Application of Bhave Toolset for System Control and Electronic System Design

K.L. Man[*], T. Krilavičius[†], C. Chen[‡] and H.L. Leung[§]

Abstract— **Behavioural Hybrid Process Calculus (BHPC) is a formalism for modelling and analysis of hybrid systems combining process algebras and the behavioural approach for modelling of instantaneous changes and continuous evolution. BHPC is supported by Bhave toolset, containing a tool for a novel way of visualisation of hybrid systems simulations msp-svg and a new version of hybrid simulator. We present the latest developments of Bhave toolset and apply it for case studies of system control and mixed-signal systems design.**

Keywords: formal methods, electronics, hybrid systems, simulation

1 Introduction

Process algebras/calculi [2,11,19] are formal languages in *Computer Science* that have formal syntax and semantics for specifying and reasoning about different systems. In simple words, process algebras are theoretical frameworks for formal specification and analysis of the behaviour of various systems. Serious efforts have been made in the past to deal with various systems (e.g. *discrete event systems* [20,25], *real-time systems* [10,12,26] and *hybrid systems* [1, 3, 5, 6, 17, 27]) 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.

Hybrid systems are systems that exhibit both discrete and continuous behaviour. Such systems have proved fruitful in a great diversity of engineering application areas including air-traffic control, automated manufacturing, chemical process control and system control. On the other hand, mathematically, the behaviour of electronic system design (e.g. digital, analog and mixed-signal design) can be described by discrete variables, continuous variables and a set of differential equations, whereas switching-modes can be used for modelling mixed models (i.e. mixed-signal design). Due to all these, digital, analog and mixed-signal design can be mathematically described as hybrid systems (with various level of abstraction) by nature.

Computer simulation is a powerful tool for analysing and optimising real-world systems with a wide range of successful applications. It provides an appealing approach for the analysis of dynamic behaviour of processes and helps decision makers identify different possible options by analysing enormous amounts of data.

Behavioural Hybrid Process Calculus (BHPC) [17] is a hybrid process algebra which was specifically designed for the description of the dynamic behaviour of hybrid systems along with a powerful simulator called BHAVE TOOLSET. Currently, simulation results obtained by means of the BHPC simulator can also be visualised and analysed via *Message Sequence Plots* (MSP) [17].

In this paper, we first present the latest development of BHAVE TOOLSET. Through case studies, we show the use of BHAVE TOOLSET for addressing several aspects of system control and mixed-signal design. Related work of the research activities presented in this paper can be found at [17, 18].

2 Behavioural Hybrid Process Algebra

One of the useful techniques for simulation of hybrid systems that includes continuous evolution and discrete changes, is Behavioural Hybrid Process Calculus (BHPC) [4, 17], an extension of classical process algebra that is suitable for the modelling and analysis of continuous and hybrid dynamical systems and can be seen as a generalisation of the behavioural approach [22] in a hybrid setting. The main strengths of the BHPC are the following.

Sound mathematical foundations. BHPC has sound mathematical foundations. It means that rigorous reasoning can be applied to investigate diverse properties of models.

Behavioural approach. In BHPC continuous evolution is defined in the behavioural setting [22] making it more general in contrast to other hybrid process algebras (Hybrid χ [27], HyPA [6], ACP^{srt}_{hs} [3]), i.e. it

[*]Xi'an Jiaotong-Liverpool, University (XJTLU), China, e-mail: ka.man@xjtlu.edu.cn.
[†]Vytautas Magnus University (VDU), Kaunas, Lithuania, e-mail: t.krilavicius@if.vdu.lt.
[‡]Global Institute of Software Technology, Suzhou, China, e-mail: catherinechenchen@gmail.com.
[§]Solari, Hong Kong, e-mail: sales@solari-hk.com.

is defined using trajectories (solutions of differential equations is one of ways of defining trajectories), not just (solutions of) differential equations.

Separation of concerns. Continuous and discrete behaviours are specified orthogonally, therefore they can be changed and analysed separately as well as in hybrid setting.

Bisimulation is congruence in BHPC, i.e. substituting bisimilar (processes, that exhibit the same observable behaviour up-to the branching structure) does not change behaviour of the system.

Tools support. BHPC is supported by Bhave toolset, see Section 3.

We present main ideas of the BHPC in this section, see [17] for the details.

Trajectories We define trajectories over bounded time intervals $(0,t]$, and map to a *signal space* $\mathbb{W} = (W_1 \times \cdots \times W_n, (q_1, \ldots, q_n))$. Components of the signal space $W \in \mathcal{W}$ correspond to the different aspects of the continuous-time behaviour, such as current or voltage, and are associated with *trajectory qualifiers* $q_i \in \mathcal{T}$ identifying them. A *trajectory* in signal space $\mathbb{W}$ is a function $\varphi : (0,t] \to W_1 \times \cdots \times W_n$, where $t \in \mathbb{R}_+$ is the duration of the trajectory. We define conditions on the *end-points* of trajectories or the *exit conditions*. $\Downarrow$ denotes such conditions, as the restrictions on the set of trajectories: $\Phi \Downarrow \mathcal{P}red_{\text{exit}} = \{\varphi : (0,u] \to W_1, \ldots, W_n \in \Phi \mid \mathcal{P}red_{\text{exit}}(\varphi(u))\}$, where u is a time parameter, Φ is a set of trajectories and $\mathcal{P}red_{\text{exit}}(\varphi(u))$ is a predicate that defines restrictions. The set of trajectories Φ can be defined in different ways, e.g. by ODE/DAE. See [17] for the formal treatment.

Hybrid transition system All behaviours of BHPC specification are defined by a *hybrid transition system* $HTS = \langle S, \mathcal{A}, \to, \mathbb{W}, \Phi, \to_c \rangle$

- S is a state space.
- $\mathcal{A}$ is a *finite set of (discrete) actions names.*
- $\to \subseteq S \times \mathcal{A} \times S$ is a *discrete transition relations*, where $\mathsf{a} \in \mathcal{A}$. We will denote it $s \xrightarrow{\mathsf{a}} s'$.
- $\mathbb{W}$ is a *signal space.*
- Φ is a *set of trajectories.*
- $\to_c \subseteq S \times \Phi \times S$ is a *continuous transition relation*, where $\varphi \in \Phi$ are trajectories. We will denote continuous transitions $s \xrightarrow{\varphi} s'$ for the convenience.

Language A core language is used for defining evolution and interaction of systems

$$B ::= \mathbf{0} \;\Big|\; \mathsf{a} \,.\, B \;\Big|\; [f \mid \Phi] \,.\, B \;\Big|\; \sum_{i \in I} B_i \;\Big|\; B \parallel_A^H B \;\Big|\; P$$

We will require a *consistent signal flow*, i.e. only the parallel composition is allowed to change the set of trajectory qualifiers in the process.

Only a subset of complete language is introduced, see [17] for auxiliary operators, such as renaming or hiding. Moreover, other operators can be defined on top of the core language for convenience. We demonstrate it by introducing *parametrised action prefix* and *guard*.

Stop 0 is the process that does not exhibit any behaviour.

Action prefix $\mathsf{a} \,.\, B$ performs a and continues as B. A special *silent action* τ defines directly unobservable behaviour, and is usually used to specify a non-determinism (e.g. as *internal actions* in [19, p. 37–43]).

We will use parametrisation of action prefix as in [19, p. 53–58] $\mathsf{a}(v : V) \,.\, B(v) = \sum_{v \in V} \mathsf{a}(v) \,.\, B(v)$.

Trajectory prefix $[f \mid \Phi] \,.\, B\,(f)$, where f is a trajectory variable, starts with a trajectory or a prefix of a trajectory from the set of trajectories Φ. If a trajectory or a part of it was taken and there exists a continuation of the trajectory, then the system can continue with a trajectory from the set of such continuations. If a whole trajectory (e.g., as defined by exit conditions) was taken, then the system can continue with B.

Choice $\sum\{B(v) \mid v \in I\}$ is a generalised *nondeterministic* choice of processes (I is an arbitrary index set). It chooses before taking an action prefix or trajectory prefix. Binary version of choice is denoted by $B_1 + B_2$.

Parallel composition $B_1 \parallel_A^H B_2$, where A and H are sets of synchronising action names and trajectory qualifiers, respectively, models the behaviour of two parallel processes. Synchronisation on actions has an interleaving semantics. Trajectory prefixes evolve only in parallel, and only if the evolution of the coinciding trajectory qualifiers is equal.

Recursions allows defining processes in terms of each other, as in the equation $P = B$, where P is the process identifier and B is a process expression that may only contain actions and signal types of B.

Guard $\langle \mathcal{P}red \rangle$ operator evaluates $\mathcal{P}red$ conditions, and if they are not satisfied, stops the progress of the process.

$$\langle \mathcal{P}red(\mathbf{x}) \rangle \,.\, B(\mathbf{x}) = \sum_{\mathbf{w} \models \mathcal{P}red(\mathbf{w})} B\,(\mathbf{w})$$

Here $\mathbf{x}$ are process parameters (variables).

Strong bisimulation for hybrid transition systems requires both systems to be able to execute the same trajectories and actions and to have the same branching structure.

The *hybrid strong bisimulation relation* (equivalence) defined for the HTS is a congruence relation w.r.t. all operations defined above [17]. Hence, bisimilar components can be interchanged without changing systems behaviour, and that can be effectively employed while building and improving systems (models).

3 Bhave toolset

BHPC is supported by Bhave toolset [13]. The toolset allows modelling, simulation and visualisation of the hybrid models [14]. It consists of several tools.

Discrete Bhave [16] allows discrete simulation of the BHPC specifications.

Bhave simulator allows hybrid simulation of BHPC specifications. Current version supports a subset of BHPC. A snapshot of the system with examples is available from `bhpc-simulator.sourceforge.net`.

BHPC2Mod can translate a restricted set of BHPC models to Modelica [9] language, and then simulate them using Dymola [8] or OpenModelica [21]. However, because Modelica does not have formal semantics, translation does not necessary preserves all the properties. Moreover, parallel composition is not translated [28].

msp-svg is a visualisation tool that uses Message Sequence Plots (MSP) [17, 24] approach for visualising hybrid evolution. It is available at `http://msp-svg.sourceforge.net/`. See Section 3.1 for details about MSP-SVG.

The current versions of both tools (BHAVE and MSP-SVG) are built not just as a prototypes, but also as a hybrid "sand-box", a place to experiment with BHPC and related developments. Architecture and implementation of the both tools allows accommodating diverse changes and test the algorithms developed for BHPC or other (hybrid) process algebras or MSP-based visualisation techniques comparatively easy.

Our plans include further development of the process algebra and BHAVE TOOLSET. We are planning to improve and extend BHAVE and integrate it with MSP-SVG.

3.1 Visualisation of Hybrid Evolutions

Simulation results usually visualise the evolution of the system in time. Event traces or message sequence charts (MSC) [23] adequately represent discrete system behaviour, and graphs are convenient for the ordinary continuous systems. However, in hybrid systems we have both the evolution of system variables and events. Hence, a combined view is crucial to fully analyse hybrid system behaviour. See [17, p. 118-124] for the details.

We believe, that Message Sequence Plots (MSP) [17, p. 118-124] contains all necessary components for adequate visualisation of hybrid systems. It has two compounds: message-sequence charts rotated 90° combined with plots. We explain MSP by an example depicted in Figure 1.

Plots over time-lines show continuous-time evolution.

A legend allows selecting qualifiers of interest, that are depicted in the plot. If several processes evolve concurrently, the synchronising qualifiers appear for both processes. In Figure 1 qualifiers $qual_1, qual_2, qual_3$ and qual_4 are depicted. $\mathbf{Process}_\mathrm{i}$ is related with qualifiers $qual_1, qual_2$ and $qual_3$, and only qualifiers $qual_1$ and $qual_3$ are selected to be visible. $\mathbf{Process}_\mathrm{j}$ is related with qualifiers $qual_1, qual_2$ and $qual_4$, and qualifiers $qual_2$ and $qual_4$ are selected to be visible.

Single horizontal lines connected to the corresponding boxes with process identifiers, represent processes and the *time-line* (or *life-line* in MSC terminology). Time is assumed to flow to the right along each time-line at the same speed. $\mathbf{Process}_\mathrm{i}$ and $\mathbf{Process}_\mathrm{j}$ are represented by the horizontal lines and boxes with processes identifiers in the example.

Labelled vertical lines going across time-lines represent communication, i.e. (parameterised) action prefixes in BHPC. Notice that we use simple lines instead of arrows, because communication in BHPC is not directed. Communication of $\mathbf{Process}_\mathrm{i}$ and $\mathbf{Process}_\mathrm{j}$ consists of actions $\mathsf{act}_1, \mathsf{act}_2$ and act_3.

Triple horizontal time-lines depict suspension of the time-flow. Single actions are placed on the time-line at the time that relates to their moment of occurrence. A sequence of actions occurs at one moment in time, when there is no continuous behaviour between the actions. We suspend the flow of time to allow insight in the ordering of these actions. In the example, suspension of the time is depicted on the time-line as three parallel solid lines.

Figure 1 contains all information that would be available in an ordinary plot. Correspondingly, all information that is visible in message sequence charts, is also visible in MSP. Furthermore, in MSP all processes and communication between them are visualised. The proposed technique can be easily adopted to other hybrid system

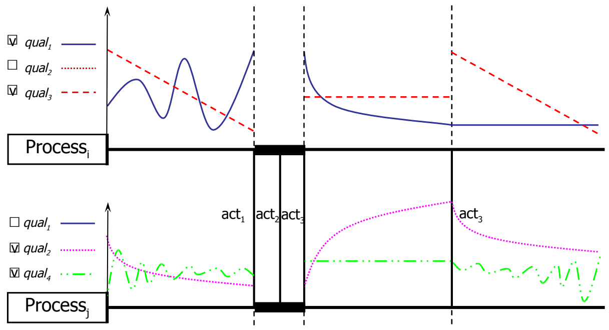

Figure 1: MSP example.

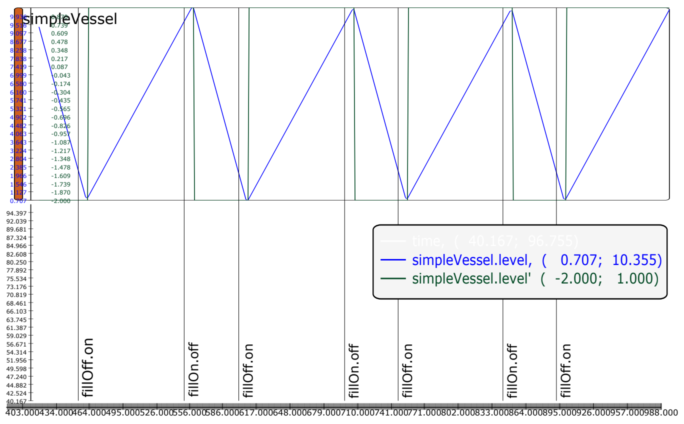

Figure 2: Simulation of the simple vessel.

modelling frameworks with minimal changes, e.g., if communication is directed, arrows can be used to depict it.

Additional notation, such as decorating the results with process expressions (e.g. action and trajectory prefixes), adding recursive calls, providing information about renaming of qualifiers in the legend, forking of processes to depict parallelism. See [15] for details.

Figure 2 depicts an evolution of the simple vessel simulation 4.1 using a proof of concept tool MSP-SVG. Executable and source code of MSP-SVG are available at `sourceforge.net/projects/msp-svg/`.

4 Application of BHPC

4.1 Simple Vessel

Simple vessel models a system with a constant outflow of fluid (l_{out}) and controlled inflow ($l_{\text{in}} \geq l_{\text{out}}$). Level of the fluid should be maintained in a certain interval $[l_{\min}, l_{\max}]$. Inflow is controlled by opening (open) and closing (close) a valve.

We provide a model in BHPC that simulates behaviour of such systems.

```
SimpleVessel(level) =
OutFlow(level)[simpleVessel.level/level];

InFlow(level) =
      [  level' =  (3 - 2) |
       (level - 10 - 0.5 * rand()) ]
      . off
      . OutFlow(level)[InFlow.level/level];

OutFlow(level) =
      [  level' = (0 - 2) |
       (level - 1 + 0.5 * rand()) ]
      . on
      . InFlow(level)[OutFlow.level/level];
```

It consists of main process **SimpleVessel** and two subprocesses **SimpleVessel**, **InFlow** and **OutFlow**, that, respectively, simulate inflow and outflow modes. Switching occurs in the predefined ranges, i.e. the system switches to inflow mode when *level* $\in [l_{\min}, l_{\text{on}}]$ (in our case *level* $\in [0.5, 1.5]$) and to outflow mode when *level* $\in [l_{\text{off}}, l_{\max}]$ ($[9.5, 10.5]$). Ranges allow to model measuring devices errors and delays. We model non determinism using *rand()* function, but in the future we are planning to introduce other techniques [17, p.115-117] for it. Inflow and outflow are, respectively 3 and -2 units of fluid per time unit. Simulation results are depicted in Figure 2. Slanted (blue) lines depict growing and decreasing level. Inflow and outflow speed (derivatives of fluid level change) are depicted by black horizontal lines, respectively, at level 1 and -2 (notice, different axes) in the upper part of the figure. Actions are depicted by vertical lines, and shown in the lower part of the figure.

4.2 Application of BHPC for Tunnel Diode Modelling

We present results of experiments with the tunnel-diode circuit from [7]. The circuit is depicted in Figure 3. State of the system is defined by two state variables: the current through the inductor I_l and the voltage across the diode V_d. Behaviour is defined by differential equations

$$\dot{I}_l = \frac{1}{C}\left(-I_d(V_d) + I_l\right) \tag{1}$$

$$\dot{V}_d = \frac{1}{L}\left(E - RI_l - V_d\right). \tag{2}$$

where

$$I_d(V_d) = V_d^3 - 1.5V_d^2 + 0.6V_d$$

defines non-linear characteristics of the tunnel diode.

We provide a simple model of the system, that consists of one process, namely **TunnelDiode**, that contains a trajectory prefix defining evolution of the circuit over time. Exit conditions of the trajectory prefix define expected intervals for the current I_l and voltage V_d.

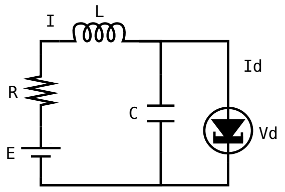

Figure 3: Tunnel diode circuit.

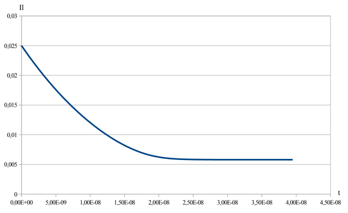

Figure 4: Non-oscillating tunnel diode circuit: current

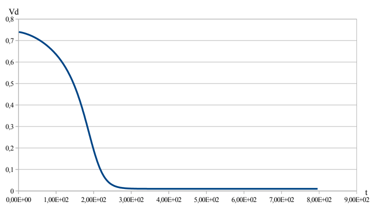

Figure 5: Non-oscillating tunnel diode circuit: voltage

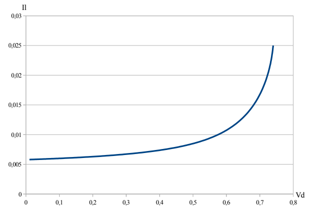

Figure 6: Non-oscillating tunnel diode circuit: current vs voltage

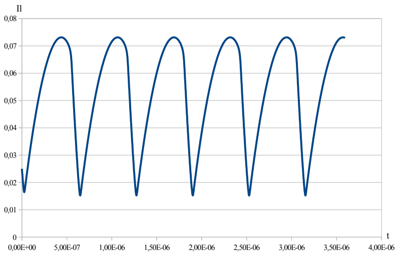

Figure 7: Oscillating tunnel diode circuit: current

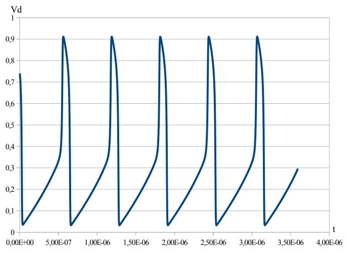

Figure 8: Oscillating tunnel diode circuit: voltage

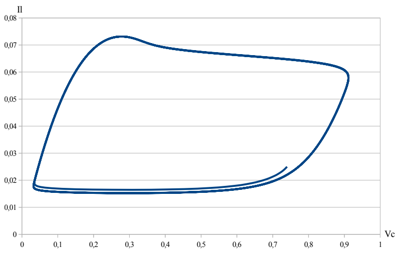

Figure 9: Ooscillating tunnel diode circuit: current vs voltage

```
TunnelDiode(Il, Vd) =
     [ Vd’ = (1 / C) * (Il -
      (Vd * Vd * Vd  -
       1.5 * Vd * Vd + 0.6 * Vd));
       Il’ = (1 / L) * (E - R * Il - Vd)
      | (Vd - 0); (Vd - 0.92);
        (Il - 0); (Il - 0.08) ]
      . stop;
```

We performed simulation with two different sets of parameters.

	Non-oscillating	Oscillating
$C\ (F)$	10^{-9}	10^{-9}
$L\ (H)$	10^{-6}	10^{-6}
$E\ (V)$	0.3	10^{-9}
$R\ (\Omega)$	50	0.3

Initial values: $I_l = 0.025A$ and $V_d = 0.74V$. Notice, that only voltage differs.

As expected, in the first case simulation stabilises in the equilibrium. Figures 4, 5 and 6 depict the current, voltage and current vs voltage of the non-oscillating tunnel diode circuit, respectively.

With the second set of parameters we get an oscillating system in the expected intervals. Simulation of the oscillating tunnel diode circuit is depicted in Figures 7, 8 and 9.

5 Conclusions

Modelling and analysis of the tunnel diode circuit shows that Behavioural Hybrid Process Calculus and its toolset can be used for the formal specification and simulation of electronic systems.

Application of the proof-of-concept MSP-SVG tool for visualisation of simple vessel simulation demonstrates advantages of the Message Sequence Plots over simple plots, because not only switching points are visible, but a cause (a related event) as well.

Our future work will focus on several aspects.

- Application of the toolset to analog and mixed electronic systems.
- Modular specifications of diverse circuits, i.e. modelling of circuits as parallel components, e.g. modelling tunnel diode circuit as parallel interconnection of capacitor, inductor, resistor, tunnel diode and power source.
- Improvements of the modelling language. Currently we use the language, that consists only of basic constructs. We are planning to add some syntactical constructs for convenience.
- Improvements of tools. We are planning to improve simulator and MSP-SVG tools, integrate them.

References

[1] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P. H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *TCS*, 138(1):3–34, 1995.

[2] J. C. M. Baeten and W. P. Weijland. *Process Algebra*, volume 18 of *Camb. Tracts in TCS*. Camb. Univ. Press, Cambridge, UK, 1990.

[3] J. A. Bergstra and C. A. Middelburg. Process Algebra for Hybrid Systems. *TCS*, 335(2/3):215–280, 2005.

[4] E. Brinksma, T. Krilavičius, and Y. S. Usenko. Process Algebraic Approach to Hybrid Systems. In *Proc. of 16th IFAC World Congress*, pages 1–6, Prague, Czech Rep., July 2005.

[5] L. P. Carloni, R. Passerone, A. Pinto, and A. L. Sangiovanni-Vincentelli. Language and Tools for Hybrid Systems Design. *J of Found. and Trends*, 1:1–177, 2005.

[6] P. J. L. Cuijpers and M. A. Reniers. Hybrid Process Algebra. *JLAP*, 62(2):191–245, 2005.

[7] W. Denman. Formal Verification of Analog and Mixed Signal Designs. Master's thesis, Concordia Univ., 2009.

[8] Dynasim, 2006. Last accessed: 2006 May 11.

[9] P. Fritzson and V. Engelson. Modelica - A Unified Object-Oriented Language for System Modelling and Simulation, 1998.

[10] M. Geilen. *Formal Techniques for Verification of Complex Real-time Systems*. PhD thesis, Tech. Univ. of Eindhoven, 2001.

[11] C. A. R. Hoare. *Communicating Sequential Processes*. Prent.-Hall, 1985.

[12] J. Hooman. *Specification and Compositional Verification of Real-Time Systems*. Springer, 1991.

[13] T. Krilavičius. Bhave :: Simulation of Hybrid Systems, 2006.

[14] T. Krilavičius. Simulation of Mechatronic Systems Using Behavioural Hybrid Process Calculus. *Electron. and Elec. Eng.*, 1(81):45–48, 2008.

[15] T. Krilavičius and K.L. Man. *Intelligent Automation and Computer Engineering*, chapter Behavioural Hybrid Process Calculus for Modelling and Analysis of Hybrid and Electronic Systems. Springer, 2009.

[16] T. Krilavičius and H. Schonenberg. Discrete Simulation of Behavioural Hybrid Process Calculus. In P. M. E. Bra and J. J. van Wijk, editors, *IFM2005 Doctoral Symposium*, pages 33–38, Eindhoven, Netherlands, November 2005. Tech. Univ. of Eindhoven, Dept. of Math. and CS.

[17] T. Krilavčius. *Hybrid Techniques for Hybrid Systems*. PhD thesis, Univ. of Twente, 2006.

[18] K. L. Man and M. P. Schellekens. *Current Trends in Intelligent Systems and Computer Engineering*, chapter Interoperability of Performance and Functional Analysis for Electronic System Designs in Behavioural Hybrid Process Calculus (BHPC). Springer, 2008.

[19] R. Milner. *Communication and Concurrency*. Pren.-Hall, 1989.

[20] G. Naumoski and W. Alberts. *A Discrete-Event Simulator for Systems Engineering*. PhD thesis, Tech. Univ. of Eindhoven, 1998.

[21] OpenModelica System website. OpenModelica System, 2009. `http://www.ida.liu.se/~pelab/modelica/OpenModelica.html`.

[22] J. W. Polderman and J. C. Willems. *Introduction to Mathematical Systems Theory: a behavioral approach*. Springer, 1998.

[23] E. Rudolph, P. Graubmann, and J. Grabowski. Tutorial on Message Sequence Charts. *Comput. Netw. ISDN Syst.*, 28(12):1629–1641, 1996.

[24] M. H. Schonenberg. Discrete Simulation of Behavioural Hybrid Process Algebra. Master's thesis, Univ. of Twente, 2006. Master thesis.

[25] D. A. van Beek, S. H. F. Gordijn, and J. E. Rooda. Integrating Continuous-Time and Discrete-Event Concepts in Modelling and Simulation of Manufacturing Machines. *Sim. Pract. and Theory*, 5(7-8):653–669, 1997.

[26] D. A. van Beek, K. L. Man, M. A. Reniers, J. E. Rooda, and R. R. H. Schiffelers. Syntax and semantics of timed Chi. Technical Report CS-Report 05-09, Tech. Univ. of Eindhoven, Dept. of CS, The Netherlands, 2005.

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

[28] A. van Putten. Behavioural Hybrid Process Calculus Parser and Translator to Modelica. Master's thesis, Univ. of Twente, 2006.