On the Formal Analysis of Analog Systems using Interval Abstraction

Mohamed H. Zaki, Ali Habibi, Sofiène Tahar and Guy Bois §

Dept. of Electrical & Computer Engineering, Concordia University
1455 de Maisonneuve W., Montreal, Quebec, H3G 1M8, Canada
§Genie Informatique, Ecole Polytechnique de Montreal
Pavillon Decelles, 5255 Avenue Decelles, Montreal, QC H3T 2B1, Canada
{mzaki,habibi,tahar}@ece.concordia.ca, guy.bois@polymtl.ca

Abstract. Formal methods have been advocated for the verification of digital design where correctness is proved mathematically. In contrast to digital designs, the verification of analog and mixed signal (AMS) systems is a challenging task that requires lots of expertise and deep understanding of their behavior. In this paper, we intend to present an ongoing project for developing an approach for the analysis of analog systems using formal methods. The proposed method is based on the computation of finite-state conservative abstraction of the design. For this purpose, we use abstract interpretation, which is a theoretical framework for the analysis of infinite state systems and in this paper we choose interval arithmetics as the basis of the abstraction. To test and validate our methodology, we have used interval arithmetic tool AWA to describe the behavior of tunnel diode oscillator.

1 Introduction

Formal methods have been advocated for the verification of digital design where correctness is proved mathematically. In contrast to digital designs, the verification of analog and mixed signal (AMS) systems is a challenging task that requires lots of expertise and deep understanding of their behavior. The functionality of analog circuits is defined directly in terms of continuous electrical quantities and is usually sensitive to environment factors like signal noise, current leakage, temperature, etc, in addition to higher order physical effects when designing in deep submicron. Traditionally simulation techniques have been used as the main verification tool in the AMS design process, however the limitation of simulation in terms of coverage affect the confidence in the verification results. Symbolic analysis have been developed as a complementary to numerical simulation, where the effect of parameters variations on the system behavior is analyzed. Although successful, challenging problems (like non linear effects) make this techniques only suitable for simple designs.

The last decade saw the emergence of a new engineering field known as hybrid system theory where researchers have developed formal techniques for the design and analysis automation of systems with real time and continuous

behavior and which are described by a composition of continuous time systems and discrete time systems. Model checking tools such as HyTech [3], CheckMate [2] and d/dt [1] were successful in the verification of hybrid systems with different degrees of complexity. Motivated by the success of the application of formal verification methods for real-time and hybrid systems, researchers started in recent years studying the applicability of such techniques for the verification of analog and mixed signal systems. In this paper, we intend to present an ongoing project for developing an approach for the analysis of analog systems using formal methods. The proposed method is based on the computation of finite-state conservative abstraction of the design. The methodology should guarantee that the verification on the abstract model is conservative. For this purpose, we use abstract interpretation, which is a theoretical framework for the analysis of infinite state systems and in this paper we choose interval arithmetics as the basis of the abstraction. In the hybrid system theory context, abstract interpretation was only used with simple systems with linear dynamics while other heuristic methods were applied for the abstraction of the nonlinear dynamics. In contrast, we propose a methodology for abstracting nonlinear dynamics using abstract interpretation, which will guarantee property preservation. To test and validate our methodology, we have used interval arithmetic tool AWA [10] to describe the behavior of tunnel diode oscillator.

The rest of the paper is organized as follows: In section 2, we give a brief overview of the project, followed by some preliminaries definitions in section 3. The abstraction methodology is described in section 4, with primary experimental results in 5. Related work is presented in 6 and finally, we conclude the paper with section 7.

2 Methodology

We present in this section our methodology for modeling and verification of continuous time systems. The basic idea is to adopt a multi abstraction technique in order to generate a compact model that can be automatically verified using model checking techniques, yet a conservative one as we need to guarantee property preservation. We start by solving the system of differential equations symbolically. As a start, we choose interval based methods proposed by Moore [11] which was developed to guarantees solution inclusion for differential equations, and used by Cousot and Cousot in software execution analysis [7]. The symbolic execution is monitored by temporal properties with predicates representing properties on the continuous state variables. These predicates form the base of the abstract model which is generated at the back end. Each abstract state is represented by region representing a conjunction of satisfied predicates. The advantage of this method is the accuracy of the analysis as symbolic methods are more accurate than qualitative methods avoiding the imprecision of these techniques. On the other hand, generation of the abstract model by monitoring the solution, once proved sound and conservative, avoids the cost of expensive

model checking of the whole discretized continuous behavior. Figure 1 gives an overview about the methodology.

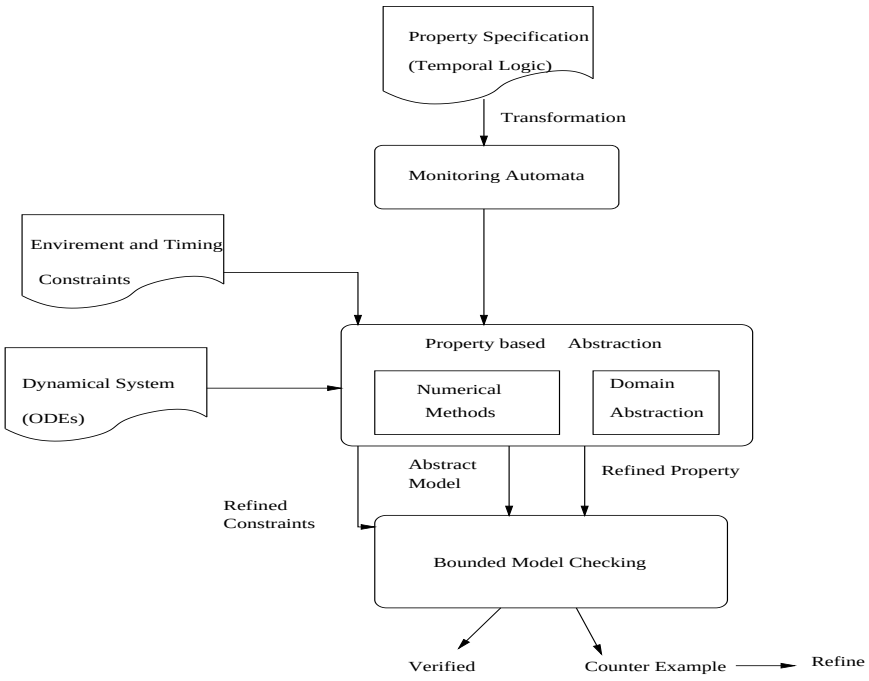


Fig. 1. Proposed verification methodology for continuous systems

In this paper, we focus only on part concerning the solution of differential equation using interval arithmetic. We demonstrate how to use this theory in order to infer reachability properties of non linear dynamical systems.

3 Modeling Continuous Systems

In this section we describe the class of non linear continuous systems represented by a system of ordinary differential equations. This modeling is suitable for describing analog systems behavior.

Definition 1. *A continuous dynamical system is a triple* $CS = (\mathbb{X}, \mathbb{X}_0, f)$ *with* $\mathbb{X} \subseteq \boldsymbol{R}^n$ *is the continuous state space,* $\mathbb{X}_0 \subseteq \mathbb{X}$ *and* $f : \mathbb{X} \to \mathbb{R}^n$ *is the continuous vector field.*

The behavior of a continuous dynamical system is governed by the differential equation $\dot{x} = f(x)$. We assume that the differential equation has a unique solution for each initial value $X(0) \in \mathbb{X}_0$, where $\mathbb{X}_0 \subseteq \mathbb{X}$ is the set of initial

continuous space. The continuous system is deterministic in the sense that from a given point it generates a unique trajectory.

Definition 2. *The semantics of a continuous dynamical system* $CS = (\mathbb{X}, \mathbb{X}_0, f)$ *over a continuous time period (an interval)* $T_c = [\tau_0, \tau_1] \subseteq \mathbb{R}^+$ *can be described as Trajectory. A trajectory of a continuous dynamical system is a continuous behavior* $\Phi_x : T_c \to \mathbb{X}$ *for* $x \in \mathbb{X}_0$ *such that* $\Phi_x(t)$ *is the solution of equation* $\dot{x} = f(x)$ *with initial condition* $\Phi_x(0) = x$ *and* $t \in T_C$*, is a time point and* T_c *is the continuous time domain.*

We can present the behavior of the continuous system by a transition system as follows:

Definition 3. *A dynamical continuous system can be considered as a transition system* $T_c = (Q, Q_0, \sigma, L)$ *where:*

- $q \in Q$ *is a configuration* (x, Γ)*, where* $x \in \mathbb{X}$ *and set of intervals* $\Gamma = \{t_0, \ldots\}$ *such that* $\cup_{i \geq 0} t_i \subseteq \mathbb{R}^+$*, where* $i \subseteq \mathbb{N} \cup \{\infty\}$*. We have* $t_1, t_2 \in \Gamma$ *for* $\Phi_{x'}(t_1) = \Phi_{x''}(t_2) = x$ *and* $x', x'' \in \mathbb{X}_0$ *.*
- $q \in Q_0$*, when* $t_0 \in \Gamma$ *and* $\Phi_x(t_0) = x$ *and* t_0 *is the interval* $[0, 0]$*,*
- L *is an interpretation function such that* $L : Q \to \mathbb{R}^n \times 2^{\mathbb{R}^+}$*.*
- $\sigma \subseteq Q \times Q$ *is a transition relation such that* $(q_n, q_m) \in \sigma$ *iff* $\exists t_n \in \Gamma_n$*,* $\exists t_m \in \Gamma_m$*.* $t_n < t_m$ *and* $\lim_{t_n \to t_m} \Phi_x^{q_n}(t_n) = \Phi_x^{q_m}(t_m)$*,* $x \in \mathbb{X}_0$*.*

The set of reachable states of a continuous system can be defined as follows:

Definition 4. *For a continuous dynamical system, the set of reachable states* $Reach \in \Gamma$ *can be defined as:*

- $Reach^{(0)} := Q_0$
- $Reach := \{q' \in Q | \exists q \in Reach^{(0)}, t \in L_\Gamma(q'), x' = L_x(q'), x = L_x(q)$ *such that* $\Phi_x(t) = x'\}$

Given a transition system C and a set of (un)safe states $B \subseteq \mathbb{X}$, whether $Reach \cap B$ is empty or not.

Remark 1. We have considered Γ associated with each state, as a set of closed time intervals and with a slight abuse of notations, we have referred to a trajectory between two time points as

$$\Phi_x : I_i \to \mathbf{R}^n$$

$$: [t_1, t_2] \mapsto x$$

We can described a discrete system as a transition system as follows:

Definition 5. *A transition system is a triple* $DS = (S, S_0, \sigma)$ *with* S *is the set of states,* $S_0 \subseteq S$ *is the set of initial states and* $\sigma \subseteq S \times S$ *is a relation capturing transitions.*

Definition 6. *The semantics of discrete system can be defined as a set of traces, where trace of a transition system* $DS = (S, S_0, \sigma)$ *is a sequence* $\pi : \boldsymbol{N} \rightarrow S$ *such that* $\pi(0) \in S_o$ *and* $\forall k \geq 0 : (\pi(k), \pi(k+1)) \in \pi$ *and* $\boldsymbol{N}$ *is the natural number domain.*

Usually the analytic solution for the differential equation is not possible and approximate methods are used instead. Given a CS with a configuration $\mathbf{C}$; to describe a correspondence between discrete evolution $\pi : \mathbb{N} \rightarrow \mathbf{C}$ and continuous evolution $\Phi : [0, \infty) \rightarrow \mathbf{Q}$, Tiwari *et.al* [31] defined the notion of sufficient complete discretization. This can be understood as sampling criteria that captures all the different states in the given continuous evolution. It can be formally defined as follows:

Definition 7. *A discrete evolution* $\pi : \mathbb{N} \rightarrow \boldsymbol{C}$ *is a sufficiently complete discretization of a continuous evolution* $\Phi : [0, \infty) \rightarrow \boldsymbol{Q}$, *if* $\pi(i) = \Phi^q_{x_0}(t_i) = x$ *where* $x = L_x(q) = L_x(c)$, $q \in \boldsymbol{Q}$, $c \in \boldsymbol{C}$, $i \in L_\Gamma(c)$, $t_i \in L_\Gamma(q)$, $x_0 = L_X(q_0)$ *and* $q_0 \in \boldsymbol{Q}_0$, *for all* $i \in \mathbb{N}$ *and* $\bigcup_{i \geq 0} t_i = \mathbb{R}^+$

Note: Practically, discretization is based on choosing time steps $\triangle t$, which can be varied or fixed and can be chosen in domains other than $\mathbb{N}$ (e.g. values of $\triangle t$ are in $\mathbb{Q}$ or rounded $\mathbb{R}$)

4 Abstract Interpretation

Given the concrete $D^\flat$ and abstract $D^\sharp$ semantics domains of the system under analysis. A soundness relation σ is used to reason about the correspondence between a concrete and abstract semantics. A soundness relation can be formulated as $\sigma \in \wp(D^\flat \times D^\sharp)$. We say there exists an abstract approximation if we assume that for every concrete semantics we have an abstract approximation: $\forall s \in D^\flat . \exists t \in D^\sharp : (s, t) \in \sigma$. More closely, to ensure the soundness of the methodology, we use Galois connection between $D^\flat$ and $D^\sharp$. By associating partial order relation with the semantics domain (i.e. ($\mathrm{D}^\flat$, $\sqsubseteq$), ($\mathrm{D}^\sharp$, $\preceq$) which are partial order set), we say (α, γ) is a Galois connection between $D^\flat$ and $D^\sharp$, iff $\alpha : \mathrm{D}^\flat \rightarrow \mathrm{D}^\sharp$, $\gamma : \mathrm{D}^\sharp \rightarrow 2\mathrm{D}^\flat$ *and*$\forall s \in \mathrm{D}^\flat$, $\mathrm{t} \in \mathrm{D}^\sharp$, $\alpha(s) \preceq t \Leftrightarrow s \sqsubseteq \gamma(t)$. To build an abstract state space $S^\sharp$, which is an overapproximation of the concrete state space $S^\flat$, that is $\forall s \in S^\flat . \exists t \in S^\sharp . t = \alpha(s)$. As we build our abstraction, it is essential that all transitions of the concrete system are preserved in the abstract, but the concretization of abstract transitions does not result in spurious transitions.

Note. Imposing the existence of Galois connection between concrete and abstract domains is a tight requirement as sometimes it is not easy of even impossible to find the abstract α function. Cousot [8], proposed relaxing this requirement by only working with concretization function like in the case of Interval domain as shown below.

4.1 Abstraction

Definition 8. *Let CS be a continuous dynamical system and DS be a discrete transition system with configurations $\boldsymbol{Q}$ (with domain $\mathbb{X}$) and $\boldsymbol{A}$ (with domain $\mathbb{A}$) respectively. We say DS is an abstraction for CS, if there exists a concretization mapping $\gamma : \boldsymbol{A} \rightarrow 2^{\boldsymbol{Q}}$, such that :*

- *$A = (a, \tau)$ is a configuration where $a \in \mathbb{A}$ and τ is the set of time intervals, such that every interval is an increasing sequence of time steps during which the state is not varied.*
- *$\mathbb{A}_0 = \{a \in \mathbb{A} | \exists x \in \gamma(a) \wedge x \in \mathbb{X}_0\}$, $A_0 = (a, \tau)$ and $t_0 \in \tau$ where t_0 is the singular interval.*
- *For every continuous evolution Φ, if π is a sufficiently discretization of $\alpha(\Phi)$, then π is a discrete trace of DS.*
- *$\sigma \subseteq \mathbb{A} \times \mathbb{A}$ is a relation capturing abstract transitions; $\{a \rightarrow' a' | \exists x \in \gamma(a),\ t \in \mathbb{R} : x' = \Phi_x(t) \in \gamma(a') \wedge x \rightarrow x'\}$*

Lemma 1. *For a concrete transition system with transition relation $\rightarrow$ and a corresponding abstract transition system with transition relation $\rightarrow'$, we have $\rightarrow \subseteq \gamma(\rightarrow')$*

The set of successor states of $a \in \mathbb{A}$ is $Post(a) = \{a' \in \mathbb{A} | a \rightarrow' a'\}$ and the set of reachable states *Reach* of a transition system can then be defined as follow:

Definition 9. *For a transition system, the set of reachable states $AReach \subseteq \boldsymbol{A}$ is defined as:*

- *$AReach^{(0)} := \boldsymbol{A}_0$*
- *$AReach^{inc(i)} := Post(AReach^{(i)})$, $\forall i > 0$*
- *$AReach := \bigcup_{i \geq 0} AReach^{(i)}$*

The verification problem is stated as follows: Given a transition system DS and a set of (un)safe states $B \subseteq \mathbf{A}$, is there a trace starting from $\mathbf{A}_0$ that can reach B; whether $AReach \cap B$ is empty or not. We say that AReach is an over approximation of Reach

Lemma 2. *Let CS be a continuous dynamical system and DS a discrete transition system as its abstraction, we have: $Reach \subseteq \{q \in \boldsymbol{Q} | \exists a \in AReach \wedge q \in \gamma(a)\}$*

4.2 Interval Abstract Domain

In this paper, we choose intervals as the numerical abstract domains with which we compute the abstract semantics of continuous systems. We will briefly outline constructing the abstract domain by following the modular approach presented by Mine in [9] were he propose starting from a basis representing abstraction of state variables as well as basic operations and using lifting of the concretization and abstraction functions to sets and functions representing expressions and

transfer functions. The interval abstract domain D^ι is based on the classical concept of interval arithmetics [11] and was adapted by Cousot and Cousot in [7].

The relation between the concrete and abstract domain can be described by a partial Galois connection $(\wp(\mathbb{R}), \leq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (B, \sqsubseteq_B^\sharp)$ with:

Interval Basis B^ι. B is a set of intevals with bounds in $\mathbb{R}$, where B : $\{\bot_B^\iota\} \cup \{[a,b] | a \in \mathbb{R} \cup \{-\infty\}, b \in \mathbb{R} \cup \{\infty\}, a \leq b\}$. We say B^ι contains $[-\infty, \infty]$ which denote the whole set $\mathbb{R}$, the singletons $[a,a]$, when $a \in \mathbb{R}$, and the empty interval denoted by $\bot_B^\iota$.

Interval Basis Structure The partial order $\sqsubseteq_B^\iota$ is defined as $[a,b] \sqsubseteq_B^\iota [a',b'] \triangleq a \geq a'$ and $b \leq b'$ and $\bot_B^\iota$ is the smallest element. The basis concretization γ_B^ι is defined as γ_B^ι ([a,b]) $\triangleq \{x \in \mathbb{R} | a \leq x \leq b\} and \gamma_B^\iota(\bot_B^\iota) \triangleq 0$.

Interval Basis Operators. We define the required operators for a basis the following way:

– $\cup_B^\iota$ and $\cup_B^\iota$:

$$X^\sharp \cup_B^\iota Y^\sharp \triangleq \begin{cases} [min(a,a'),\ max(b,b')] & if\ X^\sharp = [a,b] \text{and} Y^\sharp = [a',b'] \\ X^\sharp & if Y^\sharp = \bot_B^\iota \\ Y^\sharp & if X^\sharp = \bot_B^\iota \end{cases}$$

$$X^\sharp \cap_B^\iota Y^\sharp \triangleq \begin{cases} [max(a,a'),\ min(b,b')] & if\ X^\sharp = [a,b]\ and\ Y^\sharp = [a',b'] \\ & and\ max(a,a') \ \leq \ min(b,b') \\ \bot_B^\iota & otherwise \end{cases}$$

– In general, abstraction for arithmetics operators can be described as follows:
- $\neg^\sharp : B \to B$ such that$\gamma_B^\iota(\neg^\sharp X^\sharp) \supseteq \{\neg x | x \in \gamma_B^\iota(X^\sharp)\}$
- $\diamondsuit^\sharp : B^2 \to B$ such that$\gamma_B^\iota(X^\sharp\ \diamondsuit^\sharp Y^\sharp) \supseteq \{x \diamondsuit y | x \in \gamma_B^\iota(X^\sharp$, y $\in \gamma_B^\iota(Y^\sharp)\}$ and $\diamondsuit \in \{+, -, \times, \div\}$

For interval arithmetics we have the following:

[a,b]$^\iota$	$\triangleq$	[a+b]	
[a,b] $+^\iota [a',b']$	$\triangleq$	[a+a', b+b']	
[a,b] $-^\iota [a',b']$	$\triangleq$	[a-b', b-a']	
[a,b] $\times^\iota [a',b']$	$\triangleq$	[min (aa', ab', ba', bb'), max (aa', ab', ba', bb')]	
$1 \div [a,b]$	$\triangleq$	$[1 \div b, 1 \div a]$	$if\ 0 \notin [a,b]$
[a, b] $\div [a',b']$	$\triangleq$	[a,b] $\times 1 \div [a',b']$	

Note We use α^ι and γ^ι also to denote liftings of the functions α_B^ι and γ_B^ι to sets, relations and functions (e.g, expression evaluation, variables assignments, etc.).

Interval domain $\mathbb{I}$ is a conservative domain which overapproximate the original one $\mathbb{R}$. We can describe the discrete interval transition systems as following:

Definition 10. *A discrete interval transition system is a discrete transition system* $T_c = (A, A_0, \sigma, L)$ *with domain* $\mathbb{I}^n$*, where* n *is the dimension of the state space, where:*

- $I = (a, \tau)$ *is a configuration where* $a \in \mathbb{I}$ *and* τ *is the set of time intervals, such that every interval is an increasing sequence of time steps during which the state is not varied.*
- $\mathbb{I}_0 = \{a \in \mathbb{I} | \exists x \in \gamma(a) \wedge x \in \mathbb{X}_0\}$*,* $I_0 = (a, \tau)$ *and* $t_0 \in \tau$ *where* t_0 *is the initial singular interval.*
- $\sigma \subseteq \mathbb{I} \times \mathbb{I}$ *is a relation capturing abstract transitions;* $\{a \rightarrow a' | \exists x \in \gamma(a),\ t \in \mathbb{R} \ :\ x' = \Phi_x(t) \in \gamma(a') \wedge x \rightarrow x'\}$

We can therefore deduce the following theorem:

Theorem 1. *Let CS be a continuous system and DS be a discrete Interval Transition system, as defined above. Then DS is an abstraction for CS.*

5 Application: Tunnel Diode Oscillator

Several interval arithmetic implementation for the the initial value problem have been developed, see for instance [12] for an overview. We have used in this report, AWA tool developed by Lohner [10], it has the advantage of efficiently dealing with the wrapping effect (error resulting from enclosing non rectangular regions by rectangular ones, which can lead to exponential growth in overapproximation, hence reducing the precision). To illustrate our methodology, we used the tunnel diode oscillator (see Figure2).

Tunnel diode oscillator circuit has attracted the interest of many researchers working on the verification of AMS designs. See for instance [27, 23, 15]. This is largely due to its wide usage in analog system designs (i.e.,the most common application of a tunnel diode is in high-frequency oscillator circuits) and technically because of its non linear behavior from which several properties can be deduced. Tunnel diodes exploit a phenomenon called resonant tunneling to provide interesting forward-bias characteristics, due to its negative resistance characteristic at very low forward bias voltages. That means that for some range of voltages, the current decreases with increasing voltage. This is in contrast with conventional diodes have a nonlinear I-V characteristic, but the slope of the curve is always positive. This characteristic makes the tunnel diode useful as oscillator. When a small forward-bias voltage is applied across a tunnel diode, it begins to conduct current. As the voltage is increased, the current increases and reaches a peak value called the peak current. If the voltage is increased a little more, the current actually begins to decrease until it reaches a low point called the valley current. If the voltage is increased further yet, the current begins to increase again, this time without decreasing into another valley.

In this section, we present results from experiments with the tunnel-diode oscillator circuit. We focus on the current I_L and the voltage V_C across the

tunnel diode in parallel with the capacitor of a serial RLC circuit (see Figure 2). The state equations of the circuits are given as follows:

$$\dot{V}_C = \frac{1}{x}(-I_d(V_C) + I_L) \text{ and } \dot{I} = \frac{1}{L}(-V_C - \frac{1}{G}I_L + V_{in})$$

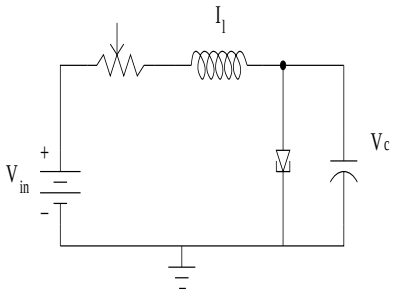


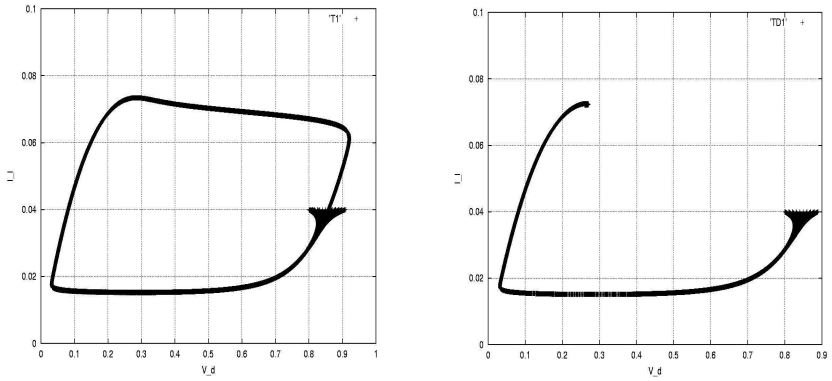
Fig. 2. Oscillation

Where $I_d(V_C)$ describes the non-linear tunnel diode behavior. We analyze the circuits in two modes. The first when the circuit is in stable oscillation for a given set of parameters, the other case when this oscillation dies out. The kind of properties we are interested to verify can be for example: *The system behavior will be the same for the set of initial condition*, or *For which set of parameters values, circuit oscillates?*. In this paper, we limits ourself with properties of the first type, similar to the one verified in [23, 15]. We use a variant of ACTL temporal logic extended with predicates of Real and time intervals restricting temporal operators, see [13] for more details.

We chose these two different set of parameters values of the oscillator circuit $\{C = 1000e^{-12},\ L = 1e^{-6},\ G = 5000e^{-3},\ Vin = 0.3\}$ and $\{C = 1000e^{-12},\ L = 1e^{-6},\ G = 2000e^{-3},\ Vin = 0.3\}$ along with the set of initial values of voltages $[0.8\ V,\ 0.9\ V]$ and currents $0.04\ mA$ and the analysis region of interest $-1\ V \leq V_C \leq 1\ V$ and $0.01\ mA \leq I_L \leq 0.9\ mA$. By using interval based analysis, with the first set of parameters, we can find out that the circuit is oscillating for the given set of initial conditions (see Figure 3.a). By applying the reachability analysis in Definition 9, we verify that for the given initial conditions, the trajectory will be within the analysis region. Suppose we want to verify the following property on the set of trajectories[23]:

$$\forall\Box_{[0,1e^{-6}]}(\forall\Diamond(I_L \leq 0.02)) \ \wedge \ \forall\Box_{[0,1e^{-6}]}(\forall\Diamond(I_L \geq 0.06))$$

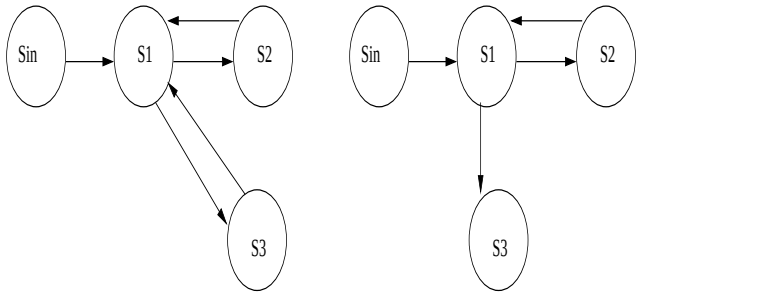
Which can be understood as within the time interval $[0, 1e^{-6}]$ on every computation path, first predicate is true at some future time as well as for the second predicate. This property checks for oscillation behavior. We can divide the state space into 3 states, each corresponding to a set of predicate inequalities extracted



(a. Oscillation) (b. No Oscillation)

Fig. 3. Oscillation

from the monitors in addition to the initial condition (as shown in figure 4.a). It is abvious that within the specified time interval, the property in verified.



(a. Oscillation TS) (b. Non Oscillating TS)

Fig. 4. Transition Systems

Which can be understood as within the time interval $[0, 1e^{-6}]$ on every computation path, first predicate is true at some future time as well as for the second predicate. This property checks for oscillation behavior. We divide the state space into 3 states in addition to the initial condition, each corresponding to a set of predicate inequalities extracted from the formulas ($P_1 = I_L \leq 0.02$, $P_2 = 0.02 < I_L < 0.06$, $P_3 = \ I_L \geq 0.06$), as shown in Figure 4.a. We can use a labeling function *Prop* associating a set of atomic propositions to each state; $Prop : Q \rightarrow 2^{AP}$. For example, $Prop(S1) = \{\bar{P_1}, P_2, \bar{P_3}\}$. Monitors synthesized from the temporal logic can be used similar to threshold detection. Each time a condition is changed, monitors triggers a change of states of the abstract tran-

sition system. For example, there is a transition between state $S1$ and state $S2$, as the trajectories generated by interval analysis and monitored by a monitoring automata cross the region satisfying the conjunction of atomic propositions $\bar{P_1} \wedge P_2 \wedge \bar{P_3}$ to the region satisfying $P_1 \wedge \bar{P_2} \wedge \bar{P_3}$. In order, to verify oscillation condition using model checking on the abstract model , we translate CT-CTL properties to TCTL properties. The model checking procedure is then straight forward and model checking tools like SMV or Hytech can be used to check such properties.

By following the same procedure for the system with the second set of parameters, but with the same initial conditions, we can find out that the circuit is non oscillating. When the circuit starts up, the energy of the system is lost due to the positive circuit resistance. Starting from any point in the analysis region, the oscillations die down to the equilibrium point (see Figure 3.b). By applying the reachability analysis in Definition 9, we verify, however, that for the given initial conditions, the trajectory will be within the analysis region. If we want to verify the oscillation property, we start by creating the abstract transition system as shown in Figure 4.b. and by applying model checking algorithms, we find out that the condition is not holding. This can be inferred by looking at the transition system in Figure 4.b, as there is no transition out from $S3$, which can be interpreted as once the system reaches $S3$, it remains deadlocked there during the verification period.

6 Related Work

An important issue of algorithmic methods in formal verification and model checking of AMS circuits are the solution of continuous systems; that is, the collection of continuous time trajectories starting from a set of initial continuous states where in practice the initial conditions are usually not known exactly but only known to lie within some range. Several methods for approximating reachable sets for continuous dynamics have been proposed. These methods rely on the discretization of the continuous state space. Among the most important approximations are those based on cubical and polyhedral representation. This approaches were pursued by Kurshan and McMillan in [19] and Greenstreet [20, 21] respectively were they proposed verifying digital properties at the transistor level. A variant approach of polyhedral based analysis was adapted by Dang and Maler [1] and implemented in the tool d/dt and by Chutinan and Krogh and implemented in their tool Checkmate [2] which supports in addition temporal verification. In [22], Dang *et. al*, Dang and Maler, used d/dt for the verification of analog systems described with differential algebraic equations (DAEs) and apply it to the verification of a biquad low-pass filter. In addition they proposed using techniques from optimal control (i.e hybrid constrained optimization) in order to find bounds of the reachability and they applied this technique to the verification of first order $\Delta - \Sigma$ modulator. In [23], the authors used Checkmate tool for the verification of AMS designs, i.e, tunnel diode oscillator and $\Delta - \Sigma$ modulator

In [15], Hartong *et. al* proposed discretizing the whole state space into variable sized regions to represent the state space and he used some kind of estimation techniques to describe possible transitions between partitions. The discretized state space is then encoded and CTL based model checking is applied. The paper gave no proof of soundness and It is not clear how the whole CTL is supported as in general approximation can only preserves a subset namely the ACTL. The proposed approach was implemented in a tool called Amcheck, developed at University of Hannover. in [16], they extended their methodology for the verification of time properties (i.e rise and fall time) of analog circuits.

Several abstraction techniques have been proposed for continuous and Hybrid systems. One of the early work for applying abstract interpretation to Hybrid systems was proposed by Halbwachs *et.al* [28] as extension to a previous work with Cousot [14]. In this work, convex approximation of linear equations is described. A variant of this work is latter implemented in HyTech [3]. The main limitation of this approach is the applicability only for linear systems, which practically restrict the class of systems under verification. Hypertech [4], is an extended version of Hytech, where they add an interval library to support verification of non linear systems. The idea presented is similar to our proposal, however we diverge from them in that we propose combining interval analysis with property guided abstraction which can lead to efficient analysis of the non linear systems. Henzinger *et. al* [5] present a methodology for algorithmically analyzing nonlinear hybrid systems by first translating the system to linear hybrid automata, and then using automated model-checking tools. In a linear hybrid automaton, the continuous environment is partitioned into a finite number of classes such that within each class, the continuous variables are governed by a constant polyhedral differential inclusions. Although, the idea is very interesting, generated linear hybrid automata are still large enough to be easily model checked. One other problem of this approach is the linearization of the dynamics which results in losing information which might be of importance like the effect of external disturbance for example, make it impractical for the verification of analog circuits.

Predicate abstractions [30] have been successful in the verification of infinite systems. I have been extended to the abstraction of verification of hybrid systems with different complexity, see for instance the work by Alur *et. al* [6], Ahish *et. al* [31]. One main problem in this approach is in choosing the correct predicate. In our proposed methodology, we plan to use predicates in temporal logic as a mean to build the abstract state space, by monitoring the execution, each set of solutions satisfying certain predicates are represented by a certain discrete state and finally, Clarke *et. al* [32], extended the Checkmate verification toolbox with an abstraction refinement methodology.

Program monitoring have been applied sucessffuly for hardware and software analysis. Motivated with the success of PSL assertion languages for hardware verification, Maler *et. al.* [26] proposed a simple methodology for monitoring the simulation of continuous signals by extending the PSL logic to support predicates over the reals (signals) , the goal is to verify continuous and analog systems.

Monitoring was applied within the Charon design framework [25] where timed automata and linear hybrid automaton can be used to monitor real-time and hybrid behavior. Recently similar ideas using hybrid automata as monitors have been integrated with the PHVAR a hybrid system analysis tool [27] that provides sound verification results based on linear hybrid automata approximations and was used to verify properties of piecewise models of oscillator like amplitude bounds and phase jitter.

Interval and affine arithmetics have been used in the analysis of analog circuits like in [17], were the authors presents an approach for equivalence checking of linear analog circuits with parameter tolerances. The goal is to prove that an actual circuit fulfills a specification in a given frequency interval for all parameter variations and in [18], affine arithmetics was used to for helping in the circuit sizing.

7 Conclusion

The lack of methods for computing reachable sets of continuous dynamics has been the main obstacle towards an algorithmic verification methodology for hybrid systems. This motivated us to tackle first the reachability problem of continuous systems. Unlike the conventional approaches which attempt to find exact solutions and are thus limited by undecidability of most non-trivial systems, our approach is based on an efficient method for abstracting the continuous behavior using combination of techniques from numerical methods, abstract interpretation, and program monitoring.

Future Work. The description of the methodology presented in this section was general. Different issues will be raised throughout the development. We present below some issues we think are of important interests:

- Extends the methodology for other domains like affine arithmetics which have been developed to enhance precision.
- Explore more complex case studies; currently we are working on the verification of phase locked loop designs and scmitt trigger.
- Extends the methodology for Hybrid systems.

References

1. Eugene Asarin, Thao Dang, Oded Maler: The d/dt Tool for Verification of Hybrid Systems. in Computer Aided Verification, LNCS 2404, Springer, 2002, pp. 365-370.
2. A. Chutinan and B. H. Krogh, Computational techniques for hybrid system verification. IEEE Trans. on Automatic Control, vol. 48, no. 1, 2003, pp. 64-75.
3. T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: A Model Checker for Hybrid Systems. Software Tools for Technology Transfer 1:110-122, 1997.
4. Thomas A. Henzinger, Benjamin Horowitz, Rupak Majumdar, Howard Wong-Toi: Beyond HYTECH: Hybrid Systems Analysis Using Interval Numerical Methods. HSCC 2000: 130-144

5. Thomas A. Henzinger, Pei-Hsin Ho: Algorithmic Analysis of Nonlinear Hybrid Systems. CAV 1995: 225-238
6. R. Alur, T. Dang, and F. Ivancic. Counter-example guided predicate abstraction of hybrid systems, Ninth International Conference on Tools and Algorithms for the Construction and Analysis of Systems, 2003
7. P. Cousot, R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In Conference Record of the Fourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pages 238252, Los Angeles, California, 1977. ACM Press, New York, NY, USA.
8. Patrick Cousot, Radhia Cousot: Abstract Interpretation Frameworks. J. Log. Comput. 2(4): 511-547 (1992)
9. A. Mine, Weakly Relational Numerical Abstract Domains. PhD thesis Report, Computer Science Department, the Ecole Normal Suprieure , France, Dec. 2004
10. R. J. Lohner, Enclosing the solutions of ordinary initial and boundary value problems, in Computer Arithmetic: Scientific Computation and Programming Language, Wiley-Teubner Series in Computer Science, Stuttgart, 1987, 255-286
11. R. E. Moore Methods and Applications of Interval Analysis, Society for Industrial Applied Mathematics, Philadelphia, 1979, ISBN 0898711614.
12. R. B. Kearfott. Interval computations: Introduction, uses, and resources. Euromath Bulletin, 2(1):95-112, 1996.
13. M. Zaki, S. Tahar, G. Bois, Formal Verification of Analog and Mixed Signal Systems. In preparation
14. P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In Proceedings of the 5th Annual ACM Symposium on Principles of Programming, pages 84–97, 1978.
15. W. Hartong, R. Klausen, L. Hedrich, "Formal Verification for Nonlinear Analog Systems: Approaches to Model and Equivalence Checking," Advanced Formal Verification, R. Drechsler, ed., Kluwer Academic Publishers, Boston, January 2004, pp. 205-245.
16. D. Grabowski, D. Platte, L. Hedrich, and E. Barke, Time Constrained Verification of Analog Circuits using Model-Checking Algorithms,Workshop on Formal Verification of Analog Circuits, 2005.
17. L. Hedrich and E. Barke,A Formal Approach to Verification of Linear Analog Circuits with Parameter Tolerances. Design, Automation and Test in Europe, 1998, pp. 649-654.
18. Andreas C. Lemke, Lars Hedrich, Erich Barke: Analog circuit sizing based on formal methods using affine arithmetic. ICCAD 2002: 486-489
19. R.P. Kurshan and K.L. McMillan. Analysis of digital circuits through symbolic reduction. IEEE Trans. on Computer-Aided Design 10:13501371, 1991.
20. Mark R. Greenstreet, Ian Mitchell: Reachability Analysis Using Polygonal Projections. HSCC 1999: 103-116
21. Mark R. Greenstreet, Ian Mitchell: Integrating Projections. HSCC 1998: 159-174
22. Thao Dang, Alexandre Donze, O.M.: Verification of analog and mixed-signal circuits using hybrid system techniques. In: Formal Methods in Computer-Aided Design (FMCAD 2004), Austin, Texas, November 14-17, 2004. (2004)
23. Gupta, S.; Krogh, B.H.; Rutenbar, R.A.:Towards formal verification of analog designs, IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004. , 7-11 Nov. 2004 Page(s):210 - 217

24. A. Bemporad and M. Morari Verification of hybrid systems via mathematical programming.In F.W. Vaandrager and J.H. van Schuppen, editors, Hybrid Systems: Computation and Control, volume 1569 of Lecture Notes in Computer Science, pages 31–45. Springer Verlag, 1999.
25. Li Tan, Jesung Kim, Insup Lee: Testing and Monitoring Model-based Generated Program. Electr. Notes Theor. Comput. Sci. 89(2): (2003)
26. Oded Maler, Dejan Nickovic: Monitoring Temporal Properties of Continuous Signals. FORMATS/FTRTFT 2004: 152-166
27. G. Frehse, B. Krogh, R. Rutenbar, O. Maler Time Domain Verification of Oscillator Circuit Properties, Workshop on Formal Verification of Analog Circuits, 2005
28. N. Halbwachs, P. Raymond, and Y. Proy. Verification of linear hybrid systems by means of convex approximations. In B. LeCharlier, editor, Proceedings of International Symposium on Static Analysis, volume 864 of Lecture Notes in Computer Science. Springer-Verlag, September 1994.
29. T. A. Henzinger and P. Ho. A note on abstract-interpretation strategies for hybrid automata. In Hybrid Systems II, Lecture Notes in Computer Science 999, Springer-Verlag, 1995, pp. 252-264.
30. Susanne Graf, Hassen Sai"di: Construction of Abstract State Graphs with PVS. CAV 1997: 72-83
31. Ashish Tiwari, Gaurav Khanna: Series of Abstractions for Hybrid Automata. HSCC 2002: 465-478
32. Edmund M. Clarke, Ansgar Fehnker, Zhi Han, Bruce H. Krogh, Olaf Stursberg, Michael Theobald: Verification of Hybrid Systems Based on Counterexample-Guided Abstraction Refinement. TACAS 2003: 192-207