Formal Verification of the Quasi-Static Behavior of Mixed-Signal Circuits by Property Checking

Martin Freibothe, Jens Schönherr, Bernd Straube

Fraunhofer Institute for Integrated Circuits (IIS) Erlangen
Design Automation Department EAS
Dresden, Germany

Abstract

This paper proposes a verification flow for mixed-signal circuits. The presented flow is based on ‘bounded model checking’, a formal verification method. The behavior of the analog parts of a mixed-signal circuit is described with the help of rational numbers within the circuit description and in the properties, respectively. Our implemented Property-Checker checks formal properties for a given mixed-signal circuit design over a finite interval of time. The internal representation of the rational numbers has an almost arbitrary accuracy. By using the presented flow, the quasi-static behavior of a mixed-signal circuit can be exhaustively verified.

Keywords: bounded model checking, mixed-signal, quasi-static

1 Introduction

In today’s circuit designs, more and more analog parts are integrated on system-on-chips (SoCs) and ASICs. Usually in such designs the analog parts comprise only a comparatively small fraction of the overall chip area. On the contrary, a large part of the development time of a mixed-signal circuit is spent on the analog parts. Digital circuit designs can be formally verified in order to find design errors. In contrast to simulation-based verification,
formal verification techniques reach a 100% coverage of the functionality of a circuit design. However, in current verification flows applied to mixed-signal circuits, the analog and the digital parts are verified separately by simulation and formal verification, respectively.

The verification of the overall behavior of the circuit and, hence, the connection of the analog and digital parts is then carried out based on simulation. All simulation-based verification methods have the disadvantage of requiring a test-bench and simulation stimuli. Providing both of them requires a high manual effort. Despite this effort, only a relatively low coverage of the functionality of the circuit can be achieved.

In this work, a verification flow for the formal verification of the quasi-static behavior of mixed-signal circuits is presented. The proposed approach is based on the method of ‘bounded model checking’. This method is state-of-the-art for the verification of digital circuit designs. We extended this method in order to allow its application to analog circuits. Nowadays most applications of ‘bounded model checking’ are based on boolean validity checking of formulae. In the presented flow however, we additionally allow the usage of arithmetic expressions in the formulae describing the behavior and the specification of the circuit. Using this extension, we can now apply this verification flow to circuits that are composed of analog and digital parts, i.e. mixed-signal circuits.

2 State of the art

Different approaches to the formal verification of mixed-signal circuits are presented in the literature.

The authors of [10,11] propose a method to verify mixed-signal circuit designs on the functional level. The functional behavior of the analog parts of the design is described by symbolic models that represent the quasi-static behavior of the circuit. In those models, analog voltages are represented by integer numbers of a finite interval. These numbers are encoded using boolean vectors of finite lengths. The finite lengths of those vectors limit the accuracy of this verification flow on the one hand and introduce the problem of arithmetic overflows on the other hand. These problems significantly limit the usability of this approach.

Qualitative modelling [6,5,15] is based on the representation of a circuit’s analog parts by differential algebraic equations (DAEs). Qualitative equations (confluences) are derived from the DAEs and are used to represent an abstracted behavior of the circuit. Real numbers that represent analog voltages and currents are abstracted by the three-element set $M = \{-, 0, +\}$ and thus reduced to their sign. All statements that can be derived from this model are
limited to the qualitative values of the set $M$. As recognized by [15], even simple linear systems cannot be modelled adequately using this approach.

‘Timed automata’ [18] are not suitable to represent mixed-signal circuits because of their restrictions on the values that can be assigned to variables. However, an extension of this approach are hybrid automata [1,12]. They are used for modelling systems that consist of discrete parts and continuous parts. Simple mixed-signal circuits can be represented by such deterministic automata. Verifying a circuit design is then carried out by performing a reachability analysis over the state-space of the automaton. Even under strong restrictions, such as piecewise constant derivatives of the variables, the reachability problem remains undecidable for this class of automata, for the state-space is infinite in general [13,14]. Recent approaches to the verification of hybrid automata utilize different techniques to approximate the reachable state sets. Standard model checking algorithms are then applied to these finite-state abstractions [8,9]. The approach proposed in [3] suggests to perform a reachability analysis on a certain class of hybrid systems that can be represented as mixed logical dynamical or piecewise affine systems to get a piecewise linear approximation of the exact solution. Drawbacks of these approaches are that they introduce an approximation error, are not sound [9] or their performance depends on the shape of the reachable sets of states [3].

A concept for an approach to ‘bounded model checking’ and inductive verification of hybrid discrete-continuous systems is presented in [2]. A hybrid discrete-continuous system description is translated into an instance of a combined SAT/arithmetic problem. The result of this translation represents a piecewise linear over-approximation of the actual problem. It is proposed to combine different verification and analysis approaches to SAT- and optimization problems such as MILP (‘mixed integer linear programming’) and planning from AI in order to improve the performance of the verification process for hybrid systems. Counter-examples that may arise as the result of the verification process are going to be back-translated into the original discrete-continuous domain. One major disadvantage of this approach is the piecewise linear over-approximation of the actual problem. Any generated counter-example is thus again just an over-approximation and its back-translation into valid paths of the hybrid automaton is not trivial. This translation task is transformed into a set of search-problems, whereas the possible range of input values for a state transition determines the search space that has to be traversed. Because of the usage of integer numbers for representing the analog parts of a mixed-signal circuit in this approach, one has to deal with problems such as limited accuracy and arithmetic overflows. These problems limit the usability of this approach for the verification of analog components.
In [16] an approach to the semi-formal verification of the quasi-static behavior of mixed-signal circuits is presented. The proposed approach is based on SAT-based property-checking. Starting point of the described verification flow is the quasi-static behavioral description of a mixed-signal circuit in VHDL. In such a circuit description, VHDL’s floating point type \texttt{real} is used to characterize the analog behavior. Given such a mixed-signal circuit description, a verification-oriented model is derived for which SAT-based property-checking is carried out. In the verification-oriented model the floating point type \texttt{real} used to describe analog behavior is abstracted by integer numbers of a finite interval. Such a model is represented by a finite Mealy automaton and can thus be verified using well-established formal verification methods. However, the verification-oriented model is only a quantized representation of the VHDL behavioral description of the circuit. Possible overflows due to arithmetic operations are recognized during the verification process. In such cases, the quantization parameters have to be adjusted, and the verification has to be re-run. The higher the accuracy of the model is chosen, the more complex is the resulting verification problem. Disadvantages of this verification flow are the utilization of an abstract (so-called ‘verification-oriented’) model for the verification on the one and the quantization errors that are introduced when the verification-oriented model is created on the other hand.

3 Verification flow

The task of the verification is to check whether a circuit design meets its specification or not. For checking if a given mixed-signal circuit is free of design errors, the proposed verification flow requires a formal specification. Such a specification has to be provided as a set of formal properties. These formal properties are manually derived from the verbal specification of the circuit. The presented flow for the formal verification of mixed-signal circuit designs is based on the principle of ‘bounded model checking’. This verification technique checks formal properties for a given circuit over a finite interval of time. We have implemented a prototype, the Property-Checker, for the presented verification flow. The Property-Checker utilizes the validity checker SVC [17] to check the validity of formulae. SVC supports arithmetic expressions in the formulae to be checked.

The overall formal verification flow using the Property-Checker is depicted in Figure 1 on the left. The prototype expects the mixed-signal circuit design as an automaton and the formal specification in the form of properties as input. As the result the Property-Checker returns the information, whether a given property holds for that mixed-signal circuit or not.
Our proposed flow for checking the validity of a property for a mixed-signal circuit using the validity checker SVC is depicted in Figure 1 on the right. The automaton representing the mixed-signal circuit is transformed into its internal representation as an iterative array by using the assumptions from the property to determine the length of the iterative array. Again using the assumptions from the property, a formula of first-order logic is derived from this iterative array in a subsequent step. The validity of this formula represents the validity of the property for the mixed-signal circuit design. The formula is handed over to SVC where its validity is checked. As a result, one gets the information whether the property holds for the circuit or not. In case the property fails to hold, a counter-example will be generated that can be used for debugging.

3.1 Circuit description

We believe that the representation of analog behavior in terms of DAEs is too detailed and thus not suitable for the formal verification of the interaction of the analog and digital components of a mixed-signal circuit. On the contrary, our proposed verification flow is based on a more abstract view of a circuit’s behavior. The mixed-signal circuit is described as a deterministic abstract Mealy automaton as input for the Property-Checker. Due to the representation of the circuit as automaton, only its quasi-static, steady-state behavior can be described within the model. We chose a simple XML syntax
as description language for the mixed-signal circuit design in the form of an automaton as this requires a rather simple parser to be developed.

Such an automaton is described by internal variables, input and state variables as well as the state transition functions \( \delta \) and the output functions \( \lambda \). For these descriptions the types \( \text{bool} \) and \( \text{rat} \) are available. These two types can be used for defining constants, variables and expressions of two-valued logic and rational numbers, respectively. Additionally, language elements such as \( \text{if-then-else} \) statements, the usual boolean operations and the arithmetic operations like addition, subtraction, multiplication and division with constant divisors over the range of rational numbers are available. As it is allowed to use rational numbers in the automaton’s description, the set of input symbols of such an automaton is infinite. As a result, the automata representing mixed-signal circuits in the presented approach are infinite.

The digital components’ behavior of a mixed-signal circuit is represented by expressions over the type \( \text{bool} \) in such a circuit description. By using rational numbers (type: \( \text{rat} \)), one can additionally describe the behavior of analog components. Using rational numbers for the description of analog behavior is sufficiently precise, as the rational numbers are dense.

Figure 2 shows an example for a sequential mixed-signal circuit. The XML behavioral description of this circuit is depicted in Figure 3. Such an XML description is used as input for the Property-Checker. Depending on the value of the digital input \( i_{\text{mode}} \) of the mixed-signal circuit, one of the two analog inputs \( \text{in0} \) or \( \text{in1} \) is selected in the next clock cycle. The voltage at the selected analog input will be amplified by the factor of \( (1 + 1000\Omega/2000\Omega) \) or by the factor of \( (1+1000\Omega/500\Omega) \), respectively, depending on the value of the digital...
input $v_{\text{mode}}$ during the previous clock cycle. The amplified voltage can be measured at the circuit’s analog output $output$.

In the XML description of the automaton, an output function $\lambda_i$ defining the value of an internal variable or an output variable is specified after the keyword $\text{def}$. The state transition function $\delta_i$ that is specified after the keyword $\text{next}$ in the description of the automaton returns for a state-variable $s_i$ its corresponding value in the next clock-cycle.

$$\begin{align*}
\text{<module name = "Example">} \\
\text{<in name = "v_mode" type = "bool" />} \\
\text{<in name = "i_mode" type = "bool" />} \\
\text{<in name = "in0" type = "rat" />} \\
\text{<in name = "in1" type = "rat" />} \\
\text{<out name = "output" type = "rat" def = "( o1 )" />} \\
\text{<statevar name = "v_mode_d" type = "bool" init = "( false )" next = "(v_mode)" />} \\
\text{<statevar name = "i_mode_d" type = "bool" init = "( false )" next = "(i_mode)" />} \\
\text{<internal name = "j1" type = "rat" def = "(ite (i_mode_d) in1 in0 )" />} \\
\text{<internal name = "r1" type = "rat" def = "(ite (v_mode_d) 500|1 2000|1)" />} \\
\text{<internal name = "o1" type = "rat" def = "(* j1 (/ (+ 1000|1 r1) r1))" />} \\
\text{</module>}
\end{align*}$$

Figure 3. XML behavioral description of the example mixed-signal circuit

### 3.2 Description of the properties

The formal specification of a given mixed-signal circuit design usually consists of a set of formal properties. If all formal properties are proved to be valid by the Property-Checker, the specification holds for the mixed-signal circuit. As input for the presented flow, the formal properties to be checked need to be provided in a language of discrete, linear temporal propositional logic.

A formal property basically consists of time-stamped assumptions about the mixed-signal circuit’s inputs, outputs, states and internal wires. The model that is used in our verification flow abstracts from continuous time to discrete time-points. In a property, time-points are represented by integer numbers $\mathbb{Z}$ of a finite interval. Thus it is only possible to specify the quasi-static, steady-state behavior of a mixed-signal circuit design in the properties. The bounds within our proposed verification flow performs the ‘bounded model checking’ are specified by the first and last point of time that is associated with the assumptions in the property.

The following language elements can be used for formulating assumptions about the behavior of a mixed-signal circuit:
(i) Boolean constants and variables

\[
\text{simple}_\text{bool} ::= \text{const}_\text{bool} | \text{var}_\text{bool}, \text{timepoint} \quad \text{timepoint} \in \mathbb{Z}
\]

(ii) Constants and variables representing rational numbers

\[
\text{simple}_\text{rat} ::= \text{const}_\text{rat} | \text{var}_\text{rat}, \text{timepoint} \quad \text{timepoint} \in \mathbb{Z}
\]

(iii) Boolean expressions

\[
\text{expr}_\text{bool} ::= \text{simple}_\text{bool} | \begin{array}{l}
(\text{expr}_\text{bool} \odot \text{expr}_\text{bool}) | \\
(\text{expr}_\text{rat} \star \text{expr}_\text{rat}) | \\
\text{if} (\text{expr}_\text{bool}) \text{then} \text{expr}_\text{bool} \text{else} \text{expr}_\text{bool} \text{end if}
\end{array}
\]

\[\diamond \in \{\land, \lor, \neg, \rightarrow\} \quad \star \in \{=, <, >, \leq, \geq\}\]

(iv) Expressions over the type representing the rational numbers \(\mathbb{Q}\)

\[
\text{expr}_\text{rat} ::= \text{simple}_\text{rat} | \begin{array}{l}
(\text{expr}_\text{rat} \odot \text{expr}_\text{rat}) | \\
(\text{expr}_\text{rat} / \text{const}_\text{rat}) | \\
\text{if} (\text{expr}_\text{bool}) \text{then} \text{expr}_\text{rat} \text{else} \text{expr}_\text{rat} \text{end if}
\end{array}
\]

\[\diamond \in \{+, -, \ast\}\]

There are the usual boolean operations available for describing the digital behavior of a mixed-signal circuit with the help of expressions over the type \text{bool}. The behavior of the analog components is described by assumptions using expressions representing constants and variables of the type of the rational numbers \text{rat}. For those expressions, the same arithmetic operations as in the description of the circuit can be used: addition, subtraction, multiplication and division with constant divisors. The subscript \text{timepoint} of a variable denotes the time-point this variable is associated with within the property.

An example for the definition of a formal property is given below. This property partly describes the behavior of the mixed-signal circuit depicted in Figure 2.

\[(\text{in}0_0 \geq 0.0) \land (\text{in}0_0 \leq 2.5)) \rightarrow (\text{output}_1 = \text{src}(\text{i.mode}_0) \ast \text{ampl}(\text{v.mode}_0))\] (1)

\[
s\text{rc}(a_i : \text{boolean}) ::= \text{if}(a_i = \text{false}) \text{then} \text{in}0_{i+1} \text{else} \text{in}1_{i+1} \text{end if} \]

(2)

\[
\text{ampl}(b_i : \text{boolean}) ::= \text{if}(b_i = \text{false}) \text{then} 1.5 \text{else} 3.0 \text{end if} \]

(3)

In general, a property is written as an implication. On its left side there are assumptions about the circuit’s inputs and states at the first time-point of the considered interval. The right hand side of such an implication usually contains assumptions about the outputs and next-states of the circuit.

The implication (1) represents the assumption about the behavior of the mixed-signal circuit to be verified. On the left side of the implication, it is assumed that the analog input voltage \text{in}0 is in the range between 0.0V and 2.5V at the time-point ‘0’. On the implication’s right side it is claimed that one clock cycle later, i.e. at the time-point ‘1’, the analog input that is selected
by the circuit’s digital input $i_{\text{mode}}$, is amplified with the appropriate factor and can be measured at the analog output $output$. In the given example, the verification window within the validity of the property is checked is denoted by the interval $[0, 1]$.

In the given property, (2) and (3) define two macros. The first one, $src$, either returns the analog input $in0$ or $in1$ in the next clock cycle, depending on the value of its parameter $a$. The second macro, $ampl$, either returns the rational number $1.5$ or $3.0$, depending on the value of its parameter $b$. This rational number represents an amplification factor.

### 3.3 Semantics of the properties

The verification window that is defined by a property determines the bounds within ‘bounded model checking’ is carried out by the Property-Checker. Within a property, a time-step is defined by the difference of two consecutive integer numbers $\text{timepoint}$. This is equal to one clock-cycle in the representation of the mixed-signal circuit as automaton. In the internal representation, the mixed-signal circuit is copied once for each time-point of the finite interval that is given by the verification window. This way, a combinatorial representation of a given sequential mixed-signal circuit, the so-called ‘iterative array’, is derived. The left bound of the verification window is always the time-point denoted by the integer number ‘0’. However, it is allowed to use negative time-stamps when writing a property. In such cases, the verification window is adjusted appropriately. At time-point ‘0’, all state variables are treated as inputs of the iterative array.

Figure 4 shows the representation of a sequential circuit as an iterative array within the interval $[0, z], z \in \mathbb{Z}$. Below each copy of the internal representation of the circuit there is depicted the time-point this copy represents within the iterative array. Based on the mixed-signal circuit’s representation as an iterative array, a formula of first-order logic that represents the validity of the property to be verified is derived. Such a formula is derived by substituting the variables from the property by the corresponding expressions from
the copy of the circuit that represents the same point of time as the variable to be substituted. For example, for the mixed-signal circuit described in section 3.1, the following expression would be substituted for the variable output at time-point '1', i.e. \( \text{output}_1 \): 

\[
\left( \left( \text{if} \left( i_{\text{mode}} = 0 \right) \text{ then } \text{in}_1 \text{ else } \text{in}_0 \text{ end if} \right) \cdot \left( \left( \text{if} \left( v_{\text{mode}} = 0 \right) \text{ then } \frac{500}{1} \text{ else } \frac{2000}{1} \text{ end if} \right) + \frac{1000}{1} \right) \right) / \left( \left( \text{if} \left( v_{\text{mode}} = 0 \right) \text{ then } \frac{500}{1} \text{ else } \frac{2000}{1} \text{ end if} \right) \right)
\]

The problem of 'state space explosion', that mainly arises when verifying more complex circuit designs, is avoided in the proposed approach. This is due to the fact that there is no reachability analysis carried out for the state variables of the automaton. Because of that reason, the results of the proposed verification flow suffer from possible false-negatives. As all state variables at the time-point '0' are treated as inputs of the iterative array, the mixed-signal circuit can be in any arbitrary state at this time-point. This also incorporates the possibility of the circuit being in a state that is unreachable under realistic circumstances. Because of that, any counter-example for a property that does not hold must be checked to make sure that it does not represent a false-negative. In case a false-negative is recognized, assumptions that restrict the values of state variables at the time-point '0' can be added to the property in order to avoid unreachable initial states.

3.4 Invocation of the validity checker

For every single formal property the mixed-signal circuit’s formal specification consists of, the Property-Checker derives a formula of first-order logic. The validity of this formula represents the validity of the corresponding property for the mixed-signal circuit. In the current version of the Property-Checker, the validity of formulae is checked using SVC. This piece of software was developed at the University of Stanford.

SVC is a validity checker that supports a decidable subset of first-order logic. It also allows the usage of arithmetic expressions in the formulae. Rational numbers which can be used in arithmetic expressions are represented using the GMP [7] library. The accuracy of the rational numbers’ representation is thus only limited by the amount of local memory that is available. Using the GMP library avoids the occurrence of arithmetic overflows. So far, SVC only supports the usage of linear inequalities and linear arithmetic expressions in formulae. If the formal property and the Mealy automaton representing the mixed-signal circuit are piecewise linear, then these requirements are met by the formula that is handed over to SVC by the Property-Checker. If the formula is proved to be valid, the property holds for the circuit. Oth-
erwise the property fails. In such cases a counter-example will be generated. The counter-example can be used to find the design error that caused the property to fail.

The validity checker SVC cannot handle formulae that contain arithmetic expressions for subtraction or division with constant divisors. Thus, a formula that contains such expressions is equivalently transformed into one that does not contain such expressions. The resulting formula is then used as input for the validity checker SVC.

The utilization of any other piece of software for validity checking of formulae is easily possible. Preliminary experiments have already been carried out using Mathematica and CVCL [4], a successor of SVC. CVCL additionally supports certain non-linearities in the formulae. Due to the limits of SVC, up to now only piecewise linear properties can be formally verified for mixed-signal circuits that are represented by piecewise linear Mealy automata with the help of the Property-Checker. These constraints can be weakened when other tools such as Mathematica are utilized for checking the validity of formulae.

4 Evaluation and conclusion

In contrast to other approaches in the field of mixed-signal verification, in the presented flow the behavior of analog components is represented using rational numbers. The internal representation of both, the mixed-signal circuit and the properties provides an almost arbitrary accuracy of rational numbers. The two main advantages of using rational numbers over integer numbers are on the one hand that rational numbers represent analog behavior with an arbitrary accuracy while the accuracy using integer numbers is limited and in any case introduces a quantization error. On the other hand the verification problem’s complexity exponentially grows using integer numbers the finer the discretization granularity is chosen such that the possible accuracy is very limited in practice. The problems of other approaches in terms of possible overflows caused by arithmetic operations and limited accuracy are avoided in the presented approach by representing rational numbers using the GMP library. This way, a verification flow that is well-established in the field of digital verification, ‘bounded model checking’, has been extended to enable its application to analog and mixed-signal circuits. Using this approach, the quasi-static behavior of mixed-signal circuits can be formally verified.

Considering two simple mixed-signal circuits as examples, the performance of the Property-Checker has been compared to the verification flow proposed in [16]. The approach that is presented there is based on boolean SAT check-
ing. The verification flow using the Property-Checker achieved a better performance in terms of run-time and memory consumption. At the same time, the accuracy of the representation of the analog components using rational numbers is arbitrarily high. On the contrary, the approach of [16] uses a limited accuracy that has to be specified by the user before the verification process and thus arithmetic overflows might occur. In such cases, the quantization parameters have to be adjusted. However, the run-time exponentially increases, with the accuracy that is chosen for modelling the analog behavior.

With the current version of the Property-Checker, only piecewise linear properties can be checked for piecewise linear mixed-signal circuits. In order to allow the verification of non-linear behavior, another tool than SVC for checking the validity of formulae must be used by the Property-Checker. It is easily possible to adopt the output format of the formula that is handed over to this tool by the Property-Checker to any other validity checker’s input format. It is also possible to extent the internal representation of the property and the mixed-signal circuit towards other types (e.g. real numbers) and operations on them (e.g. exponentiation and logarithm functions).

As many other digital and mixed-signal verification flows, the presented approach using the Property-Checker also abstracts from the continuous time to discrete time-points. Thus, only the steady-state, quasi-static behavior of a mixed-signal circuit can be verified using this verification flow. However, even if the quasi-static behavior of a circuit has been verified, there are often constraints for the transient response of a circuit, that cannot be verified with the current version of the Property-Checker. Hence, future work will focus on the verification of the dynamic behavior of mixed-signal circuits.

References

[1] R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-H. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In R. L. Grossmann, A. Nerode, A. P. Ravn, and H. Rischel, editors, Hybrid Systems, volume 736 of LNCS, pages 209–229. Springer, 1993.

[2] B. Becker, M. Behle, F. Eisenbrand, M. Fränzle, M. Herbsttritt, C. Herde, J. Hoffmann, D. Kröning, B. Nebel, I. Polian, and R. Wimmer. Bounded model checking and inductive verification of hybrid discrete-continuous systems. In ITG/GI/GMM-Workshop “Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen”, 2004.

[3] A. Bemporad, G. Ferrari-Trecate, and M. Morari. Observability and controllability of piecewise affine and hybrid systems. IEEE Trans. Automatic Control, 45:1864–1876, 2000.

[4] Cooperating validity checker lite (CVCL), version 1.1.0. http://verify.stanford.edu/CVCL/, 2004. University of Stanford.

[5] J. de Kleer. How circuits work. Artificial Inteligence, 24:205–280, 1984.
[6] J. de Kleer and J. S. Brown. A qualitative physics based on confluences. *Artificial Intelligence*, 24:7–83, 1984.

[7] Gnu multiple precision arithmetic library, version 4.1.2. http://www.swox.com/gmp/, 2003.

[8] S. Gupta, B.H. Krogh, and R.A. Rutenbar. Towards formal verification of analog designs. In *Computer Aided Design, 2004. ICCAD-2004. IEEE/ACM International Conference on*, pages 210–217, Nov. 2004.

[9] W. Hartong, L. Hedrich, and E. Barke. Model checking algorithms for analog verification. In *DAC’02: Proceedings of the 39th conference on Design automation*, pages 542–547. ACM Press, 2002.

[10] S. Hendricx and L. Claesen. A symbolic modelling approach for the formal verification of integrated mixed-mode systems. In *Proceedings of 3rd Designing Correct Circuits (DCC)*, September 1996.

[11] S. Hendricx and L. Claesen. A symbolic core approach to the formal verification of integrated mixed-mode applications. In *Proceedings European Design & Test Conference (ED&TC)*, pages 432–436. IEEE Computer Society, 1997.

[12] T. A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS 1996)*, pages 278–292, 1996.

[13] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What’s decidable about hybrid automata? In *Twenty-Seventh Annual ACM Symposium on Theory of Computing (STOC 1995)*, pages 373–382. ACM, 1995.

[14] T. A. Henzinger and J.-F. Raskin. Robust undecidability of timed and hybrid systems. In Nancy A. Lynch and Bruce H. Krogh, editors, *Hybrid Systems: Computation and Control, Third International Workshop (HSCC 2000)*, volume 1790 of *LNCS*, pages 145–159. Springer, 2000.

[15] J. Lunze. *Künstliche Intelligenz für Ingenieure*, volume II. R. Oldenbourg Verlag, München, Wien, 1995.

[16] J. Schönherr, M. Freibothe, B. Straube, and J. Bormann. Semi-formal verification of the quasi-static behavior of mixed-signal circuits by SAT-based property cchecking. In *ISO/SA*, 2004.

[17] Stanford validity checker (SVC), version 1.1. http://verify.stanford.edu/SVC/, 2003. University of Stanford.

[18] S. Yovine. Model checking timed automata. In G. Rozenberg and F. Vaandrager, editors, *Lectures on Embedded Systems*, volume 1494 of *LNCS*, pages 114–152. Springer, 1998.