Piecewise Linear Modeling of Nonlinear devices for Formal Verification of Analog Circuits

Yan Zhang, Sriram Sankaranarayanan and Fabio Somenzi.
University of Colorado, Boulder, CO. Email: {yan.zhang,srirams,fabio}@colorado.edu

***Abstract*—We consider different piecewise linear (PWL) models for nonlinear devices in the context of formal DC operating point and transient analyses of analog circuits. PWL models allow us to encode a verification problem as constraints in linear arithmetic, which can be solved efficiently using modern SMT solvers. Numerous approaches to piecewise linearization are possible, including piecewise constant, simplicial piecewise linearization and canonical piecewise linearization. We address the question of which PWL modeling approach is the most suitable for formal verification by experimentally evaluating the performance of various PWL models in terms of running time and accuracy for the DC operating point and transient analyses of several analog circuits. Our results are quite surprising: piecewise constant (PWC) models, the simplest approach, seem to be the most suitable in terms of the trade-off between modeling precision and the overall analysis time. Contrary to expectations, more sophisticated device models do not necessarily provide significant gains in accuracy, and may result in increased running time. We also present evidence suggesting that PWL models may not be suitable for certain transient analyses.**

I. Introduction

In this paper, we evaluate piecewise linear models (PWL) for the verification of nonlinear analog circuits. Analog circuits are indispensable in modern integrated circuits. Although, in a typical IC design, the analog circuitry occupies a small fraction of the entire die area, its design and verification requires considerable effort compared to its digital counterpart [1]. Because analog design is error-prone, many simulations, each of which may take several hours or even several days, are needed to convince the designers that the specification is met. Even with this effort, many designs still fail to work after fabrication. Therefore, formal verification techniques for analog circuits have recently emerged, as shown in Table I. Most efforts in formal verification have focused on two problems:

- *DC Operating Point Analysis:* Satisfiability solvers are used to characterize all operating points of a circuit. In many cases, SPICE simulations may miss metastable operating points that can be captured by a formal approach [2], [3].
- *Dynamic (Transient) Analysis:* Various models of dynamical systems, including ODEs [2], [4]–[6], hybrid automata [7], hybrid Petri nets [8] and frequency domain transfer functions [9] are used to study the evolution of a circuit in time.

This work was funded in part by the US National Science Foundation (NSF) under grants CNS-1016994 and CAREER grant CNS-0953941. All opinions expressed here are those of the authors and not necessarily of the NSF.

These problems are complex due to the presence of nonlinear devices such as diodes, transistors, non-Ohmic resistors and nonlinear capacitors. While solvers for reasoning about nonlinear systems are becoming sophisticated [10]–[13], their capabilities are far exceeded by the linear arithmetic SMT solvers such as MathSAT, Yices and Z3 [14]. As a result, the problem of approximating nonlinear devices by a piecewise linear model naturally presents itself. Fortunately, PWL modeling of analog devices has been well studied by the analog circuit simulation community [15]–[17]. In this regard, a wide variety of PWL modeling approaches for transistors have been considered, including simplicial piecewise linearization [16], canonical piecewise linearization [15] and their many refinements. Recent work by Tiwary et al. [2] uses simple piecewise constant (PWC) models with interval uncertainties to approximate the nonlinear characteristics of transistors, encoding DC operating point and transient analyses problems as linear arithmetic constraint satisfaction with promising results.

The key advantage of PWL models lies in their translation to linear arithmetic. On the other hand, the abstraction of transistor characteristics by PWL models can potentially miss DC operating points or transient behaviors, unless the modeling can be made "sound" as defined in Section II. However, a sound model may introduce spurious behaviors that do not exist in the real circuit. To address this issue, Tiwary et al. present a model refinement procedure that constructs a PWL model using a restricted input domain provided by results found from a coarse model.

Thus far, little work has been done to consider the "best" piecewise linear modeling approach for DC and transient analyses of analog circuits. In this paper, we ask the following question: is one modeling approach necessarily better than the other in terms of performance (time taken) vs. accuracy (fewer spurious DC operating points, fewer spurious paths)?

We compare three different PWL modeling approaches, including PWL modeling with simplicial decomposition, the canonical PWL function proposed by Chua et al. [15] and the PWC model used by Tiwary et al. [2] with different modeling parameters. Our comparisons are based on the DC operating point and transient analyses framework of Tiwary et al. [2]. Our comparisons consider the running times, number of SMT solver queries and the precision in terms of spurious results. Our findings are quite surprising: for DC operating point analysis, PWC models are more efficient in terms of performance while providing very little difference in terms

978-0-9835678-2-0/12/$31.00 ©2012 FMCAD Inc.

of precision. We also present evidence suggesting that PWL models may not be suitable for certain transient analyses.

In the next section, we discuss the device modeling approaches. In section III, we present the setup of the formal DC operating point analysis. Next, we present the formal transient analysis. Finally we discuss the experimental results.

II. Device Modeling Approaches

In this section, we consider the piecewise linear (PWL) modeling of nonlinear analog devices. We discuss the modeling problem based on simulation and introduce the notion of models that are "sound" with respect to data points.

A. Modeling Nonlinear Devices

A device model is a function $\mathbf{y} = \mathbf{F}(\mathbf{x}, \mathbf{p})$, where $\mathbf{y}$ represents the dependent variables, $\mathbf{x}$ the independent variables and $\mathbf{p}$ the device parameters that vary with the fabrication process, voltage and temperature (PVT). For example, a model for a NMOS transistor is

$$I_{DS} = F(V_{GS}, V_{DS}, \mathbf{p})\,.$$

In sophisticated device models, such as BSIM4 [1] and PTM [2], F is often complicated and expressed as C (FORTRAN) language subroutines which can be used by SPICE. In order to enable formal verification, we need to abstract F to a simpler, more tractable form.

Device Approximation. Approximations are achieved by means of relational models $R(V_{GS}, V_{DS}, \mathbf{p}, I_{DS})$, that relate possible voltages, parameters values and currents over some domain $\mathbb{D}$. The domain is defined by intervals for V_{GS} and V_{DS} which typically range from 0 to the supply voltage, and some range of parameter uncertainties for $\mathbf{p}$. We require the relation R to be *sound* with respect to the device model.

Definition II.1 (Sound Abstraction)**.** A relational model of a device $R(V_{GS}, V_{DS}, \mathbf{p}, I_{DS})$ is sound with respect to a functional model $I_{DS} = F(V_{GS}, V_{DS}, \mathbf{p})$ if for all $(V_{GS}, V_{DS}, \mathbf{p}) \in \mathbb{D}$,

$$I_{DS} = F(V_{GS}, V_{DS}, \mathbf{p}) \;\Rightarrow\; R(V_{GS}, V_{DS}, \mathbf{p}, I_{DS})\,.$$

In other words, the relation R over-approximates the behavior of the device modeled using the function F.

The purpose of piecewise linear modeling of a device is to find a relation R that is sound with respect to some device model such that R is expressible as a linear arithmetic formula. A standard approach for piecewise linear modeling is to find a piecewise linear approximation $\tilde{F}(V_{GS}, V_{DS}, \mathbf{p})$ that minimizes some penalty function

$$\epsilon = \max_{(V_{GS}, V_{DS}, \mathbf{p}) \in \mathbb{D}} |F(V_{GS}, V_{DS}, \mathbf{p}) - \tilde{F}(V_{GS}, V_{DS}, \mathbf{p})|\,.$$

Given such an $\tilde{F}$, we can obtain a relation R by "bloating" $\tilde{F}$ using the error ϵ:

$$R(V_{GS}, V_{DS}, I_{DS}, \mathbf{p}) :\; I_{DS} \in \tilde{F}(V_{DS}, V_{GS}, \mathbf{p}) + [-\epsilon, \epsilon]\,.$$

[1] Cf. http://www-device.eecs.berkeley.edu/bsim/.

[2] Cf. http://ptm.asu.edu/.

If $\tilde{F}$ can be expressed in linear arithmetic, then R itself can be expressed in linear arithmetic. In practice, however, the device model F is often not available in a simple closed form, which makes the computation of ϵ difficult. Instead, we use a large number of samples

$$(V_{GS}^{(0)}, V_{DS}^{(0)}, \mathbf{p}^{(0)}, I_{DS}^{(0)}),\ \ldots,\ (V_{GS}^{(N)}, V_{DS}^{(N)}, \mathbf{p}^{(N)}, I_{DS}^{(N)})\,,$$

each consisting of observed voltages, currents and parameter values, to compute the $\tilde{F}$ that minimizes the sample error. We then compute the relation R by using the interval defined by the sample error. The resulting relation R is *sound with respect to samples*, but not necessarily with respect to F. Often, the samples are divided into a smaller *training set* that is used to find $\tilde{F}$, and a large *evaluation set* that is used to compute the error estimate ϵ. If the number of sample points is large and the sampling is uniform, then soundness with respect to samples can be used as a basis for constructing formal models.

Through the rest of the paper, whenever we claim "soundness" of a device model, it refers to soundness with respect to some pre-specified, sufficiently large number of data points.

B. Piecewise Linear Functions

We now discuss various forms of piecewise linear functions and the approximation of nonlinear devices using them.

Consider a domain $\mathbb{D} \subseteq \mathbb{R}^n$. A function $\mathbf{f}(\mathbf{x}) : \mathbb{D} \to \mathbb{R}^m$ is piecewise linear (PWL) if there exists a K-piece partition $S_1, \ldots, S_K$ of $\mathbb{D}$ such that $\mathbf{f}(\mathbf{x})$ can be written as

$$\mathbf{f}(\mathbf{x}) = \begin{cases} \mathbf{a}_1 + \mathbf{B}_1\mathbf{x} & \mathbf{x} \in S_1 \\ \cdots & \cdots \\ \mathbf{a}_K + \mathbf{B}_K\mathbf{x} & \mathbf{x} \in S_K \end{cases} \tag{1}$$

where $\mathbf{a}_i$ are $m \times 1$ vectors, $\mathbf{x}$ is an $n \times 1$ vector, and $\mathbf{B}_i$ are $m \times n$ matrices. We call Equation (1) the conventional form of PWL functions.

Any continuous function $g(\mathbf{x})$ over a bounded domain $\mathbb{D}$ can be approximated to arbitrary accuracy by a PWL function $\tilde{g}(\mathbf{x})$ with a large enough K. However, as K grows, PWL functions become unwieldy.

Simplicial Form. One way to construct a PWL function in conventional form is based on *simplicial decomposition* [39], which subdivides a domain into many simplices $S_1, \ldots, S_K$. Recall that an n dimensional simplex is a polyhedron with $n+1$ vertices. For instance, a 2-simplex is a triangle and a 3-simplex is a tetrahedron. Algorithms for simplicial decomposition of a domain are well-known. If the domain $\mathbb{D}$ is a box, a simplicial decomposition can be constructed in two steps: (1) partition $\mathbb{D}$ into smaller boxes $\mathbb{D}_1, \ldots, \mathbb{D}_k$ by choosing cutpoints along each dimension; (2) subdivide $\mathbb{D}_j$ into simplices. For example, a rectangle yields 2 simplices, and a cuboid yields 5 simplices. Known simplicial decomposition schemes yield $O(n!)$ simplices of n dimensions.

Once we decompose $\mathbb{D}$ into K simplices $S_1, \ldots, S_K$, we assume that for S_i, the linearization is an unknown function $f_i = a_i + \langle \mathbf{b}_i, \mathbf{x} \rangle$, where $\langle\rangle$ denotes inner product. Since S_i has $n+1$ vertices, we evaluate the function $f(\mathbf{x})$ at each vertex and

TABLE I
A CROSS SECTION OF VERIFICATION APPROACHES FOR ANALOG CIRCUITS.

References	Description
Althoff et al. [7]	Verification of Phase Locked Loop.
Frehse et al. [18]	Using Phaver [19] to verify oscillators.
Dang et al. [20]	Verification of $\Delta - \Sigma$ modulator.
Gupta et al. [21]	Using checkmate to verify $\Delta - \Sigma$ modulator [22], [23].
Tiwary et al. [2]	SAT-based D.C analysis, piecewise interval device modeling.
Zaki et al. [6]	Taylor Models Interval Arithmetic [24], [25].
Zaki et al. [3]	DC operating point analysis using nonlinear solvers [10].
Steinhorst et al. [26]	Specification language and model checking by guaranteed integration.
Hartong et al. [4], [27]	Discretization of dynamics and Model checking.
Hartong et al. [28]	Equivalence checking by sampling vector fields at finitely many points.
Little et al. [29]–[31]	Translation to Hybrid Petri Nets and model-checking.
Clarke et al. [32], [33]	Stat. model checking [34] of $\Delta - \Sigma$ modulators.
Singhee et al. [35], [36]	Monte Carlo methods, rare event simulations [37], [38].
Denman et al. [9]	Deriving Laplace transfer functions and verifying using theorem proving.

set up $n+1$ linear equations in terms of a_i and $\mathbf{b}_i$, which yields a unique solution. The resulting f is continuous since the values of f_i and f_j agree at the common vertices of S_i and S_j. A PWL function constructed using simplicial decomposition is said to be a simplicial PWL (SPWL) function.

SPWL functions are practical when the number of inputs is relatively small. For instance, if we assume that the parameters of an NMOS device are fixed, SPWL decomposes the input space in terms of V_{GS} and V_{DS} into triangles. However, if there are many inputs (e.g., V_{GS}, V_{DS} and a number of uncertain transistor model parameters), models based on SPWL can be quite expensive due to the exponential blowup in the number of simplices.

Canonical Form. A continuous PWL function can also be written as:

$$\mathbf{f}(\mathbf{x}) = \mathbf{a} + \mathbf{B}\mathbf{x} + \sum_{i=1}^{\sigma} \mathbf{c}_i \left|\langle \alpha_i, \mathbf{x}\rangle + \beta_i\right| \tag{2}$$

where $\mathbf{a}$ and $\mathbf{c}_i$ are $m \times 1$ vectors, $\mathbf{x}$ and α_i are $n \times 1$ vectors, B is an $m \times n$ matrix, and β_i is a scalar. Equation (2) is known as the *canonical* form [15], which is more succinct than the conventional form.

We construct a PWL function in canonical form (CPWL) as follows. First, we sample over $\mathbb{D}$ to obtain N samples $\mathbf{x}_i, y_i$, where $1 \leq i \leq N$. Next, we use a gradient descent method that minimizes the error between the output values and the sample points. The gradient descent method is detailed in [15]. Here we present a brief description.

Consider a real-valued function $f(\mathbf{x})$ and a CPWL function

$$\hat{f}(\mathbf{x}) = a + \mathbf{b}\mathbf{x} + \sum_{j=1}^{\sigma} c_j \left|\langle \alpha_j, \mathbf{x}\rangle + \beta_j\right| ,$$

where a, $\mathbf{b}$, c_j, α_j and β_j are unknown coefficients. Given a set of N samples $\{(\mathbf{x}_i, y_i) \mid y_i = f(\mathbf{x}_i)\}$, let

$$z_1 \equiv [a\ b_1 \cdots b_n\ c_1 \cdots c_\sigma]^T ,$$
$$z_2 \equiv [\alpha_{1,1} \cdots \alpha_{k,n}\ \beta_1 \cdots \beta_k]^T ,$$

and define the L_2-norm error as

$$E(z_1, z_2) \equiv \sum_{i=1}^{N} \left[w^{(i)} \left(\hat{f}(\mathbf{x}_i) - f(\mathbf{x}_i) \right)^2 \right] ,$$

where $w^{(i)}$ is the weight of the i-th sample. The L_2-norm error $E(z_1, z_2)$ is minimized by iteratively moving z_2 along the steepest descent direction and computing the local minimum with respect to z_1. When the error reaches a minimum, or is below some threshold, we find an approximation $\hat{f}$.

We simplify the above algorithm as follows. We fix z_2 such that it subdivides the domain $\mathbb{D}$ into hyper-rectangles. Then we compute the local minimum of E, where we get a set of values for z_1. The resulting z_1, along with the pre-selected z_2, leads to a function that generally does not have the minimal error. As shown later, we will "bloat" this function into a sound abstraction. Therefore, instead of getting the CPWL function with minimal error, our concern is more on obtaining a reasonable approximation with low computational effort.

Piecewise Constant Functions. When $\mathbf{B}_1, \ldots, \mathbf{B}_K$ in Equation (1) are set to zero, a useful sub-class of functions is obtained: piecewise constant functions (PWC). They trade off accuracy for computational efficiency.

C. PWL Device Modeling

A PWL device model of a set of samples $\{\mathbf{x}_i, F(\mathbf{x}_i)\}$ is a pair of PWL functions $\tilde{F}_l$ and $\tilde{F}_u$ such that for all i, $\tilde{F}_l(\mathbf{x}_i) \leq F(\mathbf{x}_i) \leq \tilde{F}_u(\mathbf{x}_i)$. Hence, a PWL device model is a relational model that is sound with respect to the samples. We assume that for $\mathbf{x} \in \mathbb{D}$, $F(\mathbf{x})$ can be evaluated. Without loss of generality, we also assume that $\mathbb{D}$ is a box obtained as the Cartesian product of intervals $I_1, \ldots, I_n$ for each x_i of $\mathbf{x}$.

Model Generation. We generate a PWL model for $F(\mathbf{x})$ as follows. First, we construct a PWL function $\tilde{F}(\mathbf{x})$ using the procedures described in the previous section. Then for a set of N samples $\{\mathbf{x}_i, F(\mathbf{x}_i)\}$, which we call the evaluation set, we compute an empirical error estimate

$$\hat{\epsilon} = \max_{1 \leq i \leq N} \left| F(\mathbf{x}_i) - \tilde{F}(\mathbf{x}_i) \right| .$$

We add the interval $[-\hat{\epsilon}, \hat{\epsilon}]$ to the function $\tilde{f}$ to obtain a relational model that is sound with respect to the samples.

One refinement of this approach computes $\hat{\epsilon}_l$ and $\hat{\epsilon}_u$ that capture the under-approximation and over-approximation errors respectively. Furthermore, $\hat{\epsilon}_l$ and $\hat{\epsilon}_u$ can be computed for each S_i, which provides a more fine-grained error estimation. For CPWL functions, the piecewise estimation is not immediate since the subdivisions are represented implicitly.

The construction of a PWC model is straightforward. Given a partition $S_1, \ldots, S_K$, we simply compute the minimal and maximal values of $F(\mathbf{x})$ for each S_i. This results in a function interval $[\tilde{F}_l, \tilde{F}_u]$ that contains all the samples.

Model Encoding. Finally, we consider the encoding of the models in linear arithmetic. Figure 1 shows the schema for encoding PWC, SPWL and CPWL models. Let $n = |\mathbf{x}|$ be the number of inputs and assume that each component x_i is subdivided into k parts. We define the size of a formula in terms of n and k. A PWC model considers $K = k^n$ boxes. For each box, the size of the formula is $O(n)$. Hence, the size of the encoding is $O(k^n n)$. For an SPWL model, we have $K = O(k^n n!)$, assuming a fixed simplicial decomposition scheme that divides a cube into $n!$ simplices. The size of a formula for each box is also $O(n)$, resulting in a encoding whose size is $O(k^n n!)$. A CPWL model encodes $K = (k+1)n$ boundaries (the absolute value terms in the canonical representation) rather than boxes. Each boundary equation has a size of $O(n)$, yielding an encoding of size $O(kn^2)$.

Thus, CPWL models have the most economical encoding, while SPWL results are potentially the least efficient. However, note that even if two formulae are of the same size, they are not necessarily equivalent in terms of computational effort.

III. Formal DC Operating Point Analysis

The goal of formal DC operating point analysis is to list all DC operating points of a circuit. The standard approach to this problem consists of two steps: [2], [3] (a) Encode the DC operating point condition as constraints, and (b) subdividing the input and output voltages into many regions, query the solvers to find if an operating point can exist inside a given input/output region pair. The nonlinear devices are modeled as described in Section II.

We note that DC operating regions, especially metastable regions are relatively hard to identify using simulation tools like SPICE. A common approach is to perform simulations with the circuit initialized near a potential DC operating point and check if the circuit settles to a nearby DC operating point (see, for example [40]).

Circuit Encoding. The circuit encoding consists of the Kirchhoff's current law (KCL) and the device models. The KCL asserts that the current flowing into a node is equal to the current flowing out. If a node connects to a voltage source or ground, its encoding is unnecessary since the current of a voltage source or ground is unconstrained. PWL device models are generated and encoded in linear arithmetic as discussed in Section II.

Abstraction Refinement. The DC analysis can be performed "monolithically" by a single fine-grained encoding, followed by numerous queries over regions that can "pinpoint" a DC operating point to the required degree of precision. A more efficient top-down approach is suggested by Tiwary et al. [2] wherein the DC operating points are discovered by repeated subdivision much like a branch-and-bound scheme. Initially, large regions are queried for the presence of a DC operating point using a coarse PWL model. If the solver returns a satisfiable answer, then the regions are subdivided and refined PWL models are fitted to these regions.

Spurious Region. We call a region *spurious* if it is reported by the analysis, but does not actually contain operating points. Spurious regions are produced by the sound abstraction of device models which over-approximates the behavior of devices. Consider the inverter in Figure 2 with its input fixed to $0.5V$. The output can vary between $0.4V$ and $0.6V$ due to the abstraction (in contrast to $0.5V$ in reality). Suppose the transistors are linearized on the regions $0.35 \leq V_{out} \leq 0.45$, $0.45 \leq V_{out} \leq 0.55$ and $0.55 \leq V_{out} \leq 0.65$. Then the regions $[0.35, 0.45]$ and $[0.55, 0.65]$ become spurious. A finer abstraction may lead to fewer spurious regions. But it also results in a more complicated model.

IV. Formal Transient Analysis

The abstraction of nonlinear devices also enables formal transient analysis. Formal transient analysis deals with reachability problems, i.e, given an initial condition, whether the circuit output can reach values in some range. A simple approach proposed by Tiwary et al. [2], is to generate an approximate transition relation by encoding the change in voltages and currents across capacitors and inductors in the circuit. The resulting change is approximated by an Euler step. While such a transient analysis is a poor alternative to the more sophisticated approach adopted by linear hybrid systems based approaches [7], [41], it allows us to encode the approximate reachability by means of a BMC formula. This can be a potentially faster approach to exploring all possible behaviors for a bounded time interval.

We employ the transient analysis scheme as yet another evaluation method for comparing the various PWL models considered in Section II. However, we note that the Euler step can be a large over-approximation unless the time step is small. However, a small time step also means that the depth of the BMC encoding needs to be larger to perform time bounded reachability up to the same time interval.

V. Experimental Evaluation

In this section, we compare the various device modeling approaches, PWC, SPWL and CPWL. We apply them to formal DC operating point and transient analyses, as described in Section III and Section IV. We implement (using the Python programming language) the various modeling approaches, DC and transient analyses. Our program processes the input circuit as a net list and builds a linear arithmetic formula. We use the Z3 SMT solver to check the satisfiability of these formulae

$$\begin{array}{l} S_1(\mathbf{x}) \;\Rightarrow\; y \in f_1(\mathbf{x}) + [-\epsilon_1, \epsilon_1] \\ \cdots \\ S_K(\mathbf{x}) \;\Rightarrow\; y \in f_K(\mathbf{x}) + [-\epsilon_K, \epsilon_K] \end{array}$$

$$\begin{array}{c} y \in \mathbf{a} + B\mathbf{x} + \sum_{i=1}^{K} c_i r_i + [-\epsilon, \epsilon] \\ \langle \alpha_1, \mathbf{x}_1 \rangle + \beta_1 \geq 0 \Rightarrow r_1 = \langle \alpha_1, \mathbf{x}_1 \rangle + \beta_1 \\ \langle \alpha_1, \mathbf{x}_1 \rangle + \beta_1 < 0 \Rightarrow r_1 = -(\langle \alpha_1, \mathbf{x}_1 \rangle + \beta_1) \\ \cdots \\ \langle \alpha_K, \mathbf{x}_K \rangle + \beta_K \geq 0 \Rightarrow r_K = \langle \alpha_K, \mathbf{x}_K \rangle + \beta_K \\ \langle \alpha_K, \mathbf{x}_K \rangle + \beta_K < 0 \Rightarrow r_K = -(\langle \alpha_K, \mathbf{x}_K \rangle + \beta_K) \end{array}$$

Fig. 1. **(Left)** Schema for encoding SPWL and PWC models; and **(Right)** CPWL encoding.

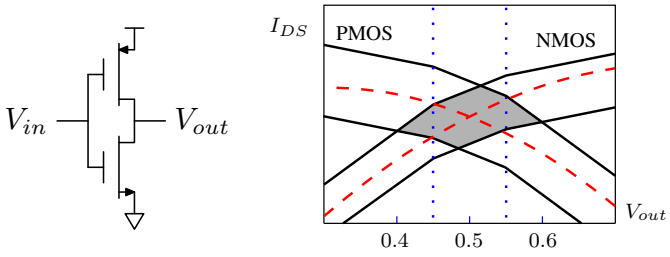


Fig. 2. An inverter and its I/O characteristics. Straight lines show the PWL models of the two devices. Dashed lines are the SPICE models. Shaded region illustrates the approximation due to abstraction. Note that there is only one operating point in the region.

under different input/output intervals. All experiments were run using a Ubuntu 11.10 Desktop on a Quad-core 2.8 GHz machine with 9 GB memory.

We list the benchmarks with brief descriptions in Table II. The letters in the second column refer to the types of devices. M stands for MOSFET transistors, R for linear resistors, C for linear capacitors and L for linear inductors. The numbers count how many devices there are in each type.

The first four benchmarks are ring oscillators with different numbers of stages. The benchmarks starting with "evenosc" are even-stage oscillators from [1]. Their schematic is shown in Figure 3(a). The suffixes "lcbr" and "scbr" denote oscillator benchmarks with known bugs: the oscillators fail due to incorrect transistor sizing [1]. The benchmark "sqwavegen" is a square-wave generator based on a CMOS Schmitt trigger. The "lctankvco" is a voltage-controlled oscillator that uses the inductors and capacitors as the source of oscillation and the cross-coupled pair of transistors as negative resistors to compensate the energy dissipation in the inductor resistance. The schematics of "sqwavegen" and "lctankvco" are shown in Figure 3(b) and 3(c), respectively.

TABLE II
BENCHMARKS FOR DC AND TRANSIENT EXPERIMENTS.

Name	Size	Description
ringosc3s	$6M$	Ring oscillators
ringosc5s	$10M$	
ringosc7s	$14M$	
ringosc9s	$18M$	
evenosclcbr	$16M$	Even-stage oscillators
evenoscscbr	$16M$	
evenoscncbr	$16M$	
sqwavegen	$6M, 1R, 1C$	A square-wave generator
lctankvco	$4M, 2R, 2C, 2L$	An LC-tank VCO

A. Formal DC Operating Point Analysis

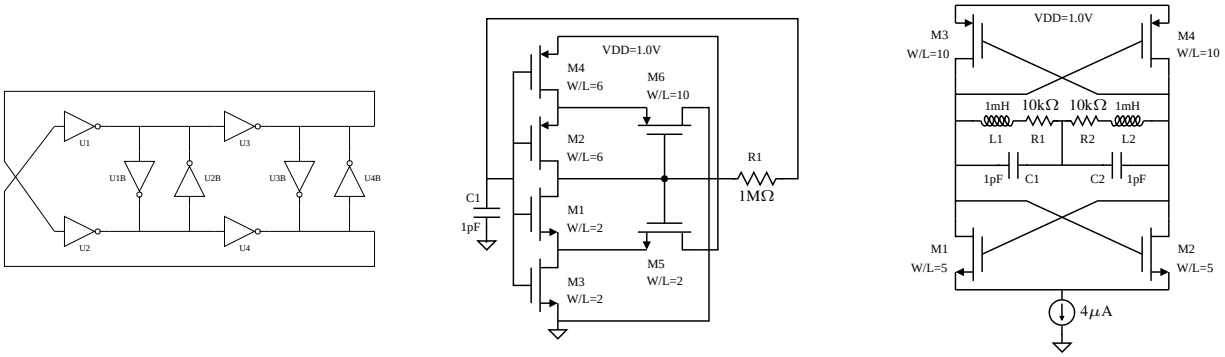
In this part, our goal is to compare the performance of different device modeling approaches in terms of accuracy with respect to SPICE simulation, the number of SMT queries and the running time. The setup is as follows: we fix the device parameters and apply the abstraction refinement described in Section III. The initial number of subdivisions is set to 2 along each dimension. Each refinement step further subdivides each region by splitting each axis into 2 pieces. The refinement process is applied recursively to further subdivide regions that are deemed to contain potential operating points. This process stops after a depth-cutoff that is set to 3 for our experiments.

We carry out our experiments using PWL models with k subdivisions along each dimension, where $k = 1, 2, 4, 6, 8, 16$. We prefix k to denote the specific approach. For instance, 4-PWC stands for PWC models with $k = 4$. We omit 1-CPWL because it does not fit into the algorithm for generating PWL functions in canonical form [15].

We first report the number of regions that may contain operating points by each approach in Table III. The number of operating regions confirmed by SPICE is shown in the last column. The regions found by various PWL models that are not confirmed by SPICE are spurious. In Table III, the number of spurious regions is simply the total number of reported regions minus the number of real operating points. The SPICE-based DC operating point discovery is a trial-and-error process, since the operating points may be metastable.

Accuracy. We compare accuracy in terms of the number of spurious regions in Table III. Observe that SPWL and CPWL are only marginally better than PWC. For the ring oscillator examples, none of the methods reports spurious regions. For the rest of the examples, the number of regions is generally more than twice larger than the number SPICE confirmed. Not surprisingly, we observe that the spurious regions are neighbors to the confirmed regions. We also observe that with more subdivisions, the three approaches tend to get similar results. This shows that the error in the approaches are negligible making their predictions very similar.

SMT Queries. Next, we compare the number of SMT queries in Table IV. The number of SMT queries is a proxy for the number of regions where DC queries occur for a potential operating point. Here, we see that PWC models consistently require more queries than SPWL and CPWL. That is because PWC models over-approximate the underlying device behavior the most, and therefore produce a lot of false positives that are subsequently pruned by refinement. Also, we see that SPWL



(a) **(b)** **(c)**

Fig. 3. Circuit diagrams for **(a)** an even-stage oscillator [1], **(b)** a square-wave generator, and **(c)** a voltage controlled oscillator

TABLE III

THE NUMBER OF SPURIOUS REGIONS, WHICH MAY NOT CONTAIN OPERATING POINTS, REPORTED BY EACH APPROACH. THE LAST COLUMN SHOWS THE NUMBER OF REAL OPERATING POINTS OBTAINED FROM SPICE SIMULATION. THE NUMBERS IN THE SECOND ROW REFER TO THE SUBDIVISION OF THE CORRESPONDING APPROACH ALONG EACH DIMENSION. "TO" MEANS MORE THAN 500 SECONDS.

	PWC						SPWL						CPWL					SPICE
	1	2	4	6	8	16	1	2	4	6	8	16	2	4	6	8	16	
ringosc3s	– 0 –																	1
ringosc5s	– 0 –																	1
ringosc7s	– 0 –																	1
ringosc9s	TO	– 0 –										TO	– 0 –					1
evenosclcbr	14	4	4	4	4	0	4	4	0	0	TO	TO	4	4	4	4	TO	3
evenoscscbr	10	10	6	6	6	6	6	6	6	6	TO	TO	6	6	2	2	TO	3
evenoscncbr	22	12	12	12	12	12	12	12	12	TO	TO	TO	2	TO	2	2	TO	1
sqwavegen	7	5	4	3	3	1	2	1	3	3	3	1	3	2	2	1	1	1
lctankvco	9	9	5	5	3	3	1	3	2	2	2	2	5	5	4	3	3	1

TABLE IV

THE NUMBER OF SMT QUERIES FOR EACH APPROACH. ">" MEANS TIME-OUT WITH A 500 SECONDS LIMIT, AND THE FOLLOWING VALUE IS THE NUMBER OF QUERIES AT TIME-OUT.

	PWC						SPWL						CPWL				
	1	2	4	6	8	16	1	2	4	6	8	16	2	4	6	8	16
ringosc3s	97	13	7	7	7	7	33	13	7	7	7	7	13	7	7	7	7
ringosc5s	833	27	7	7	7	7	149	27	7	7	7	7	27	7	7	7	7
ringosc7s	7937	63	7	7	7	7	653	63	7	7	7	7	63	7	7	7	7
ringosc9s	>43291	157	7	7	7	7	2841	157	7	7	7	>5	157	7	7	7	7
evenosclcbr	929	91	35	27	27	27	213	71	35	27	>26	>3	63	35	27	27	>25
evenoscscbr	801	75	63	55	47	47	221	67	63	55	>34	>3	59	55	39	31	>17
evenoscncbr	1185	119	75	63	55	55	177	79	63	>33	>14	>3	83	>67	63	55	>10
sqwavegen	161	55	25	27	19	15	57	35	25	23	17	15	51	25	27	19	13
lctankvco	149	71	29	27	17	13	59	53	15	19	17	13	71	21	19	17	13

is a slightly better than CPWL. Again, the approaches become quite similar as the number of subdivisions increases.

Running Time. Finally, we consider the running time of each approach with different subdivisions in Table V. It is obvious that PWC models are superior to the remaining models, even though they require more queries to the SMT solver. A likely explanation is that each SMT query from the PWC approach is simpler and far easier to solve than the corresponding queries from the SPWL and CPWL approaches. The results for CPWL models are interesting. First, the running times do not necessarily grow with decreased granularity. Even for a single SMT query, the average solving time does not increase as fast as the other two approaches. A possible explanation is that the complexity of CPWL models grows linearly with the number of subdivisions, unlike the other two approaches, whose models grow exponentially. On the other hand, even the simplest CPWL models take considerably more time than PWC and SPWL counterparts. It suggests that the CPWL encoding is difficult for SMT solvers.

In summary, as the number of subdivisions in the models increases, PWC outperforms SPWL and CPWL in the given set of benchmarks. The running times of PWC and SPWL grow as the granularity decreases. On the other hand, the running times of CPWL are less predictable.

TABLE V
RUNNING TIME FOR EACH APPROACH WITH A 500 SECONDS TIME-OUT.

	PWC						SPWL						CPWL				
	1	2	4	6	8	16	1	2	4	6	8	16	2	4	6	8	16
ringosc3s	<0.1	0.1	0.1	0.4	0.9	11	0.4	0.4	1.6	15	34	339	0.5	1.5	8.6	6.7	33
ringosc5s	2.8	0.3	0.2	1.5	1.5	18	2.8	1.4	3.3	17	42	365	15	4.4	8.3	13	54
ringosc7s	75	0.6	0.2	0.8	2	26	17	4.2	4.3	15	30	376	17	5.1	19	16	104
ringosc9s	>500	2.1	0.3	1	2.6	33	95	16	6.8	18	44	>500	61	17	36	82	247
evenosclcbr	7.4	1.1	1.3	3.2	9.2	146	17	29	152	368	>500	>500	59	158	111	171	>500
evenoscscbr	6.5	1	2.2	7	16	277	17	29	169	406	>500	>500	85	300	371	319	>500
evenoscncbr	9.7	1.7	3.5	11	29	444	22	73	246	>500	>500	>500	199	>500	343	438	>500
sqwavegen	0.2	0.5	0.5	1.6	2.7	27	0.9	1.1	3.5	9.2	16	120	0.8	1.1	2.6	4	19
lctankvco	0.3	0.7	0.6	2	2.7	35	0.6	1.4	2	8.4	15	125	0.9	0.8	1.8	2.7	22

B. Formal Transient Analysis

We use the ring oscillator benchmarks to compare the performance of the three approaches, PWC, SPWL and CPWL, on formal transient analysis. The setup is as follows: for each method, we perform time-bounded reachability queries for different time frames, and compare the accuracy of the results relative to SPICE simulations, which report a single concrete voltage value at each time interval. A reachability query checks whether the output can reach a certain interval in a specified time frame. We set the initial output voltage to $1.0V$ and subdivide the range of the output voltages into ten intervals, each of which is queried individually. We use backward Euler integration to solve the transient behavior of active devices. The time step is fixed to a value that is small enough to obtain accurate integration results.

TABLE VI
REACHABLE INTERVALS (OVER-APPROXIMATIONS) FOUND BY VARIOUS APPROACHES FOR DEPTHS 1, 5 AND 10 STEPS OF TRANSIENT ANALYSIS FOR THE THREE-STAGE RING OSCILLATOR. THE REACHABLE SETS ARE INDEPENDENT OF TRANSISTOR SIZING. RUNNING TIMES ARE LISTED IN TABLE VII.

	Reachable Interval	Approach
1-step	[0.9,1.0]	all approaches
5-step	[0.7,1.0]	16-PWC
	[0.6,1.0]	remaining approaches
10-step	[0.4,1.0]	4-PWC, 6-PWC and 8-PWC
	[0.3,1.0]	remaining approaches

We simulate for one time frame, five time frames and ten time frames respectively. The results are shown in Table VI and Table VII. Barring timeouts, the three approaches produce almost identical reachability results.

First, let us observe that a better model is helpful in getting a more accurate reachable interval. For instance, 4-PWC reports a reachable interval of $[0.4, 1.0]$ at the tenth time frame, while 2-PWC concludes a larger reachable interval: $[0.3, 1.0]$. However, notice that the reachable intervals are generally too over-approximate compared to the SPICE simulations which report a single concrete value at each time step. Therefore, the approximations seem to be too coarse to provide useful reachability information. On the other hand, increasing the number of subdivisions makes the computation intractable (Table VII).

In terms of running time, there are many time-outs for each approach. This may seem surprising since the benchmarks are small and the number of time frames is not large. We suspect that the PWL encoding forces the SMT solver to explore a large set of transistor mode combinations, wherein each subdivision in the PWL represents a transistor mode. The number of such mode combinations increases exponentially as the unrolling depth is increased. The observations suggest that a simple BMC-style encoding of transient analysis may be suboptimal in terms of performance and accuracy.

VI. CONCLUSION

To summarize this paper, we compare the applicability of three device modeling approaches, PWC, SPWL and CPWL, to the formal DC operating point and formal transient analysis. We find that PWC is the most suitable approach for operating point analysis. Both SPWL and CPWL generate more complicated models. The benefits from those models, for instance, fewer spurious regions and fewer SMT queries, do not compensate the extra cost in terms of solving time.

On the other hand, none of the approaches performs well for transient analysis with the described simulation scheme in the selected benchmark set. The results suggest that with a sound abstraction of device models, the simple BMC-style unrolling does not work well.

In the future, it is interesting to identify whether a region contains a stable or metastable operating point. Also, we can utilize the unsatisfiable core of SMT queries, which can potentially rule out more than one region each time. Techniques from unsatisfiablity solvers, such as iSAT [10], can also be applied to the DC operating points analysis.

REFERENCES

[1] K. D. Jones, J. Kim, and V. Konrad, "Some "real world" problems in the analog and mixed signal domains," in *Proceedings of Designing Correct Circuits*, 2008.

[2] S. K. Tiwary, A. Gupta, J. R. Phillips, C. Pinello, and R. Zlatanovici, "First steps towards SAT-based formal analog verification," in *ICCAD*, 2009, pp. 1–8.

[3] M. H. Zaki, I. M. Mitchell, and M. R. Greenstreet, "DC operating point analysis - a formal approach," in *Proceedings of Formal Verification of Analog Circuits (FAC)*, 2009.

[4] L. Hedrich and E. Barke, "A formal approach to nonlinear analog circuit verification," in *ICCAD*, 1995, pp. 123–127.

[5] M. Freibothe, J. Döge, T. Coym, S. Ludwig, B. Straube, and E. Kock, "Verification-oriented behavioral modeling of nonlinear analog parts of mixed-signal circuits," in *Advances in Design and Specification Languages for Embedded Systems*, 2007, pp. 37–51.

TABLE VII
RUNNING TIME OF 1-STEP, 5-STEP AND 10-STEP TRANSIENT ANALYSIS.

		PWC					SPWL					CPWL				
		2	4	6	8	16	2	4	6	8	16	2	4	6	8	16
1-step	ringosc3s	0.1	0.1	0.2	0.3	2.4	0.1	0.3	0.5	1.1	5.6	0.1	0.2	0.3	0.5	2.8
	ringosc5s	0.1	0.2	0.3	0.8	7.1	0.2	0.6	1.4	3.1	18	0.1	0.4	1	1.3	7.8
	ringosc7s	0.1	0.2	0.6	1.2	11	0.3	0.8	2	3.8	27	0.2	0.5	0.8	1.8	13
	ringosc9s	0.2	0.4	0.8	1.8	17	0.5	1.4	3.2	6.4	47	0.3	0.6	1.3	2.1	17
5-step	ringosc3s	0.9	4.7	9	13	80	10	53	191	372	>500	41	44	92	106	430
	ringosc5s	1.9	16	40	59	430	18	221	>500	>500	>500	>500	95	258	317	>500
	ringosc7s	4.3	34	94	162	>500	31	306	>500	>500	>500	>500	161	384	>500	>500
	ringosc9s	3.9	57	146	200	>500	32	>500	>500	>500	>500	>500	478	>500	>500	>500
10-step	ringosc3s	6.7	50	88	185	>500	200	>500				>500				
	ringosc5s	44	>500				372	>500				>500				
	ringosc7s	23	>500				>500	>500				>500				
	ringosc9s	45	>500				>500	>500				>500				

[6] M. H. Zaki, G. Al-Sammane, S. Tahar, and G. Bois, "Combining symbolic simulation and interval arithmetic for verification of AMS designs," in *FMCAD*, 2007, pp. 207–215.

[7] M. Althoff, A. Rajhans, B. Krogh, S. Yaldiz, X. Li, and L. Pileggi, "Formal verification of phase-locked loops using reachability analysis and continuation," in *ICCAD*, 2011, pp. 659–666.

[8] S. Little, D. Walter, K. Jones, C. J. Myers, and A. Sen, "Analog/mixed-signal circuit verification using models generated from simulation traces," *International Journal of Foundations of Computer Science*, vol. 21, no. 2, pp. 191–210, 2010.

[9] W. Denman, B. Akbarpour, S. Tahar, M. H. Zaki, and L. C. Paulson, "Formal verification of analog designs using MetiTarski," in *FMCAD*, 2009, pp. 93–100.

[10] M. Fränzle, C. Herde, S. Ratschan, T. Schubert, and T. Teige, "Efficient solving of large non-linear arithmetic constraint systems with complex Boolean structure," *Journal on Satisfiability, Boolean Modeling and Computation, Special Issue on SAT/CP Integration*, pp. 209–236, 2007.

[11] S. Gao, M. K. Ganai, F. Ivancic, A. Gupta, S. Sankaranarayanan, and E. M. Clarke, "Integrating ICP and LRA solvers for deciding nonlinear real arithmetic problems," in *FMCAD*, 2010, pp. 81–89.

[12] M. K. Ganai and F. Ivancic, "Efficient decision procedure for non-linear arithmetic constraints using CORDIC," in *Proceedings of the Formal Methods in Computer Aided Design*, 2009, pp. 61–68.

[13] P. Nuzzo, A. Puggelli, S. A. Seshia, and A. L. Sangiovanni-Vincentelli, "CalCS: SMT solving for non-linear convex constraints," in *FMCAD*, 2010, pp. 71–79.

[14] L. M. de Moura and N. Bjørner, "Z3: An efficient SMT solver," in *(TACAS)*, 2008, pp. 337–340.

[15] L. O. Chua and A.-C. Deng, "Canonical piecewise-linear modeling," *IEEE Transactions on Circuits and Systems*, no. 5, pp. 511–525, 1986.

[16] M.-J. Chien and E. S. Kuh, "Solving nonlinear resistive networks using piecewise-linear analysis and simplicial subdivision," *IEEE Transactions on Circuits and Systems*, vol. 24, no. 6, pp. 305–317, 1977.

[17] V. B. Rao, D. V. Overhauser, T. N. Trick, and I. N. Hajj, *Switch-Level Timing Simulation of MOS VLSI Circuits*. Kluwer Academic Publishers, 1989.

[18] G. Frehse, B. H. Krogh, and R. A. Rutenbar, "Verifying analog oscillator circuits using forward/backward abstraction refinement," in *DATE*, 2006, pp. 257–262.

[19] G. Frehse, "Phaver: Algorithmic verfication of hybrid systems past hytech," in *HSCC*, 2005, pp. 258–273.

[20] T. Dang, A. Donze, and O. Maler, "Verification of analog and mixed-signal circuits using hybrid systems techniques," in *FMCAD*, 2004.

[21] S. Gupta, B. H. Krogh, and R. A. Rutenbar, "Towards formal verification of analog designs," in *ICCAD*, 2004, pp. 210–217.

[22] B. I. Silva, K. Richeson, B. H. Krogh, and A. Chutiman, "Modeling and verification of hybrid dynamical system using checkmate," in *ADPM*, 2000.

[23] A. Chutiman and B. Krogh, "Computing polyhedral approximations to flow pipes for dynamic systems," in *Proceedings of IEEE CDC*, 1998.

[24] R. E. Moore, R. B. Kearfott, and M. J. Cloud, *Introduction to Interval Analysis*. SIAM, 2009.

[25] M. Berz and K. Makino, "Performance of taylor model methods for validated integration of odes," vol. 3732, pp. 69–74, 2005.

[26] S. Steinhorst and L. Hedrich, "Model checking analog systems using an analog specification language," in *DATE*, 2008, pp. 324–329.

[27] W. Hartong, L. Hedrich, and E. Barke, "Model checking algorithms for analog verification," in *DAC*, 2002, pp. 542–547.

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

[29] S. Little, D. Walter, N. Seegmiller, C. Myers, and T. Yoneda, "Verification of analog and mixed-signal circuits using timed hybrid Petri nets," in *Automated Technology for Verification and Analysis*, 2004, pp. 426–440.

[30] S. Little, N. Seegmeller, D. Walter, C. Myers, and T. Yoneda, "Verification of analog/mixed-signal circuits using labeled hybrid Petri nets," in *ICCAD*, 2006, pp. 275–282.

[31] D. Walter, S. Little, C. J. Myers, N. Seegmiller, and T. Yoneda, "Verification of analog/mixed-signal circuits using symbolic methods," *IEEE Transactions on CAD of Integrated Circuits and Systems*, vol. 27, no. 12, pp. 2223–2235, 2008.

[32] E. M. Clarke, A. Donze, and A. Legay, "Statistical model checking of analog mixed-signal circuits with an application to a thrid order δ-σ modulator," in *Proceedings of the 4th International Haifa Verification Conference on Hardware and Software: Verification and Testing*, 2009, pp. 149–163.

[33] E. M. Clarke and P. Zuliani, "Statistical model checking for cyber-physical systems," in *Automated Technology for Verification and Analysis*, 2011, pp. 1–12.

[34] H. L. S. Younes and R. G. Simmons, "Statistical probabilistic model checking with a focus on timed-bounded properties," *Information and Computation*, vol. 204, no. 9, pp. 1368–1409, 2006.

[35] A. Singhee and R. A. Rutenbar, "From finance to flip flops: A study of fast quasi-monte carlo methods from computational finance applied to statistical circuit analysis," in *Proceedings of the 8th International Symposium on Quality Electronic Design*, 2007, pp. 685–692.

[36] A. Singhee, S. Singhal, and R. A. Rutenbar, "Practical, fast Monte Carlo statistical static timing analysis: Why and how," in *ICCAD*, 2008, pp. 190–195.

[37] R. Y. Rubinstein and D. P. Kroese, *Simulation and the Monte Carlo Method*. Wiley Series in Probability and Mathematical Statistics, 2008.

[38] ——, *The Cross-entropy Method: An Unified Approach to Combinatorial Optimization, Monte-Carlo Simlation and Machine Learning*. Springer-Verlag, 2004.

[39] A. Hatcher, *Algebraic Topology*. Cambridge university Press, 2002.

[40] P. Varma, B. S. Panwar, and K. N. Ramganesh, "Cutting metastability using aperture transformation," *IEEE Transactions on Computers*, vol. 53, pp. 1200–1204, 2004.

[41] G. Frehse, C. L. Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, "SpaceEx: Scalable verification of hybrid systems," in *CAV*, ser. Lecture Notes in Computer Science, vol. 6806. Springer, 2011, pp. 379–395.