Formal Verification of Analog Designs using MetiTarski

William Denman*, Behzad Akbarpour*, Sofiène Tahar* Mohamed H. Zaki‡ and Lawrence C. Paulson§
*Dept. of Electrical & Computer Engineering, Concordia University, Montréal, Canada, {w_denm, behzad, tahar}@ece.concordia.ca
‡Dept. of Computer Science, University of British Columbia, Vancouver, Canada, mzaki@cs.ubc.ca
§Computer Laboratory, University of Cambridge, England, larry.paulson@cl.cam.ac.uk

***Abstract*—MetiTarski, an automatic theorem prover for inequalities on real-valued elementary functions, can be used to verify properties of analog circuits. First, a closed form solution to the model of the circuit is obtained. We present two techniques for obtaining the closed form solution. One is based on piecewise linear modeling and the inverse Laplace transform. The other is based on small-signal analysis and transfer function theory. Second, the properties of interest are turned into a set of inequalities involving analytic functions, which are proved automatically using MetiTarski. We verify properties concerning oscillation and the change in gain due to component tolerances.**

I. Introduction

The verification of analog integrated circuits is time consuming and requires a great deal of expertise on part of the designer. Unlike digital designs, the behaviour of analog circuits varies over continuous electrical quantities. Therefore they are highly sensitive to factors including signal noise, temperature and component variation. In addition, higher order physical effects such as parasitics and current leakage arise when designing at the submicron level. With the constant demand of shorter time-to-market, the development of computer aided and automated tools for verifying analog designs is of great importance.

Traditionally, simulation is used to verify analog designs. However, because the state space search cannot be complete, simulation methods lack the rigor to ensure the correctness of the design. By contrast, formal methods can be used to verify a model completely. Unlike in the digital domain, scalable solutions for the automated and formal verification of analog circuits remain elusive. Promising abstraction based and model checking methods have been developed where properties can be checked and counter-examples automatically generated. In particular, theorem proving can deliver the highest level of assurance for verification: an explicit formal proof.

MetiTarski [1] is an automatic theorem prover for real-valued analytical functions, including the trigonometric and exponential functions. It works by a combination of resolution inference and algebraic simplification, invoking a decision procedure (QEPCAD) [2] to prove polynomial inequalities. Its axiomatic basis consists primarily of upper and lower bounds for the special functions, obtained from their power series or continued fraction expansions. The conjecture to be proved is transformed in stages, replacing occurrences of special functions by appropriate bounds. The general resolution procedure, aided by heuristics that isolate function occurrences, accomplishes this transformation. Proofs are typically found in a few seconds [3]. MetiTarski outputs machine-readable resolution proofs, which include algebraic simplification and decision procedure calls in addition to the familiar resolution rules. These proofs, which can be checked separately, provide hard evidence for the correctness of the results.

In the last decade a new engineering field called hybrid systems has emerged. It encompasses techniques for the automatic design and analysis of systems with real-time and continuous behavior. Much work has thus been conducted on the formal verification of hybrid systems. A hybrid system can be viewed as the mathematical model of an analog circuit, which is essentially a set of differential algebraic equations. Formal methods are now a serious candidate for the verification of analog systems. In analog circuit verification, one is interested in properties connected to the dynamic behavior of the system. We are interested in properties such as: *"Will the circuit oscillate for a given set of parameters?"* and *"For all sets of constant input voltages, will switching occur in less than a specific amount of time?"*.

We demonstrate in this paper a methodology for the automatic verification of functional properties of analog designs using MetiTarski. We apply the verification methodology on two examples including a tunnel diode oscillator and an operational amplifier. The rest of the paper is organized as follows: We start with an overview of the relevant work in Sect. II. Then describe the internals of MetiTarski in Sect. III. After that we describe the verification methodology in Sect. IV. This is followed by the application examples in Sect. V. The results are shown in Sect. VI before concluding the paper with Sect. VII.

II. Related Work

The verification of analog circuits started with the work on developing finite-state discrete abstractions for computing reachability relations. Unfortunately, these methods are time bounded and computationally expensive. Greenstreet and Mitchell [4] attempted to overcome these limitations by discretizing the state space by incorporating projection techniques on the state variables. This introduces larger overapproximations but makes the verification more tractable. This allowed circuits with a large state space to be verified using reachability analysis. These ideas inspired later work as in the model checking tools d/dt [5], Checkmate [6] and PHaver [7] and were respectively used in the verification of a biquad low-pass filter, a tunnel diode oscillator and a voltage controlled oscillator. Unfortunately, these three tools still rely on the use of time bounded reachability algorithms.

Another track of work has been conducted on qualitative based methods for the construction and verification of abstract models, which overcomes the time bound requirement of the reachability methods. In [8], the authors used HybridSAL [9] to generate an abstract model of several analog oscillators. Symbolic model checking was then used to prove safety properties on the generated abstract state space. The difficulty in particular with this method is that the generation of the predicates that define the abstract model is nontrivial. Human intervention is required to choose the useful and correct ones. Additionally, the abstraction can cause spurious counterexamples to be generated even if the circuit's behaviour is correct.

Our concern is the automated verification of analog circuits using deductive methods. In an early attempt at using theorem proving for the formal verification of synthesized analog circuits, Ghosh and Vermuri [10] proved the equivalence of analog designs that contain linear components and components with behaviour that can be represented by piecewise-linear (PWL) models. The PVS higher-order logic theorem prover is then used to prove the implication between implementations and behavioural specifications built in VHDL-AMS.

In similar work with theorem provers, Hanna [11] uses formal logic to define the behaviour of predicates over voltage and current waveforms. The basic behaviour of components such as resistors, power supplies and transistors are defined and then used to verify the behaviour of a NOT gate.

These early attempts are mostly based around heuristics for constructing the circuit component models and for determining the specification of the observed behaviour. Due to the underlying higher-order logic, they cannot be automated and are therefore not suited for larger applications. The methodology we present in this paper can be entirely automated and therefore could be applied to more than just basic academic problems. For information about the state of analog and mixed-signal verification, see the survey article [12].

III. METITARSKI: AN AUTOMATIC PROVER FOR ELEMENTARY FUNCTIONS

MetiTarski is an automated theorem prover for real-valued special functions such as arctan, log, exp, sin, cos and sqrt. It consists of a resolution theorem prover (Metis) combined with a decision procedure (QEPCAD) for the theory of real-closed fields. Resolution works with clauses, which are typically disjunctions of inequalities, and the decision procedure assists resolution by deleting from a clause any inequalities that it finds to be inconsistent with known facts or assumptions. Deleting a literal makes progress because the aim of resolution is to generate the empty clause, which represents contradiction.

MetiTarski further depends upon being supplied with axioms approximating the functions of interest with upper or lower bounds. These approximations could be polynomials, ratios of polynomials or expressions involving other functions. For example, one axiom asserts that

$$-(x^3 + 12x^2 + 60x + 120)/(x^3 - 12x^2 + 60x - 120)$$

is an upper bound for exp(x) provided $0 \leq x \leq 4$. Each axiom will give a good approximation for some part of the real line, but typically several axioms are needed to solve a problem. Other axioms allow division (which QEPCAD does not accept) to be replaced by multiplication. The resolution proof procedure automatically tries various combinations until it is successful. A failing proof typically runs forever, though in some cases MetiTarski recognizes that no proof exists and halts with an appropriate message.

Competing methods [13], [14] typically use a combination of constraint programming and interval arithmetic. They are often powerful, but have their own limitations. They do not return proof certificates, and they require all variables to be bounded by finite intervals. They can also run forever under certain circumstances.

IV. VERIFICATION METHODOLOGY

The methodology we follow to verify analog properties is shown in Fig. 1. Starting with an analog circuit, a functional property describing some required behaviour is chosen. Using the computer algebra system Maple [15], the behavioural model of the circuit is transformed into a closed form. The property is then combined with the closed form solution and manually transformed into an inequality. The resulting expression is then processed by MetiTarski which automatically generates a proof if it can determine that the inequality holds. This resulting proof indicates that the property is true.

If MetiTarski is successful, it delivers a proof and we are done. If unsuccessful, it will run until terminated by the user. Additional axioms are then added or removed to aid MetiTarski in formulating a proof. There are certain axioms

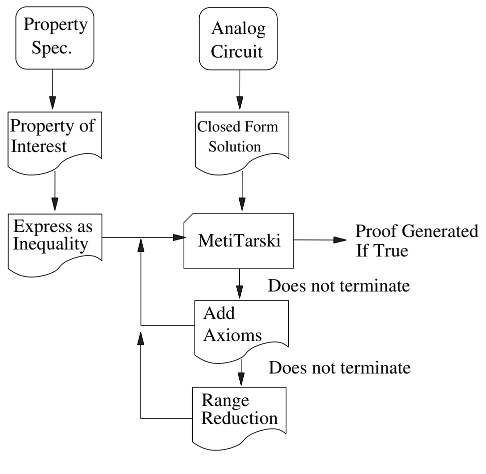

Fig. 1. MetiTarski Verification Methodology

that are available for special functions that take on extreme values. Including them unnecessarily in proofs will increase the computation time.

If still unsuccessful, an attempt at applying basic range reduction is made to the trigonometric functions to further eliminate any extreme values that can cause problems for MetiTarski's decision procedure.

There are two main difficulties: one is obtaining the closed form solution and the other is transforming the property into an inequality. The choice of property governs whether the closed form solution should be dependent on time or frequency. As well, the number and type of components in the design determine the required solution generation method.

Two closed form solution generation methods are presented and they both rely on some amount of linearization. One is based on converting the nonlinear behaviour of a component into a piecewise linear approximation. The other, similar in nature to the first, is based on linearizing the entire circuit at its DC-operating point. These methods inherently introduce some degree of error to the verification problem. In this work, we assume that the linearization is valid in the chosen neighbourhood.

A. Obtaining the Closed Form Solution: Piecewise Linearization

To obtain a closed form solution for nonlinear components, we follow the method described in Fig. 2. The idea is to separate the behaviour of the circuit in terms of its discrete "modes", such as the oscillation modes of an oscillator. Over each mode, the circuit will operate according to a different mathematical relation.

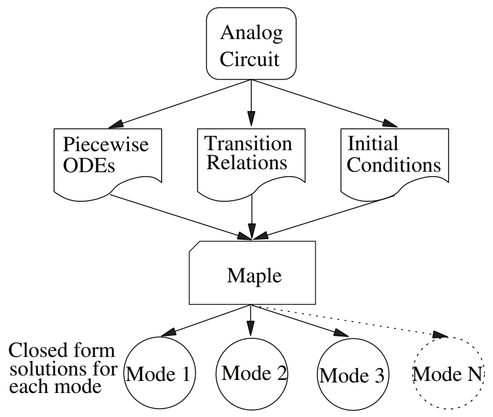

Fig. 2. Generation of the Closed Form Solution of each Mode

We first obtain the system of differential equations from the circuit of interest. Any nonlinear elements are transformed into an approximated PWL model. Due to this, there will be a certain amount of error introduced at this stage. The degree of error is set by choosing the number of segments in the PWL model. The higher number of segments, the more precise the model will be, but with an increased computing cost. Even though precision is lost with this transformation, we defend our modeling choice for the following reasons [16]:

- Piecewise-linear circuits are the simplest class of nonlinear circuits.
- The behaviour of many op-amp and diodes and switch circuits can be reasonably approximated as piecewise-linear.
- Linear methods are substantially more tractable than nonlinear ones, even when they divide the problem into multiple modes.

The transition relation between each mode of the PWL model is determined and ordinary differential equations (ODEs) are constructed over each piecewise segment. The work performed by the Maple computer algebra system is shown in Fig. 3. Starting in any mode, the ODEs and initial conditions are supplied to Maple's inverse Laplace transform routine (*invlaplace*) to calculate a closed form solution for each state variable as a function of time. Using the transition relations, the numerical solver (*fsolve*) determines the exact time where the system switches modes. At that time instance, the initial conditions for the next mode are then evaluated (*eval*) and the inverse Laplace transform is performed again to find the closed form solution. This is repeated until each mode of the model has been visited.

Note. We take the results of Maple as being correct even though no formal proof of its transformations is produced.

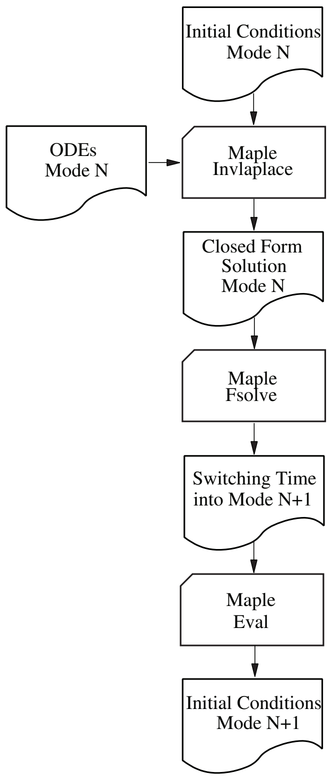
Initial Conditions Mode N
ODEs Mode N
Maple Invlaplace
Closed Form Solution Mode N
Maple Fsolve
Switching Time into Mode N+1
Maple Eval
Initial Conditions Mode N+1

Fig. 3. Determining the Closed Form Solutions for Each Mode

B. Obtaining the Closed Form Solution : Linearization at the DC Operating Point

In the first method, we were concerned with separating the modes of operation of a single nonlinear component. This method works well when dealing with a design that contains components that operate equally over each mode of operation. When the number of nonlinear circuit elements increases, the amount of work required to keep track of the states and the transitions becomes increasingly difficult. One simplification is to assume that the components only operate over a single mode and are centered at a single voltage (DC operating point). This is the case with a transistor that operates linearly in its saturation mode of operation. By assuming that a small AC signal is superimposed on top of the DC signal, it is possible to use *small signal* analysis for calculating the closed form solution. Linearizing the entire circuit at a DC operating point greatly simplifies the generation of a single closed form solution for designs with several nonlinear components.

To obtain a closed form solution we follow the method described in Fig. 4. The circuit is linearized at its operating point. Then using circuit analysis, a transfer function that relates the input to the output is extracted from the simplified model. This method is particularly useful for generating a closed form solution that is dependent on frequency.

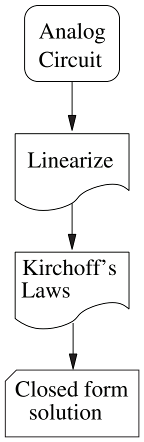

Fig. 4. Generation of the Parametric Based Closed Form Solution

C. Property Transformation

The next step is to turn the verification property into an inequality over special functions, as shown in Fig. 1. A first-order formula in the Thousands of Problems for Theorem Provers (TPTP) format, including the corresponding axioms, is then supplied to MetiTarski. MetiTarski uses an extension of the TPTP format, including infix notation for the arithmetic and relational symbols [17], [18].

There exist advanced methods to automatically extract ODEs from a circuit description. In our previous work [8], we used the Dymola modeling framework to extract simplified ODEs from a SPICE netlist. Chua and Deng [19] provide an automated method to generate the PWL model of certain op-amps, operational transconductance amplifiers and diodes. The work done with Maple is interactive and could easily be automated. The verification performed by MetiTarski is entirely automated. To show the feasibility of the proposed methodology we have applied it on several standard analog designs, two of which we present next.

V. Applications

In this section, we will describe the application of our methodology to an analog oscillator and an Operational Amplifier (Op-Amp). Oscillators play a critical role in many communication systems, in particularly for generating a periodic signal needed for the frequency translation between carriers. The tunnel diode oscillator has been previously used in [6], [20], as a benchmark for analog formal verification techniques and thus serves as an appropriate example for demonstrating our methodology. Amplifiers are the most basic component in analog circuits, which are used to control and manipulate the currents and voltages to achieve the required specifications. One of the issues with verifying such circuits is that their operation is highly dependent on process variations and therefore require many lengthy simulations.

A. Tunnel Diode Oscillator

The tunnel diode oscillator shown in Fig. 5 demonstrates the effect of resonant tunneling that causes a negative resistance to appear at small forward bias voltages as shown in Fig. 6. Essentially, for some range of voltages the current through the tunnel diode decreases with increasing voltage. This negative resistance can be used to create a reliable oscillator that functions under many different operating conditions. We intend to verify that for certain initial states and component values, the tunnel diode oscillator will not oscillate. By verifying this property, we will be able to eliminate designs that do not work.

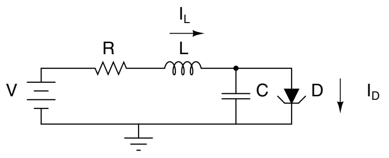

Fig. 5. Tunnel Diode Oscillator

Circuit analysis is used to determine the differential equations of the circuit. They are defined as

$$\dot{V_C} = \frac{1}{C}(-I_D(V_C) + I_L)$$
$$\dot{I_L} = \frac{1}{L}(-V_C - R \times I_L + V_{in})$$

where $I_D(V_C)$ is a PWL model that has three modes of operation. Taking E_1 and E_2 to represent the voltages where the model switches modes and G_0, G_1 and G_2 to represent the separate contributions to the slope of the best fit curve in each mode, we can define the PWL model [16] of the tunnel diode as

$$I_D(V_C) = -\frac{1}{2}(G_1E_1 + G_2E_2) + (G_0 + \frac{1}{2}G_1 + \frac{1}{2}G_2)V_D + \frac{1}{2}G_1\,|V_C - E_1| + \frac{1}{2}G_2\,|V_C - E_2|$$

Fig. 6 shows the real continuous behaviour of the tunnel diode as well as the PWL approximation. From the graph, the linearized variables are: in region 1, $g_1 = G_0$. In region 2, $g_2 = G_0{+}G_1$. In region 3, $g_3 = G_0{+}G_1{+}G_2$. In our example, $G_0 = 0.2616$, $G_1 = -0.3608$, $G_2 = 0.3591$, $E_1 = 0.276$ and $E_2 = 0.723$ giving

$$I_D(V_C) = \begin{cases} 0.2616V_C & \text{if } V_C < 0.276 \\ -0.0992V_C + 0.0997 & \text{if } 0.276 < V_C < 0.723 \\ 0.2599V_C - 0.1599 & \text{if } V_C > 0.723. \end{cases}$$

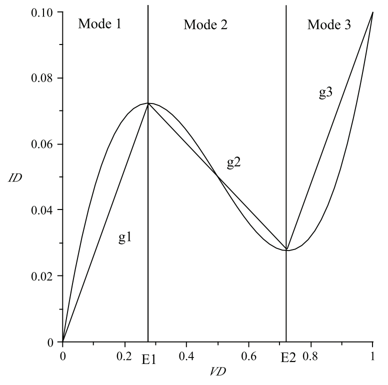

Fig. 6. Tunnel Diode Current Linearization

The system is now completely specified. Each mode is defined by a set of ODEs and switching constraints. The resulting time-deterministic hybrid model can be illustrated as an FSM as shown in Fig. 7. Each mode of operation is represented by a state circle and the switching constraints are indicated above each directional arrow.

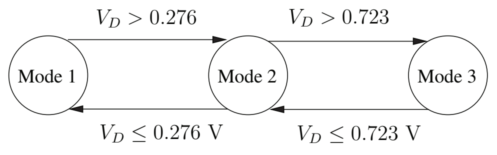

Fig. 7. The Hybrid Model of the Tunnel Diode Current

Suppose the parameter values are $R = 50\ \Omega$, $L = 10^{-6}$ H, $C = 10^{-9}$ F and $V = 0.3$ V, the dynamics of Mode 3 can be written as the first-order linear differential system $\dot{x} = Ax + B$, where the A matrix represents the coefficients of the state variables and the B matrix represents the constants.

$$x = \begin{bmatrix} I_L \\ V_C \end{bmatrix}, A = \begin{bmatrix} -3 \times 10^5 & -10^6 \\ 10^9 & -2.621 \times 10^8 \end{bmatrix}$$

$$B = \begin{bmatrix} 3 \times 10^5 \\ 0 \end{bmatrix}$$

Let X denote the Laplace transform of x (X = $\mathcal{L}x$); then $sX - x_0 = AX + \frac{B}{s}$, and solving for X we have $X = (sI - A)^{-1}(x_0) + \frac{B}{s}$. With the initial states as x_0 = (0.025,0.74)$^{\mathrm{T}}$ (a transposed matrix) and using Maple we construct the matrix as shown in Fig. 8.

The closed form solutions of the state variables are obtained by taking the inverse Laplace transform $\mathcal{L}^{-1}X$ and we obtain

$$V_C(t) = 0.116e^{-2.58\times10^8 t} + 0.278 - 0.262e^{-4.19\times10^6 t}$$
$$I_L(t) = 0.448 \times 10^{-3}e^{-2.58\times10^8 t} + 0.0727 - 0.0677e^{-4.19\times10^6 t}$$

Now we have the state space representation of the system for Mode 3 in Fig. 6. The next step is to determine the time when the tunnel diode switches from Mode 3 to Mode 2. By using Maple, we determine that the condition $V_C \leq 0.723$ is true at $t = 2.38 \times 10^{-9}$ s. The values of both V_C and I_L are evaluated at this time. We then use these values for x_0 and again repeat the process of finding the matrix X, and taking its inverse Laplace transform. This is repeated as shown in Fig. 3 until we have visited each mode and have generated the closed form solutions for the two state variables.

For Mode 2 in Fig. 6, the closed form solutions are

$$V_C(t) = 0.278 + 0.0025e^{8.79\times10^7 t} - 0.0045e^{-1.10\times10^7 t}$$
$$I_L(t) = 0.0727 + 0.00039e^{-1.10\times10^7 t} - 0.000028e^{8.79\times10^7 t}$$

For Mode 1 in Fig. 6, the closed form solutions are

$$V_C(t) = 0.323 - 01.64e^{-2.56\times10^8 t} + 0.56e^{-4.21\times10^6 t}$$
$$I_L(t) = -0.076 - 0.00064e^{-2.56\times10^8 t} + 0.144e^{-4.21\times10^6 t}$$

To demonstrate the power of MetiTarski, we seek to define an oscillation property that can be proved over all modes of operation. One such property is *"For a set of initial conditions, the circuit will not oscillate"*. In this example we focus on the current through the inductor. When the Tunnel

$$X = \begin{bmatrix} \dfrac{(s + 0.262 \times 10^9)(0.55 \times 10^{-2} + \dfrac{0.300 \times 10^6}{s})}{(s^2 + 0.262 \times 10^9 s + 0.108 \times 10^{16})} - \dfrac{0.131 \times 10^6}{(s^2 + 0.262 \times 10^9 s + 0.108 \times 10^{16})} \\ \dfrac{(0.550 \times 10^7 + \dfrac{0.300 \times 10^{15}}{s})}{(s^2 + 0.262 \times 10^9 s + 0.108 \times 10^{16})} + \dfrac{(0.131s + 0.393 \times 10^5)}{(s^2 + 0.262 \times 10^9 s + 0.108 \times 10^{16}))} \end{bmatrix}$$

Fig. 8. Tunnel Diode Matrix

Diode oscillates, the current through the inductor will range between a set of values. A necessary condition for oscillation is that the current pass some threshold. Since this requirement is not sufficient for oscillation to occur we must focus on non-oscillation. If we choose an initial point and the current does not exceed the threshold, then we can conclude that the circuit does not oscillate. This property can be more exactly defined as *"For all time and all possible paths, the current through the inductor will never pass some upper or lower bound"*. For example, when the upper bound is 0.03, the property can be expressed as:

Property 1: $[I_L \leq 0.03]$

The first order formula we supply to MetiTarski is in its TPTP-syntax. For example, to prove that in mode 1, I_L is always less than 0.03 we use the following

```
fof(
Tunnel,conjecture, ! [X] :
((0 <= X & X <= 2.39*10^(-9)) =>
 (- 0.076 - 0.00064*exp(-2.56*10^8*X)
  + 0.144*exp(-4.21*10^6*X))
  < 0.03)).
```

where *'fof'* indicates to MetiTarski that the logic language used is a first-order formula. It is then followed by a label of the proof as well as the keyword *'conjecture'* indicating that the following formula is to be proved with the included axioms. The conjecture is read as follows: For all (represented by "!") X between 0 and 2.39×10^{-9} the formula is always less than 0.03.

Now suppose we choose the component values $R = 0.3\,\Omega$, $L = 10^{-6}$ H, $C = 10^{-9}$ F, $V = 0.3$ V. Using the same inverse Laplace transform methodology, we get the closed form solutions of the state variables. The property of interest is now: *For a set of initial conditions, the trajectory of the oscillation reaches a final set and remains bounded* [21]. The variables of the circuit that oscillate are V_C and I_L. This can be described formally as:

Property 2: $[V_C > 0 \wedge V_C < 0.9 \wedge I_L > 0 \wedge I_L < 0.08]$

MetiTarski proves both properties over the three modes of operation. For property 1, it is proved that the circuit does not oscillate. For property 2, it is proved that the oscillation present in the circuit is bounded. Complete runtime results of this example can be found in Tables I and II.

B. Operational Amplifier

In this final example, a frequency domain property of a CMOS Operational Amplifier will be analyzed and verified. The Op-Amp is a popular device because of its versatility [22]. It is a fundamental building block of many designs including differential amplifiers, integrators, differentiators and digital to analog converters. One characteristic that makes verification of Op-Amps a simpler task is that its behaviour approaches the idealized model under certain operating conditions.

The analysis of the frequency domain is important since an input signal is usually not constrained to a single frequency. The performance of a device will behave differently at high frequencies. Consider the circuit in Fig. 9, as the frequency of the input signal increases, there will be a point where the gain drops below a specified level.

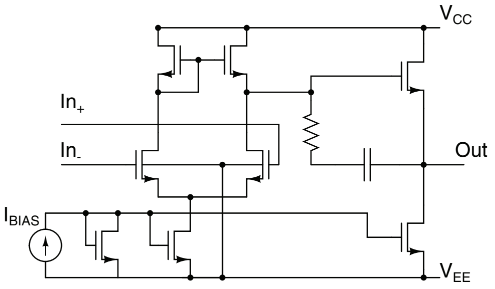

Fig. 9. Operational Amplifier [23]

To begin verification, the circuit is first linearized at its operating point and then using nodal analysis (Kirchhoff's current and voltage laws), the following transfer function is extracted

$$H(s) = \frac{A_0 e^{\Delta\Phi}}{1 + A_0 \frac{\sin(phm)}{gbw 2\pi} s + A_0 \frac{\cos(phm)-1}{gbw^2 4\pi^2} s^2} [23]$$

where phm represents the phase margin, gbw the gain bandwidth, $\Delta\Phi$ the phase tolerance and A_0 the closed loop gain. The phase margin is an indicator of amplifier stability. The phase tolerance represents the change in phase from input to output. The closed loop gain represents the gain of the Op-Amp when connected in a feedback configuration. What we would like to determine is that over a certain range of parameter values, does the gain of the circuit remain above some minimum value.

By taking the absolute value or magnitude of $H(s)$, a closed form solution for the gain of the circuit is obtained. With the closed loop gain chosen to be 93 dB and the gain bandwidth to be 5 MHz, the gain is now characterized by the equation in Fig. 10.

From the specification of the circuit [23], we are given that in the frequency range of 100 to 120 Hz the gain of the circuit should be greater than 57000 and this can be expressed as:

$$|H(jw)| = \frac{5.9 \times 10^{19}}{\sqrt{\frac{10^{29}}{0.1} - \frac{10^9 w^4 \cos(phm)}{0.28} - \frac{10^5 w^4 \cos(phm)}{0.83} - \frac{10^{20} w^2 \cos(phm)}{0.83} + w^4 + \frac{10^{24} w^2}{0.28} - \frac{10^{24} w^2 \cos(phm)^2}{0.28}}}$$

Fig. 10. Gain of the Operational Amplifier

Property 3:

$[Freqs.100$ to 120 Hz, phm 45 to 60 Deg : $|H(s)| > 57000]$

Using MetiTarski

```
fof(OPAMP,conjecture, ! [X,Y] :
((100 <= X & X <= 120 &
  PI/4 <= Y & Y <= PI/3) =>
 (5.9*10^19/
  (sqrt(9.7*10^29
   - 3.6*10^9*X^4*cos(Y)^2
   - 1.2*10^5*X^4*cos(Y)
   - 1.2*10^20*X^2*cos(Y)
   + X^4 + 3.6*10^24*X^2
   - 3.6*10^24*X^2*cos(Y)^2))
  > 5700))).
```

MetiTarski proves that the property holds over the entire frequency range. Specifically, that the gain of the circuit does not decrease below the required level. The runtime results are found in Table III.

Mode	Variable	Bound	CPU Time (sec.)
1	I_L	U	0.1
2	I_L	U	4.0
3	I_L	U	0.3

TABLE I
TUNNEL DIODE OSCILLATOR - PROPERTY 1 RESULTS

Mode	Variable	Bound	CPU Time (sec.)
1	V_C	U	0.2
1	V_C	L	0.4
2	V_C	U	2.7
2	V_C	L	0.6
3	V_C	U	0.3
3	V_C	L	0.5
1	I_L	U	0.5
1	I_L	L	0.3
2	I_L	U	0.6
2	I_L	L	3.9
3	I_L	U	0.3
3	I_L	L	0.6

TABLE II
TUNNEL DIODE OSCILLATOR - PROPERTY 2 RESULTS

Mode	Variable	Bound	CPU Time (sec.)
Saturation	$\|H(s)\|$	L	8.64

TABLE III
OPAMP - PROPERTY 3 RESULTS

VI. EXPERIMENTAL RESULTS

In the previous section we presented concrete examples of how, using the theorem prover MetiTarski, various analog circuit properties could be verified. The experimental results are presented in Tables I, II and III. In each table, the name of each experiment represents the mode (1, 2 or 3), variable under test (V_C or I_L) and either the upper (U) or lower (L) bound. For the tunnel diode oscillator we first proved in one case that oscillation is not present. The results show three experiments that indicate I_L never passes some upper bound in any mode. In the other case, it was necessary to conduct 12 experiments to prove that each of the three variables are bounded in each mode. The frequency dependent gain of an operational amplifier was also verified. The runtimes were measured on a 2.8 GHz Dual Quad-Core Mac Pro, with 4GB of RAM.

The experimental results indicate that it is possible to solve simple analog circuit verification problems using an automated theorem prover. We obtain formal proofs that can be inspected in order to increase our confidence that the design correctly matches its specification. Most of the experiments return in less than 5 seconds. For those that took longer, this is explained by the extreme values taken by the special functions of the closed form solutions. It is sometimes possible to perform range reduction to reduce the time that is necessary to complete the proof. Unfortunately, range reduction is not trivial to apply to trigonometric equations.

VII. CONCLUSION

First and foremost we have developed a viable methodology for the automated verification of analog designs. Starting with the system of equations model of the analog circuit, the closed form solutions of each mode of operation is generated using Maple. The closed form solutions are then passed to the MetiTarski theorem prover along with properties of interest defined in terms of inequalities. MetiTarski then generates a full proof of its claim of truth. Secondly, we have demonstrated that the methodology can be applied to a certain set of analog circuits. The tunnel diode oscillator analyzed in the paper has an interesting and complex behaviour that requires a high level of verification to ensure proper functionality. The results that we have obtained are promising and we are now interested in applying the methodology to different classes of circuits. The proofs we have obtained are performed quickly and this is an indication that our methodology could be scaled to more complicated problems.

To scale to larger problems, we will need to investigate efficient methods for analyzing nonlinear systems. Extensions to our work could include methods for analytically solving systems of polynomial nonlinear ordinary differential equations. One such method is the Prelle-Singer procedure [24], which is implemented in computer algebra systems such as REDUCE (the PSODE package [25]) and Maple (the PSsolver package [26]). Furthermore, the automation of the mechanical steps must be addressed. This will include an investigation on methods to automatically calculate the piecewise linear functions of nonlinear circuit elements and to automate the work performed by Maple.

We are quite motivated by the results of the work in this paper and further experimentation is ongoing. A necessary addition to the methodology would be to increase the precision of the PWL models by introducing an error bound. Indeed, we will need to apply MetiTarski to bigger and more complex examples, where a limiting factor is formulating the property of interest in terms of analytical functions.

REFERENCES

[1] B. Akbarpour and L. C. Paulson, "MetiTarski: An automatic prover for the elementary functions," in *Intelligent Computer Mathematics*, ser. LNCS, vol. 5114. Springer, 2008, pp. 217–231.

[2] C. W. Brown, "QEPCAD B: a program for computing with semi-algebraic sets using CADs," *SIGSAM Bulletin*, vol. 37, no. 4, pp. 97–108, 2003.

[3] B. Akbarpour and L. C. Paulson, "Applications of MetiTarski in the verification of control and hybrid systems," in *Hybrid Systems: Computation and Control*, ser. LNCS, vol. 5469. Springer, 2009, pp. 1–15.

[4] M. R. Greenstreet and I. Mitchell, "Reachability analysis using polygonal projections," in *Hybrid Systems: Computation and Control*, ser. LNCS, vol. 1569. Springer, 1999, pp. 103–116.

[5] T. Dang, A. Donze, and O. Maler, "Verification of analog and mixed-signal circuits using hybrid system techniques," in *Formal Methods in Computer-Aided Design*, ser. LNCS, vol. 3312. Springer, 2004, pp. 21–26.

[6] S. Gupta, B. H. Krogh, and R. A. Rutenbar, "Towards formal verification of analog designs," in *IEEE/ACM International Conference on Computer Aided Design*, 2004, pp. 210–217.

[7] G. Frehse, B. H. Krogh, and R. A. Rutenbar, "Verifying analog oscillator circuits using forward/backward abstraction refinement," in *Proc. IEEE/ACM Design, Automation and Test in Europe*, 2006, pp. 257–262.

[8] W. Denman, M. Zaki, and S. Tahar, "A bond graph approach for the constraint based verification of analog circuits," in *Workshop on the Formal Verification of Analog Circuits*, Jul. 2008.

[9] A. Tiwari, "HybridSal: A tool for abstracting hybridsal specifications to SAL specifications." http://sal.csl.sri.com/hybridsal/

[10] A. Ghosh, R. Vemuri, and D. R. Vemuri, "Formal verification of synthesized analog designs," in *IEEE International Conference on Computer Design*, 1999, pp. 40–45.

[11] K. Hanna, "Reasoning about analog-level implementations of digital systems," *Formal Methods in System Design*, vol. 16, no. 2, pp. 127–158, 2000.

[12] M. Zaki, S. Tahar, and G. Bois, "Formal verification of analog and mixed signal designs: A survey," *Microelectronics Journal*, vol. 32, no. 12, pp. 1395–1404, Dec. 2008.

[13] S. Ratschan and Z. She, "HSolver : Verification of hybrid systems based on the constraint solver RSolver." http://hsolver.sourceforge.net/

[14] M. Franzle and C. Herde, "HySAT : An efficient proof engine for bounded model checking of hybrid systems," *Formal Methods in System Design*, vol. 30, no. 3, pp. 179–198, Jun. 2007.

[15] "Maple 12 : The essential tool for mathematics and modelling." http://www.maplesoft.com/

[16] W. K. Chen, *The Circuits and Filters Handbook*. CRC Press LLC, New York, 2006.

[17] G. Sutcliffe and C. Suttner, "The TPTP Problem Library: CNF Release v1.2.1," *Journal of Automated Reasoning*, vol. 21, no. 2, pp. 177–203, 1998.

[18] ——, "The TPTP problem library for automated theorem proving," 2009. http://www.cs.miami.edu/~tptp/

[19] L. O. Chua and A.-C. Deng, "Canonical piecewise-linear analysis: Generalized brake point hopping algorithm," *International Journal of Circuit Theory and Applications*, vol. 14, no. 1, pp. 35–52, 1985.

[20] W. Hartong, K. Klausen, and L. Hedrich, "Formal verification for non-linear analog systems: Approaches to model and equivalence checking," in *Advanced Formal Verification*. Kluwer, 2004, pp. 205–245.

[21] G. Frehse, "PHAVer: Algorithmic verification of hybrid systems past HyTech," in *Hybrid Systems: Computation and Control*, ser. LNCS, vol. 3414. Springer, 2005, pp. 263–279.

[22] A. S. Sedra and K. C. Smith, *Microelectronic Circuits*. Oxford University Press, 2004.

[23] L. Hedrich and E. Barke, "A formal approach to verification of linear analog circuits with parameter tolerances," in *IEEE/ACM Design, Automation and Test in Europe*, 1998, pp. 649–654.

[24] M. S. M. Prelle, "Elementary first integrals of differential equations," *Transactions of the American Mathematical Society*, vol. 279, no. 1, pp. 215–229, Sep. 1983.

[25] Y. Man, "Computing closed form solutions of first order odes using the prelle-singer procedure," *Journal of Symbolic Computing*, vol. 16, no. 5, pp. 423–443, 1993.

[26] L. Duarte, S. Duarte, L. da Mota, and J. Skea., "An extension of the prelle- singer method and a maple implementation," *Computer Physics Communications*, vol. 144, no. 1, pp. 46–62, Mar. 2002.