JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 6, May 2009

Integrating Abstraction Techniques for Formal Verification of Analog Designs

Mohamed H. Zaki* , William Denman† , and Sofiène Tahar‡ ,
Concordia University, Montreal, Quebec, Canada

and

Guy Bois§
Ecole Polytechnique de Montreal, Montreal, Quebec, Canada

DOI: 10.2514/1.44289

The verification of analog designs is a challenging and exhaustive task that requires deep understanding of physical behaviors. In this paper, we propose a qualitative-based predicate abstraction method for the verification of a class of nonlinear analog circuits. In the proposed method, system equations are automatically extracted from a circuit diagram by means of a bond graph. Verification is applied based on combining techniques from constraint solving and computer algebra along with symbolic model checking. Our methodology has the advantage of avoiding exhaustive simulation normally encountered in the verification of analog designs. To this end, we have used Dymola, Hsolver, SMV, and Mathematica to implement the verification flow. We illustrate the methodology on several analog examples including Colpitts and tunnel diode oscillators.

I. Introduction

THE successful application of formal methods to the verification of digital and software systems has motivated research toward extending the verification techniques beyond the discrete domain. Consequently, this has encouraged the development of techniques to verify real-time and hybrid behaviors. Such behavior can be observed in both the aerospace and aeronautical domains, where formal verification has been used to ensure safety and correctness properties. For instance, Xia et al. [1] combined constraint solving and abstraction to check for collision avoidance. A mechanical landing gear system was formally verified using theorem proving in [2], whereas the correctness of the embedded software in avionics applications was checked using abstract interpretation [3] as part of the Astree project. The focus of our paper is the verification of the analog behavior of embedded systems that can be used in designs with safety critical environments.

In general, embedded systems are characterized by their reactive and real-time dynamical behavior with respect to their environment. This interaction is often facilitated through sensors to capture the state of the environment and actuators to change or update the environment (see Fig. 1a). Analog designs are a cornerstone in embedded systems, which are required at the interface between the circuitry and the environment. The important functionalities

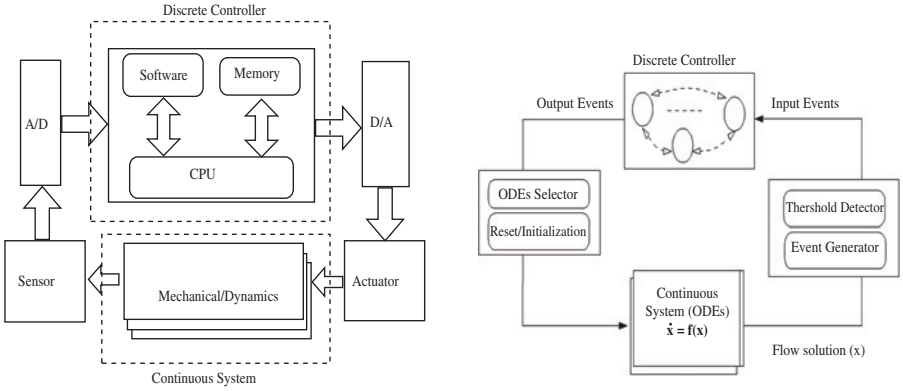
Received 10 March 2009; accepted for publication 15 March 2009.

* Department of Electrical and Computer Engineering, Concordia University, Montreal, Quebec, Canada. mzaki@ece.concordia.ca.

† Department of Electrical and Computer Engineering, Concordia University, Montreal, Quebec, Canada. w_denm@ece.concordia.ca.

‡ Department of Electrical and Computer Engineering, Concordia University, Montreal, Quebec, Canada. tahar@ece.concordia.ca.

§ Genie Informatique, Ecole Polytechnique de Montreal, Montreal, Quebec, Canada. guy.bois@polymtl.ca.



a) Architecture model b) Behavioral modeling

Fig. 1 Embedded systems.

of such designs are the processing of analog signals. Other functionalities include filtering, frequency synthesis, and generating timing references [4].

Hybrid systems theory was developed to deal with heterogeneous behavior. Specifically, to fully understand the system's behavior and meet high-performance specifications, the designer must model all dynamic interactions. These interactions can become very important when there are tight integrations or strong interactions among different parts of the system. For instance, at the specification level, the embedded system architecture illustrated in Fig. 1a can be modeled in an abstract way as shown in Fig. 1b. The digital controller is modeled by finite state machines (FSMs), whereas the dynamic environment is described using systems of ordinary differential equations (ODEs). In addition, the sensor and A/D interface can be modeled as a threshold detector and event generator, respectively, whereas the actuator and D/A components can be modeled as switches that choose between different system ODEs and set the initialization and reset condition necessary for correct functionality.

In this respect, the dynamic behavior of analog systems is generally modeled using systems of differential algebraic equations (DAEs), but generating the equations from a circuit diagram is not trivial. Specifically, the DAEs must accurately describe the behavior of the circuit while remaining simple enough to be verified using automated tools.

In addition, the verification of analog designs is a challenging task because of the complexity of modeling and verifying continuous-time behavior, when compared to digital designs. For instance, the digital design verification is based on the validation of abstract models that reside in a finite state space. In contrast, the functionality of analog designs depends on continuous electrical quantities, device parameters, in addition to parasitics and current leakage. All those factors can drastically change the behavior of an analog design, making conventional finite-state verification techniques inadequate. Consider the situation that a voltage at a specific node should not exceed a certain value. Such a property is important, as a voltage exceeding a certain specified value can lead to failure of functionality and ultimately to a breakdown of the design, which can result in undesirable consequences.

Traditionally, simulation is used for the evaluation of a system's functionality. However, simulation is often done manually in an informal fashion and the search of the state space is not complete. As a consequence, simulation methods lack the rigor needed to ensure correctness of the design. In addition, simulation falls short of validating interesting properties of the design behavior such as temporal requirements. Another problem is caused by the fact that although a design is defined in advance, one cannot ensure a priori that the desired properties will exactly be met during manufacturing of the actual circuit.

In summary, the analog design process must ensure, with a high degree of confidence, the proper functionality in all possible situations and be able to meet the performance requirements. This motivates the necessity of using formal verification methodologies throughout the design process.

This paper demonstrates a novel verification flow to verify functional properties of analog designs. The basic idea is to extract the design equations automatically from the corresponding circuit diagram, by means of bond graph transformations [5]. An approach based on combining predicate abstraction and constraint solving is then applied to verify the properties of interest.

Bond graphs are a domain independent framework for modeling physical systems, which is based on the flow of power between abstract objects. This allows for the universal treatment of different physical domains. The benefit of using bond graphs as a modeling framework is the representation of designs using the concepts of energy flow, effort and conservation. Additionally, the causality of bond graphs can be automatically generated [6], which leads to the automatic extraction of DAEs. Moreover, since bond graphs are object oriented, larger models can be built from simpler blocks reducing the need for a complex equation layer [5].

Abstraction methods for verification started with the seminal paper on abstract interpretation [7]. Since then, different abstraction approaches were developed to tackle different issues in the verification. One of the most successful approaches is predicate abstraction [8]. In this approach, the state space is divided into a finite set of regions and a set of rules is used to define the transition between these regions in a way that the generated state transition system can be verified using model checking. Among the proposed enhancements of predicate abstraction is the lazy abstraction approach [9]. The basic idea here is that, instead of generating the entire abstract model, a region is abstracted only when it is needed in the verification step.

The different steps of the proposed methodology are shown in Fig. 2. The methodology consists of two parts: namely modeling and verification. In the modeling section, the circuit model is analyzed and simplified to obtain the system of equations necessary for the verification.

At first, we require that the circuit in question be described in Dymola [10] using an electrical circuit diagram, which can be translated automatically to the corresponding bond graph. The circuit components are then represented by generic objects that have the same physical quantities as in the circuit diagram, but are connected by bonds that explicitly show the flow of power. The bond graphs are inherently acausal, but by assigning causality to the components, the system's state equations can be automatically generated using Dymola and the BondLib library [5]. Dymola is a modeling framework and BondLib contains the bond graph models and components. The advantage of using Bondlib is that it preserves the behavior of the corresponding Spice models of electrical components.

Given the design equations and the property of interest, the verification consists of two complementary stages. First invariant checking is applied to verify properties on the extracted system of equations. Because of incompleteness, a negative result does not disprove the property; in this case a refinement of the abstract states by predicate abstraction techniques is used to verify the properties. The property verification provides the advantage of avoiding explicit

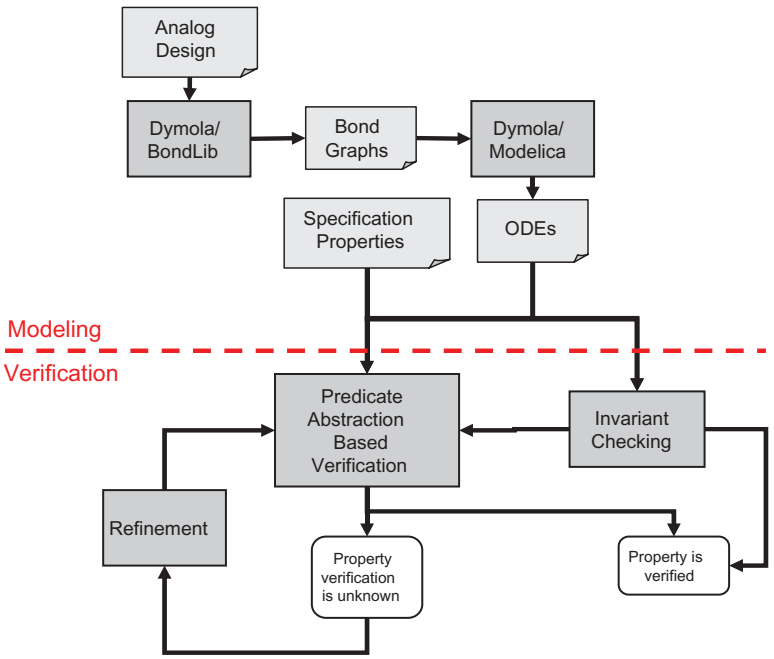


Fig. 2 Proposed verification flow.

computation of reachable sets. If the property cannot be verified at this stage, refinement is needed only for the nonverified regions by adding more predicates. Verification is then applied on the newly generated abstract model[¶] .

The proposed methodology has the advantage of avoiding exhaustive simulation usually encountered during the verification. To this end, we have combined several tools to implement the verification flow. Basically, Dymola [10] modeling engine is used to extract the design equations from the circuits' schematics, whereas Hsolver [11] and Mathematica [12] along with the model checker SMV [13] are used in the verification phase to construct the abstract model from the design and to verify it against the specification properties. We illustrate the methodology on several analog examples including Colpitts and tunnel diode oscillator circuits.

The rest of the paper is organized as follows: we start with an overview of the relevant work in Sect. II. In Sect. III, we provide the theory behind bond graphs and analog modeling followed by the verification techniques in Sect. IV. Experimental results are shown in Sect. VII, before concluding the paper with Sect. VIII.

II. Related Work

The proposed verification methodology spans through many different research domains. Therefore we will only highlight the most crucial information including the work on bond graphs for the analysis of analog designs.

A. Bond Graphs for System Design and Verification

Bond graphs have been successfully extended to aid in the verification of aeronautical systems. In [14], bond graphs are used to model mechanics for different aeronautical systems. The accuracy of the model was proven via simulation. The first work for the formal verification of bond graph models was proposed in [15], where bond graphs were used to represent the complex mechanics of a landing gear system. Verification using extended duration calculus was applied on the extracted equations. However, the proposed approach was limited to simple linear continuous, whereas our methodology is developed to deal with more realistic nonlinear behaviors.

Researchers also explored the modeling of analog designs using bond graphs. In [5], bond graphs are used to model an analog inverter, demonstrating that bond graphs constructed at different levels of abstraction can represent simpler models. In [6], bond graphs are suggested as an addition to the framework of SystemC-AMS to aid in the modeling and simulation of analog circuits. Simulation was the standard tool for the analog design. On the contrary, we benefit from advances in formal verification of analog designs to propose a novel verification framework for analog designs modeled using bond graphs.

B. Analog Design Verification

The verification of analog circuits started with the work in [16], where the authors constructed 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 [17], the authors tried to overcome the expensive computational method in [16], by combining discretization and projection techniques of the state space, hence reducing its dimension. Although the approach in [17] 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 [18], *Checkmate* [19], and *PHaver* [20] were adapted and used in the verification of a biquad low-pass filter [18], a tunnel diode oscillator and a $\Delta\Sigma$ modulator [19], and voltage controlled oscillators [20]. In [21], the authors used intervals to construct the abstract state space, while using heuristics to identify possible transition between adjacent regions. The main difference with [16] is that they allow variable-sized regions. In [22], the authors proposed a nonlinear approximation for reachable states of analog designs, where the state space exploration algorithms are handled with Taylor approximations over interval domains. All of the above surveyed formal methods limit the verification of the circuit to a predefined time bound because they depend on explicit state exploration. In contrast, we propose in this paper qualitative-based methods for the construction and verification of abstract models, which overcome the time bound requirement. A detailed literature overview of analog formal verification can be found in [23].

[¶] We use a simple refinement procedure based on interval methods for ODEs that identifies and eliminates the spurious counterexamples, however, its description is outside the scope of this paper.

C. Predicate Abstraction

In [24], the authors combined predicate abstraction with convex polyhedral analysis for the verification of reachability properties of linear hybrid systems. A similar but more general abstraction approach was proposed in [25]. In [26] a qualitative-based approach was developed for an abstract model generation for hybrid systems, based on higher derivative analysis. We distinguish ourselves from the above in several aspects. First, although the predicates used in [24] are manually provided, we extract qualitative predicates from the system behavior, which can complement the qualitative predicates presented in [26]. We also use different ideas for the transition relation, based on a variant of the mean value theorem (MVT).

An invariant-based approach was proposed in [27], where the problem of constructing invariants is turned into a constraint-solving problem. In [28], the authors proposed a similar framework using the idea of barrier certificates. Barrier certificates, if they exist, are invariants that separate system behavior from a bad state and hence provide a safety verification approach. The work presented in this paper is different from the above mentioned work. We distinguish ourselves by not limiting the verification to invariant checking, allowing more reasoning capabilities on the circuits of interest.

III. Analog Design Modeling

The analog component of embedded systems is usually composed of circuits built from basic passive and active components (resistors, capacitance, inductance, transistors, etc.), connected to various current and voltage sources in a certain topology, achieving a specific desirable behavior (e.g., filtering, amplification, etc.). In this section we will provide the method of obtaining the equations describing the design behavior from the design description.

A. Bond Graphs as a Model for Analog Designs

Bond graphs were introduced by Paynter [29] who hypothesized that all physical systems and the interactions between them could be modeled using energy and power alone. His work was extended later on by Karnopp and Rosenberg [30] to enable the bond graph theory to be used in practice. They developed multiport objects that could be used with power bonds to model the flow of energy and information. The benefit of a modeling framework based on energy flow is that different domains can be analyzed using the same methodology.

Bond graphs define a necessary and sufficient set of primitives for the modeling of a wide range of practical systems. The necessary and sufficient set of primitives consists of five elements, but normally a more practical set of nine elements is used as shown in Table 1. The storage group contains the elements for capacitive storage (C type) and inductive storage (I type). The supply group contains the sources of effort and flow. The reversible transformation group contains a transducer and gyrator. The irreversible transformation group contains the elements for thermal losses and entropy-producing processes, whereas the distribution group contains junctions that represent the generalized domain independent Kirchhoff current and voltage laws [31].

1. Connections

Bond graphs are based on the first principle of energy conservation. The most basic element of a bond graph is the power bond (Fig. 3a). It is the energy link between two components. It is represented graphically by a harpoon (half arrow), which points in the direction of positive power flow. The bond represents two variables, effort and flow. In the electrical domain the effort variable is represented by voltage and the flow by current. It follows that the product of the effort and flow variables represents the power flowing through the bond. Additional variables can also be derived

Table 1 Basic objects of bond graphs

Group	Components	Electrical domain example
Storage	Capacitive inertial	Capacitance inductance
Supply	Source of effort, source of flow	Voltage source, current source
Reversible transformation	Transducer, gyrator	Transformer
Irreversible transformation	Entropy-producing process	Thermal resistance
Distribution	0 and 1 junctions	KVL, KCL

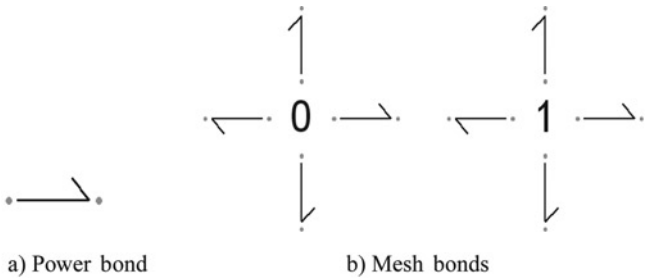


Fig. 3 Basic bonds.

from the bonds. The displacement and momentum energy variables are related to the energy and flow by their time derivatives.

The next basic component is the junction, which represents a circuit node or mesh (Fig. 3b). At the 0 or common-effort junction, the efforts are equal, which is analogous to a node in a circuit. At the 1 or common-flow junction, the flows are equal, which is analogous to a mesh in a circuit.

2. Components

Using the bonds and junctions, it is possible to connect components together in a bond graph. There are different types of single and multiport interfaces that can be used to represent many configurations. The single-port components are described below. The first basic elements are the sources of effort or flow. They are analogous to voltage and current sources in circuit diagrams. Additional single-port components are used to represent resistors, capacitors, and inductors. They are denoted using the letters R, L, or C (Fig. 4a).

It is possible to represent other electrical circuit components, such as transformers, gyrators, and switches using two port interfaces but their application and description are beyond the scope of this paper. It is important to note though that more advanced components exist and they can be used to model electronic components beyond simple analog ones.

We have now seen how a given bond graph and a set of constitutive relations maps to a mathematical model of the underlying system. A preferred alternative is a sequence of directed assignment statements such that unknowns can be immediately and sequentially computed from the knowns on the right-hand side. Such a model is sometimes referred to as a computational model. Such a causal computational model requires the model variables to be ordered in a specific cause-effect relationship.

3. Causality

Causality is the determination and representation of the directional relationship between an input and an output [30]. By adding a causal bar to the end of a bond, the system equations that represent the two variables of effort and flow can be indicated explicitly. There are many rigorous explanations on how to assign the causality of a bond and how it relates to the system as a whole [29–31]. Fortunately, a simple definition exists that can be used for the direct translation of circuit diagrams. The causal stroke is attached to the side of the bond that computes the flow variable [32]. It is not necessary to assign causality because tools exist to automatically assign it to bond graphs. It is important for the modeler to know how to assign causality manually because it can aid in the development of complex bond graphs (Fig. 4b).

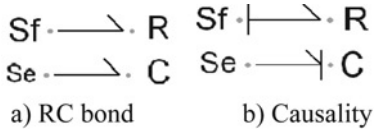


Fig. 4 Bond graphs basics.

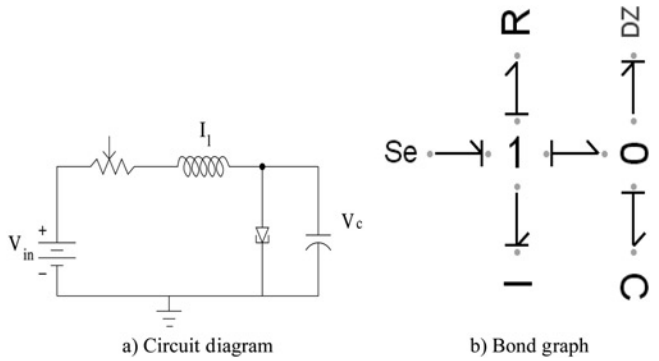


Fig. 5 Tunnel diode oscillator.

Example 1.
The tunnel diode oscillator circuit in Fig. 5a, which has been used by many researchers (e.g., [19,21]) as a benchmark, will be used as an example throughout the paper to demonstrate each step of our methodology. The tunnel diodes exploit a phenomenon called resonant tunneling due to its negative resistance characteristic at very low forward bias voltages. This means that for some range of voltages, the current decreases with increasing voltage. This characteristic makes the tunnel diode useful as an oscillator.

The corresponding bond graph generation goes as follows. Each circuit diagram component is transformed into its bond graph counterpart. Circuit nodes are represented by 0 junctions and meshes are represented by 1 junctions as shown in Fig. 5b. This is done according to the bond graphs rules described earlier.

4. Simplification

By choosing to combine certain bond graph elements, it is possible to reduce the complexity of the system without affecting the overall function. This can result in simpler DAEs that are extracted from the reduced bond graph model. By using a simpler model, the number of states can be reduced, allowing for a less complex verification problem.

The BondLib library developed by Cellier et al. [5] demonstrates the benefit of object-oriented modeling with bond graphs. The transistor models for bipolar junction transistor (BJTs) and MOSFETS are true HSpice models that can be set to different levels of complexity [5]. At each level, parasitics, current leakages, and nonideal effects can be added to the model by specifying the correct parameter. The parameters are available to the modeler to dynamically alter the bond graph level.

B. Describing the Analog Behavior

Once the bond graph is built, the set of system equations can be extracted and simplified. We use Mathematica simplification functionalities [12] in order to remove redundant equations through rewriting techniques. The final system of equations is the computational model on which we apply the verification.

The dynamical behavior of analog designs is usually represented through equations describing the progressive change of the state variables. These state variables can be regarded as memory elements that are able to preserve previous states for a certain amount of time. For instance at the circuit-level capacitance can be seen as a voltage storage element, whereas inductance as a current storage element.[#] Analog circuits can be described by nonlinear polynomial ODEs as follows:

Definition 1 (Analog circuit model).
An analog circuit model is a tuple $\mathcal{A} = (\mathcal{X}, \mathcal{X}_0, \mathcal{P})$, with $\mathcal{X} = V_{c_1} \times V_{c_n} \times \cdots \times I_{l_m} \subseteq \mathbb{R}^d$ as the continuous state space with d-dimensions, where V_{c_i} and I_{l_j} are the voltage across the capacitance c_i and the current through the

[#] It is worth noting that a resistance is a memoryless element.

inductance l_j, respectively. $\mathcal{X}_0 \subseteq \mathcal{X}$ is the set of initial states (initial voltages on the capacitances and currents through the inductance), and $\mathcal{P} : \mathcal{X} \to \mathbb{R}^d$ is the continuous vector field.

The behavior of such an analog circuit model $\mathcal{A}$ is governed by polynomial differential equations of the form:

$$\dot{x_k} = \frac{\mathrm{d}x_k}{\mathrm{d}t} = \mathcal{P}_k(x_1, \ldots, x_d) = a_0 + \sum_{l=1}^{m} \mathcal{P}_{l,k}(x_1, \ldots, x_d)$$

where t is the independent real time, $\mathcal{P}_k$ $(k = 1, \ldots, d)$ is a polynomial of degree m, a_0 is a constant, and $\mathcal{P}_{l,k}$ is a polynomial of degree l with

$$\mathcal{P}_{l,k} = \sum_{i_1+\cdots+i_d=l} a_{i_1,\ldots,i_d} x_1^{i_1} \ldots x_d^{i_d}$$

where $a_{i_1,\ldots,i_d}$ is a constant. We assume that the differential equation has a unique solution for each initial value.

The semantics of the analog model $\mathcal{A} = (\mathcal{X}, \mathcal{X}_0, \mathcal{P})$ over a continuous time period $T_c = [\tau_0, \tau_1] \subseteq \mathbb{R}^+$ can be described as a trajectory $\Phi_x : T_c \to \mathcal{X}$ for $x \in \mathcal{X}_0$ such that $\Phi_x(t)$ is the solution of $\dot{x_k} = \mathcal{P}_k(x_1, \ldots, x_d)$, with initial condition $\Phi_x(0) = x$ and $t \in T_c$, being a time point. We can view the behavior of the analog model $\mathcal{A}$ as a transition system:

Definition 2 (Analog transition system).
The transition system for analog model $\mathcal{A}$ is described as a tuple $\mathcal{T}_\mathcal{A} = (Q, Q_0, \sigma, L)$, where $q \in Q$ is a configuration (x, Γ), $x \in \mathcal{X}$ and Γ is a set of intervals where $\cup_{i \geqslant 0} t_i \subseteq \mathbb{R}^+$, $t_i \in \Gamma$. We have $t_1, t_2 \in \Gamma$ for $\Phi_{x'}(t_1) = \Phi_{x''}(t_2) = x$ and $x', x'' \in \mathcal{X}_0$. $q \in Q_0$, when $t_0 \in \Gamma$ and t_0 is the singular interval, $\sigma \subseteq Q \times Q$ is a transition relation such that $(q_n, q_m) \in \sigma$ if and only if $\exists t_n \in \Gamma_n$, $\exists t_m \in \Gamma_m$. $t_n < t_m$, and $\lim_{t_n \to t_m} \Phi_x^{q_n}(t_n) = \Phi_x^{q_m}(t_m)$, $x \in \mathcal{X}_0$. Finally, L is an interpretation function such that $L : Q \to \mathbb{R}^n \times 2^{\mathbb{R}^+}$.

The set of reachable states Reach can then be defined as follows: Reach $:= \{q' \in Q | \exists q \in \text{Reach}^{(0)}, t \in L_\Gamma(q'), x' = L_x(q'), x = L_x(q)$ such that $\Phi_x(t) = x'\}$, where $\text{Reach}^{(0)} := Q_0$.

Example 2.
Consider again the tunnel diode oscillator circuit in Fig. 5a. We focus on the current I_L and the voltage V_C across the tunnel diode in parallel with the capacitor. The tunnel diode state equations are extracted from the simplified bond graph (Fig. 5b) using Dymola. The final equations are given as $\dot{V_C} = \frac{1}{C}(-I_d(V_C) + I_L)$ and $\dot{I_L} = \frac{1}{L}(-V_C - \frac{1}{G} I_L + V_{\text{in}})$, where $I_d(V_C)$ describes the nonlinear tunnel diode behavior and V_{in} is the DC voltage source. The extracted equations will be used as an input for the verification engine described in the next section.

IV. Verification Methodology

The verification methodology we propose is illustrated in Fig. 6. Starting with a circuit description as a system of ODEs, along with specification properties provided in computational temporal logic (∀CTL) [13], we symbolically extract qualitative predicates of the system. The abstract model is constructed in successive steps. In the basis step, we only consider predicates that define the invariant regions for the system of equations based on the Darboux theory of integrability [33]. Informally, the Darboux theory is concerned with the identification of the different qualitative behaviors of the continuous state space of the system. We make use of such an idea to divide the analog design state space into qualitatively distinct regions where no transition is possible between states of the different regions. Satisfaction of properties is verified on these regions using constraint-based methods, which rely on qualitative properties of the system, by generating new constraints that prove or disprove a property. Hence property verification provides the advantage of avoiding explicit computation of reachable sets.

If the property cannot be verified at this stage, refinement is needed only for the nonverified regions by adding more predicates. Conventional model checking is then applied on the newly generated abstract model. The extraction of the predicates is incremental in the sense that more precision can be achieved by adding more information to the

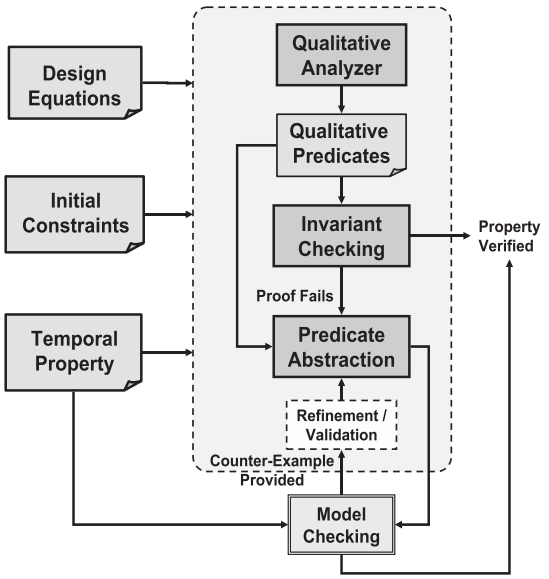


Fig. 6 Verification methodology.

original construction of the system. When the property is marked violated, one possible reason is because of the false negative problem due to the over-approximation of the abstraction. In this case, refinement techniques may be introduced.

A. Predicate Abstraction

Predicate abstraction is a method where the set of abstract states is encoded by a set of Boolean variables each representing a concrete predicate. Based on [24], we define a discrete abstraction of the analog model $\mathcal{A}$ with respect to a given n-dimensional vector of predicates $\Psi = (\psi_1, \ldots, \psi_n)$, where $\psi : \mathbb{R}^d \to \mathbb{B}$, with $\mathbb{B} = \{0, 1\}$ and d being the dimension of the ODE system. A polynomial predicate is of the form $\psi(x) := \mathcal{P}(x_1, \ldots, x_d) \sim 0$, where $\sim \in \{<, \geqslant\}$. Hence, the infinite state space $\mathcal{X}$ of the system is reduced to 2^n states in the abstract system, corresponding to the 2^n possible Boolean truth evaluates of Ψ.

Definition 3 (Abstract transition system).
An abstract transition system is a tuple $\mathcal{T}_\Psi = (Q_\Psi, \rightsquigarrow, Q_{\Psi,0})$, where:

- $Q_\Psi \subset L \times \mathbb{B}^n$ is the abstract state space for n-dimensional vector predicates, where an abstract state is defined as a tuple (l, b), with $l \in L$ being a label and $b \in \mathbb{B}^n$.
- $\rightsquigarrow \subseteq Q_\Psi \times Q_\Psi$ is a relation capturing abstract transitions such that $\{b \rightsquigarrow b' | \exists x \in \Upsilon_\Psi(b), t \in \mathbb{R}^+ : x' = \Phi_x(t) \in \Upsilon_\Psi(b') \wedge x \to x'\}$, where the concretization function: $\Upsilon_\Psi : \mathbb{B}^n \to 2^{\mathbb{R}^d}$ is defined as $\Upsilon_\Psi(b) := \{x \in \mathbb{R}^d | \forall j \in \{1, , \ldots, n\} : \psi_j(x) = b_j\}$.
- $Q_{\Psi,0} := \{(l, b) \in Q_\Psi | \exists x \in \Upsilon_\Psi(b), x \in \mathcal{X}_0\}$ is the set of abstract initial states.

We define the set of reachable states as follows: $\text{Reach}_\Psi = \bigcup_{i \geqslant 0} \text{Reach}_\Psi^{(i)}$, where $\text{Reach}_\Psi^{(0)} = Q_{\Psi,0}$, $\text{Reach}_\Psi^{(i+1)} = \text{Post}_c(\text{Reach}_\Psi^{(i)})$, $\forall i \geqslant 0$ and $\text{Post}_c(l, b) := \{(l', b') \in Q_\Psi | (l, b) \rightsquigarrow (l', b')\}$. We can then deduce the following property between concrete and abstract reachable states.

Lemma 1.
Given an analog abstract transition system $\mathcal{T}_\Psi(\mathcal{A})$ and a vector of predicates Ψ, the following holds: $\text{Reach} \subseteq \{q \in Q | \exists (l, b) \in \text{Reach}_\Psi : x \in \Upsilon_\Psi(b) \wedge L_x(q) = x\}$**

** The proof of the lemma can be found in [34].

B. Abstraction-Based Verification

The common concept between safety verification based on constraint solving and model checking based on predicate abstraction is the requirement of over-approximation for the reachable states.

Given that the analog model transition system $\mathcal{T}_\mathcal{A}$ representing the analog behavior and a property φ expressed in ∀CTL, the problem of checking that the property holds in this model written as $\mathcal{T}_\mathcal{A} \models \varphi$ can be simplified to the problem of checking that a related property holds on an approximation of the model $\mathcal{T}_\Psi$, i.e., $\mathcal{T}_\Psi \models \varphi$. More formally, the main preservation theorem can be stated as follows [25]:

Theorem 1.
Suppose $\mathcal{T}_\Psi$ is an abstract model of $\mathcal{T}_\mathcal{A}$, then for all ∀CTL state formulas describing $\mathcal{T}_\Psi$ and every state of $\mathcal{T}_\mathcal{A}$, we have $\tilde{s} \models \tilde{\varphi} \Rightarrow s \models \varphi$, where $s \in \gamma(\tilde{s})$. Moreover, $\mathcal{T}_\Psi \models \tilde{\varphi} \Rightarrow \mathcal{T}_\mathcal{A} \models \varphi$.

If a property is proved on an abstract model $\mathcal{T}_\Psi$, then we are done. If the verification of $\mathcal{T}_\Psi$ reveals $\mathcal{T}_\Psi \not\models \tilde{\varphi}$, then we cannot conclude that $\mathcal{T}_\mathcal{A}$ is not safe with respect to $\tilde{\varphi}$, since the counterexample for $\mathcal{T}_\Psi$ may be spurious. To remove spurious counterexamples, refinement methods can be applied on the abstract model. The proof of Theorem 1 can be found in [25].

C. Invariants

Usually, a continuous system has a behavior that varies in different regions of phase space which boundaries are defined by special system solutions known in the literature as Darboux invariants [33]. These invariants partition the concrete state space into a set of qualitative distinctive regions.

Definition 4.
Given the system of ODEs $\frac{\mathrm{d}x_k}{\mathrm{d}t} = \mathcal{P}_k(x_1(t), \ldots, x_d(t))$, with $k = 1, \ldots, d$ ($\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{P}(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^d$, and $\mathbf{P} = (\mathcal{P}_1, \ldots, \mathcal{P}_d)$), we define the corresponding *vector field* as $\mathcal{D}_\mathbf{P} = \mathbf{P}.\partial_\mathbf{x} = \sum_{k=1}^{d} \mathcal{P}_k \frac{\partial}{\partial x_k}$.

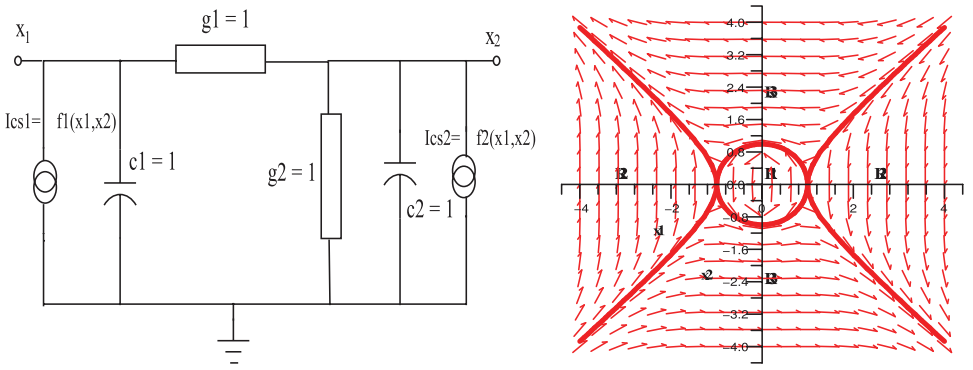
The correspondence between the system of ODEs and the vector field $\mathcal{D}_\mathbf{P}$ is obtained by defining the time derivative of functions of $\mathbf{x}$ as follows. Let $\mathcal{G}$ be a function of $\mathbf{x}$: $\mathcal{G} : \mathbb{R}^k \to \mathbb{R}$, then $\frac{\mathrm{d}\mathcal{G}}{\mathrm{d}t} := \dot{\mathcal{G}} = \mathcal{D}_\mathbf{P}(\mathcal{G}) = \mathbf{P}.\partial_\mathbf{x}\mathcal{G}$. The time derivative is called the derivative along the flow since it describes the variation of function $\mathcal{G}$ of $\mathbf{x}$ with respect to t as $\mathbf{x}$ evolves according to the differential system. When $D_\mathbf{P}(\mathcal{G}) = 0$, $\forall \mathbf{x} \in \mathbb{R}^k$, we have a time independent first integral of $D_\mathbf{P}$. Several methods were developed recently based on the Darboux integrability theory [21], which is a theory concerned with finding closed form solutions of a system of ODEs, to tackle the problem by looking for a basis set of invariants, i.e., Darboux invariants. Rather than looking at functions that are constant on all solutions, we look at functions that are constant on their zero level set. Darboux polynomials $\mathcal{J}_i$ provide the essential skeleton for the phase space from which all other behaviors can be qualitatively determined.

Definition 5 (Darboux Polynomials [33]).
Given a vector field $\mathcal{D}_\mathcal{P} = \sum_{i=1}^{d} \mathcal{P}_i \frac{\partial}{\partial x_i}$ associated with the system $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{P}(\mathbf{x})$, a Darboux polynomial is of the form $\mathcal{J}(\mathbf{x}) = 0$ with $\mathcal{J} \in \mathbb{R}[\mathbf{x}]$, when $D\mathcal{J} = \mathcal{K}\mathcal{J}$, where $\mathcal{K} = \mathcal{K}(\mathbf{x})$ is a polynomial called the cofactor of $\mathcal{J} = 0$.

Lemma 2.
Given a system of ODEs and a vector field $\mathcal{D}_\mathbf{f}$, $\mathcal{J}$ is an invariant of the system if $\mathcal{J}$ divides $\mathcal{D}_\mathbf{f}$, more formally, if there exists $\mathcal{K} \in \mathbb{R}[\mathbf{x}]$ such that $\mathcal{D}_\mathbf{f}(\mathcal{J}) = \mathcal{K}\mathcal{J}$. The solution set of the system vanishes on the curve of $\mathcal{J}$.

Proof. We can always represent the system by the associated vector field at each point $\mathcal{F}(\mathbf{x}) = \mathbf{P}(\mathbf{x})$ and $\nabla\mathcal{J} \cdot \mathcal{F} = k\mathcal{J}$, where $\nabla\mathcal{J}$ denotes the gradient vector related to $\mathcal{J}(\mathbf{x})$ and $\cdot$ is the scalar product. When $\mathcal{J} = 0$, $\nabla\mathcal{J} \cdot \mathcal{F} = 0$, meaning that $\nabla\mathcal{J}$ is orthogonal to the vector field $\mathcal{F}$ at these points. Therefore $\mathcal{F}$ is tangent to $\mathcal{J} = 0$.

In the context of abstraction, we define the invariant regions as conjunction of Darboux invariant predicates. An invariant region can be considered as an abstraction of the state space that confines all the system dynamics initiated in that region: ■



a) Analog circuit schematic b) Darboux invariants

Fig. 7 Illustrative nonlinear analog circuit.

Definition 6 (Invariant Regions).
We say that a region $\mathcal{V}$ is an invariant region of an analog model $\mathcal{A}$ such that $\mathcal{P}(\mathbf{x}(0)) = s_0 \models \mathcal{V}$, $\mathcal{P}(\mathbf{x}(\varsigma)) = \mathbf{s}_\varsigma \models \mathcal{V}$, and $\forall t \in [0, \varsigma]$, $\mathcal{P}(\mathbf{x}(t)) = s_t \models \mathcal{V}$. Let $\mathcal{V} = \{x \in \mathbb{R}^k | x \models \Gamma\}$, be an invariant region, where Γ is a conjunction of Darboux predicates (each is of the form $p(\mathbf{x}) \sim 0$, where p is a polynomial function and $\sim \in \{<, \geqslant\}$). If $\mathbf{x}(0)$ is some initial state, then $\mathcal{V} = \mathcal{V}(\mathbf{x}(0))$ denotes an over-approximation of the set of states reachable from $\mathbf{x}(0)$.

Example 3.
Consider the nonlinear circuit shown in Fig. 7a, where the nonlinearity comes from the voltage controlled current sources for which currents I_{cs1} and I_{cs2} are described, respectively, as $f_1 = -x_2^3 + x_1 - x_2$ and $f_2 = -x_1^3 + 2x_2$. The voltages across the capacitors c_1 and c_2 can be described using ODEs, respectively, as follows: $\dot{x_1} = -x_2^3$ and $\dot{x_2} = x_1 - x_1^3$. We identify the corresponding invariants: $j_1 = 1 - x_1^2 - x_2^2$ and $j_2 = 1 - x_1^2 + x_2^2$, which are used to form three invariant regions: $R_1 = j_1 \geqslant 0 \wedge j_2 \geqslant 0$, $R_2 = j_1 < 0 \wedge j_2 < 0$, and $R_3 = j_1 < 0 \wedge j_2 \geqslant 0$ as shown in Fig. 7b. Note that $j_1 \geqslant 0 \wedge j_2 < 0$ is infeasible and therefore discarded.

D. Constraint-based Verification

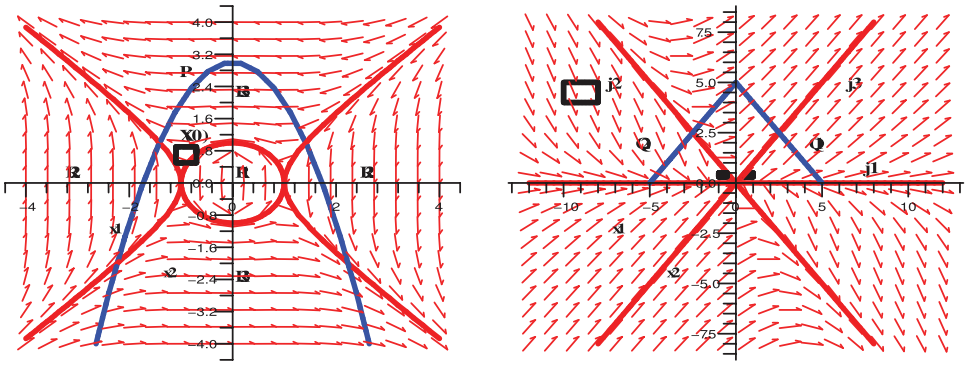
Constraint solving is the study of systems based on constraints (relation between the variables of the system). The idea of constraint solving is to solve problems by stating constraints about the problem area and, consequently, finding solutions satisfying all the constraints. Two categories of constraint solvers are identified [35]:

- Satisfiability constraint solvers: when a constraint solver pronounces the existence of a solution, the constraints are guaranteed to have a numerical solution. In addition, if a solution is produced, then it is guaranteed that this solution satisfies the constraints. One such solver is Rsolver [36] and Mathematica built-in functions like *Reduce* and *FindInstance* [12].
- Unsatisfiability constraint solvers: if a constraint solver pronounces the infeasibility of the input constraints, then this result is sound. If no solution is produced, then this means that the system is infeasible. Realpaver [37] is an example of this category.

In constraint-solving techniques, the uncertainty of numerical variables are over-approximated using intervals of real numbers to make safe decisions possible. Interval-based arithmetic techniques provide efficient and safe methods for solving continuous constraint satisfaction problems where real variables are constrained by equalities and inequalities. The soundness is inherited from the inclusion property of interval arithmetics [38].

V. Invariant Based Verification

In this section, we propose a qualitative verification approach for analog circuits based on constraint-based methods. The basic idea is to apply quantified constraint-based techniques to answer questions about qualitative behaviors of the designs, by constructing functions that validate or falsify the property. The idea is different from



a) Safety verification (Example 4) b) Switching verification (Example 5)

Fig. 8 Constraint based verification for the circuit in Fig. 7(a).

conventional approaches as it does not require the computation of the explicit reachable states. We consider two types of properties that can be verified using that approach, namely safety and switching properties.

A. Safety Properties

Safety properties can be expressed in CTL [13] as $\forall\Box p$; meaning that always on all executions the constraint predicate p is satisfied for a set of initial conditions. The verification starts by getting the dual property $\exists\Diamond\neg p$ (which means that there is an execution falsifying the constraint p) and applies constraint solving on the dual property within the invariant regions of interest. In case of unsatisfiability, we conclude that the original property is satisfied in the region, otherwise we cannot conclude the truth of the property and a refinement model providing more details of the region is constructed.

Proposition 1 (Safety Property Verification).
$\forall\Box\mathcal{P}$ is always satisfied in an invariant region $\mathcal{V}$, if its dual property $\exists\Diamond\neg\mathcal{P}$ is never satisfied in that region[††] .

Example 4.
Consider the circuit in Example 3, with initial conditions $x_1(0) \in [-0.7, -1.1]$ and $x_2(0) \in [0.5, 0.9]$. Suppose the property to check is $\forall\Box\mathcal{P} := x_1^2 + x_2 - 3 < 0$ (see Fig. 8a), meaning that all flows initiated from $\mathbf{x}(0) = (x_1(0), x_2(0))$, will be bounded by $x_1^2 + x_2 - 3$. The following regions satisfy the initial conditions $R_1 = j_1 \geqslant 0 \wedge j_2 \geqslant 0$ and $R_3 = j_1 < 0 \wedge j_2 \geqslant 0$. We check whether $\exists\Diamond\mathcal{P} := x_1^2 + x_2 - 3 \geqslant 0$ is satisfiable in the invariant regions R_1 and R_3. By applying constraint solving in *Mathematica*, we find that for the region R_3, the constraints system is satisfiable, and hence the original property cannot be verified and the state space of the region needs to be refined. For the region R_1, the constraints system is infeasible, and therefore we conclude that the safety property is satisfied.

It is worth noting that the barrier-certificate method [28][‡‡] can be applied as a complementary to our method. In fact, Darboux predicates used as the basis of invariant regions can be considered as natural barrier certificates that are constructed without the need of initial and final constraints, hence reducing computational efforts.

B. Switching Properties

A special case of reachability verification $\exists\Diamond\mathcal{Q}$ is the switching condition verification, i.e., starting from a set of initial conditions, the system will eventually cross a threshold, triggering a switching condition. Such property

[††] More details about the techniques as well as the proofs of the propositions in this paper can be found in [34].
[‡‡] A barrier certificate is an invariant that separates system behavior from a bad state and hence is used as a proof of safety verification.

is of great importance, for instance, a MOSFET transistor acting as a switch changes states based on the voltage condition applied on its gate. We consider here a restricted form of the switching property, where we assume that threshold predicates divide the invariant region by intersecting the invariant region boundaries (at least two Darboux predicates). Given an invariant region $\mathcal{V}$, a predicate $\mathcal{Q}$ is a *switching condition* if $\bigwedge_{i=0}^{k} \exists \mathbf{x}.(\mathcal{Q}(\mathbf{x}) = 0) \wedge (\mathcal{I}_i(\mathbf{x}) = 0)$, where $k \leqslant 2$ and $\mathcal{I}$ is a Darboux invariant. The switching verification can be stated as follows:

Proposition 2 (Switching Property Verification).
$\exists \Diamond \mathcal{Q}$ is satisfied in a region $\mathcal{V}$, if $\mathcal{Q}(\mathbf{x}(0)) < 0$ and $\mathcal{D}_{\mathbf{P}}(\mathcal{Q}) > 0$ or if $\mathcal{Q}(\mathbf{x}(0)) > 0$ and $\mathcal{D}_{\mathbf{P}}(\mathcal{Q}) < 0$, in the region $\mathcal{V}$. If these conditions are satisfiable, we conclude that the property is verified and switching occurs.

Example 5.
Consider the circuit shown in Fig. 7(a), where the voltages across the capacitors c_1 and c_2 are described, respectively, as follows: $\dot{x_1} = x_1^2 + 2x_1x_2 + 3x_2^2$ and $\dot{x}_2 = 4x_1x_2 + 2x_2^2$, with initial conditions $x_1(0) \in [0.5, 1]$ and $x_2(0) \in [0.3, 0.5]$. Suppose that the switching condition property to check is $\exists \Diamond x_1 + x_2 - 5 = 0$, meaning that switching occurs when a certain trajectory crosses the threshold $Q_1 := x_1 + x_2 - 5 = 0$ (see Fig. 8b). We construct the Darboux functions: $j_1 := x_2$, $j_2 := x_1 + x_2$, $j_3 := x_1 - x_2$. The region $R_1 = j_1 > 0 \wedge j_2 > 0 \wedge j_3 > 0$ satisfies the initial conditions. In addition, the predicate $x_1 + x_2 - 5 < 0$ satisfies the initial condition and $\mathcal{D}_{\mathbf{P}}(x_1 + x_2 - 5) > 0$ because $\mathcal{D}_{\mathbf{P}}(x_1 + x_2 - 5) = (x_1 + x_2)(x_1 + 5x_2)$ is always positive in R_1. Consider the initial conditions $\mathbf{X}(0)_1 :=$ ($x_1(0) \in [-10, -8]$ and $x_2(0) \in [4, 5]$) and $\mathbf{X}(0)_2 :=$ ($x_1(0) \in [-0.5, -1]$ and $x_2(0) \in [0.3, 0.5]$) in the invariant region $R_2 = j_1 > 0 \wedge j_2 < 0 \wedge j_3 < 0$. For the switching condition $Q_2 := -x_1 + x_2 - 5 = 0$, we find that the initial condition $\mathbf{X}(0)_1$ satisfies $-x_1 + x_2 - 5 > 0$, and $\mathbf{X}(0)_2$ satisfies $-x_1 + x_2 - 5 < 0$ while $\mathcal{D}_{\mathbf{P}}(-x_1 + x_2 - 5) = -(x_1 - x_2)^2$ will always be negative in region R_2, and therefore we conclude that the switching will occur for the initial condition $\mathbf{X}(0)_1$ but not for $\mathbf{X}(0)_2$.

Sometimes constraint-based verification fails to provide answers for the verification problem as the above methods are not complete in general. In addition, more complex properties like oscillation cannot be proved using the above method. We complement the approaches described in this section, by the predicate abstraction method allowing conventional model checking to be applied.

VI. Predicate Abstraction

A. Abstract State Space

In general, the effectiveness of the predicate abstraction method depends on the choice of predicates. In addition to using Darboux predicates described in Sect. IV.C, we choose predicates identified in the properties of interest. In addition to temporal property predicates, basic ideas from the qualitative theory of continuous systems can be adapted within the predicate abstraction framework. The termination of the predicate generation phase is not necessary for creating an abstraction. We can stop at any point and construct the abstract model. A larger predicate set yields a finer abstraction as it results in a larger state space in the abstract model.

A set of predicates can be constructed using the notion of *critical forms*, which are special functions along them, the vector field direction being either vertical or horizontal. In between these forms, there can be no vertical nor horizontal vectors. In a region (abstract state) determined by the critical forms, all vectors follow one direction. These predicates can be obtained easily by setting $\dot{\mathbf{x}} = 0$. A generalization of critical forms is the concept of *isoclines*. Isoclines are functions over which the system trajectories have a constant slope. A predicate π is an isocline of a system of ODEs if and only if $\exists a_i \in \mathbb{R}$ with $i = 1, \ldots, d$ such that $\Sigma_{i=1}^{d} a_i \mathcal{P}_i(\mathbf{x})|_{\pi} = 0$. Isocline and critical forms provide qualitative information about the system behavior. Hence, such information can be used in refuting certain behavior that is shown unreachable. For instance, by knowing different constants a_i, we deduce the direction of the flow crossing the isoclines and therefore we decide how to build transitions between abstract states. Finding different isocline predicates within an invariant region can be achieved by solving constraints on the parameters of predefined forms of an isocline predicate.

Another kind of predicates, we propose, referred to as *conditioned predicates*, has the property that under specific conditions, they provide certain information about the solution flow. A predicate π is a conditioned predicate of a system of ODEs with conditions $\Gamma_1, \ldots, \Gamma_d$, if it is of the form $\sum_{i=1}^{n} \Gamma_i \mathcal{P}_i(\mathbf{x})|_{\pi} = 0$, where the conditions Γ_i are

polynomials with $i = 1, \ldots, d$, and d is the system dimension. For instance, consider the 3-dimensional system with the state variables x, y, z, and the property predicate $z > 1$. We can construct another predicate that intersects $z > 1$ at specific conditions, say $\frac{\dot{y}}{\dot{x}} = 0$. Then, the new predicate is of the form $\dot{y} - (z-1)\dot{x} = 0$.

Example 6.
Consider the analog circuit in Example 3. The critical forms predicates are $p_1 := x_1$, $p_2 := x_2$, $p_3 := 1 - x_1$, and $p_4 := 1 + x_1$, as shown in Fig. 9a. For illustration purposes, we choose two isocline predicates $p_5 := x_1 - x_1^3 + x_2^3$ and $p_6 := x_1 - x_1^3 - x_2^3$ as shown in Fig. 9b. Suppose we are interested to verify a property including the predicate $p_7 := x_2 - x_1 > 0.3$, we can construct the conditioned predicate $p_8 := \dot{x_2} - (x_2 - x_1 - 0.3)\dot{x_1} = 0$ as shown in Fig. 9c. To build the abstract state space, we have three invariant regions and eight predicates. As certain combination of predicates are infeasible, the number of abstract states is $<2^8$ abstracts states. In fact, region $R_1 = j_1 \geqslant 0 \wedge j_2 \geqslant 0$ is subdivided into 29 abstract states.

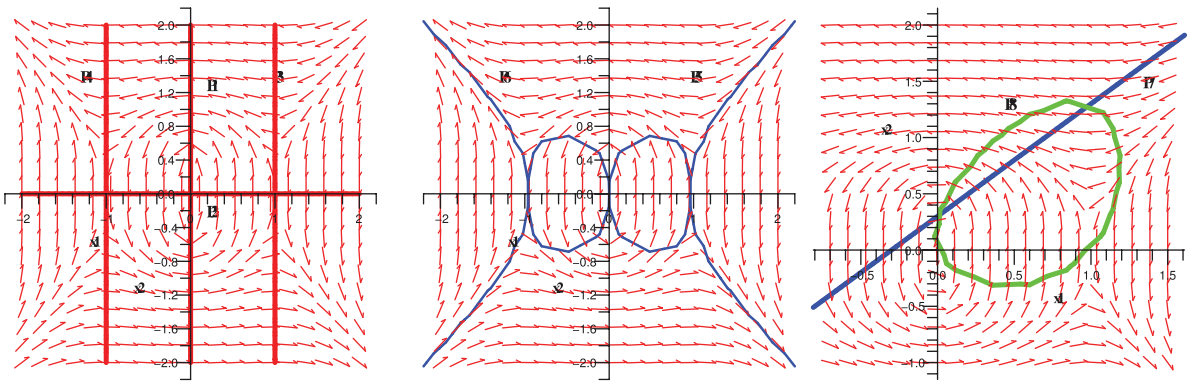
Other methods for finding useful predicates were developed in [26], where the authors proposed a way to extract predicates from polynomial ODEs by looking at higher derivatives. If $p \in P$, then add $\dot{p}$, the derivative (with respect to time) of p, to the set P unless $\dot{p}$ is a constant or a constant factor multiple of some existing polynomial in P.

Predicates related to the basic functionality of the design of interest can also be provided in a manual fashion. The conventional analysis of circuits can be an interesting direction for obtaining attractive predicates. It is worth noting that the termination of the predicate generation phase is not necessary for creating an abstraction. We can stop at any point and construct the abstract model. A larger predicate set yields a finer abstraction as it results in a larger state space in the abstract model.

B. Computing Abstract Transitions

One main issue in constructing abstract state transition systems is the identification of the possible transitions. As we divide the state space into invariant regions, we need only to construct transitions between abstract states within a region. Therefore, we do not need to construct an abstract model for the whole state space. In general, information from the solution of the ODEs is required to describe transitions between abstract states. In practice, each abstract transition is initialized to the trivial relation relating all states and then stepwise refined by eliminating infeasible transitions. This guarantees that any intermediate result represents an abstraction and the refinement can be stopped at any point of time. In the remaining of this section, we use a set of different rules to construct transition between abstract states.

The simplest rule to use is the *Hamming distance (HD) rule* [26]. The HD is the number of predicates for which the corresponding valuations are different in different abstract states. For instance, the HD between state $s_1 := (p_1 = 1 \wedge p_2 = 0 \wedge p_3 = 1 \wedge p_4 = 1)$ and state $s_2 := (p_1 = 1 \wedge p_2 = 0 \wedge p_3 = 0 \wedge p_4 = 1)$ is 1, written as $\mathrm{HD}(s_1, s_2) = 1$. Given two abstract states s_1 and s_2, we say that a transition exists between two abstract states only



a) Critical forms predicates b) Isocline predicates c) Conditioned predicates

Fig. 9 Predicates for the circuit in Fig. 7(a).

if $\mathrm{HD}(s_1, s_2) = 1$. The next rule we apply is based on the *generalized MVT* [35], which is an extension of the MVT for n-dimension.

Theorem 2 ([35]).
If $\mathbf{x}(t)$ is continuous on a time interval $t_1 \leqslant t \leqslant t_2$ and differentiable on $t_1 < t < t_2$, and assuming that there exists a vector $\mathbf{V}$ orthogonal to $\mathbf{x}(a)$ and to $\mathbf{x}(b)$, then there is $t_c : t_1 < T_c < t_2$ such that $\mathbf{V}$ is orthogonal to $\dot{\mathbf{x}}(t_c)$.

We use quantified constraint-based methods to check whether such a condition is satisfied between two abstract states. If the MVT is not satisfied, we deduce that no transition exists between the two states. The above rules give an over-approximation of the transition system as no information about the vector field direction is used. To remove such redundant transitions in the region of interest, we complement the above rules by applying the *intermediate value theorem* as a way to identify the flow direction. In the context of abstraction, a transition between two abstract states exists if a predicate valuation changes during the execution over an interval domain as follows:

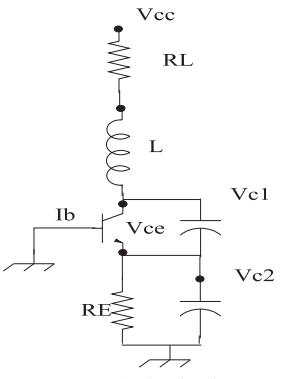
Theorem 3.
Given a predicate λ, two states $S_1 = (l, b)$ and $S_2 = (l', b')$ differing only on the valuation of λ and a time step interval solution $\mathcal{I} : \{a_1 \leqslant x \leqslant a_2\}$, there is a transition between S_1 and S_2 if $b \models [\![\lambda]\!]_{a_1}$ (i.e., $\lambda(a_1) \in \gamma(b)$), $b' \models [\![\lambda]\!]_{a_2}$ (i.e., $\lambda(a_2) \in \gamma(b')$), $[\![\lambda]\!]_{a_1} \neq [\![\lambda]\!]_{a_2} \neq 0$ and $\exists x$ such that $[\![\lambda]\!]_x = 0$, with the interpretation function $[\![.]\!] : \mathbb{R}^d \rightarrow \{+, -, 0\}$.

To check for the above condition, we use interval analysis to guarantee that the solution is reliable; the real solutions are enclosed by the computed intervals. Such a guarantee is derived from the fundamental theorem of interval analysis [38].

Practically, building the transitions is based on using constraint solving as a means for refuting invalid transitions. This can be achieved by posing the conditional predicates on a possible transition as a safety property. If the property is unsatisfied, then it is implied that the transition does not exist.

Example 7.
Consider the BJT-based Colpitts oscillator shown in Fig. 10a. Correct functionality ensures that the BJT will never go to the saturation region [39]. In fact, the BJT will either be in the Cut-off mode or Forward active mode. The state space is subdivided into four regions according to the BJT modes of operations (Cut-off, Reverse active, Forward active, and Saturation) with threshold voltage $V_{\mathrm{th}} = 0.75$.

Fig. 10b is a snapshot of the Hsolver code written to represent the abstract states (each corresponding to a BJT mode of operation), the possible transitions between the states, and the property to verify. For instance, the property



a) Circuit diagram

VARIABLES $[V_{C_1}, V_{C_2}, I_L]$
MODES [m1,m2,m3,m4]
STATESPACE
. . .
INITIAL
m1$\{V_{C_1} = 3 \wedge V_{C_2} = 0 \wedge I_L = 0\}$] BJT initially at Forward Active
FLOW
. . .
JUMP
$m1 -> m2\{[V_{C_2} > 0.75 \wedge V_{C_1} + V_{C_2} > 0] \wedge [V'_{C_1} = V_{C_1} \wedge V'_{C_2} = V_{C_2} \wedge I'_L = I_L]\}$
$m2 -> m1\{[V_{C_2} <= 0.75 \wedge V_{C_1} + V_{C_2} > 0] \wedge [V'_{C_1} = V_{C_1} \wedge V'_{C_2} = V_{C_2} \wedge I_L{}' = I_L]\}$
. . .
. . .
$m1 -> m3\{[V_{C_2} < 0.75 \wedge V_{C_1} + V_{C_2} < 0] \wedge [V'_{C_1} = V_{C_1} \wedge V'_{C_2} = V_{C_2} \wedge I'_L = I_L]\}$] Forward Active to Saturation
. . .
UNSAFE
m1$\{VC2 < 0.75 \wedge VC1 + VC2 < 0\}$

b) HSolver code

Fig. 10 BJT Colpitts circuit.

shown ensure that no transition occurs from Forward active (m_1) to Saturation (m_3). It can be validated by proving that $\forall \mathbf{G}\ V_{C_2} < 0.75 \wedge V_{C_1} + V_{C_2} < 0$ is False, where V_{C_1} and V_{C_2} are voltages across the capacitors C_1 and C_2.

VII. Implementation and Experiments

A. Implementation

For experimentation purposes, we used Mathematica's algebraic manipulation and quantified constraint-solving capabilities [12] for the constraint-based verification and for the construction of the abstract model. Conventional model checking on the abstract models is applied using SMV and Hsolver. For instance, the built-in Mathematica function Reduce[expr, vars] simplifies the statement expr by solving equations or inequalities for the state variables vars $= \{v_1, v_2, \ldots, v_m\}$ and eliminating quantifiers. Reduce gives True if the expr is proved to be always true, False if expr is proved to be always false and a reduced expr otherwise. For example, the safety verification problem in Example 4 can be formulated using Reduce as follows: Reduce[Exists[$\{x_1, x_2\}$, $1 - x_1^2 - x_2^2 \geqslant 0 \&\& 1 - x_1^2 + x_2^2 \geqslant 0, -3 + x_1^2 + x_2 \geqslant= 0$], $\{x_1, x_2\}$].

The problem of finding invariants is an important part of the methodology. We need to find Darboux invariants and in the case of reachability verification, we look for invariants bounding the reachable states. Finding invariants is based on the evaluation of the coefficients of the predefined forms of polynomials. In this algorithm, we start with an invariant form with an initial degree and check if such an invariant exists; if not, we increase the degree to form a new polynomial. A bound on the degree must also be specified to ensure termination of the search of the invariants. An arbitrarily assigned bound at the beginning of the algorithm is usually proposed, hence ensuring termination. This is possible using the Mathematica FindInstance function, for example. FindInstance[expr, vars] finds an instance of vars that makes expr True if an instance exists, and gives {} if it does not. The result of FindInstance is of the form $\{\{v_1 \to \text{inst}_1, v_2 \to \text{inst}_2, \ldots, v_m \to \text{inst}_m\}\}$, where inst_i is the provided value. For example, to find the Darboux invariants j we apply FindInstance as follows: FindInstance[ForAll[$\{x, y\}, \mathcal{D}j == \mathcal{K}j$], {coefs}], where j is a polynomial in x, y, with unknown coefficients coefs and K is the cofactor.

B. Experimentation Results

We have applied the verification methodology proposed in this paper to a variety of circuits including Colpitts, Tunnel diode oscillator, and other basic RLC circuits[¶¶].

1. Tunnel Diode Circuit

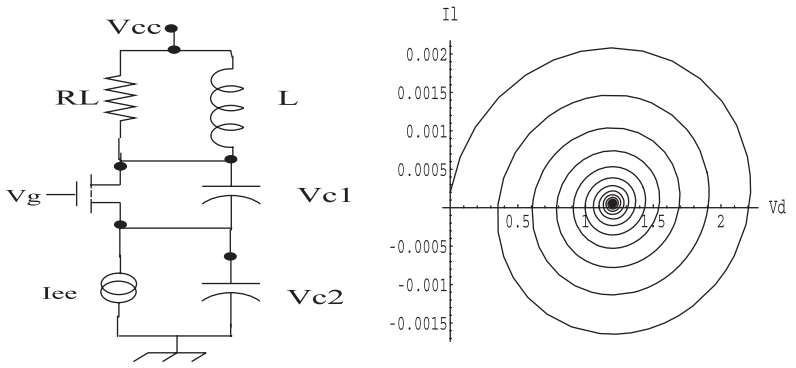
Consider the tunnel diode circuit in Example 2 with the set of parameters $\{C = 1000e^{-12},\ L = 1e^{-6},\ G = 2000e^{-3},\ V_{\text{in}} = 0.3\}$ and the initial values $\{V_C = 0.131V,\ I_L = 0.055A\}$. We are interested to verify the circuit behavior in the region bounded by the constraints $-0.5\,V \leqslant V_C \leqslant 1.2\,V$ and $-0.5\,A \leqslant I_L \leqslant 0.2\,A$ using predicate abstraction.

We verify that the preceding combination of parameters and initial conditions do not produce oscillatory behavior. The behavior in question is stated as the safety property $\mathbf{G}v \leqslant 0.3$. The validation of the property ensures the non-existence of oscillation. We code the circuit equations (determined in Example 2) along with the property in the HSolver language. After verification, the results indicate that the property is satisfied. We can therefore conclude that the chosen parameters do not allow the circuit to oscillate.

2. MOSFET-Based Colpitts Oscillator

The circuit diagram for a MOS transistor-based circuit is shown in Fig. 11a. For the correct choice of component values the circuit will oscillate. This is due to the bias current and negative resistance of the passive tank. The property we analyze is whether for the given parameters and initial conditions the circuit will die out (not oscillate) as shown

[¶¶] More details about experimental studies and the bond graph transformations for the circuits will not be presented here because of space constraints; details of the verification can be found in [34,40].



a) Circuit diagram b) Simulation

Fig. 11 Colpitts circuit.

in Fig. 11b. The extracted equations are described as follows:

$$Vc_1{}' := \frac{1.2 - (Vc_1 + Vc_2)}{R * C} + \frac{Il}{C} - \frac{I_{ds}}{C}, \quad Vc_2{}' := \frac{-Iss}{C} + \frac{1.2 - (Vc_1 + Vc_2)}{R * C} + \frac{Il}{C}, \quad \text{and}$$

$$Il' := \frac{1.2 - (Vc_1 + Vc_2)}{L}$$

with

$$I_{ds} := \begin{cases} 0, & Vc_2 > 0.3 \\ kp * \dfrac{w}{l} * ((0.3 - Vc_2) * (Vc_1 + Vc_2) - 0.5 * (Vc_1 + Vc_2)^2), & Vc_1 + Vc_2 < 0.3 \\ \dfrac{kp}{2} * \dfrac{w}{l} * (0.3 - Vc_2)^2, & Vc_1 + Vc_2 \geqslant 0.3 \end{cases}$$

Oscillation will not occur if the current cannot exceed a certain bound. More precisely, if verified to be true, the property $\forall \mathbf{G} I_l > 0.004 \wedge I_l < 0.004$ implies no oscillation. The system equations, the property of interest along with the required constraints were then translated into the HSolver code. To apply predicate abstraction, the state space is abstracted into three regions (abstract states) because of the different states of the MOSFET transistor within the circuit. The property was verified to be true indicating no oscillation. Fig. 11b represents the circuit simulation for the given parameters, where the current will oscillate until it dies out at $I_l = 0$.

3. Non-linear Analog Circuit

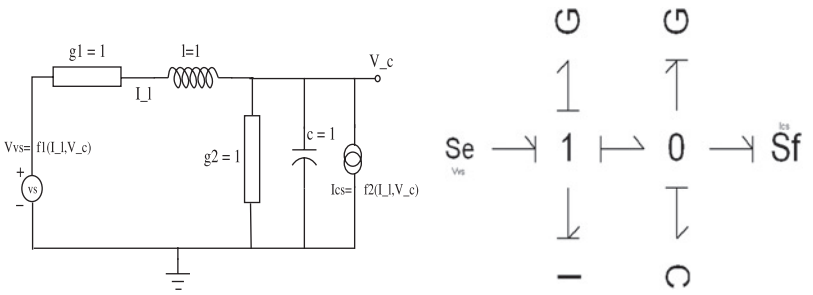
Consider the circuit in Example 3, with initial conditions $x_1(0) \in [-0.7, -1.1]$ and $x_2(0) \in [0.5, 0.9]$. We want to verify the following ∀CTL property on the set of trajectories:

$$\forall \mathbf{F} \mathcal{P} := x_1^2 + x_2 - 3 \geqslant 0$$

which can be understood given the set of initial conditions, on every computation path, in the future the vector field will always cross a threshold condition. We already verified in Example 4 that this cannot happen for the initial conditions inside Region R_1, but with the invariant checking applied, we could not deduce information regarding the behavior in region R_3. After providing the required set of predicates, we only construct corresponding abstract state transition graphs (ASTG) for regions R_1, and R_3. Using the SMV model checker [13], we find, that given the initial conditions, such a property will be indeed satisfied in region R_3.

4. RLC Circuit Oscillator

Checking for occurrence of oscillation is not always possible using predicate abstraction, due to the difficulty of generating an abstract model with no spurious transitions, in some cases we were successful to accomplish the verification.



a) Analog circuit b) Bond Graph

Fig. 12 Non-linear oscillator.

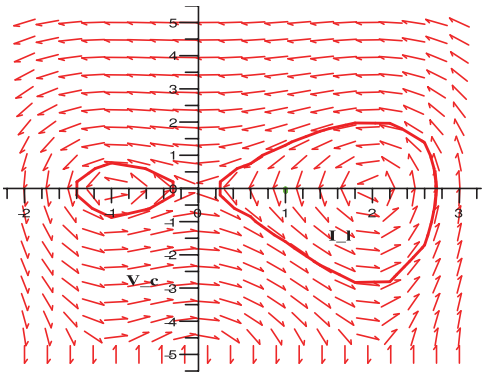


Fig. 13 Phase portrait and invariant regions for circuit in Fig.12(a).

We verified the oscillation property for the circuit shown in Fig. 12a, with a nonlinear voltage source and nonlinear current source *cs* described using ODEs, respectively, as follows:

$$\dot{I}_l = -V_c - \frac{1}{5}V_c^2 \quad \text{and} \quad \dot{V}_c = -2I_l - I_l^2 + I_l^3$$

The equations are extracted from the bond graph shown in Fig. 12b as explained in the methodology. After that, using Mathematica, we generate the following invariants:

$$j_1 = 1 - 5I_l^3 - 15I_l^2 + V_c^3 + \tfrac{15}{2}V_c^2 + \tfrac{15}{4}I_l^4$$

We can therefore construct two invariant regions $R1 := j_1 \leqslant 0$ and $R2 := j_1 > 0$. Given the state space and invariant regions as shown in Fig. 12b, we verify the following ∀CTL property on the set of trajectories:

$$\forall\mathbf{G}(\forall\mathbf{F}(V_c > I_l)) \wedge \forall\mathbf{G}(\forall\mathbf{F}(V_c < I_l))$$

which can be understood, as on every computation path, whenever the capacitor voltage V_c value exceeds the inductor current value I_l, it will eventually decrease below I_l again and vice versa. This property checks for oscillation behavior of the circuit. We constructed the abstract transition graph for each region and verified the property using SMV. We found indeed that the circuit will always oscillate only inside the bounded regions as illustrated in Fig. 13.

VIII. Conclusion

In this paper, we proposed a novel approach for the verification of analog designs. The greatest advantage of our methodology is the lack of an exhaustive simulation that is commonly encountered in the formal verification of analog designs. The major contributions are the following:

- By using bond graphs as a framework to represent circuits, models can be constructed at several levels of abstraction. This can reduce the complexity of the system equations as well as simplify complex behavior.
- A qualitative abstraction approach for the verification of analog properties was proposed using a combination of techniques from predicate abstraction and constraint solving along with model checking.

  We adapted the concept of lazy abstraction for the verification of analog designs. To this aim, we identified a set of basic qualitative predicates (Darboux polynomials) as invariance predicates, which helped avoiding the construction of an abstract model for the whole state space. We proposed a constraint-solving approach for the verification of safety and switching properties. Our method does not require explicit representation of the state space and relies on functions that prove or disprove circuit properties
- When compared to similar research, our verification methodology overcomes the time bound limitations of other exhaustive methods.

Future work includes investigating switched bond graphs for models of mixed signal designs. This will allows us to extend the predicate abstraction to support analog and mixed signal systems. We also plan to explore the verification of more case studies that include in addition the AMS designs, RF, and mechanical components.

References

[1] Xia, S., Divito, B., and Munoz, C., "Toward Automated Test Generation for Engineering Applications," *Proc. IEEE/ACM International Conference on Automated Software Engineering*, 2005, pp. 283–286.

[2] Nadjm-Tehrani, S., and Stromberg, J., "Formal Verification of Dynamic Properties in an Aerospace Application," *Formal Methods in System Design*, Vol. 14, No. 2, 1999, pp. 135–169.

[3] Cousot, P., "Proving the Absence of Run-Time Errors in Safety-Critical Avionics Code," *Proc. IEEE/ACM Conference on Embedded Software*, 2007, p. 79.

[4] Vlach, J., and Singhal, K., *Computer Methods for Circuit Analysis and Design*, Kluwer, Norwell, MA, 2003.

[5] Cellier, F. E., Clauss, C., and Urquia, A., "Electronic Circuit Modelling and Simulation in Modelica," *Proc. Eurosim Congress on Modelling and Simulation*, Vol. 2, 2007, pp. 1–10.

[6] Maehne, T., and Vachoux, A., "Proposal for a Bond Graph based Model of Computation in SystemC-AMS," *Proc. Languages for Formal Specification and Verification, Forum on Specification & Design Languages*, 2007.

[7] Cousot, P., and Cousot, R., "Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints," *Proc. ACM Principles of Programming Languages*, 1977, pp. 238–252.

[8] Graf, S., and Saidi, H., "Construction of Abstract State Graphs with PVS," *Computer Aided Verification*, LNCS 1254, Springer, Berlin, 1997, pp. 72–83.

[9] Henzinger, T., Jhala, R., Majumdar, R., and Sutre, G., "Lazy Abstraction," *Proc. ACM Principles of Programming Languages*, 2002, pp. 58–70.

[10] Elmqvist, H., "Dymola – Dynamic Modelling Language," *User's Manual*, Dynasim, 1994, http://www.dynasim.se.

[11] Ratschan, S., and She, Z., "Safety Verification of Hybrid Systems by Constraint Propagation Based Abstraction Refinement," *Hybrid System: Computation and Control*, LNCS 3414, Springer, Berlin, 2005, pp. 573–589.

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

[13] Clarke, E., Grumberg, O., and Peled, D. A., *Model Checking*, MIT Press, Cambridge, MA, 1999.

[14] Granda, J., and Montgomery, A., "Automated Modelling and Simulation Using the Bond Graph Method for the Aerospace Industry," *Proc. AIAA modelling and Simulation Technologies Conference and Exhibit*, 2003.

[15] Stromberg, J.-E., Nadjm-Tehrani, S., and Top, J., "Switched Bond Graphs as Front-end to Formal Verification of Hybrid Systems," *Proc. of Verification and Control of Hybrid Systems*, LNCS 1066, Springer, Berlin, 1996, pp. 282–293.

[16] Kurshan, R. P., and McMillan, K. L., "Analysis of Digital Circuits through Symbolic Reduction," *IEEE Transactions on Computer-Aided Design*, Vol. 10, 1991, pp. 1350–1371.

[17] Greenstreet, M. R., and Mitchell, I., "Reachability Analysis Using Polygonal Projections," *Hybrid System: Computation and Control*, LNCS 1569, Springer, Berlin, 1999, pp. 103–116.

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

[19] Gupta, S., Krogh, B. H., and Rutenbar, R. A., "Towards Formal Verification of Analog Designs," *Proc. IEEE/ACM Conference on Computer Aided Design*, 2004, pp. 210–217.
[20] Frehse, G., Krogh, B. H., and Rutenbar, R. A., "Verifying Analog Oscillator Circuits Using Forward/Backward Abstraction Refinement," *Proc. IEEE/ACM Design, Automation and Test in Europe*, 2006, pp. 257–262.
[21] Hartong, W., Klausen, R., and Hedrich, L., "Formal Verification for Nonlinear Analog Systems: Approaches to Model and Equivalence Checking," *Advanced Formal Verification*, Kluwer, Norwell, MA, 2004, pp. 205–245.
[22] Zaki, M., Al Sammane, G., Tahar, S., and Bois, G., "Combining Symbolic Simulation and Interval Arithmetic for the Verification of AMS Designs," *Proc. IEEE International Conference on Formal Methods in Computer-Aided Design*, 2007, pp. 207–215.
[23] Zaki, M., Tahar, S., and Bois, G., "Formal Verification of Analog and Mixed Signal Designs: A Survey," *Microelectronics Journal*, Vol. 39, No. 12, Elsevier B.V. Pub., pp. 1395–1404.
[24] Alur, R., Dang, T., and Ivancic, F., "Reachability Analysis via Predicate Abstraction," *Hybrid Systems: Computation and Control*, LNCS 2289, Springer, Berlin, 2002, pp. 35–48.
[25] Clarke, E., Fehnker, A., Han, Z., Krogh, B. H., Stursberg, O., and Theobald, M., "Verification of Hybrid Systems based on Counterexample-Guided Abstraction Refinement," *Tools and Algorithms for the Construction and Analysis of Systems*, LNCS 2619, Springer, Berlin, 2003, pp. 192–207.
[26] Tiwari, A., "Abstractions for hybrid systems," *Formal Methods in System Design*, Vol. 32, No. 1, 2008, pp. 57–83.
[27] Sankaranarayanan, S., Sipma, H., and Manna, Z., "Constructing Invariants for Hybrid Systems," *Hybrid Systems: Computation and Control*, LNCS 2993, Springer, Berlin, 2004, pp. 539–554.
[28] Prajna, S., and Jadbabaie, A., "Safety Verification of Hybrid Systems Using Barrier Certificates," *Hybrid Systems: Computation and Control*, Springer, Berlin, 2004, pp. 477–492.
[29] Paynter, H. M., *Analysis and Design of Engineering Systems*, The MIT Press, 1961.
[30] Karnopp, D., and Rosenberg, R., *System Dynamics: A Unified Approach*, Wiley, New York, 1975.
[31] Broenink, F., "Introduction to Physical Systems Modelling with Bond Graphs," Technical Report, Simulation in Europe (SiE) Working Group, 1999.
[32] Cellier, F. E., and Nebot, A., "The Modelica Bond Graph Library," *Proc. of the Modelica Conference*, 2005, pp. 57–65.
[33] Goriely, A., "Integrability and Nonintegrability of Ordinary Differential Equations," *Advanced Series on Nonlinear Dynamics*, Vol. 19, World Scientific, Singapore, 2001.
[34] Zaki, M. H., "Techniques for the Formal Verification of Analog and Mixed-Signal Designs", Ph.D. Thesis, Department of Electrical and Computer Engineering, Concordia University, September 2008.
[35] Furi, M., and Martelli, M., "A Multidimensional Version of Rolle's Theorem," *The American Mathematical Soc*iety, Vol. 102, No. 3, 1995, pp. 243–249.
[36] Ratschan, S., "Continuous First-Order Constraint Satisfaction," *Artificial Intelligence, Automated Reasoning, and Symbolic Computation*, LNCS 2385, Springer, Berlin, 2002, pp. 181–195.
[37] Granvilliers, L., "On the Combination of Interval Constraint Solvers," *Reliable Computing*, Vol. 7, No. 6, 2001, pp. 467–483.
[38] Moore, R. E., "Methods and Applications of Interval Analysis," Society for Industrial and Applied Mathematics, Philadelphia, PA, 1979.
[39] Kennedy, M. P., "Chaos in the Colpitts Oscillator," *IEEE Transactions on Circuits and Systems*, Vol. 1, No. 41, 1994, pp. 771–774.
[40] Denman, W., Zaki, M., and Tahar, S., "Analog Formal Verification via Bond Graphs and Constraint Solving," Technical Report, ECE Department, Concordia University, Montreal, Quebec, Canada, April 2008, http://www.hvg.ece.concordia.ca/Publications/TECH REP/AMS BG TR08

Michael Hinchey
Associate Editor