Analysis of a Mixed-Signal Circuit in Hybrid Process Algebra ACP_{hs}^{srt}

K.L. Man and M.P. Schellekens *

Abstract— **ACP_{hs}^{srt} is a hybrid process algebra obtained by extending a combination of two existing extensions of Algebra of Communicating Processes (ACP), namely the process algebra with continuous relative timing and the process algebra with propositional signals, for the formal specification and analysis of hybrid systems. In addition to equational axioms, this hybrid process algebra has rules to derive equations with the help of real analysis. ACP_{hs}^{srt} is also closely related to the theory of hybrid automata, so ACP_{hs}^{srt} can be reasonable easily translated to hybrid automata. This enables automatic verification of ACP_{hs}^{srt} specifications using existing hybrid automaton-based verification tools (e.g. PHAVer and HyTech).**

Since ACP_{hs}^{srt} is a well-developed algebraic theory from the field of hybrid process algebras with the above-mentioned features, it seems to allow rigorous the formal specification and analysis of digital/analog/mixed-signal circuits. In order to explore this fact, in this paper, we study a mixed-signal circuit: a half wave rectifier in ACP_{hs}^{srt}. Some basic properties of the half wave rectifier are analysed in a formal way and some correctness requirements are also proven to be satisfied.

Keywords: ***hybrid process algebras, formal languages, formal semantics, formal specification, formal analysis, digital/analog/mixed-signal circuits***

1 Introduction

Process algebras [1] are formal languages in *Computer Science* that have formal syntax and semantics for specifying and reasoning about different systems. They are also useful tools for verification of various systems. Generally speaking, process algebras describe the behaviour of processes and provide operations that allow to compose systems in order to obtain more complex systems. Moreover, the analysis and verification of systems described using process algebras can be partially or completely carried out by mathematical proofs using equational theory. In simple words, process algebras are theoretical frameworks for the formal specification and analysis of the behaviour of various systems. Serious efforts [2, 3, 9, 11, 13, 15, 17, 19] have been made in the past to deal with various systems (e.g. discrete event systems, real-time systems and hybrid systems) in a process algebraic way. Over the years, process algebras have been successfully used in a wide range of problems and in practical applications in both academia and industry for analysis of many different systems.

*K.L. Man and M.P. Schellekens are with the Centre for Efficiency-Oriented Languages (CEOL), Department of Computer Science, University College Cork (UCC), Cork, Ireland, Email: pafesd@gmail.com, m.schellekens@cs.ucc.ie

Recently, through novel language constructs and well-defined formal semantics in a standard *Structured Operational Semantics* (SOS) style [8], several process algebras/calculi (Hybrid Chi [9, 10, 15], HyPA [11, 17], ACP_{hs}^{srt} [13], Behavioural Hybrid Process Calculus-BHPC [18, 19] and ϕ-Calculus [21]) have been developed for hybrid systems. Hybrid systems are systems that exhibit discrete and continuous behaviour. Such systems have proved fruitful in a great diversity of engineering application areas including air-traffic control, automated manufacturing, and chemical process control.

On the other hand, mathematically, the behaviour of analog circuits can be described by continuous variables and a set of differential equations, whereas discrete variables and switching-modes are also used to model mixed-signal circuits. In simple words, analog and mixed-signal circuits are hybrid systems by nature.

Several attempts [4, 5, 7] have been made over the last two years to apply hybrid process algebras in the context of the formal specification of analog and mixed-signal circuits. However, to the best of our knowledge, no case study of the formal analysis of analog/mixed-signal circuits described in hybrid process algebras has been reported yet to really show that it is possible to analyse analog/mixed-signal circuits (in general) in hybrid process algebras. Hence, in this paper, a mixed-signal circuit: a half wave rectifier circuit is studied. We first give the formal specification of the half wave rectifier circuit in hybrid process algebra ACP_{hs}^{srt}. Then some properties of such a circuit are analysed in a formal way and some correctness requirements are also proven to be satisfied.

ACP_{hs}^{srt} is a hybrid process algebra obtained by extending a combination of two existing extensions of Algebra of Communicating Processes (ACP) [1], namely the process algebra with continuous relative timing and the process algebra with propositional signals (see [13] for details), for the formal specification and analysis of hybrid systems. In addition to equational axioms, this hybrid process algebra has rules to derive equations with the help of real analysis. ACP_{hs}^{srt} is also closely related to the theory of hybrid automata [24, 28] (see also [14] for details), so ACP_{hs}^{srt} can be reasonable easily translated to hybrid automata. This enables automatic verification of ACP_{hs}^{srt} specifications using existing hybrid automaton-based verification tools (e.g. PHAVer [26] and HyTech [28]). In this paper, for illustration purpose, we choose the hybrid process algebra ACP_{hs}^{srt} as the main reference process algebra for hybrid systems. This particular choice is immaterial and other hybrid process algebras/calculi may be used as well.

1.1 Related works

Some comparisons and related works of the hybrid process algebras/calculi (Hybrid Chi, HyPA, $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$, BHPC and ϕ-Calculus) can already be found in [13, 15, 17, 19, 23]. HyPA was shown to be useful for the formal specification of hardware and analog circuits in [4]. Also, as reported in [5], $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ could reasonably and effectively be used to give formal specifications of mixed-signal circuits. Note that this paper is the extended and revised version paper of [6].

1.2 Outline

This paper is organised as follows. Section 2 provides a brief overview of the hybrid process algebras $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$. A sample (modelling a half wave rectifier circuit) of the application of $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ is shown in Section 3. A variety of approaches that can be used for the formal analysis of specifications described in hybrid process algebras $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ is discussed in Section 4. The model checker PHAVer and the formalism of hybrid I/O-automata [25, 26] are presented in Section 5. Formal analysis of the half wave rectifier circuit specification in $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ using the model checker PHAVer is given in Section 6. To gain confidence of the analysis results presented in Section 6, the half wave rectifier circuit is further modelled using the Modelica language [31] and analysed in the OpenModelica System [32] environment. Finally, concluding remarks and future works can be found in Section 8.

2 Hybrid Process Algebra $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$

In this section, we give a (very short) introduction of a small subset (that is relevant for this paper) of the hybrid process algebra $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ proposed in [13]. For an extensive treatment of $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$, the reader is referred to [13]. $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ is defined according to the following grammar for process P:

$$\begin{aligned} P = \;& \tilde{\tilde{a}} \mid \sigma_{\mathrm{rel}}^{r}(P) \mid P_1 \cdot P_2 \mid P_1 \parallel P_2 \\ & \mid \psi \,\text{◭}\, P \mid \psi :\!\rightarrow P \mid \phi \,\text{◸}_V\, P \mid \chi \,\text{◸}\, P \end{aligned}$$

In $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$, the atomic processes are undelayable actions $\tilde{\tilde{a}}$. The basic way of timing processes is relative delay. Let P be a process and r be a non-negative unit of time, then $\sigma_{\mathrm{rel}}^{r}(P)$ denotes that the process idles for a period of time r and then behaves like P.

The basic ways of combining processes are sequential composition and parallel composition. Let P_1 and P_2 be processes. The sequential composition of P_1 and P_2, written as $P_1 \cdot P_2$, is the process that behaves as P_1 until P_1 terminates, and then continues to behave as process P_2. Then parallel composition of P_1 and P_2, written as $P_1 \parallel P_2$, is the process proceeds with P_1 and P_2 in parallel with the possibility of communication between processes P_1 and P_2.

Furthermore, signal emission operator and conditional proceeding operator are used to deal with propositions. Let ψ be a proposition. Then P emitting signal ψ, written as $\psi \,\text{◭}\, P$, is the process that behaves like P and moreover emits signal ψ. P proceeding conditionally on ψ, written as $\psi :\!\rightarrow P$, is the process that behaves like P if proposition ψ holds at its start, and otherwise behaves like an undelayable deadlock.

In order to deal with continuous state evolutions and instantaneous state transitions, the signal evolution operator and signal transition operator are defined. Let V, ϕ and χ be a set of state variables, a state proposition and a transition proposition respectively. Then P in evolution according to ϕ with V smooth, written as $\phi \,\text{◸}_V\, P$, is the process P of which the emitted signal changes continuously till it performs its first action in such a way that ϕ is satisfied and without discontinuities for state variables in V. Then P in transition according to χ, written $\chi \,\text{◸}\, P$, is the process P of which the emitted signal changes instantaneously over performing its first action in such a way that χ is satisfied.

$\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ has an operational semantics in a SOS style (see [13] for more details) that associates a hybrid transition system with a process and a valuation of environment variables (i.e. a state). Five kinds of transition relations are defined:

1. termination step;
2. action step;
3. time step;
4. signal relations;
5. discontinuity relations.

3 A Half Wave Rectifier Circuit

This section is a sample of the application of $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$. We give a first impression of how one describes the behaviour of a mixed-signal circuit (revised from [5]) using hybrid process algebra $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$. Figure 1 shows the half wave rectifier circuit. It consists of an ideal diode D, two resistors with resistance R_0 and R_1, respectively, a capacitor with capacity C_0, a voltage source with voltage v_0 and a ground voltage v_G. The specification is a parallel composition (of processes) of all the above-mentioned components of the half wave rectifier circuit. In the specification, symbols C_0, R_0 and R_1 denote constants.

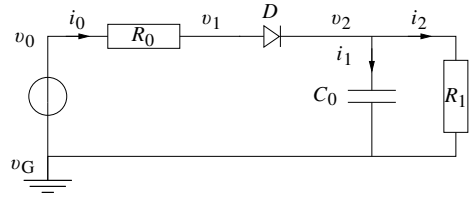

Figure 1: Half wave rectifier circuit.

An ideal diode can either be in the *on* mode (i.e. conducts the current) or *off* mode. When it is in the *off* mode, the diode

voltage must be smaller than or equalled to zero and the diode current equals to zero. When it is in the *on* mode, the diode voltage equals to zero and the diode current must be greater than or equalled to zero. The switching-mode behaviour of the ideal diode (i.e. the process *diode*) can be formally described using $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ by the following recursive equations:

$$\begin{aligned} D^{\mathrm{off}} &= (v_2 \geq v_1 \wedge i_0 = 0) \trianglerighteq_V \sigma^*_{\mathrm{rel}}((i_0 \geq 0) : \rightarrow \\ &\quad ((i_0{}^\bullet = {}^\bullet i_0) \trianglerighteq \widetilde{\mathrm{D}_{\mathrm{t-on}}} \cdot D^{\mathrm{on}})), \\ D^{\mathrm{on}} &= (v_1 = v_2 \wedge i_0 \geq 0) \trianglerighteq_V \sigma^*_{\mathrm{rel}}((v_2 \geq v_1) : \rightarrow \\ &\quad ((v_1{}^\bullet = {}^\bullet v_1 \wedge v_2{}^\bullet = {}^\bullet v_2) \trianglerighteq \widetilde{\mathrm{D}_{\mathrm{t-off}}} \cdot D^{\mathrm{off}})). \end{aligned}$$

The notation $\sigma^*_{\mathrm{rel}}(t)$ is to be read as the relative delay of t for an arbitrary period of time and $V = \{i_0, i_1, i_2, v_G, v_0, v_1, v_2\}$. Two unimportant undelayable actions $\widetilde{\mathrm{D}_{\mathrm{t-on}}}$ and $\widetilde{\mathrm{D}_{\mathrm{t-off}}}$ are purely introduced to satisfy the syntax required by $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$. Also some predicates (e.g. $i_0{}^\bullet = {}^\bullet i_0$) are needed to make precise that variables do not change during an instantaneous state transition. Note that $x^\bullet$ and ${}^\bullet x$ denote the value of variable x before and after (respectively) execution of an instantaneous state transition.

Applying *algebraic reasoning* (see Section 4.1 for details), we can obtain a simpler specification which is bisimilar (i.e. equivalent to) the specification of the parallel composition of the processes of all components of the half wave rectifier circuit excluding the ideal diode as follows:

$$\begin{aligned} P &= ((v_0 = F_{\mathrm{time}}) \parallel ((v_0 - v_1 = i_0 R_0) \wedge \\ &\quad (C_0(\dot{v_2} - \dot{v_G}) = i_1) \wedge (v_2 - v_G = i_2 R_1) \\ &\quad \wedge (v_G = 0) \wedge (i_0 = i_1 + i_2))) \trianglerighteq_\emptyset \sigma^*_{\mathrm{rel}}(\widetilde{\delta}). \end{aligned}$$

F_{time} is an arbitrary function of time and $\widetilde{\delta}$ represents an undelayable deadlock. For readability (see also Section 6), we write $v_0 = F_{\mathrm{time}}$ as process *generator* and $((v_0 - v_1 = i_0 R_0) \wedge (C_0(\dot{v_2} - \dot{v_G}) = i_1) \wedge (v_2 - v_G = i_2 R_1) \wedge (v_G = 0) \wedge (i_0 = i_1 + i_2))$ as process *rectifier* respectively.

Initially, the diode is in the *off* mode. The behaviour of the half wave rectifier circuit can be formally described using $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ as follows:

$$\mathrm{half\_wave} = \left((v_1 = v_2) \trianglerighteq D^{\mathrm{off}}\right) \parallel P.$$

4 Formal Analysis of $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ Specifications

This section presents several formal approaches that can be used for the analysis of specifications described in $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$.

4.1 Algebraic reasoning

The strength of the field of process algebras lies on the ability to use algebraic reasoning (also known as equational reasoning) that allows rewriting specifications using axioms (e.g. for commutativity and associativity) to a simpler form. This can be advantageous for many forms of analysis.

$\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ has a comprehensive set of axioms and lifting rules which allow results from real analysis to be lifted to equations about processes. Using the axioms and lifting rules of $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$, various non-trivial case studies of hybrid systems were already analysed in [13, 23].

4.2 Relating $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ to the theory of hybrid automata

The theory of hybrid automata [24, 28] is the most well-known formalism for modelling and analysis of hybrid phenomena. Over the past years, various formal techniques (e.g. model checking based on reachability analysis [26, 28]) and tools (e.g. HyTech and PHAVer) have been successfully introduced into the analysis of hybrid systems modelled as hybrid automata.

Since $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ and the framework of hybrid automata both adopt the view that a hybrid system in which an instantaneous state transition occurs on the system performing an action and a continuous state evolution occurs on the system idling between performing successive actions. This makes the formal translation from $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ to the theory of hybrid automata possible and straightforward.

Intuitively, another possibility to analyse $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ specifications is to translate them formally into the corresponding hybrid automaton models that are the input languages of existing formal verification tools (hybrid automaton-based). For instance, we can use the model checker PHAVer as a verification engine for $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ specifications, by translating them formally to the theory of hybrid I/O-automata[1] [25, 26] that is the input language of PHAVer.

4.3 Analysis of the half wave rectifier circuit

Using the axioms and lifting rules of $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ to analyse $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ specifications may not be intuitive to those who have not a strong Computer Science background, because a lot of rewriting of the specifications (based on axioms) and equational reasoning have to be made. Hence, in this paper, we intend to analyse the half wave rectifier circuit in $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ using the model checker PHAVer via the translation of such a specification to the corresponding hybrid I/O automaton model that is the input format of PHAVer. We believe that this analysis approach is more suitable and intuitive for architects, engineers and researchers from the electronic design community.

One may argue that we should model the half wave rectifier circuit (directly) in the theory of hybrid I/O-automata for

[1] In the literature, many different hybrid automaton definitions already exist. Loosely speaking, the theory of hybrid I/O-automata is a dialect of the (most common) theory of hybrid automata with two additional disjoint sets of variables (in the syntax) representing input variables and output variables of a hybrid automaton.

this analysis approach. However, the theory of hybrid I/O-automata does not offer the possibility to use algebraic reasoning and the hybrid automaton-based verification techniques always suffer from the state space explosion problem. As we can see from Section 3, algebraic reasoning has already been applied to achieve a simpler and more elegant specification of the half wave rectifier circuit in $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$. This also helps to obtain (by translation) the corresponding hybrid I/O-automaton model in a smaller size.

Generally speaking, analysis of specifications described in hybrid process algebras/calculi via formal translation to the theory of hybrid (I/O)-automata (that enables automatic verification of such specifications using existing hybrid automaton-based verification tools) is one of many analysis possibilities offered by hybrid process algebras/calculi.

4.4 Linearisation algorithms

Recently, *linearisation algorithms* (more sophisticated approach for analysis) have been developed for hybrid process algebras HyPA and Hybrid Chi (see [12, 29] for details). In process algebras, linearisation is a transformation of a recursive specification into a linear representation, i.e. a kind of normal form that is convenient for many forms of analysis. Note that these linear representations are expressed as recursive specifications as well, but they use only a small subset of the full process algebra. In general, such linear representations can also be considered very compact representations of a possibly infinite state space. The original recursive specification and its transformation are required to be bisimilar, which ensures that the relevant specification properties are preserved.

5 Hybrid I/O-Automata and PHAVer

In this section, we give a brief summary (that is relevant for this paper) of the model checker PHAVer and the formalism of hybrid I/O-automata. For a more extensive treatment, the reader is referred to [26].

5.1 PHAVer

PHAVer (Polyhedral Hybrid Automaton Verifyer) is a tool for analysing linear hybrid I/O-automata (i.e. a sub-class of hybrid I/O-automata which only allows linear dynamics) with the following characteristics:

1. exact and robust arithmetic based on the Parma Polyhedra Library;
2. on-the-fly over-approximation of piecewise affine dynamics;
3. conservative limiting of bits and constraints in polyhedral computations;
4. supports for compositional and assume-guarantee reasoning.

5.2 Hybrid I/O-automata

In the definition of hybrid I/O-automata, some functions and notations may be used.

- Given a set Var of variables, a *valuation* β: $\mathrm{Var} \to \mathbb{R}$ maps a real number to each variable.
- Let V(Var) denote the set of valuations over Var.
- An *activity* is a function f: $\mathbb{R}_{\geq 0} \to V$ in C^∞ (i.e. a function is C^∞ if the n-th derivative exists and is continuous for all n) and describes the change of valuations over time.
- Let Act(Var) denote the set of activities over Var.
- Let $f + t$ for $f \in \mathrm{Act(Var)}$ and $t \in \mathbb{R}_{\geq 0}$ be defined by $(f + t)(d) = f(d + t)$, $d \in \mathbb{R}_{\geq 0}$.
- A set $S \subseteq \mathrm{Act(Var)}$ of activities is *time invariant* if for all $f \in S, t \in \mathbb{R}_{\geq 0}$ $f + t \in S$.

A hybrid I/O-automaton *HIOA*=(Loc,$\mathrm{Var_S}$,$\mathrm{Var_I}$,$\mathrm{Var_O}$,Lab, $\to$, Act,Inv,Init) consists of the following components:

- a finite set Loc of locations;
- a finite and disjoint sets of state and input variables $\mathrm{Var_S}$ and $\mathrm{Var_I}$, and of output variables $\mathrm{Var_O} \subseteq \mathrm{Var_S}$, and let $\mathrm{Var} = \mathrm{Var_S} \cup \mathrm{Var_I}$;
- a finite set Lab of labels;
- a finite set of discrete transitions $\to \subseteq$ Loc $\times$ Lab $\times$ $2^{\mathrm{V(Var)} \times \mathrm{V(Var)}}$ $\times$ Loc;
- a transition $(l, a, \mu, l') \in \to$ is also written as $l \xrightarrow{a,\mu}_H l'$;
- a mapping (a labelling function) Act: Loc $\to$ $2^{\mathrm{Act(Var)}}$ from locations to time invariant sets of activities;
- a mapping Inv: Loc $\to$ $2^{\mathrm{V(Var)}}$ from locations to sets of valuations;
- a set Init $\subseteq$ Loc $\times$ V(Var) of initial states.

The semantics of a hybrid I/O-automaton is defined in terms of a time transition system. Let *HIOA* =(Loc,$\mathrm{Var_S}$,$\mathrm{Var_I}$,$\mathrm{Var_O}$,Lab, $\to$,Act,Inv,Init) be a hybrid I/O-automaton. A *state* of *HIOA* is a pair $(l, v) \in \mathrm{Loc} \times \mathrm{V(Var)}$ of a location and a valuation. The transition system interpretation of *HIOA*, written [*HIOA*], is the time transition system (Loc, $\mathrm{Var_S}$, $\mathrm{Var_I}$, $\mathrm{Var_O}$, Σ', $\to_{LH}$, Init), where $\Sigma' = \mathrm{Lab} \cup \mathbb{R}_{\geq 0} \cup \{\epsilon\}$ and $\to_{LH}$ is the union of $\xrightarrow{a}$, for $a \in \Sigma'$. The transition relations of such a time transition system are defined as follow:

1. $(l, v) \xrightarrow{a}_{LH} (l', v')$ iff $l \xrightarrow{a,\mu}_H l'$, $(v, v') \in \mu$, $v \in \mathrm{Inv}(l)$, $v' \in \mathrm{Inv}(l')$ (discrete transitions);

2. $(l, \upsilon) \xrightarrow{t}_{LH} (l', \upsilon')$ iff $l = l'$ and there exists $f \in \mathrm{Act}(l)$, $f(0) = \upsilon$, $f(t) = \upsilon'$, and $\forall t', 0 \leq t' \leq t \;\; f(t') \in \mathrm{Inv}(l)$ (timed transitions);

3. $(l, \upsilon) \xrightarrow{\epsilon}_{LH} (l', \upsilon')$ iff $l = l'$, $\upsilon \restriction \mathrm{Var_S} = \upsilon' \restriction \mathrm{Var_S}$, and $\upsilon, \upsilon' \in \mathrm{Inv}(l)$ (environment transitions).

These three kinds of transition relations are differentiated by their label, the time elapse involved, and a special label ϵ that represents changes in the input variables by the environment.

6 Analysis Strategy and Results

This section shows how to analyse the half wave rectifier circuit specification in $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ using PHAVer through the translation to hybrid I/O-automata. For brevity, in what follows, we may refer to hybrid I/O-automaton/a as hybrid automaton/a.

6.1 Translation

Defining the formal translation scheme from $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ to the theory of hybrid I/O automata and studying the correctness of the translation at the semantical level are far beyond the scope of this paper. Nevertheless, related works in this direction can be found in [15]. For simplicity, we briefly describe the informal translation from the half wave rectifier circuit specification in $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ to the corresponding hybrid automaton model as follows:

- The recursive equations D^{off} and D^{on} are translated to the two locations of a hybrid automaton *diode* (with two locations only). Also, the state propositions of recursive equations are translated to the activities of the corresponding locations with appropriated initial conditions that are assumed for the analysis. The emitting signals associating to the recursive equations are translated to the corresponding discrete transitions associating to the hybrid automaton *diode*. Two unimportant undelayable actions $\widetilde{\mathrm{D}_{\mathrm{t-on}}}$ and $\widetilde{\mathrm{D}_{\mathrm{t-off}}}$ are translated to two unimportant synchronisation labels. Figure 2 captures the main ideas of this translation as explained previously (due to the reason of space, only the graphical representation of hybrid automaton *diode*, i.e. the translation of recursive equations D^{off} and D^{on} representing the process *diode*, is shown). Note that some variables/labels are renamed (e.g. $\widetilde{\mathrm{D}_{\mathrm{t-off}}}$ to *jump*), because they are preently required by the parser of PHAVer.

- In a similar way, process *rectifier* is translated to the corresponding hybrid automaton *rectifier* with two locations.

- The translation of the process *generator* will be discussed in Subsection 6.2.

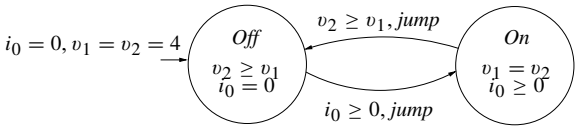

Figure 2: A hybrid automaton model of the process *diode*.

6.2 Approximation and refinement

Since most of the formal verification tools (including PHAVer) only allow linear dynamics for analysis using such tools, we have to approximate/refine the behaviour of the half wave rectifier circuit specification in $\mathrm{ACP}_{\mathrm{hs}}^{\mathrm{srt}}$ as follows:

1. We apply the same approximation used in [30][2] for F_{time} in the process *generator* as a sinusoidal voltage source. So, see [30] and the PHAVer code of the generator (as the translation of the process *generator*) in Subsection 6.3 for details.

2. We refine the predicate $C_0(\dot{\upsilon_2} - \dot{\upsilon_G}) = i_1$ in the process *rectifier* to $C_0(\dot{\upsilon_2}) = i_1$. This is allowed, because the predicate $\upsilon_G = 0$ must always hold in the process *rectifier*.

6.3 PHAVer codes of the half wave rectifier circuit

The input language of PHAVer [27] is a straightforward textual representation of linear hybrid I/O-automata. Using such an input language of PHAVer to describe the hybrid I/O-automata *diode*, *rectifier* and *generator*, we obtain PHAVer codes (i.e. the translations of the processes *diode*, *rectifier* and *generator*) as follows:

```
// -------------------
// Half wave rectifier
// -------------------
// ---------
// Constants
// ---------
C0:=0.01; R0:=10; R1:=100;
a1:=0.01272; au:=0.01274;
b1:=4; bu:=4; cu:=1.4143;
v0min := -bu; v0max :=  bu;
x0min := -au; x0max :=  au;
// -----------
// System description
// -----------
automaton diodo
state_var: v1, v2, i0;
synclabs: jump;
loc off: while v2>=v1 & i0==0 wait {true}
when i0>=0 sync jump do {i0'==i0} goto on;
```

[2]It is worth mentioning that a full wave rectifier was analysed using PHAVer in [30]. However, this analysis was done by leveraging abstraction to build system with few variables (only the input voltage source and the output voltage of the circuit were modelled).

```
loc on: while  v1==v2 & i0>=0 wait {true}
when v2>=v1 sync jump do
      {v1'==v1 & v2'==v2} goto off;
initially: on & v1==4 & v2==4 & i0==0;
end

automaton rectifier
state_var: v0, v1, v2, vG, i0, i1, i2;
synclabs: jumpp;
loc rect: while v0-v1==i0*R0 & v2-vG==i2*R1
          & vG==0 & i0==i1+i2
          wait {C0*v2'==i1} when true sync
          jumpp do {true} goto dead;
loc dead: while false  wait {false}
when true sync jumpp do {true} goto dead;
initially: rect & v0==0 & v1==4 & v2==4
                & vG==0 & i0==0 & i1==0
                & i2==0;
end

automaton generator
state_var: x0, v0;
synclabs: B;
loc CC: while x0min <= x0 & x0 <= x0max &
          v0min <= v0 & v0 <= v0max &
          v0 >= bl-bl/al*x0&
          v0 <= cu*bu-bl/au*x0max &
           0 <= x0&v0 <= x0max &
           0 <= v0 & v0 <= v0max
          wait {x0'==v0 & v0'== -98596*x0};
          when true sync B goto DD;
          when true sync B goto FF;
loc DD: while
          x0min <= x0 & x0 <= x0max &
          v0min <= v0 & v0 <= v0max &
          v0 >= bl-bl/al*(-x0)&
          v0 <= cu*bu-bl/au*(-x0)&
          x0min <= x0 & x0 <= 0 &
          0 <= v0 & v0 <= v0max
          wait {x0'==v0 & v0'== -98596*x0};
          when true sync B goto CC;
          when true sync B goto EE;
loc EE: while
          x0min <= x0 & x0 <= x0max &
          v0min <= v0 & v0 <= v0max &
          -v0 >= bl-bl/al*(-x0)&
          -v0 <= cu*bu-bl/au*(-x0)&
          x0min <= x0 & x0 <= 0 &
          v0min <= v0 & v0 <= 0
          wait {x0'==v0 & v0'== -98596*x0};
          when true sync B goto DD;
loc FF: while
          x0min <= x0 & x0 <= x0max &
          v0min <= v0 & v0 <= v0max &
          -v0 >= bl-bl/al*x0&
          -v0 <= cu*bu-bl/au*x0&
          0 <= x0 & x0 <= x0max &
          v0min <= v0 & v0 <= 0
          wait {x0'==v0 & v0'== -98596*x0};
          when true sync B goto EE;
initially: $ & x0== -0.01273 & v0==0;
end
```

In order to increase the readability of the system description, some simplifications are introduced in the PHAVer codes and also some unimportant labels are used.

6.4 Analysis by means of model checking

For illustration purposes, we analyse (by means of model checking) the half wave rectifier circuit specification in ACP_{hs}^{srt} using PHAVer through the translation to hybrid I/O-automata. We want to verify two basic properties (for safety) of the half wave rectifier circuit as follows:

1. the voltage v_2 is never negative;
2. for a given $v_0 = 4\sin(2\pi(50)t)V$ and the initial condition $v_2(0) = 4V$ (as already indicated in the PHAVer codes), $v_2(t)$ does not drop below a threshold $1.5V$ for any time t.

The following PHAVer analysis commands were used to specify the two safety properties as forbidden states:

```
sys = diodo&rectifier&generator;
reg = sys.reachable;
reg.remove(x0);
reg.print;
forbidden = sys.{$&v2<0|v2<1.5};
echo "";
reg.intersection_assign(forbidden);
echo "intersection with forbidden states:";
reg.is_empty;
```

PHAVer reported that these two safety properties held in all locations of the system (i.e. the parallel composition of hybrid automata *diode*, *rectifier* and *generator*).

7 Further Analysis

To gian confidence of the analysis results shown in Subsection 6.4, in this section, we further model the half wave rectifier circuit using the Modelica language and analyse the same propoerties presented in Subsection 6.4 in the OpenModelica System enviroment.

7.1 OpenModelica system and Modelica language

The OpenModelica System is an efficient interactive computational environment for the Modelica language. The Modelica language is primarily a modelling language that allows one to specify mathematical models of complex physical systems. It is also an object-oriented equation based programming language, oriented towards computational applications with high complexity requiring high performance. The four most important features (taken from [31] and [32]) of the Modelica language are:

1. It is based on equations instead of assignment statements. This allows acausal modelling that gives better reuse of

models since equations do not specify a certain data flow direction. Thus a Modelica model can adapt to more than one data flow context.

2. It has multi-domain modelling capability, meaning that model components corresponding to physical objects from different domains including hybrid systems.
3. It is an object-oriented language with a general class concept that unifies classes, generic - known as templates in C++, and general sub-typing into a single language construct. This facilitates reuse of components and evolution of models.
4. It has a strong software components model, with constructs for creating and connecting components. Thus it is ideally suited as an architectural description language for complex physical systems and to some extent for software systems.

Loosely speaking, a Modelica model (also called class) contains variable declarations (possibly with initial values) and equation sections containing equations. For illustration purposes, below is a sample of a Modelica model:

```
model Sample
 Real x(start = 1); // variable declarations,
                       x starts at 1
 parameter Real a = 1;

equation
 der(x) = -a*x;      // equation sections
end Sample;
```

To handle large models, in Modelica, a model can be built up from connections. Various components can be connected using the "connect" statement. Furthermore, Modelica has a electrical component library which consists of many electrical components (e.g. resistor, capacitor, inductor and ideal diodo). Such components can be freely instantiated for reuse and are also the key to effective modeling complex systems. For illustration purpose, we provide a resistor model in Modelica as follows:

```
model Resistor
 Pin p, n;    // "Positive" and "negative"
                 pins parameter Real R
                "Resistance"

equation
 n.i = p.i; // assume both n.i and p.i to
               be positive
            // when currents flows from p
               to n
 R*p.i = p.v - n.v;
end Resistor;
```

7.2 Modelica half wave rectifier circuit model

Below is the half wave rectifier circuit described in Modelica:

```
model HalfWaveRectifier
  Modelica.Electrical.Analog.Basic.
                      Resistor R0(R=10);
  Modelica.Electrical.Analog.Basic.
                      Resistor R1(R=100);
  Modelica.Electrical.Analog.Ideal.
                      IdealDiode DD;
  Modelica.Electrical.Analog.Basic.
                      Capacitor C0(C=0.01);
  Modelica.Electrical.Analog.Sources.
                      SineVoltage AC(V=4);
  Modelica.Electrical.Analog.Basic.
                      Ground G;

equation
  connect(AC.p, R0.p);
  connect(R0.n, DD.p);
  connect(DD.n, C0.p);
  connect(C0.p, R1.p);
  connect(C0.n, AC.n);
  connect(R1.n, AC.n);
  connect(AC.n, G.p);
end HalfWaveRectifier;
```

7.2.1 Interactive session in OpenModelica system

Following the OpenModelica System commands *loadFile*, *simulate*, *instantiateModel* and *plot*, the Modelica half wave rectifier model was loaded into the system, was instantiated with appropriated parameters and was simulated. OpenModelica System reported all these as follows:

```
>>loadFile("C:/OpenModelica1.4.2/testmodels
/HalfWaveRectifier.mo")
true
>> simulate(HalfWaveRectifier,startTime=0.0,
stopTime=100.0)
record
resultFile = "HalfWaveRectifier_res.plt"
end record

>> instantiateModel(HalfWaveRectifier)

"fclass HalfWaveRectifier
Real R0.v "Voltage drop between the two
pins (= p.v - n.v)";
Real R0.i "Current flowing from pin p to
pin n";
Real R0.p.v "Potential at the pin";
Real R0.p.i "Current flowing into the pin";
Real R0.n.v "Potential at the pin";
Real R0.n.i "Current flowing into the pin";
parameter Real R0.R = 10 "Resistance";
Real R1.v "Voltage drop between the two pins
(= p.v - n.v)";
Real R1.i "Current flowing from pin p to
pin n";
Real R1.p.v "Potential at the pin";
Real R1.p.i "Current flowing into the pin";
```

```
Real R1.n.v "Potential at the pin";
Real R1.n.i "Current flowing into the pin";
parameter Real R1.R = 100 "Resistance";
Real DD.v "Voltage drop between the two pins
                        (= p.v - n.v)";
Real DD.i "Current flowing from pin p to pin
                                        n";
Real DD.p.v "Potential at the pin";
Real DD.p.i "Current flowing into the pin";
Real DD.n.v "Potential at the pin";
Real DD.n.i "Current flowing into the pin";
parameter Real DD.Ron(min = 0.0) = 1e-05
"Forward state-on differential resistance
(closed diode resistance)";
parameter Real DD.Goff(min = 0.0) = 1e-05
"Backward state-off conductance(opened
diode conductance)";
parameter Real DD.Vknee(min = 0.0) = 0
"Forward threshold voltage";
Boolean DD.off(start = true)
"Switching state";
Real DD.s "Auxiliary variable: if on then
current, if opened then voltage";
Real C0.v "Voltage drop between the two pins
(= p.v - n.v)";
Real C0.i "Current flowing from pin p to
pin n";
Real C0.p.v "Potential at the pin";
Real C0.p.i "Current flowing into the pin";
Real C0.n.v "Potential at the pin";
Real C0.n.i "Current flowing into the pin";
parameter Real C0.C = 0.01 "Capacitance";
Real AC.v "Voltage drop between the two pins
(= p.v - n.v)";
Real AC.i "Current flowing from pin p to
pin n";
Real AC.p.v "Potential at the pin";
Real AC.p.i "Current flowing into the pin";
Real AC.n.v "Potential at the pin";
Real AC.n.i "Current flowing into the pin";
parameter Real AC.offset = 0
"Voltage offset";
parameter Real AC.startTime = 0
"Time offset";Real AC.signalSource.y
"Connector of Real output signal";
parameter Real AC.signalSource.amplitude
= AC.V "Amplitude of sine wave";
parameter Real AC.signalSource.freqHz =
AC.freqHz "Frequency of sine wave";
parameter Real AC.signalSource.phase =
AC.phase "Phase of sine wave";
parameter Real AC.signalSource.offset =
AC.offset "Offset of output signal";
parameter Real AC.signalSource.startTime =
AC.startTime "Output = offset for
time < startTime";
constant Real AC.signalSource.pi =
3.14159265358979;
parameter Real AC.V = 4 "Amplitude of sine
wave";
parameter Real AC.phase = 0 "Phase of sine
wave";
parameter Real AC.freqHz = 1 "Frequency of
sine wave";
Real G.p.v "Potential at the pin";
Real G.p.i "Current flowing into the pin";
equation
R0.R * R0.i = R0.v;
R0.v = R0.p.v - R0.n.v;
0.0 = R0.p.i + R0.n.i;
R0.i = R0.p.i;
R1.R * R1.i = R1.v;
R1.v = R1.p.v - R1.n.v;
0.0 = R1.p.i + R1.n.i;
R1.i = R1.p.i;
DD.off = DD.s < 0.0;
DD.v = DD.s * if DD.off then 1.0 else
DD.Ron + DD.Vknee;
DD.i = DD.s * if DD.off then DD.Goff else
1.0 + DD.Goff * DD.Vknee;
DD.v = DD.p.v - DD.n.v;
0.0 = DD.p.i + DD.n.i;
DD.i = DD.p.i;
C0.i = C0.C * der(C0.v);
C0.v = C0.p.v - C0.n.v;
0.0 = C0.p.i + C0.n.i;
C0.i = C0.p.i;
AC.signalSource.y = AC.signalSource.offset
+ if time < AC.signalSource.startTime
then 0.0 else AC.signalSource.amplitude *
Modelica.Math.sin(6.28318530717959 *
AC.signalSource.freqHz *(time -
AC.signalSource.startTime) +
AC.signalSource.phase );
AC.v = AC.signalSource.y;
AC.v = AC.p.v - AC.n.v;
0.0 = AC.p.i + AC.n.i;
AC.i = AC.p.i;
G.p.v = 0.0;
R1.n.i + C0.n.i + AC.n.i + G.p.i = 0.0;
R1.n.v = C0.n.v;
C0.n.v = AC.n.v;
AC.n.v = G.p.v;
DD.n.i + C0.p.i + R1.p.i = 0.0;
DD.n.v = C0.p.v;
C0.p.v = R1.p.v ;
R0.n.i + DD.p.i = 0.0;
R0.n.v = DD.p.v;
AC.p.i + R0.p.i = 0.0;
AC.p.v = R0.p.v;
end HalfWaveRectifier;
"

>> plot({DD.n.v})
true
>>
```

7.2.2 Analysis by means of simulation

The waveform shown in Figure 3 was obtained by simulating the Modelica half wave rectifier model with OpenModel-

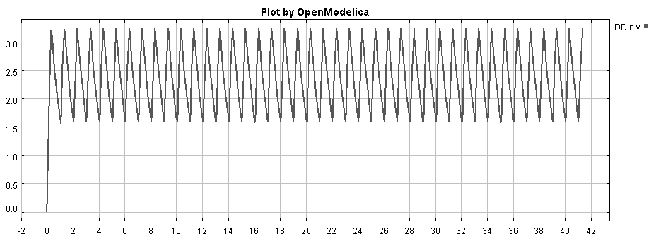

Figure 3: Simulation results of the Modelica half wave rectifier model.

ica System. It is not hard to see that the value of DD.n.v in the Modelica half wave rectifier model (i.e. the voltage v_2 as shown in Figure 1) is never negative and is always above $1.5V$. These are also the two properties of the half wave rectifier circuit analysed in Subsection 6.4.

Clearly, the analysis results of the half wave rectifier circuit by means of model checking and simulation are correlated to various properties.

8 Conclusions and Future Works

We were not surprised to find out that it was possible to specify and analyse the half wave rectifier circuit in $\text{ACP}_{\text{hs}}^{\text{srt}}$. In order to illustrate our work clearly, only a simple mixed-signal circuit was given in this paper. Nevertheless, the use of $\text{ACP}_{\text{hs}}^{\text{srt}}$ is generally applicable to all sizes and levels of mixed-signal circuits even for analog and digital circuits.

It is also worth mentioning that there are tools for other hybrid process algebras/calculi. Below is a summary for hybrid process algebras/calculi tools:

1. *Hybrid Chi Python simulator* [33]: a symbolic simulator for Hybrid Chi specifications;
2. *Chi2HA translator* [33]: Chi2HA translates (a subset of) Hybrid Chi specifications into corresponding hybrid automata (with urgency) that can be verified directly by the model checker PHAVer;
3. *HyPA linearisation tool* [12]: a number of HyPA specifications can be linearised using this tool;
4. *HyPA simulator* [16]: a simulator for HyPA specifications;
5. *BHAVE* [34] : a prototype of BHPC simulation tool;
6. *BHPC2MOD* [20]: an automatic translator which accepts a BHPC specification (restricted to a subset of BHPC) as input and translates it to the corresponding Modelica specification.
7. *SPHIN* [22]: SPHIN is a hybrid model checker for ϕ-calculus specifications.

Our future work will focus on a comparative study of hybrid algebras/calculi for the formal specification and analysis of analog and mixed-signal circuits. We are also interested to survey performance and to investigate the theoretical foundations behind of the above-mentioned tools for the analysis of analog and mixed-signal circuits.

Acknowledgements

K.L. Man wishes to thank Jos Baeten, Bert van Beek, Mohammad Mousavi, Koos Rooda, Ramon Schiffelers, Pieter Cuijpers, Michel Reniers, Kees Middelburg, Uzma Khadim and Muck van Weerdenburg for many stimulating and helpful discussions (focusing on process algebras for distinct systems) in the past few years.

He also would like to thank Rolf Theunissen for some helpful discussions to master the use of the model checker PHAVer.

Availability

For research purposes, we would be pleased to receive interesting case studies on formal specification and analysis of analog/mixed-signal (AMS) and hardware system circuits from anyone working in these areas.

For more information, please send mail to pafesd@gmail.com or visit the Process Algebras for Electronic System Designs (PAFESD) homepage in [35].

References

[1] J.C.M. Baeten, W.P. Weijland, Process Algebra, Number 18 in *Cambridge Tracts in Theoretical Computer Science*, Cambridge University Press, 1990.

[2] J.C.M. Baeten, C.A. Middelburg, Process Algebra with Timing, in *EATCS Monographs Series*, Springer-Verlag, 2002.

[3] D.A. van Beek, K.L. Man, M.A. Reniers, J.E. Rooda, R.R.H. Schiffelers, Syntax and Semantics of Timed Chi, in *Technical Report CS-Report 05-09*, Eindhoven University of Technology, The Netherlands, 2005.

[4] K.L. Man, M.A. Reniers, P.J.L. Cuijpers, Case Studies in the Hybrid Process Algebra HyPA, in *International Journal of Software Engineering & Knowledge Engineering*, 15(2):299-305, 2005.

[5] K.L. Man, M.P. Schellekens, M. Boubekeur, Formal Specification and Analysis of Analog and Mixed-Signal Circuits Using Process Algebras for Hybrid Systems (with a focus on hybrid process algebra $\text{ACP}_{\text{hs}}^{\text{srt}}$), in *Proceedings of the IEEE International SoC Design Conference (ISOCC), Seoul, Korea*, October, 2006.

[6] K.L. Man, M.P. Schellekens, Analysis of a Mixed-Signal Circuit in Hybrid Process Algebra ACP_{hs}^{srt}, in *Proceedings of the International MultiConference of Engineers and Computer Scientists, Hong Kong*, March, 2007.

[7] K.L. Man, M.P. Schellekens, Interoperability of Performance and Functional Analysis for Electronic System Designs in Behavioural Hybrid Process Calculus (BHPC), invited book chapter, to appear in *Springer edited book: Current Trends in Intelligent Systems and Computer Engineering*.

[8] Gordon D. Plotkin, A Structural Approach to Operational Semantics, in *Report DAIMI FN-0.59*, Computer Science Department, Aarhus University, 1981.

[9] D.A. van Beek, K.L. Man, M.A. Reniers, J.E. Rooda, R.R.H. Schiffelers, Syntax and Consistent Equation Semantics of Hybrid Chi, in *Journal of Logic and Algebraic Programming*, 68(1-2):129-210, 2006.

[10] D.A. van Beek, K.L. Man, M.A. Reniers, J.E. Rooda, R.R.H. Schiffelers, Relating Hybrid Chi to Other Formalisms, to appear in *Electronic Notes in Theoretical Computer Science*.

[11] P.J.L. Cuijpers, M.A. Reniers, Hybrid Process Algebra, in *Journal of Logic and Algebraic Programming*, 62(2):191-245, 2005.

[12] P.C.W. van den Brand, M.A. Reniers, P.J.L. Cuijpers, Linearization of Hybrid Processes, in *Journal of Logic and Algebraic Programming*, 68(1-2):54-0.504, 2006.

[13] J.A. Bergstra, C.A. Middelburg, Process Algebra for Hybrid Systems, in *Theoretical Computer Science*, 335(2/3):215-280, 2005.

[14] J.A. Bergstra, C.A. Middelburg, Continuity Controlled Hybrid Automata, in *Journal of Logic and Algebraic Programming*, 68(1/2):5-53, 2006.

[15] K.L. Man, R.R.H. Schiffelers, Formal Specification and Analysis of Hybrid Systems, in *Ph.D. thesis*, Eindhoven University of Technology, The Netherlands, 2006.

[16] R.A. Schouten, Simulation of Hybrid Processes, in *Master's thesis*, Eindhoven University of Technology, The Netherlands, 2005.

[17] P.J.L. Cuijpers, Hybrid Process Algebra, in *Ph.D. thesis*, Eindhoven University of Technology, The Netherlands, 2004.

[18] Ed Brinksma, Tomas Krilavičius, Behavioural Hybrid Process Calculus, in *CTIT Technical Report TR-CTIT-05.45*, University of Twente, The Netherlands, 2005.

[19] Tomas Krilavičius, Hybrid Techniques for Hybrid Systems, in *Ph.D. thesis*, University of Twente, The Netherlands, 2006.

[20] A. van Putten, Behavioural Hybrid Process Calculus Parser and Translator to Modelica, in *Master's thesis*, University of Twente, 2006.

[21] W. Rounds, H. Song, The ϕ-Calculus: A Language for Distributed Control of Reconfigurable Embedded Systems, in *Lecture Notes in Computer Science*, vol. 2623:435-449, Springer-Verlag, 2003.

[22] W. Rounds, H. Song, K.J. Compton, SPHIN: A Model-checker for Reconfigurable Hybrid Systems Based on SPIN, in *Electronic Notes on Theoretical Computer Science*, vol. 145:167-183, 2006.

[23] U. Khadim, A Comparative Study of Process Algebras for Hybrid Systems, in *Technical Report CS-06.23*, Eindhoven University of Technology, The Netherlands, 2006.

[24] T.A. Henzinger, The Theory of Hybrid Automata, in *Proceedings of the 11th Annual Symposium on Logic in Computer Science, New Brunswick, NJ, USA*, 1996.

[25] N.A. Lynch, R. Segala, F.W. Vaandrager, H.B. Weinberg, Hybrid I/O Automata, in *Lecture Notes in Computer Science 1066*, Springer-Verlag, pp. 496-510, 1996.

[26] Goran Frehse, PHAVer: Algorithmic Verification of Hybrid Systems Past HyTech, in *Lecture Notes in Computer Science 3414*, Springer-Verlag, pp. 258-273, 2005.

[27] Goran Frehse, Language Overview for PHAVer version 0.35, available at *http://www.cs.ru.nl/ goranf/*.

[28] T.A. Henzinger, P.H. Ho, H. Wong-Toi, HyTech: A Model Checker for Hybrid Systems, in *Journal of Software Tools for Technology Transfer*, 1:110-0.522, 1997.

[29] J.C.M. Baeten, D.A. van Beek, J.E. Rooda, Handbook of Dynamic System Modeling (chapter process algebra), in *CRC Press LLC*, 2006.

[30] L.P. Carloni, R. Passerone, A. Pinto, A.L. Sangiovanni-Vincentelli, Language and Tools for Hybrid Systems Design, in *Journal of Foundation and Trends*, vol. 1:1-177, 2005.

[31] Modelica, A Unified Object-Oriented Language for Physical Systems Modeling, *Modelica homepage: http://www.Modelica.org*.

[32] OpenModelica System is available at *http://www.ida.liu.se/ pelab/modelica/*.

[33] Tools and manuals for Hybrid Chi are availabe at *http://se.wtb.tue.nl/sewiki/chi/*.

[34] BHAVE prototype of BHPC simulation tool can be found at *http://fmt.cs.utwente.nl/tools/bhave/*.

[35] PAFESD Homepage: *http://digilander.libero.it/systemcfl/ pafesd/*.