IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 19 June 2023, accepted 9 July 2023, date of publication 17 July 2023, date of current version 25 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3295825

RESEARCH ARTICLE

Reachability Analysis for Nonlinear Analog/Mixed-Signal Circuits With Trajectory-Based Reachable Sets

SEYOUNG KIM[1,2] AND JAEHA KIM[1], (Senior Member, IEEE)
[1]Department of Electrical and Computer Engineering, Inter-University Semiconductor Research Center, Seoul National University, Seoul 08826, South Korea
[2]Memory Business Division, Samsung Electronics, Hwaseong 18448, South Korea

Corresponding author: Jaeha Kim (jaeha@snu.ac.kr)

This work was supported in part by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government [Ministry of Science and Technology (MSIT)] under Grant 2020R1A4A4079177, and in part by the IC Design Education Center (IDEC), South Korea, for Electronic Design Automation (EDA) tool.

ABSTRACT This paper presents an efficient and scalable reachability analysis algorithm for nonlinear analog/mixed-signal circuits. In particular, it addresses the challenges in computing the time evolution of the reachable set of the circuit's continuous states and its intersection with the guard planes, i.e., the partitioning planes of the equivalent piecewise-linear system modeling the circuit's dynamics. The proposed algorithm utilizes a trajectory form of the reachable set, which can describe its exact evolution over time using analytical expressions until the set crosses one of the guard planes of the system. When it does, a naive computation of the reachable set may require splitting the set into multiple sub-sets, each using its own trajectory form. This is problematic since the number of reachable sets may grow indefinitely over time. To mitigate this, this work proposes a way of processing a group of reachable sets together that cross a common guard plane during a finite time interval. This method can keep the number of sets and the associated computational cost constant over time. The experimental results with a DC–DC converter example demonstrate that the proposed algorithm can achieve the average speed-ups of 79–107× compared to the existing algorithms with errors of less than 2%.

INDEX TERMS Analog/mixed-signal circuits, guard intersection, hybrid systems, piecewise-linear system, reachability analysis, safe operating area, safety verification.

I. INTRODUCTION

REACHABILITY analysis (RA) [1], [2], [3], [4] verifies the safety of a system by finding all the states that can be reached from a specified range of initial states (called the reachable set $\mathcal{R}$) and checking if $\mathcal{R}$ is within a desired target range. RA is a fundamental method for performing model checking on communication protocols [1], digital hardware systems [5], [6], and software [7]. This is widely being used for verifying modern safety-critical systems, wherein failures have significant consequences, i.e., enormous costs or even loss of human lives [8], such as sudden unintended acceleration [9], and collisions involving autonomous vehicles [10], [11] or aircraft [12]. Ever since it was shown that RA can be extended to hybrid systems [2], [3], which contain both discrete and continuous states, there has been a myriad of efforts attempting to verify analog/mixed-signal (AMS) circuits or systems. However, these efforts remain constrained to the analysis of the simple behavior of very small circuits, such as verifying the start-up of oscillators [13], [14], flip-flop [15], $\Delta\Sigma$-modulators [13], static random access memory (SRAM) [16], phase-locked loops (PLL) [17], [18], and DC–DC converters [19], [20], rendering them unsuitable for real-world problems. This paper presents an efficient and scalable approach to performing RA for nonlinear AMS circuits.

Most RA methods for AMS circuits in literature start by modeling a nonlinear AMS circuit as piecewise-linear (PWL)

The associate editor coordinating the review of this manuscript and approving it for publication was Sneh Saurabh.

systems [21]. In other words, the continuous state space of a nonlinear circuit or system is partitioned into a set of disjoint sub-regions $S_1, S_2, \ldots, S_p$, and the continuous state $x(t) \in \mathbb{R}^n$ within each sub-region is governed by a different linear differential equation. Early works focused on representing the region occupied by the reachable states, e.g., using a set of hypercubes [7], polyhedra [22], or ellipsoids [23]. Later works focused on improving the efficiency of computing the time evolution of the reachable states by utilizing superposition using zonotope [24] and support functions [25], and combining machine learning (ML) and satisfiability modulo theories (SMT) techniques to further accelerate the state exploration [18].

Our previous work [26] showed that the problem of computing the time evolution of the reachable set within a linear sub-region can be solved by using a trajectory form, which describes the reachable set as a zonotope each of which endpoints has an analytical expression describing its trajectory over time t. This *trajectory form of reachable state (TRS)* can describe the exact time evolution of the states without having to discretize the state space or time and, therefore, avoid all the related issues found in the previous works.

Now the remaining challenge for performing RA for a PWL system is with computing the intersection between the reachable state at time t, $\mathcal{R}(t)$, and the boundaries between the sub-regions, called *guard planes* $\mathcal{G}$, as illustrated in Fig. 1. Particularly, the geometrical computation of the intersection $\mathcal{R}(t) \cap \mathcal{G}$ and its subsequent evolution in a new sub-region often requires set splitting. For example, [27] describes a way of computing the intersection of the flowpipe reachable set with polyhedral guards by formulating a convex minimization problem. To reduce over-approximation errors, one flowpipe may have to be split into multiple flowpipes as a result [28], and despite the efforts of merging them afterward, the number of flowpipes to be maintained during the analysis may grow indefinitely over time, limiting scalability. The same challenge exists with the trajectory form as well.

To address this challenge, this paper proposes a way of computing the guard intersection (GI) $\mathcal{R}(t) \cap \mathcal{G}$ while combining multiple reachable sets described in trajectory forms, i.e., TRSs. The proposed method leverages the fact that we can compute the exact time interval of a TRS crossing a given $\mathcal{G}$ by solving the analytical expressions. This time interval can be divided into a set of multiple-unit intervals with finite durations. When multiple TRSs cross the same $\mathcal{G}$ within the same unit time interval, we take this opportunity to compute their GIs together and produce a merged result. For computing the intersection itself, we adopted the scalable technique presented in [29]. We found this approach can effectively keep the number of TRSs from growing indefinitely, since in most of the systems with recurring GIs, the GIs tend to occur at clustered time intervals.

The experimental results show that an average speed-up of 79–107× is possible compared to the existing methods including SpaceEx [28], with a safety-bound estimation error of less than 2%. More importantly, the example of a digitally-controlled closed-loop DC–DC converter demonstrates that the number of TRSs can be kept constant even when the GI occurs repeatedly, with a 40× average speed-up and an error of less than 2% using the same configuration.

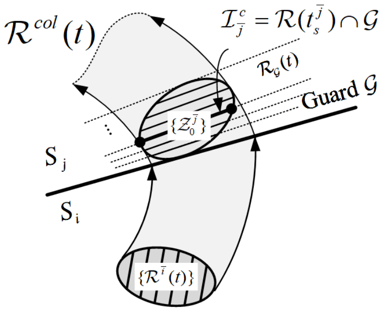

$\mathcal{R}^{col}(t)$
$\mathcal{I}^{c}_{\bar{j}} = \mathcal{R}(t^{\bar{j}}_{s}) \cap \mathcal{G}$
$\mathcal{R}_{\mathcal{G}}(t)$
Guard $\mathcal{G}$
$\{\mathcal{Z}^{\bar{j}}_{0}\}$
S_j
S_i
$\{\mathcal{R}^{\bar{i}}(t)\}$

FIGURE 1. **The proposed multiple trajectories form reachable set $\mathcal{R}^{col}(t)$ for PWL systems. The subsequent $\mathcal{R}^{col}(t)$ after GI starts from the subset of $\mathcal{G}$.**

The remainder of the paper is organized as follows: Section II introduces the TRS, and Section III describes the proposed RA algorithm with further details on computing the GIs and merging multiple TRSs. Section IV presents the experimental results with a few AMS circuit examples, and Section V concludes the study.

II. TRAJECTORY FORM OF REACHABLE SET

This section defines the TRS $\mathcal{R}(t)$ in a PWL system. The continuous space X of the employed system is partitioned into a set of disjoint sub-regions S_q's, indexed by the discrete state $q \in Q$. Assume that $\mathcal{R}(t)$ from sub-region S_i intersects the sub-region S_j where $i, j \in Q$, this transition is denoted as (i, j) and its hyperplane guard $\mathcal{G}_{(i,j)}$ is defined as follows:

$$\mathcal{G}_{(i,j)} : w \cdot x + b = 0, \tag{1}$$

where, in simplified notation, $w = w_{(i,j)} \in \mathbb{R}^n$ is a unit vector normal to each hyperplane, and $b = b_{(i,j)} \in \mathbb{R}^n$ is an offset.

The continuous state variable $x(t)$ and output $y(t) \in \mathbb{R}^{n_o}$ are governed by the linear differential equation of the system, given by

$$\dot{x}(t) = A_i x(t) + Bu(t), \quad y(t) = Cx(t) + Du(t), \tag{2}$$

where $A_i \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times n_i}$, $C \in \mathbb{R}^{n_o\times n}$, and $D \in \mathbb{R}^{n_o\times n_i}$. The trajectory of $x(t)$ starting from the initial state x_0 at time $t = t_0$ can be expressed as

$$x(t) = e^{A_i t} x_0 + \int_{t_0}^{t} e^{A_i(t-\tau)} Bu(\tau) d\tau. \tag{3}$$

When an external control signal $d(t)$ is given, the input of the system equation of S_i, $u(t) \in \mathbb{R}^{n_i}$ is generated depending on the circuit topology. It has been shown in [30] that when each $u(t)$ has the form:

$$u(t) = \sum_k c_k t^{m_k} e^{-a_k t}, \tag{4}$$

where the coefficients a_k's and c_k's are complex numbers, and m_k's are non-negative integers, the solutions for $x(t)$'s and $y(t)$'s governed by (2) have the same expressions. Although we assume a single-input case (i.e., $n_i = 1$) here, multi-input cases can be accommodated in a straightforward manner. The main advantage of this representation is that it is possible to express all types of analog input signals of $u(t)$ using the sum of exponential representations, i.e., tuples of coefficients $\{(a_k, c_k, m_k), \ldots\}$'s representing the s-domain signal form. For example, a step input of $u(t)$ can be expressed as $U(s) = 1/s \Leftrightarrow \{(0+0j, 1+0j, 1)\}$, a ramp function can be expressed as $U(s) = 1/s^2 \Leftrightarrow \{(0+0j, 1+0j, 2)\}$, and a function of the step response of a linear filter having a pole frequency ω_0 can be expressed as $U(s) = 1/s - 1/(s+\omega_0) \Leftrightarrow \{(0+0j, 1+0j, 1), (\omega_0+0j, -1+0j, 1)\}$. Note that this form of $u(t)$ can be extended to a concatenated form where $u(t)$ takes a different expression depending on the time interval (e.g.,$u_0(t-t_0)$ starting from t_0, $u_1(t-t_1)$ starting from t_1, and so on). For simple notation, we abbreviate this as $u(t)$ without the subscripts for time intervals.

The initial set of states at time t_0, that is, $\mathcal{R}(t_0)$, is represented as a zonotope $\mathcal{Z}_0$ as

$$\mathcal{Z}_0 = (c, \langle g_1, \ldots, g_r\rangle) = \{c + \sum_h \alpha_h g_h | \alpha_h \in [-1, 1]\}, \quad (5)$$

where $c \in \mathbb{R}^n$ is a center point, and g_h's $\in \mathbb{R}^n$ are a set of generators [24]. Any point within the zonotope in (5) can be expressed as a linear combination of g_h's from c.

It has been shown in [26] that, using the superposition principle, each $\mathcal{R}(t) \subseteq S_i$ for $t \geq t_0$ can be represented as zonotope where all the vectors c and g_h's can be expressed as functions of time t using (4), that is,

$$\mathcal{R}(t) = (c(t), \langle g_1(t), \ldots, g_r(t)\rangle). \quad (6)$$

Sampling values for $\mathcal{R}(t)$ at t yields a zonotope set of states as in (5), which is denoted as `sample(·)`.

In this study, a procedure named `getTrajectoryZonotope(·)` computes the TRS $\mathcal{R}(t)$ from the $\mathcal{Z}_0$ sampled by $\mathcal{R}(t_0)$ in the current sub-region S_i, as given in (6). In [30], it was shown that this computation can be carried out in the Laplace s-domain. For instance, the time-domain expressions of $u(t)$ can be transformed into the Laplace-domain expression $U(s)$ as

$$u(t) = \sum_k c_k t^{m_k} e^{-a_k t} \xrightarrow{\mathcal{L}} U(s) = \sum_k \frac{b_k}{(s+a_k)^{m_k+1}}, \quad (7)$$

and the Laplace-domain transform of $x(t)$, $X(s)$, can be computed using the matrices given in (2) as

$$X(s) = (sI - A_i)^{-1}BU(s) + (sI - A_i)^{-1}x(t_0), \quad (8)$$

which can be transformed back into a time-domain expression as in (4). The TRS $\mathcal{R}(t)$ in (5) is computed by applying this procedure to the vectors of $\mathcal{Z}_0$, or equivalently, to its center c and generators g_h's individually. To avoid redundant computations of the transfer-function matrices $(sI - A_i)^{-1}B$ and $(sI - A_i)^{-1}$ for the same linear differential equation, the proposed method stores a collection of previously calculated transfer-function matrices.

Algorithm 1 Computing the Guard Intersection

```
Input : $\mathcal{Z}_0 := \{\mathcal{Z}_{\bar{i}=1,\ldots,N_s}\}$, $\mathcal{G} := \mathcal{G}_{(i,j)} = (w_{(i,j)}, b_{(i,j)})$,
        $u(t)$
Output: $\mathcal{I}^c$, $t_s$
1  $\mathcal{R}(t), \mathcal{I}_0 \leftarrow \emptyset$;
2  for $\bar{i} \in (1, \ldots, N_s)$ do
3      $\mathcal{R}^{\bar{i}}(t) \leftarrow getTrajectoryZonotope(i, \mathcal{Z}^{\bar{i}}, u(t))$;
4      $\mathcal{R}^{col}(t) \leftarrow \mathcal{R}^{col}(t) \cup \{\mathcal{R}^{\bar{i}}(t)\}$;
5  end
6  $([t_1, t_2], flag) \leftarrow findCrossZonotope(\mathcal{R}^{col}(t), T)$;
7  $t_s \leftarrow \{t_s^{\bar{j}} = (t_1 + \bar{j}\frac{(t_2-t_1)}{N_s}) \mid_{\bar{j}=0,\ldots,N_s}\}$ ;
8  $D \leftarrow getOrthogonalBasis(\mathcal{G}, \mathcal{R}^{col}(t))$;
9  for $\bar{i} = (1, \ldots, N_s) \wedge \bar{j} = (0, \ldots, N_s) \wedge \bar{k} = (1, \ldots, n-1)$
   do
10     $\mathcal{Z}_s^{\bar{i},\bar{j}} \leftarrow sample(\mathcal{R}^{\bar{i}}(t), t_s^{\bar{j}})$ ;
11     $\mathcal{I}^c_{\bar{i},\bar{j},\bar{k}} \leftarrow getIntersect2D(\mathcal{Z}_s^{\bar{i},\bar{j}}, l_{\bar{k}})$ /* $[m, M]$ */
12 end
13 for $\bar{j} = (0, \ldots, N_s) \wedge \bar{k} = (1, \ldots, n-1)$ do
14     $\mathcal{I}^c_{\bar{j},\bar{k}} \leftarrow \bigcup_{\bar{i}=1}^{N_s} \mathcal{I}^c_{\bar{i},\bar{j},\bar{k}}$
15 end
16 return $\mathcal{R}^{col}(t), \mathcal{I}^c, t_s$
```

III. HYBRID REACHABILITY ANALYSIS WITH GUARD INTERSECTION

The proposed algorithm computes the GIs of the TRS, i.e., $\mathcal{R}^{col}(t) \cap \mathcal{G}$ as an approximate zonotope subset of $\mathcal{G}$. It computes the GI as a set of ranges $[m, M]$ for each direction defined by a set of unit vectors $l_{\bar{k}}$ for $\bar{k} = 1, 2, \ldots$ and transforms it back to TRS again to be computed iteratively. The algorithm is explained in the following two subsections.

A. COMPUTATION OF GUARD INTERSECTION

This subsection describes the procedure for computing the GI ranges in four steps: (i) building the TRS from an initial set, (ii) detecting the GI, (iii) computing the ranges representing the GI by projection, and (iv) merging the ranges. The pseudo-code of the algorithm for computing the GI ranges is listed in Algorithm 1.

1) BUILDING TRAJECTORY-FORM REACHABLE SETS $\mathcal{R}^{col}(t)$

In the first step of the algorithm, the procedure `GetTrajectoryZonotope` computes the TRS $\mathcal{R}^{col}(t)$ in (9) using the corresponding linear system given in (2), associated with the state for the current sub-region q from $\mathcal{Z}_0 = \bigcup_{\bar{i}=1}^{N_s} \mathcal{Z}^{\bar{i}}$. The resulting $R^{col}(t)$, which contains N_s-fold $\mathcal{R}^{\bar{i}}(t)$'s in (6), is defined as

$$R^{col}(t) = \{\mathcal{R}^{\bar{i}}(t), \mathcal{R}^{\bar{j}}(t) \;\text{ for }\; \bar{i}, \bar{j} = 1, \ldots, N_s\}, \quad (9)$$

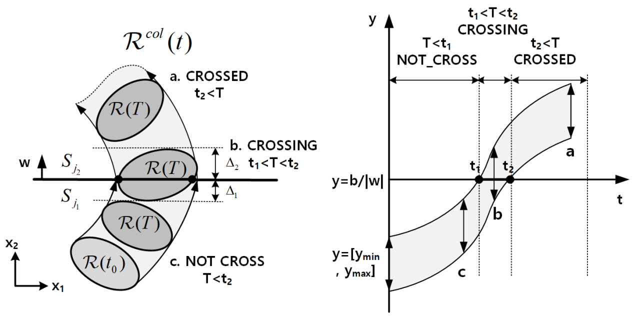


FIGURE 2. The procedure 'findCrossZonotope' detects the occurrence of GIs, resulting in three different cases depending on the relation between $[t_1, t_2]$ and T.

where $\bar{i}, \bar{j}$ are the set index for subregions S_i and S_j, respectively.

2) DETECTION OF GUARD INTERSECTIONS

Next, the algorithm detects every GI between $\mathcal{R}^{\bar{i}}(t) \in \mathcal{R}^{col}(t)$ and $\mathcal{G}$. It computes the time range $[t_1, t_2]$ for which the intersections $\mathcal{R}^{col}(t) \cap \mathcal{G} \neq \emptyset$ occur within the local time bound T where the current input $u(t)$ is valid. As shown in Fig. 2, using the distance function of zonotopes introduced in [28], the `findCrossZonotope(·)` procedure iteratively finds the values of the entering time t_1 and exiting time t_2 until T. The function has been extended for the TRS by computing the distance of each $\mathcal{R}^{\bar{i}}(t)$ at arbitrary $t = t'$ from the $\mathcal{G}$, given as

$$y_{min}(w, t') = c(t') \cdot w - \sum\nolimits_{h=1}^{r} |g_h(t') \cdot w|,$$
$$y_{max}(w, t') = c(t') \cdot w + \sum\nolimits_{h=1}^{r} |g_h(t') \cdot w|, \qquad (10)$$

each of which is represented as a piecewise-continuous function based on the polarity of the right term $g_h(t') \cdot w$. Because each continuous segment of the function is represented in trajectory form as in (4), finding t_1 and t_2 is equivalent to finding the t' used in the expressions $y_{min}(w, t') = 0$ and $y_{max}(w, t') = 0$. The values of t' can be found using a scalar optimization algorithm.

Three distinguishable intersecting cases arise for this evaluation depending on the position of $\mathcal{R}(T)$ with respect to the $\mathcal{G}$, as shown in Fig. 2. If the procedure finds t_1 and t_2 within the local time bound ($t_1, t_2 < T$), it normally returns the resulting pair of times and the flag 'CROSSED'. If t_1 and t_2 are not found within the T, then it returns the last set of states $\mathcal{R}^{col}(T)$ sampled at $t = T$ with the flag 'NOT CROSS.' If the time ranges are found but the end time of the intersection t_2 is greater than the local time bound T ($t_1 < T < t_2$), it returns the flag 'CROSSING' but requires some modification for generating the next initial set for $\mathcal{R}^{col}(t)$, which is explained in the next subsection.

3) COMPUTING INTERSECTIONS AS RANGES $\mathcal{I}^c_{\bar{i},\bar{j},\bar{k}}$ WITH 2-D PROJECTION

We define the set of ranges expressing the GI as $\mathcal{I}^c = \{\mathcal{I}^c_{\bar{i},\bar{j},\bar{k}}\}$ for directions of $l_{\bar{k}}$'s. Thus, we need to find $l_{\bar{k}}$ that spans $\mathcal{G}$.

Algorithm 2 GetIntersect2D($\mathcal{Z}_s^{\bar{i},\bar{j}}$, $l_{\bar{k}}$)

```
Input  : $\mathcal{Z}_s^{\bar{i},\bar{j}}$, $w$, $l_{\bar{k}}$, $D$
Output: $\mathcal{I}^c_{\bar{i},\bar{j},\bar{k}}$
/* Projected zonotope for $(w, l_{\bar{k}})$ */
$\mathcal{Z}_{(w,l_{\bar{k}})} \leftarrow (Proj_{w,l_{\bar{k}}}(c), \langle Proj_{w,l_{\bar{k}}}(g_1), \ldots, Proj_{w,l_{\bar{k}}}(g_r)\rangle)$ ;
Project $\mathcal{G}_h \rightarrow x = \gamma_k$ ;
Pick $g_0$ from generators $\langle g_h \rangle$ of $\mathcal{Z}_{(w,l_{\bar{k}})}$ s.t. $v_1$ is most close
  to $x = \gamma_k$ ;
$v_1 \leftarrow c + g_0$ ;
Sort $\langle g_h \rangle$ in tangent order $arctan(g_x/g_y)$ ;
Pop $g := (g_x, g_y)$ in $\langle g_h \rangle$ ;
$v_2 \leftarrow c + g$ ;
/* traverse vertex by counter-clockwise) */
while $\langle g_h \rangle \neq \emptyset$ do
    $y \leftarrow$ Get line intersection between $\overline{v_1 v_2}$ and $x = \gamma_k$ ;
    /* next line segment */
    $v_1 \leftarrow v_2$ ;
    Pop $g := (g_x, g_y)$ in $\langle g_h \rangle$ ;
    $v_2 \leftarrow v_2 + 2g$ ;
    /* counter-clockwise */
    $\mathcal{I}^c_{\bar{i},\bar{j},\bar{k}} \leftarrow [m, M]$ ;
    $M \leftarrow y$ if $y > M$
end
/* by clockwise) */
$\langle g_h \rangle$ in tangent order $arctan(g_x/g_y)$ ;
while $\langle g_h \rangle \neq \emptyset$ do
    ...
    $v_2 \leftarrow v_2 - 2g$/* clockwise */
    ...
    $m \leftarrow y$ if $y < m$
end
$\mathcal{I}^c_{\bar{i},\bar{j},\bar{k}} \leftarrow [m, M]$ ;
return $\mathcal{I}^c_{\bar{i},\bar{j},\bar{k}}$
```

The discrete time steps are defined as $t_s^{\bar{j}} = t_1 + \bar{j}(t_2 - t_1)/N_s$ spanning the intersecting time range $[t_1, t_2]$, where $\bar{i}, \bar{j} = 0, \ldots, N_s$ and $\bar{k} = 1, \ldots, n$. Each resulting $\mathcal{I}^c_{\bar{i},\bar{j},\bar{k}}$ is given by $[m, M]$ where $m, M \in \mathbb{R}$ denote the minimum and maximum values of the range of values, respectively.

The proposed algorithm adopts the projection method introduced in [29] that efficiently computes GIs in high-dimensional state space and extends the algorithm for the TRS. It is implemented as the `getIntersect2D(·)` procedure that computes the GI by intermediately computing $\mathcal{Z}_s^{\bar{i},\bar{j}}$ representing zonotope sampled from $\mathcal{R}^{\bar{i}}(t)$ at time $t_s^{\bar{j}}$ via the projection of $\mathcal{R}^{\bar{i}}(t)$ into two-dimensional spaces generated for pairs of vectors, w and $l_{\bar{k}}$'s. The projection of a continuous state $x(t)$ is defined by $Proj_{w,l_{\bar{k}}}(x(t)) = (w \cdot x(t), l_{\bar{k}} \cdot x(t))$. Using the above property, the projection for a zonotope $\mathcal{Z}$ given in (5) can be expressed as $Proj_{w,l_{\bar{k}}}(\mathcal{Z}) \mapsto \mathcal{Z}_{(w,l_{\bar{k}})} = (Proj_{w,l_{\bar{k}}}(c), \langle Proj_{w,l_{\bar{k}}}(g_1), \ldots, Proj_{w,l_{\bar{k}}}(g_r)\rangle)$, whose resulting projected zonotope, $\mathcal{Z}_{(w,l_{\bar{k}})} \in \mathbb{R}^2$, is a subset of the two-dimensional state space, i.e., the $(w, l_{\bar{k}})$-space. Finally, the procedure `getIntersect2D` computes the GI for each projected zonotope $Proj_{w,l_{\bar{k}}}(\mathcal{Z}_s^{\bar{i},\bar{j}})$ and yields $\mathcal{I}^c_{\bar{i},\bar{j},\bar{k}}$ that

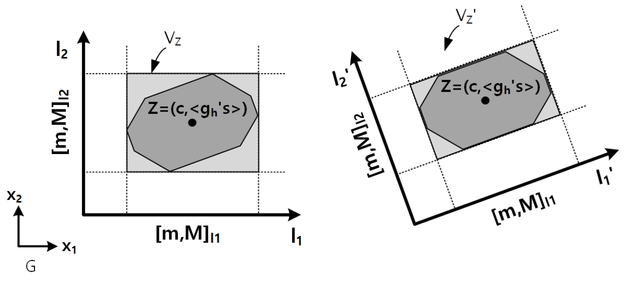


FIGURE 3. Finding a set of optimal basis D using orthogonal vectors $l_{\bar{k}}$'s spanning $\mathcal{G}$. The initial choice using one of the g_h's helps to find the optimal $l_{\bar{k}}$'s.

scales in each direction along $l_{\bar{k}} \in D$. The ranges of 2-D intersection between $\overline{v_1 v_2}$, where v_1 and v_2 are the vertices of the intersection, and the projected guard hyperplane $x = \gamma$ can be simply obtained by the following algebra as:

$$y = (\gamma_k - v_{1x})\frac{v_{2y} - v_{1x}}{v_{2x} - v_{1x}} + v_{1y} \tag{11}$$

where $v_1 = (v_{1x}, v_{1y})$ and $v_2 = (v_{2x}, v_{2y})$. The detailed algorithm used in `getIntersect2D` is listed in Algorithm 2.

Note that this approximation method would result in an additional over-approximation error depending on how the $l_{\bar{k}}$'s are chosen, as shown in Fig. 3. Therefore, the `getOrthogonalBasis`$(\cdot)$ procedure finds the set of optimal orthogonal $l_{\bar{k}}$'s, by selecting the one that has a minimum-volume hypercube enclosing the GIs, $\bigcup_{\bar{i}} \mathcal{R}^{\bar{i}}(t') \cap \mathcal{G}$ sampled at $t' = t_{mid} = (t_1 + t_2)/2$. The search process finds the $l_{\bar{k}}$'s that minimizes the expression for the volume of zonotope $V_{\mathcal{Z}} = \prod_{\bar{k}} \sum_h |g_h \cdot l_{\bar{k}}|$ for all $\bar{k}$ and h. If an initial value of $l_{\bar{k}}$ is selected from the g_h's of the sampled zonotope $\mathcal{R}^{\bar{i}}(t_{mid})$, the initial search time of the process can be reduced, as illustrated in Fig. 3.

Even though the zonotopes are projected in a two-dimensional space, enumerating all the vertices of the zonotope with many generators can slow down the analysis. However, this procedure can efficiently compute intersections by prioritizing the vertices based on their distance to the $\mathcal{G}$ without enumerating all the vertices, extending the idea in [29] to the proposed method.

4) MERGING INTERSECTIONS $\mathcal{I}^c_{\bar{i},\bar{j},\bar{k}}$ TO $\mathcal{I}^c_{\bar{j},\bar{k}}$

In addition, the resulting set of GI ranges $\mathcal{I}^c$ for $\mathcal{R}^{\bar{i}}(t)$'s consists of $N_s(N_s + 1)(n - 1)$-fold pairs of values $[m, M]$. Based on the fact that every $\mathcal{I}^c_{\bar{i},\bar{j},\bar{k}}$ with an identical index $\bar{i}$ shares the same guard-crossing time $t_s^{\bar{j}}$ and the same direction of $l_{\bar{k}}$, they can be merged by taking their union, resulting in the new $(N_s + 1)(n - 1)$-fold $\mathcal{I}^c_{\bar{j},\bar{k}}$.

B. SUBSEQUENT INITIAL ZONOTOPE

Algorithm 3 lists the procedure for computing $\mathcal{Z}_0$ in the new sub-region S_j from $\mathcal{I}^c_{\bar{j},\bar{k}}$, where $\mathcal{Z}_0$ is N_s-fold zonotopes, i.e.,

Algorithm 3 Computing Next Initial Zonotope $\mathcal{Z}_0^{\bar{j}}$

```
Input  : $\mathcal{I}^c := \{[m, M]_{\bar{j}=(0,\ldots,N_s),\bar{k}=(1,\ldots,n-1)}\}$
Output: $\mathcal{Z}_0 := \{\mathcal{Z}_0^{\bar{j}} |_{\bar{j}=(1,\ldots,N_s)}\}$, $\mathcal{R}^{col}(t)$, $t_r$
$\mathcal{Z}_0 \leftarrow \emptyset$; $\mathcal{Z}_{\mathcal{G}0} \leftarrow \langle c, D \rangle$;
$\mathcal{R}_{\mathcal{G}}(t') \leftarrow getTrajectoryZonotope(j, \mathcal{Z}_{\mathcal{G}0}, u(t'))$;
for $\bar{j} \in (0, 1, \ldots, N_s)$ do
    $t_r^{\bar{j}} \leftarrow t_2 - t_1 - t_s^{\bar{j}}$;
    $\mathcal{Z}_{\mathcal{G}}^{\bar{j}} \leftarrow sample(\mathcal{R}_{\mathcal{G}}(t'), t_r^{\bar{j}})$;
end
for $\bar{j} \in (1, \ldots, N_s)$ do
    for $\bar{k} \in (1, \ldots, n-1)$ do
        $\mathcal{I}_{\bar{j},\bar{k}} \leftarrow \mathcal{I}^c_{\bar{j}-1,\bar{k}} \cup \mathcal{I}^c_{\bar{j},\bar{k}}$ ;
    end
    /* Initial set for next cycle */
    $\mathcal{Z}_c^{\bar{j}-1} \leftarrow I2Z(\mathcal{Z}_{\mathcal{G}}^{\bar{j}-1}, \mathcal{I}_{\bar{j}-1,\bar{k}})$; $\mathcal{Z}_c^{\bar{j}} \leftarrow I2Z(\mathcal{Z}_{\mathcal{G}}^{\bar{j}}, \mathcal{I}_{\bar{j},\bar{k}})$ ;
    $\mathcal{Z}_0^{\bar{j}} \leftarrow \widehat{CH}(\mathcal{Z}_c^{\bar{j}-1}, \mathcal{Z}_c^{\bar{j}})\}$;
    $\mathcal{Z}_0 \leftarrow \mathcal{Z}_0 \cup \mathcal{Z}_0^{\bar{j}}$;
    /* Reachable sets in $S_j$ */
    $\mathcal{Z}^{\bar{j}} \leftarrow \widehat{CH}(I2Z(\mathcal{Z}_{\mathcal{G}}^{N_s-1}, \mathcal{I}_{\bar{j}-1,\bar{k}}), I2Z(\mathcal{Z}_{\mathcal{G}}^{N_s}, \mathcal{I}_{\bar{j},\bar{k}}))$ ;
    $\mathcal{R}^{\bar{j}}(t) \leftarrow getTrajectoryZonotope(q_j), \mathcal{Z}^{\bar{j}}, u_i(t))$;
    $\mathcal{R}^{col}(t) \leftarrow \mathcal{R}^{col}(t) \cup \mathcal{R}^{\bar{j}}(t)$;
end
return $\mathcal{Z}_0, \mathcal{R}^{col}(t), t_r$
```

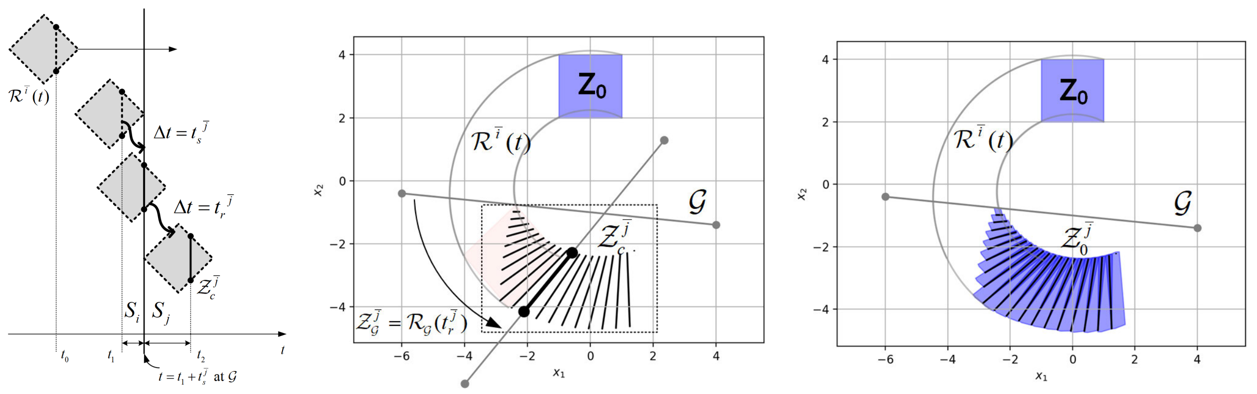
$\mathcal{Z}_0 = \bigcup_{\bar{j}} \mathcal{Z}_0^{\bar{j}}$, valid at $t = t_2$, that can be computed as the trajectory form of (4) in the subsequent GI cycle.

1) BUILDING REACHABLE SET OF GUARD HYPERPLANE $\mathcal{R}_{\mathcal{G}}(t)$

To obtain $\mathcal{Z}_0$, the proposed algorithm represents the time evolution of the states from the GI by obtaining a reachable set of $\mathcal{G}$ itself, i.e., $\mathcal{R}_{\mathcal{G}}(t' = t_{\bar{j}}^r)$, where t' denotes time after $t_{\bar{j}}^s$, that is, $t' = t - t_{\bar{j}}^s$, and $t_{\bar{j}}^r = t_2 - t_{\bar{j}}^s$ indicates the remaining period until $t = t_2$ in the new sub-region S_j, as illustrated in Fig. 4 (a). After defining $\mathcal{Z}_{\mathcal{G}0} = (c, \langle l_1, \ldots, l_{n-1} \rangle)$ as the zonotope spanning the entire hyperplane of $\mathcal{G}$ where $c = |b|w \in \mathcal{G}$, we compute the TRS of $\mathcal{G}$, $\mathcal{R}_{\mathcal{G}}(t') = (c(t'), \langle l_1(t'), \ldots, l_{n-1}(t') \rangle)$ governed by a different linear system in (2) in the subsequent sub-region S_j starting from $\mathcal{Z}_{\mathcal{G}0}$.

Each $\mathcal{I}^c_{\bar{j},\bar{k}}$ representing GI at $t = t_s^{\bar{j}}$ that scales the zonotope sampled from $\mathcal{R}_{\mathcal{G}}(t)$, i.e., $\mathcal{Z}_{\mathcal{G}}^{\bar{j}} = \mathcal{R}_{\mathcal{G}}(t' = t_r^{\bar{j}})$ will undergo the conversion procedure `I2Z`$(\cdot)$. The procedure yields an equivalent cross-sectional zonotope valid at $t = t_2$, $\mathcal{Z}_c^{\bar{j}} = (c^{\bar{j}}, \langle g_1^{\bar{j}}, \ldots, g_{n-1}^{\bar{j}} \rangle)$, given as

$$c^{\bar{j}} = c(t_r^{\bar{j}}) + \sum_{\bar{k}=1}^{n-1} (m_{\bar{j},\bar{k}} + M_{\bar{j},\bar{k}}) l_{\bar{k}}(t_r^{\bar{j}})/2$$
$$g_{\bar{k}}^{\bar{j}} = (M_{\bar{j},\bar{k}} - m_{\bar{j},\bar{k}}) l_{\bar{k}}(t_r^{\bar{j}})/2, \tag{12}$$


$\mathcal{R}^{\bar{i}}(t)$
$\Delta t = t_s^{\bar{j}}$
$\Delta t = t_r^{\bar{j}}$
$\mathcal{Z}_c^{\bar{j}}$
S_i
S_j
t_0
t_1
t_2
t
$t = t_1 + t_s^{\bar{j}}$ at $\mathcal{G}$
Z_0
$\mathcal{R}^{\bar{i}}(t)$
$\mathcal{G}$
$\mathcal{Z}_c^{\bar{j}}$
$\mathcal{Z}_{\mathcal{G}}^{\bar{j}} = \mathcal{R}_{\mathcal{G}}(t_r^{\bar{j}})$
x_1
x_2
Z_0
$\mathcal{R}^{\bar{i}}(t)$
$\mathcal{G}$
$\mathcal{Z}_0^{\bar{j}}$
x_1
x_2

(a) Discrete time steps $t_s^{\bar{j}}$ and $t_r^{\bar{j}}$ (b) Cross-sectional zonotope $\mathcal{Z}_c^{\bar{j}}$ (N_s=20). I2Z(·) converts ranges into a zonotope $\mathcal{Z}_c^{\bar{j}}$. (c) Next initial set $\mathcal{Z}_0 = \{\mathcal{Z}^{\bar{j}}\}$ by merging pairs of zonotopes $\mathcal{Z}_c^{\bar{j}}$.

FIGURE 4. Example of reachable set computation from the ranges using Algorithm2.

where $m_{\bar{j},\bar{k}}$ and $M_{\bar{j},\bar{k}}$ indicate the minimum and maximum values in $\mathcal{I}_{\bar{j},\bar{k}}$, respectively, and the new time steps $\Delta t = t_r^{\bar{j}}$ specify the duration for the $\mathcal{R}_{\mathcal{G}}^{\bar{j}}(t')$ evolving from the time instant of GI $t = t_s^{\bar{j}}$, which is globally equivalent to $t = t_2$.

2) CONVERTING RANGES $\mathcal{I}_{\bar{j},\bar{k}}^c$'s INTO NEXT INITIAL SET $\mathcal{Z}_0^{\bar{j}}$'s

Next, the procedure $\widehat{\text{CH}}()$, which computes the convex hull of two zonotopes incorporated in [24], iteratively combines a pair of adjacent zonotopes $\mathcal{Z}_c^{\bar{j}-1}$ and $\mathcal{Z}_c^{\bar{j}}$ for $\bar{j} = 1, \ldots, N_s$, resulting in $\mathcal{Z}_0$ as $\mathcal{Z}_0 = \bigcup_{\bar{j}=1}^{N_s} \mathcal{Z}_0^{\bar{j}}$, as shown in Fig 4 (c). However, directly applying this procedure to the resulting zonotopes can result in a large over-approximation error owing to the zonotopes' asymmetry, as seen in the leading and trailing zonotopes in Fig 4 (b). Therefore, we consider the maximum range of the pair of $\mathcal{I}_{\bar{j}-1,\bar{k}}^c$ and $\mathcal{I}_{\bar{j},\bar{k}}^c$ for each $[t_s^{\bar{j}-1}, t_s^{\bar{j}}]$ in $\bar{j} = 0, \ldots, N_s$ to balance the size of the paired zonotopes. When the range of the smaller ones in all the paired zonotopes is extended, we obtain a hull of zonotopes with a small over-approximation error, as the zonotopes $\mathcal{Z}_0^{\bar{j}}$'s shown in Fig 4 (c).

In the case of a 'CROSSING' state, the algorithm cannot be applied for the $\mathcal{R}^{col}(t)$ because it partially remains in the current sub-region S_i, and the remaining part does not intersect $\mathcal{G}$ until the local time bound T, as shown in Fig. 4. To address this, we split the GI time range $t = [t_1, t_2]$ into two, depending on whether the corresponding part of $\mathcal{R}^{col}(t)$ crosses the guard or not: $[t_1, T]$ and $[T, t_2]$. First, we discretize $[T, t_2]$ into $t_s^{\bar{j}_2}$ with $\bar{j}_2 = 1, \ldots, N_s - K$ and compute a set of zonotopes $\mathcal{Z}_{0,2}$, as in the previously described method. For the remaining $[t_1, T]$, we segment the reachable set at $t = T$, i.e., $\mathcal{Z}_{0,1}^{\bar{j}} = \mathcal{R}^{\bar{j}_1}(t = T)$ into K-fold zonotopes $\mathcal{Z}_{0,1}^{\bar{j}_1}$ with $\bar{j}_1 = 1, \ldots, K$ that are uniformly spaced and spatially parallel to the guard plane. Note that we can choose a non-negative integer $K \in \mathbb{N}$ to be proportional to the distance from the guard, i.e., choosing $K/N_s \approx \Delta_1/(\Delta_1 + \Delta_2)$ will yield good accuracy. Then, the resulting initial set is given by the union of the two sets representing each part at $t = T$, as follows: $\mathcal{Z}_0 = \mathcal{Z}_{0,1} \cup \mathcal{Z}_{0,2}$ Thus, we can maintain N_s for $\mathcal{R}^{\bar{j}}(t)$'s with a relatively minimal error, without increasing the complexity of the overall algorithm.

3) COMPUTING REACHABLE SETS $\mathcal{R}^{\bar{j}}(t)$'s UNTIL $\mathcal{Z}_0^{\bar{j}}$'s

The reachable sets after each GI until $t = t_2$ are defined as a set of a new TRS $\mathcal{R}^{\bar{j}}(t)$ representing a set of states while $t = [t_{\bar{j}-1}^s, t_{\bar{j}}^s]$ for $\bar{j} = 1, \ldots, N_s$ to $t = t_2$. Each pair of zonotopes $\mathcal{Z}_{c0}^{\bar{j}-1}$ and $\mathcal{Z}_{c0}^{\bar{j}}$ are computed from the GI ranges as $\texttt{I2Z}(\mathcal{Z}_{\mathcal{G}}^{N_s-1}, \mathcal{I}_{\bar{j}-1,\bar{k}})$ and $\texttt{I2Z}(\mathcal{Z}_{\mathcal{G}}^{N_s}, \mathcal{I}_{\bar{j},\bar{k}})$, respectively. The initial set $\mathcal{Z}^{\bar{j}}$ for $\mathcal{R}^{\bar{j}}(t)$ can then be obtained by merging the pair by $\widehat{\text{CH}}(\mathcal{Z}_{c0}^{\bar{j}-1}, \mathcal{Z}_{c0}^{\bar{j}})$. Finally, we obtain $\mathcal{R}^{\bar{j}}(t)$ from each $\mathcal{Z}^{\bar{j}}$ using the GetTrajectoryZonotope procedure and collect them in $R^{col}(t)$.

C. TIME COMPLEXITY

The GetTrajectoryZonotope procedure has a complexity of $O(n^2 r)$ where r is the number of generators in the zonotope. In contrast, getIntersect2D has a complexity of $O(1)$. Because Algorithm 1 calls GetTrajectory-Zonotope N_s times and getIntersect2D $N_s^2 \cdot n$ times, its overall time complexity is $O(N_s^2 n + N_s n^2 r)$ and is independent of n. Thus, if N_s is smaller than n and r is limited to scale linearly with n using the dimensionality reduction technique in [24], the overall complexity becomes $O(n^3 N_s)$. Because Algorithm 2 also calls GetTrajectoryZonotope $N_s + 1$ times, the overall time complexity for both algorithms is $O(n^3 N_s)$.

D. SAFETY BOUNDS OF TRAJECTORY-FORM REACHABLE SETS

To verify the safety of the circuits using the computed TRS $\mathcal{R}^{col}(t)$, we compute its occupying region in the state space.

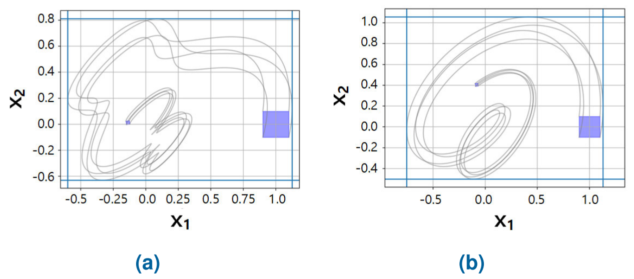


FIGURE 5. The examples of obtaining the bounds of the operating region with varying input $u_i(t)$. The bounds are indicated in blue lines.

For example, in the state space of (I_L, V_C), each range can be obtained using unit vectors $(1, 0)$ for I_L and $(0, 1)$ for V_C. Then, minimum and maximum values of the range can be found by searching the global peak value of all $\mathcal{R}^{\bar{i}}(t) \in \mathcal{R}^{col}$ using the distance function defined in (10), given by $[\mathrm{Min}_{\forall t' \in [0,T_{max}]}(y_{min}(l, t')), \mathrm{Max}_{\forall t' \in [0,T_{max}]}(y_{max}(l, t'))]$ for all $\bar{i}$, where T_{max} indicates the overall time bound of the analysis.

Fig. 5 displays examples of computing the bounds of $\mathcal{R}(t)$ obtained from the system by varying the input $u(t)$ in two axes. The system matrix of the given example is $A = [-1 - 4; 4 - 1]$ and $u(t) = e^{j\omega_0 t}$, where the frequency of $u(t)$ is randomly chosen, i.e., $\omega_0 \in [-50, 50]$ rad/s. For any trajectory of zonotope sets, the algorithm can accurately compute the bounds of the moving shapes regardless of the size of the time steps as it computes the minimum or maximum bounding values using an iterative search process from the continuous function defined in (4), instead of from the segmented sets sampled at discrete time steps.

IV. EXPERIMENTAL RESULTS

This section discusses the experimental results obtained from a 2-D hybrid system and DC–DC converters. The proposed algorithms were implemented in Python/C++, and the runtimes were measured using a 3-GHz Intel Xeon 8124M single-core CPU with 16GB memory.

A. NUMERICAL EXAMPLE

The first example is a 2-D linear hybrid system with no input, which is partitioned by the guard $\mathcal{G}_1$ with $w = (0.1, 1)$ and $b = -1$. The guard splits the state space into two regions, each with the system matrix

$$A_1 = \begin{pmatrix} 1 & -10 \\ 10 & 0 \end{pmatrix}, \; A_2 = \begin{pmatrix} 2 & -20 \\ 20 & 0 \end{pmatrix}. \tag{13}$$

Fig. 6 compares the reachable sets computed using the proposed algorithm, Girard's algorithm [29], and brute-force Monte–Carlo (MC) simulations, when the system starts with the initial states ranges $x_1 = [-1,1]$ and $x_2 = [2.5,3.5]$. The blue-colored areas highlight the reachable sets immediately after each GI, i.e., at $t = t_2$. The over-approximation error of each algorithm compared to the MC results can be computed from the area of its $\mathcal{R}$ and the hull of the MC samples $\mathcal{R}_{MC}$ as given below:

$$err_1 = Area(\mathcal{R} - \mathcal{R}_{MC})/Area(\mathcal{R}) \tag{14}$$

where the areas are estimated using another MC integration method. For four cycles of GIs, the proposed algorithm has err_1 of 0.506%, whereas Girard's algorithm has 2.477%, achieving a 4.89× reduced over-approximation error.

We compared the runtime and accuracy of the proposed algorithm to those of the SpaceEx algorithms [28], while computing the safety bounds of the system, with two accuracy options that configured the set shape, *box* (box) and *octagon* (oct). The default time step was $T_s = 0.01$s, and the error tolerance was $\epsilon = 0.01$. The error in the safety bounds (err_2) in comparison to the MC results can be computed as:

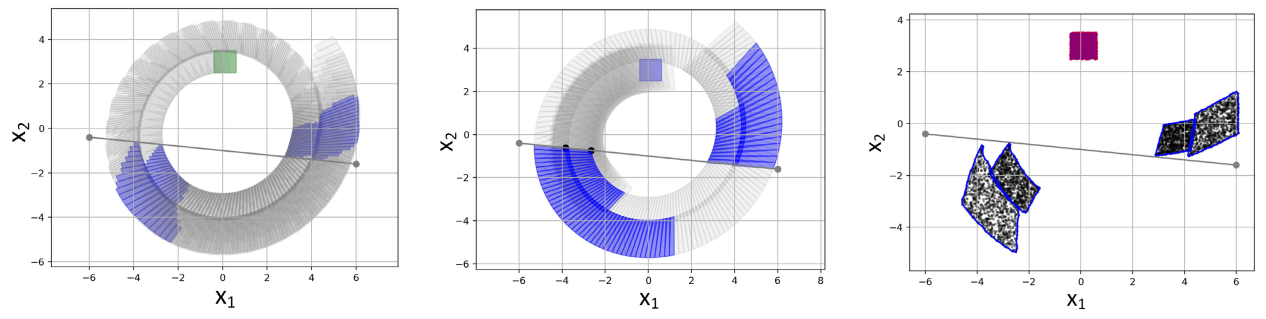
$$err_2 = \frac{1}{n}\sum_{i=1}^{n} \frac{|\hat{x}_{i,max} - x_{i,max}| + |\hat{x}_{i,min} - x_{i,min}|}{x_{i,max} - x_{i,min}} \tag{15}$$

where $[x_{i,min}, x_{i,max}]$ is the range of each state variable computed by the MC simulation and $[\hat{x}_{i,min}, \hat{x}_{i,max}]$ is the range computed with each algorithm or configuration.

Fig. 7 shows that the runtime of the proposed algorithm increases linearly with N_{GI} because it assigns a fixed N_s for each cycle of the algorithm. In computing the safety bounds for the given 12 cycles, the proposed algorithm shows a 13.8× average speed-up compared to the less accurate 'box' option, yielding an err_2 of 8.55%, and a 1074× speed-up compared to the accurate but slow 'oct' option, yielding an err_2 of 4.26% in average. Conversely, the proposed algorithm keeps the error below 1% for all the cycles. The err_2 that is a relative error metric decreases with cycles, mainly due to the increase in the safety bounds, whereas the absolute error stays approximately constant.

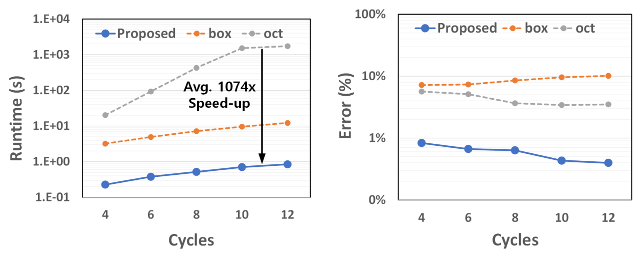
B. OPEN-LOOP DC–DC CONVERTERS

The second experiment was conducted on a DC–DC buck converter circuit, as shown in Fig. 8 (a) [19], [20]. It consists of a MOSFET switch, diode, inductor ($L = 10\mu H$), capacitor ($C = 20\mu F$), and load resistor ($R_L = 10\Omega$). It produces an output voltage V_C, a step-down voltage of the input voltage V_{IN} (=$10V$). In the case of pulse-width modulation (PWM) control, the steady-state value of V_C is proportional to the duty cycle D of the fixed-period control modes, i.e., the charging mode when the MOSFET switches on ($d(t) = q_1$) and the discharging mode when the MOSFET switches off ($d(t) = q_2$), i.e., $V_C \approx D \cdot V_{IN}$. However, the conductance of the diode varies based on the voltage across it ($V_D = -I_L R_{D,OFF}$) with $R_{D,ON} = 1\,\Omega$ for the on-state and $R_{D,OFF} = 1\,\mathrm{G}\Omega$ for the off-state. Consequently, in the discharging state, if the inductor current I_L at the instant when the MOSFET switches off is positive, the diode turns on with a negative V_D. The resistance becomes low ($R_{D,ON}$), whereas I_L stays positive. Otherwise, the diode turns off with positive terminal voltage V_D and zero inductor current $I_L = 0$. The former operation is



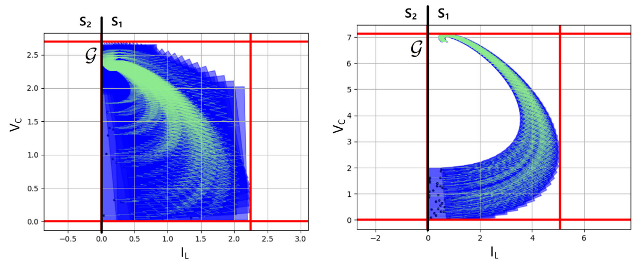
(a) Proposed **(b)** Girard's **(c)** Monte–Carlo

FIGURE 6. **The reachable sets computed for a 2-D linear hybrid system.**



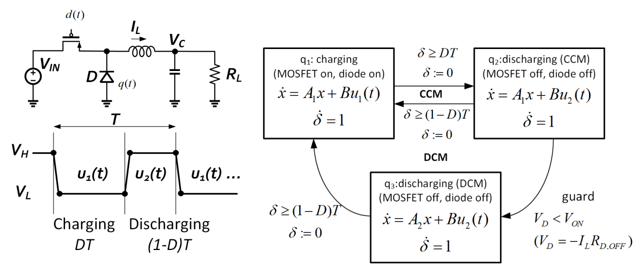
(a) Runtimes **(b)** Errors

FIGURE 7. **The runtime and error comparison (err_2) of two-dimensional hybrid system.**



(a) D=0.25 **(b)** D=0.75

FIGURE 9. **The reachable sets computed for the DC–DC buck converter circuit.**


d(t)
I_L
V_C
V_IN
D
q(t)
R_L
T
V_H
V_L
u₁(t)
u₂(t)
u₁(t) ...
Charging
DT
Discharging
(1-D)T
q₁: charging (MOSFET on, diode on)
q₂:discharging (CCM) (MOSFET off, diode off)
q₃:discharging (DCM) (MOSFET off, diode off)
CCM
DCM
guard

(a) buck converter circuit **(b)** hybrid automata model

FIGURE 8. **DC–DC buck converter (a) circuit and (b) hybrid automata.**

called the *continuous conduction mode* (CCM) and the latter the *discontinuous conduction mode* (DCM), which can be analyzed using a PWL system and the proposed reachability analysis algorithm.

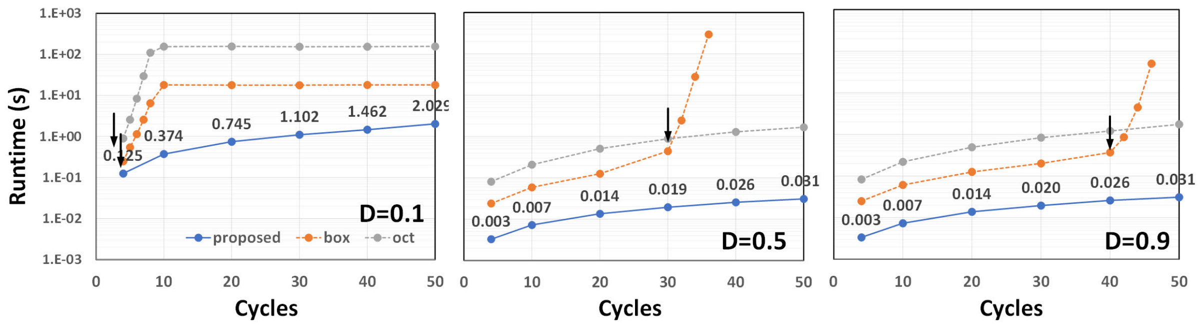
Fig. 8 (b) shows the hybrid automata model that describes the operation of the circuit in both the CCM and DCM. We define the state of the automata with two discrete state variables $q(t) \in Q$ and $d(t) \in Q_d$, where Q is the discrete state space indicating that the diode operating regions in X and Q_d is that of the external control states for the MOSFET switch state. Depending on both states, the resulting discrete states of the automata model consist of three discrete modes, q_1, q_2, and q_3. When the MOSFET is turned on, the state of the automata is given by q_1 regardless of the diode state. When the MOSFET is turned off, the state is split into two, the CCM state q_2 in which the diode is on and the circuit operates in the sub-region S_1 and the DCM state q_3 in which the diode is off and the circuit operates in the sub-region S_2, where $\{1, 2\} \in Q$. The hybrid automata model also shows the generation of $u_1(t)$ and $u_2(t)$ with a period of $T = 1\mu$s using an additional state variable δ. The system matrices for the model are

$$A_1 = \begin{pmatrix} -10^5 & -10^5 \\ 5\times 10^4 & -5\times 10^3 \end{pmatrix},\ A_2 = \begin{pmatrix} 0 & 0 \\ 0 & -5\times 10^3 \end{pmatrix},$$

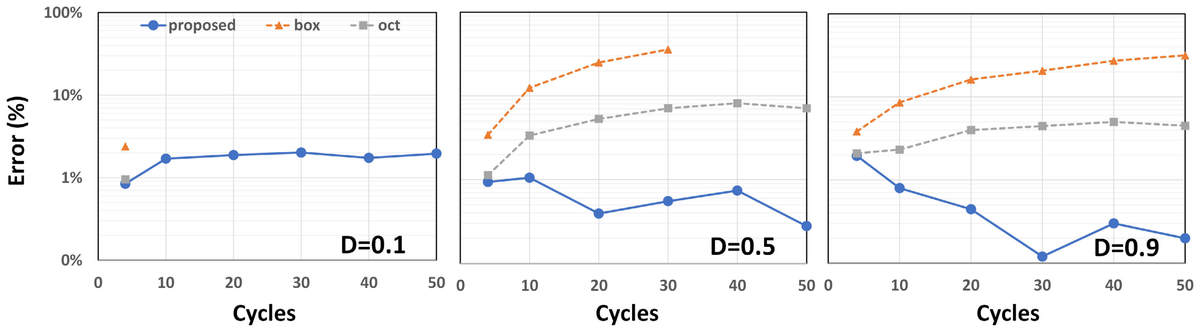
where the original values of A_2 needed to be changed to avoid numerical instability issues with SpaceEx.

Fig. 9 shows the reachable sets after 100 µs, starting with the initial state $V_C = [0,2]$ V and $I_L = [0,2]$ A. The reachable sets in blue are compared to those with a 1000-point MC simulation in green for two different duty cycles ($D = 0.25$, 0.75), resulting in the safety bounds $I_L = [0,2.25]$, $V_C = [0,2.69]$ for $D = 0.25$ and $I_L = [0,5.06]$, $V_C = [0,7.11]$ for $D = 0.75$. The analysis required runtimes of 4.967 s and 0.062 s for $D = 0.25$ and 0.75, respectively. The resulting errors err_2 were 1.97% for $D = 0.25$ and 0.11% for $D = 0.75$.

In Fig. 10, the runtime and accuracy (err_2) of the proposed algorithm in computing the safety bounds are compared with those of the SpaceEx algorithms, for $D = 0.1$, 0.5, and 0.9. For all cases, the proposed algorithm achieved the fastest runtimes of the range 0.03 s to 2.02 s and the lowest average error 0.99% (<3.48% for 'oct') of the range 0.12 % to



(a) Runtimes



(b) Errors

FIGURE 10. The runtimes and safety bound errors (err_2) of the proposed algorithm and SpaceEx algorithms.

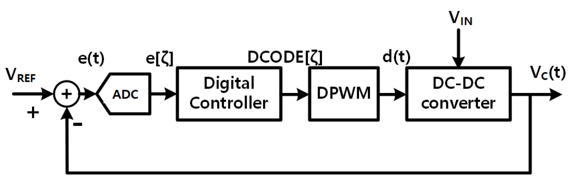


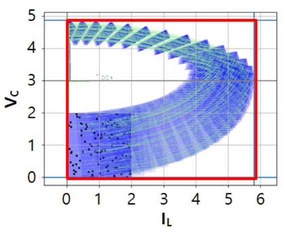
FIGURE 11. Digitally-controlled DC–DC converter.

2.02 %. The speed-up factor was 2 × to 656 × (avg. 107×) compared to the SpaceEx results for ‘oct’ and 7 × to 414 × (avg. 79×) compared to ‘box.’ The runtimes with $D = 0.1$ were longer than others because the circuit operates in DCM, and therefore requires more GI computations. Nonetheless, the runtime only scaled linearly with the number of cycles, demonstrating the effectiveness of the proposed algorithm. Note that the runtimes of SpaceEx with $D = 0.1$ stopped increasing after 10 cycles because it failed to converge within the pre-set iteration limit owing to the exponential increase in the runtime, as indicated by the arrows in Fig. 10 (a).

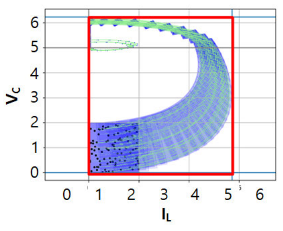
C. CLOSED-LOOP DIGITAL PWM BUCK CONVERTER

Finally, the applicability of the proposed method was demonstrated on a digitally controlled DC–DC converter [31]. This circuit differs from the previous one because of the presence of the digital feedback loop that regulates the output voltage V_c based on the reference voltage V_{REF}. Applying RA on the digital control of analog circuits leads to the accumulation of large over-approximation errors from the GI whenever the digital controller compares the circuit states $x = (I_L(t), V_C(t))$, represented by $\mathcal{R}(t)$ at every period of the digital clock $t = T_\zeta$. The duty cycle D of the control switch pulse $d(t)$ is digitally controlled by the polarity of the error level $e(t) = V_C(t) - V_{REF}$ (i.e., distance from the guard defined by $V_C(t) = V_{REF}$ in the state space), referred to as bang-bang (BB) control [32]. First, the controller measures the $V_C(t)$ at the end of the switching pulse $d(t)$ for each period and compares the measured value with the desired $V_{REF}(t)$. If the measured $V_C(t)$ is larger than V_{REF}, the controller increases the digital duty value $DCODE[\zeta]$ encoding the real-valued duty-cycle $D(t)$ by +1 before the next cycle begins. Otherwise, $DCODE[\zeta]$ is decreased by −1. Here, ζ indicates the index of discrete-control periods. The default circuit parameters are the same as those used in the previous buck converter example.

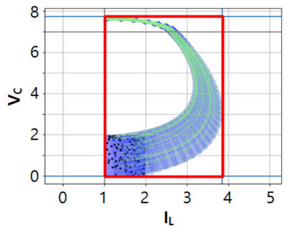
Note that most digital PWM control schemes use linear proportional-integral-derivative (PID) control [31]. In this control scheme, $e(t)$ is converted into a digital n_e-bit code $e[\zeta]$, yielding a duty control code $DCODE[\zeta]$ depending on the intended transfer function. However, implementing them into RA would result in a large number of branches whenever the controller computes the next $DCODE[\zeta]$ from $e(t)$. This would lead to an exponential increase in the runtime with respect to the number of cycles $|\zeta|$, i.e., $O(|\zeta|) = N_e^{|\zeta|}$, where



(a) $V_{REF} = 3V$

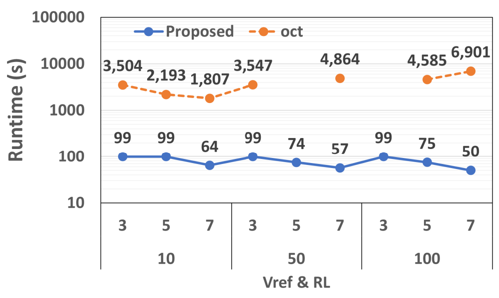


(b) $V_{REF} = 5V$

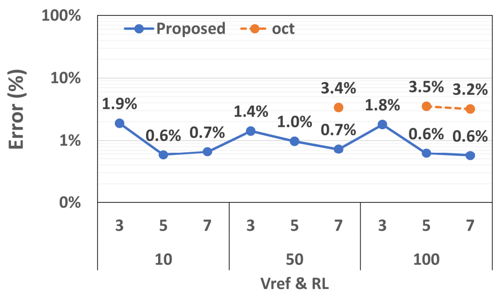


(c) $V_{REF} = 7V$

FIGURE 12. **Reachable sets with digital PWM control varying V_{REF}.**



(a) Runtime



(b) Error

FIGURE 13. **The runtimes and accuracy measured at the DPWM controlled DC–DC converter circuit varying circuit parameters V_{REF} and R_L. The runtimes of the SpaceEx increase exponentially after the onset of the GI at the boundary of the DCM and CCM sub-regions.**

the exponent N_e denotes the number of discrete levels for $e(t)$, i.e., $N_e = 2^{n_e}$. Resolving this issue is critical but beyond the scope of this study. Instead, we modeled a BB controlled converter, in which the bound of the runtime increases slightly by $O(\zeta) = 2^{\zeta}$ by encoding $e(t)$ with a single bit ($N_e = 2^1$). Therefore, RA generates fewer branches in each cycle.

TABLE 1. **Definition of math symbols.**

Symbol	Definition
n	Number of state dimensions
n_i, n_o	Number of input/output dimensions
t	Continuous time variable
c, g_h	Center point/generator of a zonotope
r	Number of generators in a zonotope
α_h	Coefficient in zonotope representation
$\mathcal{I}^c_{\bar{i},\bar{j},\bar{k}}$	GI range for the corresponding indices
$l_{\bar{k}}$	Direction vector spanning $\mathcal{G}$
D	Set of direction vector $l_{\bar{k}}$
$\mathcal{I}^c$	Set of $\mathcal{I}^c_{\bar{i},\bar{j},\bar{k}}$
$t_s^{\bar{j}}$	Discrete time steps spanning the intersecting time range $[t_1, t_2]$ for $\bar{j}$
N_s	Number of time steps discretizing $[t_1, t_2]$
m, M	Minimum and maximum values of the range of values
i, j	Subregion index for before- and after-GI
$\bar{i},\bar{j}$	Set index for before- and after GI
$\bar{k}$	Index for direction to project zonotope
$\mathcal{Z}_s^{\bar{i},\bar{j}}$	Zonotope sampled from $\mathcal{R}^{\bar{i}}(t)$ at time $t_s^{\bar{j}}$
w	Projection vector
$\mathcal{Z}_{(w,l_{\bar{k}})}$	Projected zonotope in the $(w, l_{\bar{k}})$-space
t'	Time variable after GI in the new subregion
$\mathcal{R}_{\mathcal{G}}(t')$	Reachable set of $\mathcal{G}$ at time t'
$t_r^{\bar{j}}$	Remaining time period until $t = t_2$ in the new subregion S_j
$\mathcal{Z}_{\mathcal{G}0}$	Zonotope spanning the entire hyperplane of $\mathcal{G}$
$\mathcal{Z}_{\mathcal{G}}^{\bar{j}}$	Zonotope sampled from $\mathcal{R}_{\mathcal{G}}(t)$ at time $t_r^{\bar{j}}$
$\mathcal{R}_{\mathcal{G}}(t)$	Reachable set of the guard hyperplane $\mathcal{G}$ at time t
$\mathcal{I}^c_{\bar{i},\bar{j},\bar{k}}$	Ranges of GI the $\bar{i}$-th zonotope for the $\bar{j}$-th time interval and the $\bar{k}$-th dimension
$\mathcal{Z}_c^{\bar{j}}$	Cross-sectional zonotope for the $\bar{j}$-th time interval
$\mathcal{Z}_0^{\bar{j}}$	Initial zonotope for the $\bar{j}$-th time interval
$\mathcal{R}^{\bar{j}}(t)$	Reachable set of the system in the $\bar{j}$-th time segment for $t \in [t^s\bar{j} - 1, t^s\bar{j}]$, represented as a zonotope

Fig. 12 shows the computed $\mathcal{R}^{col}(t)$ obtained by varying the $V_{REF} = 3\,\text{V}$, $5\,\text{V}$ and $7\,\text{V}$ in the state space of the $x(t) = (I_L(t), V_C(t))$, compared to the equivalent MC simulation trajectories. The state space is divided into two sub-regions depending on the level of $V_C(t)$ with respect to V_{REF}. The computed $R^{col}(t)$ originating from the initial set demonstrates the expected convergence behavior with respect to V_{REF} on varying D depending on the operating sub-region. Each reachable set accurately encloses the state trajectories from the initial randomly selected samples in the initial set that ranges $I_L = [0, 2]$A and $V_C = [0, 2]$V while converging to each of the different V_{REF}.

Fig. 13 shows the runtime of the proposed method and err_2 in (15) compared to the MC simulation results. The improvements in speed compared to the SpaceEx algorithms with octagonal shapes (oct) are shown in Fig. 13 (b). The speed-up ranges from 61–138× (avg. 40 ×), whereas the error is maintained below 2 % compared to that of the MC simulation references. The minimum speed-up occurs for $R_L = 100\,\Omega$ while the maximum speed-ups occur for $V_{REF} = 5\,\text{V}$ and $V_{REF} = 7\,\text{V}$. The resulting runtime ranges from 50–99 seconds, which is considerably longer than that of the previous open-loop example. This is because the computation of the reachable set of the closed-loop feedback system consistently generates new branches of reachable sets, thereby exponentially increasing the required runtime for the completion of the predefined number of cycles (i.e., equivalent to the time bound) similar to conventional methods. The proposed method not only outperformed previous methods in terms of runtimes but also achieved the lowest error in all cases. Note that Fig. 13 (b) has some missing data for SpaceEx when it could not complete the analysis due to timeout failures.

V. CONCLUSION

This study presented a new algorithm for performing RA on nonlinear AMS circuits modeled as a PWL system. The algorithm accurately expressed the time evolution of continuous states with sets of trajectory functions in the PWL system without increasing the number of sets by proposing a new scalable GI computation method. When verifying the safety bounds of a DC–DC converter, the proposed algorithm demonstrated $79\times$ to $107\times$ average speed-ups compared to STC algorithms in SpaceEx while maintaining the safety-bound estimation error within 2% for 50-cycle iterations. To the best of our knowledge, this is the first study to address the problem of set increases with respect to GI, and it suggests a way to compute it in a linear time with good accuracy. While this work demonstrated the effectiveness of the proposed method with ideal circuits, we believe it can be further extended to include noise and process variation effects as well.

APPENDIX A
MATH SYMBOL DEFINITION

Table 1 lists the definitions of the mathematical symbols used in this study.

REFERENCES

[1] G. V. Bochmann, "Finite state description of communication protocols," *Comput. Netw.*, vol. 2, nos. 4–5, pp. 361–372, Sep. 1978.

[2] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, "The algorithmic analysis of hybrid systems," *Theor. Comput. Sci.*, vol. 138, no. 1, pp. 3–34, Feb. 1995.

[3] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi, "HYTECH: A model checker for hybrid systems," *Int. J. Softw. Tools Technol. Transf.*, vol. 1, nos. 1–2, pp. 110–122, Dec. 1997.

[4] T. A. Henzinger, "The theory of hybrid automata," in *Verification of Digital and Hybrid Systems*. Berlin, Germany: Springer, Jul. 2000, pp. 265–292, doi: 10.1007/978-3-642-59615-5_13.

[5] J. Glanz, "Mathematical logic flushes out the bugs in chip designs," *Science*, vol. 267, no. 5196, pp. 332–333, Jan. 1995.

[6] A. P. L. Ferreira, "Model checking," in *Proc. Workshop-School Theor. Comput. Sci.*, Aug. 2011, pp. 9–14.

[7] R. P. Kurshan and K. L. Mcmillan, "Analysis of digital circuits through symbolic reduction," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 10, no. 11, pp. 1356–1371, Nov. 1991.

[8] P. Herber and S. Glesner, "Verification of embedded real-time systems," in *Formal Modeling and Verification of Cyber-Physical Systems*. Wiesbaden, Germany: Springer, 2015, pp. 1–25, doi: 10.1007/978-3-658-09994-7_1.

[9] P. Koopman, "A case study of Toyota unintended acceleration and software safety," Univ. California, Berkeley, Berkeley, CA, USA, Nov. 14, 2014. [Online]. Available: http://chess.eecs.berkeley.edu/pubs/1081.html

[10] M. Althoff and J. M. Dolan, "Online verification of automated road vehicles using reachability analysis," *IEEE Trans. Robot.*, vol. 30, no. 4, pp. 903–918, Aug. 2014.

[11] H. Ahn and D. D. Vecchio, "Safety verification and control for collision avoidance at road intersections," *IEEE Trans. Autom. Control*, vol. 63, no. 3, pp. 630–642, Mar. 2018.

[12] C. J. Tomlin, I. Mitchell, A. M. Bayen, and M. Oishi, "Computational techniques for the verification of hybrid systems," *Proc. IEEE*, vol. 91, no. 7, pp. 986–1001, Jul. 2003.

[13] S. Gupta, B. H. Krogh, and R. A. Rutenbar, "Towards formal verification of analog designs," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design*, Nov. 2004, pp. 210–217.

[14] W. Denman, B. Akbarpour, S. Tahar, M. H. Zaki, and L. C. Paulson, "Formal verification of analog designs using MetiTarski," in *Proc. Formal Methods Comput.-Aided Design*, Nov. 2009, pp. 93–100.

[15] C. Yan and M. R. Greenstreet, "Circuit level verification of a high-speed toggle," in *Proc. Formal Methods Comput.-Aided Design*, Austin, TX, USA, Nov. 2007, pp. 199–206. [Online]. Available: http://ieeexplore.ieee.org/document/4402001/

[16] Y. Song, H. Yu, and S. M. P. DinakarRao, "Reachability-based robustness verification and optimization of SRAM dynamic stability under process variations," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 4, pp. 585–598, Apr. 2014.

[17] M. Althoff, S. Yaldiz, A. Rajhans, X. Li, B. H. Krogh, and L. Pileggi, "Formal verification of phase-locked loops using reachability analysis and continuization," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2011, pp. 659–666.

[18] H. Lin, P. Li, and C. J. Myers, "Verification of digitally-intensive analog circuits via kernel ridge regression and hybrid reachability analysis," in *Proc. 50th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, May 2013, pp. 1–6.

[19] E. M. Hope, X. Jiang, and A. D. Dominguez-Garcia, "A reachability-based method for large-signal behavior verification of DC–DC converters," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, no. 12, pp. 2944–2955, Dec. 2011.

[20] O. A. Beg, H. Abbas, T. T. Johnson, and A. Davoudi, "Model validation of PWM DC–DC converters," *IEEE Trans. Ind. Electron.*, vol. 64, no. 9, pp. 7049–7059, Sep. 2017.

[21] E. Sontag, "Nonlinear regulation: The piecewise linear approach," *IEEE Trans. Autom. Control*, vol. AC-26, no. 2, pp. 346–358, Apr. 1981.

[22] E. Asarin, T. Dang, and O. Maler, "d/dt: A verification tool for hybrid systems," in *Proc. 40th IEEE Conf. Decis. Control*, Dec. 2001, pp. 2893–2898.

[23] A. A. Kurzhanskiy and P. Varaiya, "Ellipsoidal techniques for reachability analysis of discrete-time linear systems," *IEEE Trans. Autom. Control*, vol. 52, no. 1, pp. 26–38, Jan. 2007.

[24] A. Girard, "Reachability of uncertain linear systems using zonotopes," in *Proc. ACM Int. Conf. Hybrid Syst. Comput. Control*, Apr. 2005, pp. 291–305.

[25] C. Le Guernic and A. Girard, "Reachability analysis of hybrid systems using support functions," in *Proc. Int. Conf. Comput. Aided Verification*, Jun. 2009, pp. 540–554.

[26] S. Kim, H. Park, and J. Kim, "Safety verification of AMS circuits with piecewise-linear system reachability analysis," in *Proc. 18th Int. SoC Design Conf. (ISOCC)*, Oct. 2021, pp. 203–206.

[27] G. Frehse and R. Ray, "Flowpipe-guard intersection for reachability computations with support functions," in *Proc. IFAC Conf. Anal. Design Hybrid Syst.*, Jun. 2012, pp. 94–101.

[28] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, "SpaceEx: Scalable verification of hybrid systems," in *Proc. Int. Conf. Comput. Aided Verification*, Jul. 2011, pp. 379–395.

[29] A. Girard and C. Le Guernic, "Zonotope/hyperplane intersection for hybrid systems reachability analysis," in *Proc. Int. Workshop Hybrid Syst. Comput. Control*, Apr. 2008, pp. 215–228.

[30] J. E. Jang, M. Park, and J. Kim, "An event-driven simulation methodology for integrated switching power supplies in SystemVerilog," in *Proc. 50th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, May 2013, pp. 1–7.

[31] M. Lee, Y. Choi, and J. Kim, "A 500-MHz, 0.76-W/mm power density and 76.2% power efficiency, fully integrated digital buck converter in 65-nm CMOS," *IEEE Trans. Ind. Appl.*, vol. 52, no. 4, pp. 3315–3323, Jul. 2016.

[32] M. Park and J. Kim, "Pseudo-linear analysis of bang-bang controlled timing circuits," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 6, pp. 1381–1394, Jun. 2013.

SEYOUNG KIM received the B.S. and M.S. degrees in electrical engineering from Seoul National University, Seoul, South Korea, where he is currently pursuing the Ph.D. degree in electrical engineering. In 2005, he joined Samsung Electronics and performed research on memory design methodologies and computer-aided engineering. His current research interests include design and verification methodologies for memory and analog/mixed-signal circuits.

JAEHA KIM (Senior Member, IEEE) received the B.S. degree in electrical engineering from Seoul National University, Seoul, South Korea, in 1997, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, USA, in 1999 and 2003, respectively. From 2001 to 2003, he was a Circuit Designer with True Circuits Inc., Los Altos, CA, USA. From 2003 to 2006, he was a Postdoctoral Researcher with the Inter-University Semiconductor Research Center (ISRC), Seoul National University. From 2006 to 2009, he was a Principal Engineer with Rambus Inc., Los Altos. From 2009 to 2010, he was an Acting Assistant Professor with Stanford University. In 2010, he joined Seoul National University, where he is currently a Professor. In 2015, he founded Scientific Analog Inc., Palo Alto, CA, USA, an EDA company focusing on analog/mixed-signal verification. His research interests include low-power mixed-signal systems and their design methodologies. He served on the technical program committees for the International Solid-State Circuits Conference (ISSCC), the Custom Integrated Circuits Conference (CICC), the Design Automation Conference (DAC), the International Conference on Computer-Aided Design (ICCAD), and the Asian Solid-State Circuit Conference (ASSCC). He was a recipient of the Takuo Sugano Award for Outstanding Far-East Paper from the 2005 ISSCC. He was cited as the Top 100 Technology Leader of Korea by the National Academy of Engineering of Korea (NAEK), in 2020.