Abstraction Based Verification of Analog Circuits Using Computer Algebra and Constraint Solving

Mohamed H. Zaki[1] Sofiène Tahar[1] Guy Bois[2]

***Abstract*—Formal methods have been advocated for the verification of digital designs where correctness is proved mathematically. Unlike its digital counterpart, analog and mixed-signal (AMS) systems verification is a challenging and exhaustive task that requires lots of expertise and deep understanding of their behavior. In this paper, we propose an abstraction methodology for the verification of safety properties of a class of non linear analog circuits described by polynomial differential equations. The method is based on combining techniques from constraint solving and computer algebra. For illustration purposes, we have used the computer algebra system Maple and the constraint solvers RealPaver and Rsolver to analyze the behavior of a non linear oscillator circuit.**

I. Introduction

Formal methods like model checking have been advocated for the verification of digital designs where their correctness is proved mathematically against some properties, i.e., establishing whether the design meets the specifications. Unlike its digital counterpart, analog and mixed-signal (AMS) systems verification is a challenging and exhaustive task that requires lots of expertise and deep understanding of their behavior. Traditionally, simulation based techniques were complemented by symbolic techniques where the effect of parameters variations on the system behavior is analyzed. Although successful, challenging problems (like non linear effects) make these techniques only suitable for simple designs. Researchers started lately studying the applicability of formal methods for the verification of AMS systems. A typical verification task is to prove that a circuit behaves correctly for all possible input signals and initial conditions and that none of them drives the system into a bad state. Among the most important properties are those of safety and reachability which are computationally hard even for the simplest AMS systems. Hence, a direct application of model checking on continuous systems is very difficult and abstraction techniques are required in order to achieve this task. Among them predicate abstraction, first developed in [15], is a technique where the state space is divided into a finite set of regions and a set of rules is used to build the transition between the reachable states such that the generated state transition system can be verified using model checking.

This paper describes a predicate abstraction methodology to extract finite state models from non linear analog circuits represented by polynomial differential equations by means of computer algebra and constraint solving. Conventional model checking algorithms can then be applied to verify safety properties on the generated models. Satisfaction of the property in the finite state model guarantees its satisfaction in the analog circuit-level model over a continuous range of input conditions. For illustration purposes, we have used the computer algebra system Maple [17] and the constraint solvers RealPaver [11] and Rsolver [19] to analyze the behavior of a non linear oscillator circuit.

The rest of the paper is organized as follows: We start by discussing relevant related work in Section II, followed by an overview of the verification methodology in Section III. We proceed in Section IV by defining the analog system model and explaining predicate abstraction, then we introduce invariant theory in Section V. Abstract model construction is developed in Section VI with illustrative example shown in Section VII, before concluding with a discussion in Section VIII.

II. Related Work

AMS formal verification: Several methods for approximating reachable sets for continuous dynamics have been proposed for the verification of AMS systems. These methods rely on the discretization of the continuous state space and were pursued in [16] and [12]. For instance, in [16], the authors tried to construct a finite-state discrete abstraction of an electronic circuits by partitioning the continuous state space representing the characteristics of transistors into hypercubes using a fixed grid and computed the reachability relations between these cubes using optimization techniques. In [12], the authors tried to overcome the expensive computational method in [16], by using discretization and projection techniques of the state space, reducing the dimension of the state space. While the approach is less precise due to the use of projection techniques, it is still sound. Variant approaches of polyhedral based analysis were adapted in [8] and [13]. For instance, the model checking tools d/dt [8] and Checkmate [13] were used in the verification of a biquad low-pass filter [8], and tunnel diode oscillator and $\Delta - \Sigma$ modulator [13], respectively. Other computational techniques for AMS verification can be found in [10]. For an overview and comparison of the different projects related to the subject, refer to [26].

Predicates abstraction [15] was used for the verification of hybrid systems in [1], [5] and [22]. In [1], the authors combined predicate abstraction with convex polyhedral analysis for the verification of reachability properties of linear hybrid systems. The authors of [22] proposed a way to extract predicates from polynomial differential

[1] Dept. of Electrical & Computer Engineering, Concordia University, 1455 de Maisonneuve W., Montreal, Quebec, H3G 1M8, Canada, Email: {mzaki,tahar}@ece.concordia.ca.

[2] Genie Informatique, Ecole Polytechnique de Montreal, C.P. 6079, succ. Centre-Ville, Montreal Quebec, H3C 3A7 , Canada. Email: guy.bois@polymtl.ca

equations by looking at higher derivatives. They used qualitative based reasoning techniques to describe the transition between abstract states. In order to enhance their methodology, the authors in [23], proposed the use of computer algebra techniques to generate invariance. In [21], the authors presented an approach for generating polynomial invariants for hybrid systems with general polynomial dynamics where the invariant generation is turned into a constraint solving problem on the coefficients of a template polynomial. In the present paper, we followed the general idea of this work, but we propose different abstraction techniques. In the above mentioned works, the invariants generation problem is analog to the first integrability problem which is usually very difficult to handle while in our work we are only interested in second (Darboux) integrals polynomial invariants which is simpler to find using computer algebra algorithms and contains sufficient information for the analysis. In addition, we are only interested in invariants which can be qualitatively extracted from the systems automatically providing useful information about the behavior, avoiding the generation of redundant or hard to interpret invariants. In addition, the generated invariants are independent of the initial conditions making them more general for the system analysis.

III. Verification Methodology

Figure 1 shows the methodology we propose for the verification of non-linear analog circuits. Based on nodal analysis techniques, a system of non-linear ordinary polynomial differential equations (ODEs) is set up for the circuit including the state variables representing the energy storing elements (voltages at capacitances, currents through inductors). We use the set of ODEs along with the circuit properties described in computational temporal logic (∀CTL) [6], to construct a finite abstract model of the circuit which can be verified using model checking. The abstract model is constructed in successive steps. We start by identifying the invariant regions for the system of equations using qualitative analysis based on Darboux theory of integrability [14]. Qualitative and specification based predicates are then provided such that for each invariant region along with the set of associated predicates, we construct the abstract state transition graph using constraint solving techniques. Abstract states are formed by conjunction of predicates and transition between the states are constructed using a set of mathematical rules.

The novelty in this paper comes from using Darboux polynomials as invariance predicates which helps avoid the generation of an abstraction model for the whole state space, while guaranteeing soundness of the abstraction method.

IV. Preliminaries

A. System Description

We consider analog systems which can be modelled by non linear polynomial ordinary differential equations (ODEs). An *analog system* is a tuple $CS = (\mathcal{X}, \mathcal{X}_0, \mathcal{P})$ with $\mathcal{X} \subseteq \mathbb{R}^d$ is the continuous state space, $\mathcal{X}_0 \subseteq \mathcal{X}$ is the set of initial states and $\mathcal{P} : \mathcal{X} \to \mathbb{R}^d$ is the continuous vector field. The behavior of the system is governed by polynomial differential equations of the form:

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

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

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

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 an analog system $CS = (\mathcal{X}, \mathcal{X}_0, \mathcal{P})$ over a continuous time period (an interval) $T_c = [\tau_0, \tau_1] \subseteq \mathbb{R}^+$ can be described as 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 $t \in T_c$, is a time point.

B. Predicate Abstraction

In the abstraction method, we start first by defining the abstract states and the maps from concrete to abstract states. An abstract transition system is then created by constructing the abstract initial states and abstract transition relations. In order to fulfill these steps a sound relationship between the concrete and abstract domain should be defined. Predicate abstraction [1] is a method where the set of abstract states is encoded by a set of boolean variables representing each a concrete predicate.

Based on [1], we define a discrete abstraction of the system with respect to a given n-dimensional vector of predicates $\Psi = (\psi_1, \ldots, \psi_n) \in (\Xi)^n$, where Ξ is the set of polynomial predicates $\psi : \mathbb{R}^d \to \mathbb{B}$, with $\mathbb{B} = \{0, 1\}$. A polynomial predicate is of the form:

$$\psi(x) := \mathcal{P}_k(x_1, \ldots, x_d) \sim 0$$

Where $\sim \in \{<, >\}$. 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.

An abstract state transition system (ASTG) is a tuple $T_\Psi = (Q_\Psi, \rightarrow', Q_{\Psi,0})$, where:

- $Q_\Psi \subset L \times \mathbb{B}^n$ is the abstract state space for a n-dimensional vector predicates, where an abstract state is defined as a tuple (l, b), with $l \in L$ is a label and $b \in \mathbb{B}^n$.
- $\rightarrow' \subseteq Q_\Psi \times Q_\Psi$ is a relation capturing abstract transition such that:

$$\{b \rightarrow' b' | \exists x \in \Upsilon_\Psi(b), t \in \mathbb{R}^+ : x' = \Phi_x(t) \in \Upsilon_\Psi(b') \wedge x \rightarrow x'\}$$

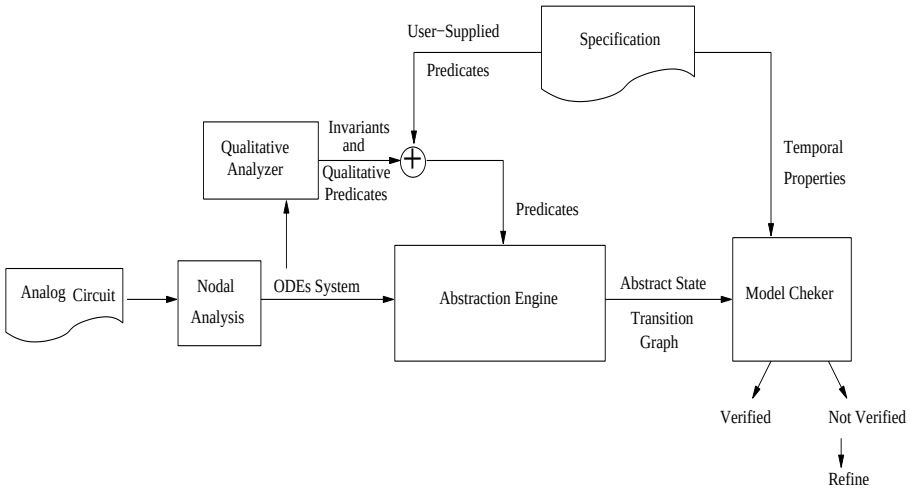


Fig. 1. Verification Methodology for Analog Systems

Where the concretization function: $\Upsilon_\Psi : \mathbb{B}^n \rightarrow 2^{\mathbb{R}^d}$ 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,

V. State Space Invariants

Usually, an abstract state transition system representing an over-approximation of the set of reachable states is sufficient for the verification of different properties of the system such as safety and invariant properties. However, finding invariant properties or invariant predicates is itself a very difficult task and have been studied in the context of programming languages [3] and hybrid systems [23] [21]. The main principle for proving that a predicate $\mathcal{I}$ is an invariant of some system is to prove that every initial state s satisfies the predicate and such satisfaction is preserved under all transitions [23] [21]. A function $\mathcal{I}$ is called an invariant of a system CS such that $\mathcal{P}(\mathbf{x}(0)) = s \models \mathcal{I},\ \mathcal{P}(\mathbf{x}(\varsigma)) = \mathbf{s}_\varsigma \models \mathcal{I}$ *and* $\forall t \in [0,\varsigma], \mathcal{P}(\mathbf{x}(t)) \models \mathcal{I}$.

In the context of continuous systems, usually, the system qualitative behavior varies in the different invariant regions of the state space. While in general, it is very difficult to identify such regions, successful algorithms have been developed recently based on the Darboux integrability theory to deal with this problem for the class of polynomial dynamical systems. Darboux polynomials $\mathcal{J}_i$ (also known as second integrals, invariant polynomials) have been investigated in the qualitative and algebraic analysis of continuous systems. Darboux polynomials split the phase portrait into regions where the behavior is qualitatively different (bound or unbound). These functions $\mathcal{J}_i$ provide the essential skeleton for the phase space from which all other behaviors can be qualitatively determined. They are going to be the *bricks* with which we will build the invariant state space by constructing invariant regions used in the abstraction methodology [14].

Given the system of ODEs $\frac{dx_k}{dt} = \mathcal{P}_k(x_1(t), \ldots, x_d(t))$, with $k = 1, \ldots d$ ($\frac{d\mathbf{x}}{dt} = \mathbf{P}(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^d$ and $\mathbf{P} = (\mathcal{P}_1, \ldots, \mathcal{P}_d)$), we can define a 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 $\mathbf{x}$: $\mathcal{G} : \mathbb{R}^k \rightarrow \mathbb{R}$, then

$$\frac{d\mathcal{G}}{dt} := \dot{\mathcal{G}} = \mathcal{D}_\mathbf{P}(\mathcal{G}) = \mathbf{P}.\partial_\mathbf{x}\mathcal{G}$$

A Darboux polynomial is of the form $\mathcal{J}(\mathbf{x}) = 0$, when $D\mathcal{J} = \mathcal{K}\mathcal{J}$, with $\mathcal{J} \in \mathbb{R}[\mathbf{x}]$ and $\mathcal{K} = \mathcal{K}(\mathbf{x})$ is a polynomial called the cofactor of $\mathcal{J} = 0$, with a degree of at most $\mathcal{M} - 1$.

The problem of finding the invariants is based on the evaluation of the coefficients of the predefined forms of Darboux polynomials $\mathcal{J}$ and their cofactors $\mathcal{K}$ by solving the algebraic system obtained identifying to zero the coefficients of the polynomials in state variables. In order to satisfy the entire system of the algebraic equations, it is necessary then to introduce the conditions on the parameters of the differential system.

Example V.1: Using MAPLE PSsolver [9], we extract the invariants for the system [14]:

$$\dot{x} = 3(x^2 - 4) \ and \ \dot{y} = 3 + xy - y^2$$

and get three corresponding invariant polynomials

$$j_1 = x^2 - 4,\ j_2 = y^4 - 6y^2 - 4xy - 3,$$

$$j_3 = y^4 + 2xy^3 + 6y^2 + 2xy + x^2 - 3$$

Given the set of invariant polynomials, we can define a set of predicates which can be used to construct the invariant regions. So in the context of abstraction, invariant regions can be considered as abstract states and we obtain the following theorem.

Theorem V.1: Let $\mathcal{V} = \{x \in \mathbb{R}^k | x \models \Gamma\}$ be an invariant region, where Γ is a conjunction of invariant predicates. If $\mathbf{x}(0)$ is an initial state, then

$$\mathcal{V} = \mathcal{V}(\mathbf{x}(0))$$

denotes an over-approximation of the set of states reachable from $\mathbf{x}(0)$.

Being inside an invariant regions means that the system dynamics will always stay in this region. In order to enhance the precision of the generated model, each invariant region can be subdivided into regions based on predicate abstraction and hence an abstract state transition graph can be constructed for each invariant region.

VI. Abstract Model Generation

In this section, we will describe how to generate the abstract state transition model by constructing the abstract state space and identifying transitions between the abstract states.

A. Constructing Abstract States Using Constraint Solving

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. If a value is a feasible solution, this value is declared consistent with respect to the constraint. We say a constraint solving problem is consistent if and only if it has at least one solution otherwise it is inconsistent.

Two categories of constraint solvers are identified [25]:

- 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 [19].
- 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 unfeasible. Realpaver [11] is an example of this category.

In the remaining of this section, we will show how both types of solvers can be used in the construction of the state space.

User-Provided Predicates: One important issue arise in the methodology presented in this paper is to associate with each invariant region, the corresponding set of user-provided predicates dividing the region into abstract states. Let $\{\mathcal{T}_k\} \in 2^{Pre}$, $k = 1, \ldots, n$, where Pre is the set of available predicates. $\{\mathcal{T}_k\}$ is the set of predicates associated with invariant region $\mathcal{V}_k$, then

$$\mathcal{T}_k = \{p \in Pre | \exists x \in \mathcal{V}_k \; such \; that \; p(x) \; is \; True \; \}$$

This can be achieved using unsatisfiability solvers which can be used for refutation of some abstract state by predicates elimination. RealPaver is able to solve nonlinear equations or inequality constraints over real numbers, where each domain is represented by a closed interval. Given a system of constraints, RealPaver computes a union of boxes that contains all the solutions satisfying these constraints, if no box is computed by RealPaver, then this system is guaranteed to has no solution.

Qualitative Predicates: In addition to user provided predicates, we propose using qualitative properties of the system of ODEs to extract useful predicates. More specifically, we propose using *isoclines* as predicates.

Isoclines are set of points where trajectories have the same slope, hence they divide the state space into regions with different behaviors. Finding different isocline predicates within an invariant region can be achieved by solving constraints on the parameters of predefined form of isocline predicates. Any consistent solution define an isocline predicate. To achieve our goal, we use the satisfiability constraint solver Rsolver. Rsolver uses constraint satisfaction techniques for solving first order system of equality and inequalities over real domains by finding intervals that enclose the solutions.

Example VI.1: Given the system:

$$\dot{x_1} = x_1^2 \; and \; \dot{x_2} = x_2(2x_1 - x_2)$$

Suppose we are looking for linear isoclines of the form

$$x_2 = ax_1 + b$$

with $b = 0$ for $x > 0$. We set

$$\frac{dx_2}{dx_1} = \frac{x_2(2x_1 - x_2)}{x_1^2} = c$$

Where $c \in \mathbb{R}$. Using Rsolver, we can define this problem using quantified constraints as

$$\forall x \in [0, \hat{x}], [cx^2 - ax(2x - ax) = 0]$$

Where $\hat{x}$ is a predefined upper bound.

B. Computing Abstract Transitions

One main issue in constructing abstract state transition systems is identification of possible transitions between the states. In order to achieve this, usually information from the solution of the ODEs is required to describe transitions between abstract states (s_i). In this section, we show that transitions can be identified using ideas inspired from qualitative reasoning [22].

Given a set of predicates $P = \{p1, ..., p_n\}$, we state a set of rules to identify the set of possible transitions T between abstract states in the following way:

- $\exists t := (s_1, s_2) \in T$ such that $s_0 \models (p_1 = 0 \wedge p_2 > 0)$ $\wedge s_2 \models (p_1 > 0 \wedge p_2 > 0)$ $\wedge$ $\exists s_3$ such that $s_3 \models (p_2 = 0 \wedge p_1 > 0)$
- $\exists t := (s_1, s_2) \in T$ such that $s_0 \models (p_1 = 0 \wedge p_2 > 0)$ $\wedge s_2 \models (p_1 < 0 \wedge p_2 > 0)$ $\wedge$ $\exists s_3$ such that $s_3 \models (p_2 = 0 \wedge p_1 < 0)$
- $\exists t := (s_1, s_2) \in T$ such that $s_0 \models (p_1 = 0 \wedge p_2 < 0)$ $\wedge s_2 \models (p_1 > 0 \wedge p_2 < 0)$ $\wedge$ $\exists s_3$ such that $s_3 \models (p_2 = 0 \wedge p_1 > 0)$

- $\exists t := (s_1, s_2) \in T$ such that $s_0 \models (p_1 = 0 \wedge p_2 < 0) \wedge s_2 \models (p_1 < 0 \wedge p_2 < 0) \wedge \exists s_3$ such that $s_3 \models (p_2 = 0 \wedge p_1 < 0)$

The above rules give an over-approximation of the transition system as no information about the vector field direction is used. In order to remove such redundant transitions in region, we can use extended mean value theorem [20] as a way to identify flow direction.

VII. Application

Consider the non linear circuit shown in Figure 2, where the non linearity comes from the voltage controlled current source i_2. This circuit exhibits an oscillatory behavior for specific initial capacitors' voltages. We use the methodology described in this paper to show that such property indeed exists. We start by extracting, using nodal analysis techniques, the systems equations which are stated as follows:

$$\dot{x_1} = x_2 \ and \ \dot{x_2} = -x_1 + x_1^3$$

Where the state variables x_1 and x_2 represent the voltages across capacitors c_1 and c_2, respectively.

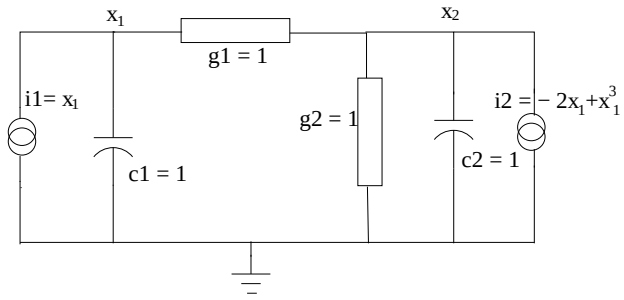


Fig. 2. Oscillating Circuit Example

Using MAPLE PSsolver [9] we identify the corresponding invariants:

$$j_1 = 1 + \sqrt{2}x_2 - x_1^2 \ and \ j_2 = 1 - \sqrt{2}x_2 - x_1^2$$

We form four invariant regions:

$$R_1 = j_1 > 0 \wedge j_2 > 0 \quad R_2 = j_1 > 0 \wedge j_2 < 0$$

and

$$R_3 = j_1 < 0 \wedge j_2 > 0 \quad R_4 = j_1 < 0 \wedge j_2 < 0$$

Suppose for the initial capacitors' voltages

$$x_1 \in [0, 01] \ and \ x_2 \in [0.4, 0.8]$$

We want to verify the following ACTL property on the set of trajectories:

$$\forall\Box(\forall\Diamond(x_2 > x_1)) \ \wedge \ \forall\Box(\forall\Diamond(x_2 < x_1))$$

which can be understood as on every computation path, whenever the voltage x_2 will exceed voltage x_1, it will eventually decrease below x_1 again and vise-versa. This property checks for oscillation behavior of the circuit. We supply the user-predicates

$$p_1 := x_2 - 0.5 \sim 0, \ p_2 := x_2 + 0.5 \sim 0$$

and

$$p_3 := x_1 - x_2 \sim 0, \ p_4 := x_1 \sim 0, \ p_5 := x_2 \sim 0$$

along with the isocline predicates

$$p_6 := -x_1 + x_1^3 - x_2 \sim 0, \ p_7 := -x_1 + x_1^3 + x_2 \sim 0$$

and

$$p_8 := -x_1 + x_1^3 - \frac{2}{5}x_2 \sim 0, \ p_9 := -x_1 + x_1^3 + \frac{2}{5}x_2 \sim 0$$

with $\sim \in \{<, >\}$, (see Figure 3). Based on initial conditions (initial voltages values), we only construct corresponding abstract state transition graphs (ASTG) for regions R_1, R_2. The ASTG for region R_1 is shown in Figure 4. By applying model checking (SMV [7] model checker in this paper), we find that the circuit is oscillating only for region R_1; ACTL property is satisfied, while the property is violated in region R_2, hence the system is not oscillating in this region.

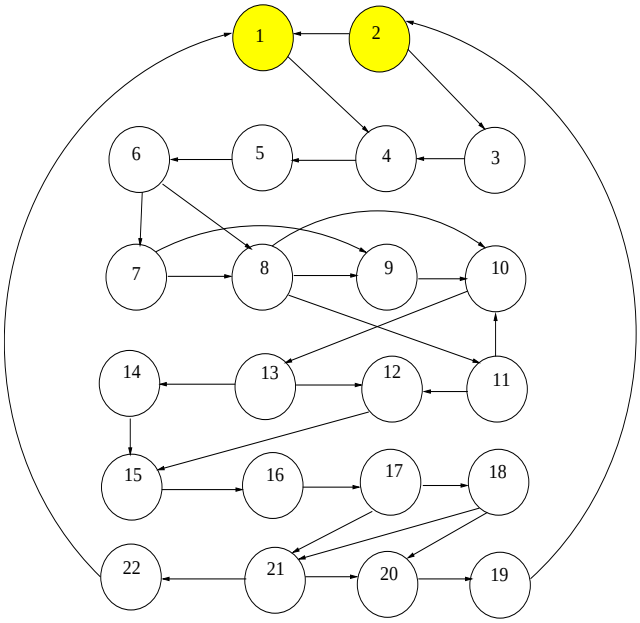
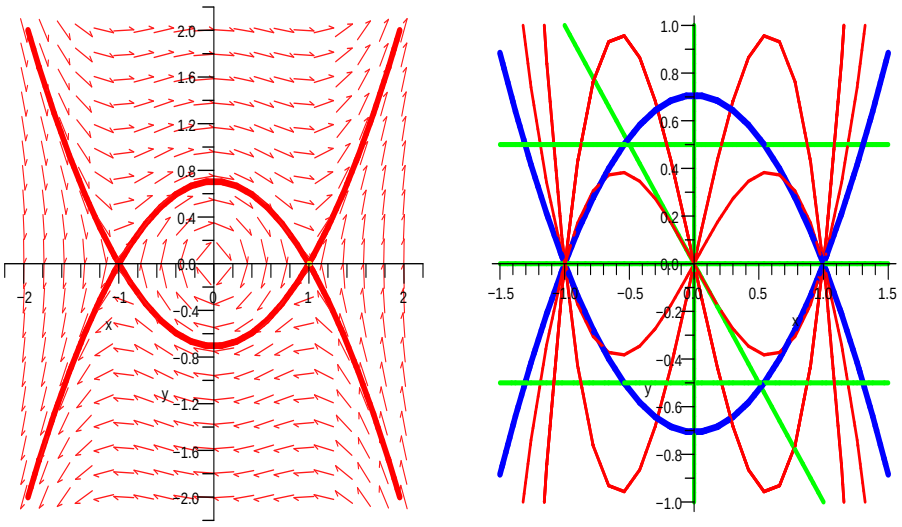


Fig. 4. ASTG of Region R_1 for the Circuit in Figure 2

VIII. Conclusion

The lack of methods for computing reachable sets of continuous dynamics has been the main obstacle towards an algorithmic verification methodology for continuous systems. Unlike conventional approaches which attempt to find exact solutions and are thus limited by undecidability of most non-trivial systems, we present a practical verification approach which is based on an efficient method for abstracting the continuous behavior using a combination of techniques from predicate abstraction and invariance



(a). Darboux Invariants (b). Abstract State Space

Fig. 3. Phase Portrait of Circuit in Figure 2

generation along with model checking to prove properties about the systems. One advantage of our methodology is the ability to avoid generating an abstraction for the whole state space, meanwhile the soundness of the abstraction is guaranteed. We have used packages in the computer algebra system Maple along with constraint solving tools to construct the abstract transition graph of continuous and analog systems. Future work includes extending the predicate abstraction to support hybrid systems, explore more complex case studies especially, we are interested in the verification of switched-capacitors based designs.

References

[1] R. Alur, T. Dang, F. Ivancic. Reachability Analysis Via Predicate Abstraction. In Hybrid Systems: Computation and Control, LNCS 2289, pp. 35-48. Springer-Verlag, 2002.

[2] R. Alur, T. Henzinger, G. Lafferriere, G. J. Pappas: Discrete abstractions of hybrid systems, Proccedings of the IEEE, 88(2):971-984, 2000.

[3] S. Bensalem, S. Graf, Y. Lakhnech: Abstraction as the Key for Invariant Verification. International Symposium on Verification celebrating Zohar Manna's 64th Birthday, LNCS 2772, pp. 67-99, Springer-Verlag, 2003.

[4] 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.

[5] E. Clarke, A. Fehnker, Z. Han, B. Krogh, J. Ouaknine, O. Stursberg, and M. Theobald. Abstraction and Counterexample-guided Refinement in Model Checking of Hybrid Systems. International Journal of Foundations of Computer Science, 14(4), pp. 583-604, World Scientific, 2003.

[6] E. Clarke, O. Grumberg, D. E. Long. Model checking and abstraction. ACM Symposium on Principles of Programming Languages, pp. 343 -354, 1992.

[7] E. Clarke, O. Grumberg , D.A. Peled. Model Checking. MIT Press: Cambridge, MA, 1999.

[8] T. Dang and A. Donze : Verification of Analog and Mixed-Signal Circuits Using Hybrid System Techniques. In Formal Methods in Computer-Aided Design, LNCS 3312, Springer, pp. 14–17, 2004.

[9] L.G. Duarte, S.E. Duarte, L.A. da Mota, J.E. Skea: An Extension of the Prelle-Singer Method and a Maple Implementation. Computer Physics Communications, Elsevier, 144:46-62, 2002.

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

[11] L. Granvilliers. On the Combination of Interval Constraint Solvers. Reliable Computing, 7(6):467-483, 2001

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

[13] S. Gupta, B. Krogh and R. Rutenbar, Towards formal verification of analog designs, IEEE/ACM International Conference on Computer Aided Design, pp. 210 - 217, 2004.

[14] A. Goriely. Integrability and Nonintegrability of Ordinary Differential Equations, Advanced Series on Nonlinear Dynamics, Vol 19 World Scientific 2001.

[15] S. Graf and H. Saidi. Construction of abstract state graphs with PVS. In Computer Aided Verification, LNCS 1254, Springer, pp. 72-83, 1997.

[16] R.P. Kurshan and K.L. McMillan. Analysis of digital circuits through symbolic reduction. IEEE Trans. on Computer-Aided Design 10:1350-1371, 1991.

[17] MapleSoft inc., www.maplesoft.com/

[18] M Prelle and M Singer, Elementary first integral of differential equations. Transactions of the American Mathematical Society, pp. 279-215 (1983).

[19] S. Ratschan. Continuous First-Order Constraint Satisfaction. In Artificial Intelligence, Automated Reasoning, and Symbolic Computation, LNCS 2385, pp. 181-195, Springer, 2002

[20] S. Ratschan, Z. She: Safety Verification of Hybrid Systems by Constraint Propagation Based Abstraction Refinement. In Hybrid Systems: Computation and Control, LNCS 3414, pp.573-589, Springer, 2005.

[21] S. Sankaranarayanan, H. Sipma, Z. Manna. Constructing Invariants for Hybrid Systems. In Hybrid Systems: Computation and Control, LNCS 2993, pp 539-554, Springer, 2004.

[22] A. Tiwari and G. Khanna. Series of abstractions for hybrid automata. In Hybrid Systems: Computation and Control, LNCS 2289, pp. 465-478, Springer, 2002.

[23] A. Tiwari and G. Khanna. Non-linear Systems: Approximating Reach Sets. In Hybrid Systems: Computation and Control, LNCS 2993, pp. 600-614. Springer-Verlag, 2004.

[24] J. Vlach and K. Singhal. Computer Methods for Circuit Analysis and Design. Van Nostrand Reinhold, New York, 1994.

[25] S. Xia, B. Divito, C. Munoz, Toward Automated Test Generation for Engineering Applications, IEEE/ACM International Conference on Automated Software Engineering, pp. 283-286, 2005

[26] 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.