DOI:10.1145/2507771.2507783

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

By Matthias Althoff, Akshay Rajhans, Bruce H. Krogh, Soner Yaldiz, Xin Li, and Larry Pileggi

Abstract

We present a scalable and formal technique to verify locking time and stability for charge-pump phase-locked loops (PLLs). In contrast to the traditional simulation approach that only validates the PLL at a given operation condition, our proposed technique formally verified the PLL at all possible operation conditions. The dynamics of the PLL is described by a hybrid automaton, which incorporates the differential equations of the analog circuit elements as well as the switching logic of the digital circuit elements. Existing methods for computing reachable sets for hybrid automata cannot be used to verify the PLL model due to the large number of cycles required for locking. We develop a new method for computing effective overapproximations of the sets of states reached on each cycle by using uncertain parameters in a discrete-time model to represent the range of possible switching times, a technique we call *continuization*. Using this new method for reachability analysis, it is possible to verify locking specifications for a charge-pump PLL design for all possible initial states and parameter values in time comparable to the time required for a few simulation runs of the same behavioral model.

1. INTRODUCTION

In the standard design flow for analog mixed signal (AMS) circuits, the complete circuit is decomposed into its principal elements or *blocks*, which are first analyzed and designed using idealized low-order behavioral models. Detailed circuit-level designs are implemented only after the performance specifications have been verified at the block level over the required range of parameter variations and operating conditions. The goal is to create robust designs to avoid costly redesign cycles in the downstream process.

Because of the complexity of the mixed continuous and discrete (i.e., hybrid) AMS dynamics, there are no analytical techniques to verify a given design satisfies the circuit specifications, even for the simplified block-level behavioral models. Thus, numerical simulation has been the standard tool for evaluating the performance of behavioral models. Simulation is not completely satisfactory, however, because each simulation run represents the behavior for only one set of values for the initial states and parameters, so many simulations are required to assess the robustness of the design. Moreover, some specifications can be verified only after simulations have run for very long durations, and some specifications such as stability cannot be confirmed with absolute certainty because simulations cannot be run indefinitely.

This paper demonstrates an alternative to simulation based on formal methods. Formal methods offer an attractive alternative to simulation because they can verify that specifications for a circuit are satisfied for all possible behaviors over entire ranges of initial states and parameter values. This corresponds to an infinite number of simulation runs of unbounded duration. In their survey of the literature on formal verification for AMS designs, Zaki et al. categorize the methods into equivalence checking, automated state-space exploration, run-time verification, and proof-based methods.[11] Reachability analysis, the technique developed in this paper, is a form of automated state-space exploration.

The basic idea of reachability analysis is to use the dynamic equations for the circuit to propagate the trajectories of entire sets of states over time, rather than just a single state trajectory. The key issues are how to represent sets of states numerically and how to propagate these sets efficiently. Good techniques have been developed to represent and compute reachable sets for continuous dynamic systems (see e.g. Althoff [1] and Girard et al.[8]). All of these techniques are based on overapproximations, since the actual sets of reachable states are not convex in general. These overapproximations become less accurate as time progresses, however, and for hybrid dynamic systems the overapproximations become even less accurate and more time consuming to compute due to the need to compute overapproximations of intersections of reachable sets with the surfaces representing switching conditions.[2, 6] Therefore, current reachability analysis techniques for hybrid systems are effective when there are only a few discrete transitions in the time interval of interest.

To demonstrate the applicability of formal methods and reachability analysis to AMS circuits, we consider the verification of block-level behavioral models for a class of phase-locked loops (PLLs). PLLs are integrated circuits that produce high-frequency output signals that are synchronized to and in phase with low-frequency reference signals. Originally developed in the 1930's as a circuit for radio receivers, millions of PLLs are now used in virtually all digital

The original version of this paper was published in the *Proceedings of the International Conference on Computer Aided Design*, 2011, pp. 659–666.

communication systems, from satellites to mobile phones, as well as in many other applications such as clock generation for microprocessors. The charge-pump PLL is one of the popular PLL architectures.[7] It is an AMS circuit: the error signal driving the analog feedback is generated by digital logic.[7]

The primary requirements to be verified for a PLL are the circuit's *locking time* and *stability*. These specifications are illustrated in Figure 1. Locking time is a *transient specification*: the PLL state must reach the invariant region within a specified number of cycles. Stability is an *invariant specification*: from some set of initial states, the magnitude of the phase difference must remain within a given bound indefinitely. Both of these specifications must be achieved robustly, that is, from an arbitrary initial state and over a range of parameter values that reflect the target operating conditions (e.g., a given temperature range) as well as the inherent uncertainties that will arise from the detailed design and manufacturing processes. Verifying the behavioral model of a PLL using simulation is time consuming and ultimately inconclusive because: (i) locking can take a few thousand cycles, so very long simulation runs are required; (ii) each simulation run represents the behavior for only one set of values for the initial states and parameters, so many simulations are required to assess the robustness of the design; and (iii) invariance can only be inferred, but not guaranteed, because simulations cannot be run indefinitely.

We present a method for verifying both the transient and invariant specifications for a PLL over entire ranges of initial states and parameter values using reachable set computations that can be performed in the same amount of time currently required to simulate the circuit models for just a few selected points in the design space. Our approach relies on some new techniques tailored to the PLL problem because locking can require thousands of cycles, which implies there will be thousands of discrete transitions in the switching logic. Experiments with existing methods implemented in tools such as PHAVer[5] or SpaceEx[6] show that the overapproximations using existing methods become inaccurate so quickly that it is impossible to demonstrate that locking occurs, even for simple cases where locking can be demonstrated analytically.

The main technical contribution of this paper is a new method for computing accurate overapproximations of reachable sets for hybrid systems when there are a large number of discrete state transitions. This approach leverages previous results on computing reachable sets for linear systems with bounded uncertain parameters. Using the equations that govern the continuous dynamics of the PLL, we create a discrete-time model that generates tight overapproximations of the reachable sets at the beginning of each continuous-time cycle. Since the actual times at which the discrete transitions occur can vary, we introduce bounded uncertain parameters in the linear discrete-time model that account for the variations in the actual transition times. We call this process of mapping variations in time into parameter uncertainties *continuizaton*.[3] Finally, we show that satisfaction of the PLL specifications for the discrete-time model guarantees the specifications are satisfied at all points in time. The reachable sets for the discrete time model can be computed very fast, and the time reduced further by taking advantage of certain symmetries in the PLL dynamics. Our approach illustrates how the successful use of formal methods to solve real problems often requires extensions and insights that exploit the particular structure and features of the target application. It is an enabling technique that facilitates us to efficiently verify a PLL at all possible operation conditions.

We begin in the next section by showing how a class of charge-pump PLLs can be modeled at the behavioral level using hybrid automata with uncertain parameters. Section 3 presents a conversion of the continuous-time behavioral model to a discrete-time model, which provides the solution of the original model after each cycle. Variations in switching times of the PLL are abstracted away in Section 4 using the new concept of continuization. This makes it possible to abstract the hybrid dynamics of the PLL by a linear system with uncertain parameters. Using the model resulting from continuization, Section 5 presents the application of reachability analysis for formal verification of the PLL specifications, and Section 6 presents a comparison of the verification results using reachability to the classical simulation approach. The concluding section summarizes the contributions of this paper.

2. PLL BEHAVIORAL MODEL

We consider the dual path, type II, third-order charge-pump PLL shown in Figure 2, consisting of a reference signal generator (Ref), a voltage-controlled oscillator (VCO), a phase frequency detector (PFD), and charge pumps (CPs), along with RC circuits to implement a PI controller for the

Figure 1. Transient (locking time) and invariant (stability) specifications for a PLL.

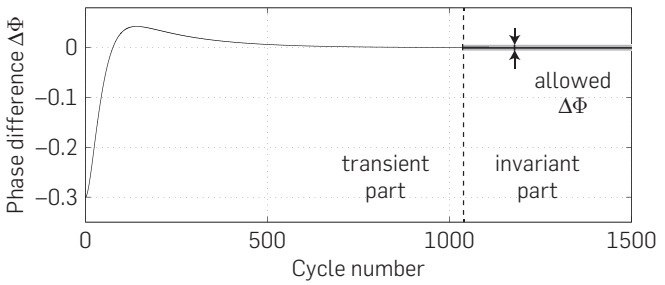
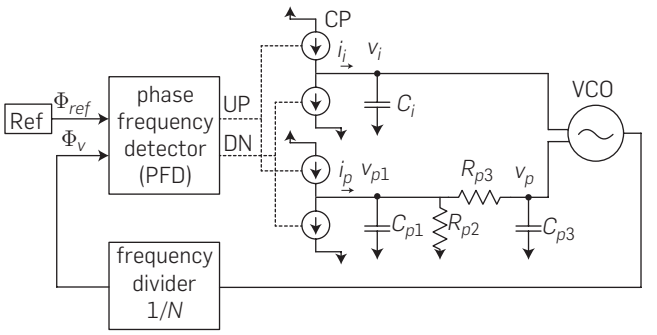


Figure 2. Dual-path charge-pump PLL.



feedback loop. The reference frequency generator produces a sinusoidal signal at a fixed low frequency (MHz), and the VCO generates a high-frequency signal (GHz). The desired output frequency of the VCO is determined by the reference frequency and the frequency divider ratio (i.e., N). The purpose of the PLL 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.

Locking of the PLL is achieved by the PFD by comparing the phases of the reference signal and the VCO signal and setting the signals UP = 1 if the reference 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. For instance, if the reference signal leads, it means that the reference signal is faster than the VCO signal (when divided by N). In this case, UP is set to 1 and the "up" current will charge the capacitors so that the voltage values v_i and v_p increase. As a result, the VCO frequency increases in order to catch the reference signal. We do not consider adaptation of PLL parameters such as the frequency divider, resistor, or capacitor values.

As one can see from Figure 2, different components of the PLL system operate at different frequencies. For instance, the reference signal is at low frequency, while the VCO signal may be at extremely high frequency if the frequency divider ratio N is large. The large difference in frequency makes PLL simulation extremely challenging, since a traditional simulation tool must adopt a very small time step to numerically solve the PLL response in time domain. It, in turn, results in a very long simulation time.

The behavioral model of the charge-pump PLL is a hybrid automaton[4] with linear continuous dynamics and uncertain parameters. Appropriate bounds on the uncertain parameters can be determined by equivalence checking with detailed circuit models.[9,10] 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 Figure 2). 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 are given in Figure 2 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

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

The output signals of the PFD are determined by threshold crossings of phase signals. The switching logic is described by the automaton shown in Figure 3, where the states are labeled as *up_active*, *dn_active*, *both_active*, and *both_off*.

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 *dn_active* when $\Phi_v = 2\pi$ is reached first. As shown in Figure 4, 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 *dn_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,

Figure 3. Hybrid automaton.

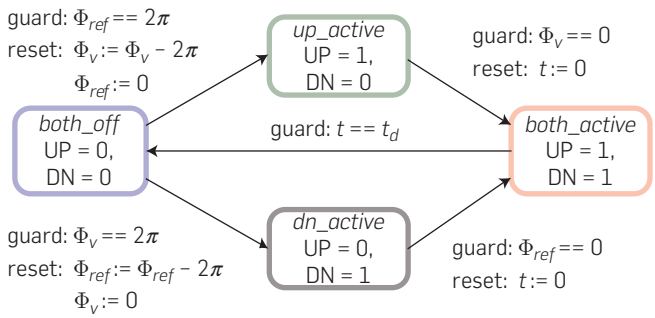
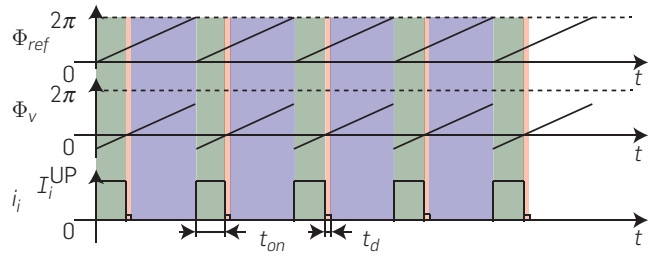


Figure 4. Typical charge pump activity.



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°].

3. TIME DISCRETIZATION

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 the reference signal, assuming the reference signal leads the VCO signal, that is, for the discrete state sequence *up_active* → *both_active* → *both_off*. 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

$$x^p(r) = \int_0^r e^{A(r-t)}dt\, u \in \underbrace{\Big(\sum_{i=0}^{\eta} \frac{1}{(i+1)!} A^i r^{i+1} \oplus \mathcal{E}^p(r)\Big)}_{=:\Gamma(r)} \otimes u, \quad (2)$$

where $\mathcal{A} \oplus \mathcal{B} = \{a + b | a \in \mathcal{A}, b \in \mathcal{B}\}$ is a Minkowski addition and $\mathcal{A} \otimes \mathcal{B} = \{ab | a \in \mathcal{A}, b \in \mathcal{B}\}$ a set-based multiplication. 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. The set of remainders $\mathcal{E}^p(r)$ is overapproximated by an interval matrix, that is, a matrix with lower and upper bounds on each element, as presented in Althoff et al.[3] 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

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

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

4. CONTINUIZATION

The model derived above computes the state of the system after one cycle when the switching time of the charge pumps is known. In this section, we develop a model that computes the range of state values that can occur at each cycle, for the entire range of possible switching times 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 only on $\Phi_v = x_4$, it is sufficient to consider (see (1))

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

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 $\dot{x}_4 \in [\underline{\upsilon}, \overline{\upsilon}]$. We further extract the bound $[\underline{\delta}, \overline{\delta}]$ on x_4 from the reachable set at the beginning of each cycle. We obtain $t_{on} \in [\underline{t}_{on}, \overline{t}_{on}] = \{x_4 / \dot{x}_4 \,|\, x_4 \in [\underline{\delta}, \overline{\delta}], \dot{x}_4 \in [\underline{\upsilon}, \overline{\upsilon}]\}$ using the fact that the reference signal is leading ($\underline{\delta}, \overline{\delta} < 0$), resulting in

$$\underline{t}_{on} = |\,\delta\,|/\overline{\upsilon}, \quad \overline{t}_{on} = |\,\underline{\delta}\,|/\underline{\upsilon}. \quad (5)$$

Using bounds on the switching times derived above, we use the concept of continuization to compute the set of reachable states resulting from uncertain switching times. To compute the reachable set under uncertain switching times (see Figure 5), we modify (3) and compute the solution successively, first of $\tilde{x}_k$ at times $t_k + \underline{t}_{on}$, then of $\tilde{x}_{k+1}$ at times t_{k+1}. This makes it possible to reset the uncertain switching time to values in the interval $[0, \Delta t_{on}]$, $\Delta t_{on} = \overline{t}_{on} - \underline{t}_{on}$ compared to $[\underline{t}_{on}, \overline{t}_{on}]$, which has computational benefits when evaluating Taylor series since higher order terms can be tightly bounded. The new equations are:

$$\begin{aligned} \tilde{x}_k &\in e^{A\underline{t}_{on}} x_k \oplus \Gamma(\underline{t}_{on})c \oplus \Gamma(\underline{t}_{on})u \\ x_{k+1} &\in e^{A(t_{cycle}-\underline{t}_{on})}\tilde{x}_k \oplus \Gamma(t_{cycle} - \underline{t}_{on})c \\ &\quad \oplus e^{A(t_{cycle}-\overline{t}_{on})} \underbrace{\left\{e^{A(\Delta t_{on}-\tilde{t})}\Gamma(\tilde{t}) \,\middle|\, \tilde{t} \in [0, \Delta t_{on}]\right\}}_{=:\mathcal{G}(\Delta t_{on})} u \\ &\quad \oplus e^{A(t_{cycle}-\overline{t}_{on}-t_d)} \underbrace{\left\{e^{A\tilde{t}} \,\middle|\, \tilde{t} \in [0, \Delta t_{on}]\right\}}_{=:\tilde{M}(\Delta t_{on})} \Gamma(t_d)u_d. \end{aligned} \quad (6)$$

We drop the cycle index k for the following deviations for simplicity. It is possible to extract the time $\tilde{t}$ from the set $\mathcal{G}(\Delta t_{on})$, to obtain the relatively tight inclusion

$$\mathcal{G}(\Delta t_{on}) \subseteq \left\{\tilde{t}\mathcal{C}(\Delta t_{on}) \,\middle|\, \tilde{t} \in [0, \Delta t_{on}]\right\},$$

where $\mathcal{C}(\Delta t_{on})$ is an interval matrix derived in Althoff et al.[3] Combining this result with $\tilde{t} \in \frac{|\,\tilde{x}_4\,|}{[\underline{\upsilon}, \overline{\upsilon}]}$ using the idea in (5) yields

Figure 5. Range of times $[\underline{t}_{on}, \overline{t}_{on}]$ when the charge pump is switched off. The mode *both_active* is not considered in this figure.

$$\mathcal{G}(\Delta t_{on}) \subseteq \mathcal{C}(\Delta t_{on}) \frac{|\tilde{x}_4|}{[\underline{v}, \overline{v}]}.$$

Now, the expression $e^{A(t_{cycle}-\overline{t}_{on})}\mathcal{G}(\Delta t_{on})u$ in (6) is written in terms of $\tilde{x}$, independent of the current mode. Thereto, we use the fact that u only changes sign between modes based on the phase difference, which is taken care of by redefining the input $u := \text{sgn}(x_4(t_k))(-u)$. We also consider that $u \in \mathcal{U}$, where $\mathcal{U}$ is an interval vector, such that

$$\begin{aligned} & e^{A(t_{cycle}-\overline{t}_{on})}\mathcal{G}(\Delta t_{on})u \\ & \subseteq -e^{A(t_{cycle}-\overline{t}_{on})}\mathcal{C}(\Delta t_{on}) \frac{|\tilde{x}_4|}{[\underline{v}, \overline{v}]} \text{sgn}(\tilde{x}_4)\mathcal{U} \\ & = \underbrace{\left[\mathbf{0} \quad \mathbf{0} \quad \mathbf{0} \quad -\frac{1}{[\underline{v}, \overline{v}]} e^{A(t_{cycle}-\overline{t}_{on})}\mathcal{C}(\Delta t_{on})\mathcal{U}\right]}_{:=\Theta(\Delta t_{on})} \otimes \tilde{x}, \end{aligned} \tag{7}$$

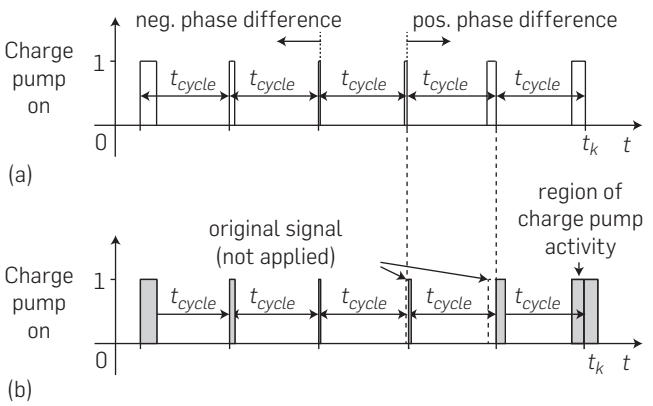
where $\mathbf{0}$ represents zero vectors of proper dimension and $\Theta(\Delta t_{on})$ is an interval matrix.

The result of (7) holds no matter if the phase difference is positive or negative 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 *dn_active* is also considered, the bounds are

$$\underline{t}_{on} = 0, \quad \overline{t}_{on} = \max(|\underline{\delta}|/\overline{v}, |\underline{\delta}|/\underline{v}, |\overline{\delta}|/\underline{v}, |\overline{\delta}|/\overline{v}). \tag{8}$$

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 *dn_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 contained 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 Figure 6.

Figure 6. Consideration of inputs when the phase difference changes from negative to positive. (a) Signal of charge pump activity; (b) signal used for reachability analysis up to time t_k.



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.

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

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

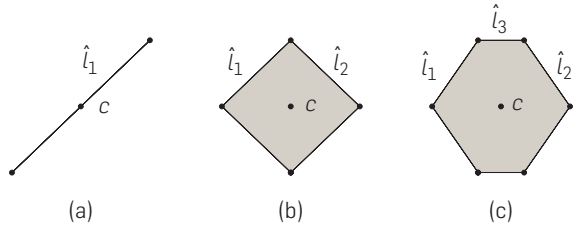
where $c \in \mathbb{R}^n$ is the zonotope center (to which a zonotope is centrally symmetric) and the $g^{(i)} \in \mathbb{R}^n$ are called generators. The order of a zonotope is defined as $o = \frac{p}{n}$. Figure 7 illustrates a zonotope being constructed step-by-step as the Minkowski sum of a finite 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 Althoff.[1]

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 $I_i^{\text{UP}} = -I_i^{\text{DN}}$ and $I_p^{\text{UP}} = -I_p^{\text{DN}}$, which can be relaxed for reachability analysis by choosing the intervals for $I_i^{\text{UP}} + I_i^{\text{DN}}$ and $I_p^{\text{UP}} + I_p^{\text{DN}}$ 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 the phases Φ_{ref} and Φ_v 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 and remove x_5 for the reachability computations.

Figure 7. Construction of a zonotope by Minkowski addition of line segments. (a) $c \oplus \hat{l}_1$ (b) $c \oplus \hat{l}_1 \oplus \hat{l}_2$ (c) $c \oplus \hat{l}_1 \oplus \hat{l}_2 \oplus \hat{l}_3$



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

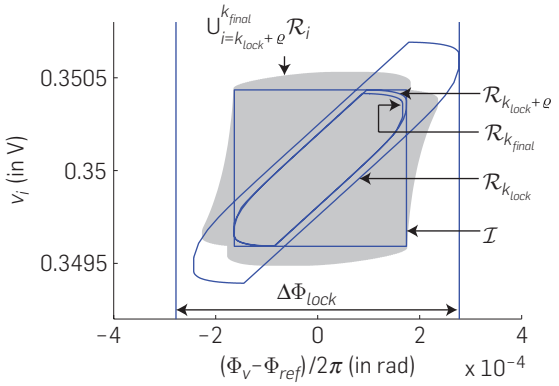
Input: Initial set $\mathcal{R}_0$, system matrix A, input set $\mathcal{U}$, input set $\tilde{\mathcal{U}}$ for *both_active*
parameters: t_{cycle}, $\Delta\Phi_{lock}$, $\underline{\upsilon}$, $\overline{\upsilon}$
Output: $\mathcal{R}_{k_{lock}}$
$k = 0$; $P = \begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}$; $\tilde{\mathcal{R}}_0 = \mathcal{R}_0$
while $|P\mathcal{R}_k| > \Delta\Phi_{lock}$ **do**
$\underline{t}_{on} = \min(|P\mathcal{R}_k|)/\overline{\upsilon}$, $\overline{t}_{on} = \max(|P\mathcal{R}_k|)/\underline{\upsilon}$
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 (7)
Compute $\tilde{M}(\Delta t_{on})$; see (6)
$\tilde{\tilde{\mathcal{R}}}_{k+1} = e^{A\underline{t}_{on}}\tilde{\mathcal{R}}_k \oplus \Gamma(\underline{t}_{on})c \oplus \Gamma(\underline{t}_{on})\mathcal{U}$
$\tilde{\mathcal{R}}_{k+1} = \left(e^{A(t_{cycle}-\underline{t}_{on})} \oplus \Theta(\Delta t_{on})\right)\tilde{\tilde{\mathcal{R}}}_{k+1}$
$\oplus\Gamma(t_{cycle} - \underline{t}_{on})c \oplus e^{A(t_{cycle}-\overline{t}_{on}-t_d)}\tilde{M}(\Delta t_{on})\Gamma(t_d)\tilde{\mathcal{U}}$
$\mathcal{R}_{k+1} = \tilde{\mathcal{R}}_{k+1} \oplus \Theta(\Delta t_{on})\tilde{\mathcal{R}}_{k+1}$ (due to 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 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 Figure 8. First, the reachable set computations are continued for ϱ extra cycles after a reachable set fulfills the locking condition in cycle k_{lock}, see Figure 8. 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 $\mathcal{R}_{k_{final}}$ represented by a zonotope is enclosed by $\mathcal{I}$ (which is computationally cheap to detect), one can conclude that the PLL is locked indefinitely.

Figure 8. Reachable sets of different stages of the invariant computation.



6. NUMERICAL RESULTS

We apply Algorithm 1 and the invariant computation to verify a 27 GHz PLL designed in 32 nm CMOS SOI technology. Note that the PLL was designed in a commercial process at an advanced technology node. Hence, it provides a practical example to demonstrate the efficacy of our proposed verification method. 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 aforementioned experiment setup allows us to formally verify the PLL with consideration of initial voltage and phase uncertainties. Note that these uncertainties cannot be efficiently incorporated into the traditional simulation approach, as a traditional simulation can only validate the PLL with a specific initial condition.

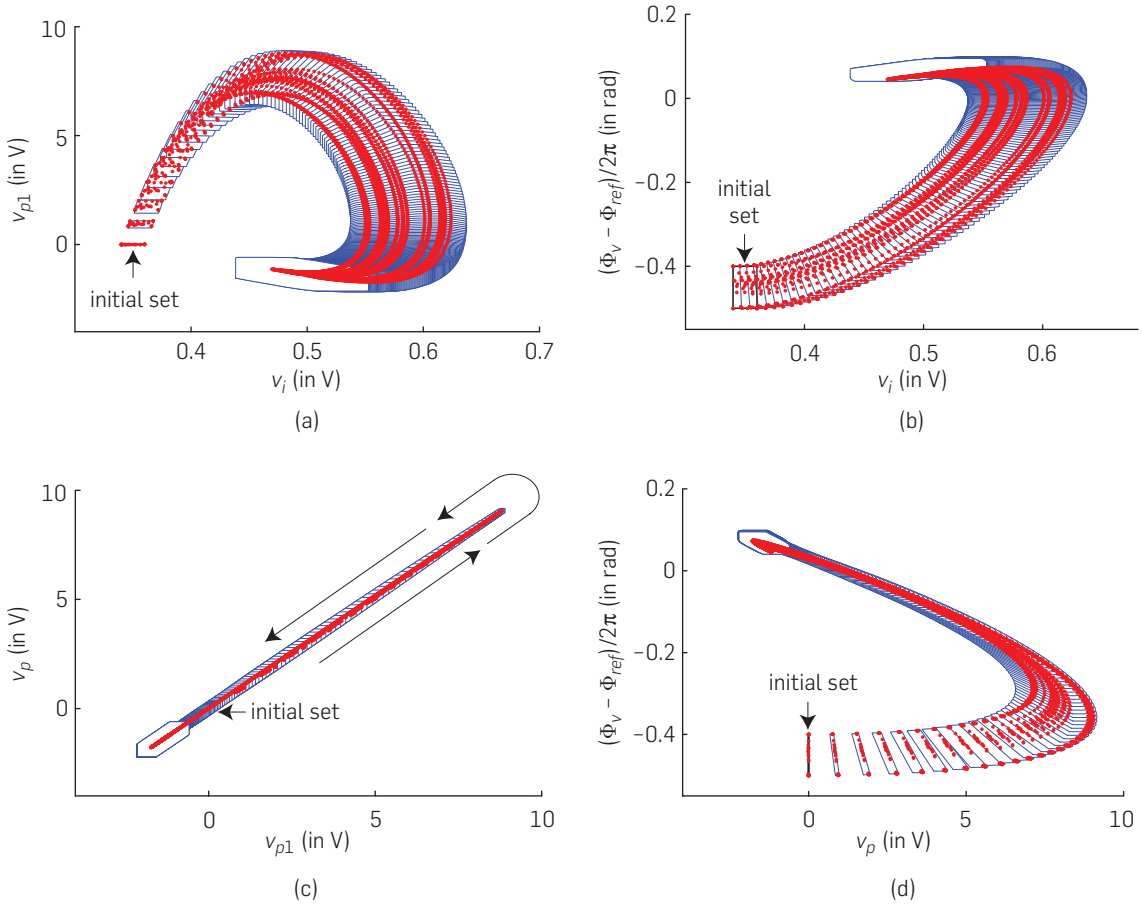
The reachable set starting with the initial phase difference $\Phi_v^1(0)$ is shown for the first 200 cycles in Figure 9 for projections onto four different pairs of state variables. The sets computed to prove locking are shown in Figure 8. In this example, the proposed verification algorithm is able to prove that independent of the initial condition, the PLL reliably locks to the reference signal. Note that the voltages in Figure 9 are as high as 10 [V] since charge pump saturation is not yet considered. It is possible to further extend our verification method to consider charge pump saturation by applying a nonlinear behavior model.

Table 2 shows the clock cycles it takes for the PLL to

Table 1. Parameters.

PLL model			Reachable set comp.	
Name	**Value**	**Unit**	**Name**	**Value**
f_{ref}	27	MHz	Max zonotope order o	100
f_0	26.93e3	MHz		0
N	1000	–	$\underline{\omega}_1$	0.7
K_i	200	MHz/V	$\overline{\omega}_1$	−4
K_p	25	MHz/V	$\underline{\omega}_3$	12
I_i	[9.9,10.1]e−6	A	$\overline{\omega}_3$	100
I_p	[495,505]e−6	A		
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		

Figure 9. The blue regions show the reachable set of each cycle for the first 200 cycles. Simulation results of each cycle are plotted by red dots. (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; and (d) projection onto v_p, $(\Phi_v - \Phi_{ref})/(2\pi)$.



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. We use the maximum lock time since the verification task is to check if the PLL always locks before a specified locking time, that is, we are investigating the worst-case behavior. Note that we are not providing any stochastic evaluation since this is not the focus of this work. 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. On the other hand, it is important to note that the traditional approach based on Monte Carlo simulation cannot guarantee to find the "true" maximum lock time. Unless an infinite number of Monte Carlo runs are performed, the maximum lock time may not be captured by one of the Monte Carlo runs.

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 2.53 GHz, which is an order of magnitude faster than reachability analysis. However, reachability analysis is still

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

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

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

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.

7. CONCLUSION

This paper presented 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, the only operations on sets that remain, can be performed 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. In addition to PLL, the proposed reachability analysis may be further extended to verify the circuit functionality and performance specifications of other AMS systems in time domain.

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 (FCRP), a Semiconductor Research Corporation entity. C

References

1. Althoff, M. 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. Althoff, M., Krogh, B.H. Avoiding geometric intersection operations in reachability analysis of hybrid systems. In *Hybrid Systems: Computation and Control* (2012), 45–54.
3. Althoff, M., Rajhans, A., Krogh, B.H., Yaldiz, S., Li, X., Pileggi, L. Formal verification of phase-locked loops using reachability analysis and continuization. In *Proceedings of the International Conference on Computer Aided Design* (2011), 659–666.
4. Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T., Ho, P., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S. The algorithmic analysis of hybrid systems. *Theoretical Computer Science, 138* (1995), 3–34.
5. Frehse, G. PHAVer: Algorithmic verification of hybrid systems past HyTech. *Int. J. Software Tool. Tech. Tran. 10* (2008), 263–279.
6. Frehse, G., Guernic, C.L., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., Maler, O. SpaceEx: Scalable verification of hybrid systems. In *Proceedings of the 23rd International Conference on Computer Aided Verification* (2011), LNCS 6806, Springer, 379–395.
7. Garder, F.M. Phaselock Techniques, third edn, John Wiley, Hoboken, NJ, 2005.
8. Girard, A., Le Guernic, C., Maler, O. Efficient computation of reachable sets of linear time-invariant systems with inputs. In *Hybrid Systems: Computation and Control* (2006), LNCS 3927, Springer, 257–271.
9. Singh, A., Li, P. On behavioral model equivalence checking for large analog/mixed signal systems. In *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (2010), 55–61.
10. Steinhorst, S., Hedrich, L. Advanced methods for equivalence checking of analog circuits with strong nonlinearities. *Formal Meth. Syst. Des. 36*, 2 (2010), 131–147.
11. Zaki, M.H., Tahar, S., Bois, G. Formal verification of analog and mixed signal designs: A survey. *Microelectron. J.39*, 12 (2008), 1395–1404.

Matthias Althoff (matthias.althoff@tu-ilmenau.de), Ilmenau University of Technology, Ilmenau, Germany.

Akshay Rajhans, Bruce H. Krogh, Xin Li, and Larry Pileggi ([arajhans, krogh, xinli, and pileggi]@ece.cmu.edu), Carnegie Mellon University, Pittsburgh, PA.

Soner Yaldiz [soner.yaldiz@intel.com], Intel Corporation, Hillsboro, OR.