Combining Symbolic Simulation and Interval Arithmetic for the Verification of AMS Designs

Mohamed H. Zaki*, Ghiath Al-Sammane*, Sofiène Tahar*, and Guy Bois ‡
*Electrical & Computer Engineering
Concordia University, Montreal, Quebec, Canada
Email: {mzaki, sammane, tahar}@ece.concordia.ca
‡Genie Informatique
Ecole Polytechnique de Montreal, Montreal, Quebec, Canada
Email: guy.bois@polymtl.ca

***Abstract*—Analog and mixed signal (AMS) designs are important integrated circuits that are usually needed at the interface between the electronic system and the real world. Recently, several formal techniques have been introduced for AMS verification. In this paper, we propose a difference equations based bounded model checking approach for AMS systems. We define model checking using a combined system of difference equations for both the analog and digital parts, where the state space exploration algorithm is handled with Taylor approximations over interval domains. We illustrate our approach on the verification of several AMS designs including $\Delta\Sigma$ modulator and oscillator circuits.**

I. Introduction

Analog and mixed signal (AMS) designs are important integrated circuits used at the interface between an electronic system and its real world. Several computer aided design tools for AMS systems have been developed to overcome challenges in the design process of such designs. Simulation based verification approaches are usually applied to check that an AMS design is robust with respect to different types of inaccuracies. However, with circuits growing in complexity, simulation is not enough to validate complex properties. Actually, it is reported that in recent chips about 50% of errors that implied redesign are due to errors in analog or mixed portions [19]. Therefore, introducing new verification methodologies for these systems is growing in importance.

Boosted by previous successes in the verification of corner cases in digital designs, formal methods became a serious candidate for the verification of AMS systems. In fact, they promise a complete verification and a high level of confidence. Usually, one is interested in global properties connected to the dynamic behavior of the AMS systems. For example, we might be interested in reachability properties, like "can we reach from the initial state a state where a certain condition holds?" or "will the circuit oscillate for giving parameters?". Unfortunately, a direct application of formal methods on AMS systems is very difficult. Unlike digital designs, the functionality of AMS systems is defined in terms of continuous quantities and in terms of continuous time, as they deal usually with factors like voltage level, signal noise and current leakage, in addition to higher order physical effects when designing in deep submicron. In fact, while the behavior of AMS systems is generally modeled using differential equations over continuous quantities, formal methods, however, are defined using models based on discrete events and automata. Today, an important gap remains in linking these two mathematical approaches.

Most research efforts concentrate on how to abstract differential equations in order to be adopted inside automata based algorithms. In this paper, we propose an alternative approach, based on bounded model checking [5] for AMS systems modeled in terms of recurrence equations. Discrete and continuous time based analog systems are described using ordinary differential equations or difference equations, respectively, while the digital parts of the AMS design are described using event based models. We then define a model checking method using a combined system of difference equations for both the analog and digital parts, where state space exploration algorithms are handled with Taylor approximations over interval domains. Such modeling allows the computation over continuous quantities while avoiding the unsoundness inherent in the numerical Taylor approximation. We illustrate the proposed method on the verification of a variety of designs including a $\Delta\Sigma$ modulator design and oscillator circuits.

The rest of the paper is organized as follows: In Section II, we give an overview of the proposed methodology, followed by system model description in Section III. Interval based analysis and Taylor models are described in Section IV. The verification algorithm along with symbolic simulation are then presented in Section V. Experimental results are shown in Section VI, and finally, in Section VII, we present related work before we conclude the paper with Section VIII.

II. Proposed Methodology

The principle of bounded model checking (BMC) is the search for a counter-example of the property checked against the model for a bounded k steps. If such counter-example is found or a fixpoint is reached, the verification task is achieved, else the number of steps can be increased for further verification.

An AMS system is a hybrid system composed in general of a digital part described using logical primitives and an analog part which can be described directly using recurrence

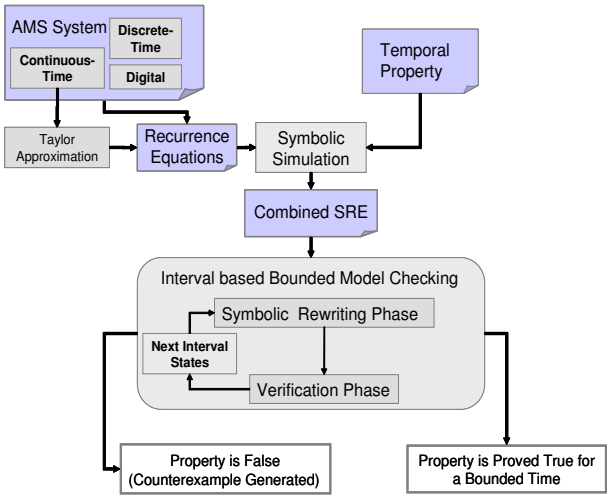

Fig. 1. Overview of the AMS Verification Methodology

equations or a set of differential equations. We propose to convert differential equations into an equivalent set of recurrence equations using the Taylor approximation method. Therefore the recurrence model gives the possibility to handle continuous behaviors like that of current and voltages, but in discrete time intervals, which cover a non-trivial class of mixed behaviors. The properties are temporal relations between signals of the system and are described using a basic subset of Linear Temporal Logic (LTL).

The proposed methodology is composed of two steps as shown in Figure 1. In the first step, the AMS description and LTL property of interest are input to a symbolic simulator that performs a set of transformations by rewriting rules in order to obtain a mathematical representation called System of Generalized Recurrence Equations (SRE) (to be described later). These are recurrence relations that give a description of the property of interest in terms of the system equations. The next step is to prove the properties using a verification engine that performs bounded model checking over interval Taylor model forms. The interval Taylor model form is a combined symbolic numerical representation of the system equations using polynomials and interval terms that ensure enclosure of the reachable states, hence providing a sound abstraction of the reachable sets.

We have implemented this verification algorithm using the computer algebra system *Mathematica*, which provides special functions for symbolic simplification, manipulation and proof of algebraic relations.

III. AMS Design Modeling

Different formalisms have been proposed for modeling systems with combination of discrete and continuous (hybrid) behavior, for instance, hybrid automata [14]. Such formalisms have been applied for AMS modeling. In this paper, we propose to use a generalization of recurrence equations to model different aspects of the AMS designs; mainly the continuous and discrete time behaviors.

A. Systems of Recurrence Equations

The notion of recurrence equation was extended in [1] to describe digital circuits using what is called generalized `If-formula`. Such formalization was found practical in modeling hybrid systems like discrete-time AMS design [2]. In the remaining of this paper, we will show how such recurrence equations can be suitable under certain conditions for modeling continuous-time AMS systems, hence allowing a unified modeling framework for discrete and continuous time AMS designs.

Definition 1: ***Generalized `If-formula`***
In the context of symbolic expressions, the generalized `If-formula` is a class of expressions that extend recurrence equations to describe digital systems. Let $\mathbb{K}$ be a numerical domain ($\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$ or $\mathbb{B}$), a generalized `If-formula` is one of the following:

- A variable $x_i(n) \in \mathbf{x}(n)$, with $i \in \{1, \ldots, d\}$, $n \in \mathbb{N}$ and $\mathbf{x}(n) = \{x_1(n), \ldots, x_d(n)\}$.
- A constant $C \in \mathbb{K}$
- Any arithmetic operation $\diamond \in \{+, -, \div, \times\}$ between variables $x_i(n) \in \mathbb{K}$
- A logical formula: any expression constructed using a set of variables $x_i(n) \in \mathbb{B}$ and logical operators: $not, and, or, xor, nor, \ldots$, etc.
- A comparison formula: any expression constructed using a set of $x_i(n) \in \mathbb{K}$ and comparison operator $\alpha \in \{=, \neq, <, \leq, >, \geq\}$.
- An expression $IF(X, Y, Z)$, where X is a logical formula or a comparison formula and Y, Z are any generalized `If-formula`. Here, $IF(x, y, z) : \mathbb{B} \times \mathbb{K} \times \mathbb{K} \longrightarrow \mathbb{K}$ satisfies the axioms:
  (1) $IF(True, X, Y) = X$
  (2) $IF(False, X, Y) = Y$

Definition 2: ***A System of Recurrence Equations (SRE)***
Consider a set of variables $x_i(n) \in \mathbb{K}$, $i \in \{1, \ldots, d\}$, $n \in \mathbb{N}$, an SRE is a system consisting of a set of equations of the form:

$$x_i(n) = f_i(x_j(n - \gamma)), (j, \gamma) \in \varepsilon_i, \forall n \in \mathbb{Z}$$

where $f_i(x_j(n - \gamma))$ is a generalized `If-formula`. The set ε_i is a finite non-empty subset of $1, \ldots, d \times \mathbb{N}$. The integer γ is called the delay.

Example 1: Figure 2 shows a first-order $\Delta\Sigma$ modulator of one-bit with two quantization levels, $+1$V and -1V. Consider the constraint that the quantizer (input signal $y(n)$) should be between -2V and $+2$V in order to not be overloaded. The SRE of the $\Delta\Sigma$ is then described as:

$$y(n) = y(n-1) + u(n) - v(n-1)$$
$$v(n-1) = IF(y(n-1) > 0, 1, -1)$$

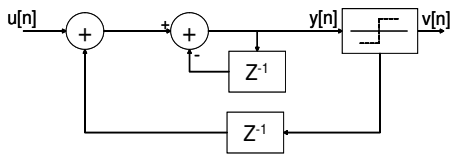

Fig. 2. First-order $\Delta\Sigma$ Modulator

B. Taylor Approximation

A large class of AMS and analog designs have continuous time behavior, usually described using a system of ordinary differential equations (ODE). Unfortunately, a closed form solution is generally not available for ODE systems and discrete approximate models are used. One basic idea is to use the approximation $\mathbf{x}[t_{k+1}] = f(\mathbf{x}[t_k]) + \mathcal{R}m$ of the ODE $\dot{\mathbf{x}} = f(\mathbf{x})$ as truncated Taylor series for $\mathbf{x}(t)$, expanded about time instant t_k, with a remainder term $\mathcal{R}m$.

Theorem 1: ***Taylor Approximation.*** Suppose a function $f : \mathbb{R}^d \to \mathbb{R}$ over states vector $\mathbf{x} \in \mathbb{R}^d$ is $m+1$ time partially differentiable on the interval $[a, b]$. Assume $\mathbf{x}_0 \in [a, b]$, such that $a, b \in \mathbb{R}^d$, then for each $\mathbf{x} \in [a, b]$, $\exists\lambda \in \mathbb{R}$, $0 \leq \lambda \leq 1$, such that:

$$f(\mathbf{x}) = \sum_{k=0}^{m} \frac{[(\mathbf{x}-\mathbf{x}_0).\nabla]^k f(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_0}}{k!} + \frac{[(\mathbf{x}-\mathbf{x}_0).\nabla]^{m+1} f(\mathbf{x})|_{\mathbf{x}=\Lambda}}{(m+1)!}$$

where $\nabla = \mathbf{i}_1 \frac{\partial}{\partial x}{}_i + \ldots + \mathbf{i}_d \frac{\partial}{\partial x}{}_d$ and $\Lambda = \mathbf{x}_0 + \lambda(\mathbf{x}-\mathbf{x}_0)$

In general, to obtain an approximate solution of the ODE system, we consider a sequence of discrete time points $t_0, t_1, \ldots, t_m$ for which the solution is approximated, with $h_i = t_{i+1} - t_i$. If the solution $\mathbf{x}(t)$ of an ODE system $\dot{\mathbf{x}} = f(\mathbf{x})$ is a function which is $p+1$ times continuously differentiable on the open interval $(t_i..t_{i+1})$, then, from the Taylor approximation theorem, we have:

$$\mathbf{x}(t_{i+1}) = \mathbf{x}(t_i) + \sum_{k=1}^{p} (\frac{h^k}{k!}\mathbf{x}^{(k)}(t_i)) + (\frac{h^{p+1}}{(p+1)!}\mathbf{x}^{(p+1)}(\xi))$$

with $h = t_{i+1} - t_i$ and $\xi = [t_i, t_{i+1}]$ and $\forall k \in [1, p+1].\mathbf{x}^{(k)} = f^{(k-1)}(\mathbf{x}(t), t)$, where the vector function f is composed by d elementary functions $f_q(x_1, \ldots, x_d)$, $q \in \{1, \ldots, d\}$, such that:

$$f_q^{(k)}(x_1, \ldots, x_d) = \sum_{m=1}^{d} (\frac{\partial f_q^{(k-1)}(x_1, \ldots, x_d)}{\partial x_m} f_m(x_1, \ldots, x_d))$$

Such representation allows giving an approximate polynomial description of the behavior of an ODE system using generalized SRE. To preserve the inherited behavior of the actual solution, the remainder term should not be discarded and instead bounds must be specified. We use interval arithmetic methods to obtain such bounds. Interval arithmetic provides an over-approximation of the original behavior of the system as will be shown later on.

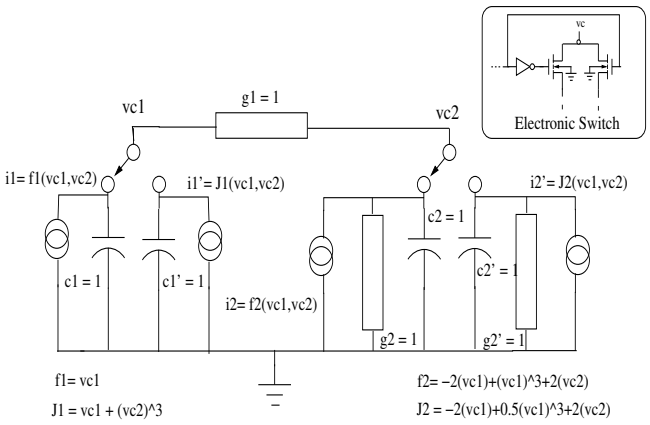

Fig. 3. Switched Analog Circuit

Example 2: Consider the analog circuit in Figure 3, composed of a network of passive components (capacitors and conductances), along with non-linear current sources and two switches. The switches can be designed using CMOS transistors working in saturation mode as shown in the figure. This circuit exhibits an oscillatory behavior when the initial capacitor voltages are within a specified range, based on the switches positions. The voltages across the capacitors can be described using ODEs as follows:

$$\begin{cases} \dot{v_{c1}} = v_{c2} \quad \text{or} \quad \dot{v_{c1}} = v_{c2} + v_{c2}^3 \\ \dot{v_{c2}} = -v_{c1} + v_{c1}^3 \quad \text{or} \quad \dot{v_{c2}} = -v_{c1} + (1/2)v_{c1}^3 \end{cases}$$

Suppose that we specify the switching conditions as

$$Cond_1 = Cond_2 := v_{c1}(n-1) \leq v_{c2}(n-1)$$

For illustration purposes and for clarity, we use Taylor approximation limited to order 2 to obtain the corresponding SREs:

$$v_{c1}(n) := IF(Cond_1, X_1, X_2) \;\; \textit{and} \;\; v_{c2}(n) := IF(Cond_2, Y1, Y2)$$

with:

- $X_1 := \frac{h^2 v_{c1}(n-1)^3}{2} - \frac{h^2 v_{c1}(n-1)}{2} + v_{c1}(n-1) + hv_{c2}(n-1) + \mathcal{R}m_1[\widetilde{v_{c1}}, \widetilde{v_{c2}}]$
- $X_2 := \frac{h^2 v_{c1}(n-1)^3}{4} + \frac{3}{4}h^2 v_{c2}(n-1)^2 v_{c1}(n-1)^3 - \frac{h^2 v_{c1}(n-1)}{2} - \frac{3}{2}h^2 v_{c2}(n-1)^2 v_{c1}(n-1) + v_{c1}(n-1) + hv_{c2}(n-1)^3 + hv_{c2}(n-1) + \mathcal{R}m_2[\widetilde{v_{c1}}, \widetilde{v_{c2}}]$
- $Y_1 := hv_{c1}(n-1)^3 + \frac{3}{2}h^2 v_{c2}(n-1)v_{c1}(n-1)^2 - hv_{c1}(n-1) - \frac{h^2 v_{c2}(n-1)}{2} + v_{c2}(n-1) + \mathcal{R}m_3[\widetilde{v_{c1}}, \widetilde{v_{c2}}]$
- $Y_2 := \frac{hv_{c1}(n-1)^3}{2} + \frac{3}{4}h^2 v_{c2}(n-1)^3 v_{c1}(n-1)^2 + \frac{3}{4}h^2 v_{c2}(n-1)v_{c1}(n-1)^2 - hv_{c1}(n-1) - \frac{h^2 v_{c2}(n-1)^3}{2} - \frac{h^2 v_{c2}(n-1)}{2} + v_{c2}(n-1) + \mathcal{R}m_4[\widetilde{v_{c1}}, \widetilde{v_{c2}}])$

where $\mathcal{R}m_i[\widetilde{v_{c1}}, \widetilde{v_{c2}}]$ are the Taylor approximation remainders, $i = \{1, \ldots, 4\}$ and h is the time step.

IV. Interval Analysis

Interval domains are numerical domains that enclose the original states of a system of equations at each discrete step [20]. Algorithms supporting such numerical domains are used to produce bounded envelopes for the reachable states not only at some discrete time points but also for all continuous ranges of intermediate states between any two consecutive time discrete points. These algorithms, generally known as *validated methods*, are an attractive tool to use in the verification of the behavior of systems with uncertainty on the design parameters or initial conditions. The fact that the generated bounds provide a sound abstraction for the reachable states, makes it attractive to be used with formal verification techniques. The basic interval arithmetics is defined as follows:

Let $I_1 = [a, b]$ and $I_2 = [a', b']$ be two real intervals (bounded and closed), the basic arithmetic operations on intervals are defined by:

$$I_1 \Phi I_2 = \{r_1 \Phi r_2 | r_1 \in I_1 \wedge r_2 \in I_2\}$$

with $\Phi \in \{+, -, \times, /\}$ except that I_1/I_2 is not defined if $0 \in I_2$ [20].

In addition, other elementary functions can be included as basic interval arithmetic operators. For example, *exp* may be defined as $exp([a, b]) = [exp(a), exp(b)]$.

The guarantee that the real solutions for a given function are enclosed by the interval representation is formalized by the following property.

Definition 3: ***Inclusion Function.***[20] Let $f : \mathbb{R}^d \rightarrow \mathbb{R}$ be a continuous function, then $F : \mathbb{I}^d \rightarrow \mathbb{I}$ is an interval extension (inclusion function) of f if

$$\{f(x_1, \ldots, x_d) | x_1 \in X_1, \ldots, x_d \in X_d\} \subseteq F(X_1, \ldots, X_d)$$

where $\mathbb{I}$ is the interval domain and $X_i \in \mathbb{I}$, $i \in \{1, \ldots, d\}$.

Inclusion functions have the property to be inclusion monotonic (i.e., $X_{\mathbb{I}} \subseteq Y_{\mathbb{I}} \rightarrow F(X_{\mathbb{I}}) \subseteq F(Y_{\mathbb{I}})$), hence allowing the checking of fixpoints.

Unfortunately, due to the over-approximation nature of interval analysis, a quick divergence in the reachability calculation may happen. This is mainly due to the following issues [20]:

- *The dependency problem* which is the lack of interval arithmetic to identify different occurrences of the same variable. For example, $x - x = 0$ holds for each $x \in [1, 2]$, but $X - X$ for $X = [1, 2]$ yields $[-1, 1]$.
- *The wrapping effect* which appears when the results of a computation are overestimated when enclosed into intervals, hence leading to error accumulation at each time step.

To overcome the above mentioned drawbacks of interval computation, Taylor model arithmetics were developed recently by Berz *et. al* [3], [18] as an interval extension to Taylor approximations allowing the non-linear approximation of system reachable states using non-convex enclosure sets. Formally, a Taylor model $T_f := p_n(x) + I$ for a given function f consists of a multivariate polynomial $p_r(\mathbf{x})$ of order r in d variables, and a remainder interval I, which encloses Lagrange remainder of the Taylor approximation. Hence, the Taylor model arithmetics use interval computation to obtain reliable enclosures not only for the error term but also for every term of the series, allowing the computation of an over-approximation of the solution function at each time point. In addition, symbolic simplifications are applied at each step, hence reducing the interval calculations and consequently delaying divergence problems, usually, associated with interval based techniques.

Definition 4: ***Taylor Model*** $T_f := (P_{r,f}, I_{r,f})$ is called a Taylor model of order r of a function $f \Leftrightarrow \forall x \in X : f(x) \in P_{r,f}(x - x_0) + I_{r,f}$, where X is an interval, $P_{r,f}(x - x_0)$ is a Taylor approximation polynomial of order r around the point x_0. An interval $I_{r,f}$ is called a remainder bound of order r of f on $X \Leftrightarrow \forall x \in X : R_{r,f}(x - x_0) \in I_{r,f}$.

The basic arithmetic rules on Taylor models are defined as follows [3], [18]:

- *Addition:* $T_{r,f+g} := T_{r,f} + T_{r,g} \triangleq (P_{r,f} + P_{r,g}, I_{r,f} + I_{r,g})$
- *Scalar multiplication:* $T_{r,\alpha f} = \alpha T_{r,f} \triangleq (\alpha P_{r,f}, \alpha I_{r,f})$, $(\alpha \in R)$
- *Multiplication:* $T_{r,fg} \triangleq T_{r,f} T_{r,g} := (P_{r,fg}, I_{r,fg})$ with:
  - $P_{r,f} P_{r,g} = P_{r,fg} + P_e$
  - $P_e \in I_{P_e}$
  - $P_{r,f} \in I_{P_{r,f}}$
  - $P_{r,g} \in I_{P_{r,g}}$
  - $I_{r,fg} := I_{P_e} + I_{P_{r,f}} I_{r,g} + I_{r,f}(I_{P_{r,g}} + I_{r,g})$

Based on the above rules, the Taylor model method extends mathematical operations and functions to Taylor models such that the inclusion relationships are preserved. This is demonstrated by the following theorem:

Theorem 2: [18] Let $f : \mathbb{R}^d \rightarrow \mathbb{R}$ be a continuous function, and $f \in T$, where T is the Taylor model of f, then $T \subseteq F$, where F is the inclusion function of f. Moreover, for two functions $f_1 \in T_1$ and $f_2 \in T_2$, we have $(f_1 + f_2) \in T_S$ and $(f_1.f_2) \in T_P$, where T_S and T_P are Taylor models for the sum and product of T_1 and T_2, respectively.

In practice, the evaluation of a function is transformed to symbolically computing the Taylor polynomial $p_r(x)$ of the function, which will be propagated throughout the evaluation steps, thus hardly affected by the dependency problem or the wrapping effect. Only the interval remainder term and polynomial terms of orders higher than r, which are usually small, are bounded using intervals as described by the rules mentioned above and are processed according to the rules of interval arithmetic. This will be demonstrated by the following example:

Example 3: In non-linear analog circuits, voltages and currents can be described using analytic functions. For example, in a BJT transistor [10], the collector current is described as $i_C = I_S e^{\frac{V_{BE}}{V_T}}(1 + \frac{V_{CE}}{V_A})$, with V_{CE} is the output voltage of a differential stage. In such case, $V_{CE} = tanh(y) + K$, where K is an arbitrary voltage. Consider the Taylor models T_1 and T_2 of the functions e^x, and $tanh(y)$, respectively, where $x = \frac{V_{BE}}{V_T}$, the multiplication $e^x tanh(y)$ can be done using Taylor model arithmetic of two Taylor models of order 3. Let $x, y \in W = [-0.693, 0.693]$ and $T_1(x) := 1 + x + \frac{x^2}{2} + [-0.11, 0.11]$ and $T_2(y) := y - \frac{y^3}{3} + [-0.108, 0.108]$. It holds that:

$$\begin{aligned} T_1(x)T_2(y) \in\ & (1 + x + \tfrac{x^2}{2})(y - \tfrac{y^3}{3}) + (1 + x + \tfrac{x^2}{2}) \\ & [-0.108, 0.108] + (y - \tfrac{y^3}{3})[-0.11, 0.11] + \\ & [-0.11, 0.11][-0.108, 0.108] \\ \subseteq\ & -\tfrac{1}{6}x^2y^3 - \tfrac{xy^3}{3} - \tfrac{y^3}{3} + \tfrac{x^2y}{2} + xy + y + \\ & (1 + W + \tfrac{W^2}{2})[-0.108, 0.108] + \\ & (W - \tfrac{W^3}{3})[-0.11, 0.11] + [-0.22, 0.22] \\ \simeq\ & -\tfrac{y^3}{3} + \tfrac{x^2y}{2} + xy + y + [-0.62, 0.54] \end{aligned}$$

In order to deal with the discrete part of the AMS design, as a generalization of the inclusion function, interval analysis provides efficient and safe methods for checking truth values of Boolean propositions over intervals by using the notion of *inclusion test*.

Definition 5: ***Inclusion Test.*** Given a constraint $c : \mathbb{R}^d \rightarrow \mathbb{B}$, we define $C_{\mathbb{I}} : \mathbb{I}^d \rightarrow \mathbb{B}_{\mathbb{I}}$ to be an inclusion test of c, with an interval domain defined with three values set; $\mathbb{B}_{\mathbb{I}} = \{0, 1, [0, 1]\}$, where 0 stands for false, 1 for true and $[0, 1]$ for indeterminate, iff:

$$\{c(x_1, \ldots, x_d) | x_1 \in X_1, \ldots, x_d \in X_d\} \subseteq C_{\mathbb{I}}(X_1, \ldots, X_d)$$

where $X_i \in \mathbb{I}$, $i \in \{1, \ldots, d\}$.

Inclusion test can be used during the verification algorithm to prove whether the reachable interval states satisfy a given property, or not. We define the inclusion test as follows: $C_{\mathbb{I}}(X) = 1 \Rightarrow \forall x \in X, c(x) = 1$ and $C_{\mathbb{I}}(X) = 0 \Rightarrow \forall x \in X, c(x) = 0$. For instance, given a set of reachable interval states and a property predicate, we remove the states that do not satisfy

$$1 \in C_{\mathbb{I}}(X_1, \ldots, X_n)$$

Therefore, if $C_{\mathbb{I}}(X_1, \ldots, X_n) = \emptyset$, then we have a guarantee that the property is not satisfied.

Let $x_{\mathbb{I}} = [a, b]$ and $y_{\mathbb{I}} = [a', b']$ be two real intervals (bounded and closed), a set of the main logical rules that define the inclusion test is given as follows:

$$\begin{aligned} x_{\mathbb{I}} \leq^{\iota} y_{\mathbb{I}} = 1 \quad & \Leftrightarrow \quad b \leq a' \\ x_{\mathbb{I}} \in^{\iota} y_{\mathbb{I}} = 1 \quad & \Leftrightarrow \quad x_{\mathbb{I}} \in y_{\mathbb{I}} \\ & \Leftrightarrow \quad a \geq a' \ and \ b \leq b' \end{aligned}$$

Example 4: Consider the switching condition in the circuit of Figure 3 defined as $Cond_1 := v_{c1}(n-1) \leq v_{c2}(n-1)$, then we have the following:

$$\begin{aligned} (v_{c1}(n-1) := [1, 3]) \quad & \leq \quad (v_{c2}(n-1) := [3, 5]) \quad = 1 \\ (v_{c1}(n-1) := [1, 3]) \quad & \leq \quad (v_{c2}(n-1) := [2, 5]) \quad = [0, 1] \end{aligned}$$

V. Interval based bounded Model Checking

In this section, we present bounded model checking (BMC) algorithm to support AMS designs. We explore a solution relying on symbolic and interval computational methods. Our BMC approach is based on modeling the transition function as SREs over the Taylor models forms. We proceed on the SREs traces using a time step $\hbar$ which implies that our answer is relative to a limited time interval. For recurrence equations, we have $\hbar = 1$. For differential equations, we approximate them using Taylor model with $\hbar \in \mathbb{R}$, ensuring the accumulated error due to $\hbar$-approximation is confined in the Interval part of the Taylor model. We consider properties specified in a LTL like language.

In the remaining of this section we will describe the symbolic simulation, and the property checking algorithm of the proposed methodology.

A. The Symbolic Simulation Algorithm

The generation of the SREs and the evaluation of Taylor model forms rely on rewriting rules based on the symbolic simulation algorithm developed in [1] for digital systems and extended for discrete-time AMS designs in [2]. The symbolic simulation algorithm $ReplaceRepeated(Expr, R)$ shown in Algorithm 1 is based on rewriting by repetitive substitution, which applies recursively a set of rewriting of rules R on an expression $Expr$ until a fixpoint is reached.

Algorithm 1 $ReplaceRepeated(Expr, R)$

```
1: Expr = expr
2: repeat
3:     Expr_t = ReplaceList(Expr, R)
4:     Expr = Expr_t
5: until FP(Expr_t, R)
```

ReplaceList(Expr, R): The substitution function $ReplaceList$ takes as arguments an expression $Expr$ and a list of substitution rules $R = \{R_1, R_2, \ldots, R_n\}$. It applies each rule sequentially on the expression.

FP(Expr, R): A substitution fixpoint $FP(Expr, R)$ is obtained, if:

$$ReplaceList(expr, R) \equiv ReplaceList(ReplaceList(expr, R), R)$$

The correctness of this rewriting algorithm as well as the proof of termination and confluence of the rewriting system is discussed in [1].

Depending on the type of expressions, we distinguish the following kinds of rewriting rules over Boolean and Real domains:

- *Polynomial symbolic expressions:* R_{Math} for the simplification of polynomial expressions ($\mathbb{R}^n[x]$).
- *Logical symbolic expressions:* R_{Logic} for the simplification of Boolean expressions and to eliminate obvious ones like $(and(a, a) \rightarrow a)$ and $(not(not(a)) \rightarrow a)$.
- *If-formula expressions:* R_{IF} for the simplification of computations over `If-formulae`. The definition and properties of the IF function, like reduction and distribution, are used.
  - IF Reduction: $IF(x, y, y) \rightarrow y$
  - IF Distribution: $f(A_1, \ldots, IF(x, y, z), \ldots, A_n) \rightarrow IF(x, f(A_1, \ldots, y, \ldots, A_n), f(A_1, \ldots, z, \ldots, A_n))$ For example $a + IF(x > 0, b, a) \rightarrow IF(x > 0, b + a, a + a)$

For Taylor model generation and evaluation over intervals, we used the following rules which where developed based on the properties described in Section IV

- *Taylor expressions:* R_{Tlr} are rules intended for the simplification of Taylor model expressions ($T_{r,f}$).
- *Interval expressions:* R_{Int} are rules intended for the simplification of interval expressions.
- *Interval-Logical symbolic expressions:* $R_{Int-Logic}$ are rules intended for the simplification of Boolean expressions over intervals.

B. Temporal Properties

We use an LTL like syntax to represent the properties. The syntax is composed of formulae $P(n)$ defined recursively and built using Boolean expressions over atomic propositions with temporal operators: eventually F and always G. To describe properties on analog signals like current and voltages, atomic propositions encode predicates (inequalities) over reals; $p(n) \sim c$, where $p(n)$ is a polynomial over the state variables, $\sim \in \{<, \leq, >, \geq, =, \neq\}$, $c \in \mathbb{R}$. As in traditional BMC, we define temporal operators regarding a bounded time step k.

Always operator **G**: $\mathbf{G}P(n)$ specifies that a property $P(n)$ holds in the current time step of a given path iff the property and the operand hold at the current state and all previous states. Iteratively, we write:

$$\mathbf{G}P(n+1) = \bigwedge_{k=1}^{n+1} P(k)$$

Eventually operator **F**: $\mathbf{F}P(n)$ specifies that a property holds at the current state or at a previous state. Iteratively, we write:

$$\mathbf{F}P(n+1) = \bigvee_{k=1}^{n+1} P(k)$$

In fact, the inverse of the property ($\neg P$) under verification is used in the BMC algorithm. When a satisfying valuation is returned by the solver, it is interpreted as a counterexample of length k and the property P is proved satisfied ($\neg P$ is satisfied). However, if the problem is determined to be unsatisfiable, the solver produces a proof (of unsatisfiability) of the fact that there are no counterexamples of length k.

C. Verification Algorithm

The bounded forward reachability algorithm starts at the initial states and at each step computes the image, which is the set of reachable interval states. This procedure is continued until either the property is falsified in some state or no new states are encountered. We define the interval based transition system denoting the behavior of the system as follows:

Definition 6: ***Interval based state machine.*** An *Interval based state machine* is a tuple $\mathcal{T}_I = (S_I, S_{I,0}, \rightarrow_{T_f})$, where S_I is the interval state space, $S_{I,0} \subseteq S_I$ is the set of initial interval states, $\rightarrow_{T_f} \subseteq S_I \times S_I$ is a relation defined using Taylor model forms T_f and capturing the abstract transition between interval states such that:

$$\{s \rightarrow_{T_f} s' | \exists a \in s, \exists b \in s' : b = f(a) \;\; and \;\; f \in T_f\}$$

where $a, b \in \mathbb{R}^d$, $s, s' \in S_I$, $f = \{f_1, \ldots, f_d\}$, $T = \{T_{f_1}, \ldots, T_{f_d}\}$ with $f_i : \mathbb{R}^d \rightarrow \mathbb{R}$ is a continuous function, $i \in \{1, \ldots, d\}$ and $f_i \in T_{f_i}$, where T_{f_i} is the Taylor model of f_i.

Bounded model checking over interval domains is then defined as follows:

Definition 7: ***BMC*** Given a natural number $k \geq 0$, an interval based state machine $(S_I, S_{I,0}, \rightarrow_{T_f})$ as defined above, and a property $Prop$, we say that property $Prop$ is verified for k steps if:

$$\forall s \in \mathcal{R}^k(S_0) : s \models Prop$$

where S_0 is the set of initial states.

The different steps for checking safety properties is shown in Algorithm 2. The system equations and the (negated) property $\neg Prop[n]$ to be verified are given, with the equations initialized are provided. The loop in lines (1-13) describes the verification procedure for N_{max} time steps. At each step n, we use an evaluation over the Taylor model forms (line 2) to check whether the property is satisfied or not (line 3). If $\neg Prop[n]$ is satisfied then a counterexample is generated (line 10), if not, then check for fixpoint (line 5), otherwise update the reachable states (line 12) and go to the next time step verification.

SRE($\mathcal{A}, r$): Given an AMS system ($\mathcal{A}$) and an order r, $SRE(\mathcal{A}, r)$ returns the generalized SREs by applying the symbolic rules described earlier. For the case of a continuous function, the Taylor approximation of order r is applied to

Algorithm 2 Safety Verification

Require: $x[n] := SRE(\mathcal{A}, r)$
Require: $\neg Prop[n]$
Require: $\mathcal{R}^0 = S_0$
1: **for** $n = 1$ to N_{max} **do**
2: $T_{o_t,x[n]} := TM\_Form(x[n], o_t)$
3: **if** $Prop\_Check(\neg Prop[n], T_{o_t,x[n]}) == False$ **then**
4: **if** $Reach[T_{o_t,x[n]}] \subseteq \mathcal{R}^{n-1}$ **then**
5: **return** fixpoint reached
6: **else**
7: Inc_Step(n)
8: **end if**
9: **else**
10: Generate_CE
11: **end if**
12: $\mathcal{R}^{n-1} = Update\_Reach(\mathcal{R}^{n-2}, Reach[T_{o_t,x[n-1]}])$
13: **end for**

generate the SREs.

TM_Form(x[n],o_t): Given a set of SREs, TM_Form returns the corresponding Taylor model with order o_t at the specified time step. Such model will be checked against properties for satisfaction using $Prop\_Check$

Prop_Check: Given the Taylor model forms representing the transition function and the property $\neg Prop()$, apply algebraic decision procedures to check for satisfiability. The safety verification at a given step n can be defined with the following formula:

$$Prop\_Check \triangleq \mathbf{x}[n] = T_{o_t,x[n]}(\mathbf{x}[n-1], \hbar) \wedge \neg Prop(\mathbf{x}[n])$$

Reach[$T_{o_t,x[n]}$]: Given the Taylor model form at an arbitrary time step, **Reach** evaluates the reachable states at according to the following definition:

Definition 8: ***1-Step Reachable states.*** The set of reachable states in 1 step from a given set of states $S_k \subseteq \mathbb{I}^d$, is denoted by $\mathcal{R}_1(S_k)$ and is defined as:

$$\mathcal{R}_1(S_k) \triangleq \{s' \in S_{k+1} | \exists s \in S_k : \overrightarrow{F}_1(s) = s'\}$$

where $S_{k+1} \subseteq \mathbb{I}^d$, $\overrightarrow{F} = (F_1, \ldots, F_d)$, with $F_i : \mathbb{I}^d \to \mathbb{I}$ is an interval evaluation of Taylor model form of the function $f_i : \mathbb{R}^d \to \mathbb{R}$, $i \in \{1, \ldots, d\}$.

Update_Reach(R_1, R_2): The function returns the union of the states in the sets R_1 and R_2 according to the following definition:

Definition 9: The set of reachable states in less than k steps $(0 < l < k)$, from a given set of S_0 of states, is denoted by $\mathcal{R}^{<k}(S_0)$, and is defined as:

$$\mathcal{R}^{<k}(S_0) \triangleq \bigcup_{l<k} \mathcal{R}^l(S_{l-1})$$

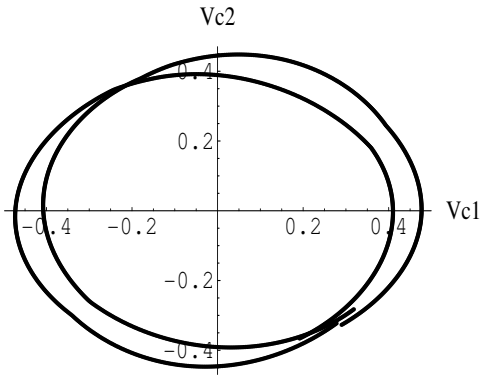

Fig. 4. Oscillation Behavior for Circuit in Example 2

For instance, in Algorithm 2 (line 12) we have $\mathcal{R}^{n-2} = \mathcal{R}^{<n-1}(S_0)$ and $Reach[T_{o_t,x[n-1]}] = R_1(s_{n-1})$

VI. Experimental Results

We have applied the verification algorithm on the analog circuit described in Example 2 and a third-order $\Delta\Sigma$ modulator. For the analog circuit, we checked the oscillation property for given set of initial voltages and for all switching conditions. We formally describe the oscillation property as:

$$Prop_1 : \mathbf{G}((p_1 \Rightarrow \mathbf{F}p_2) \wedge (p_2 \Rightarrow \mathbf{F}p_1))$$

where $p_1 = \neg p_2 := V_{c1} < V_{c2}$. For instance, when the analog circuit is described by $\dot{v_{c1}} = v_{c2}$ and $\dot{v_{c2}} = -v_{c1} + v_{c1}^3$, the reachable states for the oscillation behavior are shown in Figure 4, bounded by the corresponding Taylor model. We also checked several safety properties, e.g.,

$$Prop_2 : \mathbf{G}(-0.5 < V_{c1} < 0.5) \wedge (-0.5 < V_{c2} < 0.5)$$

and

$$Prop_3 : \mathbf{G}(-1 < V_{c2} < 1)$$

Details on initial conditions are shown below:

$$Parameters_1 \rightarrow \begin{cases} a \rightarrow [-0.03, 0.03] & b \rightarrow [-0.03, 0.03] \\ h \rightarrow 0.01 & \\ x[0] = 0.3 + a & y[0] = -0.3 + b \end{cases}$$

$$Parameters_2 \rightarrow \begin{cases} a \rightarrow [-0.03, 0.03] & b \rightarrow [-0.03, 0.03] \\ h \rightarrow 0.01 & \\ x[0] = 1 + a & y[0] = 0.2 + b \end{cases}$$

The verification results for 2 possible switching cases of this circuit (we refer to as circuit 1 and circuit 2) are shown in Table I. For the first set of initial conditions, we find that the circuit is behaving in accordance with the properties, hence the properties are satisfied. For the second set of initial conditions, the safety properties $Prop_2$ and $Prop_3$ are violated while divergence prevent us to check whether the

circuits are oscillating or not [1].

Table II shows the verification results for a third-order $\Delta\Sigma$ modulator [2]. A $\Delta\Sigma$ modulator is said to be stable if the integrator output remains bounded under a bounded input signal, thus avoiding that the quantizer in the modulator becomes overloaded which leads to instability. The stability properties is written as: $P(k) := \mathbf{G}(-1 < x3(k) < 1)$, where $x3$ is the input to the quantizer. For a first set of initial constraints, the modulator loses stability after 5 time steps. For the second, set of constraints the modulator was proved stable for 45 time steps.

TABLE II
VERIFICATION RESULTS FOR 3^{rd} ORDER $\Delta\Sigma$ MODULATOR

Initial Constraints	**Property Evaluation for $n = 0$ to N_{max} Cycles**	**CPU time Used**
Initial constraints 1	$N_{max} = 15$ $n = 0$ to 5 True $n > 5$ False	3.89 sec
Initial constraints 2	$N_{max} = 45$ True	120.8 sec

VII. RELATED WORK

We can identify two classes of verification techniques for AMS designs, namely state exploration methods (e.g., reachability analysis) and algebraic methods (e.g., constraint solving). Common to the proposed state based methods is the necessity of the explicit computation of either exact or approximate reachable sets corresponding to the continuous dynamics, hence deducing properties about the properties of the design under verification. In the constraint based methods, the AMS design is described by a set of equations (algebraic or difference equations) along with a set of constraints. Algebraic and logical rules are then applied to check whether the system satisfies such constraints or not.

For instance, model checking and reachability analysis were proposed for validating AMS designs over a range of parameter values and a set of possible input signals. Several methods for approximating reachable sets for continuous dynamics have been proposed in the open literature. They rely on the discretization of the continuous state space by using over-approximating domains like polyhedra and intervals. In [15], the authors construct a finite-state discrete abstraction of electronic circuits by partitioning the continuous state space into fixed size hypercubes and computed the reachability relations between these cubes using numerical techniques. In [11], the authors tried to overcome the expensive computational method in [15], by combining discretization and projection techniques of the state space, hence reducing its dimension. While the approach in [11] is less precise due to the use of projection techniques, it is still sound. Variant approaches of the latter analysis were proposed. For instance, the model checking tools d/dt [6], *Checkmate* [9] and *PHaver* [8] were adapted and used in the verification of a biquad low-pass filter [6], a tunnel diode oscillator and a $\Delta\Sigma$ modulator [9], and voltage controlled oscillators [8]. In [13], the authors used intervals to construct the abstract state space, while used heuristics to identify possible transition between adjacent regions, main difference with [15], is that they allow variable sized regions. Petri nets based models and algorithms have been developed also for the reachability analysis of AMS designs in [17], [16].

The AMS verification we present in this paper is in the same spirit as the above mentioned works in terms of requirement for state exploration. However, we can identify two distinct points. First, we rely on a recurrence equation form as a way to model the design rather than automata, which provide us with more compact representation. Second, we apply the verification over Taylor model forms which provide tight bounds for the reachable states by using non-convex over approximation. In addition, Taylor models allow the symbolic representation of the reachable states using polynomials terms, therefore minimizing the risk of state explosion.

In [12], the author proposed an approach for specifying and reasoning about digital systems that are described at the analog level of abstraction. The approach relies upon specifying the behaviors of analog components by piecewise-linear predicates on voltages and currents. Theorem proving and constraint based methods are then used to check for the implication relation between the implementation and the specification. In [7], the authors developed a bounded model checking prototype tool (*Property-Checker*) for the verification of the static behavior of AMS designs. The basic idea is based on validity checking of first-order formulae over a finite interval of time. In [7], the authors trade-off accuracy with efficiency by basing the analysis on rational numbers rather than real numbers, however affecting the soundness of the verification. In addition to the loose approximations in [12], [7], the verification is only possible for static behavior.

In [2], [23], the authors propose an induction verification approach for AMS designs using symbolic methods. The procedure is iterative in the sense that if the proof is obtained, then the property is verified. Otherwise, generated counterexamples are analyzed and constraints refinement is applied and verification is repeated until the property is verified or a concrete counterexample is identified. Such methodology is limited for AMS systems that can be described using discrete time models, while our approach consider continuous time systems. More details about the application of formal methods to the verification of AMS designs can be found in [24].

VIII. CONCLUSION

In this paper, we have defined a bounded model checking approach for AMS systems modeled in terms of combination between SRE and differential equations. We have proposed a semi-symbolic modeling of the state space using the principle of Taylor models which provide a way for representing a combination of representation using a combination of polynomials and interval terms. The main advantage of such modeling is

[1]The experiments were performed on Intel Core2 1900 MHz processor and 2GB of RAM

[2]Details about the design can be found in [22]

TABLE I
VERIFICATION RESULTS

Circuit & Properties	**BMC Verification for $k = 0$ to N_{max} Steps**	**CPU & Memory Used**
Circuit 1 (Parameters 1)	$N_{max} = 700$	107.39 sec
Oscillation Property	$Proved\ True$	7.93 MB
$Prop_2$	$Proved\ True$	
$Prop_3$	$Proved\ True$	
Circuit 1 (Parameters 2)	$N_{max} = 700$	108.41 sec
Oscillation Property	Not Verified (Divergence)	7.14 MB
$Prop_2$	$Proved\ False$ at $k = 18$	
$Prop_3$	$Proved\ False$ at $k = 18$	
Circuit 2 (Parameters 1)	$N_{max} = 1200$	583.75 sec
Oscillation Property	$Proved\ True$	51.15 MB
$Prop_2$	$Proved\ True$	
$Prop_3$	$Proved\ True$	
Circuit 2 (Parameters 2)	$N_{max} = 1200$	584.05 sec
Oscillation Property	Not Verified (Divergence)	50.60 MB
$Prop_2$	$Proved\ False$ at $k = 4$	
$Prop_3$	$Proved\ False$ at $k = 9$	

the fact, that the polynomial representation helps slowing the divergence due to the over-approximated intervals, meanwhile, the interval part provides an important abstraction to handle the continuous behavior.

We have developed and implemented this arithmetic as a set of simplification rules a the bounded model checking algorithm. Experimental results have proven the feasibility and the utility of the approach. However, the method is still limited in terms of capacity. In fact, we have implemented our methodology using standard libraries for symbolic computation available in $Mathematica$.

Future research directions include investigating alternative implementations to improve the experimental capacity over more complex systems and to measure the limitation of the proposed methodology. Also, an important effort is needed to classify the kind of properties and AMS systems that can be verified using this verification approach.

REFERENCES

[1] G. Al-Sammane. Simulation Symbolique des Circuits Decrits au Niveau Algorithmique. PhD thesis, Université Joseph Fourier, Grenoble, France, July 2005.

[2] G. Al Sammane, M. Zaki, and S. Tahar. A Symbolic Methodology for the Verification of Analog and Mixed Signal Designs. IEEE/ACM Design Automation and Test in Europe, pp. 249-254, 2007.

[3] M. Berz, G. Hoffsttter. Computation and Application of Taylor Polynomials with Interval Remainder Bounds, Reliable Computing, 4(1): 83-97, Springer, 1998.

[4] E.M. Clarke, O. Grumberg, and D.A. Peled. Model Checking. MIT Press, 2000.

[5] E. M. Clarke, D. Kroening, J. Ouaknine, and O. Strichman. Computational challenges in bounded model checking. International Journal on Software Tools for Technology Transfer, 7(2): 174-183, Springer, 2005.

[6] T. Dang, A. Donze, O. Maler. Verification of Analog and Mixed-signal Circuits using Hybrid System Techniques. In Formal Methods in Computer-Aided Design, LNCS 3312, pp. 14-17, Springer, 2004.

[7] M. Freibothe, J. Schoenherr, and B. Straube. Formal Verification of the Quasi-Static Behavior of Mixed-Signal Circuits by Property Checking, Electr. Notes Theor. Comput. Sci., 153(3):23-35, Elsevier, 2006.

[8] G. Frehse, B.H. Krogh, R.A. Rutenbar. Verifying Analog Oscillator Circuits Using Forward/Backward Abstraction Refinement. IEEE/ACM Design Automation and Test in Europe, pp. 257-262, 2006.

[9] S. Gupta, B.H. Krogh, R.A. Rutenbar. Towards Formal Verification of Analog Designs, IEEE/ACM International Conference on Computer Aided Design, pp. 210-217, 2004.

[10] P.R. Gray, P.J. Hurst, S.H. Lewis, and R.G. Meyer. Analysis and Design of Analog Integrated Circuits, Wiley, 2001

[11] M. R. Greenstreet, I. Mitchell. Reachability Analysis Using Polygonal Projections. In Hybrid Systems: Computation and Control, LNCS 1569, pp. 103–116, Springer, 1999.

[12] K. Hanna. Automatic Verification of Mixed-Level Logic Circuits. In Formal Methods in Computer-Aided Design, LNCS 1522, pp.133-166, Springer, 1998.

[13] W. Hartong, R. Klausen, and L. Hedrich. Formal Verification for Nonlinear Analog Systems: Approaches to Model and Equivalence Checking, Advanced Formal Verification, pp. 205-245, Kluwer, 2004.

[14] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: A Model Checker for Hybrid Systems. Software Tools for Technology Transfer, 1(1–2):110-122, Kluwer, 1997.

[15] R.P. Kurshan and K.L. McMillan. Analysis of Digital Circuits Through Symbolic Reduction. IEEE Trans. on Computer-Aided Design, 10(11): 1356-71, 1991.

[16] S. Little, N. Seegmiller, D. Walter, C. Myers, and T. Yoneda. Verification of Analog/mixed-signal Circuits using Labeled Hybrid Petri Nets, IEEE/ACM International Conference on Computer Aided Design , pp. 275-282, 2006.

[17] S. Little, D. Walter, N. Seegmiller, C. Myers, and T. Yoneda. Verification of Analog and Mixed-Signal Circuits Using Timed Hybrid Petri Nets. In Automated Technology for Verification and Analysis, LNCS 3299, pp. 426-440, Springer, 2004.

[18] K. Makino, M. Berz. Remainder Differential Algebras and their Applications. In Computational Differentiation: Techniques, Applications, and Tools, pp. 63-75, SIAM, 1996.

[19] C. J. Myers, R. R. Harrison, D. Walter, N. Seegmiller, S. Little. The Case for Analog Circuit Verification. Electr. Notes Theor. Comput. Sci., Elsevier, 153(3):53-63, 2006.

[20] R.E. Moore. Methods and Applications of Interval Analysis, Society for Industrial and Applied Mathematics, 1979.

[21] S. Wolfram. Mathematica: A System for Doing Mathematics by Computer. Addison Wesley Longman Publishing, USA, 1991.

[22] M. Zaki, G. Al Sammane, S. Tahar, and G. Bois. A Bounded Model Checking Approach for AMS Designs. Technical Report, ECE Dept., Concordia University, Montreal, Quebec, Canada, May 2007. http://hvg.ece.concordia.ca/Publications/TECH_REP/ AMS_BMC_TR07

[23] M. Zaki, G. Al Sammane and S. Tahar. Formal Verification of Analog and Mixed Signal Designs in Mathematica, Proc. International Conference on Computational Science, LNCS 4488, pp. 263-267, Springer, 2007.

[24] M. Zaki, S. Tahar, and G. Bois. Formal Verification of Analog and Mixed Signal Designs: Survey and Comparison, IEEE Northeast Workshop on Circuits and Systems, pp. 281-284, 2006.