Verifying nonlinear analog and mixed-signal circuits with inputs

Chuchu Fan ∗ Yu Meng ∗ Jürgen Maier ∗∗ Ezio Bartocci ∗∗ Sayan Mitra ∗ Ulrich Schmid ∗∗

∗ University of Illinois at Urbana-Champaign, (emails: {cfan10, yumeng5, mitras}@illinois.edu)
∗∗ Technische Universität Wien, (emails: {jmayer, s}@ecs.tuwien.ac.at, ezio.bartocci@tuwien.ac.at)

Abstract: We present a new technique for verifying nonlinear and hybrid models with inputs. We observe that once an input signal is fixed, the sensitivity analysis of the model can be computed much more precisely. Based on this result, we propose a new simulation-driven verification algorithm and apply it to a suite of nonlinear and hybrid models of CMOS digital circuits under different input signals. The models are low-dimensional but with highly nonlinear ODEs, with nearly hundreds of logarithmic and exponential terms. Some of our experiments analyze the metastability of bistable circuits with very sensitive ODEs and rigorously establish the connection between metastability recovery time and sensitivity.

1. INTRODUCTION

Analog and mixed-signal circuits have provided a well-spring of hard problem instances for formal verification of hybrid systems (HS). Tools like HyTech (Henzinger et al., 1997), PHAVer (Frehse, 2008), SpaceEx (Frehse et al., 2011), Checkmate (Gupta et al., 2004), \( \frac{d}{dt} \) (Dang et al., 2004), and Coho (Yan and Greenstreet, 2008) have targeted and successfully verified linear dynamical and hybrid models for tunnel-diode oscillators (Lata and Jamaagni, 2010), \( \Delta \Sigma \) modulators (Gupta et al., 2004; Dang et al., 2004), filtered oscillators (Frehse et al., 2011), and digital arbiters (Yan and Greenstreet, 2008). Only recently, verification tools such as Flow∗ (Chen et al., 2013), NLToolbox (Dang et al., 2009), iSAT (Fränzle et al., 2007), dReach (Kong et al., 2015), C2E2 (Fan et al., 2016) and CORA (Althoff and Grebchenuk, 2016), have demonstrated the feasibility of verifying nonlinear dynamic and hybrid models. These tools are still limited in terms of the complexity of the models and the type of external inputs they can handle, and they require quite often manual tuning of algorithmic parameters. The verification challenge for nonlinear circuits is further exacerbated by the fact that these problems often require state exploration in regions, where the model is very sensitive. For example, bi-stable circuits like a storage element or a flip-flop can be driven into a metastable state where the circuit may experience the same input (Theorem 6). Using this new algorithm for discrepancy computation, we have generalized the verification algorithm to handle nonlinear hybrid models with inputs. This approach was later inte-
Supremum of the distance between any pair of points in $l^\infty$ matrices:

$$\sup_{A, B \in L^\infty} \|A - B\|_\infty$$

**Notations**

$B_\delta(x)$ is the Minkowski sum of sets $B$ and $\delta$-balls around $x$. For a pair of matrices $A$ and $B$, $A \subseteq B$ if $A$ is a subset of $B$. $B_{\delta}(S) = \bigcup_{x \in S} B_\delta(x)$, the Minkowski sum of sets $B$ and $S$. $B_\delta(x)$ is the closed $\delta$-ball around $x$. $\delta$-balls around sets are defined as $B_\delta(S) = \bigcup_{x \in S} B_\delta(x)$. $S \oplus S'$ is the Minkowski sum of sets $S$ and $S'$. For a real matrix $A$, $\text{diag}(A)$ is the diagonal matrix, and $\text{eig}(A)$ is the set of eigenvalues of $A$. $\text{diag}(A)$ is the largest eigenvalue of $A$. For a pair of matrices $A, \bar{A}$ with $A_{ij} \leq (\bar{A})_{ij}$ for all $1 \leq i, j \leq n$, we define the interval matrix as the set of matrices:

$$[A, \bar{A}] \triangleq \{ A \in \mathbb{R}^{n \times n} | (A)_{ij} \leq (\bar{A})_{ij}, 1 \leq i, j \leq n \}.$$

**2. SIMULATION-DRIVEN VERIFICATION**

**Notations**

For a real vector $x \in \mathbb{R}^n$, $\|x\|$ denotes its $l^\infty$ norm. For a set $S \subset \mathbb{R}^n$, the diameter $\text{diam}(S)$ is the supremum of the distance between any pair of points in $S$. For $S \subset \mathbb{R}^n$, the closed $\delta$-ball around $x$. Closed $\delta$-balls around sets are defined as $B_\delta(S) = \bigcup_{x \in S} B_\delta(x)$. $S \oplus S'$ is the Minkowski sum of sets $S$ and $S'$. For a real matrix $A$, $\text{diag}(A)$ is the diagonal matrix, and $\text{eig}(A)$ is the set of eigenvalues of $A$. For a pair of matrices $A, \bar{A}$ with $(A)_{ij} \leq (\bar{A})_{ij}$ for all $1 \leq i, j \leq n$, we define the interval matrix as the set of matrices:

$$[A, \bar{A}] \triangleq \{ A \in \mathbb{R}^{n \times n} | (A)_{ij} \leq (\bar{A})_{ij}, 1 \leq i, j \leq n \}.$$

**2.1 Dynamic systems with inputs**

An $n$-dimensional dynamic system with $m$-dimensional input is described by an ordinary differential equation:

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

where $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is a continuously differentiable function, and a compact set $\Theta \subset \mathbb{R}^n$ of initial states. The input is an integrable function $u : [0, \infty) \to U$, where

1 Files can be found at [https://publish.illinois.edu/c2e2-tool/gate/](https://publish.illinois.edu/c2e2-tool/gate/).

**2.2 Safety verification problem**

Given an $n$-dimensional dynamic system, an input signal $u(t)$, a compact initial set $\Theta \subset \mathbb{R}^n$, an unsafe set $\text{unsafe} \subset \mathbb{R}^n$, and a time bound $T > 0$, the safety verification problem is to check whether $\text{Reach}_u(\Theta, [0,T]) \cap \text{unsafe} = \emptyset$.

Safety verification of nonlinear ODEs and hybrid models is difficult even in the absence of inputs. For closed models (without inputs), recently developed simulation-driven verification algorithms decide the safety verification question rigorously by combining numerical simulations with sensitivity analysis of the trajectories with respect to their initial states (Donzé, 2010; Duggirala et al., 2013; Fan et al., 2016). These approaches are most effective when the sensitivity of the solutions to initial states can be precisely approximated.

These existing approaches do not support models with inputs. The seemingly natural idea of explicitly modeling the input $u$ as a state variable, i.e., its controlling ODE, and then verifying the resulting closed model does not work. This is because inputs often model unstable signals—like the pulse $u$ in Example 1—and in such cases the trajectories of the resulting closed system will turn out to be extremely sensitive with respect to the initial states and render simulation-driven verification ineffective. In Example 1, if we treat $u$ as a state variable, the over-approximation reach set of $x_1(t)$ using C2E2 is shown in Figure 2. The (prohibitive) blow-up in the over-approximation is due to the unstable input $u = u(1.8 - 1.5u) + 0.0015$ that models the rising transition of the smoothed pulse.

![Fig. 1. Input signal $u(t)$ (left), corresponding trajectory of $x_1, x_2$ (center), and reach sets from $B_\delta((0.5, 0.24)^2)$](image1)

![Fig. 2. Over-approximation of $x_1(t)$ without separate input from state variables. Note the different scale compared to Figure 1.](image2)
Lemma 4. For any compact sets \( J \) and \( A \), \( \forall \) bounded, i.e., where the integral is component-wise.

Definition 2. Given an input signal \( u(t) \), a continuous function \( \beta_u : \mathbb{R}_{\geq 0} \times \mathbb{R} \rightarrow \mathbb{R}_{\geq 0} \) is a discrepancy function of (1) if

1. for any pair of states \( x, x' \in \mathbb{R}^n, \) and any time \( t \geq 0, \)
\[
\|\xi_u(x, t) - \xi_u(x', t)\| \leq \beta_u(\|x - x'\|, t),
\]
and
2. for any \( t, \lim_{\|x-x'\| \to 0^+} \beta_u(\|x-x'\|, t) = 0.\)

Definition 2 generalizes the discrepancy functions defined in Fan and Mitra (2015); Duggirala et al. (2013). Observe that at any time \( t \), the ball with radius \( \beta_u(\delta, t) \) centered at \( \xi_u(x_0, t) \) contains the reach set of (1) starting from \( B_\delta(x_0, t) \). Therefore, by bloating the simulation trajectories \( \xi_u(\cdot) \) using the corresponding discrepancy function, we can obtain reach set over-approximations. Several techniques for computing discrepancy functions for closed systems have been presented in the literature (see Fan and Mitra (2015) and the references therein). The technique discussed next works for open systems and exploits the fact that the input signal is fixed for trajectories starting from different initial states.

First, we introduce a basic result that follows from the high-order mean value theorem (Lemma 3) to connect the differential equation with its Jacobian matrices. Then we show that the terms of the Jacobian matrix with respect to the state variables are bounded over compact sets (Lemma 4). Using these two results, we establish that the distance between neighboring trajectories actually follows a differential equation related to the bound of the Jacobian matrix (Lemma 5). Finally, we prove that the upper bound on the largest eigenvalue of the symmetric part of the Jacobian provides us with a suitable discrepancy function (Theorem 6).

The Jacobian of \( f \) with respect to the state \( J_x \) and the input \( J_u \) are matrix-valued functions of all the first-order partial derivatives of \( f \):

\[
(J_x(x, u))_{ij} = \frac{\partial f_i(x, u)}{\partial x_j}; \quad (J_u(x, u))_{ij} = \frac{\partial f_i(x, u)}{\partial u_j}.
\]

The following lemma from Fan and Mitra (2015) relates \( f \) with its Jacobian matrices based on the generalized mean value theorem, see Fan and Mitra (2015) for the detailed proof.

Lemma 3. For any continuously differentiable vector-valued function \( f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n, x, r \in \mathbb{R}^n \) and \( u, w \in \mathbb{R}^m, \)
\[
f(x+r, u+w) - f(x, u) = \left( \int_0^1 J_x(x+sr, u+w)ds \right) \cdot r + \left( \int_0^1 J_u(x, u+\tau w)d\tau \right) \cdot w,
\]
where the integral is component-wise.

If \( f \) is continuously differentiable, all terms in the Jacobian matrix are continuous. Since the input signals are bounded, i.e., \( \forall t > 0, u(t) \in U \subseteq \mathbb{R}^m \), the Jacobian matrix \( J_x(x, u) \) over compact sets is also bounded:

Lemma 4. For any compact sets \( S, U \) there exists an interval matrix \([A, \bar{A}]\) such that
\[
\forall x \in S, u \in U, J_x(x, u) \in [A, \bar{A}].
\]

The lemma follows from the fact that each term of \( J_x(x, u) \) is a continuous function of \( x, u \), and over the compact domains \( S, U \), the function has a maximum and minimum value that defines the matrix pair \([A, \bar{A}]\). The bounds of such values can be computed for a broad class of nonlinear functions using optimization and interval arithmetic solvers.

Lemma 5. Fix an input signal \( u(t) \). Suppose there exists a compact convex set \( S \subseteq \mathbb{R}^n \) and a time interval \( [0, t_1] \) such that for any \( x \in \Theta, \forall t \in [0, t_1], \xi_u(x, t) \in S \). Then for any \( x, x' \in \Theta \), for any fixed \( t \in [0, t_1], \) the distance \( d_u(t) = \xi_u(x', t) - \xi_u(x, t) \) satisfies \( d_u(t) = A(t)d_u(t) \), for some \( A(t) \in [A, \bar{A}], \) where \([A, \bar{A}]\) is an interval matrix satisfying Lemma 4.

Proof. Using Lemma 3 we have the following:
\[
\begin{align*}
\dot{y}_u(t) &= \xi_u(x', t) - \xi_u(x, t) \\
&= f(\xi_u(x', t), u(t)) - f(\xi_u(x, t), u(t)) \\
&= \left( \int_0^1 J_x(\xi_u(x, t) + sy_u(t), u(t))ds \right) \cdot y_u(t) \\
&\quad + \left( \int_0^1 J_u(\xi_u(x, t), u(t))ds \right) \cdot (u(t) - u(t)) \\
&= \left( \int_0^1 J_x(\xi_u(x, t) + sy_u(t), u(t))ds \right) \cdot y_u(t) \quad (3)
\end{align*}
\]

Given a compact convex set \( S \) and bounded set \( U \), the interval matrix \([A, \bar{A}]\) satisfies the conditions of Lemma 4. For any fixed \( t \), \( \int_0^1 J_x(\xi_u(x, t) + sy_u(t), u(t))ds \) is a constant matrix. Because \( \xi_u(x, t) \), \( \xi_u(x', t) \) are contained in the convex set \( S \), according to the convexity assumption of \( S \), \( \forall s \in [0, 1], \xi_u(x, t) + sy_u(t) + s(\xi_u(x', t) - \xi_u(x, t)) \) is also contained in \( S \). Thus, at \( t \), \( J_x(\xi_u(x, t) + sy_u(t), u(t)) \in [A, \bar{A}] \). Since the integration is from 0 to 1, it is straightforward to check that also
\[
\int_0^1 J_x(\xi_u(x, t) + sy_u(t), u(t))ds \in [A, \bar{A}].
\]

We rewrite (3) as
\[
\dot{y}_u(t) = A(t)y_u(t), \quad A(t) \in [A, \bar{A}], \quad (4)
\]

which means that at any fixed time \( t \in [0, t_1], \) we always have \( y_u(t) = A(t)y_u(t) \), where \( A(t) \) is unknown but \( A(t) \in [A, \bar{A}] \). □

Using the differential equation in Lemma 5, we can get a discrepancy function by bounding the eigenvalues of \([A, \bar{A}];\)

Theorem 6. Fix the input signal \( u(t) \) for system (1). Suppose the assumptions in Lemma 5 hold, and \( \exists \gamma \in \mathbb{R} \) such that \( \forall A(t) \in [A, \bar{A}], \)
\[
eig(A^T(t) + A(t))/2 \leq \gamma;
\]
then for any \( x, x' \in \Theta \) and for any \( t \in [0, t_1], \)
\[
\|\xi_u(x, t) - \xi_u(x', t)\| \leq \|x - x'\| e^{\gamma t}.
\]

Proof. Fixing the two initial states \( x, x' \in \Theta \), from Lemma 5, we know that \( y_u(t) = A(t)y_u(t) \), for some \( A(t) \in [A, \bar{A}], \) where \( y_u(t) = \xi_u(x', t) - \xi_u(x, t) \). By differentiating \( \|y_u(t)\|^2 \), we have that for any fixed \( t \in [0, t_1], \)
the following steps: a) First, using the Lipschitz constant, 

\[ \frac{d}{dt} [y_u(t)] = y_u^T(t) y_u(t) + y_u^T(t) y_u(t) \]

\[ = y_u^T(t) \left( A(t)^T + A(t) \right) y_u(t). \]  

(6)

If \( \varepsilon \text{ig}(A^T(t) + A(t))/2 \leq \gamma \), then the eigenvalues of \( B = A(t)^T + A(t) - 2 \gamma I \), where \( I \) is the identity matrix, are all less or equal to zero, so \( B \) is negative semi-definite. Therefore, 

\[ y_u^T(t) \left( A(t)^T + A(t) - 2 \gamma I \right) y_u(t) \leq 0. \] Hence, (6) becomes 

\[ \frac{d\beta_u}{dt} \leq 2\gamma \| y_u(t) \|^2. \] After applying Grönwall’s inequality, we have 

\[ \| y_u(t) \| \leq \| y_u(0) \| e^{-\gamma t}, \forall t \in [0, t_1]. \] \( \square \)

Theorem 6 obviously provides a discrepancy function 

\[ \beta_u(x - x', t) = \| x - x' \| e^{-\gamma t}. \]

However, computing one \( \gamma \) for the entire time horizon is usually too conservative to be directly used. Algorithm 2 in Fan and Mitra (2015) provides a method for constructing a reachtube using one simulation trajectory and initial partition size \( \delta \) as input, and produces a sequence of coefficients that defines the piece-wise exponential discrepancy function. The algorithm consists of the following steps: a) First, using the Lipschitz constant, a coarse over-approximation of the reachable set up to a short time horizon \( T_0 \) is constructed. Let this set be \( S \). b) Compute the interval matrix \( [A, A] \), which bounds the possible values of the Jacobian matrix \( J_x(x, u) \). c) Compute the largest eigenvalue \( \varepsilon \text{ig} \left( (A + A) + (A + A)^T \right)/2 \). From this value, an upper bound \( \gamma \) of the eigenvalue of \( A + A^T/2 \) for all \( A \in [A, A] \) is computed using a theorem from matrix perturbation theory. d) The upper bound \( \gamma \) (possibly negative) defines the discrepancy function 

\[ \beta_u(\delta, t) = \beta_u^0(t_0) e^{\gamma(t-t_0)} \] 

over the simulation time interval \([t_0, t_0 + T_0]\), where \( \beta_u^0(t) \) is the previous piece of the discrepancy function. Using this piece-wise exponential discrepancy function, an over-approximation of the reachable set is finally computed.

Example 7. For Example 1, restricting \( x_1 \) to be within the range \([0.4, 0.6]\) and \( u \) to be within the range \([0.1, 0.2]\) provides \( J_x \in [A, A] \) for 

\[ A = \begin{bmatrix} -3.06 & 0.2 \\ -2.1 & 0.4 \end{bmatrix} \text{ and } A = \begin{bmatrix} -3.06 & 0.2 \\ -2.1 & 0.4 \end{bmatrix} \]

Using Algorithm 2 in Fan and Mitra (2015), we get that \( \gamma = -1.05 \) satisfies Equation (5). Therefore, 

\[ \beta_u(\| x - x' \|, t) = \| x - x' \| e^{-1.05t} \] 

is a discrepancy function for this system with fixed input \( u(t) \).

4. VERIFYING SYSTEM WITH FIXED INPUTS

To implement our novel method for computing discrepancy functions of open systems, the algorithm for simulation-driven verification (see Algorithm 1) published in Fan and Mitra (2015) can be used with minor modifications. For sake of completeness, we briefly discuss the key features here; for more details the reader is referred to (Fan and Mitra, 2015). Throughout this section, we fix an input signal \( u(t) \) for the system (1).

Function \( \text{Cover}() \) returns a set of triples \( \{(x, \delta, \epsilon)\} \), where \( x \)'s are sample states, the union of \( B_\delta(x) \) covers \( \Theta \) completely, and \( \epsilon \) is the precision of the simulation. Function \( \text{Bloat}() \) expands the simulation trace \( \psi \) by \( \beta_u \) to get the reachtube \( R = \{(O_i, t_i)\}^k_{i=1} \). That is, for each \( i = 1, \ldots, k, O_i \leftarrow \text{hull}(R_{t_{i-1}}, R_{t_i}) \cap \max_{t_{i-1} \leq t \leq t_i} \beta_u((\delta + \epsilon), t) \). From Theorem 6, it follows that \( \text{Bloat}(\psi, \delta, \epsilon) \) contains \( \text{Reach}_u(B_\delta(x), [0, T]) \). There are two important data structures used in Algorithm 1: \( C \) is a collection of the triples returned by \( \text{Cover}() \), which represents the subset of \( \Theta \) that has not yet been proved safe, and \( \text{STB} \) that stores the bounded-time reachtube.

Initially, \( C \) contains a singleton \( (x_0, \delta_0 = \text{dia}(\Theta, \epsilon_0)) \), where \( \Theta \subseteq B_{\delta_0}(x_0) \) and \( \epsilon_0 \) is a small positive constant. For each triple \( (x, \delta, \epsilon) \in C \), the \text{while-loop} from Line 3 checks the safety of the reachtube from \( B_\delta(x) \), which is computed in Line 5-6. \( \psi \) is a \((\epsilon, \tau, T)\)-simulation from \( x \) with input \( u(t) \), which is a sequence of time-stamped rectangles \( \{(R_t, t_i)\}^T_{t=0} \) and is guaranteed to contain the trajectory \( (x, \tau, T) \). Bloating the simulation result \( \psi \) by the discrepancy function \( \beta_u \) we get \( R, (B_\delta(x), T) \)-reachute with input \( u(t) \). If \( \text{R} \) is disjoint from \( \text{unsafe} \), then the reachtube from \( B_\delta(x) \) is safe and the corresponding triple can be safely removed from \( C \). If for some \( j, R_j \) (one rectangle of the simulation) is completely contained in the \( \text{unsafe} \) set, then we can get a counterexample of a trajectory that violates the safety property. Otherwise, the safety of \( \text{Reach}_u(B_\delta(x), [0, T]) \) is inconclusive and a refinement of \( B_\delta(x) \) is made with some smaller \( \delta \) and smaller \( \epsilon, \tau \).

Recall that the safety verification problem requires us to check whether \( \text{Reach}_u(\Theta, [0, T]) \cap \text{unsafe} = \emptyset \). If there exists some \( \epsilon > 0 \) such that \( B_\delta(\Theta, [0, T]) \cap \text{unsafe} = \emptyset \), we say the system is robustly safe. If there exists some \( \epsilon > 0, x \in \Theta \), such that \( B_\delta(\Theta, [0, T]) \cap \text{unsafe} \) for some \( R_i \) in the simulation from \( x \), then we say the system is robustly unsafe. The algorithm returns \( \text{SAFE} \) if \( \text{Reach}_u(\Theta, [0, T]) \) has no intersection with \( \text{unsafe} \), along with a robustly safe reachtube \( \text{STB} \). It returns \( \text{UNSAFE} \) upon finding a counter-example, i.e., the simulation \( \psi \) with an interval fully contained in \( \text{unsafe} \).

According to Theorem 6, if \( \delta \) gets smaller, the value of the discrepancy function \( \beta_u \) becomes smaller (i.e., the reachable set is arbitrary close to the simulation), which guarantees that the algorithm always terminates.

Theorem 8. (Soundness & completeness). Given an unsafe set \( \text{unsafe} \), time bound \( T \) and fixed input \( u(t) \) for system (1), if Algorithm 1 using the discrepancy of Theo-

Algorithm 1: Verification of systems with fixed input

\[
\begin{align*}
& \text{input: } \Theta, u(t), T, \text{unsafe}, \epsilon_o, \tau_o \\
& \text{1 } C \leftarrow \text{Cover}(\Theta, \epsilon_o, \tau_o); \\
& \text{2 } \text{while } C \neq \emptyset \text{ do} \\
& \quad \text{for } (x, \delta, \epsilon) \in C \text{ do} \\
& \quad \quad \psi = \{(R_i, t_i)\} \leftarrow \text{Simulate}(x, u, T, \epsilon, \tau); \\
& \quad \quad R \leftarrow \text{Bloat}(\psi, \delta, \epsilon); \\
& \quad \quad \text{if } R \cap \text{unsafe} = \emptyset \text{ then} \\
& \quad \quad \quad C \leftarrow C \cup \{(x, \delta, \epsilon)\}; \\
& \quad \quad \text{STB} \leftarrow \text{STB} \cup R; \\
& \quad \quad \text{else if } j, R_j \subseteq \text{unsafe} \text{ then} \\
& \quad \quad \quad \text{return } \text{(UNSAFE, } \psi); \\
& \quad \quad \text{else} \\
& \quad \quad \quad C \leftarrow C \cup \text{Cover}(B_\delta(x), \frac{\delta}{2}, \frac{\epsilon}{2}); \\
& \quad \quad \text{return } \text{(SAFE, } \text{STB}); \\
& \end{align*}
\]
rem 6 returns SAFE or UNSAFE, then (1) is safe or unsafe, respectively. It terminates if (1) is robustly safe or unsafe.

When extending this verification algorithm to work for open hybrid models, the main complication is that spurious transitions may arise from the over-approximations in the computed reach sets. Thus, we have to keep track of possibly spurious mode changes from genuine ones. This is what is implemented in the new version of C2E2 used in Section 6 for verifying hybrid circuit models.

5. Modeling of CMOS Circuits

To investigate the feasibility of our approach, we analyzed models of complementary metal-oxide-semiconductor (CMOS) circuits, the most common technology nowadays. Its basic components are two types of transistors (nMOS and pMOS), which can be used to build any desired logic. Essentially, both deliver current based on the voltages applied to their gate (G), drain (D) and source (S) contact. Modern digital simulation tools like ModelSim or NC-Verilog consider transistors as simple switches, however. Such tools allow fast functional and timing analysis of complex circuits, but lack sufficient accuracy for critical parts of a circuit design. The latter is provided by analog simulations, using state-of-the-art tools like Spice. They are capable of handling very detailed transistor models, consisting of tens to hundreds of equations and configured by hundreds of manufacturer-provided parameters. However, analog simulations quickly reach their limits in terms of simulation complexity for circuits consisting of more than a few tens of transistors and/or signal traces beyond milliseconds in real-time.

In order to decrease simulation times, simplified models have been developed (e.g., Arora, 1993). They are smaller than Spice models (at most six equations are required), and thus amenable to general simulation tools like MATLAB. Despite the reduced complexity, these models can still capture subtle phenomena like channel length modulation and carrier velocity saturation.

Actually, NMOS and PMOS transistors differ substantially in physical and, hence, electrical properties. In our model (Maier, 2017), every transistor operate in one of three different operation regions: the sub-threshold (ST) region, where very little current is delivered, the ohmic region (OHM), where the current scales linearly, and the saturation region (SAT), where the current only changes moderately. The actual behavior within each region is described by a set of differential equations, which involve several fitting parameters. These differ for NMOS and PMOS transistors and are either inferred directly from Spice model variables or fitted to Spice simulations.

5.1 Hybrid inverter model

The simplest CMOS gate is an inverter (see Figure 3), which consists of two transistors stacked above each other. Its output voltage $V_{out}$ is the inverse of the input voltage $V_{in}$, ideally $V_{out} = V_{DD} - V_{in}$ with $V_{DD}$ denoting the supply voltage. In reality, $V_{out}$ is determined by

$$V_{out} = \frac{1}{C_L} I_{out}$$

where $C_L$ is the external load capacitance seen by the output. The output current $I_{out}$ is the difference between the current delivered by $\oplus$ and the current consumed by $\ominus$, which both depend upon $V_{in}$ and $V_{out}$. Since each of the two transistors can operate in three different regions, our basic hybrid inverter model has nine modes. As two of those modes are unreachable in reality, the hybrid model (called InvHy shown in Figure 5 in the sequel) has only 7 modes.

![Fig. 5. Hybrid model automata of a CMOS inverter. The nodes represent the different modes, each involving specific transition guards and invariants. The label shows the operation regions of the transistors, the arcs indicate the possible transitions.](image)

5.2 Uniform model

Given that the number of modes increases exponentially with the number of transistors in a circuit, it is natural to consider ways of avoiding multiple modes already in the transistor models: If the behavior of a transistor could be described by a single, possibly more complex equation that is valid for all operation regions, the need for a hybrid model vanishes altogether.

Our uniform model InvUni (Maier, 2017) accomplishes this by smoothening the boundaries between different regions, using suitably chosen continuous functions. This results in a single non-linear equation (involving exponentials and logarithms), which describes the current through the transistor over the whole operation range. In conjunction with equation (7), this finally leads to a non-linear ODE that describes the behavior of $V_{out}$ depending on $V_{in}$. Empirical validations using Spice simulations revealed a surprisingly good modeling accuracy.
Apart from dramatically reduced model complexity, a key feature of our uniform model is the straightforward development of models for multi-transistor circuits like the NOR gate shown in Figure 4. In a hybrid model, this gate would blow up to a system of $3^4 = 81$ states; here, we end up with a system of two non-linear ODEs only:

$$\dot{V}_{in} = \frac{1}{C_m}(I_1 - I_2); \dot{V}_{out} = \frac{1}{C_L}(I_2 - I_3 - I_4)$$

Herein, $I_X$ represents the current through transistor $X$. The change of $V_{in}$ is proportional to the current charging $C_m$ (cp. eq. (7)) and is just the difference between the currents flowing through the transistors $\bar{3}$ and $\bar{0}$. Note that $C_m$ represents the capacitances of the transistor contacts only, and is hence several orders of magnitude smaller than $C_L$. The derivative of $V_{out}$ is finally determined by the current passing through $\bar{3}$ minus the ones consumed by the transistors $\bar{0}$ and $\bar{2}$.

6. EXPERIMENTS AND RESULTS

We have implemented the discrepancy computation and verification algorithm for open models in the new version of C2E2 and used it to verify several challenging CMOS circuits (see footnote 1). Due to lack of space, we restrict our discussion here to a few examples that demonstrate the principal feasibility, as well as particular strengths, of our approach. Experimenting with larger and more complex circuits, which is mandatory for validating scalability, for example, will be part of our future work.

6.1 Input, simulation and verification

As external input signals, we use both ramp (Ramp) and sigmoidal signals (Sig), which are generated using two separate hybrid automata; a 4-state one for Ramp and 2-state one for Sig. We successfully verified several properties of InvHy, InvUni, NOR-gate, and OR-gate models using the tool. For all the models, we set the unsafe set to be $V_{out} > 1.32 \text{V}$ and the time horizon to be 6.4 s. The first one uses the hybrid model presented in Section 5.1, so we end up with $7 \times 4 = 28$ modes in the Ramp case and $7 \times 2 = 14$ in the Sig case. All other circuits are based on the uniform model. The OR gate is easily derived from the NOR shown in Figure 4 by appending an inverter. All circuit models based on the uniform model have very complex descriptions, i.e., hundreds of logarithmic and exponential terms in their ODEs. Figure 6 shows some simulation results for the NOR-gate.

![NOR gate output voltage over-approximation set for $V_{in} = \text{Ramp}$ (left) and $V_{in} = \text{Sig}$ (right). Different colors indicate different modes in the model.](image)

In addition, we also investigated a two-inverter loop, where the input of one inverter is connected to the output of the other one, implementing a simple state-holding device. In contrast to the other circuits used in our experiments, however, it does not have an external input. Consequently, we just set the output voltages to some initial values and let the circuit run.

Generally, all the simulations behave as expected and show smooth output transitions even when activated by a ramp at its input. Verification shows that, despite initial state uncertainty, the unsafe set $V_{out} > 1.32 \text{V}$ is not reached and simultaneously provides the complete reachtube within the safe set. For example, Fig. 6 shows that the reachtube converges quickly to a deterministic signal trace. Total verification time, split between simulation (Sim.) and discrepancy computation (Discr.), is shown in Table 1.

| Model  | Verification parameters | Timing split [s] | 1/O | time [s] |
|--------|-------------------------|------------------|-----|---------|
| InvHy  | 25.6k $V_{out} \in [1.15, 1.2]$ | 5 5 2 14 |    |         |
| InvUni | 12.8k $V_{out} \in [1.15, 1.2]$ | 7 12 2 21 |    |         |
| NOR    | 320k $V_{out} \in [1.15, 1.2]$ | 223 708 108 1039 |    |         |
| OR     | 320k $V_{out} \in [1.15, 1.2]$ | 766 1406 122 2294 |    |         |
| InvHy  | 25.6k $V_{out} \in [1.15, 1.2]$ | 7 1 1 9 |    |         |
| InvUni | 12.8k $V_{out} \in [1.15, 1.2]$ | 3 13 2 18 |    |         |
| NOR    | 320k $V_{out} \in [1.15, 1.2]$ | 112 822 66 1000 |    |         |
| OR     | 320k $V_{out} \in [1.15, 1.2]$ | 384 1617 74 2075 |    |         |
| InvLoop| 64k $V_{out} \in [1.15, 1.2]$ | 18 119 2 139 |    |         |

6.2 Metastability analysis

Any bistable digital circuit can be driven into a metastable state (Marino, 1981) in which it may output voltage values in the forbidden region between 0 and 1 or experience very high-frequency oscillations for an arbitrary time, before it resolves to a proper digital state again. Verifying whether a circuit may experience metastable behavior is challenging because it arises in highly nonlinear and sensitive parts of its state space.

In order to demonstrate the capability of C2E2 to predict metastable behavior correctly, we use an OR-gate with its output fed back to one of its inputs. This circuit implements a storage loop, which is capable of memorizing a rising transition on its second input. It has been shown in (Függer et al., 2015) that it can be driven into a metastable state, namely, by an input pulse that is shorter than the delay of the feedback loop.

Figure 7 shows input (top) and simulation traces (middle) of this circuit computed by C2E2 for different initial values of $V_{out}$. The reachtube (bottom), corresponding to the output trace sticking longest to a value around 0.6 V, shows a blow up to several thousand Volts, which is physically impossible but indicates the very high sensitivity of the underlying system of ODEs in the metastable region. Even the slightest disturbances of the initial state results in very different trajectories, in particular, in very different metastability resolution times, after which $V_{out}$ resolves to 0 or 1. Albeit this is in accordance with what is
known about metastability, to the best of our knowledge, this is the first reachability analysis of circuits demonstrating metastable behavior.

6.3 Comparison with existing verification tools

We provide a comparison with several state-of-the-art nonlinear\textsuperscript{2} hybrid system verification tools, namely, Flow* (Chen et al., 2013), dReach (Kong et al., 2015) and CORA (Althoff and Grebienyu, 2016). A comparison of C2E2 with these tools on systems without inputs was previously reported in Fan and Mitra (2015); Duggirala et al. (2013).

Flow* is a reachability-based verification tool for hybrid systems. It over-estimates the reachable states for nonlinear systems directly on the vector field and computes Taylor model flowpipes for the ODEs. While C2E2 performs safety verification for a time horizon of 20s with a runtime of 0.1 s for the cardiac oscillator in Example 1, Flow* finishes the flowpipe computation (compare Figure 8 (left)) with a runtime of 5.96 s.

We managed to transfer four models from C2E2 to Flow*, however, none could be verified, for example, on the same set of parameters. InvUni with ramp input runs for more than 10 minutes for the initial mode with the initial set \(V_{\text{out}} \in [1.18, 1.20]\) and a time step of \(5 \times 10^{-5}\) s and quits with the error message “divided by 0” at a simulated time of 0.44 s. Any changes on these initial parameters (either larger initial set range or time step) resulted in either a “divided by 0” or “segment fault” error. Similar problems occurred with InvUni and sigmoidal input. With InvHy, the flowpipe computation keeps oscillating between the modes B and D and the process was manually killed after 60 minutes. For the remaining models, we were not able to avoid the “divided by 0” and the “segment fault” errors by any choice of parameters. We suspect that some of these problems are arising from the large size and complexity of these models.

CORA is a MATLAB toolbox for reachability analysis of nonlinear dynamical systems and polynomial differential inclusions (Althoff, 2013). We modeled the cardiac oscillator from Example 1 in CORA as a hybrid system with two modes, with nonlinear dynamics in each, representing the system receiving an upward and a downward input stimulus. Since the initial sets are represented as boxes or zonotopes, we needed to also specify a small interval (we chose the smallest possible) for the initial conditions of the stimulus. As Figure 8 illustrates, the reach set computed by CORA for this example is less precise in the first mode compared to C2E2 (Figure 2). The implementation of our models with CORA was a much harder task than initially anticipated. Only with tremendous support from the author, who had to adapt the original CORA version, we were able to run at least the cardiac oscillator and InvUni. However, the results for these simple examples already revealed a quite poor performance compared to C2E2. For the InvUni, performing the reachability analysis for a time-horizon of 0.1 s and a time step of 0.01 s took 222 seconds, which caused us to refrain from further experiments with CORA.

dReach is an SMT-based \(\delta\)-reachability analysis tool for hybrid systems. In contrast to C2E2, dReach picks a single start state in the initial set and attempts to find a counter-example that reaches the unsafe set. If it succeeds, a (spurious or real) counter-example has been found and the tool finishes. Otherwise, it picks a suitable starting state for the next trace to be processed. Before it can assure that the goal condition is unsatisfiable, i.e., the system is safe, the entire initial range has to be covered. In sharp contrast to C2E2, however, dReach does not display any information of the reachtubes in that case. To achieve a similar output as provided by C2E2, one would have to (1) check that no unsafe state is reachable, (2) determine the reachtube for each start state to the complement of the goal condition (i.e., the safe set), and (3) plot their union.

Since the basic operations of dReach and C2E2 are fundamentally different, a fair comparison is hard. We managed to build models for the cardiac oscillator in Example 1 (see Figure 8 (right)), InvUni and InvHy, whereas the latter had to be split in up and down transition to work properly. The simulation times, despite the fact that only a single trace from initial is computed, is up to \(4 \times\) slower than C2E2.

On the other hand, operationally, dReach is well-suited for metastability analysis, as bad traces, i.e., ones that end up in the metastable region, are of interest here. With dReach, this could be achieved by defining the metastable region as the goal condition. Unfortunately, however, it was not possible to implement any of the models—NOR, OR or InvLoop—which are the required for metastability analysis: The tool issued a “log(): domain error” after short time, which indicates, according to the developers, that the differential equations are too complex for effectively controlling the errors.

7. CONCLUSIONS AND FUTURE WORK

In this paper we introduced a novel approach suitable for verifying highly sensitive non-linear ODEs with arbitrary external inputs. Its modeling power was shown by successfully implementing several CMOS circuit models like inverter, NOR gate and memory elements, based on both a hybrid and a uniform non-linear transistor model with highly sensitive ODEs and hundreds of nonlinear terms. Moreover, we also succeeded to verify the metastable

---

\textsuperscript{2} A comparison with tools like SpaceEx and Coho, which support only linear systems, is omitted.
behavior of a memory element, which demonstrates the ability of our approach to handle highly sensitive ODEs.

The results of this paper suggest several interesting directions for future research. First, for addressing the relatively large simulation time for complex models, it would be worthwhile to investigate state-of-the-art robust ODE-solvers for stiff ODEs. Another direction would be to generalize the core verification algorithm in order to handle infinite sets of input signals. Finally, we envision promising applications in the area of advanced digital circuit analysis, where C2E2 could be used for verifying metastable behavior of circuits like Schmitt-Triggers (Steininger et al., 2016).

ACKNOWLEDGMENTS

We would like to thank the developers of CORA (especially Matthias Althoff), dReach and Flow* for their support while porting our models.

REFERENCES

Althoff, M. (2013). Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets. In Proc. of HSCC ’13: the 16th International Conference on Hybrid Systems: Computation and Control, 173–182. ACM. doi:10.1145/2461328.2461358.

Althoff, M. and Grebenyuk, D. (2016). Implementation of interval arithmetic in CORA 2016. In ARCH Workshop, 91–105.

Arona, N. (1993). MOSFET models for VLSI circuit simulation: theory and practice. Computational microelectronics. Springer.

Chen, X., Abrahám, E., and Sankaranarayanan, S. (2013). Flow*: An analyzer for non-linear hybrid systems. In CAV, 258–263.

Dang, T., Donzé, A., and Maler, O. (2004). Verification of analog and mixed-signal circuits using hybrid system techniques. In FMCAD, 21–36. doi:10.1007/978-3-540-30494-4_3.

Dang, T., Le Guernic, C., and Maler, O. (2009). Computing reachable states for nonlinear biological models. In CMSB, volume 5688 of LNCS, 126–141. Springer. doi: 10.1007/978-3-642-03845-7_9.

Donzé, A. (2010). Breach, a toolbox for verification and parameter synthesis of hybrid systems. In CAV, 167–170.

Duggirala, P.S., Mitra, S., and Viswanathan, M. (2013). Verification of annotated models from executions. In EMSOFT, 1–10.

Fan, C. and Mitra, S. (2015). Bounded verification with on-the-fly discrepancy computation. In ATVA, 446–463.

Fan, C., Qi, B., Mitra, S., Viswanathan, M., and Duggirala, P.S. (2016). Automatic reachability analysis for nonlinear hybrid models with C2E2. In CAV, 531–538. Springer.

Franzle, M., Herde, C., Teige, T., Ratschan, S., and Schubert, T. (2007). Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure. JSAT, 1(3-4), 209–236.

Frehse, G. (2008). Phaver: algorithmic verification of hybrid systems past hytech. International Journal on Software Tools for Technology Transfer, 10(3), 263–279. doi: 10.1007/s10009-007-0062-x.

Frehse, G., Le Guernic, C., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., and Maler, O. (2011). SpaceX: Scalable verification of hybrid systems. In CAV, 379–395.

Függer, M., Najvirth, R., Nowak, T., and Schmid, U. (2015). Towards binary circuit models that faithfully capture physical solvability. In DATE, 1455–1460.

Gupta, S., Krogh, B.H., and Rutenbar, R.A. (2004). Towards formal verification of analog designs. In ICCAD, 210–217. doi:10.1109/ICCAD.2004.1382573. URL http://dx.doi.org/10.1109/ICCAD.2004.1382573.

Henzinger, T.A., Ho, P.H., and Wong-Toi, H. (1997). Hytech: A model checker for hybrid systems. In CAV, 460–463. Springer.

Kong, S., Gao, S., Chen, W., and Clarke, E. (2015). dReach: δ-reachability analysis for hybrid systems. In TACAS, 200–205.

Lata, K. and Jamadagni, H.S. (2010). Formal verification of tunnel diode oscillator with temperature variations. In ASPDAC, 217–222. IEEE Press.

Maier, J. (2017). Modeling the CMOS inverter using hybrid systems. Technical Report TUW-259633, E182-TI; TU Wien. URL http://publik.tuwien.ac.at/files/publik_259633.pdf.

Marino, L.R. (1981). General theory of metastable operation. IEEE ToC, 30(2), 107–115.

Steininger, A., Maier, J., and Najvirth, R. (2016). The metastable behavior of a Schmitt-Trigger. In ASYNC, 57–64. doi:10.1109/ASYNC.2016.19.

Yan, C. and Greenstreet, M.R. (2008). Verifying an arbiter circuit. In FMCAD, 7:1–7:9. IEEE Press, Piscataway, NJ, USA. URL http://dl.acm.org/citation.cfm? id=1517424.1517431.