CASE STUDIES IN THE HYBRID PROCESS ALGEBRA HyPA

K.L. MAN, M.A. RENIERS and P.J.L. CUIJPERS
Department of Computer Science, Eindhoven University of technology (TU/e), Den Dolech 2
5612 AZ Eindhoven, The Netherlands
k.l.man, m.a.reniers, p.j.l.cuijpers@tue.nl

HyPA is an algebraic theory based on the classical process algebra Algebra of Communicating Processes (ACP) for the specification and analysis of hybrid systems. We have the idea that HyPA is also well suited for addressing various aspects of digital embedded systems including hardware, software and concurrency, as well as mixed-signal designs. To show that HyPA is useful for the specification and analysis of hybrid systems and that our idea is correct, we illustrate the use of HyPA with some case studies: a point-to-point communication, a thermostat, a positive-edge-triggered D flip flop, and a small part of a mixed-signal fuzzy controller.

1. Introduction

Process algebras are useful tools for the specification and verification of various systems. Generally speaking, process algebras describe the behavior 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 a process algebra can be partially or completely carried out by mathematical proofs using the equational theory.

Recently, several process algebras have been developed for hybrid systems. Hybrid systems are systems that both exhibit discrete and continuous behavior. Such systems have proved fruitful in a great diversity of engineering application areas including air-traffic control, automated manufacturing, and chemical process control.

The introduction of the hybrid process algebra HyPA (see Ref. 1) for the specification and analysis of hybrid systems, initiated an attempt to extend the knowledge and experience of the field of process algebra to the field of hybrid systems. The language HyPA has a formal semantics defined by means of deduction rules in a standard structured operational semantics (SOS) style that associates a hybrid transition system with each process term. The goal of developing a formal semantics is to provide a complete and unambiguous specification of the language. It also contributes significantly to the sharing, portability and integration of various applications in simulation, synthesis and formal verification.

Over the past years, research in formal semantics in the electronic community mainly focused on Verilog, VHDL, SystemC and VHDL-AMS. Their definitions are based on Abstract State Machine (ASM) specifications and Denotational
It is generally believed that the SOS style semantics is more intuitive, and the methods of ASM specifications and denotational semantics appear to be difficult to apply to describe the dynamic behavior of processes. Since processes are the basic units of execution for simulating the behavior of a device or a system within these languages, process algebras with the SOS style semantics are potentially a good candidate for giving formal specifications of systems in the electronic community.

The goal of this paper is to show that HyPA is useful for addressing various aspects of digital embedded systems including hardware, software and concurrency, as well as mixed-signal designs. To achieve this goal, we illustrate the use of HyPA through case studies taken from different industrial areas and literature. To our knowledge, this is the first paper to report the modeling of mixed-signal designs using a process algebra based formalism.

Similar to HyPA, the languages Hybrid Chi (see Ref. 6) and ACP_{hs} (see Ref. 7) are also process algebras for the specification and analysis of hybrid systems. These two formalisms adopt the view that a hybrid system is a system in which a discrete state transition takes place when the system performs an action and a continuous state evolution occurs while the system is idling between performing successive actions. This is not sufficient for giving specifications of mixed-signal designs. On the other hand, passage of time allows mode changes in HyPA. This is well suited for modeling mixed-signal designs.

This paper is organized as follows. In Sec. 2, we review the syntax and formal semantics of the hybrid process algebra HyPA. Sec. 3 illustrates the use of HyPA with some case studies taken from different industrial areas and literature: a point-to-point communication, a thermostat, a positive-edge-triggered D flip flop, and a mixed-signal fuzzy controller. Finally, some concluding remarks are made in Sec. 4.

2. Hybrid Process Algebra

In this section, we give an overview of the subset of the hybrid process algebra HyPA that is used in this paper. A process $P$ in HyPA is built from atomic process terms and operators applied to process terms. It is defined according to the following grammar:

\[ P ::= a \mid c \mid d \gg P \mid P \oplus P \mid P \ominus P \mid P \succ P \mid P \parallel P \mid \partial H(P) \mid [V \mid P] \]

Discrete actions $a \in A$ are used to model discrete and computational behavior. Flow clauses $c \in C$ are used to model continuous physical behavior. A flow clause is a pair $(V \mid P_f)$ of a set of model variables $V$ and a flow predicate $P_f$. The set $V$ models the variables that are not allowed to jump at the beginning of a flow. Predicate $P_f$ models the continuous behavior. More precisely, $P_f$ describes a set of flows, where a flow is a partial function of time to the valuations of model variables. The flows that are described by a flow predicate are called the solutions of that predicate. The process re-initialization operator $d \gg p$ models the behavior of process $p$ where
the model variables are submitted to a discontinuous change as specified by the re-initialization clause $d$. A re-initialization clause $d$ is a pair $[V | P_r]$ of a set of model variables $V$ and a re-initialization predicate $P_r$. The set $V$ models the variables that are allowed to change. Note that this is precisely opposite to flow clauses, where $V$ denotes those variables that do not change (initially). Predicate $P_r$ models the discontinuous changes. In a predicate, $x^-$ denotes the valuation of a variable $x$ before re-initialization, and $x^+$ denotes the valuation of a variable $x$ after re-initialization. Similar to flow clauses, the re-initializations that are described by a re-initialization predicate are called the solutions of that predicate. Alternative composition $p \oplus q$ models a non-deterministic choice between processes $p$ and $q$. Sequential composition $p \odot q$ models a sequential execution of processes $p$ and $q$. Disrupt $p \triangleright q$ models a kind of sequential composition where process $q$ may take over execution from process $p$ at any moment, without waiting for its termination. The parallel composition $p \parallel q$ is used to express parallelism, i.e., discrete actions are executed in an interleaving manner with the possibility of communication (defined by means of a communication function), while flow clauses are forced to synchronize, and can synchronize only if they accept the same solution. The encapsulation $\partial(H)$ is used to block the execution of the actions that occur in $H$. The variable abstraction operator $|V | p|$ abstracts from the variables from the set $V$, i.e., the value of those variables is invisible to the environment of the process.

The semantics of HyPA is defined by means of deduction rules in SOS style that associate a hybrid transition system with each process term as explained in Ref. 1. Such a hybrid transition system has two different kinds of transitions, namely one associated with computational behavior (i.e., discrete actions), and the other associated with physical behavior (i.e., flow clause).

3. Case Studies

In this section, we show that HyPA can be reasonably and efficiently used to specify systems with different characteristics such as those exploiting concurrency or those used in the areas of hybrid systems, embedded hardware, and mixed-signal designs.

3.1. Concurrency

In this example, we show how the communication of some data values from one process to some other process can be specified in HyPA. Suppose a sender $Prod$ has the desire to send the value of some given expression $e$ to a receiver $Cons$, where this value is to be stored in a certain variable, say $in$. In this expression $e$ model variables may be used. In HyPA it is not possible to communicate values from one process to another directly. This type of communication therefore has to be modeled by using a fresh shared variable, say $m$.

$$Prod \approx [\{m, in\} | m^+ = e^-] \gg send, \quad Cons \approx [\{in, m\} | in^+ = m^+] \gg rec.$$
The notation $e^{-}$ represents expression $e$ where all occurrences of model variables are replaced by $-$ superscripted occurrences of these variables. In HyPA, the actions that communicate and the result of such a communication is specified by means of a communication function. For this case, we have $\gamma(\text{send}, \text{rec}) = \text{comm}$ and $\gamma$ is undefined otherwise. In HyPA, the actions $\text{send}$ and $\text{rec}$ are not forced to synchronize. In order to force them to synchronize we encapsulate the actions $\text{send}$ and $\text{rec}$ by means of the encapsulation operator $\partial H$, where $H = \{\text{send}, \text{rec}\}$.

Furthermore, we abstract from the auxiliary variable $m$ that was only introduced to achieve the communication of the value of the expression $e$ by means of the variable abstraction operator: $[\{m\} | \partial H(\text{Prod} \parallel \text{Cons})]$. Using the axioms of HyPA given in Ref. 1 and Ref. 9, one can derive that this process is equivalent to the process $[\{\text{in}\} | \text{in}^+ = e^-] \Rightarrow \text{comm}$. Using these axioms, many system descriptions can be transformed into a linear representation, i.e., a kind of normal form that is convenient for many forms of analysis. Generally, such linear representations can also be considered very compact symbolic representations of possibly infinite state spaces.

### 3.2. Hybrid systems

Some complex case studies of the application of HyPA to model and to perform analysis of hybrid systems already appeared in Ref. 8. For illustration purposes, we only give a simple case study in this subsection. This case study, adapted from Ref. 10, models a thermostat that keeps the temperature between 1 and 3 degrees. The behavior of the thermostat is described as follows. Initially, the temperature equals 2 degrees, and the thermostat is off. The temperature falls according to the differential equation $\dot{x} = -x$. The thermostat will be turned on when the temperature reaches 1 degree, then the temperature rises according to the differential equation $\dot{x} = -x + 5$. The thermostat will be turned off again when the temperature reaches 3 degrees. A HyPA specification of the thermostat is given as follows:

On $\approx$ $\{\{y, z\} | x \leq 3 \land \dot{x} = -x + 5 \land \dot{y} = 1 \land \dot{z} = 1\} \triangleright ([\emptyset | x^- = 3] \gg \text{Off})$, 

Off $\approx$ $\{\{y, z\} | 1 \leq x \land \dot{x} = -x \land \dot{y} = 1 \land \dot{z} = 0\} \triangleright ([\emptyset | x^- = 1] \gg \text{On})$, 

Thermostat $\approx$ $\{\{y, z\} | x^+ = 2 \land y^+ = z^+ = 0\} \gg \text{Off}$.

The processes $\text{On}$ and $\text{Off}$ model the behavior of the thermostat when it is turned on and off respectively. State/mode changes (i.e., from $\text{On}$ to $\text{Off}$ and vice versa) are modeled using the disrupt operator ($\triangleright$). The initial values of the model variables are initialized through the process re-initialization operator ($\gg$). For verification purposes, two auxiliary variables $y$ and $z$ are introduced in this case study, in a way that does not alter the behavior of the thermostat. The variable $y$ is a clock (i.e., $\dot{y} = 1$) that measures the elapsed time; the variable $z$ is a stopwatch (i.e., $\dot{z} = 0$ or $\dot{z} = 1$) that measures the accumulated time spent when the thermostat is on.

Note that no discrete actions are used at all. In some formalisms for the description of hybrid systems, such as hybrid automata (see Ref. 10), re-initialization of variables and mode switching can only be achieved by means of discrete events.
It is very important to study the correctness of models. Based on the method for the analysis of safety for actions and safety for predicates proposed in Ref. 8, we can study and verify some safety properties in terms of predicates on the model variables. For example, the safety property that the thermostat is active at most \( \frac{2}{3} \) of the time can be expressed as \( z \leq \frac{2}{3}y \). Due to space limitations, the method for the analysis of safety for actions and safety for predicates is not presented in this paper.\(^8\) As an alternative to this kind of analysis one can use the HyPA simulator, which is currently being developed, to obtain some confidence in the correctness of the model.\(^11\)

For example, the model can be extended by the following monitor process \( \text{Mon} \) that records in a Boolean variable \( \text{violation} \) whether or not the property is violated:

\[
\text{Mon} \approx (\text{violation} = (z > 2y/3)).
\]

The whole system that is put into the simulator is then given by the parallel composition of the thermostat process and the monitor process: \( \text{Thermostat} \parallel \text{Mon} \). Now one only has to observe that the variable \( \text{violation} \) never has the value true.

### 3.3. Embedded hardware

Positive-edge-triggered D flip-flops are among the basic building blocks of RTL (Register-Transfer Level) designs.\(^4\) Such a flip-flop has a clock input \( \text{cl} \) in the sensitivity list, a data input \( d \), and a data output \( Q \). In the HyPA specification the inputs and output are modeled by Boolean variables. When a positive edge occurs in the clock signal (which means \( \text{cl}^+ \land \neg \text{cl}^- \)), the value of input port \( d \) is assigned to output port \( Q \). A HyPA specification is given as follows:

\[
\text{DFF} \approx (\{\text{cl}, Q\} \mid \dot{Q} = \text{cl} = 0) \triangleright (\{\{\text{cl}, d, Q\} \mid \text{cl}^+ \land \neg \text{cl}^- \land Q^+ = d^- \} \gg \text{DFF}) \\
\oplus (\{\{\text{cl}, d\} \mid \neg \text{cl}^+ \land \text{cl}^- \} \gg \text{DFF})).
\]

### 3.4. Mixed-signal design

This subsection presents a small part of a case study of a mixed-signal design from Ref. 12. This case study is a complex mixed signal fuzzy controller chip...
implemented in a 0.35 µm CMOS technology. This mixed-signal design has a 3-dimensional fuzzy partition membership function that is the core of a mixed-signal fuzzy controlled integrated circuit. The analogue part holds a key advantage over the digital part in applications where computational speed is needed. In stand alone mode, it is a fuzzy controller able to elaborate analog and digital signals and to communicate through an \(^{2}\)C bus with an EEPROM, necessary to fix the weights of the fuzzy system. In slave mode, the device acts as a peripheral of a micro-controller. The digital part has a micro-controller, an \(^{2}\)C driver and a RAM that is composed of 2864 standard cells, and the analog fuzzy engine that is composed of 29673 MOSFETs.

The fundamental block of the complete system is the membership function circuit (see Fig. 1). It consists of current mirrors and 4 differential pairs, biased by the voltages \(V_{Li}\) and \(V_{Ri}\). The whole system contains 31 of such blocks. The HyPA specification for the membership function circuit describes the dependence of the 5 output currents with the input voltages of a membership function block as follows:

\[
\begin{align*}
Out1 & \approx (I_0 \mid I_{out1} = I_{MLi}), \\
Out2 & \approx (I_0 \mid I_{out2} = I_{MRi} - I_{MRj}), \\
Out3 & \approx (I_0 \mid I_{out3} = I_{MRj} - I_{MRk}), \\
Out4 & \approx (I_0 \mid I_{out4} = I_{MRj} - I_{MRk}), \\
Out5 & \approx (I_0 \mid I_{out5} = I_{MRj}),
\end{align*}
\]

where, for \(i \in \{1, \ldots , 5\}\) and \(k\) a given constant,

\[
I_{MLi} = \begin{cases} 0, & \text{if } v_{Li} - v_{Ri} \leq - \sqrt{\frac{I_0}{k}}, \\
\frac{I_0}{2} + \sqrt{2}k\left(\frac{v_{Li} - v_{Ri}}{2}\right)^2 & \sqrt{\frac{I_0}{k}} < v_{Li} - v_{Ri} < \sqrt{\frac{I_0}{k}}, \\
I_0, & \text{if } v_{Li} - v_{Ri} \geq \sqrt{\frac{I_0}{k}},
\end{cases}
\]

\[
I_{MRi} = \begin{cases} 0, & \text{if } v_{Li} - v_{Ri} \leq - \sqrt{\frac{I_0}{k}}, \\
\frac{I_0}{2} + \sqrt{2}k\left(\frac{v_{Li} - v_{Ri}}{2}\right)^2 & - \sqrt{\frac{I_0}{k}} < v_{Li} - v_{Ri} < \sqrt{\frac{I_0}{k}}, \\
I_0, & \text{if } v_{Li} - v_{Ri} \geq \sqrt{\frac{I_0}{k}}.
\end{cases}
\]

Processes \(Out1, \ldots , Out5\) describe the 5 output currents. The membership function circuit is the parallel composition of the processes describing the output currents: \(Fuzzy \approx Out1 \parallel Out2 \parallel Out3 \parallel Out4 \parallel Out5\).

4. Concluding Remarks

HyPA can be reasonably and effectively used to give specifications of systems from different application areas as indicated in this paper. It should be noted though that for describing the transmission of data from one process to another a variable needs to be used. In some other process algebras (e.g., Ref. 6), this is achieved in a more straightforward way.

The validation and verification of systems described in HyPA is still very immature. Currently one can reason algebraically about equivalence of processes using the axioms from Ref. 1 and Ref. 9, use the linearization algorithm of Ref. 9 to
Case Studies in the Hybrid Process Algebra HyPA

rewrite most HyPA specifications into linear representation, apply the safety verification method described in Ref. 8, or simulate the system using the HyPA simulator (Ref. 11).

Another possibility is to translate a restricted class of HyPA specifications into the input languages of existing formal verification tools. For instance, we can use the HyTech model checker as a verification engine for HyPA specifications, by translating them formally to the theory of hybrid automata that is the input language of HyTech.\(^\text{10}\)

Acknowledgments

The authors would like to thank Jos Baeten, Bert van Beek, MohammadReza Mousavi, Koos Rooda, and Ramon Schiffelers for many stimulating and helpful discussions. The authors also would like to thank Rogier Schouten for validating some of the models using the HyPA simulator prototype that he has developed.

References