14

Algebra of Communicating Timing Charts for Describing and Verifying Hardware Interfaces

Bachir Berkane, Simona Gandrabur, Eduard Cerny
Dép. d'IRO, Université de Montréal
Pavillon Andre-Aisenstadt, C.P. 6128, Succ. A
Montreal, Qc H3C 3J7, CANADA
Phone number: (514) 343 6111, Fax: (514) 343-5834
{berkane, gandrabu, cerny}@IRO.UMontreal.CA

Abstract

This paper addresses the specification and verification of hardware interface behaviors using an algebra of communicating timing charts (ACTC for short) whose underlying user model is the well known timing diagram. Terms modeling hierarchical timing charts are built in two steps. First, basic terms corresponding to the leaf charts are built, and then hierarchical terms are constructed using hierarchical operators. The basic terms model elementary behaviors as observed at the interface, and the hierarchical terms allow modeling behaviors at a higher level of abstraction: e.g., modeling an exception handling mechanism. We define an equivalence relation over the chart terms which is preserved by all the language constructs, and give a strategy to decide whether two deadlock-free basic terms are equivalent.

Keywords

Timing chart algebra, formal semantics, reactivity, equivalence

1 INTRODUCTION

We proposed in (Berkane *et al.* 1996) a structured language for describing the behavior in dense real time of digital electronic systems as seen from their external interfaces. In this paper, we develop a proof system for a subset of the language. The language is based on the user-friendly modeling methodology of Khordoc *et al.* (1993). The method consists of describing input/output behaviors recursively using annotated timing charts (TCs). Leaf TC templates are defined over sets of resources - ports: the loci of event sequences. The occurrence times of events are constrained using timing constraints. Hierarchical

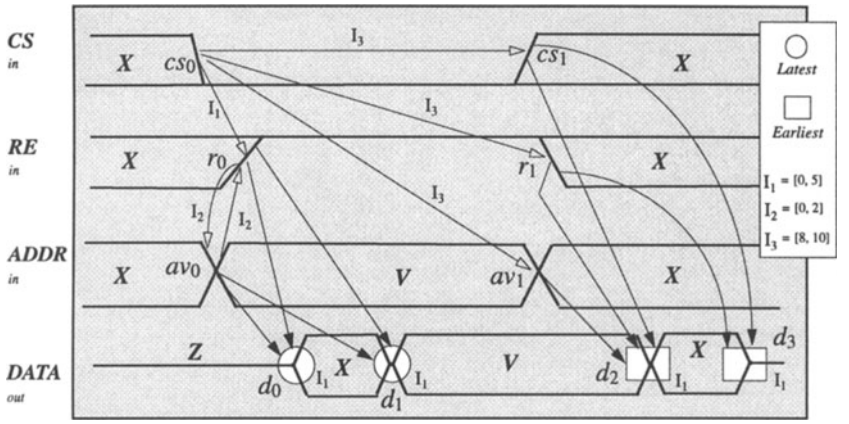

Figure 1 Timing Chart: An Example

charts can be formed using a number of composition operators. As an Example, Figure 1 depicts a leaf timing chart specifying an interface behavior*. The chart is defined over four ports: CS, RE and ADDR (input ports), and DATA (output port). Atomic events take place on each port (falling/rising signal transitions on control ports and transitions from {Valid (V), Don't care (X), High impedance (Z)} onto {V, X, Z} on data ports); the actions range over the vocabulary $\{cs_0, cs_1, r_0, r_1, av_0, av_1, d_0, d_1, d_2, d_4\}$. The occurrence times of these events are restricted using timing constraints, e.g., $cs_0 \xrightarrow{[0,5]} r_0$ states that the temporal distance between events cs_0 and r_0 must fall within the time interval [0, 5]. We distinguish two types of constraints:(1) assume constraints (empty arrowheads) and (2) reactive constraints (full arrowheads). Constraints of the assume intent are conjunctive and jointly define the occurrence time of the input events $\{cs_0, cs_1, r_0, r_1, av_0, av_1\}$. Reactive constraints define the occurrence time of the output events $\{d_0, d_1, d_2, d_4\}$ from the occurrence time of their source events. Reactive constraints are combined using the Latest and the Earliest constructs.

As in (Berkane *et al.* 1996) we adopt here the process algebra style "à la CCS (Milner 1989)" in the presentation of the language endowed with a structured operational semantics, in order to exploit the structural properties inherent in the charts. Within this process-algebra framework, timing charts are algebraic terms constructed using a few operators and a set of atomic events. Terms modeling hierarchical timing charts are built in two steps. First, basic terms corresponding to the leaf charts are built, and then hierarchical terms are constructed using hierarchical operators.

In (Berkane *et al.* 1996) we use the assumption construct of Klusener (1993) and the reactive operators (Latest and Earliest) to bound free occurrence of events to the resources of the interface. Assumption **cons** : *Chart* states

*The example is inspired by the read cycle of the NS100415 memory (NS Corp.)

that *Chart* must execute its events at the time instants that validate the condition *cons* which consists of a system of linear inequalities. The reactive constructs define the reaction of an action within a time interval *w.r.t.** a set of actions: e.g., in the example of Figure 1 if we assume that the timing chart without the reactive constraints is represented by the term *Chart* then **Latest**$(d_0, \{r_0, av_0\}, [m, M], Chart)$ describes the fact that the event d_0 will be executed within the interval $[m, M]$ after the execution of the last event in $\{r_0, av_0\}$. Restriction can be used to describe the timing assumptions about the environment under which the system described by means of the reactive constructs operates. Note that in recent years many works in the literature used assumption/reaction reasoning, e.g. (Abadi and Lamport 1990).

In this paper, the model of the leaf charts is based on the following principles:

- *An action is present on a port for a strictly positive amount of time.* Therefore, two actions cannot be present at the same time on the same port.

- *All the timing assumptions define a single environment.* We give Assumption the following semantics: "**cons**: *Chart*" executes the events of *Chart* under the condition that the time instants that validate *cons* validate also all the timing assumptions of *Chart*.

Also, we introduce a new built-in reactive construct that allows to describe in a uniform fashion the essential aspects of the Latest and Earliest constructs.

Next, using the operational semantics we define a compositional equivalence relation based on the bisimulation of Milner (1989). We give a verification algorithm to check whether or not a leaf chart may stop executing its actions and enter deadlock (signaling an inconsistent system of timing constraints). We then show that it is possible to establish whether or not two deadlock-free leaf chart terms are bisimilar which allows us to equate leaf charts that describe the same behavior.

The paper is organized as follows: Section 2 states the basic definitions and notation. Section 3 gives a syntax and a semantics to the language of leaf timing charts, and defines bisimulation equivalence. Section 4 gives the verification algorithms. Section 5 defines the hierarchical language. Section 6 compares the charts language with related languages. Section 7 concludes the presentation. Proof of theorems can be found in (Berkane *et al.* 1996[†]) and (Gandrabur 1997).

---

* "with respect to"

2 BASIC CONCEPTS AND NOTATION

- *Ports and actions*: Each timing chart is defined over a set of resources called ports, the loci of events (e.g., rising transitions of a signal); we distinguish input ports and output ports. Events occurring on an input port are called *input actions* and those occurring on an output port *output actions*. The execution of an action on port takes a certain time. Therefore, two actions cannot be executed at the same time instant on the same port. Let In, Out and Act be the input, output and action domains, respectively; $Act \subseteq In \cup Out$. Elements of In, Out and Act are denoted by i, o, a or b (sometimes indexed), respectively*.

- *Timing constraints*: A *timing constraint* is a Boolean expression. We let Θ be the constraint domain; its elements are defined by the following grammar:

$$cons := true \mid false \mid t_a - t_b \sim m \mid cons \wedge cons$$

where m is a rational or ∞, $\sim \in \{<, \leq\}$, and t_a is a time variable associated with the action a. We will use $m \leq t_a - t_b \leq M$ to represent $(t_b - t_a \leq -m) \wedge (t_a - t_b \leq M)$.

$Tvar(cons)$ and $act(cons)$ denote the time variables and the actions occurring in $cons$, respectively. The time variables domain is R_+ (non negative reals) and the time constants are denoted by $v, v_0, v_1, \ldots$ We denote by $cons_{v/t}$ the constraint $cons$ in which the variable t is replaced by v. A valuation of a constraint $cons$ consists of assigning a time constant v_i to each time variable t_i in $Tvar(cons)$. The set of all the assignment vectors that validate $cons$ is denoted by $Sol[cons]$.

Finally, we extend the domain Θ of timing constraints into Θ_+ whose elements are defined by the following grammar ($t \in Tvar(cons)$):

$$cons := cons \in \Theta \mid cons_{v/t} \mid cons \wedge t \sim v \mid cons \wedge -t \sim v$$

3 LANGUAGE ACTC$_\beta$ OF LEAF TIMING CHARTS

We present the chart language as an action-based timed process algebra where the timing charts behaviors are represented by algebraic expressions built from a few operators and a set of atomic actions. A *term* of the language is constructed in two steps. First, basic terms corresponding to the leaf charts are built using a set of basic constructs. Then, hierarchical terms are constructed using hierarchical operators. The definition of the hierarchical language is postponed till Section 5. Here we define the process algebra ACTC$_\beta$ of leaf timing charts with a structure similar to real time ACP (Klusener 1993).

We provide the formal meaning of ACTC$_\beta$ constructs by *labeled rooted*

*In the examples, we will use appropriate lower-case strings to refer to actions.

graphs constructed using of a set of action rules. The graph of a chart term consists of a possibly infinite set of *nodes* (syntaxic states represented by terms of the language), and a possibly infinite set of *labeled edges* (next-state relations) representing the effects of executing time-stamped actions.

Finally, we define an observational schema based on strong bisimulation equivalence of Milner (1989), to identify charts that describe the same behavior, and then show that this equivalence relation is the appropriate equivalence notion for ACTC_β in the sense that it is a congruence *w.r.t.* to all the operators of the language.

3.1 Syntax and intuitive semantics of ACTC_β

Let H be a set of input or output actions, and *cons* a constraint in $\Theta_+{}^*$. A leaf chart term *Chart* is constructed inductively as follows:

$$\begin{aligned} \mathit{Chart} ::= {} & \mathit{UntimedChart} \mid \mathbf{cons}\colon \mathit{Chart} \mid \mathbf{Latest}(H, o, [m, M], \mathit{Chart}) \\ & \mid \mathbf{Earliest}(H, o, [m, M], \mathit{Chart}) \mid \mathbf{Span}_{ctl}(H, o, [m, M], \mathit{Chart}) \end{aligned}$$

$$\mathit{UntimedChart} ::= \mathit{PortBehavior} \mid \mathbf{Par}(\mathit{UntimedChart}, \mathit{UntimedChart})$$

$$\mathit{PortBehavior} ::= \mathbf{nothing} \mid a\ \mathbf{then}\ \mathit{PortBehavior}$$

We assume that any action occurs at most once in *Chart*. This restriction does not limit the expressive power of the language needed to model the leaf timing charts. Indeed, if an action is needed twice to model a leaf chart: e.g., two rising transitions of the same signal, we use a different label to model the second occurrence of the action in the chart. Typical elements of ACTC_β are denoted by *Chart* or C (sometimes quoted or indexed). In the examples, we will use appropriate strings beginning with upper-case letter to refer to chart terms.

In the following we give intuitive semantics for the term forms of ACTC_β.

– **nothing** represents inaction.

– *Prefixing*: a **then** *Chart* executes the action a at any instant, and then evolves into *Chart* which is restricted to executing its actions only after the time instant v. Sometimes we will use "a" to mean "a **then nothing**".

– *Parallel composition*: $\mathbf{Par}(\mathit{Chart}_1, \mathit{Chart}_2)$ performs actions of Chart_1 and Chart_2 concurrently without any rendez-vous synchronization.

*The full domain will be used only in the operational semantics. Leaf charts are constructed using the sub-domain Θ.

– *Assumption* (Klusener 1993): "**cons** : *Chart*" binds *Chart* with the constraint *cons* in such way that the occurrence times of actions in *Chart* must satisfy the temporal restriction imposed by *cons*. If *cons* contradicts the timing information of *Chart*, then **cons**: *Chart* deadlocks.

– *Reactive constructs* (Latest, Earliest and Span): The reactive constructs bind the occurrence time of an output action o, called the sink action, with the occurrence times of a set of actions H (called the *source set*). We call the interval $[m, M]$ in Latest and Earliest the reactive interval. The scope of Span includes the constants m and M called respectively the minimum and maximum relative reactive times, and the variable ctl in $\{block, release\}$ called the *Span* control. The informal meaning of these constructs is as follows:

- **Latest**$(o, H, [m, M], Chart)$ delays the execution of the sink action o in *Chart* by some amount of time within the reactive interval $[mM]$ after the occurrence of the last action in the source action set H.

- **Earliest**$(o, H, [m, M], Chart)$ is similar to Latest except that the occurrence of the sink action o is relative to the earliest action in H.

- **Span**$_{ctl}(o, H, [m, M], Chart)$ forces the execution of the sink action s before the relative time instant M *w.r.t.* to the occurrence time of the latest action in the source set H. Furthermore, if $ctl = block$ the construct also delays the execution of the sink action after the time instant m *w.r.t.* to the occurrence time of the earliest action in the source set.

In Latest and Span$_{block}$ (*resp.* Earliest and Span$_{release}$) we refer to the latest (*resp.* the earliest) action in the source set as the source of the construct.

Example 1 *The ACTC$_\beta$ specification of the timing chart depicted in Figure* 1 *is given by the expression Read (Figure* 2*). As we shall see in Section* 4*, the order in which we apply the assumption and the reactive constructs has no effect on the behavior of a term.*

3.2 Operational semantics

We present here an operational semantics for the constructs of ACTC$_\beta$ that connect each term to a labeled rooted graph representing its behavioral interpretation.

$$
\begin{array}{lcl}
Cs & \stackrel{def}{=} & cs_0 \textbf{ then } cs_1 \textbf{then nothing} \\
Re & \stackrel{def}{=} & r_0 \textbf{ then } r_1 \textbf{then nothing} \\
Addr & \stackrel{def}{=} & av_0 \textbf{ then } av_1 \textbf{then nothing} \\
Data & \stackrel{def}{=} & d_0 \textbf{ then } d_1 \textbf{ then } d_2 \textbf{ then } d_3 \textbf{ then nothing} \\
UntimedChart & \stackrel{def}{=} & \textbf{Par}(Cs, Re, Addr, Data) \\
Chart_1 & \stackrel{def}{=} & \textbf{cons}: UntimedChart \\
cons & \stackrel{def}{=} & 0 \leq t_{r_0} - t_{cs_0} \leq 5 \ \wedge \ 8 \leq t_{cs_1} - t_{cs_0} \leq 10 \ \wedge \\
 & & 8 \leq t_{r_1} - t_{cs_0} \leq 10 \wedge \ 8 \leq t_{av_1} - t_{cs_0} \leq 10 \ \wedge \\
 & & -2 \leq t_{r_0} - t_{av_0} \leq 2 \\
Chart_2 & \stackrel{def}{=} & \textbf{Latest}(d_0, \{r_0, av_0\}, [0,5], Chart_1) \\
Chart_3 & \stackrel{def}{=} & \textbf{Latest}(d_1, \{r_0, av_0\}, [0,5], Chart_2) \\
Chart_4 & \stackrel{def}{=} & \textbf{Earliest}(d_2, \{cs_1, r_1, av_1\}, [0,5], Chart_3) \\
Read & \stackrel{def}{=} & \textbf{Earliest}(d_3, \{r_1, av_1\}, [0,5], Chart_4)
\end{array}
$$

Figure 2 The ACTC$_\beta$ term of the chart diagram of Figure 1

Definition 1 (*Labeled rooted graph LRG*) *A labeled rooted graph is the structure* $(\mathcal{N}, n_0, \mathcal{L}, \xi)$, *where* $\mathcal{N}$ *are the nodes,* n_0 *is the root,* $\mathcal{L}$ *is a set of labels, and* $\xi = \mathcal{N} \times \mathcal{L} \times \mathcal{N}$ *is the transition relation; elements of* ξ *are called transitions or edges.* □

Definition 2 (*Deterministic LRG*) *A labeled rooted graph* $(\mathcal{N}, n_0, \mathcal{L}, \xi)$ *is deterministic iff it satisfies the following requirement:*

$$\forall n, n', n'' \in \mathcal{N}, \forall l \in \mathcal{L}.\ (n, l, n') \in \xi \ \wedge \ (n, l, n'') \in \xi \implies n' = n'' \quad \square$$

Table 1 and 2 present the action rules for the timed constructs: Assumption, Latest and Earliest in the style of (Klusener 1993)*; the other rules can be found in (Berkane *et al.* 1996†). These rules associate to each term *Chart* a deterministic LRG $G(Chart) = (\mathcal{T}, Chart, Act \cup \{\delta\} \times R_+, \xi)$, where $\mathcal{T}$ is a set of ACTC$_\beta$ terms, δ is a special action (which is not in *Act*) that denotes a deadlock when executed, and ξ is defined by the action rules.

We associate with each term *Chart* a time constant $\alpha(Chart)$ denoting the time at which *Chart* starts its execution. The notation $Chart \stackrel{a(v)}{\rightarrow} Chart'$ represents an edge of $G(Chart)$ and denote that the term *Chart* executes a at $v \geq \alpha(Chart)$ and then behaves like *Chart'* with $\alpha(Chart') = v$. We assume that terms with starting times different from 0 are derived from the action rules.

*The rules are in the panth syntactic format (Verhoef 1994).

- **Assumption**

$$\mathbf{A}_1)\ \frac{Chart \overset{a(v)}{\rightarrow} Chart',\ First_v(a, \mathbf{cons}\colon Chart)}{\mathbf{cons}\colon Chart \overset{a(v)}{\rightarrow} \mathbf{cons}_{v/t_a}\colon Chart'}$$

$$\mathbf{A}_3)\ \frac{Chart \overset{\delta(v)}{\rightarrow} \mathbf{nothing}}{\mathbf{cons}\colon Chart \overset{\delta(v)}{\rightarrow} \mathbf{nothing}}$$

$$\mathbf{A}_2)\ \frac{Chart \overset{a(v)}{\rightarrow} \mathbf{nothing},\ First_v(a, \mathbf{cons}\colon Chart)}{\mathbf{cons}\colon Chart \overset{a(v)}{\rightarrow} \mathbf{nothing}}$$

$$\mathbf{A}_4)\ \frac{\forall a \in Act.\ \neg First_v(a, \mathbf{cons}\colon Chart))}{\mathbf{cons}\colon Chart \overset{\delta(\alpha)}{\rightarrow} \mathbf{nothing}},\quad \delta(\alpha) \equiv \delta(\alpha(Chart))$$

- **Latest**

$$\mathbf{L}_1)\ \frac{Chart \overset{a(v)}{\rightarrow} Chart',\ a \neq o, |H - \{a\}| \geq 1}{\mathbf{Latest}(o, H, [m, M], Chart) \overset{a(v)}{\rightarrow} \mathbf{Latest}(o, H - \{a\}, [m, M], Chart))}$$

$$\mathbf{L}_2)\ \frac{Chart \overset{a(v)}{\rightarrow} Chart'}{\mathbf{Latest}(o, \{a\}, [m, M], Chart) \overset{a(v)}{\rightarrow} \boldsymbol{m} \leq \boldsymbol{t}_o - \boldsymbol{v} \leq \boldsymbol{M}\colon Chart'}$$

$$\mathbf{L}_3)\ \frac{Chart \overset{a(v)}{\rightarrow} \mathbf{nothing},\ a \neq o}{\mathbf{Latest}(o, H, [m, M], Chart) \overset{a(v)}{\rightarrow} \mathbf{nothing}}$$

$$\mathbf{L}_4)\ \frac{\forall a, a \neq o, \forall v.\ Chart \overset{a(v)}{\not\rightarrow}}{\mathbf{Latest}(o, H, [m, M], Chart) \overset{\delta(\alpha)}{\rightarrow} \mathbf{nothing}}$$

$$\mathbf{L}_5)\ \frac{Chart \overset{\delta(v)}{\rightarrow} \mathbf{nothing}}{\mathbf{Latest}(o, H, [m, M], Chart) \overset{\delta(v)}{\rightarrow} \mathbf{nothing}}$$

Table 1 Action rules for the constructs Assumption and Latest

Predicate $Chart \overset{a(v)}{\not\rightarrow}$ expresses that for all terms $Chart'$ there is no edge $Chart \overset{a(v)}{\rightarrow} Chart'$ in $G(Chart)$, i.e., $Chart$ cannot execute a at time v. Finally, we need the predicate $First_v(a, Chart)$ which expresses that a can be executed at the time instant v before any other action in $Chart$. In the following we explain those rules that may need it.

Assumption: We explain Rules $\mathbf{A}_1$ and $\mathbf{A}_4$. Assume that $Chart \overset{a(v)}{\rightarrow} Chart'$. Rule $\mathbf{A}_1$ stipulates that Assumption remains active with the constraint $cons_{v/t_a}$ if a can be executed first in $\mathbf{cons}\colon Chart$. Rule $\mathbf{A}_4$ stipulates that if for all the actions that $Chart$ can execute the condition of Rule $\mathbf{A}_1$ is not satisfied, then Assumption deadlocks.

$$\mathbf{E}_1)\ \frac{Chart \xrightarrow{a(v)} Chart',\ a \neq o, a \notin H}{\mathbf{Earliest}(o, H, [m, M], Chart) \xrightarrow{a(v)} \mathbf{Earliest}(o, H, [m, M], Chart')}$$

$$\mathbf{E}_2)\ \frac{Chart \xrightarrow{b(v)} Chart',\ a \neq o, a \in H}{\mathbf{Earliest}(o, H, [m, M], Chart) \xrightarrow{a(v)} m \leq t_o - v \leq M : Chart'}$$

$$\mathbf{E}_3)\ \frac{Chart \xrightarrow{a(v)} \mathbf{nothing},\ a \neq o}{\mathbf{Earliest}(o, H, [m, M], Chart) \xrightarrow{a(v)} \mathbf{nothing}}$$

$$\mathbf{E}_4)\ \frac{\forall a, a \neq o, \forall v.\ Chart \overset{a(v)}{\not\rightarrow}}{\mathbf{Earliest}(o, H, [m, M], Chart) \xrightarrow{\delta(\alpha)} \mathbf{nothing}}$$

$$\mathbf{E}_5)\ \frac{Chart \xrightarrow{\delta(v)} \mathbf{nothing}}{\mathbf{Earliest}(o, H, [m, M], Chart) \xrightarrow{\delta(v)} \mathbf{nothing}}$$

Table 2 Action Rules for the construct Earliest

Latest: The first two rules correspond to the two actions that may be executed. If $Chart \xrightarrow{a(v)} Chart'$ with $a \neq o$ and $|H - \{a\}| \geq 1$ then we use Rule $\mathbf{L}_1$ to keep the operator active with the source set $H - \{a\}$. If $H = \{a\}$, Rule $\mathbf{L}_2$ is used to restrict $Chart'$ with the constraint $m \leq t_o - v \leq M$, i.e., o must be executed within the reactive interval. If $Chart$ can execute only the action o, Rule $\mathbf{L}_4$ stipulates that $\mathbf{Latest}(o, H, [m, M], Chart)$ blocks and signals a deadlock when it starts.

Earliest: Assume that $Chart \xrightarrow{a(v)} Chart'$ with $a \neq o$. If a is not in H, then Rule $\mathbf{E}_1$ is used to keep the operator active. If a is in H, Rule $\mathbf{E}_2$ is used to restrict $Chart'$ with the constraint $m \leq t_o - v \leq M$, i.e., o must be executed within the reactive interval. The remaining actions in H have no effect on the execution of o.

3.3 Equivalence

To identify terms that describe the same behavior, we define here an equivalence relation based on the strong bisimulation relation of Milner (Milner 1989), and state that this equivalence relation is a congruence with respect to all the operators of ACTC_β. To distinguish terms that may have different behaviors in the same assumption context, we introduce the mapping *ACons* that returns the assumption timing information of a term. *ACons* is defined recursively

on the terms of ACTC_β as follows ($act(Chart)$ are the actions of $Chart$):

$$
\begin{array}{lcl}
ACons(\textbf{nothing}) & = & true \\
ACons(a\ \textbf{then}\ Chart) & = & \bigwedge_{b \in act(Chart)} t_b > t_a \wedge ACons(Chart) \\
ACons(\textbf{Par}(Chart_1, Chart_2)) & = & ACons(Chart_1) \wedge ACons(Chart_2) \\
ACons(\textbf{cons}: Chart) & = & cons \wedge ACons(Chart) \\
ACons(\textbf{ROp}(Chart)) & = & ACons(Chart)
\end{array}
$$

Now we are ready to define the equivalence relation.

Definition 3 (*Bisimulation equivalence*) *Two $ACTC_\beta$ terms $Chart_1$ and $Chart_2$ are bisimilar, denoted by $Chart_1 \cong Chart_2$, if there exists a symmetric binary relation ρ on the terms of $ACTC_\beta$ such that $(Chart_1, Chart_2)$ in ρ and:*

1. If $C_1 \stackrel{a(v)}{\rightarrow} C_1' \wedge (C_1, C_2) \in \rho$ then $C_2 \stackrel{a(v)}{\rightarrow} C_2'$ for some $C_1' \wedge (C_1', C_2') \in \rho$
2. $Sol[Acons(C_1)] = Sol[Acons(C_2)] \wedge \alpha(C_1) = \alpha(C_2)$ □

The relation $\cong$ is context-independent in the sense that two bisimilar terms remain equivalent in all the contexts of the language. This then allows to capture the concept of substitution. The following theorem state this fact.

Theorem 1 (*Congruence*) *$\cong$ is a congruence with respect to all the operators of* ACTC_β. □

Proof. Using Milner's technique (Milner 1989). Another way to show that $\cong$ is a congruence is to use the result of Verhoef (1994). He claims that if the transition rules are in the panth syntactic format and stratifiables (a rule is stratifiable if the complexity of the conclusion is greater then the complexity of the premises) then bisimulation is a congruence. However, it is not clear to us how to interpret and apply his result in this case. ◇

4 VERIFICATION

Combining Assumption and reactive constructs in the specification of a timing chart may give rise to terms that can eventually enter a deadlock (i.e., stop executing actions and cannot proceed). We refer to such terms as *ill-reactive terms*, in contrast to *well-reactive terms* that terminate correctly for all their execution options. In this section we formulate a procedure for checking that a leaf chart term is well-reactive. Then, we develop a strategy to decide whether or not two well-reactive chart terms are equivalent. First, we define a term form of the basic language called *the intermediate form*, and state that each leaf chart term can be reduced to this form.

Definition 4 (*Intermediate form*) *Let Chart be a basic term constructed using n reactive operators with* $Act(Chart) = \{a_1, \ldots, a_m\}$. *Chart is an* intermediate form *if it has the following structure:*

$$\begin{array}{lcl} Chart & \stackrel{def}{=} & \mathbf{ROp}(o_1, H_1, [m_1 M_1], Chart_1) \\ \vdots & \vdots & \vdots \\ Chart_n & \stackrel{def}{=} & \mathbf{ROp}(o_n, H_n, [m_n M_n], Chart_{n+1}) \\ Chart_{n+1} & \stackrel{def}{=} & \mathbf{cons}\colon \mathbf{Par}(a_1, \ldots, a_m) \end{array}$$

where **ROp** *is a reactive operator. The subterm* $\mathbf{cons}\colon \mathbf{Par}(a_1, \ldots, a_m)$ *is called the kernel of Chart.* □

Proposition 1 *For each basic term Chart there exists an intermediate form* $\mathcal{IF}$ *such that* $Chart \cong \mathcal{IF}$. □

Proof. (Sketch) Using intermediate results that state: (1) Prefixing can be eliminated by introducing Assumption and Parallel, (2) the Assumption and the reactive constructs are commutative, and (3) the parameters of the assumption constructs can be regrouped. Then the result follows by construction. Recall that $\cong$ is a congruence, thus we can substitute equivalent terms.◇

Example 2 *The intermediate form of the term Read given in Figure* 2 *is obtained by substituting the subterm* $Chart_1$ *by the equivalent term:*

$$\begin{array}{lcl} Kernel & \stackrel{def}{=} & \mathbf{cons}\colon \mathbf{Par}(cs_0, r_0, av_0, d_0, cs_1, r_1, av_1, d_0, d_1, d_2, d_3) \\ cons & \stackrel{def}{=} & 0 \leq t_{r_0} - t_{cs_0} \leq 5 \wedge 8 \leq t_{cs_1} - t_{cs_0} \leq 10 \wedge 8 \leq t_{r_1} - t_{cs_0} \leq 10 \\ & & \wedge\ 8 \leq t_{av_1} - t_{cs_0} \leq 10 \ \wedge\ -2 \leq t_{r_0} - t_{av_0} \leq 2 \wedge\ t_{cs_0} < t_{cs_1} \\ & & \wedge\ t_{r_0} < t_{r_1} \ \wedge\ t_{av_0} < t_{av_1} \ \wedge\ t_{d_0} < t_{d_1} < t_{d_2} < t_{d_3} \end{array}$$

Kernel is obtained by eliminating the prefixing constructs and introducing Parallel and Assumption. Then, the constraints are regrouped in one assumption construct.

4.1 Reactivity

We formulate here an algorithm for checking that a leaf chart term in intermediate form is well-reactive, i.e. the chart is deadlock-free. Many works in the literature, e.g. (Walkup and Borriello 1994) proposed algorithms for determining time separation of events and consistency in timing interfaces modeled using a conjunctive system of linear, latest and/or earliest timing relations. These results can not be used to check the reactivity of the charts, since our chart specifications are based on assumption/reaction reasoning where the as-

Input: *Chart* in intermediate form built using the set *ROpset* of reactive constructs with $Act = act(Chart)$ and *cons* the constraint of Assumption.

- $First(Act, cons)$: The set of actions in Act that can fire first in *cons*.
- $\mathcal{F}(a, Act) \stackrel{def}{=} \bigwedge_{b \in Act} t_a \leq t_b$, - $t^\bullet$: Past time variable.
- $Reactive(ROpSet, a)$: The reactive constraints that links a with the sink actions of constructs in *ROpSet* for which a is the source.
- $Past(cons)$: Projection of *cons* on the past time variables.
- $Update(ROpSet, a)$: Update the set ROpSet *w.r.t.* the firing of a.

```
function Reactivity(ROpSet, cons, Act)
 begin
 if Sol[cons] = ∅ then returns (deadlock)
 else
   if ROpSet = ∅ then returns (well-reactive chart)
   else foreach a in First(a, Act) do
     cons1 = cons ∧ F(a, Act)
     (t_a ⟵ t_a^•) in cons1
     cons ⟵ cons1 ∧ Reactive(ROpSet, a)
     if Sol[cons] = ∅ or Sol[Past(cons)] ≠ Sol[Past(cons1)]
     then returns (deadlock) else
       ROpset ⟵ Update(ROpSet, a)
       Act ⟵ Act − {a}
       Reactivity(ROpSet, cons, Act) od
 end
```

Figure 3 The reactivity procedure

sumption linear timing relations *implies* the reaction latest, earliest and span timing relations.

Figure 3 presents the verification procedure. The procedure explores using on-the-fly transversal strategy a finite partition of the chart rooted graph, denoted $\mathcal{P}(G)$, *w.r.t.* the execution order of the chart actions. Each node in $\mathcal{P}(G)$ is a chart term in intermediate form. Possible successors of a node are obtained by executing the actions that can fire first, then by adding the reactive timing information if the fired action is the source of a reactive construct. The transversal of $\mathcal{P}(G)$ is stopped when all the nodes have been enumerated or when a deadlock is detected (either the constraint of Assumption of a node has no solution vector or it restricts the past timing information of the Assumption constraint of its source node). The number of the nodes in $\mathcal{P}(G)$ in the worse case is exponential *w.r.t.* the number of the chart actions. However in practice this parameter in leaf charts is small.

Example 3 *In Example of Figure* 1, *the reactivity procedure starts exploring from the root node represented by its ACTC intermediate form* (*Example* 2). *A finite partition of its rooted graph is built as follows: Since only* cs_0 *can be executed first, the root node has one possible deadlock-free successor* n_1. *Then by considering either* r_0 *or* av_0 *as the first action to fire,* n_1 *has two possible deadlock-free sucessors* n_2 *and* n_3. *From* n_2 *(resp.* n_3*) a source of the latest constructs fires and leads to a deadlock-free node* n_3 *(resp.* n_4*) since the constraint of Assumption of* n_3 *(resp.* n_4*) has solution vectors and the past timing information of the Assumption constraint in* n_2 *(resp.* n_3*) implies the one of* n_3 *(resp.* n_4*). The transversal of the graph is stopped when all the nodes have been enumerated. A node has no sucessor if its intermediate form has no reactive construct. For the example, all the enumerated nodes are deadlock-free. Thus, the chart of Figure* 1 *is well-reactive.*

4.2 Equivalence verification

We will give here a strategy to decide whether two well-reactive leaf-chart terms are equivalent. First, let us give the basic concepts needed in the development of the verification method.

- **ACTC$^{\oplus}_{\beta}$**: To eliminate the reactive constructs, we augment the basic language with the non deterministic choice operator. The terms of ACTC$^{\oplus}_{\beta}$ are defined by the following BNF grammar: $C := Chart_{\beta} \mid \mathbf{Choice}(C, C)$, where $\mathbf{Choice}(C_1, C_2)$ denotes the weak alternative choice between C_1 and C_2 such that the passing of time does not eliminate the possibility of the choice, and thus can lead to a deadlock.

- **Trace equivalence**: Let C be a ACTC$^{\oplus}_{\beta}$ term, and $G(C) = (\mathcal{T}, C, Act \cup \delta \times R_+, \xi)$ its associated LRG. We call *run* of C any sequence $a_1(v_0)a_2(v_1) \ldots a_n(v)$ that satisfy the following requirement:

$$\exists C_1, \ldots, C_n \in \mathcal{T}.\ C_{i-1} \xrightarrow{a_i(v_i)} C_i \in \xi \text{ and } v_i = \alpha(C_i) \text{ for all } i > 0.$$

The set of all runs of C is denoted by $\Sigma(C)$. Two terms C_1 and C_2 are trace equivalent, denoted by $C_1 \cong_{\sigma} C_2$, iff $\Sigma(C_1) = \Sigma(C_2)$.

- **Normal forms**: A term in ACTC$^{\oplus}_{\beta}$ is a *normal form* if it is of the form $Choice(C_1, \ldots, C_n)$, where $C_i, 1 \leq i \leq n$, is of the form $\mathbf{cons}_i$: $\mathbf{Par}(a^i_1, \ldots, a^i_{m_i})$.

Figure 4 presents the verification algorithm for checking if two charts terms $Chart_1$ and $Chart_2$ are bisimilar. It consists of rewriting the chart terms into their normal forms *w.r.t.* $\cong_{\sigma}$, $\mathcal{NF}_1 \stackrel{def}{=} \mathbf{Choice}(C^1_1, \ldots, C^1_{n_1})$ and $\mathcal{NF}_2 \stackrel{def}{=} \mathbf{Choice}(C^2_1, \ldots, C^2_{n_2})$. Then, the equivalence verification *w.r.t.* $\cong$ comes down

Input: well-reactive leaf Chart terms $Chart_1$ and $Chart_2$ with $act(Chart_1) = act(Chart_2)$

rewrite each term $Chart_\imath, \imath = 1, 2$ into its normal form $w.r.t \cong$ as follows:

1. Construct the intermediate form
2. Get rid of the reactive operators
3. Eliminate the deadlocks and redundancies

$\mathcal{NF}_\imath \stackrel{def}{=} \mathbf{Choice}(C^\imath_1, \ldots, C^\imath_{n_\imath})$

$C^\imath_j \stackrel{def}{=} \mathbf{cons}^\imath_j$: $\mathbf{Par}(a_1, \ldots, a_m)$,

if $Sol[\bigvee_j cons^1_j] = Sol[\bigvee_j cons^2_j]$ and

$Sol[Acons(Chart_1)] = Sol[Acons(Chart_2)]$

then returns $(Chart_1 \cong Chart_2)$

else returns $(Chart_1 \ncong Chart_2)$

Figure 4 Equivalence verification

to checking if the disjunction of the Assumption constraints of $C^1_\imath, 1 \leq \imath \leq n_1$, and the disjunction of the Assumption constraints of $C^2_\imath, 1 \leq \imath \leq n_2$ have the same solution vectors. Note that for deterministic transition systems trace equivalence and bisimulation are equivalent (Hirshfeld and Moller 1996).

The construction of the normal forms is based on the elimination of the reactive operators. Informally, the reactive constructs in a well-reactive term *Chart* in intermediate form behave in terms of linear executions as an *a priori* choice between a set of terms. Each term is constructed by restricting the kernel of *Chart* by an assumption construct parametrized by a constraint that imposes for each reactive construct the source of the sink action, and adds the timing information carried by the reactive construct: e.g. for Latest, the constraint imposes an action a as the last action in the source set to be executed and forces the execution of the sink action within the reactive interval *w.r.t.* to the time occurrence of a.

Finally, the normal forms are obtained by eliminating deadlocks and redundancies*.

Example 4 *Consider the well-reactive term Read (Figure 2) describing the timing chart of Figure 1. Let* $cons_{\sigma_\imath}, 1 \leq \imath \leq n$, *be the constraint that forces an execution order in the source sets of the reactive operators by choosing a source action for each construct. Notice that* $|H_L| \cdot |H_{E_1}| \cdot |H_{E_2}| = 12$, *where* H_L, H_{E_1} *and* H_{E_2} *are the source sets of the latest and earliest constructs. Therefore* $n = 12$. *Since all the 12 execution orders are possible, the normal form* $\mathcal{NF}$ *of Read is:*

*following the equations: $\mathbf{cons}: C \cong_\sigma (t_\delta = 0) : \delta$ if $sol[cons] = \emptyset$,
$\mathbf{Choice}(C, (t_\delta = 0) : \delta) \cong_\sigma C, \mathbf{Choice}(C, C) \cong_\sigma C$

$$
\begin{array}{lll}
\mathcal{NF} & \stackrel{def}{=} & \mathbf{Choice}(C_1, \ldots, C_{12}) \\
C_i & \stackrel{def}{=} & \mathbf{cons} \wedge \mathbf{cons}_{\sigma_i}\colon \mathbf{Par}(cs_0, r_0, \ldots, d_3) \\
cons & \stackrel{def}{=} & 0 \leq t_{r_0} - t_{cs_0} \leq 5 \ \wedge \ 8 \leq t_{cs_1} - t_{cs_0} \leq 10 \ \wedge \ 8 \leq t_{r_1} - t_{cs_0} \leq 10 \\
 & & \wedge\, 8 \leq t_{av_1} - t_{cs_0} \leq 10 \ \wedge \ -2 \leq t_{r_0} - t_{av_0} \leq 2 \wedge \ t_{cs_0} < t_{cs_1} \\
 & & \wedge\, t_{r_0} < t_{r_1} \ \wedge \ t_{av_0} < t_{av_1} \ \wedge \ t_{d_0} < t_{d_1} < t_{d_2} < t_{d_3}
\end{array}
$$

5 LANGUAGE OF HIERARCHICAL TIMING CHARTS

We define the ACTC language by adding a set of hierarchical operators to ACTC$_\beta$. Hierarchical chart terms are constructed using (1) the usual operators: *Time shift, Alternative choice, Sequential composition, Rendez-vous composition* and *Loop*. Rendez-vous synchronization in Rendez-vous composition supports the causality principle, that is, if two systems execute simultaneously an action on the same port, the occurrence time of this action is determined by the system that uses the shared port as output.

Also, we introduce two new operators: *Delayed Choice* and *Exception*. Delayed Choice joins the common input behaviors of two charts and delays the alternative choice between them, and Exception implements an exception handling mechanism. Examples of behavioral specification using these constructs can be found in (Khordoc and Cerny 1994).

To capture rendez-vous synchronization between actions, we use the binary communication mapping of ACP (Klusener 1993) (denoted "$\mid$") on the action domain $Act \cup \{\delta\}$ augmented by the special symbol "$\perp$" such that, for all a in $Act \cup \{\delta, \perp\}$, and for all o_1, o_2 in Out:

$$(a \mid b) \mid c = a \mid (b \mid c), a \mid b = b \mid a, \ a \mid \perp = \perp, \ a \mid \delta = \perp, \ o_1 \mid o_2 = \perp$$

Action $c = a|b$ is called a *communication action*. $a|b$ is $\perp$ if no rendez-vous synchronization between a and b is required. Finally, we let the letter f range over the renaming functions over Act.

5.1 Syntax and intuitive semantics

The definition of the hierarchical language and its informal meaning are as follows ($Chart_\beta$ is a well-reactive leaf chart term). The formal semantics of the hierarchical constructs can be found in (Berkane *et al.* 1996[†]).

$$
\begin{array}{l}
Chart ::= Chart_\beta \mid \mathbf{Shift}_v(Chart) \mid \mathbf{AShift}_v(Chart) \mid \mathbf{Seq}(Chart, Chart) \\
\quad \mid \mathbf{Choice}(Chart, Chart) \mid \mathbf{RComp}_{\mathcal{H}}(Chart, Chart) \mid \mathbf{Rel}_f(Chart) \\
\mid \mathbf{DChoice}(Chart, Chart) \mid \mathbf{Excep}(Chart, Chart, Chart) \mid \mathbf{Loop}(Chart)
\end{array}
$$

– $\mathbf{Shift}_v(Chart)$ and $\mathbf{AShift}_v(Chart)$ are respectively the *time shift* and the *absolute time shift* operators (Klusener 1993), (Beaten and Bergstra 1991). $\mathbf{Shift}_v(Chart)$ (*resp.* $\mathbf{AShift}_v(Chart)$) defines a chart term that starts executing its actions at the instant v (*resp.* after the instant v).

– $\mathbf{Seq}(Chart_1, Chart_2)$ denotes the sequential composition of $Chart_1$ and $Chart_2$; $Chart_2$ is initiated when $Chart_1$ terminates correctly.

– $\mathbf{Choice}(Chart_1, Chart_2)$ is the weak alternative choice between $Chart_1$ and $Chart_2$ already introduced in Section 4.2.

– The rendez-vous composition operator $\mathbf{RComp}_{\mathcal{H}}(Chart_1, Chart_2)$ is similar to **Par**, but forces the synchronization between $Chart_1$ and $Chart_2$ by performing an action from the set $\mathcal{H}$ of communication actions. Unlike the classical rendez-vous, when an output action communicates with a set of input actions, the occurrence time of the communication action is determinated by the output action. Note that $\mathbf{RComp}_{\emptyset} \stackrel{def}{=} \mathbf{Par}$.

– $\mathbf{Rel}_f(Chart)$ is a term in which the actions of $Chart$ are relabeled using the function f.

– The delayed choice $\mathbf{DChoice}(Chart_1, Chart_2)$ joins the common initial input actions of $Chart_1$ and $Chart_2$ and delays the selection of $Chart_1$ or $Chart_2$ until one of them executes a non-common action or a common action at a non-common time instant. The delayed choice also requires that the selection be made before one of the two terms performs an output action or terminates successfully. Otherwise, the construct deadlocks.

– The exception operator $\mathbf{Excep}(Chart, CondChart, ExcepChart)$ binds the term *Chart*, called the body of the Exception, with two chart terms: the condition term *CondChart*, that can contain only input actions, and the exception term *ExcepChart*. Intuitively while *Chart* and *CondChart* execute their actions concurrently, the operator joins their common initial input actions. If *CondChart* terminates before *Chart* or simultaneously with *Chart*, the exception term *ExcepChart* is initiated. Otherwise (*Chart* terminates before *CondChart*), the exception construct terminates.

– The Loop operator $\mathbf{Loop}(Chart)$ defines an infinite sequential composition of the term *Chart* with itself.

5.2 Hierarchical chart example

To illustrate a hierarchical chart description, consider an interface behavior specified by means of the chart term *Read* (described in Figure 2) and by the chart term *Halt* that specifies its behavior on the port *HALT*. *Halt* executes the input action *halt* and then becomes inactive. The execution of *Read* can be interrupted by the execution of *halt*, then *Read* is initiated again until it terminates successfully. This behavior is specified by the term *ExcepHandler* described below. We use the exception construct with *Halt* as the condition chart to interrupt the execution of *Read*, and the loop construct to restart its execution. We encapsulate this behavior in another exception construct which terminates when the last action d_3 of *Read* occurs.

$$
\begin{array}{lcl}
ExcepHandler & \stackrel{def}{=} & \mathbf{Excep}(\mathbf{Loop}(HaltRead), Condition, \mathbf{nothing}) \\
HaltRead & \stackrel{def}{=} & \mathbf{Excep}(Read, Halt, \mathbf{nothing}) \\
Halt & \stackrel{def}{=} & halt\ \mathbf{then}\ \mathbf{nothing} \\
Condition & \stackrel{def}{=} & d_3\ \mathbf{then}\ \mathbf{nothing}
\end{array}
$$

5.3 Composability

As for the basic language, we consider the bisimulation equivalence $\cong$ on hierarchical terms as the observational schema that captures the behavioral differences between hierarchical terms. $\cong$ is a congruence with respect to all the hierarchical constructs. This result implies that if $Chart_1 \cong Chart_2$, then the substitutions (in turn) of $Chart_1$ and $Chart_2$ into an hierarchical context give rise to two terms that are bisimilar as well.

Theorem 2 (*Congruence*) $\cong$ *is a congruence w.r.t. to all hierarchical constructs.* □

6 ACTC AND RELATED WORKS

We compare here ACTC with other formalisms of the literature.

- *Process algebra*: In recent years many works extended CCS-like process algebra to real time; see (Klusener 1993) for overview. The structure of ACTC is similar to the real time version of ACP presented by Fokkink and Klusener (1995) where a restricted version of assumption parameterized by a set a linear inequalities was introduced. The two main differences between ACTC and the algebra of Klusner *et al.* are: (1) ACTC supports the assumption/reaction reasoning, and (2) Assumption in ACTC can bind the occurrence time of an action with the occurrence times of actions that may occur in the future, which

is not allowed in real time ACP. Moreover, as far as we are aware, the temporal Delayed Choice and Exception constructs are available only in ACTC, required for modeling practical systems, e.g. (Khordoc and Cerny 1994). Note that Beaten and Mauw (1991) have defined independently an untimed version of Delayed choice.

- *Timing diagram dialects*: Other works proposed specification languages whose underlying model are timing diagrams. Borriello (1992) presented a formalization (in an informal way) of timing diagrams: a specification is a hierarchy of segments which consist of a collection of events and causal timing relations between them. Borriello uses the usual hierarchical constructs: Parallel, Choice, Sequential Composition and Loop. Lenk (1994) presented a graphical specification language and provided its formal semantics in terms of the calculus T-LOTOS (Quemada *et al.* 1989). In other words, any specification in the Lenk Language corresponds to a term of T-LOTOS.

The similarities between these languages and ACTC consist of the two-steps syntax of the specifications, introduced to make the modeling method akin to the one used by system designers. However, Borriello's and Lenk's languages are not based on the assumption/reaction philosophy needed in the specification of reactive systems such as hardware interfaces. Moreover, the expressive power of ACTC (with its hierarchical constructs Delayed choice, Exception and Composition that supports the causality principle) is greater then the one of other timing diagram dialects. Finally, we provide our language, unlike other timing diagrams languages, with a compositional form of structural operational semantics which allows to exploit the structured properties of the timing charts and to develop a proof system for the basic language.

- *Timed automata*: Timed automata (TAs) (Dill 1989) are finite automata enhanced with clocks. A branching-time logic interpreted over such automata has also been defined, e.g. (Daws *et al.* 1996)), including algorithms for real-time model checking. We presented in (Berkane *et al.* 1996) a translation method from ACTC into TAs, and demonstrated that the number of locations in a TA corresponding to a term *Chart* grows exponentially with the number of the *Chart* actions. The work presented in (Berkane *et al.* 1996) shows the need for an intermediate model (e.g., an extended version of TAs) that takes into account the structure of the charts.

7 CONCLUDING REMARKS

In this paper, we presented a formal language for describing hardware interfaces inspired by (Khordoc and Cerny 1994). We defined it in terms of a timed process algebra with a structure similar to real-time ACP (Klusener 1993). The behavior of an interface is described by means of an algebraic term. A hierarchical chart term is built in two steps. First, leaf chart terms are built

using Prefixing, Parallel composition and a few timed constructs to bound free occurrence of actions to the resources of the interface. Then, hierarchical charts are constructed using a set of hierarchical operators on well-reactive basic terms: *terms that terminate correctly for all their execution options.*

The formal meaning of a chart term is defined in terms of a labeled rooted graph. Using this graph, we define the equivalence relation $\cong$ based on strong bisimulation of Milner (1989), and state its compositionality. The decidability of $\cong$ has been proven only on well-reactive leaf chart terms. We are looking presently for a reduction strategy that preserves the relation $\cong$ and reduces hierarchical terms into canonical forms.

Work is underway to translate the chart specifications into timed automata (TAs) (Berkane *et al.* 1996). The advantage of translating a chart specification into a TA is that all the analysis techniques and tools already developed for the latter, e.g (Daws *et al.* 1996), can be used to verify timing properties of the former specifications. Finally, we have implemented a translator of chart specifications to VHDL. This allows to simulate chart descriptions on a countable subset of the time domain (e.g., a subset of the natural or the rational numbers) and includes functional descriptions using procedural annotations attached to actions. In this way one can visualize their behaviors and increase confidence in the correctness of the specification.

Acknowledgments: This work was partially supported by NSERC Canada Grant No. STR0167079, Micronet, and Nortel Ltd.

REFERENCES

Martin Abadi, Leslie Lamport. (1990) Composing Specifications. *Digital Equipment Corp., Report* #66.

Beaten, J.C.M. and Bergstra, J.A. (1991) Real-Time Process Algebra. *Journal of Formal Aspects of Computing.*, **3(2)**, 142-188.

Beaten, J.C.M. and Mauw, S. (1996) Delayed Choice: an Operator for Joining Message Sequence Charts, in *Proceedings of the 8rd Int'l Conference on Formal Description Techniques for Distributed Systems and Communication Protocols*, Berne, Switzerland.

Berkane, B., Gandrabur, S. and E. Cerny. (1996) Timing Diagrams: Semantics and timing analysis, in *Proceedings of the 3rd Asian-Pacific Conference on Hardware Description Languages*, Bangalore, India.

Berkane, B., Gandrabur, S. and E. Cerny. (1996†) ACTC: Algebra of Communicating Timing Charts. *U. of Montreal/DIRO, Technical Report.*

Borriello, G. (1992) Formalized Timing Diagrams, in *Proceedings of the European Conference on Design Automation*, Brussels, Belgium.

Daws, C., Olivero, A., Tripakis S. and Yovine S. (1995) The Tool Kronos. *Workshop on Hybrid Systems and Autonomous Control*, DIMACS.

Dill, D. (1989) Timing Assumptions and Verification of Finite-State Concur-

rent Systems. *Int'l Workshop on Automatic Verification Methods for Finite State Systems*, LNCS 407, Grenoble, France.

Fokkink, W. and Klusener, S. (1995) An Effective Axiomatization for Real Time ACP. *CWI, Technical Report # CS-R*9542.

Gandrabur, S. (1997) Une axiomatisation partielle d'une algèbre de chronogrammes communicants. *U. of Montreal/DIRO, Predoc Report.*

Hirshfeld, Y. and Moller, F. (1996) Decidability Results in Automata and Process Theory. *Logic for Concurrency: Structure vs. Automata*, (eds F. Moller, G. Birtwistle), LNCS 1043.

Khordoc, K., Dufresne, M., Cerny, E. *et al.* (1993) Integrating Behavior and Timing in Executable Specifications, in *Proceedings of the Int'l Conference on Hardware Description Languages*, Ottawa, Canada.

Khordoc, K. and Cerny, E. (1994) Modeling Cell Processing Hardware with Action Diagrams, in *Proceedings of the Int'l Symposium on Circuits And Systems*, Ottawa, Canada.

Klusener, A. S. (1993) Models and Axioms for a Fragment of Real Time Process Algebra. *Ph.D. Thesis*, CWI, Amsterdam.

Lenk, S. (1994) Extended Timing Diagrams as Specification Language, in *Proceedings of the European Conference on Design Automation (EURO-DAC)*, Grenoble, France.

Milner, R. (1989) *Communication and Concurrency.* Prentice International Series in Computer Science.

National Semiconductor Corp. ( ) 100415 1024 × 1-Bit RAM Data Sheet.

Quemada, J. *et al.* (1989) A Timed Calculus for LOTOS, in *Proceedings of the 2nd Int'l Conference on Formal Description Techniques for Distributed Systems and Communication Protocols*, Vancouver, Canada.

Verhoef, C. (1994) Congruence Theorem for Structured Operational Semantics, in *Proceedings of CONCUR'*94, LNCS #836.

Walkup, E. and Borriello, G. (1994) Interface Timing Verification with Application to Synthesis, in *Proceedings of the* 31*st DAC.*

BIOGRAPHY

o **Bachir Berkane** received his Ph.D. degree from *Polytechnique de Grenoble*, France in 1992. Since 1993 is within the LASSO research Unit of the Computer Science Department, University of Montreal. His research interests include formal specification, verification and synthesis of hardware systems.

o **Simona Gandrabur** is a Ph.D. student at University of Montreal, Computer Science Department. Her current research include formal characterization of interface specifications, real time process algebra and verification.

o **Eduard Cerny** is a Professor at the Department of Computer Science University of Montreal, head of the LASSO research Unit and the GRIAO center. He works in a number of areas related to hardware and computer systems: simulation, verification, timing analysis, synthesis, ...