Ph.D. DISSERTATION

Safety Verification of Analog Mixed-Signal Circuits Using Reachability Analysis

도달가능성 알고리즘을 이용한 아날로그 혼성회로 시스템의 정형 안전 검증 기법

BY

Kim Seyoung

AUGUST 2023

DEPARTMENT OF ELECTRICAL ENGINEERING AND
COMPUTER SCIENCE
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Ph.D. DISSERTATION

Safety Verification of Analog Mixed-Signal Circuits Using Reachability Analysis

도달가능성 알고리즘을 이용한 아날로그 혼성회로 시스템의 정형 안전 검증 기법

BY

Kim Seyoung

AUGUST 2023

DEPARTMENT OF ELECTRICAL ENGINEERING AND
COMPUTER SCIENCE
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Safety Verification of Analog Mixed-Signal Circuits Using Reachability Analysis

도달가능성 알고리즘을 이용한 아날로그 혼성회로 시스템의 정형 안전 검증 기법

지도교수 김 재 하

이 논문을 공학박사 학위논문으로 제출함

2023년 8월

서울대학교 대학원

전기 · 정보 공학부

김 세 영

김세영의 공학박사 학위 논문을 인준함

2023년 8월

위 원 장: _____
부위원장: _____
위　　원: _____
위　　원: _____
위　　원: _____

Abstract

This dissertation proposes a methodology to verify the safe operation of analog mixed-signal (AMS) circuits under formal specifications. With integrating many AMS integrated circuits (IC) in a car, assuring the safety of the system is getting a critical issue for developing such safety-critical systems. While the operating ranges of analog circuits and their semiconductor devices such as MOSFETs and diodes should be limited under safe-operating area (SOA) specified in the process design kit (PDK) or datasheets, the existing verification procedures relying on SPICE simulations in industry lack of capability to cover all possible dynamic behavior that causes the change of the applied terminal voltages or branch current of the devices. The formal verification algorithm, especially reachability analysis (RA), can prove that the system is safe by computing all reachable states from the specified initial conditions and finding if the reachable set of states intersects with the unsafe set of states causing failures of devices or circuits. However, computing the reachable set is burdensome for practical circuits since it takes too long to compute reachable sets in a high dimensional system dynamically operating in a high frequency, due to the continuous nature of the states that require quantization to many discrete steps of continuous value. This dissertation addresses the challenges while applying reachable analysis to AMS circuits and proposes an efficient way to compute reachable sets.

The proposed RA algorithm can compute the reachable set of analog mixed-signal circuits in a fast and accurate way by proposing a novel trajectory form of zonotope, combining two main ideas using scalable geometric set representation, such as zonotope and the XMODEL algorithm that computes analog signals in a trajectory form that represents the signal in a closed functional form with few algebraic coefficients. Any analog circuits, even those containing nonlinear components can be analyzed using the efficient piecewise-linear (PWL) approximation technique that only parti-

tions the state space of the circuit depending on the operating modes of the included devices. The resulting PWL system in the partitioned state space with several sub-regions is more difficult to be computed with conventional RA algorithms relying on time discretization since it calls for many additional computationally expensive geometric intersection operations, resulting in exponentially increasing runtimes with the number of transitions over sub-regions. Applying the proposed trajectory form is not straightforward because the intersection between the trajectory form and the switching boundary can not maintain the same form, requiring additional segmentation of the set shapes to keep the original form as before. Recognizing that every state after the intersection starts from the boundary plane, by computing the time evolution of the boundary itself, it has been shown that the associated computation can be done with the proposed trajectory form without increasing runtime.

The proposed method was demonstrated by verifying the exemplary DC-DC switching converters, yielding over 107x speed-up than the existing reachability analysis algorithm in SpaceEx. The computational complexity for state dimensions was reduced by two orders than SpaceEx without losing accuracy, compared to the bound obtained equivalent Monte-Carlo simulations. Finally, SOA for DC-DC converters varying circuit parameters was derived and compared to typical 90-nm process criteria, showing the proposed method usable for verifying SOA specification of AMS circuits.

keywords: Reachability analysis, formal verification, safety verification, analog mixed-signal circuits, DC-DC converters, Laplace s-domain analysis, XMODEL
student number: 2018-36370

Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

1.1 Background and Challenges

With emerging autonomous systems such as electrical vehicles, drones, and robots, verifying such safety-critical systems becomes a significant issue today, since failures or malfunction of these systems can lead to severe economical loss or environmental damage and even risks to human lives. For example, Toyota's recent brake problems were known as software defects due to incomplete verification and Toyota's market capitalization of $25 billion was lost in the two months with the recalls in January 2010 [1]. The famous floating-point division (FDIV) bug in the Pentium IV processor in the 90s cost $475M charge for Intel to cover the replacement of every defective CPU [2]. As, however, the number of its parameters that interacts with each block in the system increases with the growing design complexity, randomly generated test benches mostly failed to detect the rare bug hidden in the deep state of complex designs, which is like hitting the center of the target with a dart [3]; therefore, formal verification is necessary and indispensable for state-of-the-art complex mixed-signal integrated circuits (IC) and system-on-chips (SoC).

In particular, semiconductor devices such as MOSFETs and diodes, and other metal interconnection layers are vulnerable to exposure to high current or voltage.

Therefore, analog circuit designers generally ensure reliable operation of their AMS circuits according to the *safe-operating area* (SOA) specification for all devices and circuits [4], which is specified in process design kits (PDK) that describes the behaviors and requirements of the design components including active elements such as MOSFETs and passive interconnecting elements such as metal layers or resistors of the corresponding manufacturing process used. Circuit designers should verify that the devices used in their circuits should operate within the allowed regions defined in terms of voltages and currents. However, the conventional design methodology for analog circuits relies on limited simulation-based methodologies that only simulate parts of test cases based on heuristics, leading to incomplete verification and most of them can cause malfunction of the entire system such as car accidents. In addition, due to the scaling of the process, the design window for SOA is getting smaller than before and the complex interaction between analog and digital blocks also becomes more threats of unexpected behavior, increasing the likelihood of such critical design errors.

Formal verification can prove the correctness of the systems with given specifications considering every possible state of the system with mathematical algorithms [5], while simulations never guarantee. Numerous formal verification solutions have been developed for verifying complicated control algorithms for software and hardware after the success story of Pentium bug hunting [2] with the development of algorithmic logical reasoning such as *model checking* [5] and *theorem proving* methodologies [6]; however, compared to the rich achievements for fore-mentioned areas, a way towards formally verifying analog mixed-signal (AMS) circuits is still far from the sight.

The main difficulty comes from the fact that the analog circuits operate with continuous state values represented in the state space with continuous time. For example, let's assume that one needs to verify the function of two circuits given shown in Fig. 1.1 [7]. While the possible states of the logical AND gate with two inputs and single output have only four cases, verifying an analog op-amp circuit requires exploring an infinite

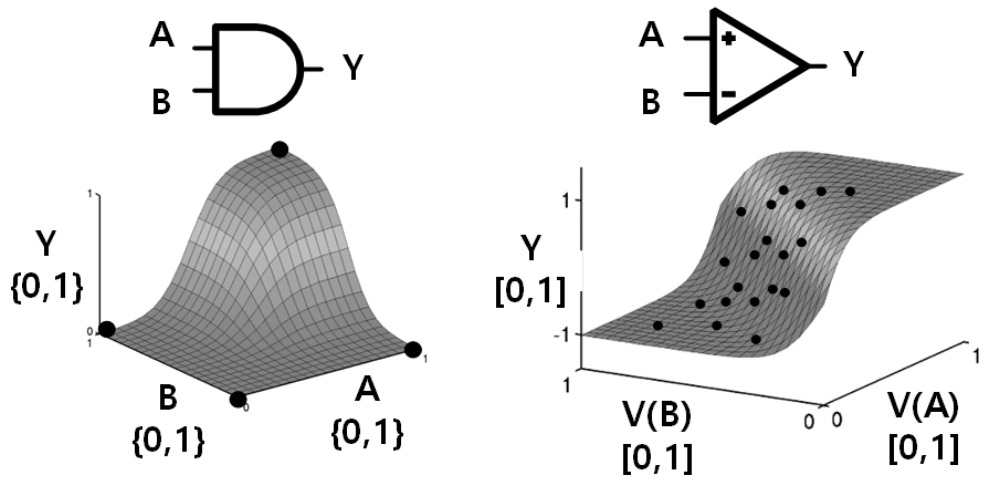


Figure 1.1: Analog verification challenge vs. digital. (a) Logical AND gate and (b) analog OP amp. Analog formal verification requires additional quantization, resulting in the dimensional complexity problem.

number of cases for exhaustively exploring possible states due to the continuous nature of the input and state variables. This requires quantization of a set of continuous states, leading to additional dimensional complexity problems that require exponentially increasing complexity with respect to the number of state dimensions at least. In addition, as one uses more accurate shapes for reducing the over-approximation error, the computational time will be further increased. This work addresses a way to verify the safety of AMS circuits by efficiently modeling AMS circuits and computing their states fast and accurately with tacking the time complexity problem from the quantization of continuous value.

1.1.1 Safety Verification of AMS Circuits Using Reachability Analysis

We focus on verifying the safety of AMS circuits in this dissertation. In practice, safety verification of analog circuits involves two steps: i) **Formal abstraction**: abstraction of circuits to explore reachable states such as *hybrid system* and ii) **Reachability anal-**

ysis: computing the evolution of continuous states with time t by computing the set containing the states and their time evolution; therefore, it requires computationally efficient set structure with less approximation errors.

1.1.2 Hybrid System

We model AMS circuits as hybrid systems in this study. A hybrid system can capture every possible dynamic behavior of AMS circuits, which is a system combining the behavior of both discrete variables $q \in Q$ and continuous variable $x \in X$, where we can apply formal analysis techniques. Its model is represented by hybrid automata

$$M = \langle X, Q, E, F, \mathcal{G} \rangle \tag{1.1}$$

which can be considered as a graph structure, where each node corresponds to a continuous system associating its differential equations F, a set of discrete transitions, the guard $\mathcal{G}$ that conditions for the variable that triggers the discrete transition in E (or jump in the different literature) and assignment associated with the transition (or reset). Each differential equation F can be linear or nonlinear; when every differential equation F is linear, then the automata M is defined by linear hybrid automata, which we address how to efficiently transform the nonlinear circuits to this class of automata that we mainly use through the dissertation. A simple example of hybrid automata is illustrated in Fig. 1.2. The automata has two discrete states, also referred to as a location. The transition between the locations in the automata is determined by the value of continuous variable t. When the variable t becomes greater than T, it switches its location and the associated differential equation simultaneously.

1.1.3 Reachability Analysis

The main topic of this dissertation is reachability analysis [8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 6, 21, 22, 23, 24, 5], which is one of the most prominent formal algorithms for checking the model that can prove the safety property of the system;

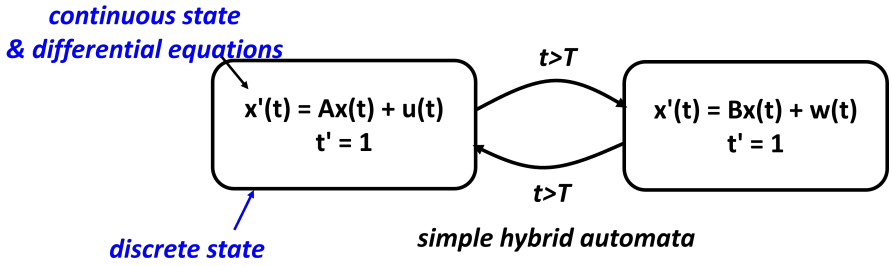


Figure 1.2: Simple hybrid automata example with two discrete states.

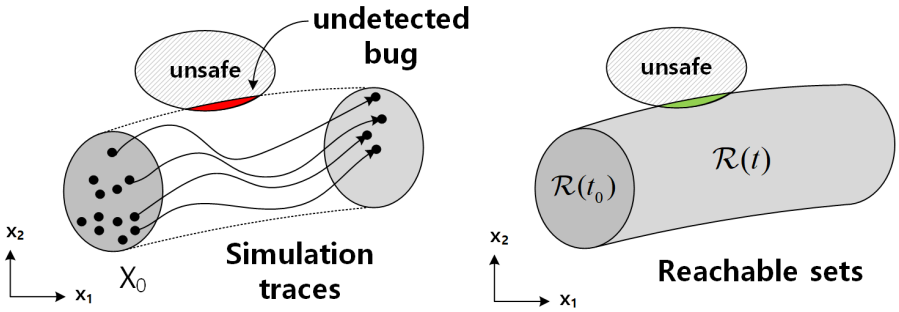


Figure 1.3: Difference between (a) test-bench simulations and (b) reachability analysis. Simulation may not capture hidden bugs while reachability analysis exhaustively finds possible set of states from the initial set of states.

it is originally developed for verifying complex communication protocols in 70s and expanding its application area even for continuous systems. While original automata can only consider the case that its continuous variable and their derivative is limited as constant such as $\dot{x} = 1$, so-called timed automata, the system we focus on in this dissertation is the linear hybrid system mentioned above. The main difference is that the derivative of the continuous variable has the linear equation of the variable. Same to the general cases, hybrid reachability analysis verifies safety by finding all hybrid states that can be reached from a specified range of initial states and checking if the reachable states are within a target range.

Safety verification using reachability analysis is a formal procedure that finds every

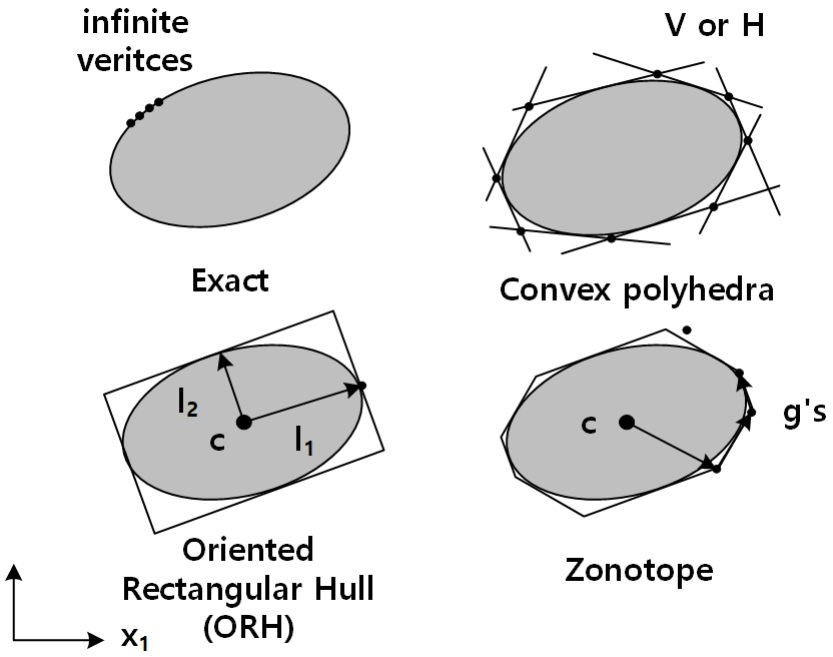


Figure 1.4: Over-approximation depending on geometrical set representations.

reachable state from the initial states X_0 and then collecting the resulting states yields the set of reachable states $\mathcal{R}$, commonly referred as *reachable set*, and this procedure is called *reachability analysis*. We can conclude the system is safe if the reachable set $\mathcal{R}$ does not contain any unsafe states defined in unsafe set $\mathcal{U}$.

1.1.4 Main Challenges

The main challenge in computing the reachable sets for an AMS circuit is three-fold.

Space discretization problem First, as shown in Fig. 1.4, computing a set of analog states in the state space introduces a geometrical shape that encloses every possible state and is represented by a set of vectors representing the vertices or faces. However, exact representation for an arbitrary region of reachable sets $\mathcal{R}$ is impossible because it requires an infinite number of vertices or faces; thus, one needs to conservatively

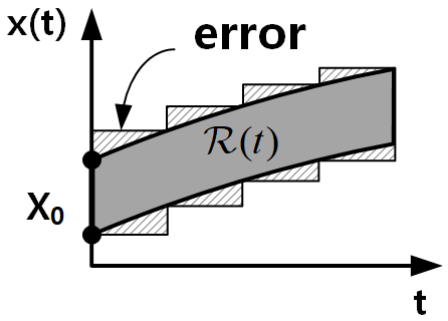


Figure 1.5: Over-approximation depending on time steps.

approximate the exact one with efficient geometrical shapes with fewer vertices and should balance the *trade-off* between the accuracy and the computational efficiency of the geometry. As shown in the Fig. 1.4, an oriented rectangular set (ORH) using fewer vectors causes a large over-approximation error due to its coarse shape, while the complex and fine shape using more vectors can reduce the resulting over-approximation error but increase the runtime of the analysis. After decades of efforts for finding computationally efficient geometrical set representations, several set representations using linear algebra are gaining popularity for this purpose, such as oriented rectangular hull [25], ellipsoid [18], convex polyhedra [16] and zonotopes [14]. These set representations are very efficient for computing the reachable set in linear systems even with hundreds of state variables, but have limitations in that they can only be applied to linear systems.

Time discretization problem Secondly, to compute the change of a set of states for a given time period, conventional reachability algorithms first segment the analysis time duration into several discrete steps of time and then compute the reachable sets by iteratively multiplying the system matrix with the sets at each time steps, incurring a large computational cost. Furthermore, mixed with the first problem concerning space discretization, it needs to split the period with more fine time steps if it fails to

accurately represent the trajectory of the set with the selected set representation with given error tolerance. However, there have been few approaches to address this issue yet.

nonlinearity Lastly, typical analog circuits are nonlinear, while those methods described above are applied only for linear systems; thus, they need to be linearized into several disjoint sub-regions. However, the computational cost for the transition between each region grows fast with the number of time steps to compute the time evolution of the reachable sets.

The most simple and efficient approach is converting them into a piecewise linear system that operates in different differential equations in each region segmented by specified switching conditions [26, 27]. However, we found that conventional reachability tools suffer from computing the transition at the switching boundary, which is commonly referred to as a guard intersection and is a computationally expensive geometrical intersection operation. *guard* set is equivalent to the set enclosing the switching condition, and their intersection should be computed in other sub-region of the PWL system with its own set of differential equations. While sliding the reachable set with time over the boundary, it intersects with the boundary and generates infinite sets of intersections. With the conventional method with time discretization reachable set computation, the analysis incurs a tremendous number of computations proportional to $N_t \times N_t$ where N_t is the number of steps resulting from time discretization. Even worse, every increasing number of states to be computed is multiplied by each other, generating an exponential increase with respect to the number of discrete transitions over the guard.

1.2 Main Contribution

This work proposes a methodology for verifying the safe operation of AMS circuits using a fast trajectory-form reachability analysis algorithm. The key idea is to reduce

the number of evaluations at the discrete time steps to represent the trajectory of the reachable sets $\mathcal{R}$ as a sequence of sets $\{\mathcal{R}_\prime, ..., \mathcal{R}_{N_t}\}$, where the number of time steps by discretizing the period of continuous time $[0, T]$. Instead, in this work, the reachable set is represented by a closed and exact form of the function $\mathcal{R}(t)$, where each vertex follows a trajectory described by an exact, analytical function of time t.

This approach has mainly two advantages. At first, it can compute the exact trajectory from the initial set at once without discrete evaluation at discrete time steps of t similar to the numerical integration method in the simulation algorithm. This adopts the idea of [28] representing analog signals as s-domain signals using the coefficients representing the poles, zeros, and gains of the signal. In this way, it outperformed typical numerical solvers when solving AMS circuit behaviors, demonstrating wide ranges of AMS circuits and systems such as switching power supplies, decision feedback equalizers (DFE), phase-locked loops (PLL) and etc. Therefore, it efficiently avoids the concern of trade-offs scarifying the resulting accuracy to get better performance. One can represent the exact trajectory $\mathcal{R}(t)$ starting from the initial set X_0 as shown in Fig. 1.5.

Second, this idea can further improve the scalability especially when computing the reachable set of hybrid systems, which increases the number of computations proportional to $N_t \times N_t$. This limitation prohibits most of the existing reachability tools fails to evaluating the practical nonlinear circuits even for simple and trivial circuits. Using the proposed trajectory-form reachability analysis, one can accurately compute the guard intersection $\mathcal{R}(t) \cap \mathcal{G}$ without generating additional time steps to get an accurate intersecting range or region. The subsequent reachable set from the guard intersection also can be efficiently evaluated.

The dissertation demonstrates that the proposed method can verify the safe operation of practical switching power supply circuits: buck converters and various switching pulse control schemes. Firstly, we demonstrate the basic buck converter circuits operating in continuous conduction mode (CCM) changing their behavior depending

on the duty cycle of the input pulse applied to its transistor switch to show the computational scalability for simple hybrid systems. Secondly, we demonstrate a more general buck converter exhibiting discontinuous conduction mode (DCM) operation due to the non-linear characteristic of the diode, showing the improved performance when computing the continuous guard intersection $\mathcal{R}(t) \cap \mathcal{G}$ occurs. The term *continuous* means that the intersections occur in the continuous time period $[t_1, t_2]$ not at the specific time instant.

In the experiment, resulting accuracy was estimated compared to the equivalent Monte-Carlo simulations, and the performance is compared to one of the reachability analysis tools, SpaceEx [17], which implements the convex polyhedra set-based reachability analysis tool using the support function and the other advanced algorithms. It serves as a golden reference for evaluating the performance of the new reachability analysis algorithm owing to its flexibility.

1.3 Thesis Organization

Chapter 2 explains the concepts of formal verification of circuits. Chapter 3 describes the proposed trajectory-form reachability analysis in detail with mathematical definition and derivation. Chapter 4 presents the proposed reachability analysis algorithm for non-linear circuits. Chapter 5 demonstrates the proposed methodology with practical circuit examples for verifying their safe operations. Chapter 6 draws the conclusion of this thesis.

Chapter 2

Formal Safety Verification of AMS circuits

This chapter explains the concepts of formal safety verification of AMS circuits. With the growing complexity of hardware design having millions of lines of code, human verification reached its limitation, creating tons of trivial bugs and imposing critical system risks. Consequently, formal verification gains attention for decades from theoretical to practical views. Its category can be classified by its intent to verify: equivalence checking, model checking, and theorem proving. At the beginning of the 70s, there were approaches using logical inductive reasoning like human proof with pencil-and-paper, which we call theorem proving. This is ideal proof, but hard for a non-expert to use. Instead, with the recent great improvement of computing capability with massively parallel computing, more practical methods gain more attention, so-called model checking. In this chapter, we explain how we can guarantee the safe operation of AMS circuits using formal model-checking techniques, and reachability analysis.

2.1 Overview of Model Checking

Model Checking is a general concept of a computer-assisted method for the analysis of dynamical systems that can be modeled by state-transition systems. Model checking is now widely used for formal verification of software and hardware in industry,

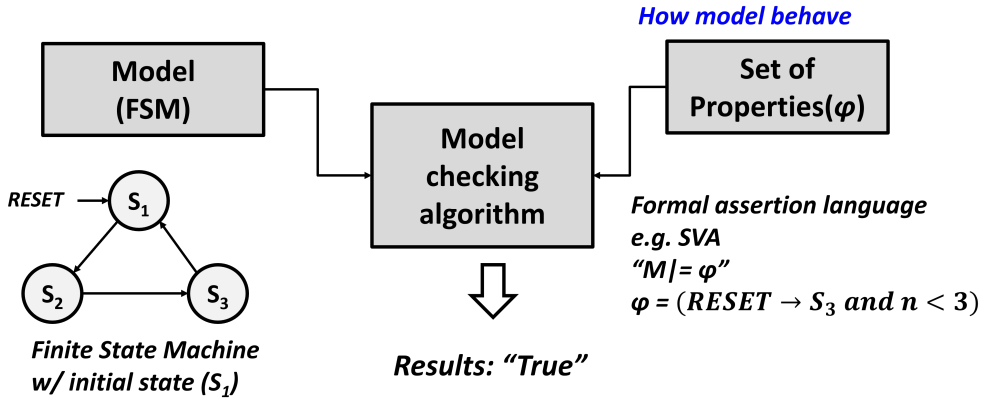


Figure 2.1: Concepts of model checking in discrete systems represented as a finite state machine (FSM).

including many mathematical algorithms with model structure abstracting given systems. Model checking consists of three main concepts: modeling, specification, and algorithms. As we stated in the definition, its model has the form of a finite-state machine. It can be applied to any dynamical system and can be represented in the form of a state transition system. Second, verifying a system requires specifying the design intent formally. It is typically given by the description of the system specification, which should be transformed into a formal representation. For example, temporal logics such as LTL and CTL, or Synopsys Verification Assertion (SVA) language are used in industry for formal verification. Finally, there is a decision procedure for determining (i.e. verifying) that the given *model* behavior always satisfies the given specification. This concept of model checking can be summarized as shown in Fig. 2.1. The procedure of model checking is to check if model M satisfies the specification φ, where the system starts from the state S_1 and reaches the state S_3 in less than three steps if not, it produces the counterexample. One can see that the result will become true because the FSM reaches S_3 in two steps.

The main challenge of model checking is the state-explosion problem due to its exhaustive nature, increasing its runtime proportionally. The resulting number of states

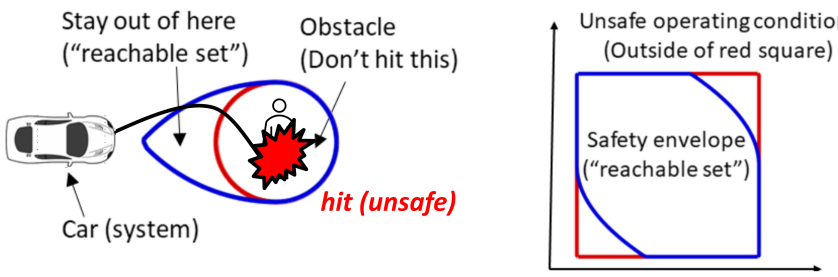


Figure 2.2: Concepts of safety verification.

to explore exponentially increase with a combinatorial number of bit size of states (e.g. memory of computer program or number of flip-flops of digital system). Therefore, computing directly the model of a given system in the real world is impossible; instead, one needs to model the system in an efficient way.

2.2 Problem Definition

In this section, we formally define the problem of verifying the safe operation of circuits and highlight the goal of this thesis.

2.2.1 Formal Definition of Safety Verification

Safety verification is widely used in verifying safety-critical hybrid dynamic systems such as robots, autonomous control of cars and drones, etc. The goal of dynamic control of these hybrid systems is not to hit critical obstacles such as humans or valuable objects. This is a reachability problem that checks if all the reachable states from any possible initial conditions after the finite duration of time reach safe or unsafe regions as shown in Fig. 2.2.

Formally, this model checking problem of any safety property can be reduced to the reachability problem of deciding reachability to a set of bad state $\mathcal{U}$. If $M \models \varphi$, then φ is called an *invariant* of model M. That is, collecting every reachable state, i.e. *reachable set* $\mathcal{R}$ is equivalent to getting invariant φ of the model M; therefore, safety

verification is a reachability problem that checks if there is any state s satisfying:

$$(s \in \mathcal{R} \subseteq \varphi) \wedge (s \in \mathcal{U}). \tag{2.1}$$

This is equivalent to finding the intersection of $\mathcal{R} \cap \mathcal{U}$ and if there is any state in $\mathcal{U}$. We can apply this concept of safety verification to verify the safe operation of circuits in the same manner in the state space of circuits.

2.2.2 Safe operation of AMS circuits

Verifying the safe operation of circuits not to be damaged by high operating voltages or current applied to devices also can be represented in the safety verification problem of (2.1). Fig. 2.3 illustrates the concepts in terms of verifying the safety of circuits. In this case, the reachable set $\mathcal{R}(t)$ can be computed by a set of the moving states $x(t)$ in the state space of circuit state variables consisting of capacitor voltages $V_C(t)$ and inductor currents $I_L(t)$. Then, if the reachable set $R(t)$ containing the circuit states $x(t)$ intersects the unsafe region $\mathcal{U}$ that is defined in terms of critical voltage or current specified safe operating area (SOA) in PDKs.

Fig. 2.4 illustrates the problem of verifying if the operating region of a circuit is kept within the allowed safe-operating area in the state space. Given a circuit as Fig. 2.4, the behavior of the circuit is described with two continuous state variables I_L and V_C as Fig. 2.4 (b). However, the circuit can start with any initial values of the state variables I_L and V_C, leaving the possibility of un-tested errors that may cause damage to circuits. Reachability analysis can compute the reachable set of circuits from a set of initial conditions given by specification, leading to correct verification results as illustrated in Fig. 2.4 (d).

2.3 Conventional Hybrid System Reachability Analysis

Conventional hybrid system reachability analysis methods are mostly based on hybrid system abstraction and set-based reachability analysis. The algorithm can compute the

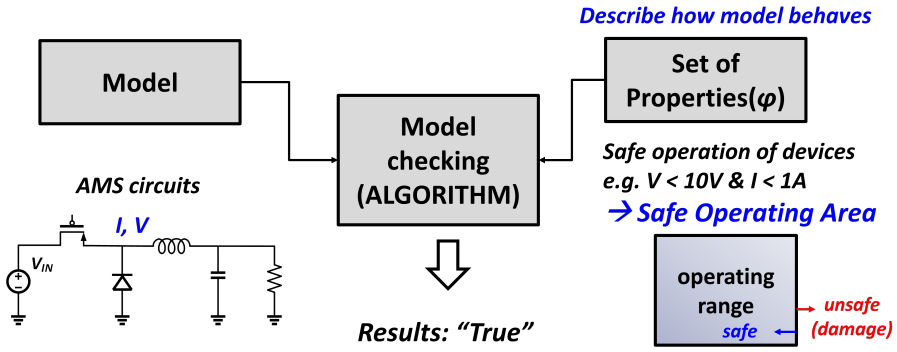


Figure 2.3: Model checking of safe operation of circuits.

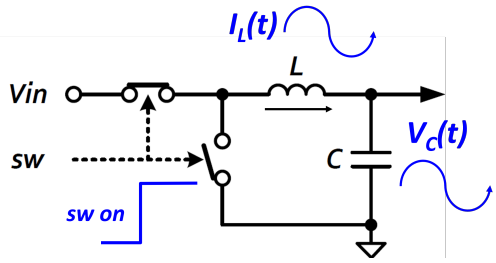
time evolution of a set of states using efficient set representation and hybrid dynamics formally specified by the hybrid automata model.

As we mentioned above, the efficiency of this technique highly depends on the selected set representation. In this section, we review the formal definition of the widely used set representation and their fundamental limitations. Lastly, we also explain the state-of-the-art reachability analysis tool SpaceEx, to which we will compare the proposed method.

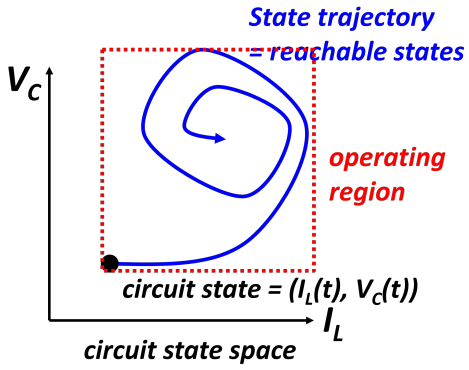
2.3.1 SpaceEx

The SpaceEx is a tool platform implementing algorithms related to reachability and safety verification suitable for continuous hybrid dynamic systems. It models the target system as hybrid automata with hierarchical automata and templates. The model editor for model hybrid automata is implemented with visual user interfaces as shown in Fig. 2.5.

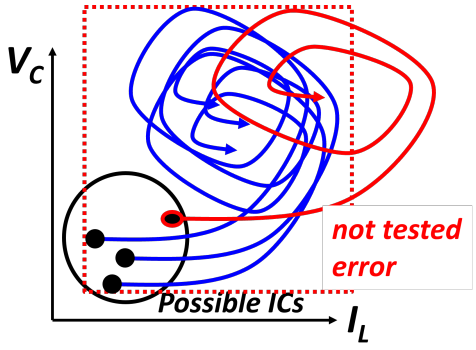
It basically based on computing reachable sets depending on time steps multiplied by δ. Then, the resulting reachable sets in each time interval are computed as in $\Omega_0, \Omega_1, ...$ There are several options to express the reachable set depending on the options depending on target accuracy and the resulting reachable sets are displayed over the output user interfaces as blue polygons as in Fig. 2.6, which shows the comparison



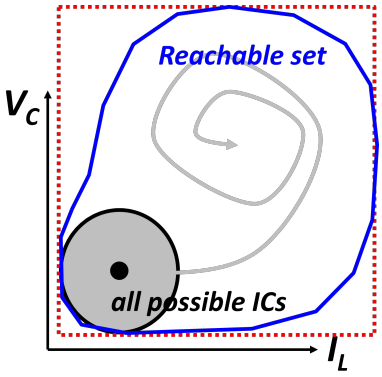
(a) circuit



(b) Operating region of a circuit in state space

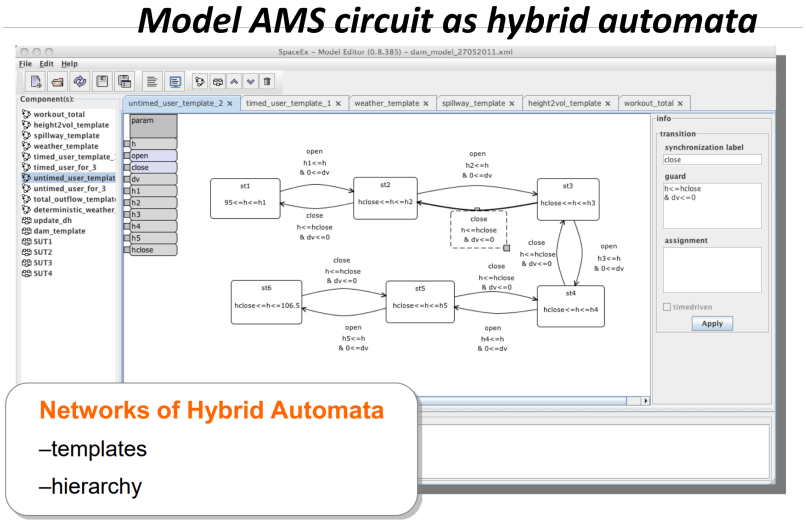


(c) Undiscovered bugs with set of initial conditions

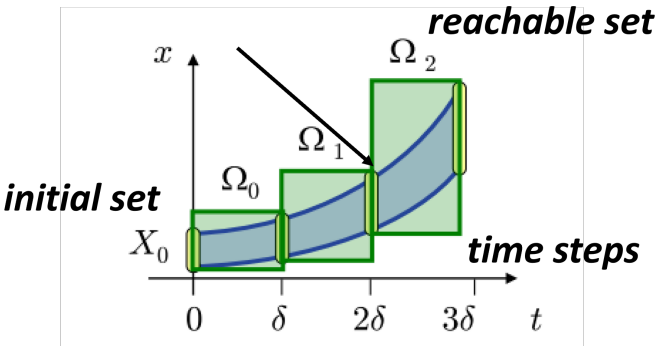


(d) Operating region obtained from reachable sets

Figure 2.4: SOA verification of a circuit with two state variables.

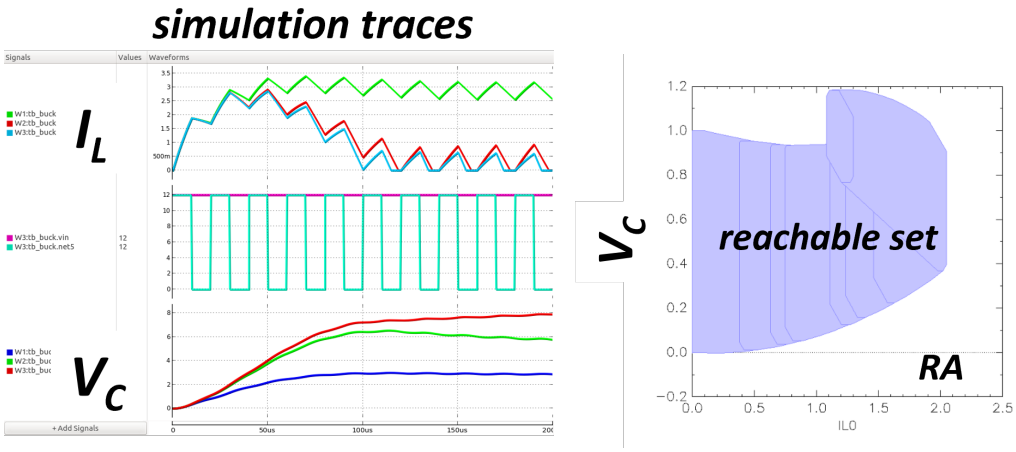


(a) circuit



(b) SOA in state space

Figure 2.5: Conventional reachability analysis tool SpaceEx.



(a) Simulation results varying load resistances using XMODEL.

(b) Reachable set using SpaceEx until t=10us.

Figure 2.6: Reachable set obtained in SpaceEx compared to simulation traces in XMODEL.

of the output simulation traces for currents and voltages of the exemplary circuits.

However, it fails to run until the end of the analysis due to exponentially increasing runtimes for our initial model describing the piecewise linear behavior of the typical circuits with MOSFETs or didoes. The rest of the dissertation tackles the problem associated with the conventional reachability analysis algorithm based on time discretization similar to the numerical integration algorithm of analog simulators such as SPICE simulators.

Chapter 3

TRAJECTORY-FORM REACHABILITY ANALYSIS

This chapter presents the main idea of the proposed trajectory-form reachability analysis methodology for analog/mixed-signal (AMS) circuits. We consider here specifically a linear hybrid system, which is a hybrid system, and its behavior in each continuous system is described by a linear system. First, we transform the linear circuit from topology to state space representation using the modified nodal analysis technique (MNA). Then, the obtained system matrices are transformed into transfer functions using the XMODEL algorithm. Combined with the s-domain input expression, we can obtain the trajectory form of the reachable set from the circuits.

3.1 Reachability Analysis on Linear Circuits

3.1.1 General Trajectory Form of Reachable Set

If a system representing the circuit is linear, computing reachable sets is straightforward. The dynamics of the continuous variable in the subset S of the state space, $x(t) \in S \subseteq \mathbb{R}^n$ and the output $y(t) \in \mathbb{R}^{n_o}$ with the input $u(t) \in \mathbb{R}^{n_i}$ in a n-

dimensional system are defined by

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{3.1}$$

$$y(t) = Cx(t) + Du(t) \tag{3.2}$$

where the system matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times n_i}$, $C \in \mathbb{R}^{n_o\times n}$, and $D \in R^{n_o\times n_i}$. n_i and n_o are the numbers of inputs and outputs, respectively. Then, the state trajectory $x(t) \in \mathbb{R}^n$ from the initial condition, $x_0 \in X_0$ governed by the linear system (3.1) is given by

$$x(t) = e^{At}x_0 + \int_0^t e^{A(t-\tau)}Bu(\tau)d\tau \tag{3.3}$$

Therefore, the reachable set $\mathcal{R}(t)$ from the initial conditions $X_0 \subseteq \mathbb{R}^n$ in the linear system (3.1) is defined by set of state trajectories (3.3) as

$$\mathcal{R}(t) = \{x(t) = e^{At}x_0 + \int_0^t e^{A(t-\tau)}Bu(\tau)d\tau \mid x_0 \in X_0\}. \tag{3.4}$$

It has been shown in [28] that when each of the input variable $u(t)$ has the form:

$$u(t) = \sum_j c_j t^{m_j} e^{-a_j t} \tag{3.5}$$

where the coefficients a_j's and c_j's are complex numbers and m_j's are non-negative integers, then the time-domain solutions for the state variables $x(t)$ and output variables $y(t)$ governed by (3.1) can be represented by the identical form again. That is, if the set of initial condition $\mathcal{R}_0$ is specified by vector set $C = \{v_1, v_2, ..., v_k \in \mathbb{R}^n\}$ by

$$\mathcal{R}_0 = \{\lambda_1 v_1 + \lambda_2 v_2 + ... \mid v_i \in C, \lambda_i \in \mathbb{R}, i = 1, 2, ...\}. \tag{3.6}$$

Recognizing that each vector changes with time t and the scaling factor λ_i's remains constant, we only need to compute the time evolution of vector v_i's; thus, applying the trajectory form of (3.5), we get the identical form with (3.6) having its vector element as trajectory form of (3.5) as:

$$\mathcal{R}(t) = \{x(t) = \sum_i \lambda_i v_i(t) \mid v_i(t) = \sum_j c_{ij} t^{m_{ij}} e^{-a_{ij}t}, \lambda_i \in \mathbb{R}\}. \tag{3.7}$$

This general form of the reachable set can express any states $x(t)$ starting from the vector set R_0 including the initial conditions X_0 at arbitrary time t by a linear combination of vector v_i's expressed as a set of real coefficients $\{(Re(c), Im(c), Re(a), Im(a), m)_j, ...\}$. In addition, a set of states sampled at a specific time instant t', $\mathcal{R}(t')$ have an identical form with the initial condition set $\mathcal{R}_0$, from which we can iteratively compute the discrete change of the input $u(t)$ or the system matrices of the circuit. This concept can be extended to any kind of set representations introduced in the previous chapter. However, we focus on representing a reachable set with zonotope owing to its scalability.

3.1.2 Zonotope Trajectory Form of Reachable Set

Let assume that the range of initial states $\mathcal{R}(t = 0)$ is given as a zonotope $\mathcal{Z}$, which is described by a point, referred as center $c \in \mathbb{R}^n$ and a set of generators g_i's $\in \mathbb{R}^n$ [14]:

$$\mathcal{Z} = (c, \langle g_1, ..., g_r) = \{c + \sum_{i=1}^{r} \alpha_i g_i \mid \alpha_i \in [0, 1]\} \tag{3.8}$$

which is a special case of (3.7) having $1 + r$ vectors and its scaling factors have range of values $\lambda_0 = 1$ for center and $\lambda_i = \alpha_i = [0, 1]$ for $k = 1, ..., r$ for the others. This is the special case of (3.6) when the one of vectors in set C is point c and the other vectors g_i's are vectors originating from the point, referred to as an affine combination [29]. Therefore, any point within a zonotope can be expressed as a linear combination of generator g_i's from the center c. Its computational complexity for a linear map depends on the number of generators r, yielding

This is also a linear combination of vectors with constant scaling factor α_i thus we can define its trajectory form of reachable set $\mathcal{R}_{\mathcal{Z}}(t)$ having input $u(t)$ of (3.5)[30] as:

$$\mathcal{R}_{\mathcal{Z}}(t) = (c(t), \langle g_1(t), ..., g_i(t)\rangle) \tag{3.9}$$

$$= \{c(t) + \sum_{i} \alpha_i g_i(t) \mid \alpha_i \in [0, 1], g_i = \sum_{j} c_{ij} t^{m_{ij}} e^{-a_{ij} t}, \} \tag{3.10}$$

is the zonotopic trajectory form of the reachable set $\mathcal{R}_{\mathcal{Z}}(t)$. For convenience, after that, simply we denote the trajectory form reachable set $\mathcal{R}(t)$ for the zonotopic trajectory form of reachable set $\mathcal{R}_{\mathcal{Z}}(t)$.

3.1.3 Computing Trajectory Form using Laplace s-domain Transfer Function

In this subsection, we explain a procedure named *getTrajectoryZonotope*() that computes the trajectory form of $\mathcal{R}(t)$ from the zonotope range of the initial states $\mathcal{R}(t=0)$ defined by zonotope $\mathcal{Z}$ in (3.8) and the inputs expressed in the form of (3.5). For instance, the time-domain expressions of $u(t)$ in (3.5) can be transformed into the Laplace s-domain expression $U(s)$:

$$u(t) = \sum_j c_j t^{m_j} e^{-a_j t} \rightarrow U(s) = \sum_j \frac{b_j}{(s+a_j)^{m_j+1}} \tag{3.11}$$

and the Laplace-domain transform of resulting state trajectory $x(t)$, $X(s)$ can be computed using the matrices in (3.1) as:

$$X(s) = (sI - A)^{-1}BU(s) + (sI - A)^{-1}x(0), \tag{3.12}$$

which can be transformed back to a time-domain expression in (3.5). The zonotope reachable set $\mathcal{R}(t)$ is computed by applying this procedure to the vectors of the initial zonotope $\mathcal{Z}$. That is, the Laplace transform of vectors in the zonotope $\mathcal{Z}$ is computed by the same matrices derived from the given circuit as:

$$c(s) = (sI - A)^{-1}BU(s) + (sI - A)^{-1}c(0) \tag{3.13}$$

$$g_i(s) = (sI - A)^{-1}BU(s) + (sI - A)^{-1}g_i(0), \tag{3.14}$$

and its time domain solution is given by its inverse Laplace transform

$$c(t) = \sum_j c_{0j} t^{m_{0j}} e^{-a_{0j} t} \tag{3.15}$$

$$g_i(t) = \sum_j c_{ij} t^{m_{ij}} e^{-a_{ij} t}. \tag{3.16}$$

That is, keeping a set of coefficients $\{(a_{00}, c_{00}, m_{00}), ..., (a_{ij}, c_{ij}, m_{ij}), ...\}$ in the expression (3.16), one can sample arbitrary states in the reachable set $\mathcal{R}(t)$ for range of time $t \in [0, t]$. Lets define it as the *canonical* form the zonotopic reachable set $\mathcal{R}(t)$, which is

$$\mathcal{R}(t) = \{(a, c, m)_{ij} \mid i = (0, ..., r), j = (0, ..., N_p)\} \tag{3.17}$$

where a's, and c's are complex-numbered vectors $\in \mathbb{R}^n$ and m's are integers $\in \mathbb{C}^n$. The coefficient of $i = 0$ denotes that of center vector c and the others are for those of generators g_i's. It has been shown that the coefficients can be computed by the partial fraction decomposition of s-domain expression and the number of coefficients N_p increase with the number of resulting first order poles in [28]. Therefore, we can express a continuous set of states until time t using a zonotope reachable set with r generators by $(r + 1) \times 5N_p$ number of real vectors $\in \mathbb{R}^n$.

To avoid redundant matrix computations of the transfer function $(sI - A)^{-1}$ and $(sI - A)^{-1}B$, we define the expression of transfer function

$$H(s) = (sI - A)^{-1}. \tag{3.18}$$

for the same linear differential equation in (3.1), we keeps the previously computed transfer function $H(s)$ and reuse in the the remaining computation. For example, given a zonotope having comprising two generators $\mathcal{Z}_0 = (c(0), \langle g_1(0), g_2(0)\rangle)$, the reachable set from the zonotope $\mathcal{Z}_0$, is obtained as $c(s) = H(s)(BU(s) + Ec(0))$ and $g_1(s) = H(s)(Eg_1(0))$,$g_2(s) = H(s)(Eg_2(0))$ in s-domain. The time-domain state of the reachable set can be obtained from the inverse Laplace transform, i.e. the form in (3.16).

3.1.4 Example: Reachable set of RC circuit

Now, assume that a linear circuit having a resistor R and capacitor C in Fig 3.1 with a range of initial condition $V_c(0) = [1, 2]$. Then, applying nodal analysis to the given circuit, the behavior of the state variable V_c can be represented as a first-order linear

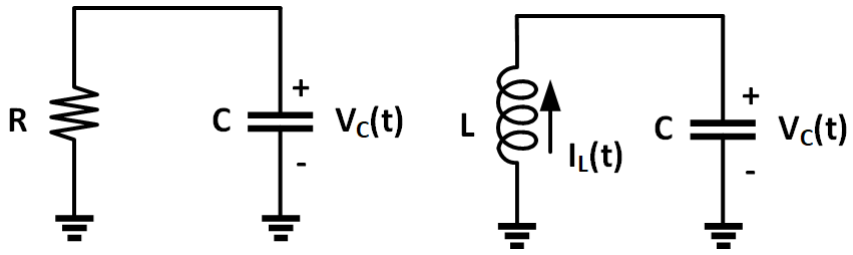


Figure 3.1: Simple exemplary circuits (a) RC circuit (b) LC circuit.

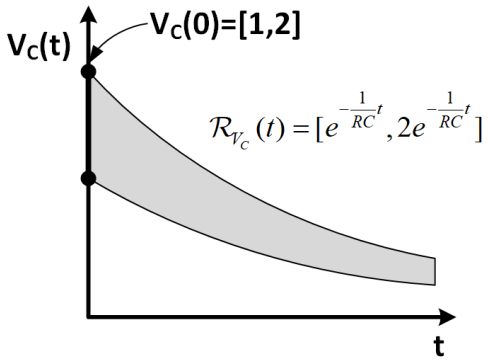


Figure 3.2: Reachable set of the exemplary circuits (a).

system of a state space representation as:

$$\dot{V_c} = -\frac{1}{RC}V_c \tag{3.19}$$

Then, its reachable set $\mathcal{R}(t)$ of the form (3.17) from the initial condition $V_c(0)$ can be obtained as:

$$\mathcal{R}_{V_C}(t) = \{c = (\frac{1}{RC}, 1.5, 1), \langle g_0 = (\frac{1}{RC}, 0.5, 1)\rangle\} \tag{3.20}$$

This represents the set of trajectories having infinite states from the initial states X_0 until t in the circuit using only two sets of algebraic coefficients. Fig. 3.2 illustrates the resulting reachable set.

3.1.5 Example: Reachable set of LC circuit

In the same manner, the example of LC circuits exhibits harmonic oscillating behavior repeating charging and discharging the LC energy storage elements. The state space representation of the circuit is given by :

$$\dot{I_L} = -\frac{1}{L}V_c \tag{3.21}$$

$$\dot{V_c} = \frac{1}{C}I_L \tag{3.22}$$

and the corresponding system matrix $A \in \mathbb{R}^2$ can be derived from (3.22) as

$$A = \begin{pmatrix} 0 & -1/L \\ 1/C & 0 \end{pmatrix}. \tag{3.23}$$

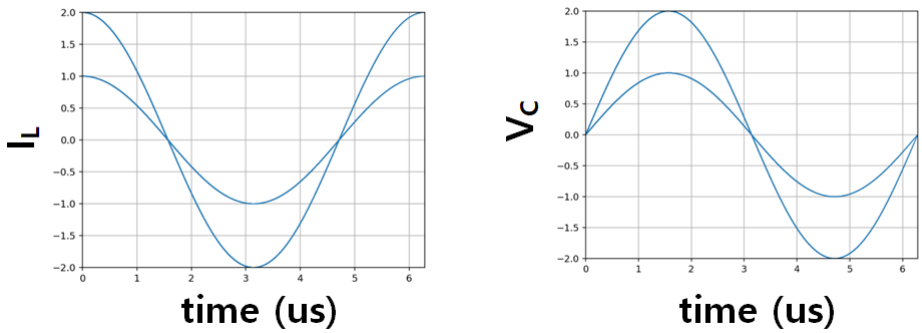
In the same manner, its reachable set $\mathcal{R}_{LC}(t)$ is defined by a two-dimensional zonotope having a single 2-D generator vector as:

$$\mathcal{R}_{LC}(t) = \{c = \begin{pmatrix} \{(\frac{1}{RC}, 1.5, 1)\} \\ \{(\frac{1}{RC}, 1.5j, 1)\} \end{pmatrix}, \langle g_0 = \begin{pmatrix} \{(\frac{1}{RC}, 0.5, 1)\} \\ \{(\frac{1}{RC}, 0.5j, 1)\} \end{pmatrix} \rangle\} \tag{3.24}$$

Fig. 3.3 shows that the proposed reachable set of the circuit can accurately represent every state $x(t) = (I_L(t), V_c(t))$ from $x_0 \in X_0$ with respect to a single period of time $t \in [0, 2\pi\sqrt{LC}]$ by a linear sum of the trajectory of center point $c(t)$ and scaled generator vector $g_1(t)$. Each circuit element has the value of $L = 1\,\mu\text{H}$ and $C = 1\,\mu\text{F}$, respectively, resulting in the harmonic oscillation with the frequency of $\omega_0 = 2\pi f_0 = 1/\sqrt{LC} = 1\,\frac{\text{Mrad}}{\text{s}}$.

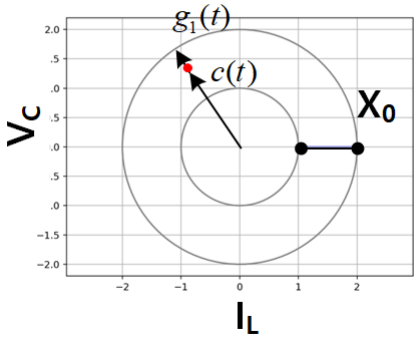
3.1.6 Comparison with Existing Algorithms

Conventional reachability analysis algorithm iteratively computes the reachable set at each time step $t_k = \{k\Delta t \mid k = 0, ..., N_t\}$ where the size of the unit time step $\Delta t = T/N_t$, resulting in the trade-off between the accuracy and the runtime of the computation of reachable sets. In this subsection, we compare the results of the equivalent LC harmonic oscillator system performed in SpaceEx.



(a) (b)

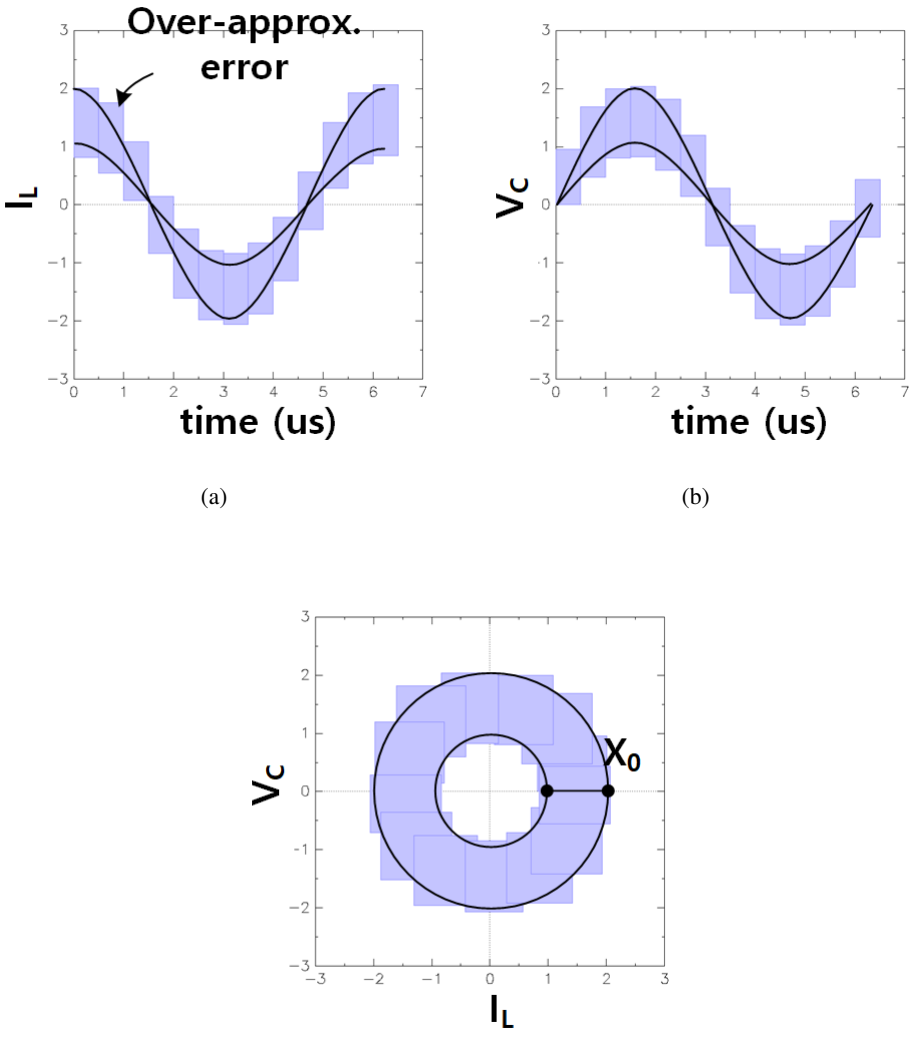
$$x(t) = (I_L(t), V_C(t)) = c(t) + \alpha_1 g_1(t)$$



(c)

Figure 3.3: Reachable set of the LC harmonic oscillator circuit. (a) I_L-t (b) V_C-t (c) V_C-I_L.

Fig. 3.4 shows the reachable set computed using the size of the unit time step $\Delta t = 0.5\,\mu s$. It took $0.004\,s$ to compute a single period of oscillation but exhibits a large over-approximation error compared to the ideal time-domain response having a form of sine wave as shown in Fig. 3.3. With finer time steps of $\Delta t = 0.1\,\mu s$, it shows a more accurate reachable set compared to the ideal one, but the computation time increased $5\times$ than that of coarse time steps. Furthermore, if the circuit operates with a higher frequency, this discrepancy would become larger, which is the problem

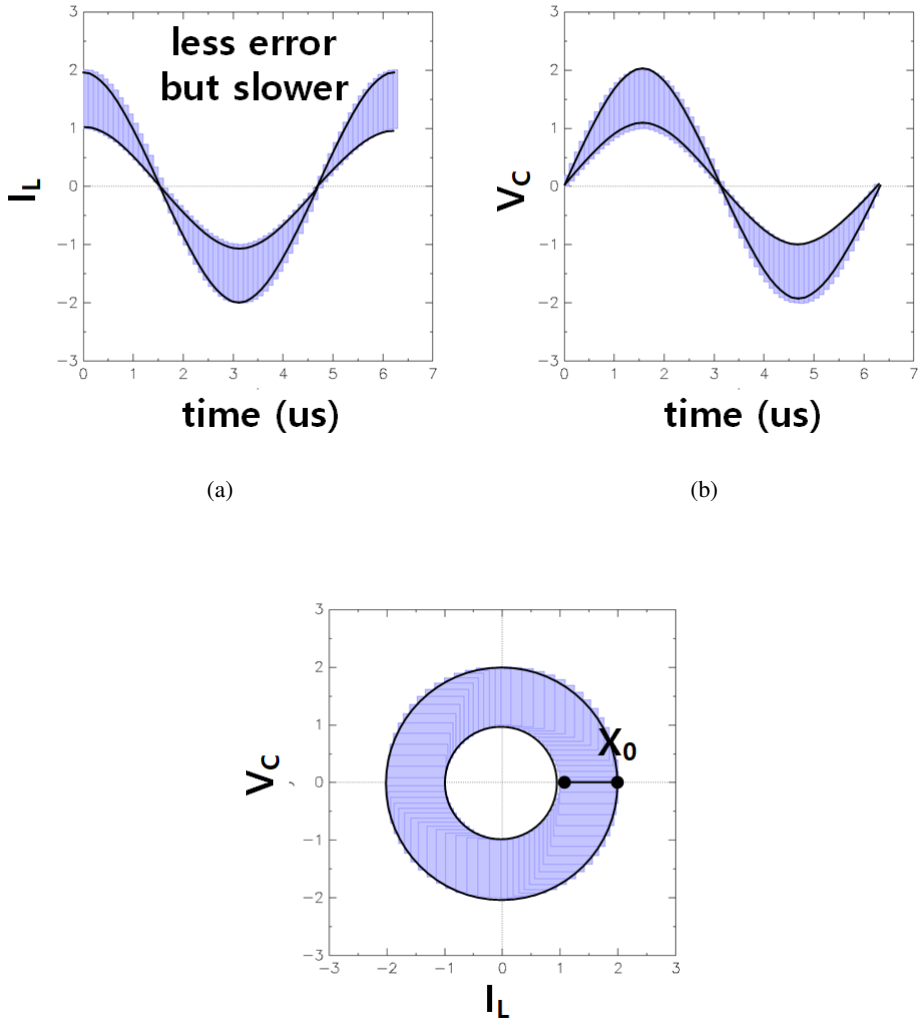


(a) (b)

(c)

Figure 3.4: Reachable set in SpaceEx (a) I_L-t (b) V_C-t (c) V_C-I_L.

that recent reachability analysis suffers from. The proposed trajectory-form reachable set can obtain the exact reachable set within $26\,\mu$s, outperforming the conventional algorithm by $69\times$.



(a) (b)

(c)

Figure 3.5: Reachable set in SpaceEx (a) I_L-t (b) V_C-t (c) V_C-I_L.

3.2 Hybrid System Reachability Analysis

3.2.1 Hybrid System Representation of AMS Circuits

Analog and mixed-signal circuits are in general non-linear so they require hybrid system representation, i.e. hybrid automata. This is a finite state machine that has con-

tinuous and discrete variables, of which each discrete state $q \in Q$ has continuous states $x \in X \subseteq \mathbb{R}^n$ that operate as a different linear system represented by a set of differential-algebraic equations F. Mixed signal circuits change their discrete states that represent the topology of the circuit depending on the state of the switches connecting the remaining circuit. Then, in the selected linear system, the time evolution of the continuous variable $x(t)$ is iteratively computed from the set of continuous states $x_0 = x(t')$ in the previous discrete state. The resulting hybrid automata are defined by $M = \langle Q, X, F, E, G \rangle$ where E is a set of discrete transitions that describe the switching condition of the circuit, and $\mathcal{G}$ is a set of guard condition that determines when to transit to next discrete state.

3.2.2 Hybrid System Reachability Analysis Using Trajectory Form

From the hybrid automata M representing a given circuit and the initial conditions $X_0 \subseteq X$ and $q_0 \in Q$, the proposed algorithm computes the reachable set $\mathcal{R}(t)$ of the continuous variable $x(t) \subseteq X$ and discrete variable $m(t) \subseteq Q$. The resulting hybrid reachable set is a series of tuples consisting of values of discrete state $m(t)$ and each continuous reachable set $\mathcal{R}_\zeta(t)$ and the corresponding each time segment $[t_{\zeta-1}, t_\zeta]$ the trajectory form of the reachable set $\mathcal{R}(t)$ of (3.16) as :

$$\mathcal{R}(t) = \{(m_1, [t_0, t_1], \mathcal{R}_1(t)), (m_2, [t_1, t_2], \mathcal{R}_2(t)), ..., (m_\zeta, [t_{\zeta-1}, t_\zeta], \mathcal{R}_\zeta(t), ...)\} \tag{3.25}$$

where ζ is the index of each time period of operating the linear system of ζ-th discrete state of the sequence $m_\zeta \in Q$.

Algorithm 1 summarizes the described iterative procedure to compute the reachable set with given conditions. This assumes the number of cycles for analysis is bounded to N_{cycle} to avoid undecidable conditions such as infinite loops. In each loop, the procedure *GetTrajectoryZonotope*() computes the reachable set of continuous state $x(t)$ in the discrete state $m(t) = q_\zeta$, i.e. $\mathcal{R}_q(t), t \in [t_\zeta, t_{\zeta+1}]$ using the trajectory form in (3.16). This begins with the last set of states sampled at the previous discrete state

Algorithm 1: Computing hybrid reachable set $\mathcal{R}(t)$.

Input : $M, (q_0, \mathcal{Z}_0)$

Output: $\mathcal{R}(t)$

$\mathcal{R}(t) \leftarrow \emptyset$;

for $\zeta \in (1, ..., N_{cycle})$ **do**

  $\mathcal{R}_j(t) \leftarrow GetTrajectoryZonotope(\zeta, \mathcal{Z}_0)$ ▷ `ζ-th trj. form`

  $(q', t_{\zeta+1}) \leftarrow GetNextState(\zeta, \mathcal{R}_\zeta(t))$;

  $\mathcal{R}(t) \leftarrow \mathcal{R}(t) \cup \{(q_0, [t_\zeta, t_{\zeta+1}], \mathcal{R}_\zeta(t))\}$;

  $q_0 \leftarrow q'$;

  $\mathcal{Z}_0 \leftarrow \mathcal{R}_\zeta(t_{\zeta+1} - t_\zeta)$;

end

return $\mathcal{R}(t)$

at t_ζ, $\mathcal{Z}_0$. It returns the resulting reachable set $\mathcal{R}_\zeta(t)$ for the time range $t = [t_\zeta, t_{\zeta+1}]$. The hybrid reachability analysis algorithm iteratively computes the reachable set using this procedure until the bounded number of cycles N_{cycle}. Then, the procedure *GetNextState* determines the next value of the continuous and discrete states, q' and $\mathcal{R}_\zeta(t_{\zeta+1} - t_\zeta)$ from the next beginning time $t_{\zeta+1}$ by computing the guard intersection $\mathcal{R}_\zeta(t) \cap \mathcal{G}$. However, in this section, we limit the scope of analysis by assuming that the guard set is only set for the time variable t, i.e. timed automata with a hybrid system, not by the internal continuous state variable $x(t)$, which will be discussed at the next chapter.

3.2.3 Example: Switched RC Circuit

As an illustrative example, we analyze the behavior of an AMS circuit, switched RC circuit controlled by input pulses for switches. It is simple but represents the basic building block of a wide range of circuit applications such as charge pumps, DRAM cells, and other switching power supplies. This can be equivalently modeled by a switched linear system as previously studied in [28].

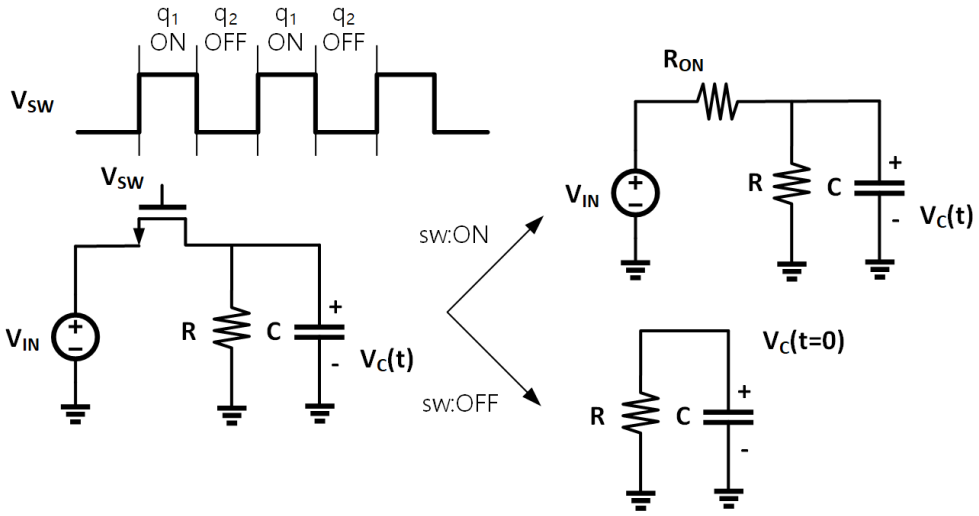


Figure 3.6: Switched RC Circuit with cyclic pulse inputs and the corresponding linear circuits.

Fig. 3.6 shows the circuit under analysis, which changes its discrete state $q \in Q$ depending on the input level V_{sw} of the switch transistor, which is often controlled by pulse-width modulation or pulse-frequency modulation control. We assume it has two discrete states $q_1, q_2 \in Q$ and switched depending on the T_{on}, T_{off}, periodically, where q_1 corresponds to the charging state where the external voltage V_{IN} charges the capacitor C and q_2 is discharging state where the charges in the capacitor C discharges to the shunt resistor R. When the input V_{sw} is larger than the threshold of the transistor ($q = q_1$), the control switch is turned on or is turned off ($q = q_2$), elsewhere. Therefore, the set of discrete transitions depending on the switch state q is defined by $E = \{q_1 \rightarrow q_2$(sw:off), $q_2 \rightarrow q_1$(sw:on)$\}$. Each linear system corresponding to two discrete states q_1 and q_2 is derived using basic circuit analysis as :

$$\dot{V_c} = -(\frac{1}{RC} + \frac{1}{R_{ON}C})V_c + \frac{1}{R_{ON}C}V_{IN} \text{ for } q_1 \tag{3.26}$$

$$\dot{V_c} = -\frac{1}{RC}V_c \text{ for } q_2 \tag{3.27}$$

The trajectory form from (3.27) can be solved by using the Laplace technique in

(3.25). For switch on state q_1, the resulting trajectory-form reachable set is :

$$V_c(s) = \frac{V(0)}{s + \frac{1}{RC} + \frac{1}{R_{ON}C}} + \frac{1}{s + \frac{1}{RC} + \frac{1}{R_{ON}C}} \cdot \frac{V_{IN}}{sR_{ON}C} \tag{3.28}$$

The second term in (3.28) results in two sets of coefficients by the partial fraction decomposition $\frac{a'}{s} + \frac{b'}{s+\frac{1}{RC}+\frac{1}{R_{ON}C}}$ where $b' = -a' = \frac{V_{IN}}{R_{ON}}/(\frac{1}{RC} + \frac{1}{R_{ON}C})$. That is, we require three tuples of coefficients for representing the voltage V_c when the circuit is in the charging state q_1 with a range of initial condition $V_c(0) = [V_{c,lb}, V_{c,ub}]$. Therefore, the resulting reachable set is given by :

$$\begin{aligned} \mathcal{R}_{q_1}(t) = \{&c = \{(\omega_{p1}, V_c', 1), (\omega_{p2}, a', 1), (\omega_{p1}, b', 1)\} \\ &\langle g = \{(\omega_{p1}, V_c'', 1), (\omega_{p2}, a', 1), (\omega_{p1}, b', 1)\rangle\} \end{aligned} \tag{3.29}$$

where at the beginning of the analysis the initial voltage is given by $V_c' = (V_{c,lb} + V_{c,ub})/2$ and $V_c'' = (V_{c,lb} - V_{c,ub})/2$ for the center c_{q1} and the generator g_{q1}, respectively, and each system pole $\omega_{p1} = \frac{1}{RC} + \frac{1}{R_{ON}C}$ and $\omega_{p2} = 0$. The reachable set in discharging state q_2 also can be solved in the same manner, yielding :

$$\begin{aligned} \mathcal{R}_{q_2}(t) =&\{c = (\omega_p, V_c', 1), \\ &\langle g = (\frac{1}{RC} + \frac{1}{R_{ON}C}, V_c'', 1)\rangle\} \end{aligned} \tag{3.30}$$

where the initial values V_c' and V_c'' are computed at the last time in the previous discrete state q_1. The resulting reachable set is given by a set of (3.29) and (3.30).

$$\begin{aligned} \mathcal{R}(t) = \{&(q_1, [t_0, t_1], \mathcal{R}_{q_1}(t)), \\ &(q_2, [t_1, t_2], \mathcal{R}_{q_2}(t)), ... \\ &(q_1, [t_2, t_3], \mathcal{R}_{q_1}(t)), \\ &(q_2, [t_3, t_4], \mathcal{R}_{q_2}(t)), ...\} \end{aligned} \tag{3.31}$$

Chapter 4

HYBRID SYSTEM REACHABILITY ANALYSIS OF NONLINEAR CIRCUIT

We discussed the hybrid system with linear circuits in the previous chapter. In this chapter, we present the methodology for the hybrid system including nonlinear circuit behavior [31].

4.1 Piecewise-Linear Modeling of AMS Circuits

The most prevalent approach for analyzing nonlinear circuits is to model them as a piecewise-linear (PWL) system that divides a state space having nonlinear dynamics into several regions operating in linear systems. It has been announced that most analog circuits are nonlinear owing to two nonlinear elements, diodes, and MOSFETs; their non-linear characteristics are efficiently and accurately modeled by PWL model with the threshold voltage V_{TH} and the on-resistance R_{ON} [26].

As shown in Fig. 4.1, when any circuit has a diode, the circuit can be represented by two linear systems depending on the terminal voltage V_D, and each linear system is represented in two systems having R_{ON} and R_{OFF}(or open). In addition, the MOSFET characteristics also can be efficiently represented with two PWL diodes, and the resulting system would be partitioned by four linear systems. Using this PWL parti-

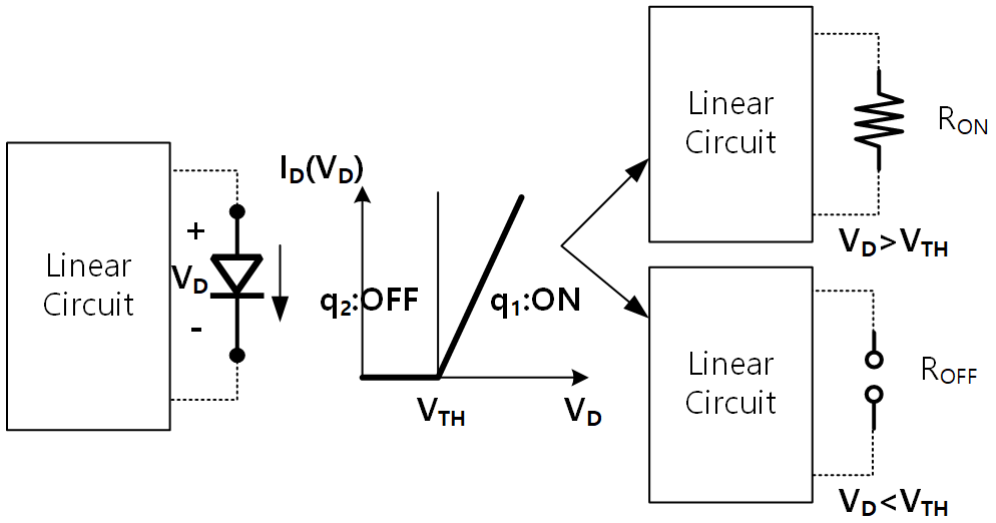


Figure 4.1: PWL partitioning of the circuit with a diode.

tioning, we can model a nonlinear AMS circuit and apply hybrid reachability analysis.

4.2 Piecewise-Linear Approximation of Nonlinear Circuits

In the PWL approximation, the continuous state space of a non-linear system is partitioned into a set of disjoint sub-regions $\{S_1, ..., S_p, ..., S_{N_r}\}$, and the continuous state $x(t) \in S_p$ is governed by a different linear system depending on the sub-region $x(t)$ belong to, where each sub-region S_p for $p = 1, ..., N_r$ corresponds to the discrete state $d(t) \in \{1, ..., N_r\} \in Q_d$. The key challenge in computing the reachable set $\mathcal{R}(t)$ of $x(t)$ and $d(t)$ (or with $m(t)$) representing the change of linear system due to time t lies mainly with computing the guard intersection $\mathcal{R}(t) \cap \mathcal{G}$. This chapter addresses how to compute this guard intersection of continuous state $x(t)$ efficiently and accurately.

The challenges in computing the guard intersection $\mathcal{I} = \mathcal{R}(t) \cap \mathcal{G}$ in conventional methods are two-fold. First, when the reachable set $\mathcal{R}(t)$ is evaluated at discrete steps of time t, the guard intersection must be computed repeatedly for each time step, incurring a large computational cost. In addition, this cost further increases when comput-

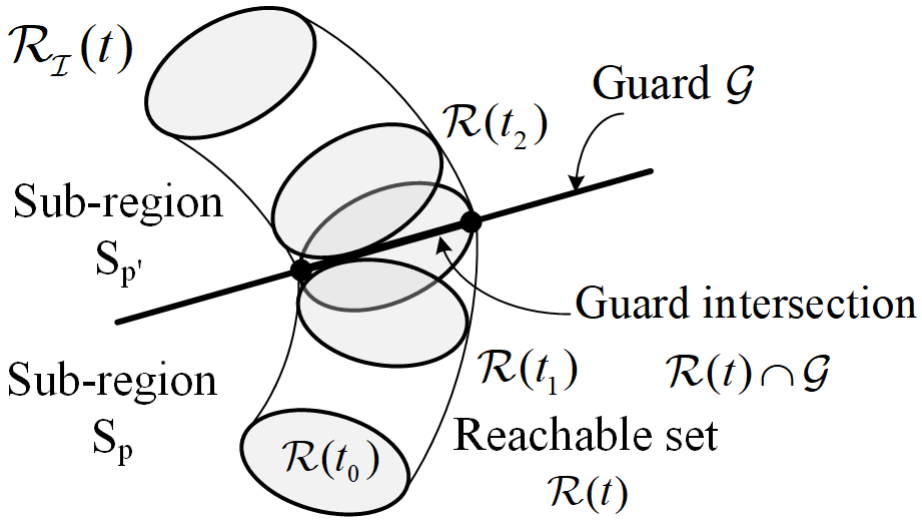


Figure 4.2: The problem of computing the guard intersection $\mathcal{R}(t) \cap \mathcal{G}$ in PWL system representation.

ing the subsequent reachable set from the guard intersection $\mathcal{I}$ since each intersection requires computing the successor reachable set $R_{\mathcal{I}}(t)$, resulting in the exponential increase of runtime with respect to the number of guard intersection, referred to *cycles* here.

To address this challenge, this work proposed an efficient, scalable way of computing the guard intersection of reachable set $\mathcal{R}(t)$ represented in a trajectory form of (3.16). In the previous chapter, it has been shown that for linear hybrid systems, computing the time evolution of $\mathcal{R}(t)$ within a region S_i does not need the evaluation of $\mathcal{R}(t)$ at discrete steps of time. Instead $\mathcal{R}(t)$ can be represented using a set of vertices, where each vertex follows a trajectory described by an exact and analytical function of time t of (3.7). The proposed method can compute the intersection of $\mathcal{R}(t)$ in this trajectory form with a linear guard $\mathcal{G}$, and also a new reachable set $\mathcal{R}(t)$ starting from the computed intersection $\mathcal{I}$, both in the trajectory forms that do not rely on time discretization. The proposed method also adopts the scalable method in [15] that computes the intersections in the high-dimensional state space using only two-dimensional

operations via projections.

4.3 Computing Guard Intersections at PWL Switching Boundary

4.3.1 Hybrid System Representation of PWL circuits

The continuous state space of the PWL approximated circuit is partitioned into a set of sub-regions S_p for $p = 1, ..., N_p$ by a set of linear hyperplane guards $\mathcal{G}'s$:

$$\mathcal{G}_h : w_h \cdot x + b_h = 0 \text{ for } h = 1, 2, ... \tag{4.1}$$

where $w_h \in \mathbb{R}^n$ is an unit vector normal to each hyperplane and $b_h \in \mathbb{R}^n$ is an offset. Within a sub-region S_p and for a given discrete state $d(t) \in 1, ..., q$, the continuous state variable $x(t) \in \mathbb{R}^n$, the input $u(t) \in \mathbb{R}^{n_i}$, and the output $y(t) \in \mathbb{R}^{n_o}$ are governed by a linear differential equation :

$$\begin{aligned} \dot{x}(t) &= A_p x(t) + B_p u(t), \\ y(t) &= C_p x(t) + D_p u(t) \end{aligned} \tag{4.2}$$

where $A_p \in \mathbb{R}^{n \times n}$, $B_p \in \mathbb{R}^{n \times n_i}$, $C_p \in \mathbb{R}^{n_o \times n}$, and $D_p \in R^{n_o \times n_i}$; and p denotes the index of the operating sub-region of PWL system. Then, the trajectory of $x(t)$ starting from an initial state x_0 can be represented by the hybrid system trajectory form $\mathcal{R}(t)$ of (3.25). Each transfer function $H_p(s) = (sI - A_p)^{-1}$ for the system matrices in (4.2) is computed in the same manner in the previous chapter, indexed by the sub-region S_p.

4.3.2 Computing Guard Intersection Using Trajectory Form

This section describes the proposed algorithm for computing the guard intersection $\mathcal{I} = \mathcal{R} \cap \mathcal{G}_h$. For a zonotope reachable set $\mathcal{R}(t)$ in (3.16) and a hyperplane guard $\mathcal{G}_h$ in (4.1), using a set of orthogonal unit vectors l_k's to the vector w_h of the guard $\mathcal{G}_h$. The pseudo-codes of the overall algorithm are listed in Algorithm 2.

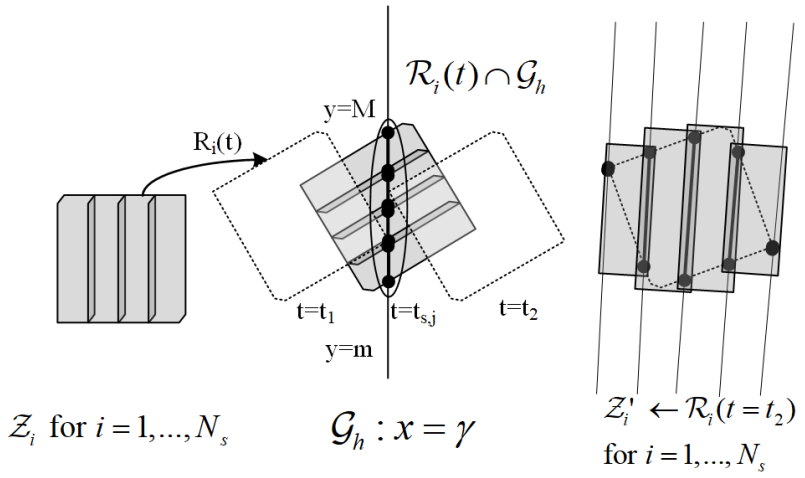


Figure 4.3: The overall algorithm to compute the guard intersection at the disjoint regions S_p and $S_{p'}$.

.

The algorithm first obtains the reachable set in the initial sub-region S_p as:

$$\mathcal{R}_{\bar{i}}(t) = \textit{getTrajectoryZonotope}(p, \mathcal{Z}_{\bar{i}}). \tag{4.3}$$

where p is the index for the corresponding linear system and $\bar{i}$ denotes the index for the initial zonotopes, previously segmented by $\bar{i} = 1, ..., N_s$. Then, the reachable set in the sub-region S_p is added by $\mathcal{R}(t) = \mathcal{R}(t) \cup \{\mathcal{R}_{\bar{i}}(t)\}$ at each iteration of $\bar{i} = 1, ..., N_s$.

Then, the algorithm can determine the time range $[t_1, t_2]$ for which the intersection $\mathcal{R} \cap \mathcal{G}_h \neq \emptyset$. The procedure named *findCrossZonotope*() finds the values of t_1 and t_2 by solving the equation

$$c(t)w_h + \sum_i |g_i(t) \cdot w_h| = 0 \tag{4.4}$$

iteratively until the time t reaches the maximum time bound T.

Since the guard intersection $\mathcal{R}(t) \cap \mathcal{G}$ does not yield a closed-form expression in terms of t, the algorithm computes the approximate intersections of $\mathcal{R}(t)$ sampled at $N_s + 1$ time steps spanning the range $[t_1, t_2]$ as :

$$t_{s,\bar{j}} = t_1 + \bar{j}\frac{t_2 - t_1}{N_s} \text{ for } \bar{j} = 1, ..., N_s \tag{4.5}$$

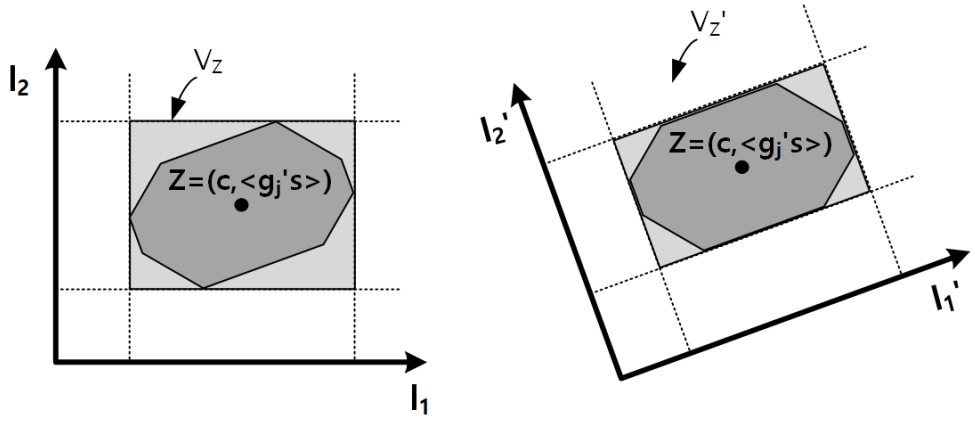


Figure 4.4: Search process for the best l_k's in the procedure for approximating $\mathcal{R}(t) \cap \mathcal{G}$ with low over-approximation error *getOrthogonalBasis*.

In addition, this algorithm adopts the projection method introduced in [15], which computes the guard intersection of the reachable set $\mathcal{R}(t)$ projected into a two-dimensional space instead of the original high-dimensional reachable set $\mathcal{R}(t)$ itself. The resulting intersections are obtained as the same zonotopes using the same set of projection vectors l_k for $k = 1, ..., n-1$, which are easy to combine with the subsequent step of computing the reachable set explained in the next section.

The procedure *getOrthogonalBasis*() finds the set of optimal orthogonal basis vectors

$$D = \{l_k \in \mathbb{R}^n \mid k = 1, ..., n-1\} \tag{4.6}$$

, with which one can describe a minimum-volume hypercube enclosing the zonotope $\mathcal{R}(t)$ at $t = t_{mid} = (t_1 + t_2)/2$. The search process guarantees that one of the basis vectors in D in (4.6) is selected from the generators g_j's and finds the set of basis vectors that minimizes the volume computed by the oriented rectangular hull (ORH) as:

$$V = \prod_{k=1}^{n-1} \sum_{i=1}^{r} |g_i \cdot l_k|. \tag{4.7}$$

Fig. 4.4 illustrates how the procedure chooses the best basis vectors l_k's. The initial

selection of l_k's shows a large over-approximation of the hypercube from l_k's approximating the zonotope $Z' = \mathcal{R}(t_{mid})$, whereas when the volume $V'_{\mathcal{Z}}$ is the minimum among every $V_{\mathcal{Z}}$, the hypercube of with l_k can tightly enclose the zonotope $\mathcal{Z}$. This search process assumes that the time evolution of zonotope $Z$is so slow for the intersecting time range $[t_1, t_2]$ that the approximation can fit the zonotope well with the chosen l_k's when the time in the middle of the period, t_{mid}. Therefore, it leaves further research issues for the method to find a better basis.

Using the vector l_k in D, the algorithm projects the zonotope reachable set $\mathcal{R}(t)$ different two-dimensional planes, each of which is spanned by the normal vector of the guard hyperplane w_h in (4.1) and one of the basis vectors l_k for $k = 1, ..., n-1$. The projection operator $\Pi(w_h, l_k)$ [15] is a linear transformation on the trajectory form $x(t)$ in (3.5) defined by

$$\Pi_{w,l_k}(x(t)) = (w \cdot x(t), l_k \cdot x(t)). \tag{4.8}$$

Note that the resulting dot product of vectors w, l_k remains with the same form of the reachable set due to linearity.

Then, the procedure getIntersect2D() computes the intersection of the zonotope reachable set $\mathcal{R}_{\bar{i}}(t_{s,\bar{j}})$ for $\bar{i} = 1, ..., N_s$ sampled at each time step $t_{s,\bar{j}}$ in (4.5) for $\bar{j} = 0, ..., N_s$ using the projections on zonotopes $\Pi_{l_k,w}(\mathcal{Z}_{\bar{i}\bar{j}})$ in (4.8) as :

$$\tilde{\mathcal{Z}}_{\bar{i}\bar{j}k}(t) = (\Pi_{w_h,l_k}(c(t)), \langle(\Pi_{w_h,l_k}(g_1(t)), \Pi_{w_h,l_k}(g_2(t)), ...\rangle) \tag{4.9}$$

where $k = 1, ..., (n-1)$ and guard hyperplane $\mathcal{G}_h$ is also projected in the same way, resulting in the line $x = \gamma_k = b/|w_h|$ in the k-th projected plane as shown in Fig. 4.3. Computing intersection in the projected plane is reduced into simple two-dimensional intersection problems that can be solved in an algebraic way. Given a line segment $\overline{v_1 v_2}$ in a 2-D plane, its intersection with the line $x = \gamma_k$ is

$$y = (\gamma_k - v_{1x})\frac{v_{2y} - v_{1x}}{v_{2x} - v_{1x}} + v_{1y} \tag{4.10}$$

where $v_1 = (v_{1x}, v_{1y})$ and $v_2 = (v_{2x}, v_{2y})$. The procedure getIntersect2D() for zonotope initially set the first point $v_1 = c + g_0$ by picking the generator g_0 so that v_1 is most close to the guard line $x = \gamma_k$; and then, the next point is iterated adding $2g_{next}$ as:

$$\begin{aligned} v_{next,1} &= v_2 \\ v_{next,2} &= v_2 + 2g_{next} \end{aligned} \tag{4.11}$$

where g_{next} is picked from the first element in the sorted generators $g = (g_x, g_y)$'s in the tangent order (i.e. $arctan(g_x/g_y)$) and this process ends when the sorted set becomes empty. This iteration is performed in two ways, upper and lower side of vertices of the zonotope $\tilde{\mathcal{Z}}_{\bar{i}\bar{j}k}(t)$ until finding two intersecting values $y = M$ and $y = m$, respectively. This pair of values indicates the range of scaling factor $[m, M]_{\bar{i}\bar{j}k}$ applied to basis vector l_k. Connecting them with the vector l_k into a $(n-1)$-dimensional hypercube yields the approximate intersection of the exact $\mathcal{R}_{\bar{i}}(t_{s,\bar{j}}) \cap \mathcal{G}$ lying on the guard hyperplane $\mathcal{G}_h$.

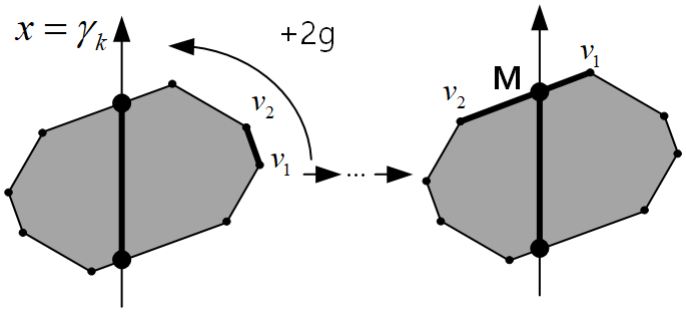
When computing the guard intersections of multiple reachable sets $\mathcal{R}_{\bar{i}}(t)$, sampled zonotopes $\mathcal{Z}_{\bar{i}\bar{j}}$ shares same direction l_k at time $t = t_{s,\bar{j}}$ in the guard-crossing time interval $[t_1, t_2]$. The procedure *mergeSegments*() combines the range segments $[m, M]_{\bar{i}\bar{j}k}$'s for the same sampling instant $t = t_{s,\bar{j}}$ and the same basis vector l_k. This reduces the resulting $N_s \times N_s$ zonotopes representing the approximate guard intersections to N_s, yielding the guard intersection by a range of segments

$$\mathcal{I} = \{[m, M]_{\bar{j}k} \mid \bar{j} = (1, ..., N_s), k = (1, ..., n-1)\}. \tag{4.12}$$

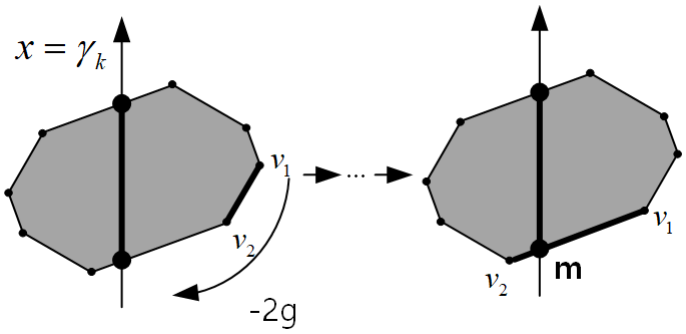
This procedure prevents the exponential growth of the number of reachable sets, keeping it at N_s.

4.3.3 Computing Reachable Sets in New Sub-Regions

Algorithm 3 lists the algorithm for computing the N_s subsequent reachable sets $\mathcal{R}_{\mathcal{I}}(t)$ from the $N_s + 1$ guard intersections $\mathcal{I}$ of (4.12). Note that these intersections are



(a) upper side



(b) lower side

Figure 4.5: The procedure *getIntersect2D* for computing the range of segment $[m, M]$ in a (w,l_k)-plane.

Algorithm 2: Computing the guard intersection.

Input : $\mathcal{Z}_0 = \{\mathcal{Z}_{\bar{i}} \mid \bar{i} = 1, ..., N_s\}$, $\mathcal{G}_h = (w_h, b_h)$

Output: $\mathcal{I} = \{[m, M]_{\bar{j}k} \mid \bar{j} = (0, ..., N_s), k = (1, ..., n-1)\}, t_s$

$\mathcal{R}(t), \mathcal{I}_0 \leftarrow \emptyset$;

for $\bar{i} \in (1, ..., N_s)$ **do**

  $\mathcal{R}_{\bar{i}}(t) \leftarrow getTrajectoryZonotope(p, \mathcal{Z}_{\bar{i}})$;

  $\mathcal{R}(t) \leftarrow \mathcal{R}(t) \cup \{\mathcal{R}_{\bar{i}}(t)\}$;

end

$[t_1, t_2] \leftarrow findCrossTimeInterval(\mathcal{R}(t))$;

$t_{s,\bar{j}} \leftarrow \{(t_1 + \bar{j}\frac{(t_2 - t_1)}{N_s})_{\bar{j}=0,...,N_s}\}$;

$D \leftarrow getOrthogonalBasis(\mathcal{G}_h, \mathcal{R}(t))$;

for $\bar{i} = (1, ..., N_s) \wedge \bar{j} = (0, ..., N_s) \wedge k = (1, ..., n-1)$ **do**

  $\mathcal{Z}_{\bar{i}\bar{j}} \leftarrow sample(\mathcal{R}_{\bar{i}}(t), t_{s,\bar{j}})$;

  $[m, M]_{\bar{i},\bar{j},k} \leftarrow getIntersect2D(\mathcal{Z}_{\bar{i}\bar{j}}, l_k)$;

  $\mathcal{I}_0 \leftarrow \mathcal{I}_0 \cup \{[m, M]_{\bar{i}\bar{j}k}\}$;

end

$\mathcal{I} \leftarrow mergeSegments(\mathcal{I}_0)$ ▷ `Merge` $[m, M]_{\bar{i}\bar{j}k}$`'s having same` $\bar{j}, k$

return $\mathcal{R}(t), \mathcal{I}, t_s$

crossing the guard $\mathcal{G}_h$ at different times of $t_{s,\bar{j}}$ for $\bar{j} = 1, ..., N_s$ and each intersection is expressed with a set of $n-1$-fold range segments along the directions of the orthogonal basis vectors l_k's for $k = 1, ..., n-1$ of (4.6). The goal is to compute a set of N_s reachable sets in the trajectory form (3.16) valid from the last time of the guard intersection $t = t_2$.

First, the algorithm combines each pair of range segments $[m, M]_{\bar{j}-1,k}$ and $[m, M]_{\bar{j}k}$ with each l_k of (4.12) at two adjacent times, $t_{\bar{j}-1}$ and $t_{\bar{j}}$ into a zonotope. In advance, let's define $\mathcal{Z}_\mathcal{G} = (c, \langle l_1, ..., l_{n-1} \rangle)$ as the zonotope spanning the entire hyperplane of the guard $\mathcal{G}_h$ where $c = |b_h| w_h \in \mathcal{G}_h$. It is noteworthy that every states from the guard $\mathcal{G}$ in the new sub-region $S_{p'}$ is included in the $\mathcal{Z}_\mathcal{G}$; thus, with the range segments (4.12) and $\mathcal{Z}_\mathcal{G}$, one can sample any set of states $\tilde{\mathcal{Z}}_{\bar{j}}$ by the new procedure *I2Z*(), which converts each set of segments $[m, M]_{\bar{j}k}$ at time $t_{s,\bar{j}}$ into a zonotope

$$\tilde{\mathcal{Z}}_{\bar{j}} = (c, \langle g_1, ..., g_{n-1} \rangle) \tag{4.13}$$

where its center and generators are

$$\begin{aligned} c &= w_h + \sum_k (m_k + M_k) l_k / 2, \\ g_k &= (M_k - m_k) l_k / 2 \text{ for } k = 1, ..., n-1. \end{aligned} \tag{4.14}$$

Since we selected the basis l_k orthogonal to each other, the resulting zonotope becomes a hypercube. The procedure $\widehat{IH}()$ in [14] computes a zonotope that tightly encloses the two zonotopes P and Q using the following equation [15] as :

$$\begin{aligned} \widehat{IH}(P, Q) = 0.5(c_P + c_Q, \langle g_{P,1} + g_{Q,1}, ..., g_{P,i} + g_{Q,i}, \\ g_{P,1} - g_{Q,1}, ..., g_{P,i} - g_{Q,i} \rangle) \end{aligned} \tag{4.15}$$

Combining pairs of intermediately generated zonotopes $\mathcal{Z}_l = \tilde{\mathcal{Z}}_{\bar{j}}(t = t_{s,\bar{j}-1})$ and $\mathcal{Z}_u = \tilde{\mathcal{Z}}_{\bar{j}}(t = t_{s,\bar{j}}$ using $\widehat{IH}()$ returns $\mathcal{Z}_{\bar{j}}$ for each $\bar{j} = 1, ..., N_s$.

Finally, from the resulting N_s zonotopes, the subsequent reachable set valid from $t = t_2$ can be computed using the procedure getTrajectoryZonotope() with the new set of matrices $A_{p'}, B_{p'}, C_{p'}, D_{p'}$ in the next sub-region.

Algorithm 3: Computing the subsequent reachable set from the guard intersection.

Input : $\mathcal{I} = \{[m, M]_{\bar{j}k} \mid \bar{j} = (0, ..., N_s), k = (1, ..., n-1)\}$

Output: $\mathcal{Z}_0 = \{\mathcal{Z}_1, ..., \mathcal{Z}_{N_s}\}, t_r$

$\mathcal{R}_{\mathcal{I}}(t), \mathcal{Z}_0 \leftarrow \emptyset$; $\mathcal{Z}_{\mathcal{G}} \leftarrow \langle c, D \rangle$;

$\mathcal{R}_{\mathcal{G}}(t) \leftarrow getTrajectoryZonotope(p, \mathcal{Z}_{\mathcal{G}}, u_j(t))$;

for $\bar{j} \in (1, ..., N_s)$ **do**

  $t_{r,\bar{j}} \leftarrow t_2 - t_1 - t_{s,\bar{j}}$ ▷ `Remaining time until` T `from` $t_{s,\bar{j}}$

  for $k \in (1, ..., n-1)$ **do**

    $[m, M]_k \leftarrow max([m, M]_{\bar{j}-1,k}, [m, M]_{\bar{j},k})$;

  end

  $\mathcal{Z}_l \leftarrow I2Z(\mathcal{R}_{\mathcal{G}}(t_{r,\bar{j}-1}), \{[m, M]_{k=(1,...,n-1)}\})$;

  $\mathcal{Z}_u \leftarrow I2Z(\mathcal{R}_{\mathcal{G}}(t_{r,\bar{j}}), \{[m, M]_{,k=(1,...,n-1)}\})$;

  $\mathcal{Z}_{\mathcal{I}} \leftarrow \widehat{IH}(\mathcal{Z}_l, \mathcal{Z}_r)\}$;

  $\mathcal{R}_{\mathcal{I},\bar{j}}(t) \leftarrow getTrajectoryZonotope(j, \mathcal{Z}_{\mathcal{I}}, u_j(t))$;

  $\mathcal{R}_{\mathcal{I}}(t) \leftarrow \mathcal{R}_{\mathcal{I}}(t) \cup \mathcal{R}_{\mathcal{I},\bar{j}}(t)$;

  $\mathcal{Z}_0 \leftarrow \mathcal{Z}_0 \cap sample(\mathcal{R}_{\mathcal{I}}(t), t_{r,\bar{j}})$;

end

return $\mathcal{R}_{\mathcal{I}}(t), \mathcal{Z}_0, t_r$

Note that in our reachability analysis method, we do not include the time variable t in a reachable set unlike the other continuous variable $x(t)$ and changes its discrete states $m(t)$ depending on the external pulse control; therefore, often the system changes its discrete state $m(t)$ independent of the state $x(t)$ while the reachable set $\mathcal{R}(t)$ is crossing the guard $\mathcal{G}$, as shown in Fig. 4.6, i.e. $t_1 < T < t_2$ where T is the maximum time bound assumed by the procedure *findCrossZonotope*. In this case, the resulting reachable set $\mathcal{R}(T)$ is divided into two sets $\mathcal{R}_1$ for $t_{\bar{j}=T}$ and $\mathcal{R}_2$ for $t = [t_1, T]$ contained in distinguished sub-regions that have different discrete state $m(t)$. After computing the latter set in the new sub-region $S_{p'}$, it returns the union of the two sets. To prevent exponential growth of the number of sets, the total num-

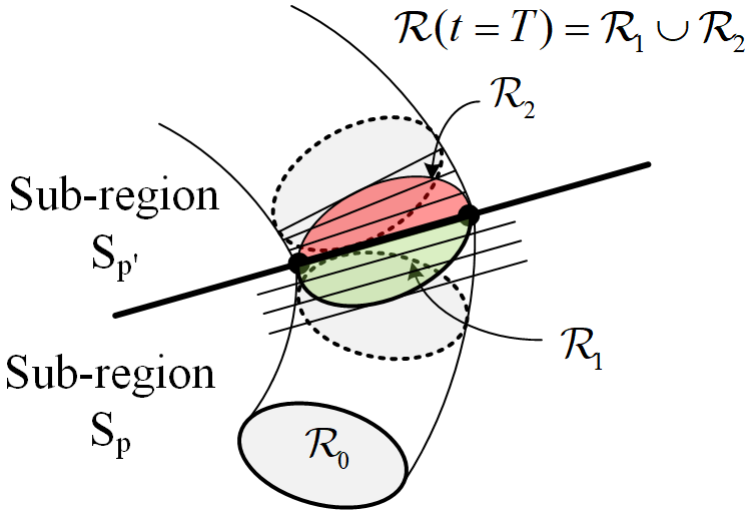


Figure 4.6: Case when external discrete state $m(t)$ changes its state at $t = T$ while the reachable set $\mathcal{R}(t)$ is crossing in $t = [t_1, t_2]$.

ber of two sets is kept at N_s depending on the ratio to the distances from the guard hyperplane.

4.4 Time Complexity Analysis

4.4.1 Trajectory Form Computation

We assume that the procedure to obtain the transfer function $H(s) = (sI - A)^{-1}$ has the complexity of $O(1)$ since we reuse the previously computed values in advance. Then The procedure *getTrajectoryZonotope*() has the complexity of $O(n^2 r)$ where n is the number of state dimensions and r is the number of generators of the zonotope reachable set $\mathcal{R}(t)$.

Proof. Given an initial zonotope set $\mathcal{Z}$ with vector c and r generators g's, of which each vector has n scalar components, then, it requires $(r+1)$-fold evaluation of trajectory form in (3.5) that incur $n \times n$ scalar multiplication of single-input-single-output (SISO) transfer functions $H_{i'j'}(s)$ and inputs $u'_i(t) \in \mathbb{R}$ and initial condition $x_{0,j'} \in \mathbb{R}$

as in (3.12) as:

$$x'_j(s) = (sI_{i'j'} - A_{i'j'})^{-1} B'_i U'_i(s) + (sI_{i'j'} - A_{i'j'})^{-1} x'_i(0) \tag{4.16}$$

for $i', j' = 1, ..., n$. This results in complexity of $O((r+1) \times n^2) = O(n^2 r)$. □

4.4.2 Guard Intersection Compuatation

Given trajectory form reachable set $\mathcal{R}(t)$ consisting of N_s subsets and a guard hyperplane $\mathcal{G}_h$ in (4.1), the complexity of computing the guard intersection $\mathcal{R}(t) \cap \mathcal{G}_j$ have complexity of $O(N_s^2 n)$.

Proof. The algorithm that computes guard intersection first computes the projection of zonotope $\mathcal{Z}_{\bar{i}} \in \mathbb{R}^{(r+1)\times n}$ at $N_s + 1$ time steps segmenting $[t_1, t_2]$, and then computes the two-dimensional intersection of the zonotopes $\mathcal{Z}_{\bar{i}}$ in the k-th plane for $k = 1, ..., (n-1)$, incur $(N_s + 1) \times N_s$-fold 2-D intersection procedure *getIntersect2D* that is pure algebraic operation of constant complexity $O(1)$ in (4.10). Therefore, the resulting complexity is $O(N_s^2 n)$. □

4.4.3 Reachable Set Computation from Guard Intersection

Given $\mathcal{I} = \{[m, M]_{\bar{j},k} \mid \bar{j} = (0, ..., N_s), k = (1, ..., n-1)\}$, computing the reachable set $\mathcal{R}_{\mathcal{I}}(t)$ have a complexity of $O(N_s n^2 r)$.

Proof. The algorithm 3 first combine the pair of two scalar value m and M in $[m, M]_{\bar{i}\bar{j}k}$ for $k = 1, ..., n-1$ having same $i, \bar{j}$ indices into N_s zonotopes, which is a series of $N_s \times (n-1)$-fold scalar interval arithmetic operations. After that, it computes the trajectory form reachable set $\mathcal{R}_{\mathcal{I},\bar{j}}(t)$ from each resulting zonotope $\mathcal{Z}_{\mathcal{I},\bar{j}}$ by *getTrajectoryZonotope*(). Therefore, overall complexity adding two procedures is given by $O(N_s(n-1) + N_s n^2 r) = O(N_s n^2 r)$ □

4.4.4 Overall Complexity

The overall complexity for the entire algorithm is $O(N_s^2 + N_s n^2 r)$, where N_s is the number of time steps discretizing the guard intersecting time range $t = [t_1, t_2]$ and is independent of the number of dimension n. So if we keep N_s small compared to n and limit the number of generators r to scale linearly with n using the dimensional reduction technique in [14, 15], the overall complexity becomes $O(n^3 N_s)$.

4.5 Computing Safety Bounds from Reachable Sets in Trajectory Form

To verify the safety of the circuits using the computed reachable sets, one needs a way to compute the range of the reachable set $\mathcal{R}(t)$ along an arbitrary direction $l \in \mathbb{R}^n$. In [16], it has been shown that such a range for time t can be computed using the *support function* $\rho_{\mathcal{R}(t)}(l)$:

$$\rho_{\mathcal{R}(t)}(l) = c(t) \cdot l + \sum_{j=1}^{r} |g_j(t) \cdot l|. \tag{4.17}$$

We need to evaluate the peak values (min/max) for a trajectory-form reachable set $\mathcal{R}(t)$ for a time range $t = [0, T]$, calls for two scalar optimization procedures with respect to time t

$$bounds_l(x(t), t = [0, T]) = [\min_{t=[0,T]} \rho_{\mathcal{R}(t)}(l), \max_{t=[0,T]} \rho_{\mathcal{R}(t)}(l)]. \tag{4.18}$$

Note that this function calls for a piecewise evaluation with the knowledge when the polarities of $g_j(t) \cdot l$ change. With g_j's expressed in the analytical, trajectory form of (3.5), one can find the values of t that satisfy $g_j(t) \cdot l = 0$ and subsequently find the minimum and maximum values of (4.17) for each time interval split by the solutions of t. Finally, by combining these results, the minimum and maximum bounds of $\mathcal{R}(t)$ along the direction of l for $t \in [0, T]$ can be found.

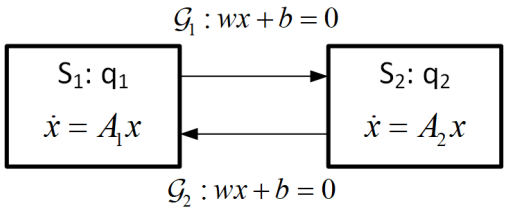


Figure 4.7: Hybrid automata of the LC oscillating system.

4.6 Benchmark: Numerical Example

This section demonstrates the proposed reachability analysis method with an illustrative linear hybrid oscillator system with associated guard hyperplanes $\mathcal{G}_1 = \mathcal{G}_2 = \mathcal{G}$ as:

$$\mathcal{G} : w \cdot x + b = 0 \tag{4.19}$$

where $w = (0.1, 1)$ and $b = -1$, which splits the continuous state space into two disjoint sub-regions at the guard hyperplane, each linear system

$$\dot{x} = A_i x \tag{4.20}$$

with the system matrix A_i of two discrete state q_1, q_2 for $i = 1, 2$ is given by :

$$A_1 = \begin{pmatrix} 1 & -10 \\ 10 & 0 \end{pmatrix}, \; A_2 = \begin{pmatrix} 2 & -20 \\ 20 & 0 \end{pmatrix}. \tag{4.21}$$

The hybrid automata in Fig. 4.7 models the behavior of the discrete state $d(t) = \{q_1, q_2\}$ and the continuous state $x(t) \in \mathbb{R}^2$. It switches its discrete states when $x(t)$ intersects the guard line $\mathcal{G}$ in (4.19).

In this section, we compare the reachable sets computed with two different algorithms, including proposed algorithm, Girard's algorithm in [14, 15, 32] with two references, reachable set $\mathcal{R}_{ref}$ using Delaunay triangulation (DT) $\mathcal{R}_{DT}$ and brute-force Monte-Carlo (MC) $\mathcal{R}_{MC}$. The reachable set using DT $\mathcal{R}_{DT}$ is to evaluate the

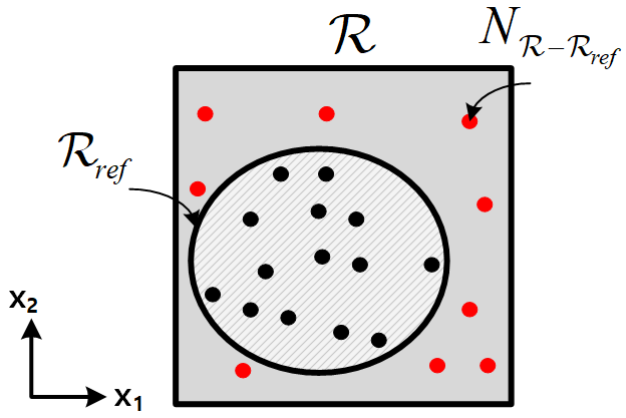


Figure 4.8: Error measure err_1 using MC integration method.

over-approximation error of the proposed method caused by the zonotope approximation since DT can accurately represent the arbitrary shapes with a fine set of triangles, which also can be computed by the trajectory form reachable set in (3.7). In addition, the brute force MC method is widely used for accounting the range of initial conditions and provides a reference for evaluating the accuracy of the reachable set $\mathcal{R}_{MC}$. Note that the results obtained from the envelope of the MC method are under-approximation of the theoretically exact reachable set, but when the number of random samples is large enough and the number of system dimensions is relatively small, then we can roughly evaluate the relative accuracy of the different reachability analysis algorithm compared to it.

4.6.1 Error Measures

We defined an error measure to compare the accuracy of reachability analysis algorithms err_1 that estimates the over-approximation error at specific sample time t. The error measures the area difference between the approximation and the exact set. The over-approximation error err_1 compared to the reference set is evaluated by

$$err_1 = \frac{Area(\mathcal{R} - \mathcal{R}_{ref})}{Area(\mathcal{R})} \tag{4.22}$$

where the reachable set $\mathcal{R}$ and $\mathcal{R}_{Ref}$ refers the set sampled at the specific sampling instant $t = t'$. However, computing the exact area (or volume) of an arbitrary polygonal intersection is a computationally expensive operation with many sets in a large-scale system so we introduced a simple Monte-Carlo integration method that computes the area (volume) of the difference set using random sampling. It generates random samples in a hypercube enclosing two sets and counts the number of samples included in the target region. Fig. 4.8 illustrates the described error evaluation method. The area of the target reachable set $\mathcal{R}(t)$ can be approximated by the number of random point $N_{\mathcal{R}(t)}$ and the over-approximation error can be estimated by :

$$err_1 \approx \frac{N_{(\mathcal{R}-\mathcal{R}_{ref})}}{N_{\mathcal{R}}} \tag{4.23}$$

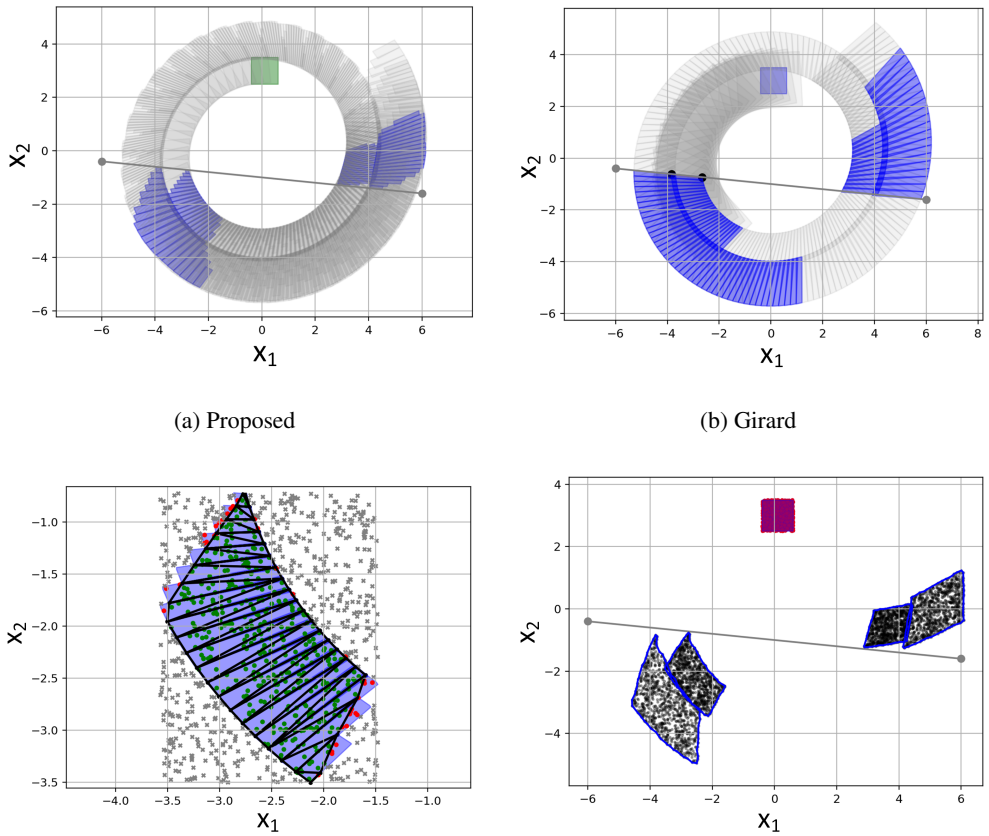
where $N_{\mathcal{R}}$ refers to the number of sample in the set $\mathcal{R}$.

4.6.2 Comparison of accuracy and runtime

The proposed method can compute the accurate reachable set after the guard intersection compared to the classic reachable set in [14, 15, 32], which assumes that every continuous state $x(t)$ starts from the guard intersection starts at the same time, adding large over-approximation each time guard intersection occurs.

Fig. 4.9 compares the proposed method and the classic reachability algorithm (Girard) with two reference sets $\mathcal{R}_{DT}$ and $\mathcal{R}_{MC}$. It can be observed that comparing the proposed reachable $\mathcal{R}(t')$ set at each guard intersection ($t' = t_2$ at every four cycles) agrees well with the reference sets in Fig. 4.9 (c) DT (Delaunay triangulation) and (d) Monte-Carlo (MC). While the classic method can enclose the states at the instants t''s, it is shown that large approximation errors are added each time with cycles.

Fig. 4.10 shows the over-approximation at the instant of the guard intersection as increasing the number of cycles. Fig. 4.10 (a) compared the reachable sets obtained from the proposed and Girard's one with the reference set from DT $\mathcal{R}_{DT}(t)$, as increasing the number of time segmentation $t_{s,\bar{j}}$ in the guard intersecting interval $[t_1, t_2]$



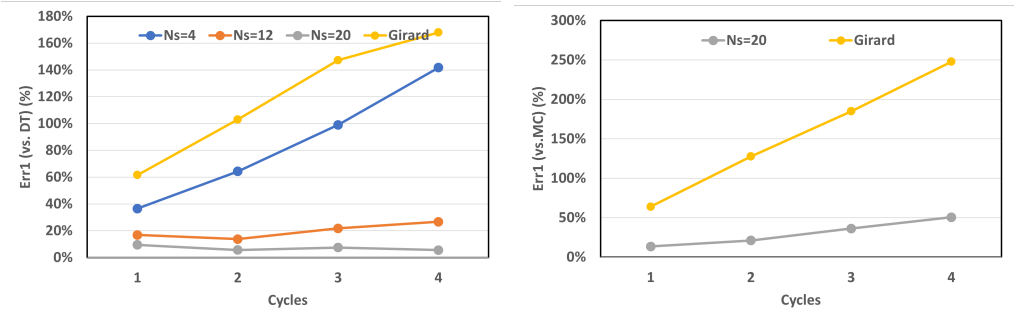
(a) Proposed

(b) Girard

(c) Delaunay triangulation (DT)

(d) Monte-Carlo (MC)

Figure 4.9: Reachable sets computed for a 2-D linear hybrid system example with a guard hyperplane.



(a) err_1 vs. DT

(b) err_1 vs. MC

Figure 4.10: Increasing err_1 as the number of cycles (guard intersections).

in (4.5). As expected, increasing the number of time steps of the range results in a smaller error, where the err_1 of $5.06\,\%$ with $N_s = 20$ leads to $25.3\times$ more accurate than $141.7\,\%$ of $N_s = 4$. In addition, even with $N_s = 4$ time steps, the proposed method achieves $18\,\%$ improvement of accuracy than $168\,\%$ of the Girard's.

Furthermore, compared with the MC reference set $\mathcal{R}_{MC}(t)$ with the same configurations, the proposed algorithm achieved a $30\times$ improvement with err_1 of $0.506\,\%$ while the Girard's algorithm has $2.477\,\%$ as shown in Fig. 4.10 (b). The error err_1 increases slowly with the number of cycles while the error of the Girard's increases fast with cycles.

Then, we evaluated the accuracy of safety bounds defined by evaluating (4.17) for each axis direction and runtimes compared to the SpaceEx, providing performance reference for the different reachability analysis algorithms. As shown in Fig. 4.11, we defined new error measure err_2, comparing the range $[x_{i,min}, x_{i,max}]$ of each state value $x_i(t)$, estimated by applying (4.17) for x_i-direction on the reachable sets $\mathcal{R}(t)$ and $\mathcal{R}_{MC}(t)$ as :

$$err_2 = \frac{1}{n}\sum_{i=1}^{n} \frac{|\hat{x}_{i,max} - x_{i,max}| + |\hat{x}_{i,min} - x_{i,min}|}{x_{i,max} - x_{i,min}} \tag{4.24}$$

where $[x_{i,min}, x_{i,max}]$ is the range of state variable $x_i(t)$ computed by the reference set $\mathcal{R}_{MC}(t)$, $\mathcal{R}_{DT}$ and $[\hat{x}_{i,min}, \hat{x}_{i,max}]$ is the range of the reachable set $\mathcal{R}(t)$ computed

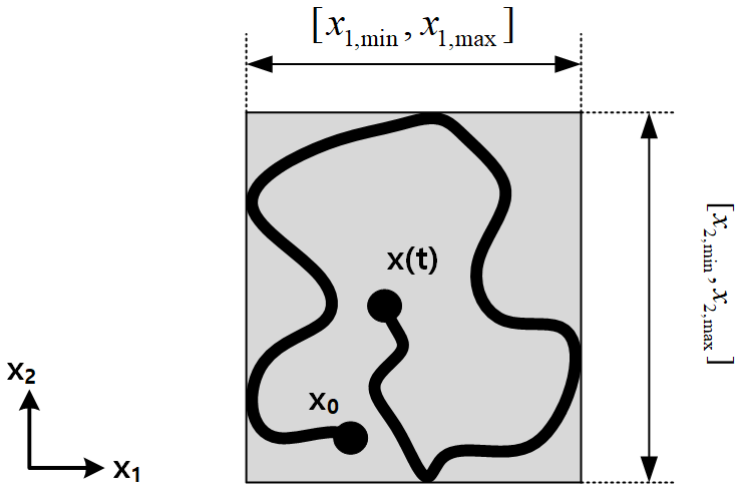


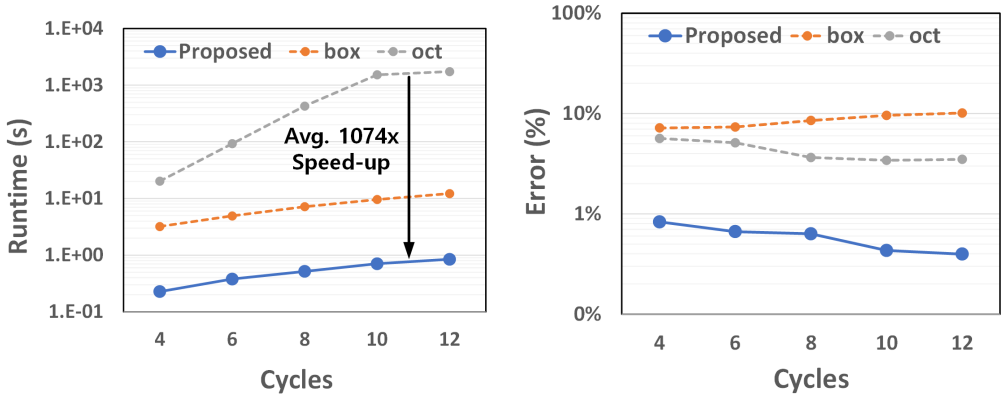
Figure 4.11: Safety bound of state $x(t)$.

with each algorithm with different options that configure the set shape, *box* (box) and *octagon* (oct), the default time step $T_s = 0.01\,\mathrm{s}$, and error tolerance for computing reachable set $\epsilon = 0.01$.

Fig. 4.12 shows that the runtime of the proposed algorithm increases linearly with the number of cycles since it keeps a fixed number of sets with cycles. Computing the safety bounds over given 12 cycles, the proposed algorithm shows a 13.8× average speed-up compared to the less accurate 'box' option yielding the err_2 of $8.55\,\%$, and a 1074× speed-up compared to the accurate but slow 'oct' option yielding $4.26\,\%$ err_2 in average. On the other hand, the proposed algorithm keeps the error below $1\,\%$ for the entire cycles. Note that the relative error metric err_2 decreases with guard intersection cycles, mainly due to the increase of the radius of the reachable set while absolute error stays almost constant with cycles.

4.7 Conclusion

In summary, we concluded that the proposed method using the trajectory form reachable set computation outperforms over 1074× speed-ups compared to the existing



(a) Runtime

(b) Error

Figure 4.12: Runtimes and error comparison vs. SpaceEx algorithms (err_2).

method relying on time discretization reachable set computation by suppressing exponentially increasing number of reachable sets. Simultaneously, it achieved better accuracy below $1\,\%$ than the time discretization method due to its nature of not requiring more fine time steps to lower the required error tolerance. These achievements have been demonstrated with a simple numerical benchmark example but it represents a wide range of circuits, whereby, in the next chapter, we demonstrate the safety of practical circuits with the proposed methodology.

Chapter 5

SOA VERIFICATION OF DC-DC BUCK CONVERTERS

In this chapter, we demonstrate that the proposed trajectory form methodology explained in Chapters 3 and 4, can verify the safety of switching power supply circuits in a fast and accurate way. A switching power supply operates as a switched linear system and its switching activities are controlled by input pulses applied to its switches, unintentionally causing abrupt changes in the internal inductor current and leading to malfunctions or permanent damage to the system.

The overall flow of the safety verification procedures is listed in Fig. 5.1. This can be done in three steps: First, the circuit topology is converted into a set of differential-algebraic equations (DAEs). Secondly, the matrices from DAEs are then converted into the hybrid automata representation M with the set of matrices F, the set of discrete transitions E, and the associated guard $\mathcal{G}$ with each element of E. Secondly, its reachable set $\mathcal{R}(t)$ is computed using the proposed method. Finally, the bounds of reachable sets are compared to the given specification for safety in ranges of operating voltages and currents for each circuit element.

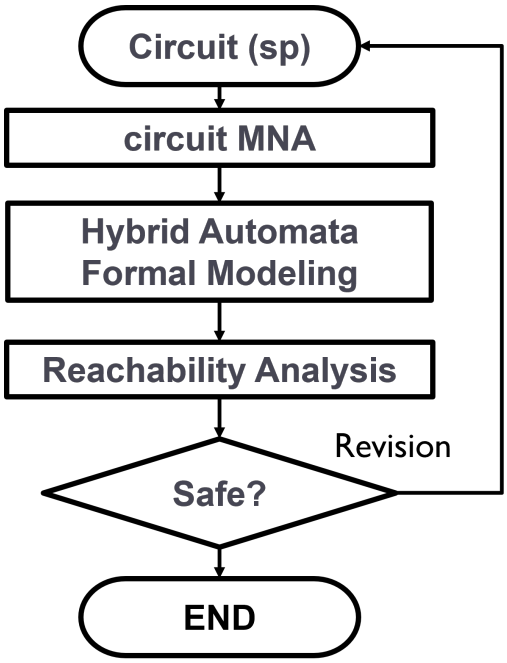


Figure 5.1: Overall flow of safety verification methodology for a given circuit.

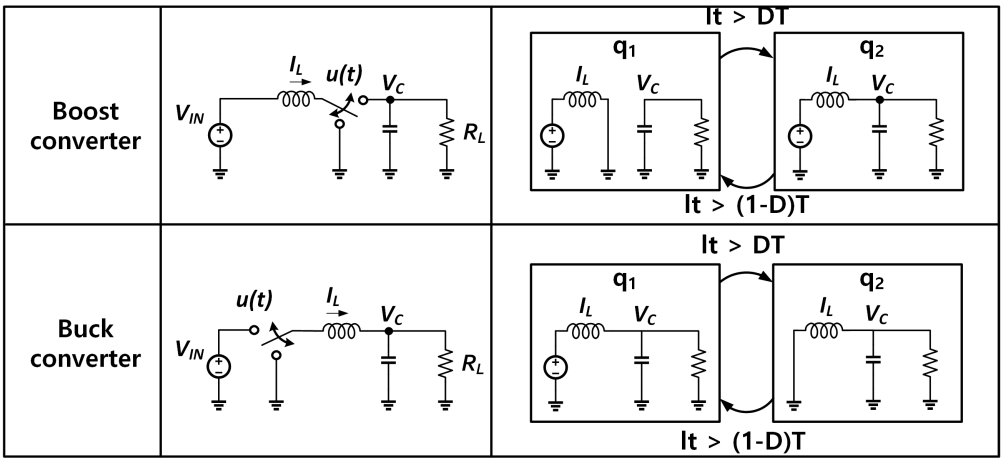


Figure 5.2: Switching power supplies operating as switched-linear systems.

5.1 SOA Verification of DC-DC Buck Converters

Fig. 5.2 shows the two representative basic circuit topologies mainly used for DC-DC converters. These two converters are distinguished depending on whether the output voltage is greater than the input voltage V_{IN} or not. However, the operating principle is similar so we focus on analyzing the buck converter.

A DC-DC buck converter topology is used for step-down of the input voltage V_{IN} by regulating its voltage by modulating the duty cycle of the switch control input $u(t)$. When the switch is turned on, the input voltage V_{IN} charges the inductor current, on the contrary, when the switch is turned off, the inductor current discharges. The steady-state value of output voltage V_C gets proportional to the duty-cycle D of the input $u(t)$, i.e., $V_C \approx D \cdot V_{IN}$.

5.2 Open-Loop Verification with PWM Control

Fig. 5.3 shows the hybrid automata model for DC-DC buck converter circuits under open-loop duty control, i.e., the input for the MOSFET switch $u(t)$ has fixed duty cycle D. The automata model involves three modes, q_1, q_2, and q_3, depending on the state of switches. The model includes the generation of $u(t)$ in the automata with the period T using an additional state variable δ. Depending on the value of local time $\delta \in [0, T]$ where T is the pulse duration of a clock period, the discrete state changes its state between q_1 and q_2 (or q_3).

While the state of the MOSFET switch is directly set by the control input $u(t)$, i.e. δ in the automata, the switch state of the diode depends on the internal circuit states, i.e., the voltage across the diode V_D. The state space of the circuit has two independent variables $I_L(t)$ and $V_C(t)$, and the diode voltage V_D is a linear sum of those variables, yielding $V_D = -I_L R_{OFF}$ in Fig. 5.3. Therefore, when the inductor current I_L at the instant when the MOSFET switches off is positive, the diode turns on with a negative V_D, and I_L stays positive. Otherwise, the diode turns off with positive V_D, and the

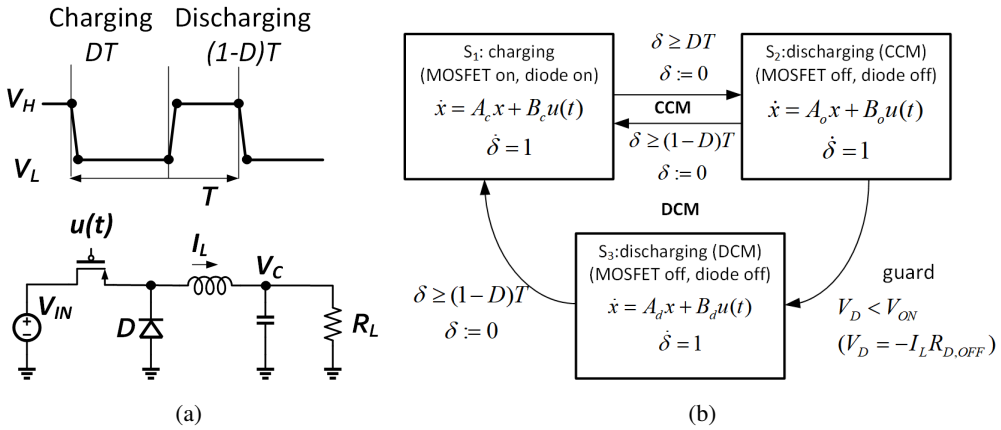


Figure 5.3: DC-DC buck converter operating in DCM.

current I_L becomes zero. The former operation is called continuous conduction mode (CCM) and the latter is called discontinuous conduction mode (DCM). This chapter addresses a way to verify the safety of the practical circuits using the methodology explained so far.

Each switching behavior of CCM operation ($q_1 \leftrightarrow q_2$) and DCM operation ($q_1 \rightarrow q_2 \rightarrow q_3 \rightarrow q_1 ...$) has different impacts on its computational time due to the way the reachable set $\mathcal{R}(t)$ intersects the guard set with time. In CCM switching mode, it is easy to compute the discrete transition of the circuit state variable $x(t) = (I_L(t), V_C(t))$ because the circuit states changes its discrete states q depending on only local time variable δ compared to the guard time DT, $(1-D)T$, independent of the internal circuit states $x(t)$. With the last sampled reachable set $\mathcal{R}(t = \delta)$, one can compute the reachable set at the next iteration, exactly.

On the other hand, when operating in DCM mode, the circuit states $V_C(t)$ and $I_L(t)$ in the reachable set $\mathcal{R}(t)$ partially changes its discrete states q with time, generating infinite small intersections until the associated guard intersection ends. These guard intersections are problematic because the required number of computations increases exponentially with the number of the guard intersection events as in Fig. 4.2. In this case, the value of the diode terminal voltage V_D set the guard $\mathcal{G} : V_D < V_{ON}$.

Table 5.1: Default Circuit Parameters

Parameter	Symbol	Value
Inductor	L	$10\,\mu\text{H}$
Capacitor	C	$20\,\mu\text{F}$
Load resistor	R	$10\,\Omega$
Input voltage	V_{IN}	$10\,\text{V}$
Switching period	T	$1\,\mu\text{s}$
Diode turn-on voltage	V_{ON}	$0.5\,\text{V}$
Diode on-resistance	R_{ON}	$1\,\Omega$
Diode off-resistance	R_{OFF}	$1\,\text{G}\Omega$

This guard condition is equivalent to the condition for the inductor current $I_L < V_{ON}/R_{D,OFF}$, represented at the guard line in the two-dimensional $I_L - V_C$ state space.

We computed the reachable sets with the circuit parameters listed in Table 5.1. Nodal analysis on the circuit yields the system matrices when the MOSFET switch is closed (q_1) and when the switch is open (q_2) state in each state while CCM, A_c for q_1 and A_o for q_2 as:

$$A_c = A_o = \begin{pmatrix} -10^5 & -10^5 \\ 5 \times 10^4 & -5 \times 10^3 \end{pmatrix}. \tag{5.1}$$

The system matrix A_d when the inductor is discharging with turned-off diode (q_3) is given by:

$$A_d = \begin{pmatrix} 0 & 0 \\ 0 & -5 \times 10^3 \end{pmatrix}, \tag{5.2}$$

The associated input matrix B_c and $B_o(B_d)$ for each mode are given by

$$B_c = \begin{pmatrix} 1 \\ 0 \end{pmatrix} B_o = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{5.3}$$

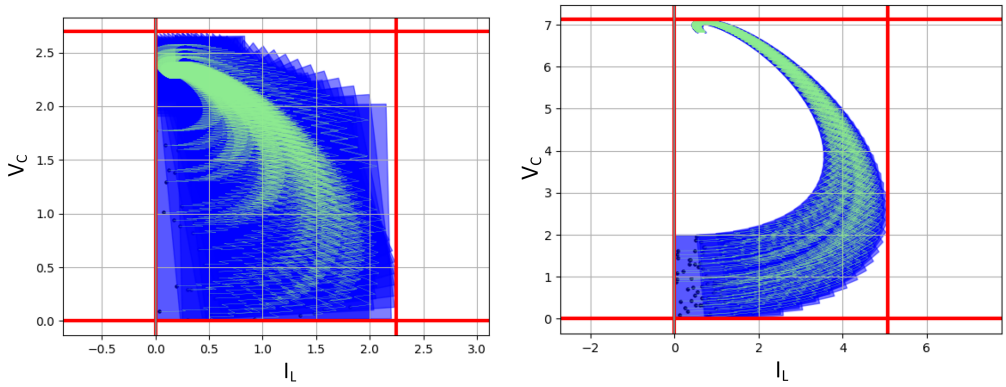
and the DC voltage source V_{IN} is converted to trajectory form by representing a step function in the s-domain :

$$U(s) = \frac{V_{IN}}{s}. \tag{5.4}$$

represented a set of coefficient $(a, c, m) = (0, V_{IN}/L, 1)$ in (3.5).

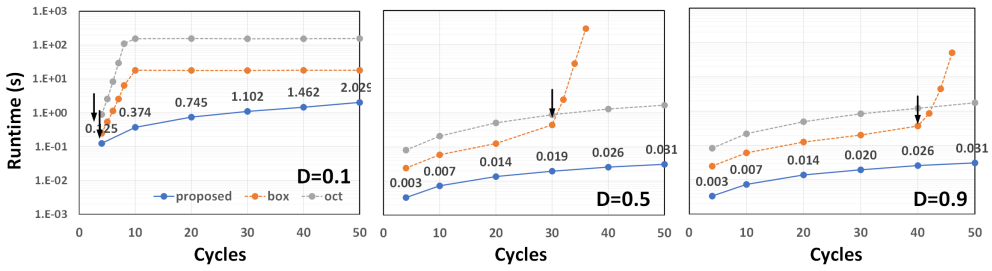
Fig. 5.4 shows the reachable sets after $100\,\mu$s, starting with the initial set of states $V_C = [0, 2]$V, $I_L = [0, 2]$A. The reachable sets obtained are represented in blue compared to those with a 1000-point MC simulation in green for two cases of duty-cycle $D = 0.25$ and $D = 0.75$, resulting in the safety bounds of $I_L = [0, 2.25]$, $V_C = [0, 2.69]$ for $D = 0.25$ and $I_L = [0, 5.06]$, $V_C = [0, 7.11]$ for $D = 0.75$, respectively. In two cases, the reachable sets computed with the proposed method show excellent agreement with trajectories of the equivalent Monte-Carlo simulations, yielding the resulting error err_2 are only $1.97\,\%$. Its runtime only took $4.967\,$s for $D = 0.25$ and $0.063\,$s for $D = 0.75$. It seems that when $D = 0.25$ the runtime abruptly increases than the other. This is because, in the case with low duty value $D = 0.25$, the circuit operates in DCM that switches its states from q_2 to q_3 when the inductor discharges, incurring massive guard intersection computations and resulting in 34-fold increased runtime than the case with low duty-cycle. On the other hand, in the case with high duty value $D = 0.75$, the buck converter operates only in CCM that switches its states only in q_1 and q_2 without any guard intersection events, resulting in fast runtime.

We evaluated the performance for various cases with varying duty $D = 0.1, 0.5, 0.9$ and the switching cycles $cycles = 4, 10, 20, 30, 40, 50$ in terms of runtime and accuracy for estimating safety (err_2), compared to the other algorithm STC of SpaceEx with different set shapes coarse 'box' and fine 'oct' shapes in the same manner as the previous chapter. Fig. 5.5 summarizes the results and comparison, showing that the proposed algorithm achieved the fastest runtimes for all cases, ranging 0.03–$2.02\,$s and the lowest average error $0.99\,\%$. The proposed method outperformed the speed-up factor was 2–$656\times$ and $107\times$ in average compared to the SpaceEx results with accurate 'oct' shapes; 7–$414\times$ and $79\times$ in average compared to those of 'box'.

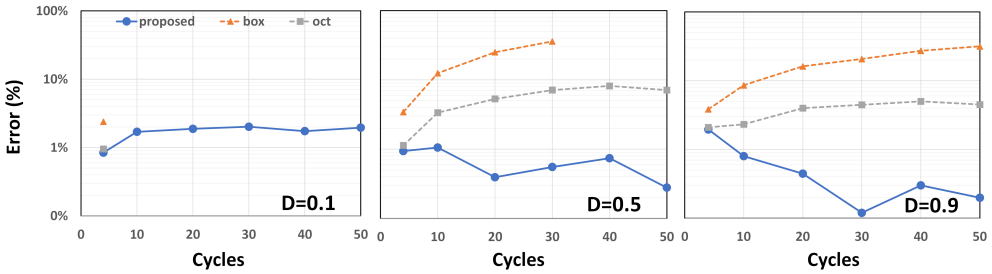


(a) D=0.25 (b) D=0.75

Figure 5.4: The reachable set computed for the DC-DC buck converter circuit.



(a) Runtime



(b) Error

Figure 5.5: Performance for open-loop operation including DCM.

Conventional time discretization-based algorithm shows the abrupt increase of the runtime at the instants where the buck converter circuit enters into DCM at $cycle > 10$ for $D = 0.1$, $cycle > 40$ for $D = 0.5$, $cycle > 50$ for $D = 0.9$. Lower duty leads to a steep increase of the runtime because the circuit operating with small duty tends to have low inductor currents, so easily reaches the guard line $\mathcal{G}$ that is equivalent to the line $I_L = 0$ at the state space.

The flat regions after the runtimes abruptly increase are because the MAX iteration limit is set to prevent infinite runtime. This is because that SpaceEx continuously keeps breaking the time steps into more fine small pieces indefinitely, causing exponentially increasing or indefinite runtimes. On the other hand, the runtimes of the proposed method scale only linearly with the number of cycles, demonstrating the effectiveness of the proposed algorithm.

5.2.1 Experimental Scalability

We also demonstrated the scalability of the proposed method with a scalable circuit, which is a DC-DC buck converter with cascaded RC loadings. Fig. 5.3 shows how the runtime increases as the number of state variables, i.e. capacitors N increases. For comparison, the results with SpaceEx are also shown. The runtime of SpaceEx scales depending on the selected algorithm LGG and STC and the complexities of the shapes 'oct' and 'box' showed different scalability.

The proposed reachability analysis algorithm has the runtime proportional to the cubic number of dimensions $O(n^2)$, whereas the SpaceEx algorithm has the same scalability with a coarse shape 'box' selected. With the more accurate shape 'oct', the runtime of the SpaceEx scales as $O(n^3)$. For example, for the circuit with 12 state variables, our algorithm achieves $8.5\times$ quicker compared to SpaceEx with STC and 'oct' shapes. It shows that the proposed trajectory form method can compute the reachable set accurate and scalable way. Note that in this comparison, the transfer function is not cached so the speed-up is relatively low than the other comparison in other sections.

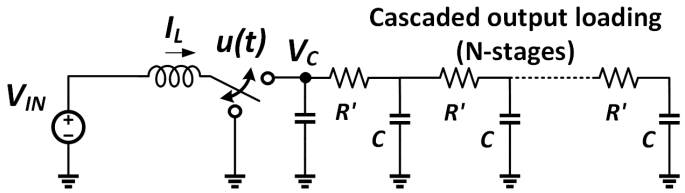


Figure 5.6: Buck converter circuit with multiple RC loadings.

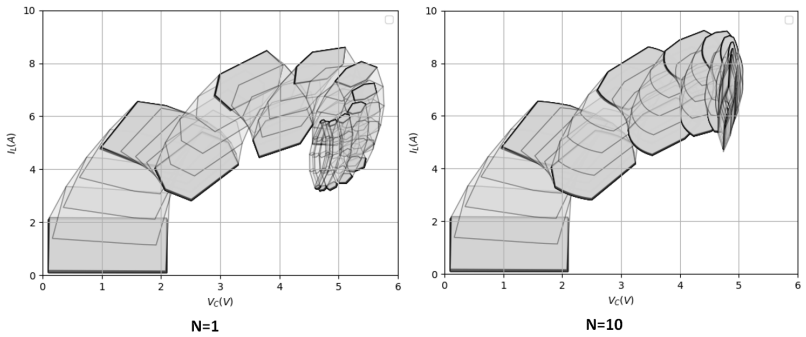


Figure 5.7: Reachable set with multiple RC loadings.

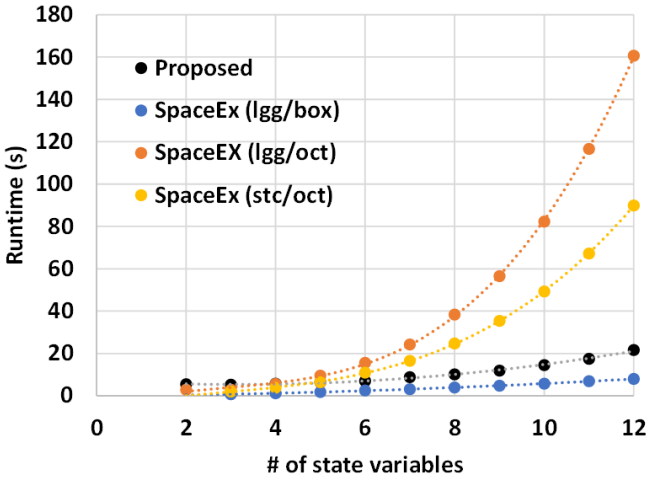


Figure 5.8: Scalability as the number of state dimensions compared to SpaceEx.

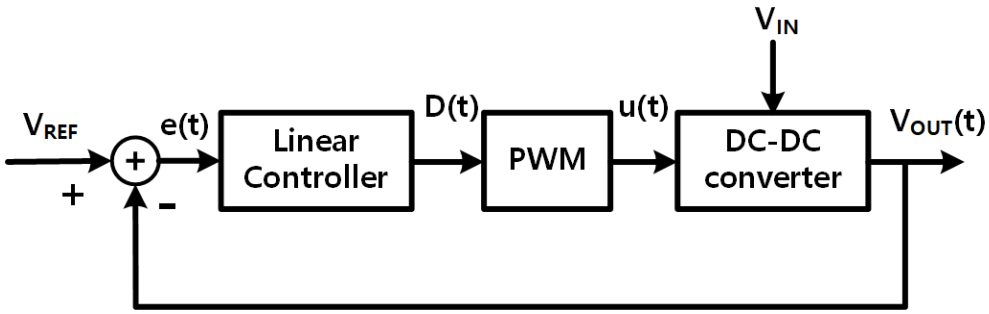


Figure 5.9: DC-DC buck converter with PWM feedback loop.

5.3 Closed-Loop Verification with PWM Control

In this section, we demonstrate our method for closed-loop controlled DC-DC buck converter examples. A practical buck converter circuit requires a feedback regulation loop that stabilizes the output voltage $V_C(t)$ at the reference voltage level $V_{REF}(t)$ against the changes of the input voltage V_{IN}, load resistance R_L, etc, which may cause unexpected large transient over-voltage issues. Unlike the previous open-loop controlled examples, we need to consider two more aspects to apply reachability analysis: i) one is the reachability algorithm requires more states to store the current states, i.e. duty values in digital or analog values, ii) the other is that the algorithm should compute the guard intersections of the current reachable states with more guard sets to get the desired duty $D(t)$ for the error $e(t)$ of the current output voltage V_C from the desired output voltage V_{REF}.

In the linear control of DC-DC buck converter, such as widely-used PID control [], the controller first measures the error compared to the reference voltage $e(t) = V_C(t) - V_{REF}$ and the duty is determined according to the transfer function $H_{PID}(s)$ and the error $e(t)$, by $D(s) = H_{PID}(s)e(s)$, whereas in digitally-controlled PWM (DPWM), the control principle is similar except that the error voltage $e(t)$ is measured at the ADC digitally and the subsequent duty values are computed using digital filters implementing the transfer function $H_{PID}(s)$. Implementing it in a reachability anal-

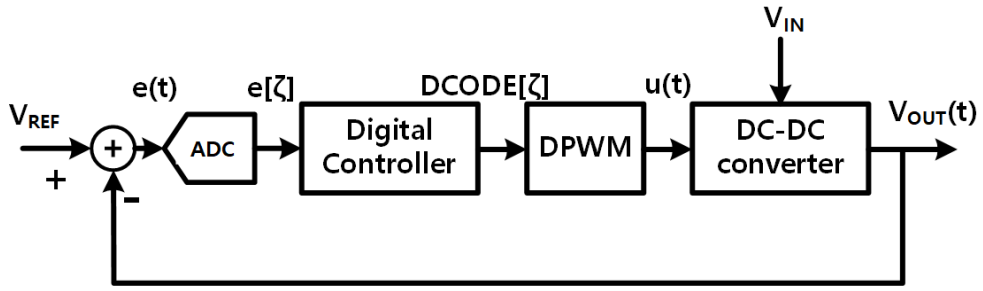


Figure 5.10: DC-DC buck converter with Digital PWM feedback loop.

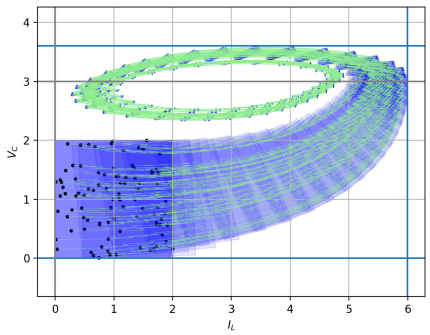
ysis algorithm requires 2^m guard sets G_{2^m} where m is the number of bits to measure discrete levels for error $e(t)$, equivalent to the number of output bits of the ADC.

In this work, we modeled a simple digital PWM DC-DC buck converter, with bang-bang (BB) control, that is ADC with one bit ($m = 1$). It is not a practically-used scheme for switching power supply due to power inefficiency caused by a number of fast switching activities, but it can provide an understandable example to show how the proposed reachability analysis algorithm can model digitally-controlled analog circuits.

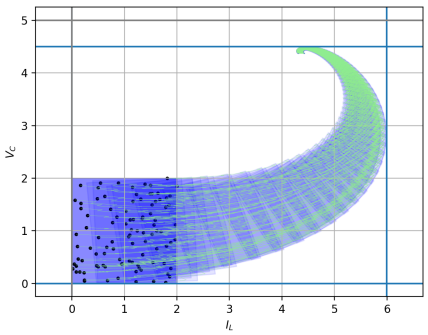
5.3.1 DC-DC Buck Converter with Digital Pulse-Width Modulation (DPWM) Control

The circuit under verification is a DC-DC buck converter circuit with the same topology in 5.3 and the circuit parameter as Table 5.1. In addition, the duty of the input $u(t)$ is controlled digitally by the polarity of the error level $e(t) = V_c(t) - V_{REF}$. The controller measures the output voltage $V_c(t)$ at the end of the switching pulse $u(t)$ period and compares the measured value with the desired reference voltage $V_{REF}(t)$. when the measured output voltage $V_C(t)$ is larger than the voltage V_{REF}, the controller increases the digital duty value encoding the real duty-cycle $D(t)$ by $+1$ before the next cycle begins.

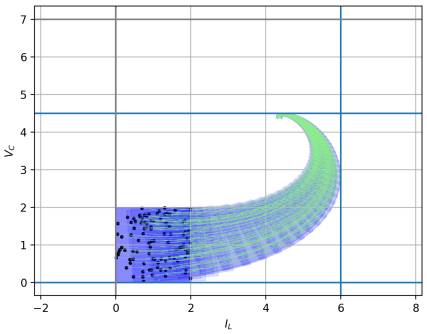
Fig. 5.11 shows the computed reachable sets $\mathcal{R}(t)$ varying the output reference



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

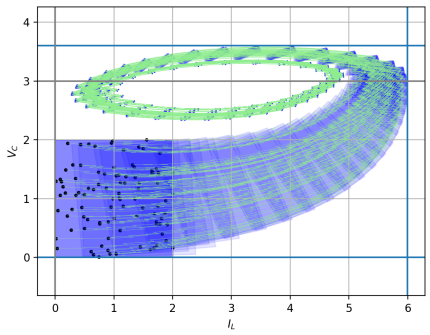


(b) $V_{ref} = 5V$

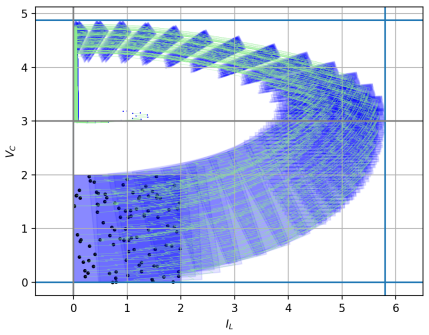


(c) $V_{ref} = 7V$

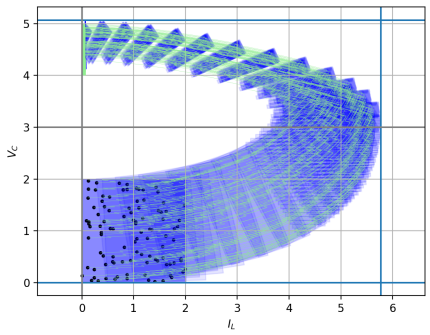
Figure 5.11: Reachable sets varying V_{REF} for PWM DC-DC buck converter.



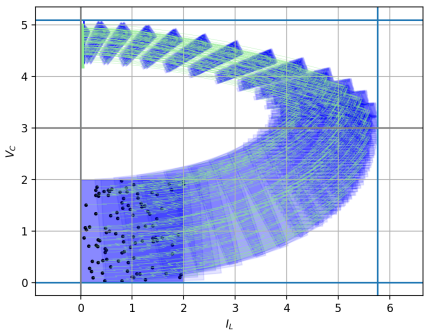
(a) $R_L = 1\,\Omega$



(b) $R_L = 10\,\Omega$



(c) $R_L = 50\,\Omega$

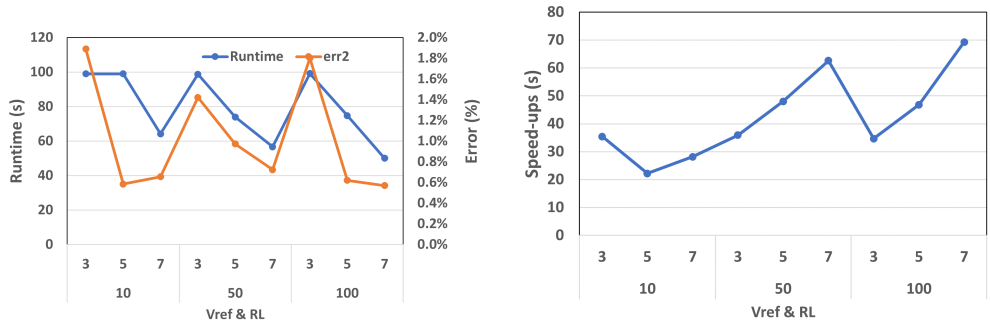


(d) $R_L = 100\,\Omega$

Figure 5.12: Reachable sets varying V_{REF} for PWM DC-DC buck converter.

voltage $V_{REF} = 3, 5, 7\,\mathrm{V}$ in the state space of the continuous circuit state $x(t) = (I_L(t), V_C(t))$, compared to the equivalent Monte-Carlo (MC) simulation trajectories. Each reachable set accurately encloses the state trajectories from initial samples randomly selected in the initial set $I_L = [0, 2]\mathrm{A}$ and $V_C = [0, 2]\mathrm{V}$ while converging to different reference voltage $V_{REF} = 3/5/7V$ set.

In the same way, the reachable set while varying the output load resistance in the range of 1,10,100,1000 Ω are shown in Fig. 5.12. The default reference voltage is set to 3 V for all cases. The case of $R_L = 1\,\Omega$ converges to reference voltage 3 V without reaching the DCM operation at the guard line $I_L = 0\,\mathrm{A}$. On the contrary, the other



(a) Runtime and error

(b) Speed-ups vs. SpaceEx

Figure 5.13: Performance comparison for PWM DC-DC buck converter.

cases reach the guard line and the voltage V_C slowly gets smaller due to discharging with low inductor current I_L, incurring massive guard intersection computation and increasing corresponding runtimes. The results also show good agreement with the MC simulation results in green.

Fig. 5.13 summarizes the runtime of the proposed methods and the relative error of safety bound measured by err_2 in (4.24) compared to MC. The speed-up improvements compared to the SpaceEx using 'STC' algorithm and 'oct' shapes are compared in Fig. 5.13 (b), showing the speed-up ranging 22–69 $\times$, while the accuracy is maintained below 2 $\%$ compared to MC simulation references. The runtime ranges 50–98 s. This is quite large than we expected. The cause of the slow runtime in the closed-loop feedback system is that consistently generated a new flow of the reachable set and also exponentially increases the required runtime until the predefined number of cycles (i.e. equivalent to the bound of time). This also suggests further research topics. Clustering several sets with the same discrete states into a new set can be a remedy for this. Otherwise, random walk or SMT algorithm can also efficiently reduce the exponential increase of computation due to combinations of possible digital states and continuous sets.

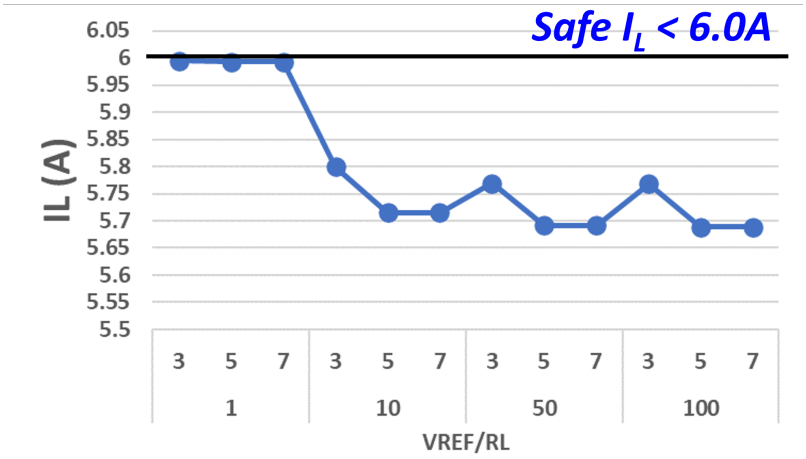
5.4 Verifying Safe Operating Area (SOA)

Finally, we verified the safe operation of the circuits using the safe operating area (SOA) specification under parameter variation of V_{REF} and the load resistance R_L for the cases in the system of Fig. 5.10. In each case, the operating regions are computed using the proposed reachability analysis method.

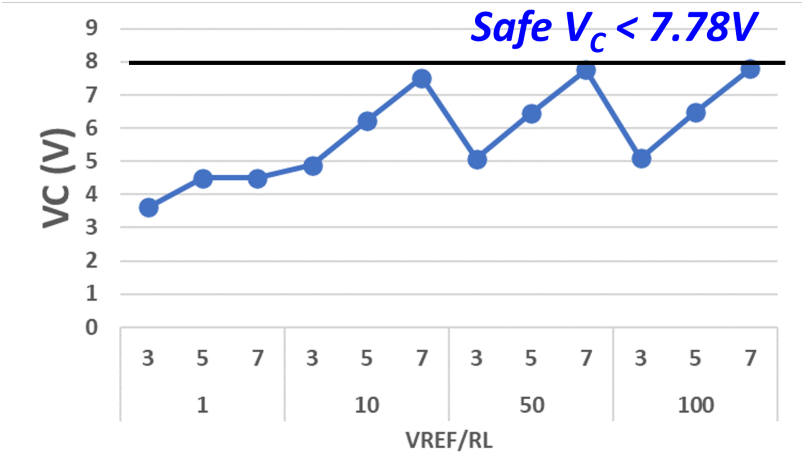
Fig. 5.14 summarizes the bounds of operating region in terms of voltage V_C and current I_L while varying circuit parameters R_L=1,10,50,100 Ω and V_{REF}=3,5,7 V. The overall bound for the inductor current I_L is computed by 6.0 A and the capacitor voltage V_C is given by 7.78 V. So, we can conclude the operating region of the circuit with the parameters are in $I_L = [0, 6.0]A$ and $V_C = [0, 7.78]V$.

Fig. 5.15 shows the comparison between the SOA specification for the 90-nm process and the compute operating regions with bounded initial conditions. For the cases of load resistance $R_L = 1\,\Omega$, the design exceeds the SOA specifications by reaching the current limit for the switch MOSFET in the test cases. Fig. 5.14 shows the bound values for current and voltages for all test cases with varying parameters. As expected, increasing load resistance R_L resulted in the decrease of the operating current of the buck converter, agreeing with the behavior with simulation data. In the case of low R_L, i.e. heavy load, current through the inductor increases and poses a high risk of circuit failure. In addition, increasing the reference voltages increases the risk that the capacitor might be damaged due to large transient over-voltage while charging the load at the initial charging period.

However, the proposed method still lacks the capability to formally verify with parameter variation caused process variation in the real world, resulting in incomplete verification. Further work to capture the parameter variation should be done in the future.

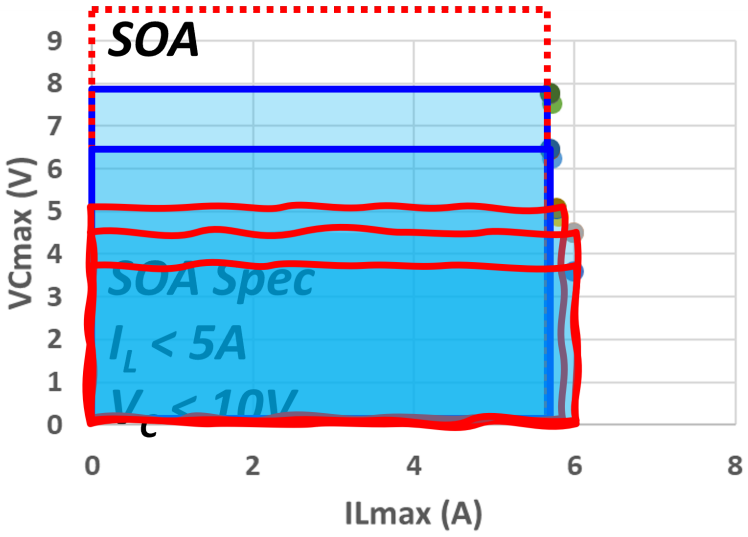


(a) Unsafe case

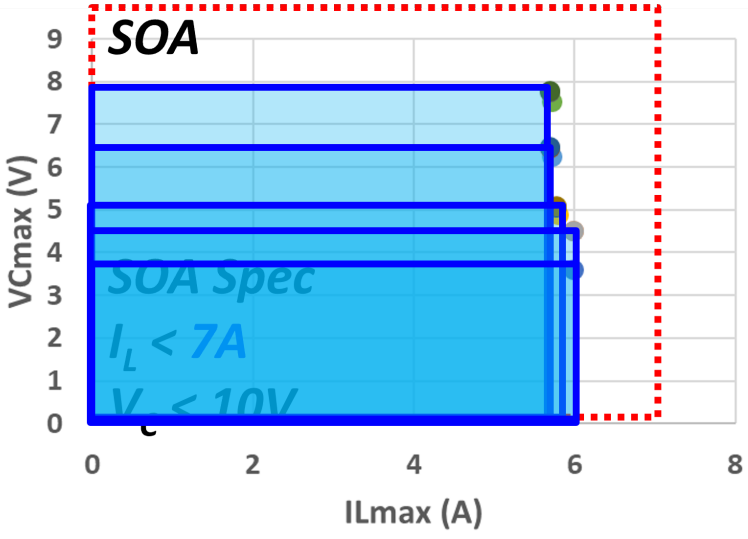


(b) Safe case

Figure 5.14: Verifying safe operating area of the circuit under verification.



(a) Unsafe case



(b) Safe case

Figure 5.15: Verifying safe operating area of the circuit under verification.

Chapter 6

CONCLUSION

This dissertation presents a formal verification methodology for the safety of AMS circuits using a reachability analysis algorithm. The proposed methodology leverages a hybrid system modeling technique that discretizes a set of continuous state values of the circuit variables similar to the boolean representation of the digital formal technique. Combined with a novel trajectory form of reachable set representation, it enables a fast, scalable, and accurate reachability analysis. In addition, efficient piecewise linear approximation of the nonlinearity of semiconductor devices, the basic building block of AMS circuits, such as MOSFETs and diodes, the proposed method can be applied to any type of AMS circuits.

The presented methodology was successfully demonstrated in verifying the safe operation of the switching power supply circuits with open/closed loop pulse width modulation control schemes. With varying circuit parameters and the initial conditions, the resulting operating range of the devices changes, and the proposed method can accurately detect its bounds compared to Monte-Carlo simulations with over $100\times$ speed-ups compared to the state-of-the-art reachability analysis algorithm.

The accuracy of the proposed RA algorithm was compared to the MC simulations and remained small, less than $2\,\%$. One might worry that even this small error can rarely lead to failures. However, the operating region computed by RA is a conserva-

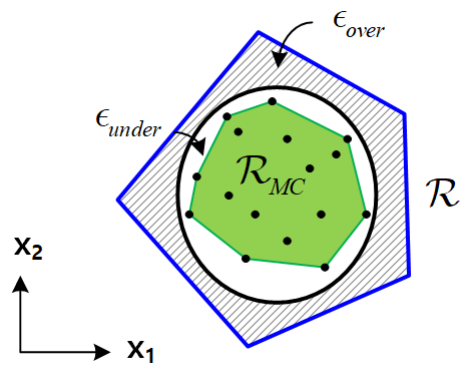
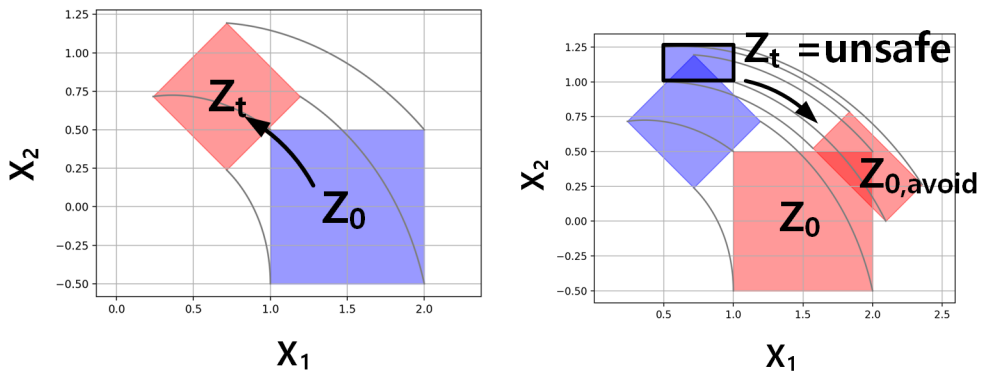


Figure 6.1: Conservative error compared to simulations.

tive approximation; that is, if this over-estimated operating region of the circuit never reaches unsafe regions, neither does the exact one. Fig. 6.1 illustrates the resulting error compared to the theoretical exact reachable set indicated as the circle, which is the sum of the over-approximation error ϵ_{over} (shaded) of the reachable set $\mathcal{R}$ and the under-approximation error ϵ_{under} of the set $\mathcal{R}_{MC}$ of states obtained from MC simulations. While the error of MC simulations ϵ_{under} contributes false-negatives that can cause real failures, the error of the proposed reachable set ϵ_{over} does not, only resulting in false-positives. However, it is desirable to reduce this over-approximation error since it leads to higher design costs.

The main contributions of the presented methodology are two aspects. One is to avoid the dependency of space and time discretization to compute the time evolution of the set of states, reducing the scalability of the method. This idea is implemented with a zonotope set in this study, but it can be extended to any type of set representations based on linear algebra such as polyhedra or other representations, which need to be further researched. The other is the efficient computation of guard intersection which is the most significant bottleneck of every hybrid system reachability analysis. In this study, based on the idea that every state is started from the guard intersection, we iteratively generate the next initial set using the linear superposition of the basis



(a) Forward RA

(b) Backward RA

Figure 6.2: Backward reachability analysis extending the proposed trajectory-form.

vectors spanning the guard set. In this way, we can easily compute the guard intersection and iterative reachable set computation from the intersection, in an accurate and fast way. However, the remaining concern is that the way to approximate the intersection on the guard hyperplane can also contribute to additional over-approximation in each iteration. This still remains an open problem requiring further research.

In addition, this work can be further extended to compute backward reachability that computes the set of initial conditions of which trajectories can reach the specified target region. From the backward reachable set computed from the given unsafe target region, one can avoid the risky initial conditions that can lead the circuit to a potentially unsafe operating region in advance without computing a reachable set for every possible initial condition. Computing the backward reachable set in the proposed trajectory-based method can be simply done by exploiting the time reversal property of Laplace analysis. That is, if a forward reachable set of trajectory is $X(s) \overset{\mathcal{L}}{\leftrightarrow} x(t)$, then the backward reachable set is just $X(-s) \overset{\mathcal{L}}{\leftrightarrow} x(-t)$. Replacing s to $-s$ in (3.12), the transfer function for backward computation is given by $H(-s) = (sI + A)^{-1}$ and the input term $U(-s)$ of time-reversed input $u(-t)$ can be obtained from the basis functions. Fig. 6.2 shows the backward reachable set using the idea. As shown in Fig.

6.2 (b), one can exclude the unsafe initial conditions that can reach the unsafe region $\mathcal{Z}_t$. Recognizing this analysis is simple linear system computation with only modifying the sign of the system matrix, extending this backward trajectory analysis to the hybrid system can be done in the same manner as Chapter 3.

Chapter 7

APPENDIX

7.1 Code Implementation

7.1.1 Example: Trajectory Form Reachable Set

```
1  import numpy as np
2  n=2 % system dimension
3  A=np.array([[0,-1],[1,0]], dtype=float)
4  E=np.identity(n,dtype=float)
5  U=[xmulan.xreal(value=[0,0,0,0,1]) for i in range(n)]
6  c = np.array([1.5,0],dtype=float)
7  G = [ np.array([0.5,0],dtype=float) ]
8  Z0 = [c,G] % initial zonotope set
9  % Trajectory Form Reachable Set
10 Ztrj = get_xreal_zonotope(E,A,Z0,U)
```

Listing 7.1: Simple code example for trajectory form reachable set.

```
1  %Zonotope Trajectory Form
2  Ztrj= [
3  [xreal(((0+1j,0.75+0j,1)) @0.000s), xreal(((0+1j,0+0.75j,1))
       @0.000s)],
```

```
4 [[xreal(((0+1j,0.25+0j,1)) @0.000s), xreal(((0+1j,0+0.25j,1))
      @0.000s)]]
5 ]
6 % Bounds for each axis also represented by Xreal Trajectory
7 bound(Ztrj,axis=0)= xreal(((0+1j,0.5+0j,1)) @0.000s) xreal
      (((0+1j,1+0j,1)) @0.000s)
8 bound(Ztrj,axis=1)= xreal(((0+1j,0+0.5j,1)) @0.000s) xreal
      (((0+1j,0+1j,1)) @0.000s)
```

Listing 7.2: Zonotope Trajectory Form Output.

Bibliography

[1] E. M. Clarke, *et al.*, *Handbook of Model Checking*. Springer, 2018.

[2] J. O'Leary, "Formal Verification in Intel CPU Design," *Proc. IEEE Int. Conf. Form. Methods Model. Co-Design*, p. 152, 2004.

[3] E. Seligman, *et al.*, *Formal Verification : An Essential Toolkit for Modern VLSI Design*. Elsevier, 2015.

[4] L. Goldgeisser, *et al.*, "Modeling Safe Operating Area in Hardware Description Languages," in *Proc. Des. Autom. Conf.*, Jul. 2007, pp. 377–382.

[5] E. M. Clarke, "Model Checking," in *Proc. Int'l Conf. Found. Software Technol. Theor. Comput. Sci.*, Dec. 1997, pp. 54–56.

[6] M. H. Zaki, *et al.*, "Formal Verification of Analog and Mixed Signal Designs: A Survey," *Microelectron. J.*, pp. 1395–1404, Dec. 2008.

[7] K. D. Jones, *et al.*, "Some Real World Problems in the Analog and Mixed Signal Domains," 2018.

[8] R. Alur, *et al.*, "The Algorithmic Analysis of Hybrid Systems," *Theor. Comput. Sci.*, pp. 3–34, Feb. 1995.

[9] T. A. Henzinger, *et al.*, "HYTECH: A Model Checker for Hybrid Systems," *Int'l J. Softw. Tools Technol. Transf.*, pp. 110–122, Dec. 1997.

[10] T. A. Henzinger, "The Theory of Hybrid Automata," *Verif. Digit. Hybrid Syst.*, pp. 265–292, Jul. 2000.

[11] W. Hartong, *et al.*, "Model Checking Algorithms for Analog Verification," in *Proc. IEEE/ACM Des. Autom. Conf.*, Jun. 2002, pp. 542–547.

[12] S. Gupta, *et al.*, "Towards formal verification of analog designs," in *Proc. IEEE/ACM Int'l Conf. Comput. Aided Des.*, Nov. 2004, pp. 210–217.

[13] T. Dang, *et al.*, "Verification of Analog and Mixed-Signal Circuits Using Hybrid System Techniques," in *Proc. Int'l Conf. Form. Methods Comput. Des.*, Nov. 2004, pp. 21–36.

[14] A. Girard, "Reachability of Uncertain Linear Systems Using Zonotopes," in *Proc. Int'l Work. Hybrid Syst. Comput. Control*, Apr. 2005, pp. 291–305.

[15] A. Girard and C. Le Guernic, "Zonotope/Hyperplane Intersection for Hybrid Systems Reachability Analysis," in *Proc. Int'l Work. Hybrid Syst. Comput. Control*, Apr. 2008, pp. 215–228.

[16] C. Le Guernic and A. Girard, "Reachability Analysis of Hybrid Systems Using Support Functions," in *Proc. Int'l Conf. Comput. Aided Verif.*, Jun. 2009, pp. 540–554.

[17] G. Frehse, *et al.*, "SpaceEx: Scalable Verification of Hybrid Systems," in *Proc. Int'l Conf. Comput. Aided Verif.*, Jul. 2011, pp. 379–395.

[18] A. A. Kurzhanskiy and P. Varaiya, "Ellipsoidal Techniques for Reachability Analysis of Discrete-Time Linear Systems," *IEEE Trans. Automat. Contr.*, pp. 26–38, Jan. 2007.

[19] H. Lin, *et al.*, "Verification of Digitally-Intensive Analog Circuits Via Kernel Ridge Regression and Hybrid Reachability Analysis," in *Proc. IEEE/ACM Des. Autom. Conf.*, Jun. 2013, pp. 1–6.

[20] S. Little, *et al.*, "Verification of Analog/Mixed-Signal Circuits Using Labeled Hybrid Petri Nets," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 30, no. 4, pp. 617–630, 2011.

[21] M. Althoff, *et al.*, "Formal Verification of Phase Locked Loops Using Reachability Analysis and Continuization," in *Proc. IEEE/ACM Int'l Conf. Comput. Aided Des.*, Nov. 2011, pp. 659–666.

[22] W. Denman, *et al.*, "Formal Verification of Analog Designs Using MetiTarski," in *Proc. Int'l Conf. Form. Methods Comput. Des.*, Nov. 2009, pp. 93–100.

[23] S. Steinhorst and L. Hedrich, "Trajectory-Directed Discrete State Space Modeling for Formal Verification of Nonlinear Analog Circuits," in *Proc. Int'l Conf. Comput. Des.*, Nov. 2012, pp. 202–209.

[24] O. A. Beg, *et al.*, "Model Validation of PWM DC-DC Converters," *IEEE Trans. Ind. Electron.*, pp. 7049–7059, Mar. 2017.

[25] O. Stursberg and B. H. Krogh, "Efficient Representation and Computation of Reachable Sets for Hybrid Systems," in *Proc. Int'l Conf. Hybrid Systems: Comput. Control*. Berlin, Heidelberg: Springer-Verlag, 2003, p. 482–497.

[26] J. Kim, "New Opportunities for Analog Formal Verification with Piecewise-Linear Device Modeling," in *Proc. Front. Analog CAD*, 2018.

[27] H. S. L. Lee, *et al.*, "Automated Generation of Hybrid System Models for Reachability Analysis of Nonlinear Analog Circuits," in *Proc. Asia South Pacific Des. Autom. Conf.*, Jan. 2015, pp. 725–730.

[28] J.-E. Jang, *et al.*, "An Event-Driven Simulation Methodology for Integrated Switching Power Supplies in SystemVerilog," in *Proc. ACM/IEEE Des. Autom. Conf.*, May 2013, pp. 1–7.

[29] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, 2010, vol. 25, no. 3.

[30] S. Kim, *et al.*, "Safety Verification of AMS Circuits with Piecewise-Linear System Reachability Analysis," in *Proc. Int'l SoC Des. Conf.*, Oct. 2021, pp. 203–206.

[31] S. Kim, and J. Kim, "Reachability Analysis for Nonlinear Analog/Mixed-Signal Circuits with Trajectory-Based Reachable Sets," *IEEE Access*, pp. 1–1, 2023.

[32] A. Girard and C. Le Guernic, "Efficient Reachability Analysis for Linear Systems Using Support Functions," in *Proc. Int'l Fed. Autom. Control*, Jul. 2008, pp. 8966–8971.

초 록

본 논문은 아날로그 혼성 회로의 안전성을 공식적으로 검증하기 위한 방법론을 제안한다. 최근 자동차에 많은 집적 회로를 통합하면서, 시스템이 안전함을 보장하는 것은 안전이 중요한 시스템을 개발하는 데 중요한 문제를 야기하고 있다. 아날로그 회로에 포함된 모스펫 및 다이오드와 같은 관련 반도체 장치의 작동 범위는 프로세스 설계 키트 또는 데이터시트에 명시된 안전 동작 영역에 한해 제한되어야 하지만, 업계의 스파이스 시뮬레이션에 의존하는 기존 검증 절차는 가능한 모든 동작을 고려하지 못하는 문제가 있다. 반면, 공식적인 검증 알고리듬, 특히 도달 가능성 분석은 지정된 초기 조건으로부터 모든 도달 가능한 상태를 계산하고 도달 가능한 상태 집합이 안전하지 않은 상태 집합과 교차하는지 여부를 발견함으로써 시스템이 안전하다는 것을 증명할 수 있다. 하지만, 도달 가능한 세트를 계산하는 것은 연속적인 상태값을 여러 개의 이산 상태값으로 분할하는 것을 필요로하기 때문에, 고주파에서 동적으로 빠르게 작동하는 고차원 시스템에서 도달 가능한 세트를 계산하는 것은 너무 오래 걸리며, 도달 가능성 알고리즘을 이용해 실제 회로를 검증하기 어렵게 한다. 본 논문은 도달 가능한 분석을 아날로그 혼성 회로에 적용하여 동작 가능 범위를 구하는 안전 영역 검증 방법론을 다루며, 이와 함께 도달 가능한 세트를 효율적으로 빠르고 정확하게 계산하는 방법을 제안한다.

제안된 도달 가능성 분석 알고리듬은 확장 가능한 기하학적 집합 표현을 사용하는 두 가지 주요 아이디어인 조노토프와 아날로그 신호를 궤적을 표현하는 닫힌 함수 형태로 표현하는 알고리듬을 결합한 조노토프의 궤적 함수 형태를 제안하여, 도달 가능한 아날로그 혼합 신호 회로 세트를 빠르고 정확한 방식으로 계산할 수 있다. 또한, 포함된 장치의 작동 모드에 따라 회로의 상태 공간만 분할하는 효율적

인 조각 선형 근사 기법을 사용하여 비선형 구성 요소를 포함하는 아날로그 회로도 분석할 수 있다. 하위 영역으로 구성된 조각 선형 시스템은 많은 계산 비용이 드는 기하학적 교차 연산을 추가로 요구하기 때문에 시간 이산화에 의존하는 기존 도달 가능성 알고리듬으로 더욱 계산하기 어려우며, 이것은 하위 영역에 대한 전환 횟수에 따라 런타임을 기하급수적으로 증가시켜 계산을 불가능하게 한다. 제안된 궤적 형태를 적용해 계산하는 것 또한 궤적 형태와 스위칭 경계 사이의 교차점이 동일한 형태를 유지할 수 없기 때문에 간단하지 않으며, 이전과 같이 원래 형태를 유지하기 위해 설정된 형태의 추가 분할이 필요로하게 된다. 본 논문에서는 교차 후 모든 상태가 경계면에서 시작된다는 것을 이용하여, 경계면 자체의 시간 진화를 계산함으로써, 런타임을 증가시키지 않고 제안된 궤적 형태로 빠르게 관련 계산이 수행될 수 있음을 보여주었다.

제안된 방법은 예시적인 전력 컨버터 회로를 검증하여 입증되었으며, 기존 도달 가능성 분석 알고리듬보다 107배 이상 빠른 속도를 구현하였다. 이는 동등한 몬테카를로 시뮬레이션과 비교하여 정확도를 유지하며, 상태 차원에 대한 계산 복잡도 또한 시스템 차원수의 제곱만큼 개선하였다. 마지막으로 전력 변환 회로의 다양한 회로 매개 변수를 위한 안전 동작 영역을 도출되고 일반적인 90나노미터 프로세스 기준과 비교하여 검증함으로써, 혼성신호 회로의 안전 동작작 사양을 검증하는 데 본 방법을 활용할 수 있음을 보였다.

주요어: Reachability analysis, formal verification, safety verification, analog mixed-signal circuits, DC-DC converters, Laplace s-domain analysis, XMODEL

학번: 2018-36370