re-use. And design re-use implies the use of software for as many parts of the functionality as possible given size, production cost and power consumption constraints. Consequently, software accounts for most of the design costs today and it is responsible for delays in product delivery because of the lack of a unified design process that can guarantee correct behavior.

Today, designers face a very diversified landscape when it comes to methodologies, supporting tools, and engineering best practices. This would not necessarily be a problem if it were not for the fact that transitioning between tools that are based on different paradigms is increasingly becoming a design productivity bottleneck as it has been underlined by the road map work performed in the framework of the ARTIST network of excellence [3]. A solution to this problem would be to impose a "homogeneous" design policy, such as the fully synchronous approach. However, implementation costs in terms of performance and components require a more diversified view. Heterogeneity will manifest itself at the component level where different models of computation may be used to represent component operation and, more frequently, at different levels of abstraction, where, for example, a synchronous-language specification of the design may be refined into a globally asynchronous locally synchronous (GALS) architecture, thus alleviating some of the cost issues outlined above. Having a mathematical framework for the heterogeneous modeling of reactive systems gives freedom of choice between different synchronization policies at different stages of the design process and provides a solid foundation to handle formally communication and coordination among heterogeneous components. Interesting work along similar lines has been the Ptolemy project [13, 15], the MoBIES project [1], the Model-Integrated Computing (MIC) framework [16], and *Interface Theories* [14].

This paper is an extension of [7] where we proposed *Tagged Systems*, a variation of Lee and Sangiovanni-Vincentelli's (LSV) Tagged-Signal Model [22], as a mathematical model for heterogeneous systems. Originally, we restricted ourselves to Tagged Systems in which parallel composition is by intersection, meaning that unifiable events of each component must have identical variable, data, and tag. While this restriction has allowed us to handle GALS models of design, it does not cover all cases of interest. For example, causality relations and scheduling constraints are not compatible with parallel composition by intersection. Neither are earliest execution times. Yet causality and scheduling constraints are very important to include when implementing an embedded systems. Hence, it is sometimes useful to have a notion of parallel composition that accepts the unification of events having different tags (while the data that they carry must still be equal). In this work, we propose an extension of Tagged Systems where the unification rule for tags is itself parameterized. We show that this model captures important properties such as causality and scheduling constraints. Then, we extend the general theorems of [7] on the preservation of semantics during distributed deployment.

2 Tagged Systems and Heterogeneous Systems

2.1 Tagged Systems and Their Parallel Composition

Throughout this paper, $\mathbf{N} = \{1, 2, \ldots\}$ denotes the set of positive integers; $\mathbf{N}$ is equipped with its usual total order $\leq$. $X \mapsto Y$ denotes the set of all partial functions from X to Y. If $(X, \leq_X)$ and $(Y, \leq_Y)$ are partial orders, $f \in X \mapsto Y$ is called *increasing* if $f(\leq_X) \subseteq \leq_Y$, i.e., $\forall x, x' \in X : x \leq_X x' \Rightarrow f(x) \leq_Y f(x')$.

Tag Structures. A *tag structure* is a triple $(\mathcal{T}, \leq, \sqsubseteq)$, where $\mathcal{T}$ is a set of *tags,* and $\leq$ and $\sqsubseteq$ are two partial orders. Partial order $\leq$ relates tags seen as time stamps. Call a *clock* any increasing function $(\mathbf{N}, \leq) \mapsto (\mathcal{T}, \leq)$. Partial order $\sqsubseteq$, called the *unification order*, defines how to unify tags and is essential to express coordination among events. Write $\tau_1 \bowtie \tau_2$ to mean that there exists $\tau \in \mathcal{T}$ such that $\tau_i \sqsubseteq \tau$. We assume that any pair (τ_1, τ_2) of tags, such that $\tau_1 \bowtie \tau_2$ holds, possesses an upper bound. We denote it by $\tau_1 \sqcup \tau_2$. In other words, $(\mathcal{T}, \sqsubseteq)$ is a sup-semi-lattice. We call $\bowtie$ and $\sqcup$ the *unification relation* and *unification map,* respectively.

We assume that unification is causal with respect to partial order of time stamps: the result of the unification cannot occur prior in time than its constituents. Formally, if $\tau_1 \bowtie \tau_2$ is a unifiable pair then $\tau_i \leq (\tau_1 \sqcup \tau_2)$, for $i = 1, 2$. Equivalently:

$$\forall \tau, \tau' : \quad \tau \sqsubseteq \tau' \;\Rightarrow\; \tau \leq \tau'. \tag{1}$$

Condition (1) has the following consequence: if $\tau_1 \leq \tau_1'$, $\tau_2 \leq \tau_2'$, $\tau_1 \bowtie \tau_2$, and $\tau_1' \bowtie \tau_2'$ together hold, then $(\tau_1 \sqcup \tau_2) \leq (\tau_1' \sqcup \tau_2')$ must also hold. This ensures that the system obtained via parallel composition preserves the agreed order of its components.

Tagged Systems. Let $\mathcal{V}$ be an underlying set of variables with domain D. For $V \subset \mathcal{V}$ finite, a *V-behaviour,* or simply behaviour, is an element:

$$\sigma \;\in\; V \mapsto \mathbf{N} \mapsto (\mathcal{T} \times D), \tag{2}$$

meaning that, for each $v \in V$, the n-th occurrence of v in behaviour σ has tag $\tau \in \mathcal{T}$ and value $x \in D$. For v a variable, the map $\sigma(v) \in \mathbf{N} \mapsto (\mathcal{T} \times D)$ is called a *signal.* For σ a behaviour, an *event* of σ is a tuple $(v, n, \tau, x) \in V \times \mathbf{N} \times \mathcal{T} \times D$ such that $\sigma(v)(n) = (\tau, x)$. Thus we can regard behaviours as sets of events. We require that, for each $v \in V$, the first projection of the map $\sigma(v)$ (it is a map $\mathbf{N} \mapsto \mathcal{T}$) is increasing. Thus it is a clock and we call it the *clock of v in σ.* A *tagged system* is a triple $P = (V, \mathcal{T}, \Sigma)$, where V is a finite set of variables, $\mathcal{T}$ is a tag structure, and Σ a set of V-behaviours.

Homogeneous Parallel Composition. Consider two tagged systems $P_1 = (V_1, \mathcal{T}_1, \Sigma_1)$ and $P_2 = (V_2, \mathcal{T}_2, \Sigma_2)$ with identical tag structures $\mathcal{T}_1 = \mathcal{T}_2 = \mathcal{T}$. Let $\sqcup$ be the unification function of $\mathcal{T}$. For two events $e = (v, n, \tau, x)$ and $e' = (v', n', \tau', x')$, define

$$\begin{aligned} &e \bowtie e' \;\text{ iff }\; v = v', n = n', \tau \bowtie \tau', x = x', \text{ and} \\ &e \bowtie e' \;\Rightarrow\; e \sqcup e' =_{\text{def}} (v, n, \tau \sqcup \tau', x). \end{aligned}$$

The unification map $\sqcup$ and relation $\bowtie$ extend point-wise to behaviours. Then, for σ a V-behaviour and and σ' a V'-behaviour, define, by abuse of notation: $\sigma \bowtie \sigma'$ iff $\sigma_{|V \cap V'} \bowtie \sigma'_{|V \cap V'}$, and then

$$\sigma \sqcup \sigma' =_{\text{def}} \left(\sigma_{|V \cap V'} \sqcup \sigma'_{|V \cap V'}\right) \cup \sigma_{|V \setminus V'} \cup \sigma'_{|V' \setminus V}.$$

where $\sigma_{|W}$ denotes the restriction of behaviour σ to the variables of W. Finally, for Σ and Σ' two sets of behaviours, define their *conjunction*

$$\Sigma \wedge \Sigma' =_{\text{def}} \{\sigma \sqcup \sigma' \mid \sigma \in \Sigma, \sigma' \in \Sigma' \text{ and } \sigma \bowtie \sigma'\} \tag{3}$$

The *homogeneous parallel composition* of P_1 and P_2 is

$$P_1 \parallel P_2 =_{\text{def}} (V_1 \cup V_2, \mathcal{T}, \Sigma_1 \wedge \Sigma_2) \tag{4}$$

2.2 Theme and Variations on the Pair $(\mathcal{T}, \sqcup)$

Parallel Composition by Intersection. This is the usual case, already investigated in [7]. It corresponds to:

- The tag set $\mathcal{T}$ is arbitrary.
- The unification function $\sqcup$ is such that $\tau \bowtie \tau'$ iff $\tau = \tau'$, and $\tau \sqcup \tau' =_{\text{def}} \tau$.

Modeling synchronous systems, asynchronous systems, timed systems, with this framework, is extensively discussed in [7]. We summarize here the main points.

To represent *synchronous systems* with our model, take for $\mathcal{T}$ a totally ordered set (e.g., $\mathcal{T} = \mathbf{N}$), and require that all clocks are strictly increasing. The tag index set $\mathcal{T}$ organizes behaviours into successive reactions, as explained next. Call *reaction* a maximal set of events of σ with identical τ. Since clocks are strictly increasing, no two events of the same reaction can have the same variable. Regard a behaviour as a sequence of global reactions: $\sigma = \sigma_1, \sigma_2, \ldots$, with tags $\tau_1, \tau_2, \ldots \in \mathcal{T}$. Thus $\mathcal{T}$ provides a global, logical time basis.

As for *asynchronous systems,* we take a very liberal interpretation for them. If we interpret a tag set as a constraint on the coordination of different signals of a system and the integer $n \in \mathbf{N}$ as the basic constraint of the sequence of events of the behaviour of a variable, then the most "coordination unconstrained" system, the one with most degrees of freedom in terms of choice of coordination mechanism, could be considered an ideal asynchronous system. Then an asynchronous system corresponds to a model where the tag set does not give any information on the absolute or relative ordering of events. In more formal way, take $\mathcal{T} = \{.\}$, the trivial set consisting of a single element. Then, behaviours identify with elements $\sigma \in V \mapsto \mathbf{N} \mapsto D$.

Capturing Causality Relations and Scheduling Specifications. How can we capture causalities relations or scheduling specifications in the above introduced asynchronous tagged systems? The intent is, for example, to state that "the 2nd occurrence of x depends on the 3rd occurrence of b". Define $\mathbf{N}_0 =_{\text{def}} \mathbf{N} \cup \{0\}$. Define a *dependency* to be a map:

$$\delta = \mathcal{V} \mapsto \mathbf{N}_0.$$

We denote by Δ the set of all dependencies, and we take $\mathcal{T} = \Delta$. Thus an event has the form:

$$e = (v, n, \delta, x), \tag{5}$$

with the following interpretation: event e has v as associated variable, it is ranked n among the events with variable v, and it depends on the event of variable w that is ranked $\delta(w)$. The special case $\delta(w) = 0$ is interpreted as the absence of dependency. We take the convention that, for e as in (5), $\delta(v) = n - 1$. Thus, on $\sigma(v)$, the set of dependencies reproduces the ranking. Δ is equipped with the partial order defined by $\delta \leq \delta'$ iff $\forall v : \delta(v) \leq \delta'(v)$. Then we define the unification map $\sqcup$ for this case:

$$\mathbf{dom}\,(\sqcup) = \mathcal{T} \times \mathcal{T} \text{ and } \delta \sqcup \delta' =_{\text{def}} \max(\delta, \delta'). \tag{6}$$

With this definition, behaviours become labelled preorders as explained next. For σ a behaviour, and e, e' two events of σ, write:

$$e' \rightarrow_\sigma e \text{ iff } \begin{cases} \quad e = (v, n, \delta, x) \\ \quad e' = (v', n', \delta', x') \\ \delta(v') = n' \end{cases} \tag{7}$$

Note that, since $n' > 0$, the condition $\delta(v') = n'$ makes this dependency effective. Definition (7) makes σ a labeled directed graph. Denote by $\preceq_\sigma$ the transitive reflexive closure of $\rightarrow_\sigma$, it is a preorder[1].

Capturing Earliest Execution Times. Here we capture earliest timed executions of concurrent systems. Take $\mathcal{T} = \mathbf{R}_+$, the set of non-negative real numbers. Thus a tag $\tau \in \mathcal{T}$ assigns a date, and we define

$$\tau \sqcup \tau' =_{\text{def}} \max(\tau, \tau').$$

Hence $\sqcup$ is here a total function. Composing two systems has the effect that the two components wait for the latest date of occurrence for each shared variable. For example, assume that variable v is an output of P and an input of Q in $P \,\|\, Q$. Then the earliest possible date of every event of variable v in Q is by convention 0, whereas each event associated to v has a certain date of production in P. In the parallel composition $P \,\|\, Q$, the dates of production by P prevail.

[1] We insist: "preorder", not "partial order"—this should not be a surprise, since the superposition of two partial orders generally yields a preorder.

Capturing Timed Systems. Various classes of timed automata models have been proposed since their introduction by Alur and Dill in [2]. In timed automata, dates of events are subject to constraints of the form $C : \tau \in \cup_{i \in I} [\![s_i, t_i]\!]$, where I is some finite set whose cardinality depends on the considered event, and $[\![= [$ or $($, and symmetrically for $]\!]$. The classes of timed automata differ by the type of constraint that can be expressed, and therefore they differ in their decidability properties. Nevertheless, they can all be described by the following kind of Tagged System[2].

Take $\mathcal{T} = Pow(\mathbf{R}_+)$, where Pow denotes powerset. Thus, a tag $\tau \in \mathcal{T}$ assigns to each event a constraint on its possible dates of occurrence. Then, several definitions are of interest:

- $\tau_1 \bowtie \tau_2$ iff $\tau_1 \cap \tau_2 \neq \emptyset$, and $\tau_1 \sqcup \tau_2 = \tau_1 \cap \tau_2$. This is the straightforward definition, it consists in regarding tags as constraints and combining them by taking their conjunction.
- the unification of tags is a total function, defined as follows: $\tau_1 \sqcup \tau_2 = \{\max(t_1, t_2) | t_1 \in \tau_1, t_2 \in \tau_2\}$. In this case, events are synchronized by waiting for the latest one.

Hybrid Tags. Define the product $(\mathcal{T}, \sqcup) =_{\text{def}} (\mathcal{T}', \sqcup') \times (\mathcal{T}'', \sqcup'')$ in a standard way. This allows us to combine different tags into a compound, heterogeneous, tag. For instance, one can consider synchronous systems that are timed and enhanced with causality relations. Such systems can be "desynchronized", meaning that their reaction tag is erased, but their causality and time tags are kept.

2.3 Running Example

The Base Case: Synchronous Systems. Let P and Q be two synchronous systems involving the same set of variables: b of type boolean, and x of type integer. Each system possesses only a single behaviour, shown on the right hand side of $P : \ldots$ and $Q : \ldots$, respectively. Each behaviour consists of a sequence of successive reactions, separated by vertical bars. Each reaction consists of an assignment of values to a subset of the variables; a blank indicates the absence of the considered variable in the considered reaction.

$P :$	$b :$	t	f	t	f	t	f	$\ldots$
	$x :$	1		1		1		$\ldots$
$Q :$	$b :$	t	f	t	f	t	f	$\ldots$
	$x :$		1		1		1	$\ldots$

The single behavior of P can be expressed formally in our framework as

$$\begin{array}{lll} \sigma(b)(2n-1) = (2n-1, t) & , & \sigma(b)(2n) = (2n, f) \\ \sigma(x)(n) \quad\;\; = (2n-1, 1) & & \end{array} \tag{8}$$

[2] Our framework of Tagged Systems handles (infinite) behaviours and is not suited to investigate decidability properties, this explains why we can subsume all variants of timed automata into a unique Tagged Systems model.

where we take $\mathcal{T} = \mathbf{N}$ to index the successive reactions. Note the absence of x at tag $2n$. Similarly, for Q we have the following where x is absent at tag $2n-1$:

$$\begin{array}{rl} \sigma(b)(2n-1) = (2n-1, t) \;\;,\;\; \sigma(b)(2n) &= (2n, f) \\ \sigma(x)(n) &= (2n, 1) \end{array} \tag{9}$$

Now, the synchronous parallel composition of P and Q, defined by intersection: $P \parallel Q =_{\mathrm{def}} P \cap Q$, is empty. The reason is that P and Q disagree on where to put absences for the variable x. Formally, they disagree on their respective tags.

Desynchronizing the Base Case. The *desynchronization* of a synchronous system like P or Q consists in (i) removing the synchronization barriers separating the successive reactions, and, then, (ii) compressing the sequence of values for each variable, individually. This yields:

$$P_\alpha = Q_\alpha : \begin{array}{l} \hline b : t\ f\ t\ f\ t\ f\ \ldots \\ \hline x : 1\ 1\ 1\ \ .\ \ .\ \ . \\ \hline \end{array}$$

where the subscript $_\alpha$ refers to asynchrony. The reader may think that events having identical index for different variables are aligned, but this is not the case. In fact, as the absence of vertical bars in the diagram suggests, there is *no alignment* at all between events associated with different variables.

Formally, we express asynchrony by taking $\mathcal{T} = \{.\}$, the trivial set with a single element. The reason is that we do not need any additional time stamping information. Thus, the single behavior of $P_\alpha = Q_\alpha$ is written as

$$\sigma_\alpha(b)(2n-1) = t, \sigma_\alpha(b)(2n) = f, \text{ and } \sigma_\alpha(x)(n) = 1. \tag{10}$$

Regarding desynchronization, the following comments are in order. Note that $P \neq Q$ but $P_\alpha = Q_\alpha$. Next, the synchronous system R defined by $R = P \cup Q$, the nondeterministic choice between P and Q, possesses two behaviours. However, its desynchronization R_α equals P_α, and possesses only one behaviour. Now, since $P_\alpha = Q_\alpha$, then $P_\alpha \parallel Q_\alpha =_{\mathrm{def}} P_\alpha \cap Q_\alpha = P_\alpha = Q_\alpha \neq \emptyset$. Thus, for the pair (P, Q), desynchronization does not preserve the semantics of parallel composition, in any reasonable sense.

Adding Causality Relations. Suppose that some analysis of the considered program allows us to add the following causality relations to P and Q:

$$\begin{array}{cr|c|c|c|c|c|c|l} \cline{2-9} & b: & t & f & t & f & t & f & \ldots \\ P_c: & & \downarrow & & \downarrow & & \downarrow & & \ldots \\ & x: & 1 & & 1 & & 1 & & \ldots \\ \hline & b: & t & f & t & f & t & f & \ldots \\ Q_c: & & & \downarrow & & \downarrow & & \downarrow & \ldots \\ & x: & & 1 & & 1 & & 1 & \ldots \\ \cline{2-9} \end{array}$$

For example, in accordance to the above causality relations, the meaning of P could be: if $b = t$ then get x (and similarly for Q). By using the dependency relation defined in (6), we can express formally the behavior of P_c as

$$\begin{array}{lll} \sigma(b)(2n-1) = ([2n-1,(x,0)],t) & , & \sigma(b)(2n) = ([2n,(x,0)],f) \\ \sigma(x)(n) \quad\;\; = ([2n-1,(b,2n-1)],1) & & \end{array}$$

and the behavior of Q_c as

$$\begin{array}{lll} \sigma(b)(2n-1) = ([2n-1,(x,0)],t) & , & \sigma(b)(2n) = ([2n,(x,0)],f) \\ & , & \sigma(x)(n) \;\; = ([2n,(b,2n)],1) \end{array}$$

As for the base case and for the same reason, $P_c \parallel Q_c = \emptyset$.

Then, Desynchronizing. Removing the synchronization barriers from P_c and Q_c yields

$$\begin{array}{c|l} \hline & b: t\ f\ t\ f\ t\ f\ \ldots \\ P_c^\alpha : & \quad\;\downarrow\quad\downarrow\quad\downarrow\quad \ldots \\ & x: 1\quad 1\quad 1\quad \ldots \\ \hline \hline & b: t\ f\ t\ f\ t\ f\ \ldots \\ Q_c^\alpha : & \qquad\downarrow\quad\downarrow\quad\downarrow \ldots \\ & x: \quad 1\quad 1\quad 1 \ldots \\ \hline \end{array}$$

We insist that, again, desynchronizing consists in (i) removing the synchronization barriers, and then (ii) compressing the sequence of values for each variable, individually—this last step is not shown on the drawing, just because it is a lot easier to draw vertical arrows. Formally, for P_c^α we have

$$\begin{array}{lll} \sigma(b)(2n-1) = ((x,0),t) & , & \sigma(b)(2n) = ((x,0),f) \\ \sigma(x)(n) \quad\;\; = ((b,2n-1),1) & & \end{array}$$

and, for Q_c^α we have

$$\begin{array}{lll} \sigma(b)(2n-1) = ((x,0),t) & , & \sigma(b)(2n) = ((x,0),f) \\ & , & \sigma(x)(n) \;\; = ((b,2n),1) \end{array}$$

These two behaviours are unifiable and yield the dependency $(b, 2n)$, by the max rule (6). In fact, the reader can check that $P_c^\alpha \parallel Q_c^\alpha = Q_c^\alpha$. Thus P_c and Q_c did not include enough causality relations for desynchronization to properly preserve the semantics.

Adding More Causality Relations. Suppose that "oblique" causality relations are added, from each previous occurrence of x to the current occurrence of b:

$$\begin{array}{cc|c|r|c|r|c|r|l} \hline & b: & t & f & t & f & t & f & \ldots \\ P_{cc}: & & \downarrow & \nearrow & \downarrow & \nearrow & \downarrow & \nearrow & \ldots \\ & x: & 1 & & 1 & & 1 & & \ldots \\ \hline \end{array}$$

$$\begin{array}{cc|r|c|r|c|r|c|l} \hline & b: & t & f & t & f & t & f & \ldots \\ Q_{cc}: & & & \downarrow & \nearrow & \downarrow & \nearrow & \downarrow & \ldots \\ & x: & & 1 & & 1 & & 1 & \ldots \\ \hline \end{array}$$

These supplementary causality relations conform to the synchronous model since they agree with the increasing reaction index. Formally, the single behavior of P_{cc} is written

$$\begin{array}{lll} \sigma(b)(2n-1) = ([2n-1,(x,0)],t) & , & \sigma(b)(2n) = ([2n,(x,n)],f) \\ \sigma(x)(n) \qquad = ([2n-1,(b,2n-1)],1) & & \end{array}$$

and the one of Q_{cc} is

$$\begin{array}{lll} \sigma(b)(2n-1) = ([2n-1,(x,n-1)],t) & , & \sigma(b)(2n) = ([2n,(x,0)],f) \\ & , & \sigma(x)(n) \;\; = ([2n,(b,2n)],1) \end{array}$$

Again, $P_{cc} \,\|\, Q_{cc} = \emptyset$.

Then, Again Desynchronizing. Removing the synchronization barriers from P_{cc} and Q_{cc} yields

$$\begin{array}{rl} & \begin{array}{l} \hline b: t \quad f\ t \quad f\ t \quad f \ldots \\ \quad \downarrow \nearrow \ \downarrow \nearrow \ \downarrow \nearrow \ \ldots \\ x: 1 \quad\ \ 1 \quad\ \ 1 \quad\ \ \ldots \\ \hline \end{array} \\ P^{\alpha}_{cc}: & \\ & \begin{array}{l} \hline b: \quad t\ f \quad t\ f \quad t\ f \ldots \\ \qquad\quad \downarrow \nearrow \ \downarrow \nearrow \ \downarrow \ldots \\ x: \qquad 1 \quad\ \ 1 \quad\ \ 1 \ldots \\ \hline \end{array} \\ Q^{\alpha}_{cc}: & \end{array}$$

In our framework, for P^{α}_{cc} we have

$$\begin{array}{lll} \sigma(b)(2n-1) = ((x,0),t) & , & \sigma(b)(2n) = ((x,n),f) \\ \sigma(x)(n) \qquad = ((b,2n-1),1) & & \end{array}$$

and, for Q^{α}_{cc} we have

$$\begin{array}{lll} \sigma(b)(2n-1) = ((x,n-1),t) & , & \sigma(b)(2n) = ((x,0),f) \\ & , & \sigma(x)(n) \;\; = ((b,2n),1) \end{array}$$

However, now the composed behavior does not coincide with Q^{α}_{cc} but is

$$P^{\alpha}_{cc} \,\|\, Q^{\alpha}_{cc} = \begin{array}{l} \hline b: \quad t\ f \quad t\ f \quad t\ f \ldots \\ \qquad\quad \updownarrow \nearrow \ \updownarrow \nearrow \ \updownarrow \ldots \\ x: \qquad 1 \quad\ \ 1 \quad\ \ 1 \ldots \\ \hline \end{array}$$

The reason for the double causality between x and f-occurrences of b is that the n-th x causes the $(2n)$-th b (i.e. the n-th f-occurrence of b) in P_{cc} whereas the $(2n)$-th b causes the n-th x in Q_{cc}. Formally, by the max rule (6), the composed behavior of $P^{\alpha}_{cc} \,\|\, Q^{\alpha}_{cc}$ is written

$$\begin{array}{lll} \sigma(b)(2n-1) = ((x,n-1),t) & , & \sigma(b)(2n) = ((x,n),f) \\ & , & \sigma(x)(n) \;\; = ((b,2n),1) \end{array}$$

In conclusion, $P^{\alpha}_{cc} \,\|\, Q^{\alpha}_{cc}$ possesses causality loops and may be considered pathological and thus "rejected" in accordance with the original semantics $P \,\|\, Q = \emptyset$.

2.4 Heterogeneous Systems

Assume a functional system specification using a synchronous approach, for subsequent deployment over a distributed asynchronous architecture (synchronous and asynchronous are considered in the sense of subsection 2.1). When we deploy the design on a different architecture, we must make sure that the original intent of the designer is maintained. This step is non trivial because the information on what is considered correct behaviour is captured in the synchronous specifications that we want to relax in the first place. We introduce the notion of semantic-preserving transformation to identify precisely what is a correct deployment. We present the idea with our running example:

Running Example, Cont'd. Regarding semantics preserving deployment, the following comments can be stated on our running example. The synchronous parallel composition of P and Q, defined by intersection: $P \parallel Q =_{\text{def}} P \cap Q$, is empty. The reason is that P and Q disagree on where to put absences for the variable x. On the other hand, since $P_\alpha = Q_\alpha$, then $P_\alpha \parallel Q_\alpha =_{\text{def}} P_\alpha \cap Q_\alpha = P_\alpha = Q_\alpha \neq \emptyset$. Thus, for the pair (P, Q), desynchronization does not preserve the semantics of parallel composition, in any reasonable sense. $\diamond$

How to model that semantics is preserved when replacing the ideal synchronous broadcast by the actual asynchronous communication? An elegant solution was proposed by Le Guernic and Talpin for the former GALS case [21]. We cast their approach in the framework of tagged systems and we generalize it.

Tag Morphisms. For $\mathcal{T}, \mathcal{T}'$ two tag structures, call *morphism* a map $\rho : \mathcal{T} \mapsto \mathcal{T}'$ which is increasing w.r.t. $\leq$ and $\leq'$, surjective, and such that

$$\rho \circ \sqcup = \sqcup' \circ (\rho, \rho) \tag{11}$$

holds, where $\circ$ denotes the composition of functions. As expected from their name, morphisms compose. For $\rho : \mathcal{T} \mapsto \mathcal{T}'$ a morphism, and $\sigma \in V \mapsto \mathbf{N} \mapsto (\mathcal{T} \times D)$ a behaviour, replacing τ by $\rho(\tau)$ in σ defines a new behaviour having $\mathcal{T}'$ as tag set. We denote it by

$$\sigma_\rho, \text{ or by } \sigma \circ \rho. \tag{12}$$

Performing this for every behaviour of a tag system P yields the tag system

$$P_\rho. \tag{13}$$

For $\mathcal{T}_1 \xrightarrow{\rho_1} \mathcal{T} \xleftarrow{\rho_2} \mathcal{T}_2$ two morphisms, define:

$$\mathcal{T}_1 \;{}_{\rho_1}\!\times_{\rho_2}\; \mathcal{T}_2 =_{\text{def}} \{ (\tau_1, \tau_2) \mid \rho_1(\tau_1) = \rho_2(\tau_2) \}. \tag{14}$$

A case of interest is $\mathcal{T}_i = \mathcal{T}'_i \times \mathcal{T}, i = 1, 2$, and the $\mathcal{T}'_i$ are different. Then $\mathcal{T}_1 \;{}_{\rho_1}\!\times_{\rho_2}\; \mathcal{T}_2$ identifies with the product $\mathcal{T}'_1 \times \mathcal{T} \times \mathcal{T}'_2$. For example, the *desynchronization* of synchronous systems is captured by the morphism $\alpha : \mathcal{T}_{\text{synch}} \mapsto \{.\}$, which erases all global timing information (see Equations (8,9), and (10)).

Heterogeneous Parallel Composition. In this subsection we define the composition of two tagged systems $P_i = (V_i, \mathcal{T}_i, \Sigma_i), i = 1, 2$, when $\mathcal{T}_1 \neq \mathcal{T}_2$. Assume two morphisms $\mathcal{T}_1 \xrightarrow{\rho_1} \mathcal{T} \xleftarrow{\rho_2} \mathcal{T}_2$. Write:

$$\sigma_1 \ {}_{\rho_1}\!\bowtie_{\rho_2} \ \sigma_2 \ \text{ iff } \ \sigma_1 \circ \rho_1 \bowtie \sigma_2 \circ \rho_2. \tag{15}$$

For (σ_1, σ_2) a pair satisfying (15), define

$$\sigma_1 \ {}_{\rho_1}\!\sqcup_{\rho_2} \ \sigma_2 \tag{16}$$

as being the set of events $(v, n, (\tau_1, \tau_2), x)$ such that $\rho_1(\tau_1) = \rho_2(\tau_2) =_{\text{def}} \tau$ and (v, n, τ, x) is an event of $\sigma_1 \circ \rho_1 \sqcup \sigma_2 \circ \rho_2$. We are now ready to define the *heterogeneous conjunction* of Σ_1 and Σ_2 by:

$$\Sigma_1 \ {}_{\rho_1}\!\wedge_{\rho_2} \ \Sigma_2 =_{\text{def}} \{ \sigma_1 \ {}_{\rho_1}\!\sqcup_{\rho_2} \ \sigma_2 \mid \sigma_1 \ {}_{\rho_1}\!\bowtie_{\rho_2} \ \sigma_2 \} . \tag{17}$$

Finally, the *heterogeneous parallel composition* of P_1 and P_2 is defined by:

$$P_1 \ {}_{(\rho_1}\|_{\rho_2)} \ P_2 = (V_1 \cup V_2 , \mathcal{T}_1 \ {}_{\rho_1}\!\times_{\rho_2} \ \mathcal{T}_2 , \Sigma_1 \ {}_{\rho_1}\!\wedge_{\rho_2} \ \Sigma_2) . \tag{18}$$

We simply write ${}_{(\rho_1}\|$ instead of ${}_{(\rho_1}\|_{\rho_2)}$ when ρ_2 is the identity.

GALS, Hybrid Timed/Untimed Systems, and More. To model the interaction of a synchronous system with its asynchronous environment, take the heterogeneous composition $P_{(\alpha}\| \ A$, where $P = (V, \mathcal{T}_{\text{synch}}, \Sigma)$ is a synchronous system, $A = (W, \{.\}, \Sigma')$ is an asynchronous model of the environment, and $\alpha : \mathcal{T}_{\text{synch}} \mapsto \{.\}$ is the trivial morphism, mapping synchrony to asynchrony (hence the special notation).

For GALS, take $\mathcal{T}_1 = \mathcal{T}_2 = \mathcal{T}_{\text{synch}}$, where $\mathcal{T}_{\text{synch}}$ is the tag set of synchronous systems. Then, take $\mathcal{T} = \{.\}$ is the tag set of asynchronous ones. Take $\alpha : \mathcal{T}_{\text{synch}} \mapsto \{.\}$, the trivial morphism. And consider $P_1 \ {}_{(\alpha}\|_{\alpha)} \ P_2$.

For timed/untimed systems, consider $P_{(\rho}\| \ Q$, where $P = (V, \mathcal{T}_{\text{synch}} \times \mathcal{T}_\varphi, \Sigma)$ is a synchronous timed system, $Q = (W, \mathcal{T}_{\text{synch}}, \Sigma')$ is a synchronous but untimed system, and $\rho : \mathcal{T}_{\text{synch}} \times \mathcal{T}_\varphi \mapsto \mathcal{T}_{\text{synch}}$ is the projection morphism.

This machinery of morphisms provides a framework for the different manipulations of tags that were performed in Section 2.3 on our running example.

3 Application to Correct Deployment

In this section we apply our framework to the formalization of the practically important—but generally informal—requirement of "correct deployment".

3.1 Preserving Semantics: Formalization

We are given a pair $P_i = (V_i, \mathcal{T}_i, \Sigma_i), i = 1, 2$, such that $\mathcal{T}_1 = \mathcal{T}_2$, and a pair $\mathcal{T}_1 \xrightarrow{\rho} \mathcal{T} \xleftarrow{\rho} \mathcal{T}_2$ of identical morphisms. We can consider two semantics:

$$\begin{aligned} \textit{The strong semantics} &: \ P_1 \,\|\, P_2 \\ \textit{The weak semantics} &: \ P_1 \ {}_{(\rho}\|_{\rho)} \ P_2 . \end{aligned}$$

We say that ρ is *semantics preserving* with respect to $P_1 \,\|\, P_2$ if

$$P_1 \;{}_{(\rho}\|_{\rho)}\; P_2 \;\equiv\; P_1 \,\|\, P_2. \tag{19}$$

Running Example, Cont'd. The reader can check the following as an exercise: $P \,\|\, Q \;=\; \emptyset$, and, as we already discussed, $P_\alpha \,\|\, Q_\alpha \;=\; P_\alpha$. Now we compute $P \;{}_{(\alpha}\|_{\alpha)}\; Q$. From (16) we get that, using obvious notations, (σ_P, σ_Q) is a pair of behaviours that are unifiable modulo desynchronization, i.e., $\sigma_P \;{}_\alpha\!\bowtie_\alpha\; \sigma_Q$. Then, unifying these yields the behaviour σ such that:

$$\forall n \in \mathbf{N} : \sigma(b)(n) = ((n,n), v_b) \text{ and } \sigma(x)(n) = ((2n-1, 2n), 1) \tag{20}$$

where $v_b = t$ if n is odd, and $v_b = f$ if n is even. In (20), the expression for $\sigma(b)(n)$ reveals that desynchronizing causes no distortion of logical time for b, since (n,n) attaches the same tag to the two behaviours for unification. On the other hand, the expression for $\sigma(x)(n)$ reveals that desynchronizing actually causes distortion of logical time for x, since $(2n-1, 2n)$ attaches different tags to the two behaviours for unification. Thus $P \,\|\, Q = \emptyset$, but $P \;{}_{(\alpha}\|_{\alpha)}\; Q$ consists of the single behaviour defined in (20). Hence, $P \;{}_{(\alpha}\|_{\alpha)}\; Q \not\equiv P \,\|\, Q$ in this case: semantics is not preserved. $\diamond$

3.2 A General Result on Correct Deployment

Here we analyse requirement (19). The following theorem holds (see (13) for the notation P_ρ used in this theorem):

Theorem 1. *The pair (P_1, P_2) satisfies condition (19) if it satisfies the following two conditions:*

$$\forall i \in \{1,2\} : (P_i)_\rho \text{ is in bijection with } P_i \tag{21}$$

$$(P_1 \;\|\; P_2)_\rho \;=\; (P_1)_\rho \;\|\; (P_2)_\rho \tag{22}$$

Comments. The primary application of this general theorem is when P and Q are synchronous systems, and $\rho = \alpha$ is the desynchronization morphism. This formalizes GALS deployment. Thus, Theorem 1 provides sufficient conditions to ensure correct GALS deployment. Conditions (21) and (22) are not effective because they involve (infinite) behaviours. In [5, 6], for GALS deployment, condition (21) was shown equivalent to some condition called *endochrony,* expressed in terms of the transition relation, not in terms of behaviours. Similarly, condition (22) was shown equivalent to some condition called *isochrony,* expressed in terms of the pair of transition relations, not in terms of pairs of sets of behaviours. Endochrony and isochrony are model checkable and synthesizable, at least for synchronous programs involving only finite data types (see [5, 6] for a precise statement of these conditions).

Proof. Inclusion $\supseteq$ in (19) always hold, meaning that every pair of behaviours unifiable in the right hand side of (19) is also unifiable in the left hand side.

Thus it remains to show that, if the two conditions of Theorem 1 hold, then inclusion $\subseteq$ in (19) does too. Now, assume (21) and (22). Pick a pair (σ_1, σ_2) of behaviours which are unifiable in $P_1 \;{}_{(\rho}\|_{\rho)}\; P_2$. Then, by definition of ${}_{(\rho}\|_{\rho)}$, the pair $((\sigma_1)_\rho, (\sigma_2)_\rho)$ is unifiable in $(P_1)_\rho \,\|\, (P_2)_\rho$. Next, (22) guarantees that $(\sigma_1)_\rho \sqcup (\sigma_2)_\rho$ is a behaviour of $(P_1 \,\|\, P_2)_\rho$. Hence there must exist some pair (σ'_1, σ'_2) unifiable in $P_1 \;\|\; P_2$, such that $(\sigma'_1 \sqcup \sigma'_2)_\rho = (\sigma_1)_\rho \sqcup (\sigma_2)_\rho$. Using the same argument as before, we derive that $((\sigma'_1)_\rho, (\sigma'_2)_\rho)$ is also unifiable with respect to its associated (asynchronous) parallel composition, and $(\sigma'_1)_\rho \sqcup (\sigma'_2)_\rho = (\sigma_1)_\rho \sqcup (\sigma_2)_\rho$. But $(\sigma'_1)_\rho$ is the restriction of $(\sigma'_1)_\rho \sqcup (\sigma'_2)_\rho$ to its events labeled by variables belonging to V_1, and similarly for $(\sigma'_2)_\rho$. Thus $(\sigma'_i)_\rho = (\sigma_i)_\rho$ for $i = 1, 2$ follows. Finally, using (21), we know that if (σ'_1, σ'_2) is such that, for $i = 1, 2$: $(\sigma'_i)_\rho = (\sigma_i)_\rho$, then: $\sigma'_i = \sigma_i$. Hence (σ_1, σ_2) is unifiable in $P_1 \,\|\, P_2$. $\diamond$

Corollary 1. *Let P_1 and P_2 be synchronous systems whose behaviors are equipped with some equivalence relation $\sim$, and assume that P_1 and P_2 are closed with respect to $\sim$. Then, the pair (P_1, P_2) satisfies condition (19) if it satisfies the following two conditions:*

$$\forall i \in \{1, 2\} : (P_i)_\rho \text{ is in bijection with } P_i/\!\sim \tag{23}$$

$$(P_1 \;\|\; P_2)_\rho \;=\; (P_1)_\rho \;\|\; (P_2)_\rho \tag{24}$$

where $P_i/\!\sim$ is the quotient of P_i modulo $\sim$.

Proof. Identical to the proof of Theorem 1 until the paragraph starting with "Finally". Finally, using (23), we know that if (σ'_1, σ'_2) is such that, for $i = 1, 2$: $(\sigma'_i)_\rho = (\sigma_i)_\rho$, then: $\sigma'_i \sim \sigma_i$. Hence (σ_1, σ_2) is unifiable in $P_1 \,\|\, P_2$, since all synchronous systems we consider are closed under $\sim$. $\diamond$

This result is of particular interest when $\sim$ is the equivalence modulo stuttering, defined in [7].

Running Example, Cont'd. Since P and Q possess a single behaviour, they clearly satisfy condition (21). However, the alternative condition (22) is violated: the left hand side is empty, while the right hand side is not. This explains why semantics is not preserved by desynchronization, for this example. In fact, it can be shown that the pair (P, Q) is not isochronous in the sense of [5, 6].

More Examples and Counter-Examples. Our running example was a counter-example where condition (22) is violated.

For the following counter-example, condition (21) is violated: P emits to Q two signals with names x and y. Signal y is emitted by P if and only if $x > 0$ (assuming, say, x integer). Signals x and y are awaited by Q. Formally:

$$\begin{aligned} P &: \begin{cases} \sigma(x)(n) = (n, -) \\ \sigma(y)(n) = (m(n), -), \text{ where} \\ \quad m(n) = \min\{m \mid m > m(n-1) \wedge \sigma(x)(m) > 0\} \end{cases} \\ Q &: \begin{cases} \sigma(x)(n) = (k(n), -) \\ \sigma(y)(n) = (l(n), -) \end{cases} \end{aligned} \tag{25}$$

In (25), symbol $-$ denotes an arbitrary value in the domain D, and $k(.), l(.)$ are arbitrary strictly increasing maps, from $\mathbf{N}$ to $\mathbf{N}$. As the reader can check, P satisfies (21) but Q does not. The desynchronization α is not semantics preserving for this pair (P, Q).

Now, consider the following modification of (P, Q): P' emits to Q' two signals with names x and y. Signal y is emitted by P' if and only if $x > 0$ (assuming, say, x integer). In addition, P' emits a boolean guard b simultaneously with x, and b takes the value *true* iff $x > 0$. Signals x and y are awaited by Q'. In addition, Q' awaits the boolean guard b simultaneously with x, and Q' knows that he should receive y simultaneously with the *true* occurrences of b. Formally:

$$
\begin{array}{l}
P' : \left\{ \begin{array}{rl}
\sigma(x)(n) &= (n, -) \\
\sigma(b)(n) &= \text{if } \sigma(x)(n)(n) > 0 \text{ then } (n, t) \text{ else } (n, f) \\
\sigma(y)(n) &= (m(n), -), \text{ where} \\
m(n) &= \min\{m \mid m > m(n-1) \wedge \sigma(x)(m) > 0\}
\end{array} \right. \\
\\
Q' : \left\{ \begin{array}{rl}
\sigma(x)(n) &= (k(n), -) \\
\sigma(b)(n) &= (k(n), -) \\
\sigma(y)(n) &= (l(n), -), \text{ where} \\
l(n) &= \min\{k(m) \mid k(m) > l(n-1) \wedge \sigma(b)(m) = t\}
\end{array} \right.
\end{array}
\tag{26}
$$

The guard b explicitly says when y must be awaited by Q', this guarantees that Q' satisfies (21) (and so does P'). On the other hand, the pair (P', Q') satisfies (22). Thus the modified pair (P', Q') is semantics preserving, for desynchronization. The modification, from (P, Q) to (P', Q'), has been by adding the explicit guard b. This can be made systematic, as outlined in [6].

4 Discussion and Perspectives

In [11] the following result was proved. For P and Q two synchronous systems such that both P, Q, and $P \parallel Q$ are functional, clock-consistent, and with loop-free combinational part, then $P \parallel Q$ can be seen as a Kahn network—for our purpose, just interpret Kahn networks as functional asynchronous systems. This result applies to functional systems with inputs and outputs, it gives no help for partial designs or abstractions. Our conditions of endochrony and isochrony allows us to deal even with partial designs, not only with executable programs. Hence, they do represent effective techniques that can be used as part of the formal foundation for a successive-refinement design methodology.

As said before, this paper extends the ideas of [21] on desynchronization. A more naive "token-based" argument to explain GALS deployment is also found in [8], Sect. V.B. This argument is closely related to the use of Marked Graphs in [12] to justify GALS desynchronization in hardware.

Another example can be found in theory of latency-insensitive design [10]: here, if $P \parallel Q$ is a specification of a synchronous system and P and Q are *stallable* processes, then it is always possible to automatically derive two corresponding *patient* processes P_p and Q_p that seamlessly compose to give a system implementation $P_p \parallel Q_p$ that preserves semantics while being also robust to arbitrary, but discrete, latency variations between P and Q. Again, $P_p \parallel Q_p$ is a correct deterministic executable system made of endochronous sub-systems. In fact, as the notion of stallable system and patient system correspond respectively to the notion of stuttering-invariant system and endochronous system, Corollary 1 subsumes the result presented in [10] on the compositionality of latency-insensitivity among patient processes.

Now, the remaining key challenge is to make the theory of this paper effective. In this respect, Theorem 1 and its corollary are not enough, since they involve (infinite) behaviours. What is needed is sort of a counterpart of “automata” for our Tagged Systems. Synchronous Transition Systems as used in [6] are an example. Order automata are another example, that can be associated to Tagged Systems with causality relations. How to define such machines for general Tagged Systems is our next objective. Then, having this at hand, we will have to properly extend the “endochrony” and “isochrony” results of [6], thus providing effective algorithms to generate adaptors that ensure correct-by-construction deployment for general Tagged Systems.

References

1. R. Alur, T. Dang, J. Esposito, Y. Hur, F. Ivancic, V. Kumar, I. Lee, P. Mishra, G. J. Pappas and O. Sokolsky. Hierarchical Modeling and Analysis of Embedded Systems. *Proc. of the IEEE,* 91(1), 11–28, Jan. 2003.
2. R. Alur and D. L. Dill. A Theory of Timed Automata. *Theor. Comp. Science,* 126(2), 183–235, Apr. 1994.
3. ARTIST Network of Excellence. Roadmap on Hard Real-Time Development Environments. Available in may 2003 from url http://www.systemes-critiques.org/ARTIST/.
4. A. Benveniste. Some synchronization issues when designing embedded systems from components. In *Proc. of 1st Int. Workshop on Embedded Software, EMSOFT'01,* T.A. Henzinger and C.M. Kirsch Eds., LNCS 2211, 32–49, Springer.
5. A. Benveniste, B. Caillaud, and P. Le Guernic. From synchrony to asynchrony. In J.C.M. Baeten and S. Mauw, Eds., *CONCUR'99, Concurrency Theory, 10th Intl. Conference*, LNCS 1664, pages 162–177. Springer, Aug. 1999.
6. A. Benveniste, B. Caillaud, and P. Le Guernic. Compositionality in dataflow synchronous languages: specification & distributed code generation. *Information and Computation,* 163, 125-171 (2000).
7. A. Benveniste, L. P. Carloni, P. Caspi, and A. L. Sangiovanni-Vincentelli. Heterogeneous reactive systems modeling and correct-by-construction deployment. In R. Alur and I. Lee, Eds., *Proc. of the Third Intl. Conf. on Embedded Software, EMSOFT 2003*, LNCS 2855, Springer, 2003.
8. A. Benveniste, P. Caspi, S. Edwards, N. Halbwachs, P. Le Guernic, and R. de Simone. The Synchronous Language Twelve Years Later. *Proc. of the IEEE*, 91(1):64–83, January 2003.
9. G. Berry. The Foundations of Esterel. MIT Press, 2000.

10. L. P. Carloni, K. L. McMillan, and A. L. Sangiovanni-Vincentelli. Theory of Latency-Insensitive Design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(9):1059–1076, September 2001.
11. P. Caspi, "Clocks in Dataflow languages", *Theor. Comp. Science,* 94:125–140, 1992.
12. J. Cortadella, A. Kondratyev, L. Lavagno, and C. Sotiriou. A concurrent model for de-synchronization. In *Proc. Intl. Workshop on Logic Synthesis*, May 2003.
13. J. Eker, J.W. Janneck, E.A. Lee, J. Liu, J. Ludwig, S. Neuendorffer, S. Sachs, and Y. Xiong. Taming heterogeneity—The Ptolemy approach. *Proc. of the IEEE,* 91(1), 127–144, Jan. 2003.
14. L. de Alfaro and T.A. Henzinger. Interface Theories for Component-Based Design. In *Proc. of 1st Int. Workshop on Embedded Software, EMSOFT'01,* T.A. Henzinger and C.M. Kirsch Eds., LNCS 2211, 32–49, Springer, 2001.
15. E.A. Lee and Y. Xiong. System-Level Types for Component-Based Design. In *Proc. of 1st Int. Workshop on Embedded Software, EMSOFT'01,* T.A. Henzinger and C.M. Kirsch Eds., LNCS 2211, 32–49, Springer, 2001.
16. G. Karsai, J. Sztipanovits, A. Ledeczi, and T. Bapty. Model-Integrated Development of Embedded Software. *Proc. of the IEEE,* 91(1), 127–144, Jan. 2003.
17. N. Halbwachs, P. Caspi, P. Raymond, and D. Pilaud. The Synchronous Data Flow Programming Language LUSTRE. *Proc. of the IEEE*, 79(9):1305–1320, Sep. 1991.
18. D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, June 1987.
19. H. Kopetz, Real-Time Systems: Design Principles for Distributed Embedded Applications. Kluwer Academic Publishers. 1997. ISBN 0-7923-9894-7.
20. P. Le Guernic, T. Gautier, M. Le Borgne, and C. Le Maire. Programming real-time applications with SIGNAL. *Proc. of the IEEE*, 79(9):1326–1333, Sep. 1991.
21. P. Le Guernic, J.-P. Talpin, J.-C. Le Lann, Polychrony for system design. *Journal for Circuits, Systems and Computers.* World Scientific, April 2003.
22. E.A. Lee and A. Sangiovanni-Vincentelli. A Framework for Comparing Models of Computation. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems,* 17(12), 1217–1229, Dec. 1998.
23. M. Mokhtari and M. Marie. Engineering Applications of MATLAB 5.3 and SIMULINK 3. Springer, 2000.