Formal Verification of Phase-Locked Loops Using Reachability Analysis and Continuization

Matthias Althoff
malthoff@ece.cmu.edu

Akshay Rajhans
arajhans@ece.cmu.edu

Bruce H. Krogh
krogh@ece.cmu.edu

Soner Yaldiz
syaldiz@ece.cmu.edu

Xin Li
xinli@ece.cmu.edu

Larry Pileggi
pileggi@ece.cmu.edu

Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213

ABSTRACT

We present an approach for verifying locking of charge-pump phase-locked loops by performing reachability analysis on a behavioral model of the circuit. Bounded uncertain parameters in the behavioral model make it possible to represent all possible behaviors of more detailed models. The dynamics of the behavioral model is hybrid (i.e., discrete and continuous) due to the switching of charge pumps that drive the analog control circuits. A unique feature of phase-locked loops compared to most other hybrid systems is that they require thousands of switchings in the continuous dynamics to converge sufficiently close to a limit cycle. This makes reachability analysis a challenging task since switches in the dynamics are expensive to compute and result in conservative overapproximations. We solve this problem by overapproximating the effects of the switching conditions with uncertain parameters in linear continuous models, a method we call *continuization*. Using efficient reachability algorithms for discrete-time linear systems, locking is verified over the complete range of possible initial states of a charge-pump PLL designed in 32nm CMOS SOI technology in comparable time required for Monte Carlo simulations of the same behavioral model.

1. INTRODUCTION

While formal verification techniques are used extensively for checking the correctness of digital circuit design, their application to analog and mixed-signal circuits has been limited. Analog and mixed-signal circuit verification suffers from the inherent complexity of continuous- and hybrid-dynamic system verification. Typically, the analysis of analog design correctness is done using transistor-level or behavioral simulations, which provide insight into the circuit behavior for particular choices of parameter values and initial conditions. A thorough analysis over a large operating range can require many simulations. Formal methods aim to verify the correctness of circuit designs over entire ranges of parameter variations and initial conditions without using simulation.

This paper focuses on the formal verification of charge-pump phase-locked loops (PLLs), which are critical components of communication and computing systems [10]. A PLL is a dynamic feedback system that synthesizes a low-noise, high-frequency signal by locking its divided phase and frequency to a low-frequency reference signal. A key performance specification for PLLs is the *lock time*, which is the time it takes to achieve phase and frequency locking after a change in division ratio or after a perturbation to the system. Although linear approximations of PLLs are useful for checking stability [10], lock time analysis requires transient simulations due to the time-varying sampled nature of the system. To reduce the simulation cost, behavioral models are used, particularly for stiff PLLs with high division ratios. There is still a need for formal verification of the PLL lock time since a finite set of simulations cannot guarantee locking for all possible initial conditions and parameter variations.

In this paper we demonstrate that the lock time for a charge-pump PLL can be formally verified in time comparable to the time required for Monte Carlo simulations of the same behavioral model. In contrast to other hybrid systems that have been analyzed in the literature, PLLs require thousands of switchings in the continuous dynamics to converge sufficiently close to a limit cycle. This makes reachability analysis a challenging task since switches in the dynamics are expensive to compute and result in conservative overapproximations. Efficient reachability analysis is achieved through *continuization* of the switching dynamics: a continuous model is derived that replaces the switching conditions with uncertain parameters. Using reachability analysis for linear systems with uncertain parameters, all states reachable by any possible simulation of the original model can be computed using the continuous model. It is also shown that the added uncertainty does not result in a large overapproximation of cycles for which locking of the PLL can be verified. Overapproximation might lead to the conclusion that the PLL does not lock, although in reality it does when overapproximations are not present. If the overapproximations are tight, a negative result still implies the circuit design is not sufficiently robust against parameter uncertainty.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...$10.00.

The following section reviews previous work on formal verification of analog circuits. In Sec. 3 we derive the behavioral model of the PLL as a hybrid automaton and specify the properties to be checked for lock time. Sec. 4 presents the abstraction of the hybrid automaton to a discrete time linear system with uncertain parameters, and Sec. 5 presents the algorithms for verifying the transient and invariant behavior of the system. Computational results are presented in Sec. 6.

2. PREVIOUS WORK

In [28], Zaki et al. survey the literature on the application of formal verification techniques in analog and mixed-signal (AMS) designs. They categorize the verification techniques used in AMS designs into equivalence checking, automated state-space exploration, run-time verification and proof-based methods. We briefly summarize these categories and also cite work that has appeared since the 2008 survey.

Equivalence checking determines the maximum error of the input-output behavior or other distance measures between two system models. Circuits are compared on the same or different levels of abstraction, e.g. SPICE netlists versus analog behavioral models [22, 23].

Run-time verification analyzes signals using monitors synthesized from specifications, which can be applied online or offline [15, 19, 20, 26]. Run-time verification is a very practical approach due to its small computational costs, but cannot guarantee conformance of specifications due to the finite number of tested signal traces.

For the verification of behavioral models, two main approaches exist: state-space exploration and theorem proving. Theorem provers guarantee properties by applying proof rules, which simplify the formula to be checked (typically requiring human intervention) until one obtains atomic statements which are true or false [6, 21].

Since we use a state-space exploration technique in this paper, we will focus the remaining literature survey on this approach. One line of research discretizes the state space of the continuous circuit dynamics to obtain purely discrete systems [11, 13, 16, 17, 25]. This makes it possible to use model checking algorithms for discrete systems [4]. This method suffers from the discrete state explosion problem, limiting its application to systems with few (up to 4) continuous state variables. Another possibility is to directly perform state exploration on the hybrid dynamics, which is also called *reachability analysis*. There is a rich literature on reachability analysis [2], but we will focus on the work that has been applied to analog circuits. In general, the set of all reachable states cannot be represented exactly. Therefore, it is overapproximated using geometrical structures such as polyhedra [5, 9, 12], regions specified by difference-bound matrices (polyhedra with 45° and 90° angles) [18], or boxes computed via Taylor approximations and interval arithmetic [27].

Other techniques that are not considered in [28] are a boolean satisfiability (SAT) based method that directly works with a circuit-level netlist [24], and statistical model checking that, in contrast to pure Monte Carlo simulation, returns probabilities on satisfying temporal properties [3].

None of the methods described above has been used to verify PLL lock time because of the extremely long transient time required for convergence. Our attempts to apply brute force hybrid system reachability using tools such as PHAVer [7] or SpaceEx [8] failed due to the slow convergence close to locking.

3. PROBLEM FORMULATION

A PLL circuit typically consists of the following parts: a reference signal generator (Ref), a voltage-controlled oscillator (VCO), a phase frequency detector (PFD), and charge pumps (CPs). We consider the dual path, type II, third order charge-pump PLL shown in Fig. 1. The reference frequency generator produces a high-quality sinusoidal signal at a fixed low frequency (MHz). The VCO, on the other hand, generates a lower quality, but high-frequency signal (GHz). The purpose of PLLs is to 'lock' the controlled frequency of the VCO so that its output has the same frequency (when divided by N) and phase as the reference signal.

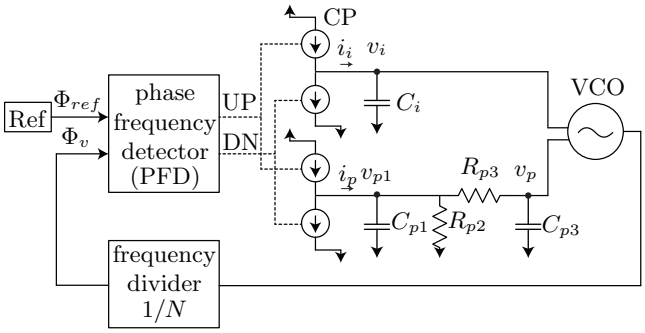


Figure 1: Dual-path charge-pump PLL.

Locking of the PLL is achieved by the PFD comparing the phases of the reference signal and the VCO signal and setting the signals UP = 1 if the Ref signal leads, and DN = 1 if it lags. These signals pump charge into or out of the capacitors, changing voltages v_p and v_i, which serve as proportional and integral (PI) control inputs to the VCO. We do not consider adaptation of PLL parameters such as the frequency divider, resistor, or capacitor values.

As a behavioral model of the charge-pump PLL, we construct a hybrid automaton with linear continuous dynamics with uncertain parameters. Appropriate bounds on the uncertain parameters can be determined by equivalence checking with detailed circuit models [22,23]. These bounds should be chosen to assure that the behavioral model represents all possible behaviors of a detailed circuit model. If the more detailed model is at the transistor level, the approach is also able to catch issues at the transistor level. However, current equivalence checking techniques are typically semi-formal such that a complete enclosure cannot yet be guaranteed.

The continuous state vector in the behavioral model is $x = [v_i\, v_{p1}\, v_p\, \Phi_v\, \Phi_{ref}]^T$ with input vector $u = [i_i\, i_p]^T$ (see Fig. 1). The dynamics are

$$\dot{x} = Ax + Bu + c, \tag{1}$$

with

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{C_{p1}}\left(\frac{1}{R_{p2}} + \frac{1}{R_{p3}}\right) & \frac{1}{C_{p1}R_{p3}} & 0 & 0 \\ 0 & \frac{1}{C_{p3}R_{p3}} & -\frac{1}{C_{p3}R_{p3}} & 0 & 0 \\ \frac{K_i}{N} & 0 & \frac{K_p}{N} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$B = \begin{bmatrix} \frac{1}{C_i} & 0 \\ 0 & \frac{1}{C_{p1}} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad c = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{2\pi}{N} f_0 \\ 2\pi f_{ref} \end{bmatrix},$$

where the resistor and capacitor values can be found in Fig. 1 and the values K_i, K_p, and f_0 determine the frequency of the VCO: $f_{VCO} = \frac{1}{2\pi}(K_i v_i + K_p v_p) + f_0$. Input values u vary depending on the signals leaving the PFD according to

$$u = \begin{cases} [I_i^{\mathrm{UP}}\ I_p^{\mathrm{UP}}]^T, \text{ if UP} = 1, \text{DW} = 0 \\ [I_i^{\mathrm{DW}}\ I_p^{\mathrm{DW}}]^T, \text{ if UP} = 0, \text{DW} = 1 \\ [I_i^{\mathrm{UP}} + I_i^{\mathrm{DW}}\ I_p^{\mathrm{UP}} + I_p^{\mathrm{DW}}]^T, \text{ if UP} = 1, \text{DW} = 1 \\ [0\,0]^T, \text{ if UP} = 0, \text{DW} = 0 \end{cases}$$

The output signals of the PFD are determined by threshold crossings of phase signals. The switching logic is described by the automaton shown in Fig. 2, where the states are labeled as *up_active*, *dw_active*, *both_active*, and *both_off*. Including the continuous dynamics into the automaton for the switching logic results in a hybrid automaton [14].

Starting in *both_off*, the next discrete state of the hybrid automaton is *up_active* if the reference signal leads by first reaching $\Phi_{ref} = 2\pi$, and *dw_active* when $\Phi_v = 2\pi$ is reached first. As shown in Fig. 3, in order to use the same phase crossings for the next cycle, the phase values are reset to $\Phi_{ref} := \Phi_{ref} - 2\pi$, $\Phi_v := \Phi_v - 2\pi$ upon continuing in *up_active* and *dw_active*. Once the lagging signal has a zero-crossing, the discrete state *both_active* is entered which models a time delay t_d for switching off both charge pumps. After the delay, the system is in *both_off* again, which completes one cycle.

Locking is achieved when the phase difference reaches and remains within the *locked condition* given by the interval $[-0.1°, 0.1°]$. There are two specifications that need to be verified:

SPECIFICATION 3.1 (TRANSIENT BEHAVIOR). *Given the PLL starting from any initial state and any valid set of parameters, verify that the locked condition is reached in less than $\overline{k}$ cycles.*

SPECIFICATION 3.2 (INVARIANT BEHAVIOR). *Given a set of states reached from any initial state and any valid set of parameters such that all states are in the locked condition, show that the locked condition is an invariant, that is, show that starting from any state in the given set, the PLL state remains in the locked condition indefinitely.*

Although simulations may show that the PLL remains in the locked condition for many cycles after $\overline{k}$, finite-length simulations can never guarantee the locked condition to hold indefinitely.

4. TIME DISCRETIZATION AND CONTINUIZATION

In this section, we create an abstraction of the PLL behavioral model to be used to perform reachability analysis. We assume a particular initial state is given and focus on the computation of the state over one cycle of the reference signal. Given the hybrid automaton behavioral model of the PLL, we first derive a discrete-time linear model with bounded uncertain parameters based on the phase of

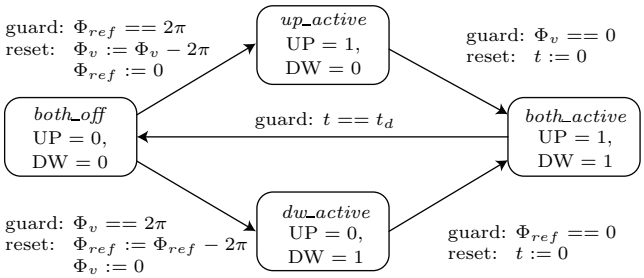


Figure 2: Hybrid automaton.

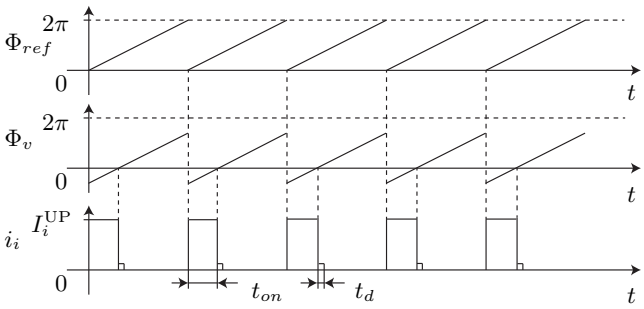


Figure 3: Typical charge pump activity.

the reference signal, assuming the reference signal leads the VCO signal, i.e., for the discrete state sequence *up_active* $\rightarrow$ *both_active* $\rightarrow$ *both_off*. We then generalize to handle uncertain switching times, making it possible to apply continuization to eliminate the need to compute the effects of the PFD switching explicitly. Finally, we incorporate the case when the reference signal lags the VCO signal.

4.1 Time Discretization

The time for a cycle of the reference signal is given by $t_{cycle} = 1/f_{ref}$. Since the continuous dynamics of the PLL is linear, we can take advantage of the superposition principle and obtain the initial state solution and the input solution separately. The initial state solution for one cycle is given by $x^h(t + t_{cycle}) = e^{At_{cycle}} x(t)$. The input solution for constant input u over the time interval $[0, r]$, where r is the time the charge pump is active, can be written using the Taylor series of e^{At} as

$$\begin{aligned} x^p(r) &= \int_0^r e^{A(r-t)}\, dt\, u \\ &= \Big(\sum_{i=0}^{\eta} \frac{1}{(i+1)!} A^i r^{i+1} + \underbrace{\sum_{i=\eta+1}^{\infty} \frac{1}{(i+1)!} A^i r^{i+1}}_{=:E^p(r)} \Big) u. \end{aligned}$$

The remainder $E^p(r)$ can be overapproximated by an interval matrix, i.e., a matrix with lower and upper bounds on each element, given by the following proposition.

PROPOSITION 4.1 (REMAINDER $\mathcal{E}^p(r)$). *The remainder matrix $E^p(r)$ can be overapproximated by an interval matrix with symmetric bounds: $\mathcal{E}^p(r) = [-\tilde{W}(r), \tilde{W}(r)]$, where*

$$\tilde{W}(r) = W(r)\, r, \quad W(r) = e^{|A|r} - \sum_{i=0}^{\eta} \frac{|A|^i r^i}{i!},$$

and $|A|$ is computed elementwise, i.e. $|A|_{ij} = |A_{ij}|$.

PROOF.

$$|E^p(r)| = \Big| \sum_{i=\eta+1}^{\infty} \frac{A^i}{(i+1)!} r^{i+1} \Big| \leq \sum_{i=\eta+1}^{\infty} \frac{|A|^i r^{i+1}}{(i+1)!}$$
$$\leq \Big(\sum_{i=\eta+1}^{\infty} \frac{|A|^i r^i}{i!} \Big) r = \underbrace{\Big(e^{|A|r} - \sum_{i=0}^{\eta} \frac{|A|^i r^i}{i!} \Big)}_{=:W(r)} r.$$

□

We define the interval matrix

$$\Gamma(r) := \sum_{i=0}^{\eta} \frac{1}{(i+1)!} A^i r^{i+1} \oplus \mathcal{E}^p(r), \tag{2}$$

where $\oplus$ denotes the Minkowski sum.[1] Since there is no input for the rest of the cycle, the input solution after one cycle is $x^p(t_{cycle}) \in e^{A(t_{cycle}-r)}\Gamma(r)u$. Let t_{on} denote the time the system is in location *up_active* and recall that t_d is the time it is in *both_active*. Also, let u denote the input in *up_active* and u_d denote the input in *both_active*. Finally, defining $x_k = x(k\, t_{cycle})$, the combination of the initial state solution and all input solutions can be written as

$$x_{k+1} \in \underbrace{e^{At_{cycle}} x_k}_{=:x^h} \oplus \underbrace{\Gamma(t_{cycle})c}_{=:x^p_{const}} \oplus \underbrace{e^{A(t_{cycle}-t_{on})}\Gamma(t_{on})u}_{=:x^p_{up}} \oplus \underbrace{e^{A(t_{cycle}-t_{on}-t_d)}\Gamma(t_d)u_d}_{=:x^p_{both}}. \tag{3}$$

The above formula is a discrete-time overapproximation after one cycle of the continuous-time evolution.

4.2 Overapproximation of Switching Times

The next step in computing the state after one cycle is to determine the switching time t_{on} (while t_d is a given constant). A closed form solution does not exist for t_{on}, which depends on the state of the system. Simulation techniques obtain t_{on} by detecting a zero-crossing which corresponds to crossing the guard condition $\Phi_v == 0$. Here we propose a more efficient method based on overapproximating the interval of possible values for t_{on}. Since t_{on} depends on only $\Phi_v = x_4$, it is sufficient to consider (see (1))

$$\dot{x}_4 = \frac{1}{N}(K_i x_1 + K_p x_3 + 2\pi f_0). \tag{4}$$

We assume user-defined bounds $x_1 \in [\underline{\omega}_1, \overline{\omega}_1]$, $x_3 \in [\underline{\omega}_3, \overline{\omega}_3]$, which are monitored during the verification process. Violation of these bounds would require to restart the verification with larger intervals. Applying interval arithmetic to (4) results in the bound $\dot{x}_4 \in [\underline{v}, \overline{v}]$. We further extract the bound $[\underline{\delta}, \overline{\delta}]$ on x_4 from the reachable set at the beginning of each cycle. We obtain $t_{on} \in [\underline{t}_{on}, \overline{t}_{on}] = \{x_4/\dot{x}_4 | x_4 \in [\underline{\delta}, \overline{\delta}], \dot{x}_4 \in [\underline{v}, \overline{v}]\}$ using the fact that the reference signal is leading ($\underline{\delta}, \overline{\delta} < 0$), resulting in

$$\underline{t}_{on} = |\overline{\delta}|/\overline{v}, \quad \overline{t}_{on} = |\underline{\delta}|/\underline{v}. \tag{5}$$

4.3 Continuization

In this section we apply the concept of continuization to compute the set of reachable states resulting from uncertain switching times. We use Minkowski addition and set-based multiplication $\mathcal{A} \otimes \mathcal{B} = \{ab | a \in \mathcal{A}, b \in \mathcal{B}\}$ as operations on sets. Note that sets can be sets of scalars, vectors, or matrices, and sets may also contain just a single (certain) element. The set multiplication sign is sometimes dropped when the context makes it clear that uncertain matrices are involved. The standard operator precedence rules apply.

To compute the reachable set under uncertain switching times, we modify (3) so as to use the input u for $[0, \underline{t}_{on}]$, and then the uncertain input $[0,1] \otimes u$ for $[\underline{t}_{on}, \overline{t}_{on}]$ when it is not exactly known if the charge pump is on or off. This is illustrated by the gray region in Fig. 4.

Figure 4: Range of times $[\underline{t}_{on}, \overline{t}_{on}]$ for which the charge pump is switched off. The mode *both_active* is not considered in this figure.

The set of possible next states is then given by

$$x_{k+1} \in e^{At_{cycle}} x_k \oplus x^p_{const} \oplus \underbrace{e^{A(t_{cycle}-\underline{t}_{on})}\Gamma(\underline{t}_{on})u}_{\ni x^{p\,**}_{up};\ \text{input solution for } [0,\underline{t}_{on}]} \oplus [0,1] \otimes \underbrace{e^{A(t_{cycle}-\overline{t}_{on})}\Gamma(\overline{t}_{on}-\underline{t}_{on})u}_{\ni x^{p\,*}_{up};\ \text{input solution for } [\underline{t}_{on},\overline{t}_{on}]} \oplus x^p_{both}.$$

The computation of x^p_{both} has to be modified for uncertain t_{on}, too, as shown below. The drawback of this procedure is that the overapproximation grows with the uncertainty of t_{on} since for each state, all possible values of t_{on} have to be assumed, although the switching time is a function of the state. To address this situation, we translate the uncertainty of the switching time to an uncertainty of the state transition matrix, thus considering the dependency of the state for the switching dynamics. Considering the varying signs of the input u makes it possible to accommodate the switching dynamics in a continuous model. In summary, the input solution for $[\underline{t}_{on}, \overline{t}_{on}]$ is given by

$$x^{p\,*}_{up} \in e^{A(t_{cycle}-\overline{t}_{on})} e^{A(\overline{t}_{on}-t_{on})} \Gamma(t_{on}-\underline{t}_{on}) \mathrm{sgn}(x_4)(-u). \tag{6}$$

Next, we present a lemma that addresses the computation of $e^{A(\overline{t}_{on}-t_{on})}\Gamma(t_{on}-\underline{t}_{on})$ in (6) when t_{on} is uncertain. In the lemma, it is assumed that $\underline{t}_{on} = 0$ which is later revoked.

LEMMA 4.1 (INPUT-TO-STATE-MATRIX SET). *The input-to-state-matrix set*

$$\mathcal{G} := \Big\{ e^{A(\overline{t}_{on}-t_{on})}\Gamma(t_{on}) \Big| t_{on} \in [0, \overline{t}_{on}] \Big\},$$

is overapproximated as

$$\mathcal{G} \subseteq \Big\{ t_{on}\mathcal{C}(\overline{t}_{on}) \Big| t_{on} \in [0, \overline{t}_{on}] \Big\}, \tag{7}$$

where

$$\mathcal{C}(\overline{t}_{on}) = \mathcal{Q}(\overline{t}_{on}) \oplus \mathcal{E}^p(\overline{t}_{on}) \otimes \Big(\sum_{i=0}^{3} \frac{A^i[0, (\overline{t}_{on})^i]}{(i+1)!} \oplus \mathcal{E}^p(\overline{t}_{on}) \Big),$$

$$\mathcal{Q}(\overline{t}_{on}) = I \oplus \Big[\frac{1}{2}\overline{t}_{on}, \overline{t}_{on}\Big] A \oplus \Big[\frac{1}{6}(\overline{t}_{on})^2, \frac{1}{2}(\overline{t}_{on})^2\Big] A^2 \oplus \Big[\frac{1}{8}(\overline{t}_{on})^3, \frac{1}{6}(\overline{t}_{on})^3\Big] A^3$$

[1] Minkowski sum of two sets: $A \oplus B = \{a + b | a \in A, b \in B\}$.

based on a cubic Taylor approximation. I is the identity matrix, and $\mathcal{E}^p(\overline{t}_{on}) = \sum_{i=3}^{\infty} \frac{1}{(i+1)!} A^i [0, (\overline{t}_{on})^i]$.

The proof is omitted due to space limitations. In the following theorem, the time t_{on} in (7) is replaced by a function of the state, making it possible to write the dynamics entirely as a function of the state and $\overline{t}_{on}$.

THEOREM 4.1 (BOUNDED INPUT SOLUTION). *The input solution after one cycle for a constant, but uncertain set of inputs $\mathcal{U}$ and uncertain switching times is (for $\underline{t}_{on} = 0$)*

$$x_{up}^{p\,*} \in \Theta(\overline{t}_{on}) \otimes x(t_0), \tag{8}$$

$$\Theta(\overline{t}_{on}) = \left[\mathbf{0} \quad \mathbf{0} \quad \mathbf{0} \quad -\tfrac{1}{[\underline{v},\overline{v}]} e^{A(t_{cycle}-\overline{t}_{on})} \mathcal{C}(\overline{t}_{on}) \mathcal{U}\right],$$

where $\mathbf{0}$ represents zero vectors of proper dimension, $[\underline{v},\overline{v}]$ is an interval, $\mathcal{C}$ is an interval matrix, and $\mathcal{U}$ is an interval vector. Thus, Θ is an interval matrix.

PROOF. Replacing $e^{A(\overline{t}_{on}-t_{on})}\Gamma(t_{on})$ in (6) by (7) yields

$$x_{up}^{p\,*} \in \Big\{ e^{A(t_{cycle}-\overline{t}_{on})} t_{on} \mathcal{C}(\overline{t}_{on}) \mathrm{sgn}(x_4(t_0))(-u) \Big| \; t_{on} \in [0, \overline{t}_{on}],\, u \in \mathcal{U} \Big\}.$$

The switching time can be overapproximated as $t_{on} \in \frac{|x_4|}{[\underline{v},\overline{v}]}$; see (5). After inserting this result into the previous set, one obtains

$$x_{up}^{p\,*} \in -e^{A(t_{cycle}-\overline{t}_{on})} \mathcal{C}(\overline{t}_{on}) \mathcal{U} \frac{x_4(t_0)}{[\underline{v},\overline{v}]}.$$

The multiplication with $x_4(t_0)$ can be rewritten as a multiplication of the state $x(t_0)$ with the interval matrix Θ. □

When both charge pumps are active, the partial solution is given by

$$x_{both}^{p} \in \underbrace{e^{A(t_{cycle}-\overline{t}_{on}-t_d)} e^{A(\overline{t}_{on}-t_{on})}}_{=e^{A(t_{cycle}-t_{on}-t_d)}} \Gamma(t_d) u_d$$
$$\in e^{A(t_{cycle}-\overline{t}_{on}-t_d)} \otimes \{e^{At} | t \in [0, \Delta t_{on}]\} \otimes \Gamma(t_d) u_d,$$

where $\Delta t_{on} = \overline{t}_{on} - \underline{t}_{on}$ since $t_{on} \in [\underline{t}_{on}, \overline{t}_{on}]$ and

$$\tilde{M}(\Delta t_{on}) := \{e^{At} | t \in [0, \Delta t_{on}]\} = \sum_{i=0}^{\eta} \frac{A^i}{i!} [0, \Delta t_{on}] \oplus \mathcal{E}(\Delta t_{on}). \tag{9}$$

Note that it is sufficient to compute $\mathcal{E}(t)$ at the last point in time since $\forall \tau : 0 < \tau < t : \mathcal{E}(t-\tau) \subset \mathcal{E}(t)$.

In Theorem 4.1 it has been assumed that $\underline{t}_{on} = 0$. This is revoked by first computing $\tilde{x}_k := x(t_k + \underline{t}_{on})$. Next, the time is reset to zero (which is possible since the system is time-invariant), and the computation is continued with $\tilde{x}_k$ as the new initial state:

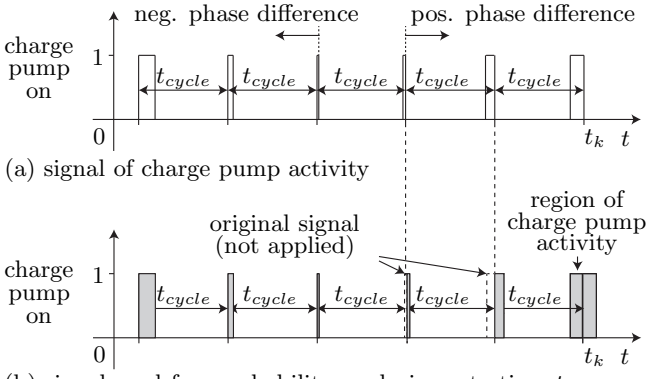
$$\tilde{x}_k \in e^{A\underline{t}_{on}} x_k \oplus \Gamma(\underline{t}_{on}) c \oplus e^{A(t_{cycle}-\underline{t}_{on})} \Gamma(\underline{t}_{on}) u$$
$$x_{k+1} \in (e^{A(t_{cycle}-\underline{t}_{on})} \oplus \Theta(\Delta t_{on})) \tilde{x}_k \oplus \Gamma(t_{cycle} - \underline{t}_{on}) c$$
$$\oplus\, e^{A(t_{cycle}-\overline{t}_{on}-t_d)} \tilde{M}(\Delta t_{on}) \Gamma(t_d) u_d. \tag{10}$$

4.4 Bounding the Solution for Leading and Lagging Reference Signal

Theorem 4.1 shows that positive and negative phase differences can be handled without any distinction of the sign of the phase difference as long as the time interval $[\underline{t}_{on}, \overline{t}_{on}]$ is correctly overapproximated. The time intervals computed in (5) are based on the cycle including *up_active*. If the cycle containing *dw_active* is also considered, the bounds are

$$\underline{t}_{on} = 0, \quad \overline{t}_{on} = \max(|\underline{\delta}|/\overline{v}, |\underline{\delta}|/\underline{v}, |\overline{\delta}|/\underline{v}, |\overline{\delta}|/\overline{v}). \tag{11}$$

When computing the state bounds for a constant cycle time t_{cycle}, the input is applied in the interval $t_k + [0, t_{on}]$ for the cycle containing *up_active* and in the interval $t_k - [0, t_{on}]$ for the cycle containing *dw_active*. As shown in Algorithm 1 below, the different times are taken care of by adding the reachable set due to the input before and after t_k when both cycles are possible. The addition of the input solution for $t_k + [0, t_{on}]$ and $t_k - [0, t_{on}]$ results in an overapproximation since the input solution contains the origin, so that the previous sets are obtained in the set after the addition. Further, it is sufficient to only keep the input applied at $t_k + [0, t_{on}]$ for subsequent computations, which is illustrated in Fig. 5. Thus, the error for adding the input solution for $t_k + [0, t_{on}]$ and $t_k - [0, t_{on}]$ does not accumulate. The procedure of only keeping the input applied at $t_k + [0, t_{on}]$ is realized by the auxiliary reachable set $\tilde{\mathcal{R}}_k$ in Algorithm 1. We skip the proof that this procedure is overapproximative due to space limitations.



(a) signal of charge pump activity

(b) signal used for reachability analysis up to time t_k

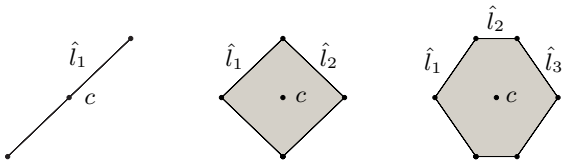
Figure 5: Consideration of inputs when the phase difference changes from negative to positive.

5. REACHABILITY ANALYSIS

We now present how to compute the reachable set for a set of initial states and a sequence of cycles. The reachable sets are represented using zonotopes which have a maximum complexity of $\mathcal{O}(n^3)$ with respect to the system dimension n for the required operations. A zonotope is defined as

$$\mathcal{Z} = \Big\{ x \in \mathbb{R}^n \Big| x = c + \sum_{i=1}^{p} \beta_i g^{(i)}, \quad -1 \le \beta_i \le 1 \Big\},$$

where $c \in \mathbb{R}^n$ is the zonotope center (to which a zonotope is centrally symmetric) and the $g^{(i)} \in \mathbb{R}^n$ are called generators. The order of a zonotope is defined as $o = \frac{p}{n}$. Fig. 6 illustrates a zonotope being constructed step-by-step as the Minkowski sum of a finite set of line segments $\hat{l}_i = [-1, 1]\, g^{(i)}$. Operations on zonotopes and operations between sets of matrices and zonotopes are presented in [1].



(a) $c \oplus \hat{l}_1$ (b) $c \oplus \hat{l}_1 \oplus \hat{l}_2$ (c) $c \oplus \hat{l}_1 \oplus \hat{l}_2 \oplus \hat{l}_3$

Figure 6: Construction of a zonotope by Minkowski addition of line segments.

5.1 Transient Analysis

The algorithm for the reachable set computation when the reference signal is initially leading is presented in Algorithm 1. An interesting property of the PLL is that the number of cycles required for locking is identical when the absolute value of the initial phase difference is equal and the corresponding initial voltages are symmetric with respect to the voltages in the completely locked state. We refer to this property as *symmetric locking time* which makes it sufficient to compute the reachable set only for the case when the reference signal is initially leading. For the symmetric locking, we additionally require that $I_i^{\mathrm{UP}} = -I_i^{\mathrm{DW}}$ and $I_p^{\mathrm{UP}} = -I_p^{\mathrm{DW}}$, which can be relaxed for reachability analysis by choosing the intervals for $I_i^{\mathrm{UP}} + I_i^{\mathrm{DW}}$ and $I_p^{\mathrm{UP}} + I_p^{\mathrm{DW}}$ large enough such that their center is 0. The proof for symmetric locking is omitted due to space limitations.

For simulation purposes, the values of both phases are needed to determine the time for turning the charge pumps on and off. In contrast, the discrete-time model for reachability analysis does not require the exact timing for switching the charge pump values; it is sufficient to keep only the phase difference $x_4 = \Phi_v - \Phi_{ref}$ as a state variable for the reachability computations.

Algorithm 1 Reachable set computation when reference signal is leading at $t = 0$

Input: Initial set $\mathcal{R}_0$, system matrix A, input set $\mathcal{U}$, input set $\tilde{\mathcal{U}}$ for *both_active*
parameters: t_{cycle}, $\Delta\Phi_{lock}$, $\underline{v}$, $\overline{v}$
Output: $\mathcal{R}_{k_{lock}}$
$k = 0$; $P = \begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}$
while $|P\mathcal{R}_k| > \Delta\Phi_{lock}$ **do**
$\underline{t}_{on} = \min(|P\mathcal{R}_{k-1}|)/\overline{v}$, $\overline{t}_{on} = \max(|P\mathcal{R}_{k-1}|)/\underline{v}$
Compute $\Gamma(t)$ for $t \in \{\underline{t}_{on}, t_d, (t_{cycle} - \underline{t}_{on})\}$; see (2)
Compute Θ for $\Delta t_{on} = \overline{t}_{on} - \underline{t}_{on}$; see Theorem 4.1
Compute $\tilde{M}(\Delta t_{on})$; see (9)
$\tilde{\tilde{\mathcal{R}}}_{k+1} = e^{A\underline{t}_{on}}\tilde{\mathcal{R}}_k \oplus \Gamma(\underline{t}_{on})c \oplus e^{A(t_{cycle}-\underline{t}_{on})}\Gamma(\underline{t}_{on})\mathcal{U}$
$\tilde{\mathcal{R}}_{k+1} = (e^{A(t_{cycle}-\underline{t}_{on})} \oplus \Theta(\Delta t_{on}))\tilde{\tilde{\mathcal{R}}}_{k+1}$
$\oplus\Gamma(t_{cycle} - \underline{t}_{on})c \oplus e^{A(t_{cycle}-\overline{t}_{on}-t_d)}\tilde{M}(\Delta t_{on})\Gamma(t_d)\tilde{\mathcal{U}}$
$\mathcal{R}_k = \tilde{\mathcal{R}}_k \oplus \Theta(\Delta t_{on})\tilde{\mathcal{R}}_k$ (consideration of lagging)
$k := k + 1$
end while
$k_{lock} = k - 1$

5.2 Invariant Computation

Once the reachable set fulfills the locking condition $|P\mathcal{R}_k| \leq \Delta\Phi_{lock}$ (see Algorithm 1), it remains to check if this condition is fulfilled indefinitely. A straightforward procedure would be to check after each cycle if $\mathcal{R}_{k+1} \subseteq \mathcal{R}_k$, meaning that $\mathcal{R}_k$ is an invariant. Checking $\mathcal{R}_{k+1} \subseteq \mathcal{R}_k$ is computationally expensive. This is because for this check, zonotopes have to be represented by polytopes and the enclosure check for polytopes is computationally expensive [1].

For this reason, we use the following alternative procedure illustrated in Fig. 7. First, the reachable set computations are continued for ϱ extra cycles after a reachable set fulfills the locking condition in cycle k_{lock}, see Fig. 7. Next, the reachable set $\mathcal{R}_{k_{lock}+\varrho}$ is overapproximated by an axis-aligned box denoted by $\mathcal{I}$. This leads to an overapproximation for the subsequent reachable sets, so ϱ should be chosen large enough such that all the subsequent sets fulfill the locking condition. Once a reachable set represented by a zonotope is enclosed by $\mathcal{I}$ (which is computationally cheap to detect), one can conclude that the PLL is locked indefinitely. This procedure is formalized in Algorithm 2.

Algorithm 2 Invariant computation

Input: $\mathcal{R}_{k_{lock}+\varrho}$, k_{lock}, ϱ, $\Delta\Phi_{lock}$.
Output: locked
$\mathcal{I} = \mathtt{box}(\mathcal{R}_{k_{lock}+\varrho})$
$\mathcal{R}_{k_{lock}+\varrho} = \mathcal{I}$
$k = k_{lock} + \varrho$
locked $= 0$; $P = \begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}$
while $|P\mathcal{R}_k| \leq \Delta\Phi_{lock}$ & locked==0 **do**
Compute $\mathcal{R}_{k+1}$ as shown in Algorithm 1
locked$= \begin{cases}1, & \text{if } \mathcal{R}_{k+1} \subseteq \mathcal{I} \\ 0, & \text{otherwise}\end{cases}$
$k = k + 1$
end while
$k_{final} = k - 1$

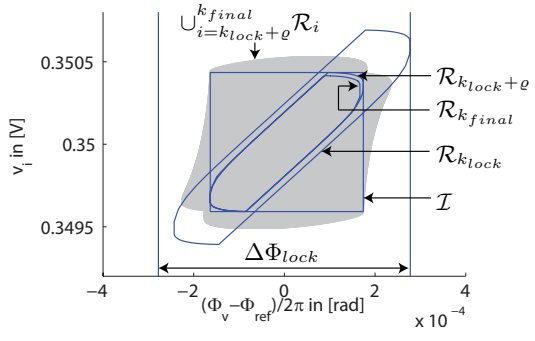


Figure 7: Reachable sets of different stages of the invariant computation.

6. NUMERICAL RESULTS

In this section we apply Algorithms 1 and 2 to verify a 27GHz PLL designed in 32nm CMOS SOI technology. The parameters of the PLL and the reachable set computation are listed in Table 1. The PLL considered here employs a simple initialization circuitry that sets the integral and proportional path voltages to common-mode levels at power up and whenever the division ratio is changed. This reduces locking time and aids the formal verification by reducing the uncertainty on the initial node voltages.

With the initialization, the initial range of node voltages are $v_i(0) \in [0.34, 0.36]$, and $v_{p1}(0), v_p(0) \in [-0.01, 0.01]$. We normalize the phases to $[0, 1]$, and we normalize the time to microseconds. The phase range of Φ_v is split into 5 subintervals $\Phi_v^i(0) \in -0.1 \cdot [i, i-1]$, where $i = 1 \ldots 5$, and without loss of generality we assume $\Phi_{ref}(0) = 0$. Because of symmetry, all possible initial phase differences are considered. The number of Taylor terms chosen depends on the time horizon. For $\Gamma(t_{cycle} - \underline{t}_{on})$, 30 Taylor terms are used and 10 Taylor terms are used for all other computations.

The reachable set starting with the initial phase difference $\Phi_v^1(0)$ is shown for the first 200 cycles in Fig. 8 for projections onto four different pairs of state variables. The sets computed to prove locking are shown in Fig. 7. Note that the voltages in Fig. 8 are as high as 10 [V] since charge pump saturation is not yet considered.

Table 2 shows the clock cycles it takes for the PLL to achieve locking for varying initial phase errors. The 1st and the 2nd columns show the results from reachability analysis. The 3rd column shows the maximum lock time obtained from 30 behavioral simulations with randomly varying initial phase errors and charge pump currents. Table 2 demonstrates that our reachability analysis efficiently provides an upper bound on the worst-case lock time in the presence of random phase error and charge pump current variations.

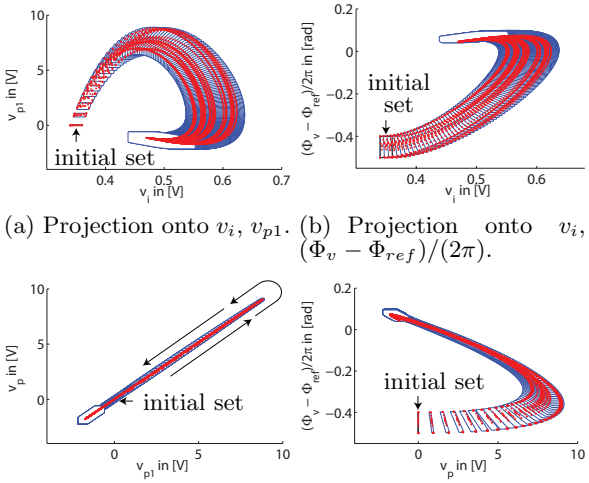
The computation times for the reachability analysis starting at different initial sets of phase differences are listed in Table 3. It can be seen that the results are obtained in less than a minute. The average computation time of the reachability analysis for a single cycle is around 27 [ms], which is only slightly longer than 24 [ms] required for a simulation of one cycle of the behavior model in MATLAB. All computations mentioned so far have been performed on an Intel i7 processor with 1.6 GHz and 6 GB memory. Simulating the behavioral model in VerilogA for a particular initial condition requires only 2 [ms] per cycle on an Intel Xeon CPU with 2.53GHz, which is an order of magnitude faster than reachability analysis. However, reachability analysis is still competitive if we consider that the VerilogA model needs to be simulated for thousands of Monte Carlo samples to capture random initial conditions and parameter variations.

Table 1: Parameters

PLL model			Reachable set comp.	
name	value	unit	name	value
f_{ref}	27	MHz	max zonotope	
f_0	26.93e3	MHz	order o	100
N	1000	—	$\underline{\omega}_1$	0
K_i	200	MHz/V	$\overline{\omega}_1$	0.7
K_p	25	MHz/V	$\underline{\omega}_3$	-4
I_i	[9.9,10.1]e-6	A	$\overline{\omega}_3$	12
I_p	[495,505]e-6	A	ϱ	100
C_i	25e-12	F		
C_{p1}	6.3e-12	F		
C_{p3}	2e-12	F		
R_{p2}	50e3	Ohm		
R_{p3}	8e3	Ohm		
t_d	50e-12	s		

7. CONCLUSIONS

This paper presents a method for verifying PLL locking using efficient reachability analysis. Efficient reachability computations are achieved using a discrete-time linear model with uncertain parameters and continuization to eliminate the complexity of switching. In contrast to applying a classical reachability approach, the intersection of guard sets can be dropped. As a consequence, only operations on sets remain for using zonotopes, which have a maximum complexity of $\mathcal{O}(n^3)$ with respect to the system dimension n. The verification of locking does not require any Lyapunov function to show convergence. For future work, we plan to consider saturations of charge pumps and varactor nonlinearities. We are also looking at other applications of continuization for hybrid systems where the transition time can be accurately overapproximated by a linear function of the state plus uncertainty.



(a) Projection onto v_i, v_{p1}. (b) Projection onto v_i, $(\Phi_v - \Phi_{ref})/(2\pi)$.

(c) Projection onto v_{p1}, v_p. (d) Projection onto v_p, $(\Phi_v - \Phi_{ref})/(2\pi)$.

Figure 8: Reachable sets of the first 200 cycles. Simulation results of each cycle are plotted by dots.

Table 2: Required cycles for locking

	reachability analysis		simulation
$\Phi_v(0)$	cycles to guarantee locking	cycles to reach $\mathcal{I}$	(max.) cycles to reach $\mathcal{I}$
$[-0.5, -0.4]$	2039	1845	1271
$[-0.4, -0.3]$	1981	1787	1225
$[-0.3, -0.2]$	1908	1714	1173
$[-0.2, -0.1]$	1811	1616	1086
$[-0.1, 0]$	1652	1457	994

Acknowledgments

The authors acknowledge the support of the NSF Award CCF0926181 and the C2S2 Focus Center, one of six research centers funded under the Focus Center Research Program

Table 3: Computation times of the PLL. Computed number of cycles equals the left column of Table 2.

$\Phi_v(0) \in -0.1 \cdot$	$[5, 4]$	$[4, 3]$	$[3, 2]$	$[2, 1]$	$[1, 0]$
Comp. times in [s]	55.0	54.4	53.5	47.8	42.9

(FCRP), a Semiconductor Research Corporation entity.

8. REFERENCES

[1] M. Althoff. *Reachability Analysis and its Application to the Safety Assessment of Autonomous Cars*. Dissertation, Technische Universität München, 2010. http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20100715-963752-1-4.

[2] E. Asarin, T. Dang, G. Frehse, A. Girard, C. Le Guernic, and O. Maler. Recent progress in continuous and hybrid reachability analysis. In *Proc. of the 2006 IEEE Conference on Computer Aided Control Systems Design*, pages 1582–1587, 2006.

[3] E. Clarke, A. Donzé, and A. Legay. On simulation-based probabilistic model checking of mixed-analog circuits. *Formal Methods in System Design*, 36:97–113, 2010.

[4] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 2000.

[5] T. Dang, A. Donzé, and O. Maler. Verification of analog and mixed-signal circuits using hybrid system techniques. In Alan J. Hu and Andrew K. Martin, editors, *FMCAD*, volume 3312 of *Lecture Notes in Computer Science*, pages 21–36. Springer, 2004.

[6] W. Denman, B. Akbarpour, S. Tahar, M. H. Zaki, and L. C. Paulson. Formal verification of analog designs using MetiTarski. In *Formal Methods in Computer-Aided Design*, pages 93 – 100, 2009.

[7] G. Frehse. PHAVer: Algorithmic verification of hybrid systems past HyTech. *International Journal on Software Tools for Technology Transfer*, 10:263–279, 2008.

[8] G. Frehse, C. Le 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 *Proc. of the 23rd International Conference on Computer Aided Verification*, LNCS 6806, pages 379–395. Springer, 2011.

[9] G. Frehse, B. H. Krogh, and R. A. Rutenbar. Verifying analog oscillator circuits using forward/backward abstraction refinement. In Georges G. E. Gielen, editor, *DATE*, pages 257–262. European Design and Automation Association, Leuven, Belgium, 2006.

[10] F. M. Garder. *Phaselock Techniques*. John Wiley, Hoboken NJ, third edition edition, 2005.

[11] D. Grabowski, D. Platte, L. Hedrich, and E. Barke. Time constrained verification of analog circuits using model-checking algorithms. *Electronic Notes in Theoretical Computer Science*, 153(3):37–52, 2006.

[12] S. Gupta, B. H. Krogh, and R. A. Rutenbar. Towards formal verification of analog designs. In *ICCAD*, pages 210–217. IEEE Computer Society / ACM, 2004.

[13] W. Hartong, R. Klausen, and L. Hedrich. Formal verification for nonlinear analog systems: Approaches to model and equivalence checking. In R. Drechsler, editor, *Advanced Formal Verification*, pages 205–245. Springer US, 2004.

[14] T. Henzinger. *Verification of Digital and Hybrid Systems*, volume 170 of *NATO ASI Series F: Computer and Systems Sciences*, chapter The theory of hybrid automata, pages 265–292. Springer, 2000.

[15] K. Jones, V. Konrad, and D. Nicković. Analog property checkers: a DDR2 case study. *Formal Methods in System Design*, 36:114–130, 2010.

[16] R. P. Kurshan and K. L. McMillan. Analysis of digital circuits through symbolic reduction. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 10(11):1356–1371, 1991.

[17] S. Little, D. Walter, K. Jones, C. J. Myers, and A. Sen. Analog/mixed-signal circuit verification using models generated from simulation traces. *Int. J. Found. Comput. Sci.*, 21(2):191–210, 2010.

[18] S. Little, D. Walter, N. Seegmiller, C. J. Myers, and T. Yoneda. Verification of analog and mixed-signal circuits using timed hybrid Petri nets. In *Proc. of the 2nd International Conference on Automated Technology for Verification and Analysis*, pages 426–440, 2004.

[19] O. Maler and D. Nickovic. Monitoring temporal properties of continuous signals. In Yassine Lakhnech and Sergio Yovine, editors, *FORMATS/FTRTFT*, volume 3253 of *Lecture Notes in Computer Science*, pages 152–166. Springer, 2004.

[20] G. Al Sammane, M. H. Zaki, Z. J. Dong, and S. Tahar. Towards assertion based verification of analog and mixed signal designs using PSL. In *FDL*, pages 293–298. ECSI, 2007.

[21] G. Al Sammane, M. H. Zaki, and S. Tahar. A symbolic methodology for the verification of analog and mixed signal designs. In Rudy Lauwereins and Jan Madsen, editors, *DATE*, pages 249–254. ACM, 2007.

[22] A. Singh and P. Li. On behavioral model equivalence checking for large analog/mixed signal systems. In *Proc. of IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*, pages 55–61, 2010.

[23] S. Steinhorst and L. Hedrich. Advanced methods for equivalence checking of analog circuits with strong nonlinearities. *Formal Methods in System Design*, 36(2):131–147, 2010.

[24] S. K. Tiwary, A. Gupta, J. R. Phillips, C. Pinello, and R. Zlatanovici. First steps towards SAT-based formal analog verification. In *Proc. of the International Conference on Computer-Aided Design*, pages 1 –8, 2009.

[25] D. Walter, S. Little, C. Myers, N. Seegmiller, and T. Yoneda. Verification of analog/mixed-signal circuits using symbolic methods. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27(12):2223 –2235, December 2008.

[26] Z. Wang, N. Abbasi, R. Narayanan, M.H. Zaki, G. Al Sammane, and S. Tahar. Verification of analog and mixed signal designs using online monitoring. In *Mixed-Signals, Sensors, and Systems Test Workshop, 2009. IMS3TW '09. IEEE 15th International*, pages 1 –8, June 2009.

[27] M. H. Zaki, G. Al Sammane, S. Tahar, and G. Bois. Combining symbolic simulation and interval arithmetic for the verification of AMS designs. In *FMCAD*, pages 207–215. IEEE Computer Society, 2007.

[28] M. H. Zaki, S. Tahar, and G. Bois. Formal verification of analog and mixed signal designs: A survey. *Microelectronics Journal*, 39(12):1395 – 1404, 2008.