Model Checking Algorithms for Analog Verification

Walter Hartong, Lars Hedrich, Erich Barke
Institute of Microelectronic Circuits and Systems, University of Hannover
Appelstrasse 4, 30167 Hannover, Germany, www.ims.uni-hannover.de
hartong, hedrich, barke@ims.uni-hannover.de

ABSTRACT

In this contribution we present the first method for model checking on nonlinear analog systems. Based on digital CTL model checking algorithms and results in hybrid model checking, we have developed a concept to adapt these ideas to analog systems. Using an automatic state space subdivision method the continuous state space is transfered into a discrete model. In doing this, the most challenging task is to retain the essential nonlinear behavior of the analog system. To describe analog specification properties, an extension to the CTL language is needed. Two small examples show the properties and advantages of this new method and the capability of the implemented prototype tool.

Categories and Subject Descriptors

I.6.m [**Computer Methodologies**]: Simulation and Modeling

General Terms

Algorithms, Language, Theory, Verification

Keywords

Model Checking, Formal Methods, Nonlinear Analog Systems

1. INTRODUCTION

In recent years formal methods, like equivalence checking and model checking, have been successfully introduced into the digital design flow, indicated by the growing number of commercial vendors. However, these methods are still almost unknown in analog verification. Symbolic analysis is a promising technique but it cannot cope with the power of formal methods in the digital world. As far as we know, there has been only one approach published on equivalence checking for analog circuits [5].

On the other hand, digital tools have been extended to special hybrid systems, being digital systems connected to some analog blocks or to an analog environment. These techniques are focused on the digital part of the system. The analog behavior is mostly restricted and the verification results are not appropriate to assess the functionality of the analog part, because the analog behavior is assumed to be correct during the verification process. The result does therefore not include useful information for analog designers. Furthermore, model checking languages used are not able to describe analog system properties. Thus, formal methods for analog systems are still lagging far behind the digital tools.

In this contribution a discrete model for the nonlinear system will be presented. Further, model checking algorithms known from digital model checking tools will be adapted to analog systems.

This paper is organized as follows. In Section 2 a general comparison between digital, hybrid and analog systems is given. The next two sections describe the main algorithms developed, namely the generation of the discrete model and the model checking methods. Finally, some experimental results are given, showing the use of the developed prototype.

DAC 2002, June 10-14, 2002, New Orleans, Louisiana, USA.

2. SYSTEM DESCRIPTION

The most general system class to be considered in this context is the transition system. Most digital and hybrid model checking tools are based on such systems.

Definition 1. A state transition system $T = (Q, Q_0, \Sigma, R)$ consists of

- a set of states Q and a set of initial states Q_0,
- a set of generators or events Σ and
- a state transition relation $R \subseteq Q \times \Sigma \times Q$.

The state transition system is a suitable model for digital as well as for analog systems [11]. In the analog case, the set of states Q can be represented by an Euclidean space (state space) spanned by the system's state variables. The number of states is infinite, due to the continuous definition of the state variables. The initial state Q_0 is a single point in the state space. Often, but not necessarily, it is the DC operating point. There are only two generators Σ causing state transitions, namely, the time t and the input values $u(t)$. The state transition relation R is a continuous function with respect to time. It can be visualized by a vector field in the state space.

Unfortunately, the representation of states and state transitions in the discrete digital case is totally different. Published model checking tools for hybrid systems, require discrete or discrete and partly linear system descriptions [1]. Our focus is on nonlinear analog behavior, which cannot be described by these simple approximations. Linear phase-portrait approximation [6] is an encouraging method, but the calculation of proper state space intersection planes seems to be complicated, especially for high dimensional and strongly nonlinear problems. There are two aspects which make a different approach necessary. Firstly, model checking for analog systems requires extensions of the language to define analog system properties in a reasonable way. Secondly, the analog system model used in the model checker needs to retain the essential analog dynamics to get useful results.

2.1 Analog System

An analog system description consists of a nonlinear first order differential algebraic equation system

$$f(\dot{x}(t), x(t), u(t)) = 0 \tag{1}$$

with the input variable vector $u(t)$ and the vector $x(t)$ of n system variables. In general, the function f is arbitrarily nonlinear. However, in practice the nonlinearities are restricted by the device models used. There are several ways to build such equations from a transistor netlist or a behavioral model, for example the modified nodal analysis (MNA).

The number of state variables is less than or equal to the number of energy storage elements (e.g. capacitors and inductors). The definition of state variables spanning the state space depends on the way to build up Equation (1) and is not unique for one system [4]. Moreover, the link between state variables and system variables is not necessarily obvious. Mathematically, this is known as initial value problem for differential algebraic equations [8]. However, since we focus on basic model checking methods, these issues will not be discussed here. To take the input values into account, the state space can be extended by the number of inputs. We call this extended state space. Since, the input values are not determined by the system itself but rather by environmental conditions, it is not possible to model their behavior within the state space.

3. DISCRETE MODEL GENERATION

Some of the following algorithms have been inspired by research in the area of approximating dynamical behavior [3]. Despite the similarities, there are a lot of differences, mainly caused by the overall target of the algorithms. Therefore, the algorithms have not been used, but some of the ideas have been adopted.

As we have seen, the continuous variables in an analog system - state values and time - have to be transfered into a discrete state space description and a state transition relation. The next three sections illustrate this process.

3.1 Discrete Time Steps

The transition relation R for an analog system is given by the state variables' time derivation $\dot{x}(t)$, which is a continuous function. The actual state transition can be calculated by integrating this function.

$$x(t) = x_0(t_0) + \int_{t_0}^{t} \dot{x}\left(x(t), u(t)\right) dt \tag{2}$$

In practice $\dot{x}(t)$ may possibly not be derived as explicit function, however, the relation is given implicitly by Equation (1). In general, $x(t)$ can only be calculated using numerical integration. This problem is well known in analog simulators, like Spice, Spectre, Saber, etc. During transient simulation the differential equation system is solved using discrete time steps. Given a small time step $\Delta t = t_1 - t_0$, the transition between the actual state $x(t_0)$ and the next state $x(t_1)$ is determined by a numerical integrator, using e.g. the backward Euler formula:

$$x(t_1) = \left\{ x \;\middle|\; f\left(\frac{x - x(t_0)}{t_1 - t_0}, x, u(t_1)\right) = 0 \right\} \tag{3}$$

A step size control uses the second derivation with respect to time for a local measurement of the integration error. If the given error threshold is exceeded, step size is reduced otherwise the transient step is accepted.

This method can be directly used in an analog model checking tool. An arbitrary test point in the state space is mapped to its successor state, depending on the actual Δt used. In contrast to transient simulation, there is no temporal predecessor state for a test point. A second time step has to be calculated for each point to determine the second derivation, enabling a local error control.

In general, the time step Δt will vary throughout the state space, due to the step size control. As we will see later, this makes the checking of explicit time dependencies difficult. To make this, easier the time step is chosen to be equal for each test point within one state space region (see Section 3.2). By this algorithm, every state space point can be mapped to its successor point including a local step length control. The resulting tuple of test and target point is represented by a successor-vector in the state space. Due to the error control, the vector length is restricted to reasonable values in terms of the second derivation. Thus, a time discrete transition relation for single points is given: $s(x(t)) = x(t + \Delta t)$.

3.2 State Space Subdivision

To get a discrete and finite state description, the continuous and infinite state space has to be bounded and subdivided. The restriction to a finite region is simply done by a user defined start area, comprising the considered system behavior. This causes special border problems, which will be discussed later. However, it does not impact the correctness of the model checking result because in real world systems there is alway a natural bound for the state variable values.

Since we do not have a digital environment, a natural subdivision for the start area, given for example by threshold values of digital state transitions, is missing [1]. Furthermore, to retain the analog system's behavior correctly, a sufficient number of subdivisions is needed, especially in state space regions with highly nonlinear behavior. On the other hand, the number of discrete regions should be as small as possible to reduce the total runtime.

The subdivision is done by rectangular boxes or hyper boxes, which are not necessarily the best choice [6]. However, for implementation reasons boxes are the far most convenient data structure. Other subdivision geometries might be considered during future improvements.

At first, we start with a user controlled uniform subdivision in all state space dimensions. Secondly, an automatic subdivision strategy is used to react on different system dynamics, depending on the actual state space region. The main target is to get a uniform behavior in each state space box. The uniformity is measured by the variation of the successor-vectors ($sv(x) = s(x) - x$), calculated in the state space (Section 3.1). Namely, vector length l_m and angle a_m between different vectors are considered. Equations (4) and (5) give the definition of these values. The function $L(\cdot)$ gives the length of the delivered vector or vector component.

$$l_m = 1 - \frac{\min_{x \in box} L(sv(x))}{\max_{x \in box} L(sv(x))} \tag{4}$$

$$a_m = \max_{\substack{x \in box \\ i \in dim}} \frac{L(sv(x_i))}{L(sv(x))} - \min_{\substack{x \in box \\ i \in dim}} \frac{L(sv(x_i))}{L(sv(x))} \tag{5}$$

Box subdivision is continued recursively until l_m and a_m drop under a given threshold or a given subdivision depth is exceeded. Within the expected accuracy, all boxes fulfilling the l_m and a_m thresholds do not contain fix points. This is, because fix points are always surrounded by regions with ununiform behavior in terms of Equation (4) and (5) (see midpoint in Figure 1). This information is stored and used in the connection relation algorithm (Section 3.3).

Additional subdivisions are applied, if the successor-vectors in a region are too short in relation to the box size. This occurs mainly in regions where the system is strongly nonlinear, which implies Δt to be very small. Each box in the state space will represent a single state in the discrete model. Thus, the set of i states is given by $Q = \{box_1, box_2, ..., box_i\}$.

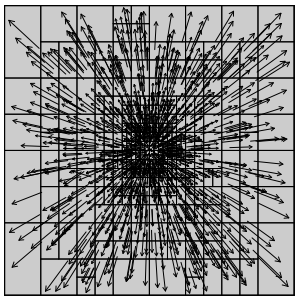

Figure 1: Automatic state space subdivision.

The result of automatic subdivision for a simple example, given by $\{\dot{x}_1 = x_1, \dot{x}_2 = x_2\}$, is shown in Figure 1. According to the uniformity values, box size is getting smaller in the middle of the picture, until the maximum subdivision depth is reached.

3.3 Connection Relation

The last step in getting a discrete system model is the connection relation between state space regions. In Section 3.1 successor states for single state space points have been defined ($s(\cdot)$). Using this point to point relation, the target region r_{target} is given by the set of all target points associated with a test point within the state space region r_{test} (see Equation (6)), as illustrated by the gray regions in Figure 2. We call this exact transformation $T_1(\cdot)$.

$$r_{target1} = \{s(y) \mid y \in r_{test}\} = T_1(r_{text}) \tag{6}$$

Since it is not practical to calculate a huge or - mathematically - infinite number of successor-vectors for each box, a good estimation or inclusion of the target region is needed. Three different approaches will be discussed.

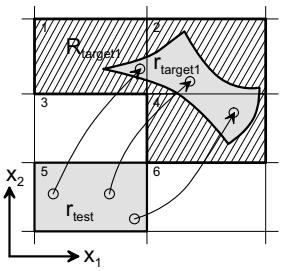

Figure 2: State space connection relation using T_1.

An inclusion $r_{target2}$ can be calculated using interval analysis [10]. This approach provides an exact and overestimated solution. That means, the correct solution $r_{target1}$ is fully included in $r_{target2}$. However, $r_{target2}$ might be much larger than $r_{target1}$. This effect is called overestimation and might be a serious problem for large systems [5]. Moreover, this approach is very time consuming.

$$r_{target2} = T_{interval}(r_{test}) = T_2(r_{test}) \supseteq r_{target1} \tag{7}$$

A more practical but also less accurate way to approximate the target region is to choose a number of test points p_{test} within the test region and to calculate the dedicated target points p_{target}. The target region $r_{target3}$ can be approximated using an appropriate inclusion of these points. As we will see below, an inclusion operation is also needed while expanding the target regions to the actual state space regions. I.e. these two steps can be combined. Even only a few test points may give a reasonable target approximation, but the region $r_{target3}$ might be under- or overestimated.

$$p_{test3} = \{s_1, s_2, ..., s_n\},\ s_i \in r_{test} \tag{8}$$

$$r_{target3} = \{\text{inclusion}(s(y)) \mid y \in p_{test3}\} \tag{9}$$

$$= \mathrm{T}_3(r_{test}) \simeq r_{target1} \tag{10}$$

It is shown in [7] that T_3 is surely overestimated if all corner values are used as test points and if $s(\cdot)$ can be assumed to be monotonic. Following the argumentation in [3], T_3 can also be made rigorous using Lipschitz constants L in each state space dimension. Using a grid of test points, spaced by h, one can calculate a extension diameter $d_{ex} = Lh$ for the target points. Expanding each target point by this diameter d_{ex} in each dimension gives a set of boxes. The union of these boxes is an overestimated target approximation $r_{target4}$.

$$r_{target4} = \left\{\text{expand}_{(Lh)}(s(y)) \mid y \in \text{grid}\,(h, r_{test})\right\} \tag{11}$$

$$= \mathrm{T}_4(r_{test}) \supseteq r_{target1} \tag{12}$$

All discussed target regions do not fit into the state space subdivisions used. Therefore, a second step is needed to extend these regions to legal sets of boxes. For example, region $R_{target1}$ (hatched areas in Figure 2) is given by the set of all boxes having contact with the target region $r_{target1}$. Fortunately, this operations is always an overestimation and does therefore not impact the correctness of the above results. Until now, only the third operation $T_3(\cdot)$ has been implemented. It has been shown that a number of 5-8 random test points are sufficient for the target box approximation.

Some additional steps are needed to optimize the connection relation for some corner cases. Namely, these are prevention of long successor-vectors, resulting in a box over-jump, boxes with self-connection and boxes with no connections to other boxes due to short successor-vectors. The last two conditions are unphysical if the box does not contain fix points (Section 3.2). As we have already mentioned in Section 3.1, no explicit time relations are considered. It might be useful or necessary in future implementations to store not only the connection relation $R \subseteq Q \times Q$ but rather this relation combined with the related transition time delays.

Until now, we have not discussed how to model the input values in the discrete extended state space. There are mainly two extreme assumptions for the transition behavior: Firstly, the input values do not change at all. That means, the model is build up as described before for several constant input values. There will be no connection between states with different input values. The second assumption is that the input value can change instantaneously over the whole input value range. A state space region has therefore not only connections to regions at the same input level but additional connections to the neighbor boxes in terms of input values. As we will see later, both of these input models are useful for certain conditions to be checked.

4. ANALOG CTL MODEL CHECKING

Given the subdivided state space and the connection relation, the continuous problem has been transfered into a discrete model. In this way, it is possibly treatable by digital model checking tools. However, these tools and also the model checking languages are not well suited for the generated models and the description of analog properties. In particular, the intensively used BDD structures are not helpful for this kind of models, because set of states can not efficiently be described by binary state variable combinations. Therefore, a modified model checker has been developed, based on the basic CTL algorithms described in [2]. There are some algorithmic modifications due to the special need of the analog model,

explained below. The language has been extended by a minimal set of operations enabling the work on analog models. Additionally, the results are visualized graphically. The meaning of the CTL operators is the same as in digital model checkers. The following table gives a syntax overview on the classical CTL language (13):

$$\phi := a \mid \phi \circ \phi \mid \neg\phi \mid \triangleright\diamond\phi \mid \triangleright\phi \,\mathsf{U}\, \phi \tag{13}$$

$$\phi := b * v \mid \phi \circ \phi \mid \neg\phi \mid \triangleright\diamond\phi \mid \triangleright\phi \,\mathsf{U}\, \phi \tag{14}$$

a	boolean variable			
b	continuous variable			
v	real value			
$*$	analog operators	$>$	$\rightarrow$	greater
		$<$	$\rightarrow$	smaller
$\circ$	boolean operators	$\vee$	$\rightarrow$	or
		$\wedge$	$\rightarrow$	and
$\neg$		$\neg$	$\rightarrow$	not
$\triangleright$	path quantifiers	E	$\rightarrow$	on some path
		A	$\rightarrow$	on all paths
$\diamond$	temporal operators	X	$\rightarrow$	next-time
		F	$\rightarrow$	eventually
		G	$\rightarrow$	always
U		U	$\rightarrow$	until

For example the formula $\Theta = \mathsf{AF}(state_1)$ can be read as follows: All paths starting in a state within Θ will eventually reach a state in which $state_1$ is true. It is obvious, that this language is not sufficient for analog model checking, since it is not possible to describe sets of states in a continuous state space. Therefore, the language has been expanded (14) by operators to describe half planes in the state space, e.g. $(x_1 > -13.2546)$. In combination with the boolean operators this enables the definition of arbitrary Manhattan polytopes. If the threshold values used in the CTL formula are not already subdivision values in the state space, they have to be added before executing the CTL formula. This makes the discrete model not only depend on the analog system but also on the CTL formula used, which is another difference to digital model checkers.
Some CTL operations are not clearly defined for analog systems. For example, the time quantifier "X - at the next time step" depends on the time step Δt used. As we have seen in Section 3.1 the time step is not always the same for the whole state space. Therefore, this operation is not reasonable in analog model checking. The other operations can be used in the same manner as in digital.
Obviously, this language is not very powerful describing analog design specifications. This has two reasons. Firstly, we want to follow the digital model checking ideas as far as possible without major changes, secondly, useful or necessary expansions have not been applied yet in the prototype tool. However, it will be shown below that it is possible to check some analog properties even with this minimal set of operations.

4.1 Border Problems

Due to the restricted state space, the border boxes get a special status, because at least one box face is connected to the outside area. Since this area is not considered in the system model, there are no connections from or to the outside area in the connection relation, even if there should be some in the none restricted case. During the CTL evaluation, the border boxes will behave incorrectly.
Consider for example differential equation (15) and CTL formula (16) in the restricted state space ($x = ([-5 .. 5], [-5 .. 5])$).

$$\{\dot{x}_1 = 1, \dot{x}_2 = 0\} \tag{15}$$

$$EG((x_2 > 1.0)\ \&\ (x_2 < 2.0)) \tag{16}$$

Theoretically, the region defined in (16) is not restricted in dimension x_1, but in our case it is, due to the given state space area. The theoretical result of (16) is $((x_2 > 1.0)\ \&\ (x_2 < 2.0))$ whereas we get $\emptyset$ in the restricted case, because some border boxes have no successor box. To avoid these problems, the border boxes are treated specially, depending on the CTL formula. As a result, the border boxes are not part of the model checking result and have to be omitted during interpretation.

4.2 Experimental Results

Three small nonlinear examples are used to show the capability of the proposed tool. The first one in a simple Schmitt trigger circuit. Secondly, a Biquad lowpass filter including two state variables is shown and the last example is a tunnel diode oscillator.

4.2.1 Schmitt Trigger Example

The inverting Schmitt trigger consists of an opamp behavioral model, two resistors, and an output capacitance shown in Figure 3. The opamp has an open loop gain of 10000. The output voltage restriction is ± 5 V and the maximum output current is 80 mA.

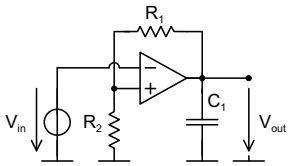

Figure 3: Schmitt trigger circuit.

Resistors R_1 and R_2 are both set to 10 kΩ. Thus, the switching threshold is about ± 2.5 V and the output voltage varies between +5 V and -5 V. This circuit has only one state, namely, the output voltage. Additionally, it has one input signal V_{in}. To consider all states that might occur in the circuit, the extended state space is chosen to be $V_{out} = [-7.7 .. 7.7]$ and $V_{in} = [-7.7 .. 7.7]$. The most interesting feature of the Schmitt trigger function are the switching properties for one output state to the other. Formulating this by CTL results in: $\Phi_1 = \mathsf{EF}(V_{out} < -4.5)$. Φ_1 is the set of states in which a path exists that will eventually reach the region $V_{out} < -4.5$. We choose the constant input value model for this calculation. The collection of boxes fulfilling this condition is shown in Figure 4 in light gray.

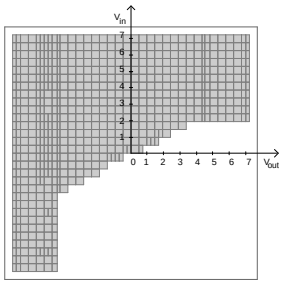

Figure 4: Schmitt trigger model checking result.

It is clearly seen, that above $V_{in} \simeq 2$V the circuit always switches to the negative output state. Below this point the switching depends on the output state V_{out}, but since this will be either 5 V or -5 V there will occur no switching in this region. An equivalent formula can be applied to find the positive switching conditions.

For analog designers such graphical output might be useful. In general however, one would expect only *true* or *false* as output for a CTL formula. To derive this, the checking condition can be expanded by an additional statement. For example

$$\Phi_2 = \mathsf{EF}(V_{out} < -4.5)\ \&\ ((V_{out} > -1)\ \&\ (V_{in} < -1)). \quad (17)$$

In this case the output is an empty set, which means there is no path form region $((V_{out} > -1)\ \&\ (V_{in} < -1))$ to the negative output state. CTL formula (17) is *false* for the whole state space. Every CTL formula can be changed in the same manner to get a binary result. However, for this contribution we prefer graphical results since they gives more insight into the algorithms.

4.2.2 Biquad Lowpass Filter

The second example is a 2^{nd} order Biquad lowpass filter, shown in Figure 5. The opamp model is the same as in the Schmitt trigger example but the maximum output voltage range is restricted to ± 1.5 V. This circuit has two state variables, namely the capacitor voltages V_{c1} and V_{c2}. Using a charge oriented capacitor model will normally lead to the two charges as state variables. We changed this, because it seems us more convenient to think in voltages than in charges. The corner frequency ω_c and the damping factor d are given by Equations (18) and (19).

$$\omega_c = \frac{1}{\sqrt{R_1 R_2 C_1 C_2}} \quad (18)$$

$$d = 0.5 C_1 \omega_c (R_1 + R_2) \quad (19)$$

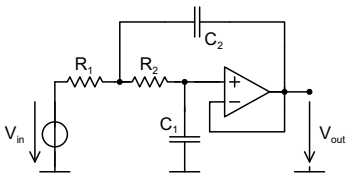

Figure 5: 2^{nd} order Biquad lowpass filter.

We use two different value sets for the resistors and capacitors, one with $\omega_c = 100\ s^{-1}$ and $d = 0.5$ and the second at the same frequency but with $d = 2$. The property to be checked in this example is the occurrence of overshooting in the two filters. Since these properties should be proved for arbitrary input signals, the appropriate input value model is chosen. The input signal range is $V_{in} = [-2\ ..\ 2]$, so that the nonlinearity due to the restricted output voltage will effect the system behavior. The state space is restricted to $V_{c1} = [-4\ ..\ 4]$ and $V_{c2} = [-2.5\ ..\ 2.5]$.
The initial state in this example is assumed to be the DC operating point at $V_{in} = 0$. The question is, which states are reachable from this point for arbitrary input signals within the given range? Directly transfered to CTL the question reads as Equation (20). Instead of a single start point a start area is used, that is for example a box surrounding the initial point. Next, operation EF is used to check which states have a path that will eventually reach this starting area. However, the direction of this operation is wrong. Inverting time (iv) gives the correct formula (Equation (20)).

$$\Phi_3 = \mathsf{iv}(\mathsf{EF}((V_{c2} < 0.5)\ \&\ (V_{c2} > -0.5)\ \&\ (V_{c1} < 0.5)\ \&\ (V_{c1} > -0.5)\ \&\ (V_{in} < 0.5)\ \&\ (V_{in} > -0.5))) \quad (20)$$

This equation is applied to both circuits. The results are shown in Figure 6 and Figure 7. The black box indicates the state space borders. The input voltage axis is perpendicular to the paper. As expected, in the highly damped circuit state V_{c1} remains within a range of ± 2 V whereas it reaches higher levels in the less damped case. It turns out that the opamp output restriction of ± 1.5 V does not have an impact on this result. This is because states V_{c1} and V_{c2} are not restricted by the opamp output.

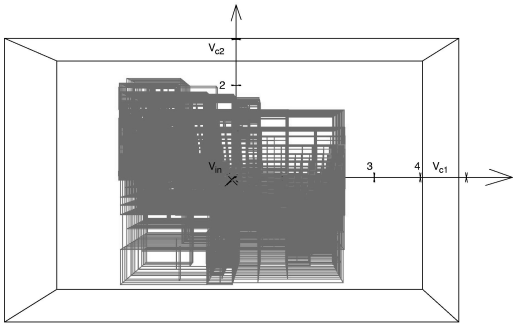

Figure 6: Result of Equation (20) for the highly damped circuit.

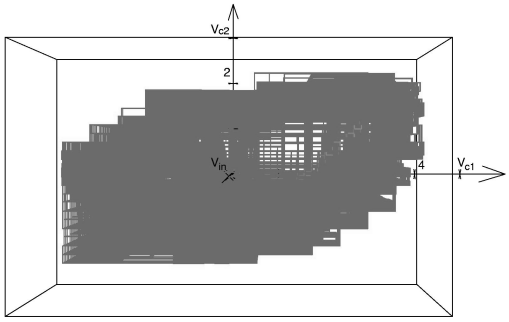

Figure 7: Result of Equation (20) for the less damped circuit.

To check the situation in more detail, one might be interested in the circuit's fix points. This is achieved by formula $\Phi_4 = \mathsf{iv}(\mathsf{EG}(1))$, where 1 denotes the whole state space.
To achieve the requested result, the input model has to be changed to constant input values, because it does not make sense to look for fix points while the input value may change arbitrarily. The graphical result is shown in Figure 8. The turning point due to the output voltage saturation is clearly visible.

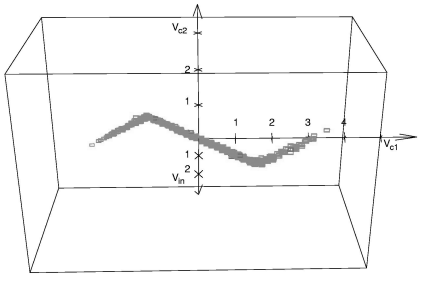

Figure 8: Fix points in the highly damped lowpass circuit.

The calculation of the discrete model in this example takes about three hours on a Sun ULTRA-II 300 MHz. Once the model is generated the calculation of CTL formulas is quite fast. It takes only seconds or a few minutes. Most of the runtime is spent in the numerical equation solver. Until now, a self written Newton solver

is used, so that a large improvement can be expected by using a commercial analog simulator for this task.
However, due to the n-dimensional subdivision algorithm, the total runtime complexity is exponential in the number of state variables, which is a serious problem for all state space exploration algorithms. Exponential runtime complexities are also known from digital model checking tools, but the use of efficient algorithms and data structures (e.g. BDDs) has extended the applicability for a wide range of circuits. Since similar techniques have not been utilized in the presented tool, a successful use of analog model checking in the future is likely, especially because the size of analog parts in todays designs is small compared to digital modules.

4.3 Tunnel Diode Oscillator

The analog system used in our third example is a simple tunnel diode oscillator circuit shown in Figure 9. The input voltage V_{in} is set to 2.6 V. In this operating point the circuit starts an oscillation automatically. The bounded state space is given by $V_C = [-0.2 \,..\, 4.4]$ and $I_L = [-0.2 \,..\, 4.0]$.

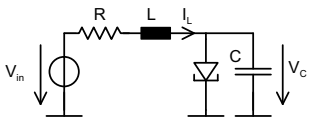

Figure 9: Tunnel diode oscillator circuit.

In a digital system a stable oscillation might be proved by $\mathsf{AG}(\mathsf{AF}(state_1))$ & $\mathsf{AG}(\mathsf{AF}(\overline{state_1}))$ [9]. This means: "On every computational path $state_1$ is *true* at some future time ($\mathsf{AF}(state_1)$). This condition is true for all future time ($\mathsf{AG}(\cdot)$). Simultaneously, this expression holds for $\overline{state_1}$". By this, the expression states that $state_1$ switches between *true* and $false$ for all future time. The model checking algorithm will find all points in the state space which fulfill this condition. We choose the same expression for the analog verification tool, except of the inner terms. They are replaced by a translation to the analog world, that is, the current I_L is larger or smaller than some value.

$$\Phi_5 = \{\mathsf{AG}(\mathsf{AF}(I_L > 2.2)) \,\&\, \mathsf{AG}(\mathsf{AF}(I_L < 1.6))\} \qquad (21)$$

The collection of boxes fulfilling this condition is shown in Figure 10 in light gray. Except of some border boxes and the middle region, the whole state space fulfills formula (21).

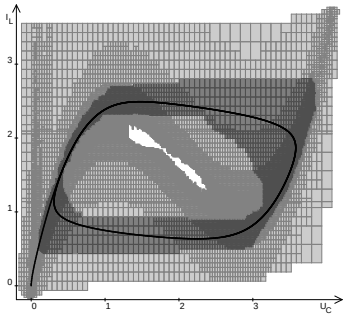

Figure 10: Model checking results Φ_5 and Φ_6.

We can conclude that nearly the whole plane will float into an stable orbit. The next question might concern the possible orbit geometry. We generate this by applying $\Phi_6 = \mathsf{iv}(\mathsf{EG}(\Phi_5))$;. By inverting the time, we exclude all points floating out of the orbit region. Since the orbit is stable, there is always a way out of the orbit at negative time. Therefore, operator EG is used instead of AG (resulting in an empty plane). The result Φ_6 contains the whole orbit calculated by an ordinary simulation (black line in Figure 10).

5. CONCLUSION

To apply digital model checking ideas to analog systems a discrete system model is needed. The main algorithmic task is to develop a state transition model in such a manner that the main nonlinear and dynamic properties of the analog system are retained. This is done using an automatic state space subdivision method and an algorithm developing the connection relation.
The implemented CTL model checker is based on digital algorithms, but it is extended by some operations, enabling the definition of analog properties in CTL. The capability of this language is shown by two experimental results. However, it is clear that the language is not sufficient for all analog properties. Especially, explicit time dependent properties, like slew rates or delays, are not covered until now.
The nonlinear examples show the use of the tool in analog design. However, applying CTL to analog systems, seems even more uncommon than doing so in digital. The result of CTL formulas can either be shown graphically or - even more formally - as a binary $true/false$ decision. The runtime is quite high for the discrete model generation, but there is a big optimization potential due to the prototype implementation.
As far as we know, the presented tool is the first approach to model checking for nonlinear analog systems. That opens a wide range of possibilities in applying formal methods not only to digital and hybrid systems but also to analog systems. Therefore, it is a step towards a more formalized analog design flow.

6. REFERENCES

[1] R. Alur, T. Henzinger, G. Lafferriere, and G. Pappas. Discrete abstractions of hybrid systems. *Proceedings of IEEE*, (88):971–984, 2000.

[2] J. Burch, E. Clarke, D. Long, K. McMillian, and D. Dill. Symbolic model checking for sequential circuit verification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(4):401–424, 1994.

[3] M. Dellnitz, G. Froyland, and O. Junge. The algorithms behind gaio - set oriented numerical methods for dynamical systems. In B. Fiedler, editor, *Ergodic Theory, Analysis, and Efficient Simulation of Dynamical Systems*, pages 145–174. Springer-Verlag, Berlin, 2001.

[4] M. Günther and U. Feldmann. Cad-based electric circuit modeling in industry, part i: Mathemetical structure and index of network equations. *Suveys on Mathematics for Industry*, 8(2):97–129, 1999.

[5] L. Hedrich and W. Hartong. Approaches to formal verification of analog circuits. In P. Wambacq, editor, *Low-Power Design Techniques and CAD Tools for Analog and RF Intergrated Circuits*, pages 155–191. Kluwer Academic Publishers, Boston, 2001.

[6] T. Henzinger and P.-H. Ho. Algorithmic analysis of nonlinear hybrid systems. *CAV '95: International Conference on Computer-Aided Verification, LNCS*, 939(7):225–238, 1995.

[7] R. Kurshan and K. McMillan. Analysis of digital circuits trough symbolic reduction. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10(11):1356–71, 1991.

[8] R. März. Numerical methods for differential algebraic equations. *Acta Numerica*, pages 141–198, 1991.

[9] K. McMillian. Symbolic model checking. *Kluwer Academic Publishers, Boston*, 1993.

[10] A. Neumaier. Interval methods for systems of equations. *Cambridge University Press, Cambridge*, 1990.

[11] A. Puri. Theory of hybrid systems and discrete event systems. *Dissertation, University of California ar Berkeley*, 1995.