Efficient Equivalence Checking of Nonlinear Analog Circuits using Gradient Ascent

Kemal Çağlar Coşkun
Institute of Computer Science, University of Bremen
Bremen, Germany
kcoskun@uni-bremen.de

Muhammad Hassan
Institute of Computer Science, University of Bremen
Cyber-Physical Systems, DFKI
Bremen, Germany
hassan@uni-bremen.de

Lars Hedrich
Institute of Computer Science, University of Frankfurt/Main
Hessen, Germany
hedrich@em.cs.uni-frankfurt.de

Rolf Drechsler
Institute of Computer Science, University of Bremen
Cyber-Physical Systems, DFKI
Bremen, Germany
drechsler@uni-bremen.de

ABSTRACT

In this paper, we present an optimized methodology for performing state-space-based equivalence checking of nonlinear analog circuits by using a gradient-ascent-based search algorithm to efficiently traverse a common state space. Essentially, the method searches for critical regions where the functional behaviors of two circuit designs show the greatest divergence. The key challenges in this approach are the mapping of both designs onto a common canonical state space, the computation of the gradient, and the exclusion of unreachable regions within the state space. To address the first challenge, we use locally linearized systems and leverage the *Kronecker Canonical Form* (KCF). To facilitate the computation of the gradient, we employ a purpose-built target function, and to exclude unreachable regions, we utilize vector projection techniques. Through experiments with nonlinear analog circuits and a scalability analysis, we demonstrate the successful and efficient computation performed with the proposed methodology, achieving speedups of up to 468 times.

CCS CONCEPTS

• **Hardware** → **Analog and mixed-signal circuits**; **Equivalence checking**; **Functional verification**.

KEYWORDS

equivalence checking, analog circuits, formal verification, state-space methods, gradient methods, optimization, circuit analysis

ACM Reference Format:
Kemal Çağlar Coşkun, Muhammad Hassan, Lars Hedrich, and Rolf Drechsler. 2024. Efficient Equivalence Checking of Nonlinear Analog Circuits using Gradient Ascent. In *Proceedings of Design Automation Conference (DAC'24).* ACM, New York, NY, USA, 6 pages.

1 INTRODUCTION

Analog design verification has been hindered by the increased complexity of analog circuits and the growing system integration of analog and digital circuits. SPICE-level simulations [16], coupled with manual inspection of simulation results, are considered the standard approach and cannot be disregarded. However, the extensive time required for SPICE-level simulations presents a fundamental barrier to automated analog verification [3]. To address this issue and achieve faster simulation speeds and early design verification of the *Design Under Verification* (DUV), various levels of circuit design abstraction can be employed. These alternative representations range in complexity from behavioral models to streamlined netlists to pre-extraction netlists that are free from parasitic elements.

As a result, the attractiveness of using top-down design principles is ever-increasing for analog systems. Particularly, the *Timed Data Flow* (TDF) *Model of Computation* (MoC) provided in SystemC AMS [4] and behavioral models in Verilog-A [2] can accelerate simulation speed by up to 100, 000 times [3] and 100 times [14], respectively, and allow for early design verification. However, the scarcity of suitable equivalence checking methodologies between more abstract (e.g., Verilog-A) and less abstract (e.g., SPICE-level) models remains a significant obstacle to the adoption of top-down design principles.

Equivalence checking techniques determine whether two design implementations are functionally equivalent. The implementations may operate at several abstraction levels and be designed in different description environments, such as transistor netlists and system-level languages. While equivalence checking methods are widely used in the digital domain [8, 15, 9], analogous techniques that are formal or at least formalized remain scarce in analog circuit design practices [21, 20, 11, 12, 10, 19, 18, 1, 17]. When examining formal and formalized methodologies for equivalence checking, the considered approaches typically include state-space coverage, model checking, and reachability based methods. Although some of these equivalence checking techniques successfully and reliably handle linear models [5, 6], nonlinear models pose a significant challenge. As a result, there is still a lack of confidence regarding the application of high-level models for nonlinear analog circuits.

Contribution: In this paper, we propose an efficient equivalence checking methodology for nonlinear analog circuits. To this end, we use a gradient-ascent-based search algorithm to efficiently traverse a common canonical state space. Essentially, the method searches for critical regions where the functional behaviors of the two designs show the greatest divergence. The key challenges in this approach are the mapping of both designs onto a common canonical state space, the computation of the gradient, and the exclusion of unreachable regions within the state space. To overcome

DAC'24, June 23–27, San Francisco, CA, USA

Table 1: Analog Equivalence Checking Approaches

Approach	Source	Verification Coverage	Applicable Circuits
Simulation-based	[18, 1, 17]	For finite number of input signals	Dynamic nonlinear
Structural analysis	[5, 6]	Complete coverage	Dynamic Linear
	[7]	Complete approximative coverage	Static nonlinear
State-space-based	[11, 12, 10, 19]	At finite number of points in the state-space	Dynamic nonlinear
	Proposed Work	Optimized coverage via gradient ascent	

these challenges, we make the following contributions:

- To create a common canonical state space, we leverage the *Kronecker Canonical Form* (KCF) that is also used in the tool Vera [11, 10].
- We introduce a purpose-built target function that enables the search for regions with the greatest divergence.
- We calculate the gradient of this target function in relation to the common canonical state space variables.
- We utilize vector projection techniques to exclude unreachable regions from the search space and eliminate false negatives.

We combine these methods in a novel algorithm to solve the aforementioned challenges in a unified approach.

Afterwards, we demonstrate the applicability and the runtime efficiency of our methodology, which achieves speedups of up to 468 times, through three case studies: the *Band-Pass Filter* (BPF) and the *Operational Transconductance Amplifier* (OTA) circuit, illustrating its applicability on weakly and strongly nonlinear circuits, respectively; and the freely scalable *active low-pass filter*, demonstrating its scalability. The models in these case studies include Verilog-A models as well as SPICE-level circuits featuring full BSIM transistor models.

2 RELATED WORK

In a survey on equivalence checking [21], the research up to 2007 was evaluated, and it was noted that every method uses a priori knowledge of the DUV during the development stage. [20] also provided a comparison of several equivalence checking techniques, besides proposing a novel method based on reachability. The authors pointed to the difficulty in defining the coverage metrics and noted that many methods attempt to strike a compromise between completeness and pessimism.

Each subsequent paragraph of this section presents a summary of a unique approach to analog equivalence checking. A concise overview is seen in Table 1, which highlights the key aspects of each approach and how the proposed work differs.

A simulation-based equivalency checking approach was studied in [18], where mapping methods for comparing signals in different domains were created. Also, by developing methods to decrease the input space, the high computational cost of simulation-based approaches was lessened. However, the utilized conventional system-level simulation stimuli do not entirely encompass all behavior. The authors emphasize this fact by referring to their method as semi-formal. Additionally, [1] established a systematic technique with an emphasis on circuit features while working on simulation-based equivalence. The potential incompleteness of the externally provided testbench, however, was not addressed. In [17] an optimization-based approach with automatic input generation was used to address the coverage issue of simulation-based verification. It is debatable however whether the specified set of input parameters can represent all necessary input shapes.

Another group of equivalence checking approaches, which employ graphs and mathematical equations, were developed in [5–7]. These approaches aimed to establish mappings between circuit models of various abstractions by utilizing graph and equation simplification methods. While the methods presented in [5, 6] achieve full coverage, they are limited to linear circuits. This limitation was partly alleviated in [7], where support for static nonlinear circuits was added, at the expense of tolerating approximations. Importantly, due to the reliance on equation simplification techniques, these methods are unable to fully support complex SPICE models.

The state-space-based equivalence checking approach in Vera [11, 12, 10, 19], compares the vector fields of two models on a point grid. Over time, this approach has been extended to support models with differential-algebraic equations [12] and multi-input multi-output circuits [19]. Also in [19], the issue of false negatives that arise due to the exploration of unreachable regions was addressed with a reachability method for the point grid.

The proposed methodology is also a state-space approach and addresses the efficiency and scalability issues of approaches that rely on point grids, such as [19]. It builds upon some of the underlying methods of Vera, particularly the canonical state space. To increase efficiency, it uses gradient ascent to swiftly traverse the canonical state space and precisely find critical regions, as seen in Table 1. It also develops a vector projection based methodology to exclude unreachable regions from the search space, and combines these techniques in a novel algorithm.

3 PRELIMINARIES

In this section, we provide a brief overview of the methods employed for mapping both circuit models to a common canonical state space. As a first step, two nonlinear, implicit *Differential-Algebraic System of Equations* (DAE) that separately describe the behavior of both circuits, are obtained. Next, these equations are linearized around a given operating point to obtain a linear DAE. Afterward, order reduction techniques are employed with the KCF to transform the equations into a linear, explicit system of *ordinary differential equations* (ODE) in a canonical form.

These steps facilitate the establishment of a mapping between both circuits for a given operating point. To approximately preserve this mapping throughout the canonical state space, both the circuit specific state space and the common canonical state space are traversed synchronously.

3.1 Linearization

The mentioned nonlinear, implicit DAE given as

$$\vec{f}(\dot{\vec{x}}(t), \vec{x}(t), \vec{u}(t)) = 0$$

which in our case is also time-independent, can be obtained with the *Modified Nodal Analysis* (MNA) [13] method.

The linearization of this DAE around an operating point *op*, with the condition $\text{op}_\text{c} = (\vec{x} = \vec{x}_\text{op}, \vec{u} = \vec{u}_\text{op})$ leads to the linear DAE given as

$$\left.\frac{\partial \vec{f}}{\partial \dot{\vec{x}}}\right|_{\text{op}_\text{c}} (\dot{\vec{x}}(t) - \dot{\vec{x}}_\text{op}) + \left.\frac{\partial \vec{f}}{\partial \vec{x}}\right|_{\text{op}_\text{c}} (\vec{x}(t) - \vec{x}_\text{op}) + \left.\frac{\partial \vec{f}}{\partial \vec{u}}\right|_{\text{op}_\text{c}} (\vec{u}(t) - \vec{u}_\text{op}) = 0$$

which we rewrite for better readability by hiding the dependency on t, using $J_\bullet$ to denote the Jacobian matrices, and representing local perturbations with $\vec{\delta}_\bullet$, to get:

$$J_{\dot{x}} \cdot \dot{\vec{\delta}}_x + J_x \cdot \vec{\delta}_x + J_u \cdot \vec{\delta}_u = J_{\dot{x}} \cdot \dot{\vec{x}}_\text{op} \tag{1}$$

3.2 Calculating the Canonical Form

Next, this linear DAE is transformed to the KCF by changing the base, and to a final form with order reduction. These steps are simultaneously done with two transformation matrices, E_r and F_r, that satisfy the following properties

$$\begin{aligned} E_r \cdot J_{\dot{x}} \cdot F_r &= I \\ E_r \cdot J_x \cdot F_r &= -\Lambda \end{aligned} \tag{2}$$

where the subscript $_r$ denotes 'reduced' and Λ is a sorted diagonal matrix with the eigenvalues on the main diagonal. For brevity, we refrain from describing the sorting rule used for Λ, and the derivation of E_r and F_r; instead, we recommend [10] for further information.

The canonical form is obtained by substituting $\vec{\delta}_x = F_r \cdot \vec{z}$ and $\dot{\vec{x}} = F_r \cdot \dot{\vec{z}}$, and multiplying with E_r from the left,

$$E_r \cdot J_{\dot{x}} \cdot F_r \cdot \dot{\vec{z}} + E_r \cdot J_x \cdot F_r \cdot \vec{z} + E_r \cdot J_u \cdot \vec{\delta}_u = E_r \cdot J_{\dot{x}} \cdot \dot{\vec{x}}_\text{op}$$

which simplifies with Eq. (2) to

$$\dot{\vec{z}} = \Lambda \cdot \vec{z} - E_r \cdot J_u \cdot \vec{\delta}_u + E_r \cdot J_{\dot{x}} \cdot \dot{\vec{x}}_\text{op} \tag{3}$$

This form is used in Sec. 4.2 and 4.3 to calculate the error, the gradient, and the locally movable directions. Additionally, F_r is used to traverse the common canonical state space synchronously with the individual state spaces of the circuits. This is accomplished by transforming the steps $\Delta\vec{z}$ in the canonical state space to steps in the individual state spaces with

$$\Delta\vec{x} = F_r \cdot \Delta\vec{z} \tag{4}$$

In the next section, we present our methodology based on gradient ascent.

4 GRADIENT-ASCENT-BASED EQUIVALENCE CHECKING METHODOLOGY

In this section, we present our novel gradient-ascent-based equivalence checking methodology for analog circuits.

First, we provide an overview of our proposed methodology along with the algorithm that drives it. Then, we describe our proposed target function and the calculated gradient. Afterwards, we explain the use of vector projection to avoid traversing unreachable regions in the state space.

1: **for** starting point $\vec{u}_s \in U_s$ **do**
2: $\quad\vec{z}_\text{op} \leftarrow \vec{0}$
3: $\quad\vec{x}_\text{op}^{(C)} \leftarrow$ DC_Analysis($\vec{u}_s$, circuit$^{(C)}$)
4: $\quad$**repeat**
5: $\quad\quad S_\text{lin}^{(C)} \leftarrow$ Linearize($\vec{x}_\text{op}^{(C)}$, circuit$^{(C)}$) ▷ Sec. 3.1
6: $\quad\quad S_\text{can}^{(C)} \leftarrow$ CalcCanonical($S_\text{lin}^{(C)}$) ▷ Sec. 3.2
7: $\quad\quad \dot{\vec{z}}_\text{op}^{(C)} \leftarrow$ Eq. (3)($S_\text{can}^{(C)}, \vec{z}_\text{op}, \vec{x}_\text{op}^{(C)}, \vec{\delta}_u = \vec{0}$)
8: $\quad\quad \vec{v}_g \leftarrow$ Eq. (6)($S_\text{can}^{(A)}, S_\text{can}^{(B)}, \dot{\vec{z}}_\text{op}^{(A)}, \dot{\vec{z}}_\text{op}^{(B)}$)
9: $\quad\quad \vec{v}_s \leftarrow$ Eq. (7)($\vec{v}_g, r_l, \dot{\vec{z}}_\text{op}^{(\cdot)}, \phi_{\delta,\max}$)
10: $\quad\quad \epsilon_{\dot{\vec{z}},\text{op}} \leftarrow \|\dot{\vec{z}}_\text{op}^{(A)} - \dot{\vec{z}}_\text{op}^{(B)}\|_2$ ▷ Eq. (5)
11: $\quad\quad \vec{z}_\text{op} \leftarrow \vec{z}_\text{op} + \vec{v}_s$ ▷ Eq. (8)
12: $\quad\quad \vec{x}_\text{op}^{(C)} \leftarrow \vec{x}_\text{op}^{(C)} +$ Eq. (4)($S_\text{can}^{(C)}, \Delta\vec{z} = \vec{v}_s$)
13: $\quad$**until** $\vec{v}_s = \vec{0}$ **or** max iterations reached
14: **end for**

- (C) stands for operations that are done for both A and B circuits.
- $S_\text{lin}^{(C)}$ and $S_\text{can}^{(C)}$ hold matrices from Eq. (1) and Eq. (3), respectively.

Algorithm 1: Algorithm overview for the proposed equivalence checking methodology

4.1 Overview and Algorithm

The overview of the algorithm driving our equivalence checking methodology is given in Algorithm 1. The core of the approach lies in the *repeat-until* loop, which implements the gradient-ascent-based traversal. The loop starts with a new point in the state space, $\vec{x}_\text{op}^{(C)}$ and obtains the canonical form (lines 5, 6) as described in Sec. 3. In line 7, $\dot{\vec{z}}_\text{op}^{(A)}$ and $\dot{\vec{z}}_\text{op}^{(B)}$, which are needed for the error and gradient calculations, are calculated. The gradient and the next step are calculated in lines 8 and 9 according to Sec. 4.2 and Sec. 4.3, respectively. In line 10 the pointwise equivalence error $\epsilon_{\dot{\vec{z}},\text{op}}$ is calculated and saved to a local file. Finally, the step in the common and individual state spaces are taken in lines 11 and 12, respectively.

To find the global maximum, we use an outer *for* loop that initiates the traversal from multiple inputs, overcoming potential local maxima. The set of inputs, U_s, is given to the algorithm as an input parameter.

Note that the approach does not involve transient simulations.

4.2 Target Function and Gradient

For equivalence checking, our methodology searches for the region in the common canonical state space, where the functional difference between both circuits is most significant. This difference is defined by the derivatives of the canonical state space variables and is given as

$$\epsilon_{\dot{\vec{z}},\text{op}} = \|\dot{\vec{z}}_\text{op}^{(A)} - \dot{\vec{z}}_\text{op}^{(B)}\|_2 \tag{5}$$

Note that, when doing a search for the maximum value of some variable x, one can set the target function to be $f(x)$, as long as f is monotonically increasing for all values of x. Since $\epsilon_{\dot{\vec{z}},\text{op}}$ is always non-negative, we define the target function as

$$\epsilon_t = \|\dot{\vec{z}}_\text{op}^{(A)} - \dot{\vec{z}}_\text{op}^{(B)}\|_2^2$$

to facilitate the calculation of the gradient, which is,

$$\vec{v}_g = \frac{\partial \epsilon_t}{\partial \vec{z}} = \frac{\partial \| \dot{\vec{z}}_{\text{op}}^{(A)} - \dot{\vec{z}}_{\text{op}}^{(B)} \|_2^2}{\partial \vec{z}}$$

Using the chain rule, this expression simplifies to

$$\vec{v}_g = \left[\frac{\partial (\dot{\vec{z}}_{\text{op}}^{(A)} - \dot{\vec{z}}_{\text{op}}^{(B)})}{\partial \vec{z}} \right]^{\text{T}} \frac{\partial \| \dot{\vec{z}}_{\text{op}}^{(A)} - \dot{\vec{z}}_{\text{op}}^{(B)} \|_2^2}{\partial (\dot{\vec{z}}_{\text{op}}^{(A)} - \dot{\vec{z}}_{\text{op}}^{(B)})}$$

$$= \left[\frac{\partial (\dot{\vec{z}}_{\text{op}}^{(A)} - \dot{\vec{z}}_{\text{op}}^{(B)})}{\partial \vec{z}} \right]^{\text{T}} 2(\dot{\vec{z}}_{\text{op}}^{(A)} - \dot{\vec{z}}_{\text{op}}^{(B)})$$

and using the following, derived relationship from Eq. (3),

$$\frac{\partial (\dot{\vec{z}}_{\text{op}}^{(\cdot)})}{\partial \vec{z}} = \Lambda^{(\cdot)}$$

we can further simplify the expression for $\vec{v}_g$ to,

$$\vec{v}_g = 2 \left[\Lambda^{(A)} - \Lambda^{(B)} \right]^{\text{T}} (\dot{\vec{z}}_{\text{op}}^{(A)} - \dot{\vec{z}}_{\text{op}}^{(B)}) \tag{6}$$

However, traversing the state space solely based on this gradient can lead to the exploration of unreachable regions. This is undesired, since, especially for behavioral models, the behavior only needs to be equal in plausible conditions. To address this issue, we utilize the vector projection method, which is explained in the next section.

4.3 Avoiding Unreachable Regions with Vector Projection

The range of a circuit's variable values is typically confined by the circuit's characteristics, the bounds of the initial states, and the bounds of inputs. Together, these feasible values define the 'reachable' regions in the state space, as opposed to the 'unreachable' regions. The methodology should avoid the unreachable regions, since comparing two models outside their designed operating conditions may produce false negatives.

To avoid unreachable regions, we neither compute the complete unreachable region, nor the complete reachable region, to save computing resources. Instead, we ensure that each step of the traversal is taken towards a reachable direction, thereby never stepping out of the reachable regions.

To compute the reachable directions for a point *op* in the state space, we first use the linear canonical form in Eq. (3) to compute the natural direction of state change, i.e. the unit vector in the direction of $\dot{\vec{z}}_{\text{op}}$, $\vec{u}_{\dot{\vec{z}},\text{op}}$. Then, a reachable direction is defined as a unit vector such that the angle between it and $\vec{u}_{\dot{\vec{z}},\text{op}}$ is at most the relaxation parameter $\phi_{\delta,\max}$. This relaxation is to account for the uncertainties in the models. The set of all reachable directions is then defined as,

$$D_{\text{op}} = \Big\{ \big[1, \quad \phi_{u,1} + \phi_\delta, \quad \cdots, \quad \phi_{u,n-1} + \phi_\delta\big]^{\text{T}} \;\Big|\; -\phi_{\delta,\max} \le \phi_\delta \le \phi_{\delta,\max} \Big\}$$

where $\phi_{u,i}$ is the i^{th} angular coordinate in the hyperspherical coordinate system for the vector $u_{\dot{\vec{z}},\text{op}}$.

The relaxation parameter, set uniquely from the interval $[0°, 90°]$ for each analysis, should reflect the uncertainty of the model that is used to calculate $\vec{u}_{\dot{\vec{z}},\text{op}}$. Just one of the two circuit models, chosen individually per analysis, is used in this procedure. Generally, the model at a lower abstraction level should be preferred, since it will be more realistic and accurate.

To calculate the next step of the traversal, $\vec{v}_s$, we adjust the magnitude of the gradient $\vec{v}_g$ (from Eq. (6)) with the learning rate r_l and project it onto D_{op}, if the next step will fall outside the set of initial states Z_i:

$$r_{v_s} = r_l \cdot r_{v_g}$$

$$\phi_{v_s,i} = \begin{cases} \phi_{v_g,i} & , (r_l \vec{v}_g + \vec{z}) \in Z_i \\ \min(\max(\phi_{v_g,i}, \phi_{\dot{\vec{z}},i} - \phi_{\delta,\max}), \phi_{\dot{\vec{z}},i} + \phi_{\delta,\max}), \text{o.w.} \end{cases} \tag{7}$$

where

$$\vec{u}_{\dot{\vec{z}},\text{op}} = \begin{bmatrix} 1, & \phi_{\dot{\vec{z}},1}, & \ldots, & \phi_{\dot{\vec{z}},n-1} \end{bmatrix}^{\text{T}},$$

$$\vec{v}_s = \begin{bmatrix} r_{v_s}, & \phi_{v_s,1}, & \ldots, & \phi_{v_s,n-1} \end{bmatrix}^{\text{T}} \text{ and}$$

$$\vec{v}_g = \begin{bmatrix} r_{v_g}, & \phi_{v_g,1}, & \ldots, & \phi_{v_g,n-1} \end{bmatrix}^{\text{T}}$$

The vector projection in Eq. (7) is not performed when the next step falls within the set of initial states Z_i, because the circuit can reach any point within Z_i, even if that specific point cannot be reached from the current state, *op*.

The next step of the traversal is then executed as

$$\vec{z}_{\text{next}} = \vec{v}_s + \vec{z} \tag{8}$$

thereby keeping the traversal in the reachable regions.

The methodology can be implemented with the derived equations 5, 6, 7, and 8 to efficiently do equivalence checking between the models. We demonstrate this on some case studies in the next section.

5 EXPERIMENTAL EVALUATION

In this section we conduct three case studies to demonstrate the applicability and efficiency of our methodology: a BPF and a highly nonlinear OTA circuit to analyze its applicability and compare it with past work, and a scalable active low-pass filter to analyze the scalability of our methodology.

Among the approaches given in Table 1, the error and runtime results are compared only against Vera [11, 12, 10, 19], since [5–7] are limited in their applicability and [18, 1, 17] provide neither available, open-source implementations, nor enough details of the models used in the experiments for replication.

5.1 Experimental Setup

For all experiments, we set $\phi_{\delta,\max}$ to 30°, based on the level of exactness of the model parameters. This value should be initially set to a high value and reduced if the location of the maximum error in the state-space seems unreasonable. How this location can be analyzed is shown in Section 5.3. Also, since $\epsilon_{\dot{\vec{z}},\text{op}}$ is an absolute value and difficult to understand, we report a relative error given as

$$\epsilon_{r,\max} = \frac{\max\limits_{\text{op}} \epsilon_{\dot{\vec{z}},\text{op}}}{2 \cdot \max\Big(\max\limits_{\text{op}} \| \dot{\vec{z}}_{\text{op}}^{(A)} \|_2, \max\limits_{\text{op}} \| \dot{\vec{z}}_{\text{op}}^{(B)} \|_2 \Big)} \tag{9}$$

which normalizes $\epsilon_{\dot{\vec{z}},\text{op}}$ such that the relative error, $\epsilon_{r,\max}$, falls in the range 0 to 1, making it more intuitive.

All computations were performed on a four-core virtual environment with 16 GB RAM, on an octa-core AMD Ryzen 7 PRO 4750U machine with 32 GB RAM.

Table 2: Equivalence Results for the BPF and OTA Models

		$\epsilon_{r,max}$		Runtime (s)		
	Avg. Step	Vera	This[a]	Vera	This[a]	Speedup
BPF	0.48	3.191×10^{-4}	2.345×10^{-4}	3.19	0.2	16×
	0.26	2.399×10^{-4}	2.257×10^{-4}	6.99	0.22	32×
	0.10	2.496×10^{-4}	2.501×10^{-4}	22.13	0.47	47×
OTA	0.24	0.3221	0.2578	14.14	2.4	5.9×

[a] **This:** The proposed methodology

5.2 Band-Pass Filter

In our first case study, we check whether the methodology can correctly analyze two weakly nonlinear and very similar BPF models, which are typically useful in fields such as sound production. The models are on the SPICE-level and are designed using a two-stage Sallen-Key topology with a linear gain of 1.957, and lower and higher cutoff frequencies of 981 Hz and 2.58 kHz, respectively. We induce a slight difference between the models by connecting a load resistance of 10 kΩ to one of the circuits and check whether our methodology can catch this difference properly.

As seen from the BP results in Table 2, we ran multiple analyses with increasing precision, i.e. decreasing average step. For all of these analyses, the initial parameters are given as $U_s = \{-1, -0.5, 0, 0.5, 1\}$ and $Z_i = \{(z_1, z_2) | z_1, z_2 \in [-0.5, 0.5]\}$. To facilitate a comparison with the original methodology from Vera, the parameter r_I was adjusted such that the average step size during the traversal was similar to the average step size taken by Vera. Furthermore, the denominators in Eq. (9) were made equal by choosing the larger value from both methodologies.

The small relative error values in the BPF results of Table 2 are as expected, given the very slight difference between the models. Also, the slight difference between the relative error measures obtained from Vera and this methodology is reasonable, since the method used to exclude unreachable regions differ between the methodologies. Therefore, the traversed state space regions are slightly different in both cases.

In the next section, we perform an additional case study that significantly differs from this one.

5.3 Operational Transconductance Amplifier

In our second case study, we apply the methodology to a scenario contrasting with the first case study by analyzing two strongly nonlinear and considerably different models. To be precise, we compare a SPICE model and a Verilog-A behavioral model, both representing an OTA circuit. The SPICE model is a single-ended, two-stage OTA with 10 transistors. The transistors are MOSFET models of level 49, which is an enhanced version of BSIM3v3.

The behavioral model implements several key behavioral properties of the OTA. Firstly, for the main behavior, the voltage-to-current amplification, it uses a hyperbolic tangent function. Secondly, it linearly models the common mode rejection ratio. Thirdly, it considers saturation conditions through the use of if statements. Finally, the model includes two internal load resistances and an internal load capacitance.

The relative error and runtime results for the OTA are also given in Table 2, where we observe that the relative error result is huge,

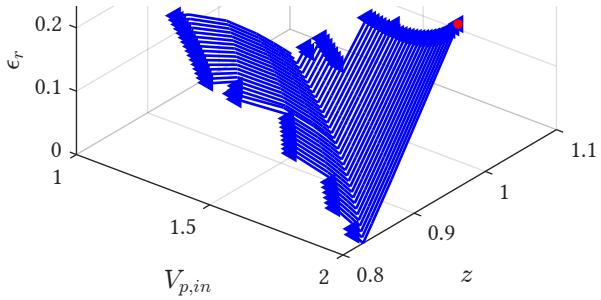

Figure 1: The state-space traversal for the OTA models.

approximately 26%. This was anticipated, due to the stark contrast between the strong nonlinearities present in the MOSFET models of the SPICE model and the relatively weaker nonlinearity of the behavioral model.

In the following, we will show how these results can be used to improve the models by looking at the traces of the state-space traversal and at the variable values when the maximum error happens. For an initial impression, we check the complete state-space traversal given in Figure 1. Since only three variables can be displayed in a 3D plot, we choose the three most relevant variables: the positive input voltage to the OTA, the z variable in the common canonical state-space, and the relative error ϵ_r. The multiple traversals (all ending with an arrowhead) are iterations of the outer for loop of Algorithm 1, used to find the global maximum. The point of maximum error is indicated on the plot with a red dot.

The positive input voltage to the OTA at the point of maximum error is 2 volts, whereas the derivative of the output voltage for the SPICE-level model and the behavioral model are -13.65×10^6 V/s and -39.55×10^6 V/s, respectively.

With a 2 pF load capacitance connected at the output, we can get the output current from the derivative of the output voltage with

$$I_{\text{out}} = C_L \frac{dV_{\text{out}}}{dt}$$

which is -27.302 μA for the SPICE-level model and -79.09 μA for the behavioral model. As discussed previously, the input-voltage-to-output-current amplification is implemented with a hyperbolic tangent function. If it is desired that the behavioral model behaves more similar to the SPICE model at an input of 2 volts, this behavior must be revised.

This analysis demonstrates that the methodology is capable of handling strongly nonlinear models as well as models that significantly differ from one another, and that it can be used to find problematic regions of the models that cause differences.

As a final note, the slight difference to the reported relative error value of Vera is once again expected, due to the differences in avoiding unreachable regions.

5.4 Scalability Analysis

In this section, we conduct a scalability analysis using a freely scalable active low-pass filter. Instead of delving into the intricacies of a detailed low-pass implementation, the focus in this section is on the scalability properties of the proposed methodology. Therefore, the implementation consists of simple RC branches and ideal voltage-dependent current sources, as seen in Figure 2. To introduce

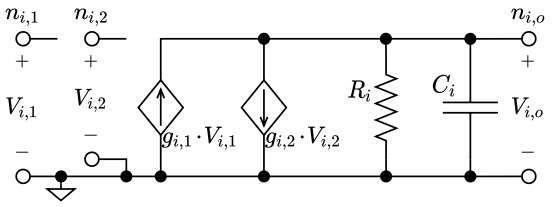

Figure 2: Three-port network used in the scalable low-pass filter.

Table 3: Runtimes (s) & Speedup for Increasing Number of Dimensions

Number of Dimensions (n)		2	4	6	8	10
Runtime	Vera	2.4	2014	-[a]	-[a]	-[a]
(in seconds)	**Proposed Work**	0.6	4.3	51	880	23400
Speedup		4×	468×			

[a] Terminated by the Linux Out-Of-Memory Killer

a nonlinear difference, a current source of one of the two circuits was modeled nonlinearly.

To create a circuit with n dimensions in the state-space, we replicate the three-port network shown in Fig. 2 n times and connect these in series. Specifically, for the i'th network, we connect $V_{i,1}$ to $V_{i-1,o}$ and $V_{i,2}$ to $V_{i+1,o}$, with two exceptions: in the first network ($i = 1$), $V_{1,1}$ is connected to a voltage source, and in the final network ($i = n$), $V_{n,2}$ is short-circuited. The n capacitors in the circuit create n dimensions in the common canonical state space, since we don't use order reduction for these experiments. Additionally, to ensure comparability across experiments, we adjust the parameters so that the average step size for all cases is approximately 0.2.

The results of this analysis are displayed in Table 3, showing the higher efficiency and scalability of the proposed methodology. This is achieved thanks to the gradient-based traversal, which eliminates the exponential complexity of point-grid-based approaches. However, after the number of dimensions reaches approximately 8, the n dimensional matrix operations, rather than the space traversal, emerge as the dominant bottlenecks of the algorithm. Nevertheless, since one of the compared circuits is usually on a higher abstraction level, and thanks to the order reduction done with E_r and F_r, the common canonical state space of the analog circuits is almost always smaller than the practical limit of the methodology.

6 CONCLUSION

In this paper, we developed a novel equivalence-checking methodology with gradient ascent for nonlinear analog circuits. The approach uses a custom gradient-ascent algorithm, designed to address the specific requirements of the equivalence-checking methodology based on the canonical state space. We used the KCF, a novel target function for the gradient, and introduced a novel application of vector projection techniques to avoid false negatives. We validate our methodology with three case studies and demonstrate its runtime efficiency and scalability as seen in Tables 2 and 3.

This paper can be extended in two potential ways. To begin, the current methodology obtains the linearized system from one of the supported external circuit simulators. Any of these simulators that are open source can be modified to provide the second derivatives of the state variables as well. Incorporating this additional information with Eq. (1) and (3), can enable us to obtain the second derivative for the target function. Consequently, a search algorithm such as the Newton–Raphson method, which needs the second derivative but has better convergence properties, may be used.

Secondly, to enhance efficiency even further, we can consider making the learning rate parameter r_l adaptive. By doing so, the methodology can dynamically adjust the learning rate based on the characteristics of the circuit being analyzed, leading to potential improvements in convergence and computational efficiency.

ACKNOWLEDGMENTS

This research was supported by BMBF under the ECXL grant (grant no. 01IW22002).

REFERENCES

[1] Antara Ain, Sayandeep Sanyal, and Pallab Dasgupta. 2017. A Framework for Automated Feature Based Mixed-Signal Equivalence Checking. In *VLSI Design and Test (VDAT)*. Roorkee, 779–791. https://doi.org/10.1007/978-981-10-7470-7_73
[2] Ramana Aisola, Kevin Cameron, Dan FitzPatrick, Vassilios Gerousis, Ian Getreu, and others. 1996. Verilog-A Language Reference Manual, Analog Extensions to Verilog HDL, Version 1.0. *Open Verilog International* (Aug. 1996).
[3] Martin Barnasconi. 2010. SystemC AMS Extensions: Solving the Need for Speed. *DAC Knowledge center* (May 2010).
[4] Martin Barnasconi, C Grimm, M Damm, K Einwich, MM Louërat, T Maehne, F Pecheux, and A Vachoux. 2010. SystemC AMS extensions user's guide. *Accellera Systems Initiative* (2010).
[5] Kemal Çağlar Coşkun, Muhammad Hassan, and Rolf Drechsler. 2022. Equivalence Checking of System-Level and SPICE-Level Models of Linear Analog Filters. In *DDECS*. Prague, Czech Republic, 160–165. https://doi.org/10.1109/DDECS54261.2022.9770142
[6] Kemal Çağlar Coşkun, Muhammad Hassan, and Rolf Drechsler. 2022. Equivalence Checking of System-Level and SPICE-Level Models of Linear Circuits. *Chips* 1, 1 (June 2022), 54–71. https://doi.org/10.3390/chips1010006
[7] Kemal Çağlar Coşkun, Muhammad Hassan, and Rolf Drechsler. 2023. Equivalence Checking of System-Level and SPICE-Level Models of Static Nonlinear Circuits. In *DATE*. IEEE, Antwerp, Belgium, 1–6. https://doi.org/10.23919/DATE56975.2023.10137179
[8] R. Drechsler (Ed.). 2004. *Advanced Formal Verification*. Springer, New York.
[9] Rolf Drechsler. 2018. *Formal System Verification*. Springer.
[10] Walter Hartong, Ralf Klausen, and Lars Hedrich. 2004. Formal Verification for Nonlinear Analog Systems: Approaches to Model and Equivalence Checking. In *Advanced Formal Verification*. Springer, New York, 205–245.
[11] L. Hedrich and E. Barke. 1995. A Formal Approach to Nonlinear Analog Circuit Verification. In *Proceedings of IEEE International Conference on Computer Aided Design (ICCAD)*. 123–127. https://doi.org/10.1109/ICCAD.1995.480002
[12] Lars Hedrich and Walter Hartong. 2001. Approaches to Formal Verification of Analog Circuits. In *Low-Power Design Techniques and CAD Tools for Analog and RF Integrated Circuits*. Springer US. https://doi.org/10.1007/0-306-48089-1_8
[13] Chung-Wen Ho, A. Ruehli, and P. Brennan. 1975. The Modified Nodal Approach to Network Analysis. *IEEE TCAS-I* 22, 6 (June 1975), 504–509.
[14] R. Marani and A. G. Perri. 2016. A Simulation Study of Analogue and Logic Circuits with CNTFETs. *ECS Journal of Solid State Science and Technology* 5, 6 (April 2016), 38. https://doi.org/10.1149/2.0241606jss
[15] Paul Molitor and Janett Mohnke. 2007. *Equivalence checking of digital circuits: fundamentals, principles, methods*. Springer Science & Business Media.
[16] Laurence William Nagel. 1973. SPICE-simulation program with integrated circuit emphasis. *Electronics Research Lab., Univ. of California, Berkeley* (1973).
[17] Muharrem Orkun Saglamdemir, Gunhan Dundar, and Alper Sen. 2015. An Analog Behavioral Equivalence Checking Methodology for Simulink Models and Circuit Level Designs. In *SMACD*. IEEE, Istanbul.
[18] Amandeep Singh and Peng Li. 2010. On Behavioral Model Equivalence Checking for Large Analog/Mixed Signal Systems. In *ICCAD*. IEEE, 55–61.
[19] Sebastian Steinhorst and Lars Hedrich. 2010. Advanced Methods for Equivalence Checking of Analog Circuits with Strong Nonlinearities. *Formal Methods in System Design* 36, 2 (June 2010), 131–147. https://doi.org/10.1007/s10703-009-0086-9
[20] Ahmad Tarraf, Lars Hedrich, Niklas Kochdumper, Malgorzata Rechmal-Lesse, and Markus Olbrich. 2020. Equivalence Checking Methods for Analog Circuits Using Continuous Reachable Sets. In *ISVLSI*. IEEE, 7–12.
[21] Mohamed H. Zaki, Sofiène Tahar, and Guy Bois. 2008. Formal Verification of Analog and Mixed Signal Designs: A Survey. *Microelectronics Journal* 39, 12 (Dec. 2008), 1395–1404. https://doi.org/10.1016/j.mejo.2008.05.013